

深度学习讨论班

第三节

Convolutional Neural Networks (卷积神经网络)

黄雷

2016-12-13

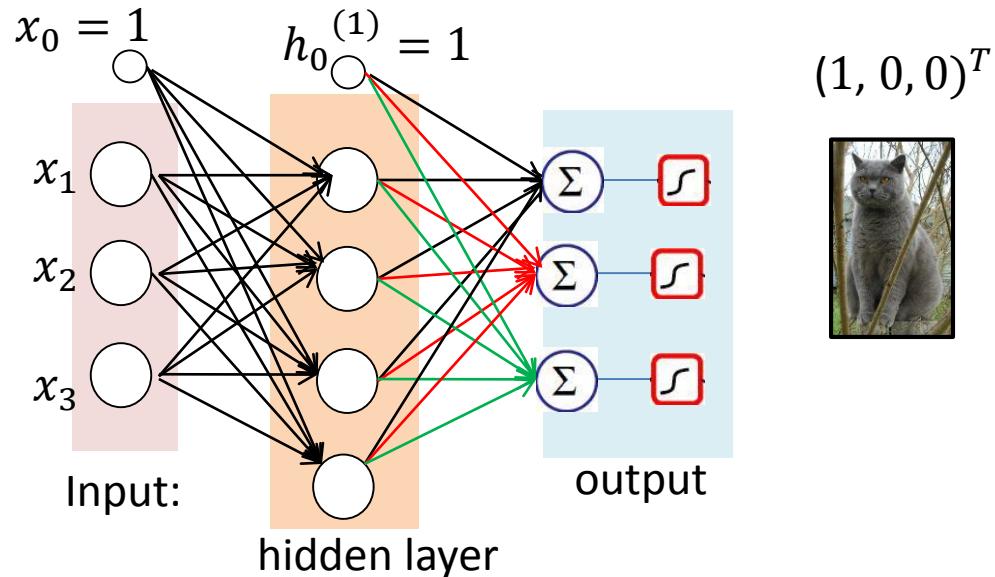
上一讲主要内容

- Linear classifier (简单线性分类器)
 - One neuron (一个神经元)
 - Multiple neurons (多个神经元)
- Multi-layer perceptron (多层感知机)
 - Model representation (模型表示)
 - Loss function: the goal for learning
 - Training
 - Gradient based optimization
 - backpropagation

Multi-layer perceptron

- Training Algorithm

- 0. 初始化权重 $\mathbf{W}^{(0)}$
- 1. 前向过程:
 - 1.1 根据输入 \mathbf{x} , 计算输出值 \mathbf{y}
 - 1.2. 计算损失函数值 $L(\mathbf{y}, \hat{\mathbf{y}})$ 。
- 2. 后向传播
 - 计算 $\frac{d L}{\mathbf{y}}$
 - 后向传播直到计算 $\frac{d L}{\mathbf{x}}$
- 3. 计算梯度 $\frac{d L}{d \mathbf{W}}$
- 4. 更新梯度
$$\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} - \eta \frac{d L}{d \mathbf{W}^{(t)}}$$

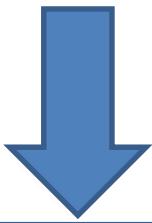


outline

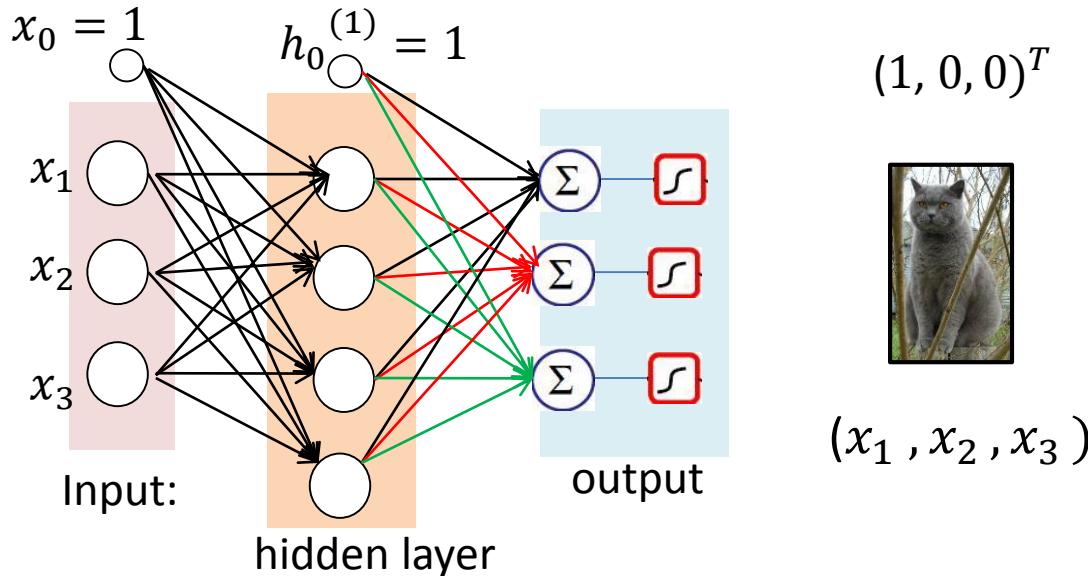
- Modeling of CNN
 - Module-wise architecture 模块化结构
- Convolutional layer (module)
 - Convolution in general
 - Filters
 - Convolution module
- Pooling layer (module)

Feature extraction

- Feature extraction
 - Pixel-wise input
 - Correlation between features



Convolutional Neural Network(CNN), 卷积神经网络



(x_1, x_2, x_3)

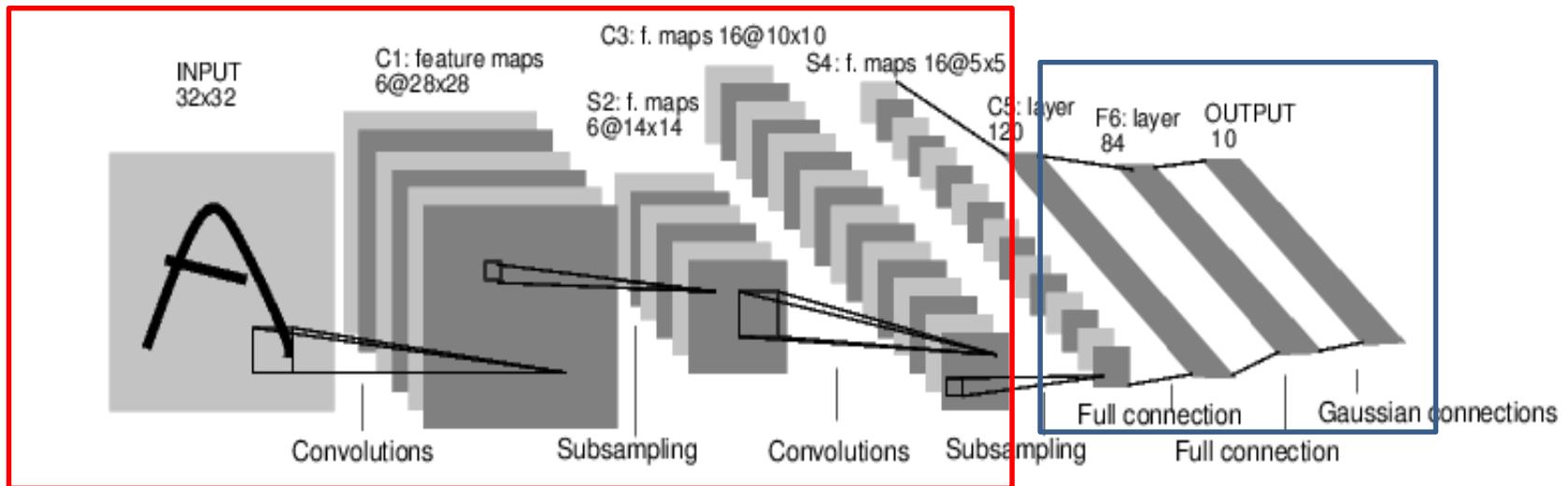


08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 71 31 45 49 49 99 40 17 81 18 57 60 87 17 40 98 49 69 14 56 45 58 62 00 81 49 31 73 55 79 14 29 93 72 40 67 39 05 30 09 49 13 36 65 92 70 95 23 04 60 11 42 63 05 56 01 32 56 71 37 02 36 91 22 31 16 71 51 67 13 59 41 92 36 54 22 40 40 28 66 33 13 80 24 47 31 05 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50 13 35 81 28 64 23 67 10 26 38 40 67 59 54 54 70 66 18 38 64 70 47 26 20 68 02 62 12 20 95 63 94 39 63 08 40 91 66 49 94 21 24 55 58 05 66 73 99 26 97 17 78 78 96 03 14 88 34 89 63 72 41 36 23 09 75 00 76 44 20 45 35 14 00 62 33 97 34 31 33 95 78 17 53 28 22 78 31 67 15 94 03 80 04 62 16 14 09 53 56 92 16 39 05 42 94 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57 86 56 05 48 35 71 89 07 08 44 44 37 44 60 21 58 51 54 17 58 19 80 81 68 05 94 47 69 21 73 99 13 88 52 17 77 04 89 55 40 04 52 08 83 97 35 99 16 07 97 57 32 14 26 28 79 33 27 98 66 14 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69 64 42 26 73 05 55 39 11 24 94 72 18 08 46 29 32 40 62 76 36 20 69 36 41 72 30 23 88 34 05 88 69 82 67 59 85 74 04 36 16 40 79 35 29 78 31 90 01 71 32 49 74 48 04 24 23 57 05 54 01 70 54 73 83 51 54 69 36 92 33 49 61 43 52 01 79 14 45
--

