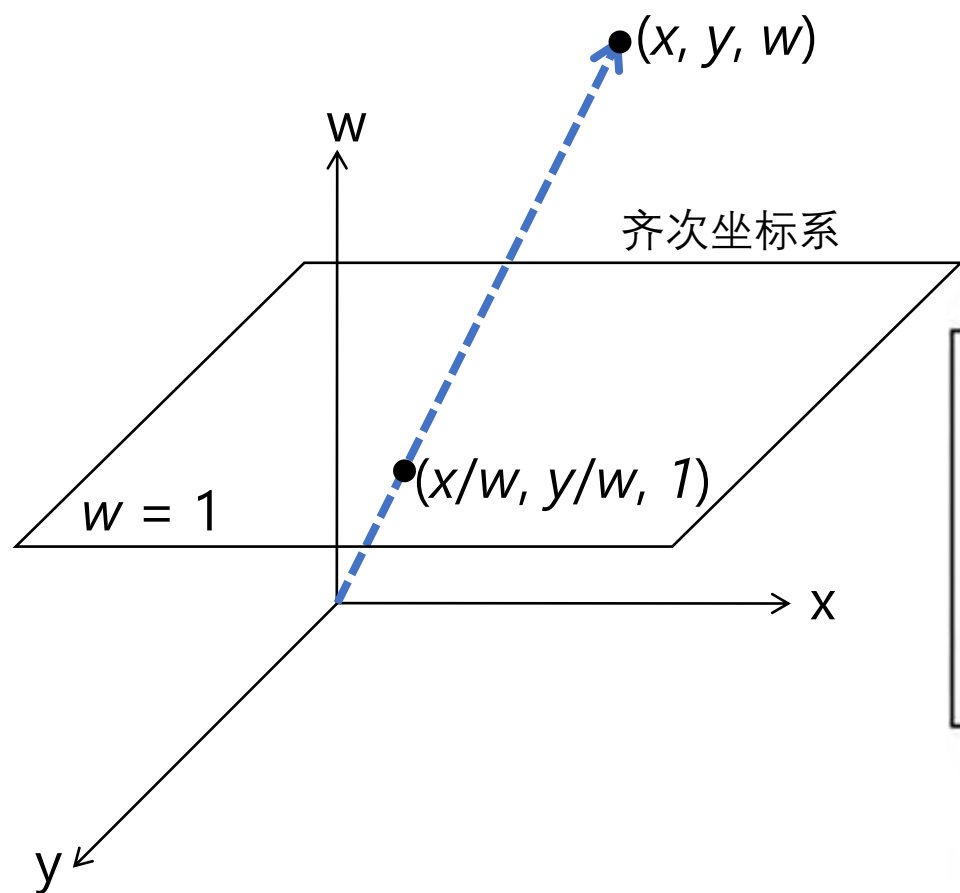
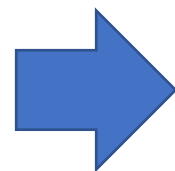


回顾——变换回归、 图像分类

求解同构映射



$$\begin{bmatrix}
 x_1 & y_1 & 1 & 0 & 0 & 0 & -x'_1 x_1 & -x'_1 y_1 & -x'_1 \\
 0 & 0 & 0 & x_1 & y_1 & 1 & -y'_1 x_1 & -y'_1 y_1 & -y'_1 \\
 & & & & & \vdots & & & \\
 x_n & y_n & 1 & 0 & 0 & 0 & -x'_n x_n & -x'_n y_n & -x'_n \\
 0 & 0 & 0 & x_n & y_n & 1 & -y'_n x_n & -y'_n y_n & -y'_n
 \end{bmatrix}
 \begin{bmatrix}
 h_{00} \\
 h_{01} \\
 h_{02} \\
 h_{10} \\
 h_{11} \\
 h_{12} \\
 h_{20} \\
 h_{21} \\
 h_{22}
 \end{bmatrix}
 =
 \begin{bmatrix}
 0 \\
 0 \\
 \vdots \\
 0 \\
 0
 \end{bmatrix}$$

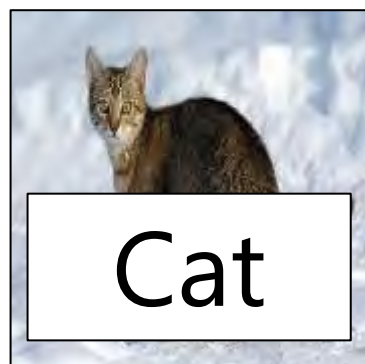
A
 $2n \times 9$

h
9

0
 $2n$

分类最简单的形式：最近邻

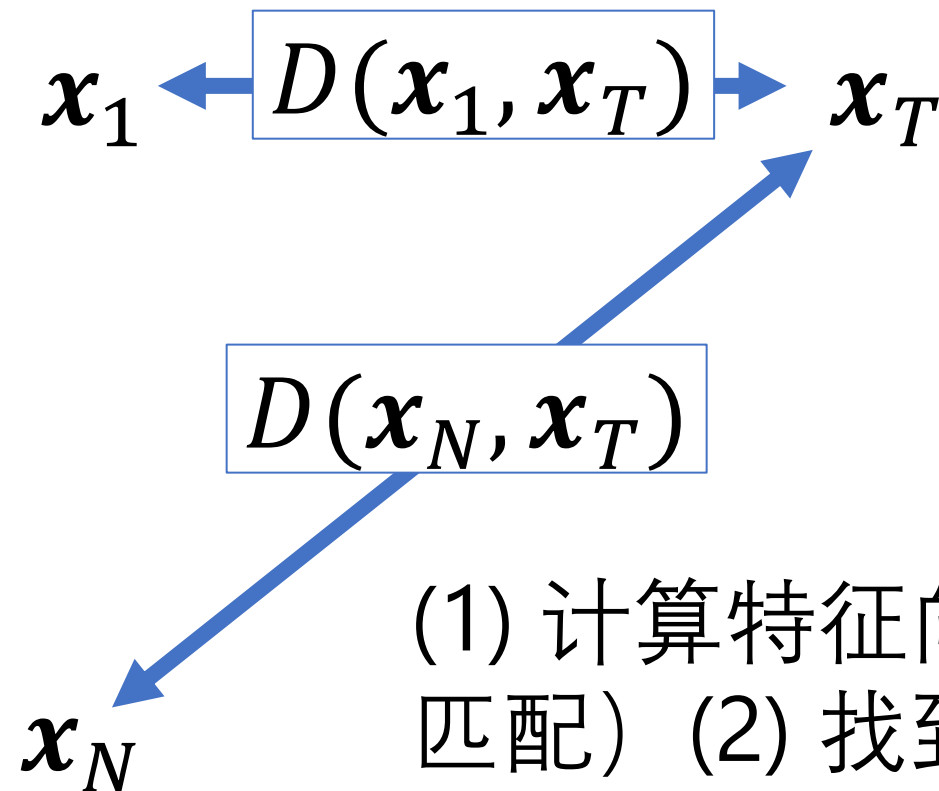
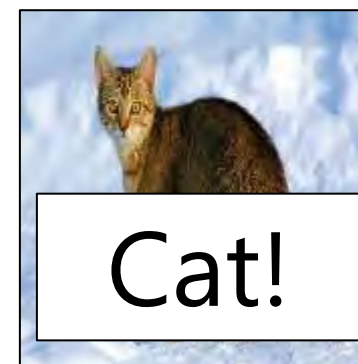
训练图像
和对应标签



...



测试图像

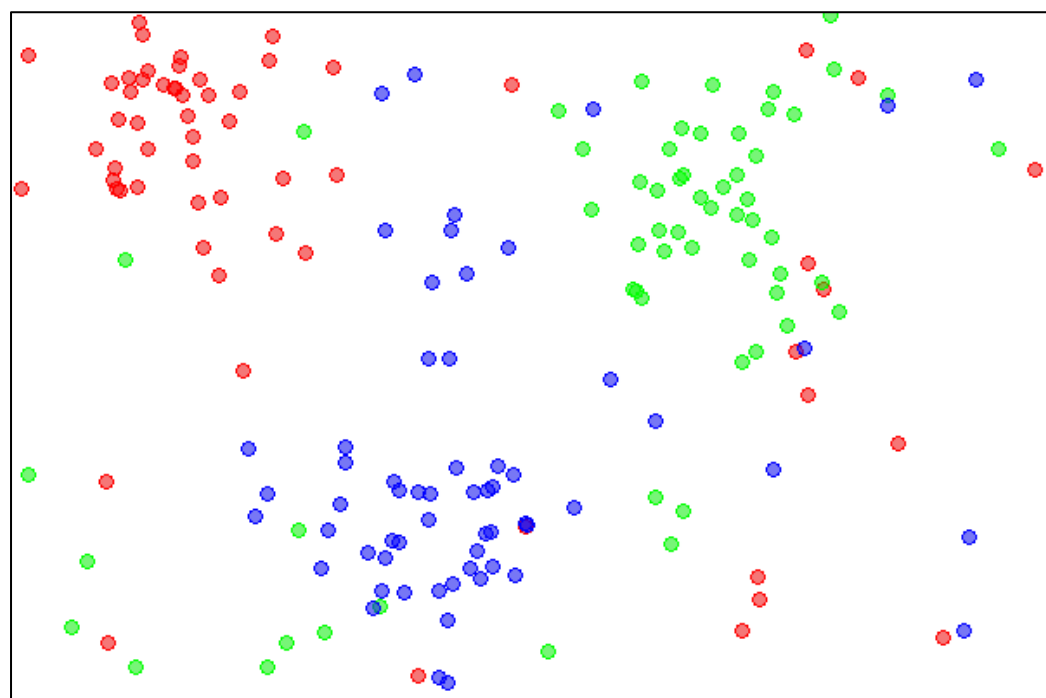


- (1) 计算特征向量的距离（特征匹配）
- (2) 找到训练集里最相似的样本
- (3) 使用最相似样本的标签。

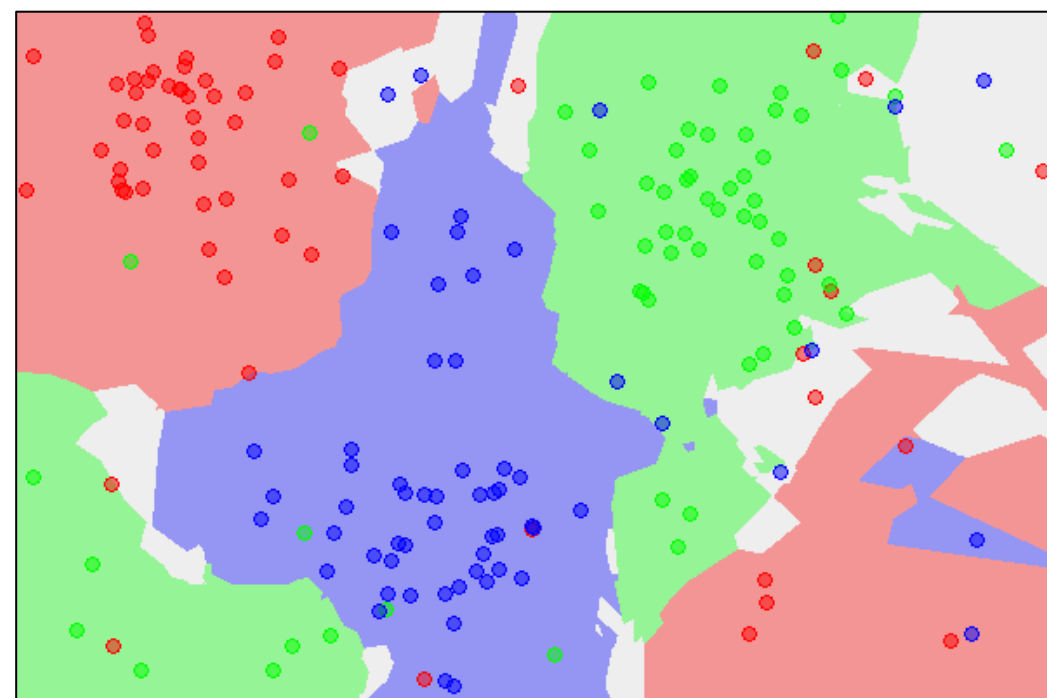
K近邻

找到前K近邻样本， 然后投票决定标签

2D Datapoints
(colors = labels)



2D Predictions
(colors = labels)



K近邻

- 虽然没有学习过程，但通常是有效的
 - 不学习参数和结构，没有实际的训练过程
- 对于每个任务都使用相同的算法
- 当数据点数量趋近于无穷时，错误率保证最多比数据上的最优解差2倍
 - **局部决策**：K-NN在决策时只考虑局部信息，即输入点的K个最近邻。因为它是基于局部信息做出决策的，所以当数据量很大时，它能够捕捉到数据的微小细节和模式。
 - **大数定律**：当训练数据量增加时，每个点的近邻都更加可能代表真实的数据分布。这意味着K-NN的决策边界变得更加准确。
 - 当K同时增长并且 K/N 趋近于0，其中N是数据点的数量
 - K-NN的错误率会收敛到贝叶斯最优错误率
 - 看似很好，但是有效率问题，且性能一般

线性模型

设定：三分类模型



模型：每个类别一个权重

$w_0^T \mathbf{x}$ big if cat

$w_1^T \mathbf{x}$ big if dog

$w_2^T \mathbf{x}$ big if hippo

w_0, w_1, w_2

全部参数：

$W_{3 \times F}$ where \mathbf{x} is in \mathbb{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.

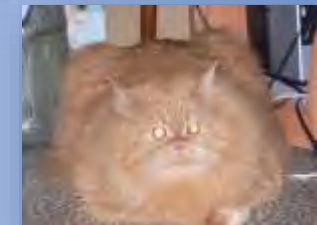
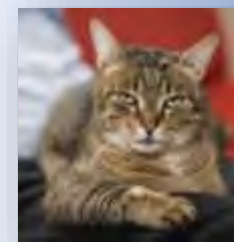
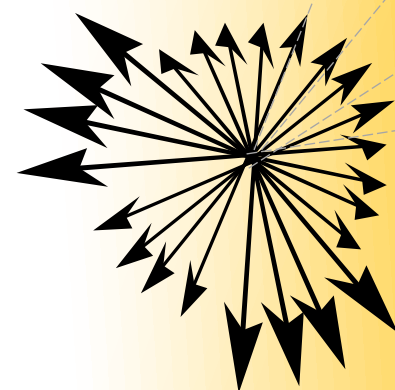


解释：几何角度

- 参数为每个类定义一个超平面:

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

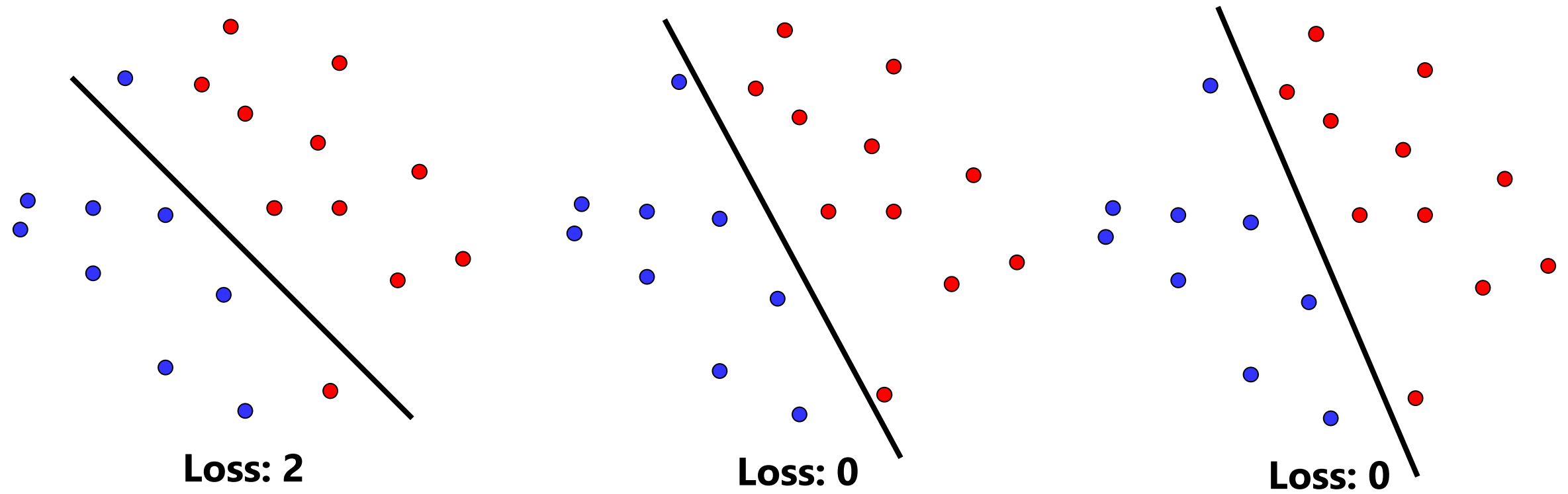
- 我们可以将每个类的得分视为定义了一个与其距离成正比分布, 距离是指从对应的超平面到点的距离。



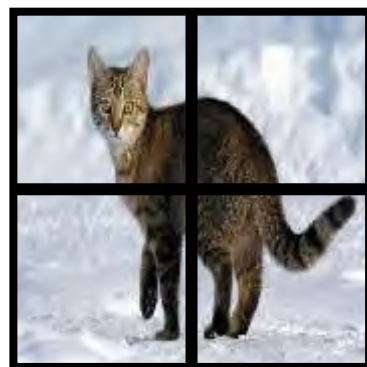
所有可能的图像构成的空间

什么是好的损失函数？

- 检视分类错的样本
 - 问题：样本是离散的，我们没办法区分相似的分类器
 - 我们需要更好的 **泛化性** *generalization*
 - 还要处理多于两类的情况



如何设计损失函数 —— 最大间距



Loss: dog score – cat score
怎么定义“dog” vs “cat”的评分？

$$(Wx)_2 - (Wx)_1$$

避免优化到负无穷

$$\max(0, (Wx)_2 - (Wx)_1)$$

这样有什么问题？

$$(Wx)_1$$

-96.8

Cat score

$$(Wx)_2$$

437.9

Dog score

$$(Wx)_3$$

61.95

Hippo score

Wx

预测结果是一个向量，
第j个值代表j类别的评分

Softmax

推断时 (x): $\arg \max_k (Wx)_k$ (找到评分最高的一类)

训练时 (x_i, y_i):

$$\arg \min_W \lambda \|W\|_2^2 + \sum_i^n -\log \left(\frac{\exp((Wx)_{y_i})}{\sum_k \exp((Wx)_k)} \right)$$

Regularization 正则项
对所有训练样本

对正确分类对应的 negative log-likelihood 进行损失优化


P(correct class)

Cross-entropy loss 交叉熵

$$f(x_i, W) = Wx_i \quad (\text{score function})$$

$$L_i = -\log \left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}} \right)$$

$P(y_i | x_i; W)$



$$L_i = -f_{y_i} + \log \sum_j e^{f_j}$$

我们称 L_i 为 交叉熵
cross-entropy loss

i.e. we're minimizing
the negative log
likelihood.

损失函数

- 交叉熵损失仅仅是一种可能的损失函数。
 - 它的一个好属性是它将得分重新解释为概率，这些概率具有自然的含义。
- 支持向量机（SVM，最大间距）损失函数过去也很受欢迎。
 - 但目前，交叉熵是最常见的分类损失。

怎么优化线性模型？

目标：找到参数 \mathbf{w} 最小化
损失函数 L .

$$\arg \min_{\mathbf{w} \in \mathbb{R}^N} L(\mathbf{w})$$

$$L(\mathbf{W}) = \lambda \|\mathbf{W}\|_2^2 + \sum_{i=1}^n -\log \left(\frac{\exp((\mathbf{W}\mathbf{x})_{y_i})}{\sum_k \exp((\mathbf{W}\mathbf{x})_k)} \right)$$

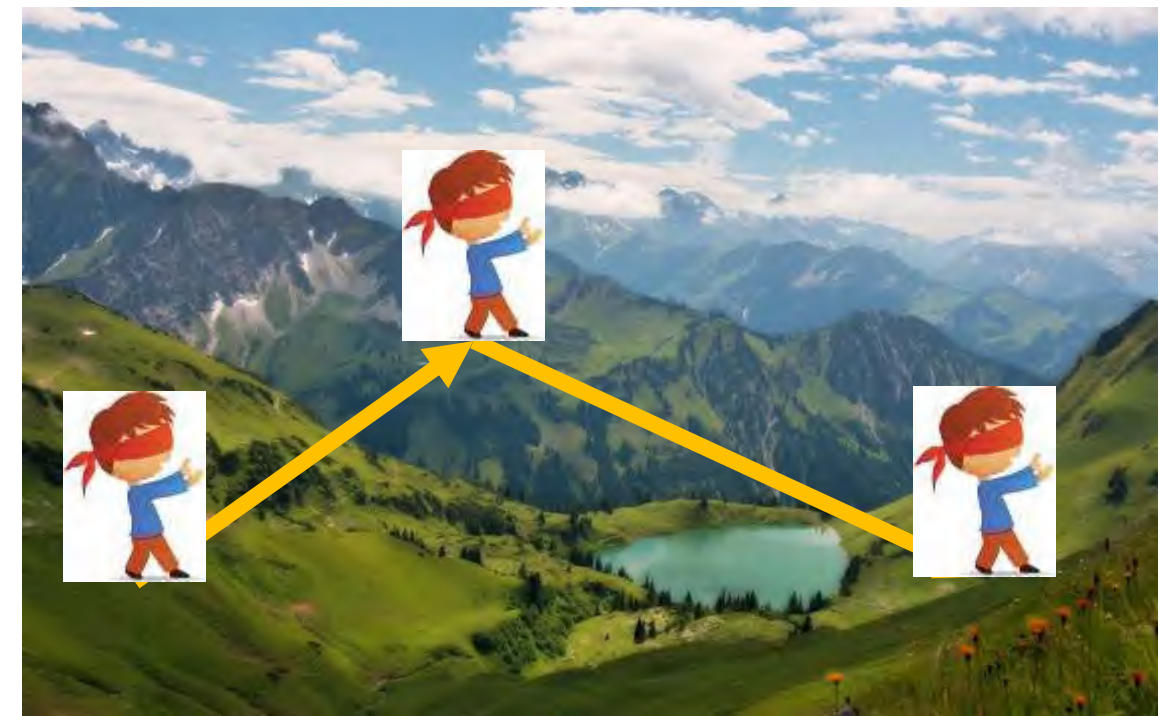
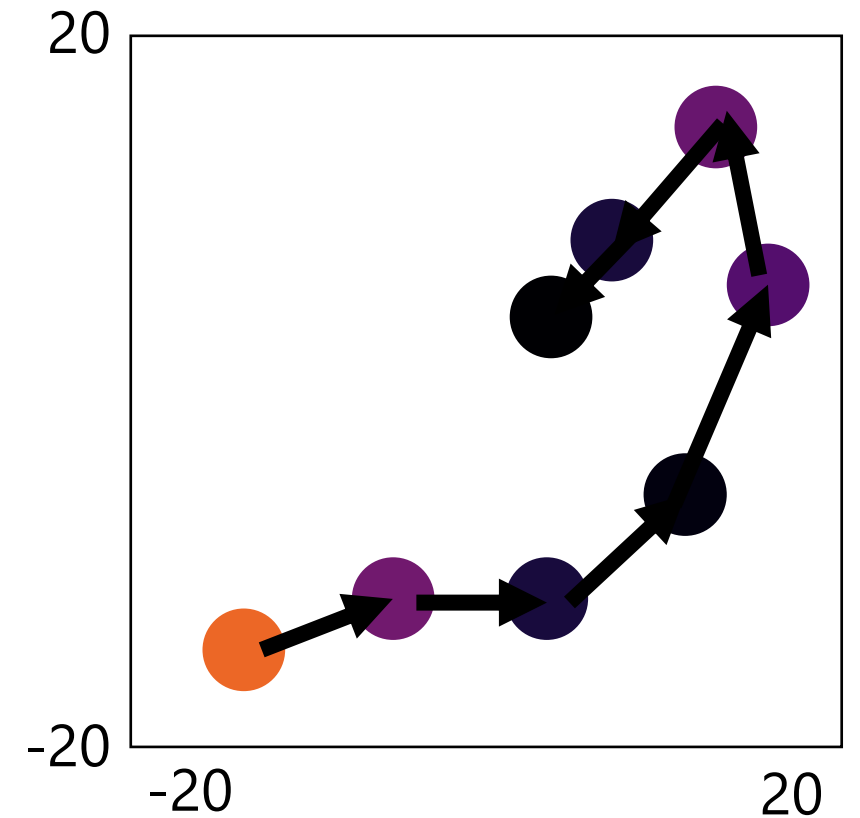
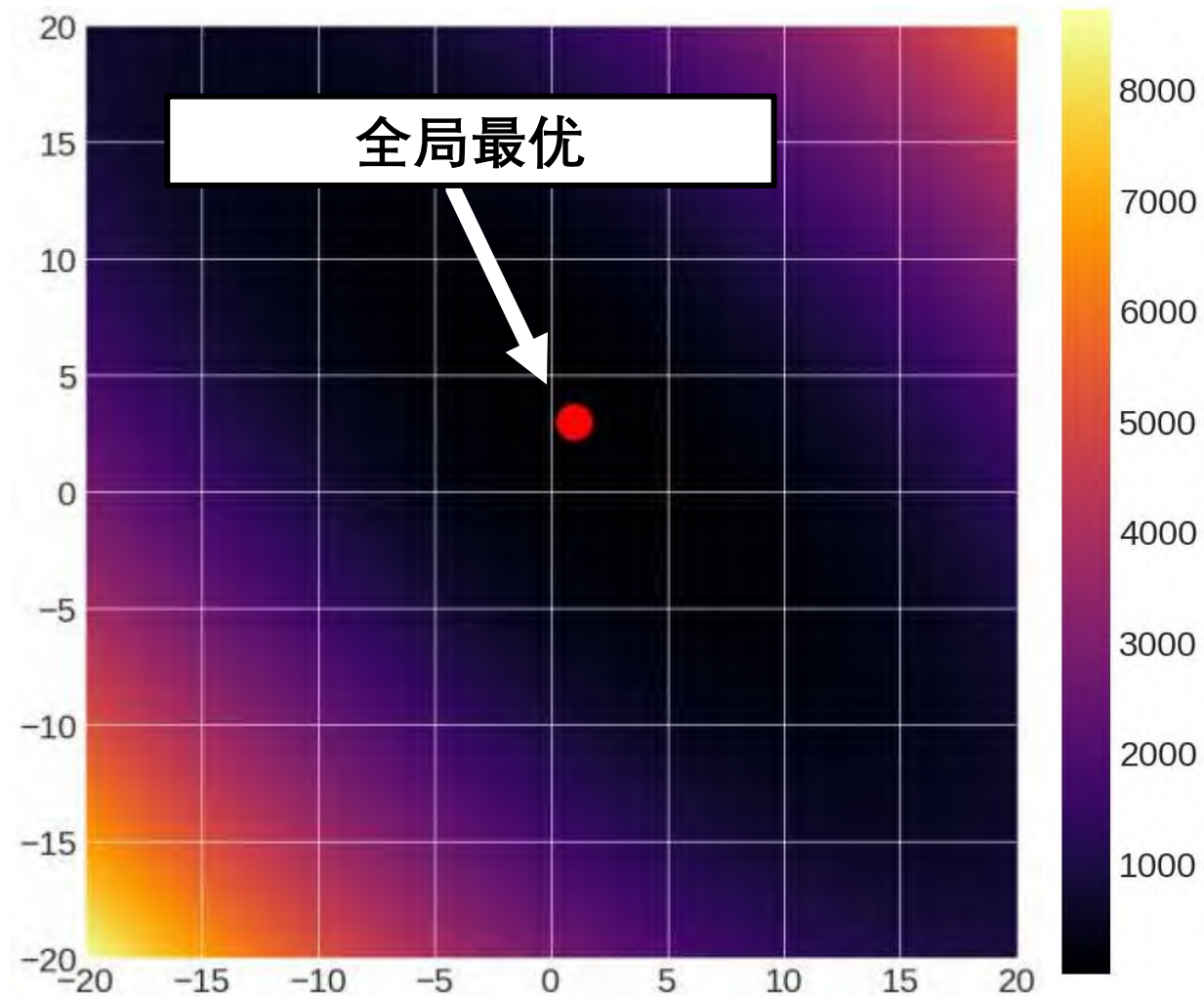
不同的损失函数L: $L(\mathbf{w}) = \lambda \|\mathbf{w}\|_2^2 + \sum_{i=1}^n (y_i - \mathbf{w}^T \mathbf{x}_i)^2$

$$L(\mathbf{w}) = C \|\mathbf{w}\|_2^2 + \sum_{i=1}^n \max(0, 1 - y_i \mathbf{w}^T \mathbf{x}_i)$$

怎么优化?

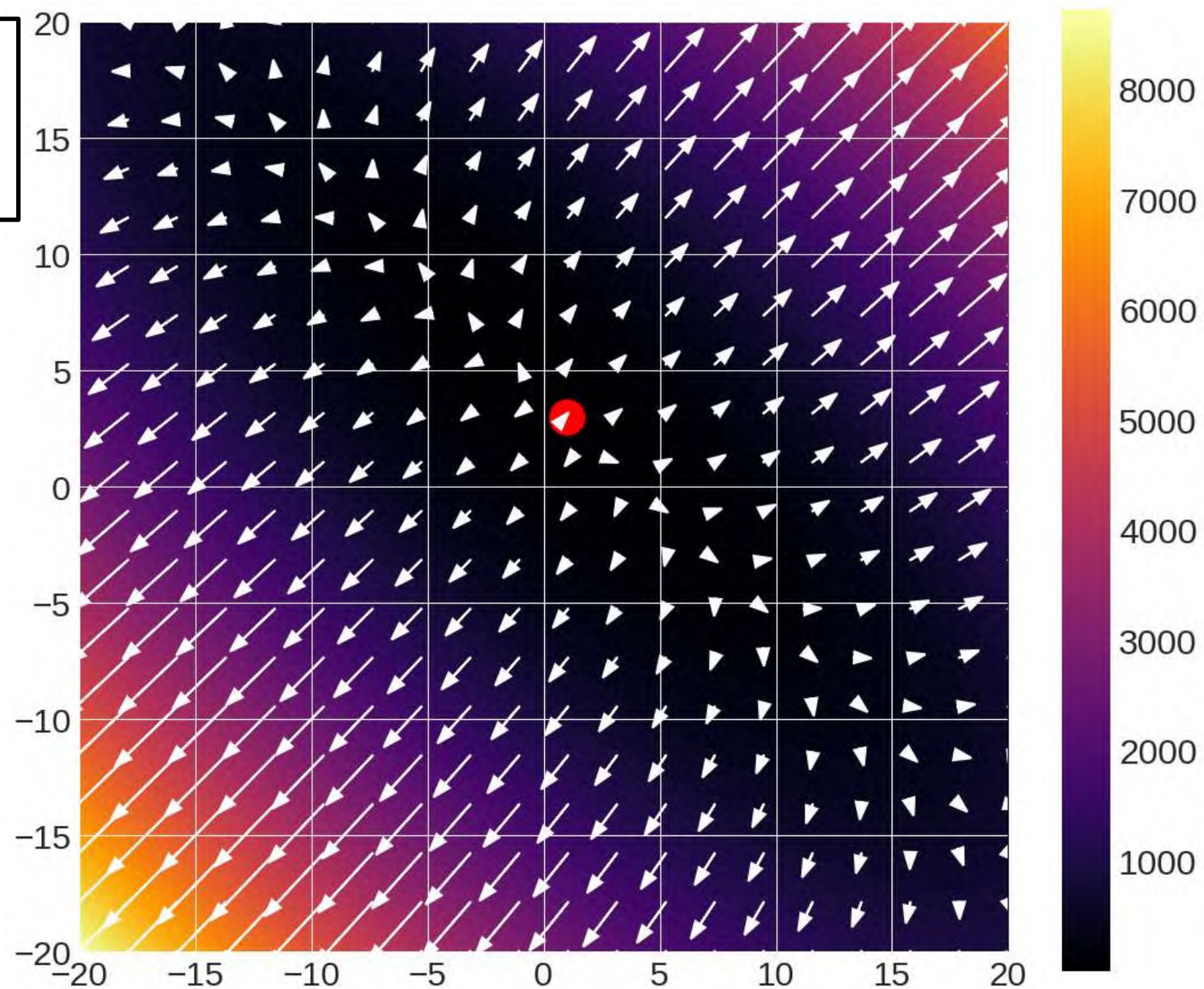
- 每个点代表函数的值
(大小参照右边热度图)
- 类似“爬山”与“下山”

$$f(x,y) = (x+2y-7)^2 + (2x+y-5)^2$$



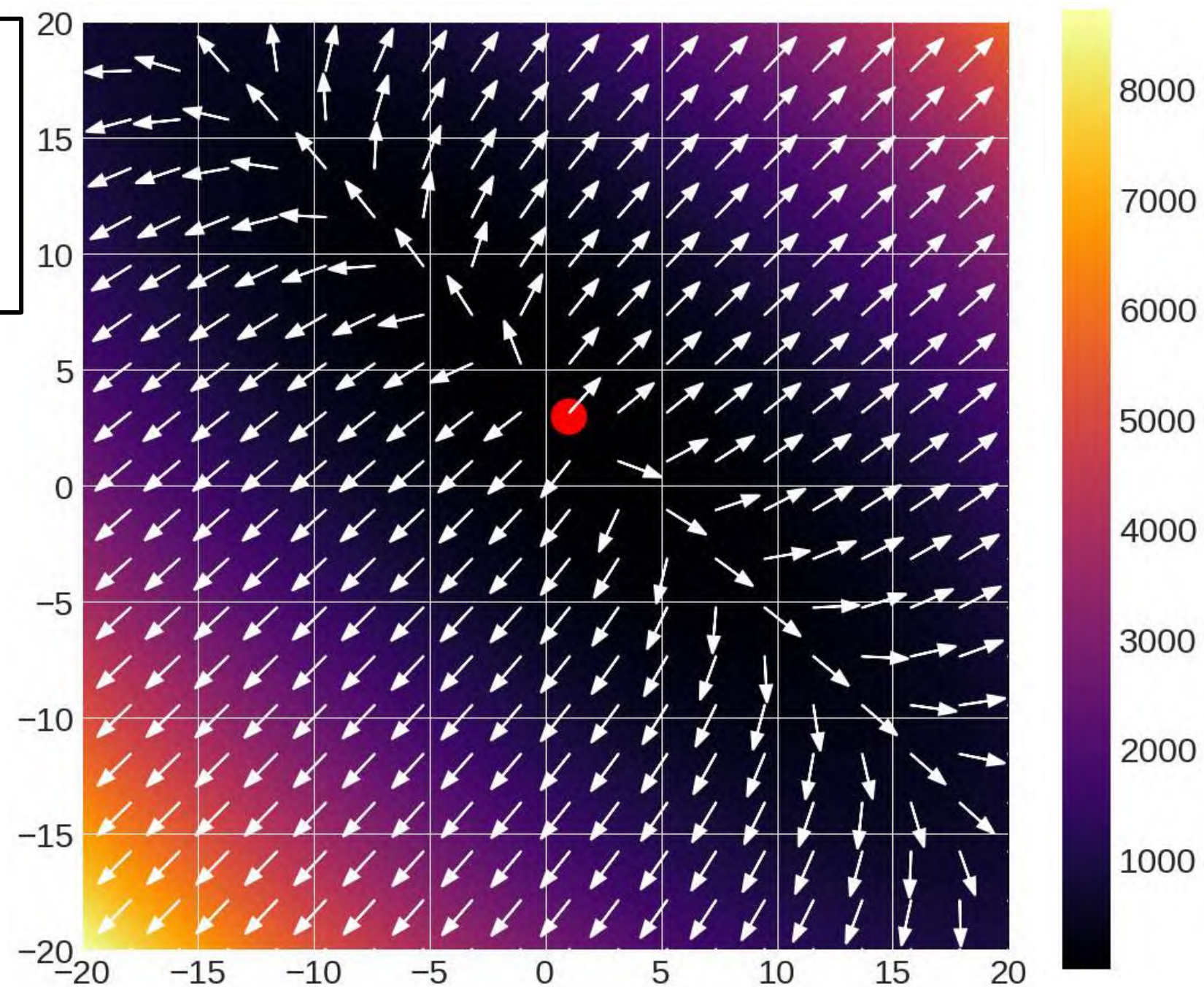
怎么优化？使用梯度

箭头：
梯度 **gradient**



怎么优化？使用梯度

箭头：
梯度方向
(单位长度)



怎么优化? 使用梯度

目标: $\arg \min_{\mathbf{w}} L(\mathbf{w})$

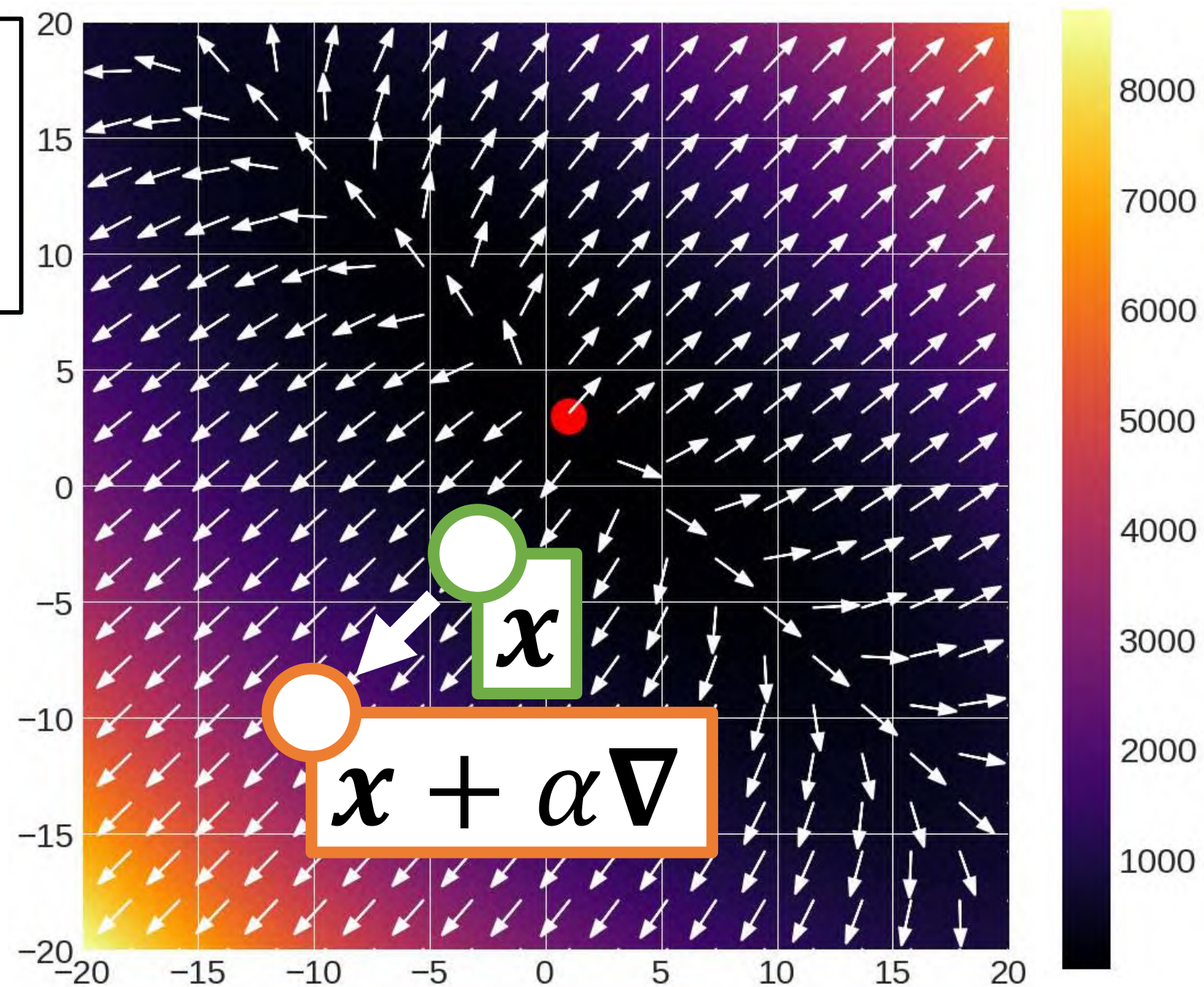
计算梯度: $\nabla_{\mathbf{w}} L(\mathbf{w}) = \begin{bmatrix} \partial L / \partial x_1 \\ \vdots \\ \partial L / \partial x_N \end{bmatrix}$

哪个更大(当 α 较小时)?

$$L(\mathbf{w}) \begin{array}{l} \leq? \\ >? \end{array} L(\mathbf{w} + \alpha \nabla_{\mathbf{w}} L(\mathbf{w}))$$

怎么优化？使用梯度

箭头：
梯度方向
(单位长度)



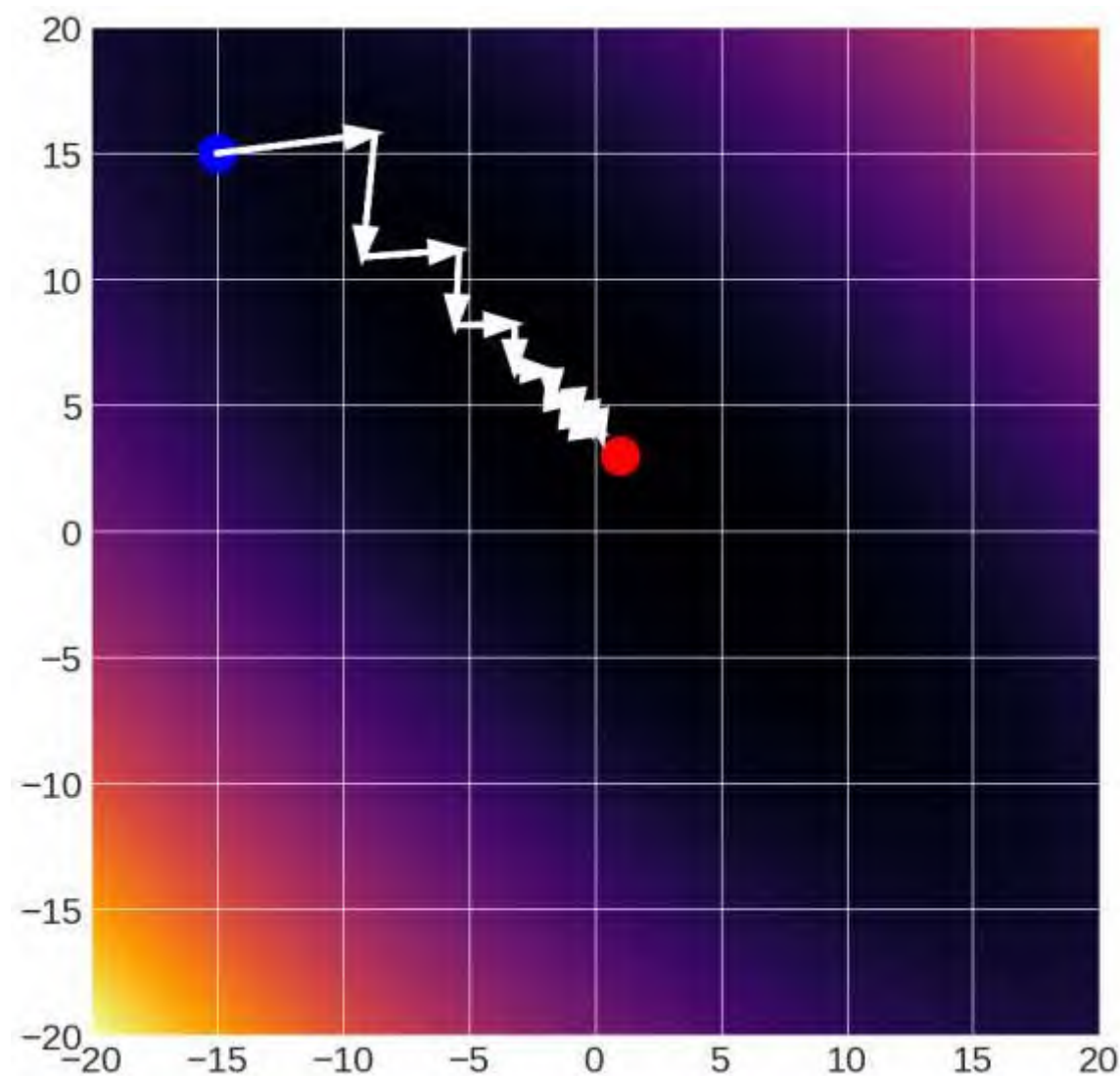
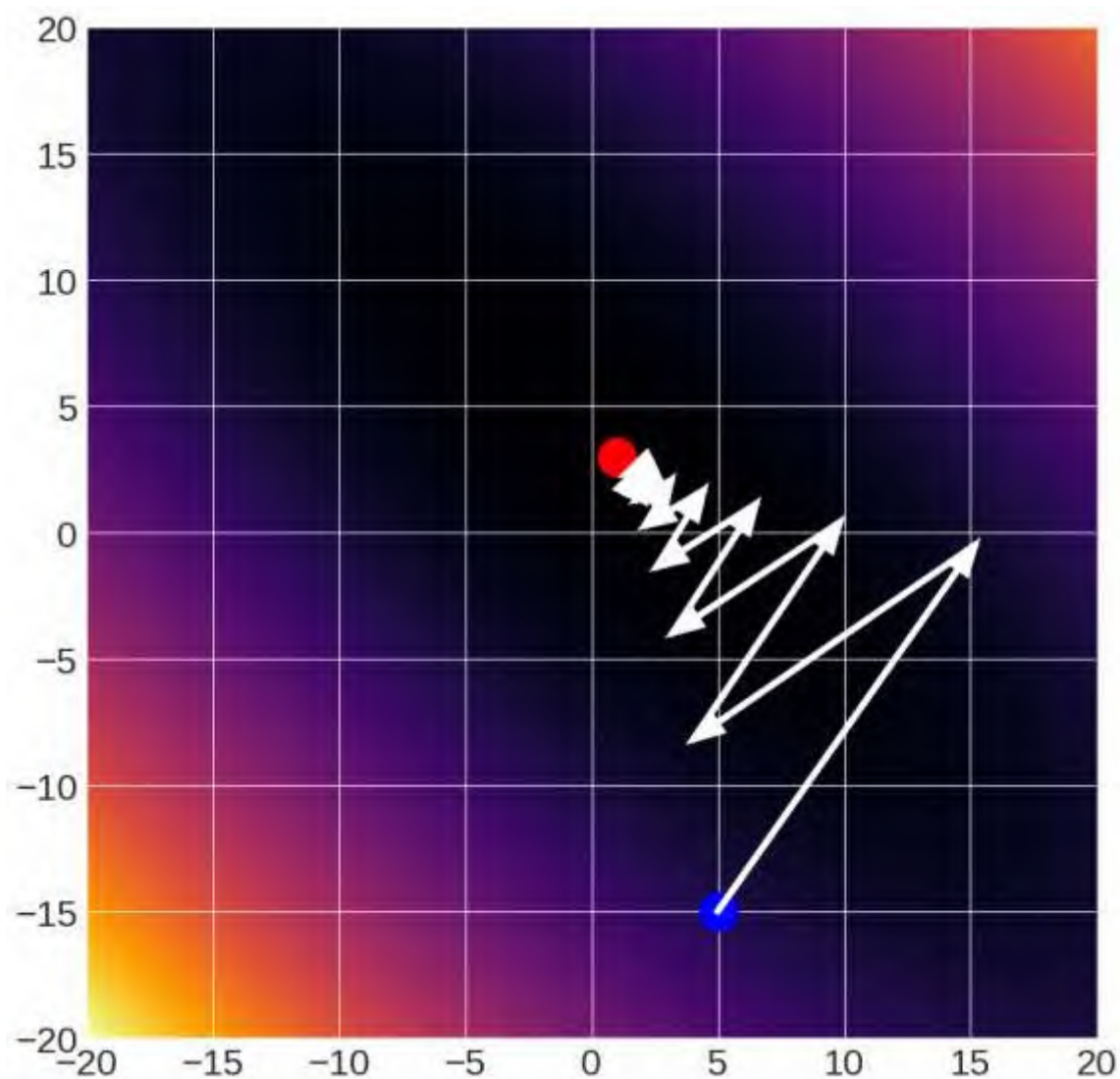
梯度下降

每一步优化，向梯度的反方向移动

```
w0 = initialize() #initialize  
for iter in range(numIters):  
    g =  $\nabla_{\mathbf{w}}L(\mathbf{w})$  #eval gradient  
    w = w + -stepsize(iter)*g #update w  
return w
```

梯度下降

基于蓝色起始点, $w_{i+1} = w_i + -9.8 \times 10^{-2} \times \text{gradient}$



梯度下降

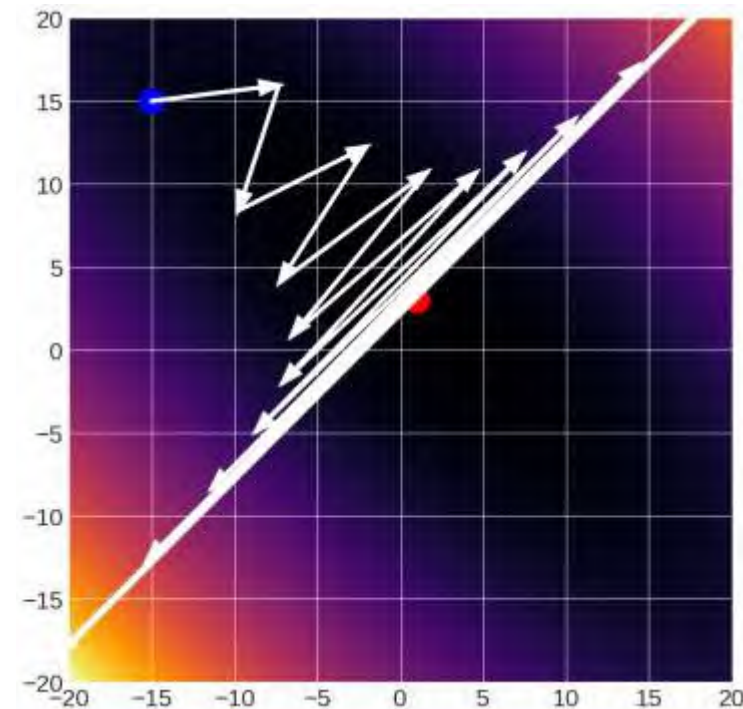
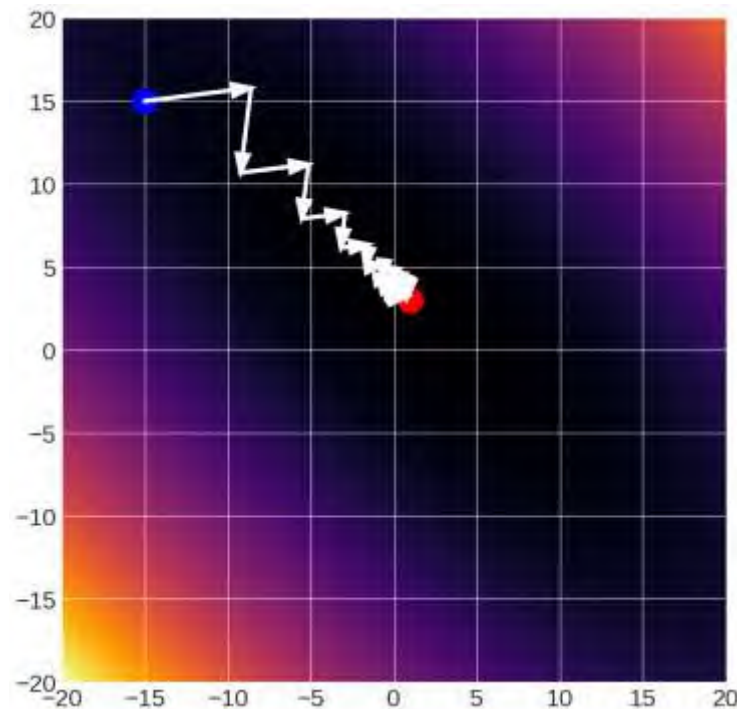
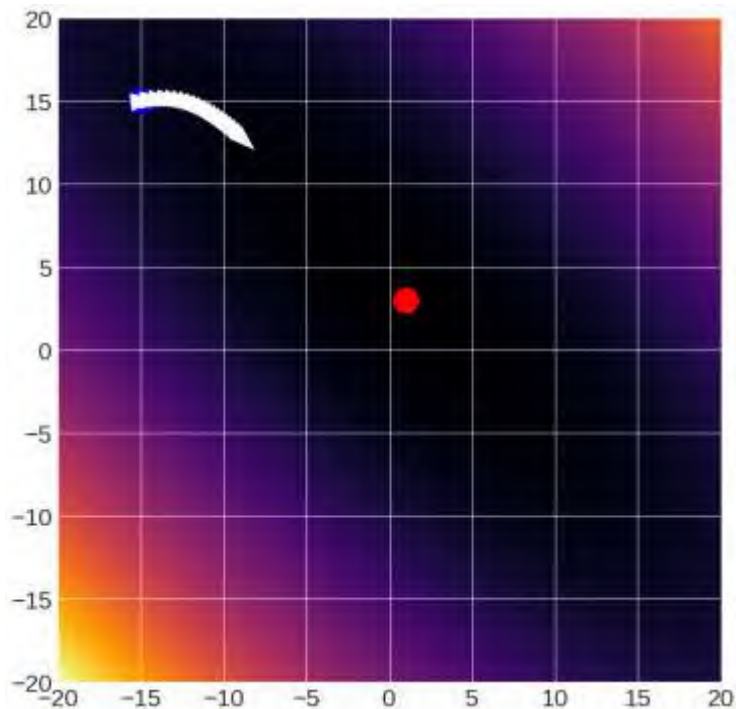
步长 Step size (也叫做 学习率 **learning rate / lr**)

十分重要的参数

1×10^{-2}
falls short

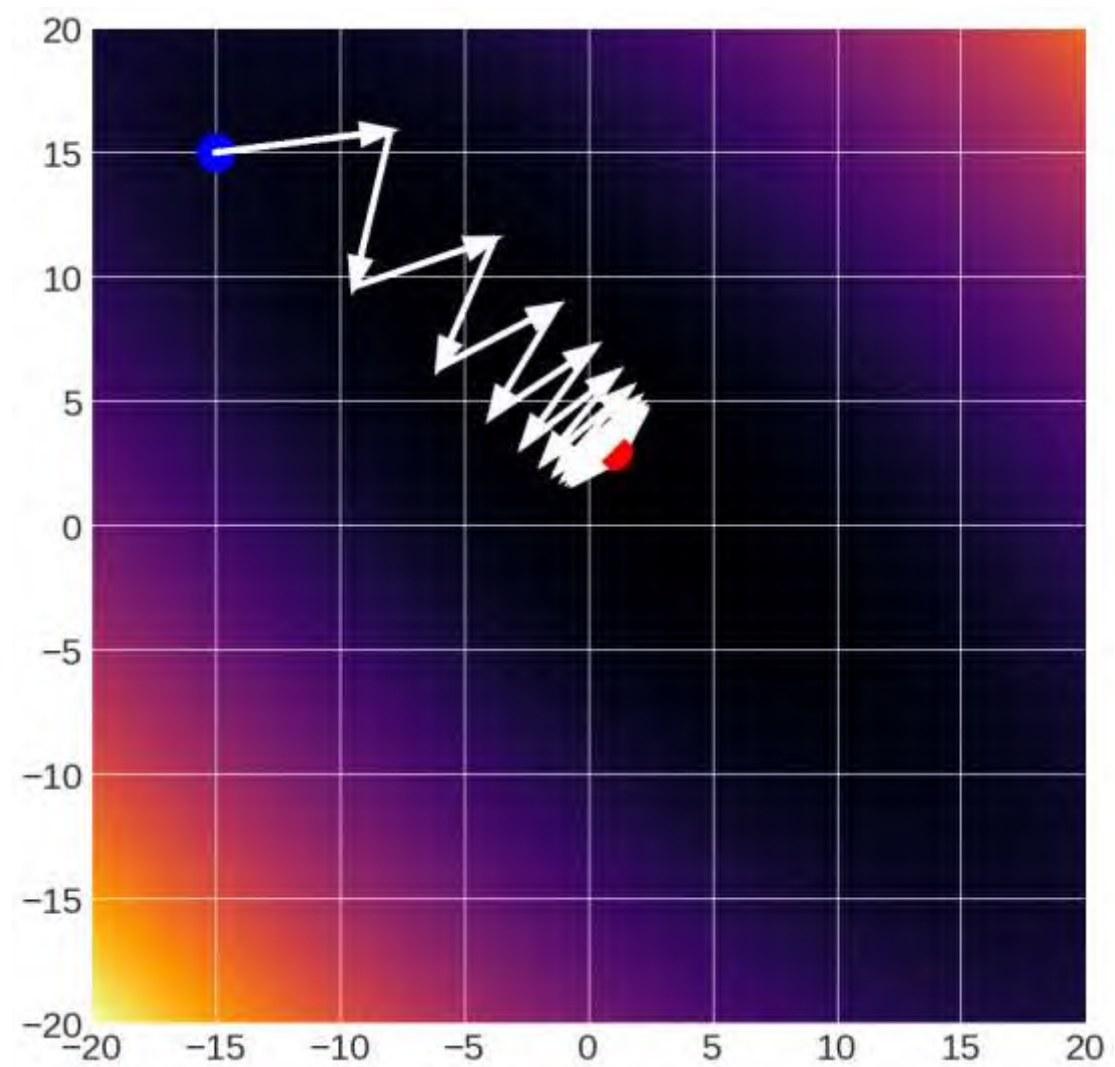
converges

12×10^{-2}
diverges



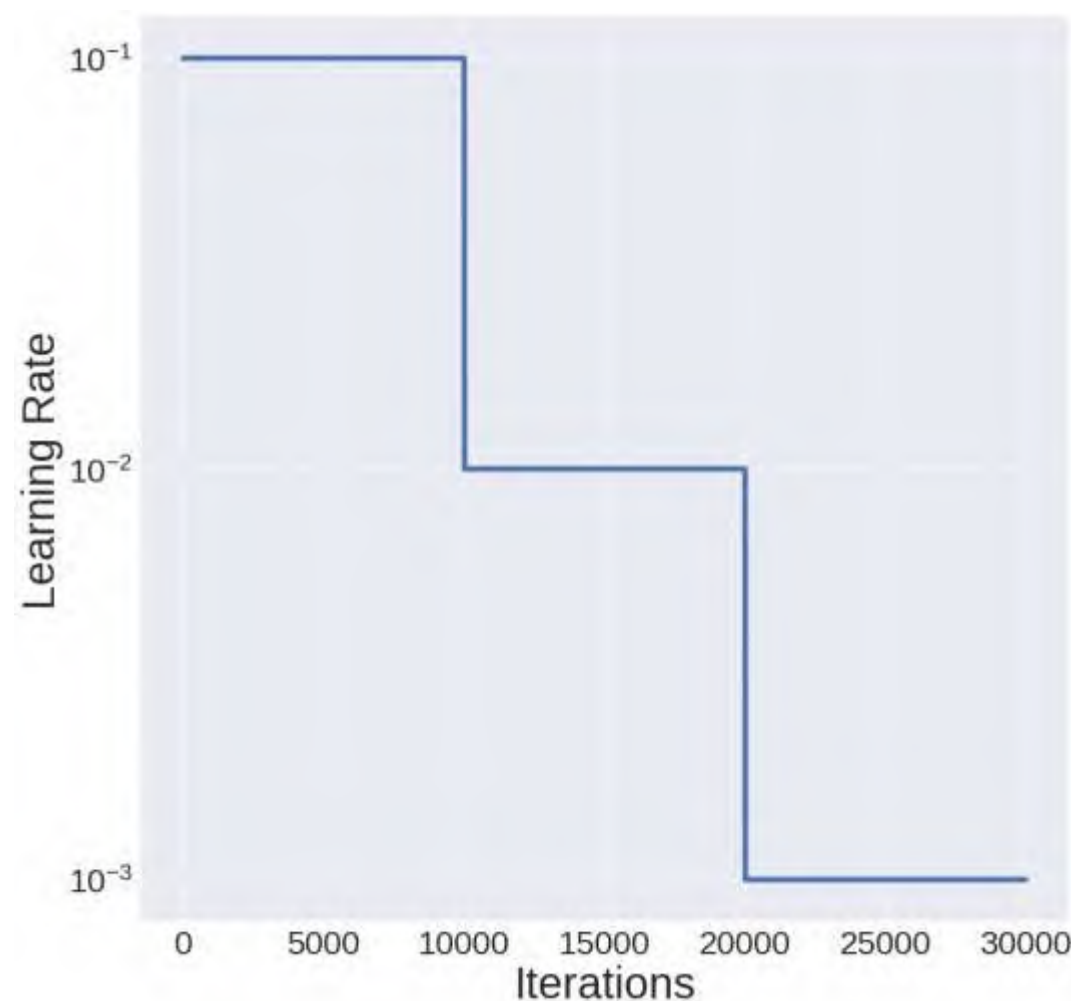
梯度下降

11×10^{-2} : oscillates
(Raw gradients)



梯度下降

使用初始的 lr, 每N步以后乘以f



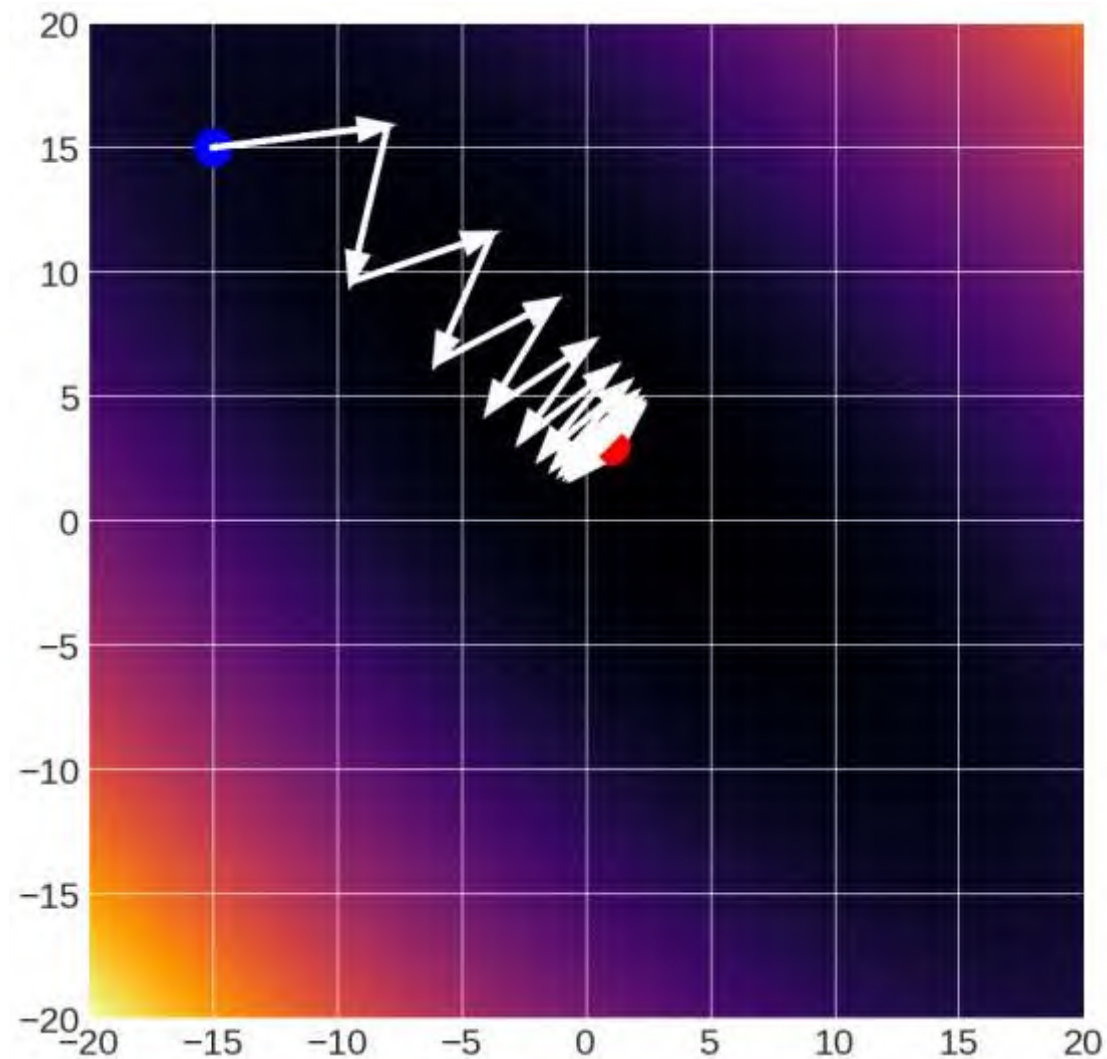
init_lr = 10^{-1}

f = 0.1

N = 10K

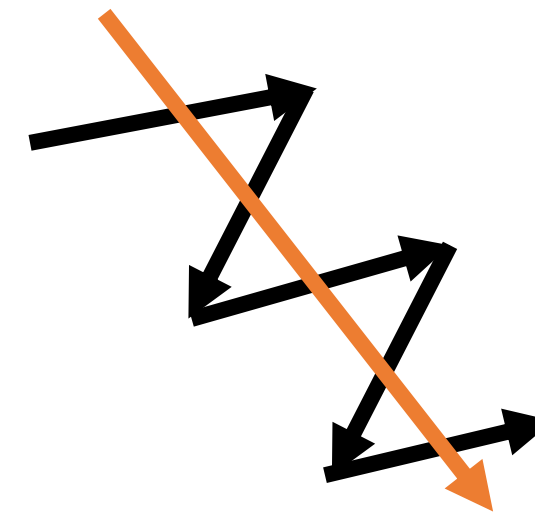
梯度下降

11×10^{-2} : oscillates
(Raw gradients)



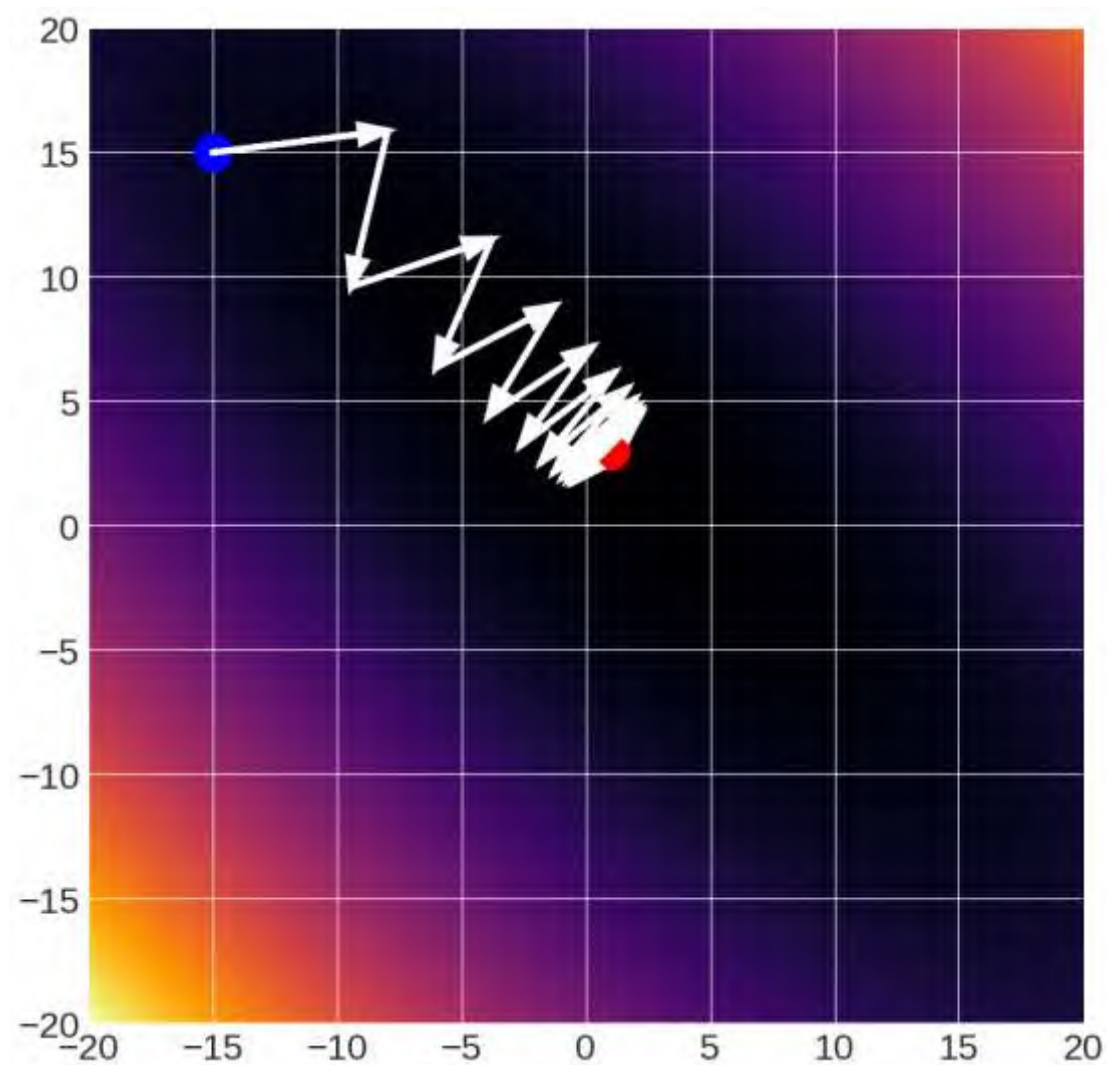
“平均梯度”

记录历史梯度，并且与当前步梯度取加权平均，也被称为动量法 “momentum”

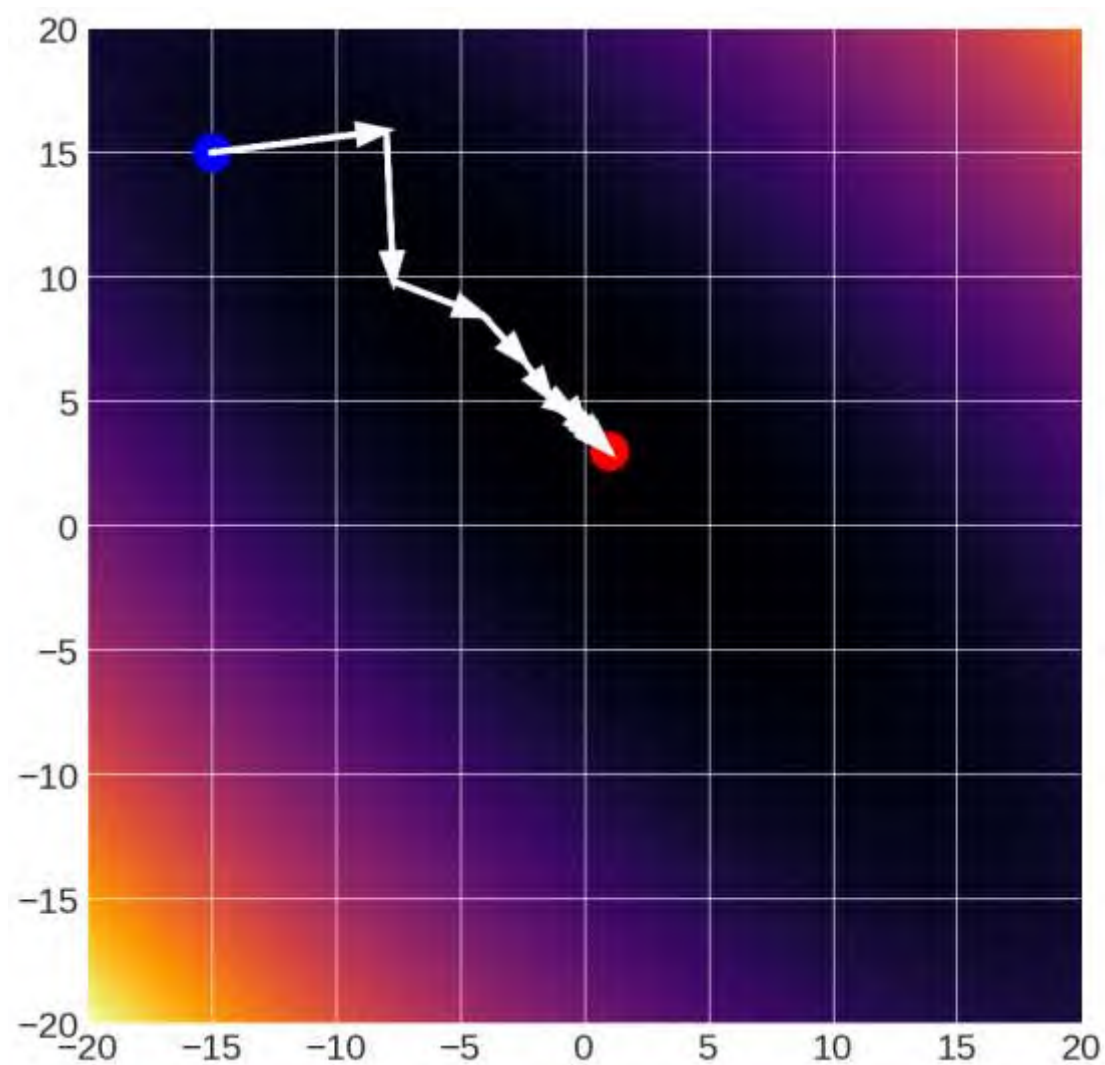


梯度下降

11×10^{-2} : oscillates
(Raw gradients)

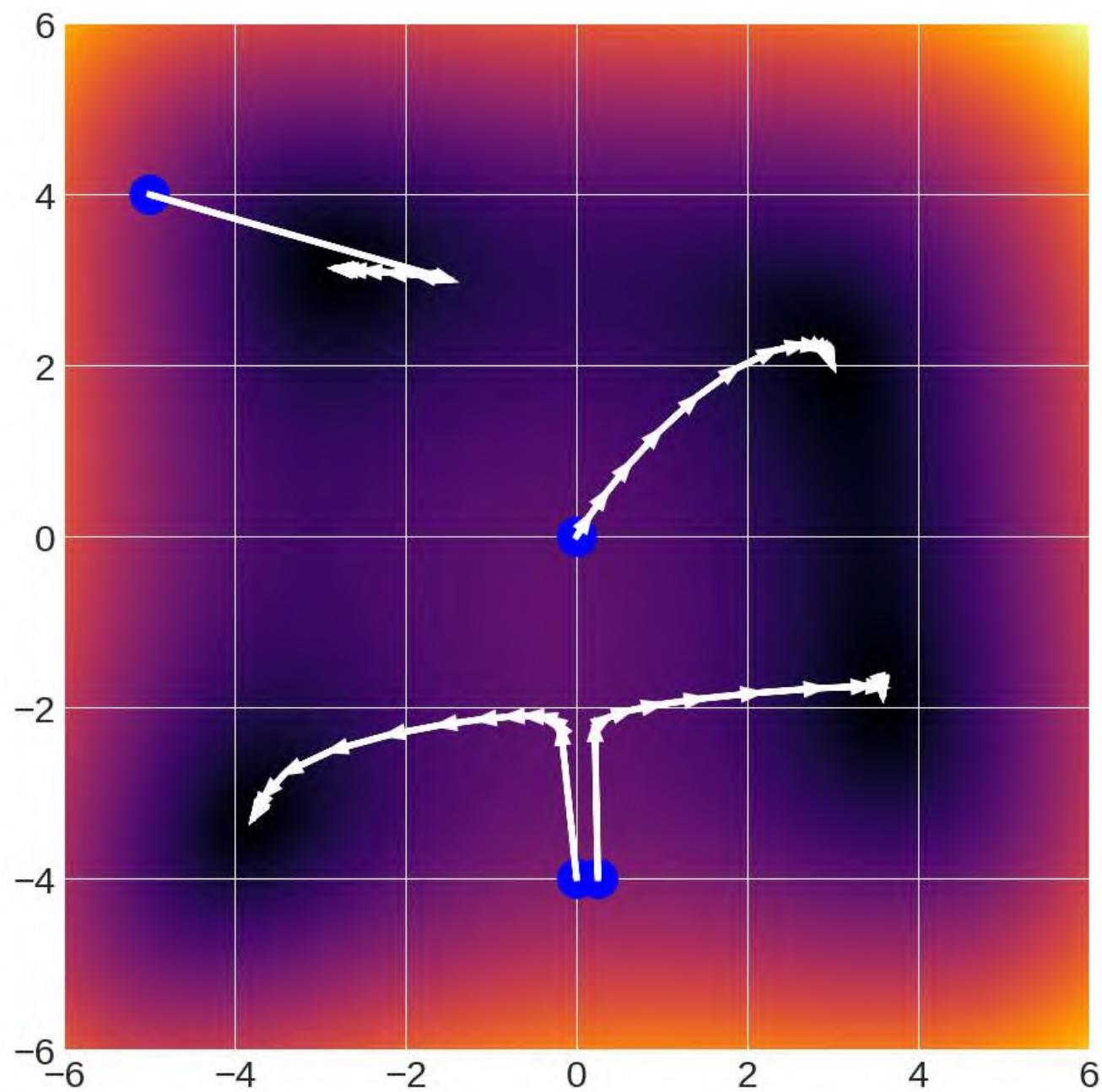


11×10^{-2}
(0.25 momentum)



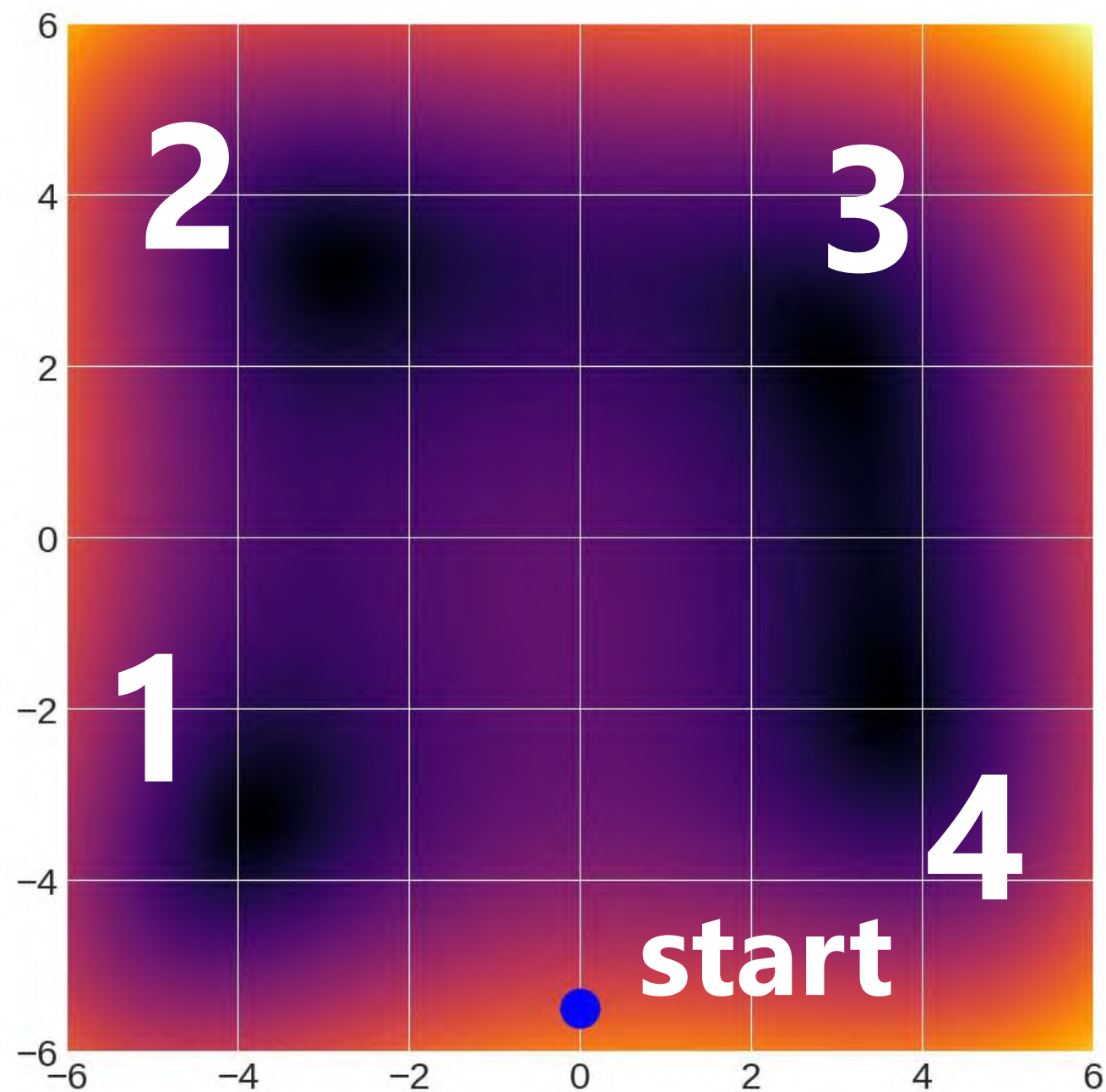
梯度下降

多个极小值时
→
梯度下降会找到
局部极小



梯度下降

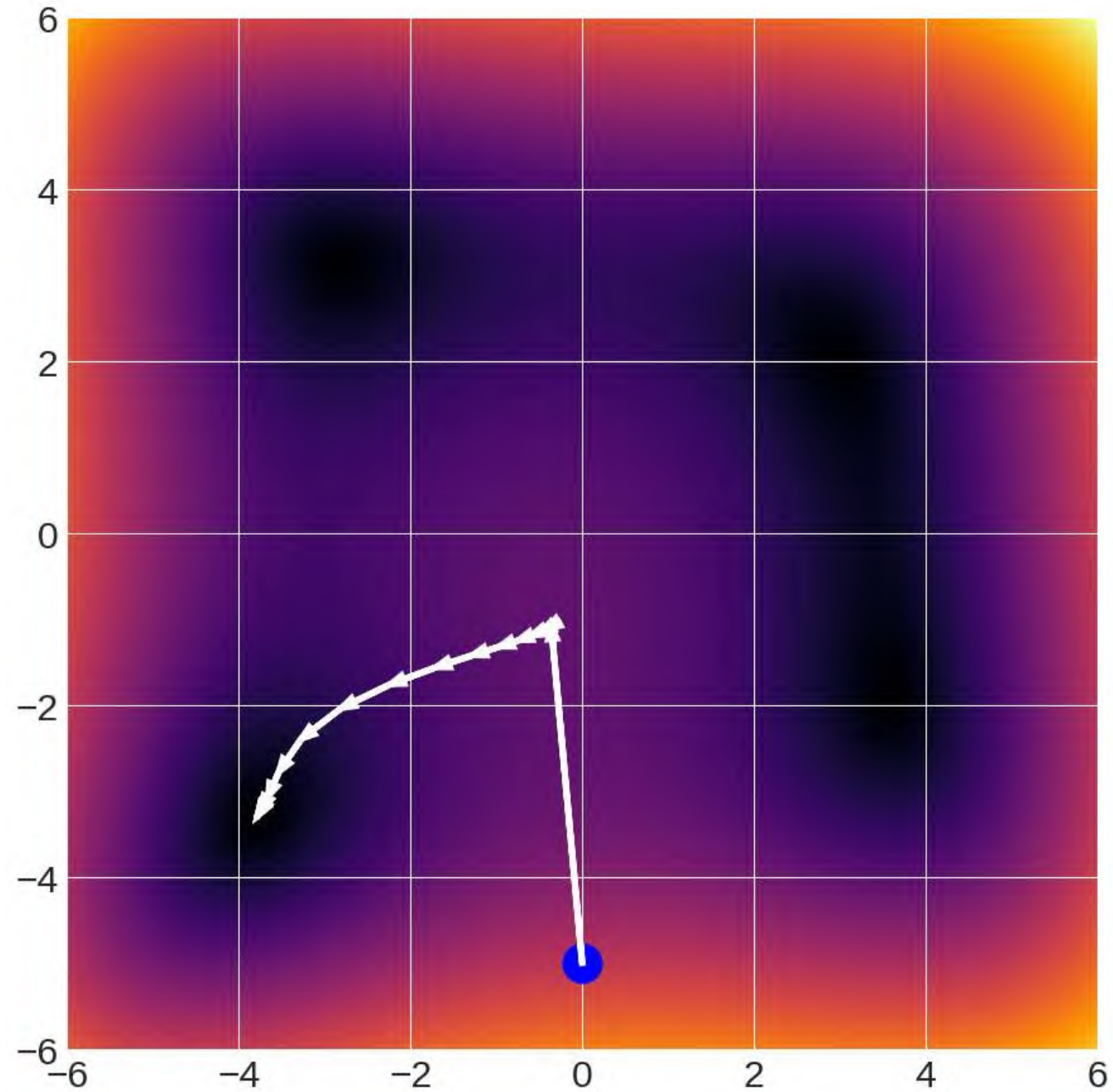
如果我们从蓝色点
开始，会走向哪个
局部最小？



梯度下降

许多函数是凸的
convex: 局部最小
就是全局最小

但是大多数函数并
不是



经验

- **一般来说:**小批量随机梯度下降 (SGD) + 动量+ 逐步调整的学习率
- 存在其他的更新规则 (例如, AdamW) 。在某些问题上, 它们经常表现得更好。

损失函数

- 交叉熵损失仅仅是一种可能的损失函数。
 - 它的一个好属性是它将得分重新解释为概率，这些概率具有自然的含义。
- 支持向量机（SVM，最大间距）损失函数过去也很受欢迎。
 - 但目前，交叉熵是最常见的分类损失。
- 交叉熵可能更容易过拟合，特别是当模型在训练数据上获得非常高的分类准确性时。
- 最大间距可能更加健壮，因为它关注的是那些难以正确分类的样本，并试图确保它们被正确分类，而不是关注所有样本。

过拟合、欠拟合与模型复杂度

多项式回归: 给定 x , 预测 y

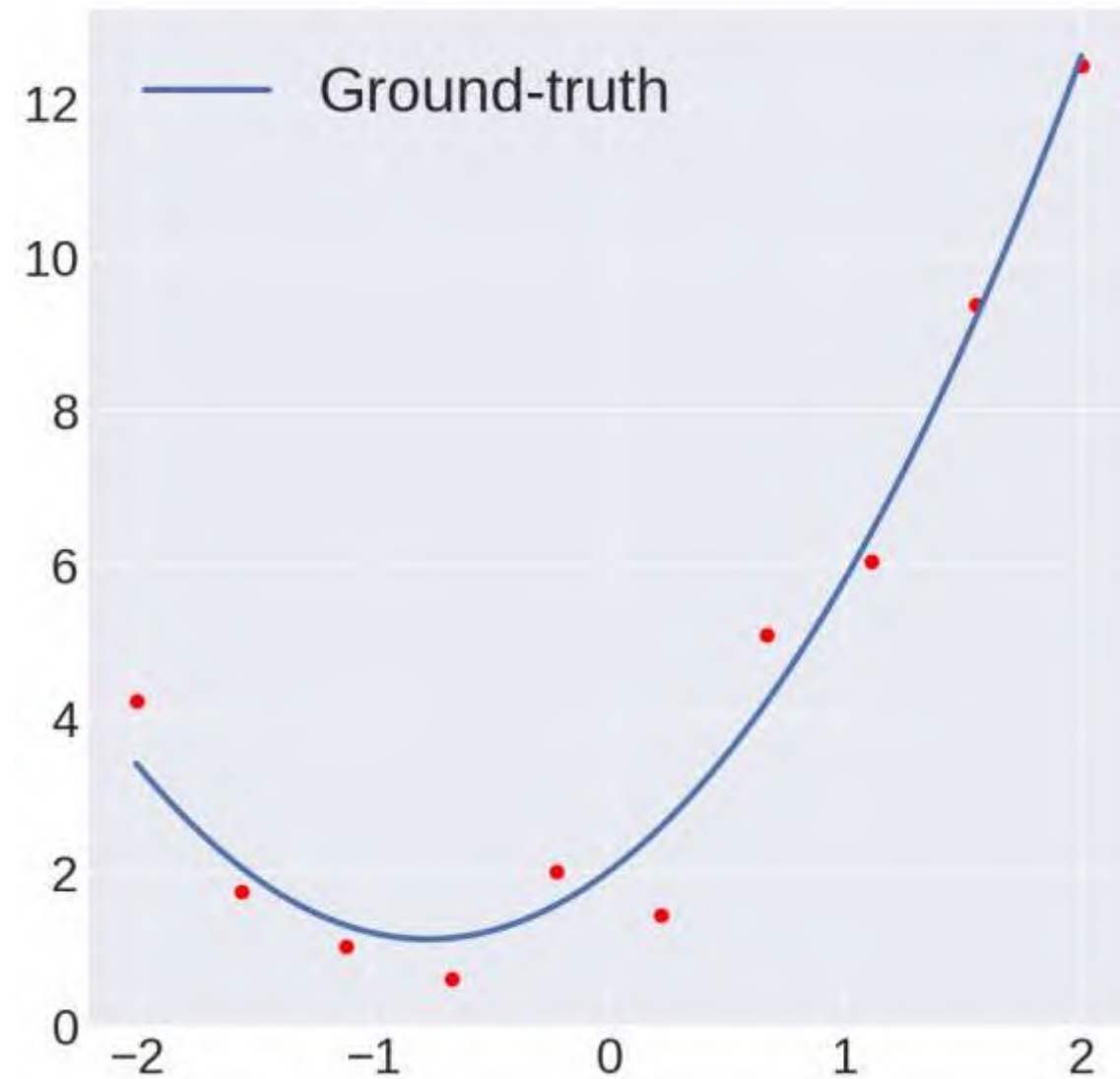
$$\begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} x_1^F & \cdots & x_1^2 & x_1 & 1 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ x_N^F & \cdots & x_N^2 & x_N & 1 \end{bmatrix} \begin{bmatrix} w_F \\ \vdots \\ w_2 \\ w_1 \\ w_0 \end{bmatrix}$$

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, 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。

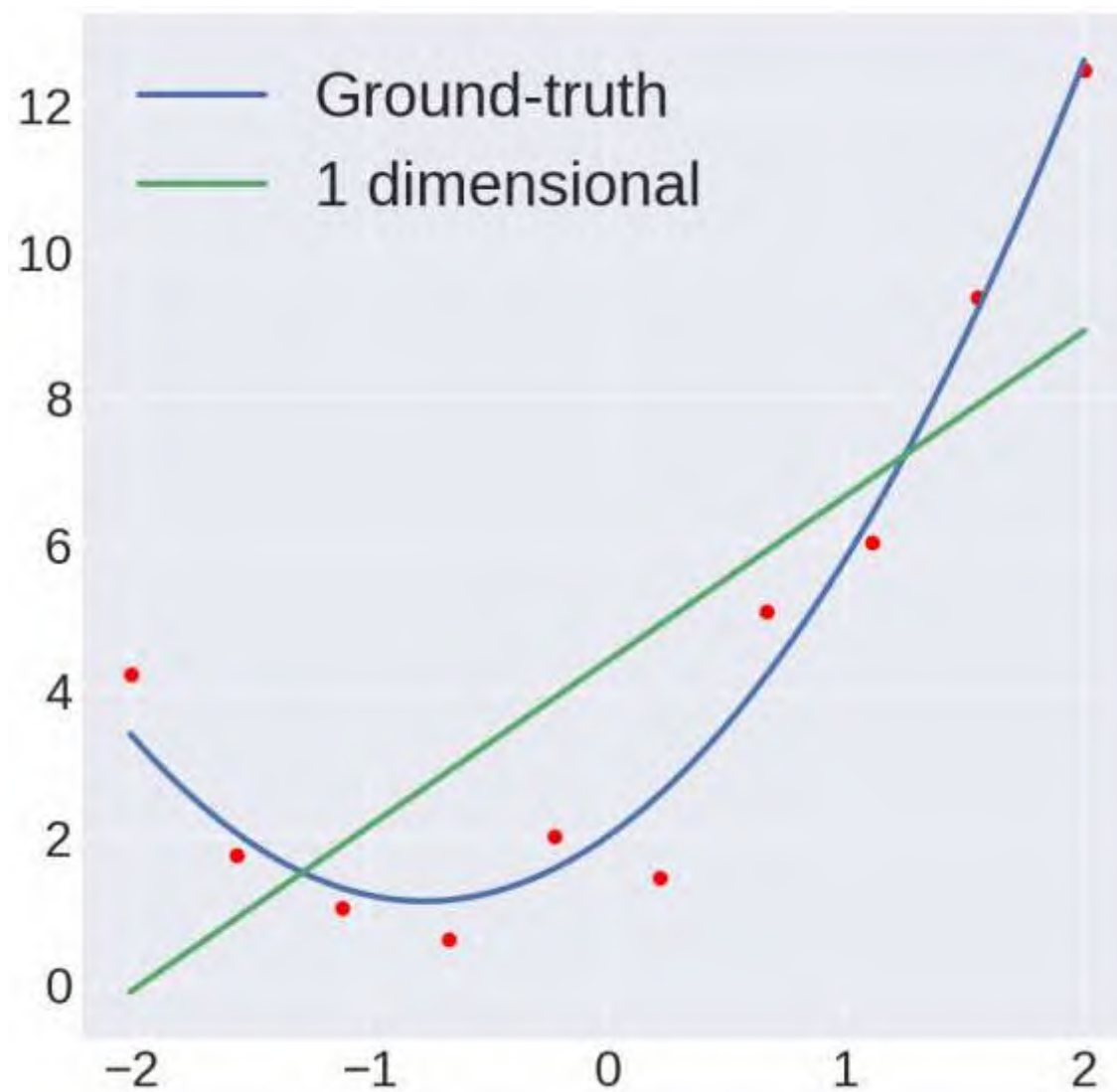
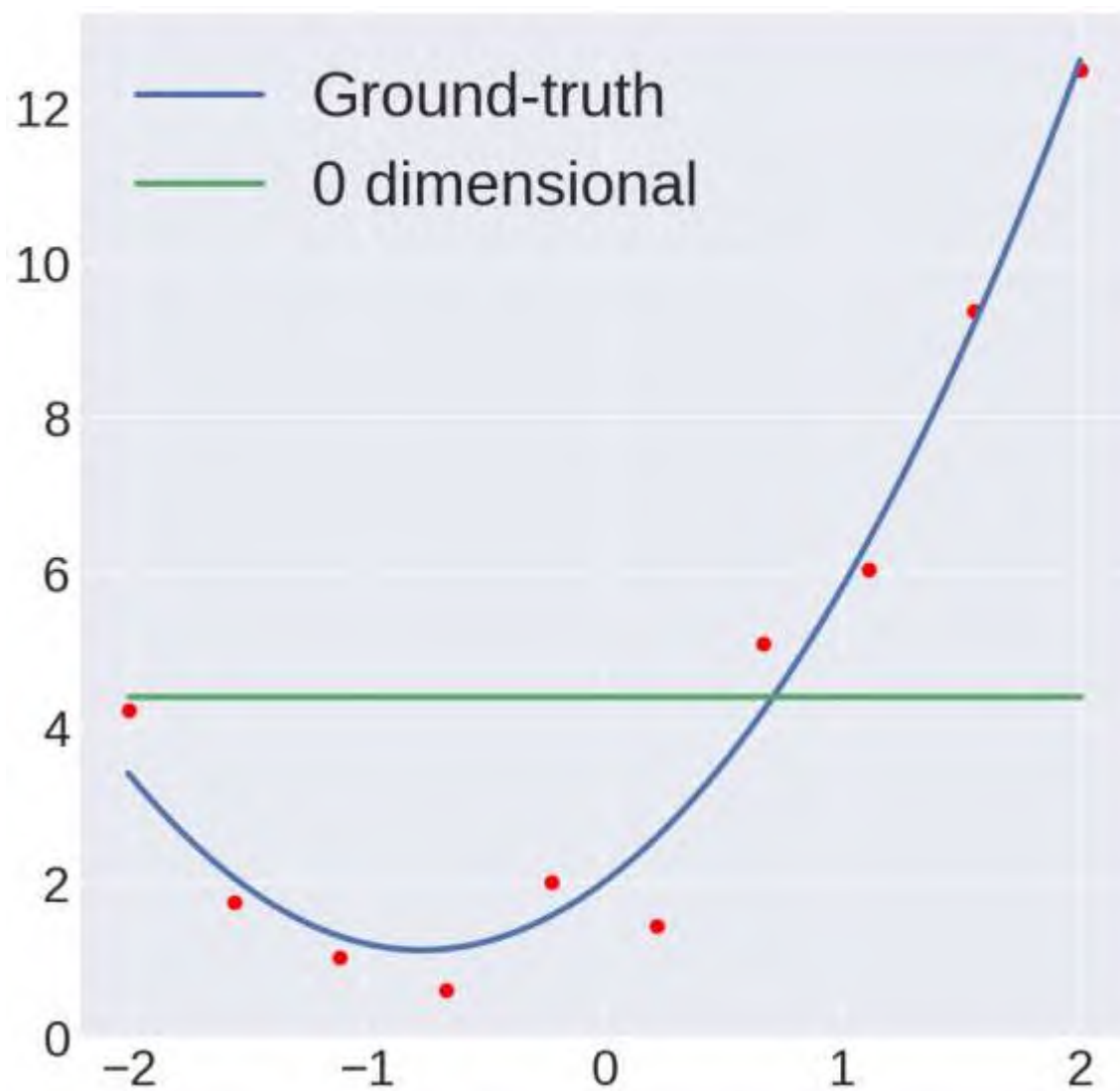
过拟合、欠拟合与模型复杂度

Model: $1.5x^2 + 2.3x + 2 + N(0, 0.5)$

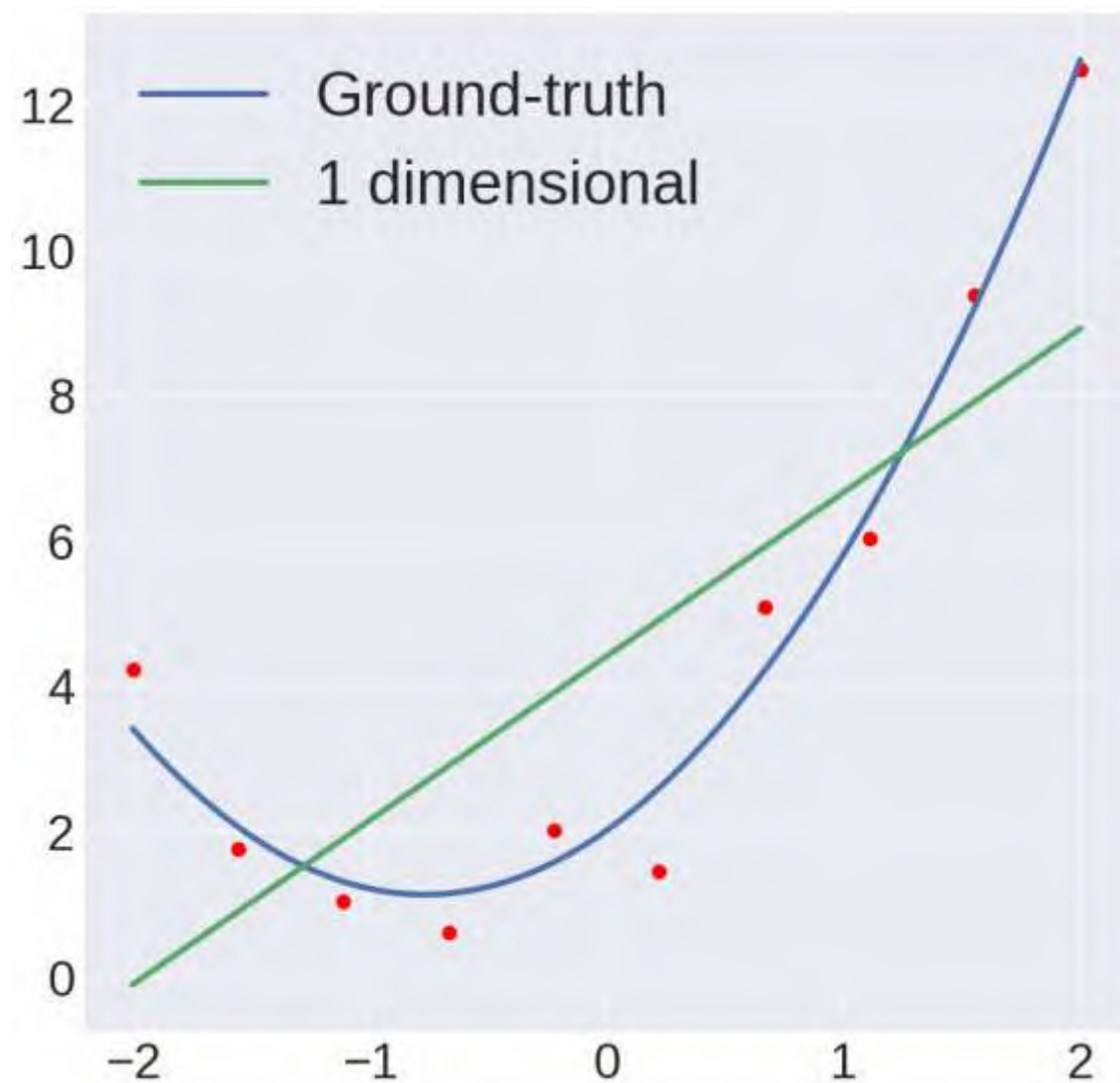


欠拟合

数据: $1.5x^2 + 2.3x + 2 + N(0,0.5)$



欠拟合

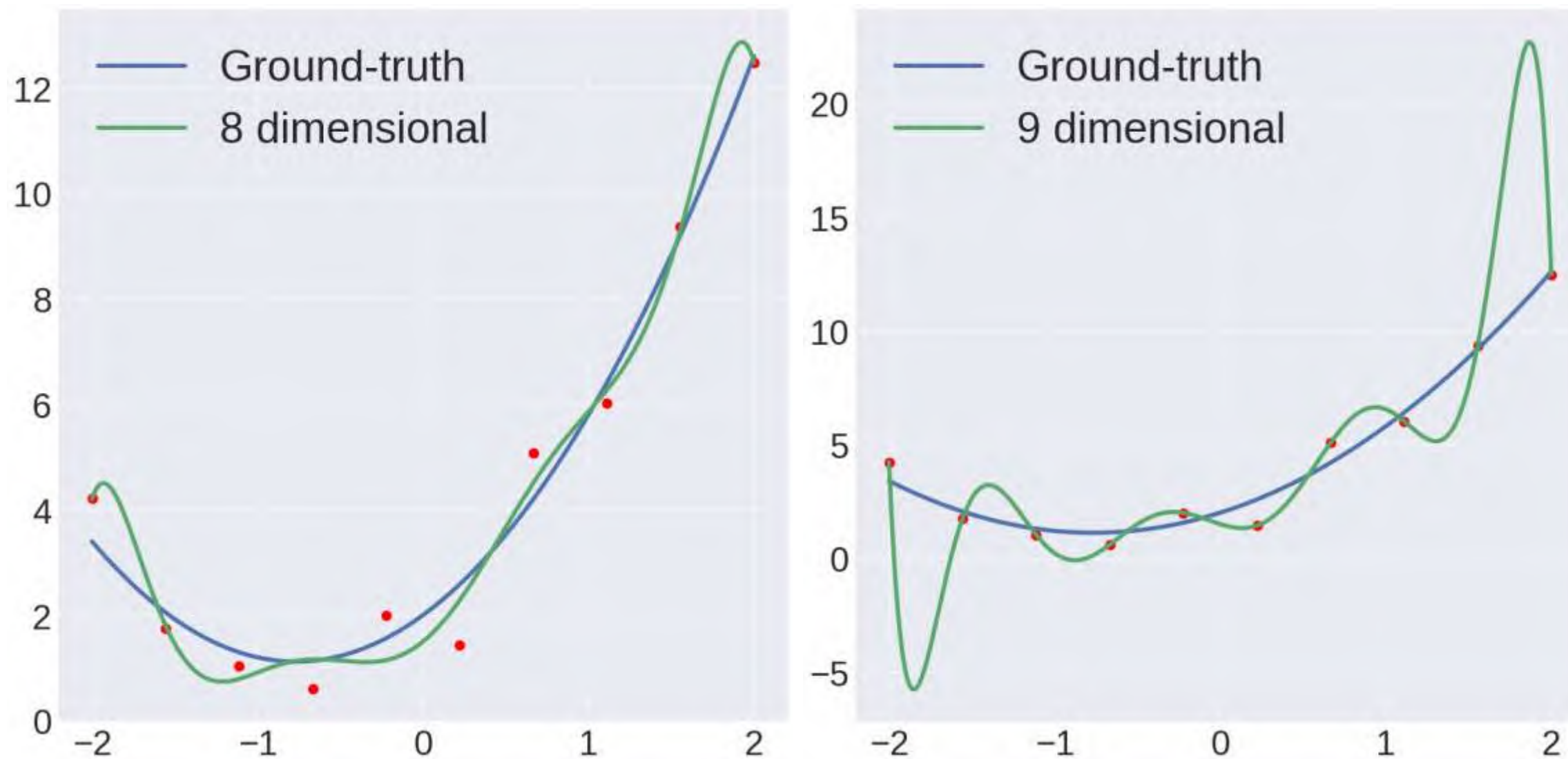


模型没有足够的参数或复杂度去拟合数据。即使在训练数据上，模型的表现也可能不尽如人意。

Bias 偏差 (statistics):
模型固有的错误

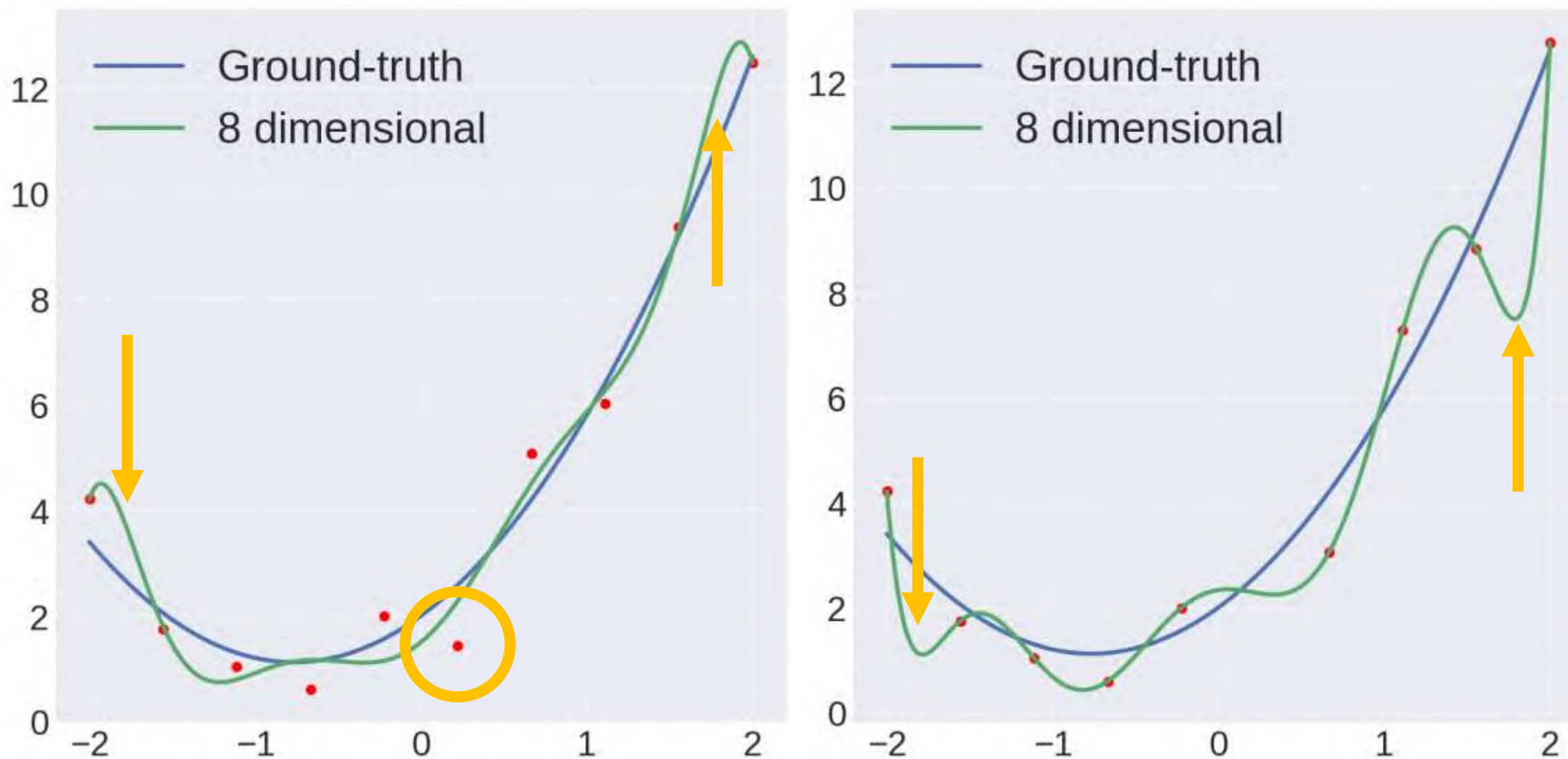
过拟合

数据: $1.5x^2 + 2.3x + 2 + N(0,0.5)$



过拟合

高方差 *variance*:过高的方差意味着模型对训练数据的微小变化非常敏感 移除 **一个数据点**, 模型就会出现很大变化



模型复杂度

$$\arg \min_W \lambda \|W\|_2^2 + \sum_{i=1}^n \underbrace{-\log \left(\frac{\exp((Wx)_{y_i})}{\sum_k \exp((Wx)_k)} \right)}_{\text{交叉熵损失}}$$

Regularization 正则化:
对模型的权重添加惩罚

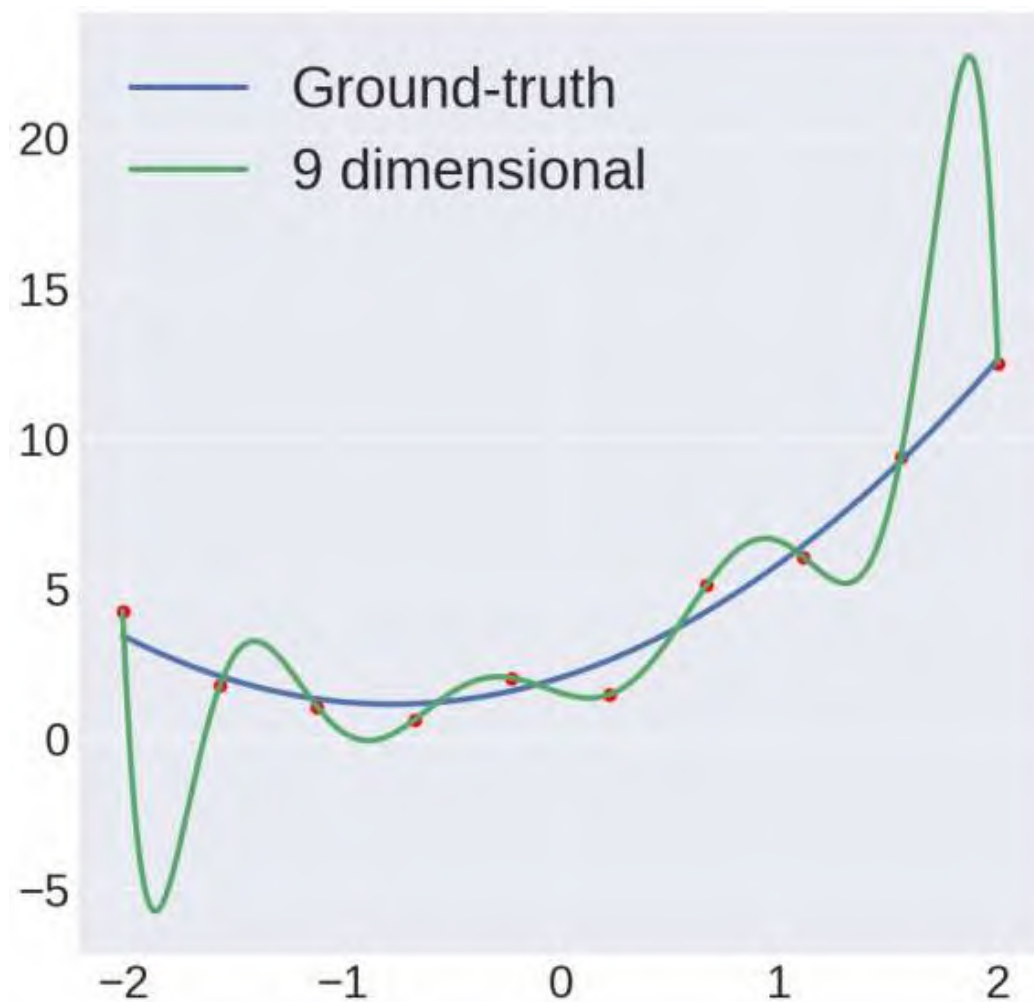
模型复杂性是指模型的容量或能力，它决定了模型可以适应多复杂的数据分布。复杂的模型可能在训练数据上拟合得很好，但在新数据上可能表现得不好，这被称为过拟合。

$$\text{Model 1: } 0.01x_1 + 1.3x_2 + -0.02x_3 + -2.1x_4 + 10$$

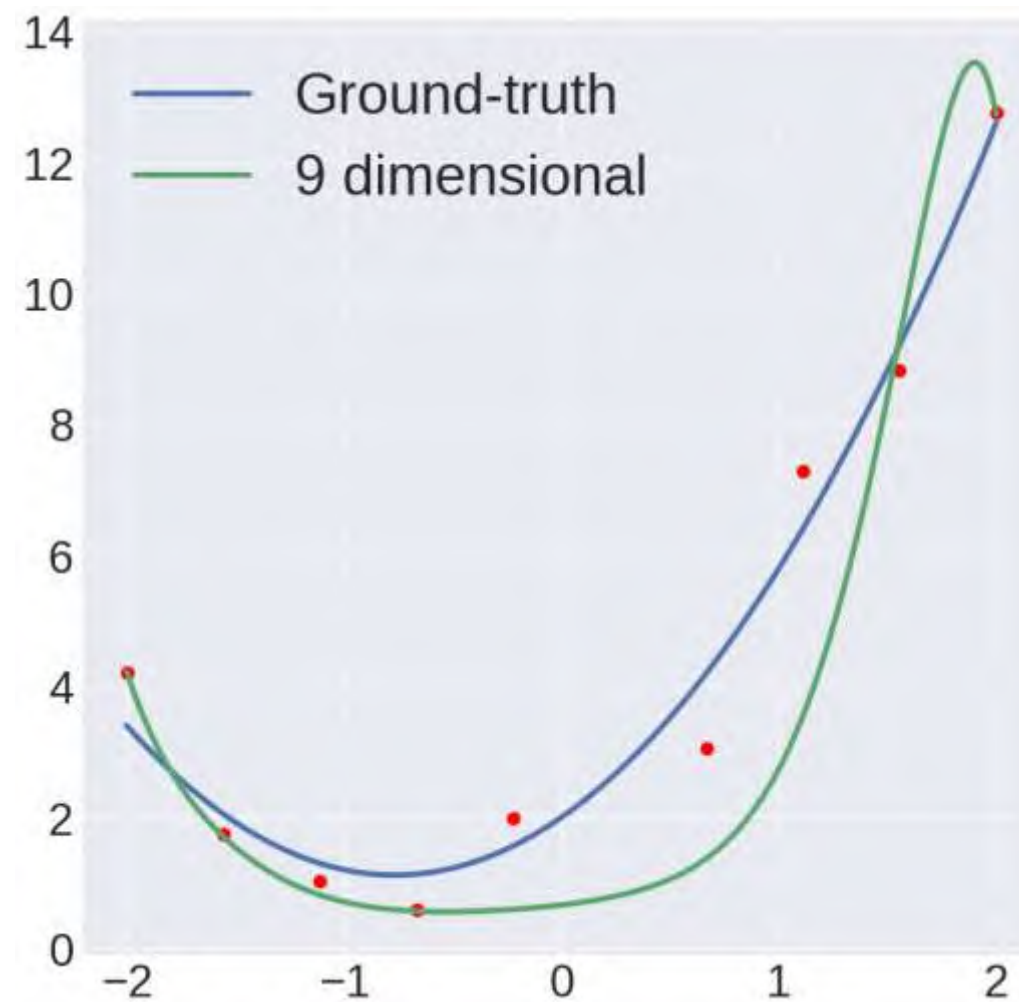
$$\text{Model 2: } 37.2x_1 + 13.4x_2 + 5.6x_3 + -6.1x_4 + 30$$

正则化

没有正则化:
(过) 拟合所有点



使用正则化:
无法拟合所有点



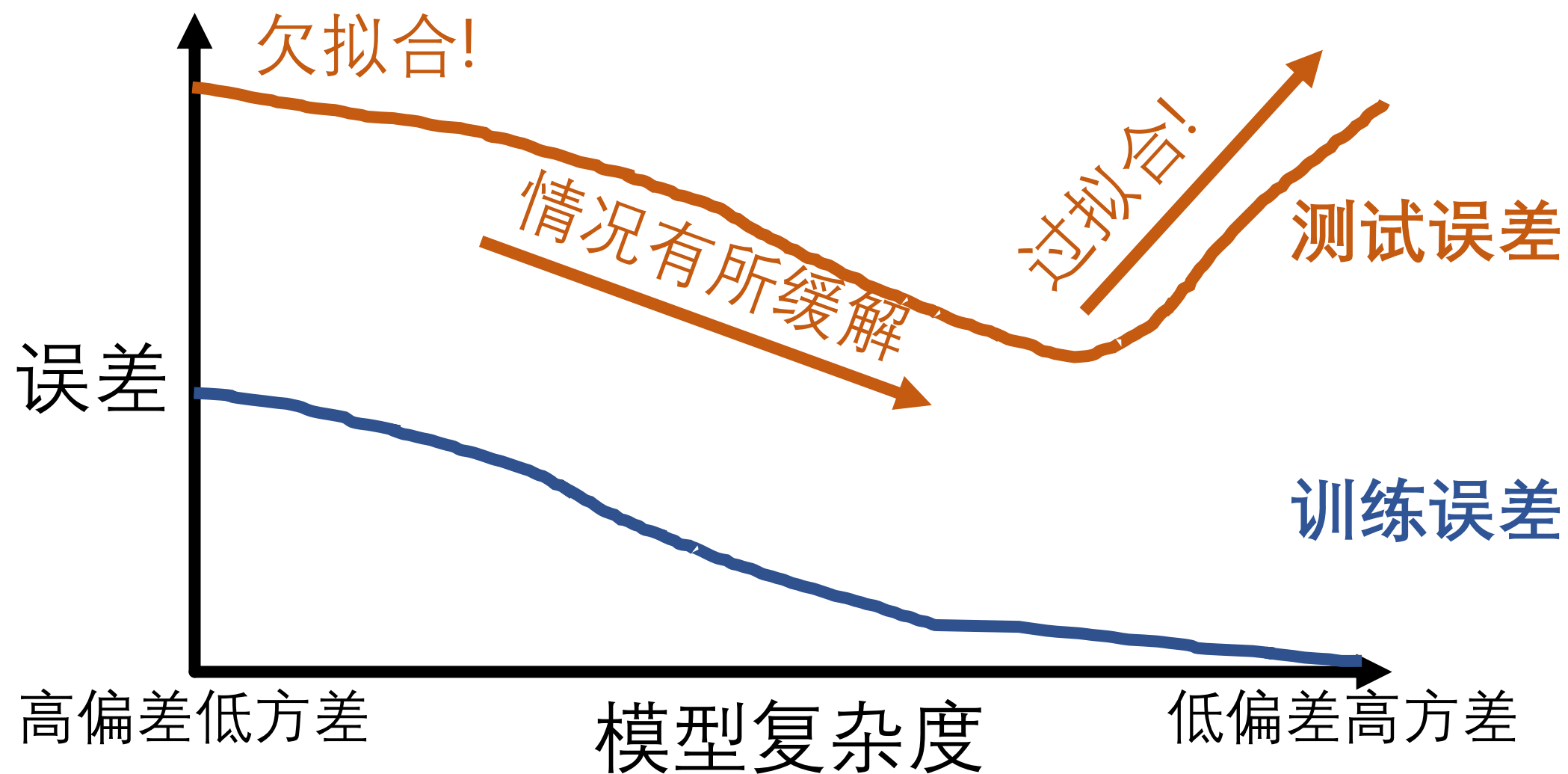
总的来说

当模型在新数据上表现不佳时，通常是由三种误差组合而成的：偏差、方差和固有误差。

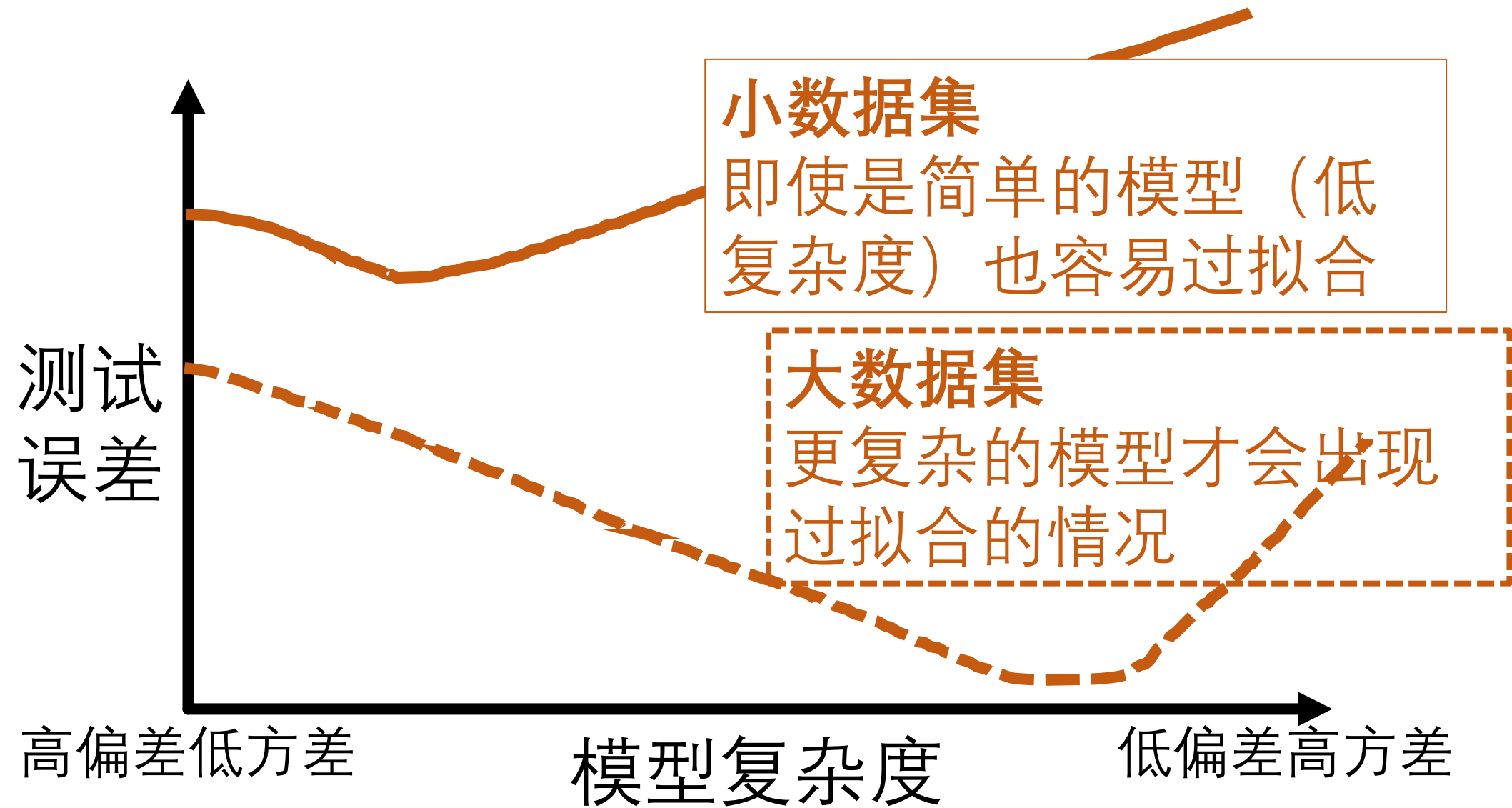
1. **Bias:** 偏差描述的是模型的简化性，它是模型假设与真实数据之间的差异。一个高偏差的模型可能太简单，无法捕捉到数据的真实模式。
2. **Variance:** 方差描述的是模型对训练数据的敏感性。高方差模型在不同的训练数据集上可能会有很大的差异。这通常是因为模型过于复杂，试图捕捉训练数据中的每一个小细节，甚至包括噪声。
3. **Inherent:** 固有误差是与数据本身相关的误差，与模型无关。例如，数据可能包含噪声，或者某些特征无法被测量。

通常，减少偏差可能会增加方差，反之亦然。选择合适的模型复杂度是关键。

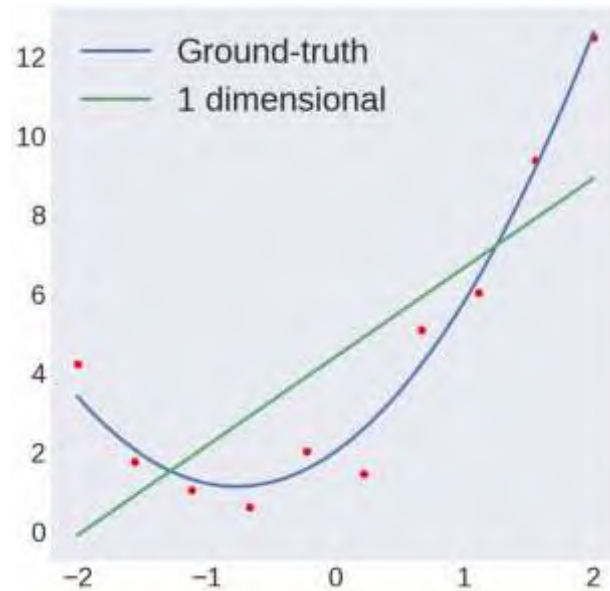
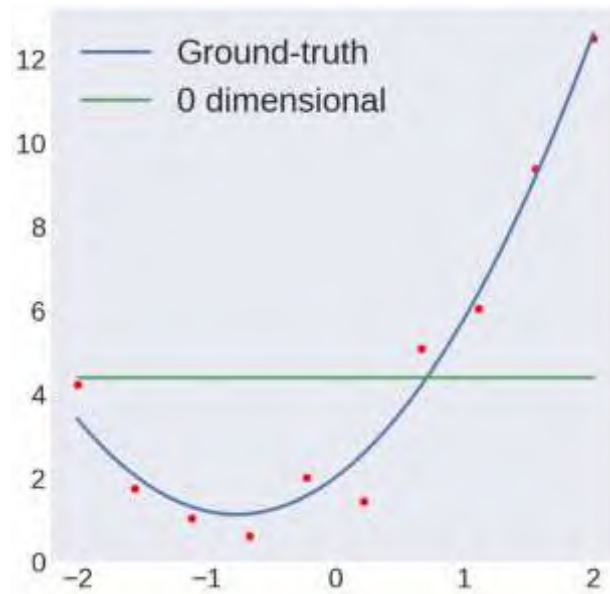
欠拟合与过拟合



欠拟合与过拟合



欠拟合

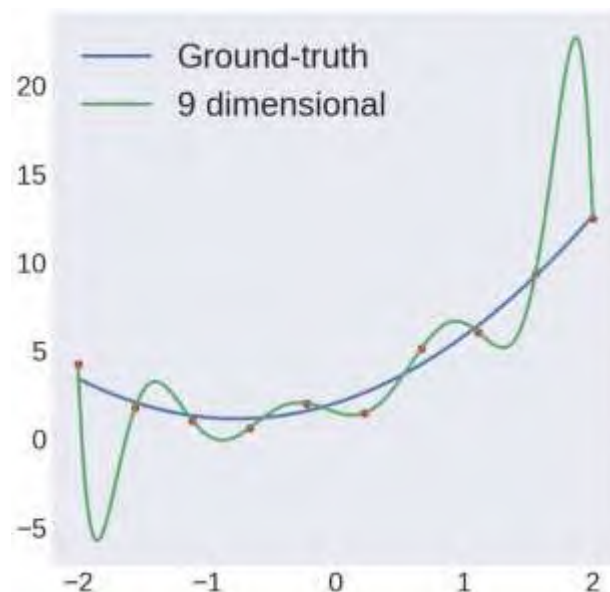
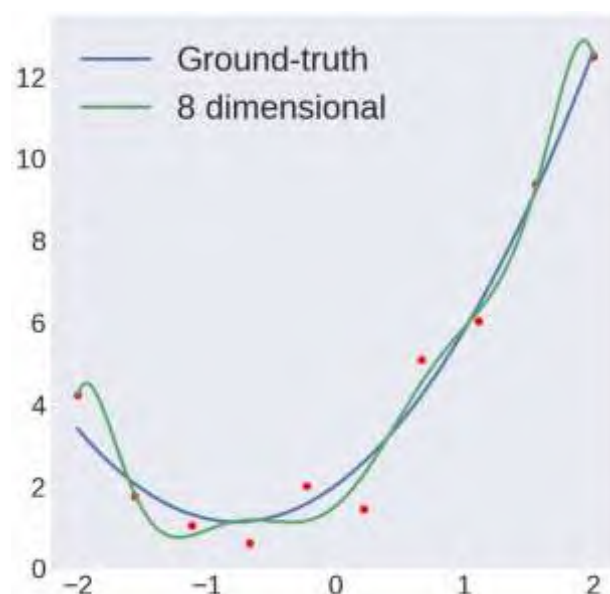


当模型的复杂度过低时，即使在训练数据上也得不到满意的拟合效果。这通常意味着模型太简单，不能捕获数据的真实模式。

解决方案:

- **增加更多特征**: 更多的特征可以增加模型的复杂度，使其更容易捕获到数据中的模式。
- **使用更强大的模型**: 如从线性模型升级到多项式模型或使用深度学习模型。
- **减少正则化**: 正则化是为了避免过拟合而添加的惩罚项，但如果模型欠拟合，则可以减少或移除正则化。

过拟合



模型在训练数据上表现很好，但在验证数据上表现较差，通常是因为模型太复杂，过度拟合了训练数据中的噪声或异常值。

解决方案:

- **增加更多数据**: 更多的训练数据可以帮助模型学习到更广泛的数据分布，从而减少过拟合。
- **使用简单模型**: 选择一个复杂度较低的模型，避免过度拟合。
- **增加正则化**: 正则化可以防止模型过于复杂，从而限制其拟合能力。

建议: 首先确保模型有足够的 ability 进行拟合 (即避免欠拟合), 然后再考虑如何避免过拟合。

深度神经网络 到 深度卷积神经网络

线性模型

设定：三分类模型



模型：每个类别一个权重

$w_0^T \mathbf{x}$ big if cat

$w_1^T \mathbf{x}$ big if dog

$w_2^T \mathbf{x}$ big if hippo

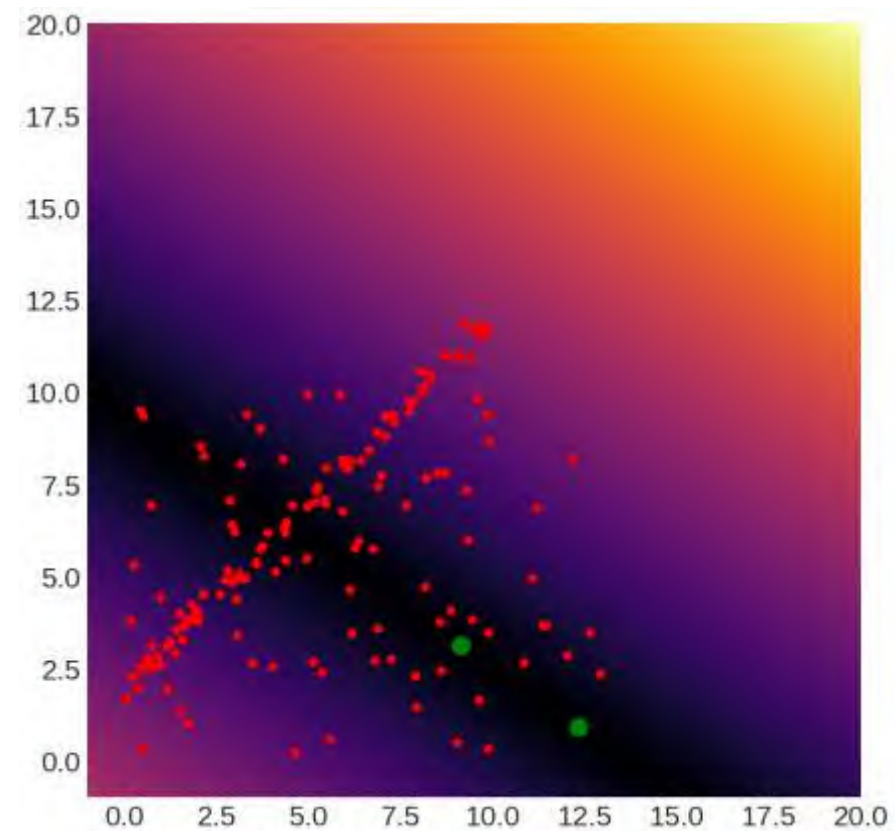
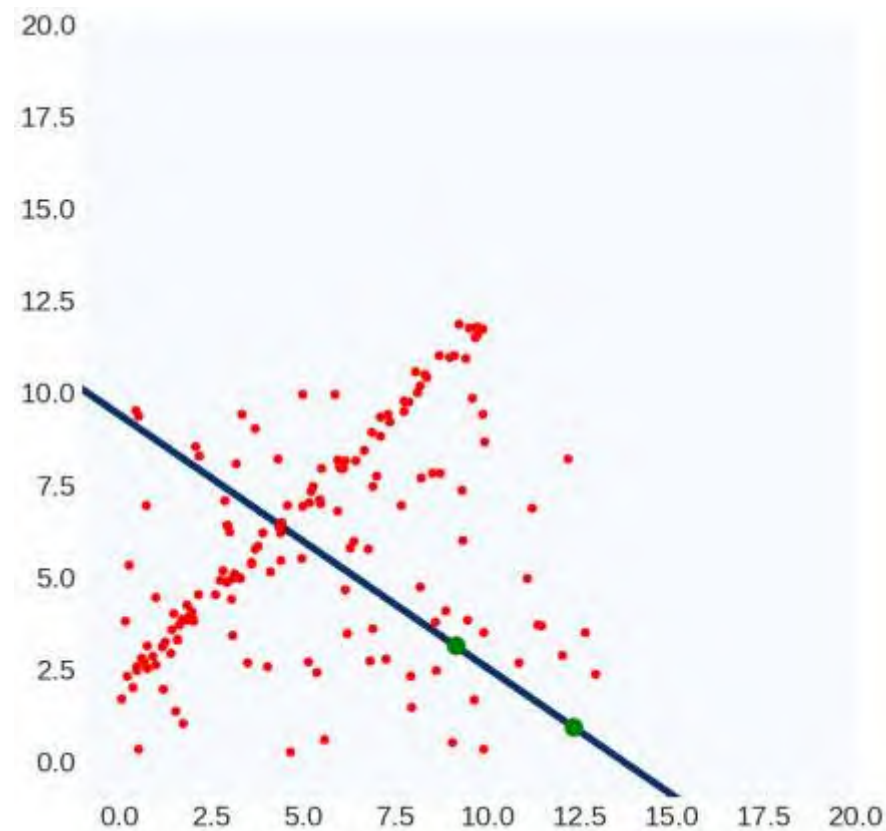
w_0, w_1, w_2

全部参数：

$W_{3 \times F}$ where \mathbf{x} is in \mathbb{R}^F

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 $\mathbf{w}^T \mathbf{x} + b$**



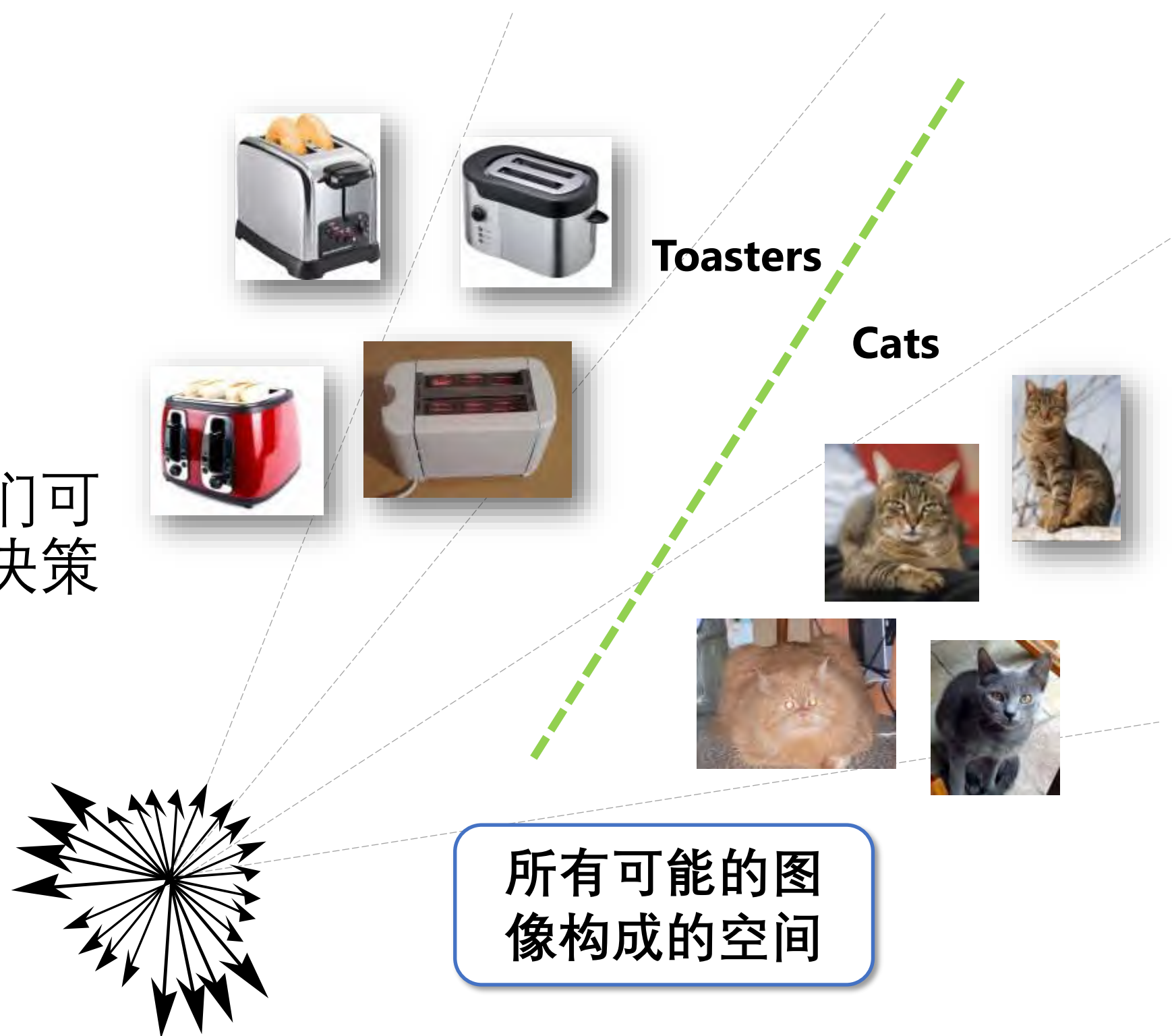
将图像视为高维向量

- **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。

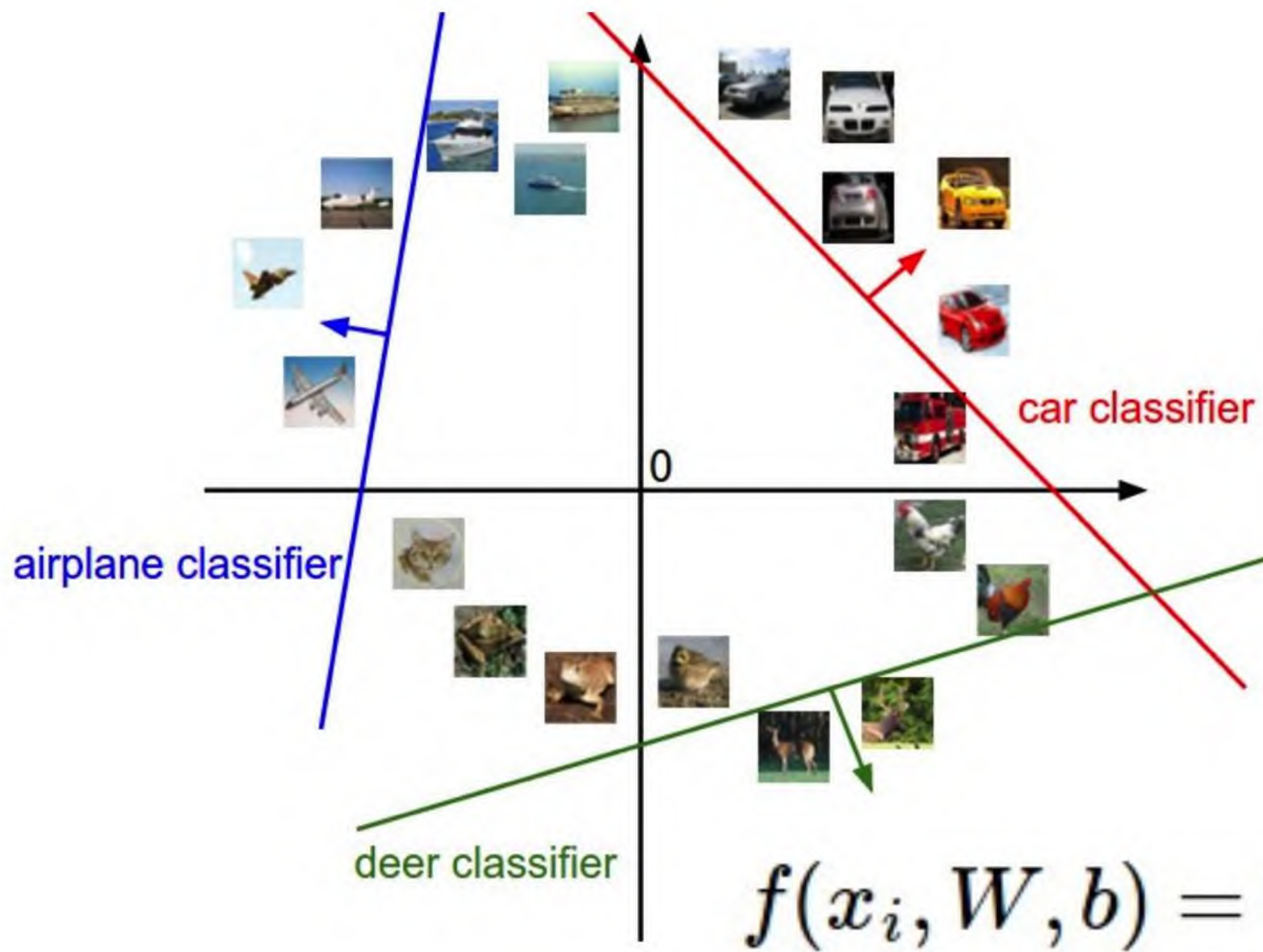


将图像视为高维向量

- 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。



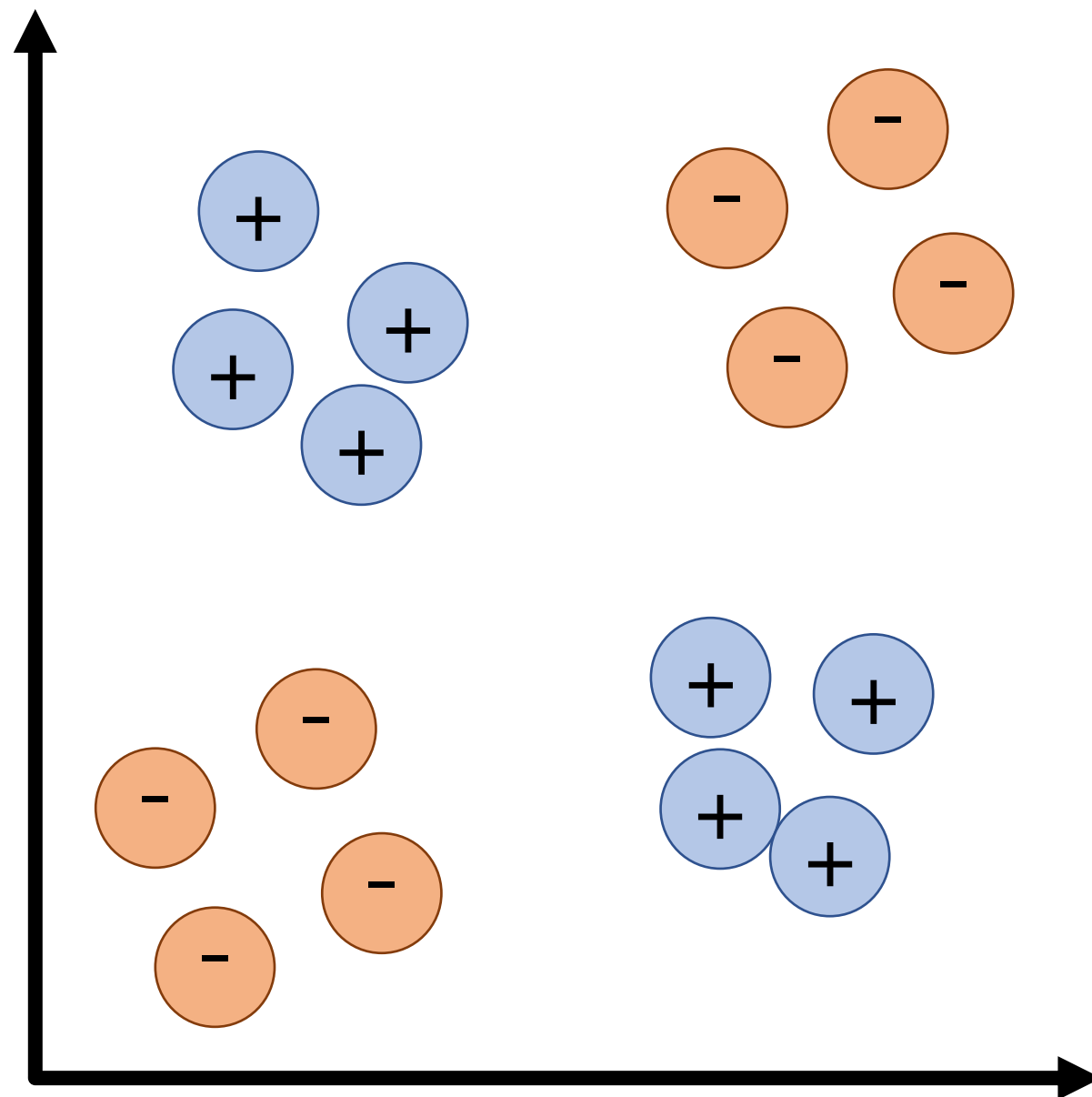
期望：图像线性可分



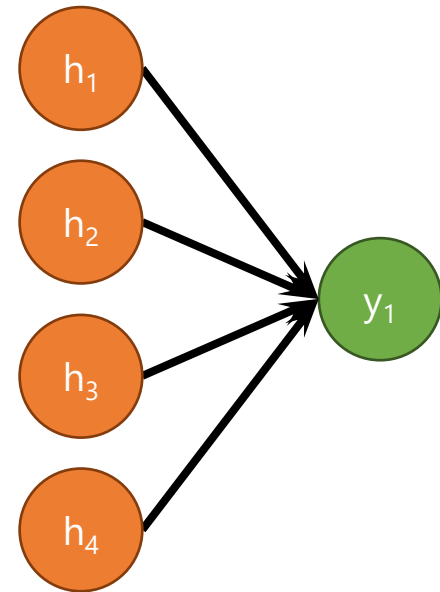
$$f(x_i, W, b) = Wx_i + b$$

现实

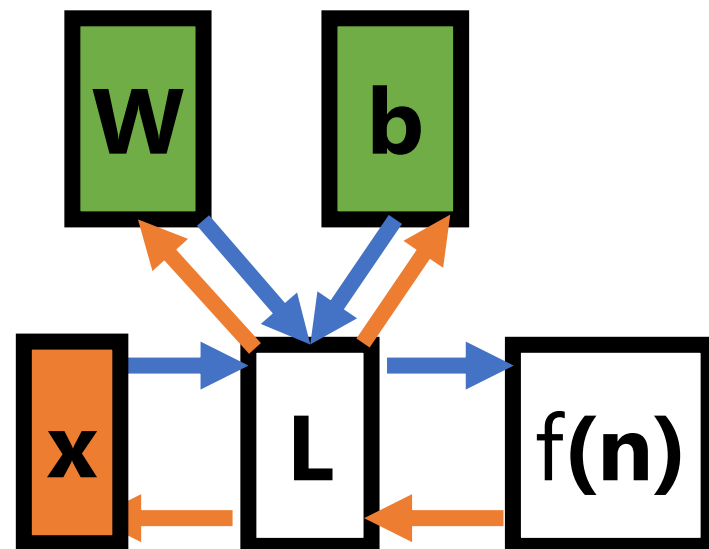
- 数据并非线性可分



线性模型能表征什么？



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



线性模型怎么处理非线性问题？

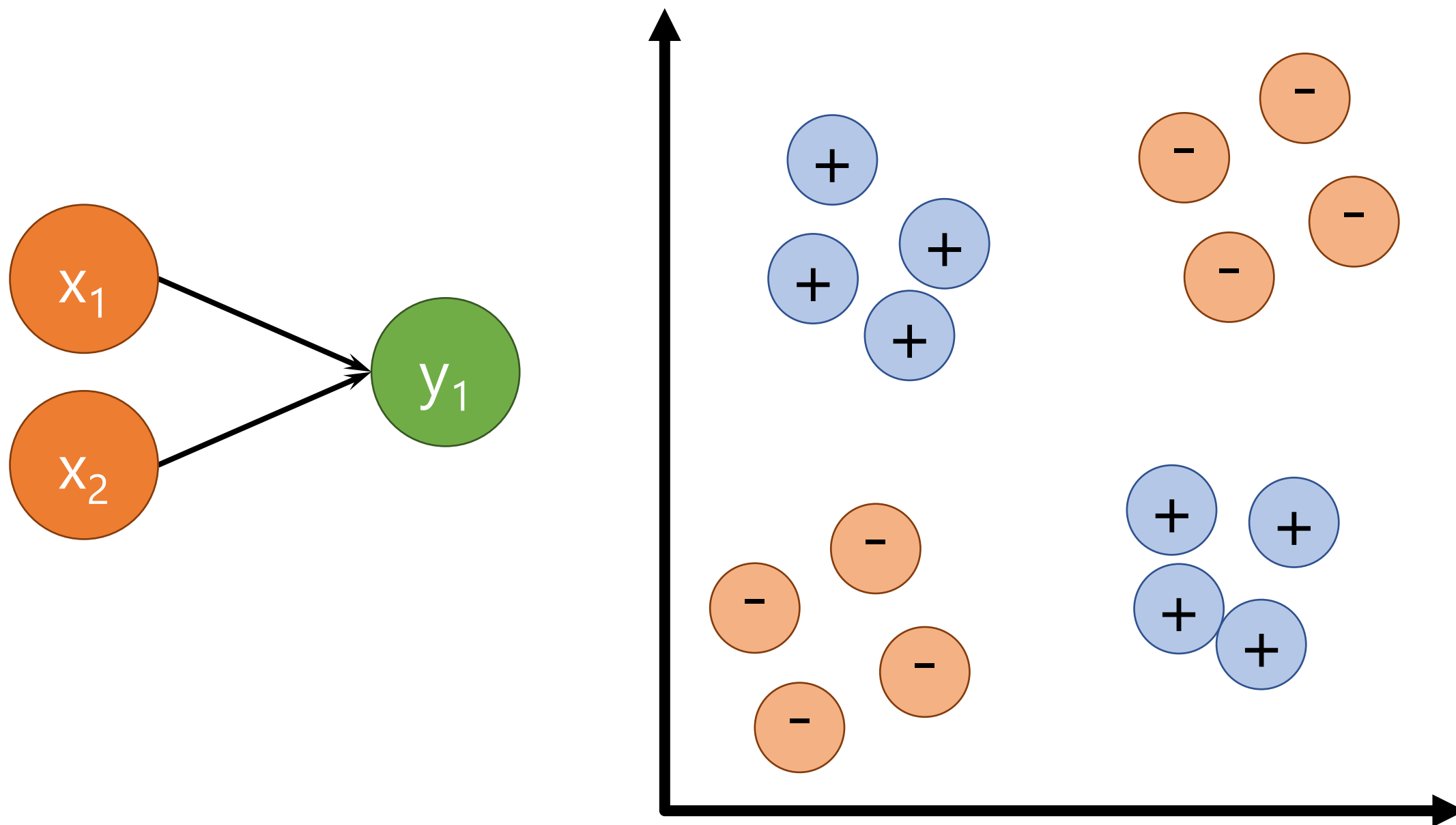
多项式回归: 给定 x , 预测 y

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

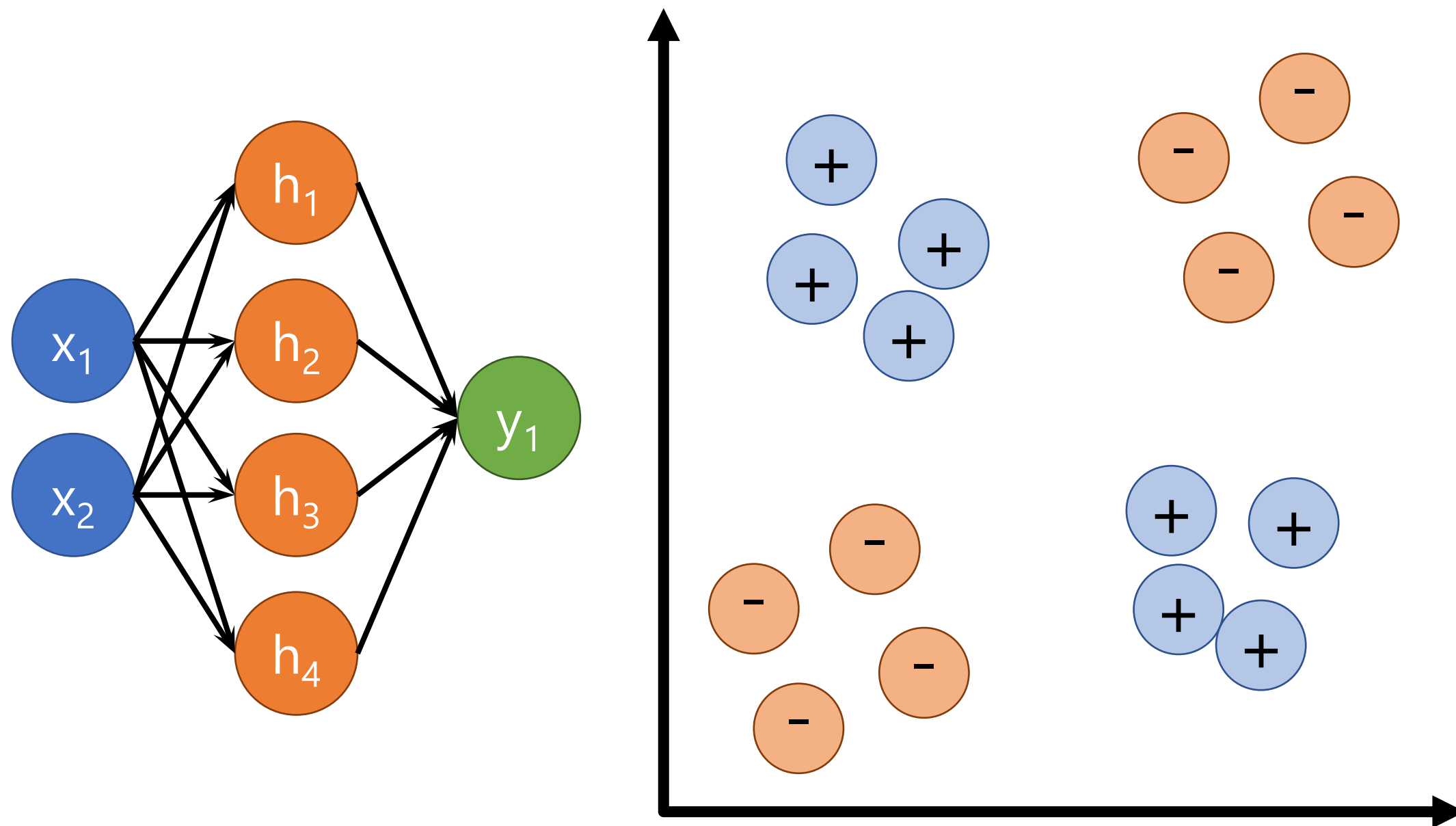
我们有一个输入矩阵 X , 其中包含了各种多项式次数的数据 (例如, x, x^2, x^3 等)。这样, 模型可以选择最佳的多项式次数来拟合数据。

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。

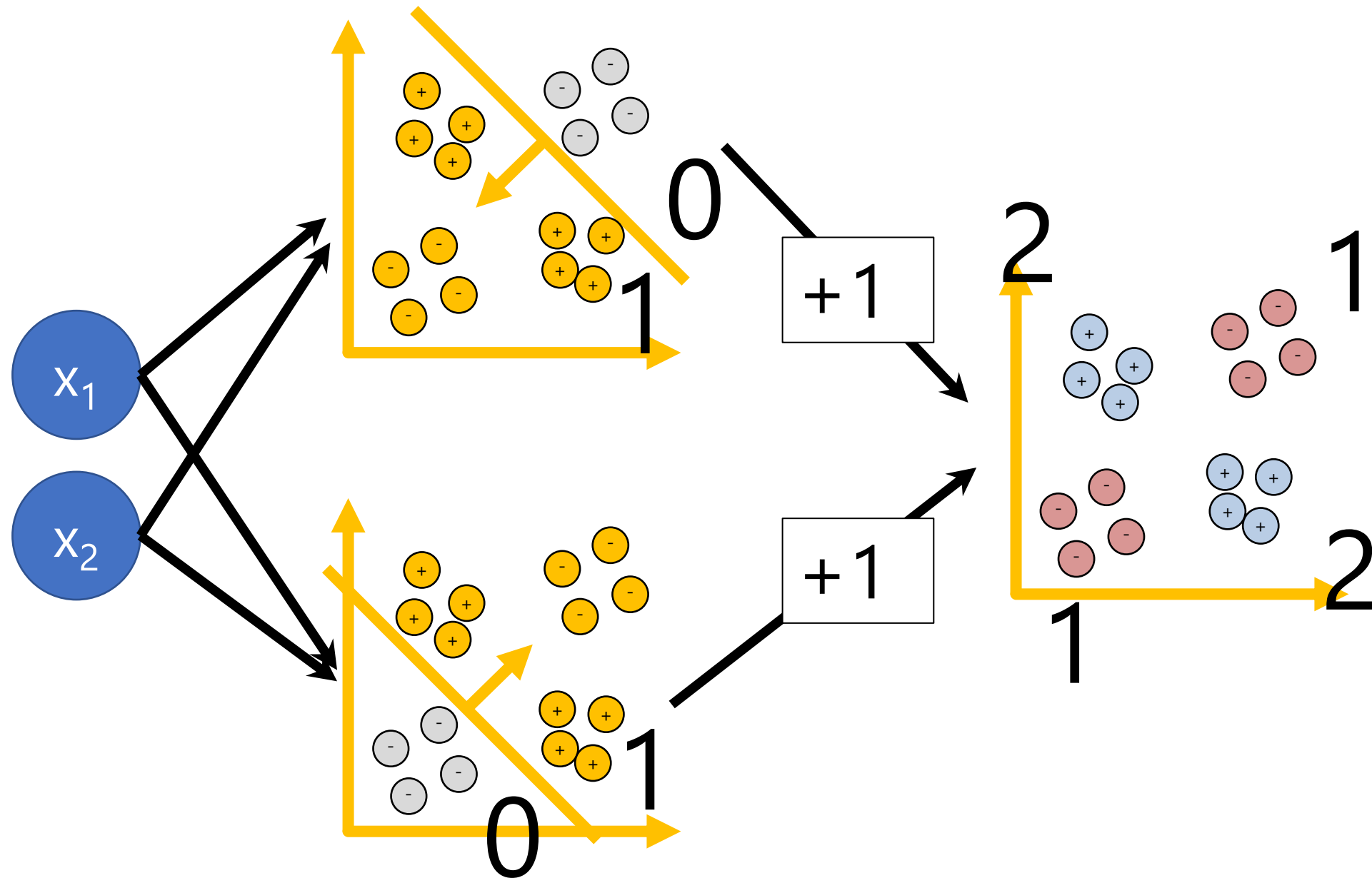
怎么训练一个线性模型来解决这个问题？



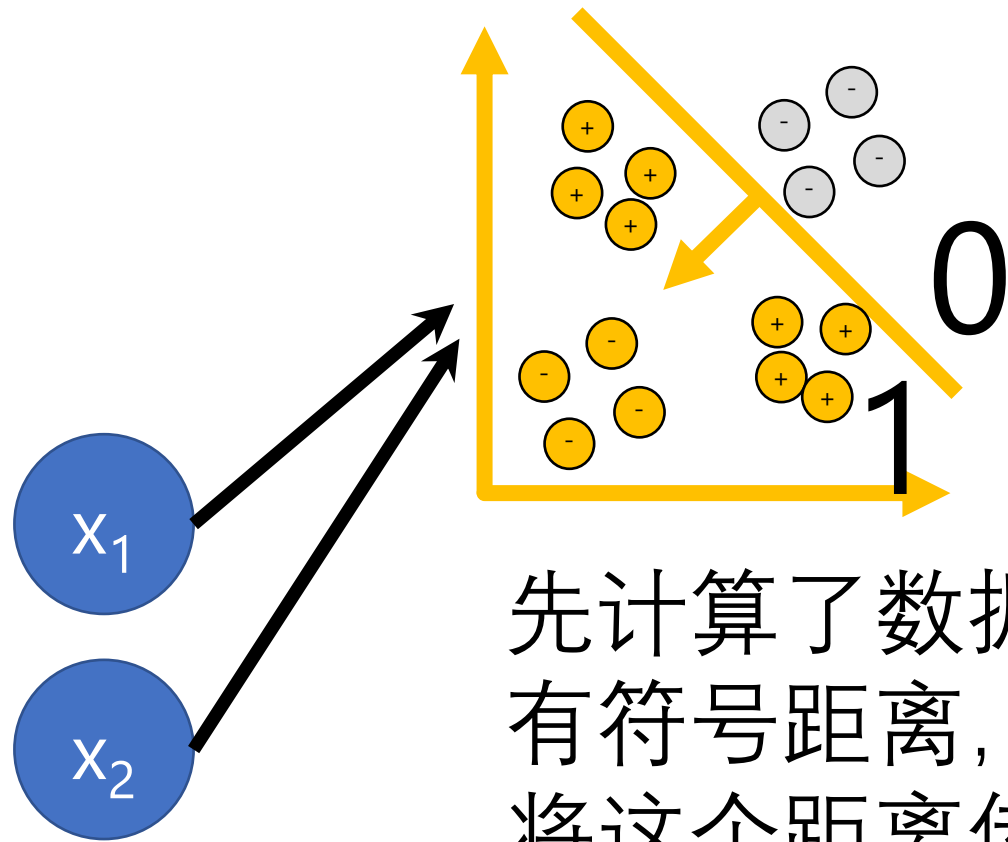
让我们把模型变得复杂一点



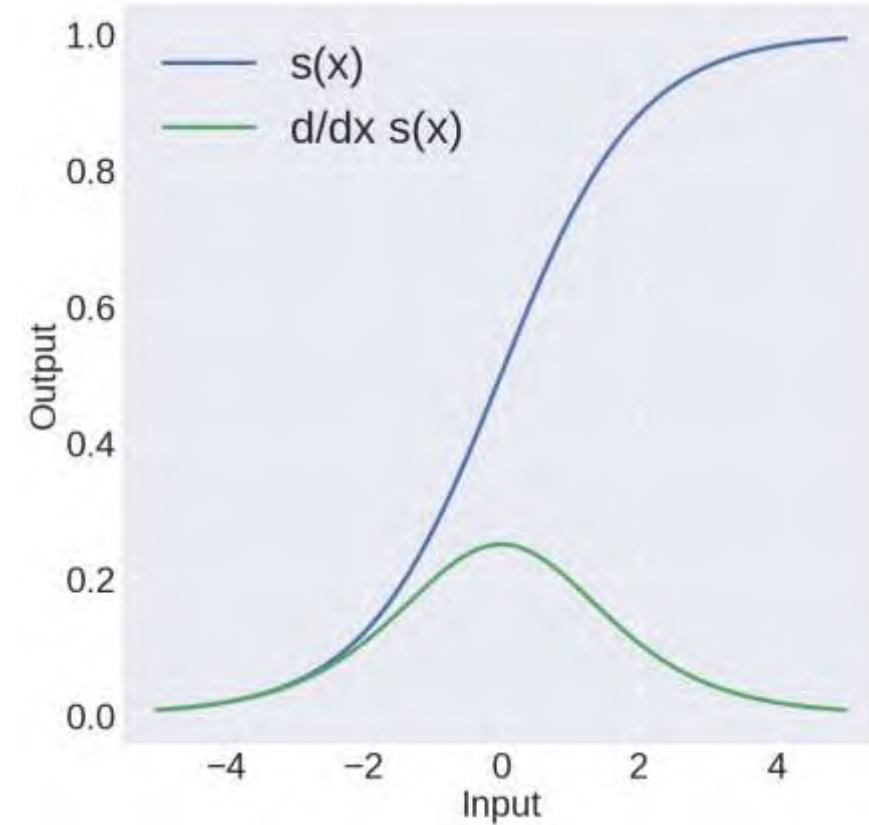
方案1



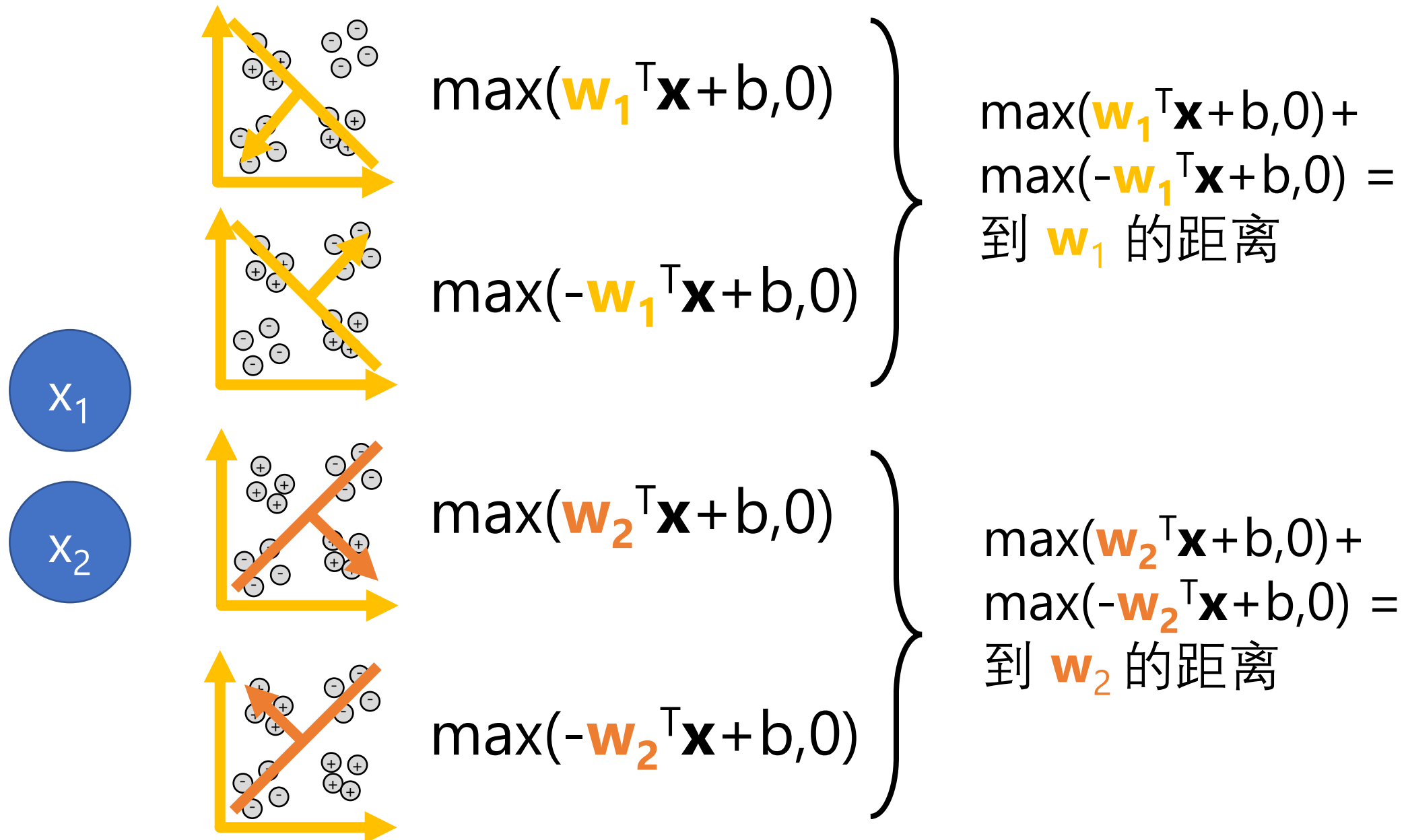
方案1



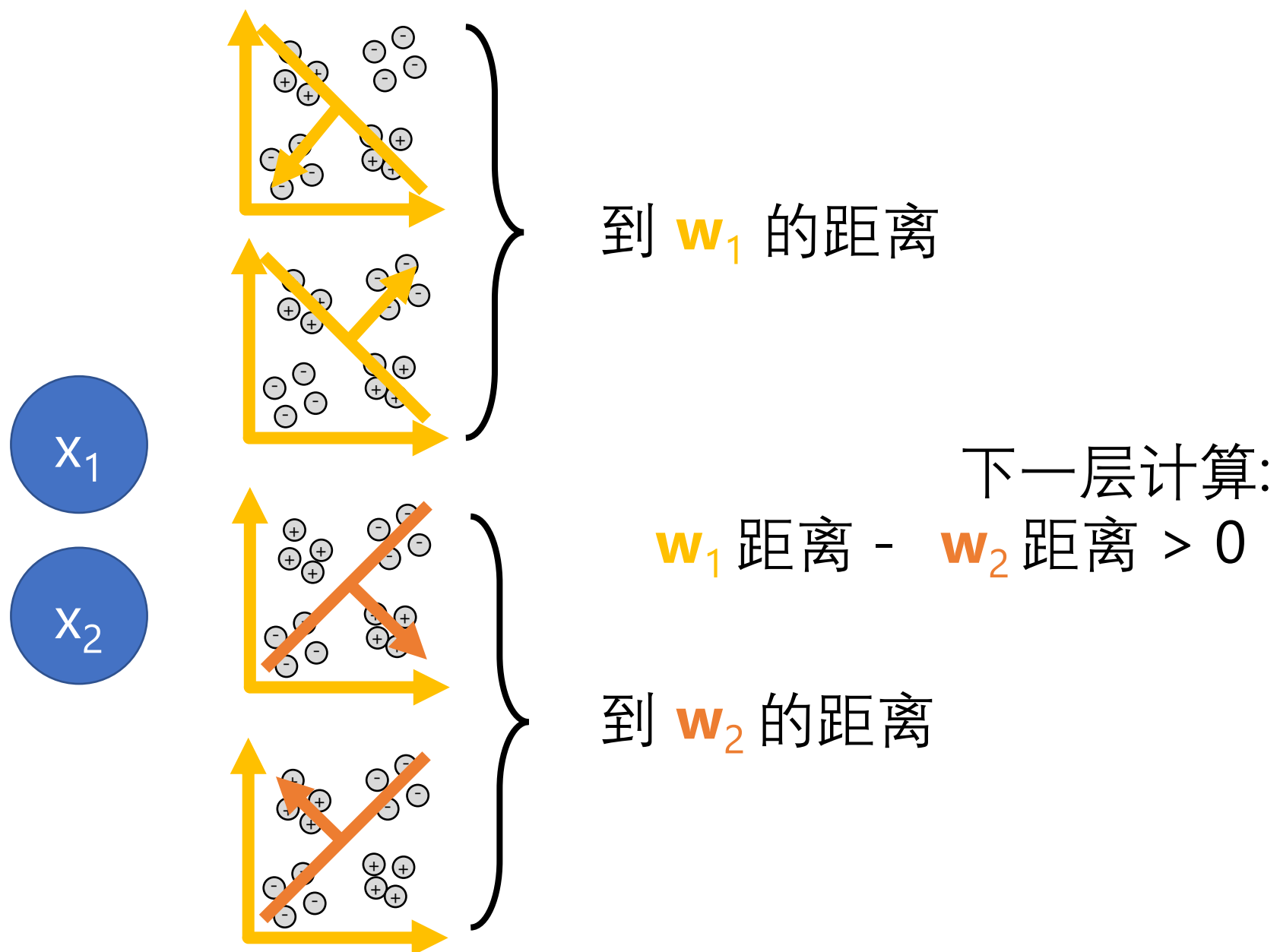
先计算了数据点的有符号距离，然后将这个距离传递给sigmoid激活函数



方案2

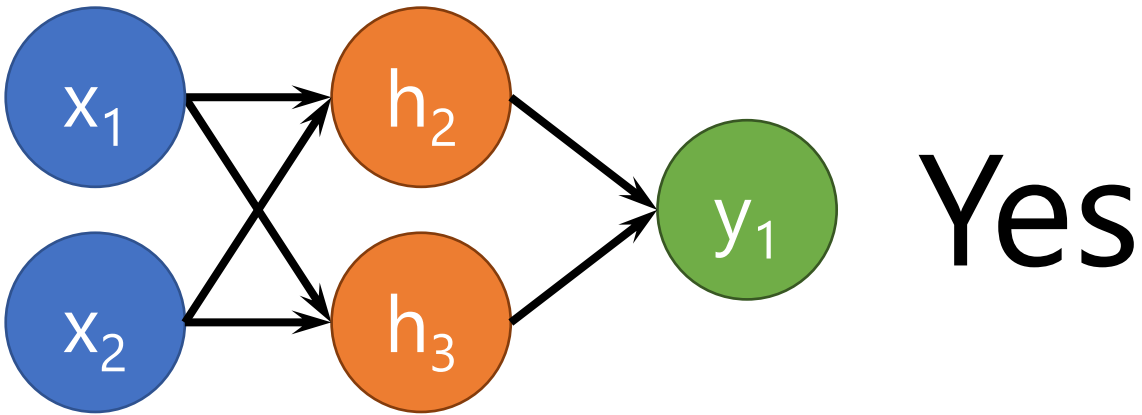
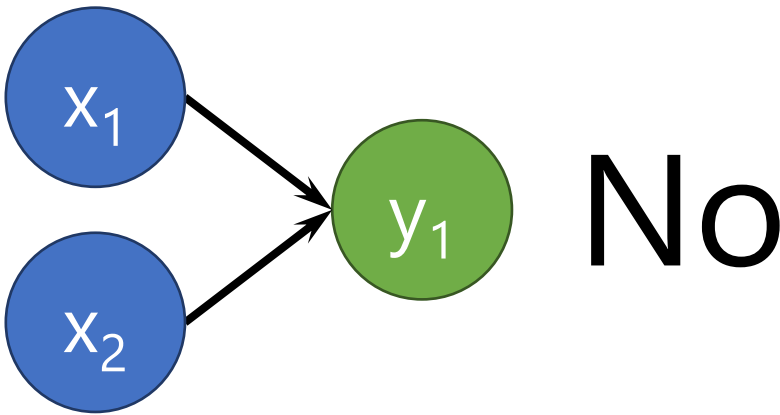
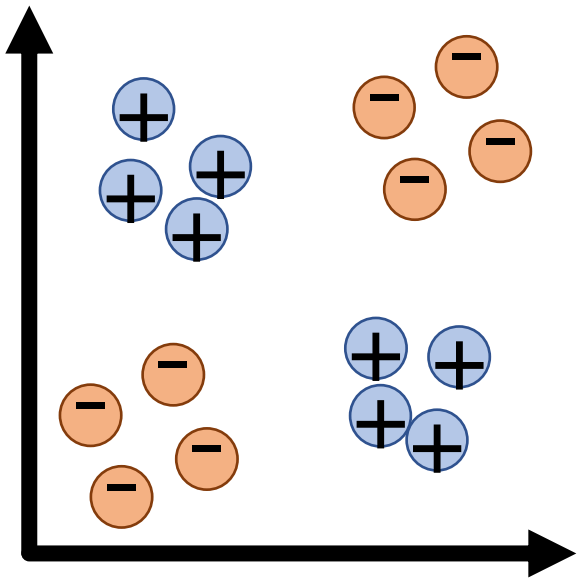


方案2



深度的重要性

可以实现xor吗?



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

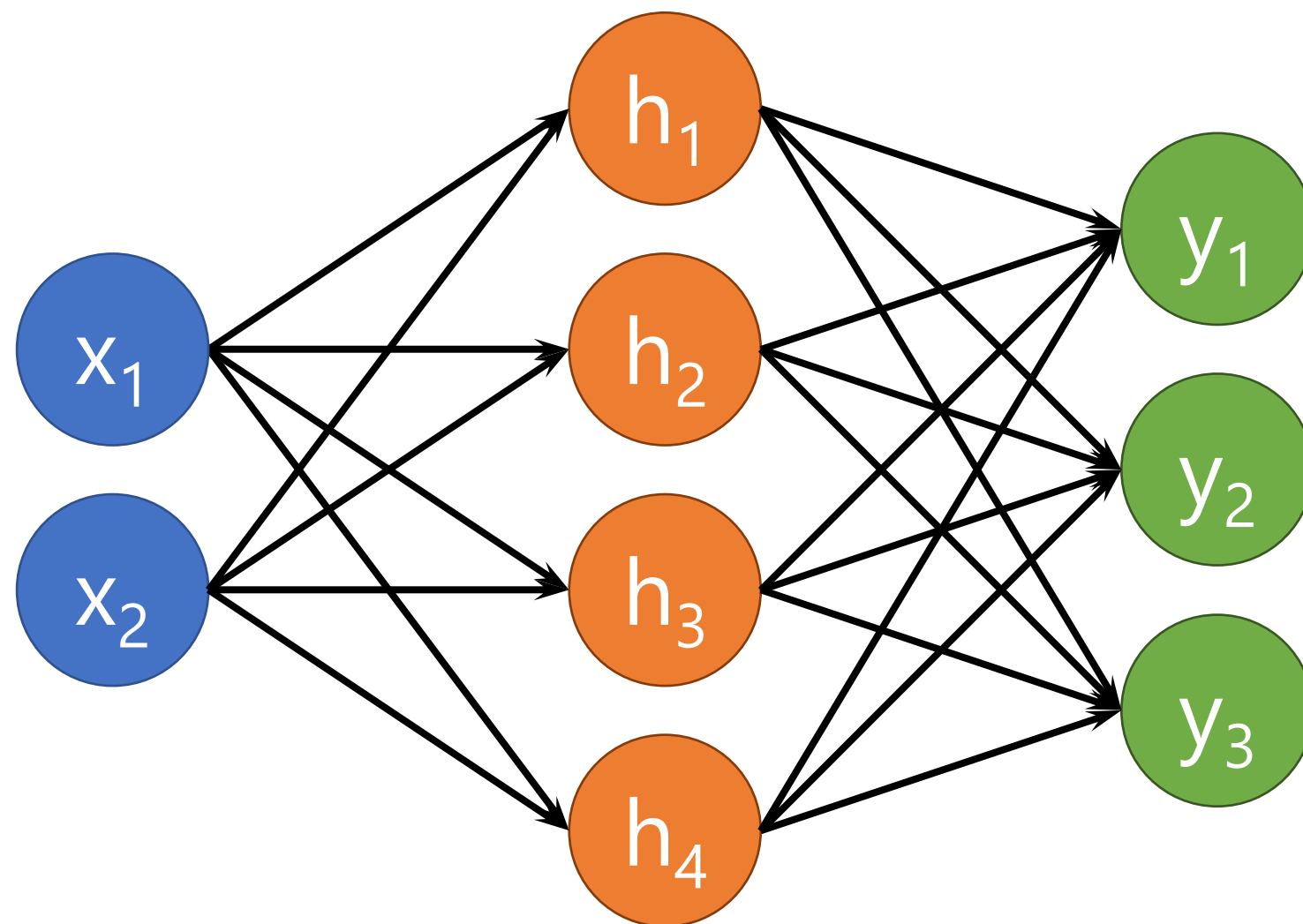
- 全面了解、掌握数据: 亲自操作和实验数据是获得直觉的最好方法。
- 持怀疑态度: 对于你所做的每件事, 甚至你所听到的每件事, 都要持怀疑态度。
- 神经网络和其他机器学习技术的背后是数学和逻辑, 不是“炼金”与“魔法”。
- 如何手动设置深度网络的权重。这个思考技巧可以帮助理解权重如何影响网络的行为和输出, 尽管在实践中我们通常不会手动设置这些权重。

神经网络

Input

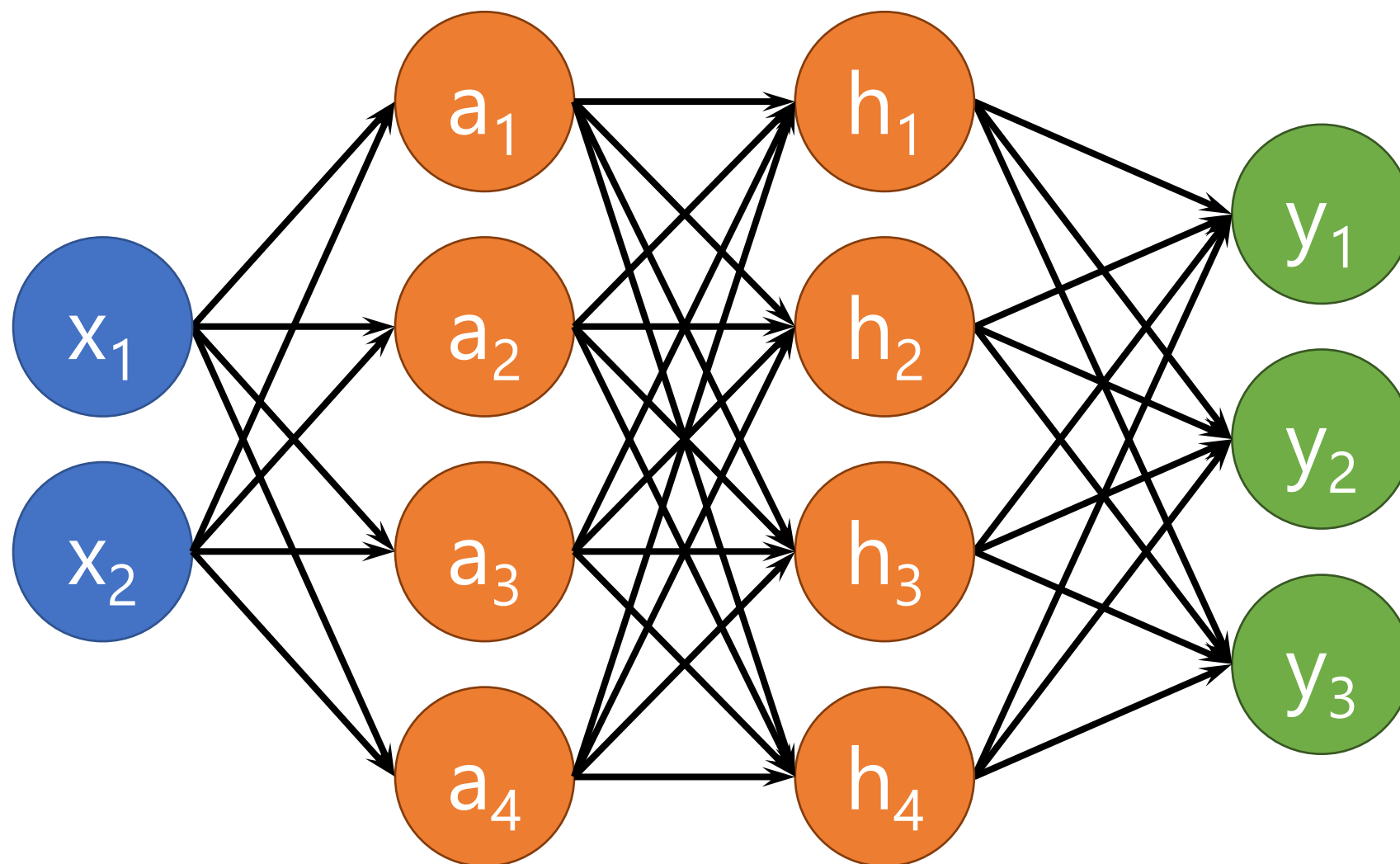
Hidden

Output

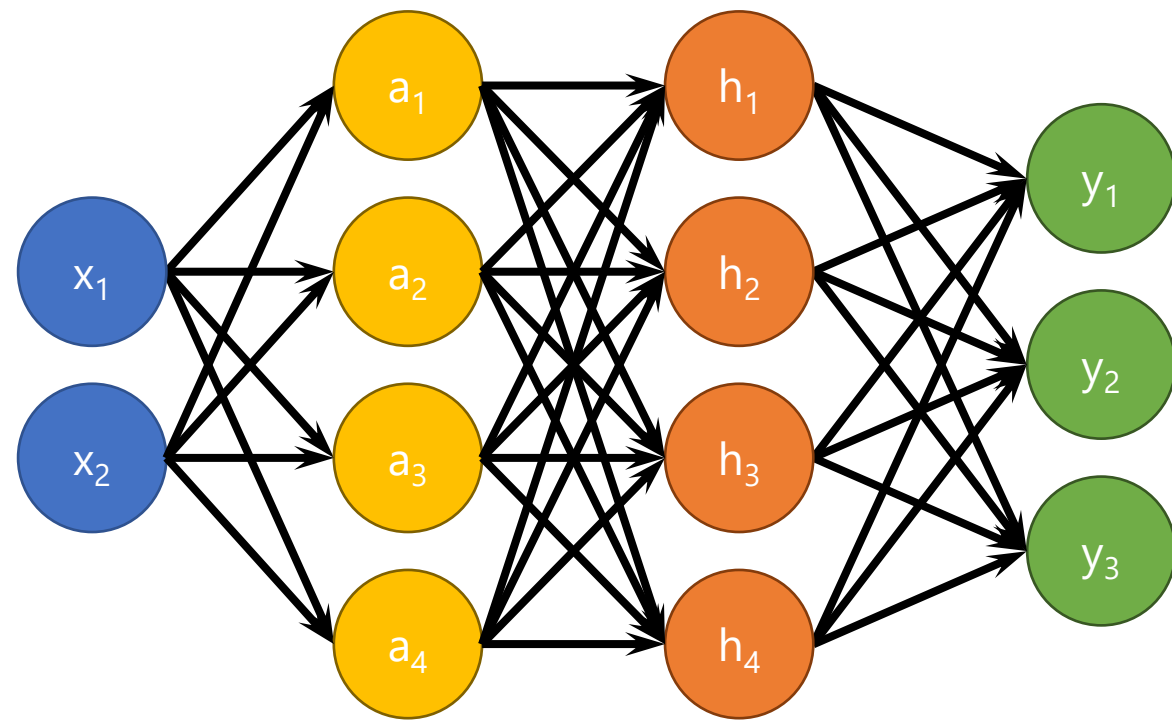


更深的神经网络

Input Hidden 1 Hidden 2 Output

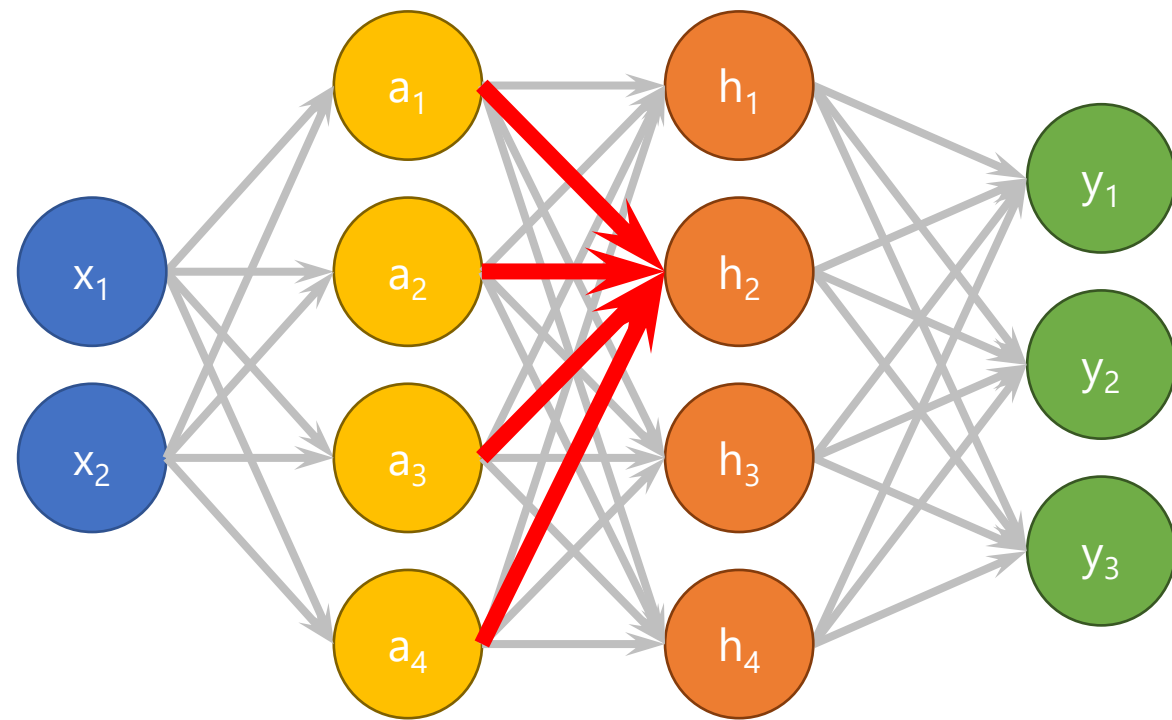


全连接网络



每个神经元都跟之前一层的神经元全部相连

全连接网络



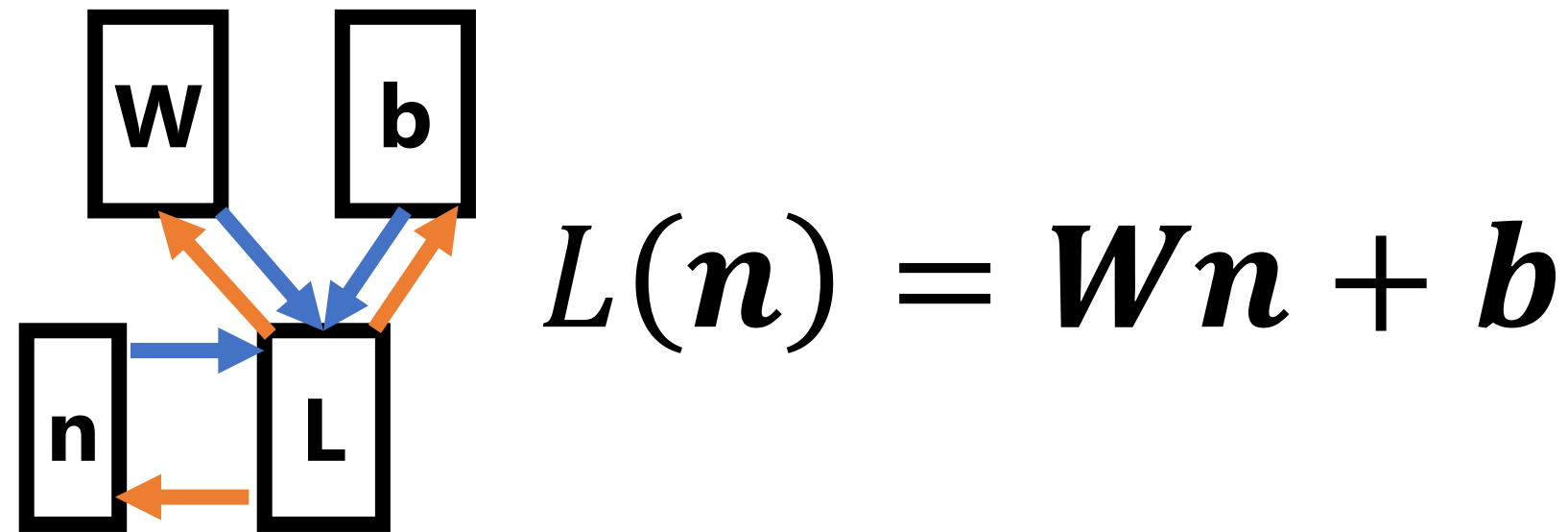
a 上一层 a 的值
 w_i, b_i 神经元 i 权重 weight, 偏置 bias
 f 激活函数

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

The equation shows the calculation of the hidden layer output h . The input vector a (yellow box) is multiplied by the weight matrix W (gray box with w_i^T rows). The result is added to the bias vector b (gray box). The final result is passed through the activation function f to produce the hidden layer output h (orange box).

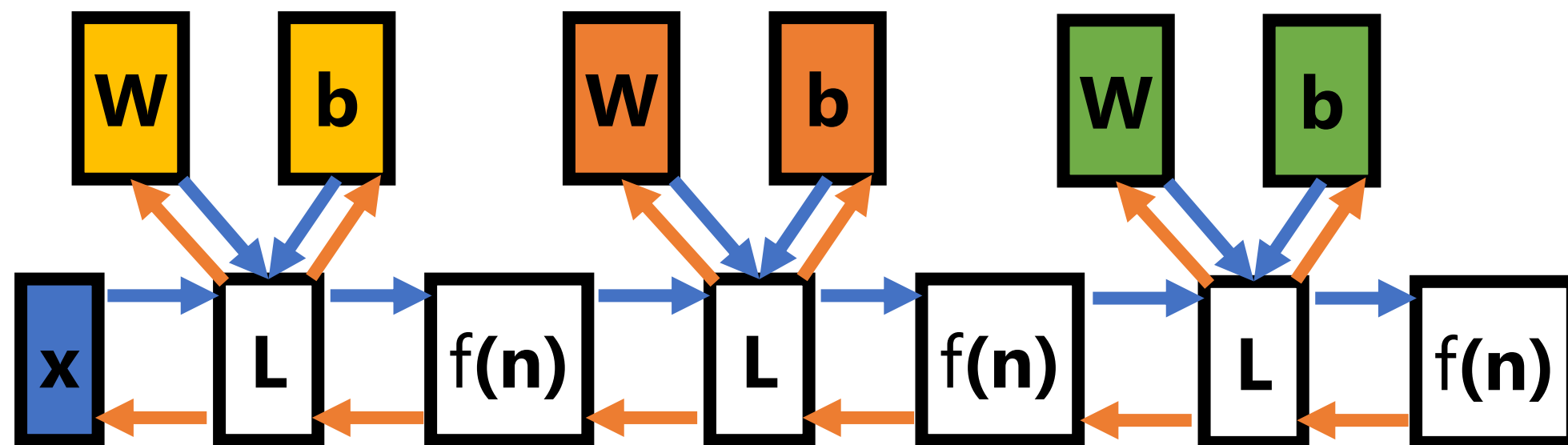
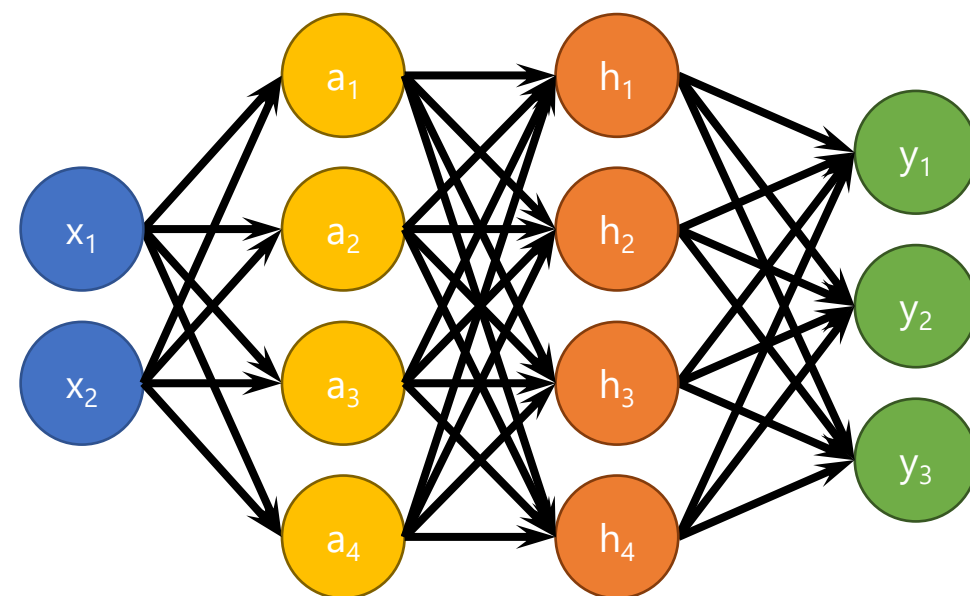
全连接网络

我们也称其为：“Linear Layer 线性层”



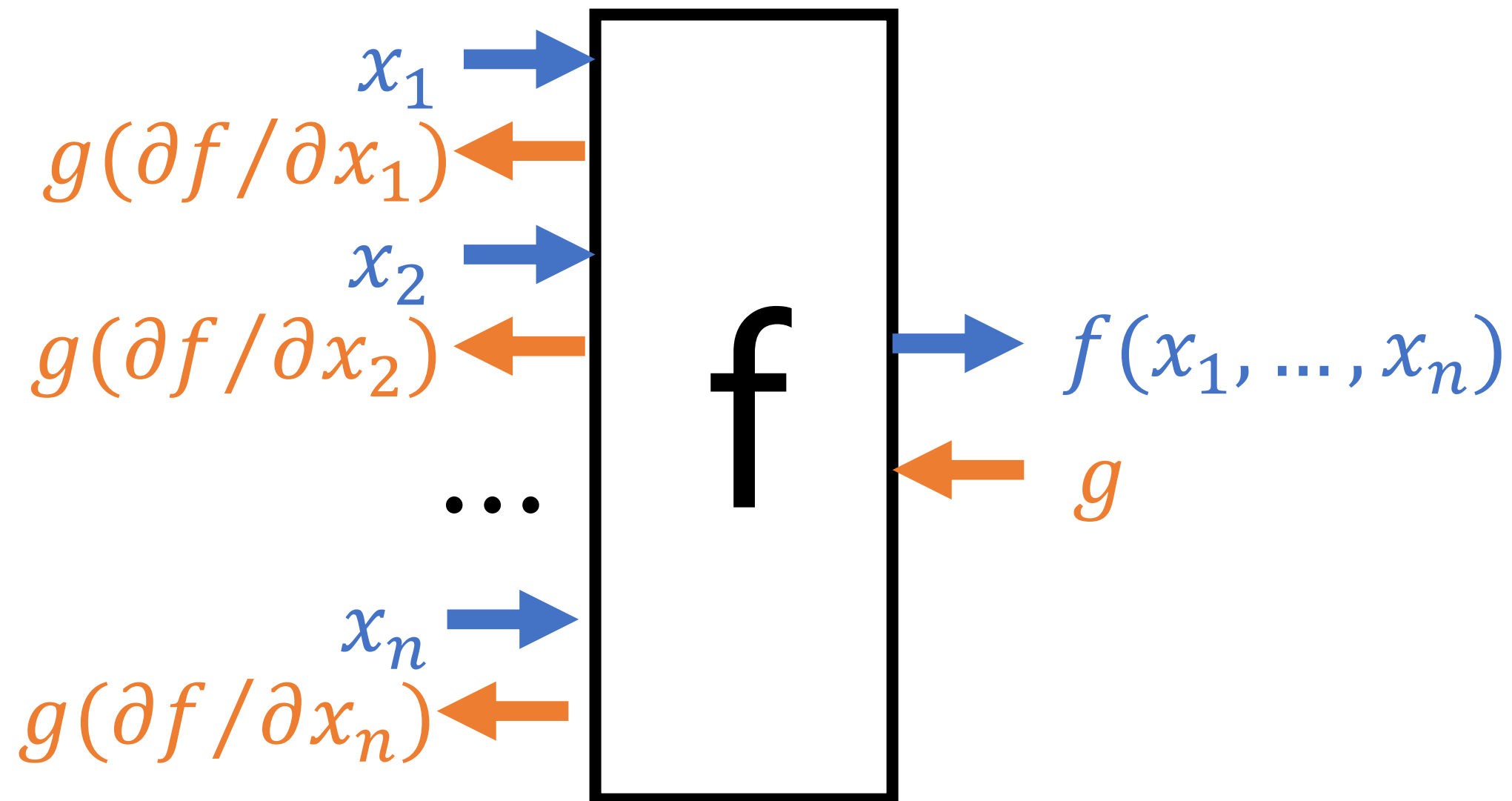
可以根据输入计算梯度优化

全连接网络



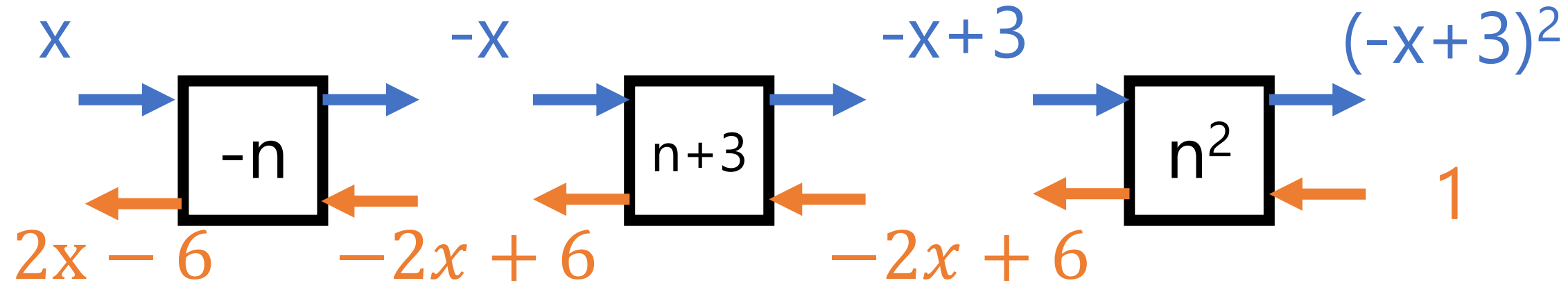
Backpropagation 反向传播

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

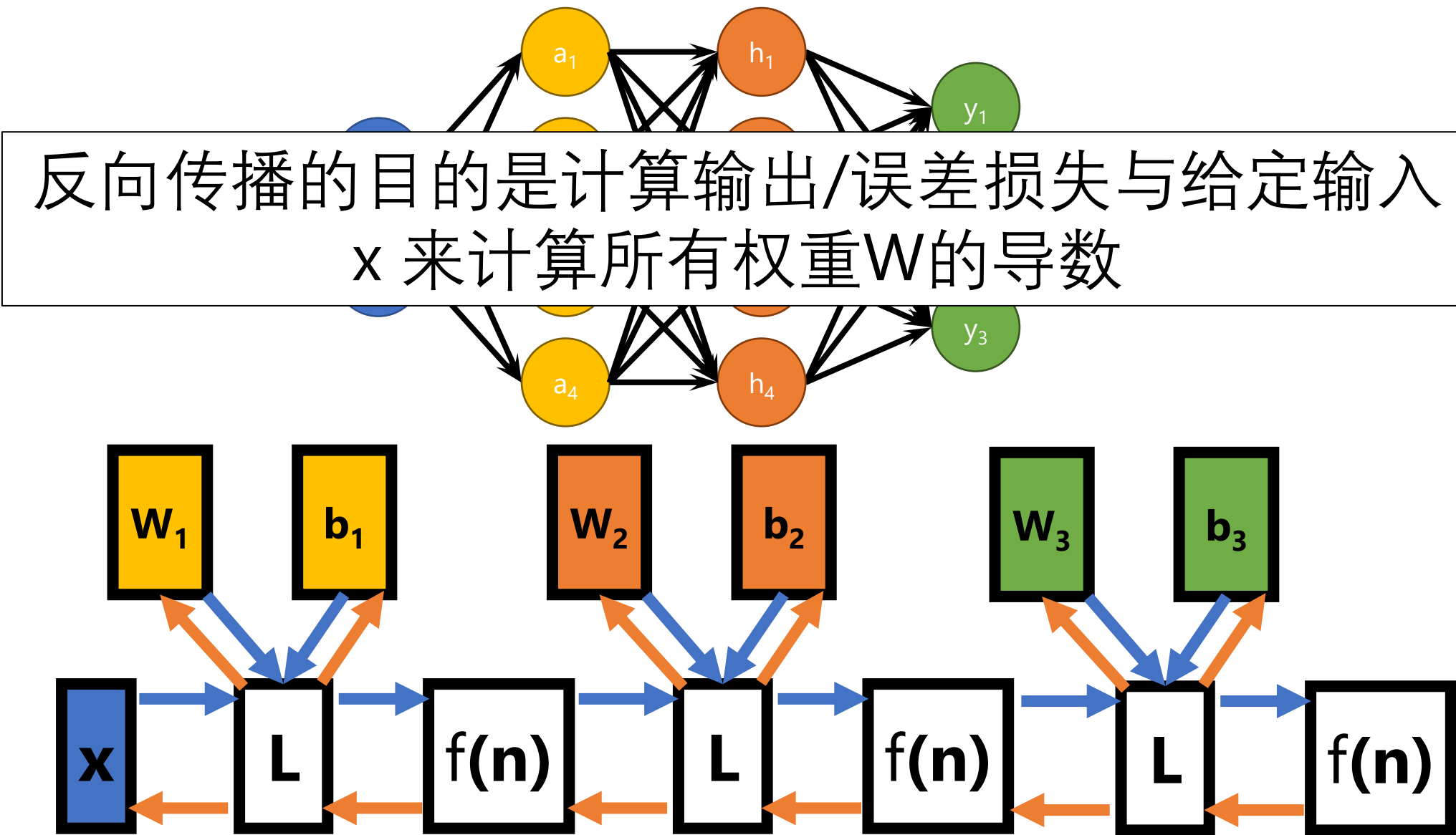
$$f(x) = (-x + 3)^2$$



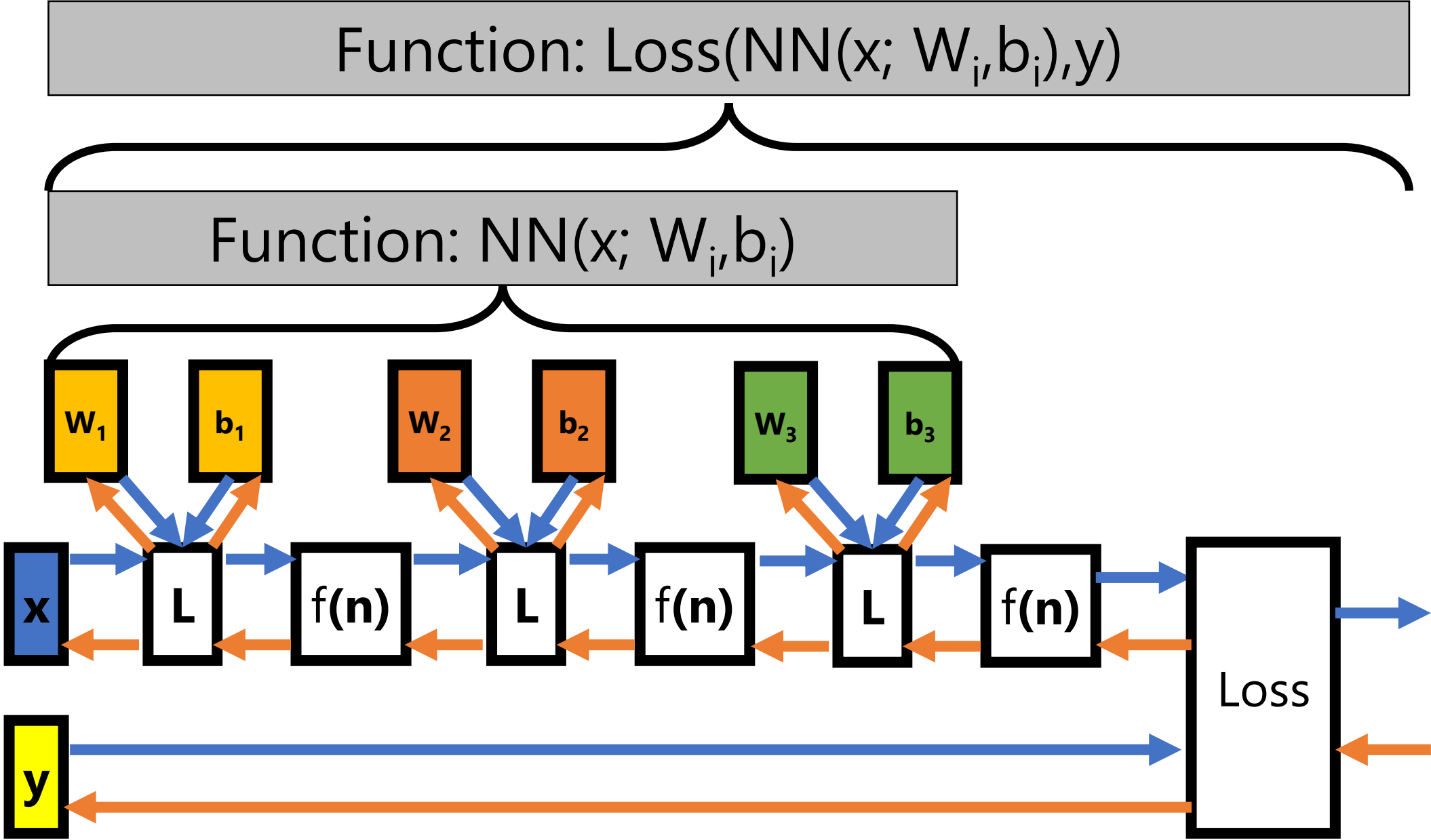
前向传递 **Forward pass**: 带入模块计算

反向传递 **Backward pass**: 计算相对模块和输入的
的导数

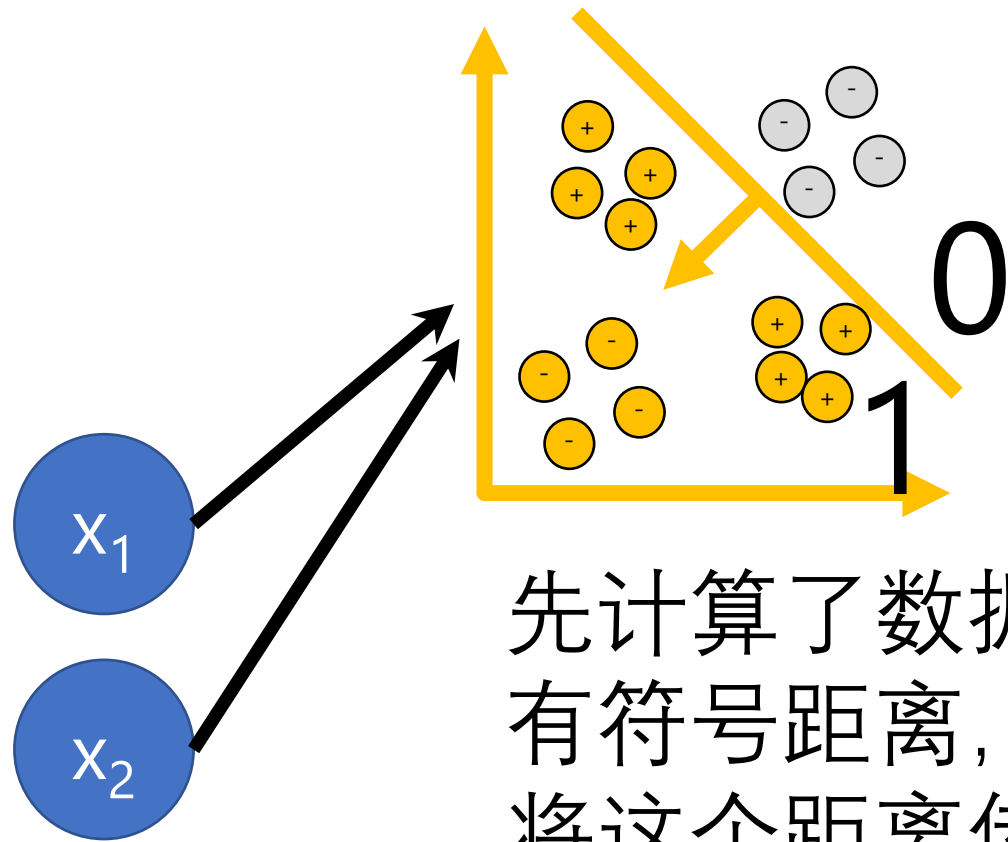
全连接网络



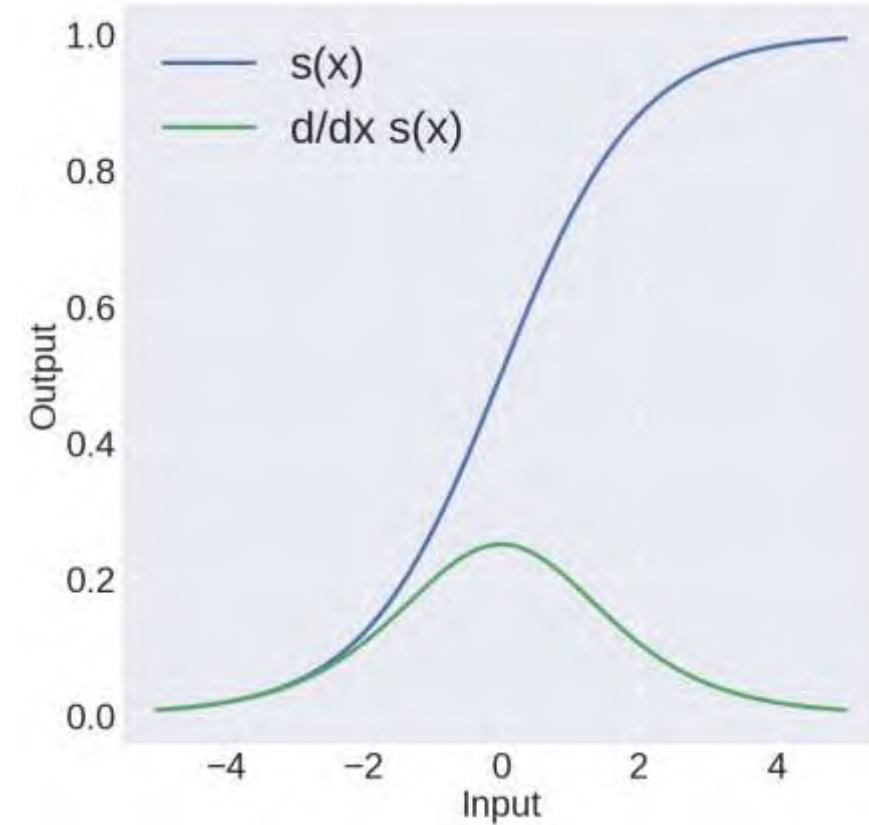
训练全连接网络



激活函数的引入



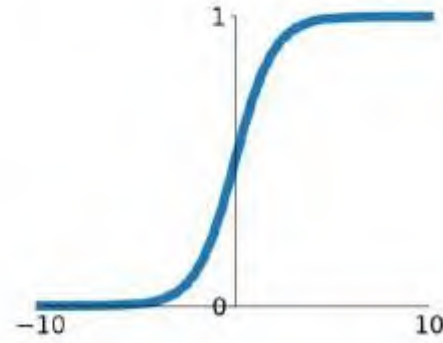
先计算了数据点的有符号距离，然后将这个距离传递给 sigmoid 激活函数



Activation functions

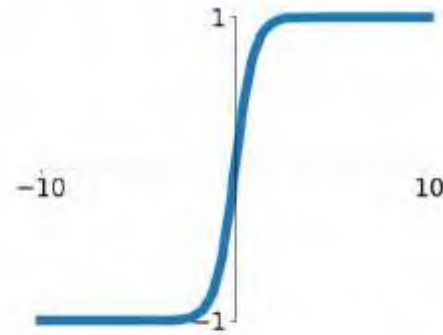
Sigmoid

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



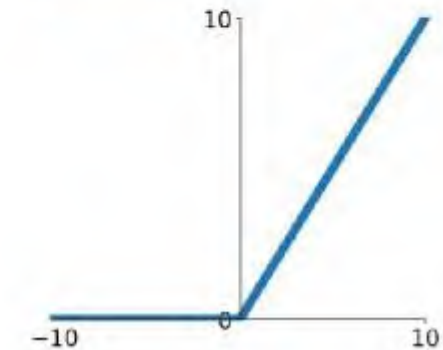
tanh

$$\tanh(x)$$



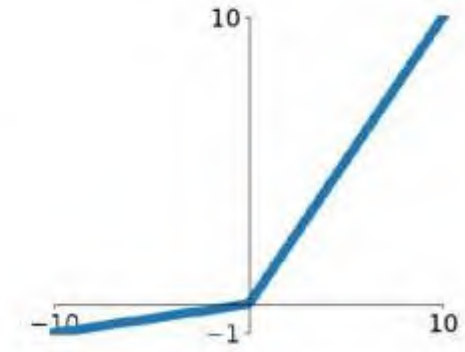
ReLU

$$\max(0, x)$$



Leaky ReLU

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

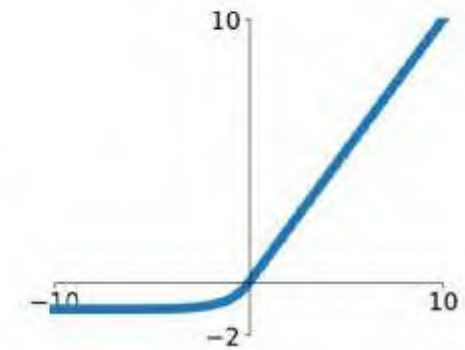


Maxout

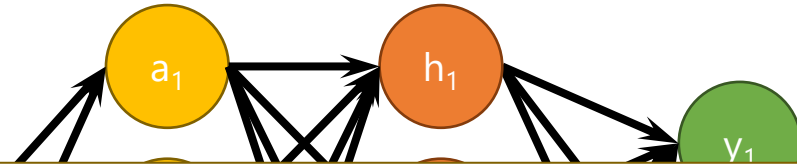
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

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

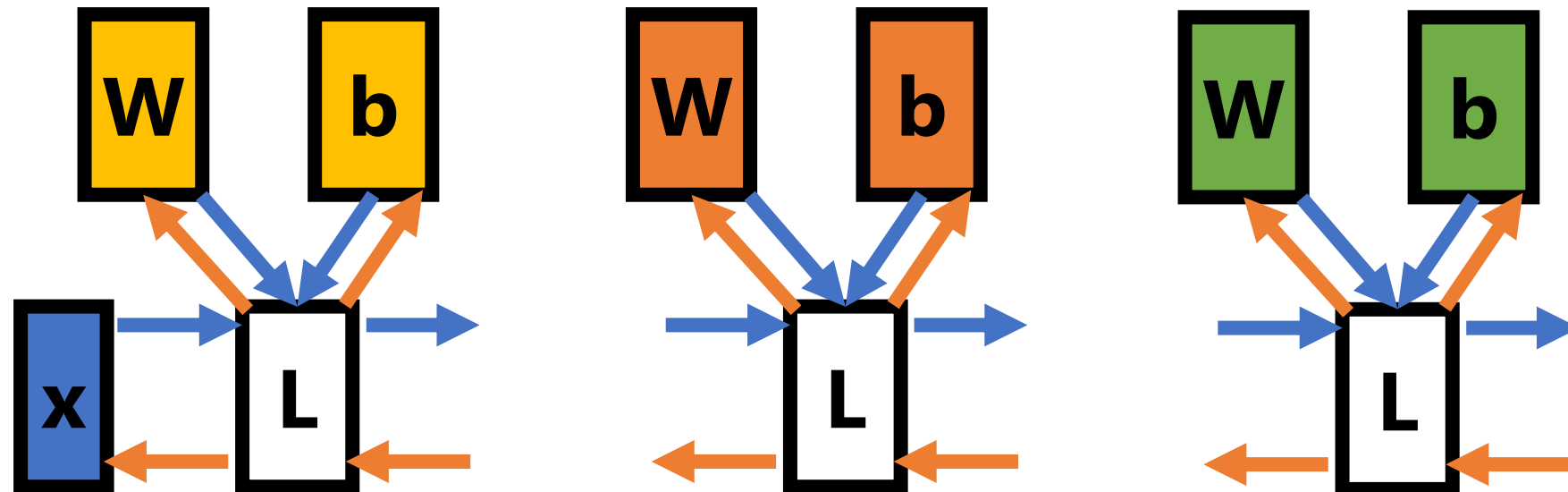


全连接网络



如果没有激活函数会发生什么？

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



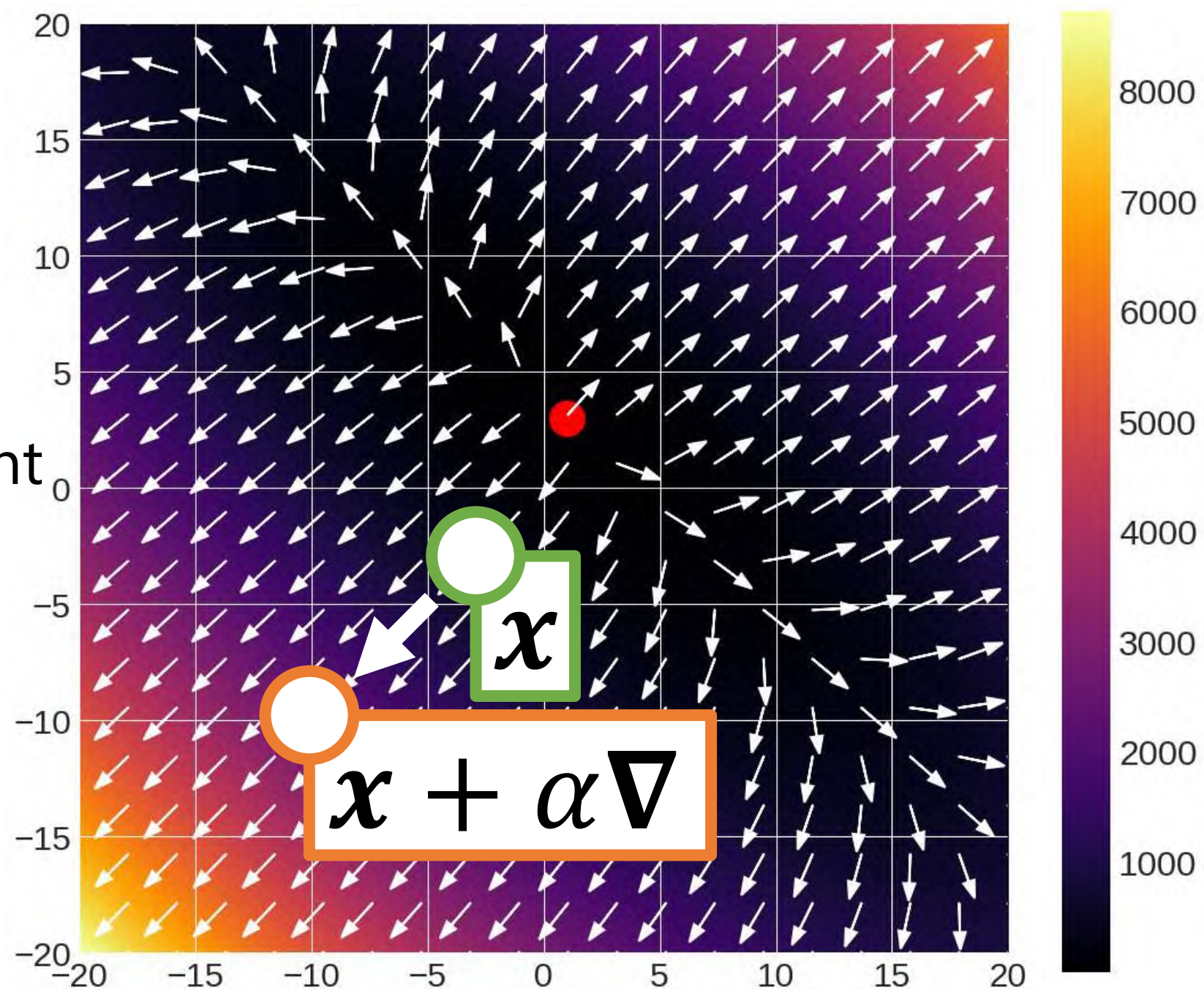
回顾——神经网络

梯度下降

每一步优化，向梯度的反方向移动

```
w0 = initialize() #initialize  
for iter in range(numIters):  
    g =  $\nabla_{\mathbf{w}}L(\mathbf{w})$  #eval gradient  
    w = w + -stepsize(iter)*g  
#update w  
return w
```

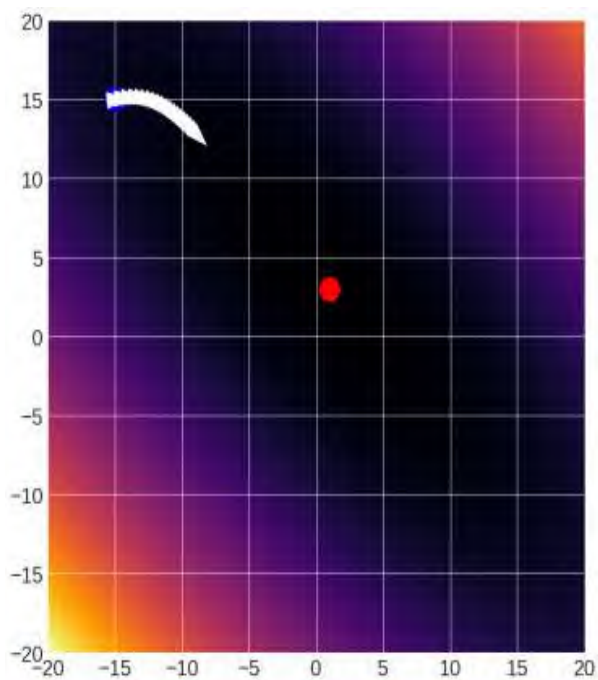
箭头: 梯度方向 (单位长度)



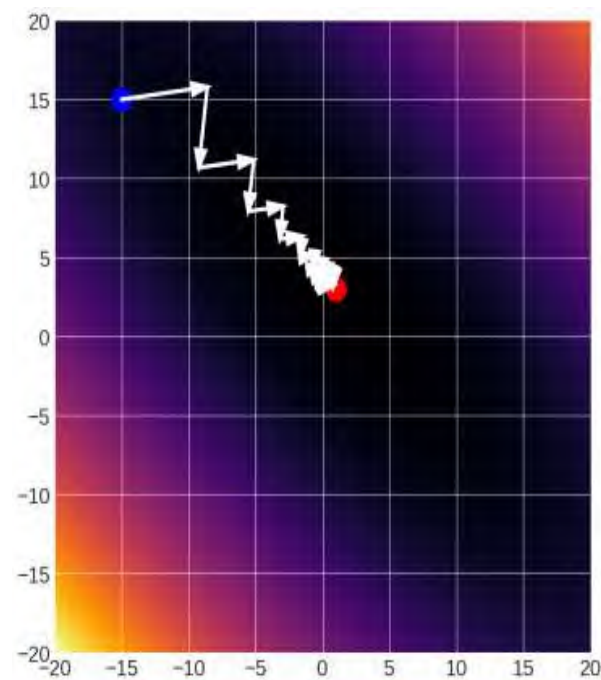
梯度下降 步长 Step size (学习率 **learning rate / lr**)

使用初始的 lr, 每N步以后乘以f

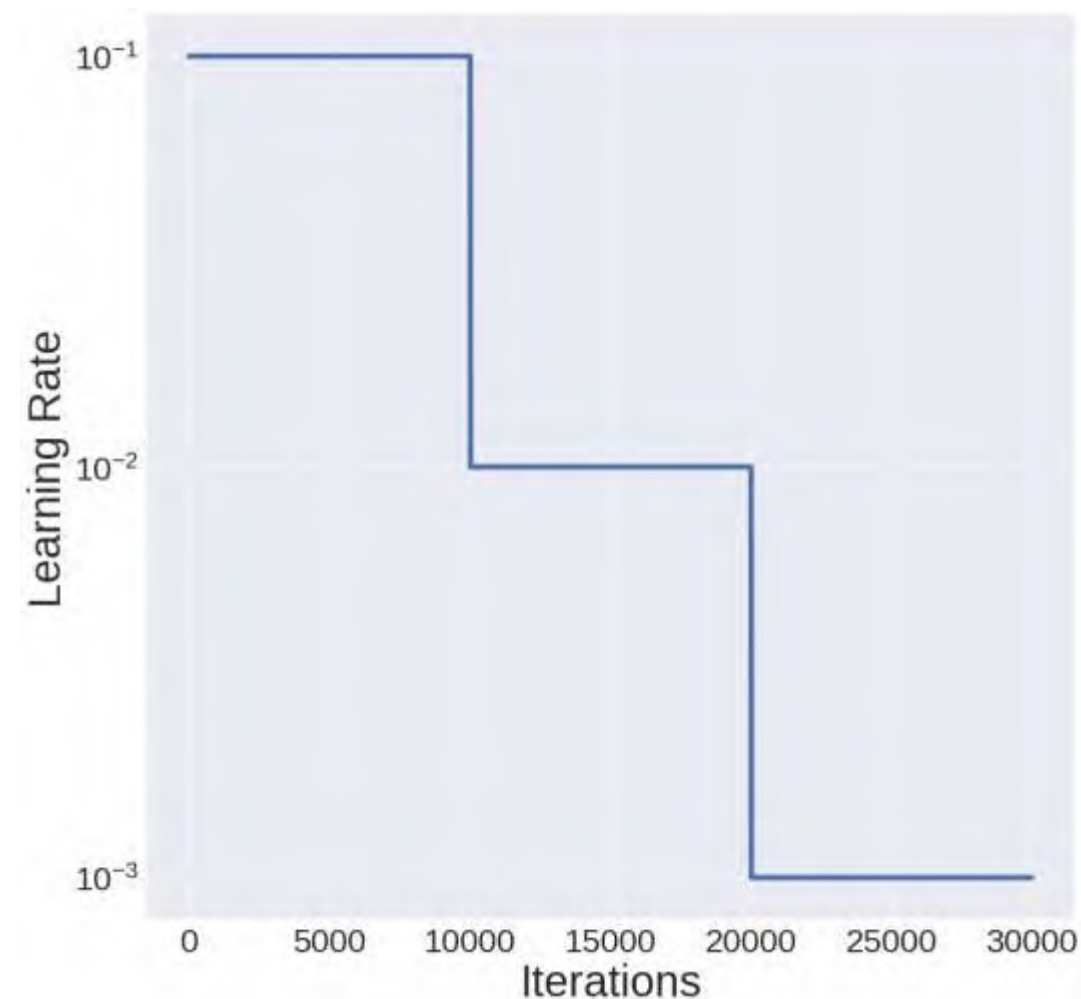
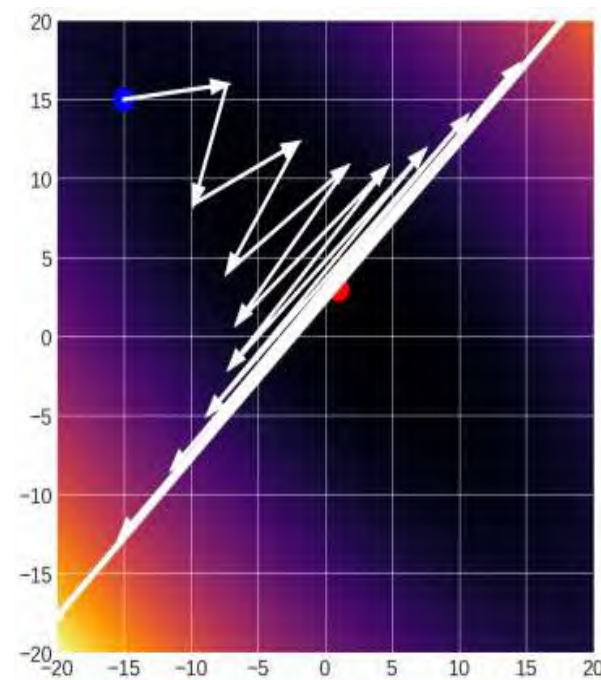
1×10^{-2}
falls short



10×10^{-2}
converges



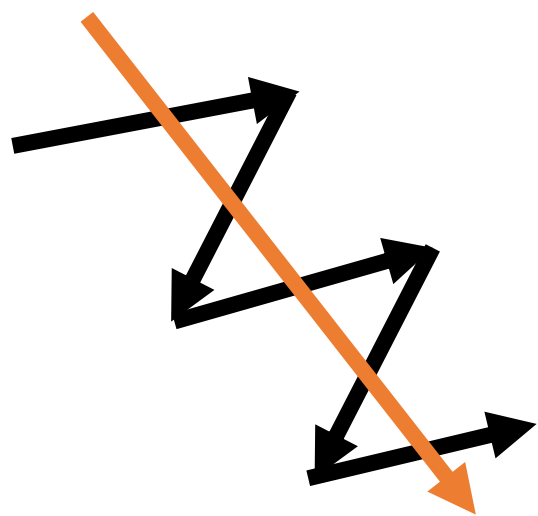
12×10^{-2}
diverges



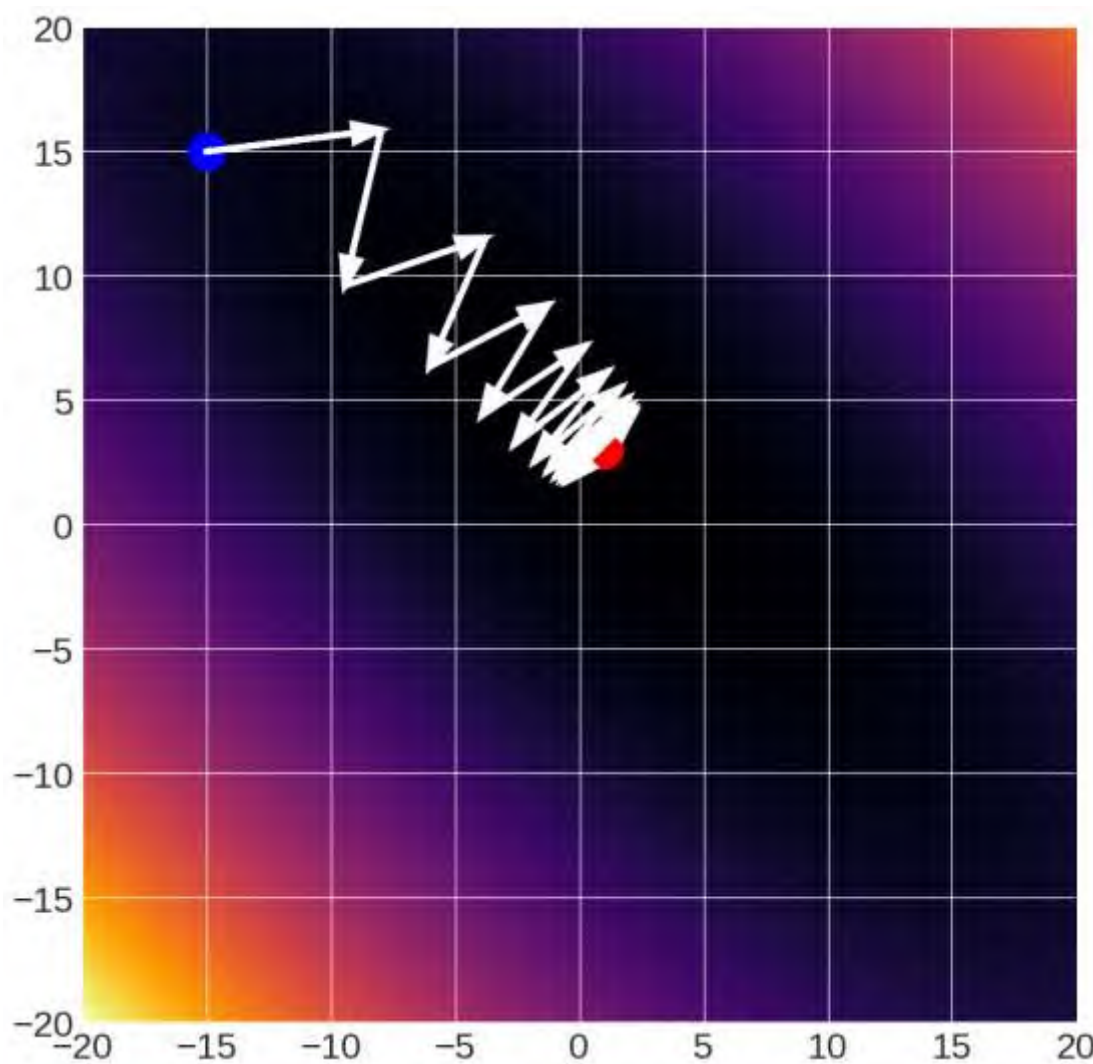
梯度下降

“平均梯度”

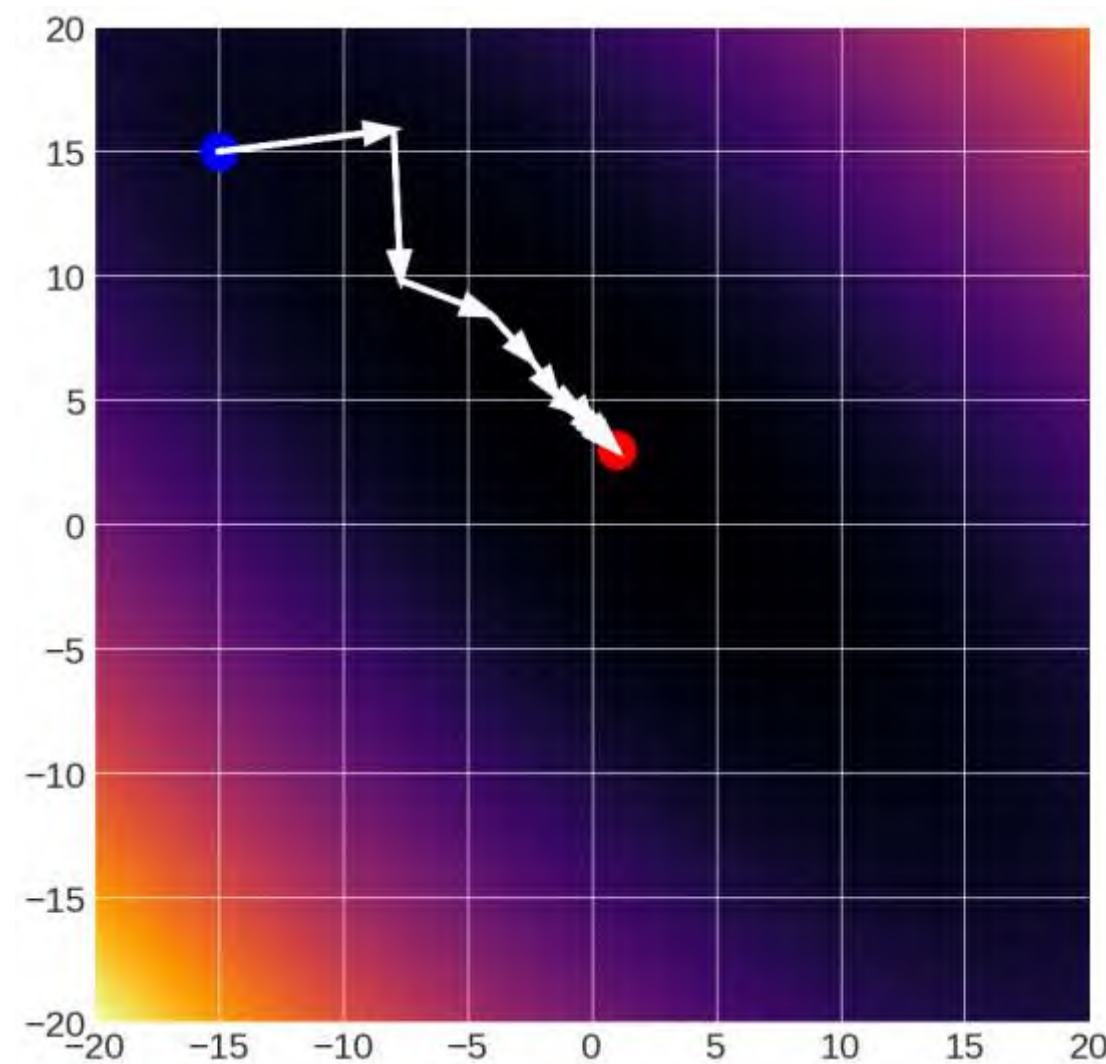
记录历史梯度，并且与当前步
梯度取加权平均， 也被称为
动量法 “momentum”



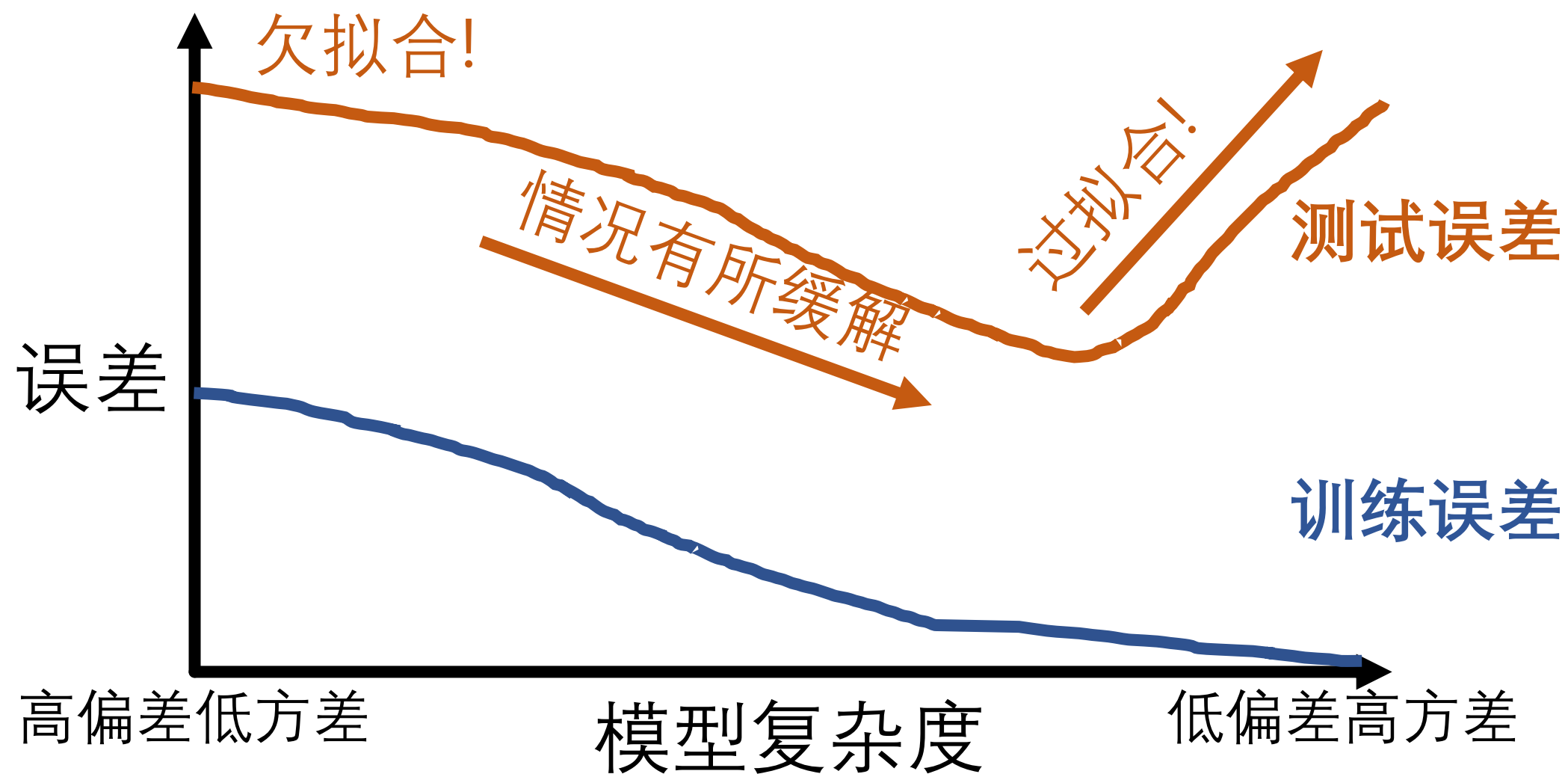
11×10^{-2} :oscillates
(Raw gradients)



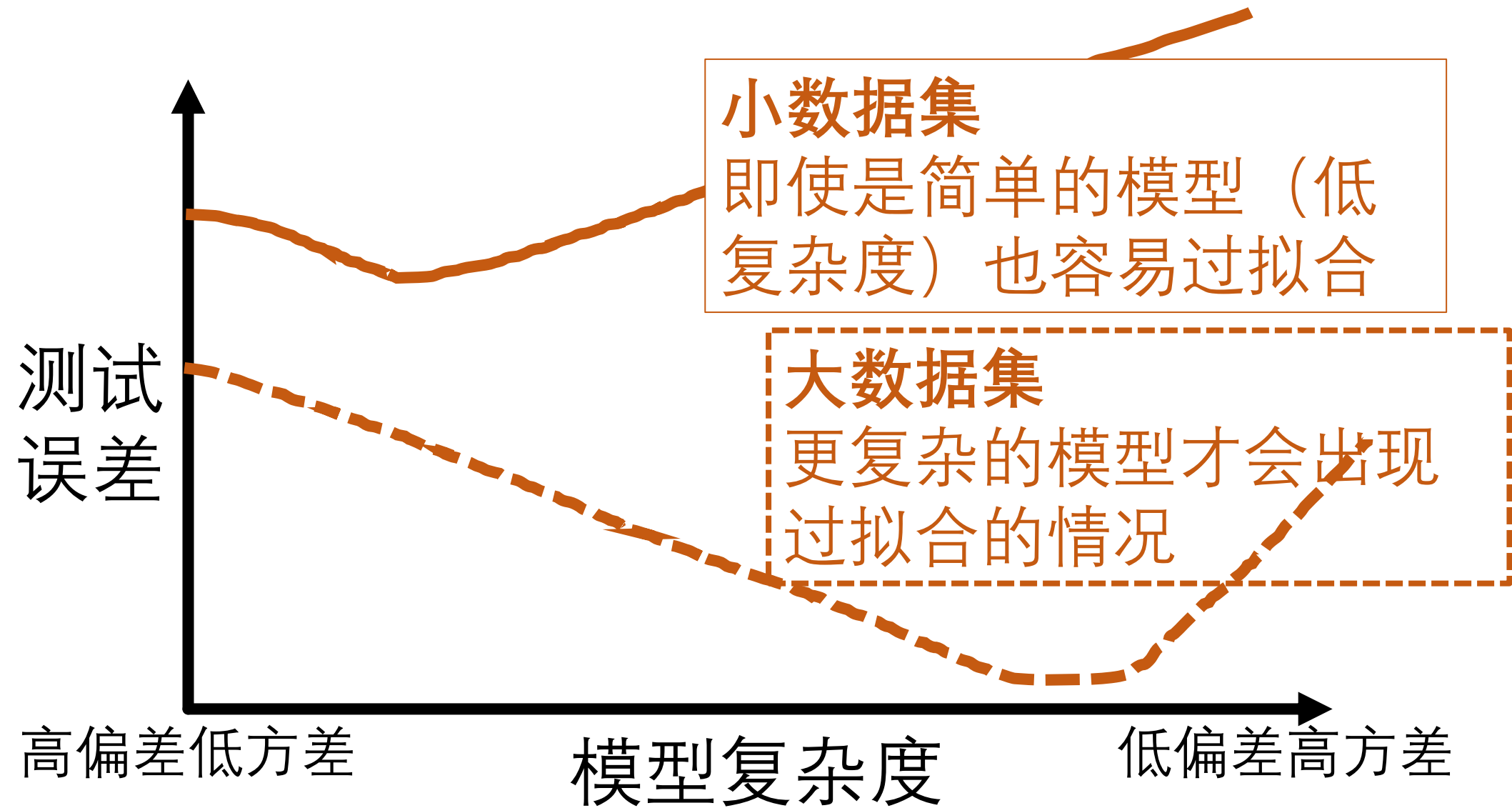
11×10^{-2}
(0.25 momentum)



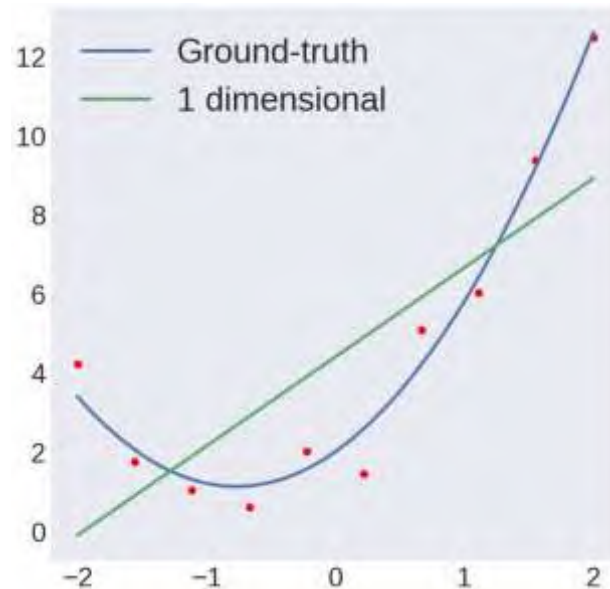
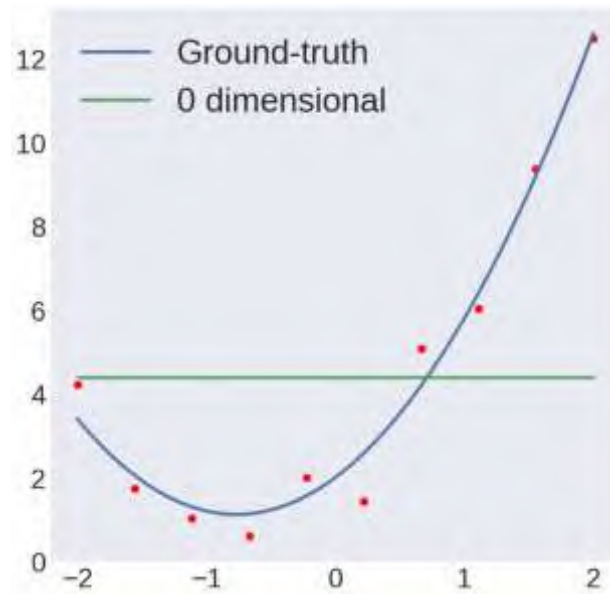
欠拟合与过拟合



欠拟合与过拟合



欠拟合

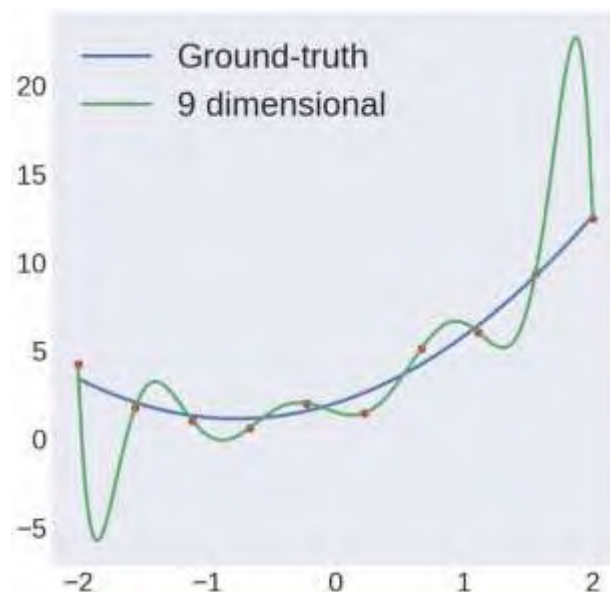
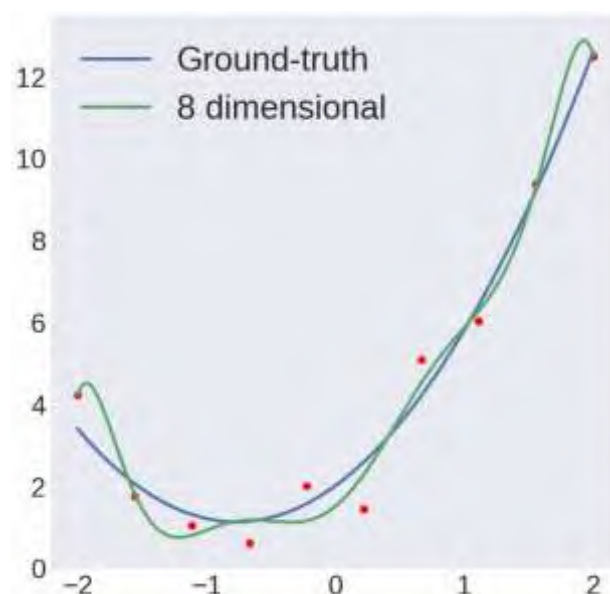


当模型的复杂度过低时，即使在训练数据上也得不到满意的拟合效果。这通常意味着模型太简单，不能捕获数据的真实模式。

解决方案:

- **增加更多特征**: 更多的特征可以增加模型的复杂度，使其更容易捕获到数据中的模式。
- **使用更强大的模型**: 如从线性模型升级到多项式模型或使用深度学习模型。
- **减少正则化**: 正则化是为了避免过拟合而添加的惩罚项，但如果模型欠拟合，则可以减少或移除正则化。

过拟合



模型在训练数据上表现很好，但在验证数据上表现较差，通常是因为模型太复杂，过度拟合了训练数据中的噪声或异常值。

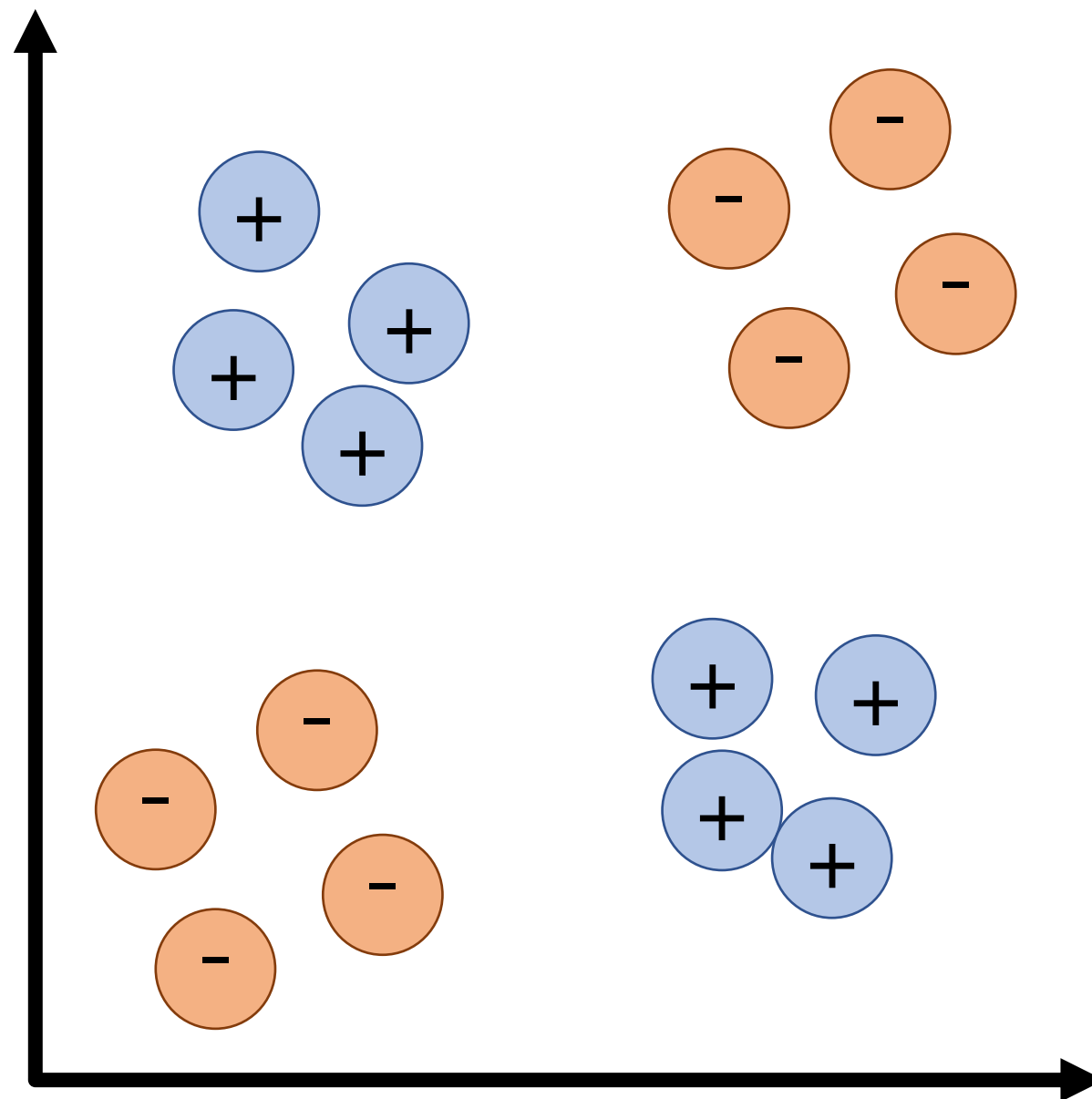
解决方案:

- **增加更多数据**: 更多的训练数据可以帮助模型学习到更广泛的数据分布，从而减少过拟合。
- **使用简单模型**: 选择一个复杂度较低的模型，避免过度拟合。
- **增加正则化**: 正则化可以防止模型过于复杂，从而限制其拟合能力。

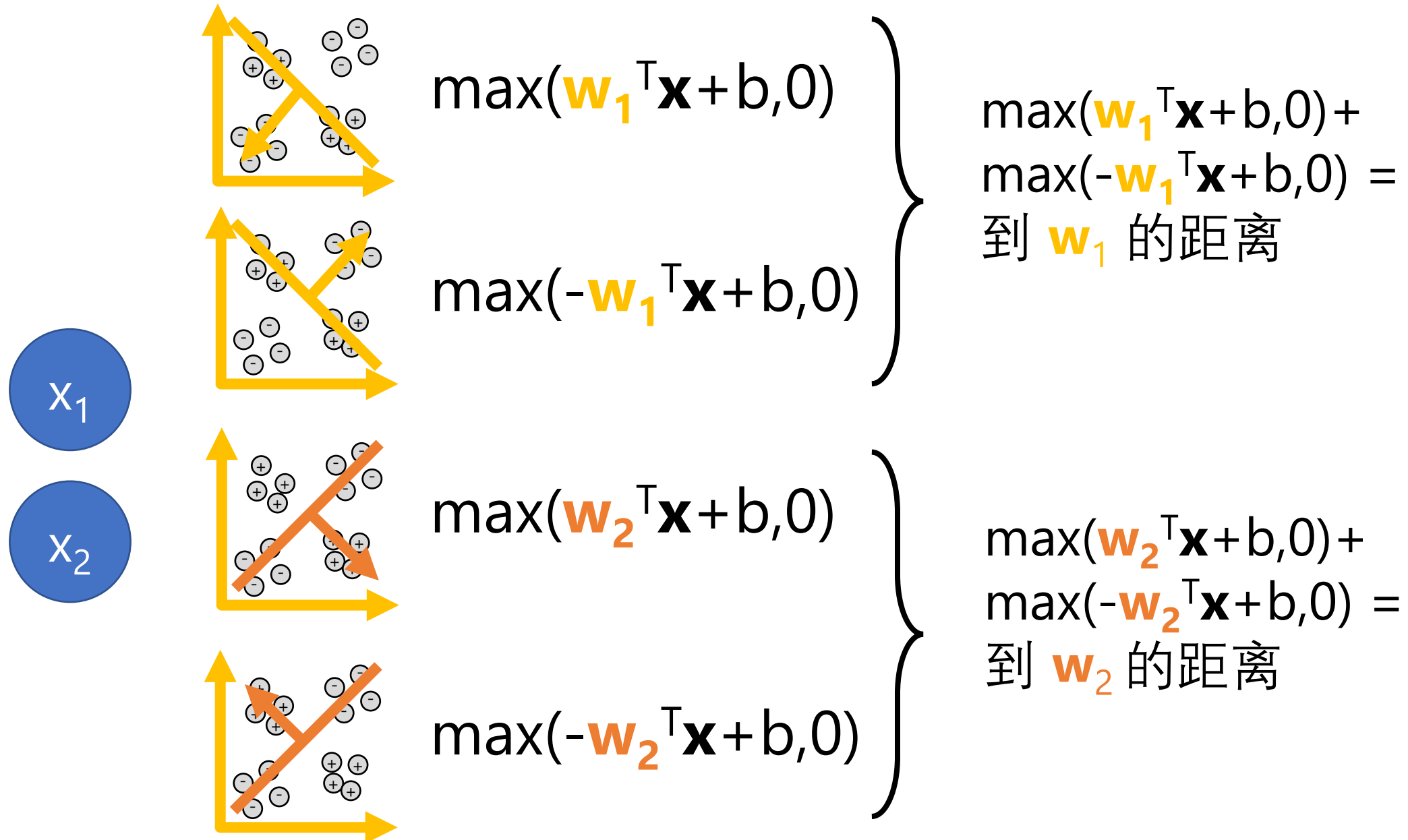
建议: 首先确保模型有足够的能力进行拟合 (即避免欠拟合), 然后再考虑如何避免过拟合。

非线性可分

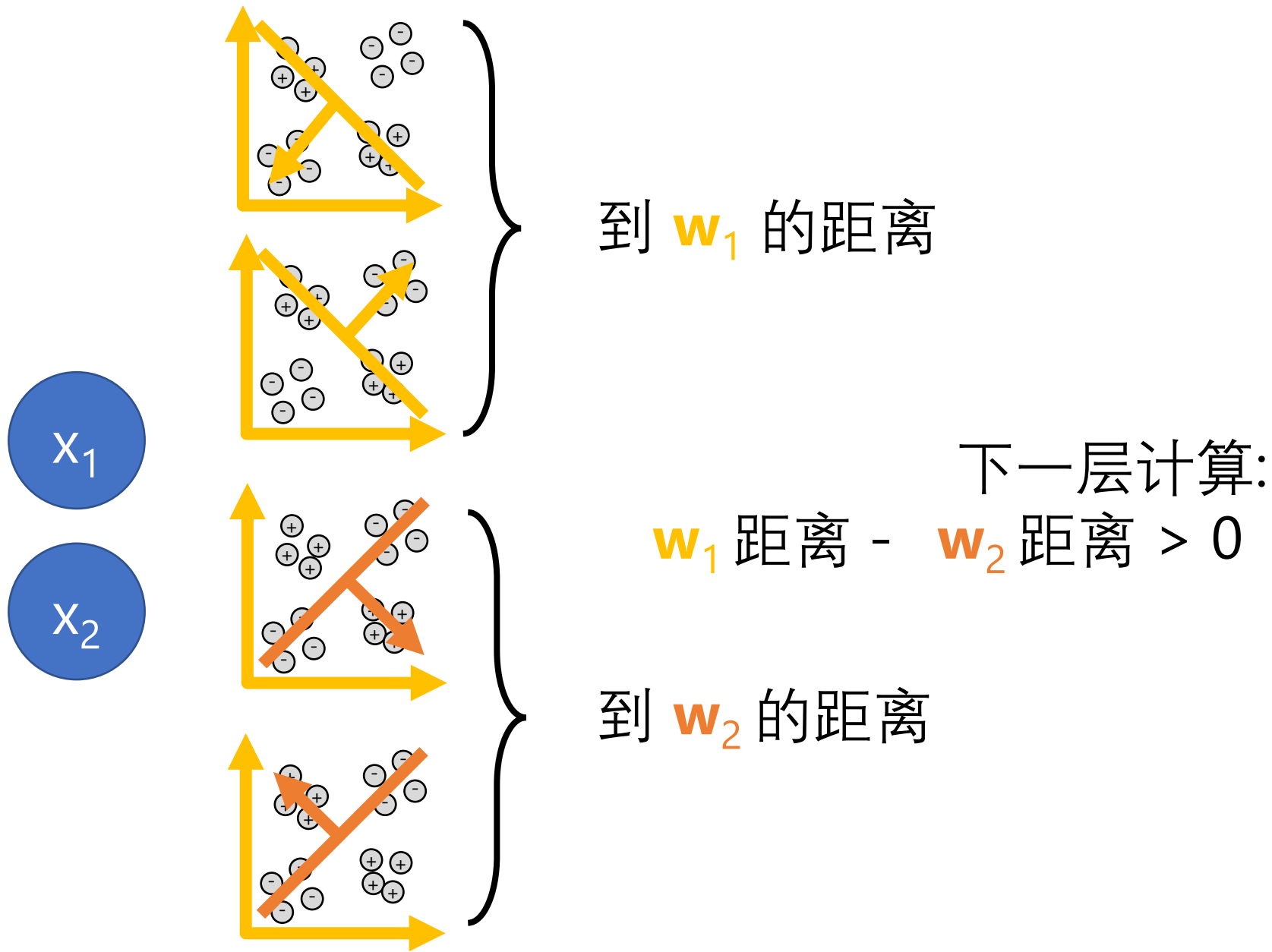
- 数据并非线性可分



非线性激活函数



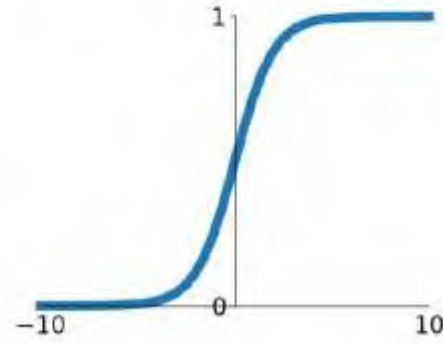
非线性激活函数



Activation functions

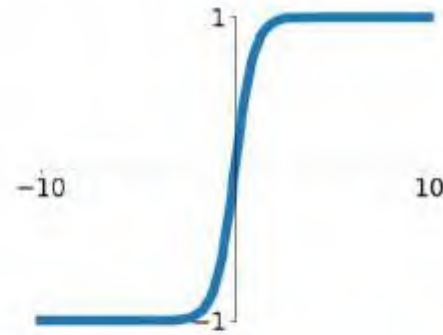
Sigmoid

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



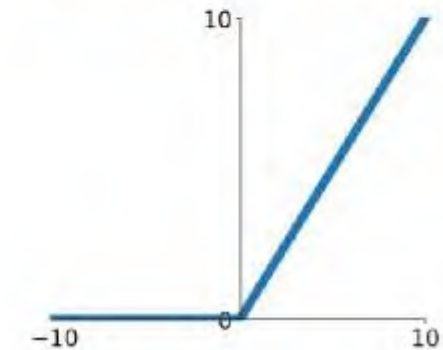
tanh

$$\tanh(x)$$



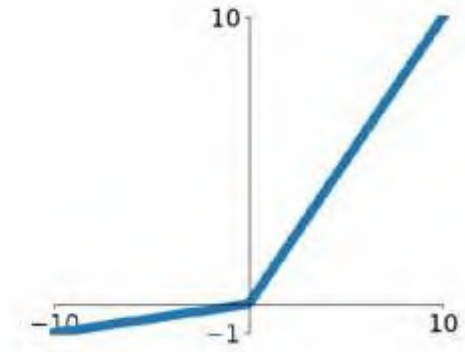
ReLU

$$\max(0, x)$$



Leaky ReLU

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

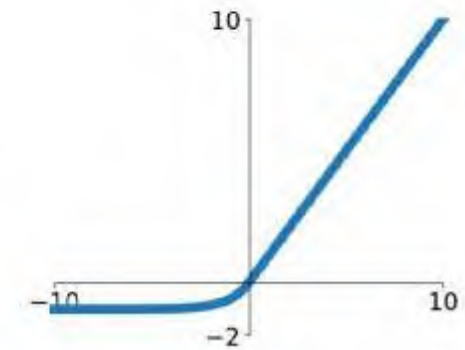


Maxout

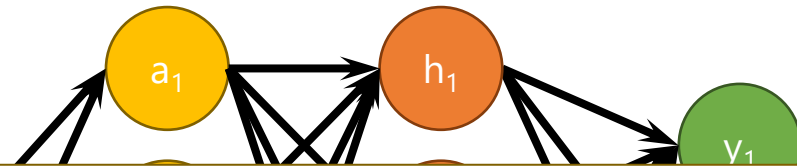
$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

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

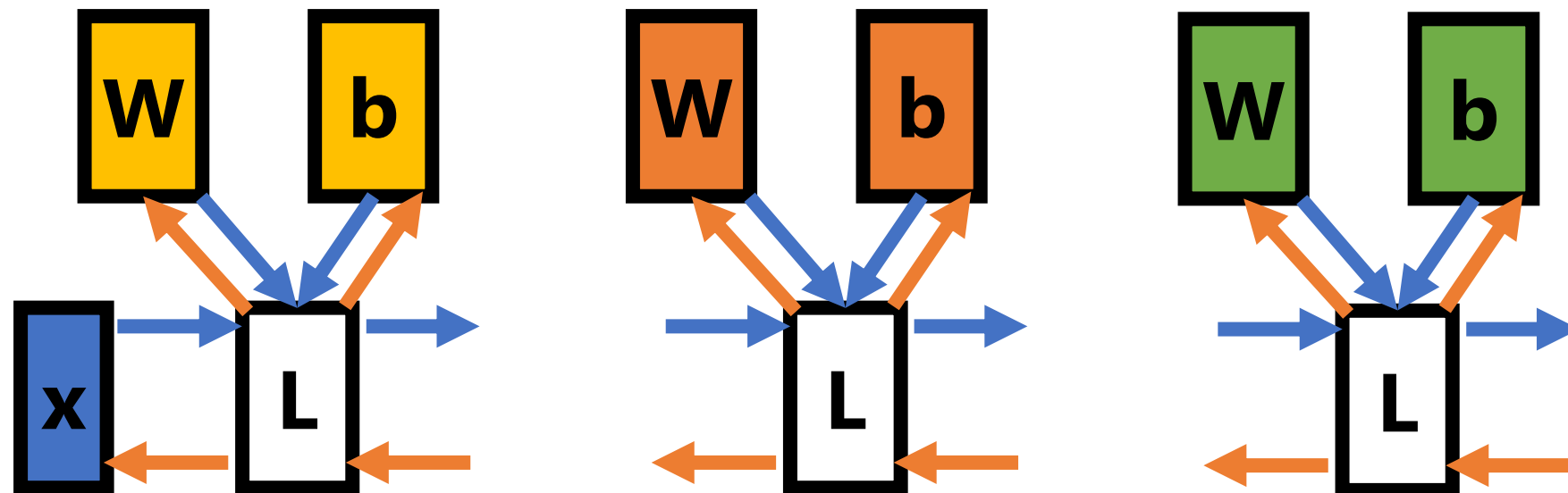


全连接网络



如果没有激活函数会发生什么？

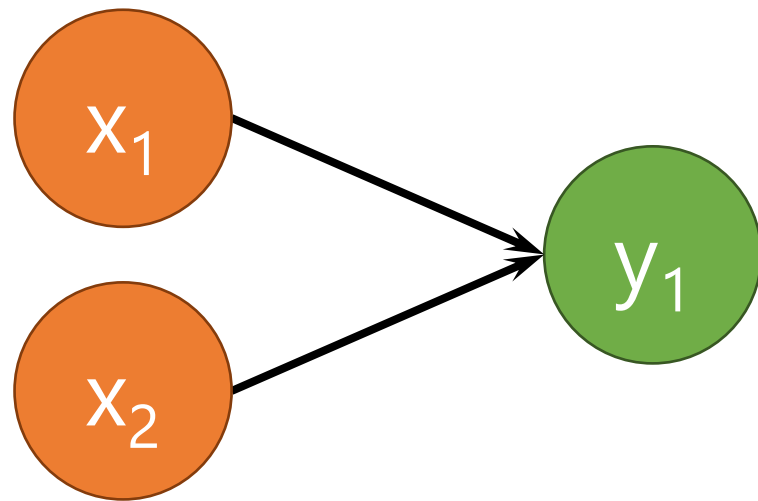
<https://cs.stanford.edu/people/karpathy/convnetjs/demo/classify2d.html>



卷积神经网络

参数量

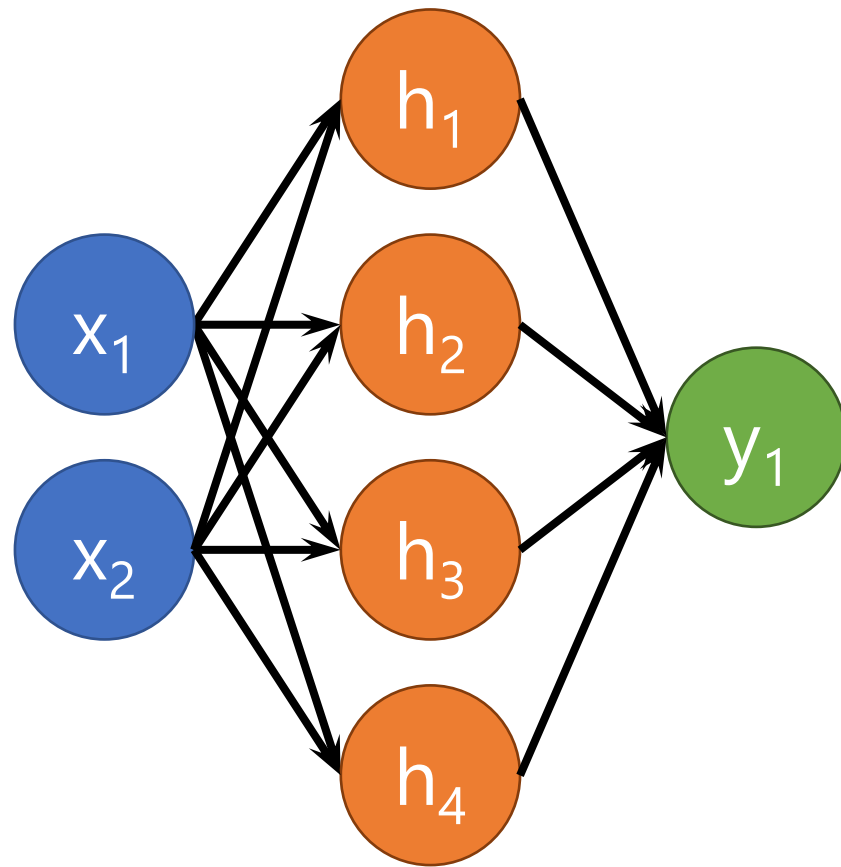
这个网络有多少参数?



