The Introduction To Artificial Intelligence

Yuni Zeng yunizeng@zstu.edu.cn

2024-2025-1

The Introduction to Artificial Intelligence

- Part I Brief Introduction to AI & Different AI tribes
- Part II Knowledge Representation & Reasoning
- Part III AI GAMES and Searching
- Part IV Model Evaluation and Selection
- Part V Machine Learning
- Part VI Neural Networks

Neural Networks

- *Brief review*
- Convolutional Neural Network

  Classical Architecture

  Gradient-Based Learning Algorithm

  Achievements and Applications

Neural Networks

- Convolutional Neural Network

  Classical Architecture

  Gradient-Based Learning Algorithm

  Achievements and Applications

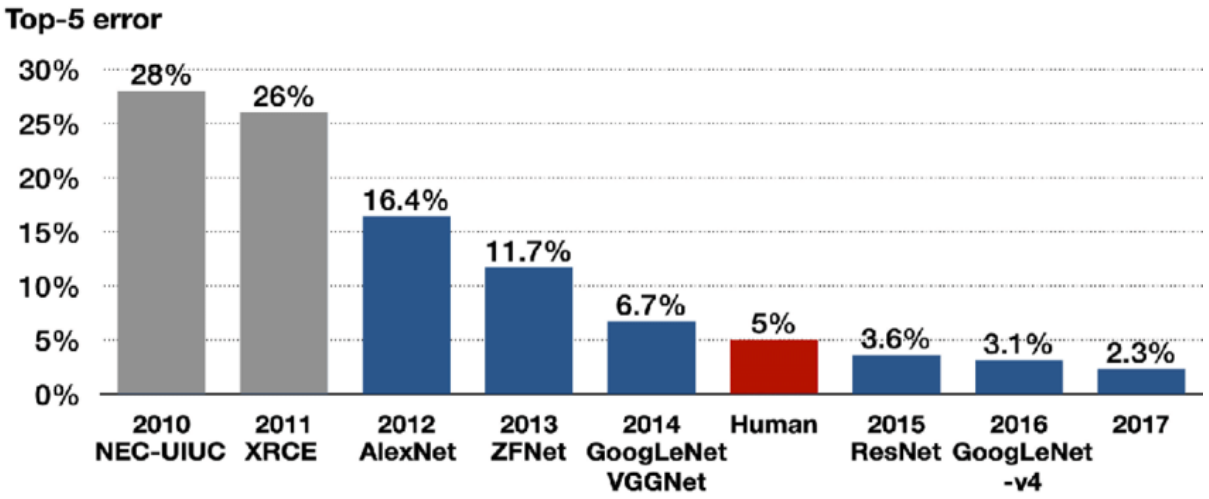
Convolutional Neural Network

- Introduction

ImageNet & Large Scale Visual Recognition Challenge

ImageNet:
14 million images;
20,000 categories

ILSVRC
1,281,167 training images;
50,000 validation images;
100,000 test images;
1000 object classes



