

# 人工智能实践教程

## 从Python入门到机器学习

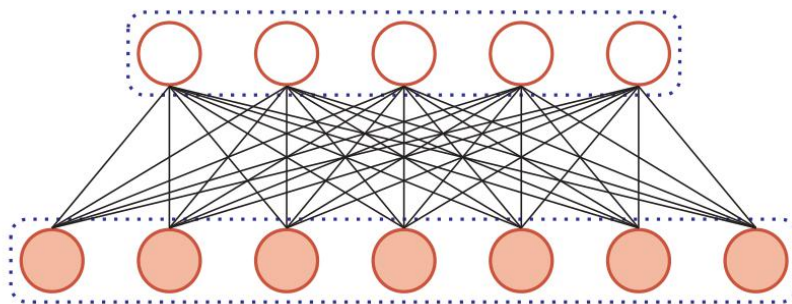


### 第三部分 神经网络 卷积神经网络

- 所有代码及ppt均可以由以下链接下载
- <https://github.com/shao1chuan/pythonbook>
- <https://gitee.com/shao1chuan/pythonbook>

# 全连接前馈神经网络

## ▶ 权重矩阵的参数非常多



## ▶ 局部不变性特征

- ▶ 自然图像中的物体都具有局部不变性特征
  - ▶ 尺度缩放、平移、旋转等操作不影响其语义信息。
- ▶ 全连接前馈网络很难提取这些局部不变特征

# 卷积神经网络

## ▶ 卷积神经网络 (Convolutional Neural Networks, CNN)

- ▶ 一种前馈神经网络

- ▶ 受生物学上感受野 (Receptive Field) 的机制而提出的

- ▶ 在视觉神经系统中，一个神经元的感受野是指视网膜上的特定区域，只有这个区域内的刺激才能够激活该神经元。

## ▶ 卷积神经网络有三个结构上的特性：

- ▶ 局部连接

- ▶ 权重共享

- ▶ 空间或时间上的次采样

# 卷积

- ▶ 卷积经常用在信号处理中，用于计算信号的延迟累积。
- ▶ 假设一个信号发生器每个时刻 $t$ 产生一个信号 $x_t$ ，其信息的衰减率为 $w_k$ ，即在 $k-1$ 个时间步长后，信息为原来的 $w_k$ 倍
  - ▶ 假设 $w_1 = 1, w_2 = 1/2, w_3 = 1/4$
- ▶ 时刻 $t$ 收到的信号 $y_t$ 为当前时刻产生的信息和以前时刻延迟信息的叠加。

# 卷积

- ▶ 卷积经常用在信号处理中，用于计算信号的延迟累积。
- ▶ 假设一个信号发生器每个时刻 $t$ 产生一个信号 $x_t$ ，其信息的衰减率为 $w_k$ ，即在 $k-1$ 个时间步长后，信息为原来的 $w_k$ 倍
  - ▶ 假设 $w_1 = 1, w_2 = 1/2, w_3 = 1/4$
- ▶ 时刻 $t$ 收到的信号 $y_t$ 为当前时刻产生的信息和以前时刻延迟信息的叠加

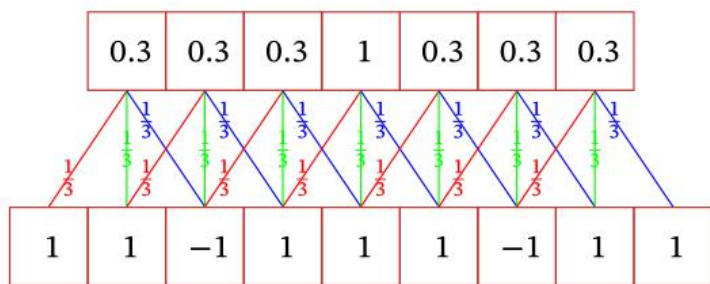
$$\begin{aligned}y_t &= 1 \times x_t + 1/2 \times x_{t-1} + 1/4 \times x_{t-2} \\ &= w_1 \times x_t + w_2 \times x_{t-1} + w_3 \times x_{t-2} \\ &= \sum_{k=1}^3 w_k \cdot x_{t-k+1}.\end{aligned}$$

滤波器 (filter) 或卷积核 (convolution kernel)



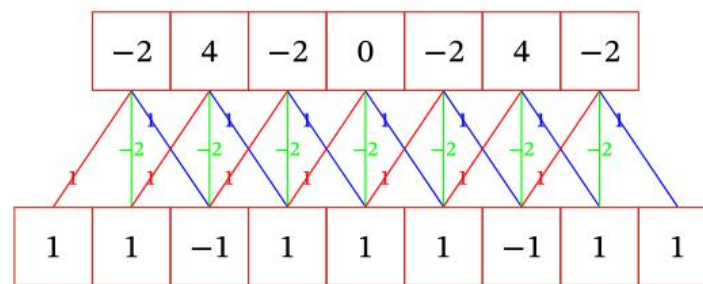
# 卷积

- 不同的滤波器来提取信号序列中的不同特征



(a) 滤波器  $[1/3, 1/3, 1/3]$

低频信息



(b) 滤波器  $[1, -2, 1]$

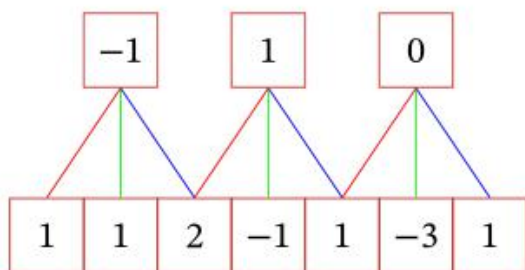
高频信息

$$y''(u) = y(u + 1) + y(u - 1) - 2y(u)$$

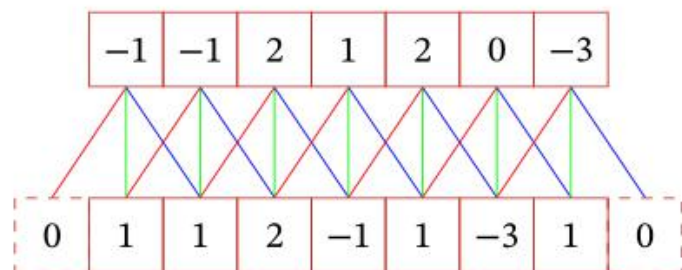
二阶微分

# 卷积扩展

- ▶ 引入滤波器的滑动步长 $S$ 和零填充 $P$



(a) 步长  $S = 2$



(b) 零填充  $P = 1$



# 卷积类型

## ▶ 卷积的结果按输出长度不同可以分为三类：

▶ **窄卷积**：步长  $\text{stride} = 1$ ，两端不补零  $\text{padding} = 0$ ，卷积后输出长度为  $\text{input\_length} - \text{kernel\_length} + 1$

▶ **宽卷积**：步长  $\text{stride} = 1$ ，两端补零  $\text{padding} = \text{kernel\_length} - 1$ ，卷积后输出长度  $\text{input\_length} + \text{kernel\_length} - 1$

▶ **等宽卷积**：步长  $\text{stride} = 1$ ，两端补零  $\text{padding} = (\text{kernel\_length} - 1) / 2$ ，卷积后输出长度  $\text{input\_length}$

▶ 在早期的文献中，卷积一般默认为**窄卷积**。

▶ 而目前的文献中，卷积一般默认为**等宽卷积**。

# 二维卷积

- 在图像处理中，图像是以二维矩阵的形式输入到神经网络中，因此我们需要二维卷积。

一个输入信息  $\mathbf{X}$  和滤波器  $\mathbf{W}$  的二维卷积定义为

$$\mathbf{Y} = \mathbf{W} * \mathbf{X},$$

$$y_{ij} = \sum_{u=1}^U \sum_{v=1}^V w_{uv} x_{i-u+1, j-v+1}.$$

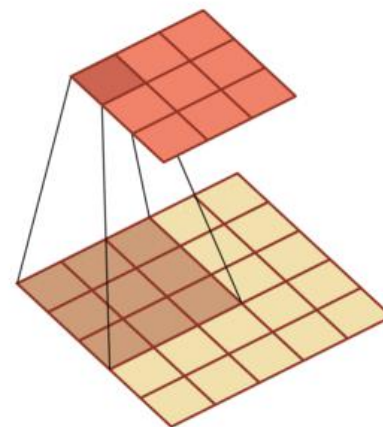
1	1	1	1	1
-1	0	-3	0	1
2	1	1	-1	0
0	-1	1	2	1
1	2	1	1	1

1	0	0
0	0	0
0	0	-1

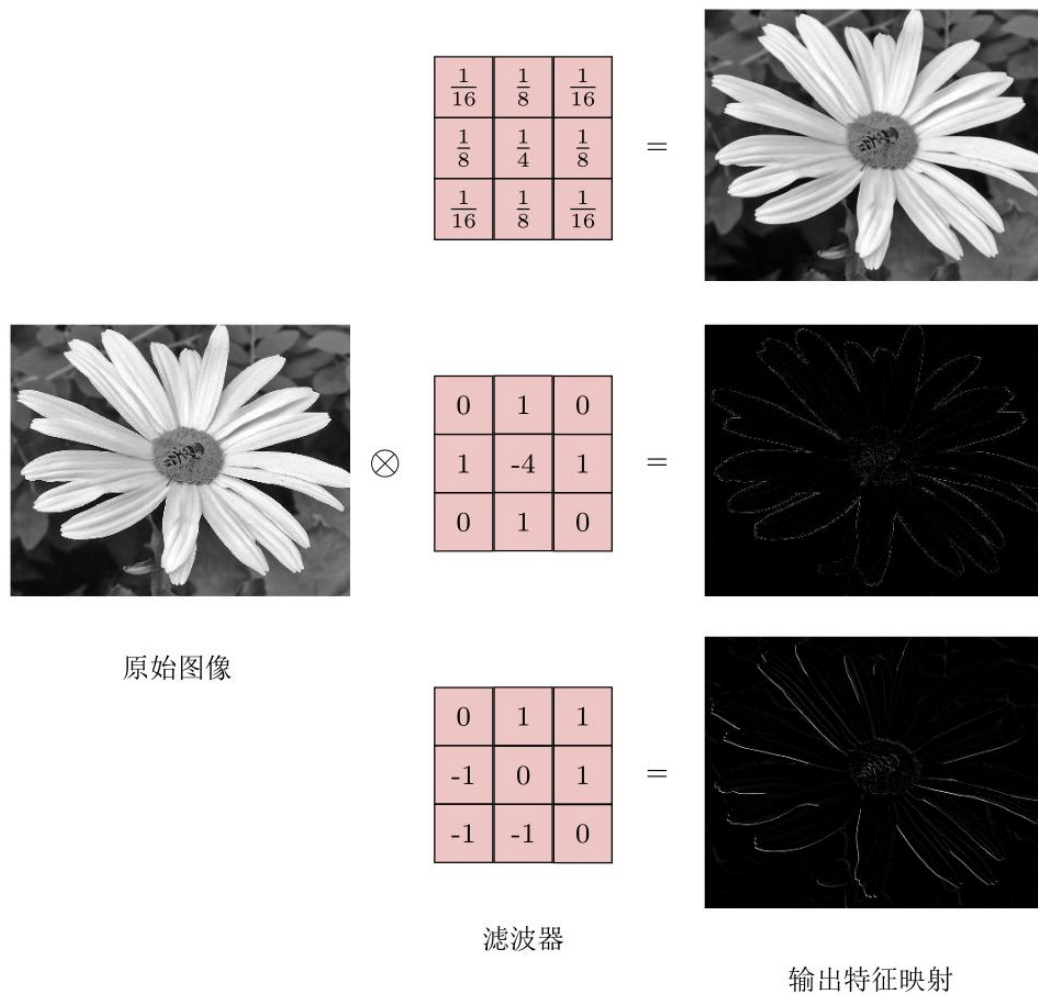
$$=$$

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

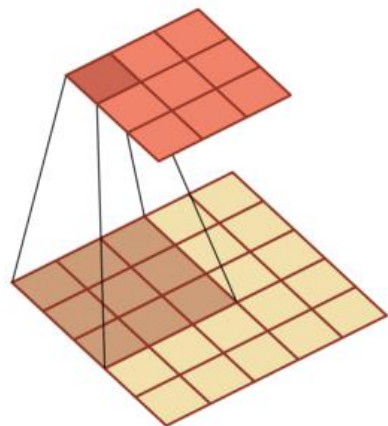
*Note: In the original image, the 3x3 kernel and the 3x3 output are highlighted in red. Blue annotations 'x-1', 'x0', and 'x1' are present under the input matrix, indicating the sliding window for each output element.*