Weights 权重: 1x2
总参数量: 3 (不要忘了偏置 bias!)

参数量

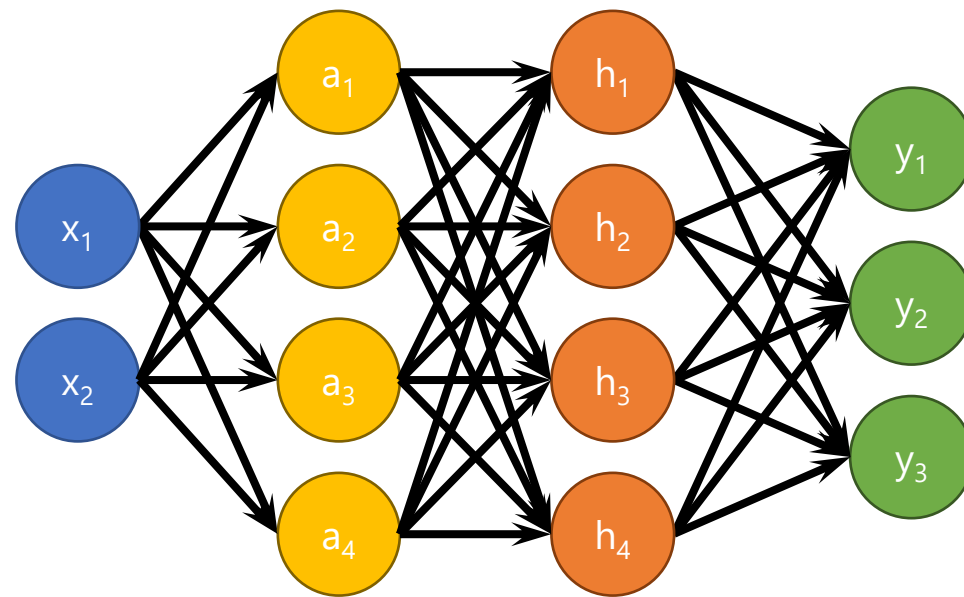
这个网络有多少参数?



权重: $1 \times 4 + 4 \times 2 = 12$
总参数量: $12 + 5 = 17$

参数量

这个网络有多少参数?



$$\text{权重: } 3 \times 4 + 4 \times 4 + 4 \times 2 = 36$$

$$\text{总参数量: } 36 + 11 = 47$$

参数量



$P \times 1$
vector

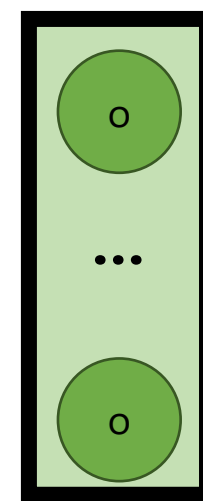
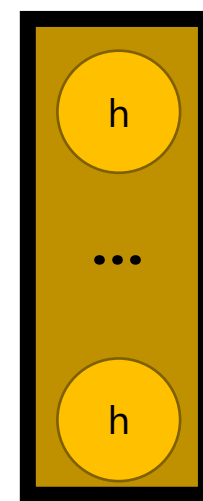
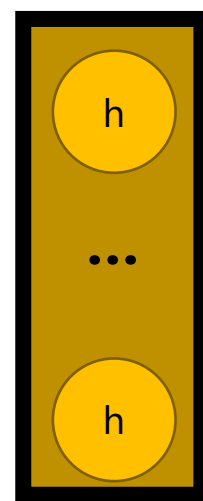
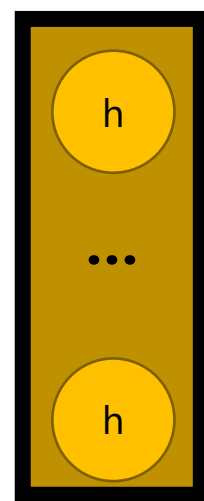
x

H^*P+
 H
 H
neurons

H^*H
 $+H$
 H
neurons

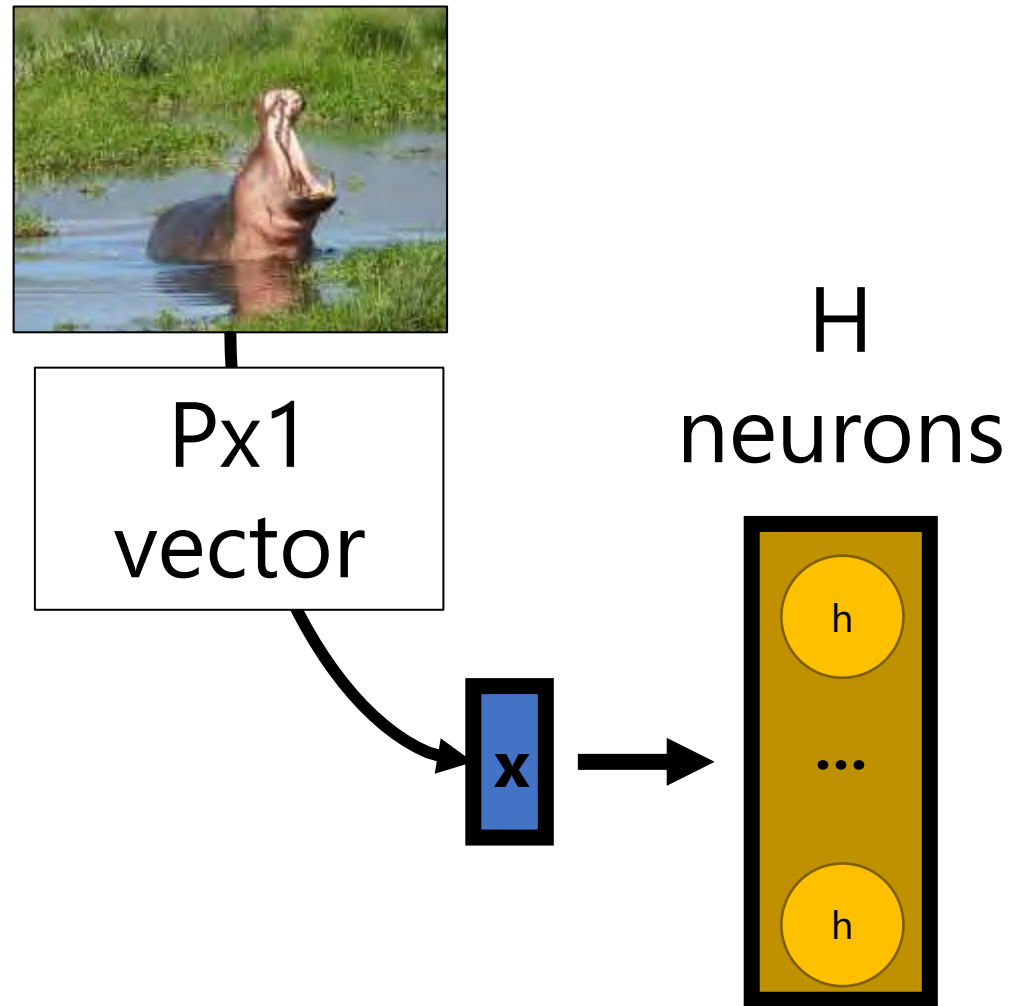
H^*H
 $+H$
 H
neurons

O^*H
 $+O$
 O
neurons



$P: 285 \times 350$, $H: 1000$, $O: 3$
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 $P \cdot H + H$
- 如果我们要计算图像梯度 dx/dy . 我们需要多少H?
- **2P!**

参数量



$P \times 1$
vector

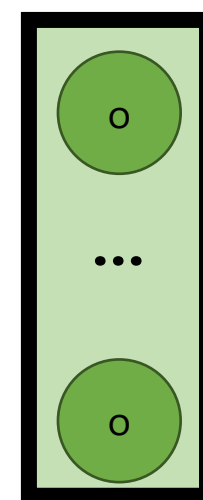
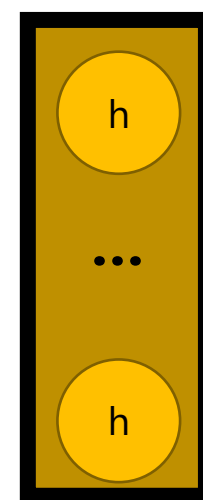
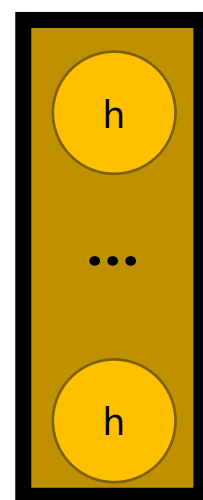
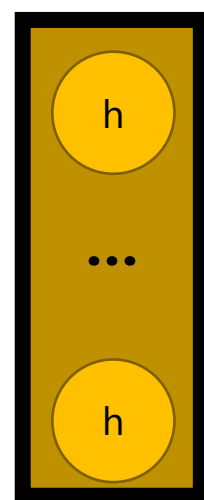
x

H^*P+
 H
 H
neurons

H^*H
 $+H$
 H
neurons

H^*H
 $+H$
 H
neurons

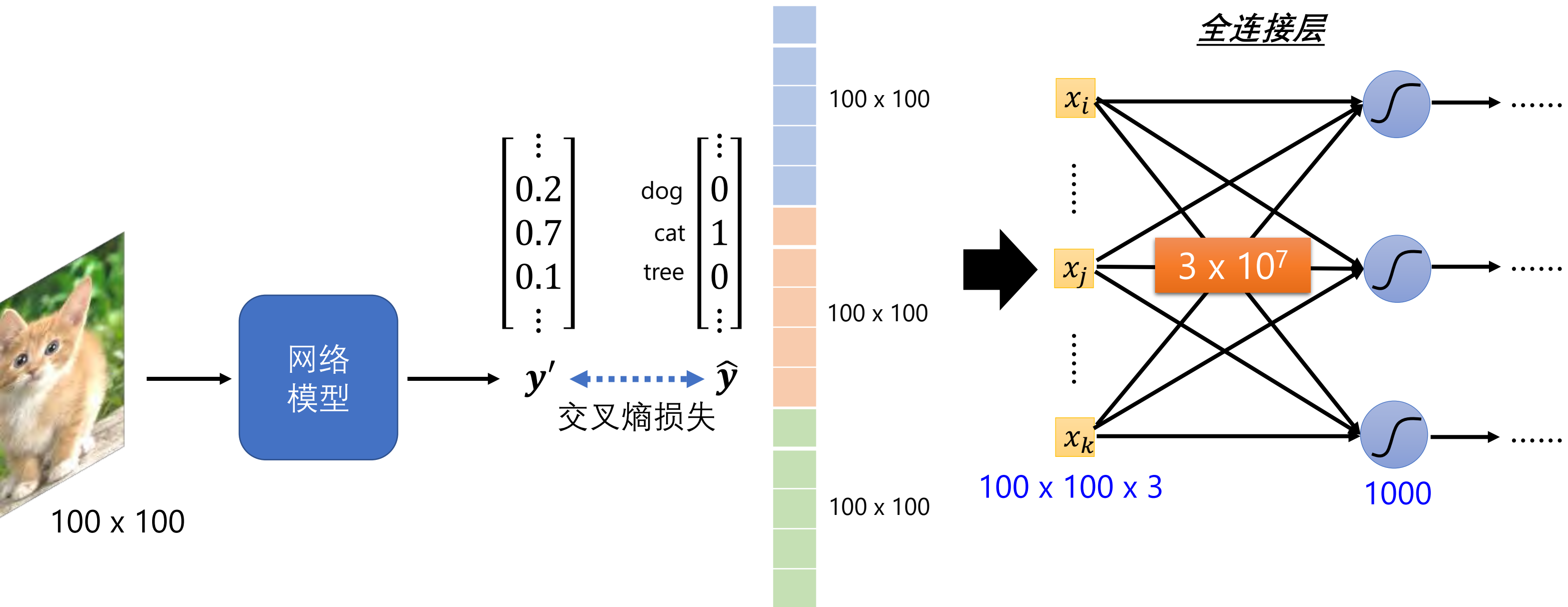
O^*H
 $+O$
 O
neurons



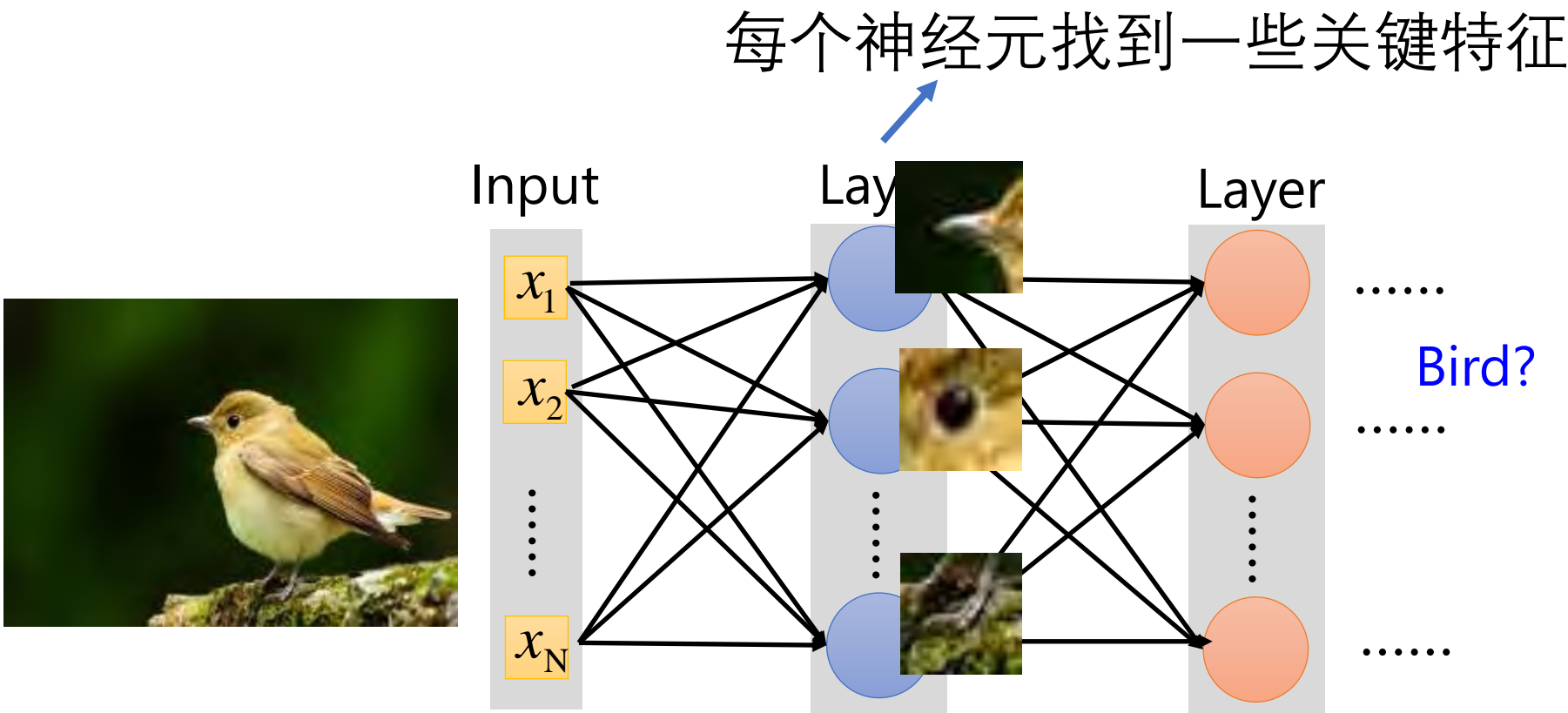
$P: 285 \times 350, H: 2P, O: 3$

100 billion 个参数 (400GB)

我们真的需要这么多的参数量吗



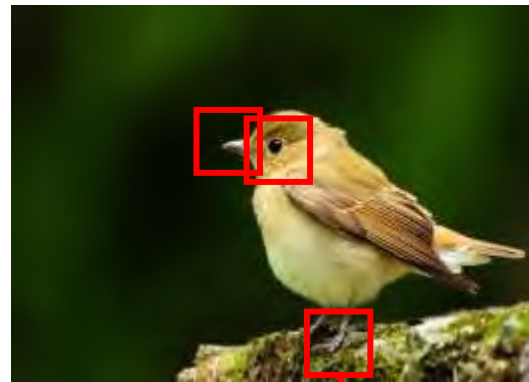
简化全连接层



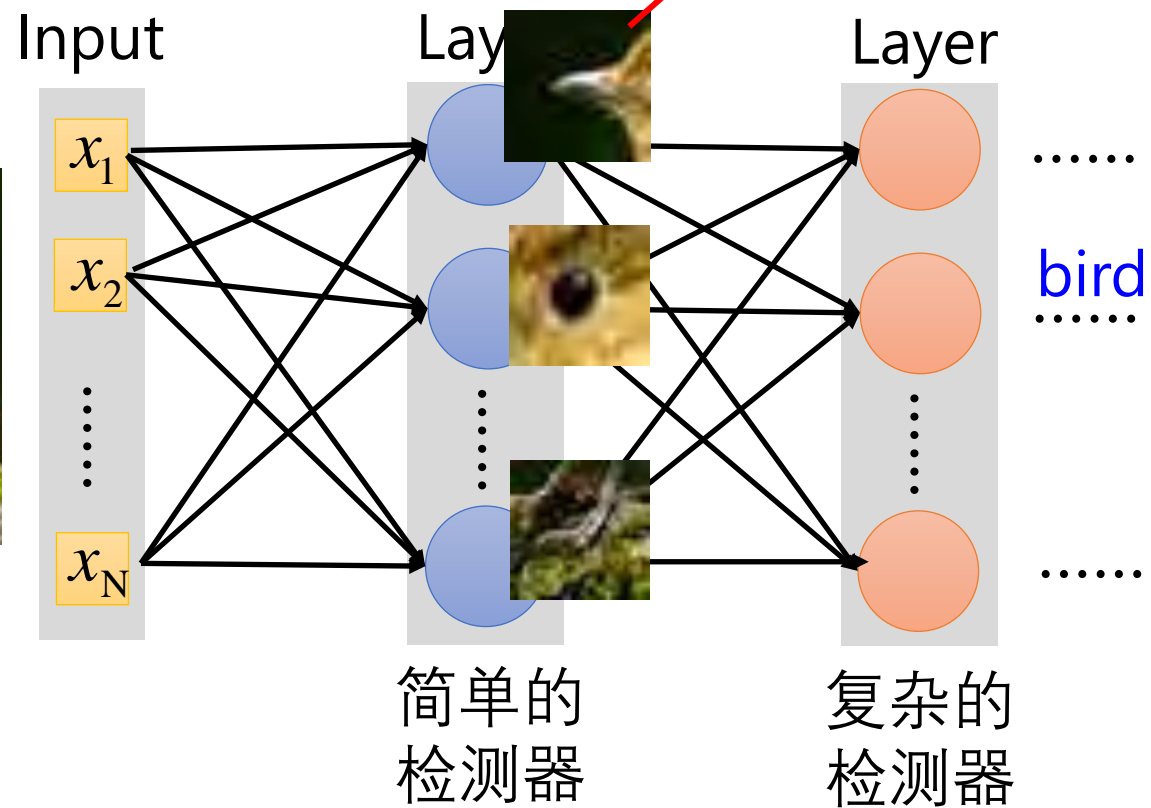
与前面提到的特征检测——提取——匹配类似，
也许人也是用类似的方式来识别鸟...

简化全连接层

我们需要看整张图来分类吗？
(回顾特征点)



每个神经元并不需要看到所有像素点 (回顾特征尺度)

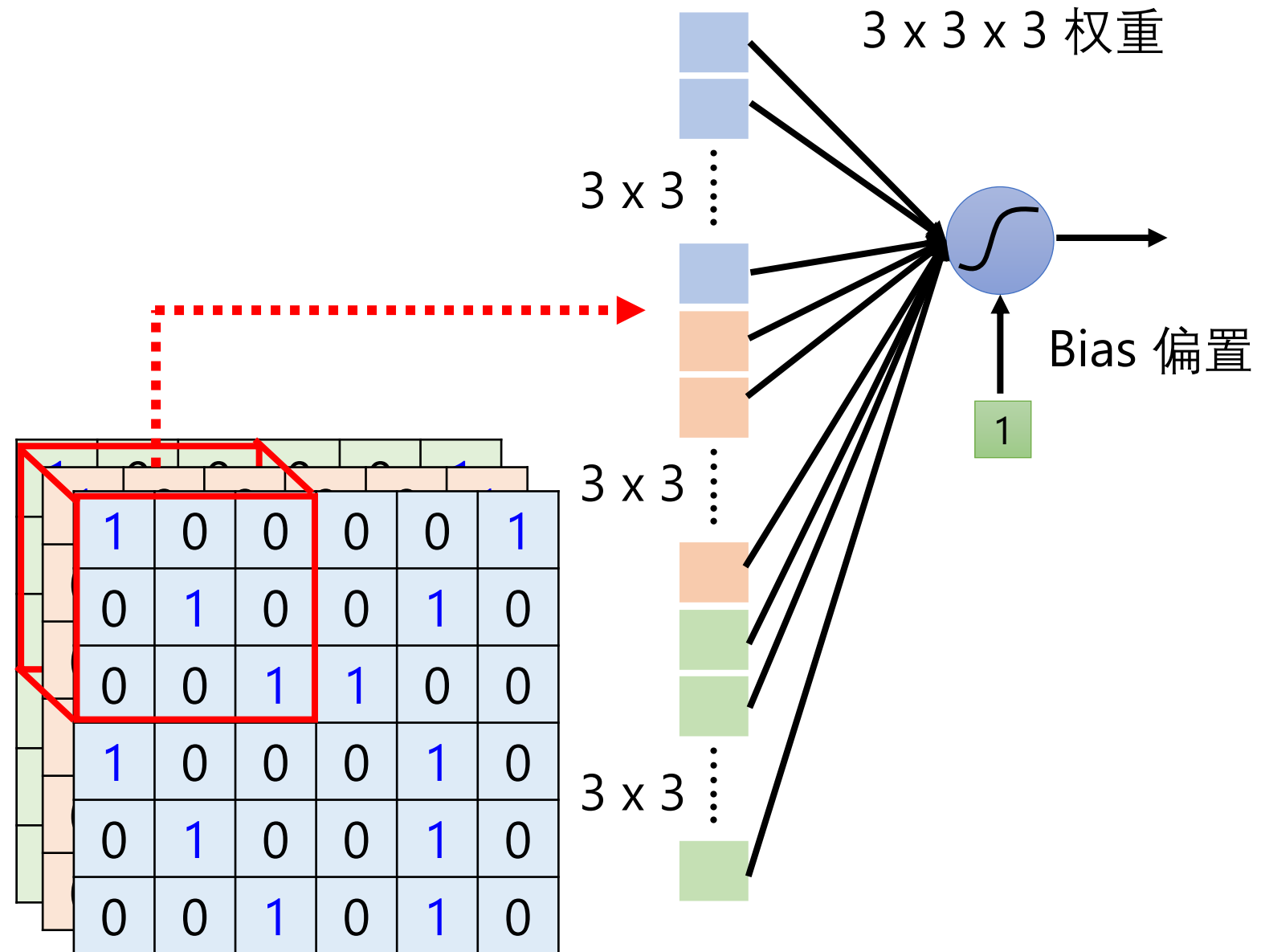


有些关键特征也许很小 (回顾特征尺度)

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。

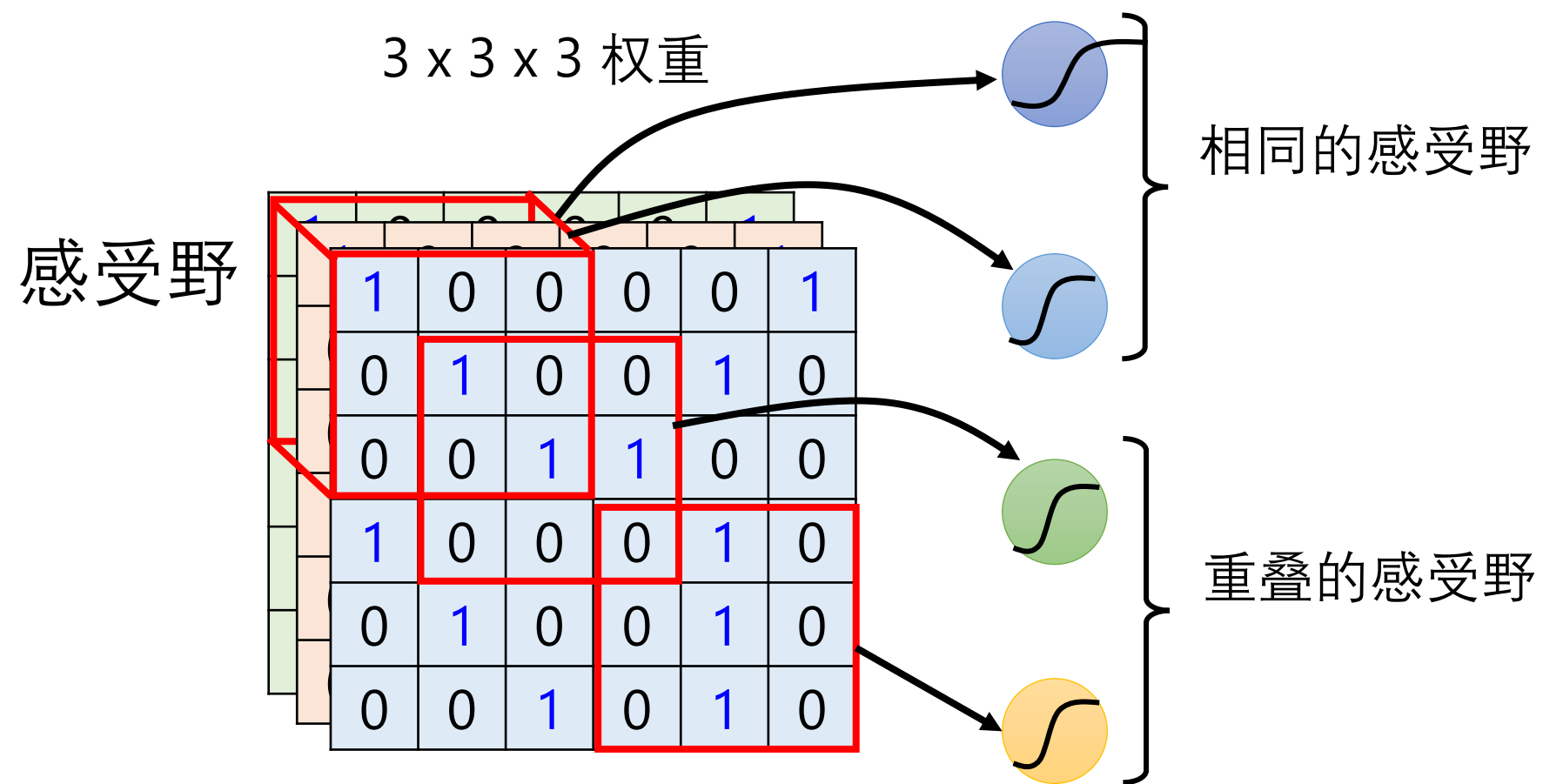
感受野 Receptive Field

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。



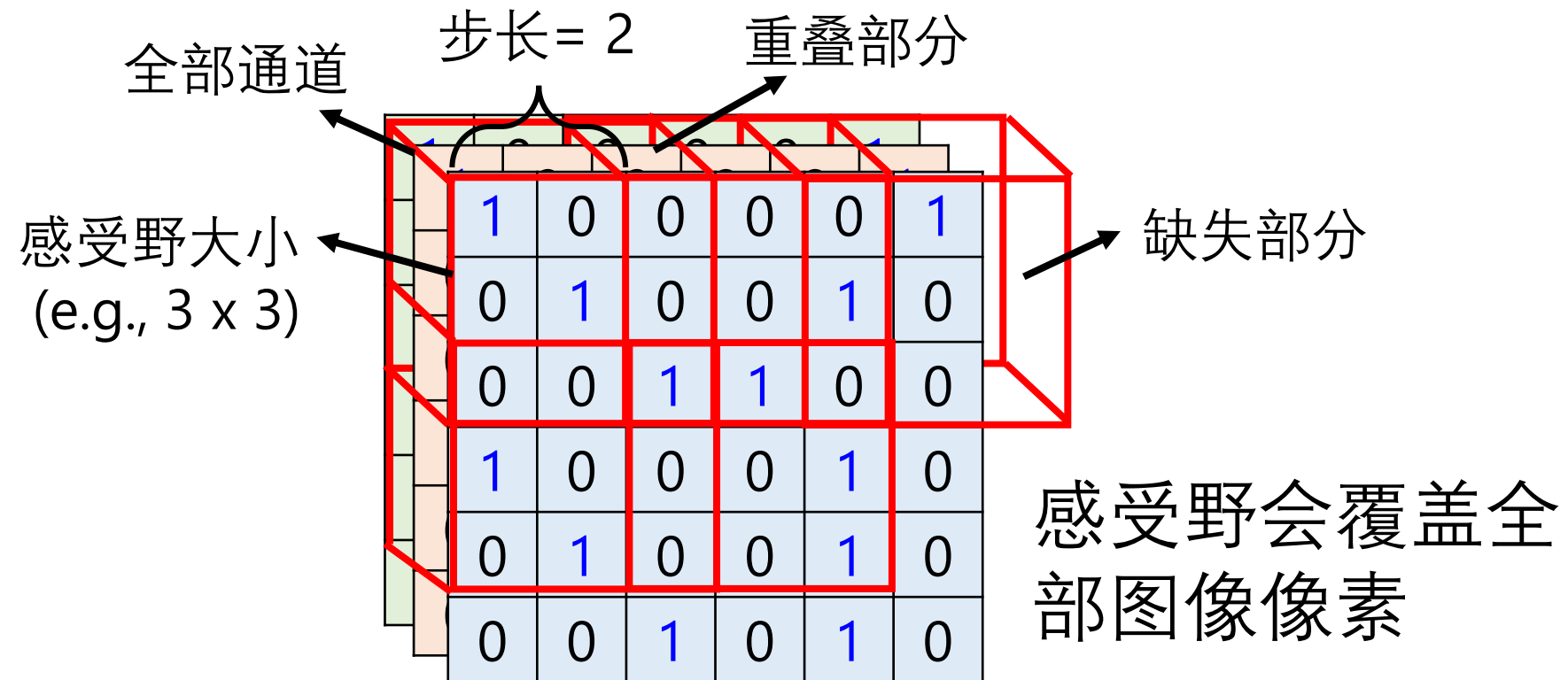
感受野

- 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？



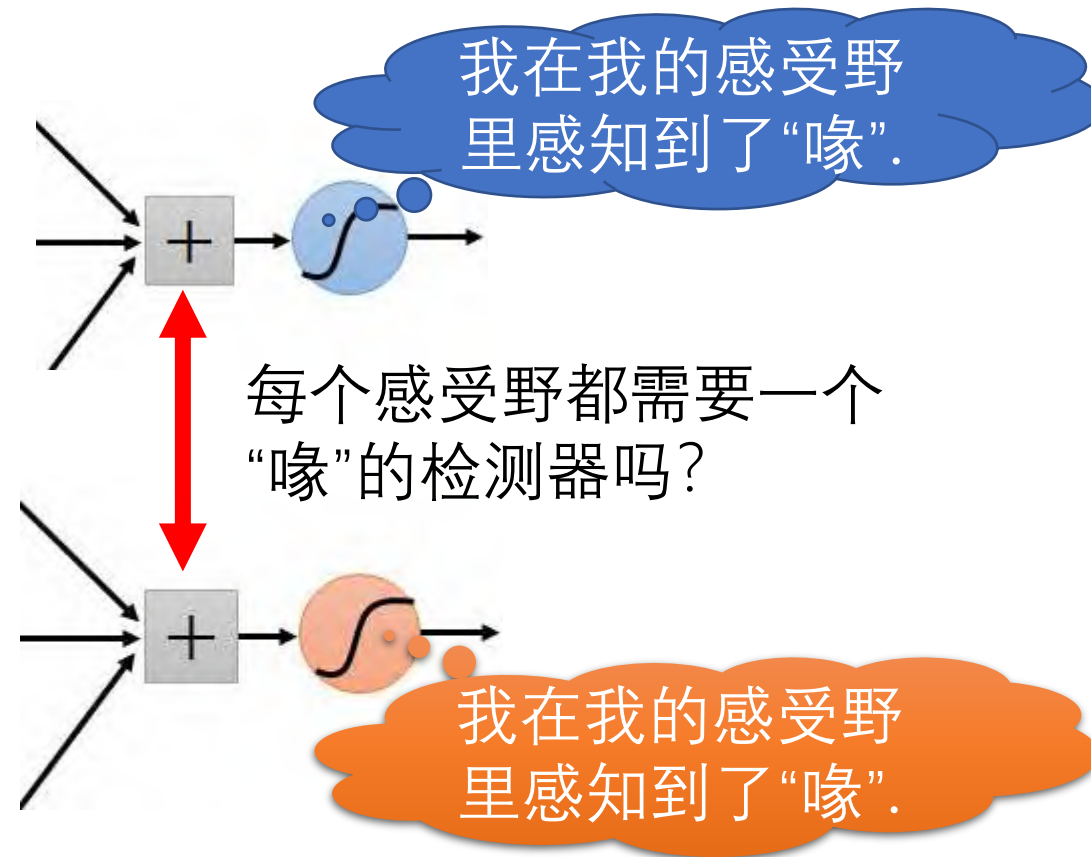
感受野

每个感受野会占用某些神经元（通道）。

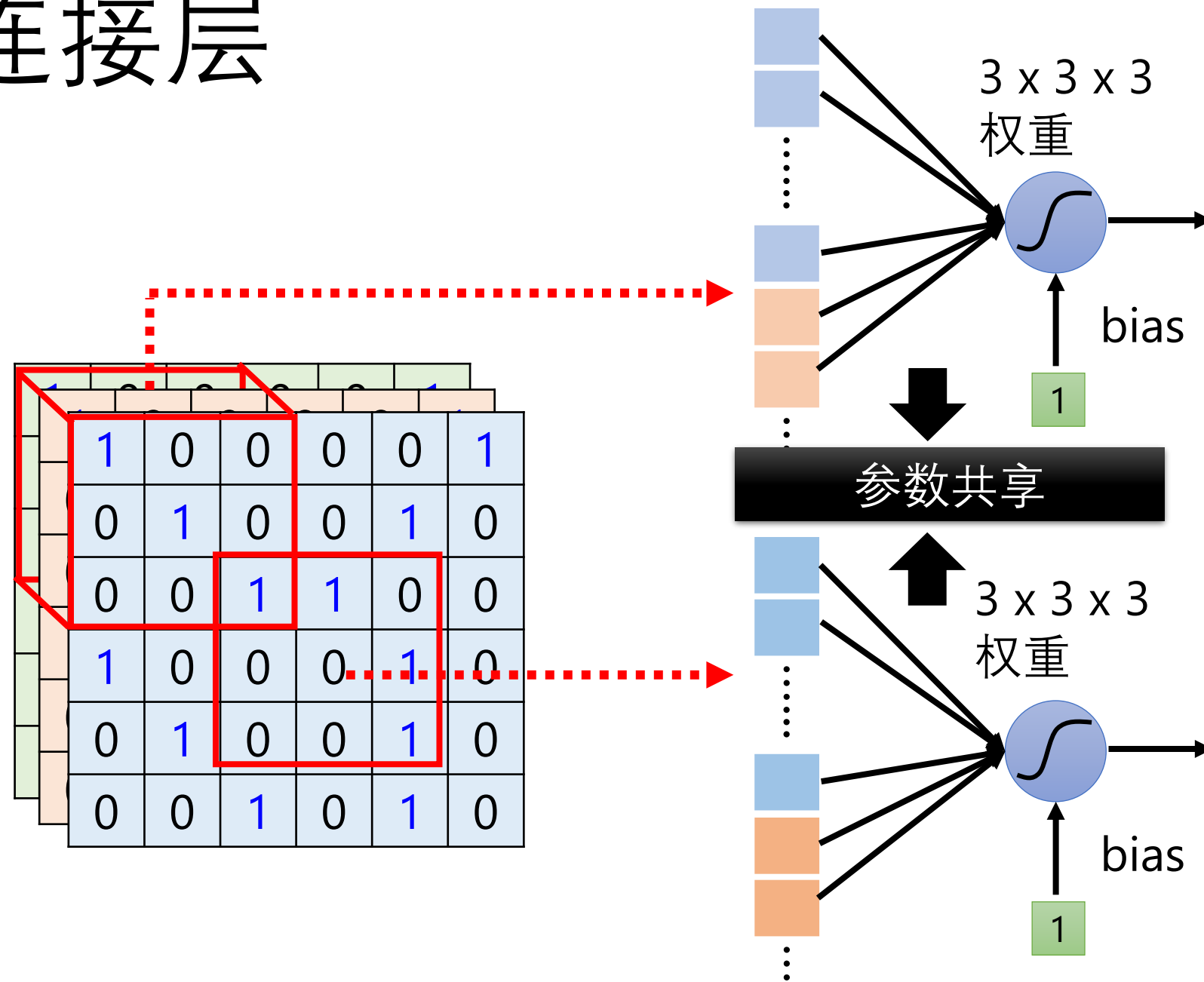


简化全连接层

- 相同的特征会出现在不同的区域.

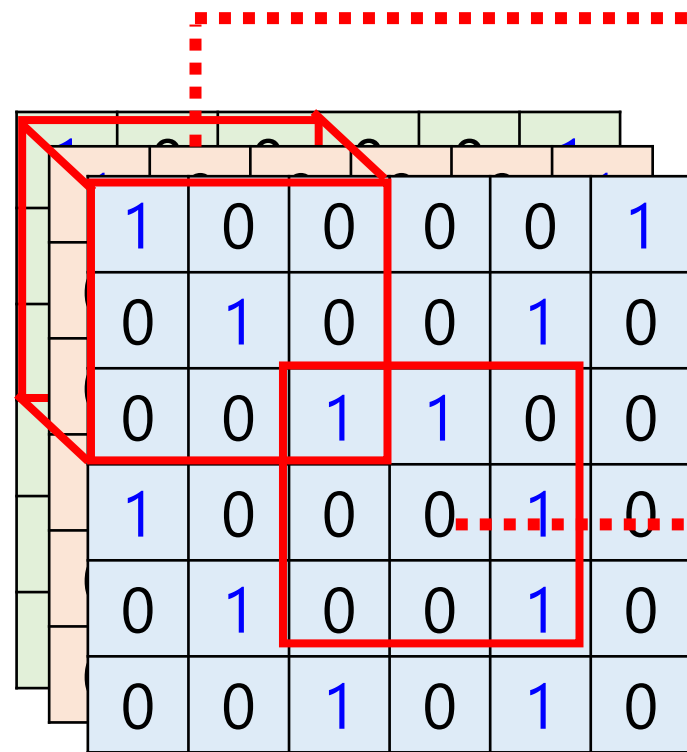


简化全连接层

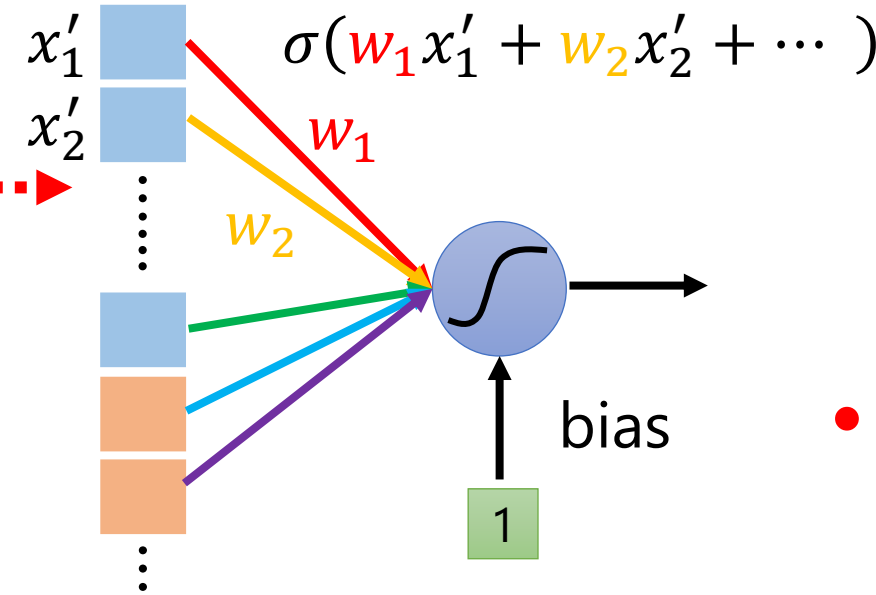
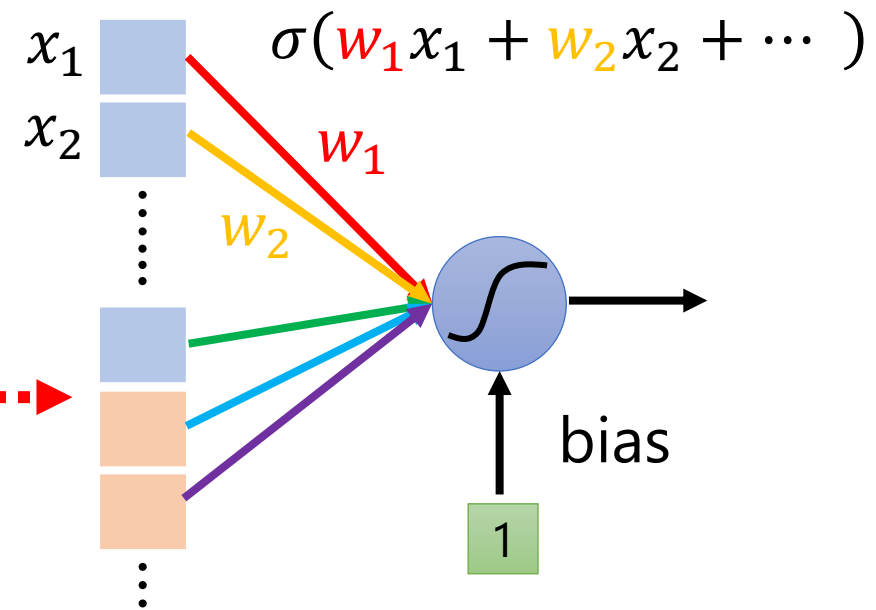


- 只需要一个 $3 \times 3 \times 3$ 的权重矩阵，就可以在整张图像上检测特征。通过参数共享，卷积神经网络可以大大减少所需的参数数量，同时确保相同的特征在不同的位置都能被检测到。

简化全连接层



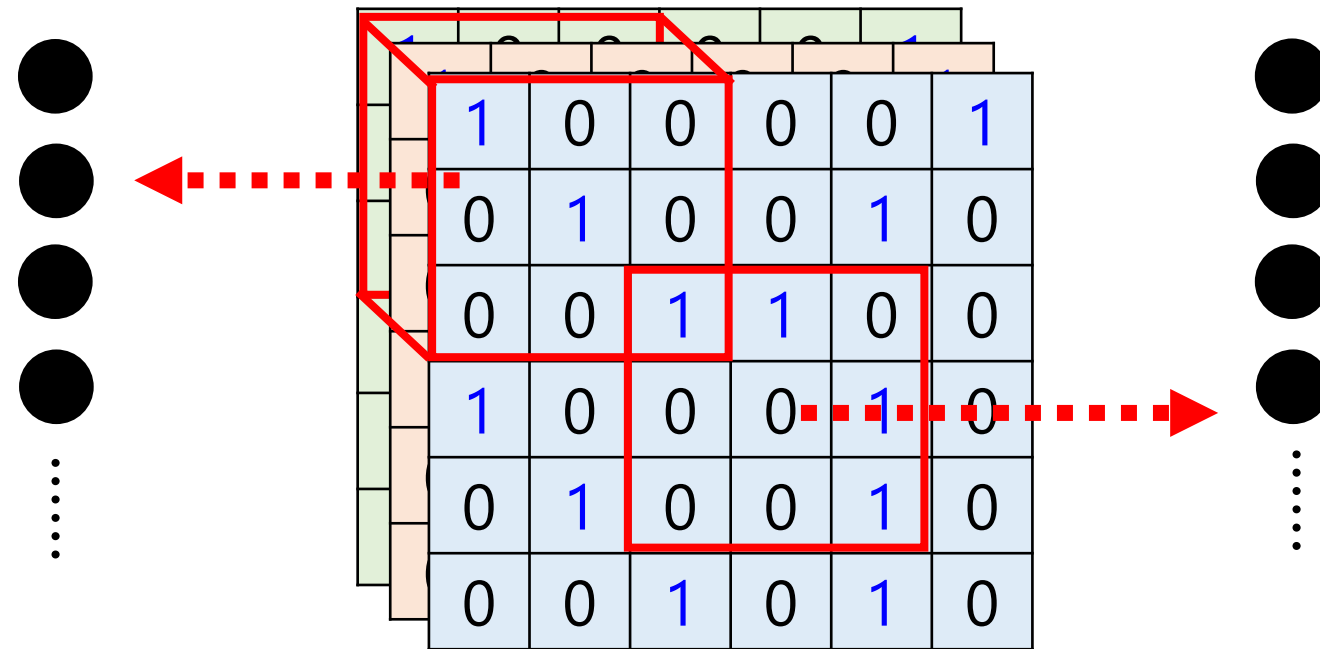
拥有相同感受野的神经元也可以不共享参数。



- 不同的神经元用于不同的特征检测。

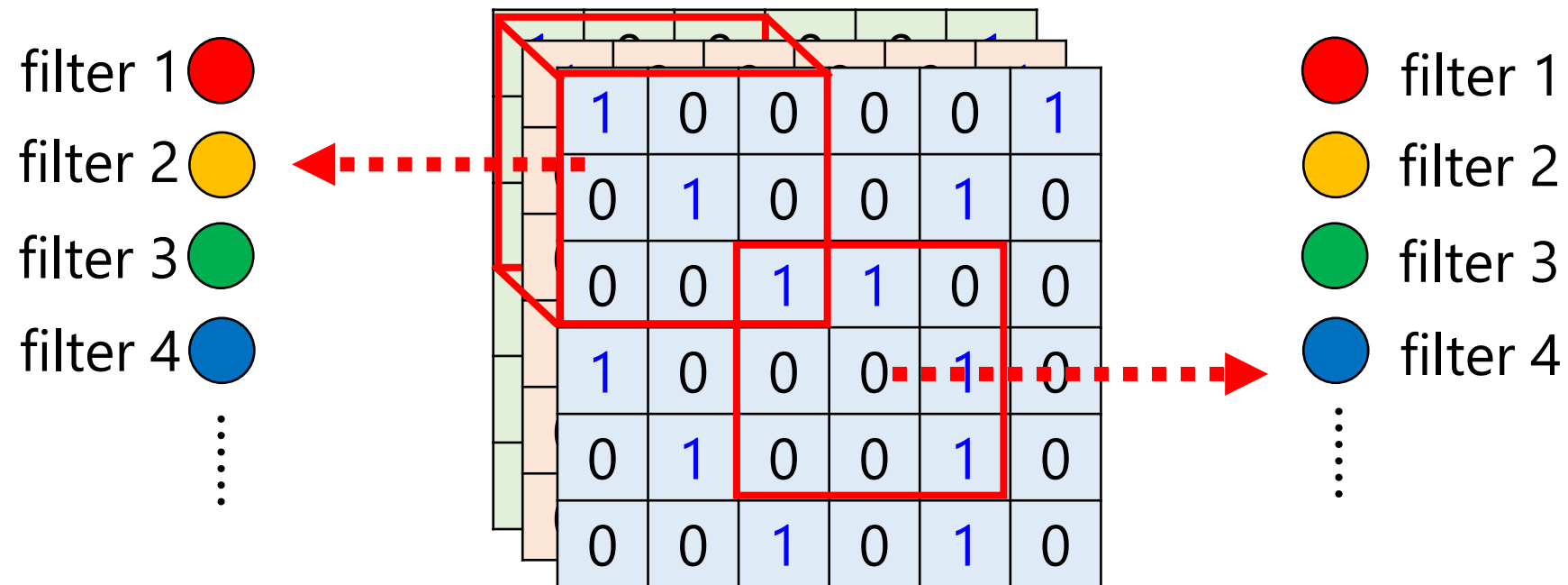
简化全连接层

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。

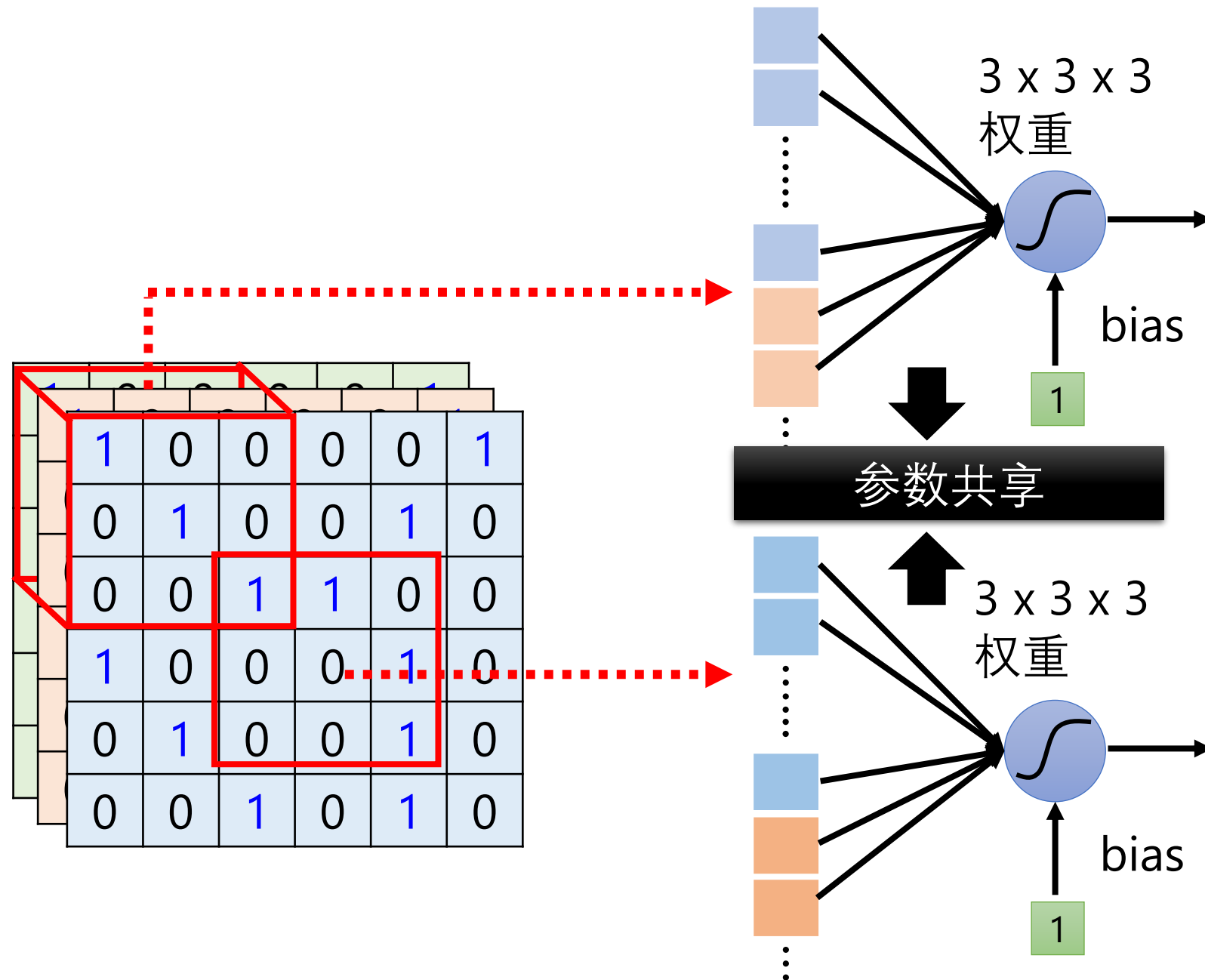


简化全连接层

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

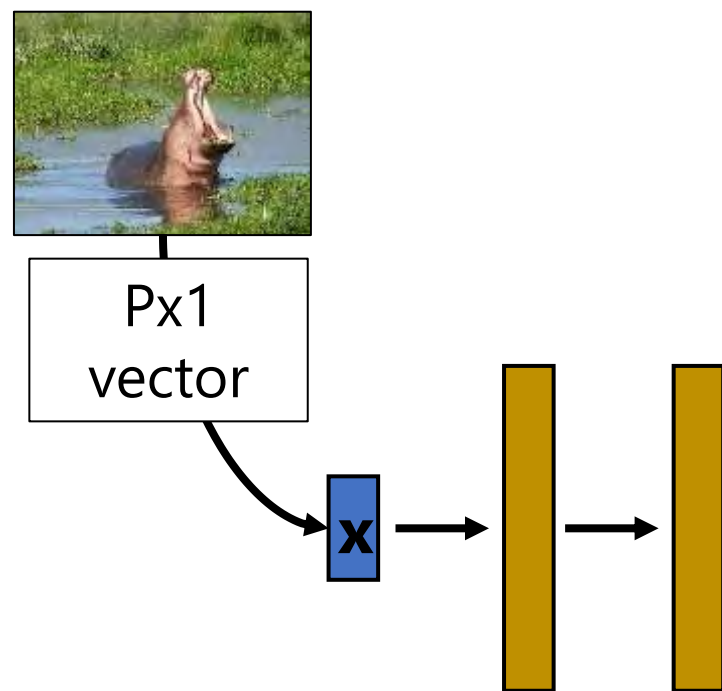


- 只需要一个3x3x3的权重矩阵，就可以在整张图像上检测特征。通过参数共享，卷积神经网络可以大大减少所需的参数数量，同时确保相同的特征在不同的位置都能被检测到。

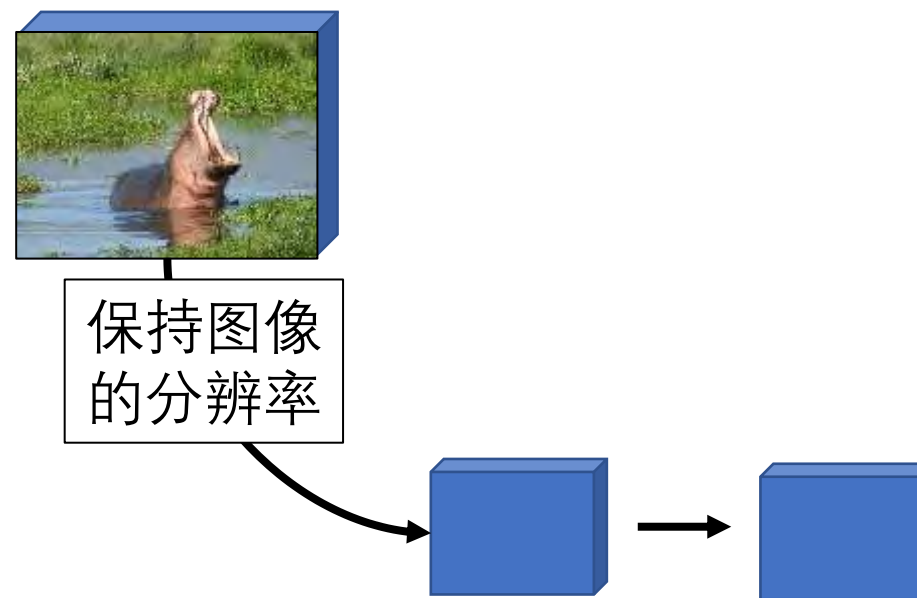
Convnets 卷积网络

不需要拉成向量， 可以保持图像的分辨率

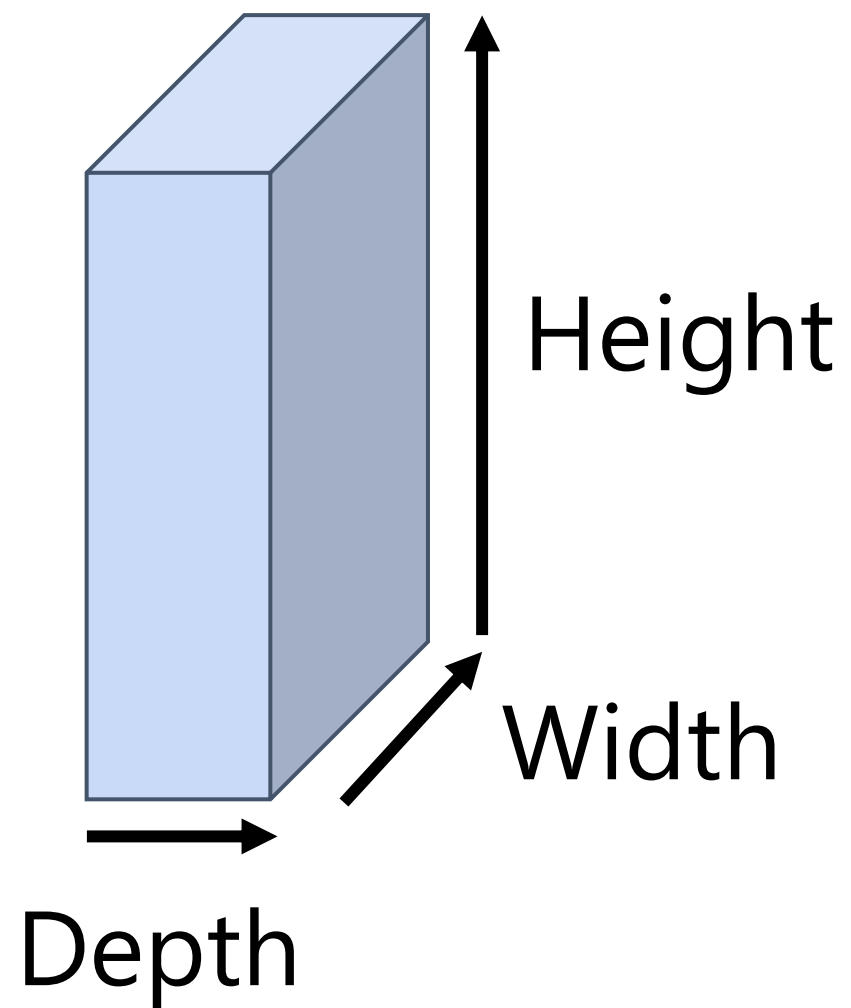
全连接网络:
数据: 向量 $F \times 1$
形式: 矩阵乘法



卷积网络:
数据: 图像 $H \times W \times F$
形式: 卷积



卷积网络



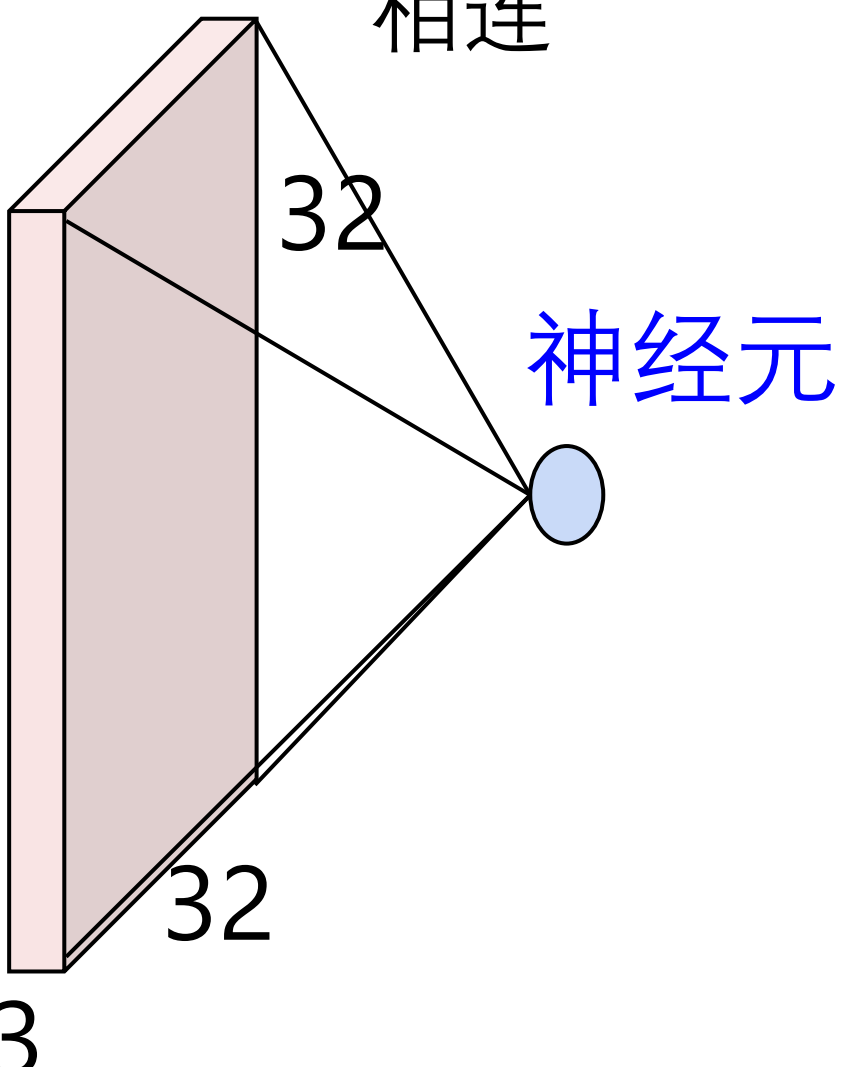
Height: 300
Width: 500
Depth: 3



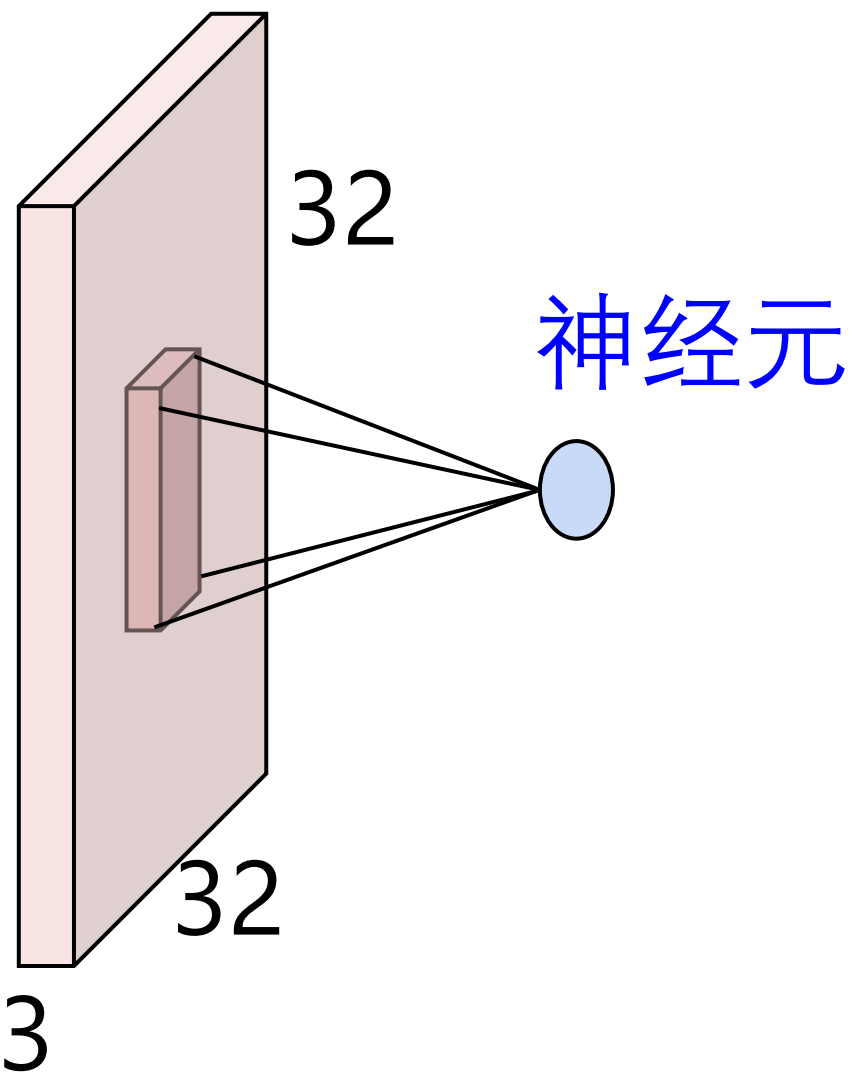
Height: 32
Width: 32
Depth: 3

卷积网络

全连接:
每个神经元与所有像素
相连



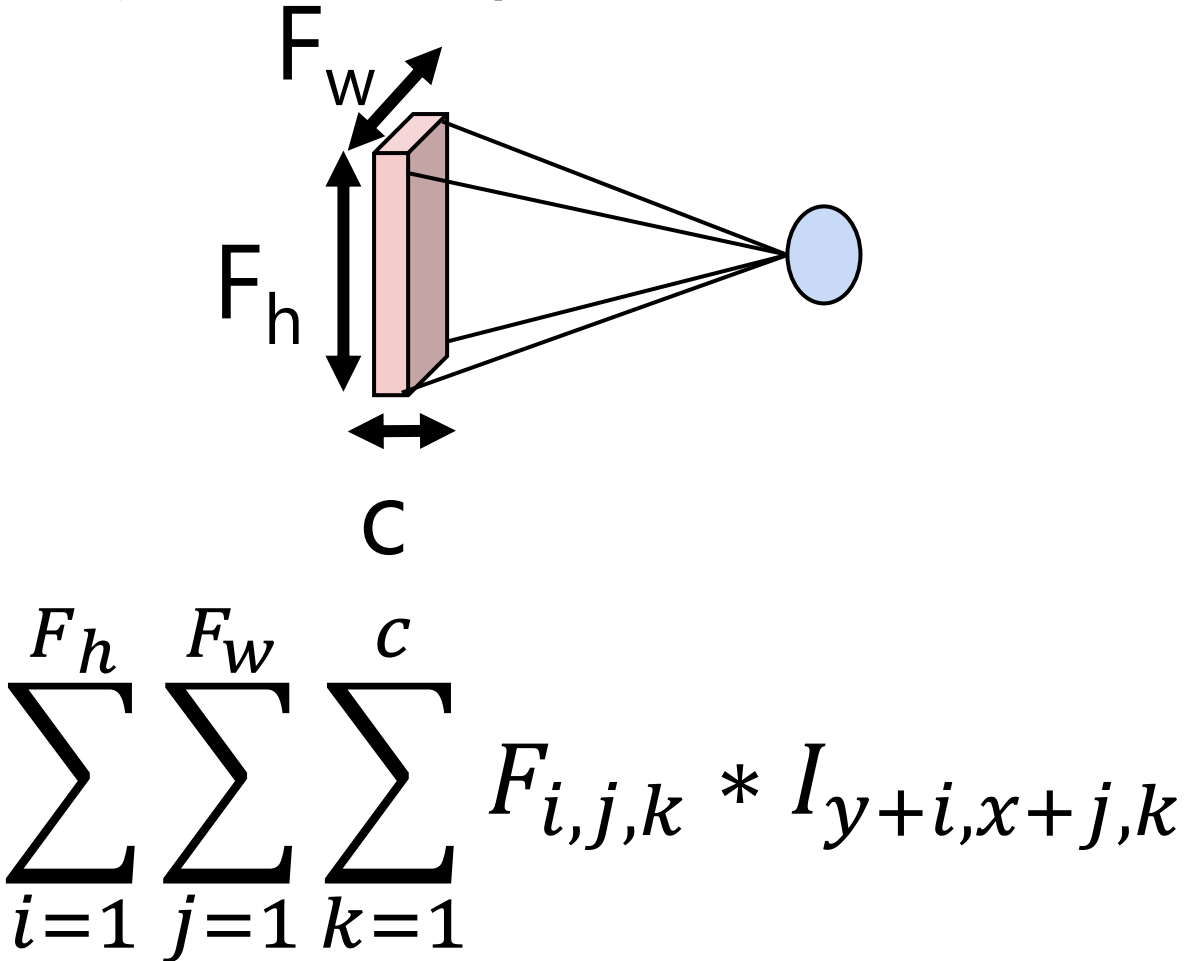
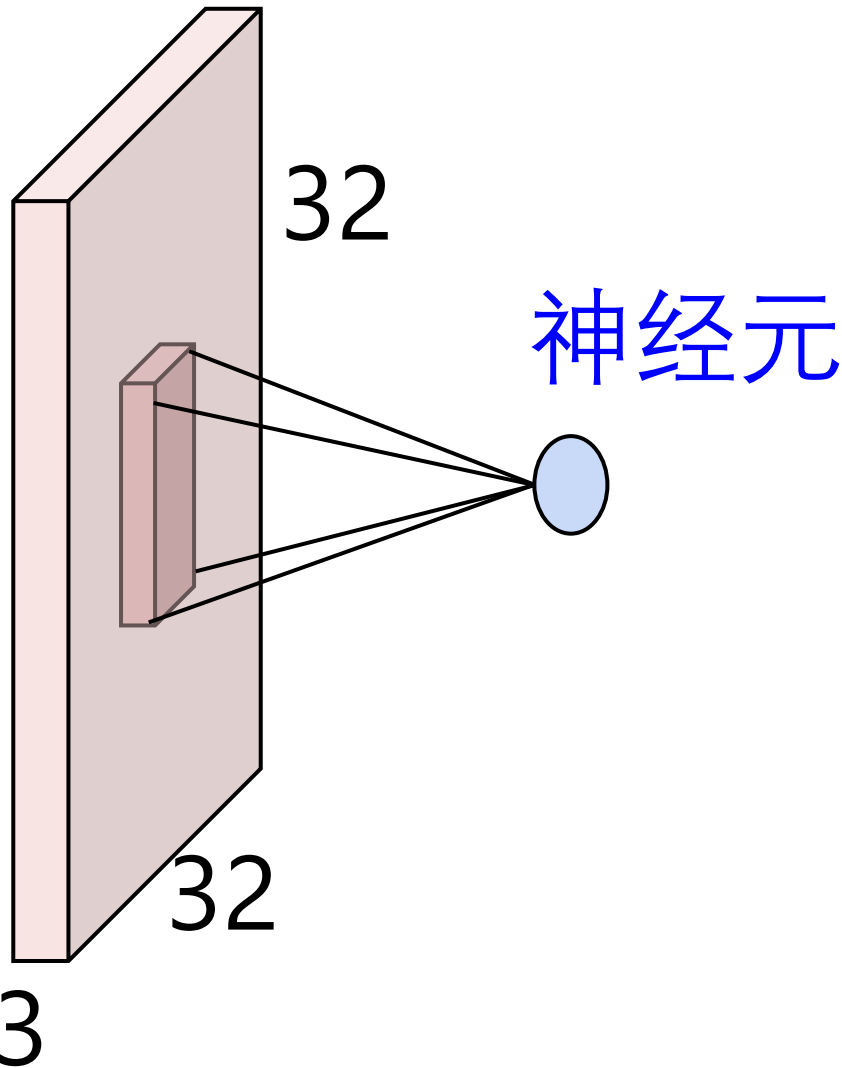
卷积网络:
只与局部相连



Slide credit: Karpathy and Fei-Fei

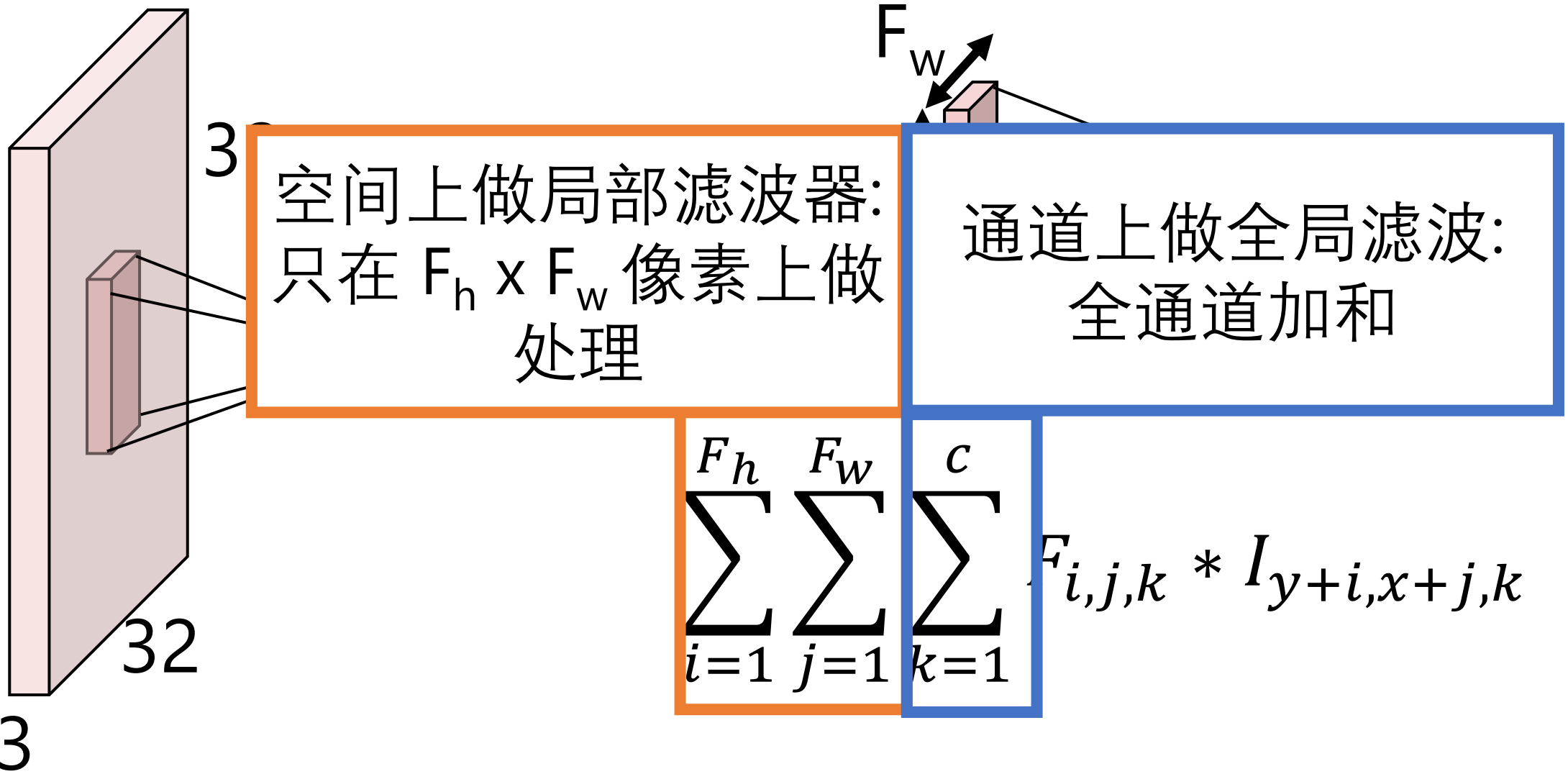
卷积网络

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



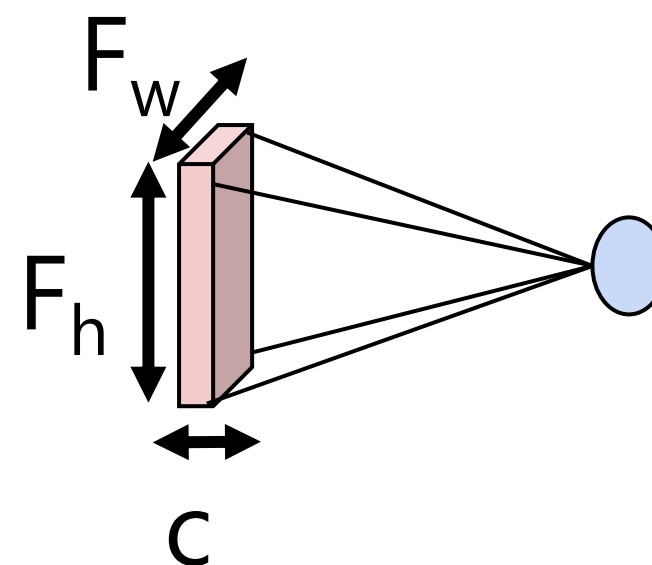
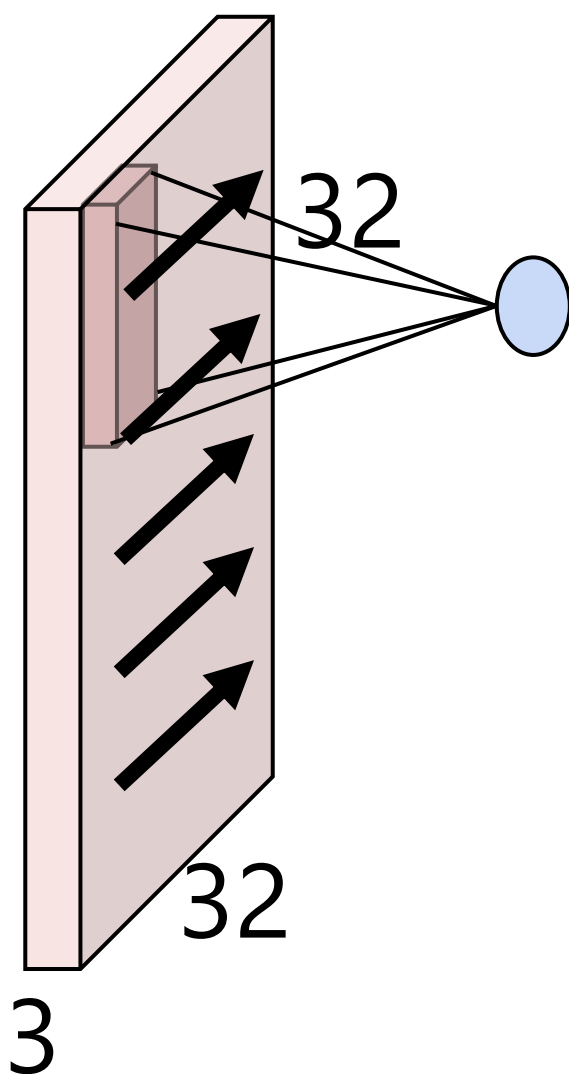
卷积网络

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



卷积网络

在图像上做滑动窗口卷积



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

与之前提到的滤波的区别

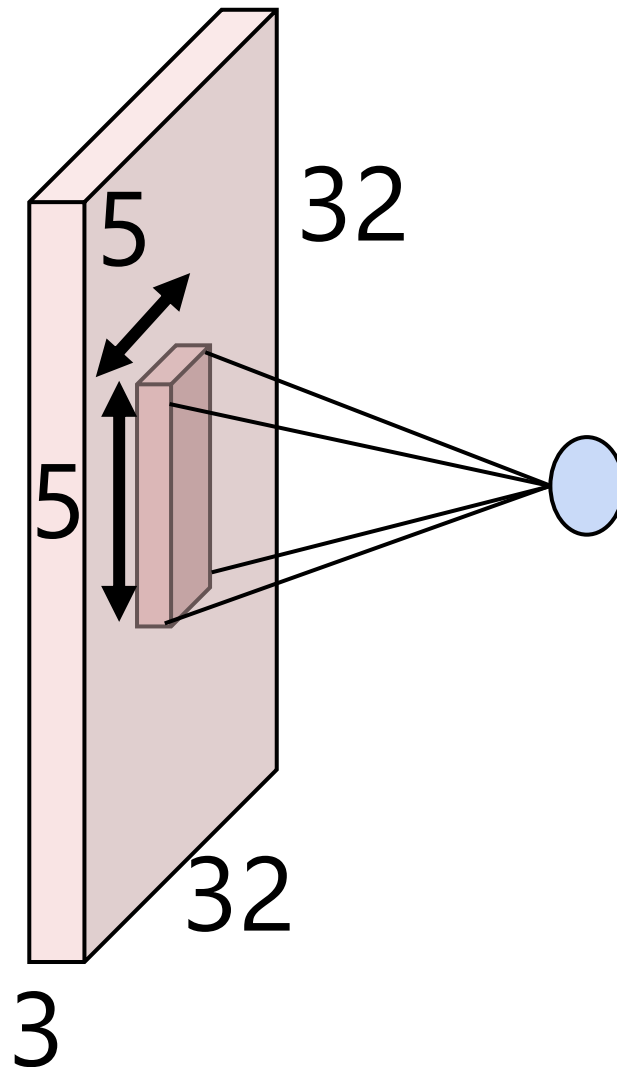
- (a) 输入的维度会比1或3更大
- (b) 忽略互相关核卷积的区别（实际操作中均使用矩阵乘法和互相关来实现卷积）

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

$$\begin{aligned} & \text{Output}[1,2] \\ &= I[1,2]*F[1,1] + I[1,3]*F[1,2] \\ & \quad + \dots + I[3,4]*F[3,3] \end{aligned}$$

卷积网络

输出有多大?



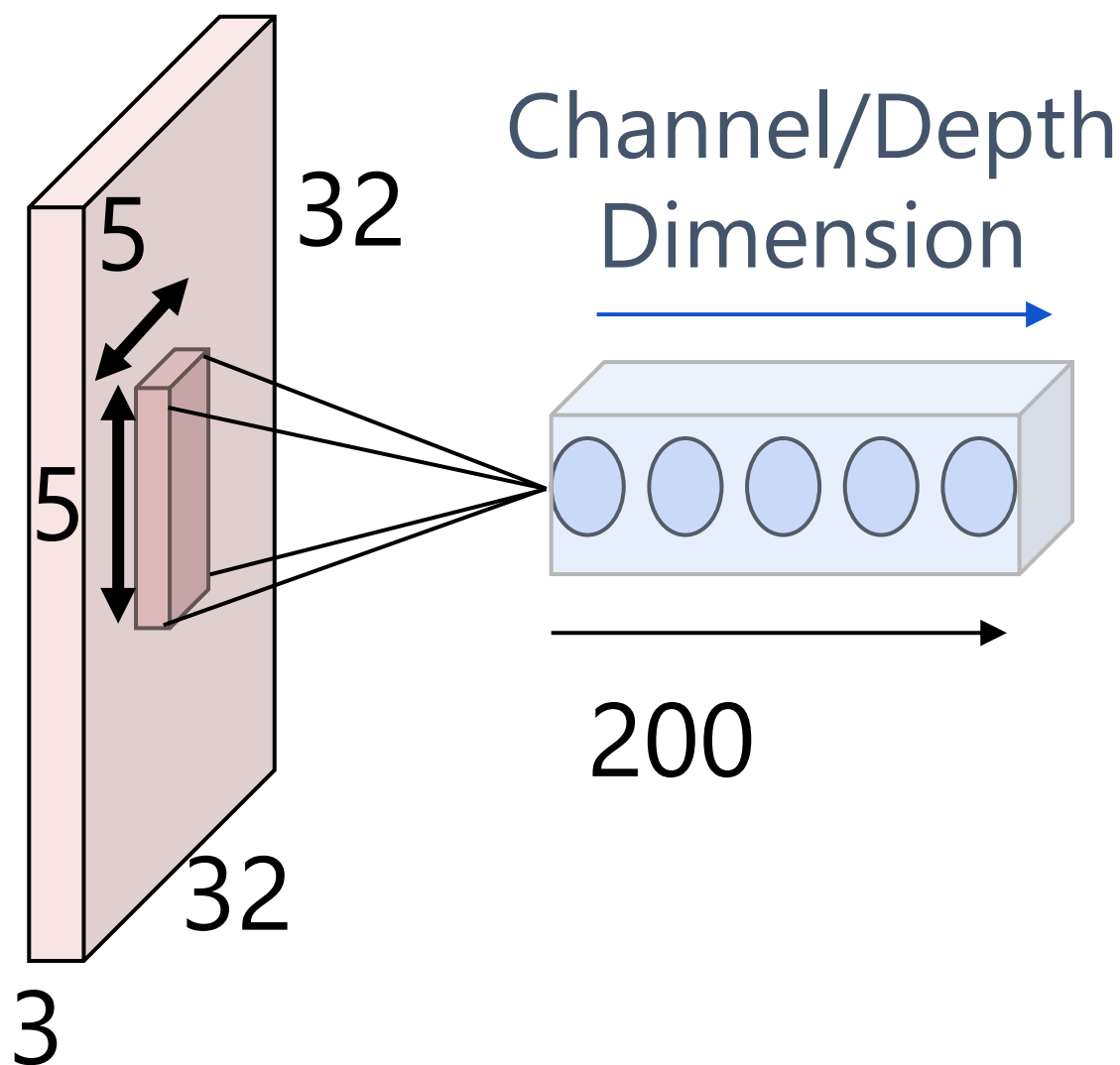
Height? $32 - 5 + 1 = 28$

Width? $32 - 5 + 1 = 28$

Channels? 1

卷积网络

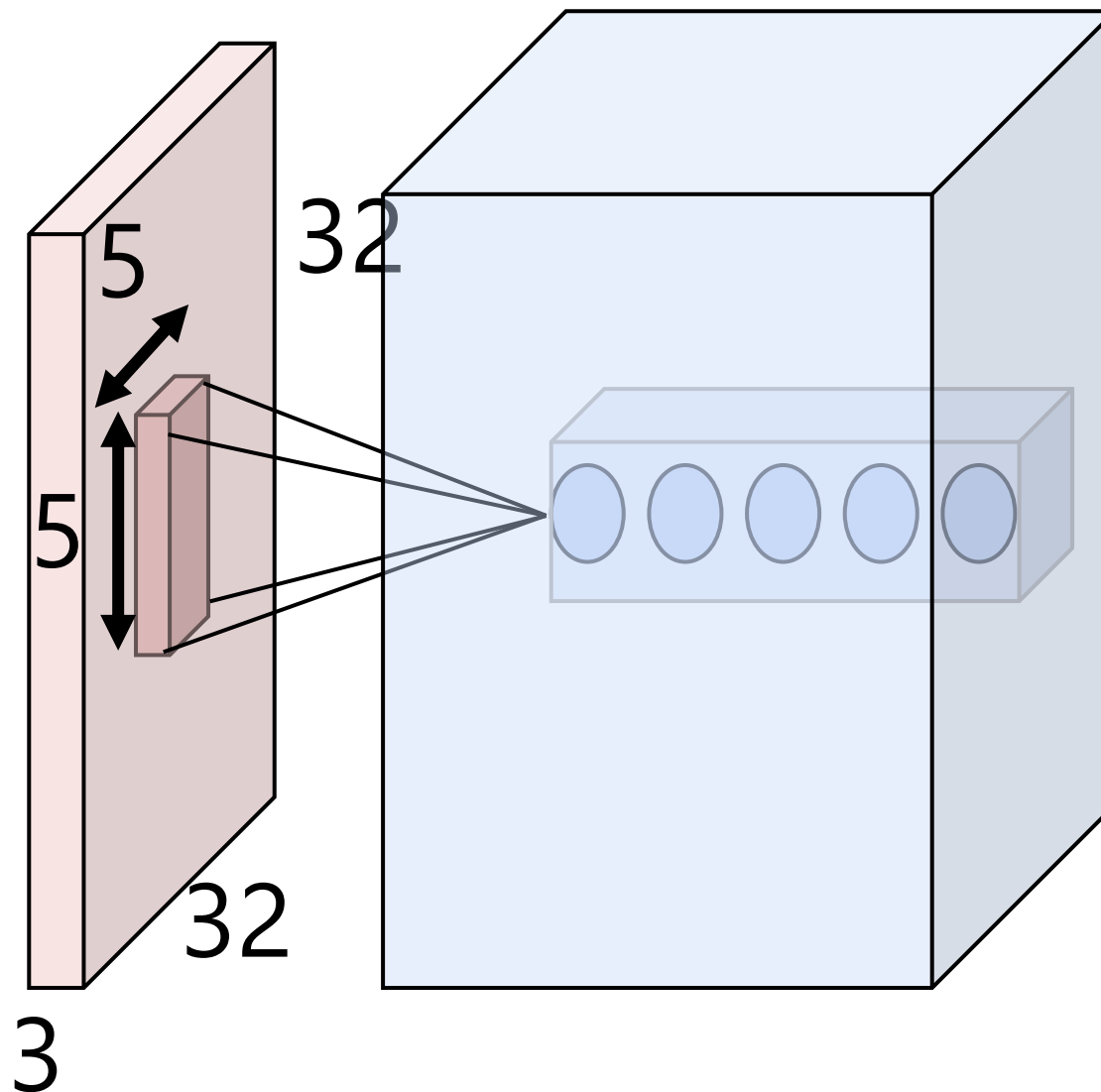
多个channel 通道对应每个滤波器.
现在输出会有多大?



Height? $32 - 5 + 1 = 28$
Width? $32 - 5 + 1 = 28$
Channels? 200

卷积网络

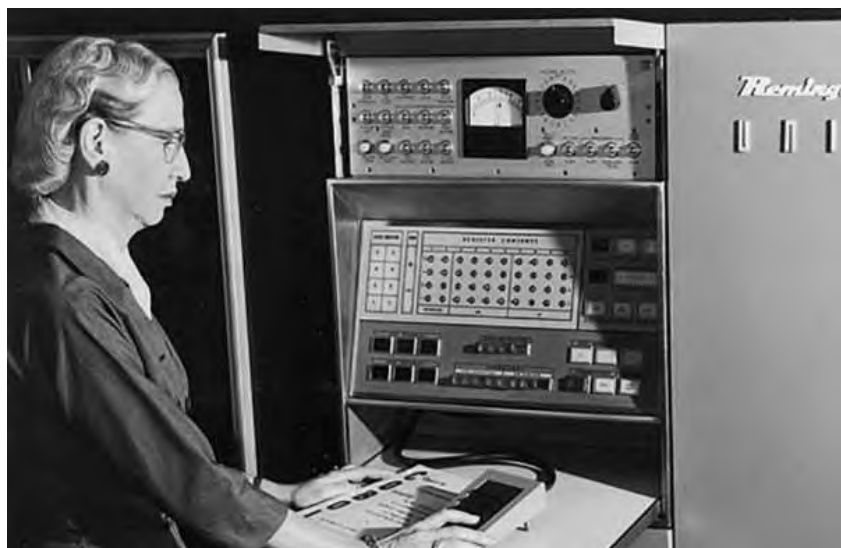
多个channel 通道对应每个滤波器.
现在输出会有多大?



Height? $32 - 5 + 1 = 28$
Width? $32 - 5 + 1 = 28$
Channels? 200

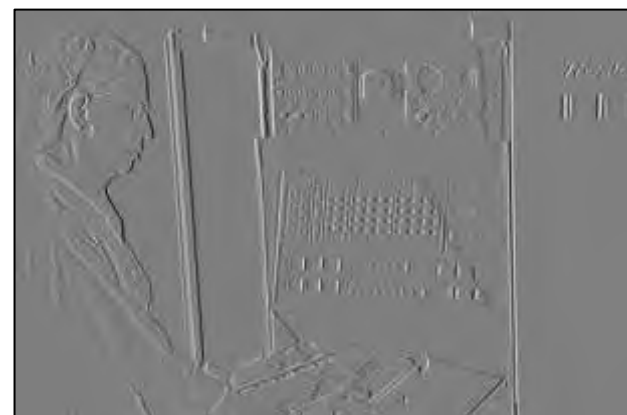
多个神经元的组合

Input: $400 \times 600 \times 1$



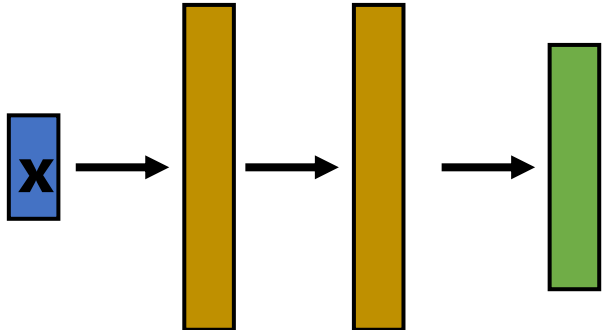
Recall: 图像梯度 怎样用卷积层实现? 用多少参数?

Output:
 $400 \times 600 \times 2$

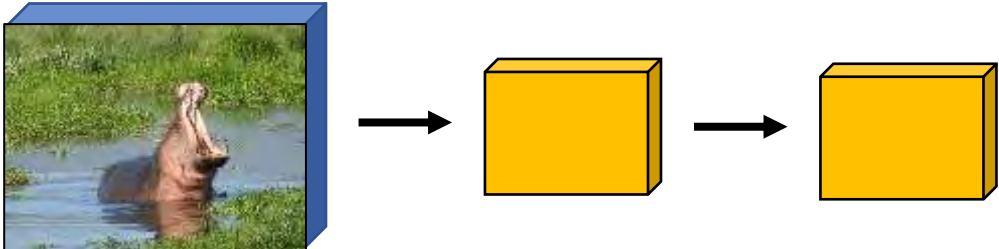


卷积网络 vs 全连接网络

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

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

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

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

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

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:

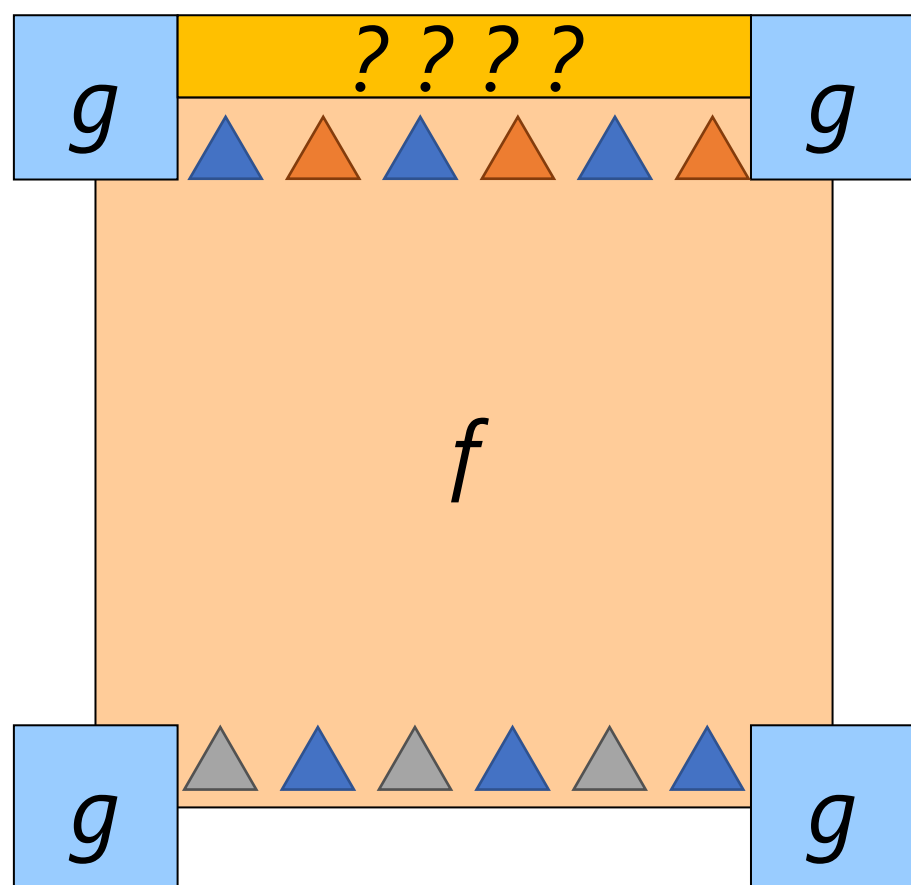
3x3 output

Stride 3 convolution:

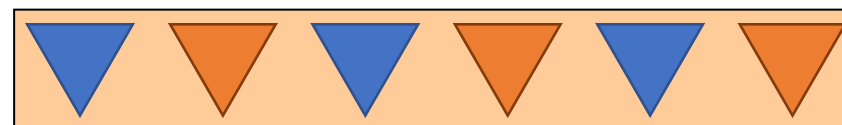
有问题! 为什么?

Padding

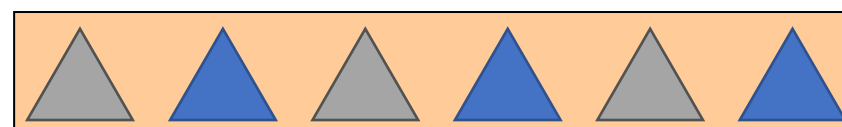
Padding用于填充外围缺失区域，形式有很多种



Symm: fold sides over



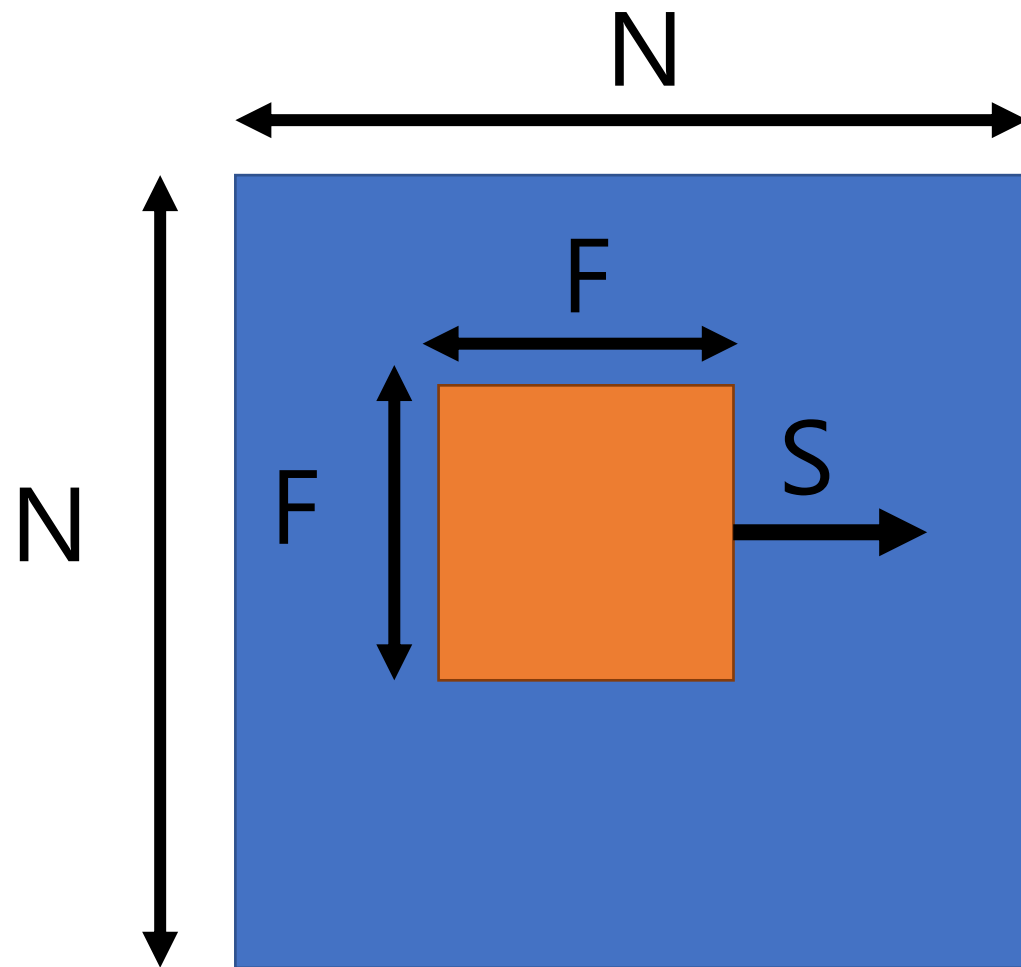
Circular/Wrap: wrap around



pad/fill: add value, often 0



总的来说...



Output Size

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

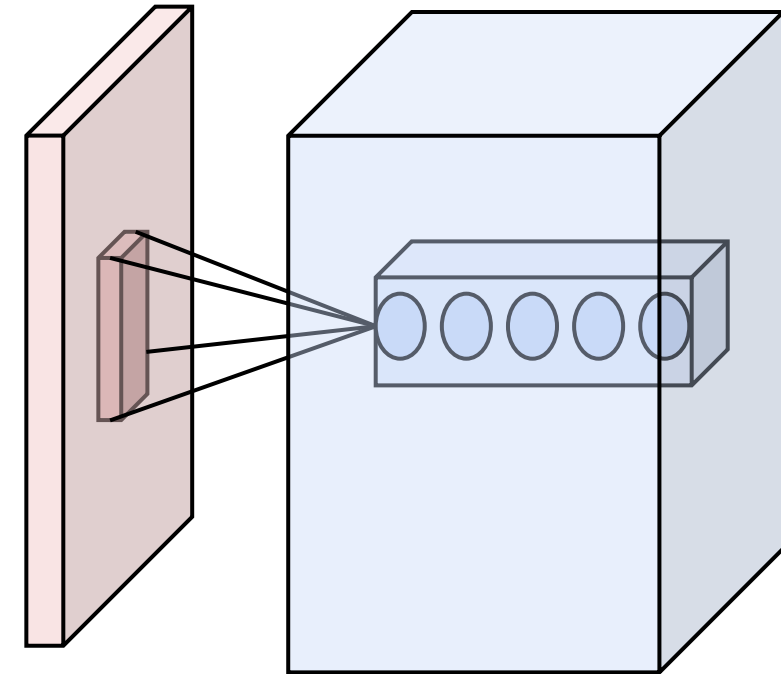
计算题

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?

计算题

输入大小: **32x32x3**

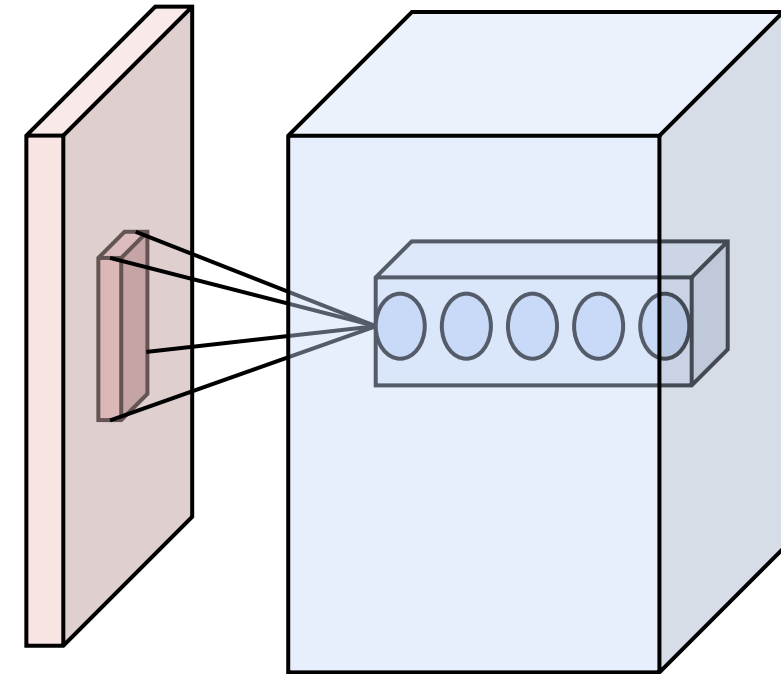
感受野/卷积核大小: **5x5, stride 1**

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?



计算题

输入大小: **32x32x3**

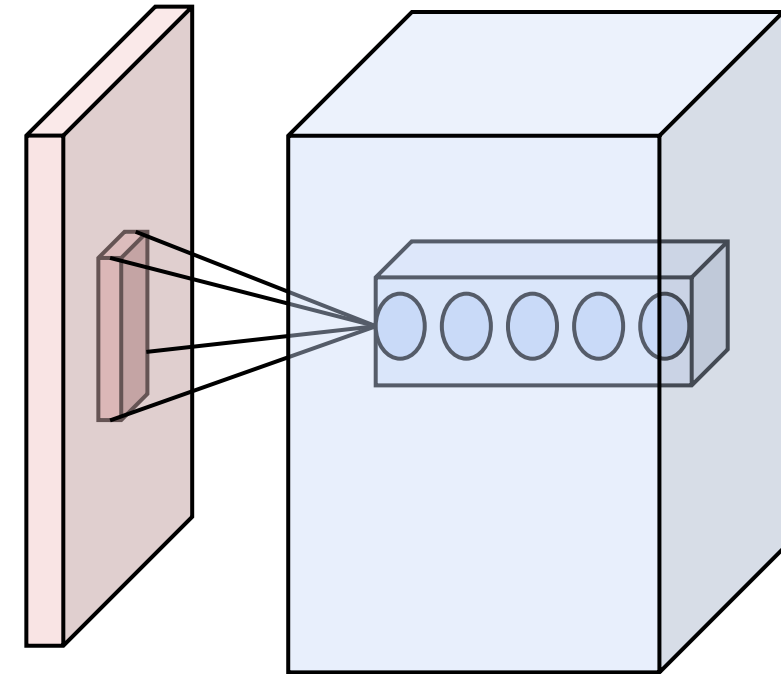
感受野/卷积核大小: **5x5, stride 1**

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**



计算题

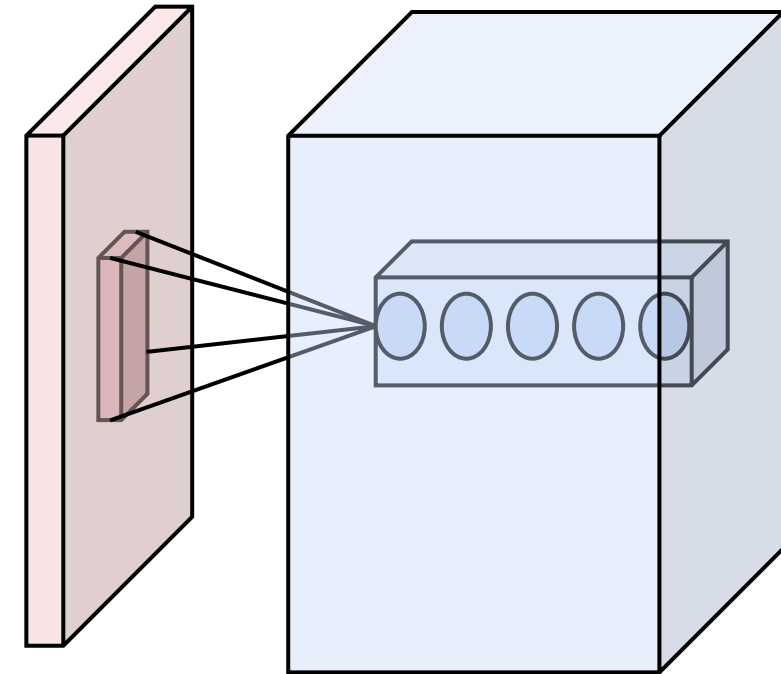
输入大小: **32x32x3**

感受野/卷积核大小: **5x5, stride 3**

神经元数量: **5**

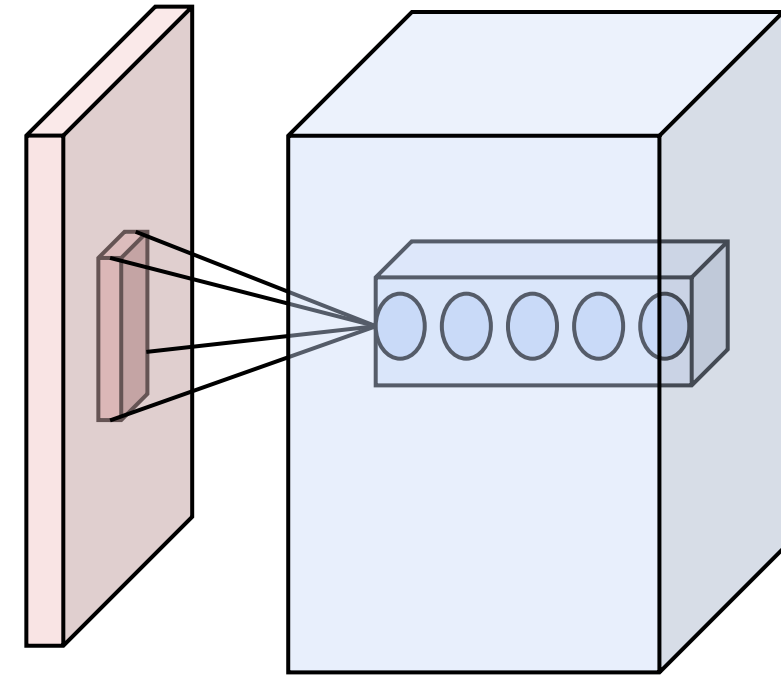
$$\frac{(N - F)}{s} + 1$$

输出大小?



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**

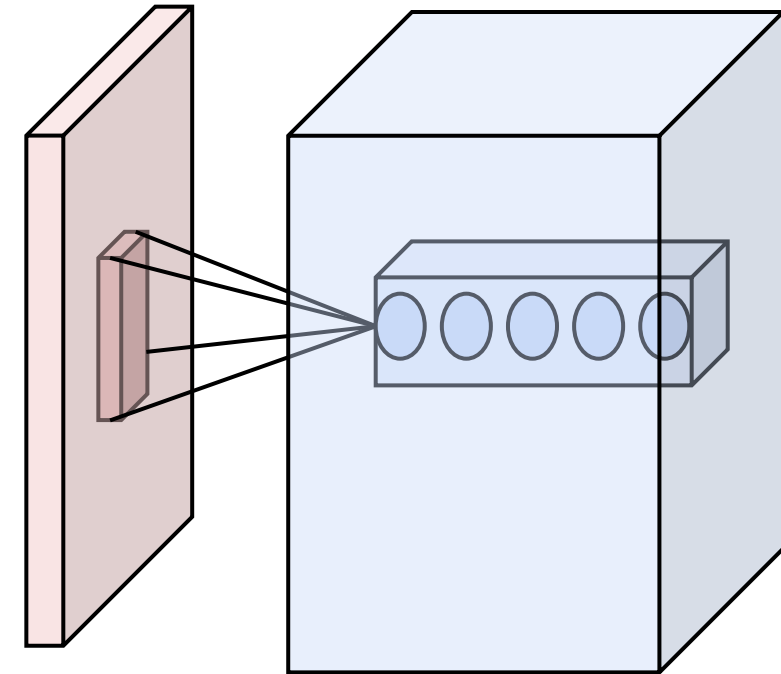
参数量?

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**

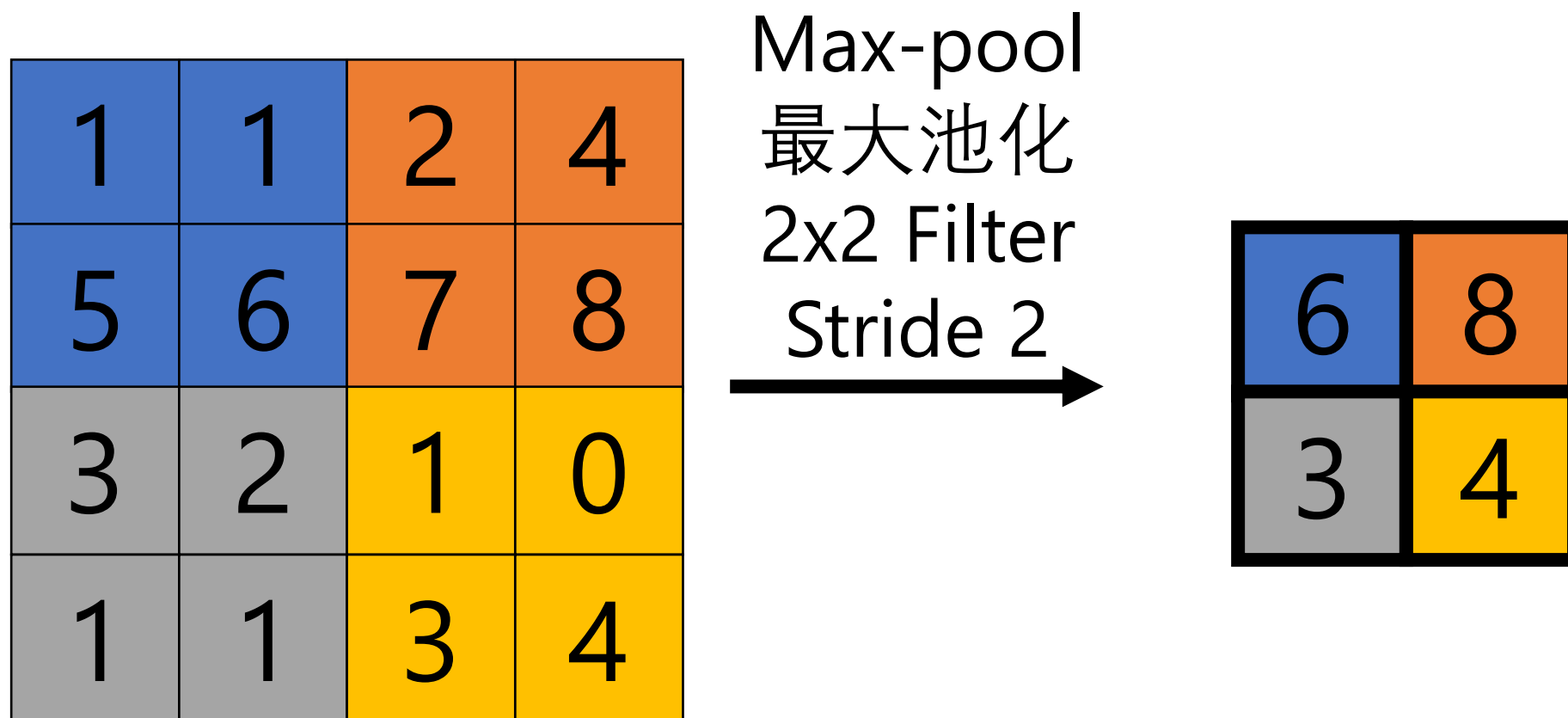


输出尺寸: $(32 - 5) / 3 + 1 = 10$, 输出大小: **10x10x5**

参数量: **5x5x3x5 + 5 = 380**. 与之前相同!

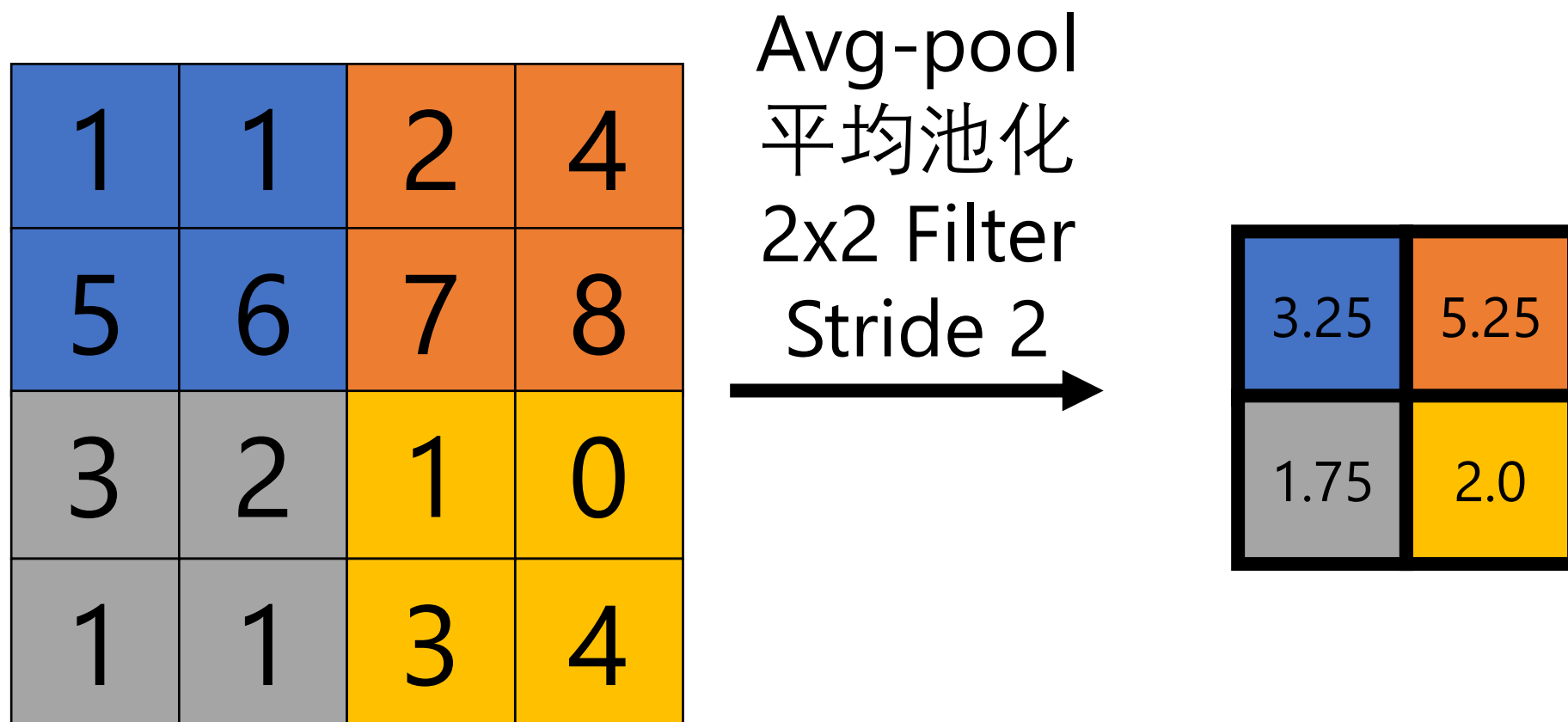
Pooling “池化”

出发点：使得特征的空间分辨率变小；
在每个通道上应用



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



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

I11	I12	I13	I14	I15	I16	I17
I21	I22	I23	I24	I25	I26	I27
I31	I32	I33	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

Max-pool
3x3 Filter
Stride 2



O11

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

I11	I12	I13	I14	I15	I16	I17
I21	I22	I23	I24	I25	I26	I27
I31	I32	I33	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

Max-pool
3x3 Filter
Stride 2



O11	O12
-----	-----

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

I11	I12	I13	I14	I15	I16	I17
I21	I22	I23	I24	I25	I26	I27
I31	I32	I33	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

Max-pool
3x3 Filter
Stride 2

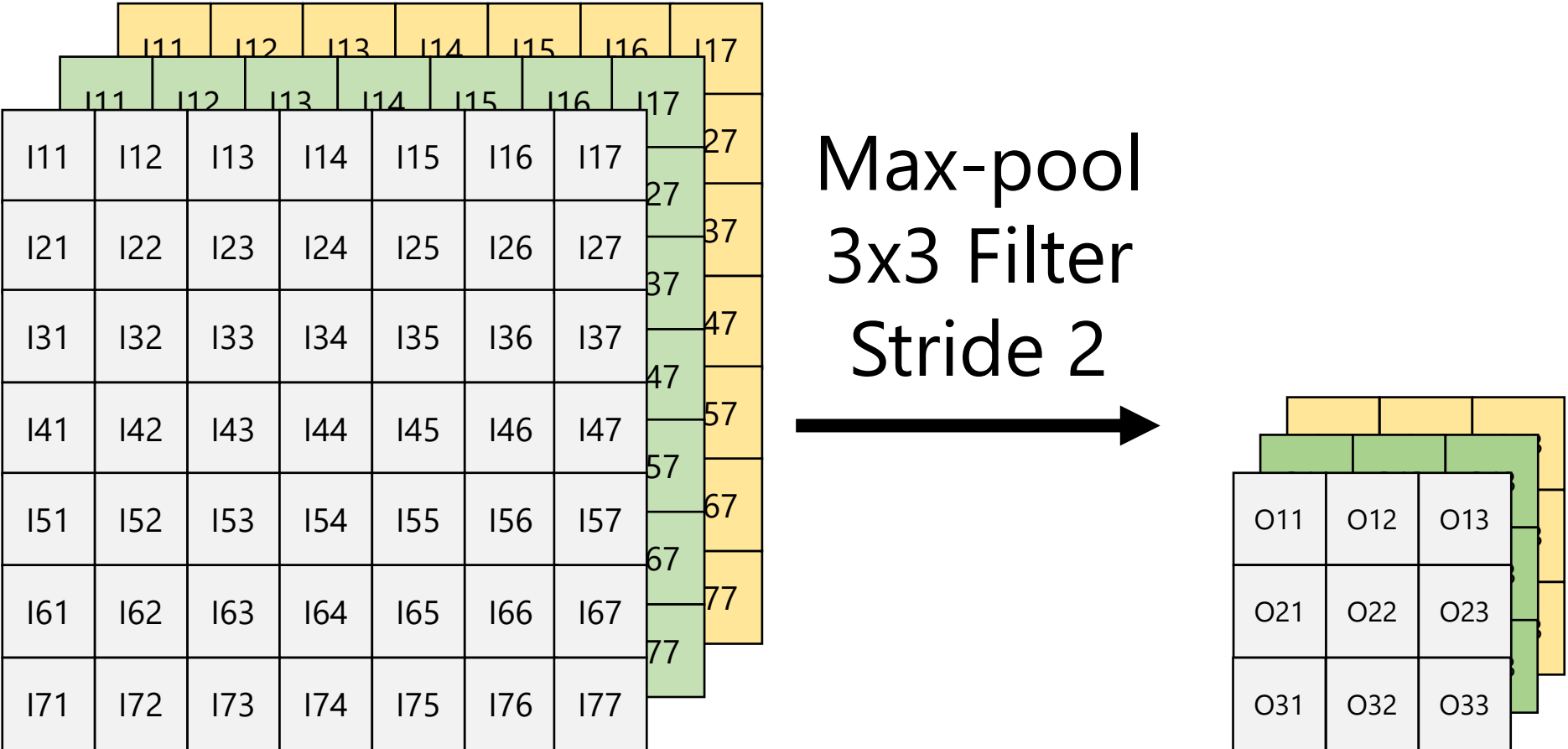


O11	O12	O13
-----	-----	-----

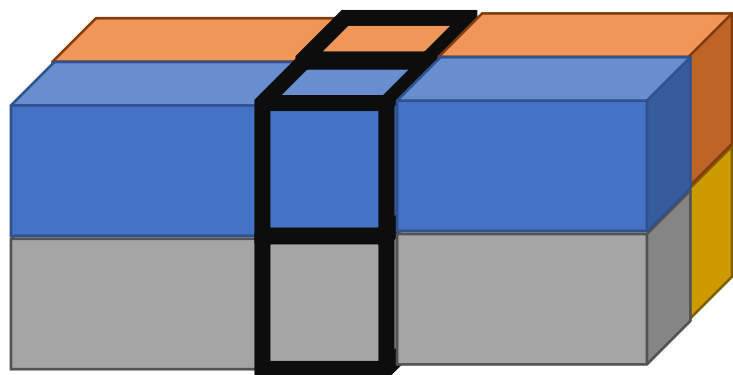
O13 = 蓝框里的最大值

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



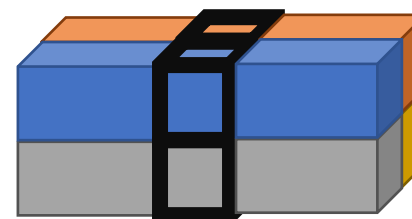
Pooling 前后对比



1	1	2	4
5	6	7	8
3	2	1	0
1	1	3	4

Max-pool
2x2 Filter
Stride 2

6	8
3	4



回顾——卷积神经网络

参数量



$P \times 1$
vector

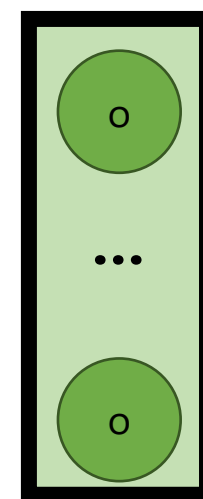
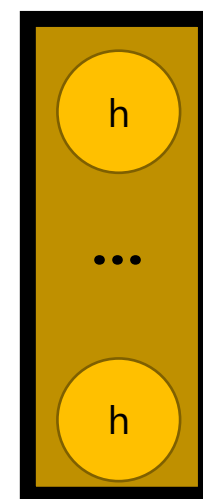
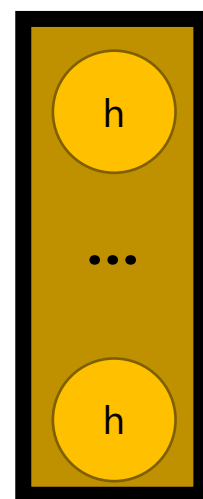
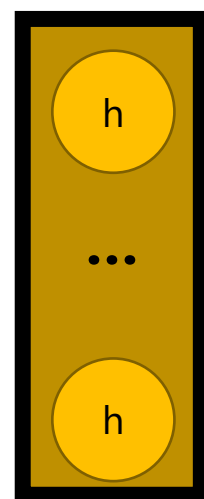
x

H^*P+
 H
 H
neurons

H^*H
 $+H$
 H
neurons

H^*H
 $+H$
 H
neurons

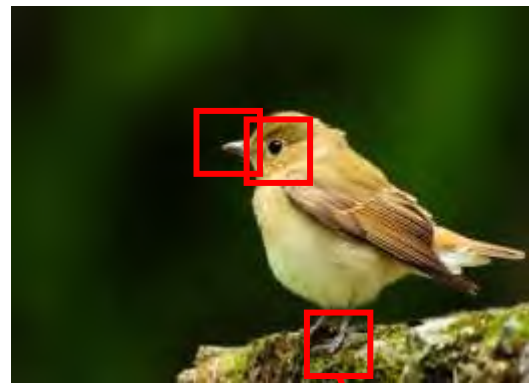
O^*H
 $+O$
 O
neurons



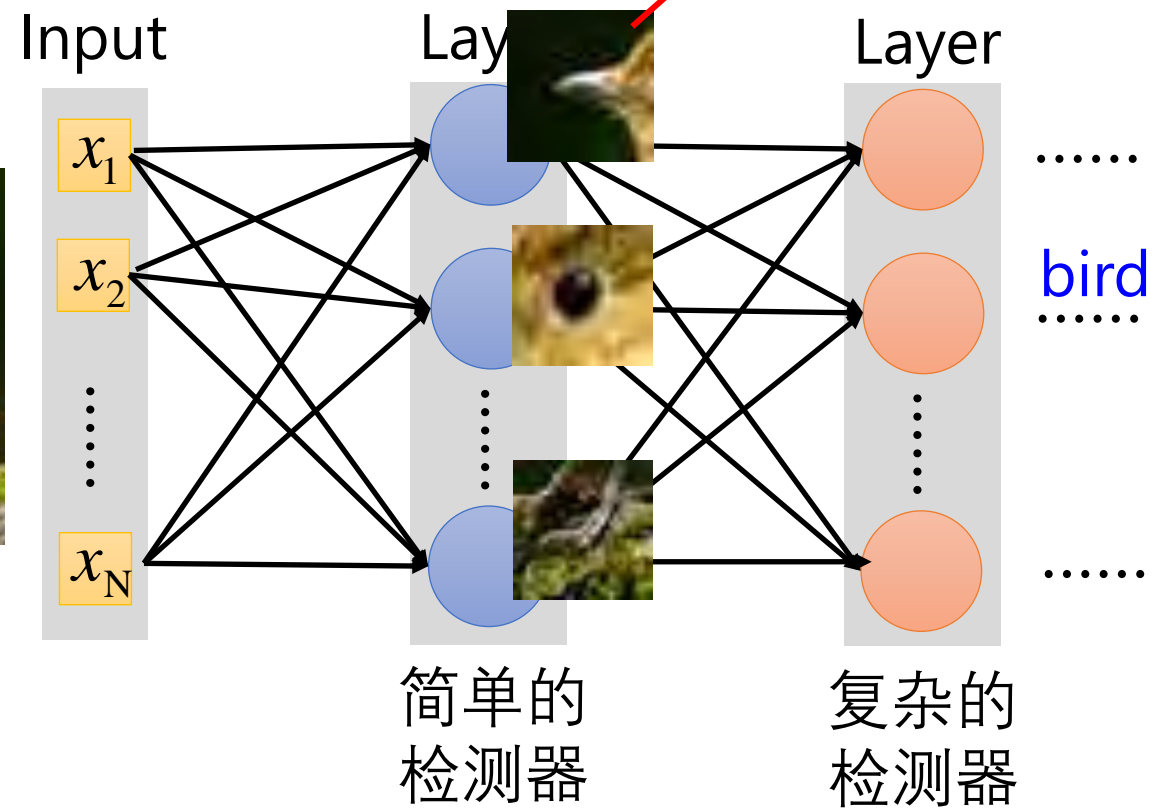
$P: 285 \times 350$, $H: 1000$, $O: 3$
102 million 个参数 (400MB)

简化全连接层

我们需要看整张图来分类吗？
(回顾特征点)



每个神经元并不需要看到所有像素点 (回顾特征尺度)

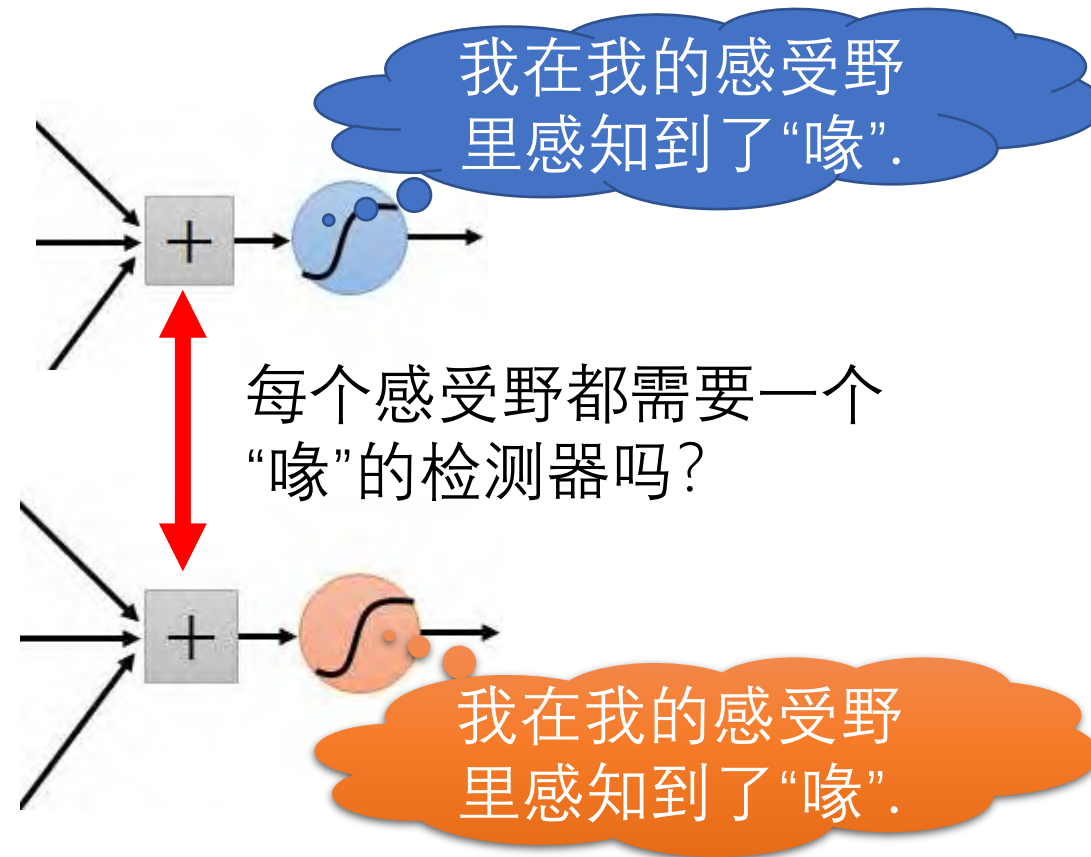
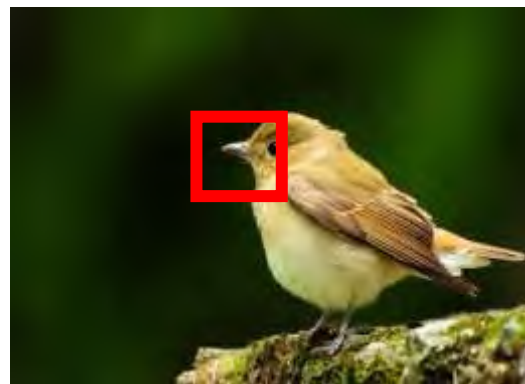
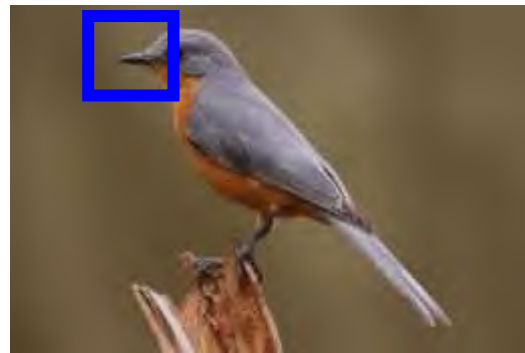


有些关键特征也许很小 (回顾特征尺度)

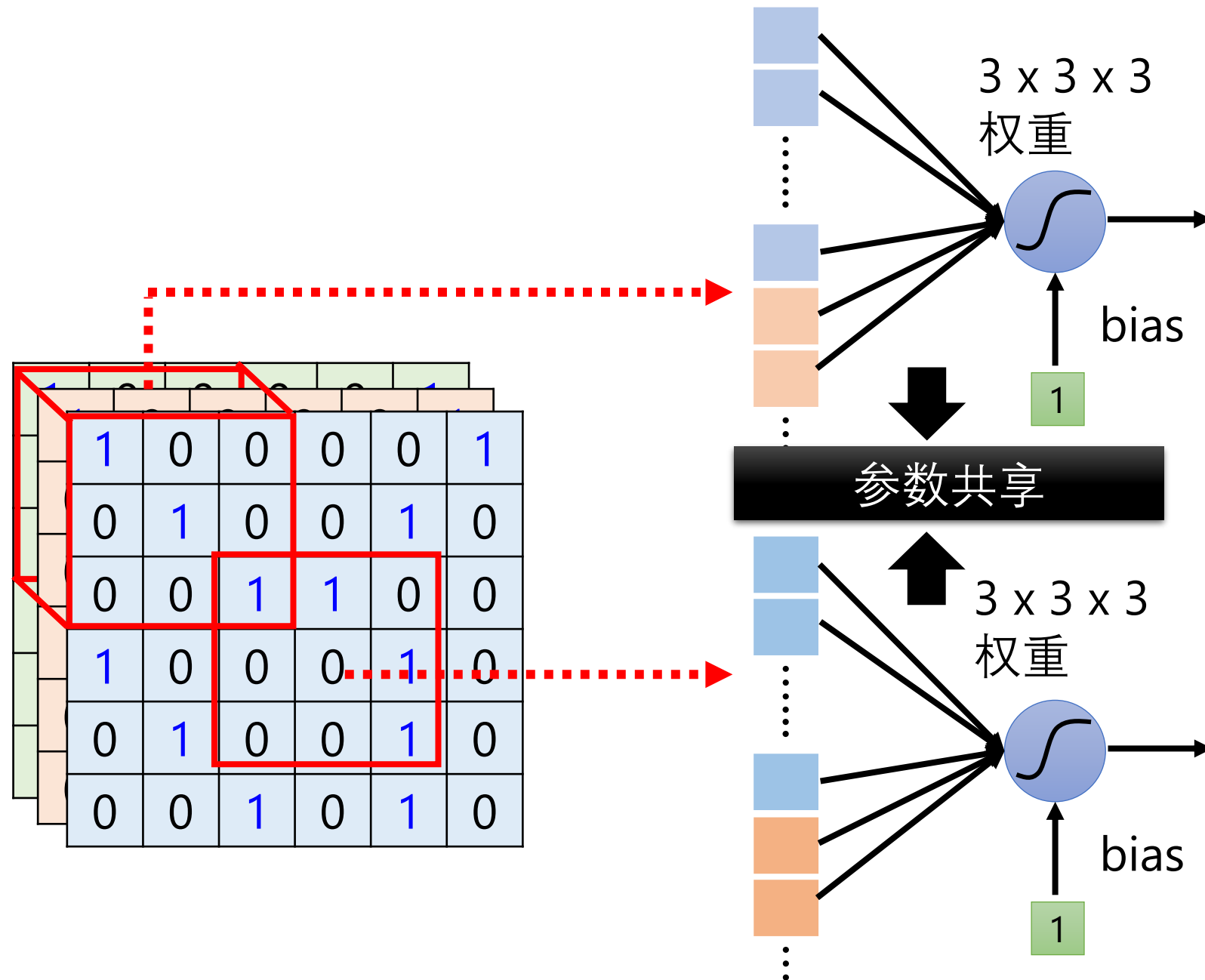
神经元通常只对图像中的局部模式或特征进行响应，而不是整张图像。

简化全连接层

- 相同的特征会出现在不同的区域.



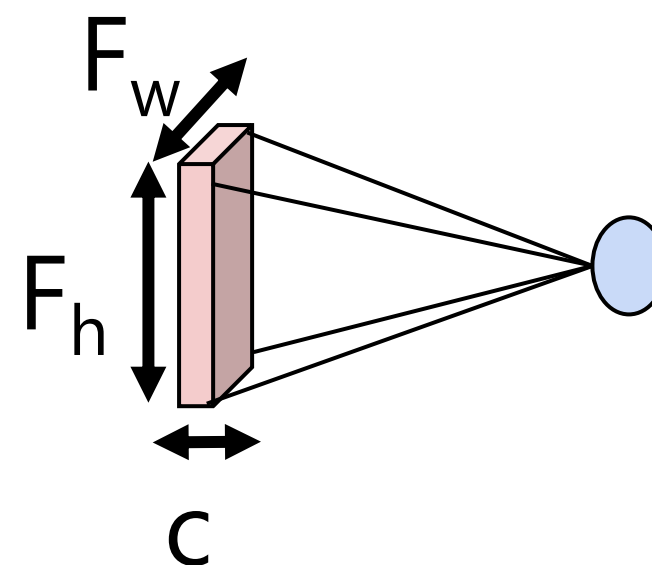
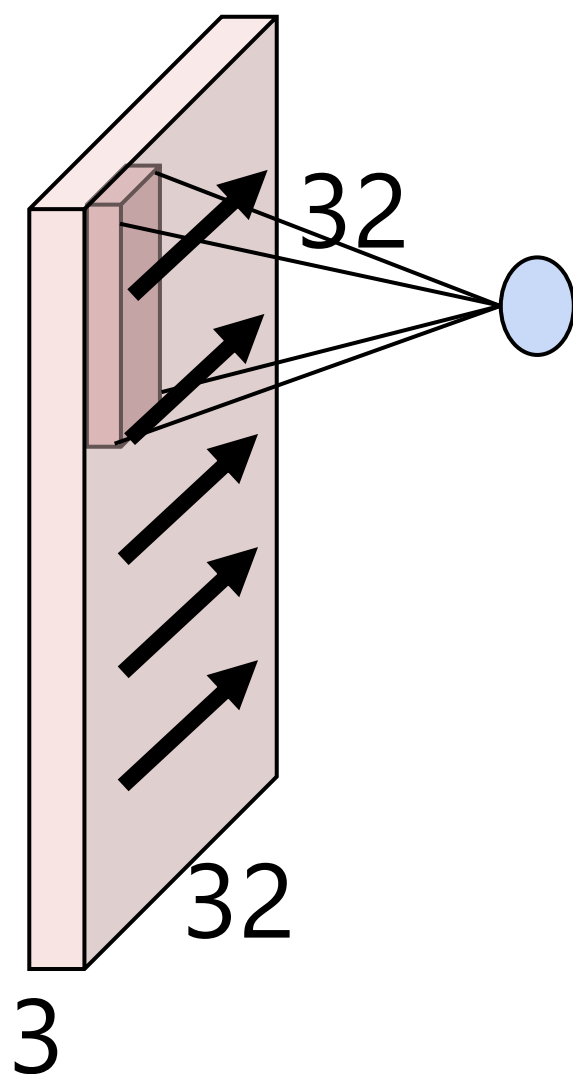
简化全连接层的关键——参数共享



- 只需要一个 $3 \times 3 \times 3$ 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



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

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):

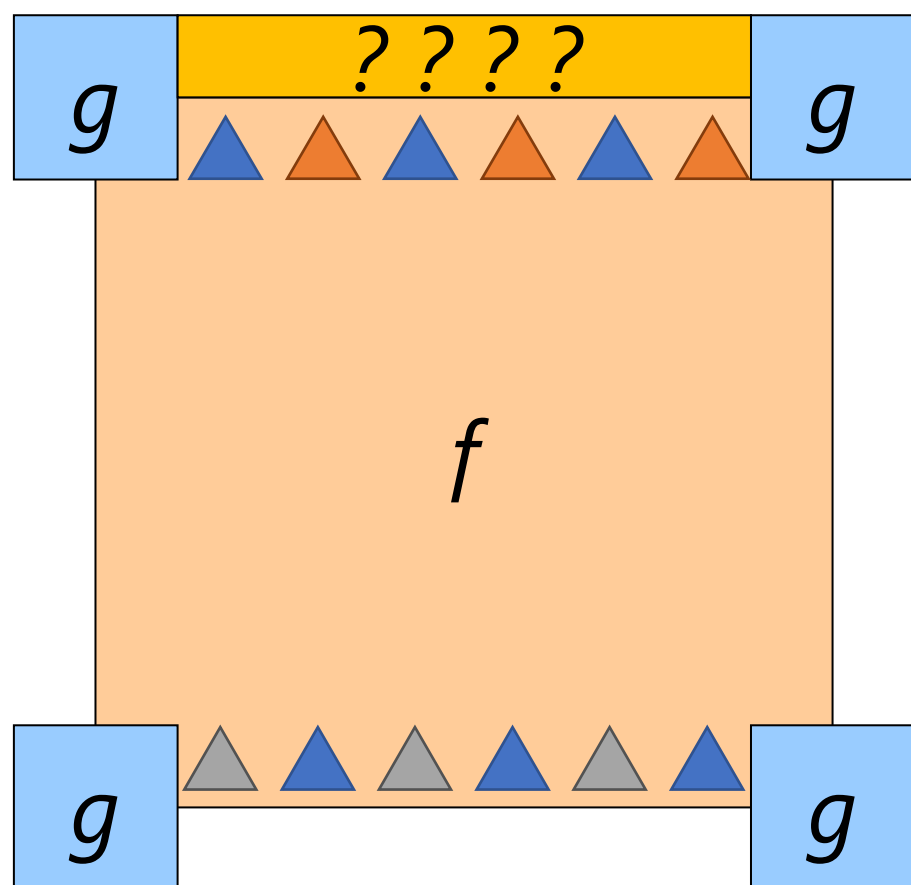
5x5 output

Stride 2 convolution:

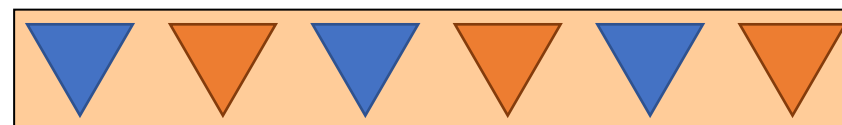
3x3 output

Padding

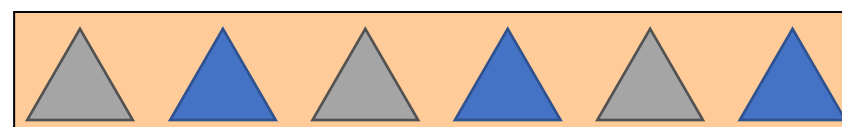
Padding用于填充外围缺失区域，形式有很多种



Symm: fold sides over



Circular/Wrap: wrap around

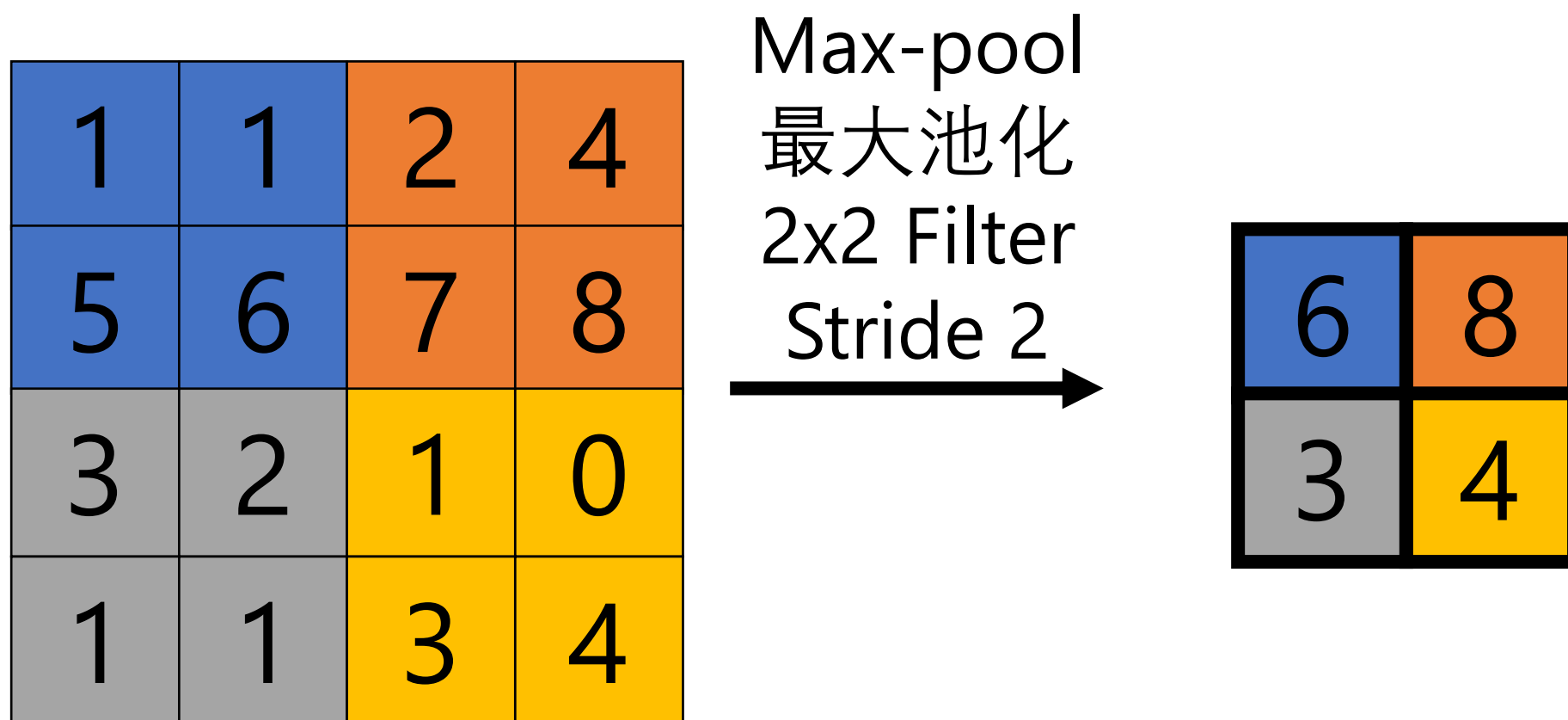


pad/fill: add value, often 0



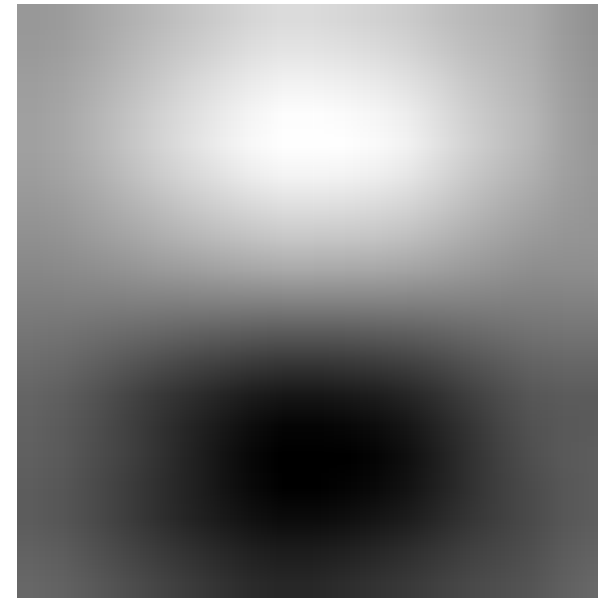
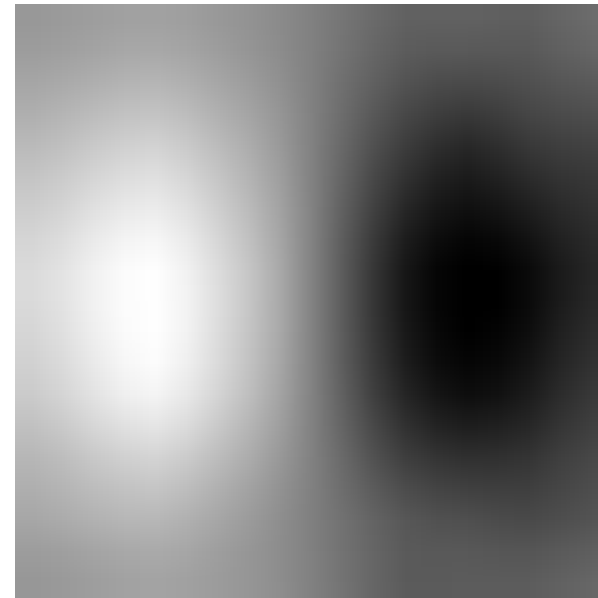
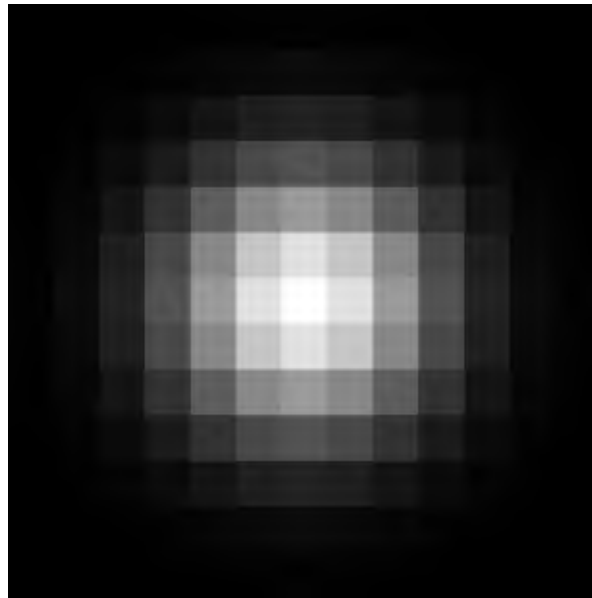
Pooling “池化”

出发点：使得特征的空间分辨率变小；
在每个通道上应用



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

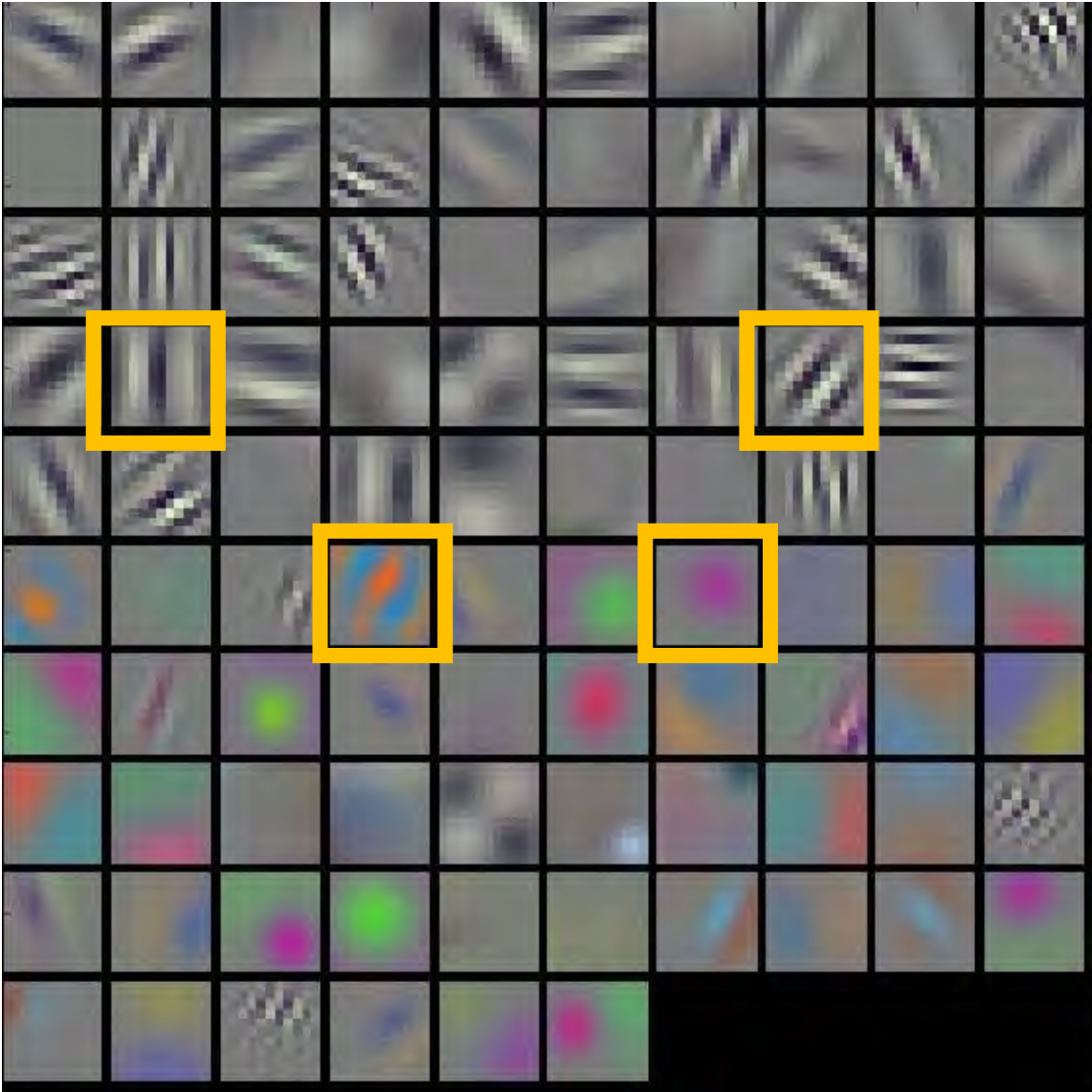


Figure Credit: Karpathy and Fei-Fei

每一层的输出

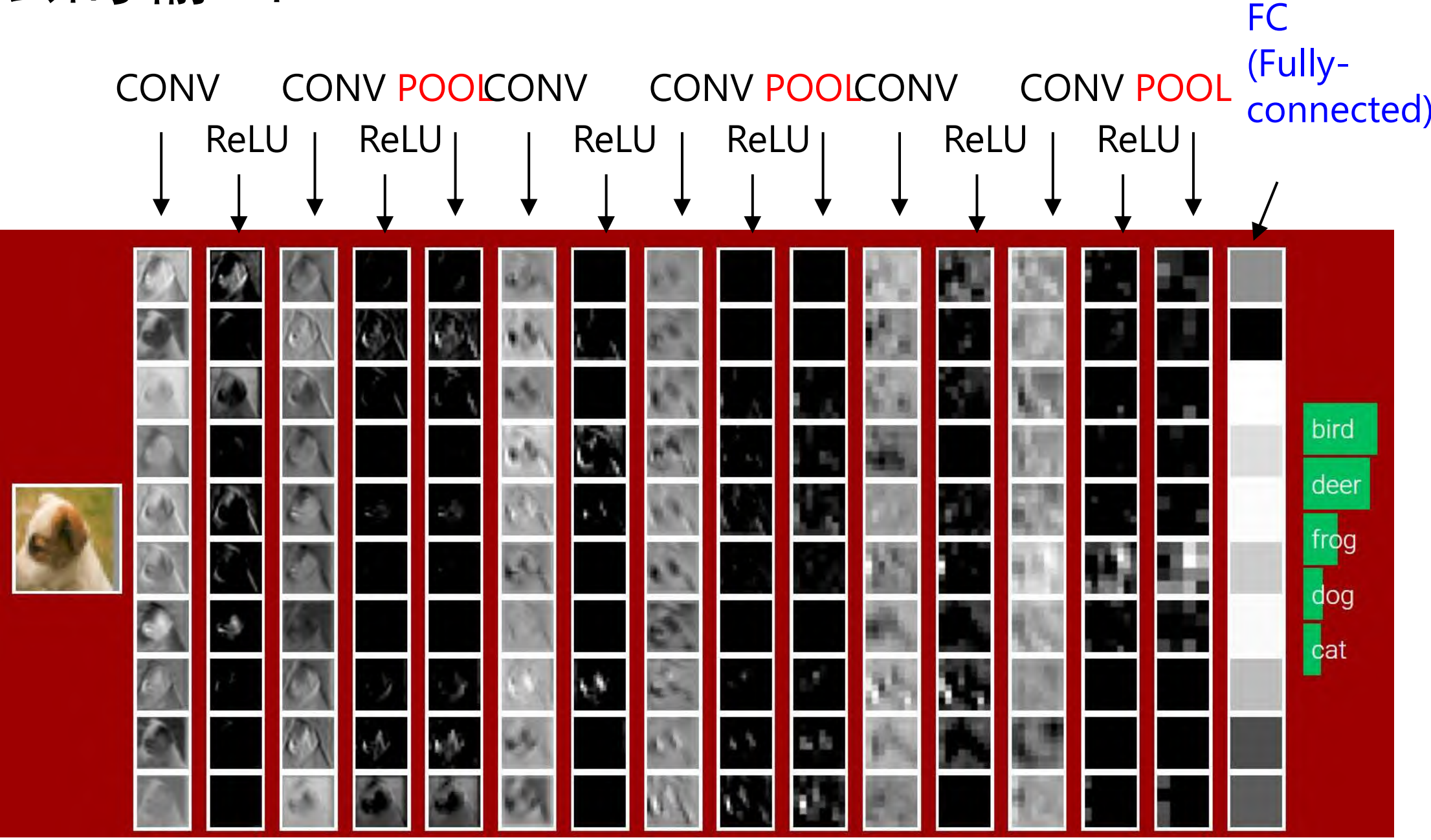
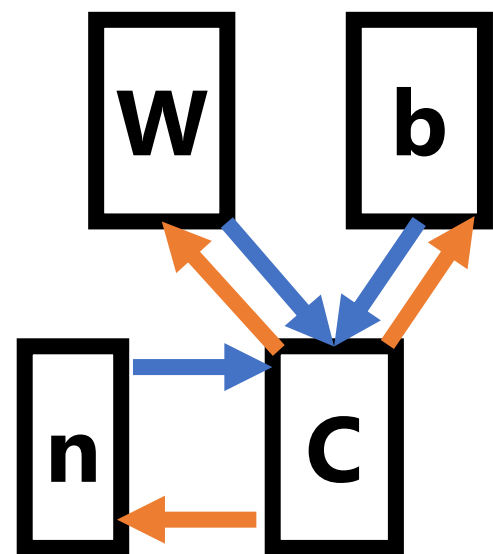


Figure Credit: Karpathy and Fei-Fei; see <http://cs231n.stanford.edu/>

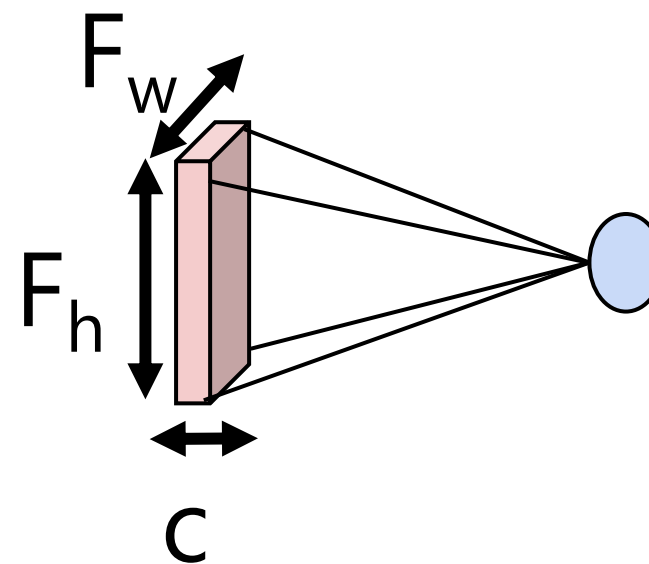
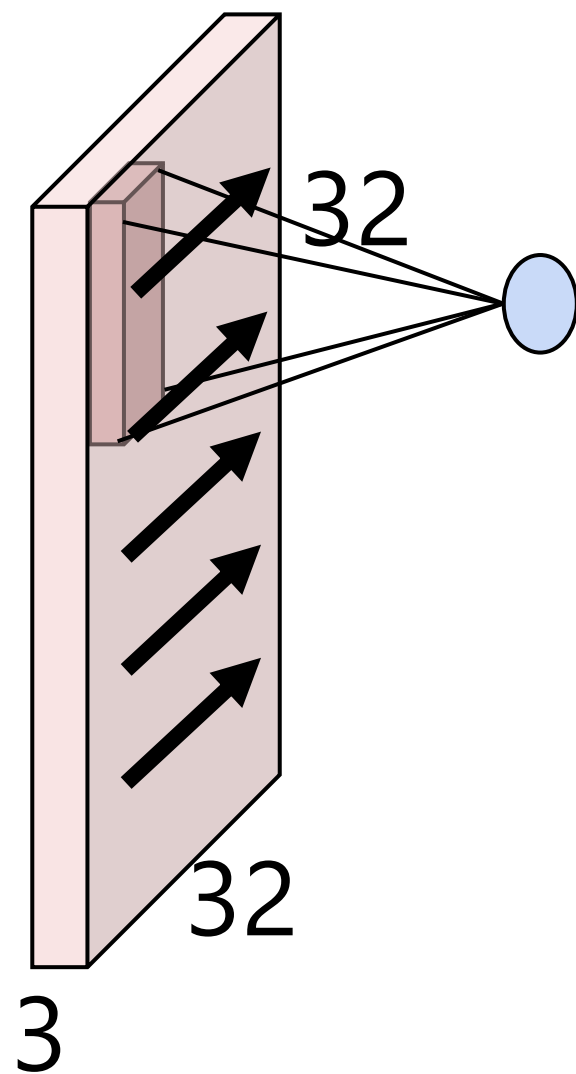
卷积层

新的网络模块：2DConv



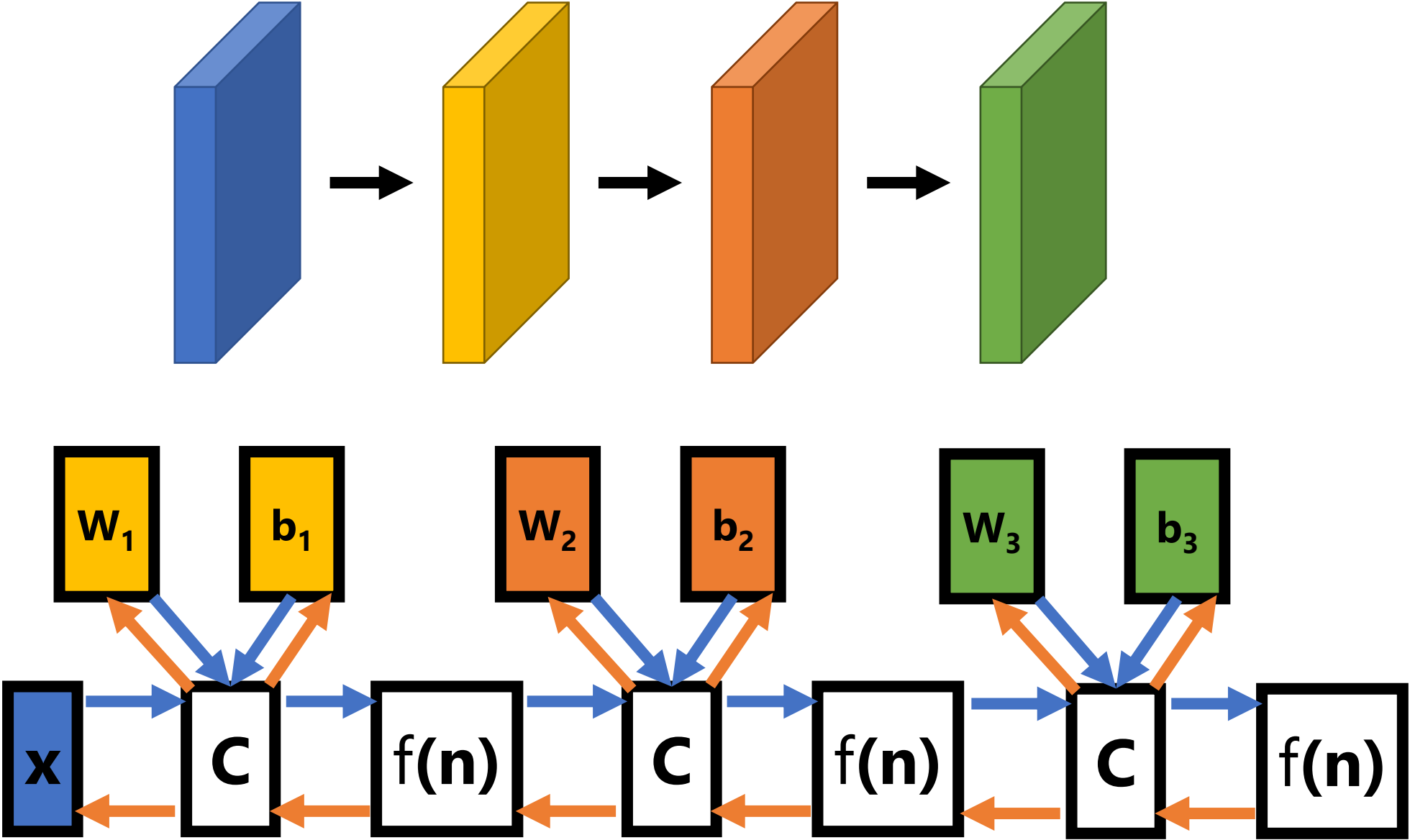
$$C(n) = n * F + b$$

卷积层

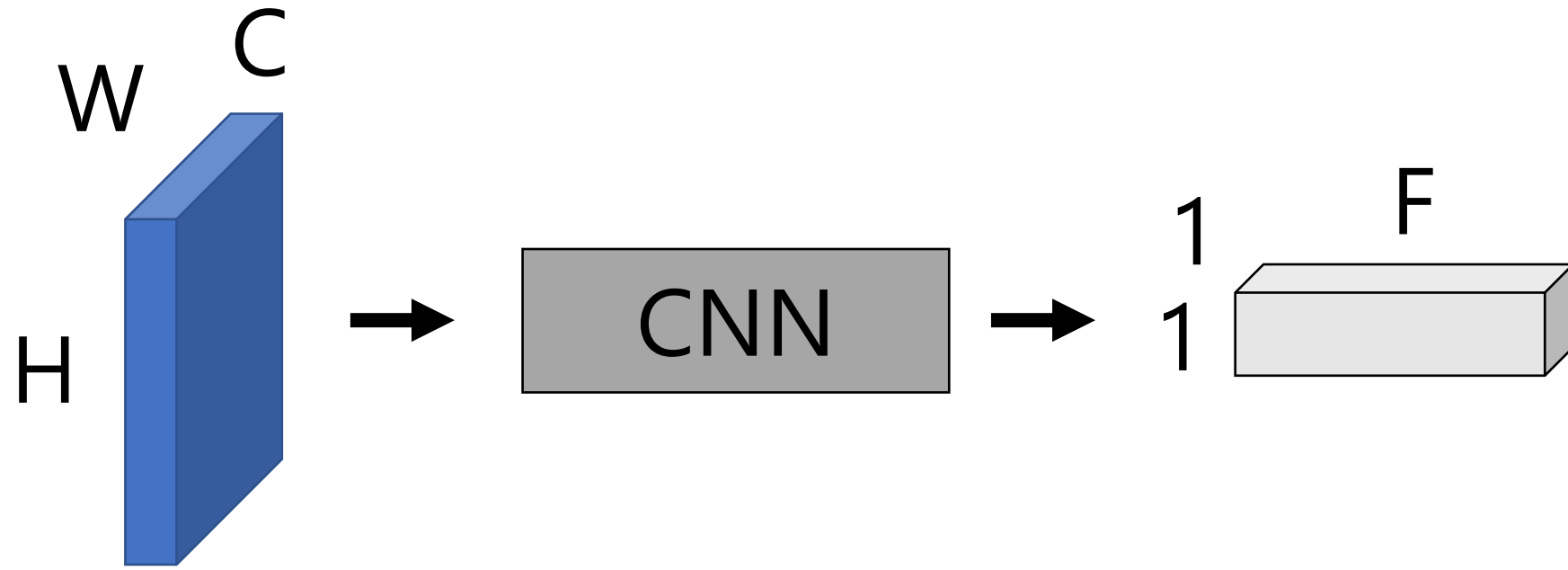


$$b + \sum_{i=1}^{F_h} \sum_{j=1}^{F_w} \sum_{k=1}^c F_{i,j,k} * I_{y+i,x+j,k}$$

Convolutional Neural Network 卷积神经网络 (CNN)



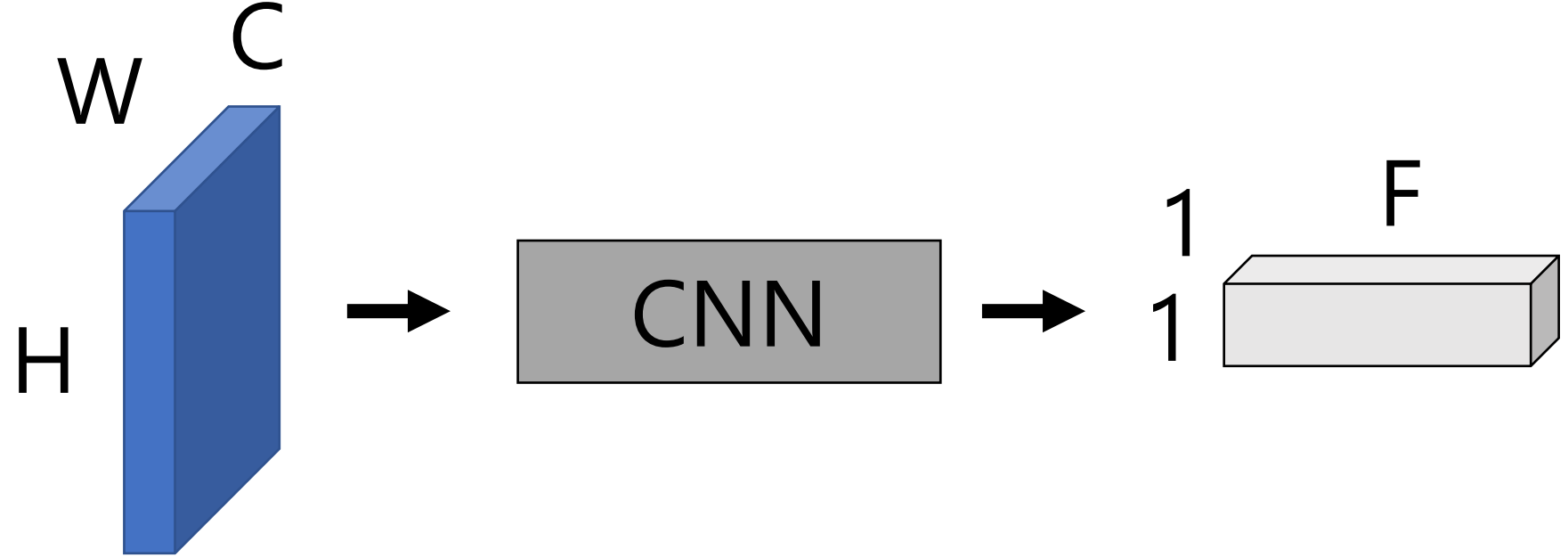
卷积神经网络能做很多任务



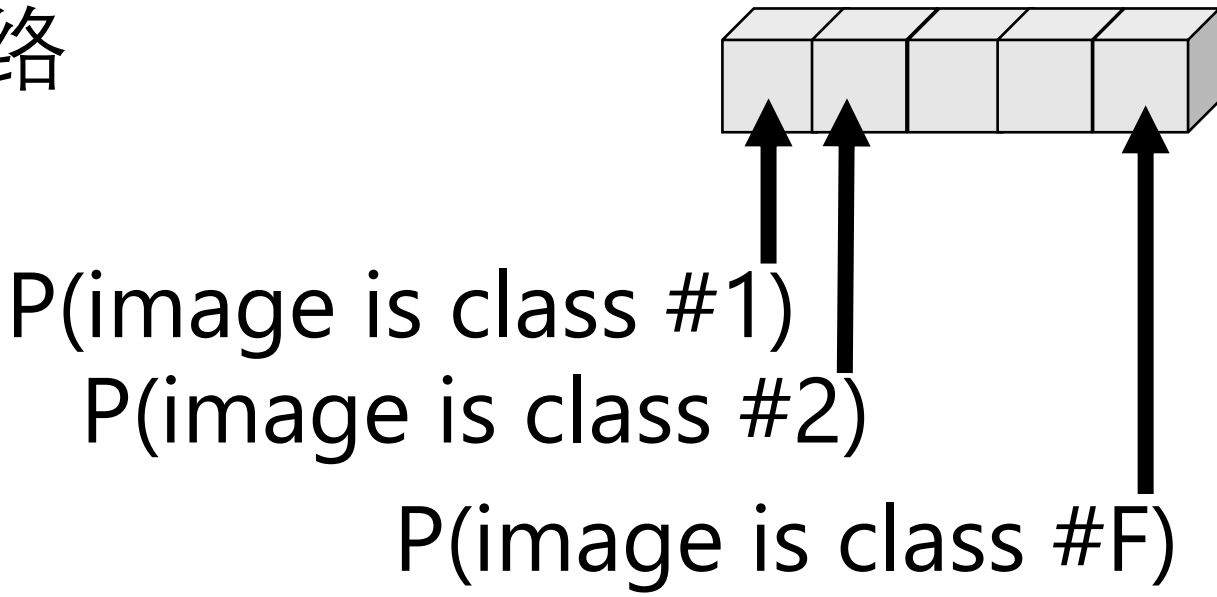
把 $H \times W$ 大小的图像转化为 F 大小的向量

- 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? ($F=56$)

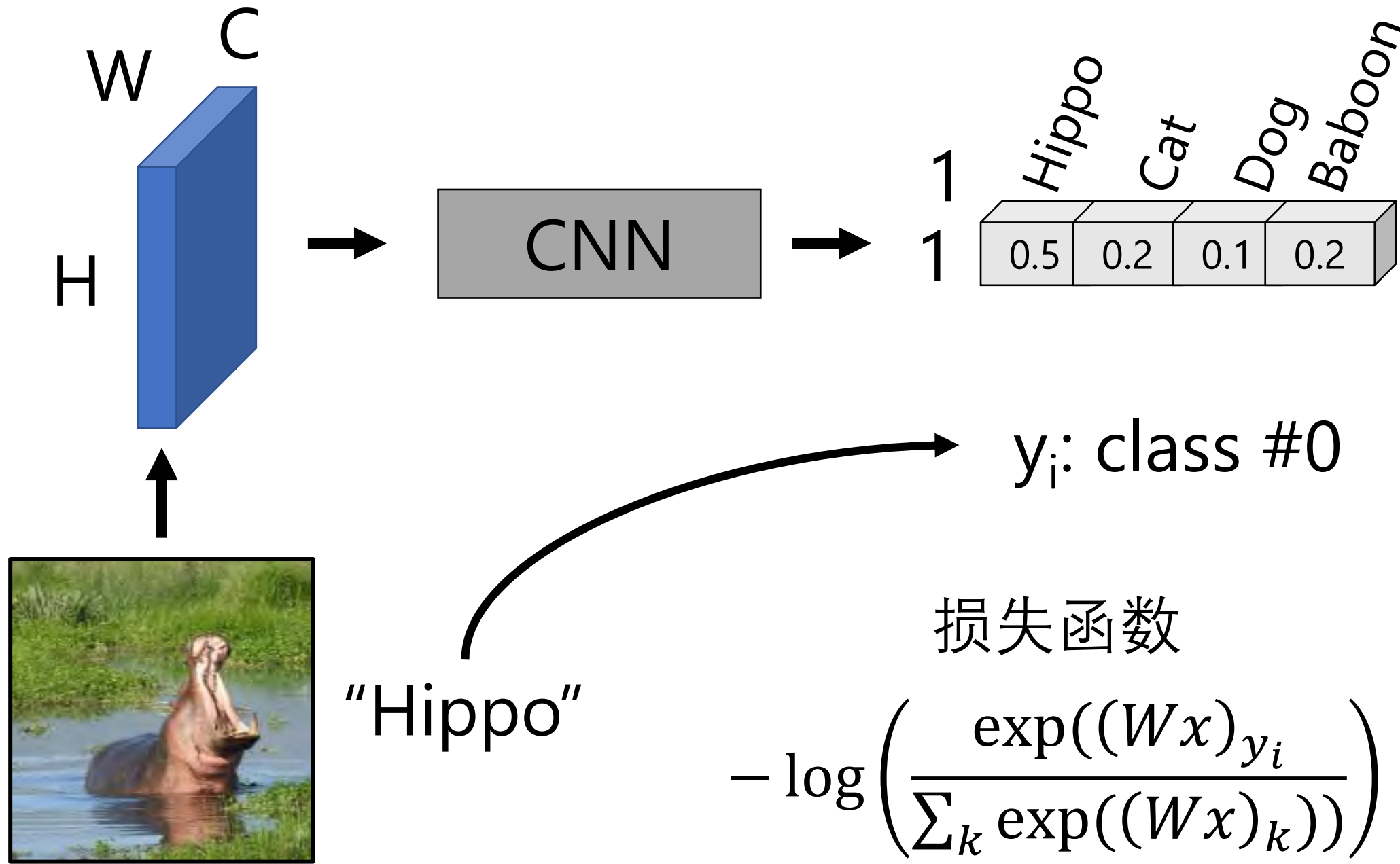
卷积神经网络：分类



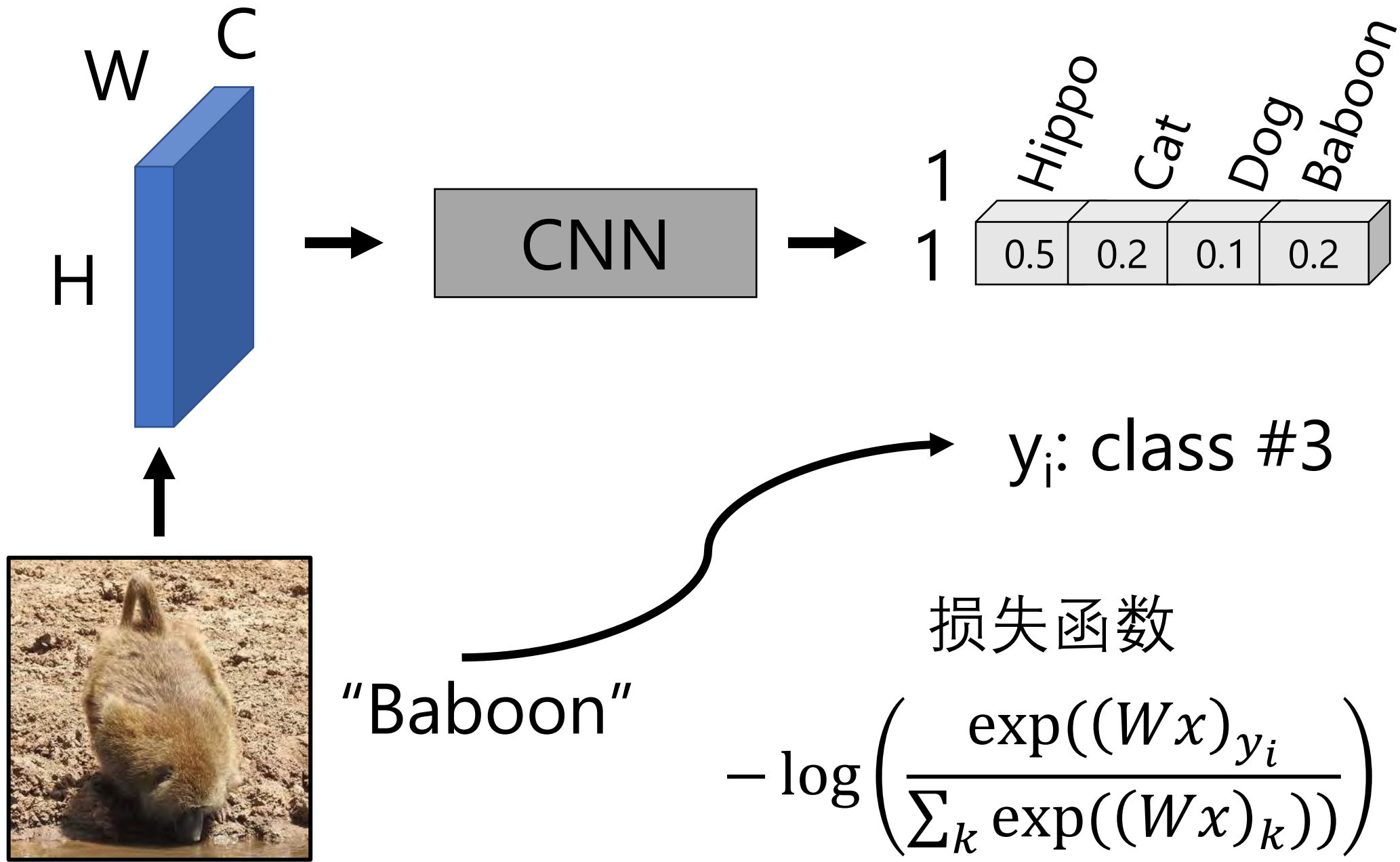
图像分类网络



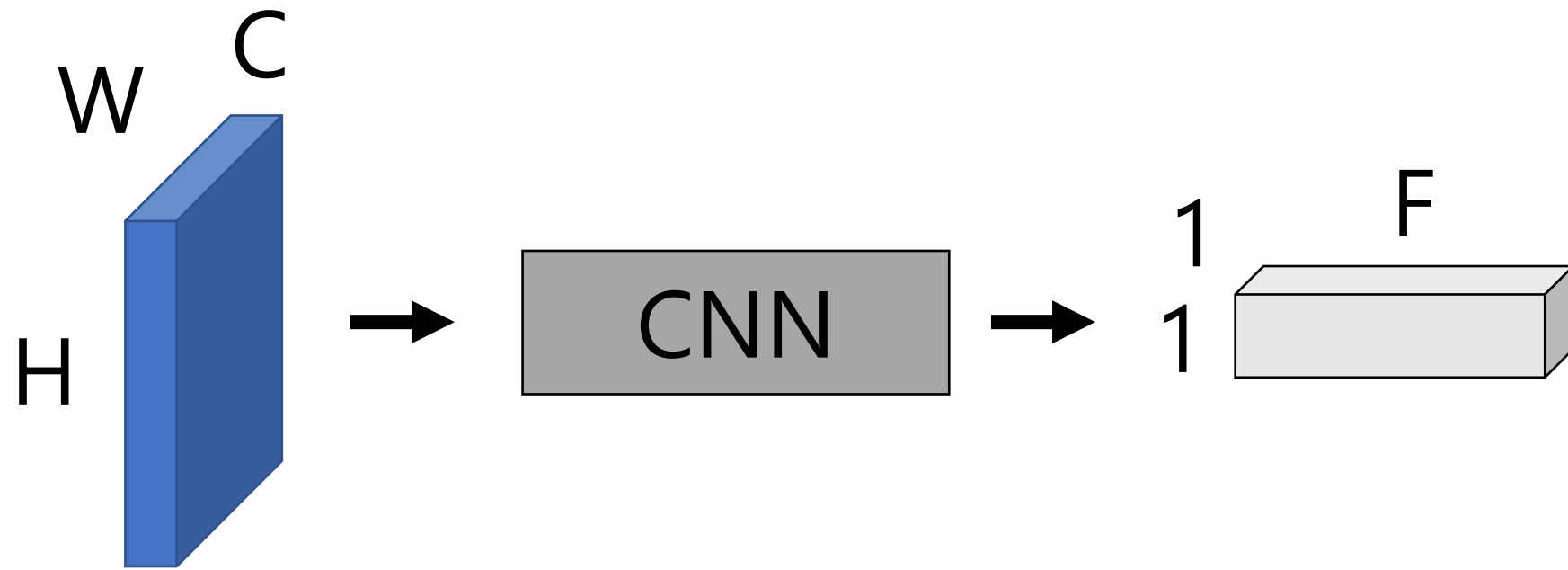
卷积神经网络：分类



卷积神经网络：分类

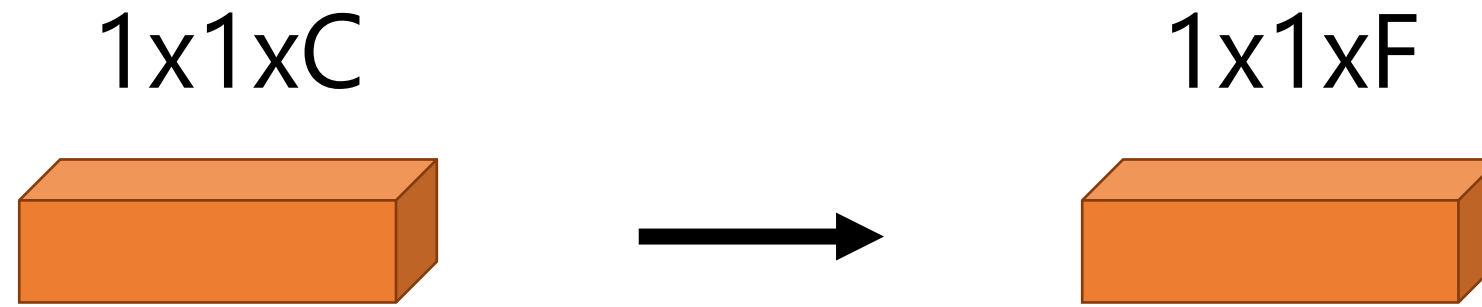


训练CNN以完成任务



- 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 $1 \times 1 \times F$ 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

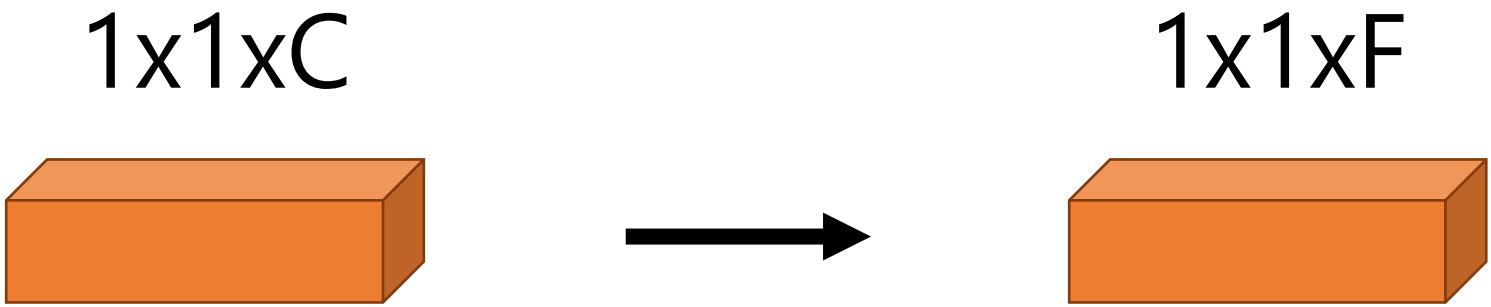
全连接层



用线性变换把 C -维 特征 映射到 F -维 特征
 W ($F \times C$ matrix) + b ($F \times 1$ vector)

可以用卷积实现吗？

Everything's a Convolution



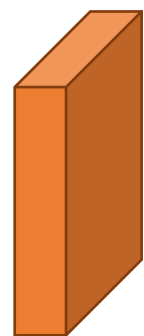
让 $F_h=1, F_w=1$

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？

$H \times W \times C$



$1 \times 1 \times F$

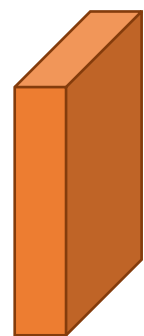


怎么做？

特征转化为向量 —— Pooling

HxWxC

1x1xF



1	1	2	4
5	6	7	8
3	2	1	0
1	1	3	4

Avg Pool
HxW Filter
Stride 1



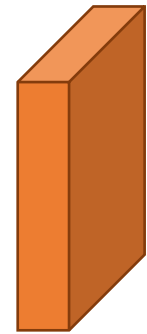
3.1

*(If F == C)

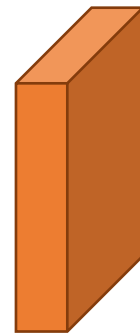
特征转化为向量——卷积

$H \times W \times C$

$1 \times 1 \times F$



$H \times W$ 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



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

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

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!

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!

样例卷积网络

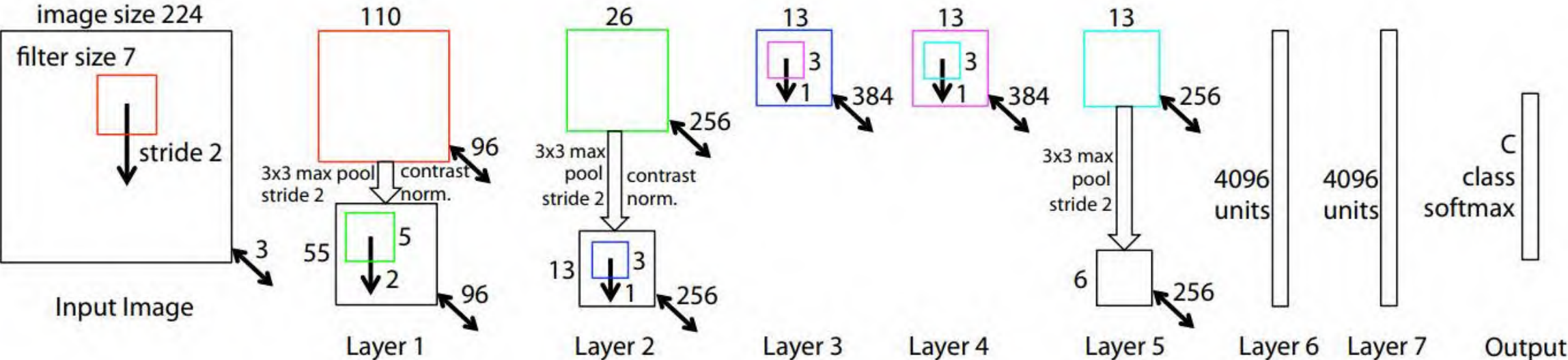
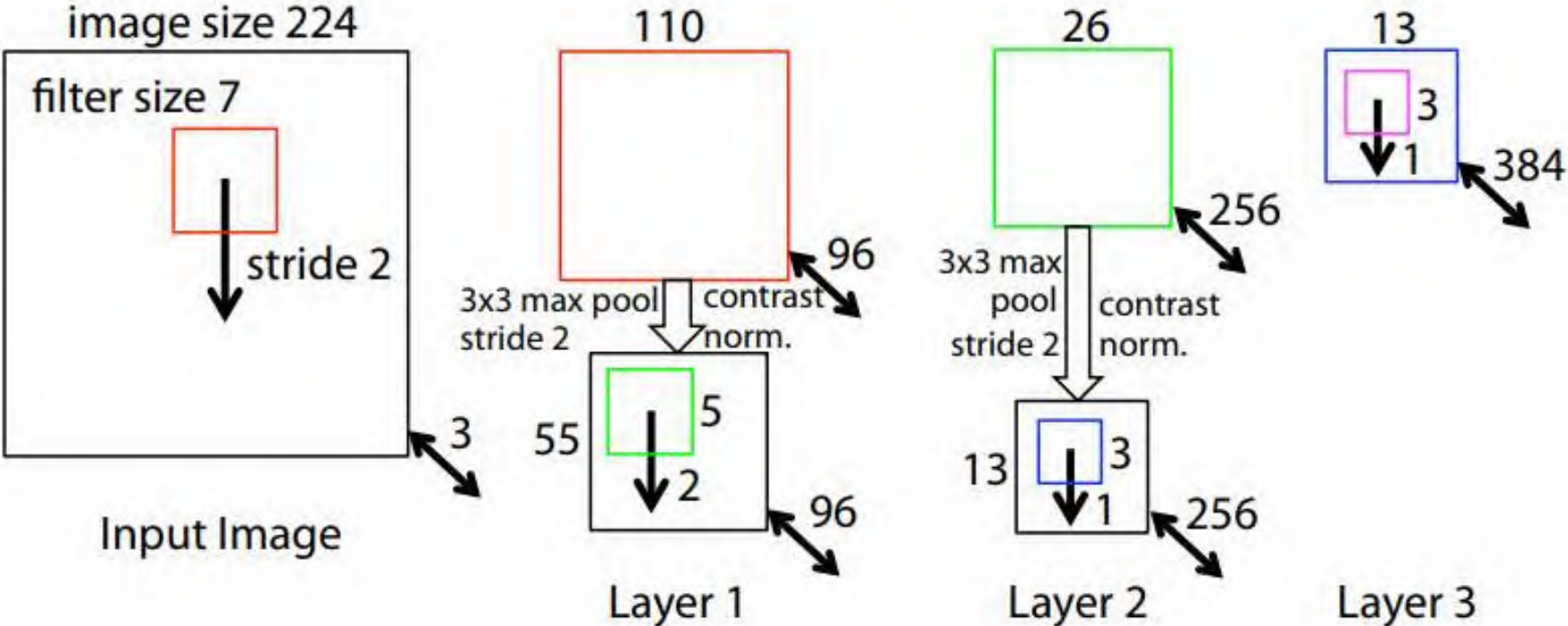


Figure Credit: Zeiler and Fergus, Visualizing and Understanding Convolutional Networks. ECCV 2014

样例卷积网络



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

经典网络设计

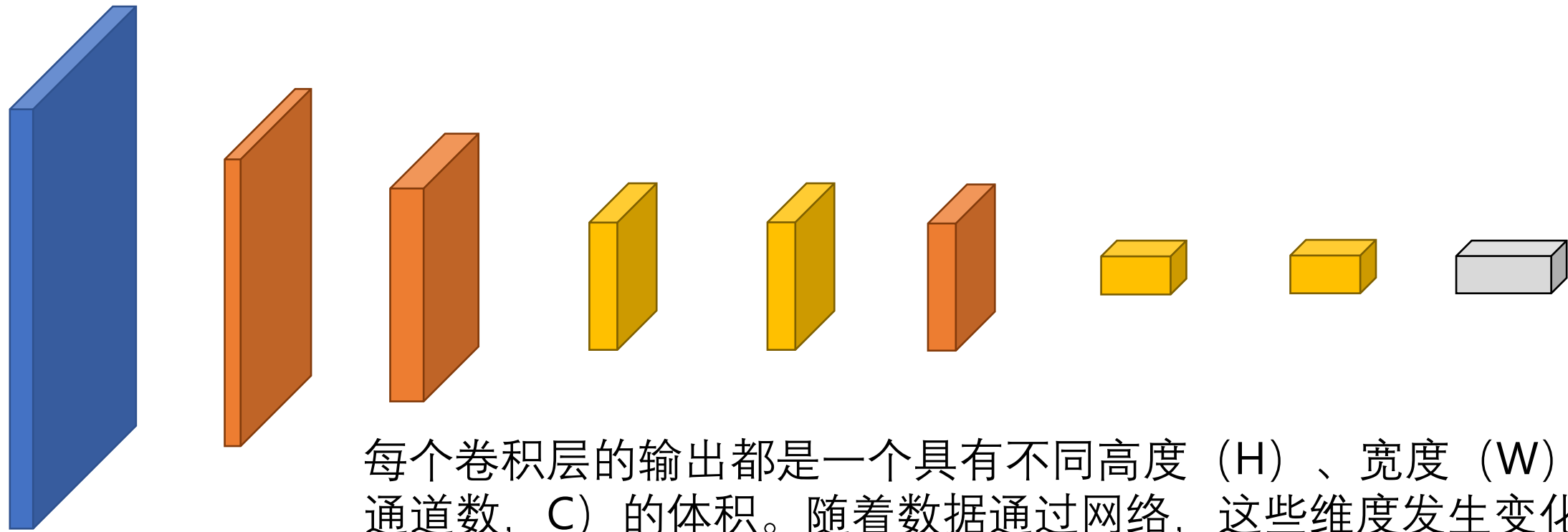
- 输出 1000 类分类评分(Imagenet)
 - Alexnet (2012)
 - VGG-16 (2014)
 - Resnet (2015)

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

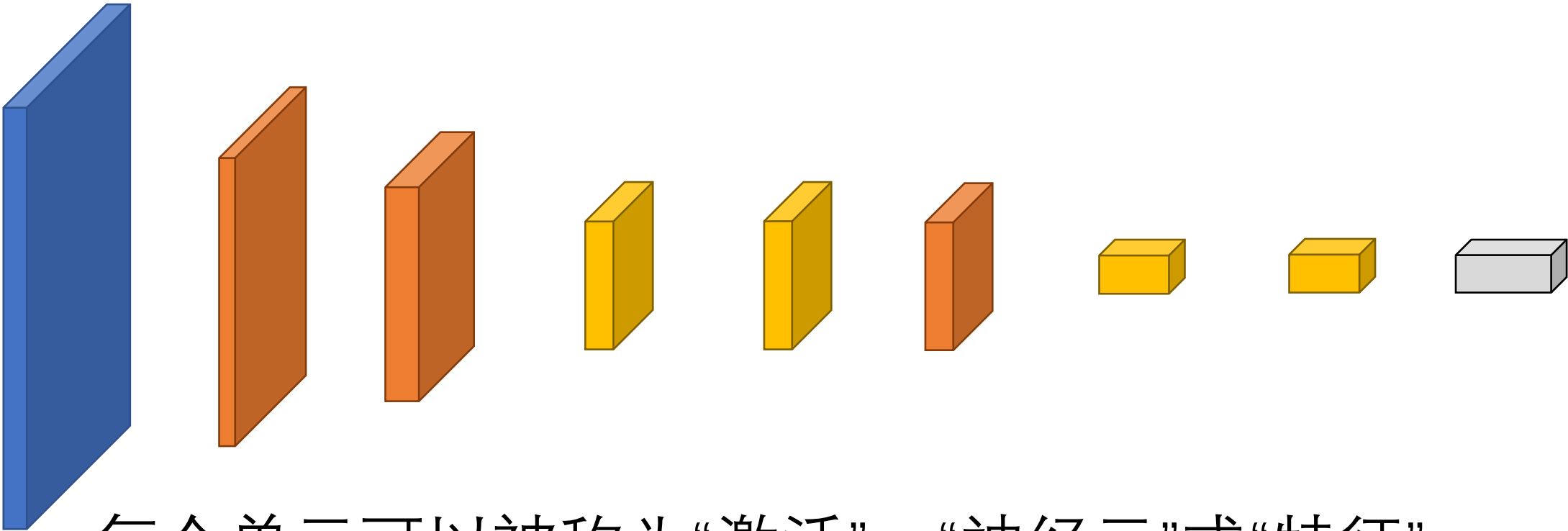
Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
227x227 3	55x55 96	27x27 256	13x13 384	13x13 384	13x13 256	1x1 4096	1x1 4096	1x1 1000



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。

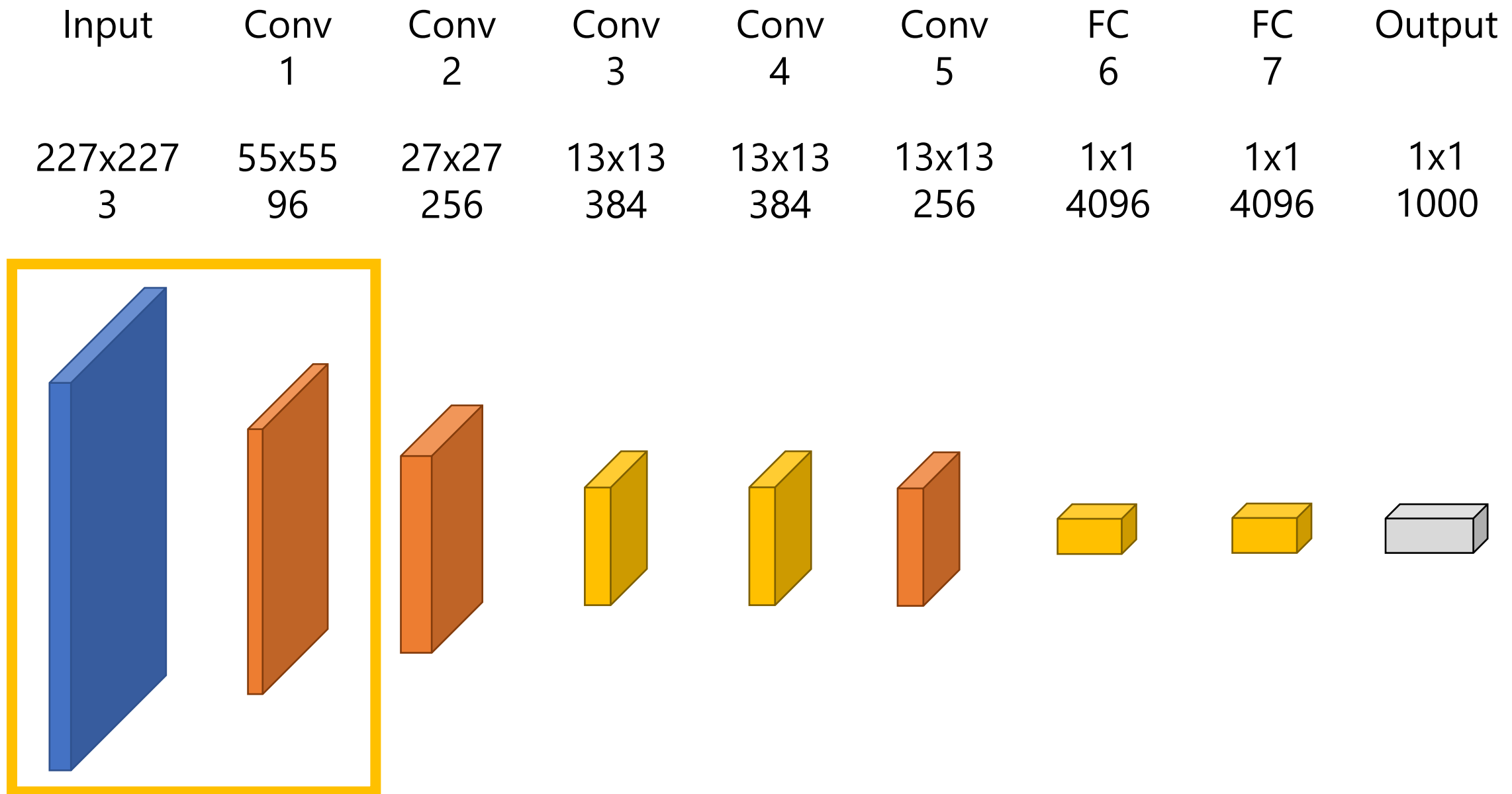
CNN基本术语

Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
227x227	55x55	27x27	13x13	13x13	13x13	1x1	1x1	1x1
3	96	56	384	384	256	4096	4096	1000

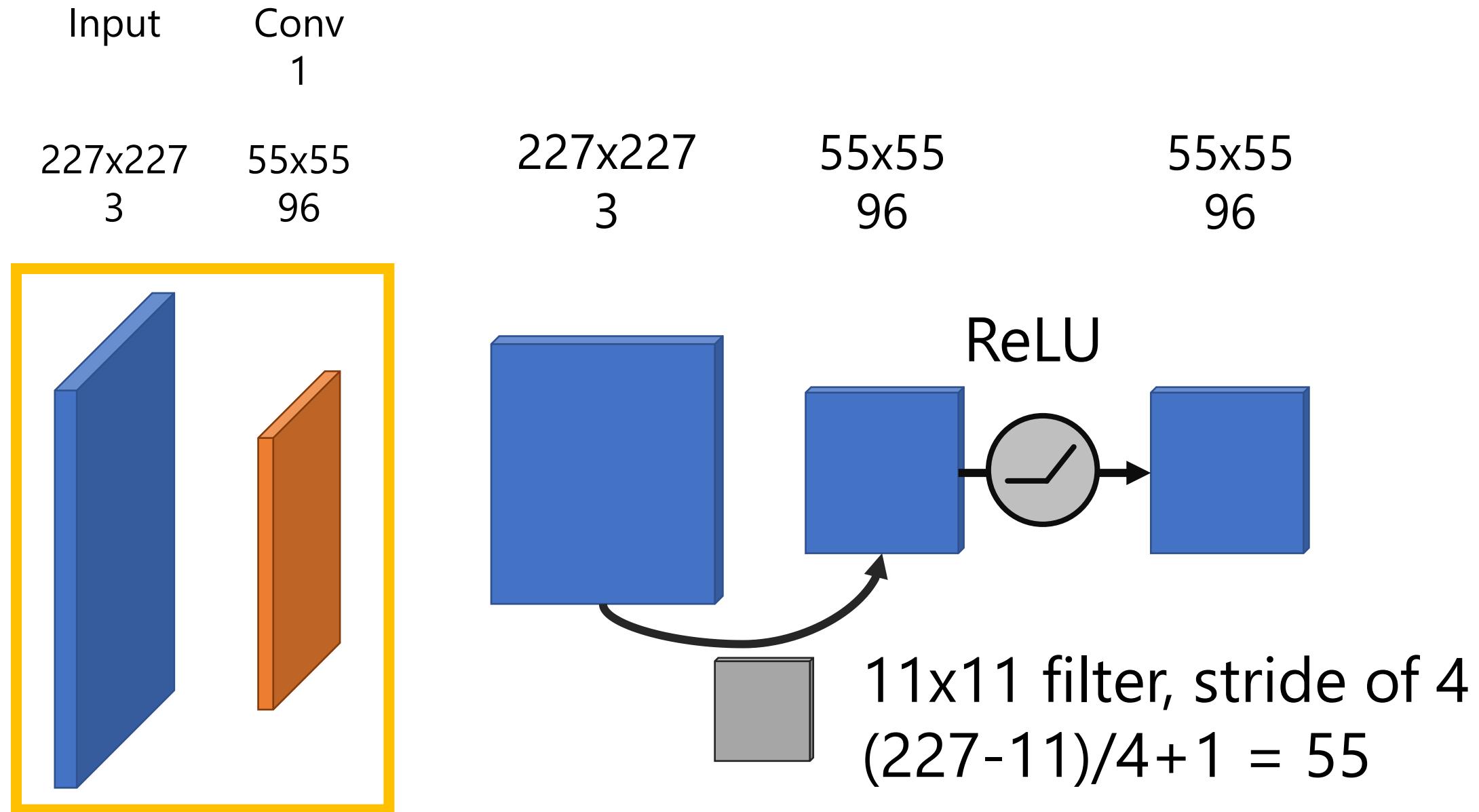


每个单元可以被称为“激活”、“神经元”或“特征”。

AlexNet

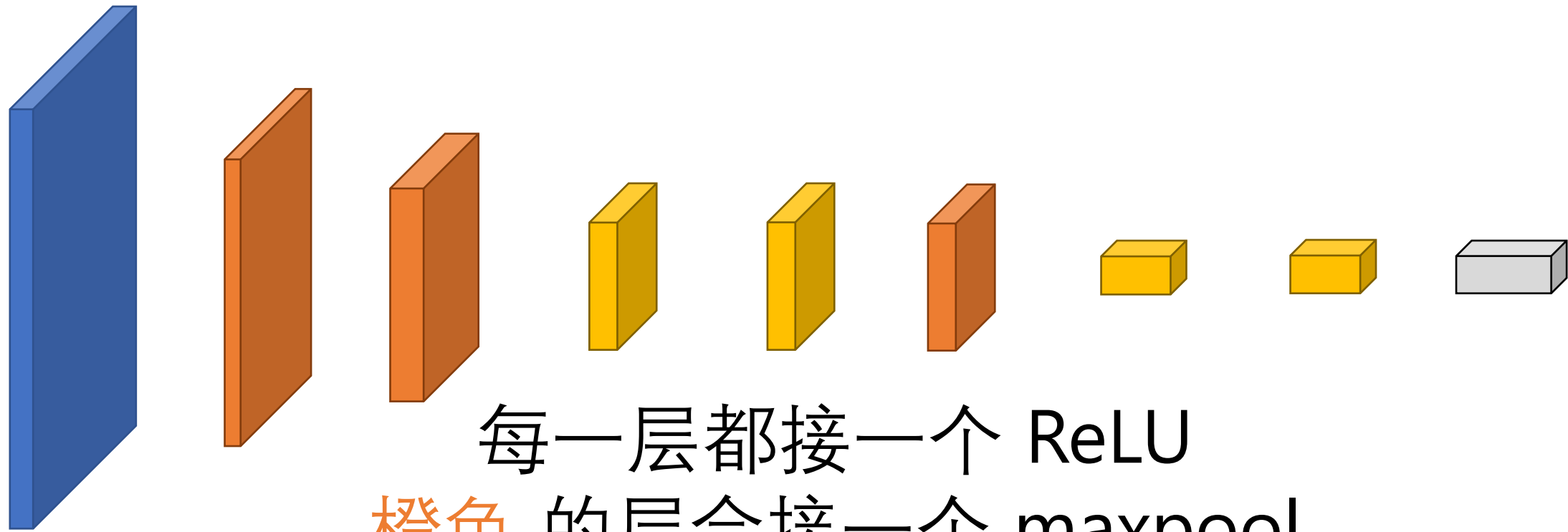


AlexNet



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



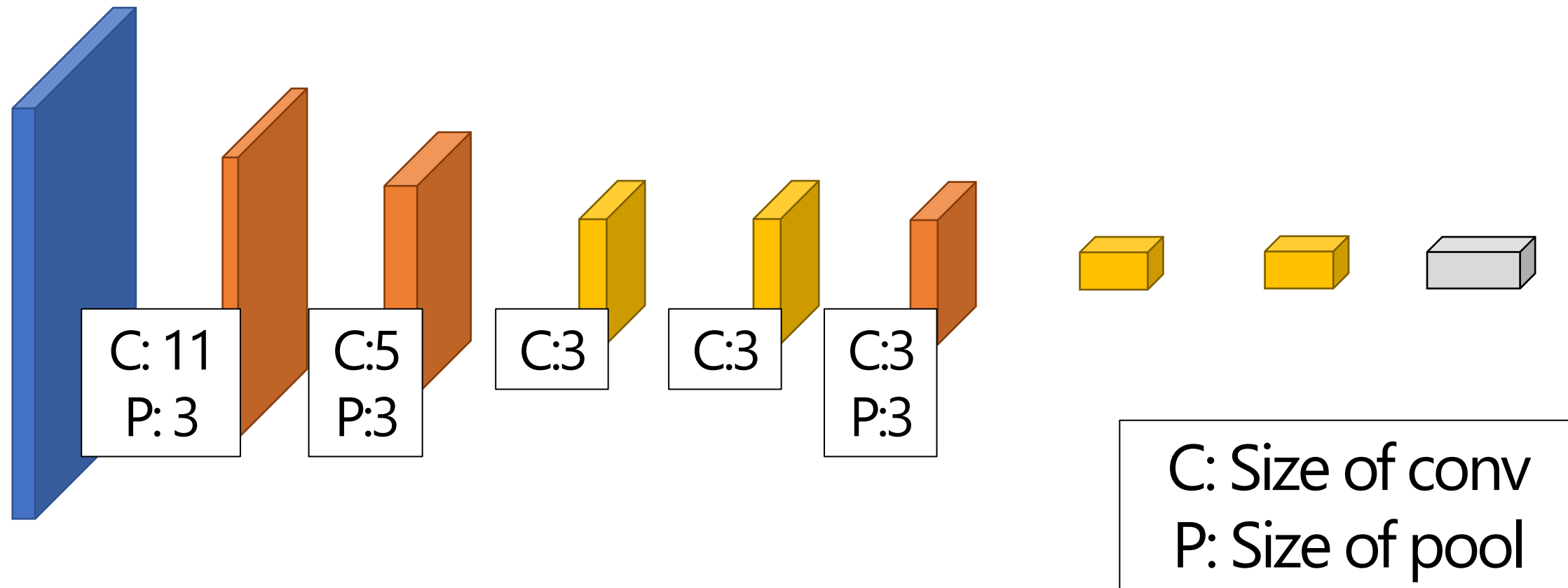
每一层都接一个 ReLU

橙色 的层会接一个 maxpool

每一层都有 "normalization" 标准化

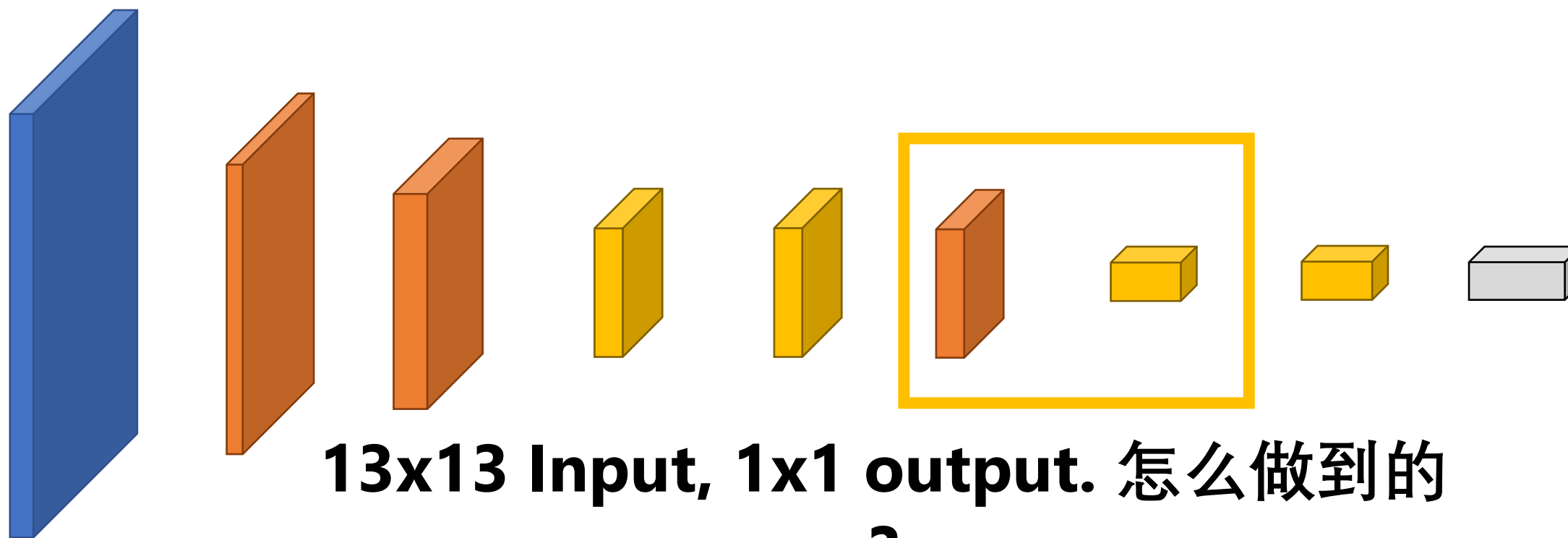
AlexNet – Details

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

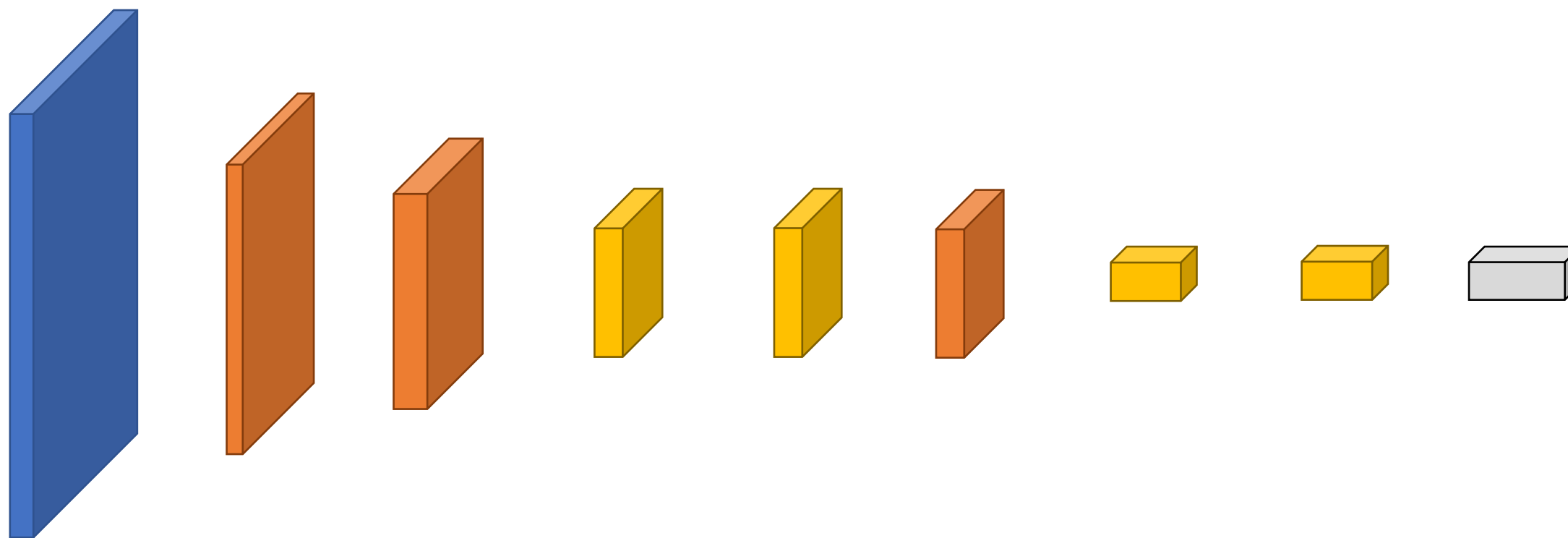
Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
227x227 3	55x55 96	27x27 256	13x13 384	13x13 384	13x13 256	1x1 4096	1x1 4096	1x1 1000



13x13 Input, 1x1 output. 怎么做到的?
?