What the computer sees

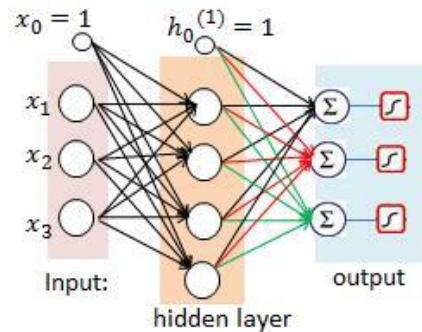
Convolution Neural Network

- Lenet-5



Convolution related layers

全连接层



outline

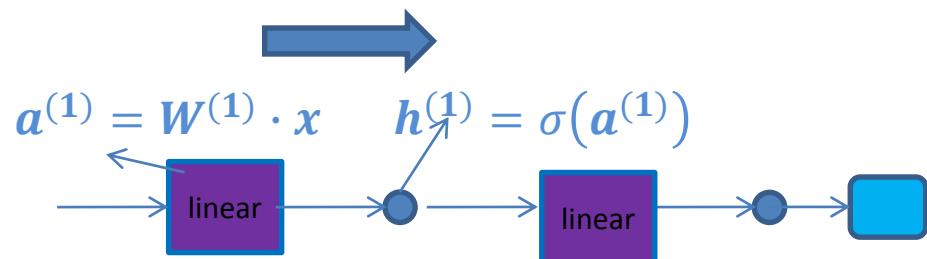
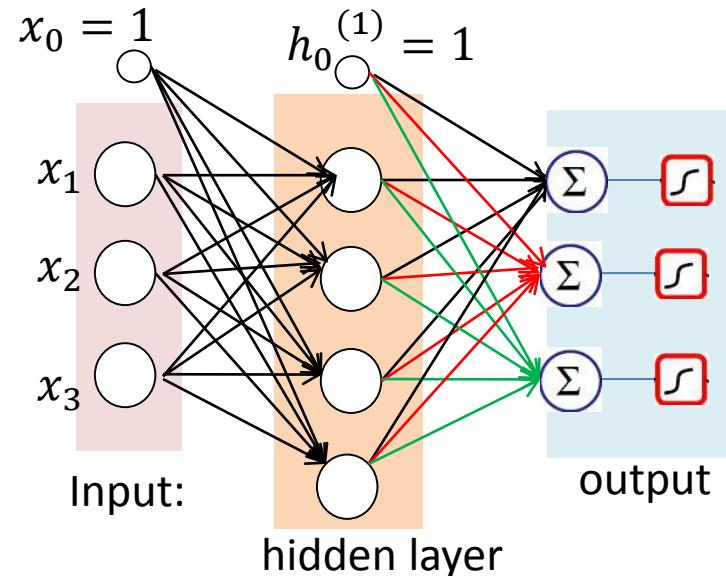
- Modeling of CNN
 - Module-wise architecture 模块化结构
- Convolutional layer (module)
 - Convolution in general
 - Filters
 - Convolution module
- Pooling layer (module)

Module-wise architecture

➤ Torch 平台

- Model construction

```
function create_model()
    model = nn.Sequential()
    model:add(nn.Linear(3, 4))
    model:add(nn.Sigmoid())
    model:add(nn.Linear(4, 3))
    model:add(nn.Sigmoid())
    criterion = nn.MSECriterion()
    return model, criterion
end
```

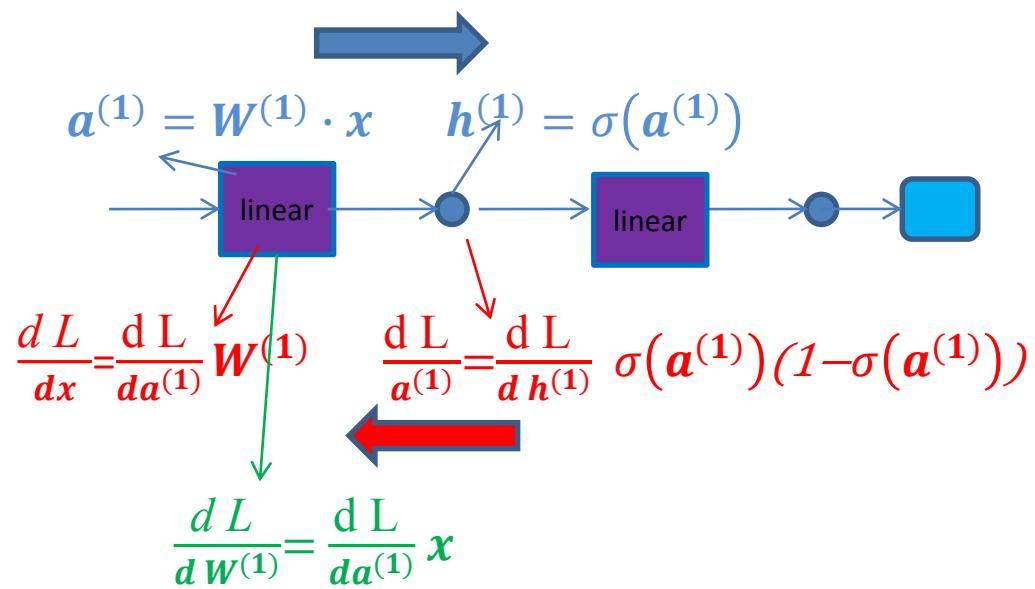
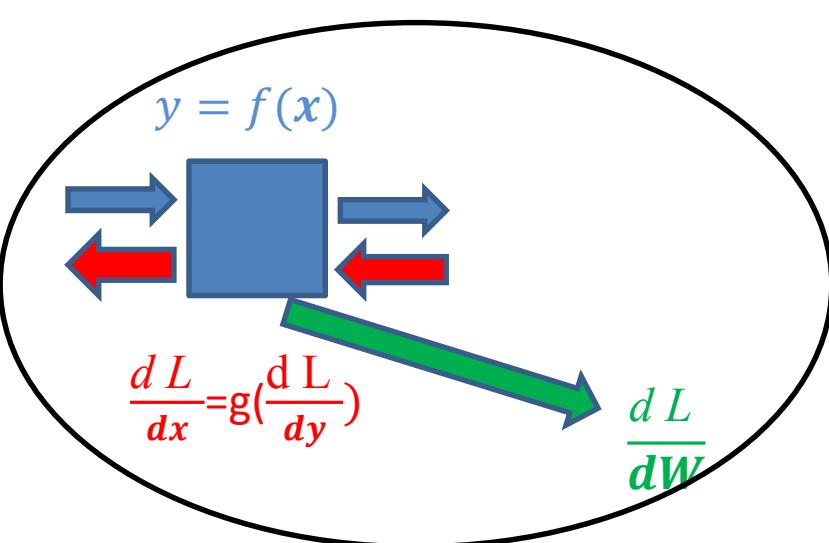
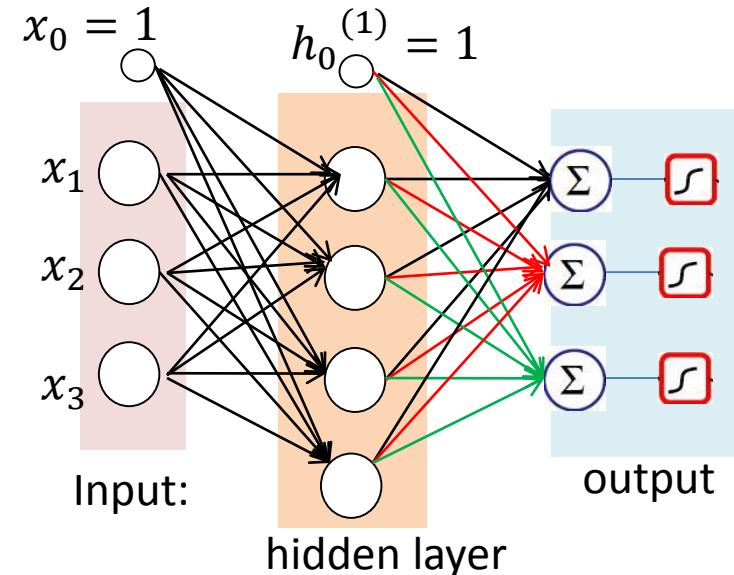


Module-wise architecture

➤ Torch 平台

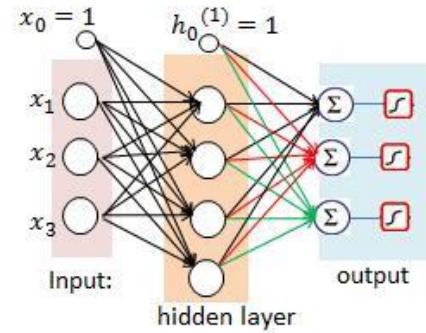
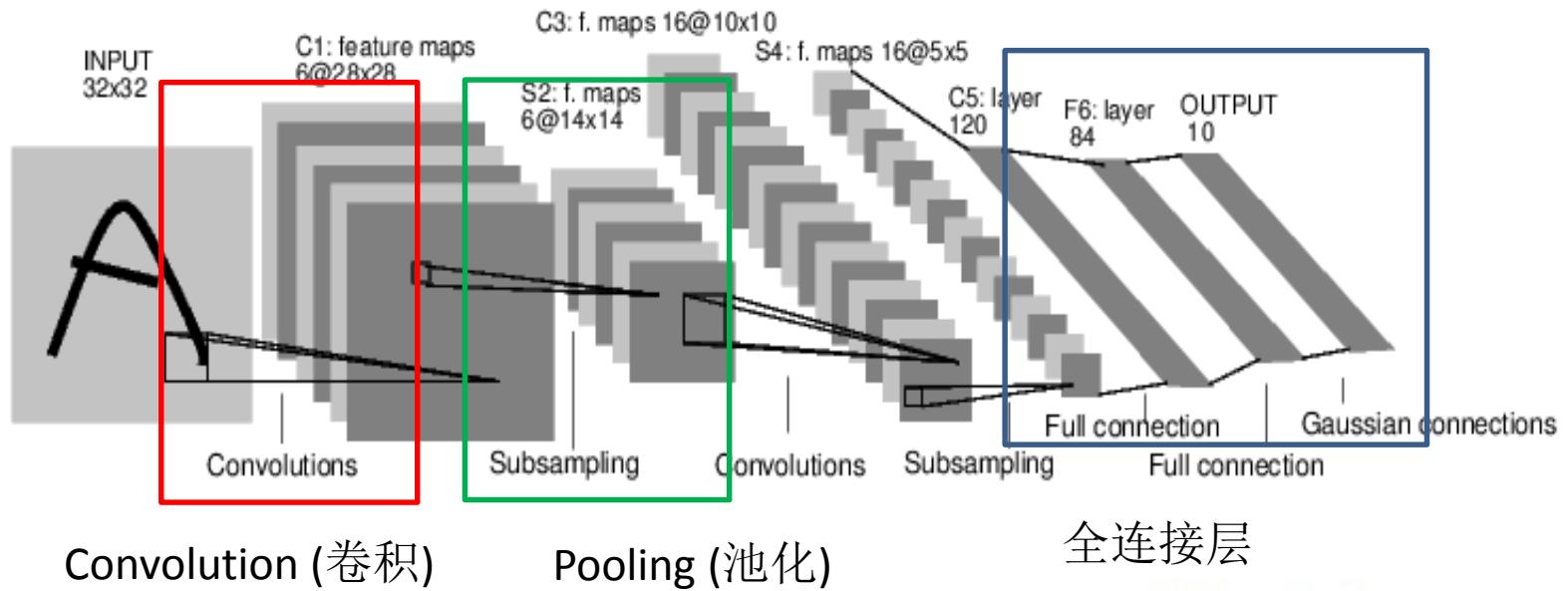
- Training per iteration:

```
-- forward  
outputs = model:forward(X)  
loss = criterion:forward(outputs, Y)  
-- backward  
dloss_doutput = criterion:backward(outputs, Y)  
model:backward(X, dloss_doutput)
```



