本科生《计算机视觉》
基于深度学习的视觉理解与生成

黄雷
人工智能研究院
huangleiAI@buaa.edu.cn

2023年10月12日

主要内容

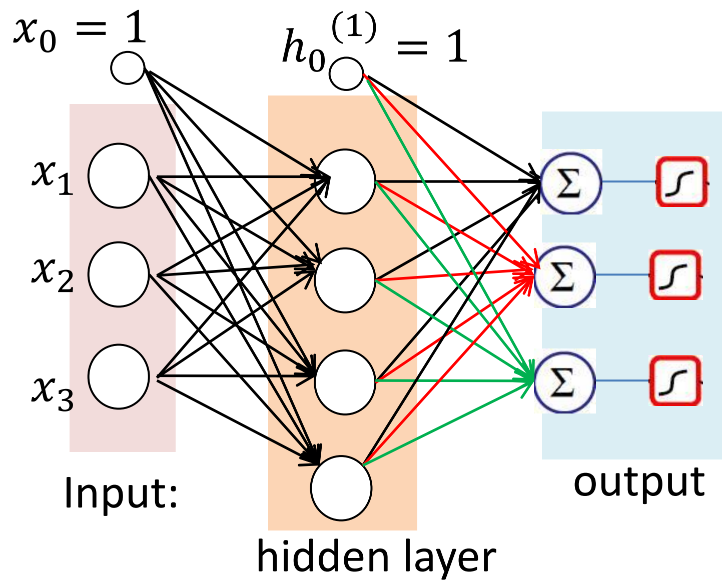
- 深度学习基础
  - 神经网络及反向传播算法
  - 卷积神经网络中的视觉表示思想
- 视觉理解任务
  - 目标检测
  - 分割
- 视觉生成
  - 深度生成模型
  - 图像翻译任务详解
- 深度神经网络训练技巧

outline

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

Feature extraction

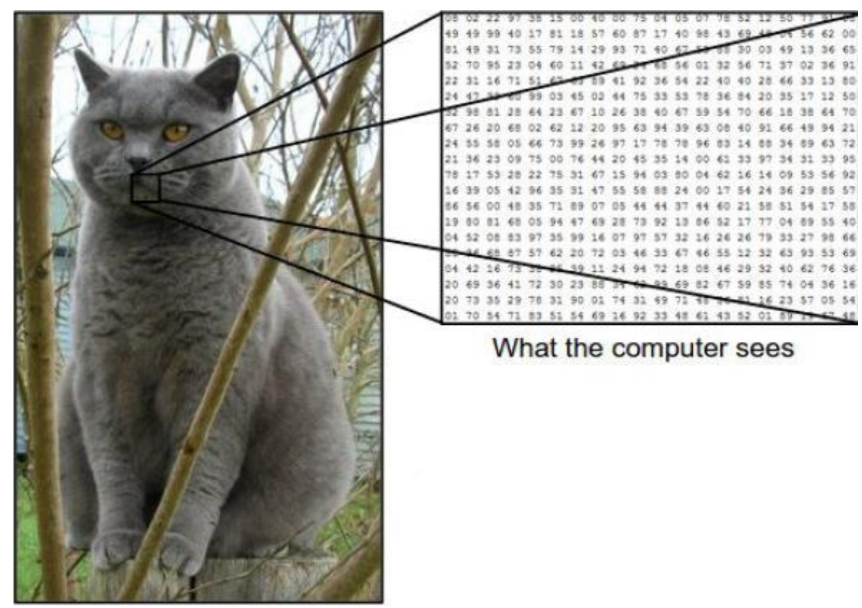
- Feature extraction
  - Pixel-wise input
  - Correlation between features



$(1, 0, 0)^T$

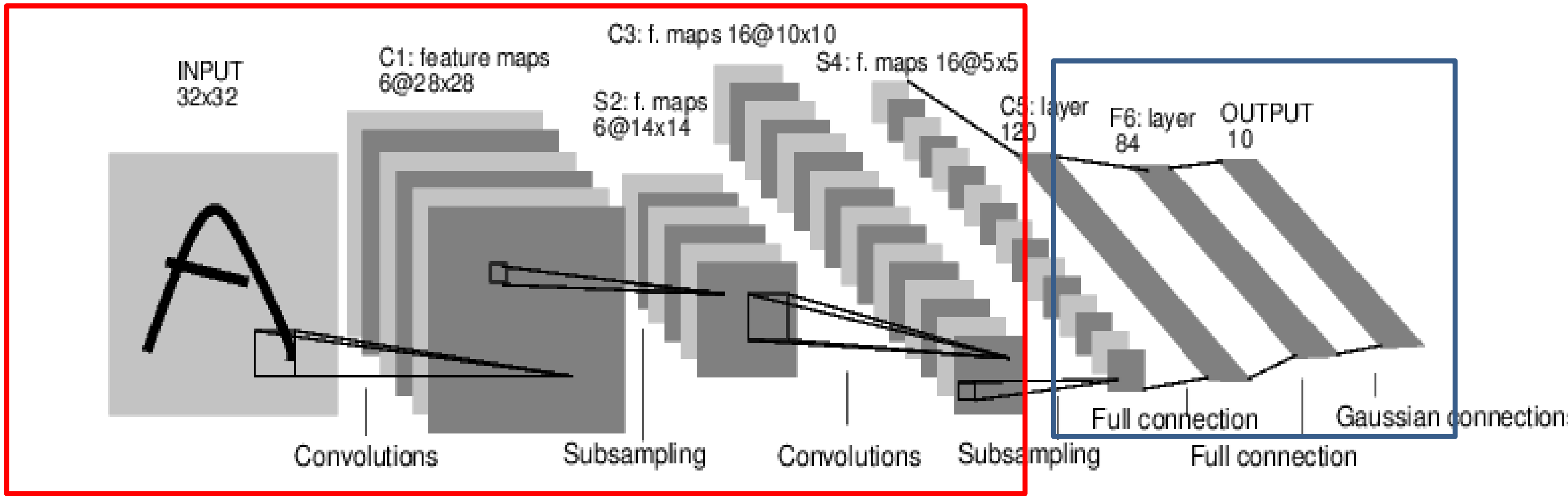
(x_1, x_2, x_3)

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



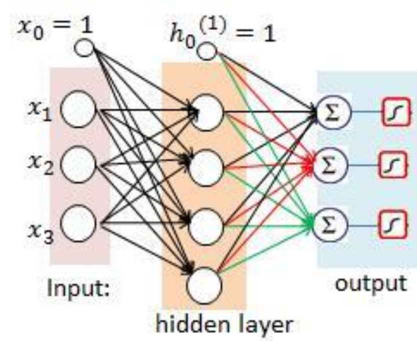
Convolution Neural Network

- Lenet-5



全连接层

Convolution related layers



outline

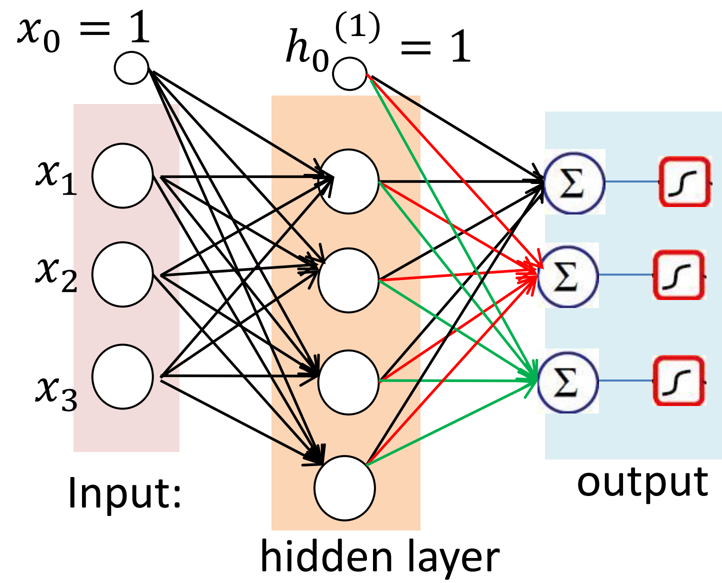
- Modeling of CNN
  - Module-wise architecture模块化结构
- Convolutional layer (module)
  - Convolution like template
  - Filters
  - Convolution module
- Pooling layer (module)

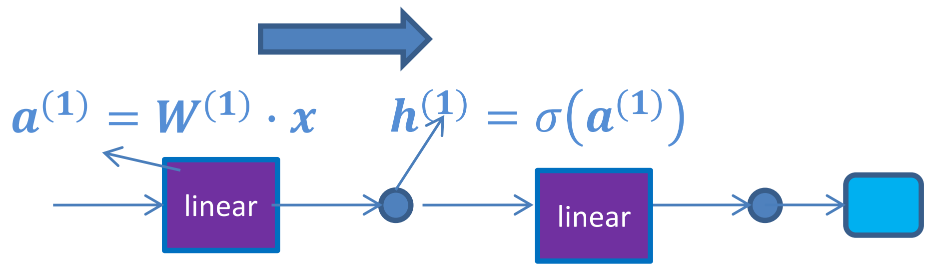
Module-wise architecture

➢ PyTorch 平台

- **Model construction**

```python
# Define model
class NeuralNetwork(nn.Module):
    def __init__(self):
        super(NeuralNetwork, self).__init__()
        self.flatten = nn.Flatten()
        self.linear_relu_stack = nn.Sequential(
            nn.Linear(28*28, 512),
            nn.ReLU(),
            nn.Linear(512, 512),
            nn.ReLU(),
            nn.Linear(512, 10)
        )

    def forward(self, x):
        x = self.flatten(x)
        logits = self.linear_relu_stack(x)
        return logits
```

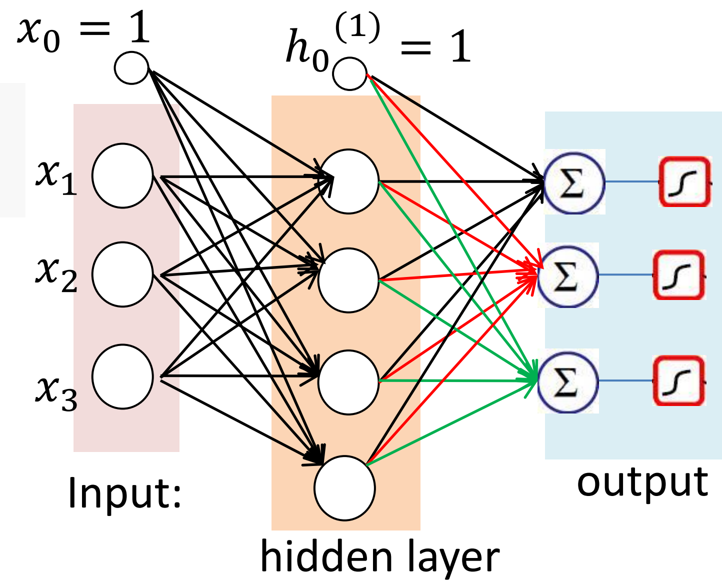




Module-wise architecture

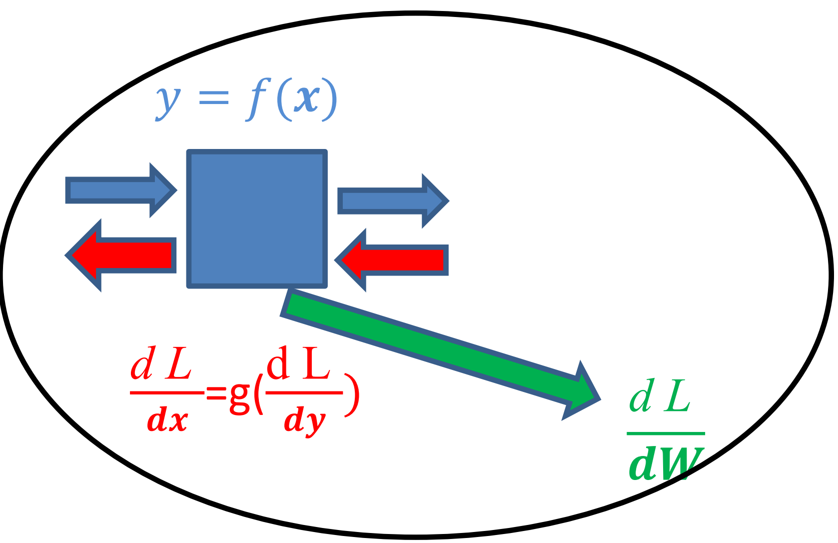
- PyTorch 平台

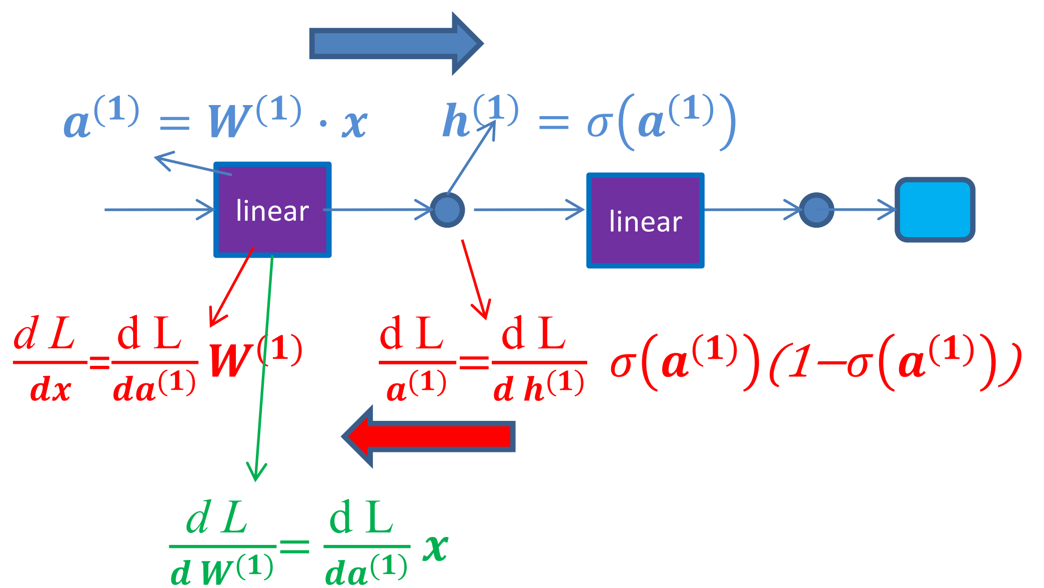
```
loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=1e-3)
```



