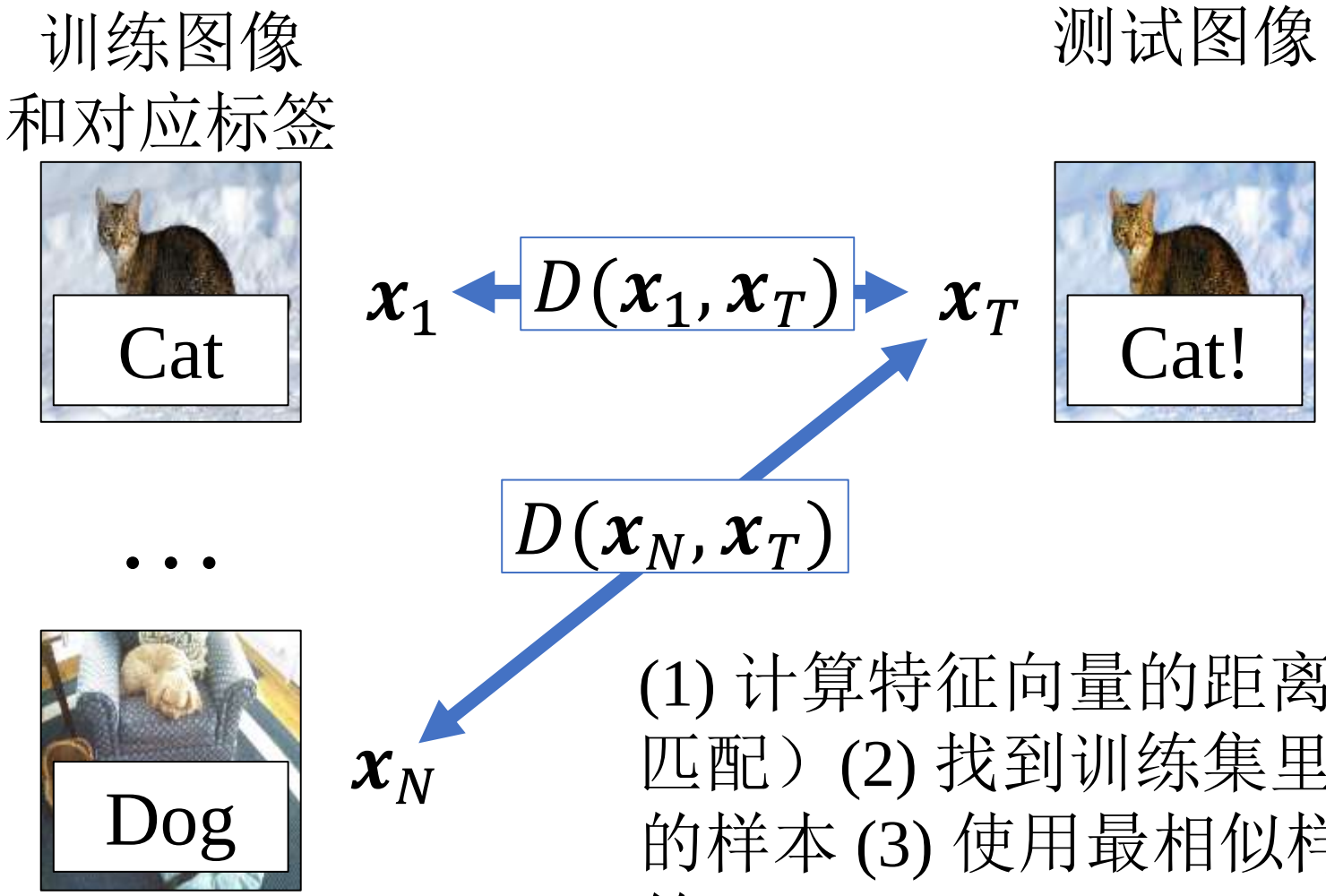
回顾——图像分类

分类最简单的形式：最近邻

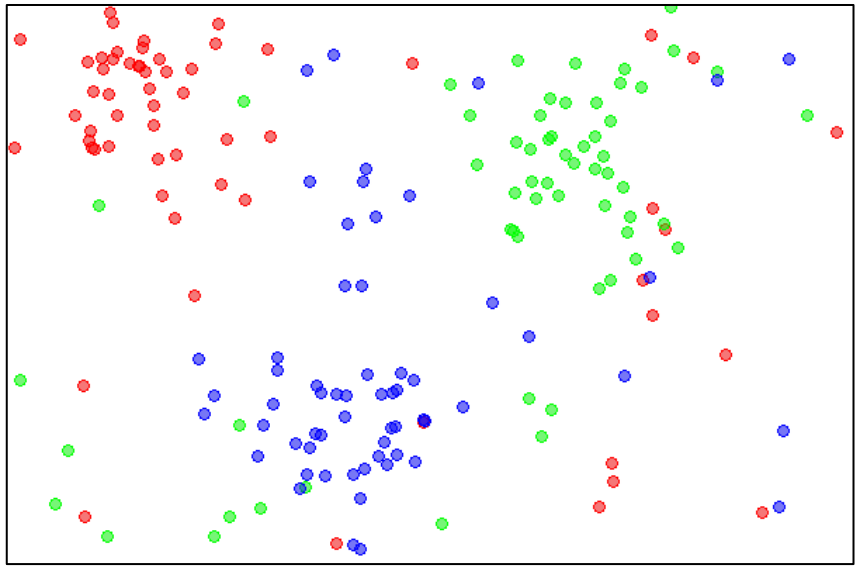


(1) 计算特征向量的距离（特征匹配）(2) 找到训练集里最相似的样本 (3) 使用最相似样本的标签.

K近邻

找到前K近邻样本，然后投票决定标签

2D Datapoints
(colors = labels)



2D Predictions
(colors = labels)

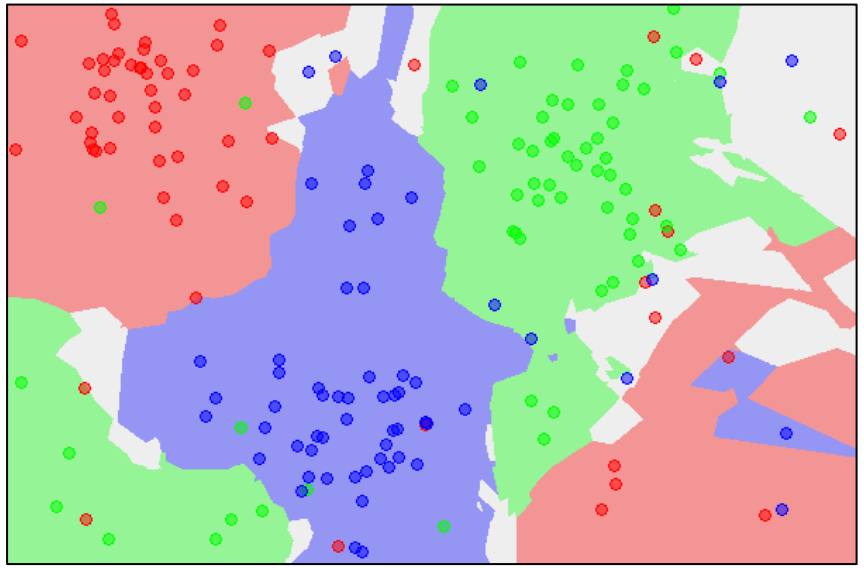


Diagram Credit: Wikipedia

线性模型

设定：三分类模型



模型：每个类别一个权重

$\boldsymbol{w}_0, \boldsymbol{w}_1, \boldsymbol{w}_2$

$\boldsymbol{w}_0^T\boldsymbol{x}$ big if cat

$\boldsymbol{w}_1^T\boldsymbol{x}$ big if dog

$\boldsymbol{w}_2^T\boldsymbol{x}$ big if hippo

全部参数: $\boldsymbol{W_{3xF}}$ where **x** is in R^F

可视化线性分类器

根据$\mathbf{w}^T\mathbf{x}$做决策. 如果很大则该样本属于这一类.

Diagram credit: Karpathy & Fei-Fei

解释：几何角度

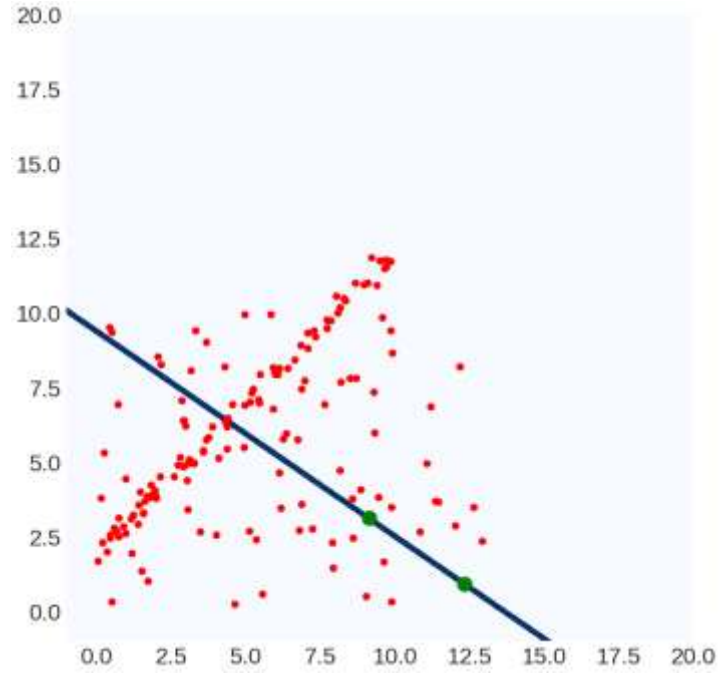
- 参数为每个类定义一个超平面:

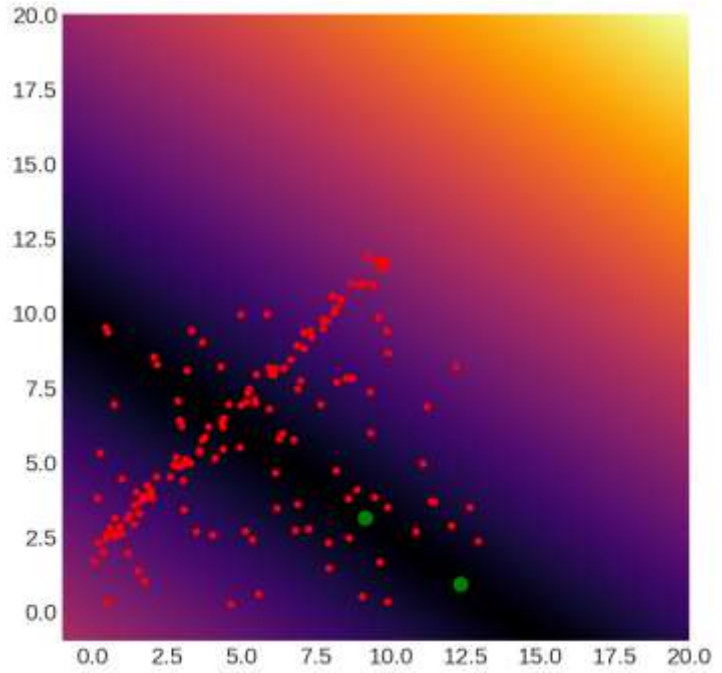
$$f(x_i, W, b) = Wx_i + b$$

- 我们可以将每个类的得分视为定义了一个与其距离成正比的分布，距离是指从对应的超平面到点的距离。

线性模型能表征什么？

- ax+by+z：
  - 描述了一个平面
  - 表示的值与到直线得到符号距离相等
- **线性模型是N维的超平面推广$\mathbf{w}^T\mathbf{x}$+b**



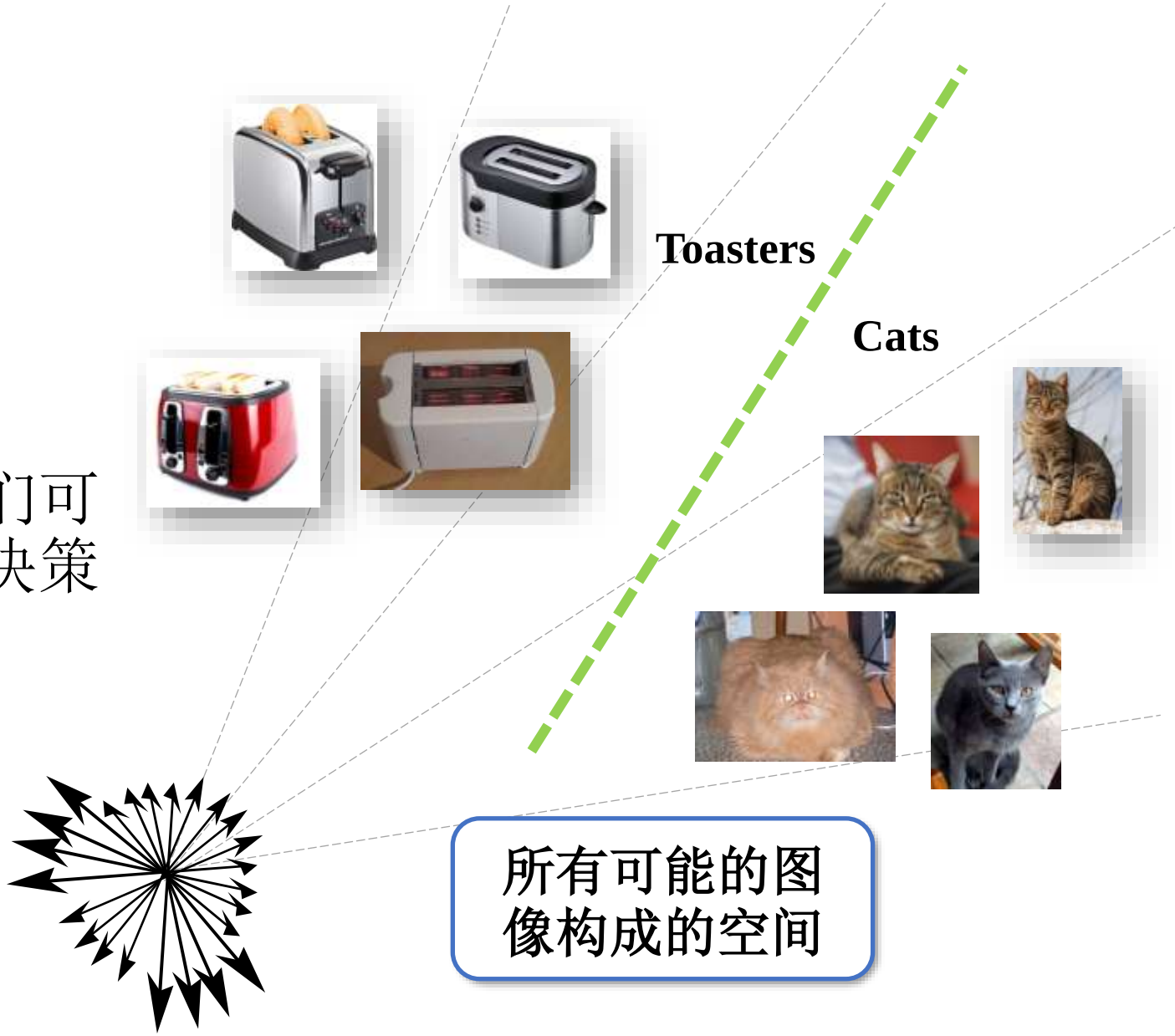


将图像视为高维向量

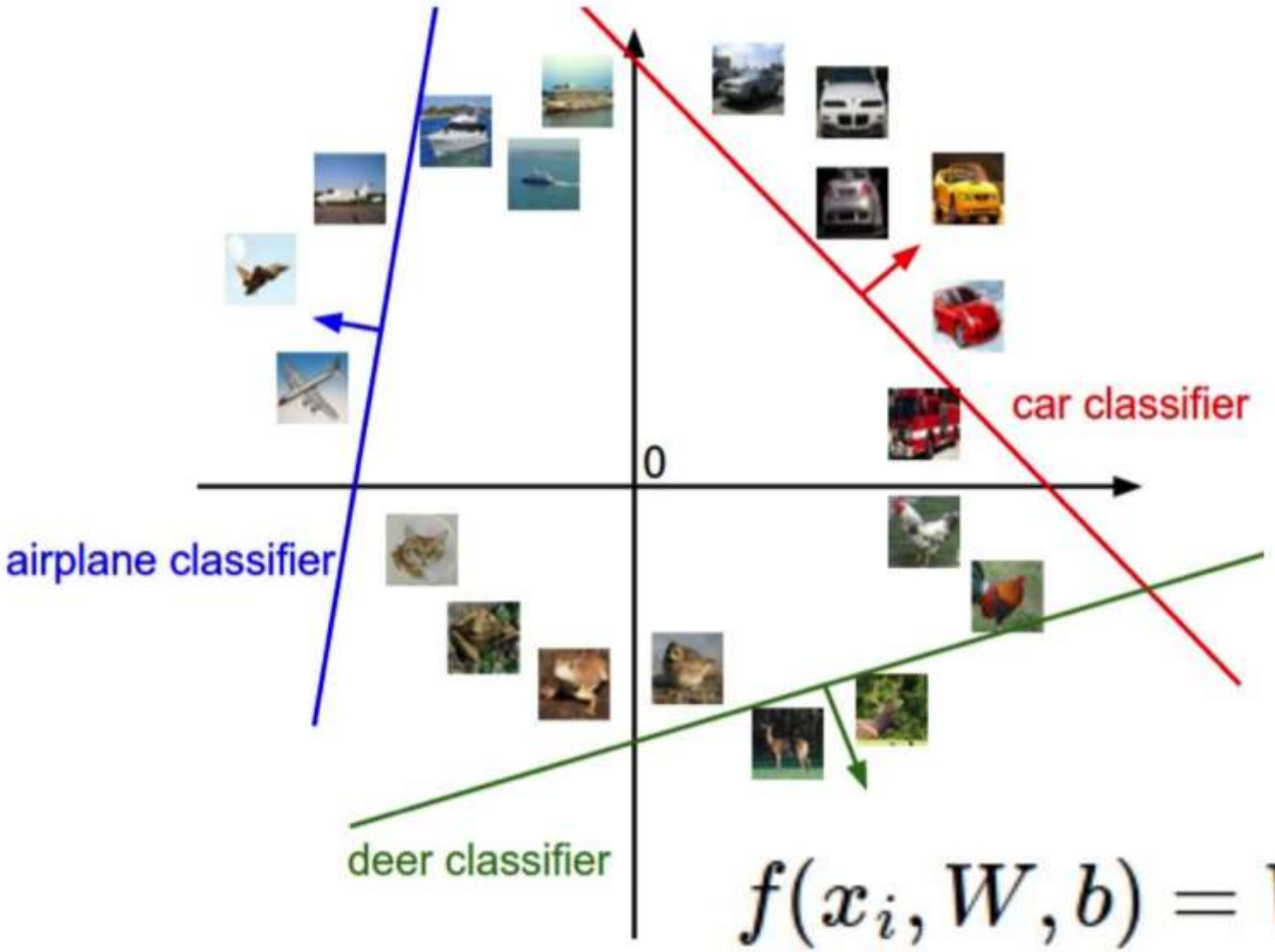
- **图像只是一组数字：**每幅图像都可以看作是一个由像素值组成的数字数组。对于灰度图像，这些数字通常是0到255之间的值。对于彩色图像，每个像素通常包含三个值（RGB）。
- **将图像转换为向量：**为了便于计算和处理，我们可以将这些数字堆叠成一个向量。例如，对于一个100x100的灰度图像，我们可以将它转换为一个10,000维的向量。
- **训练数据变为高维点集：**当我们有多幅图像时，每幅图像都可以转换为一个高维向量。因此，我们的训练数据集就可以看作是在高维空间中的一组点。

将图像视为高维向量

- 图像只是一组数字
- 将图像转换为向量
- 训练数据变为高维点集
- 将图像转换为高维向量后，我们可以尝试在这个高维空间中找到决策边界来分隔不同的类别。



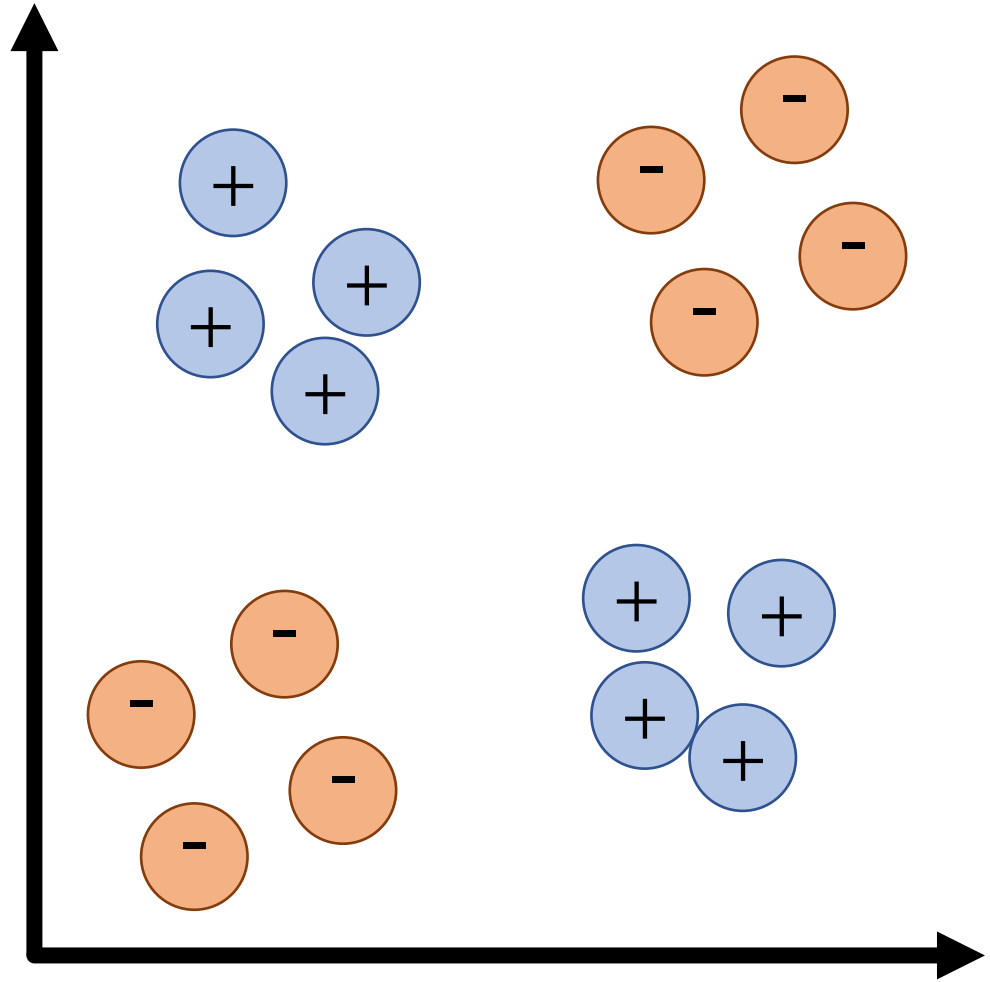
期望：图像线性可分



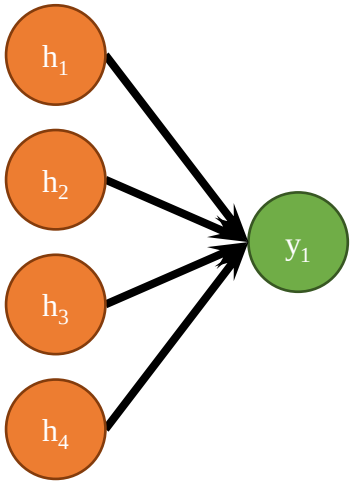
$$f(x_i, W, b) = Wx_i + b$$

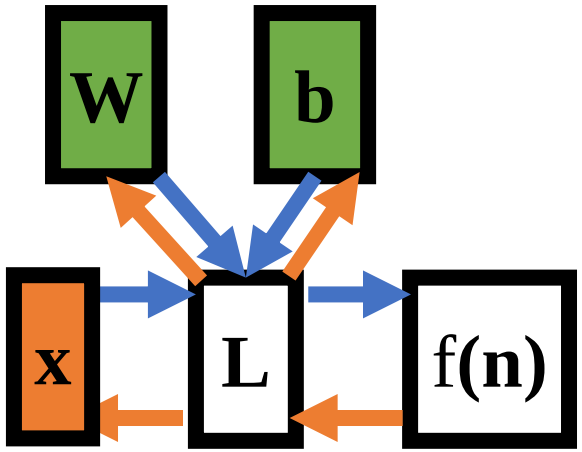
现实

- 数据并非线性可分



线性模型能表征什么？





$$L(\boldsymbol{n}) = \boldsymbol{W}\boldsymbol{n} + \boldsymbol{b}$$

线性模型怎么处理非线性问题？

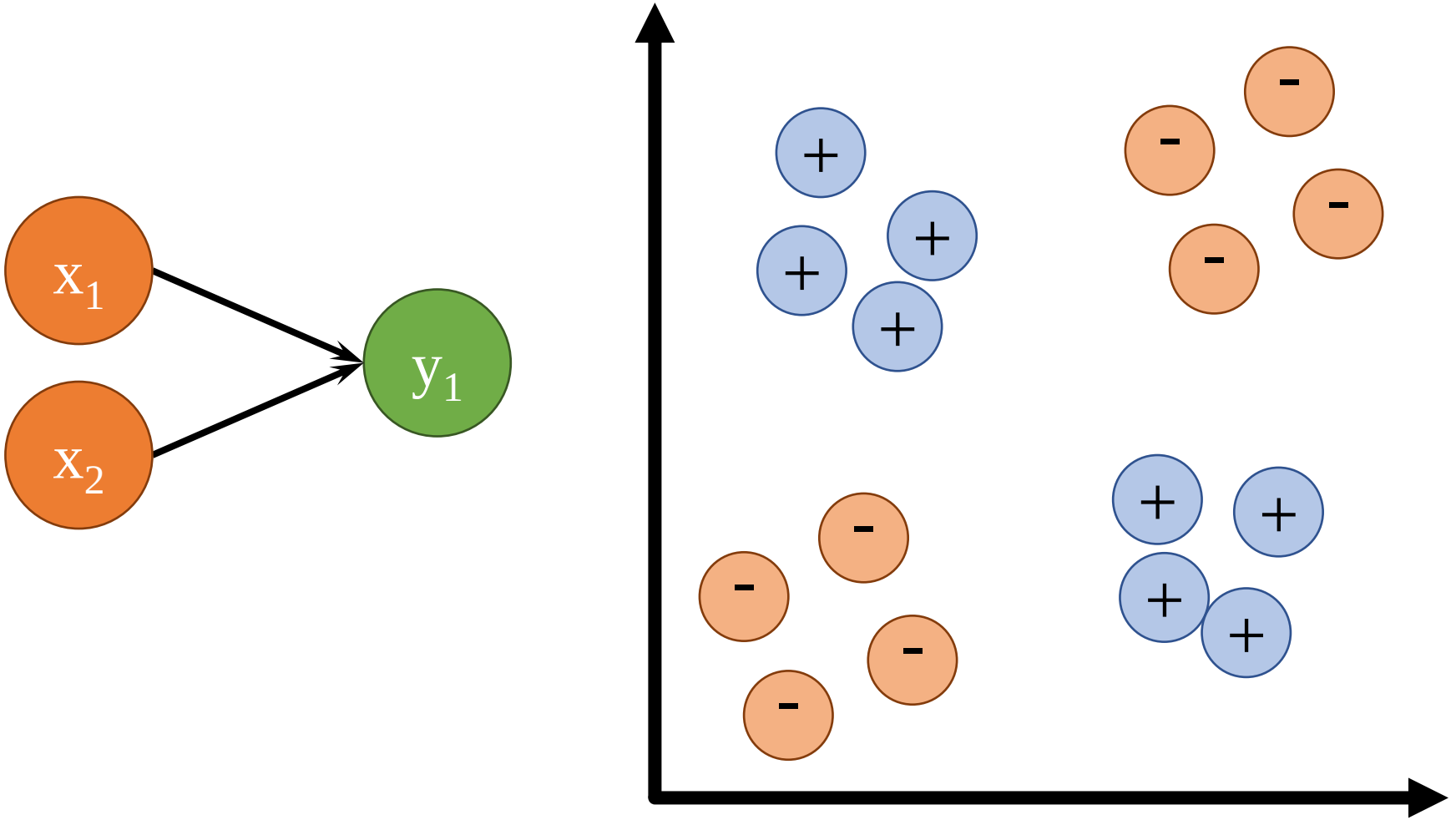
多项式回归: 给定 x, 预测 y

$$\begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} x_1^F & \cdots & x_1^2 & x_1 & 1 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ x_N^F & \cdots & x_N^2 & x_N & 1 \end{bmatrix} \begin{bmatrix} w_F \\ \vdots \\ w_2 \\ w_1 \\ w_0 \end{bmatrix}$$

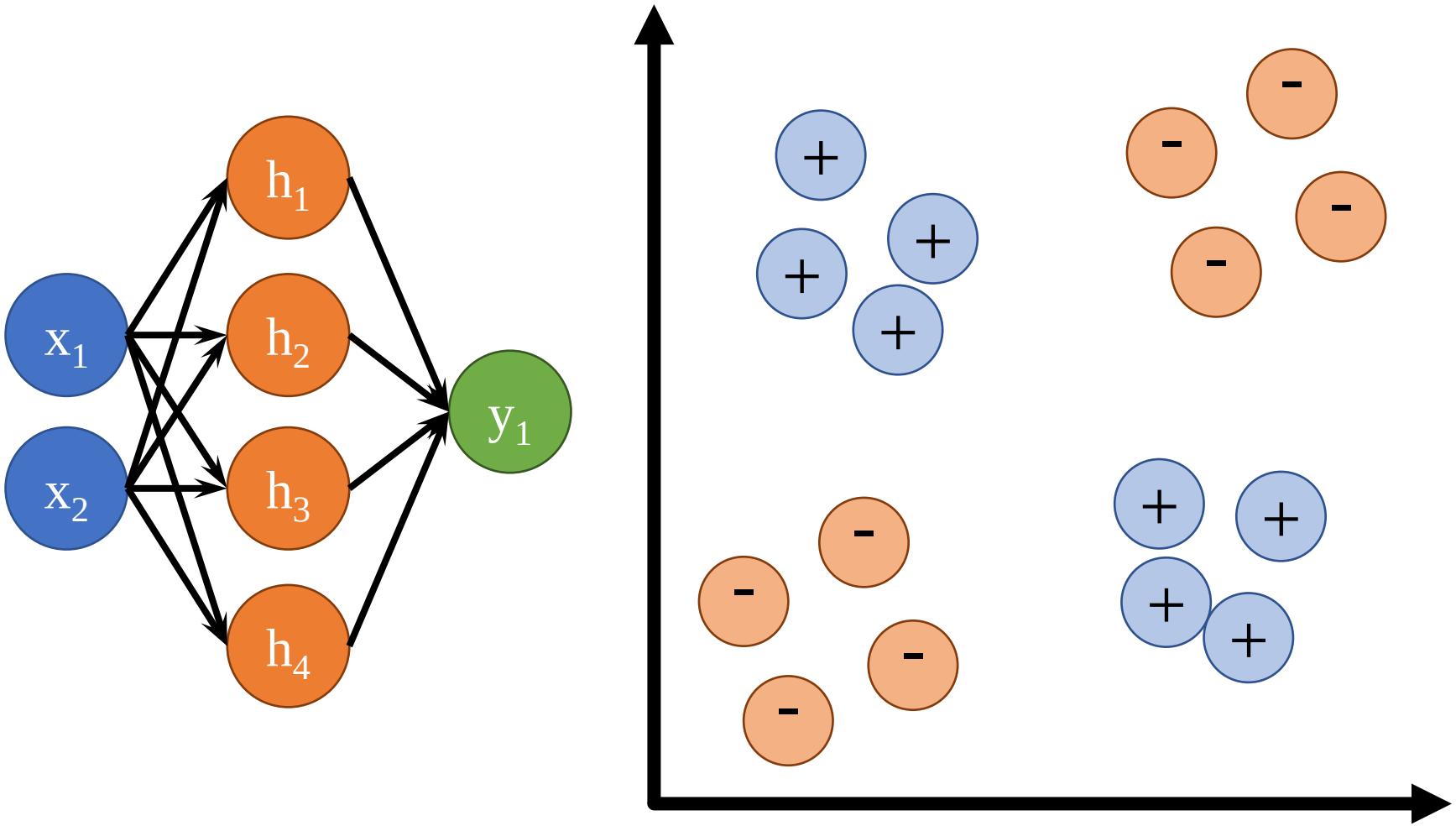
我们有一个输入矩阵 **X**，其中包含了各种多项式次数的数据（例如，$\boldsymbol{x}, x^2, x^3$ 等）。这样，模型可以选择最佳的多项式次数来拟合数据。

权重**W**: 每个多项式度数都有一个相应的权重,这些权重决定了每个多项式度数在模型中的重要性。

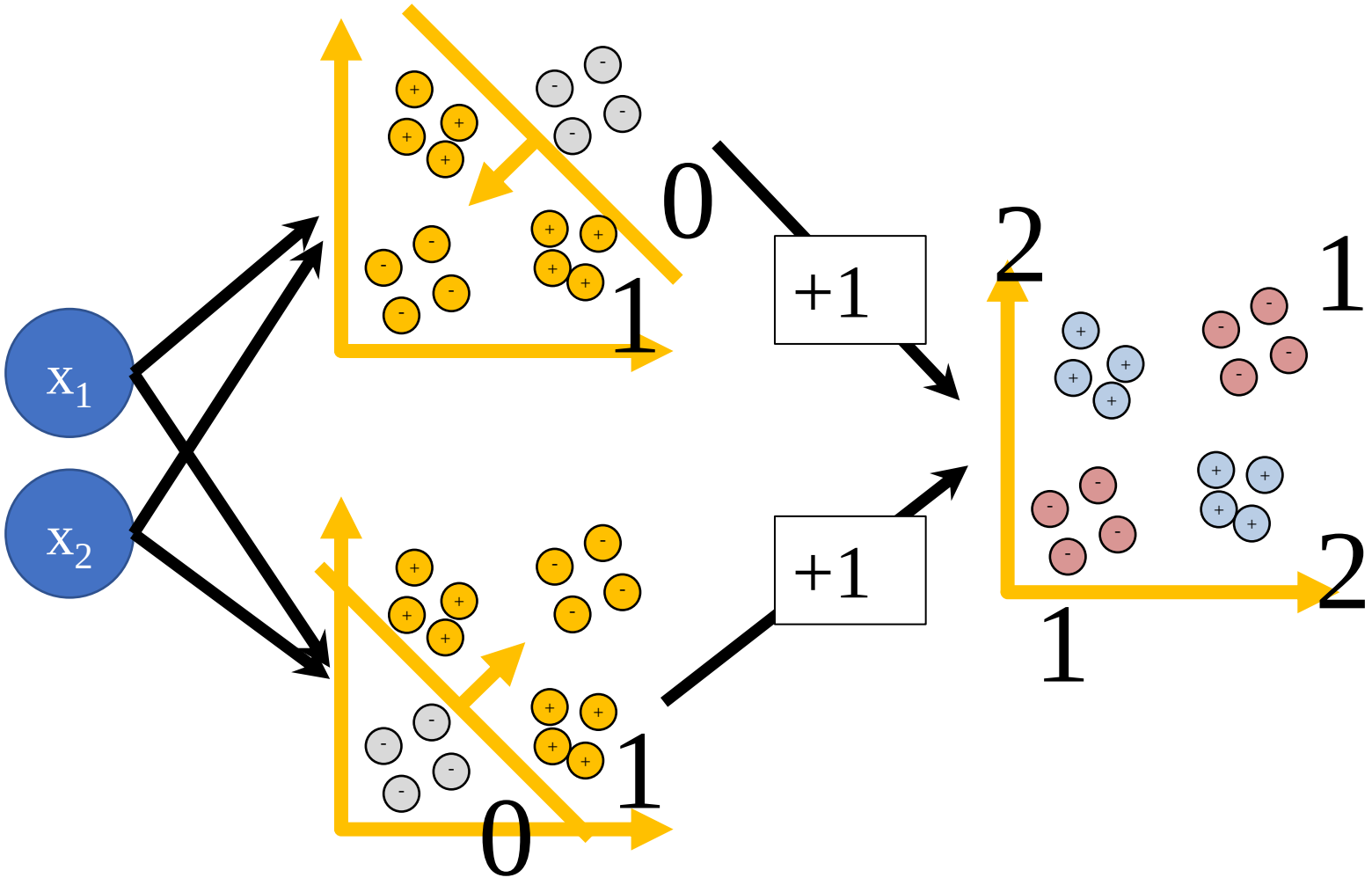
怎么训练一个线性模型来解决这个问题？



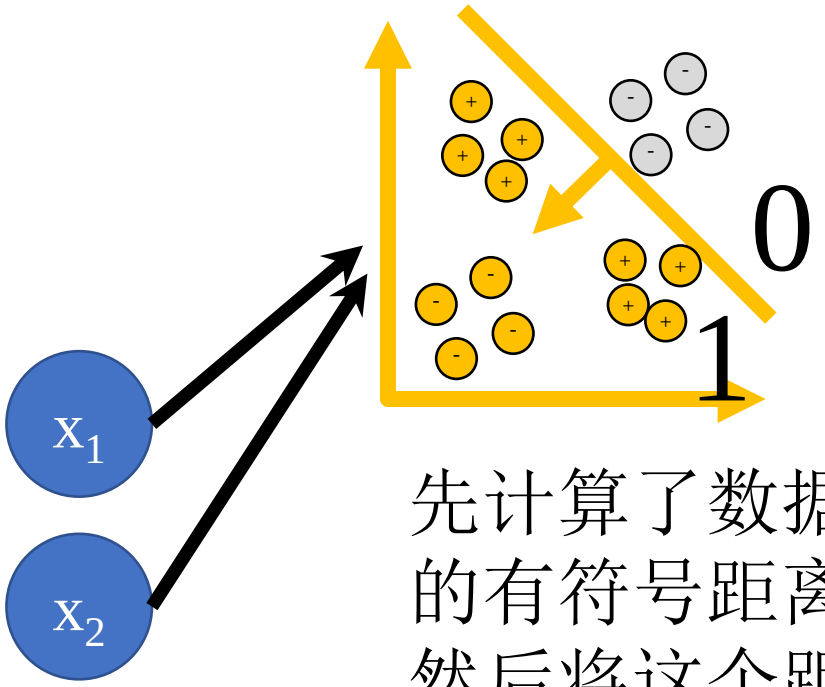
让我们把模型变得复杂一点



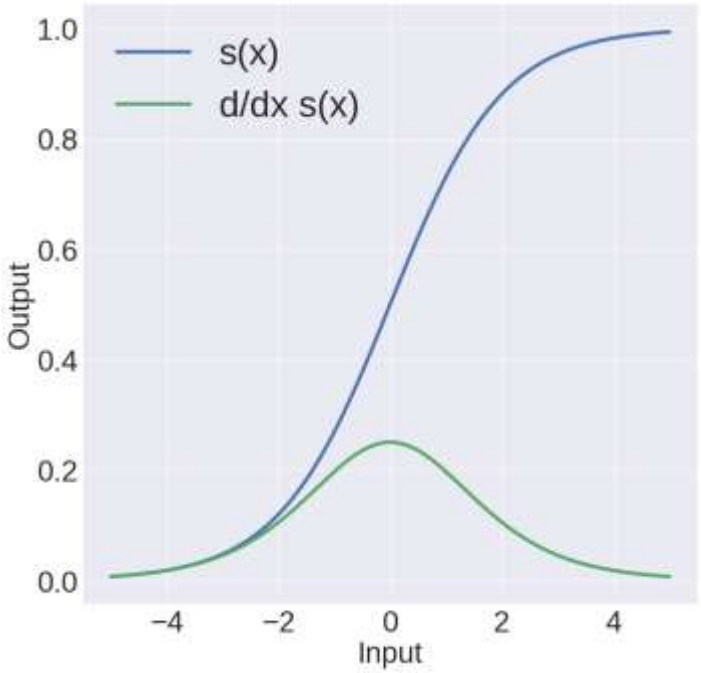
方案1



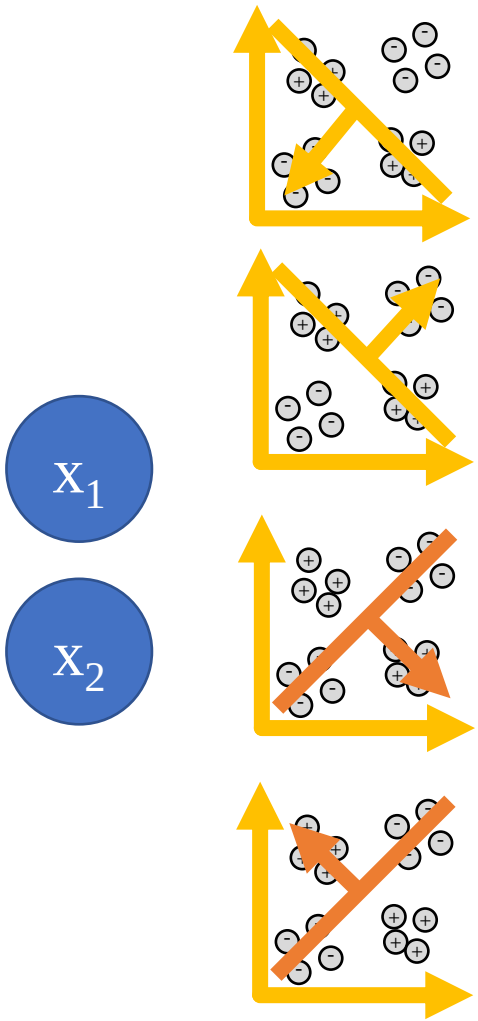
方案1



先计算了数据点的有符号距离，然后将这个距离传递给sigmoid激活函数



方案2



x_1

x_2

$\max(w_1^T x+b,0)$

$\max(-w_1^T x+b,0)$

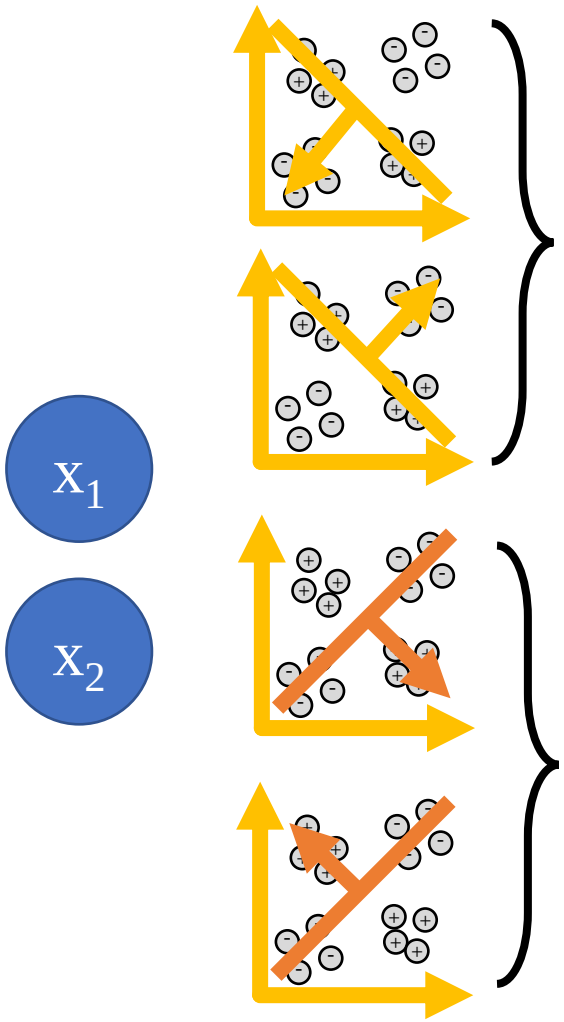
$\max(w_1^T x+b,0)+\max(-w_1^T x+b,0)=$ 到 w_1 的距离

$\max(w_2^T x+b,0)$

$\max(-w_2^T x+b,0)$

$\max(w_2^T x+b,0)+\max(-w_2^T x+b,0)=$ 到 w_2 的距离

方案2



到 w_1 的距离

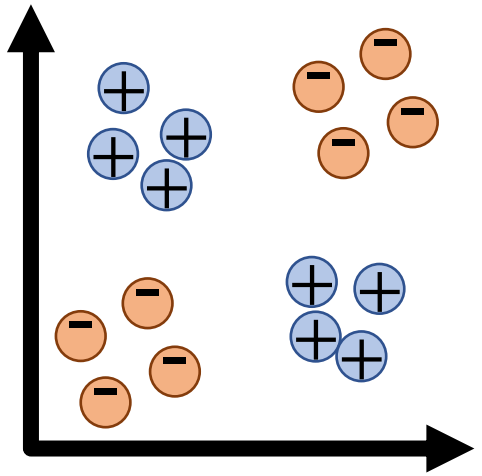
到 w_2 的距离

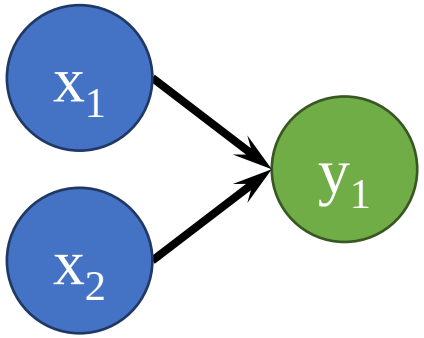
下一层计算:

w_1 距离 - w_2 距离 > 0

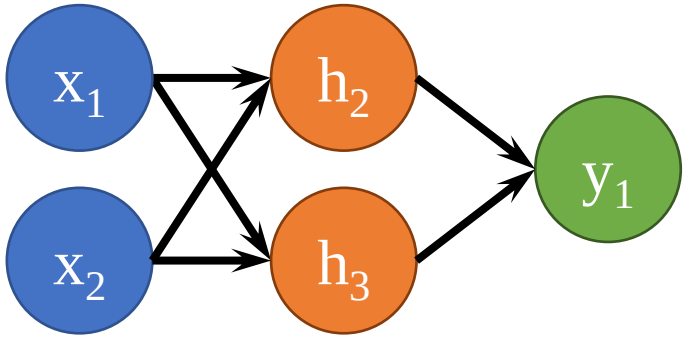
深度的重要性

可以实现xor吗?