Convolution Neural Network

- Lenet-5



outline

- Modeling of CNN
 - Module-wise architecture 模块化结构
- Convolutional layer (module)
 - Convolution in general
 - Filters
 - Convolution module
- Pooling layer (module)

Convolution

➤ 一维离散卷积例子

$$y = \begin{bmatrix} y_1 & y_2 \end{bmatrix}$$

$$y_1 = w_1 x_1 + w_2 x_2$$

$$w = \begin{bmatrix} w_1 & w_2 \end{bmatrix}$$

$$y_2 = w_1 x_2 + w_2 x_3$$

$$x = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}$$

$$y_{i'} = \sum_{i=1}^{M_f=2} w_i x_{i'+i-1}$$

Correlation operator
(similarity)

➤ Flip

$$\bar{w} = \begin{bmatrix} w_2 & w_1 \end{bmatrix}$$

$$y_{i'} = \sum_{i=1}^{M_f=2} w_{M_f+1-i} x_{i'+i-1}$$

Convolution

- 离散空间卷积:

$$y(n) = x(n) * w(n) = \sum_{i=-\infty}^{i=+\infty} x(i)w(n-i)$$

Convolution

- 离散空间卷积:

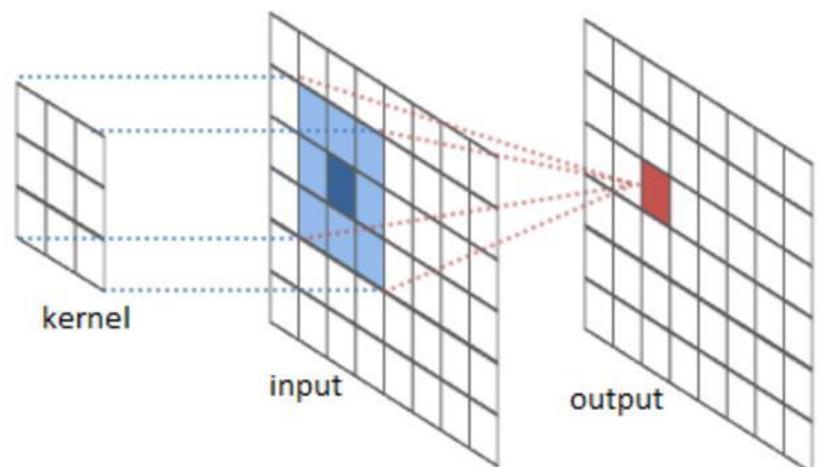
$$y(n) = x(n) * w(n) = \sum_{i=-\infty}^{i=+\infty} x(i)w(n-i)$$

- 连续空间的卷积:

$$y(t) = x(t) * h(t) = \int_{-\infty}^{+\infty} x(s)h(t-s) ds$$

- 图像卷积是二维离散卷积

$$g(i,j) = \sum_{k,l} f(k,l) w(i-k, j-l)$$



Convolution

- 图像卷积，二维，离散
 - Correlation Operator(相关算子)

定义: $g = f \otimes w$



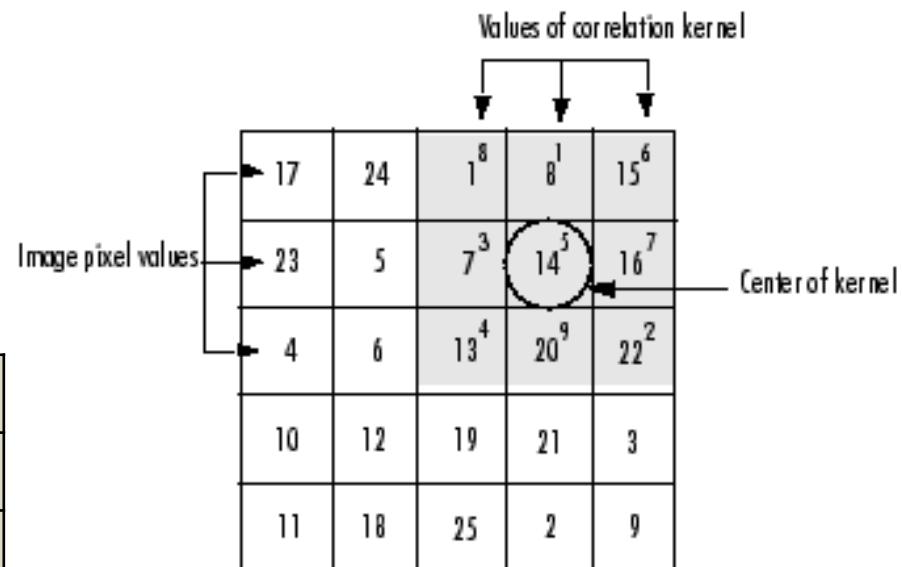
$$g(i, j) = \sum_{k,l} f(i + k, j + l) w(k, l)$$

f:

17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

w:

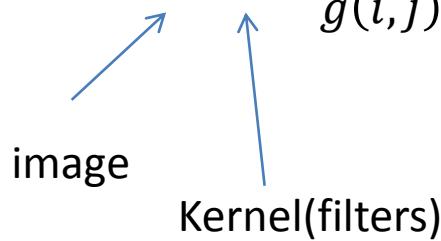
8	1	6
3	5	7
4	9	2



Convolution

- 图像卷积，二维，离散
 - Convolution operator (卷积算子)

定义: $g = f * w$



$f:$

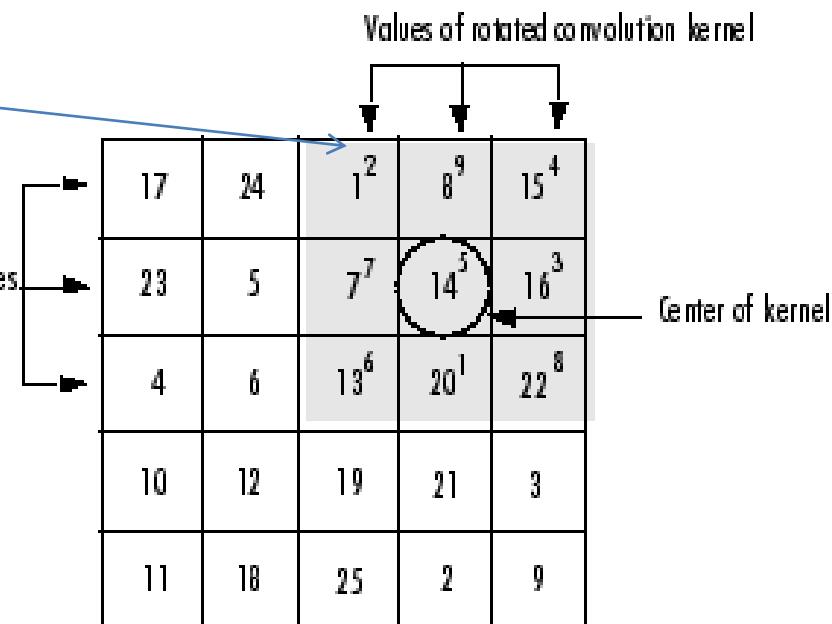
17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

$w:$

8	1	6
3	5	7
4	9	2

旋转180

Image pixel values



outline

- Modeling of CNN
 - Module-wise architecture 模块化结构
- Convolutional layer (module)
 - Convolution in general
 - Filters
 - Convolution module
- Pooling layer (module)

Practice with linear filters(线性濾波器)



Original

0	0	0
0	1	0
0	0	0

Filter



Filtered
(no change)

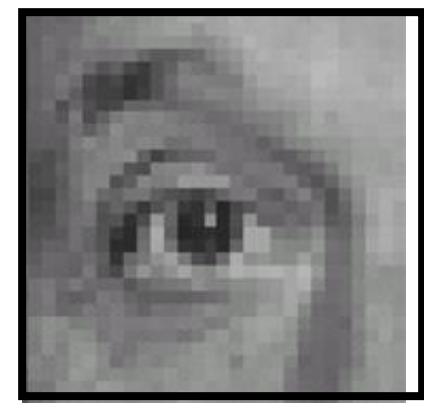
Practice with linear filters



Original

0	0	0
0	0	1
0	0	0

Filter



Shifted *left*
By 1 pixel

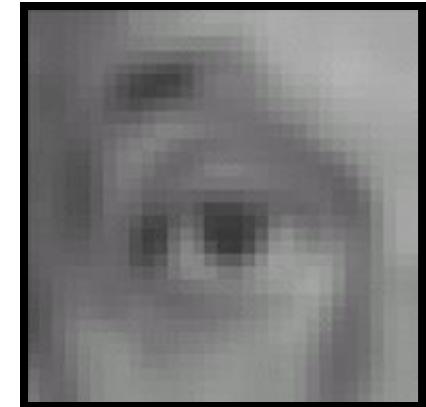
Practice with linear filters



Original

$$\frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

Filter



Blur (with a
box filter)

Practice with linear filters



Original

0	1	0
1	-4	1
0	1	0

Filter



Output Image

Edge detect (边缘检测)