- **Training per iteration:**

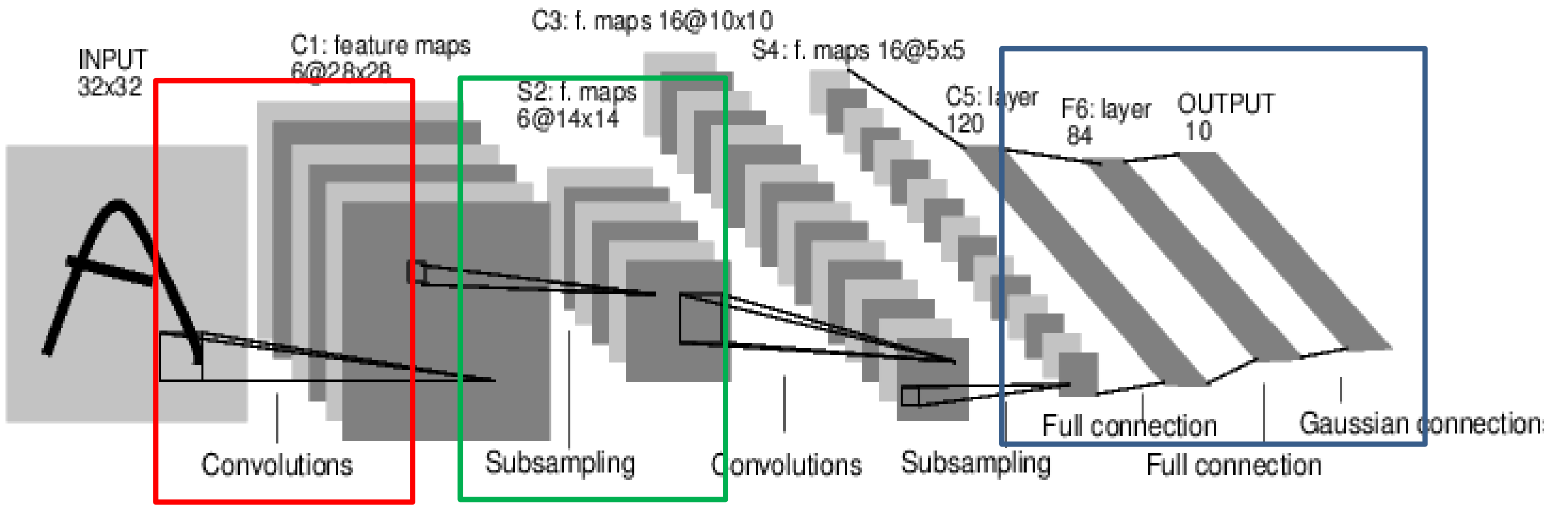
```
# Compute prediction error
pred = model(X)
loss = loss_fn(pred, y)

# Backpropagation
optimizer.zero_grad()
loss.backward()
optimizer.step()
```



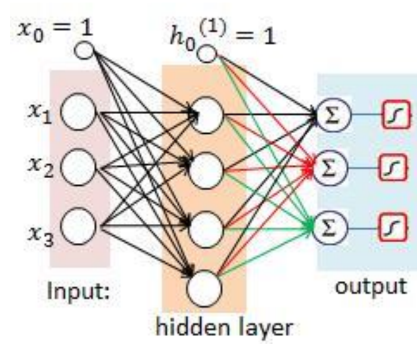


Convolution Neural Network

- Lenet-5



Convolution (卷积)   Pooling (池化)   全连接层

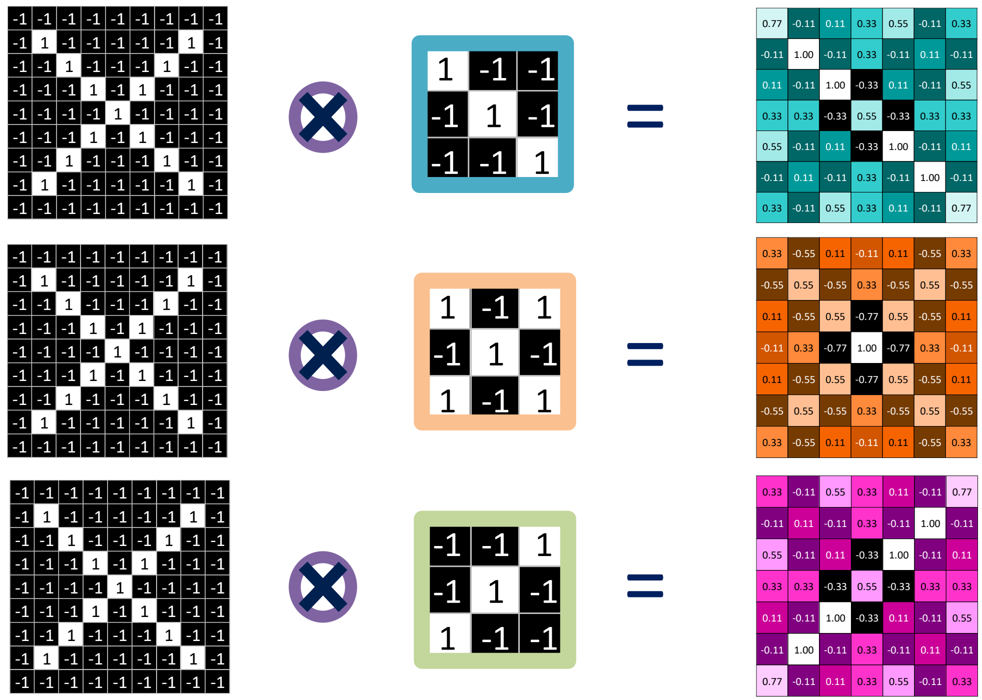


outline

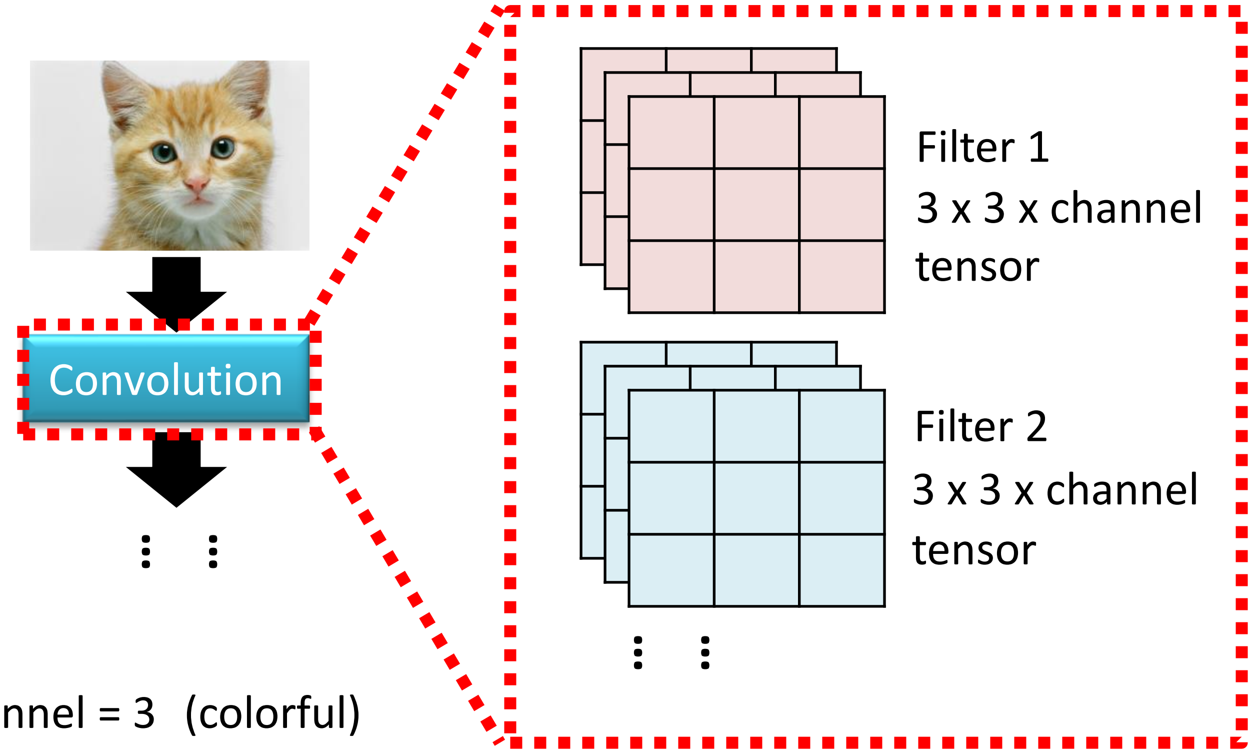
- Modeling of CNN
  - Module-wise architecture模块化结构
- Convolutional layer (module)
  - Convolution like template
  - Filters
  - Why convolution for vision
- Pooling layer (module)

Convolution

- **Match like template**



Convolutional Layer



channel = 3 (colorful)

channel = 1 (black and white)

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

Convolutional Layer

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

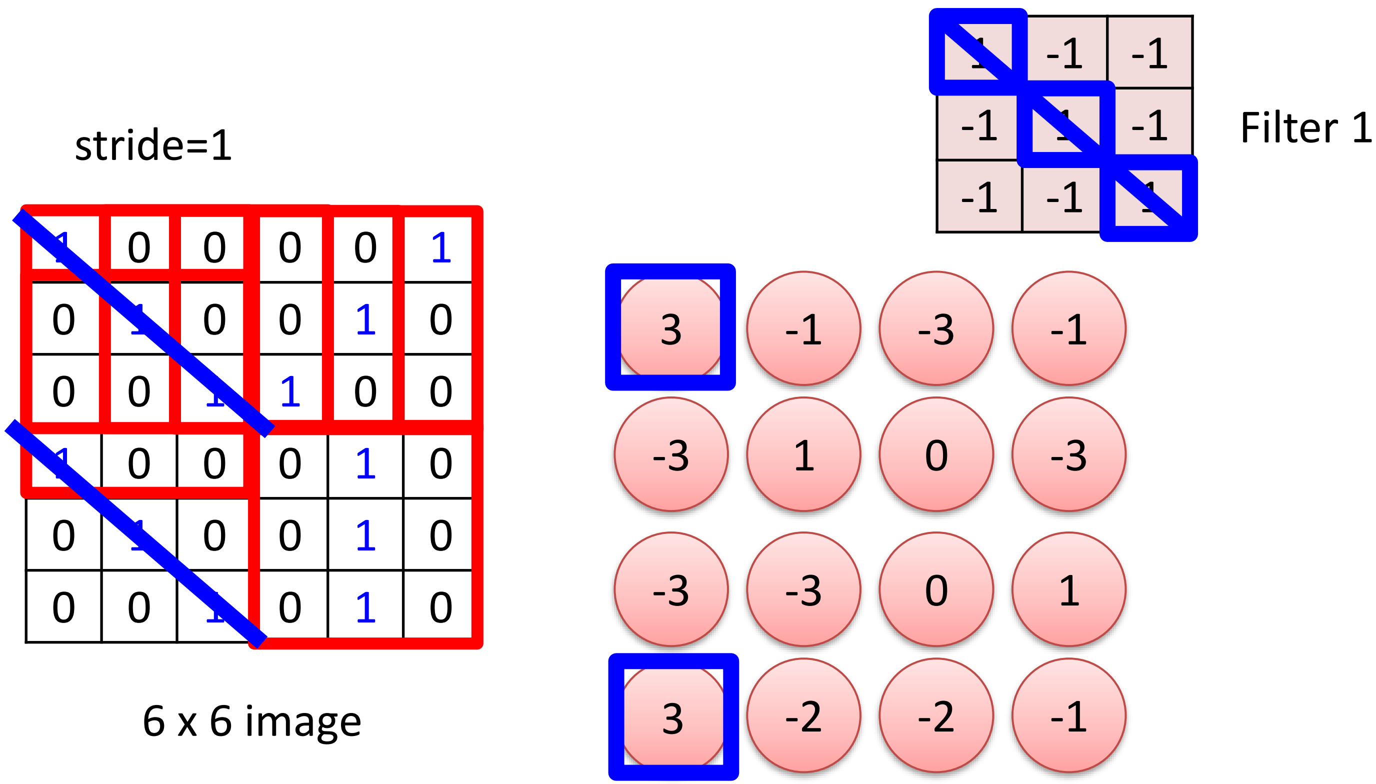
-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

⋮ ⋮

(The values in the filters are unknown parameters.)

Convolutional Layer



Convolutional Layer

stride=1

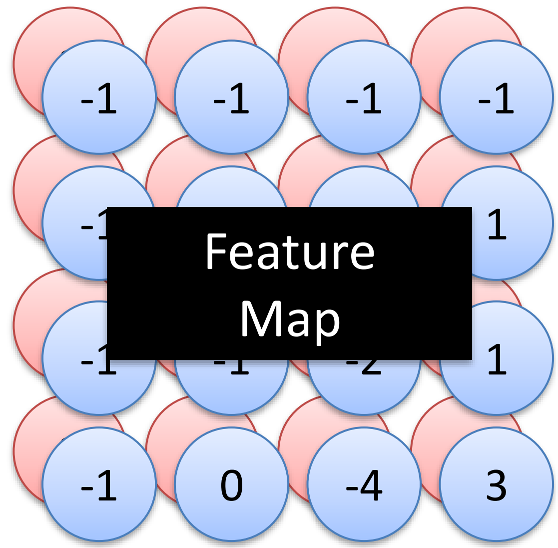
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

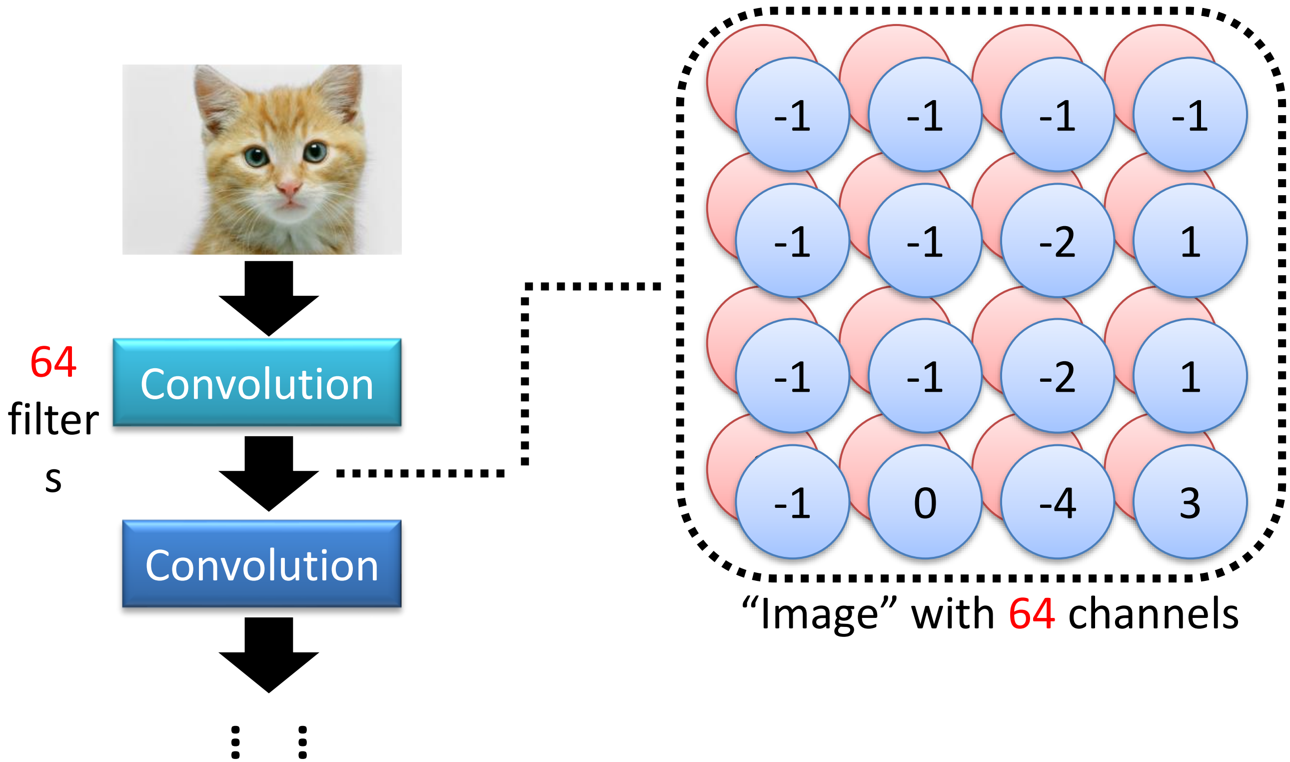
-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