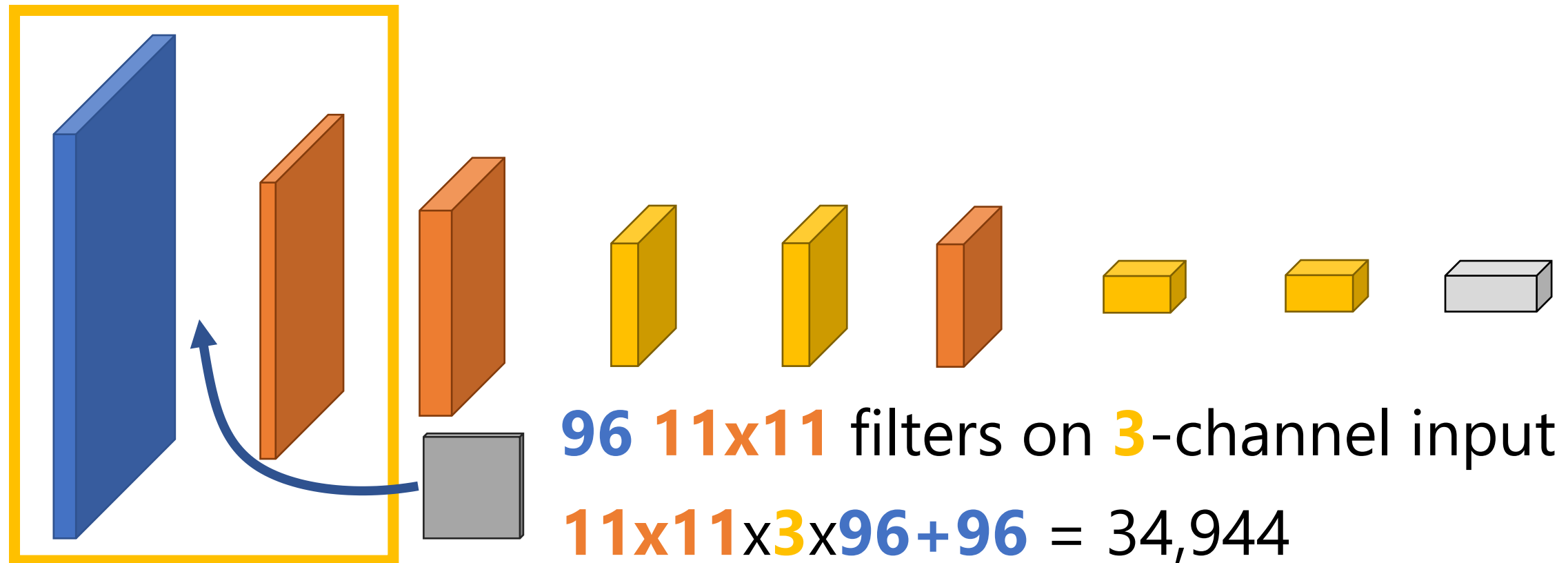
Alexnet – 有多少参数?

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



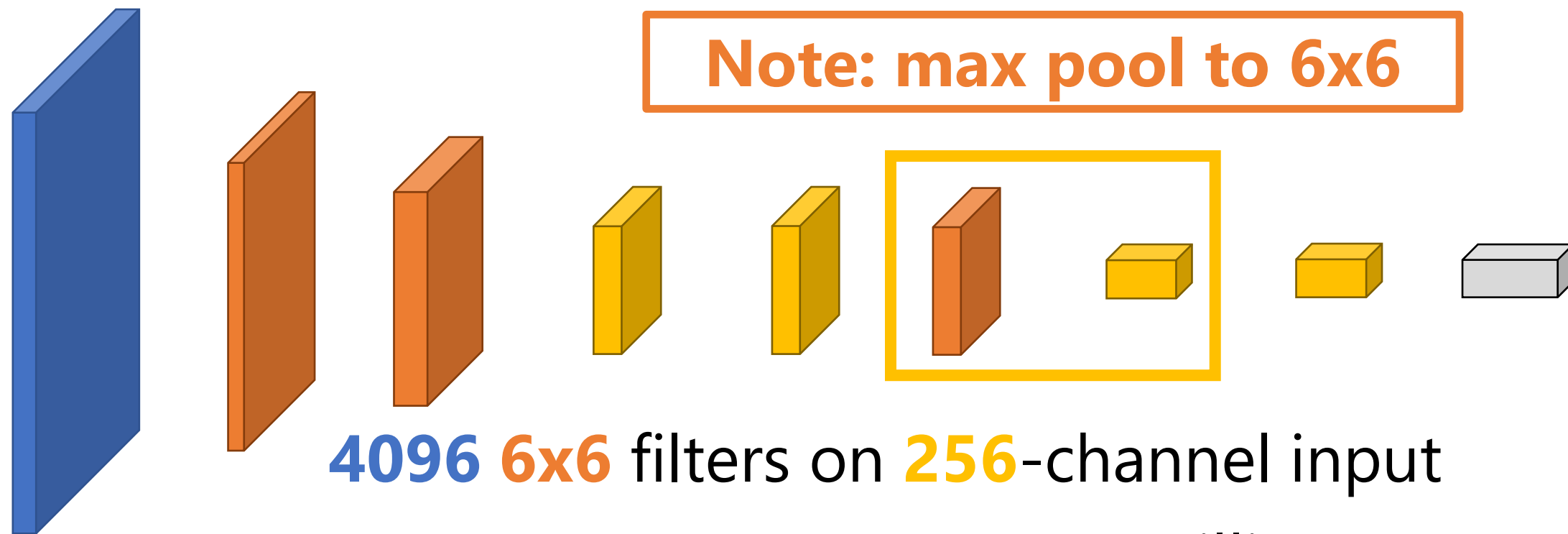
Alexnet - 有多少参数?

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



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

$$6 \times 6 \times 256 \times 4096 + 4096 = 38 \text{ million}$$

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

$$1x1x4096x4096+4096 = 17 \text{ million}$$

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

birds



flamingo



cock



ruffed grouse



quail



partridge . . .

bottles



pill bottle



beer bottle



wine bottle



water bottle



pop bottle . . .

cars



race car



wagon



minivan



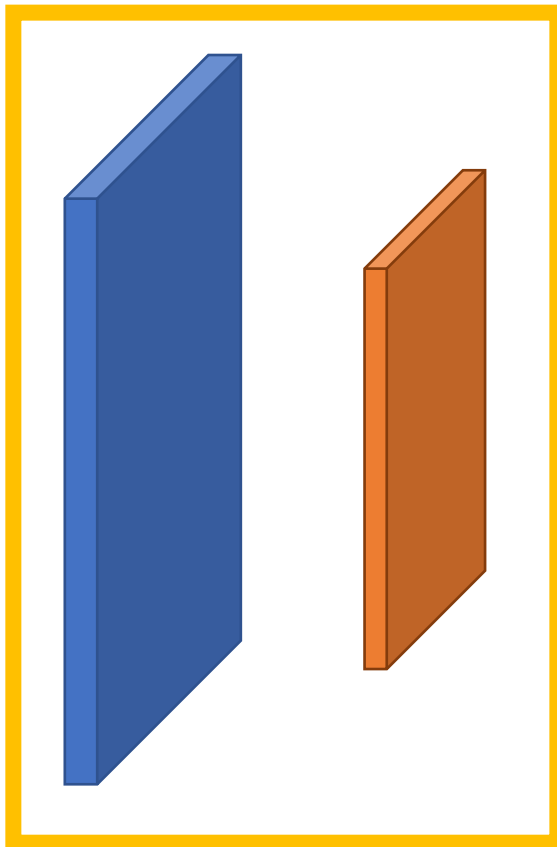
jeep



cab . . .

可视化卷积核

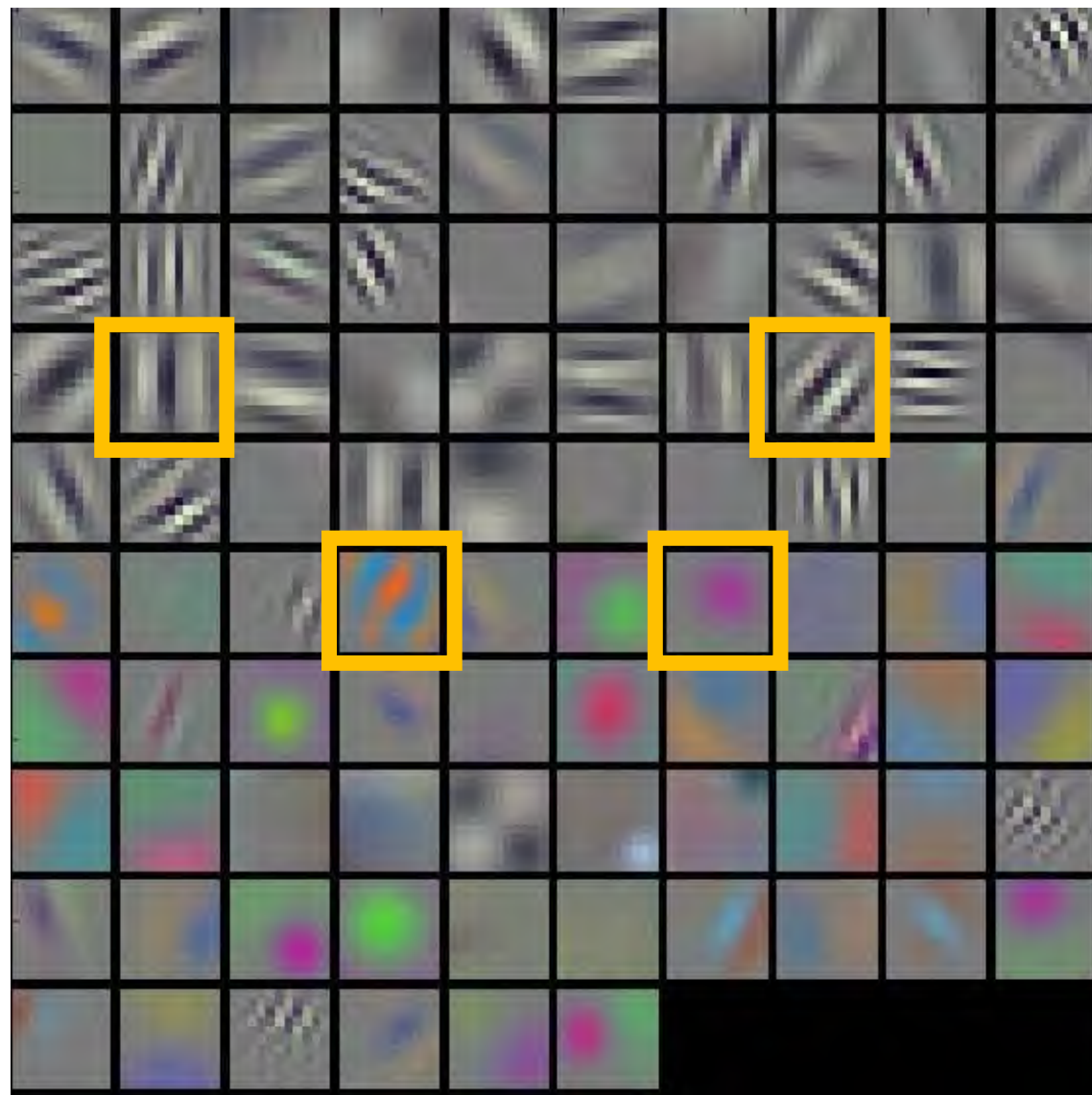
Input	Conv 1
227x227	55x55
3	6



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？

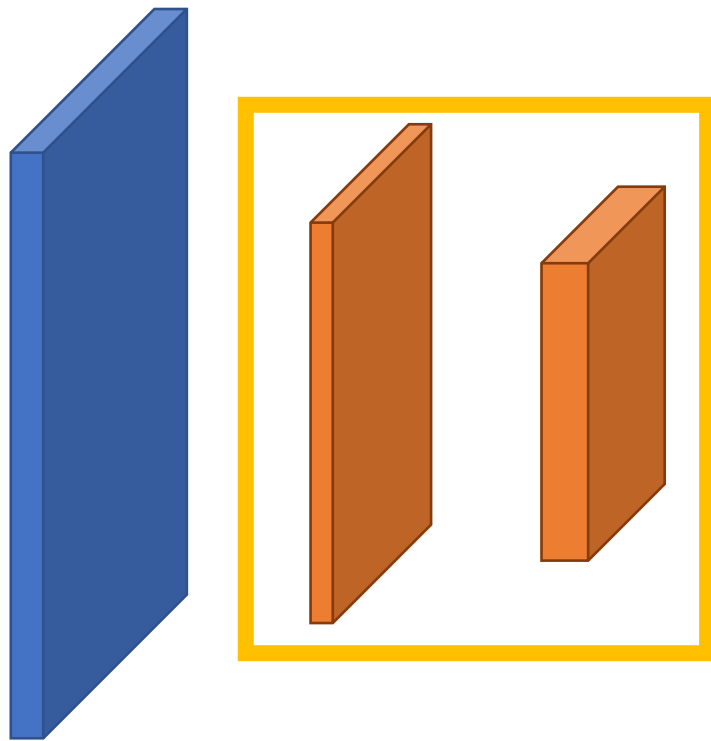


第一层卷积滤波器学到的特征，这些滤波器被训练用于识别1000个不同的对象类别。

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

更深层次的滤波器的可视化

Input	Conv 1	Conv 2
227x227 3	55x55 6	27x27 56



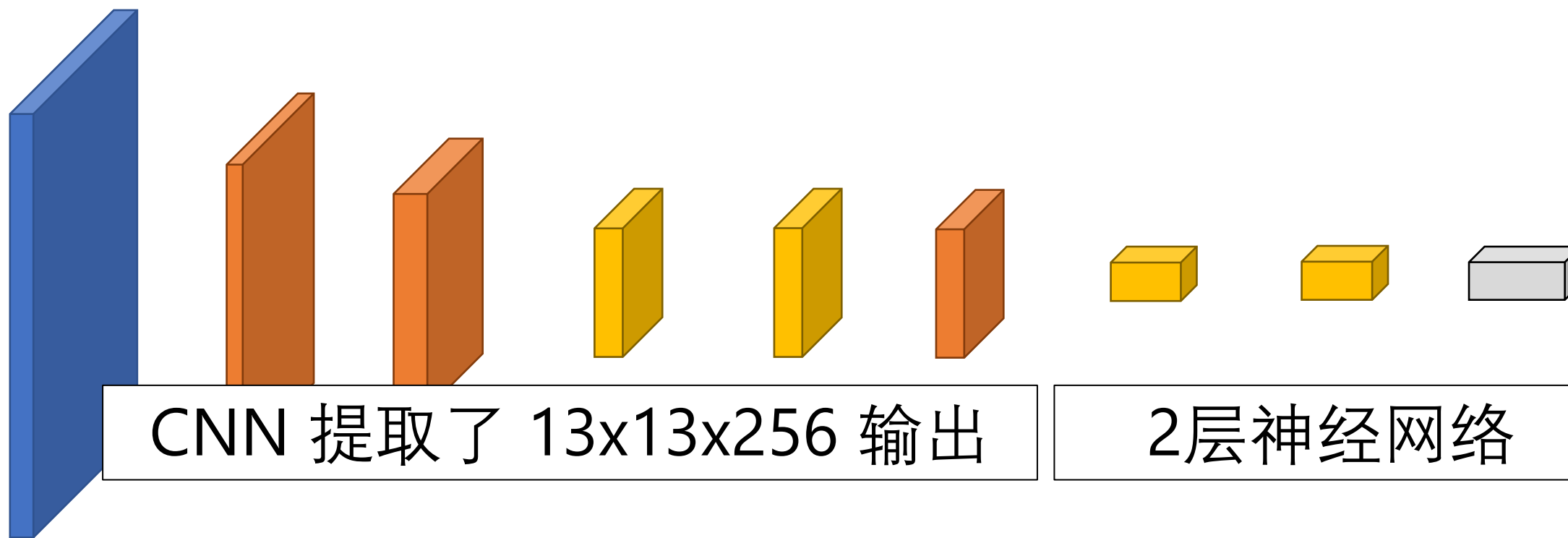
- 理解更深层卷积层的滤波器的具体值是非常困难的，甚至是不可能的，因为输入维度太多，并且输入的具体含义不明确。这是因为随着网络层数的加深，每个滤波器所代表的特征变得更加抽象和复杂，不再像初级滤波器那样容易通过视觉上直接解释。

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。。。。

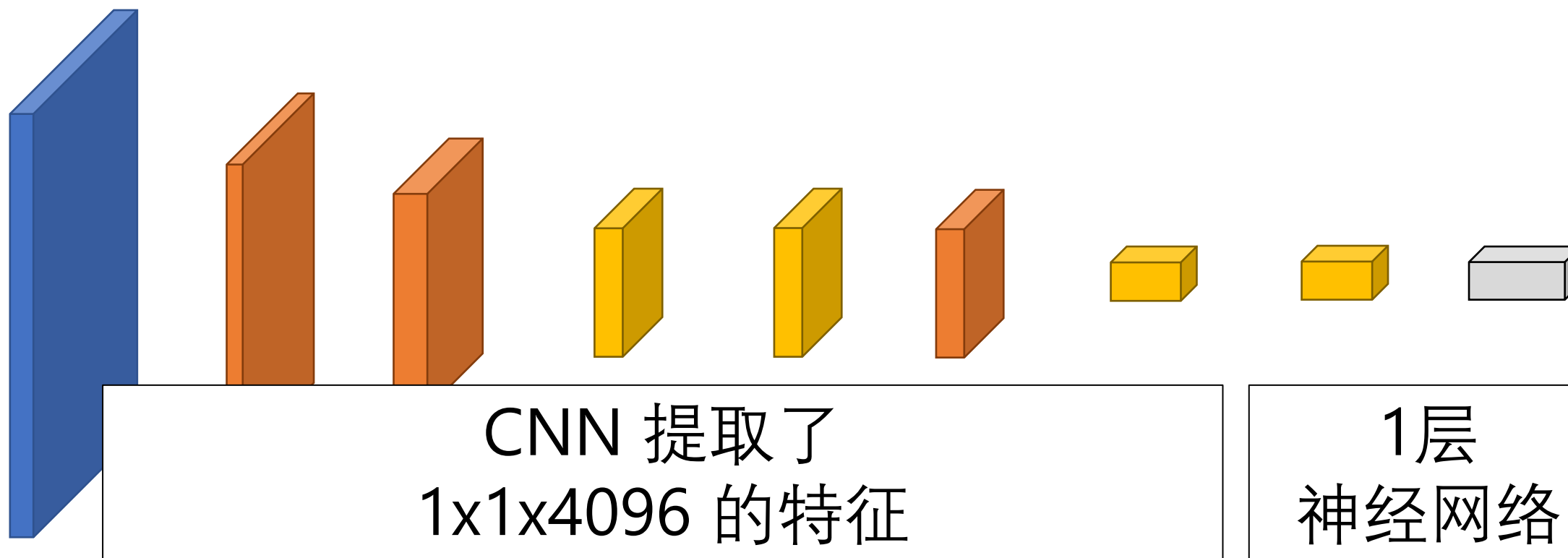
理解深层网络

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



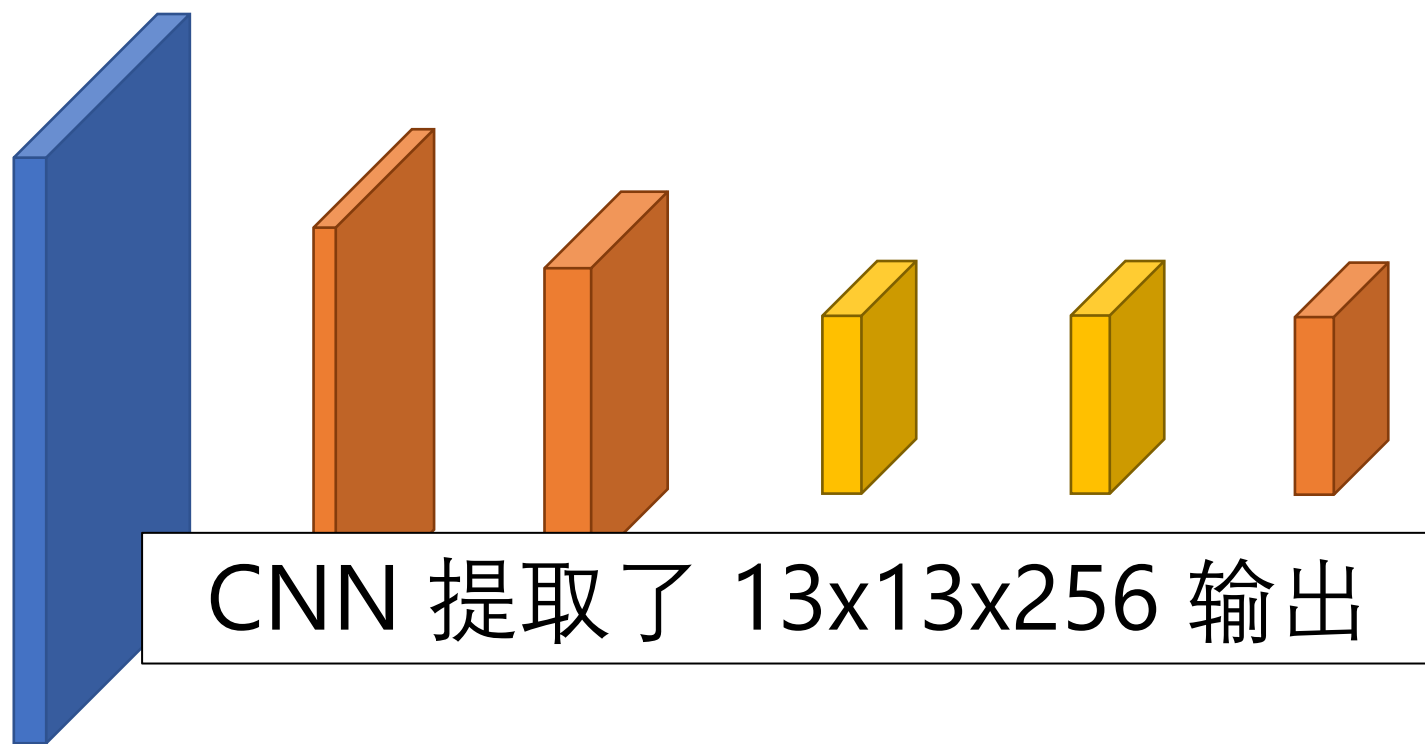
理解深层网络

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

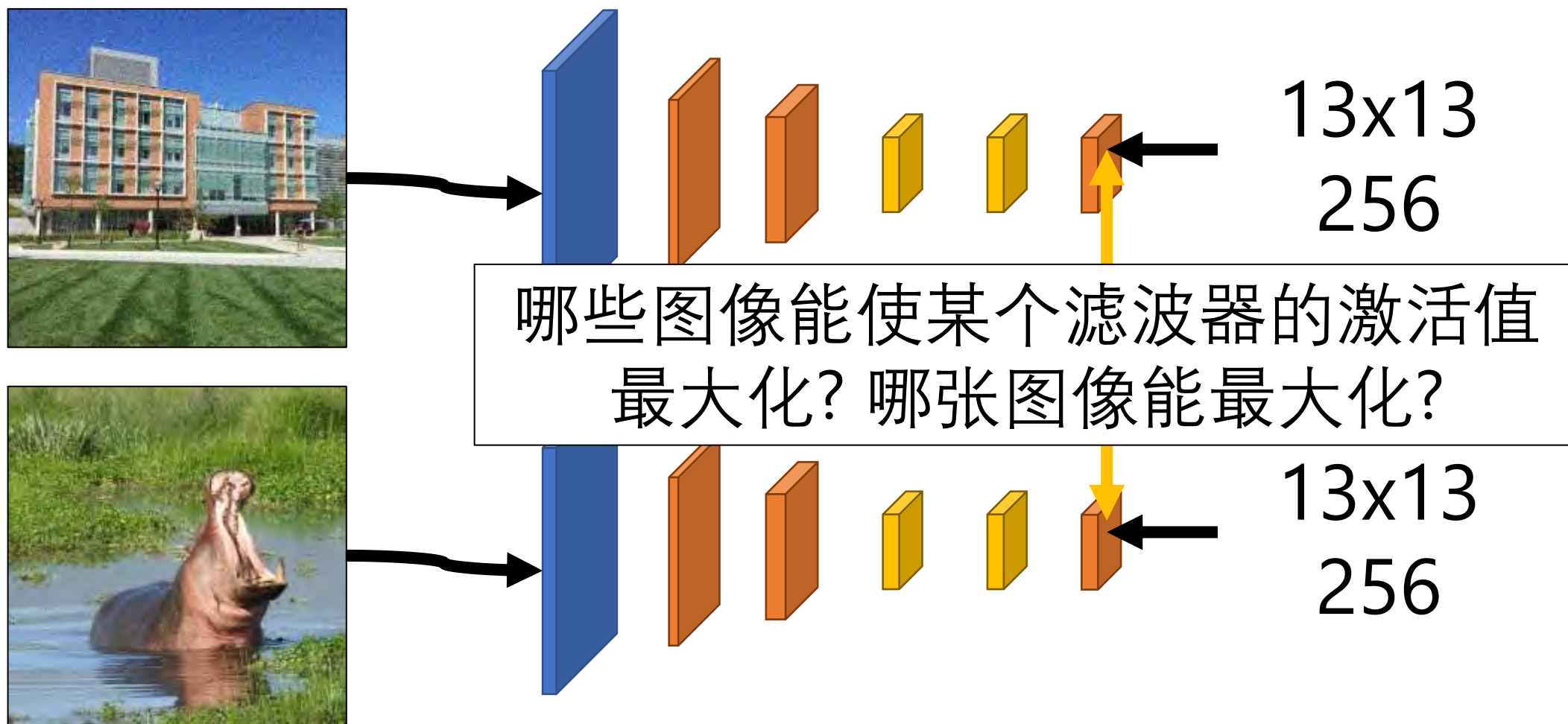
Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5
227x227 3	55x55 96	27x27 256	13x13 384	13x13 384	13x13 256



CNN 提取了 13x13x256 输出

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。



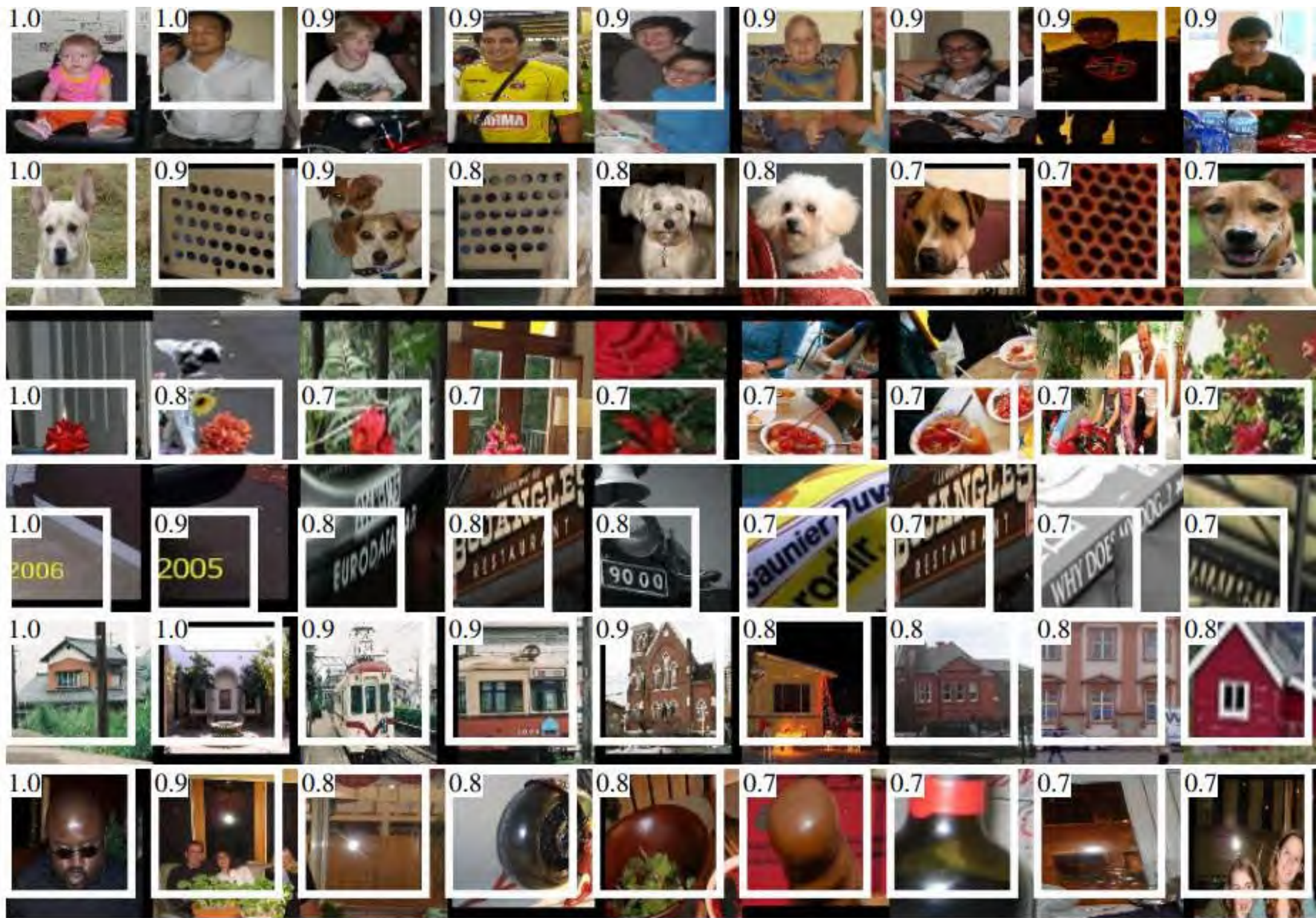
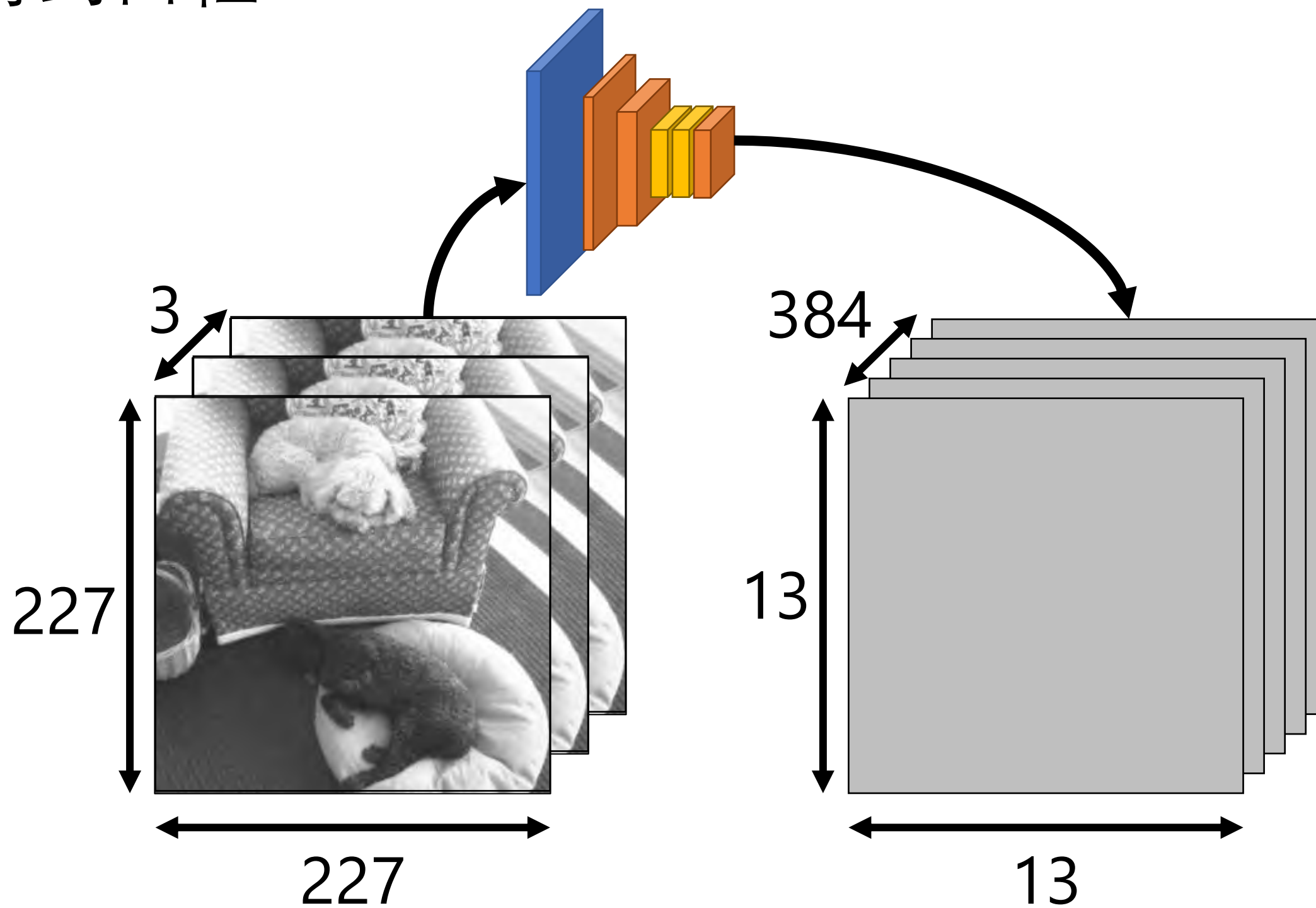
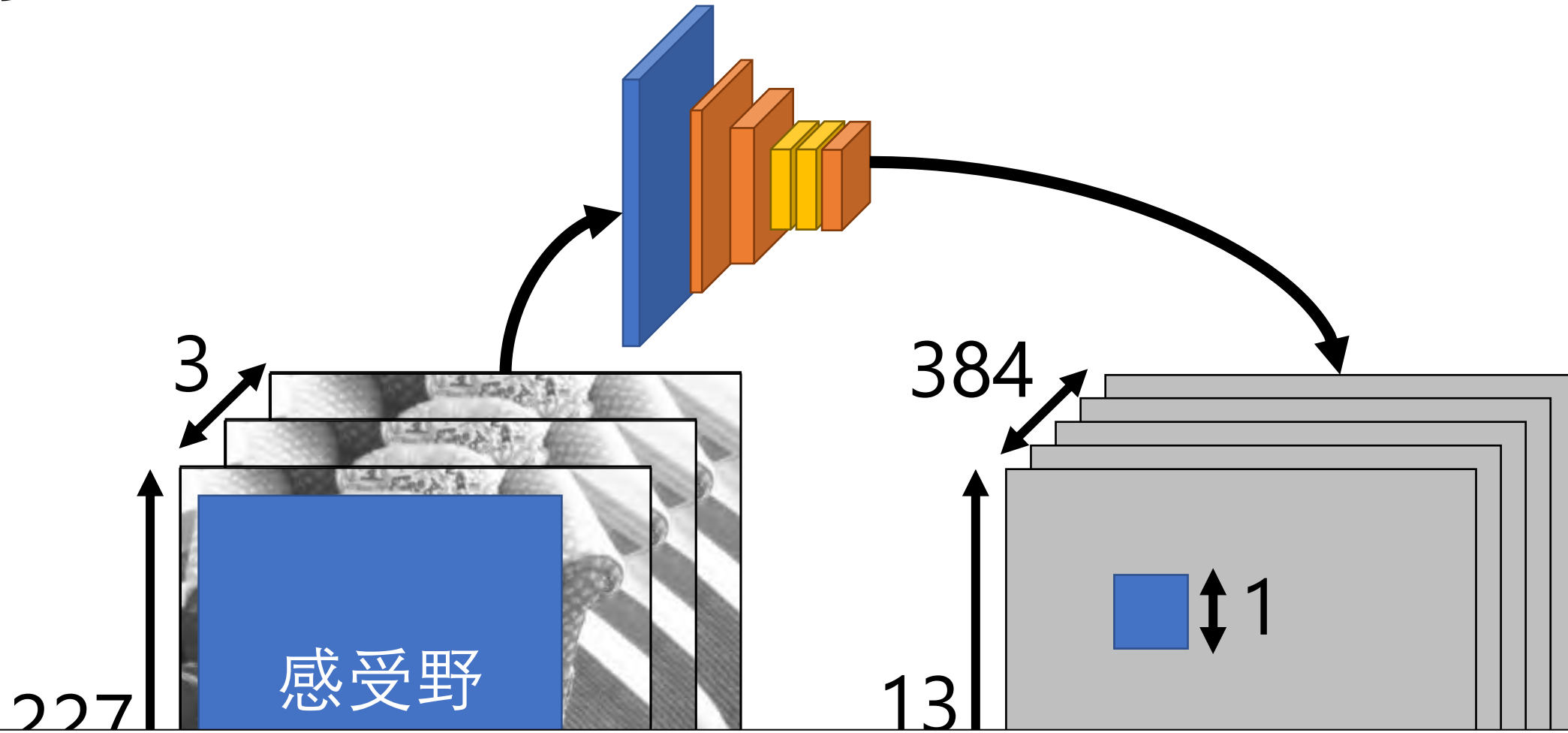


Figure Credit: Girschick et al. CVPR 2014.

怎么得到白框?

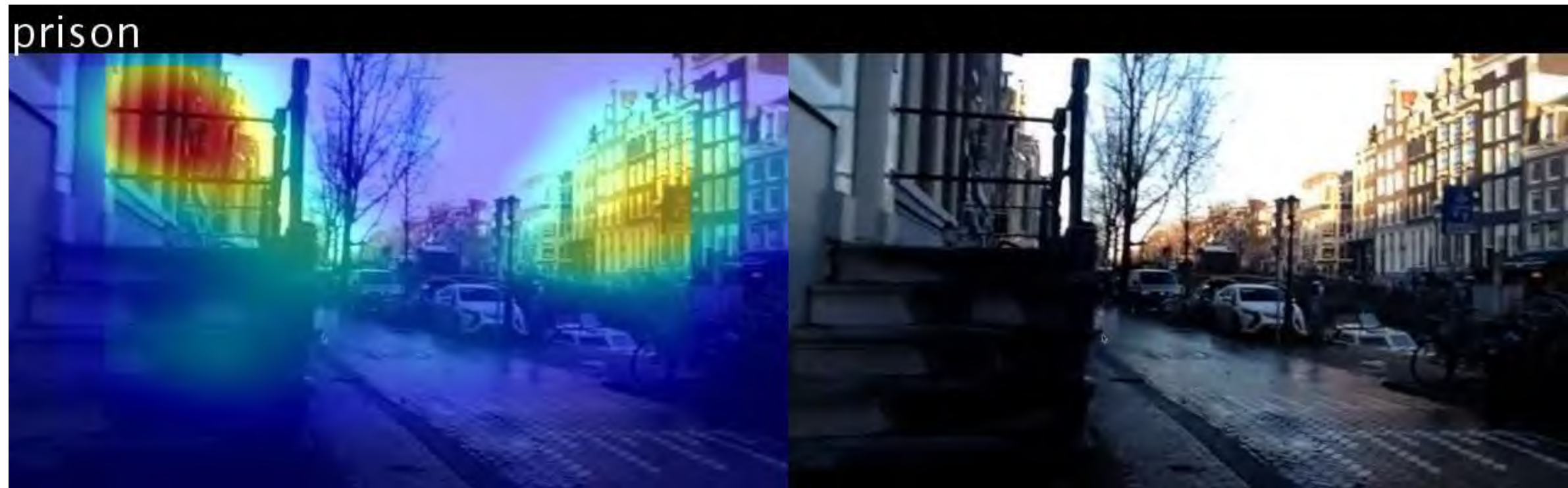


感受野



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。

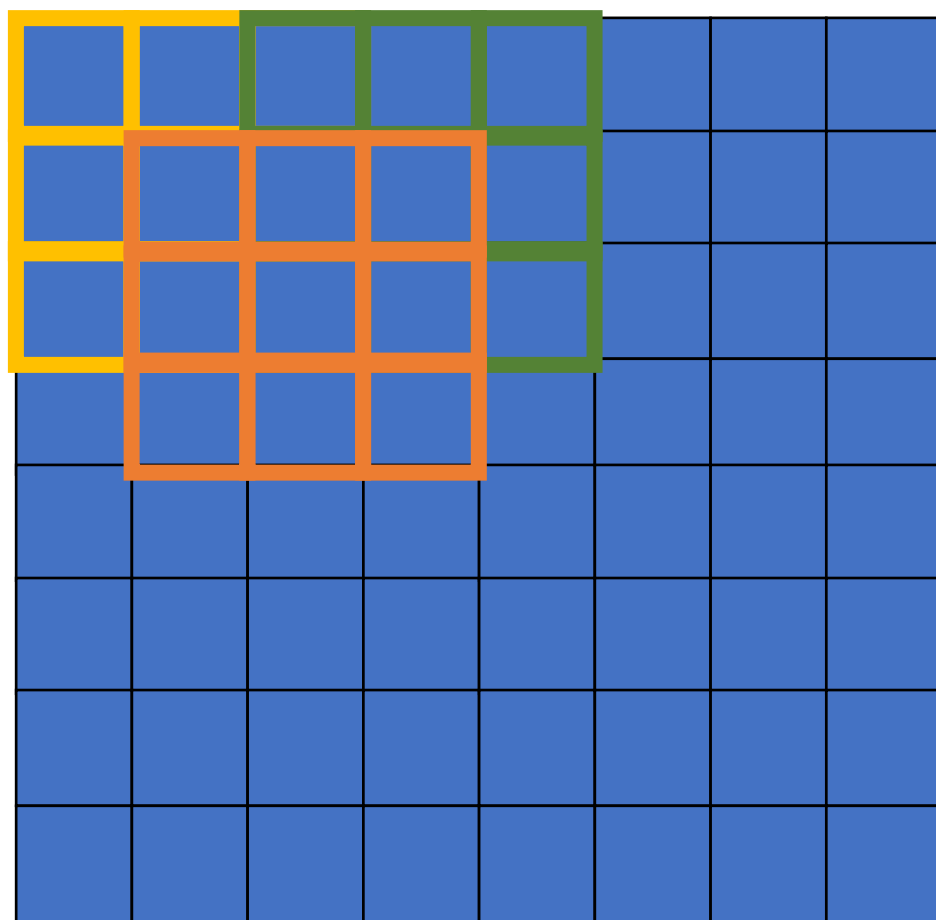
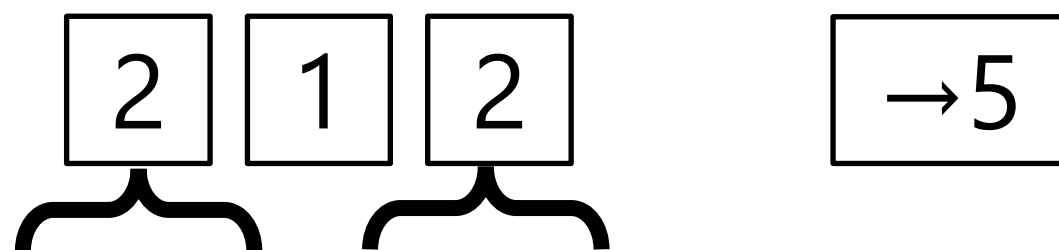
我们可以用感受野来看CNN在做决策时 关注哪些区域



3 个训练小技巧

- 3x3 Filters 3x3卷积核
- Batch Normalization 批量归一化
- Residual Learning 残差学习

3x3 滤波

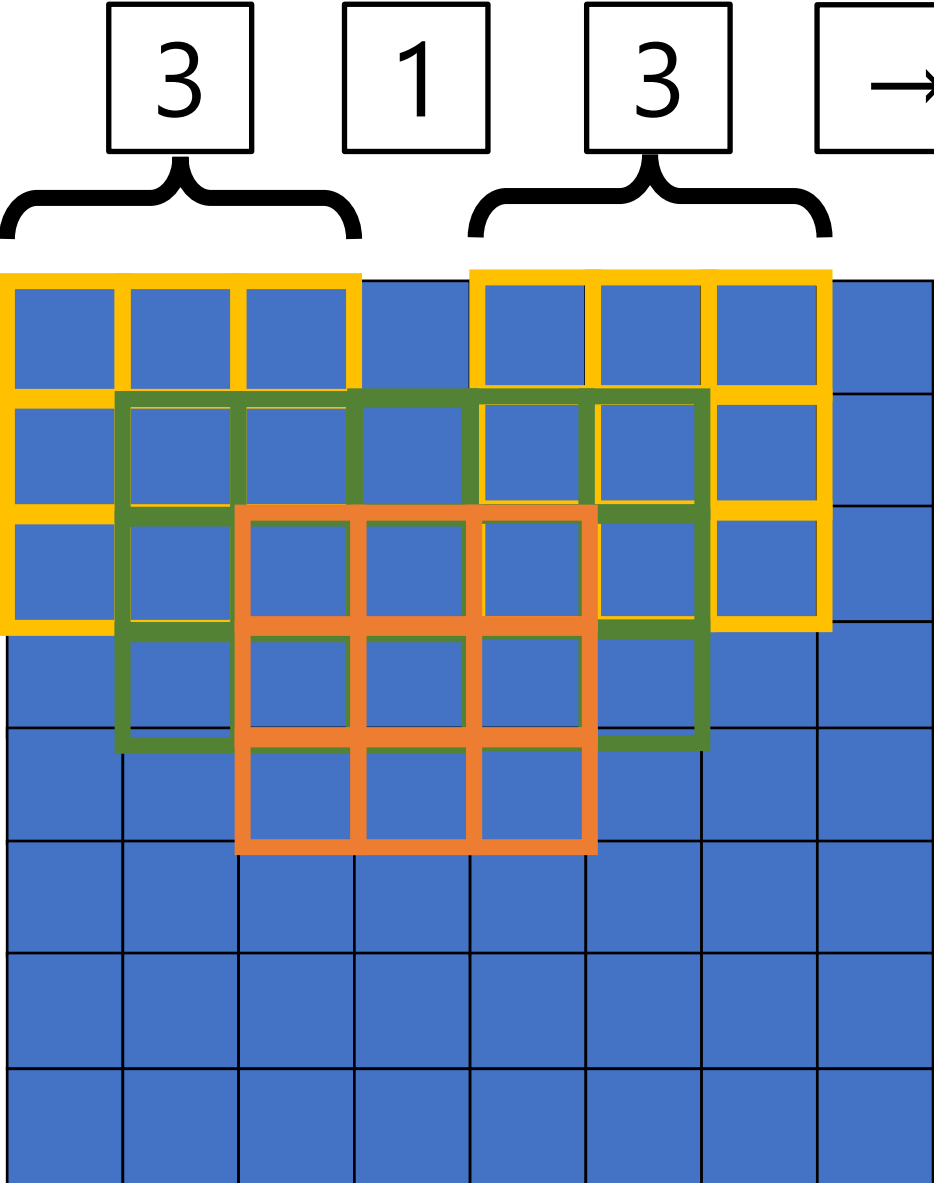


3x3 滤波 接一个
3x3 滤波

→

5x5 感受野的滤波

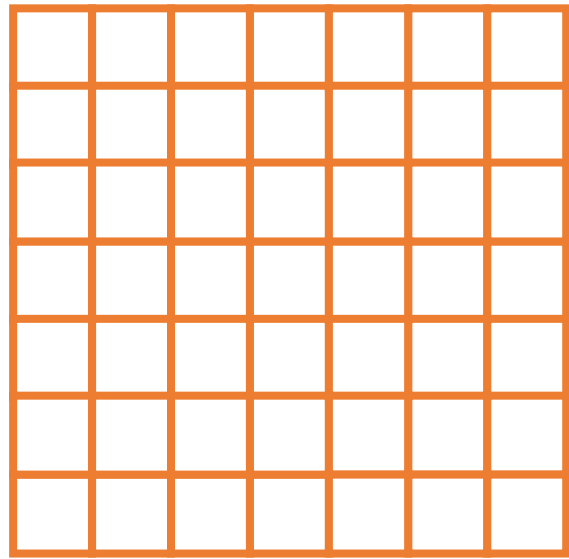
3x3 滤波



3x3 滤波 接一个
3x3 滤波 接一个
3x3 滤波

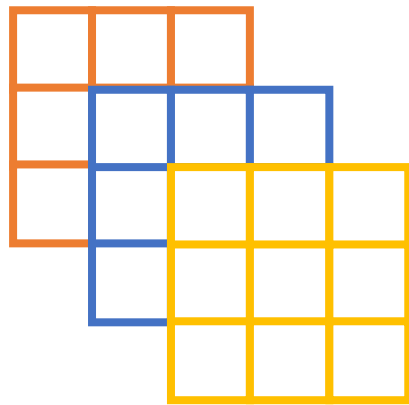
→
7x7 感受野的滤波

思考——与7x7卷积的区别？

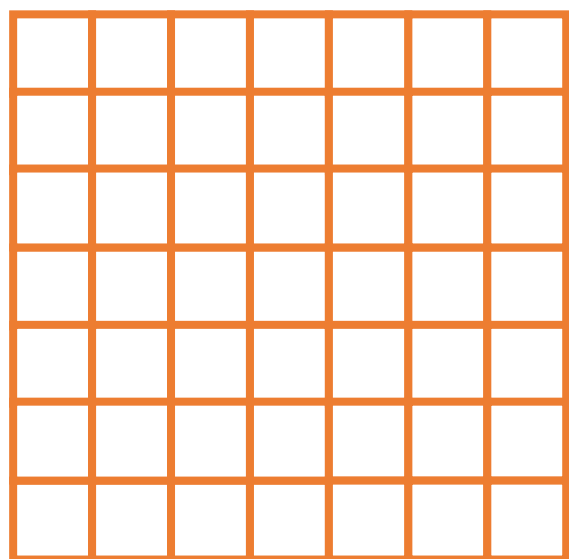


重复三次3x3 filters 一般会比直接做 a 7x7 filter 好

为什么？



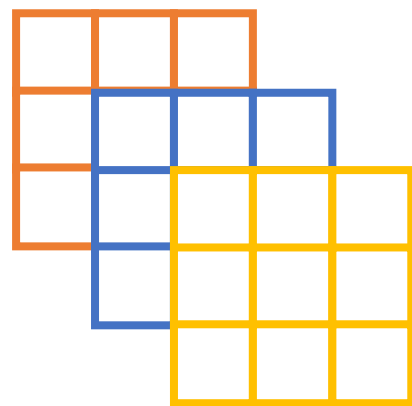
3x3 滤波



感受野: 7x7 pixels

参数量: 49

ReLU: 1



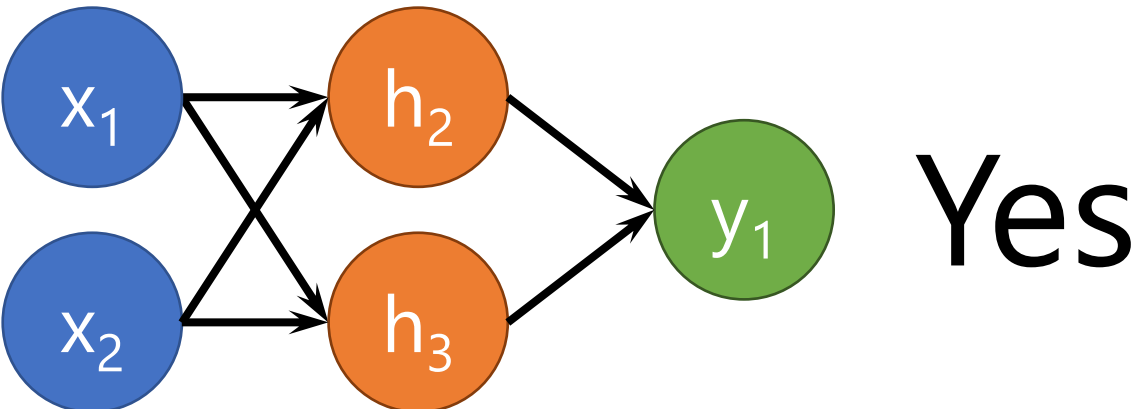
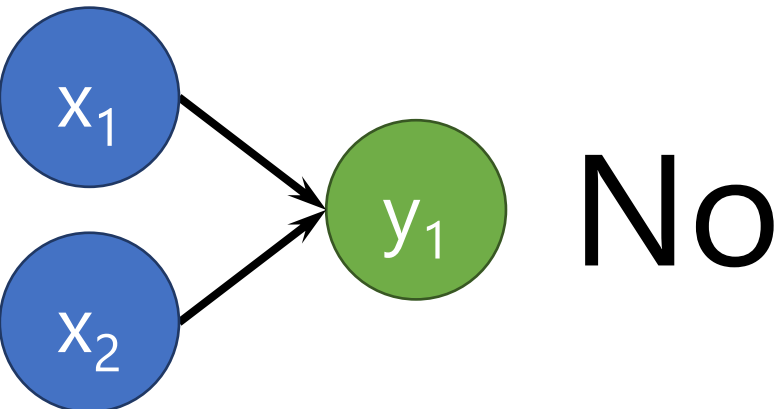
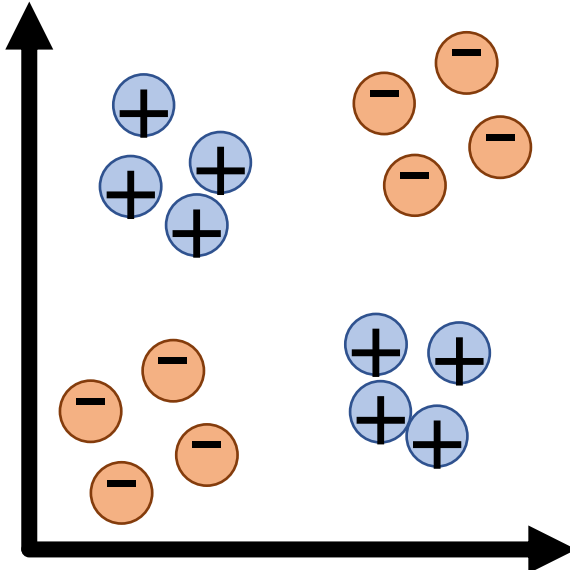
感受野: 7x7 pixels

参数量: $3 \times 3 \times 3 = \mathbf{27}$

ReLU: **3**

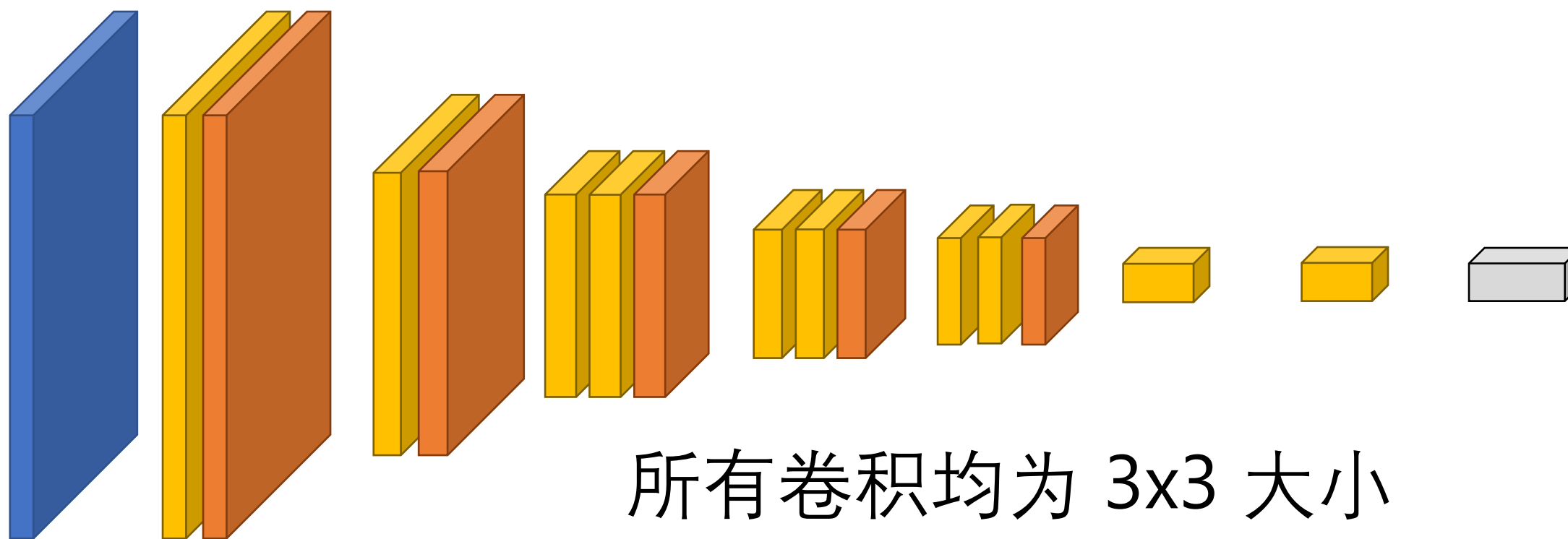
我们需要做更多的非线性层!

一个7x7可以做 XOR吗?



VGG16

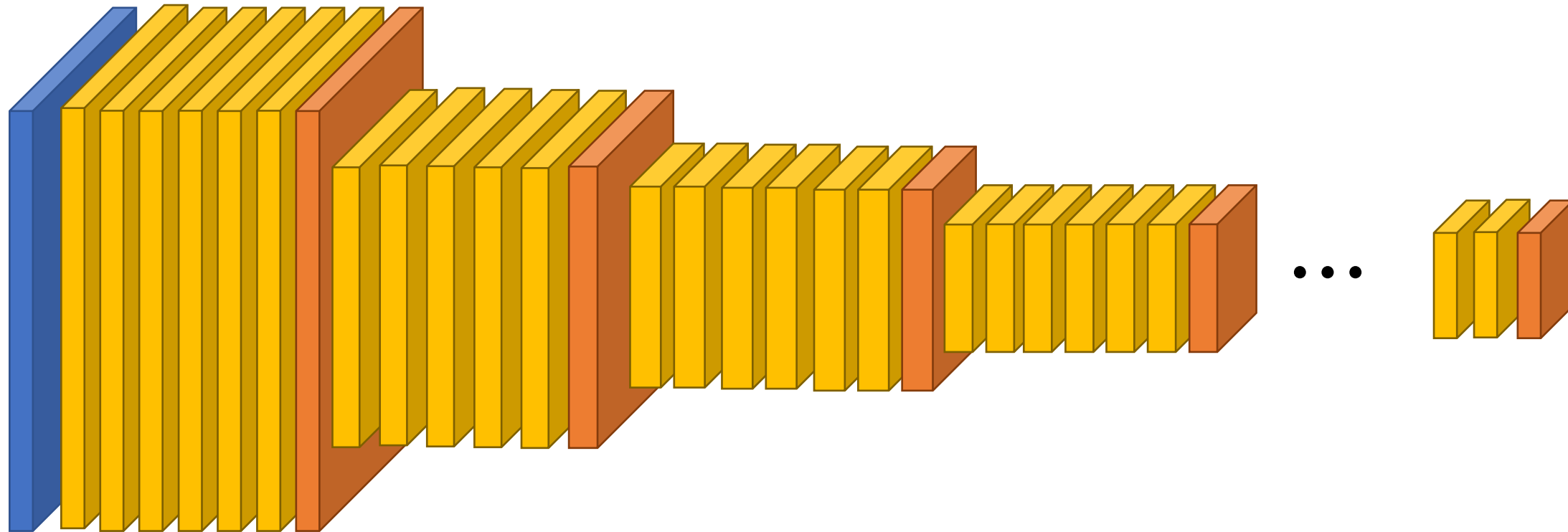
Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
224x224 3	224x224 64	112x112 128	56x56 256	28x28 512	14x14 512	1x1 4096	1x1 4096	1x1 1000



所有卷积均为 3x3 大小
所有卷积都接 ReLU

训练更深的网络

梯度反传在深层网络会出现什么问题？



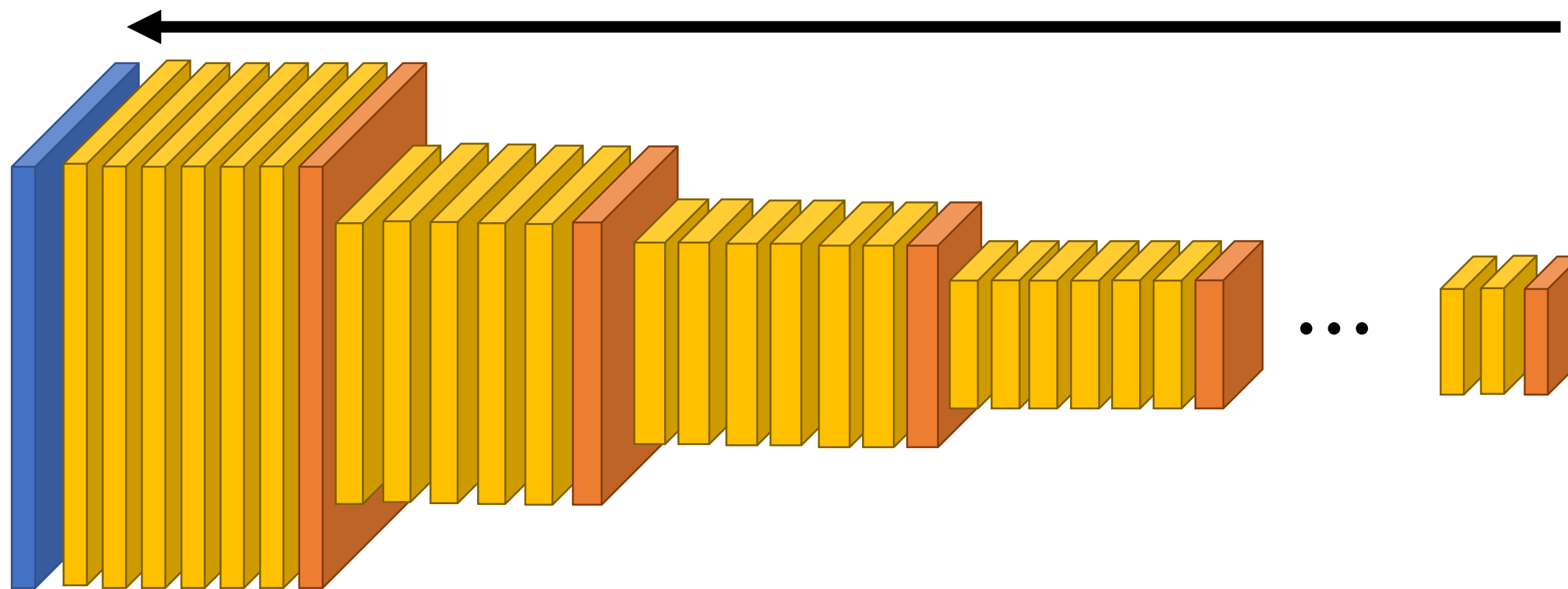
反向传播

每个反向传播操作会把局部的梯度乘上
回传的梯度

$$1 * d * d * d \dots * d = d^{n-1}$$

如果 $d \ll 1$, n 很大?

Vanishing Gradients 梯度消失



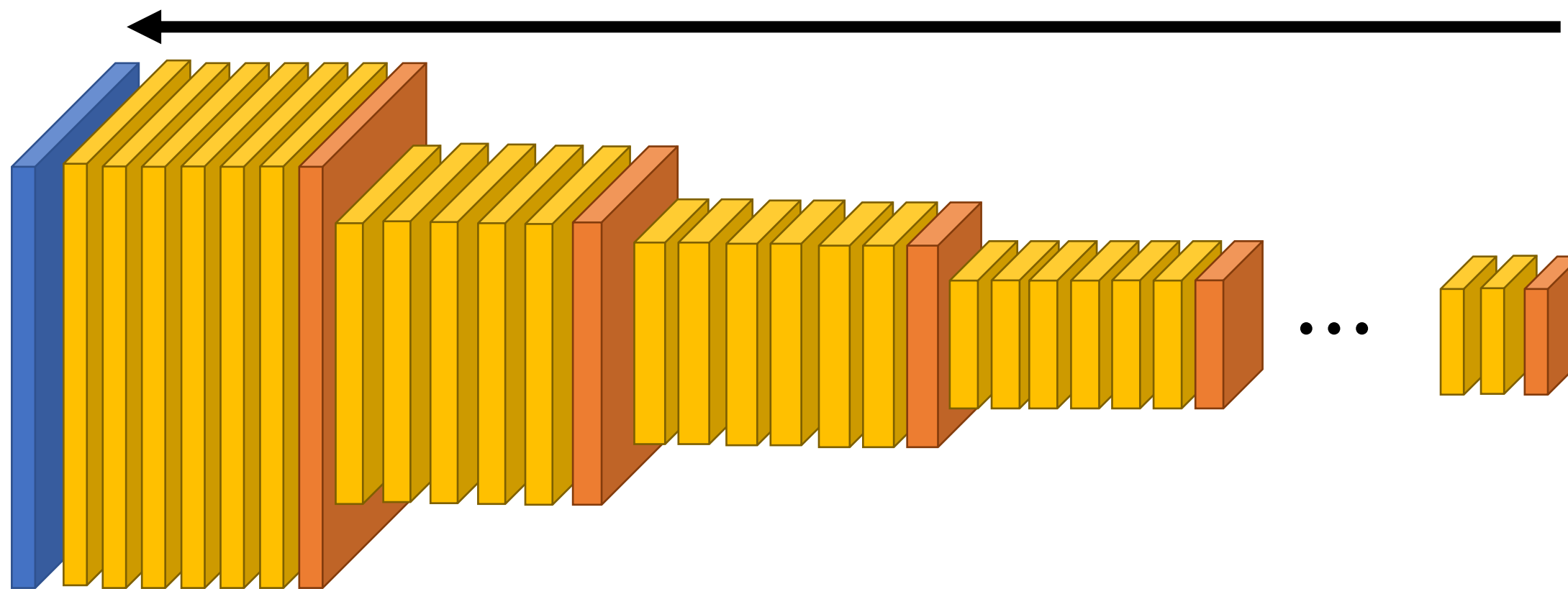
反向传播

每个反向传播操作会把局部的梯度乘上
回传的梯度

$$1 * d * d * d \dots * d = d^{n-1}$$

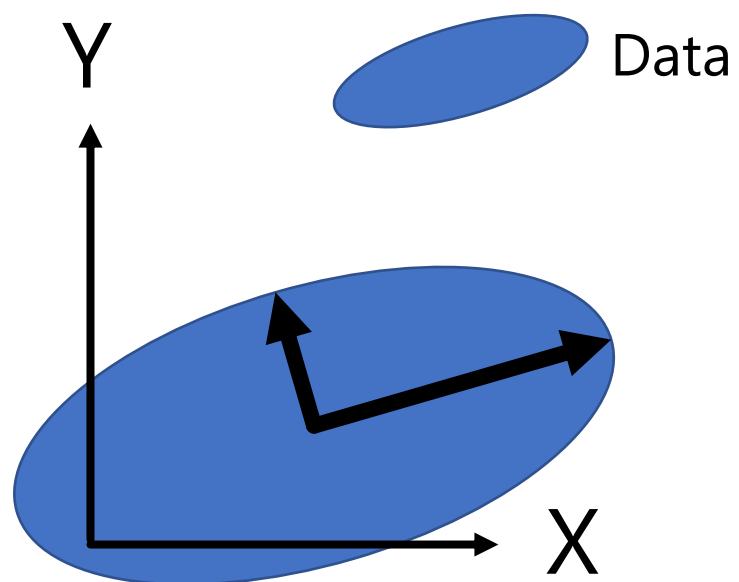
如果 $d \gg 1$, n 很大?

Exploding Gradients 梯度爆炸

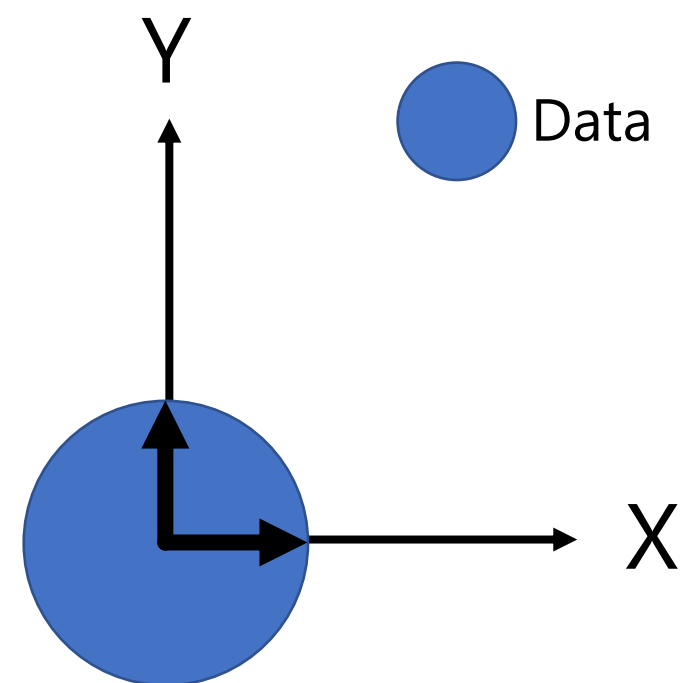


Batch Normalization

右边的情况一般会比左边的情况容易学习

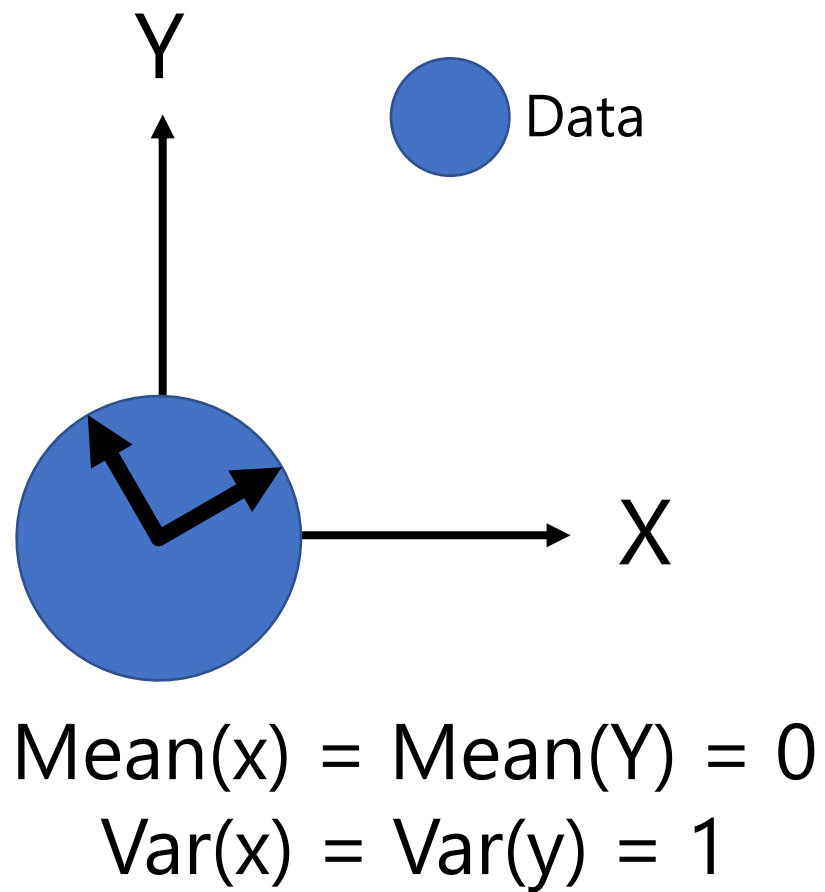


$$\begin{aligned}\text{Mean}(x) &\neq \text{Mean}(Y) \neq 0 \\ \text{Var}(x) &\neq \text{Var}(y) \neq 0 \\ \text{Cov}(x,y) &\neq 0\end{aligned}$$

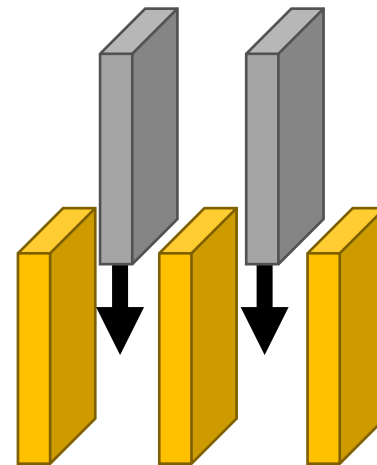


$$\begin{aligned}\text{Mean}(x) &= \text{Mean}(Y) = 0 \\ \text{Var}(x) &= \text{Var}(y) = 1 \\ \text{Cov}(x,y) &= 0\end{aligned}$$

Batch Normalization

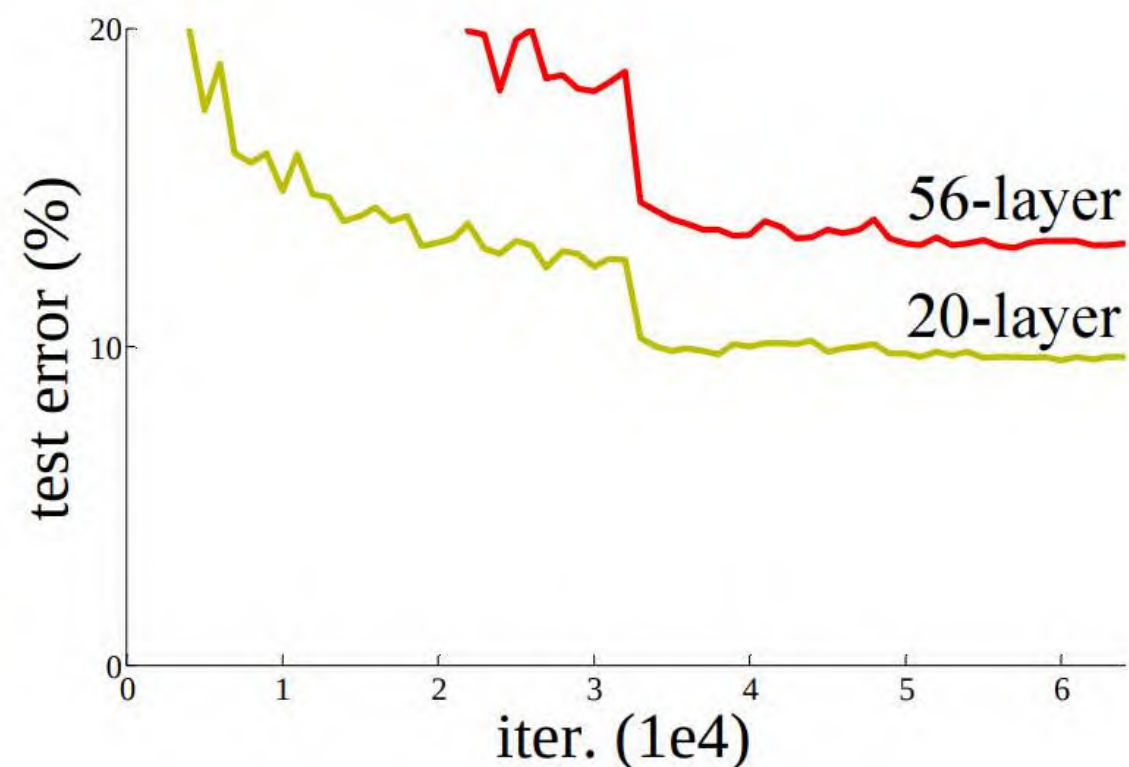
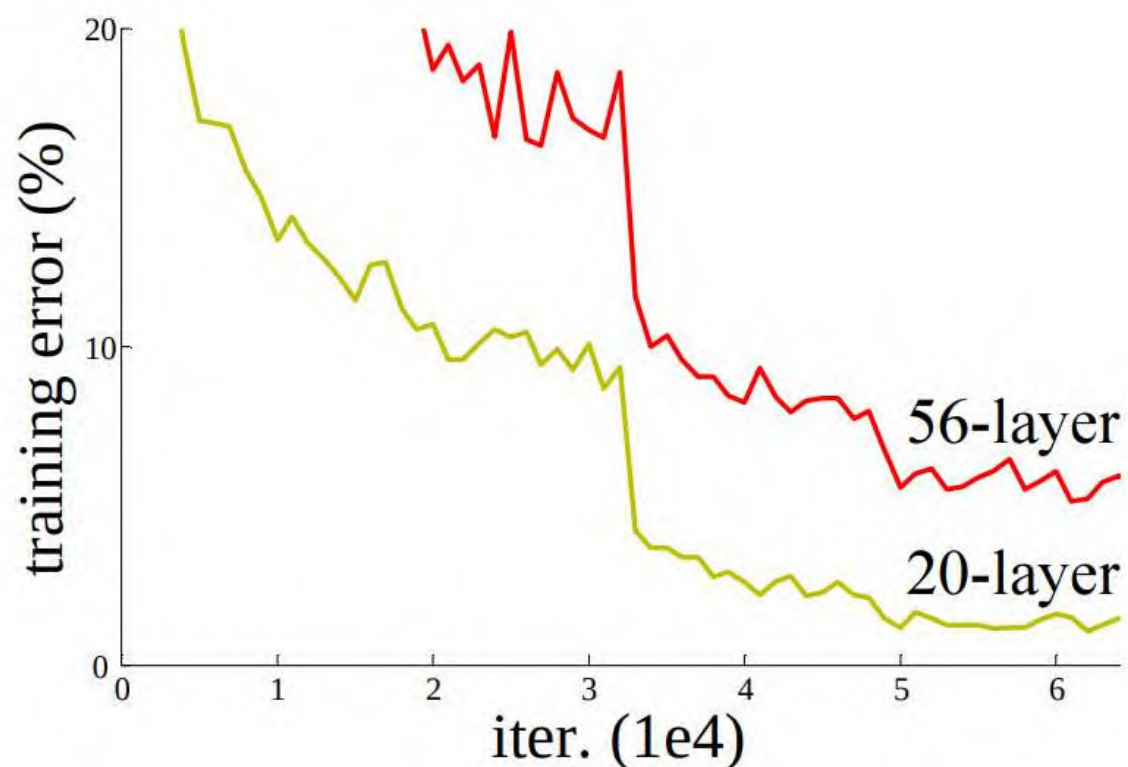


Idea: (**Batch Norm**)将批量归一化作为单独层插入到网络中的想法, 这个层会对经过它的数据进行归一化, 基于每个批次中数据的方差估计值.



现象

- **更深的模型** 在训练集上的拟合能力比 **更浅的模型** 更差.
- 深度网络可能更难训练。虽然存在更深的模型理论上能够拟合数据，但实际上我们可能找不到合适的参数来训练它。

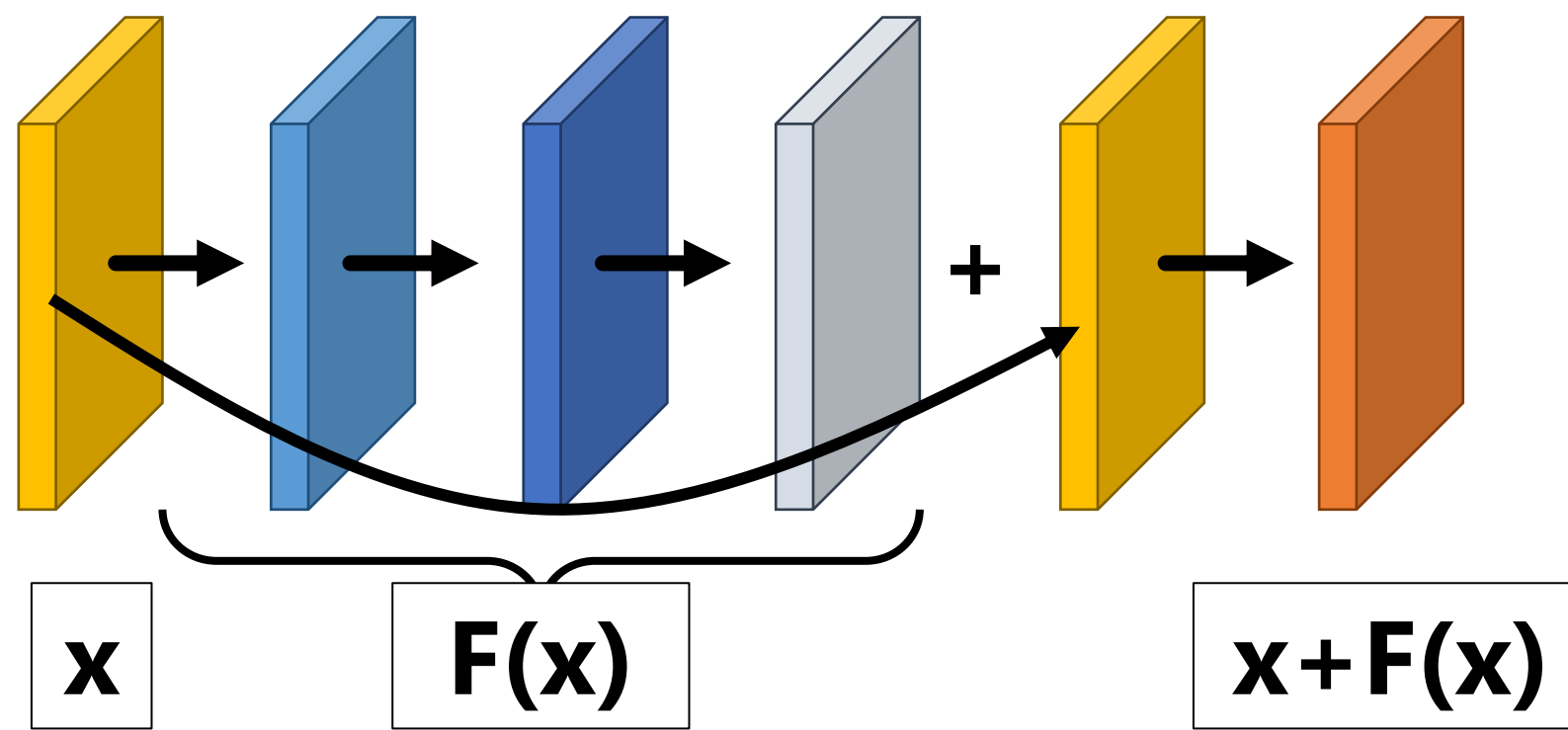


Residual Learning 残差学习

新的模块:

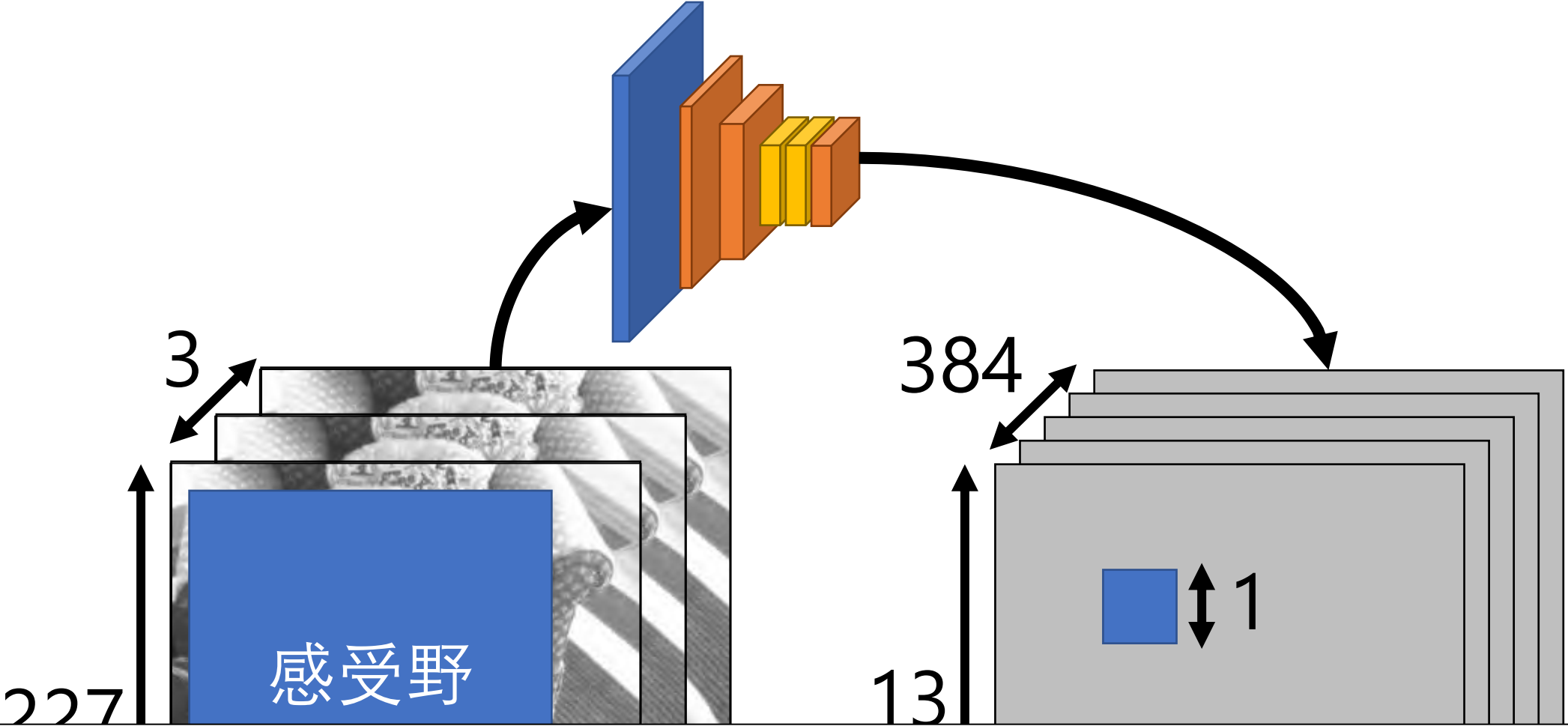
$$x + F(x)$$

让你轻松训练100层的网络.



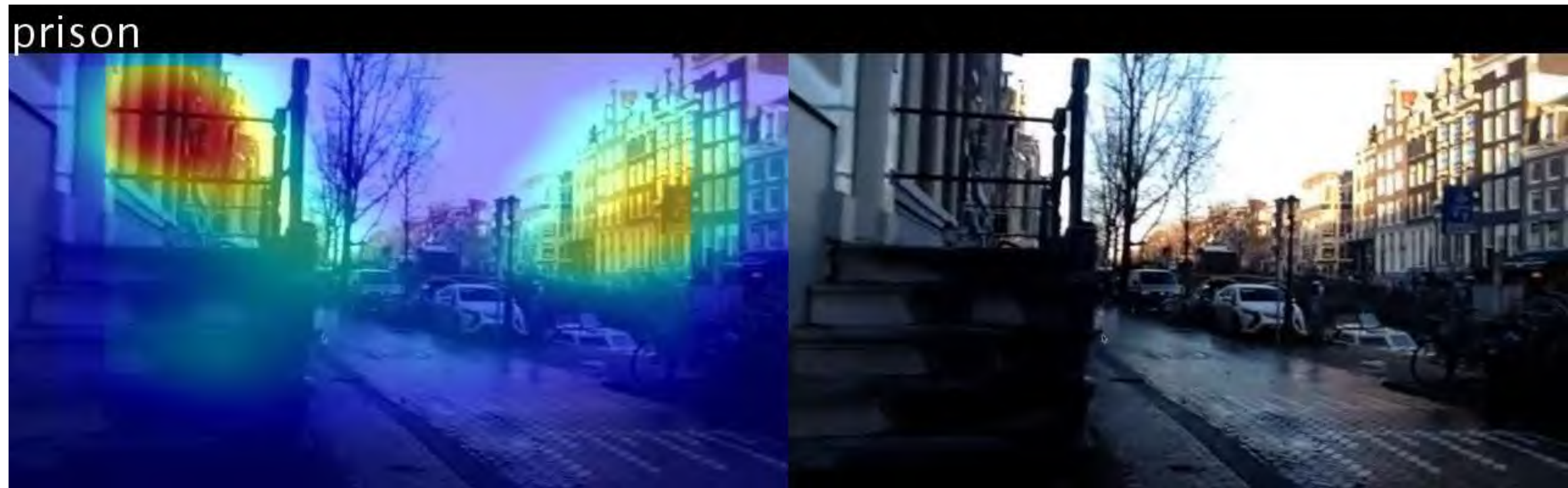
回顾——卷积神经网络

CNN可视化——感受野

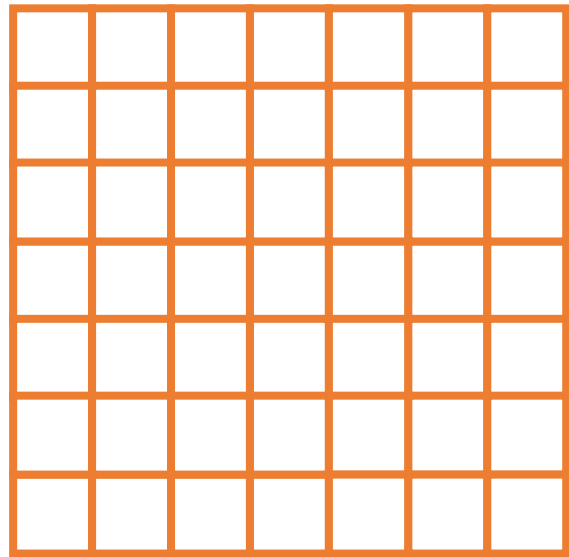


每一个输出特征图的单个像素实际上是由输入图像中的一个小区域转换而来的。卷积层不会一次性看到整个图像，而是通过滤波器窗口看到图像的一小部分，这个窗口逐步移动覆盖整个图像，从而在特征图上生成每一个单独的像素点。

我们可以用感受野来看CNN在做决策时 关注哪些区域



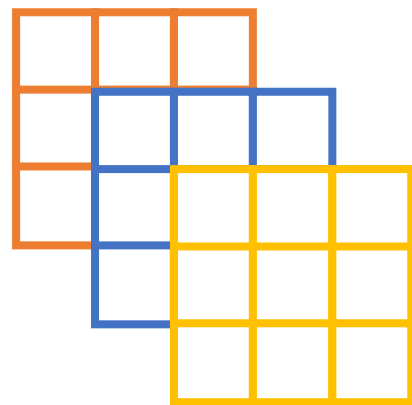
CNN设计——3x3 滤波



感受野: 7x7 pixels

参数量: 49

ReLU: 1



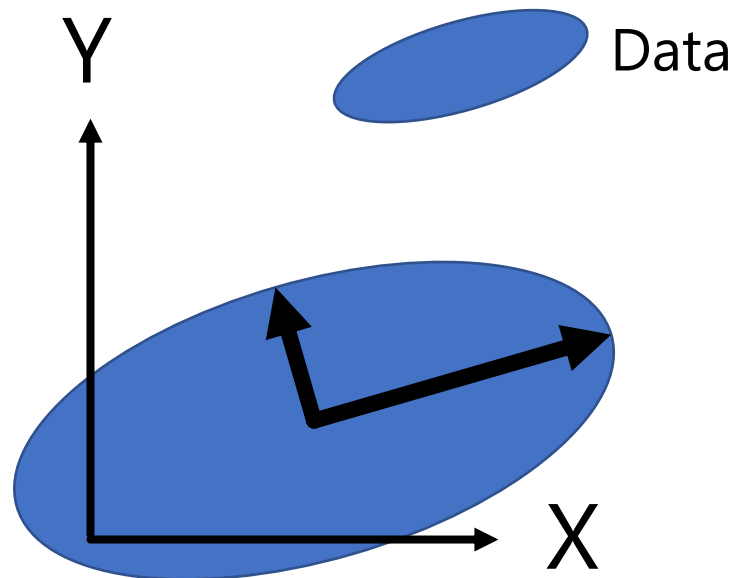
感受野: 7x7 pixels

参数量: $3 \times 3 \times 3 = \mathbf{27}$

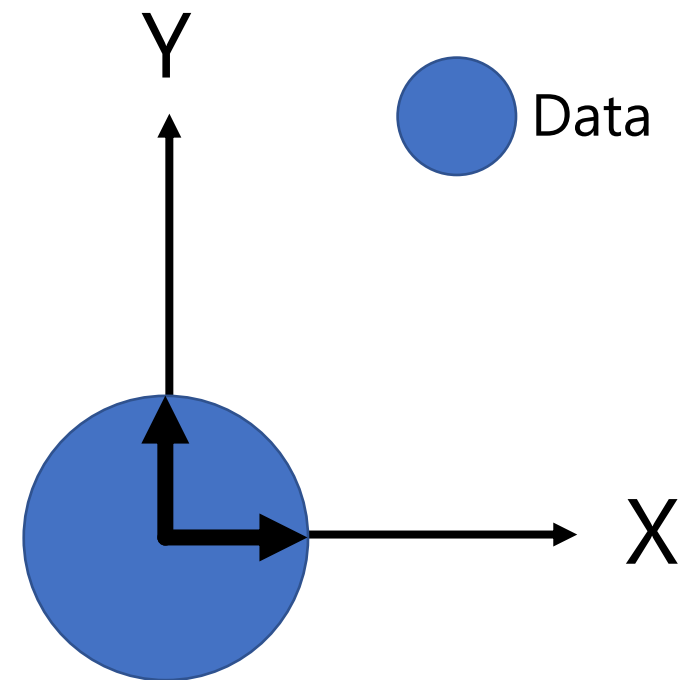
ReLU: **3**

CNN设计——Batch Normalization

右边的情况一般会比左边的情况容易学习



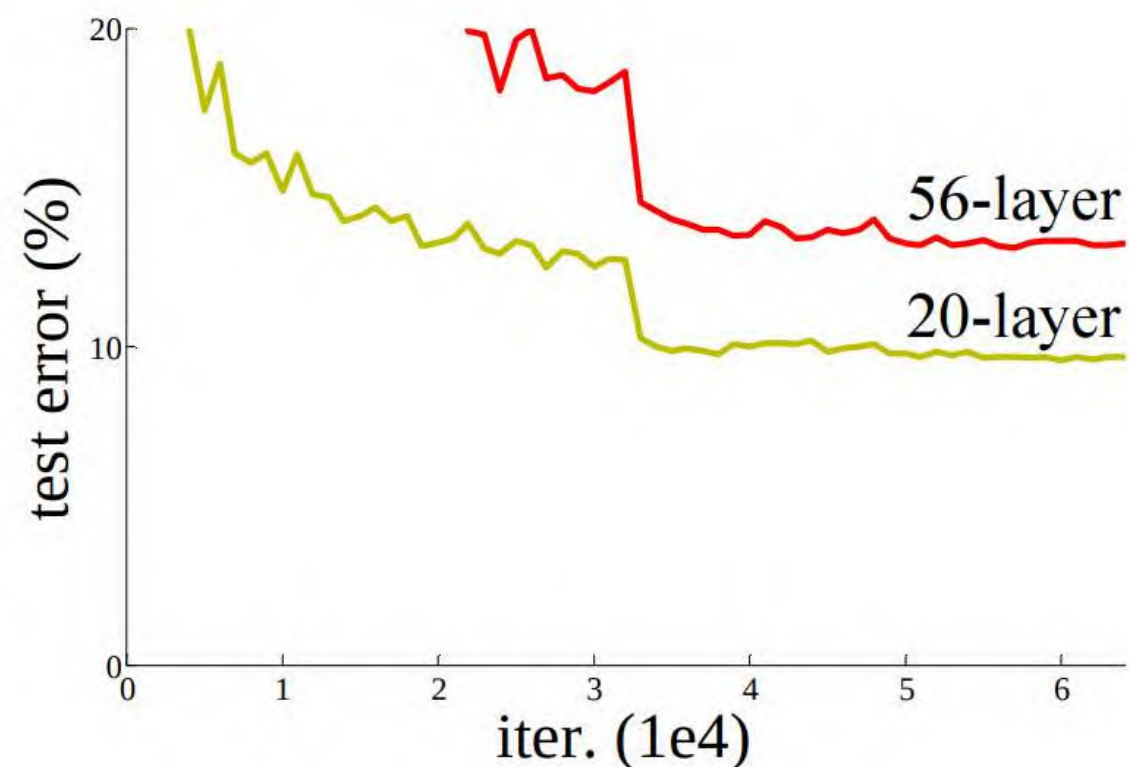
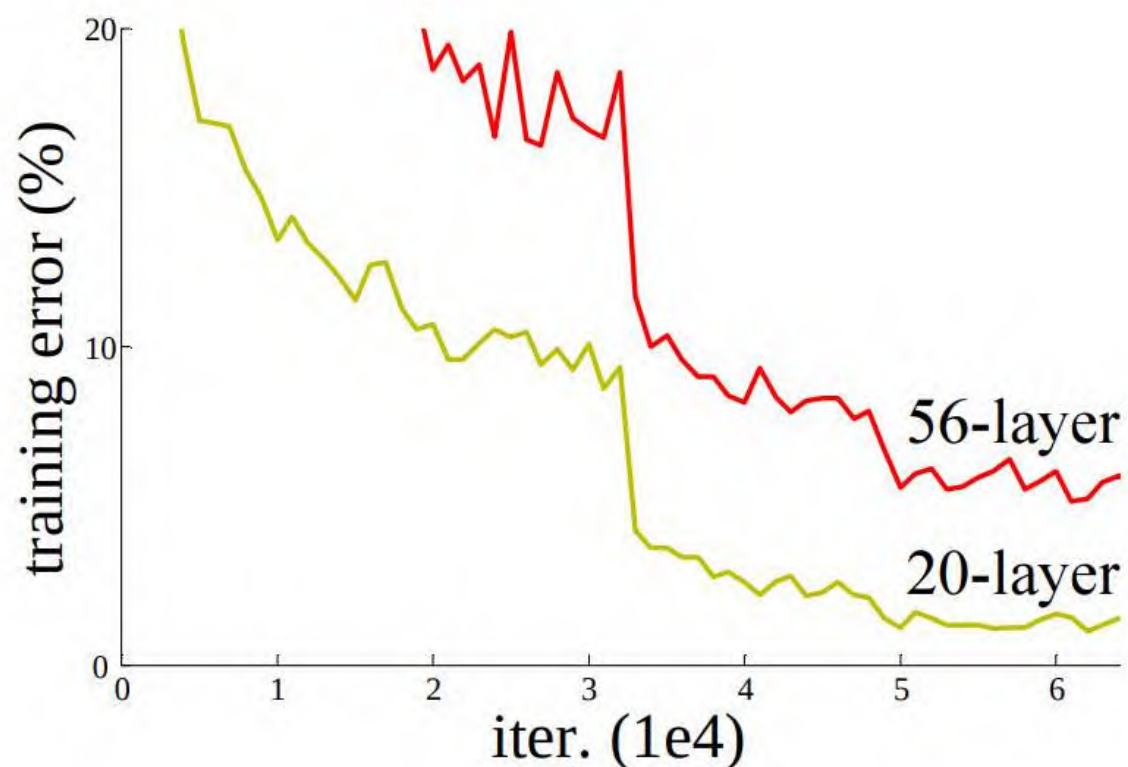
$$\begin{aligned}\text{Mean}(x) &\neq \text{Mean}(Y) \neq 0 \\ \text{Var}(x) &\neq \text{Var}(y) \neq 0 \\ \text{Cov}(x,y) &\neq 0\end{aligned}$$



$$\begin{aligned}\text{Mean}(x) &= \text{Mean}(Y) = 0 \\ \text{Var}(x) &= \text{Var}(y) = 1 \\ \text{Cov}(x,y) &= 0\end{aligned}$$

CNN设计——残差

- **更深的模型** 在训练集上的拟合能力比 **更浅的模型** 更差.
- 深度网络可能更难训练。虽然存在更深的模型理论上能够拟合数据，但实际上我们可能找不到合适的参数来训练它。

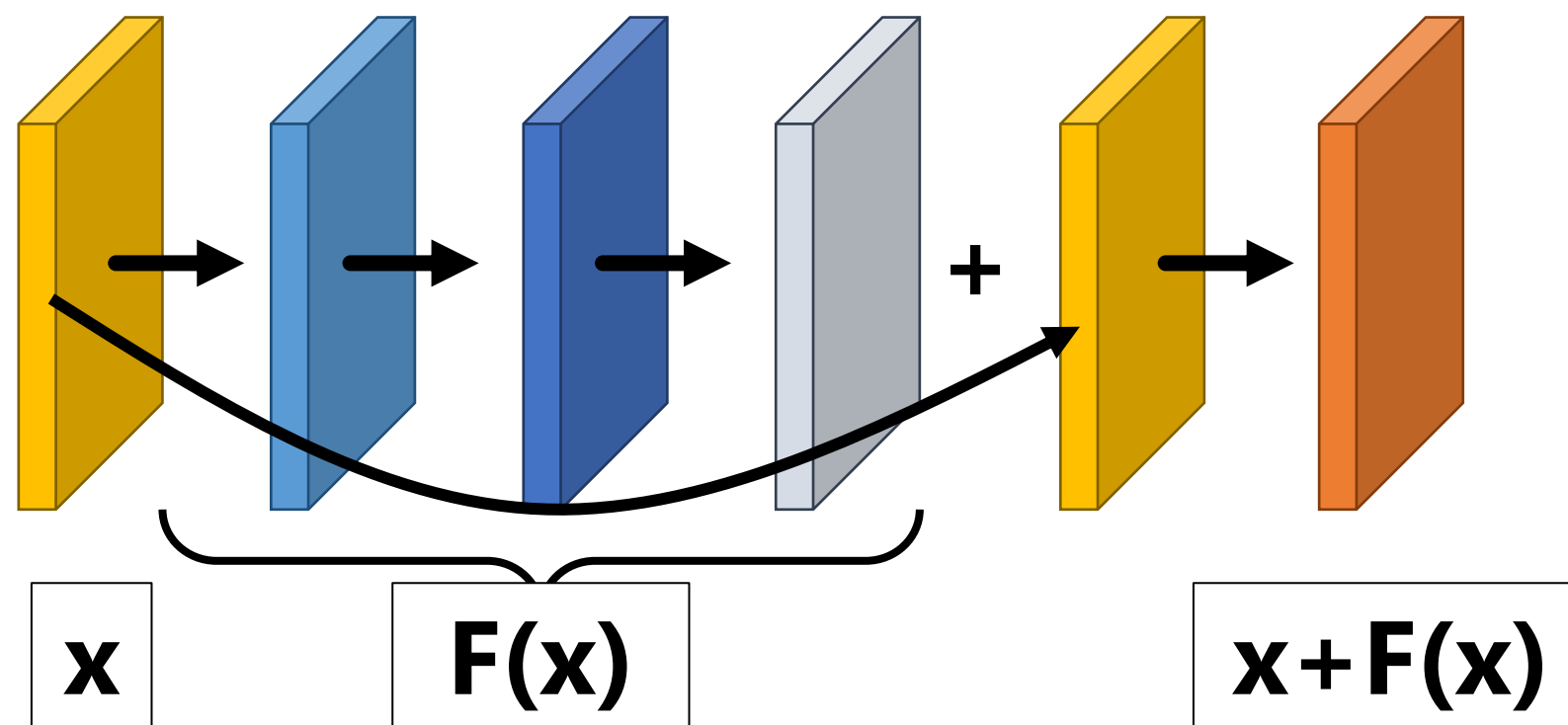


CNN设计——残差学习

新的模块:

$$x + F(x)$$

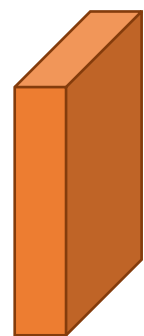
让你轻松训练100层的网络.



特征转化为向量 —— Pooling

HxWxC

1x1xF



1	1	2	4
5	6	7	8
3	2	1	0
1	1	3	4

Avg Pool
HxW Filter
Stride 1

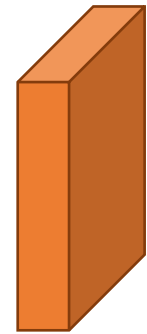


3.1

*(If F == C)

特征转化为向量——卷积

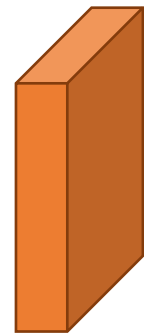
$H \times W \times C$



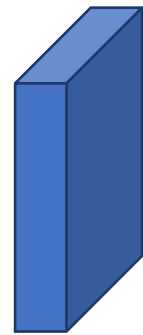
$1 \times 1 \times F$



$H \times W$ F 个滤波的卷积



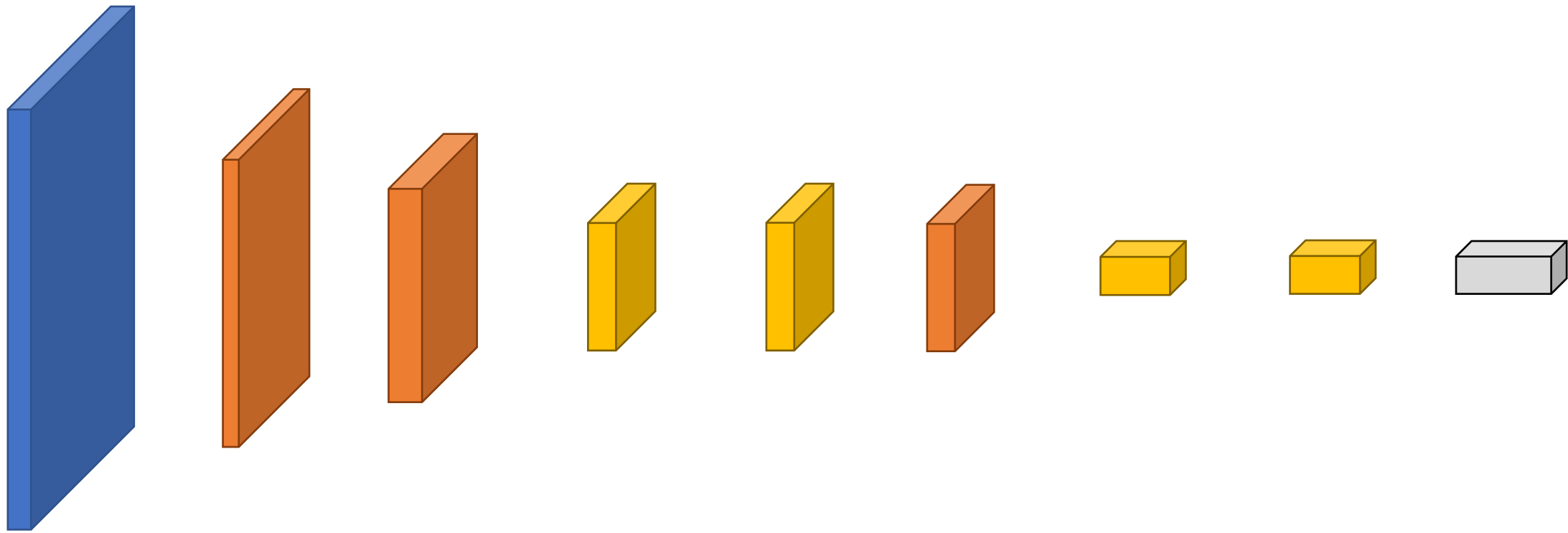
*



每个滤波只
输出一个值

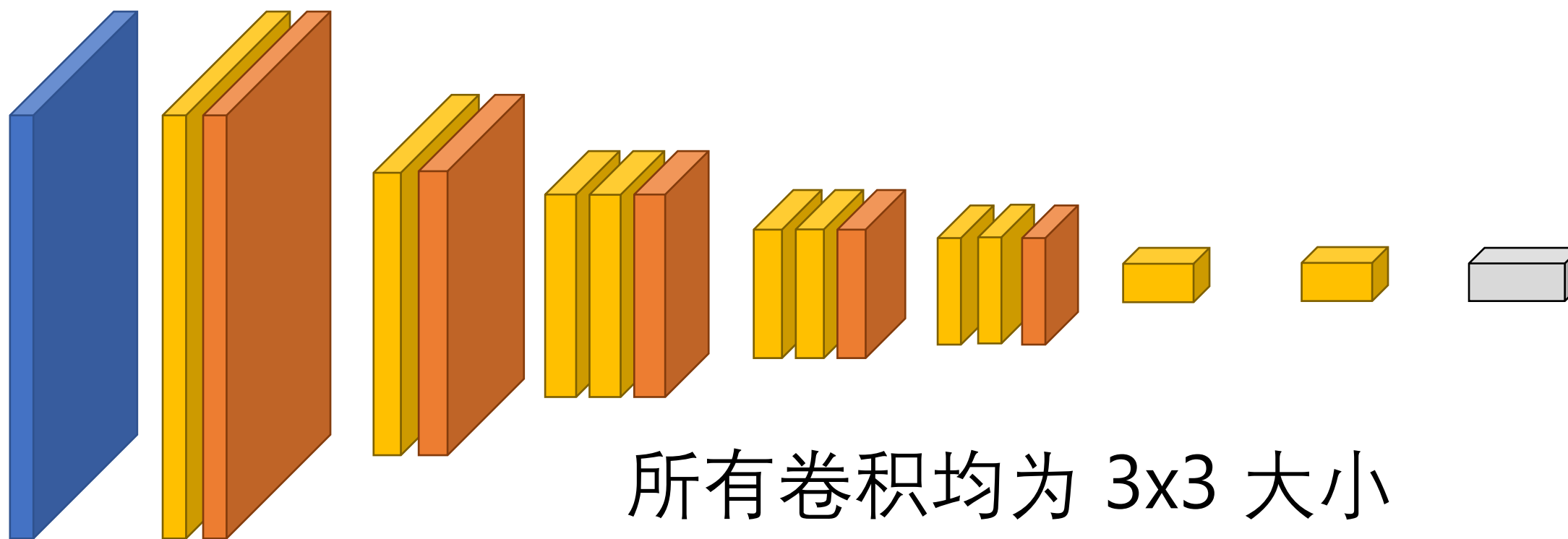
AlexNet

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



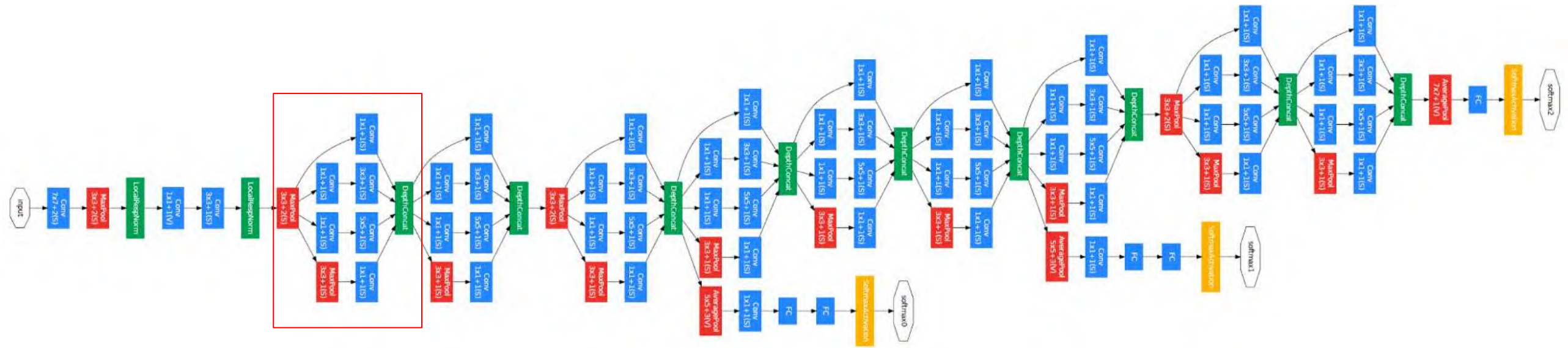
VGG16

Input	Conv 1	Conv 2	Conv 3	Conv 4	Conv 5	FC 6	FC 7	Output
224x224 3	224x224 64	112x112 128	56x56 256	28x28 512	14x14 512	1x1 4096	1x1 4096	1x1 1000

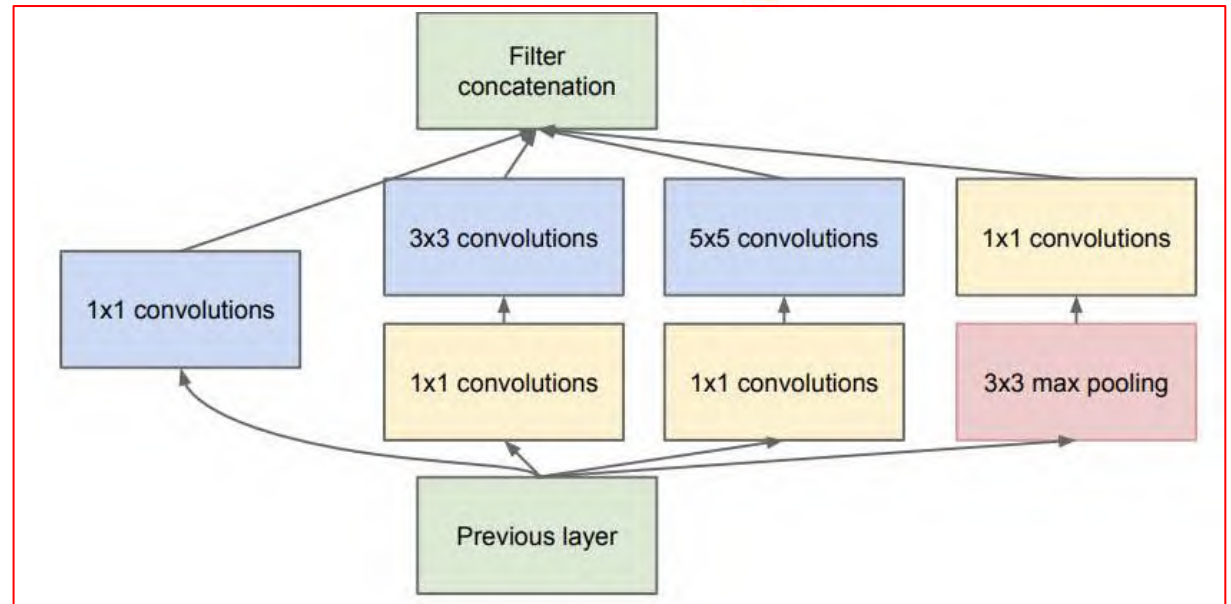


所有卷积均为 3x3 大小
所有卷积都接 ReLU

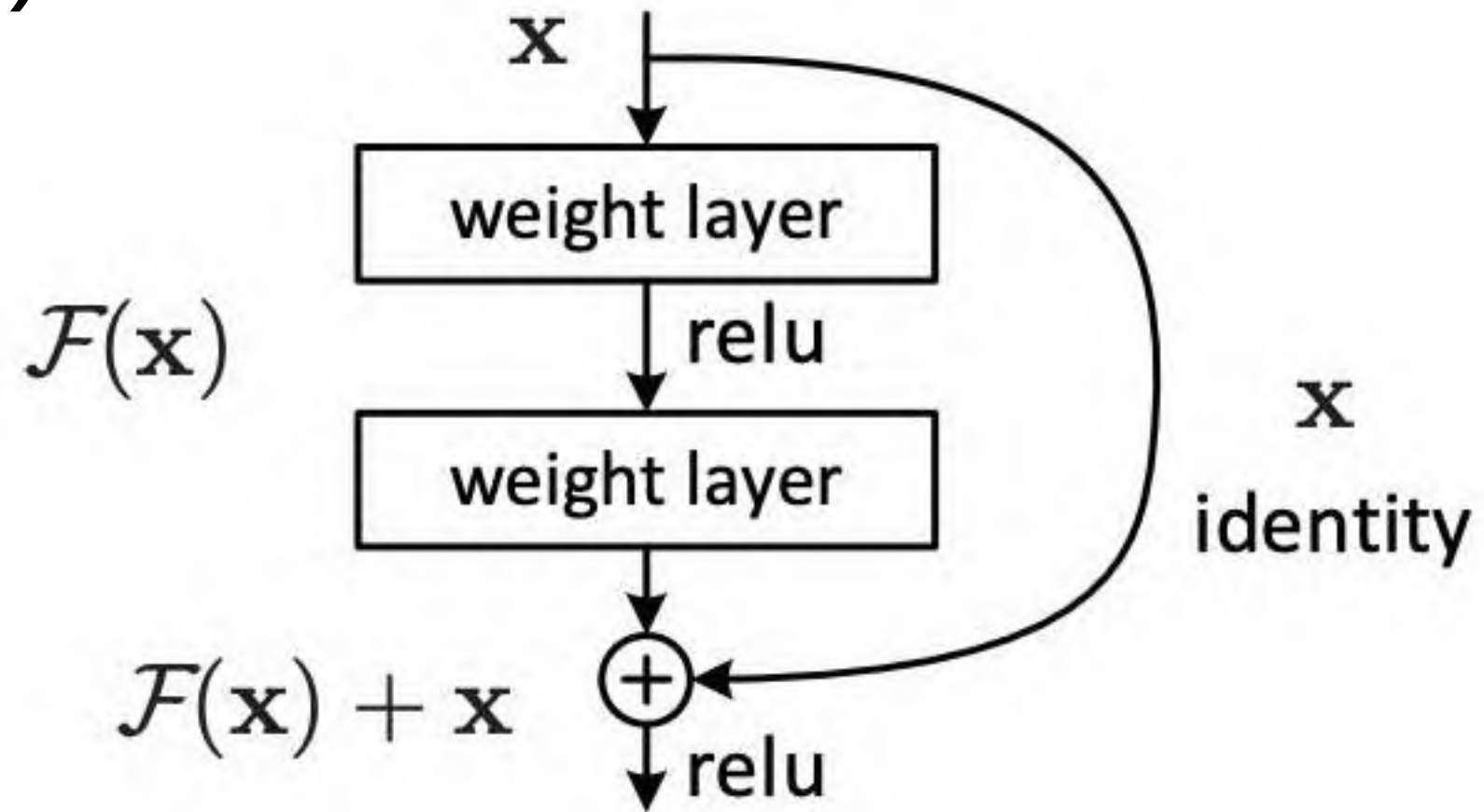
Inception Nets (2014+)



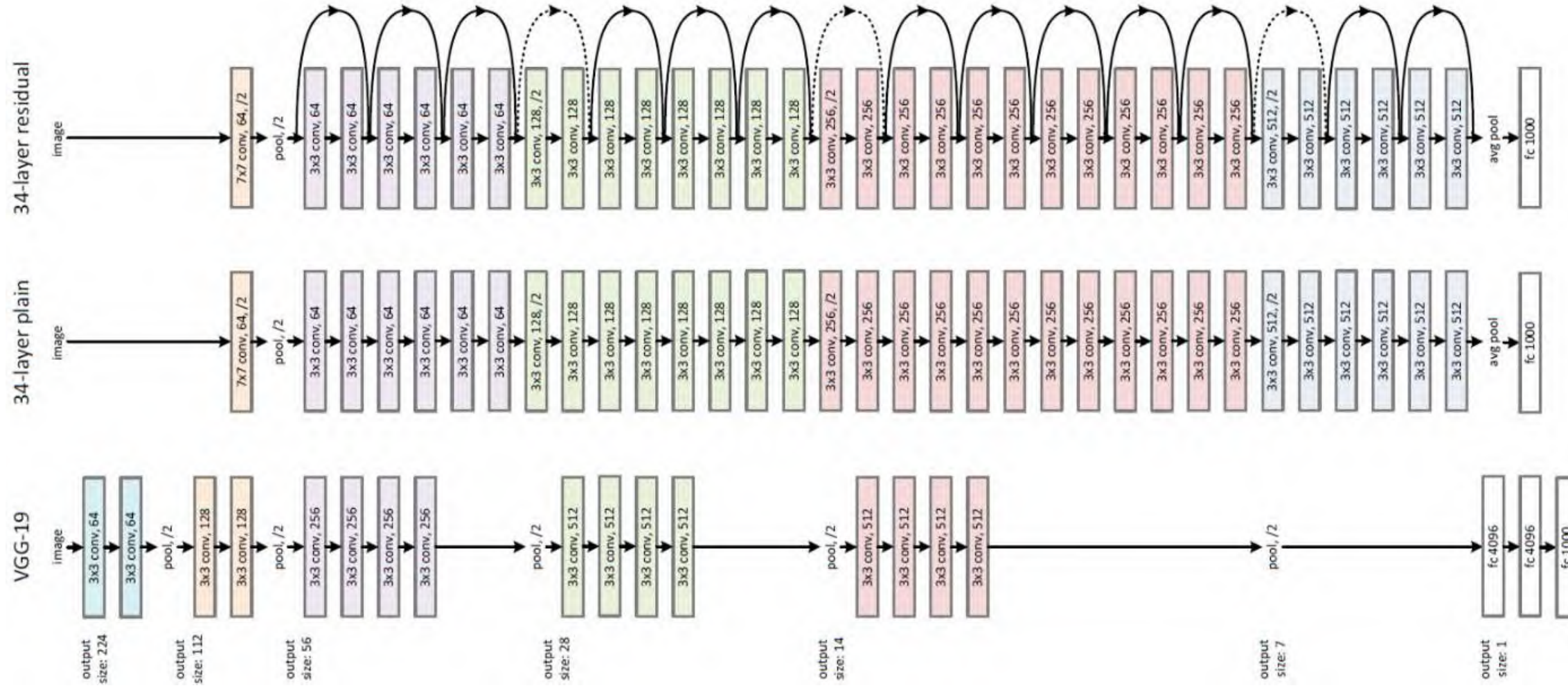
各种卷积组合



ResNet (2015)

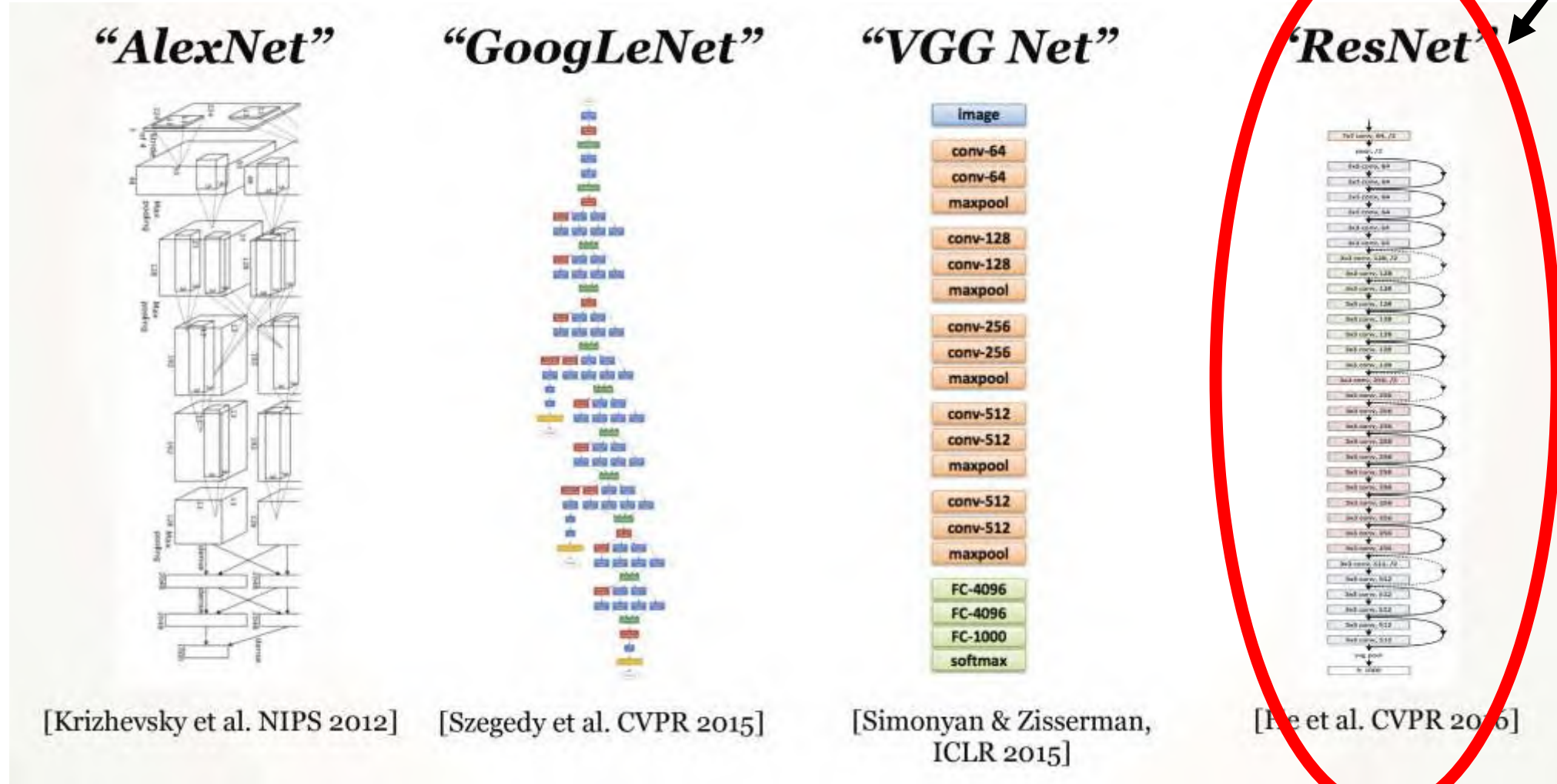


ResNet (2015)

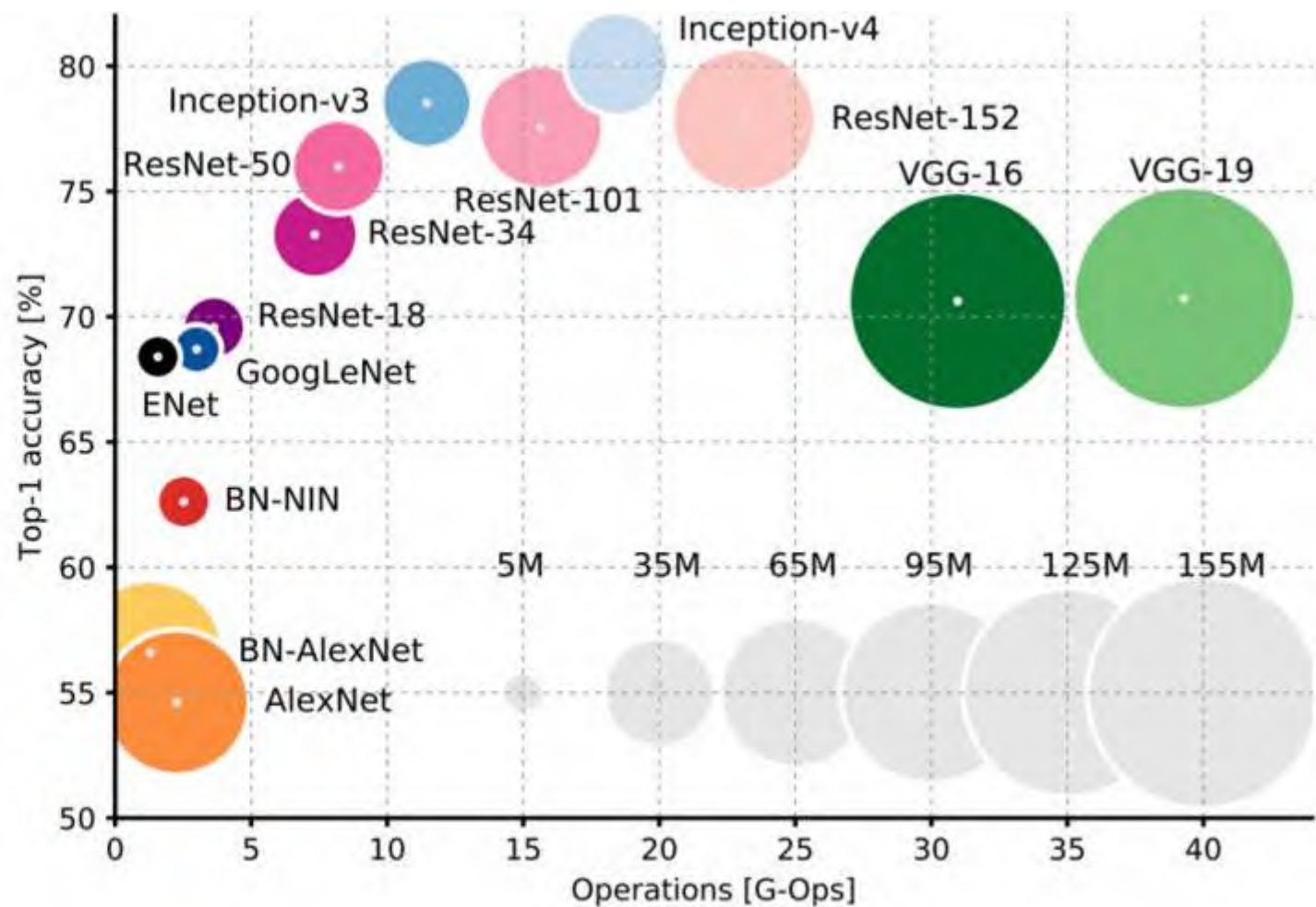
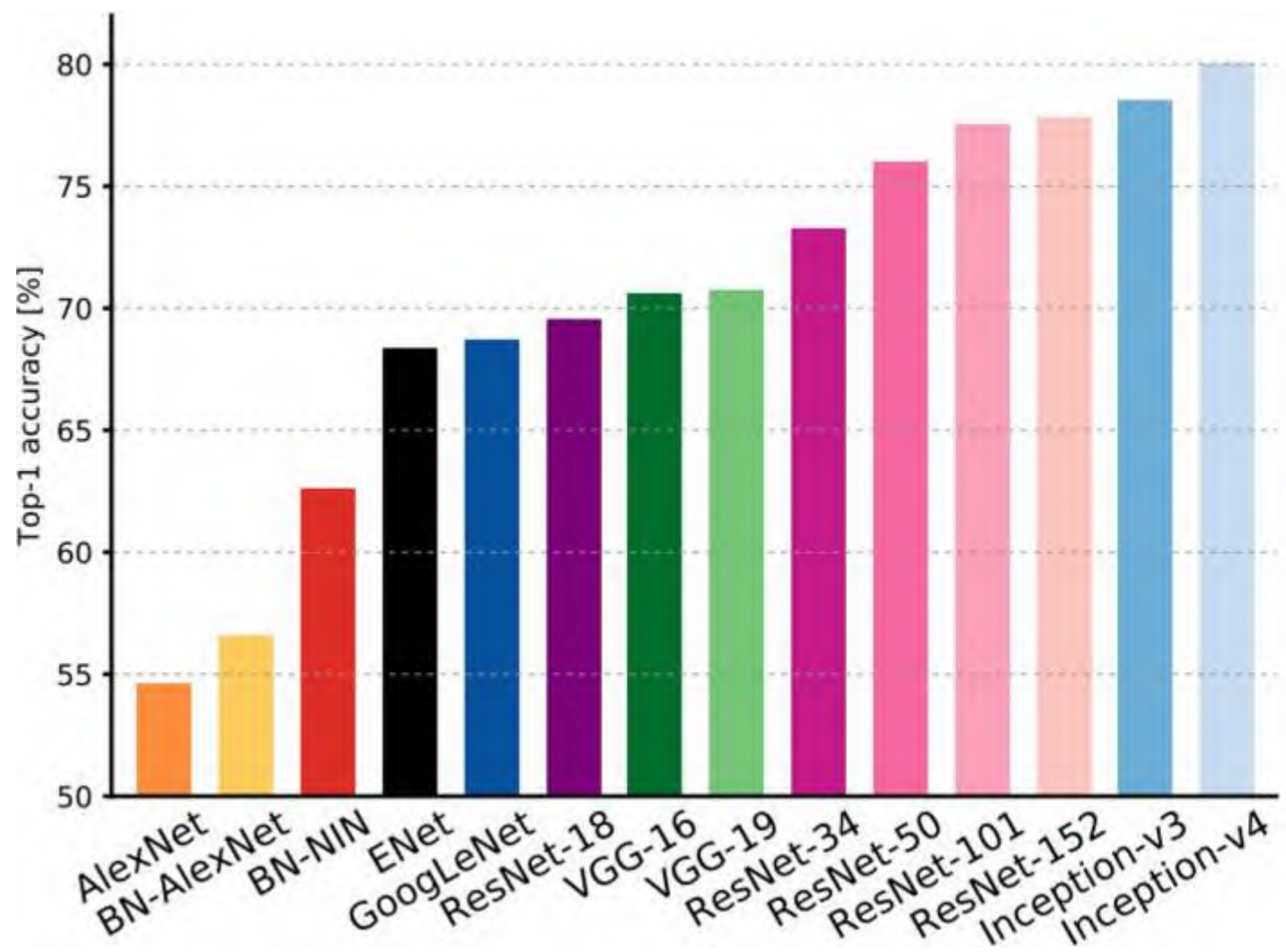


CNN网络结构

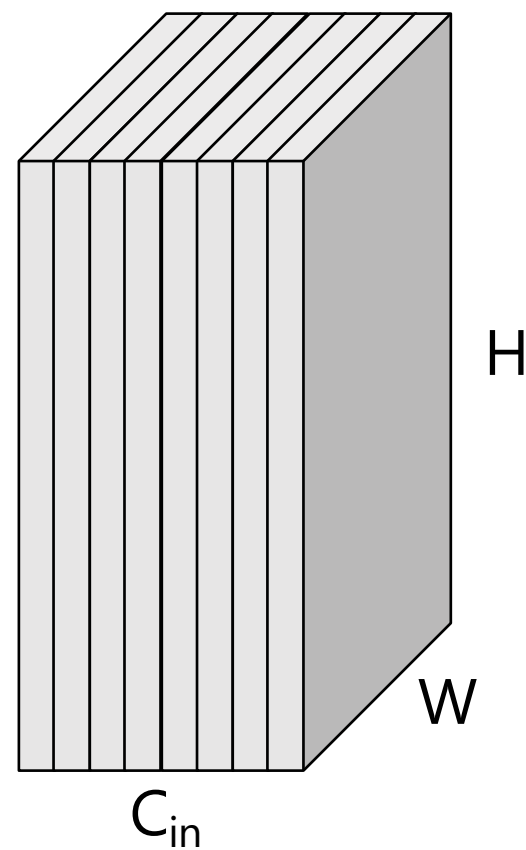
最常用的选择



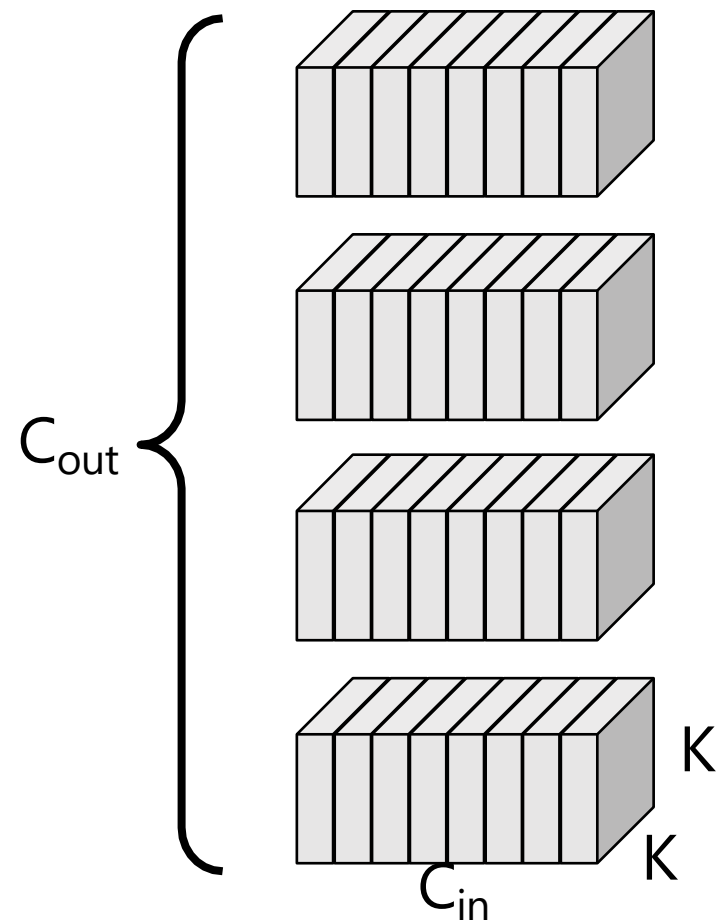
模型复杂度



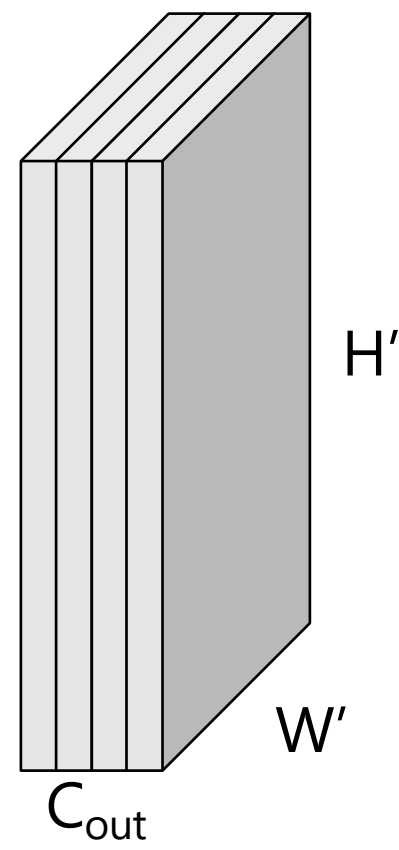
卷积层



输入: $C_{in} \times H \times W$

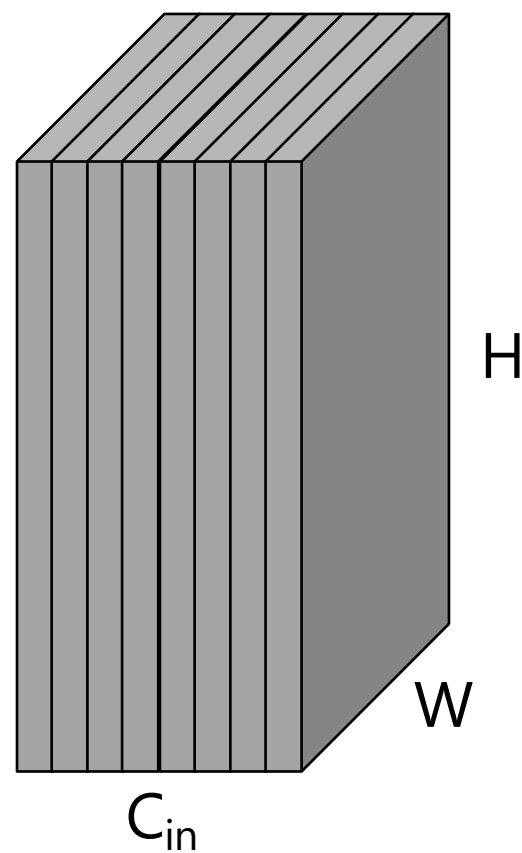


权重: $C_{out} \times C_{in} \times H \times W$

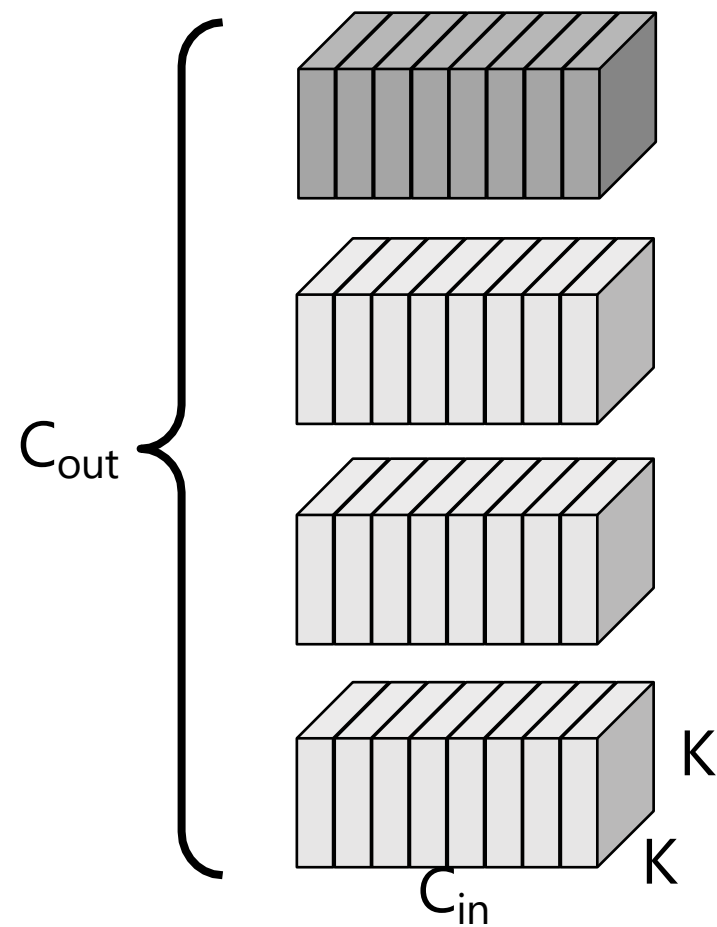


输出: $C_{out} \times H' \times W'$

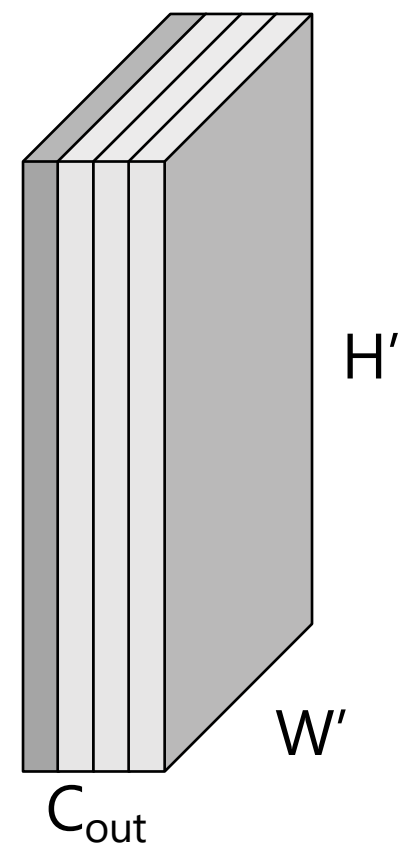
卷积层



输入: $C_{in} \times H \times W$

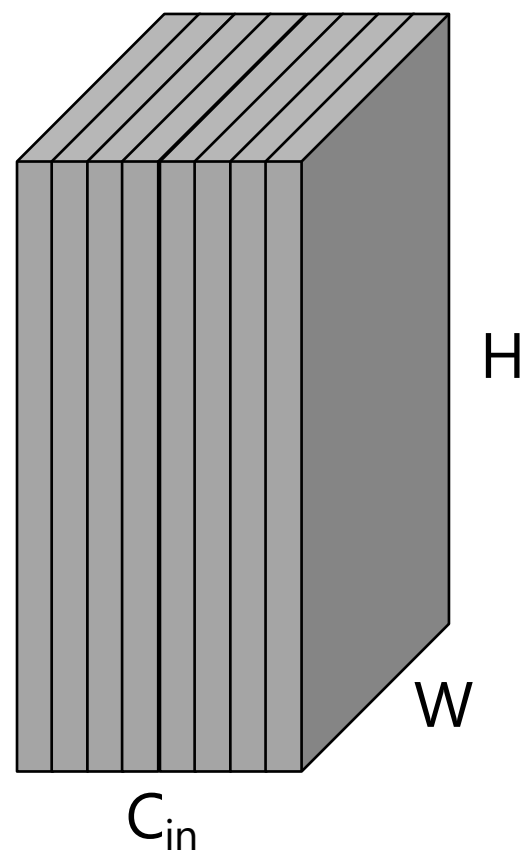


权重: $C_{out} \times C_{in} \times H \times W$

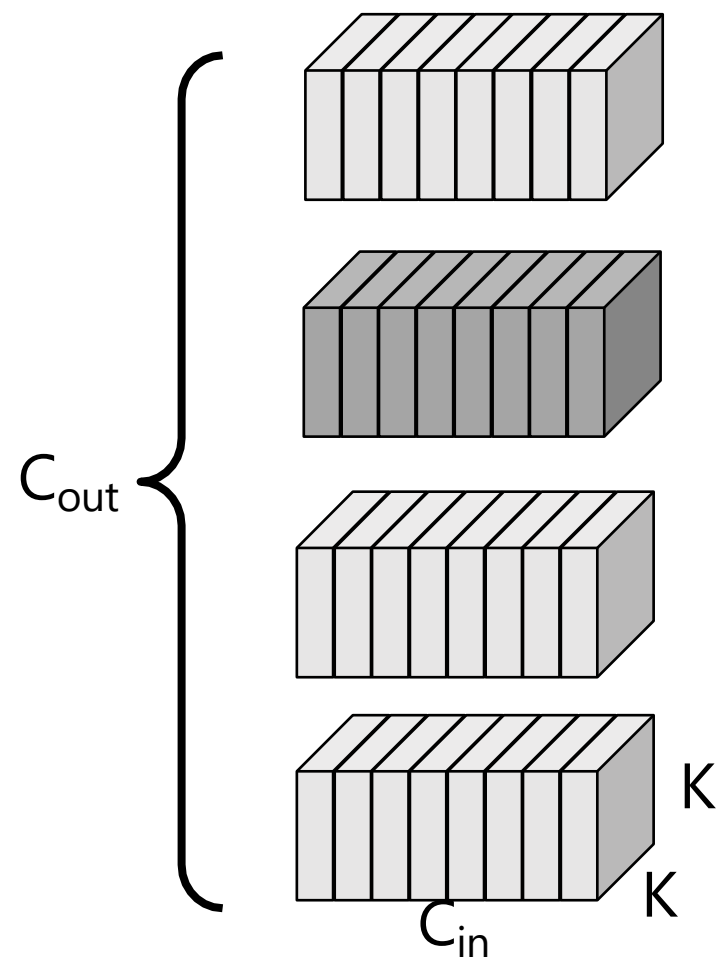


输出: $C_{out} \times H' \times W'$

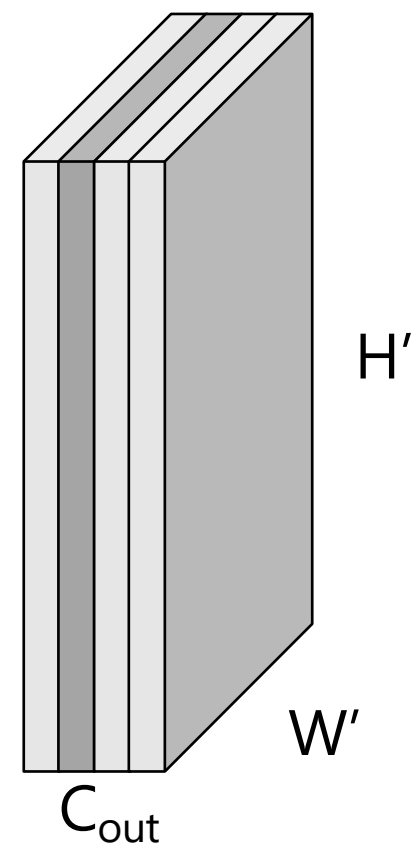
卷积层



输入: $C_{in} \times H \times W$



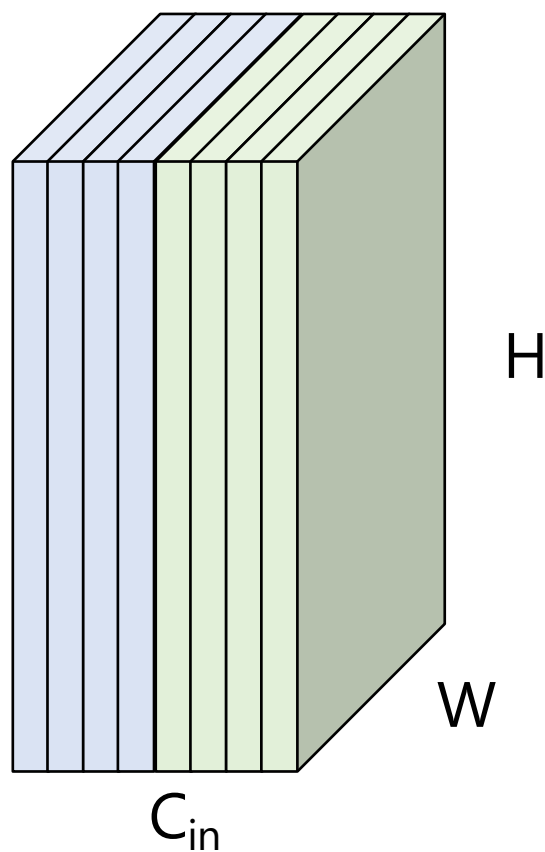
权重: $C_{out} \times C_{in} \times H \times W$



输出: $C_{out} \times H' \times W'$

Group Convolution 组卷积层

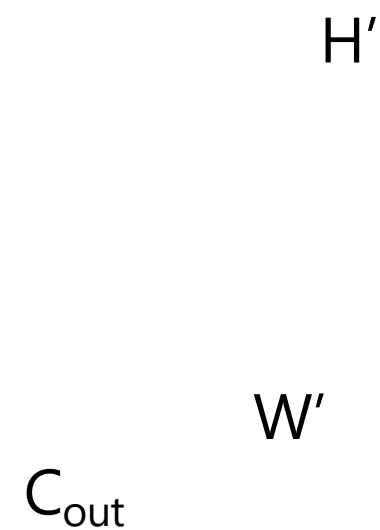
把输入通道分为 G 组
每组 (C_{in}/G) 个通道



输入: $C_{in} \times H \times W$

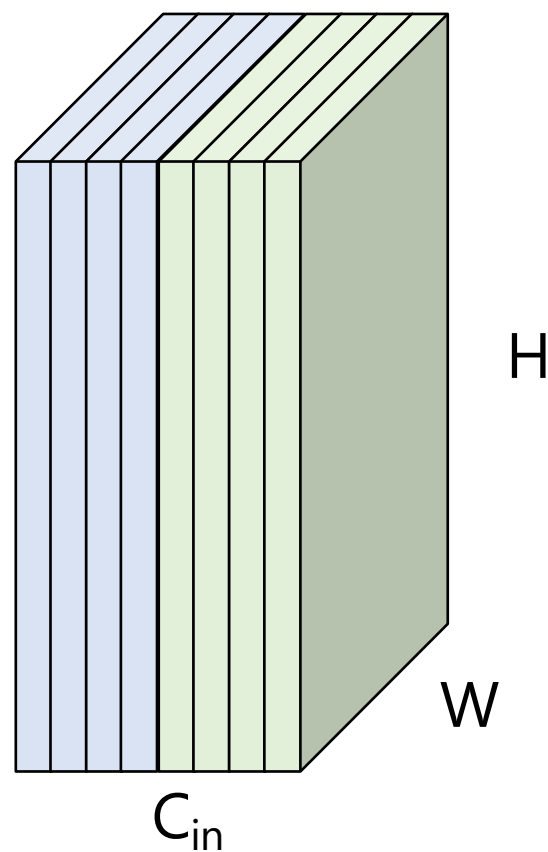
权重: $C_{out} \times (C_{in} / G) \times H \times W$

输出: $C_{out} \times H' \times W'$



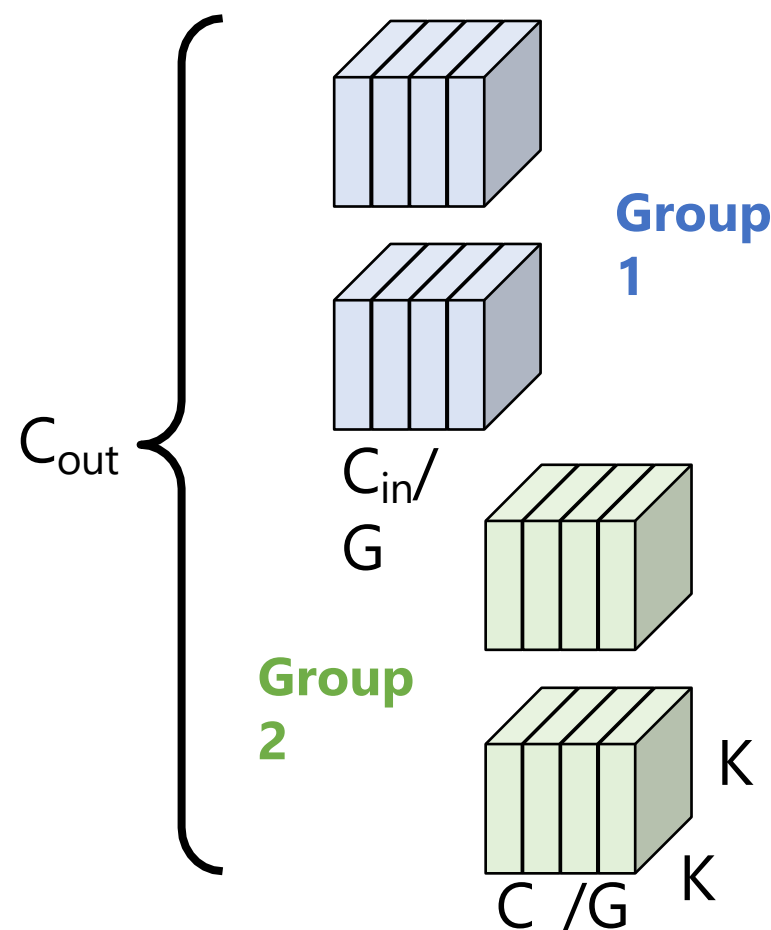
Group Convolution 组卷积层

把输入通道分为 G 组
每组 (C_{in}/G) 个通道

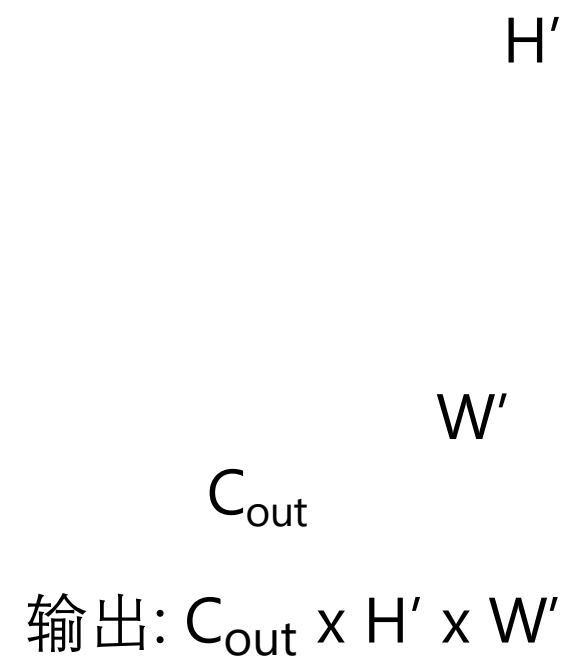


输入: $C_{in} \times H \times W$

滤波同样分为 G 组，每组只看自己组内的输入通道



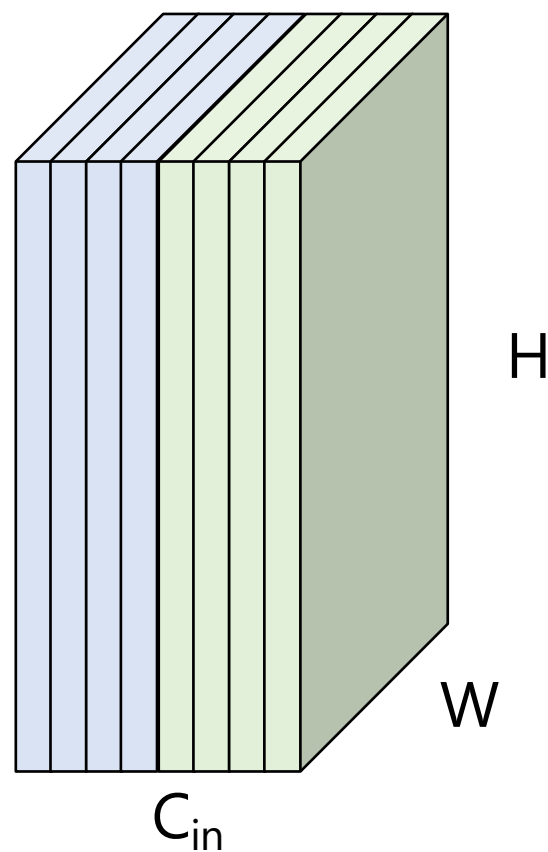
权重: $C_{out} \times (C_{in} / G) \times H \times W$



输出: $C_{out} \times H' \times W'$

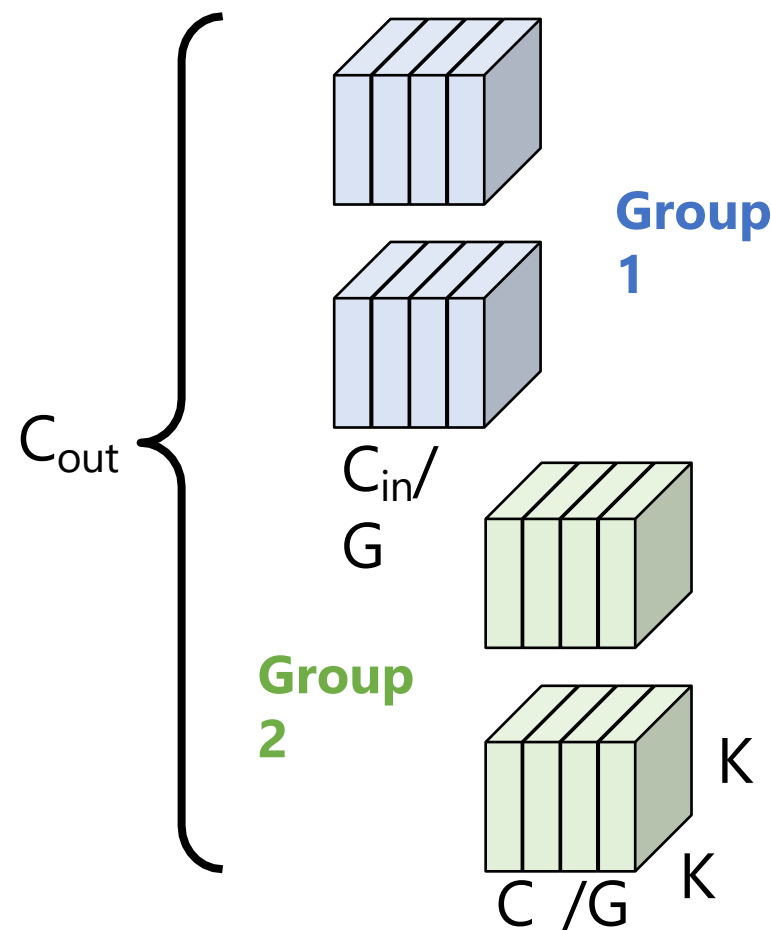
Group Convolution 组卷积层

把输入通道分为 G 组
每组 (C_{in}/G) 个通道



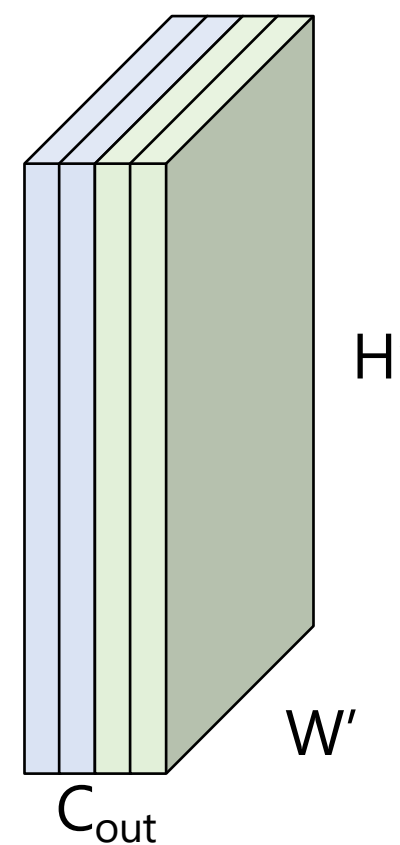
输入: $C_{in} \times H \times W$

滤波同样分为 G 组，每组只
看自己组内的输入通道



权重: $C_{out} \times (C_{in} / G) \times H \times W$

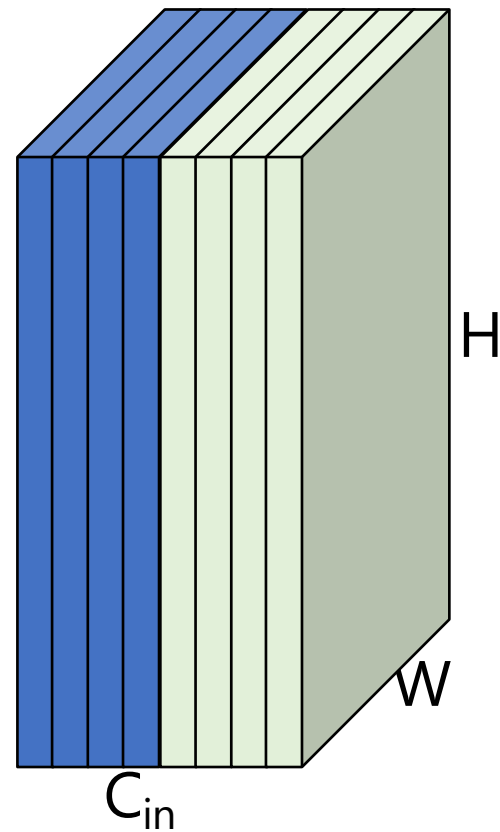
输出依照卷积滤波，只看某
组内的输入通道



输出: $C_{out} \times H' \times W'$

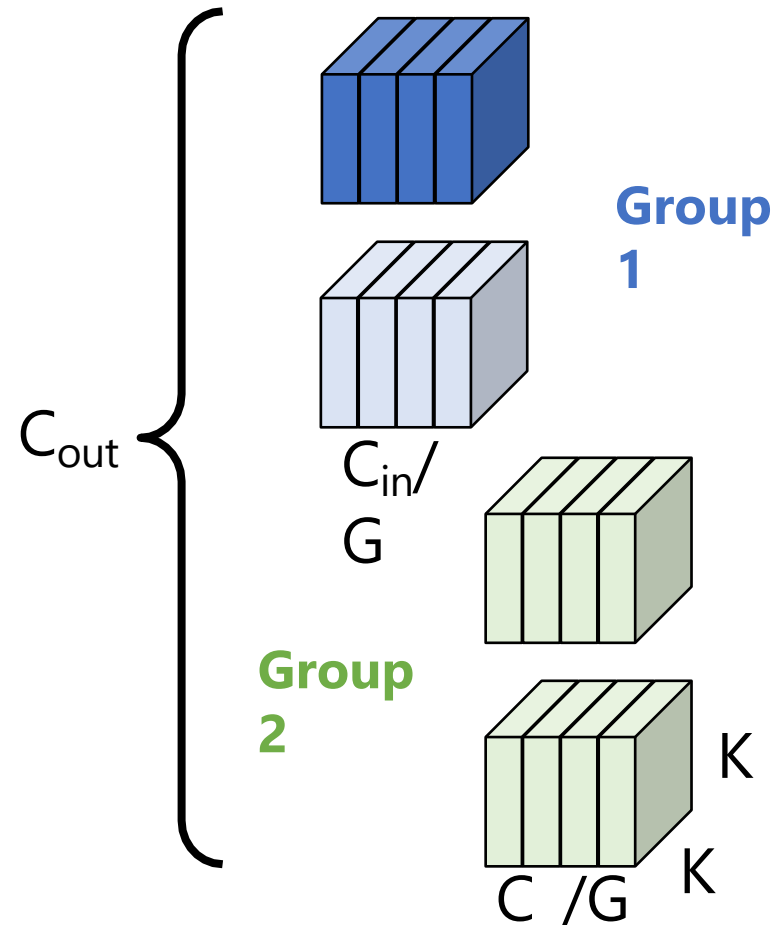
Group Convolution 组卷积层

把输入通道分为 G 组
每组 (C_{in}/G) 个通道



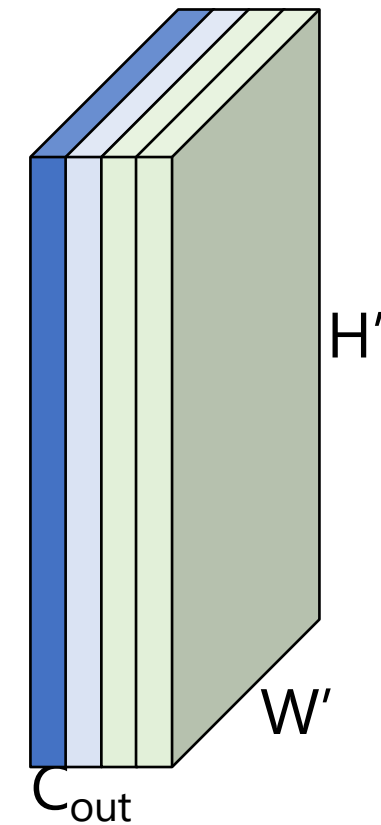
输入: $C_{in} \times H \times W$

滤波同样分为 G 组，每组只看自己组内的输入通道



权重: $C_{out} \times (C_{in} / G) \times H \times W$

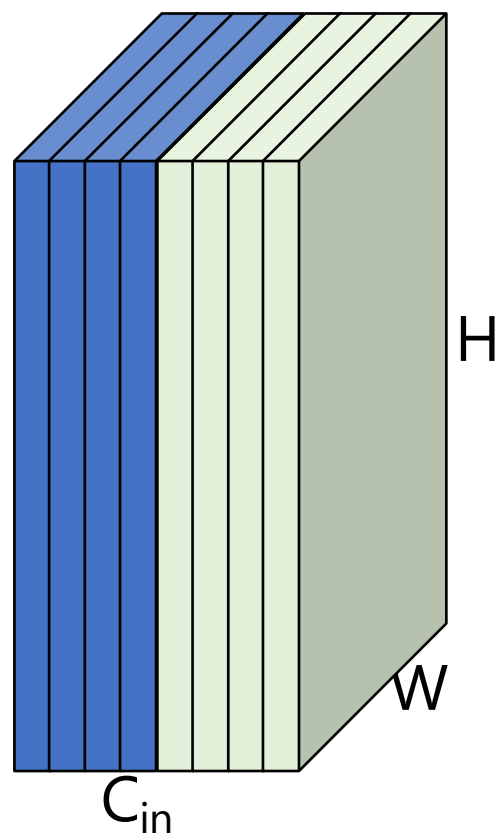
输出依照卷积滤波，只看某组内的输入通道



输出: $C_{out} \times H' \times W'$

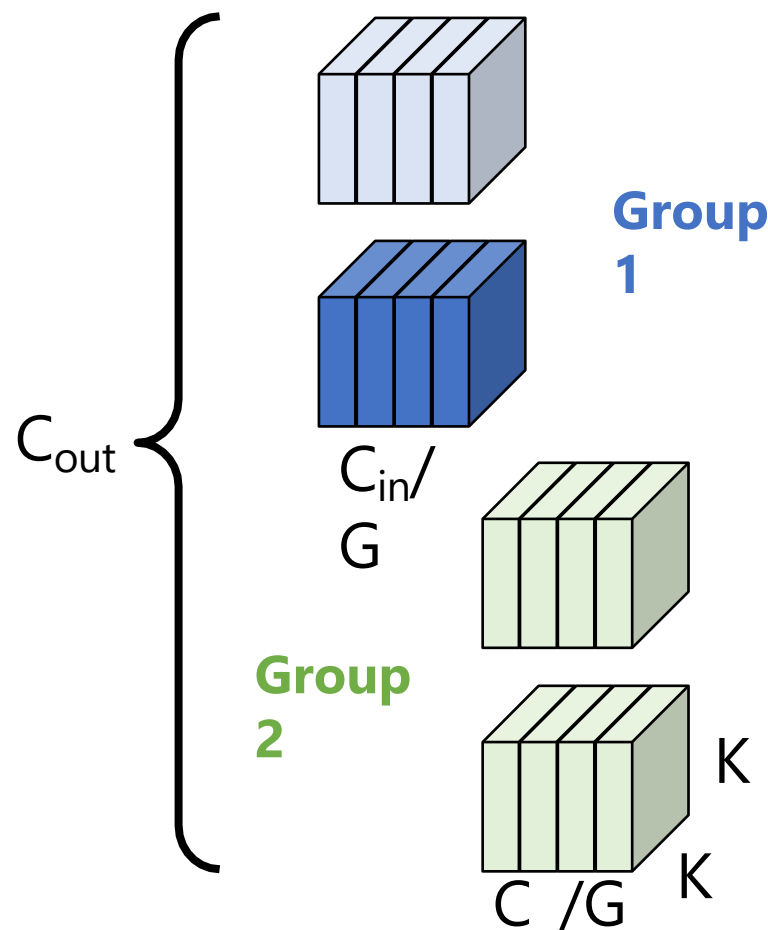
Group Convolution 组卷积层

把输入通道分为 G 组
每组 (C_{in}/G) 个通道



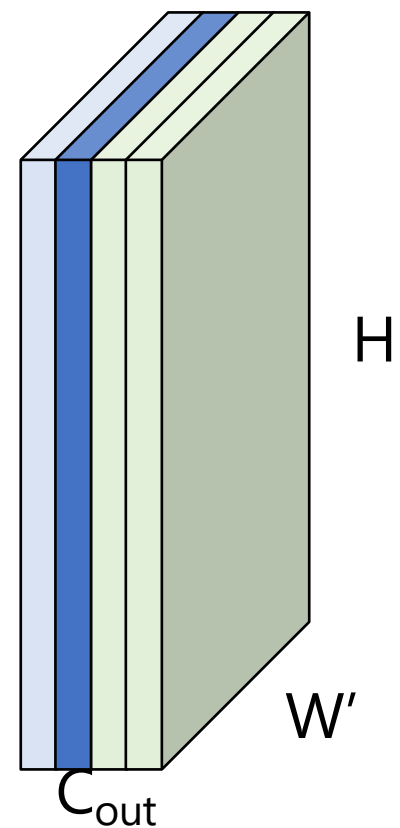
输入: $C_{in} \times H \times W$

滤波同样分为 G 组，每组只
看自己组内的输入通道



权重: $C_{out} \times (C_{in} / G) \times H \times W$

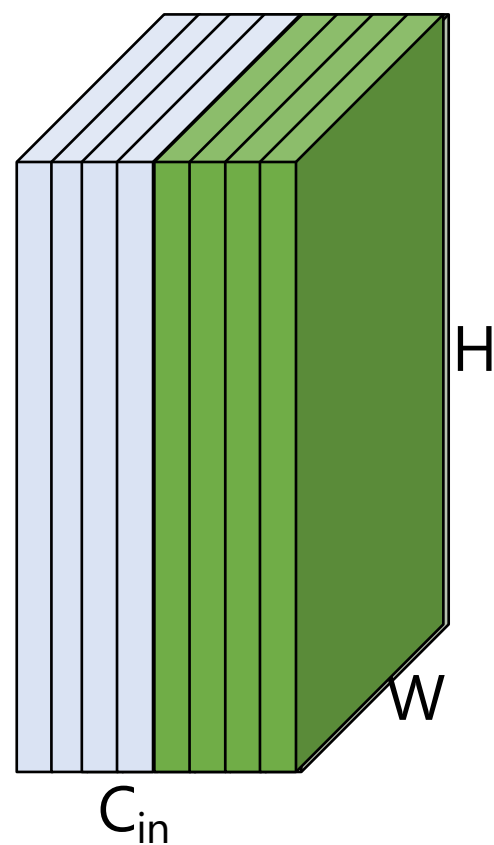
输出依照卷积滤波，只看某
组内的输入通道



输出: $C_{out} \times H' \times W'$

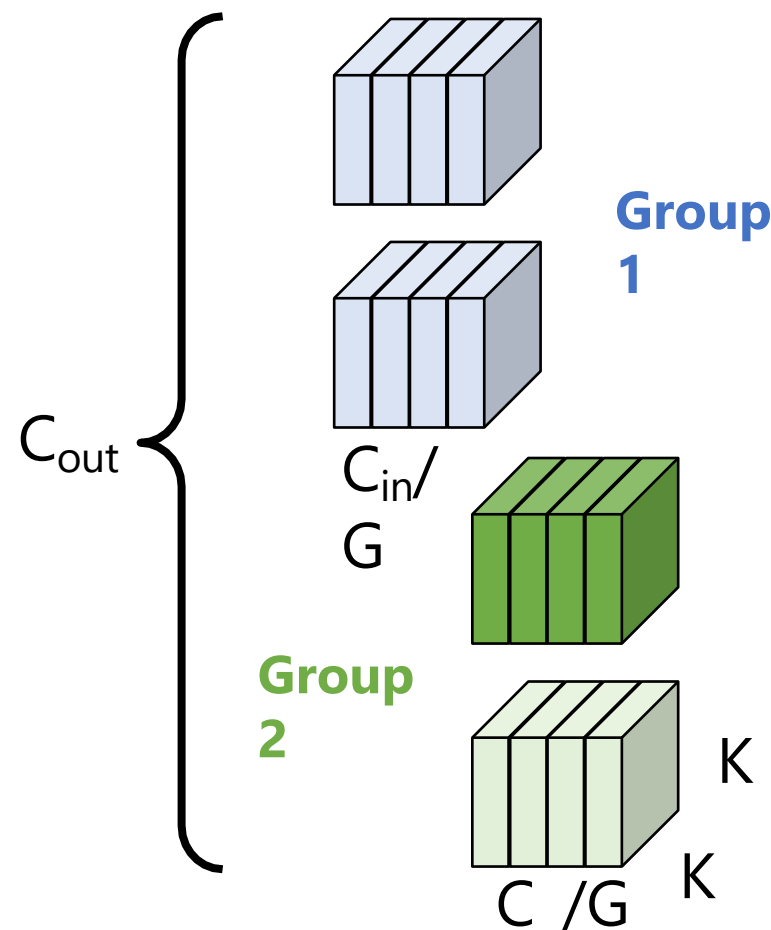
Group Convolution 组卷积层

把输入通道分为 G 组
每组 (C_{in}/G) 个通道



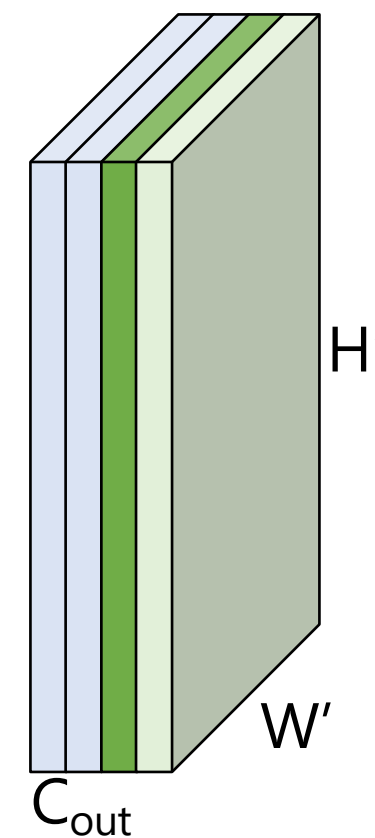
输入: $C_{in} \times H \times W$

滤波同样分为 G 组，每组只看自己组内的输入通道



权重: $C_{out} \times (C_{in} / G) \times H \times W$

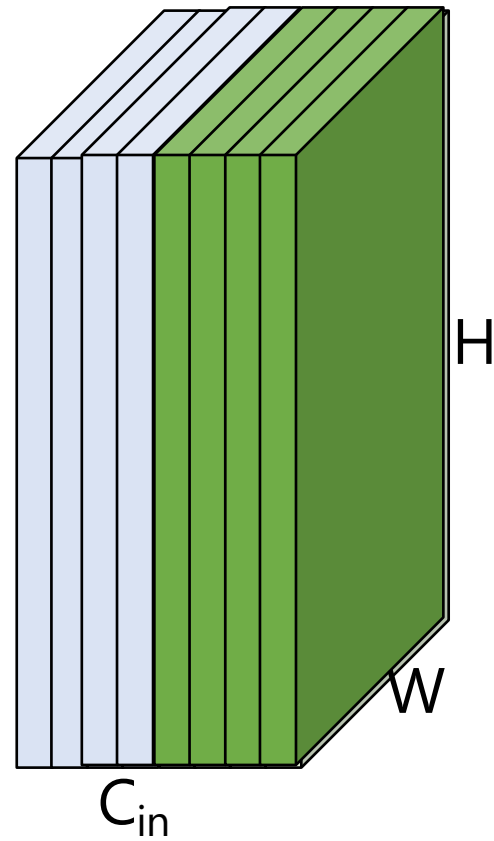
输出依照卷积滤波，只看某组内的输入通道



输出: $C_{out} \times H' \times W'$

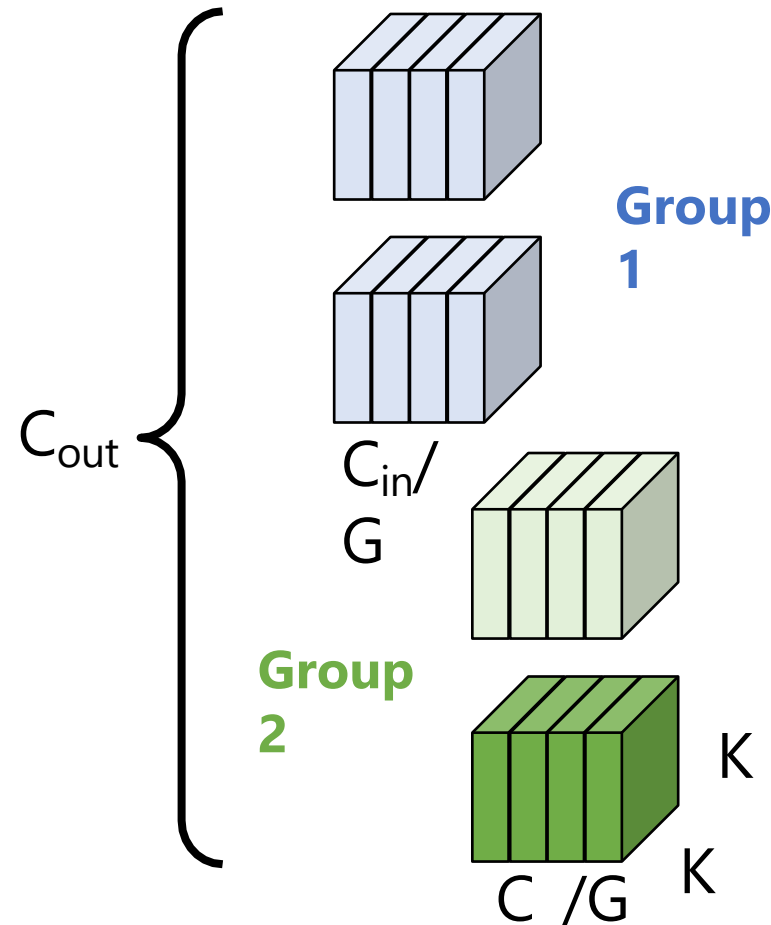
Group Convolution 组卷积层

把输入通道分为 G 组
每组 (C_{in}/G) 个通道



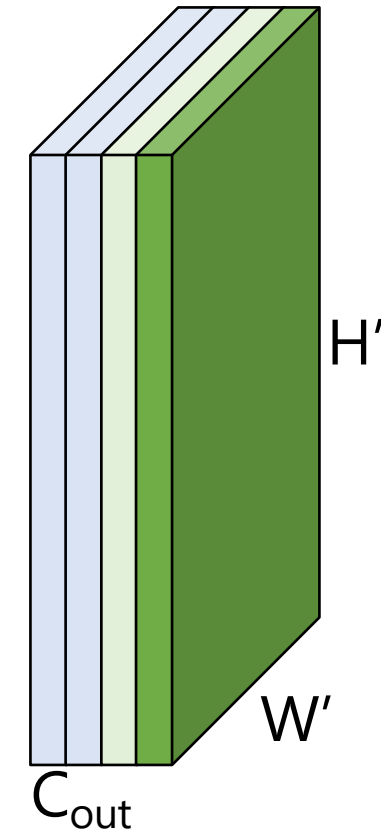
输入: $C_{in} \times H \times W$

滤波同样分为 G 组，每组只看自己组内的输入通道



权重: $C_{out} \times (C_{in} / G) \times H \times W$

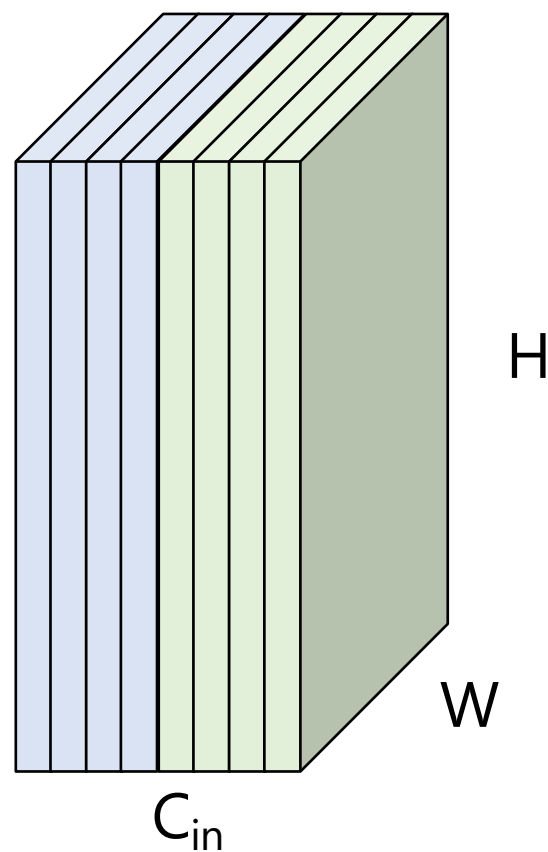
输出依照卷积滤波，只看某组内的输入通道



输出: $C_{out} \times H' \times W'$

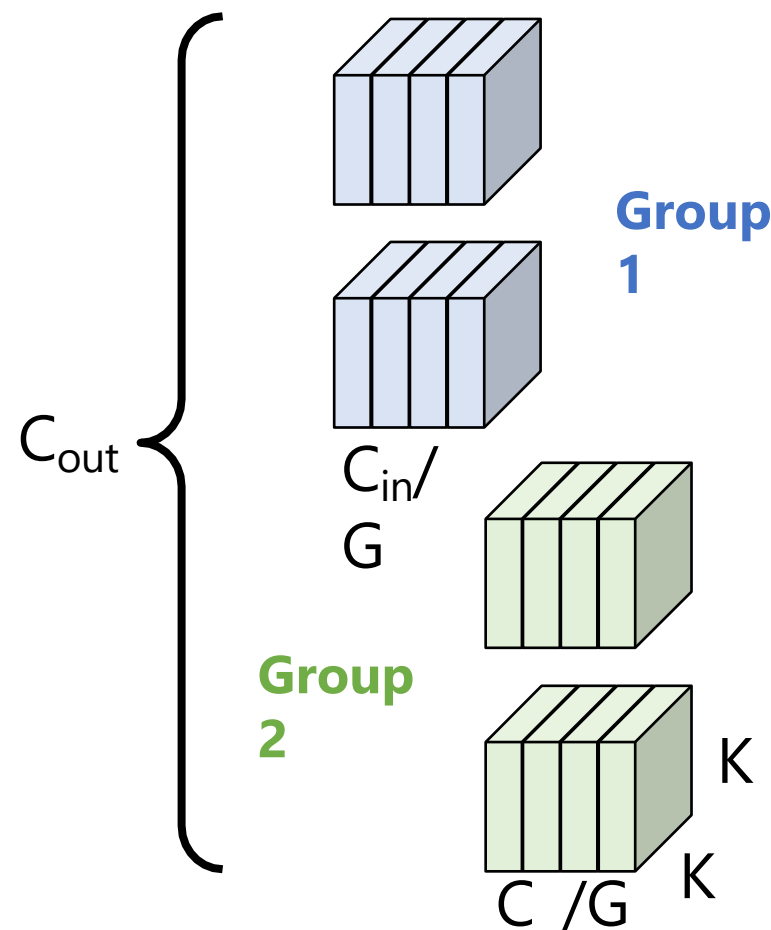
Group Convolution 组卷积层

把输入通道分为 G 组
每组 (C_{in}/G) 个通道



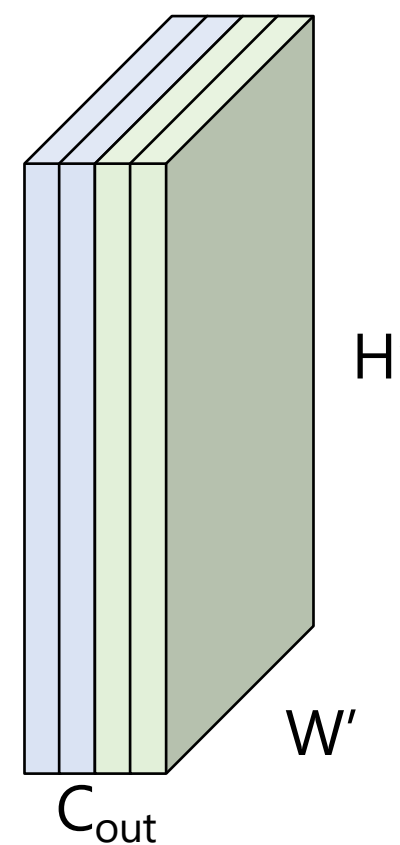
输入: $C_{in} \times H \times W$

滤波同样分为 G 组，每组只
看自己组内的输入通道



权重: $C_{out} \times (C_{in} / G) \times H \times W$

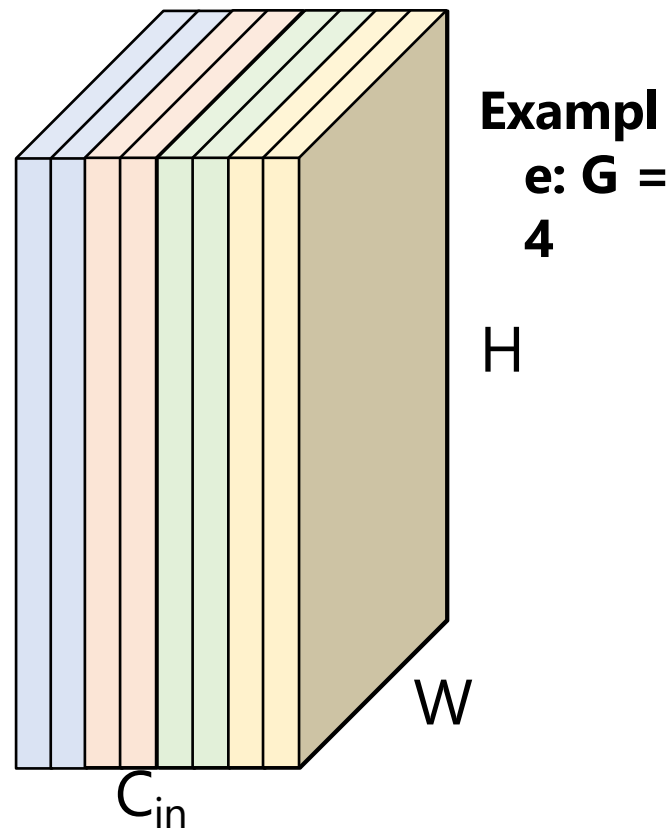
输出依照卷积滤波，只看某
组内的输入通道



输出: $C_{out} \times H' \times W'$

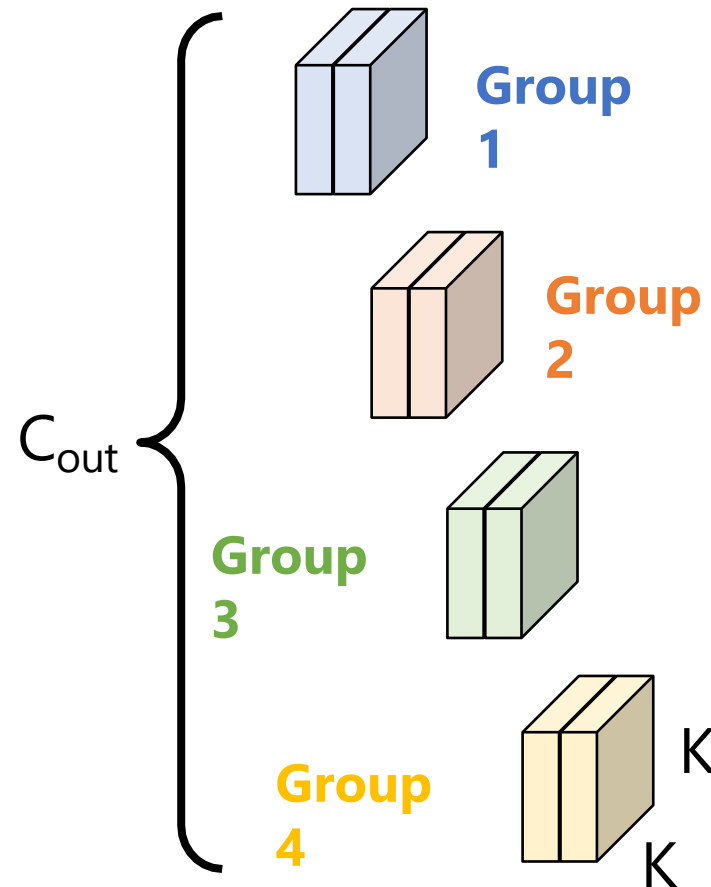
Group Convolution 组卷积层

把输入通道分为 G 组
每组 (C_{in}/G) 个通道



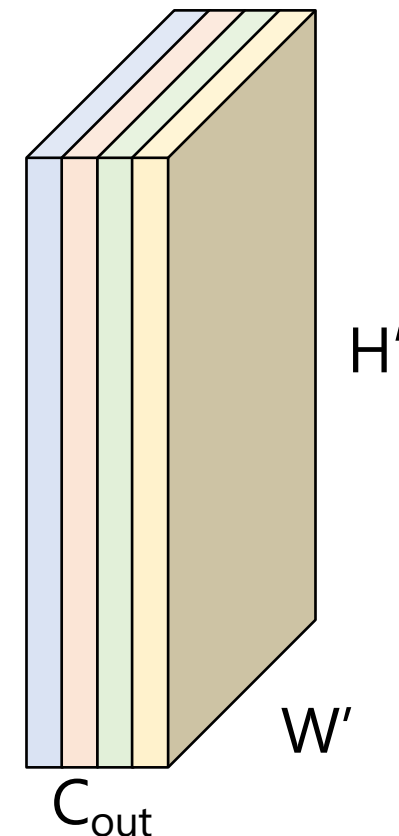
输入: $C_{in} \times H \times W$

滤波同样分为 G 组, 每组只看自己组内的输入通道



权重: $C_{out} \times (C_{in} / G) \times H \times W$

输出依照卷积滤波, 只看某组内的输入通道

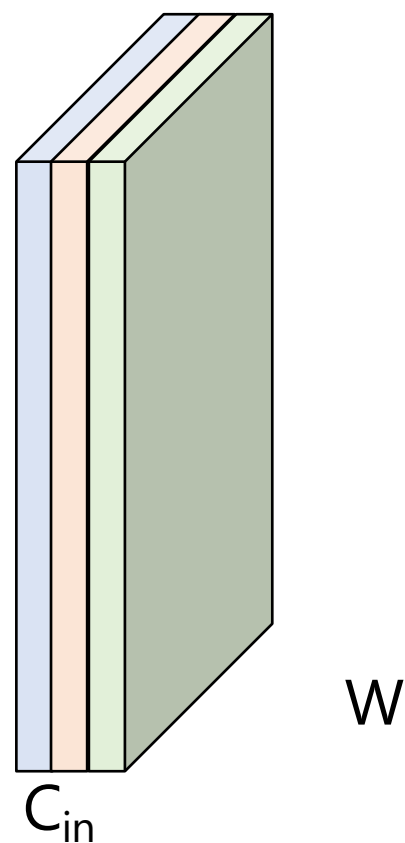


输出: $C_{out} \times H' \times W'$

Depthwise Convolution 逐深度卷积层

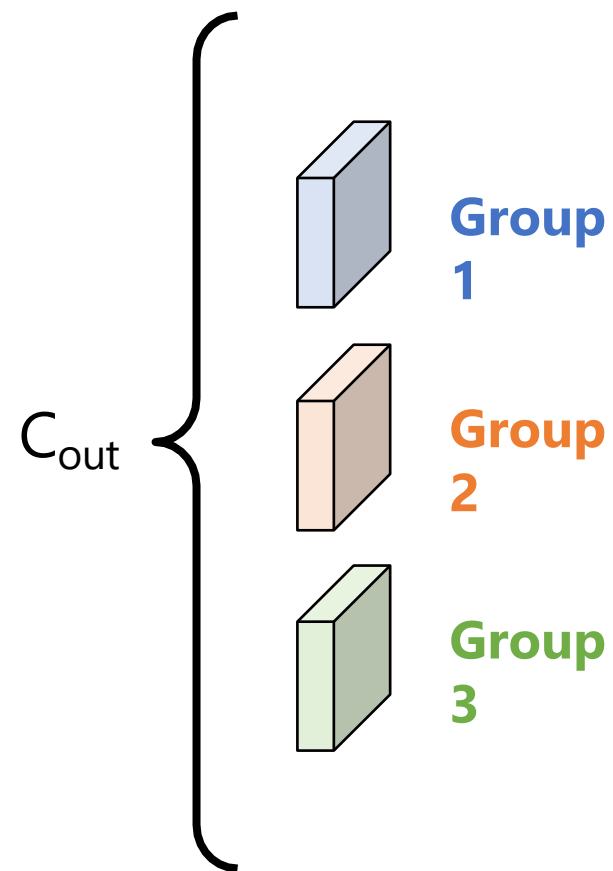
把输入通道各自分为一组,

$$G = C_{in}$$



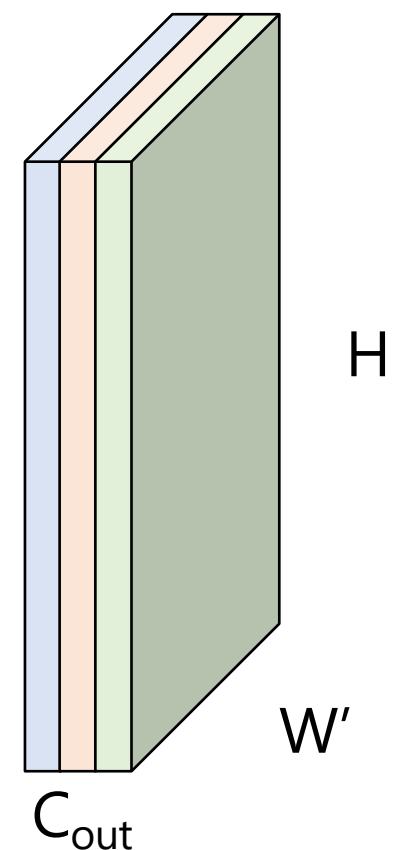
输入: $C_{in} \times H \times W$

滤波同样分为 G 组, 每组只看自己组内的输入通道



权重: $C_{out} \times 1 \times H \times W$

输出依照卷积滤波, 只看某组内的输入通道

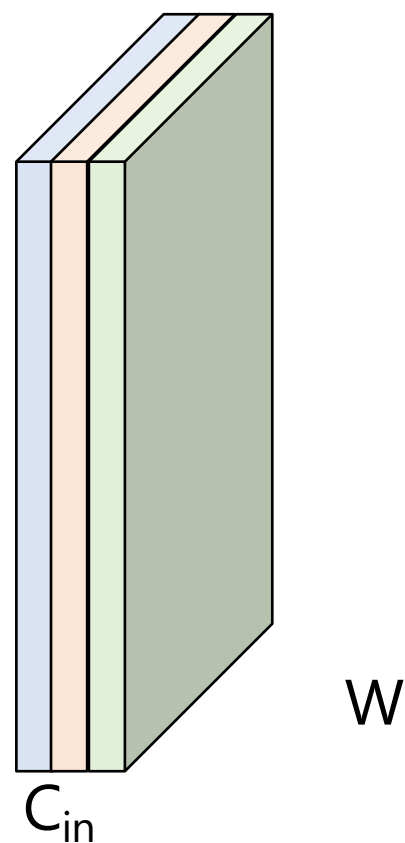


输出: $C_{out} \times H' \times W'$

Depthwise Convolution 逐深度卷积层

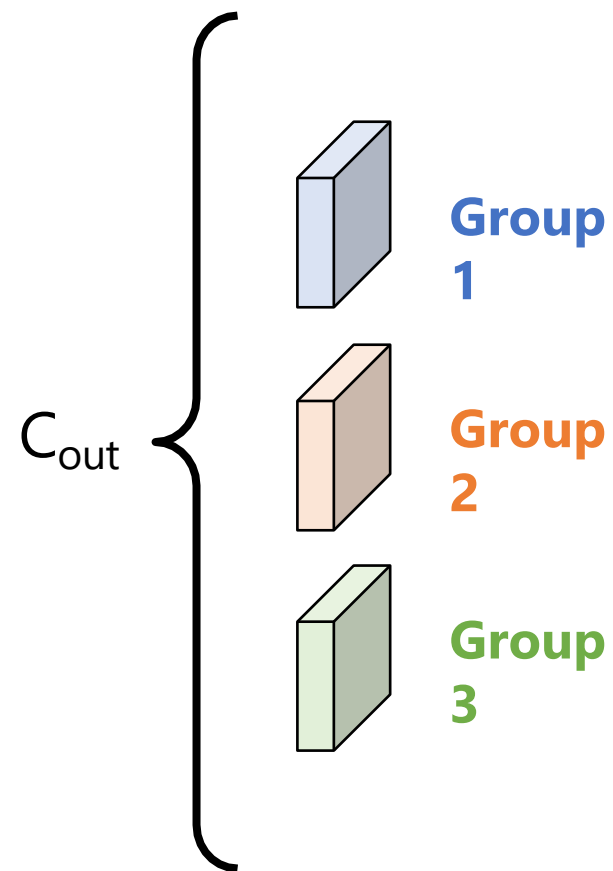
把输入通道各自分为一组,

$$G = C_{in}$$



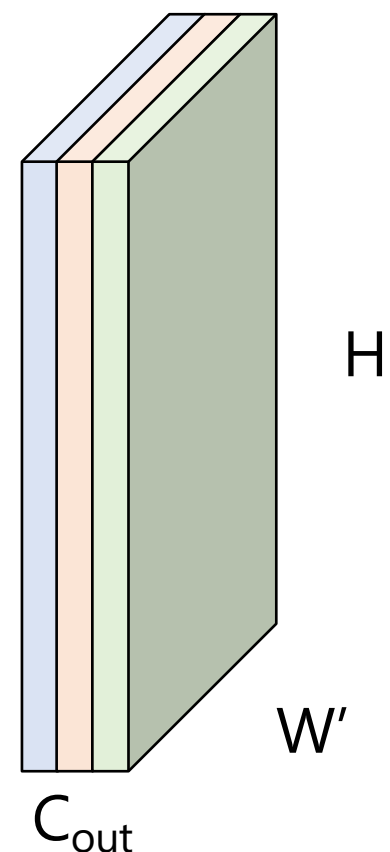
输入: $C_{in} \times H \times W$

滤波同样分为 G 组, 每组只看自己组内的输入通道



权重: $C_{out} \times 1 \times H \times W$

输出与输入通道对应, 卷积只做空间信息混合

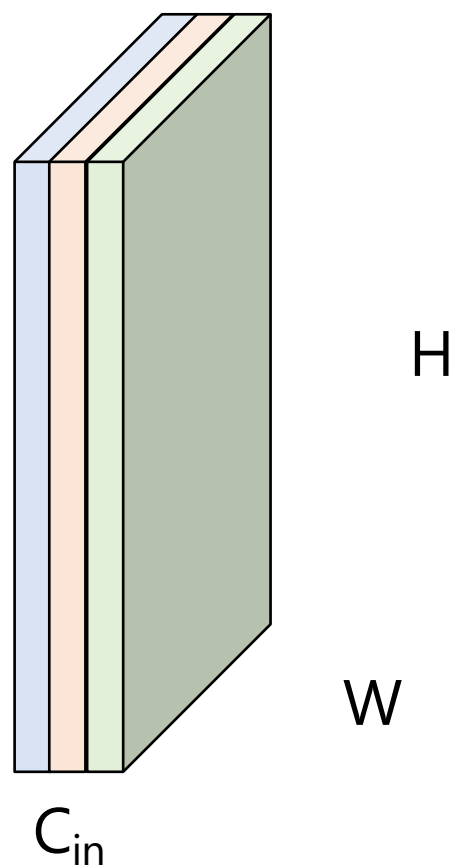


输出: $C_{out} \times H' \times W'$

Depthwise Convolution 逐深度卷积层

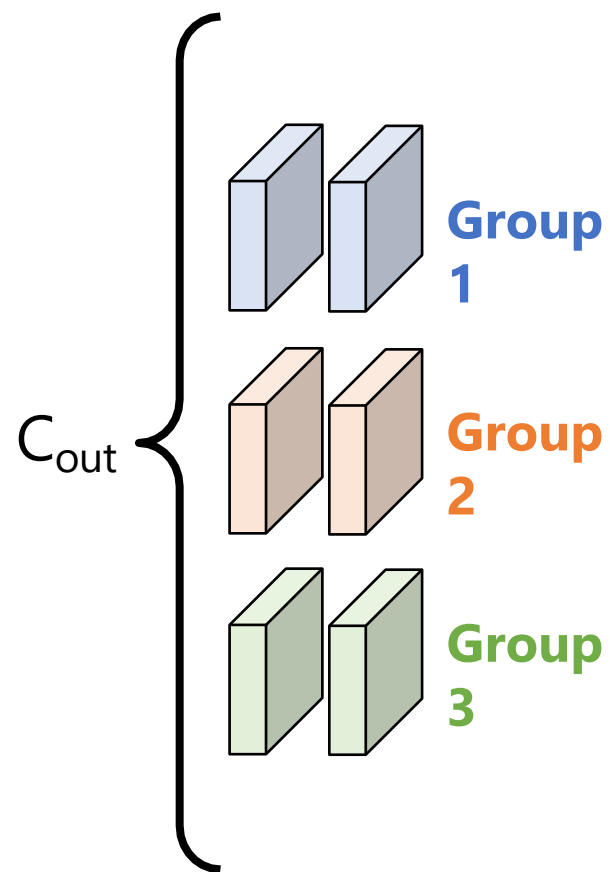
把输入通道各自分为一组,

$$G = C_{in}$$



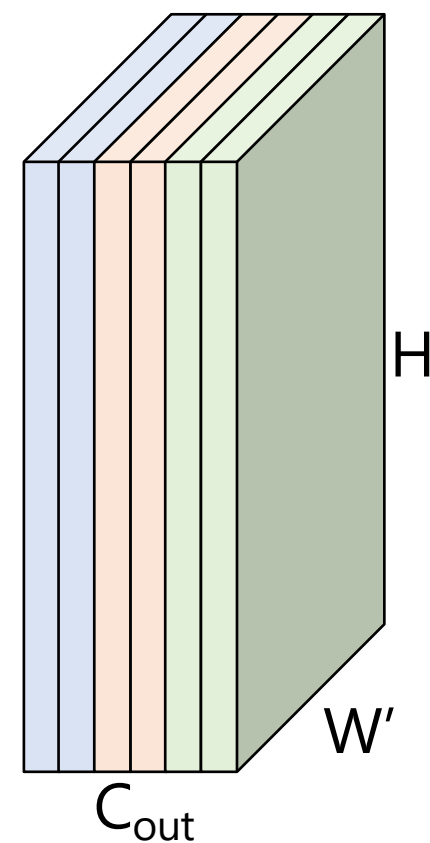
输入: $C_{in} \times H \times W$

滤波同样分为 G 组, 每组只看自己组内的输入通道



权重: $C_{out} \times 1 \times H \times W$

输出与输入通道对应, 卷积只做空间信息混合



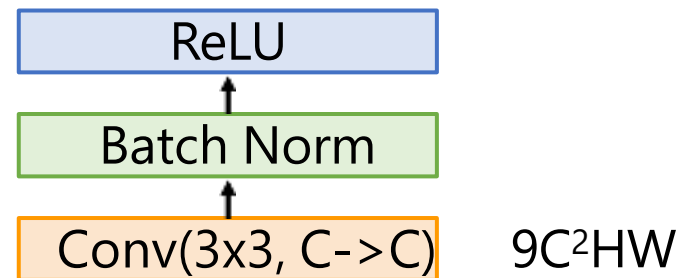
输出: $C_{out} \times H' \times W'$

MobileNet (2017)

- 目标: 减少网络参数
- 解决方案: Depthwise separable convolutions “分离卷积”

Standard Convolution Block

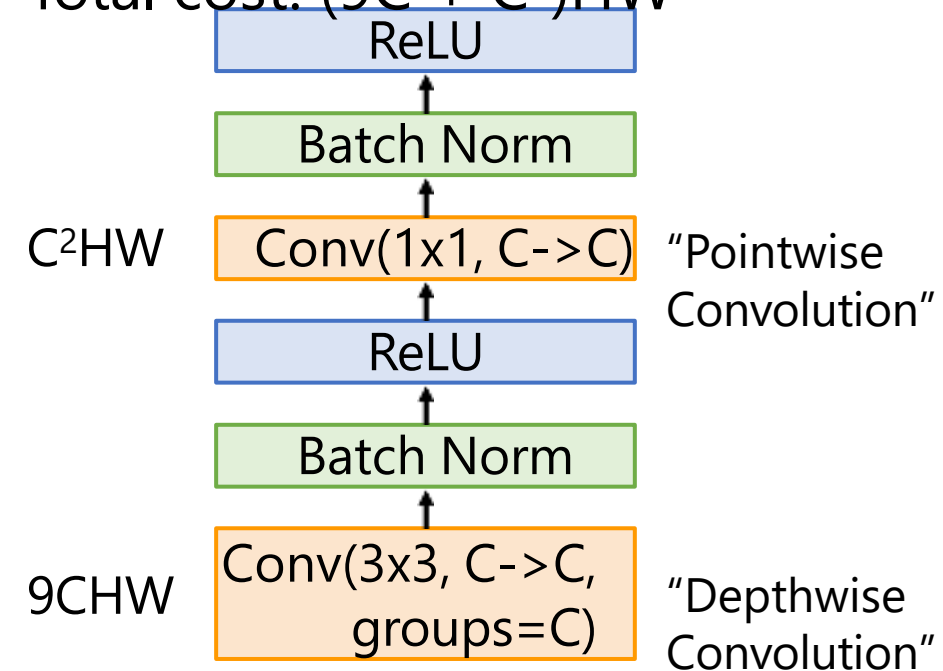
Total cost: $9C^2HW$



$$\begin{aligned} \text{Speedup} &= 9C^2 / (9C + C^2) \\ &= 9C / (9 + C) \\ &\Rightarrow 9 \text{ (as } C \rightarrow \text{inf)} \end{aligned}$$

Depthwise Separable Convolution

Total cost: $(9C + C^2)HW$



MobileNet (2017)

- 与 InceptionV3 接近的准确度
- 十倍的速度，十分之一的参数

Layer/Modification	Million Mult-Adds	Million Parameters
Convolution	462	2.36
Depthwise Separable Conv	52.3	0.27

Table 8. MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
GoogLeNet	69.8%	1550	6.8
VGG 16	71.5%	15300	138

Table 9. Smaller MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
0.50 MobileNet-160	60.2%	76	1.32
Squeezenet	57.5%	1700	1.25
AlexNet	57.2%	720	60

Table 10. MobileNet for Stanford Dogs

Model	Top-1 Accuracy	Million Mult-Adds	Million Parameters
Inception V3 [18]	84%	5000	23.2
1.0 MobileNet-224	83.3%	569	3.3
0.75 MobileNet-224	81.9%	325	1.9
1.0 MobileNet-192	81.9%	418	3.3
0.75 MobileNet-192	80.5%	239	1.9

训练与测试

训练时最小化:

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

测试时, 对预测类别 \hat{y}_i :

Accuracy 准确率: $\frac{1}{n} \sum_{i=1}^n 1(y_i = \hat{y}_i)$

多类别测试

这样图分类为猫还是狗？



为了避免有歧义的图片影响测试，一般给出五个预测结果 (top-5 accuracy):

只要五个预测有一个对就算正确.

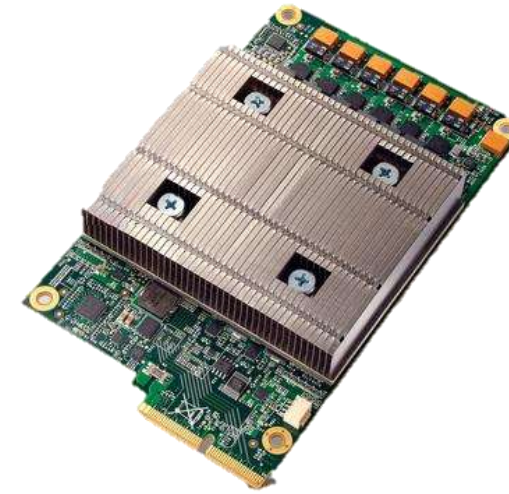
准确率

	Top 1 Error	Top 5 Error
Best Pre-Deep (~2012)	-	26.2%
Alexnet, 2012	43.5%	20.9%
VGG-16, 2014	28.4%	9.6%
ResNet-50, 2015	24.7%	7.8%
ResNet-152, 2015	21.7%	5.9%
ResNet-50 done better, 2018	20.7%	5.4%
Swin Transf., 2021	15.5%	-
ConvNeXt, 2022	14.5%	-
CoAtNet-7* 2021 (2B params!)	9.1%	-
Human*	-	5.1%

Many results from <https://paperswithcode.com/sota/image-classification-on-imagenet>. I am missing loads of great papers, and the numbers depend on tons of practical details. *Human – this number is from Andrej Karpathy and isn't really human performance with training but a ballpark. Resnet-50 one better = "Bag of Tricks for Image Classification with Convolutional Neural Networks", He et al.

训练CNN

- 下载数据集, 准备GPU
- 初始化网络参数
- for epoch in range(epochs):
 - 打乱数据集
 - for each minibatch in dataset:
 - 加载数据
 - 计算梯度
 - 优化网络



从头训练CNN

需要找到一个初始的网络权重 \mathbf{w}

- AlexNet: weights \sim Normal(0,0.01), bias = 1
- “Xavier” 初始化: Uniform($-\frac{1}{\sqrt{n}}$, $\frac{1}{\sqrt{n}}$) n 为神经元个数
- “Kaiming” 初始化: Normal(0, $\sqrt{2/n}$)

Quick Quiz

哪一张图里有河马？



Horizontal
Flip



Color
Jitter

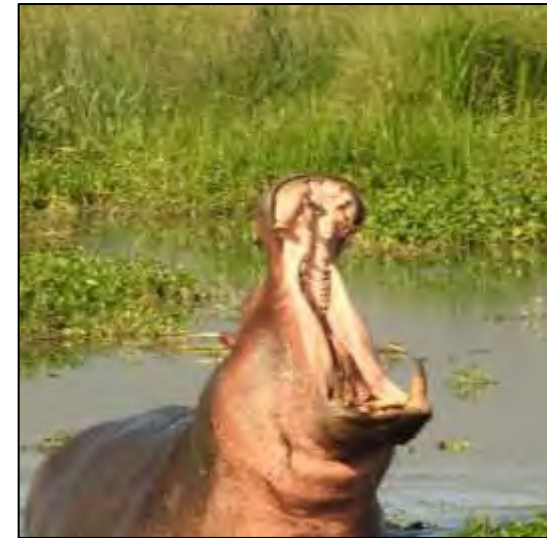
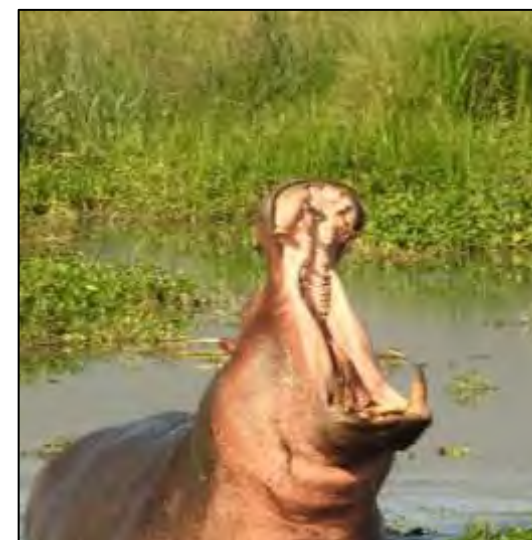
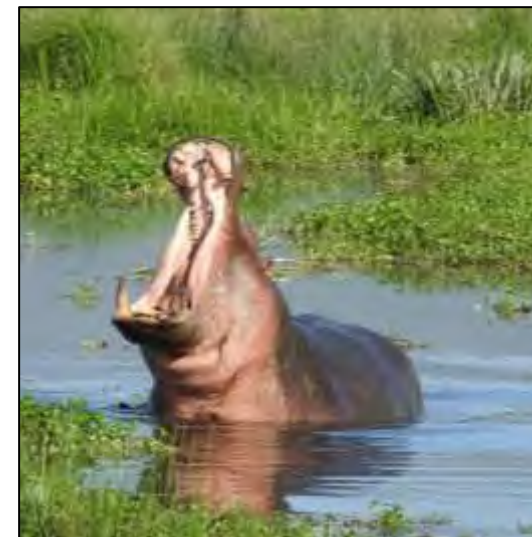


Image
Cropping

训练CNN -Augmentation 图像增广

- 应用一些不会影响图像分类结果的变换
- 请注意，一定要确保不会影响图像的表达语义

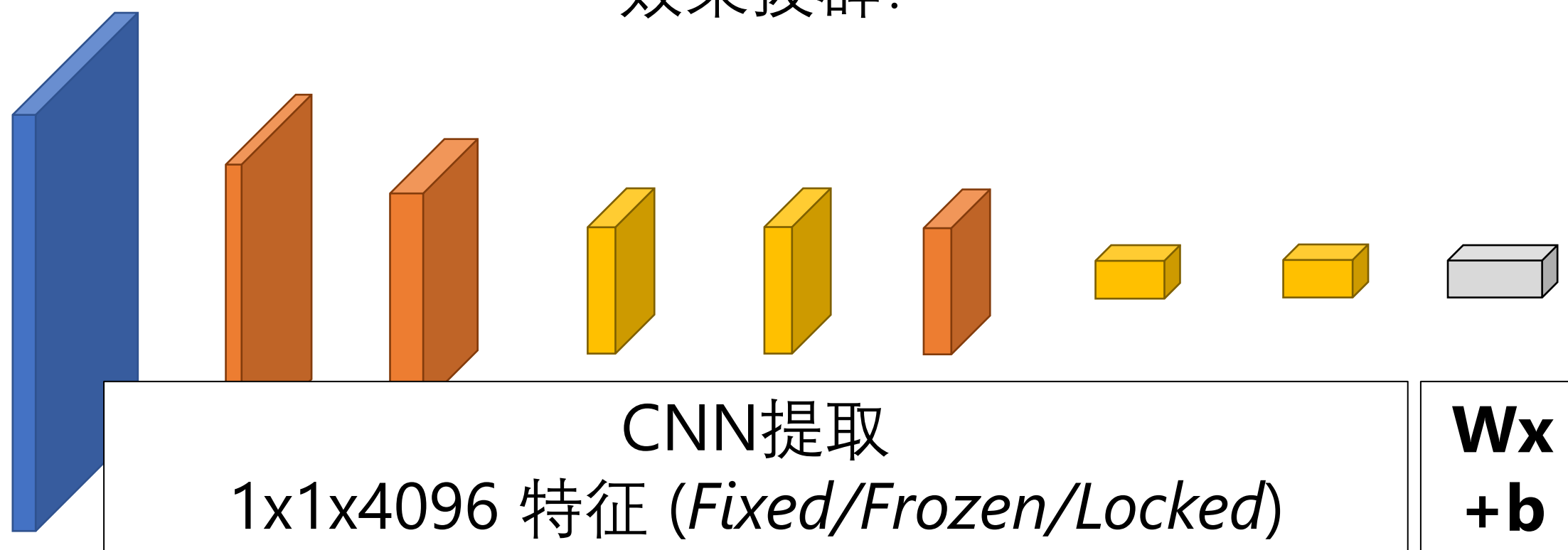


训练CNN – Fine-tuning 微调

- 如果你没有（没有大量的）训练数据呢？

预训练模型与特征

1. 从已有网络的某一层提取特征
2. 当作你的初始特征.
3. 训练线性模型.
效果拔群!

