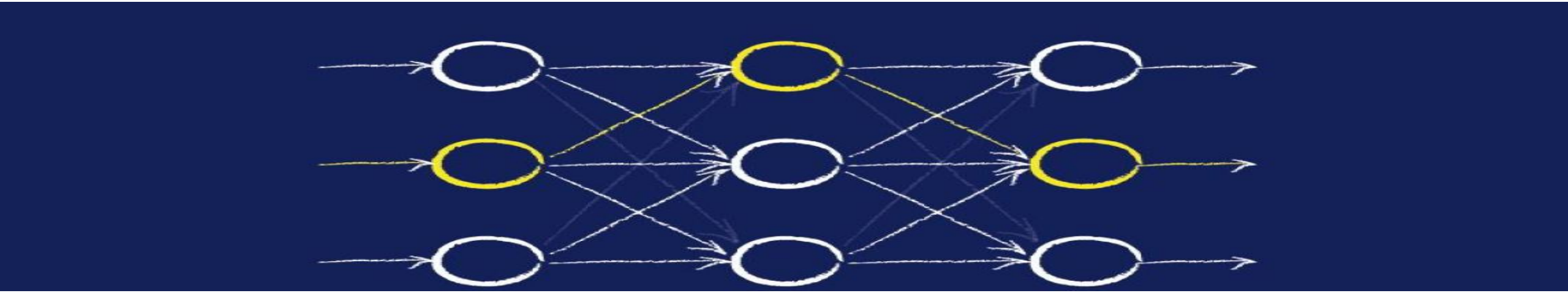
卷积神经网络

（Convolutional neural network，CNN）

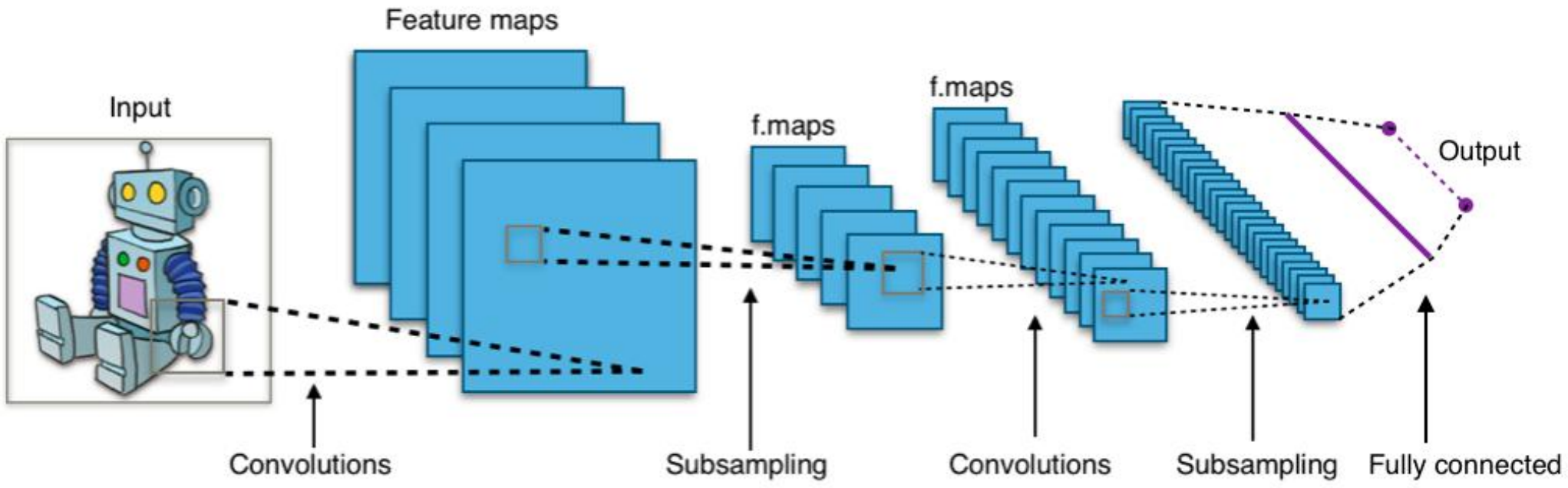


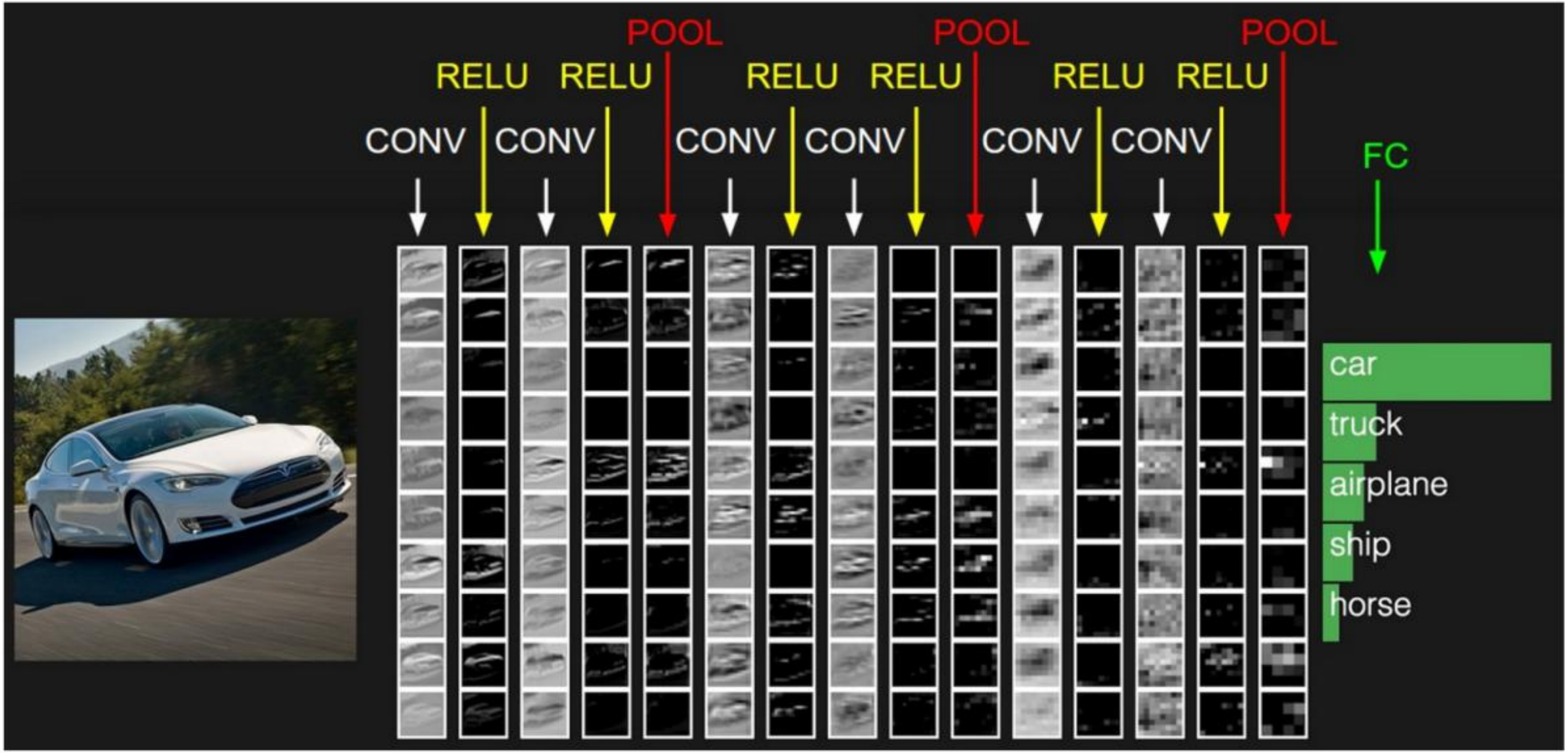
卷积神经网络

- 全连接网络
  - 权重矩阵的参数非常多
- 卷积神经网络（Convolutional Neural Networks，CNN）是一种前馈神经网络。
  - 卷积神经网络是受生物学上感受野（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。

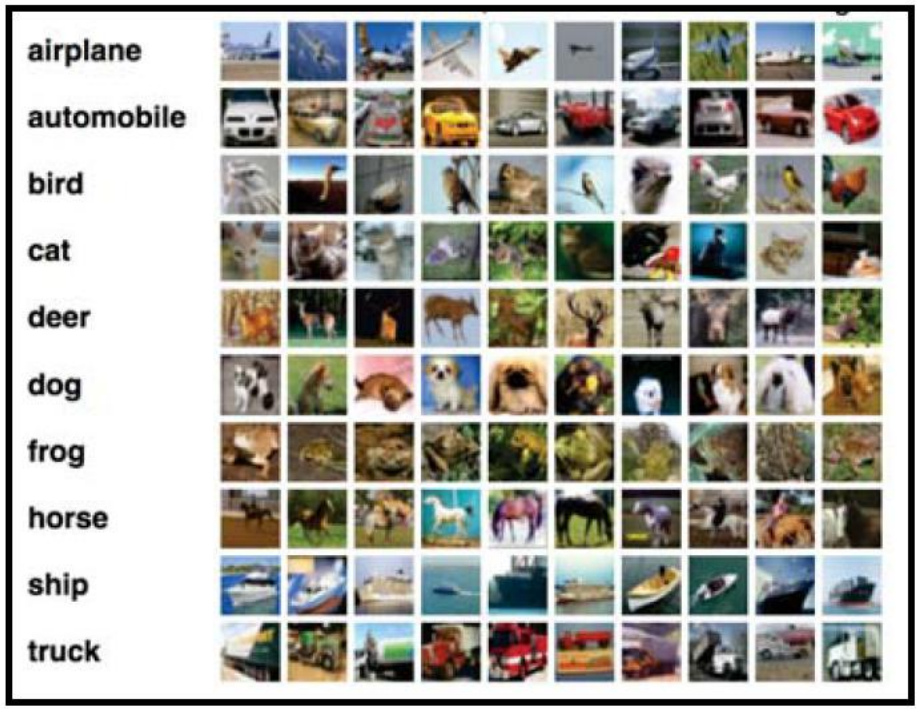
卷积神经网络

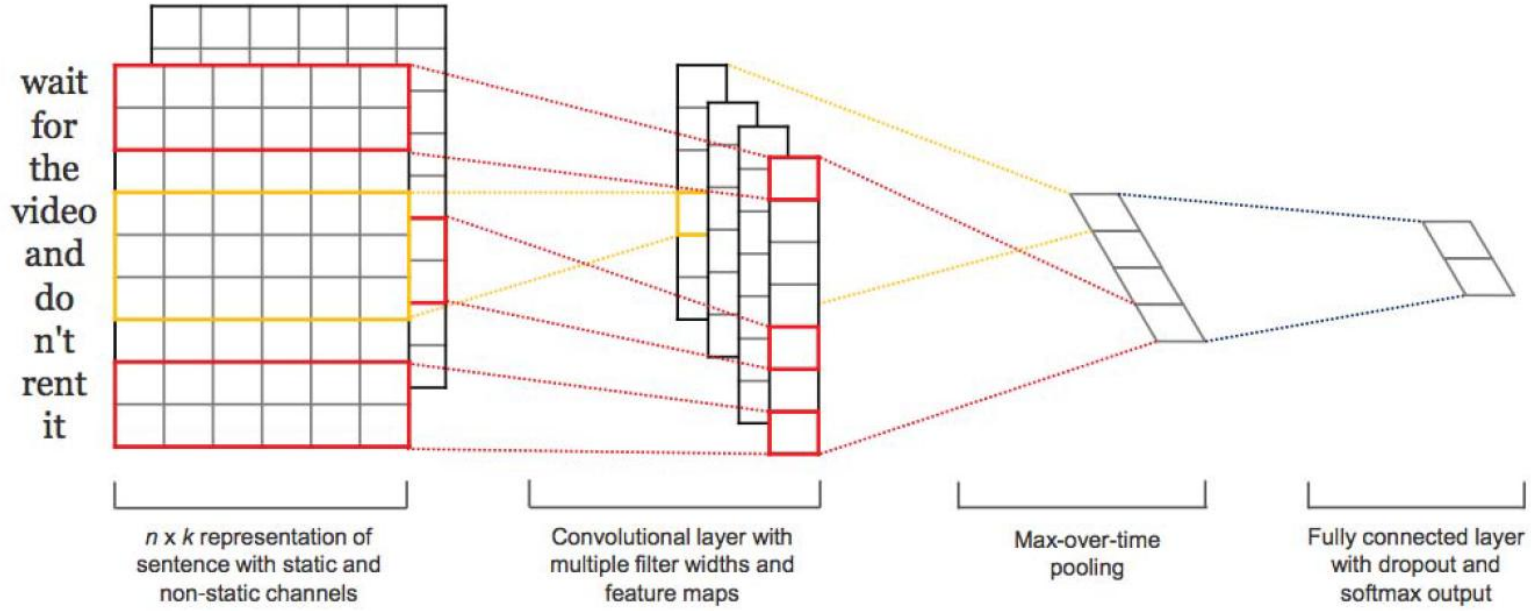
卷积网络是指那些至少在网络的一层中使用卷积运算来替代
一般的矩阵乘法运算的神经网络。




POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

卷积神经网络


airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck


wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

0. 动机

计算机视觉的核心任务之一是图像识别，我们总是在想能不能教会机器来完成图像识别这个任务。人类对于图片的识别相当容易，然而机器却面临了很多问题，如视角变换、光照条件、背景干扰、物体变形等，正是由于这些问题的干扰使得计算机在识别图片的时候准确率总是太低。

如何写一个算法来分类图片呢？和排序不同，分类图片并没有那么简单，我们不可
能自己制定一个规则决定哪张图片属于哪一类，所以需要通过学习算法让机器自己知道如何分类。我们将会给计算机提供每种类别的图片，让机器自己去学习其中的特征并形成一个算法，这就是机器学习的核心。这些算法是依赖于数据集的，所以也称为数据驱动的算法。

0. 动机

在卷积神经网络流行起来之前，图像处理使用的都是一些传统的方法。比如提取图像中的边缘、纹理、线条、边界等特征，依据这些特征再进行下一步的处理，这样的处理不仅效率特别低，准确率也不高。随着计算机视觉的快速发展，如今在某些图像集上机器的识别准确率已经超过了人类，这一切都要归功于卷积神经网络。

1. 卷积（convolution）

假设我们正在用激光传感器追踪一艘宇宙飞船的位置。我们的激光传感器给出一个单独的输出 $x(t)$，表示宇宙飞船在时刻 t 的位置。x 和 t 都是实值的，这意味着我们可以在任意时刻从传感器中读出飞船的位置。