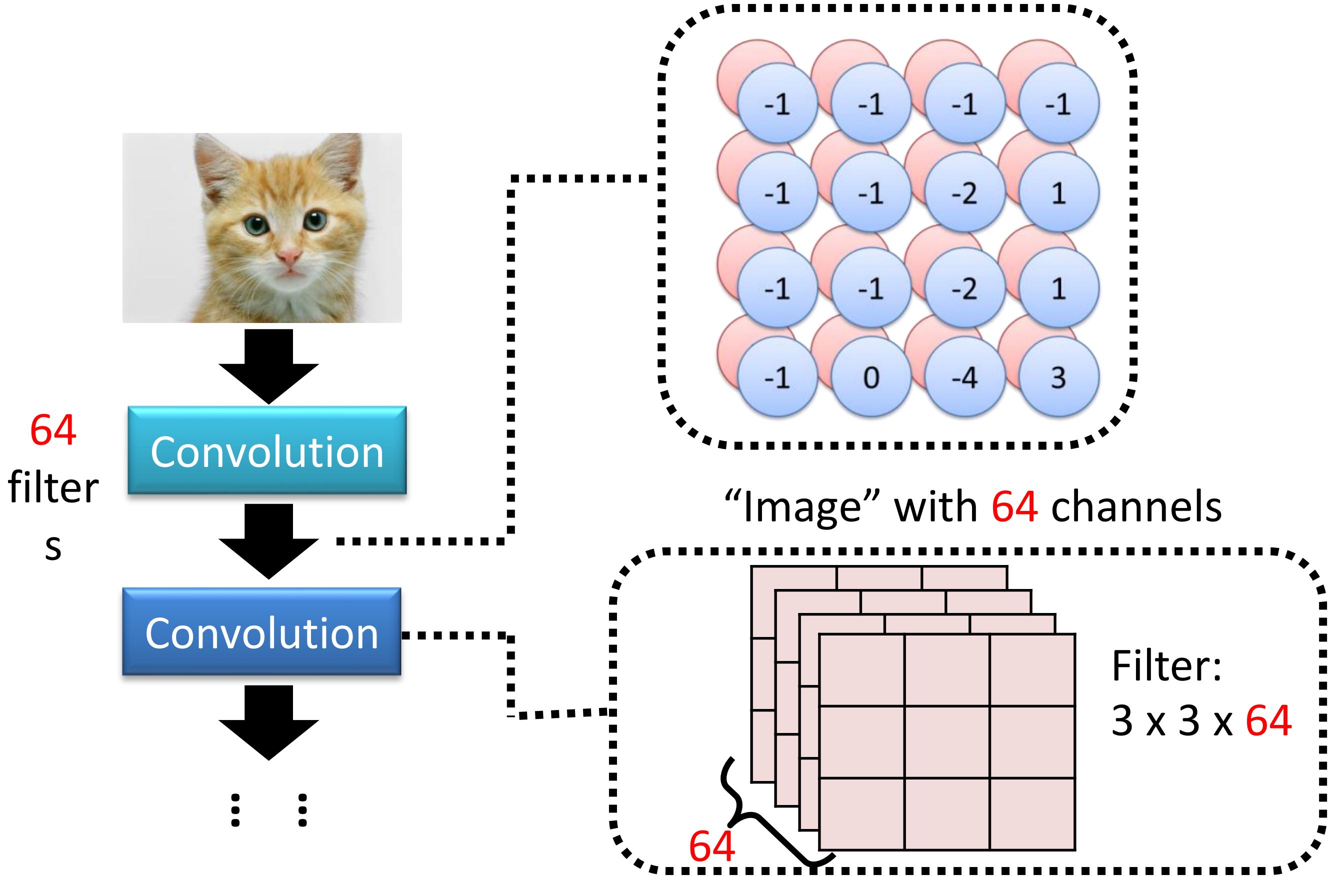
Do the same process for every filter



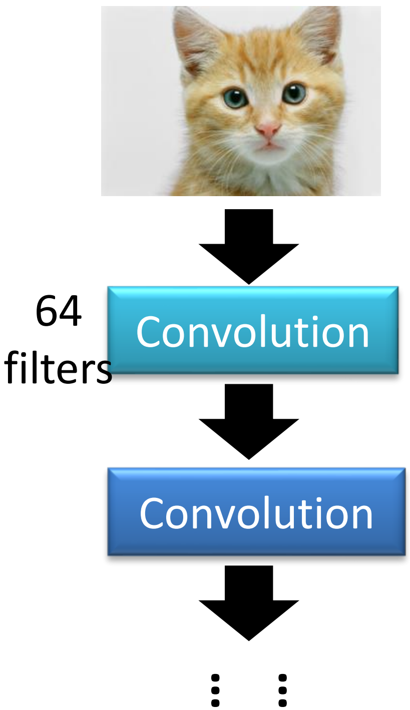
Convolutional Layer



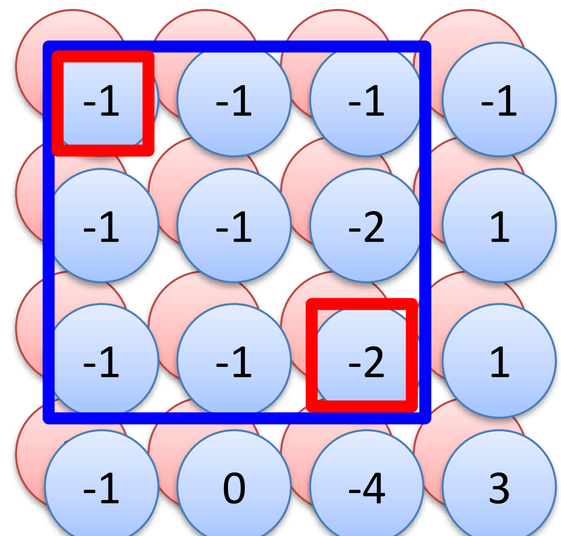
Convolutional Layer



Receptive Field（感受野）



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

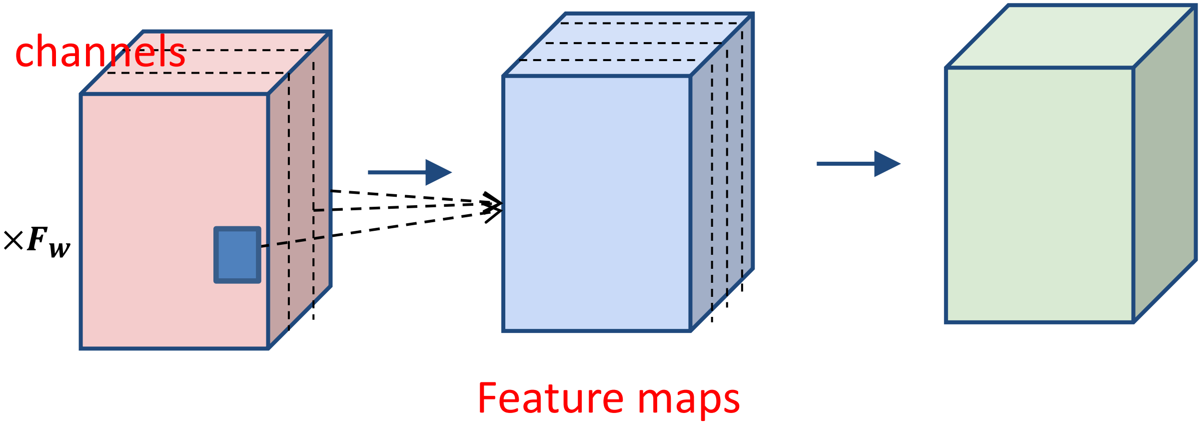


Convolution Layer (卷积层)

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

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

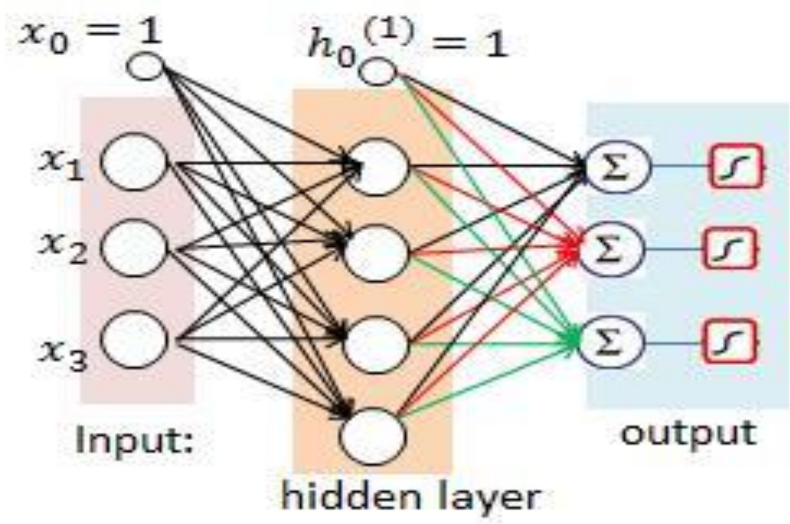
output: Y∈ $R^{d_{out}\times h\times w}$



Input: x∈ $R^{d_{in}}$

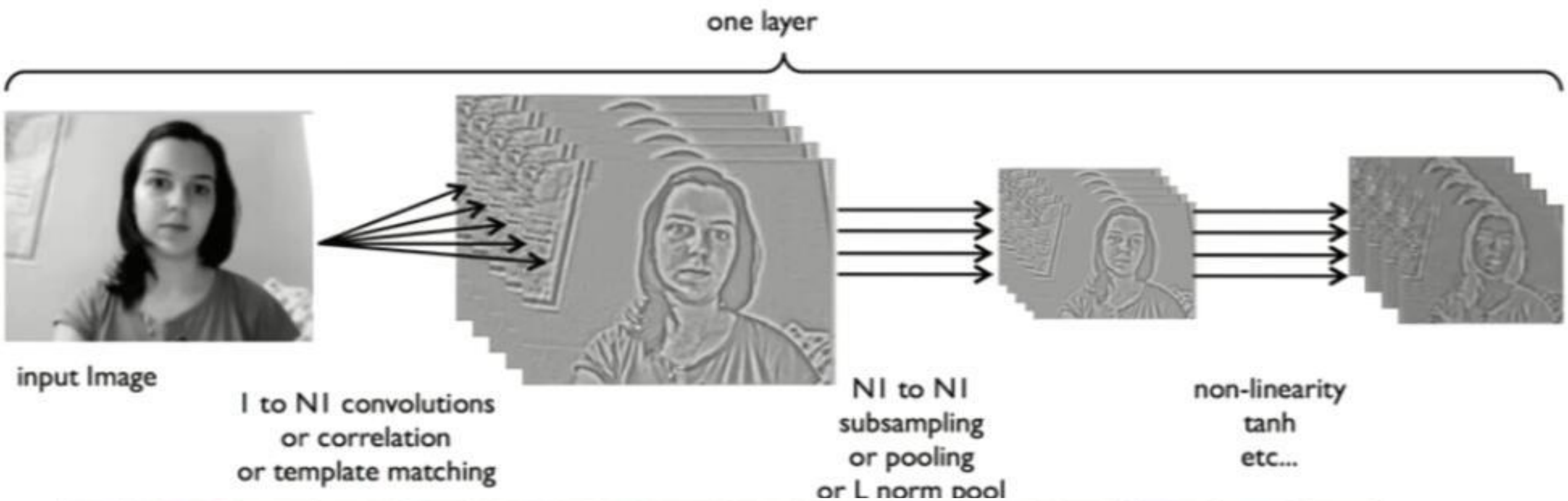
weight: W∈ $R^{d_{out}\times d_{in}}$

output: y∈ $R^{d_{out}}$



Feature detection (特征检测)

- Learning filters (weights)



outline

- Modeling of CNN
  - Module-wise architecture模块化结构
- Convolutional layer (module)
  - Convolution like template
  - Filters
  - Why convolution for vision
- Pooling layer (module)

Practice with linear filters(线性滤波器)

Original

0	0	0
0	1	0
0	0	0

Filter

Filtered
(no change)

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	0	1
0	0	0

Filter

Shifted *left*
By 1 pixel

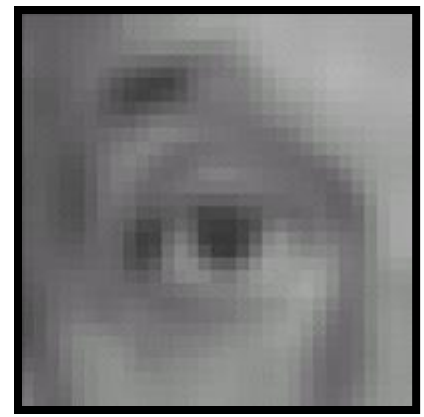
Source: D. Lowe

Practice with linear filters

Original

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Filter



Blur (with a box filter)