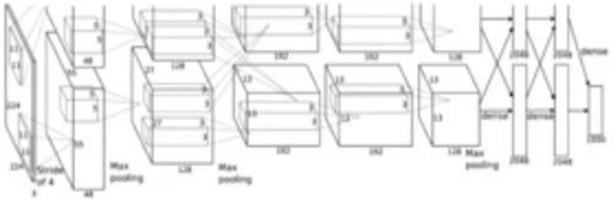
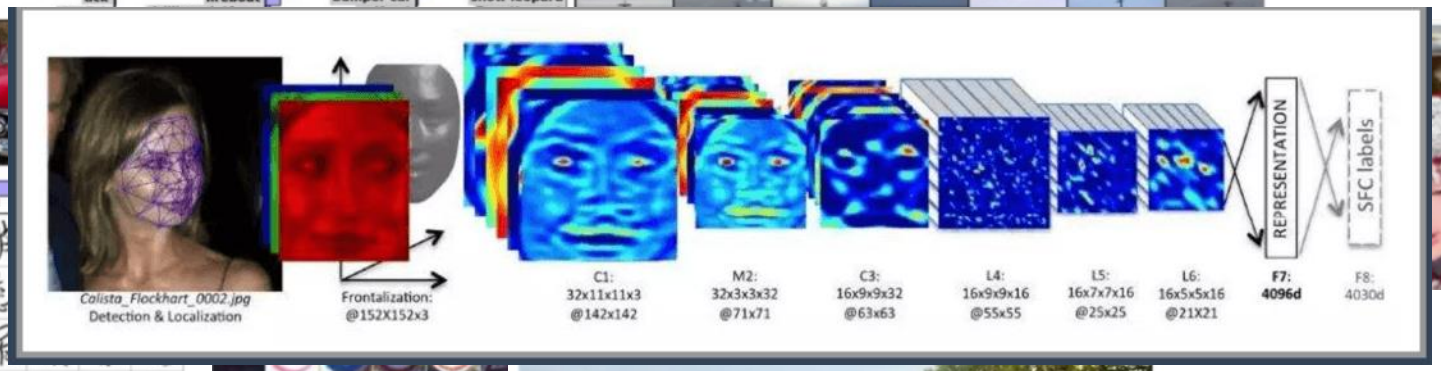
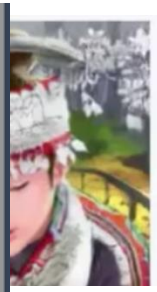
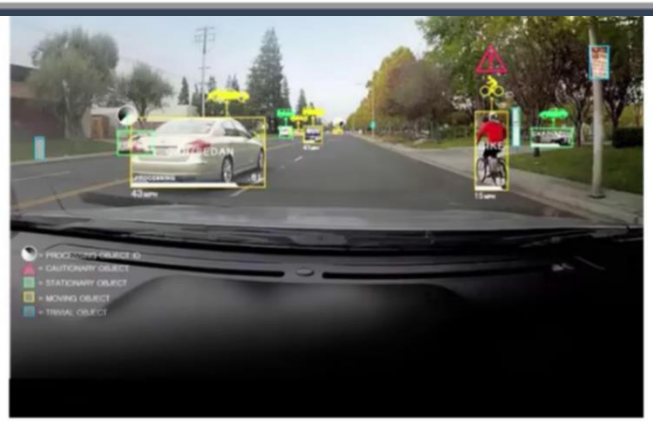


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

Convolutional Neural Network







NVIDIA Tegra X1

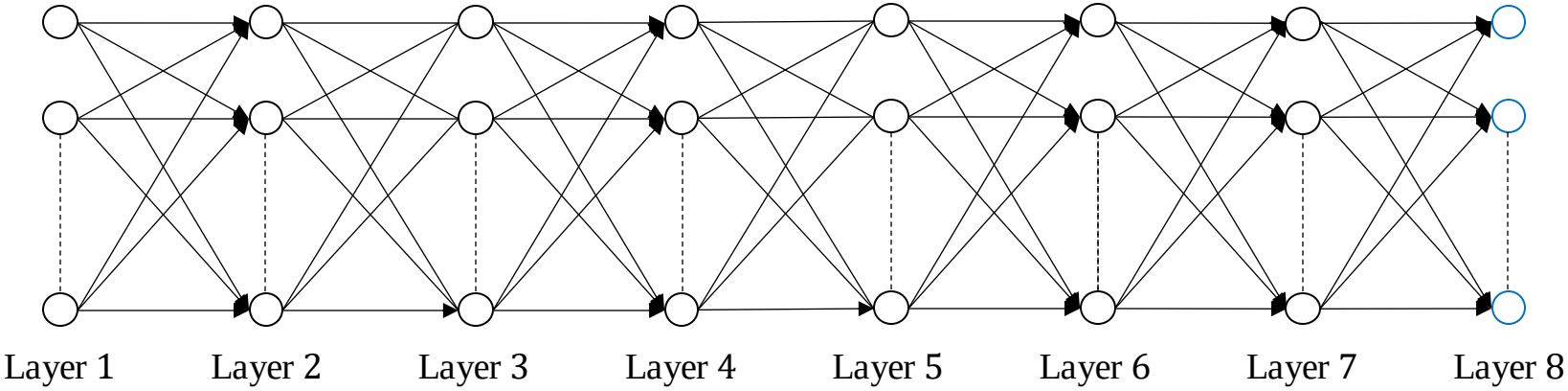
Convolutional Neural Network

Introduction

How to handle the very large colorful image in the size of 224x224x3, where the 3 denotes the R,G,B channels.