现在假设我们的传感器受到一定程度的噪声干扰。为了得到飞船位置的低噪声估计，我们对得到的测量结果进行平均。可以 采用一个加权函数 $w(a)$ 来实现，其中 a 表示测量结果距当前时刻的时间间隔。如果 对任意时刻都采用这种加权平均的操作，就得到了一个新的对于飞船位置的平滑估计函数 s：

$$s(t) = \int x(a)w(t-a)da.$$

这种运算就叫做 **卷积**（convolution）。卷积运算通常用星号表示：

$$s(t) = (x * w)(t).$$

如果我们假设 x 和 w 都定义在整数时刻 t 上，就可以定义离散形式的卷积：

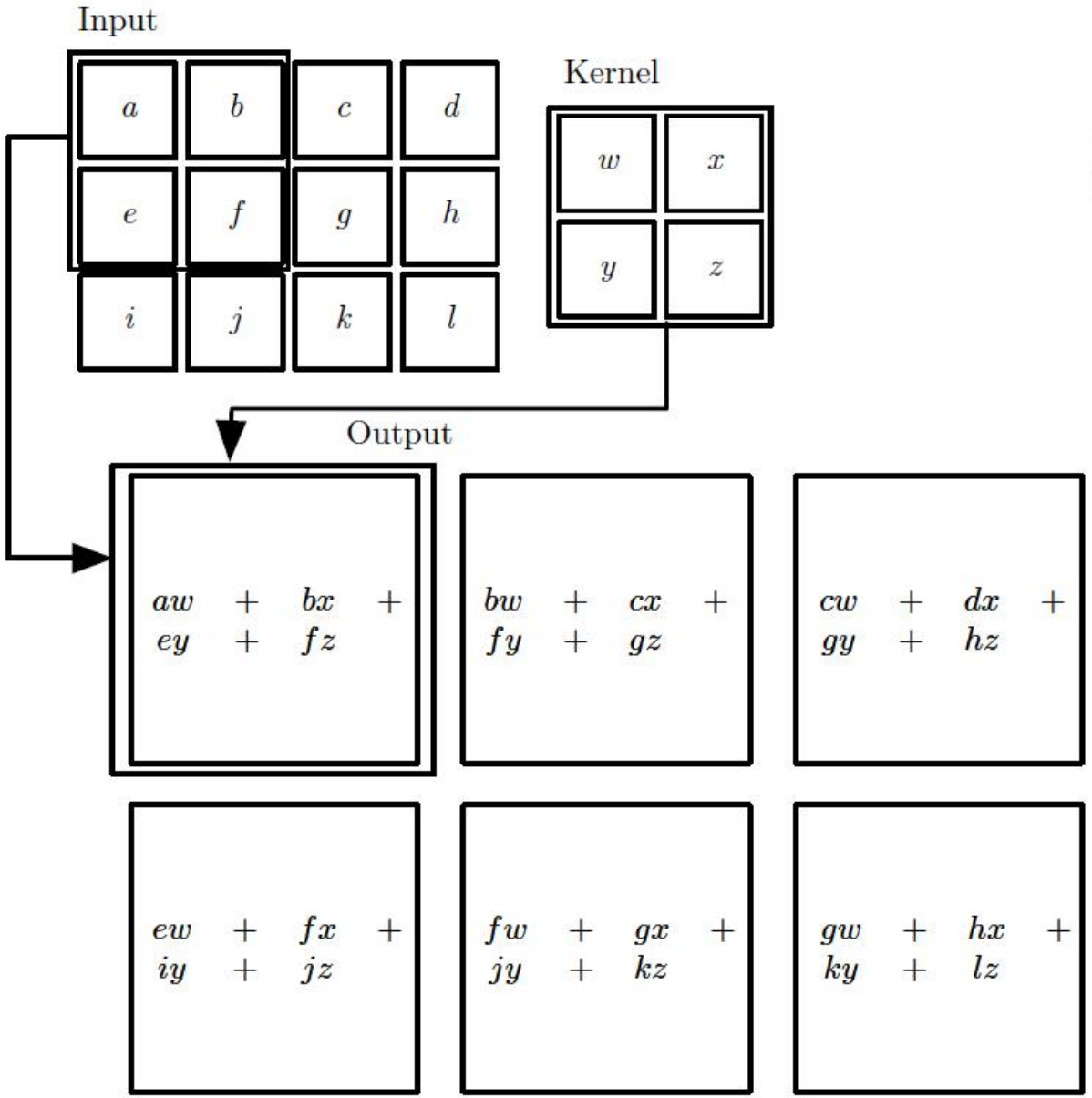
$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a).$$

二维

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n).$$

卷积是可交换的 (commutative)，我们可以等价地写作：

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n).$$



I

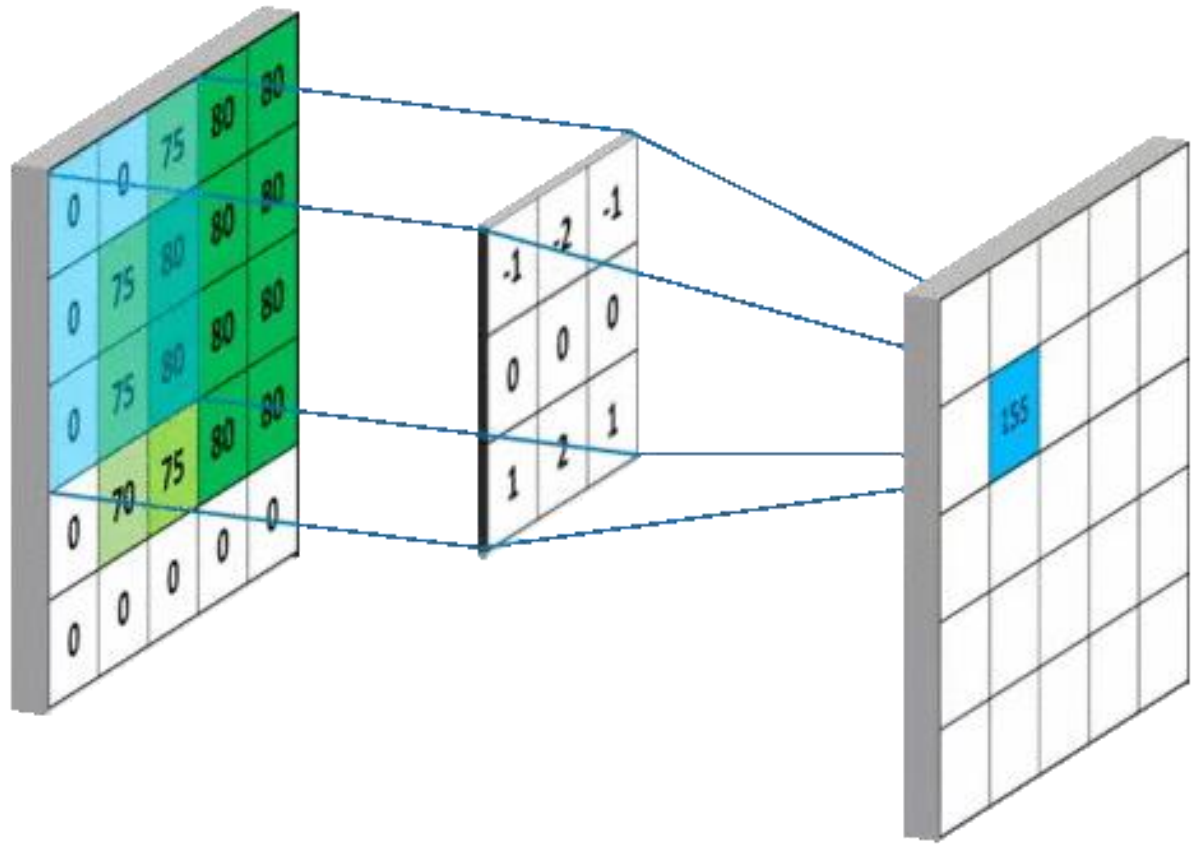
1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