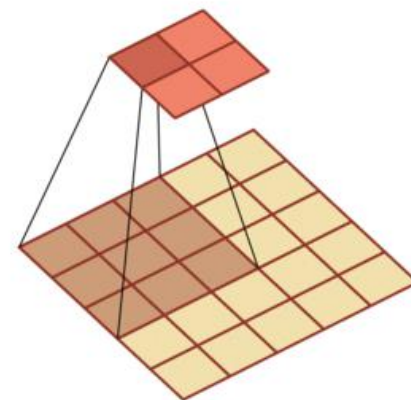
# 卷积作为特征提取器



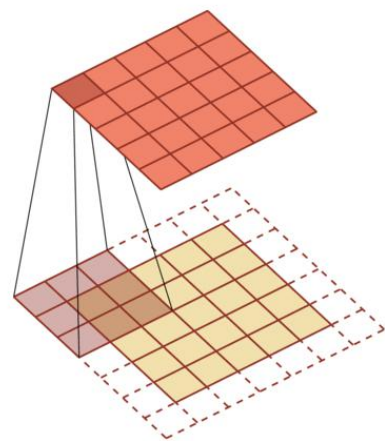
# 二维卷积



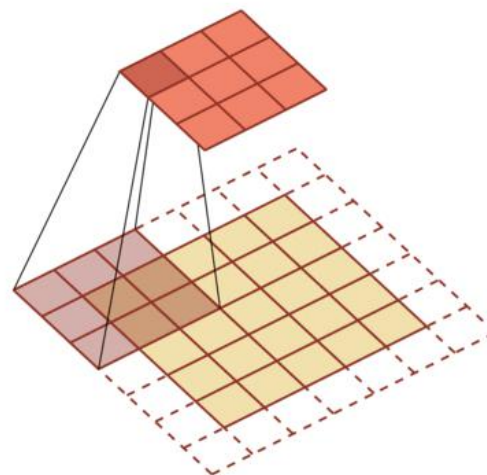
步长1，零填充0



步长2，零填充0



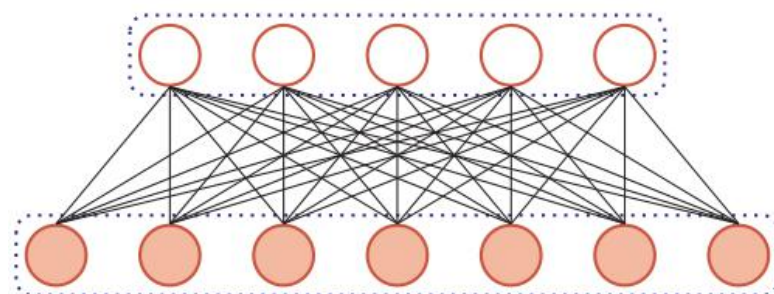
步长1，零填充1



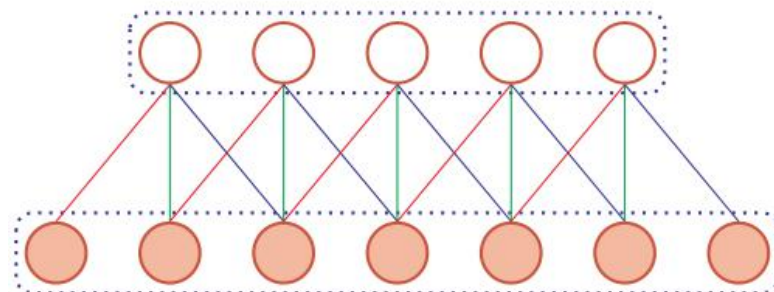
步长2，零填充1

# 卷积神经网络

## ► 用卷积层代替全连接层



(a) 全连接层



(b) 卷积层

# 互相关

- ▶ 计算卷积需要进行卷积核翻转。
- ▶ 卷积操作的目标：提取特征。

翻转是不必要的！

- ▶ 互相关

$$y_{ij} = \sum_{u=1}^m \sum_{v=1}^n w_{uv} \cdot x_{i+u-1, j+v-1}$$

除非特别声明，卷积一般指“互相关”。

# 多个卷积核

▶ 特征映射 (Feature Map) : 图像经过卷积后得到的特征。

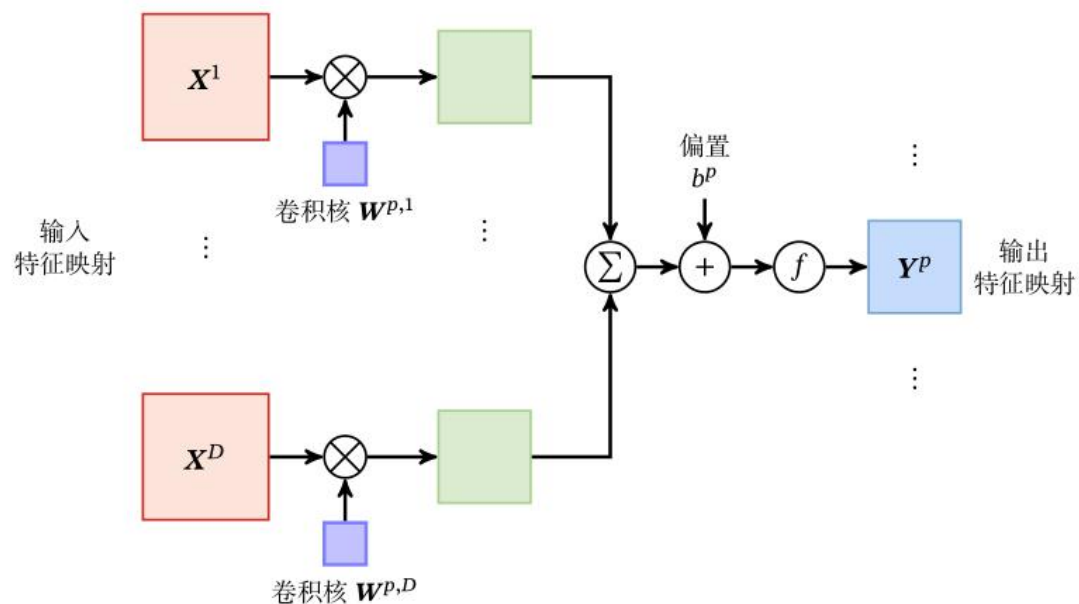