Source: D. Lowe

Practice with linear filters

Original

0	1	0
1	-4	1
0	1	0

Filter

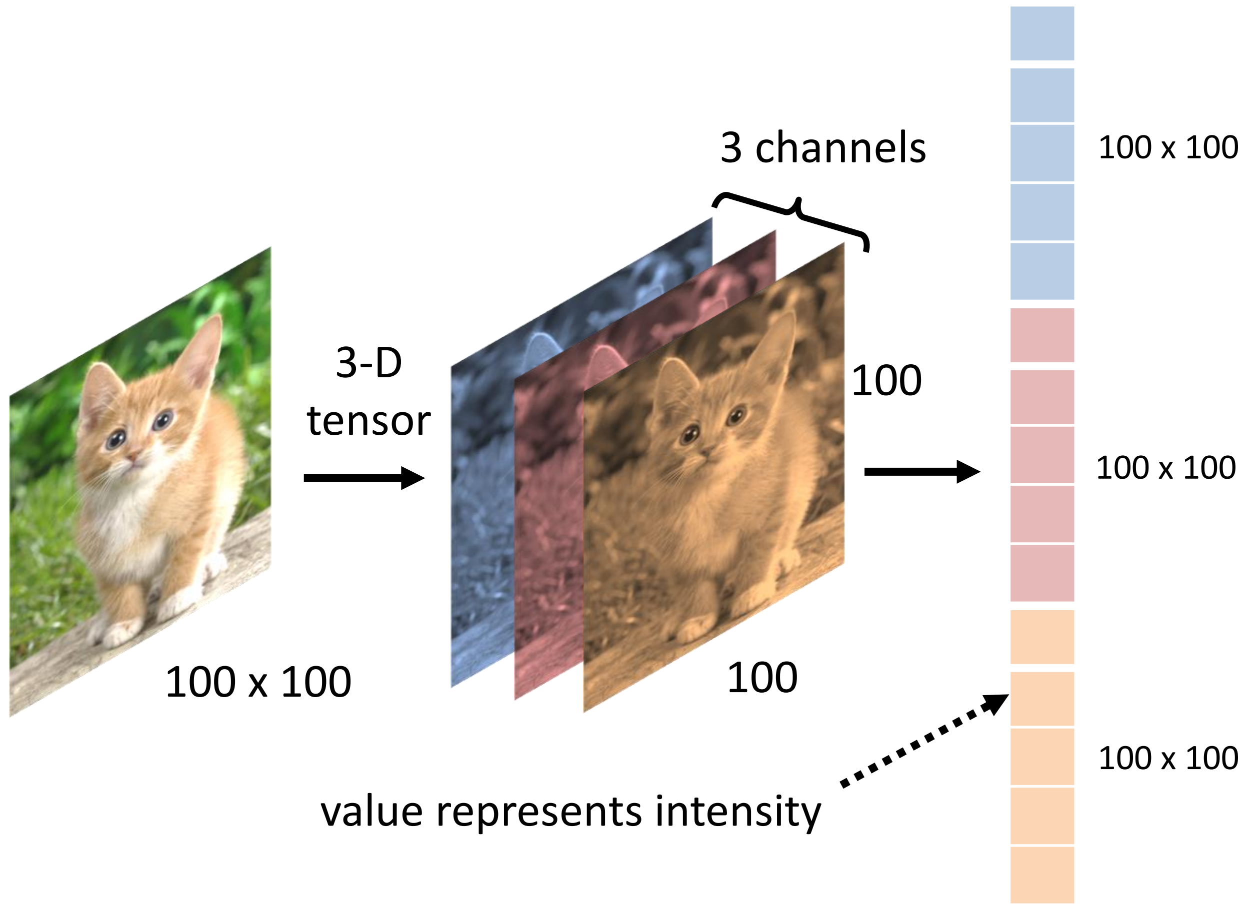
Output Image

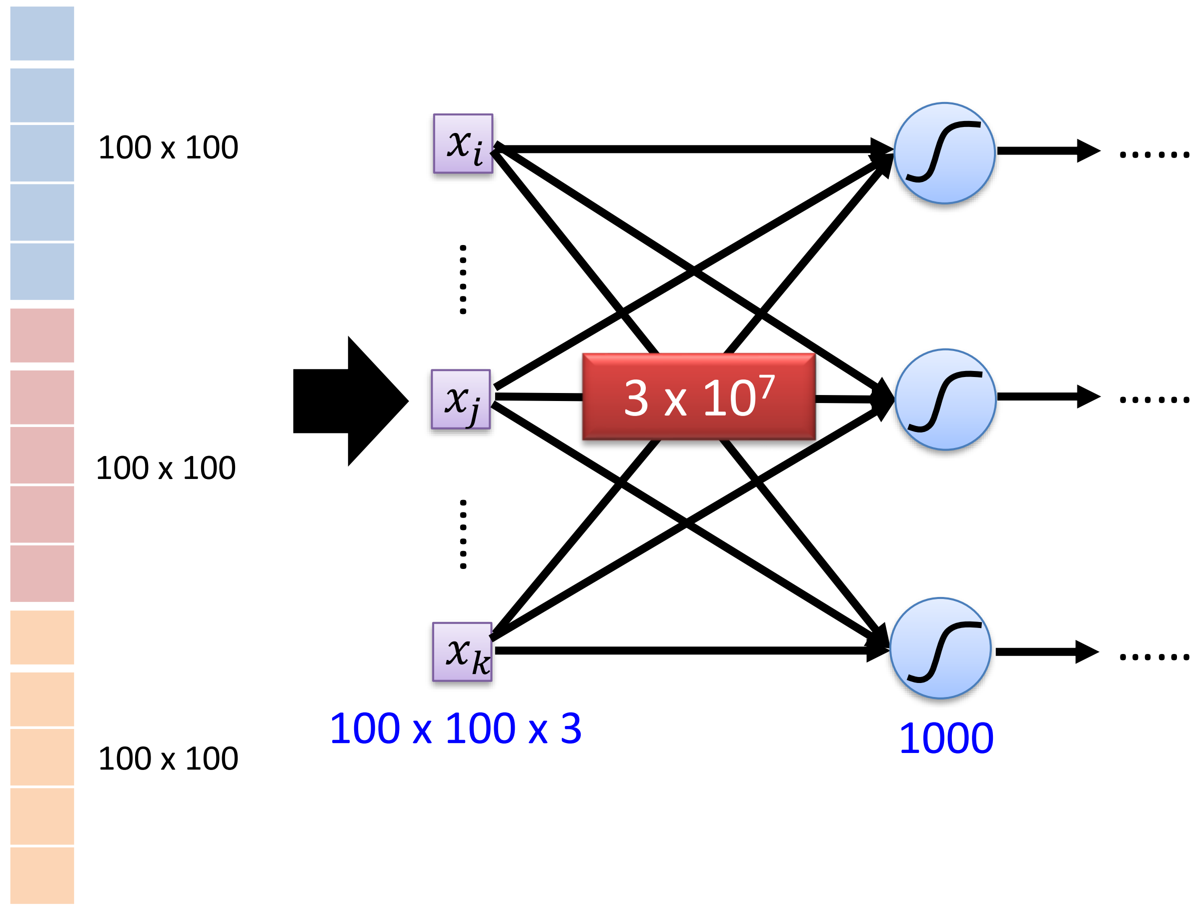
Edge detect （边缘检测）

outline

- Modeling of CNN
  - Module-wise architecture模块化结构
- Convolutional layer (module)
  - Convolution like template
  - Filters
  - Why convolution for vision
- Pooling layer (module)

Image Classification



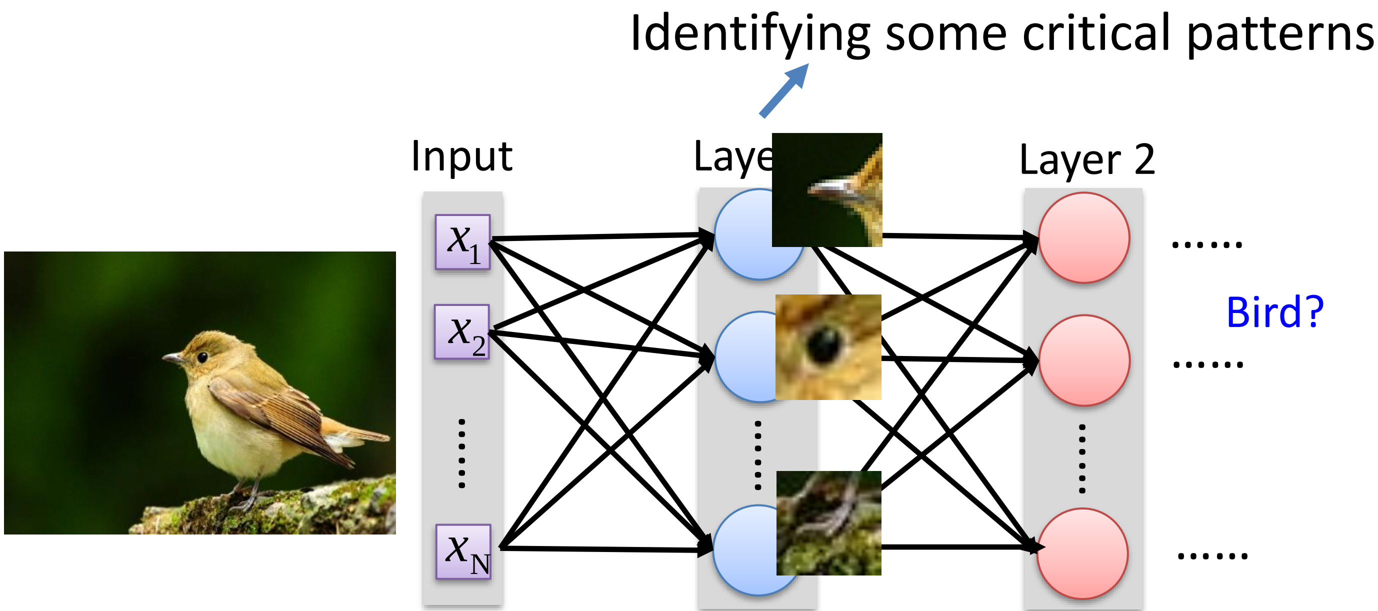

100 x 100
100 x 100
100 x 100
x_i
x_j
x_k
3 x 10^7
100 x 100 x 3
1000
……
……
……

Vision model

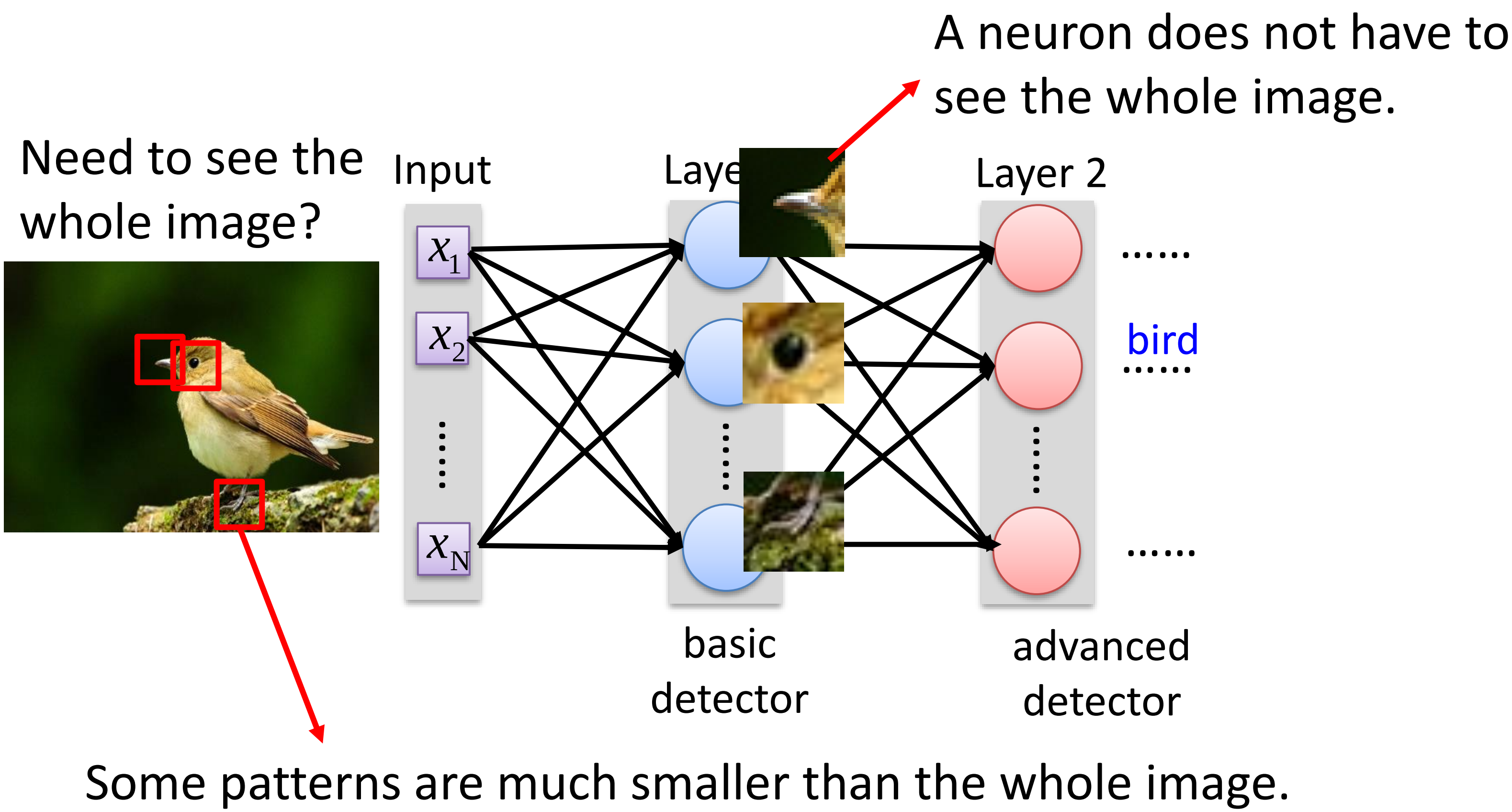
How MLP to CNN?