No



Yes

结论

有隐藏层的前馈神经网络，即使神经元数量是有限的，也能逼近任何在有界区域内的连续函数。

Cybenko (1989): Cybenko在1989年证明，当使用sigmoid激活函数时，上述结果成立。
Hornik (1991): Hornik在1991年进一步提出，这个逼近的性质不仅仅适用于sigmoid激活函数，更是普遍存在的

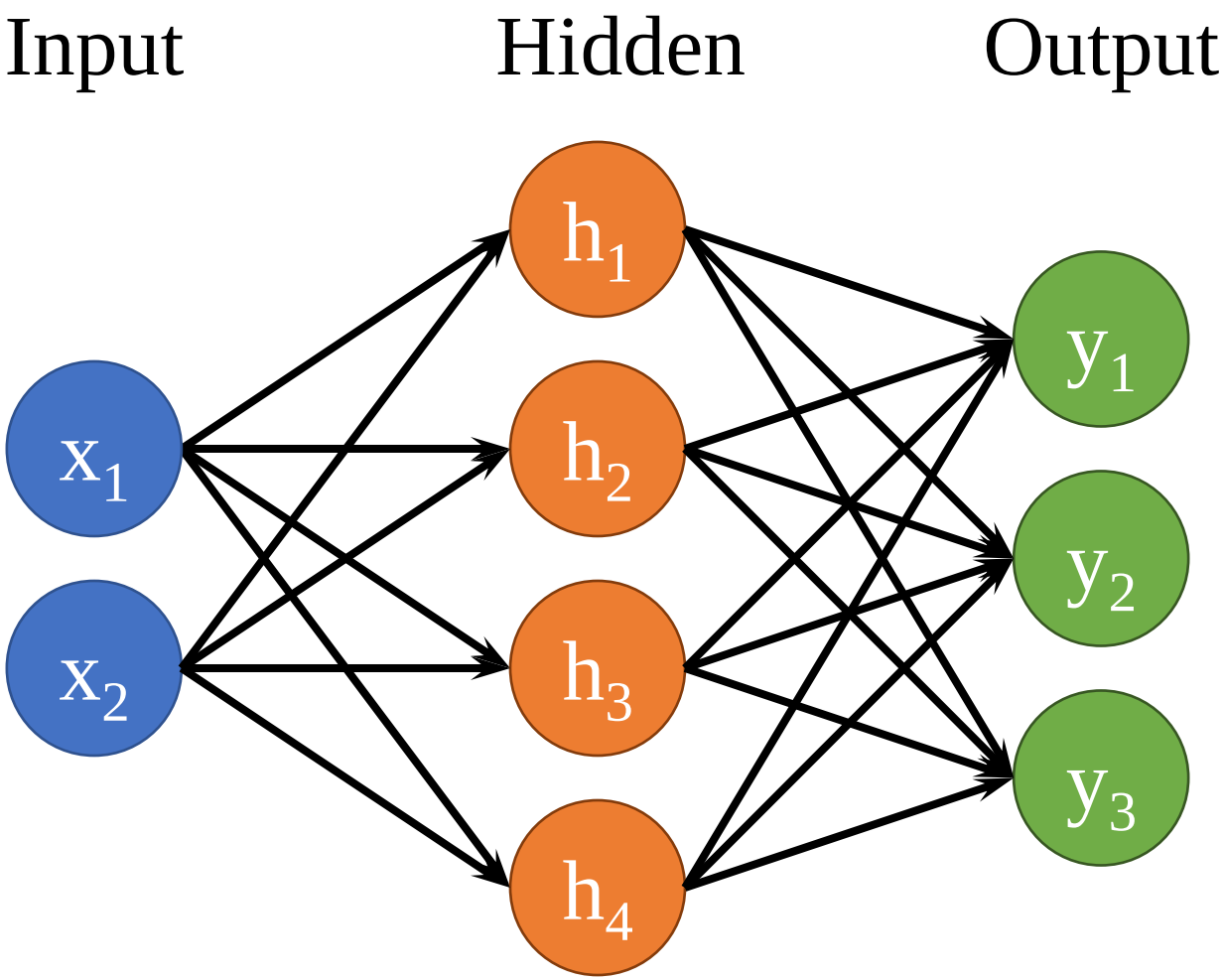
尽管理论上神经网络具有这种强大的逼近能力，但在实际应用中，这并不总是提供一个实用的保证。
Why?

忠告——过拟合、局部最优

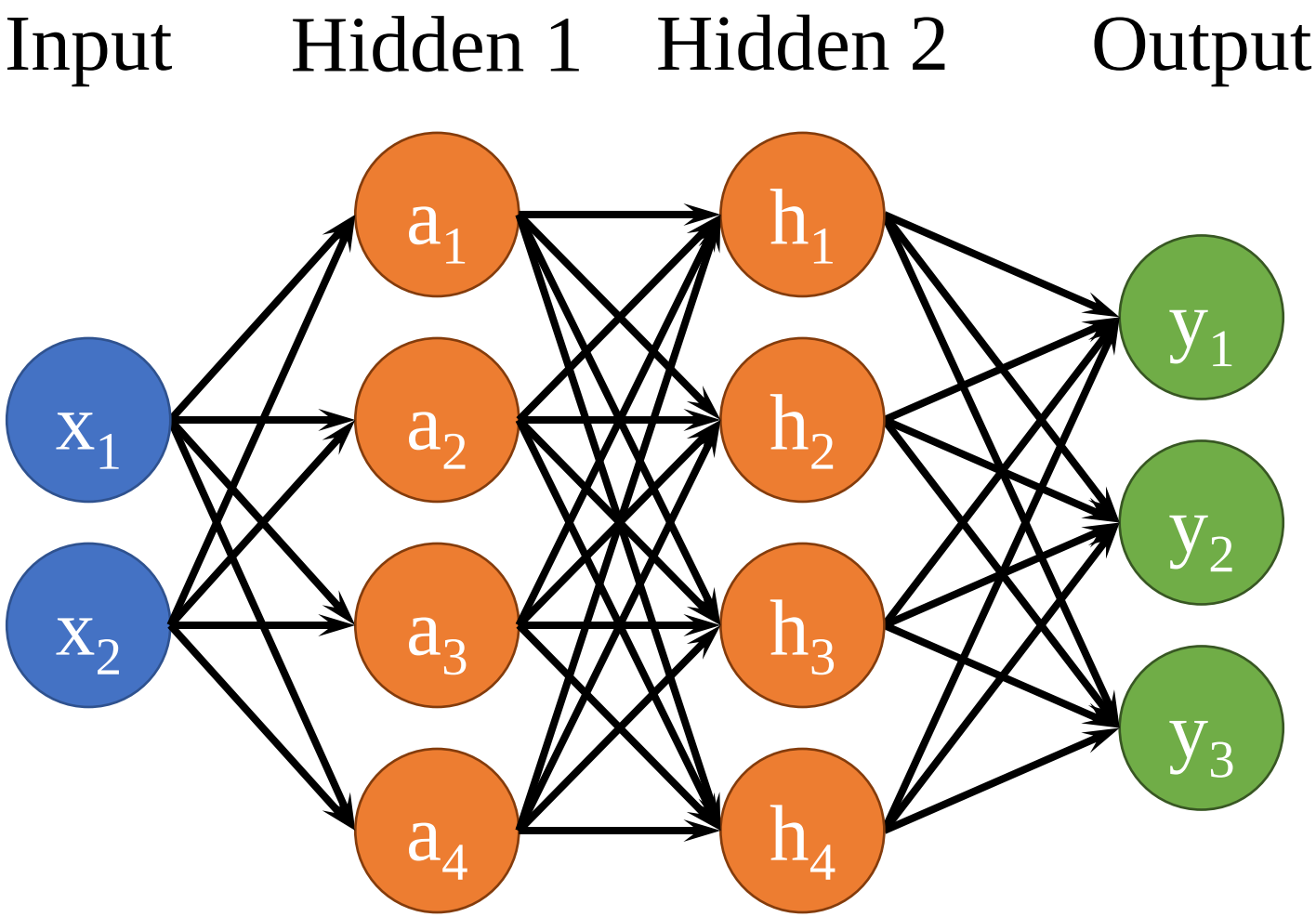
There is no royal road to geometry. – Euclid

- 全面了解、掌握数据: 亲自操作和实验数据是获得直觉的最好方法。
- 持怀疑态度: 对于你所做的每件事，甚至你所听到的每件事，都要持怀疑态度。
- 神经网络和其他机器学习技术的背后是数学和逻辑，不是“炼金”与“魔法”。
- 如何手动设置深度网络的权重。这个思考技巧可以帮助理解权重如何影响网络的行为和输出，尽管在实践中我们通常不会手动设置这些权重。

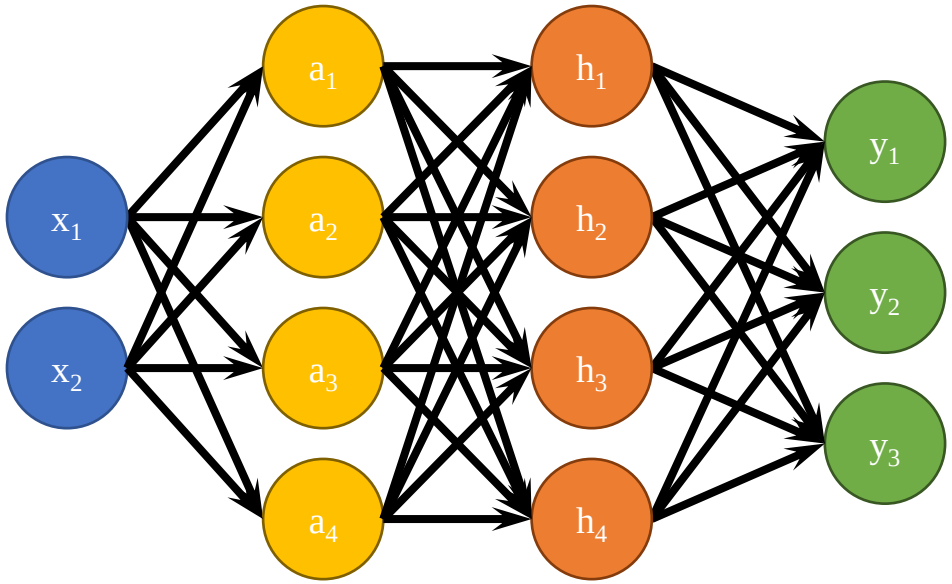
神经网络



更深的神经网络

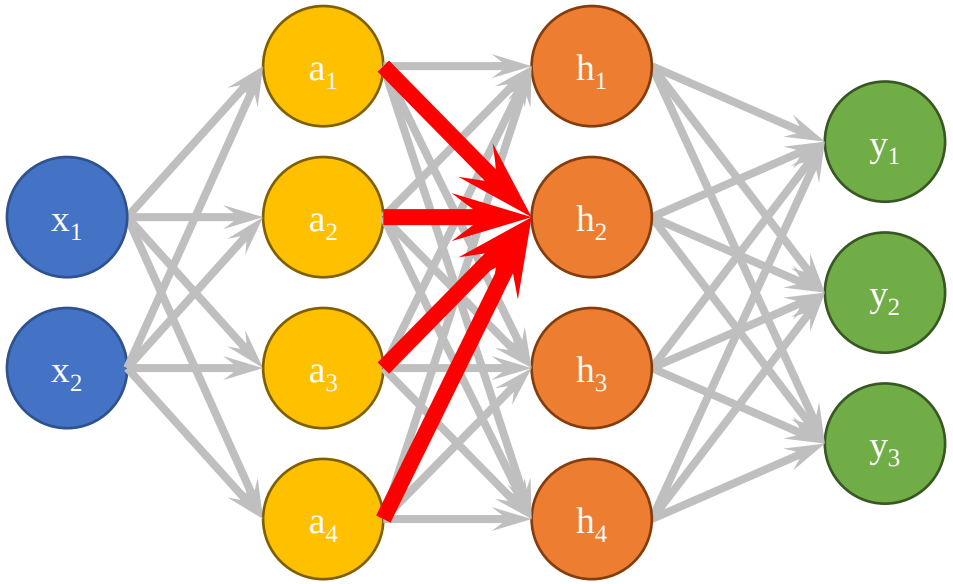

Input
Hidden 1
Hidden 2
Output
x1
x2
a1
a2
a3
a4
h1
h2
h3
h4
y1
y2
y3

全连接网络



每个神经元都跟之前一层的神经元全部相连

全连接网络



$\boldsymbol{a}$ 上一层 a 的值

$\boldsymbol{w_i, b_i}$ 神经元 i 权重 weight, 偏置 bias

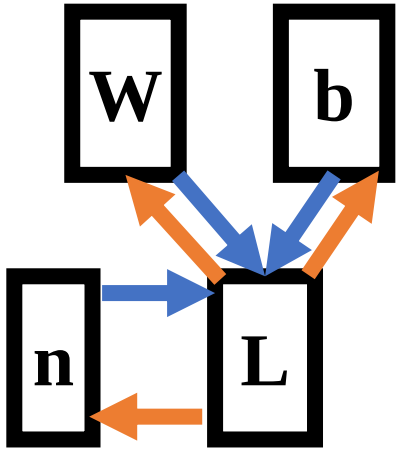
f 激活函数

$$\boldsymbol{h = f(Wa + b)}$$

$$\begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \end{bmatrix} = f\left(\begin{bmatrix} w_1^T \\ w_2^T \\ w_3^T \\ w_4^T \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \right)$$

全连接网络

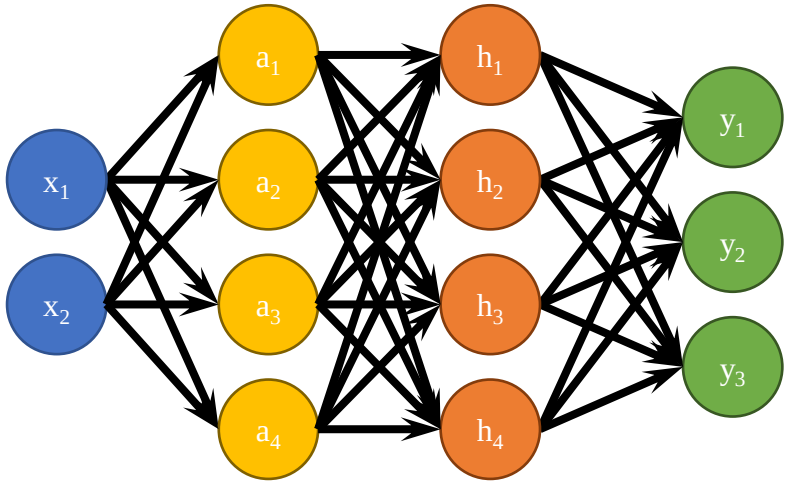
我们也称其为: “Linear Layer 线性层”

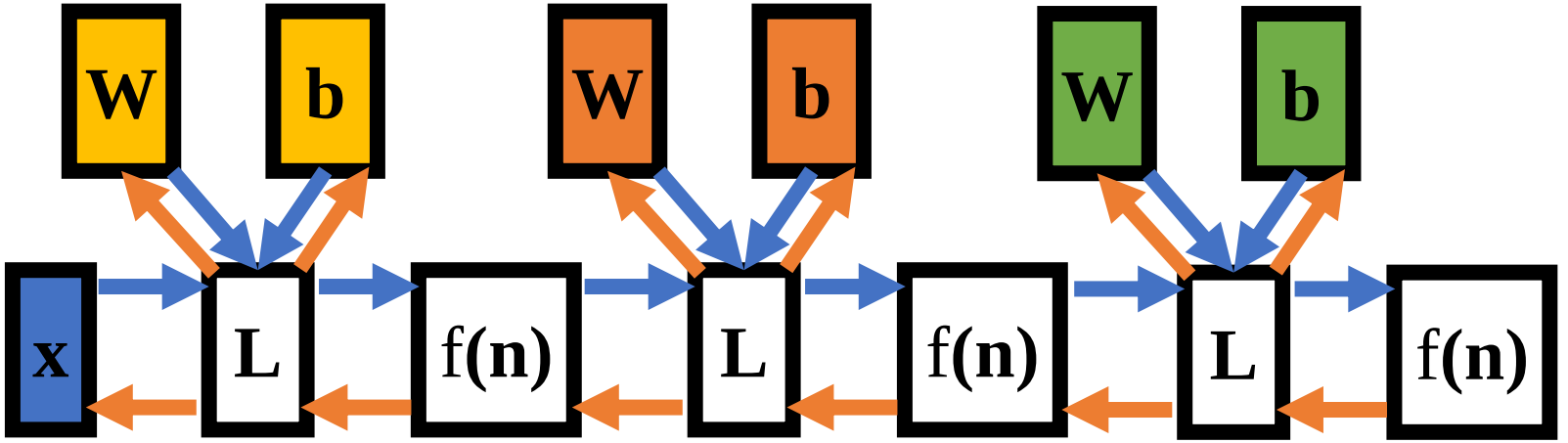


$$L(\boldsymbol{n}) = \boldsymbol{W}\boldsymbol{n} + \boldsymbol{b}$$

可以根据输入计算梯度优化

全连接网络

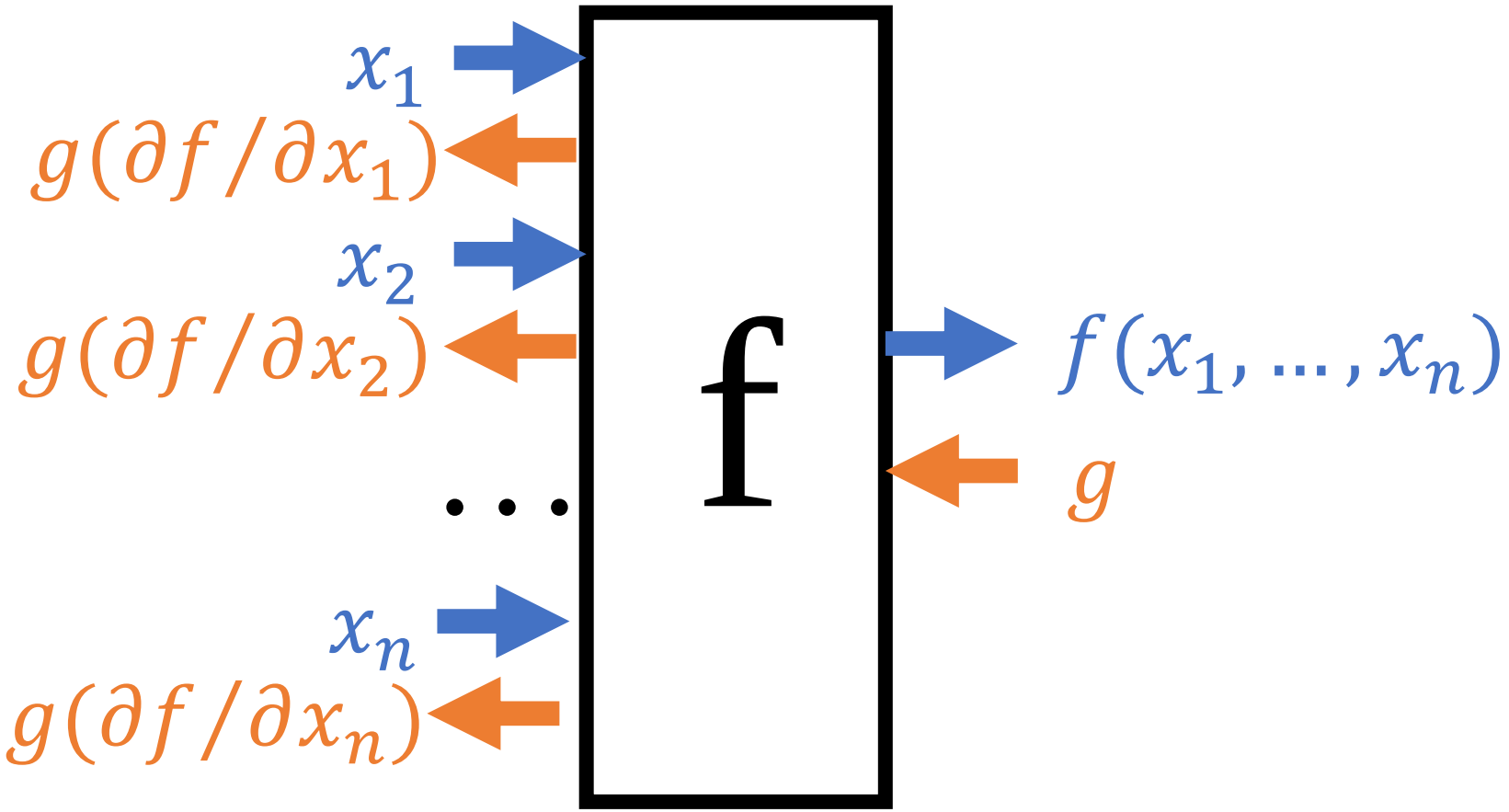




Backpropagation 反向传播

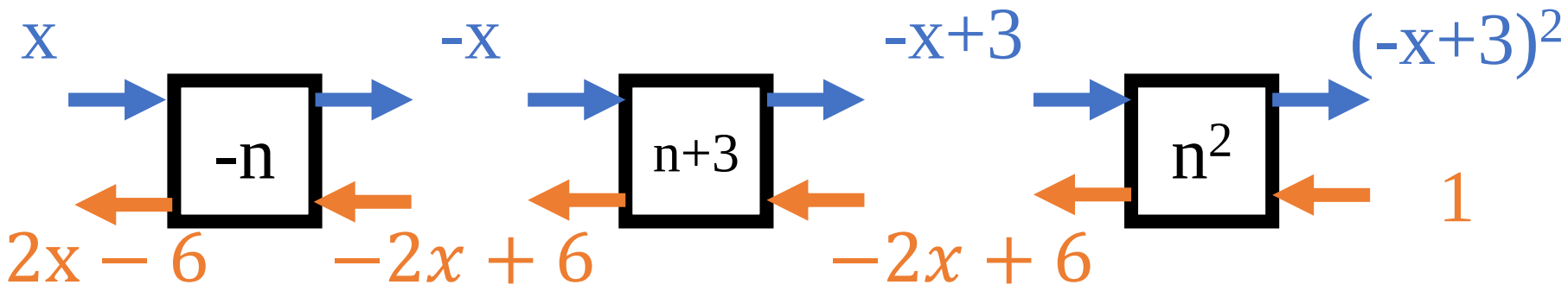
每个模块计算 反向传递 与输入的局部梯度相乘

backwards (g) * local gradient (df/dx_i)



Backpropagation 反向传播

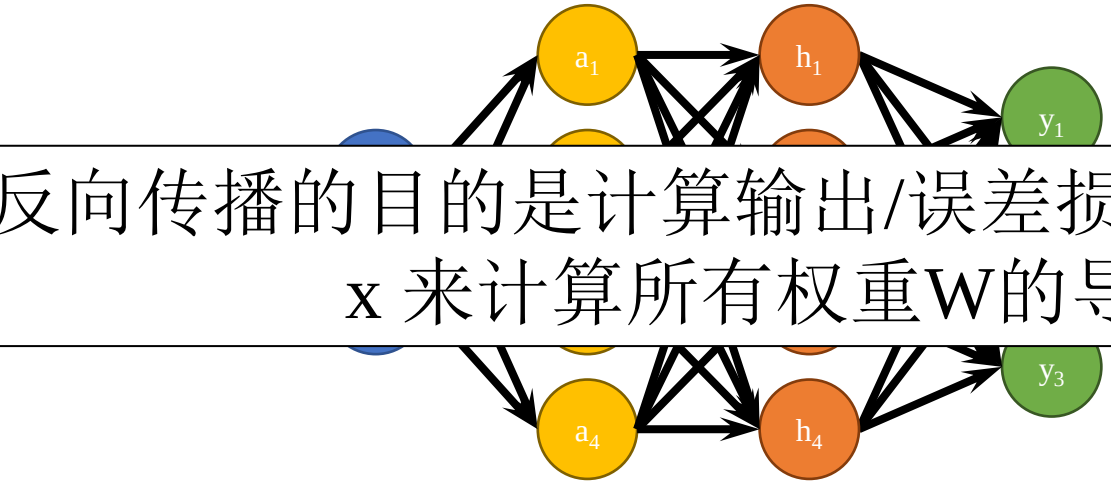
$$f(x) = (-x + 3)^2$$



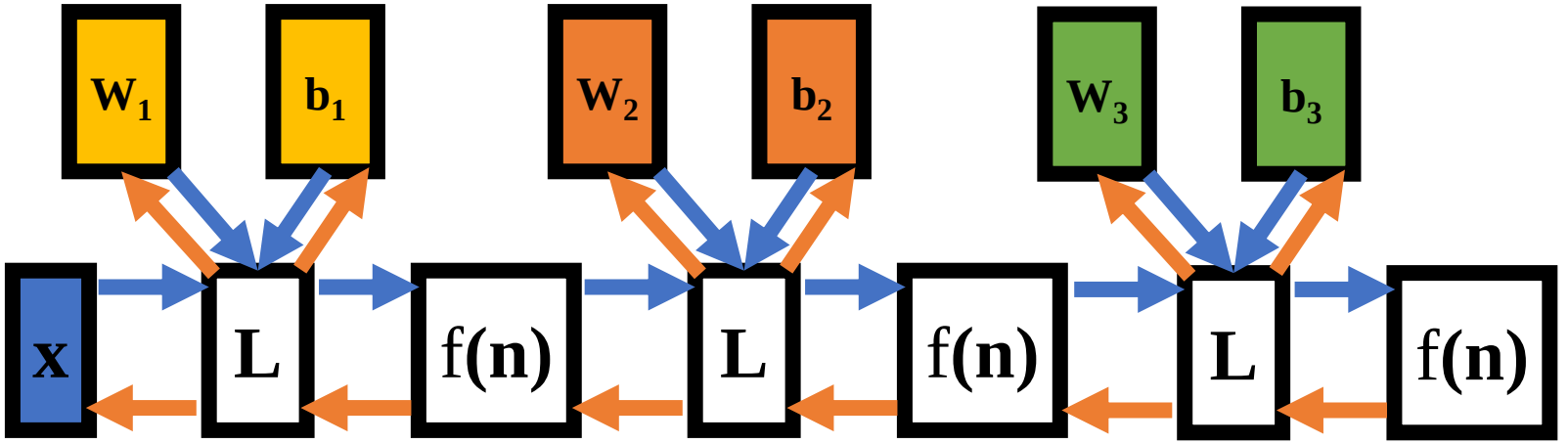
前向传递Forward pass: 带入模块计算

反向传递 Backward pass: 计算相对模块和输入的导数

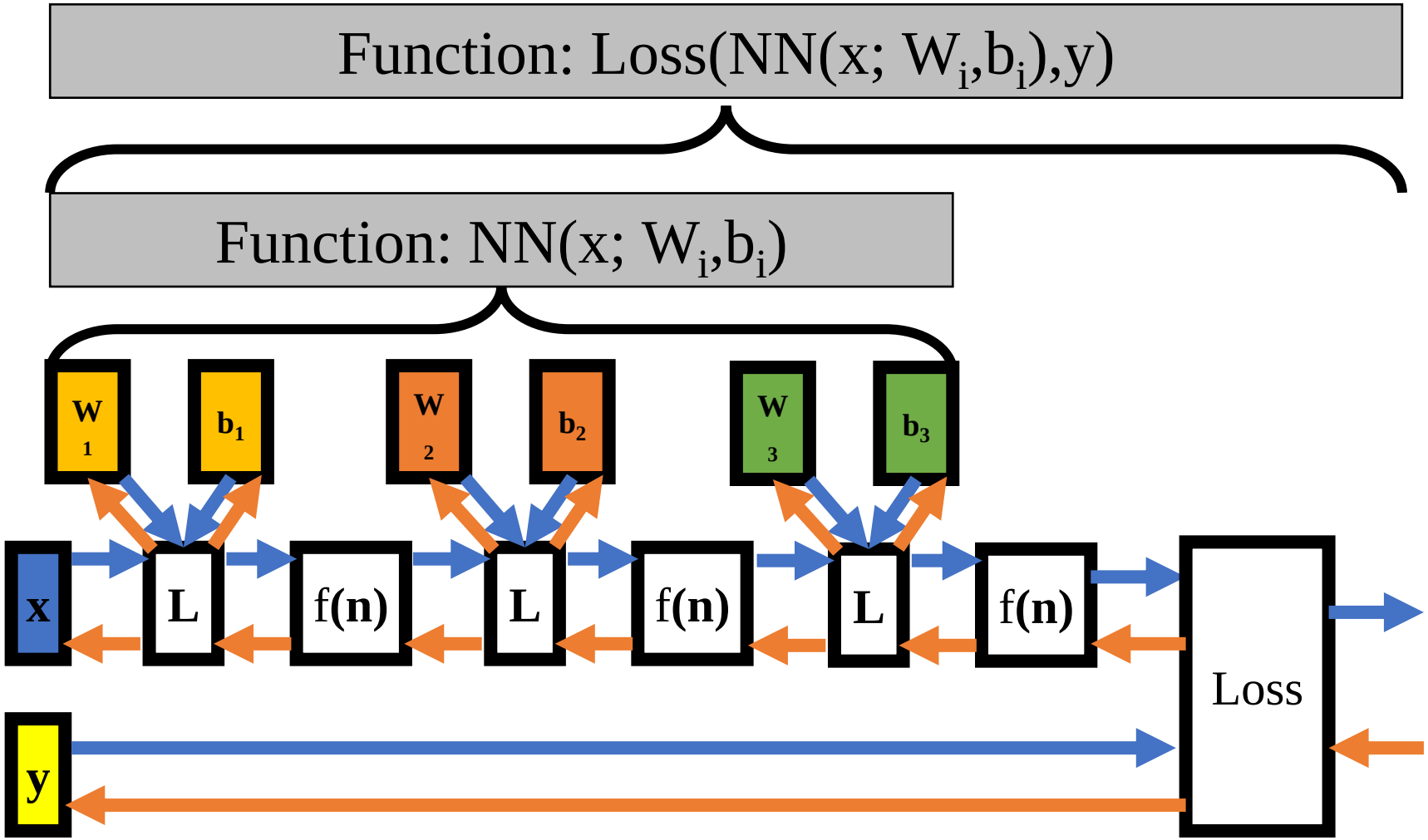
全连接网络



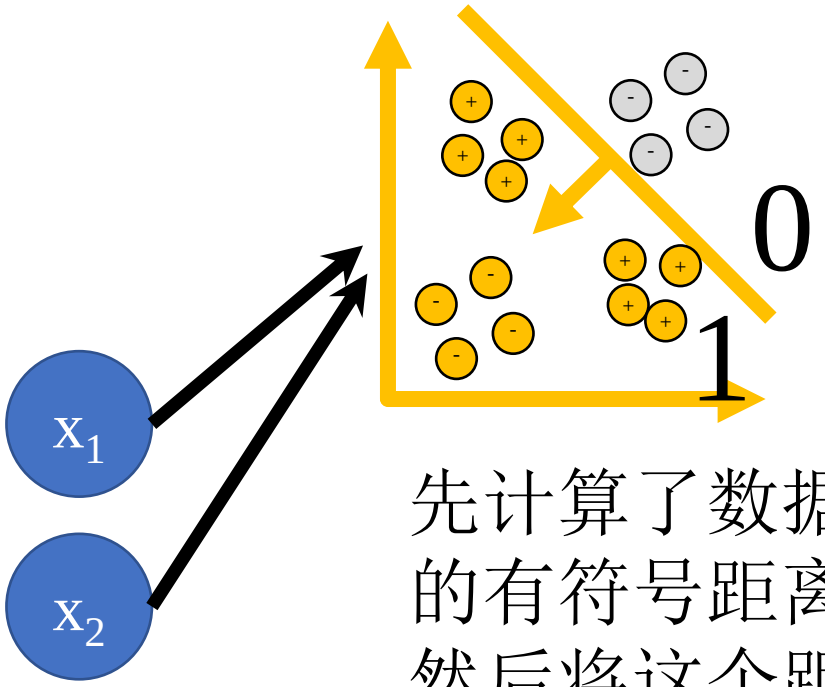
反向传播的目的是计算输出/误差损失与给定输入 x 来计算所有权重W的导数



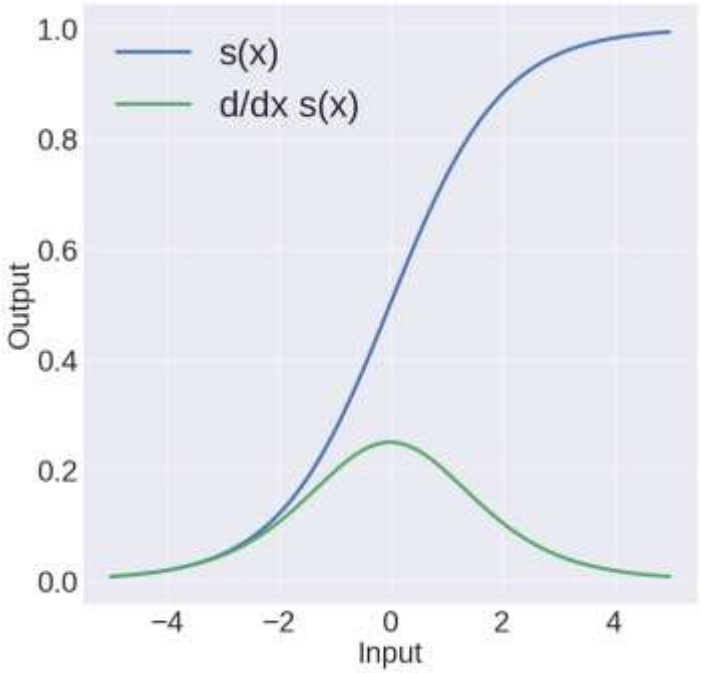
训练全连接网络


Function: Loss(NN(x; W_i,b_i),y)
Function: NN(x; W_i,b_i)
W 1
b1
W 2
b2
W 3
b3
x
L
f(n)
L
f(n)
L
f(n)
Loss
y

激活函数的引入



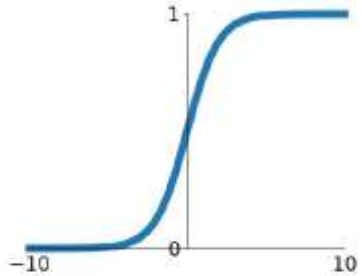
先计算了数据点的有符号距离，然后将这个距离传递给sigmoid激活函数



Activation functions

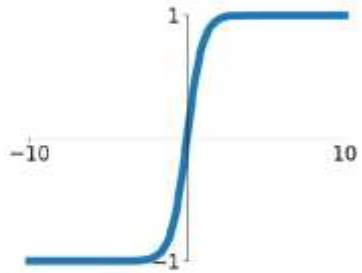
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



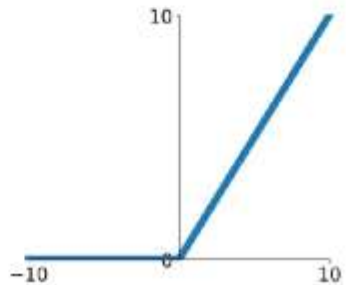
tanh

$$\tanh(x)$$



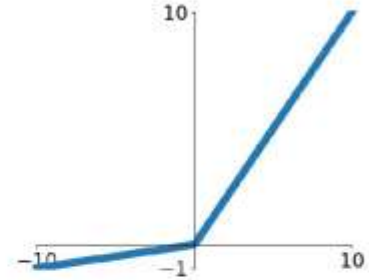
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

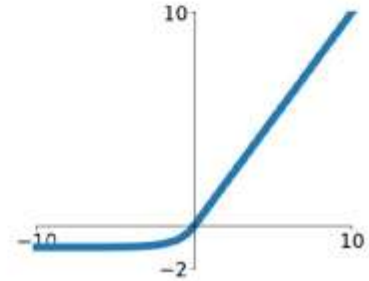


Maxout

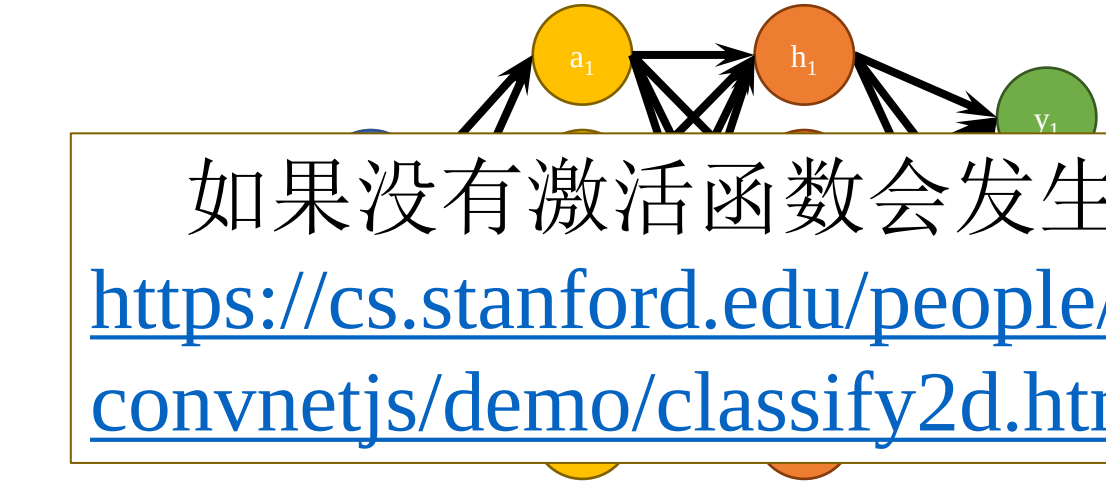
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

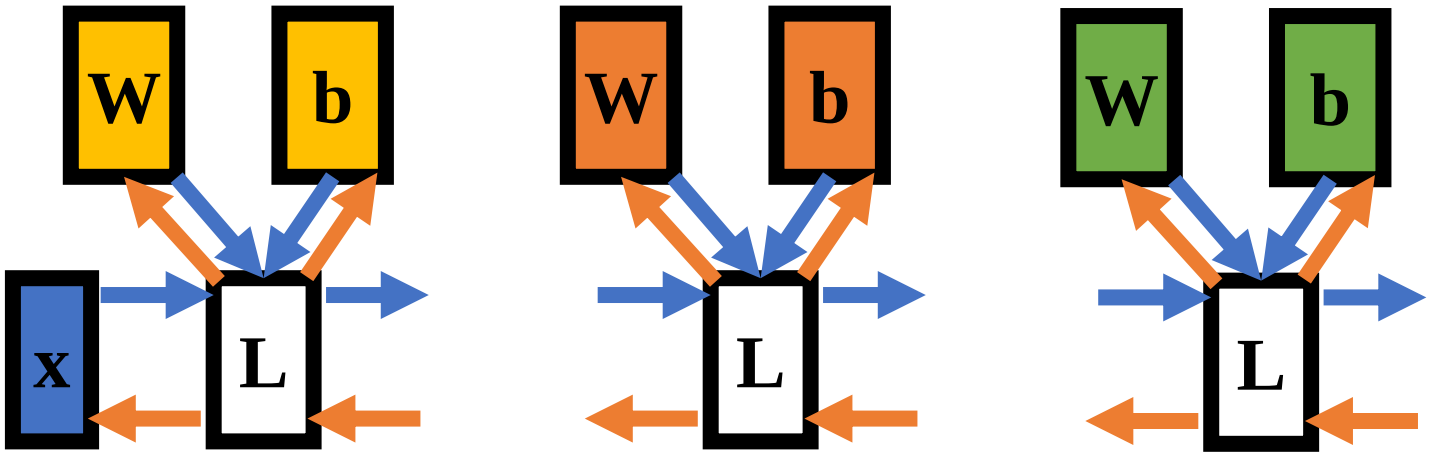


全连接网络



如果没有激活函数会发生什么？

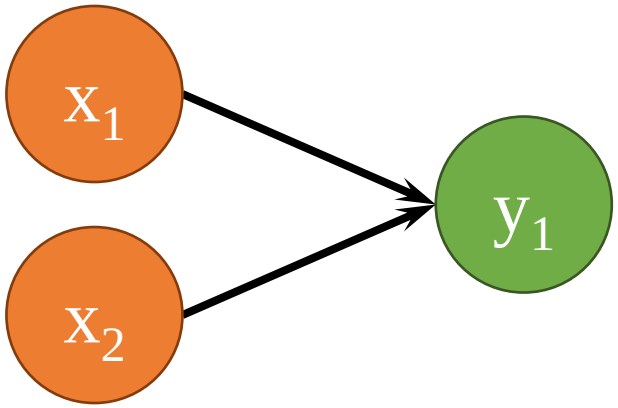
https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html



卷积神经网络

参数量

这个网络有多少参数?