▶ 卷积核看成一个特征提取器

▶ 卷积层

▶ 输入:  $D$  个特征映射  $M \times N \times D$

▶ 输出:  $P$  个特征映射  $M' \times N' \times P$

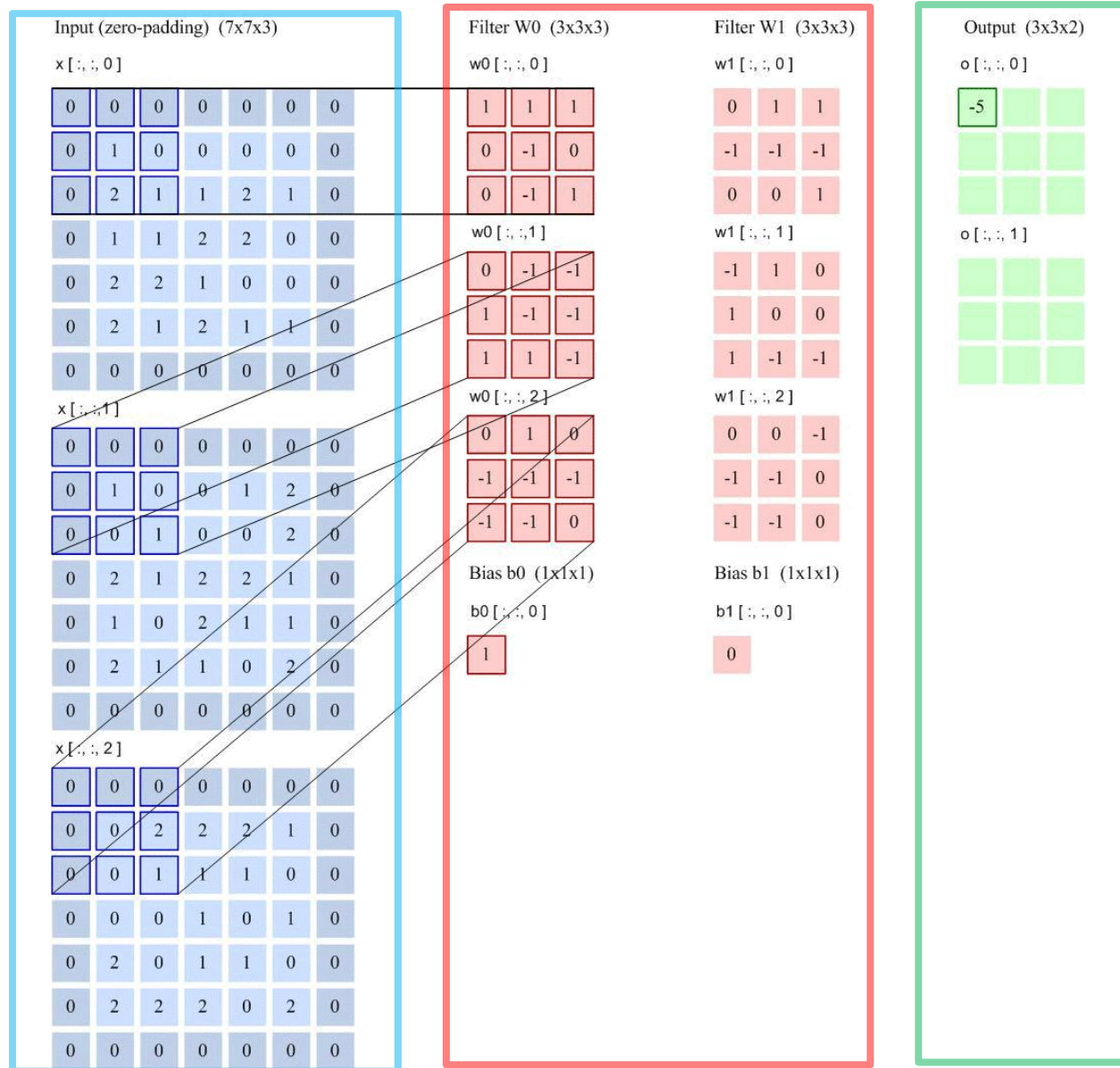
# 卷积层的映射关系



$$Z^p = W^p \otimes X + b^p = \sum_{d=1}^D W^{p,d} \otimes X^d + b^p,$$

$$Y^p = f(Z^p).$$

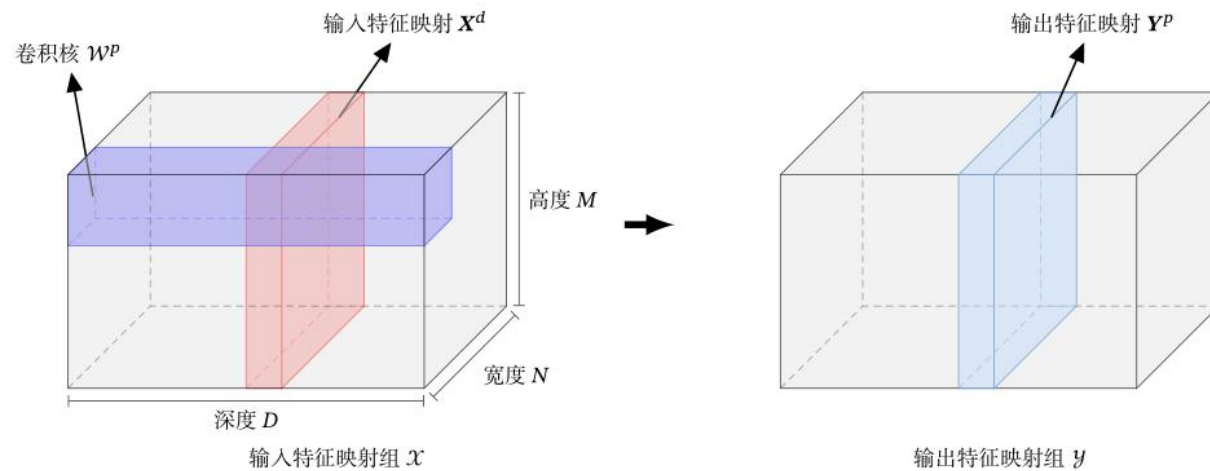




步长2  
 filter 3\*3  
 filter个数6  
 零填充 1

# 卷积层

## ► 典型的卷积层为3维结构

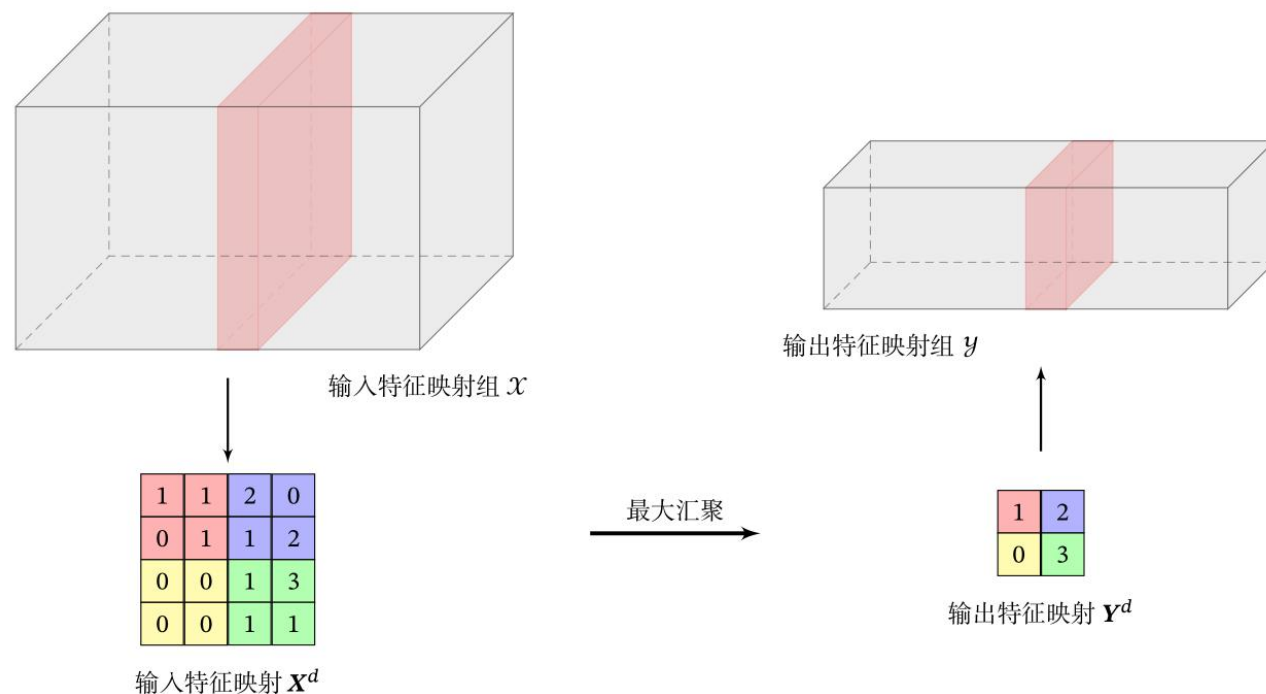


$$Z^p = W^p \otimes X + b^p = \sum_{d=1}^D W^{p,d} \otimes X^d + b^p,$$

$$Y^p = f(Z^p).$$

# 汇聚层

- 卷积层虽然可以显著减少连接的个数，但是每一个特征映射的神经元个数并没有显著减少。