28 × 28

224x224x3



Convolutional Neural Network

Introduction

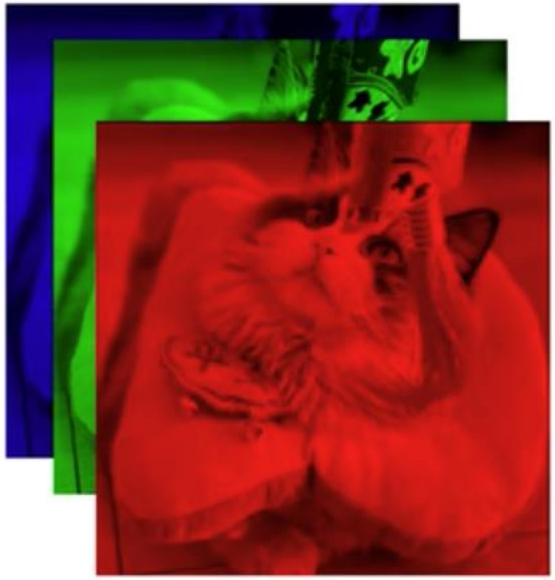
- Suppose the dimension of hidden layer is 100. The number of parameters in the first layer is 15,052,800, unacceptable !
- How does the brain process the image?

Convolutional Neural Network

Introduction

How does a computer understand a picture?

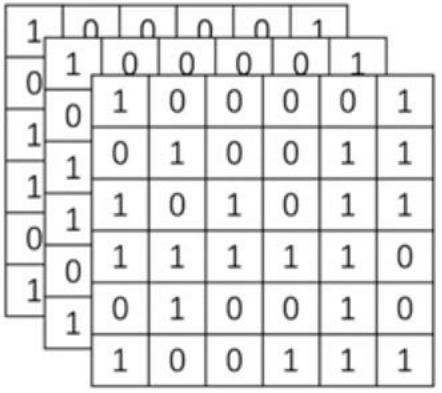
512*512 pixels



RGB three channels

```
In [6]: lena.shape
Out[6]: (512, 512, 3)

In [14]: lena
Out[14]: array([[[125, 137, 226],
                 [125, 137, 226],
                 [133, 137, 223],
                 ...,
                 [122, 148, 230],
                 [110, 130, 221],
                 [ 90,  99, 200]],
```