- Model locality
- Parameter Efficiency

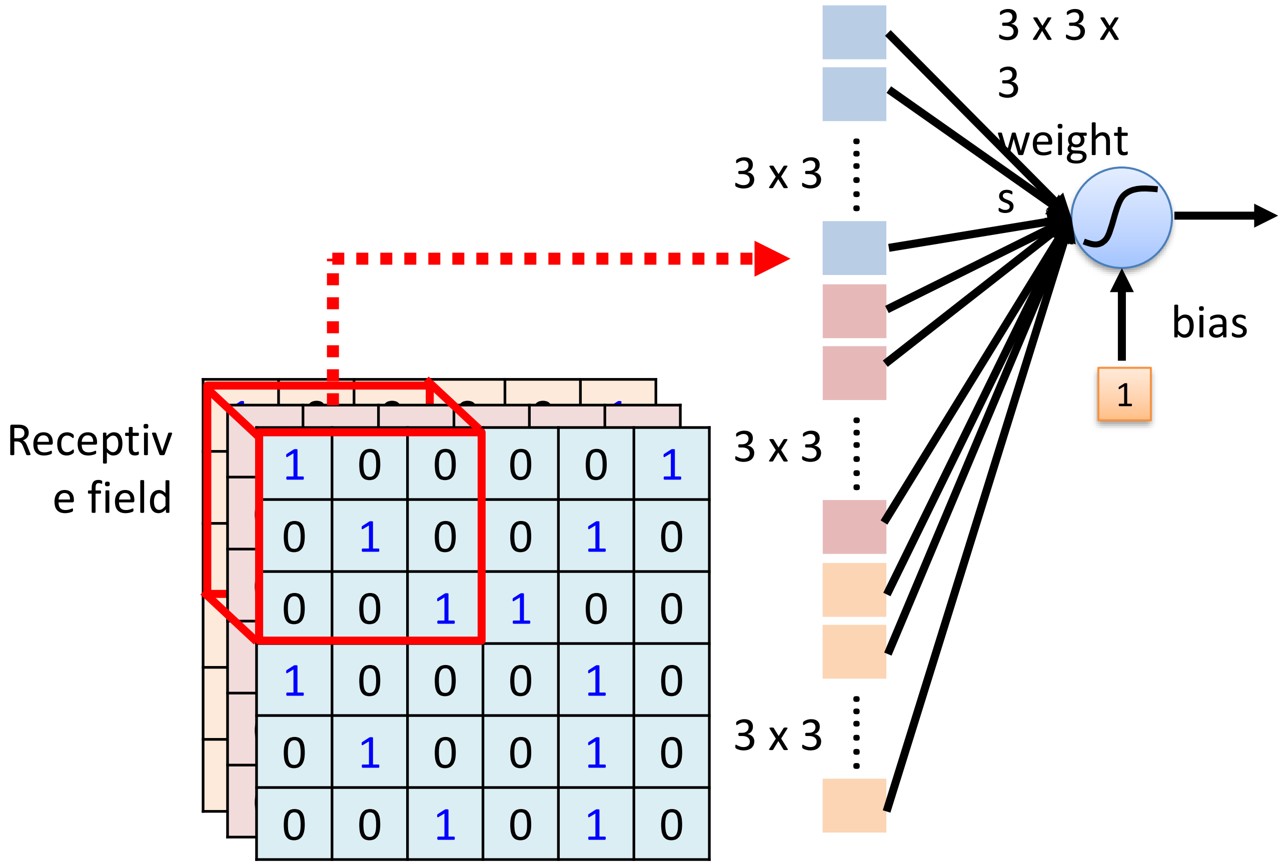
Observation 1



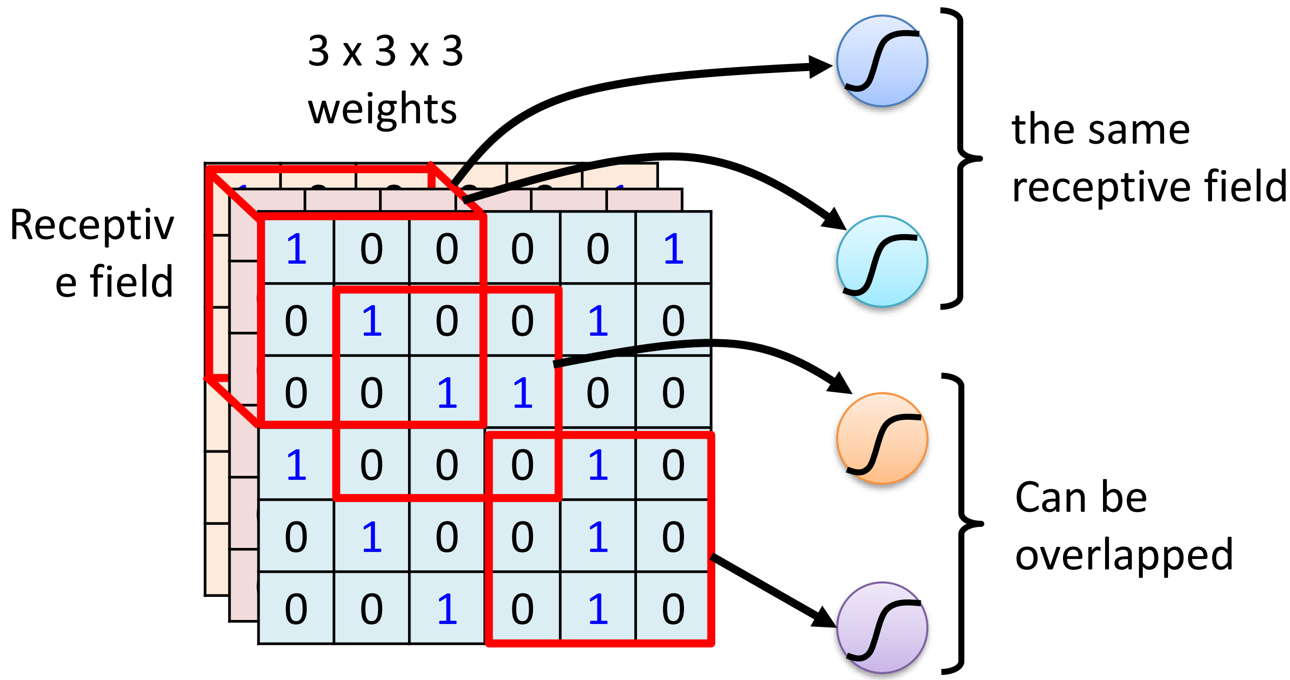
Observation 1



Simplification 1

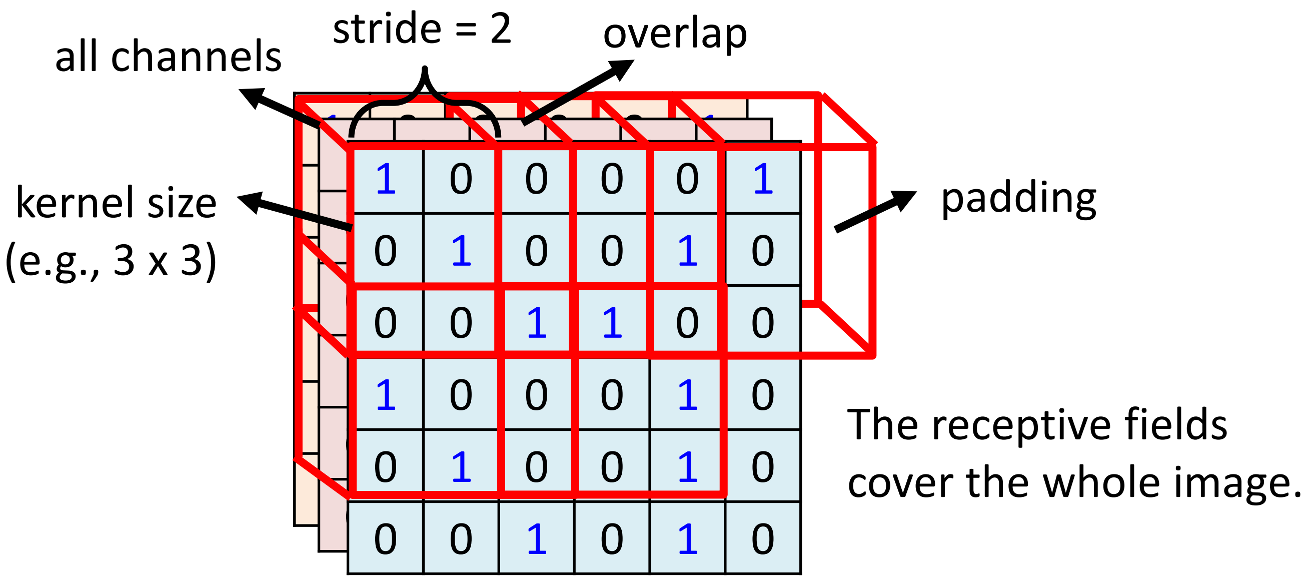


Simplification 1



Simplification 1 – Typical Setting

Each receptive field has a set of neurons (e.g., 64 neurons).



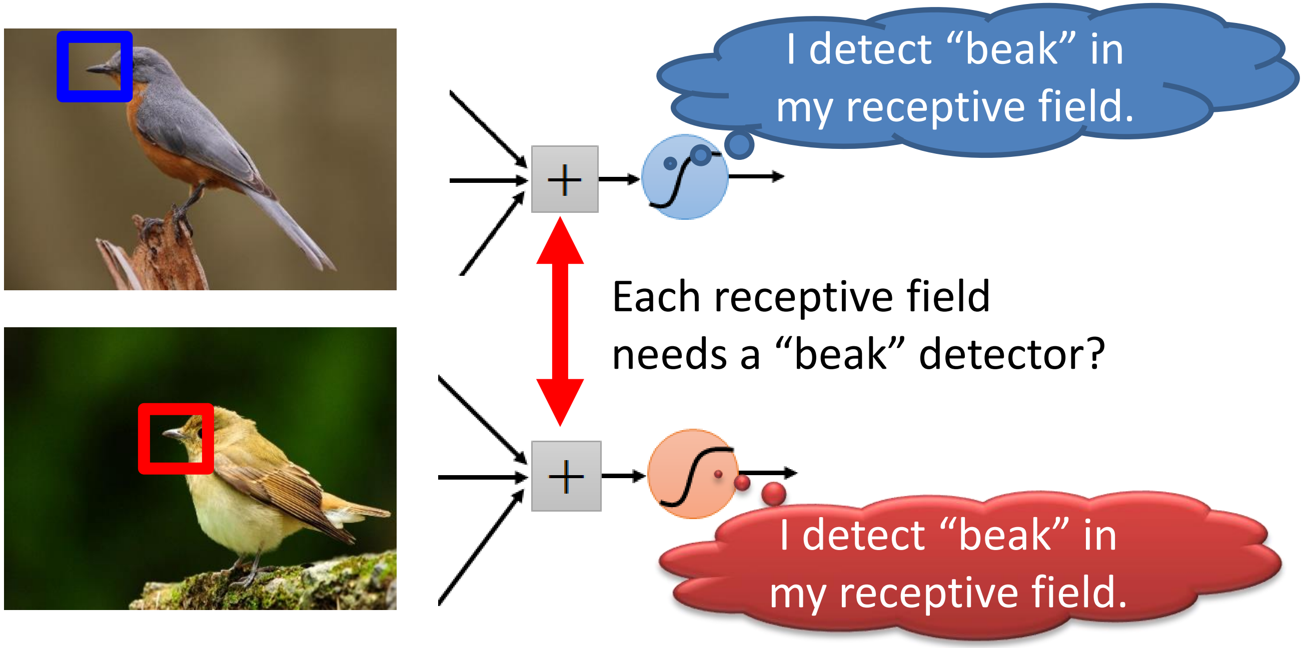
Vision model

How MLP to CNN?

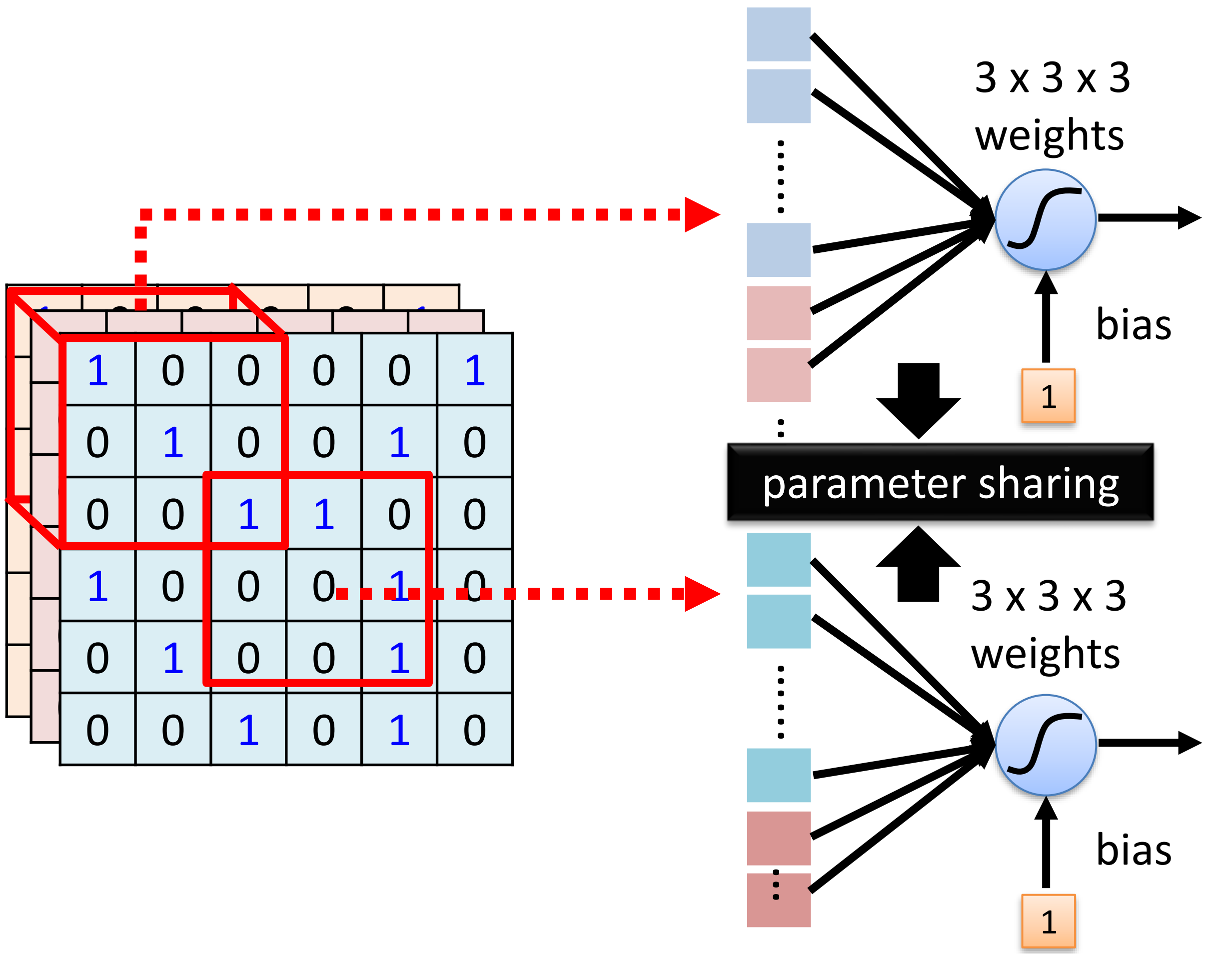
- Model locality
- Parameter Efficiency

Observation 2

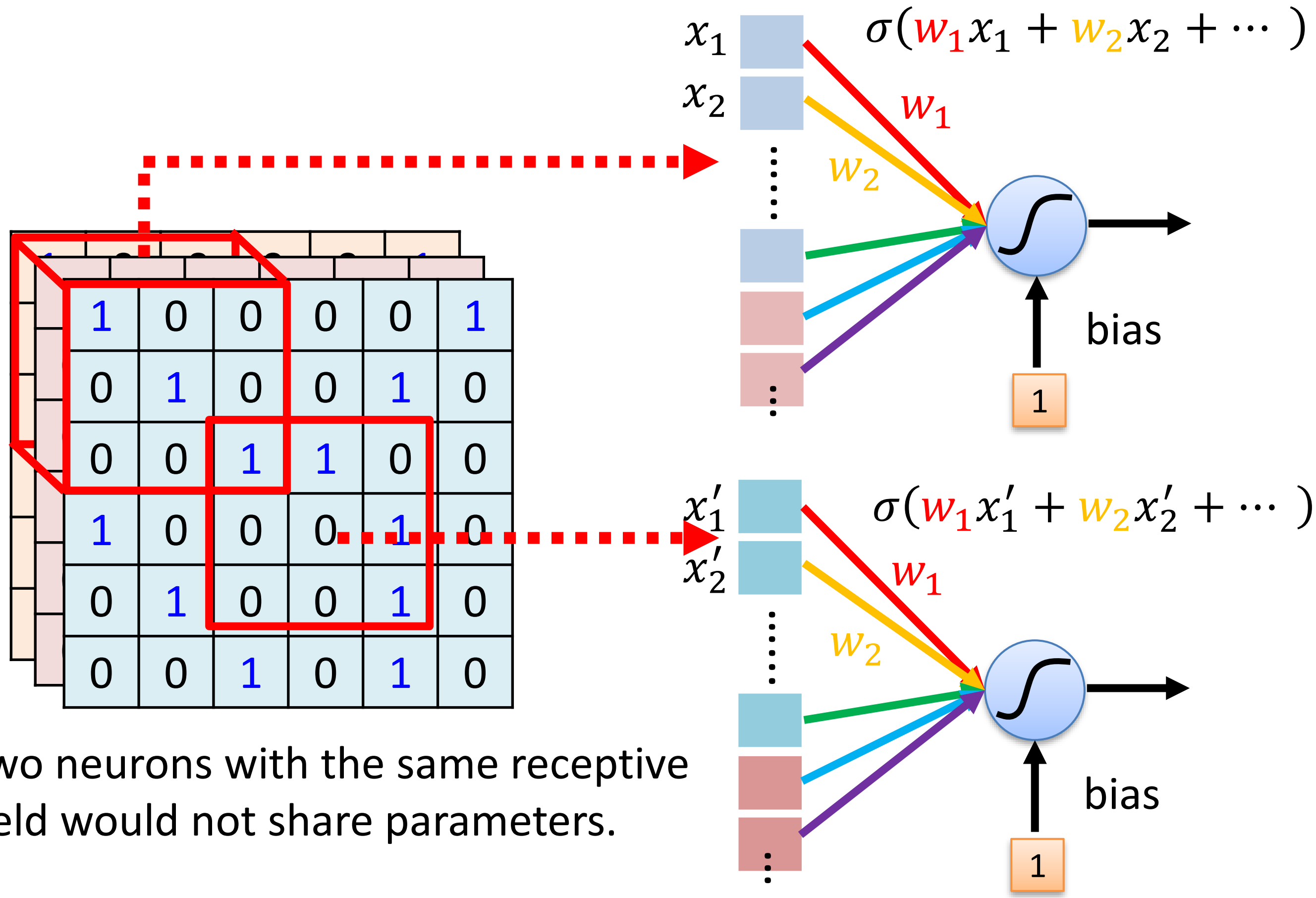
- The same patterns appear in different regions.