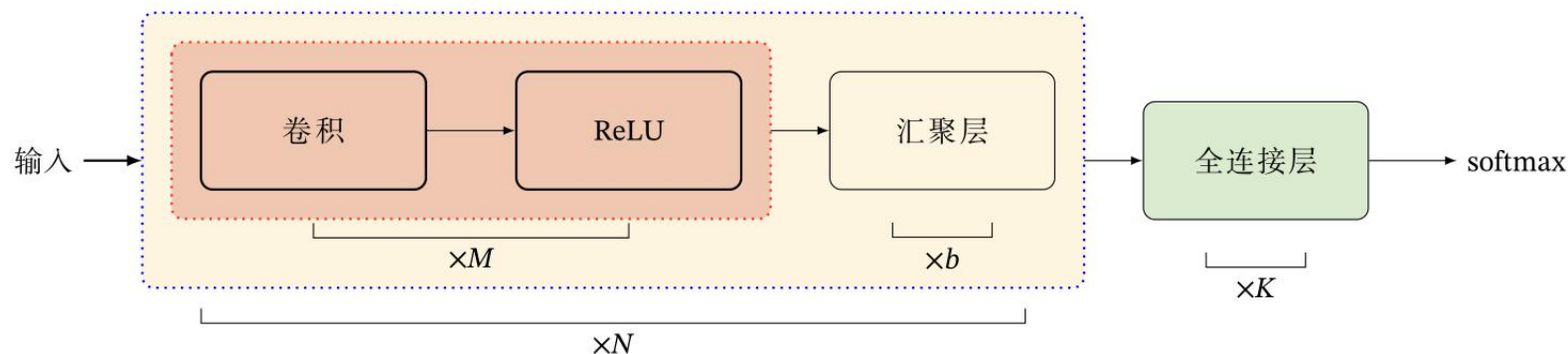
# 卷积网络结构

▶ 卷积网络是由卷积层、汇聚层、全连接层交叉堆叠而成。

▶ 趋向于小卷积、大深度

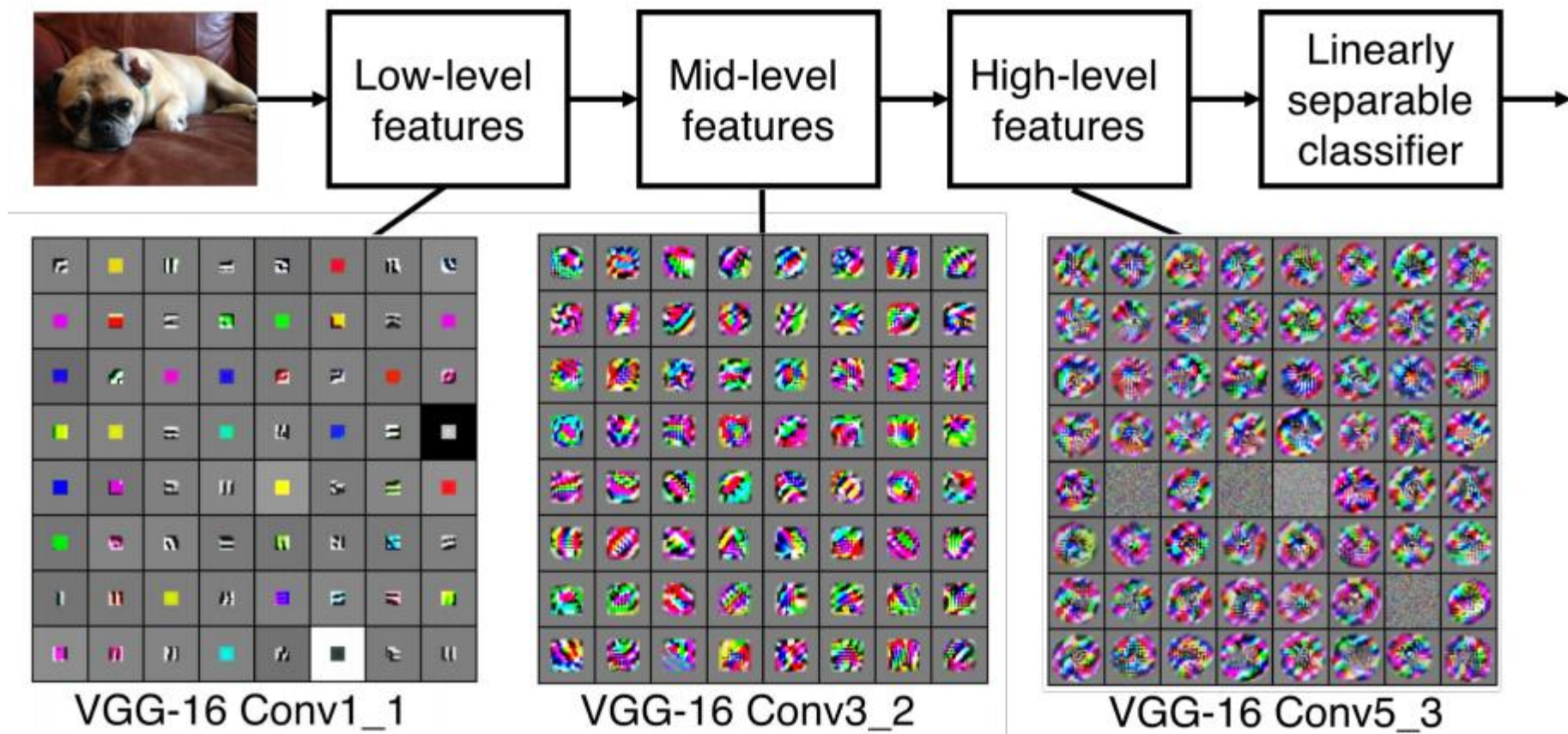
▶ 趋向于全卷积

▶ 典型结构

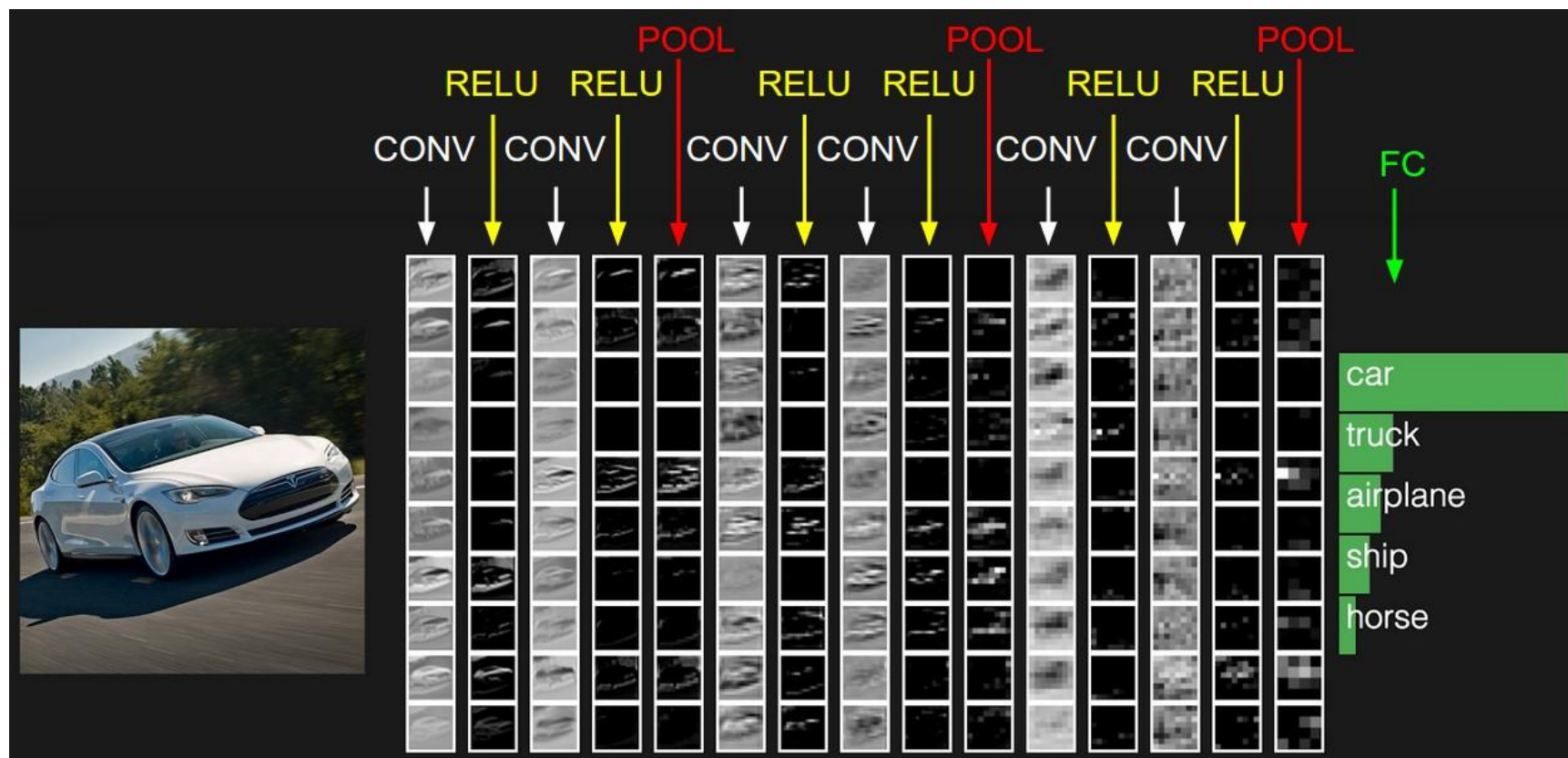


▶ 一个卷积块为连续M个卷积层和b个汇聚层（M通常设置为2~5，b为0或1）。一个卷积网络中可以堆叠N个连续的卷积块，然后在接着K个全连接层（N的取值区间比较大，比如1~100或者更大；K一般为0~2）。

# 表示学习



# 表示学习

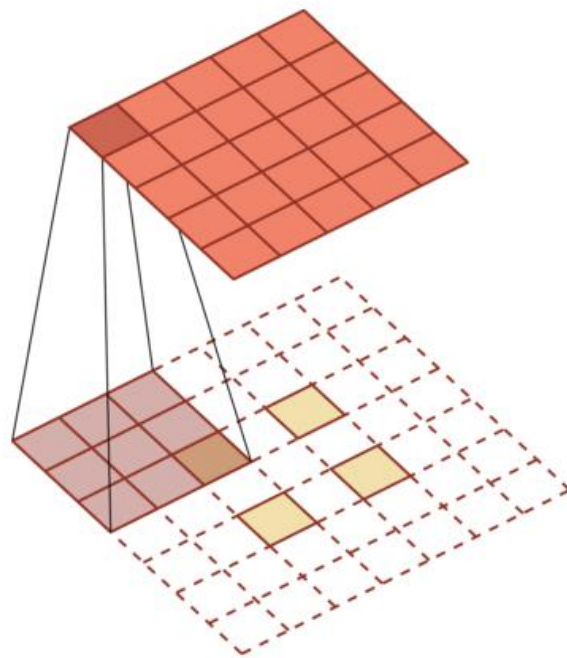
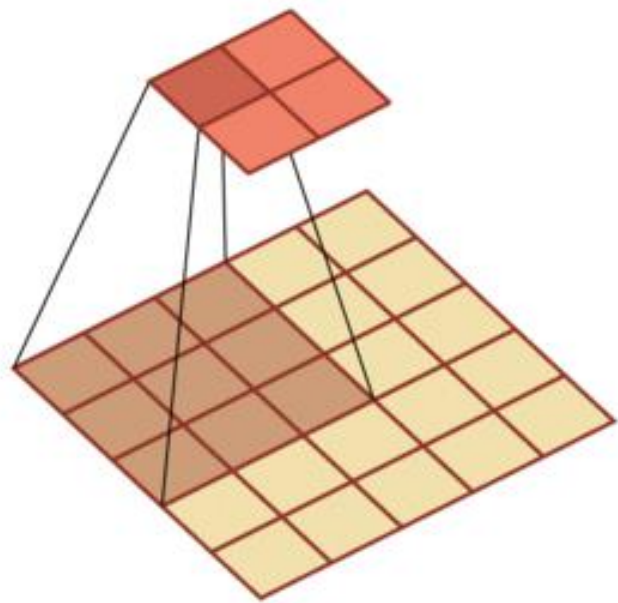


其它卷积种类



# 转置卷积/微步卷积

► 低维特征映射到高维特征





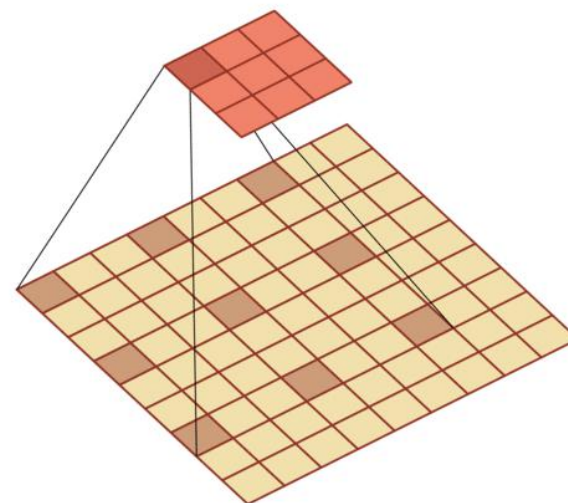
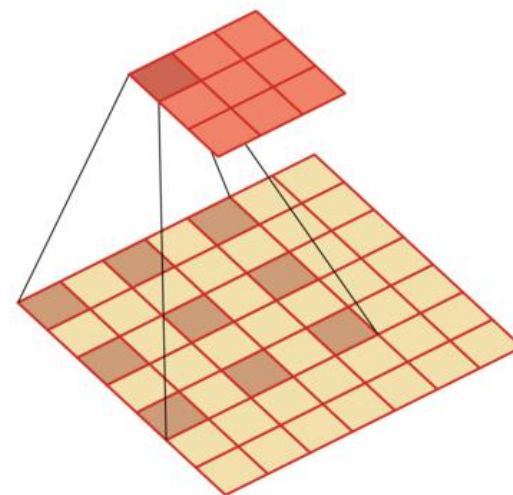
# 空洞卷积