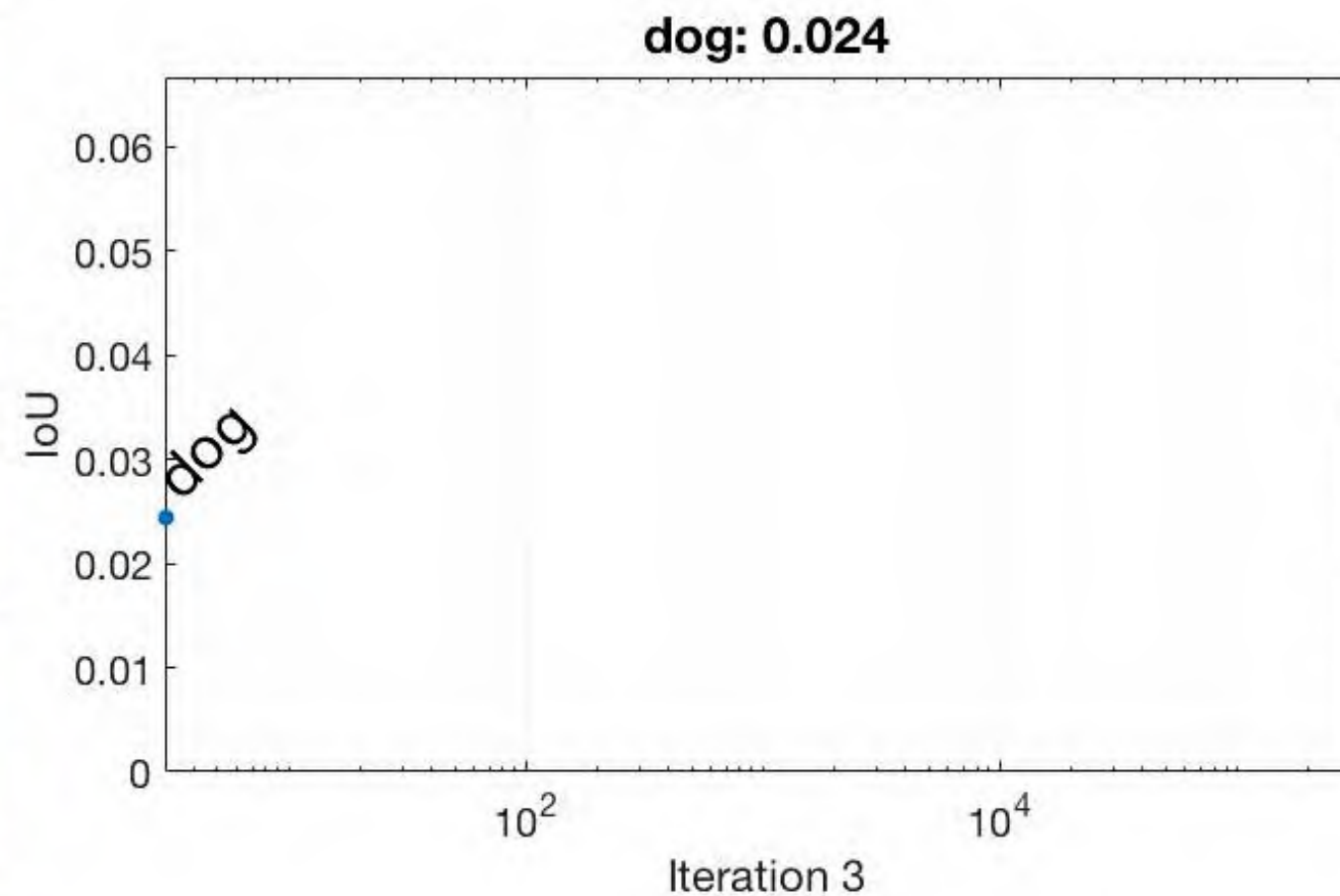


微调：迁移学习

- 与其初始化随机参数，用已经训练好的网络参数.
- 最常用的是在ImageNet训练的网络.
- 也有其他的特定任务用特定的数据集，目前也有更新更大的数据集出现

微调：迁移学习

为什么能实现？
从 物体(dog) 迁移到 场景(waterfall)



一般性建议

- <10K images: 用预训练特征
- 永远都要尝试微调 **finetune**
- >100K images: 从头训练

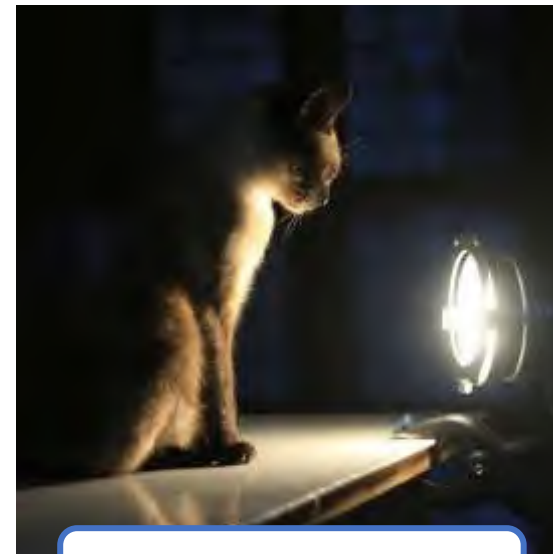
- 思考：为什么？

回顾：外观的巨大差异让识别问题困难倍增

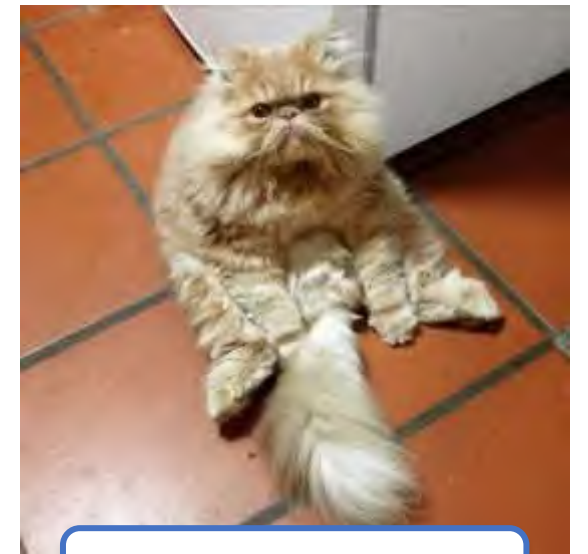
- 相同类别的物体图像差异极大



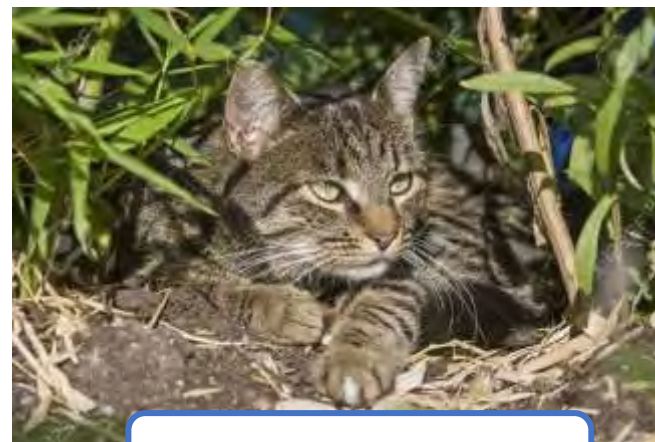
Viewpoint Variation



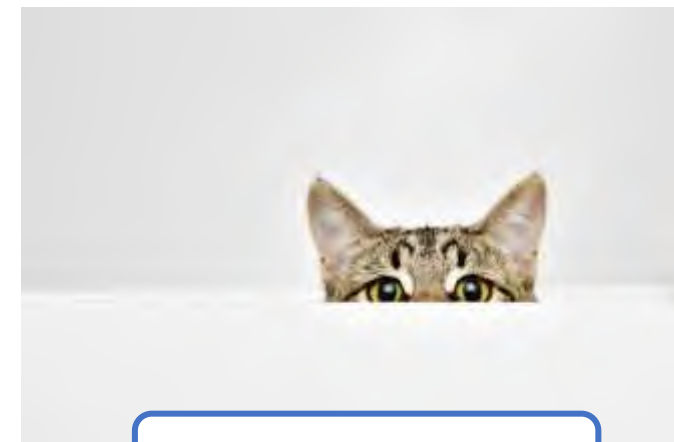
Lighting Variation



Deformation



Background Clutter



Occlusion

问题：欠约束 Under-constrained

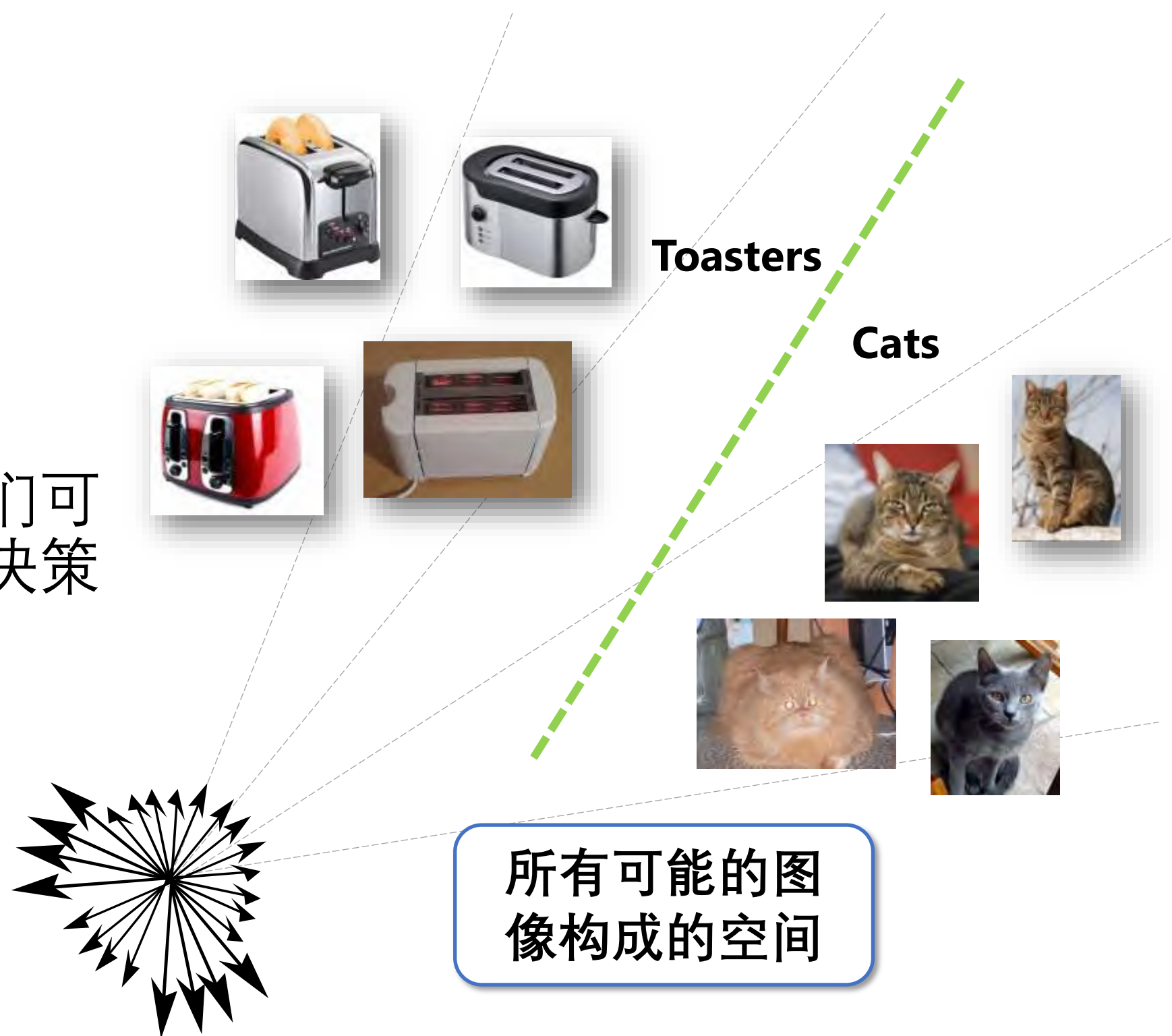
- 不同的现实情境可以产生相同的图像：这意味着从同一个图像出发，我们可能会得到多种不同的解释或解决方案。
- 通常我们不能计算“正确”的答案：由于存在多种可能的现实情境，我们通常不能确定一个图像的“正确”解释。但我们可以计算最有可能的答案。
- 我们需要某种先验来加以条件：由于问题是欠约束的，我们需要某种先验知识或假设来缩小可能的答案范围。
- 我们可以从数据中学习这个先验：通过训练数据，我们可以学习到这些先验知识或假设，以更好地解决问题。

$$f(x) = \underset{\ell_x}{\operatorname{argmax}} P(\ell_x | \text{data})$$



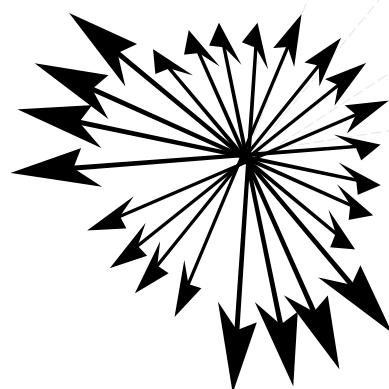
将图像视为高维向量

- 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。

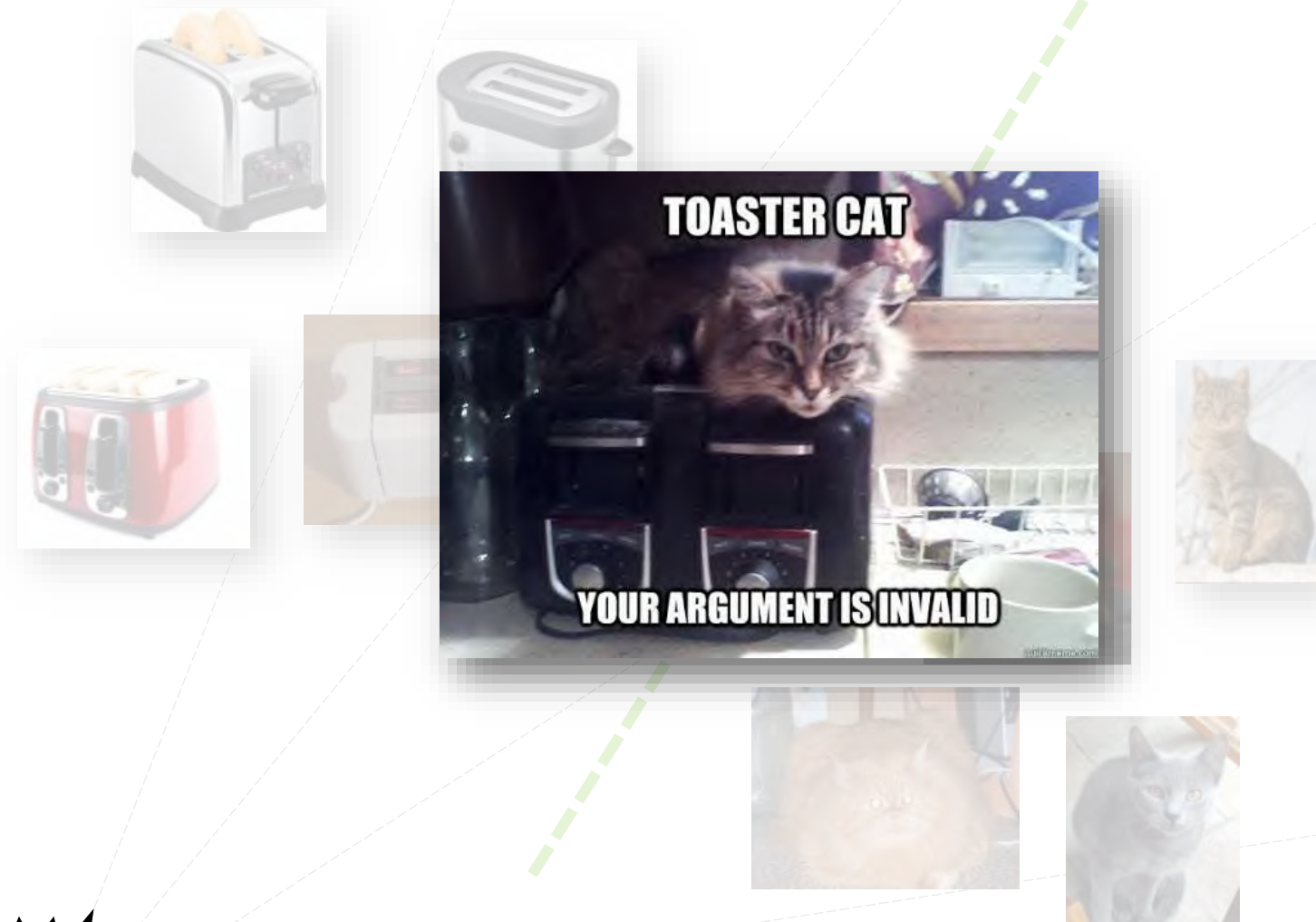


将图像视为高维向量

- 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。



所有可能的图像构成的空间

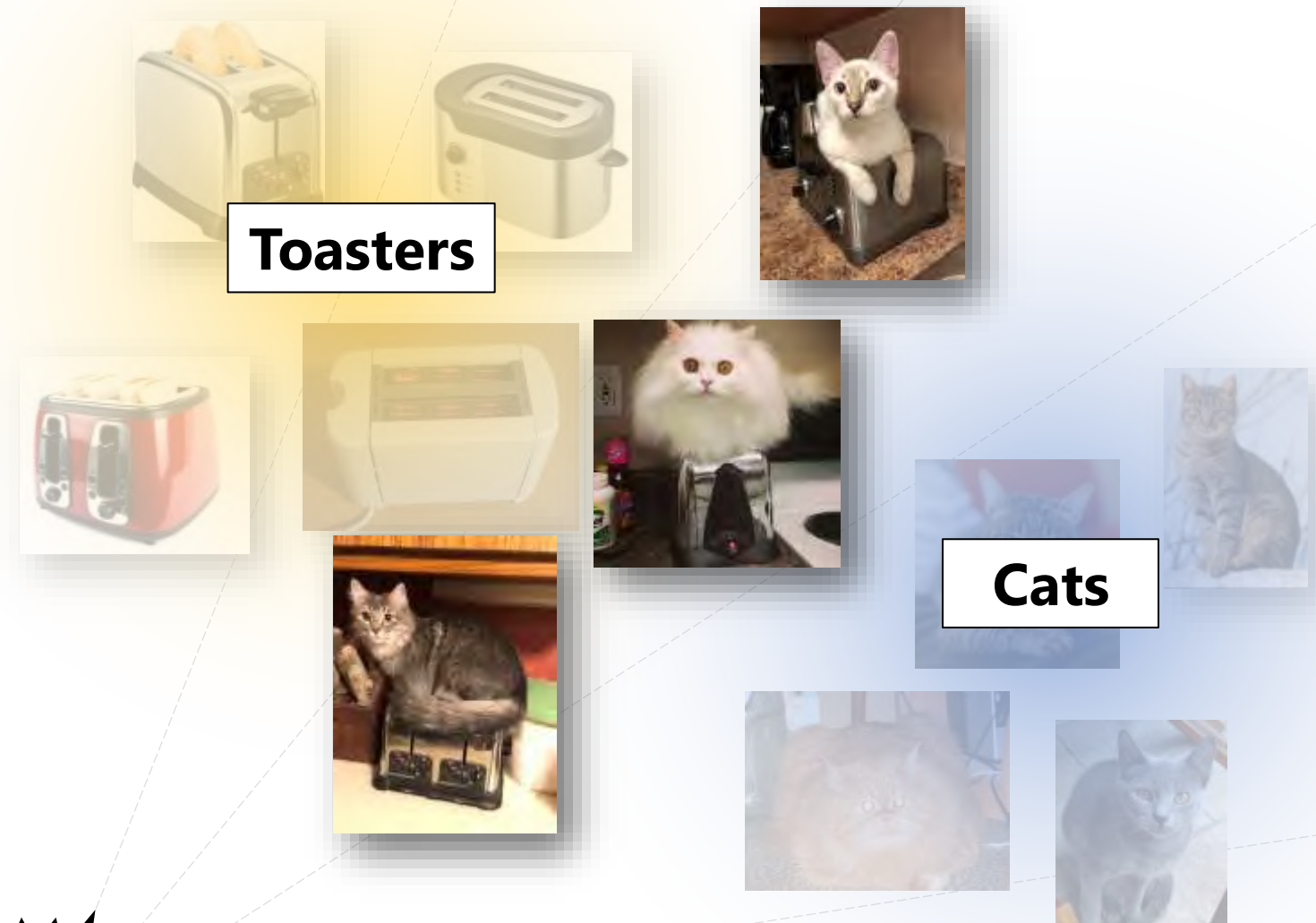


将图像视为高维向量

- 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。

或者

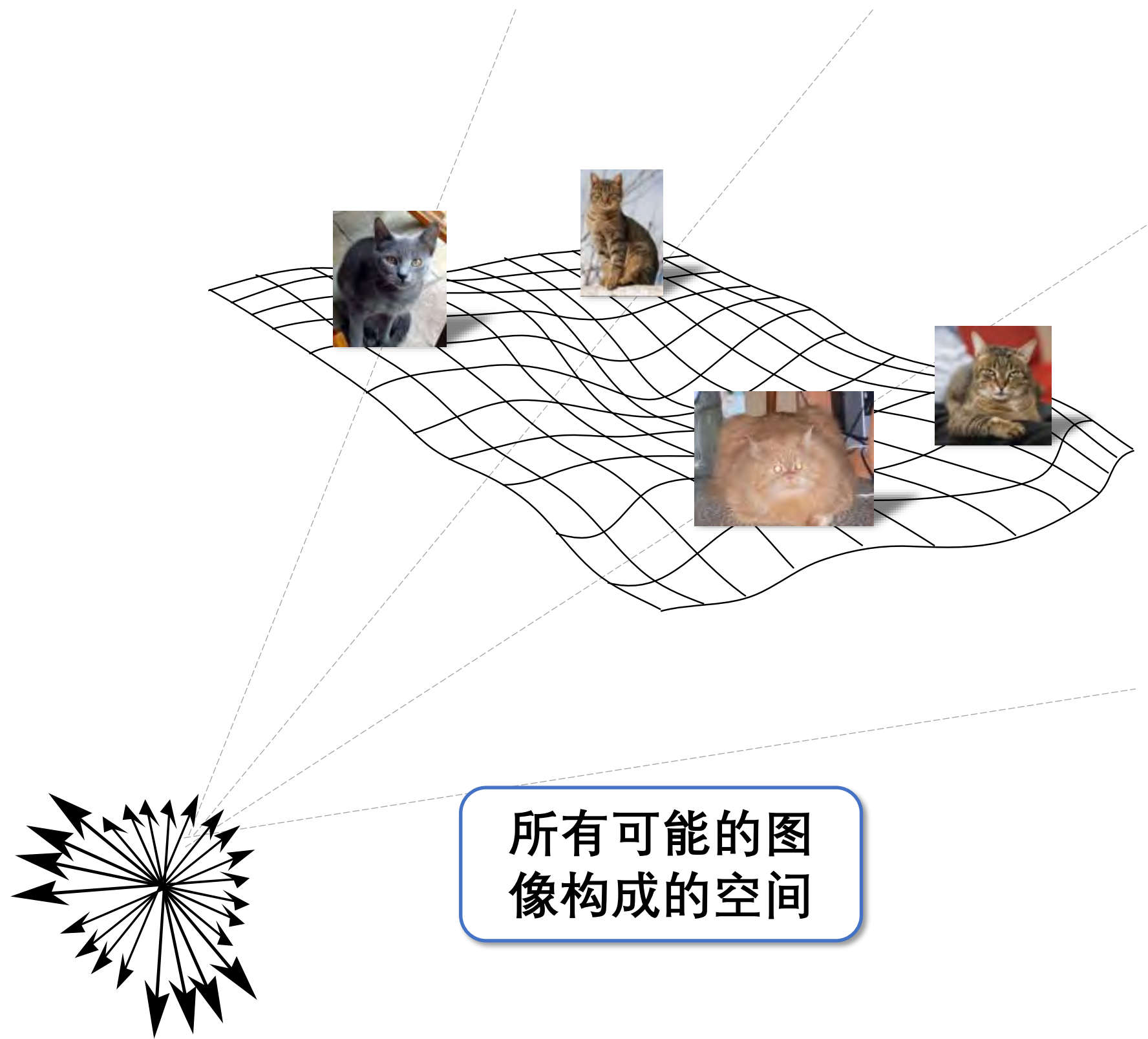
- 为每个类别定义一个概率分布。
 - 来允许toaster-cat的存在。



所有可能的图像构成的空间

图像的高维性与降维表示

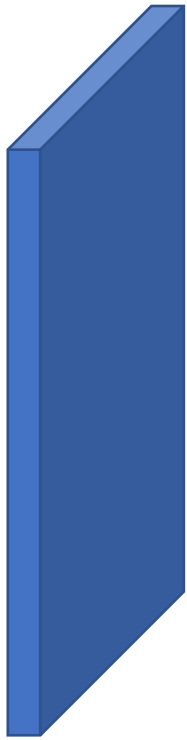
- 一个图像有多少维度？
 - 图像的所有像素和它们的颜色通道。
 - iPhone X 照片：
 - 4032 x 3024 pixels
 - 3 colors
 - 36,578,304 pixels (36.5 Mega pixels)
- 在实际应用中，虽然图像可能具有高度的维度，但许多图像都倾向于聚集在某种低维结构或流形上。
- 降维可以帮助我们提取图像的关键特征，并用更简洁的方式表示图像。



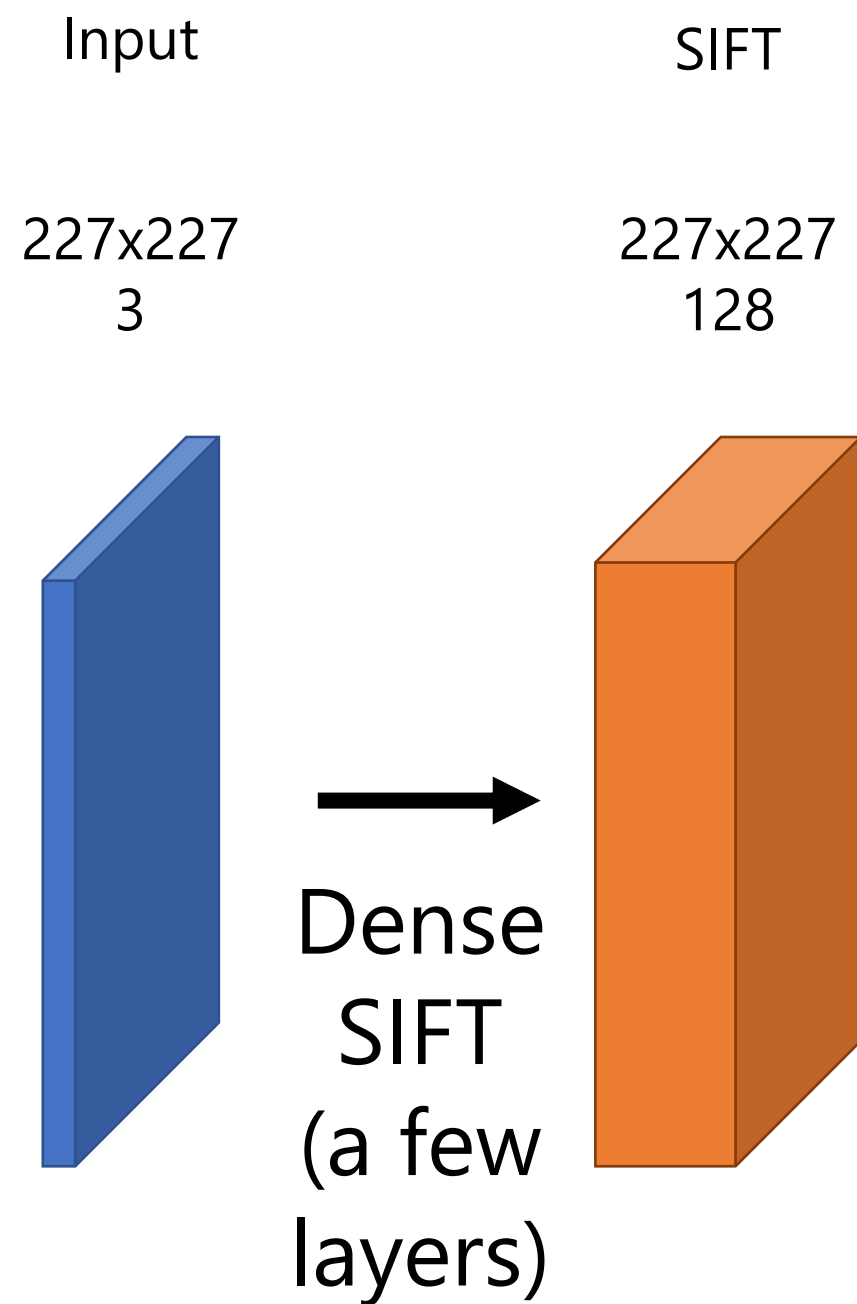
回顾：分类识别 传统模型 与 深度学习

Input

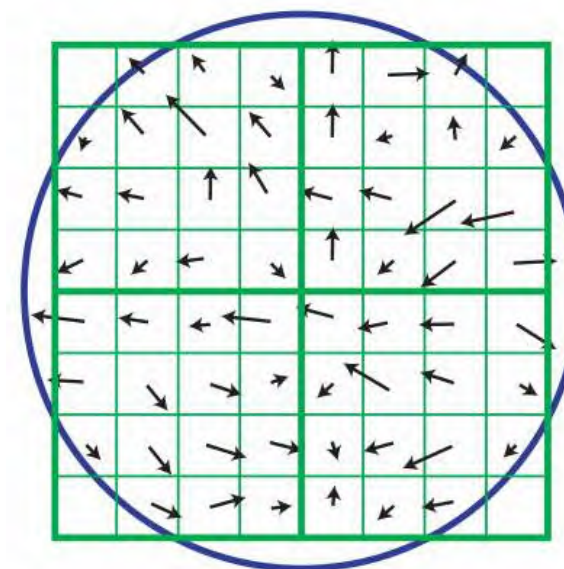
227x227
3



回顾：分类识别 传统模型 与 深度学习



回顾：可以基于图像梯度直方图计算描述子。在每个像素处密集地执行。

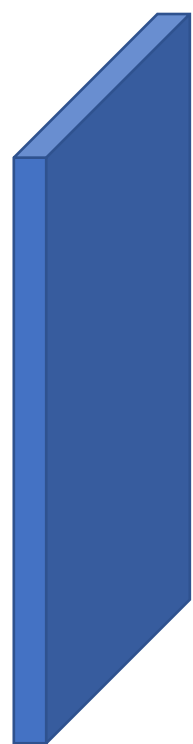


回顾：分类识别 传统模型 与 深度学习

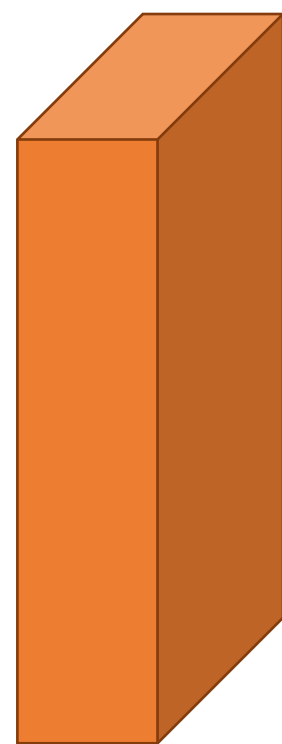
Input
227x227
3

SIFT
227x227
128

Bag of Words
HxW
#codewords

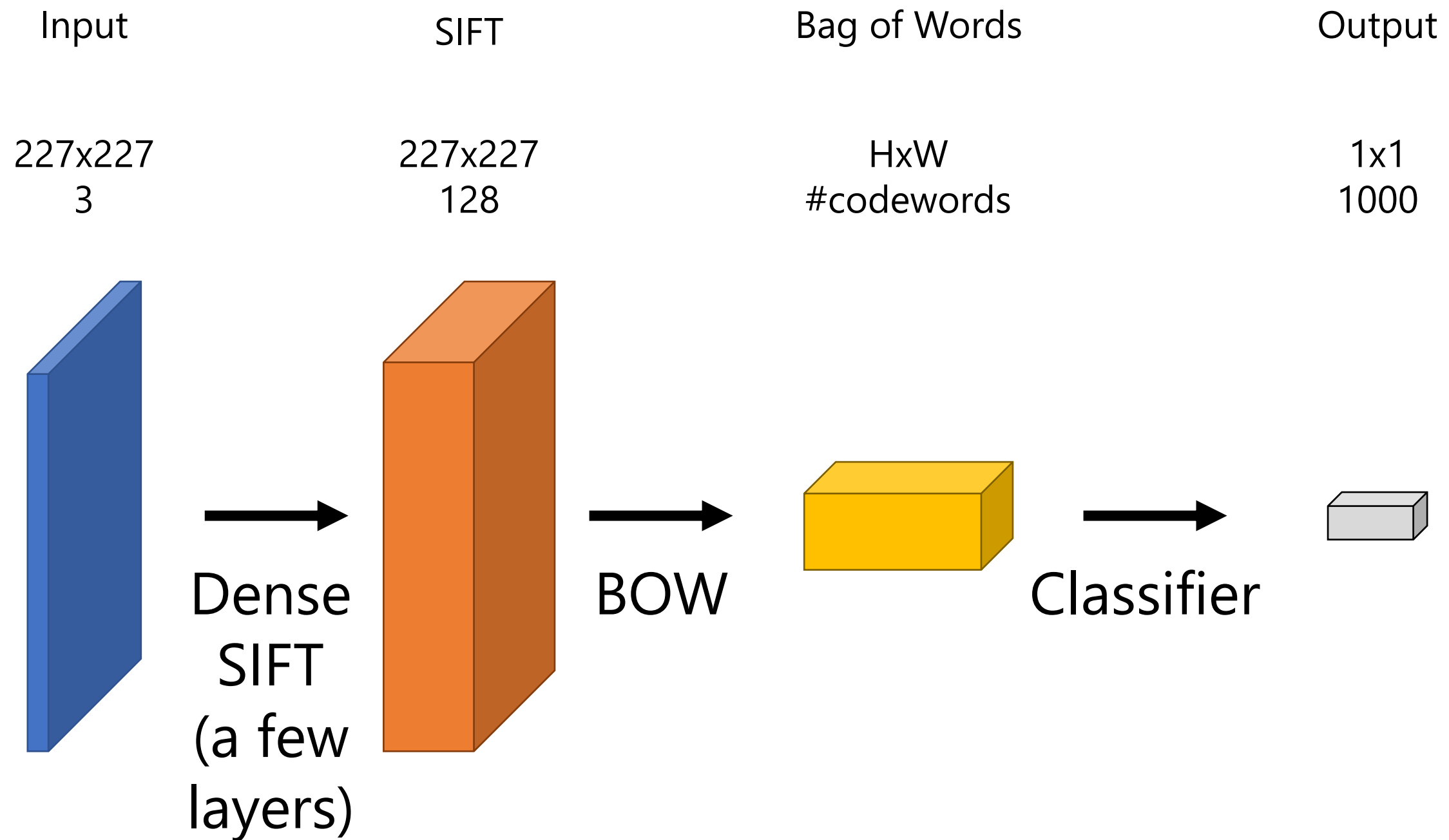


→
Dense
SIFT
(a few
layers)

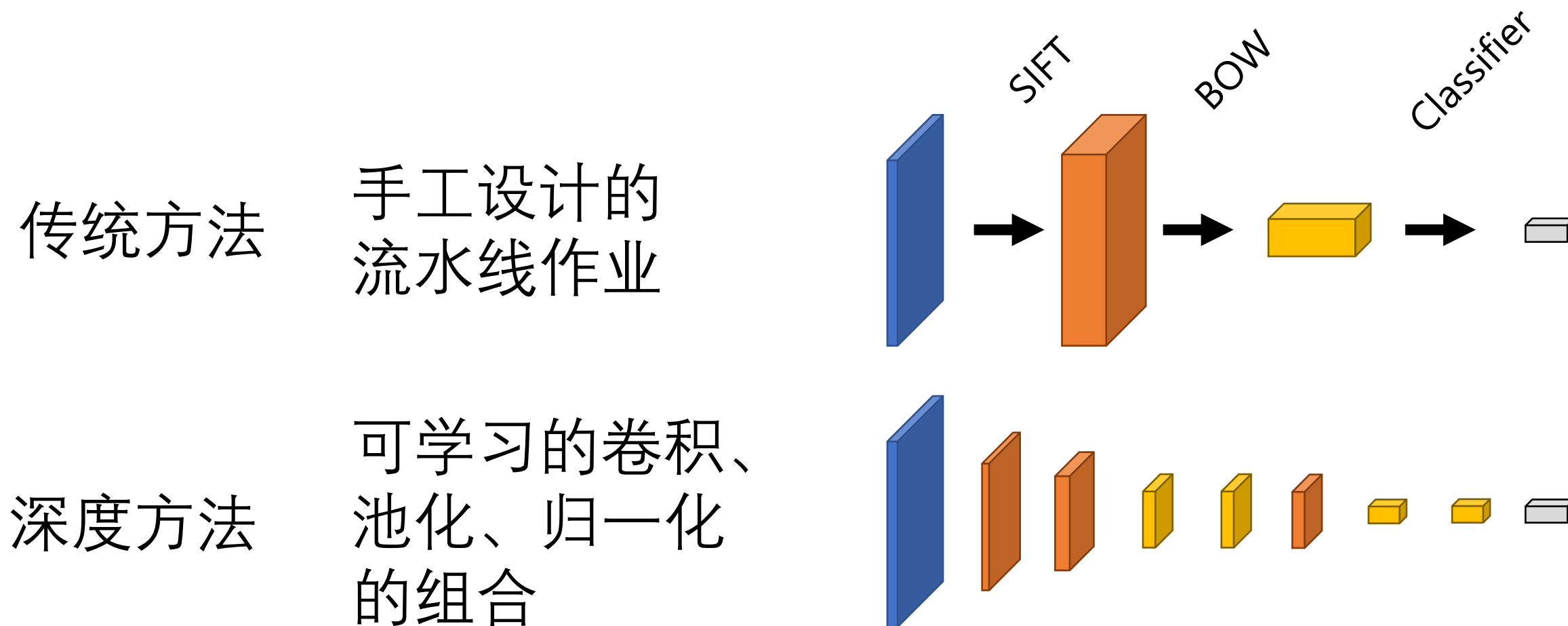


可以对SIFT执行类似词袋的技术(bag-of-words) 同时考虑到描述子在图像中的空间位置,对图像的局部区域进行编码,并改进图像分类或检索的性能。

回顾：分类识别 传统模型 与 深度学习



回顾：分类识别 传统模型 与 深度学习



有什么区别？

传统步骤中的各个组件通常是独立设计和优化的，不会“相互交流”或从数据中学习大量参数。相比之下，深度学习方法中的层是通过学习数据自动优化参数的，各层之间会有信息的交流。

语义分割

回顾：计算机视觉任务

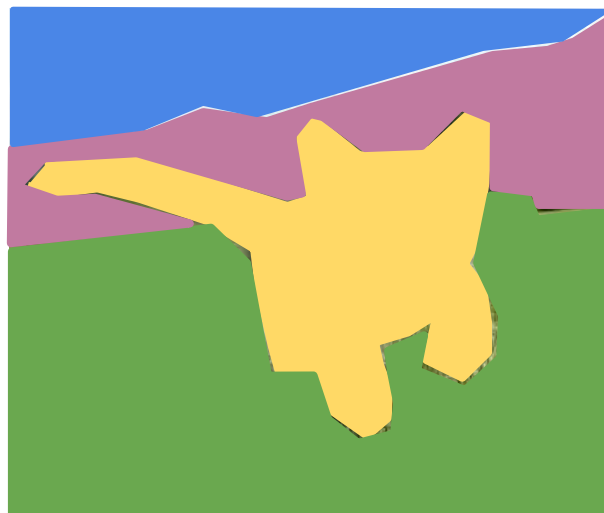
分类



CAT

无空间限定

语义分割



GRASS, CAT,
TREE, SKY

像素级预测

物体检测



DOG, DOG, CAT

多物体

实例分割



DOG, DOG, CAT

语义分割

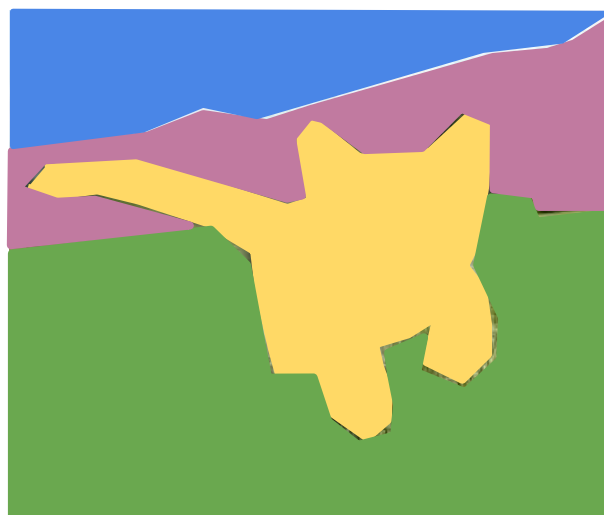
分类



CAT

无空间限定

语义分割



GRASS, CAT,
TREE, SKY

像素级预测

物体检测



DOG, DOG, CAT

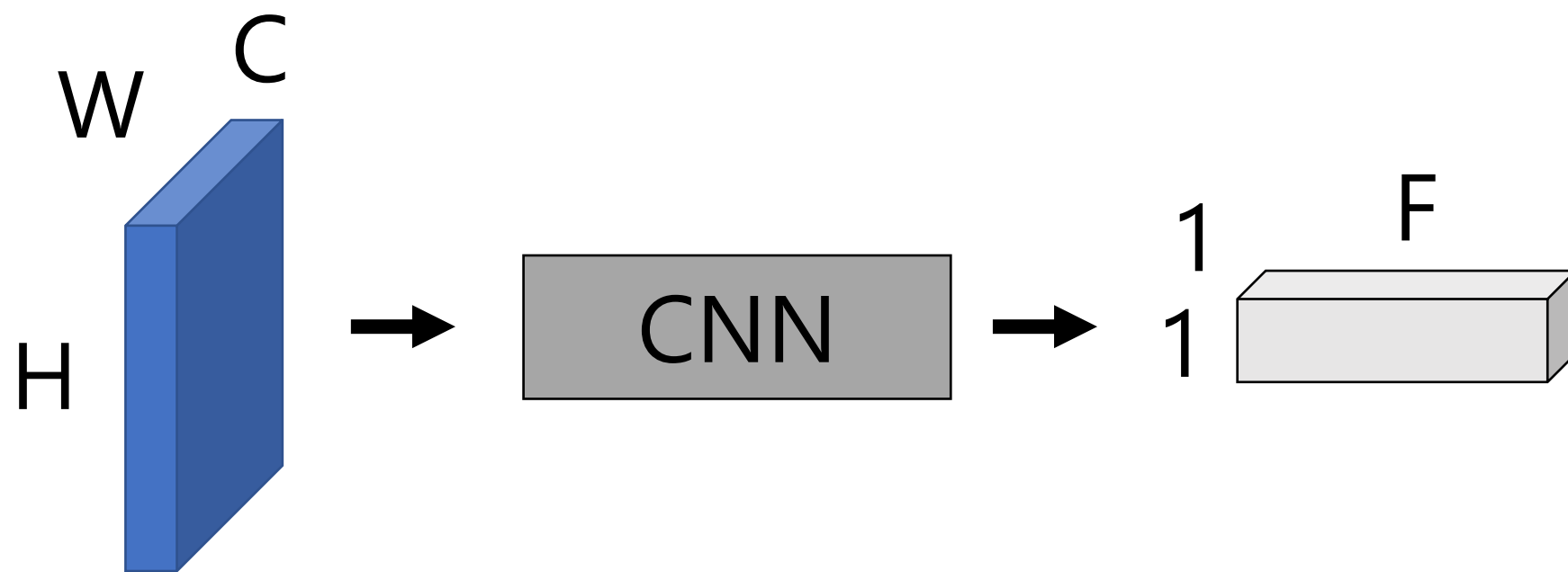
多物体

实例分割



DOG, DOG, CAT

回顾：之前的CNN



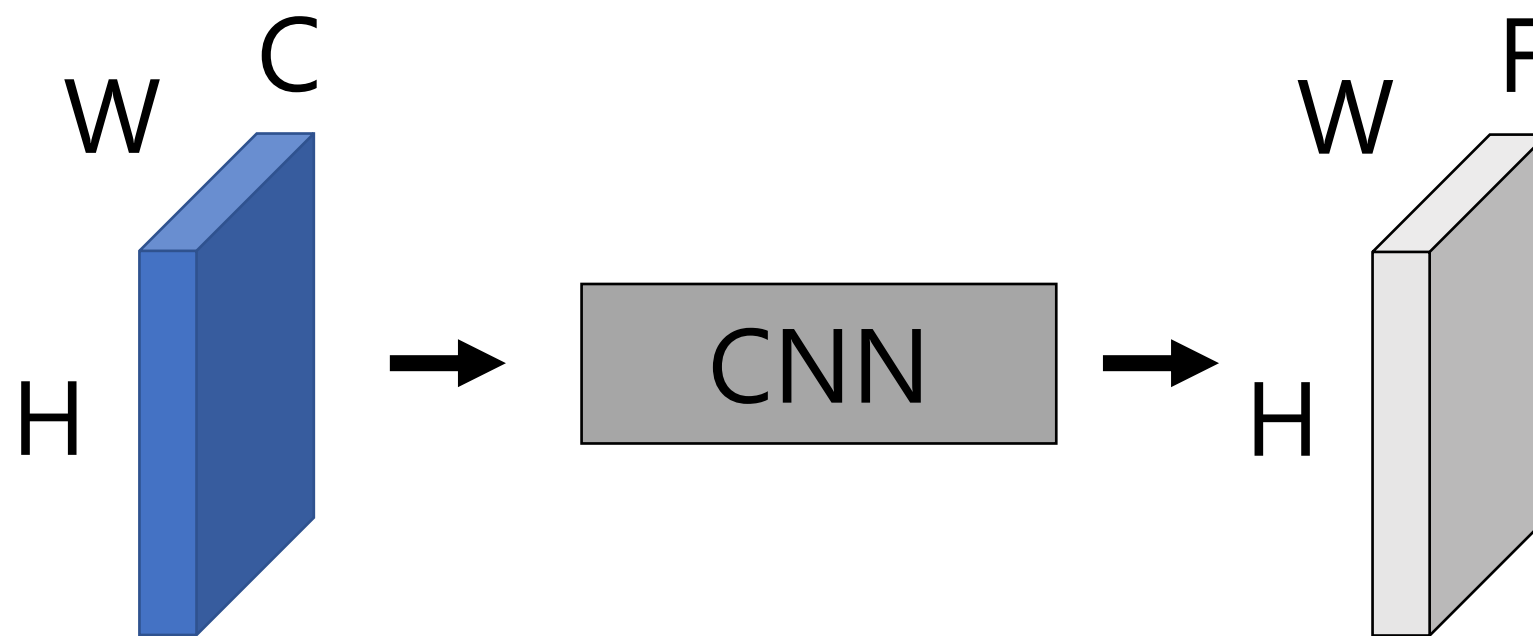
把 $H \times W$ 大小的图像转换为 F -维 向量

这是猫吗？

摄像机距离图像多远？

这是一张AI生成图像吗？

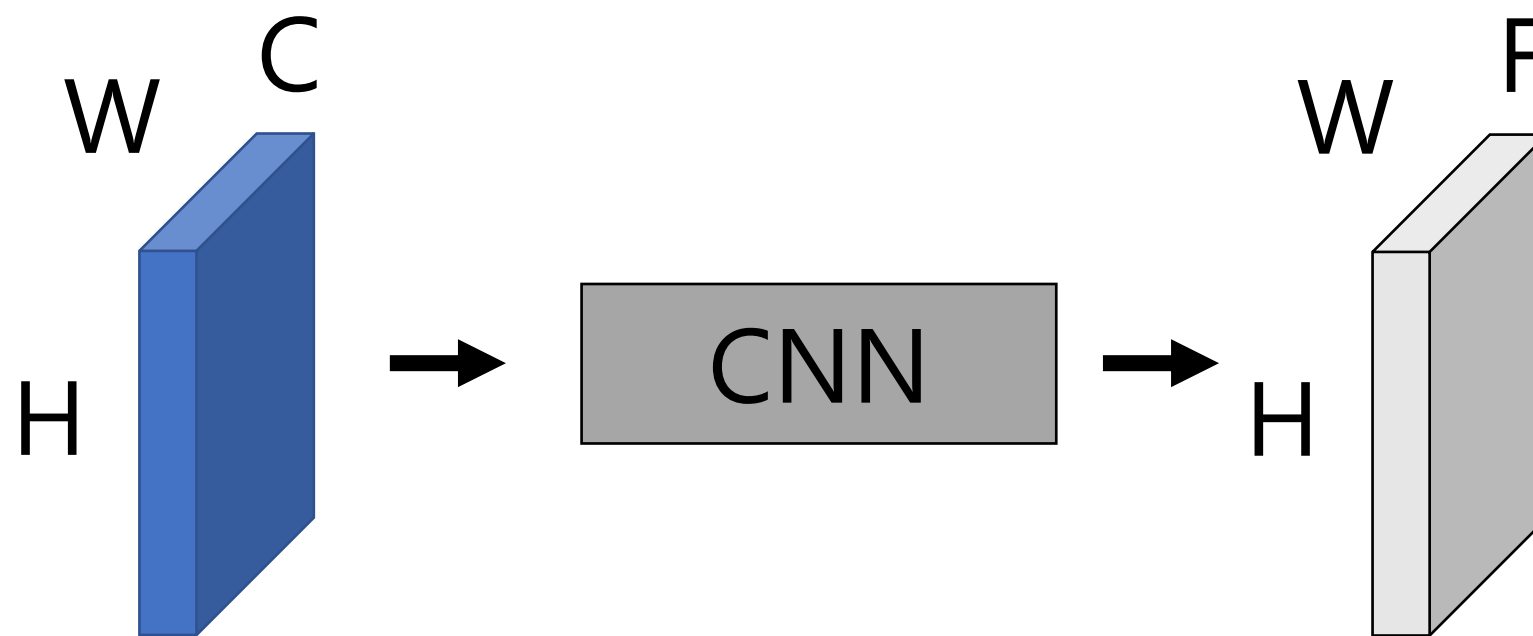
现在的任务



把 $H \times W$ 大小的图像的每个像素转换为 F -维 向量

哪些像素呈现一只猫?
每个像素距离摄像机多远?
哪些像素是伪造的?

语义分割



今天的主题

- 为每个像素提供一个 F 维的向量
 - 用于预测 F 个类别的概率
- 在**每个**像素上计算损失，回传梯度

语义分割

每个像素都有它的标签，包括背景，和未知

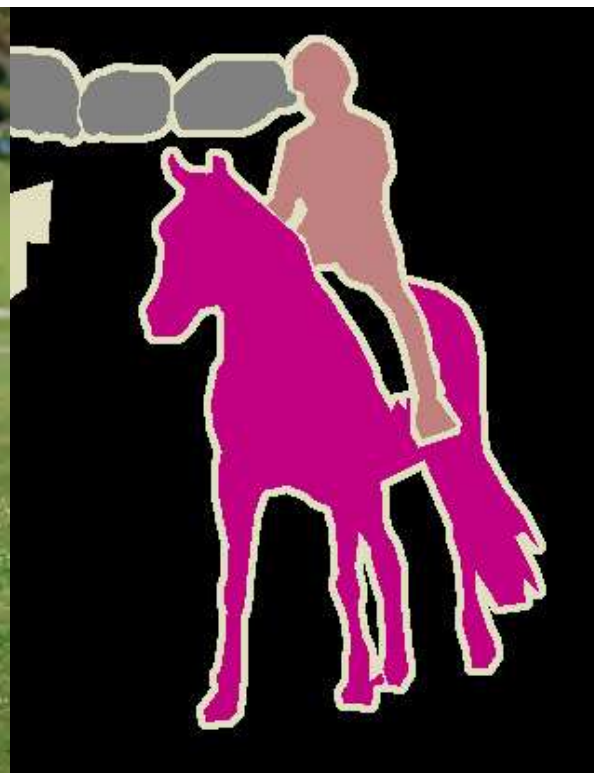
不同标签用不同颜色表示

不会区分相同类别下不同的实体 (semantic vs. instance)

Input



Label



Input



Label



语义分割

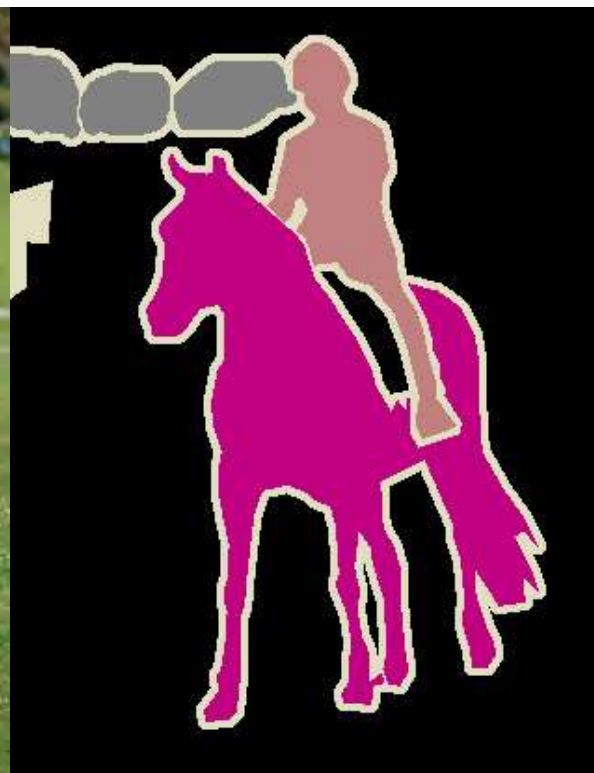
“Semantic” 语义: 不要太在意这个没有什么特殊含义的词
lol.

区分不同类型的物体, 记得其和 instance 实体的区别.

Input



Label



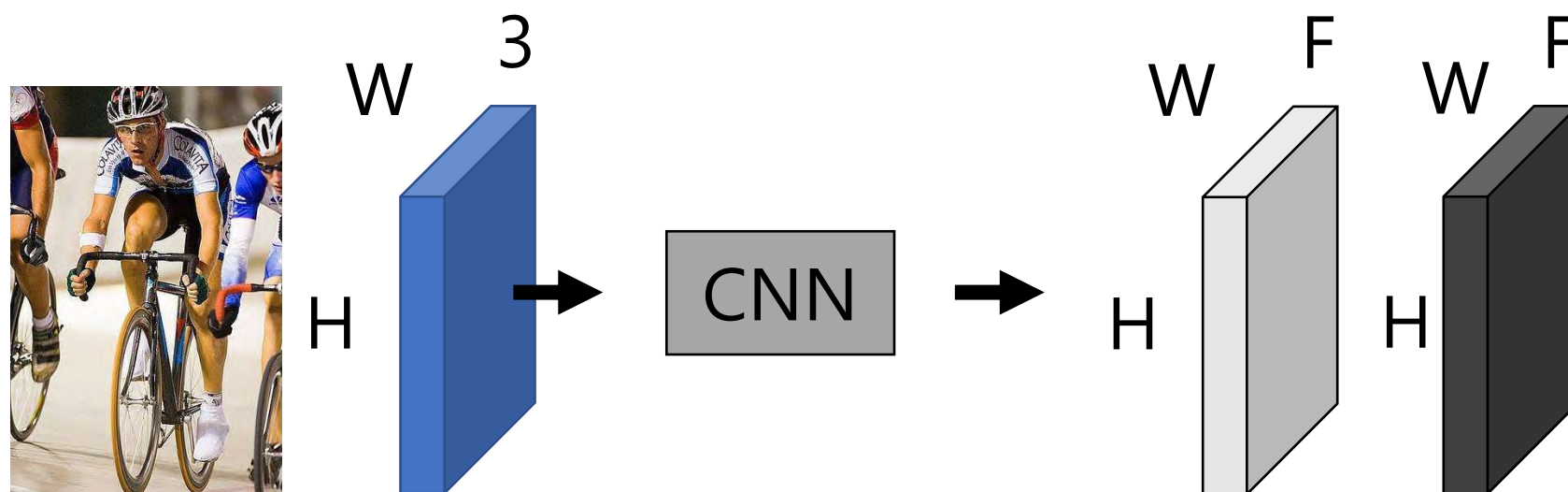
Input



Label



语义分割



在每个像素上做
F-分类交叉熵损失：
$$-\log \left(\frac{\exp((Wx)_{y_i})}{\sum_k \exp((Wx)_k)} \right)$$

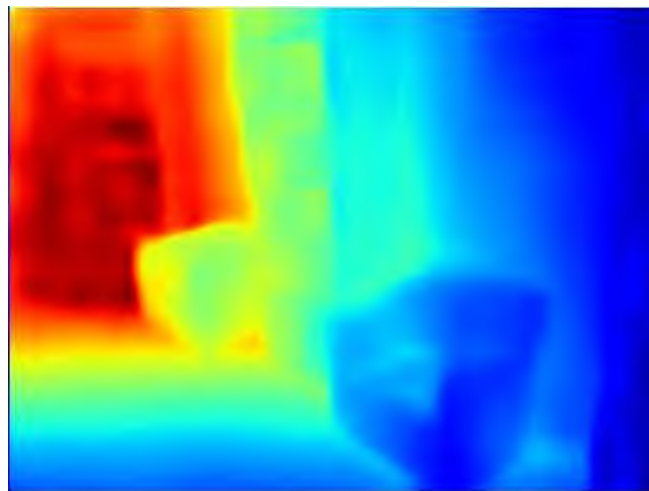
其他任务——深度预测

给定深度图标注，训练回归网络 优化 $\|z_i - \hat{z}_i\|$ ，对于像素 i ， z_i 为真实深度 \hat{z}_i 为预测深度).

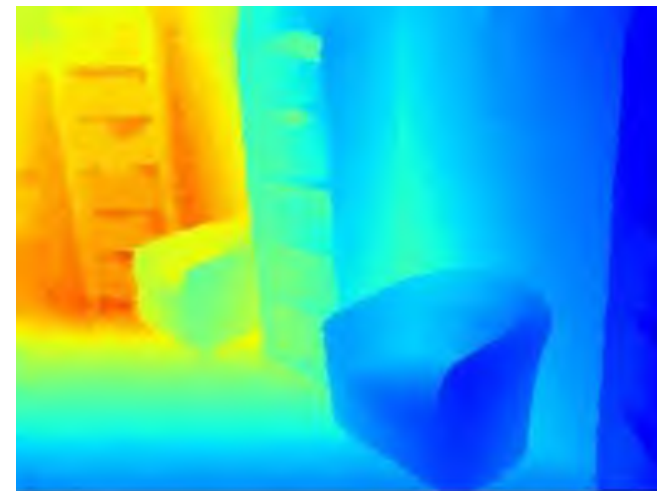
Input HxWx3
RGB Image



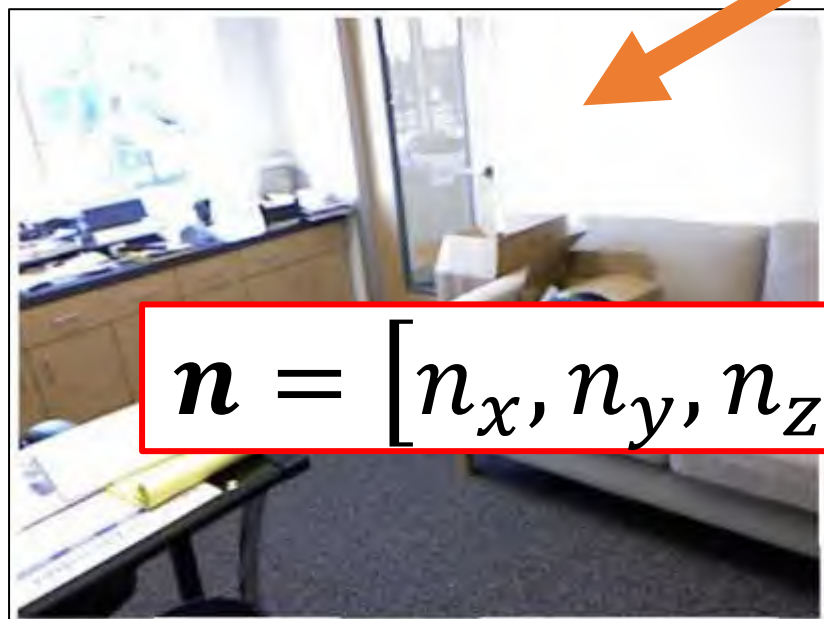
Output HxWx1
Depth Image



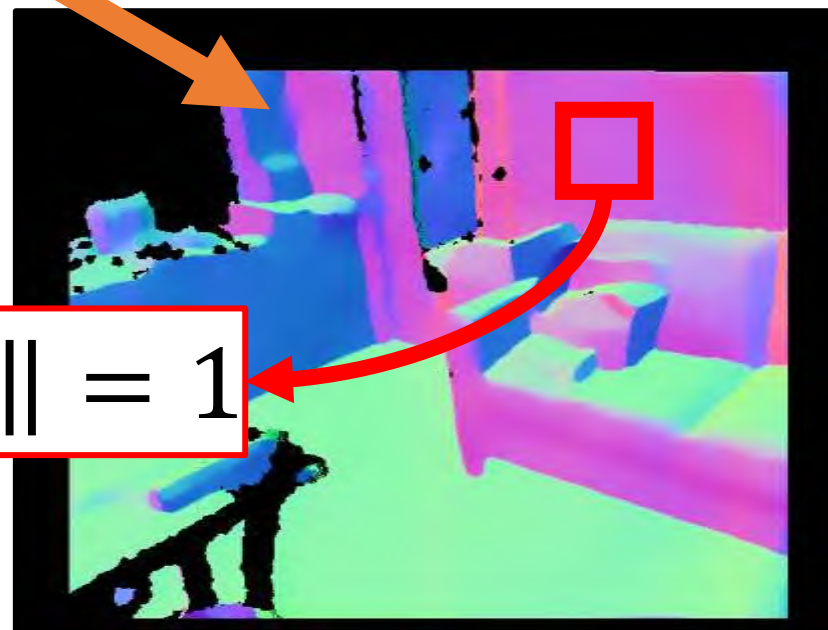
True HxWx1
Depth Image



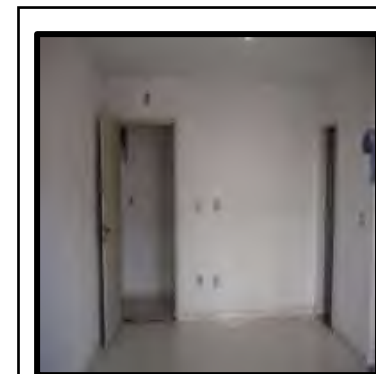
其他任务——Surface Normals 表面预测



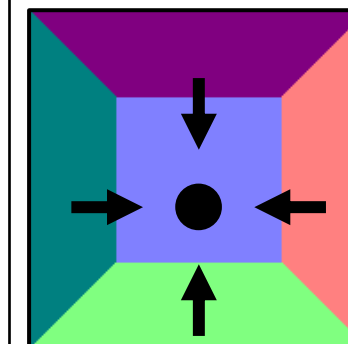
Color Image



Normals



Room



Legend

表面预测

训练网络最小化 $\|n_i - \hat{n}_i\|$,
 n_i 是真实法向量, \hat{n}_i 是对像素 i 的预测法向量.

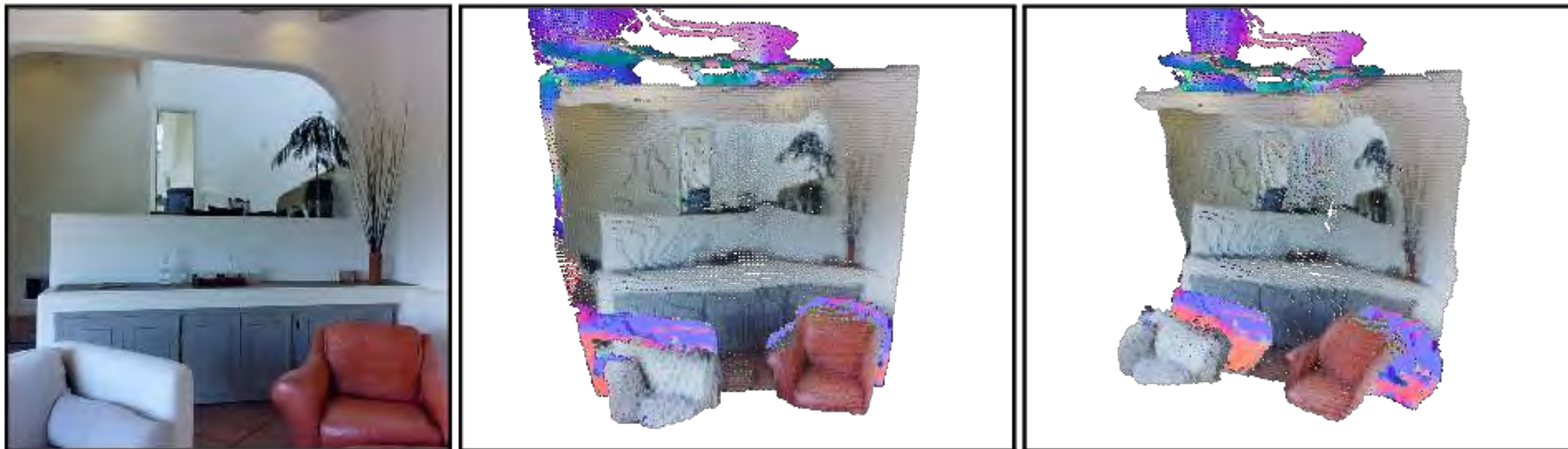
Input: HxWx3
RGB Image



Output: HxWx3
Normals



3D 重建



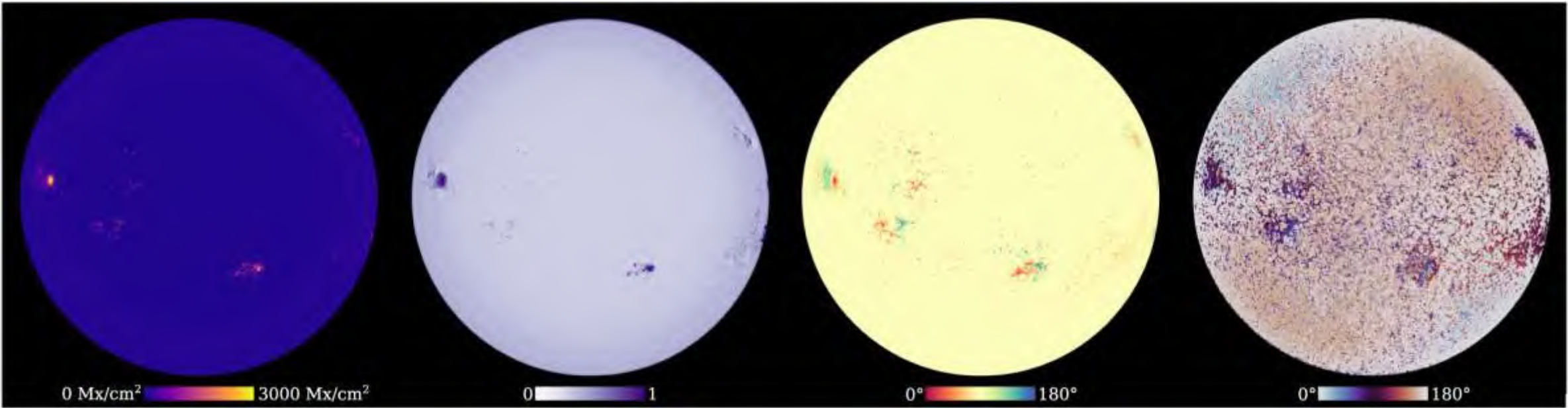
Result credit: N. Kulkarni, J. Johnson, D.F. Fouhey, *What's Behind The Couch: Directed Ray Distance Functions for 3D Reconstruction*. ???, 2022.

人体姿态估计

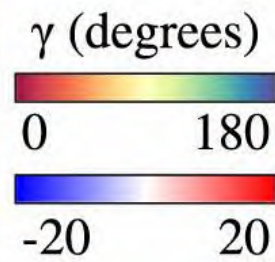
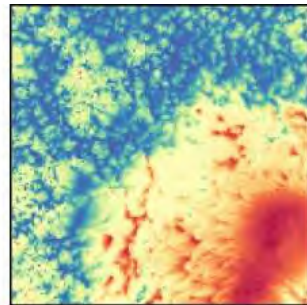
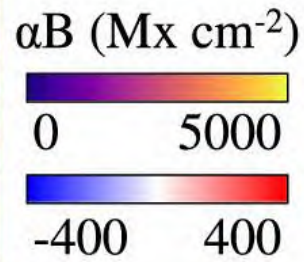
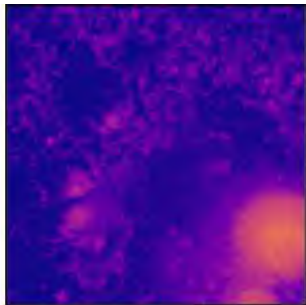
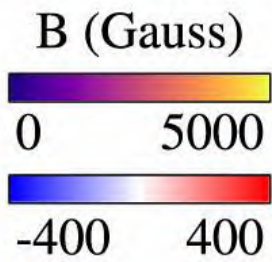
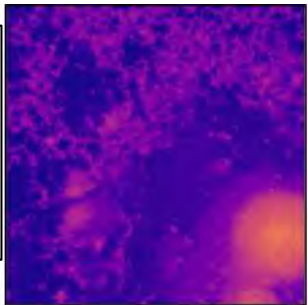


Result credit: Z. Cao et al. *Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields*. CVPR 2017.

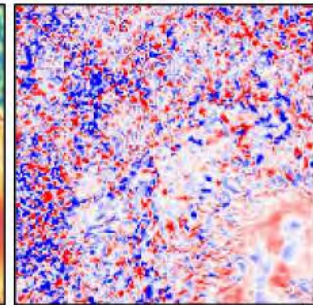
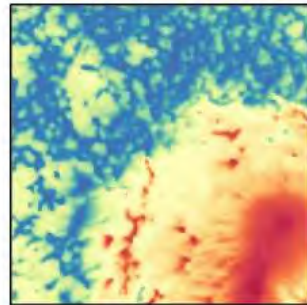
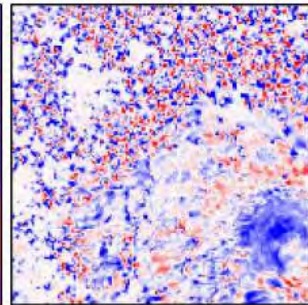
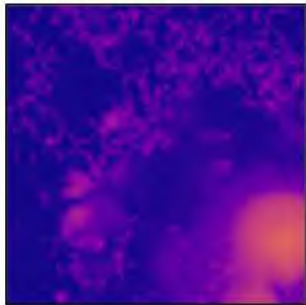
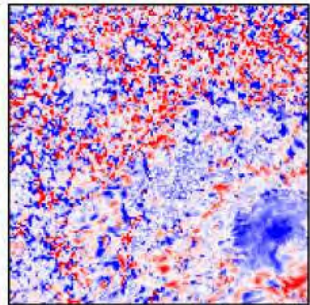
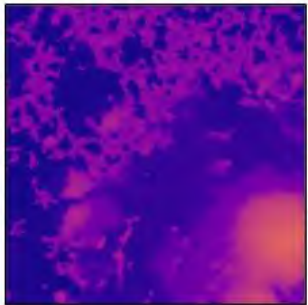
太阳



Normal
Physics
Model



Deep
Learning



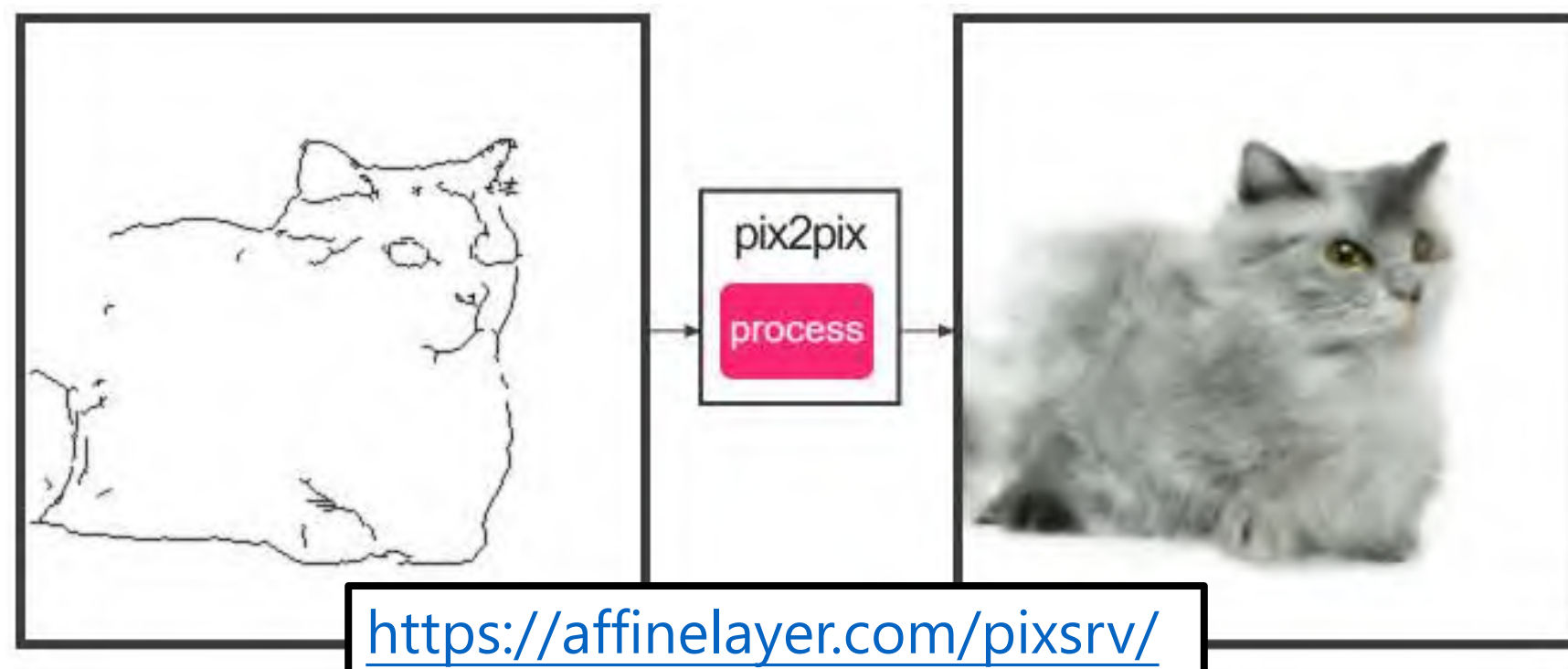
素描生成

训练网络最小化 $\|I_j - \hat{I}_j\|$

I_j 是真实图 \hat{I}_j 是在像素 j 的预测值.

Input: HxWx1
Sketch Image

Output: HxWx3
Image



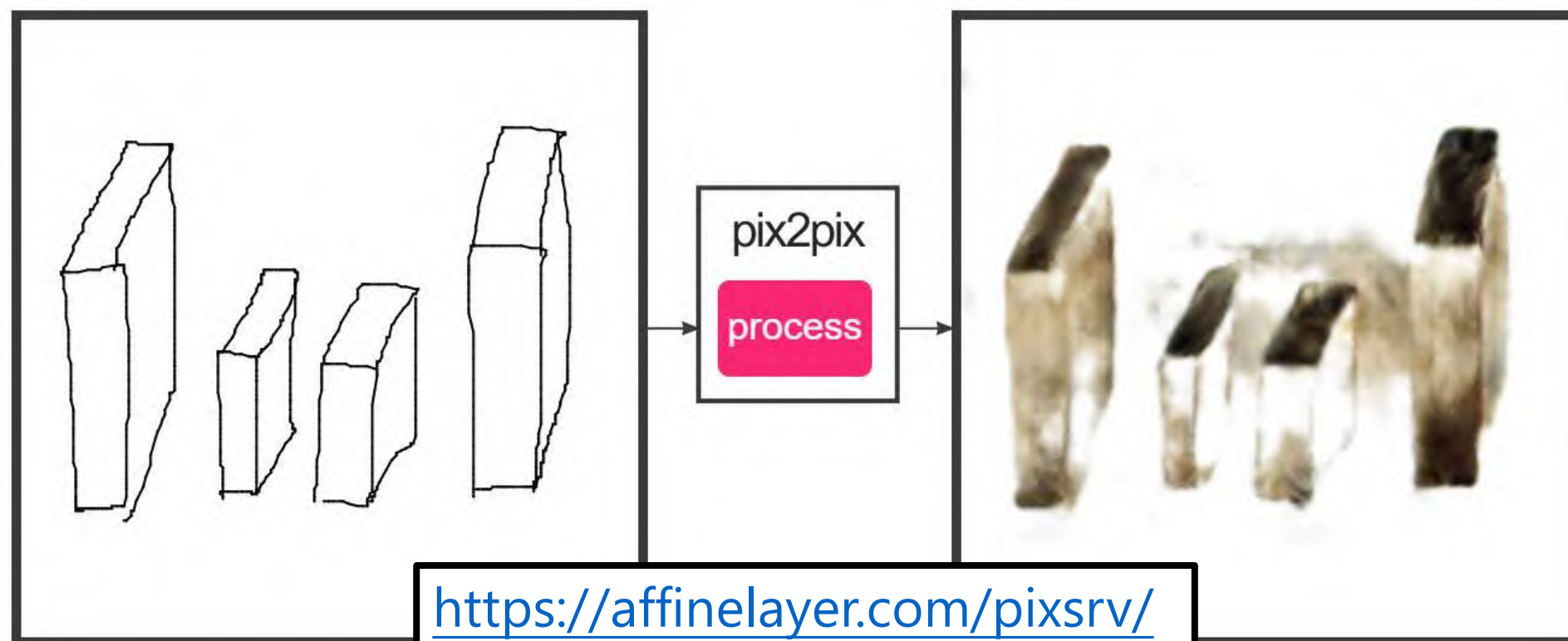
素描生成

训练网络最小化 $\|I_j - \hat{I}_j\|$

I_j 是真实图 \hat{I}_j 是在像素 j 的预测值.

Input: HxWx1
Sketch Image

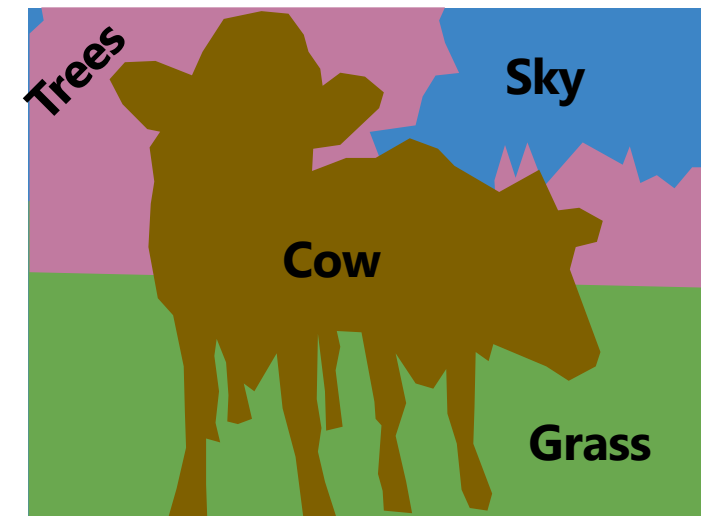
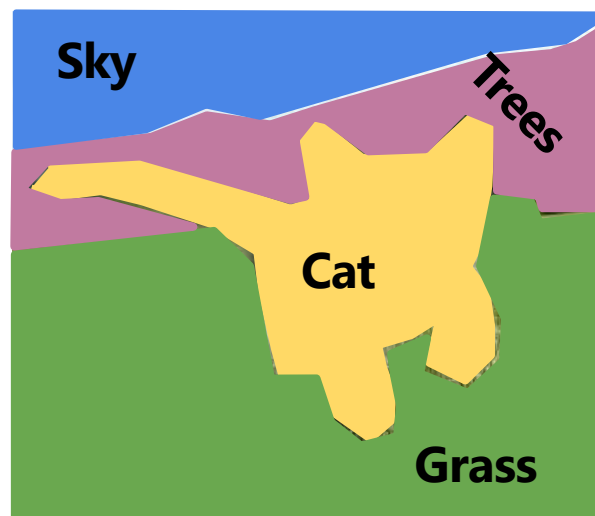
Output: HxWx3
Image



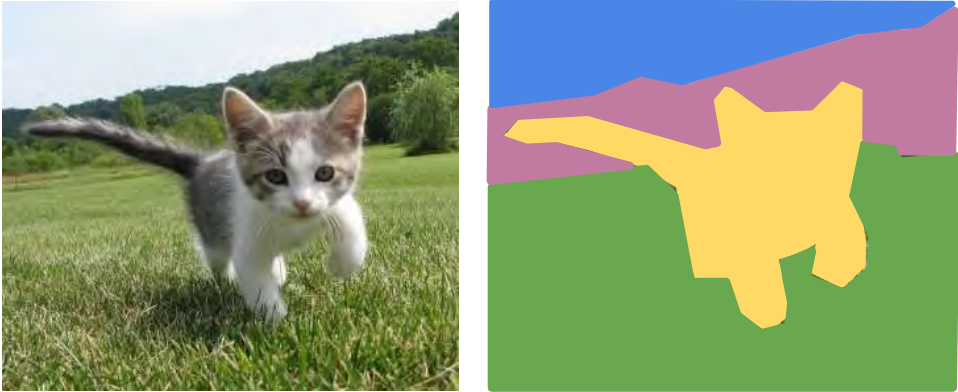
语义分割: 问题定义

每个像素点都有一个标签

不关注物体,
只关注每个像素点



语义分割: 问题定义



GRASS, CAT,
TREE, SKY, ...

成对的训练数据:
包括每个像素点的
标注



测试时预测
每个像素的类别

为什么这个任务很难？

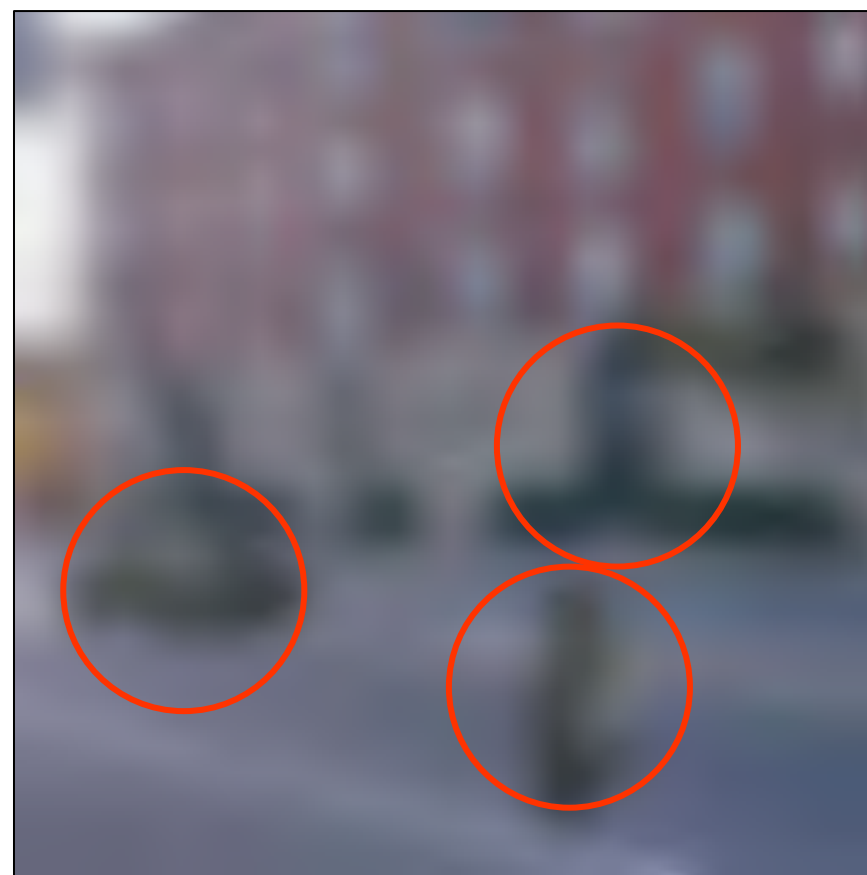
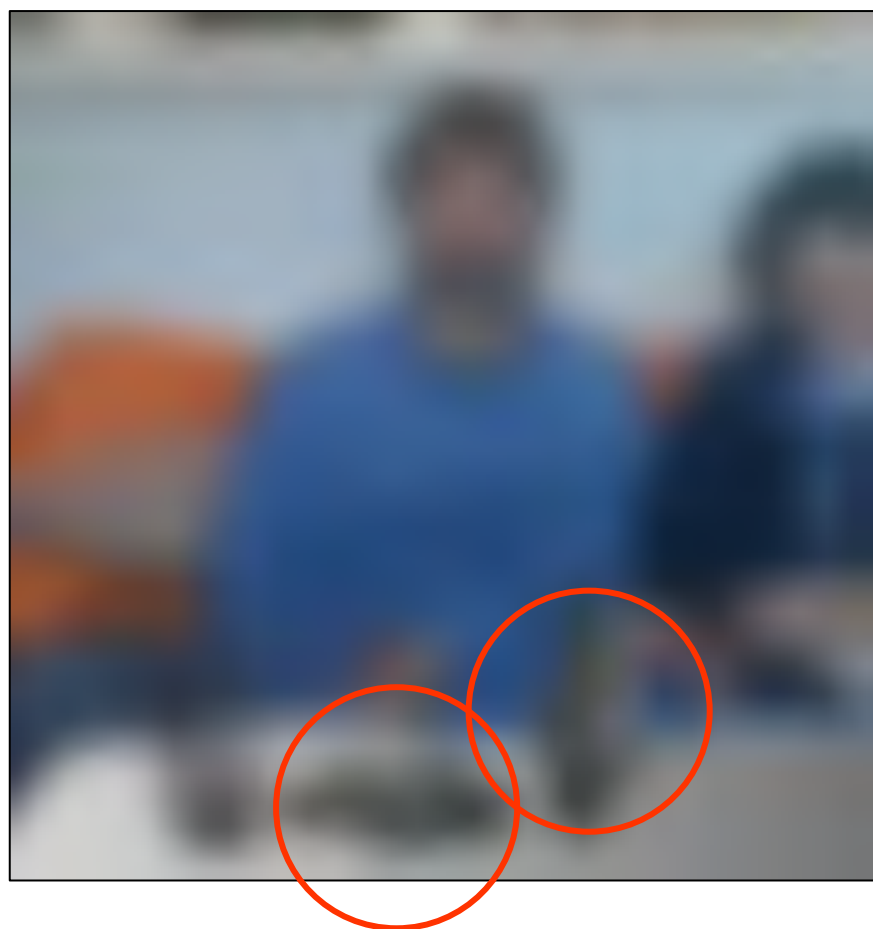
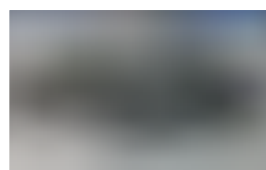
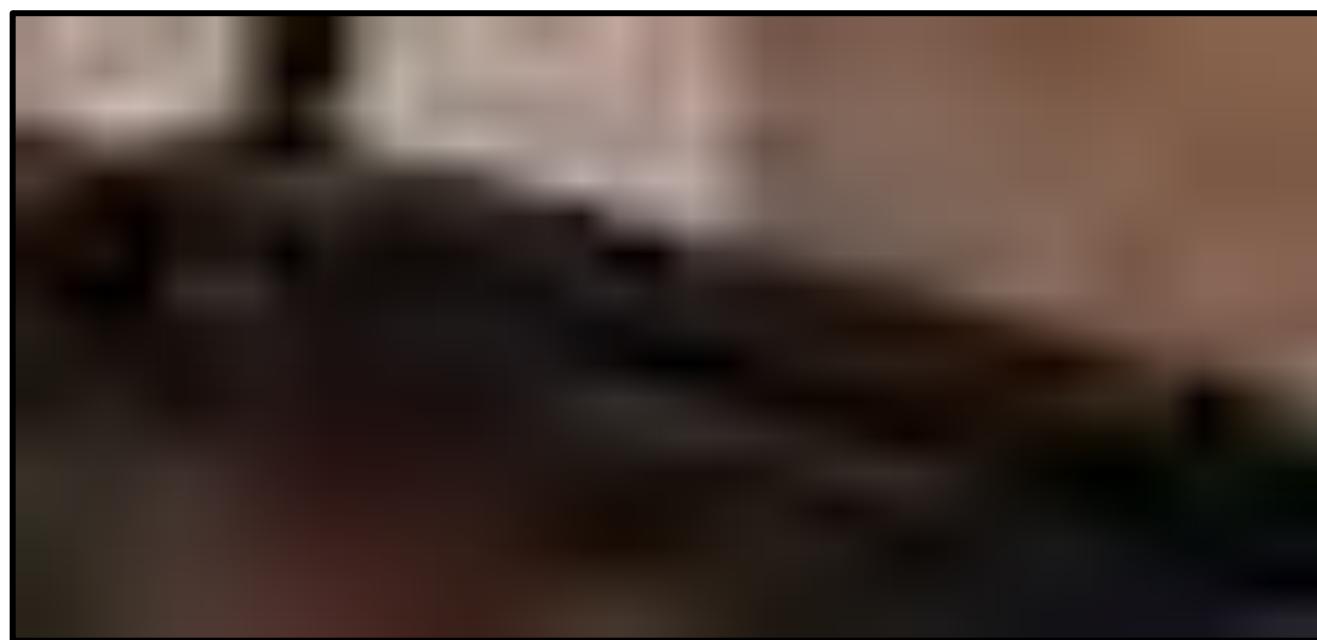


Image credit: A. Torralba

为什么这个任务很难?

这是啥?



(1) Keyboard?

(2) Hammer?

(3) Old cell phone?

(4) Xbox controller?

为什么这个任务很难?

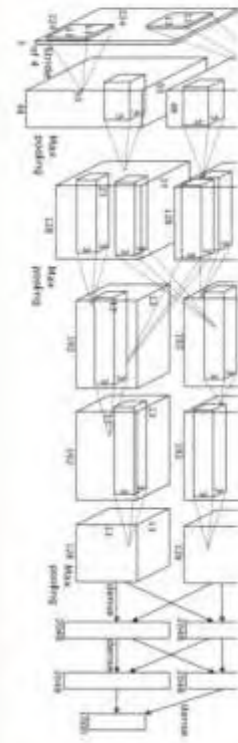


Image credit: COCO dataset

回顾——CNN训练、 语义分割

CNN网络结构

“AlexNet”



[Krizhevsky et al. NIPS 2012]

“GoogLeNet”



[Szegedy et al. CVPR 2015]

“VGG Net”



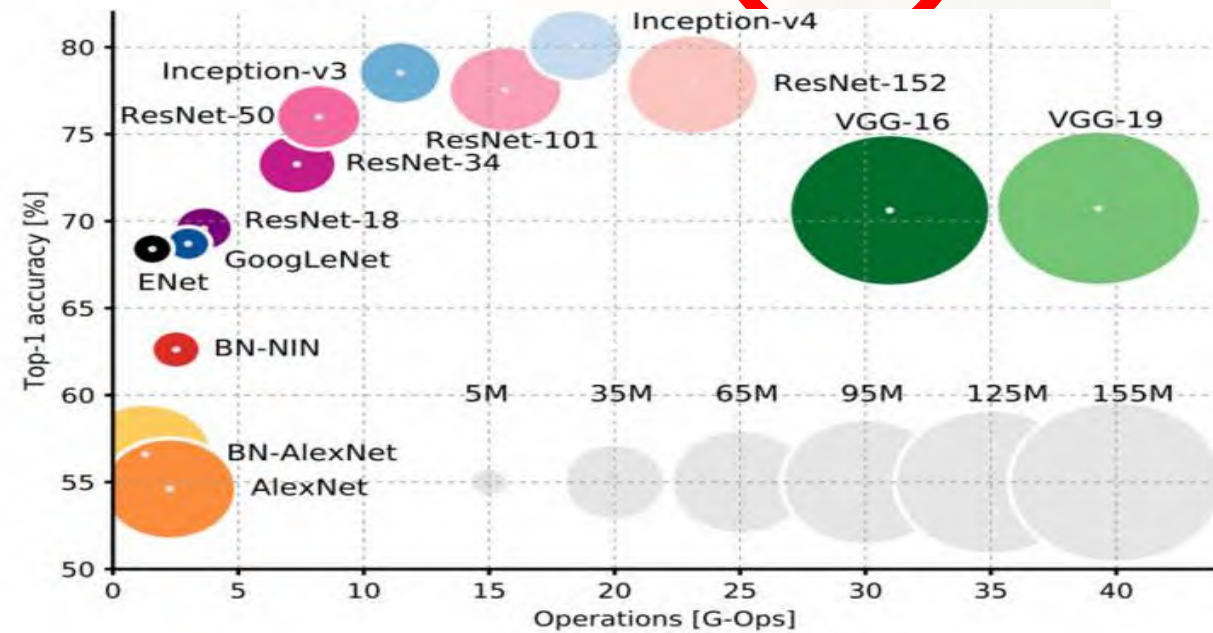
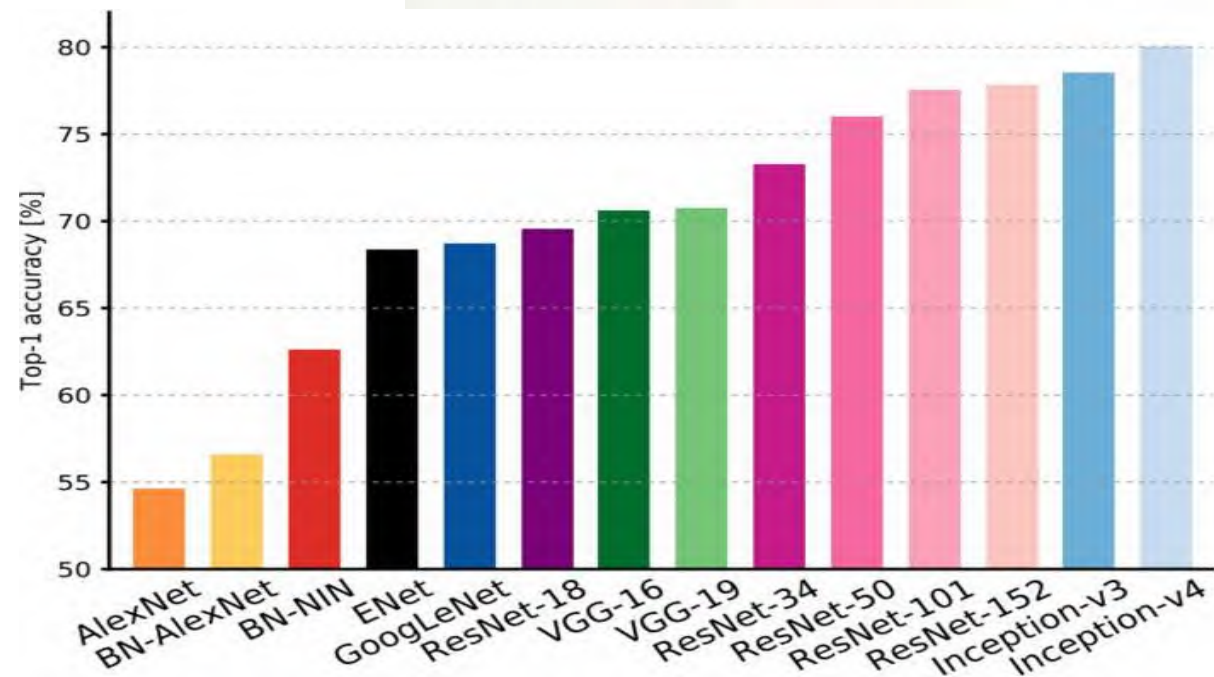
[Simonyan & Zisserman, ICLR 2015]

“ResNet”



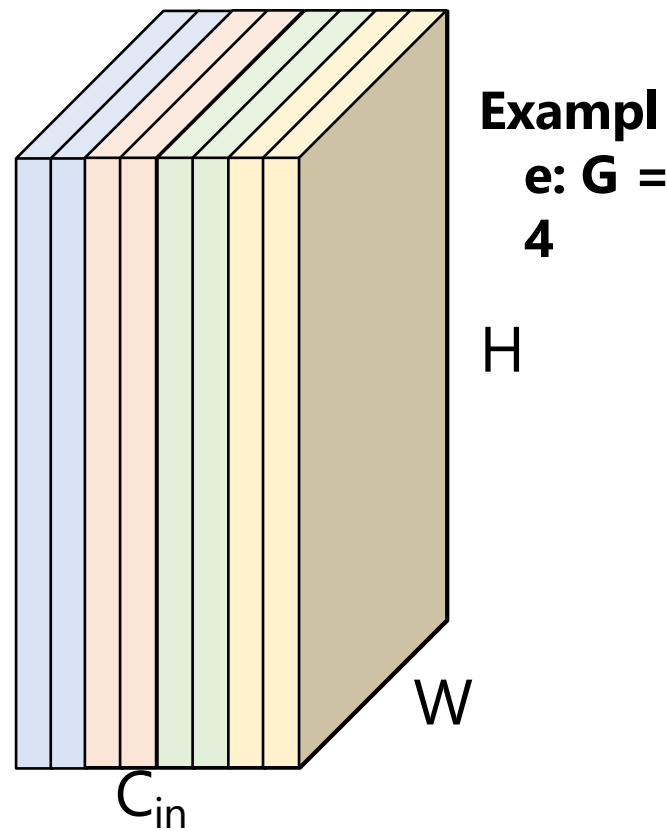
[He et al. CVPR 2016]

最常用的选择



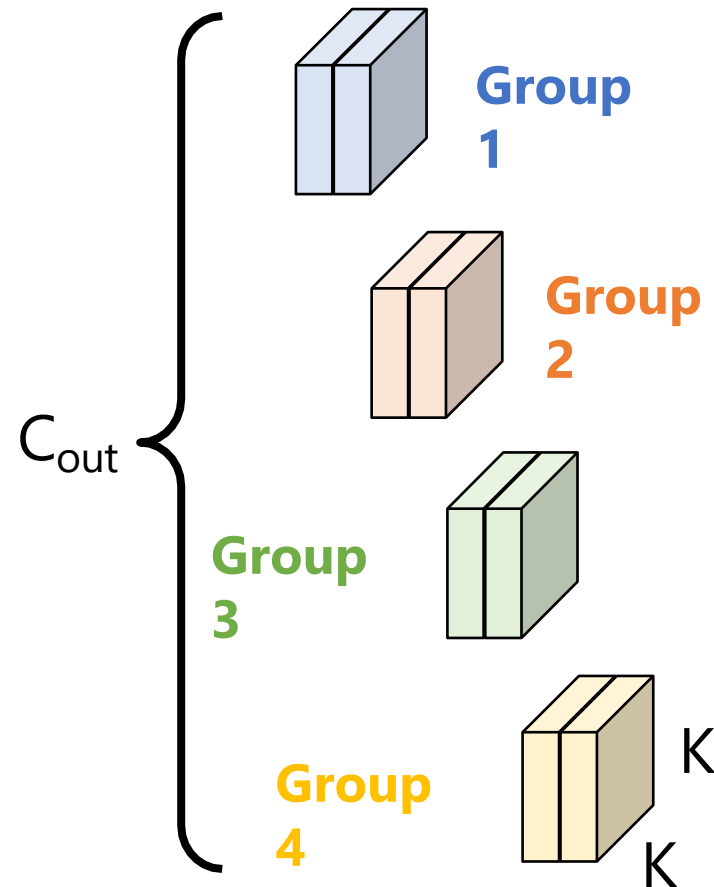
Group Convolution 组卷积层

把输入通道分为 G 组
每组 (C_{in}/G) 个通道



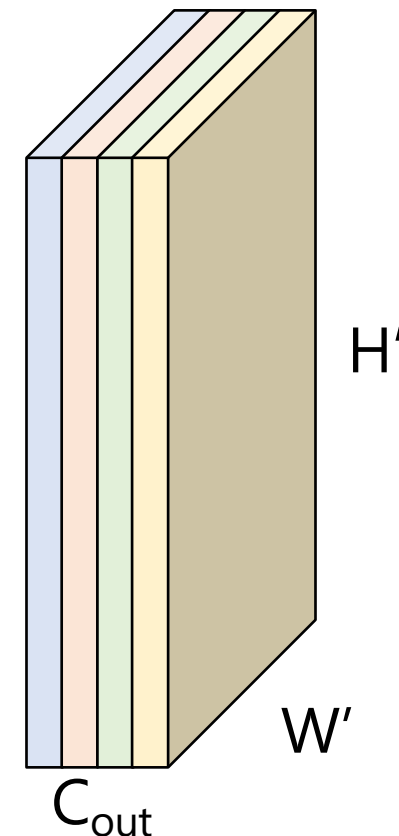
输入: $C_{in} \times H \times W$

滤波同样分为 G 组，每组只看自己组内的输入通道



权重: $C_{out} \times (C_{in} / G) \times H \times W$

输出依照卷积滤波，只看某组内的输入通道



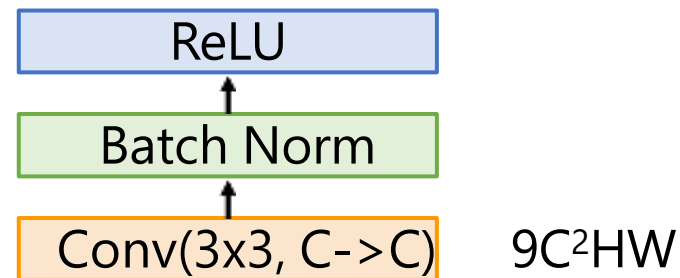
输出: $C_{out} \times H' \times W'$

MobileNet (2017)

- 目标: 减少网络参数
- 解决方案: Depthwise separable convolutions “分离卷积”

Standard Convolution Block

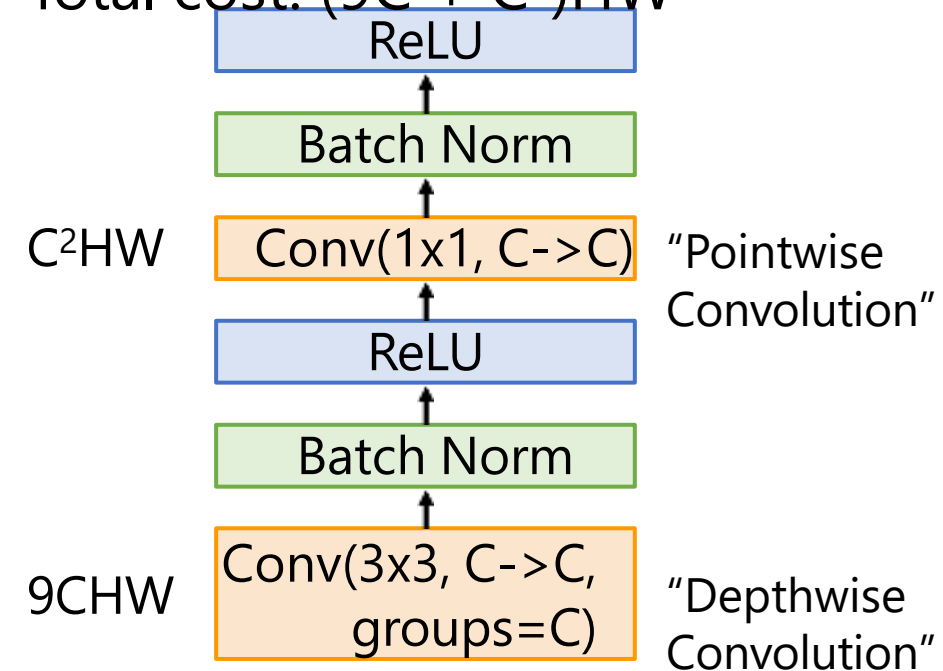
Total cost: $9C^2HW$



$$\begin{aligned}\text{Speedup} &= 9C^2 / (9C + C^2) \\ &= 9C / (9 + C) \\ &\Rightarrow 9 \text{ (as } C \rightarrow \text{inf)}\end{aligned}$$

Depthwise Separable Convolution

Total cost: $(9C + C^2)HW$



准确率

	Top 1 Error	Top 5 Error
Best Pre-Deep (~2012)	-	26.2%
Alexnet, 2012	43.5%	20.9%
VGG-16, 2014	28.4%	9.6%
ResNet-50, 2015	24.7%	7.8%
ResNet-152, 2015	21.7%	5.9%
ResNet-50 done better, 2018	20.7%	5.4%
Swin Transf., 2021	15.5%	-
ConvNeXt, 2022	14.5%	-
CoAtNet-7* 2021 (2B params!)	9.1%	-
Human*	-	5.1%

Many results from <https://paperswithcode.com/sota/image-classification-on-imagenet>. I am missing loads of great papers, and the numbers depend on tons of practical details. *Human – this number is from Andrej Karpathy and isn't really human performance with training but a ballpark. Resnet-50 one better = "Bag of Tricks for Image Classification with Convolutional Neural Networks", He et al.

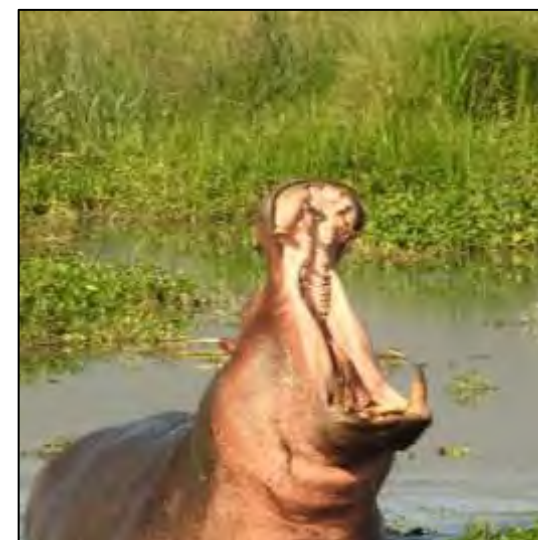
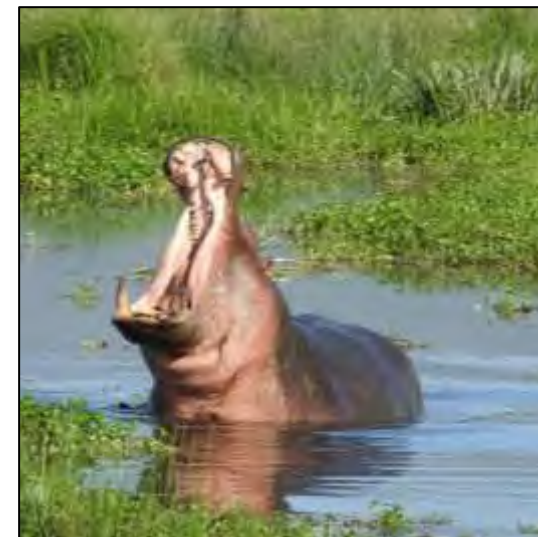
训练CNN

- 下载数据集, 准备GPU
- 初始化网络参数
- for epoch in range(epochs):
 - 打乱数据集
 - for each minibatch in dataset:
 - 加载数据
 - 计算梯度
 - 优化网络



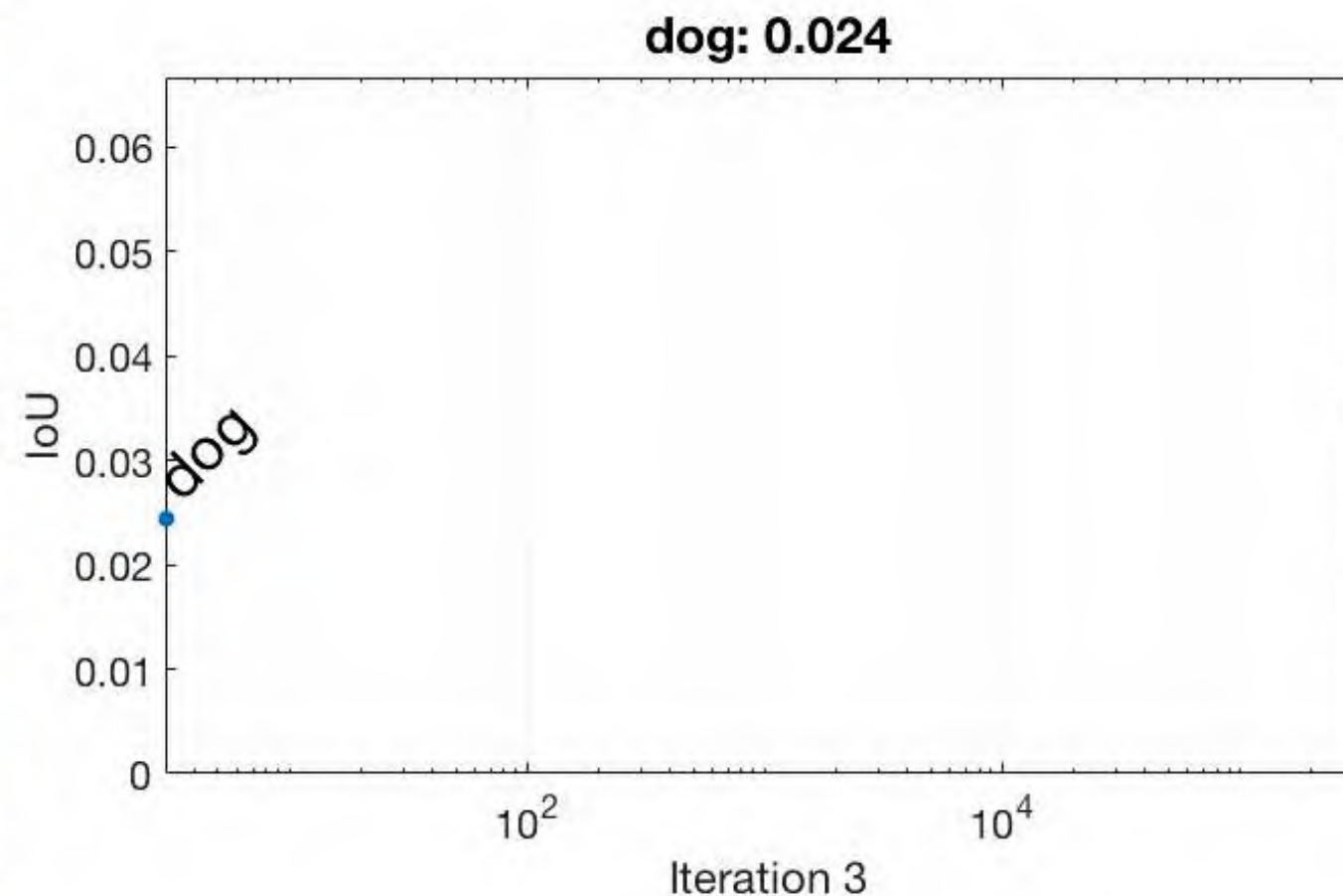
训练CNN -Augmentation 图像增广

- 应用一些不会影响图像分类结果的变换
- 请注意，一定要确保不会影响图像的表达语义



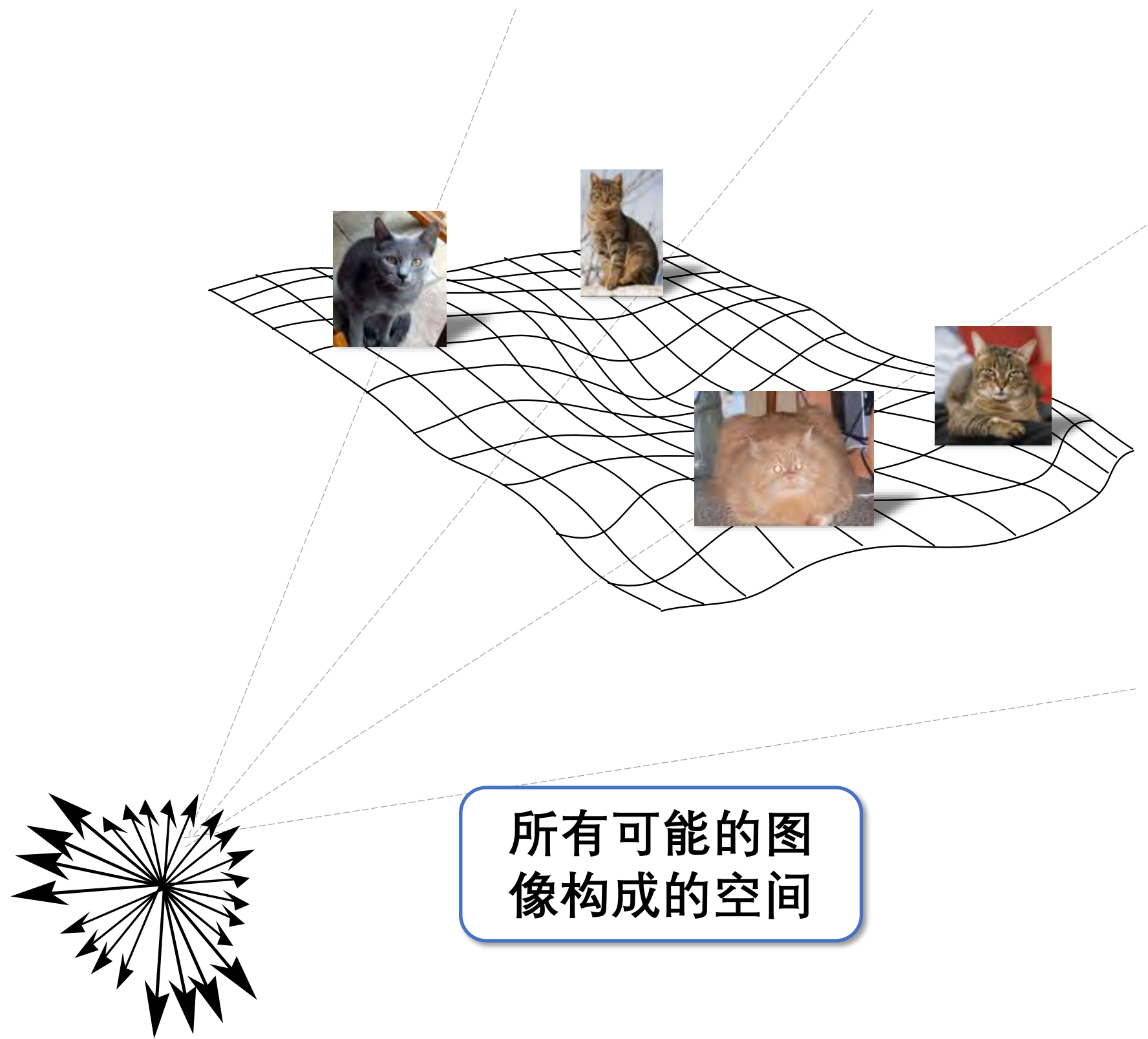
微调：迁移学习

为什么能实现？
从 物体(dog) 迁移到 场景(waterfall)

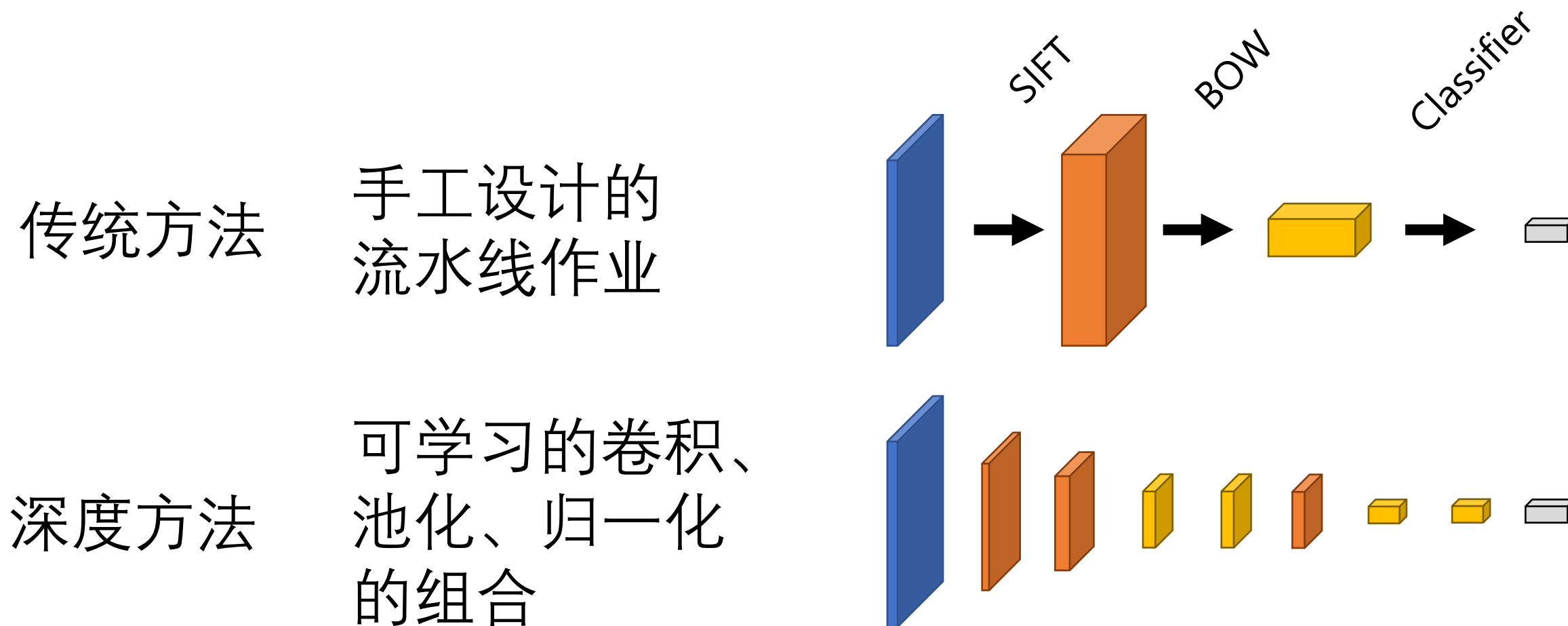


图像的高维性与 降维表示

- 一个图像有多少维度？
 - 图像的所有像素和它们的颜色通道。
 - iPhone X 照片：
 - 4032 x 3024 pixels
 - 3 colors
 - 36,578,304 pixels (36.5 Mega pixels)
- 在实际应用中，虽然图像可能具有高度的维度，但许多图像都倾向于聚集在某种低维结构或流形上。
- 降维可以帮助我们提取图像的关键特征，并用更简洁的方式表示图像。



回顾：分类识别 传统模型 与 深度学习



有什么区别？

传统步骤中的各个组件通常是独立设计和优化的，不会“相互交流”或从数据中学习大量参数。相比之下，深度学习方法中的层是通过学习数据自动优化参数的，各层之间会有信息的交流。

语义分割

每个像素都有它的标签，包括背景，和未知

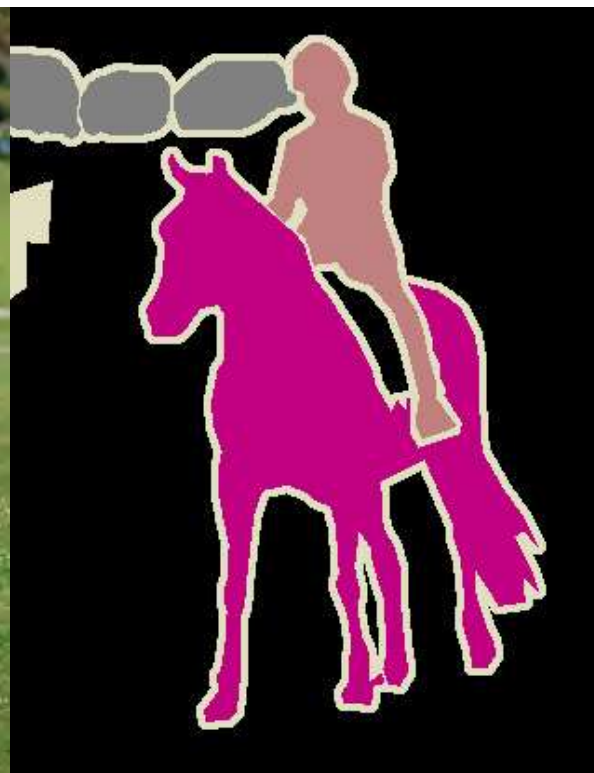
不同标签用不同颜色表示

不会区分相同类别下不同的实体 (semantic vs. instance)

Input



Label



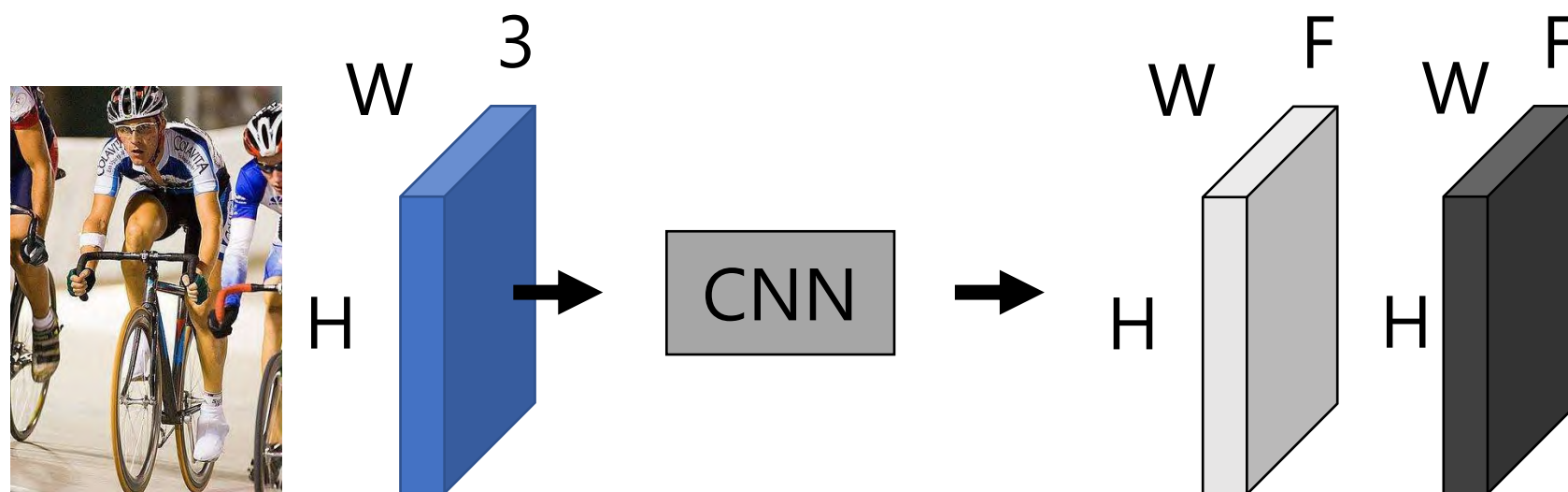
Input



Label



语义分割



在每个像素上做
F-分类交叉熵损失：
$$-\log \left(\frac{\exp((Wx)_{y_i})}{\sum_k \exp((Wx)_k)} \right)$$

为什么这个任务很难?

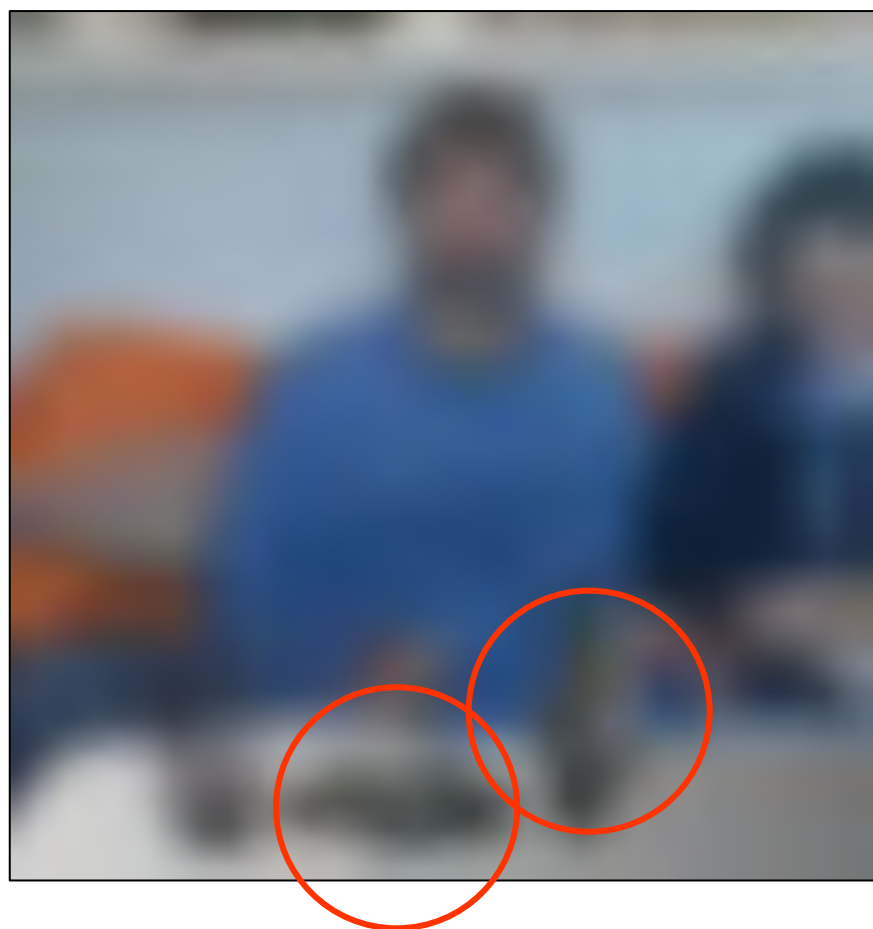
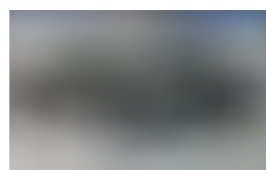
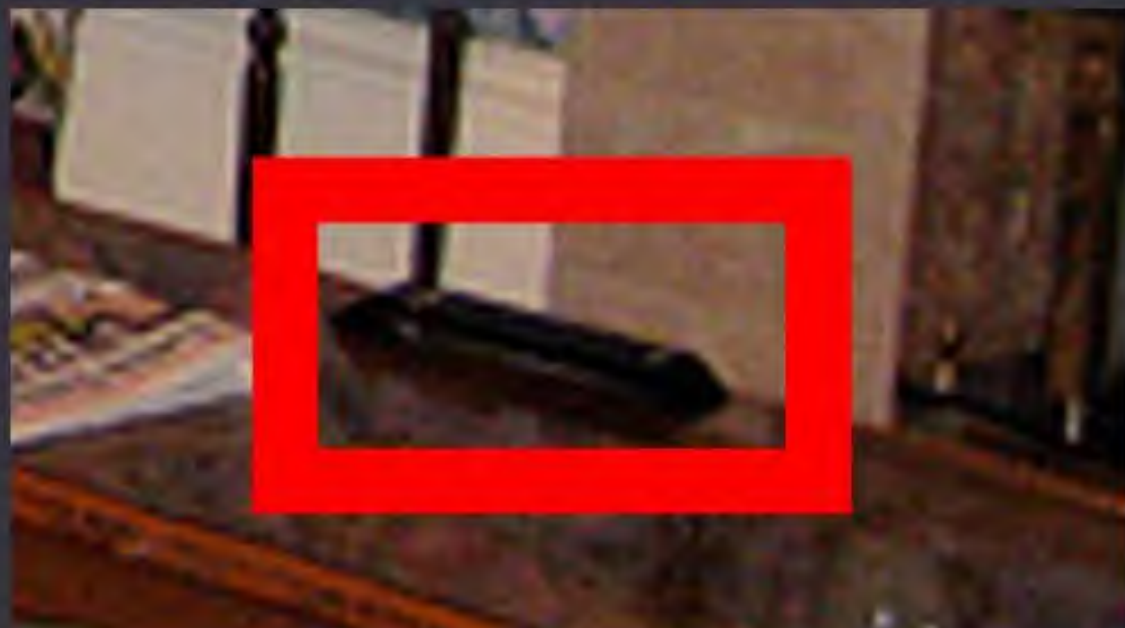


Image credit: A. Torralba

来自伟大的人工智能的回答

J You



我把局部放大了，现在请你再来看看这是什么？

ChatGPT

放大之后的图片显示，红框内的物体是一只远程控制器。可以辨认出它的典型长条形状和可能是按钮的细节，这些特征与常见的电视或音响系统的遥控器相符合。

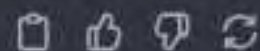
J You



图像红框里是什么物体？

ChatGPT

红框里的物体是一只猫。它似乎是在沙发边的地板上蜷缩着，与环境的其他色彩相比，它的颜色较为淡，可能导致它不那么显眼。猫的姿态表明它可能在休息或打盹。

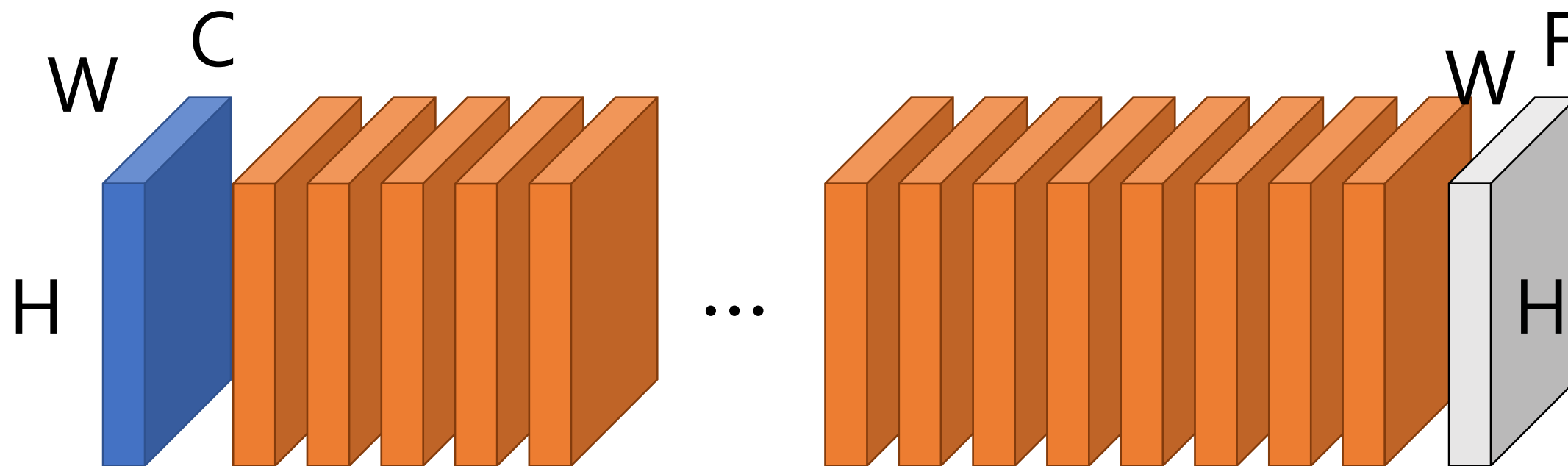


为什么这个任务很难?



Image credit: COCO dataset

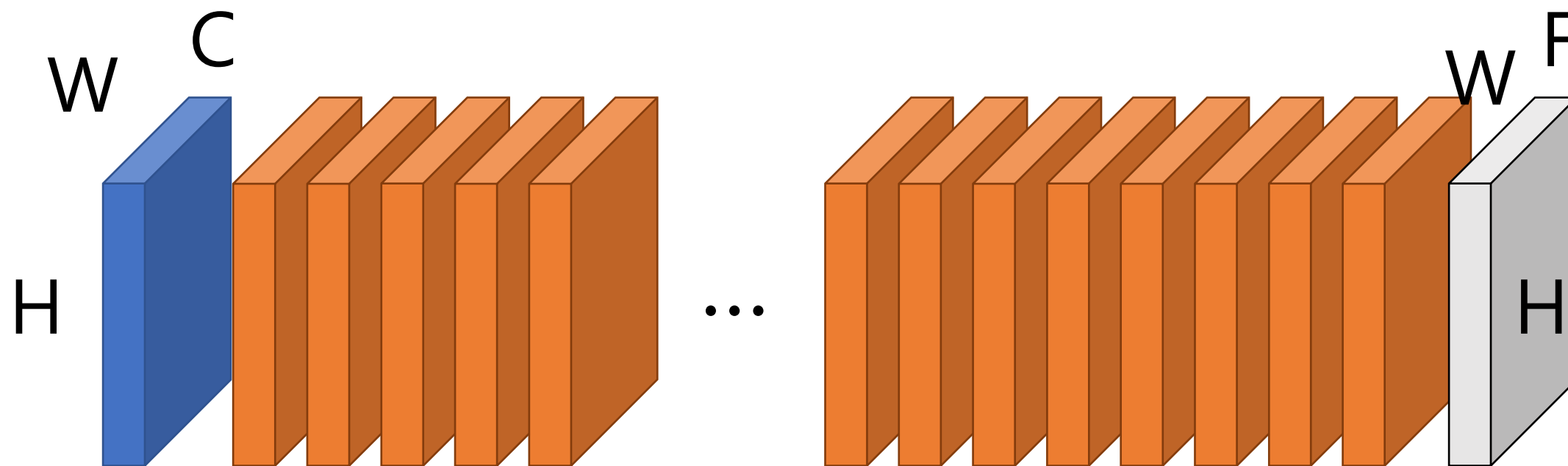
怎么做？堆叠卷积？



N 个 3×3 卷积的感受野是 $2n+1$ 大小的方形区域
如果能让感受野 ≥ 200 需要多少卷积？

100

堆叠卷积? 似乎不行



如果有 200 个 3x3 卷积通道, $H=W=400$

每层的存储开销: $200 * 400 * 400 * 4 \text{ bytes} = 122\text{MB}$

内存溢出!*

*100 层, batchsize 20 = 238GB 内存!

想法2：滑动窗口

把每个窗口的局部拿出来过CNN，
预测窗口中心的像素标签

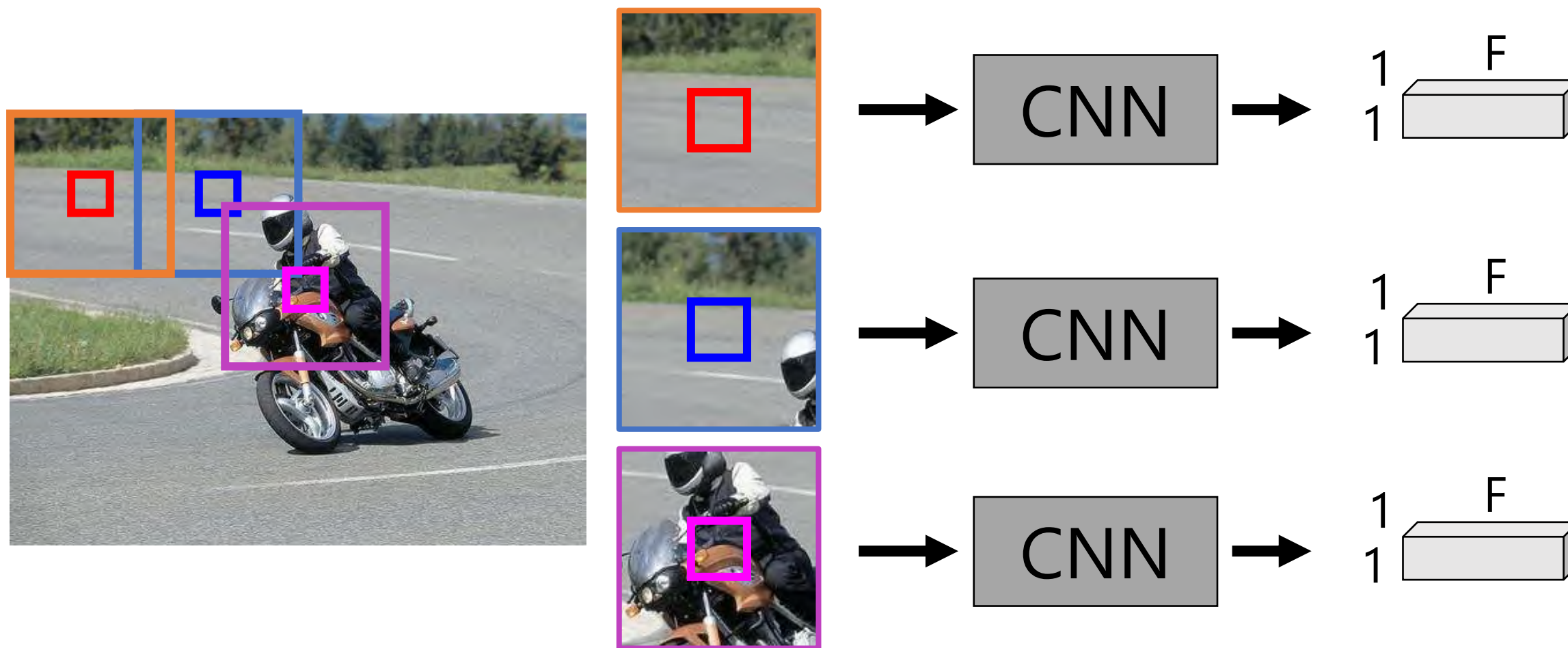
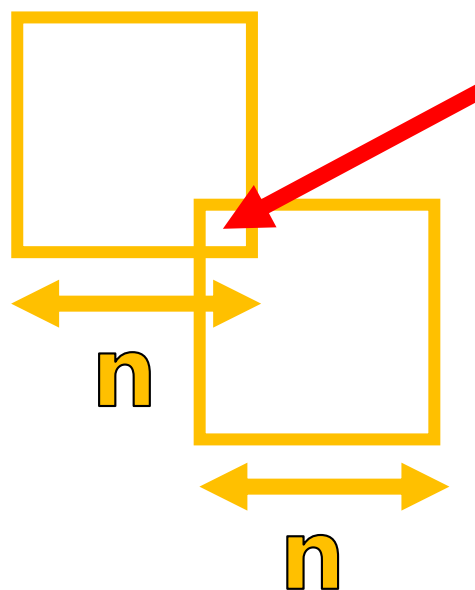
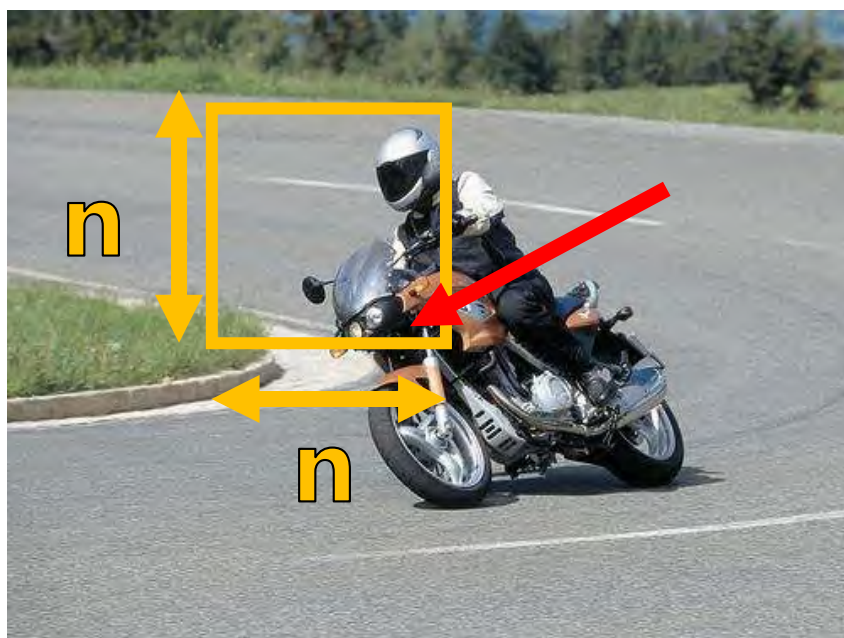


Image credit: PASCAL VOC, Everingham et al.

这样做有什么问题? ...

如果我们单独对每 $N \times N$ patch用CNN提取特征.
红色的像素点会被**CNN**计算几次?



Answer:
 $N * N$

核心挑战

我们需要:

1. 很大的感受野来判断每个像素点周围的情况
2. 不能在满足1的情况下有太大的内存开销

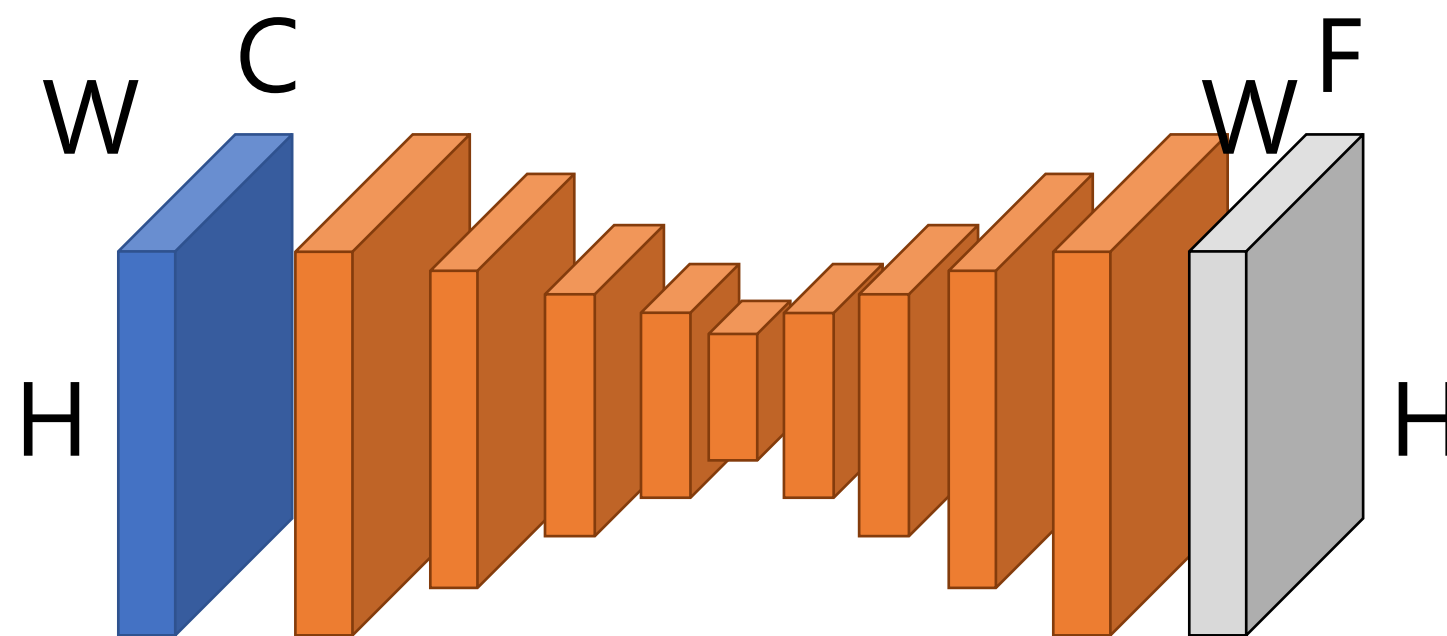
如何解决这对矛盾的问题?

编码器-解码器

核心思想: 先 **downsample** 下采样
再 **upsample** 上采样.

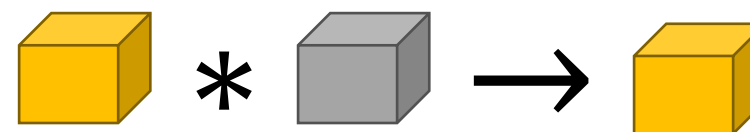
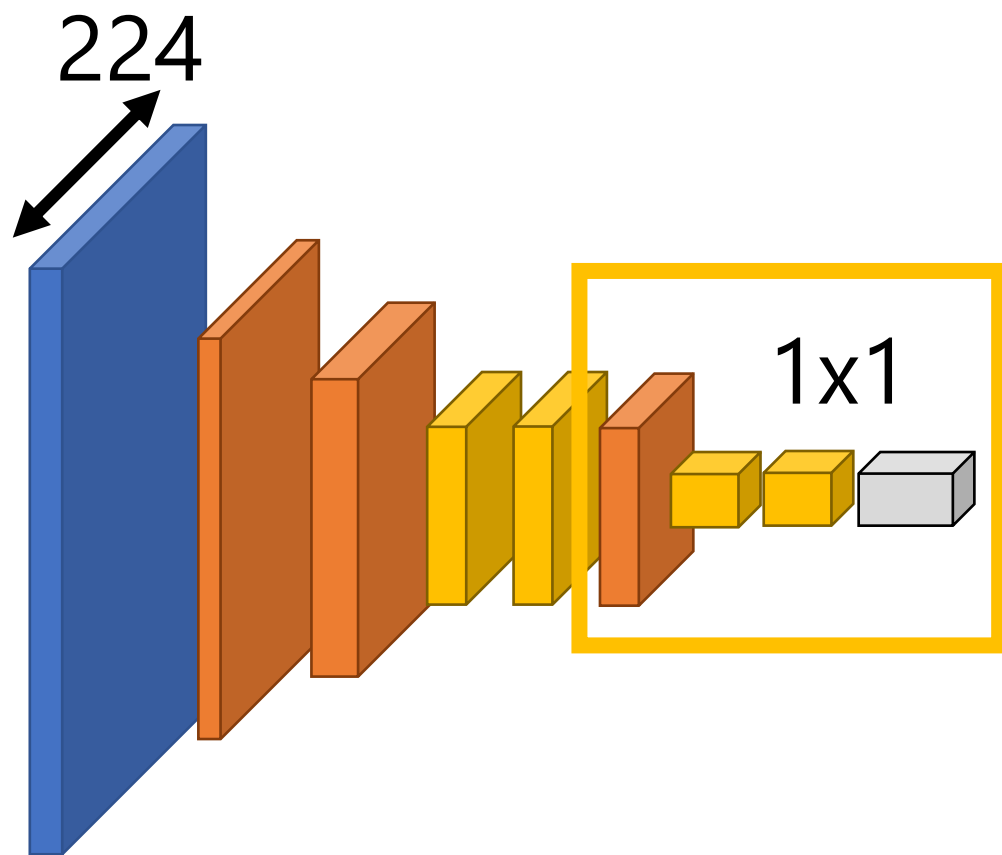
怎么做下采样?

卷积, 池化



如何设计这样的网络?

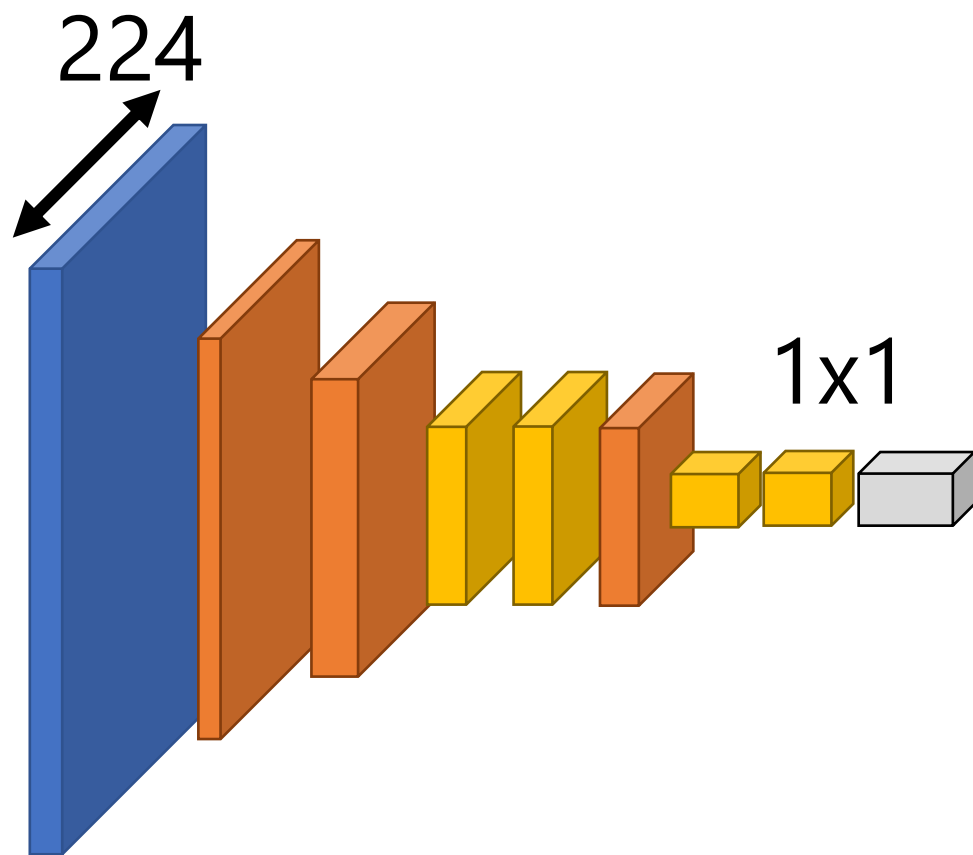
之前的网络都是1维的
最终输出



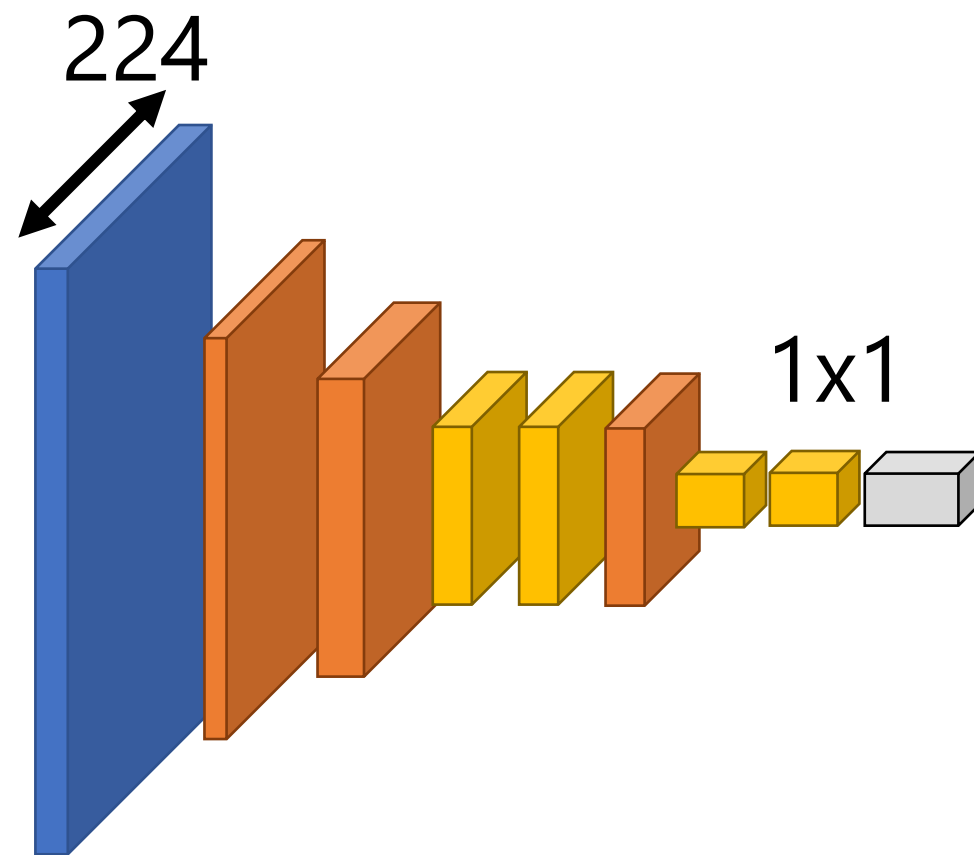
最后一层全连接层我们
可以用1*1的卷积来替代

如何设计这样的网络?

把图像映射到向量的
CNN



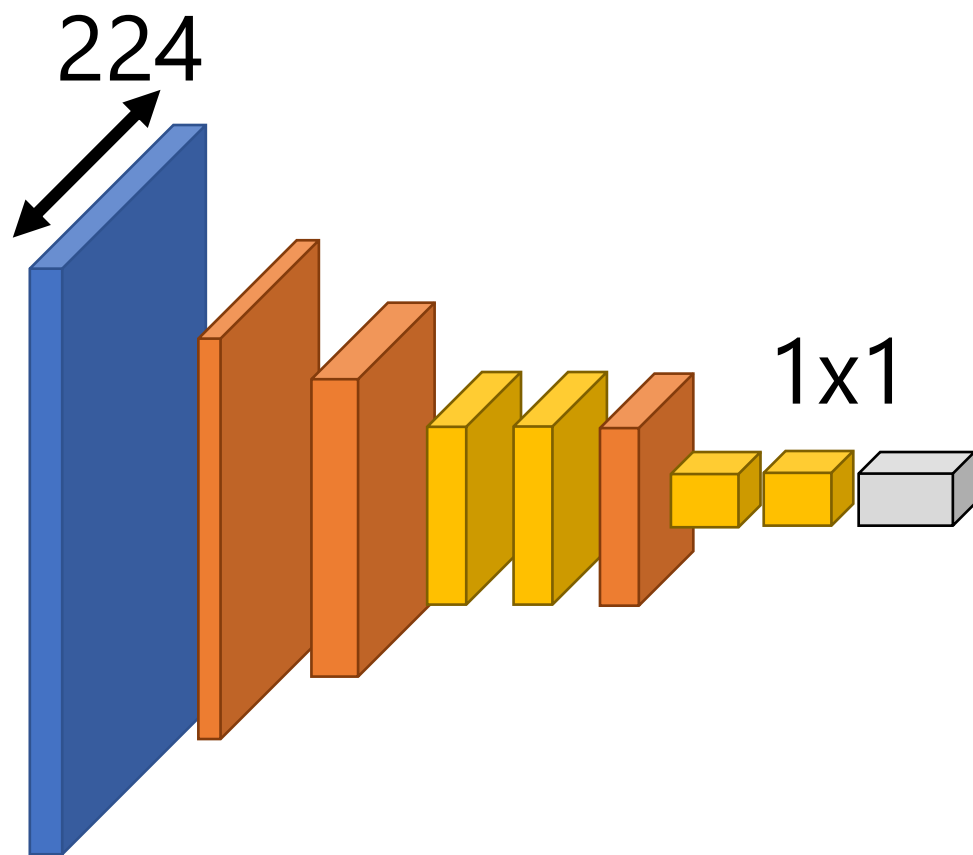
把图像映射到图像的
CNN



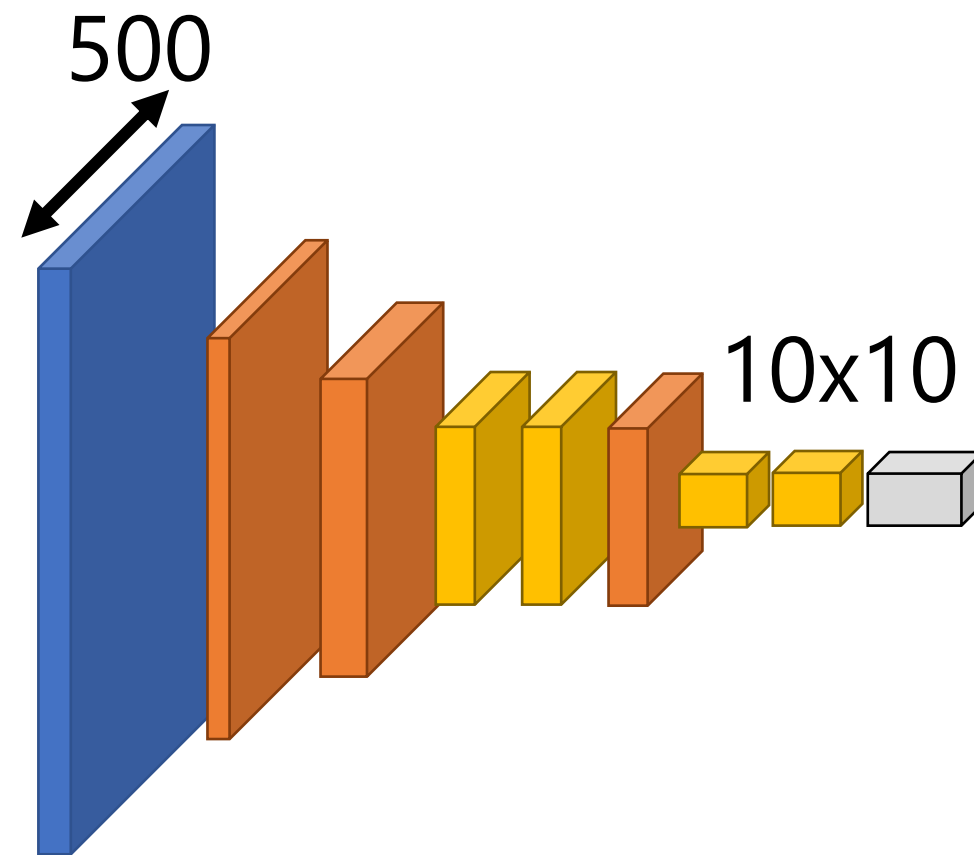
输入尺寸变大会怎样?

如何设计这样的网络?

把图像映射到向量的
CNN

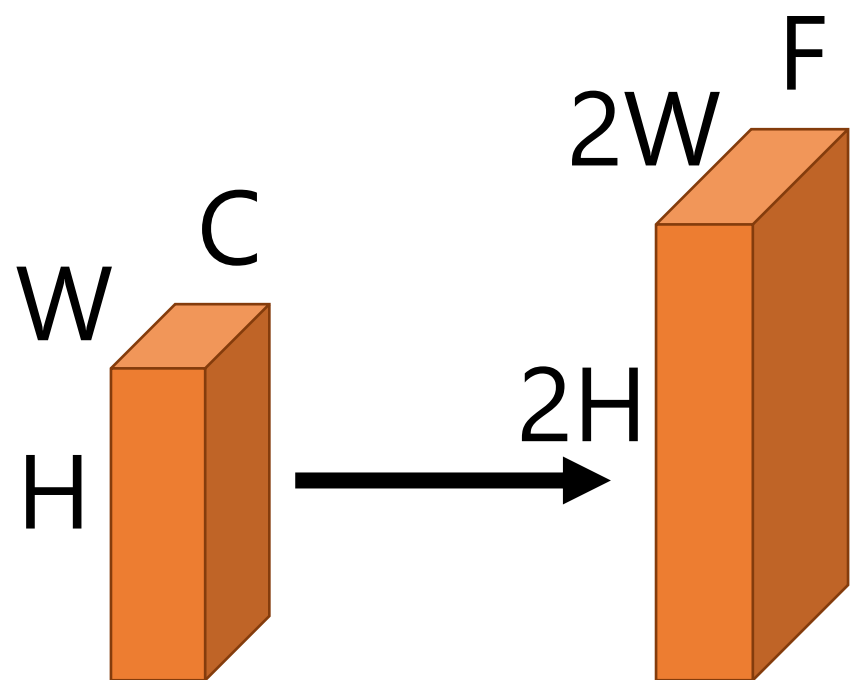


把图像映射到图像的
CNN



输出有空间尺度了! 我们可以利用它!

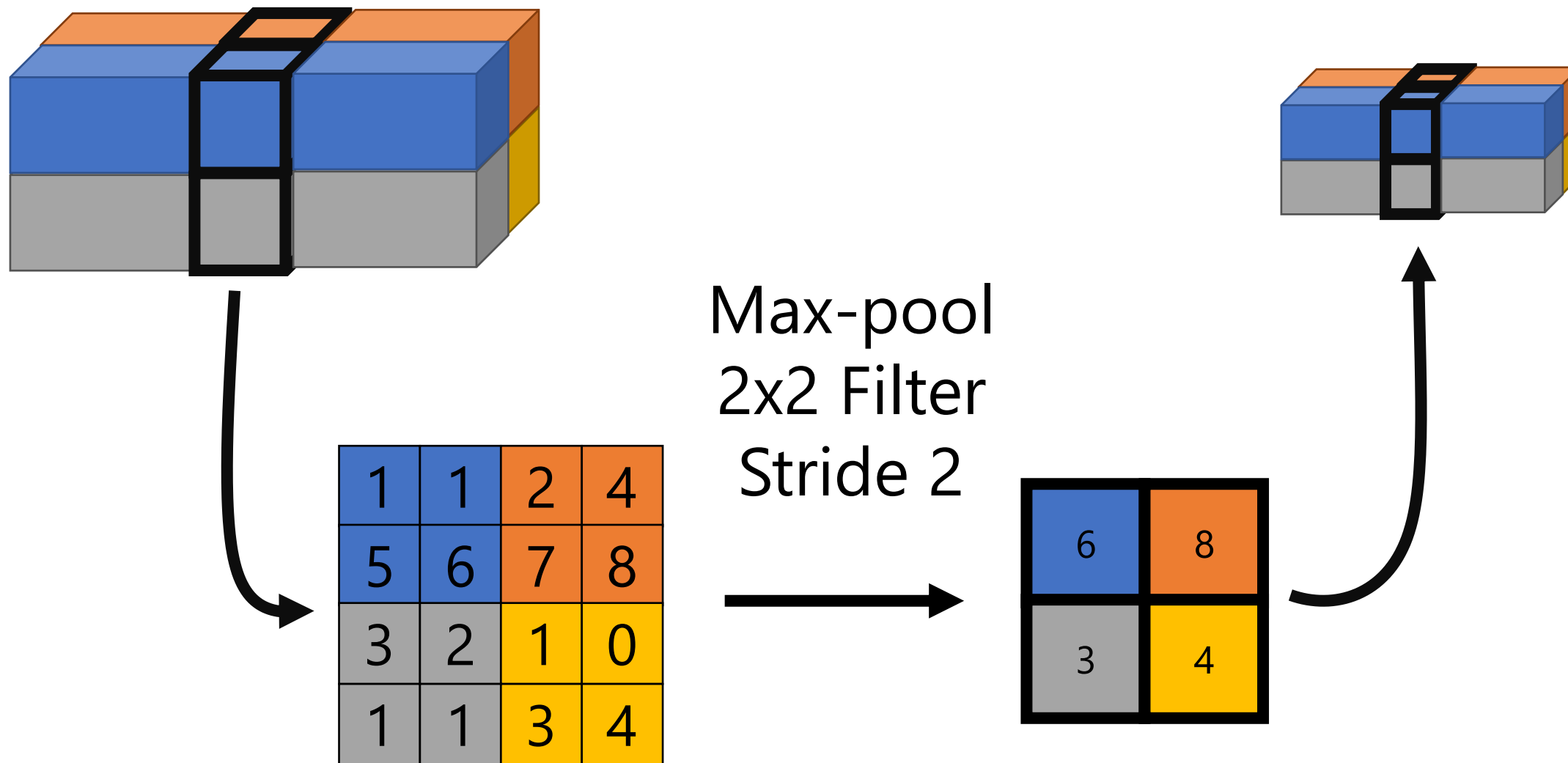
怎么做上采样?



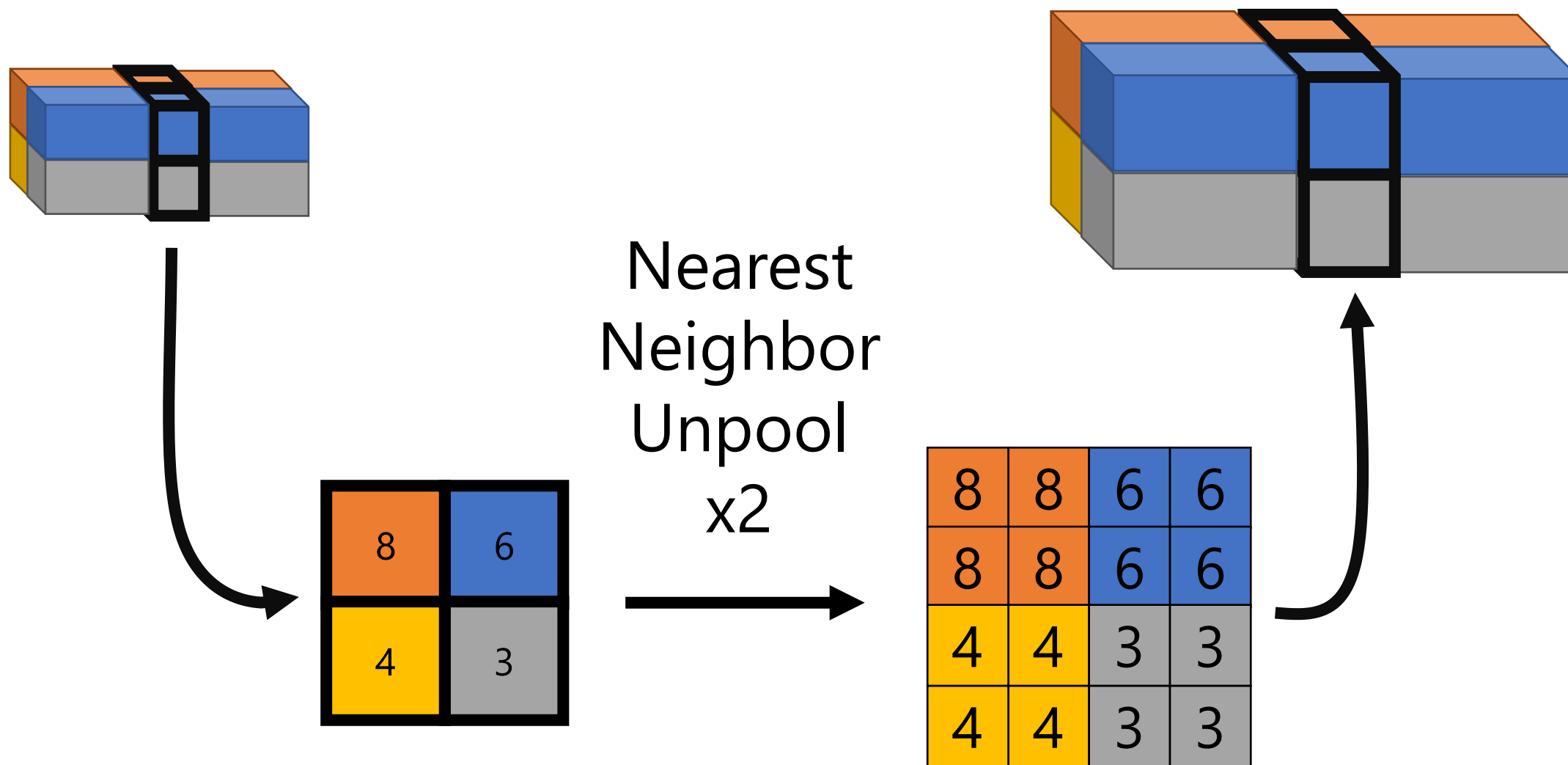
做下采样的反向操作:

1. Pooling → "Unpooling"
2. Convolution → "Transpose Convolution"

回顾：池化

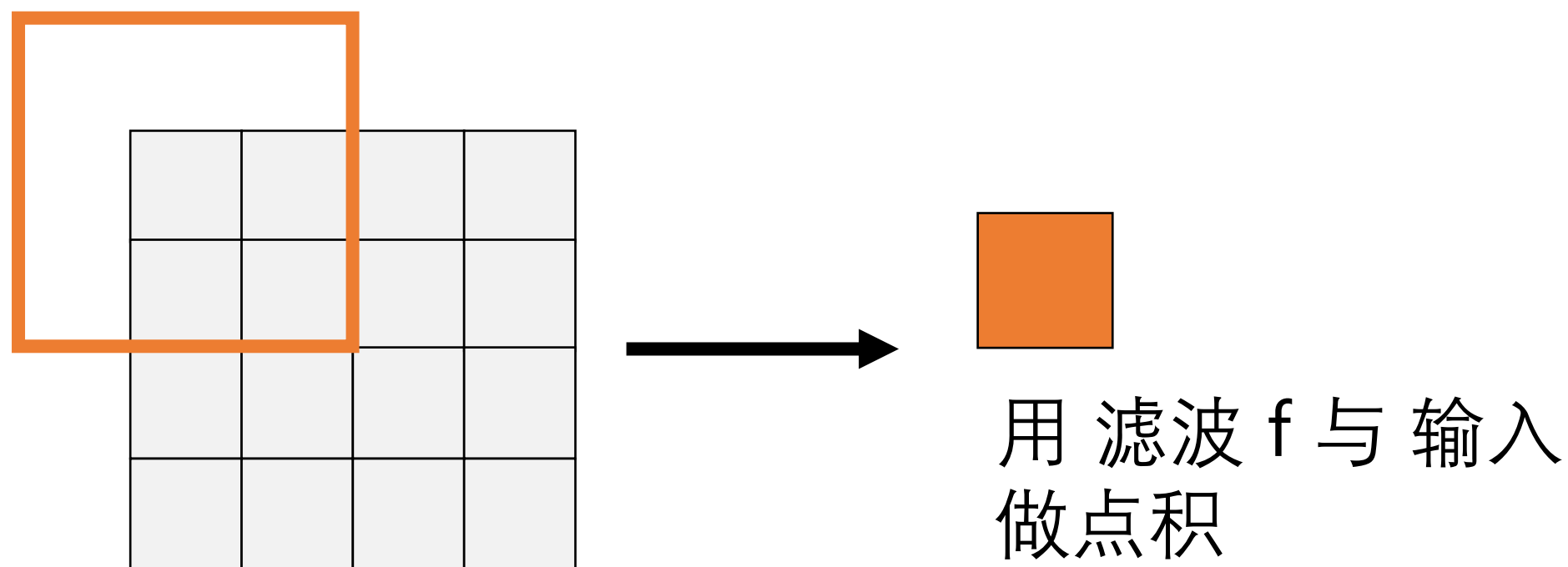


Unpooling——反池化



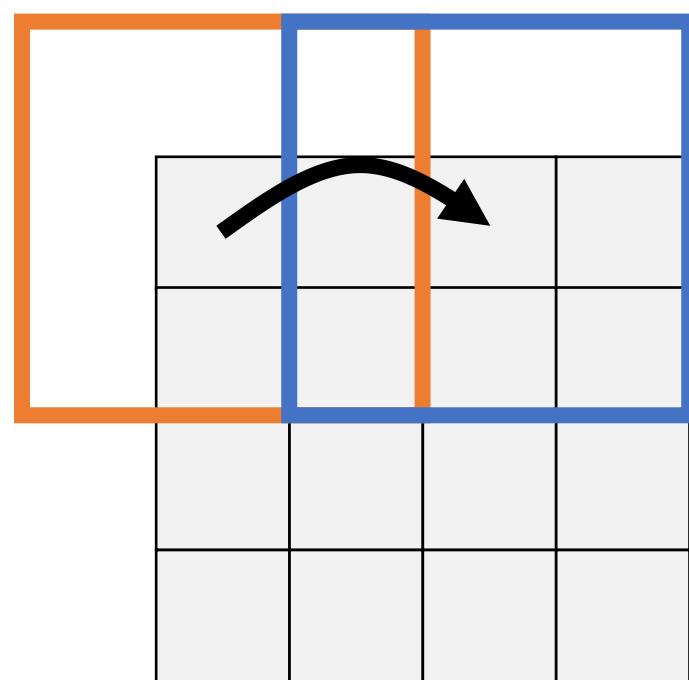
回顾：卷积

3x3卷积, Stride 2, Pad 1



回顾：卷积

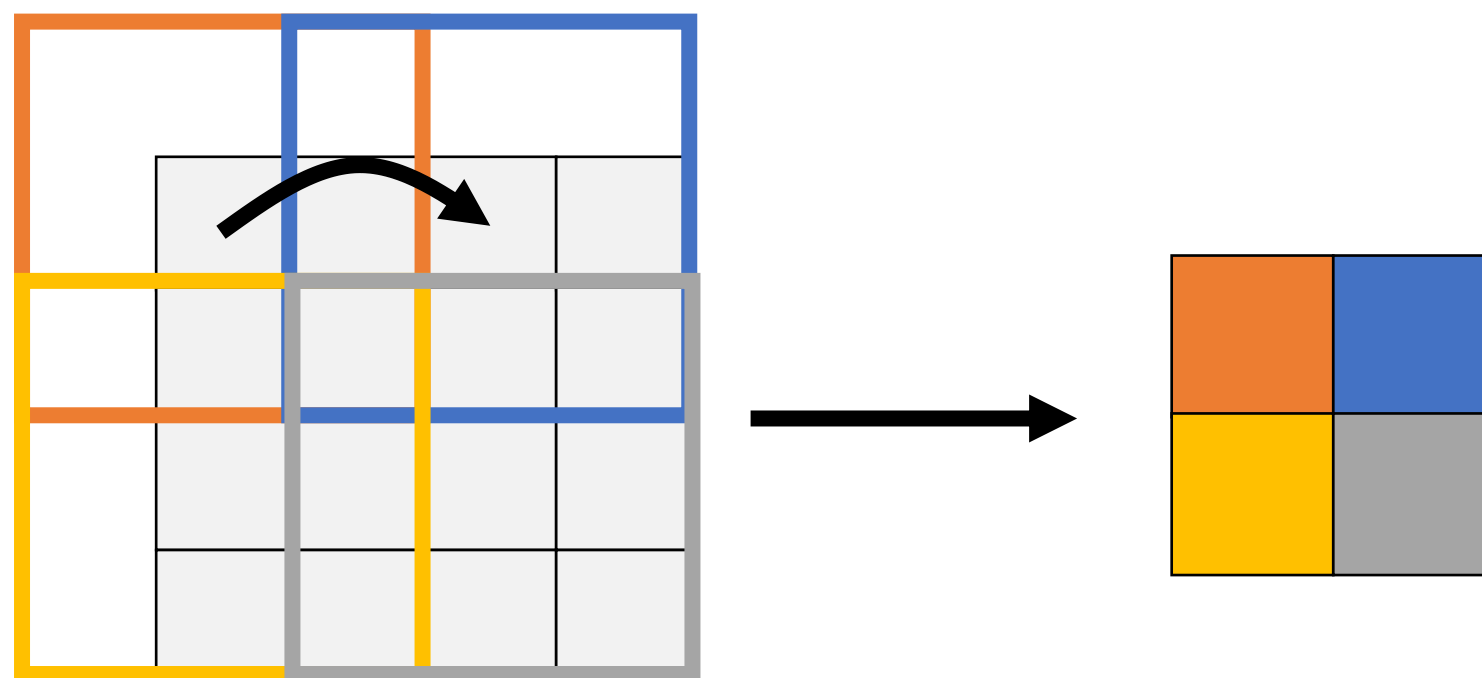
3x3卷积, Stride 2, Pad 1



用滤波 f 与输入
做点积

回顾：卷积

3x3卷积, Stride 2, Pad 1



反卷积

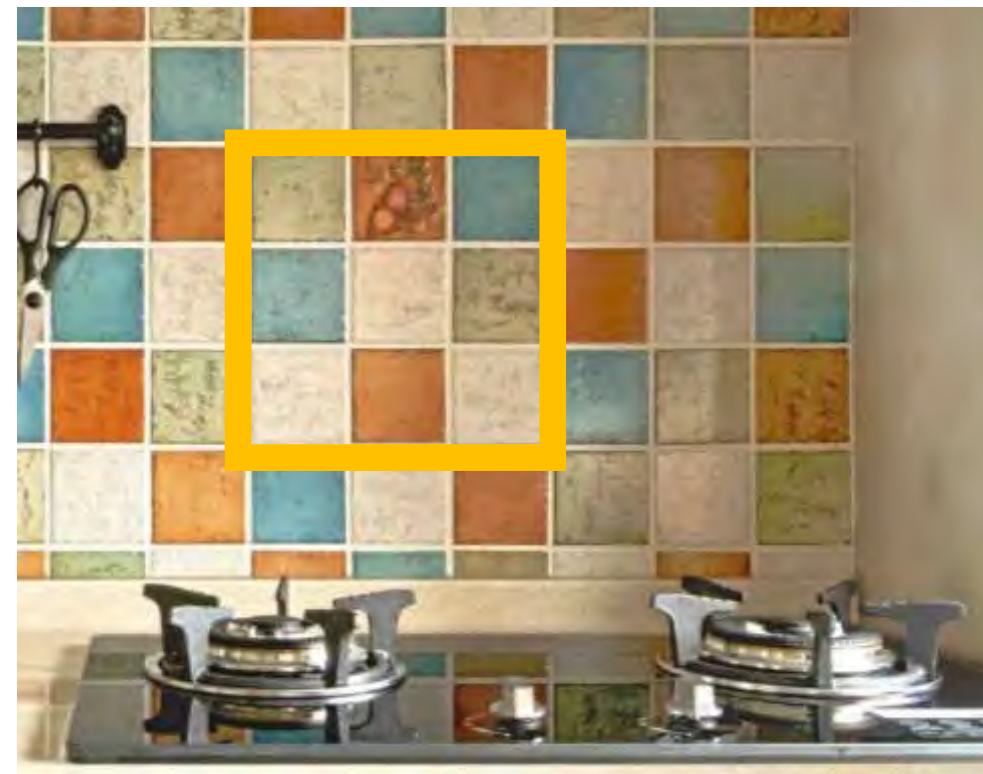
卷积

滤波的作用：关注局部区域的内容



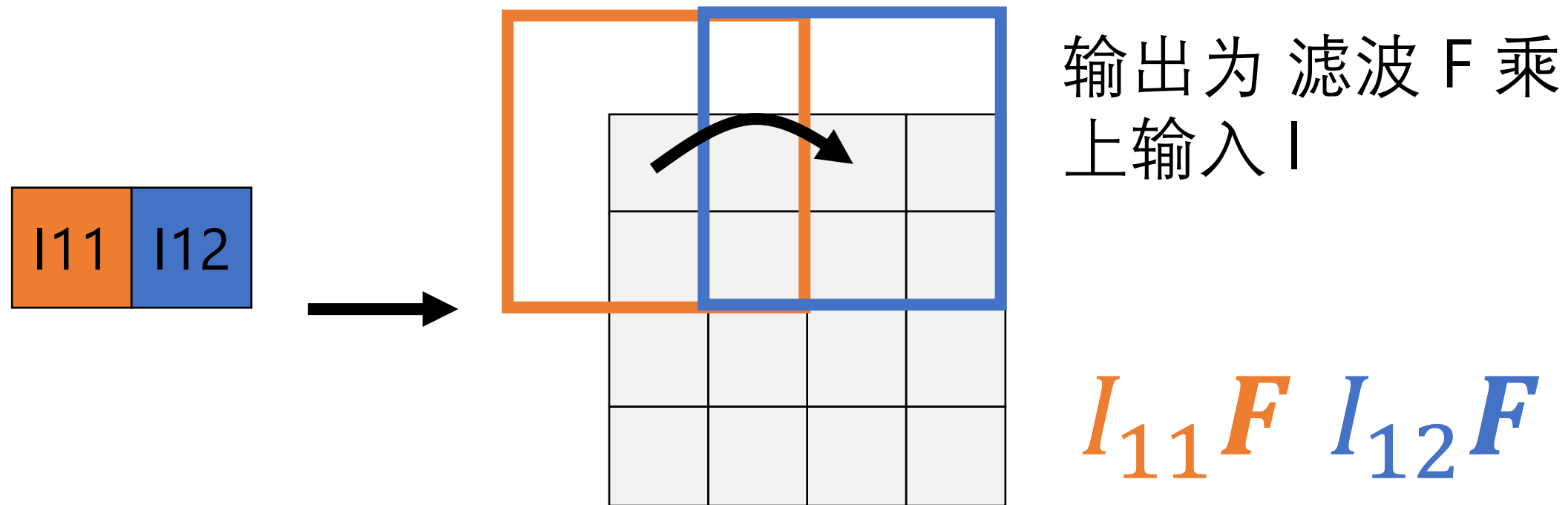
反卷积

滤波的作用：像贴瓷砖一样把滤波“铺上去”



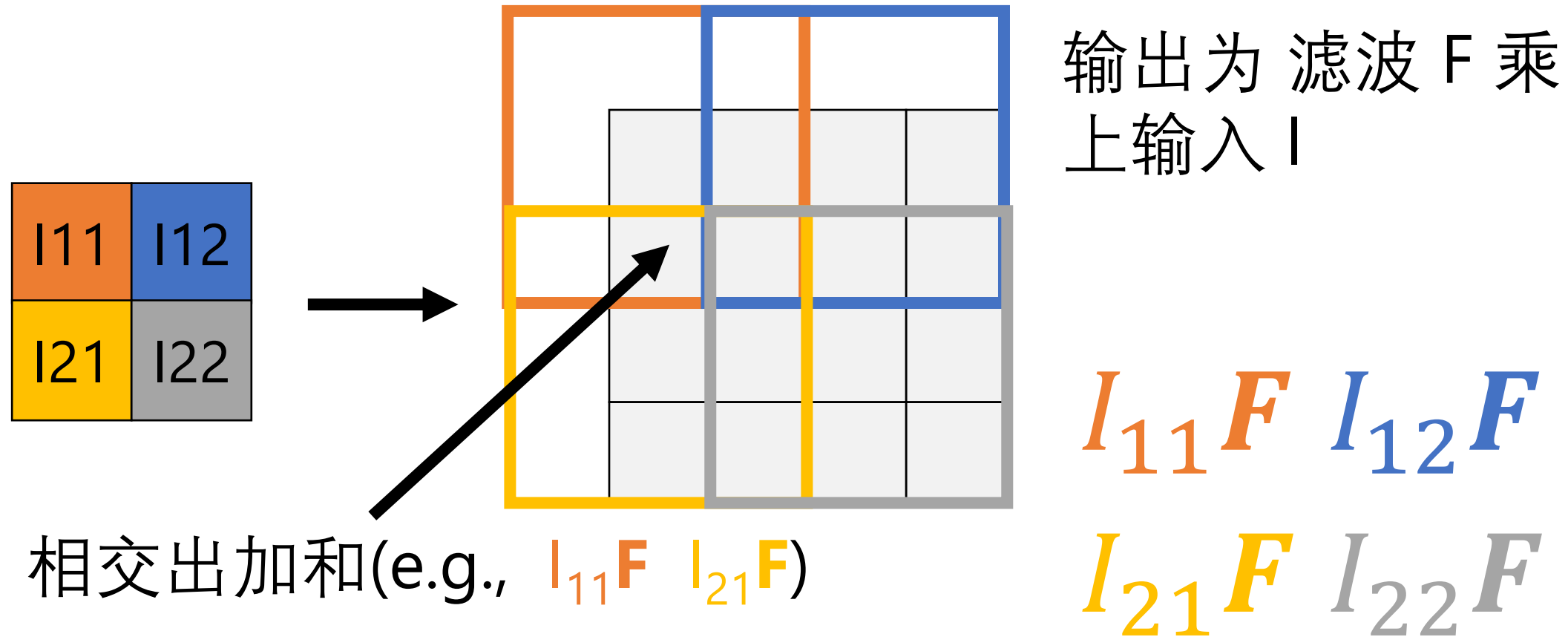
Transpose Convolution、DeConvolution——反卷积

3x3 Transpose Convolution, Stride 2, Pad 1



Transpose Convolution 、 DeConvolution——反卷积

3x3 Transpose Convolution, Stride 2, Pad 1



为什么叫“Transpose Convolution”?

我们可以用矩阵乘法的形式实现卷积

Input: 4, Filter: 3, Stride: 1, Pad: 1

$$\begin{bmatrix} a & b & c & d \end{bmatrix} * \begin{bmatrix} x & y & z \end{bmatrix}$$

$$\begin{bmatrix} x & y & z & 0 & 0 & 0 \\ 0 & x & y & z & 0 & 0 \\ 0 & 0 & x & y & z & 0 \\ 0 & 0 & 0 & x & y & z \end{bmatrix} \times \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ ax + by + cz \\ bx + cy + dz \\ cx + dy \end{bmatrix}$$

为什么叫“Transpose Convolution”?

反卷积与卷积矩阵乘法形式为转置操作

$$\begin{bmatrix} a & b & c & d \end{bmatrix} *^T \begin{bmatrix} x & y & z \end{bmatrix}$$

x	0	0	0
y	x	0	0
z	y	x	0
0	z	y	x
0	0	z	y
0	0	0	z

 \times

a
b
c
d

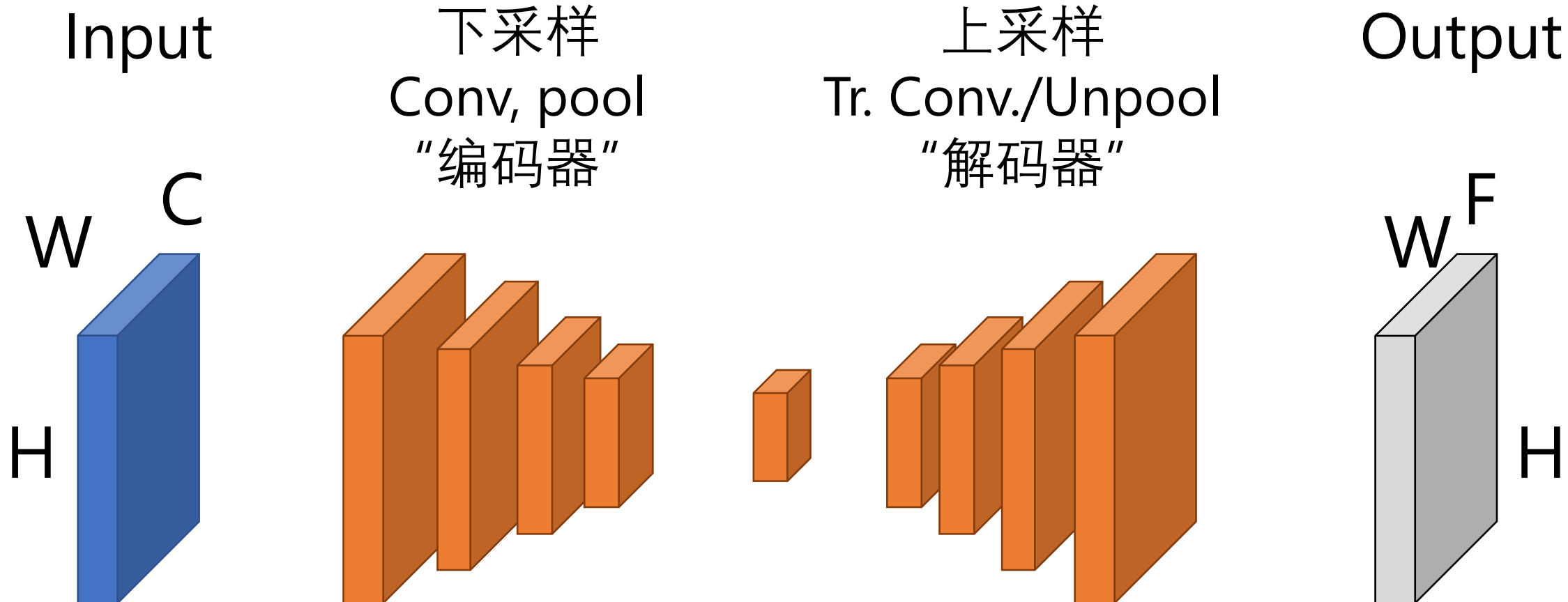
 $=$

ax
ay+bx
az+by+cx
bz+cy+dx
cz+dy
dz

 $\begin{matrix} \begin{matrix} ax \\ ay+ \\ az+ \end{matrix} & \begin{matrix} bx \\ by+ \\ bz+ \end{matrix} & \dots \\ \dots & \dots & \dots \end{matrix}$

汇总起来

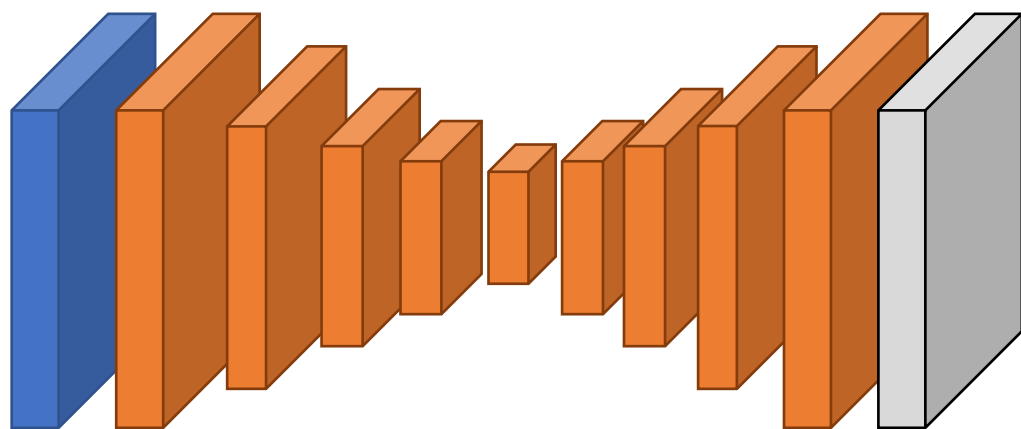
卷积 + 池化 来做 下采样/压缩/编码
反卷积/反池化 来做 上采样/解压缩/解码



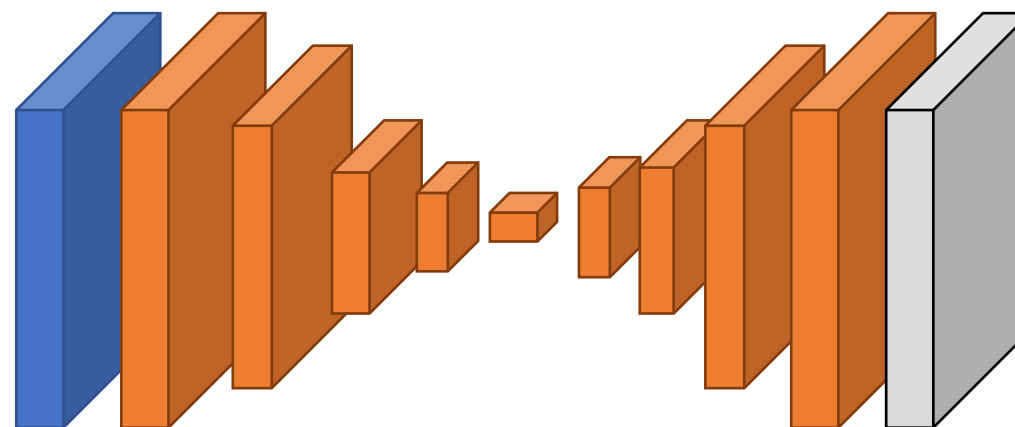
那么怎么设计这样的网络呢？

- 最好一直保留一些空间尺度

编码到较小尺度, 然后解码.

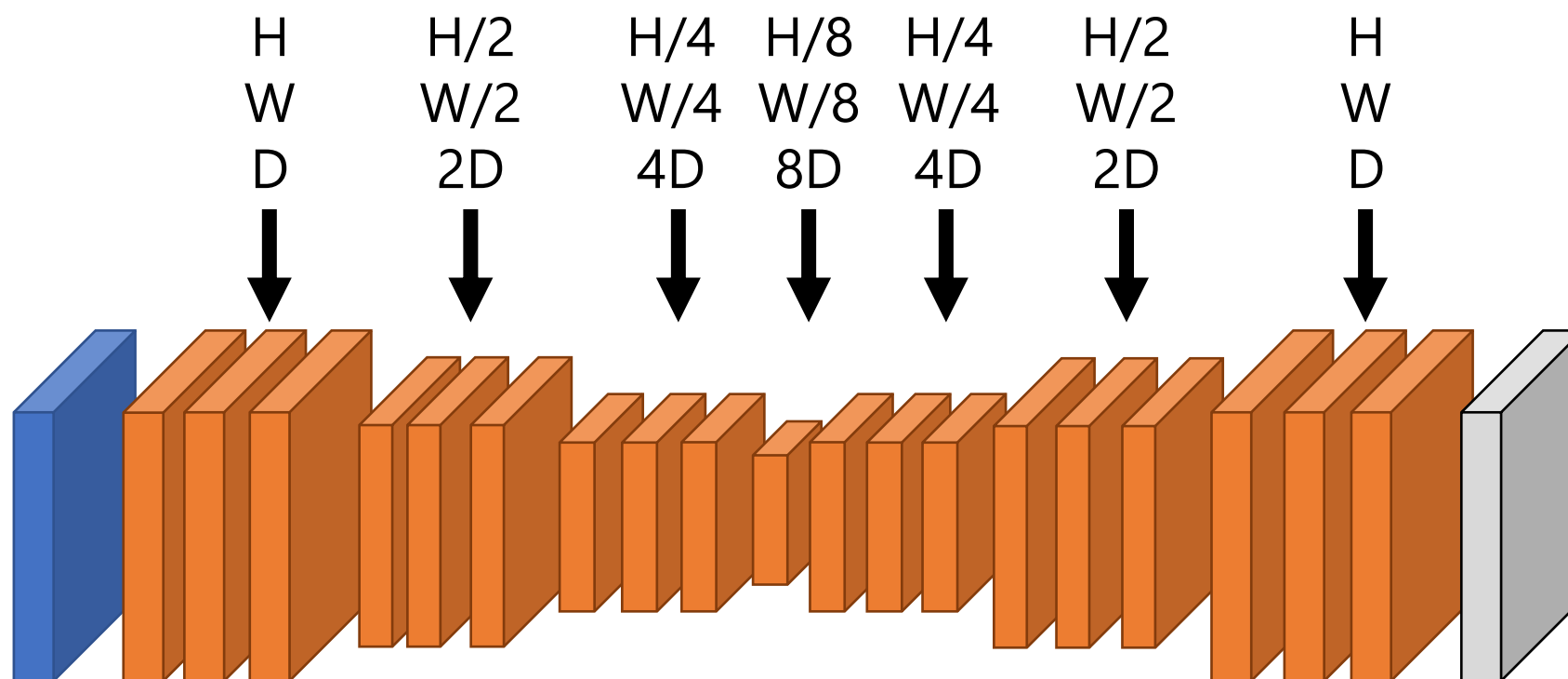


编码到1维向量, 然后解码



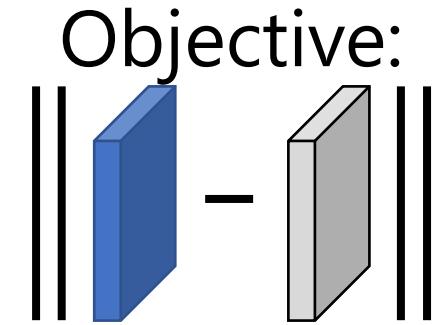
设计经验

- 在每个分辨率上堆叠几个普通卷积层.
- 经常设计成隔几层输出空间尺寸减半，通道数量翻倍



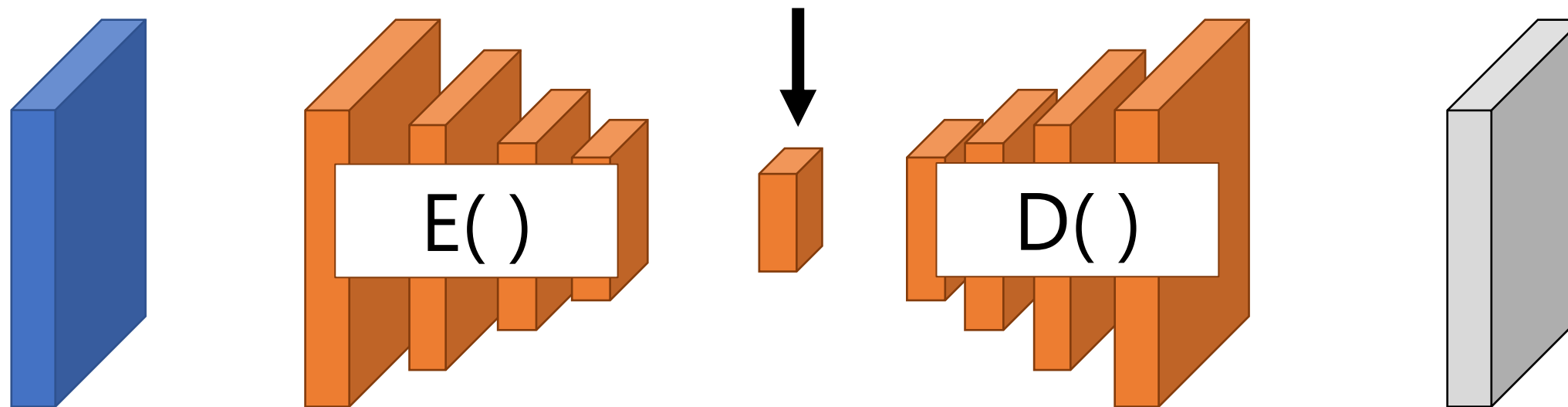
Autoencoders 自编码器

把输入编码压缩到“bottleneck”，然后
解码到输入原尺寸



$$\|D(E(X)) - X\|$$

Bottleneck 瓶颈/
Latent Space 隐变量空间/
Latent Code 隐变量编码



Latent Space* 隐变量空间

Interpolation in Latent Space



*In the interest of honesty in advertising: not an autoencoder, but a similar method with the same goal of learning a latent space

Result from Wu et al. *Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling*.
NIPS 2016

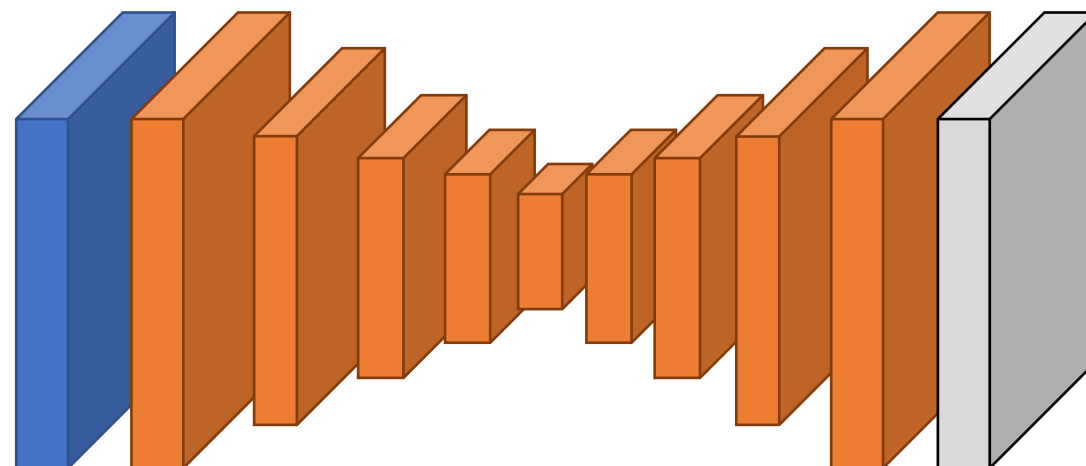
缺失的细节内容

如果输出尺寸时 $H \times W$, 直接上采样一个经过下采样的图像,
我们经常会得到一个缺失细节的结果.

为什么?



下采样时信息会丢失!



回顾：下采样



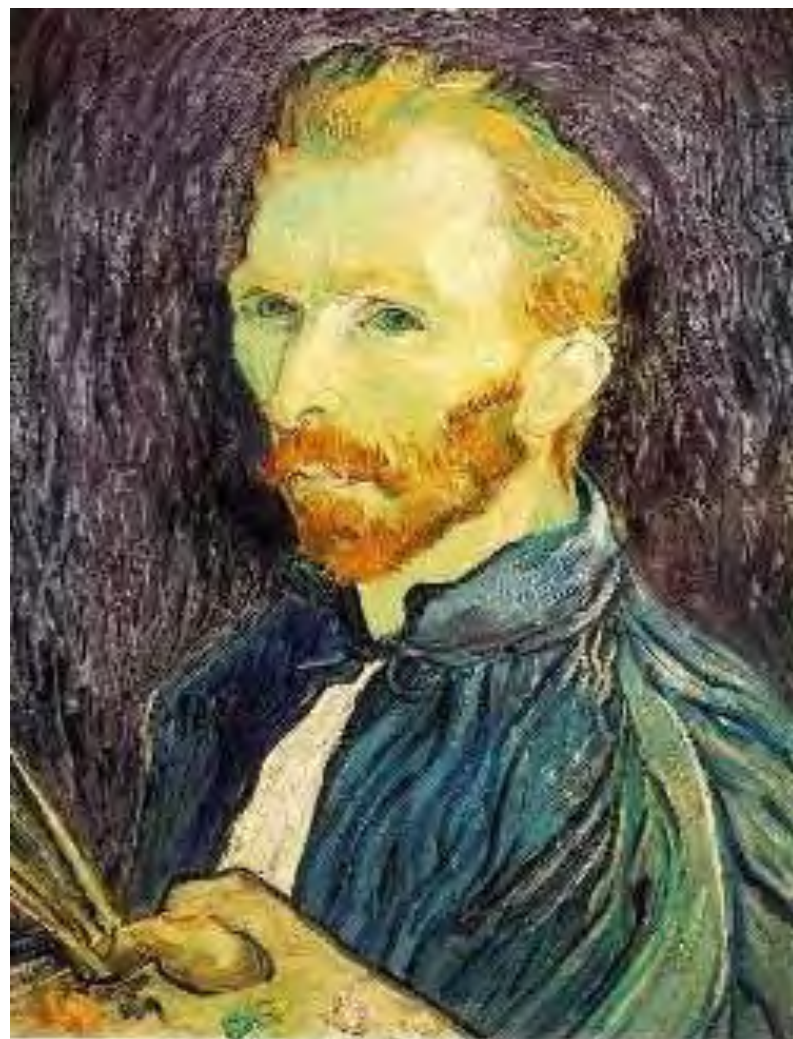
1/4



1/8

隔一行隔一列扔掉一半的图像
- 图像下采样 *image sub-sampling*

回顾：下采样



1/2



1/4 (2x zoom)



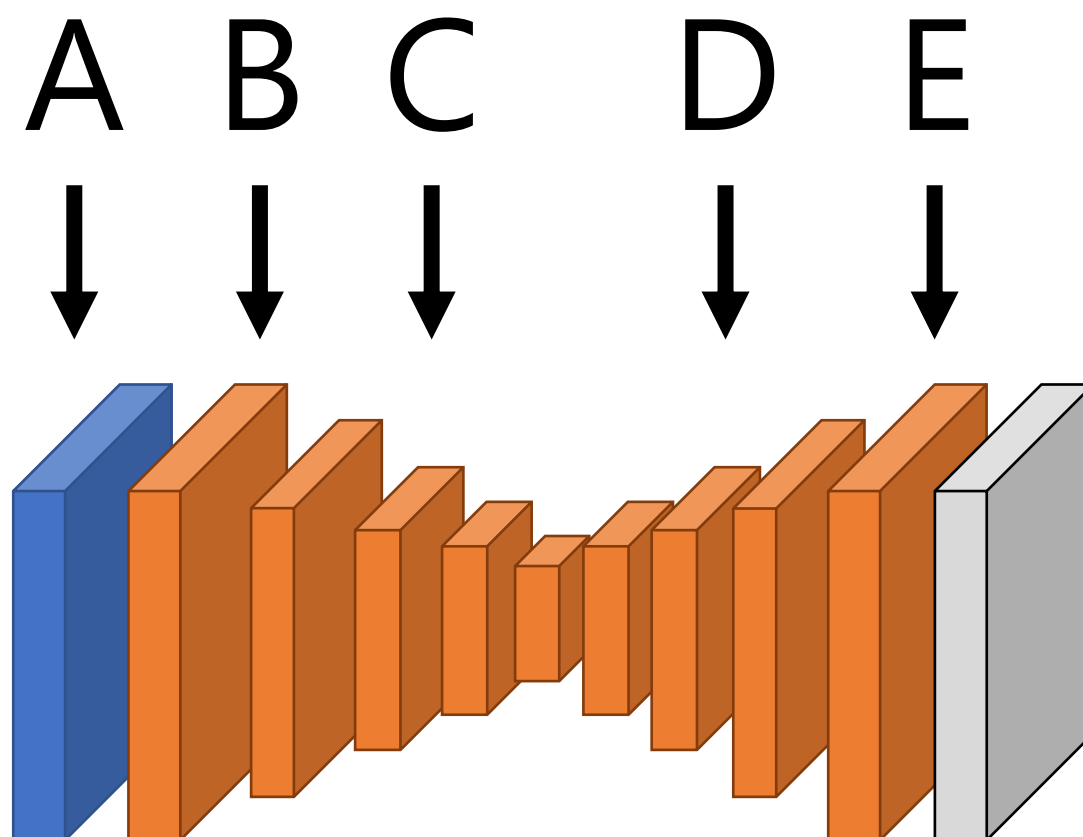
1/8 (4x zoom)

如果再上采样回来，为什么看起来这么粗糙？

Source: S. Seitz

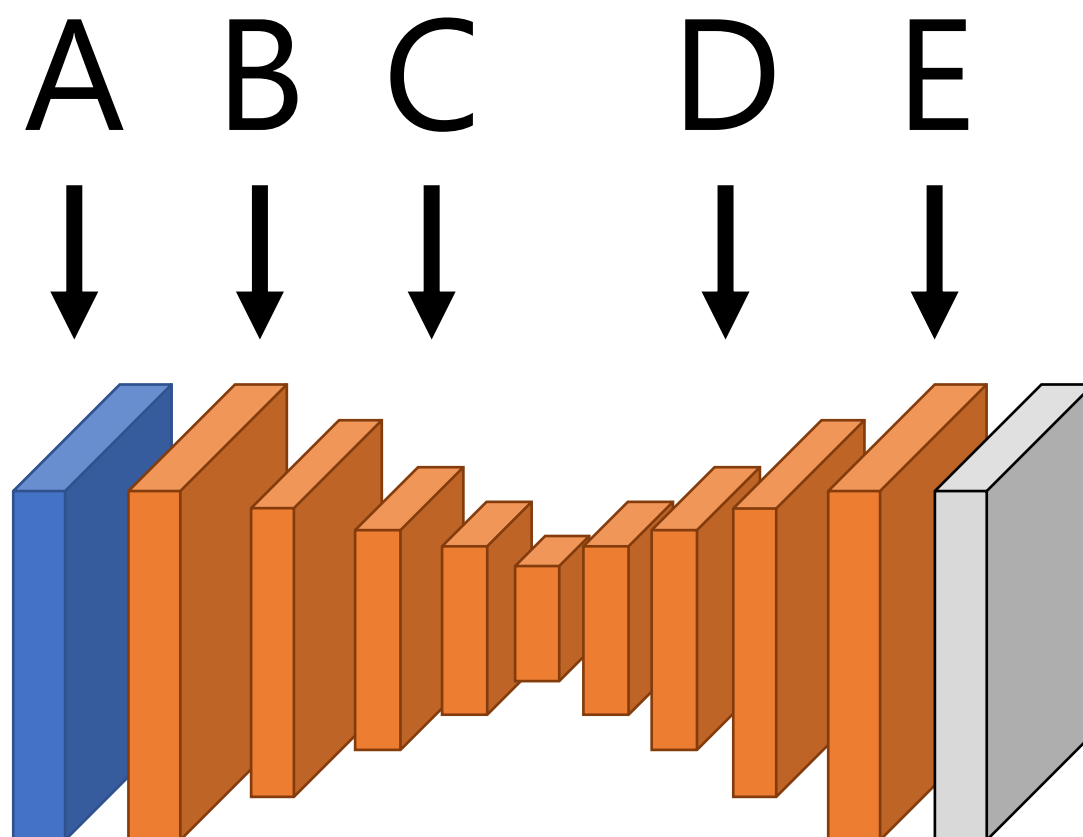
缺失的细节内容

哪里包含的图像高频信息最多？



缺失的细节内容

哪里缺失的图像高频信息最多？

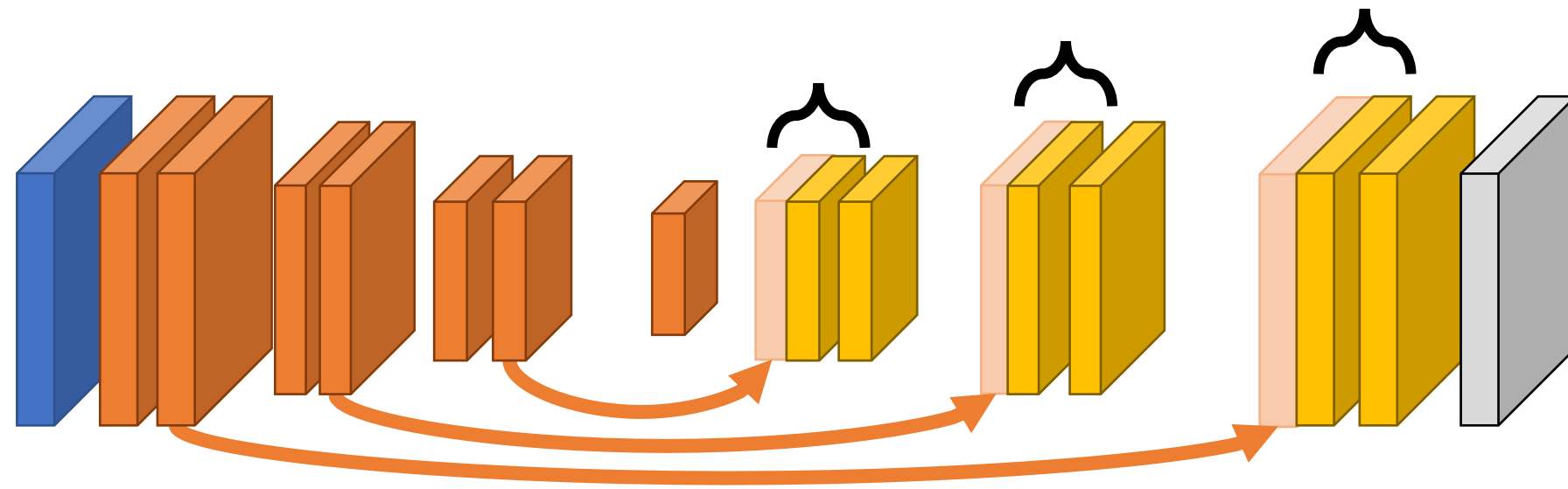


缺失的细节内容

怎么把缺失的细节补回来？

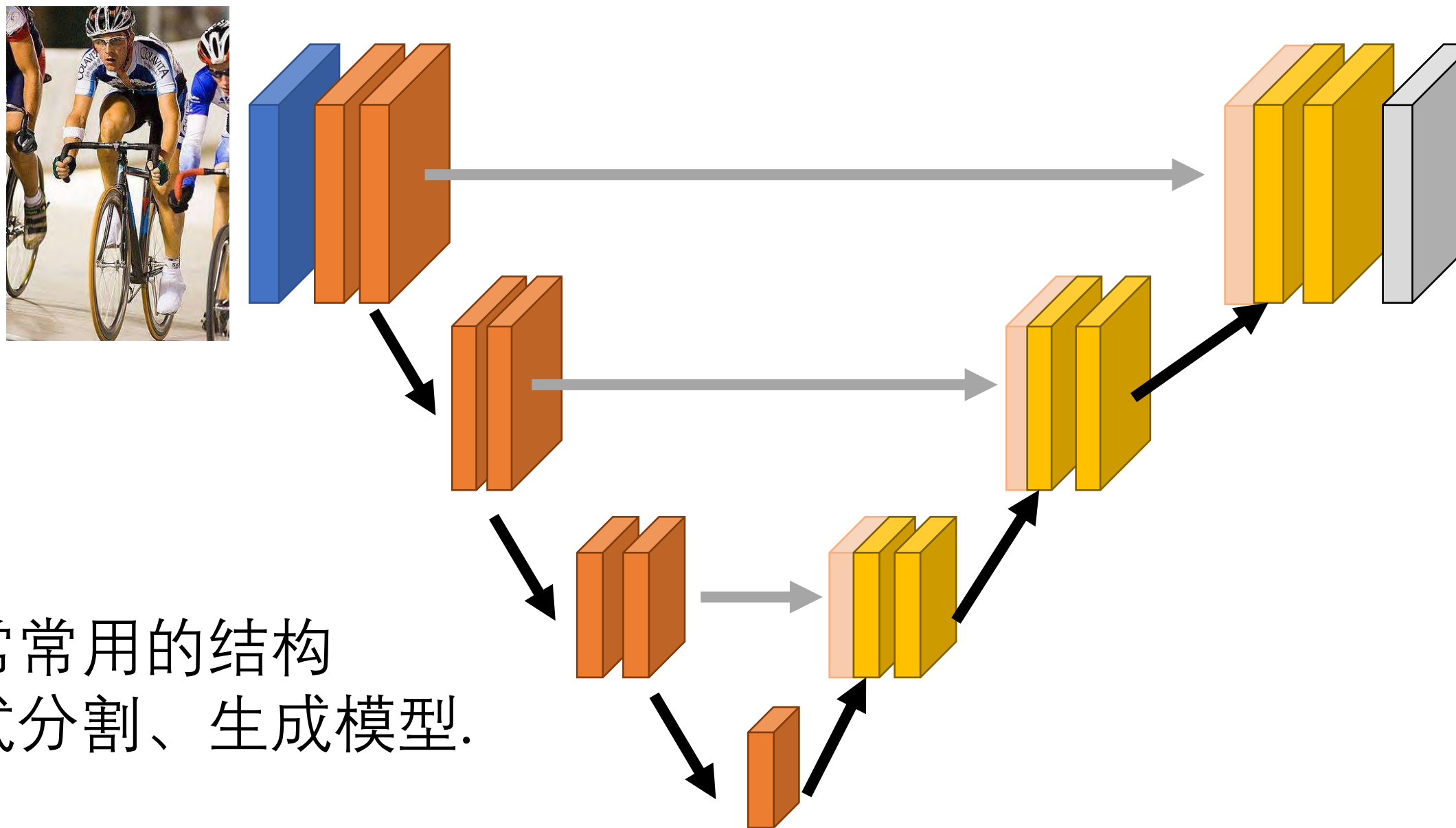
复制这些层的输出。

把他们拷贝到相同分辨率的、缺失细节的地方去。



Copy

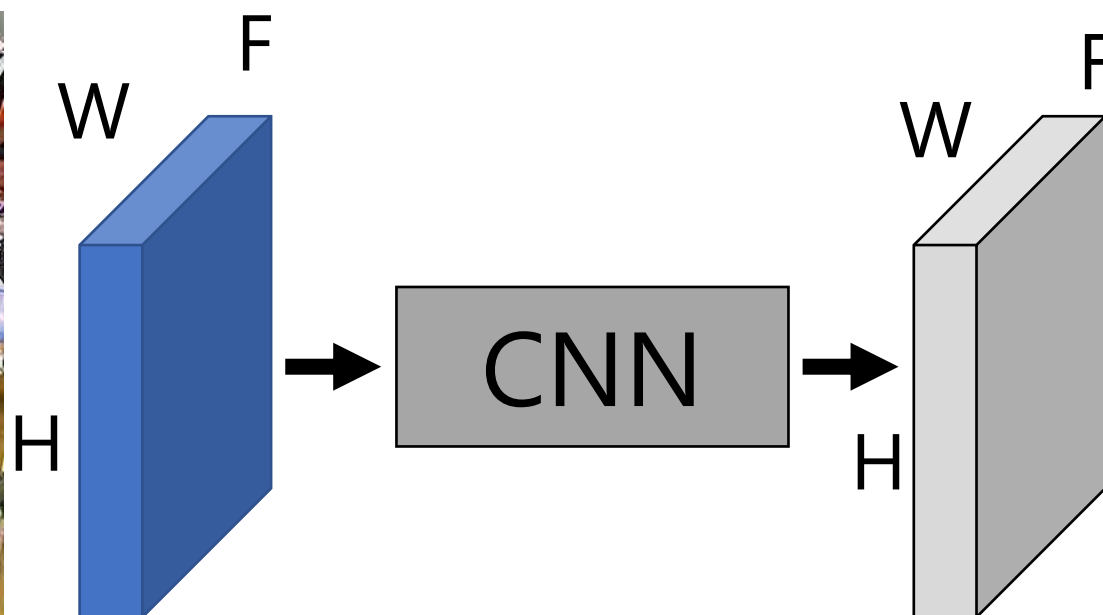
U-Net



非常非常常用的结构
用于各式分割、生成模型。

评估像素级别的预测

输入图像



Predicted
Classes



怎么得到 $H \times W \times F$ 的分类预测?

在评分输出上取argmax

评估语义分割

我们有了预测值，怎么做评估？

Input



Prediction (\hat{y})



Ground-Truth (y)

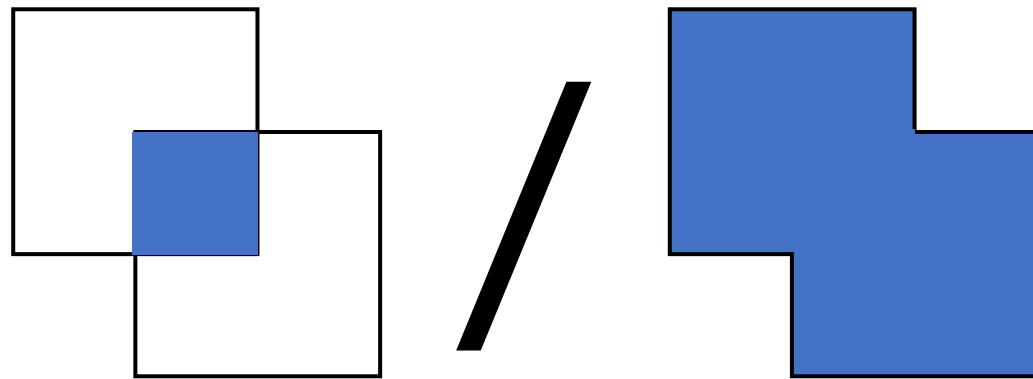


评估语义分割

每个像素都被分为F类中的某一个。

准确率: $\text{mean}(\hat{y} = y)$

交并比 IoU (Intersection over union), 预测类别和真实类别的重合区域/合并区域。所有类别取平均



Prediction
(\hat{y})

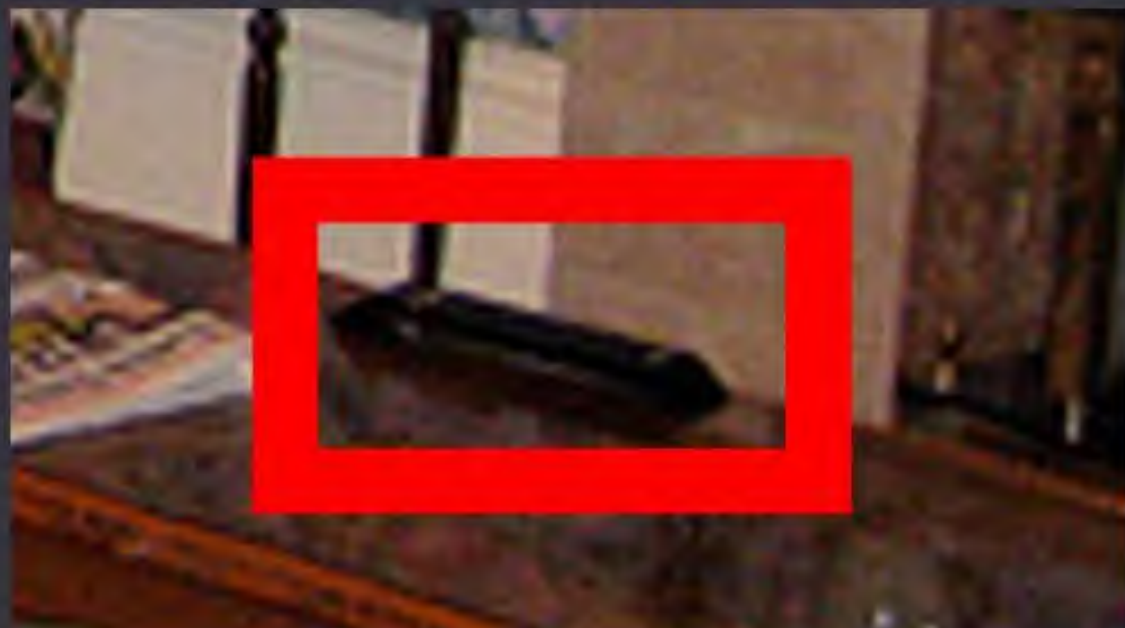


Ground-Truth
(y)



回顾： 来自伟大的人工智能的回答

J You



我把局部放大了，现在请你再来看看这是什么？

ChatGPT

放大之后的图片显示，红框内的物体是一只远程控制器。可以辨认出它的典型长条形状和可能是按钮的细节，这些特征与常见的电视或音响系统的遥控器相符合。

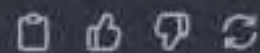
J You



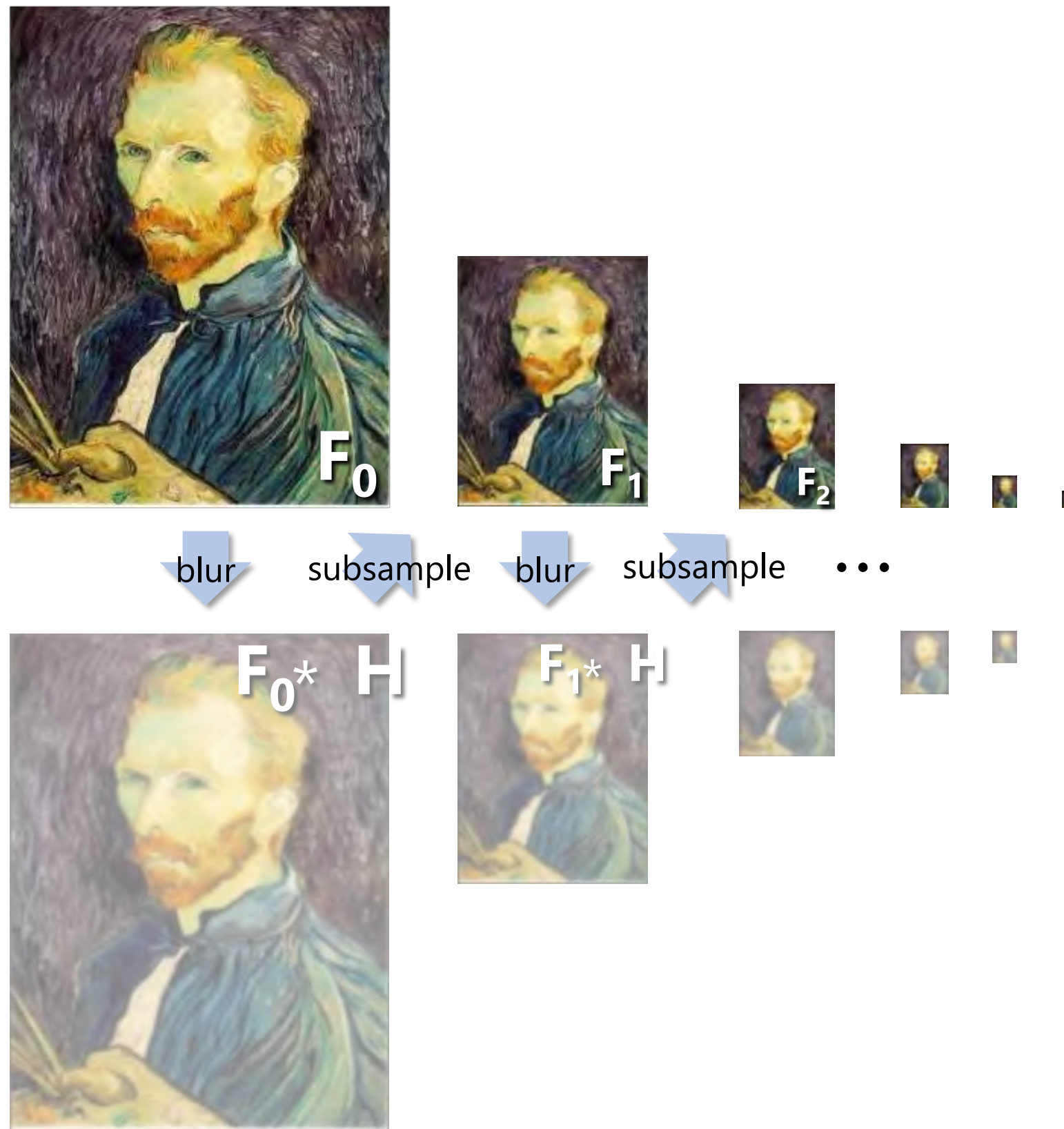
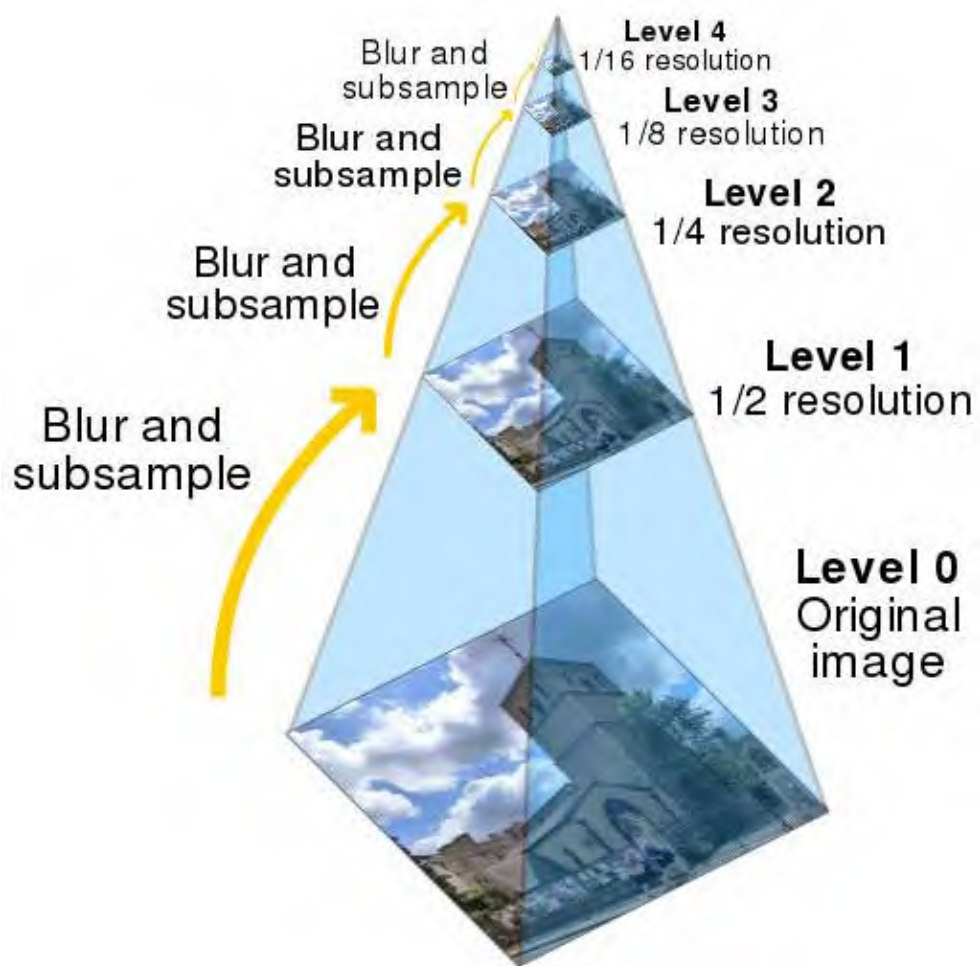
图像红框里是什么物体？

ChatGPT

红框里的物体是一只猫。它似乎是在沙发边的地板上蜷缩着，与环境的其他色彩相比，它的颜色较为淡，可能导致它不那么显眼。猫的姿态表明它可能在休息或打盹。



Gaussian pyramid
高斯金字塔



语义分割变种: DeepLab

沿用FCN/UNET网络结构

添加 Spatial pyramid pooling:

融合不同尺度的信息, 以增强分割效果
提升尺度不变性, 性能较好

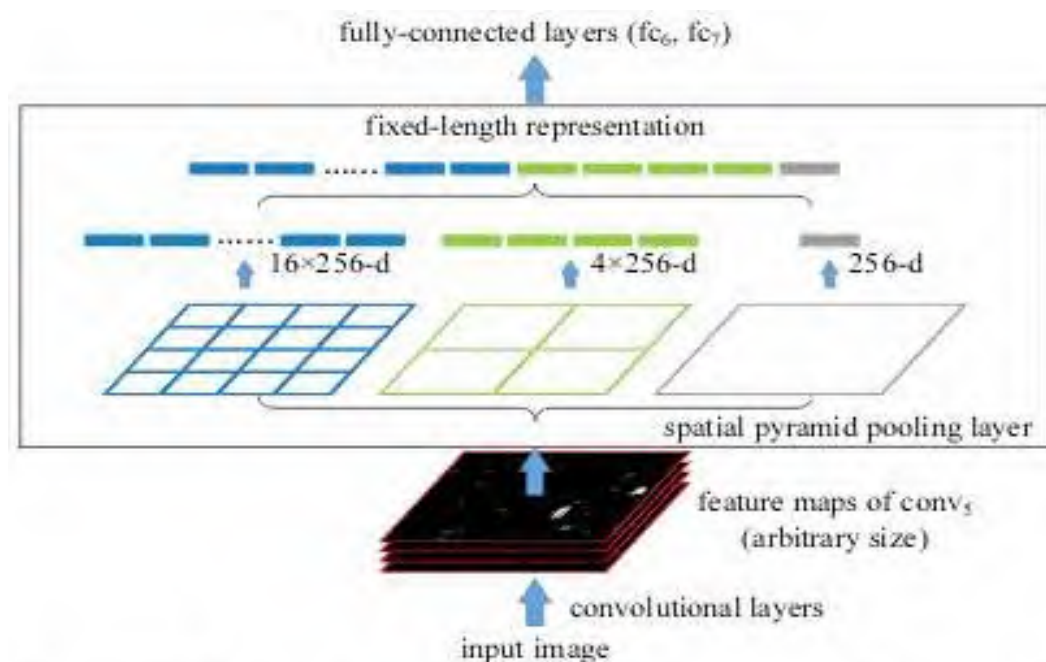


Figure 3: A network structure with a **spatial pyramid pooling layer**. Here 256 is the filter number of the conv₅ layer, and conv₅ is the last convolutional layer.

语义分割变种：HRNet

维护每个分辨率的特征

多分辨率并行多次特征融合

继续提升尺度不变性，性能更好

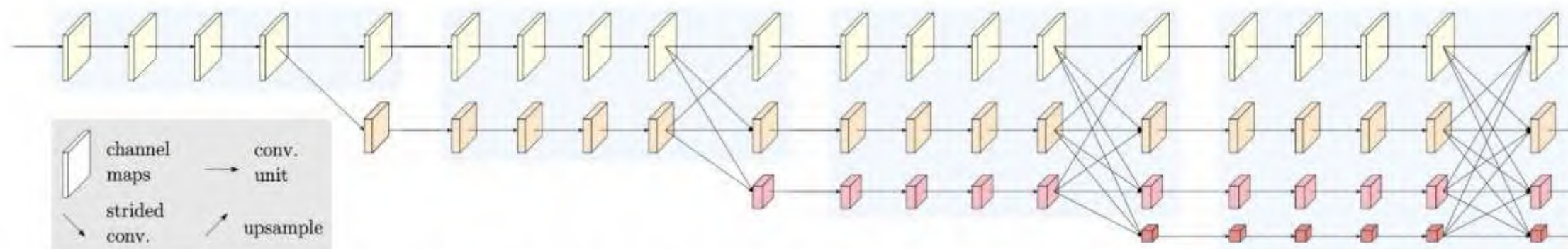


Fig. 2. An example of a high-resolution network. Only the main body is illustrated, and the stem (two stride-2 3×3 convolutions) is not included. There are four stages. The 1st stage consists of high-resolution convolutions. The 2nd (3rd, 4th) stage repeats two-resolution (three-resolution, four-resolution) blocks. The detail is given in Section 3.

实例分割与物体检测

分类



CAT

无空间限定

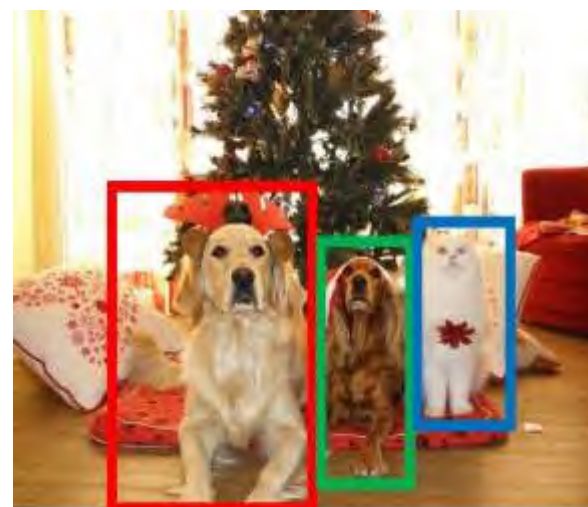
语义分割



GRASS, CAT, TREE, SKY

像素级预测

物体检测



DOG, DOG, CAT

实例分割



DOG, DOG, CAT

多物体

回顾

“语义分割”：标注每个像素点其对应的类别。

Input



Target



Object Detection: 物体检测

“Object Detection”: 对某个类别的所有实体画一个边界框

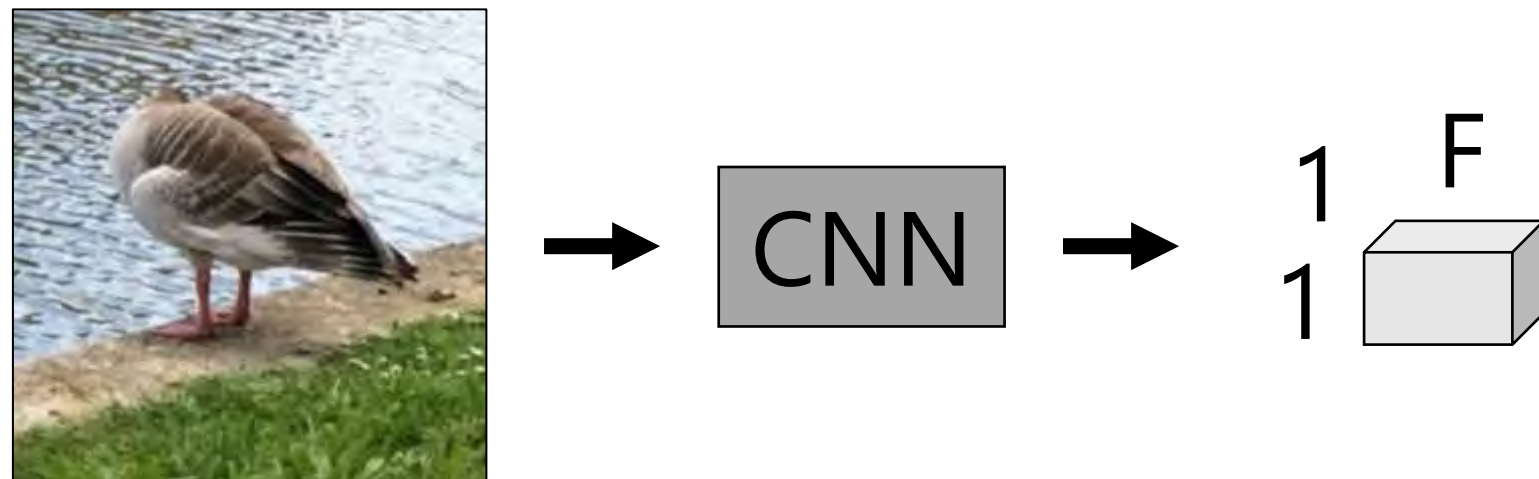
Input



Target

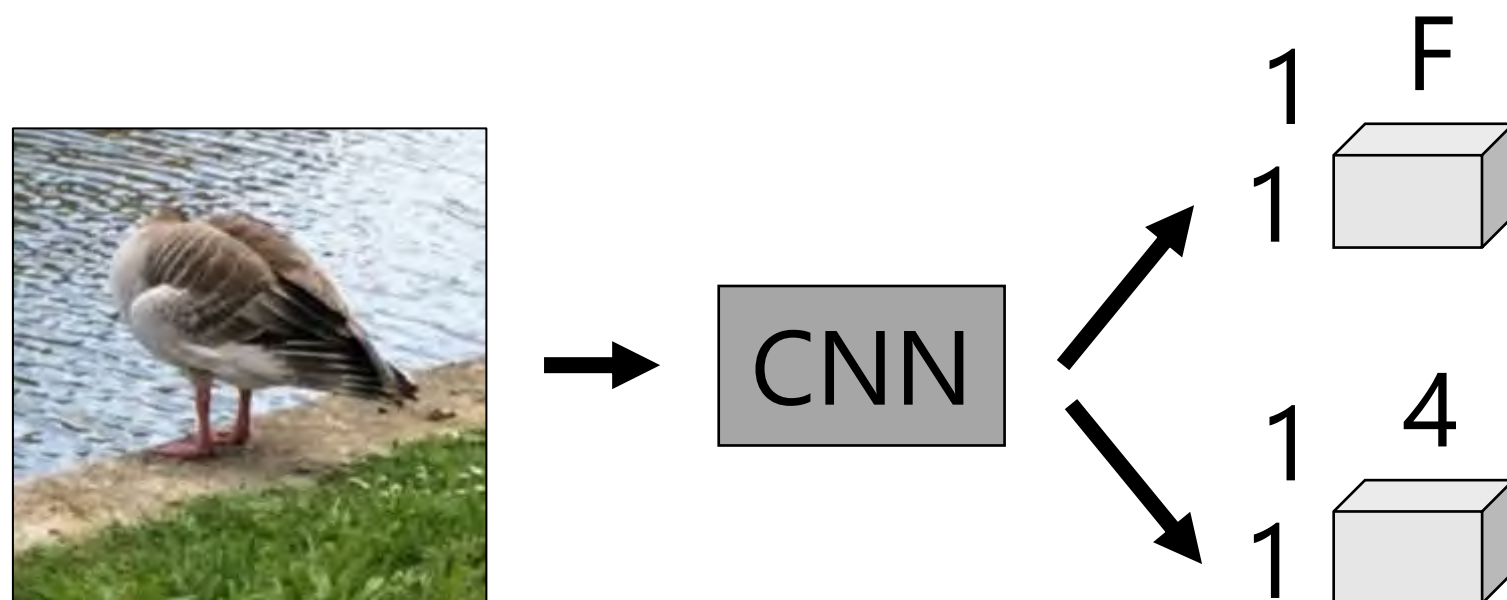


如何错误地做物体检测



让我们从图像分类开始：
可以从F个类别中进行预测
 $P(\text{cat}), P(\text{goose}), \dots P(\text{tractor})$

如何错误地做物体检测

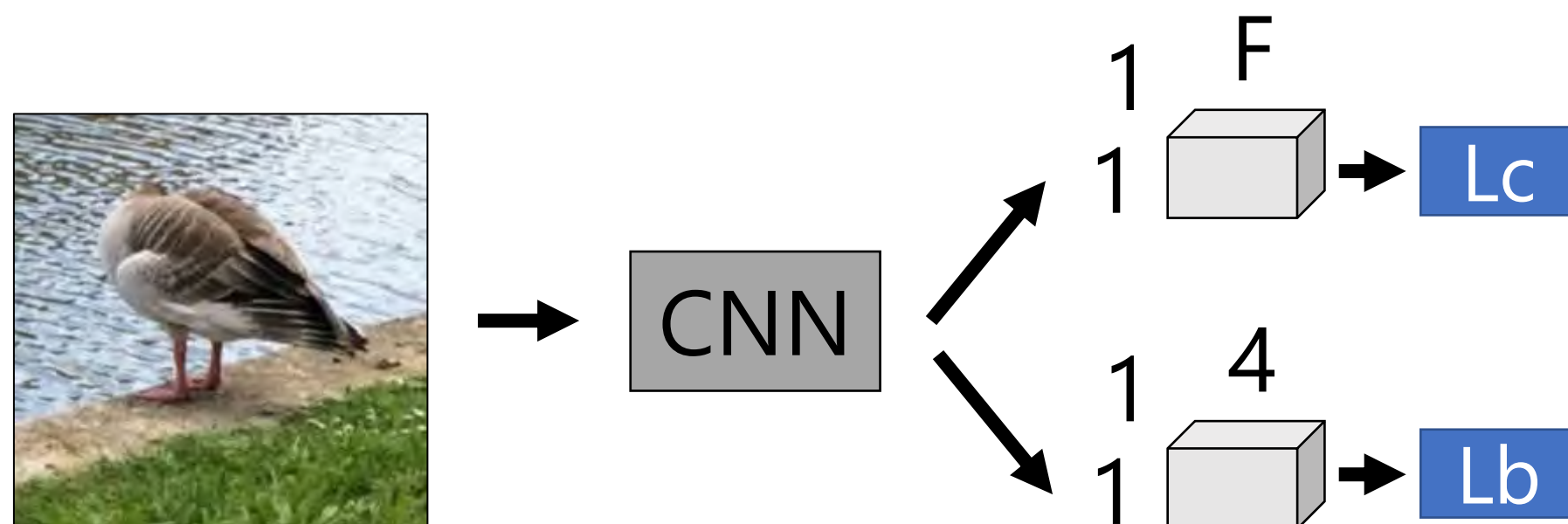


加一个额外的输出:

预测物体所在的位置

$[x,y,width,height]$ or $[minX,minY,maxX,maxY]$

如何错误地做物体检测



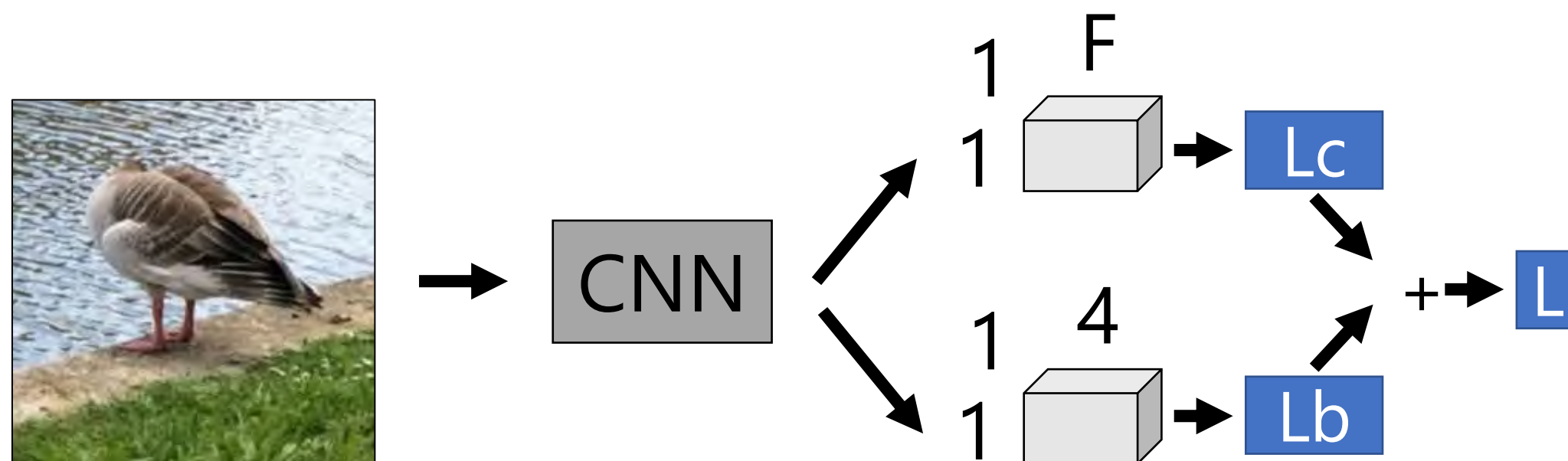
加上损失函数:

惩罚分类错误/定位错误

$L_c = \text{negative log-likelihood}$

$L_b = \text{L2 loss}$

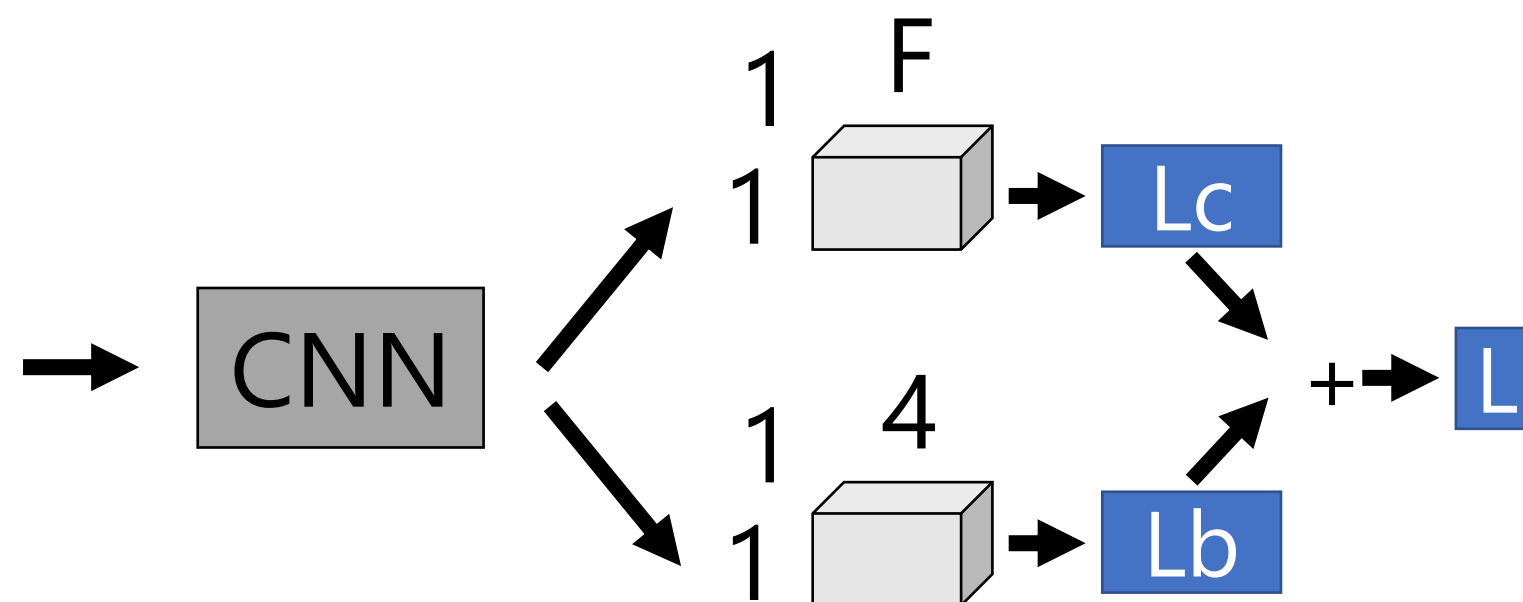
如何错误地做物体检测



计算误差，梯度反传
最终的误差: $L = Lc + \lambda Lb$

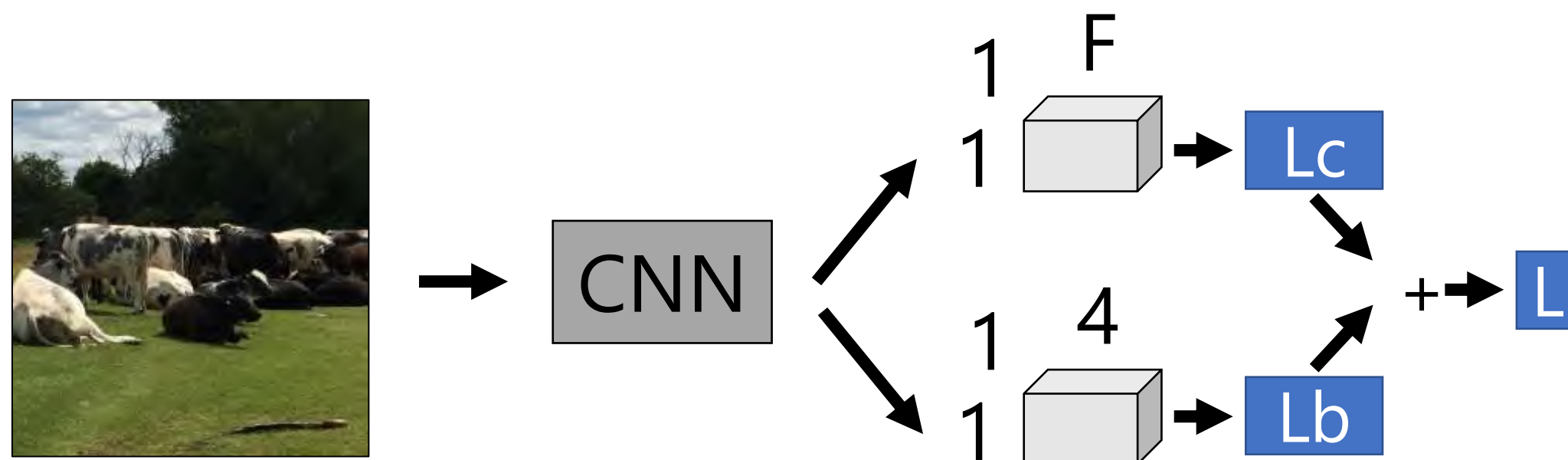
为什么会有 λ ?

如何错误地做物体检测



现在游过两只鸭了。
我们需要多少输出？
 $F, 4, F, 4 = 2*(F+4)$

如何错误地做物体检测

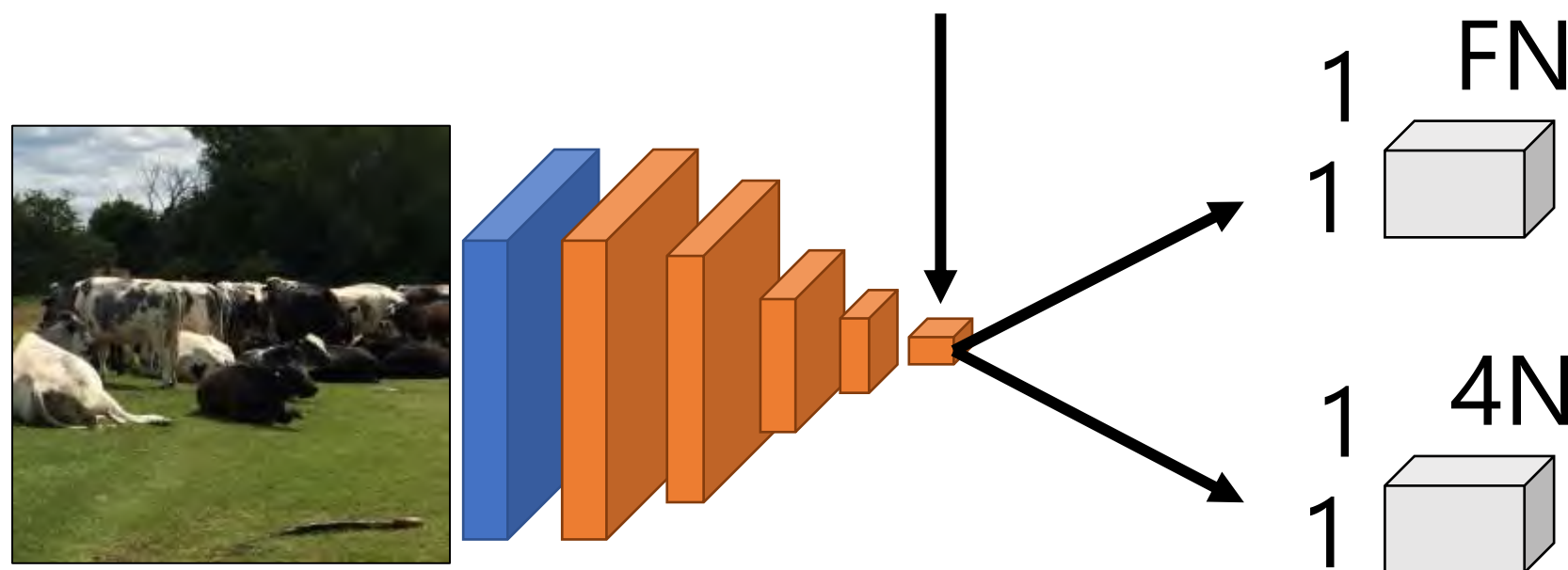


现在牛群来了。
我们需要更多的输出
(甚至根本不知道到底需要多少输出)。

总结问题——物体检测

- 我们不知道输出有多少个 因为它经常是在变化的
 - 就算我们可以, 我们怎么让网络来预测呢.

关键在于怎么让网络知道这N个物体
在哪

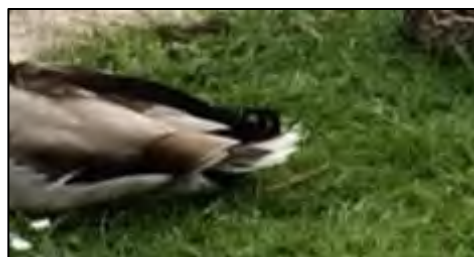


换个思路

检查所有窗口，看看这个窗口是否“紧紧包含”
某一个物体

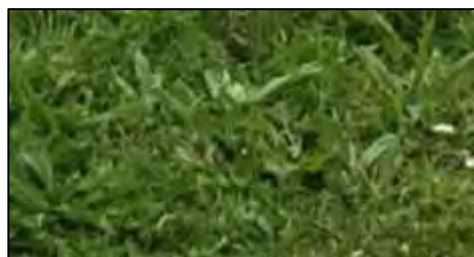


Yes



No?

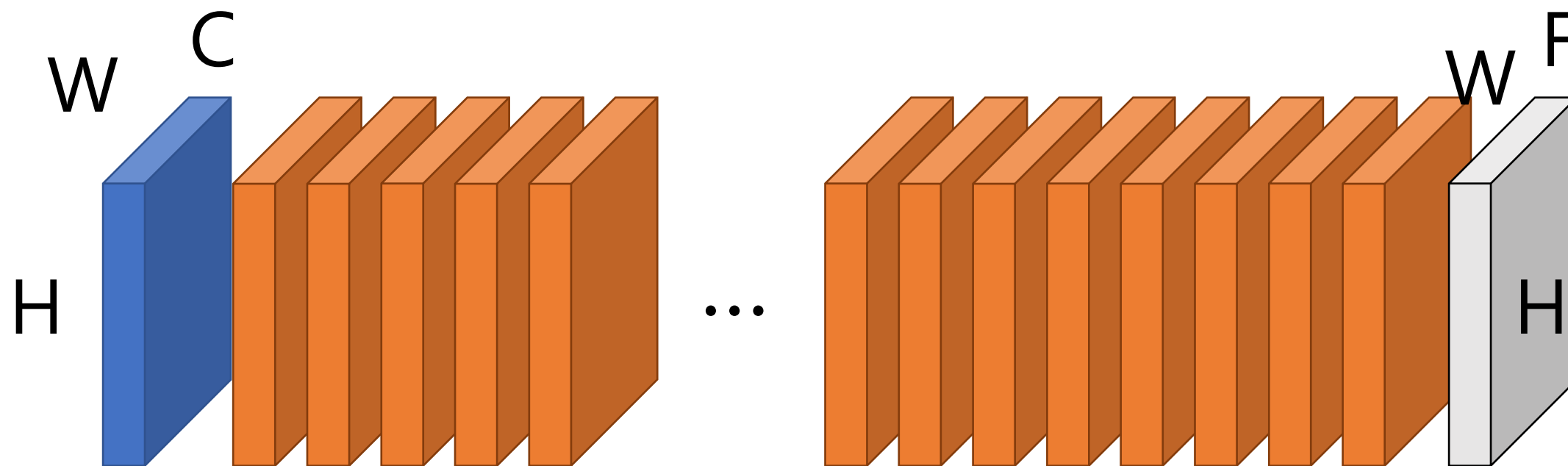
记住这个窗口!



No

回顾——语义分割、 物体检测

怎么做？堆叠卷积？



N 个 3×3 卷积的感受野是 $2n+1$ 大小的方形区域
如果能让感受野 ≥ 200 需要多少卷积？

100

滑动窗口：开销依然过大

把每个窗口的局部拿出来过CNN，
预测窗口中心的像素标签

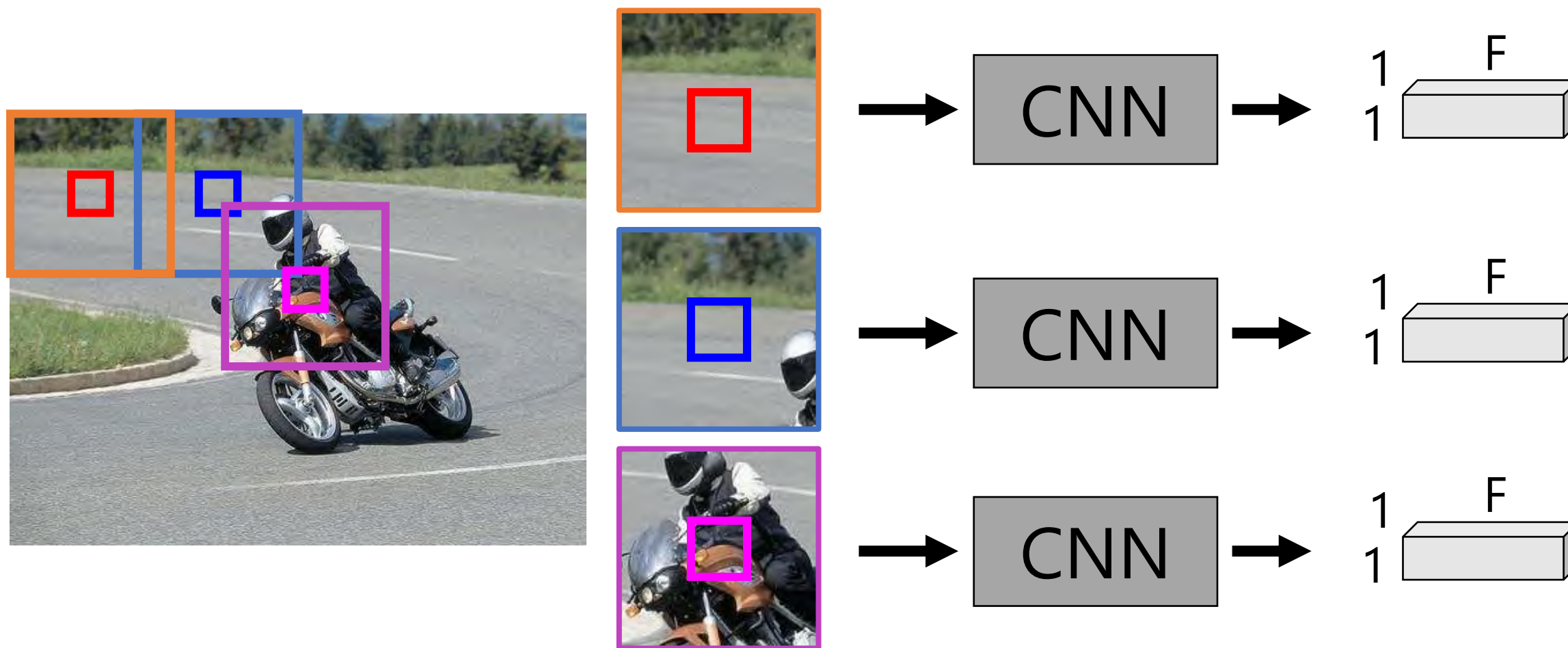


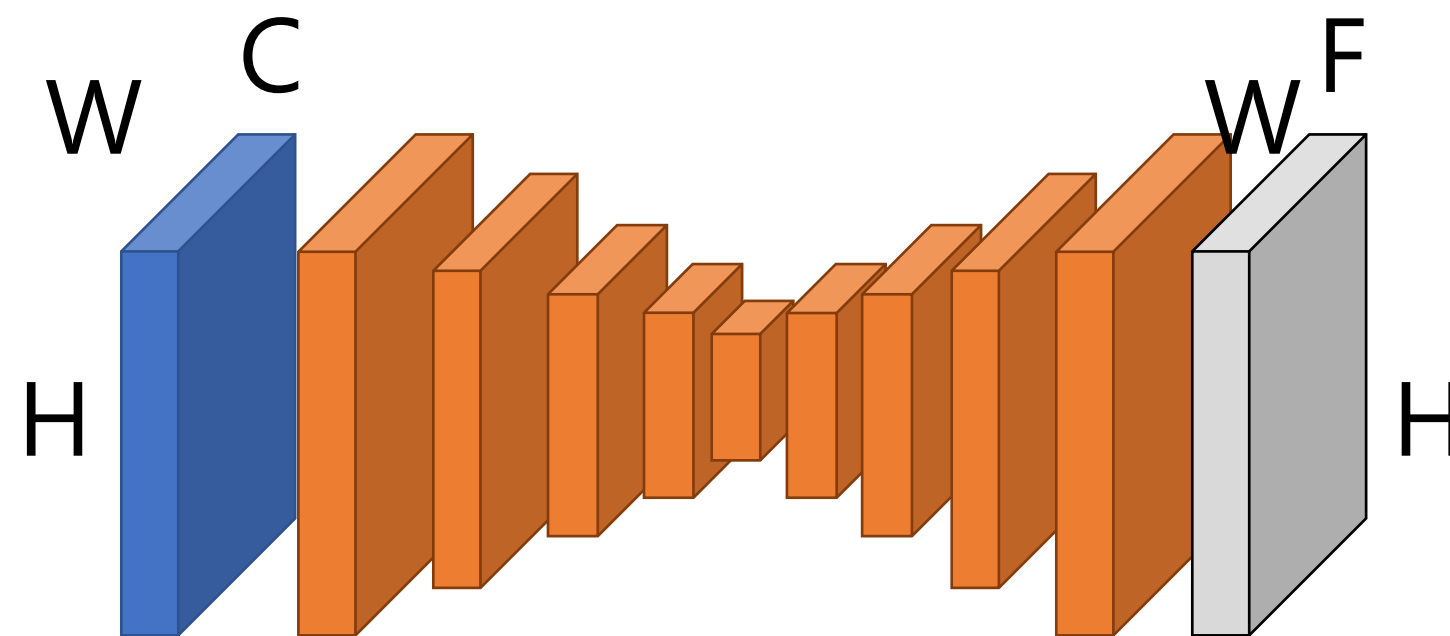
Image credit: PASCAL VOC, Everingham et al.

编码器-解码器

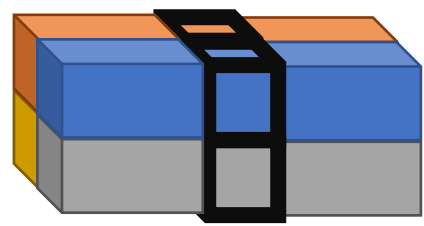
核心思想: 先 **downsample** 下采样
再 **upsample** 上采样.

怎么做下采样?