Filters in practise

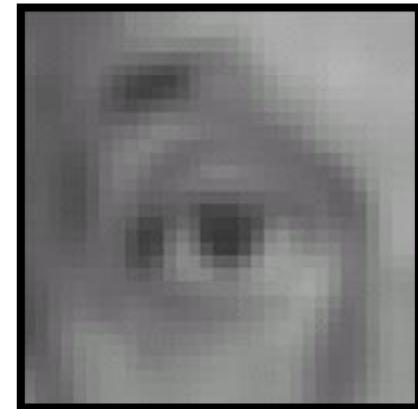


Input image

$$\frac{1}{9}$$

1	1	1
1	1	1
1	1	1

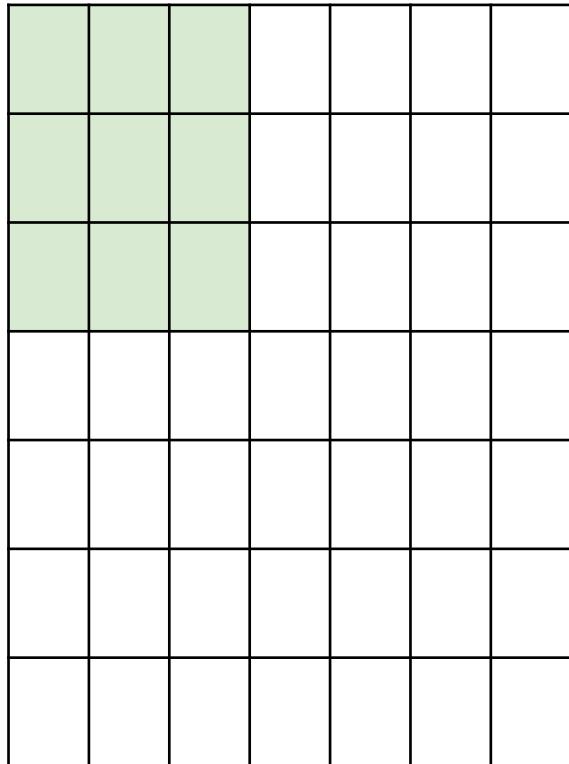
filter



output image

- Size of output image
 - How to move? **stride**
 - How about the border? **padding**

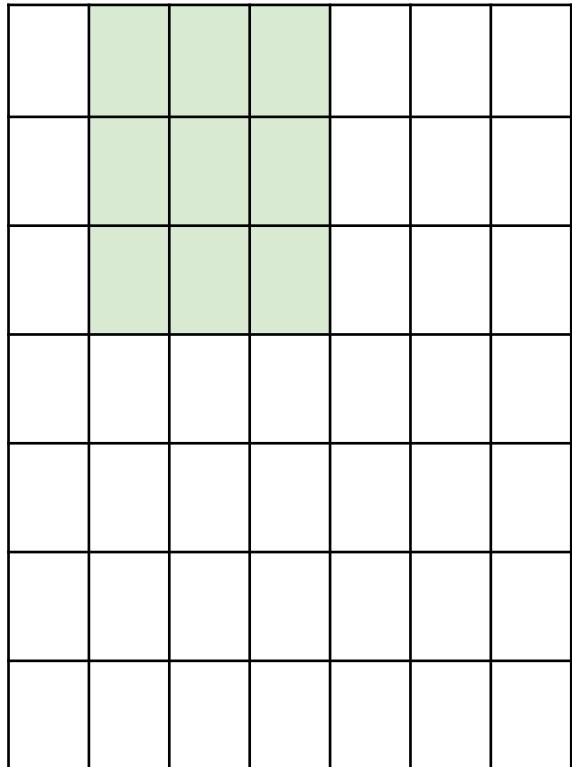
filters: stride (步幅)



7x7 input

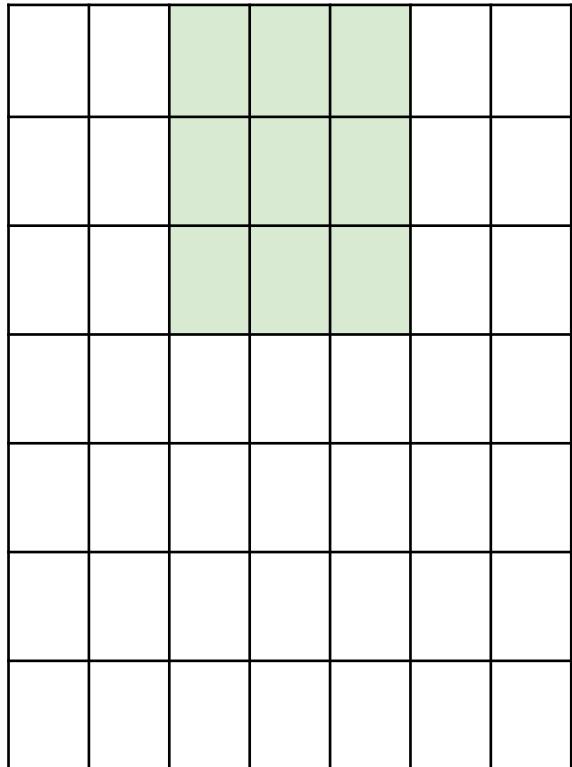
assume 3x3 connectivity, stride 1

filters: stride



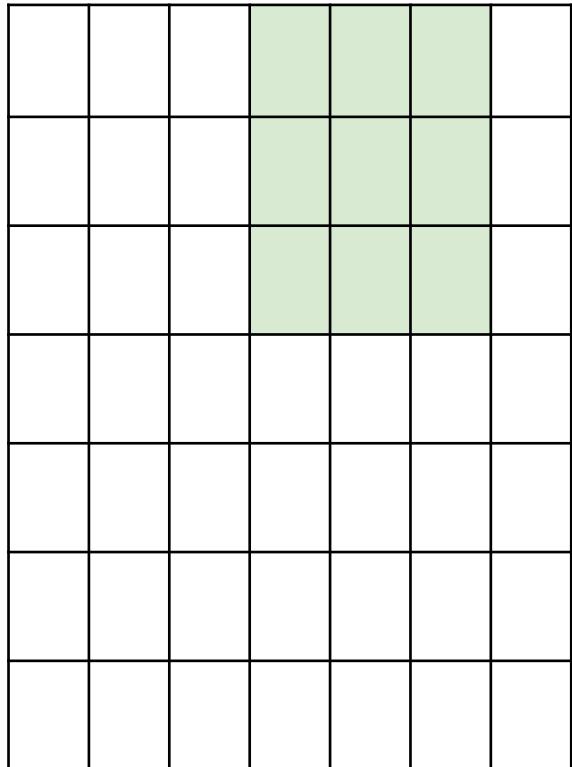
7x7 input
assume 3x3 connectivity, stride 1

filters: stride



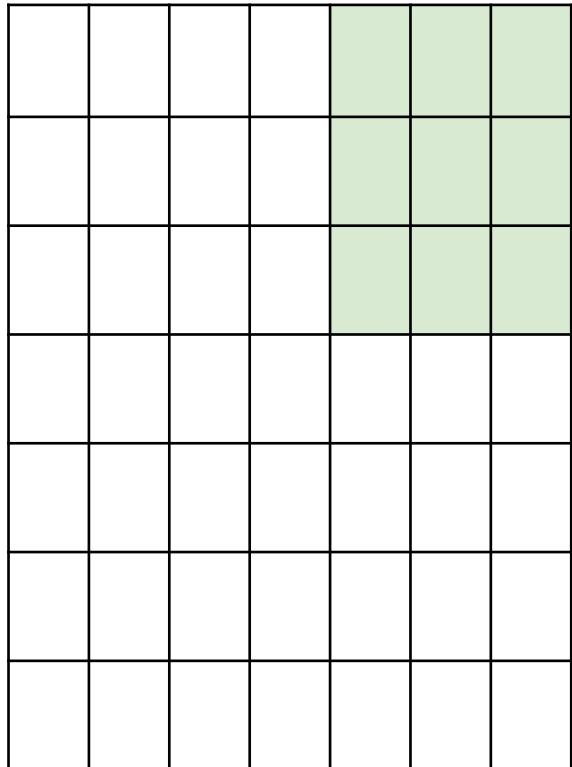
7x7 input
assume 3x3 connectivity, stride 1

filters: stride



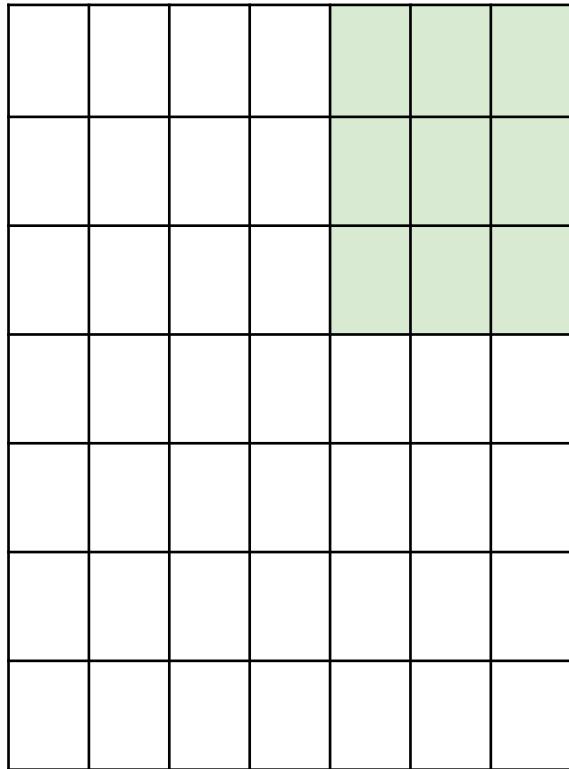
7x7 input
assume 3x3 connectivity, stride 1