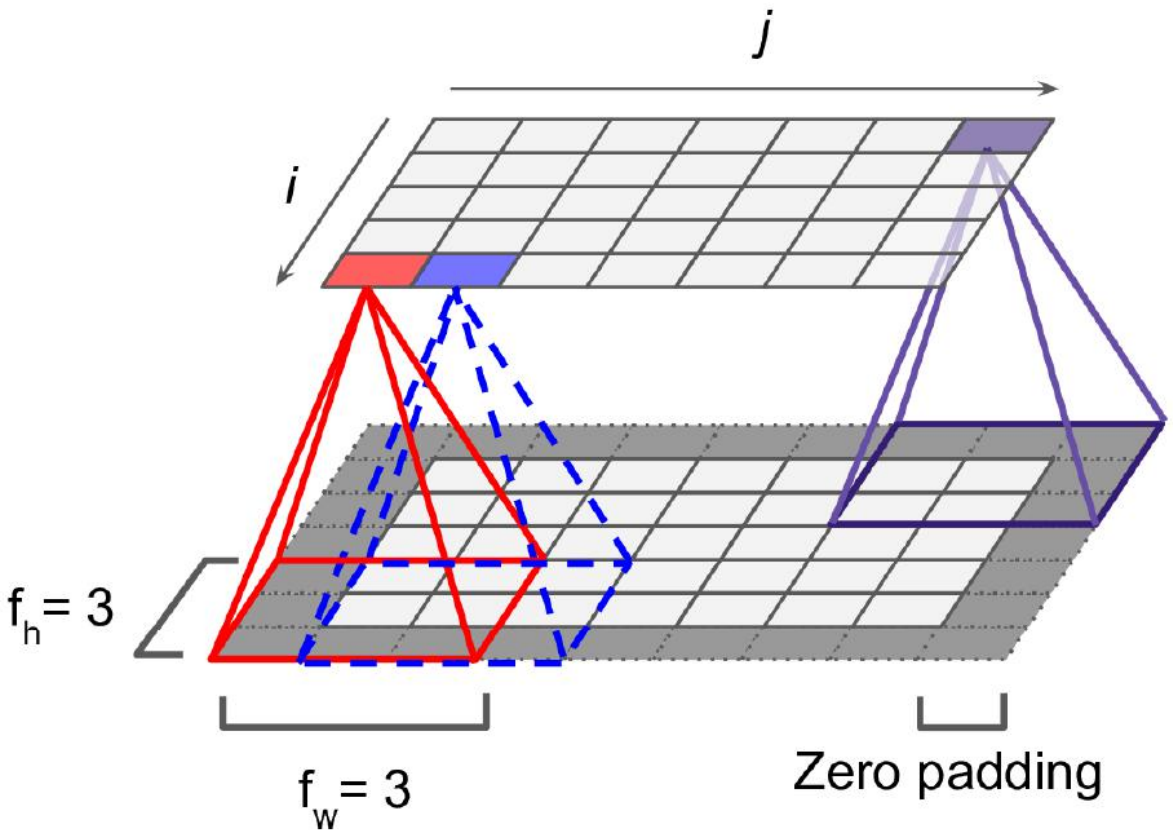
K

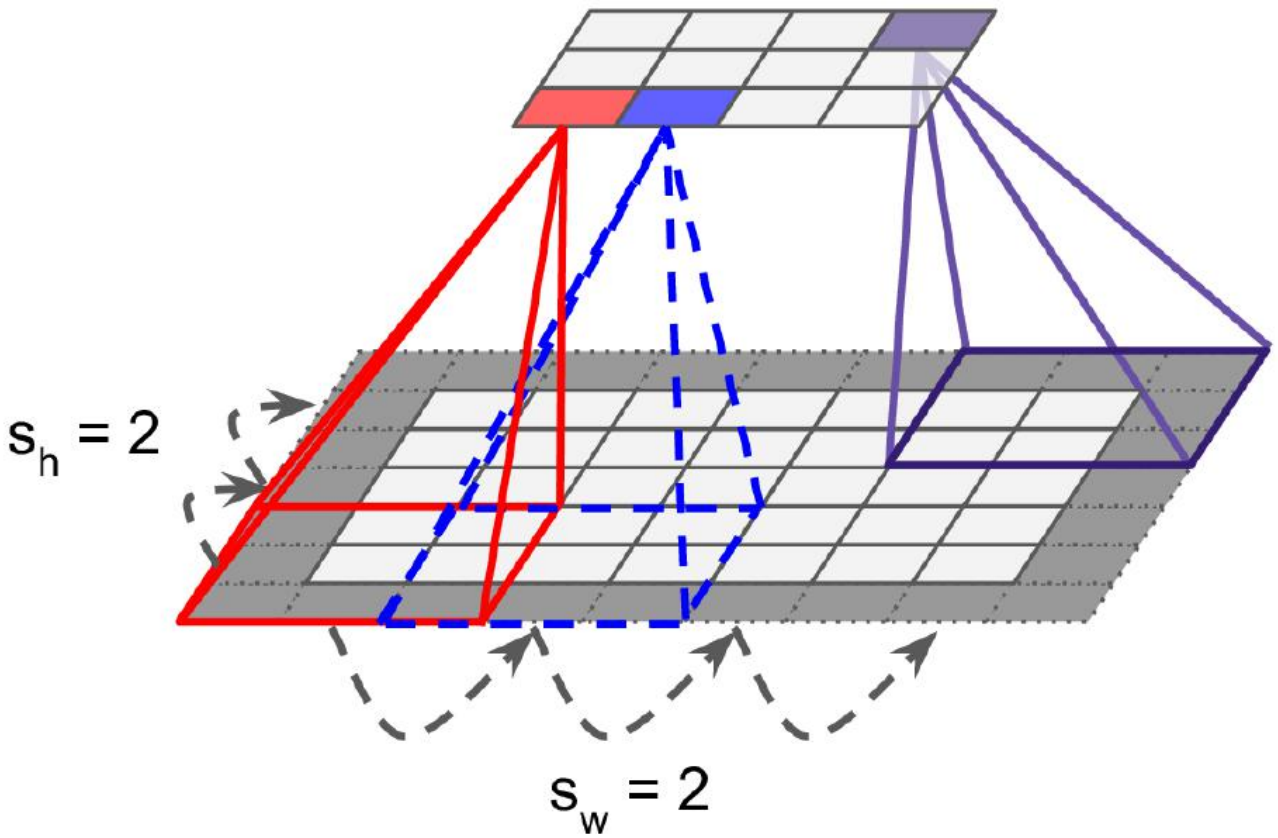
1	0	1
0	1	0
1	0	1

Convolved

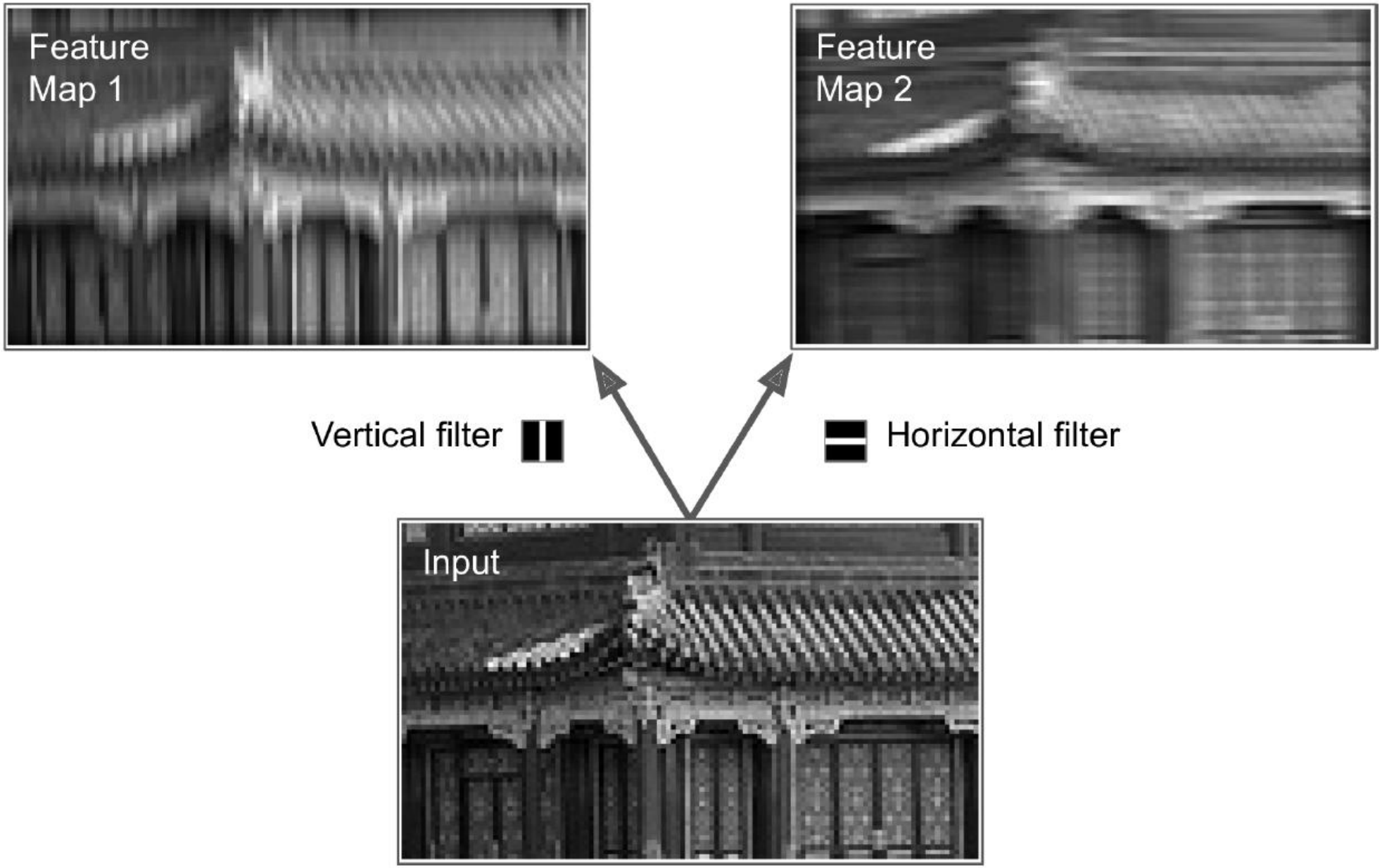
4	3	4
2	4	3
2	3	4


0 0 75 80 80
0 75 80 80 80
0 75 80 80 80
0 75 80 80 80
0 70 75 80 80
0 0 0 0 0
0 0 0 0 0
-1 -2 -1
0 0 0
1 2 1
155

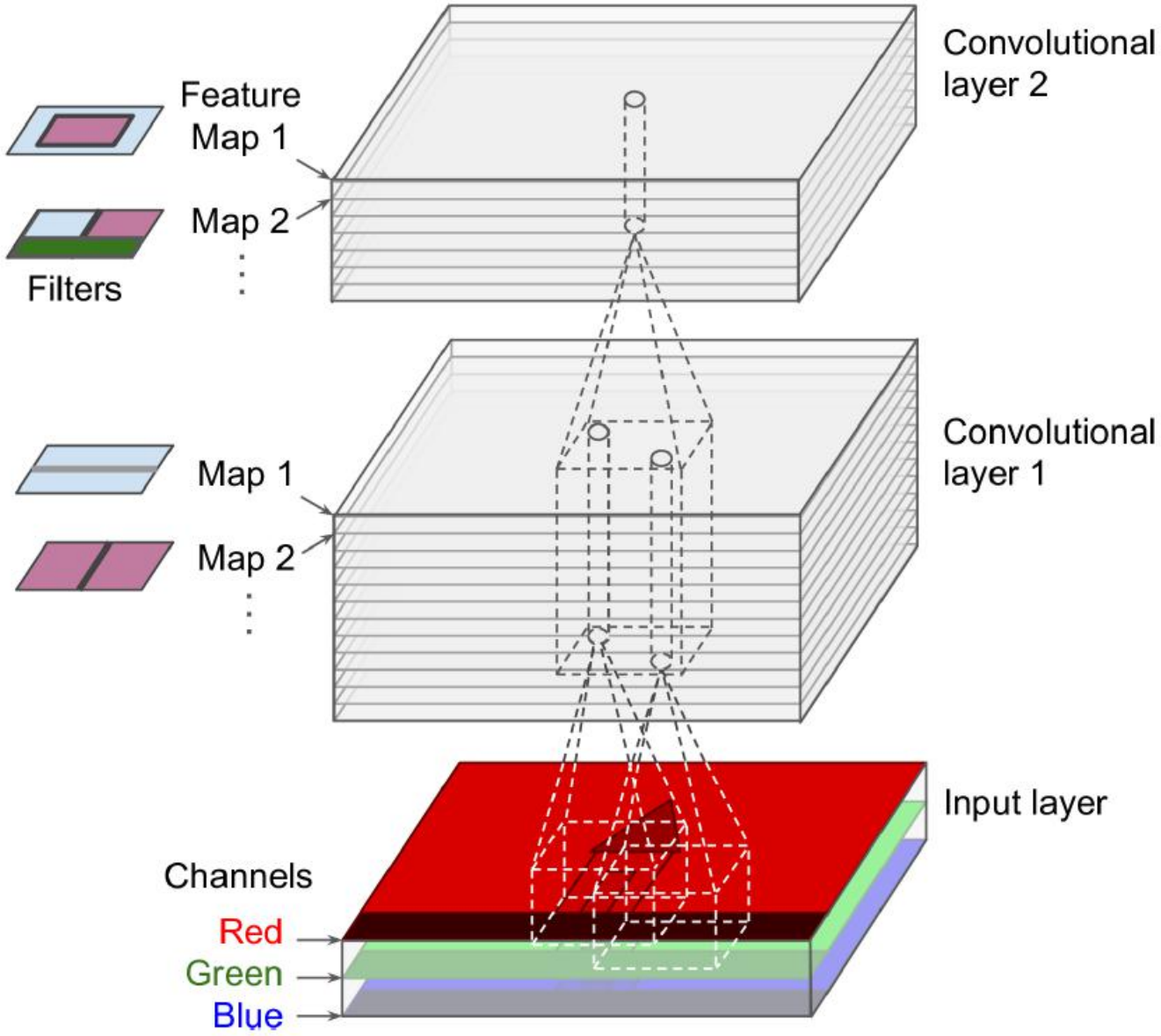




滤波器、滑动窗口的大小和填补的类型是在网络训练期间能精调的超参数



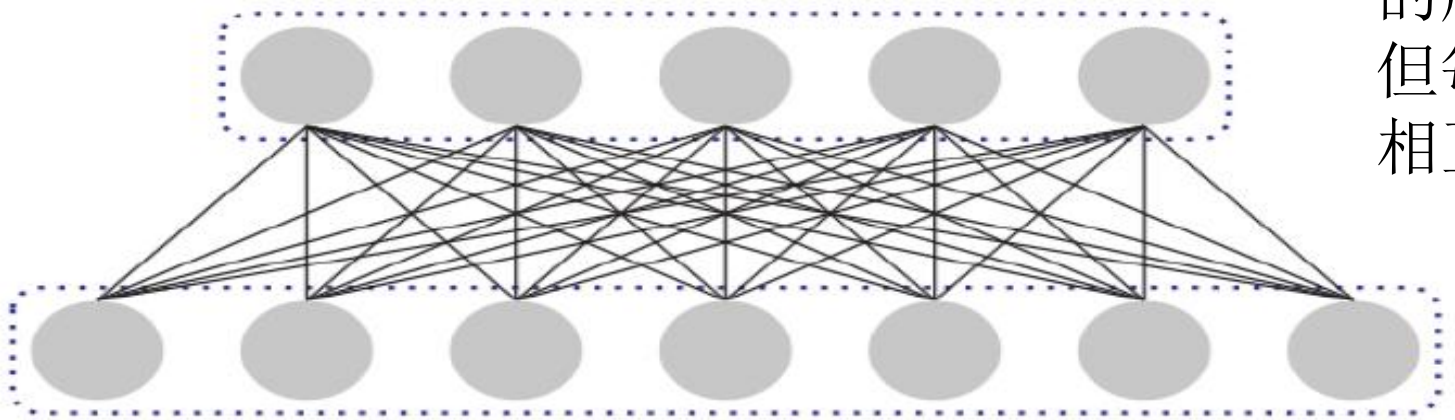
应用两个不同的滤波器得到两个特征图



多个特征图和三颜色通道图像的卷积层

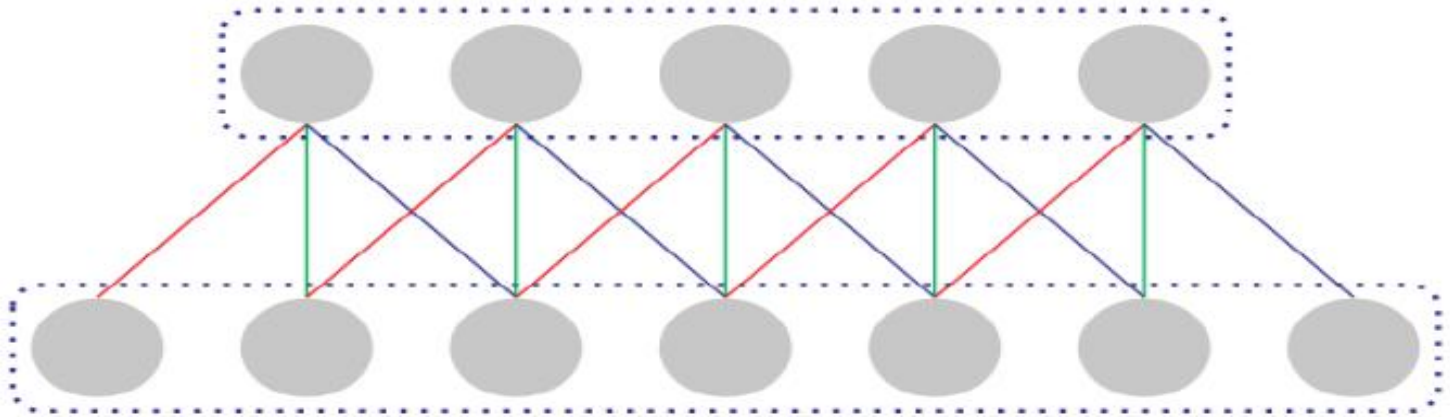
卷积神经网络

- 用卷积层代替全连接层



(a) 全连接层

每个神经元都和前一层的所有神经元相关联，但每一层中的神经元相互独立。



(b) 卷积层

稀疏连接
参数共享
平移等变

卷积层

首先介绍卷积神经网络的参数。这些参数是由一些可学习的滤波器集合构成的，每个滤波器在空间上（宽度和高度）都比较小，但是深度和输入数据的深度保持一致。举例来说，卷积神经网络的第一层卷积一个典型的滤波器的尺寸可以是5 x 5 x 3（宽和高都是5），或者是3x3x3（宽和高都是3），这里的宽度和高度可以、任意定义，但是深度必须是3，因为深度要和输入一致，而输入的图片是3通道的。在前向传播的时候，让每个滤波器都在输入数据的宽度和高度上滑动（卷积），然后计算整个滤波器和输人数据任意一处的内积。

卷积层

当滤波器沿着输入数据的宽度和高度滑动时，会生成一个二维的激活图，激活图上的每个空间位置表示了原图片对于该滤波器的反应。直观来看，网络会让滤波器学习到当它看到某些类型的视觉特征的时候就激活，具体的视觉特征可以是边界、颜色、轮廓、甚至可以是网络更高层上的蜂巢状或者车轮状图案。

在每个卷积层上，会有一整个集合的滤波器，比如20个，这样就会形成20张二维的、
不同的激活图，将这些激活图在深度方向上层叠起来就形成了卷积层的输出。

如果用大脑和生物神经元做比喻，那么输出的3D数据中的每个数据都可以看成是神经元的一个输出，而该神经元只是观察输入数据中的一种特征，并且和空间上左右两边的所有神经元共享参数（因为这些输出都是使用同一个滤波器得到的结果）。

卷积层

- 1. 局部连接
- 因为图片特征具有局部性，所以只需要通过局部就能提前出相应的特征。
- 与神经元连接的空间大小叫做神经元的感受野（receptive field），大小是超参数，就是滤波器的宽和高。

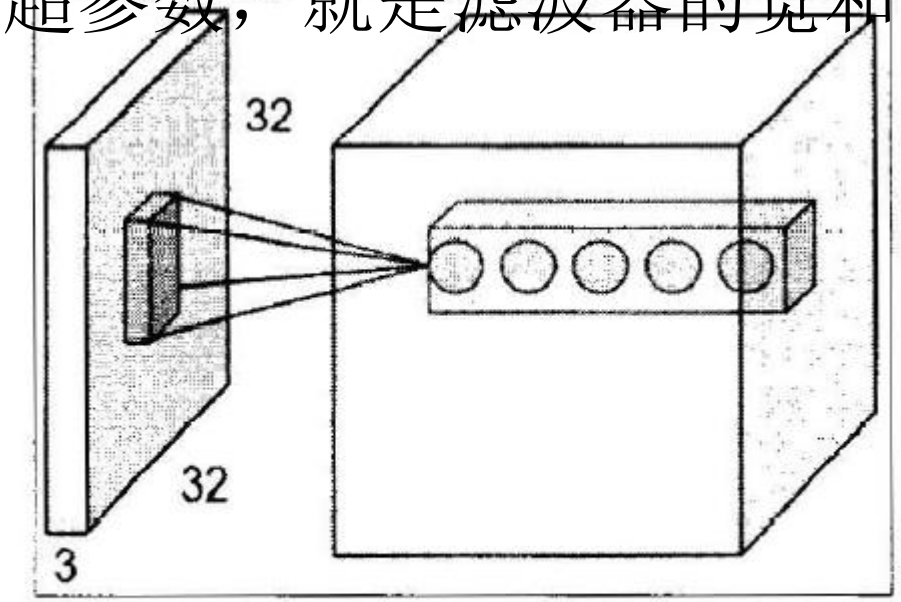


图 4.6　感受野

卷积层

- 2. 空间排列
- A. 卷积层的输出深度是一个超参数，与使用的滤波器数量一致，每种滤波器寻找一种特征，
- B. 滑动滤波器时，必须指定步长。
- C. 边界填充，可以在边界0值填充，填充尺寸作为超参数。