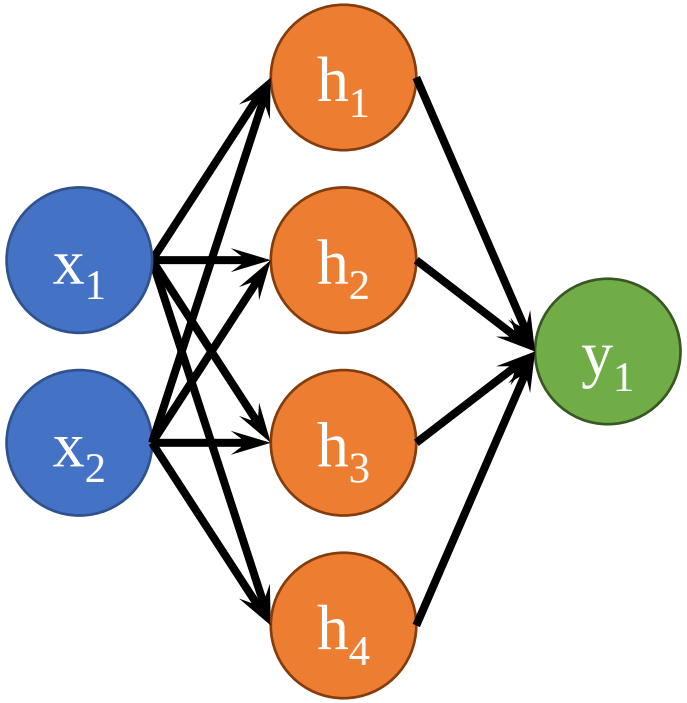


Weights 权重: 1x2

总参数量: 3 (不要忘了偏置 bias!)

参数量

这个网络有多少参数?

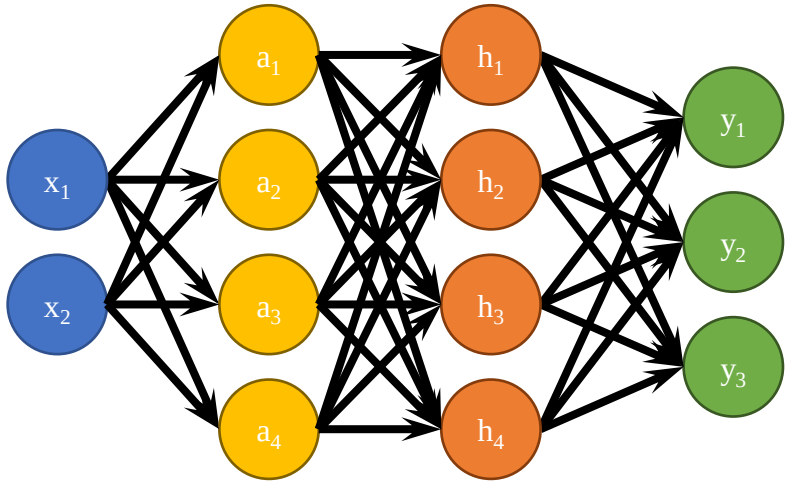


权重: 1x4+4x2 = 12

总参数量: 12+5 = 17

参数量

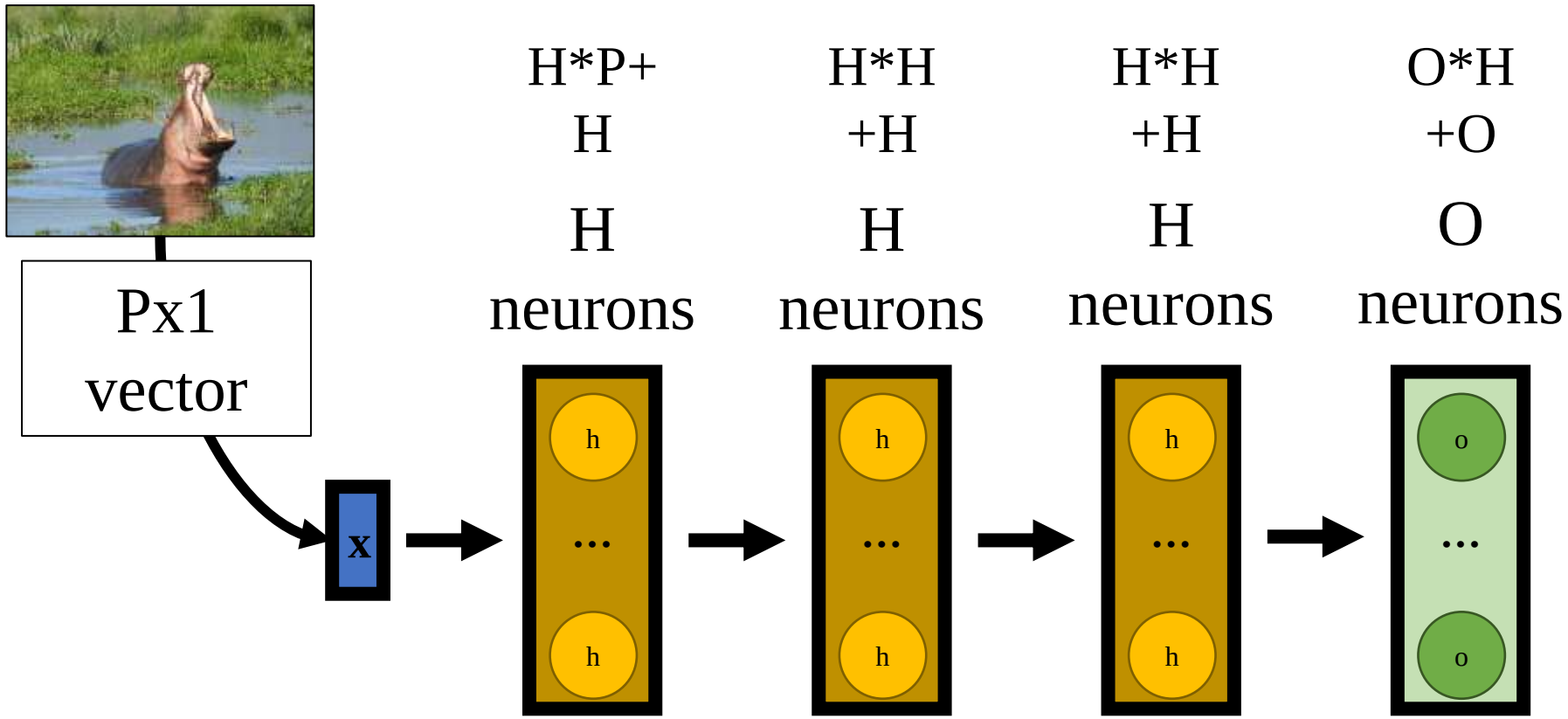
这个网络有多少参数?



权重: 3x4+4x4+4x2 = 36

总参数量: 36+11 = 47

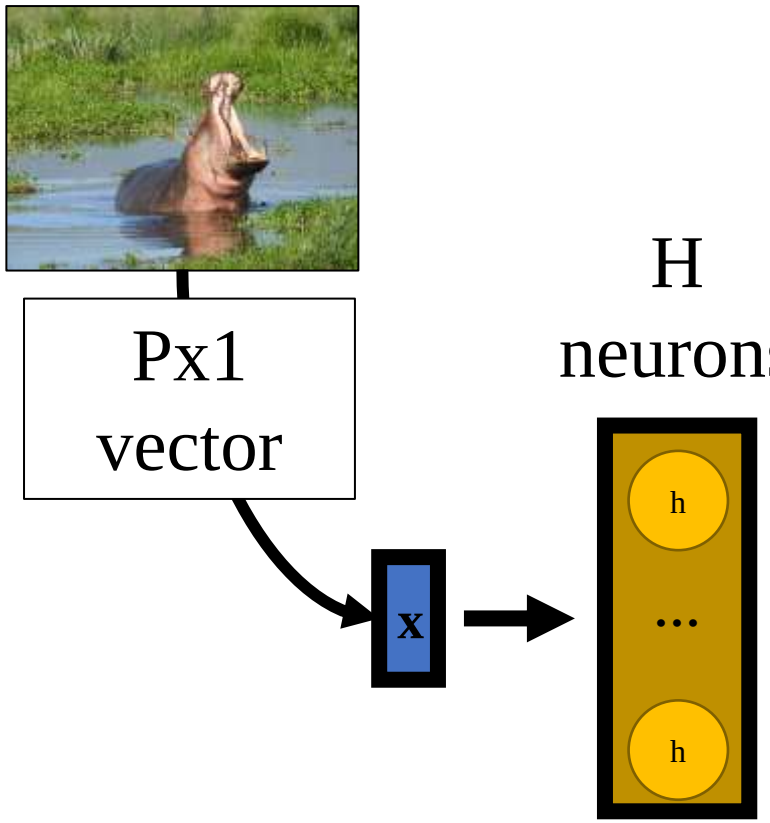
参数量



P: 285x350 , H: 1000, O: 3

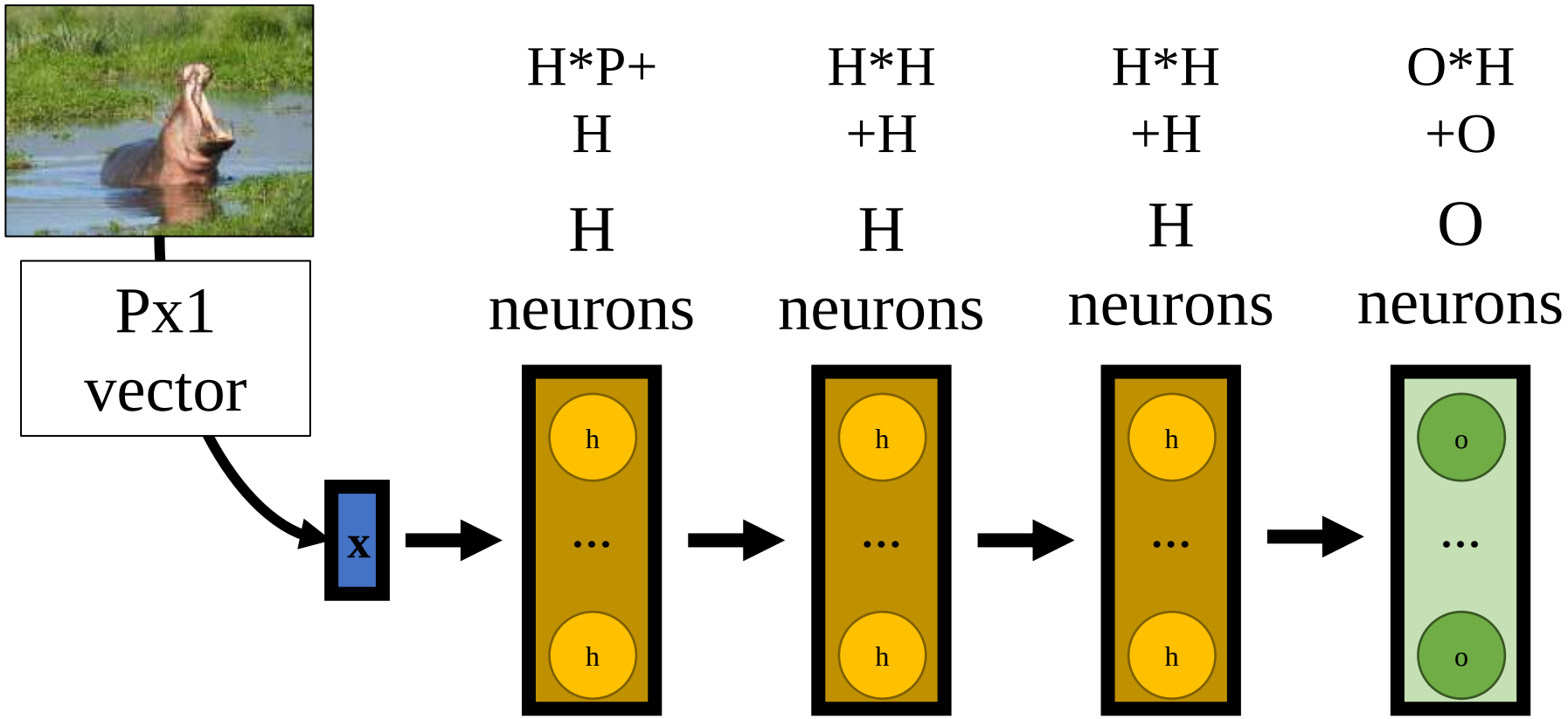
102 million 个 参数 (400MB)

参数量



- H 决定了网络的大小
- 第一层的参数量是P*H+H
- 如果我们要计算图像梯度dx/dy.**我们需要多少H?**
- **2P!**

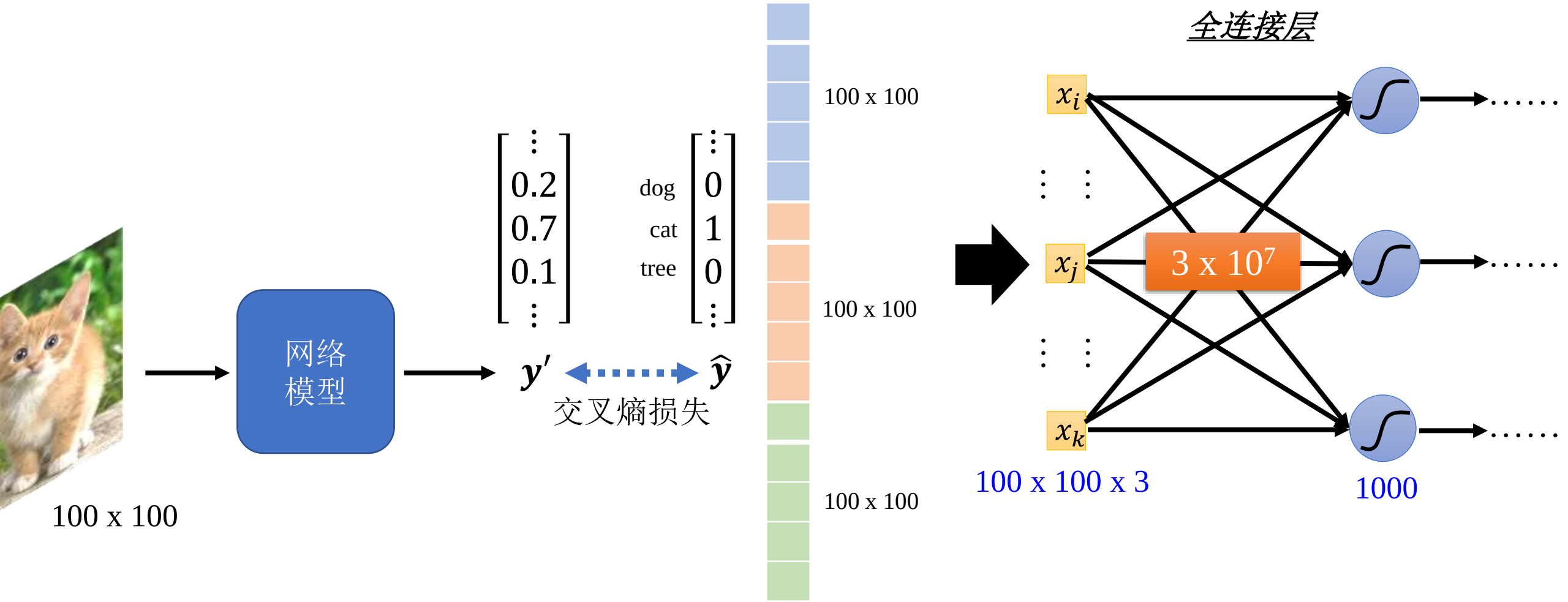
参数量



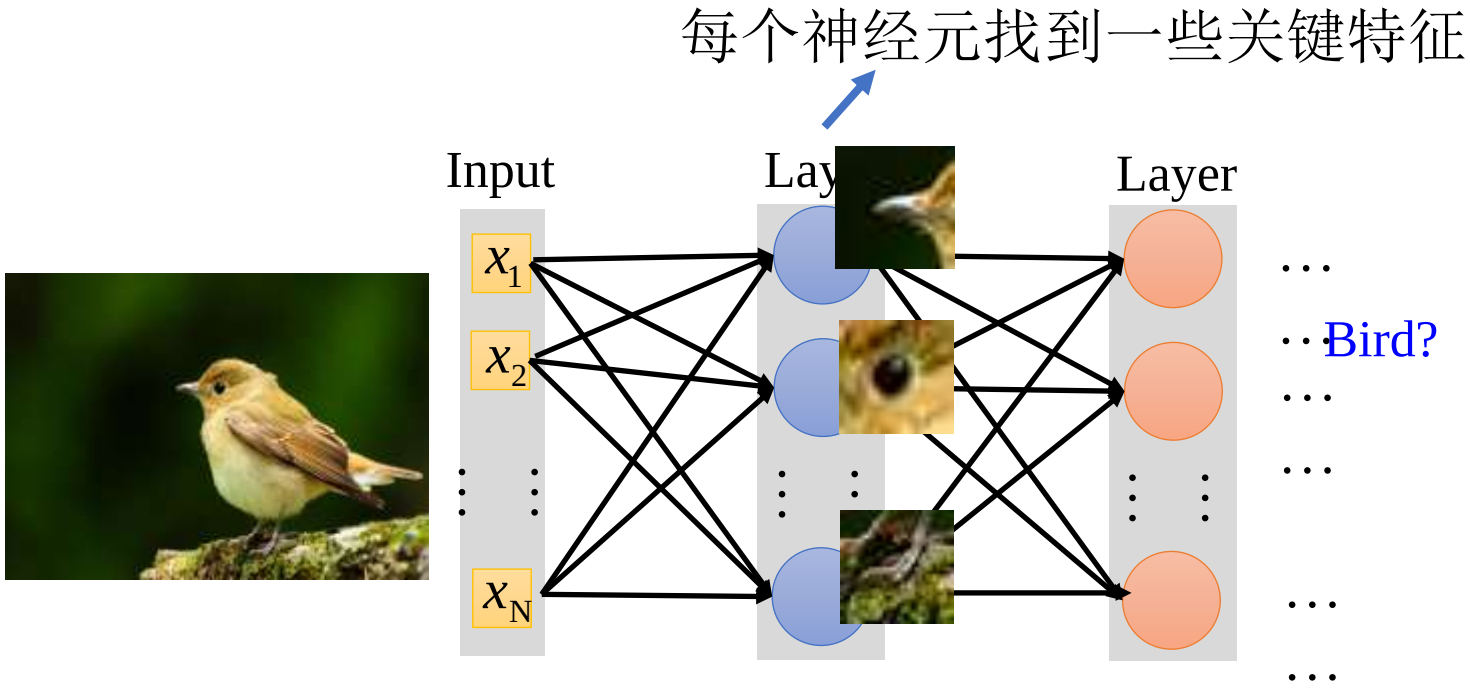
P: 285x350, H: 2P, O: 3

100 billion 个 参数 (400GB)

我们真的需要这么多的参数量吗



简化全连接层

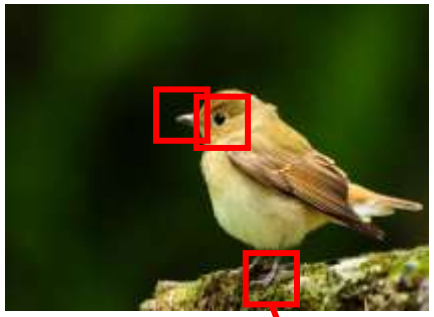


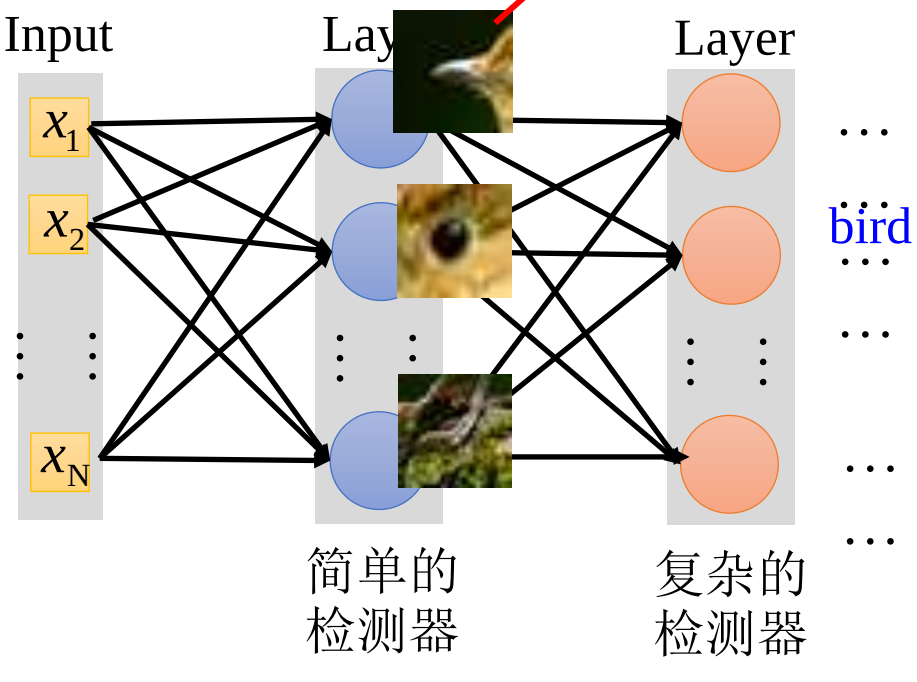
与前面提到的特征检测——提取——匹配类似，
也许人也是用类似的方式来识别鸟…

简化全连接层

每个神经元并不需要看到所有像素点（回顾特征尺度）

我们需要看整张图来分类吗？（回顾特征点）



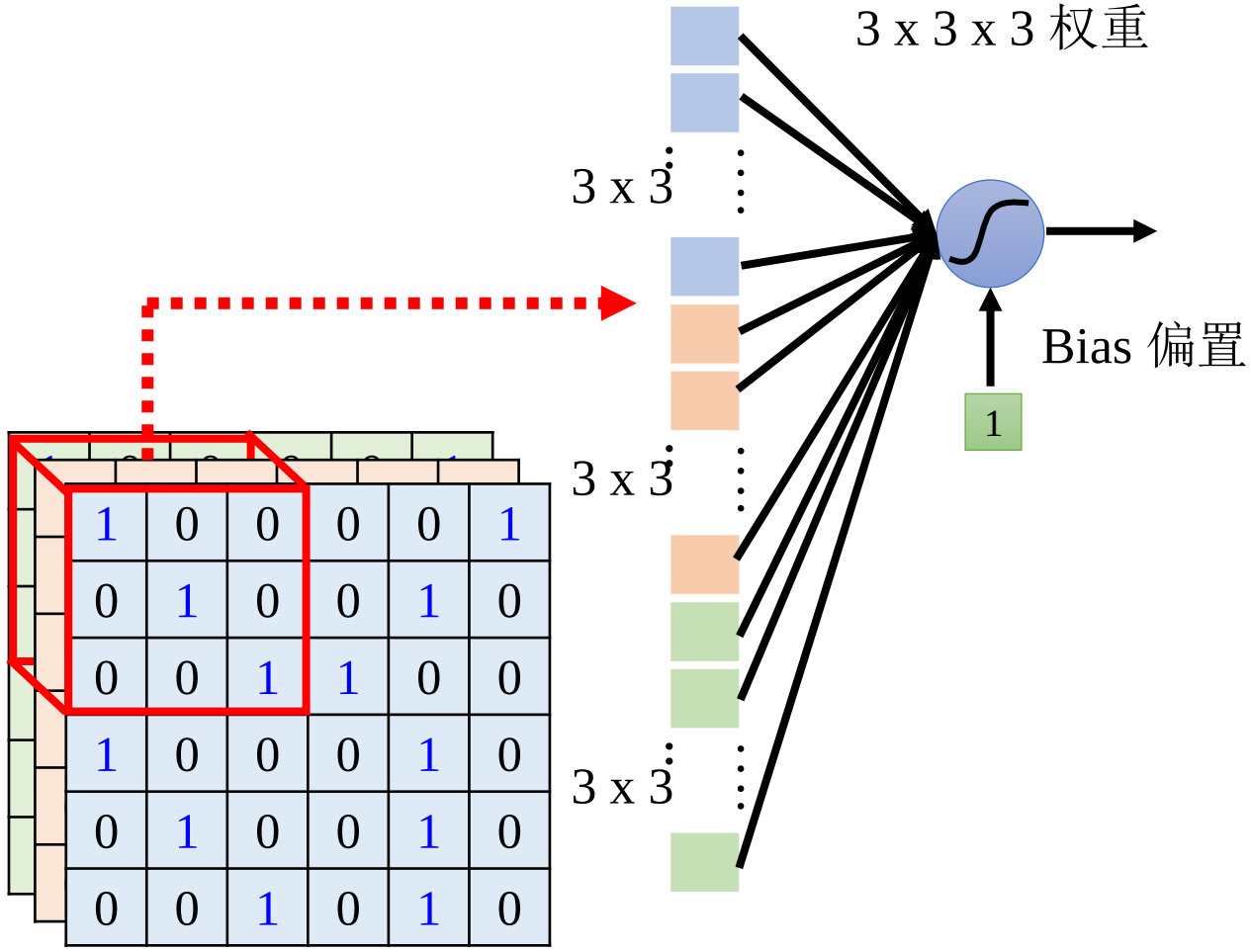


有些关键特征也许很小（回顾特征尺度）

神经元通常只对图像中的局部模式或特征进行响应，而不是整张图像。

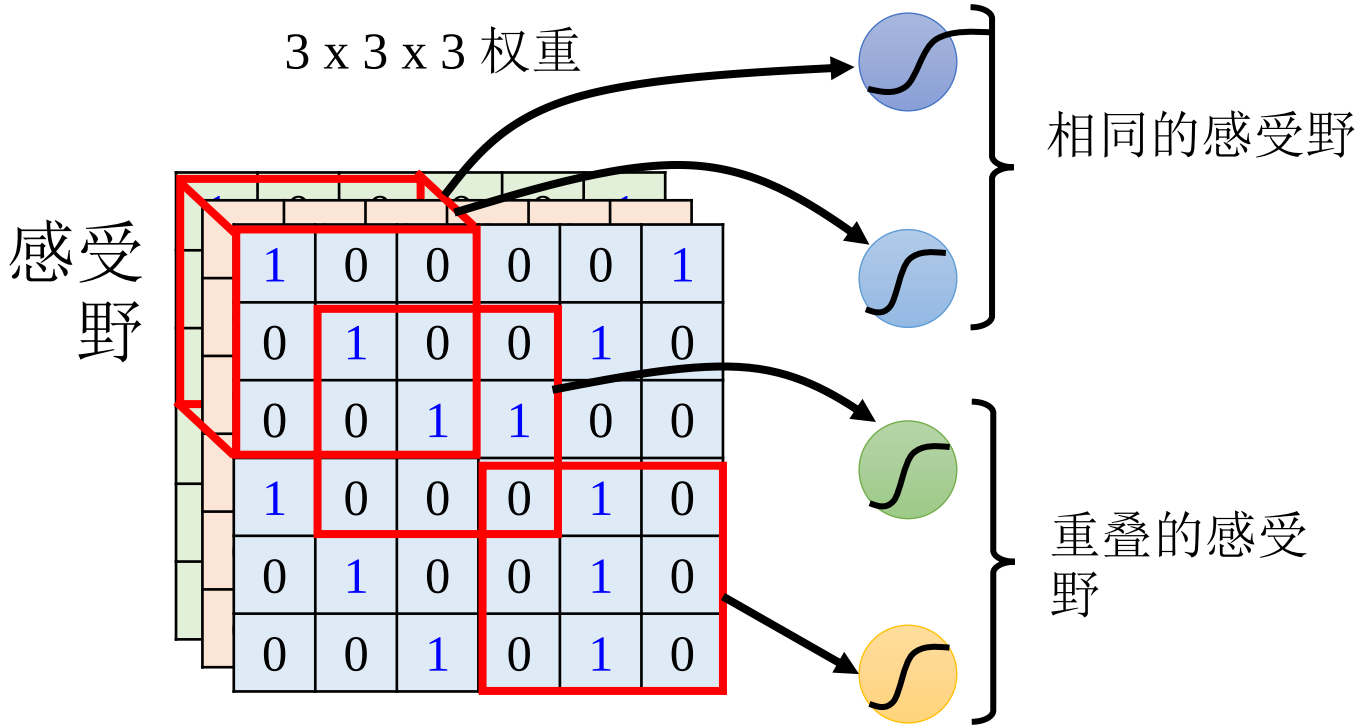
感受野 Receptive Field

感受野：图像中一个局部区域，其内容会影响到一个特定的神经元的激活。



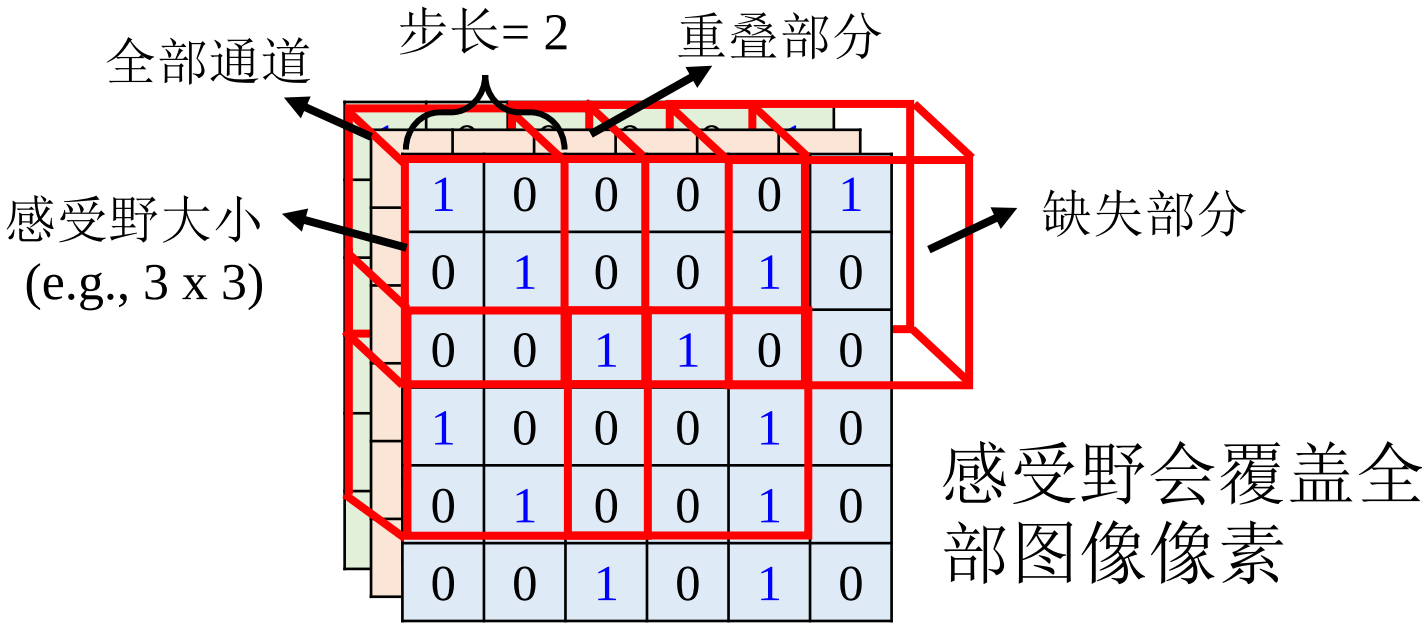
感受野

- 不同的神经元可以有不同大小的感受野吗?
- 感受野可以不涉及所有通道吗?
- 感受野可以是任意形状的吗?



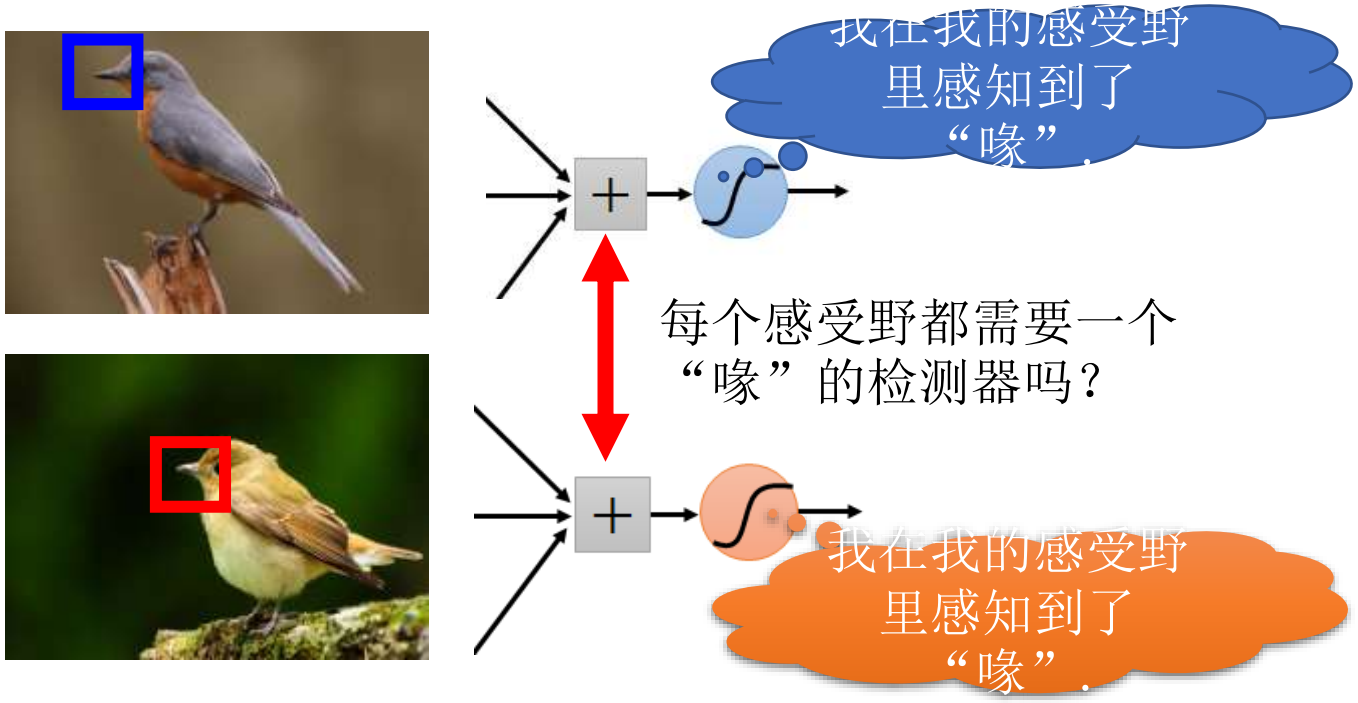
感受野

每个感受野会占用某些神经元（通道）.

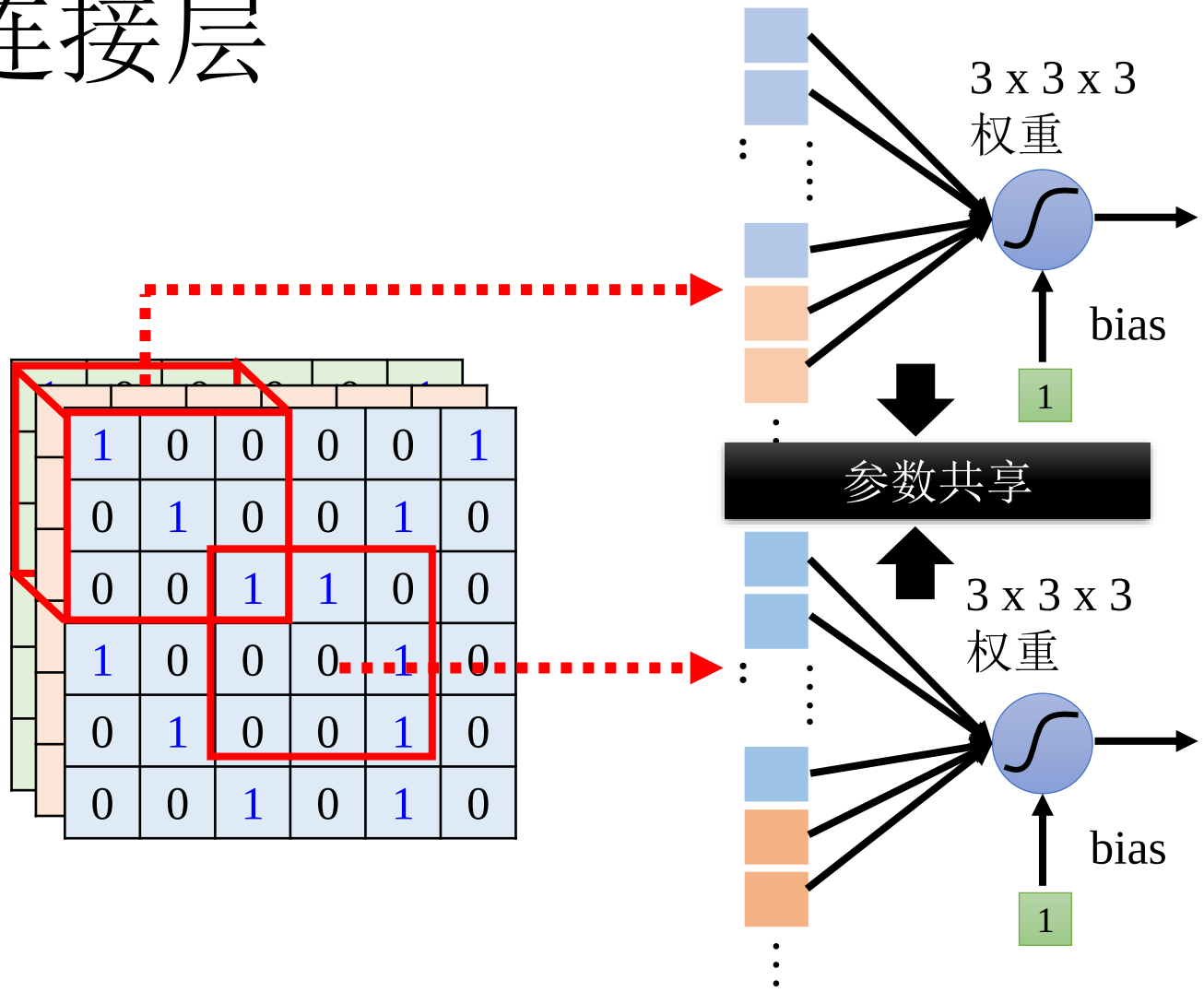


简化全连接层

- 相同的特征会出现在不同的区域.

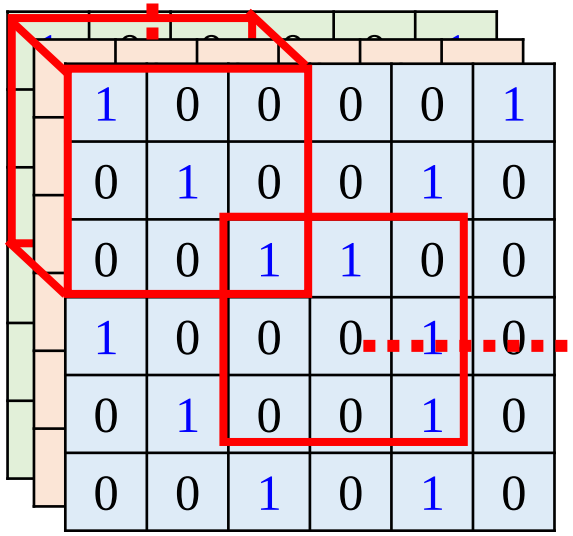


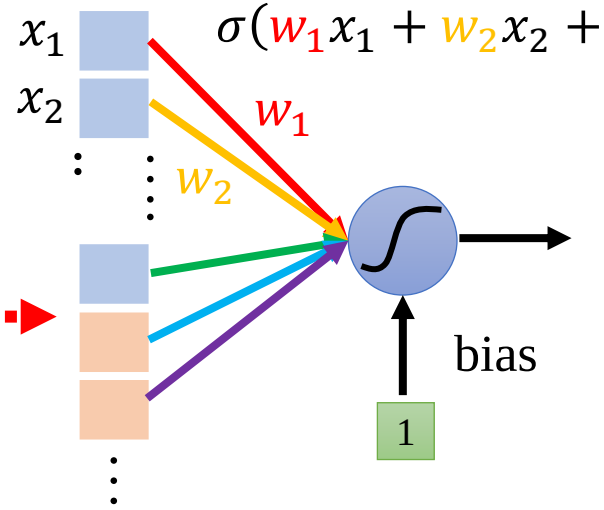
简化全连接层

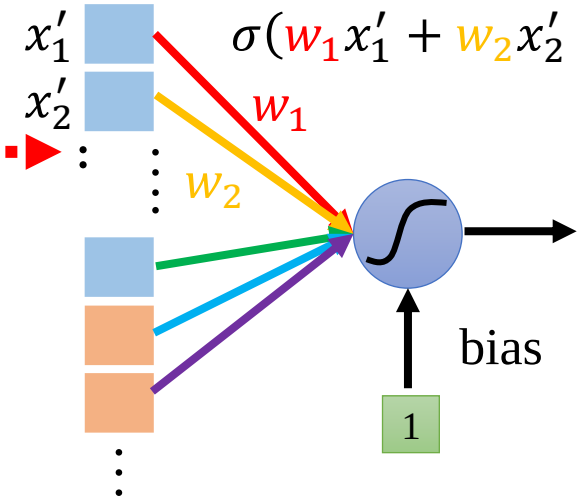


- 只需要一个3x3x3的权重矩阵，就可以在整张图像上检测特征。通过参数共享，卷积神经网络可以大大减少所需的参数数量，同时确保相同的特征在不同的位置都能被检测到。

简化全连接层





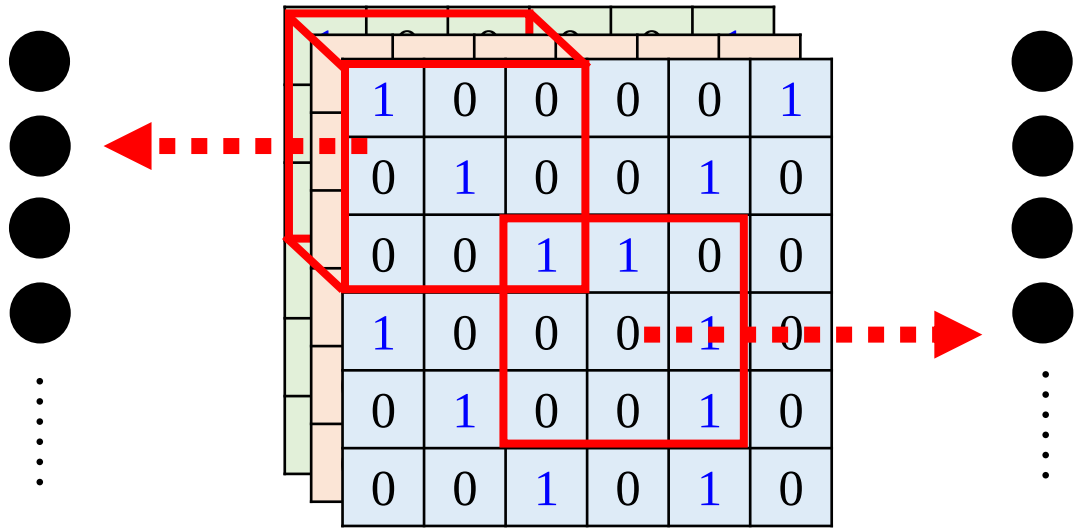


拥有相同感受野的神经元
也可以不共享参数.