filters: stride



7x7 input
assume 3x3 connectivity, stride 1

filters: stride

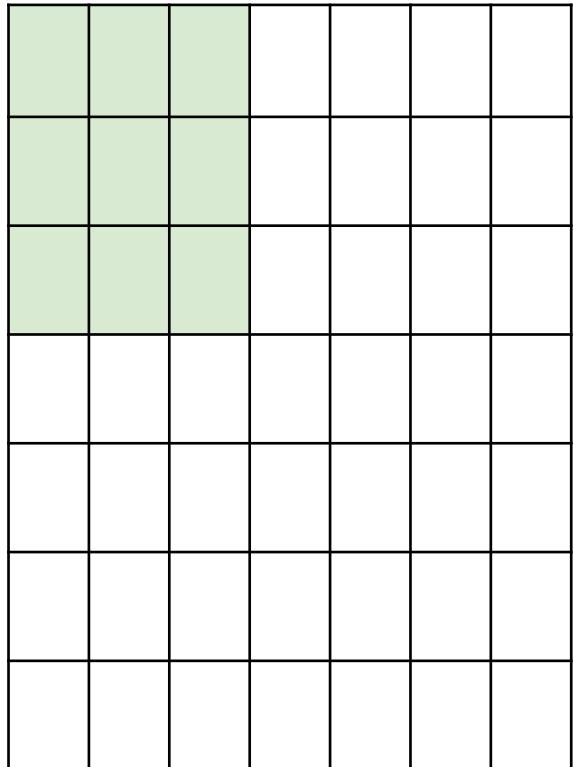


7x7 input

assume 3x3 connectivity, stride 1

=> **5x5 output**

filters: stride



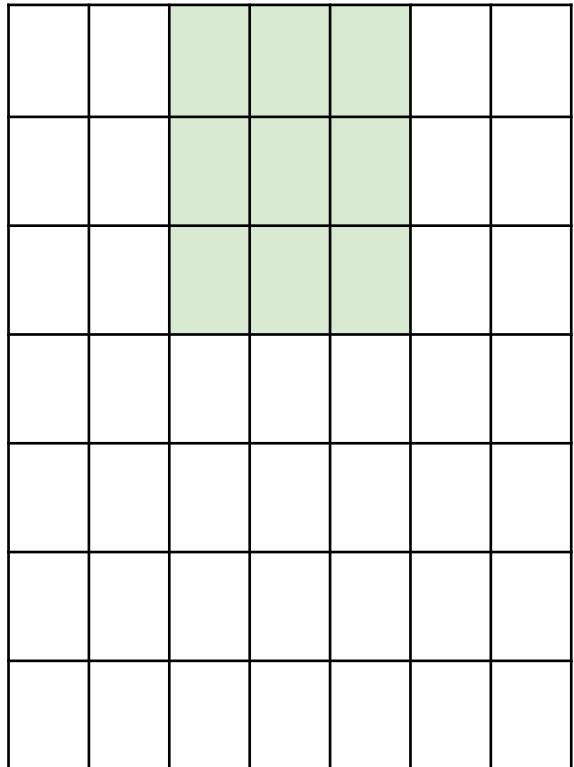
7x7 input

assume 3x3 connectivity, stride 1

=> **5x5 output**

what about stride 2?

filters: stride



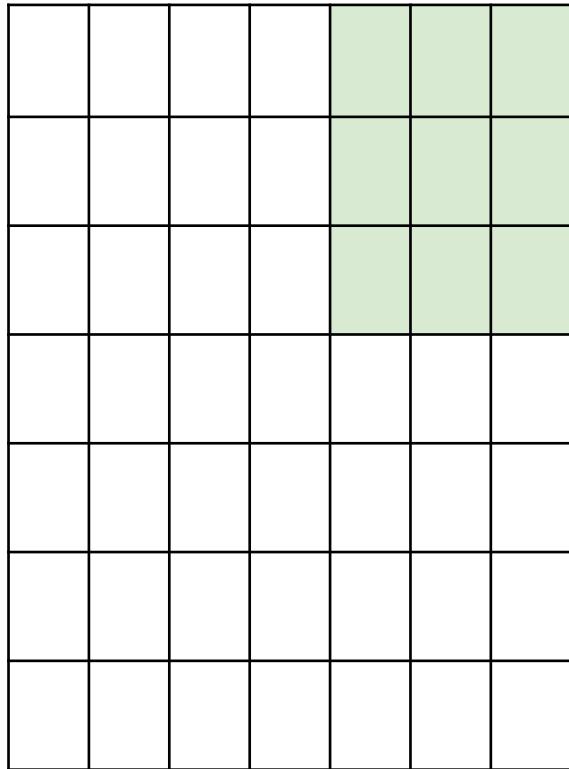
7x7 input

assume 3x3 connectivity, stride 1

=> **5x5 output**

what about stride 2?

filters: stride



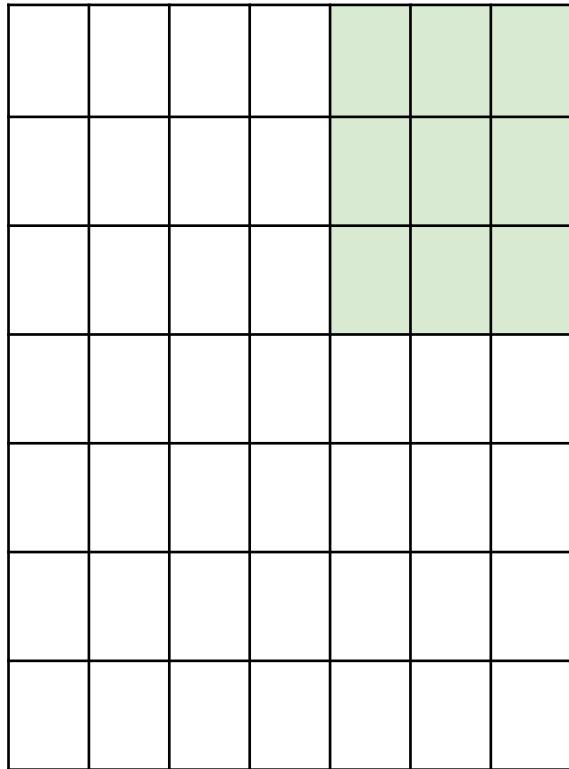
7x7 input

assume 3x3 connectivity, stride 1

=> **5x5 output**

what about stride 2?

filters: stride



7x7 input

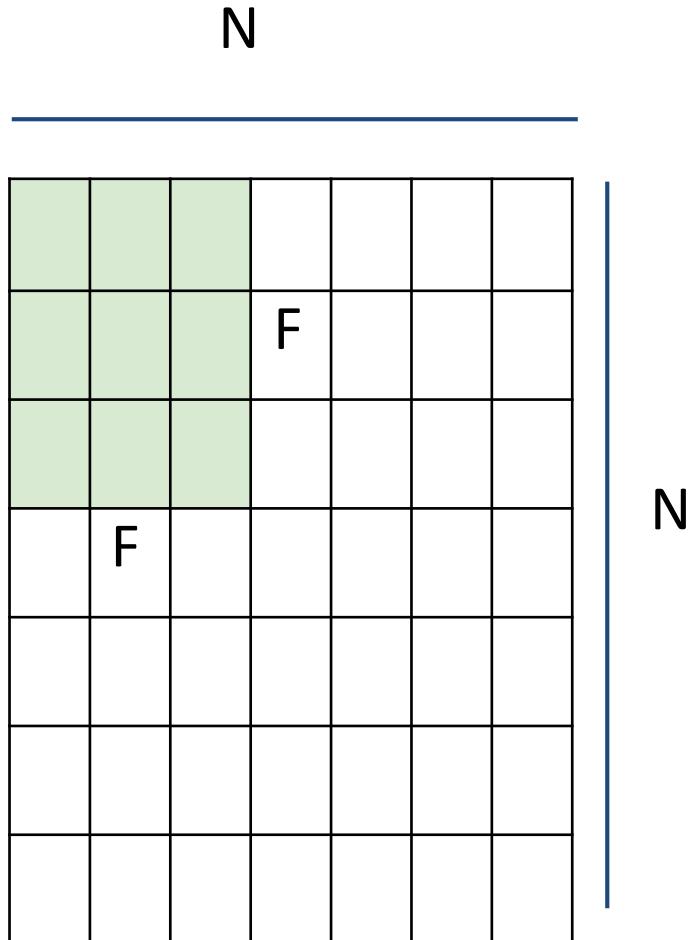
assume 3x3 connectivity, stride 1

=> **5x5 output**

what about stride 2?

=> **3x3 output**

filters: stride



Output size:
 $(N - F) / \text{stride} + 1$

e.g. $N = 7$, $F = 3$:
stride 1 $\Rightarrow (7 - 3)/1 + 1 = 5$
stride 2 $\Rightarrow (7 - 3)/2 + 1 = 3$

filters: padding

- In practice: Common to zero pad the border

0	0	0	0	0	0		
0							
0							
0							
0							

e.g. input 7x7

neuron with receptive field 3x3, stride 1
pad with 1 pixel border => what is the
output?

7x7 => preserved size!

Filters in practise

- “Same convolution”
(preserves size)

Input [9x9]

3x3 neurons, stride 1, pad **1** =>
[9x9]

- No headaches when sizing architectures
- Works well

- “Valid convolution”
(shrinks size)

Input [9x9]

3x3 neurons, stride 1, pad **0** =>
[7x7]

- **Headaches** with sizing the full architecture
- Works Worse!

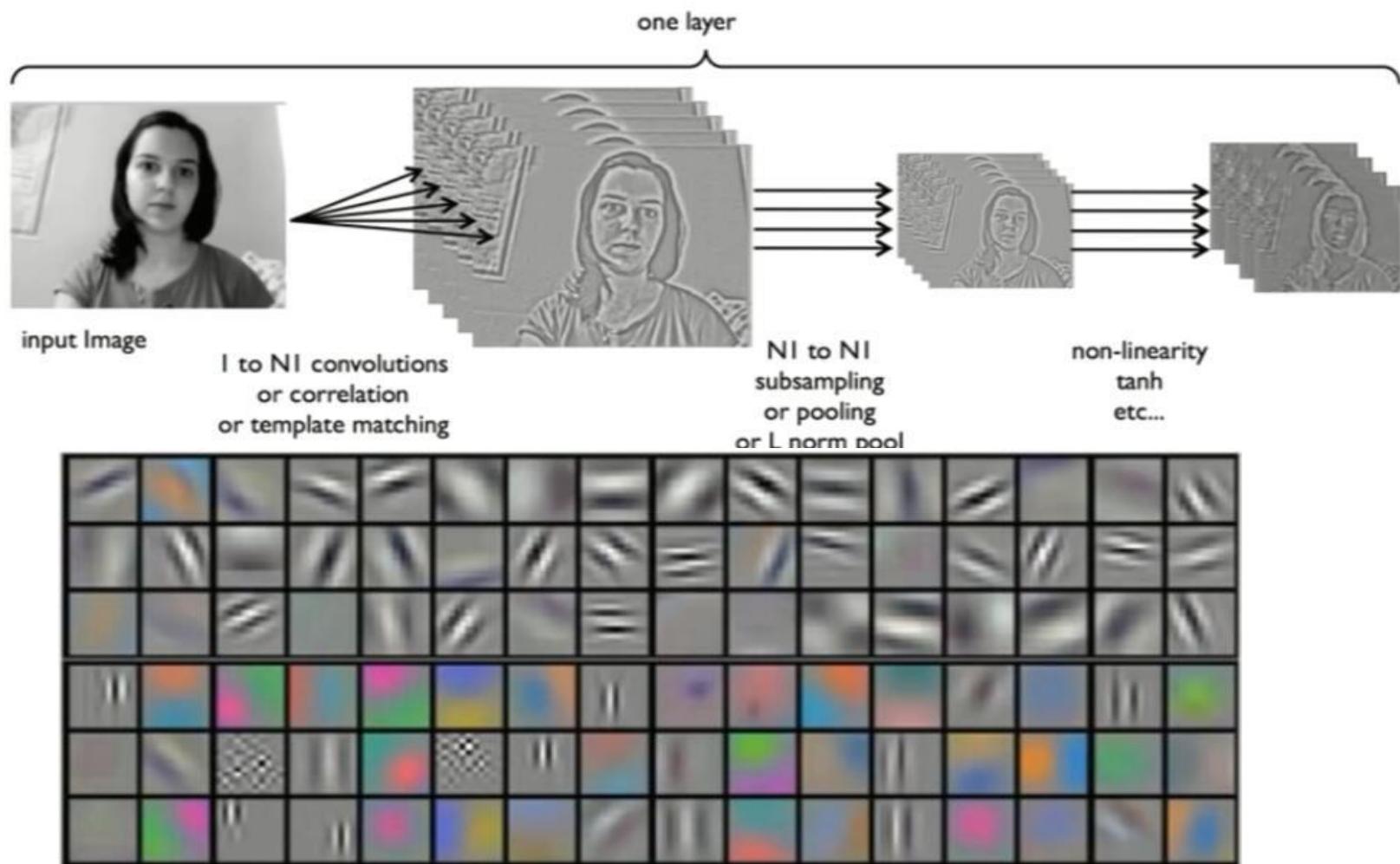
Source: Stanford CS231n,
Andrej Karpathy & Fei-Fei Li