卷积层

输出的尺寸到底是多少呢？其实可以用一个公式来计算，就是 $\frac{W-F+2P}{S}+1$，其中 W 表示输入的数据大小，F 表示卷积层中神经元的感受野尺寸，S 表示步长，P 表示边界填充 0 的数量。比如输入是 7×7，滤波器是 3×3，步长是 1，填充的数量是 0，那么根据公式，就能得到 $\frac{7-3+2\times0}{1}+1=5$，即输出的空间大小是 5×5，如果步长是 2，那么 $\frac{7-3+2\times0}{2}+1=3$，输出的空间大小就是 3×3。可以用图 4.7所示的这个一维的例子来具体说明。

右上角表示神经网络的权重，其中输入数据的大小为 5，感受野的大小为 3；左边表示滑动步长为 1，且填充也为 1；右边表示滑动步长为 2，填充为 1。

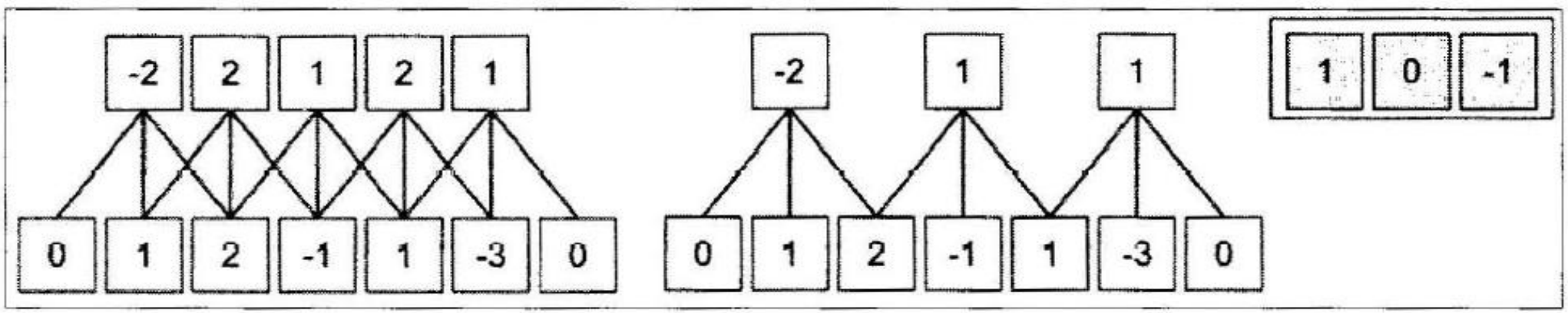


图 4.7　一维空间

卷积层

最后总结一下卷积层的一些性质。

（1）输入数据体的尺寸是 $W_1 \times H_1 \times D_1$。

（2）4 个超参数：滤波器数量 K，滤波器空间尺寸 F，滑动步长 S，零填充的数量 P。

（3）输出数据体的尺寸为 $W_2 \times H_2 \times D_2$，其中 $W_2 = \frac{W_1 - F + 2P}{S} + 1$，$H_2 = \frac{H_1 - F + 2P}{S} + 1$，$D_2 = K$。

（4）由于参数共享，每个滤波器包含的权重数目为 $F \times F \times D_1$，卷积层一共有 $F \times F \times D_1 \times K$ 个权重和 K 个偏置。

（5）在输出体数据中，第 d 个深度切片（空间尺寸是 $W_2 \times H_2$），用第 d 个滤波器和输入数据进行有效卷积运算的结果，再加上第 d 个偏置。

对于卷积神经网络的一些超参数，常见的设置是 $F = 3$，$S = 1$，$P = 1$，同时这

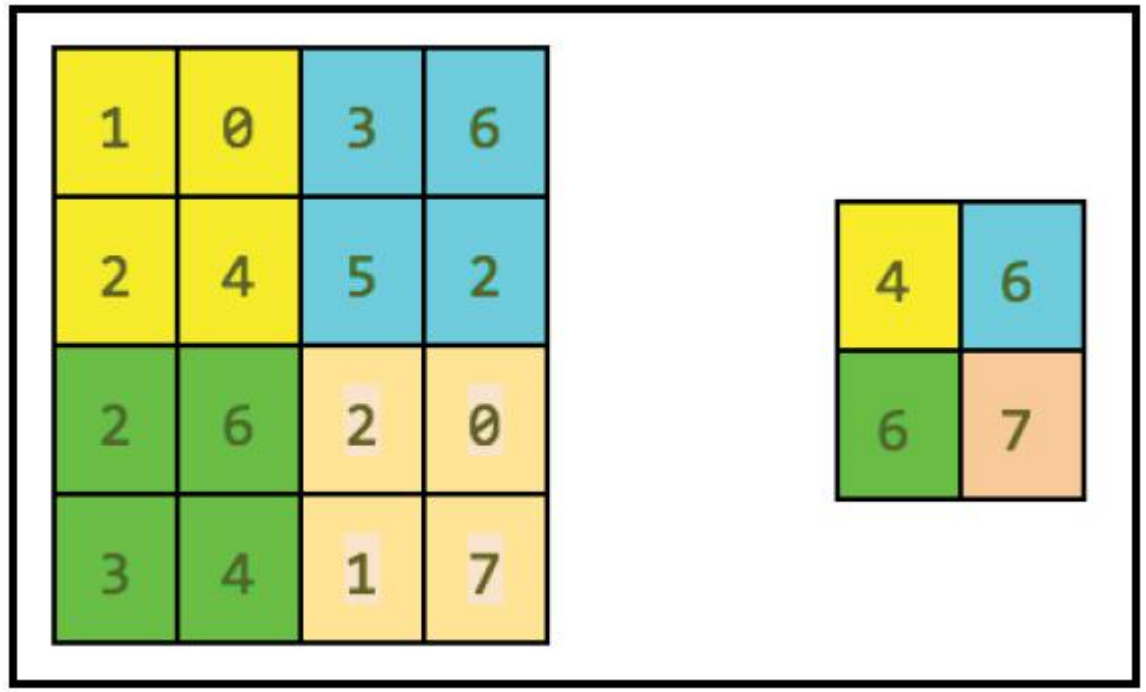
2.池化（pooling）

池化函数使用某一位置的相邻输出的总体统计特征来代替网络在该位置的输出。例如，**最大池化**（max pooling）函数 (Zhou and Chellappa, 1988) 给出相邻矩形区域内的最大值。其他常用的池化函数包括相邻矩形区域内的平均值、L^2 范数以及基于据中心像素距离的加权平均函数。

不管采用什么样的池化函数，当输入作出少量平移时，池化能够帮助输入的表示近似**不变**（invariant）。

使用池化可以看作是增加了一个无限强的先验：这一层学得的函数必须具有对少量平移的不变性。当这个假设成立时，池化可以极大地提高网络的统计效率。

通常会在卷积层之间周期性插入一个池化层，其作用是逐渐降低数据体的空间尺寸，这样就能够减少网络中参数的数量，减少计算资源耗费，同时也能够有效地控制过拟合。



池化层之所以有效，是因为之前介绍的图片特征具有不变性，也就是通过下采样不会丢失图片拥有的特征，由于这种特性，我们可以将图片缩小再进行卷积处理，这样能够大大降低卷积运算的时间。

下面先来介绍到底什么是池化层。池化层和卷积层一样也有一个空间窗口，通常采用的是取这些窗口中的最大值作为输出结果，然后不断滑动窗口，对输入数据体每一个深度切片单独处理，减少它的空间尺寸，如图4.8所示。

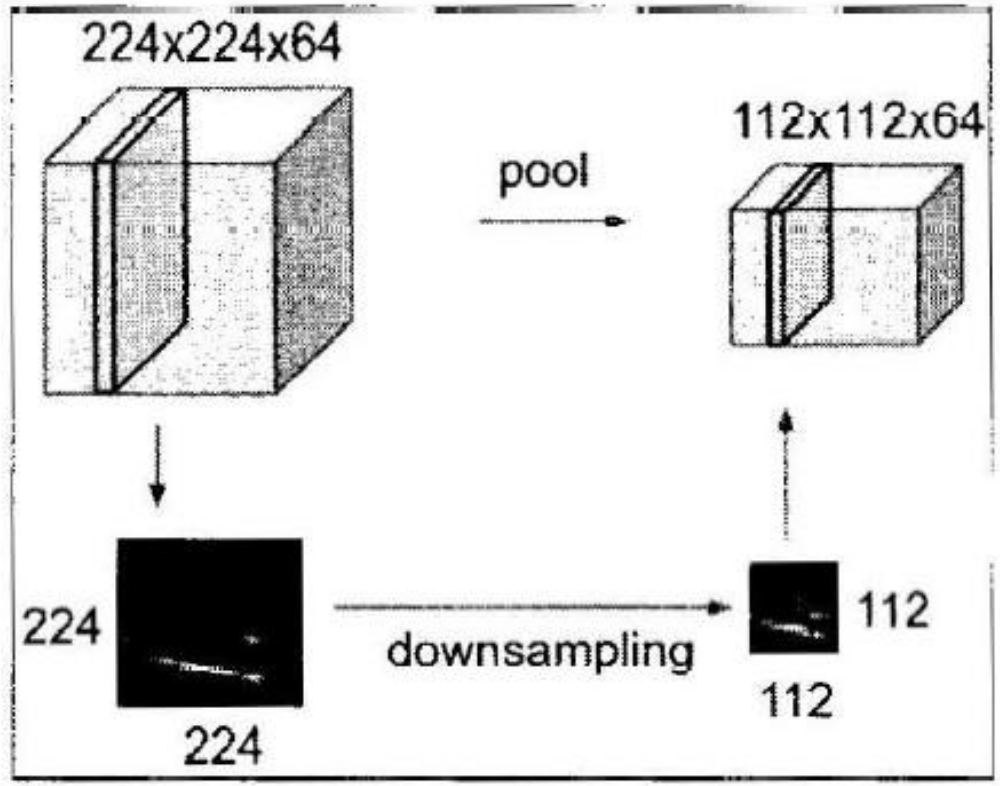
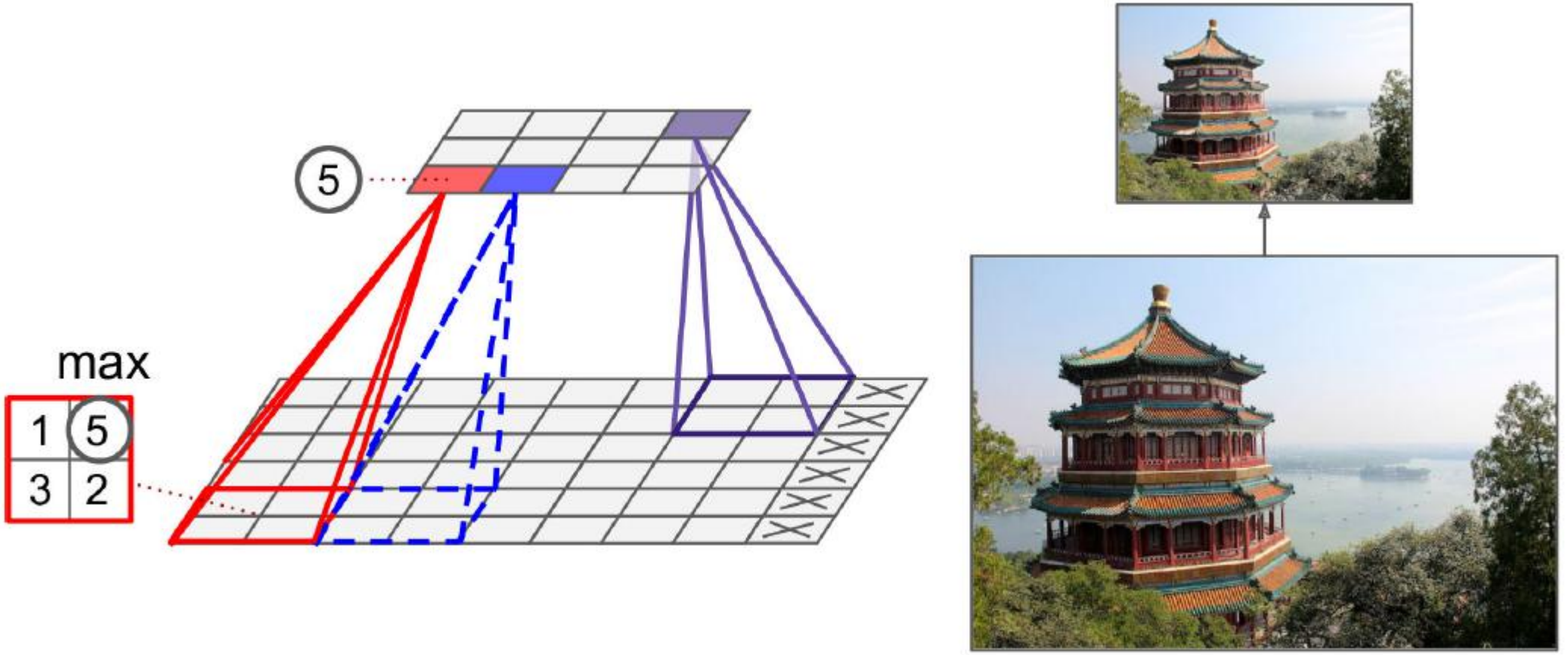


图 4.8　池化层处理效果



最大池化层（2 × 2 池化核，滑动 2，无填充）

最常用的池化层形式是尺寸为 2×2 的窗口，滑动步长为 2，对图像进行下采样，将其中 75% 的激活信息都丢掉，选择其中最大的保留下来，这其实是因为我们希望能够更加激活里面的数值大的特征，去除一些噪声信息。