- 不同的神经元用于不同的特征检测。

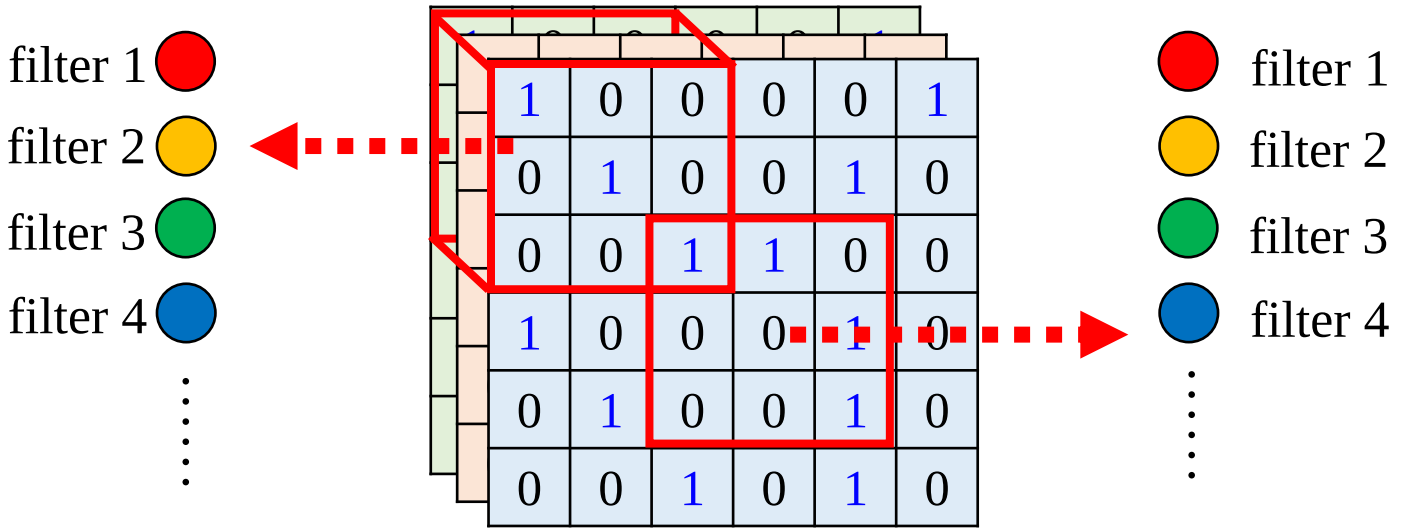
简化全连接层

一个感受野不仅仅连接到一个神经元，而是连接到一个神经元集合（例如64个神经元）。这样，网络可以在同一个感受野上检测多个特征。

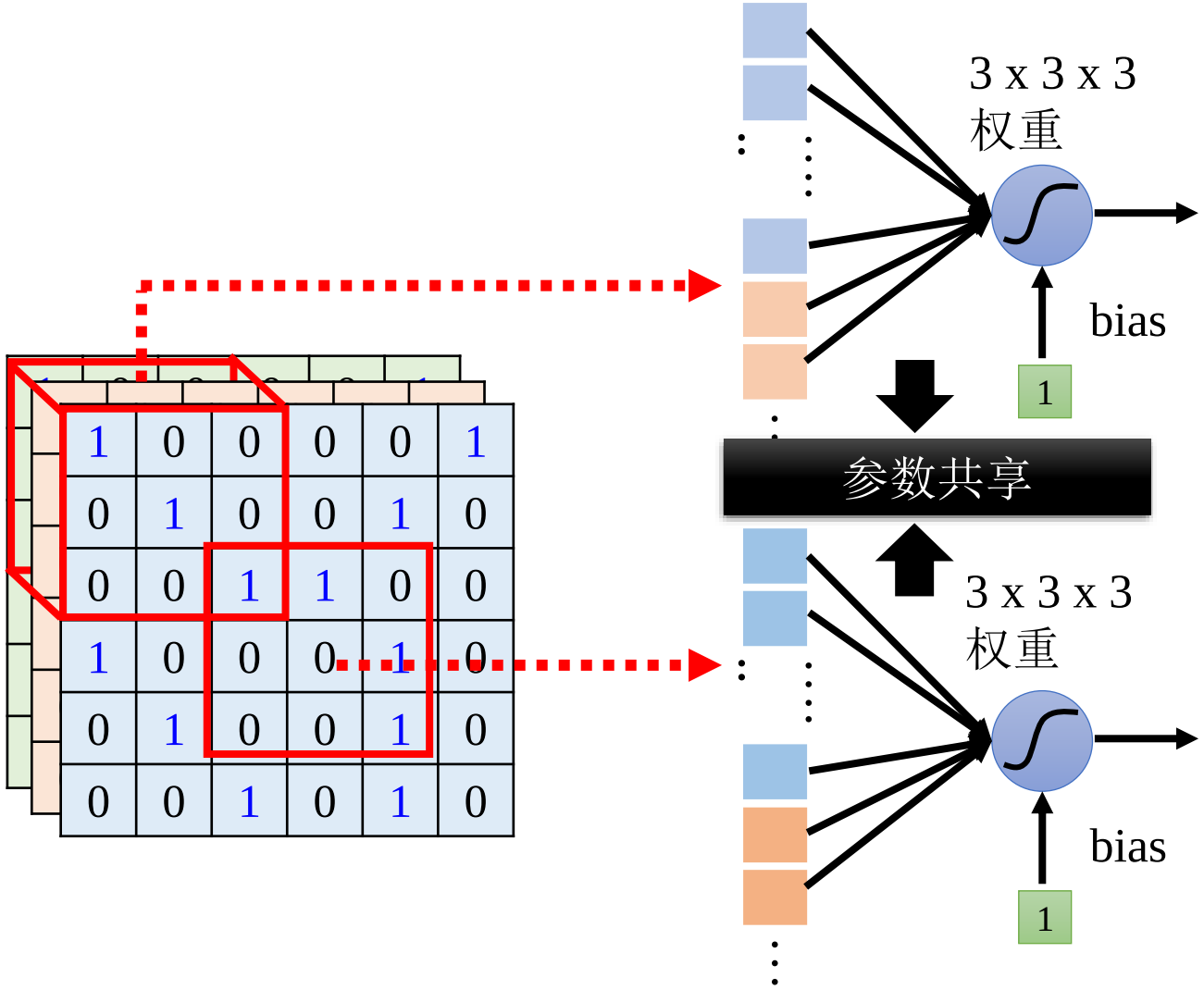


简化全连接层

一个感受野不仅仅连接到一个神经元，而是连接到一个神经元集合，会有特定的神经元起到类似滤波器的效果，在感受野内被激活，**相当于检测出不同特征**



简化全连接层的关键——参数共享

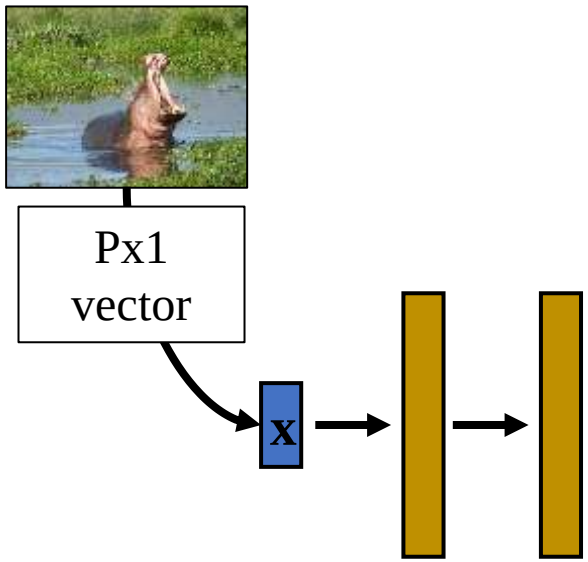


- 只需要一个3x3x3的权重矩阵，就可以在整张图像上检测特征。通过参数共享，卷积神经网络可以大大减少所需的参数数量，同时确保相同的特征在不同的位置都能被检测到。

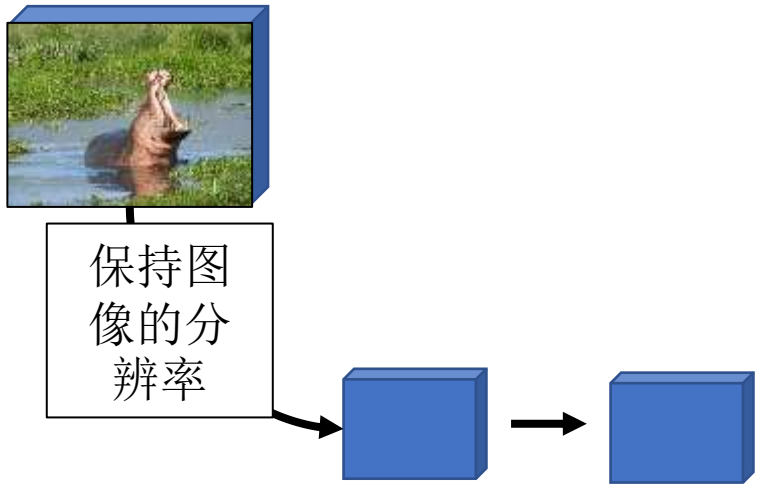
Convnets 卷积网络

不需要拉成向量，可以保持图像的分辨率

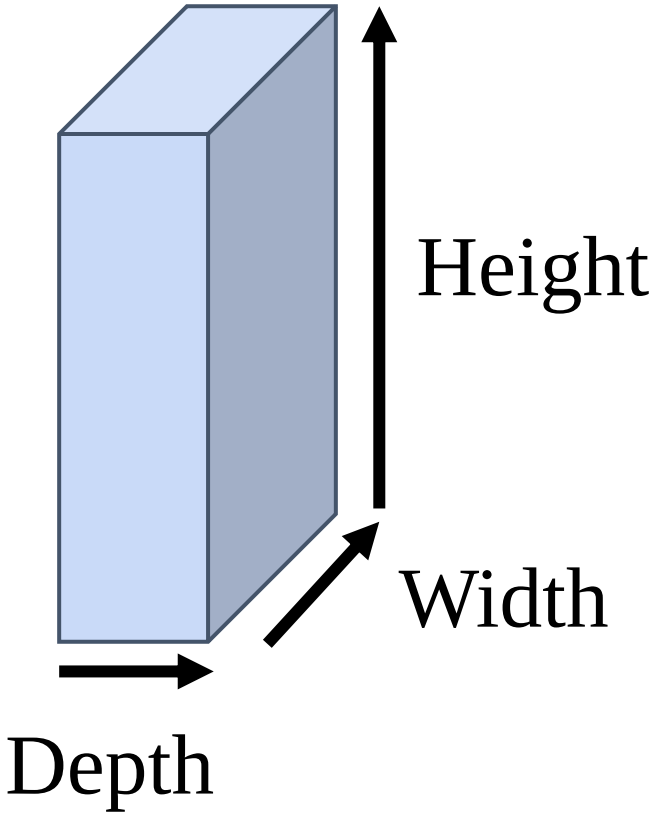
全连接网络:
数据: 向量 Fx1
形式: 矩阵乘法



卷积网络:
数据: 图像 HxWxF
形式: 卷积



卷积网络



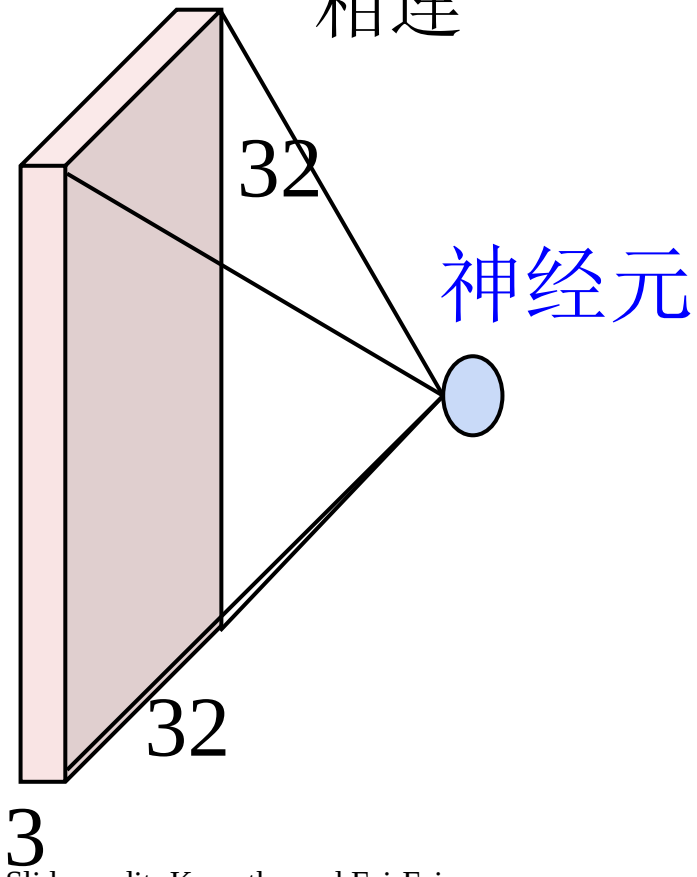
Height: 300
Width: 500
Depth: 3

Height: 32
Width: 32
Depth: 3

卷积网络

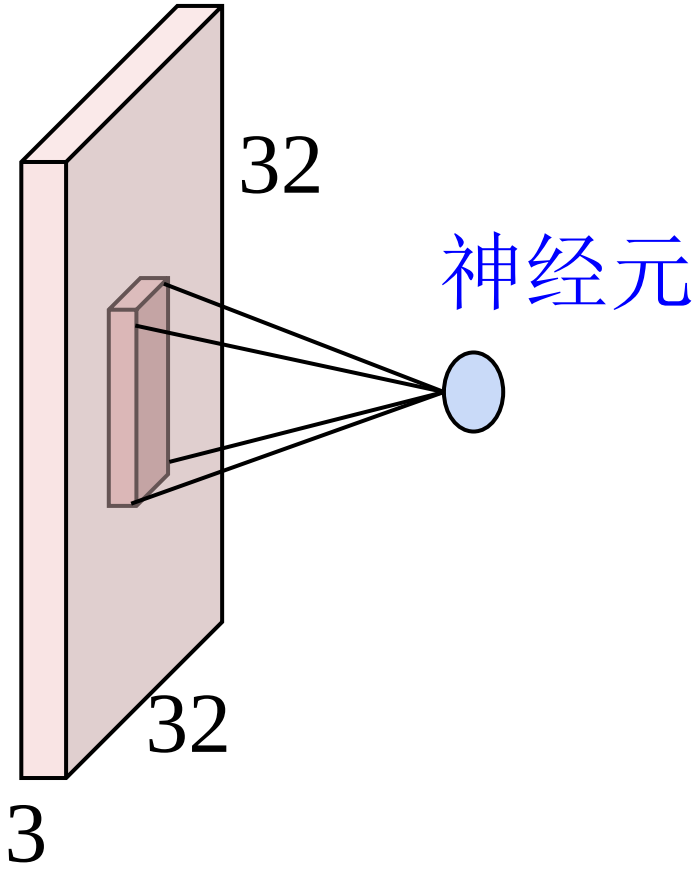
全连接:

每个神经元与所有像素相连



卷积网络:

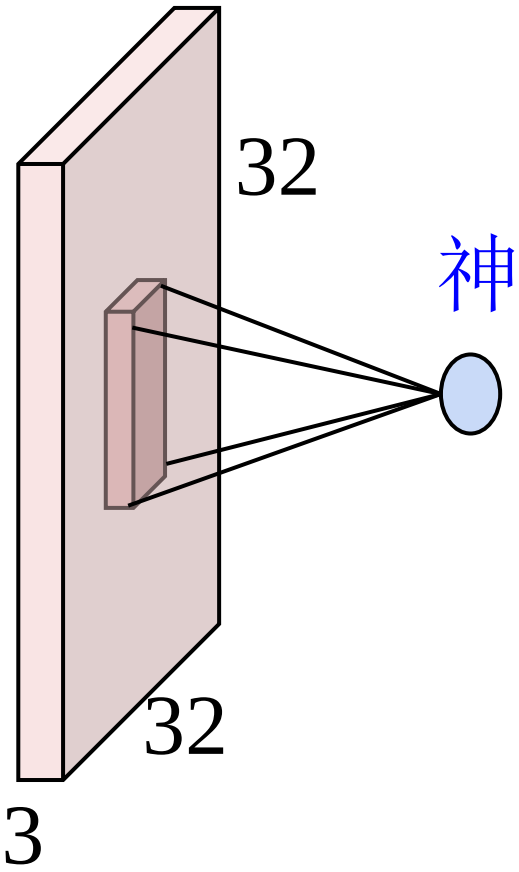
只与局部相连



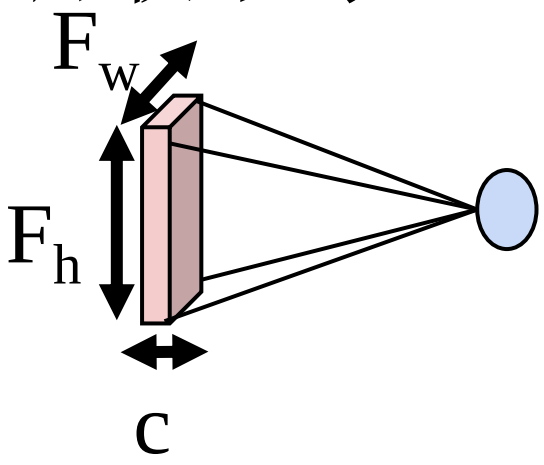
Slide credit: Karpathy and Fei-Fei

卷积网络

神经网络与卷积网络的 神经元 在处理数据的方式上是一致的: 加权平均



神经元



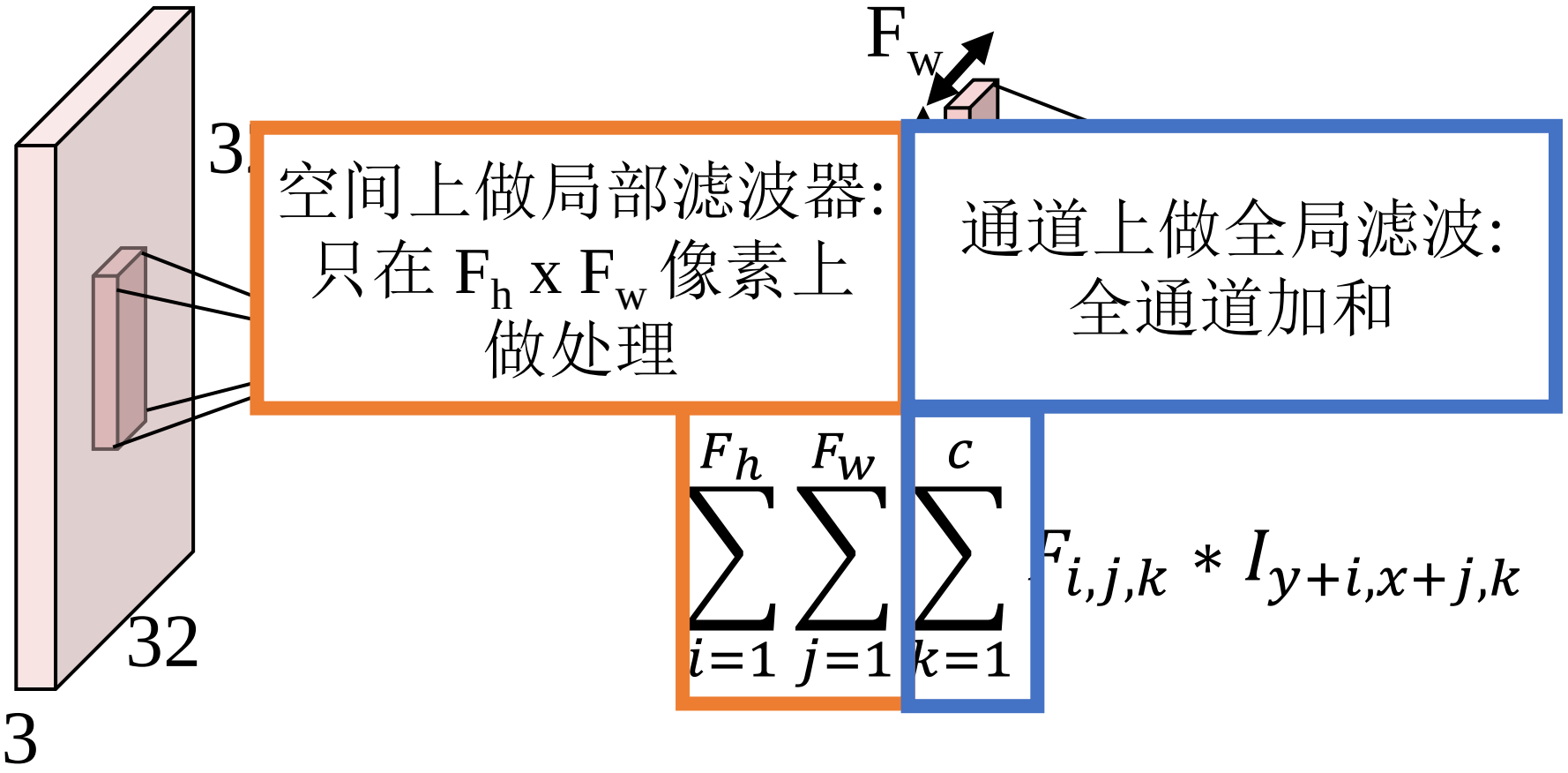
$$\sum_{i=1}^{F_h}\sum_{j=1}^{F_w}\sum_{k=1}^{c} F_{i,j,k} * I_{y+i,x+j,k}$$

Slide credit: Karpathy and Fei-Fei

卷积网络

神经元 在处理数据的方式上与神经网络是一致的:

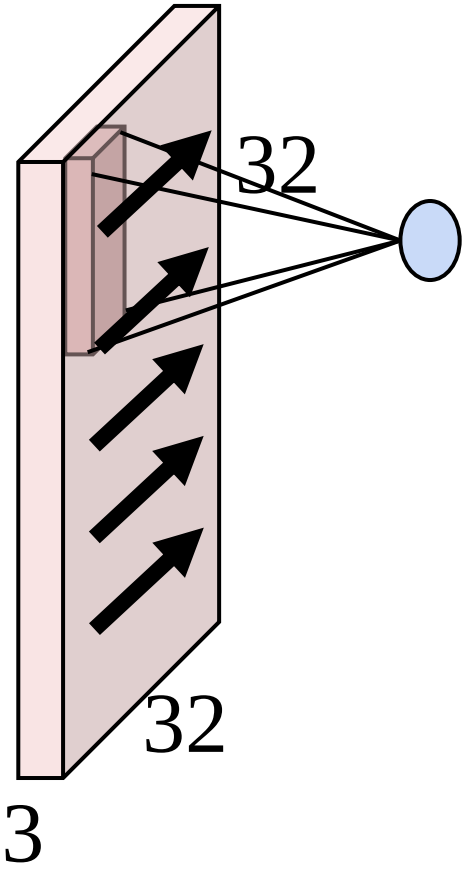
加权平均

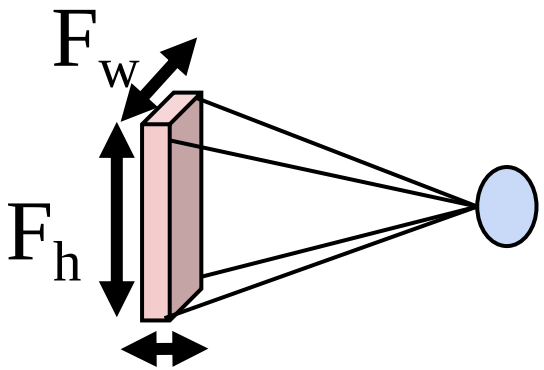


Slide credit: Karpathy and Fei-Fei

卷积网络

在图像上做滑动窗口卷积





$$\sum_{i=1}^{F_h}\sum_{j=1}^{F_w}\sum_{k=1}^{c} F_{i,j,k} * I_{y+i,x+j,k}$$

Slide credit: Karpathy and Fei-Fei

与之前提到的滤波的区别

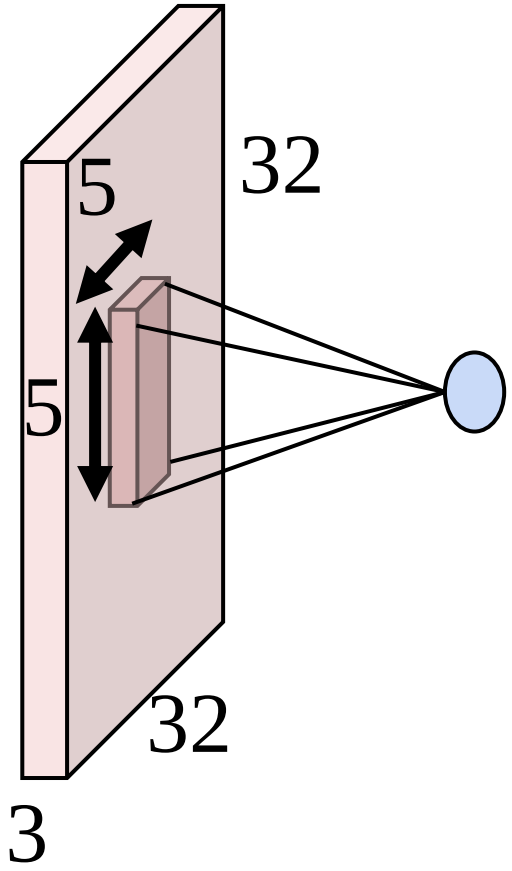
(a) 输入的维度会比1或3更大
(b) 忽略互相关核卷积的区别（实际操作中均使用矩阵乘法和互相关来实现卷积）

I11	F11	F12	F13	I15	I16
I21	F21	F22	F23	I25	I26
I31	F31	F32	F33	I35	I36
I41	I42	I43	I44	I45	I46
I51	I52	I53	I54	I55	I56

$$\text{Output}[1,2] = I[1,2]*F[1,1] + I[1,3]*F[1,2] + \ldots + I[3,4]*F[3,3]$$

卷积网络

输出有多大?



Height? 32-5+1=28

Width? 32-5+1=28

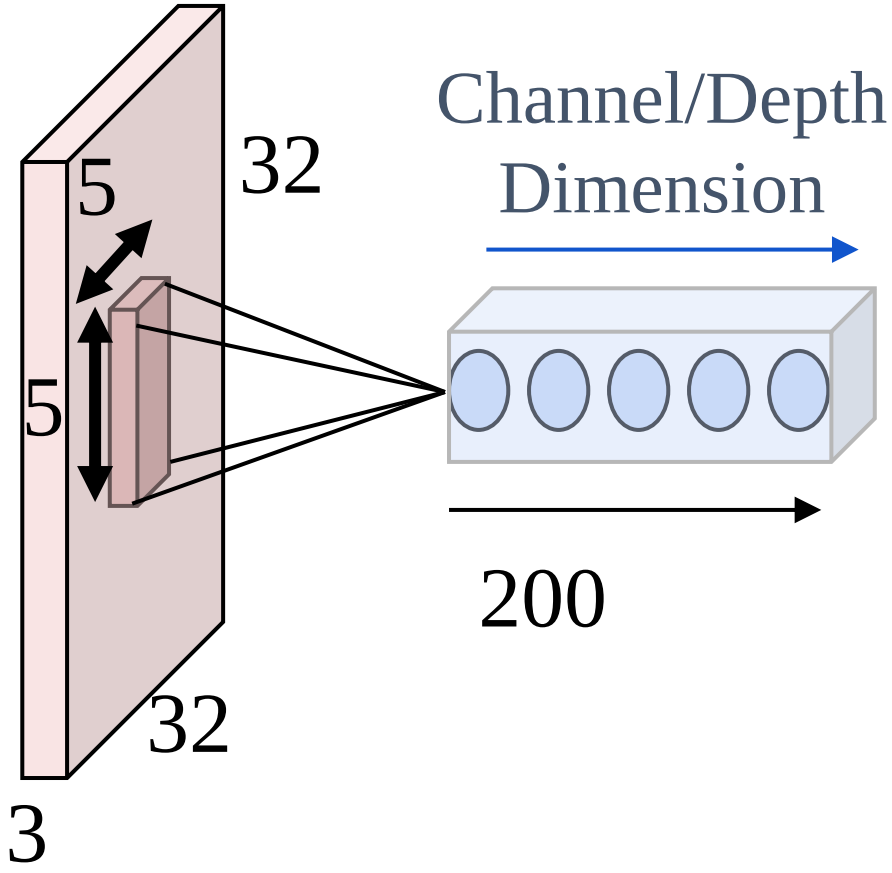
Channels? 1

Slide credit: Karpathy and Fei-Fei

卷积网络

多个channel 通道对应每个滤波器.

现在输出会有多大?



Height? 32-5+1=28

Width? 32-5+1=28

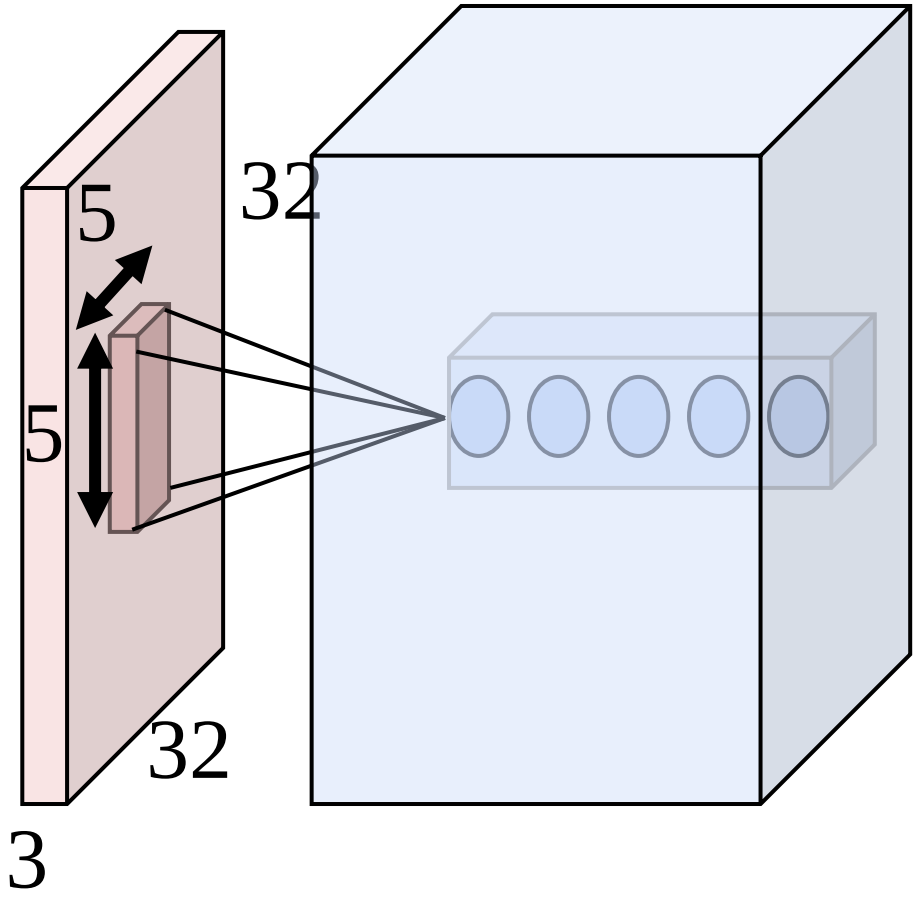
Channels? 200

Slide credit: Karpathy and Fei-Fei

卷积网络

多个channel 通道对应每个滤波器.

现在输出会有多大?



Height? 32-5+1=28

Width? 32-5+1=28

Channels? 200

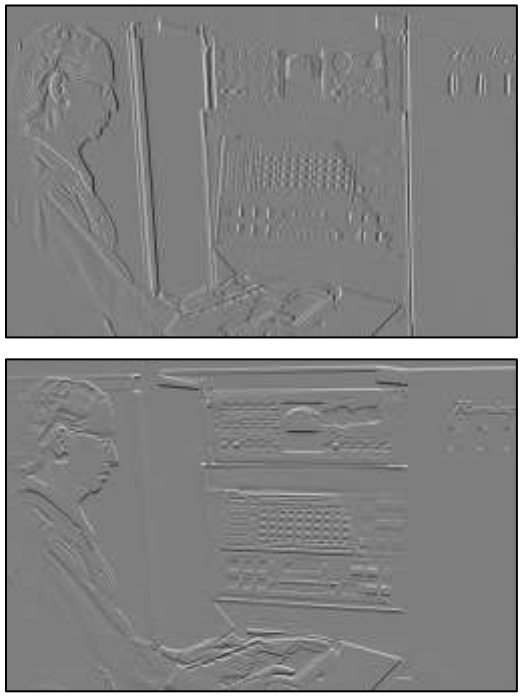
Slide credit: Karpathy and Fei-Fei

多个神经元的组合

Input: 400x600x1

Recall：图像梯度 怎样用卷积层实现？用多少参数？

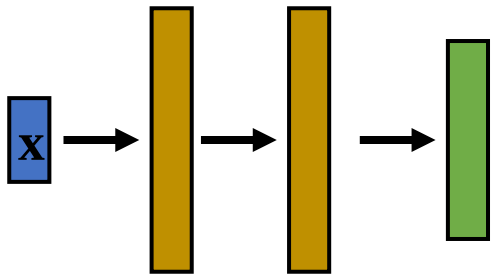
Output:
400x600x2

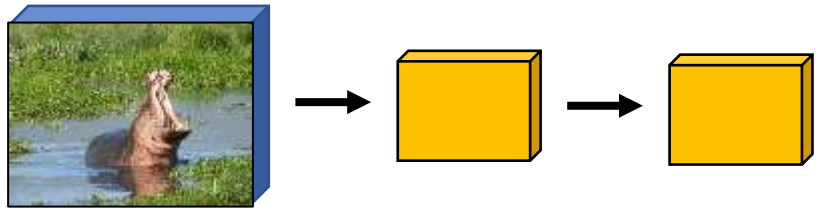


卷积网络 vs 全连接网络

全连接网络:
由矩阵参数/向量参数 **W,b**
+ 非线性激活层
梯度下降、方向传播优化

卷积网络:
由卷积滤波/向量参数 F,b +
非线性激活层
梯度下降、方向传播优化





一个新的参数 Stride 步长

输出的空间尺度是多大?

F11	F12	F13	I14	I15	I16	I17
F21	F22	F23	I24	I25	I26	I27
F31	F32	F33	I34	I35	I36	I37
I41	I42	I43	I44	I45	I46	I47
I51	I52	I53	I54	I55	I56	I57
I61	I62	I63	I64	I65	I66	I67
I71	I72	I73	I74	I75	I76	I77

普通情况(Stride 1):
5x5 output

Example credit: Karpathy and Fei-Fei

Stride 步长

Stride = 2:跳过一些窗口

F11	F12	F13	I14	I15	I16	I17
F21	F22	F23	I24	I25	I26	I27
F31	F32	F33	I34	I35	I36	I37
I41	I42	I43	I44	I45	I46	I47
I51	I52	I53	I54	I55	I56	I57
I61	I62	I63	I64	I65	I66	I67
I71	I72	I73	I74	I75	I76	I77

普通情况(Stride 1):
5x5 output

Example credit: Karpathy and Fei-Fei

Stride 步长

Stride = 2:跳过一些窗口

I11	I12	F11	F12	F13	I16	I17
I21	I22	F21	F22	F23	I26	I27
I31	I32	F31	F32	F33	I36	I37
I41	I42	I43	I44	I45	I46	I47
I51	I52	I53	I54	I55	I56	I57
I61	I62	I63	I64	I65	I66	I67
I71	I72	I73	I74	I75	I76	I77

普通情况(Stride 1):
5x5 output

Example credit: Karpathy and Fei-Fei

Stride 步长

Stride = 2:跳过一些窗口

I11	I12	I13	I14	F11	F12	F13
I21	I22	I23	I24	F21	F22	F23
I31	I32	I33	I34	F31	F32	F33
I41	I42	I43	I44	I45	I46	I47
I51	I52	I53	I54	I55	I56	I57
I61	I62	I63	I64	I65	I66	I67
I71	I72	I73	I74	I75	I76	I77

普通情况(Stride 1):
5x5 output

Stride = 2 :
3x3 output

Example credit: Karpathy and Fei-Fei

Stride 步长

如果stride=3呢

F11	F12	F13	I14	I15	I16	I17
F21	F22	F23	I24	I25	I26	I27
F31	F32	F33	I34	I35	I36	I37
I41	I42	I43	I44	I45	I46	I47
I51	I52	I53	I54	I55	I56	I57
I61	I62	I63	I64	I65	I66	I67
I71	I72	I73	I74	I75	I76	I77

普通情况(Stride 1):
5x5 output

Stride 2 convolution:
3x3 output

Example credit: Karpathy and Fei-Fei

Stride 步长

如果stride=3呢

I11	I12	I13	F11	F12	F13	I17
I21	I22	I23	F21	F22	F23	I27
I31	I32	I33	F31	F32	F33	I37
I41	I42	I43	I44	I45	I46	I47
I51	I52	I53	I54	I55	I56	I57
I61	I62	I63	I64	I65	I66	I67
I71	I72	I73	I74	I75	I76	I77

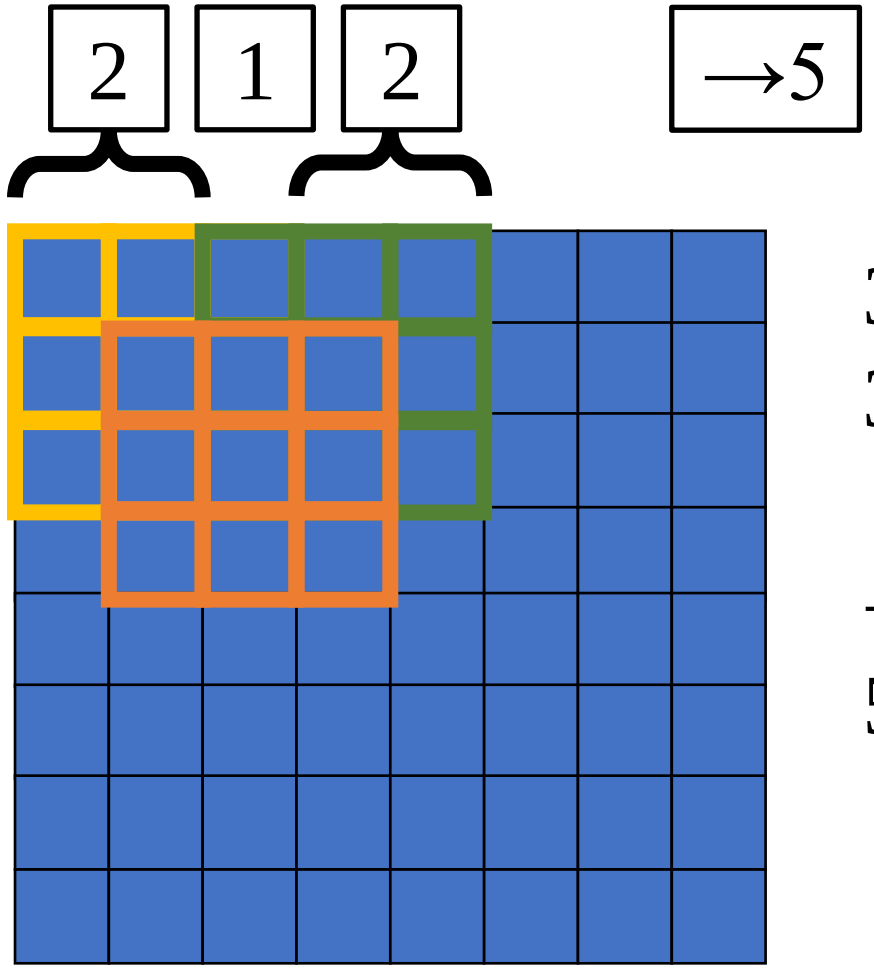
普通情况(Stride 1):
5x5 output

Stride 2 convolution:
3x3 output

Stride 3 convolution:
有问题! 为什么？

Example credit: Karpathy and Fei-Fei

3x3 滤波

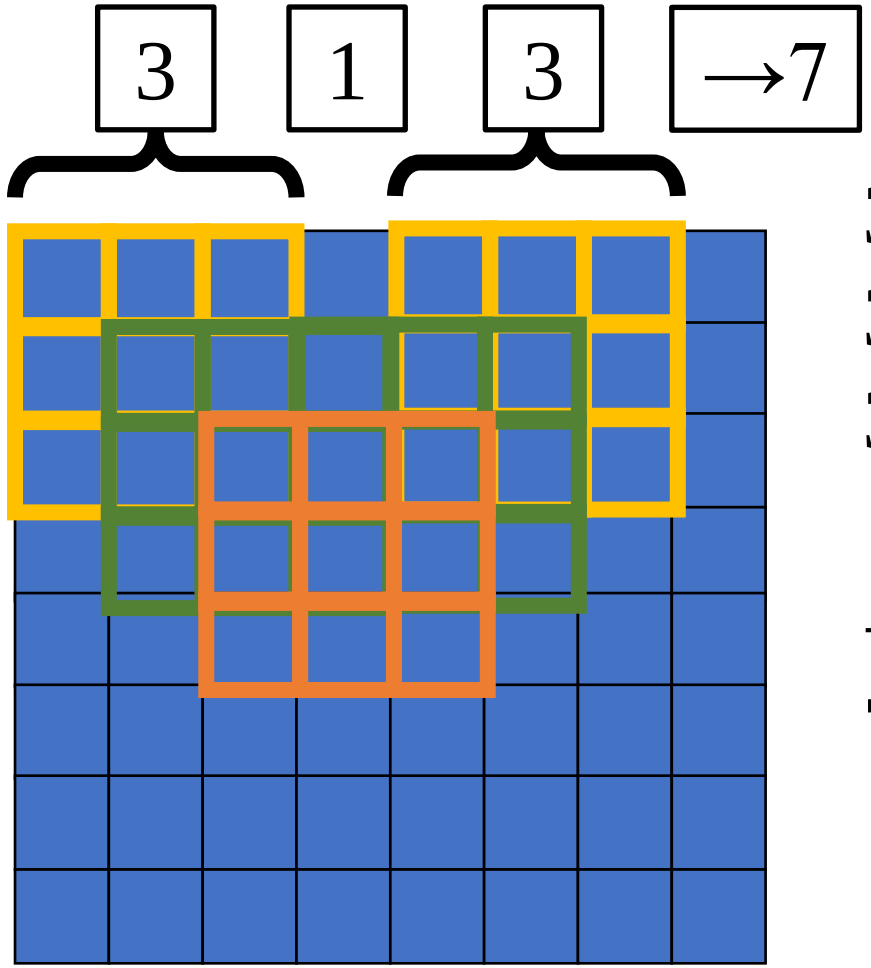


3x3 滤波 接一个
3x3 滤波

→

5x5 感受野的滤波

3x3 滤波



3x3 滤波 接一个
3x3 滤波 接一个
3x3 滤波

→

7x7 感受野的滤波

Stride 步长

如果stride=3呢

I11	I12	I13	F11	F12	F13	I17
I21	I22	I23	F21	F22	F23	I27
I31	I32	I33	F31	F32	F33	I37
I41	I42	I43	I44	I45	I46	I47
I51	I52	I53	I54	I55	I56	I57
I61	I62	I63	I64	I65	I66	I67
I71	I72	I73	I74	I75	I76	I77