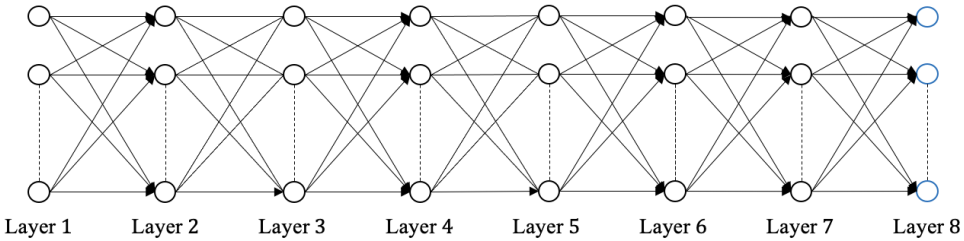


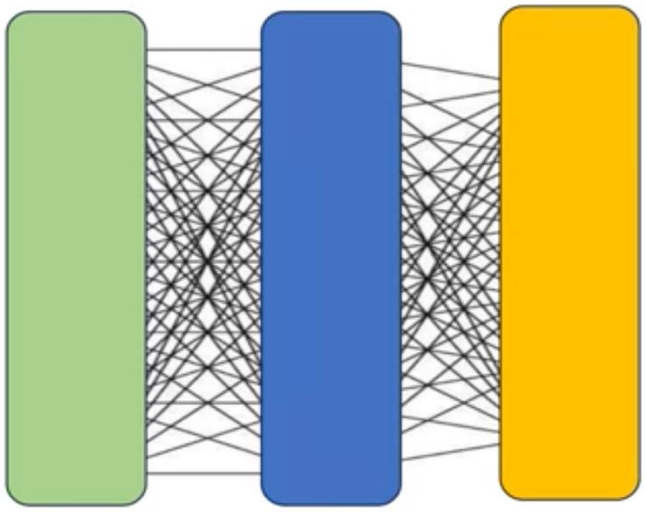
512*512*3

张量Tensor

Convolutional Neural Network

- Introduction

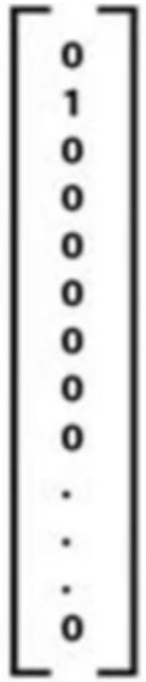




边数（要训练的权重）是多少?

151528000

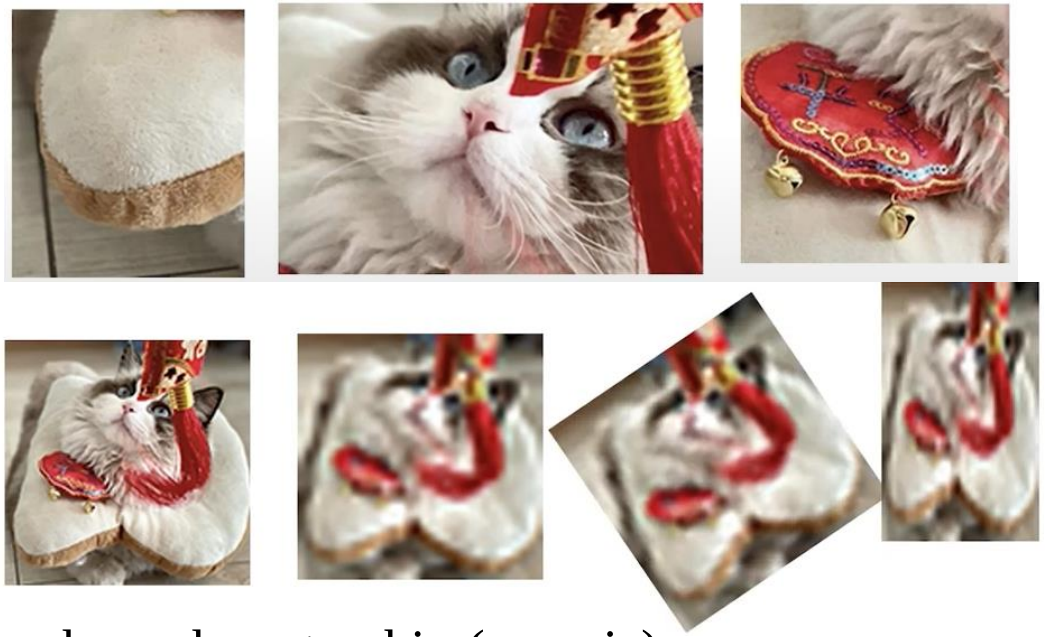
151M参数



The amount of data that needs to be processed is too large!