池化层有一些和卷积层类似的性质。

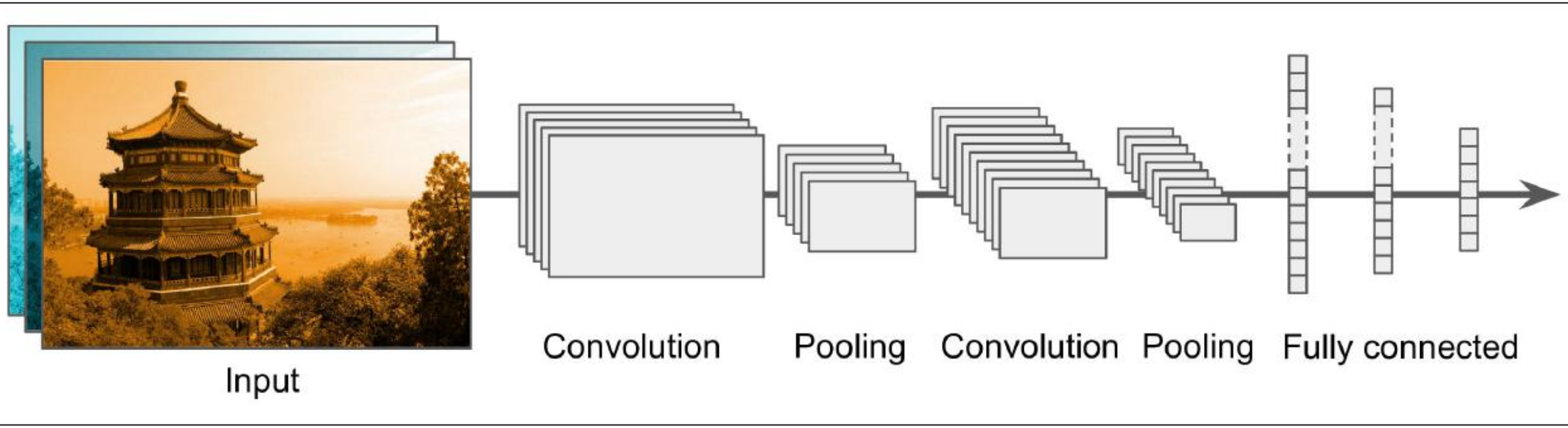
（1）输入数据体的尺寸是 $W_1 \times H_1 \times D_1$。

（2）有两个需要设置的超参数，空间大小 F 和滑动步长 S。

（3）输出数据体的尺寸是 $W_2 \times H_2 \times D_2$，其中 $W_2 = \frac{W_1 - F}{S} + 1, H_2 = \frac{H_1 - F}{S} + 1, D_2 = D_1$。

（4）对输入进行固定函数的计算，没有参数引入。

（5）池化层中很少引入零填充。


Input
Convolution
Pooling
Convolution
Pooling
Fully connected

典型的 CNN结构

conv2d 是二维度卷积，对数据在宽度和高度两个维度上进行卷积。

函数定义：

```
torch.nn.functional.conv2d(input, weight, bias=None, stride=1, padding=0, d
```

参数说明：

- input：输入的Tensor数据，格式为 (batch,channels,H，W)，四维数组，第一维度是样本数量，第二维度是通道数或者记录数，三、四维度是高度和宽度。
- weight：卷积核权重，也就是卷积核本身，是一个四维度数组，(out_channels, in_channels/groups, kH, kW)。 Out_channels 是卷积核输出层的神经元个数，也就是这层有多少个卷积核； in_channels 是输入通道数，kH 和 kW 是卷积核的高度和宽度。
- bias：位移参数，可选项，一般不用管。
- stride：滑动窗口，默认为 1，指每次卷积对原数据滑动 1 个单元格。
- padding：是否对输入数据填充 0。Padding 可以将输入数据的区域改造成卷积核大小的整数倍，这样对不满足卷积核大小的部分数据就不会忽略了。通过 padding 参数指定填充区域的高度和宽度，默认 0（就是填充区域为0，不填充的意思）。
- dilation：卷积核之间的空格，默认1。
- groups：将输入数据分组，通常不用管这个参数，没有太大意义。

max_pool2d 方法的说明如下：

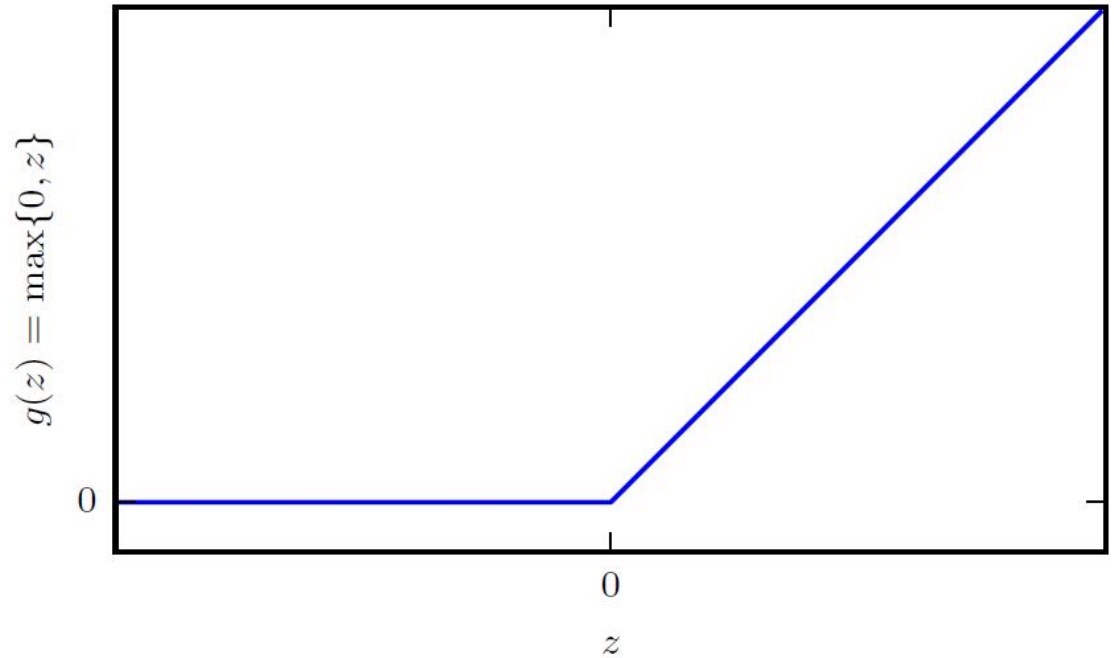
```
torch.nn.functional.max_pool2d(input, kernel_size, stride=None, padding=0,
```

那么具体的各个参数的含义说明如下：

- input：输入的 Tensor 数据，格式为 (channels,H，W)，三维数组，第一维度是通道数或者记录数，二、三维度是高度和宽度。
- kernel_size：池化区的大小，指定宽度和高度 (kh x kw)，如果只有一个值则表示宽度和高度相同。
- stride：滑动窗口，默认和 kernel_size 相同值，这样在池化的时候就不会重叠。如果设置的比 kernel_size 小，则池化时会重叠。它也是高度和宽度两个值。
- padding：是否对输入在左前端填充 0。池化时，如果剩余的区域不够池化区大小，则会丢弃掉。Padding 可以将输入数据的区域改造成是池化核的整数倍，这样就不会丢弃掉原始数据了。Padding 也是指定填充区域的高度和宽度，默认 0（就是填充区域为 0，不填充的意思）。
- ceil_mode：在计算输出 shape 大小时按照 ceil 还有 floor 计算，是数序函数（如ceil(4.5)=5;floor(4.5)=4）。
- count_include_pad：为 True 时，在求平均时会包含 0 填充区域的大小。这个参数只有在 avg_pool2d 并且 padding 参数不为 0 时才有意义。

In [4]:

```
# Convolutional network building
network = input_data(shape=[None, 32, 32, 3],
                     data_preprocessing=img_prep,
                     data_augmentation=img_aug)
network = conv_2d(network, 32, 3, activation='relu')
network = max_pool_2d(network, 2)
network = conv_2d(network, 64, 3, activation='relu')
network = conv_2d(network, 64, 3, activation='relu')
network = max_pool_2d(network, 2)
network = fully_connected(network, 512, activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 10, activation='softmax')
network = regression(network, optimizer='adam',
                     loss='categorical_crossentropy',
                     learning_rate=0.001)
```



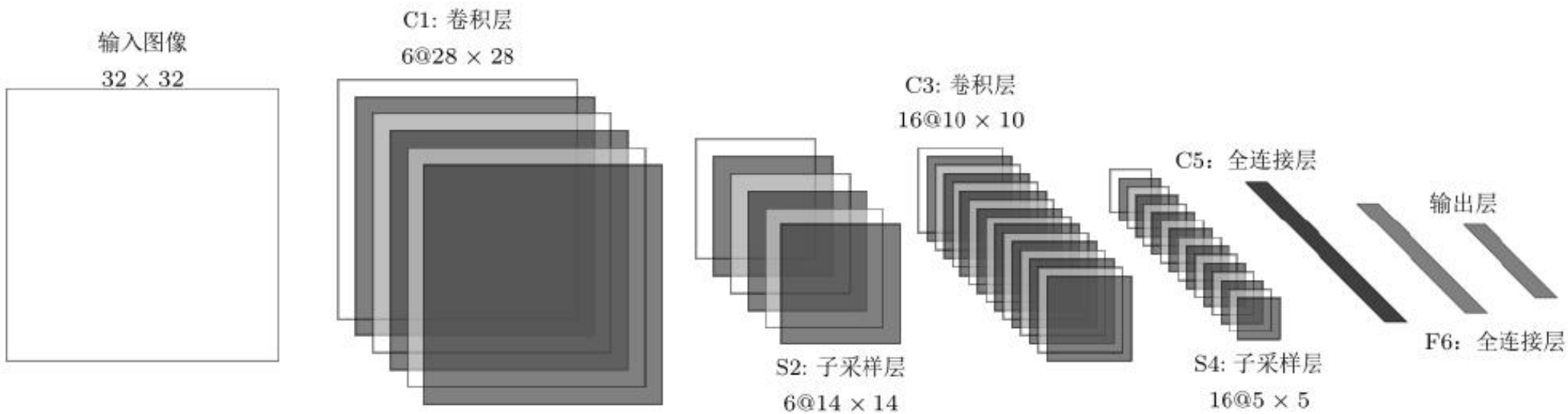
激活函数 $g(z) = \max\{0, z\}$ 定义的**整流线性单元**（rectified linear unit，ReLU）

softmax 函数的形式为

$$\text{softmax}(\boldsymbol{z})_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}.$$

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 的网络结构如图15所示。不计输入层，LeNet-5 共有 7 层。



LeNet-5

The LeNet-5 architecture is perhaps the most widely known CNN architecture. As mentioned earlier, it was created by Yann LeCun in 1998 and widely used for handwritten digit recognition (MNIST). It is composed of the layers shown in Table ·

LeNet-5 architecture

Layer	Type	Maps	Size	Kernel size	Stride	Activation
Out	Fully Connected	–	10	–	–	RBF
F6	Fully Connected	–	84	–	–	tanh
C5	Convolution	120	1×1	5×5	1	tanh
S4	Avg Pooling	16	5×5	2×2	2	tanh
C3	Convolution	16	10×10	5×5	1	tanh
S2	Avg Pooling	6	14×14	2×2	2	tanh
C1	Convolution	6	28×28	5×5	1	tanh
In	Input	1	32×32	–	–	–

Large Scale Visual Recognition Challenge

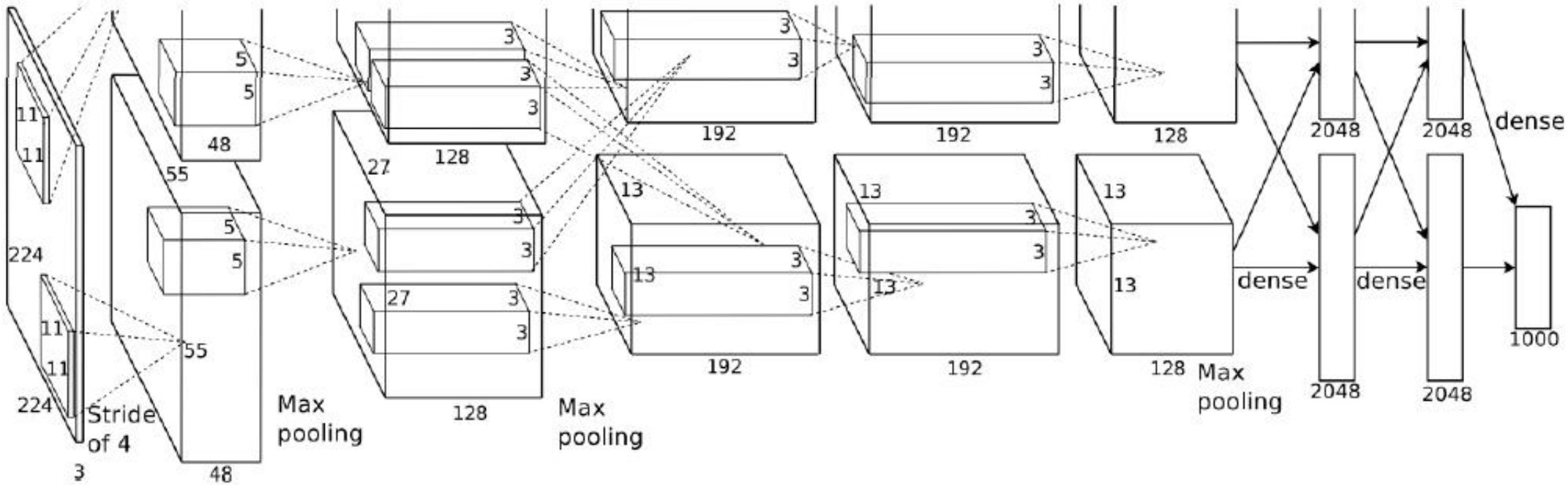
2012 Teams	%error
Supervision (Toronto)	15.3
ISI (Tokyo)	26.1
VGG (Oxford)	26.9
XRCE/INRIA	27.0
UvA (Amsterdam)	29.6
INRIA/LEAR	33.4

2013 Teams	%error
Clarifai (NYU spinoff)	11.7
NUS (singapore)	12.9
Zeiler-Fergus (NYU)	13.5
A. Howard	13.5
OverFeat (NYU)	14.1
UvA (Amsterdam)	14.2
Adobe	15.2
VGG (Oxford)	15.2
VGG (Oxford)	23.0

2014 Teams	%error
GoogLeNet	6.6
VGG (Oxford)	7.3
MSRA	8.0
A. Howard	8.1
DeeperVision	9.5
NUS-BST	9.7
TTIC-ECP	10.2
XYZ	11.2
UvA	12.1

AlexNet

- 2012 ILSVRC winner
  - （top 5 error of 16\% compared to runner-up with 26\% error ）
  - 共有8层，其中前5层卷积层，后边3层全连接层



AlexNet

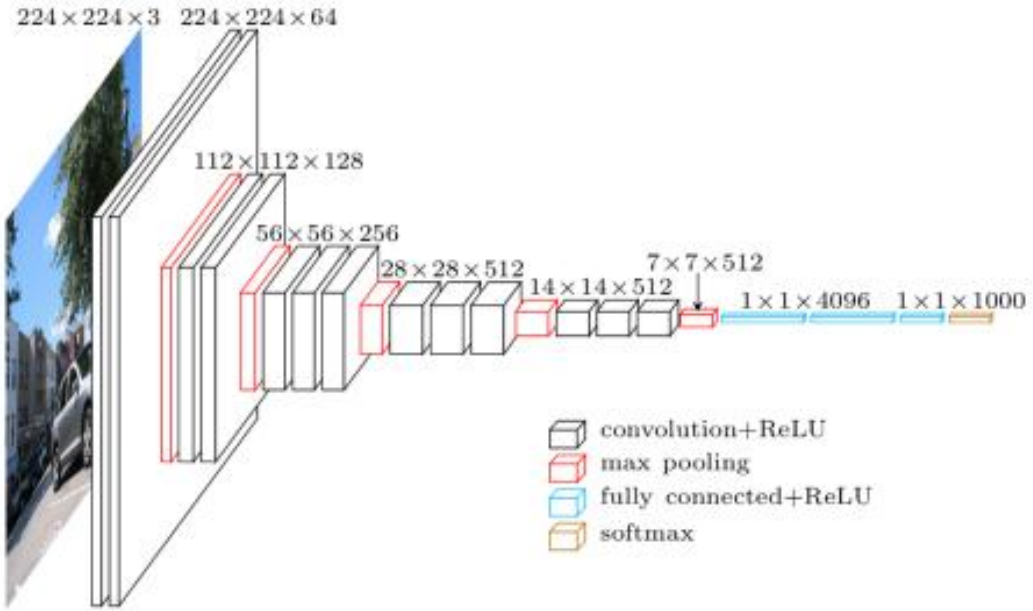
The *AlexNet* CNN architecture won the 2012 ImageNet ILSVRC challenge by a large margin: it achieved 17% top-5 error rate while the second best achieved only 26%! It was developed by Alex Krizhevsky (hence the name), Ilya Sutskever, and Geoffrey Hinton. It is quite similar to LeNet-5, only much larger and deeper, and it was the first to stack convolutional layers directly on top of each other, instead of stacking a pooling layer on top of each convolutional layer.