普通情况(Stride 1):
5x5 output

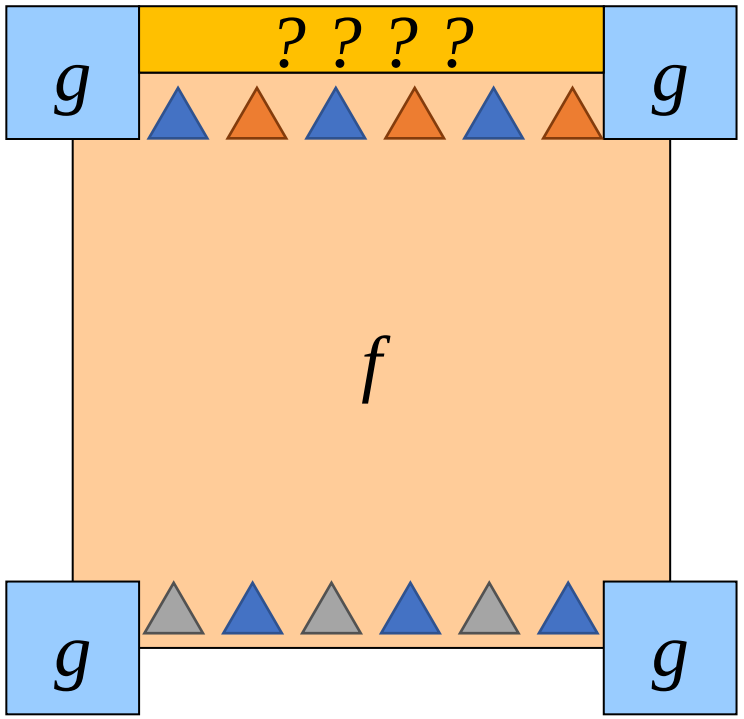
Stride 2 convolution:
3x3 output

Stride 3 convolution:
感受野无法扩大

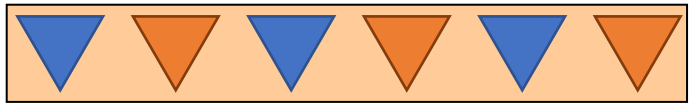
Example credit: Karpathy and Fei-Fei

Padding

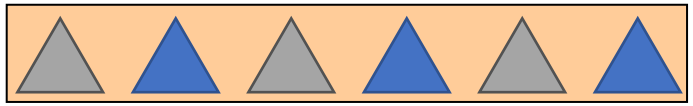
Padding用于填充外围缺失区域，形式有很多种



Symm: fold sides over

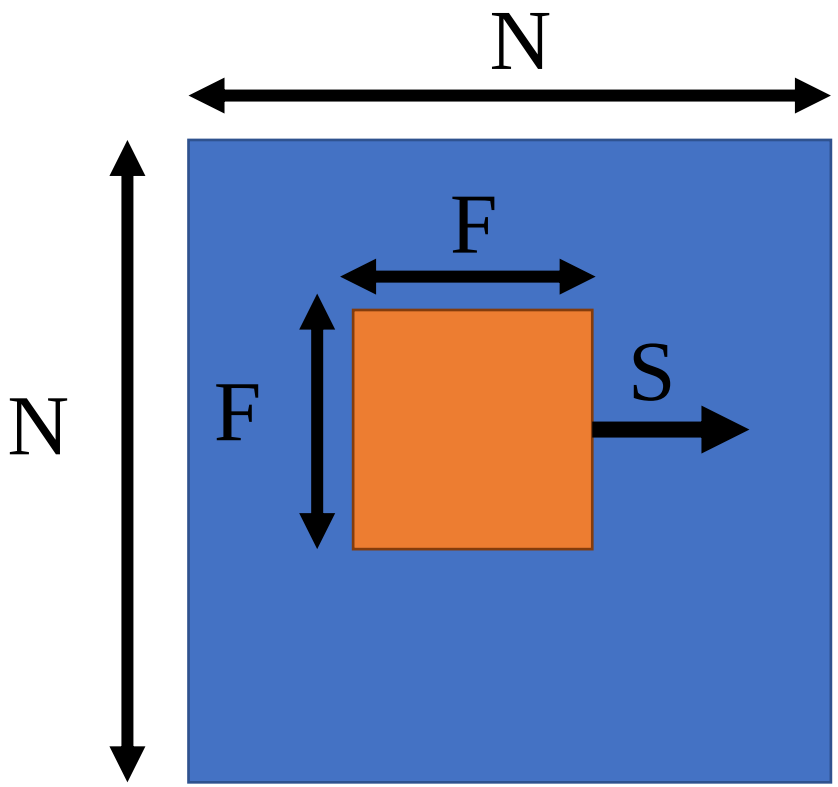


Circular/Wrap: wrap around



pad/fill: add value, often 0

总的来说…



Output Size

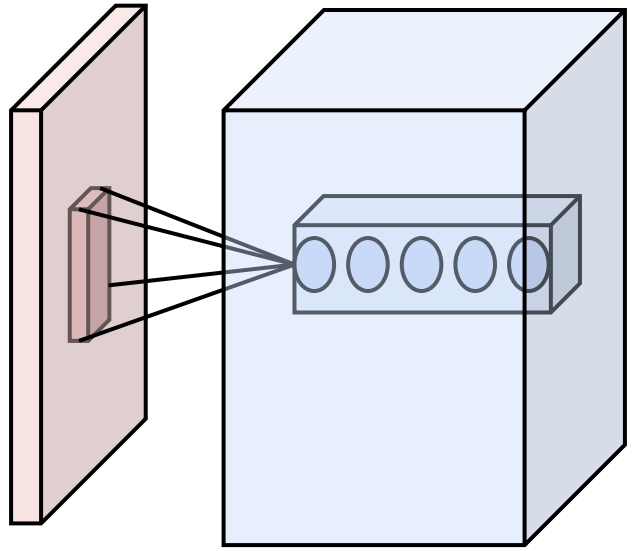
$$\frac{(N-F)}{S}+1$$

Slide credit: Karpathy and Fei-Fei

计算题

输入大小: **32x32x3**
感受野/卷积核大小: **5x5, stride 1**
神经元数量: **5**

$$\frac{(N-F)}{s}+1$$

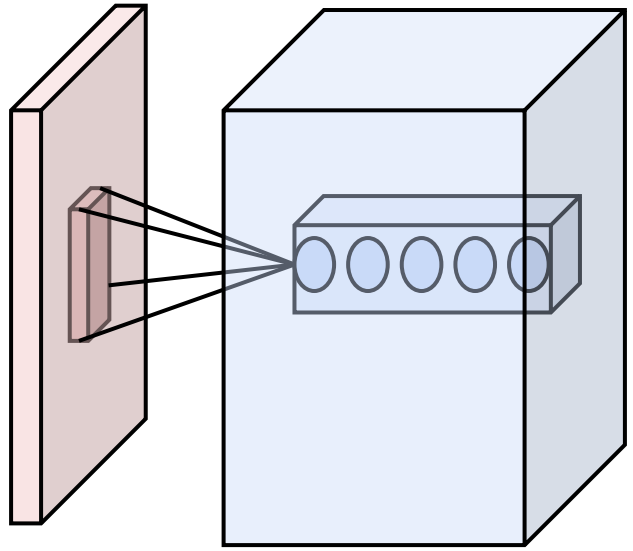


输出大小?

Slide credit: Karpathy and Fei-Fei

计算题

输入大小: **32x32x3**
感受野/卷积核大小: **5x5, stride 1**
神经元数量: **5**



$$\frac{(N-F)}{s}+1$$

输出尺寸: (32 - 5) / 1 + 1 = 28, 大小为: **28x28x5**

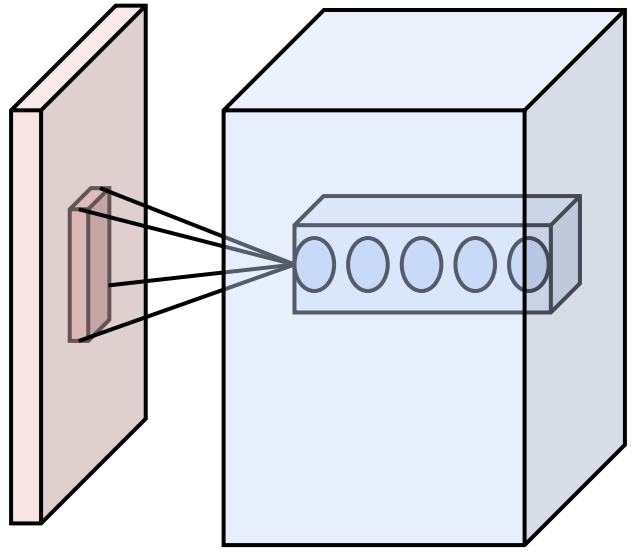
参数量?

Slide credit: Karpathy and Fei-Fei

计算题

输入大小: **32x32x3**
感受野/卷积核大小: **5x5, stride 1**
神经元数量: **5**

$$\frac{(N-F)}{s}+1$$



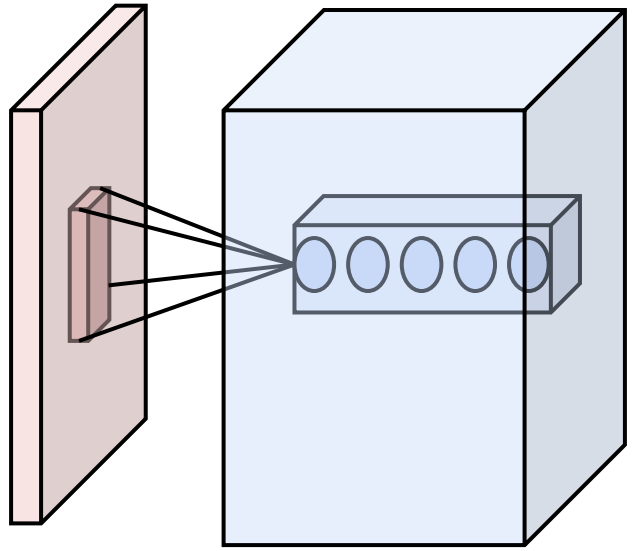
输出尺寸: (32 - 5) / 1 + 1 = 28, 大小为: **28x28x5**
参数量： **5x5x3x5 + 5 = 380**

Slide credit: Karpathy and Fei-Fei

计算题

输入大小: **32x32x3**
感受野/卷积核大小: **5x5, stride 3**
神经元数量: **5**

$$\frac{(N-F)}{s}+1$$

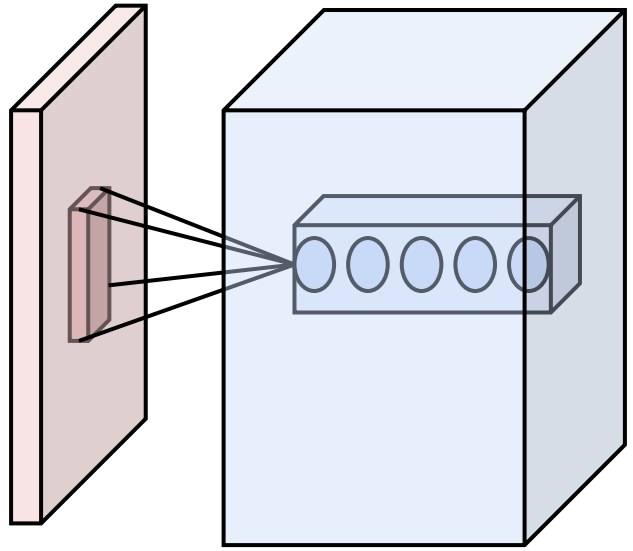


输出大小?

Slide credit: Karpathy and Fei-Fei

计算题

输入大小: **32x32x3**
感受野/卷积核大小: **5x5, stride 3**
神经元数量: **5**



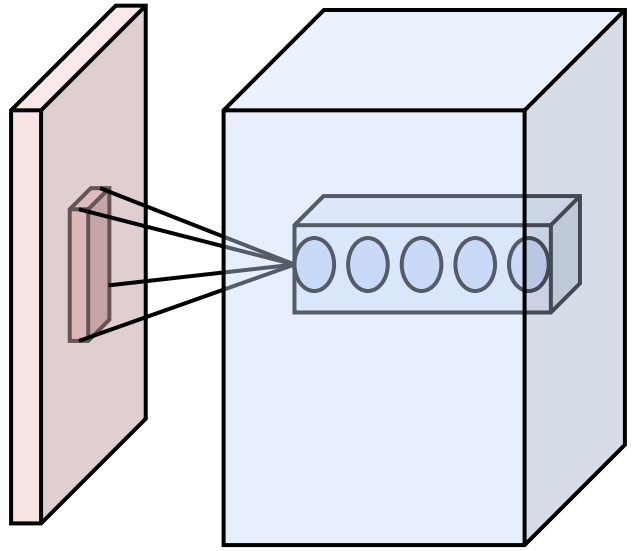
输出尺寸: (32 - 5) / 3 + 1 = 10, 输出大小: **10x10x5**

参数量?

Slide credit: Karpathy and Fei-Fei

计算题

输入大小: **32x32x3**
感受野/卷积核大小: **5x5, stride 3**
神经元数量: **5**

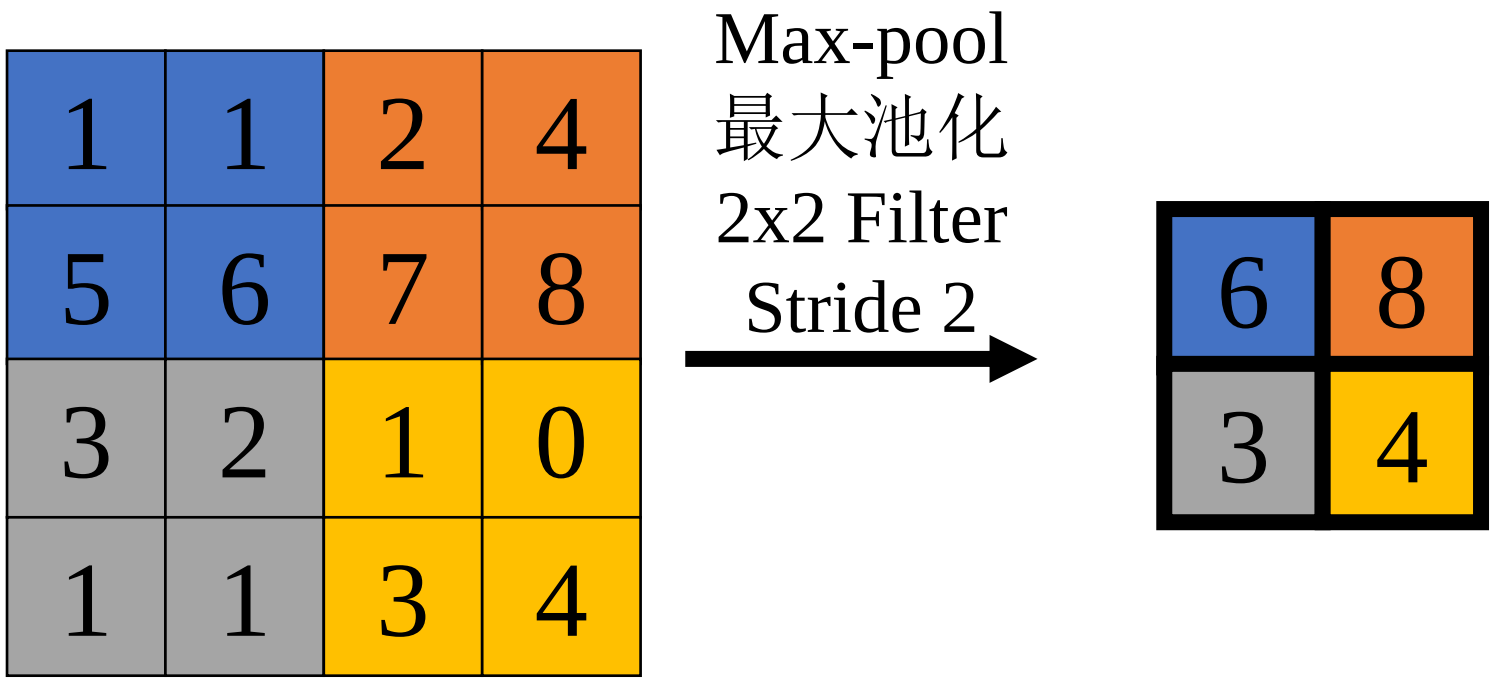


输出尺寸: (32 - 5) / 3 + 1 = 10, 输出大小: **10x10x5**
参数量： **5x5x3x5 + 5 = 380. 与之前相同!**

Slide credit: Karpathy and Fei-Fei

Pooling “池化”

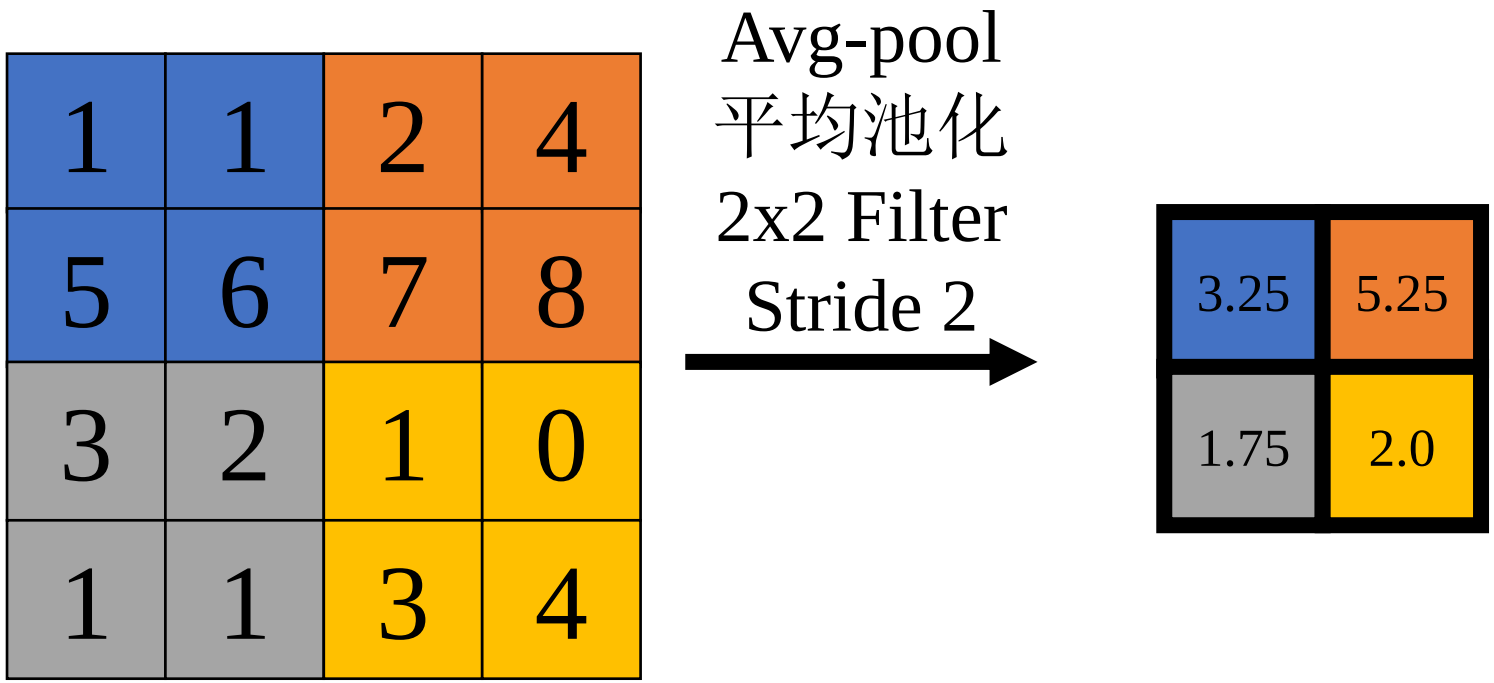
出发点：使得特征的空间分辨率变小;
在每个通道上应用



Slide credit: Karpathy and Fei-Fei

Pooling “池化”

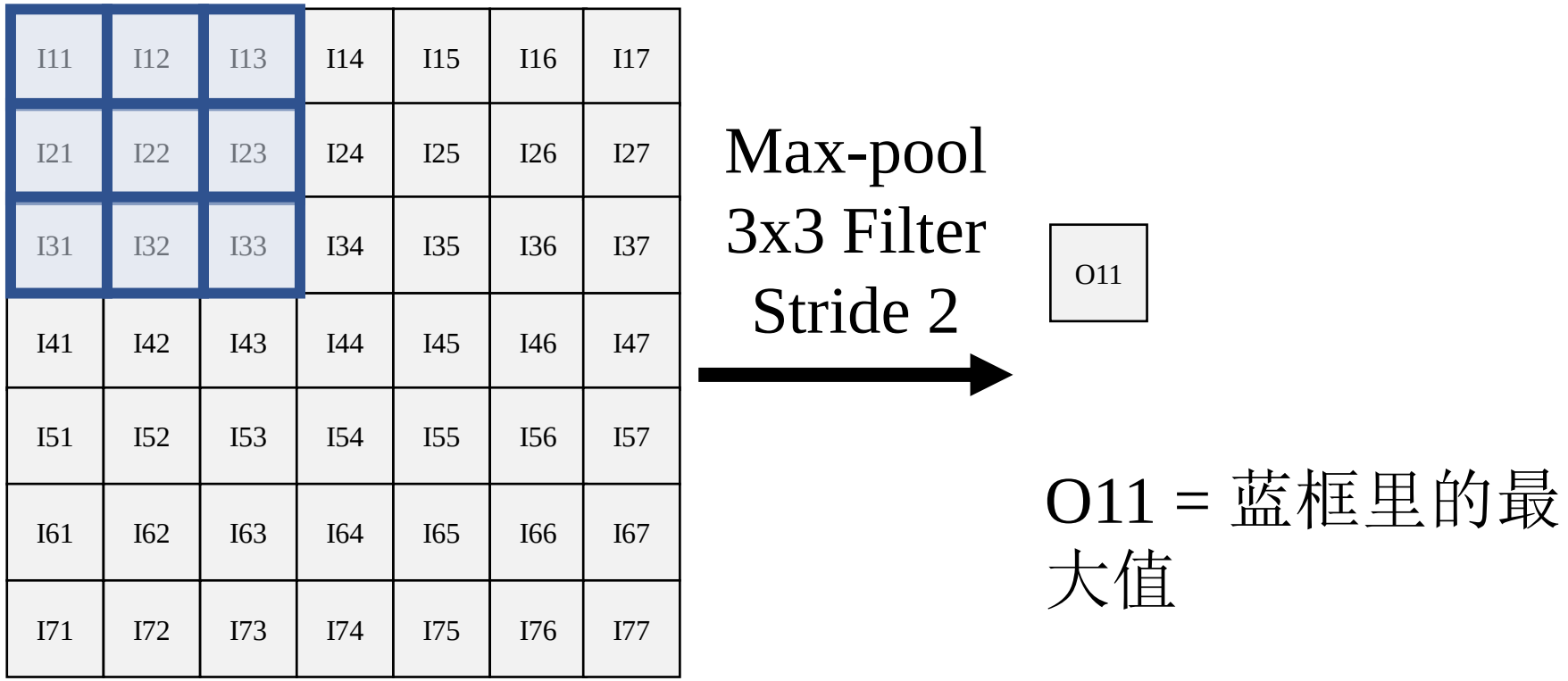
出发点：使得特征的空间分辨率变小;
在每个通道上应用



Slide credit: Karpathy and Fei-Fei

Pooling “池化” —— 滤波实现

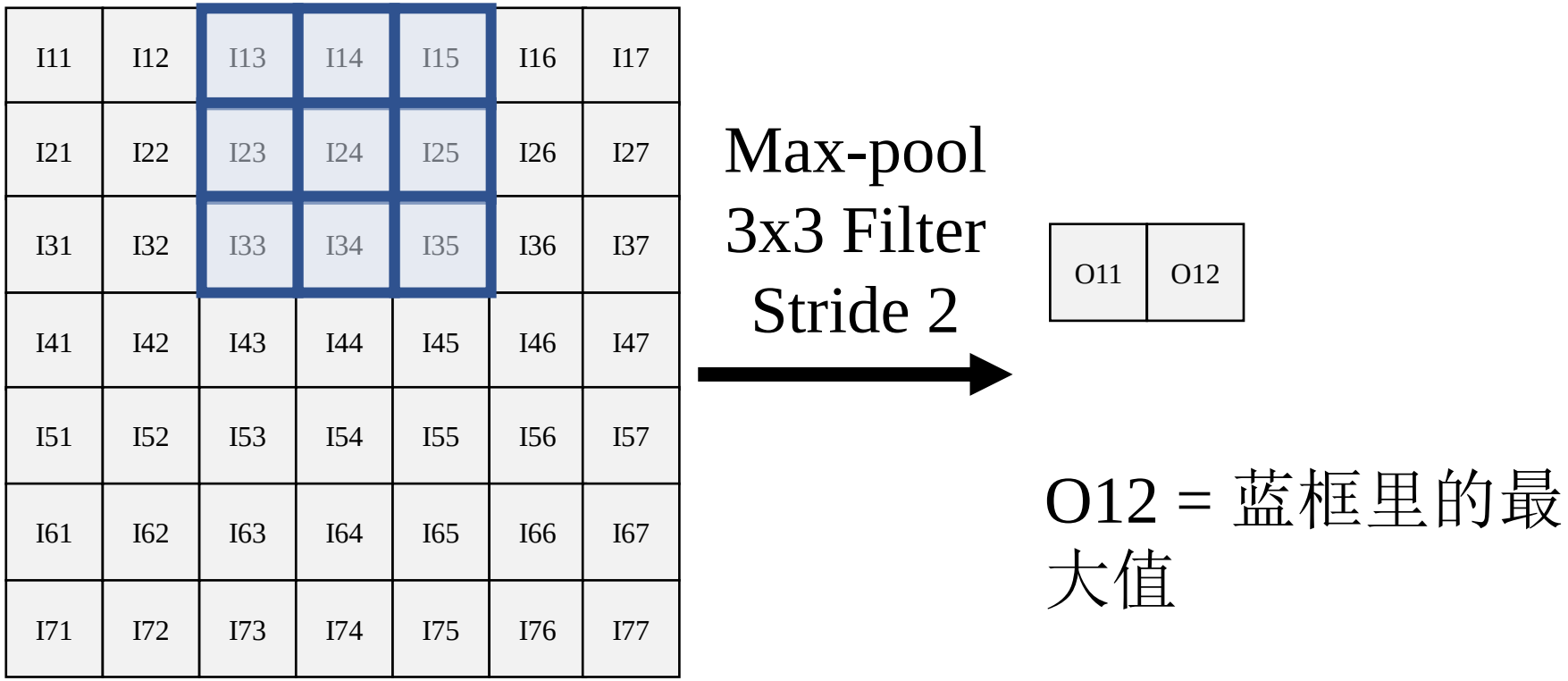
使得特征的空间分辨率变小;
在每个通道上应用



O11 = 蓝框里的最大值

Pooling “池化” —— 滤波实现

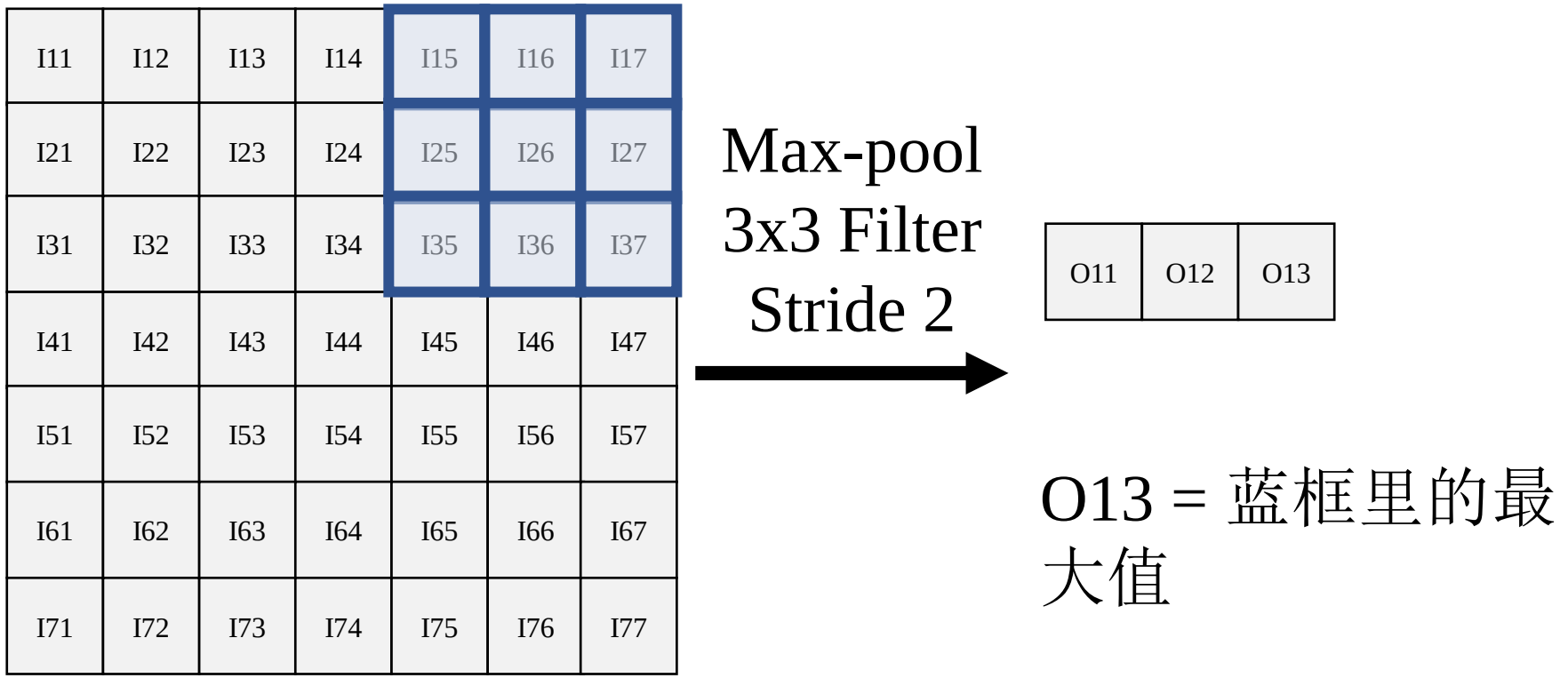
使得特征的空间分辨率变小;
在每个通道上应用



O12 = 蓝框里的最大值

Pooling “池化” —— 滤波实现

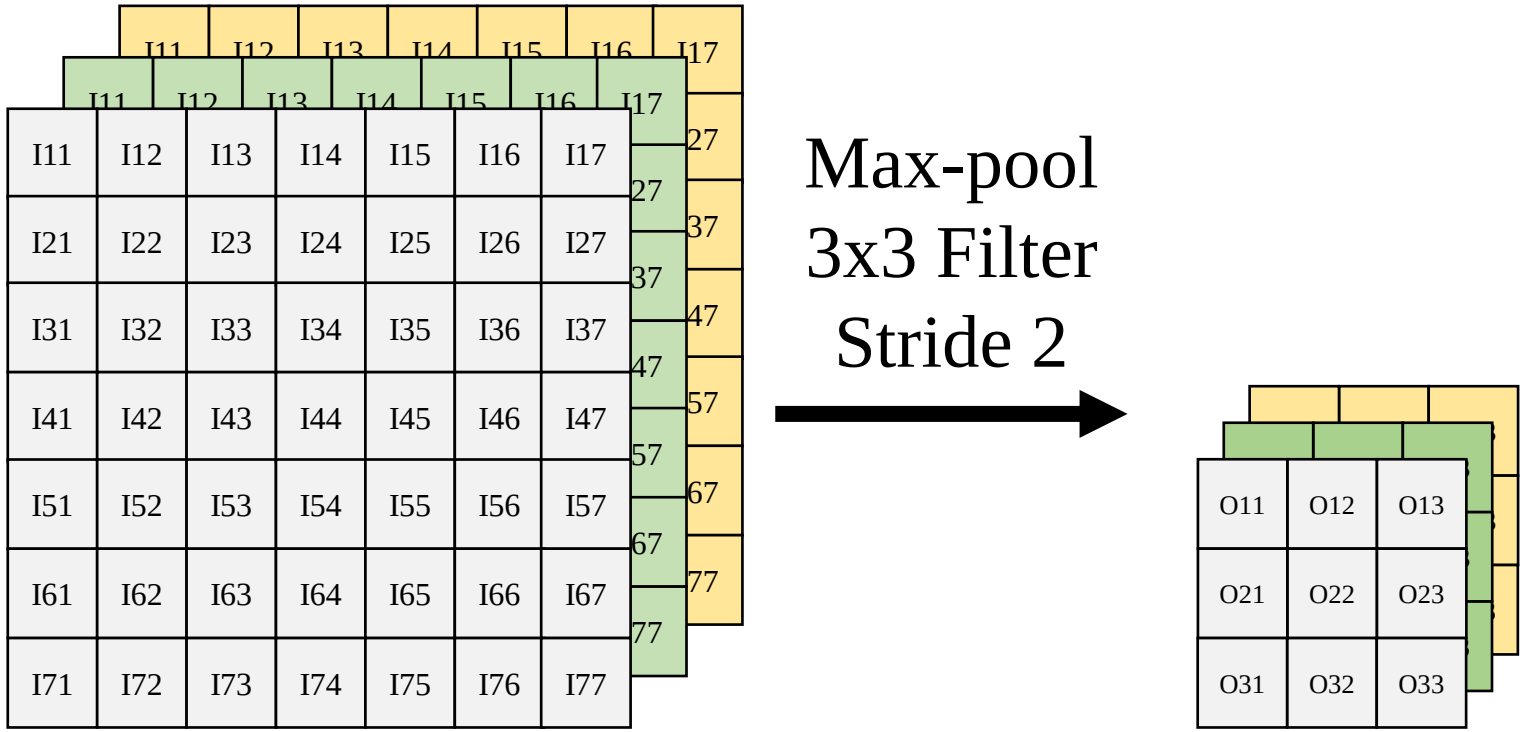
使得特征的空间分辨率变小;
在每个通道上应用



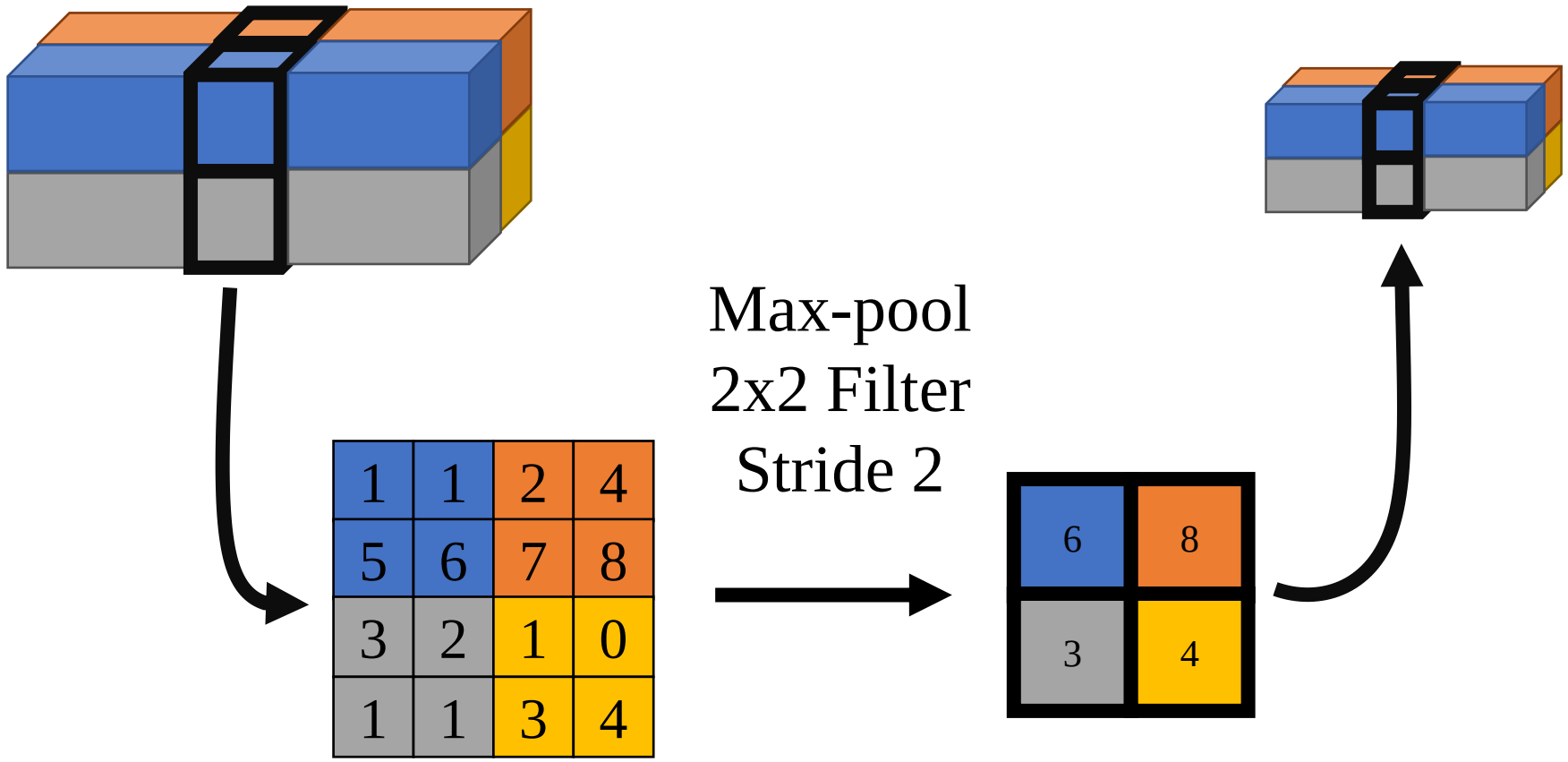
O13 = 蓝框里的最大值

Pooling “池化” —— 滤波实现

使得特征的空间分辨率变小;
在每个通道上应用

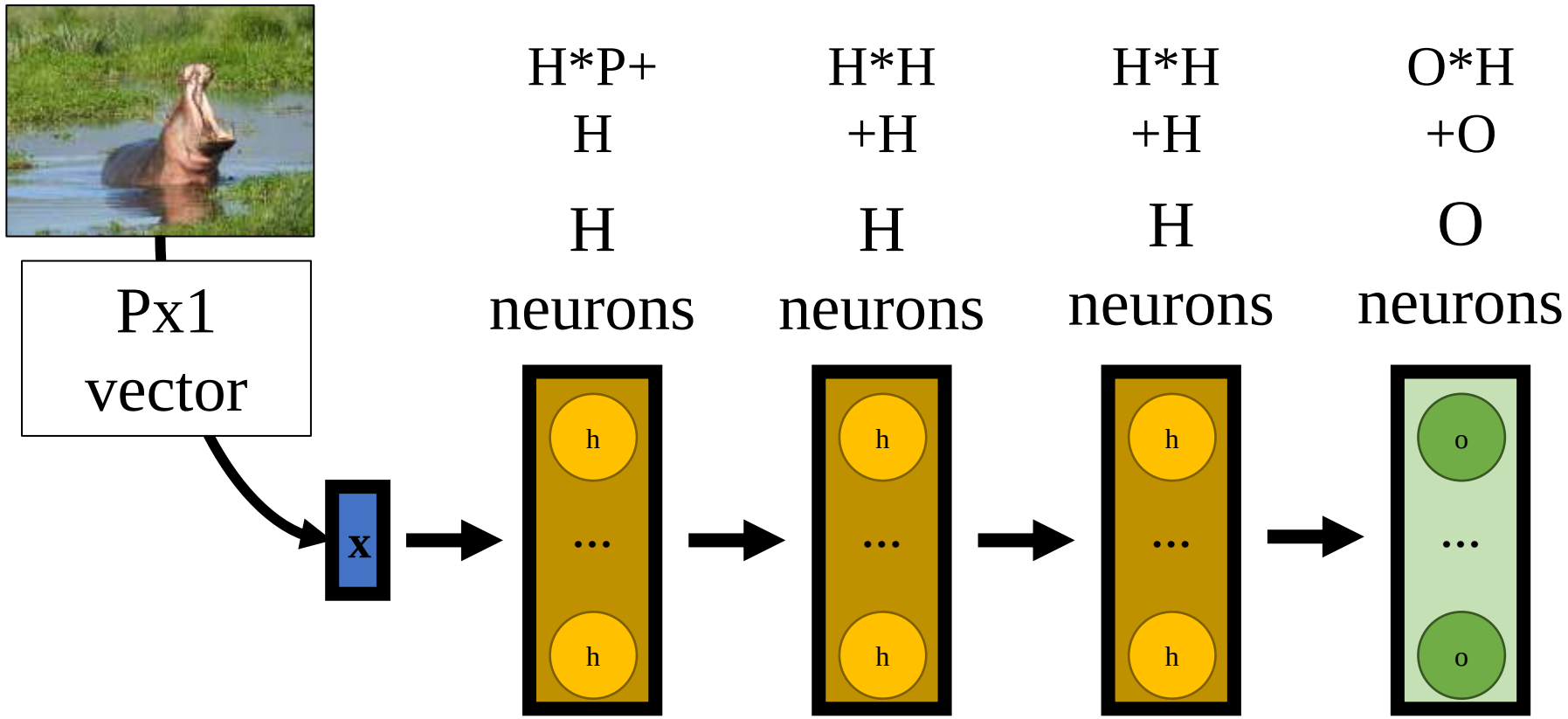


Pooling 前后对比



回顾——卷积神经网络

参数量



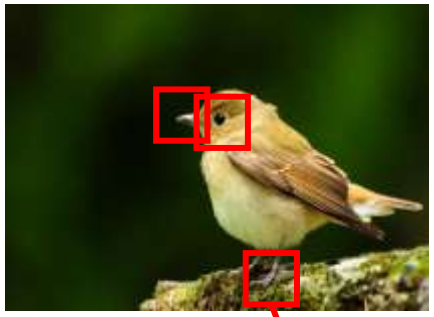
P: 285x350 , H: 1000, O: 3

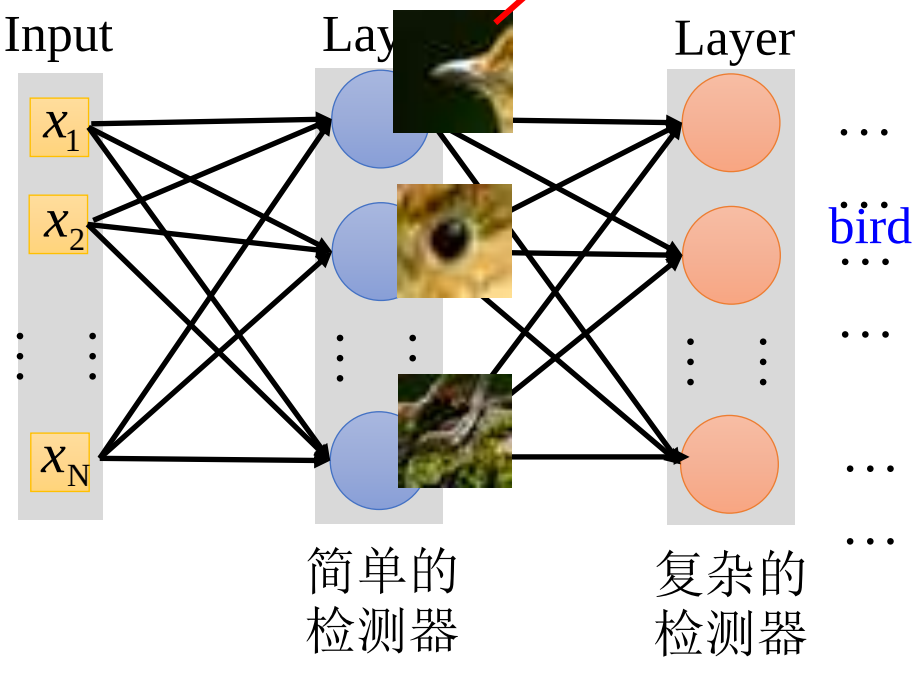
102 million 个 参数 (400MB)