## ▶ 如何增加输出单元的感受野

- ▶ 增加卷积核的大小
- ▶ 增加层数来实现
- ▶ 在卷积之前进行汇聚操作

## ▶ 空洞卷积

- ▶ 通过给卷积核插入“空洞”来变相地增加其大小。

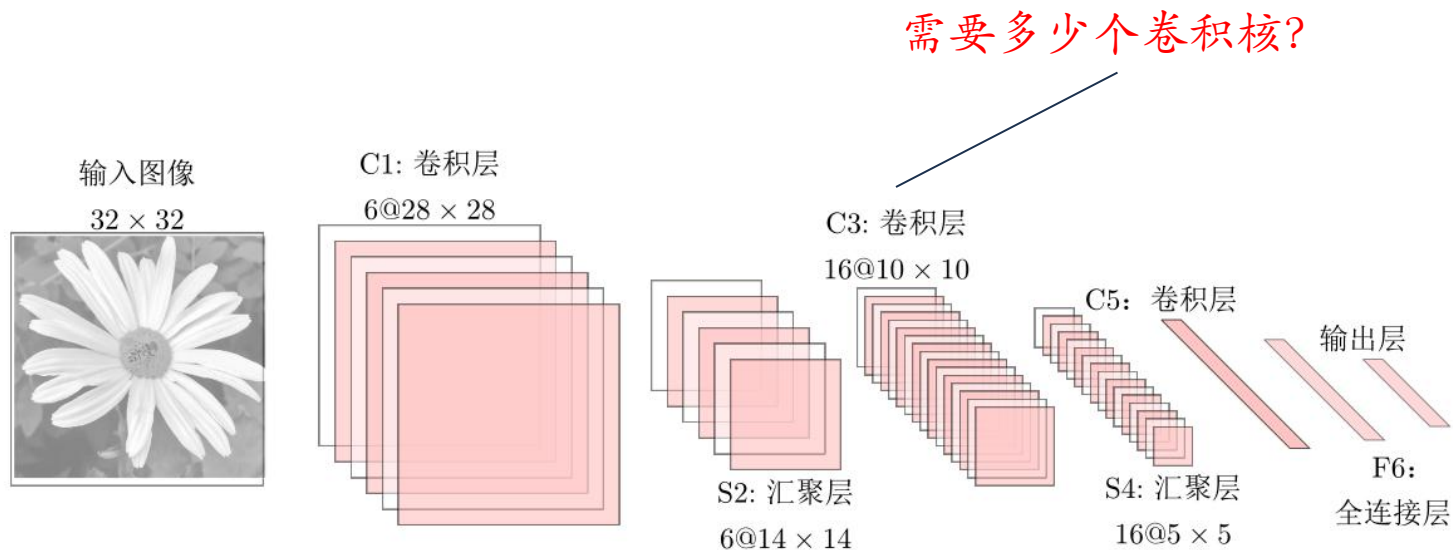


# 典型的卷积网络

# LeNet-5


▶ LeNet-5 是一个非常成功的神经网络模型。

- ▶ 基于 LeNet-5 的手写数字识别系统在 90 年代被美国很多银行使用，用来识别支票上面的手写数字。
- ▶ LeNet-5 共有 7 层。



# Large Scale Visual Recognition Challenge

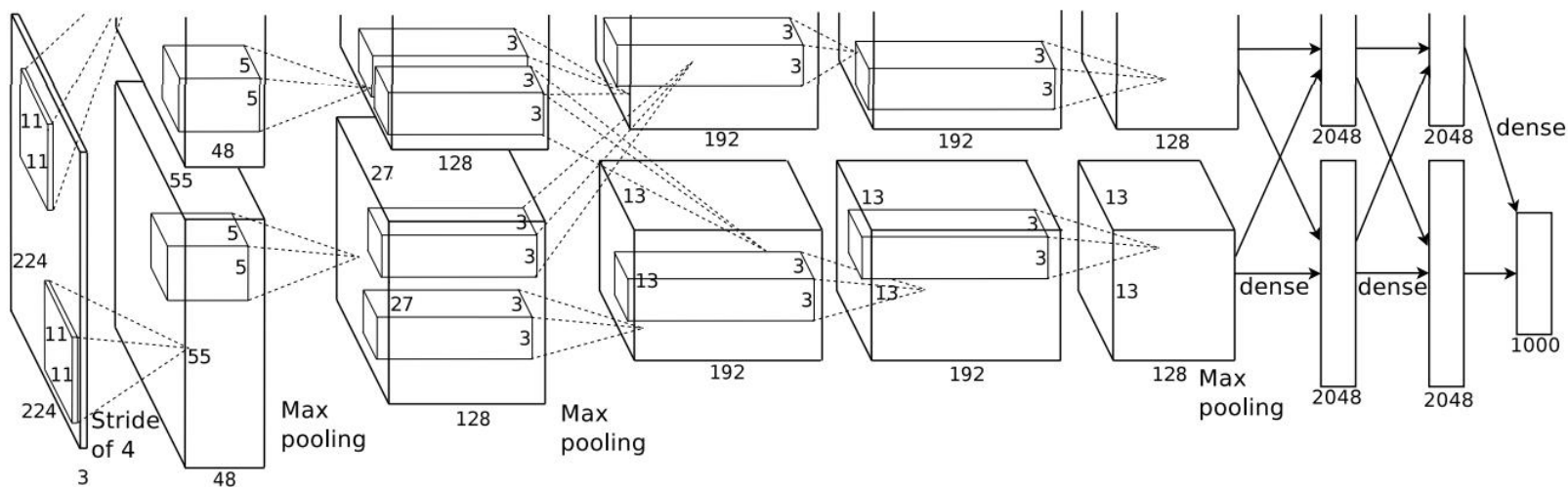
2012 Teams	%error	2013 Teams	%error	2014 Teams	%error
Supervision (Toronto)	15.3	Clarifai (NYU spinoff)	11.7	GoogLeNet	6.6
ISI (Tokyo)	26.1	NUS (singapore)	12.9	VGG (Oxford)	7.3
VGG (Oxford)	26.9	Zeiler-Fergus (NYU)	13.5	MSRA	8.0
XRCE/INRIA	27.0	A. Howard	13.5	A. Howard	8.1
UvA (Amsterdam)	29.6	OverFeat (NYU)	14.1	DeeperVision	9.5
INRIA/LEAR	33.4	UvA (Amsterdam)	14.2	NUS-BST	9.7
		Adobe	15.2	TTIC-ECP	10.2
		VGG (Oxford)	15.2	XYZ	11.2
		VGG (Oxford)	23.0	UvA	12.1



# AlexNet

## ▶ 2012 ILSVRC winner

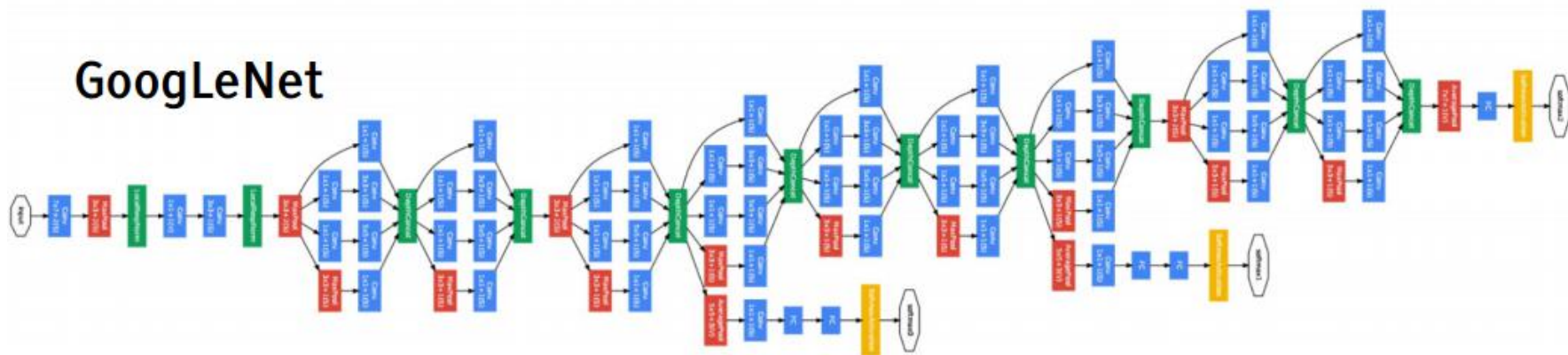
- ▶ (top 5 error of 16% compared to runner-up with 26% error)
- ▶ 第一个现代深度卷积网络模型
  - ▶ 首次使用了很多现代深度卷积网络的一些技术方法
    - 使用GPU进行并行训练，采用了ReLU作为非线性激活函数，使用Dropout防止过拟合，使用数据增强
- ▶ 5个卷积层、3个汇聚层和3个全连接层



# Inception网络

## ▶ 2014 ILSVRC winner (22层)

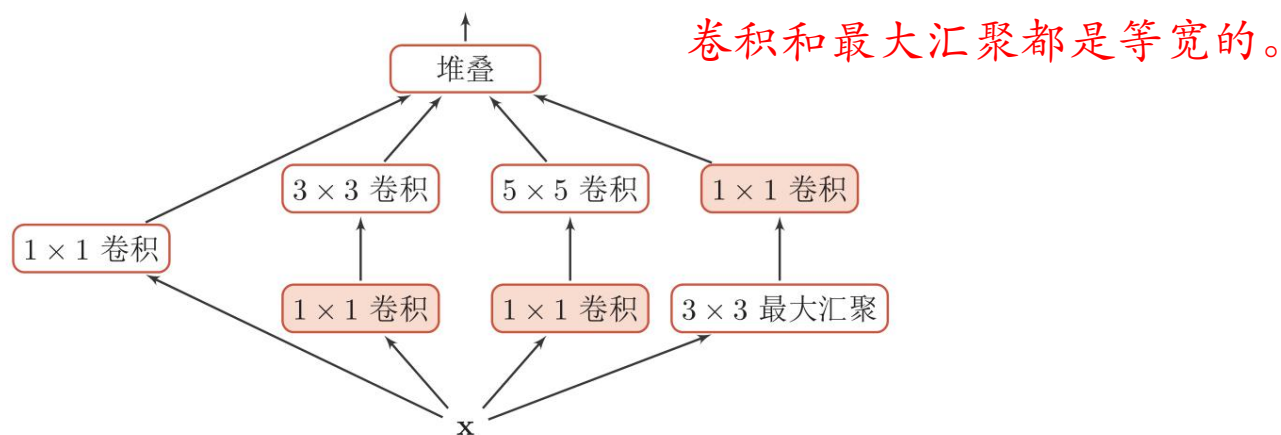
- ▶ 参数: GoogLeNet: 4M VS AlexNet: 60M
- ▶ 错误率: 6.7%
- ▶ Inception网络是由有多个inception模块和少量的汇聚层堆叠而成。





# Inception模块 v1

- ▶ 在卷积网络中，如何设置卷积层的卷积核大小是一个十分关键的问题。
- ▶ 在Inception网络中，一个卷积层包含多个不同大小的卷积操作，称为Inception模块。
- ▶ Inception模块同时使用 $1 \times 1$ 、 $3 \times 3$ 、 $5 \times 5$ 等不同大小的卷积核，并将得到的特征映射在深度上拼接（堆叠）起来作为输出特征映射。



# Inception模块 v3

- ▶ 用多层的小卷积核来替换大的卷积核，以减少计算量和参数量。
- ▶ 使用两层3x3的卷积来替换v1中的5x5的卷积
- ▶ 使用连续的nx1和1xn来替换nxn的卷积。

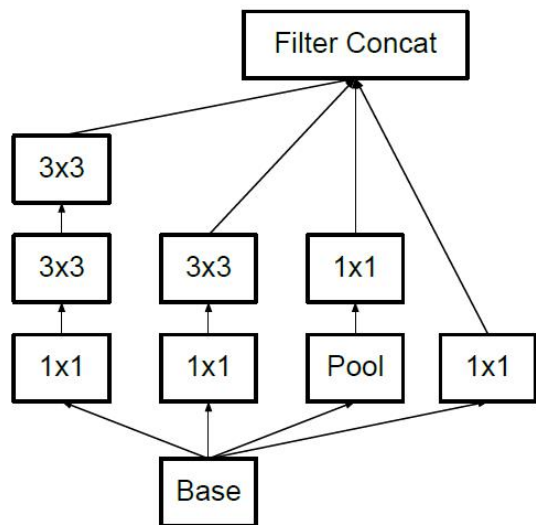


Figure 5. Inception modules where each  $5 \times 5$  convolution is replaced by two  $3 \times 3$  convolution, as suggested by principle [3] of Section 2.