卷积, 池化



Unpooling——反池化

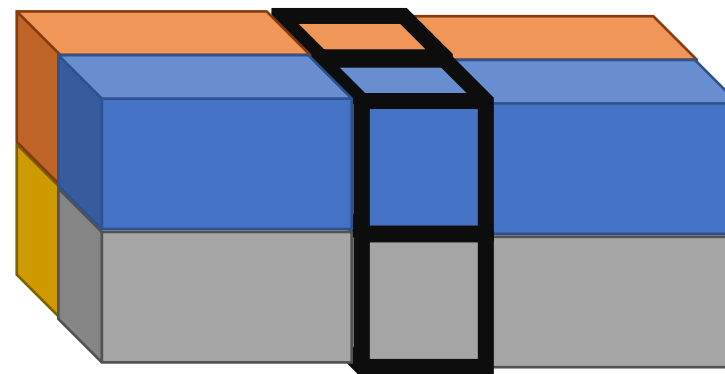


8	6
4	3

Nearest
Neighbor
Unpool
x2

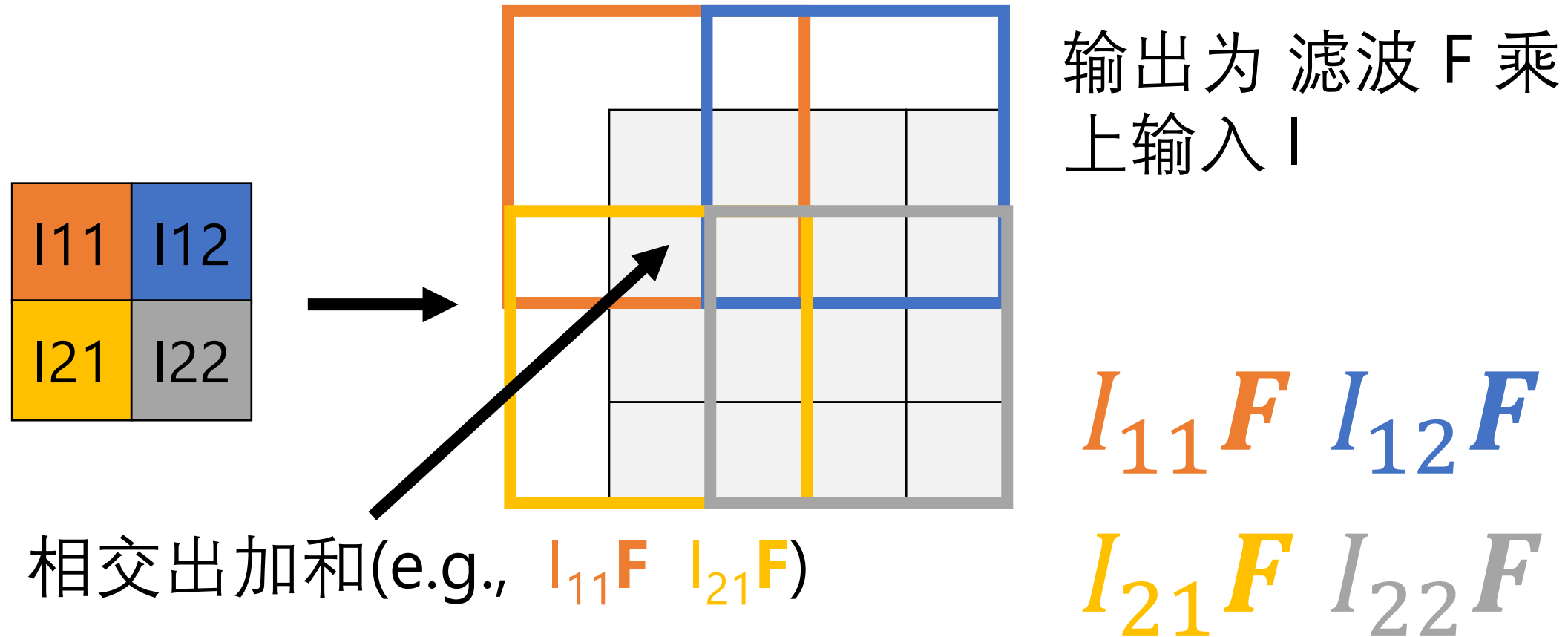


8	8	6	6
8	8	6	6
4	4	3	3
4	4	3	3



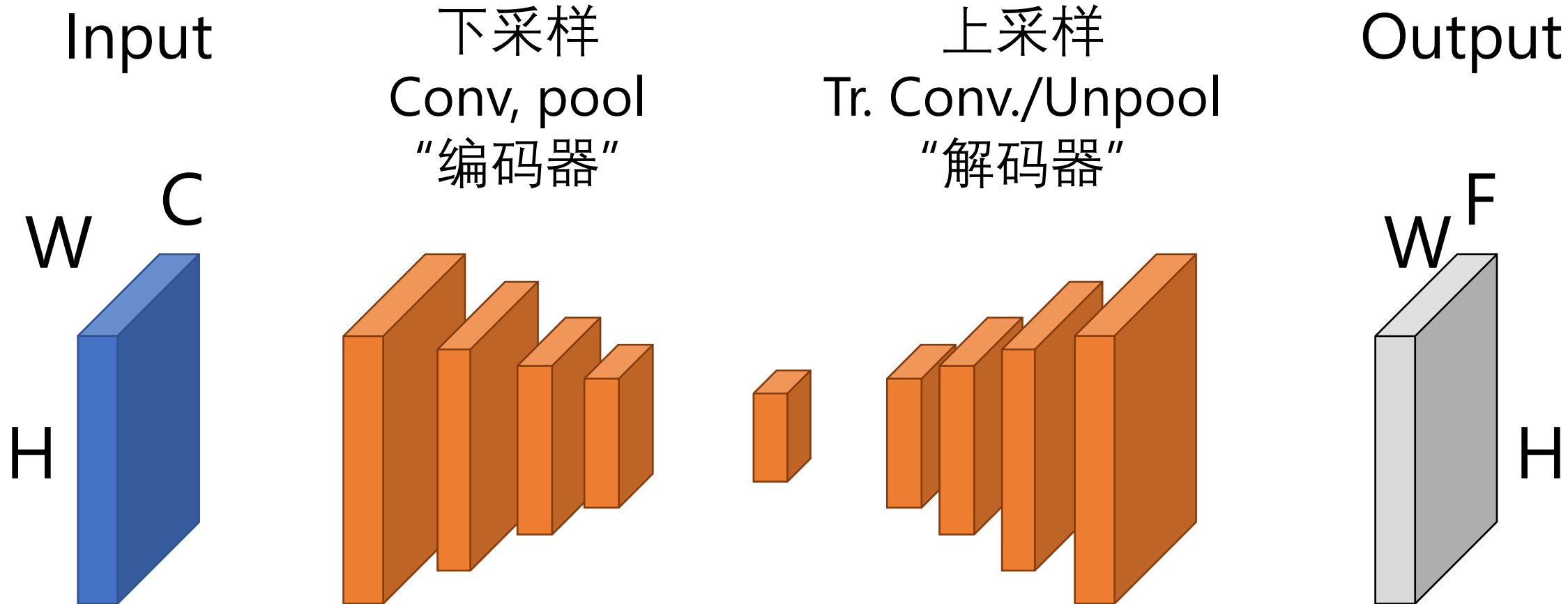
Transpose Convolution 、 DeConvolution——反卷积

3x3 Transpose Convolution, Stride 2, Pad 1



汇总起来

卷积 + 池化 来做 下采样/压缩/编码
反卷积/反池化 来做 上采样/解压缩/解码



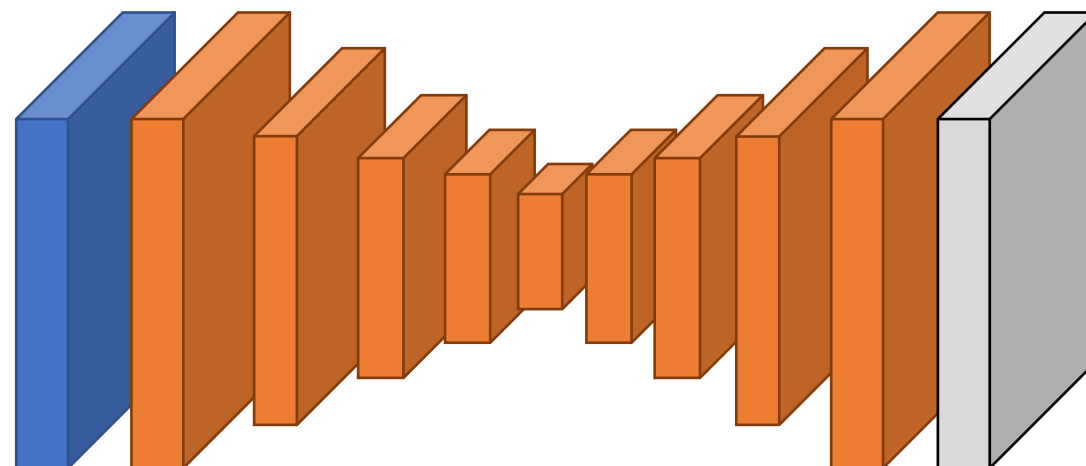
缺失的细节内容

如果输出尺寸时 $H \times W$, 直接上采样一个经过下采样的图像,
我们经常会得到一个缺失细节的结果.

为什么?



下采样时信息会丢失!

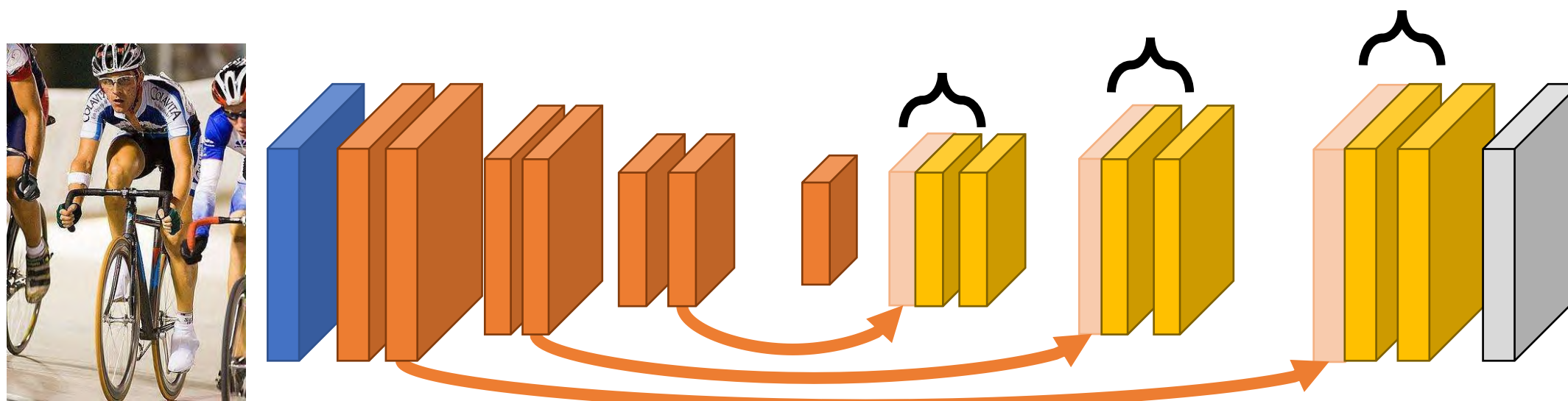


UNet

怎么把缺失的细节补回来？

复制这些层的输出。

把他们拷贝到相同分辨率的、缺失细节的地方去。



Copy

Object Detection: 物体检测

“Object Detection”: 对某个类别的所有实体画一个边界框

Input



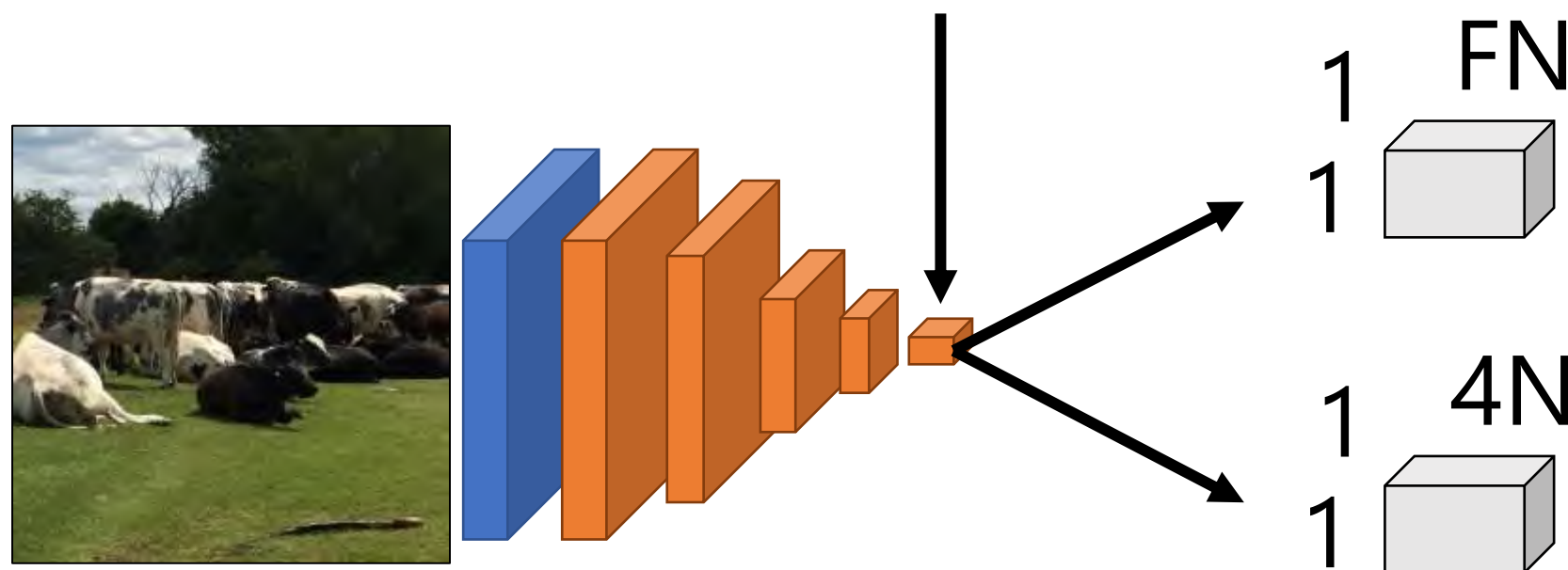
Target



总结问题——物体检测

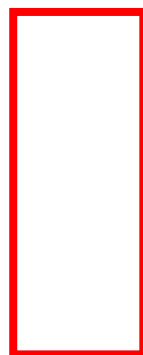
- 我们不知道输出有多少个 因为它经常是在变化的
 - 就算我们可以, 我们怎么让网络来预测呢.

关键在于怎么让网络知道这N个物体
在哪



滑动窗口检测

假设我们在固定尺寸的窗口寻找行人



滑动窗口检测

在每个窗口上寻找...



滑动窗口检测

在每个尺度每个窗口上寻找...



Note – 固定尺寸的窗口

所有的窗口



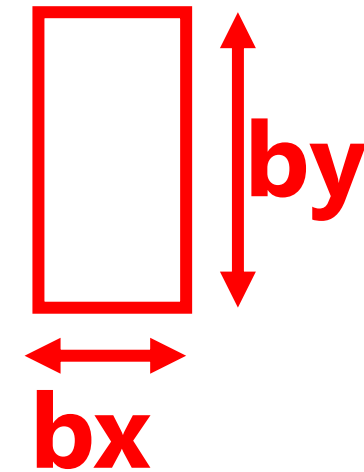
Slide credit: J. Hays

一共有多少窗口?

给定大小 $H \times W$ 的图像和一个“template 模板”大小为 by, bx .

Q. 有多少框大小为 (by, bx) ?

A. $(H-by) \times (W-bx)$



这是在考虑以下问题以前的答案:

- *scales* ($by*s, bx*s$)
- *aspect ratios* ($by*sy, bx*sx$)

物体检测的挑战

- 成千上万的框
- 却没几个有用的，包括物体的框极少

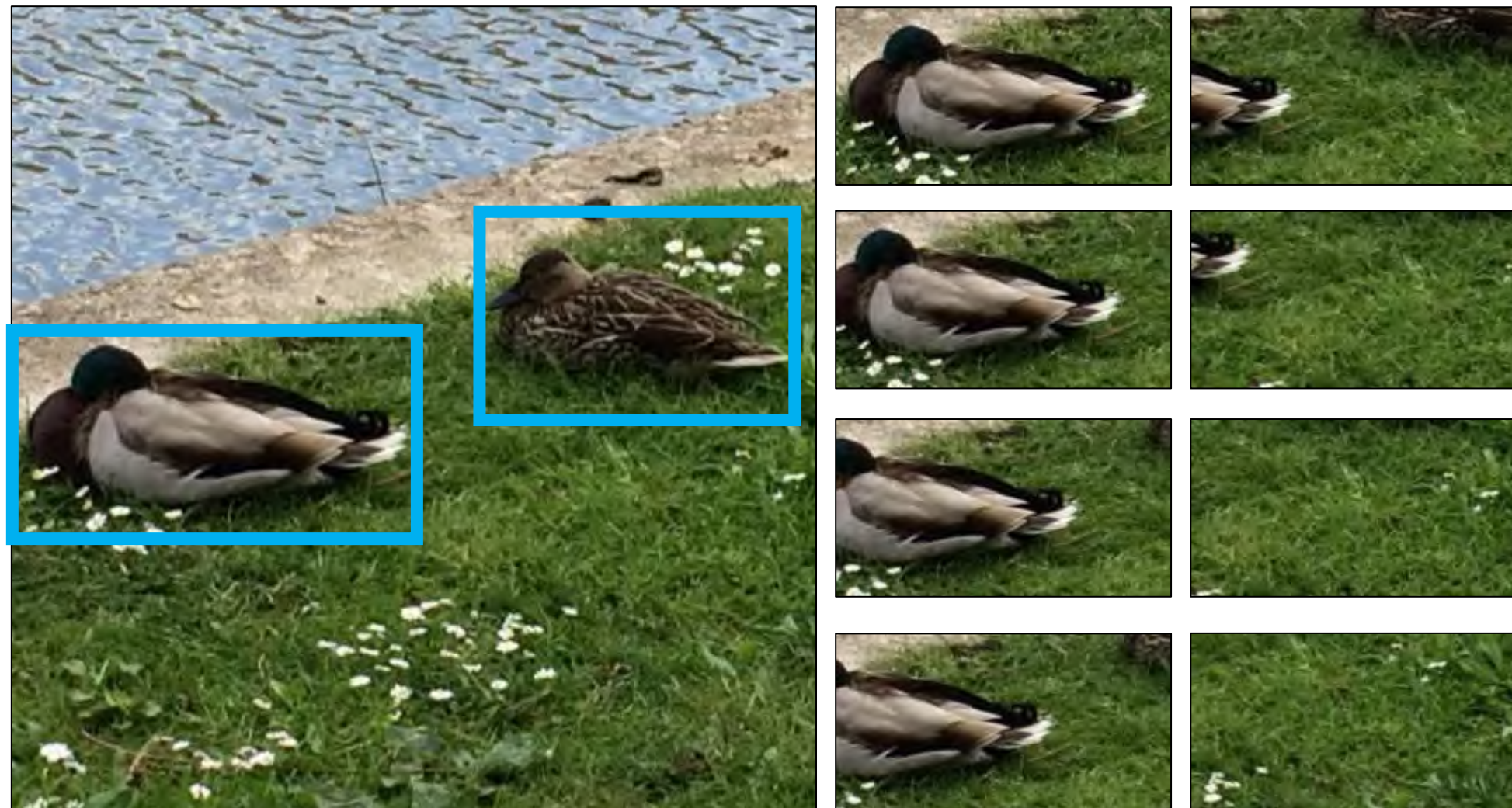


怎么整一个错误的框?

1. Wrong left x
2. Wrong right x
3. Wrong top y
4. Wrong bottom y

评估——边界框

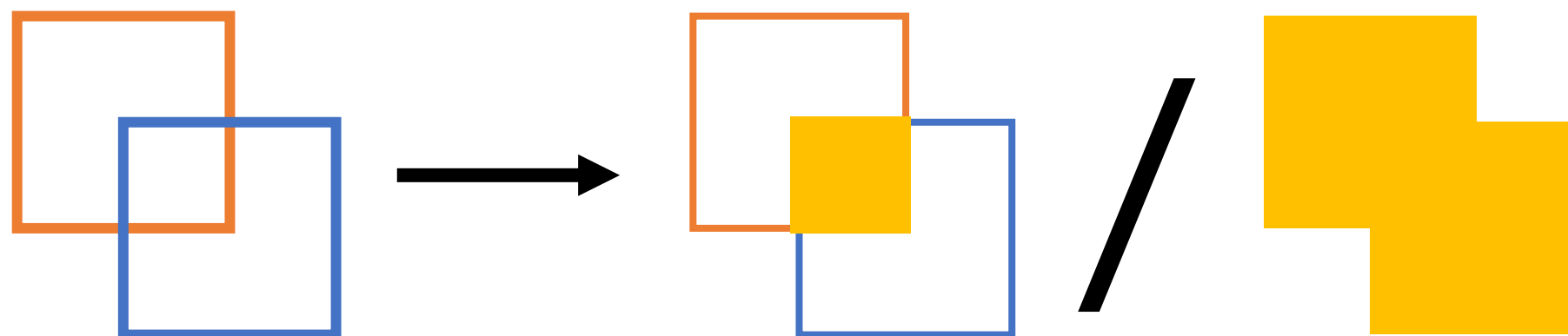
告诉我什么时候你觉得检测不再正确了。



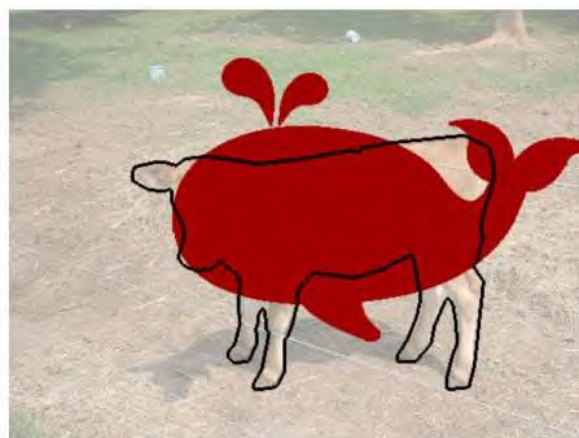
评估——边界框

标准的评价方法:

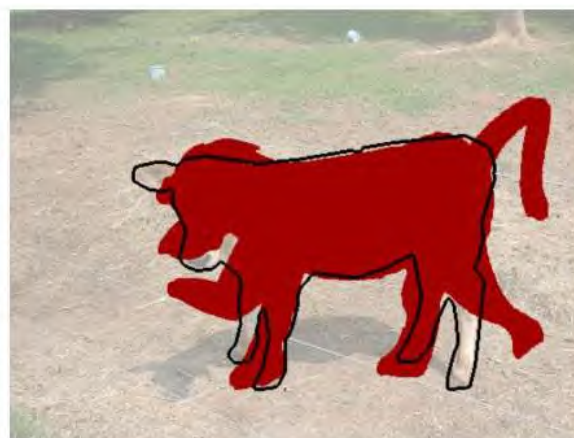
Intersection over union/IoU/Jaccard coefficient



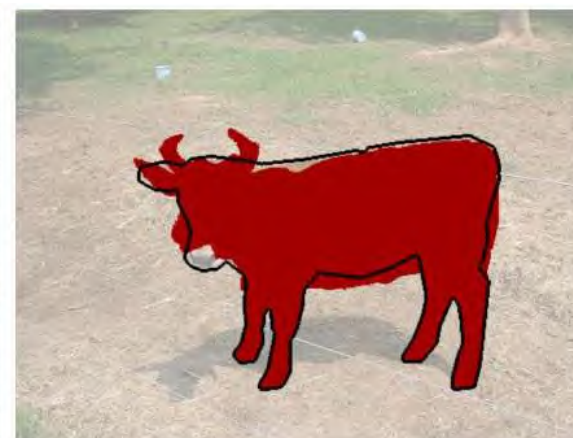
(a) Ground truth



(b) $\mathcal{J} = 0.554$



(c) $\mathcal{J} = 0.703$



(d) $\mathcal{J} = 0.910$

知识问答



你比五年级小学生聪明吗？

成年人跟五年级小学生比小学知识竞赛。

成年人不见得能赢，因为你学的知识没有用！

CV 知识问答



你的模型比随机数生成器厉害吗?

训练后的模型与随机模型相比.

如果你的评价指标不够真实, 其实并不一定哪个模型谁更好lol.

你真的比随机数生成器更智能?

- 求概率：**1000**类分类正确?
 - $1/1,000$
- 求概率：**猜边界框的四个角，且误差小于10% 图像大小?**
 - $(1/10)*(1/10)*(1/10)*(1/10)=1/10,000$
- 两者概率叠加: $1/10,000,000$
- 不能用最常出现的标签（没有物体）来直接预测

性能评估

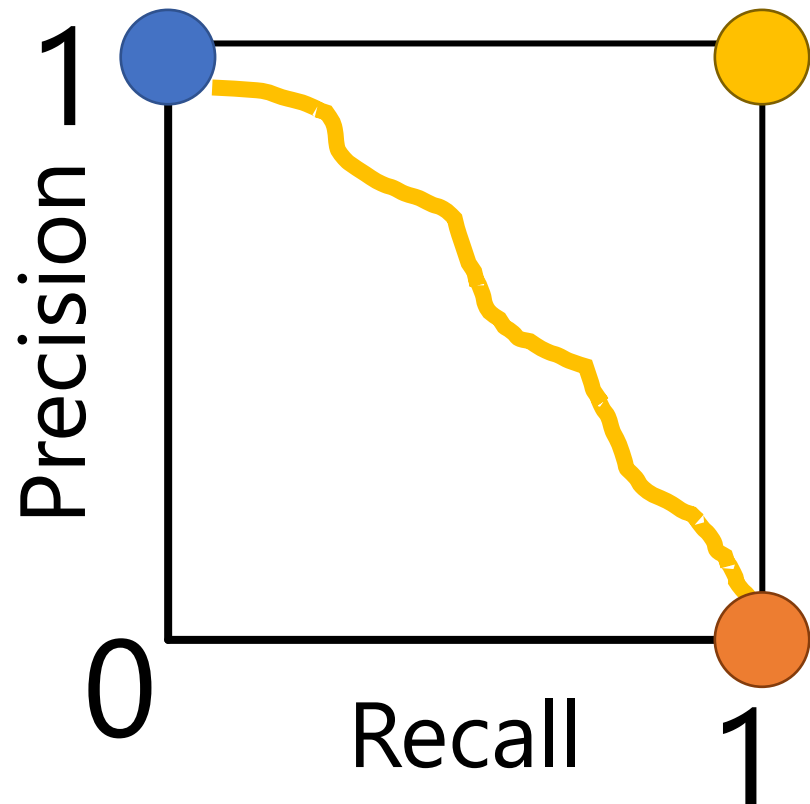
- 准确率 = 预测正确的概率
- 如果我声称我的行人检测99%正确.
- **这代表我的检测做的很好吗?**
- 大街上没有多少人!



性能评估

- True detection 真实检测 (true positive) : 高 IoU
- Precision 精确度: $\# \text{真实检测} / \# \text{检测总数}$
- Recall 召回率: $\# \text{真实检测} / \# \text{真实的物体数量}$

如果拒绝一切，那么就不会有错误

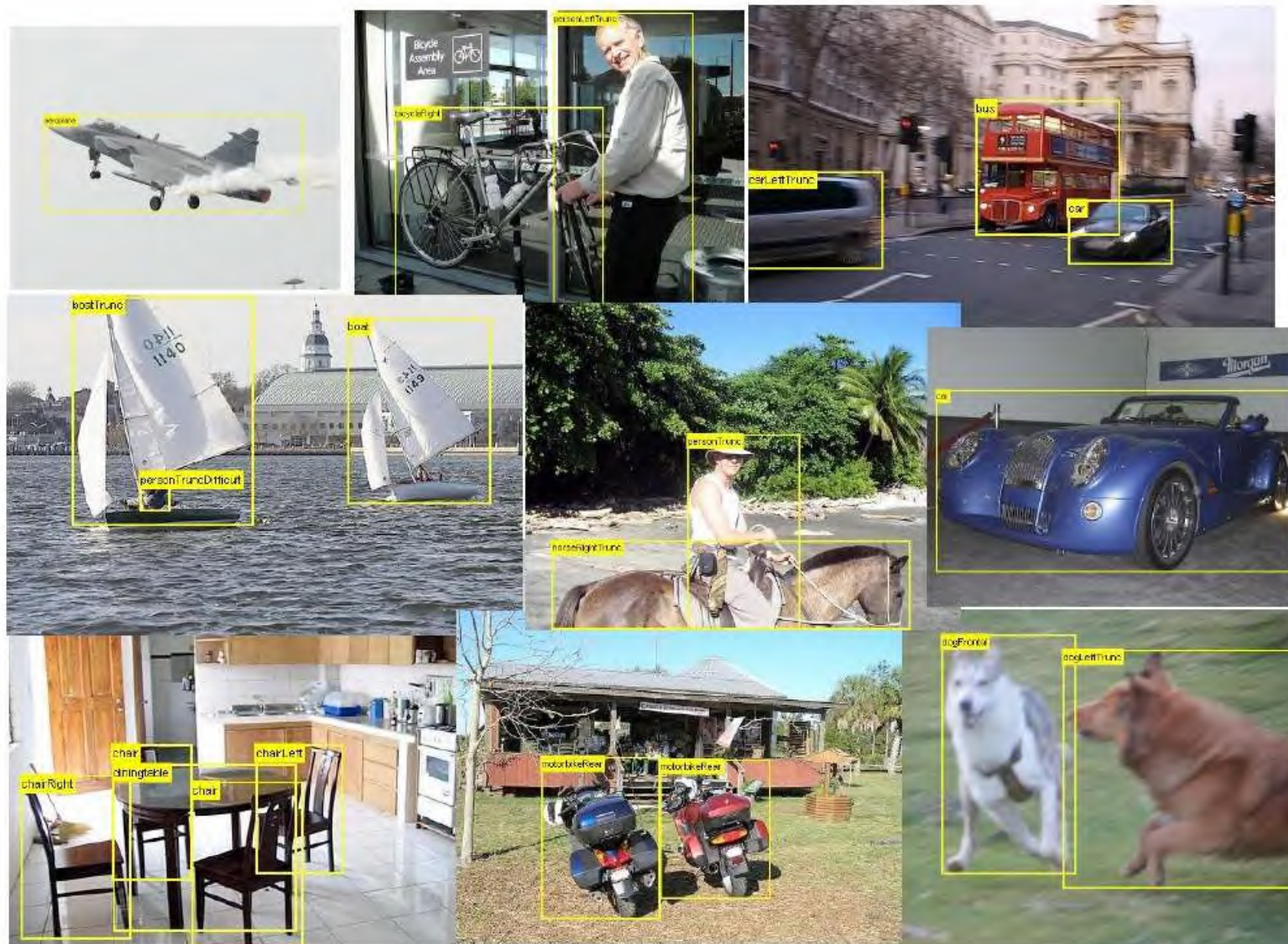


理想情况是精确度和召回率都很高!

AUC area under curve (avg. precision)

如果接受一切，那么就不会错过任何东西

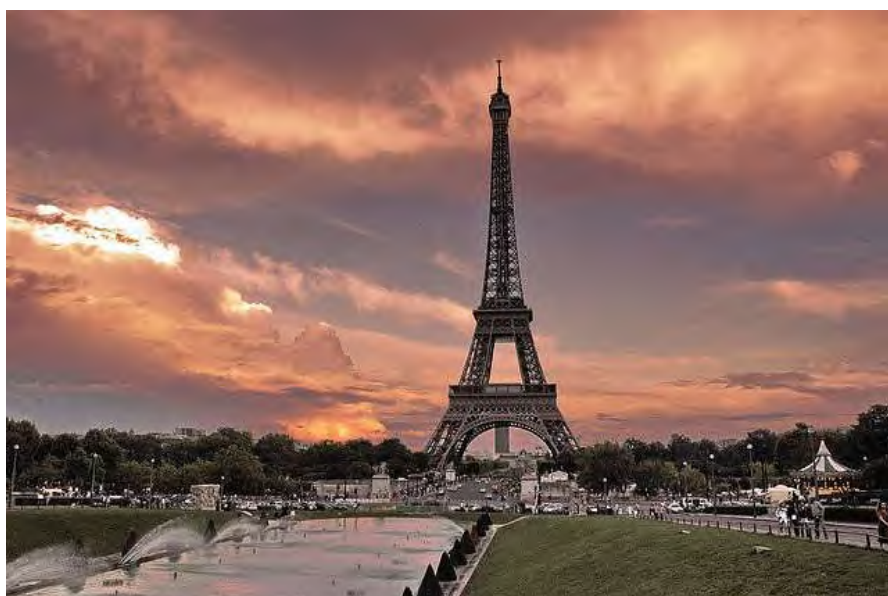
各式物体检测



传统方法：Histograms of oriented gradients (HOG)

将图像分割成多个块，并在每个块中计算梯度方向的直方图

$H \times W \times 3$ Image



$H' \times W' \times C'$ Image

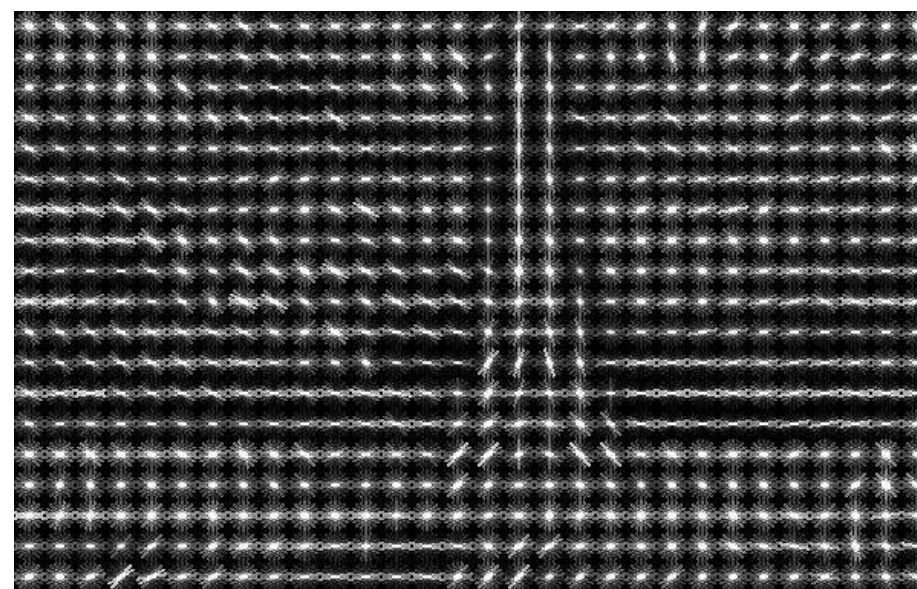


Image credit: N. Snavely

使用HOG进行行人检测

- 使用线性支持向量机 (SVM) 来训练一个行人模板

正训练样本：包含行人



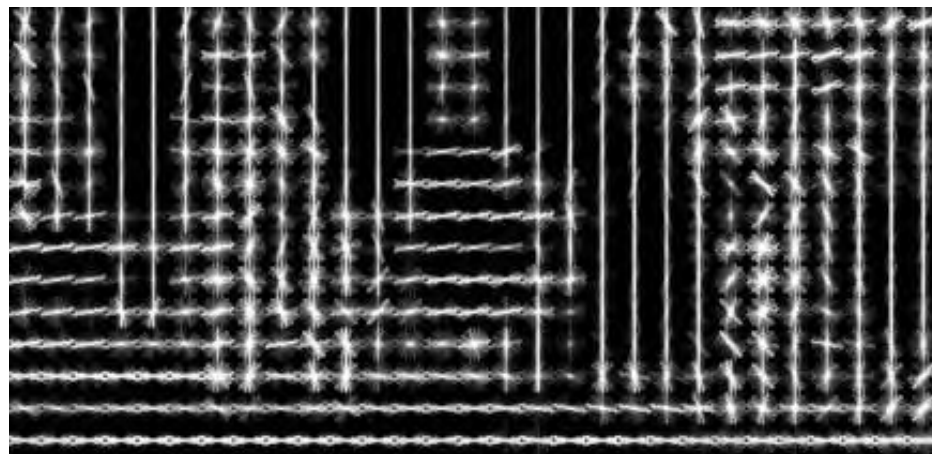
负训练样本：没有行人



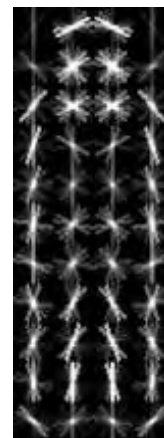
使用HOG进行行人检测

- 用线性SVM训练行人“模板”
- 测试时对HOG特征图用模板做卷积
- 找到局部最大响应值
- 不要忘了应对多尺度： HOG *pyramid* 金字塔

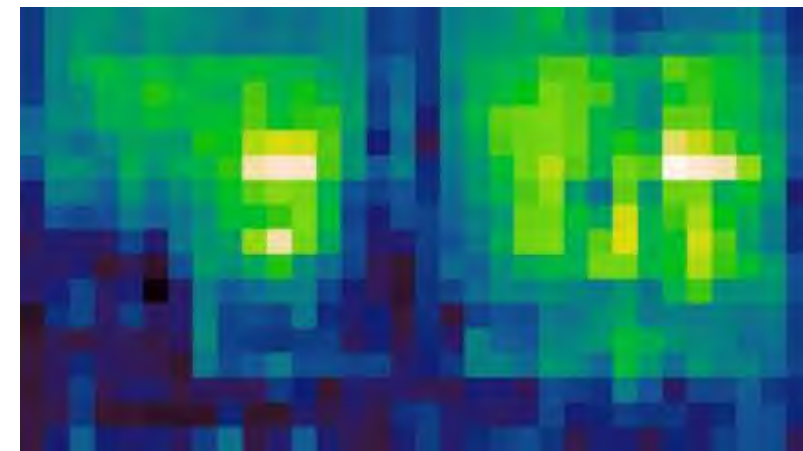
HOG 特征图



模板



检测器响应图



检测效果



[Dalal and Triggs, CVPR 2005]

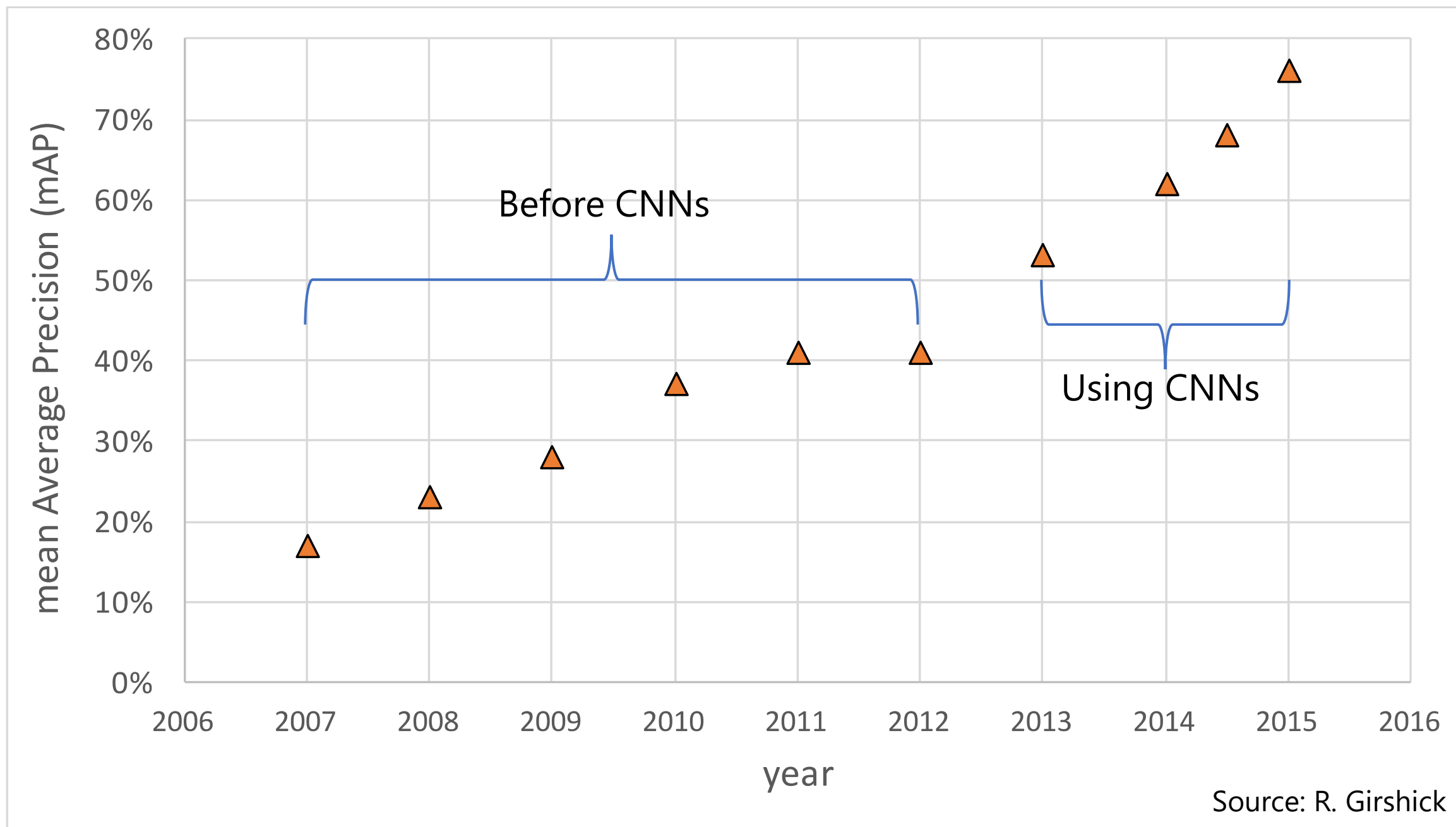
PASCAL VOC 数据集 (2005-2012)



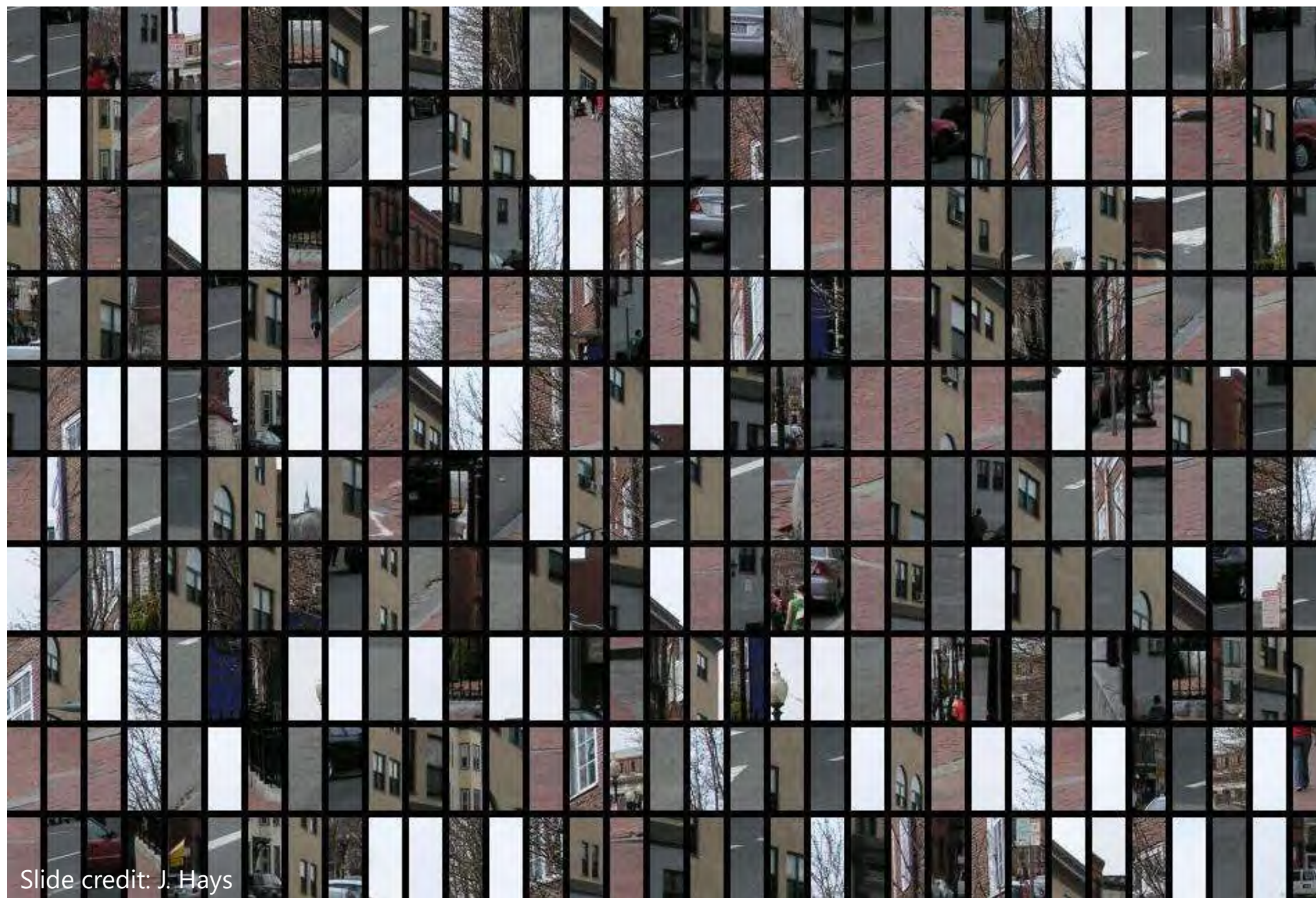
- 20 个类别:
- *Person*
- *Animals*: bird, cat, cow, dog, horse, sheep
- *Vehicles*: aeroplane, bicycle, boat, bus, car, motorbike, train
- *Indoor*: bottle, chair, dining table, potted plant, sofa, tv/monitor
- 数据集大小 (2012年): 11.5K 训练/校验图像, 27K边界框, 7K分割图

物体检测的性能发展

PASCAL VOC



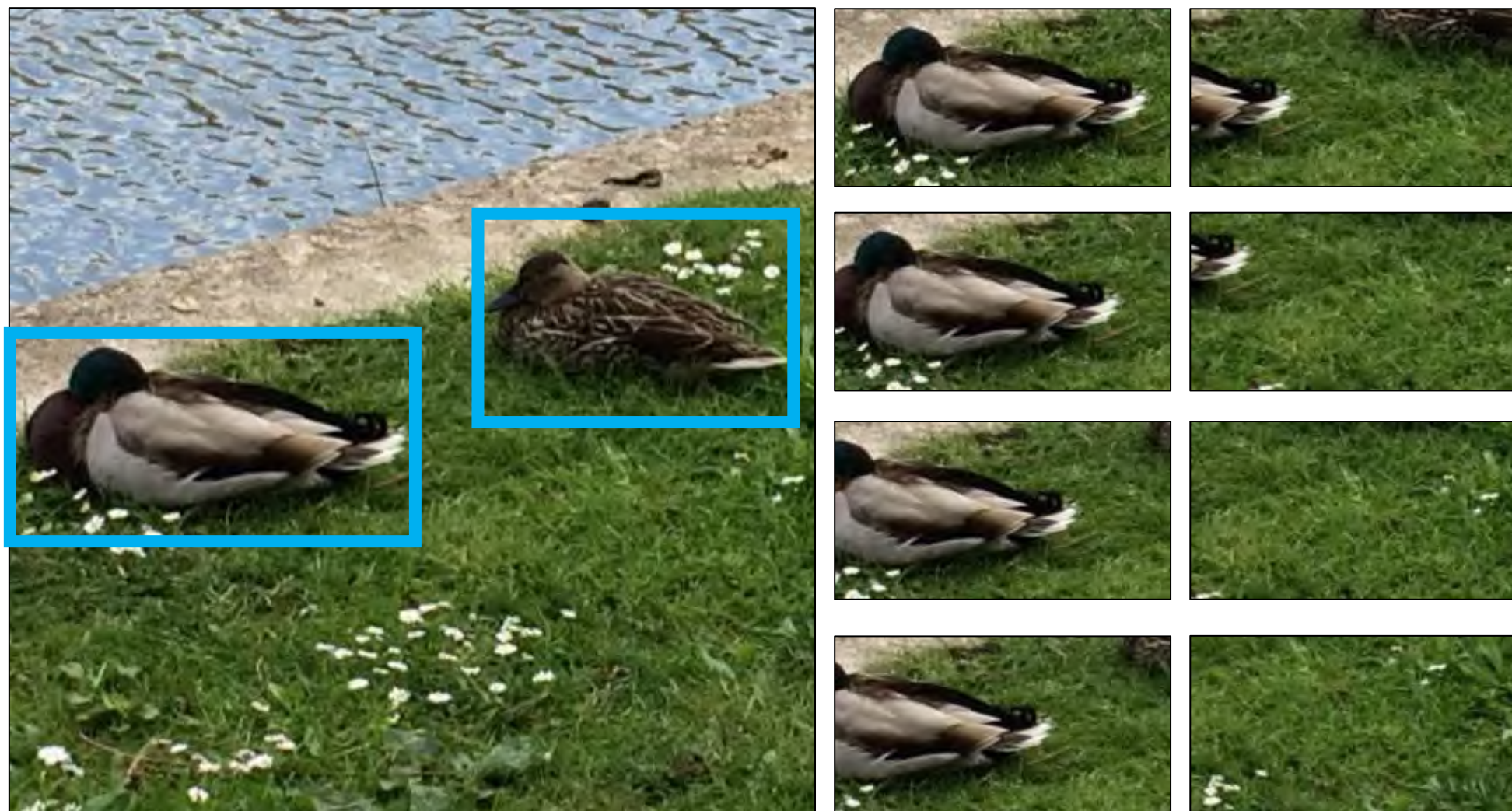
回顾：所有的窗口——只有少量窗口有目标物体



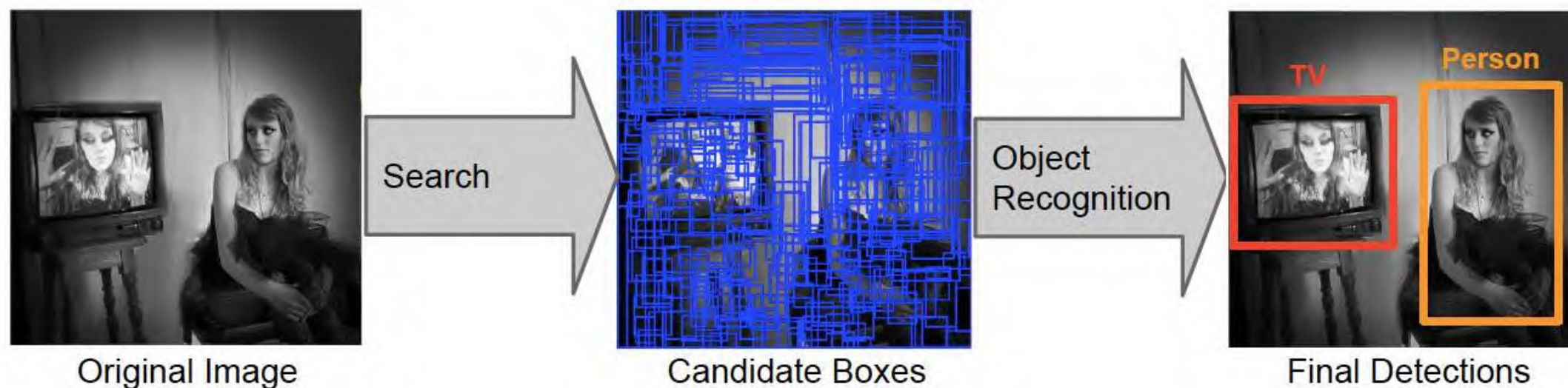
Slide credit: J. Hays

Region Proposals 区域提议

我们需要花时间对草地的所有框进行筛选吗？

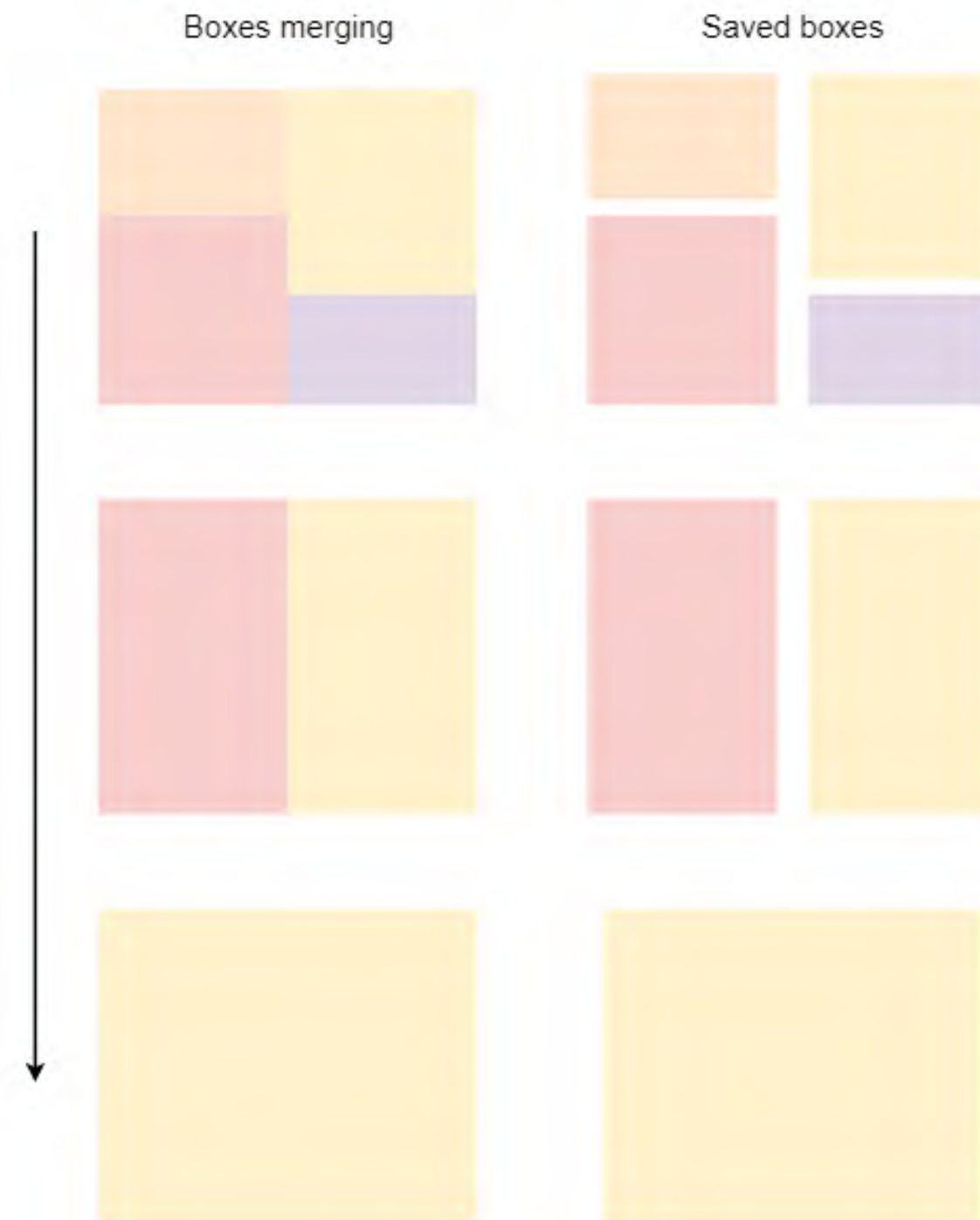


Region Proposals 区域提议



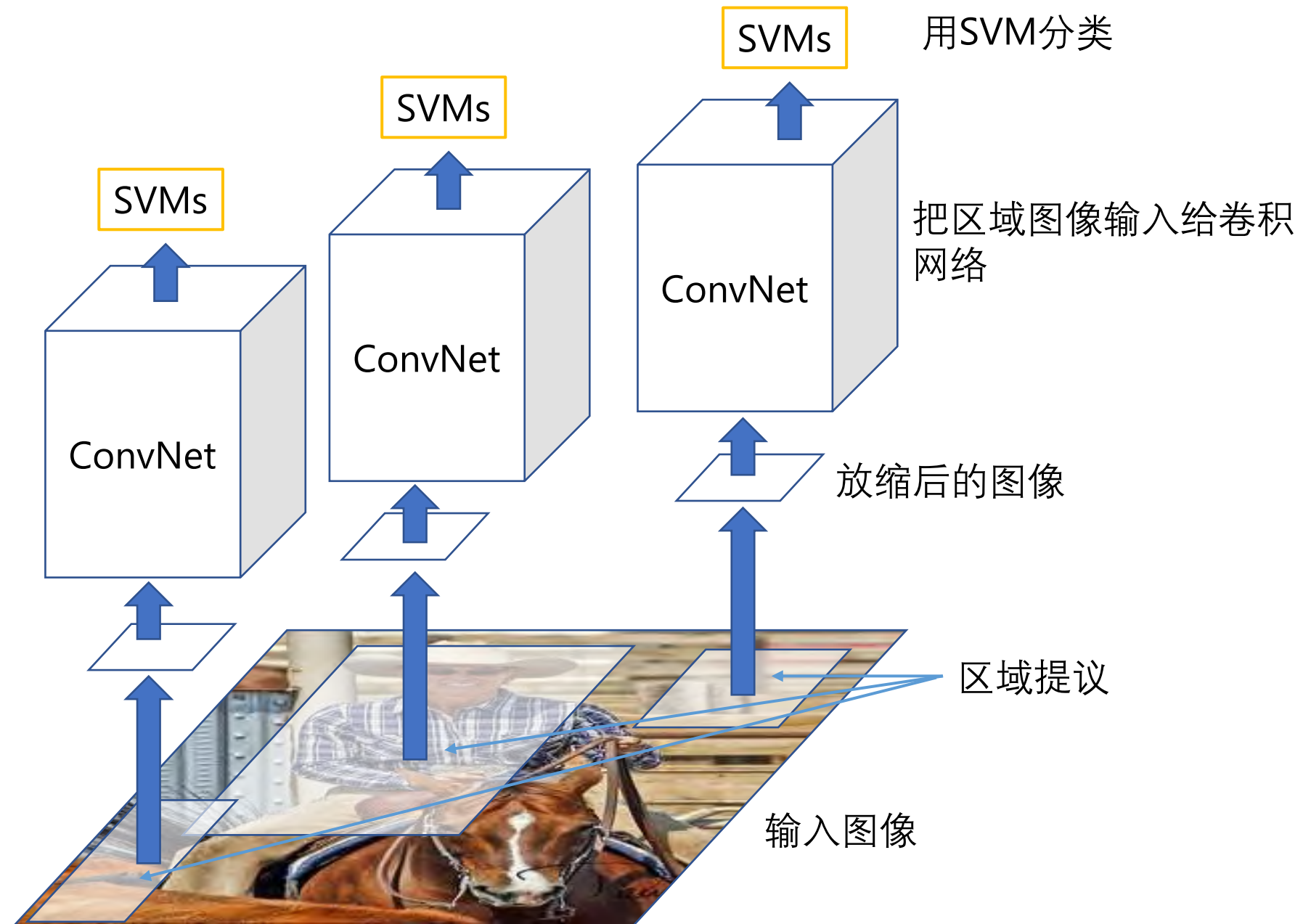
- 替代滑动窗口，只选出部分候选框作为 *region proposals* 区域提议
 - 谨记：特征提取和分类器速度较慢，区域提议可以减少候选区域，提升运行效率
 - 一般候选区域类别无关
 - 这个机制可以被训练（后面会讲）

Region Proposals: Selective Search

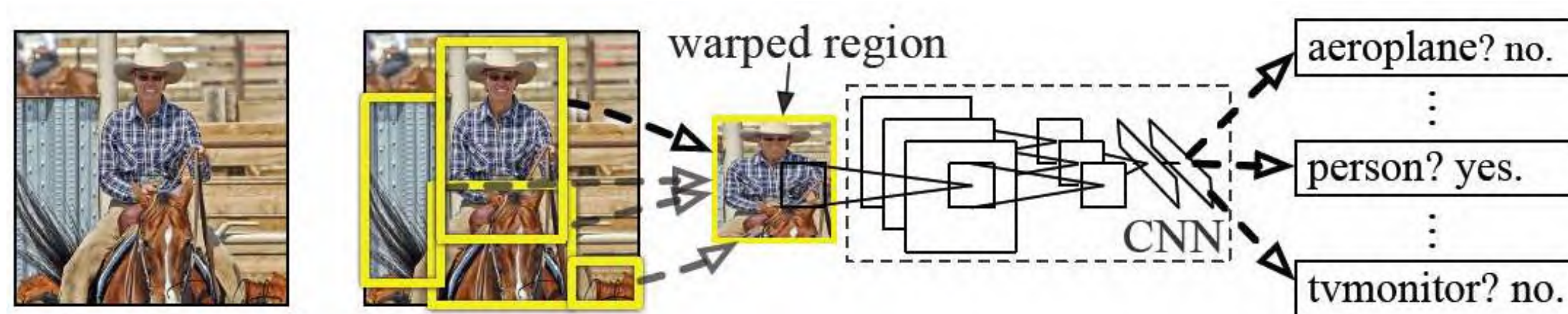


R-CNN: Region proposals + CNN features

Source: R. Girshick



R-CNN



- 候选区域: ~2000 Selective Search 候选区域
- 网络: AlexNet 预训练模型, ImageNet上预训练(1000类), PASCAL 上微调 (21类: 20+背景)
- 检测器: 放缩图像输入卷积网络, 使用 fc7 层特征 (4096维), 使用SVM分类
- 同时使用SVM回归候选框
- **性能:** mAP of 53.7% on PASCAL 2010 (vs. 35.1% for Selective Search and 33.4% for DPM).

R-CNN pros and cons

- **Pros**

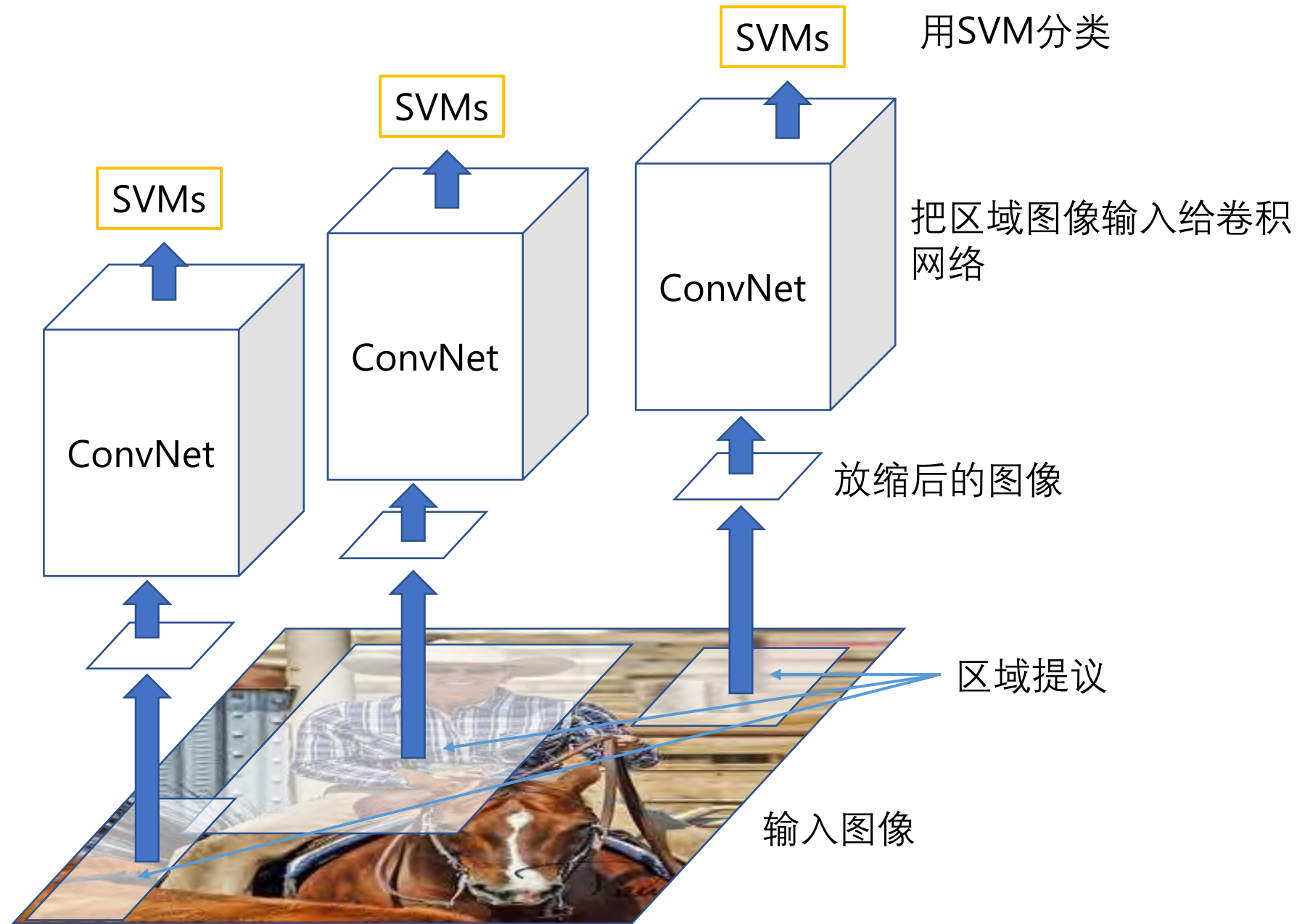
- 准! (相较以前的模型)
- 任何深度学习架构都可以即插即用。

- **Cons**

- Ad hoc: 没有统一的端到端训练, 而是基于不同的需求独立设置。
 - 网络需要用softmax分类器进行微调, 这一步使用的是对数损失 (log loss)
 - 训练后处理的线性支持向量机, 使用截断损失(hinge loss)
 - 训练后处理的边界框回归使用的是回归损失(least squares)
- 训练过程慢, 需要84小时, 且占用大量磁盘空间
 - 每张图像需要2000次CNN
- 推理 (检测) 过程慢, 使用VGG16模型时, 每张图像需要47秒

“Slow” R-CNN

Source: R. Girshick

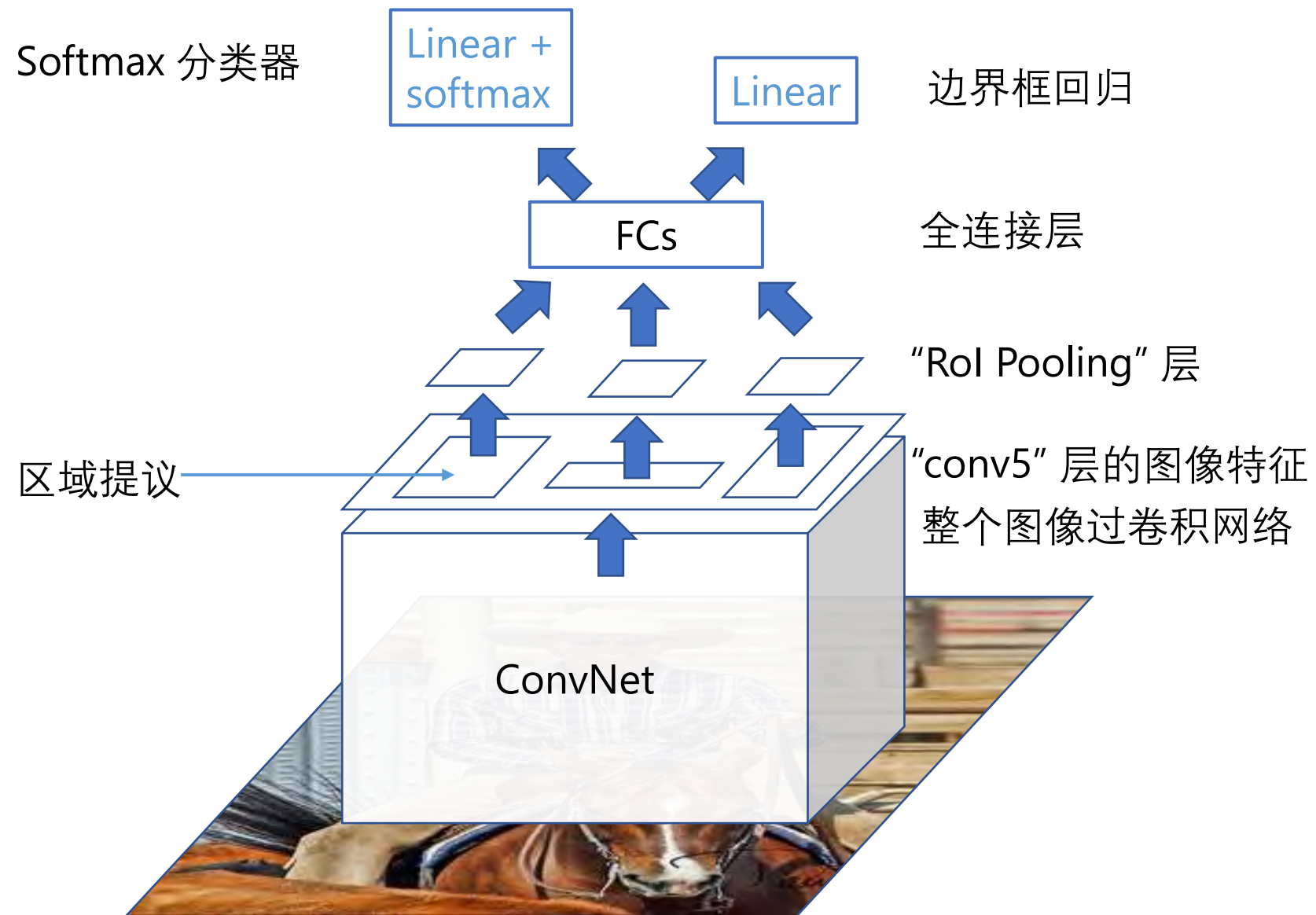


问题: 很慢!每个区域需要重新跑卷积网络

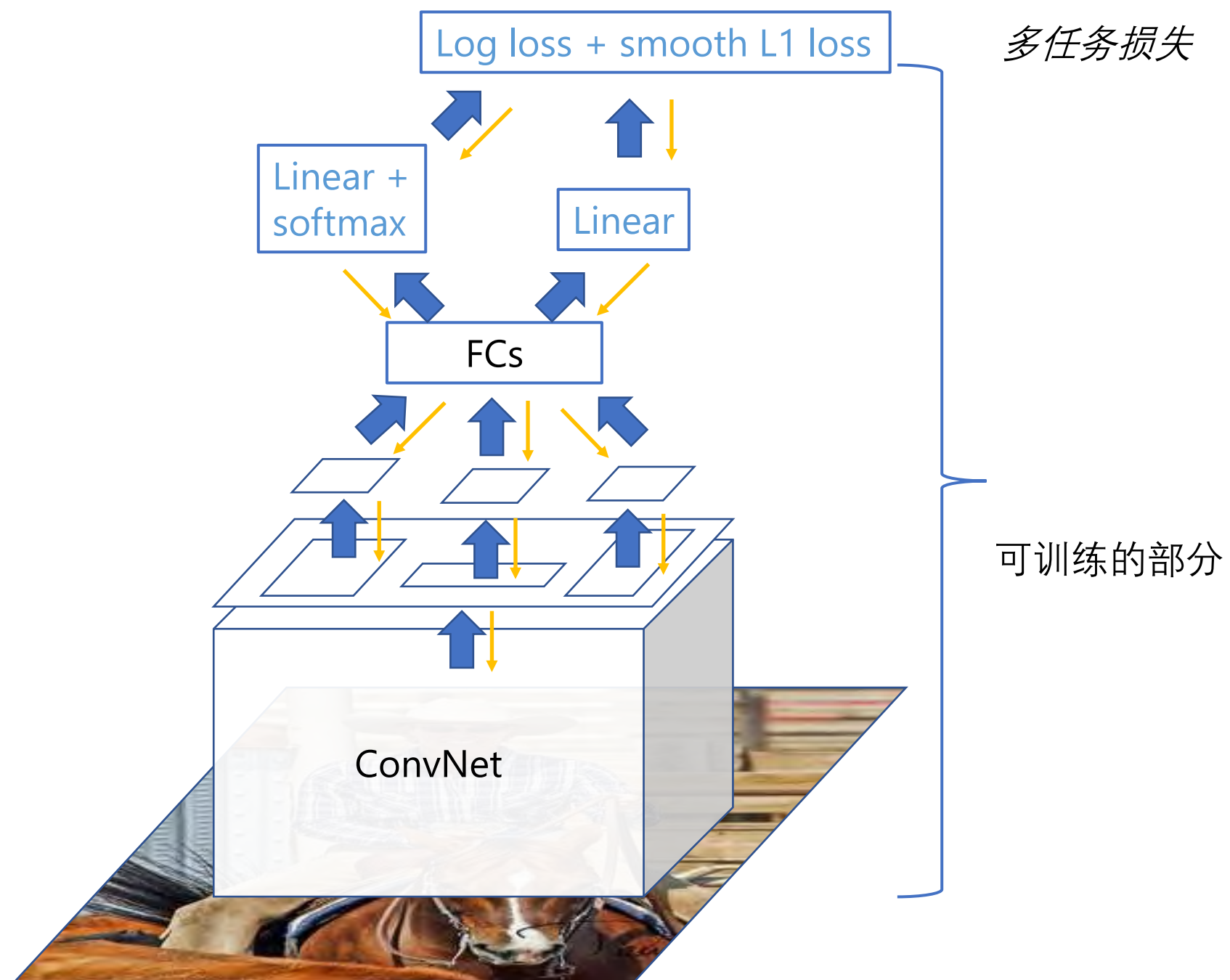
(RoI) SS需要(~2k)个框, 都要过卷积

想法: 只跑一次卷积网络, 每次提取特征基于这次网络的结果

Fast R-CNN

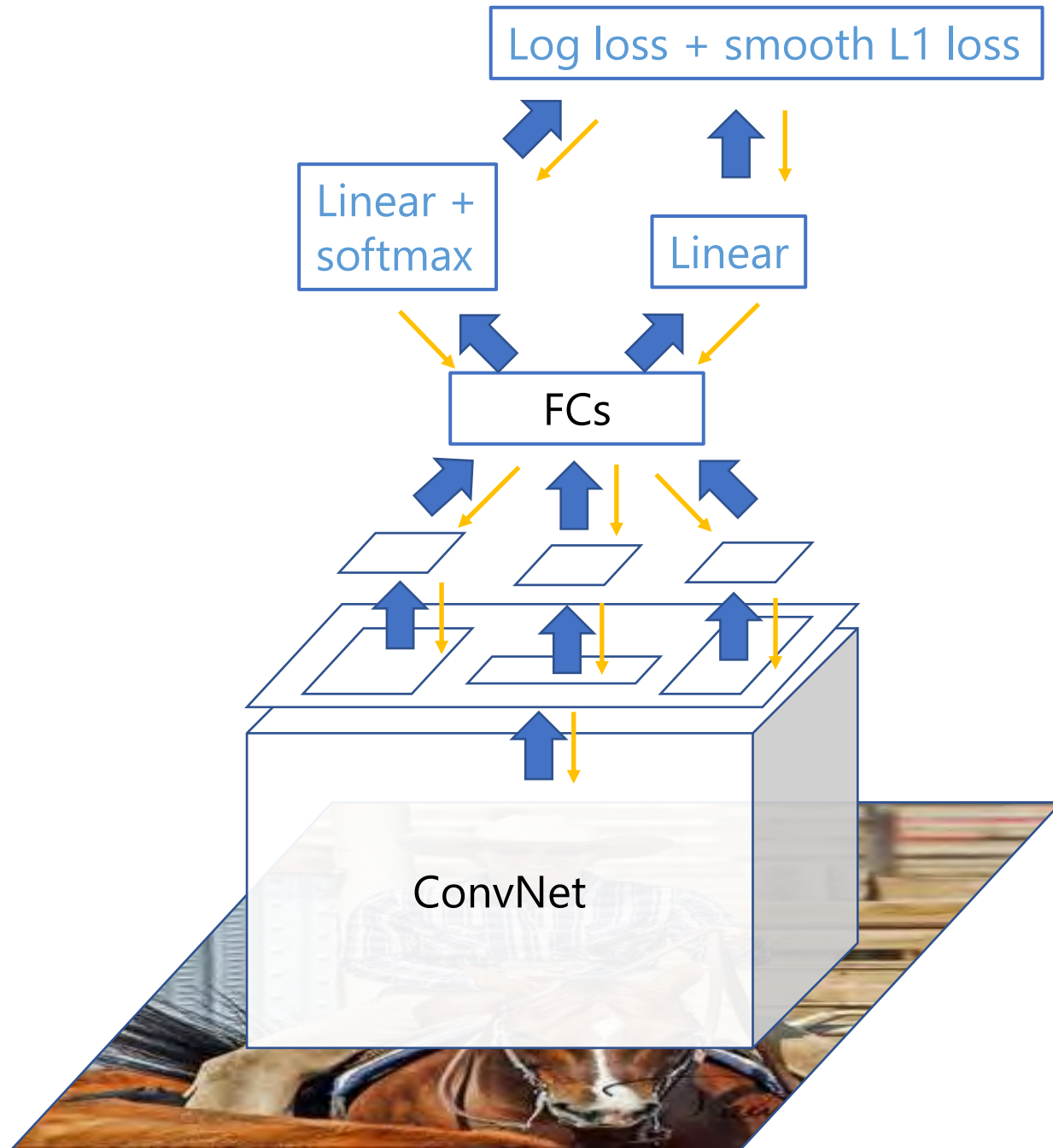


Fast R-CNN 的训练

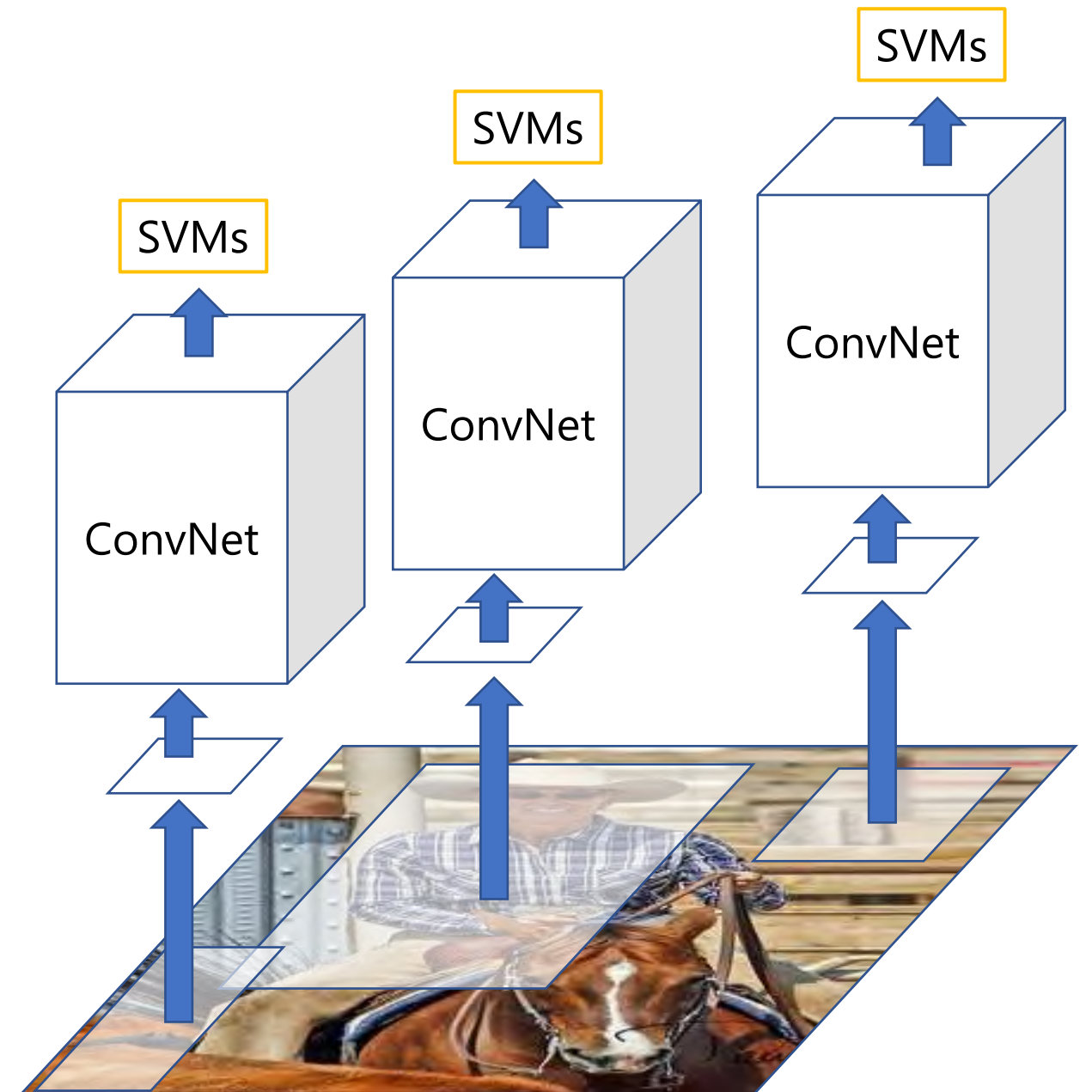


Fast R-CNN VS "Slow" R-CNN

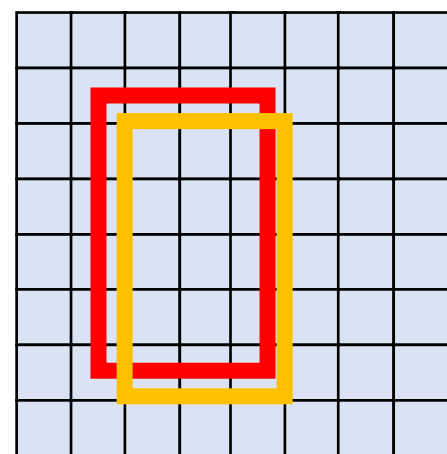
Fast R-CNN



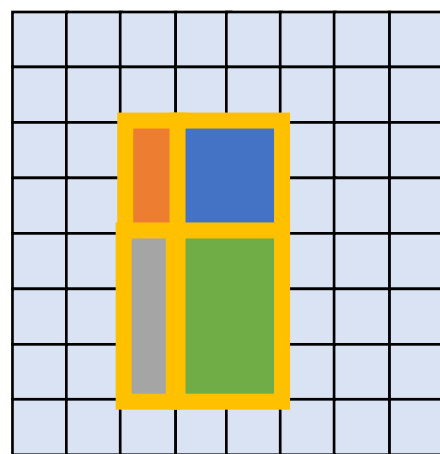
"Slow" R-CNN



Fast R-CNN – ROI-Pool



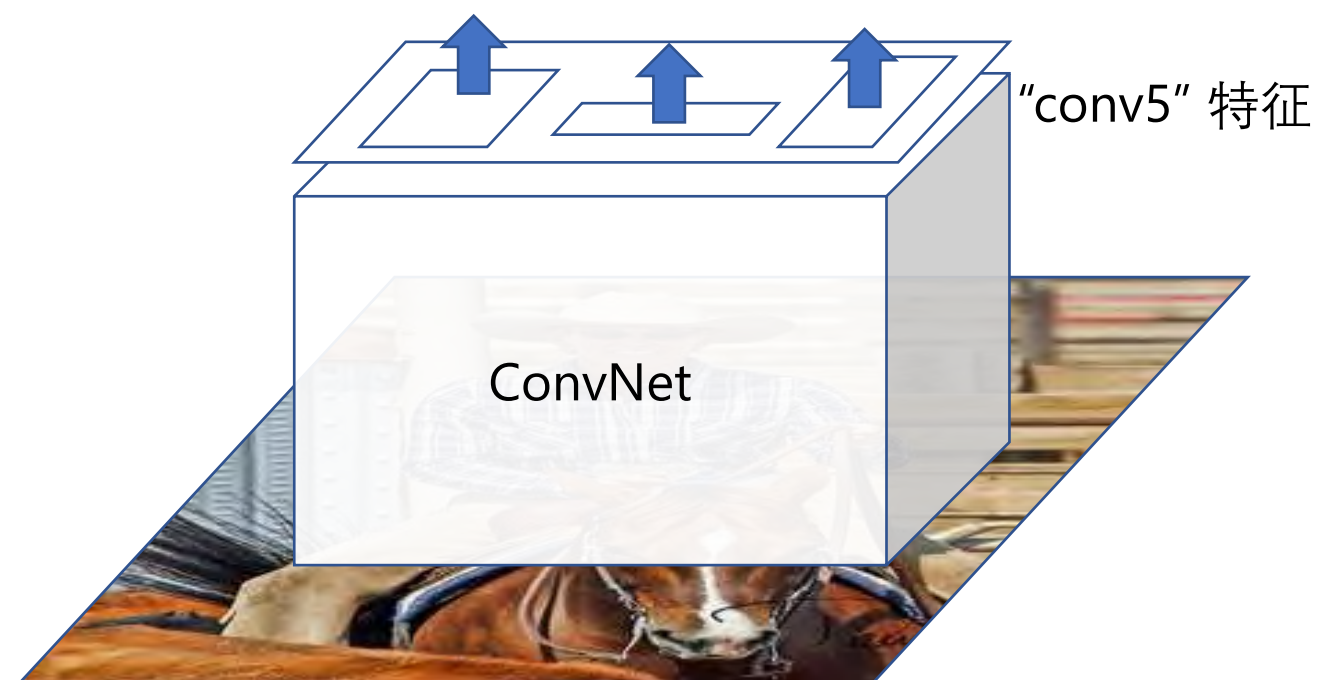
找到网格点



划分

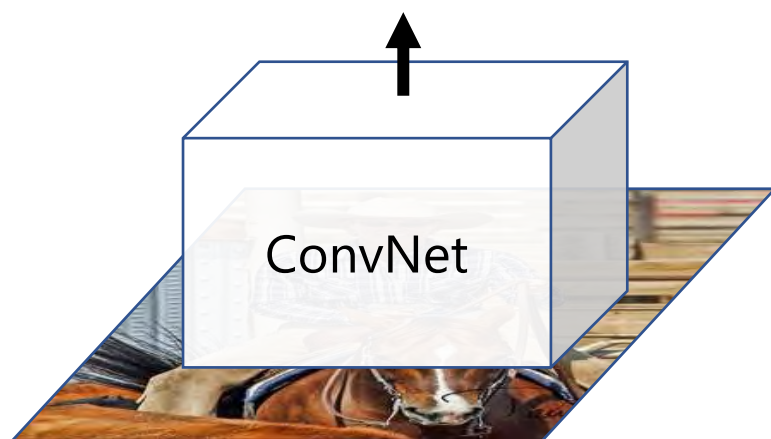
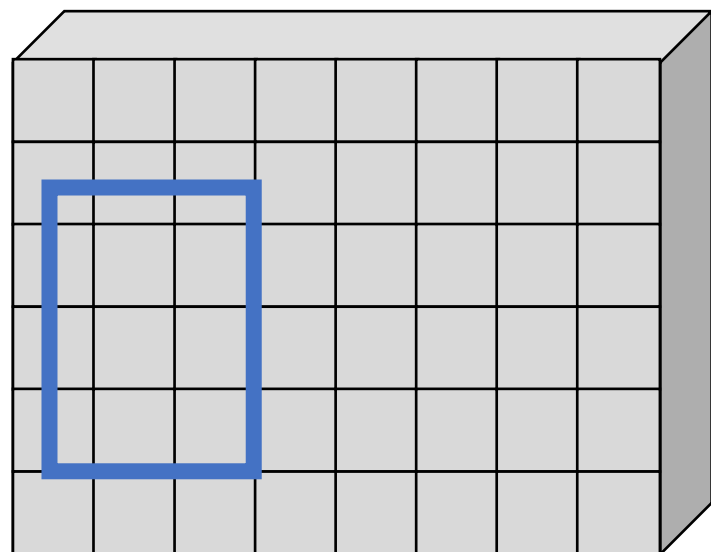


池化



ROI Pooling/Align

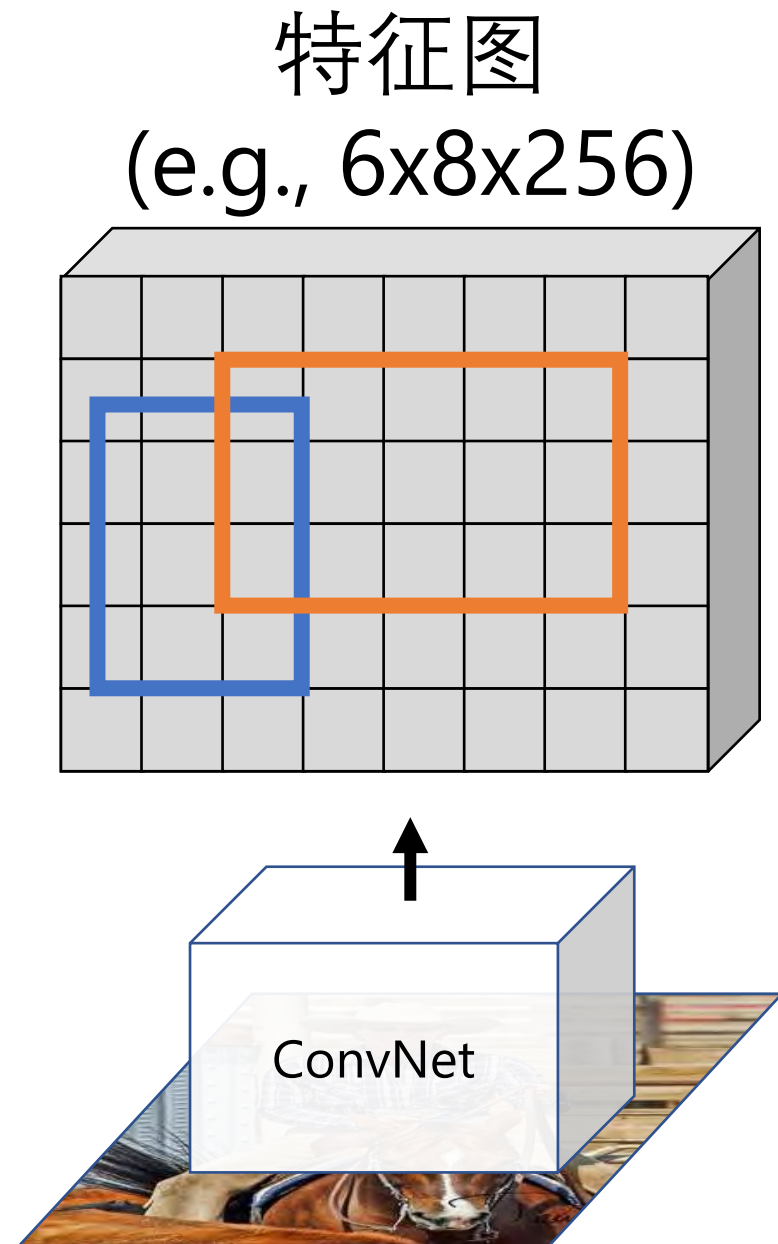
特征图
(e.g., 6x8x256)



对于原图上的边界框，找到在特征图上的位置

- 例如: $H=600, W=800$
- 特征图大小: $H'=6, W'=8$
- 边界框: left $x=50$, top $y=150$, width=250, height=350
- 特征图对应的框: left $x=0.5$, $y=1.5$, width=2.5, height=3.5

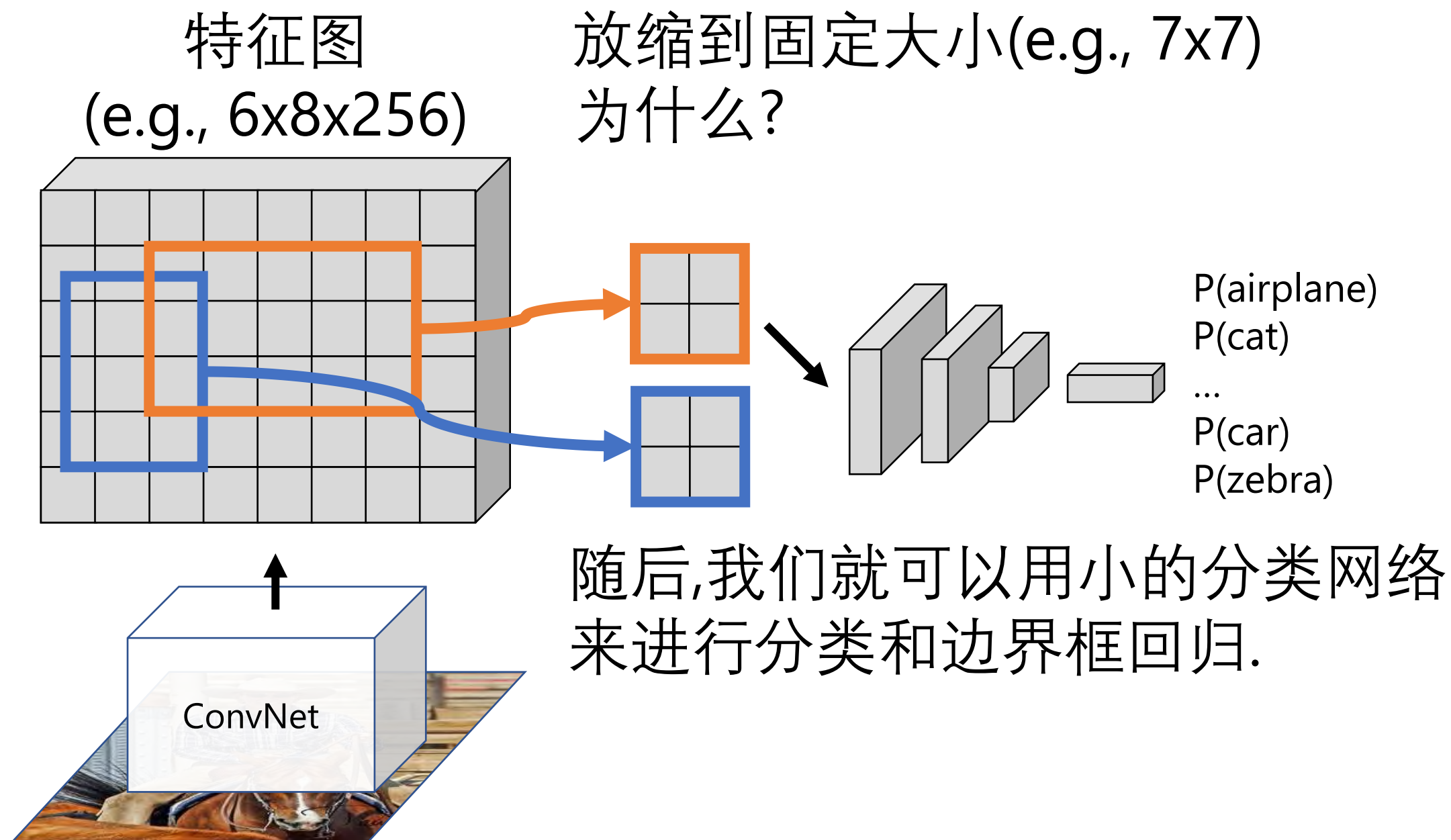
ROI Pooling/Align



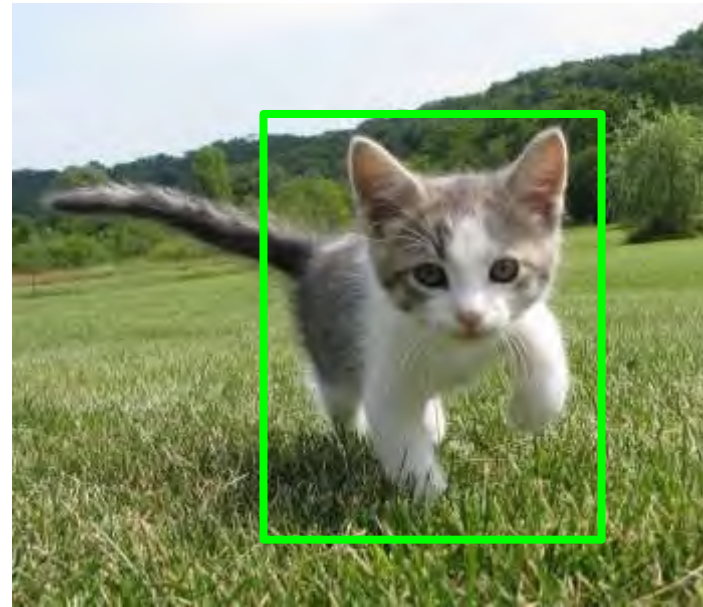
对于原图上的边界框，找到在特征图上的位置

- 例如: $H=600, W=800$
- 特征图大小: $H'=6, W'=8$
- 边界框: left $x=50$, top $y=150$, width=250, height=350
- 特征图对应的框: left $x=0.5$, $y=1.5$, width=2.5, height=3.5
- 另外一个例子: left $x=2$, top $y=1$, width=5, height=3
- 对应原图边界框是?

ROI Pooling/Align

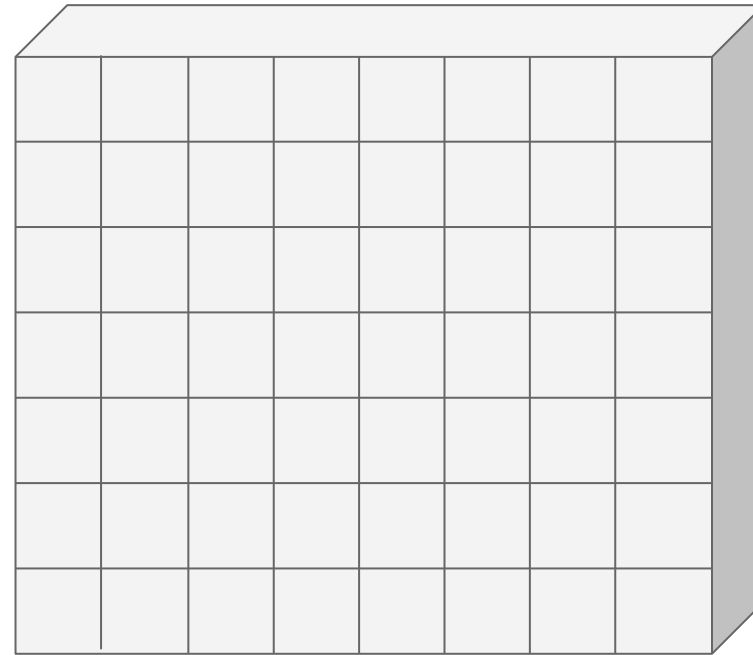


Fast R-CNN 裁剪 特征: RoI Pool



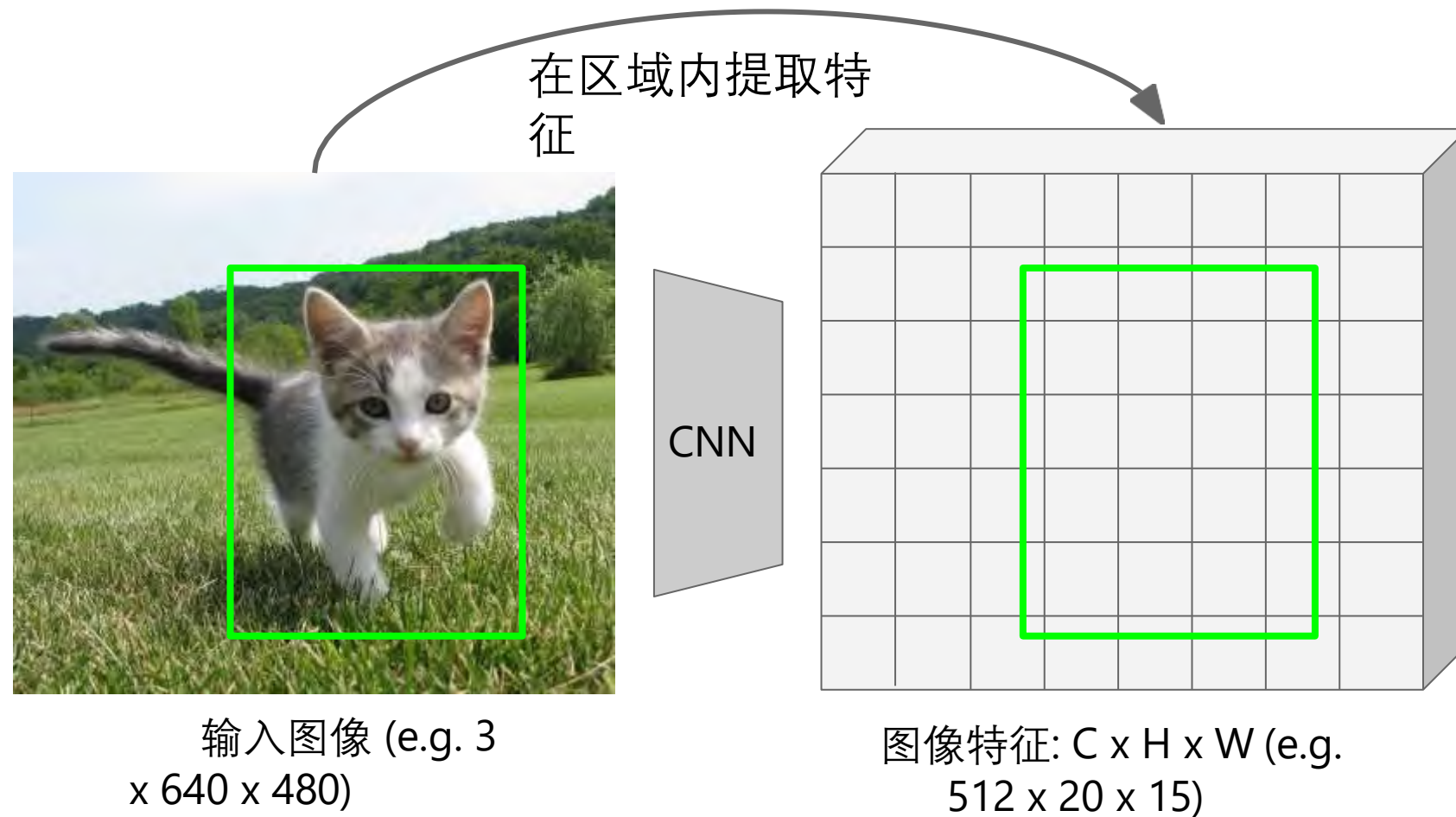
输入图像 (e.g. 3
x 640 x 480)

CNN



图像特征: C x H x W (e.g.
512 x 20 x 15)

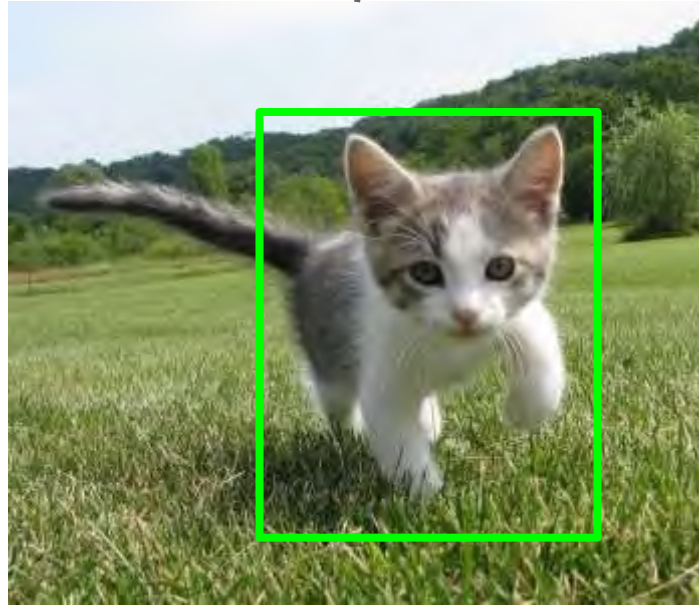
Fast R-CNN 裁剪 特征: RoI Pool



Fast R-CNN 裁剪 特征: RoI Pool

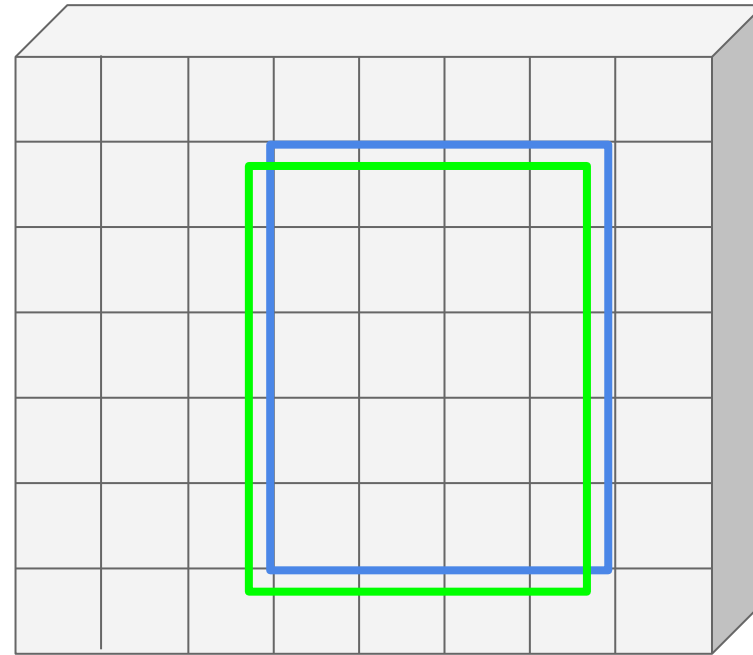
贴合网格

在区域内提取特征



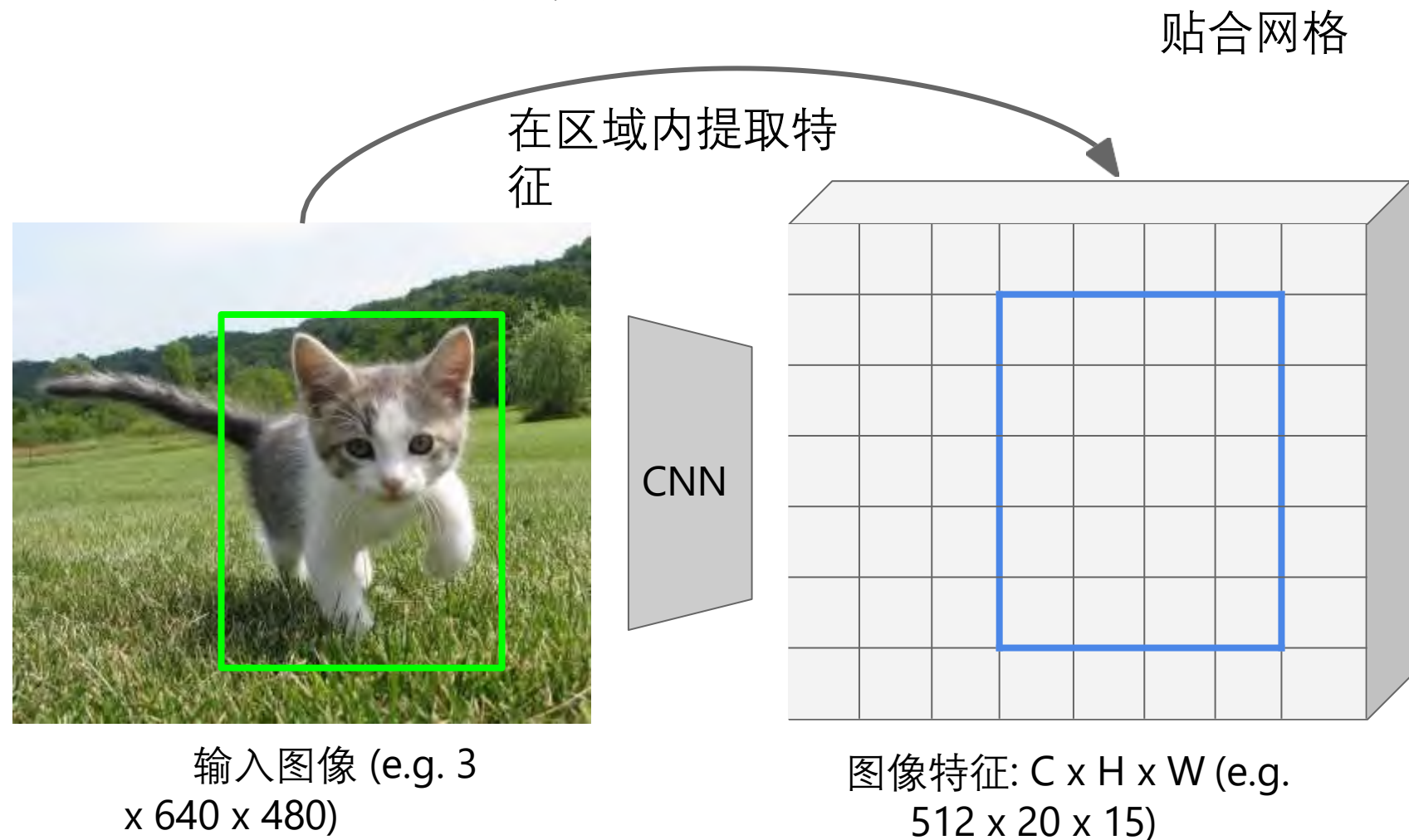
输入图像 (e.g. 3
x 640 x 480)

CNN



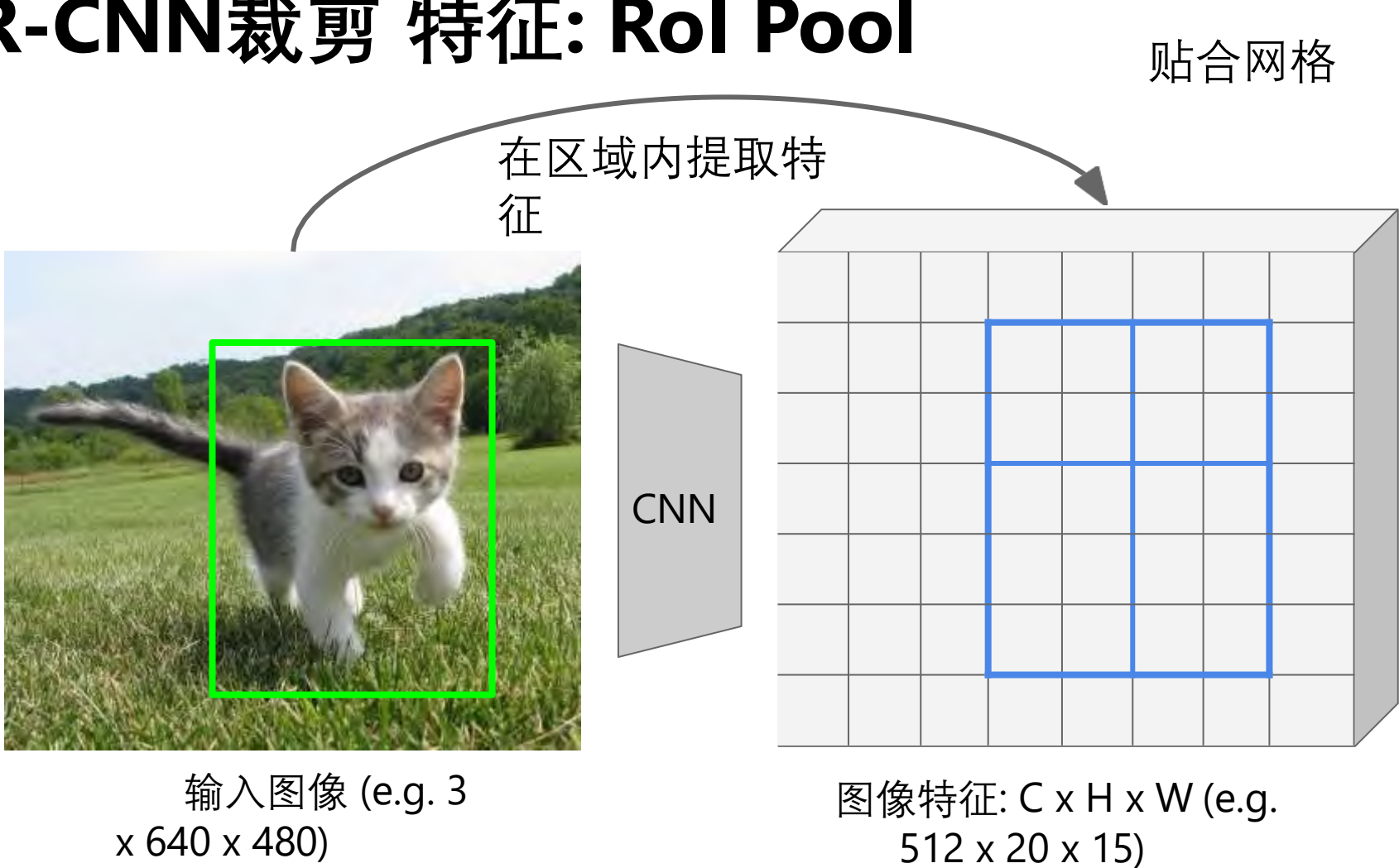
图像特征: C x H x W (e.g.
512 x 20 x 15)

Fast R-CNN 裁剪 特征: RoI Pool



怎么把 512 x 5 x 4 的区域特征 放缩到 512 x 2 x 2?

Fast R-CNN裁剪 特征: RoI Pool

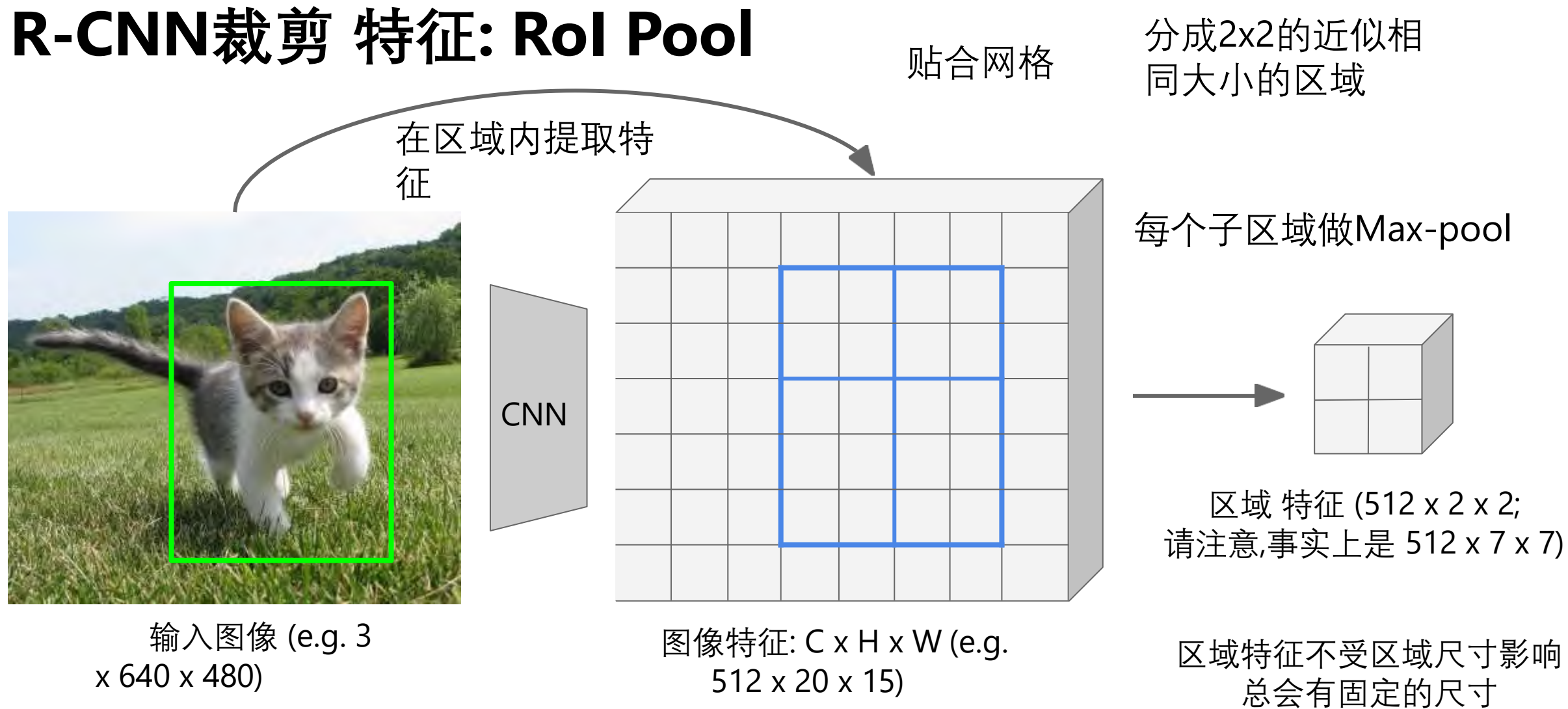


分成2x2的近似相同大小的区域

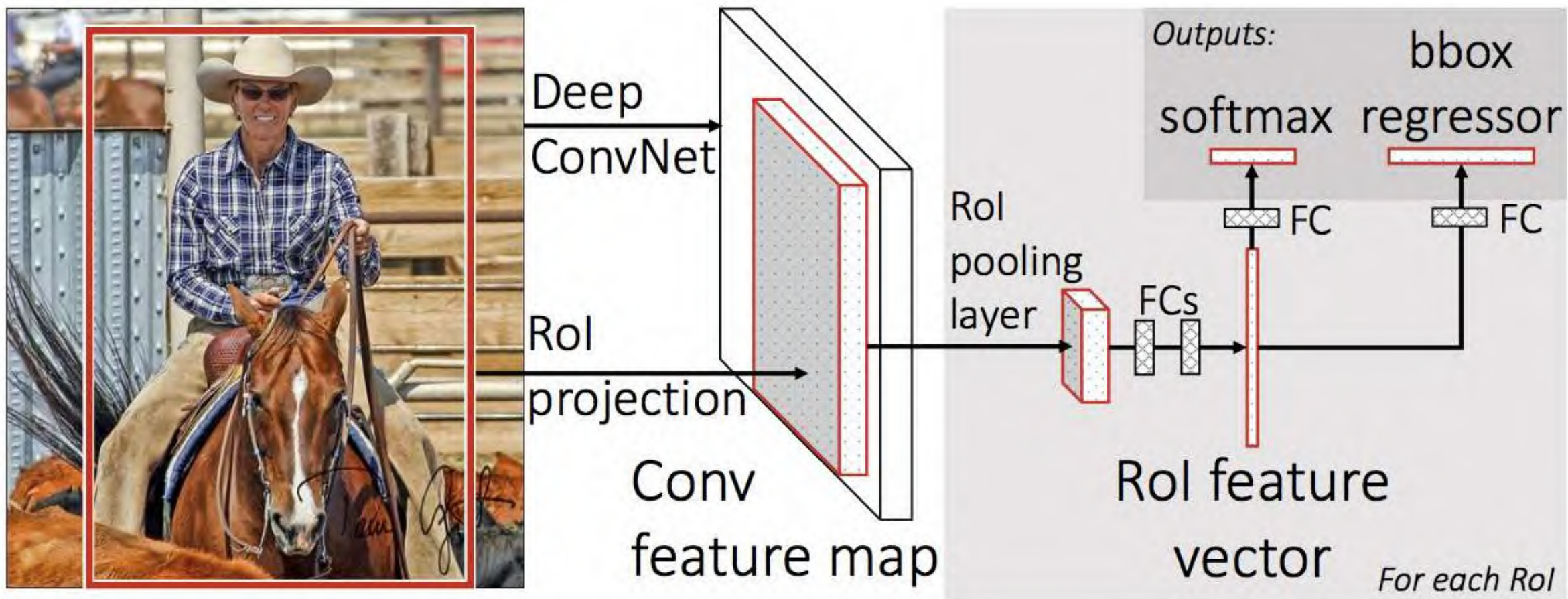
怎么把 512 x 5 x 4 的区域特征 放缩到512x 2 x 2?

Girshick, "Fast R-CNN", ICCV 2015.

Fast R-CNN裁剪 特征: RoI Pool



Fast R-CNN: 在回顾一遍



回顾——物体检测

你真的比随机数生成器更智能?

- 求概率: **1000**类分类正确?
 - $1/1,000$
- 求概率: 猜边界框的四个角, 且误差小于**10%** 图像大小?
 - $(1/10)*(1/10)*(1/10)*(1/10)=1/10,000$
- 两者概率叠加: $1/10,000,000$
- 不能用最常出现的标签 (没有物体) 来直接预测

性能评估

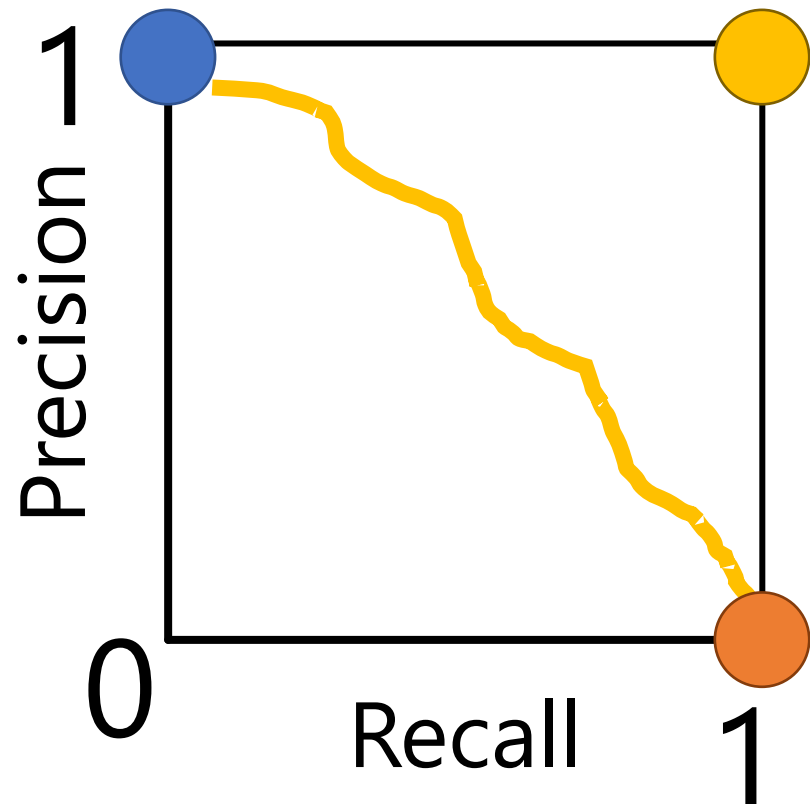
- 准确率 = 预测正确的概率
- 如果我声称我的行人检测99%正确.
- **这代表我的检测做的很好吗?**
- 大街上没有多少人!



性能评估

- True detection 真实检测 (true positive) : 高 IoU
- Precision 精确度: $\# \text{真实检测} / \# \text{检测总数}$
- Recall 召回率: $\# \text{真实检测} / \# \text{真实的物体数量}$

如果拒绝一切，那么就不会有错误



理想情况是精确度和召回率都很高!

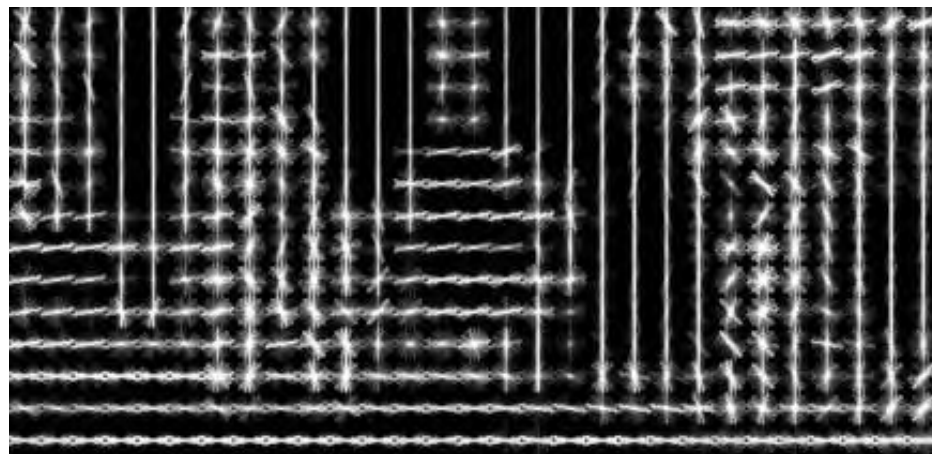
AUC area under curve (avg. precision)

如果接受一切，那么就不会错过任何东西

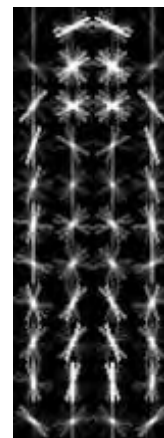
使用HOG进行行人检测

- 用线性SVM训练行人“模板”
- 测试时对HOG特征图用模板做卷积
- 找到局部最大响应值
- 不要忘了应对多尺度： HOG *pyramid* 金字塔

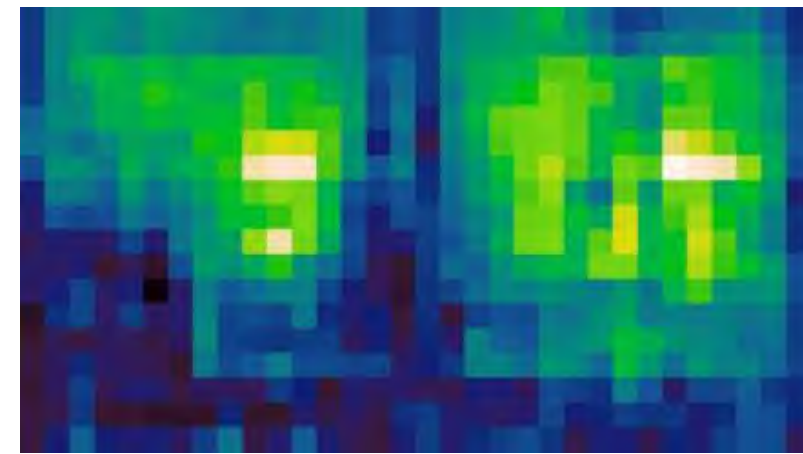
HOG 特征图



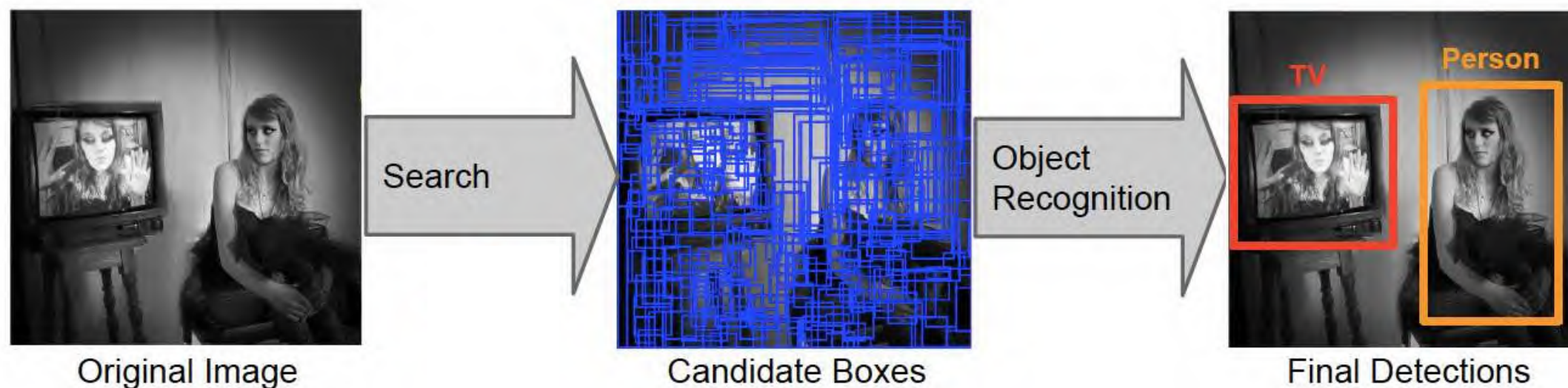
模板



检测器响应图



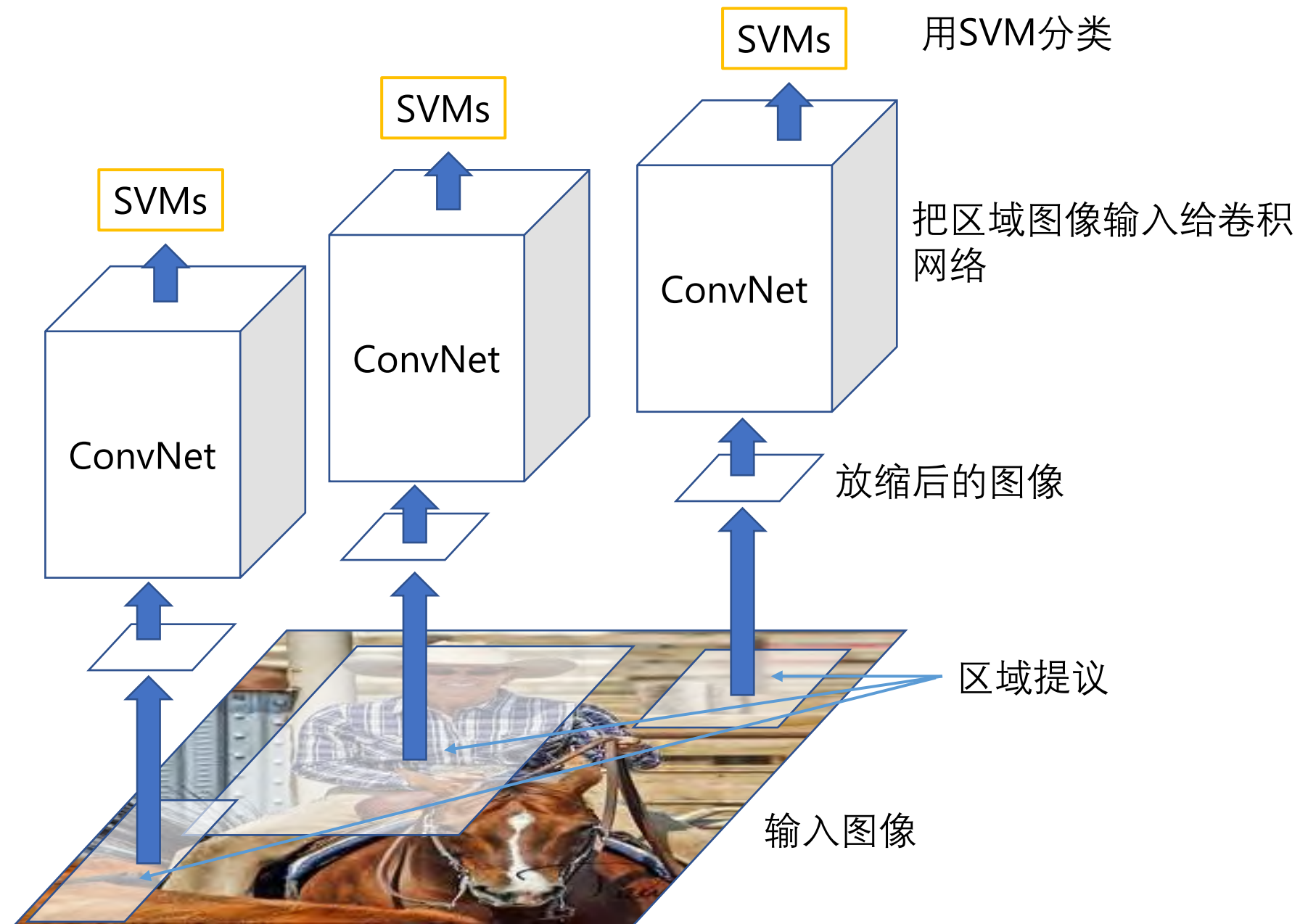
Region Proposals 区域提议



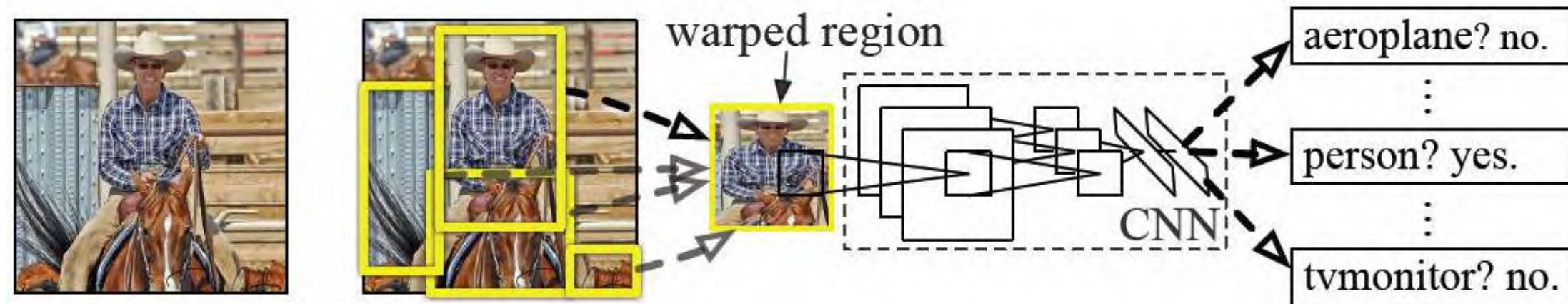
- 替代滑动窗口，只选出部分候选框作为 *region proposals* 区域提议
 - 谨记：特征提取和分类器速度较慢，区域提议可以减少候选区域，提升运行效率
 - 一般候选区域类别无关
 - 这个机制可以被训练（后面会讲）

R-CNN: Region proposals + CNN features

Source: R. Girshick



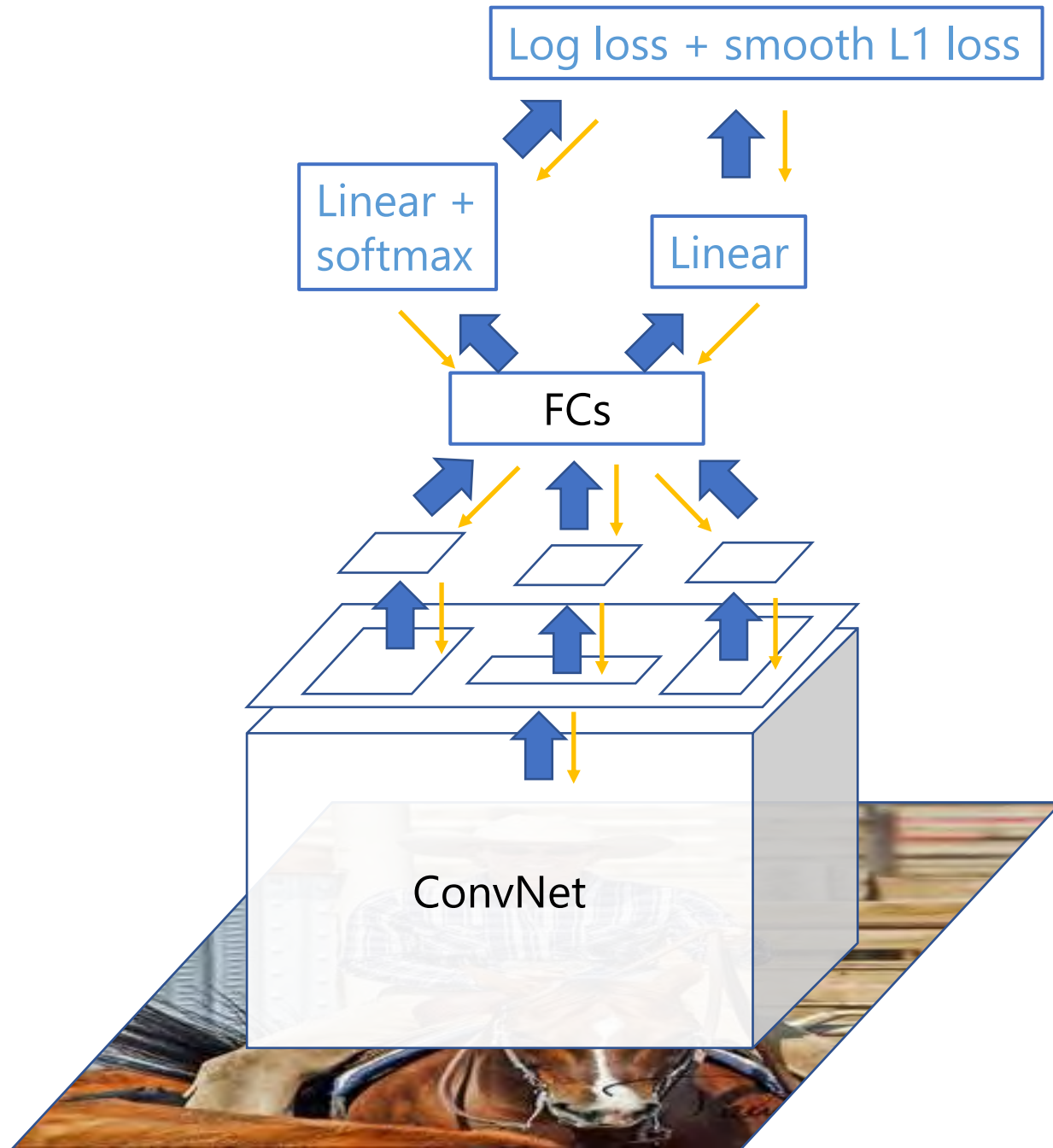
R-CNN



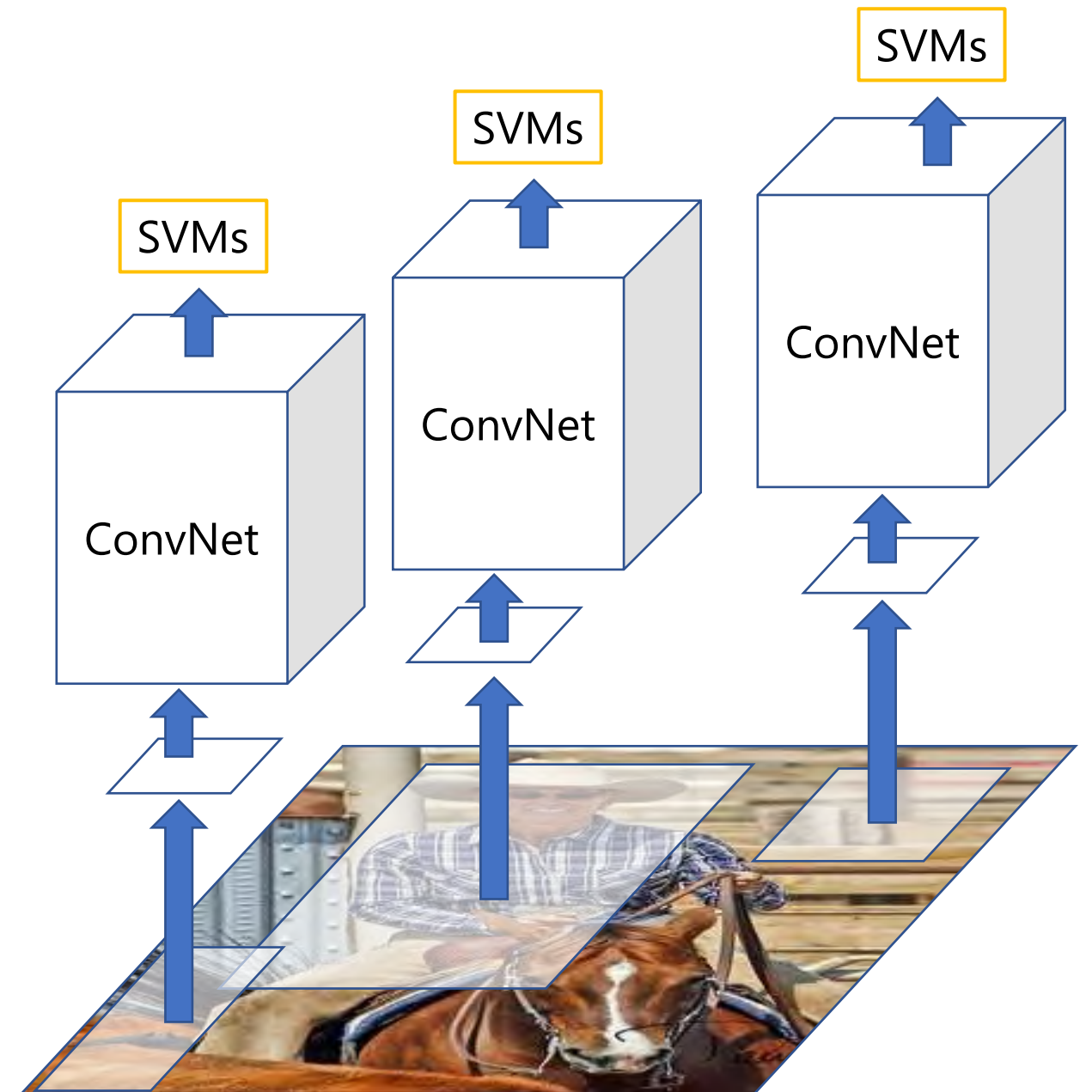
- 候选区域: ~2000 Selective Search 候选区域
- 网络: AlexNet 预训练模型, ImageNet上预训练(1000类), PASCAL 上微调 (21类: 20+背景)
- 检测器: 放缩图像输入卷积网络, 使用 fc7 层特征 (4096维), 使用SVM分类
- 同时使用SVM回归候选框
- **性能:** mAP of 53.7% on PASCAL 2010 (vs. 35.1% for Selective Search and 33.4% for DPM).

Fast R-CNN VS "Slow" R-CNN

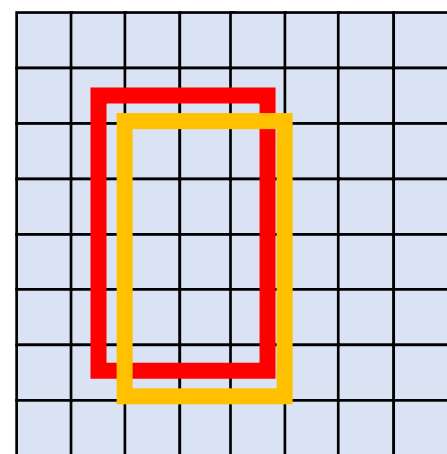
Fast R-CNN



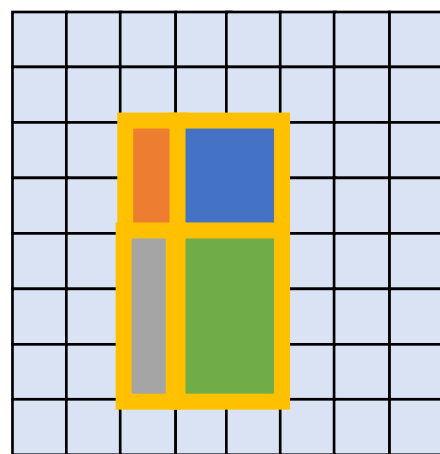
"Slow" R-CNN



Fast R-CNN – ROI-Pool



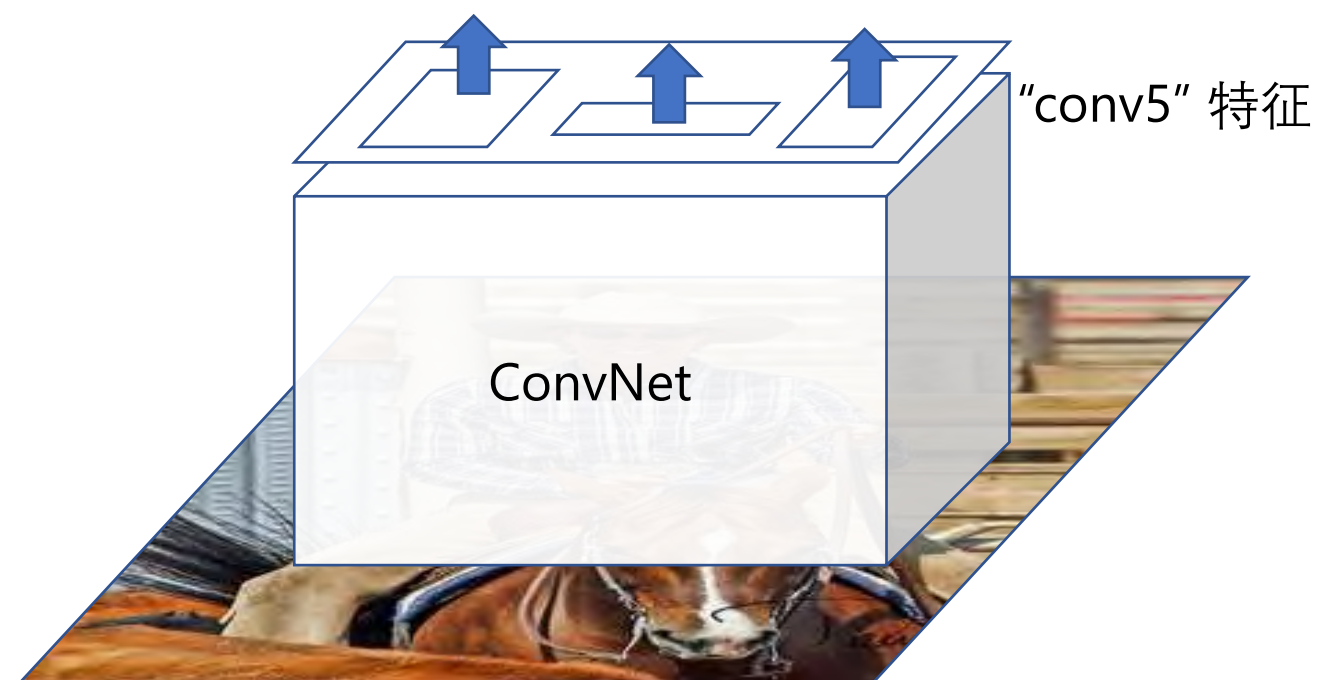
找到网格点



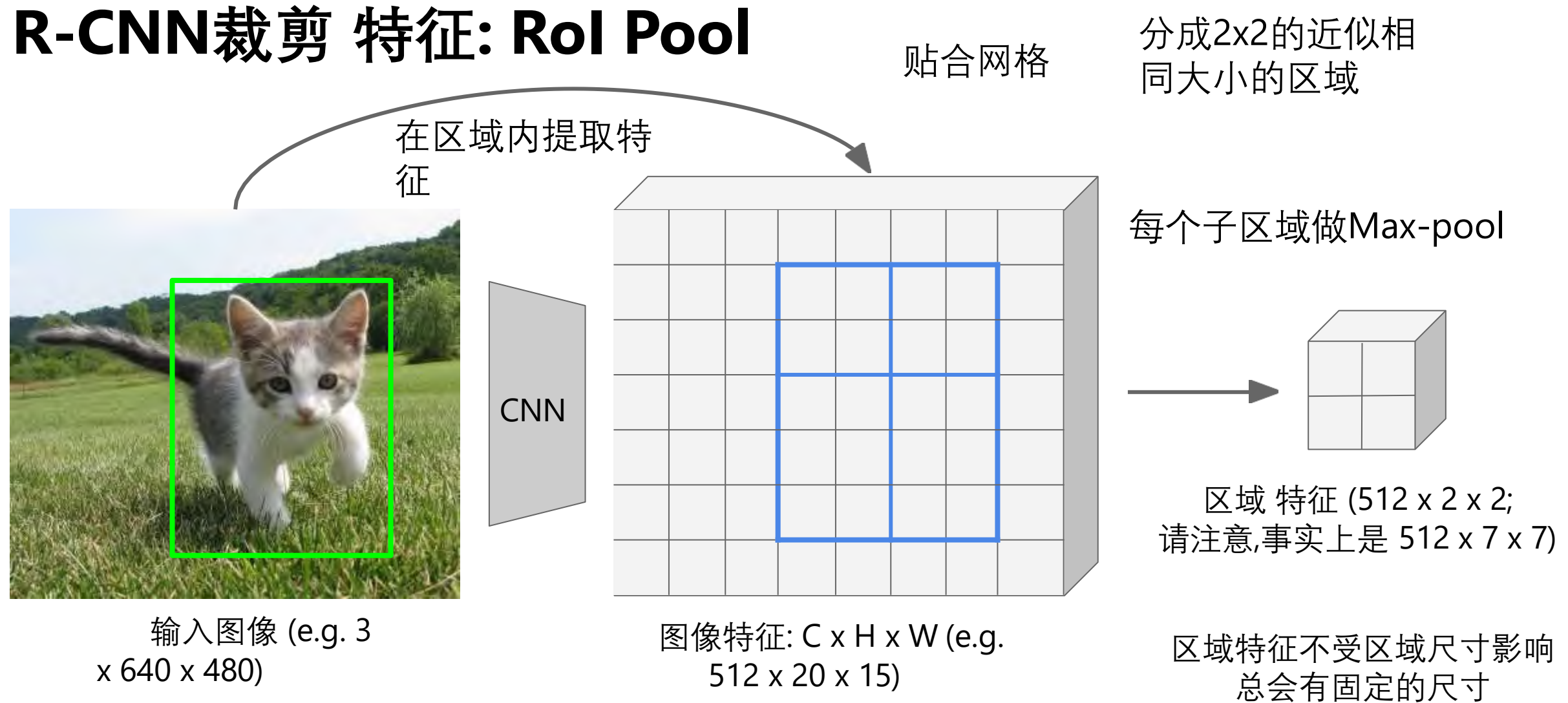
划分



池化



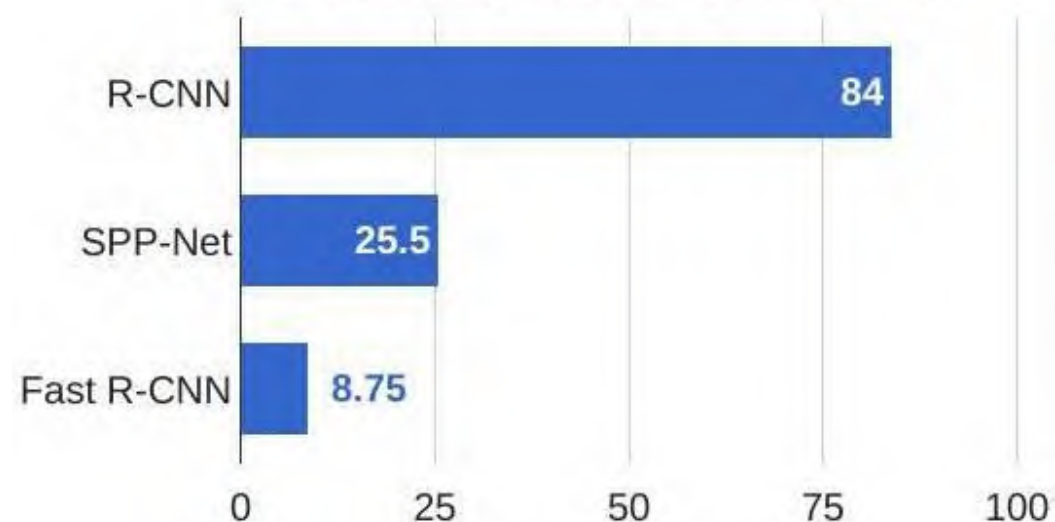
Fast R-CNN裁剪 特征: RoI Pool



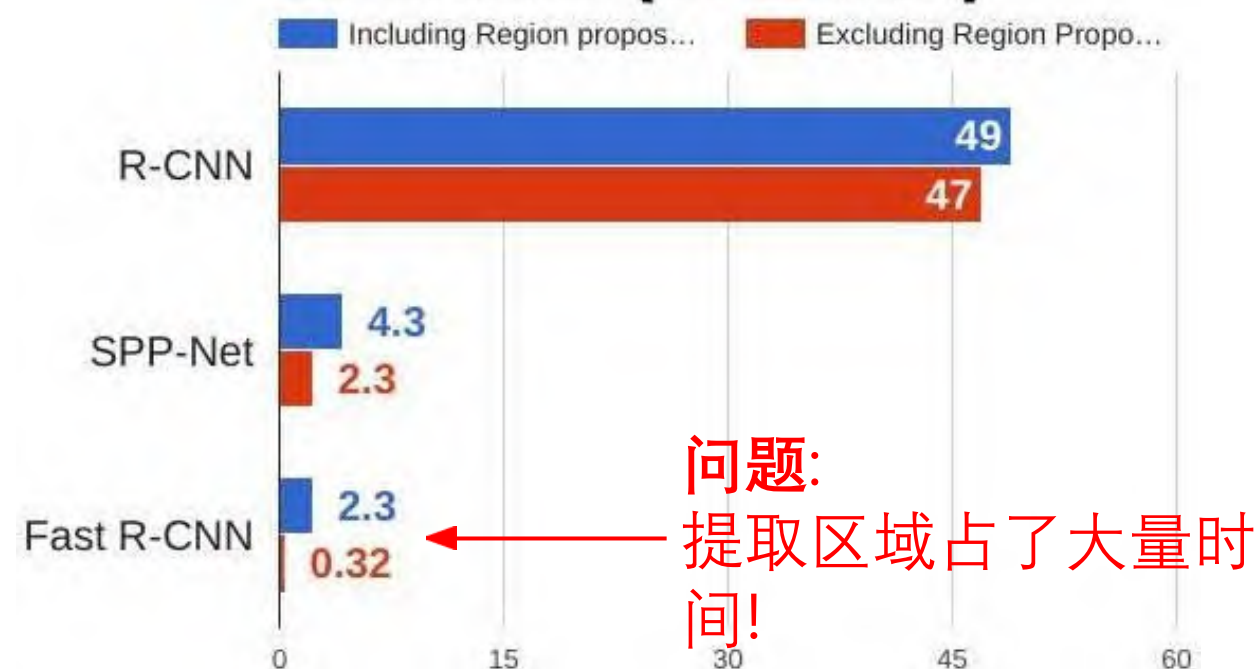
Fast R-CNN 性能

	Fast R-CNN	R-CNN
Train time (h)	9.5	84
- Speedup	8.8x	1x
Test time / image	0.32s	47.0s
Test speedup	146x	1x
mAP	66.9%	66.0%

Training time (Hours)

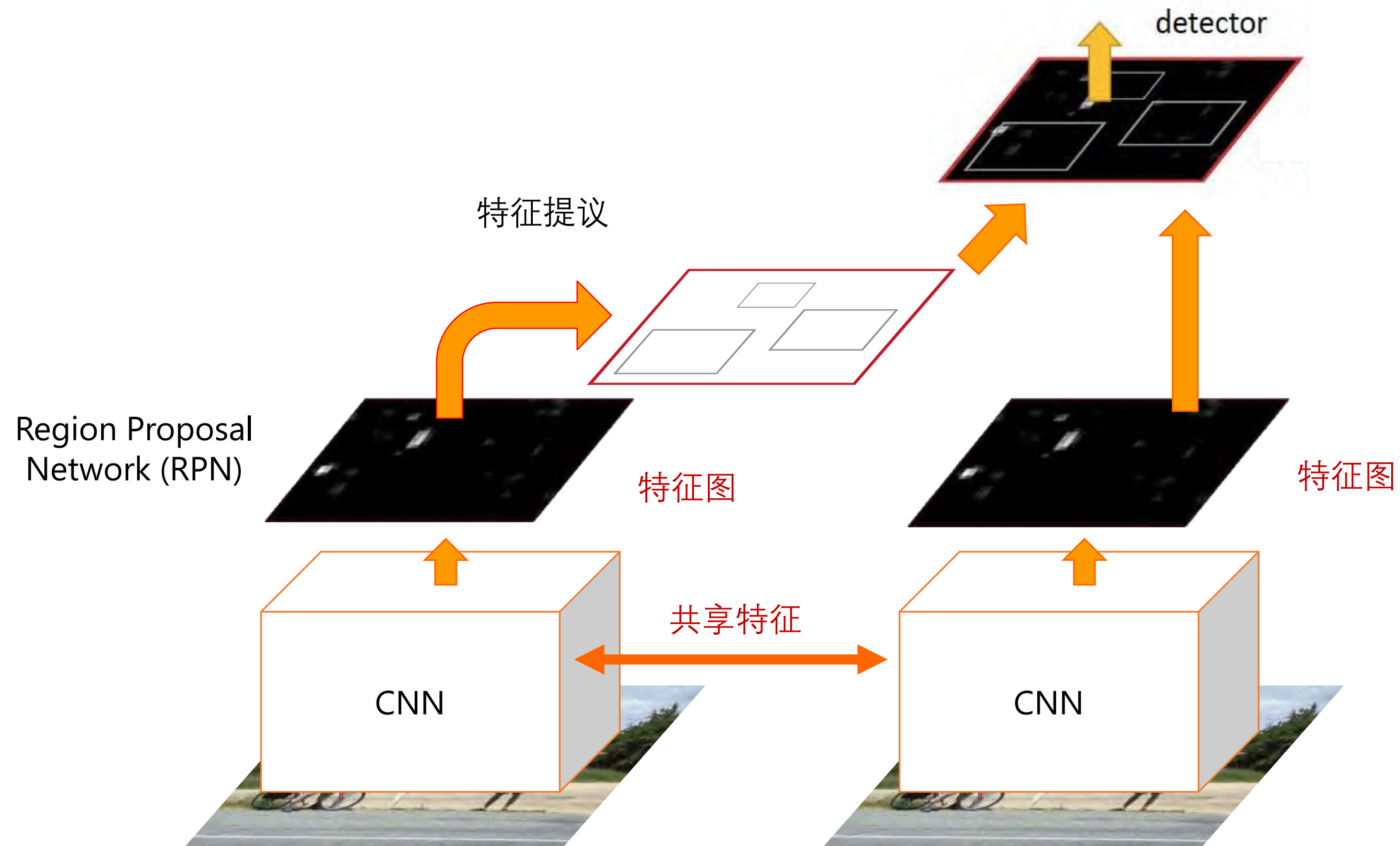


Test time (seconds)



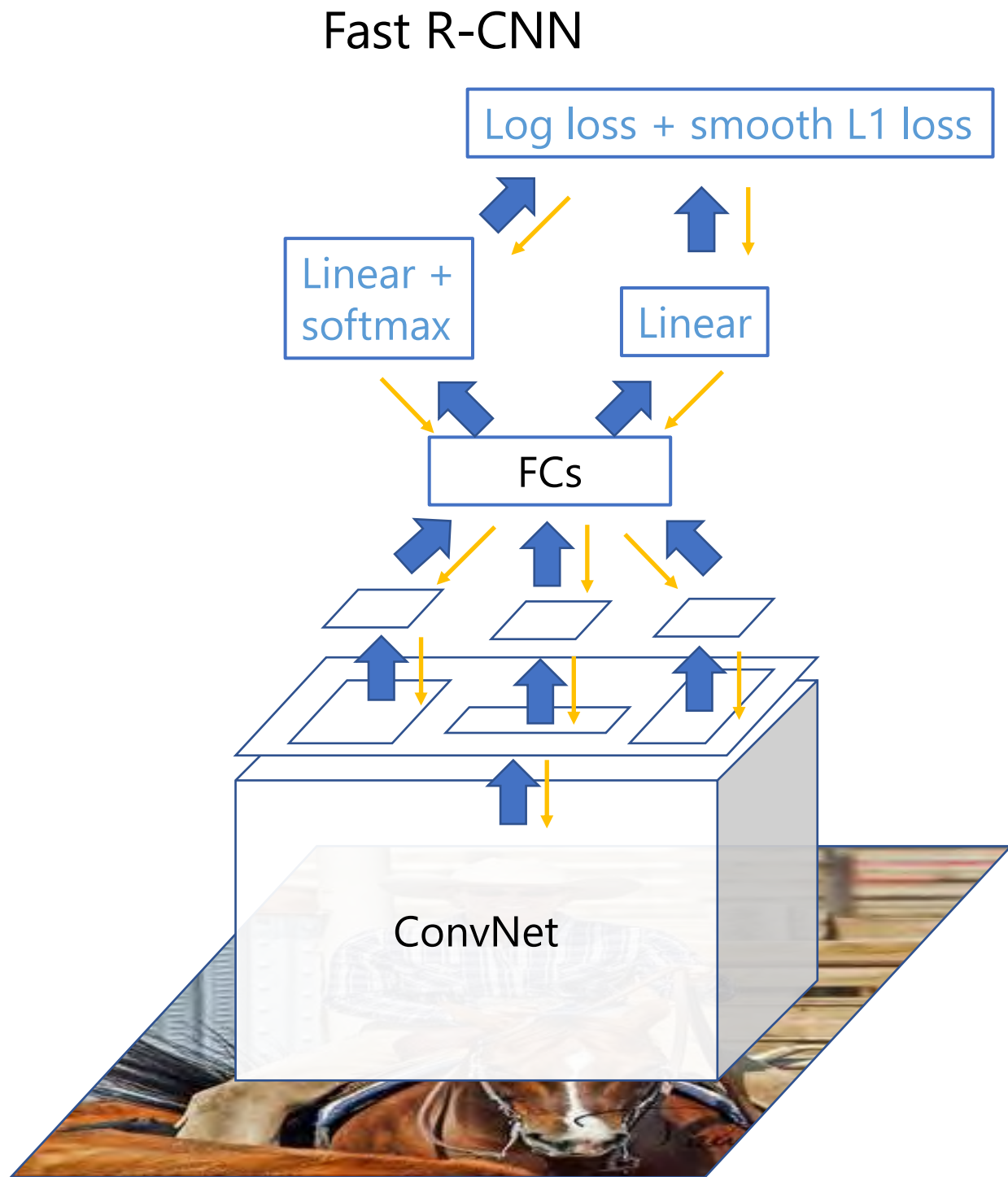
左边的表里没有计算区域提议的时间.
使用 VGG16 网络

Faster R-CNN

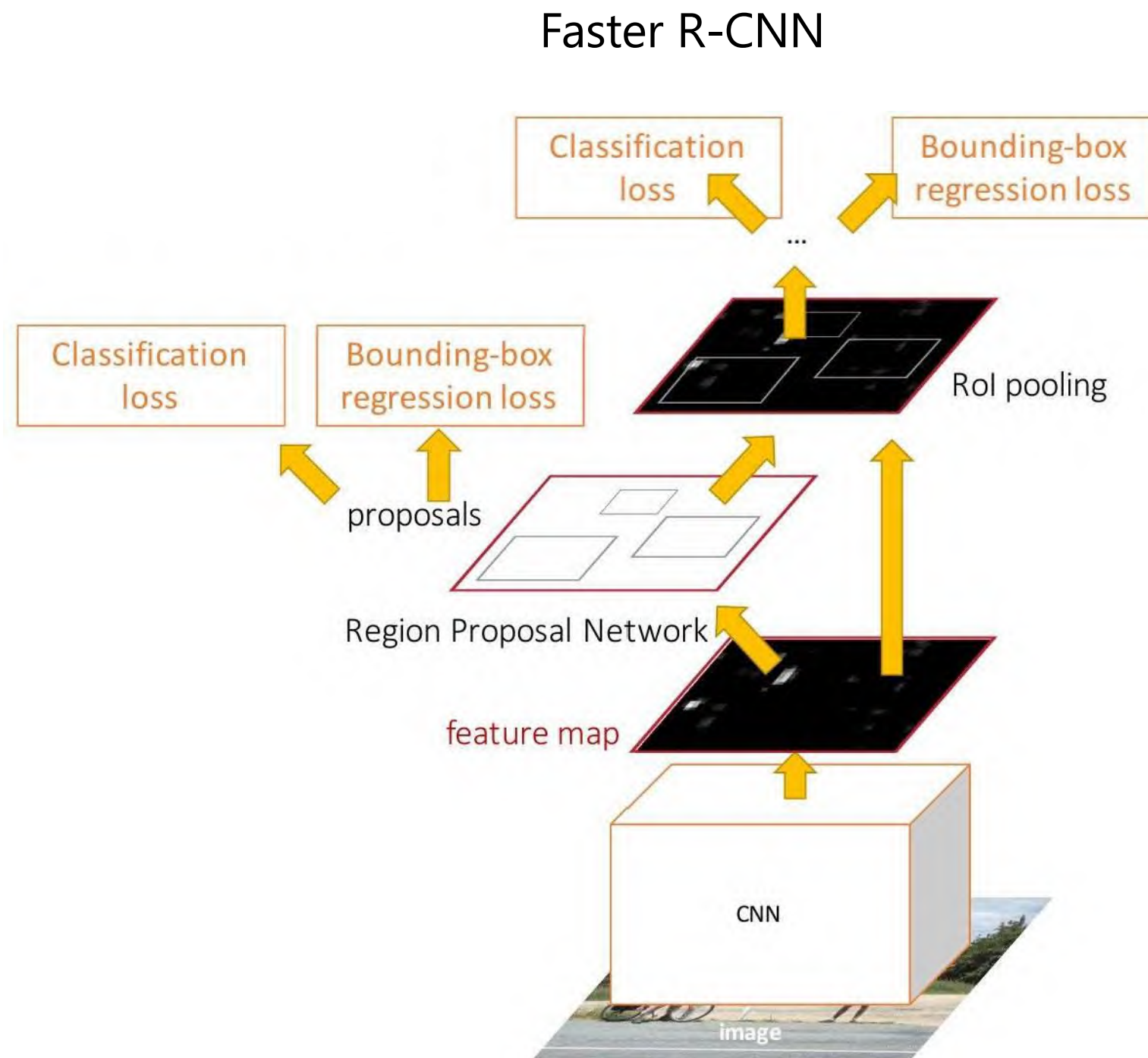


S. Ren, K. He, R. Girshick, and J. Sun, [Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks](#), NIPS 2015

Faster R-CNN: 让CNN提Proposals!

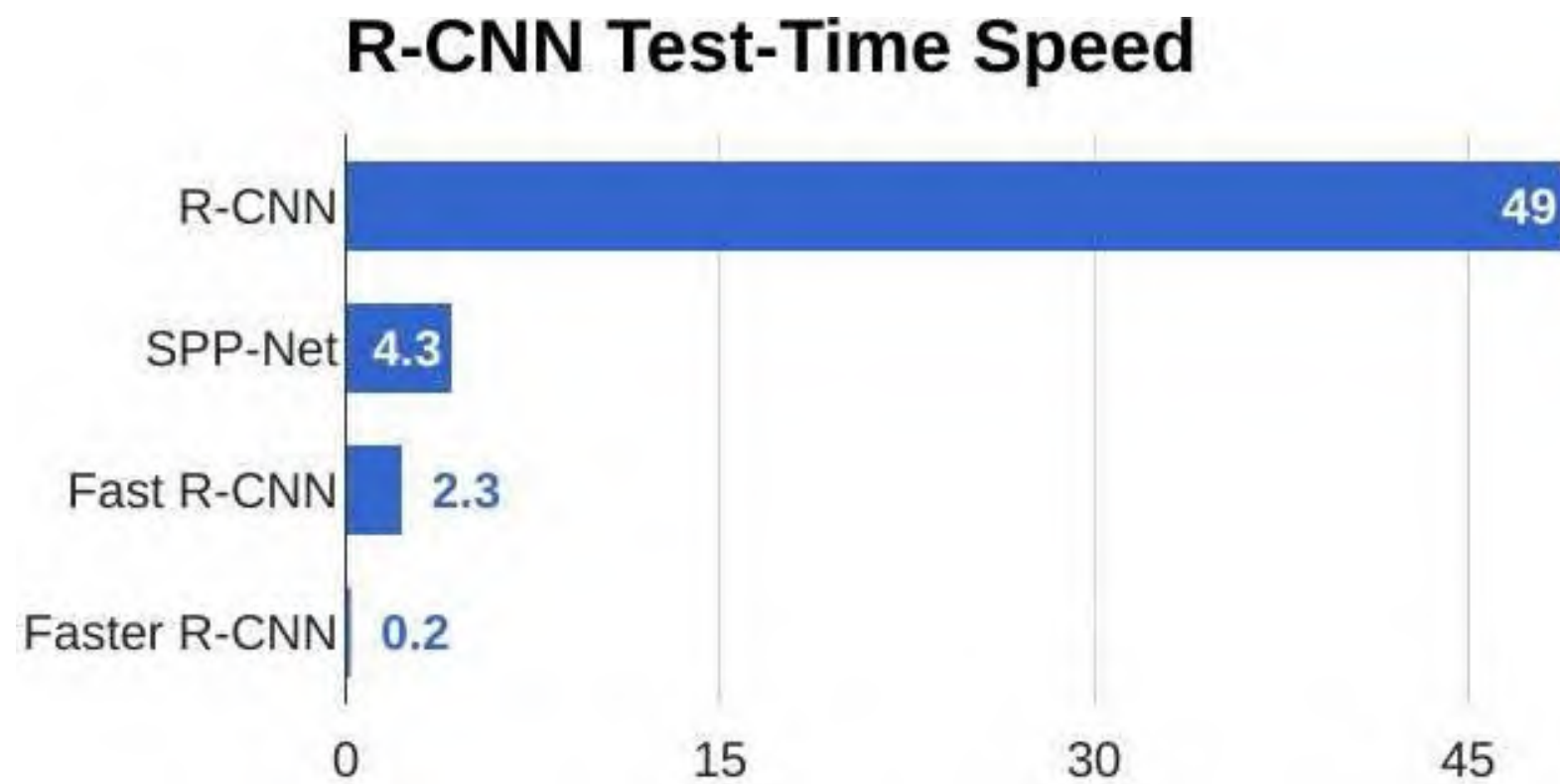


Region Proposal Network (RPN) 预测区域proposal 其余的与Fast R-CNN一致

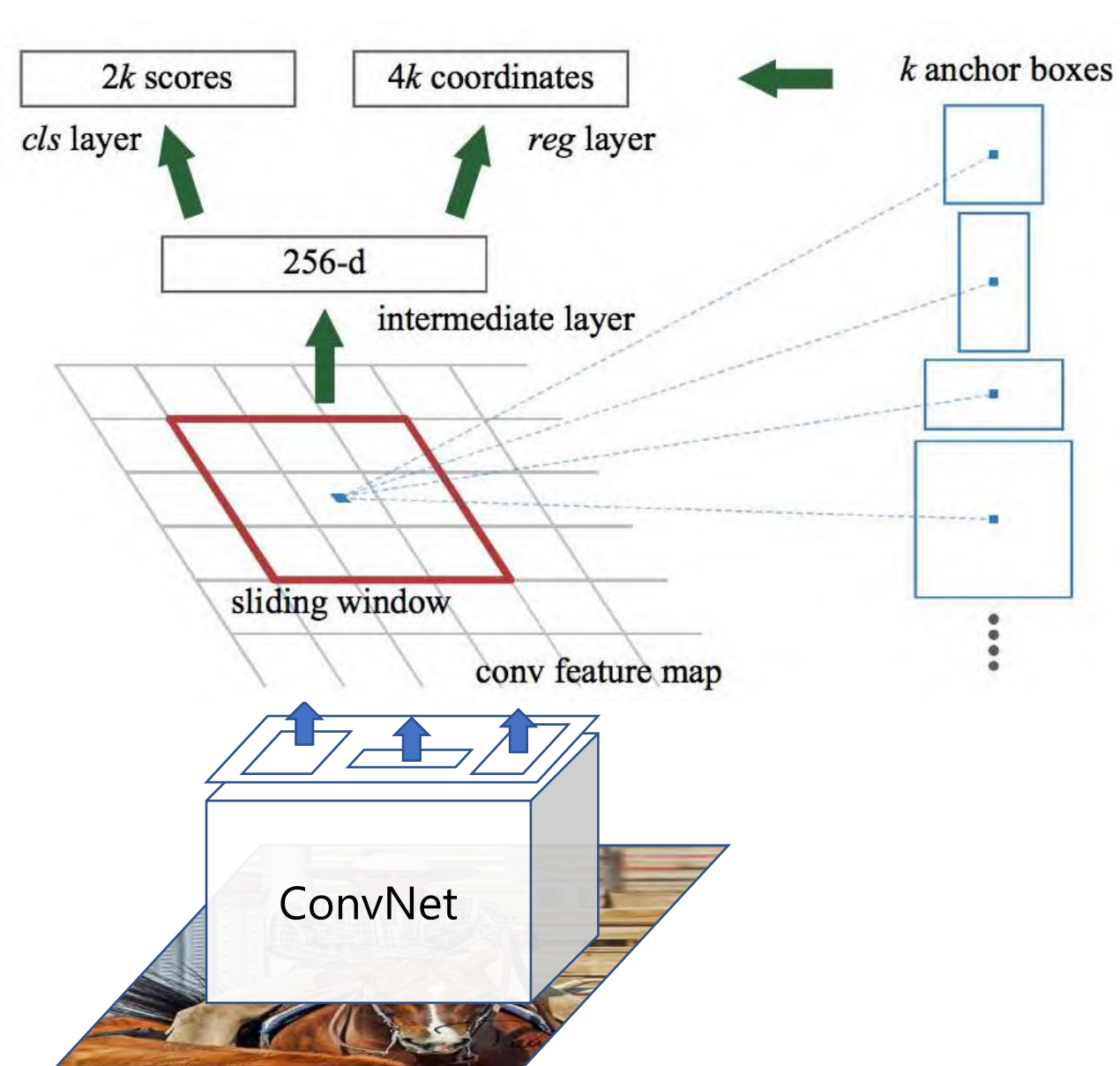


Faster R-CNN:

让CNN提Proposals!



Region Proposal Network (RPN) 区域提议网络



基于conv5 特征的小网络模块.

预测:

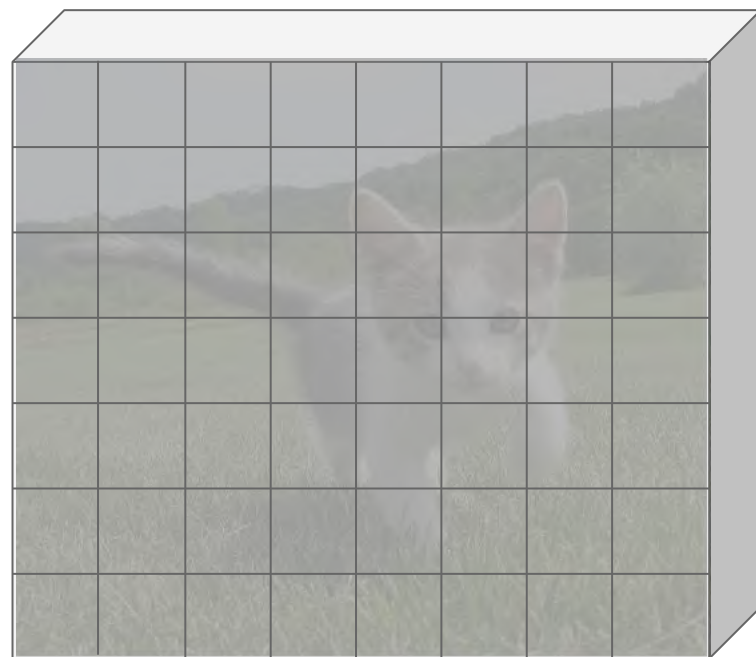
- 是否包括物体 (classification),
 - $IoU > 0.7$: 包含物体
 - $IoU < 0.3$: 不包含
 - 其他: 忽略
 - 区域是否需要校正 (regression)
- k 个根据特征图提取的 "anchors".

Region Proposal Network



输入图像 (e.g. 3
x 640 x 480)

CNN



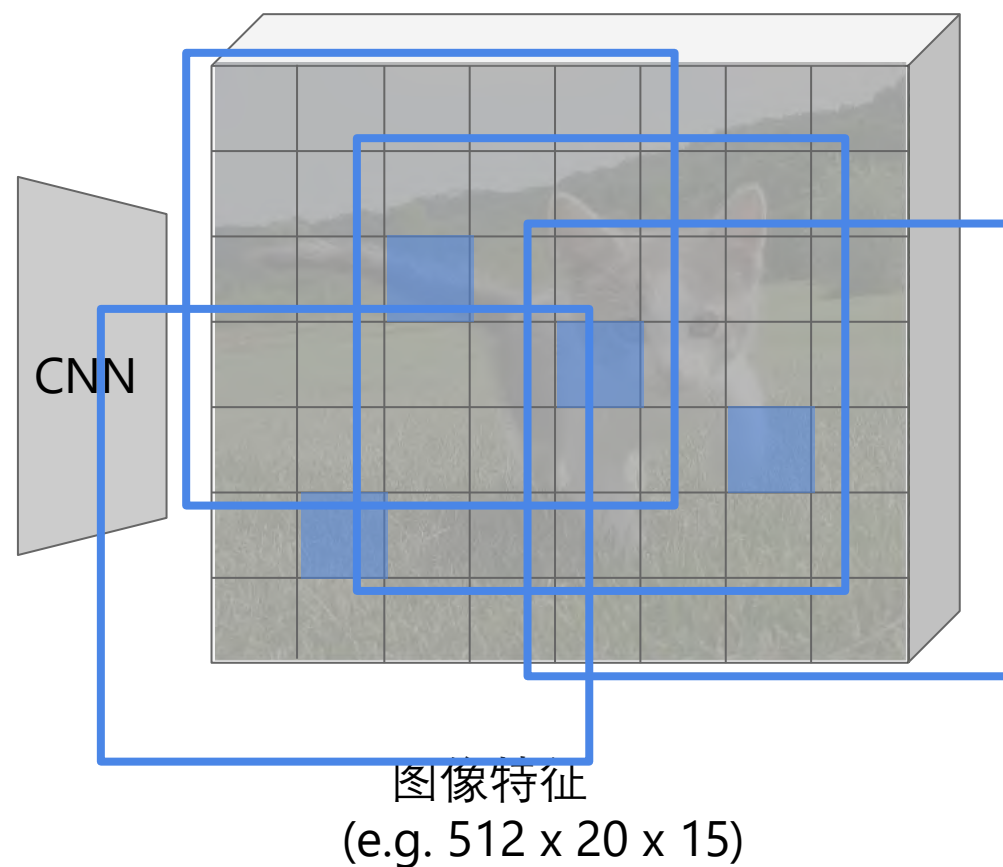
图像特征 (e.g.
512 x 20 x 15)

Region Proposal Network

想象一个有固定尺寸
anchor框



输入图像
(e.g. 3 x 640 x 480)

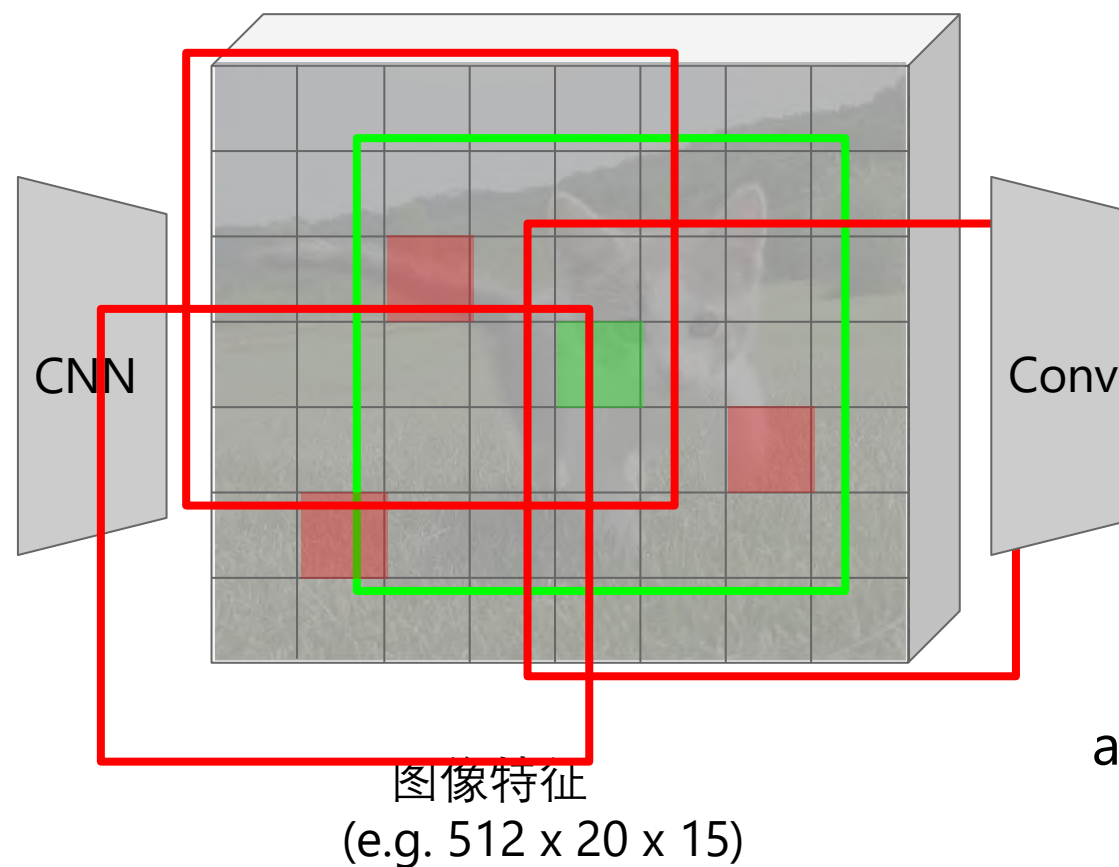


图像特征
(e.g. 512 x 20 x 15)

Region Proposal Network



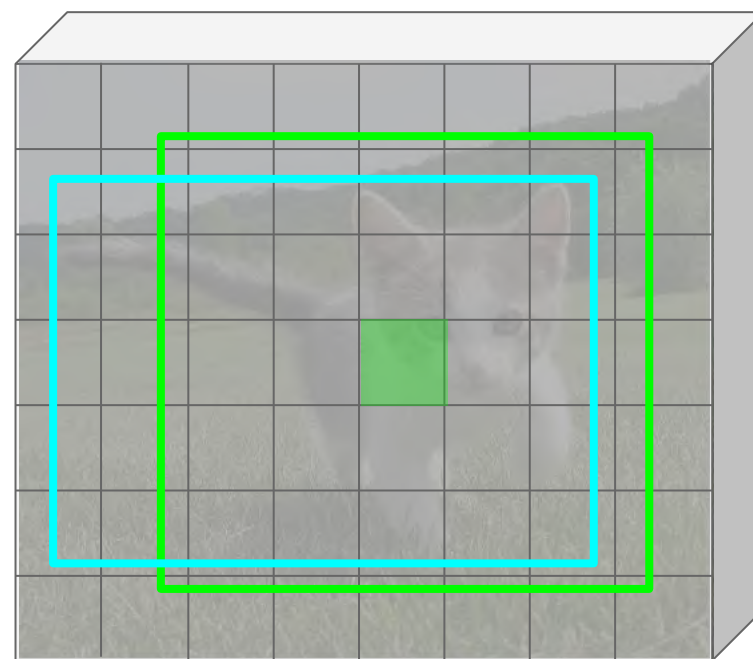
输入图像
(e.g. 3 x 640 x 480)



Region Proposal Network



CNN



图像特征
(e.g. 512 x 20 x 15)

Conv

想象一个有固定尺寸
anchor框

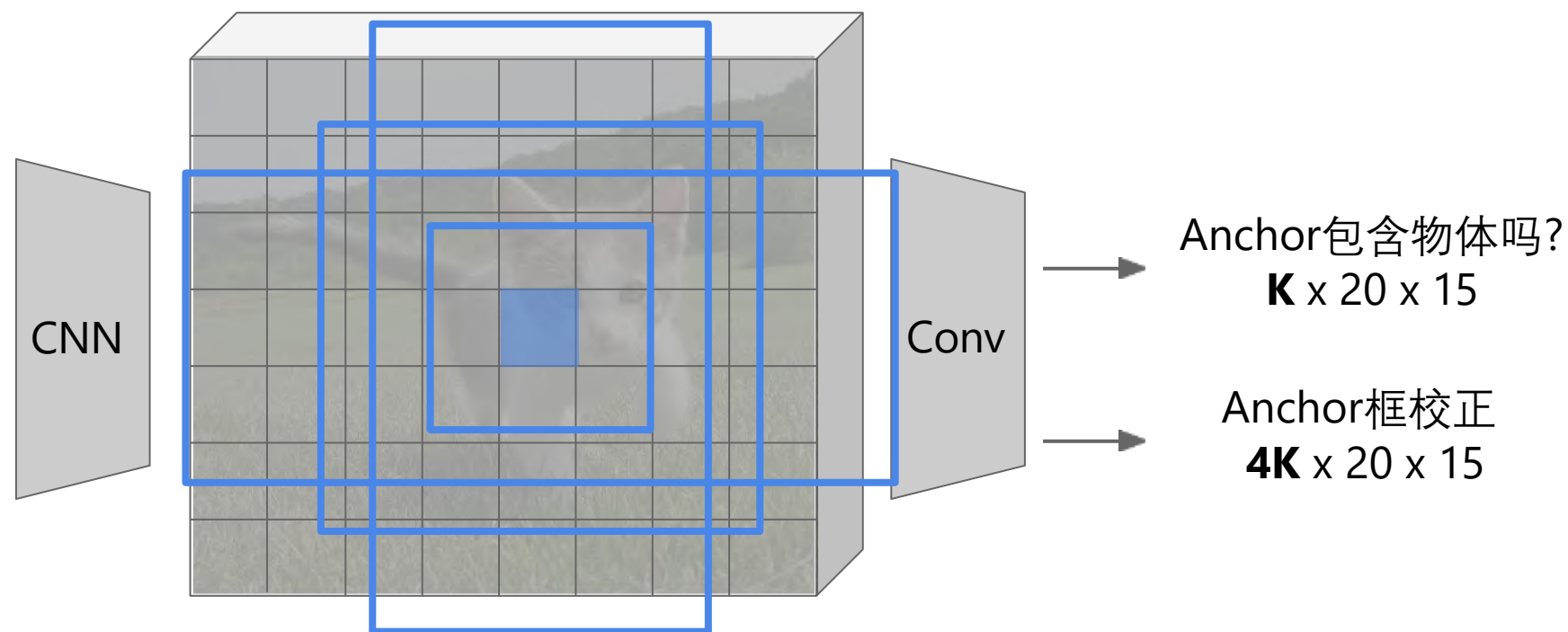
Anchor包含物体吗?
1 x 20 x 15

Anchor框校正
4 x 20 x 15

对于预测有物体的框，预测一个
对anchor框的校正

Region Proposal Network

K个anchor对应不同尺寸、
尺度、形状 的物体



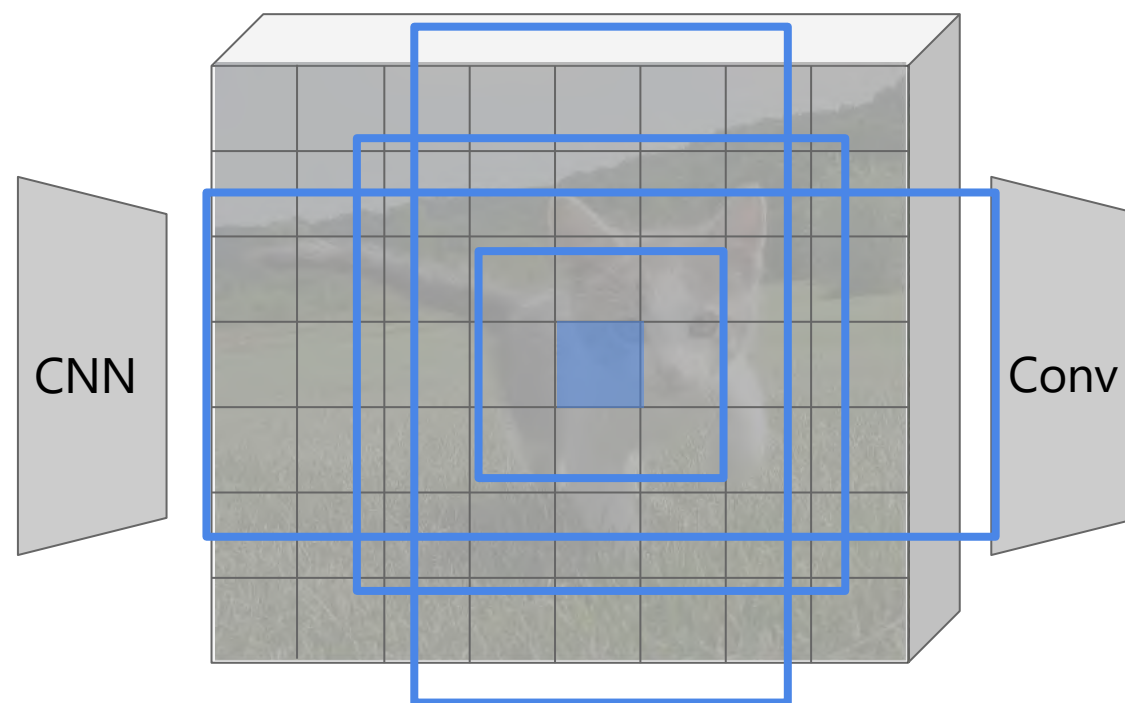
图像特征
(e.g. 512 x 20 x 15)

Region Proposal Network

K个anchor对应不同尺寸、尺度、形状



输入图像
(e.g. 3 x 640 x 480)



图像特征
(e.g. 512 x 20 x 15)

Anchor包含物体吗?
K x 20 x 15

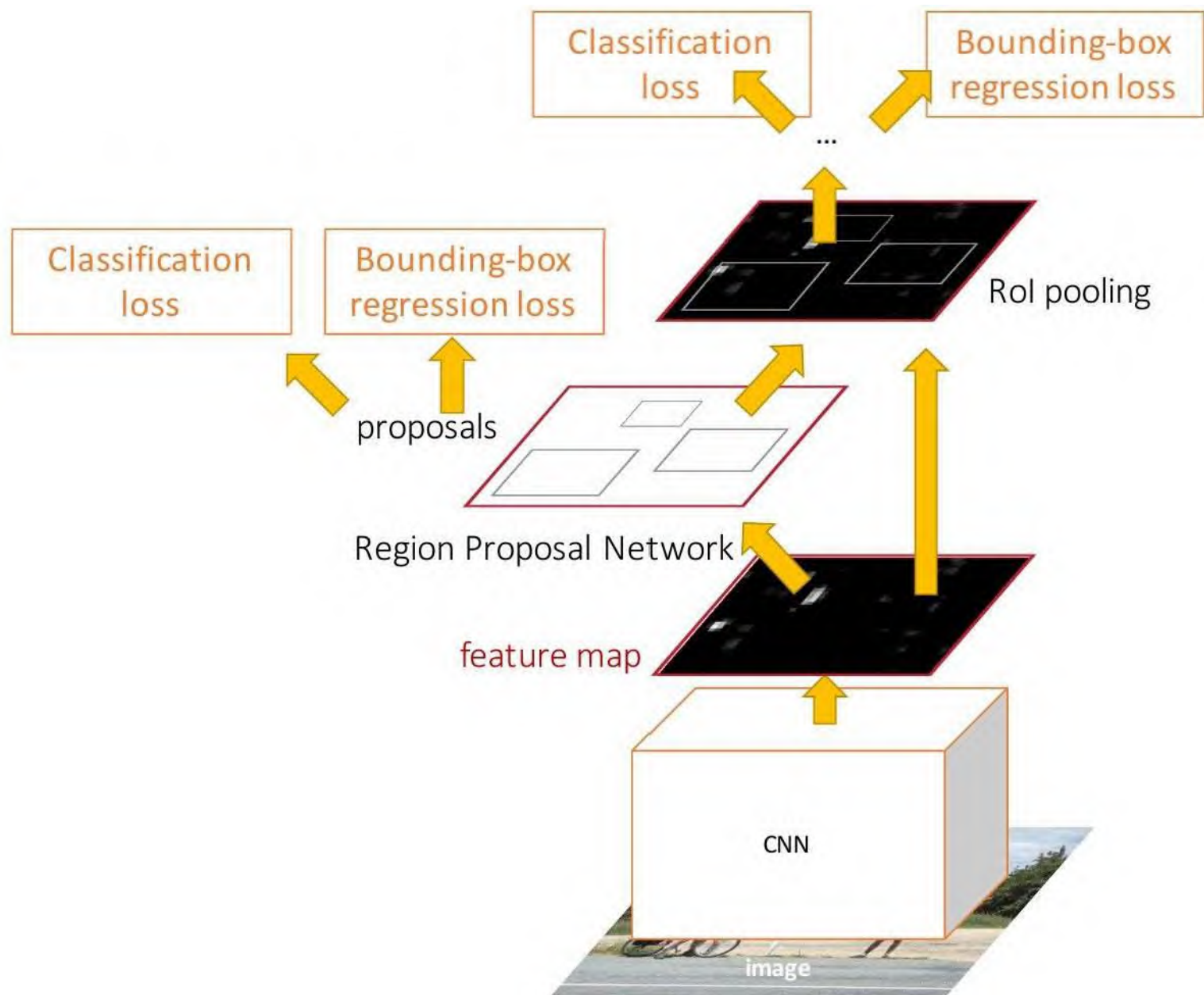
Anchor框校正
4K x 20 x 15

对K*20*15 个框“物体存在的概率”排序，取~300个进行预测，映射回原图

Faster R-CNN: 让CNN提Proposals!

训练4个 losses:

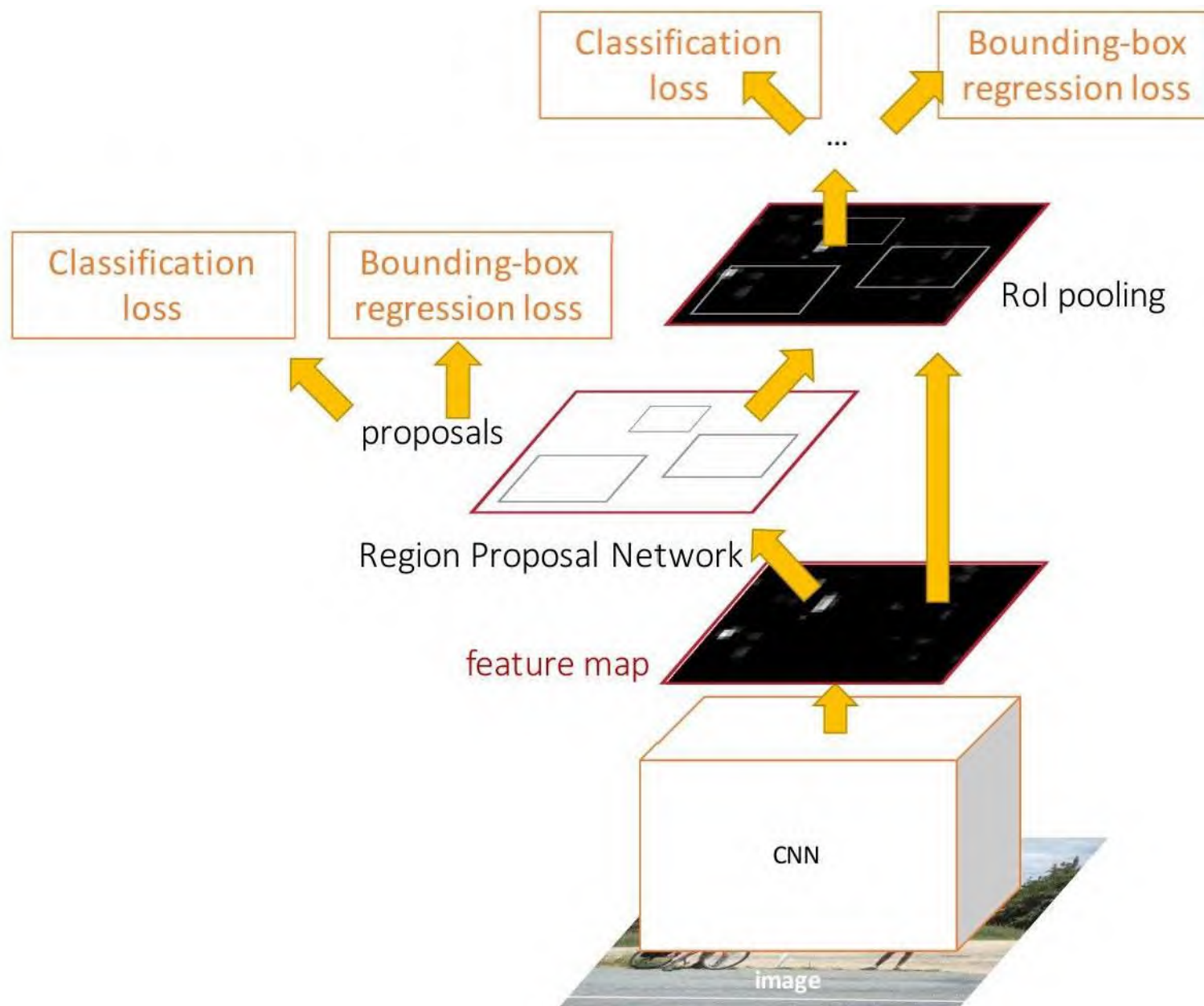
1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classify score (object classes)
4. Final box coordinates



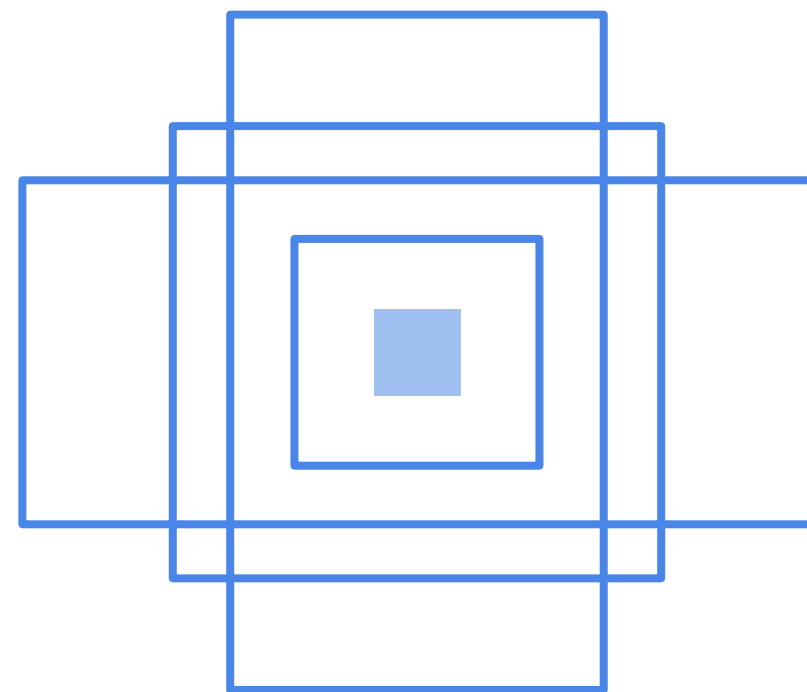
Faster R-CNN: 让CNN提Proposals!

大量的代码细节:

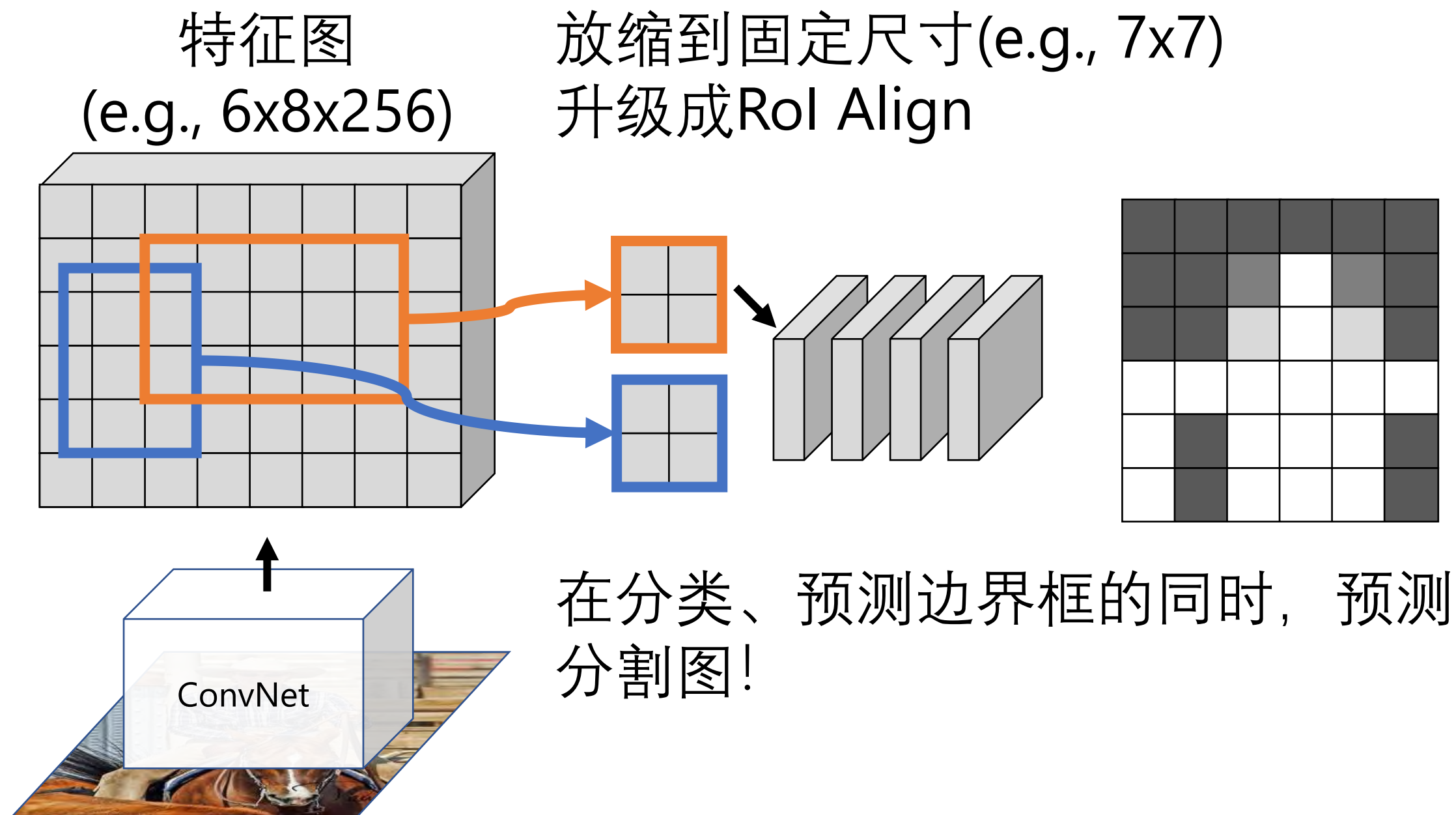
- 去掉重叠的proposal:
non-max suppression
- 多少个anchor?
- 采样多少个 物体/没有物体 的 proposal来训练?
- 怎么做物体框的回归?



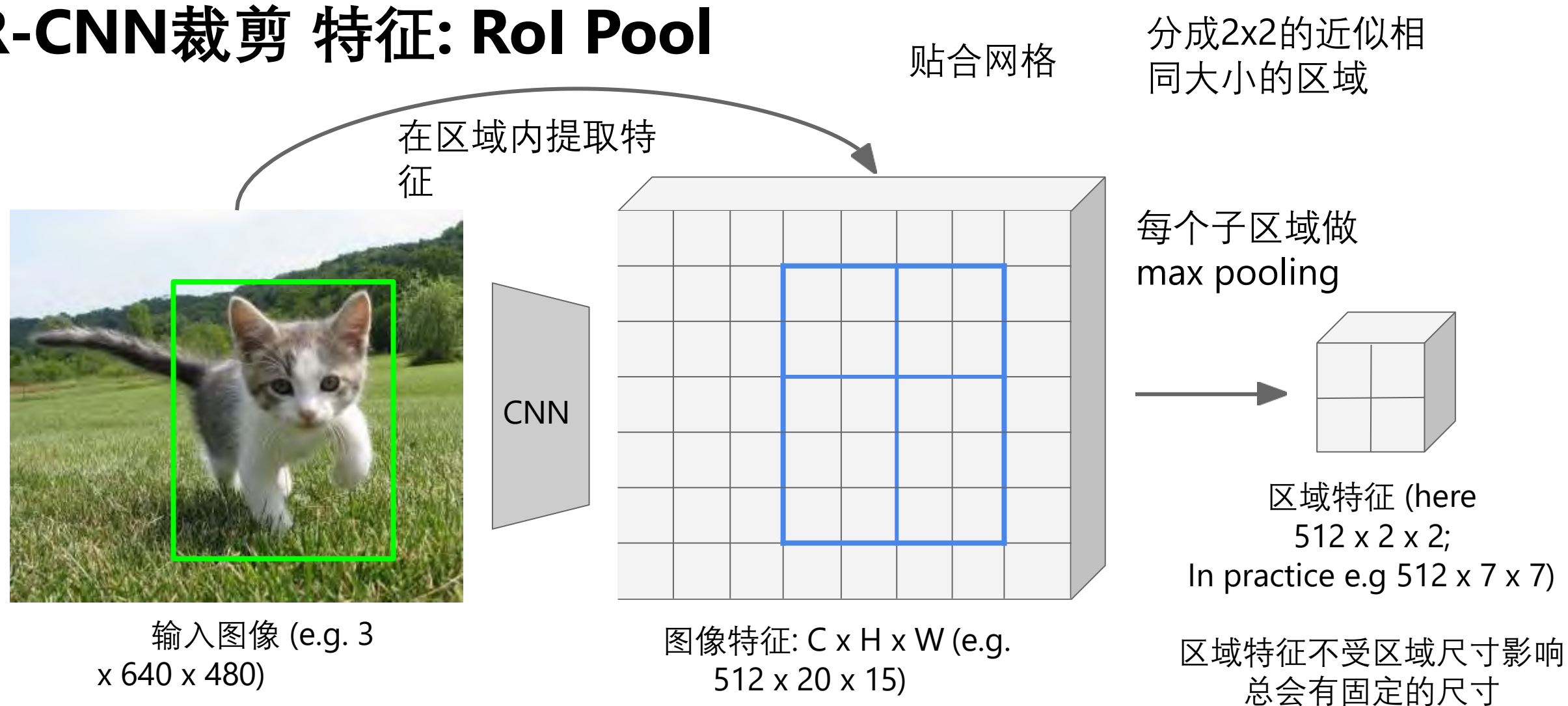
回顾：Selective Search 与 NMS



Mask RCNN

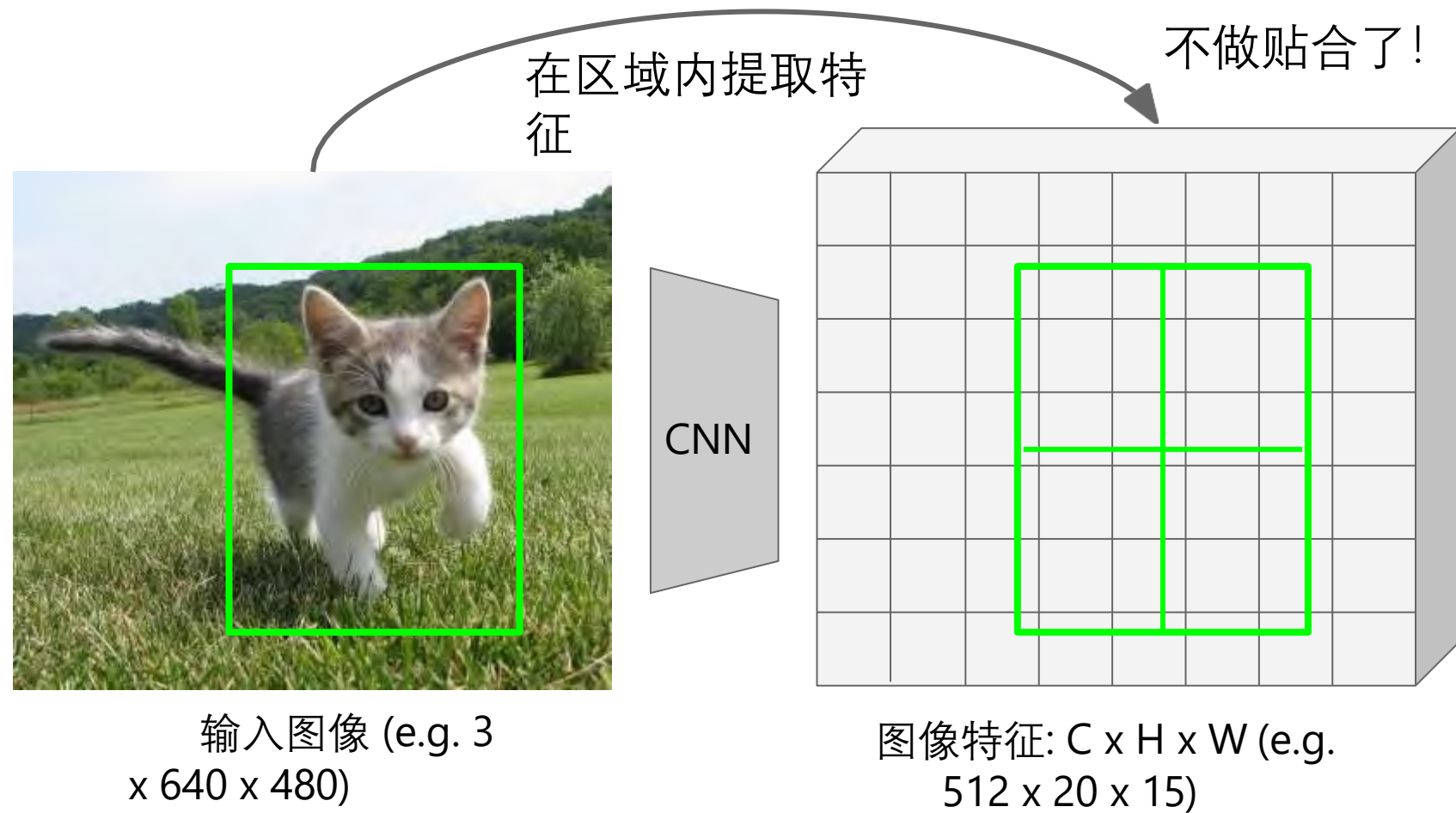


Fast R-CNN裁剪 特征: RoI Pool



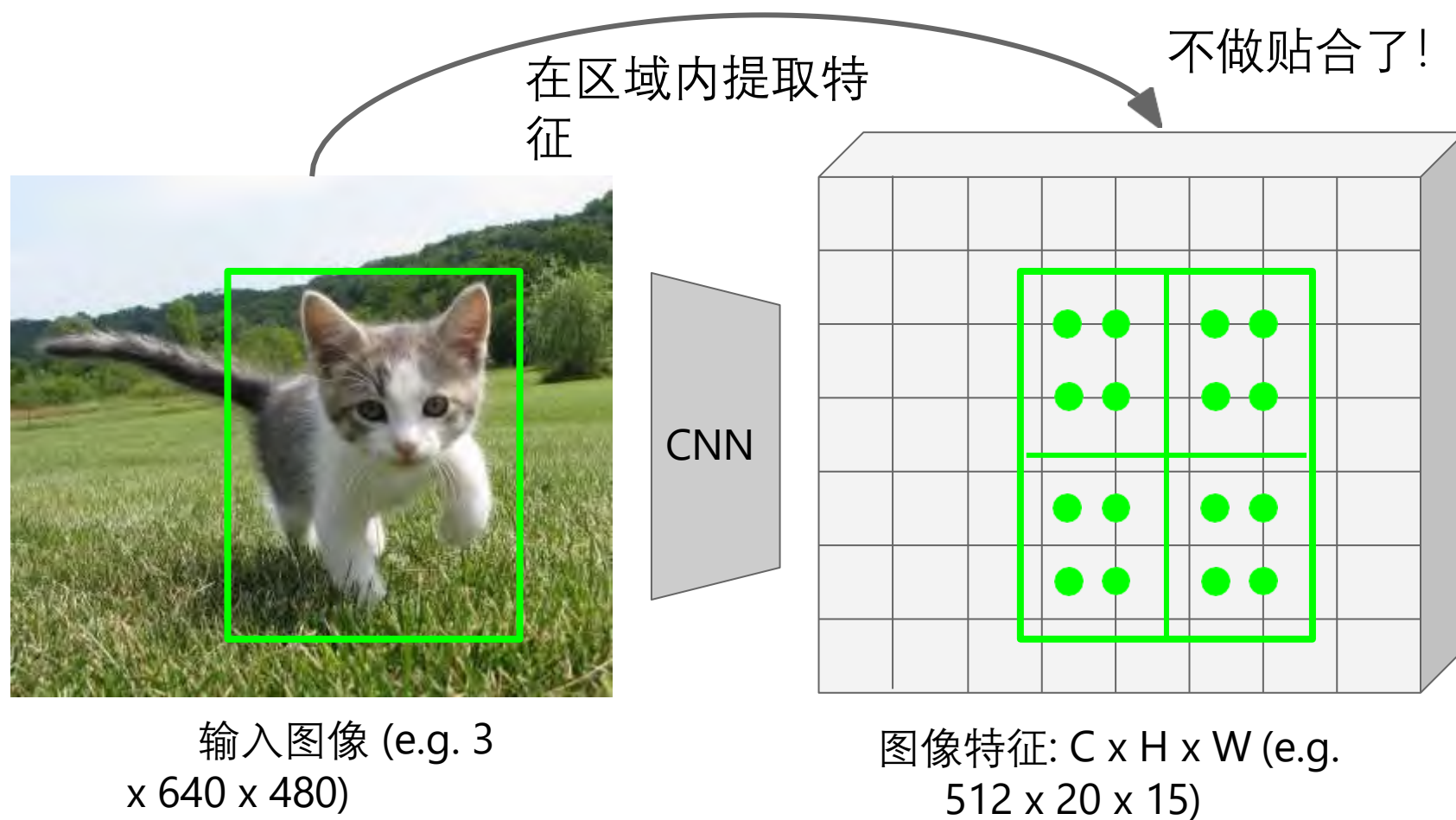
问题: 每个区域的特征格点数量上不均匀

RoI Pool升级版: RoI Align

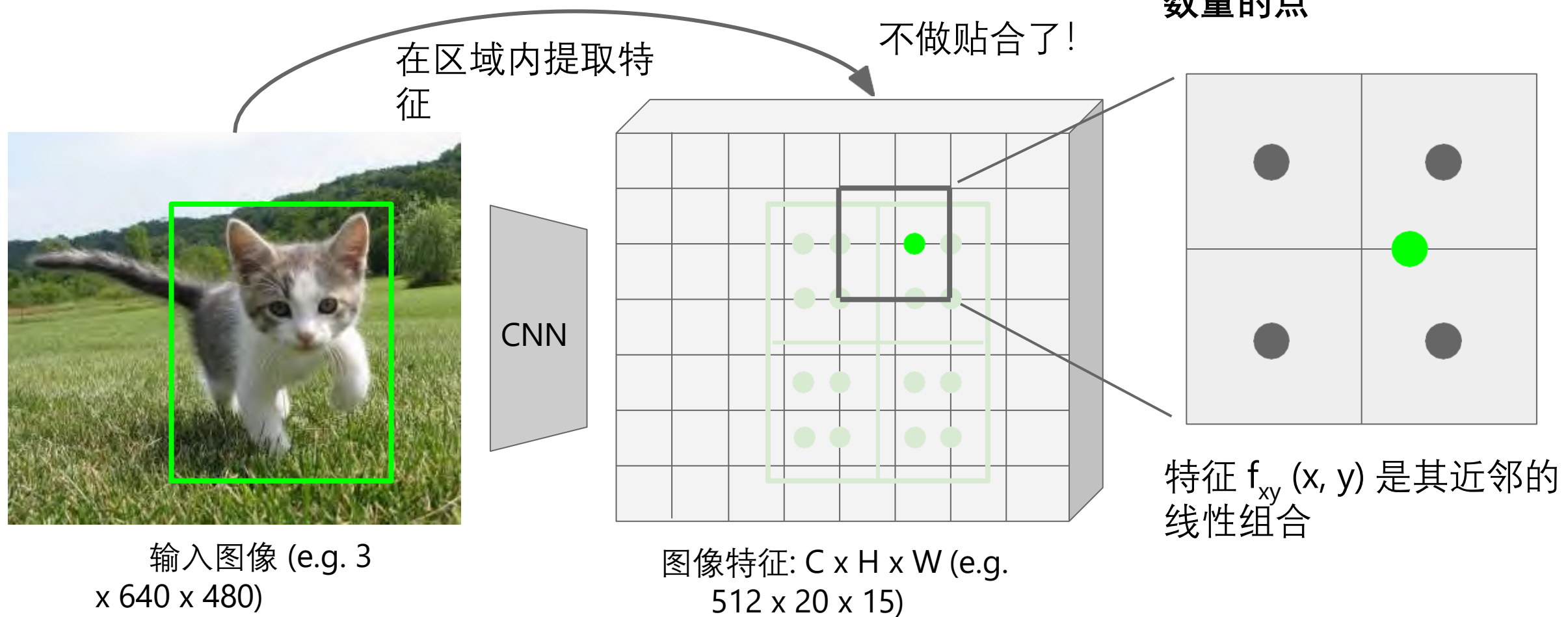


RoI Pool升级版: RoI Align

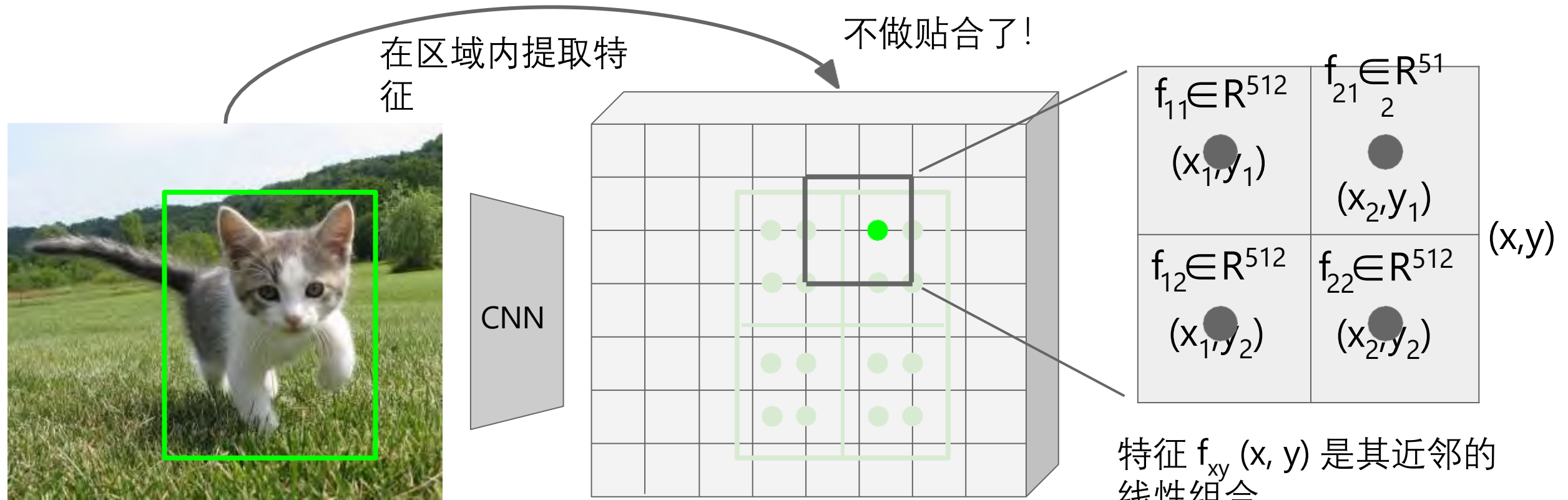
在每个子区域采样固定数量的点



RoI Pool升级版: RoI Align

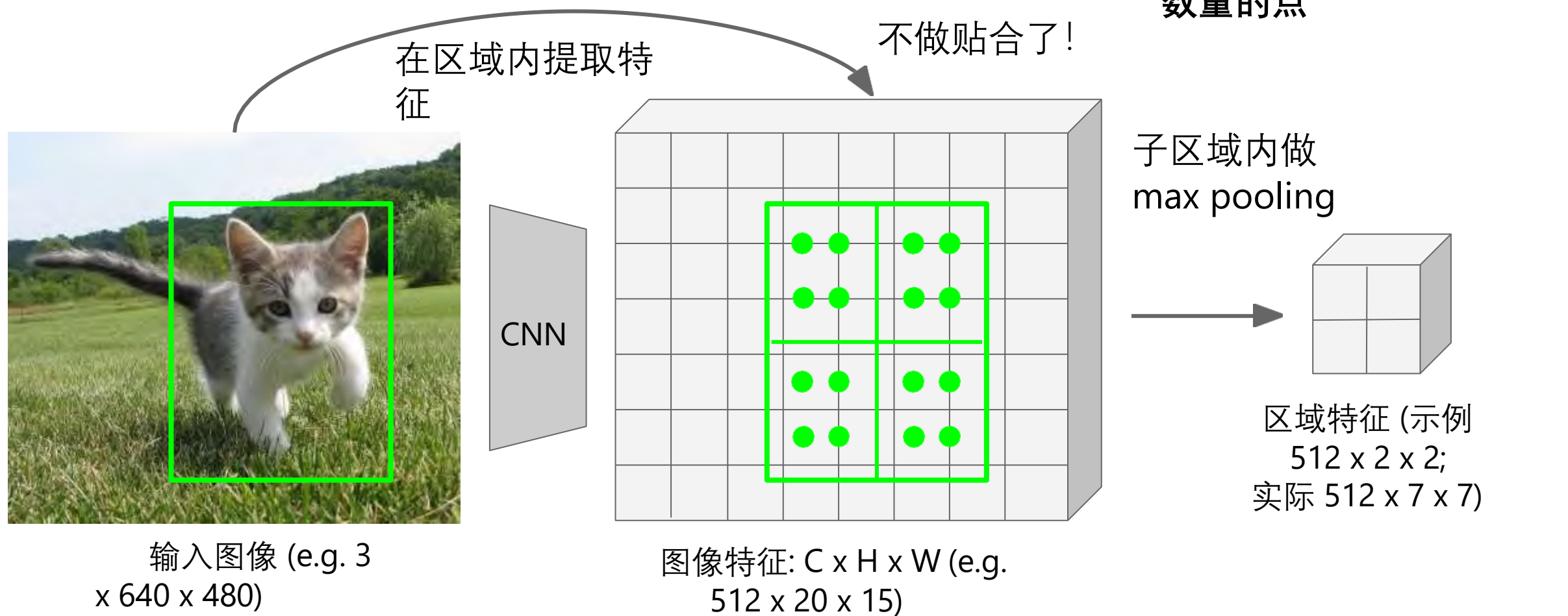


RoI Pool升级版: RoI Align



$$f_{xy} = \sum_{i,j=1}^2 f_{i,j} \max(0, 1 - |x - x_i|) \max(0, 1 - |y - y_j|)$$

RoI Pool升级版: RoI Align



MaskRCNN - 效果



Faster R-CNN: 让CNN提Proposals!

Faster R-CNN 是一个
两阶段物体检测器

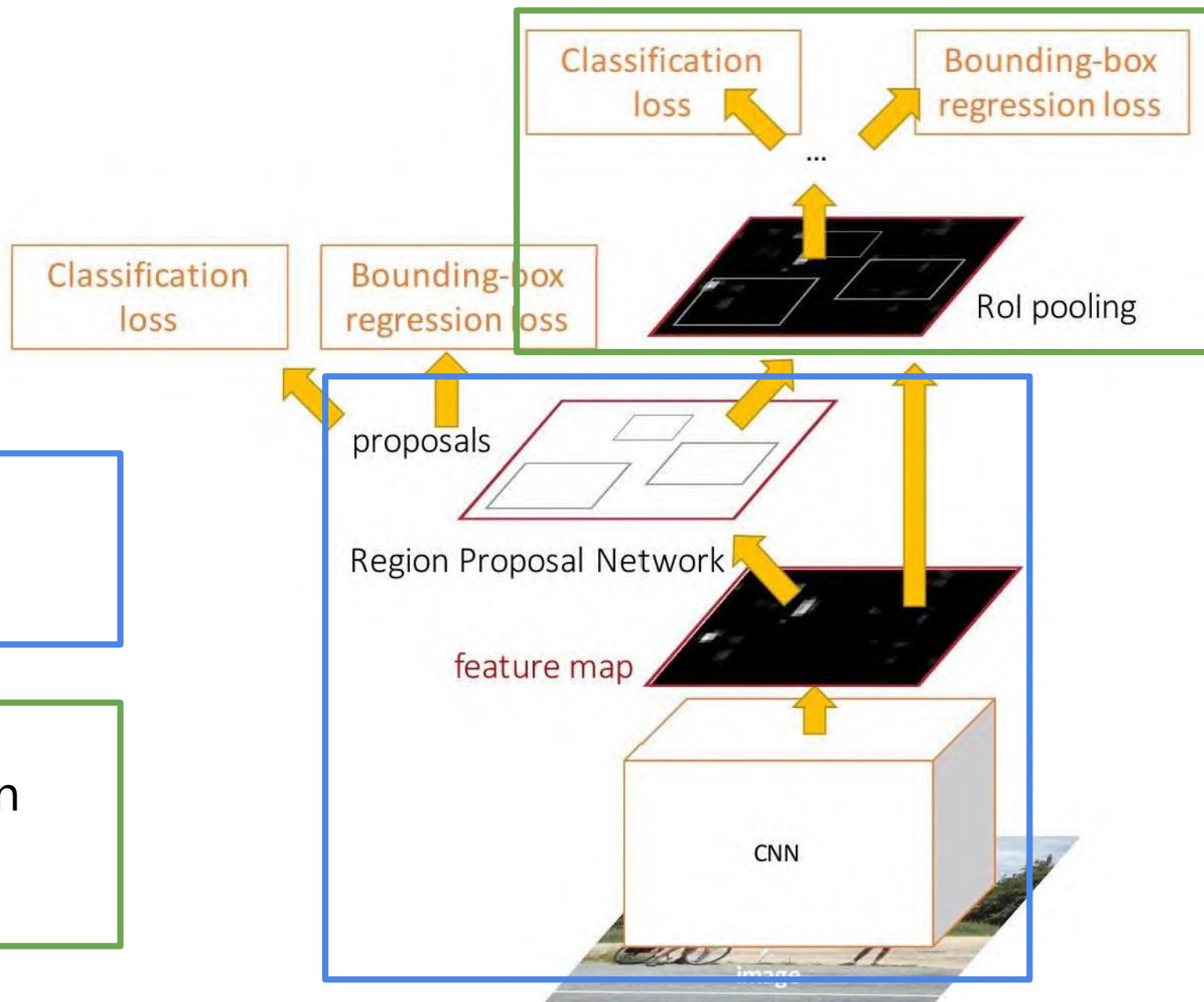
第一个阶段: 在每张图上

- Backbone network
- Region proposal network

第二个阶段: 在每个区域上

- 裁剪特征: RoI pool / align
- 预测物体类别
- 预测框回归

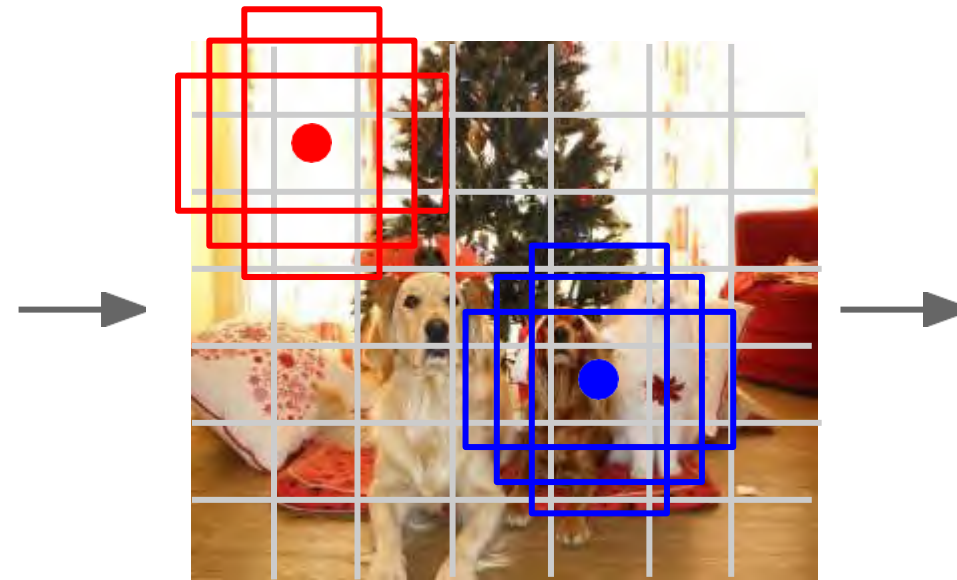
我们真的需要第二个阶段吗?



单阶段物体检测方法: YOLO / SSD / RetinaNet



输入图像 $3 \times H \times W$



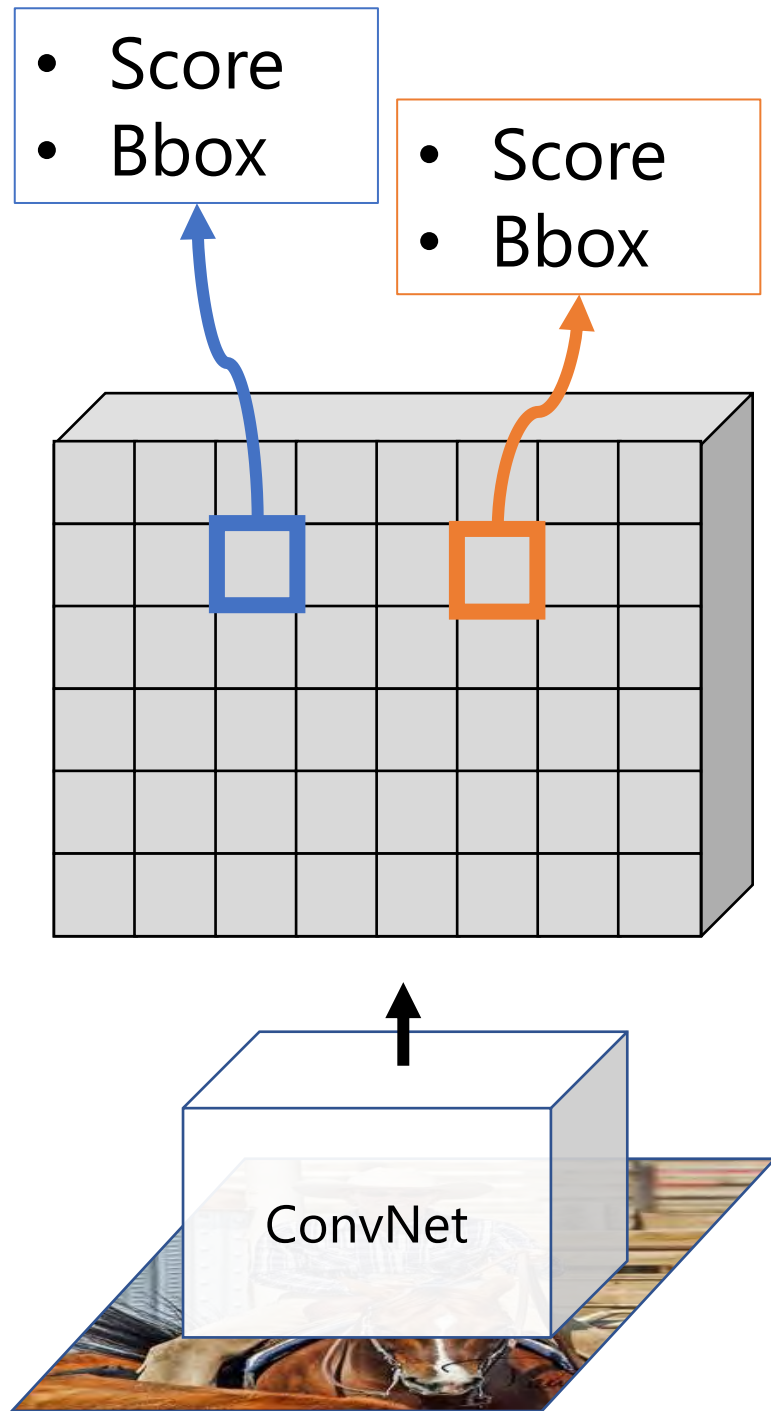
把图像分成 7×7 的格点
每张图会包含一些以格点为中心的**基础框**
Here $B = 3$

对于每一个格点:

- 从 B 个基础框 (Anchor) 来回归出最终的物体框:
($dx, dy, dh, dw, confidence$)
- 每个框对每个类都预测一个值 (包括背景)
- 类似于我们有 7×7 个 RPN

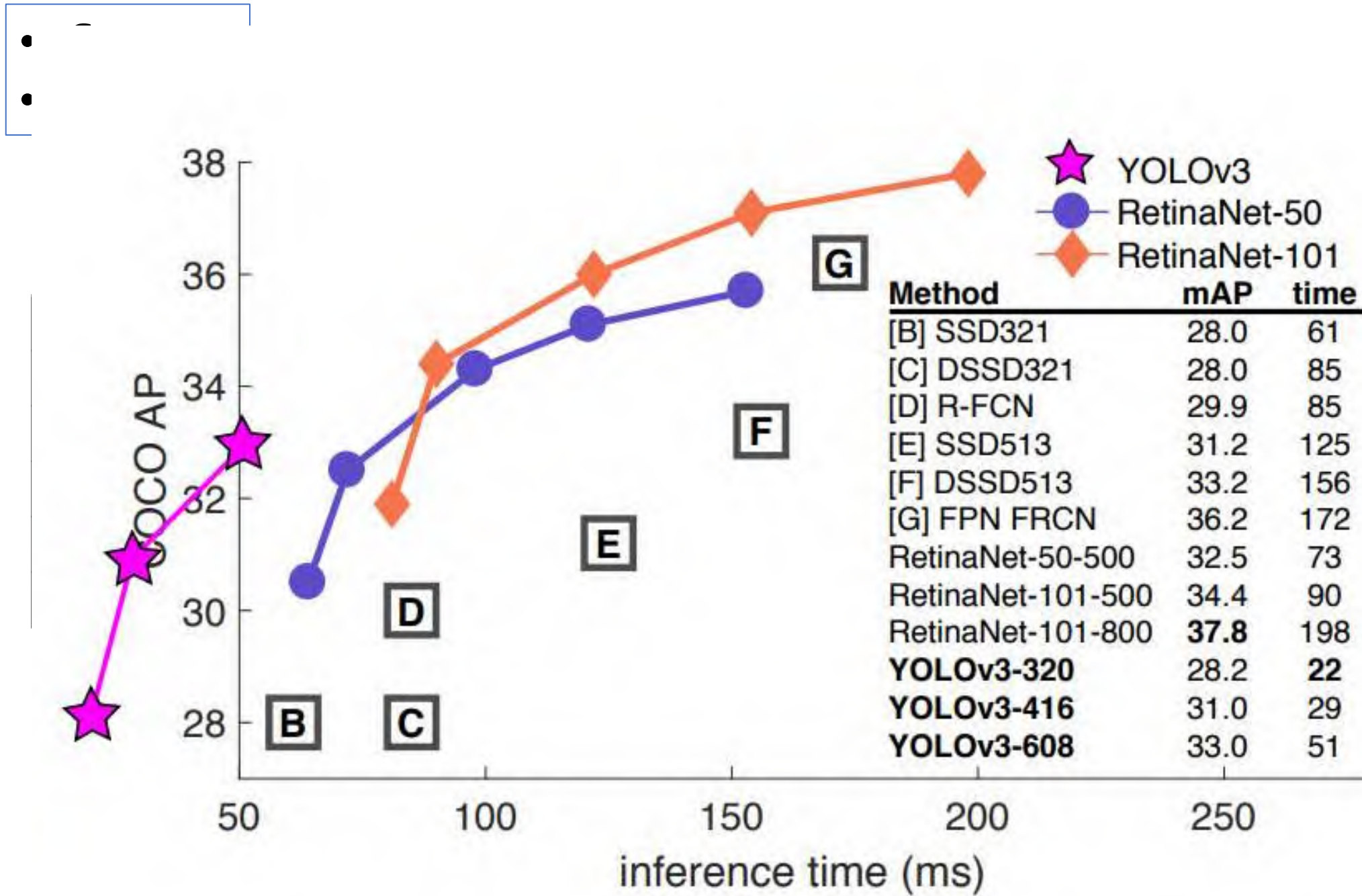
Output:
 $7 \times 7 \times (5 * B + C)$

YOLO



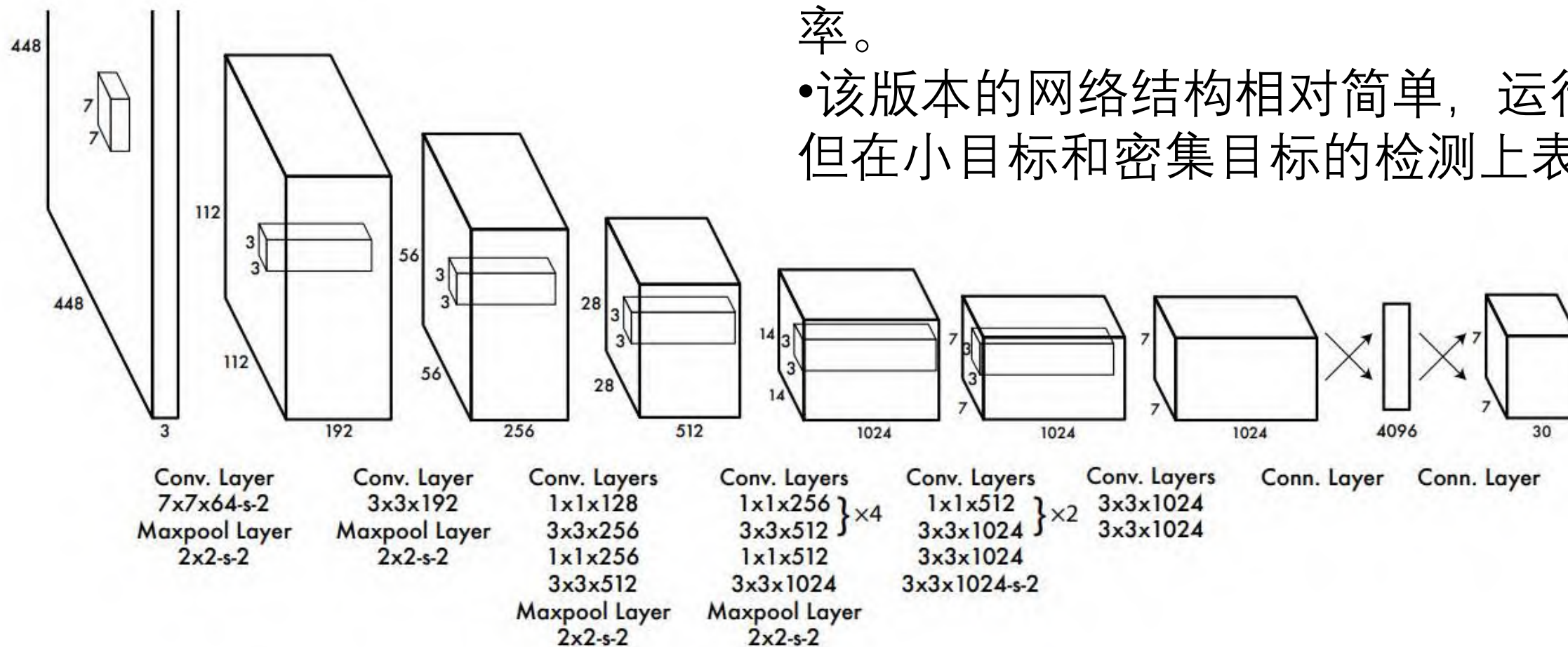
- 没有区域提议
- 对7x7的特征图上的每个点，预测分类评分+边界框校正
- 比 Faster-RCNN 快7倍，但精确度较低
- 由于其速度较快，在实际应用中，如机器人等，较常使用
- 众多版本

YOLO

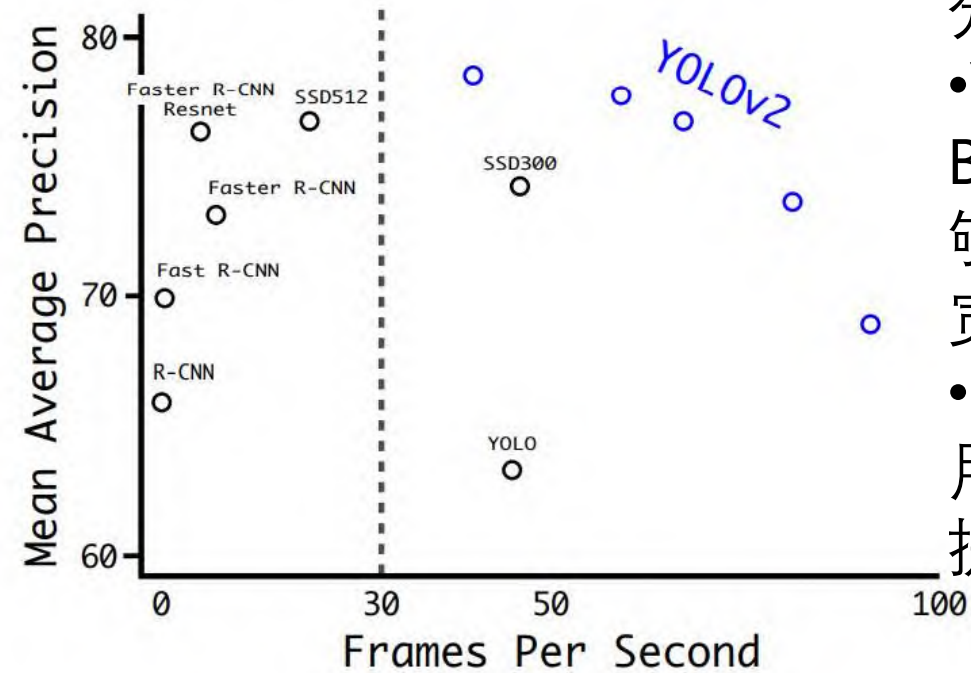
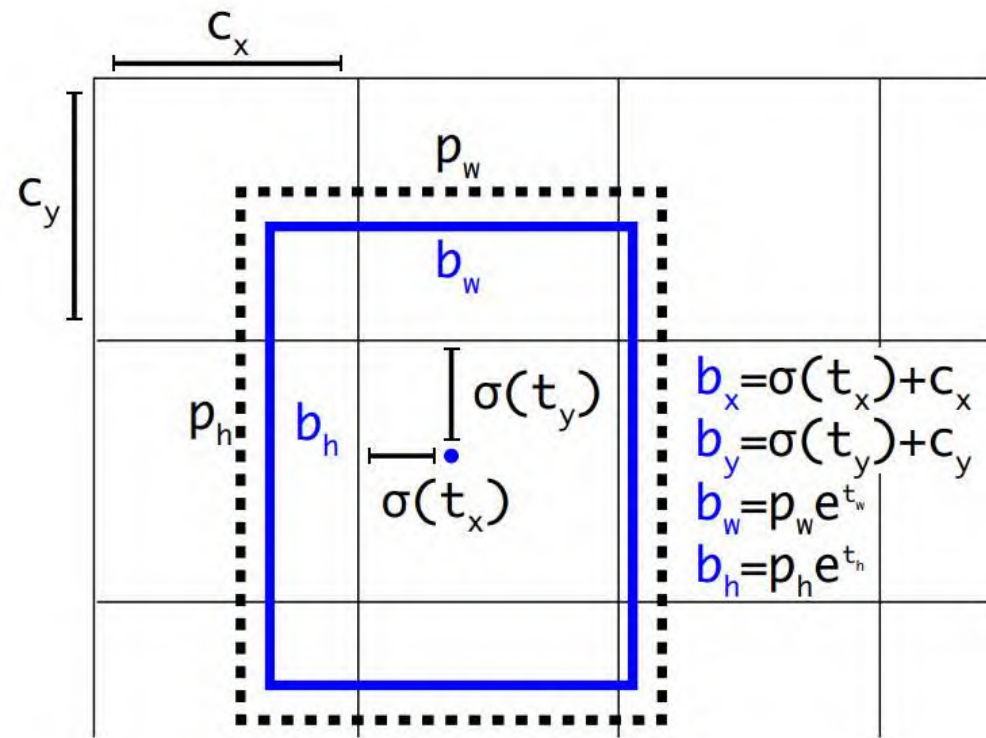


YOLO Zoo

- 发布于2015年，是YOLO系列的首个版本。
- YOLO v1采用单一卷积神经网络来进行实时目标检测，将检测问题视为回归问题，并在一次前向传递中直接预测边界框的坐标和类别概率。
- 该版本的网络结构相对简单，运行速度很快，但在小目标和密集目标的检测上表现不佳



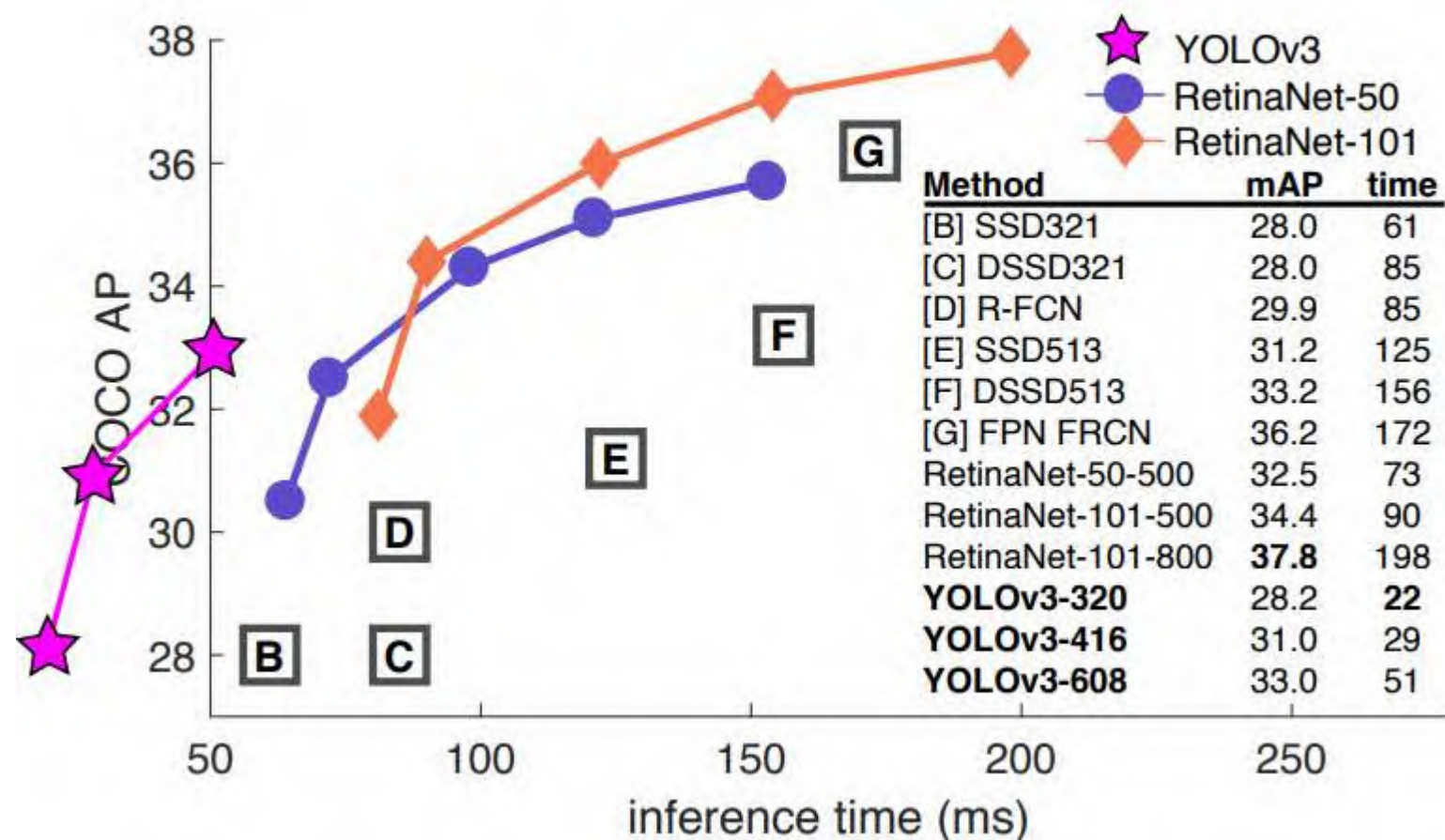
YOLO Zoo



- 发布于2016年，是对YOLO v1的改进和升级。
- 引入了Darknet-19和Darknet-53两种网络结构，分别具有19和53层。
- YOLO v2采用了Anchor Boxes的概念，使得模型能够更好地预测不同尺寸和长宽比的目标。
- 另外，YOLO v2在训练时采用了多尺度训练和预测策略，提高了对小目标的检测精度。

<https://arxiv.org/pdf/1612.08242.pdf> YOLO v2 Introduced anchor boxes to help predict different object sizes and aspect ratios.