Simplification 2



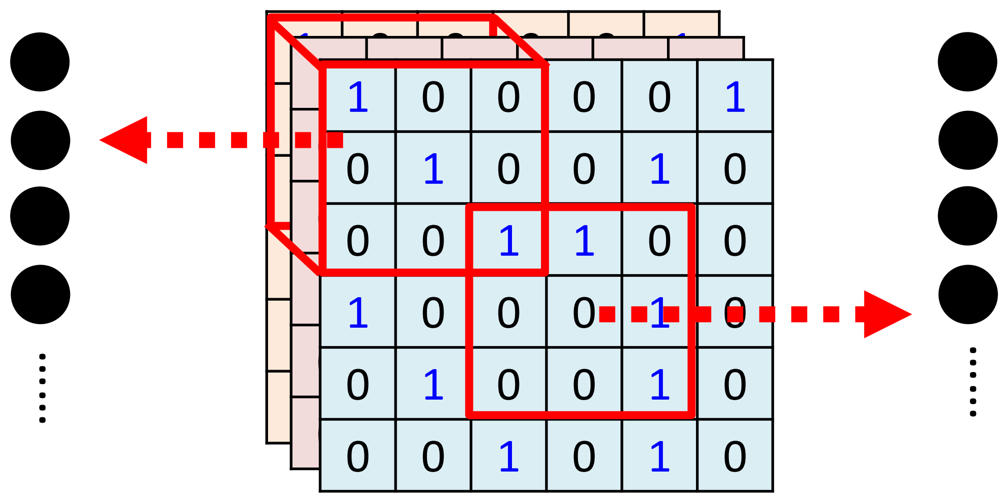
Simplification 2



Two neurons with the same receptive field would not share parameters.

Simplification 2 – Typical Setting

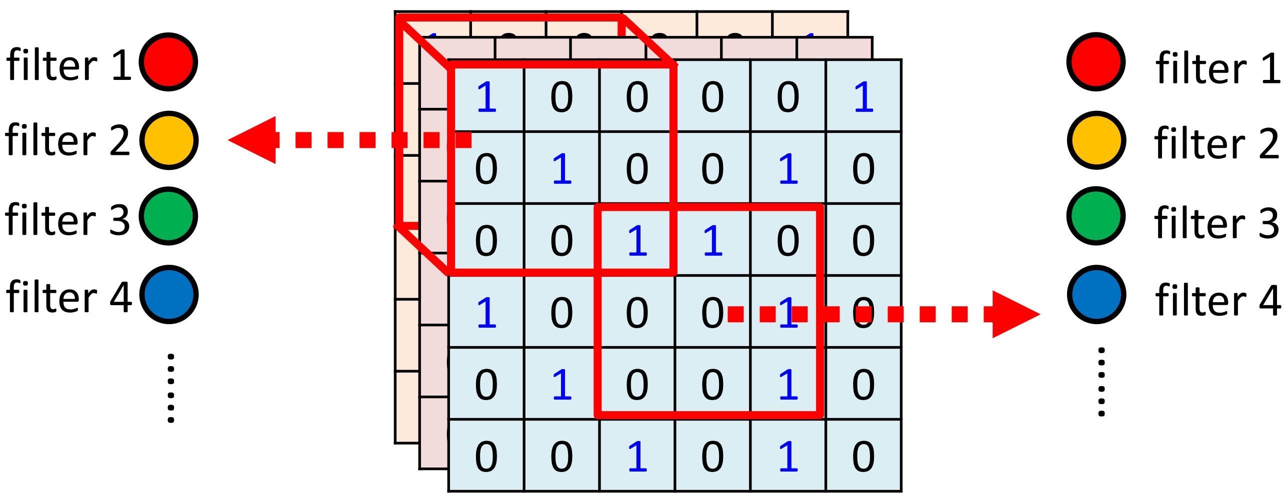
Each receptive field has a set of neurons (e.g., 64 neurons).



Simplification 2 – Typical Setting

Each receptive field has a set of neurons (e.g., 64 neurons).

Each receptive field has the neurons with the same set of parameters.

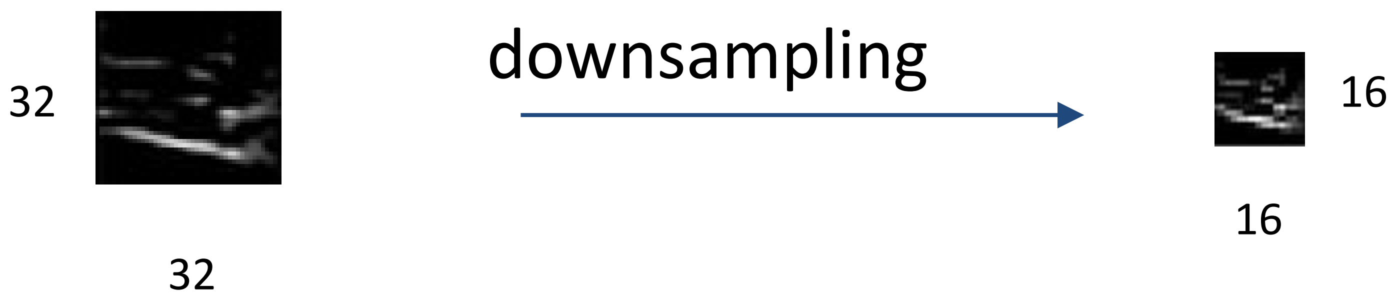


outline

- 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

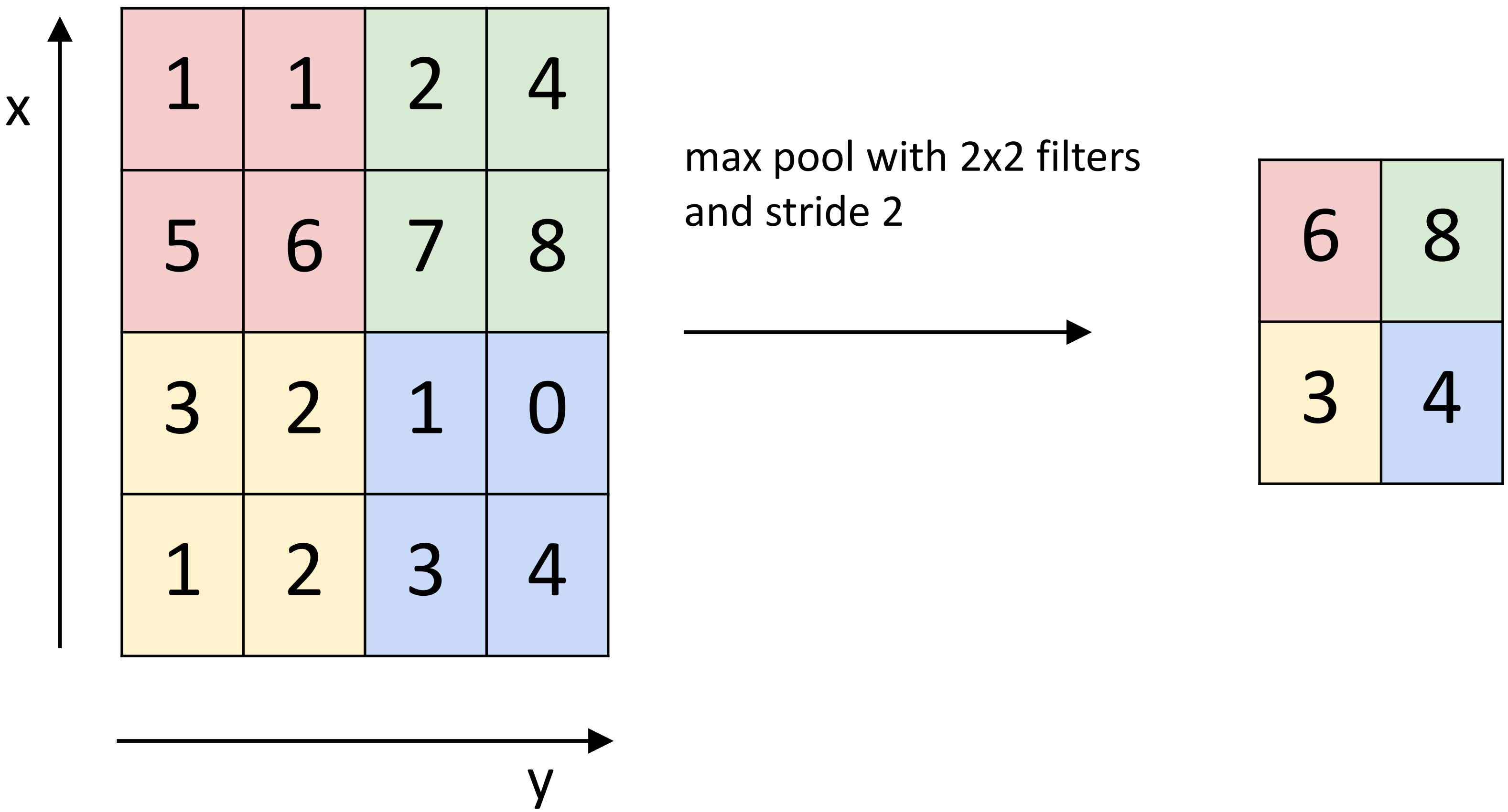
- In ConvNet architectures, **Conv** layers are often followed by **Pooling** layers
  - makes the representations smaller and more manageable without losing too much information.
  - Invariant in region.