<http://blog.csdn.net/xbinworld>

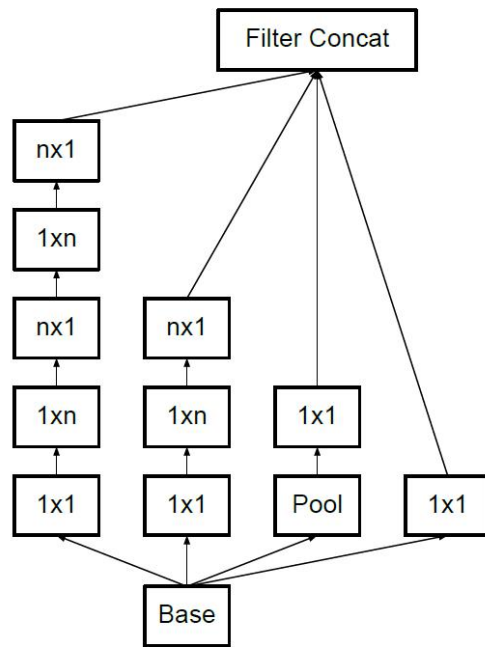


Figure 6. Inception modules after the factorization of the  $n \times n$  convolutions. In our proposed architecture, we chose  $n = 7$  for the  $17 \times 17$  grid. (The filter sizes are picked using principle [3])

<http://blog.csdn.net/xbinworld>

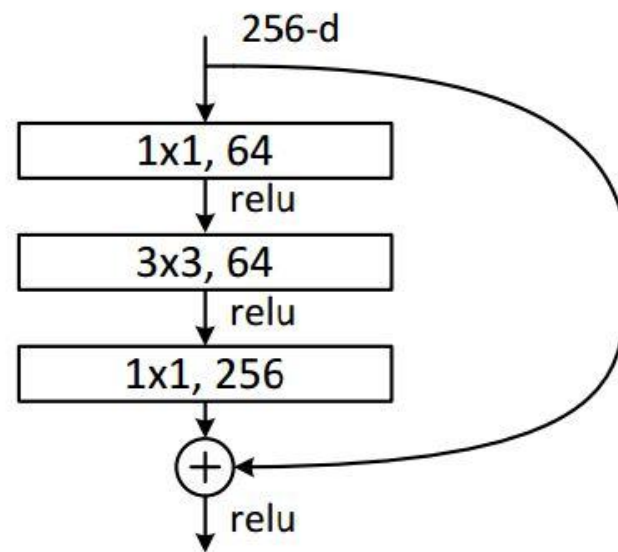
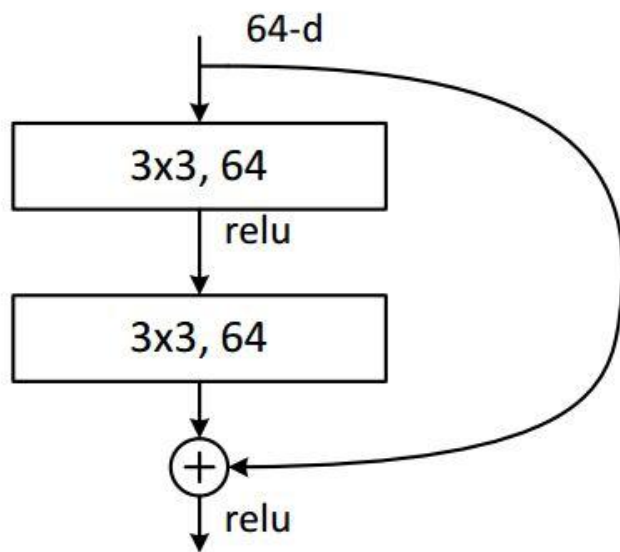
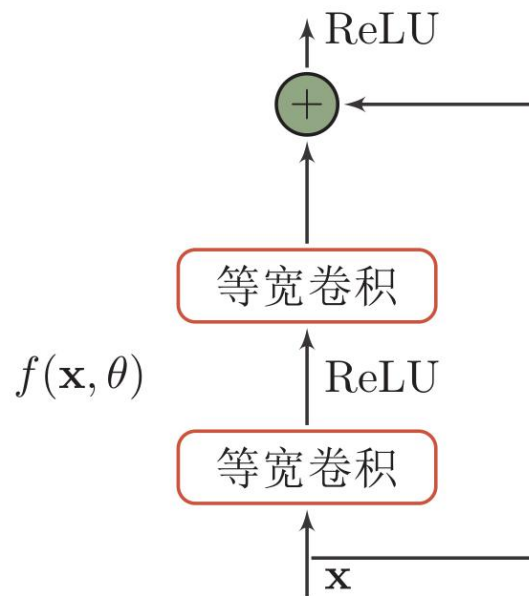


# 残差网络

- ▶ 残差网络 (Residual Network, ResNet) 是通过给非线性的卷积层增加直连边的方式来提高信息的传播效率。
- ▶ 假设在一个深度网络中, 我们期望一个非线性单元 (可以为一层或多层的卷积层)  $f(\mathbf{x}, \theta)$  去逼近一个目标函数为  $h(\mathbf{x})$ 。
- ▶ 将目标函数拆分成两部分: 恒等函数和残差函数

$$h(\mathbf{x}) = \underbrace{\mathbf{x}}_{\text{恒等函数}} + \underbrace{(h(\mathbf{x}) - \mathbf{x})}_{\text{残差函数}} \rightarrow f(\mathbf{x}, \theta)$$

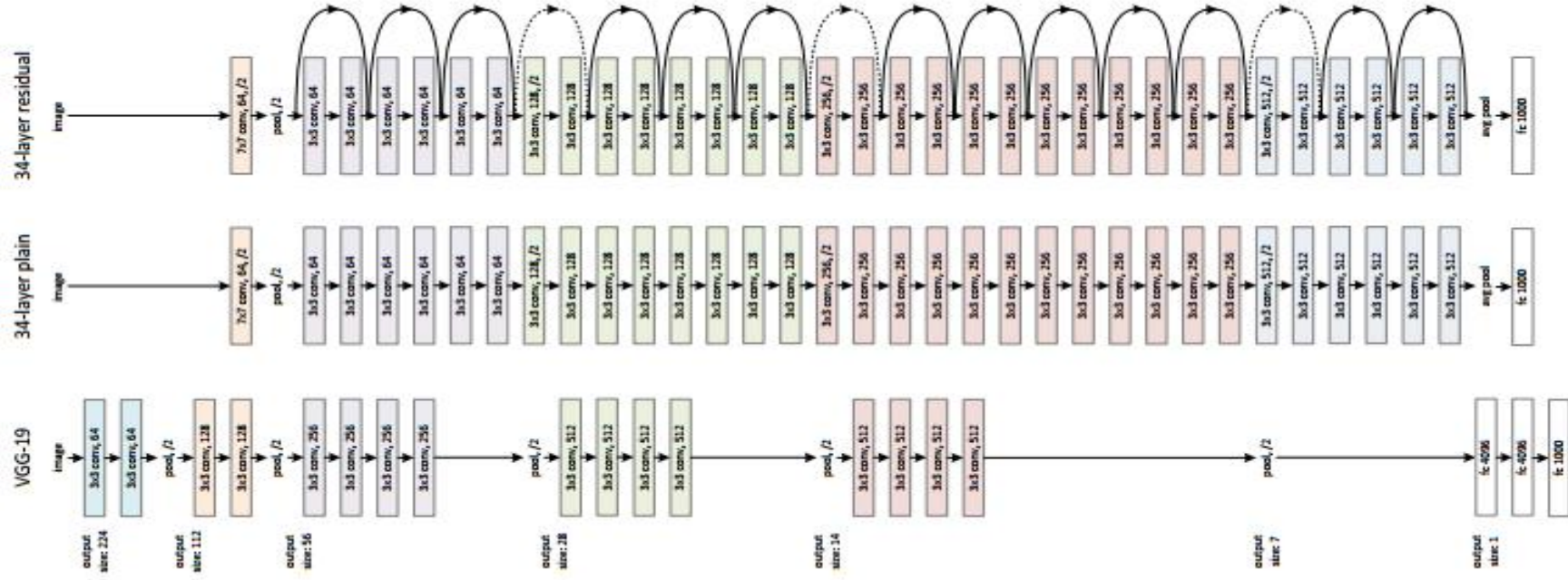
# 残差单元



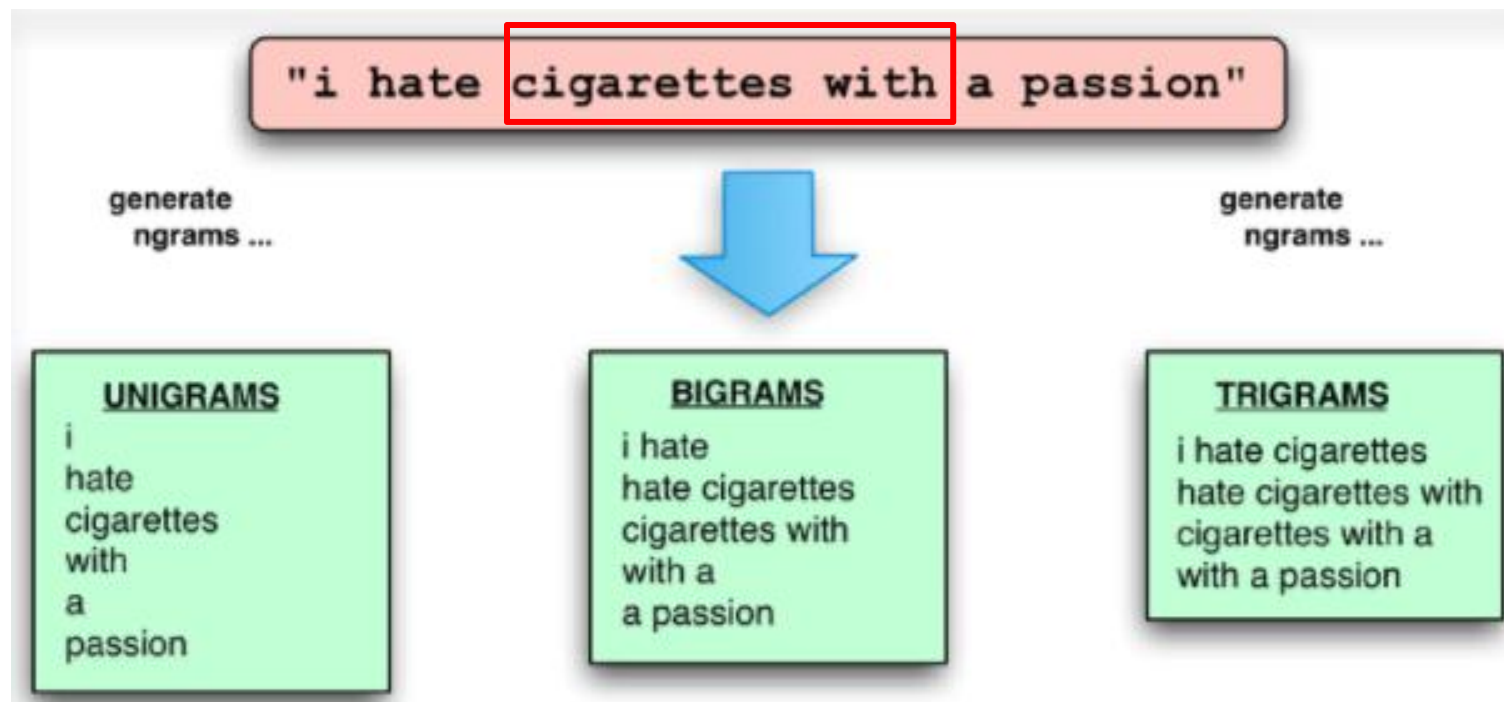
# ResNet

▶ 2015 ILSVRC winner (152层)