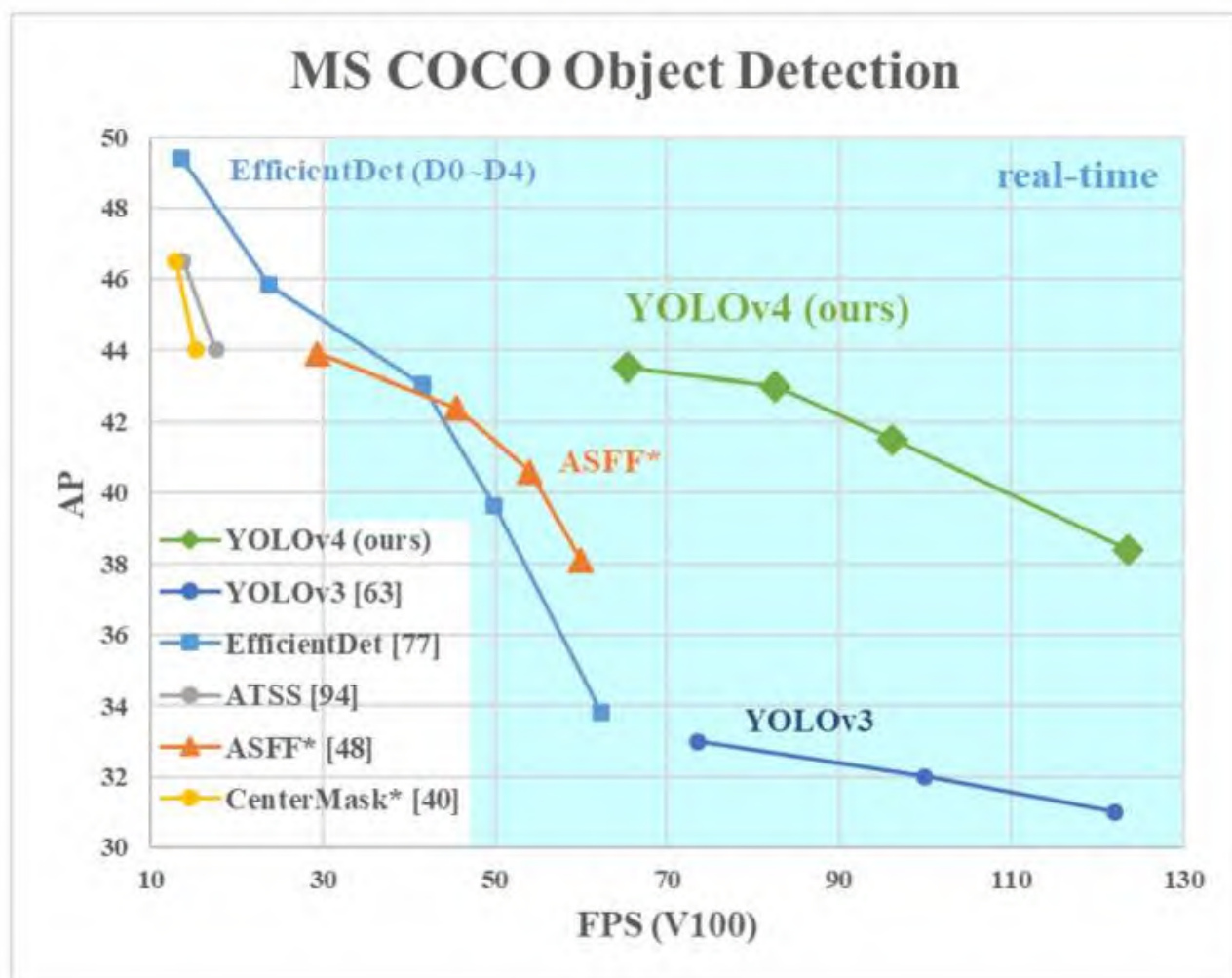
YOLO Zoo



- 发布于2018年，是YOLO系列的第三个版本。
- YOLO v3进一步增加了网络的深度和复杂性，引入了更多的卷积层。
- 使用了3种不同尺度的特征图来检测不同大小的目标，每个尺度都有自己的检测器和Anchor Boxes。
- YOLO v3采用了多尺度预测和Darknet-53网络结构，使得模型在准确性和速度上都有较好的表现。

<https://arxiv.org/pdf/1804.02767.pdf> YOLO v3 was able to detect smaller objects more effectively and achieved higher accuracy compared to its predecessors.

YOLO Zoo



- 发布于2020年，是YOLO系列的最新版本。
- YOLO v4引入了一系列技术改进，包括 CSPDarknet53、SAM（空间注意力）、PANet（多尺度融合）等，使得网络更加高效和准确。
- 采用更大的模型和更多的卷积层，进一步提升了检测精度。
- YOLO v4还引入了诸如YOLOv4-tiny和YOLOX等变体，以满足不同场景下的需求

<https://arxiv.org/pdf/2004.10934.pdf> YOLO v4 Implemented several advanced techniques such as PANet, SAM, and PAN, to enhance feature representation and object detection.

物体检测: 各种变种...

Backbone Network

VGG16

ResNet-101

Inception V2

Inception V3

Inception

ResNet

MobileNet

“Meta-Architecture”

Two-stage: Faster R-CNN

Single-stage: YOLO / SSD

Hybrid: R-FCN

Image Size

Region Proposals

...

总结

Faster R-CNN 慢, 但是准很多

SSD/YOLO快, 但是不准

更深的网络总是表现得更好

实例分割

分类



CAT

无空间限定

语义分割



GRASS, CAT,
TREE, SKY

像素级预测

物体检测



DOG, DOG, CAT

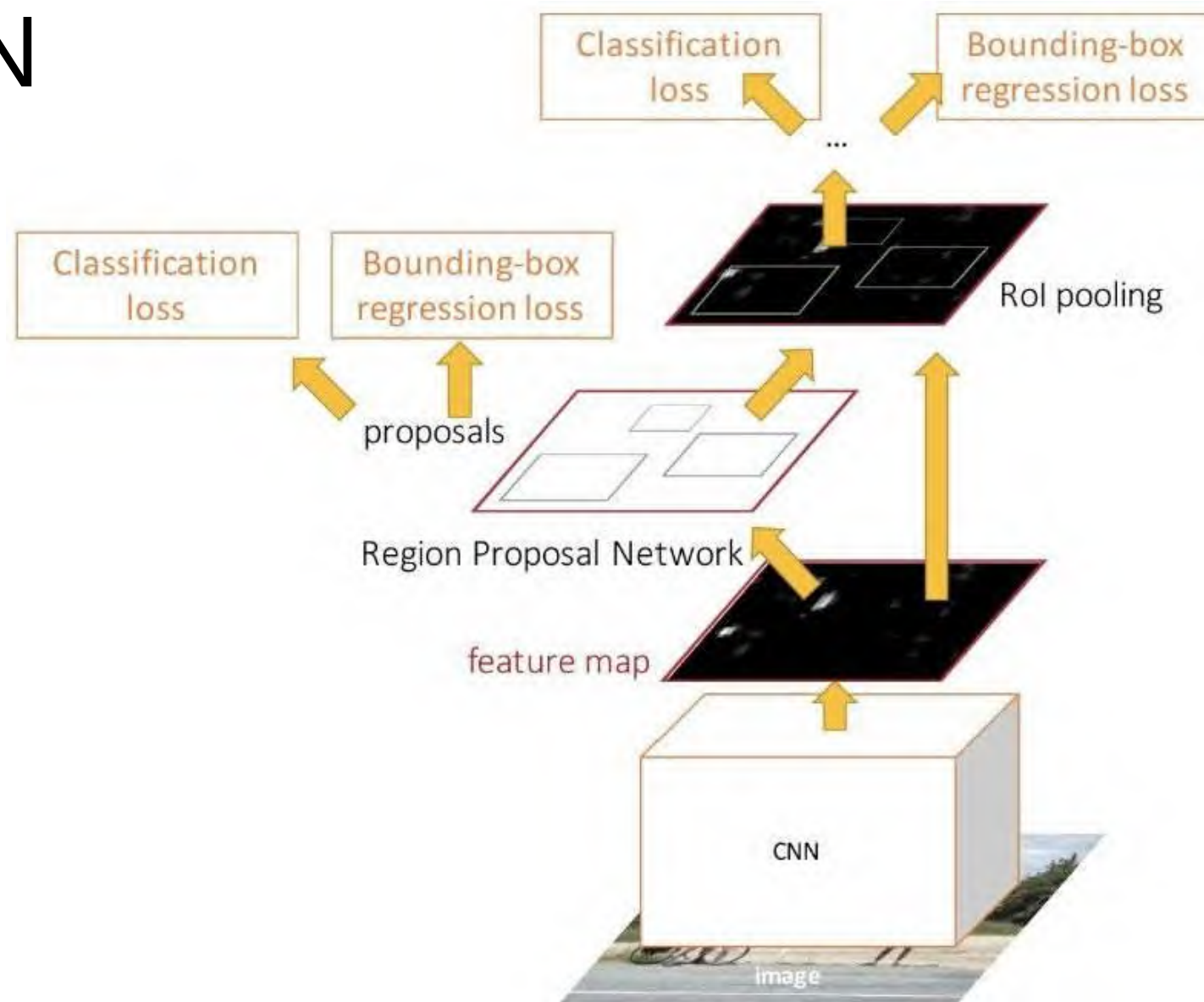
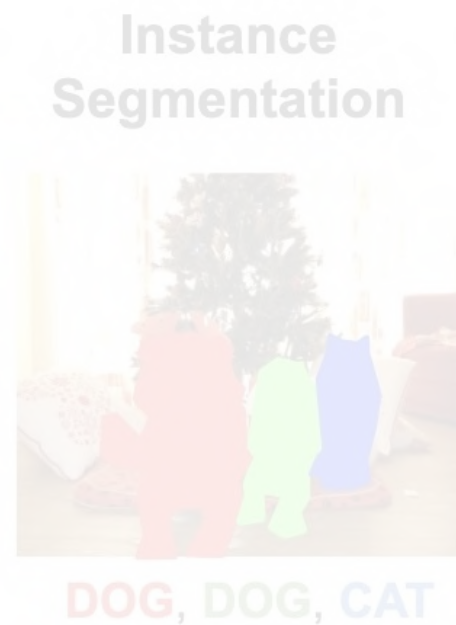
多物体

实例分割

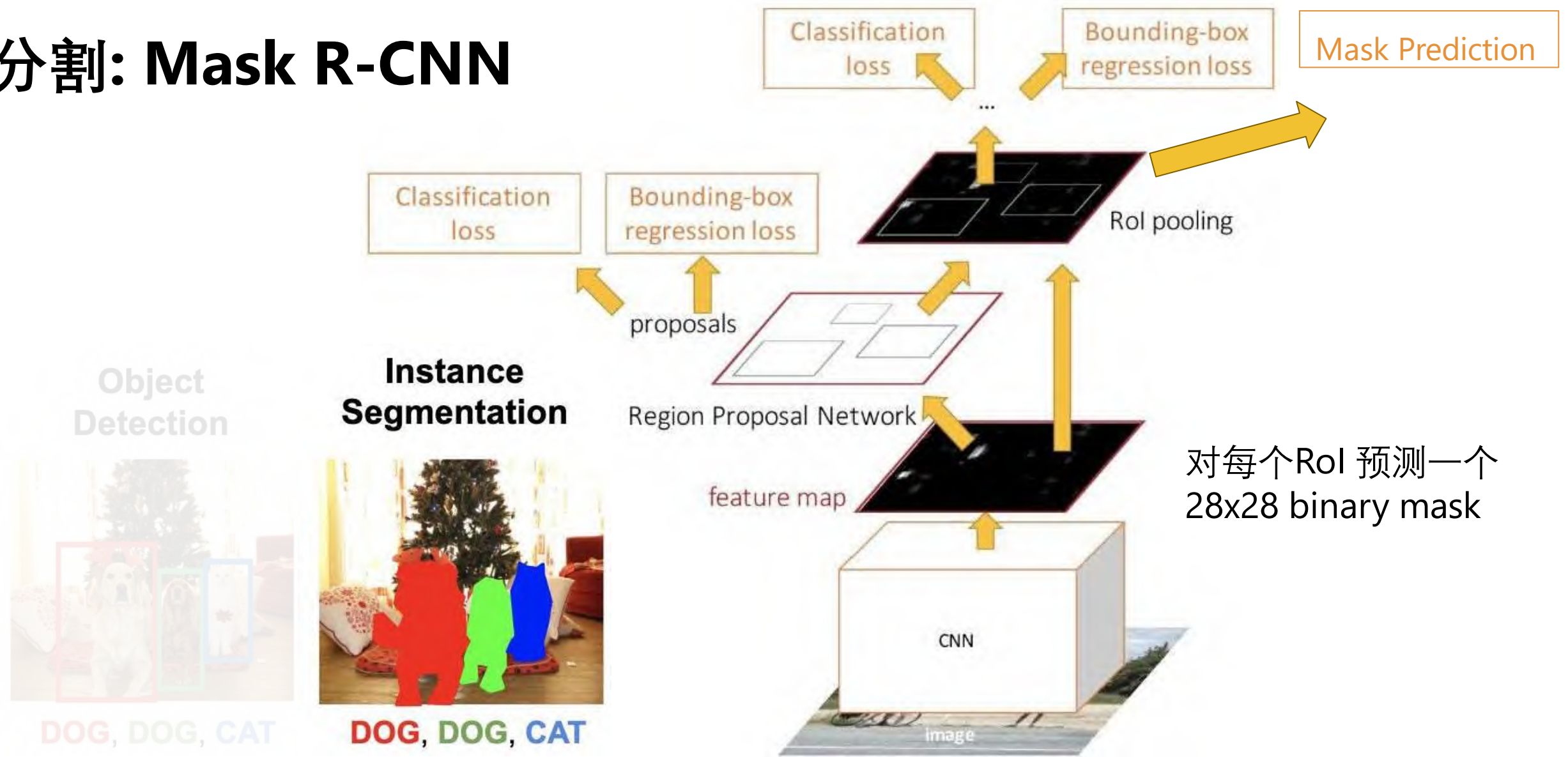


DOG, DOG, CAT

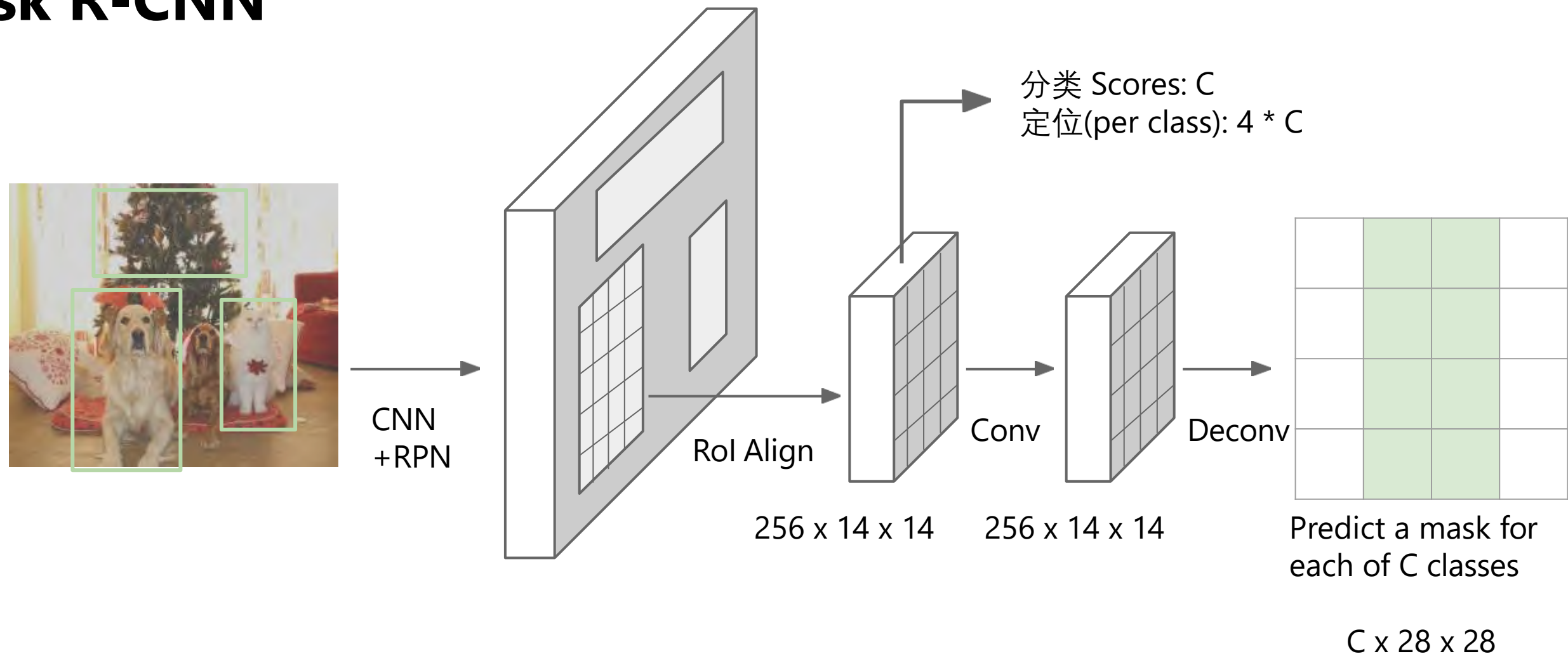
物体检测: Faster R-CNN



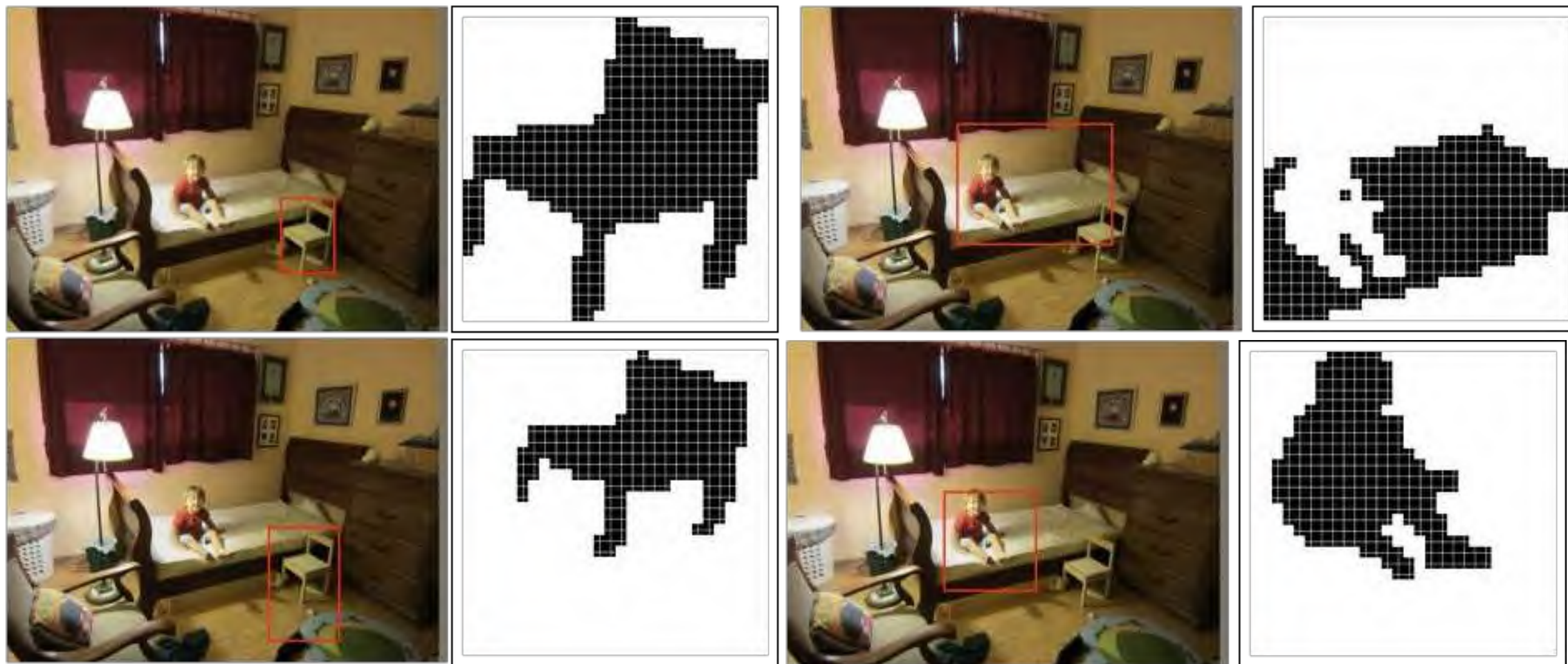
实例分割: Mask R-CNN



Mask R-CNN



Mask R-CNN: 例子



Mask R-CNN 也可以做姿态估计

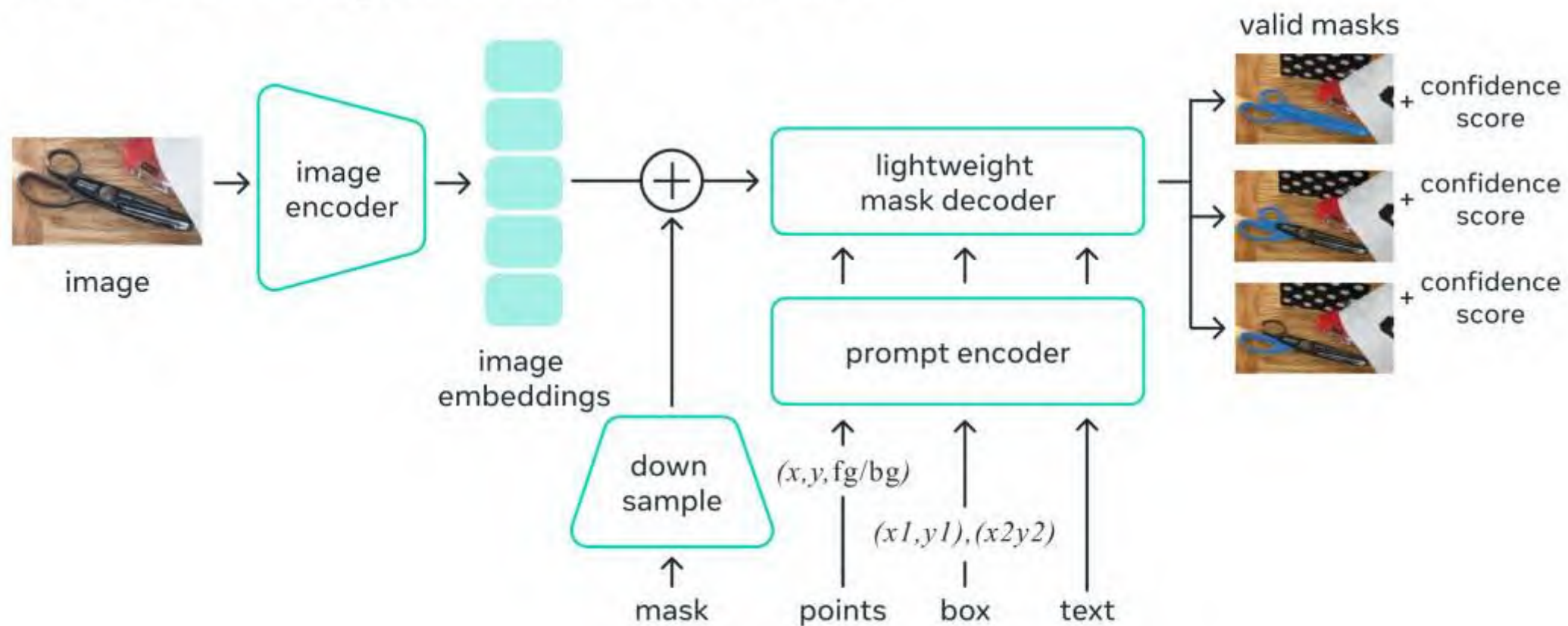


He et al, "Mask R-CNN", ICCV 2017

Segment Anything

针对任何提示返回有效的分割掩码

Universal segmentation model



Segment Anything

点击作为触发词



Segment Anything

标签作为触发词



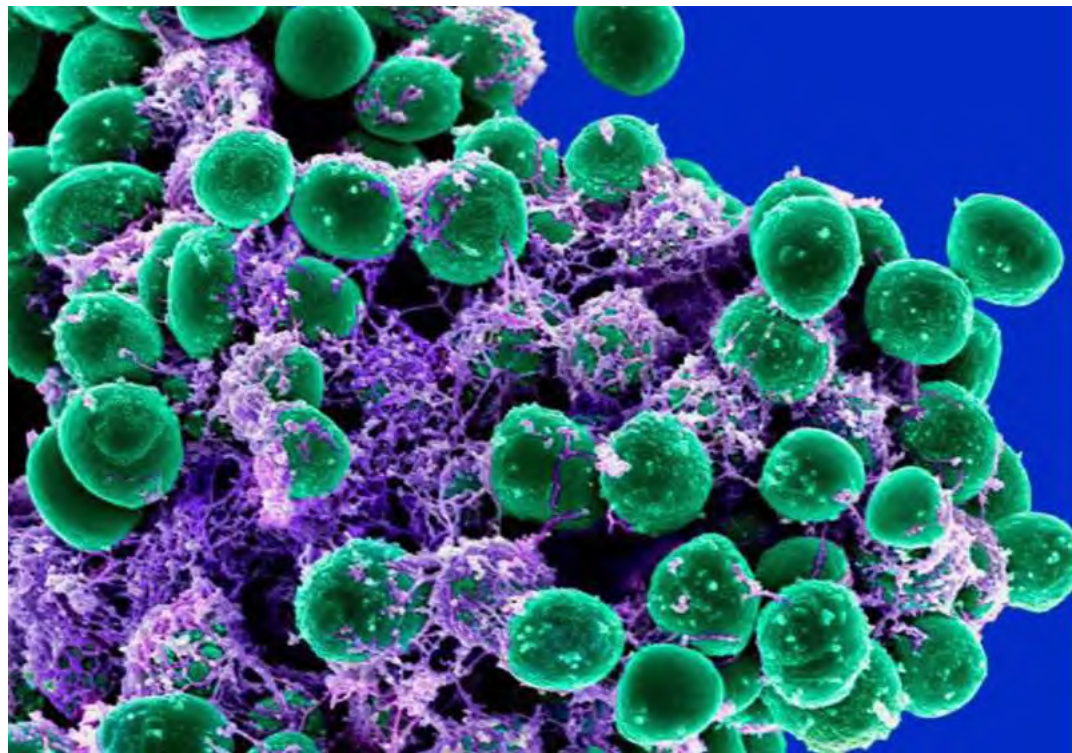
Segment Anything

“valid mask” 不同触发词对应的合理掩码

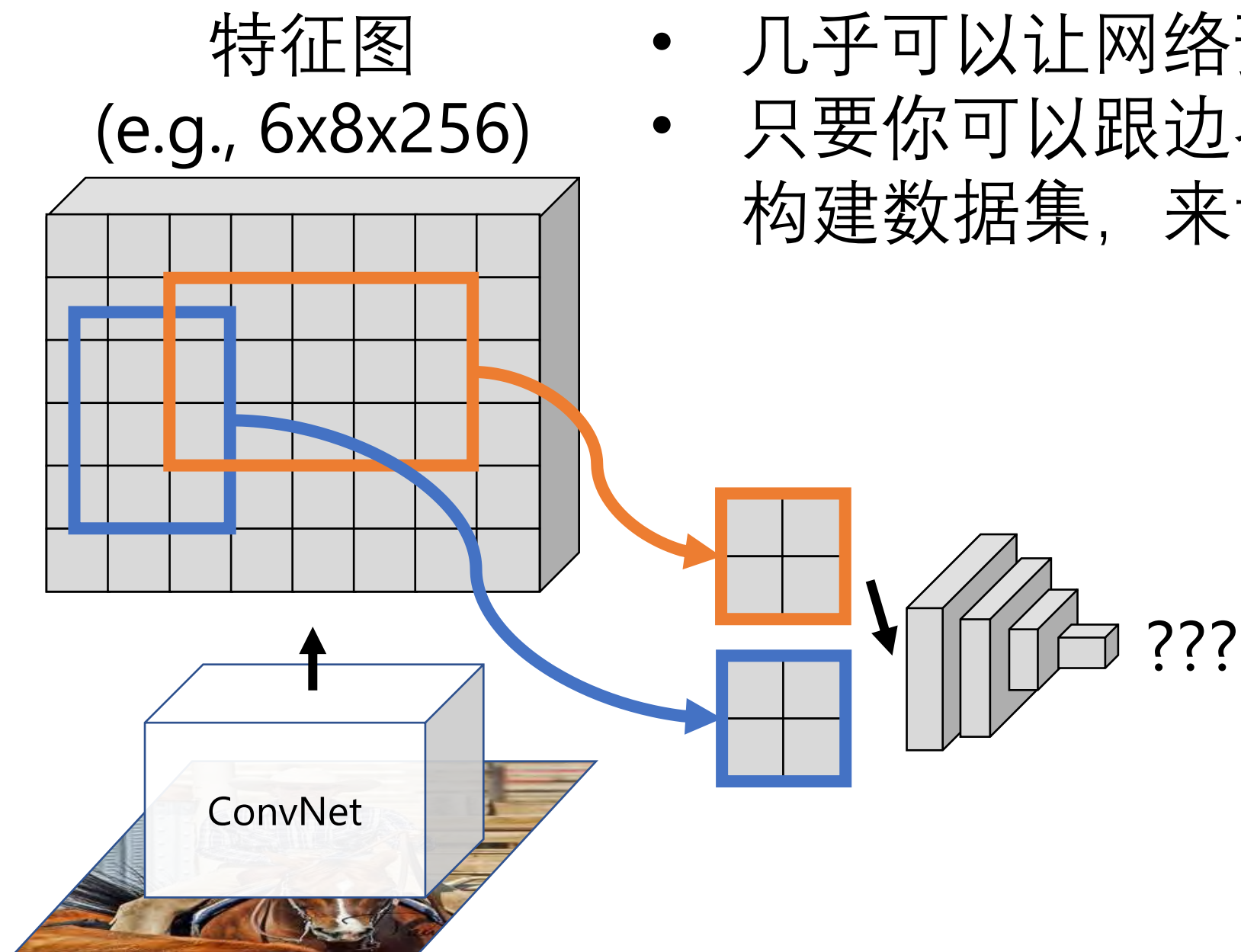


Segment Anything

零知识“zero-shot”预测



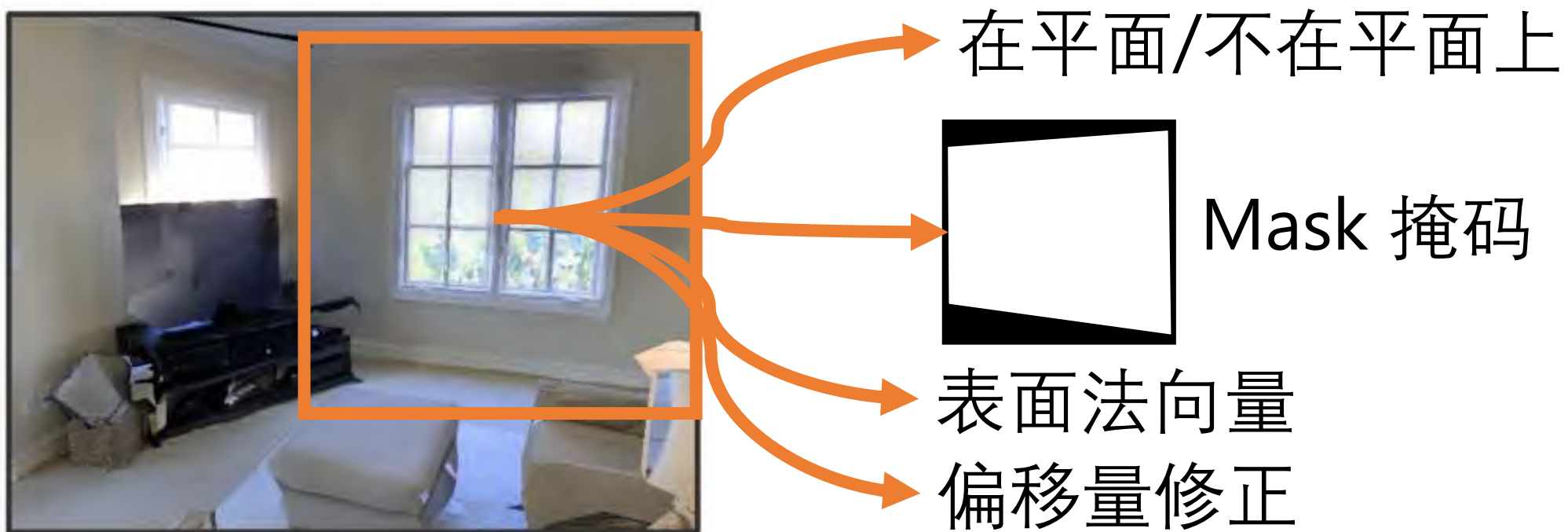
物体检测的拓展



- 几乎可以让网络预测任何东西.
- 只要你可以跟边界框结合起来, 构建数据集, 来训练网络

平面检测

示例: RGB 图像输入, 预测平面



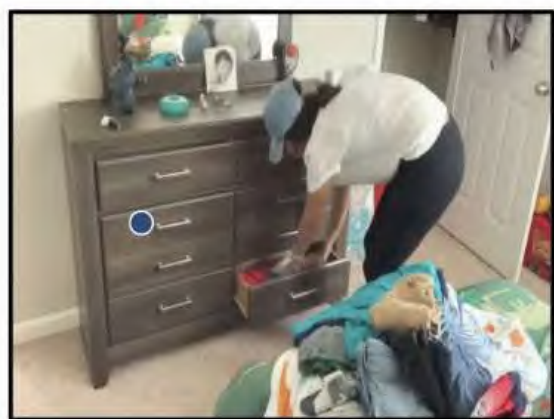
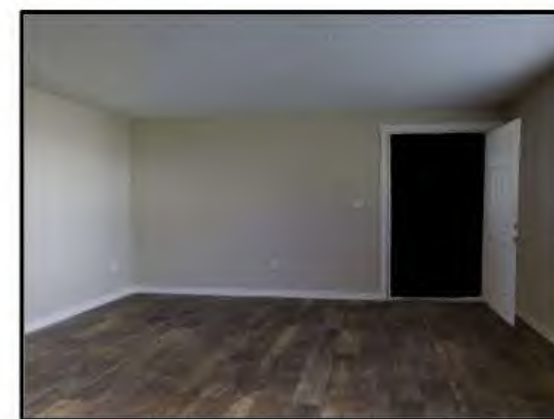
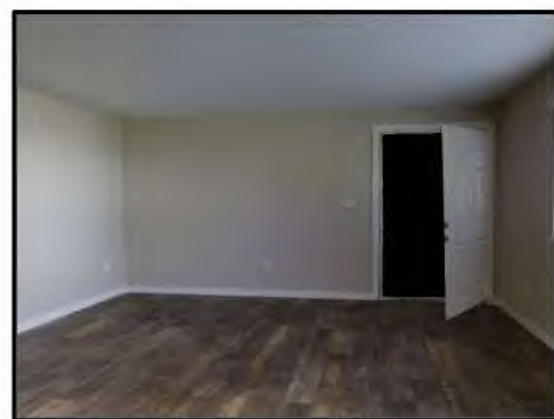
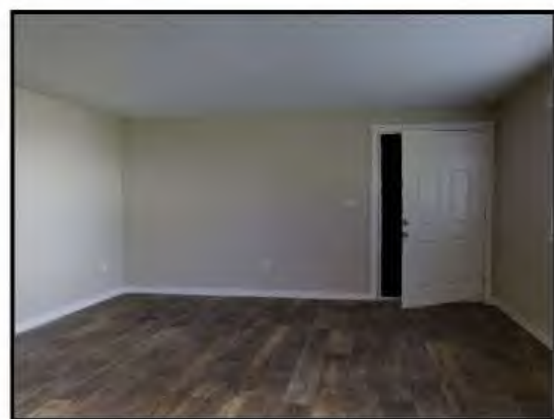
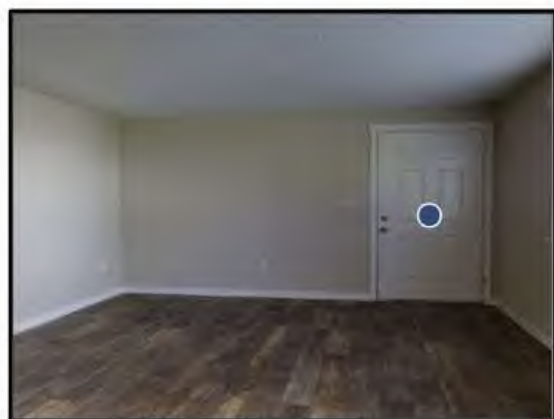
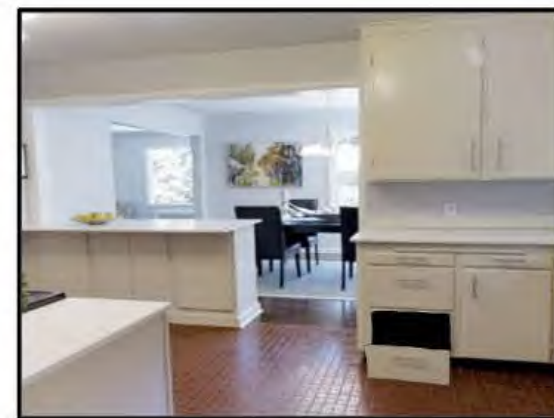
平面检测



功能：构建3D场景

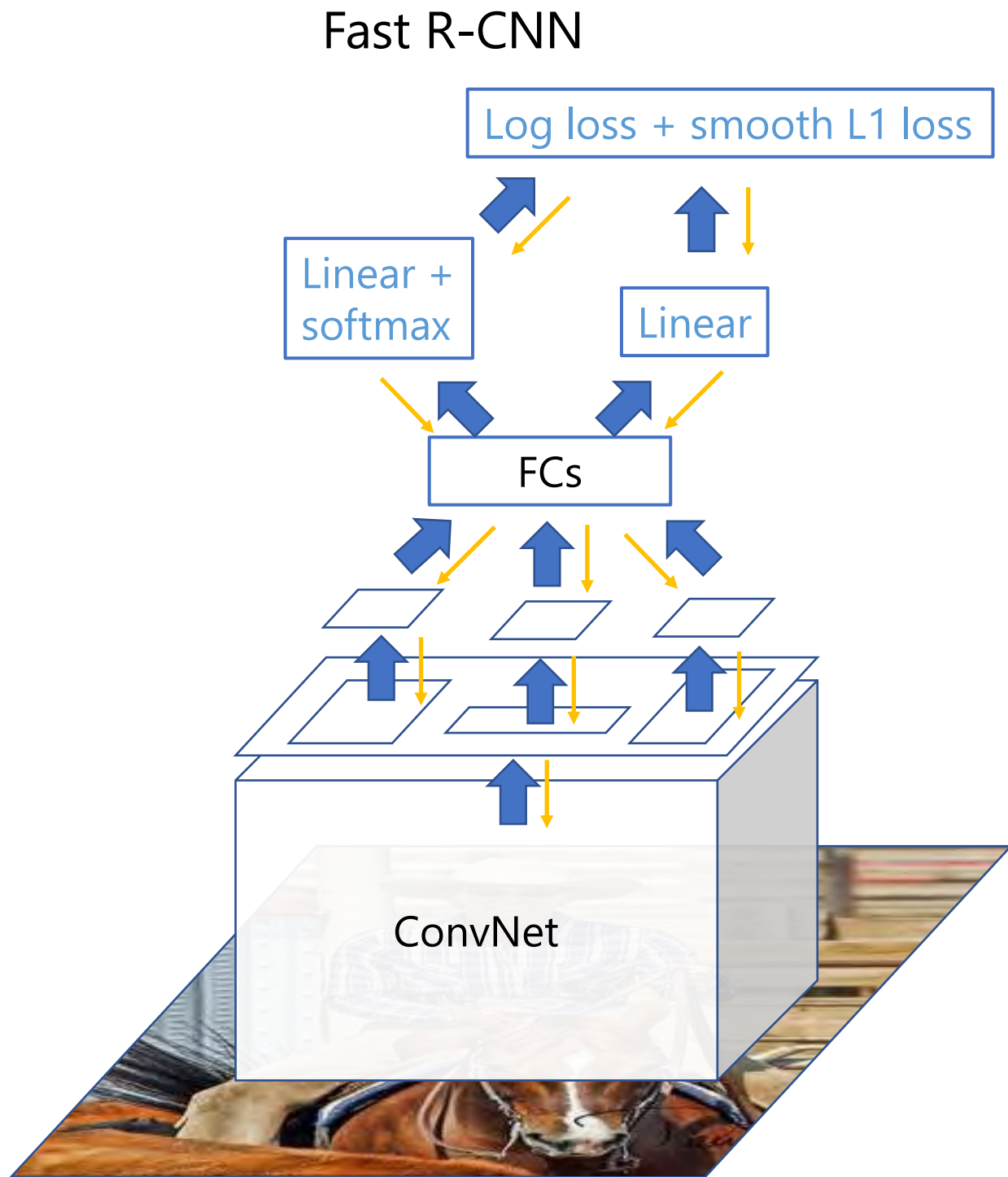


平面检测

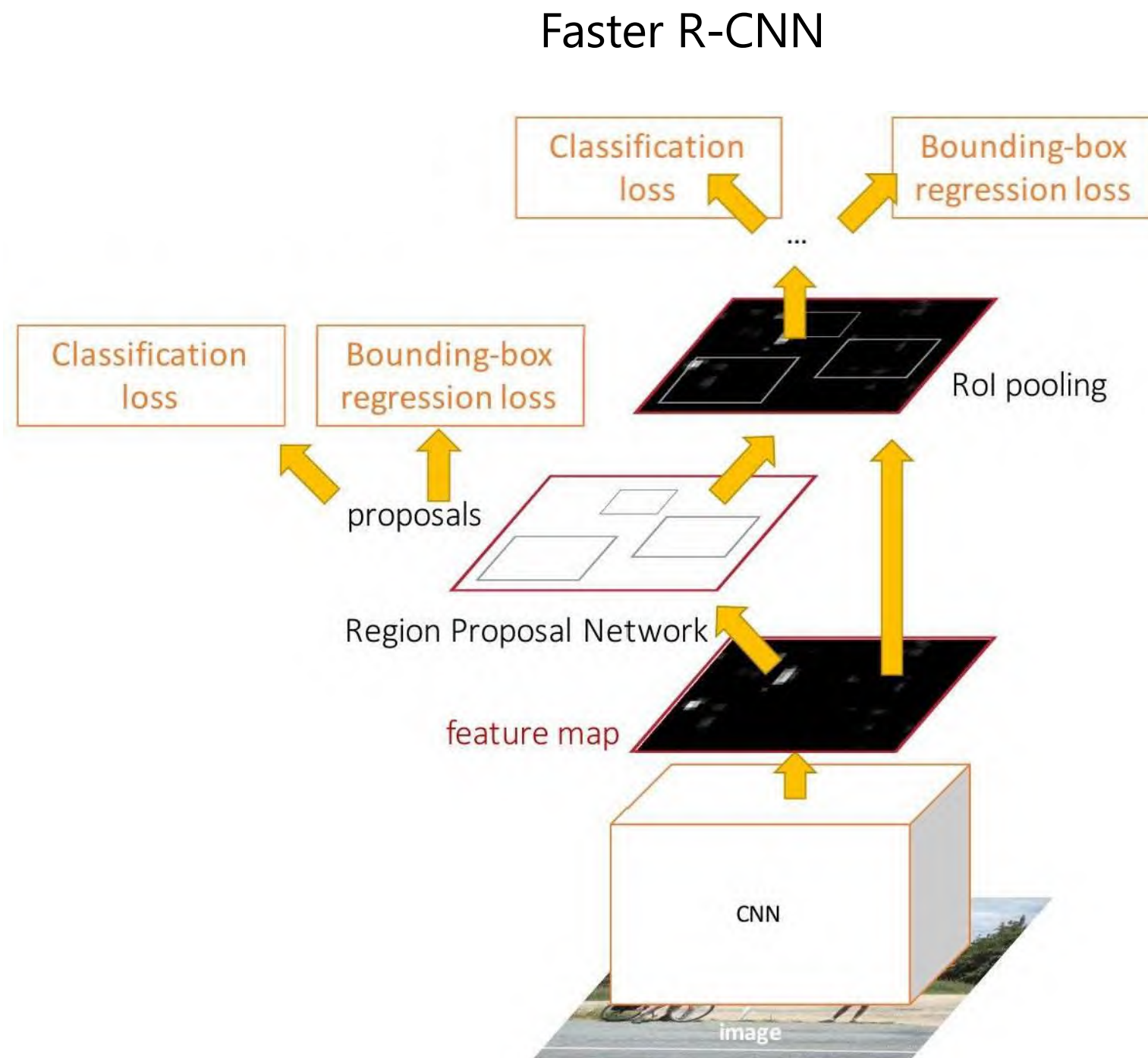


回顾——物体检测、实例分割

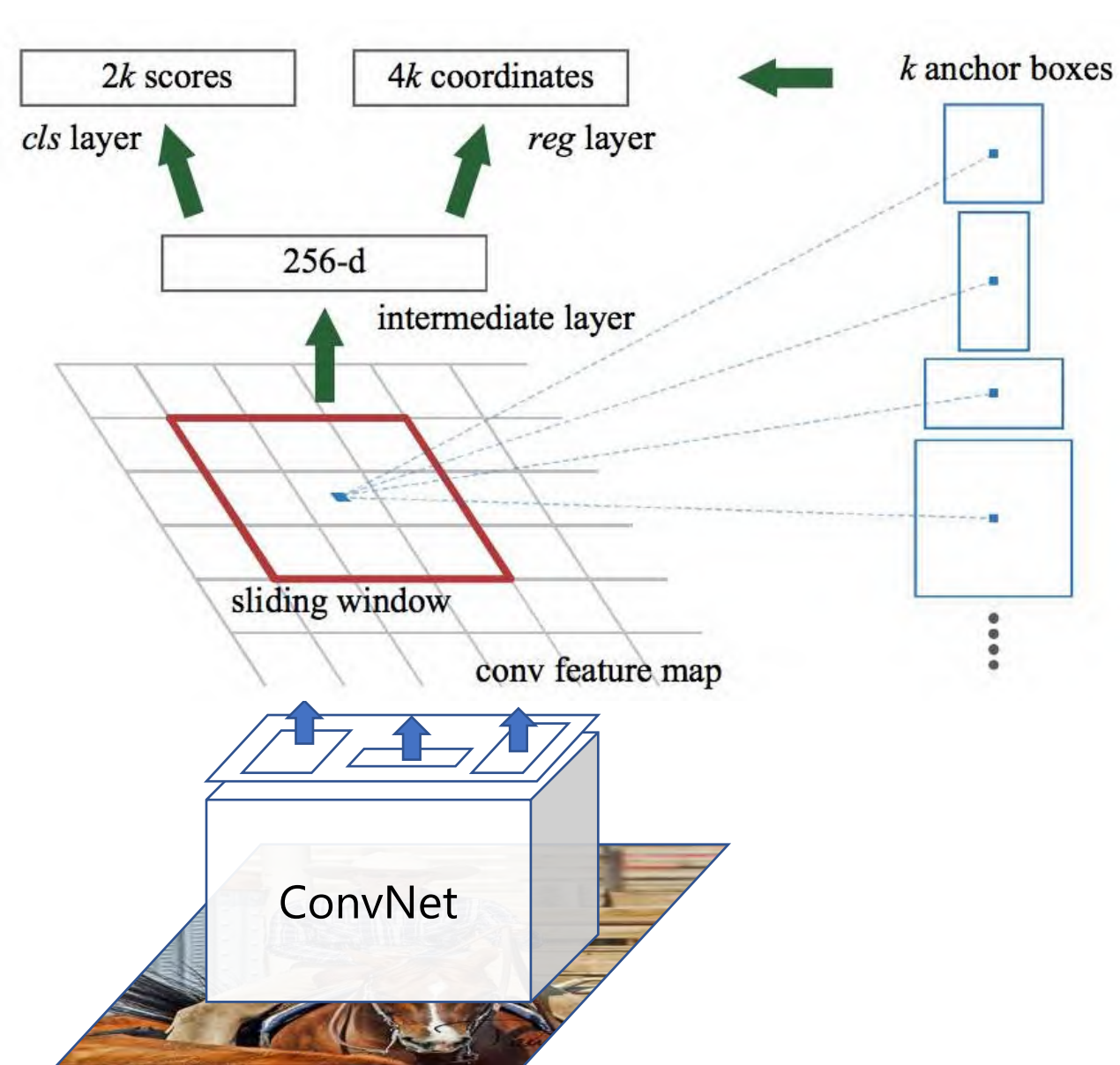
Faster R-CNN: 让CNN提Proposals!



Region Proposal Network (RPN) 预测区域proposal 其余的与Fast R-CNN一致



Region Proposal Network (RPN) 区域提议网络



基于conv5 特征的小网络模块.

预测:

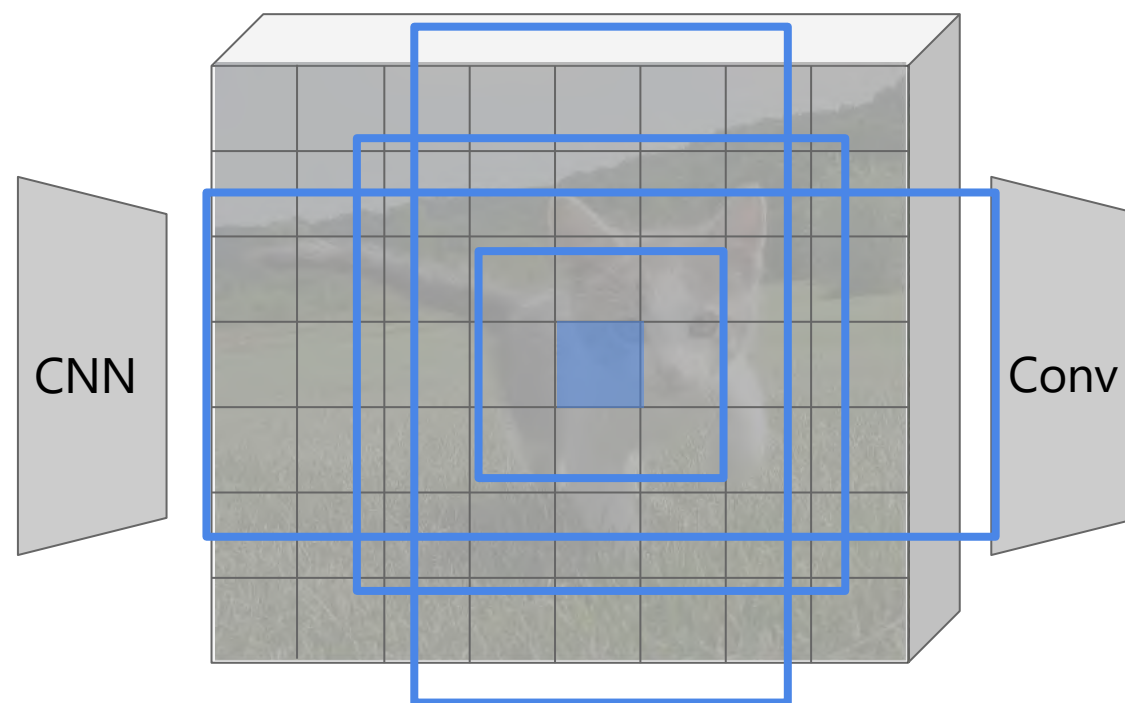
- 是否包括物体 (classification),
 - $IoU > 0.7$: 包含物体
 - $IoU < 0.3$: 不包含
 - 其他: 忽略
 - 区域是否需要校正 (regression)
- k 个根据特征图提取的 "anchors".

Region Proposal Network

K个anchor对应不同尺寸、尺度、形状



输入图像
(e.g. 3 x 640 x 480)



图像特征
(e.g. 512 x 20 x 15)

Anchor包含物体吗?
K x 20 x 15

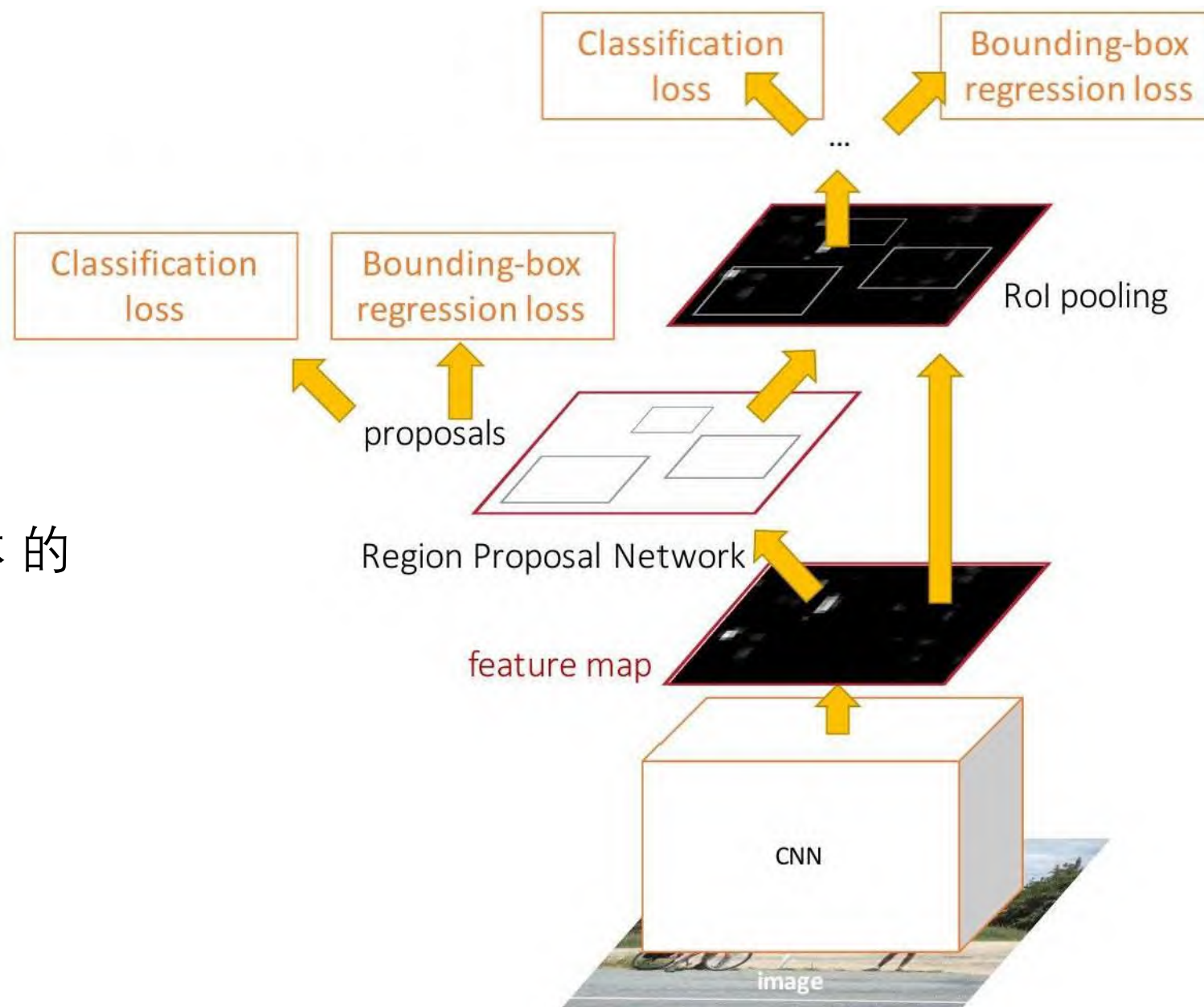
Anchor框校正
4K x 20 x 15

对K*20*15 个框“物体存在的概率”排序，取~300个进行预测，映射回原图

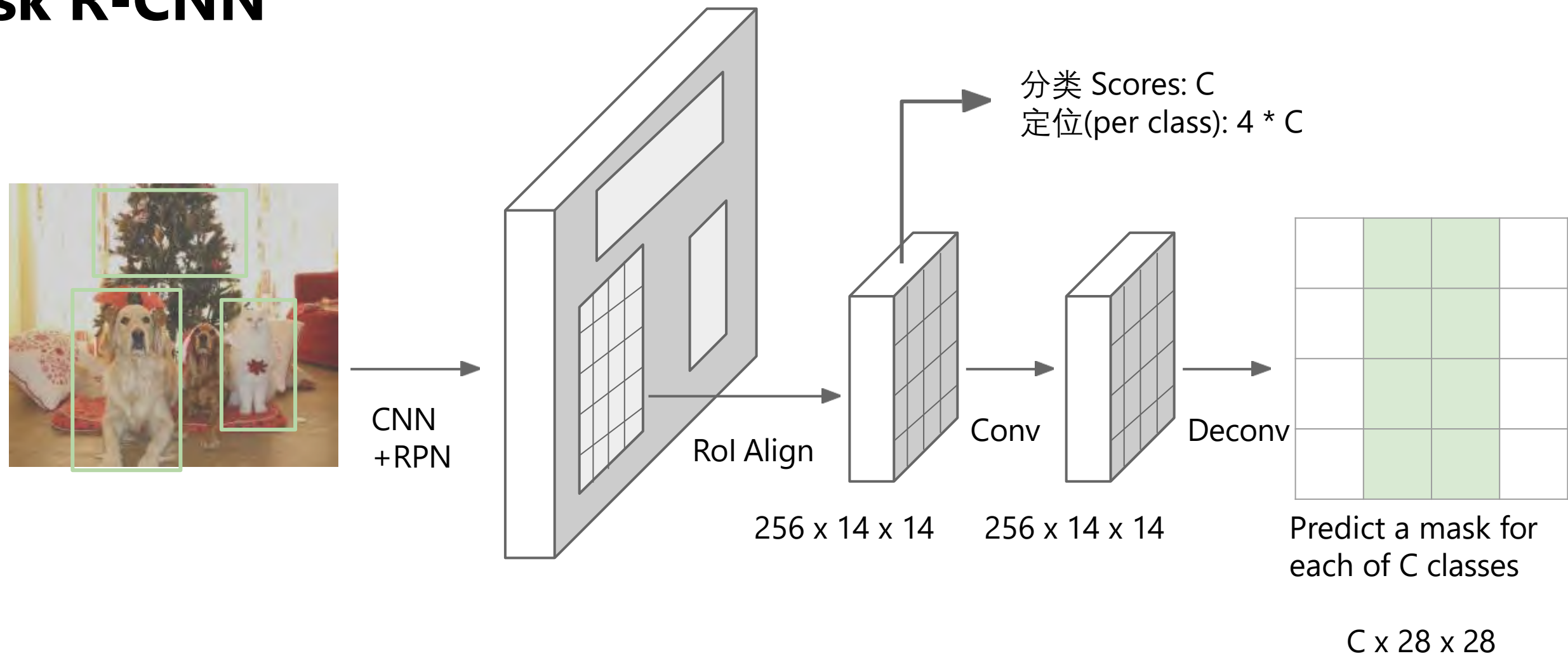
Faster R-CNN: 让CNN提Proposals!

大量的代码细节:

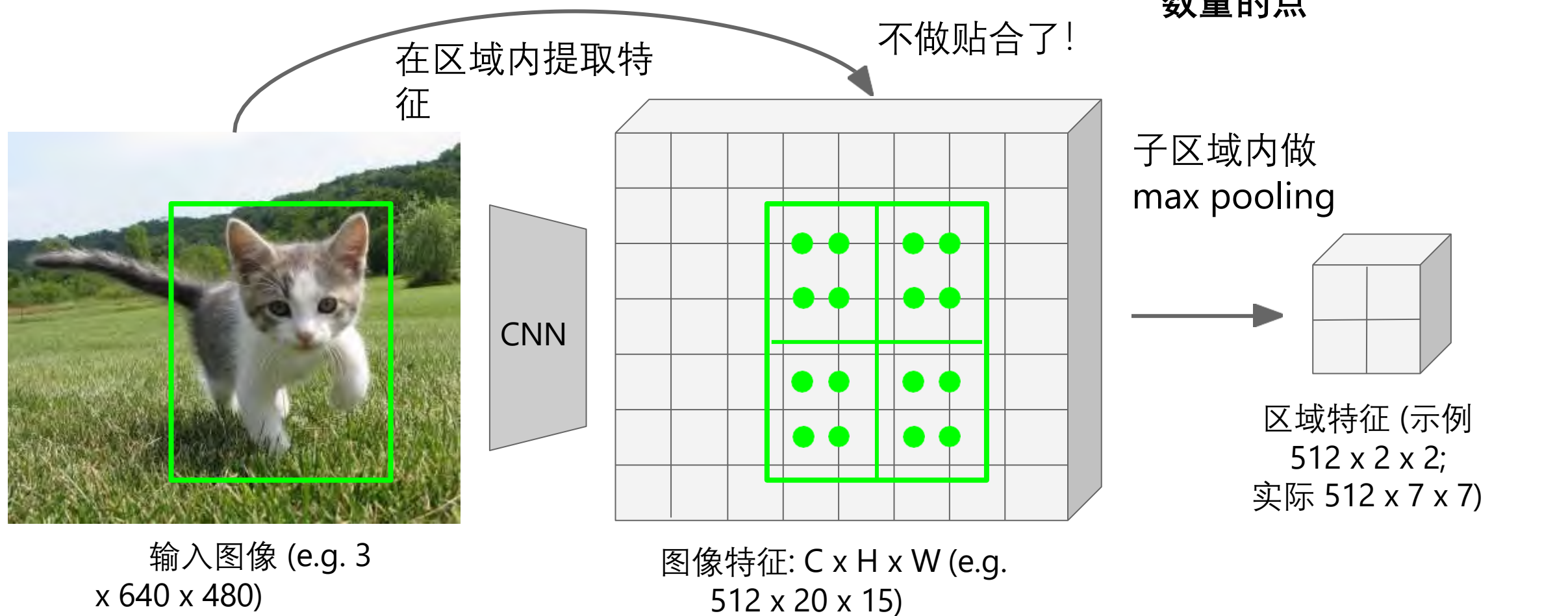
- 去掉重叠的proposal:
non-max suppression
- 多少个anchor?
- 采样多少个 物体/没有物体 的
proposal来训练?
- 怎么做物体框的回归?



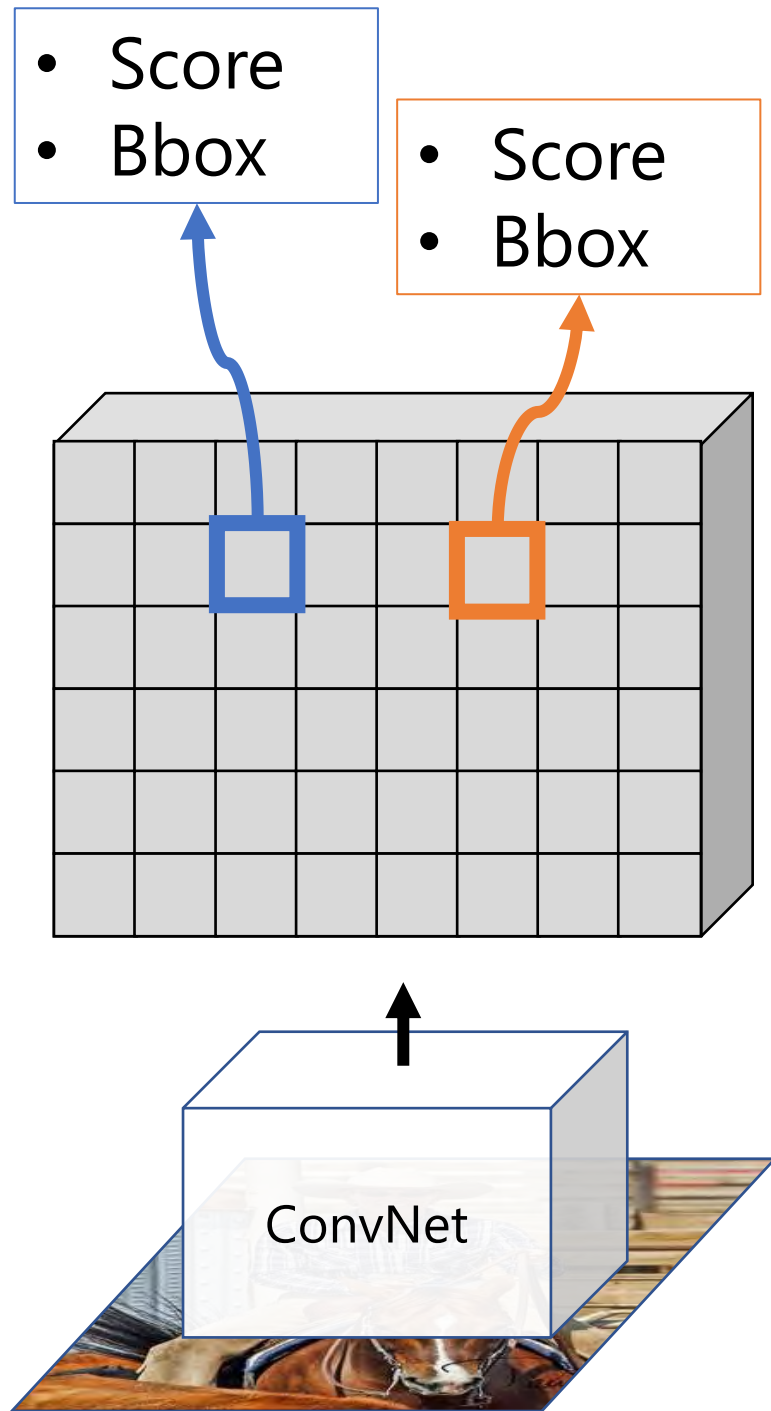
Mask R-CNN



RoI Pool升级版: RoI Align



YOLO

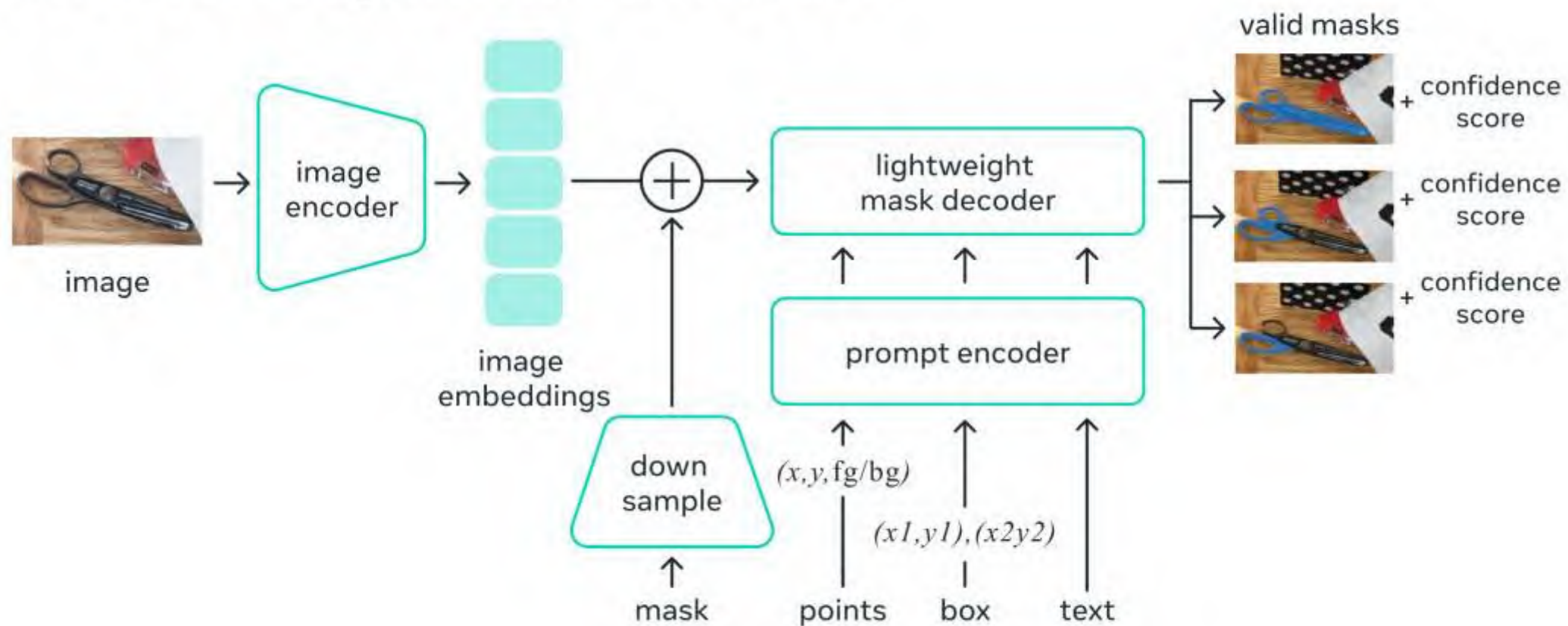


- 没有区域提议
- 对7x7的特征图上的每个点，预测分类评分+边界框校正
- 比 Faster-RCNN 快7倍，但精确度较低
- 由于其速度较快，在实际应用中，如机器人等，较常使用
- 众多版本

Segment Anything

针对任何提示返回有效的分割掩码

Universal segmentation model



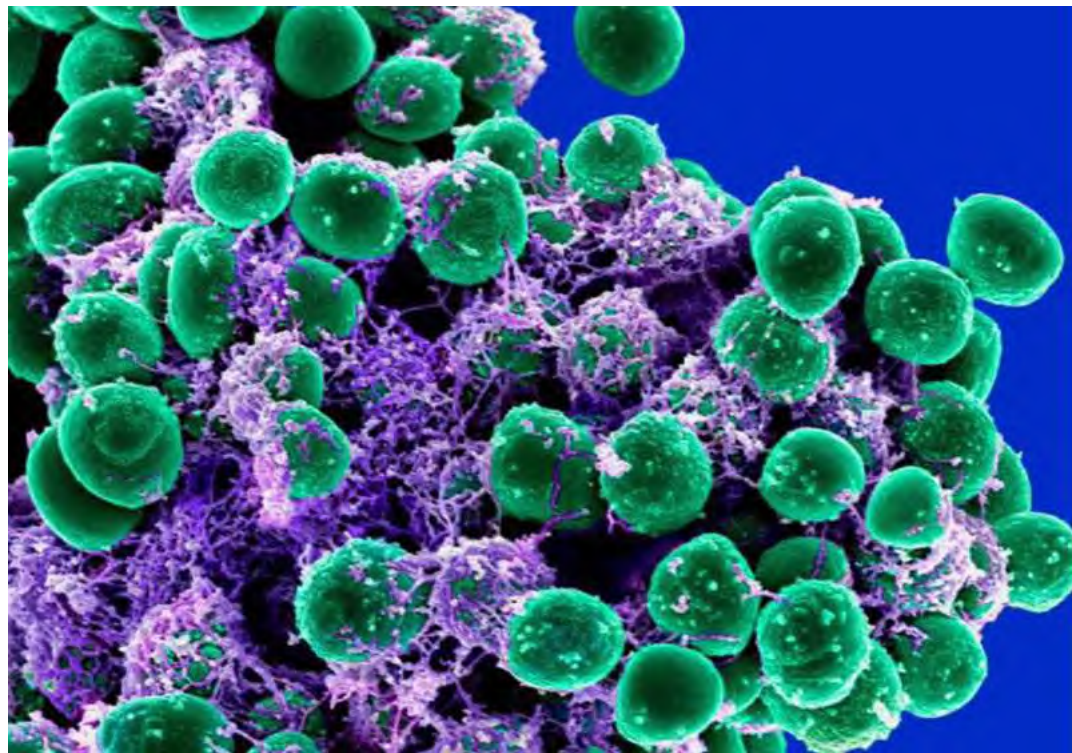
Segment Anything

“valid mask” 不同触发词对应的合理掩码




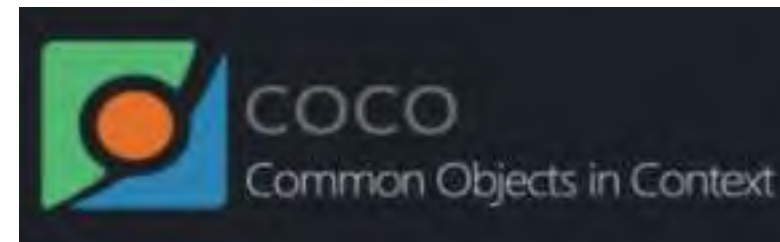
Segment Anything

零知识“zero-shot”预测



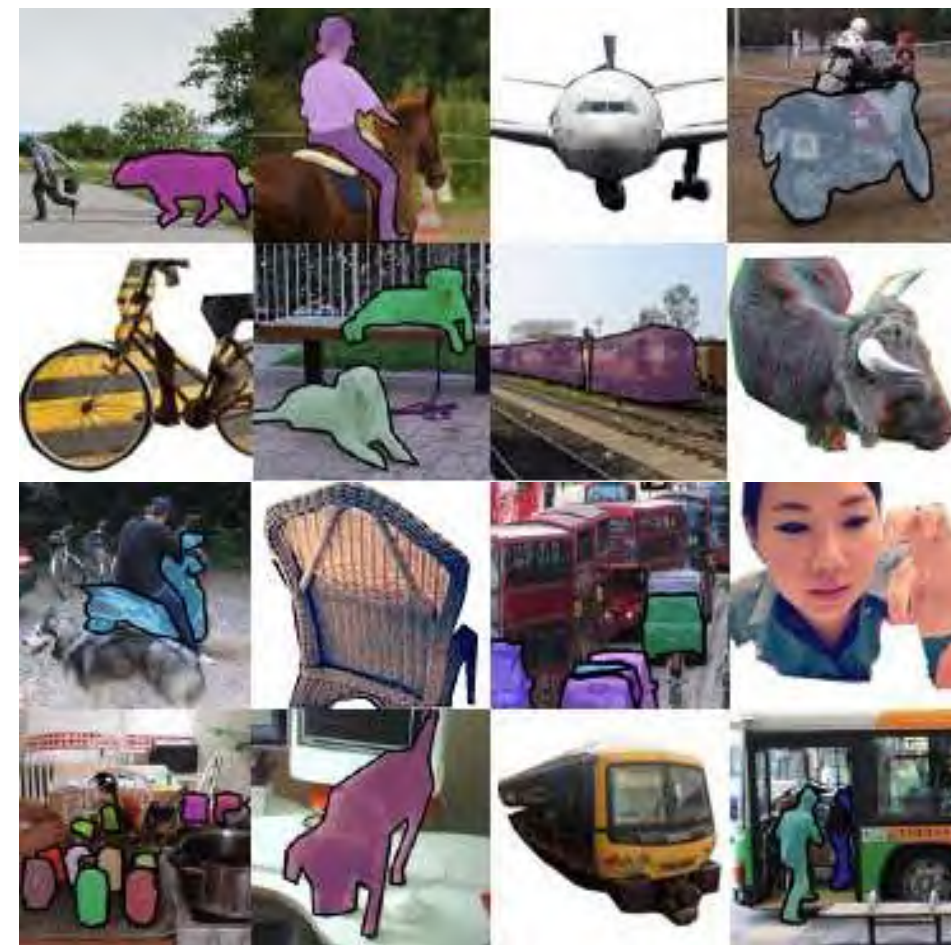
物体检测的新benchmark: COCO (2014)

- 80 类物体 (PASCAL 20)
- Current best mAP: 52%



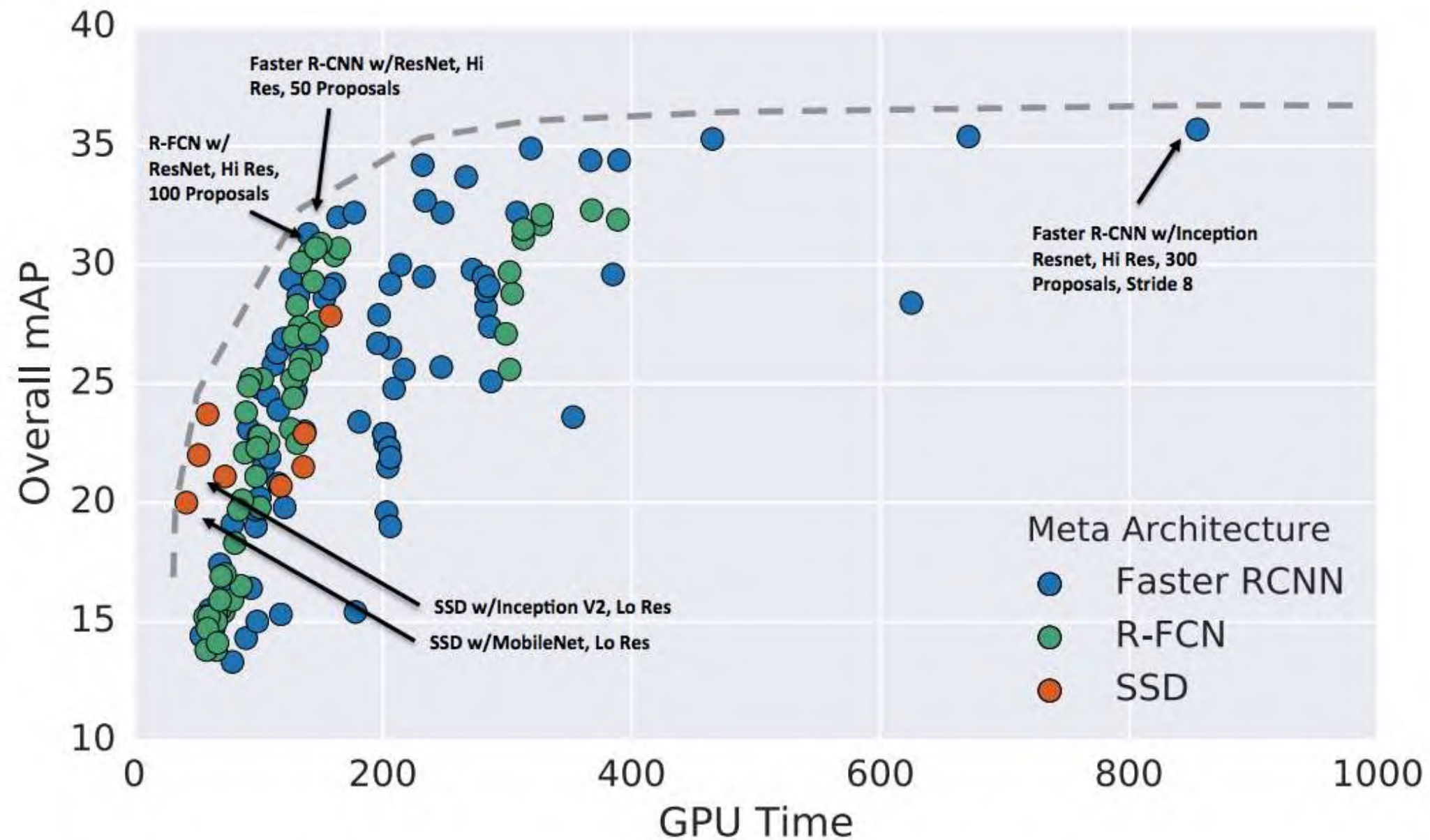
COCO is a large-scale object detection, segmentation, and captioning dataset. COCO has several features:

- ✓ Object segmentation
- ✓ Recognition in context
- ✓ Superpixel stuff segmentation
- ✓ 330K Images (>200K labeled)
- ✓ 1.5 million object instances
- ✓ 80 object categories
- ✓ 91 stuff categories
- ✓ 5 captions per image
- ✓ 250,000 people with keypoints



<http://cocodataset.org/#home>

物体检测的新benchmark: COCO (2014)



J. Huang et al., [Speed/accuracy trade-offs for modern convolutional object detectors](#), CVPR 2017

构建你的数据集。。。。

- 从网上爬取相关的照片
- Step 1: 用户拍摄一张照片
- Step 2: 用户决定上传该照片
- Step 3: 用户决定给这张图一个标签，比如“冰箱”，作为描述
- Step 4: 一个计算机视觉工作者把这张照片收录进了数据集



可是什么时候
我们会上传
一张冰箱的照片呢
?????



猜猜类别

这些是在原数据集的精度高达99%的模型, 以 $>90\%$ 置信度, 预测出来的边界框



(1) Person



(2) Giraffe

(3) Bicycle

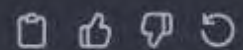
J You



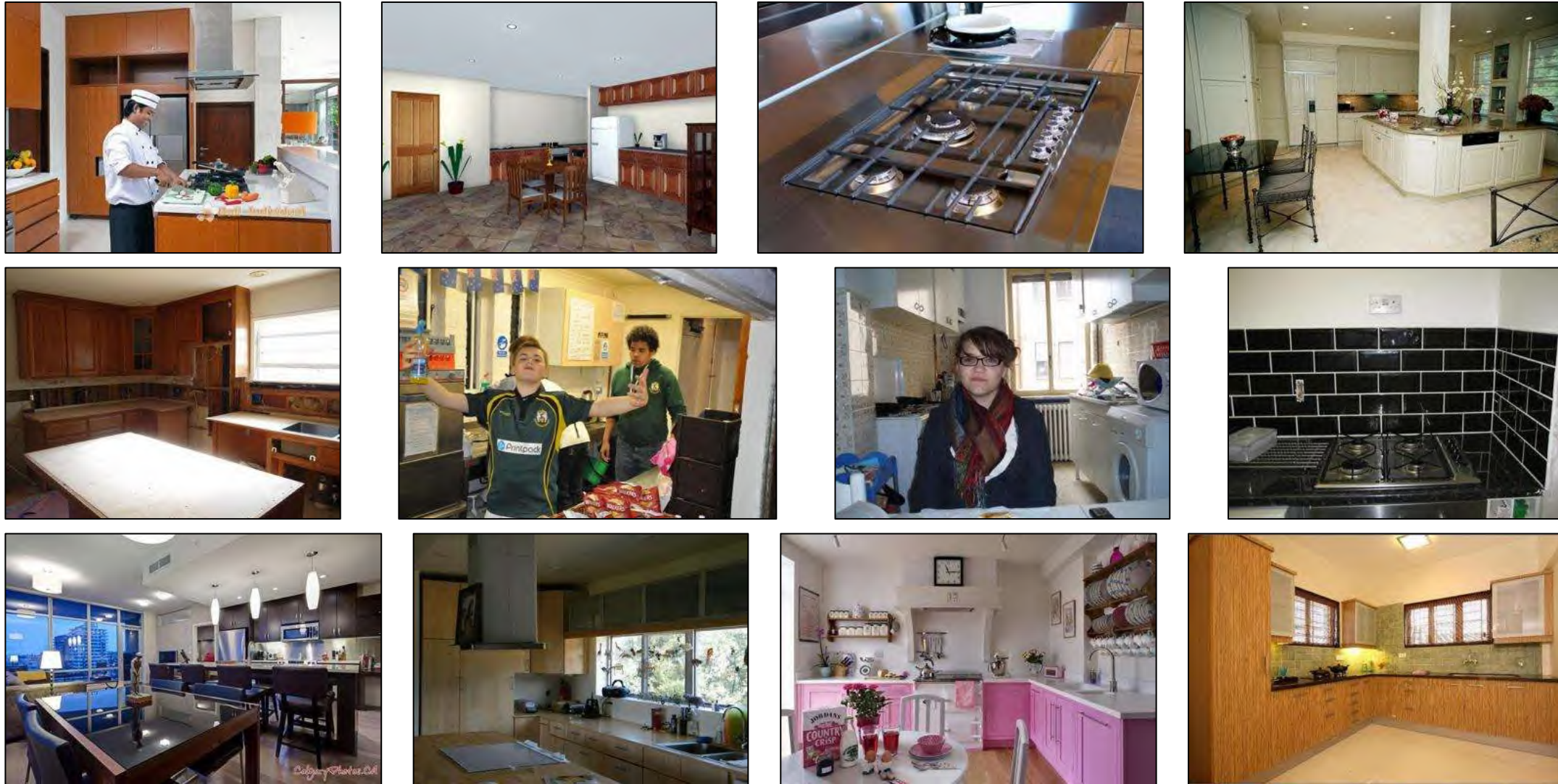
红框里面是什么?是人,长颈鹿,还是自行车?

ChatGPT

红框里面的一个穿着豹纹图案衣服的人的手臂。这不是长颈鹿，也不是自行车。图案可能会让人联想到长颈鹿，但从形状和结构上可以判断这是一个人的手臂。



搜索引擎给出的图像：厨房



物体检测能帮我们做什么？

Imaging Setup



Instance Segmentation



Skull



36.08 mm

Femur

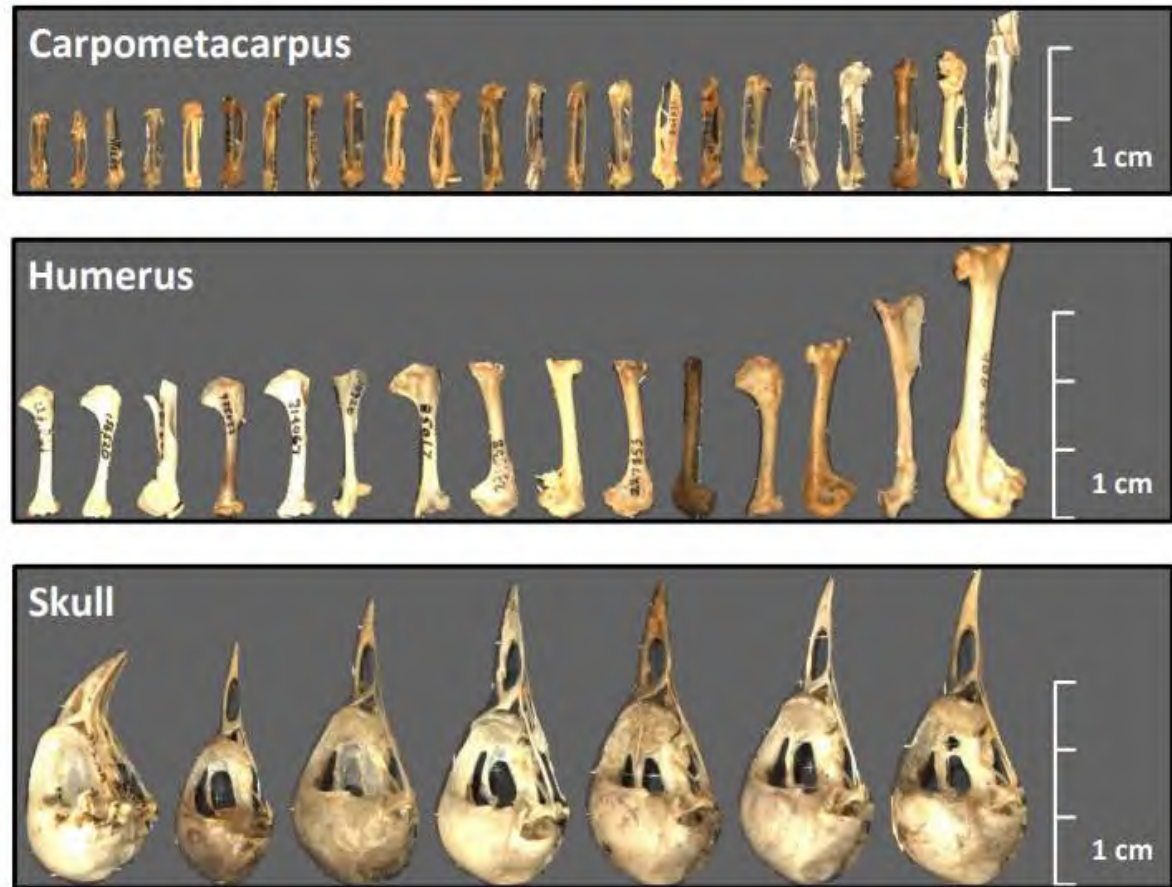


18.97 mm

Z. Zhou, G. Hassena, B.C. Weeks, D.F. Fouhey. *Quantifying Bird Skeletons*. CVPR Workshops, 2021.

B.C. Weeks, Z. Zhou, B.K. O'Brien, R. Darling, M. Dean, T. Dias, G. Hassena, M. Zhang, D.F. Fouhey. *A deep neural network for high throughput measurement of functional traits on museum skeletal specimens*. *Methods in Ecology and Evolution*, 2022.

物体检测能帮我们做什么？

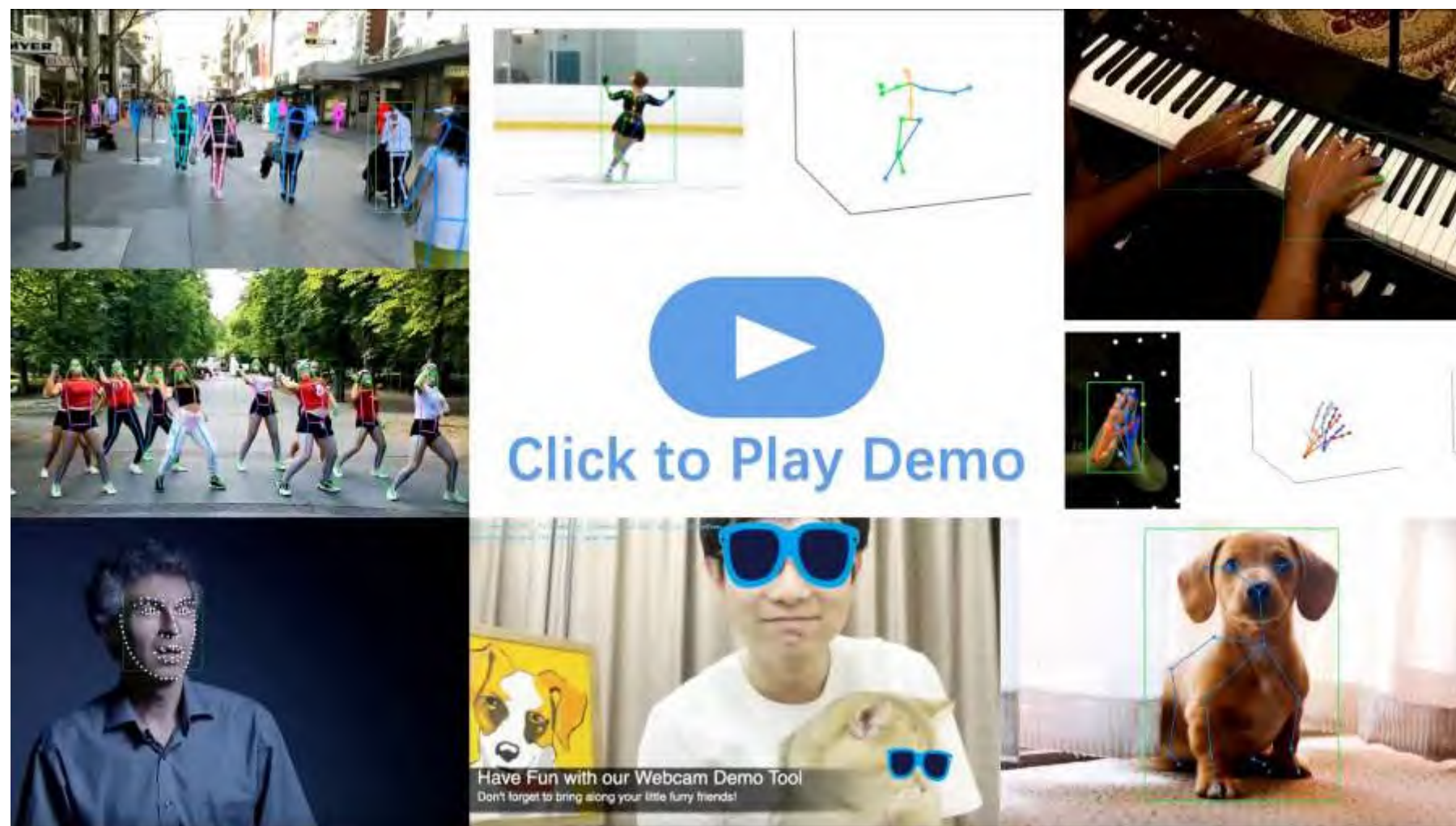


- UM 数据集有 25K 鸟类骨骼图像，人工标注十分耗时耗力.
- 新方法：平铺这些图像，用模型预测.
- 我们可以大规模地去做这一类任务

Z. Zhou, G. Hassena, B.C. Weeks, D.F. Fouhey. *Quantifying Bird Skeletons*. CVPR Workshops, 2021.

B.C. Weeks, Z. Zhou, B.K. O'Brien, R. Darling, M. Dean, T. Dias, G. Hassena, M. Zhang, D.F. Fouhey. *A deep neural network for high throughput measurement of functional traits on museum skeletal specimens*. *Methods in Ecology and Evolution*, 2022.

姿态估计

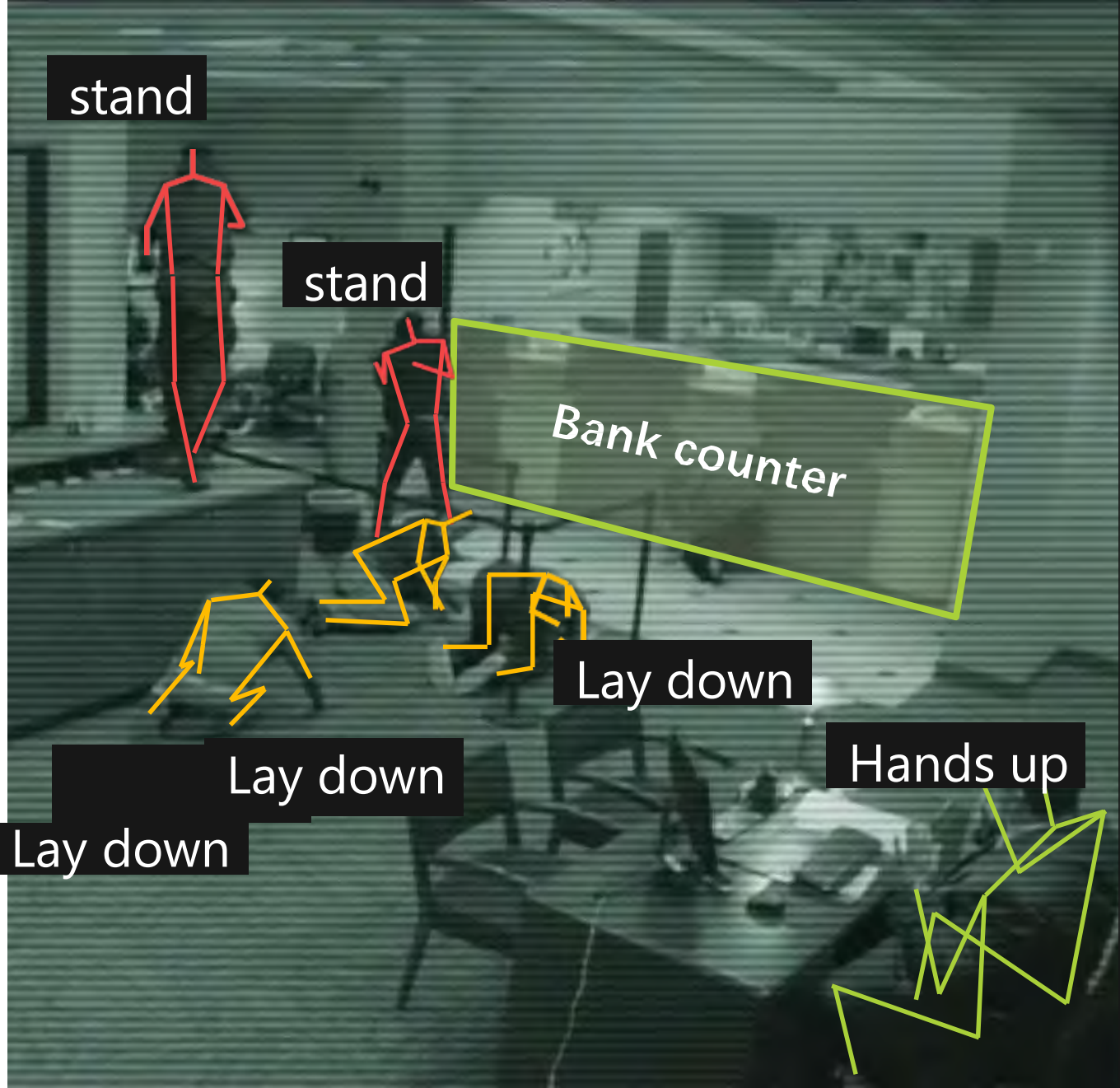


应用：动作捕捉与驱动





如何识别这个场景发生了什麼？



Cue

S

Scene: bank

Abnormal

pose

Activity:

robbery

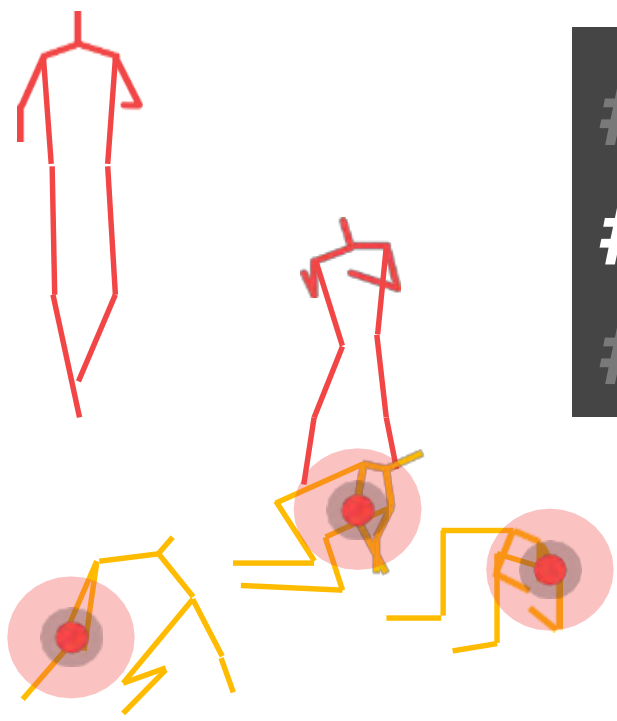


#1. Articulation

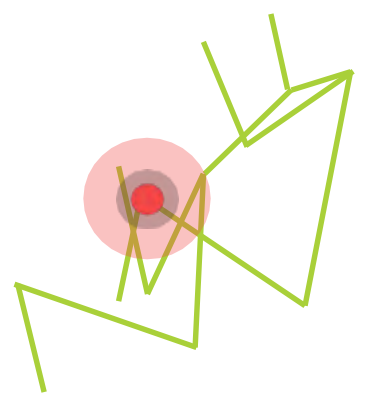
#2. Occlusion

#3. Scale variation



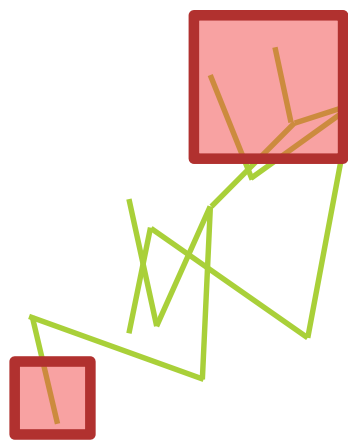


- #1. Articulation
- #2. Occlusion
- #3. Scale variation





#1. Articulation
#2. Occlusion
#3. Scale variation



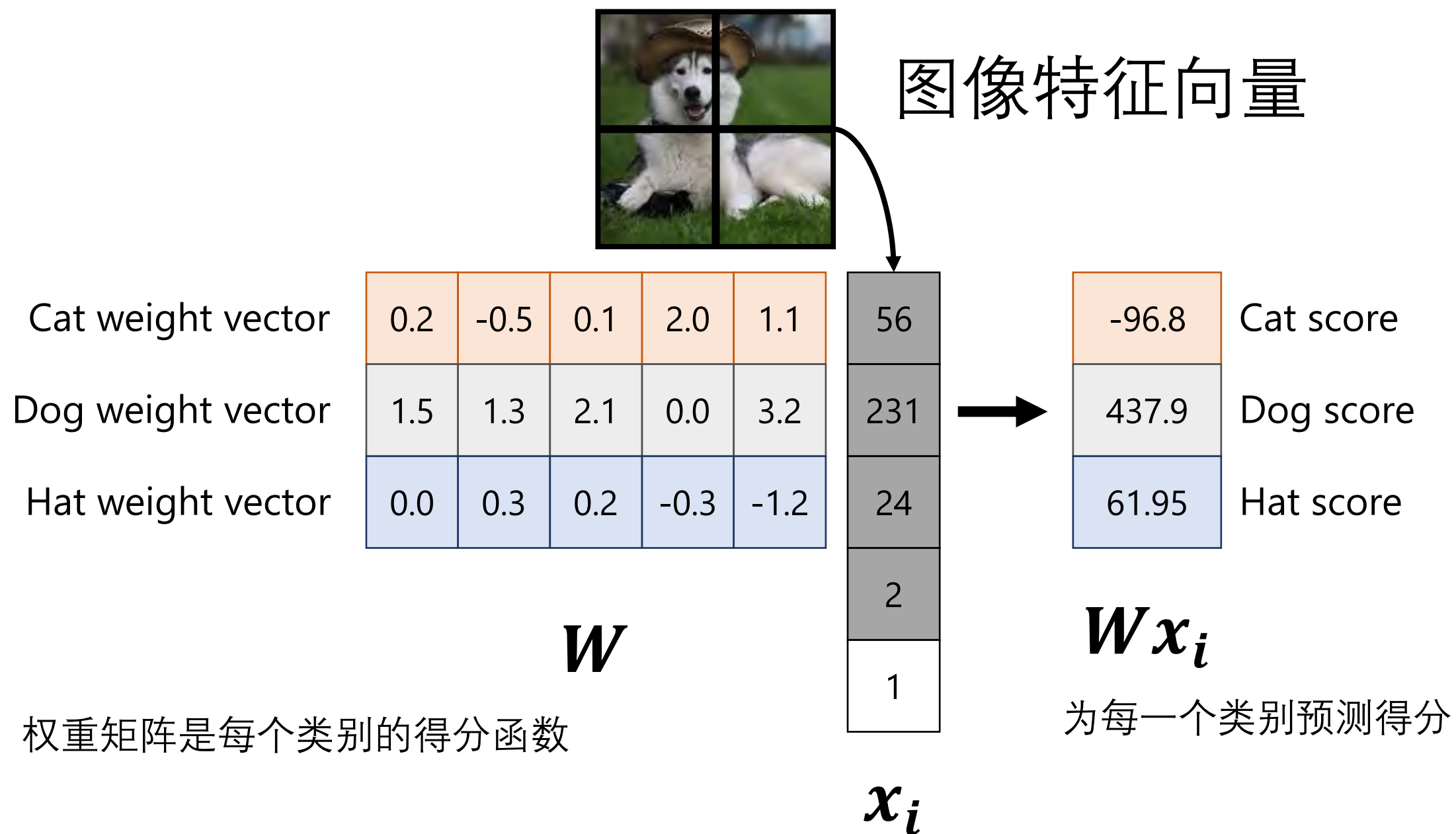
RNN、Transformer 和 ViT

这是啥？



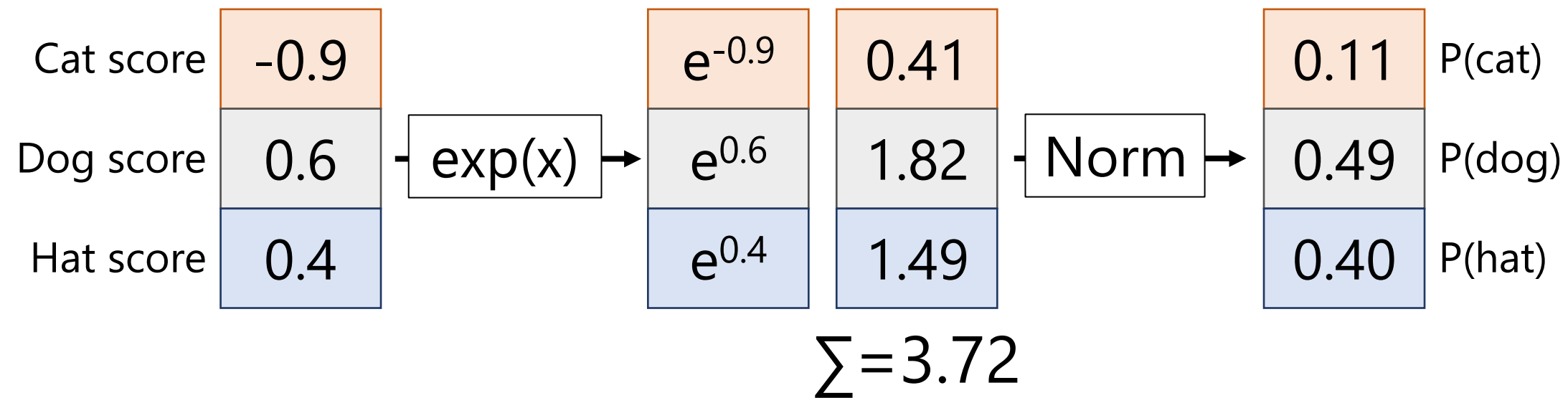
Dog image credit: T. Gupta

回顾: 图像分类



回顾: 图像分类

Softmax把得分转化为概率



P(class j):

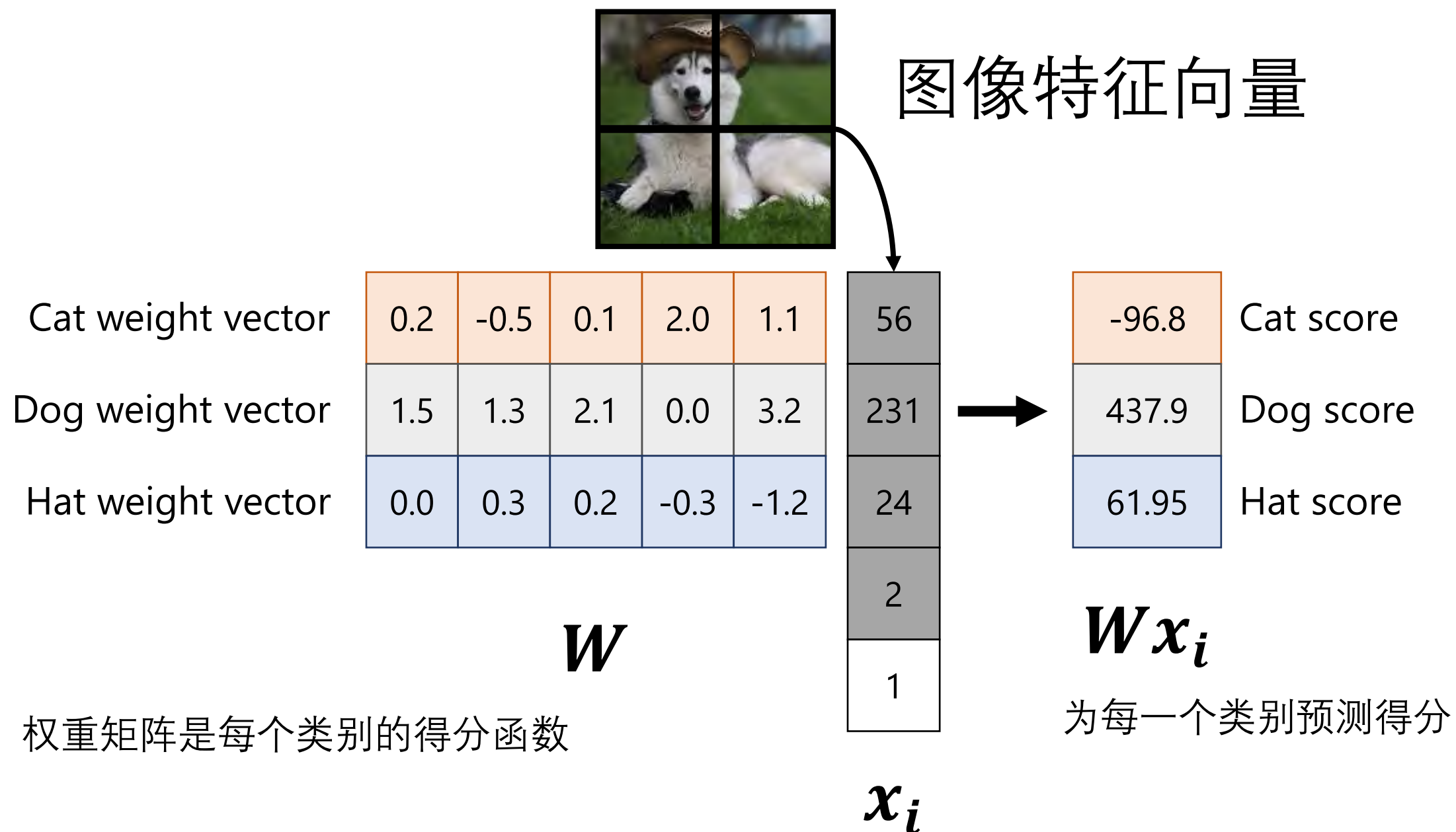
$$\frac{\exp((Wx)_j)}{\sum_k \exp((Wx)_k)}$$

这样有什么问题?



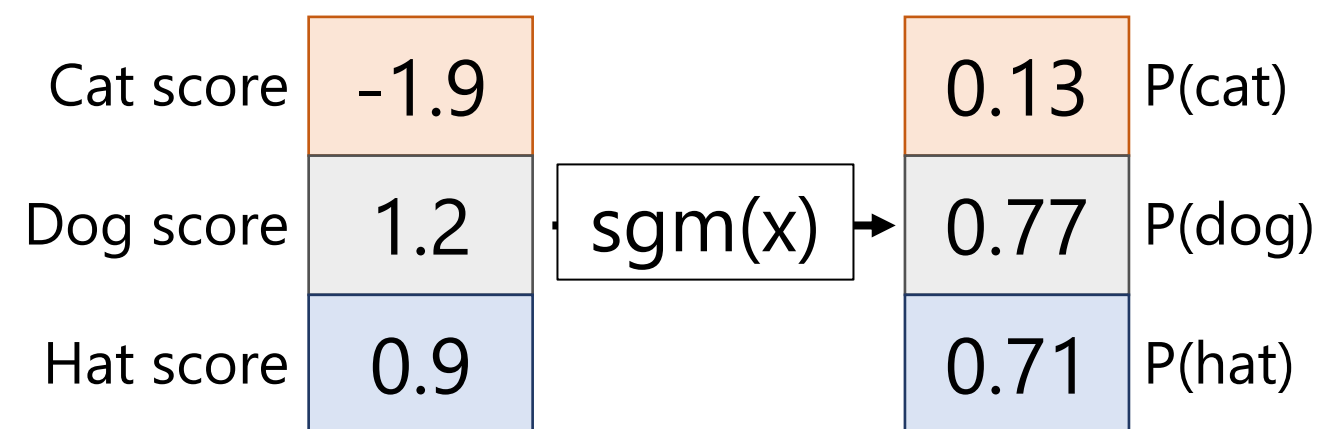
这该标注为狗?还是帽子?

多分类



多分类

用Sigmoid把得分转化为概率



77% dog
71% hat
13% cat?

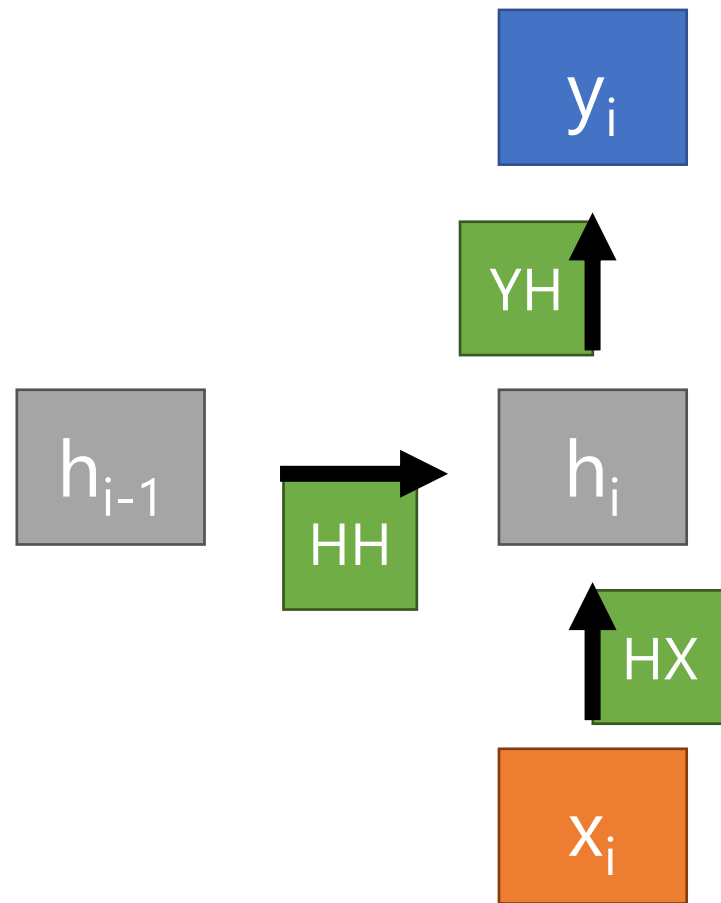
那么...

- 我们如果要做: “dog with a hat” 或者 “husky wearing a hat” 等
- 最直接的方式(给定 N 个单词,选中其中至多 C 个). 有多少个方案?
- $N=10k, C=5 \rightarrow 10$ 的20次方
- 我们不可能训练一个分类器来实现这个任务

换个方式: 序列模型

- 建立N个单词的字典
- 新的目标:顺序去做C个N分类任务
- 字典应该具备:
 - 训练集中所有可能出现的单词
 - 包括一些特殊“单词”: START, END, UNK

Sequence Modeling 序列模型



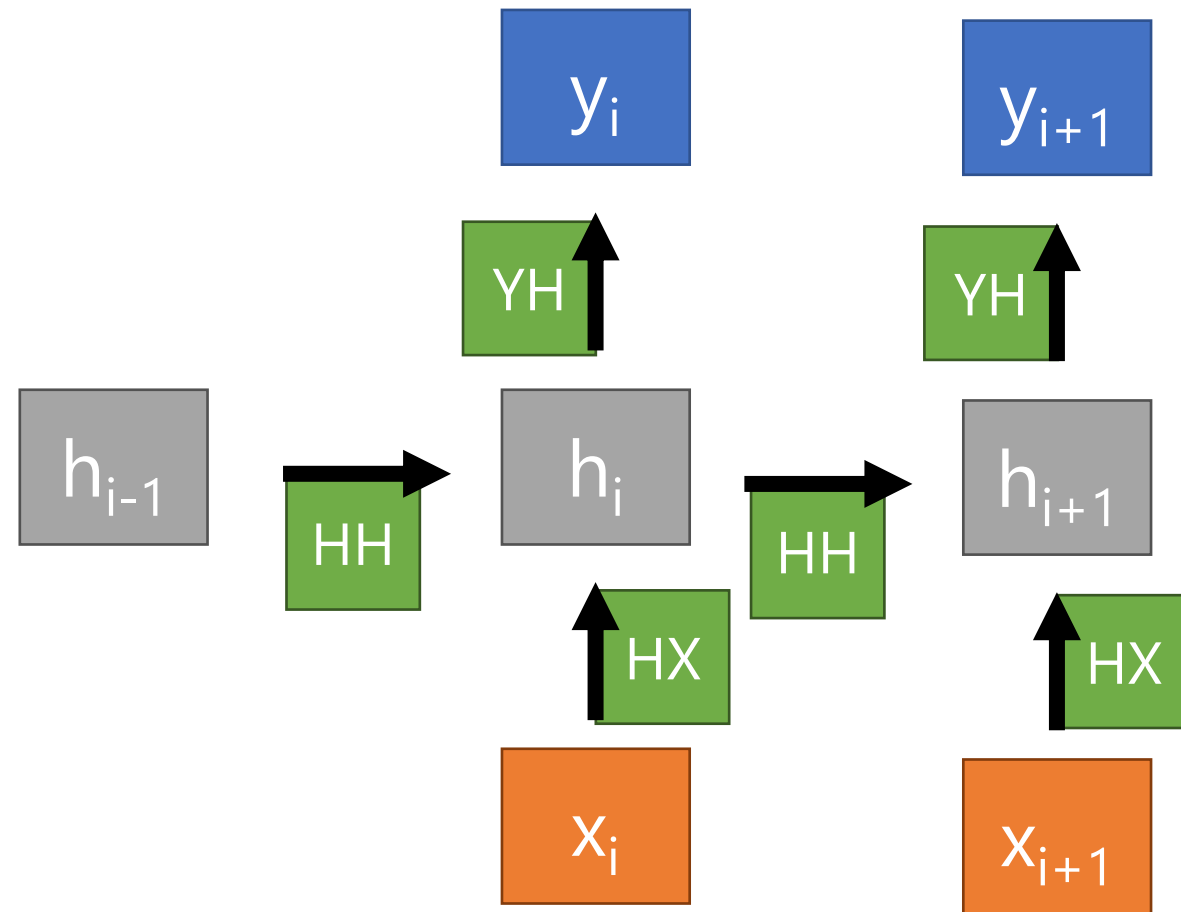
第 i 个输出 y_i 是 隐藏状态 h_i 的线性变换

$$y_i = W_{yh} h_i$$

隐藏状态 h_i 基于输入 x_i ，上一步的隐藏状态 h_{i-1} ，加上非线性层

$$h_i = \sigma(W_{hx}x_i + W_{hh}h_{i-1})$$

Sequence Modeling 序列模型

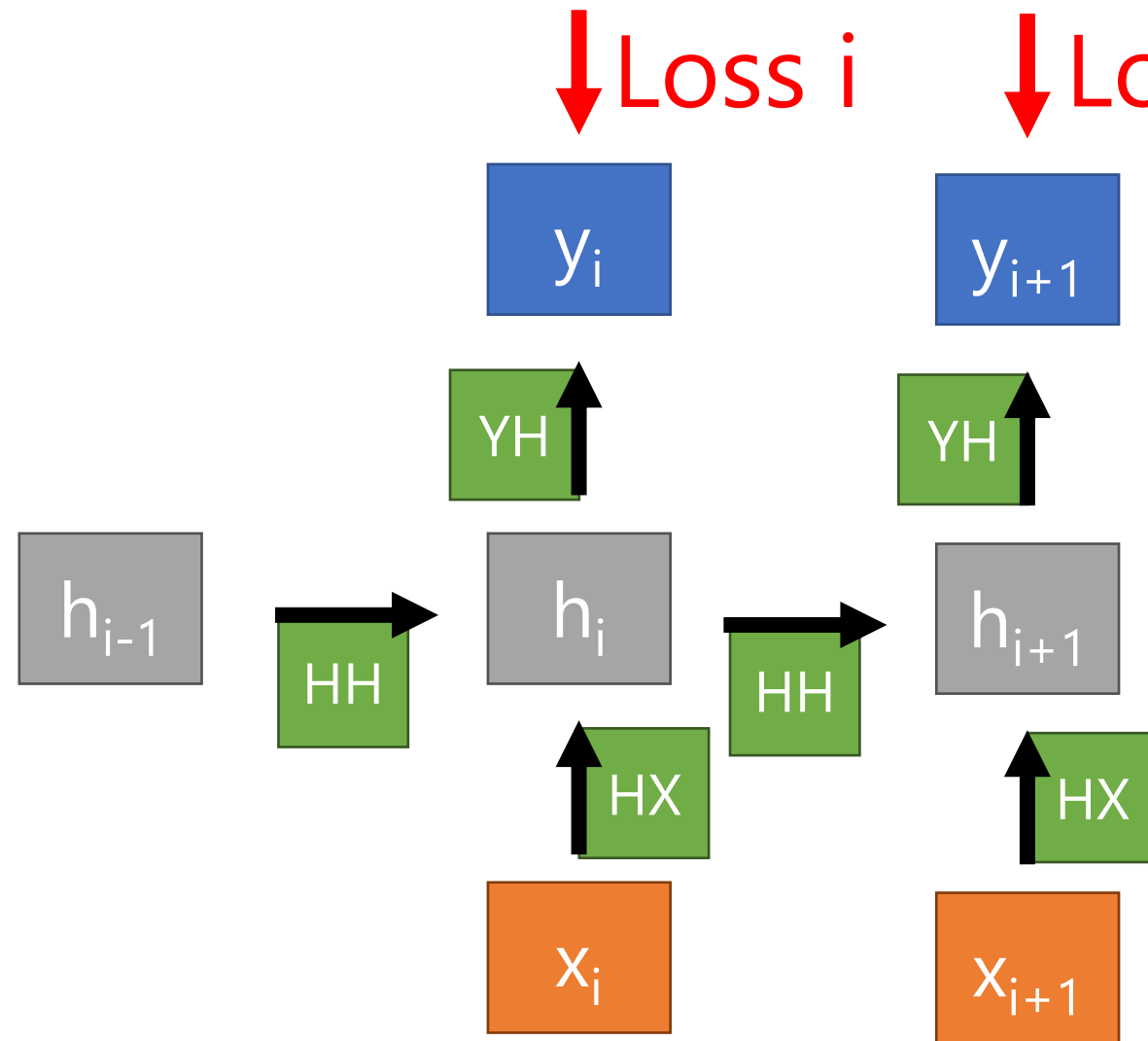


可以将多个RNN层叠加在一起，以处理更复杂的序列数据。在这种结构中，每一层的输出可以作为下一层的输入，允许网络学习更深层次的特征。模型可以用参数表达： W_{HX} , W_{HH} , W_{YH}

$$y_i = W_{yh} h_i$$

$$h_i = \sigma(W_{hx}x_i + W_{hh}h_{i-1})$$

Sequence Modeling 序列模型



可以针对每个输出定义损失，并相对于所有权重进行微分

可以量化每个时间步的预测与实际值之间的误差，并通过时间上的反向传播优化网络

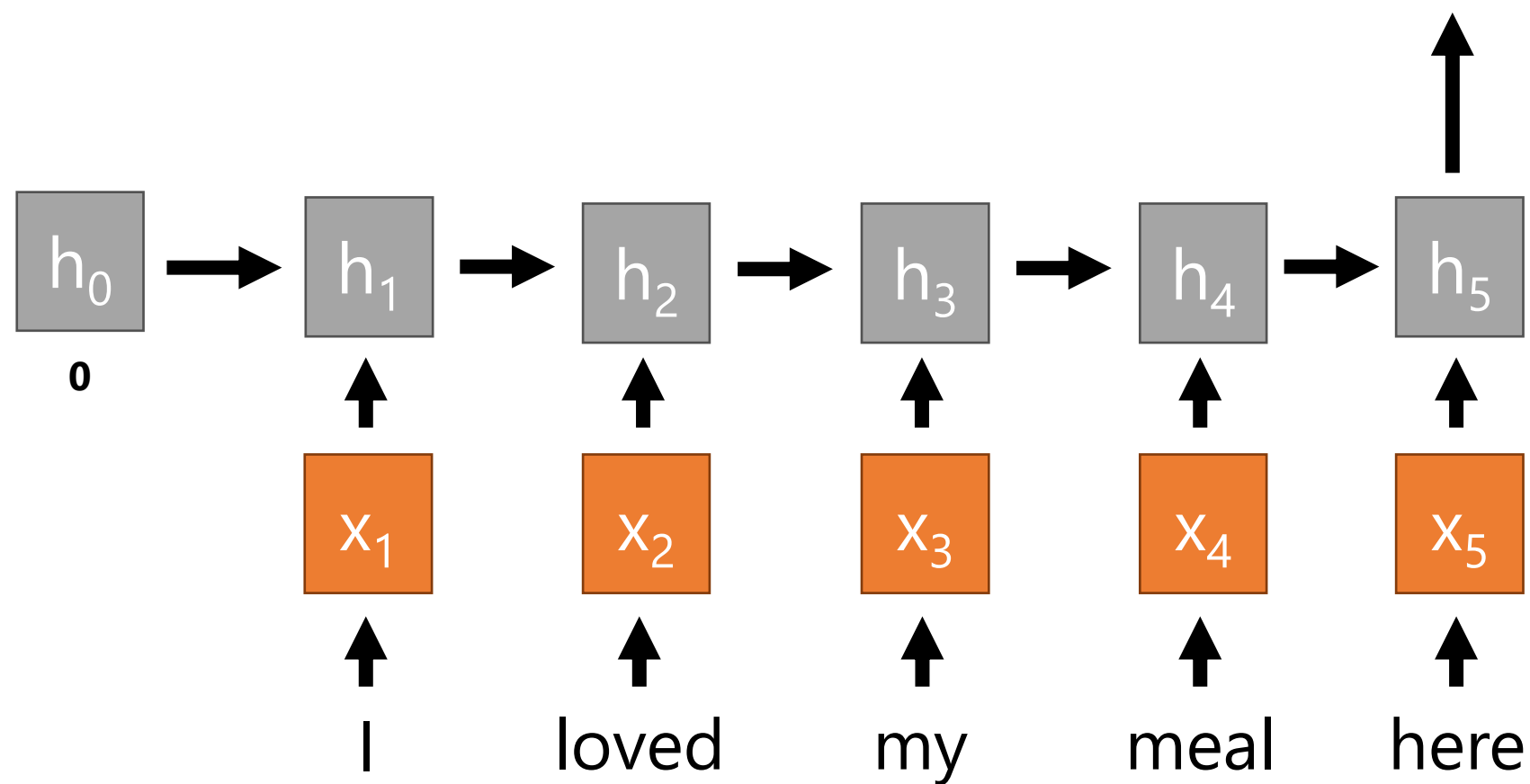
$$y_i = W_{yh} h_i$$

$$h_i = \sigma(W_{hx}x_i + W_{hh}h_{i-1})$$

序列模型

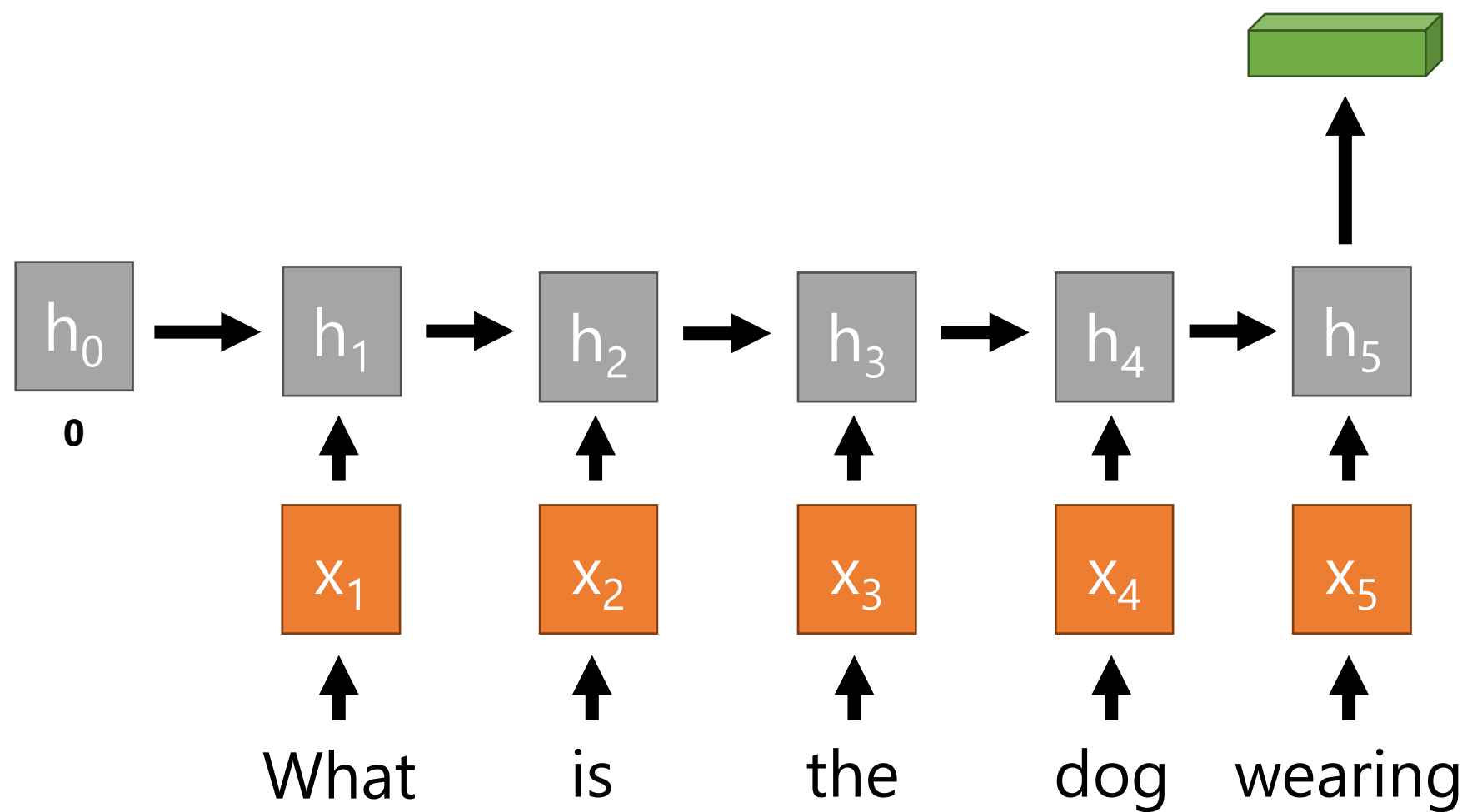
多个输入， 单个输出

评价分数



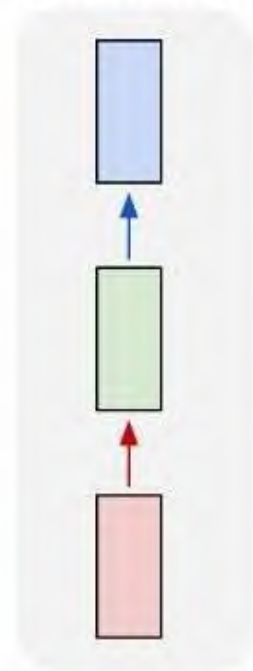
序列模型

可以是一个特征



处理序列数据——不同的任务

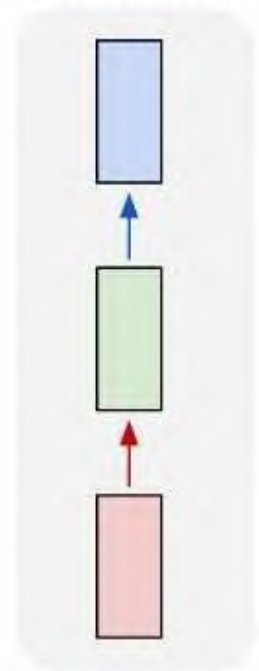
one to one



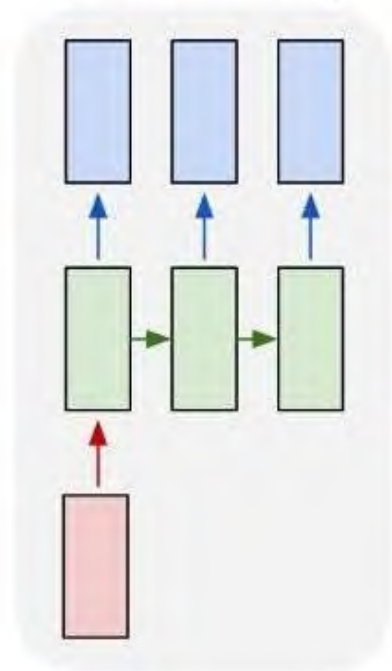
Vanilla Neural Networks

处理序列数据——不同的任务

one to one



one to many

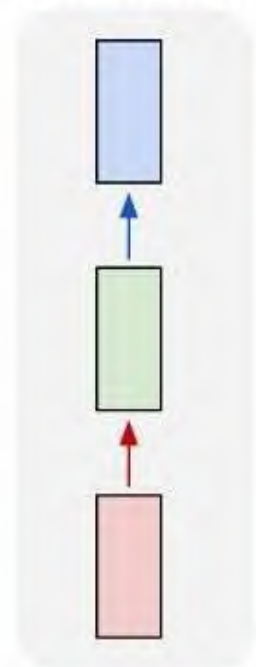


e.g. 图文描述

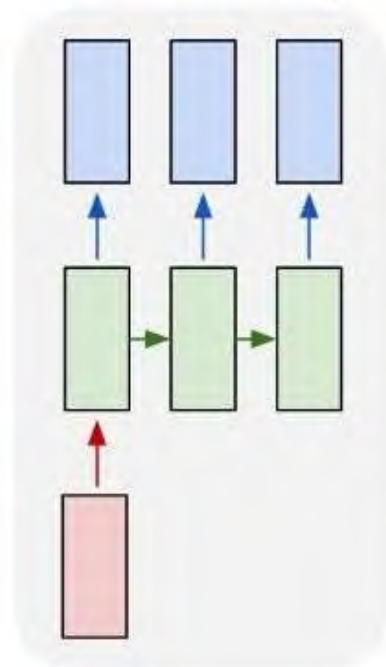
image -> sequence of words

处理序列数据——不同的任务

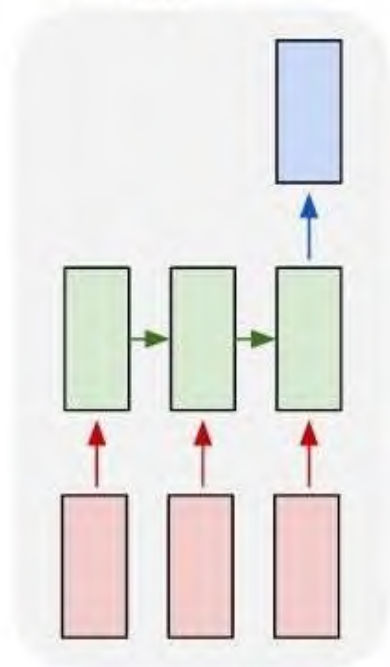
one to one



one to many



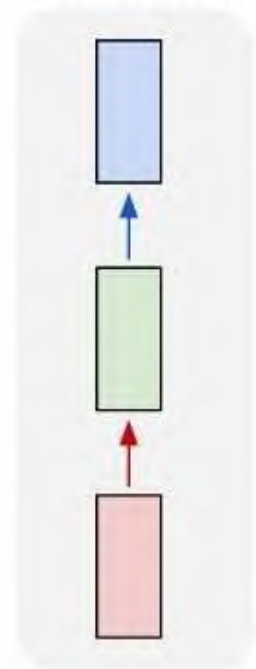
many to one



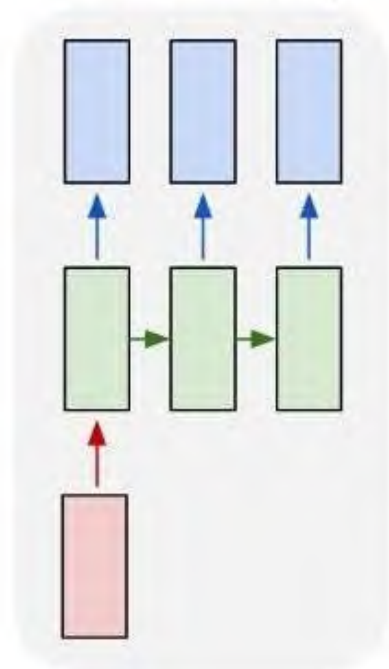
e.g. 动作/事件预测
sequence of video frames -> action class
用户评价打分
Review -> Scores

处理序列数据——不同的任务

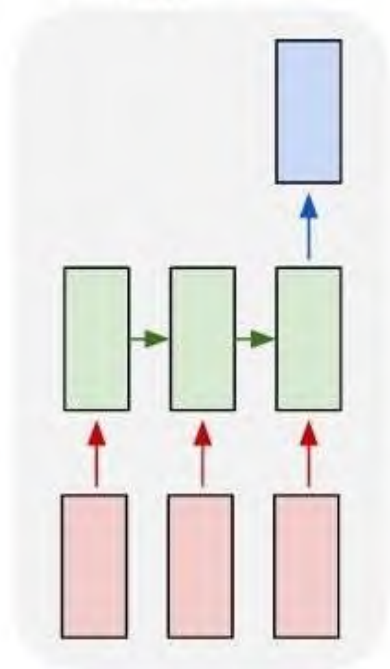
one to one



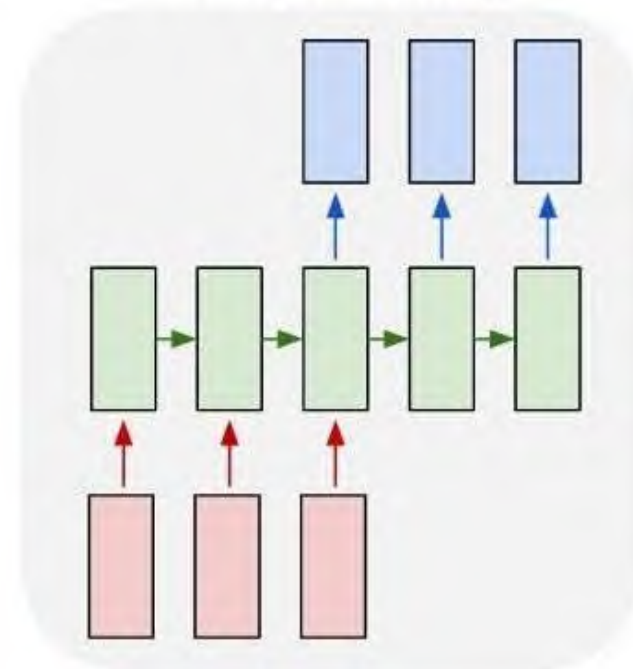
one to many



many to one



many to many

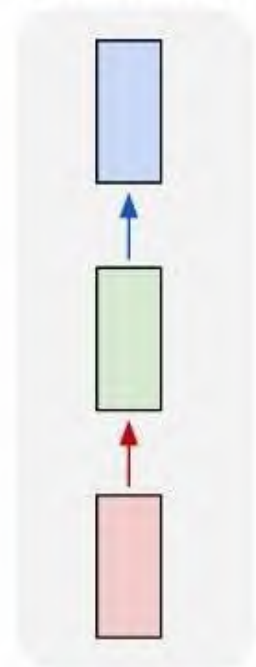


E.g. 视频配文

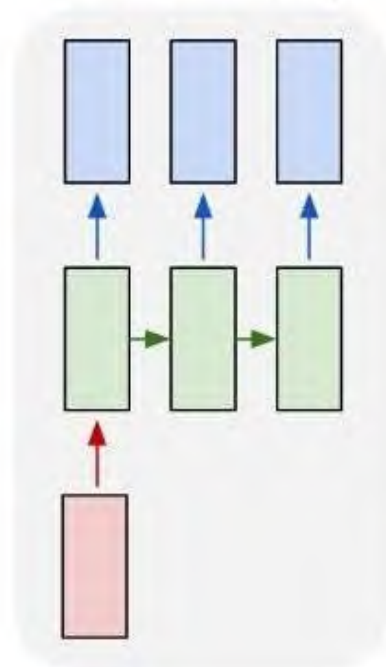
Sequence of video frames -> caption

处理序列数据——不同的任务

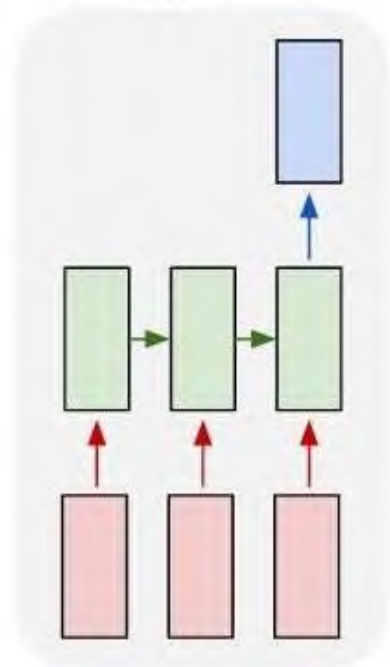
one to one



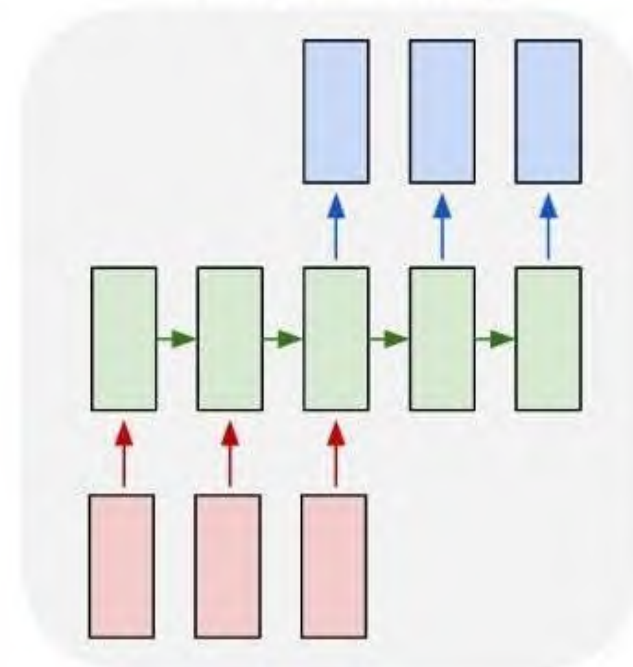
one to many



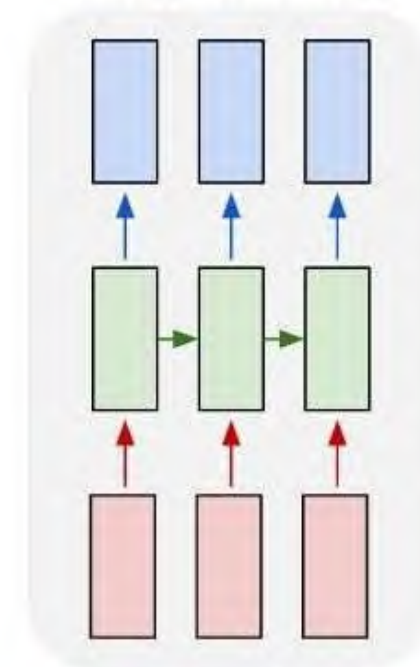
many to one



many to many



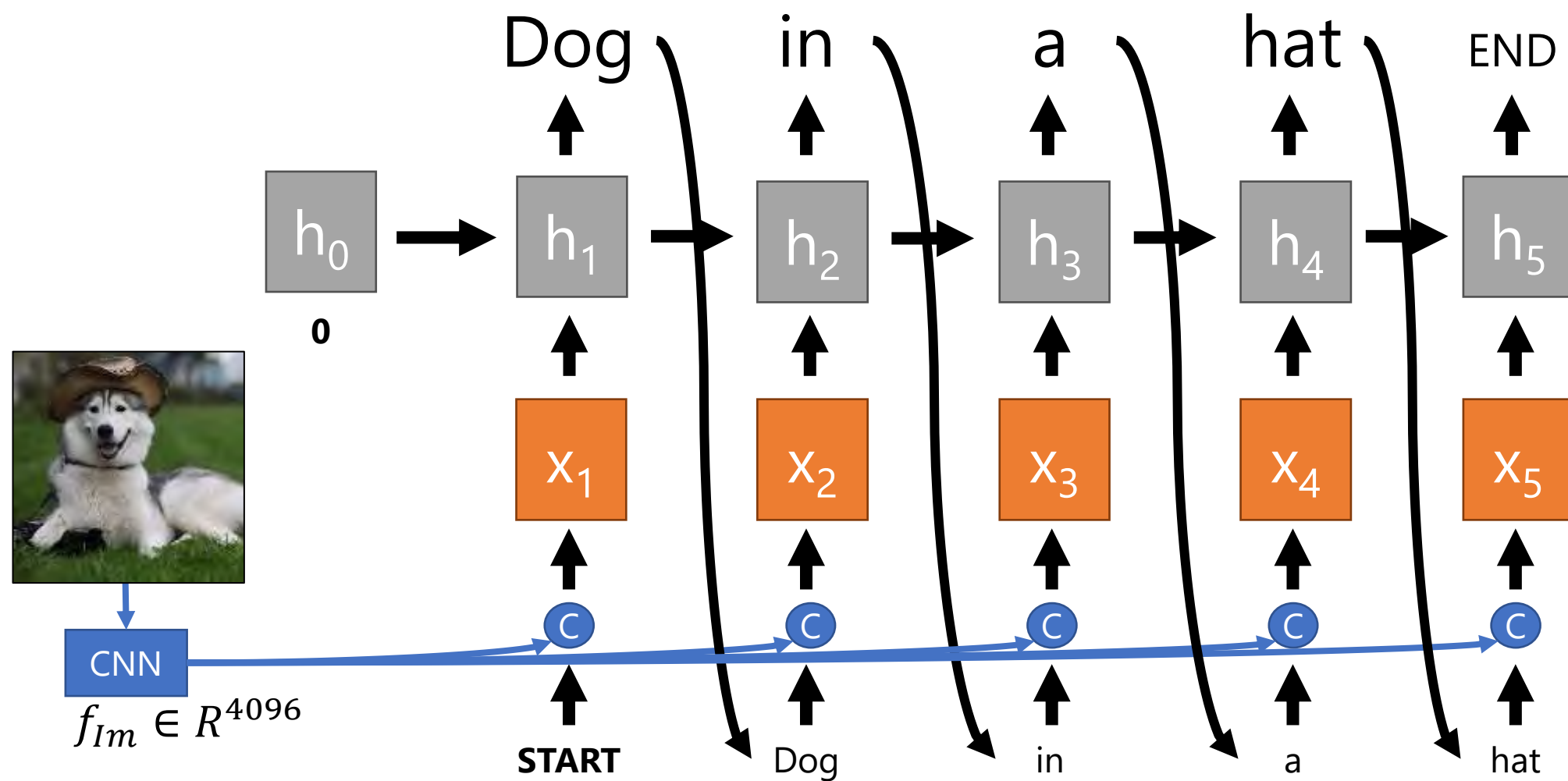
many to many



e.g. 帧级别的视频分类



Captioning: 图像描述

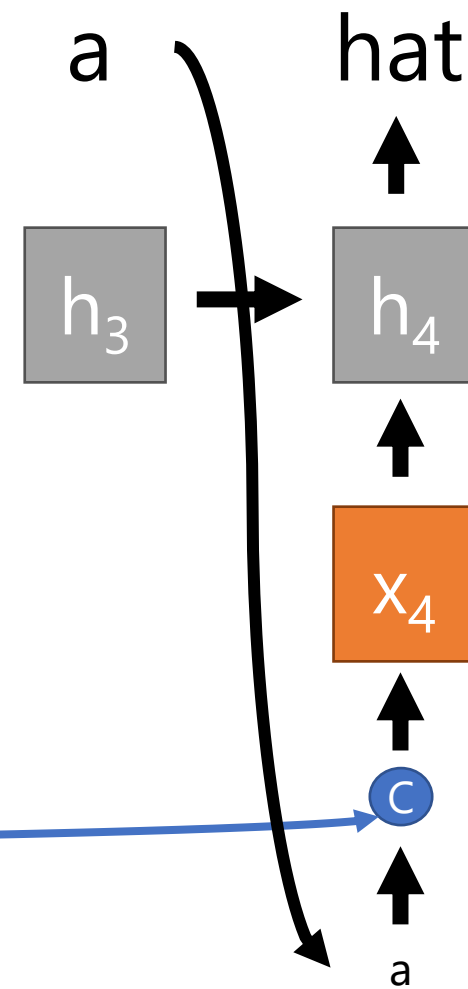


Captioning: 图像描述

每一步:查看输入和隐藏状态
来决定输出。可以用CNN来学
习图像的特征。

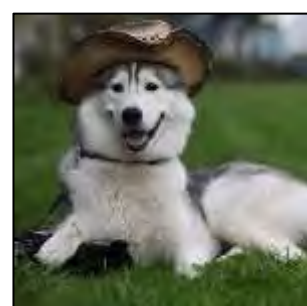


CNN
 $f_{Im} \in R^{4096}$



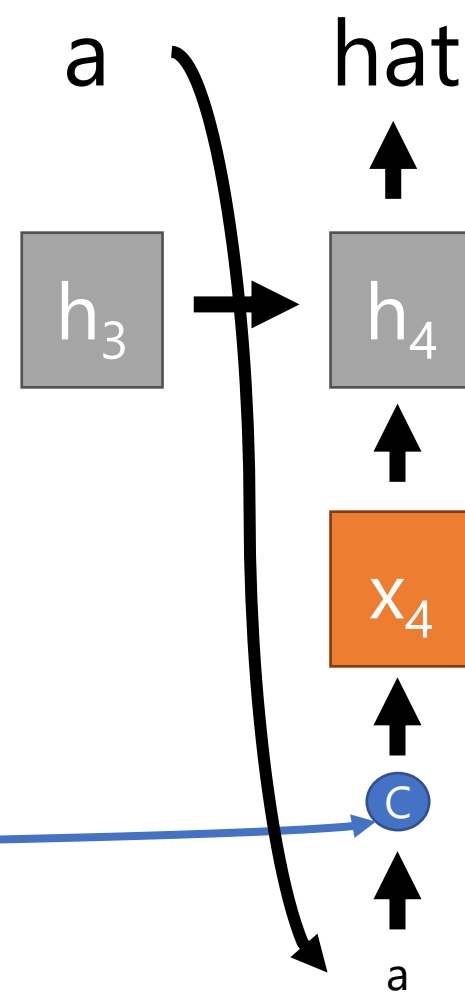
图文描述

为什么这种方法比进行数十亿次的分类问题处理要好？



CNN

$f_{Im} \in R^{4096}$



图像描述：效果



A female tennis player in action on the court.



A group of young men playing a game of soccer



A man riding a wave on top of a surfboard.



A baseball game in progress with the batter up to plate.



A brown bear standing on top of a lush green field.



A person holding a cell phone in their hand.

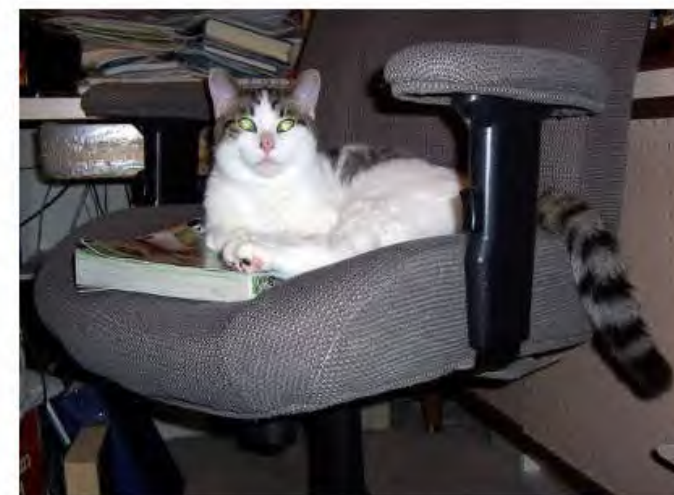
图像描述：失败案例？



A close up of a person brushing his teeth.



A woman laying on a bed in a bedroom.



A black and white cat is sitting on a chair.



A large clock mounted to the side of a building.



A bunch of fruit that are sitting on a table.



A toothbrush holder sitting on top of a white sink.

图像描述: 失败案例?



A woman is holding a cat in her hand



A person holding a computer mouse on a desk



A woman standing on a beach holding a surfboard



A bird is perched on a tree branch



A man in a baseball uniform throwing a ball

训练RNN写Linux Code

```
/*
 * If this error is set, we will need anything right after that BSD.
 */
static void action_new_function(struct s_stat_info *wb)
{
    unsigned long flags;
    int lel_idx_bit = e->edd, *sys & ~((unsigned long) *FIRST_COMPAT);
    buf[0] = 0xFFFFFFFF & (bit << 4);
    min(inc, slist->bytes);
    printk(KERN_WARNING "Memory allocated %02x/%02x, "
        "original MLL instead\n"),
        min(min(multi_run - s->len, max) * num_data_in),
        frame_pos, sz + first_seg);
    div_u64_w(val, inb_p);
    spin_unlock(&disk->queue_lock);
    mutex_unlock(&s->sock->mutex);
    mutex_unlock(&func->mutex);
    return disassemble(info->pending_bh);
}
```

RNN

RNN的输出. 蓝色到红色显示了每个代码按照时间步数生成的顺序。这能说明什么？

```
/* Duplicate LSM field information. The lsm_rule is opaque, so
 * re-initialized. */
static inline int audit_dupe_lsm_field(struct audit_field *df,
                                     struct audit_field *sf)
{
    int ret = 0;
    char *lsm_str;
    /* our own copy of lsm_str */
    lsm_str = kstrdup(sf->lsm_str, GFP_KERNEL);
    if (unlikely(!lsm_str))
        return -ENOMEM;
    df->lsm_str = lsm_str;
    /* our own (refreshed) copy of lsm_rule */
    ret = security_audit_rule_init(df->type, df->op, df->lsm_str,
                                  (void **)&df->lsm_rule);
    /* Keep currently invalid fields around in case they
     * become valid after a policy reload. */
    if (ret == -EINVAL) {
        pr_warn("audit rule for LSM \'%s\' is invalid\n",
                df->lsm_str);
        ret = 0;
    }
    return ret;
}
```


RNN

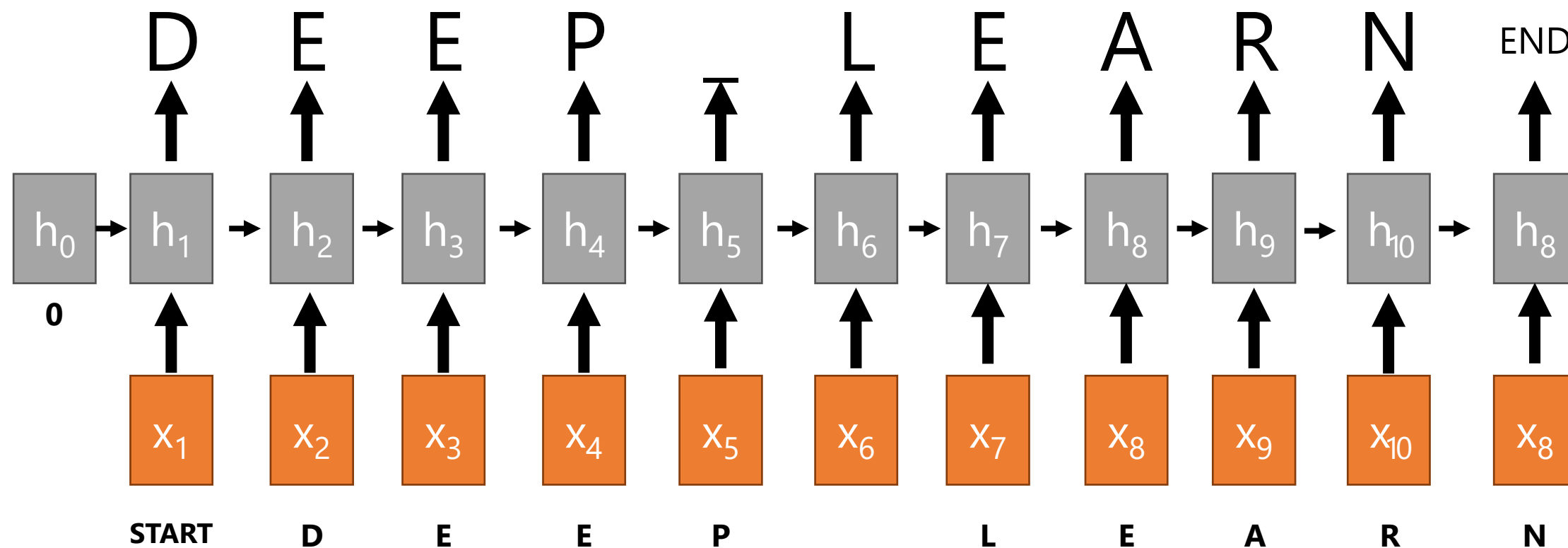
RNN的输出. 蓝色 到 红色 显示了每个代码按照时间步数生成的顺序。 这能说明什么？

```
#ifdef CONFIG_AUDIT_SYSCALL
static inline int audit_match_class_bits(int class, u32 *mask)
{
    int i;
    if (classes[class]) {
        for (i = 0; i < AUDIT_BITMASK_SIZE; i++)
            if (mask[i] & classes[class][i])
                return 0;
    }
    return 1;
}
```

可能出现的问题：迭代深度

如果迭代的步数过多会怎样？

考虑 $g^n \quad g \neq 1$
梯度爆炸或消失



可能出现的问题：迭代深度

- 有一些更复杂的方法 (LSTM, GRU) 来避免这些问题
- 总体策略（残差学习）：尽量保证隐藏状态的改变比较小，可以每步只加上一小点状态更新

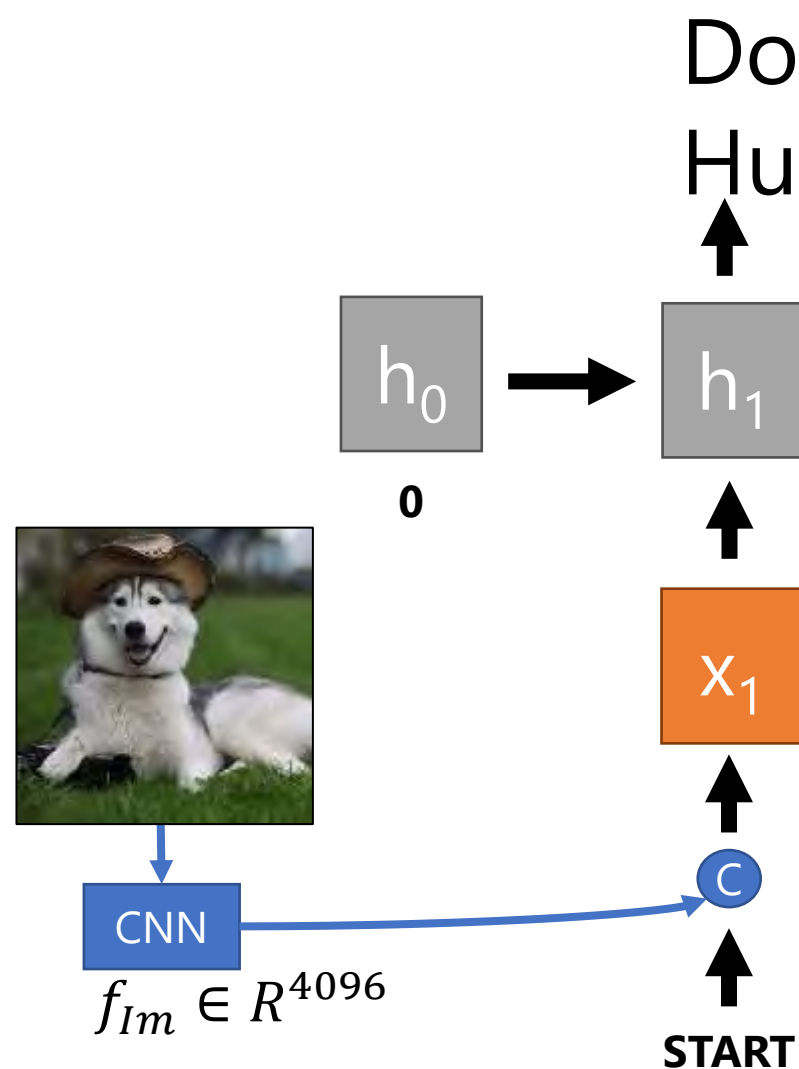
可能出现的问题：多个可能的结果

可能有很多描述都是对的!



- A dog in a hat
- A dog wearing a hat
- Husky wearing a hat
- Husky holding a camera, sitting in grass
- A dog that's in a hat, sitting on a lawn with a camera

可能出现的问题：多个可能的结果



- 根据每个单词的概率来去采样
- 可以调整“temperature”温度参数 $\exp(\text{score}/t)$ 来调节概率分布
- $\exp(5) / \exp(1) \rightarrow 54.6$
- $\exp(5/5) / \exp(1/5) \rightarrow 2.2$

温度的效果

- 关于创业投资的段落
- **中等温度（平衡点，既有一定创造性也有可预测性）**：“创业公司正处在一个转折点，人工智能的崛起使得新的数据分析工具变得至关重要。投资者在寻找那些能够利用机器学习优化其业务流程的初创企业。在这个环境下，能够清晰展示其价值主张的公司最有可能获得资金。”
- **高温（创造性高，随机性高）**：“在光滑的滨江道路上，有一个会说诗的机器人引发了投资者的狂热，他们纷纷拿出奇怪的艺术品作为资本。一个穿着西装的猫成为了下一个大亨，而闪电辩论赛决定了下一轮融资的赢家。”
- **低温（保守，重复性高）**：“创业公司需要资金来发展。投资者寻找有潜力的创业公司来投资。创业公司需要资金来发展。投资者寻找有潜力的创业公司来投资。创业公司需要资金来发展，这是一个不断循环的过程。”

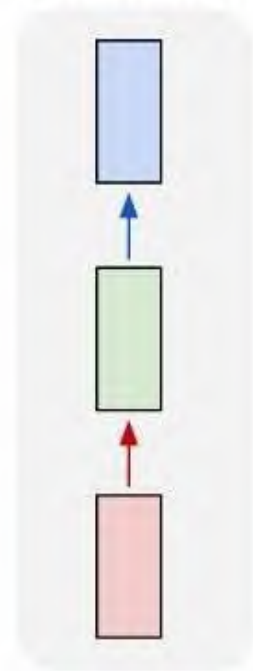
回顾：RNN与序列模型

处理序列数据——不同的任务

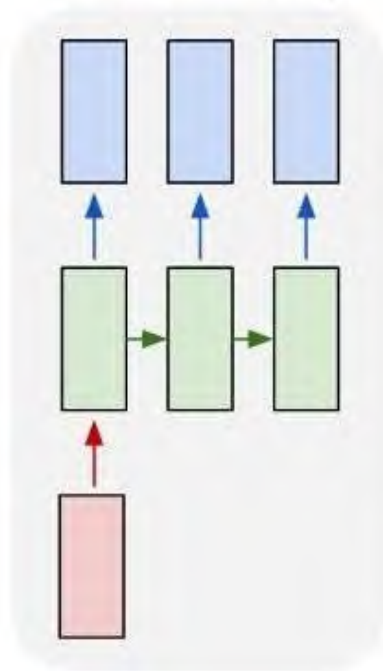
E.g. 视频配文

Sequence of video frames -> caption

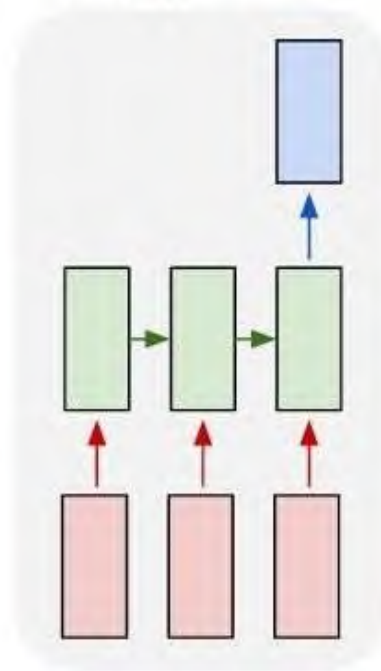
one to one



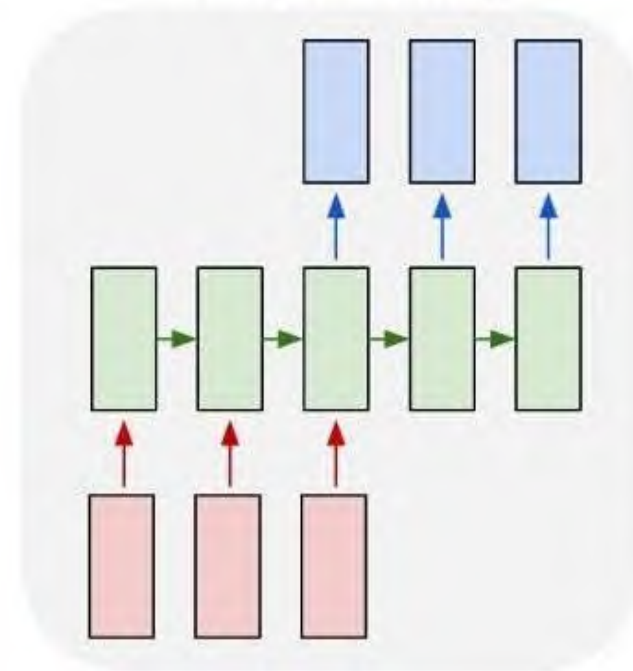
one to many



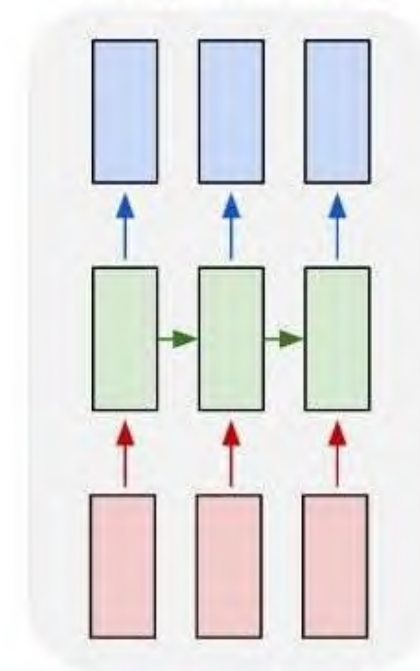
many to one



many to many



many to many



e.g. 图文描述

image -> sequence of words

e.g. 动作/事件预测

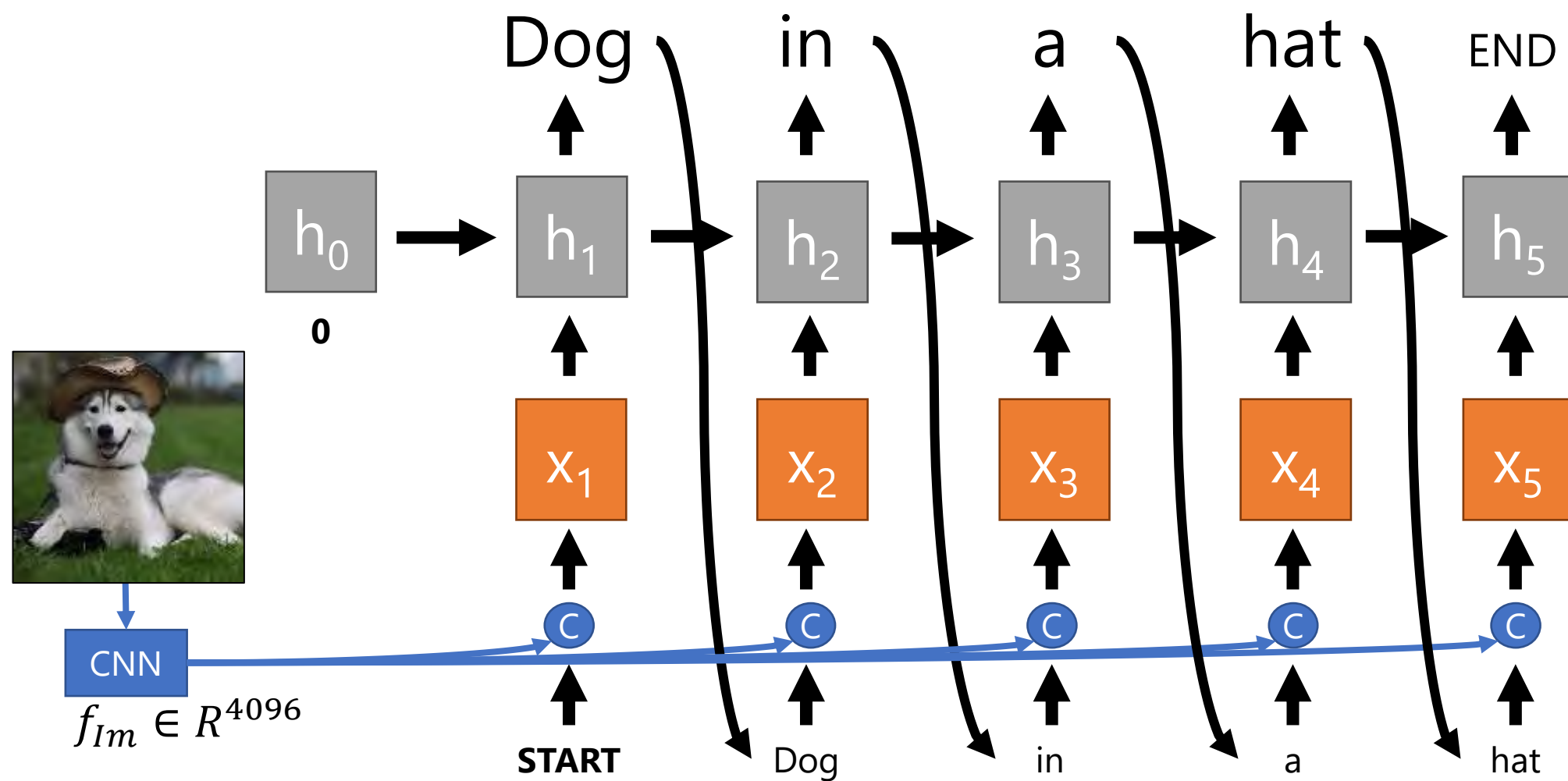
sequence of video frames -> action class

用户评价打分

Review -> Scores

e.g. 帧级别的视频分类

Captioning: 图像描述



图像描述：效果



A female tennis player in action on the court.



A group of young men playing a game of soccer



A man riding a wave on top of a surfboard.



A baseball game in progress with the batter up to plate.



A brown bear standing on top of a lush green field.



A person holding a cell phone in their hand.

图像描述：失败案例？

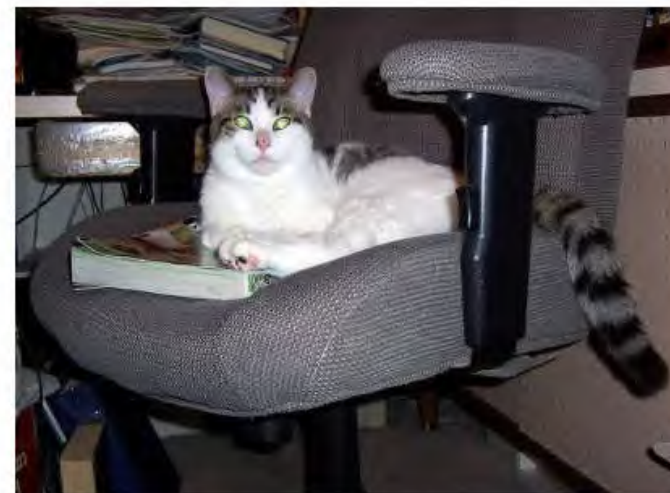
- 描述存在语法错误或前后矛盾



A close up of a person brushing his teeth.



A woman laying on a bed in a bedroom.



A black and white cat is sitting on a chair.



A large clock mounted to the side of a building.



A bunch of fruit that are sitting on a table.



A toothbrush holder sitting on top of a white sink.

- 描述存在与图片内容不符的问题

图像描述: 失败案例?



A woman is holding a cat in her hand



A person holding a computer mouse on a desk



A woman standing on a beach holding a surfboard



A bird is perched on a tree branch



A man in a baseball uniform throwing a ball

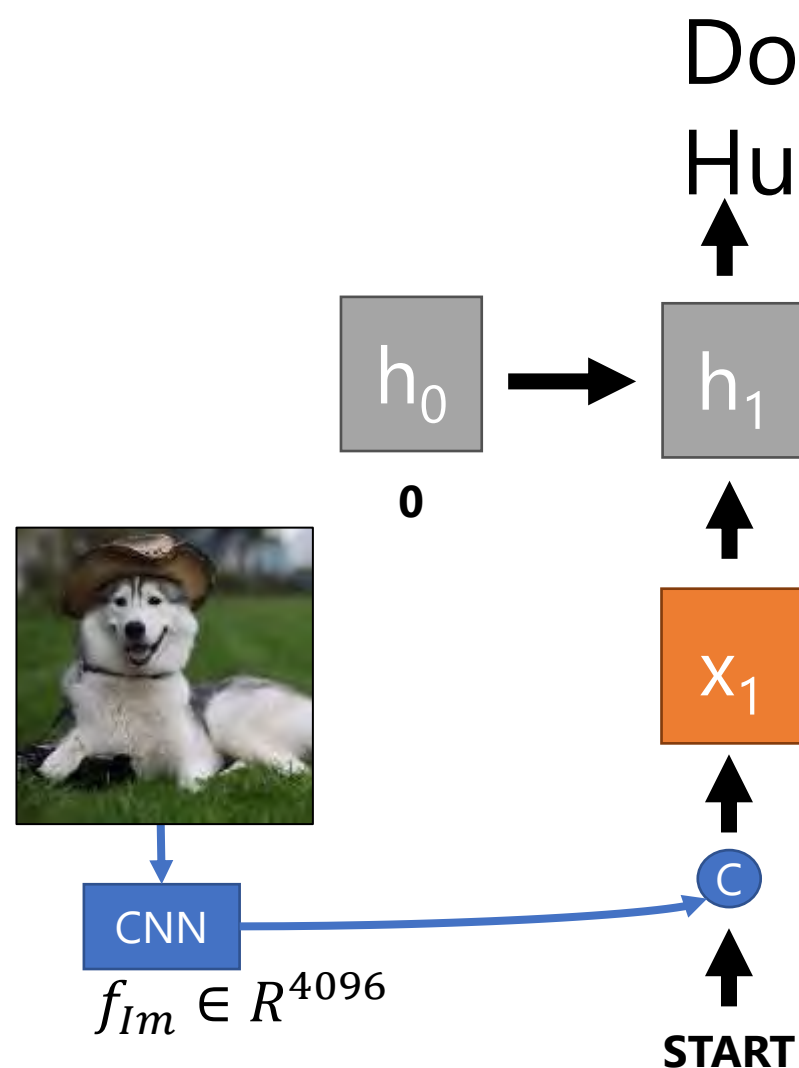
可能出现的问题：多个可能的结果

可能有很多描述都是对的!



- A dog in a hat
- A dog wearing a hat
- Husky wearing a hat
- Husky holding a camera, sitting in grass
- A dog that's in a hat, sitting on a lawn with a camera

可能出现的问题：多个可能的结果

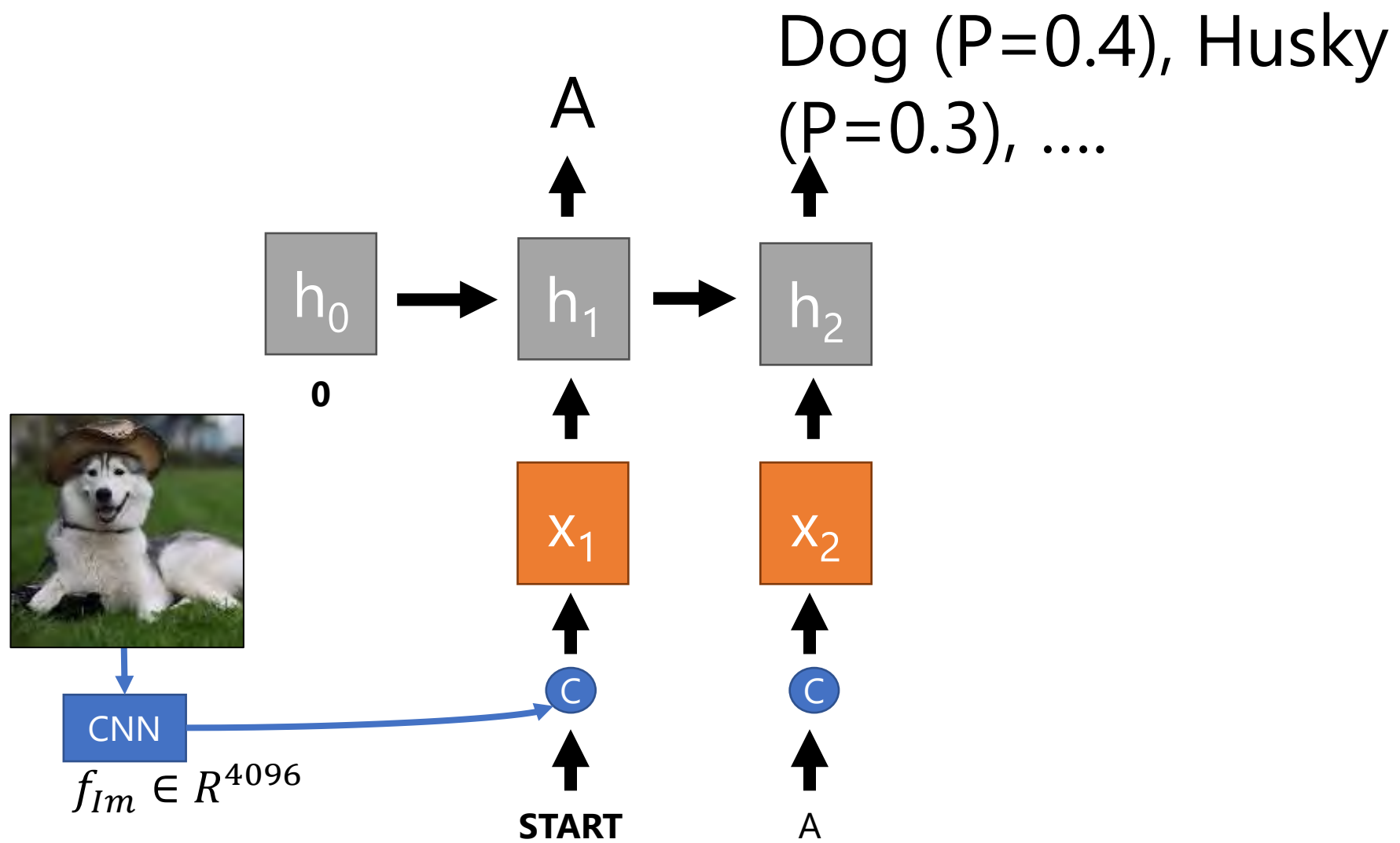


- 根据每个单词的概率来去采样
- 可以调整“temperature”温度参数 $\exp(\text{score}/t)$ 来调节概率分布
- $\exp(5) / \exp(1) \rightarrow 54.6$
- $\exp(5/5) / \exp(1/5) \rightarrow 2.2$

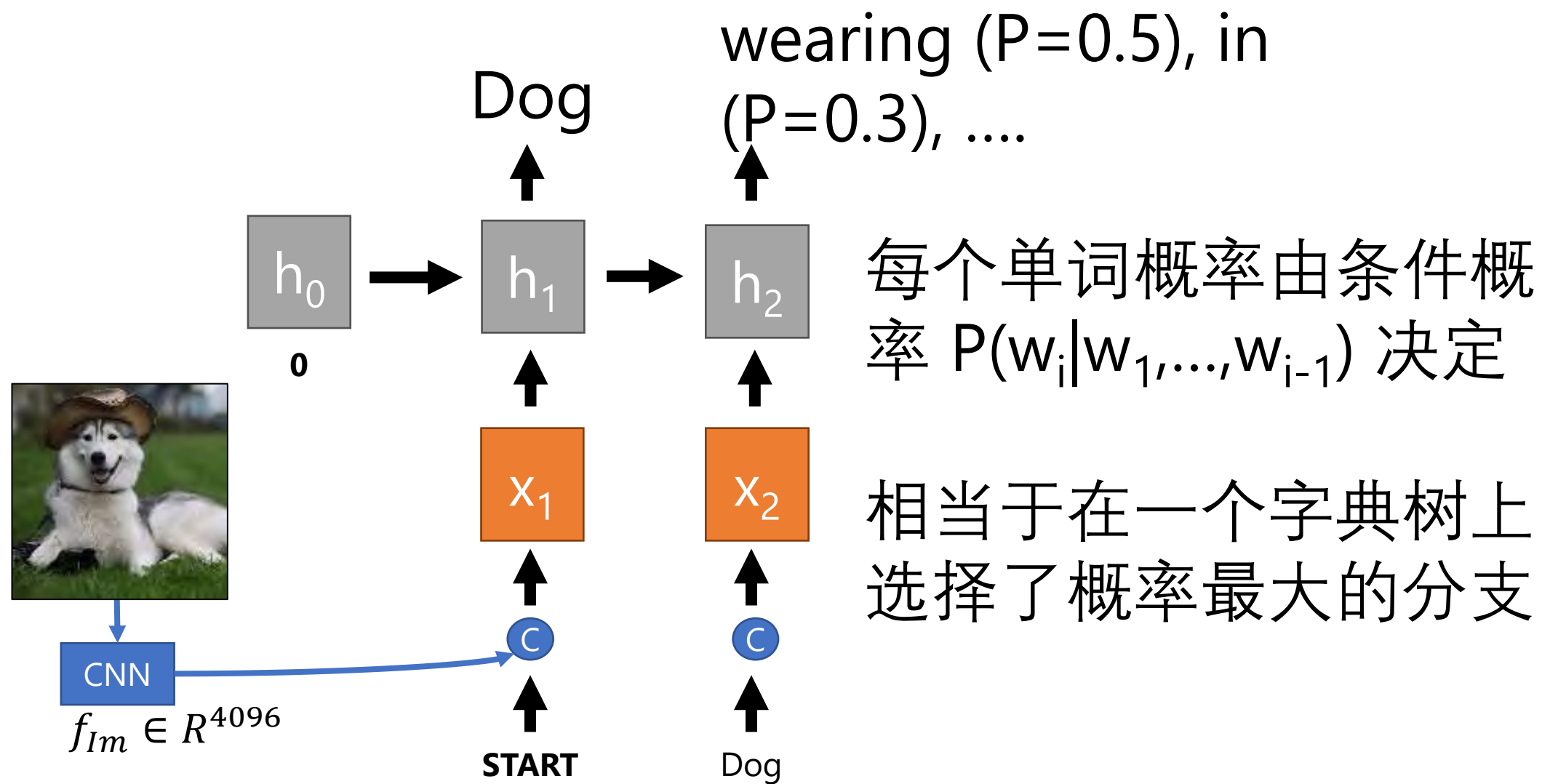
温度的效果

- 关于创业投资的段落
- **中等温度（平衡点，既有一定创造性也有可预测性）**：“创业公司正处在一个转折点，人工智能的崛起使得新的数据分析工具变得至关重要。投资者在寻找那些能够利用机器学习优化其业务流程的初创企业。在这个环境下，能够清晰展示其价值主张的公司最有可能获得资金。”
- **高温（创造性高，随机性高）**：“在光滑的滨江道路上，有一个会说诗的机器人引发了投资者的狂热，他们纷纷拿出奇怪的艺术品作为资本。一个穿着西装的猫成为了下一个大亨，而闪电辩论赛决定了下一轮融资的赢家。”
- **低温（保守，重复性高）**：“创业公司需要资金来发展。投资者寻找有潜力的创业公司来投资。创业公司需要资金来发展。投资者寻找有潜力的创业公司来投资。创业公司需要资金来发展，这是一个不断循环的过程。”

采样



采样



可能的问题：质量评估



Computer: "A husky in a hat"

Human: "A dog in a hat"

哪个描述更好?

- 1) 人工评估
- 2) 避免使用简单指标导致简单输出 (e.g., "A a a a a")
- 3) 实际常用：单词重叠度、词汇频率和召回率、特征匹配度。。。

图像内容: Attention

注意力机制会让我们关注不同的空间区域



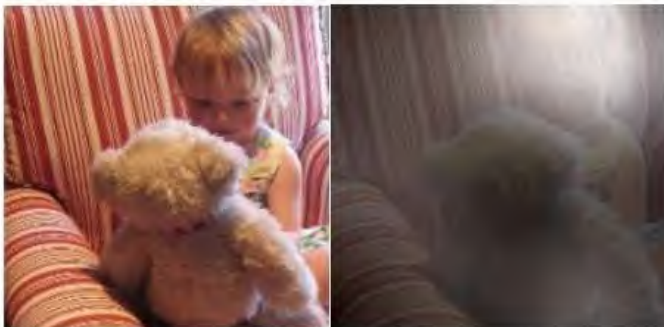
A woman is throwing a frisbee in a park.



A dog is standing on a hardwood floor.



A stop sign is on a road with a mountain in the background.



A little girl sitting on a bed with a teddy bear.



A group of people sitting on a boat in the water.



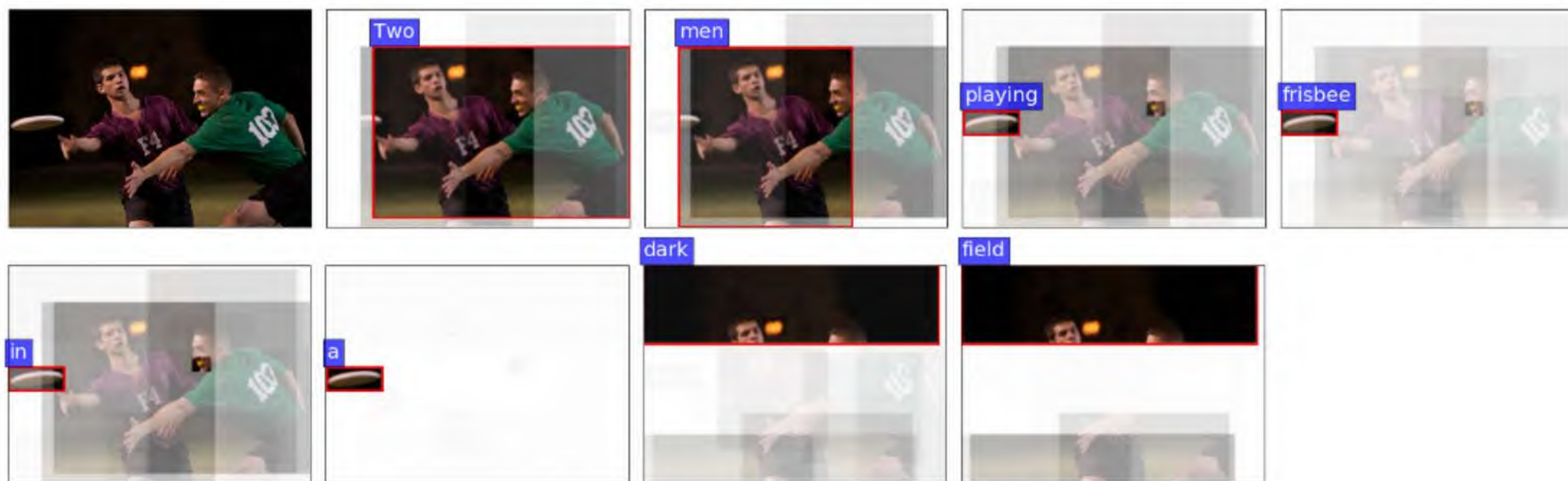
A giraffe standing in a forest with trees in the background.

Xu et al, "Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Figure copyright Kelvin Xu, Jimmy Lei Ba, Jamie Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio, 2015. Reproduced with permission.

Attention

通过attention, 关注图像当中的物体, 而不是整张图, 来提升性能



Two men playing frisbee in a dark field.

Attention

Question: What color is illuminated on the traffic light? Answer left: green. Answer right: red.



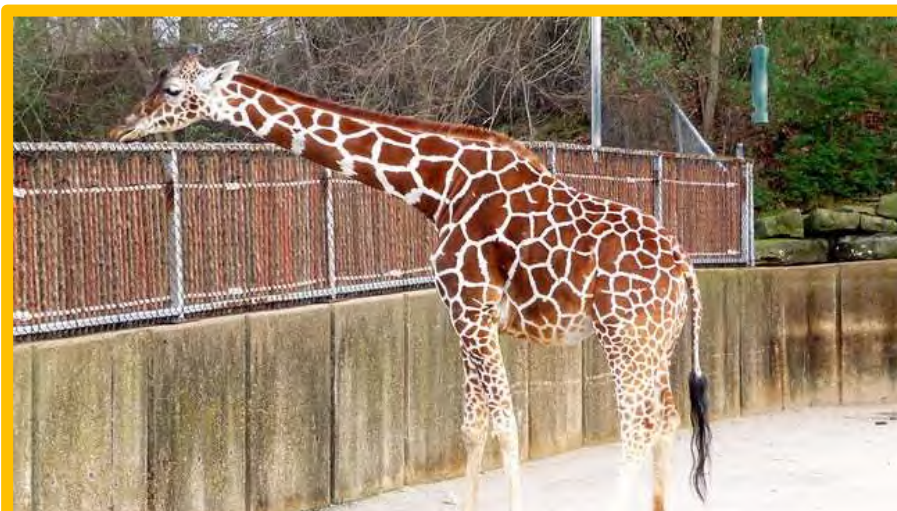
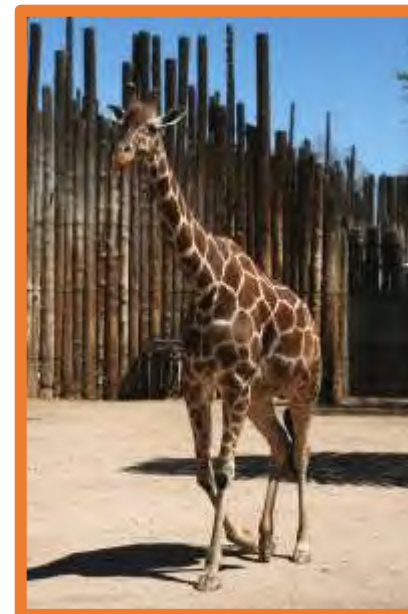
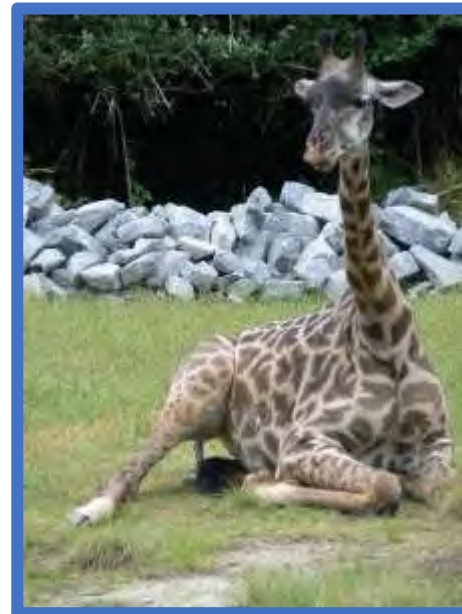
语法问题：答案空间

- 如果字典大小是10k, 20个单词的句子有多少可能的答案?
- $(10k)^{20}$? **是吗?**
- 为什么不是呢?

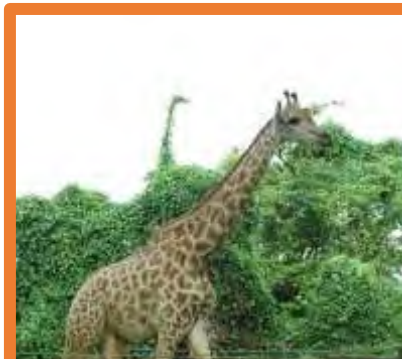
长颈鹿整天都在干啥



A giraffe sitting and resting



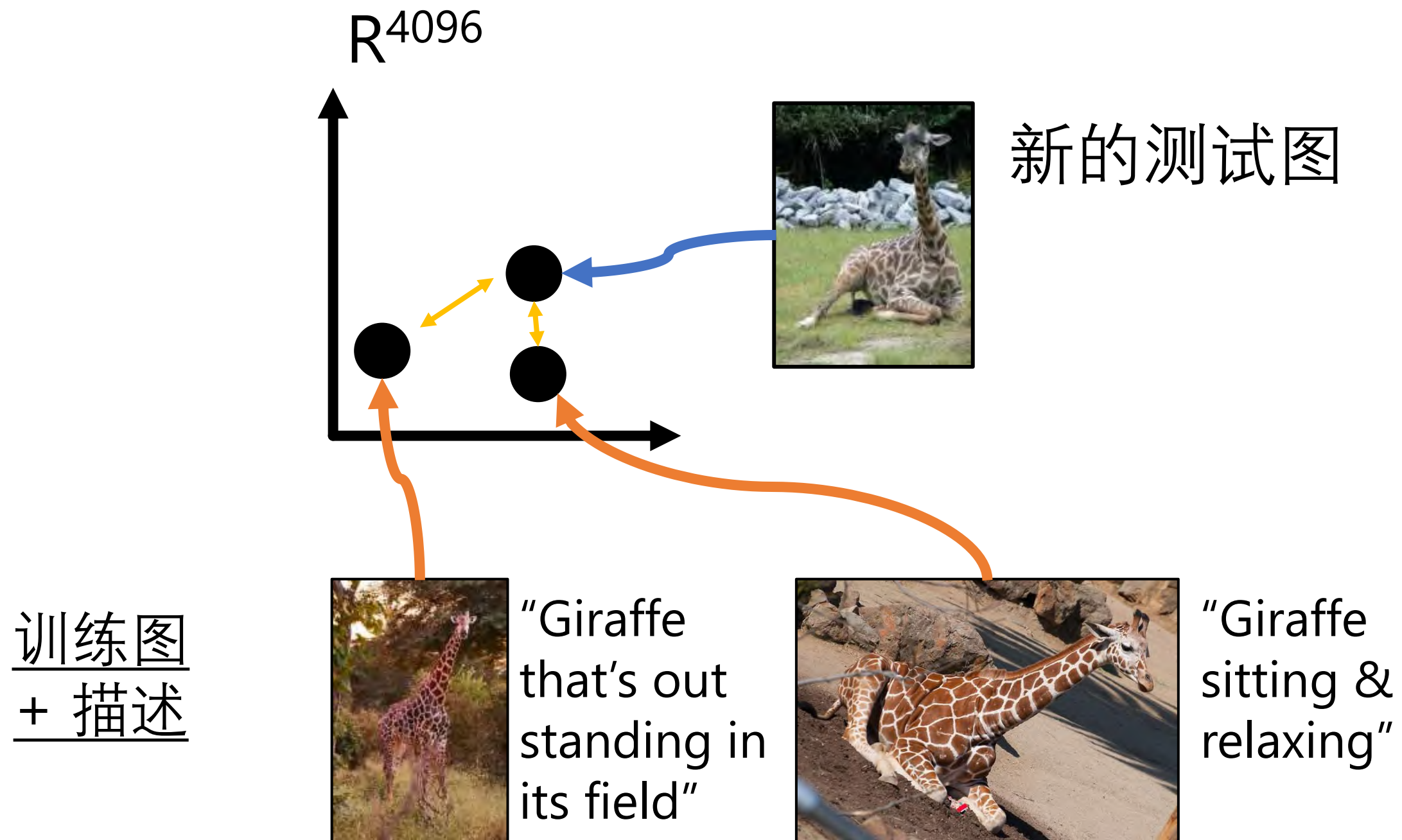
A giraffe grazing in its enclosure



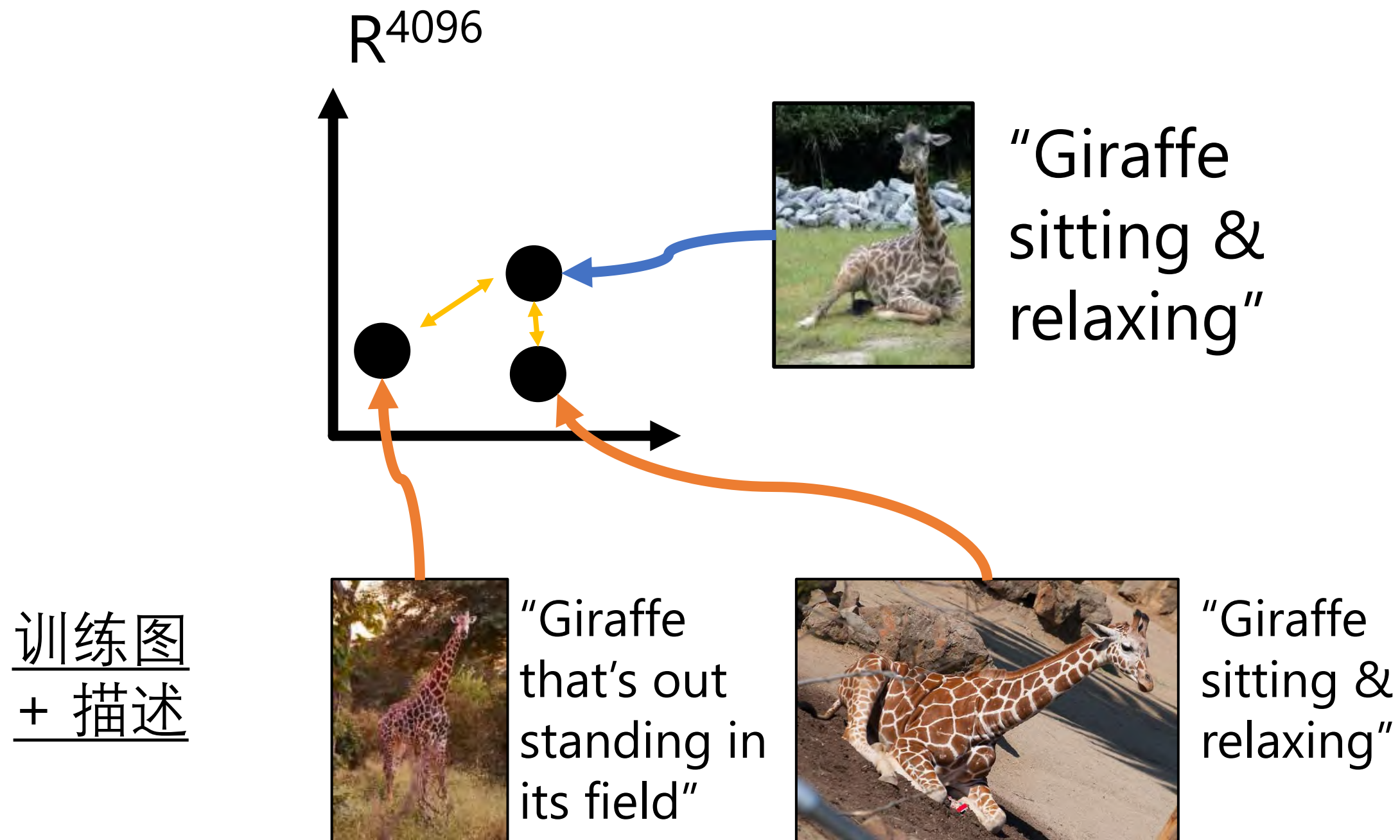
A giraffe wandering around



另一种描述的方式：检索



另一种描述的方式：检索



检索结果



A man riding a wave on a surfboard.

A man riding a wave on a surfboard in the ocean.



A person flying a kite in the sky.

A person flying a kite in the sky.



A cat sitting in a bathroom sink.

A black and white cat sitting in a bathroom sink.

检索结果

- 我们可能不太喜欢检索的结果
- 因为它们没有创造新的句子!



A wooden bench in front of a building.

A window display on the front of a building.



A building with a clock on the top.

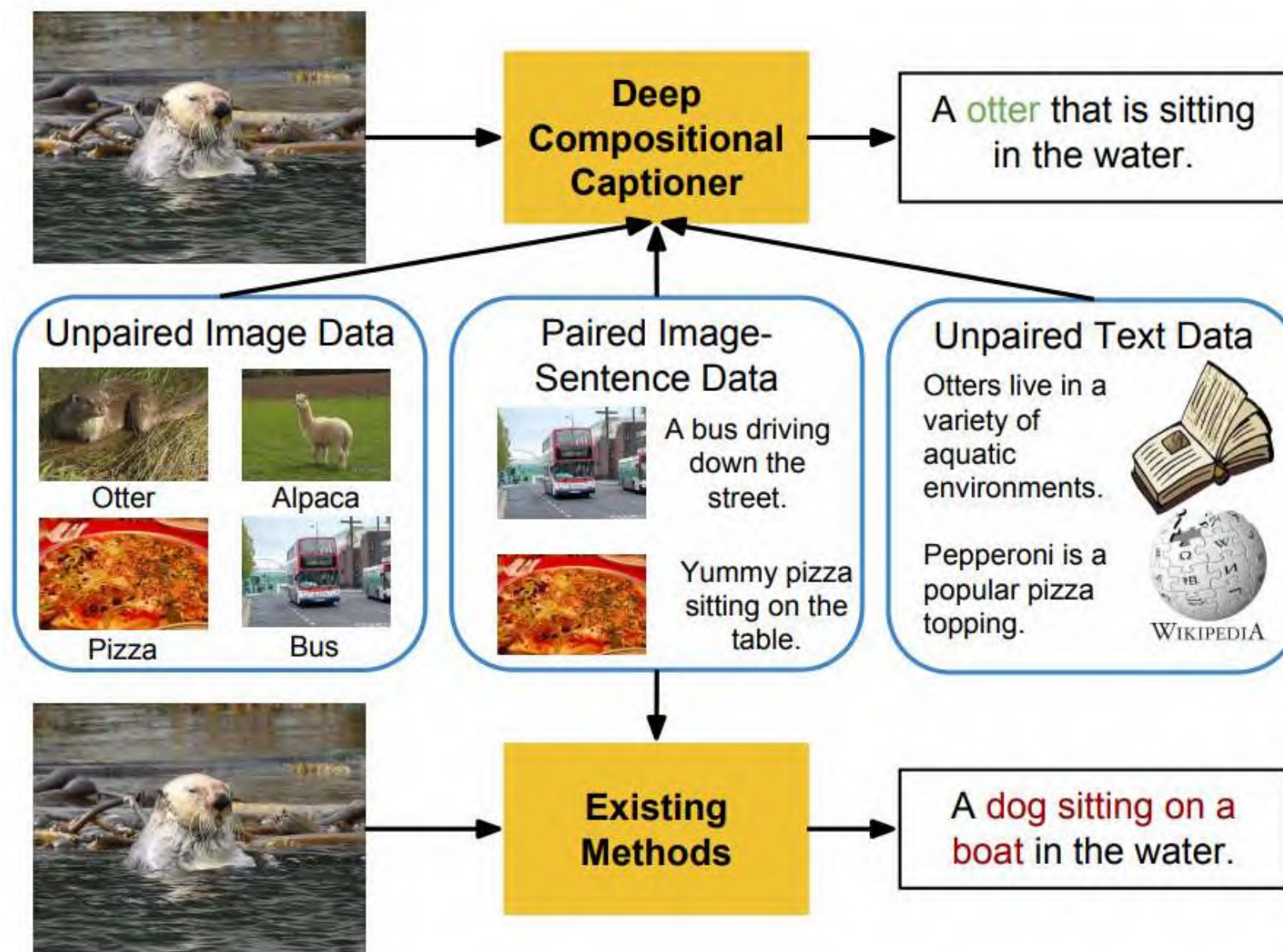
A clock tower on the top of a building.



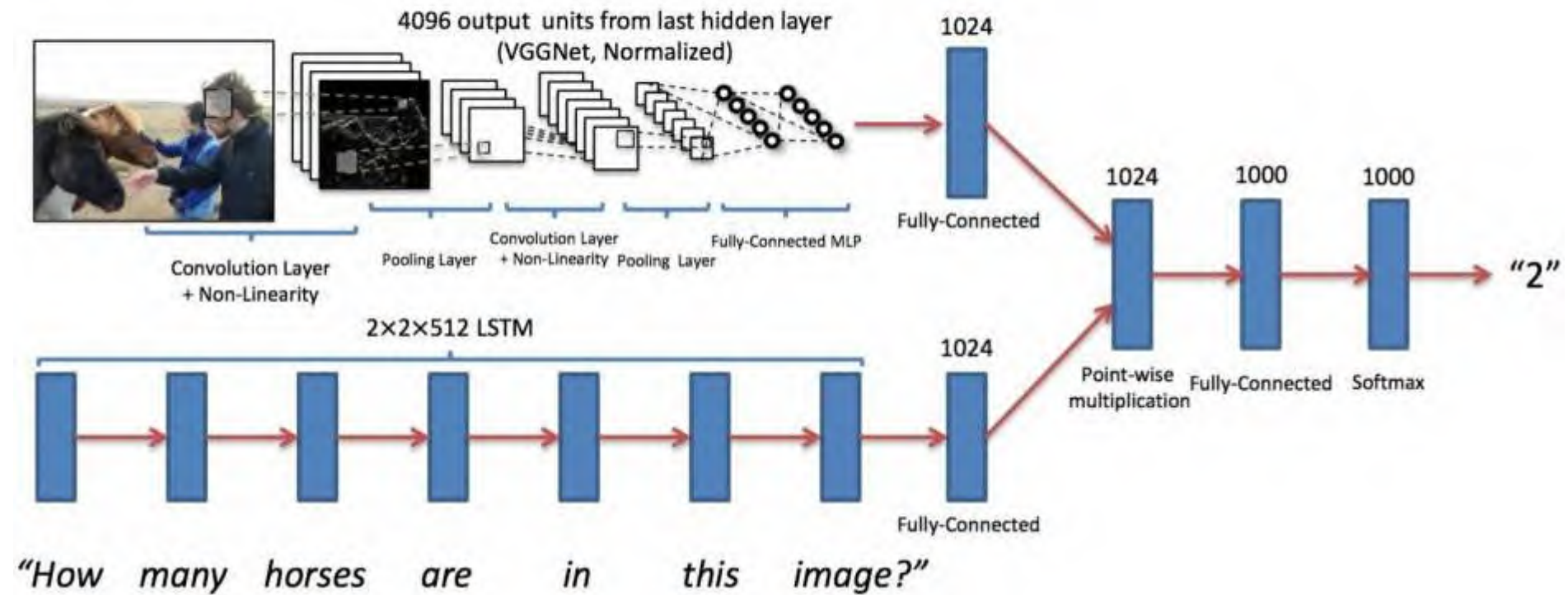
The side of a passenger train at a train station.

A bus that is on the side of a road.

怎么获得新的描述? Unpaired Data



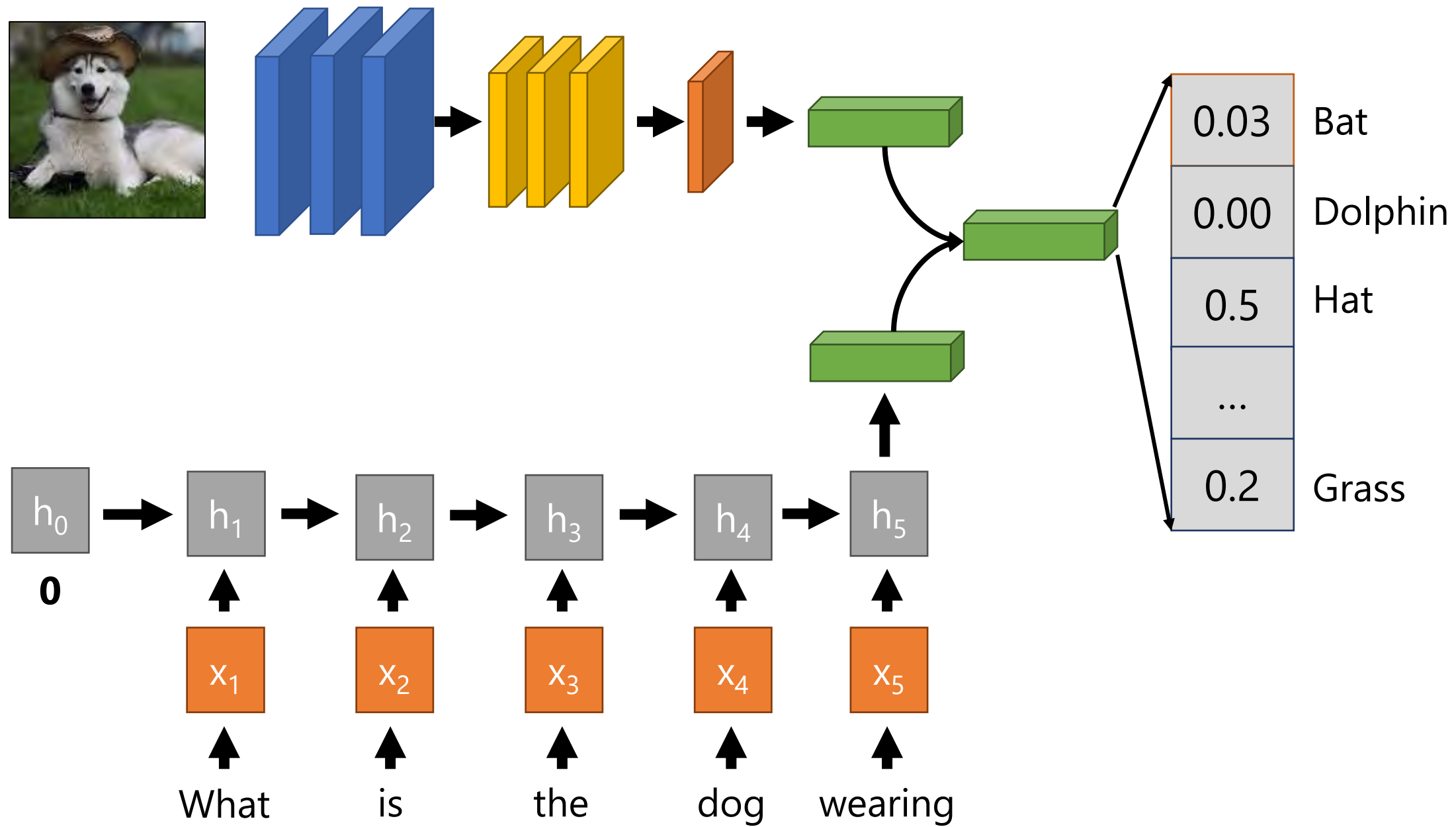
Visual Question Answering (VQA)



Agrawal et al, "Visual 7W: Grounded Question Answering in Images", CVPR 2015
Figures from Agrawal et al, copyright IEEE 2015. Reproduced for educational purposes.

需要匹配文本和图像特征

VQA模型



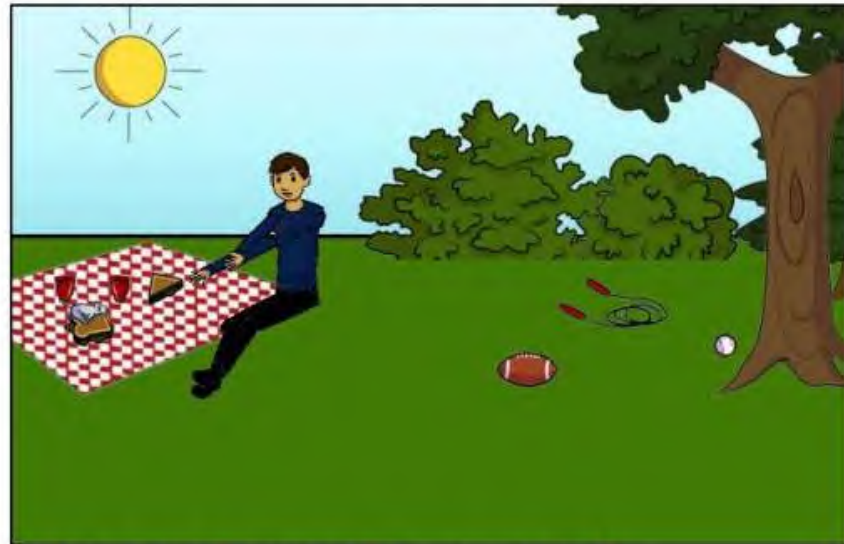
VQA



What color are her eyes?
What is the mustache made of?



How many slices of pizza are there?
Is this a vegetarian pizza?



Is this person expecting company?
What is just under the tree?



Does it appear to be rainy?
Does this person have 20/20 vision?

另一种VQA



Q: What endangered animal is featured on the truck?

- A: **A bald eagle.**
- A: A sparrow.
- A: A hummingbird.
- A: A raven.



Q: Where will the driver go if turning right?

- A: **Onto 24 1/4 Rd.**
- A: Onto 25 3/4 Rd.
- A: Onto 23 3/4 Rd.
- A: Onto Main Street.



Q: When was the picture taken?

- A: **During a wedding.**
- A: During a bar mitzvah.
- A: During a funeral.
- A: During a Sunday church service.



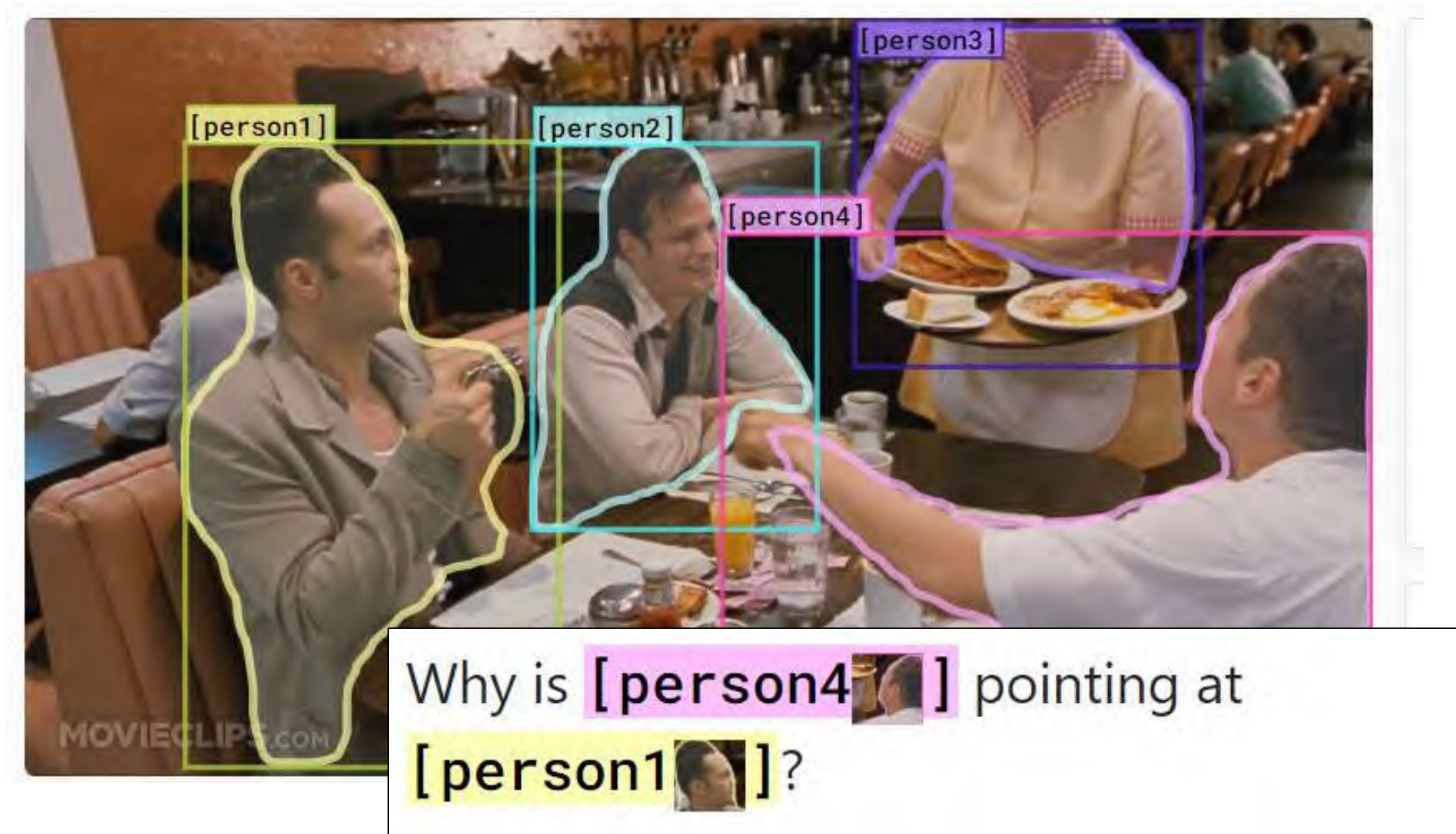
Q: Who is under the umbrella?

- A: **Two women.**
- A: A child.
- A: An old man.
- A: A husband and a wife.

Agrawal et al, "VQA: Visual Question Answering", ICCV 2015
Zhu et al, "Visual 7W: Grounded Question Answering in Images", CVPR 2016
Figure from Zhu et al, copyright IEEE 2016. Reproduced for educational purposes.


根据图片和文字提问回答问题

Visual Commonsense Reasoning (VCR)



VCR

Why is [person4] pointing at [person1]?



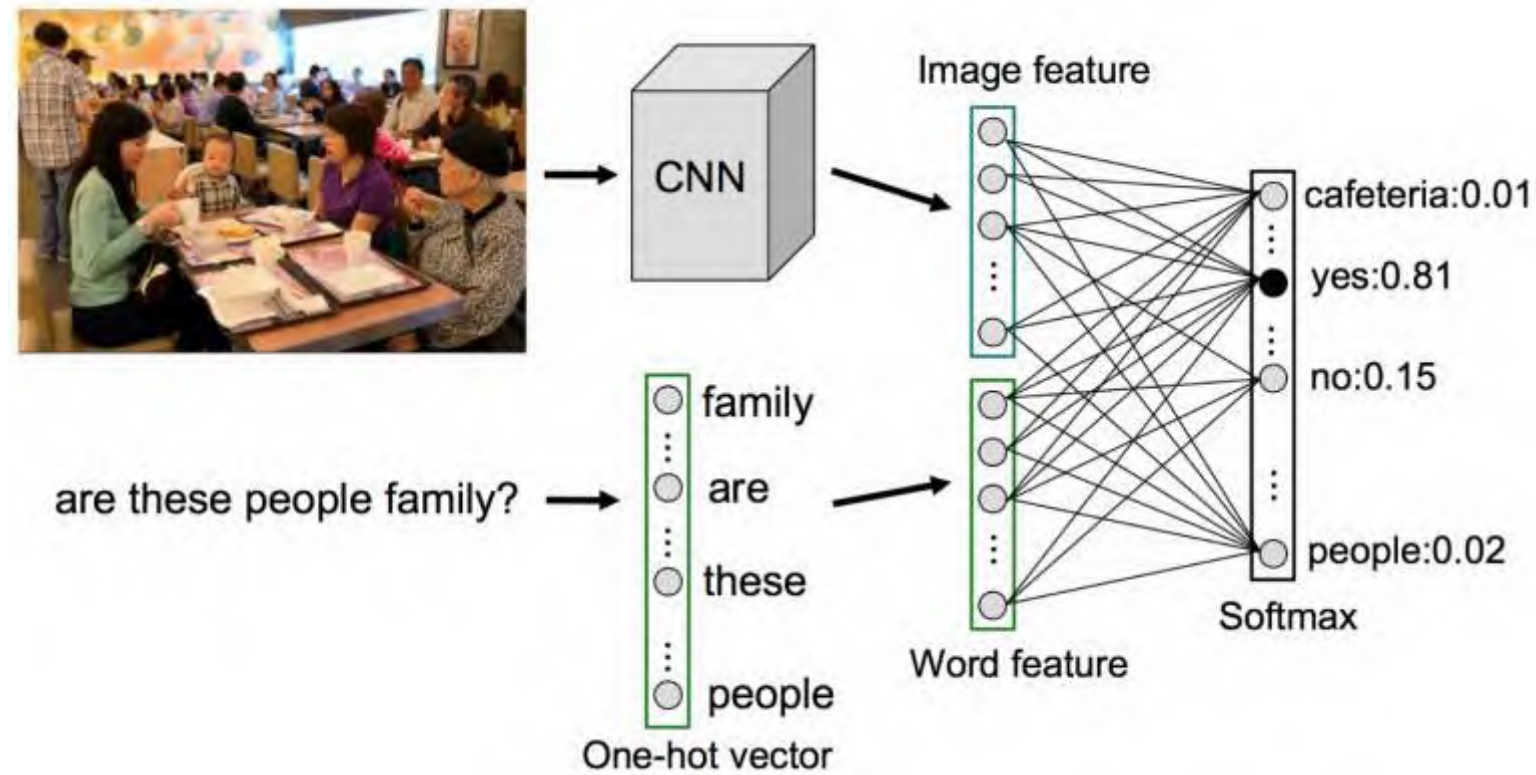
a) He is telling [person3] that [person1] ordered the pancakes.

b) He just told a joke.

c) He is feeling accusatory towards [person1].

d) He is giving [person1] directions.

一个简单的VQA实现方式



- 建立字典，包括5000个最常用的回答所包含的词汇
- 用CNN提取图像的特征 I
- 用BoW表示问题 Q
- 根据概率 $P(A|Q, I)$ 得到回答

Zhou, Bolei, et al. "Simple baseline for visual question answering." arXiv preprint arXiv:1512.02167 (2015).

Slide credit: T. Gupta

效果



Question: what are they doing

Predictions:

playing baseball (score: 10.67 = 2.01 [image] + 8.66 [word])

baseball (score: 9.65 = 4.84 [image] + 4.82 [word])

grazing (score: 9.34 = 0.53 [image] + 8.81 [word])

Based on image only: umpire (4.85), baseball (4.84), batter (4.46)

Based on word only: playing wii (10.62), eating (9.97),
playing frisbee (9.24)

Question: how many people inside

Predictions:

3 (score: 13.39 = 2.75 [image] + 10.65 [word])

2 (score: 12.76 = 2.49 [image] + 10.27 [word])

5 (score: 12.72 = 1.83 [image] + 10.89 [word])

Based on image only: umpire (4.85), baseball (4.84), batter (4.46)

Based on word only: 8 (11.24), 7 (10.95), 5 (10.89)

Zhou, Bolei, et al. "Simple baseline for visual question answering." arXiv preprint arXiv:1512.02167 (2015).

Slide credit: T. Gupta

效果



Question: which brand is the laptop

Predictions:

apple (score: 10.87 = 1.10 [image] + 9.77 [word])

dell (score: 9.83 = 0.71 [image] + 9.12 [word])

toshiba (score: 9.76 = 1.18 [image] + 8.58 [word])

Based on image only: books (3.15), yes (3.14), no (2.95)

Based on word only: apple (9.77), hp (9.18), dell (9.12)

- 语言先验可以极大程度减少回答的空间

Zhou, Bolei, et al. "Simple baseline for visual question answering." arXiv preprint arXiv:1512.02167 (2015).

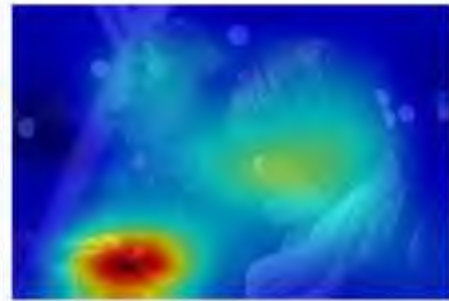
Slide credit: T. Gupta

定量评价

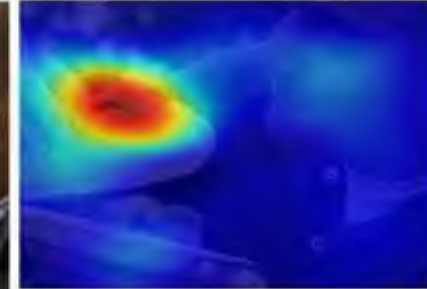
	Open-Ended				Multiple-Choice			
	Overall	yes/no	number	others	Overall	yes/no	number	others
IMG [2]	28.13	64.01	00.42	03.77	30.53	69.87	00.45	03.76
BOW [2]	48.09	75.66	36.70	27.14	53.68	75.71	37.05	38.64
BOWIMG [2]	52.64	75.55	33.67	37.37	58.97	75.59	34.35	50.33
LSTMIMG [2]	53.74	78.94	35.24	36.42	57.17	78.95	35.80	43.41
CompMem [6]	52.62	78.33	35.93	34.46	-	-	-	-
NMN+LSTM [1]	54.80	77.70	37.20	39.30	-	-	-	-
WR Sel. [13]	-	-	-	-	60.96	-	-	-
ACK [16]	55.72	79.23	36.13	40.08	-	-	-	-
DPPnet [11]	57.22	80.71	37.24	41.69	62.48	80.79	38.94	52.16
iBOWIMG	55.72	76.55	35.03	42.62	61.68	76.68	37.05	54.44

模型可以良好地定位吗?

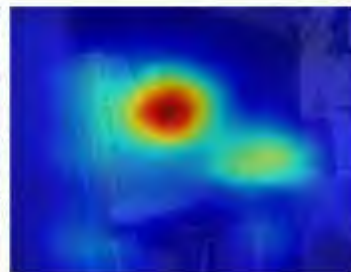
Class Activation Mapping (CAM)



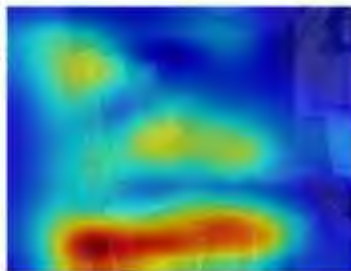
Question: What are they doing?
Prediction: texting (score: 12.02=3.78 [image] + 8.24 [word])
Word importance: doing(7.01) are(1.05) they(0.49) what(-0.3)



Question: What is he eating?
Prediction: hot dog (score: 13.01=5.02 [image] + 7.99 [word])
Word importance: eating(4.12) what(2.81) is(0.74) he(0.30)



Question: Is there a cat?
Prediction: yes (score: 11.48 = 4.35 [image] + 7.13 [word])
word importance: is(2.65) there(2.46) a(1.70) cat(0.30)



Question: Where is the cat?
Prediction: shelf (score: 10.81 = 3.23 [image] + 7.58 [word])
word importance: where(3.89) cat(1.88) the(1.79) is(0.01)

增强视觉分析

能更好地分析复杂的视觉场景，但是数据更难收集

Who is wearing glasses?

man



woman

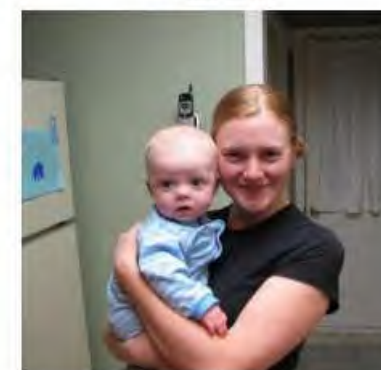


Where is the child sitting?

fridge



arms



Is the umbrella upside down?

yes



no



How many children are in the bed?