简化全连接层

每个神经元并不需要看到所有像素点（回顾特征尺度）

我们需要看整张图来分类吗？（回顾特征点）



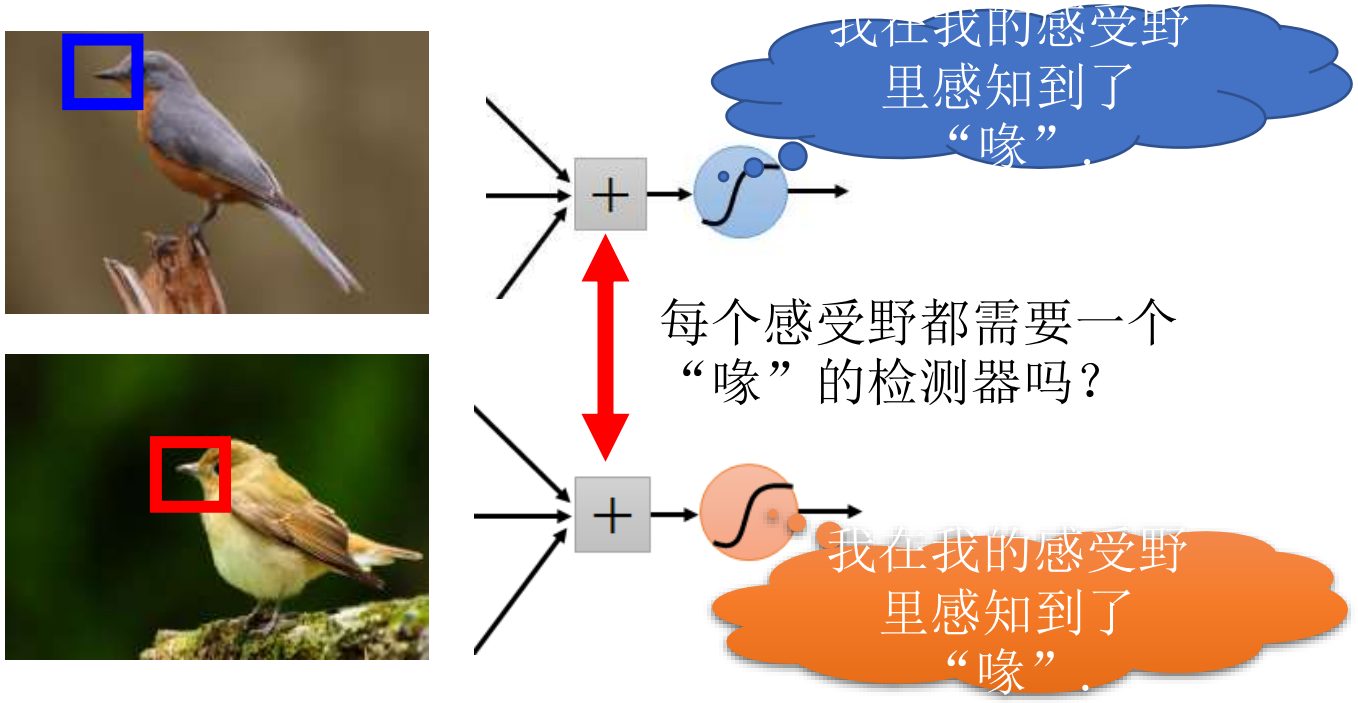


有些关键特征也许很小（回顾特征尺度）

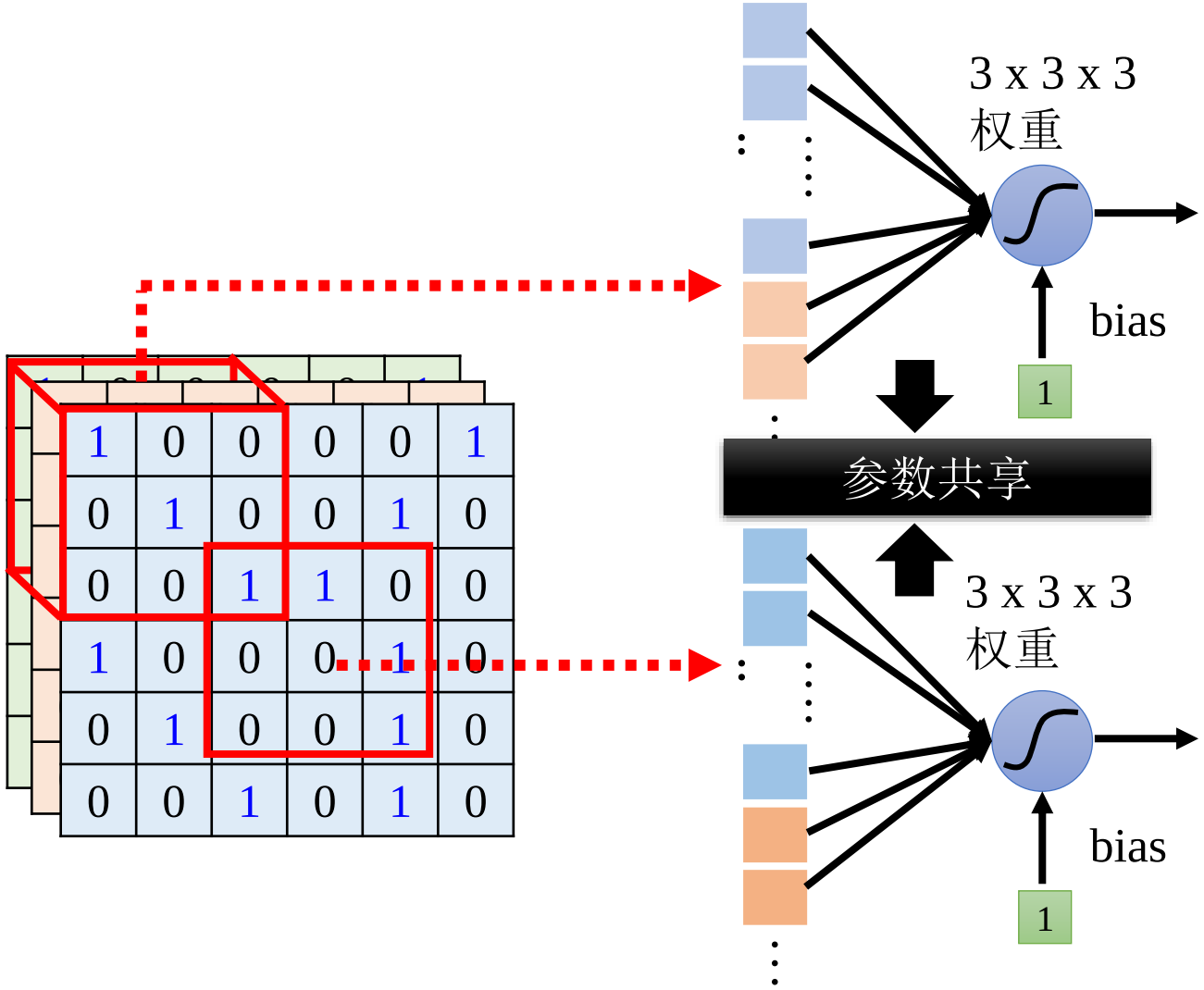
神经元通常只对图像中的局部模式或特征进行响应，而不是整张图像。

简化全连接层

- 相同的特征会出现在不同的区域.



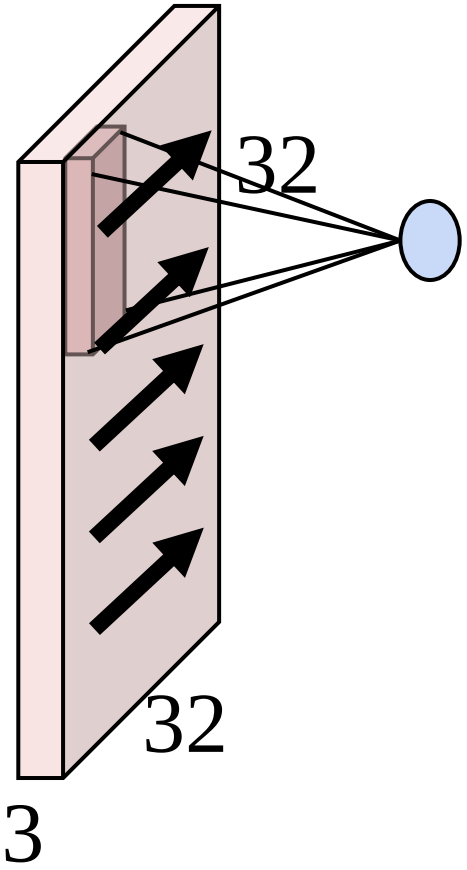
简化全连接层的关键——参数共享

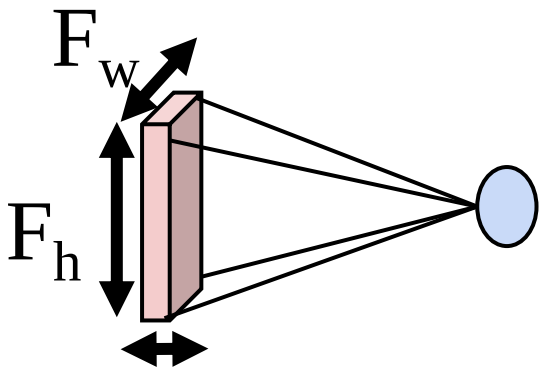


- 只需要一个3x3x3的权重矩阵，就可以在整张图像上检测特征。通过参数共享，卷积神经网络可以大大减少所需的参数数量，同时确保相同的特征在不同的位置都能被检测到。

卷积网络

在图像上做滑动窗口卷积





$$\sum_{i=1}^{F_h}\sum_{j=1}^{F_w}\sum_{k=1}^{c} F_{i,j,k} * I_{y+i,x+j,k}$$

Slide credit: Karpathy and Fei-Fei

Stride 步长

如果stride=3呢

F11	F12	F13	I14	I15	I16	I17
F21	F22	F23	I24	I25	I26	I27
F31	F32	F33	I34	I35	I36	I37
I41	I42	I43	I44	I45	I46	I47
I51	I52	I53	I54	I55	I56	I57
I61	I62	I63	I64	I65	I66	I67
I71	I72	I73	I74	I75	I76	I77

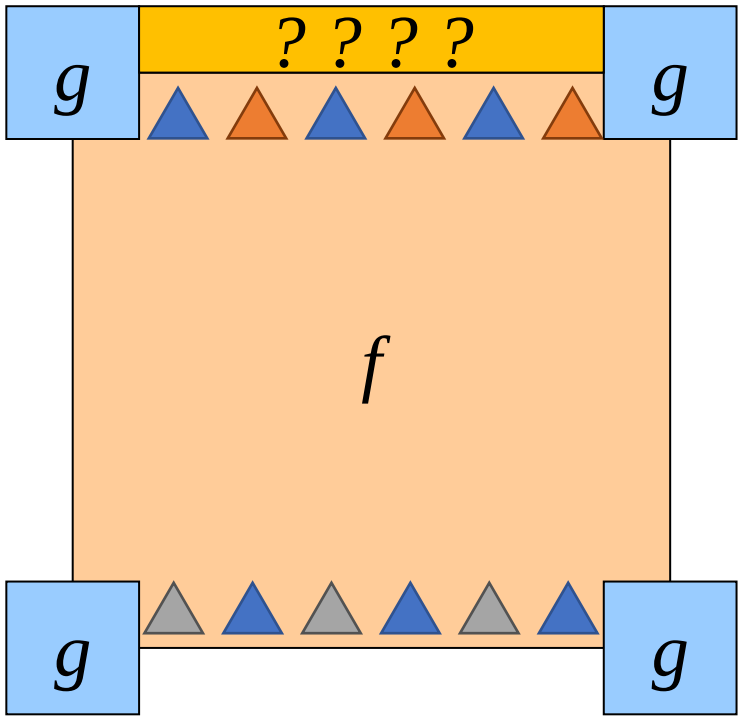
普通情况(Stride 1):
5x5 output

Stride 2 convolution:
3x3 output

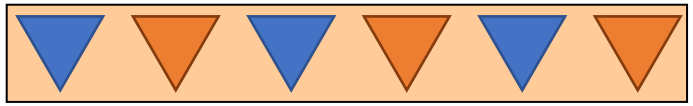
Example credit: Karpathy and Fei-Fei

Padding

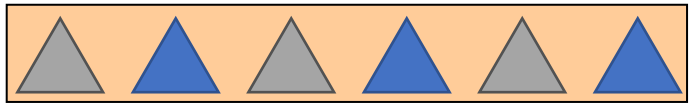
Padding用于填充外围缺失区域，形式有很多种



Symm: fold sides over



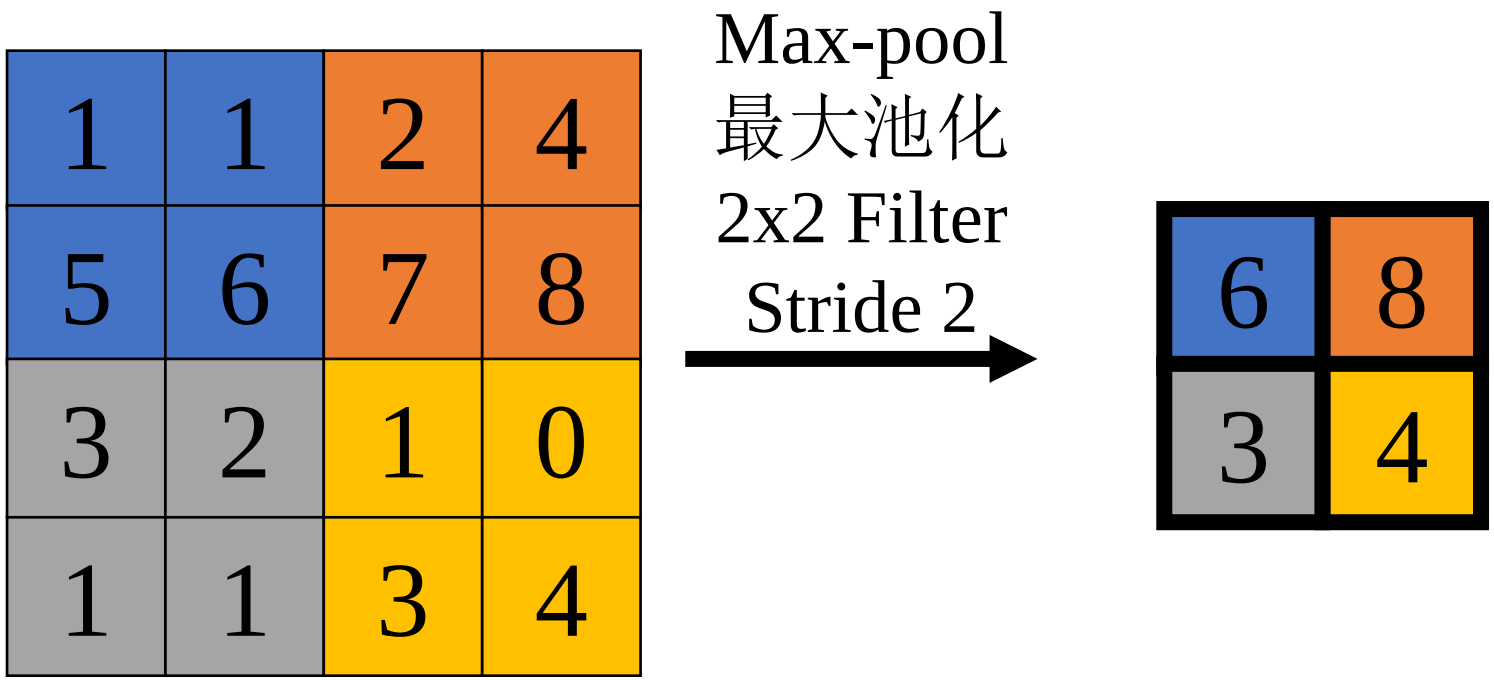
Circular/Wrap: wrap around



pad/fill: add value, often 0

Pooling “池化”

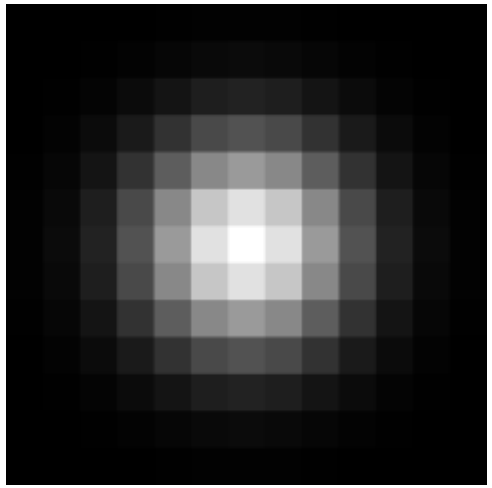
出发点：使得特征的空间分辨率变小;
在每个通道上应用

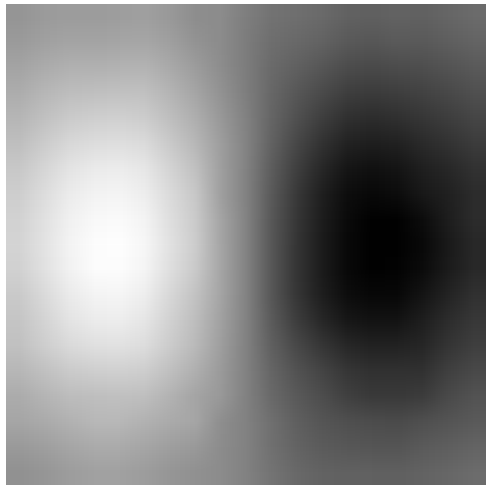


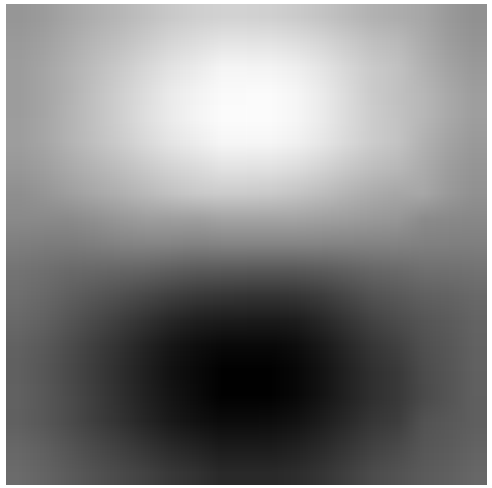
Slide credit: Karpathy and Fei-Fei

回顾滤波

还记得这些滤波在做什么吗？







我们可以可视化滤波来推断他们的功能

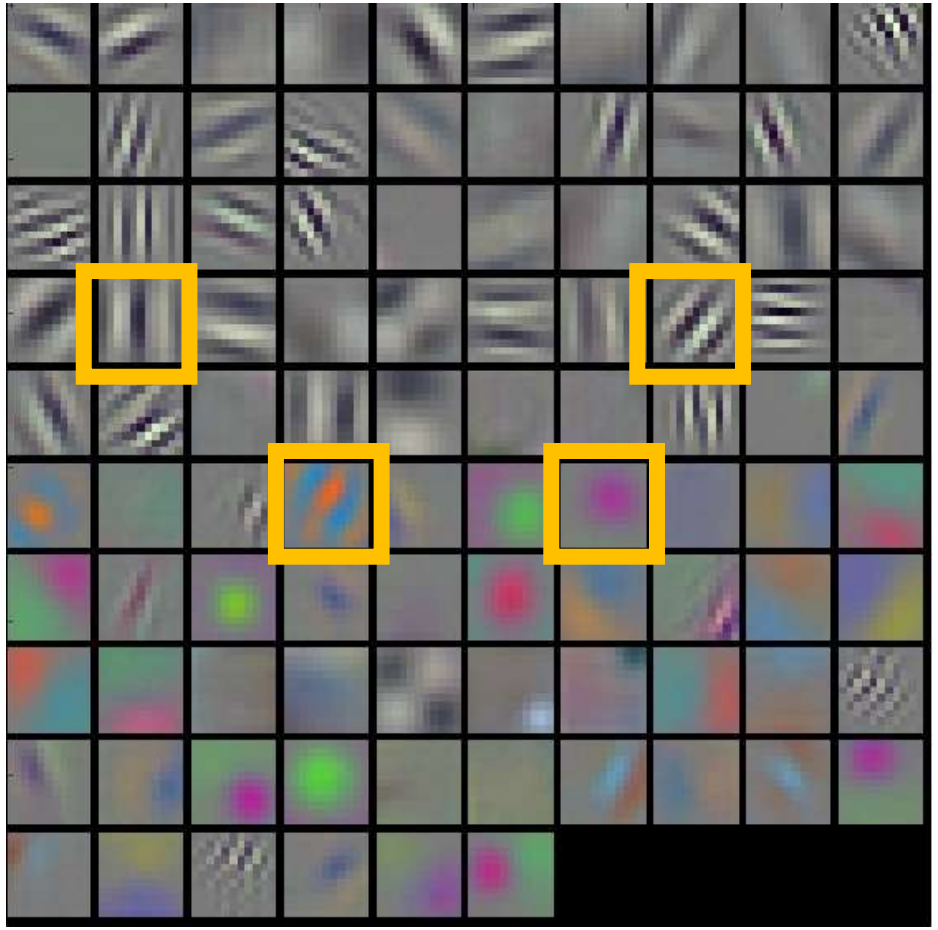


Figure Credit: Karpathy and Fei-Fei

每一层的输出

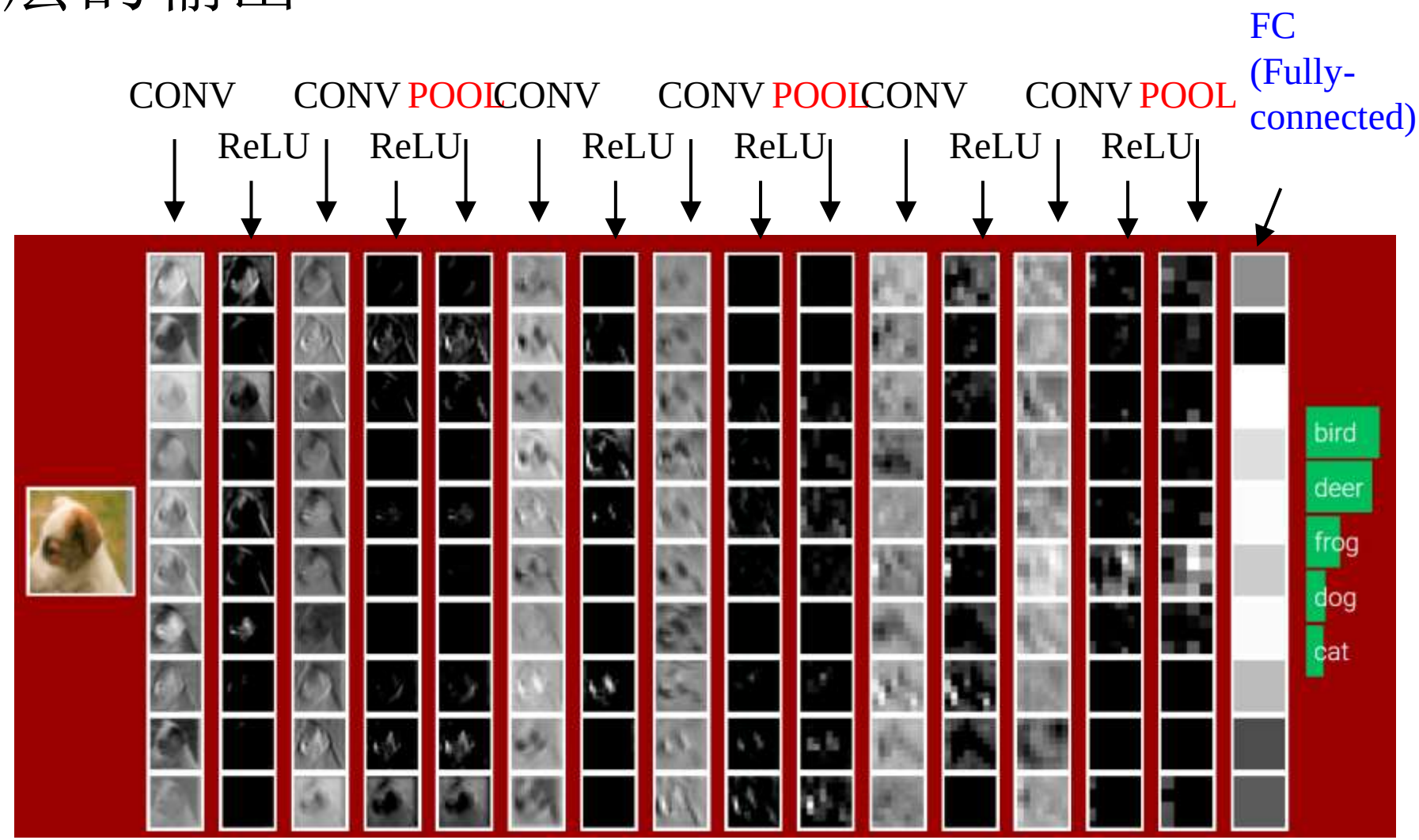
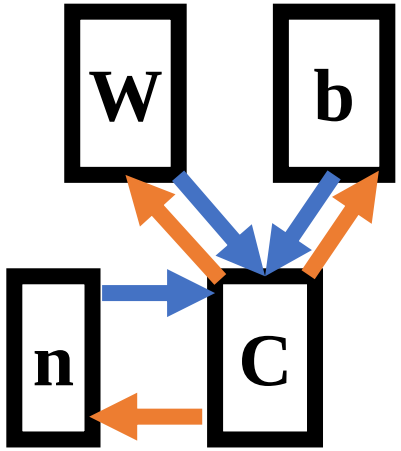


Figure Credit: Karpathy and Fei-Fei; see http://cs231n.stanford.edu/

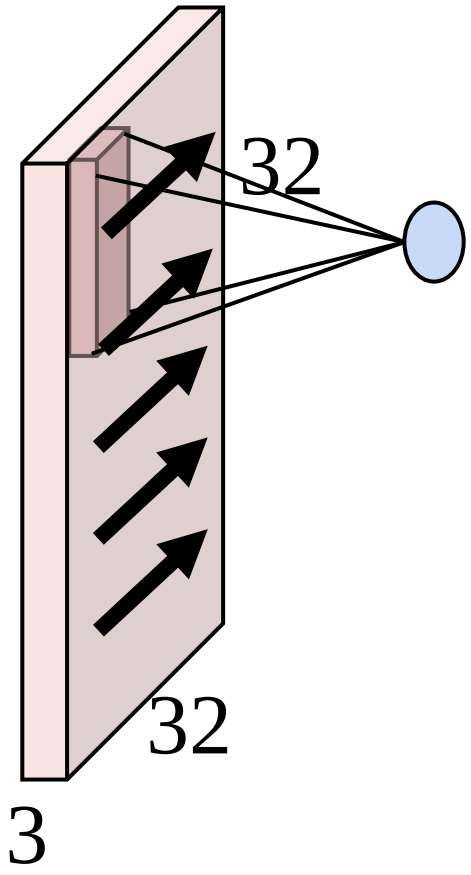
卷积层

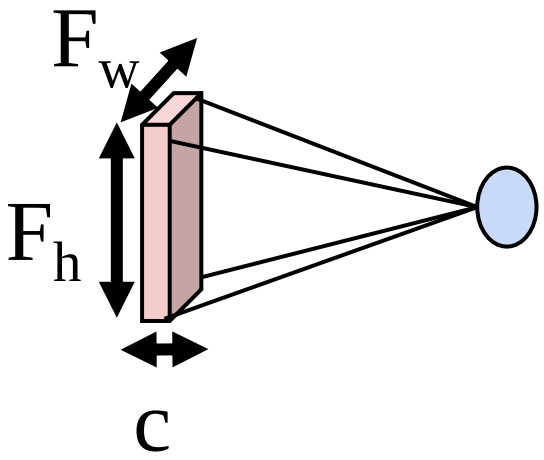
新的网络模块：2DConv



$$C(\boldsymbol{n}) = \boldsymbol{n} * F + \boldsymbol{b}$$

卷积层

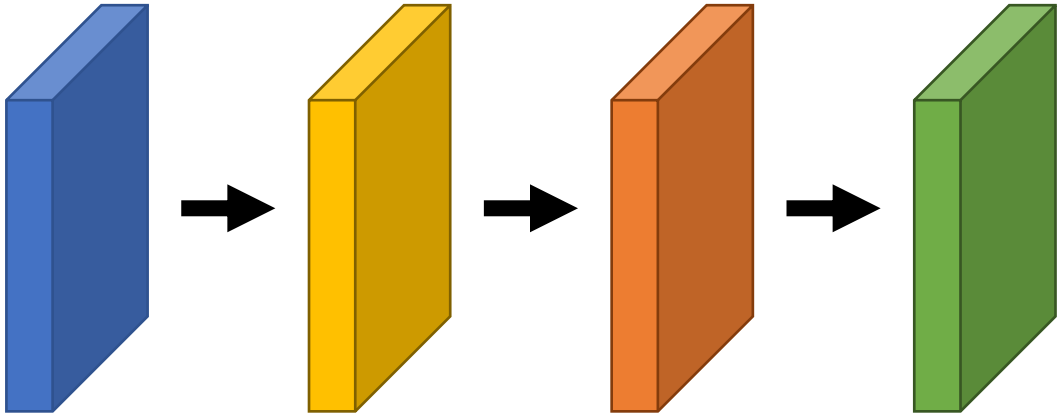


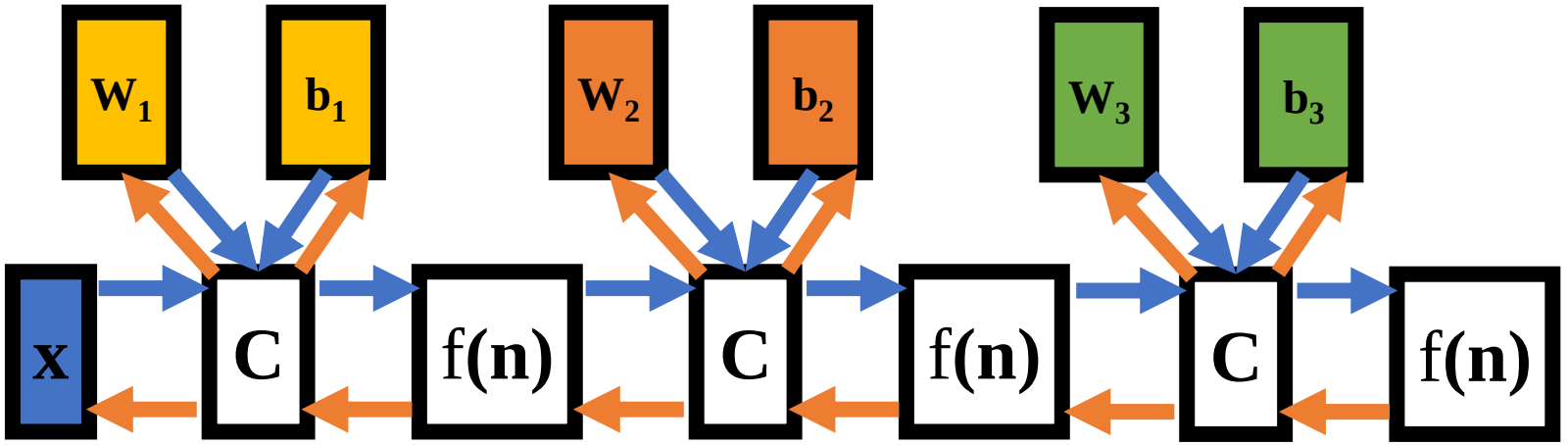


$$b + \sum_{i=1}^{F_h} \sum_{j=1}^{F_w} \sum_{k=1}^{c} F_{i,j,k} * I_{y+i,x+j,k}$$

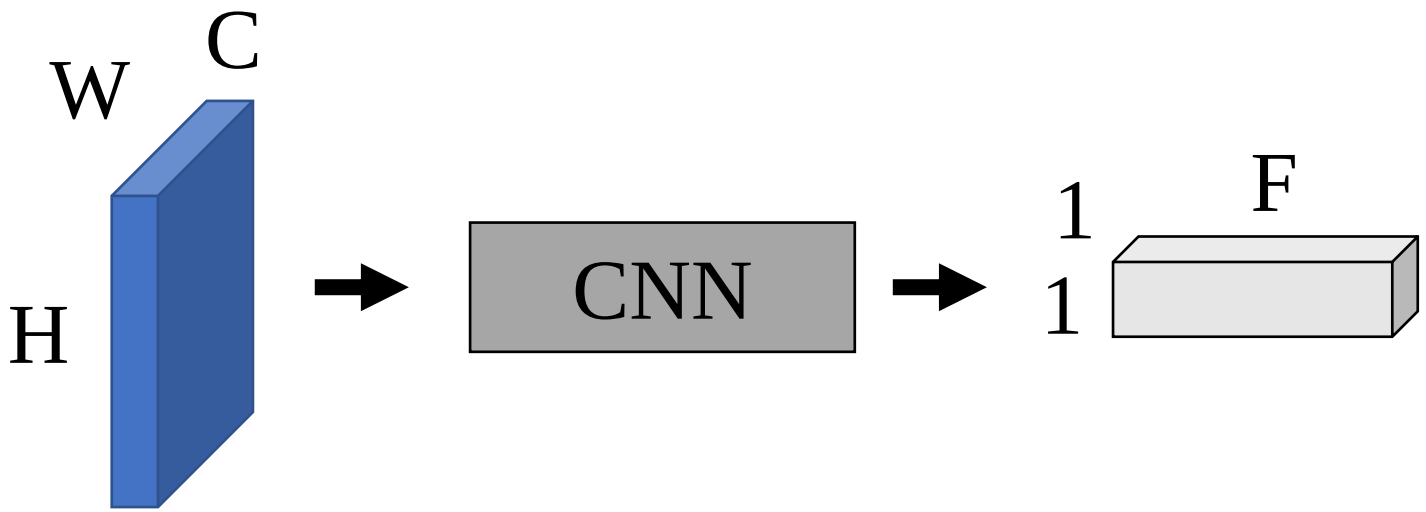
Slide credit: Karpathy and Fei-Fei

Convolutional Neural Network 卷积神经网络 (CNN)





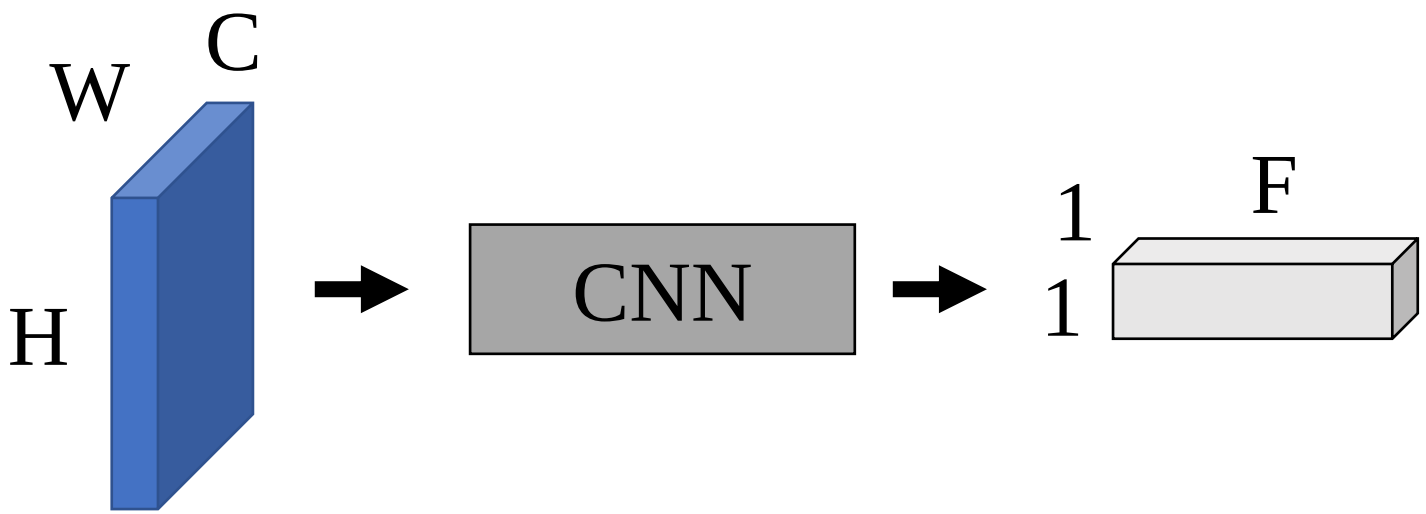
卷积神经网络能做很多任务



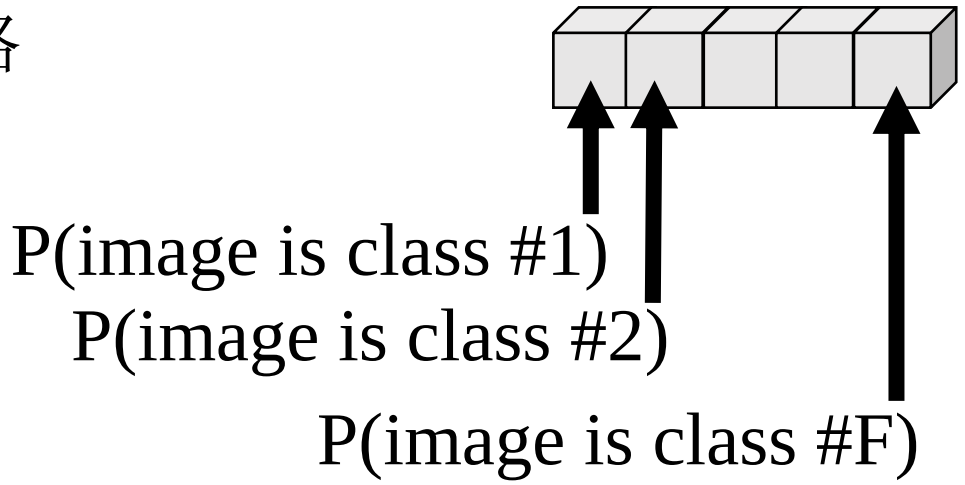
把 HxW 大小的图像转化为 F 大小的向量

- 一张图像分类为猫的概率？ (F=1)
- 1000类分类的得分? (F=1000)
- GPS定位坐标? (F=2)
- 28 个人体关键点的坐标？ (F=56)