outline

- Modeling of CNN
 - Module-wise architecture 模块化结构
- Convolutional layer (module)
 - Convolution in general
 - Filters
 - Convolution module
- Pooling layer (module)

Feature detection (特征检测)

- Learning filters (weights)

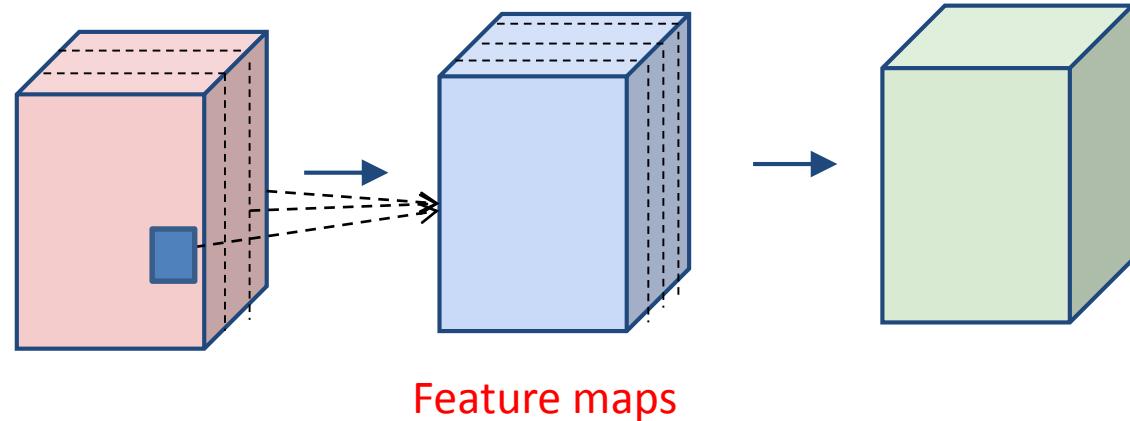


Convolution Layer (卷积层)

Input: $\mathbf{X} \in R^{d_{in} \times h \times w}$

weight: $\mathbf{W} \in R^{d_{out} \times d_{in} \times F_h \times F_w}$

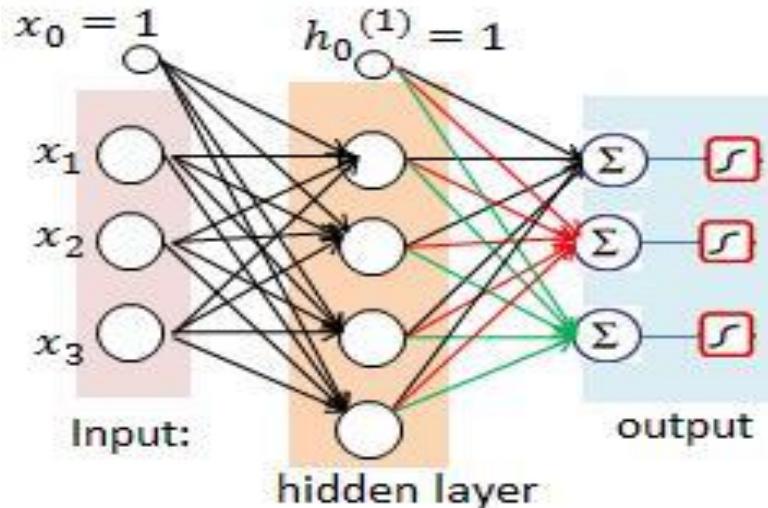
output: $\mathbf{Y} \in R^{d_{out} \times h \times w}$



Input: $\mathbf{x} \in R^{d_{in}}$

weight: $\mathbf{W} \in R^{d_{out} \times d_{in}}$

output: $\mathbf{y} \in R^{d_{out}}$

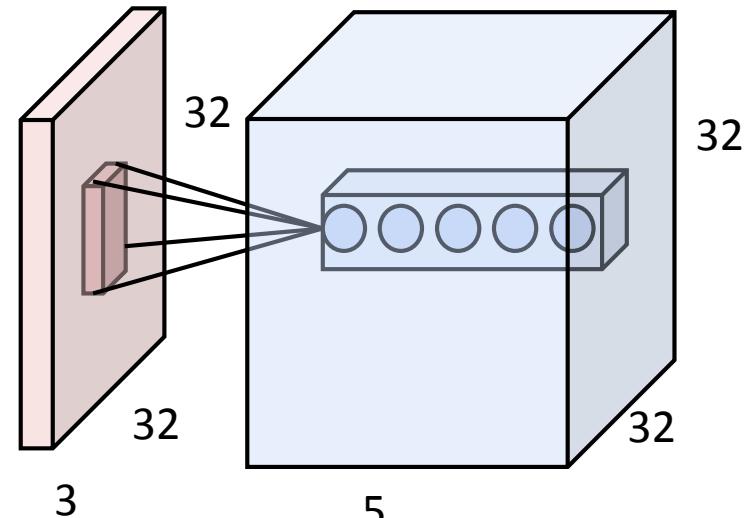


Forward (前向过程)

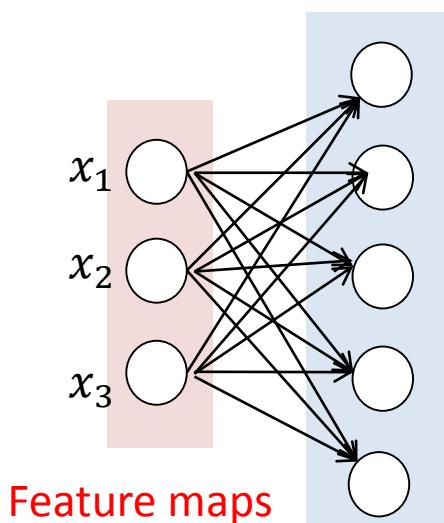
Input: $X \in R^{d_{in} \times h \times w}$

weight: $W \in R^{d_{out} \times d_{in} \times F_h \times F_w}$

output: $Y \in R^{d_{out} \times h \times w}$

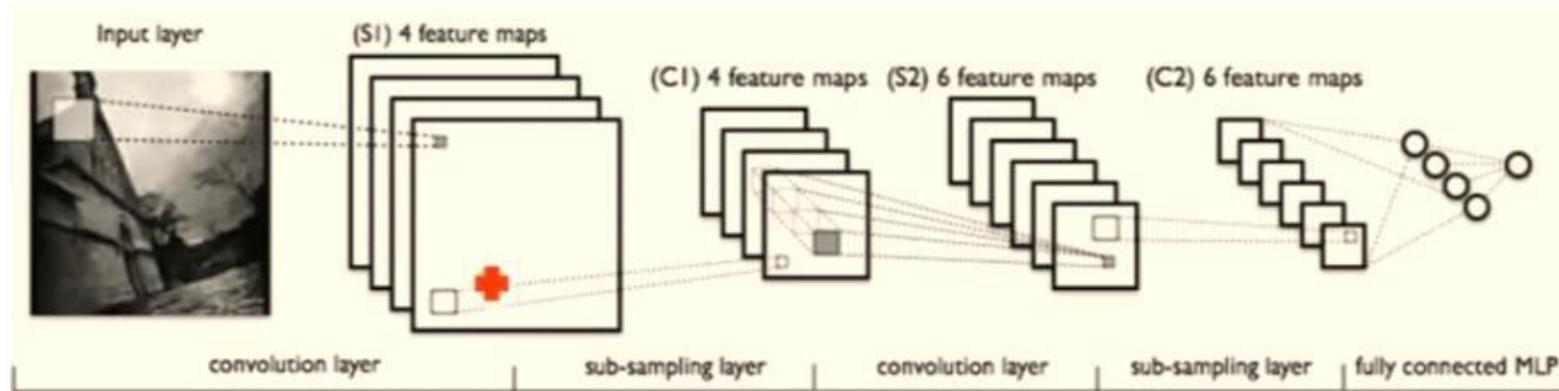


$$y_{f',i',j'} = \sum_{f=1}^{d_{in}} \sum_{i=-\frac{F_h}{2}}^{\frac{F_h}{2}} \sum_{j=-\frac{F_w}{2}}^{\frac{F_w}{2}} x_{f,i'+i-1,j'+j-1} w_{f',f,i,j} + b_{f'}$$



example

$$y_{4,10,10} = \sum_{i=-\frac{F_h}{2}}^{\frac{F_h}{2}} \sum_{j=-\frac{F_w}{2}}^{\frac{F_w}{2}} x_{1,10+i-1,10+j-1} w_{4,1,i,j} + b_4$$