2

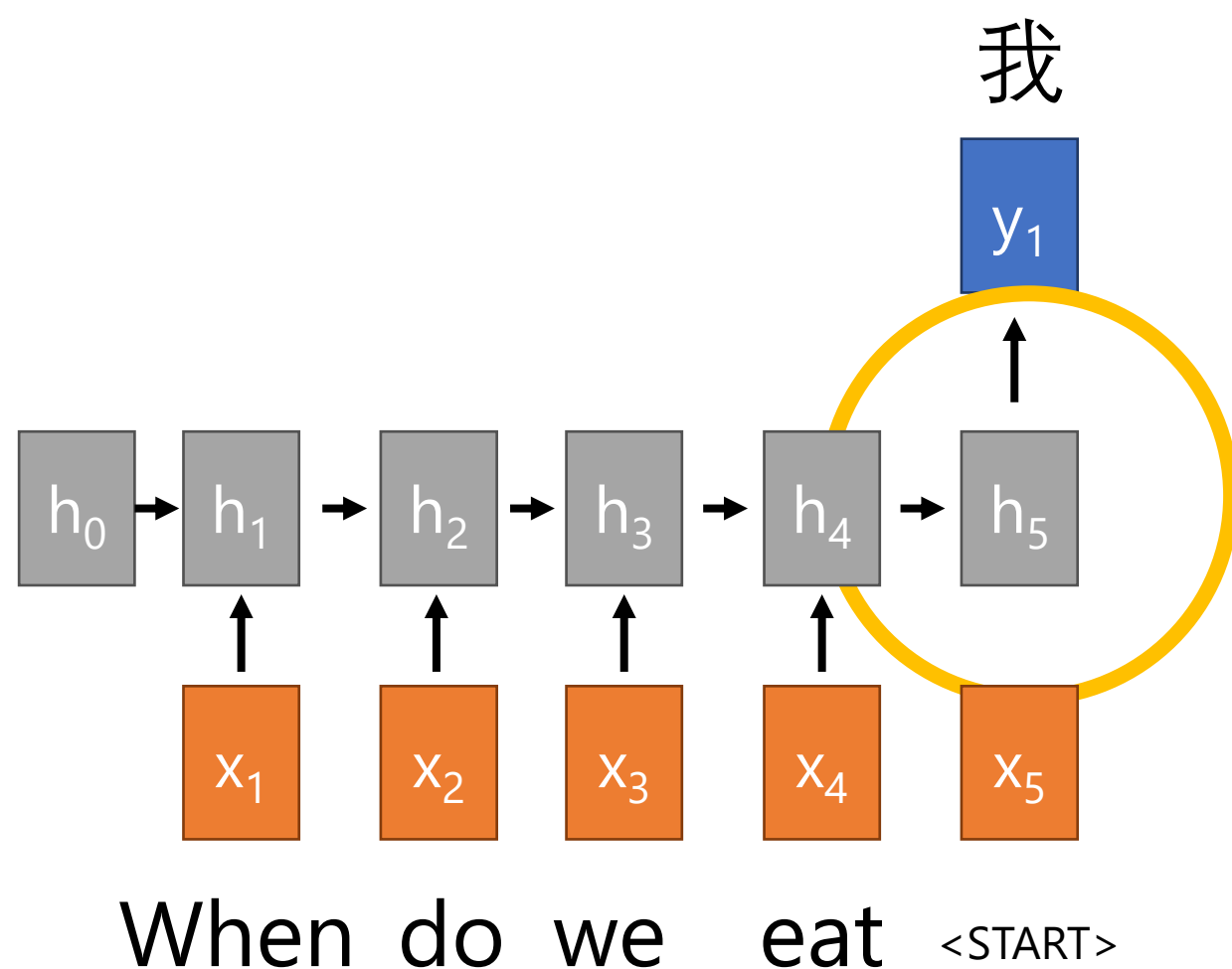


1



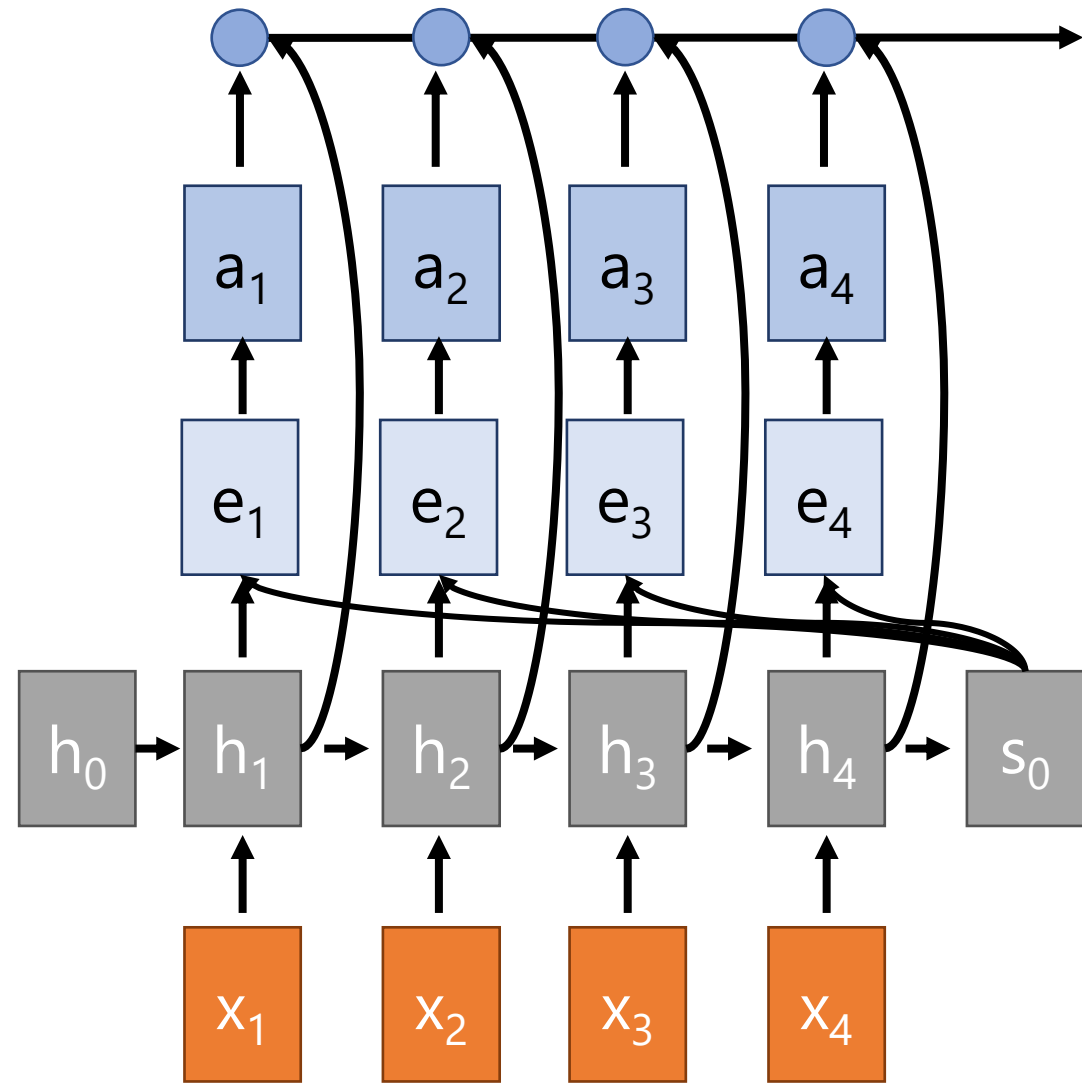
RNN的问题

任务：机器翻译



我们需要记住每一个英文单词。随着序列长度的增加，模型需要记住更多的信息，这使得保持长期依赖变得非常困难，这种现象通常称为“长期依赖问题”。

Self-Attention



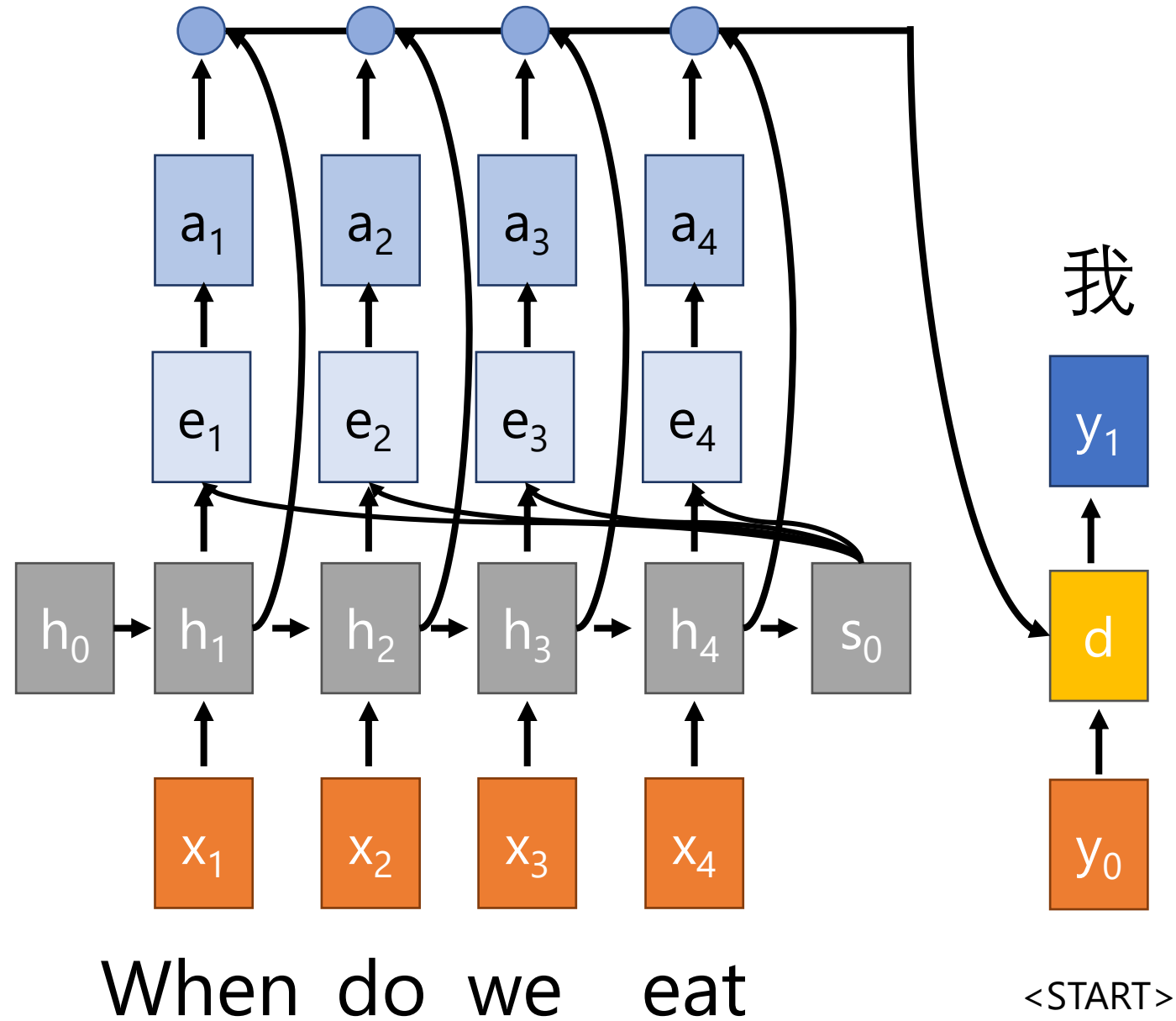
Output: $\sum_i a_i \mathbf{h}_i$

Attention: softmax over e_i
注意力权重

Alignment Score: $e_i = \mathbf{h}_i^T \mathbf{s}_0$
对齐得分

When do we eat

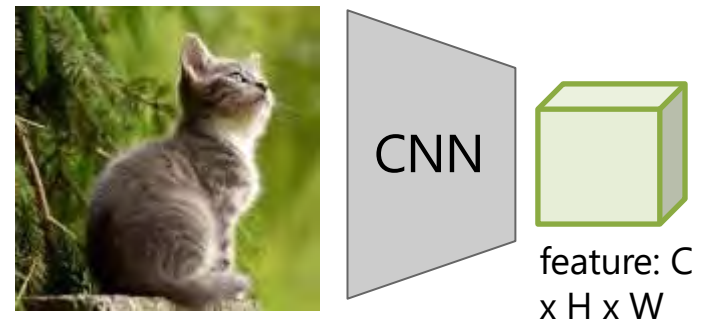
Self-Attention



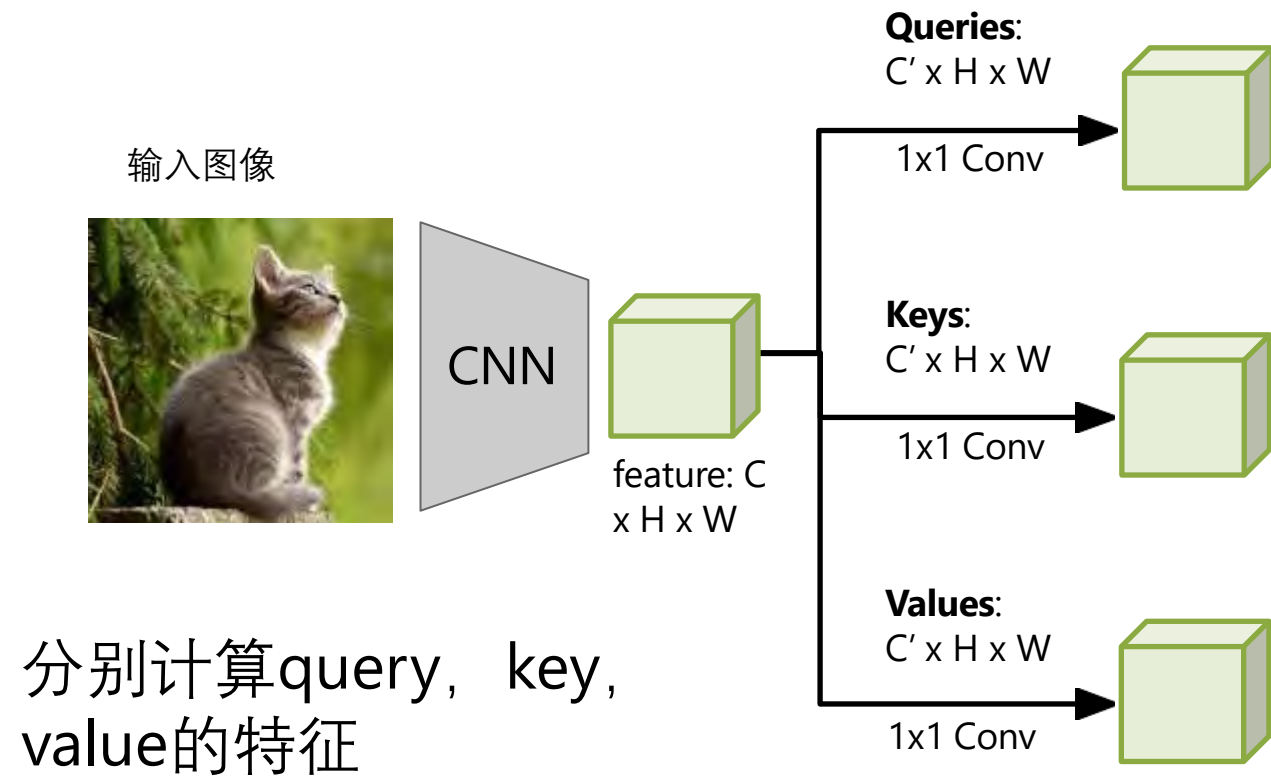
与自注意力机制允许模型在每个时间步都考虑到整个输入序列，而不是仅仅依赖于前一个隐藏状态。这种方法提高了模型捕捉长距离依赖的能力

使用CNN的Self-Attention

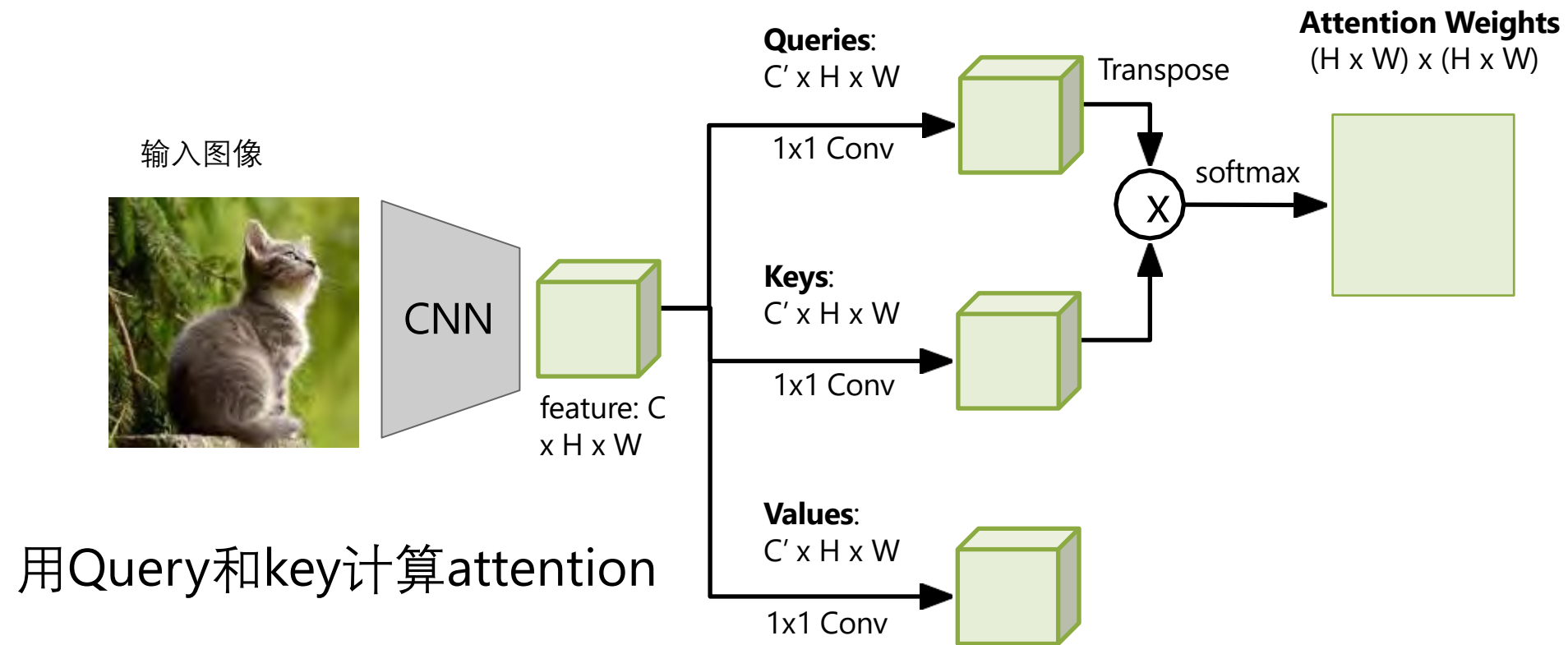
输入图像



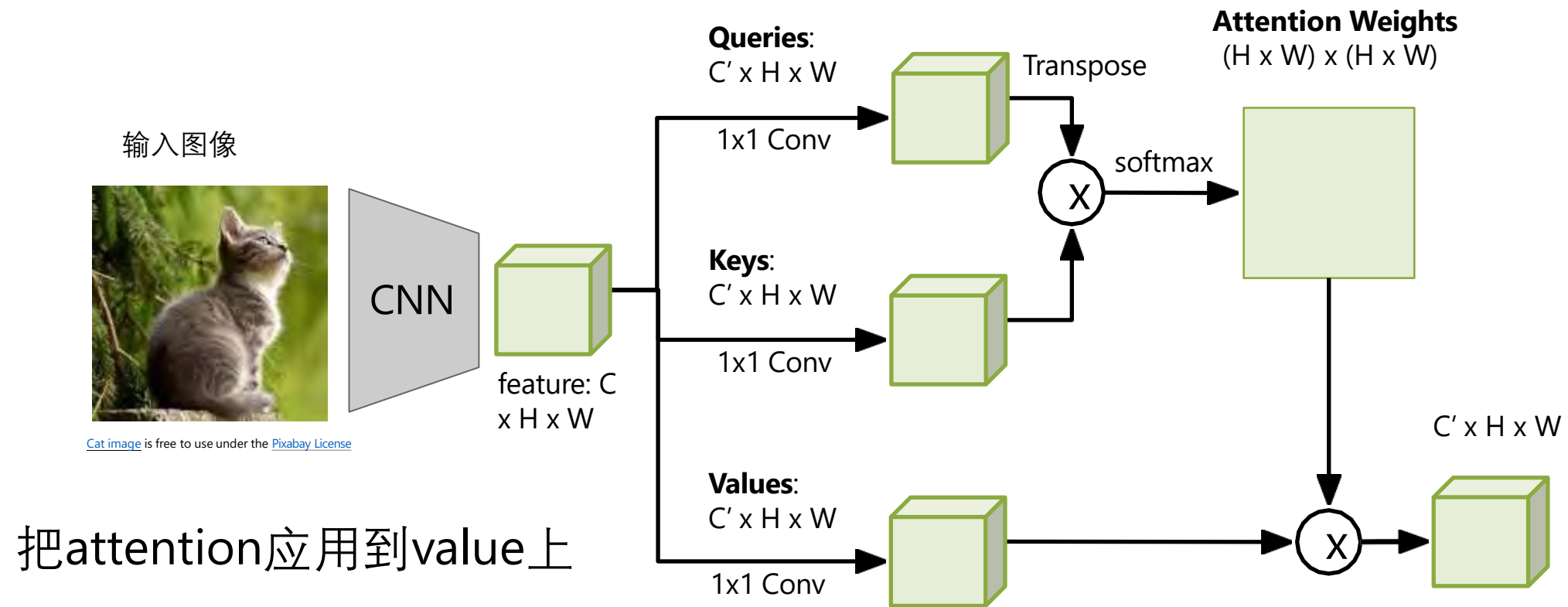
使用CNN的Self-Attention



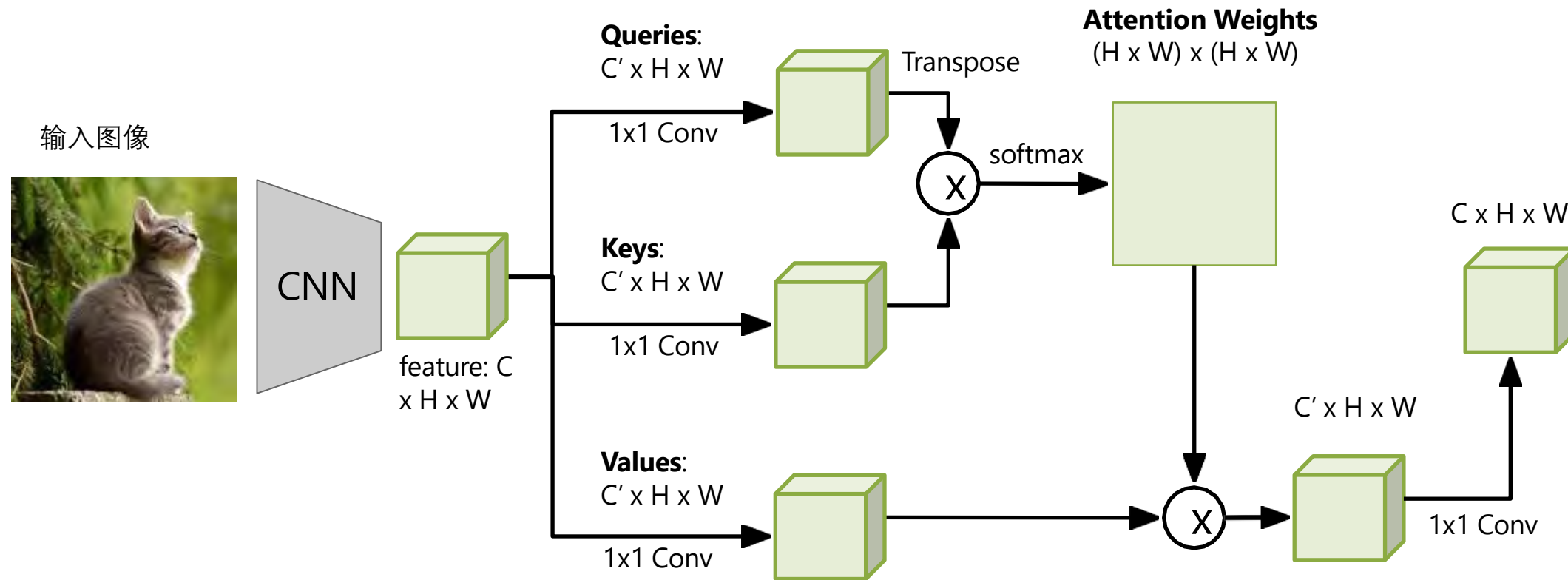
使用CNN的Self-Attention



使用CNN的Self-Attention

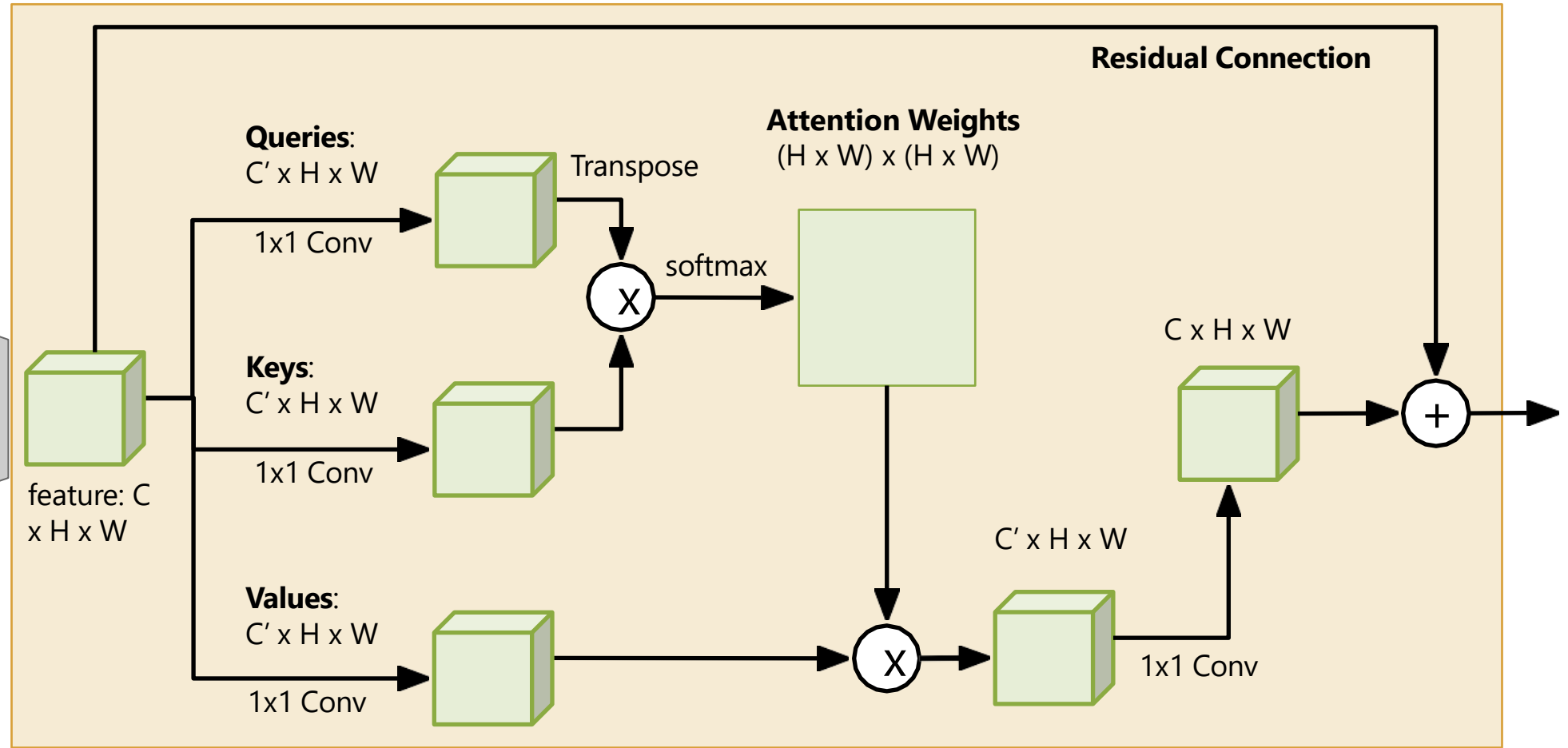
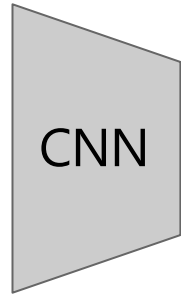


使用CNN的Self-Attention



使用CNN的Self-Attention

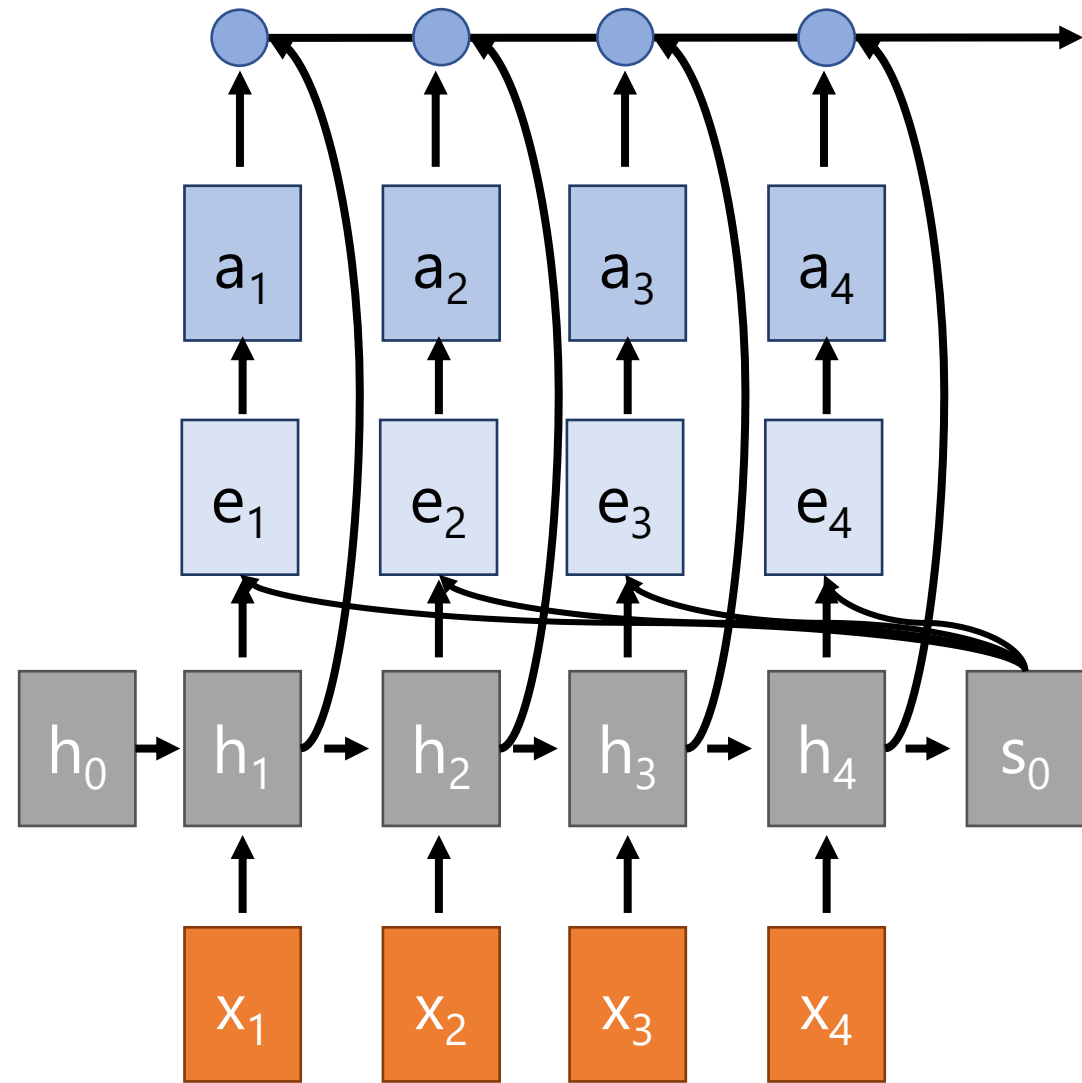
输入图像



以残差形式加到原特征上

Self-Attention Module

Transformers



When do we eat

Diagram is remake of one from J. Johnson

Old Output: $\sum_i a_i \mathbf{h}_i$

New Output: $\sum_i a_i (\mathbf{V} \mathbf{h}_i)$

表示隐藏状态的线性变换，用于修改
隐藏状态之前的加权和

Attention: softmax over e_i

Old Alignment: $e_i = \mathbf{h}_i^T \mathbf{s}_0$

New: 通过 $\mathbf{h}_i^T \mathbf{K} \mathbf{s}_0$ 得到 e_i

输入向量不做处理

New: 对于 F -维向量，用 \sqrt{F}
调整尺度，防止梯度消失

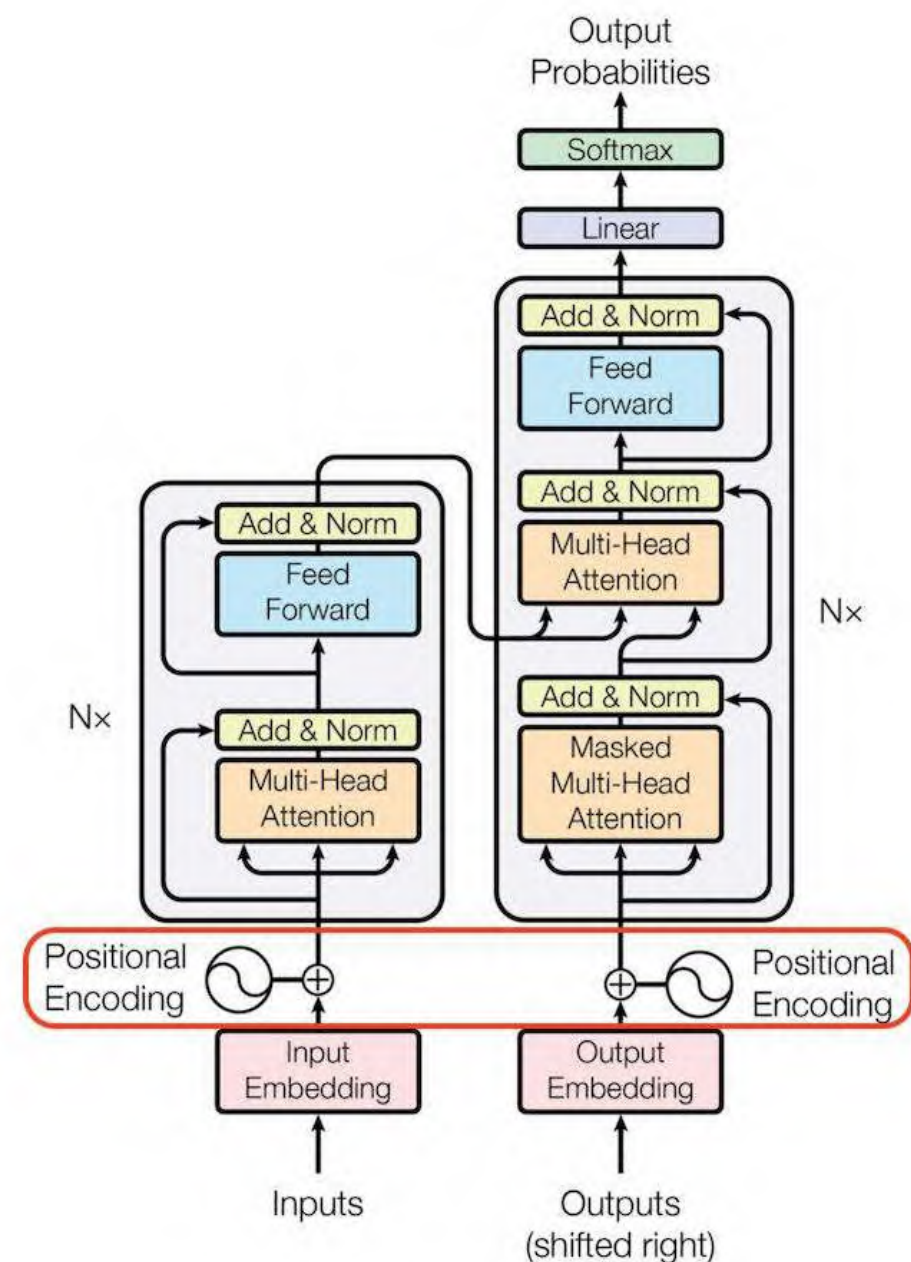
RNNs vs Transformer

- **RNNs**

- (+) 在较长的序列上表现较好.
- (-) 需要顺序序列
- (-) 需要顺序计算.

- **Transformer:**

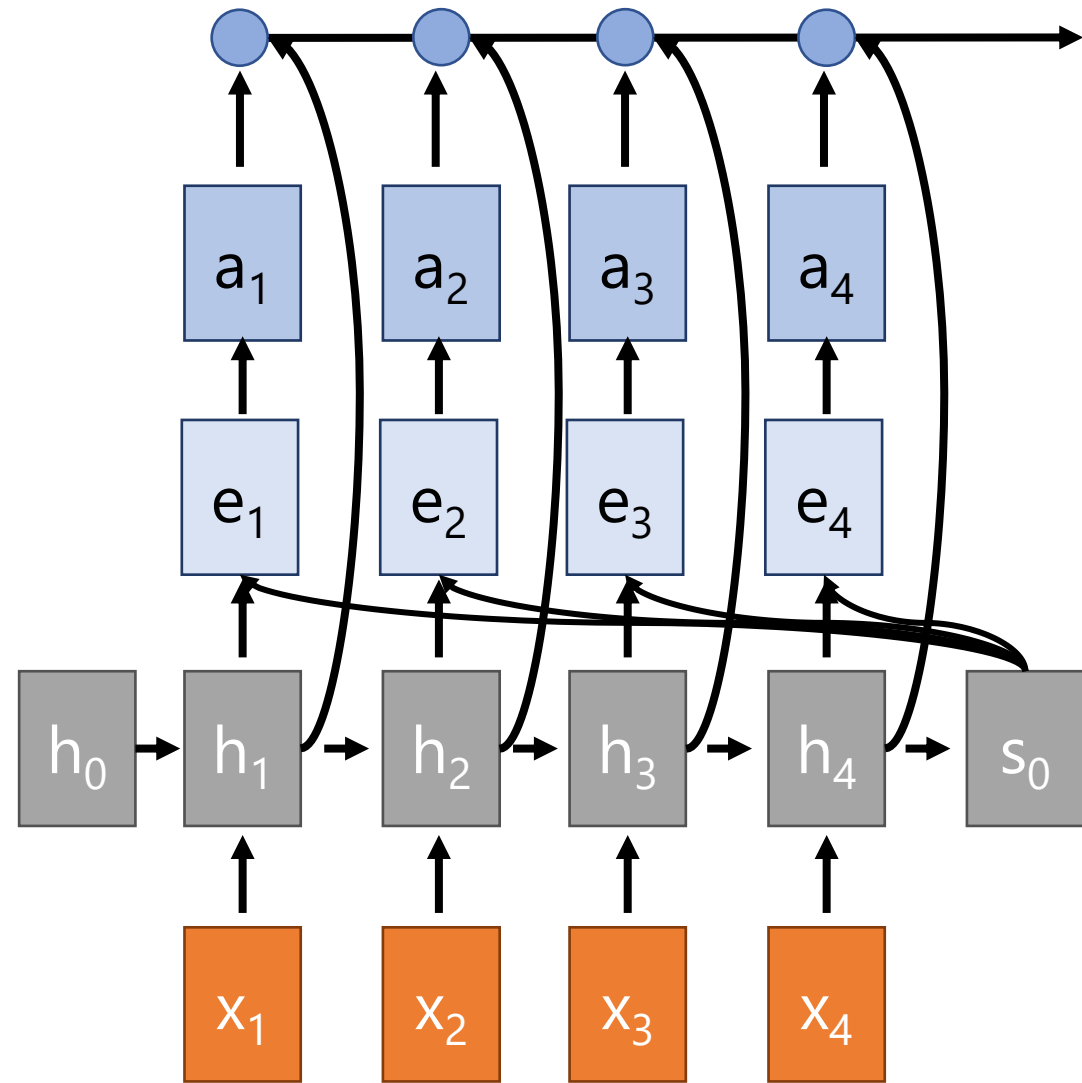
- (+) 同样在较长的序列上表现较好.
- (+) 可以在没有顺序的序列上应用.
- (+) 并行计算.
- (-) 需要大量内存. (但是可以用GPU加速)



Large-scale Language Models 大语言模型

- 在极大的尺度上训练的语言模型
- GPT: Generative Pretrained Transformers
- GPT-3:
 - 410 billion tokens 的训练数据
 - 17 billion 的参数量
- “On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? 🐦” Bender et al.
<https://dl.acm.org/doi/pdf/10.1145/3442188.3445922>

Transformers



When do we eat

Diagram is remake of one from J. Johnson

Alignment score 可以体现
每个**token**的位置吗?

Alignment Score: $e_i = \mathbf{h}_i^T \mathbf{s}_0$

Solution: 把位置信息加入到计算alignment score的过程中

Positional Encodings 位置编码

位置编码 (Positional Encoding)，其目的是给模型提供序列中各个元素位置的信息。 $f(p, 2i)$ 和 $f(p, 2i + 1)$ 分别定义了每个位置 p 的编码，其中 i 是维度索引，而 d 是编码的维度总数。对于偶数维度 (使用 $2i$)，位置 p 的编码是通过正弦函数来计算的；对于奇数维度 (使用 $2i + 1$)，使用余弦函数。

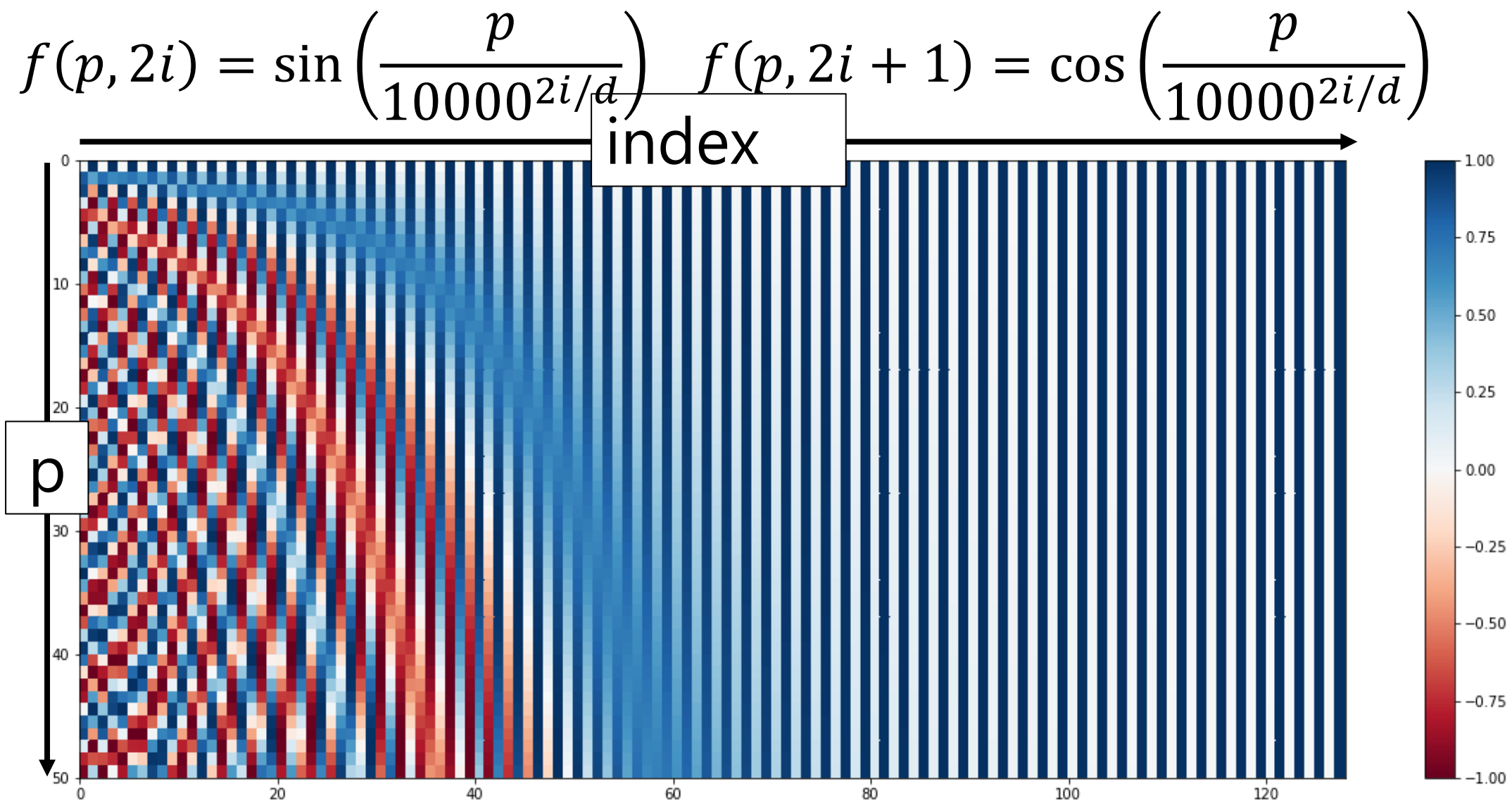


Diagram: Amirhoseein Kazemnejad,
https://kazemnejad.com/blog/transformer_architecture_positional_encoding/

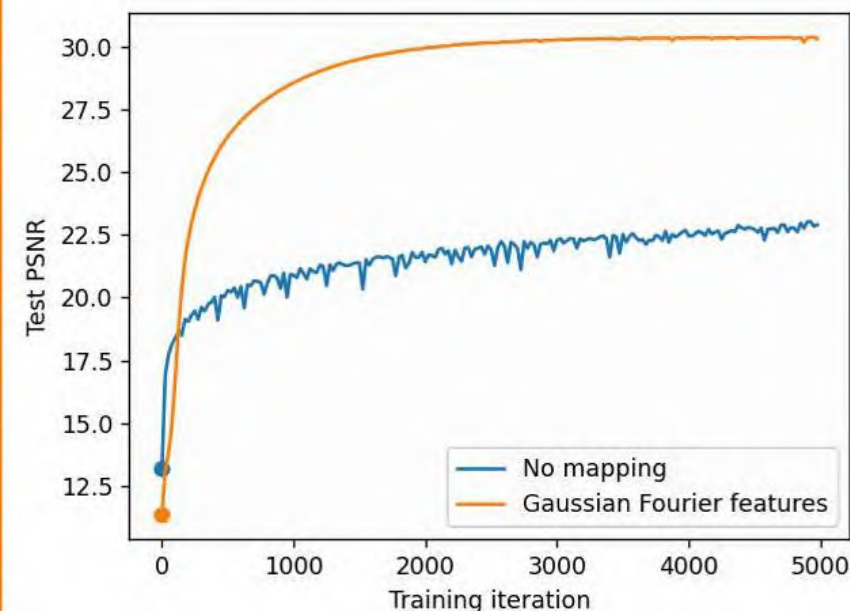
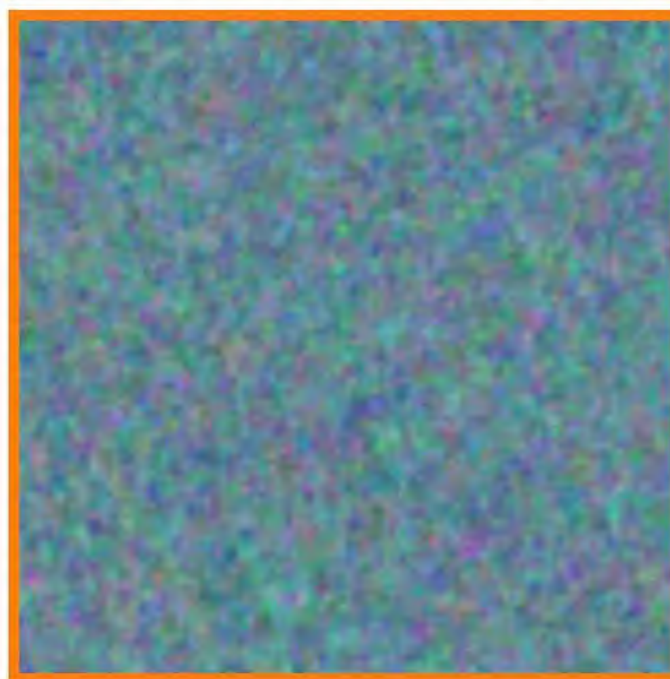
Positional Encodings 与 图像

需要学习从 (x,y) 到 (r,g,b) 的映射

Learned on
 (x,y)



w/positional
encodings)



Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains, Tancik et al. NeurIPS 2020. See also Implicit Neural Representations with Periodic Activation Functions, Sitzmann et al. NeurIPS 2020.

怎样把Transformer用到图像上?

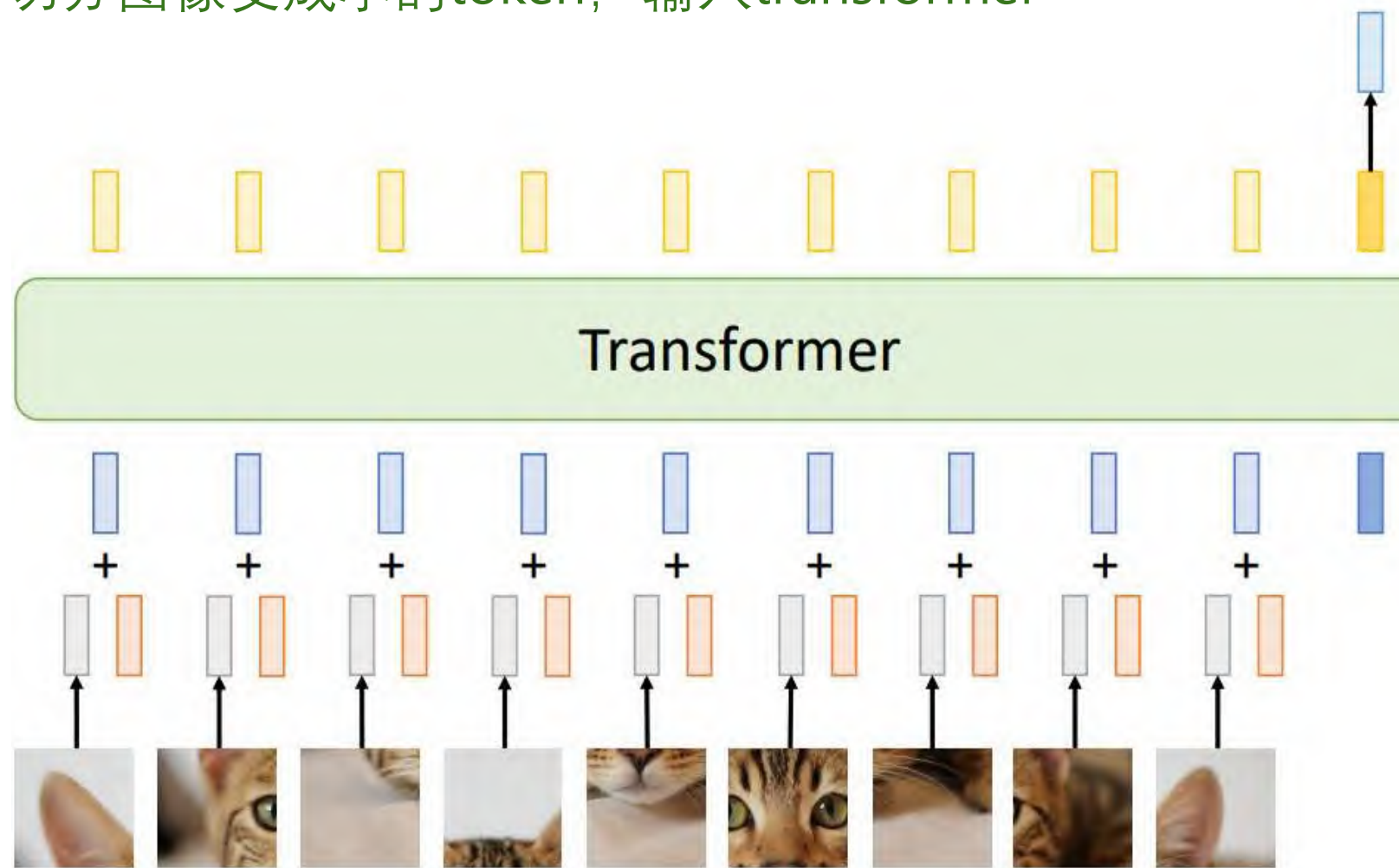


把图像切成小的局部，然后放到Transformer里

Vision Transformers



切分图像变成小的token, 输入transformer

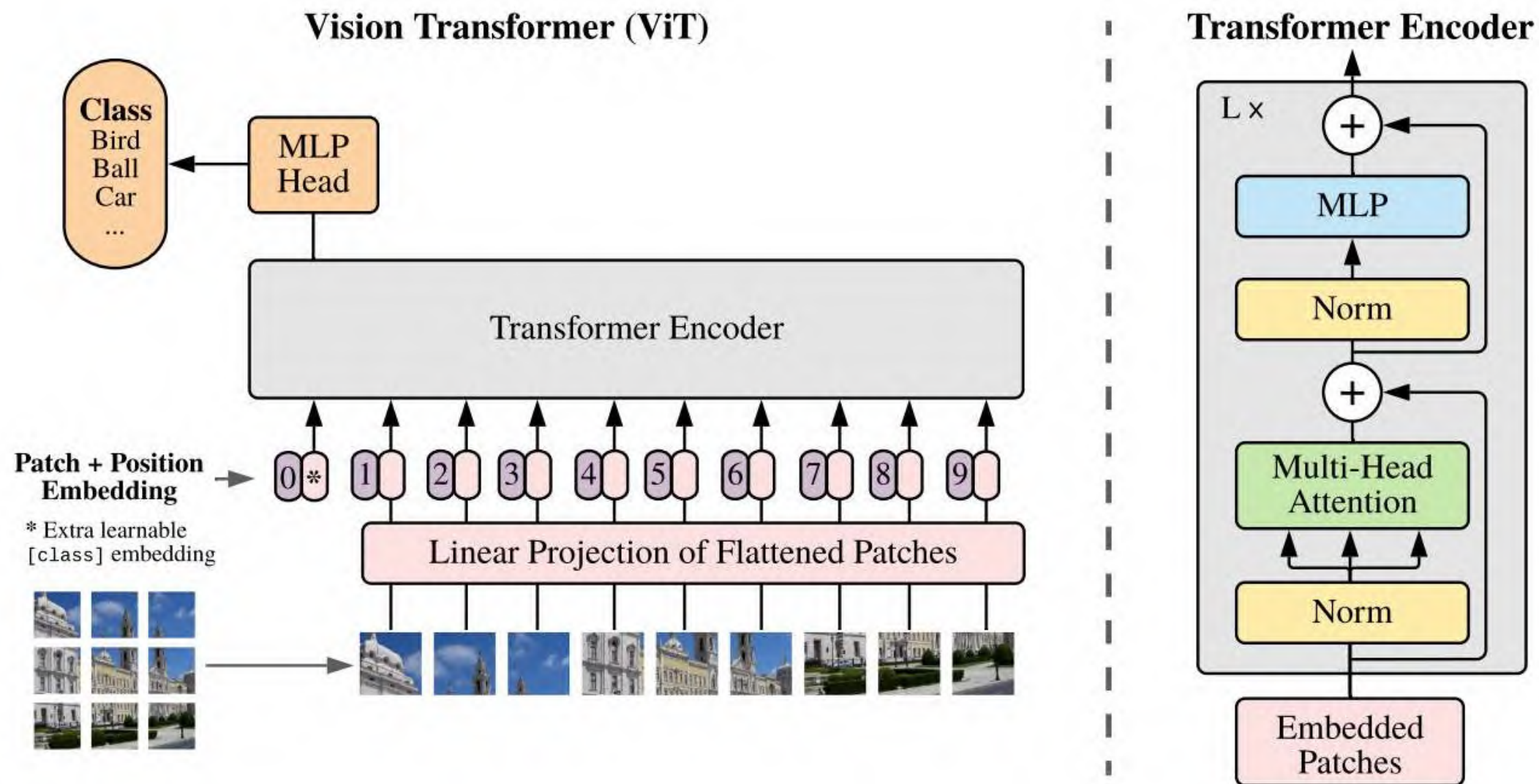


Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ArXiv 2020

[Colab link](#) to an implementation of vision transformers

Vision Transformer (ViT)

核心思想: 把图像转换为image tokens



ViTs vs. ResNets

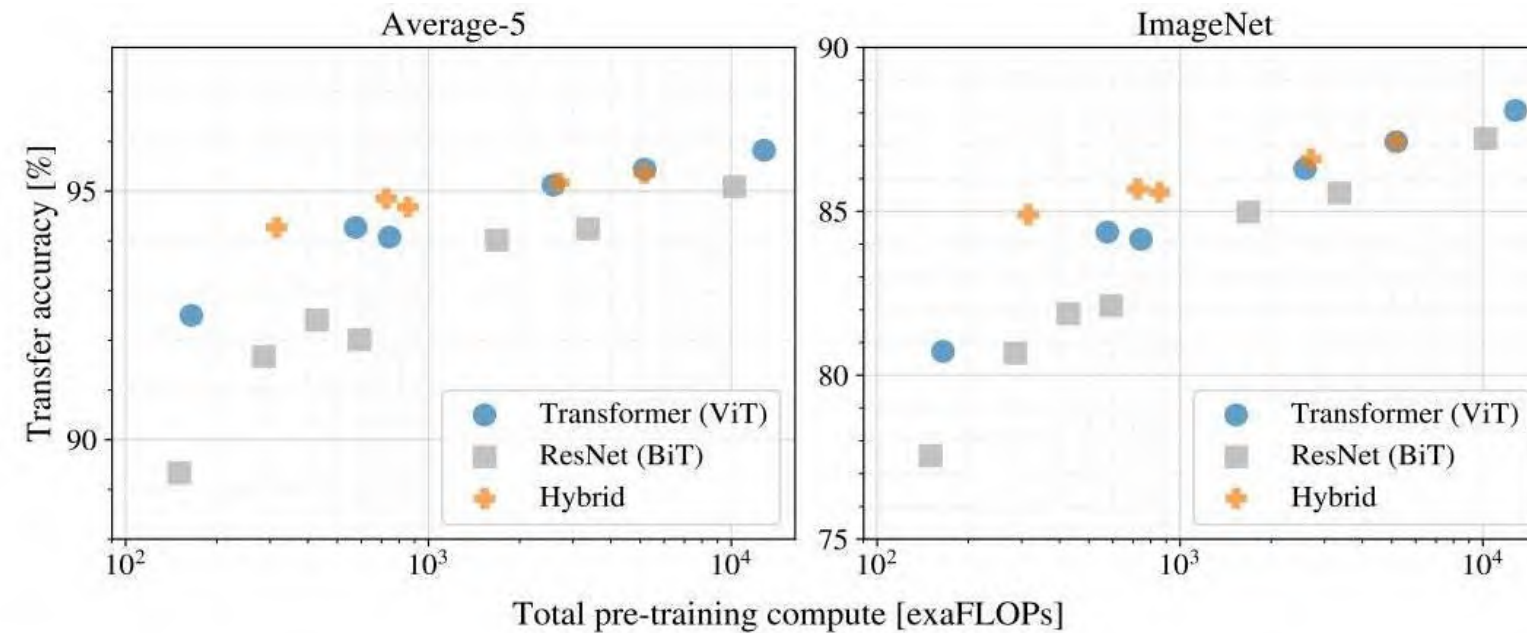
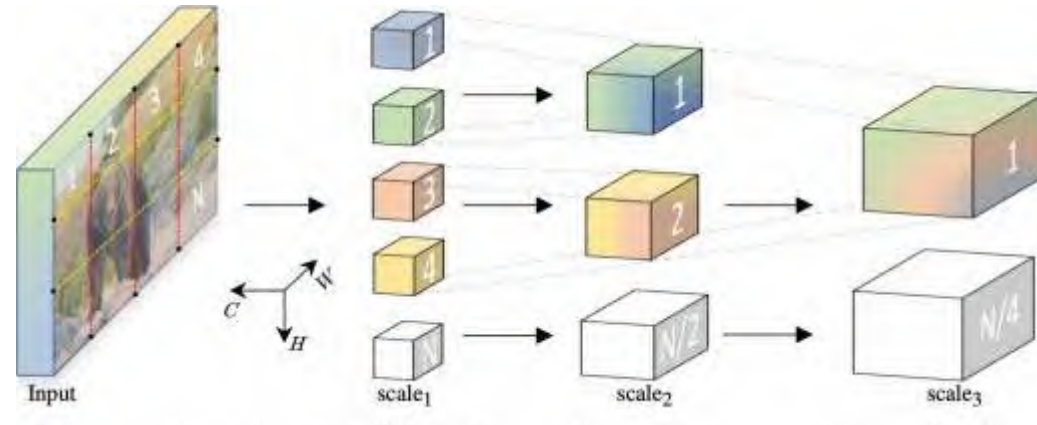


Figure 5: Performance versus cost for different architectures: Vision Transformers, ResNets, and hybrids. Vision Transformers generally outperform ResNets with the same computational budget. Hybrids improve upon pure Transformers for smaller model sizes, but the gap vanishes for larger models.

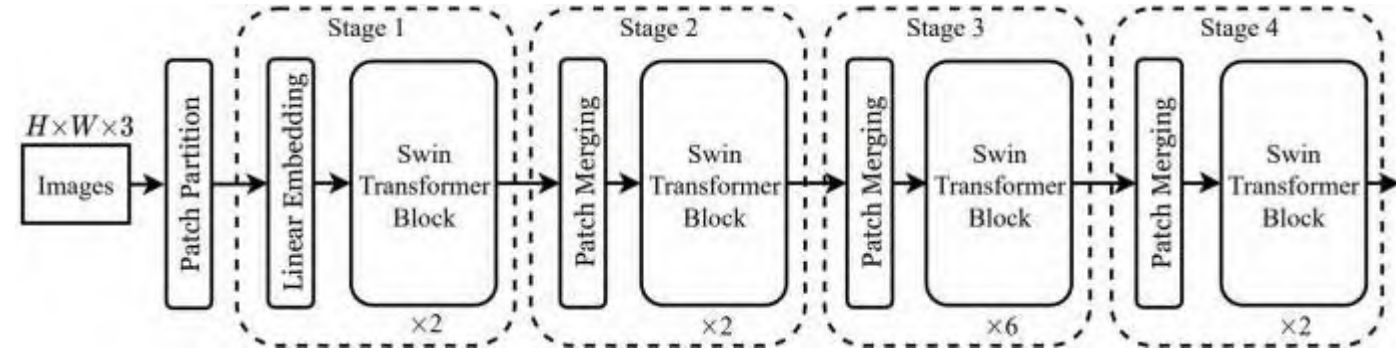
Dosovitskiy et al, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale", ArXiv 2020

[Colab link](#) to an implementation of vision transformers

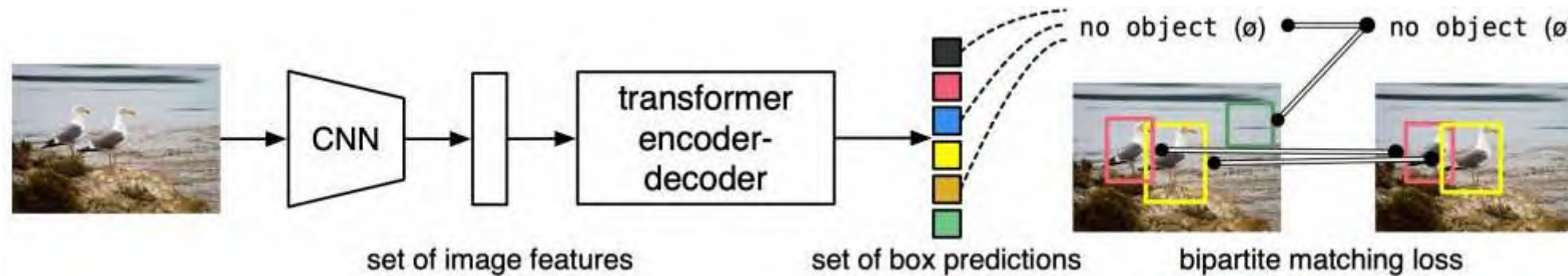
Vision Transformers (ViT)



Fan et al, "Multiscale Vision Transformers", ICCV 2021

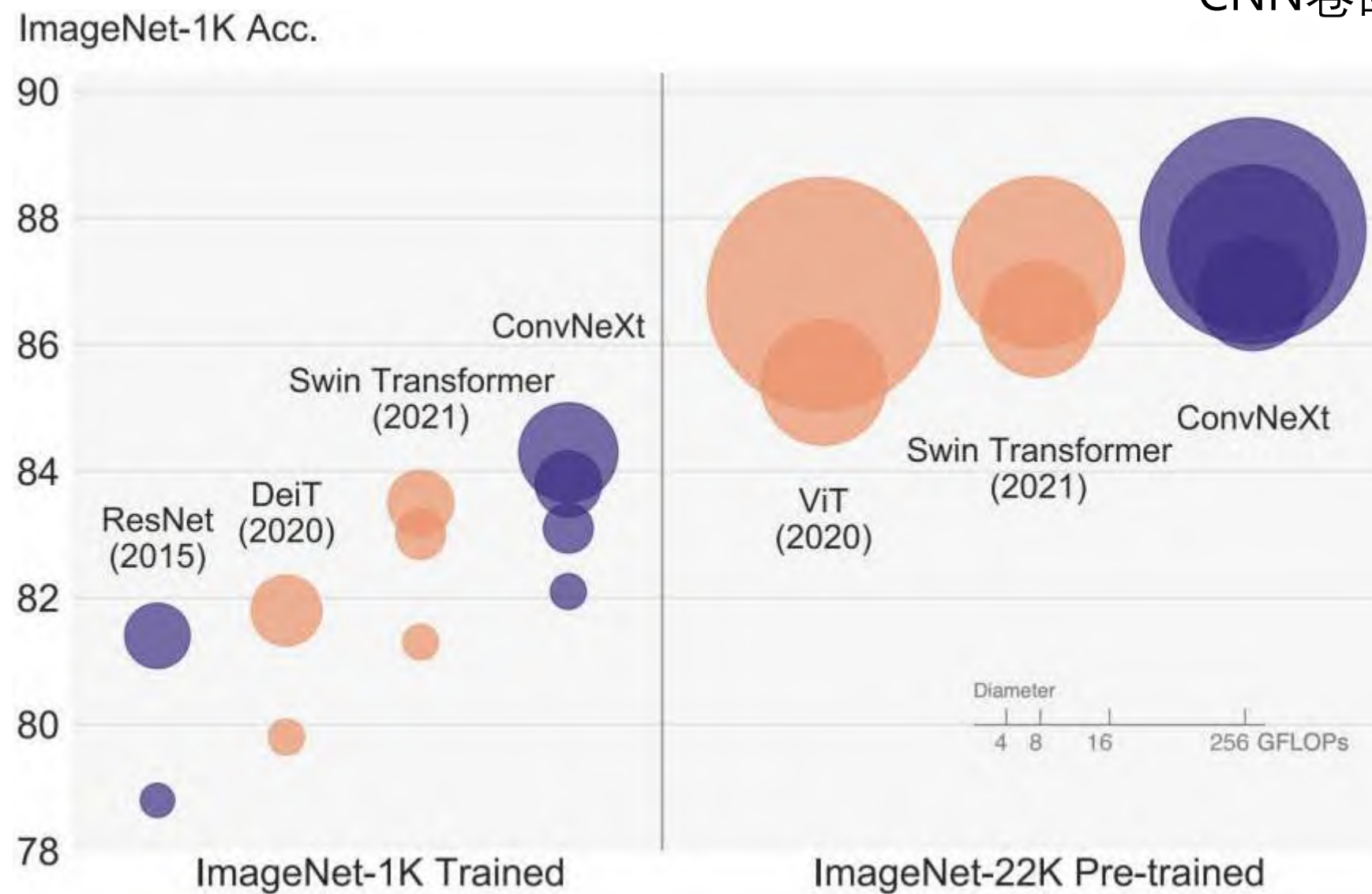


Liu et al, "Swin Transformer: Hierarchical Vision Transformer using Shifted Windows", CVPR 2021



Carion et al, "End-to-End object detection with Transformers", ECCV 2020

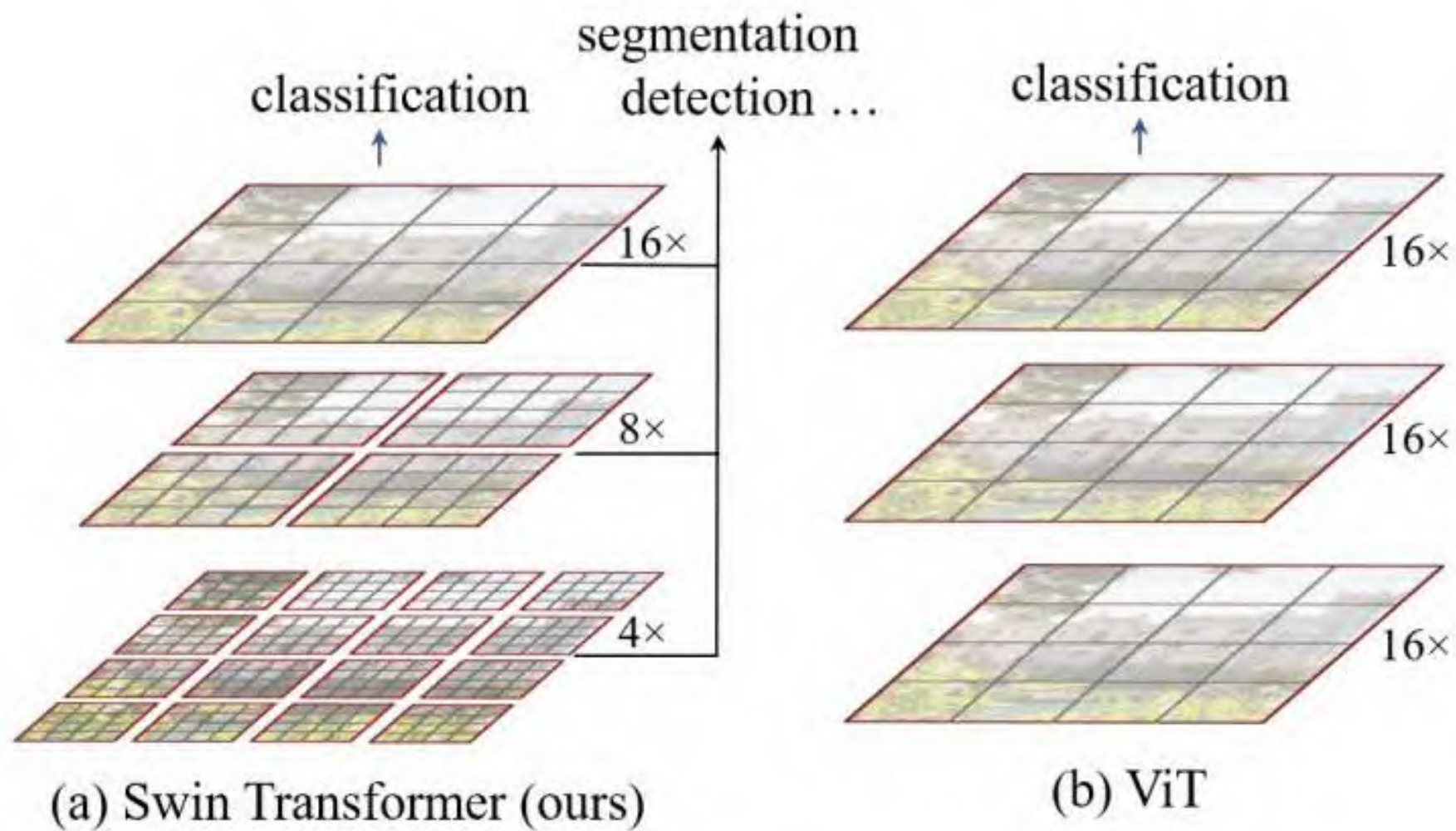
CNN卷回来了!



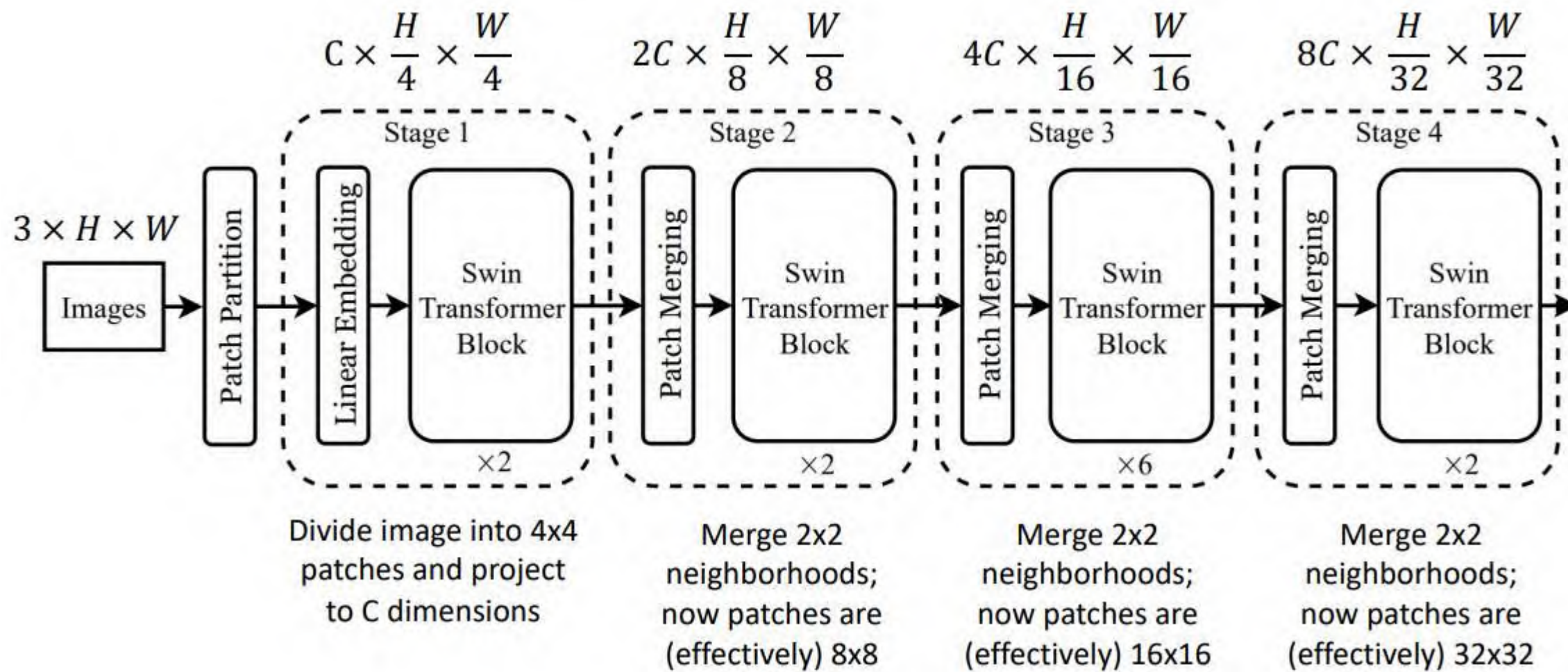
A ConvNet for the 2020s. Liu et al. CVPR 2022

Swin Transformers

分阶段在不同尺度内部做transformer

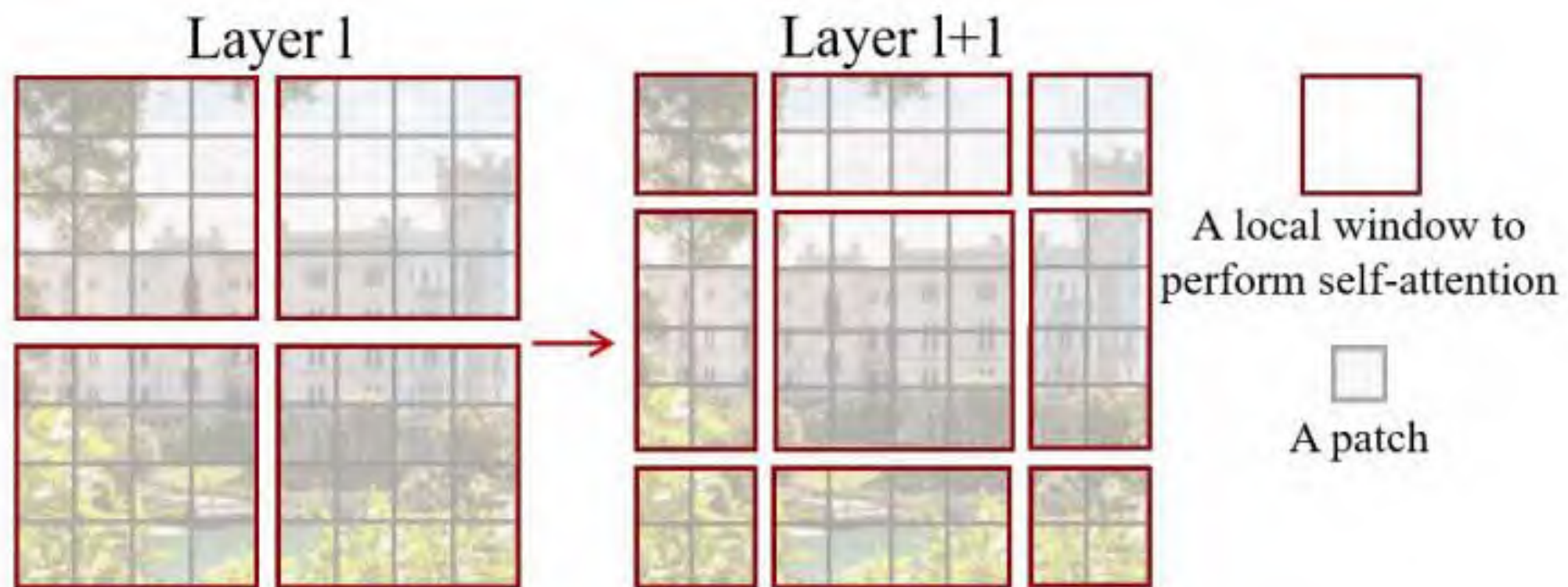


Swin Transformers



Swin Transformers

Shift window 避免信息只在固定窗口传递



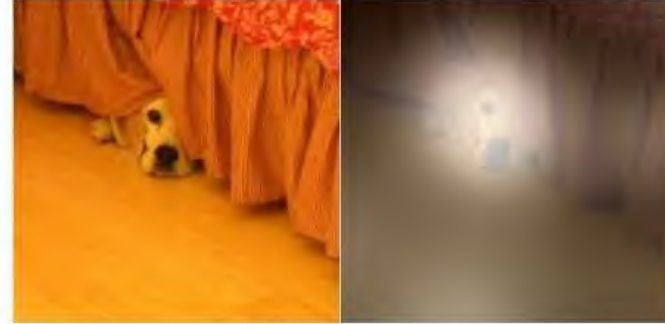
回顾：VQA、Transformer

图像内容: Attention

注意力机制会让我们关注不同的空间区域



A woman is throwing a frisbee in a park.



A dog is standing on a hardwood floor.



A stop sign is on a road with a mountain in the background.



A little girl sitting on a bed with a teddy bear.



A group of people sitting on a boat in the water.



A giraffe standing in a forest with trees in the background.

Xu et al, "Show, Attend, and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

Figure copyright Kelvin Xu, Jimmy Lei Ba, Jamie Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio, 2015. Reproduced with permission.

检索结果

- 我们可能不太喜欢检索的结果
- 因为它们没有创造新的句子!



A wooden bench in front of a building.

A window display on the front of a building.



A building with a clock on the top.

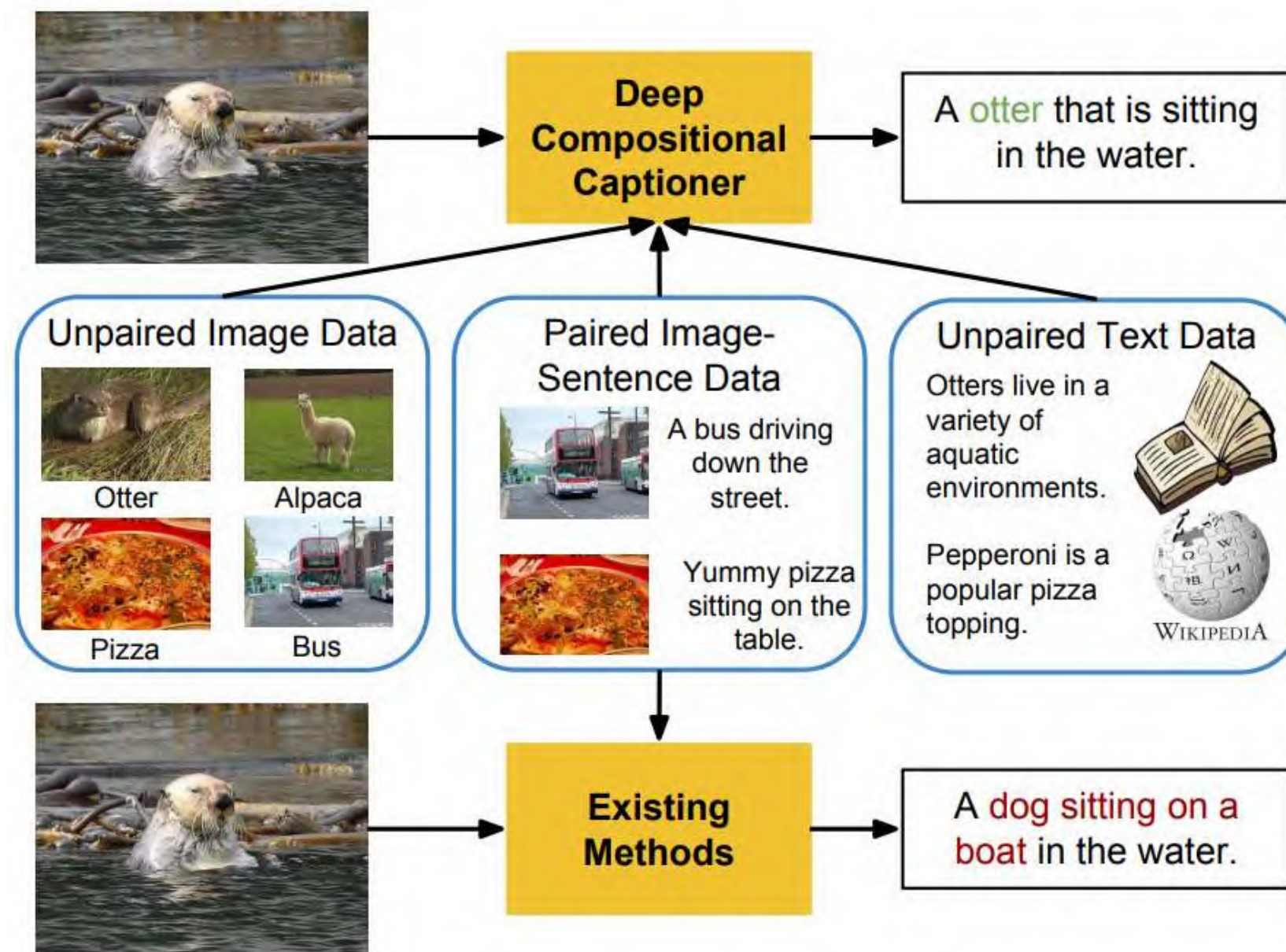
A clock tower on the top of a building.



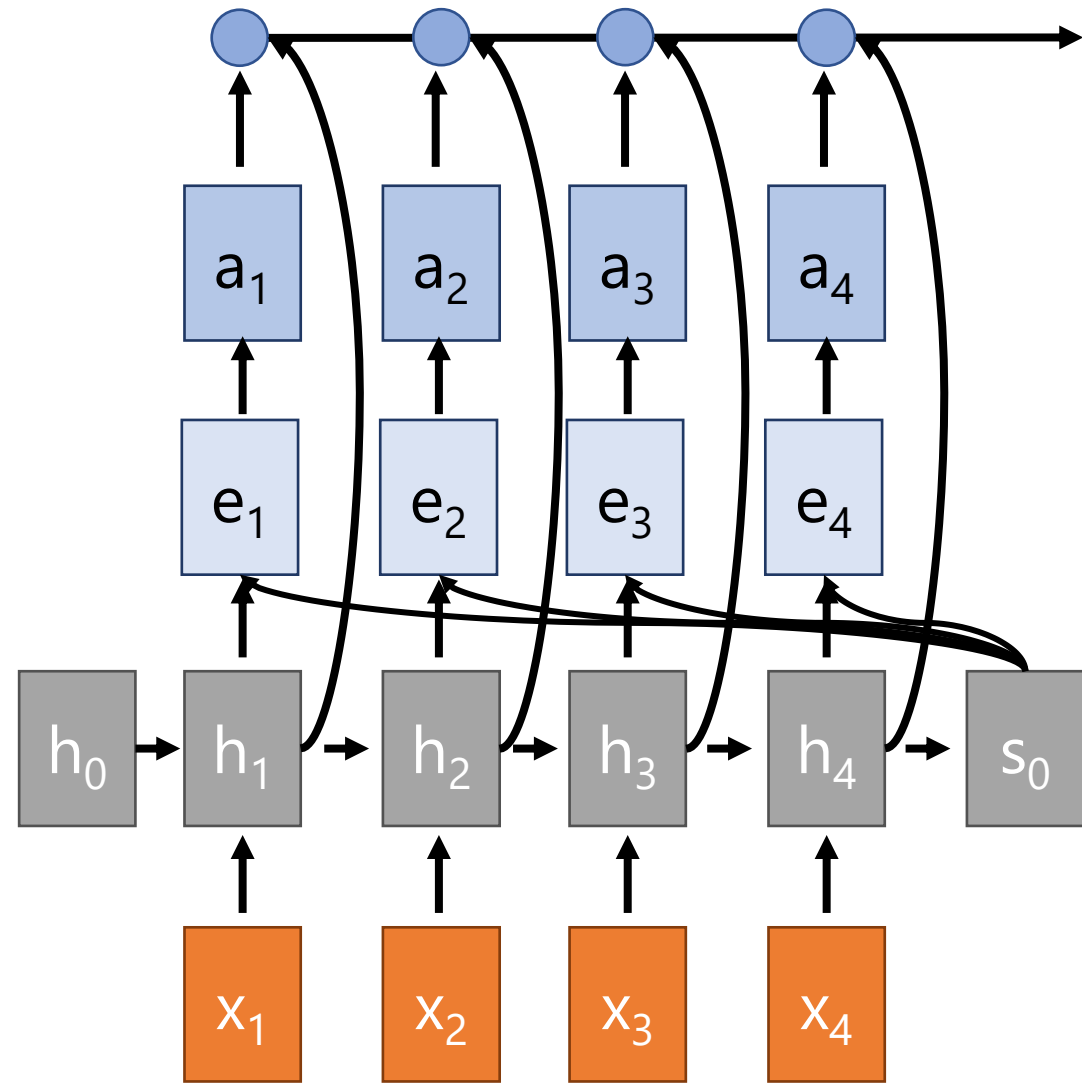
The side of a passenger train at a train station.

A bus that is on the side of a road.

怎么获得新的描述? Unpaired Data



Self-Attention



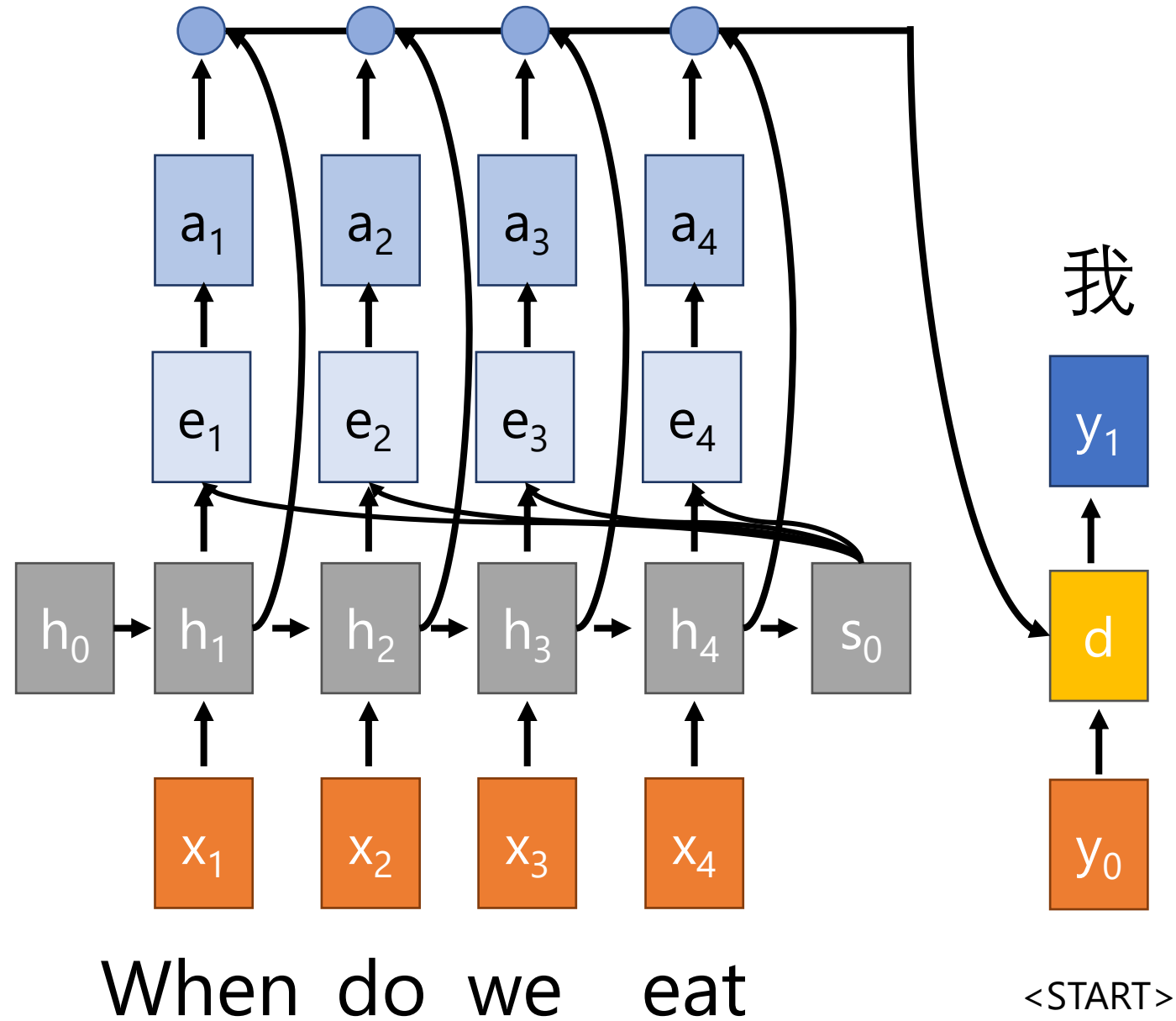
Output: $\sum_i a_i h_i$

Attention: softmax over e_i
注意力权重

Alignment Score: $e_i = h_i^T s_0$
对齐得分

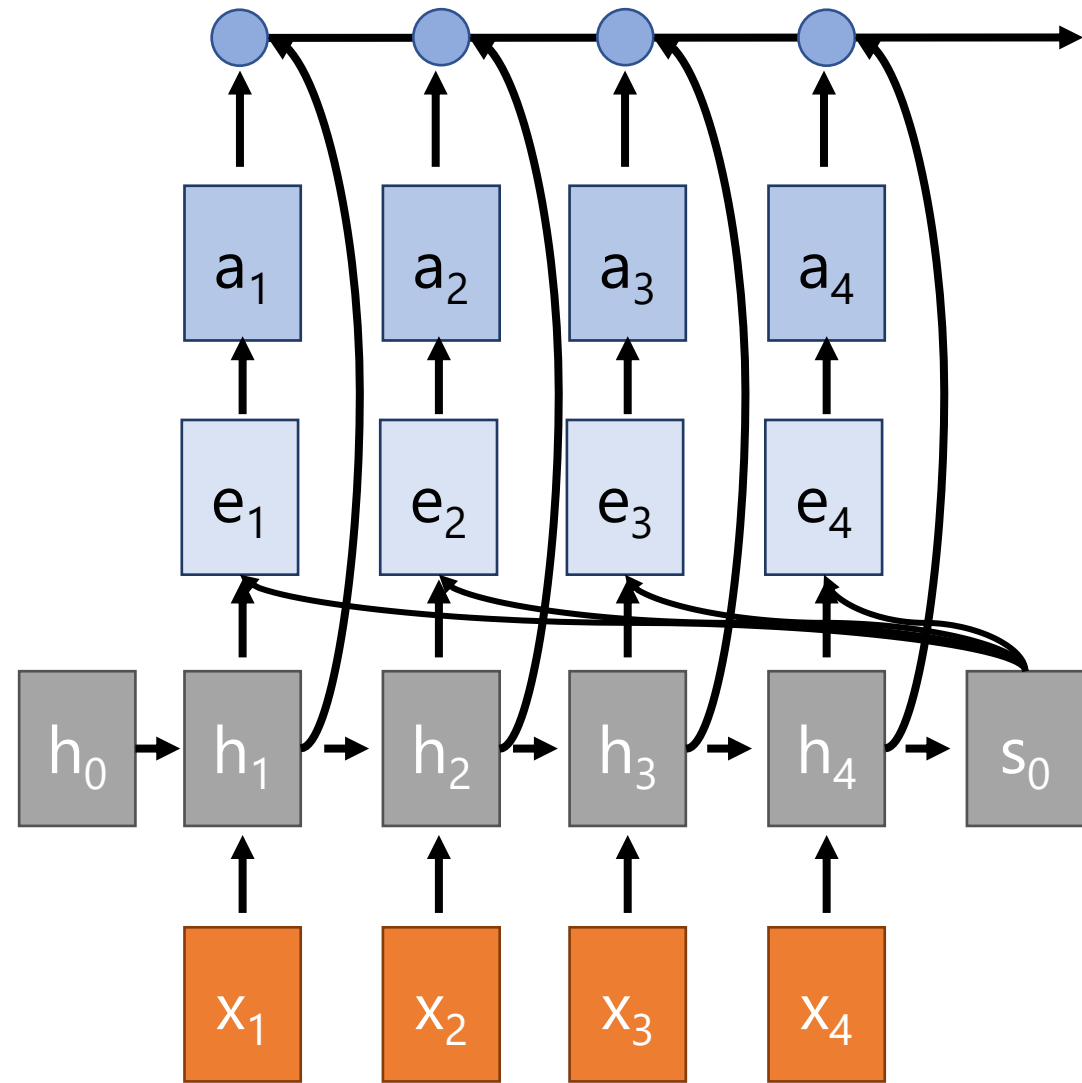
When do we eat

Self-Attention



与自注意力机制允许模型在每个时间步都考虑到整个输入序列，而不是仅仅依赖于前一个隐藏状态。这种方法提高了模型捕捉长距离依赖的能力

Transformers



When do we eat

Diagram is remake of one from J. Johnson

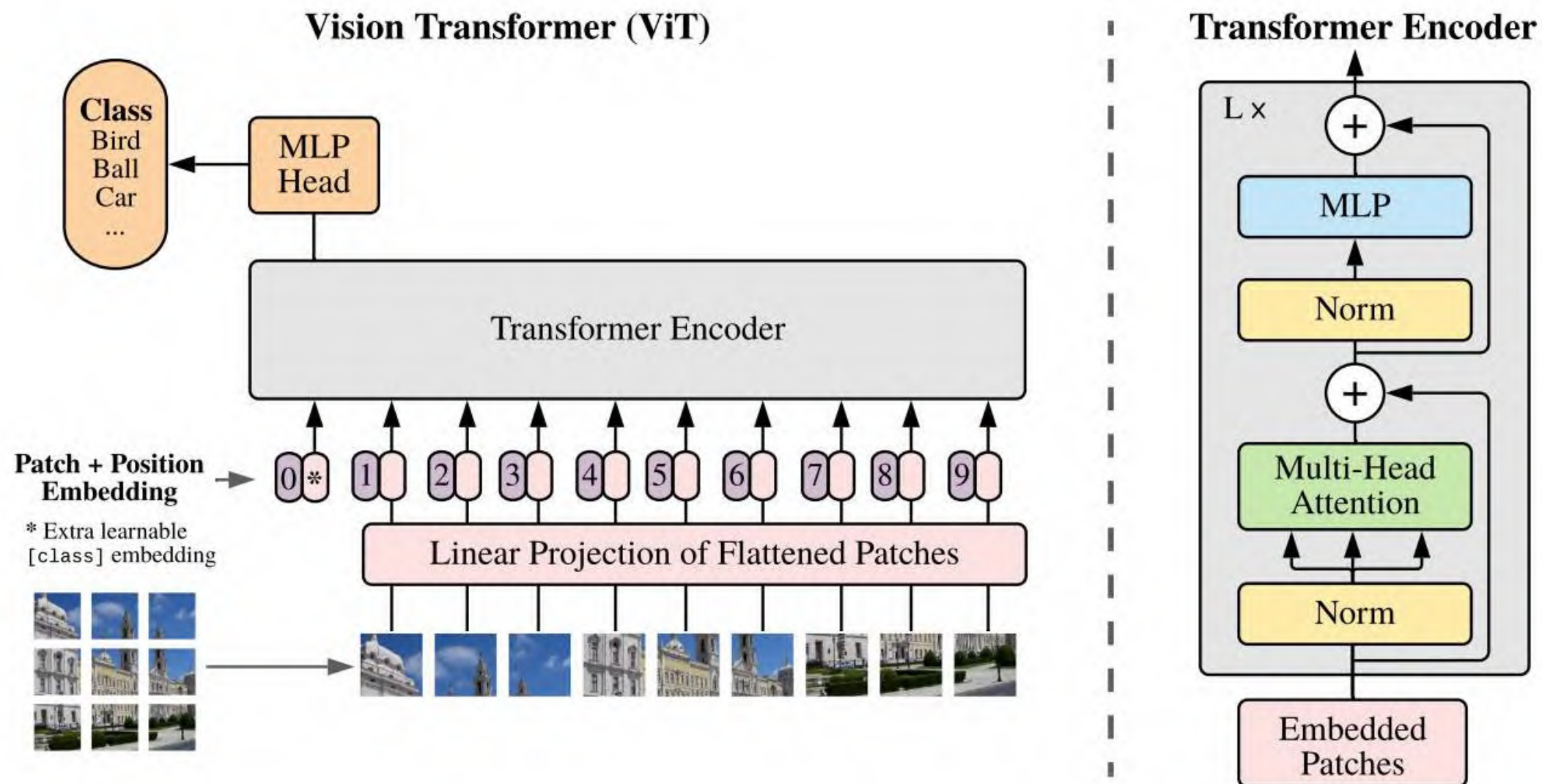
Alignment score 可以体现
每个**token**的位置吗?

Alignment Score: $e_i = \mathbf{h}_i^T \mathbf{s}_0$

Solution: 把位置信息加入到计算alignment score的过程中

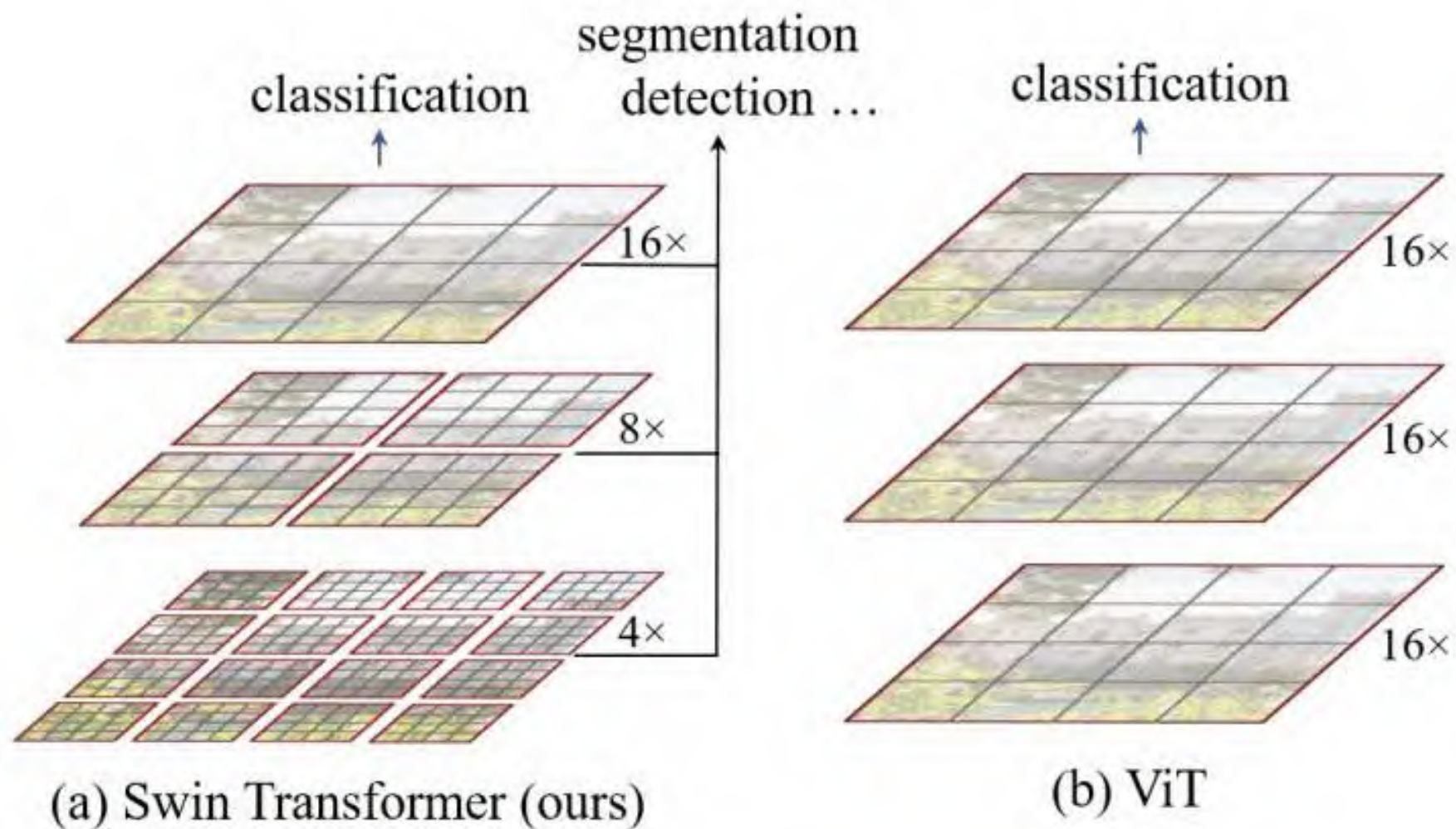
Vision Transformer (ViT)

核心思想: 把图像转换为image tokens



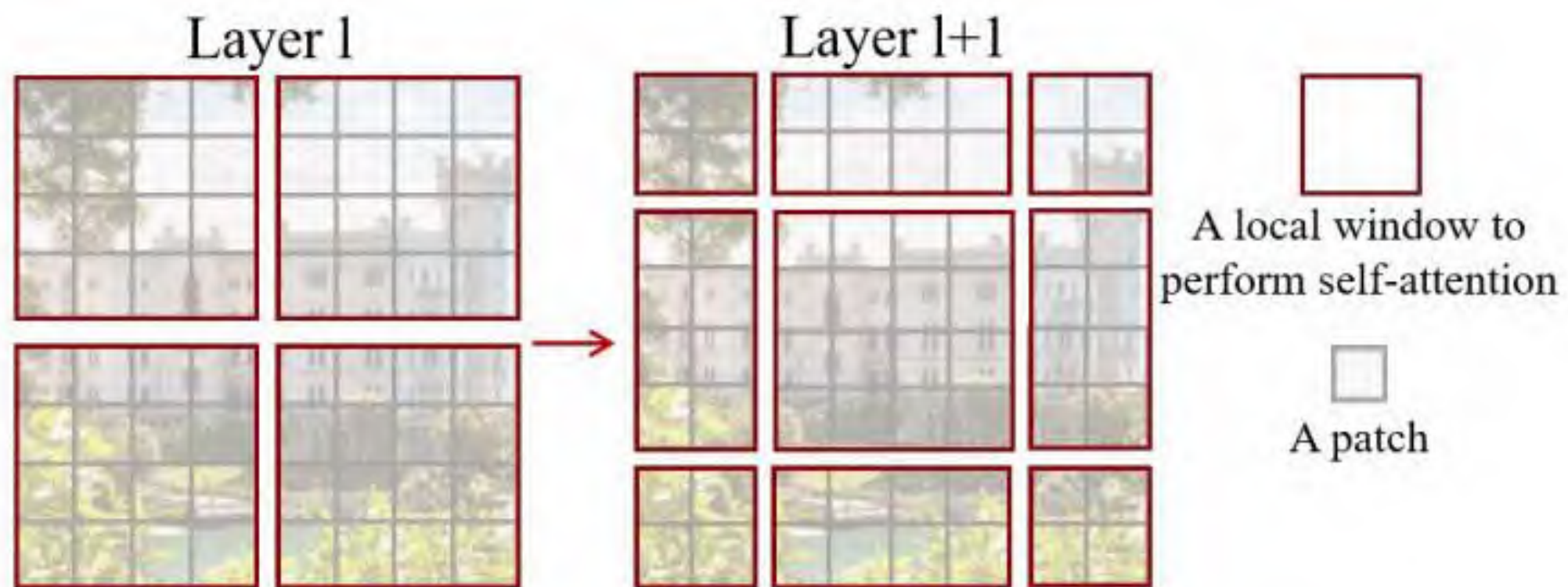
Swin Transformers

分阶段在不同尺度内部做transformer



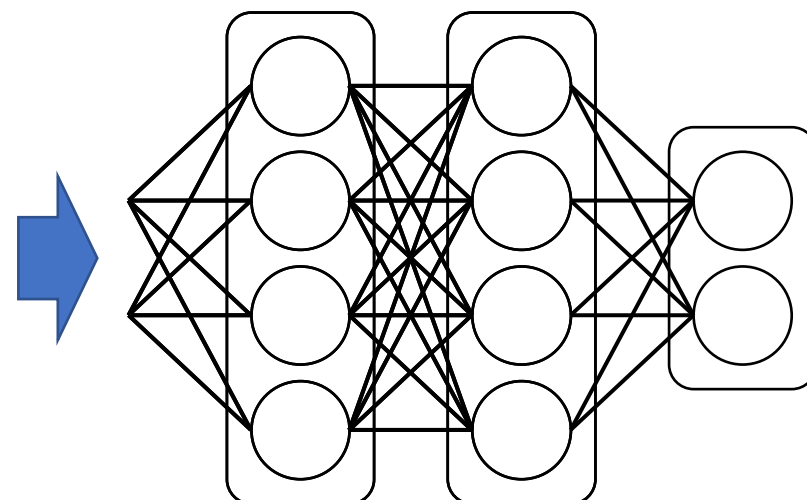
Swin Transformers

Shift window 避免信息只在固定窗口传递



自监督学习

ImageNet 预训练 + 微调

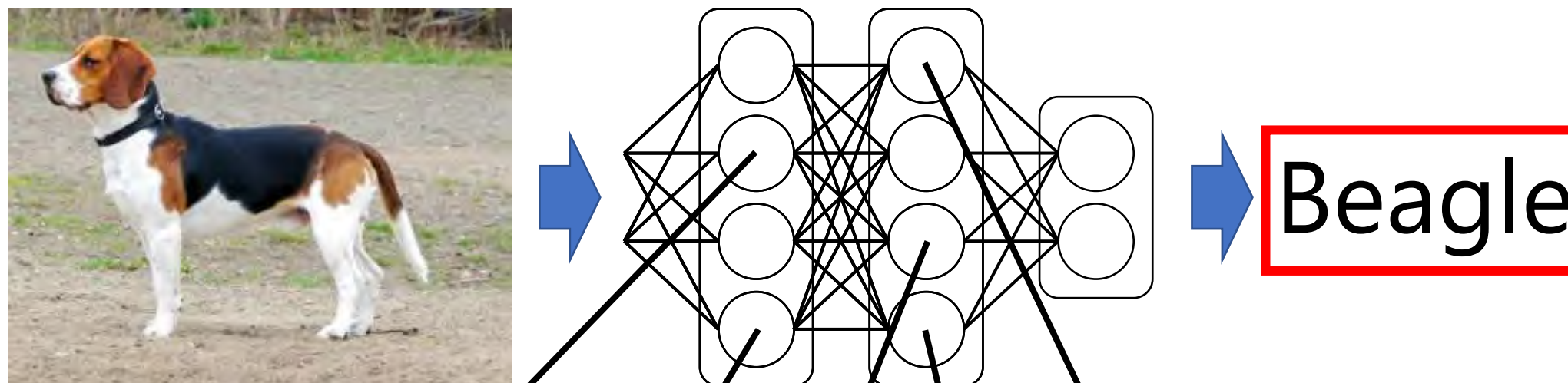


Beagle



- 图像检索
- 物体检测
- 语义分割
- 深度估计
- ...

我们学到了什么？



材质?

部件?

姿态?

几何形状?

边缘?

这些任务很难标注！我们需要做这些任务吗？

我们需要语义标注吗？

Context as Supervision 上下文监督

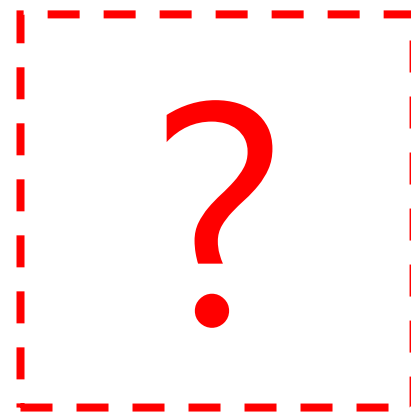
[Collobert & Weston 2008; Mikolov et al. 2013]

BERT 与 GPT 的核心概念之一

house, where the professor lived without his wife and child; or so he said jokingly sometimes: "Here's where I live. My house." His daughter often added, without resentment, for the visitor's information, "It started out to be for me, but it's really his." And she might reach in to bring forth an inch-high table lamp with fluted shade, or a blue dish the size of her little fingernail, marked "Kitty" and half full of eternal smile, but she was sure to replace these, after they had been admired, pretty near exactly where they had been. The little house was very orderly, and just big enough for all it contained, though to some tastes the bric-à-brac in the parlor might seem excessive. The daughter's preference was for the store-bought gimmicks and appliances, the toasters and carpet sweeper, but she knew that most adult visitors would

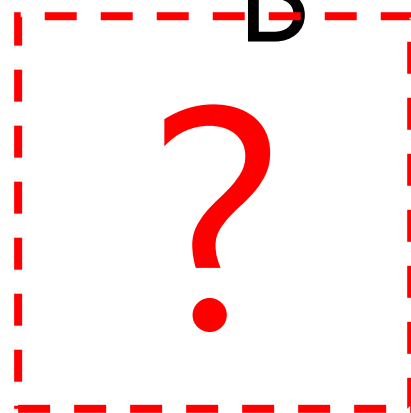
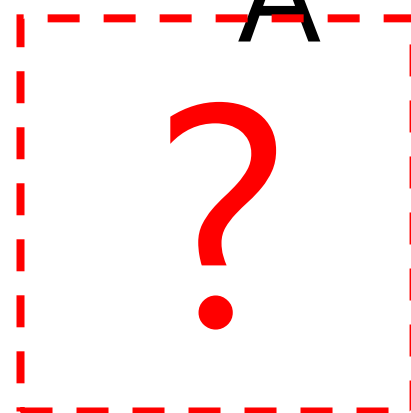
Deep
Net

图像的上下文预测

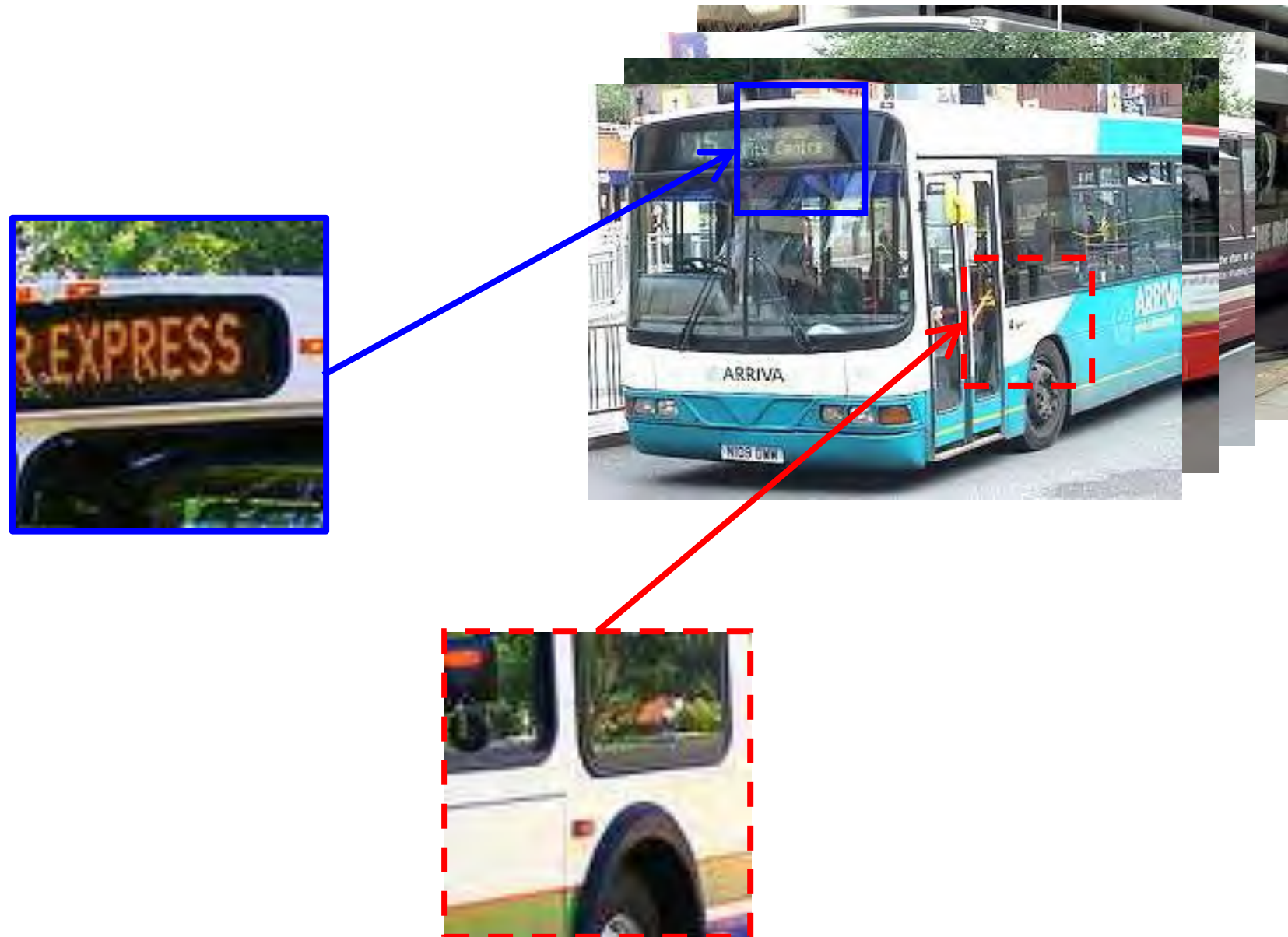


A

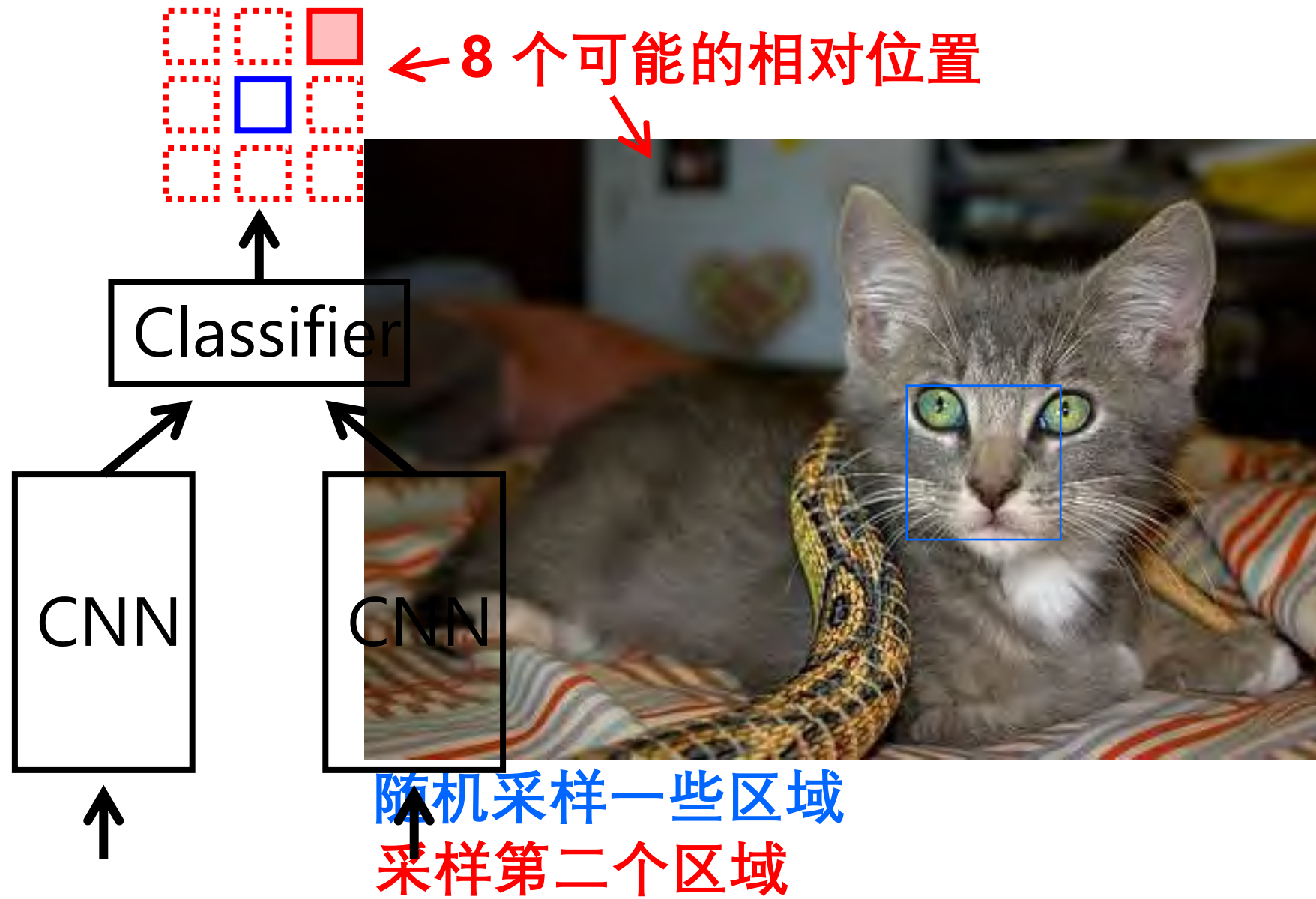
B

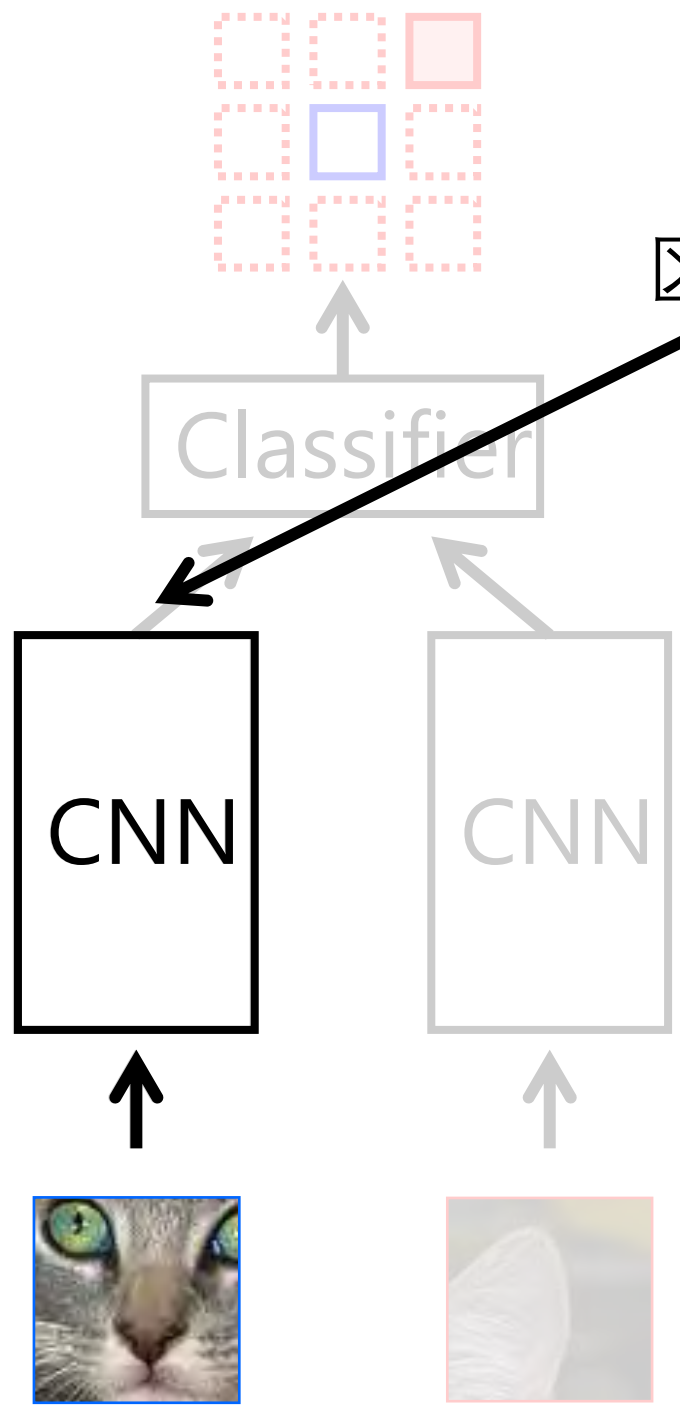


这样做能得到什么“语义”信息？



相对位置预测





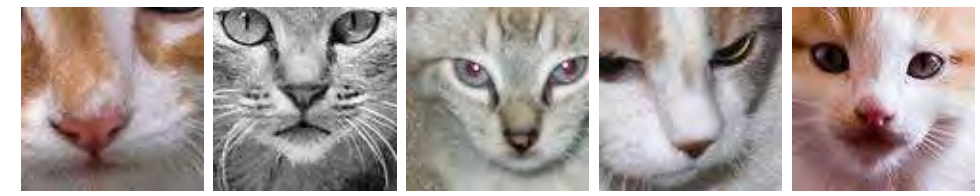
区域特征

Input



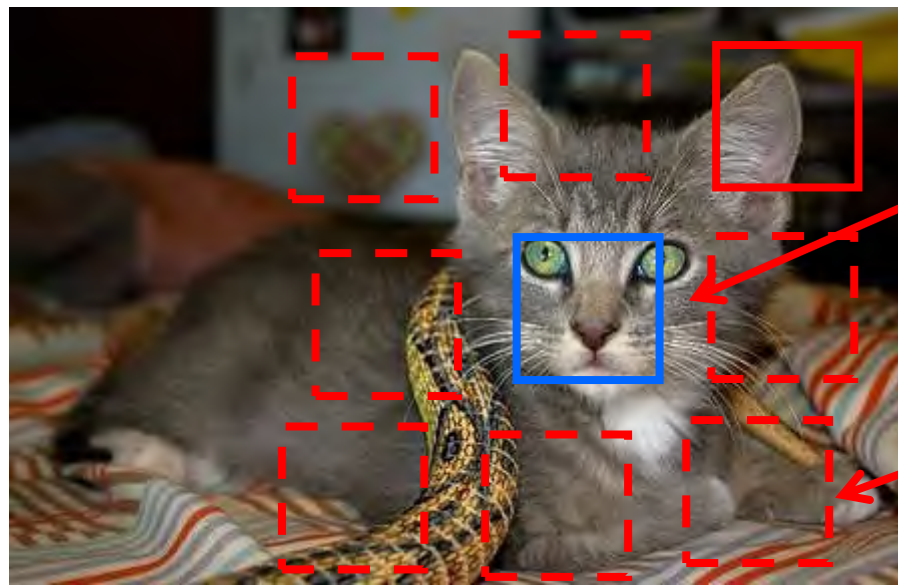
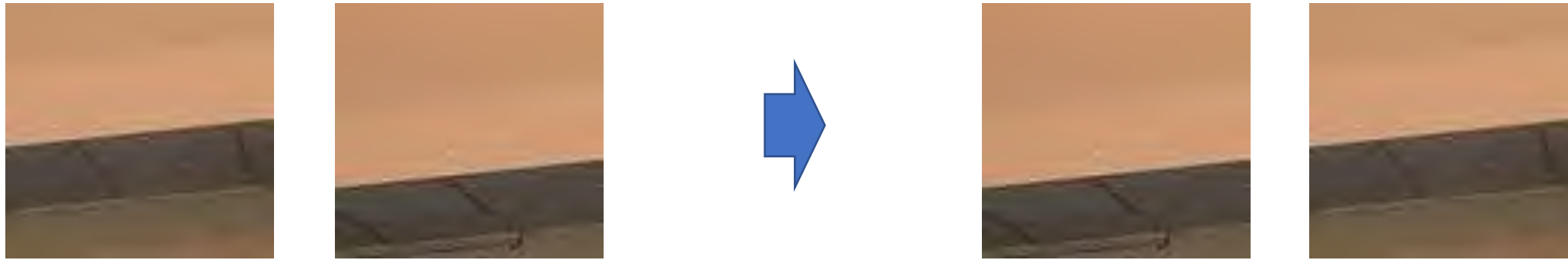
!

最近邻



在不同的实体之间检索相似的部件

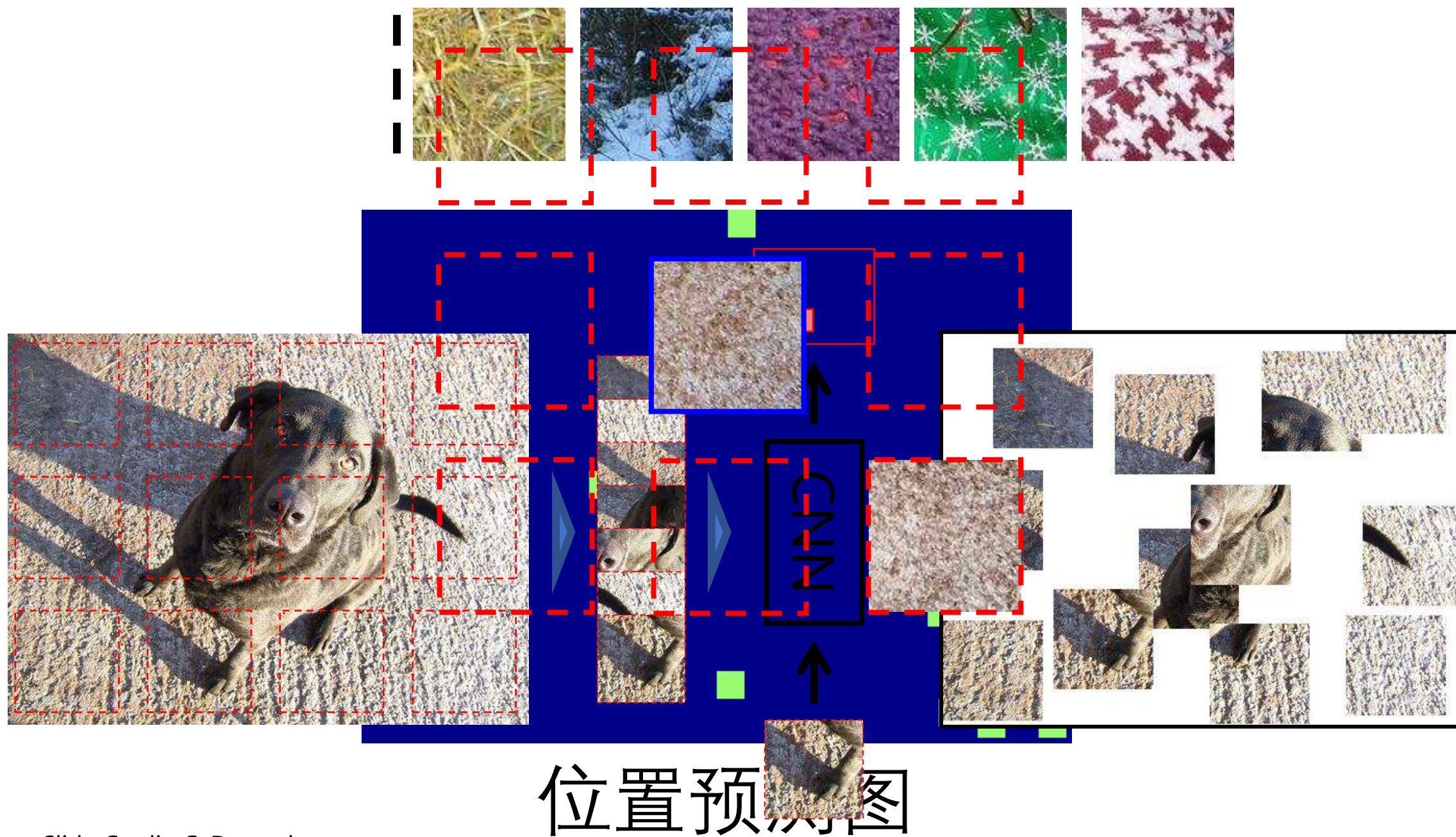
避免简单推断



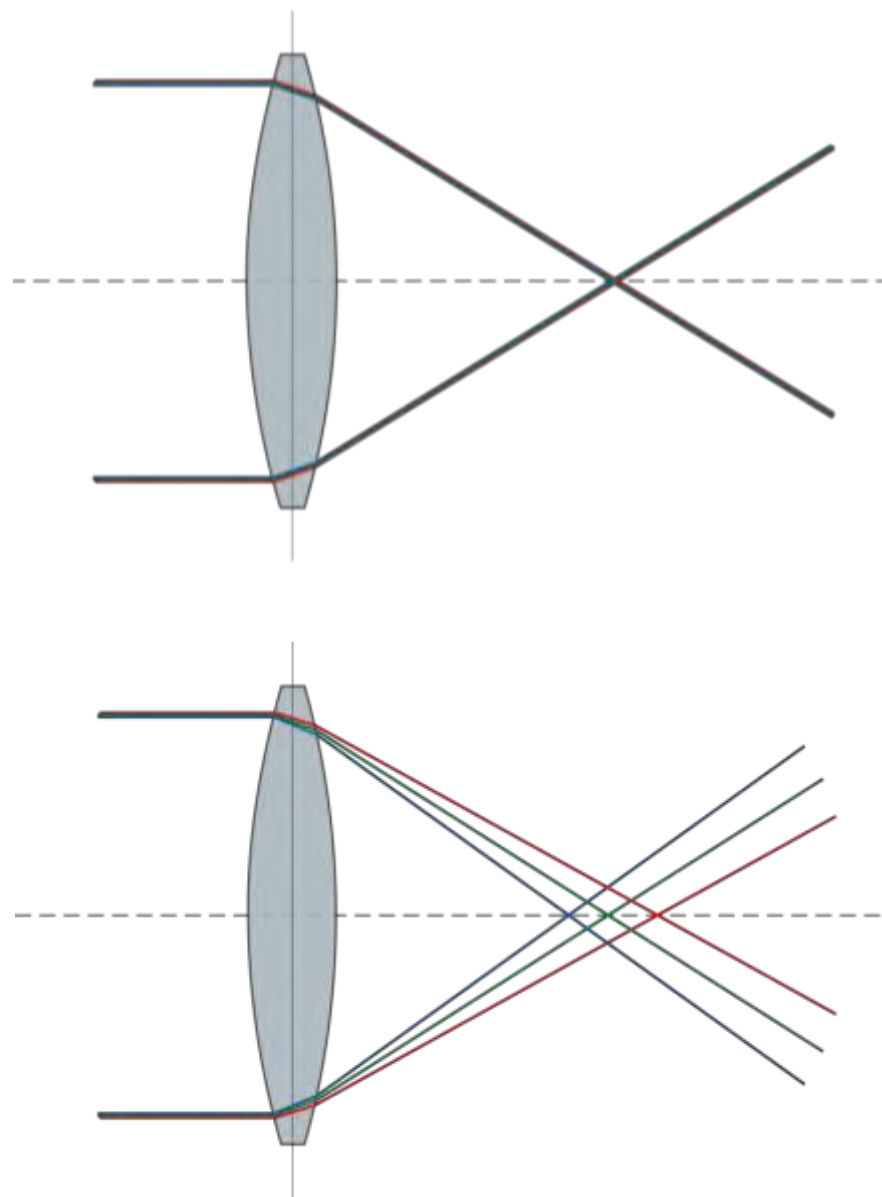
加一些间隔

区域大小进行一些微调

加入位置预测



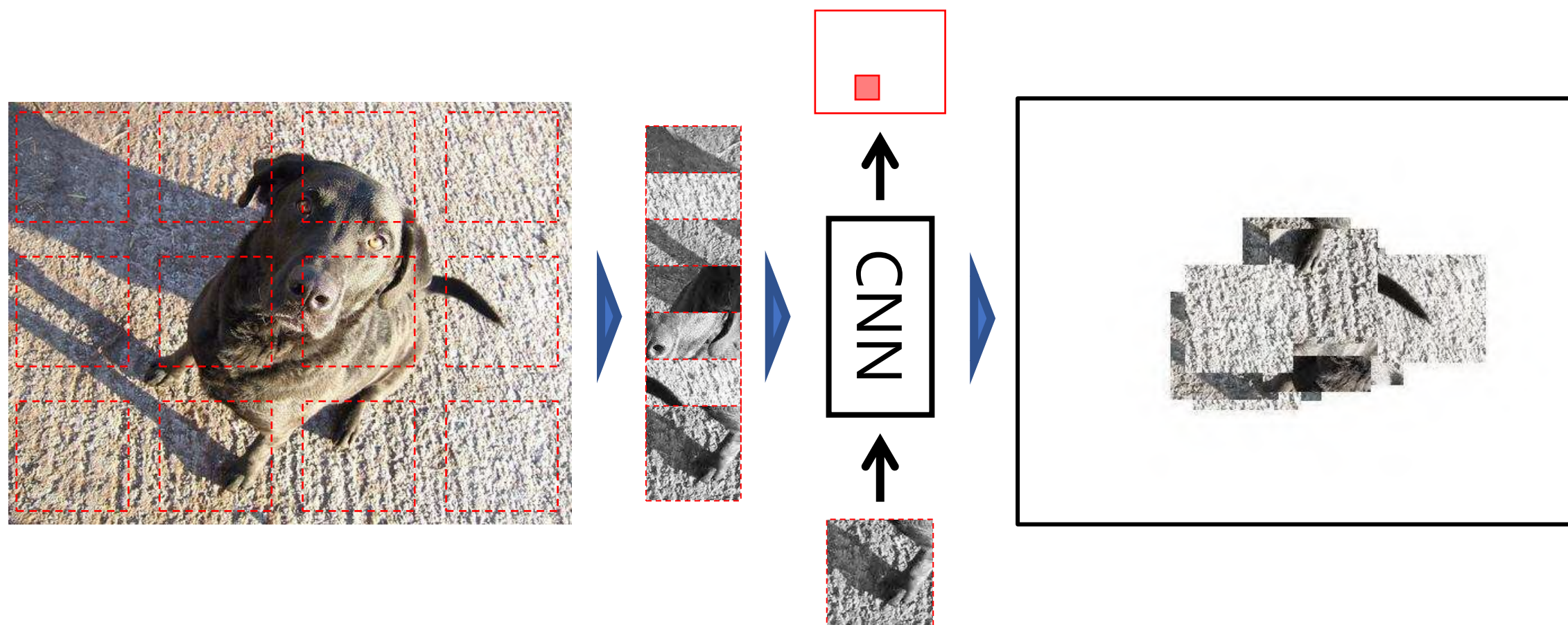
色差估计



Slide Credit: C. Doersch



去掉颜色以后。。。



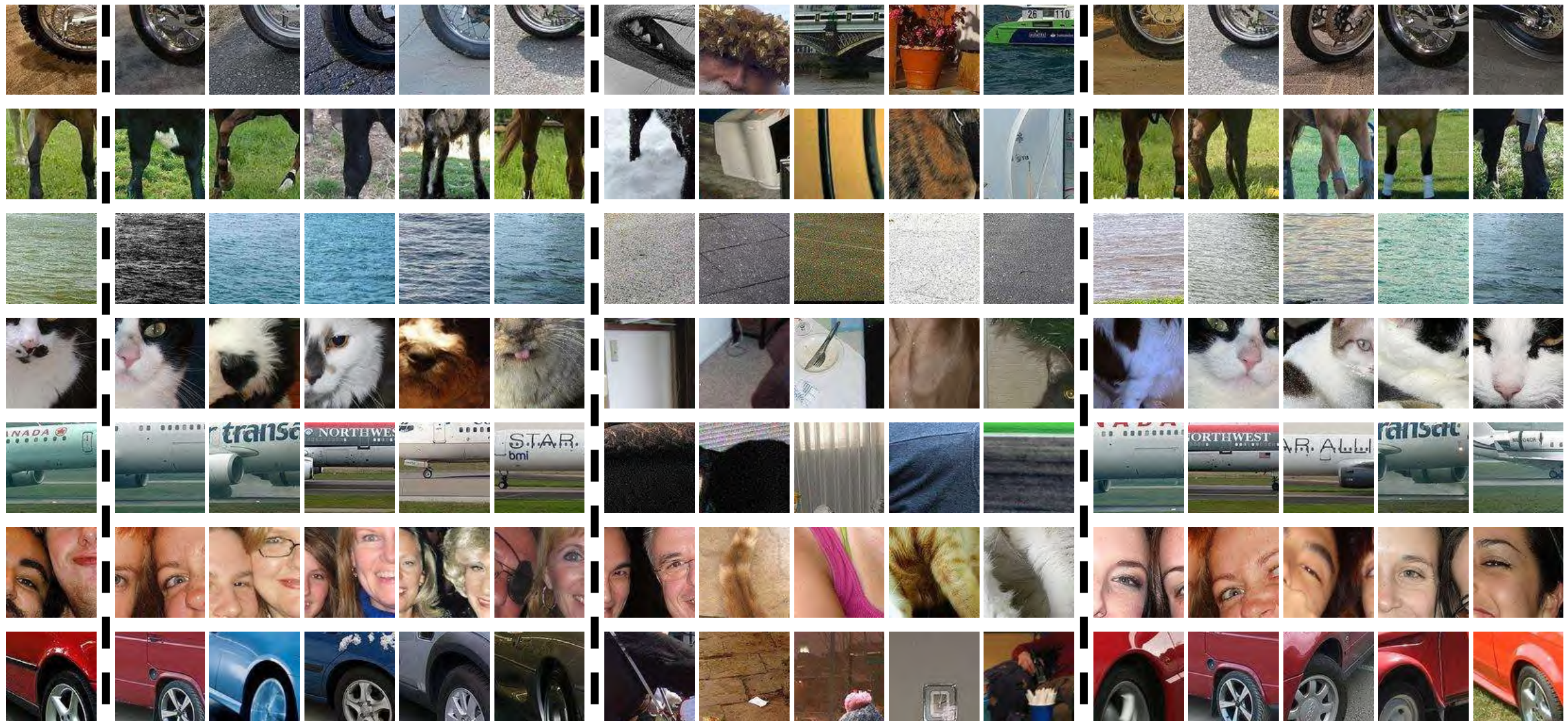
这样做学到了什么？

输入

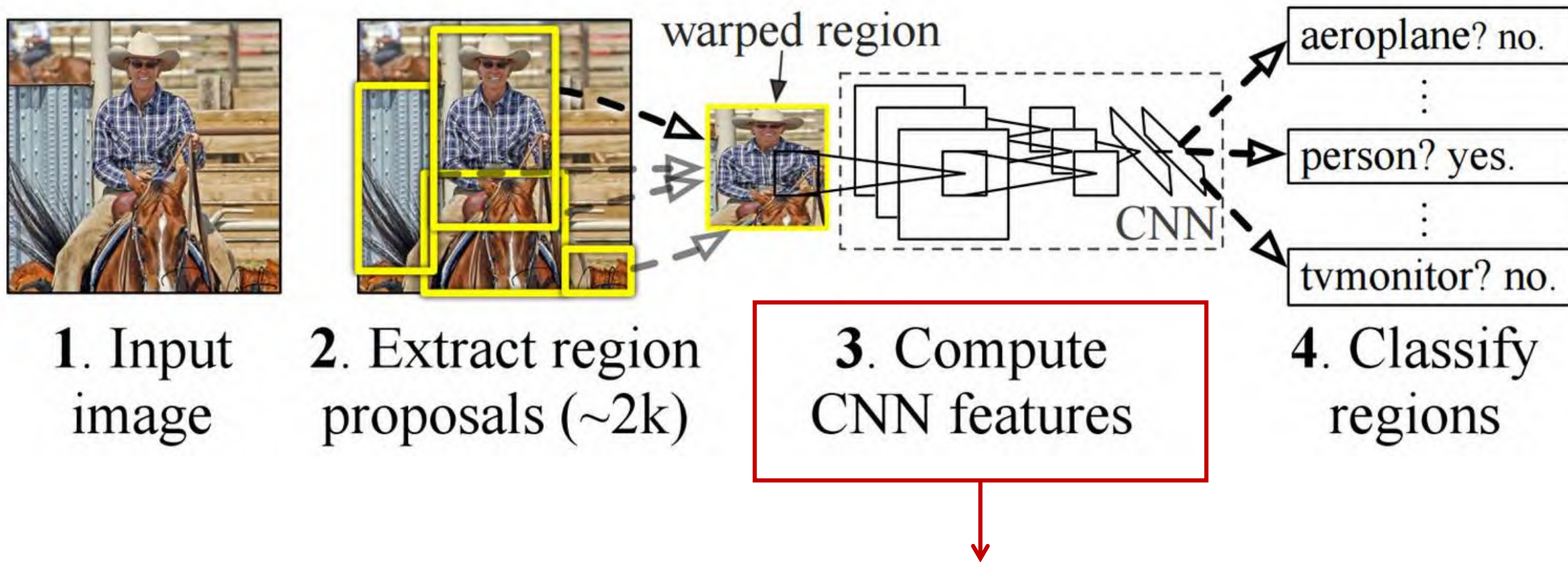
相对位置自监督

随机初始化

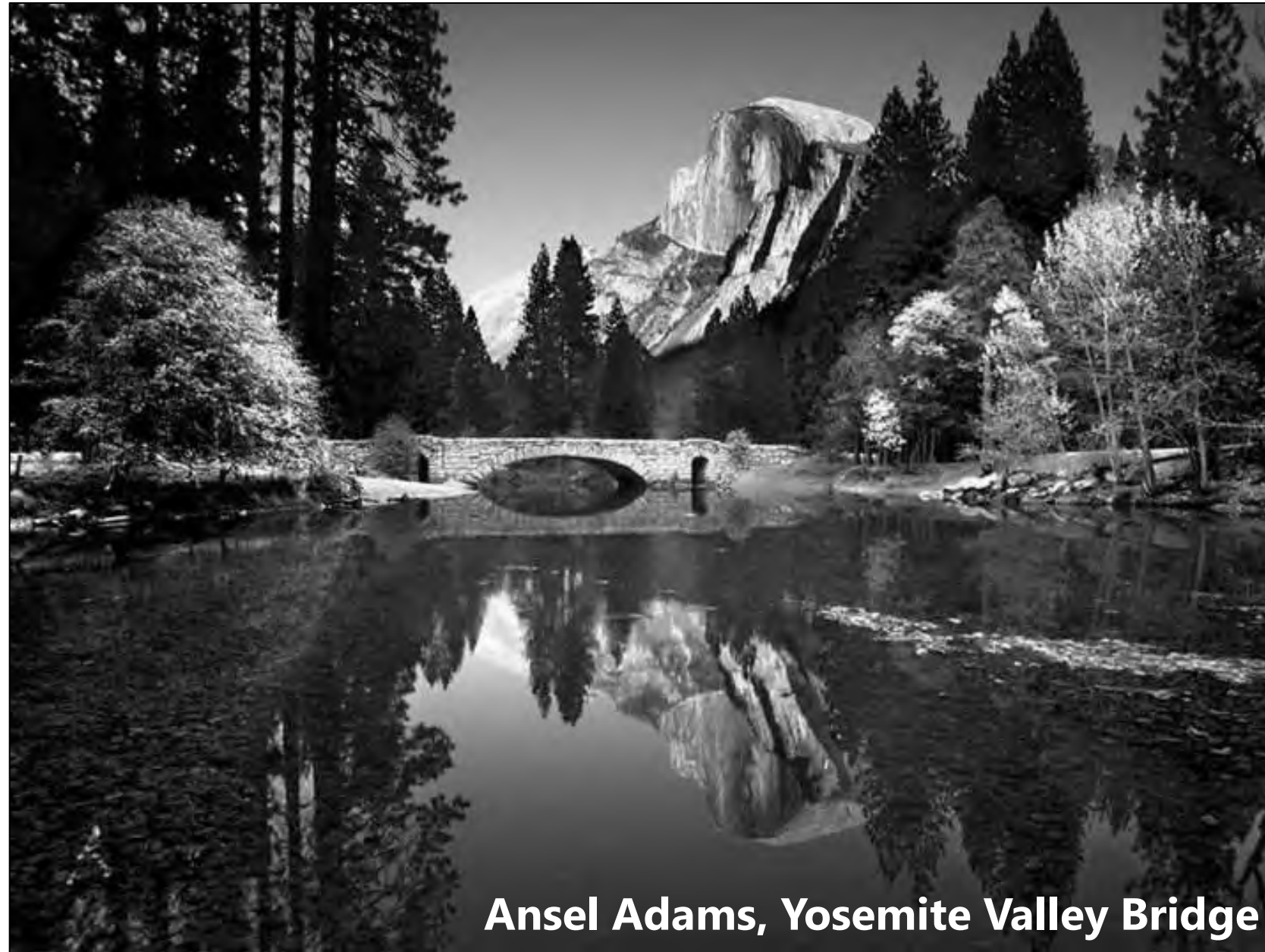
ImageNet 预训练



R-CNN 的 预训练 CNN



使用相对位置自监督学习的预训练模型

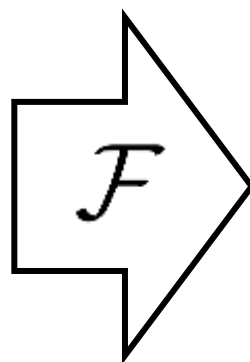


Ansel Adams, Yosemite Valley Bridge

Slide Credit: R. Zhang



Ansel Adams, Yosemite Valley Bridge – 上色

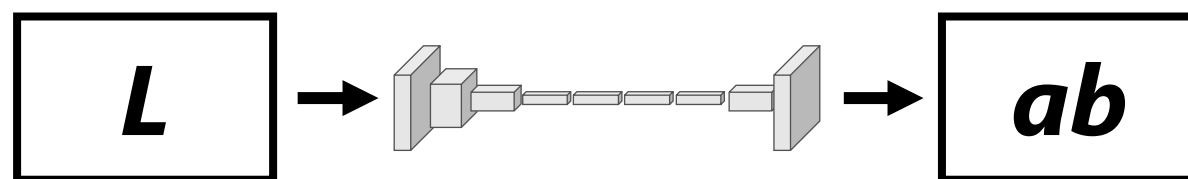


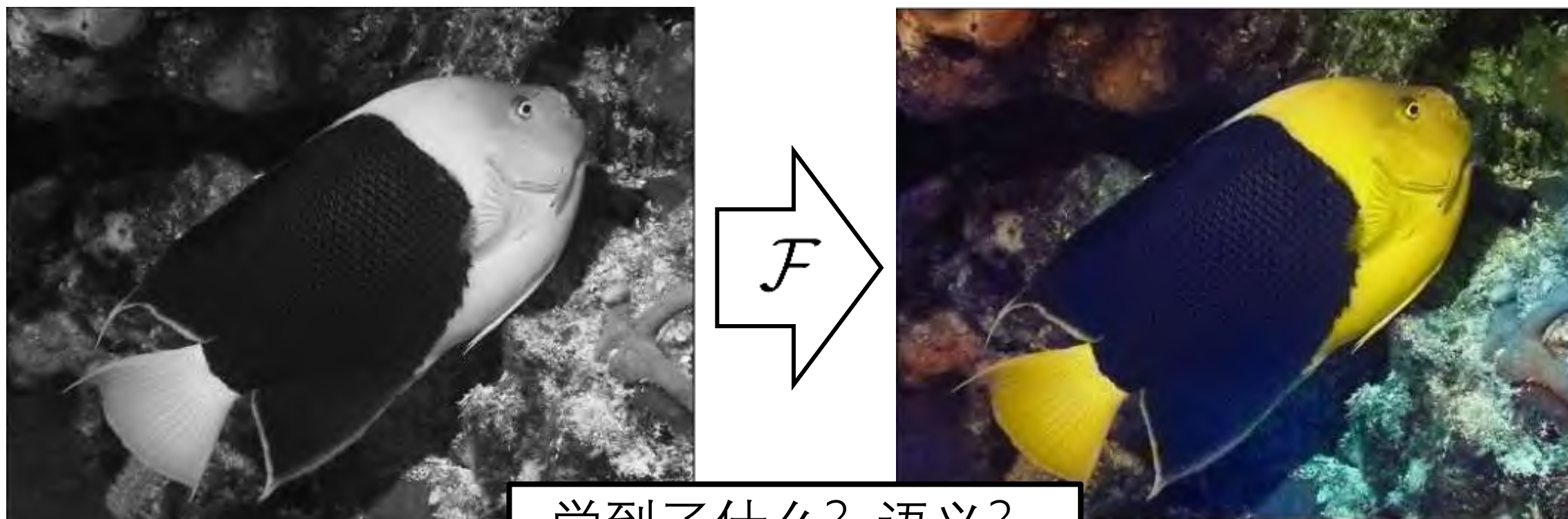
灰度图: L channel

$$\mathbf{X} \in \mathbb{R}^{H \times W \times 1}$$

色彩图: ab channels

$$\hat{\mathbf{Y}} \in \mathbb{R}^{H \times W \times 2}$$

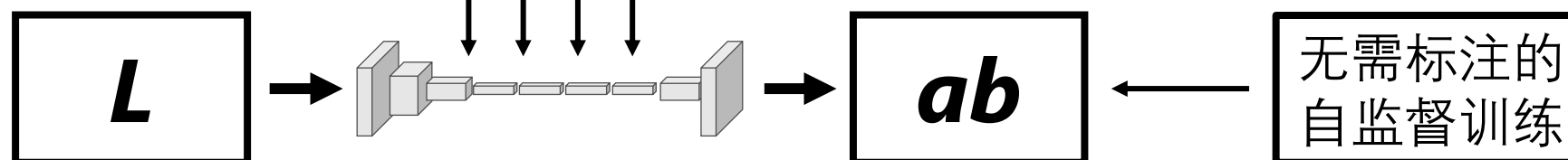




学到了什么? 语义?
抽象的信息?

灰度图: L channels
 $\mathbf{X} \in \mathbb{R}^{H \times W \times L}$

彩图: ab channels
 $(\mathbf{X}, \hat{\mathbf{Y}})$



输入



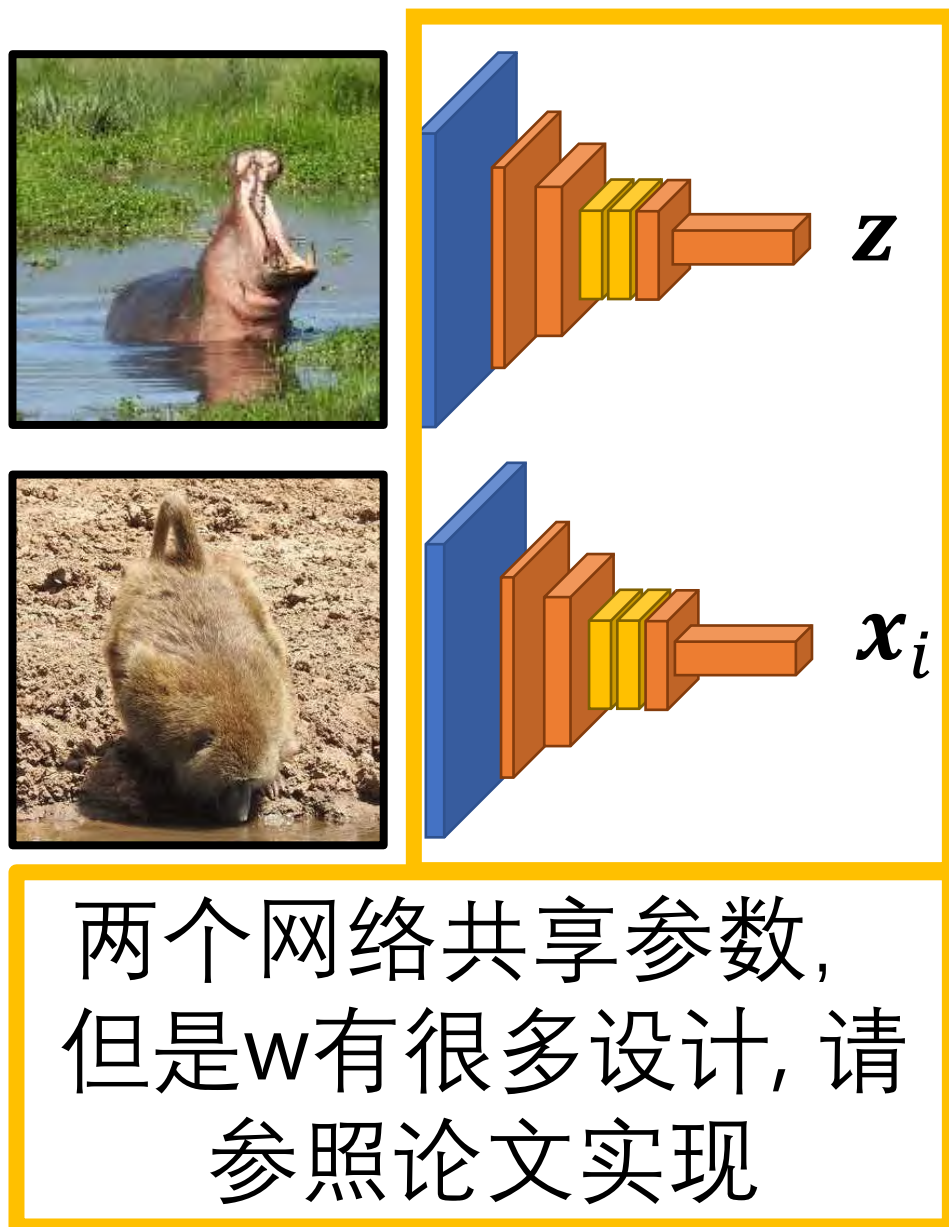
真实图



输出



Contrastive Learning 对比学习



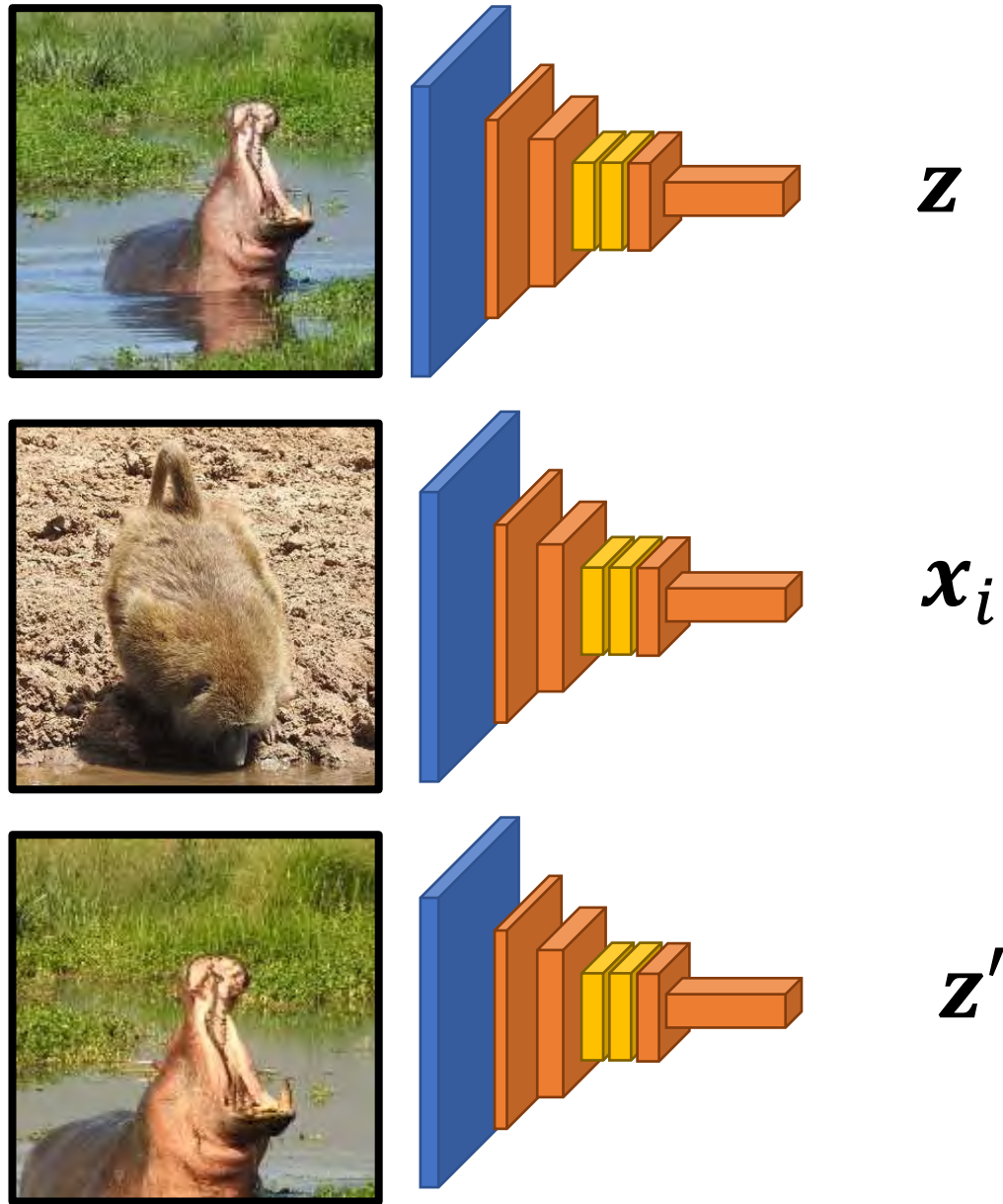
给定样本, 构建特征 z ; 对于其他一组样本图像, 最小化:

$$\frac{\exp(\mathbf{z}^T \mathbf{z})}{\exp(\mathbf{z}^T \mathbf{z} + \sum_i \mathbf{z}^T \mathbf{x}_i)}$$

会用 w 替代 z 来去计算相似度:

$$\frac{\exp(\mathbf{w}^T \mathbf{z})}{\exp(\mathbf{w}^T \mathbf{z} + \sum_i \mathbf{w}^T \mathbf{x}_i)}$$

Contrastive Learning 对比学习



最好的方式是找到一个 z 的增广样本 z' :

$$\frac{\exp(\mathbf{z}^T \mathbf{z}')}{\exp(\mathbf{z}^T \mathbf{z}' + \sum_i \mathbf{z}^T \mathbf{x}_i)}$$

目标: 增广样本 (与原图有相同语义) 的得分高于其他样本.

视频任务

视频分类：对比图像分类



Images: Recognize **objects**



Dog
Cat
Fish
Truck



Videos: Recognize **actions**



Swimming
Running
Jumping
Eating
Standing

Running video is in the public domain

问题: 视频太大了

- 视频每秒大约有30帧 (~30fps)



Input video:
 $T \times 3 \times H \times W$

Size of uncompressed video
(3 bytes per pixel):

SD (640 x 480): ~**1.5 GB per minute**

HD (1920 x 1080): ~**10 GB per minute**

问题: 视频太大了

- 视频每秒大约有30帧 (~30fps)



Input video:
 $T \times 3 \times H \times W$

Size of uncompressed video
(3 bytes per pixel):

SD (640 x 480): ~**1.5 GB per minute**

HD (1920 x 1080): ~**10 GB per minute**

解决方案 短视频 低分辨率:

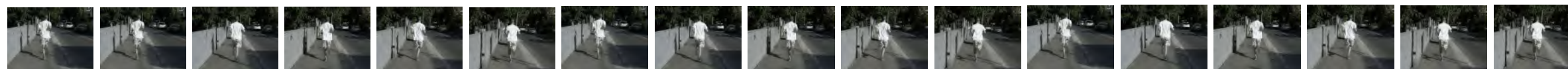
low fps and low spatial resolution

e.g. $T = 16, H=W=112$

(3.2 seconds at 5 fps, 588 KB)

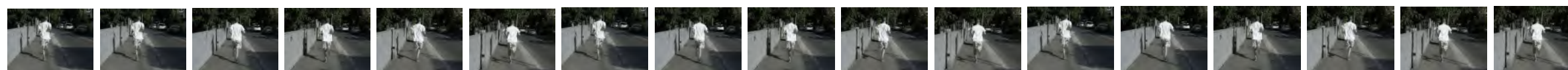
在视频上训练CNN

原视频：很长，很高的FPS



在视频上训练CNN

原视频：很长，很高的FPS

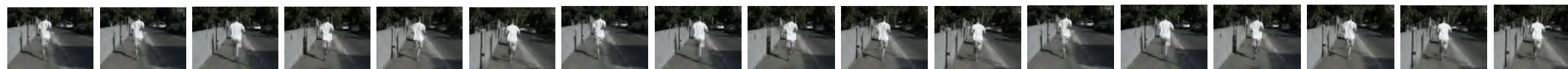


训练：短视频、低FPS



在视频上训练CNN

原视频：很长，很高的FPS



训练：短视频、低FPS

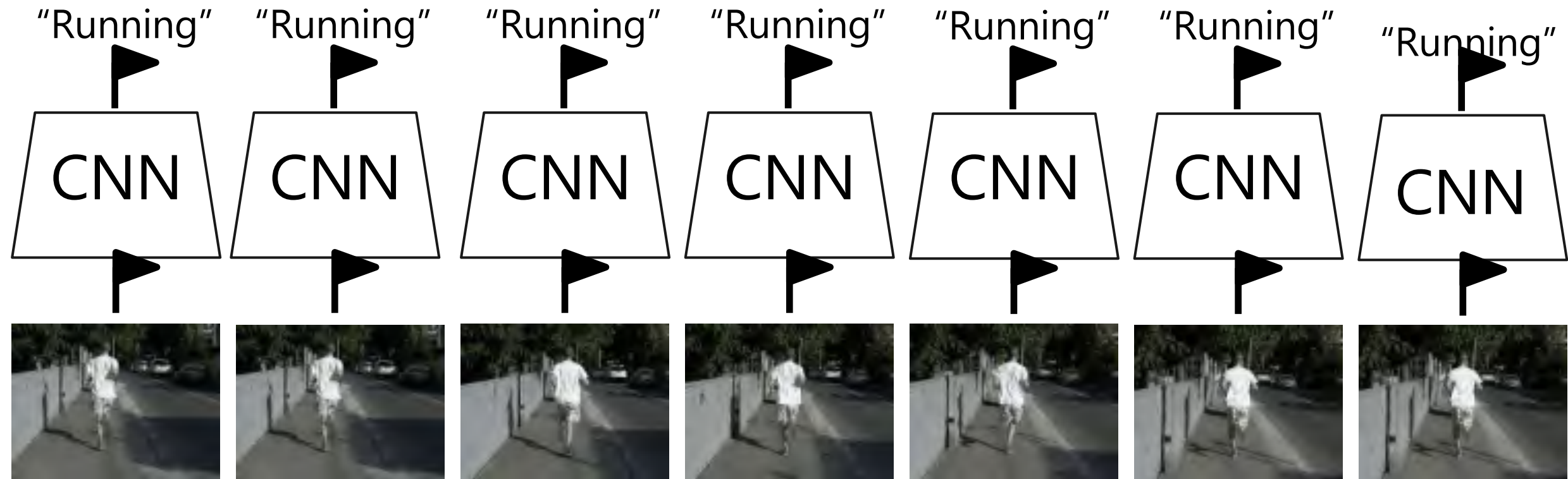


测试：在不同的短视频窗口预测，平均重叠部分



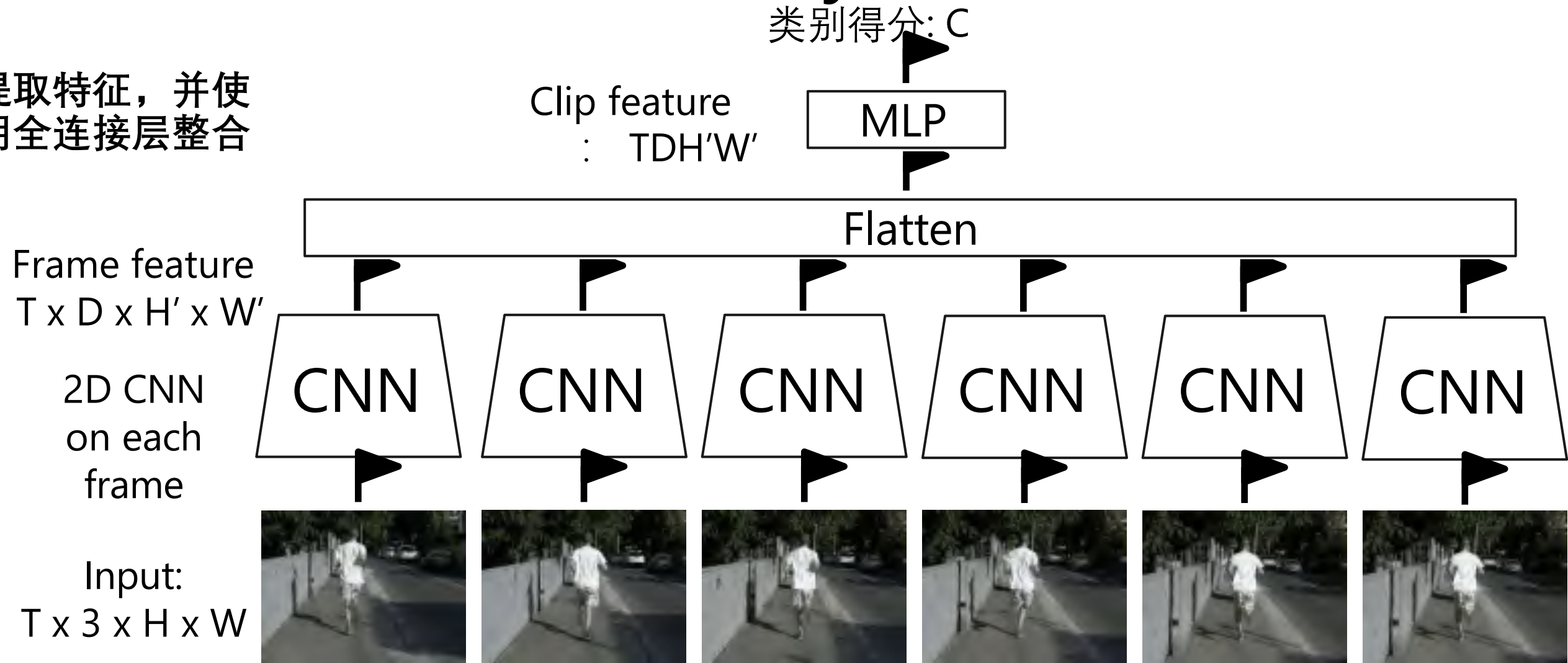
视频分类: 单帧CNN

- 想法: 训练2D CNN, 然后取均值



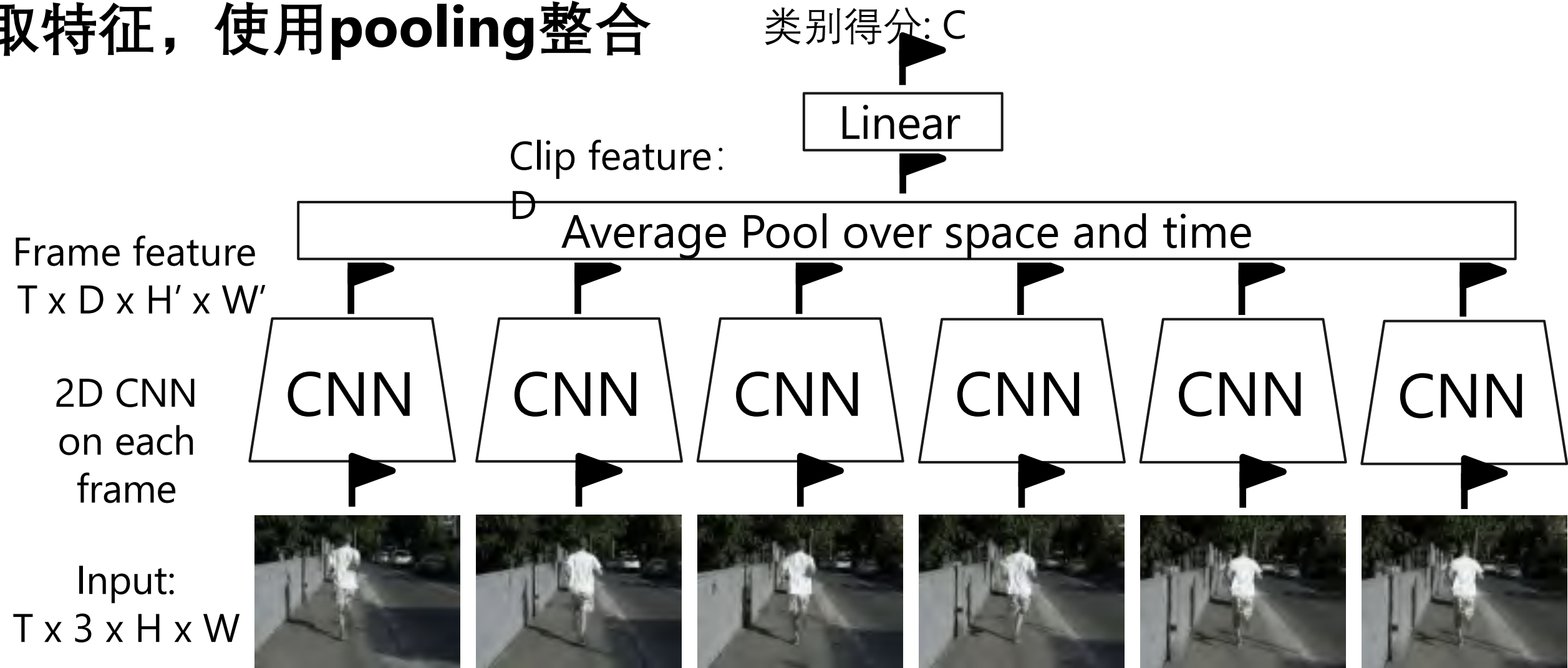
视频分类: Late Fusion (with FC layers)

- 提取特征, 并使用全连接层整合



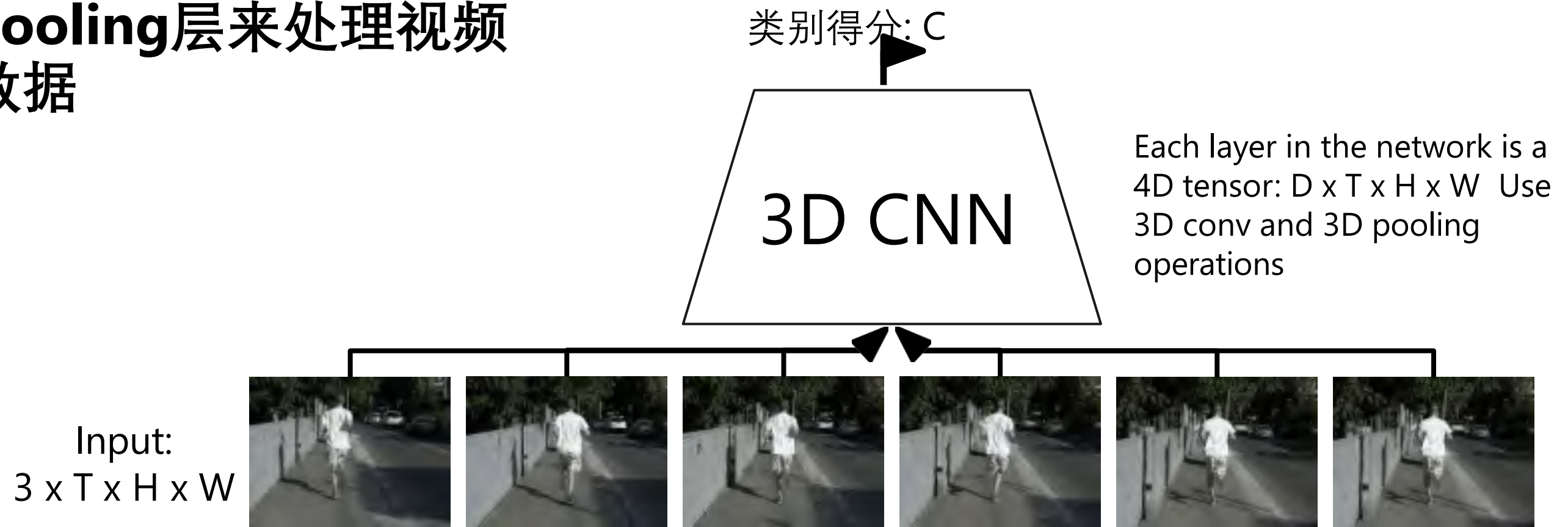
视频分类: Late Fusion (with pooling)

- 提取特征, 使用pooling整合

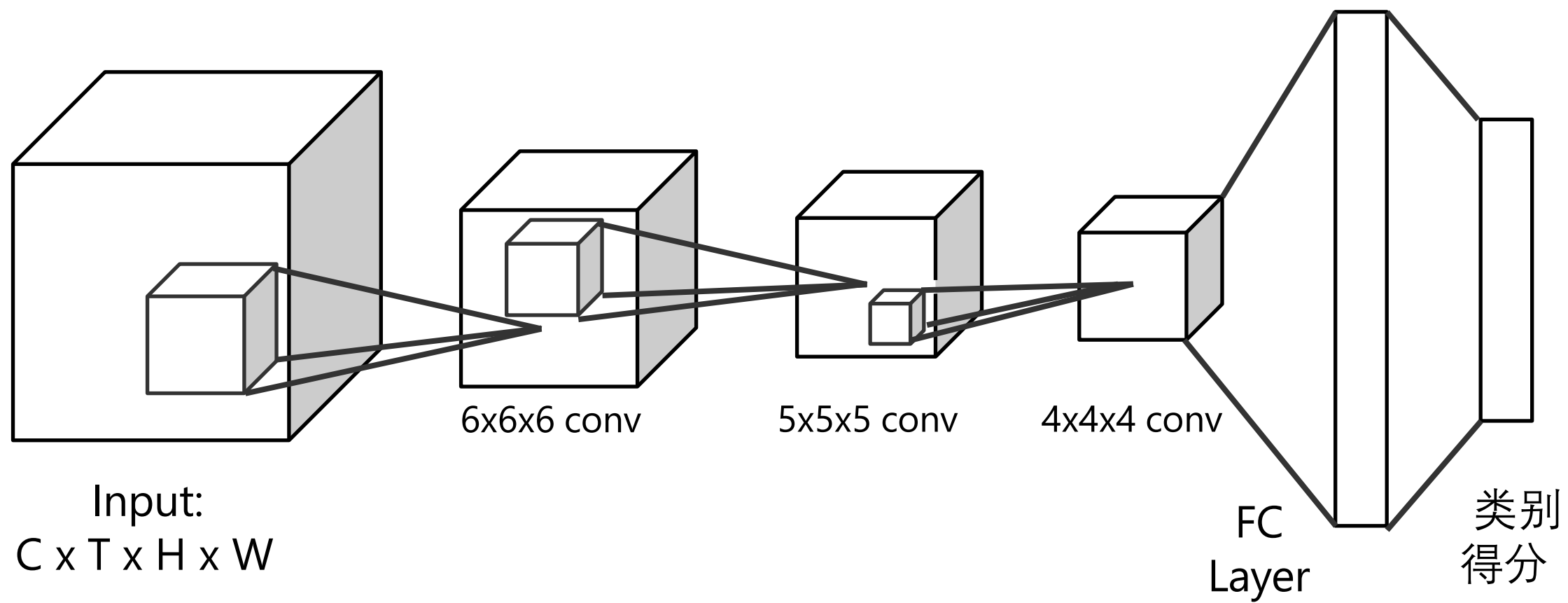


视频分类: 3D CNN

- 使用3维版本的卷积、**pooling**层来处理视频数据

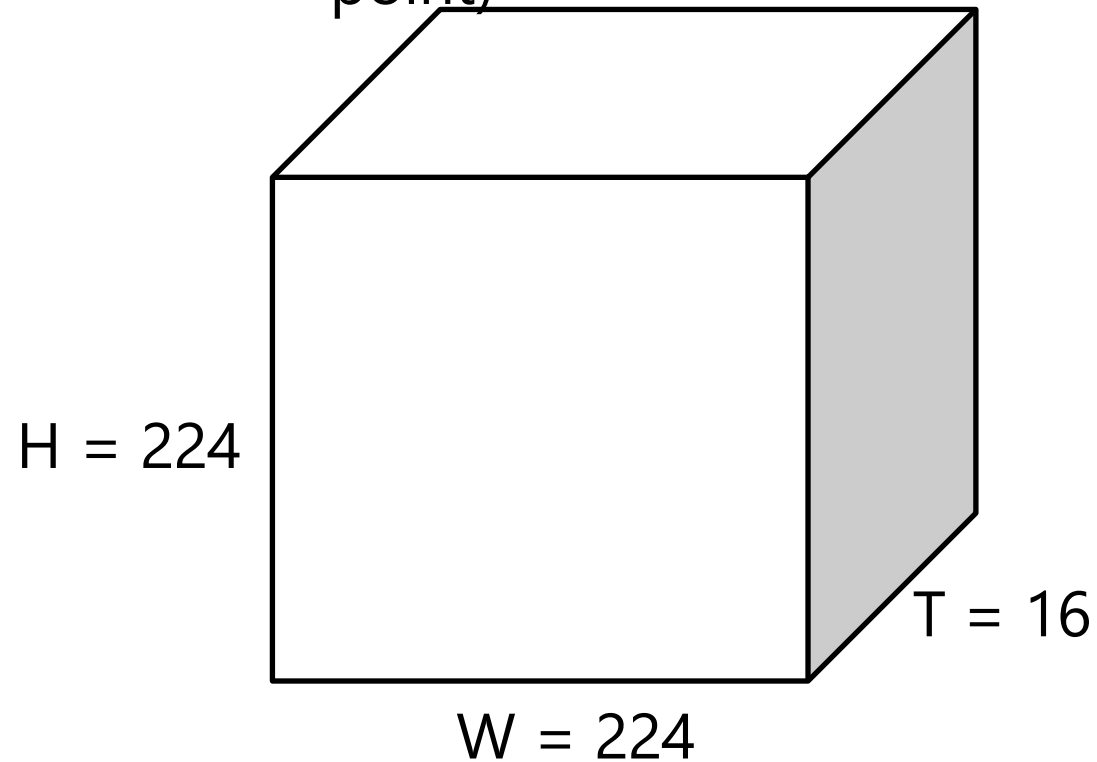


3D 卷积

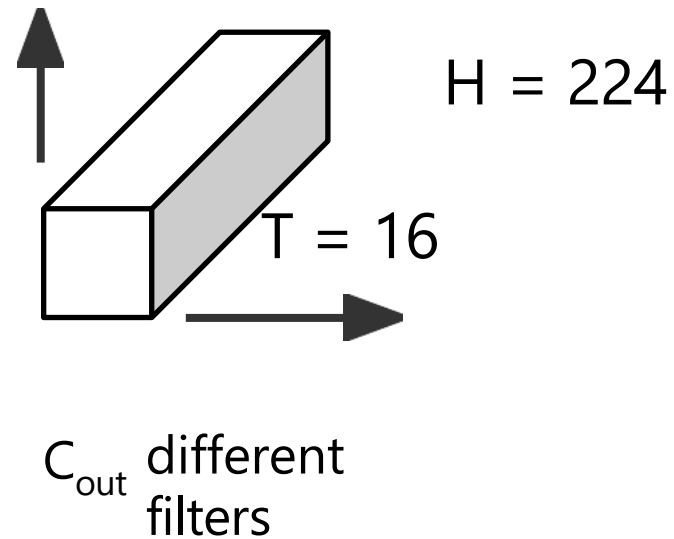


3D 卷积：所有帧全部考虑在内

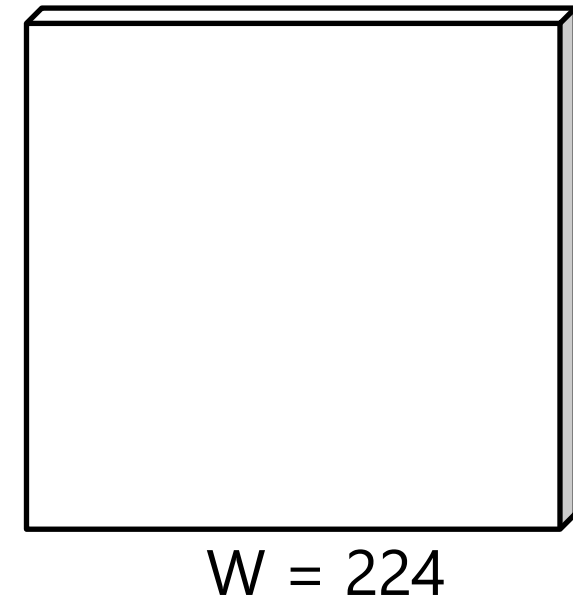
Input: $C_{in} \times T \times H \times W$
(3D grid with C_{in} -
dim feat at each
point)



Weight:
 $C_{out} \times C_{in} \times T \times 3 \times 3$
Slide over x and y

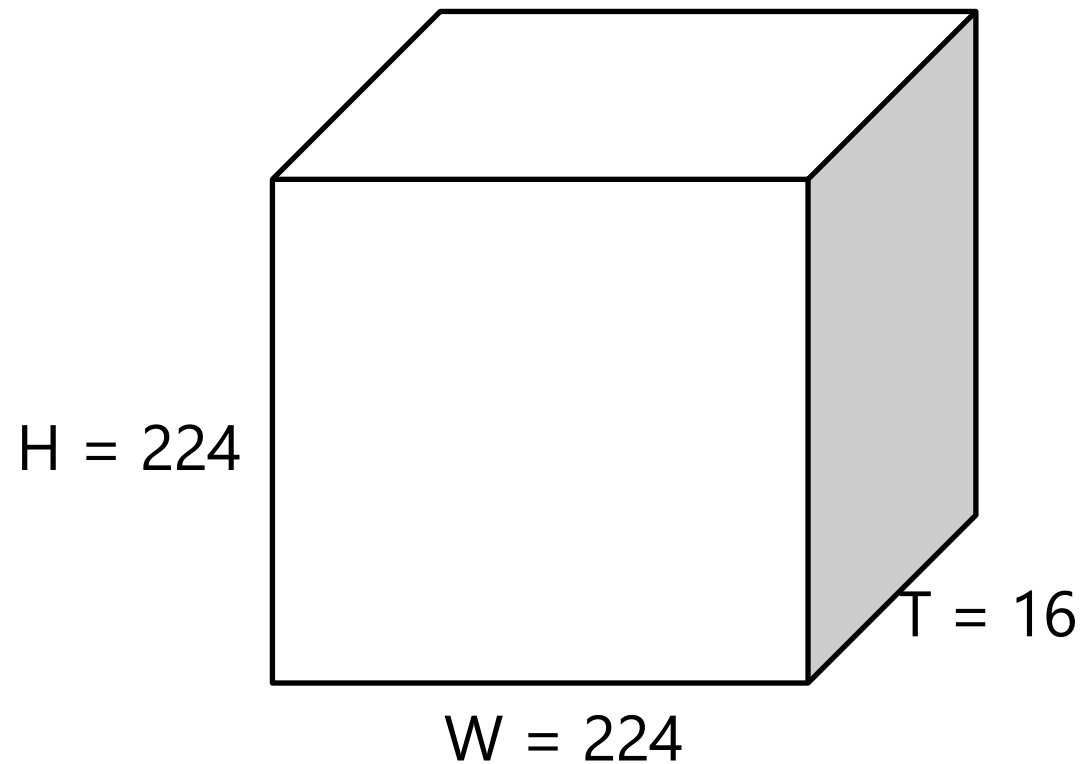


Output:
 $C_{out} \times H \times W$
2D grid with C_{out} -
dim feat at each point

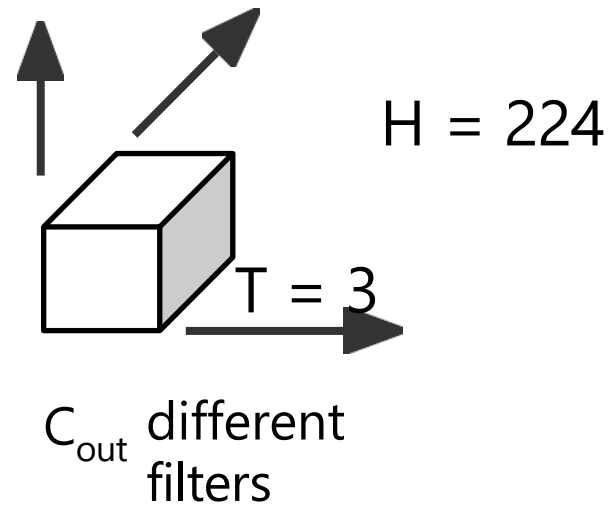


3D 卷积：时间上也做卷积

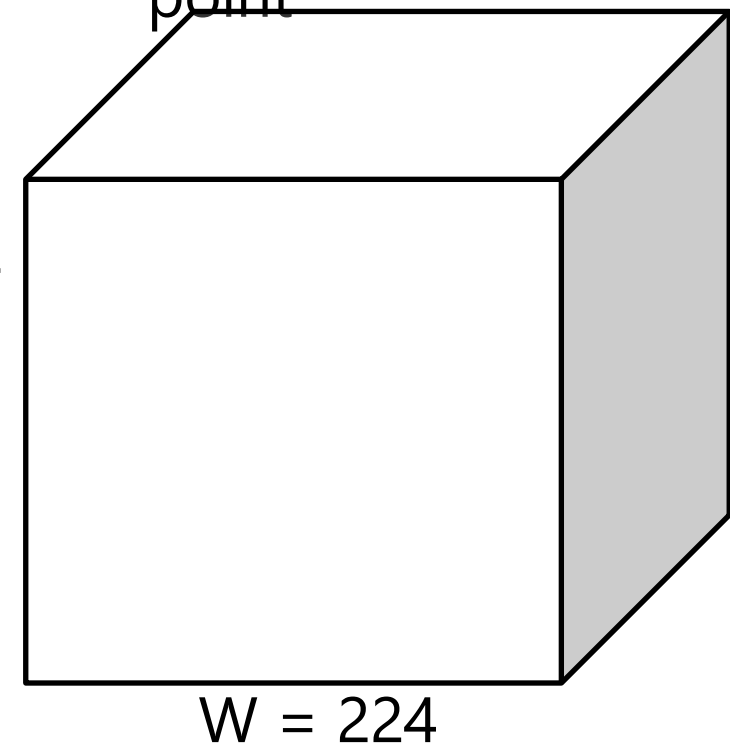
Input: $C_{in} \times T \times H \times W$
(3D grid with C_{in} -
dim feat at each
point)



Weight:
 $C_{out} \times C_{in} \times 3 \times 3 \times 3$
Slide over x and y



Output:
 $C_{out} \times T \times H \times W$
3D grid with C_{out} -
dim feat at each
point

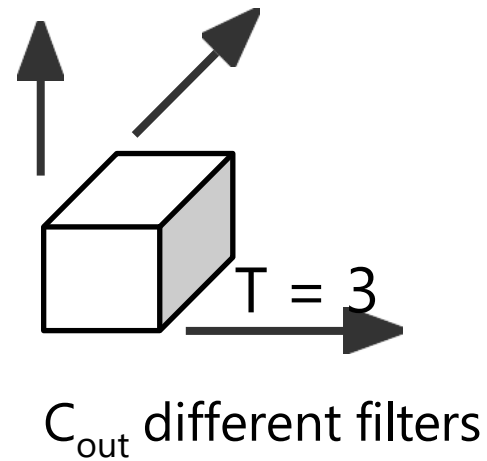
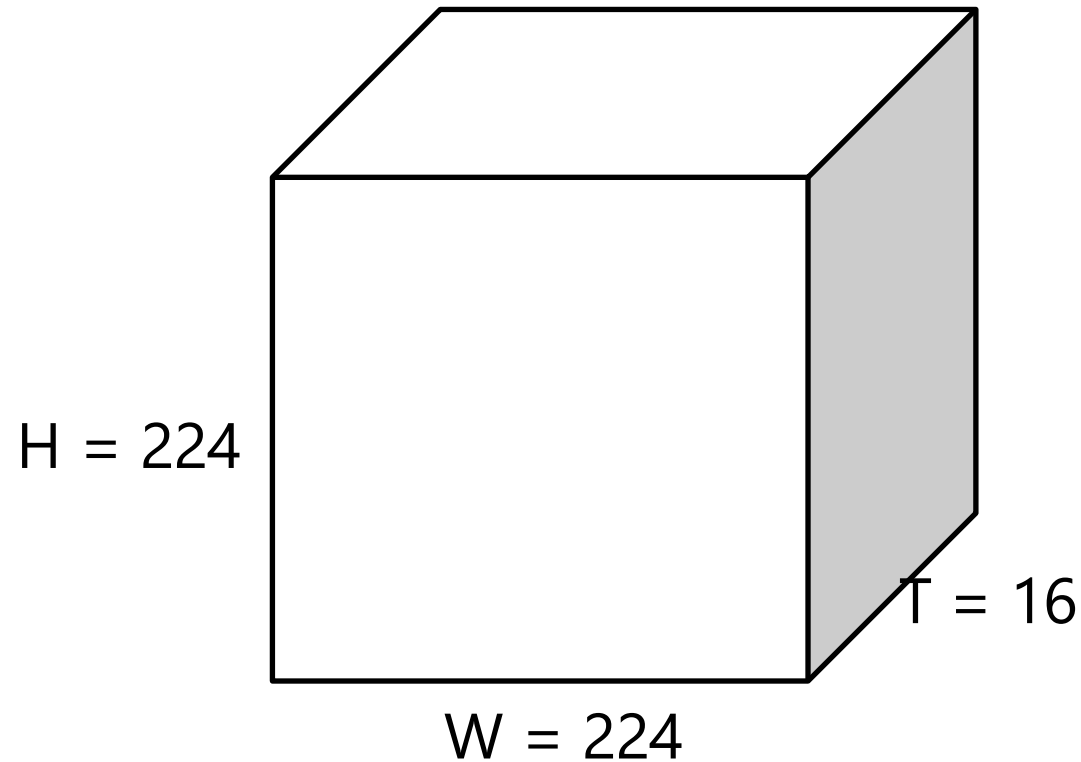


3D 卷积：同样也可以做可视化！

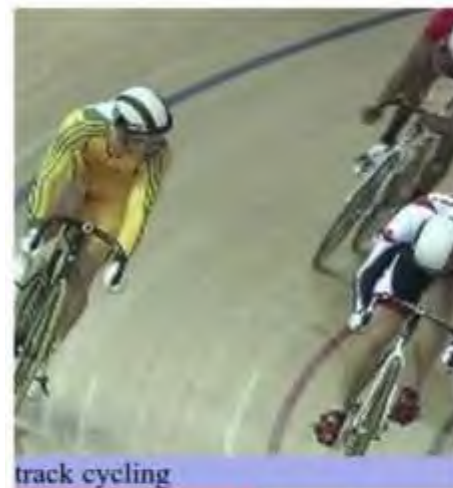
Input: $C_{in} \times T \times H \times W$
(3D grid with C_{in} -dim feat at each point)

Weight:
 $C_{out} \times C_{in} \times 3 \times 3 \times 3$
Slide over x and y

First-layer filters have shape
3 (RGB) x 4 (frames) x 5 x 5
(space)
Can visualize as video clips!



Sports-1M



track cycling
cycling
track cycling
road bicycle racing
marathon
ultramarathon



ultramarathon
ultramarathon
half marathon
running
marathon
inline speed skating



heptathlon
heptathlon
decathlon
hurdles
pentathlon
sprint (running)



bikejoring
mushing
bikejoring
harness racing
skijoring
carting

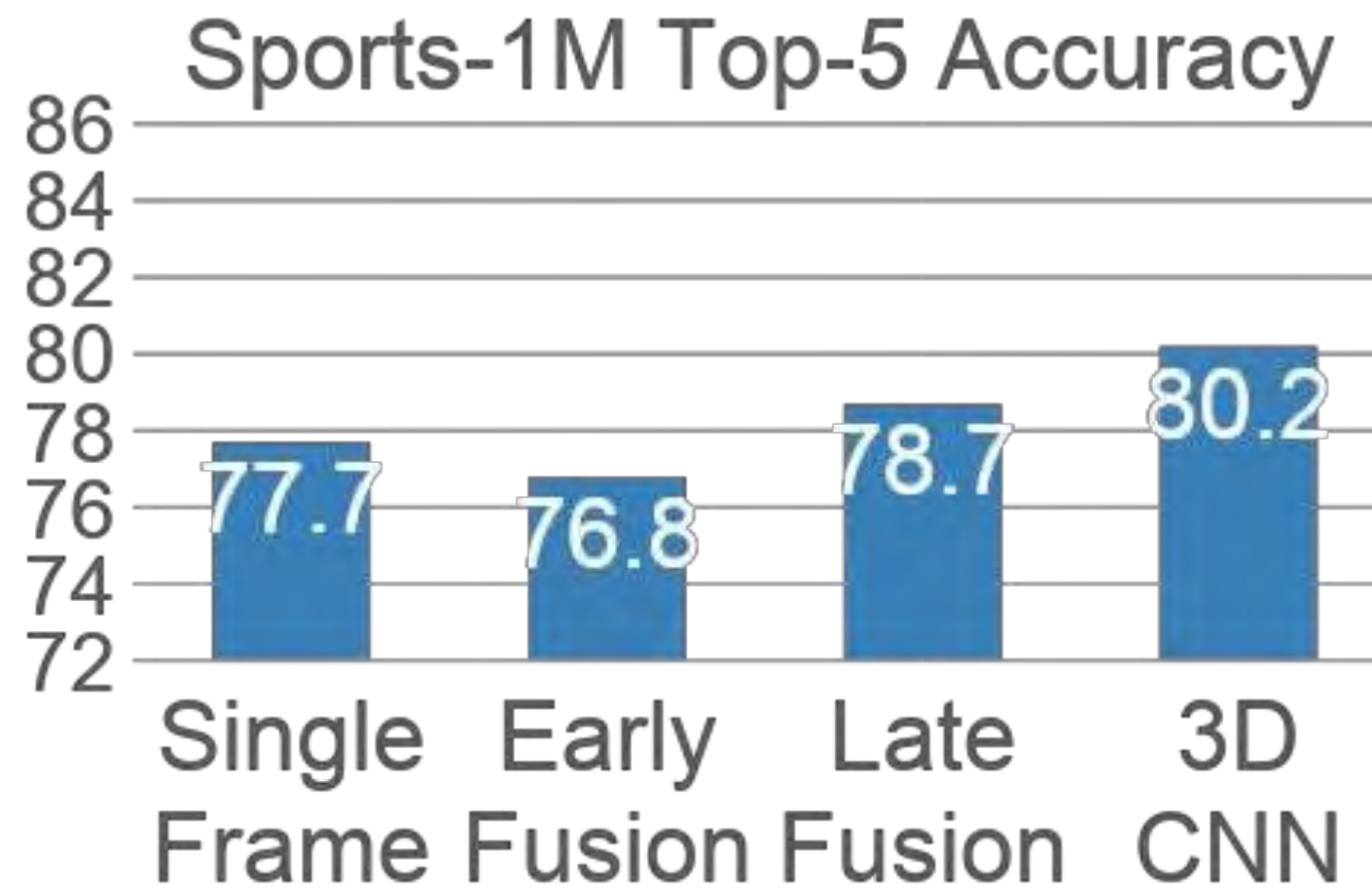


longboarding
longboarding
aggressive inline skating
freestyle scootering
freeboard (skateboard)
sandboarding

1 million YouTube videos annotated with labels for 487 different types of sports

Ground Truth
Correct prediction
Incorrect prediction

3D CNN



光流

Image at frame t



Image at frame t+1

光流

Image at frame t

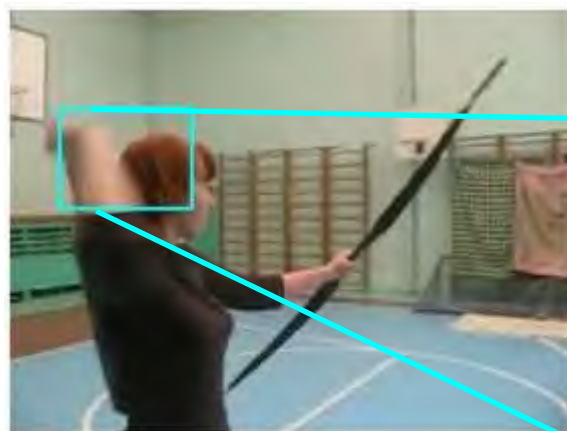
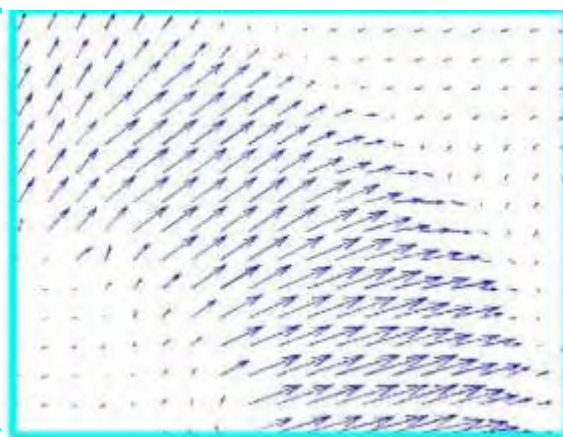


Image at frame t+1



下一帧该像素会出现在哪里： $F(x, y) = (dx, dy)$

$$I_{t+1}(x+dx, y+dy) = I_t(x, y)$$

光流

Image at frame t

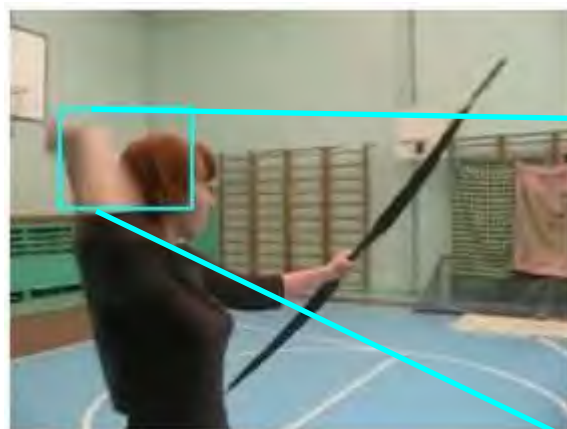
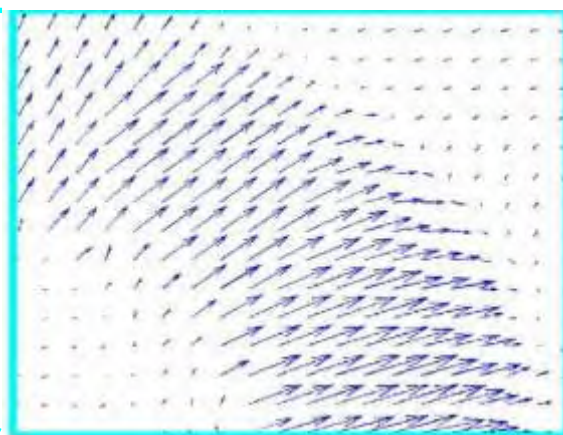
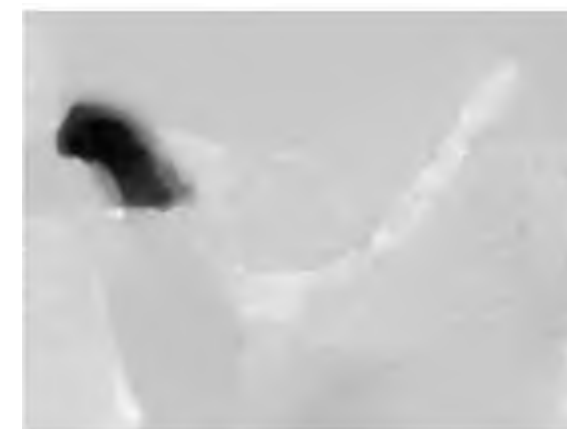
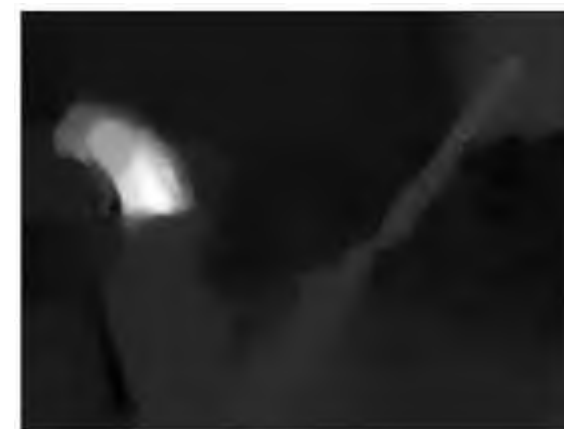


Image at frame t+1



下一帧该像素会出现在哪里： $F(x, y) = (dx, dy)$
 $I_{t+1}(x+dx, y+dy) = I_t(x, y)$

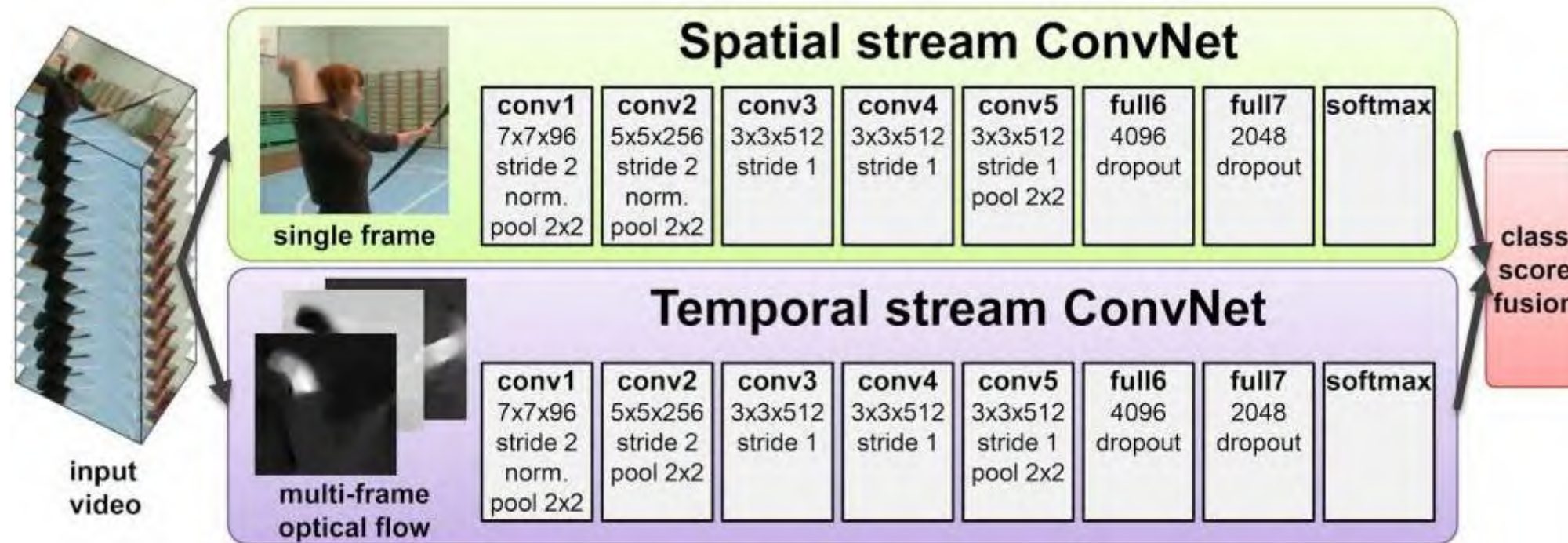
Horizontal flow dx



Vertical Flow dy

Two-Stream 网络：分开处理运动和外表

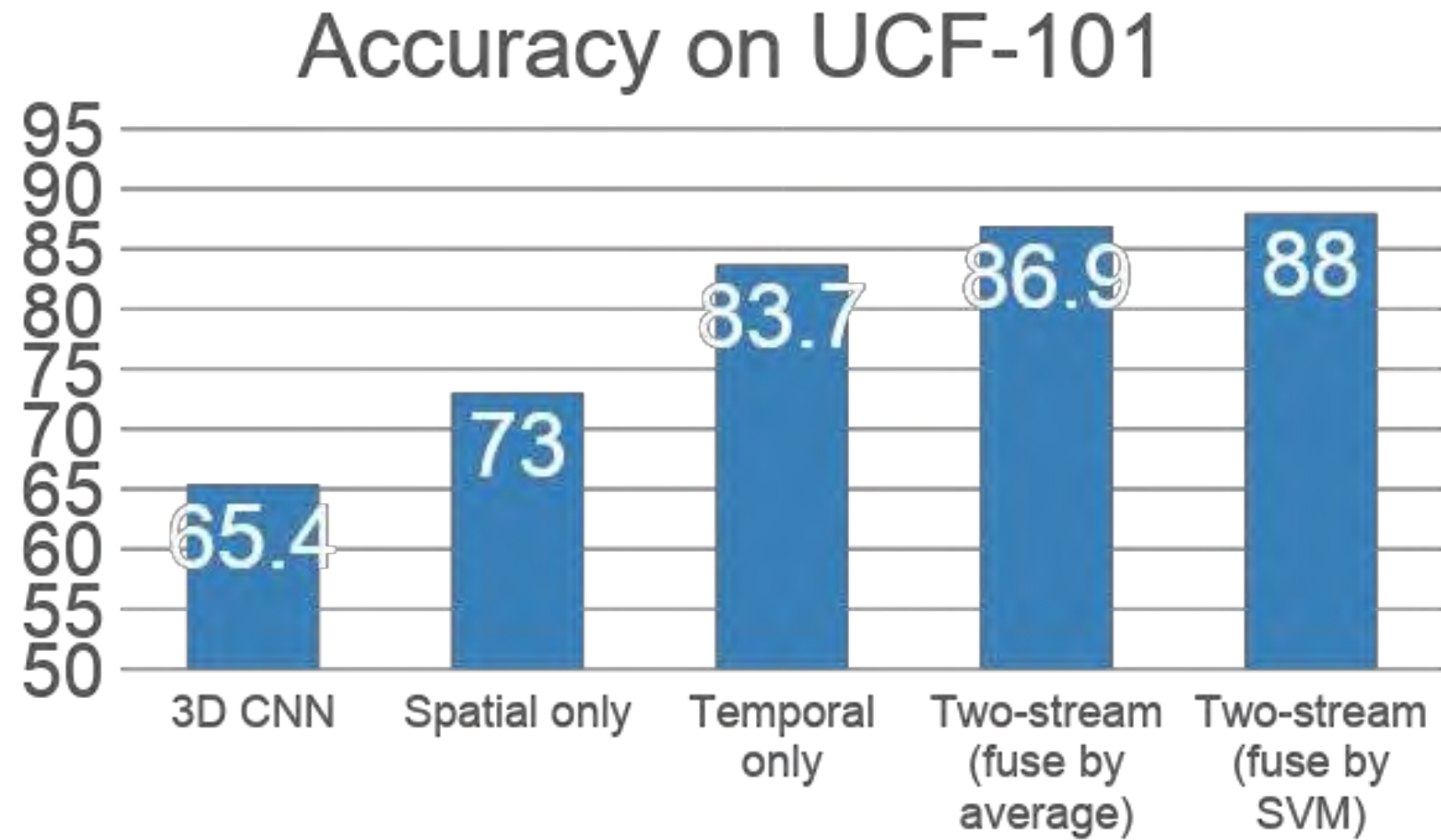
Input: Single Image
3 x H x W



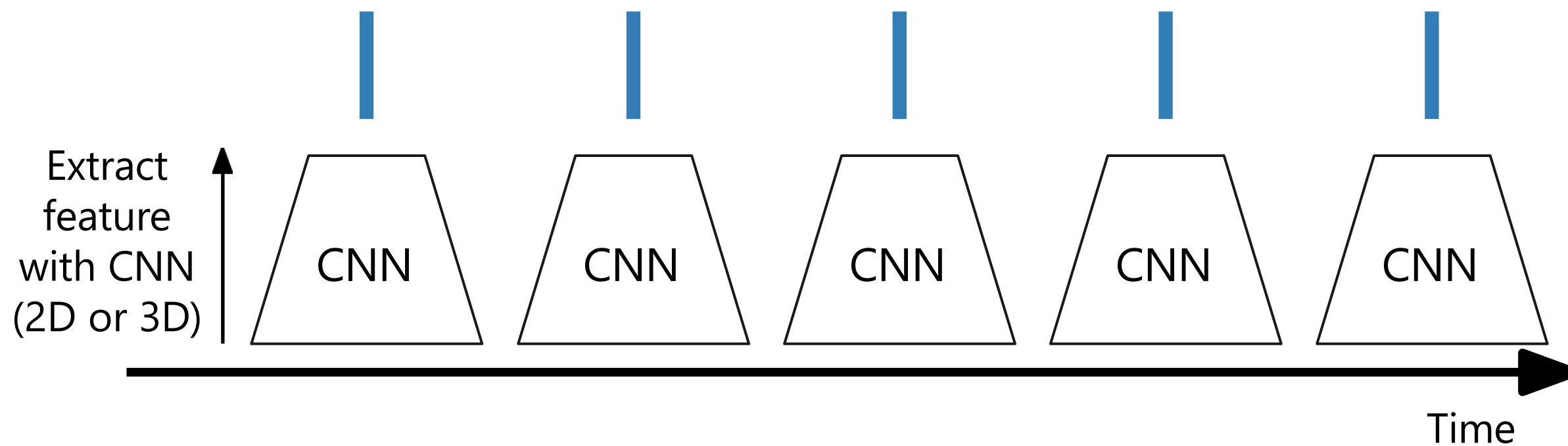
Input: Stack of optical flow:
 $[2*(T-1)] \times H \times W$

Early fusion: First 2D conv
processes all flow images

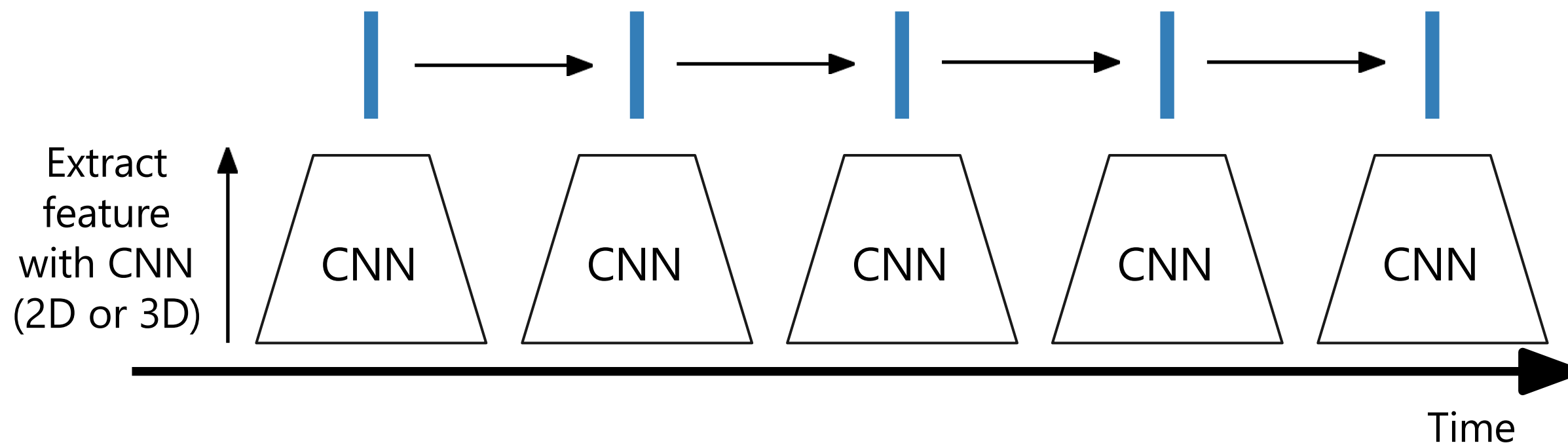
Two-Stream 网络：分开处理运动和外表



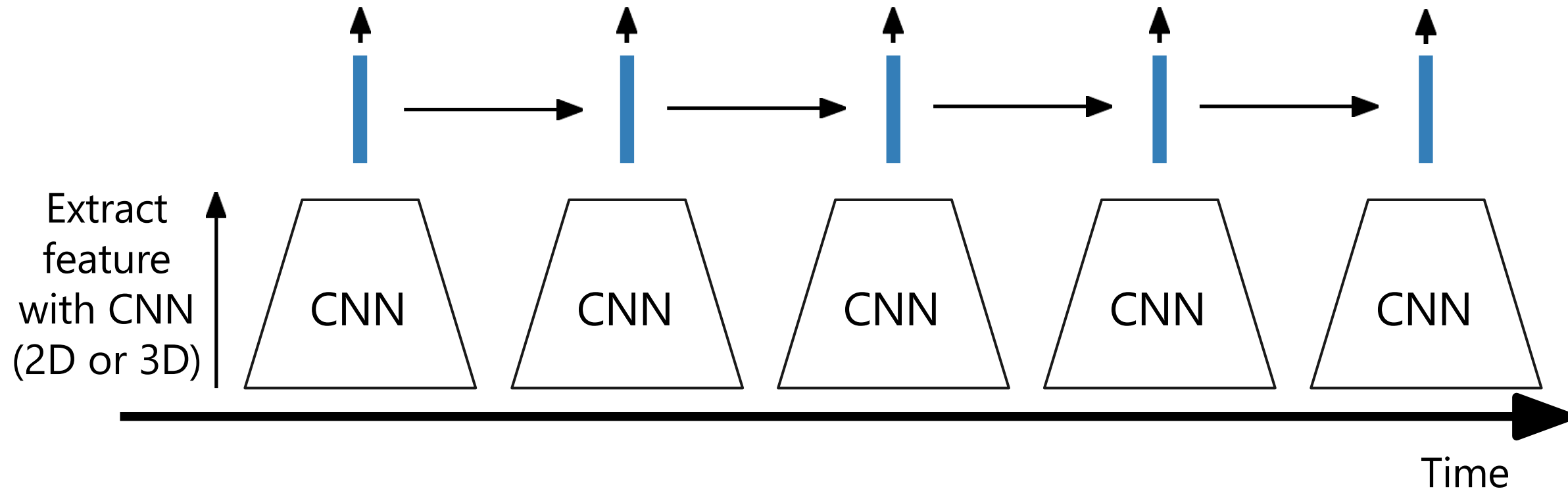
提取长远的时序特征



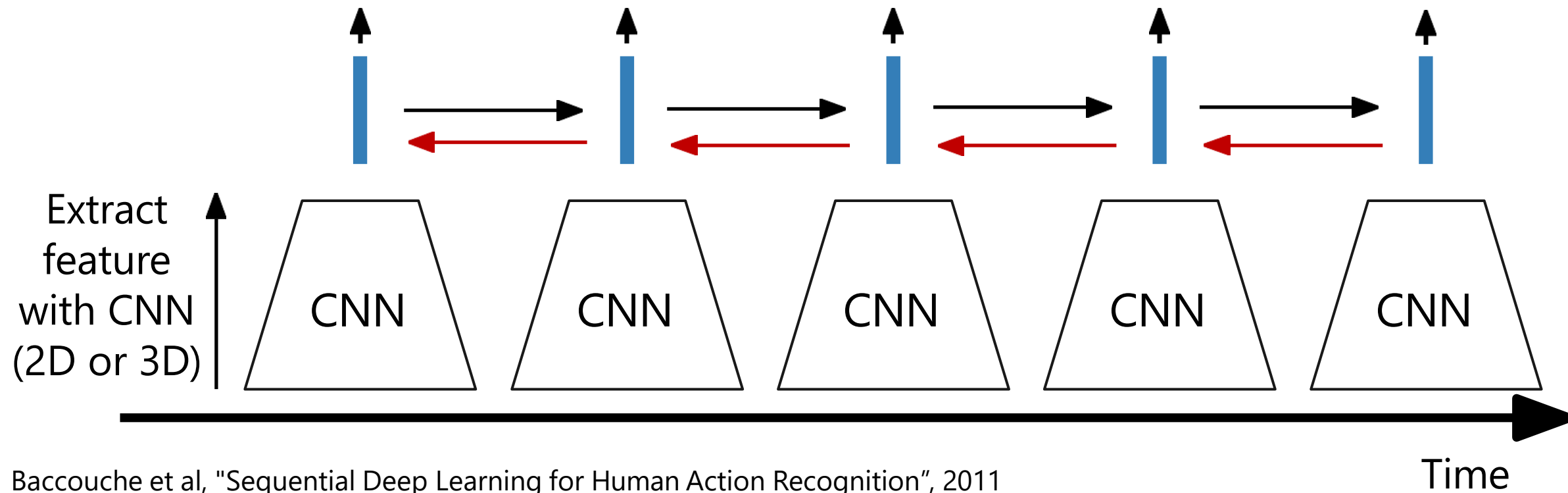
提取长远的时序特征



提取长远的时序特征



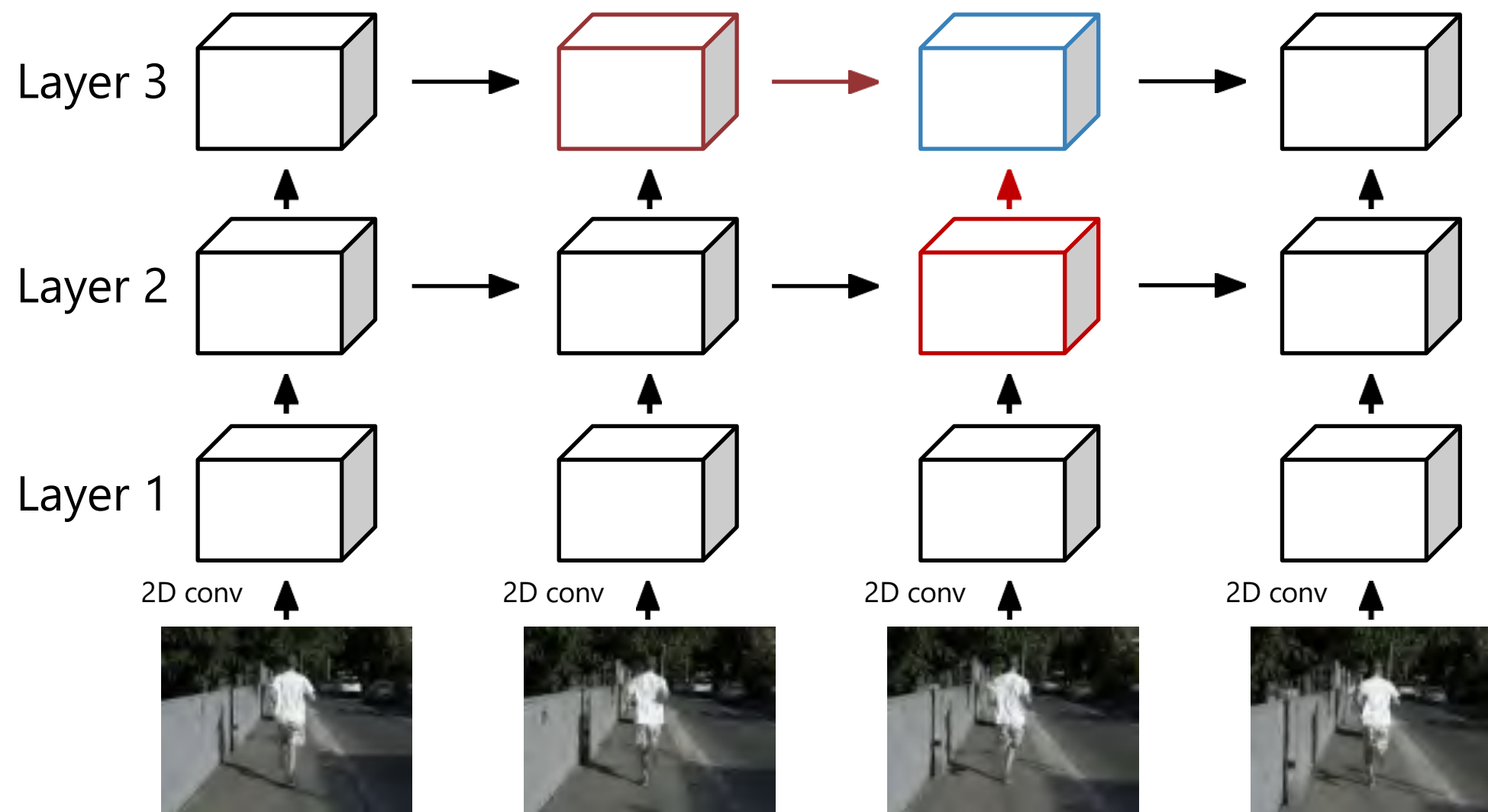
提取长远的时序特征



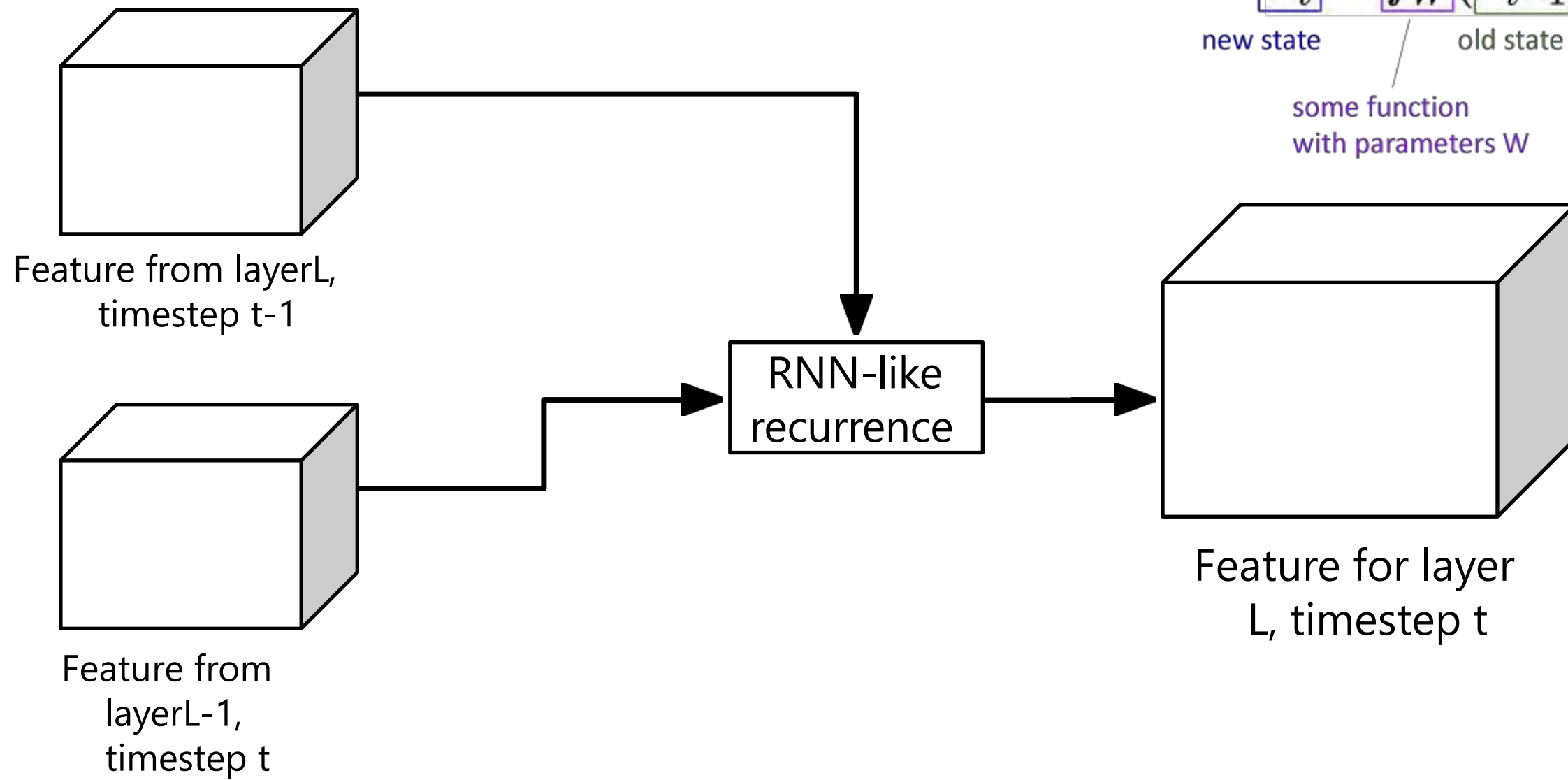
Baccouche et al, "Sequential Deep Learning for Human Action Recognition", 2011

Donahue et al, "Long-term recurrent convolution networks for visual recognition and description", CVPR 2015

RNN



RNN



Recall: Recurrent Network

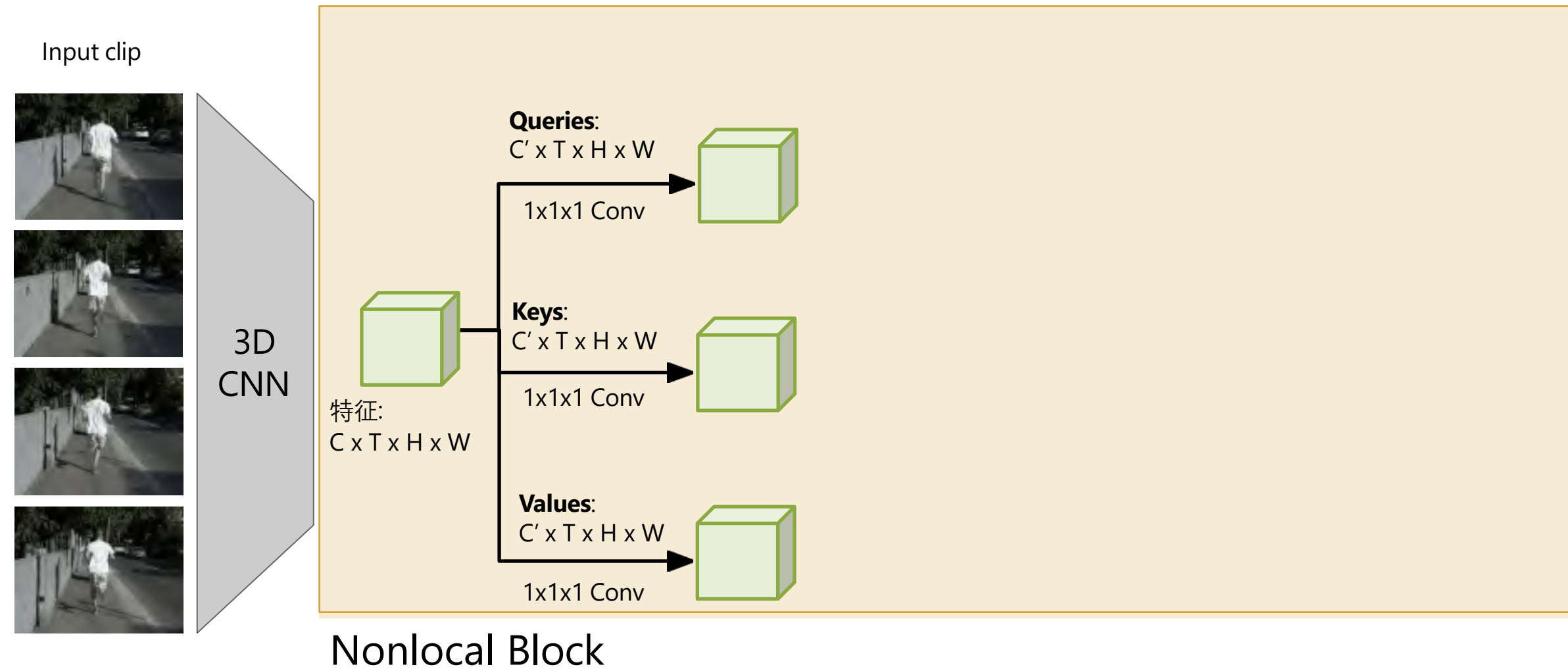
$$h_t = f_W(h_{t-1}, x_t)$$

new state

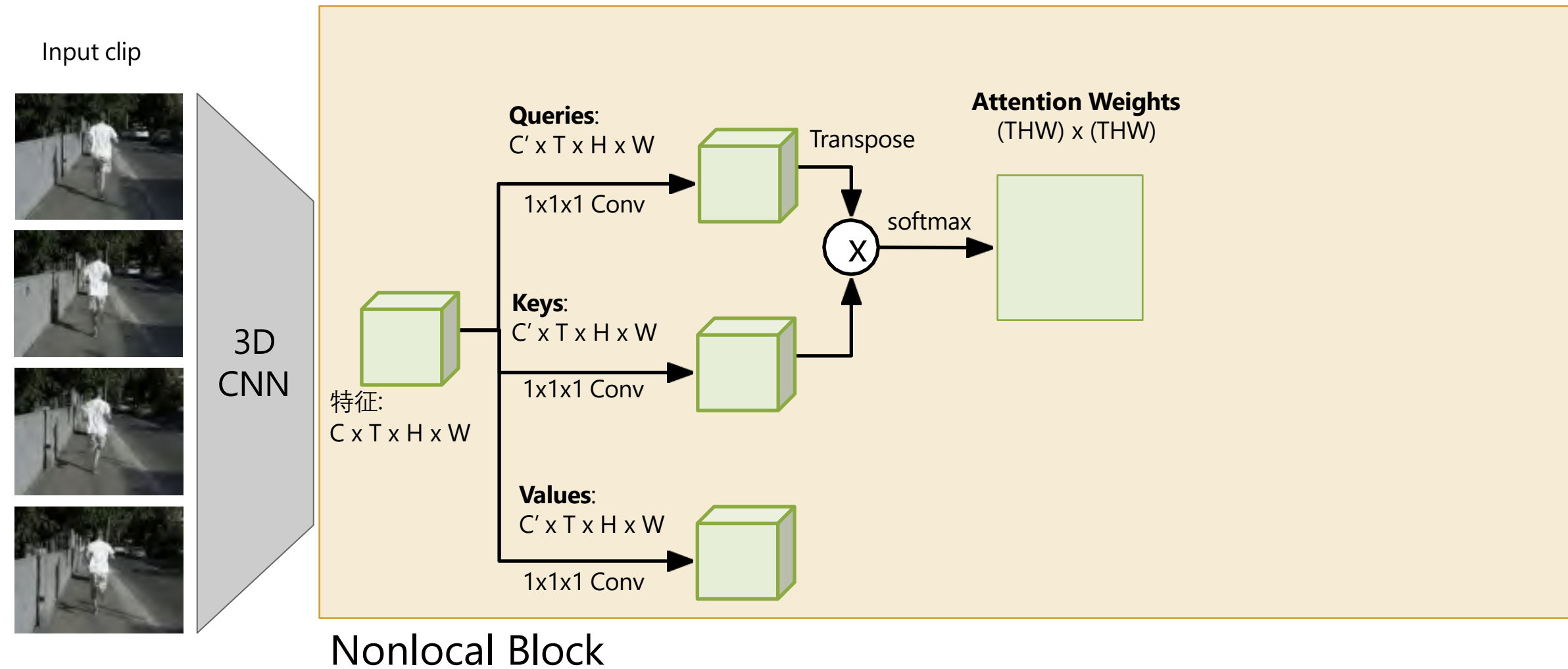
old state

some function
with parameters W

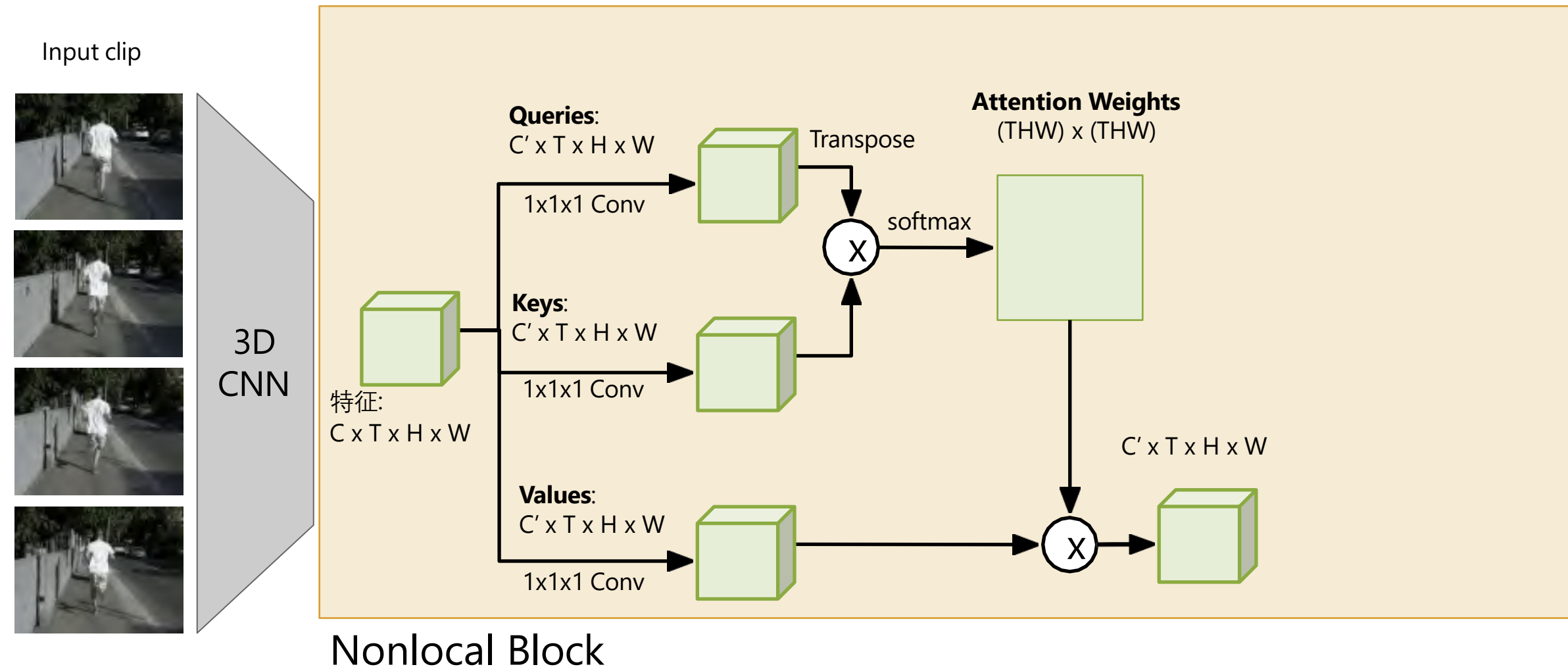
Spatio-Temporal Self-Attention (Nonlocal Block)



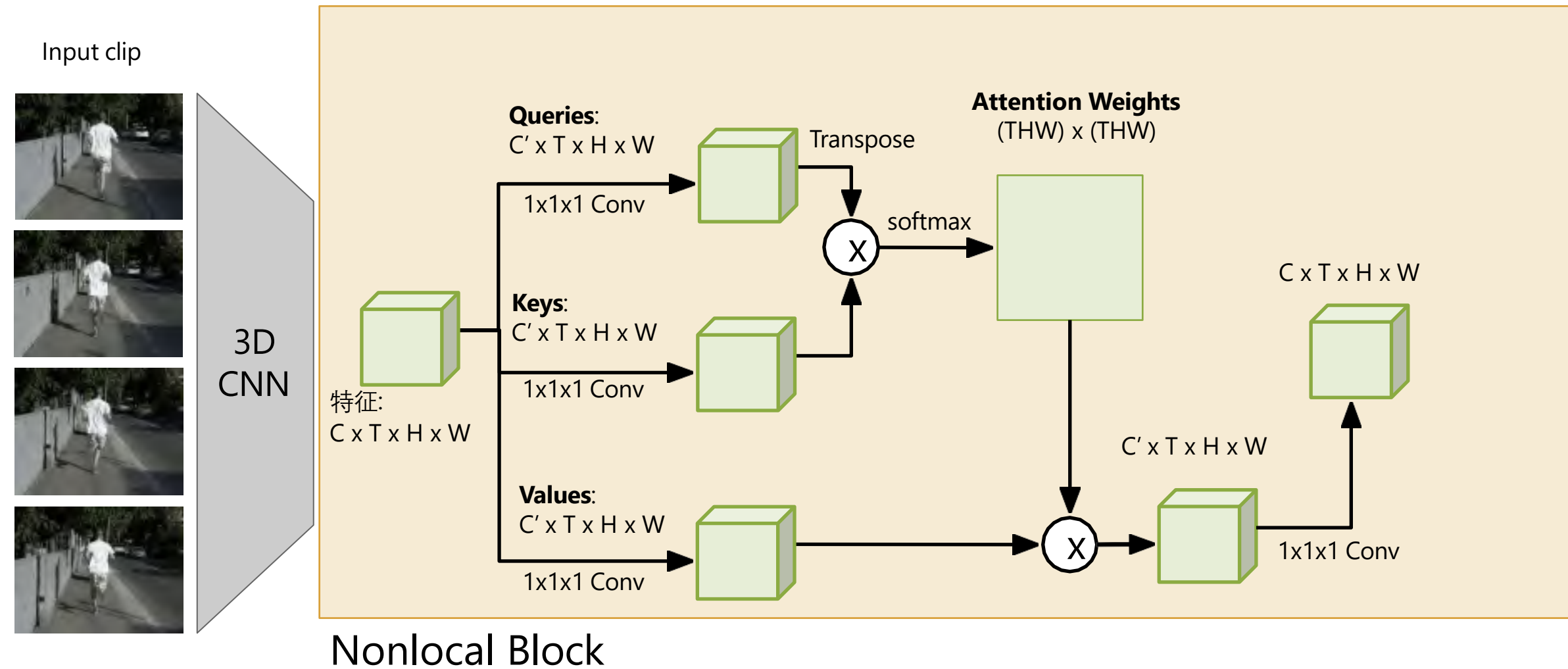
Spatio-Temporal Self-Attention (Nonlocal Block)



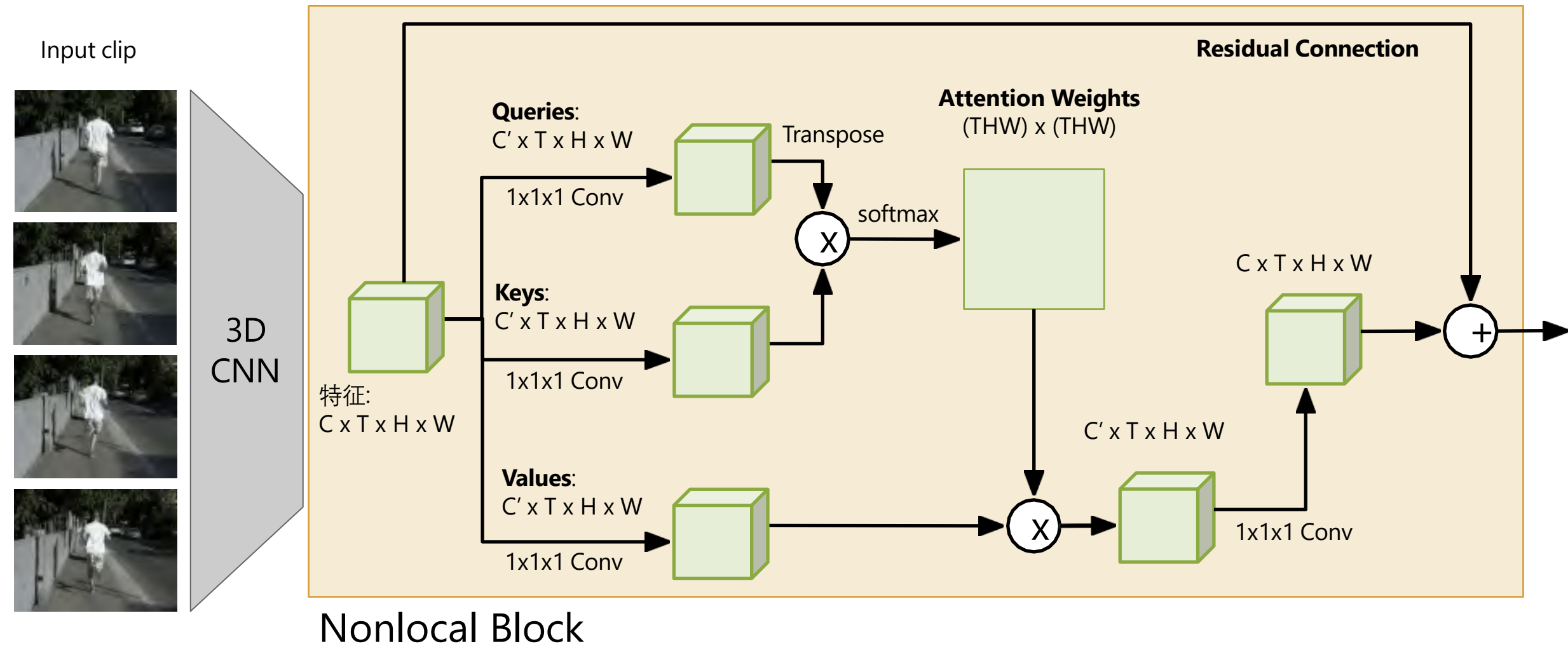
Spatio-Temporal Self-Attention (Nonlocal Block)



Spatio-Temporal Self-Attention (Nonlocal Block)

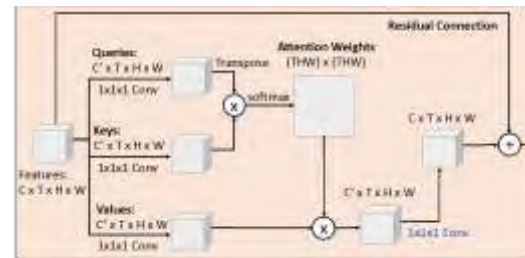
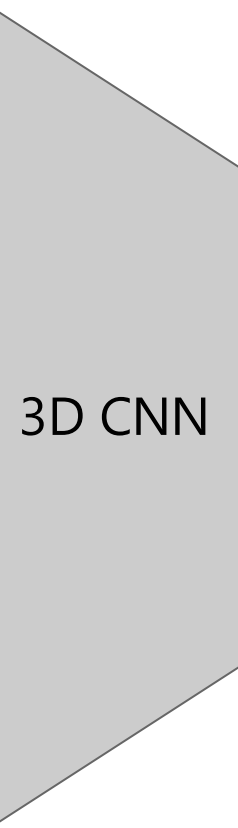
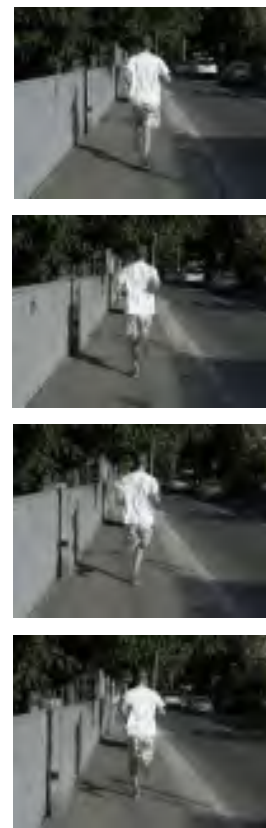


Spatio-Temporal Self-Attention (Nonlocal Block)

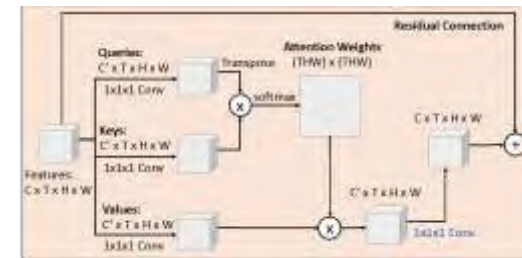
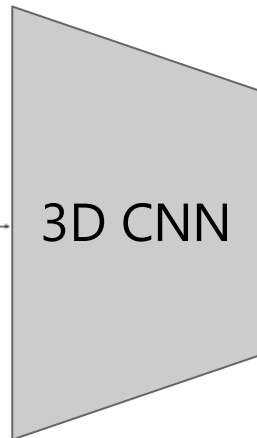


Spatio-Temporal Self-Attention (Nonlocal Block)

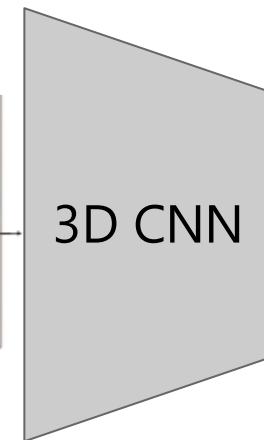
Input clip



Nonlocal Block

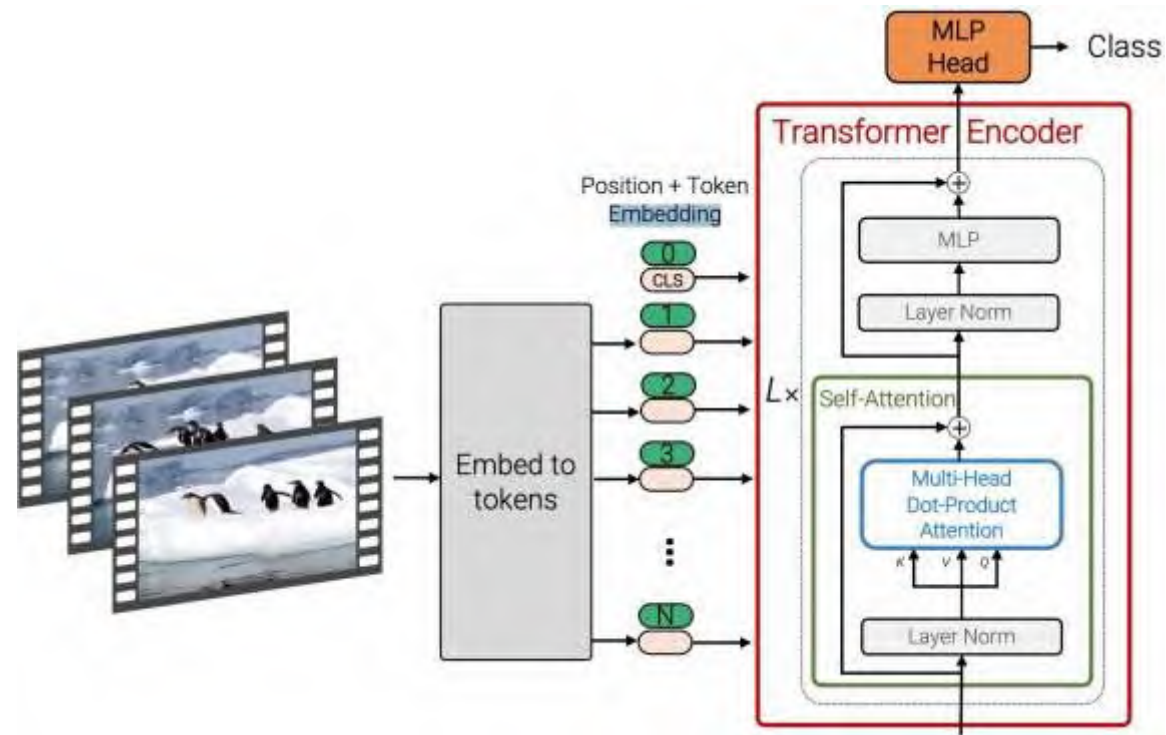


Nonlocal Block

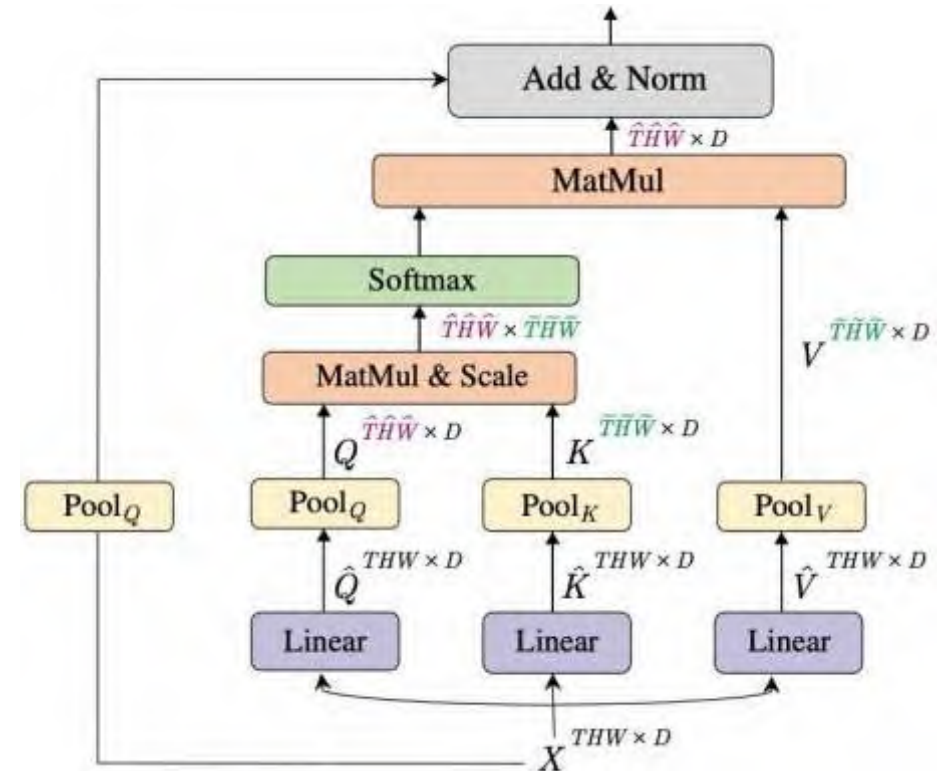


Running

Vision Transformers for Video

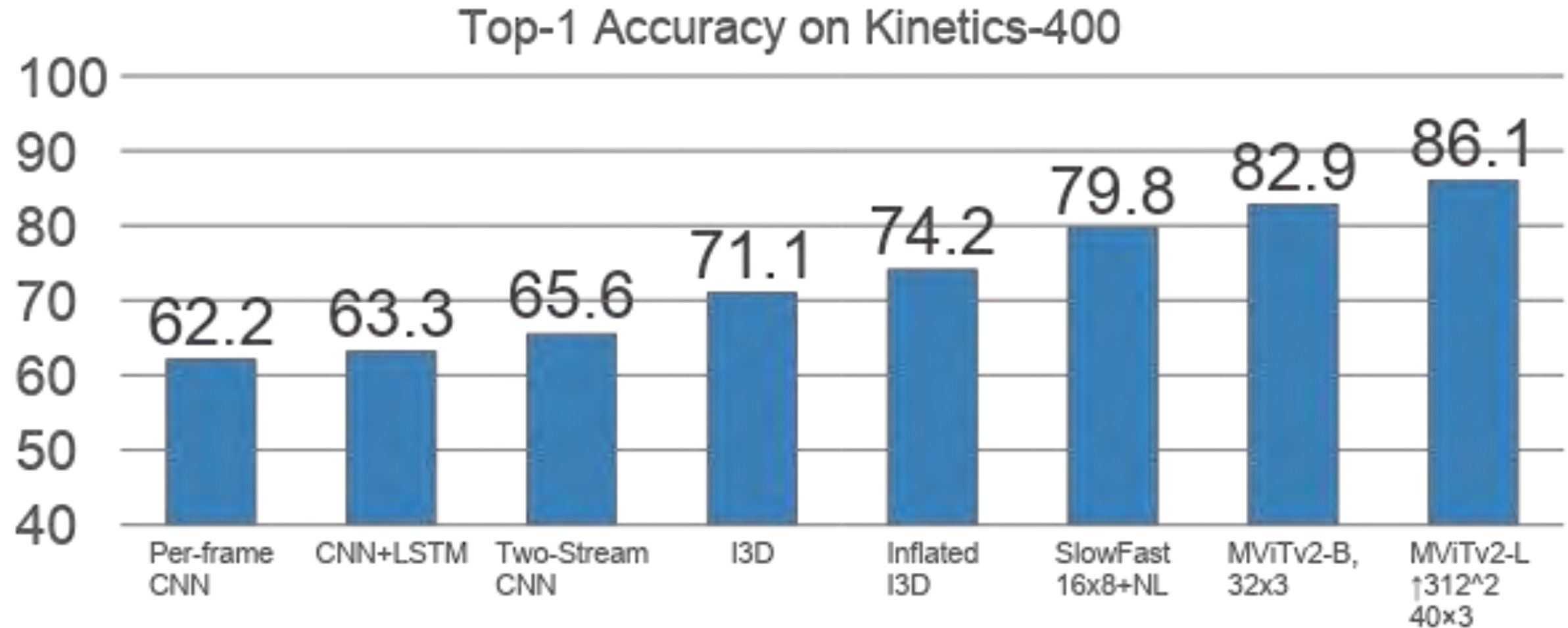


Bertasius et al, "Is Space-Time Attention All You Need for Video Understanding?", ICML 2021
 Arnab et al, "ViViT: A Video Vision Transformer", ICCV 2021
 Neimark et al, "Video Transformer Network", ICCV 2021



Fan et al, "Multiscale Vision Transformers", ICCV 2021
 Li et al, "MViTv2: Improved Multiscale Vision Transformers for classification and Detection", CVPR 2022

Vision Transformers for Video



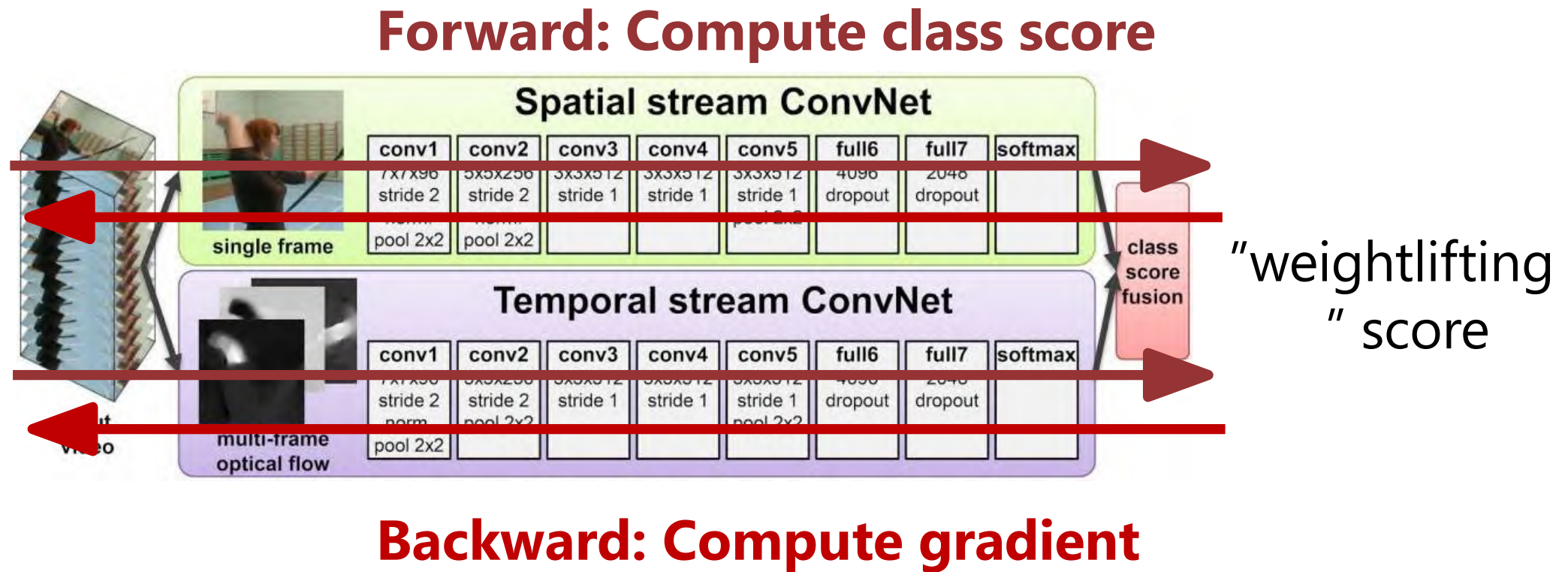
Li et al, "MViTv2: Improved Multiscale Vision Transformers for 分类 and Detection", CVPR 2022

可视化 视频模型

Image



Flow



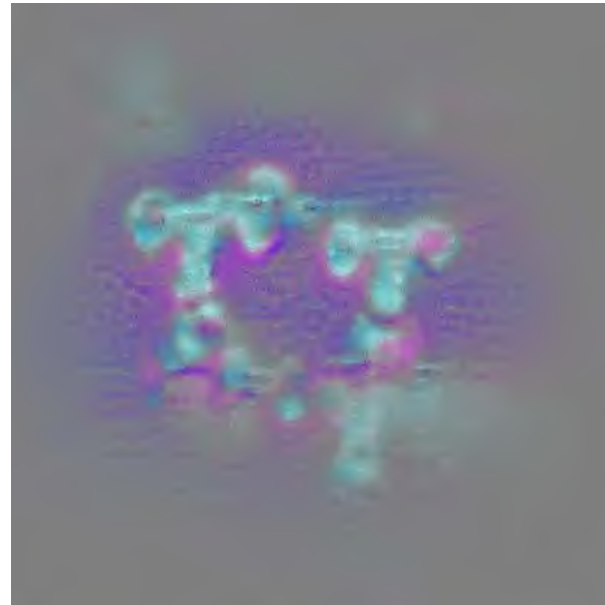
猜猜是什么动作？

Appearance
motion

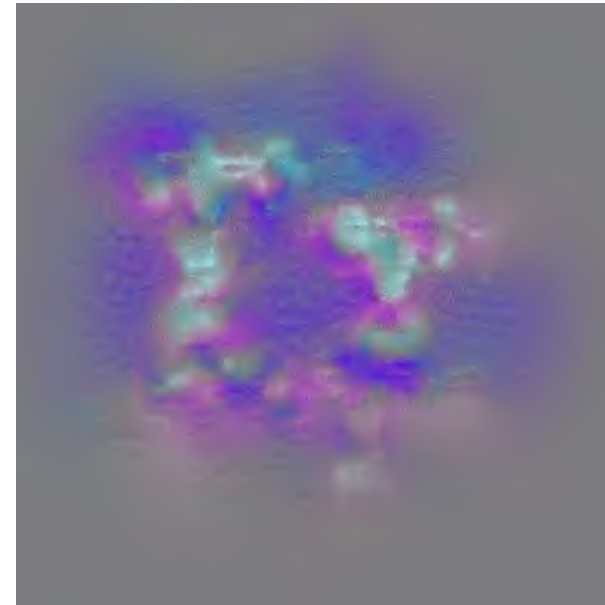


Fast motion
appearance

"Slow"



"Fast" motion



猜猜是什么动作?