Convolutional Neural Network

Introduction



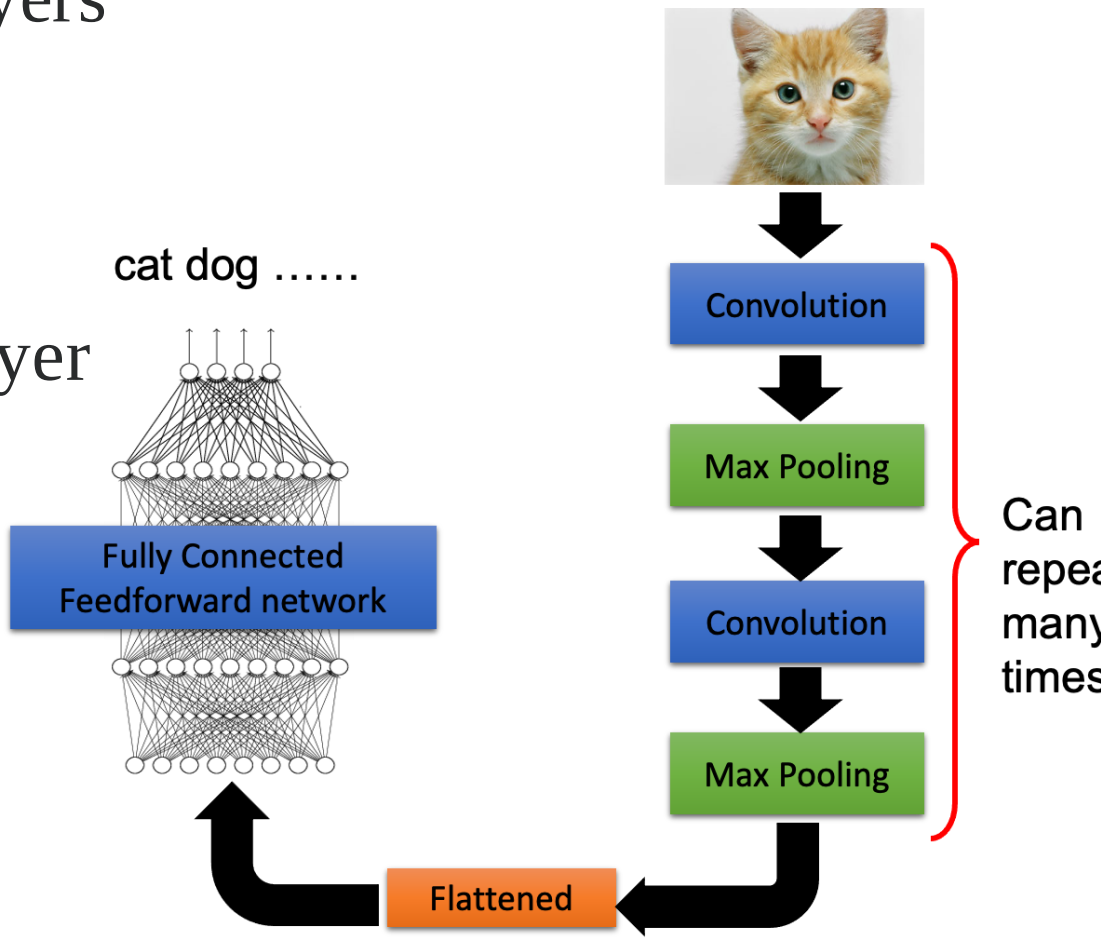
- Motivation: neural networks perhaps too big (generic)
  - Full connectivity at each step might be too much
  - Disconnecting some connections performance is not impaired or even better
- Intuition: how people extract content from an image
- Features we care about are often only a small part of the image
  - Lots of redundant information
- Presence may exist in the form of panning, zooming, rotating
- Zooming of the image (within reason) does not affect our judgment