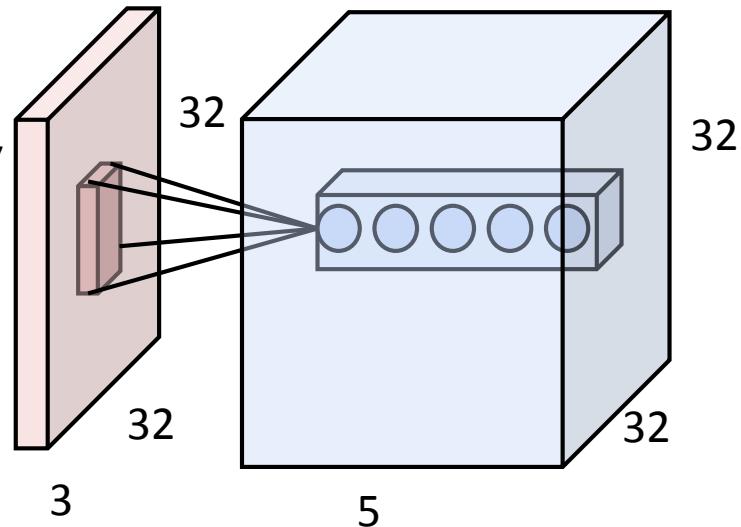
$$\begin{aligned}
 y_{4,10,100} &= \sum_{i=-\frac{F_h}{2}}^{\frac{F_h}{2}} \sum_{j=-\frac{F_w}{2}}^{\frac{F_w}{2}} x_{1,10+i-1,100+j-1} w_{4,1,i,j} + \sum_{i=-\frac{F_h}{2}}^{\frac{F_h}{2}} \sum_{j=-\frac{F_w}{2}}^{\frac{F_w}{2}} x_{2,10+i-1,100+j-1} w_{4,2,i,j} + \\
 &\quad \sum_{i=-\frac{F_h}{2}}^{\frac{F_h}{2}} \sum_{j=-\frac{F_w}{2}}^{\frac{F_w}{2}} x_{3,10+i-1,100+j-1} w_{4,3,i,j} + \sum_{i=-\frac{F_h}{2}}^{\frac{F_h}{2}} \sum_{j=-\frac{F_w}{2}}^{\frac{F_w}{2}} x_{4,10+i-1,100+j-1} w_{4,4,i,j} + b_4
 \end{aligned}$$

Back-propagation

$$y_{f',i',j'} = \sum_{f=1}^{d_{in}} \sum_{i=-\frac{F_h}{2}}^{\frac{F_h}{2}} \sum_{j=-\frac{F_w}{2}}^{\frac{F_w}{2}} x_{f,i'+i-1,j'+j-1} w_{f',f,i,j} + b_{f'}$$

$$\frac{dL}{dx_{f,i,j}} = \sum_{f'=1}^{d_{out}} \sum_{i'=1}^h \sum_{j'=1}^w \frac{dL}{dy_{f',i',j'}} \frac{dy_{f',i',j'}}{dx_{f,i,j}}$$

$$\sum_{f'=1}^{d_{out}} \sum_{i'=1}^h \sum_{j'=1}^w \frac{dL}{dy_{f',i',j'}} w_{f',f,i-i'+1,j-j'+1}$$



Input: $\mathbf{X} \in R^{d_{in} \times h \times w}$

$$\frac{dL}{w_{f',f,i,j}} = \sum_{f'=1}^{d_{out}} \sum_{i'=1}^h \sum_{j'=1}^w \frac{dL}{dy_{f',i',j'}} \frac{dy_{f',i',j'}}{w_{f',f,i,j}}$$

$$\sum_{f'=1}^{d_{out}} \sum_{i'=1}^h \sum_{j'=1}^w \frac{dL}{dy_{f',i',j'}} x_{f,i'+i-1,j'+j-1}$$

weight: $\mathbf{W} \in R^{d_{out} \times d_{in} \times F_h \times F_w}$

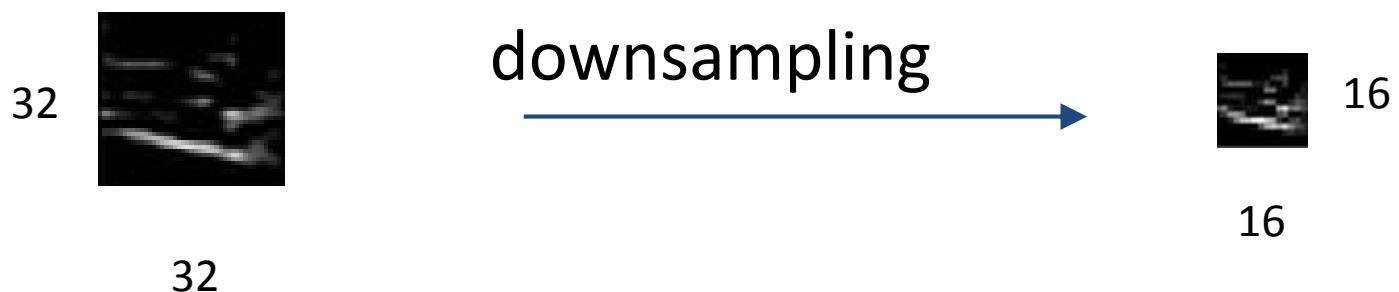
output: $\mathbf{Y} \in R^{d_{out} \times h \times w}$

outline

- Modeling of CNN
 - Module-wise architecture 模块化结构
- Convolutional layer (module)
 - Convolution in general
 - Filters
 - Convolution module
- Pooling layer (module)

POOLING Layer

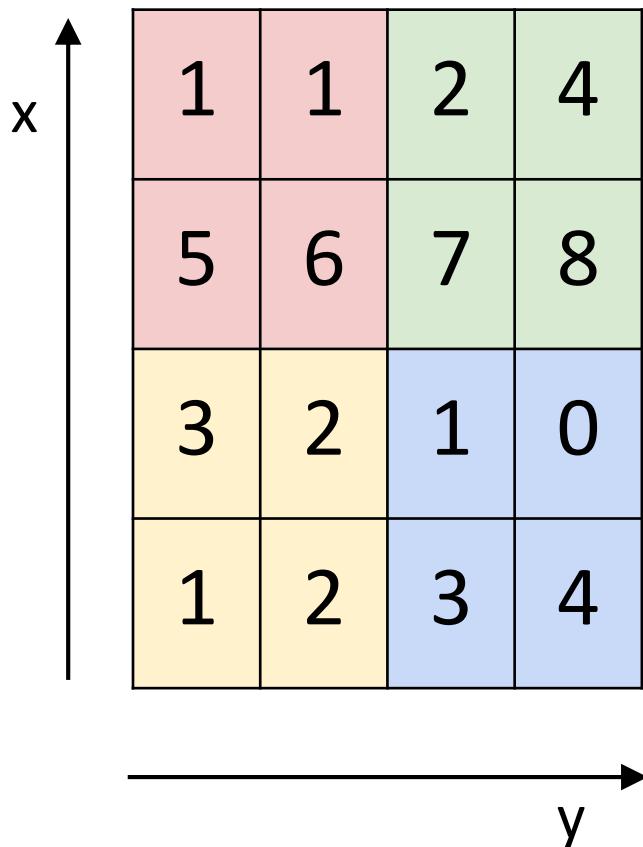
- In ConvNet architectures, **Conv** layers are often followed by **Pooling** layers
 - makes the representations smaller and more manageable without losing too much information.
 - Invariant in region.



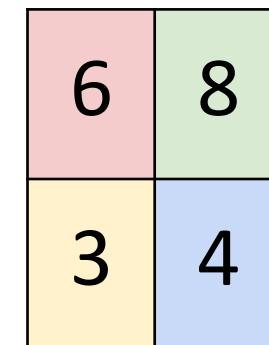
Source: Stanford CS231n,
Andrej Karpathy & Fei-Fei Li