Layer	Type	Maps	Size	Kernel size	Stride	Padding	Activation
Out	Fully Connected	–	1,000	–	–	–	Softmax
F9	Fully Connected	–	4,096	–	–	–	ReLU
F8	Fully Connected	–	4,096	–	–	–	ReLU
C7	Convolution	256	13×13	3×3	1	SAME	ReLU
C6	Convolution	384	13×13	3×3	1	SAME	ReLU
C5	Convolution	384	13×13	3×3	1	SAME	ReLU
S4	Max Pooling	256	13×13	3×3	2	VALID	–
C3	Convolution	256	27×27	5×5	1	SAME	ReLU
S2	Max Pooling	96	27×27	3×3	2	VALID	–
C1	Convolution	96	55×55	11×11	4	VALID	ReLU
In	Input	3 (RGB)	227×227	–	–	–	–

VGGNet

- 2014年，牛津大学计算机视觉组（Visual Geometry Group）和Google DeepMind公司的研究员一起研发出了新的深度卷积神经网络：VGGNet，并取得了ILSVRC2014比赛分类项目的第二名（第一名是GoogLeNet，也是同年提出的）和定位项目的第一名。
- VGGNet探索了卷积神经网络的深度与其性能之间的关系，成功地构筑了16~19层深的卷积神经网络，证明了增加网络的深度能够在一定程度上影响网络最终的性能，使错误率大幅下降，同时拓展性又很强，迁移到其它图片数据上的泛化性也非常好。到目前为止，VGG仍然被用来提取图像特征。

CNN经典模型
VGGNet



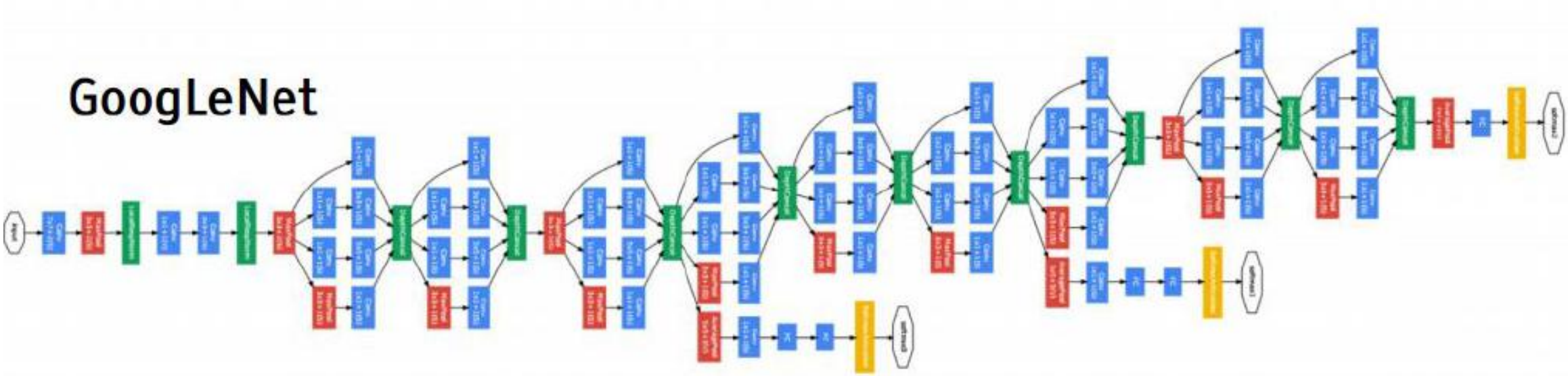
输入 —— 卷积层 —— 全连接 输出

VGG16 VGG19

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

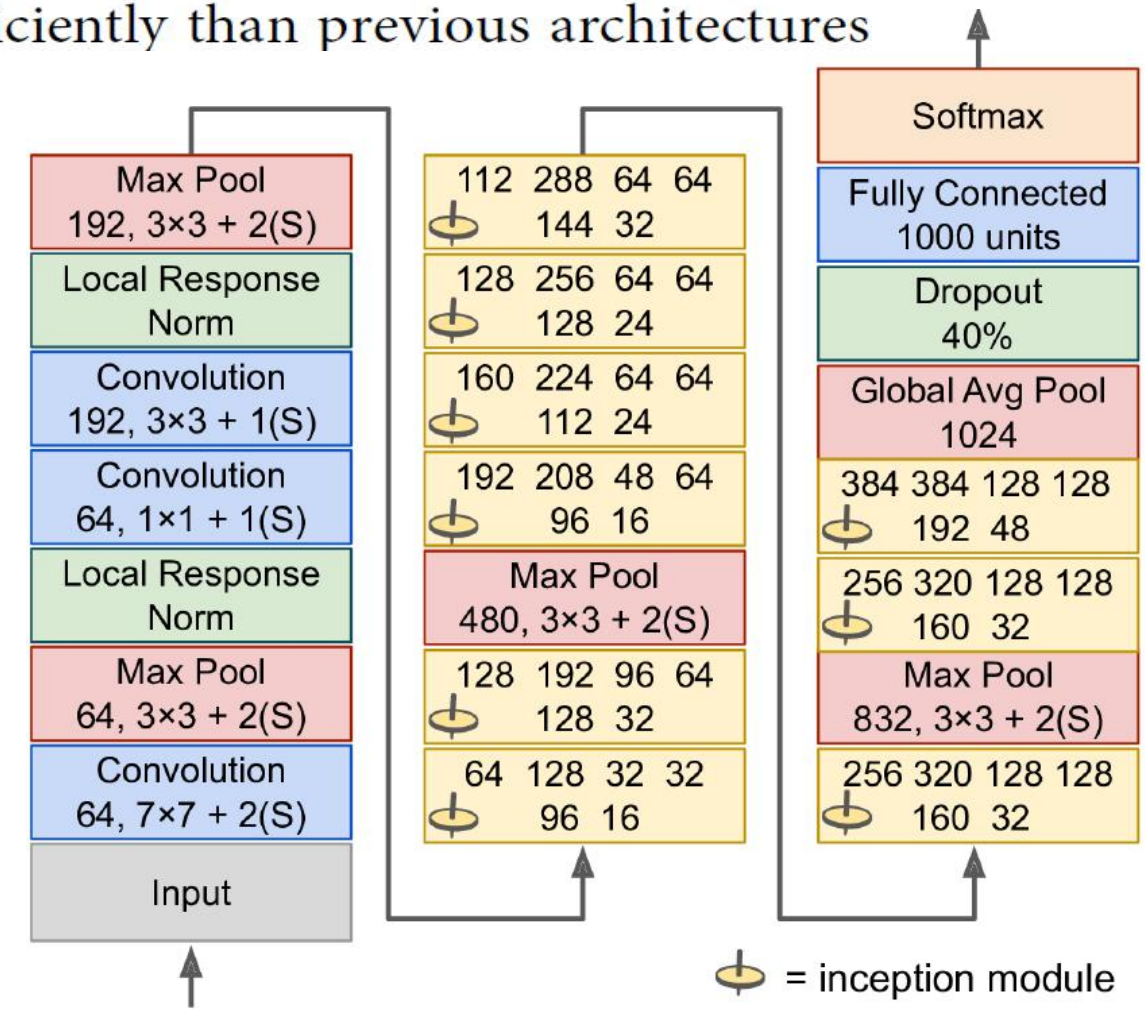
GoogLeNet

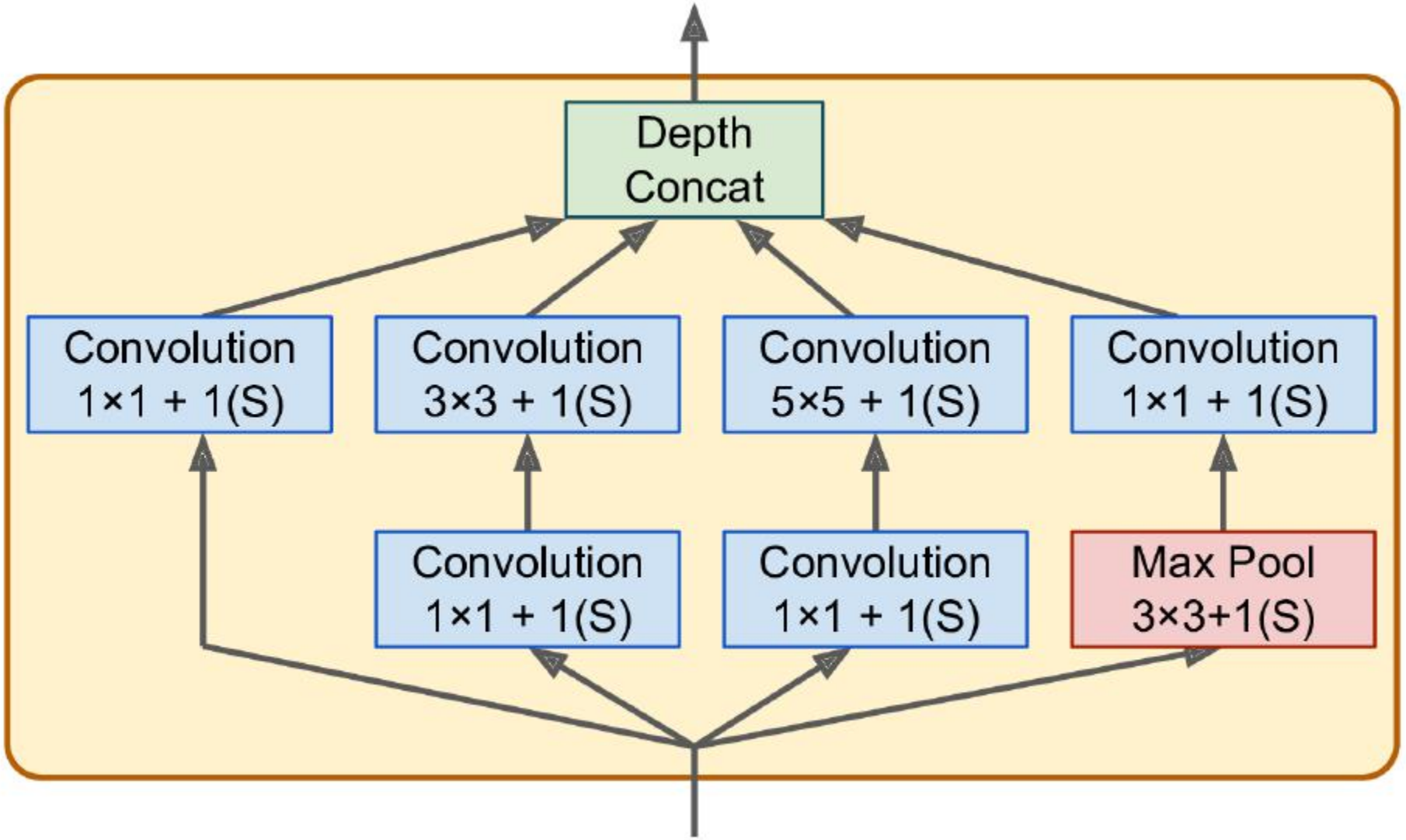
- 2014 ILSVRC winner （22层）
  - 参数：GoogLeNet：4M VS AlexNet：60M
  - 错误率：6.7%



GoogLeNet

The GoogLeNet architecture won the ILSVRC 2014 challenge by pushing the top-5 error rate below 7%. This great performance came in large part from the fact that the network was much deeper than previous CNNs . This was made possible by sub-networks called *inception modules*, which allow GoogLeNet to use parameters much more efficiently than previous architectures