▶ 错误率: 3.57%

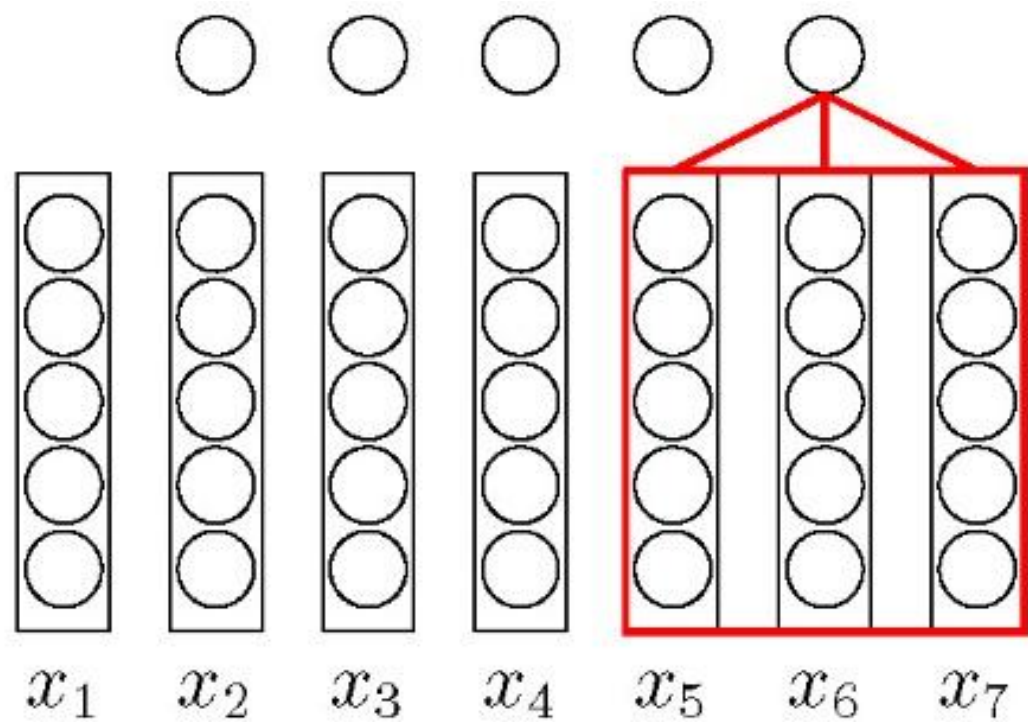


# Ngram特征与卷积

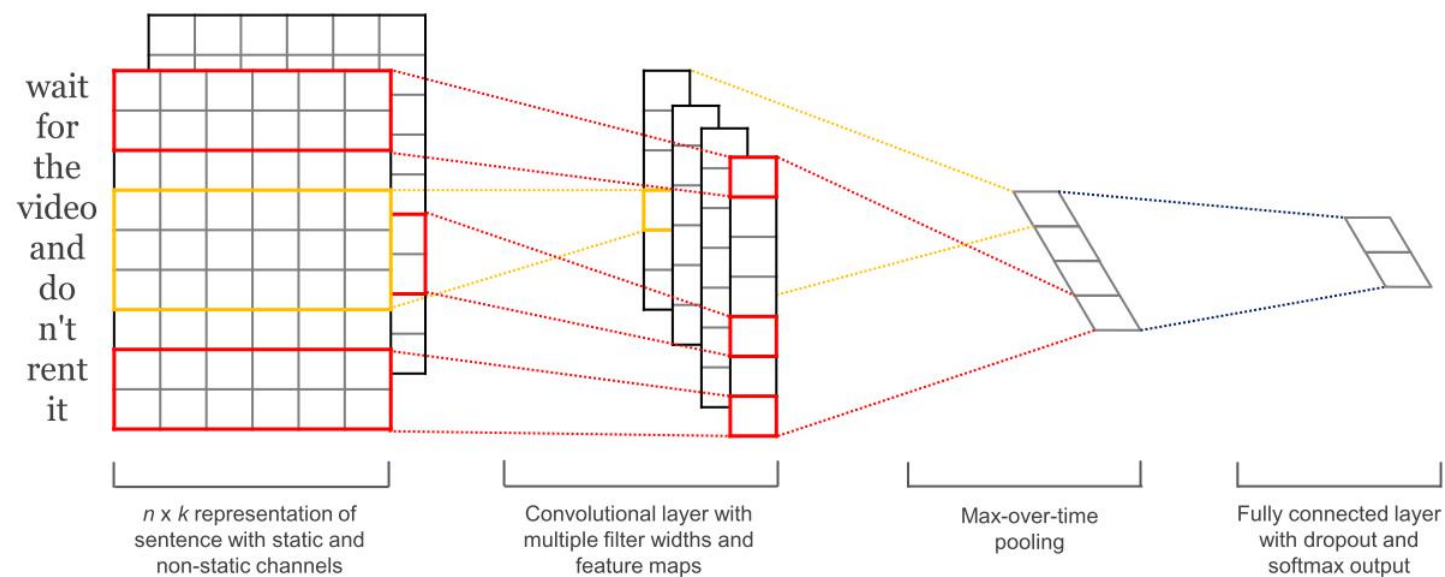


如何用卷积操作来实现?

# 文本序列的卷积

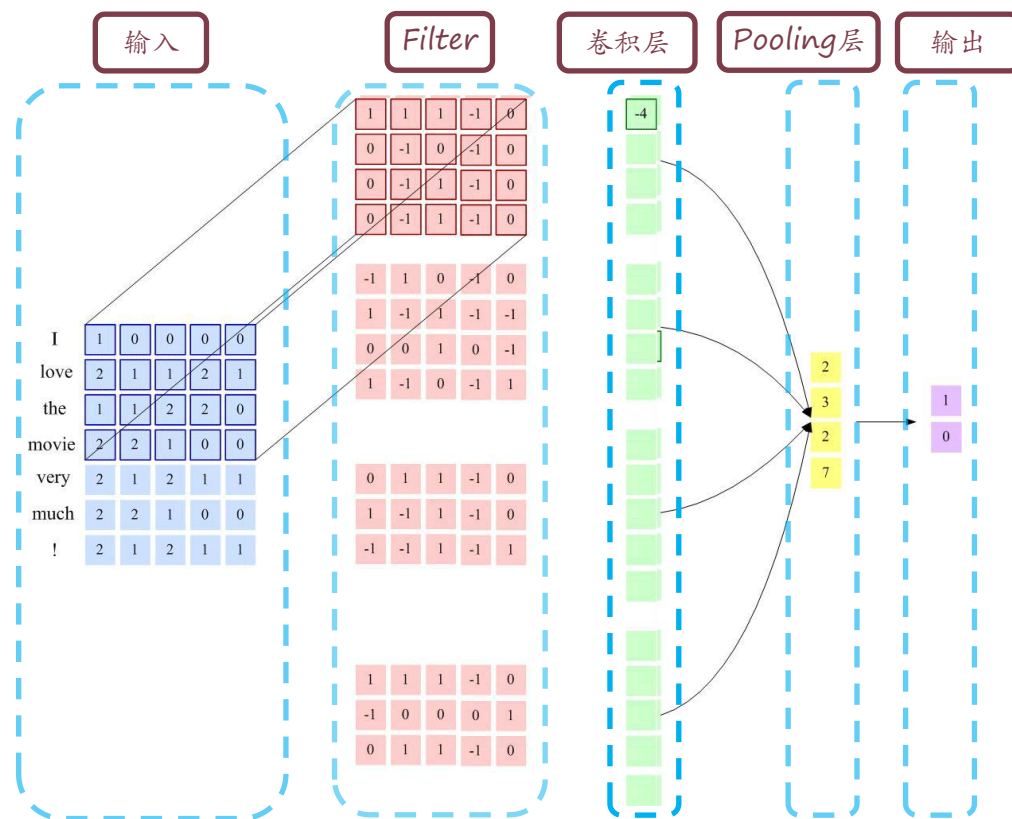


# 基于卷积模型的句子表示



Y. Kim. "Convolutional neural networks for sentence classification" . In: *arXiv preprint arXiv:1408.5882* (2014).

# 文本序列的卷积模型



## CNN 可视化：滤波器

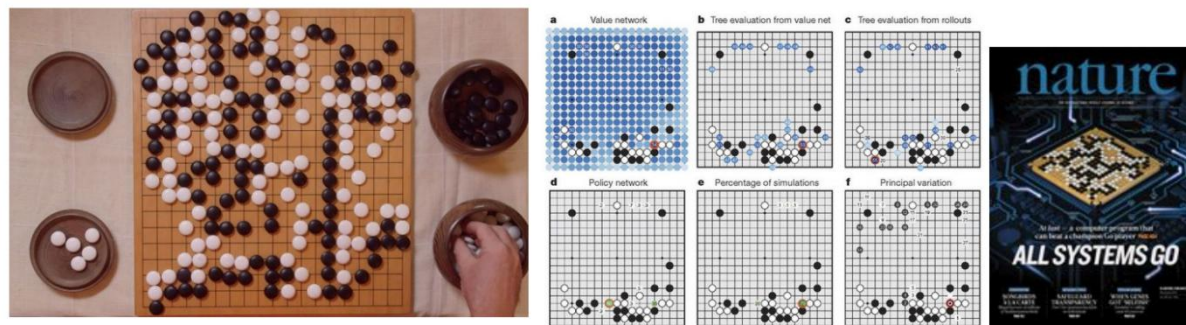
- ▶ AlexNet 中的滤波器 (96 filters [11x11x3])





# 卷积的应用

# AlphaGo



The input to the policy network is a  $19 \times 19 \times 48$  image stack consisting of 48 feature planes. The first hidden layer zero pads the input into a  $23 \times 23$  image, then convolves  $k$  filters of kernel size  $5 \times 5$  with stride 1 with the input image and applies a rectifier nonlinearity. Each of the subsequent hidden layers 2 to 12 zero pads the respective previous hidden layer into a  $21 \times 21$  image, then convolves  $k$  filters of kernel size  $3 \times 3$  with stride 1, again followed by a rectifier nonlinearity. The final layer convolves 1 filter of kernel size  $1 \times 1$  with stride 1, with a different bias for each position, and applies a softmax function. The match version of AlphaGo used  $k = 192$  filters; Fig. 2b and Extended Data Table 3 additionally show the results of training with  $k = 128, 256$  and 384 filters.