Weightlifting

Appearance
motion



"Slow"

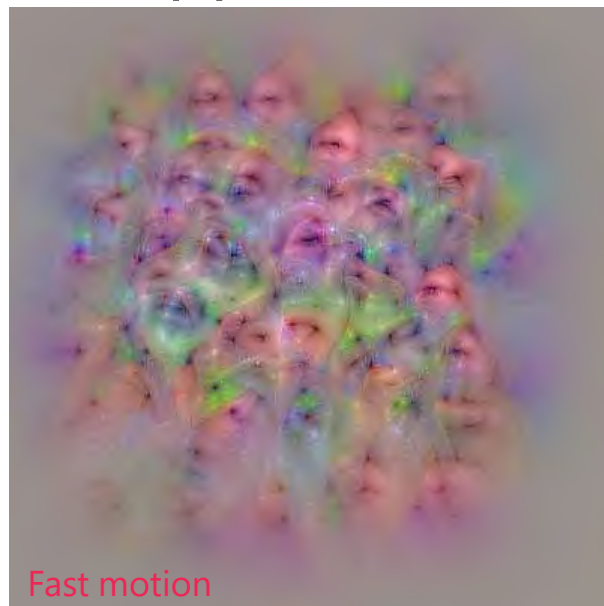


"Fast" motion



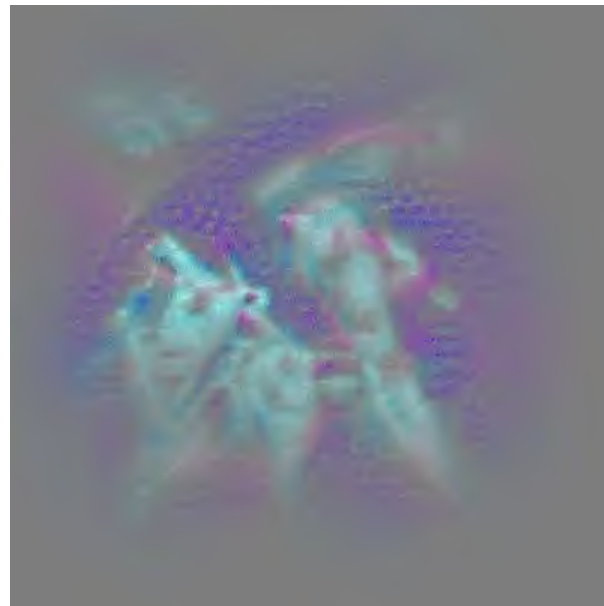
猜猜是什么动作？

Appearance

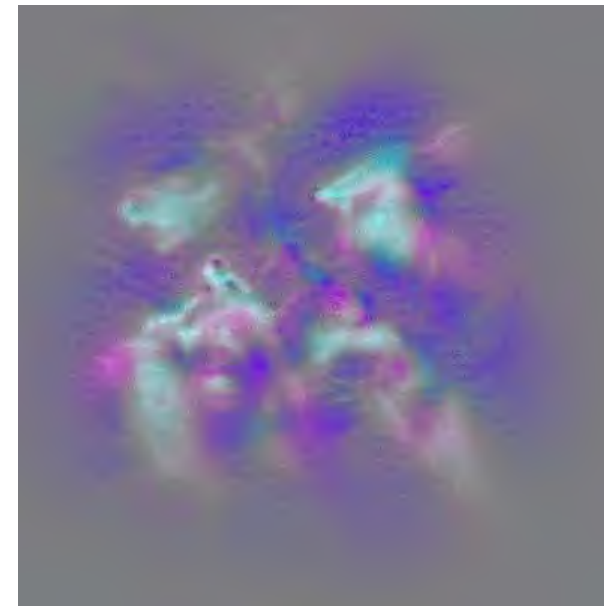


Fast motion
appearance

"Slow"



"Fast" motion



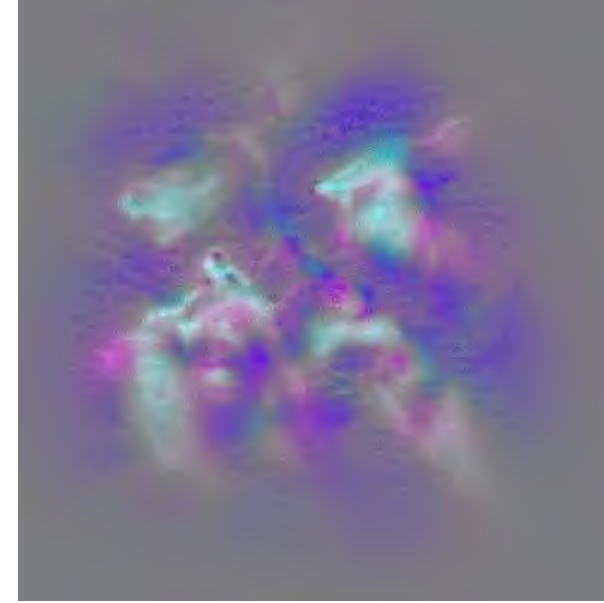
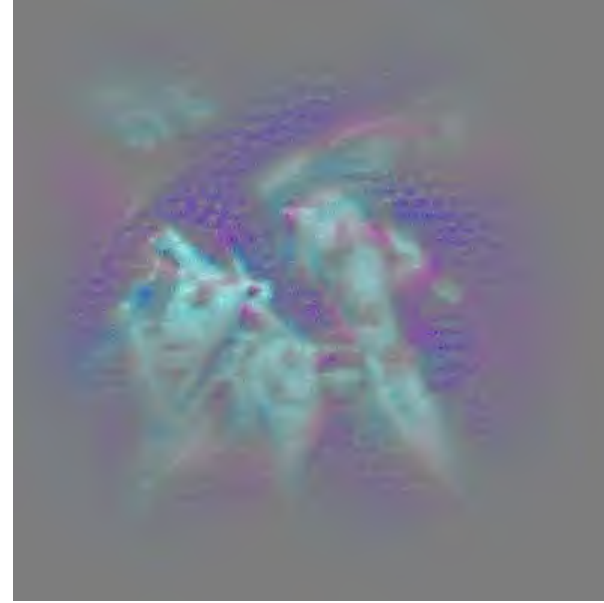
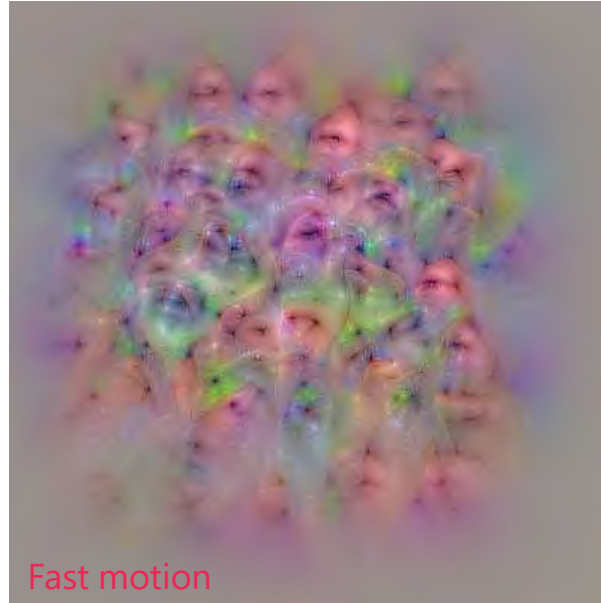
猜猜是什么动作？

Apply Eye Makeup

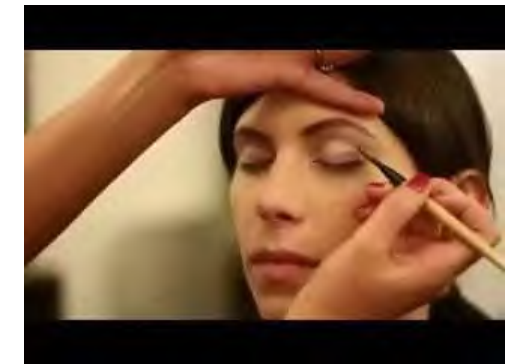
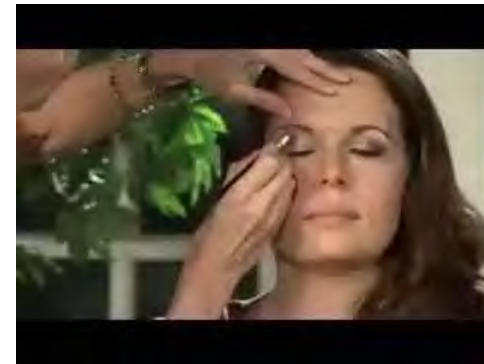
Appearance

"Slow" motion

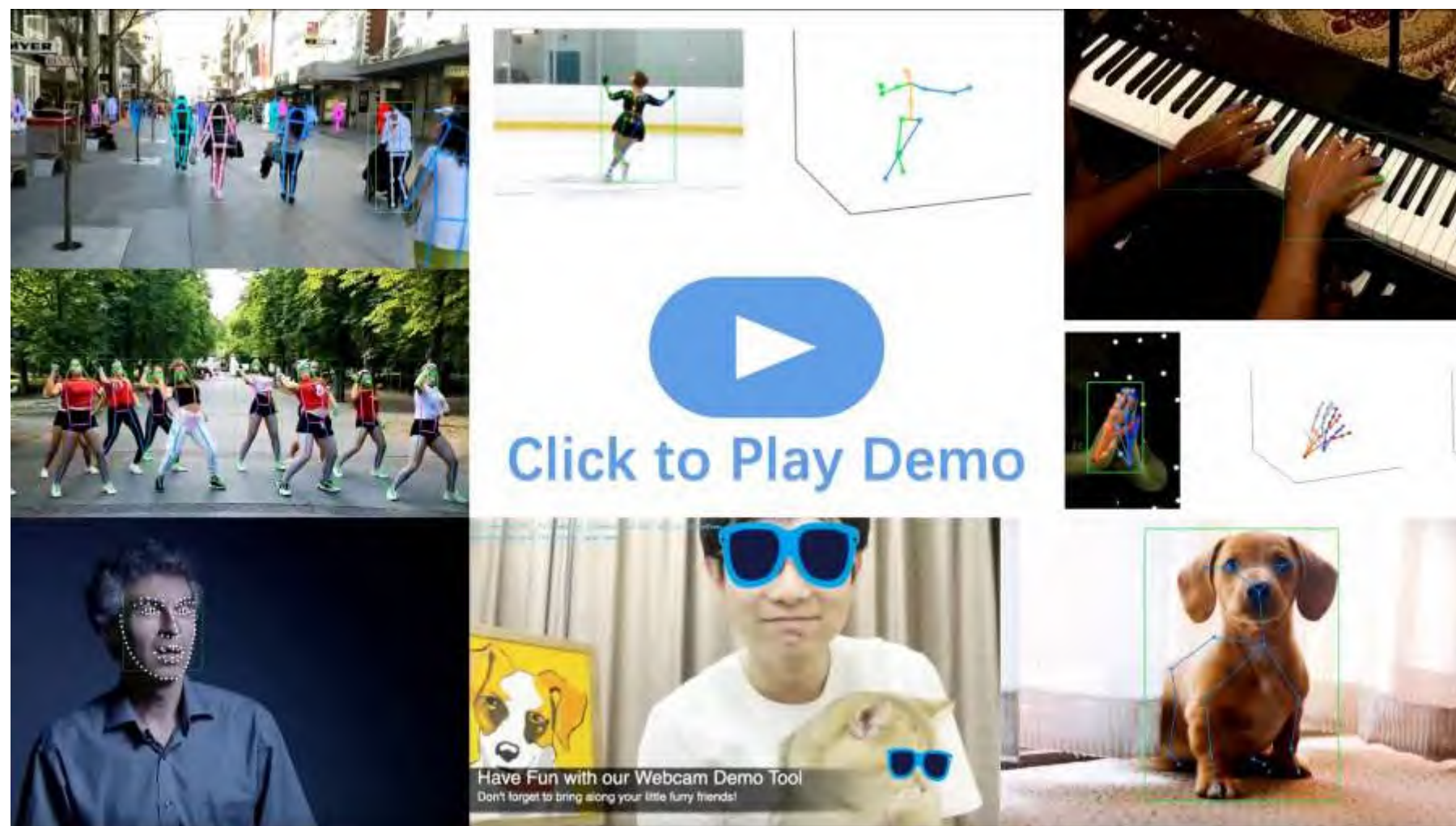
"Fast" motion



Fast motion
appearance



姿态估计

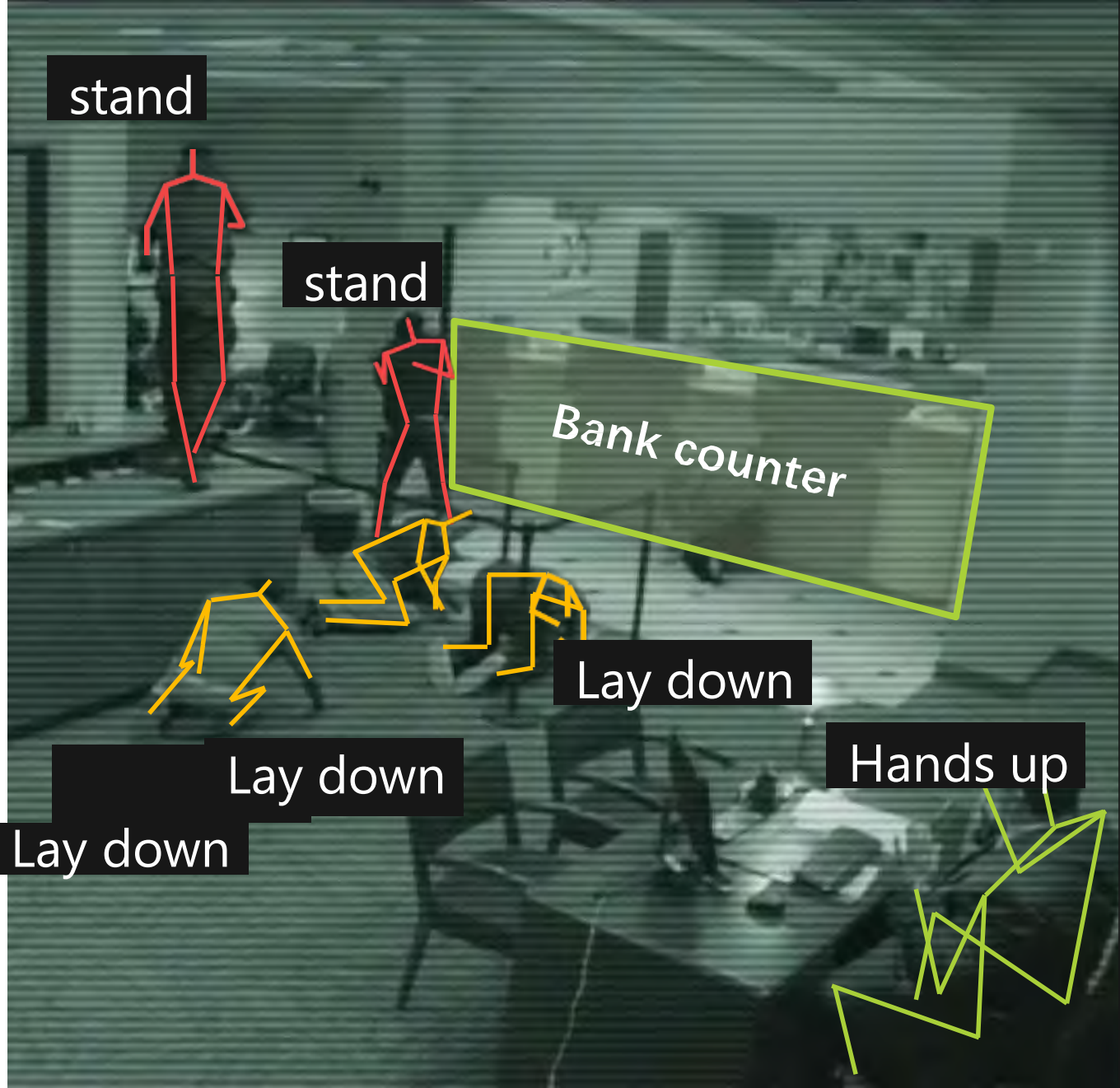


应用：动作捕捉与驱动





如何识别这个场景发生了什麼？



Cue

S

Scene: bank

Abnormal

pose

Activity:

robbery

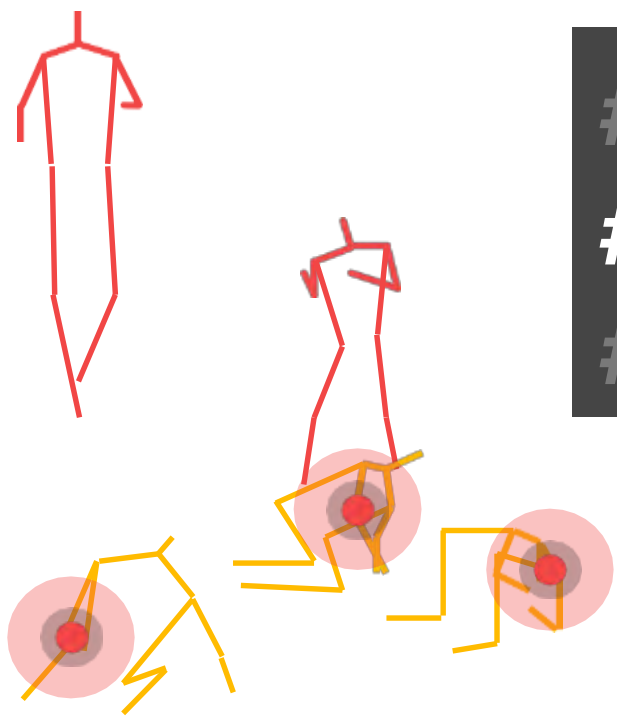


#1. Articulation

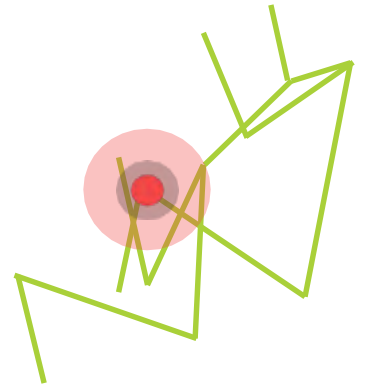
#2. Occlusion

#3. Scale variation



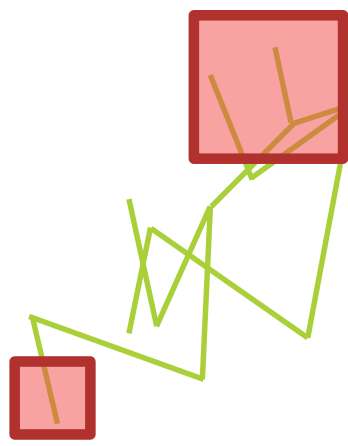


- #1. Articulation
- #2. Occlusion
- #3. Scale variation



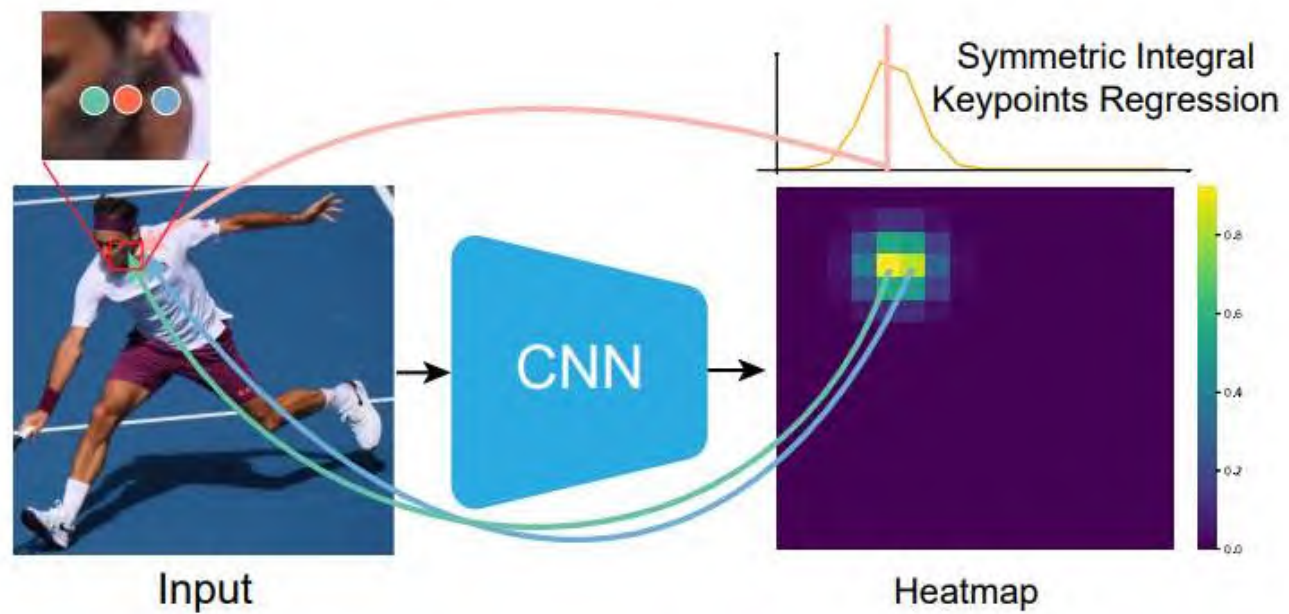


#1. Articulation
#2. Occlusion
#3. Scale variation



与分割类似

- 对热度图做回归



姿态划分

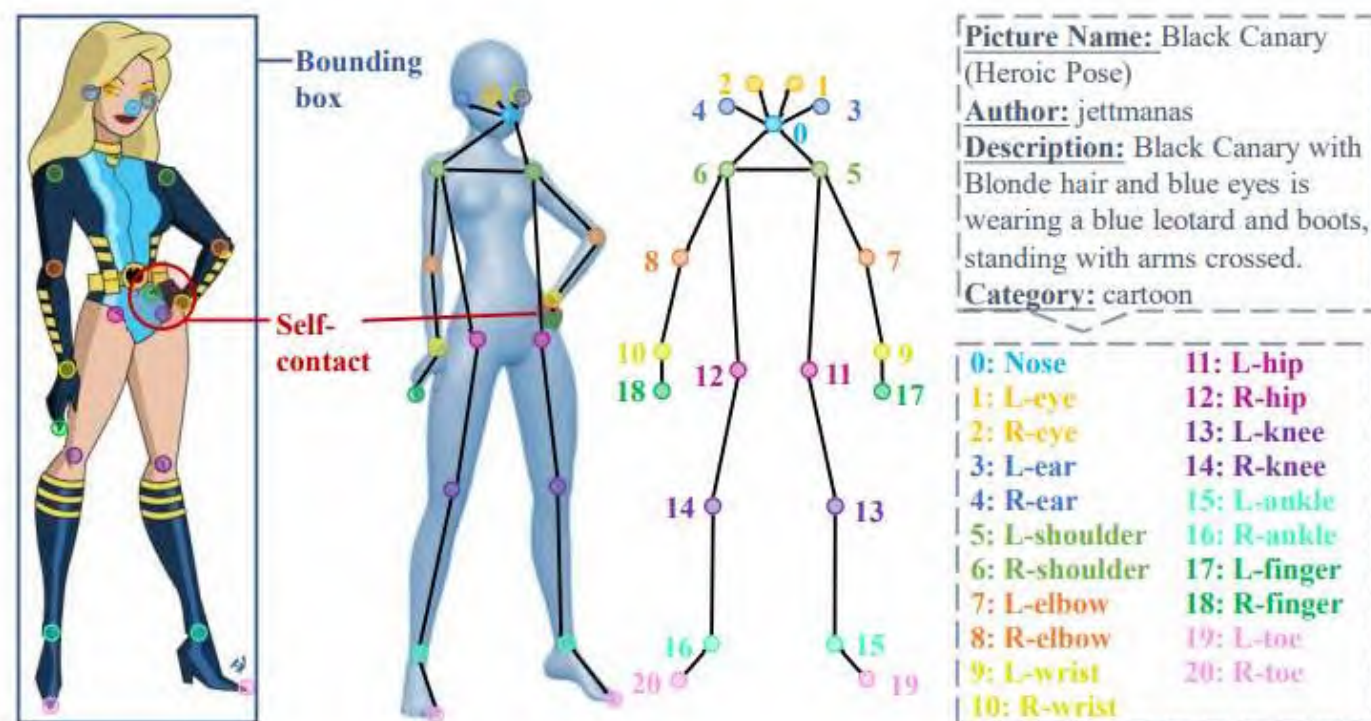
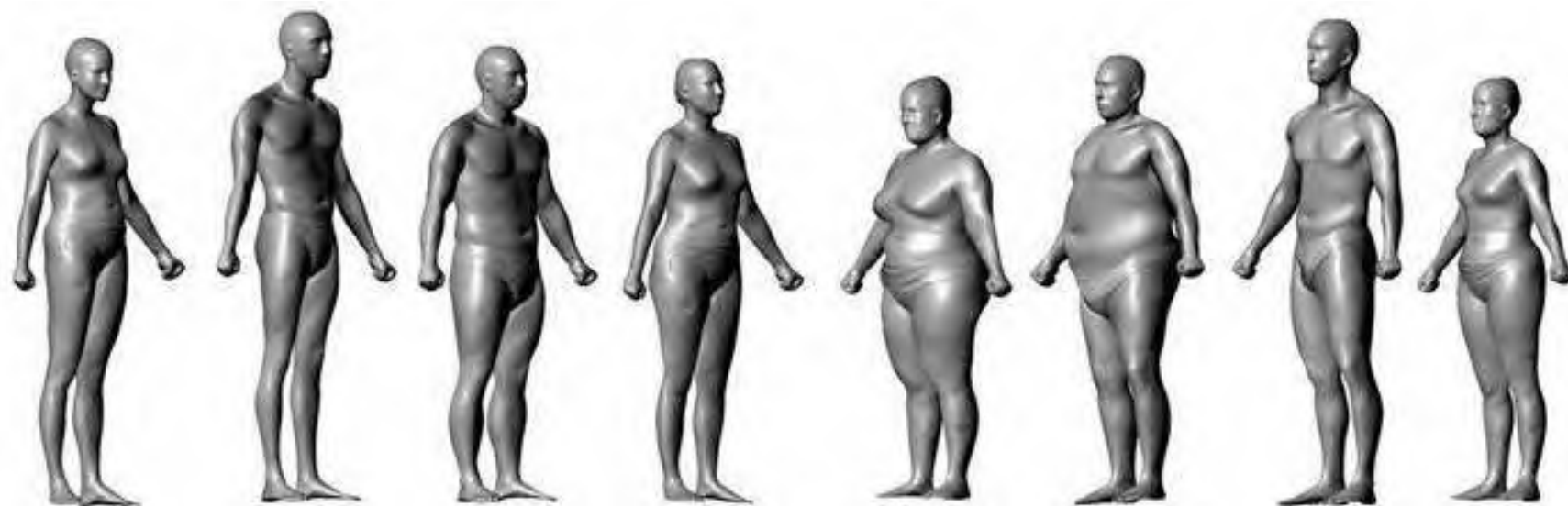


Figure 3. Illustration of the provided annotations including 2D keypoints, bounding box, self-contact point, and text description.

尺度问题

- 不同体型



Yipin Yang, Yao Yu, Yu Zhou, Sidan Du, James Davis, Ruigang Yang. Semantic Parametric Reshaping of Human Body Models. In 3DV Workshop on Dynamic Shape Measurement and Analysis, 2014.

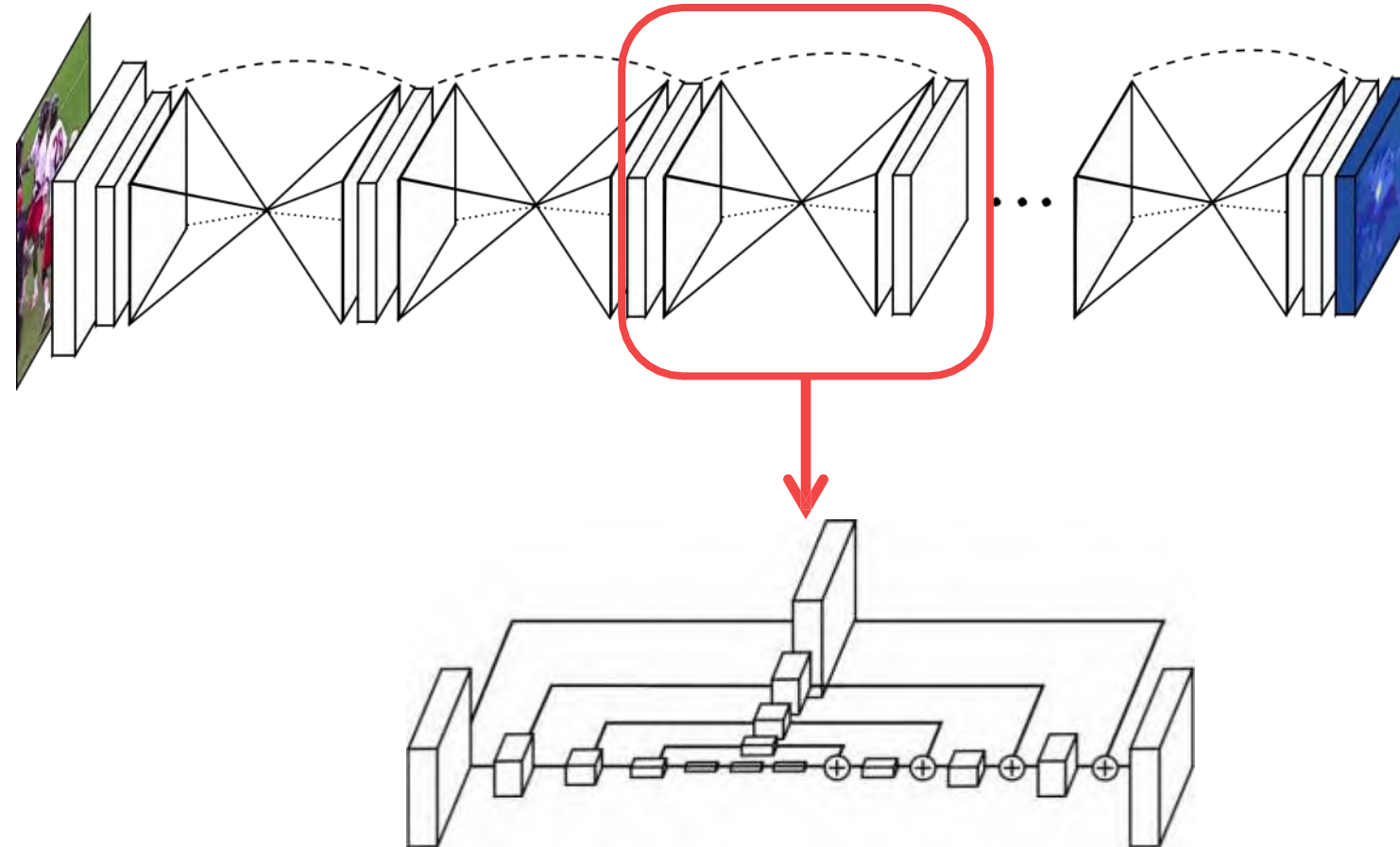
尺度问题

- 不同视角和距离



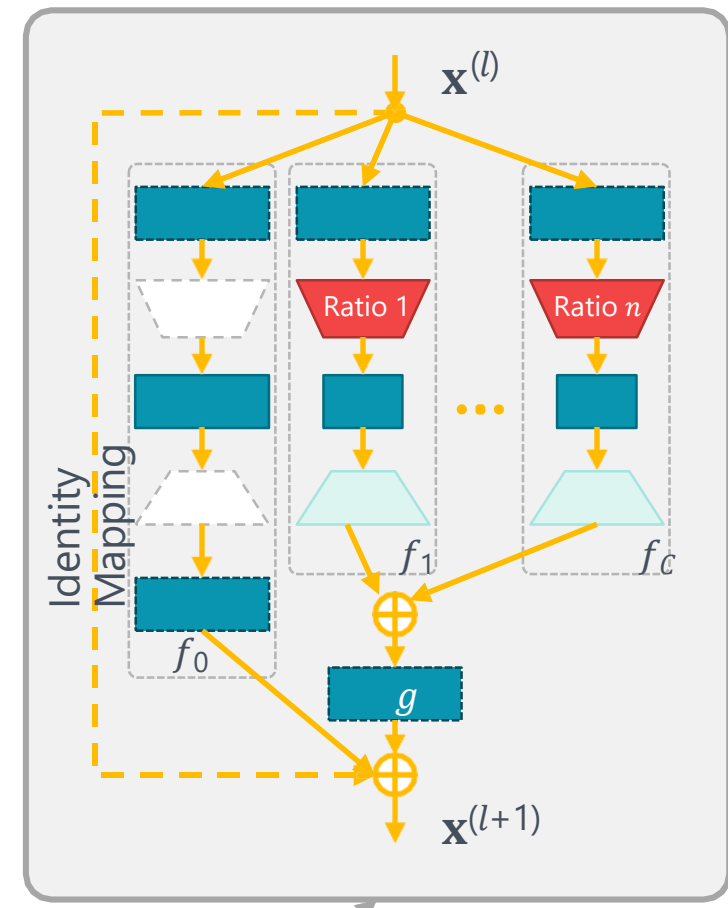
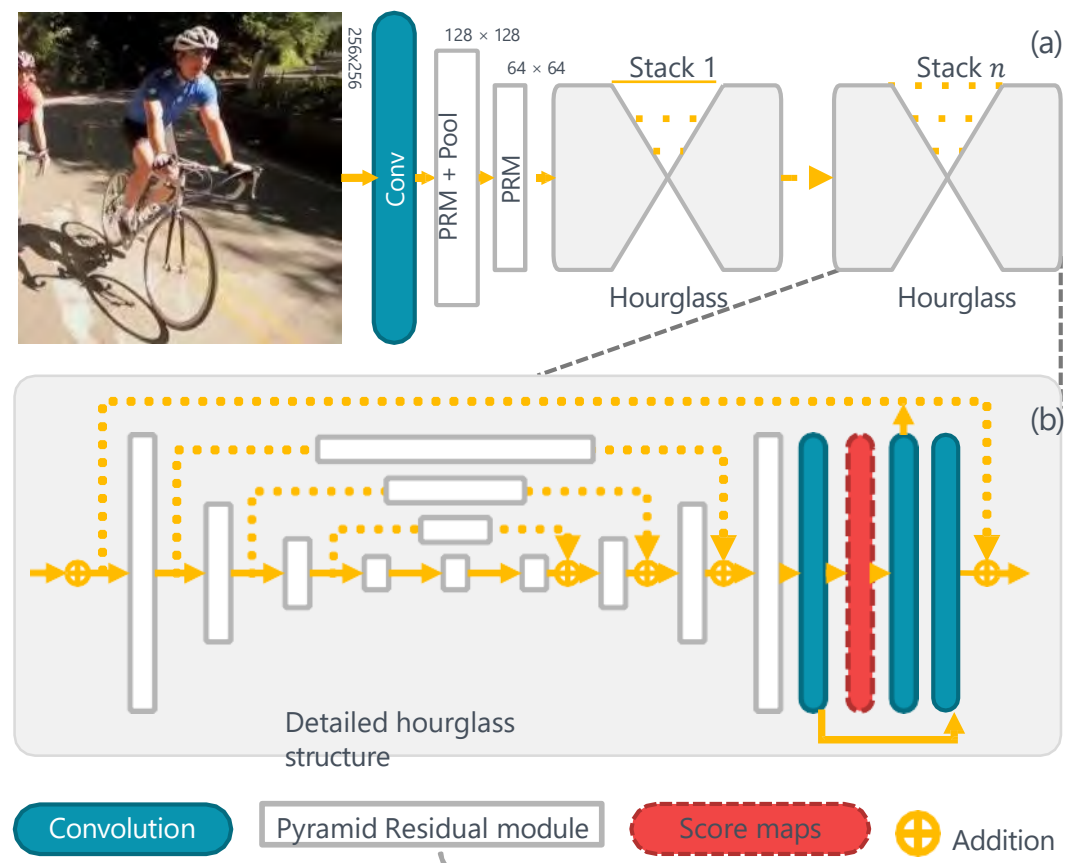
Learning Feature Pyramids for Human Pose Estimation
Wei Yang , Shuang Li, Wanli Ouyang, Hongsheng Li, Xiaogang Wang
ICCV, 2017

Hourglass (类比FCN+Unet)



Newell A, Yang K, Deng J. Stacked hourglass networks for human pose estimation[C]//European Conference on Computer Vision. Springer, Cham, 2016: 483-499.

Pyramid Residual Modules (类比DeepLab+Unet)

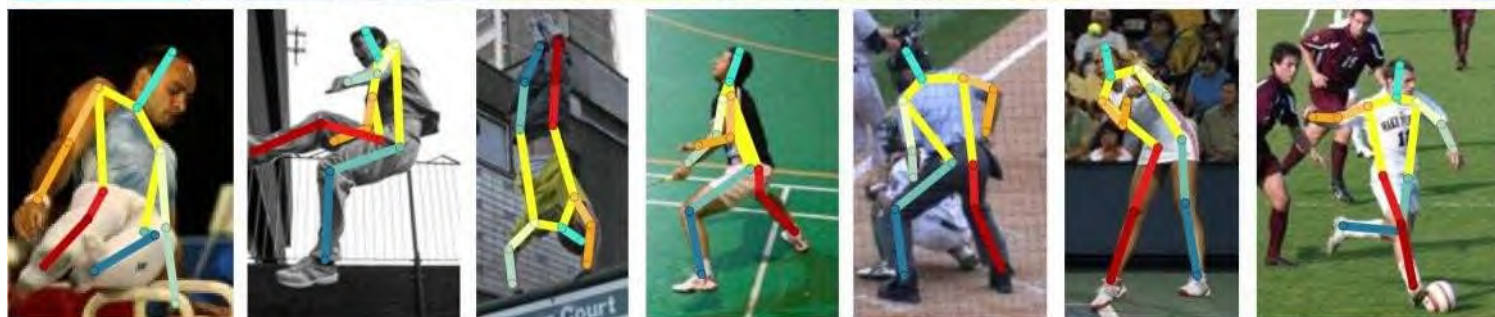


数据标注

MPII dataset



LSP dataset



数据迁移：到没有标注的数据上

Constrained scenes



Phone

Ours



Ground-truth



Domain

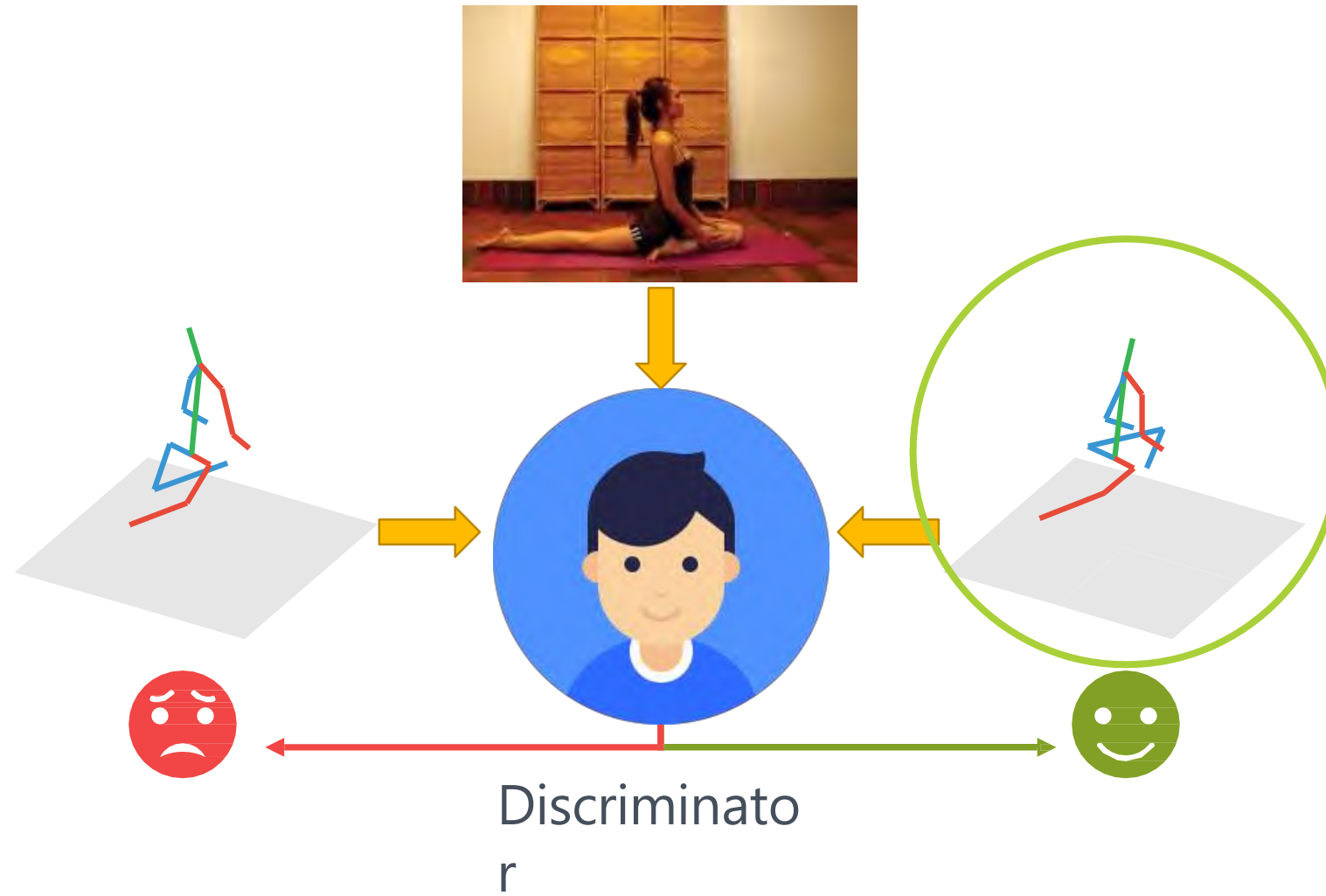


Discrepancy

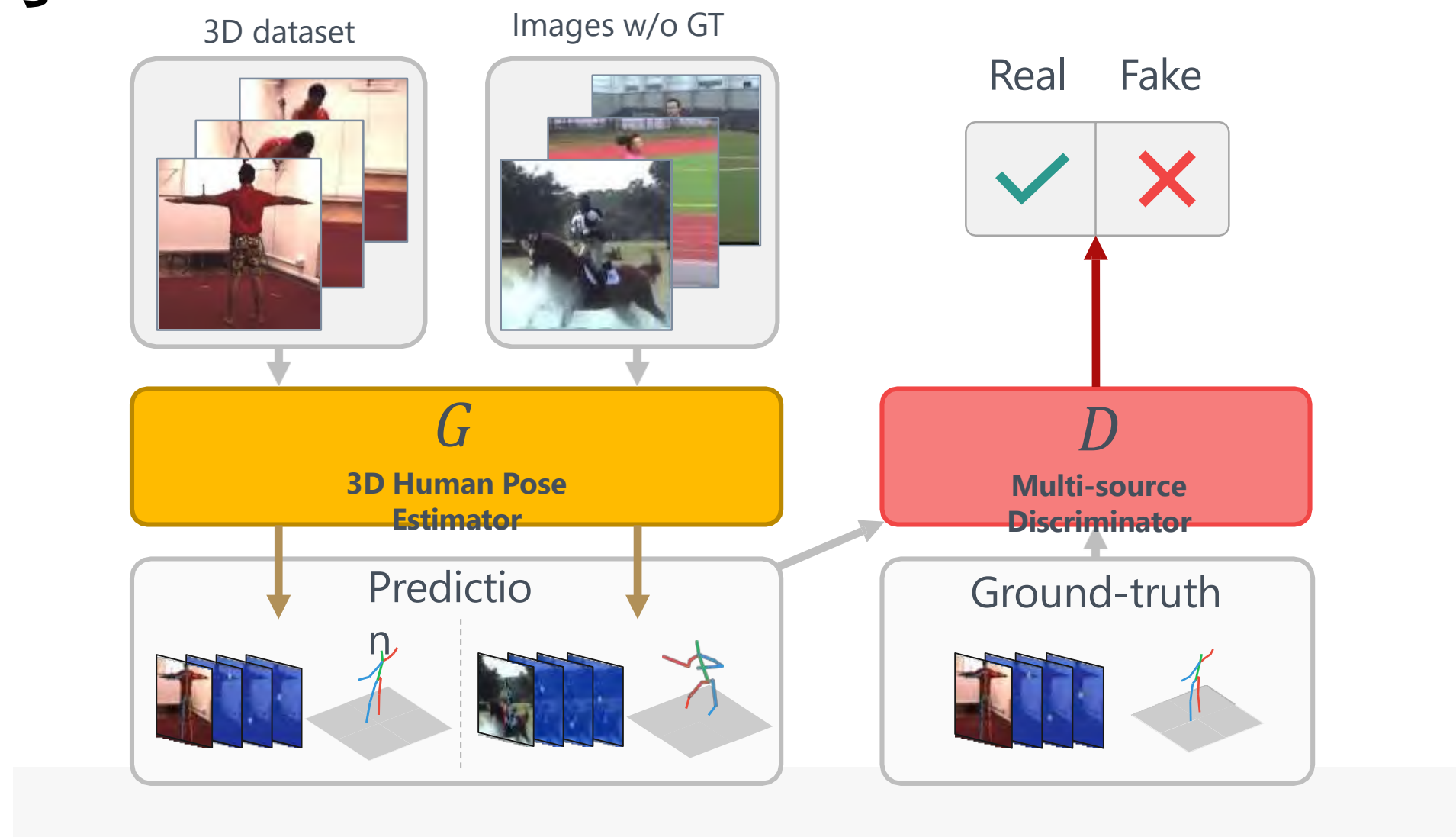
In-the-wild scenes



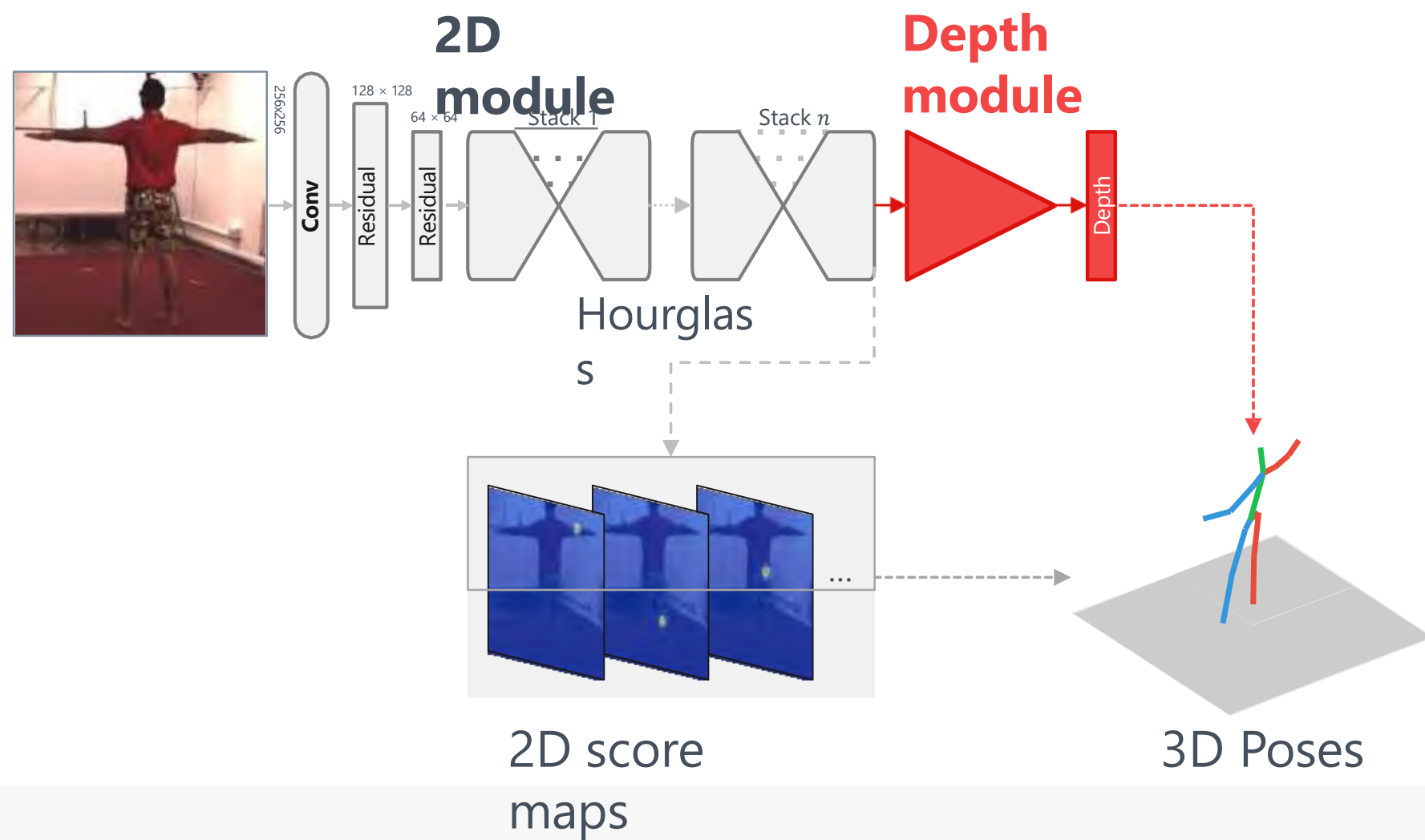
哪个动作看起来更真实？



对抗学习

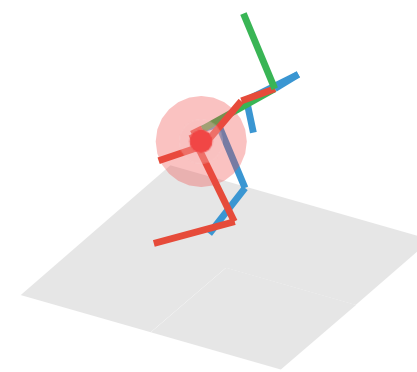
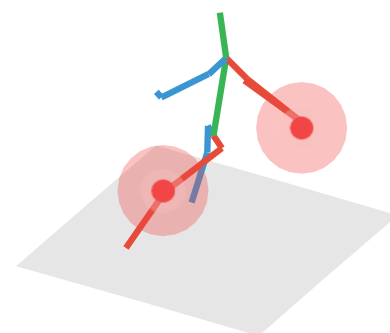


2D到3D

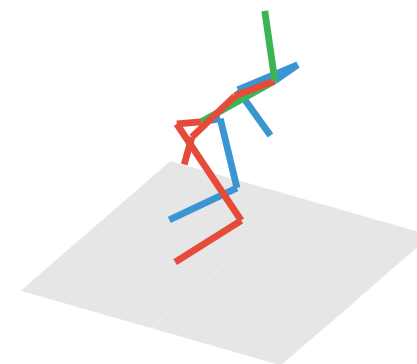
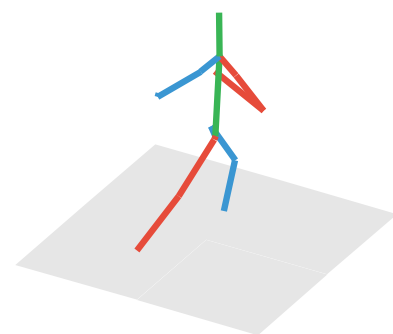


自然场景的效果

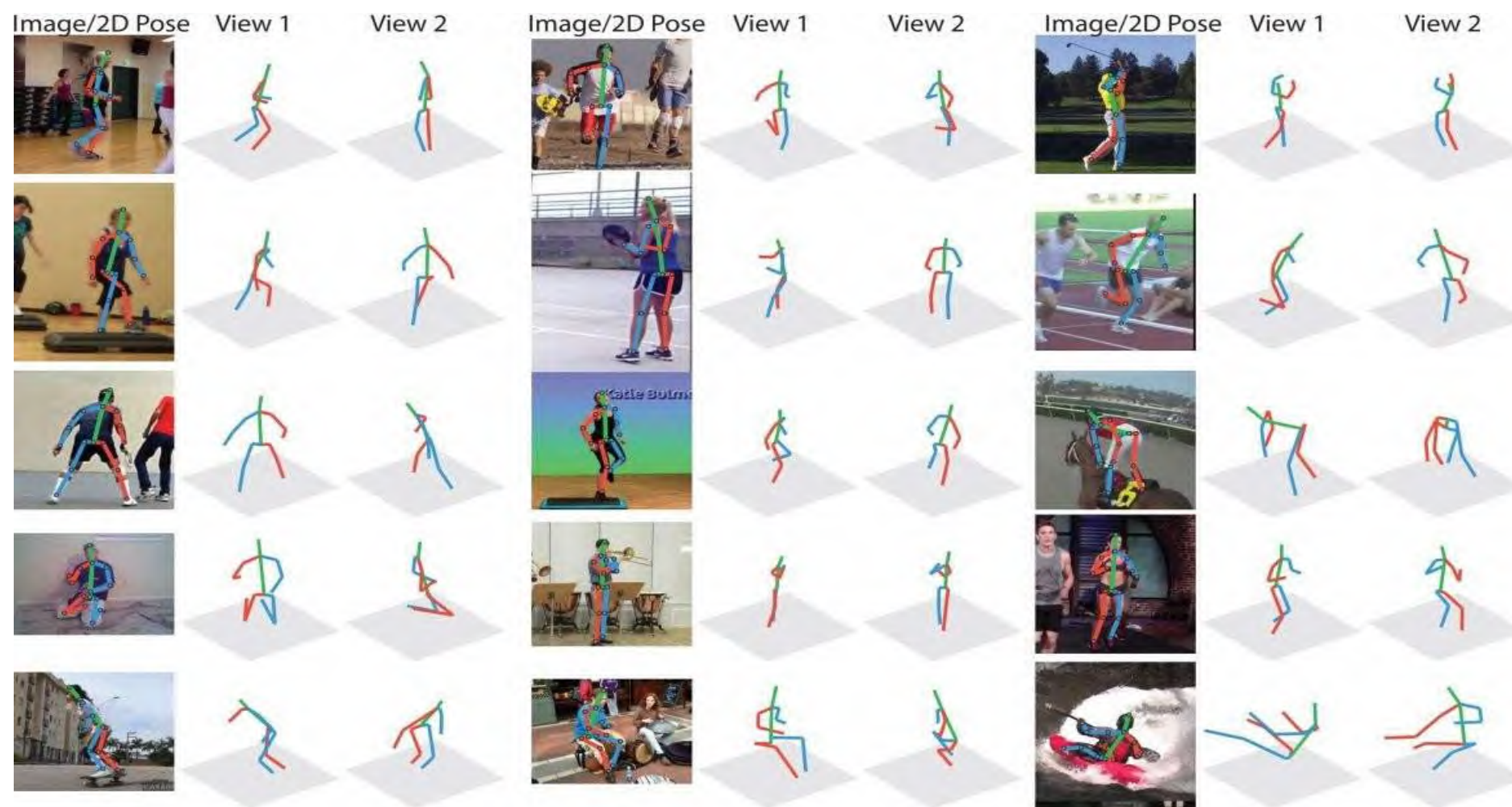
baseline



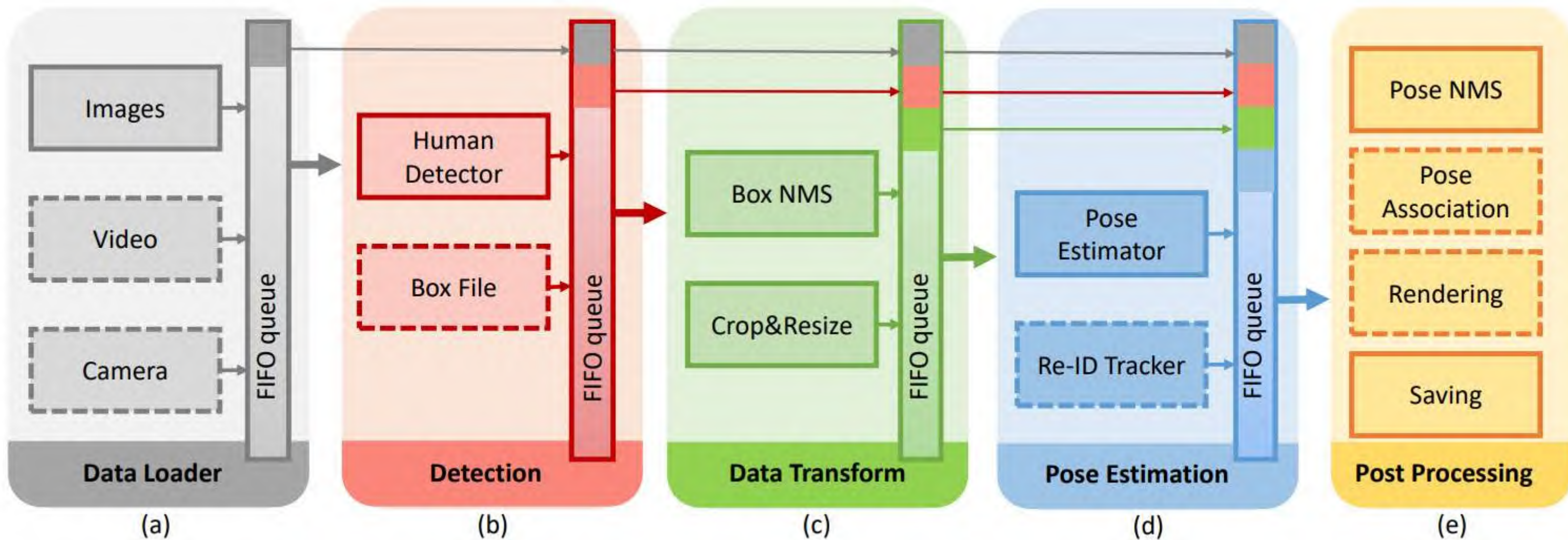
Ours



多视角效果



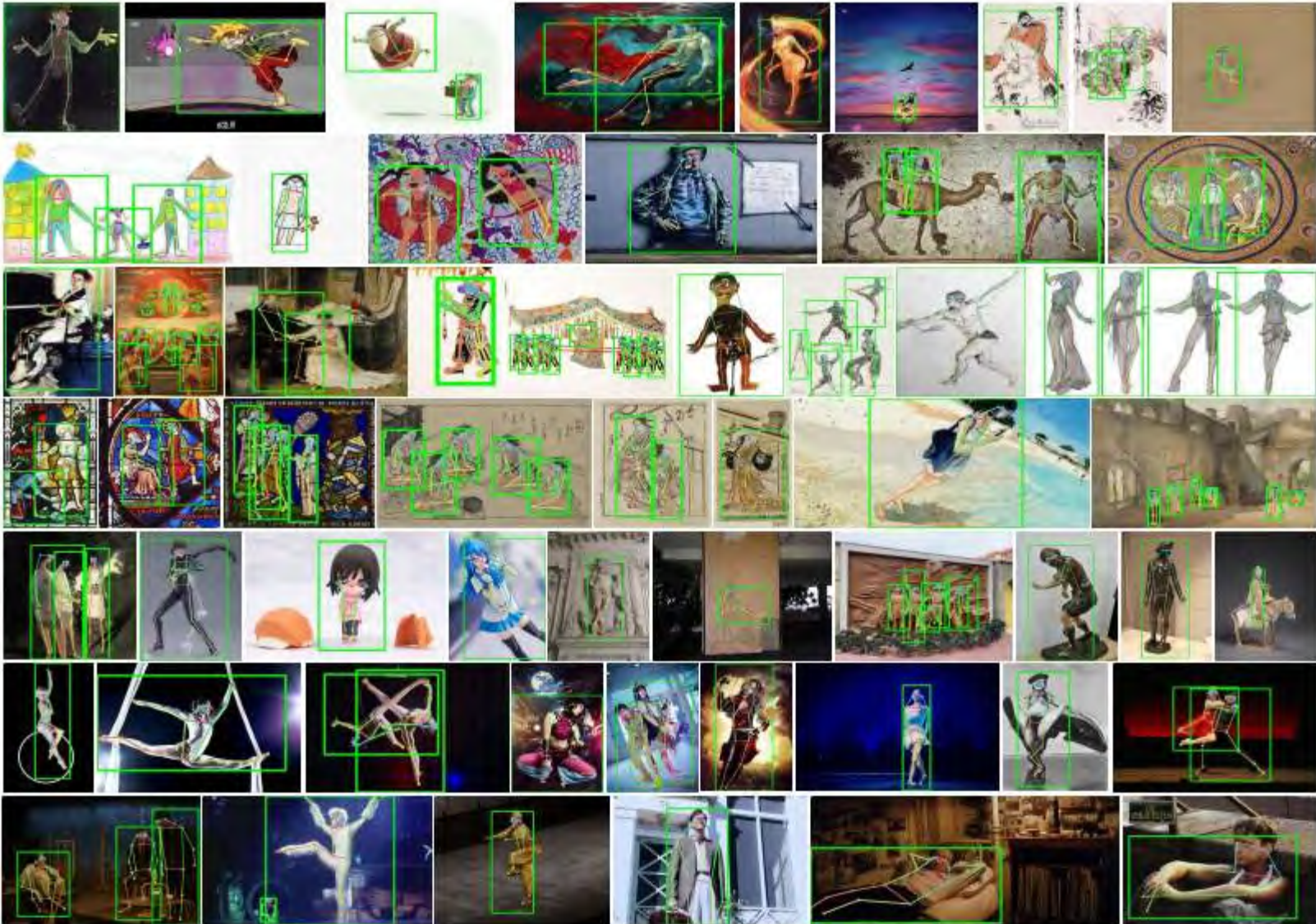
AlphaPose: 一个完整且复杂的姿态估计系统



HumanArt



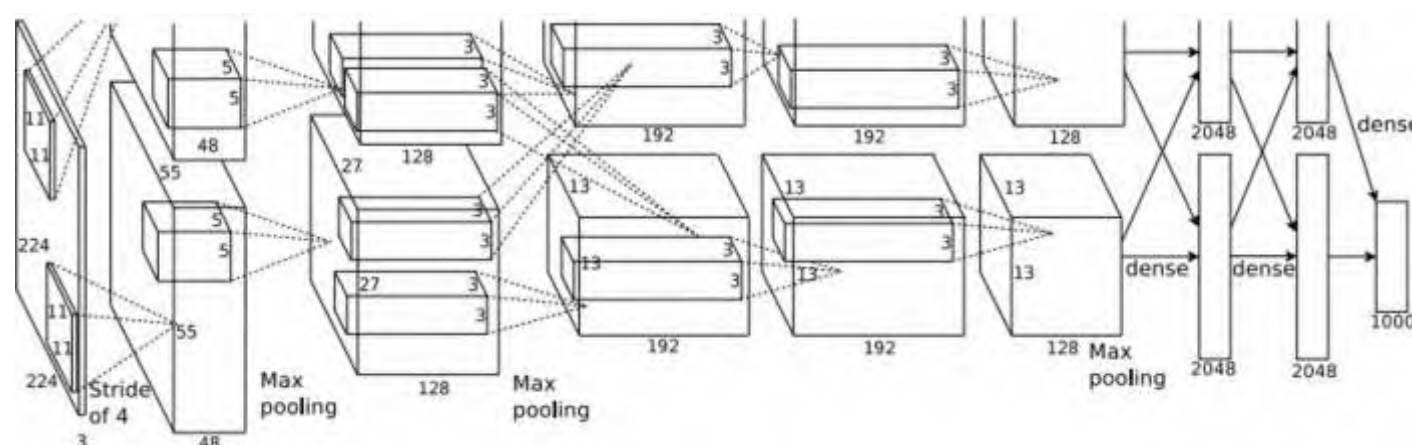
HumanArt



卷积网络到底学到了什么？



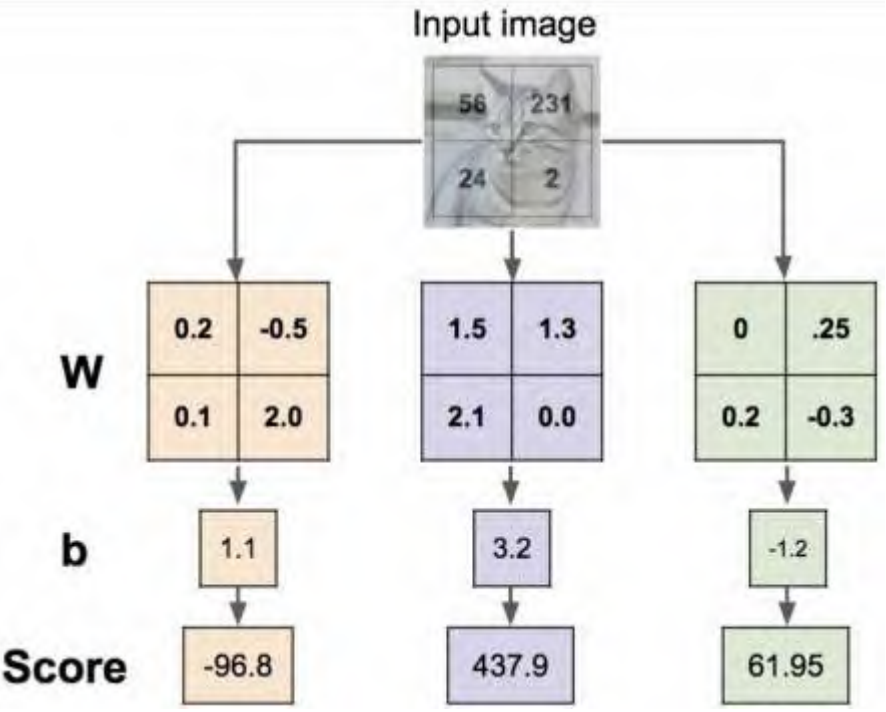
输入图像:
3 x 224 x 224



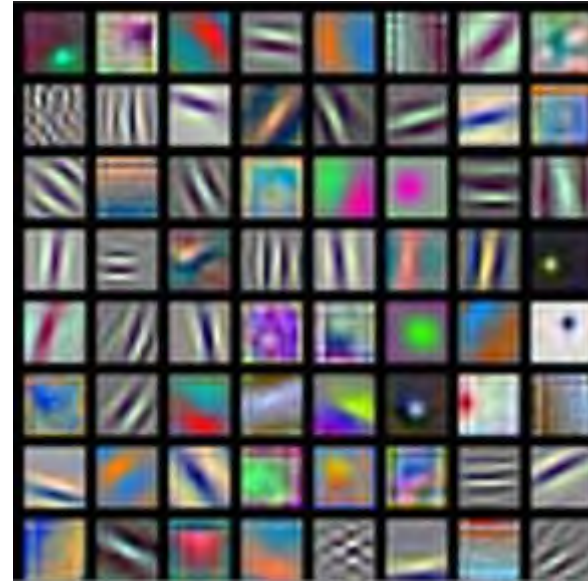
类别得分:
1000 numbers

↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑
中间特征长什么样？

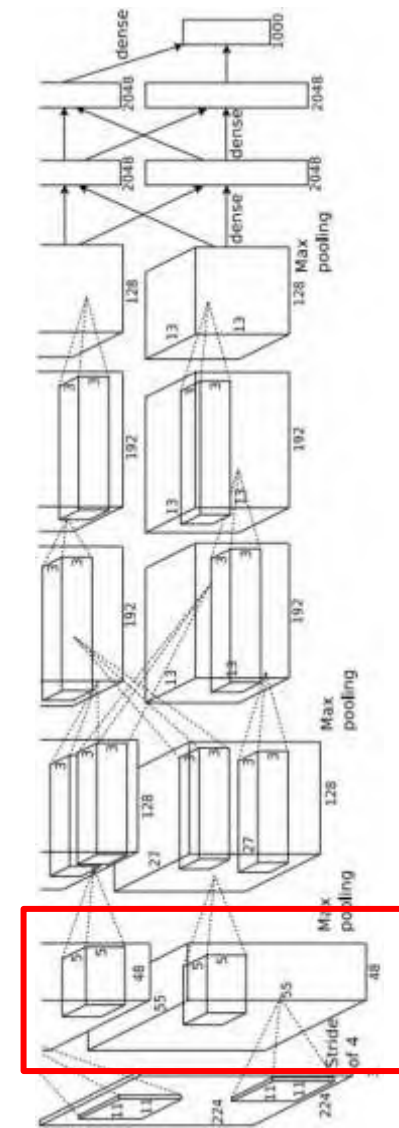
Interpreting a Linear Classifier: Visual Viewpoint



可视化卷积核

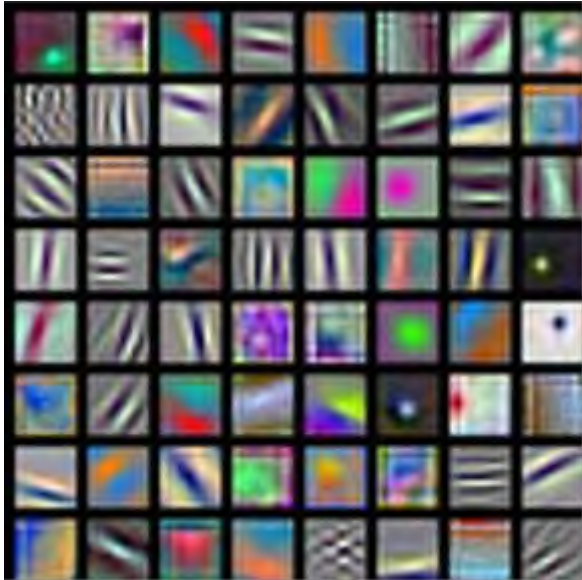


AlexNet:
64 x 3 x 11 x 11

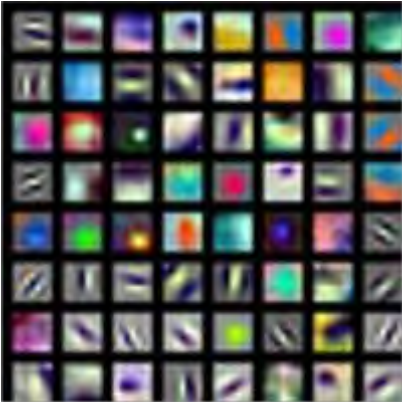


Krizhevsky, "One weird trick for parallelizing 卷积 neural networks", arXiv 2014 He et al, "Deep Residual Learning for Image Recognition", CVPR 2016
Huang et al, "Densely Connected 卷积 Networks", CVPR 2017

可视化卷积核



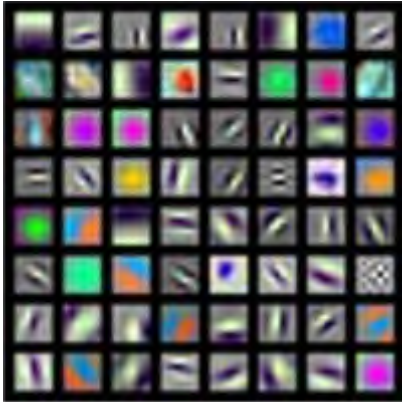
AlexNet:
64 x 3 x 11 x 11



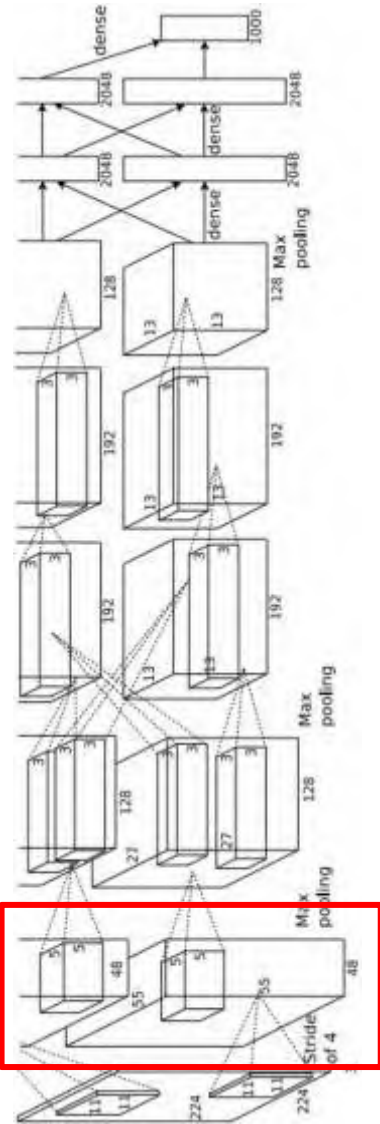
ResNet-18:
64 x 3 x 7 x 7



ResNet-101:
64 x 3 x 7 x 7



DenseNet-121:
64 x 3 x 7 x 7



Krizhevsky, "One weird trick for parallelizing 卷积 neural networks", arXiv 2014 He et al, "Deep Residual Learning for Image Recognition", CVPR 2016 Huang et al, "Densely Connected 卷积 Networks", CVPR 2017

Weights:


layer 1 weights

$$16 \times 3 \times 7 \times 7$$

Weights:


layer 2 weights

$$20 \times 16 \times 7 \times 7$$

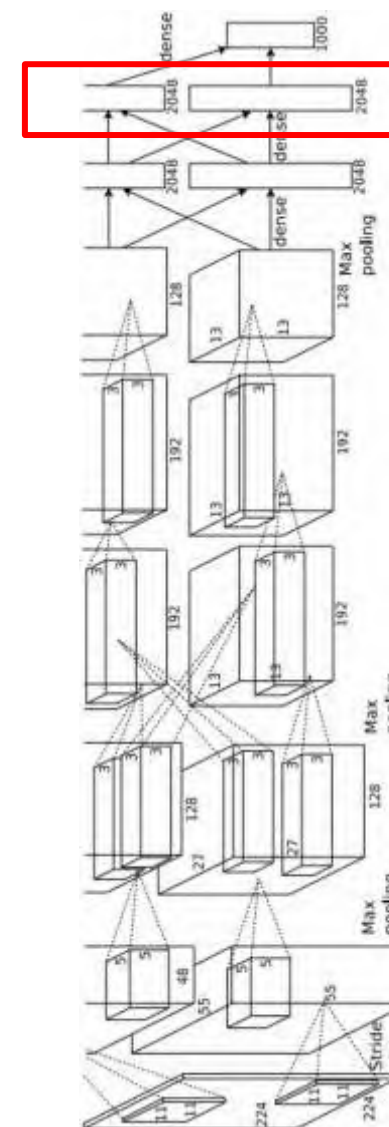
Weights:


layer 3 weights

$$20 \times 20 \times 7 \times 7$$

最后一层：近邻样本

4096-dim vector



Test image L2 Nearest neighbors in feature space



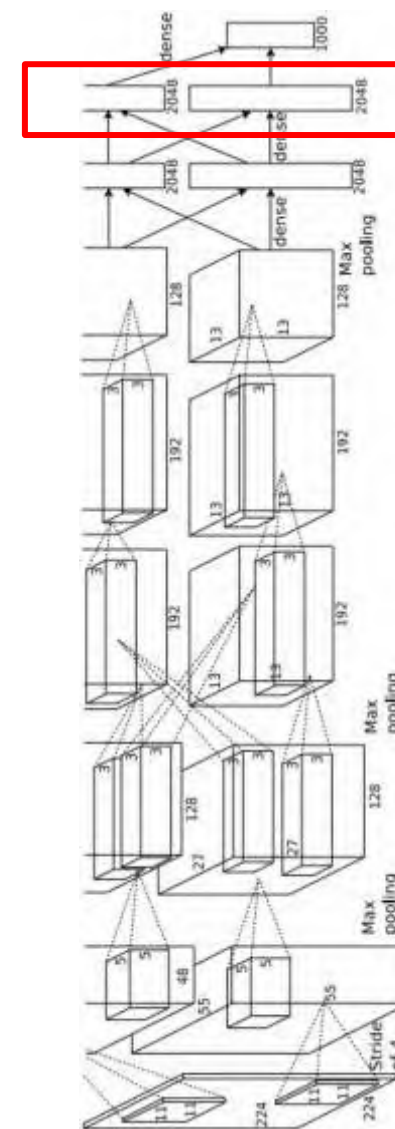
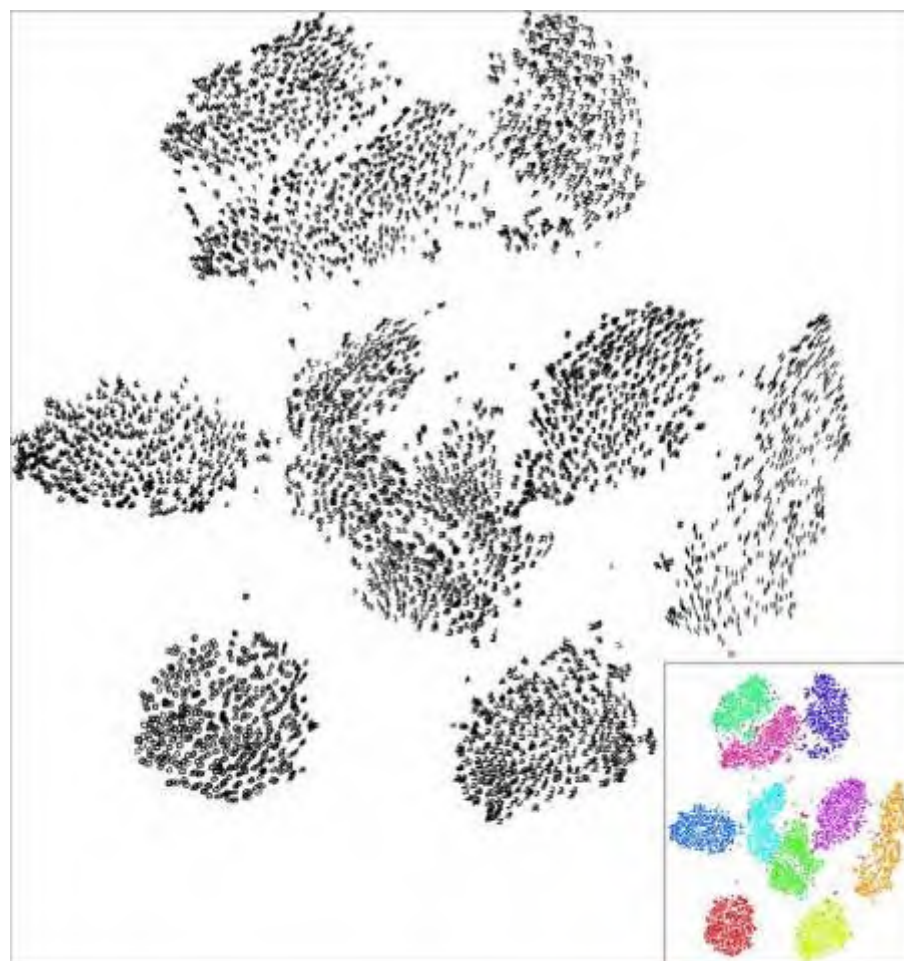
Recall: Nearest neighbors in pixel space



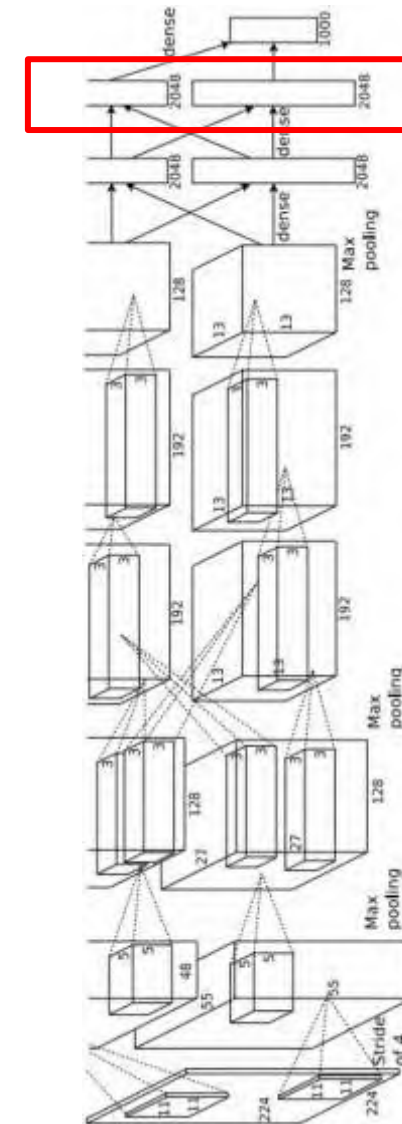
维度压缩

t-SNE:

把最后一层4096维的数据压缩成2维



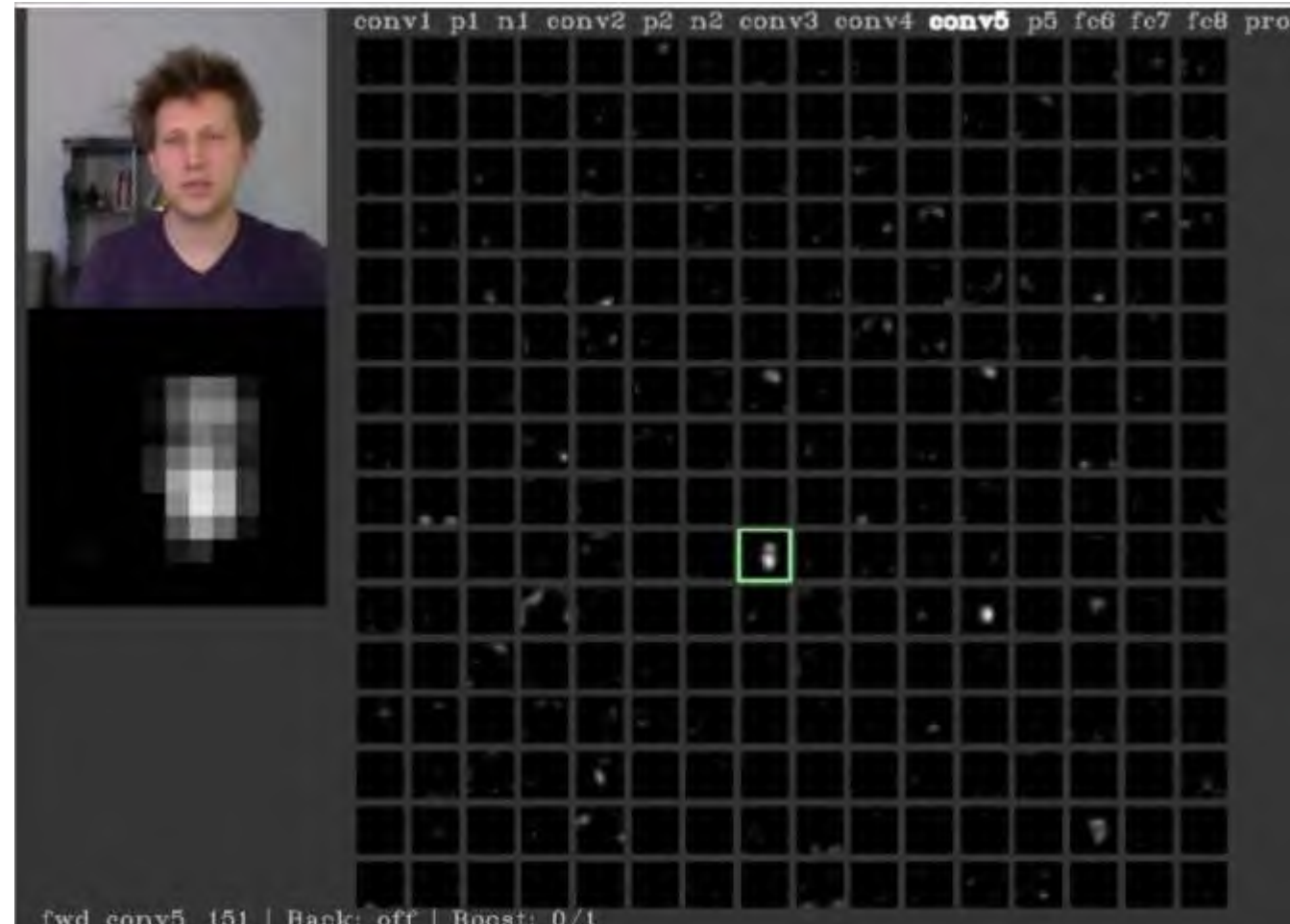
维度压缩



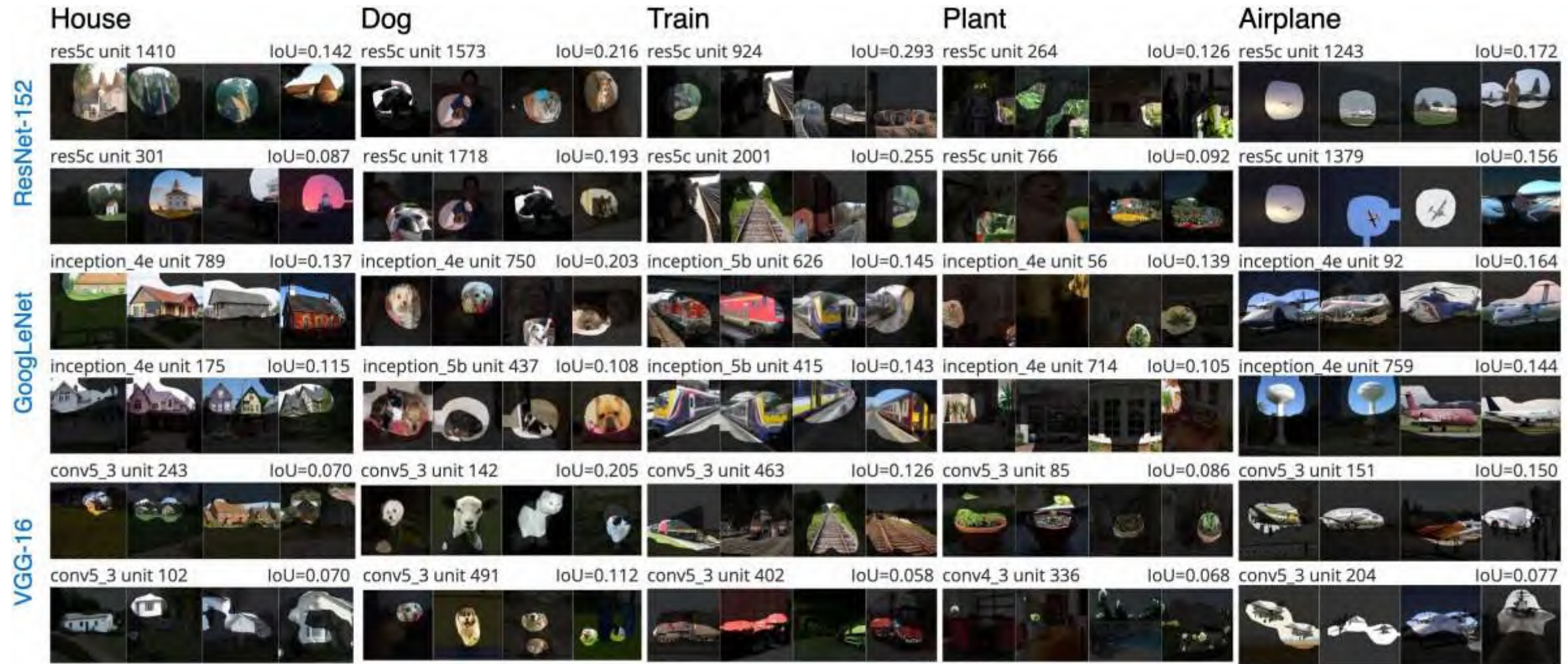
See high-resolution versions at <http://cs.stanford.edu/people/karpathy/cnnembed/>

可视化 激活值

conv5 feature map
is 128x13x13;
visualize as 128
13x13 grayscale
images

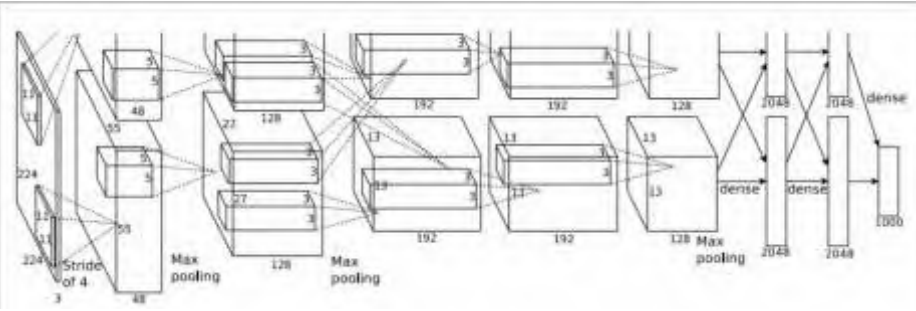


可视化激活值

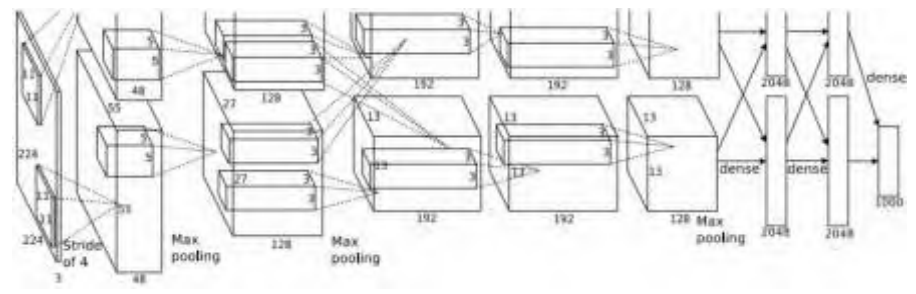
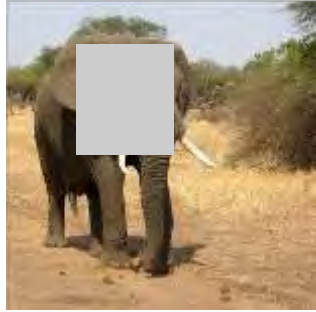


显著性可视化

盖住图像的某一个部分，再预测类别



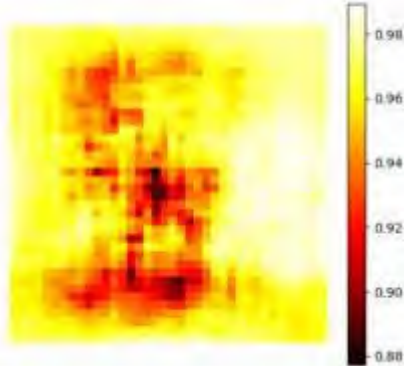
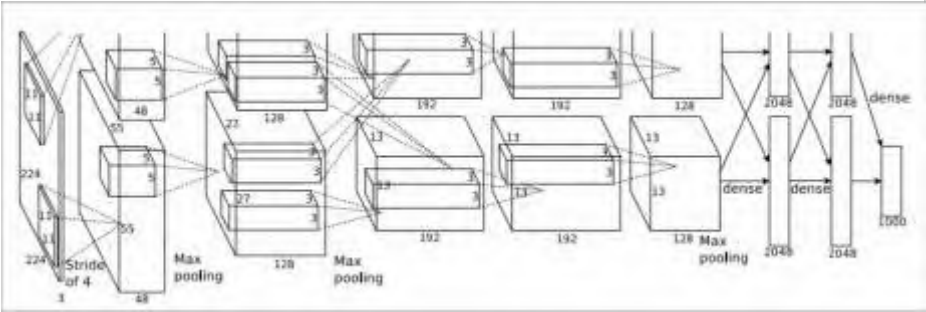
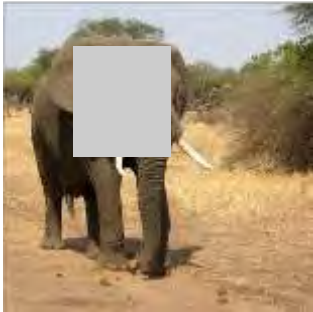
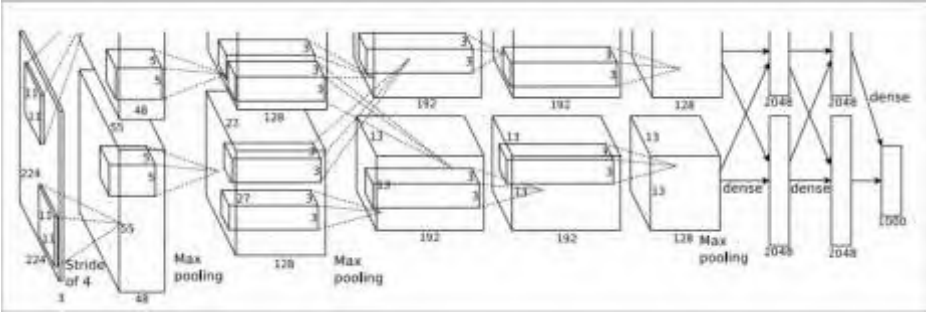
$P(\text{elephant}) = 0.95$



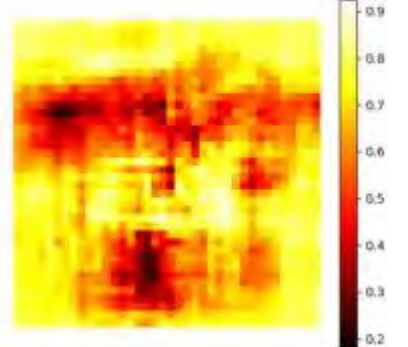
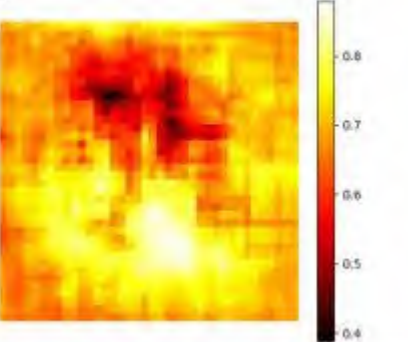
$P(\text{elephant}) = 0.75$

显著性可视化

盖住图像的某一个部分，再预测类别

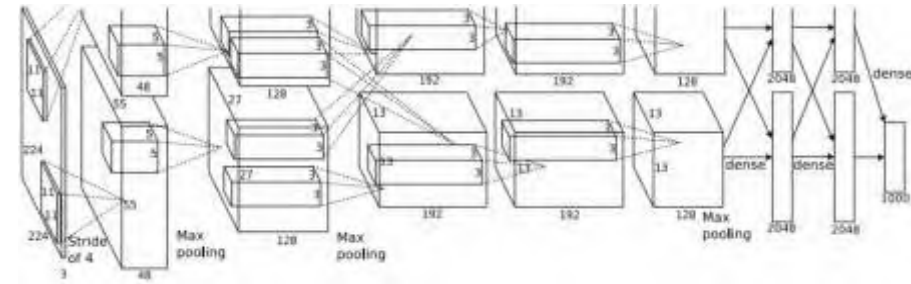


African elephant, *Loxodonta africana*



显著性可视化

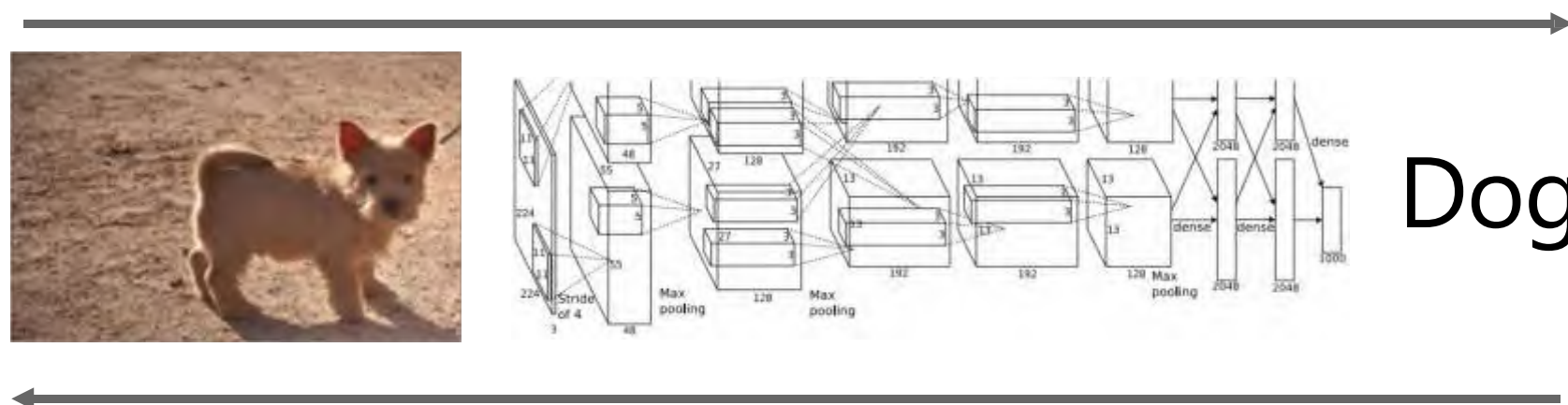
Forward pass: Compute probabilities



Dog

显著性可视化

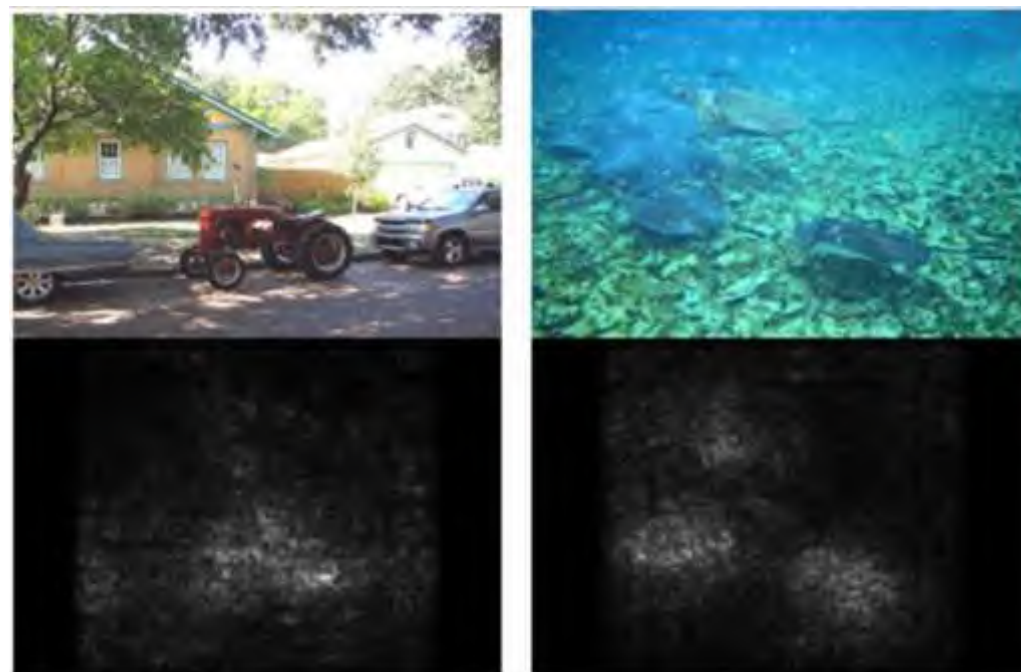
Forward pass: Compute probabilities



根据梯度来计算像素点上的响应值

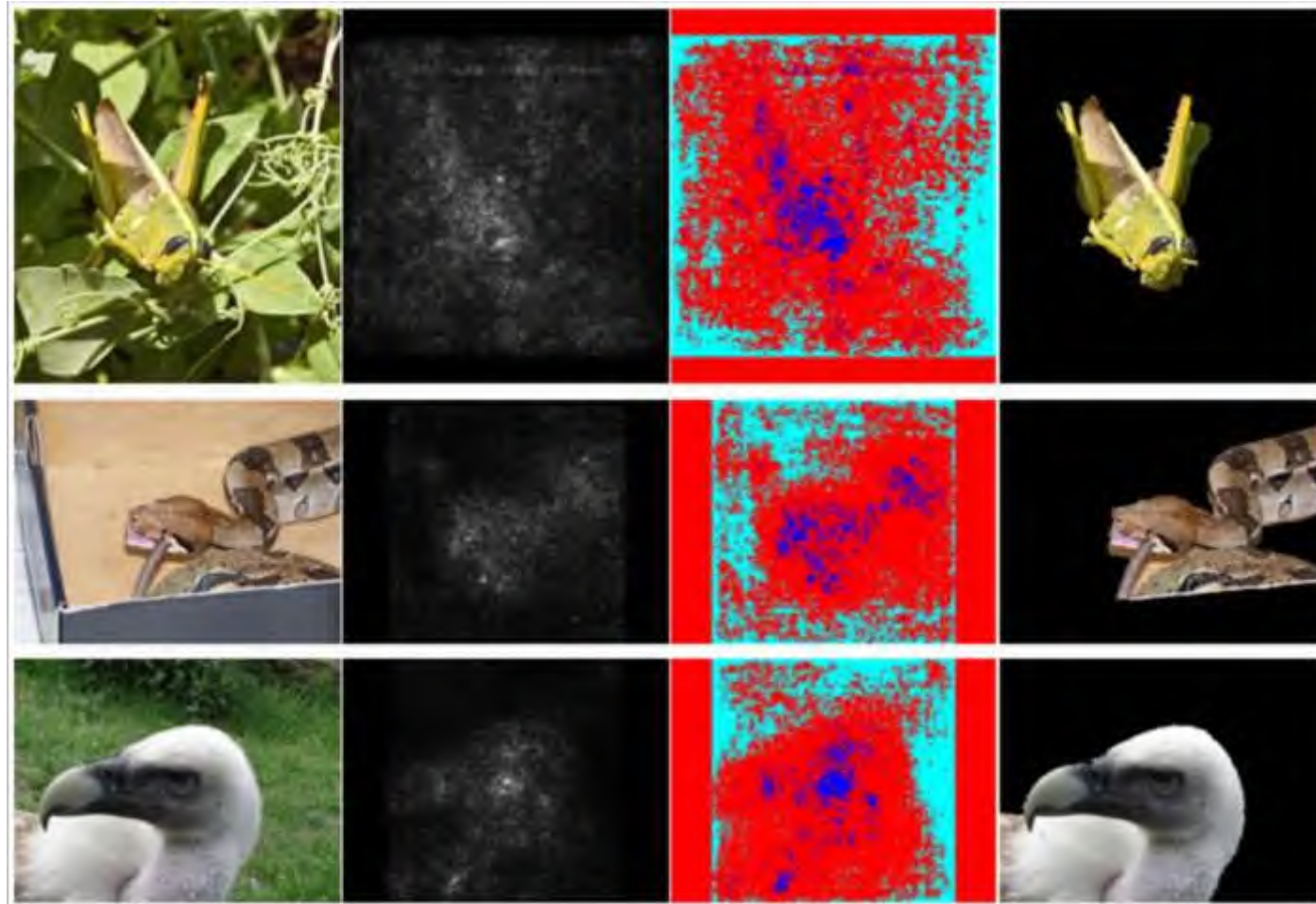


显著性可视化



显著性可视化: 无监督分割

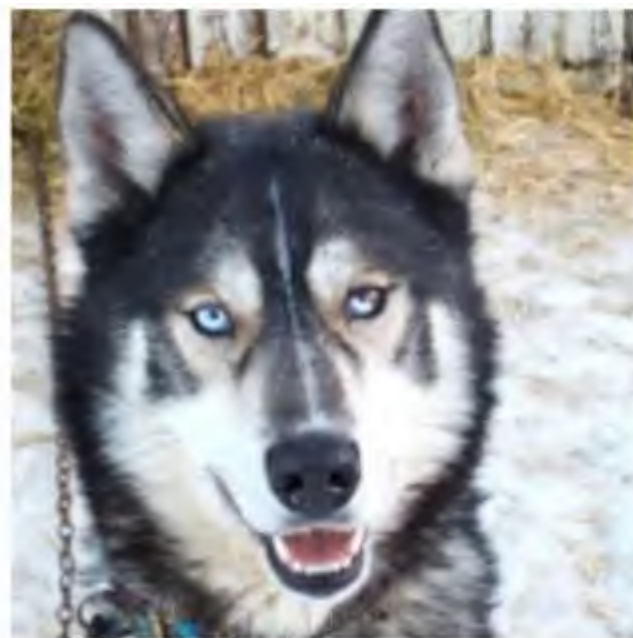
在显著性图上使用
无监督分割



显著性可视化: Uncovers biases

找到分类错误的原因

由于背景有雪所以哈士奇分成了狼



(a) Husky classified as wolf



(b) Explanation

可视化 CNN 特征: 梯度上升

梯度上升: 生成一张图, 最大化某个神经元的激活值

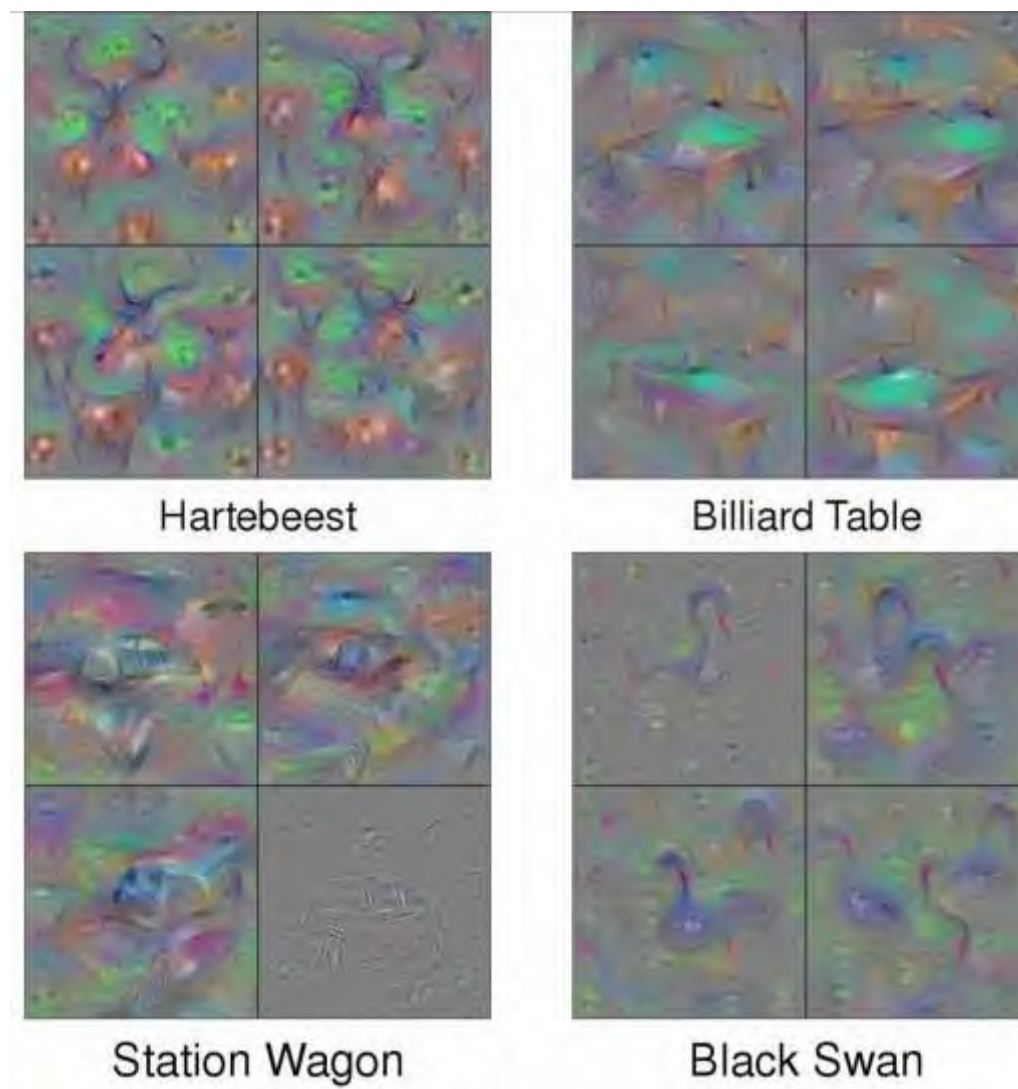
$$\arg \max_I S_c(I) - \lambda \|I\|_2^2$$

Neuron value

Natural image regularizer

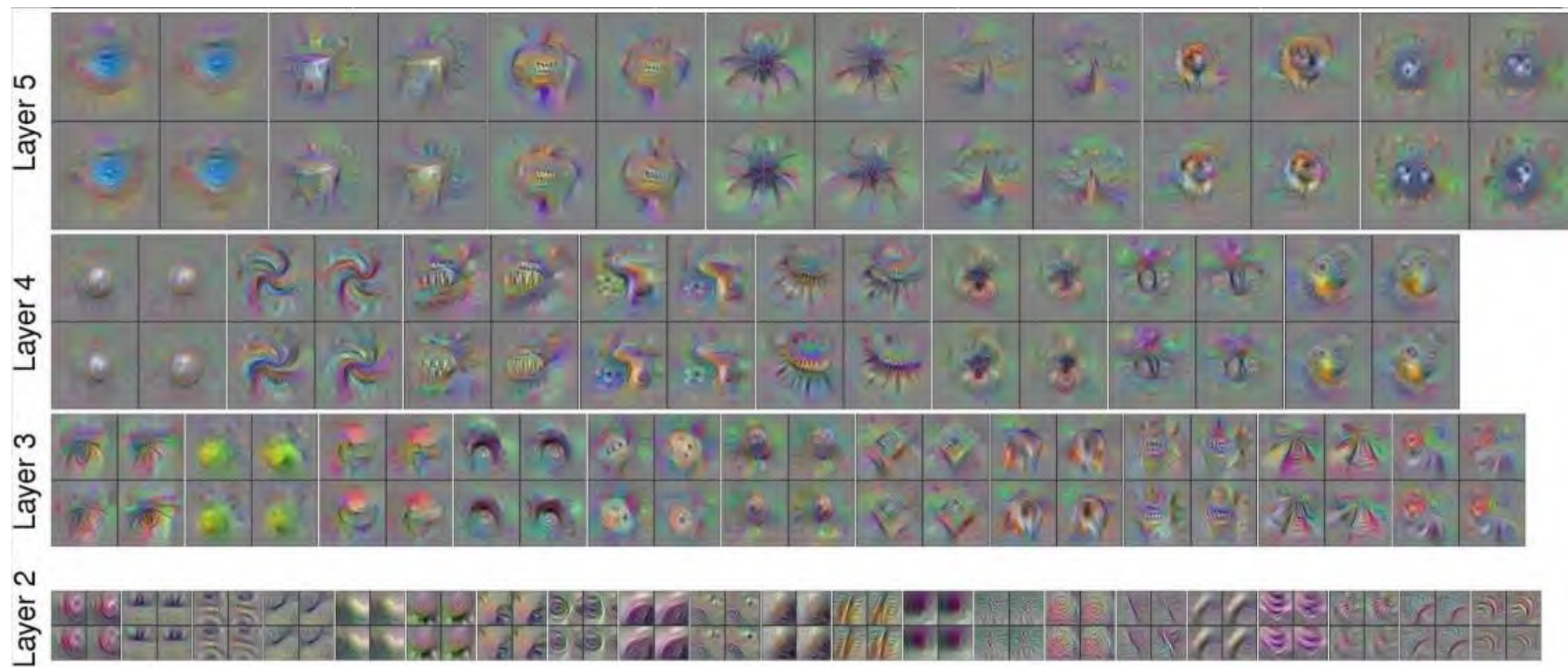
可视化 CNN 特征: 梯度上升

$$\arg \max_I S_c(I) - \lambda \|I\|_2^2$$



Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.
Figure copyright Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson, 2014. Reproduced with permission.

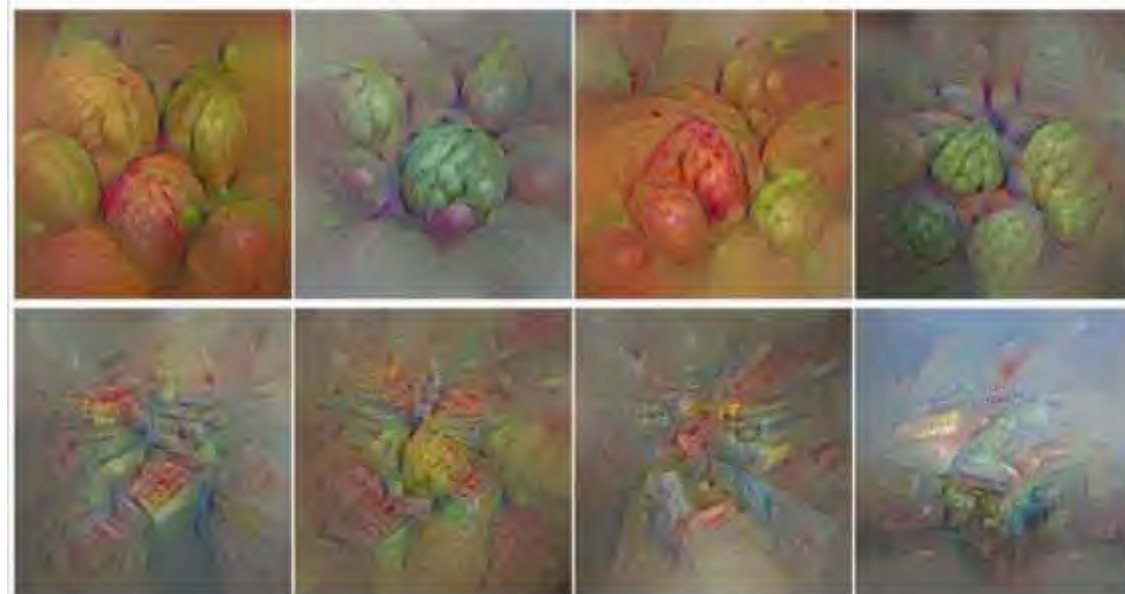
可视化 CNN 特征: 梯度上升



Yosinski et al, "Understanding Neural Networks Through Deep Visualization", ICML DL Workshop 2014.
Figure copyright Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson, 2014. Reproduced with permission.

可视化 CNN 特征: 梯度上升

Reconstructions of multiple feature types (facets) recognized by the same "grocery store" neuron



Corresponding example training set images recognized by the same neuron as in the "grocery store" class



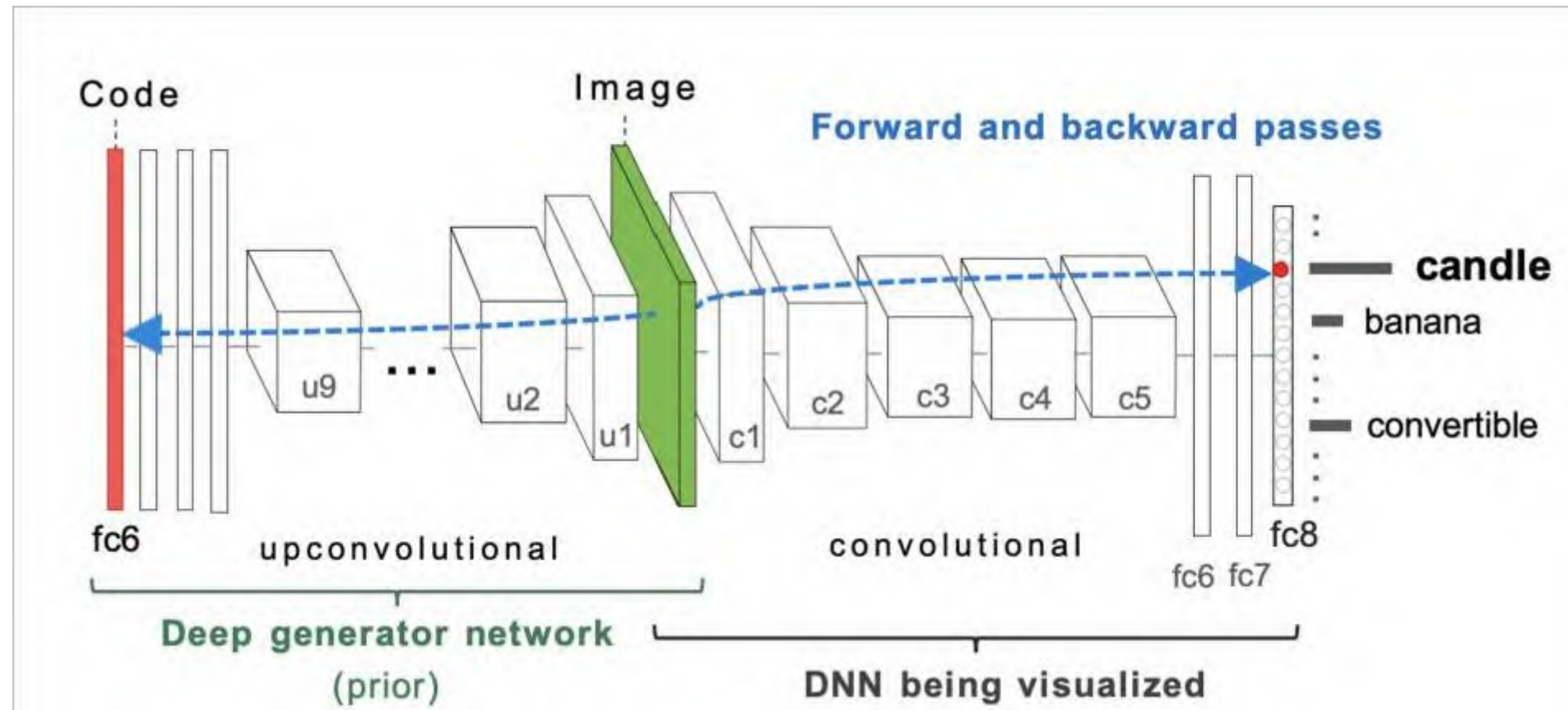
Nguyen et al, "Multifaceted Feature Visualization: Uncovering the Different Types of 特征 Learned By Each Neuron in Deep Neural Networks", ICML Visualization for Deep Learning Workshop 2016.
Figures copyright Anh Nguyen, Jason Yosinski, and Jeff Clune, 2016; reproduced with permission.

可视化 CNN 特征: 梯度上升



Nguyen et al, "Multifaceted Feature Visualization: Uncovering the Different Types of 特征 Learned By Each Neuron in Deep Neural Networks", ICML Visualization for Deep Learning Workshop 2016.
Figures copyright Anh Nguyen, Jason Yosinski, and Jeff Clune, 2016; reproduced with permission.

可视化 CNN 特征: 梯度上升



可视化 CNN 特征: 梯度上升

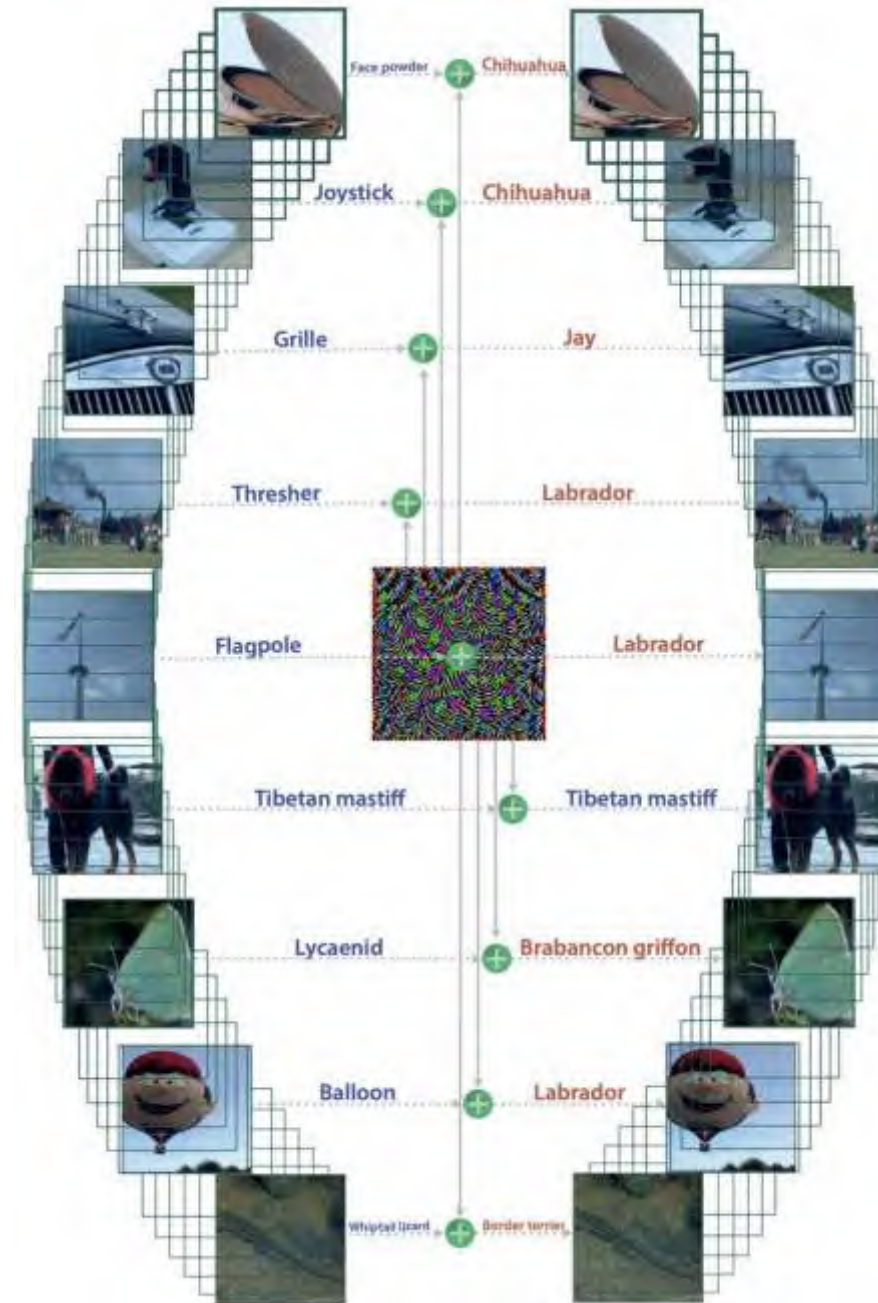


Nguyen et al, "Synthesizing the preferred inputs for neurons in neural networks via deep generator networks," NIPS 2016
Figure copyright Nguyen et al, 2016; reproduced with permission.

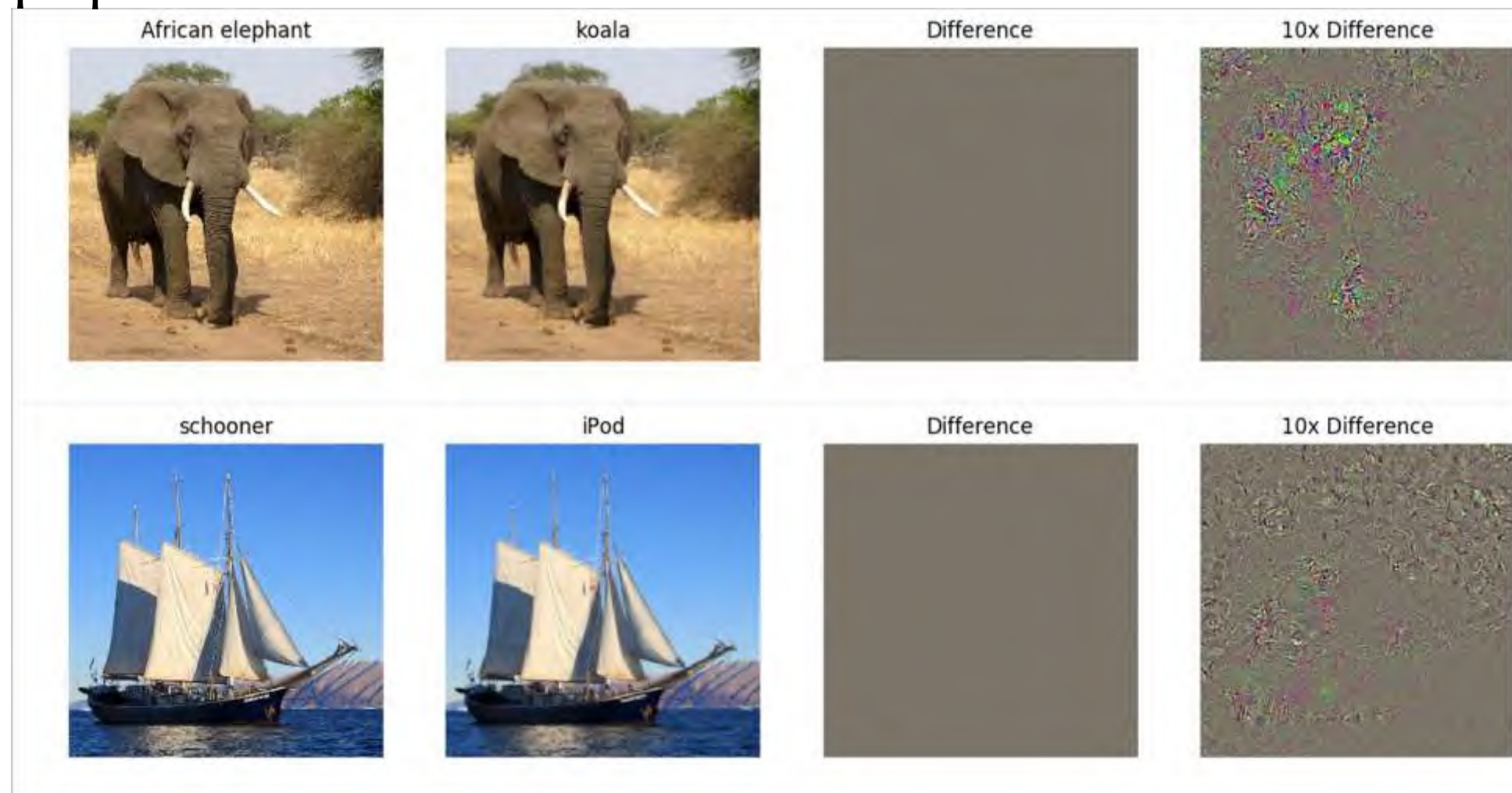
对抗样本

对图像施加小的扰动来
让CNN分类错误

Moosavi-Dezfooli, Seyed-Mohsen, et al. "Universal adversarial perturbations." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
Figure reproduced with permission



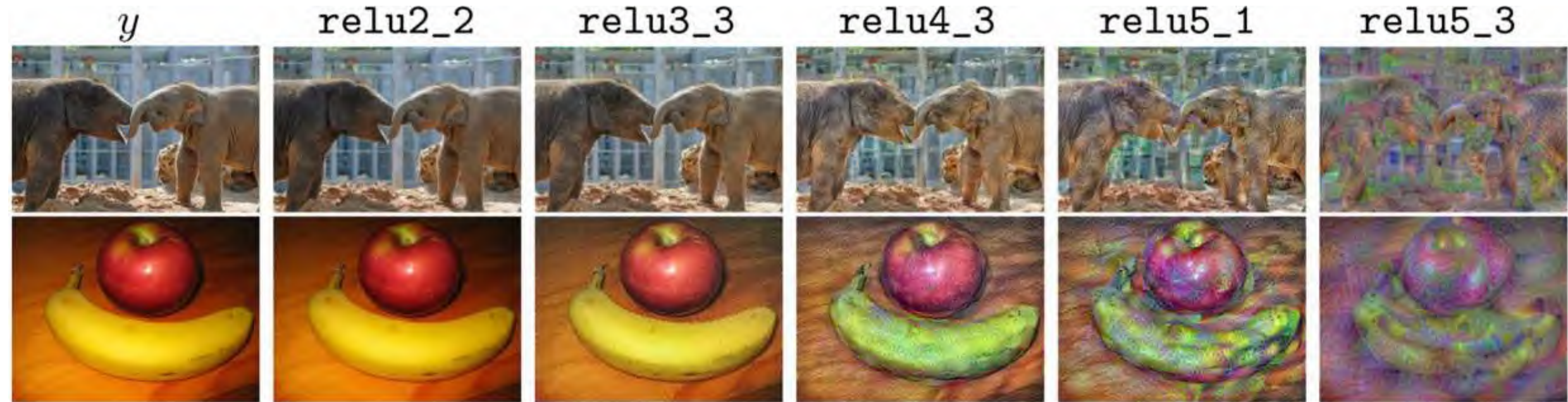
对抗样本



Boat image is [CC0 public domain](#)
Elephant image is [CC0 public domain](#)

逆特征

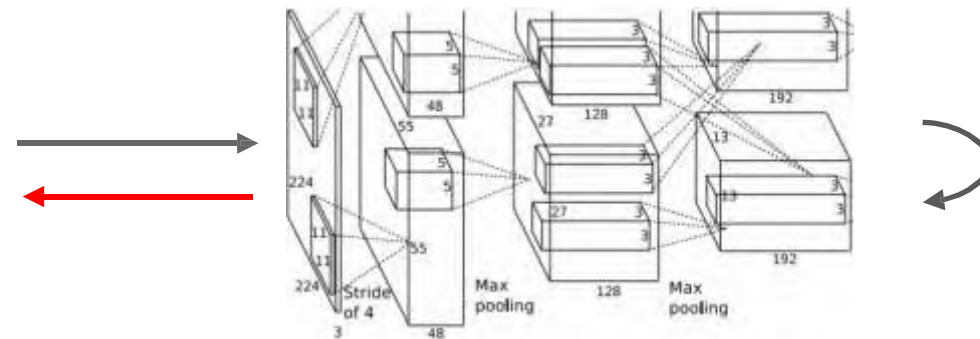
让网络输出接近某个特征，并看起来像图像



Mahendran and Vedaldi, "Understanding Deep Image Representations by Inverting Them", CVPR 2015
Figure from Johnson, Alahi, and Fei-Fei, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution", ECCV 2016. Copyright Springer, 2016.
Reproduced for educational purposes.

DeepDream: 强化特征

与其生成图像来最大化神经元相应，不如直接增强一些层中神经元的响应



Mordvintsev, Olah, and Tyka, "Inceptionism: Going Deeper into Neural Networks", [Google Research Blog](#). Images are licensed under [CC-BY 4.0](#)



[Sky image](#) is licensed under [CC-BY SA 3.0](#)



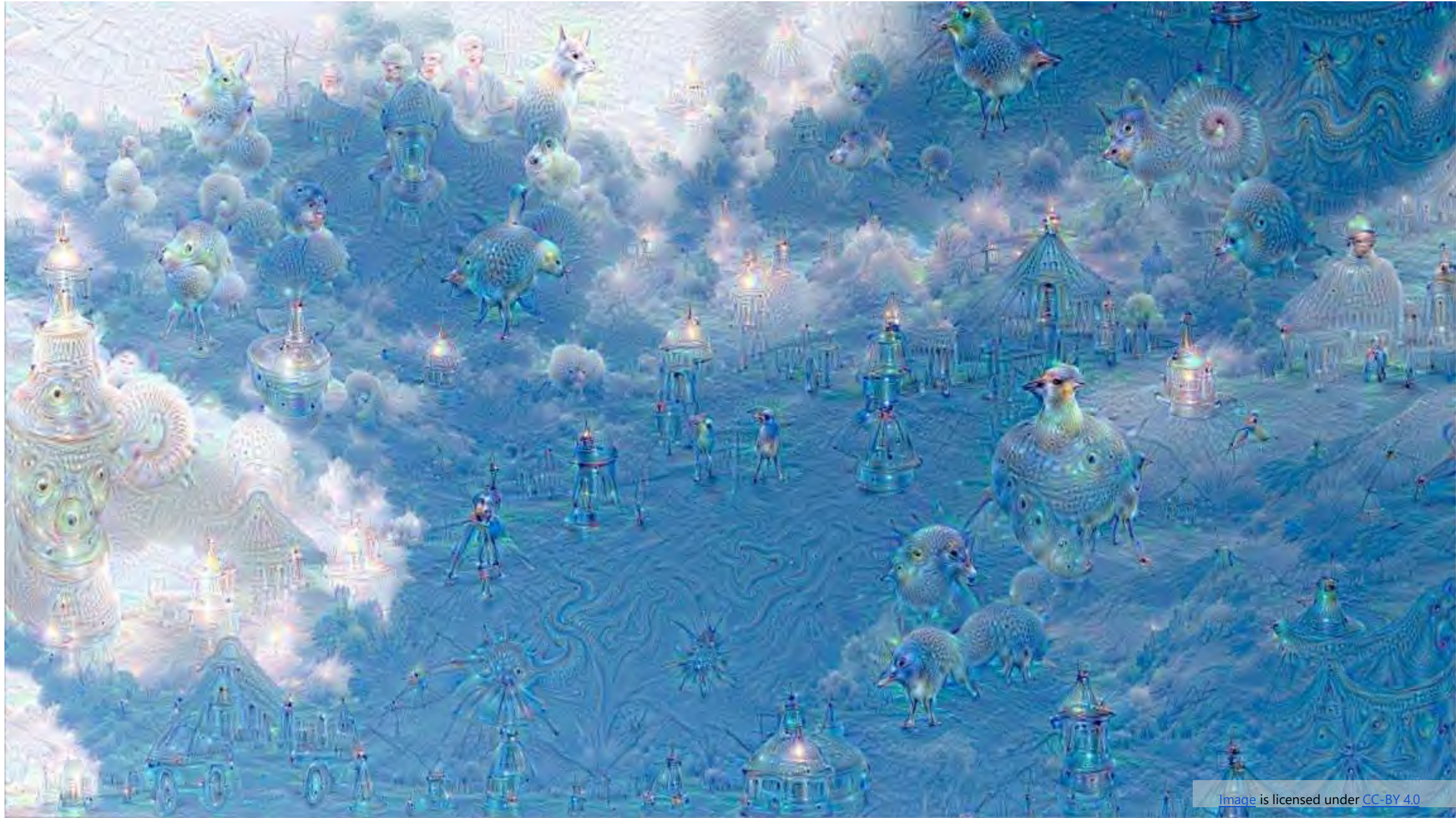
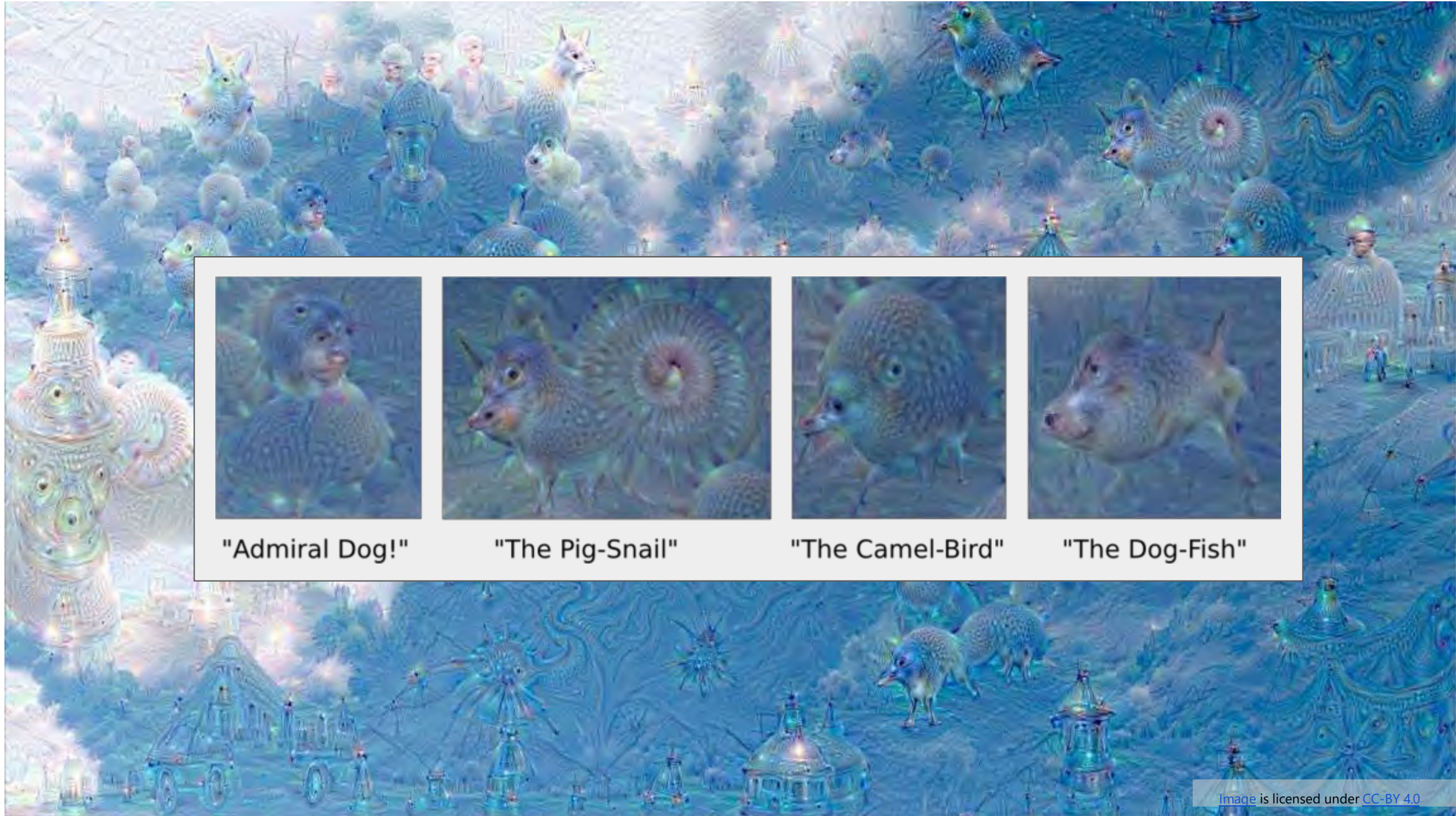


Image is licensed under [CC-BY 4.0](https://creativecommons.org/licenses/by/4.0/)



"Admiral Dog!"



"The Pig-Snail"



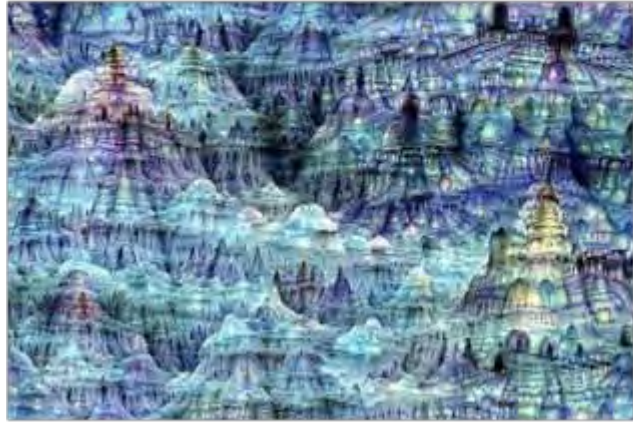
"The Camel-Bird"



"The Dog-Fish"

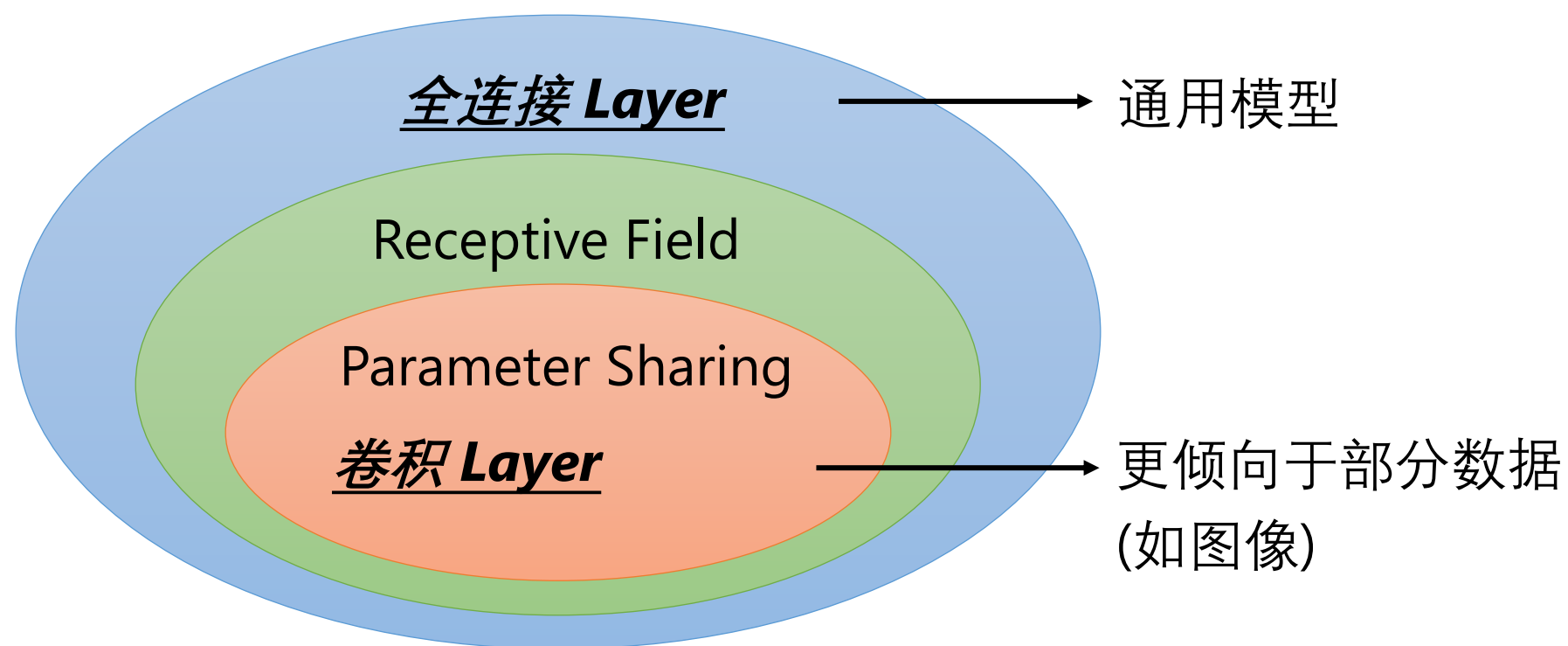


Image is licensed under [CC-BY 3.0](https://creativecommons.org/licenses/by/3.0/)



[Image](#) is licensed under [CC-BY 4.0](#)

卷积层



- 一些模式只出现在小的区域
- 相同的模式会出现在不同的区域

卷积层

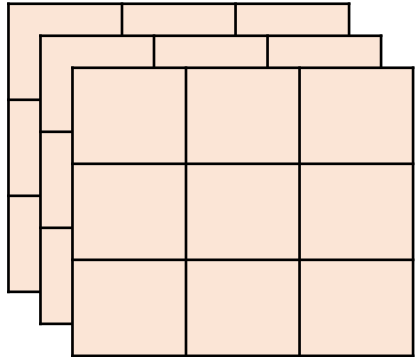


卷积

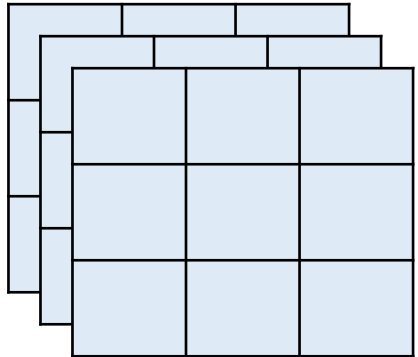


⋮ ⋮

channel = 3 (colorful)
channel = 1 (black and white)



Filter 1
3 x 3 x
channel



Filter 2
3 x 3 x
channel

⋮ ⋮

Each filter detects a small pattern (3 x 3 x channel).

卷积层

Consider channel = 1
(black and white
image)

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

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

Filter 1

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

Filter 2

⋮ ⋮

(The values in the filters
are unknown
parameters.)

卷积层

stride=1

1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

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

Filter 1

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

卷积层

stride=1

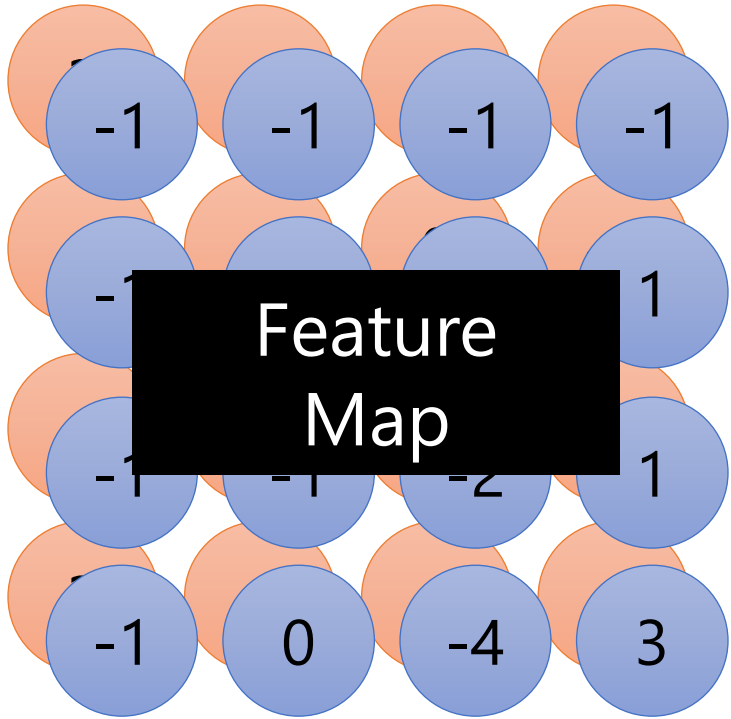
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

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

Filter 2

Do the same process for every filter



卷积层

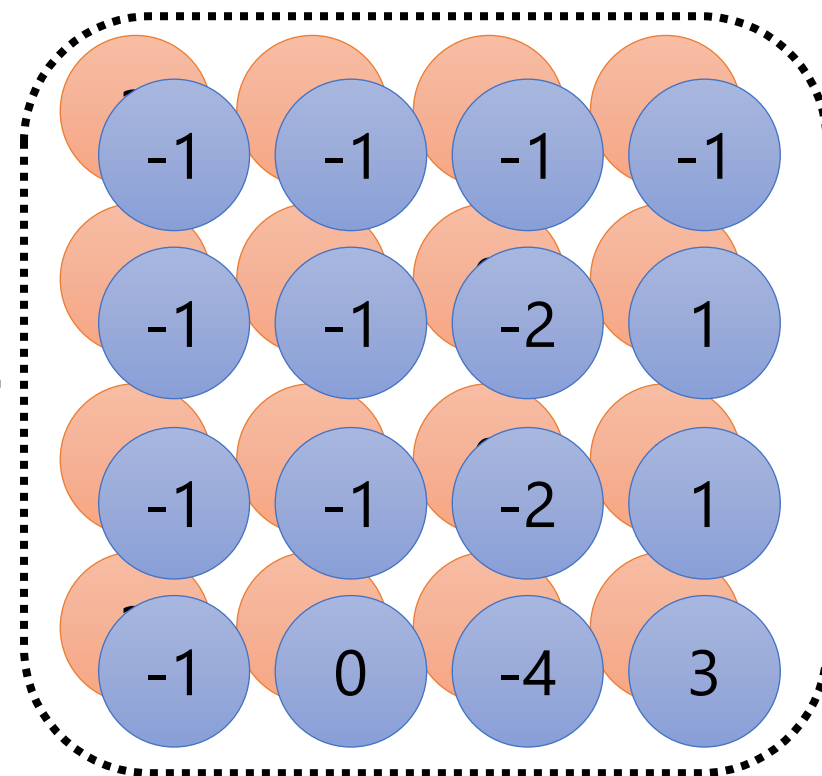


64
filters

卷积

卷积

⋮
⋮



"Image" with 64 channels

多个卷积层

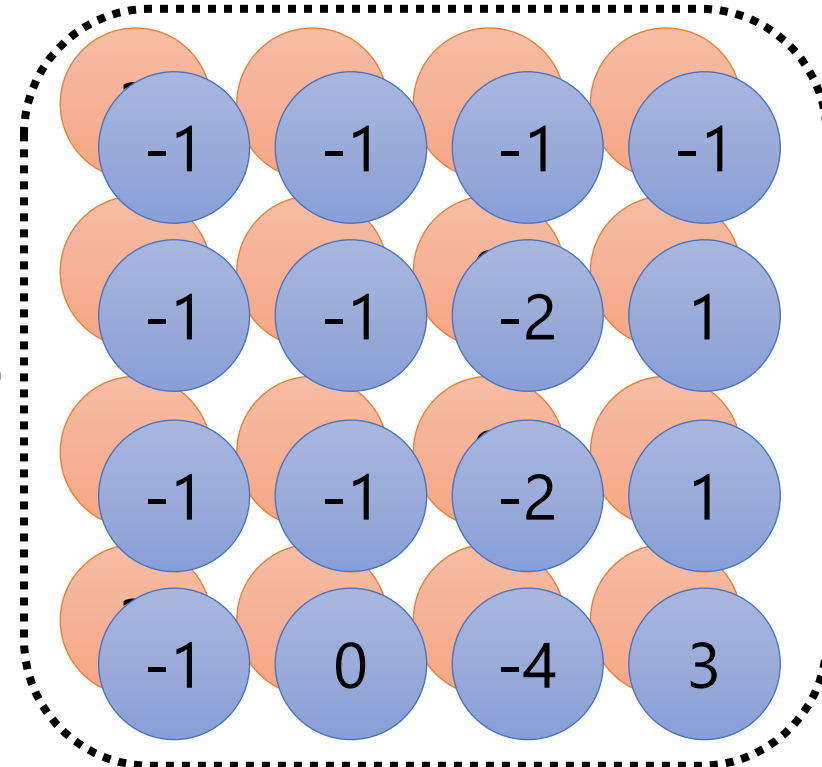


64
filters

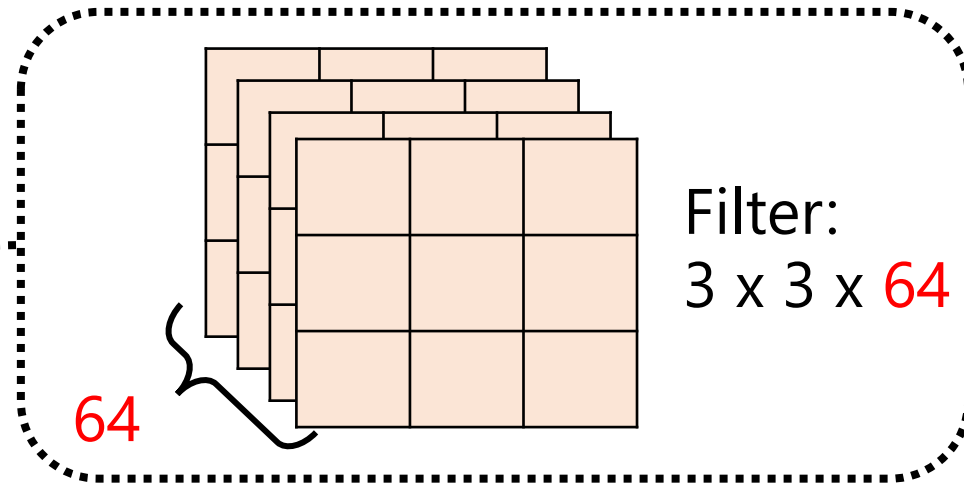
卷积

卷积

⋮
⋮



"Image" with 64 channels



多个卷积层



64
filters

卷积

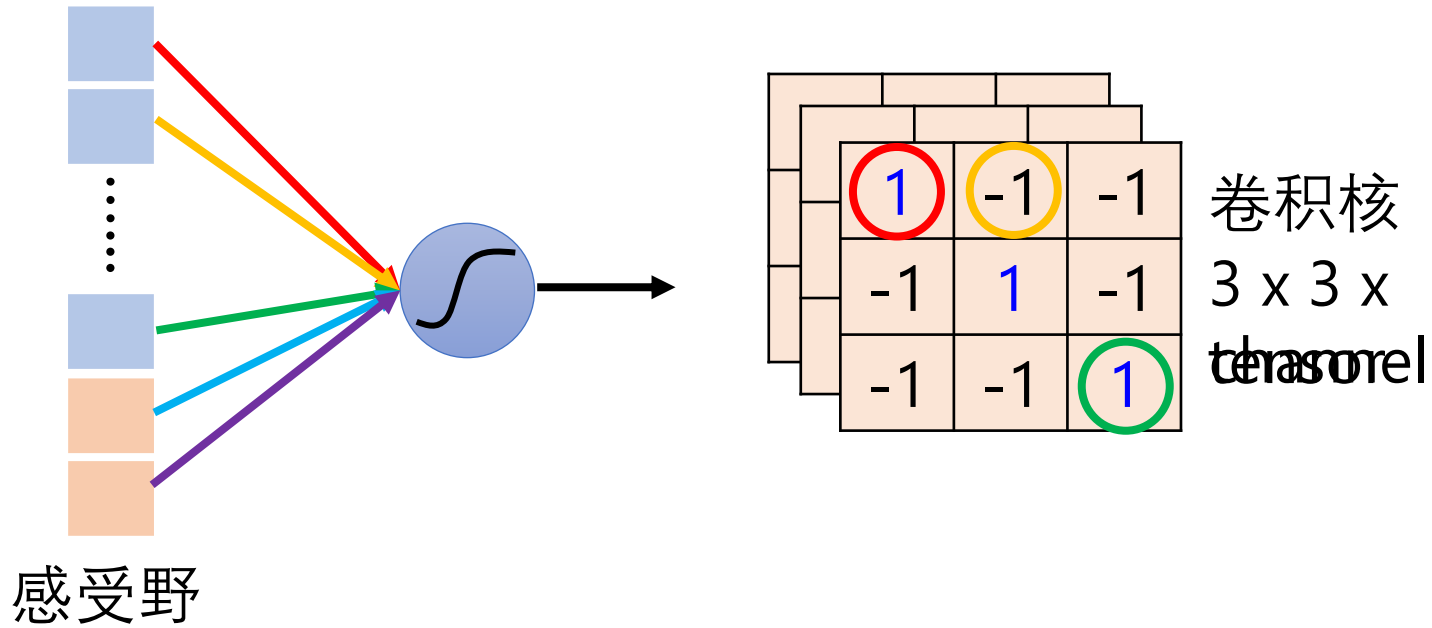
卷积

⋮
⋮

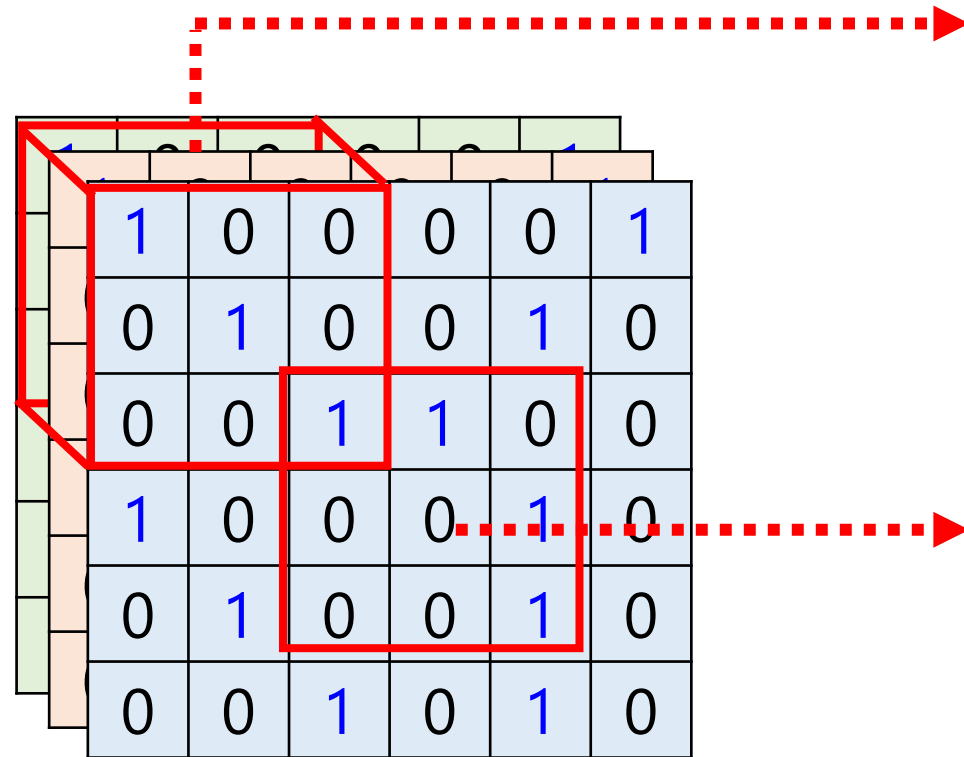
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

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

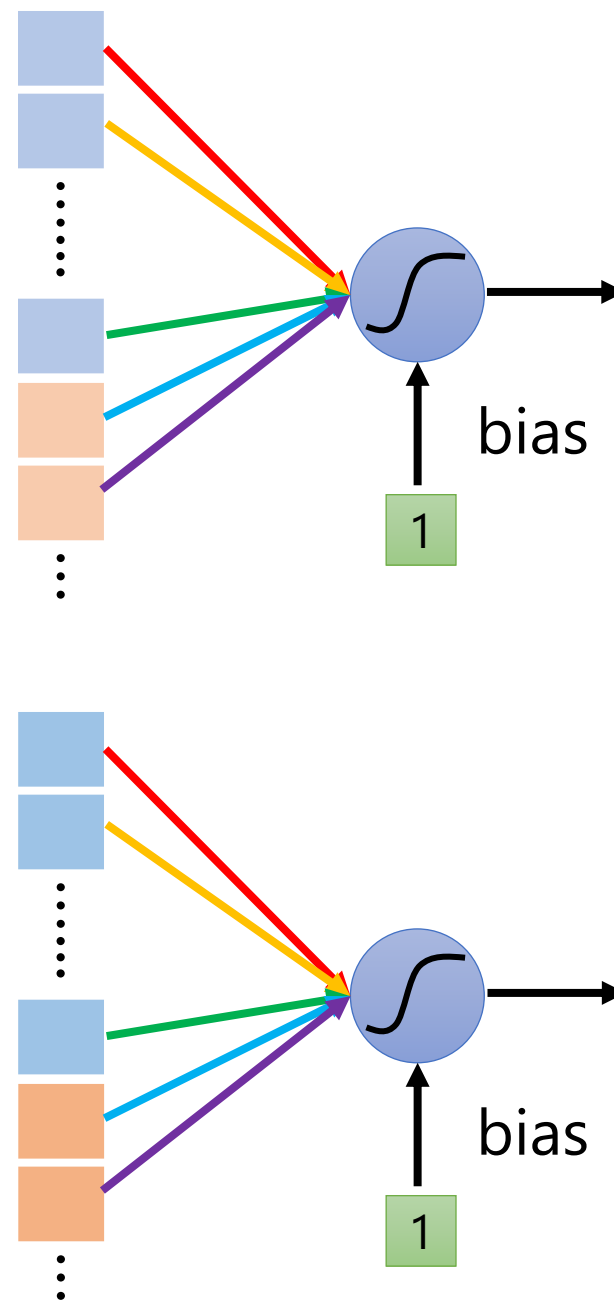
卷积层



不同感受野的神经元共享网络参数

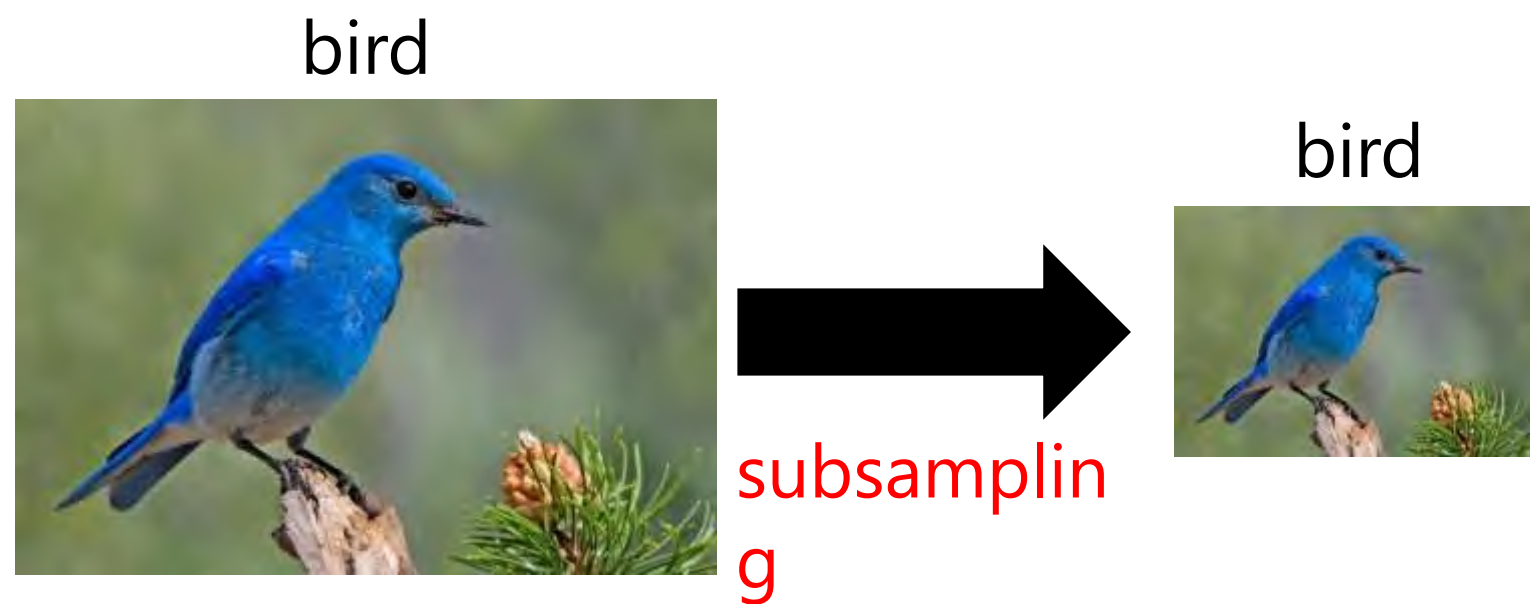


每个卷积核对所有窗口做卷积



卷积层

- 采样图像（缩放）不会改变内容



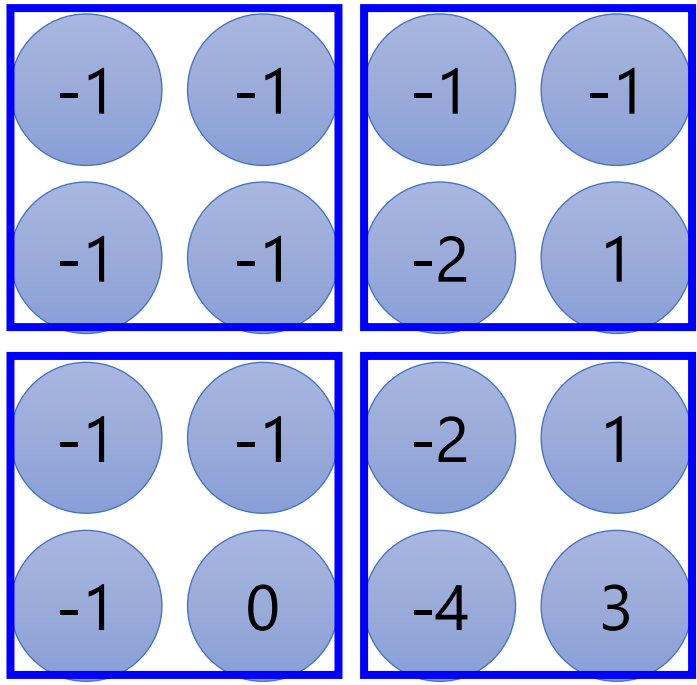
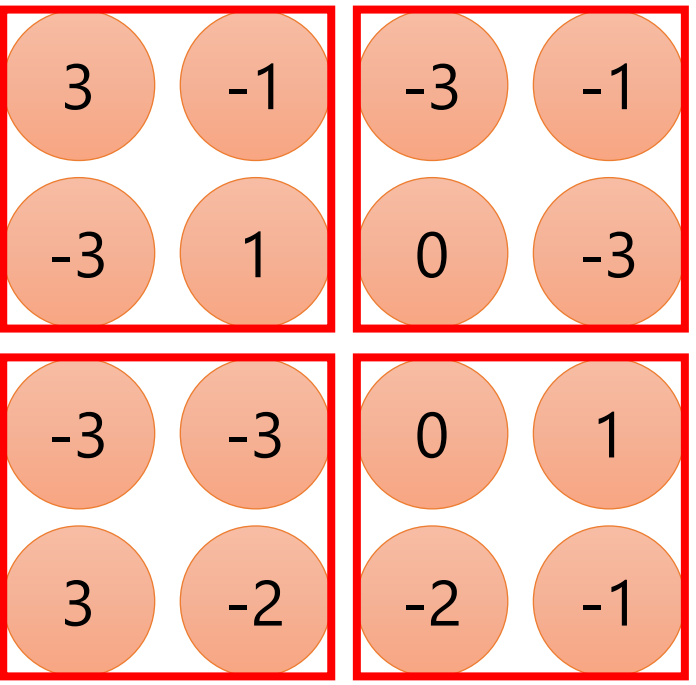
Pooling – Max Pooling

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

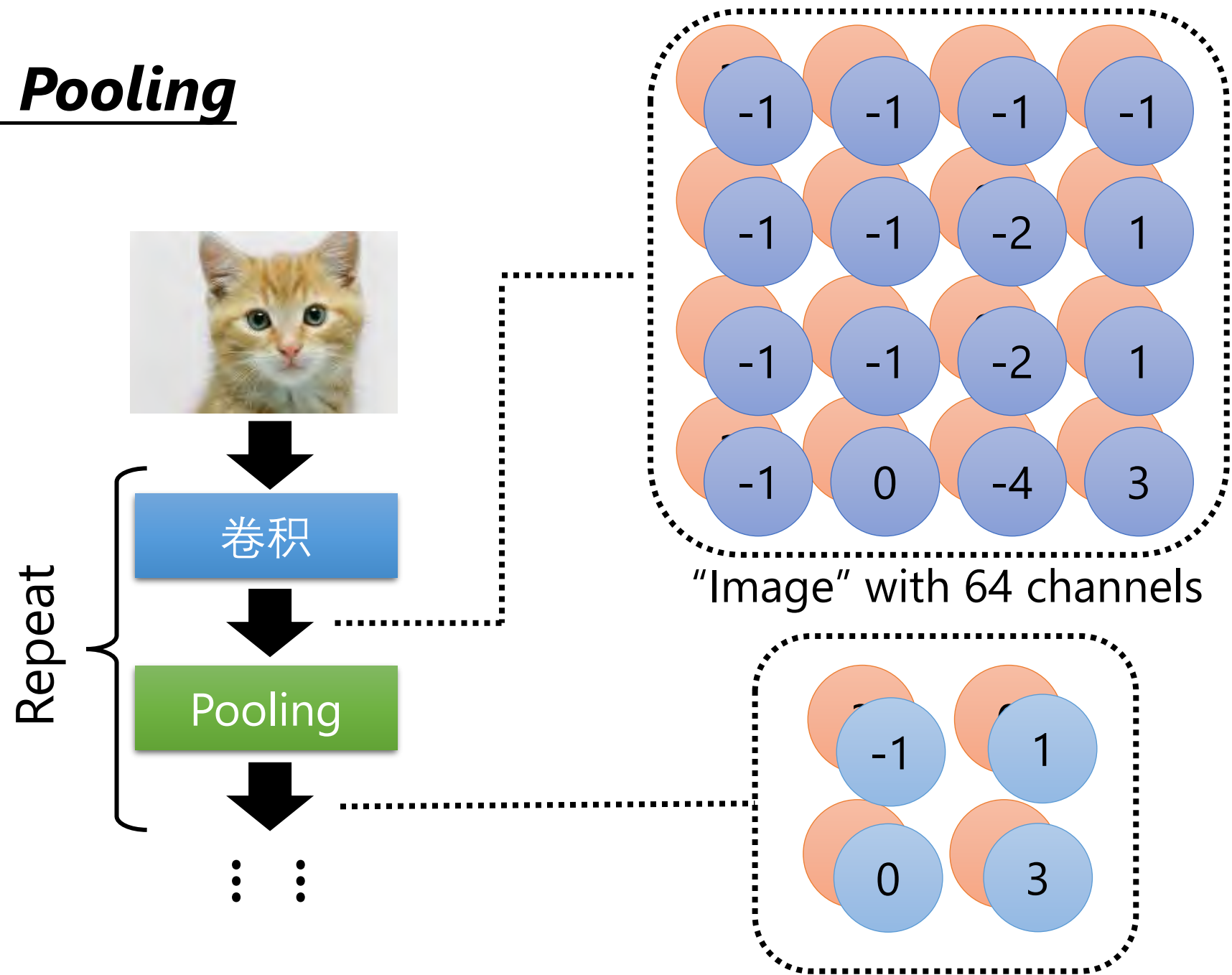
Filter 1

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

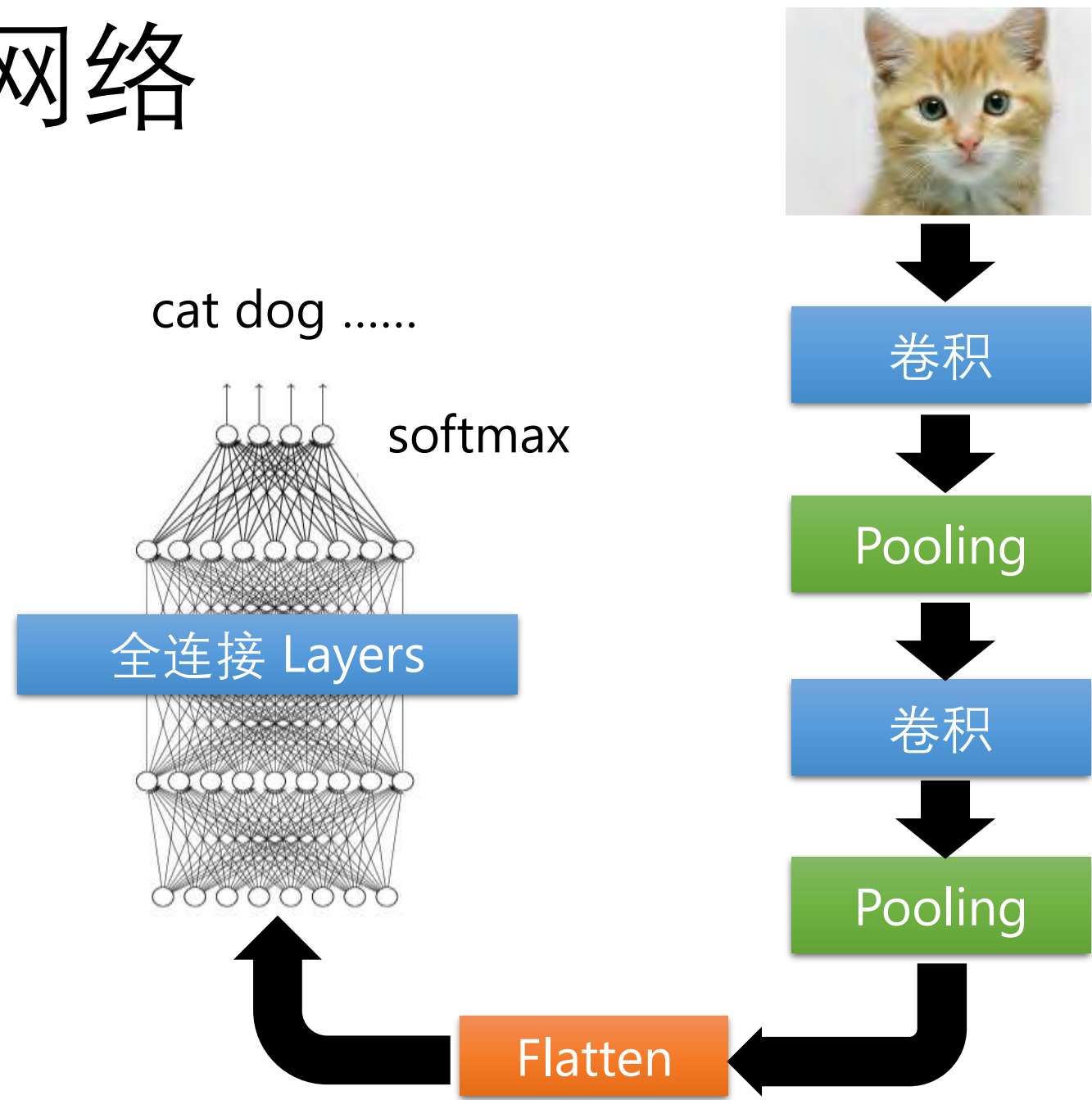
Filter 2



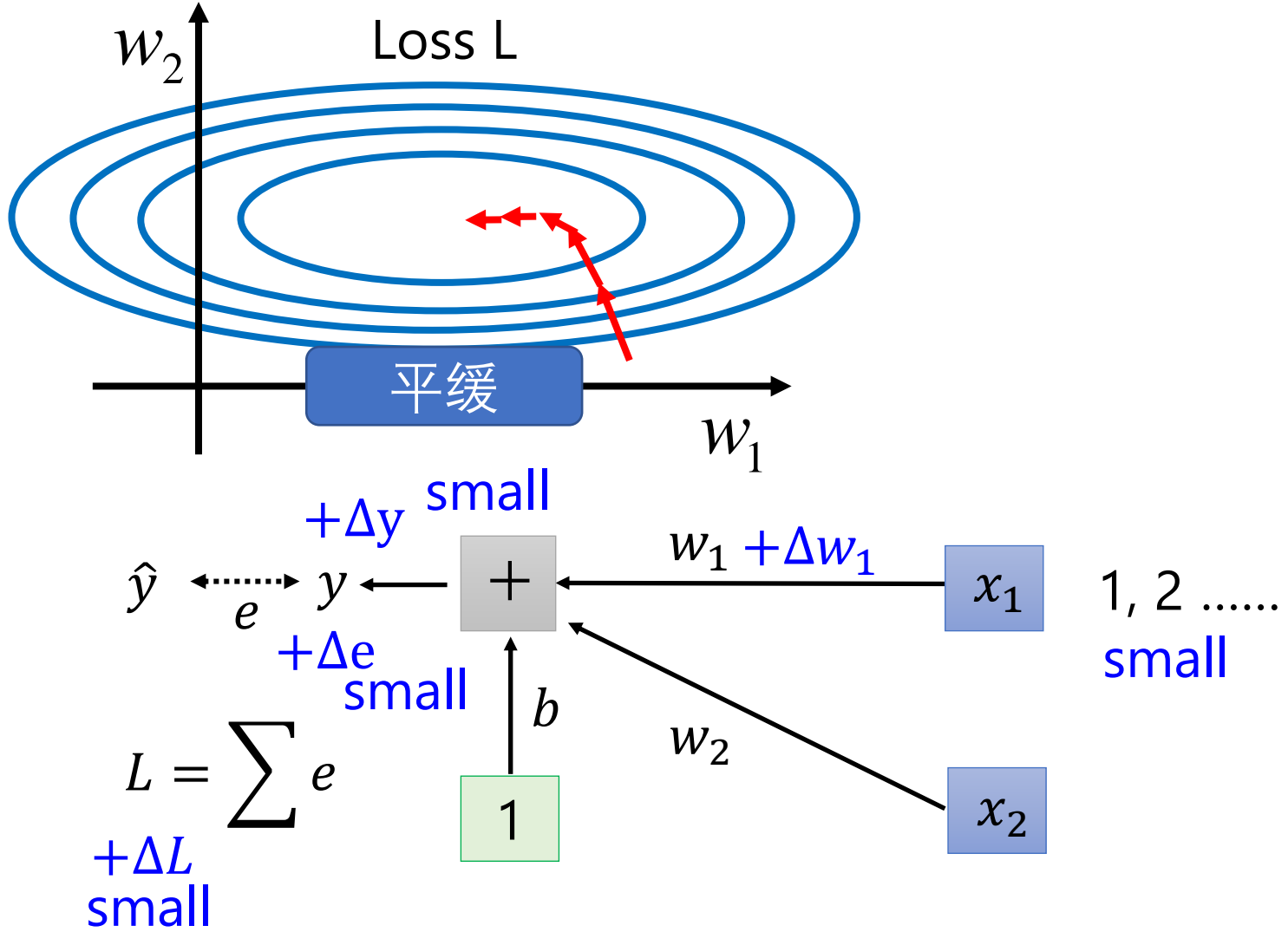
卷积层+ Pooling



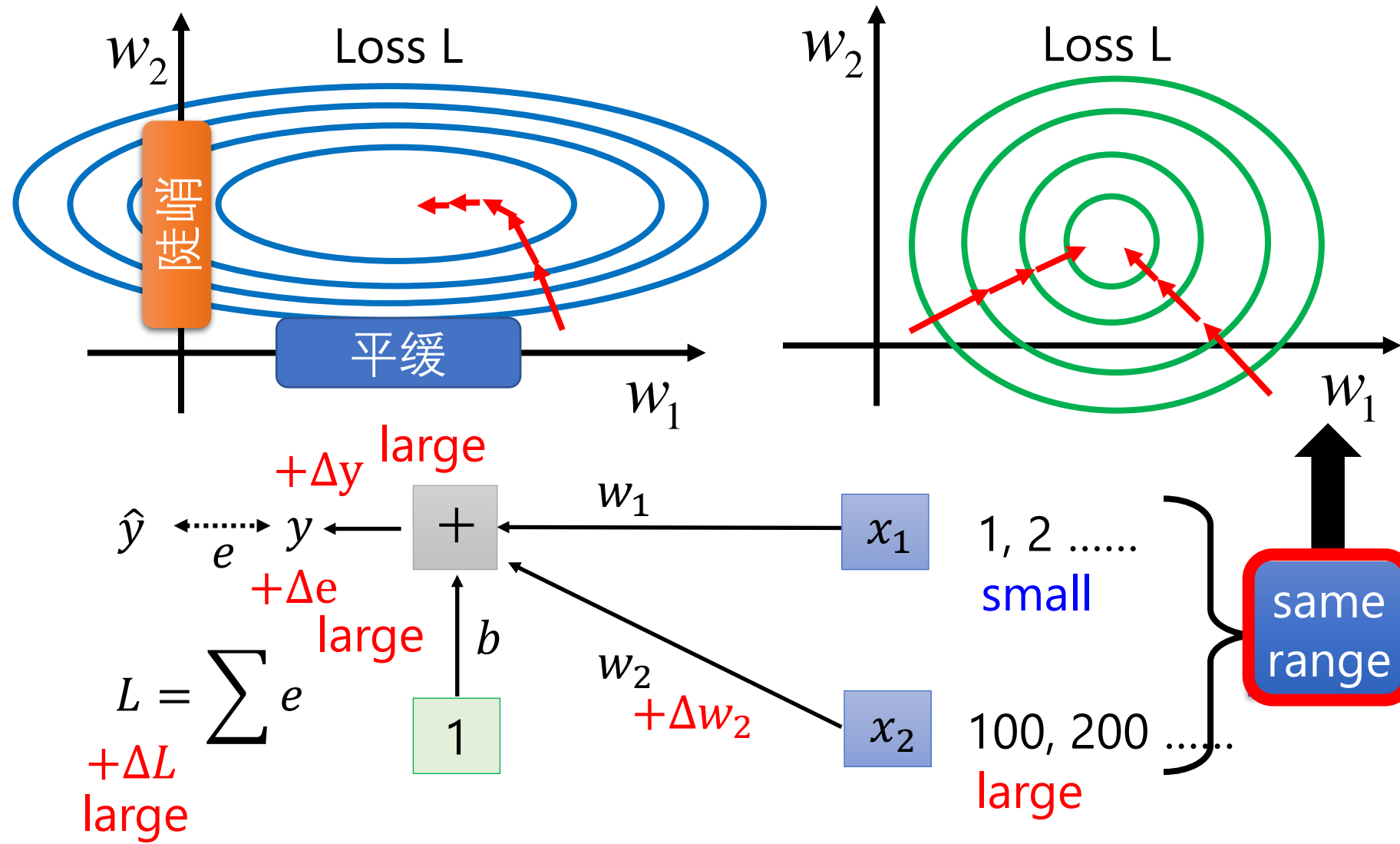
整个CNN网络



深度网络优化

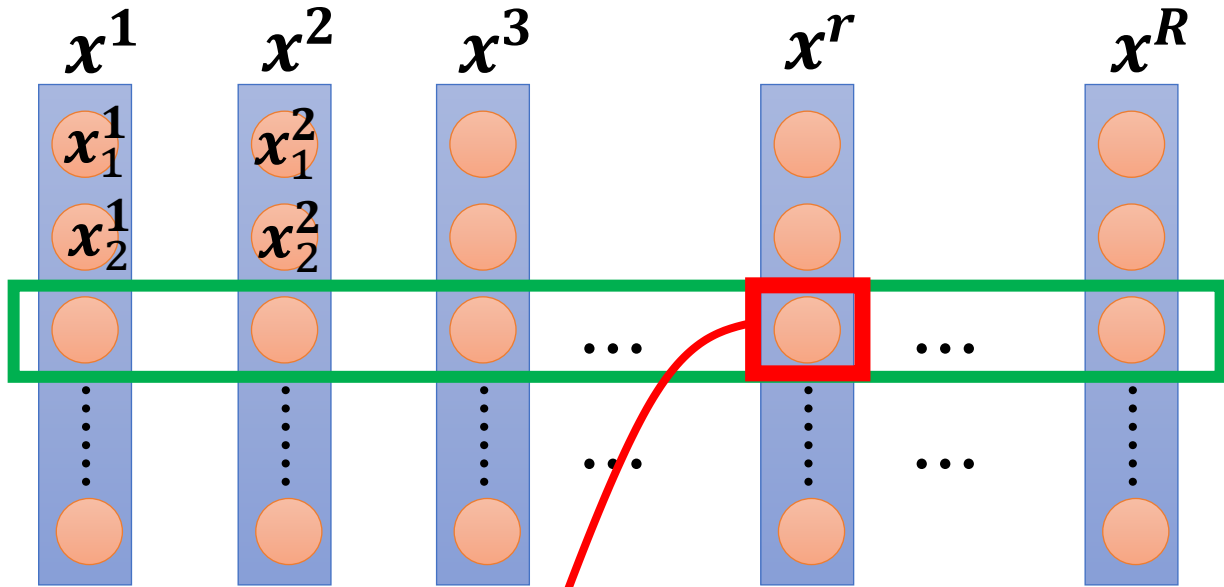


深度网络优化



提升训练速度!

特征标准化

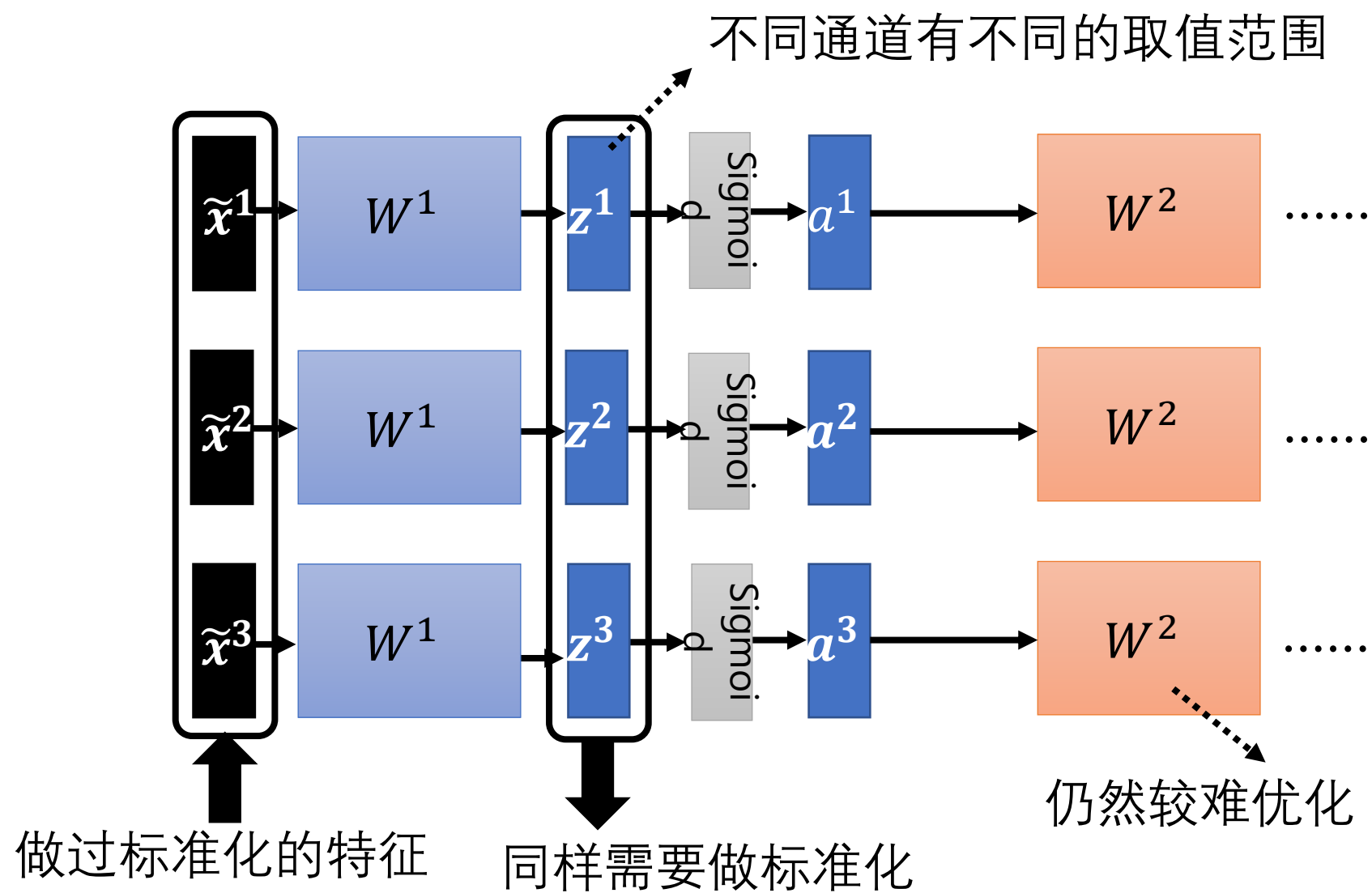


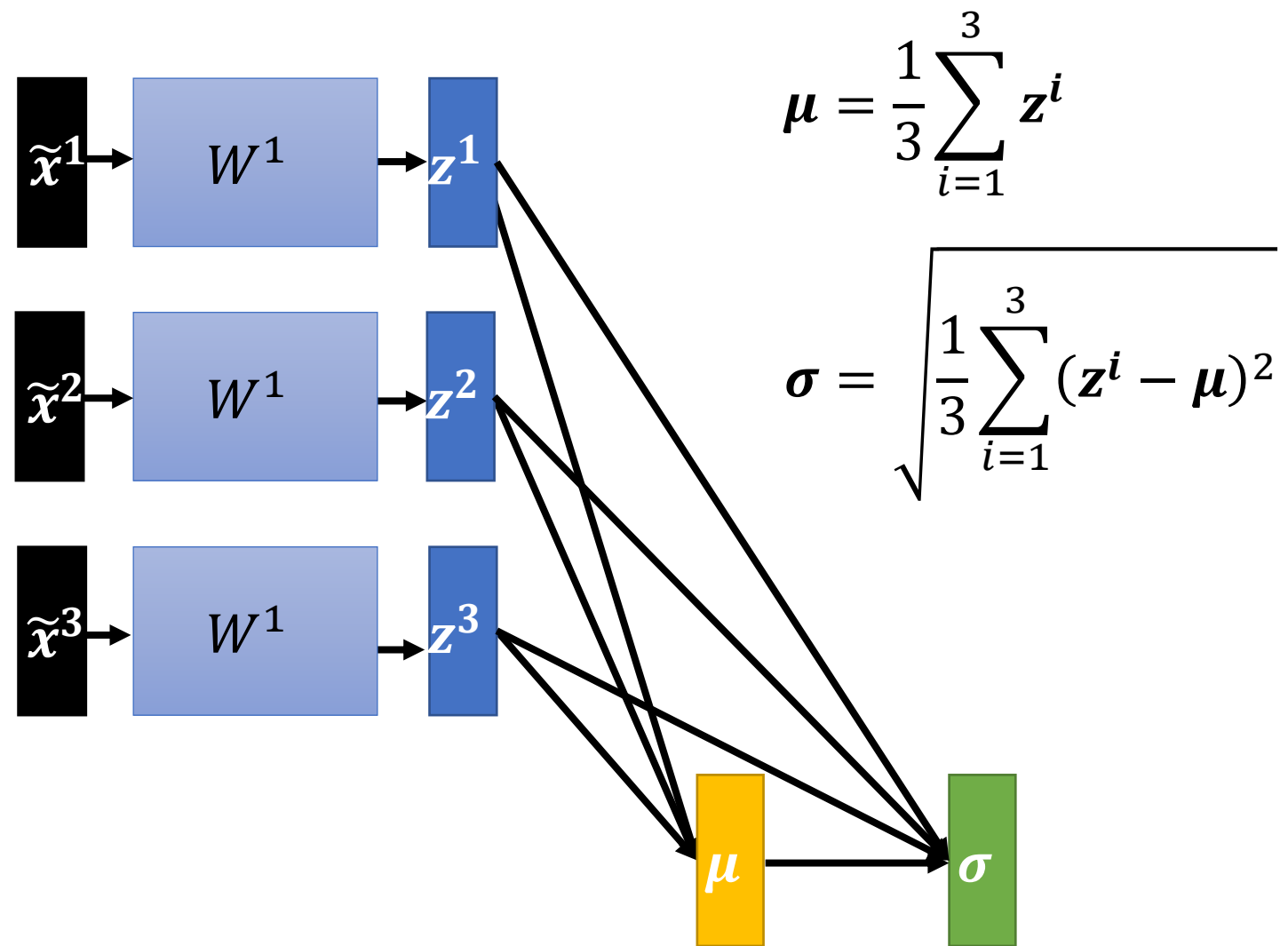
For each dimension i :
mean: m_i
standard deviation: σ_i

$$\tilde{x}_i^r \leftarrow \frac{x_i^r - m_i}{\sigma_i}$$

所有维度均值为0，方差为1

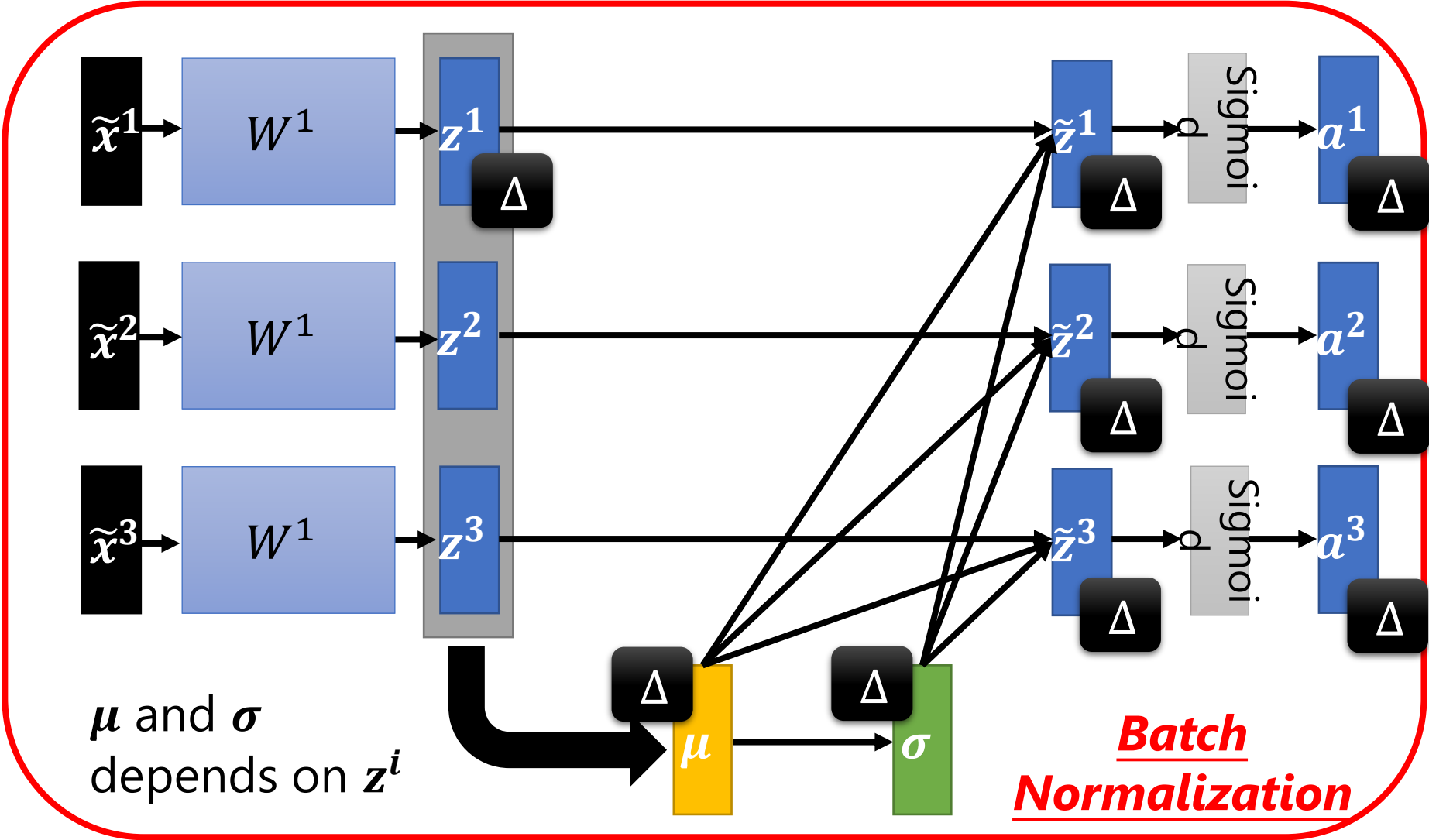
特征标准化可以使网络收敛速度变快





如果这是一个比较深的网络

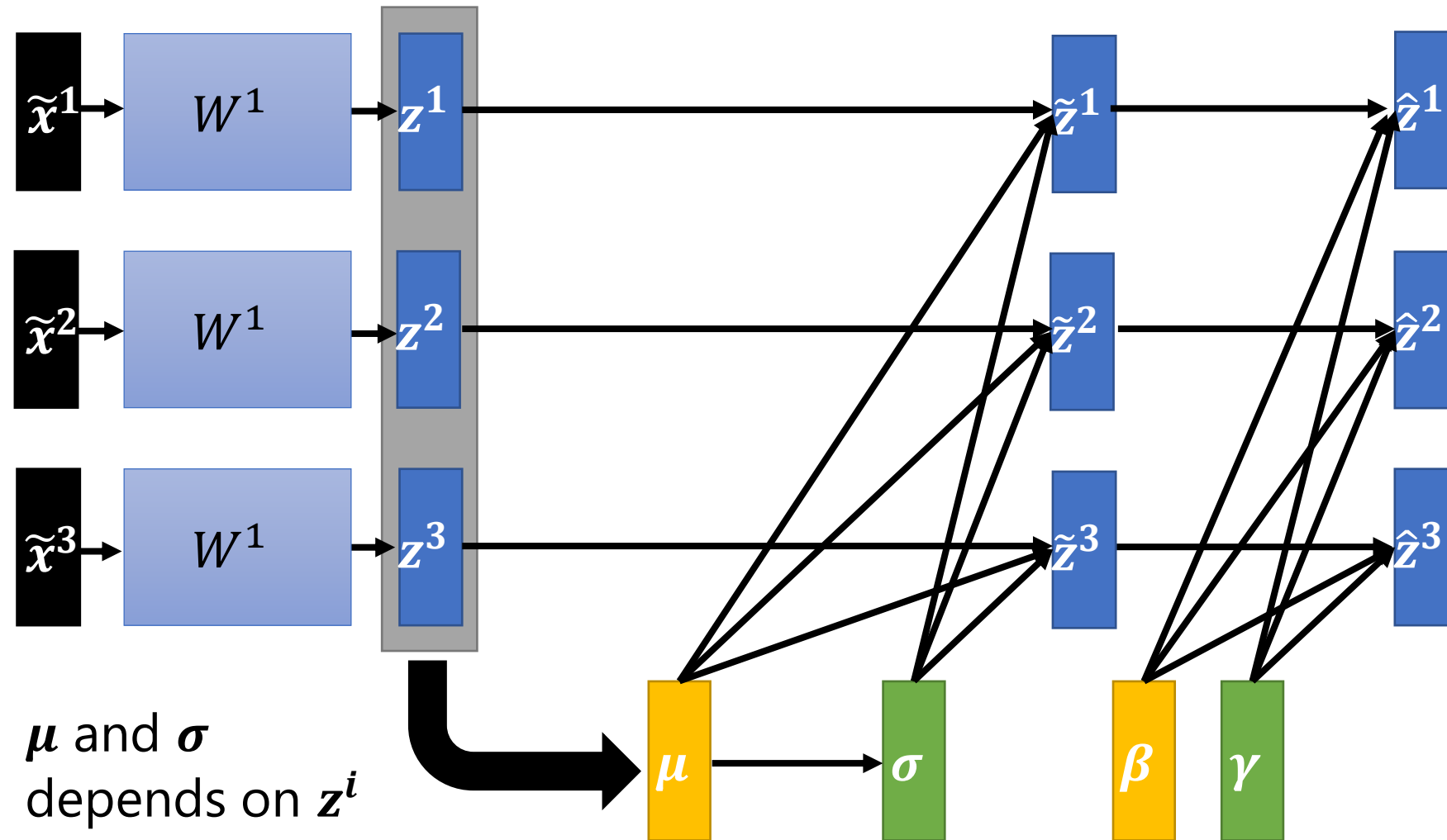
$$\tilde{z}^i = \frac{z^i - \mu}{\sigma}$$



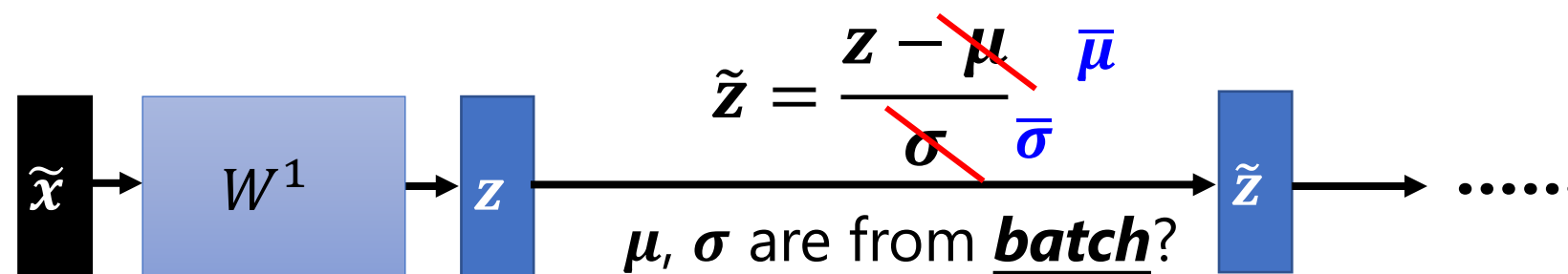
Batch normalization

$$\tilde{z}^i = \frac{z^i - \mu}{\sigma}$$

$$\hat{z}^i = \gamma \odot \tilde{z}^i + \beta$$



Batch normalization – Testing



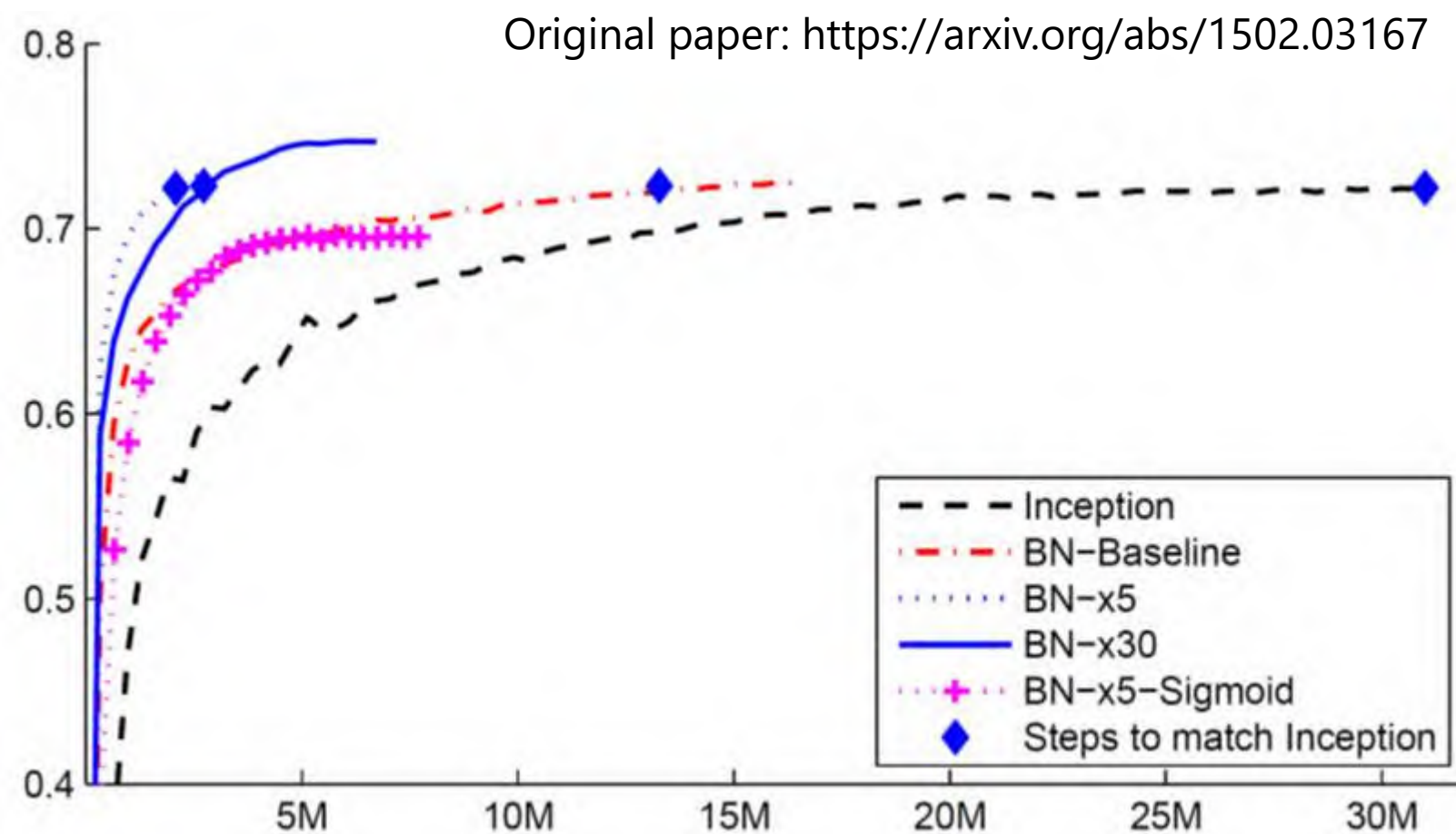
We do not always have batch at testing stage.

Computing the moving average of μ and σ of the batches during training.

$$\mu^1 \quad \mu^2 \quad \mu^3 \quad \dots \quad \mu^t$$

$$\bar{\mu} \leftarrow p\bar{\mu} + (1 - p)\mu^t$$

Batch normalization

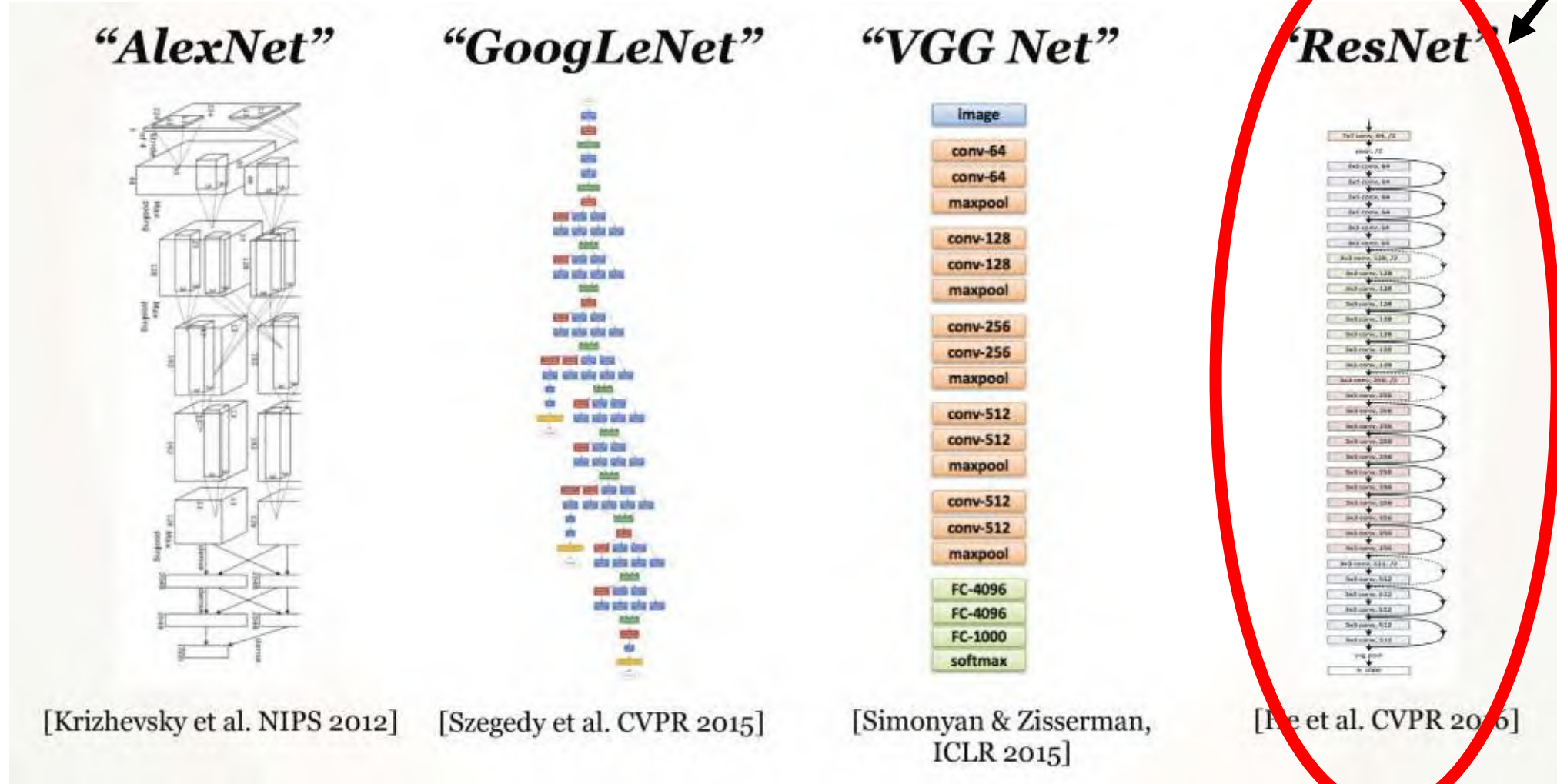


其他normalization.....

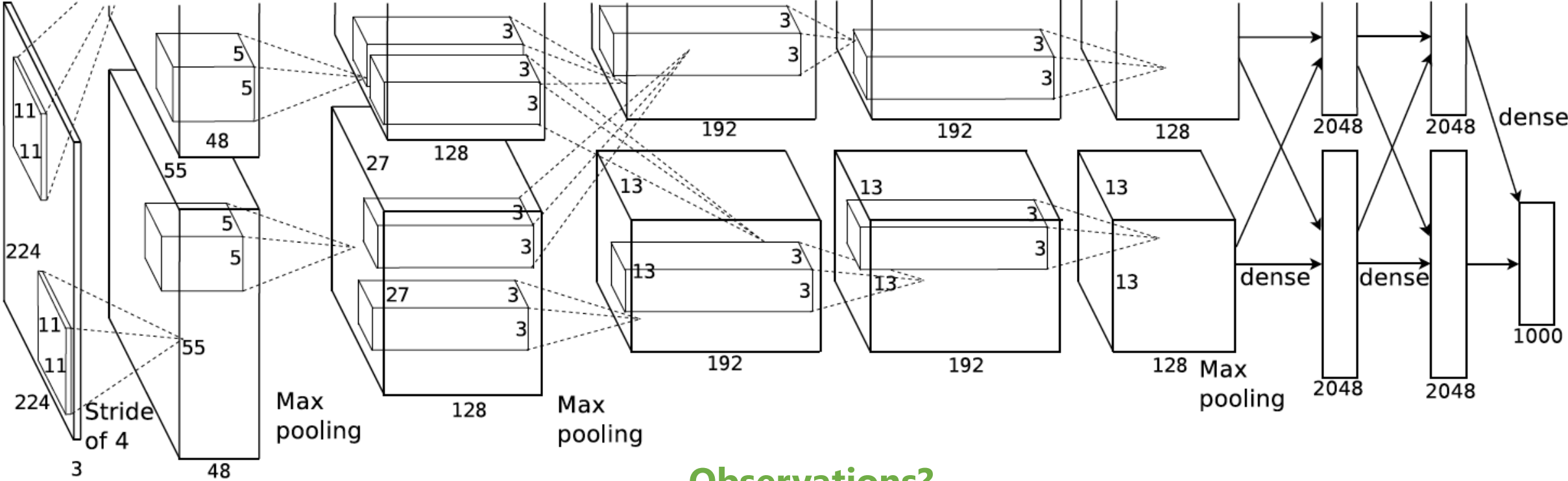
- Batch Renormalization
 - <https://arxiv.org/abs/1702.03275>
- Layer Normalization
 - <https://arxiv.org/abs/1607.06450>
- Instance Normalization
 - <https://arxiv.org/abs/1607.08022>
- Group Normalization
 - <https://arxiv.org/abs/1803.08494>
- Weight Normalization
 - <https://arxiv.org/abs/1602.07868>
- Spectrum Normalization
 - <https://arxiv.org/abs/1705.10941>

CNN网络结构

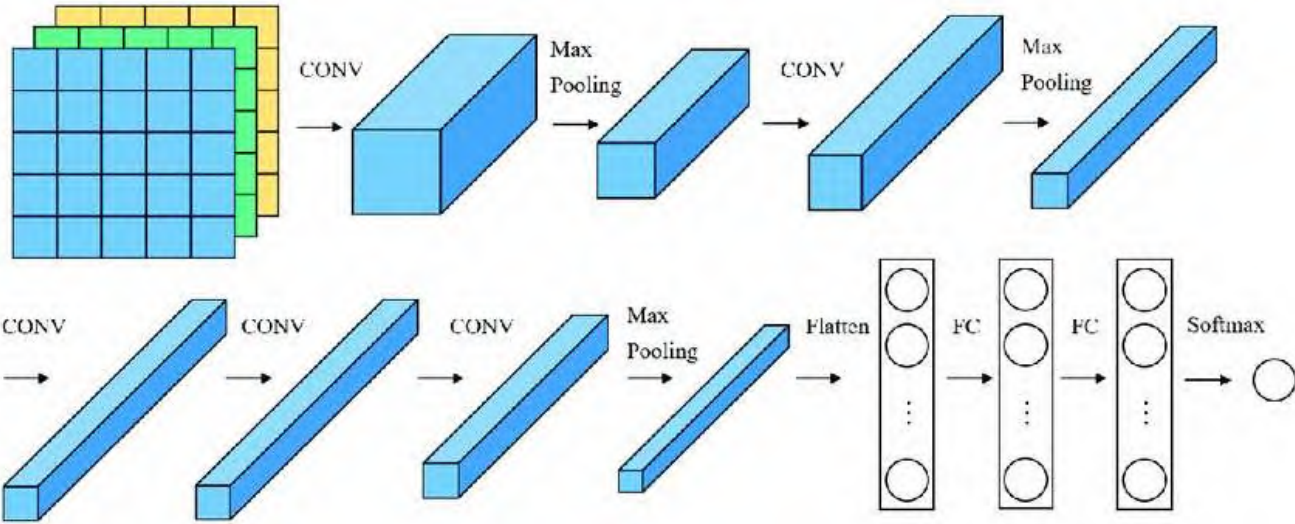
最常用的选择



AlexNet (2012)

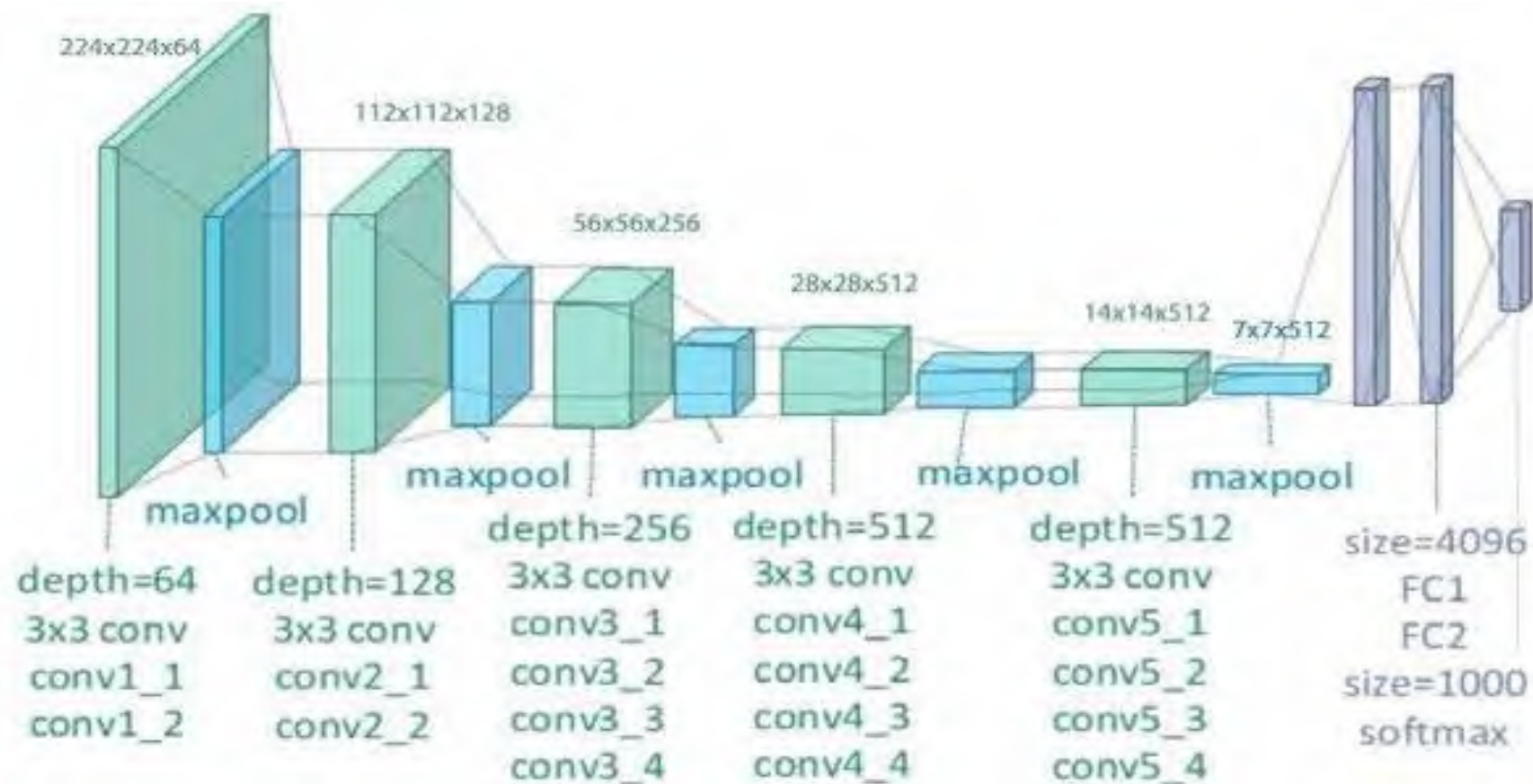


AlexNet (2012)



	Layer	Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	227x227x3	-	-	-
1	Convolution	96	55 x 55 x 96	11x11	4	relu
	Max Pooling	96	27 x 27 x 96	3x3	2	relu
2	Convolution	256	27 x 27 x 256	5x5	1	relu
	Max Pooling	256	13 x 13 x 256	3x3	2	relu
3	Convolution	384	13 x 13 x 384	3x3	1	relu
4	Convolution	384	13 x 13 x 384	3x3	1	relu
5	Convolution	256	13 x 13 x 256	3x3	1	relu
	Max Pooling	256	6 x 6 x 256	3x3	2	relu
6	FC	-	9216	-	-	relu
7	FC	-	4096	-	-	relu
8	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax

VGG (2014)



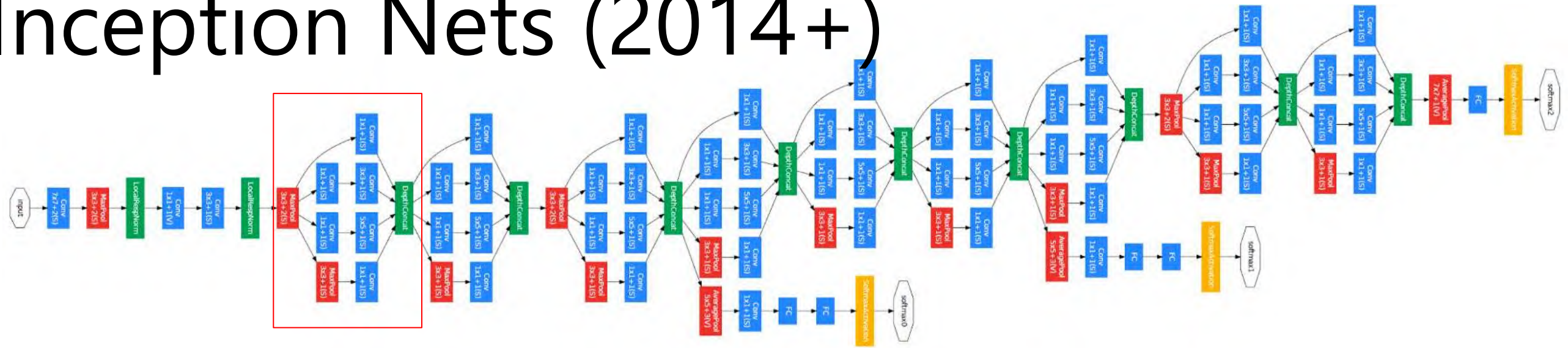
一路 3x3

VGG16 Example

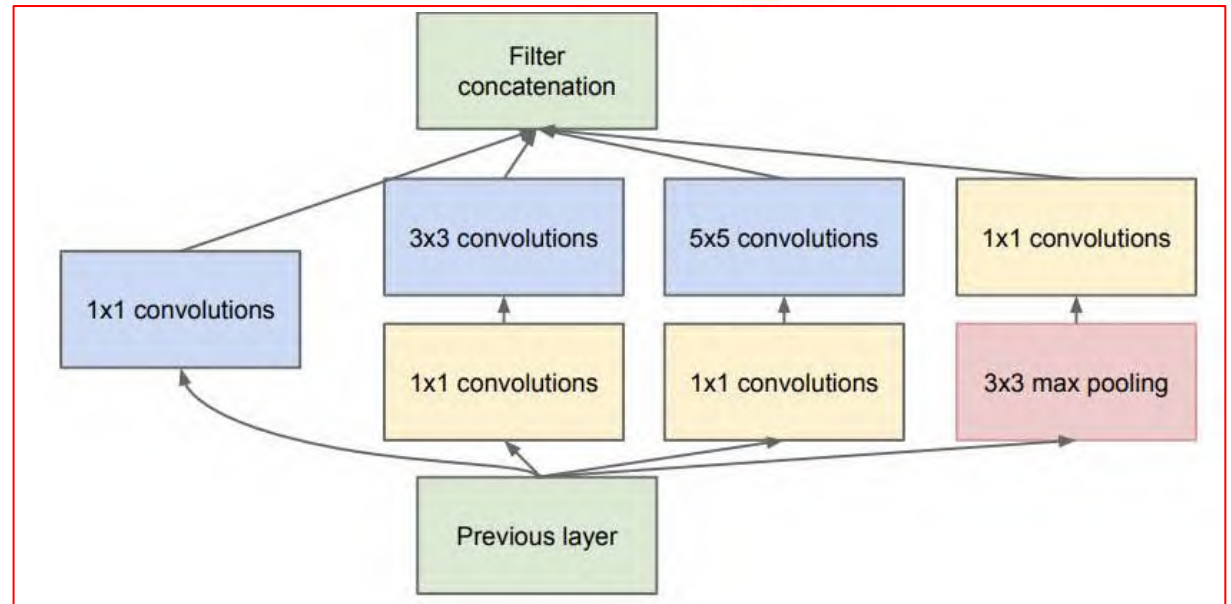
No	Convolution	Output Dimension	Pooling	Output Dimension
layer 1&2	convolution layer of 64 channel of 3x3 kernel with padding 1, stride 1	224x224x64	Max pool stride=2, size 2x2	112x112x64
layer3&4	convolution layer of 128 channel of 3x3 kernel	112x112x128	Max pool stride=2, size 2x2	56x56x128
layer5,6,7	convolution layer of 256 channel of 3x3 kernel	56x56x256	Max pool stride=2, size 2x2	28x28x256
layer8,9,10	Convolution layer of 512 channel of 3x3 kernel	28x28x512	Max pool stride=2, size 2x2	14x14x512
layer11,12,13	Convolution layer of 512 channel of 3x3 kernel	14x14x512	Max pool stride=2, size 2x2	7x7x512



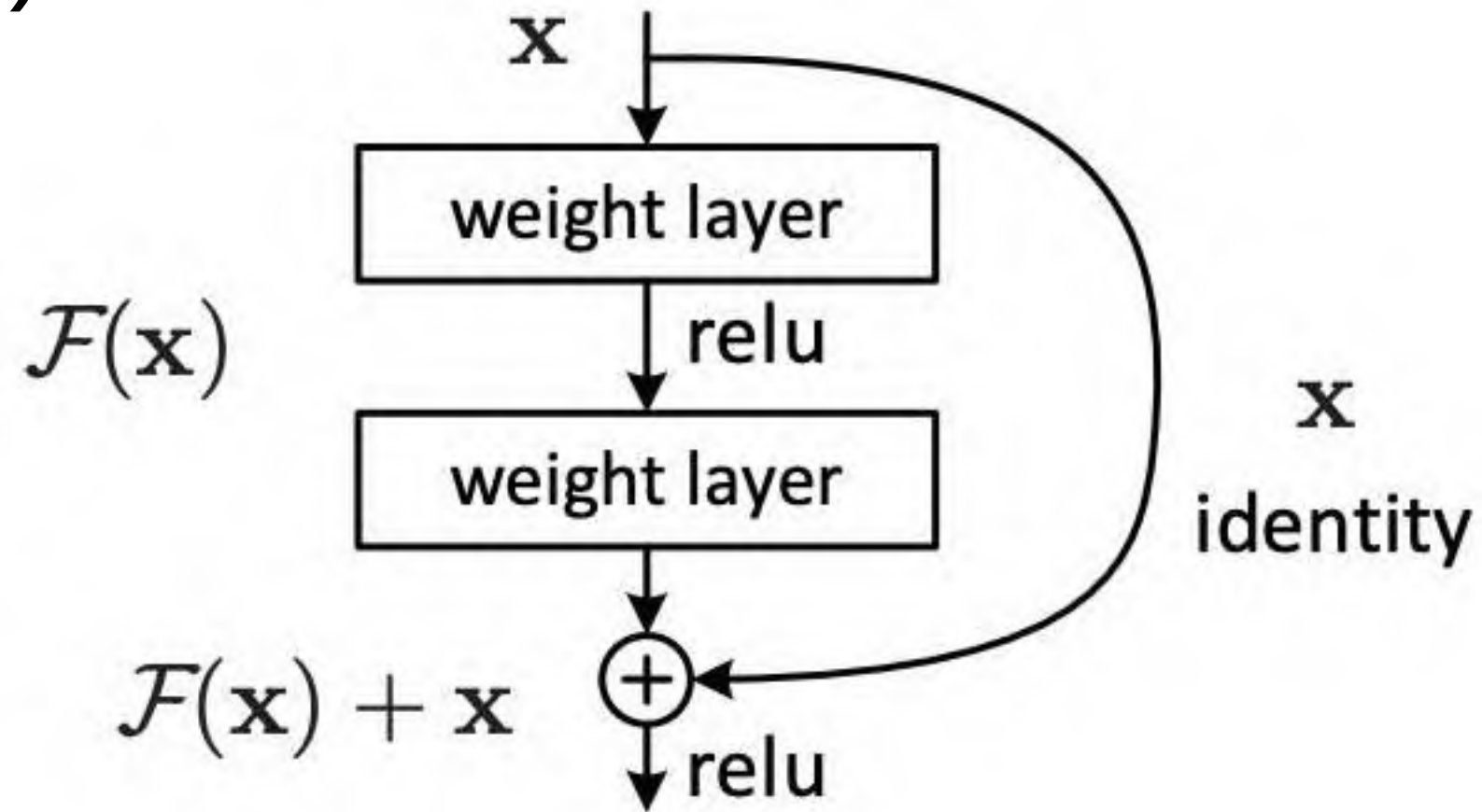
Inception Nets (2014+)



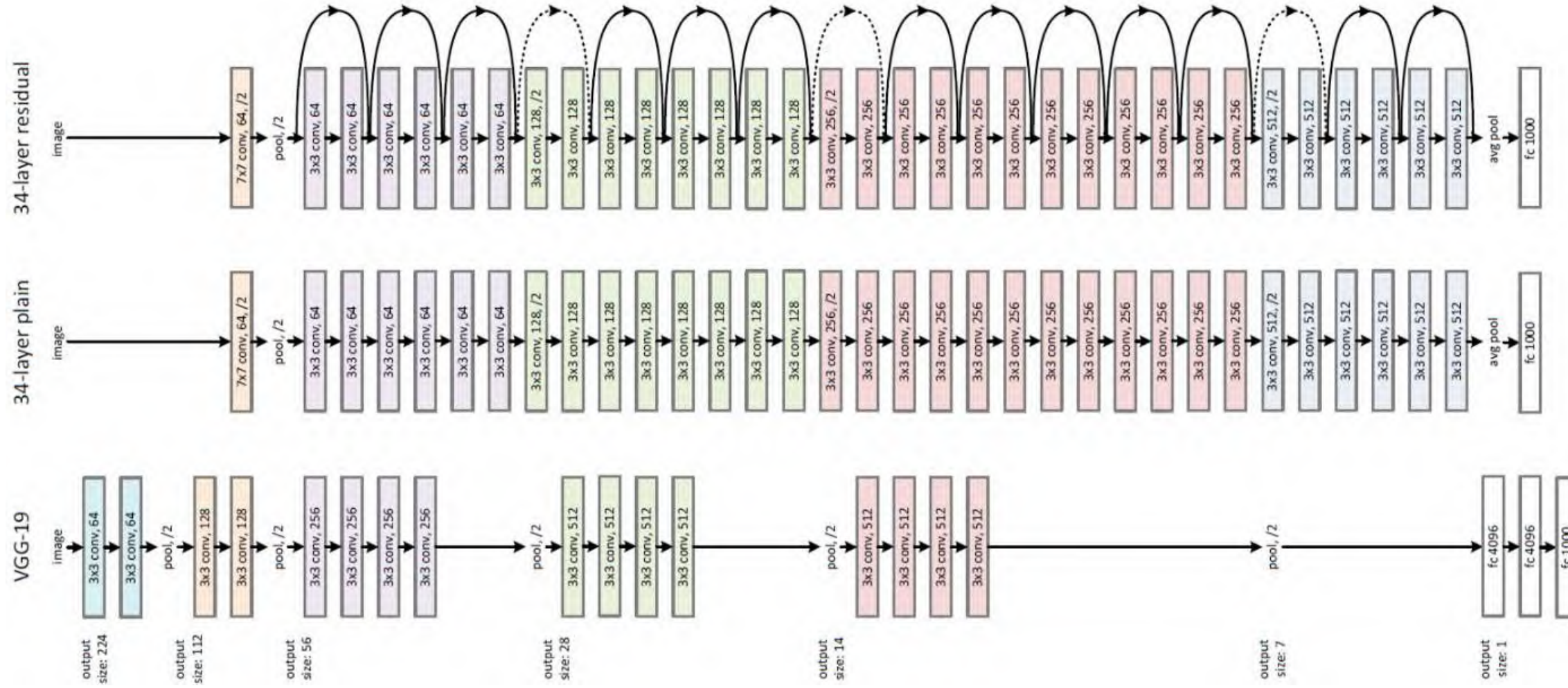
各种卷积组合



ResNet (2015)



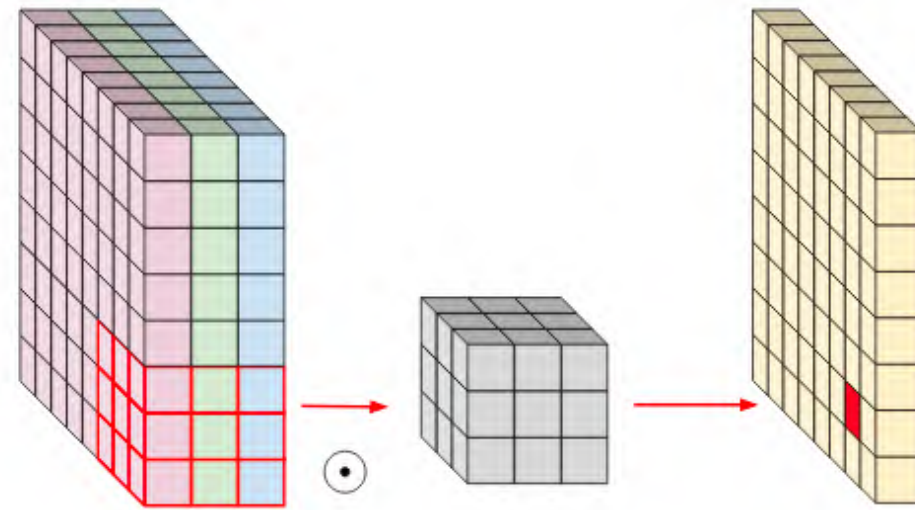
ResNet (2015)



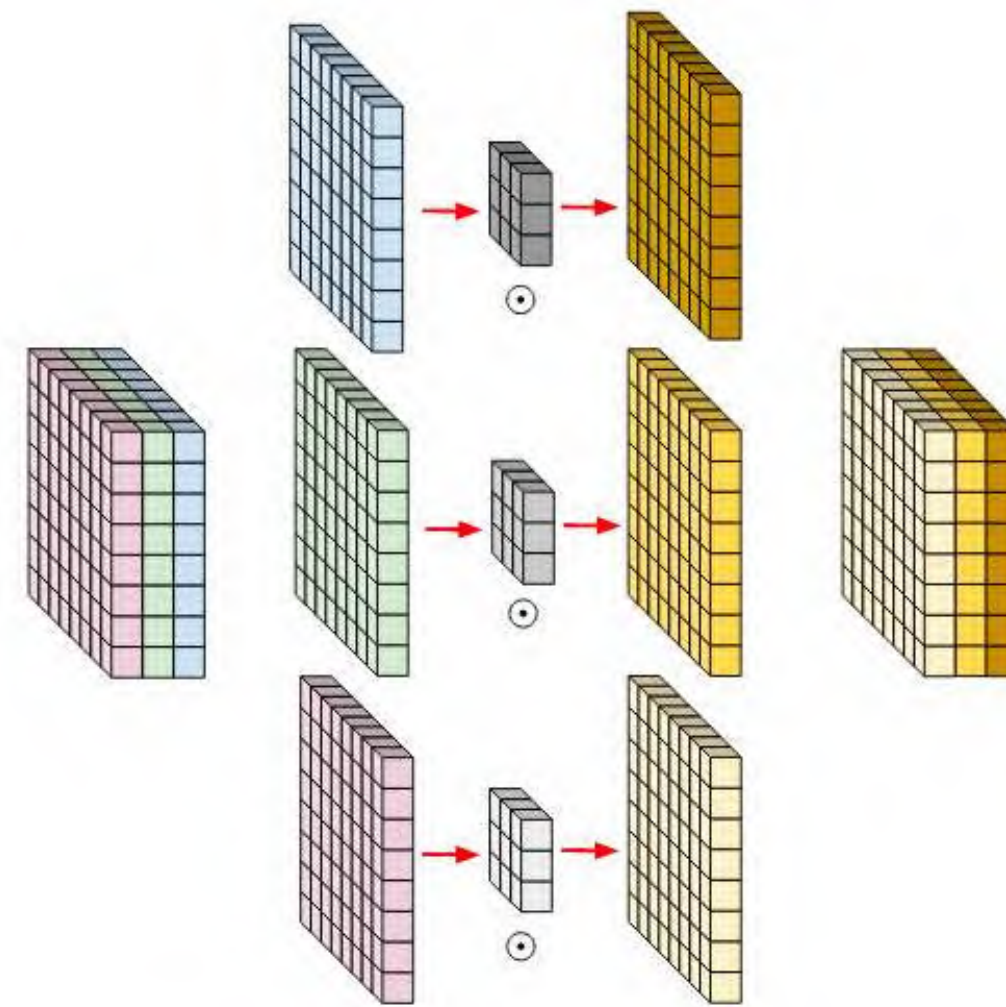
MobileNet (2017)

- Goal:
- 减少网络参数
- 是不是能分开处理每个通道，减少通道数量平方级别的网络参数？
- Answer: Depthwise separable convolutions

Depthwise Convolution

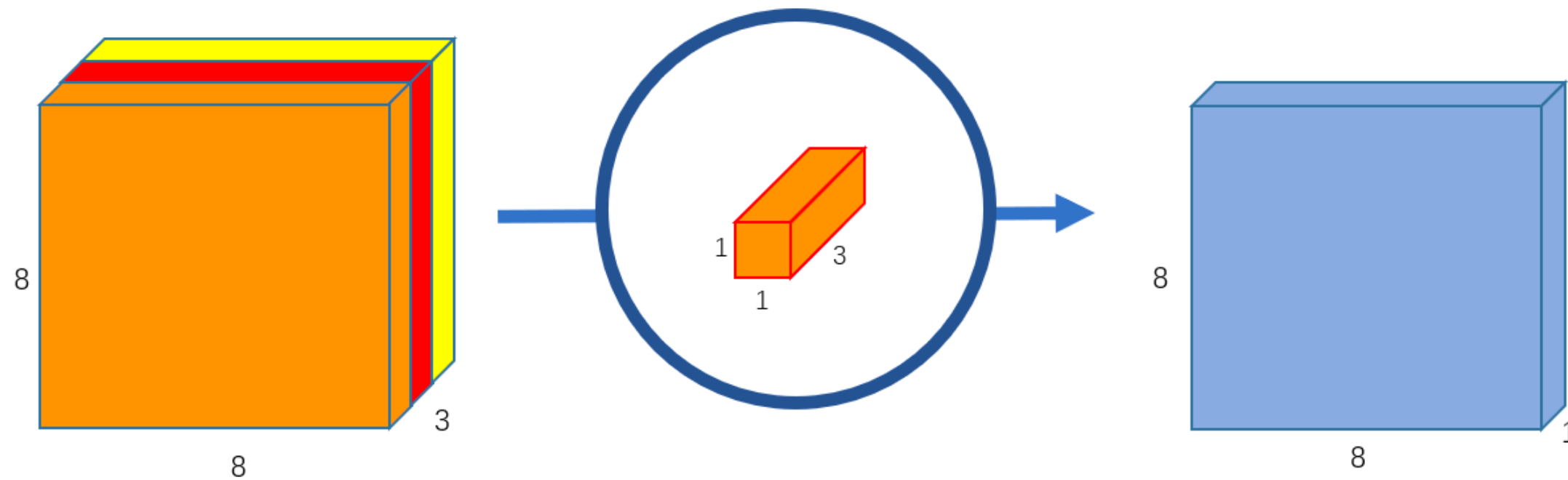


**Normal
Convolution**

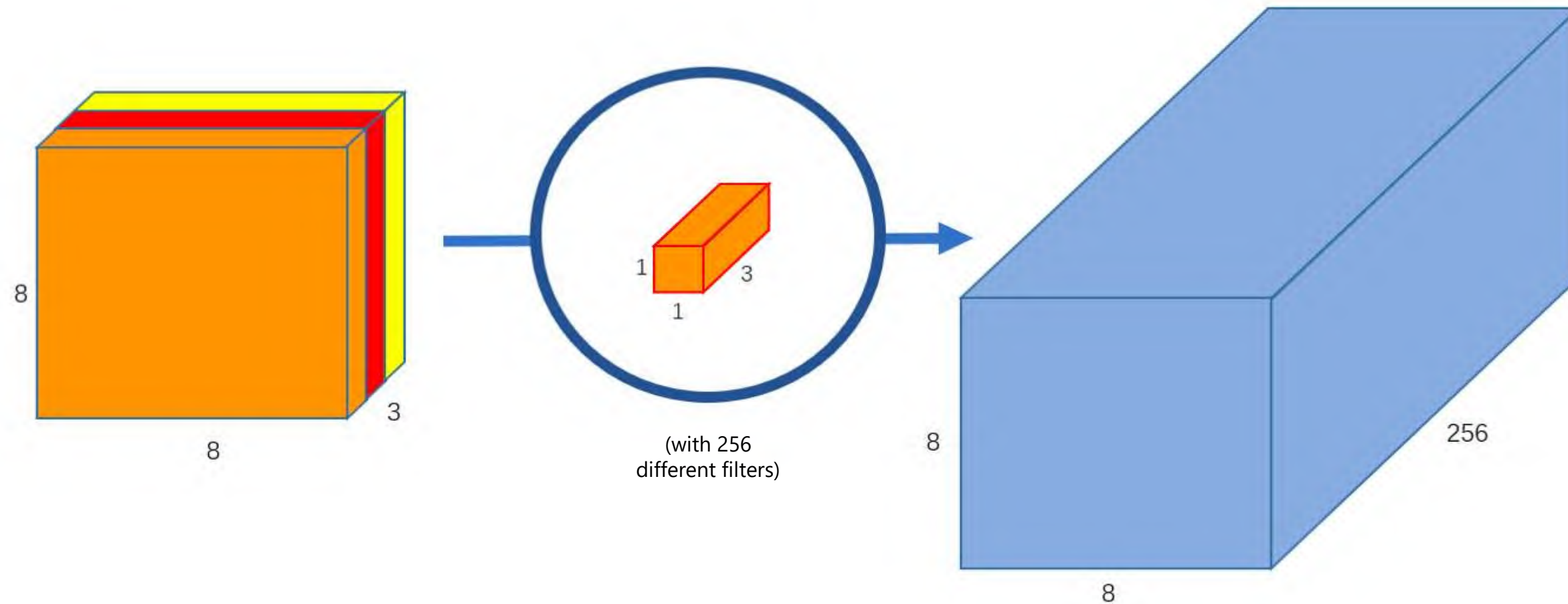


**Depthwise
Convolution**

Pointwise (1x1) Convolutions

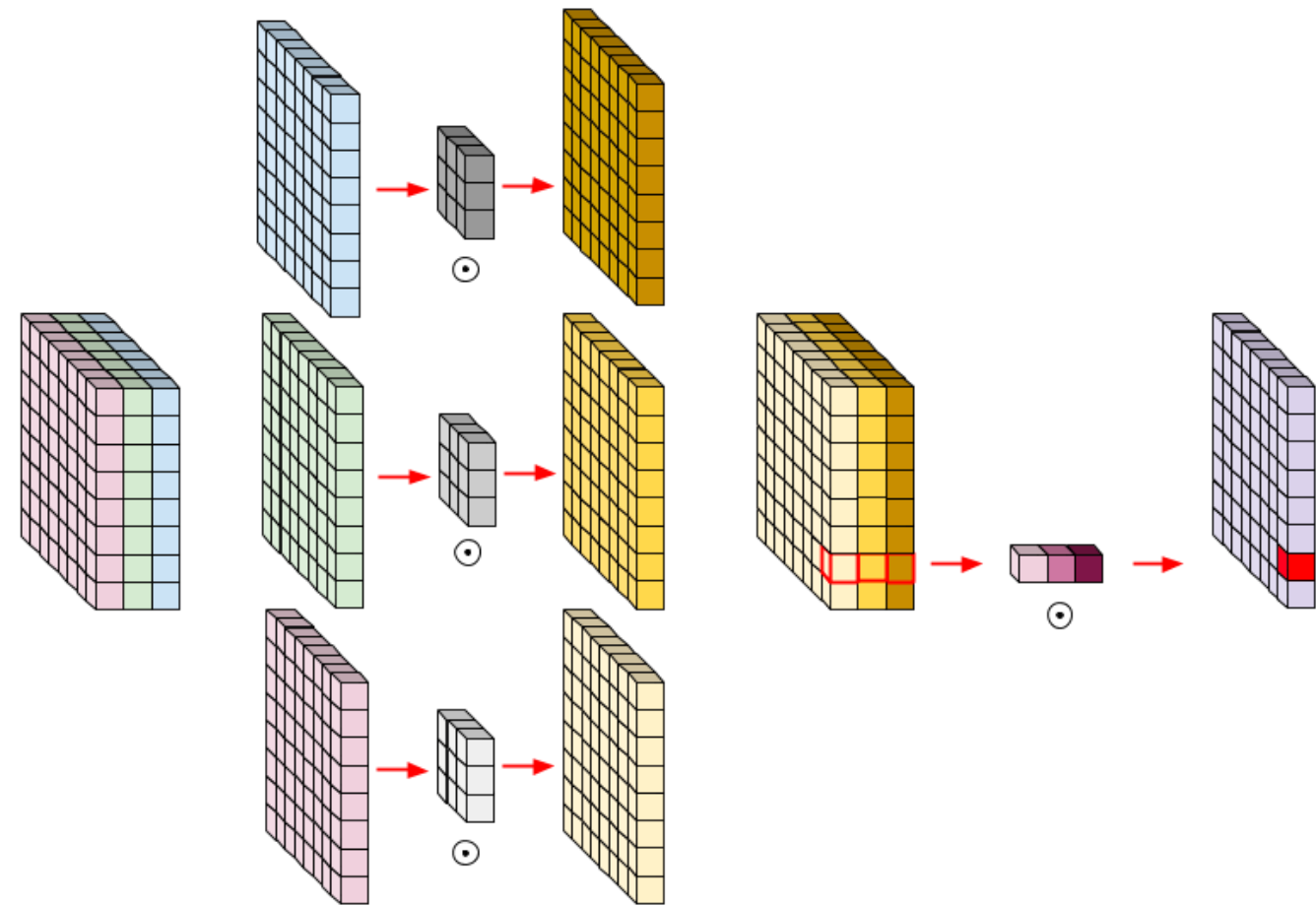


Pointwise (1x1) Convolutions



Depthwise separable convolutions

- 3 input channels, (3, 3) kernel, 8 output channels, no bias
- Answer:
- Normal: $C*H*W*K=3*3*3*8=216$
- DS: $C*H*W + C*K=3*3*3+3*8=51$



MobileNet (2017)

- 与 InceptionV3 接近的准确度
- 十倍的速度，十分之一的参数

Layer/Modification	Million Mult-Adds	Million Parameters
Convolution	462	2.36
Depthwise Separable Conv	52.3	0.27

Table 8. MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
GoogLeNet	69.8%	1550	6.8
VGG 16	71.5%	15300	138

Table 9. Smaller MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
0.50 MobileNet-160	60.2%	76	1.32
Squeezenet	57.5%	1700	1.25
AlexNet	57.2%	720	60

Table 10. MobileNet for Stanford Dogs

Model	Top-1 Accuracy	Million Mult-Adds	Million Parameters
Inception V3 [18]	84%	5000	23.2
1.0 MobileNet-224	83.3%	569	3.3
0.75 MobileNet-224	81.9%	325	1.9
1.0 MobileNet-192	81.9%	418	3.3
0.75 MobileNet-192	80.5%	239	1.9