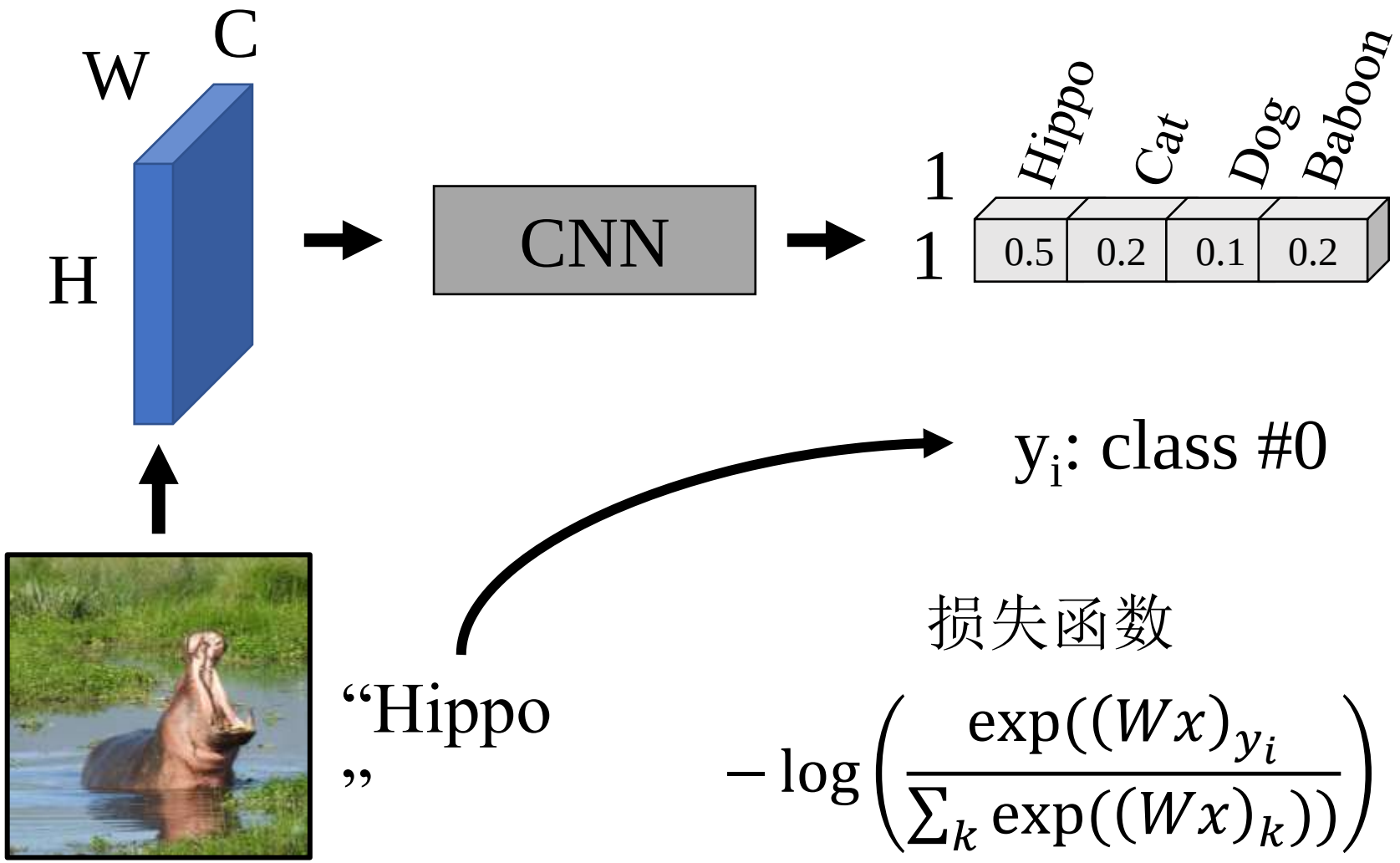
卷积神经网络：分类



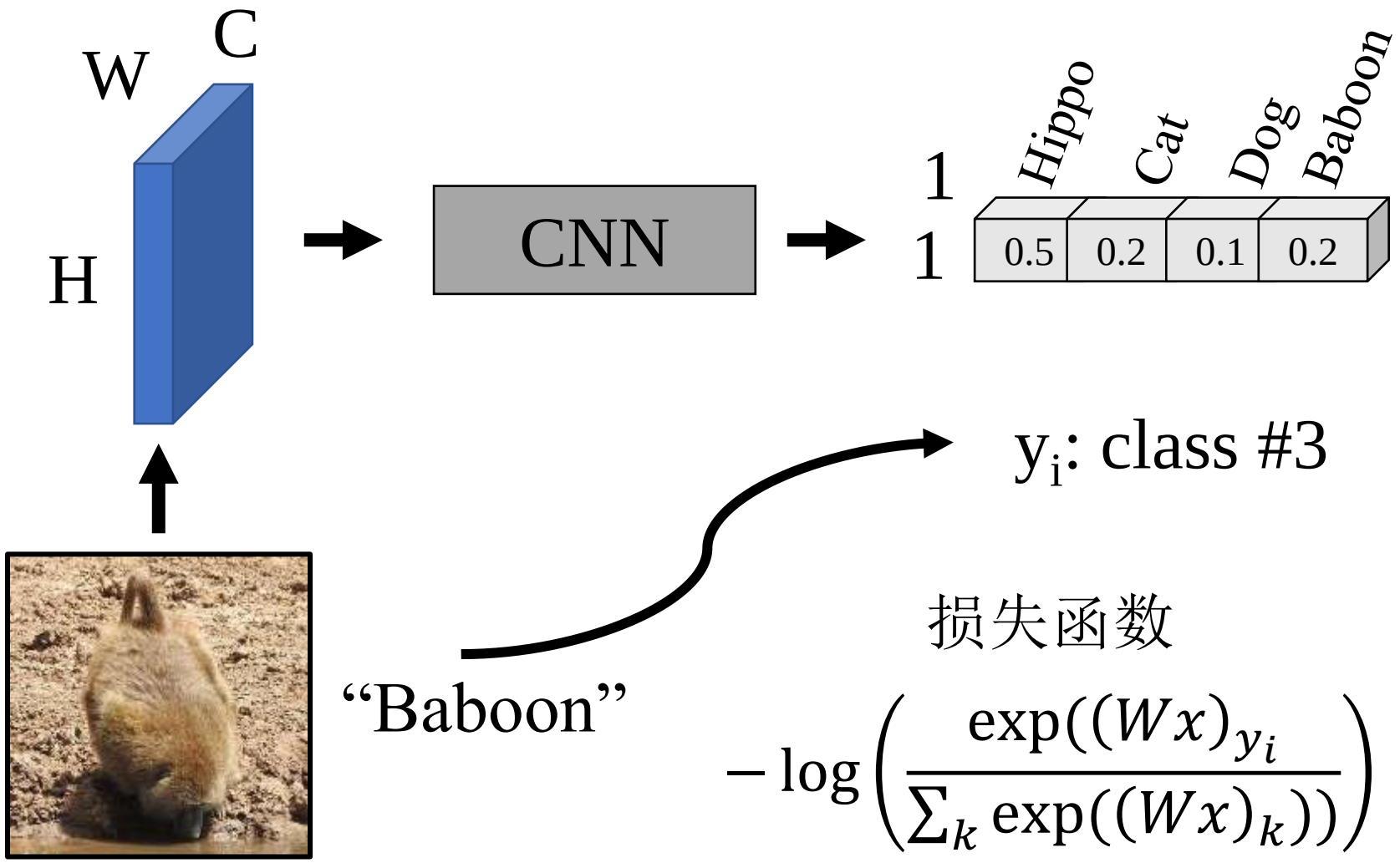
图像分类网络



卷积神经网络：分类


W
C
H
CNN
1
1
Hippo
Cat
Dog
Baboon
0.5
0.2
0.1
0.2
y_i: class #0
“Hippo
”
损失函数
$-\log\left(\frac{\exp((Wx)_{y_i}}{\sum_k \exp((Wx)_k))}\right)$

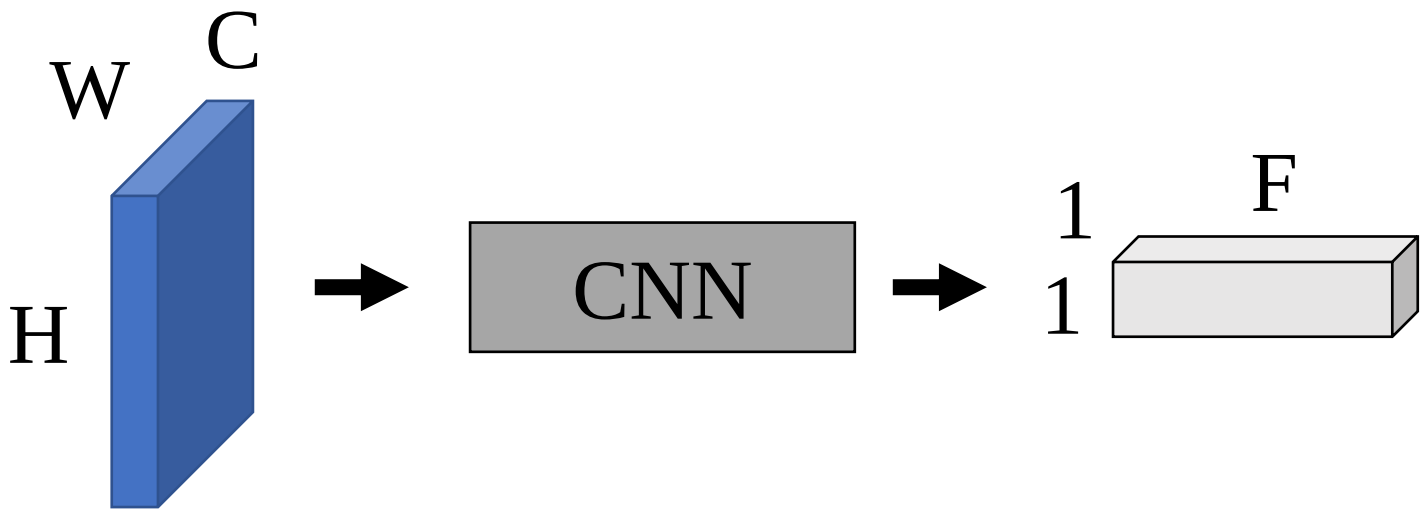
卷积神经网络：分类



损失函数

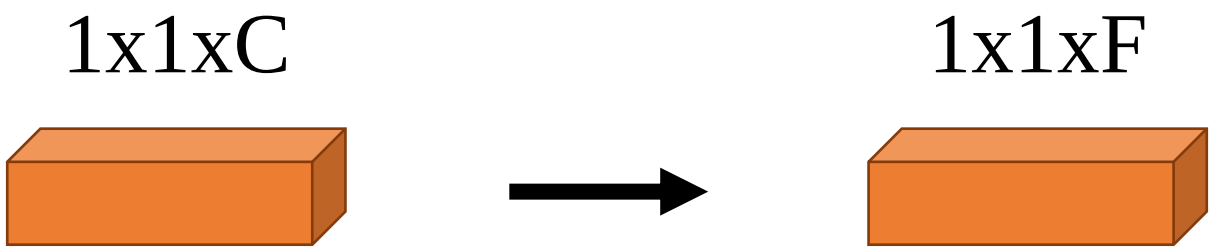
$$-\log\left(\frac{\exp((Wx)_{y_i}}{\sum_k \exp((Wx)_k))}\right)$$

训练CNN以完成任务



- 我们需要提供:
  - 训练的图像和期望的输出（标签）
  - 定义模型的流程框架，得到1x1xF 的输出
  - 定义损失函数评估训练的好坏
- 训练网络以调整网络参数来最小化损失函数

全连接层

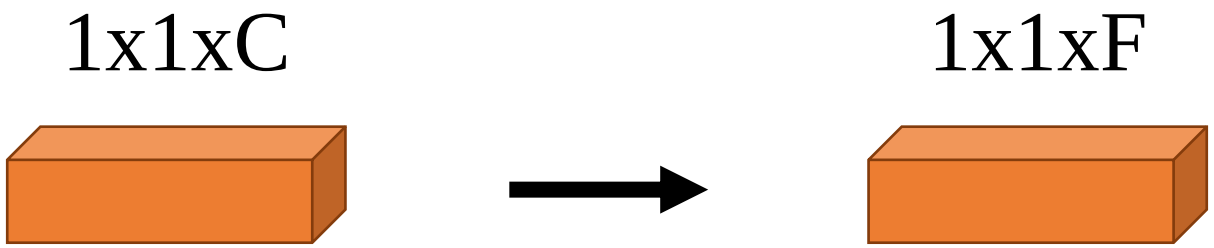


用线性变换把 C-维 特征 映射到 F-维 特征

W (FxC matrix) + b (Fx1 vector)

可以用卷积实现吗？

Everything's a Convolution



让 Fh=1, Fw=1

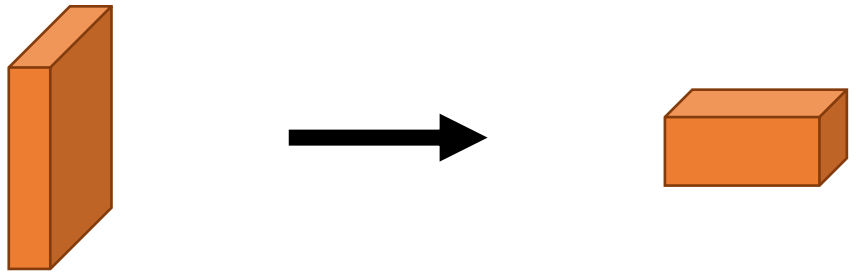
1x1 F 个滤波 的卷积

$$b + \sum_{i=1}^{F_h} \sum_{j=1}^{F_w} \sum_{k=1}^{c} F_{i,j,k} * I_{y+i,x+j,k} \longrightarrow b + \sum_{k=1}^{c} F_k * I_c$$

那么怎么把特征转化为向量呢？

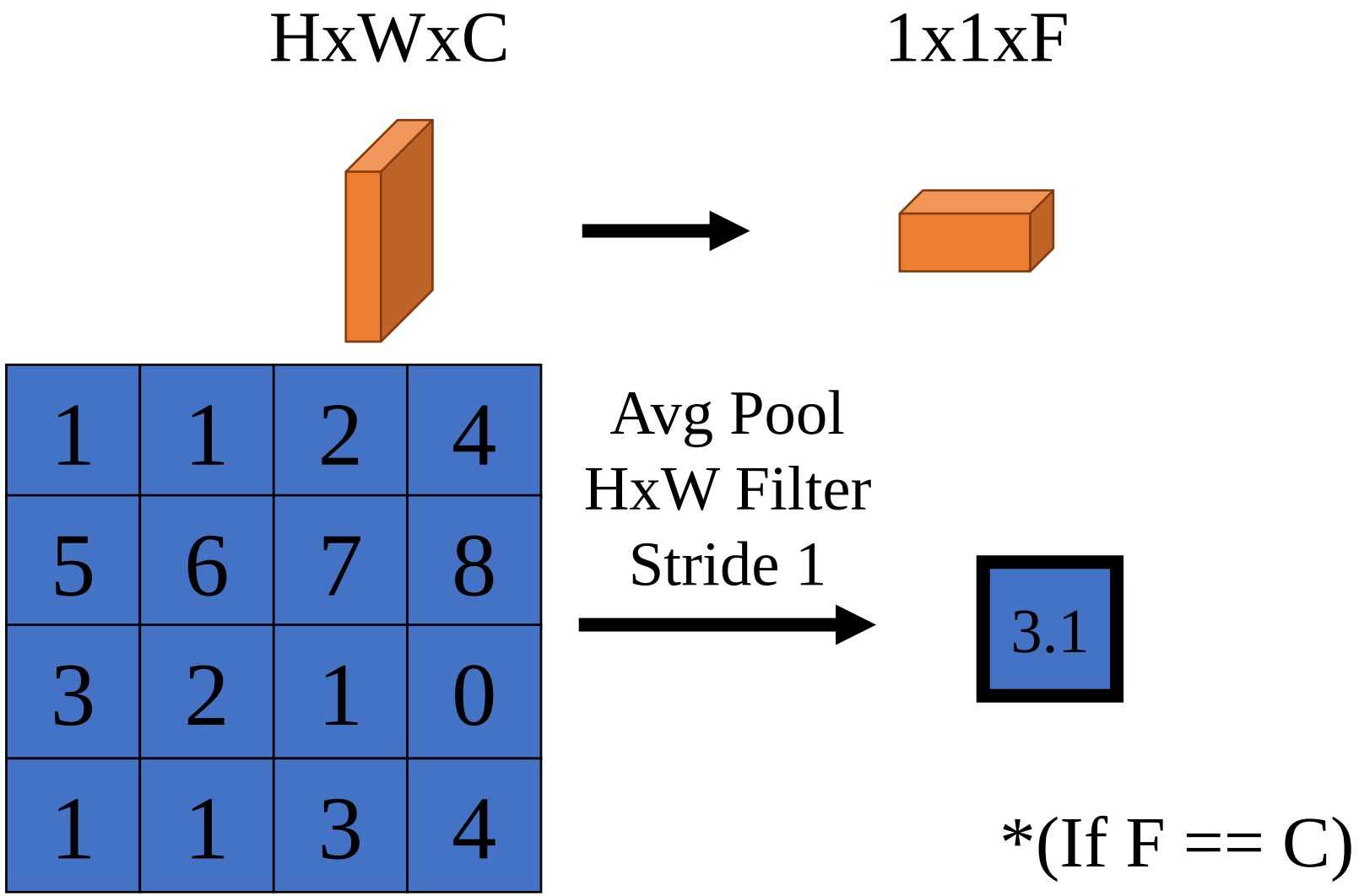
HxWxC

1x1xF



怎么做?

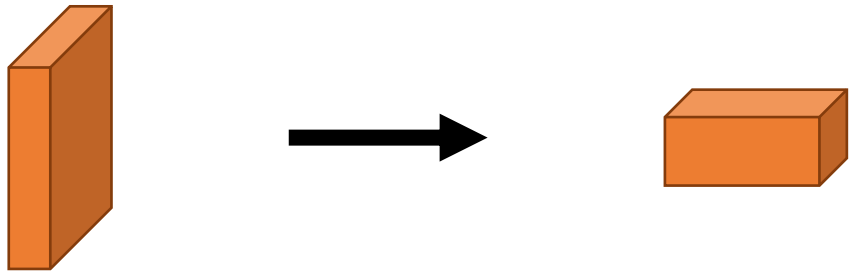
特征转化为向量 —— Pooling



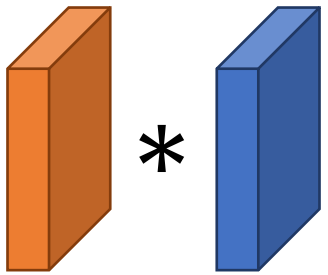
*(If F == C)

特征转化为向量—— 卷积

HxWxC　　1x1xF



HxW F 个滤波的卷积



每个滤波只
输出一个值

样例卷积网络

32x32x3 到 10x1 的分类预测

Figure Credit: Karpathy and Fei-Fei; see http://cs231n.stanford.edu/

样例卷积网络

输入: [32x32x3]
CONV with 10 3x3 filters, stride 1, pad 1:
得到: [32x32x10]
新增参数: (3*3*3)*10 + 10 = 280
RELU
CONV with 10 3x3 filters, stride 1, pad 1:
得到: [32x32x10]
新增参数: (3*3*10)*10 + 10 = 910
RELU
POOL with 2x2 filters, stride 2:
得到: [16x16x10]
新增参数: 0

Slide credit: Karpathy and Fei-Fei

样例卷积网络

之前的输出: [16x16x10]
CONV with 10 3x3 filters, stride 1:
得到: [16x16x10]
新增参数: (3*3*10)*10 + 10 = 910
RELU
CONV with 10 3x3 filters, stride 1:
得到: [16x16x10]
新增参数: (3*3*10)*10 + 10 = 910
RELU
POOL with 2x2 filters, stride 2:
得到: [8x8x10]
新增参数: 0

Slide credit: Karpathy and Fei-Fei

样例卷积网络

Conv, Relu, Conv, Relu, Pool 循环，直到得到[4x4x10]的特征

Fully-Connected FC layer to 10 neurons
(得到分类得分)
新增参数:
10 * 4 * 4 * 10 + 10 = 1610

结束!

Slide credit: Karpathy and Fei-Fei

换个做法

Conv, Relu, Conv, Relu, Pool 循环，直到得到[4x4x10]的特征

Average POOL 4x4x10 to 10 neurons
Fully-Connected FC layer to 10 neurons
(得到分类得分)
新增参数:
10 * 10 + 10 = 110

结束!

Slide credit: Karpathy and Fei-Fei

样例卷积网络

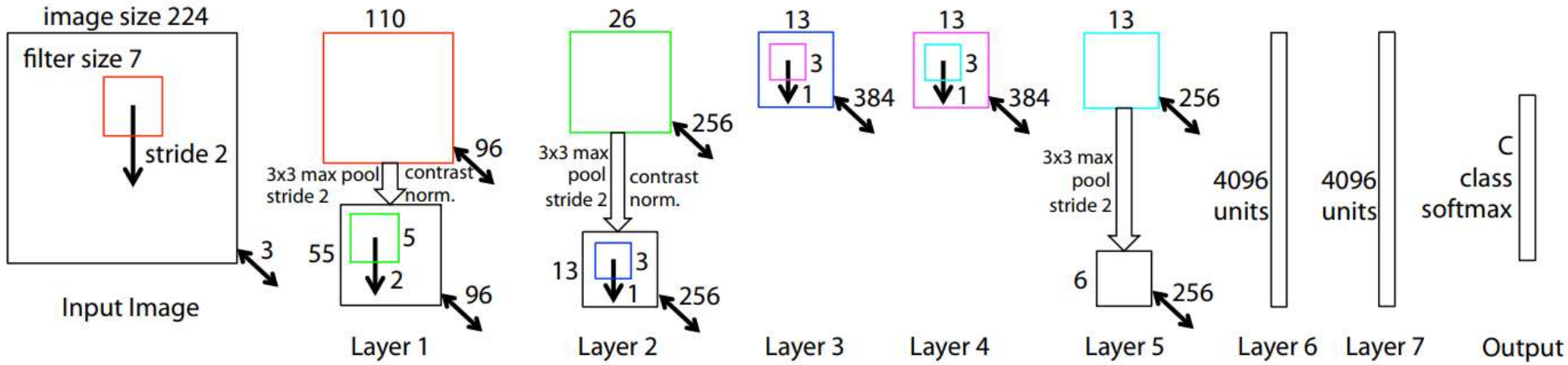
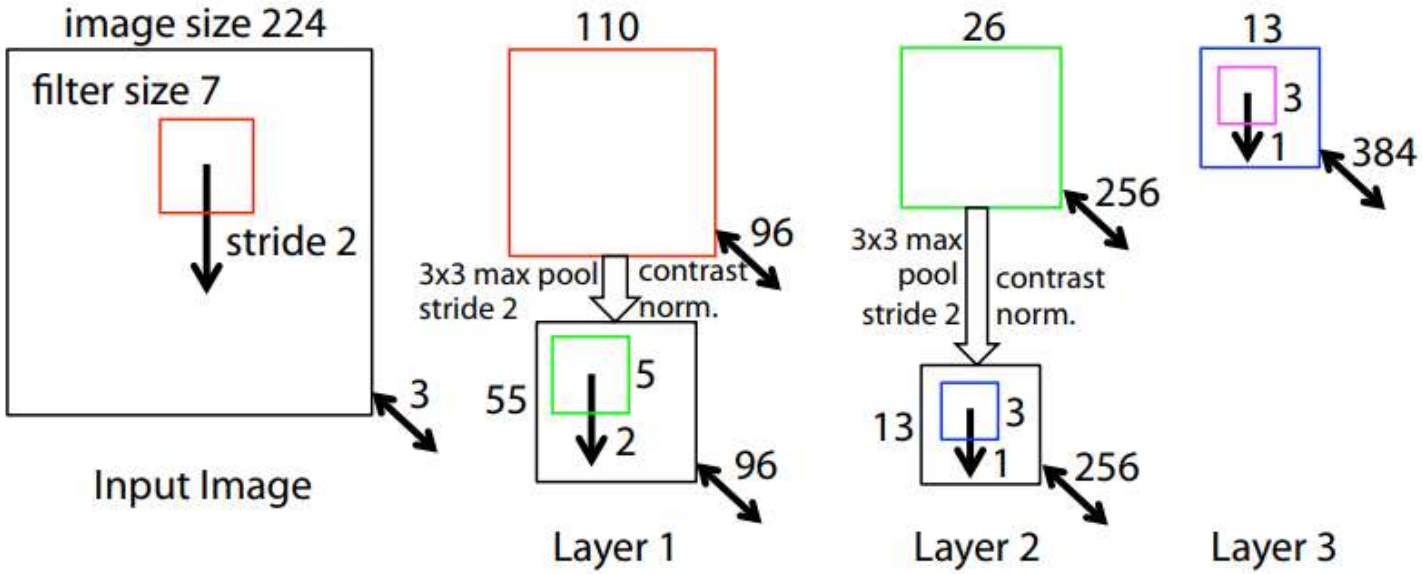


Figure Credit: Zeiler and Fergus, Visualizing and Understanding Convolutional Networks. ECCV 2014

样例卷积网络



(1) filter image with 96 7x7 filters
(2) ReLU
(3) 3x3 max pool with stride 2

Figure Credit: Zeiler and Fergus, Visualizing and Understanding Convolutional Networks. ECCV 2014

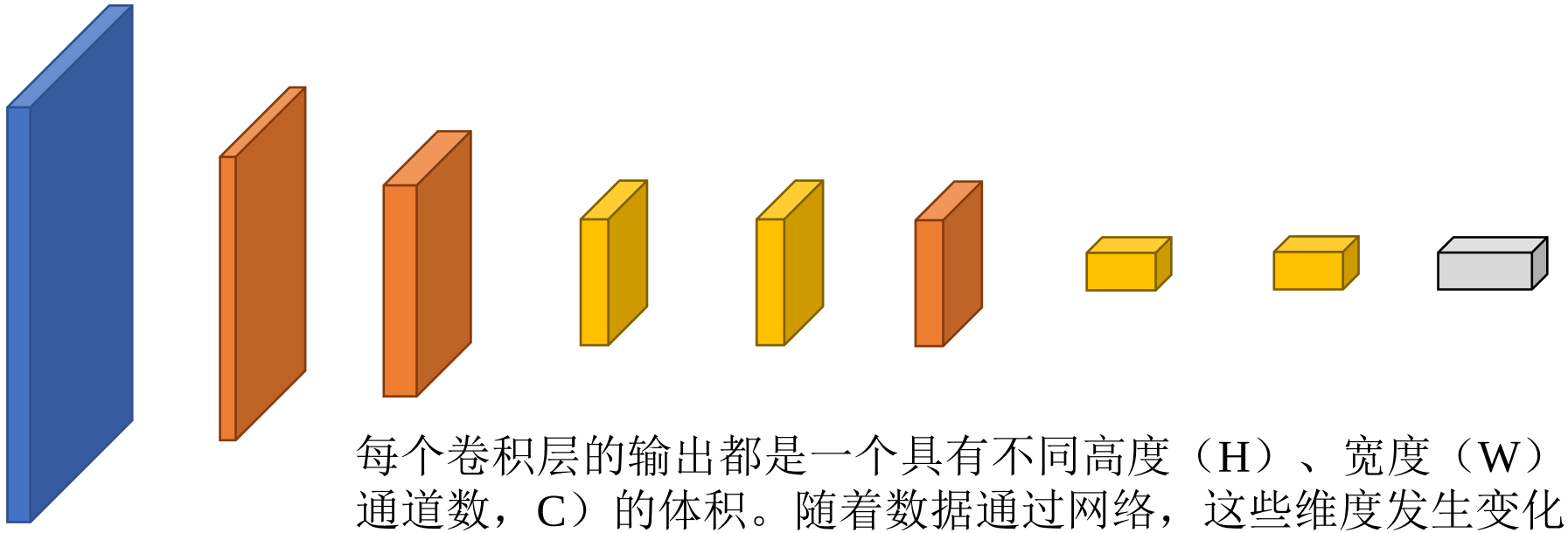
经典网络设计

- 输出 1000 类分类评分(Imagenet)
  - Alexnet (2012)
  - VGG-16 (2014)
  - Resnet (2015)

我们可以从多个基础模块中进行挑选，找到他们彼此能够适应的组合（类似搭积木）。

AlexNet

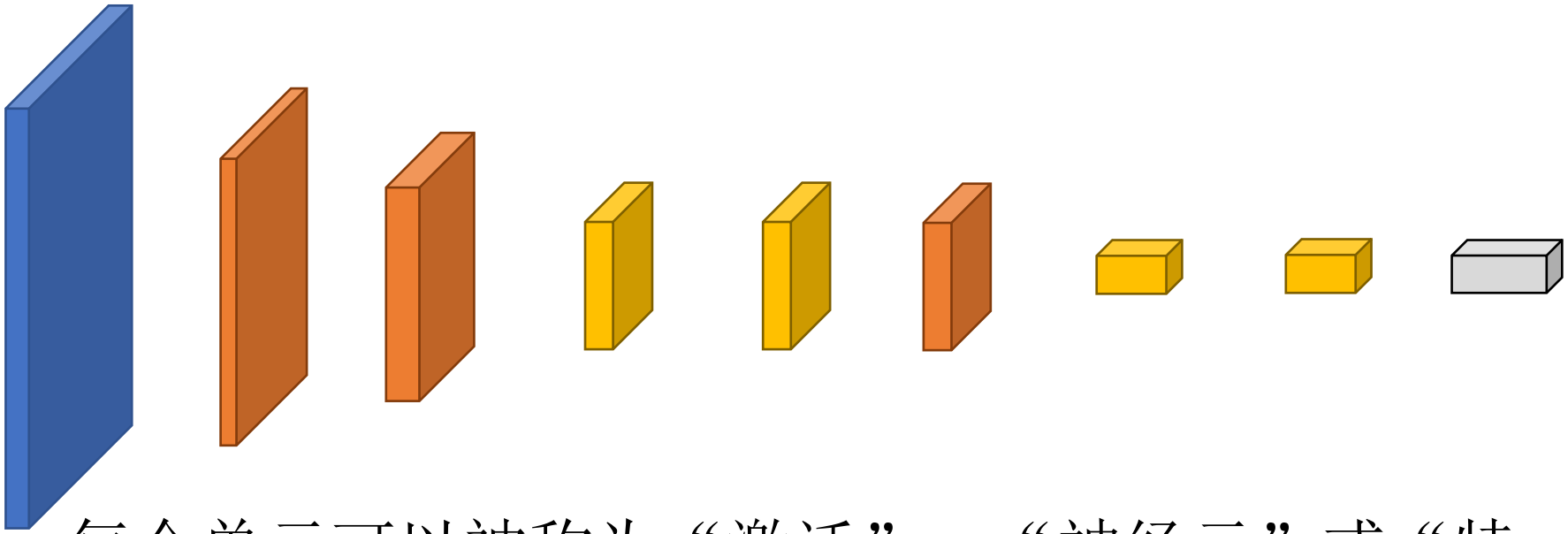
Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
227x227 3	55x55 96	27x27 256	13x13 384	13x13 384	13x13 256	1x1 4096	1x1 4096	1x1 1000



每个卷积层的输出都是一个具有不同高度（H）、宽度（W）、和深度（即通道数，C）的体积。随着数据通过网络，这些维度发生变化，最终导致全连接层，并得到最终的输出。每个块（block）表示数据在网络中流动时的体积，通过卷积操作将一个体积变换到另一个体积。

CNN基本术语

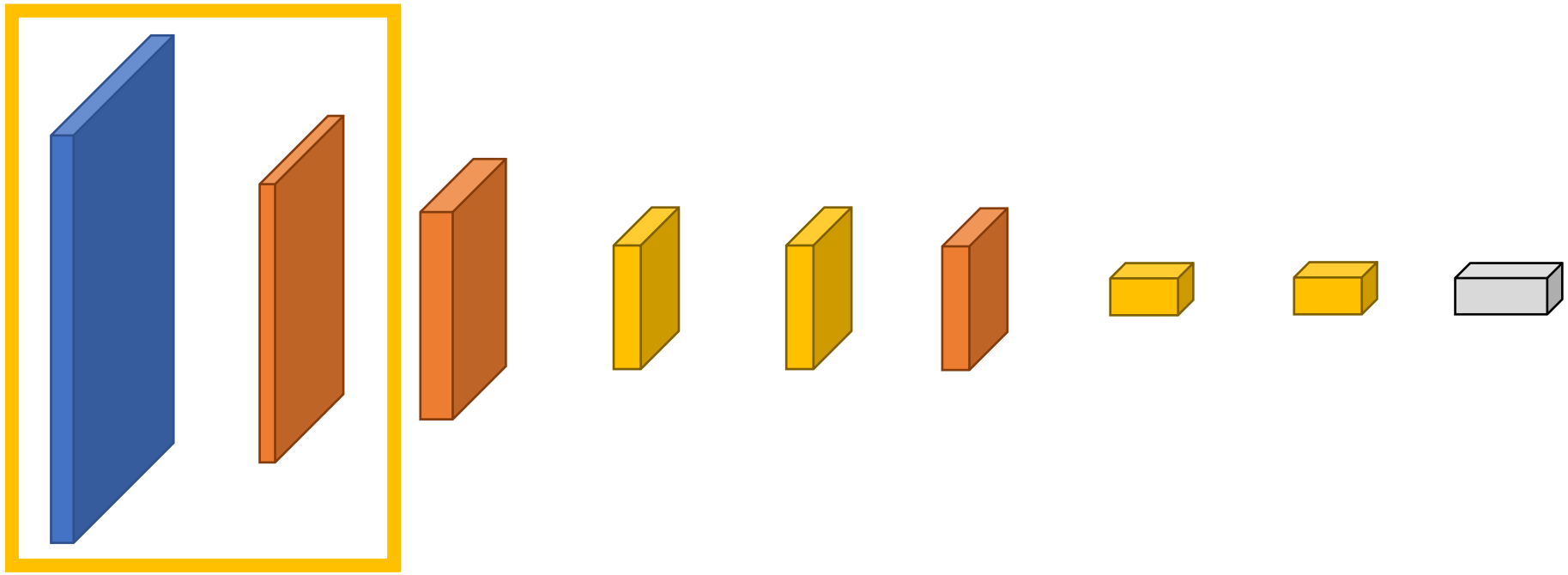
Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
227x227 3	55x55 96	27x272 56	13x13 384	13x13 384	13x13 256	1x1 4096	1x1 4096	1x1 1000



每个单元可以被称为“激活”、“神经元”或“特征”。

AlexNet

Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
227x227 3	55x55 96	27x27 256	13x13 384	13x13 384	13x13 256	1x1 4096	1x1 4096	1x1 1000



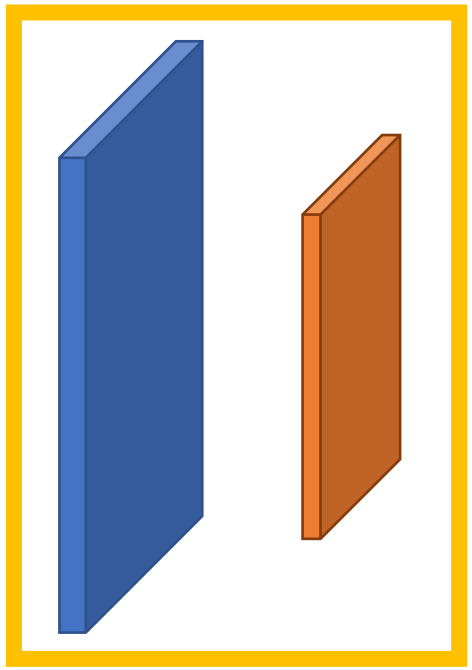
AlexNet

Input

227x227
3

Conv
1

55x55
96

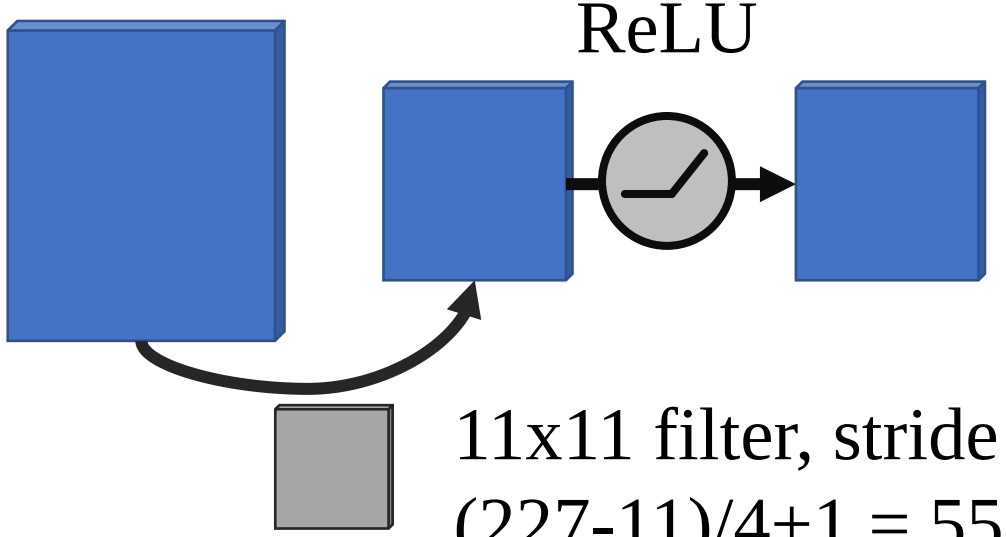


227x227
3

55x55
96

ReLU

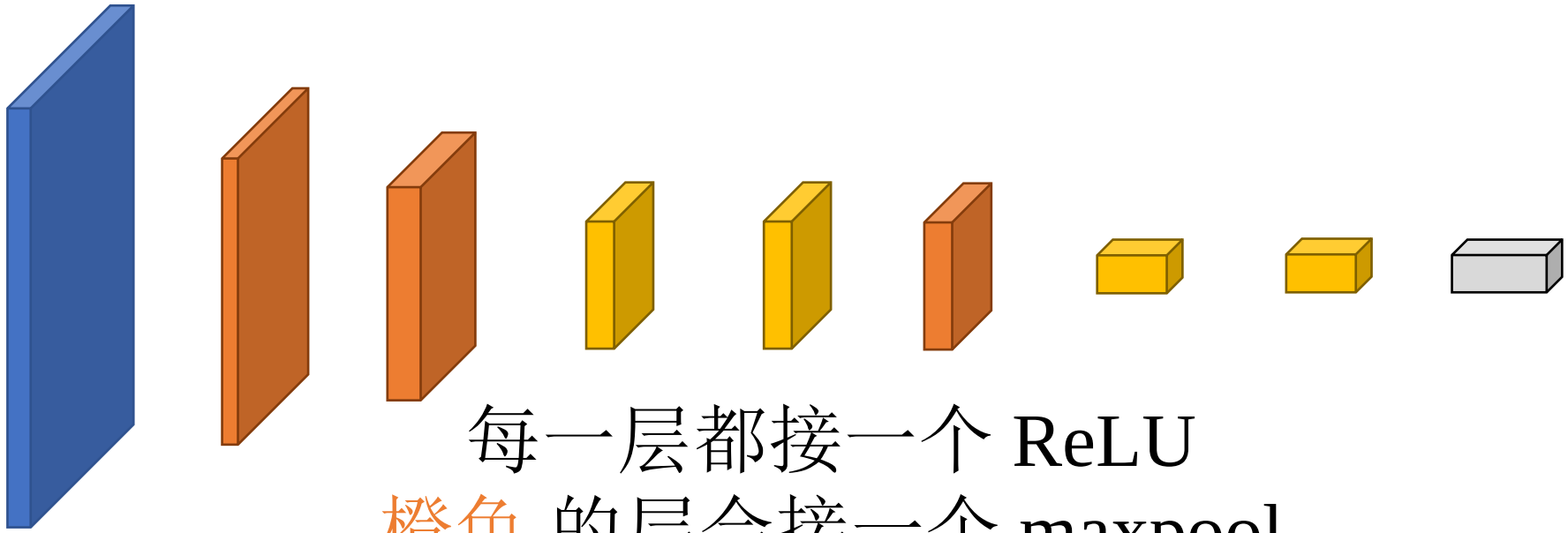
55x55
96



11x11 filter, stride of 4
(227-11)/4+1 = 55

AlexNet

Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
227x227 3	55x55 96	27x27 256	13x13 384	13x13 384	13x13 256	1x1 4096	1x1 4096	1x1 1000



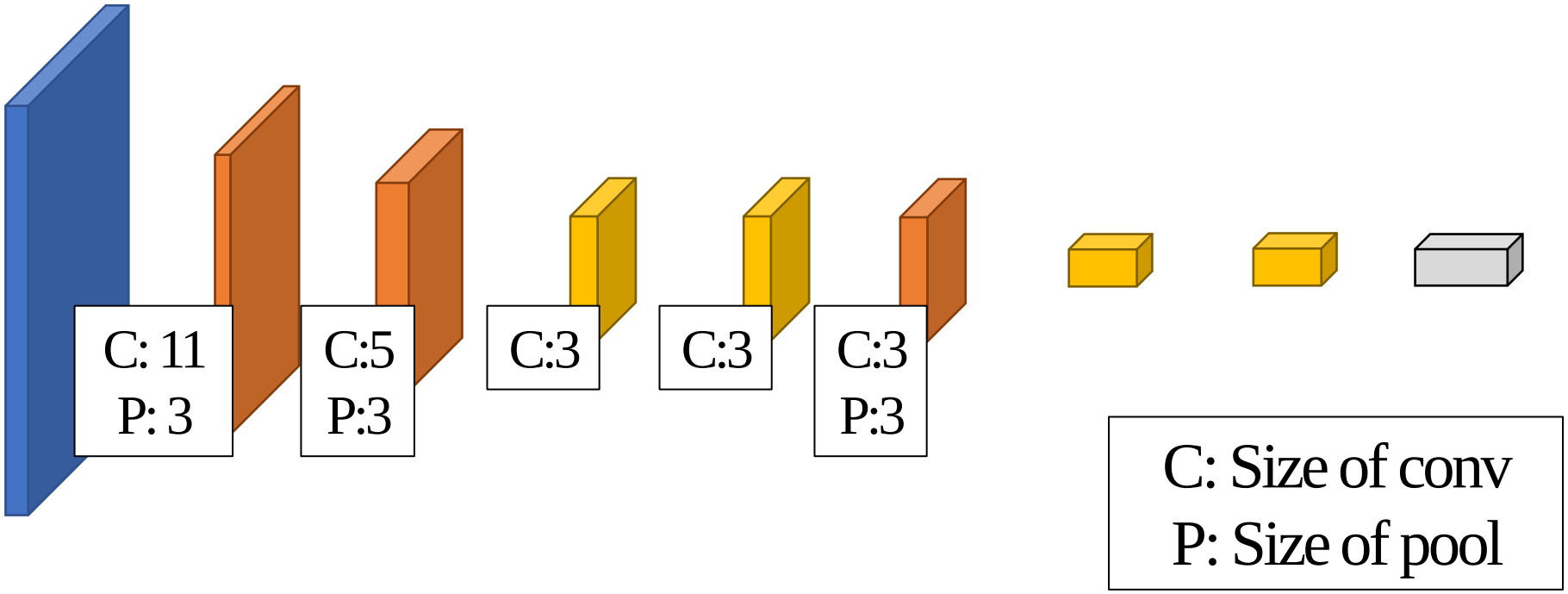
每一层都接一个 ReLU

橙色 的层会接一个 maxpool

每一层都有 “normalization” 标准化

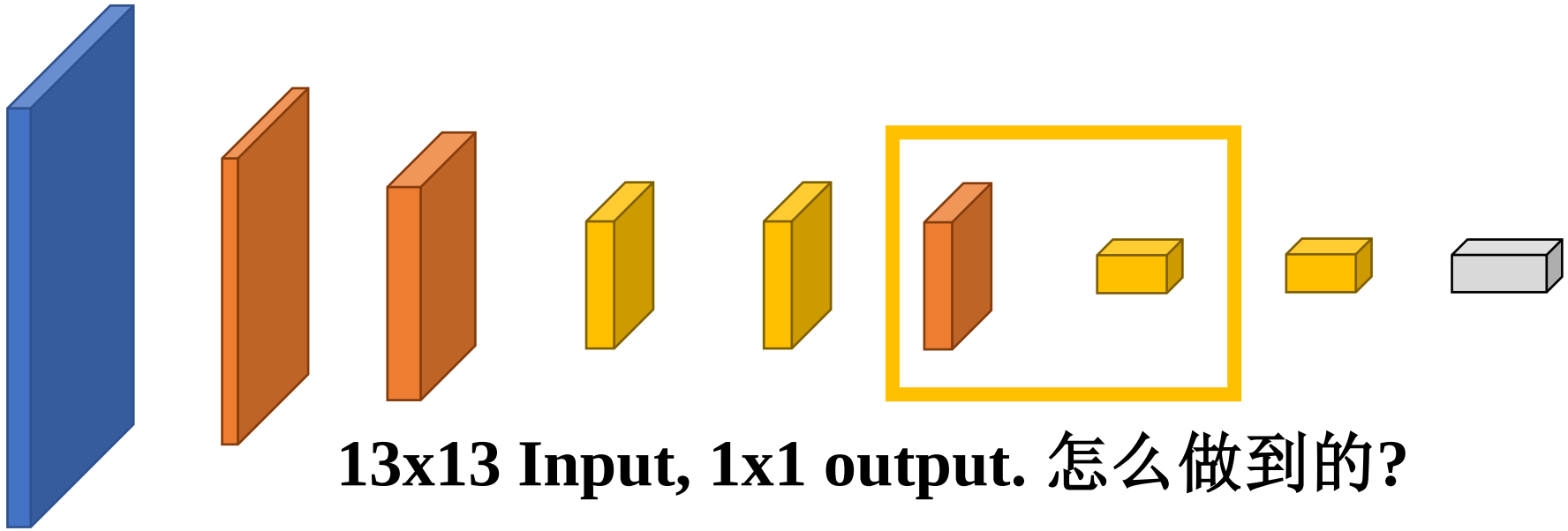
AlexNet – Details

Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
227x227 3	55x55 96	27x27 256	13x13 384	13x13 384	13x13 256	1x1 4096	1x1 4096	1x1 1000



AlexNet

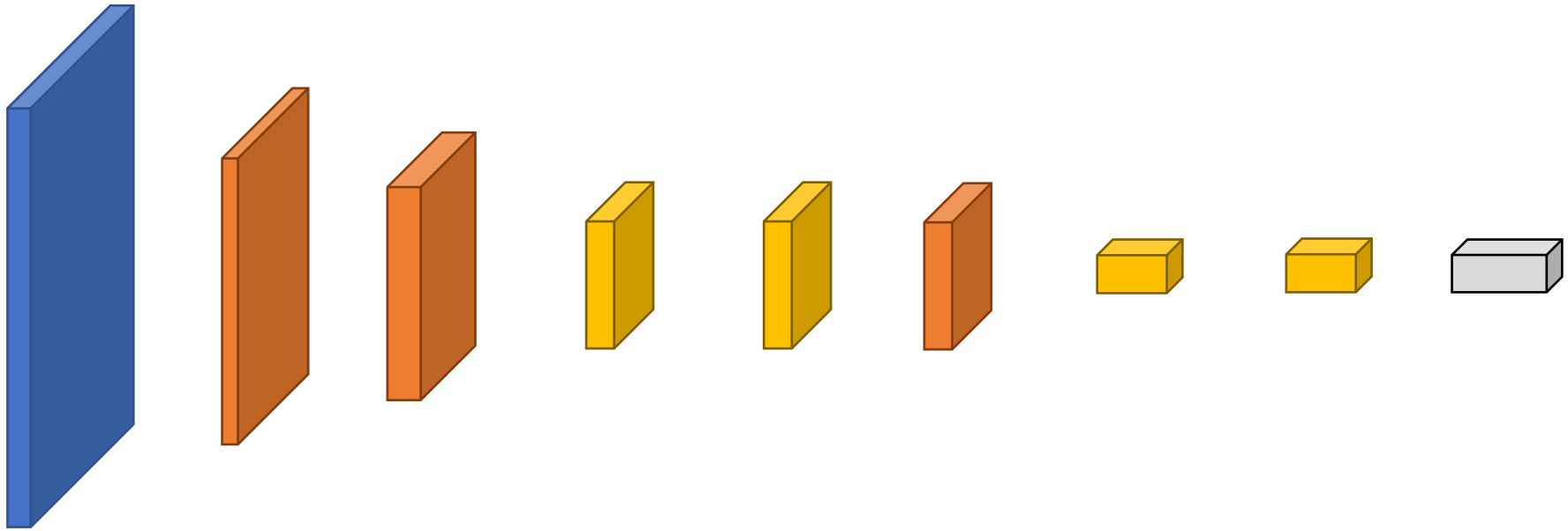
Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
227x227 3	55x55 96	27x27 256	13x13 384	13x13 384	13x13 256	1x1 4096	1x1 4096	1x1 1000



13x13 Input, 1x1 output. 怎么做到的?