Source: Andrej Karpathy & Fei-Fei Li

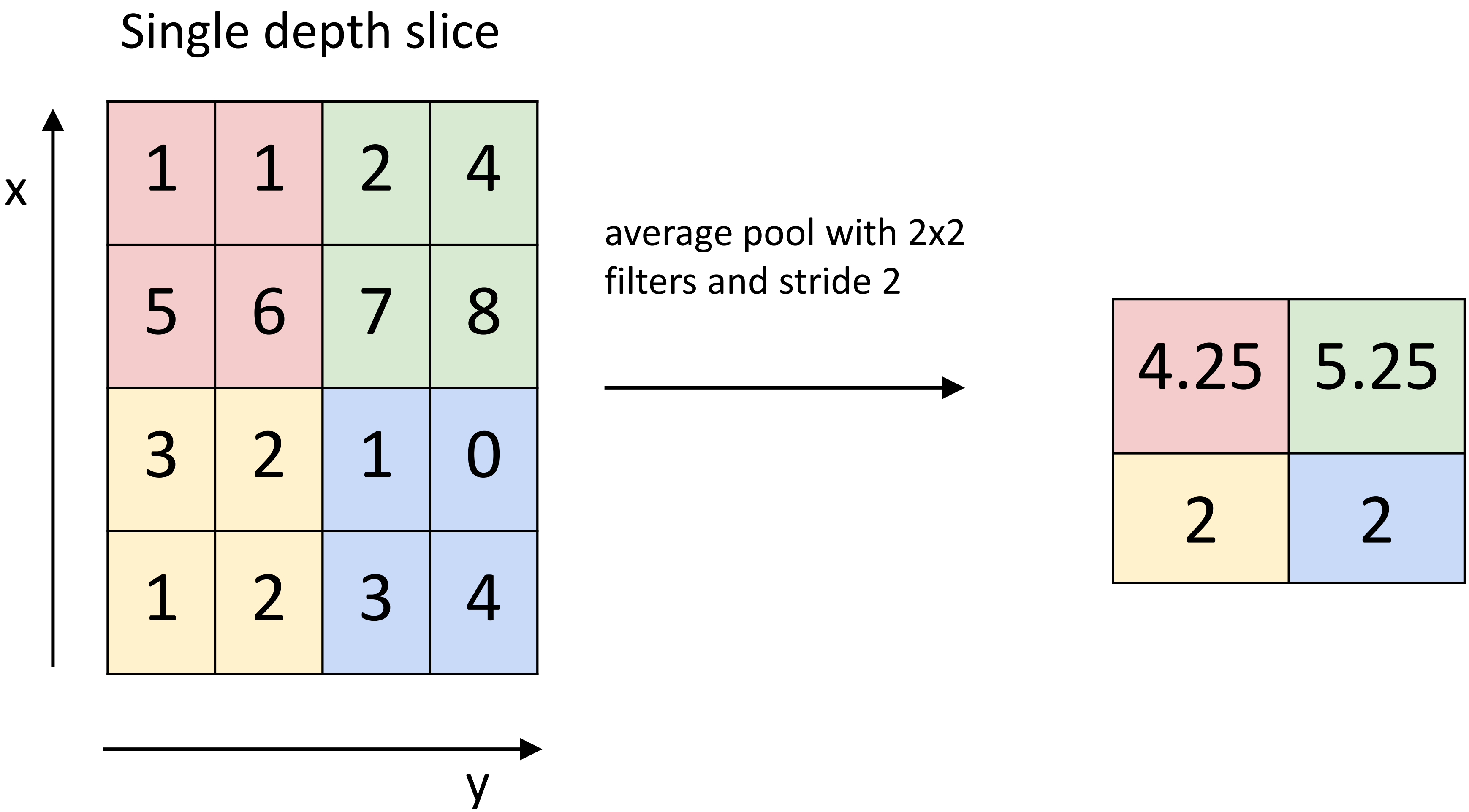
MAX POOLING

Single depth slice



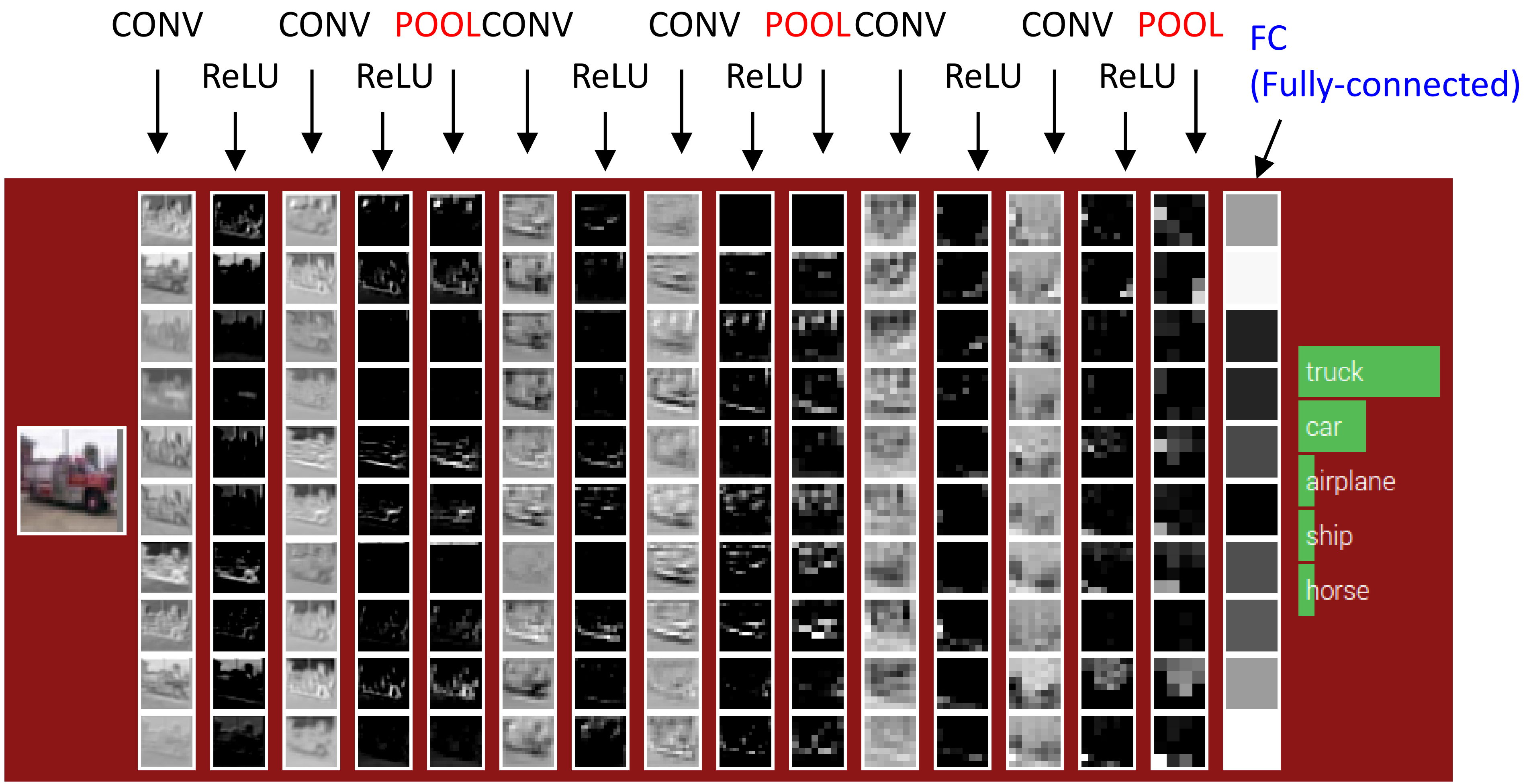
Source: Andrej Karpathy & Fei-Fei Li

Average POOLING



Source: Andrej Karpathy & Fei-

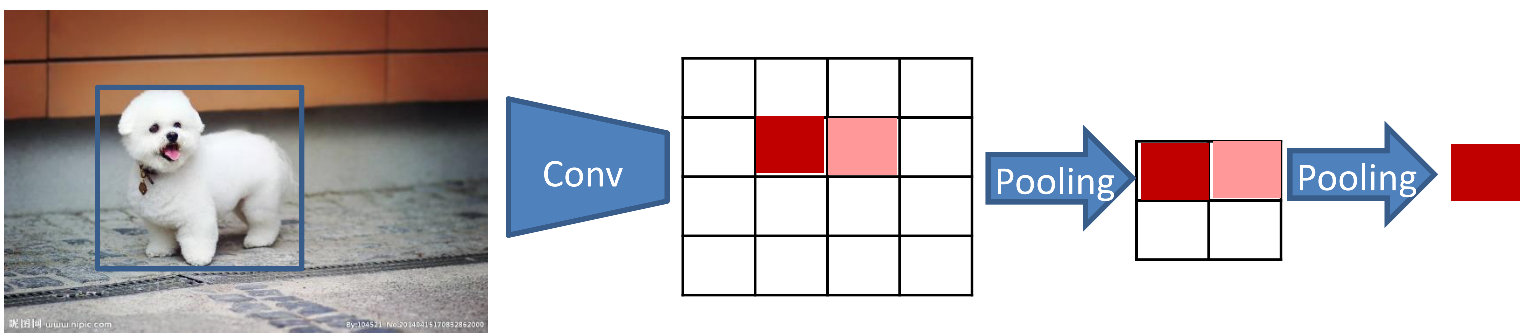
Intuitive example



Source: Andrej Karpathy & Fei-Fei Li

The characteristics of CNN

- Equivalent (等变性)----object detection
- Invariant (不变性)----Image Classification

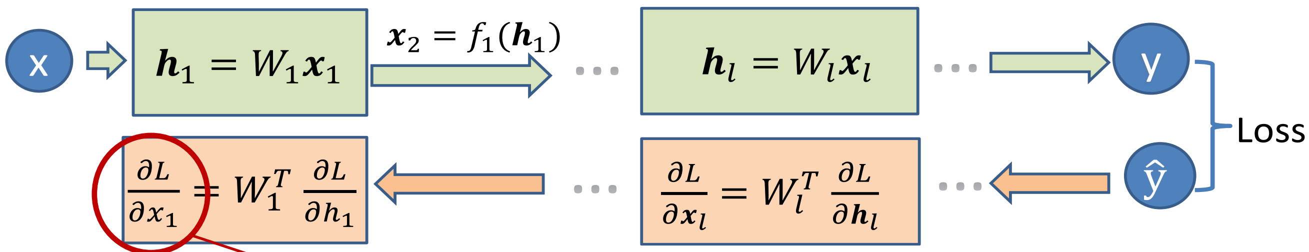


receptive field （感受野）

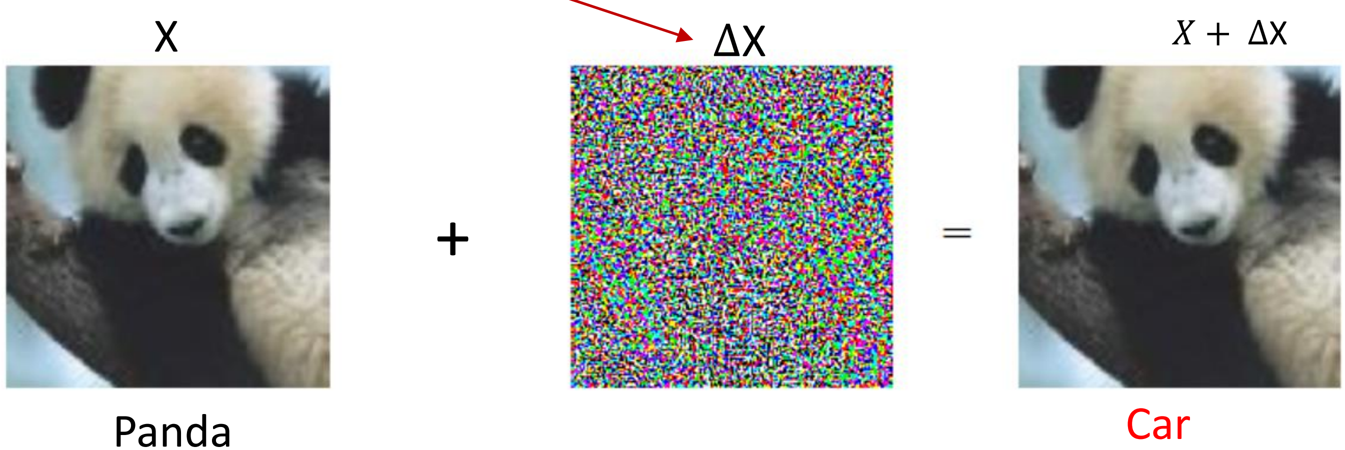
Activation on Feature map

Deep neural networks visualization

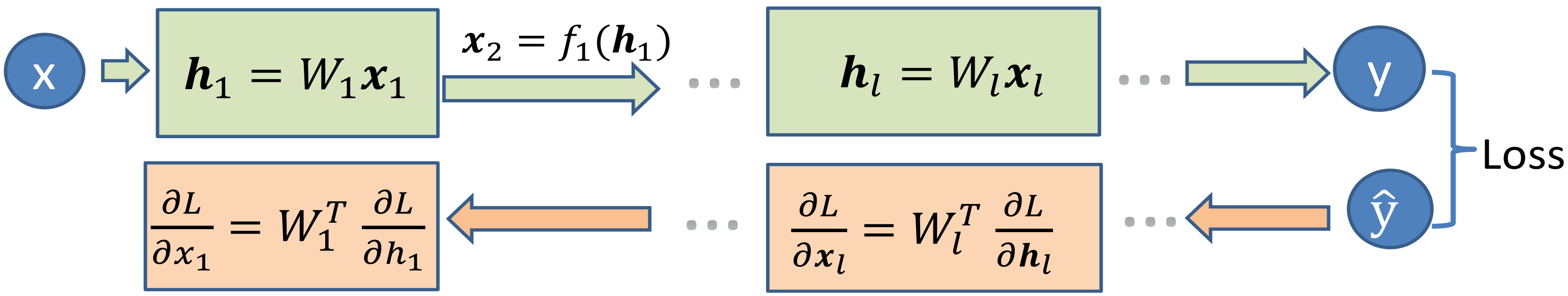
- 对抗样例（Adversarial example ）

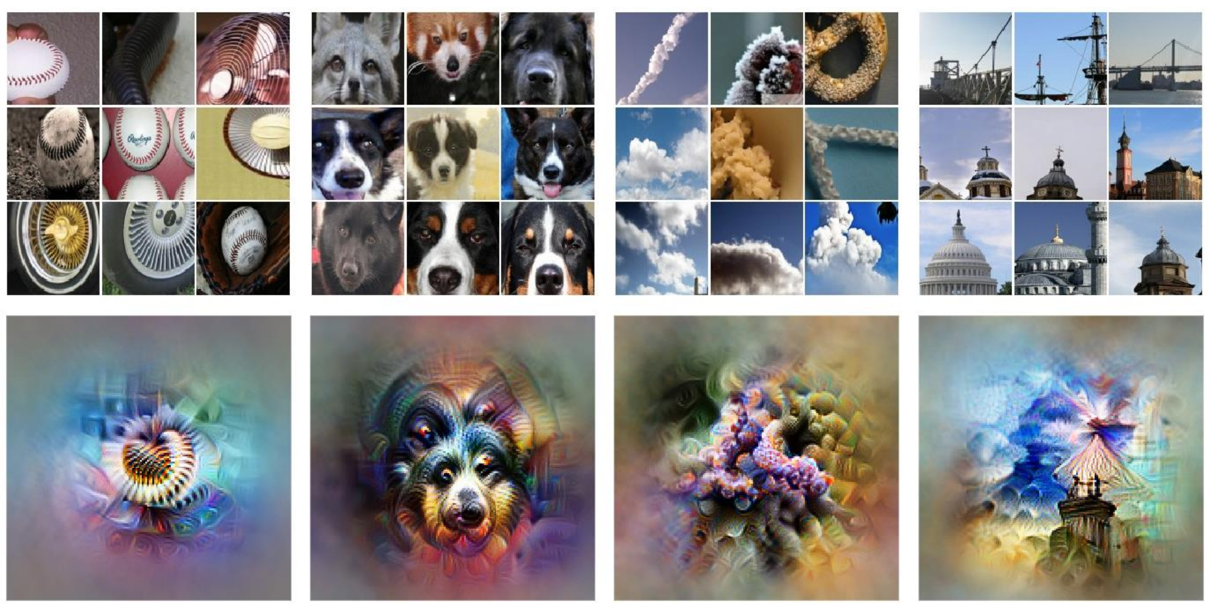


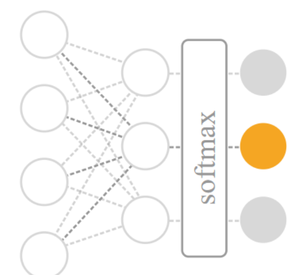
$f(X + \Delta X) \neq f(X)$, ΔX should be imperceptible by human



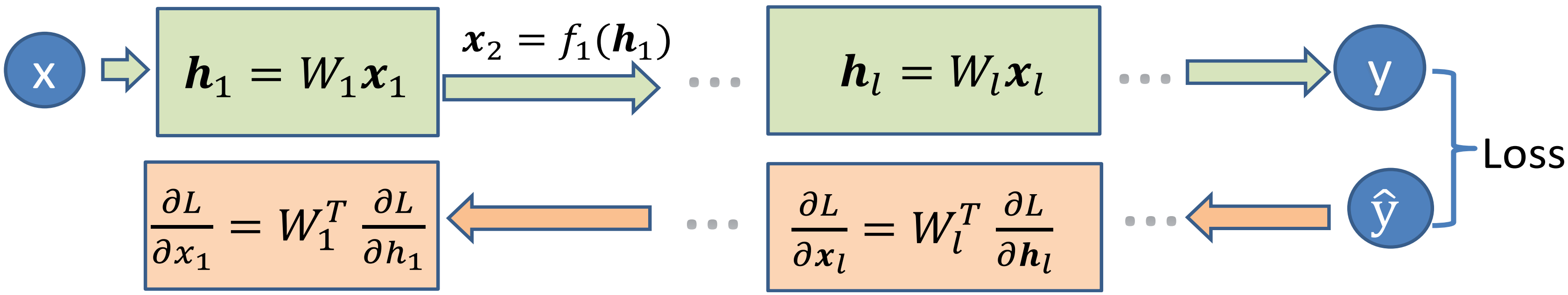
Deep neural networks visualization







Neural Style Transfer

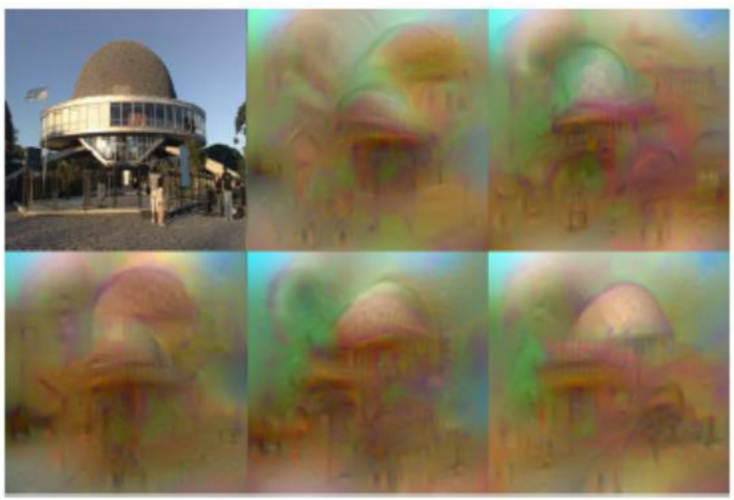


- Style
- Content

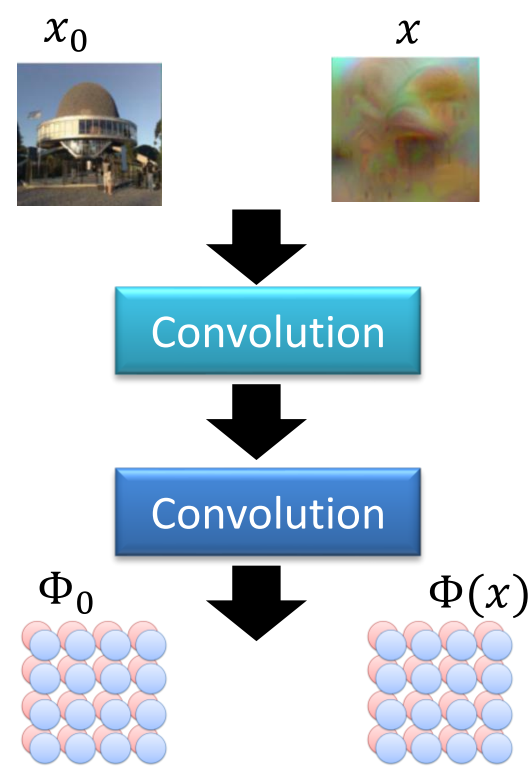
Reconstructing an image from a convolutional layer

- Representation function: $\Phi : \Re^{H \times W \times C} \rightarrow \Re^{d}$ (image space to feature space)
- Target Representation: $\Phi_0 = \Phi(x_0)$ (x0 is the original image)
- We need to find: $x \in \Re^{H \times W \times C}$ by minimizing:

$$x^* = \arg\min_{x \in \Re^{H \times W \times C}} l(\Phi(x), \Phi_0) + \lambda R(x)$$



"Understanding Deep Image Representations by Inverting Them", by Aravindh Mahendran and Andrea Vedaldi.