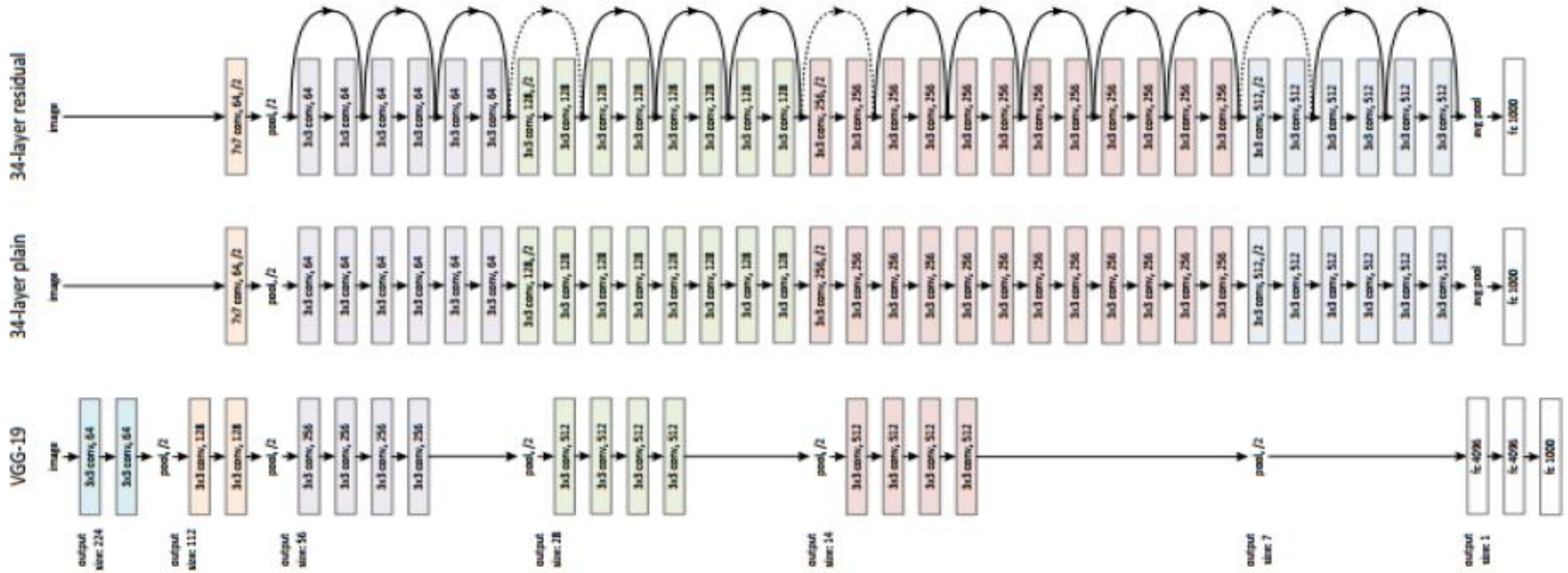




Inception module

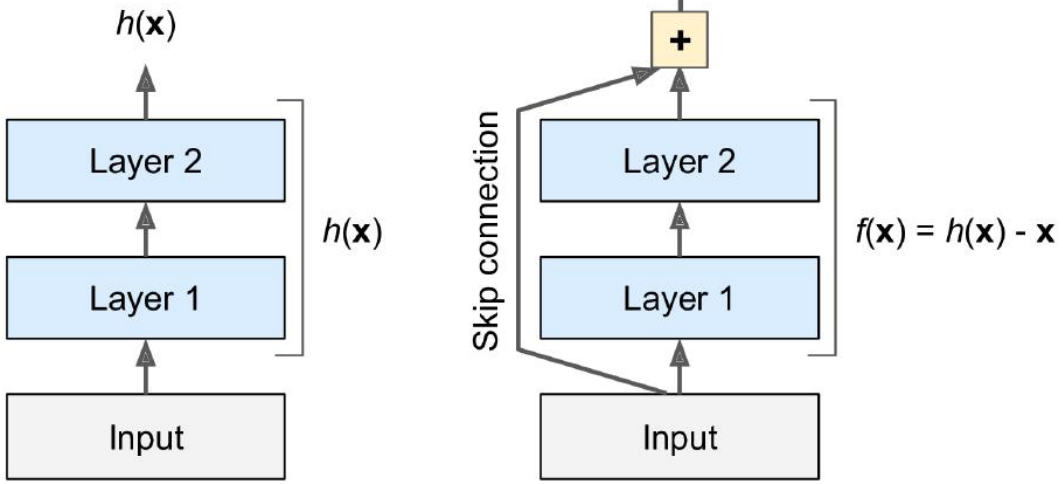
ResNet

- 2015 ILSVRC winner （152层）
  - 错误率： 3.57%

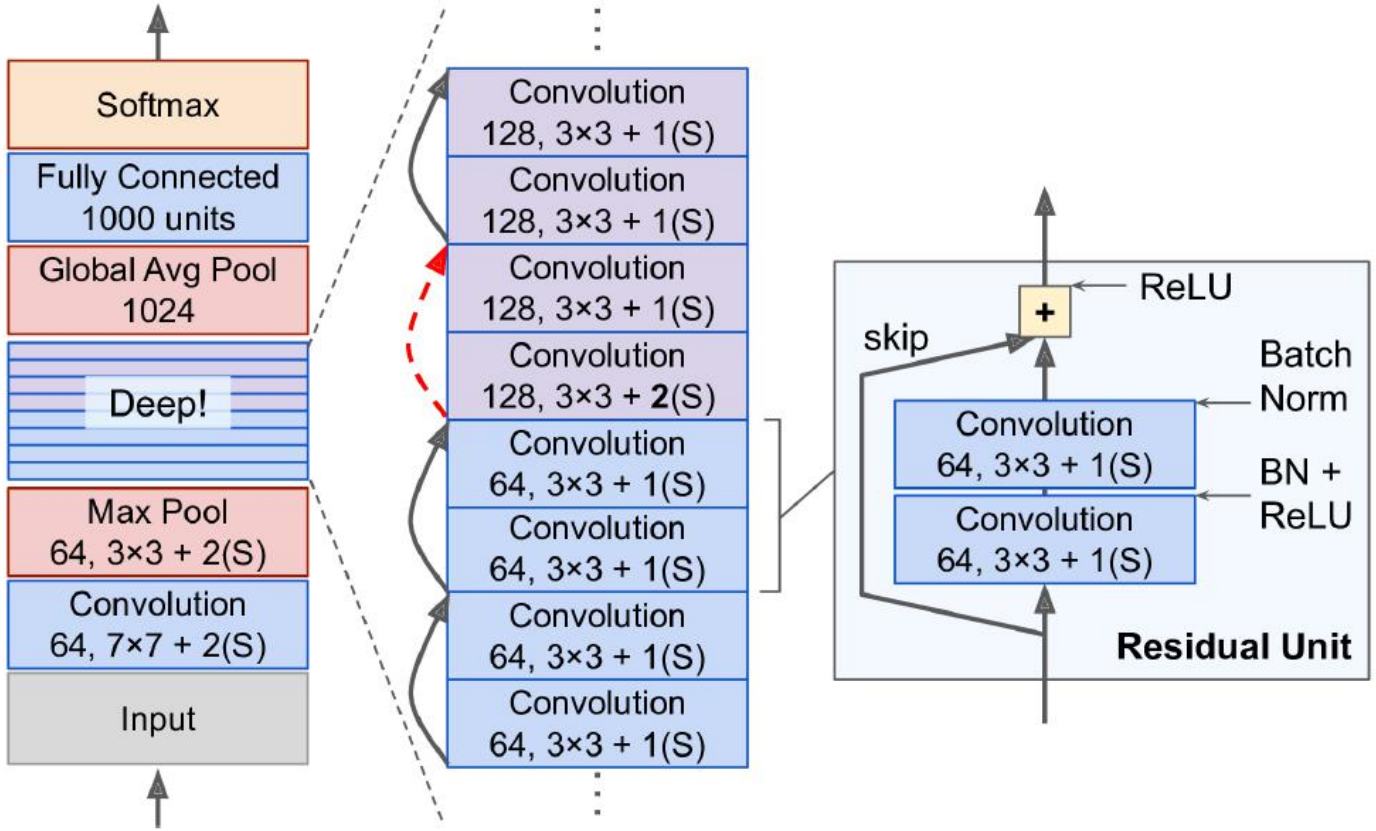


ResNet

The ILSVRC 2015 challenge was won using a *Residual Network* (or *ResNet*), developed by Kaiming He et al., which delivered an astounding top-5 error rate under 3.6%, using an extremely deep CNN composed of 152 layers. It confirmed the general trend: models are getting deeper and deeper, with fewer and fewer parameters.