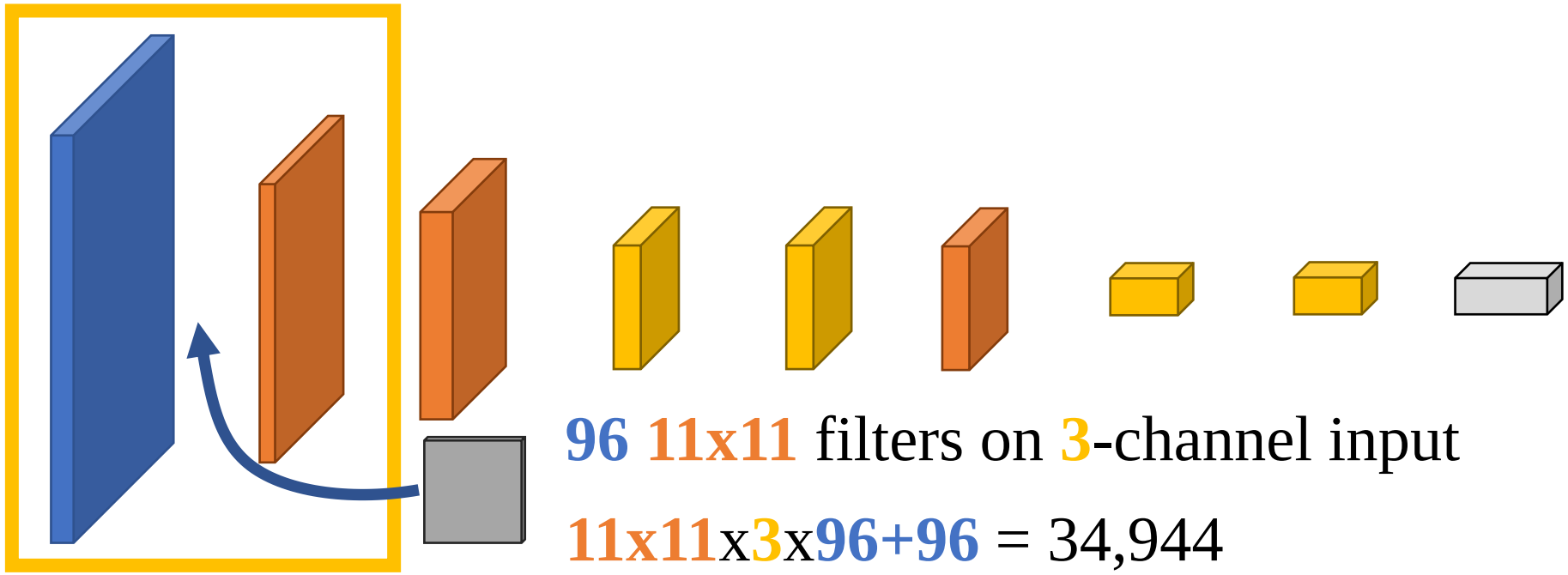
Alexnet – 有多少参数?

Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
227x227 3	55x55 96	27x27 256	13x13 384	13x13 384	13x13 256	1x1 4096	1x1 4096	1x1 1000



Alexnet –有多少参数?

Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
227x227 3	55x55 96	27x27 256	13x13 384	13x13 384	13x13 256	1x1 4096	1x1 4096	1x1 1000

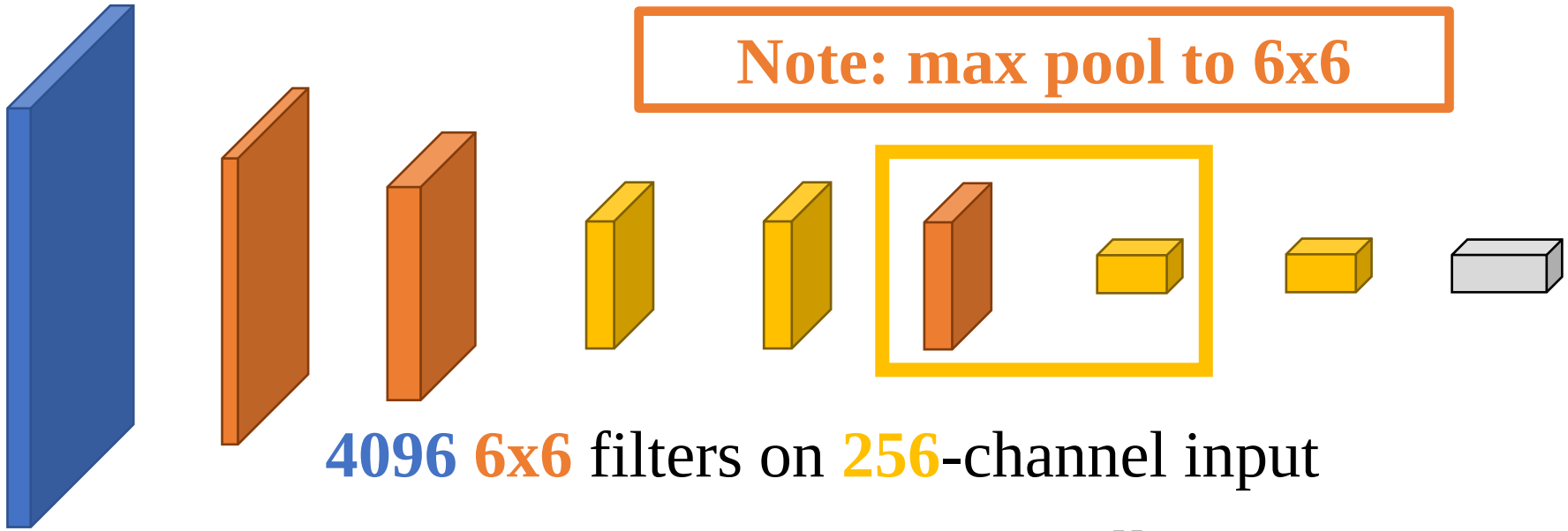


96 **11x11** filters on **3**-channel input

11x11x**3**x**96**+**96** = 34,944

Alexnet –有多少参数?

Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
227x227 3	55x55 96	27x27 256	13x13 384	13x13 384	13x13 256	1x1 4096	1x1 4096	1x1 1000

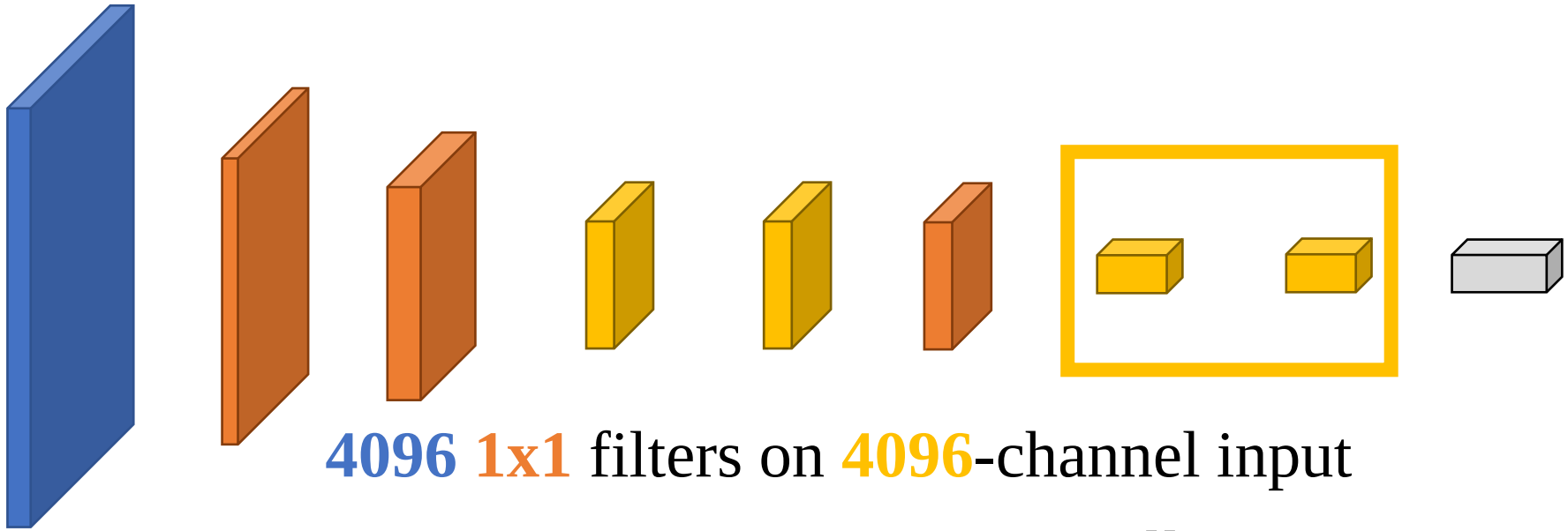


4096 **6x6** filters on **256**-channel input

6x6x**256**x**4096+4096** = 38 million

Alexnet –有多少参数?

Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
227x227 3	55x55 96	27x27 256	13x13 384	13x13 384	13x13 256	1x1 4096	1x1 4096	1x1 1000



4096 1x1 filters on **4096**-channel input

1x1x**4096**x**4096+4096** = 17 million

Alexnet –有多少参数

如果以每个参数4秒的速度列出AlexNet所有参数，会需要多长时间?

1年?　　4年?　　8年?　　16年?

- AlexNet有62.4百万个参数
- *绝大多数参数都在全连接层中*
- 论文中指出去掉卷积层会对性能产生灾难性的影响

数据集 – ILSVRC

- Imagenet Largescale Visual Recognition Challenge
- 1.4M张图像
- 1000个类别，类别通常非常精确

数据集 – ILSVRC

Figure Credit: O. Russakovsky

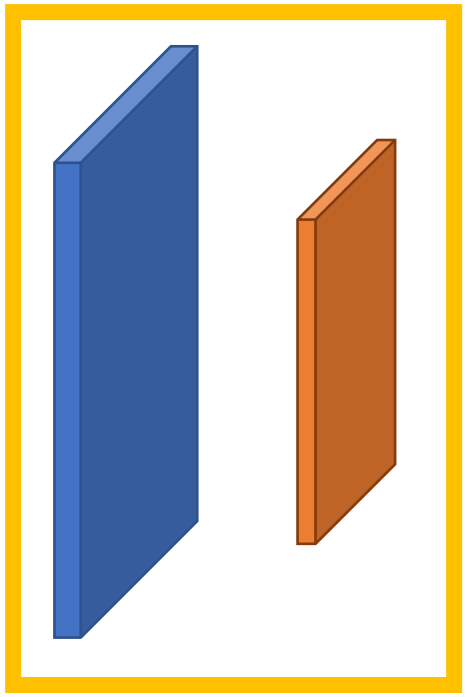
可视化卷积核

Input

227x227
3

Conv
1

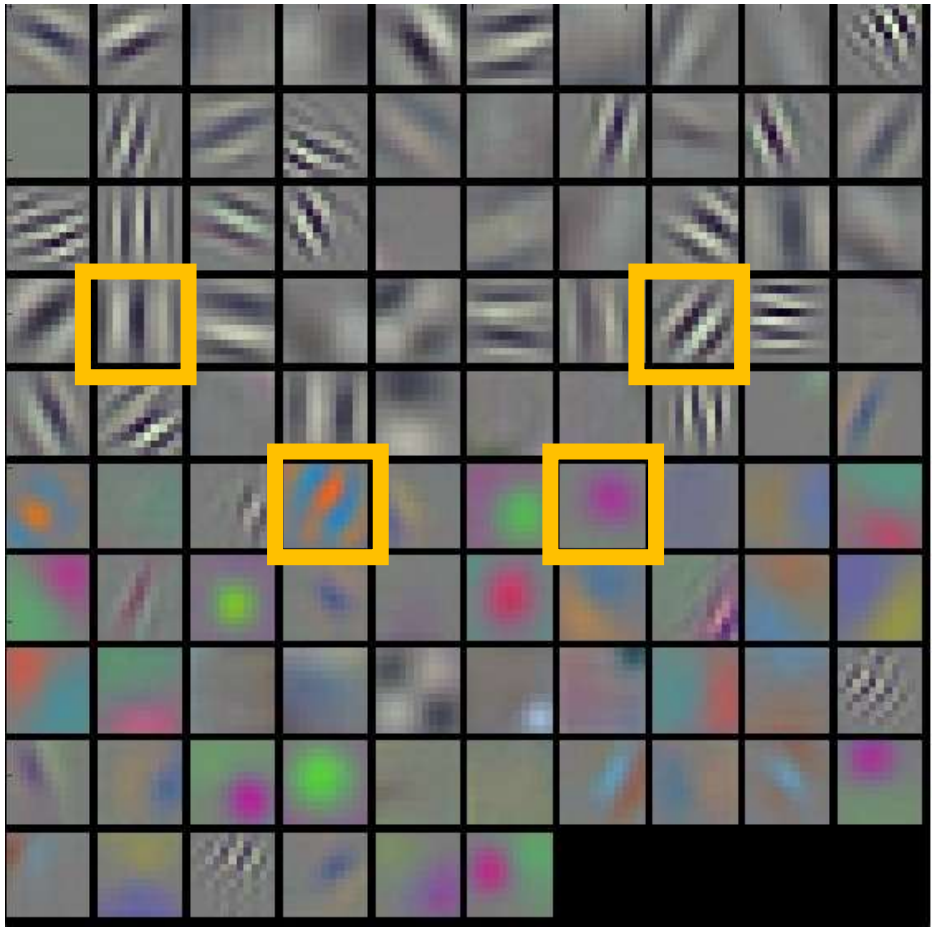
55x559
6



Conv 1 Filters

- **Q. 有多少输入通道?**
  - A: 3
- **每个输入通道代表什么?**
  - R, G, B.

卷积滤波学到了什么？



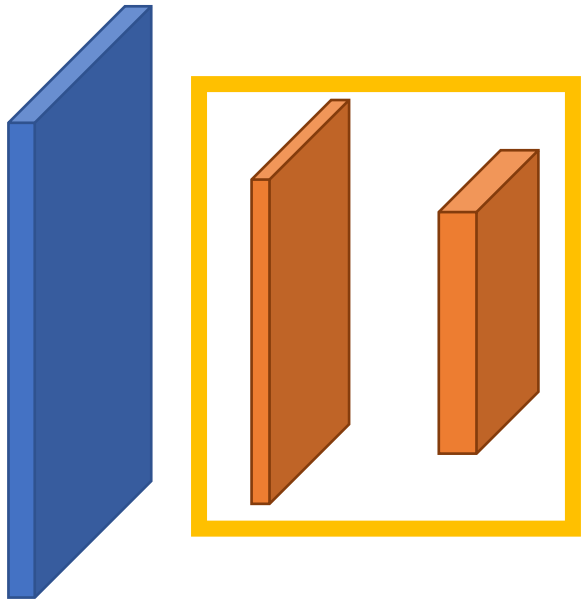
第一层卷积滤波器学到的特征，这些滤波器被训练用于识别1000个不同的对象类别。

这些滤波器是在颜色上应用的，可以捕捉到基于颜色的特征

Figure Credit: Karpathy and Fei-Fei

更深层次的滤波器的可视化

Input	Conv 1	Conv 2
227x227 3	55x559 6	27x272 56



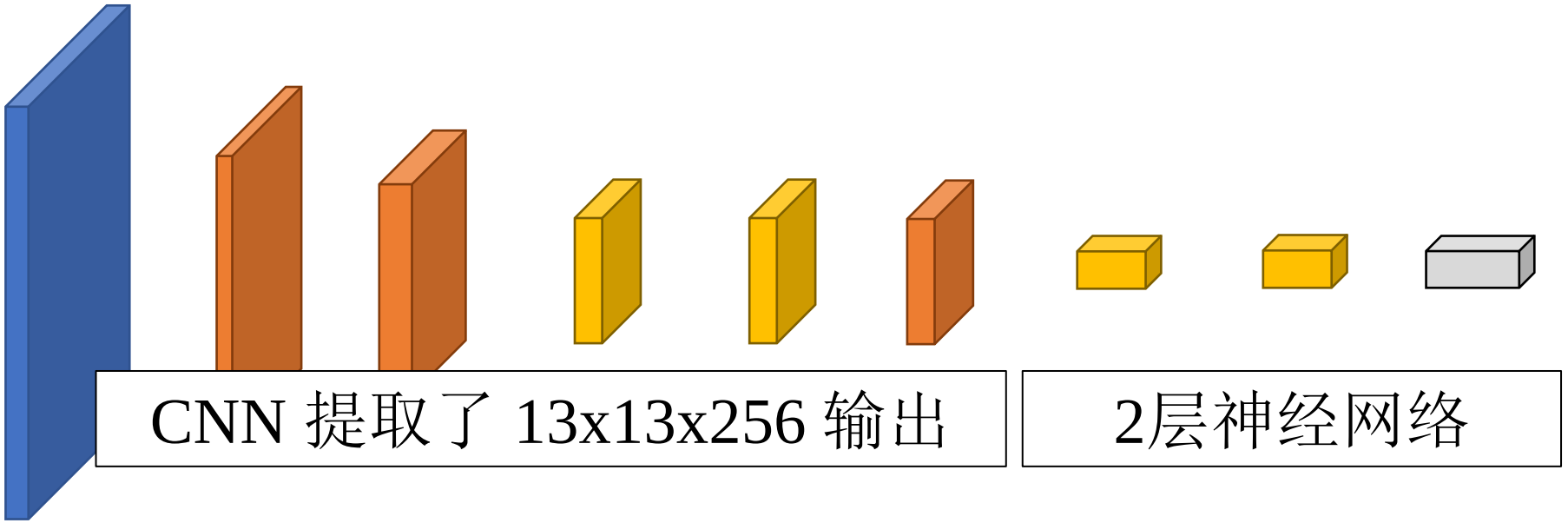
- 理解更深层卷积层的滤波器的具体值是非常困难的，甚至是不可能的，因为输入维度太多，并且输入的具体含义不明确。这是因为随着网络层数的加深，每个滤波器所代表的特征变得更加抽象和复杂，不再像初级滤波器那样容易通过视觉上直接解释。

Conv 2 Filters

- **Q. 有多少通道?**
  - A: 96….
- **每个输入通道代表?**
- 大概是基于颜色的特征。。。

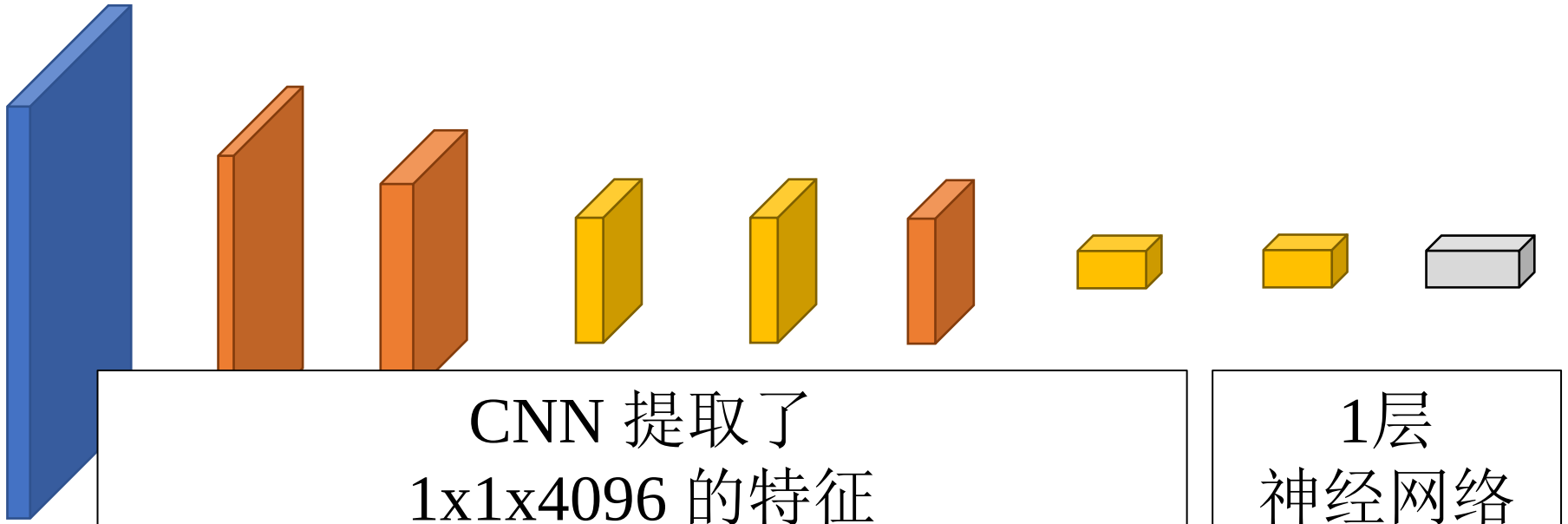
理解深层网络

Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
227x227 3	55x55 96	27x27 256	13x13 384	13x13 384	13x13 256	1x1 4096	1x1 4096	1x1 1000



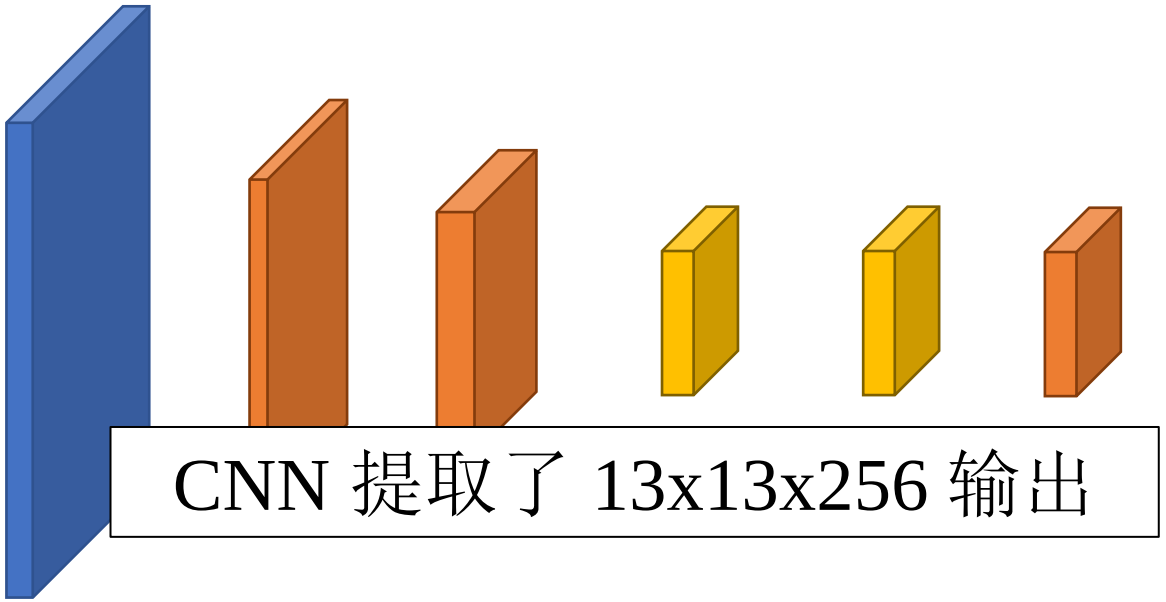
理解深层网络

Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
227x227 3	55x55 96	27x27 256	13x13 384	13x13 384	13x13 256	1x1 4096	1x1 4096	1x1 1000



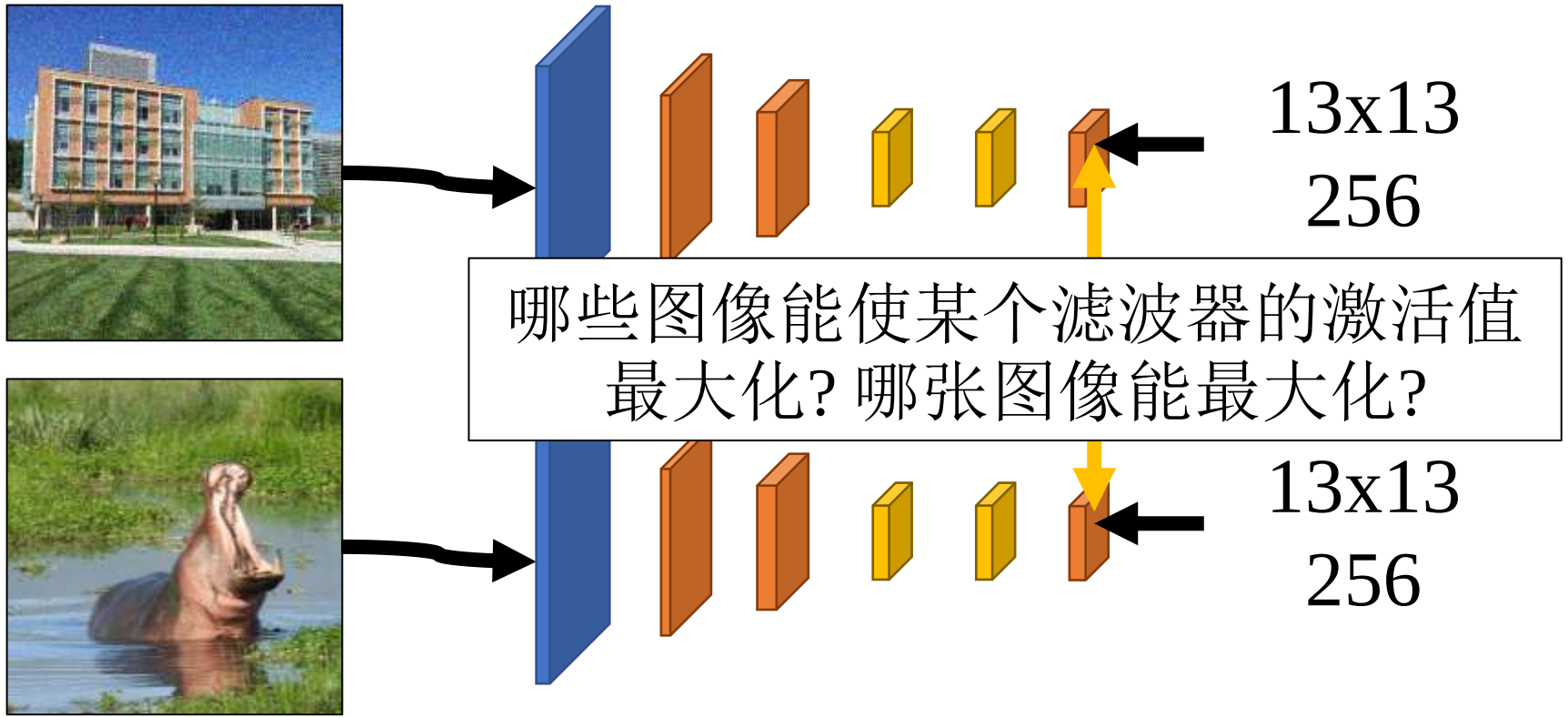
理解深层网络

Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5
227x227 3	55x55 96	27x27 256	13x13 384	13x13 384	13x13 256



理解深层网络

给CNN输入不同的图像，观察不同的滤波器如何对这些图像做出反应。这可以帮助理解特定滤波器的激活模式，即通过看哪些图像能使某个滤波器的激活值最大化，我们可以得到关于该滤波器所捕捉特征的线索。



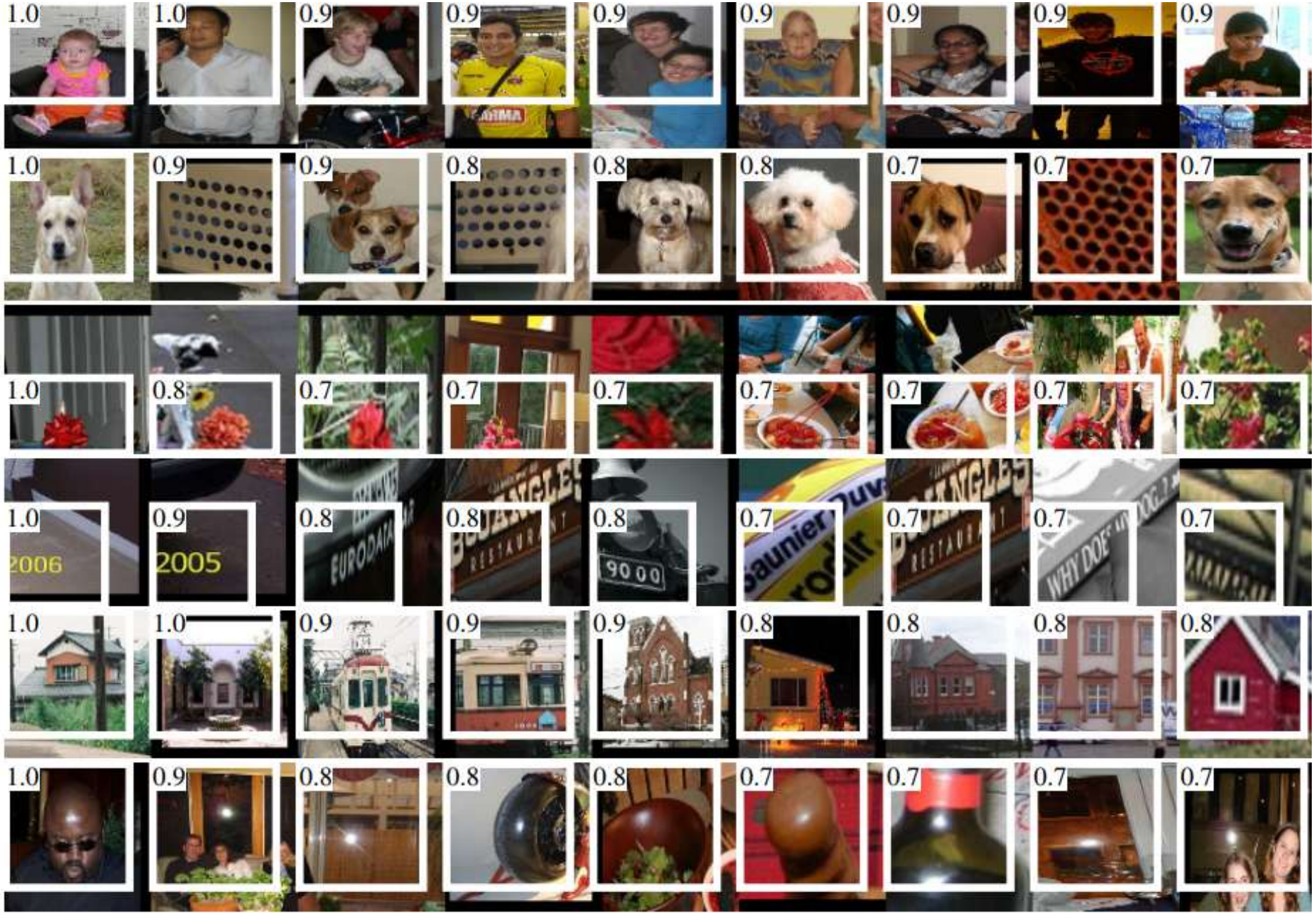
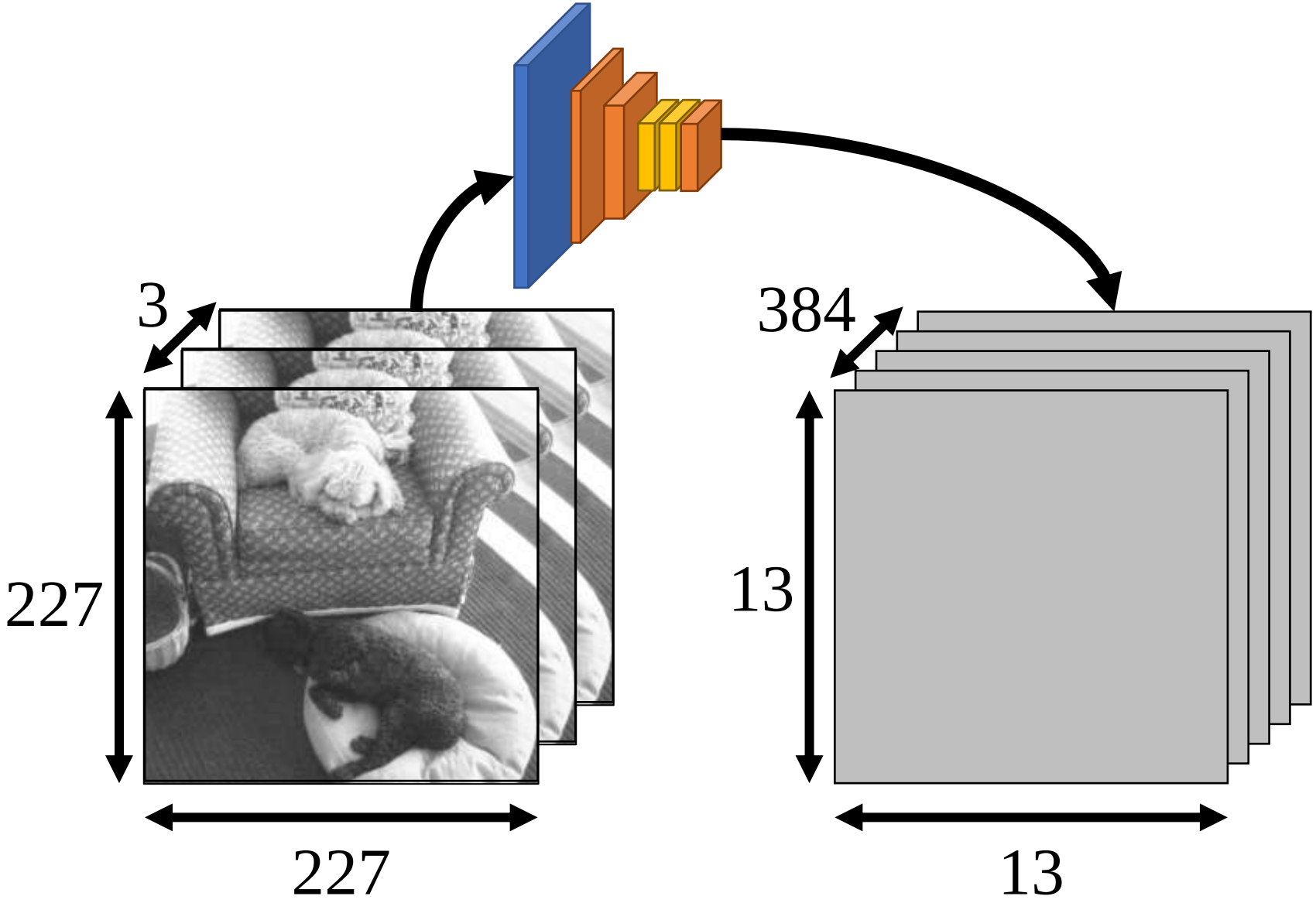
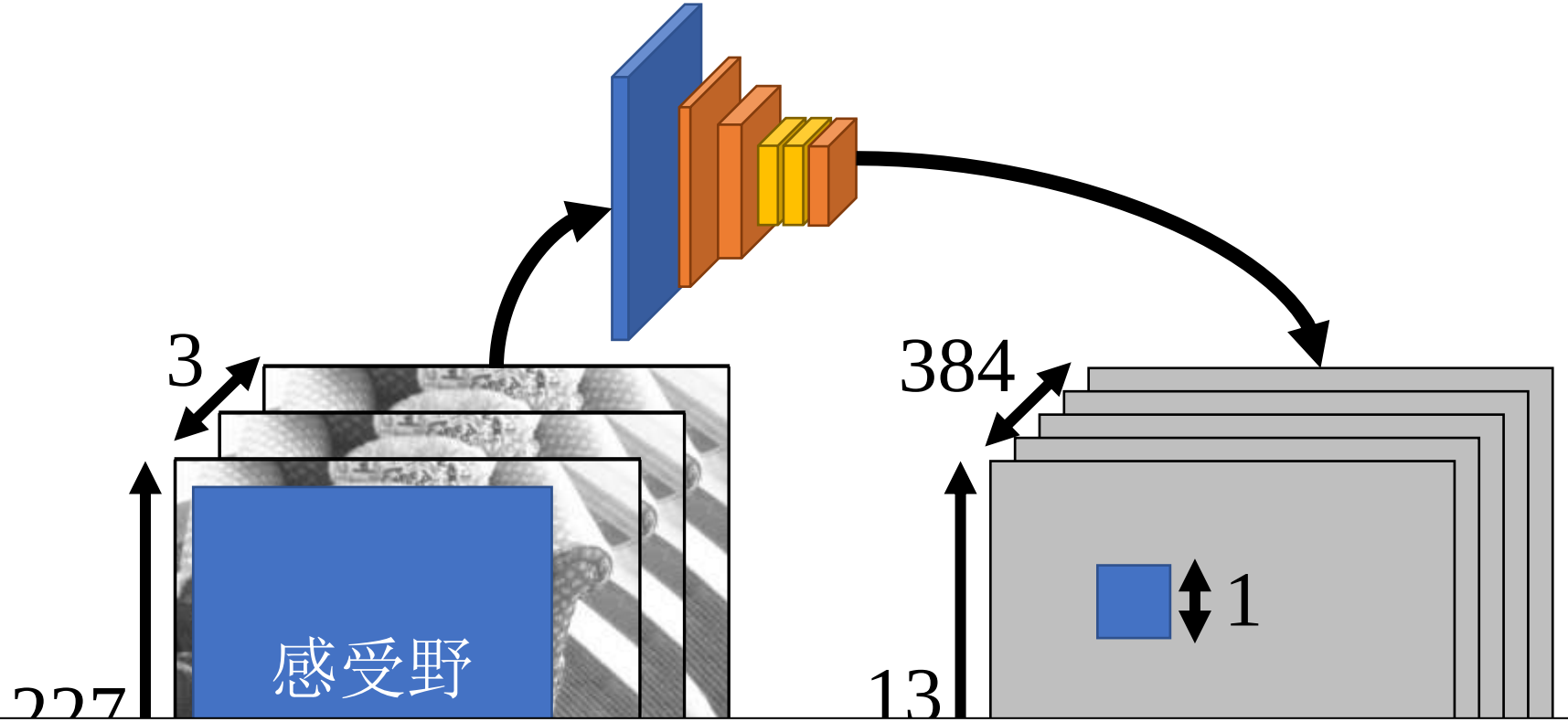


Figure Credit: Girschick et al. CVPR 2014.

怎么得到白框?



感受野



每一个输出特征图的单个像素实际上是由输入图像中的一个小区域转换而来的。卷积层不会一次性看到整个图像，而是通过滤波器窗口看到图像的一小部分，这个窗口逐步移动覆盖整个图像，从而在特征图上生成每一个单独的像素点。

我们可以用感受野来看CNN在做决策时关注哪些区域

B. Zhou et al. Learning Deep Features for Discriminative Localization. CVPR 2016.