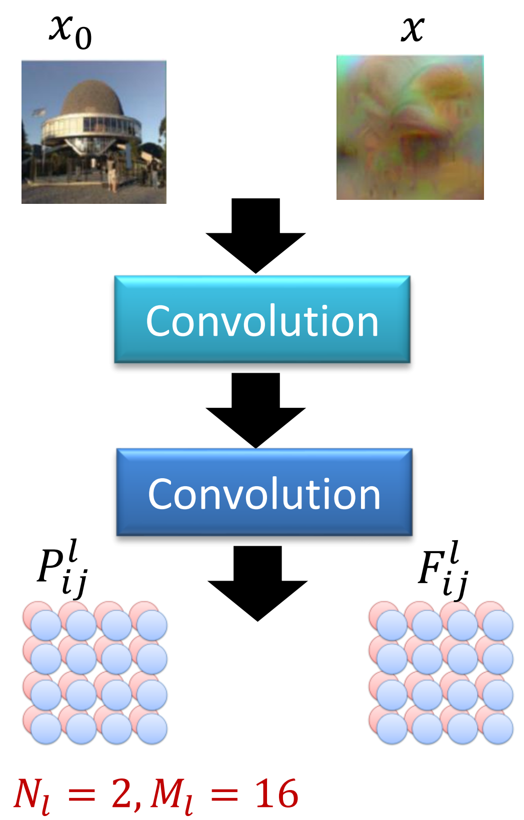


Content Loss Function

- Filters (Depths) at layer l: N_l
- The height times the width of the feature map at layer l: M_l
- Response at layer l: $F_l \in \Re^{N_l \times M_l}$

  F^l_{ij} represents the ith filter at position j in layer l
- Original image: $\vec{p}$
- We generate image: $\vec{x}$ (randomly initialized)
- Squared-error loss:

$$L_{content} = \frac{1}{2} \sum_{i,j} (F^l_{ij} - P^l_{ij})^2$$



Style Loss Function

- Filter correlations are given by the Gram matrix:

$$G^l \in \Re^{N_l \times N_l}$$

- $G^l{}_{ij}$ is the inner product between the filters i and j in layer l:

$$G^l{}_{ij} = \sum_k F^l{}_{ik} F^l{}_{jk}$$

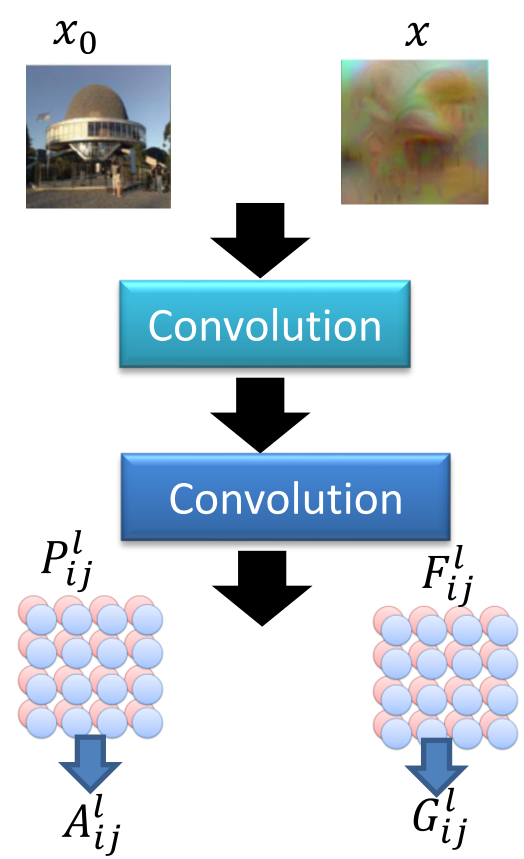
- The loss at layer l:

$$E_l = \frac{1}{4N^2{}_l M^2{}_l} \sum_{i,j} (G^l{}_{ij} - A^l{}_{ij})^2$$

A <-> original image
G <-> generated image

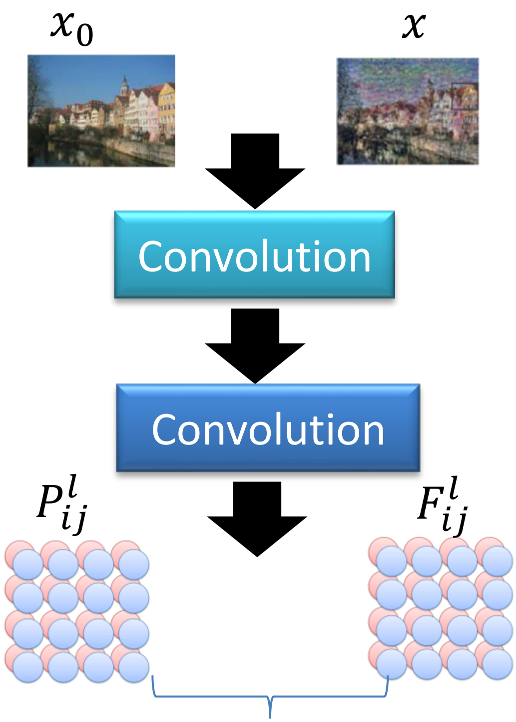
- The total style loss:

$$L_{style} = \sum_{l=0}^{L} w_l E_l$$



Content Reconstruction

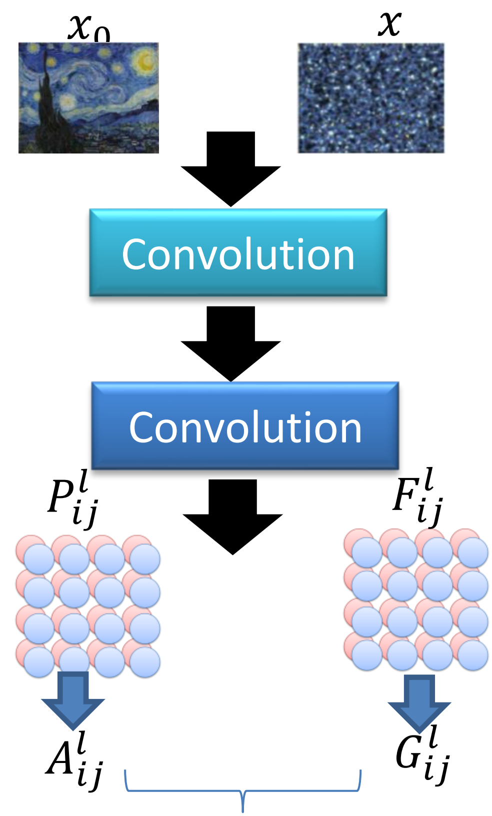
Image reconstructed from layers
(a)'conv1_1',
(b)'conv2_1',
(c)'conv3_1',
(d)'conv4_1' and
(e)'conv5_1'
of the original VGG-Network



$$L_{content} = \frac{1}{2}\sum_{i,j}(F^{l}_{ij} - P^{l}_{ij})^2$$

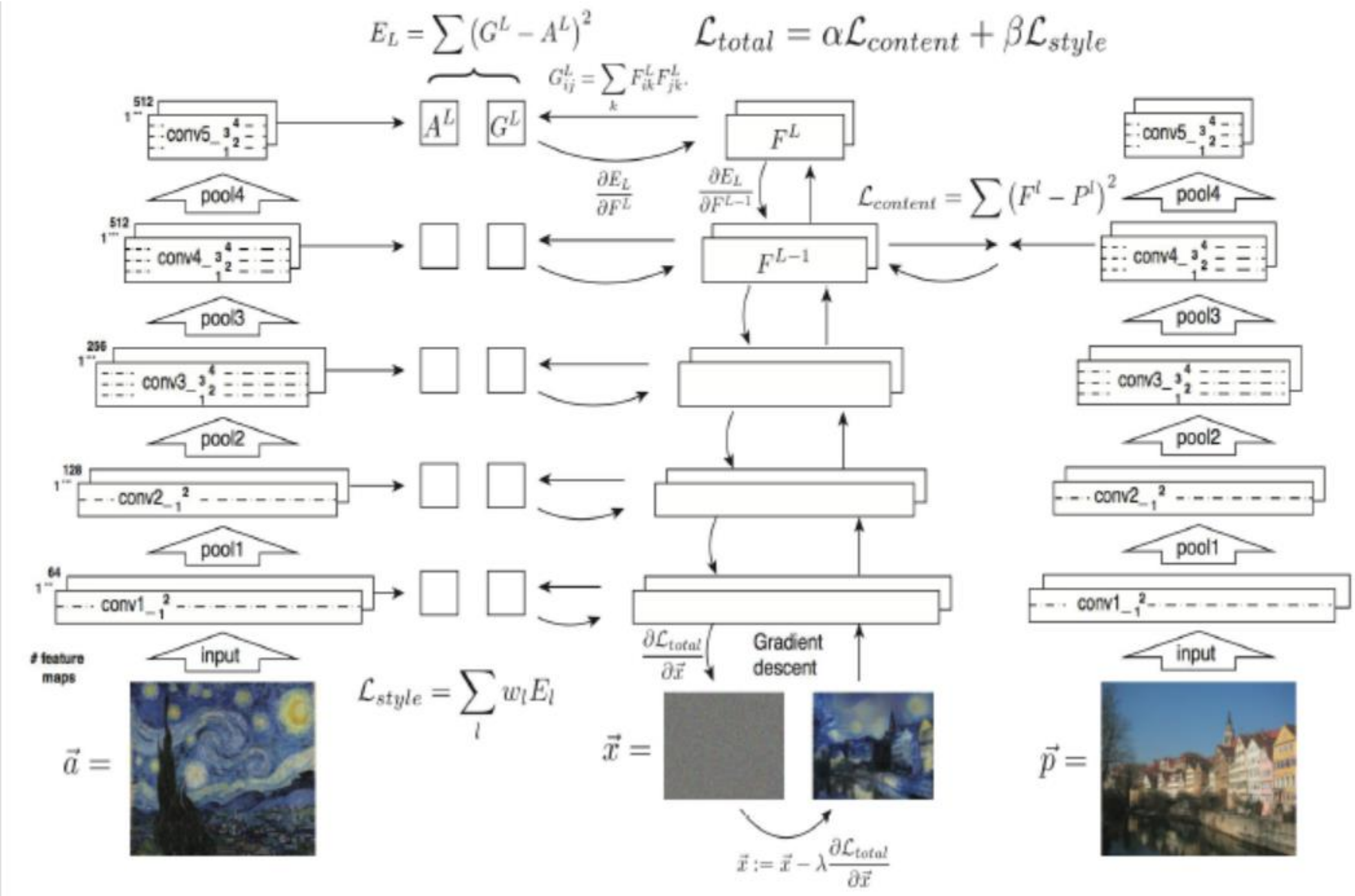
Style Reconstruction

Style representations (filer correlations) from:
(a) 'conv1_1',
(b) 'conv1_1', 'conv2_1',
(c) 'conv1_1', 'conv2_1', 'conv3_1',
(d) 'conv1_1', 'conv2_1', 'conv3_1', 'conv4_1',
(e) 'conv1_1', 'conv2_1', 'conv3_1', 'conv4_1', 'conv5_1'.

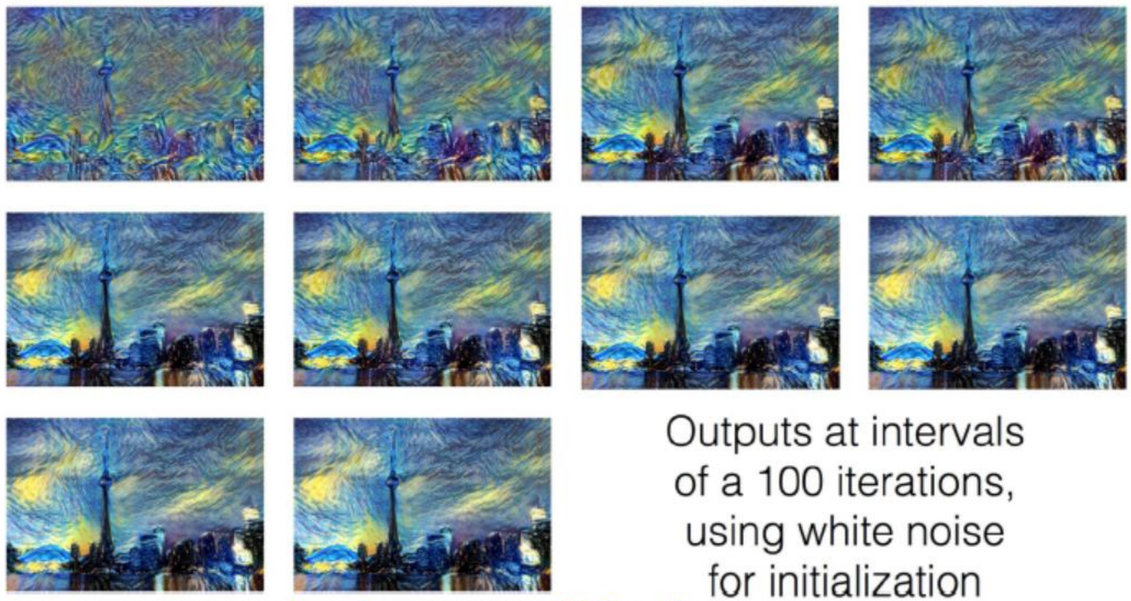


$$E_l = \frac{1}{4N^2{}_l M^2{}_l} \sum_{i,j} (G^l{}_{ij} - A^l{}_{ij})^2$$

The Total Loss Function

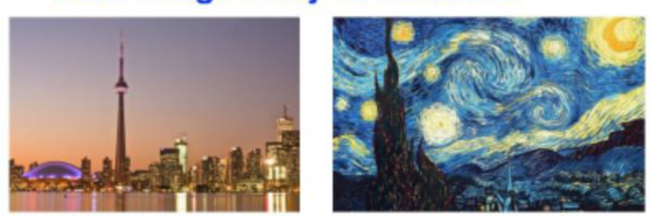


Results



Outputs at intervals
of a 100 iterations,
using white noise
for initialization

show image every 10 iterations



谢谢！