## policy network:

[19x19x48] Input

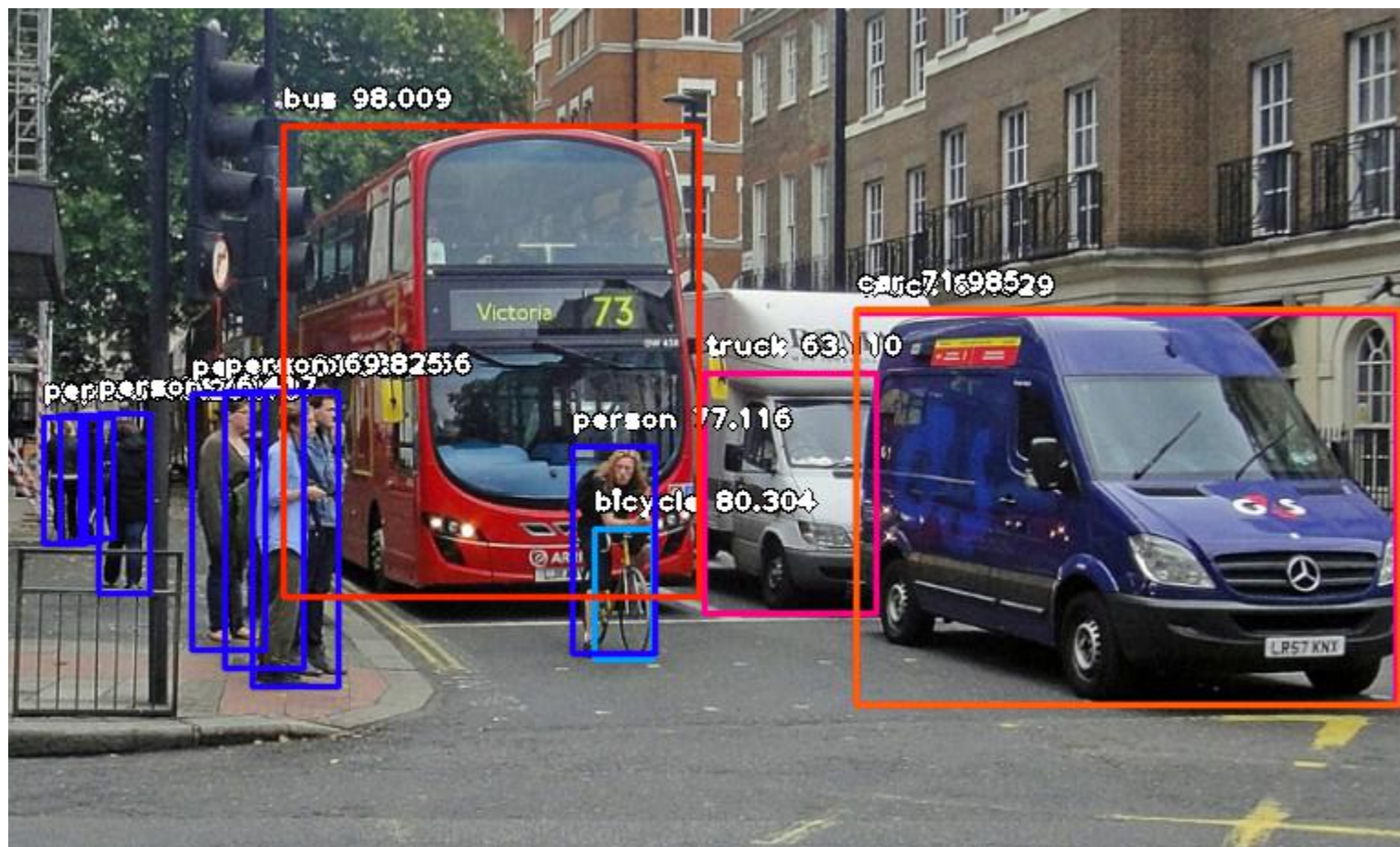
CONV1: 192 5x5 filters , stride 1, pad 2 => [19x19x192]

CONV2..12: 192 3x3 filters, stride 1, pad 1 => [19x19x192]

CONV: 1 1x1 filter, stride 1, pad 0 => [19x19] (*probability map of promising moves*)

- ▶ 分布式系统: 1202 个CPU 和176 块GPU
- ▶ 单机版: 48 个CPU 和8 块GPU
- ▶ 走子速度: 3 毫秒-2 微秒

# 目标检测 (Object Detection)





# Mask RCNN



Figure 4. More results of **Mask R-CNN** on COCO test images, using ResNet-101-FPN and running at 5 fps, with 35.7 mask AP (Table 1).





# 图像生成

	是	在	中	不	了	国	上	有	大	为	年	这	个	出	时	分	人	市	行	到
is	是	在	中	不	了	国	上	有	大	为	年	这	个	出	时	分	人	市	行	到
at	是	在	中	不	了	国	上	有	大	为	年	这	个	出	时	分	人	市	行	到
in	是	在	中	不	了	国	上	有	大	为	年	这	个	出	时	分	人	市	行	到
not	是	在	中	不	了	国	上	有	大	为	年	这	个	出	时	分	人	市	行	到
(past tense)	是	在	中	不	了	国	上	有	大	为	年	这	个	出	时	分	人	市	行	到
country	是	在	中	不	了	国	上	有	大	为	年	这	个	出	时	分	人	市	行	到
on	是	在	中	不	了	国	上	有	大	为	年	这	个	出	时	分	人	市	行	到
have	是	在	中	不	了	国	上	有	大	为	年	这	个	出	时	分	人	市	行	到
large	是	在	中	不	了	国	上	有	大	为	年	这	个	出	时	分	人	市	行	到
for	是	在	中	不	了	国	上	有	大	为	年	这	个	出	时	分	人	市	行	到
year	是	在	中	不	了	国	上	有	大	为	年	这	个	出	时	分	人	市	行	到
this	是	在	中	不	了	国	上	有	大	为	年	这	个	出	时	分	人	市	行	到
(individual)	是	在	中	不	了	国	上	有	大	为	年	这	个	出	时	分	人	市	行	到
out	是	在	中	不	了	国	上	有	大	为	年	这	个	出	时	分	人	市	行	到
time	是	在	中	不	了	国	上	有	大	为	年	这	个	出	时	分	人	市	行	到
minute	是	在	中	不	了	国	上	有	大	为	年	这	个	出	时	分	人	市	行	到
people	是	在	中	不	了	国	上	有	大	为	年	这	个	出	时	分	人	市	行	到
city	是	在	中	不	了	国	上	有	大	为	年	这	个	出	时	分	人	市	行	到
do	是	在	中	不	了	国	上	有	大	为	年	这	个	出	时	分	人	市	行	到
to	是	在	中	不	了	国	上	有	大	为	年	这	个	出	时	分	人	市	行	到

# Deep Dream



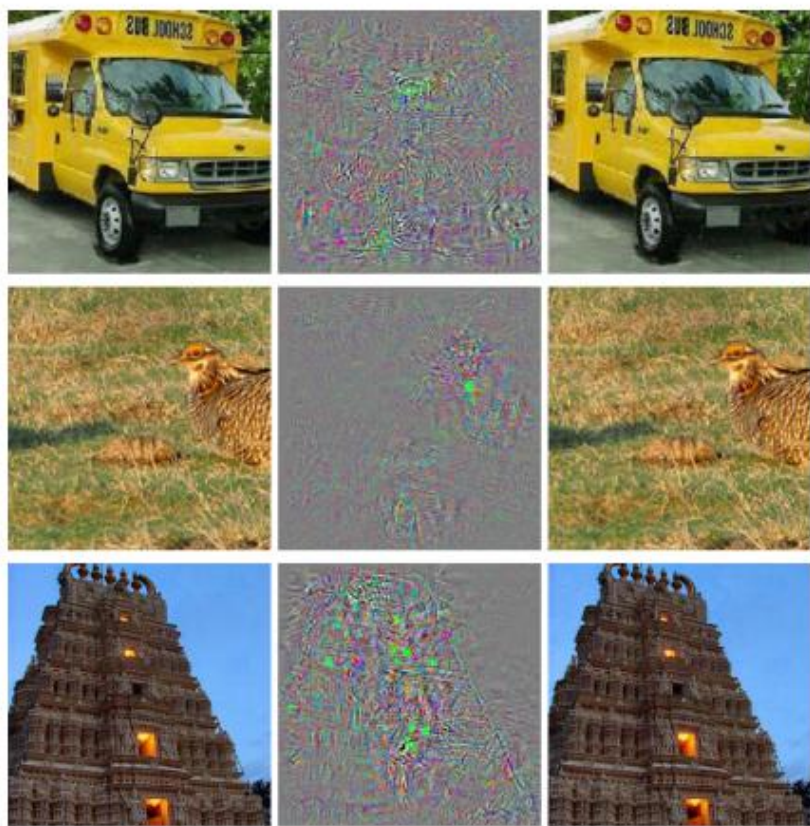


# 画风迁移

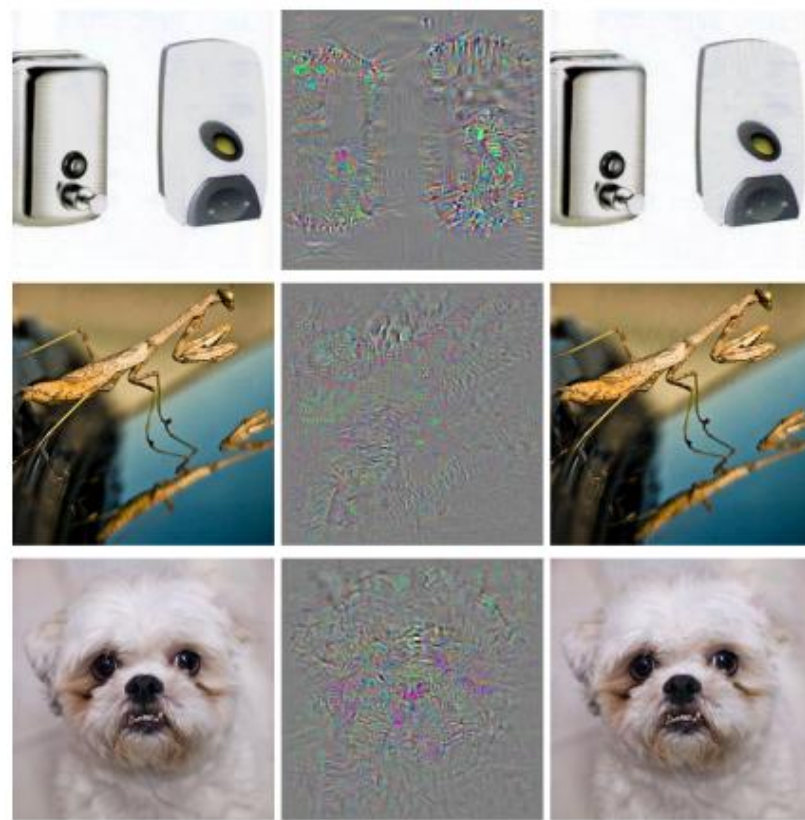




# 对抗样本



(a)



(b)