MAX POOLING

Single depth slice



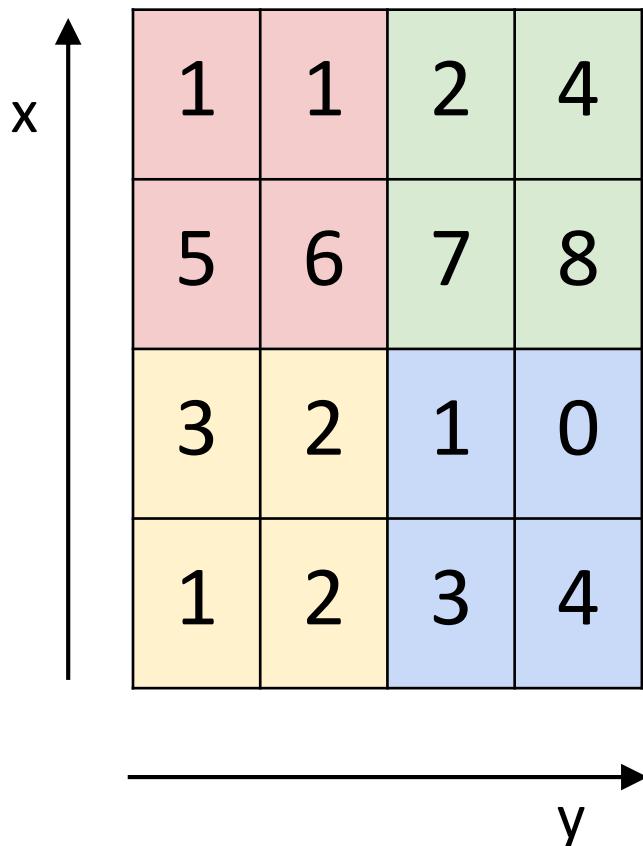
max pool with 2x2 filters
and stride 2



Source: Stanford CS231n,
Andrej Karpathy & Fei-Fei Li

Average POOLING

Single depth slice

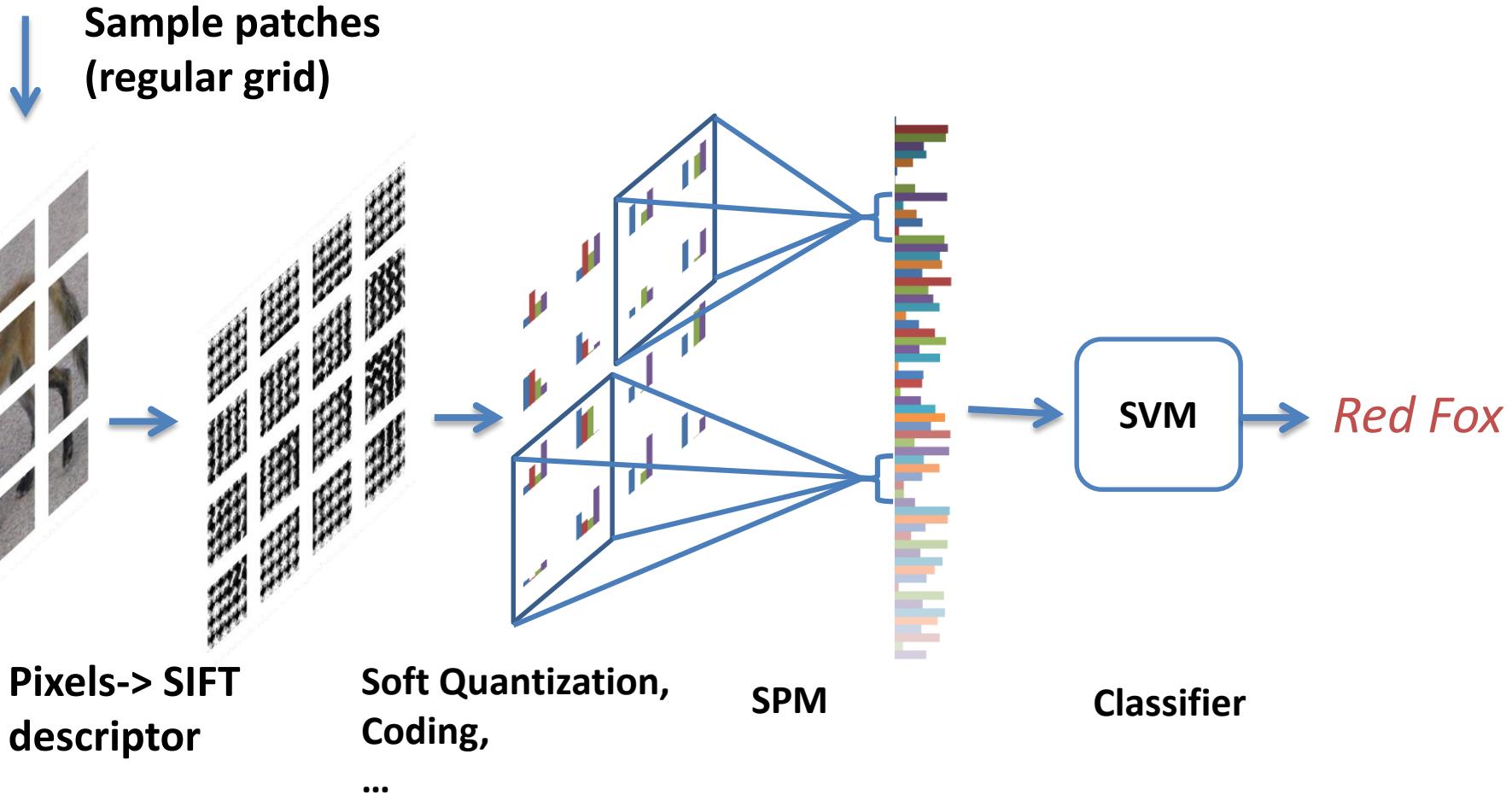


average pool with 2x2 filters and stride 2

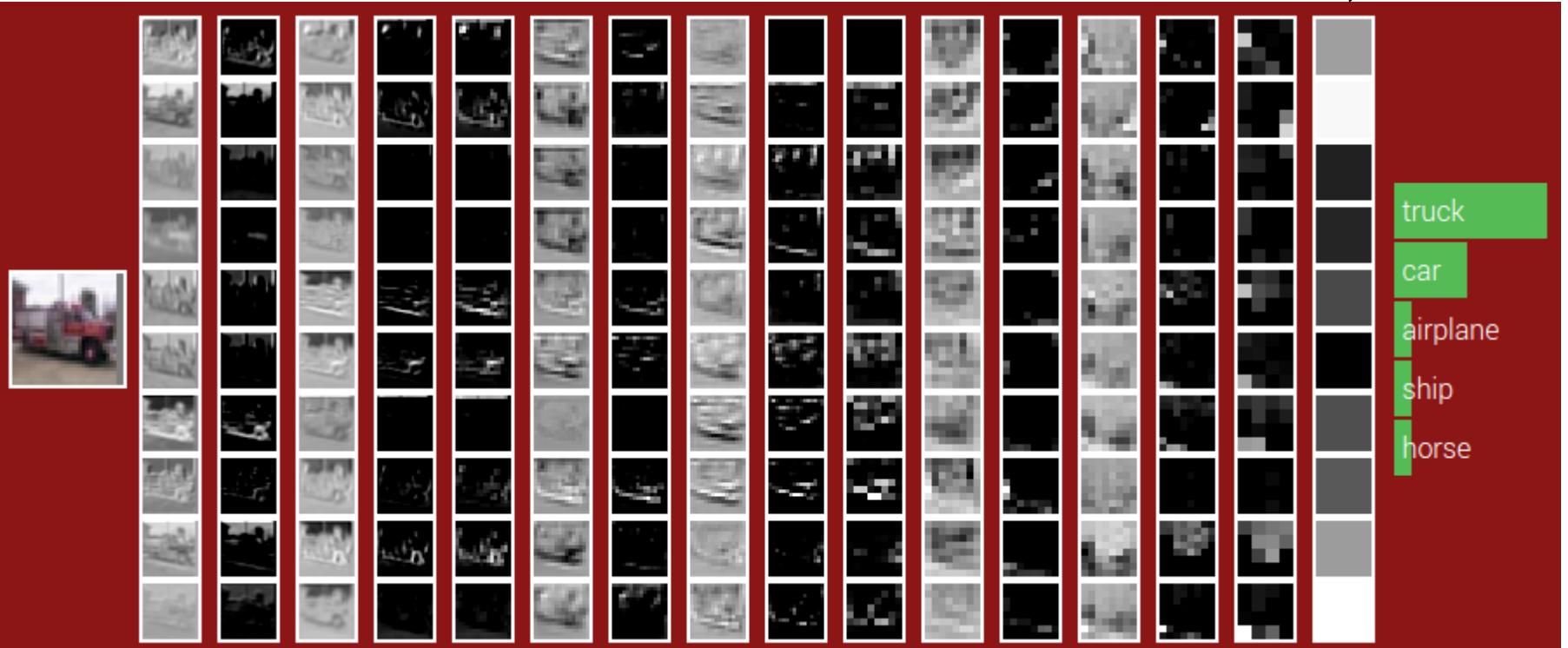
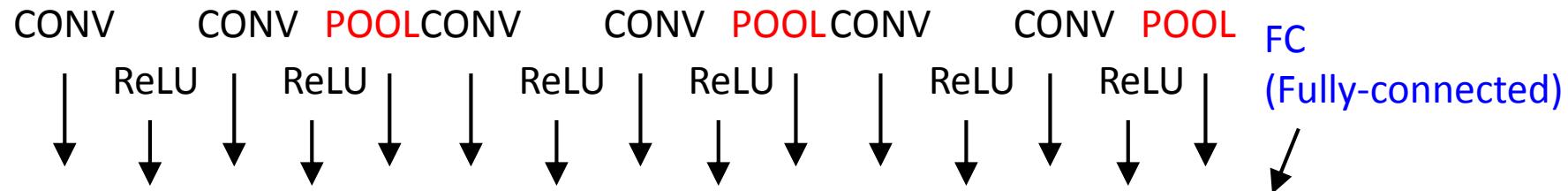


Source: Stanford CS231n,
Andrej Karpathy & Fei-Fei Li

Another Motivation of pooling



Intuitive example

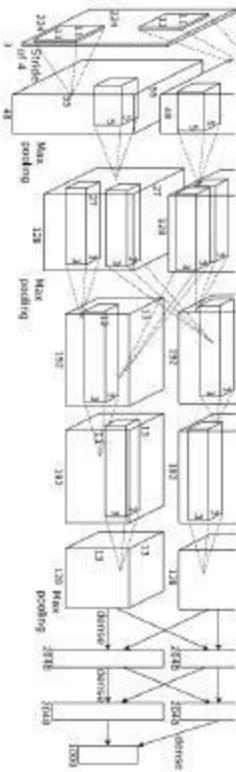


Source: Stanford CS231n,
Andrej Karpathy & Fei-Fei Li

Famous Net Architecture

Year 2012

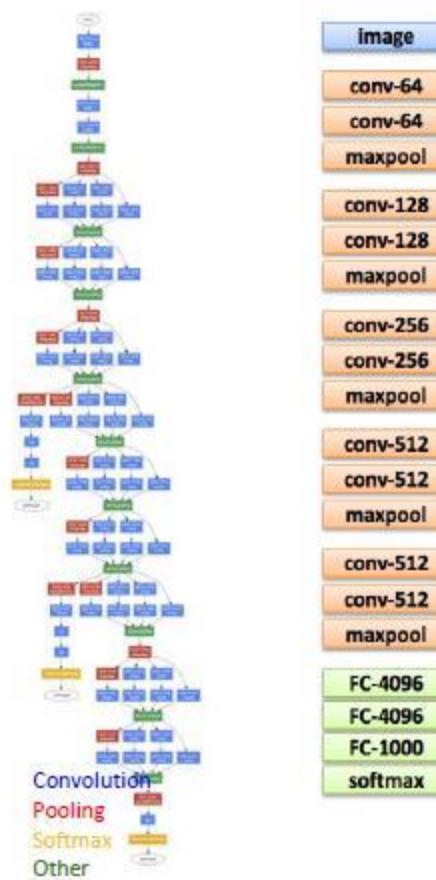
SuperVision



[Krizhevsky NIPS 2012]

Year 2014

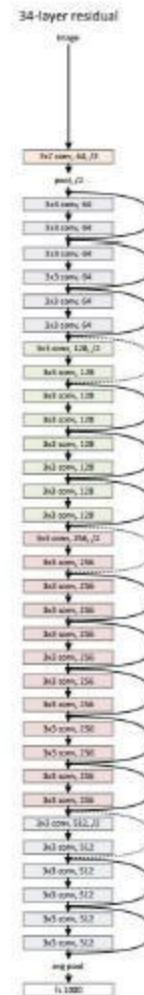
GoogLeNet



[Szegedy arxiv 2014]

Year 2015

MSRA



[Simonyan arxiv 2014]