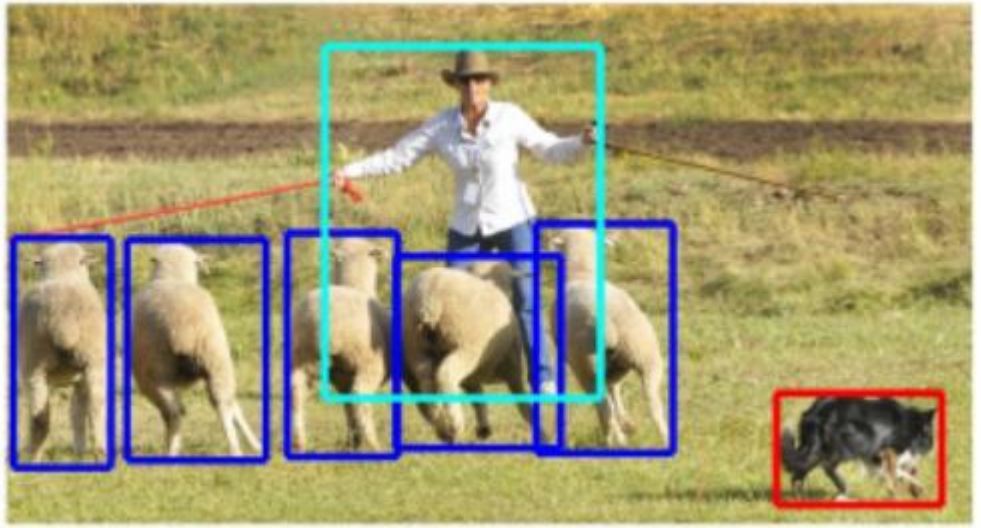


Residual learning



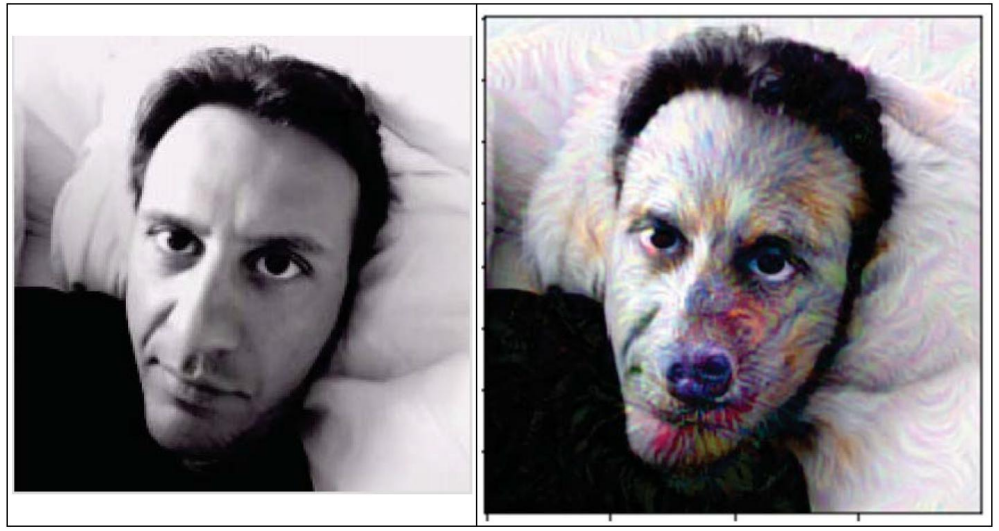
ResNet architecture

Image classification



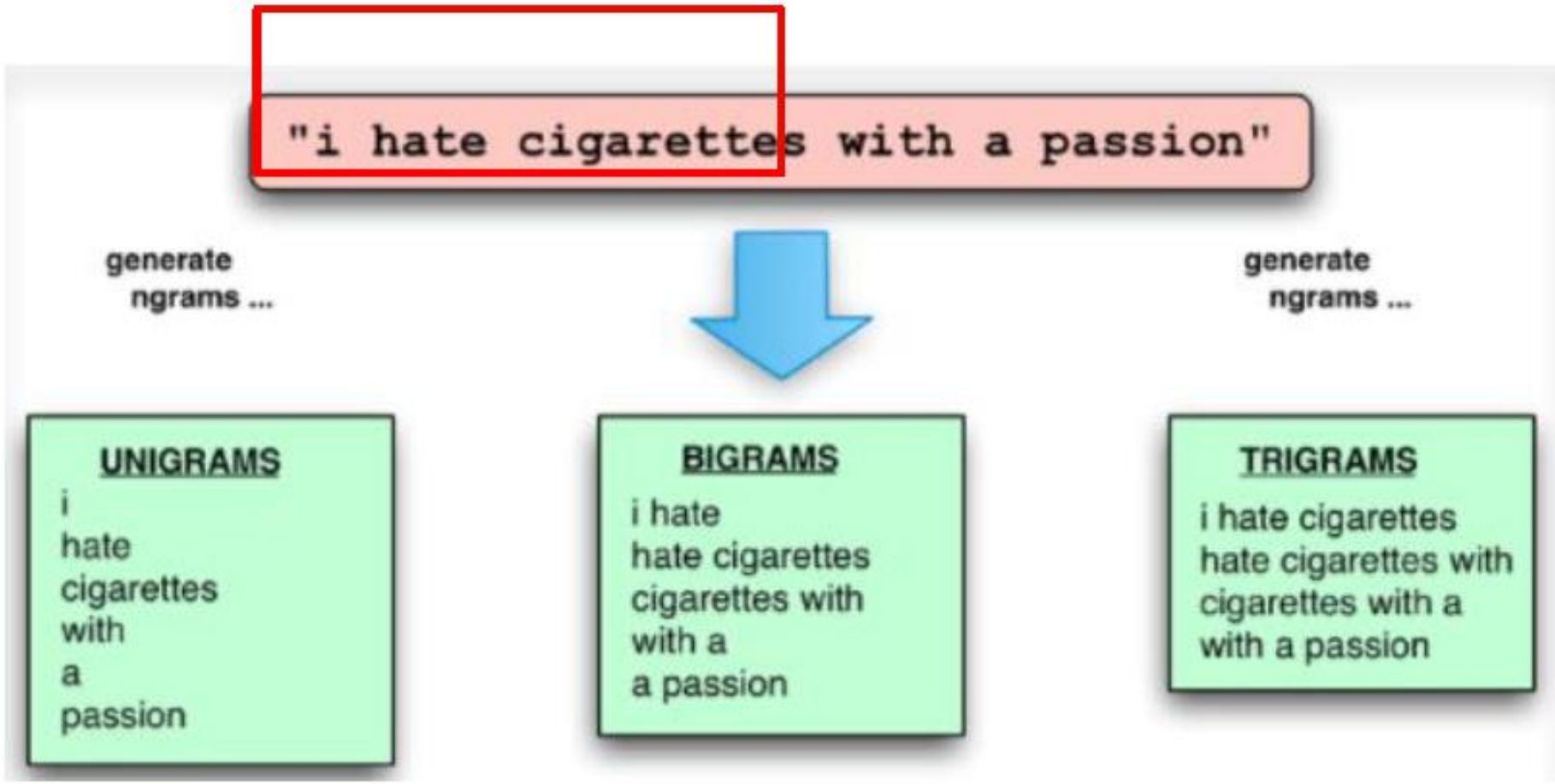
Object localization

Semantic segmentation



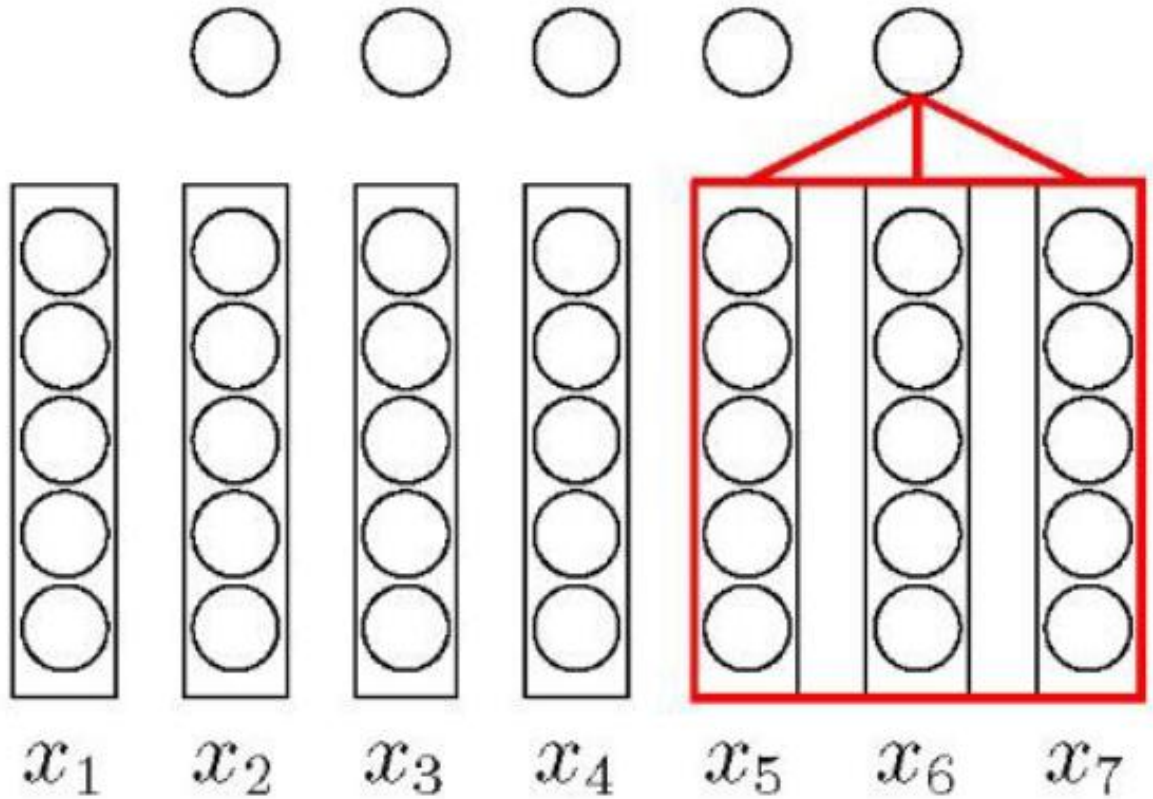
Transferring style for image repainting

Ngram特征与卷积

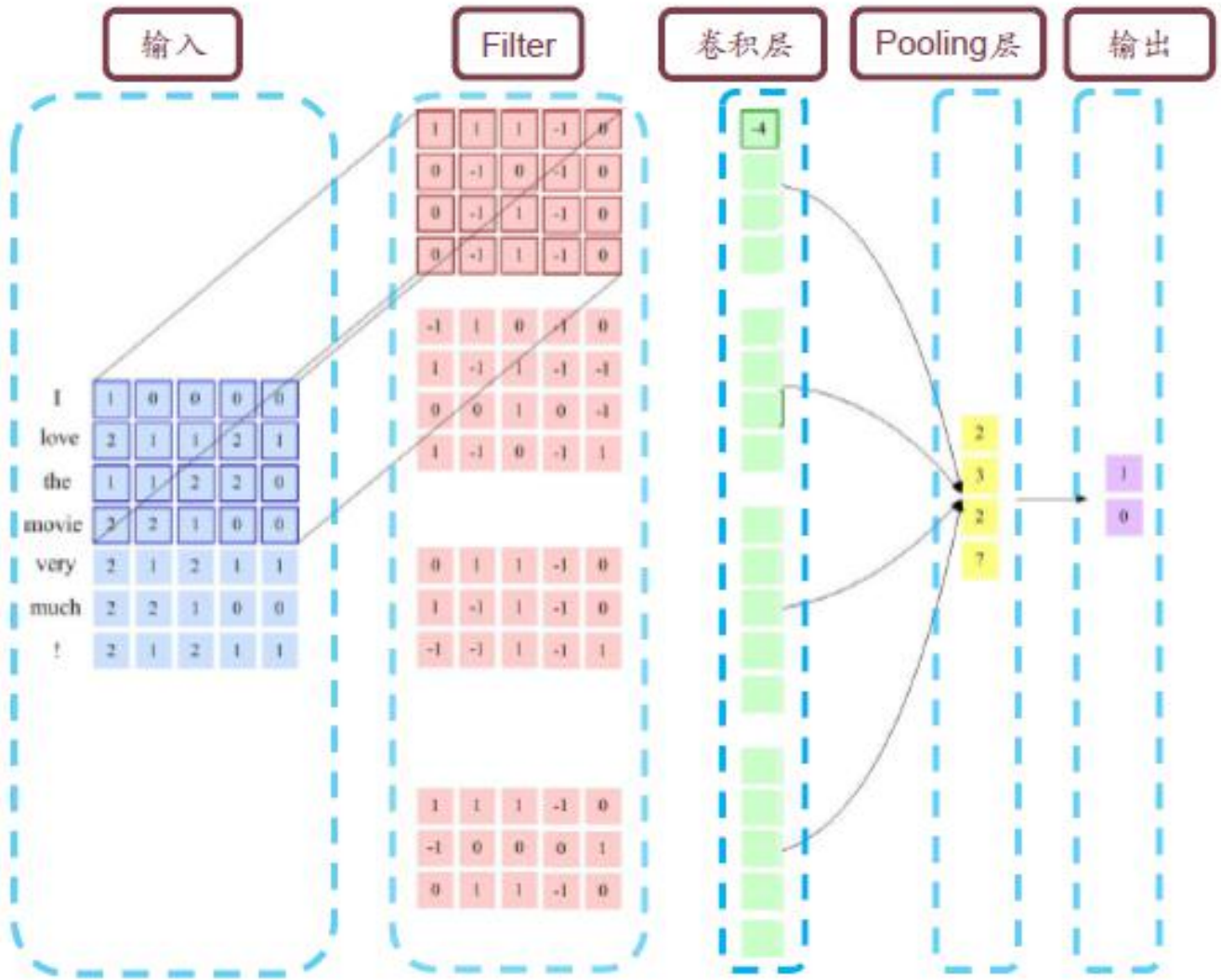


如何用卷积操作来实现?

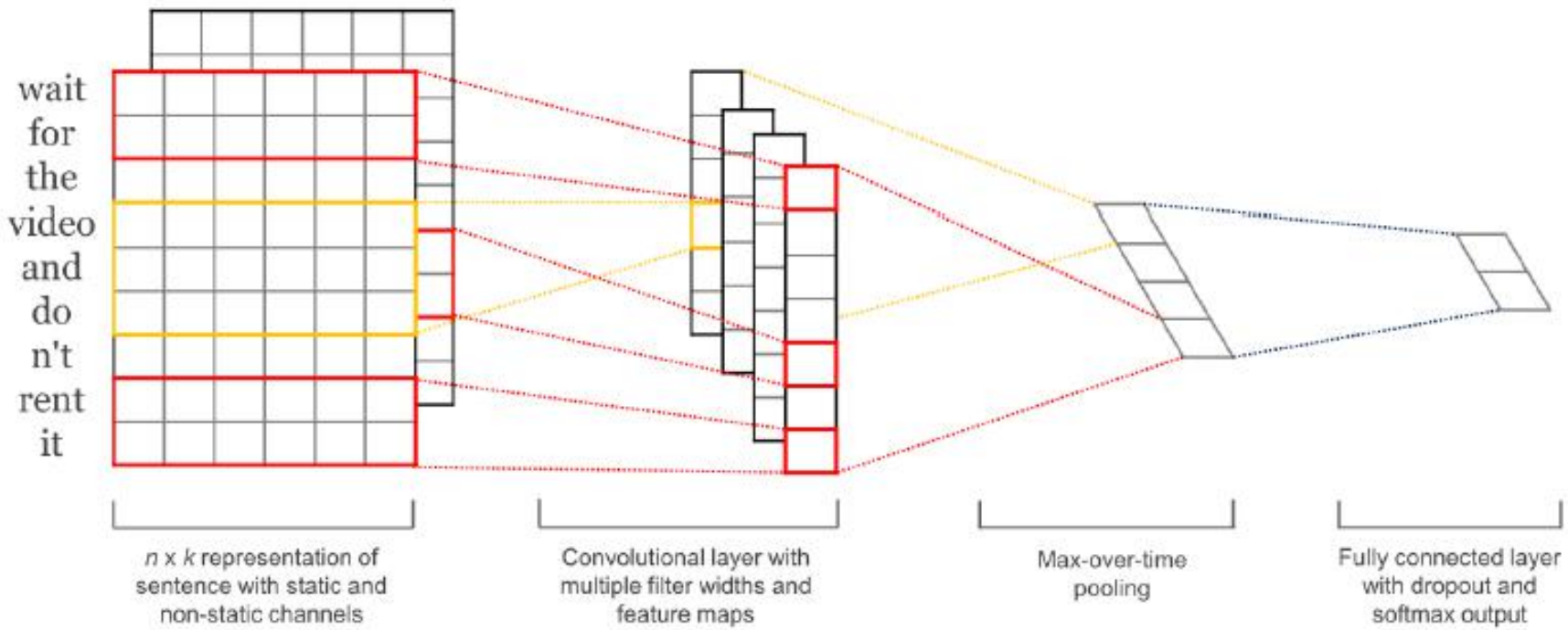
文本序列的卷积



文本序列的卷积模型

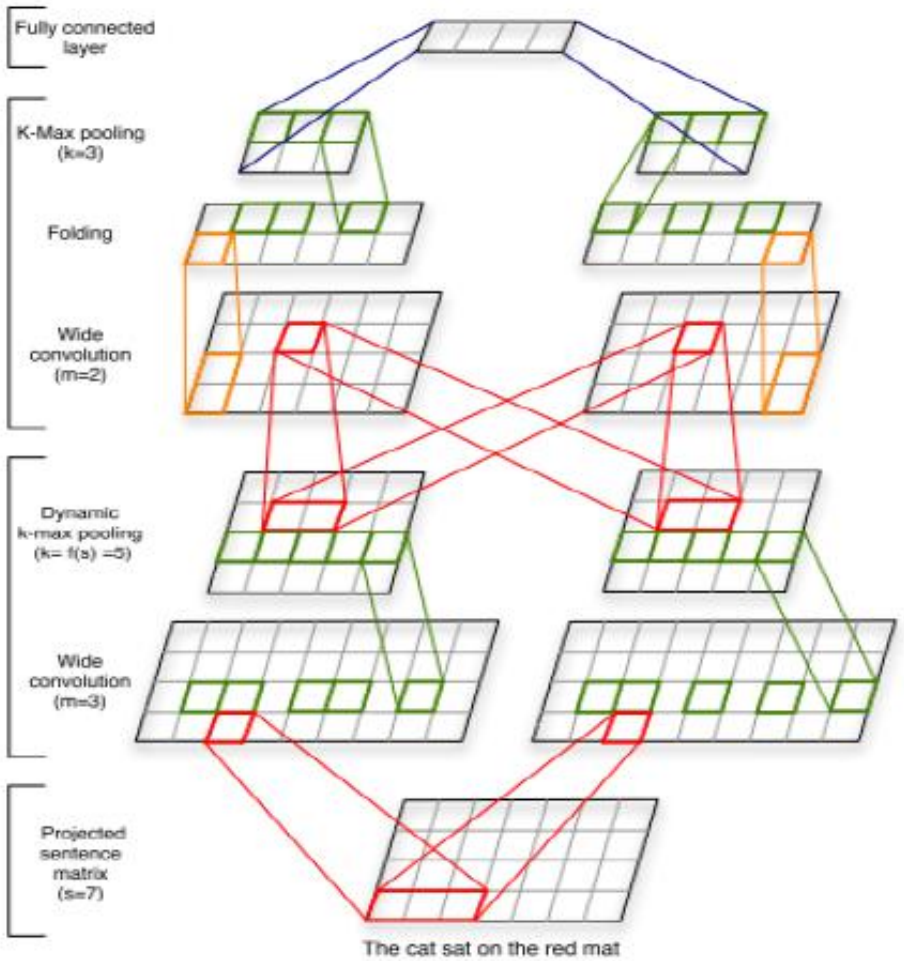


基于卷积模型的句子表示



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

基于卷积模型的句子表示



N. Kalchbrenner, E. Grefenstette, and P. Blunsom. "A Convolutional Neural Network for Modelling Sentences". In:
Proceedings of ACL. 2014

pytorch实现LeNet

LeNet 是整个卷积神经网络的开山之作，1998 年由 LeCun 提出，它的结构特别简单，我们能够由此入手，一步一步地进入时下最为流行的卷积神经网络结构。

首先，LeNet 的网络结构如图4.16所示。

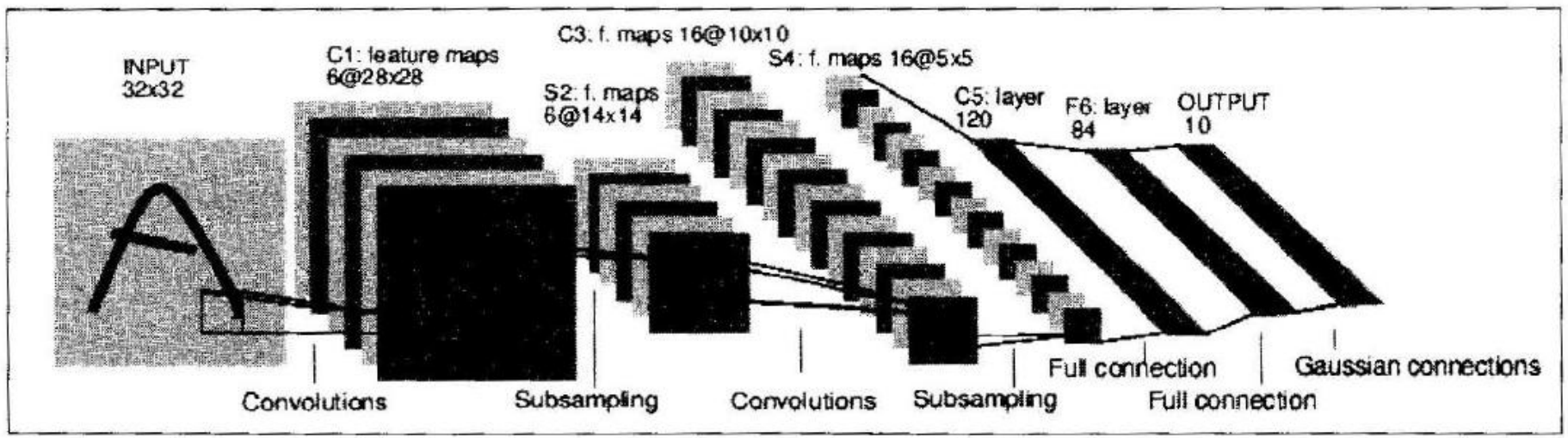


图 4.16　LeNet 网络结构

从图 4.16 可以看出整个网络结构特别清晰，一共有 7 层，其中 2 层卷积和 2 层池化层交替出现，最后输出 3 层全连接层得到整体的结果。


Thanks!
Obrigado!
Gracias!
谢谢！
Dziękuję!
Danke!
Grazie!
Merci!
شكرا لك !
감사합니다!
Kiitos!