Convolutional Neural Network

Introduction

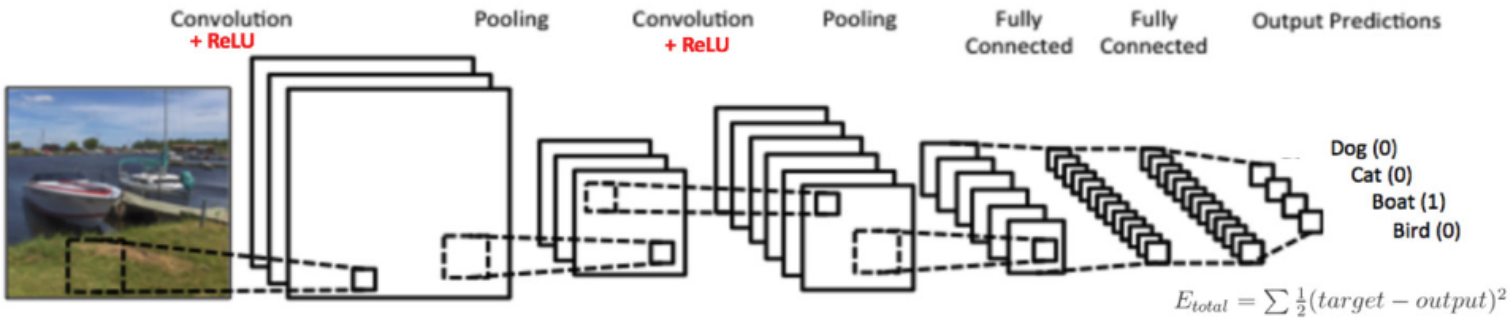
A typical CNN has 4 layers

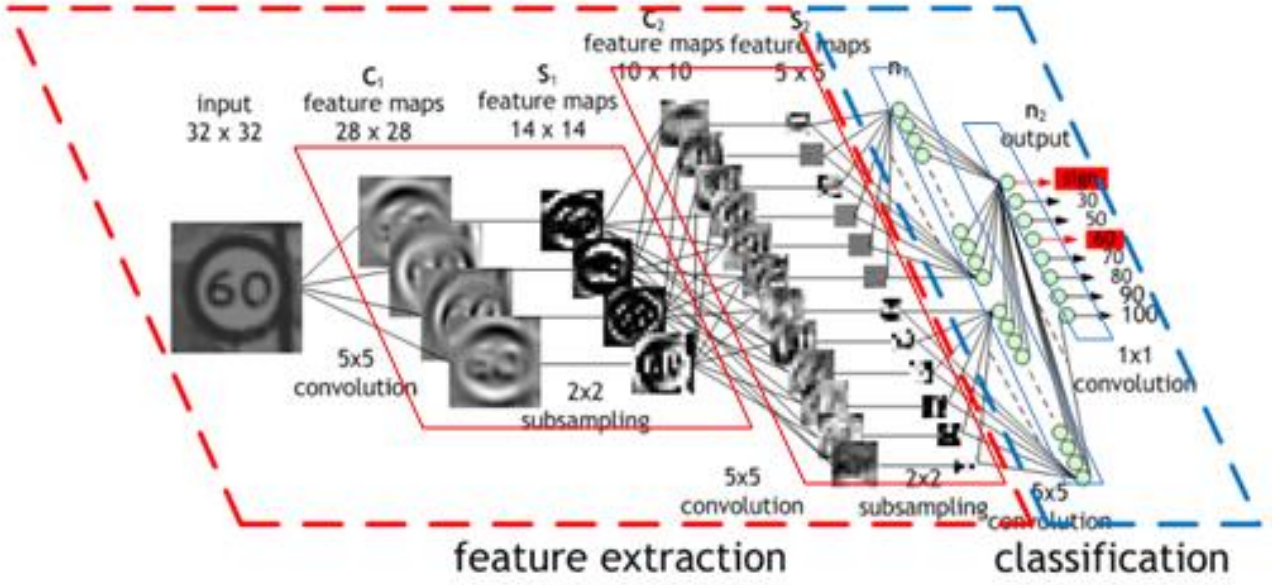
- Input layer
- Convolution layer
- Pooling layer
- Fully connected layer



Convolutional Neural Network

Introduction





Convolutional Neural Network

- Classical Architecture
  - Three Main Concepts in CNNs
    - Receptive Field
    - Convolution (Simple Cell)
    - Pooling (Complex Cell)

Convolutional Neural Network

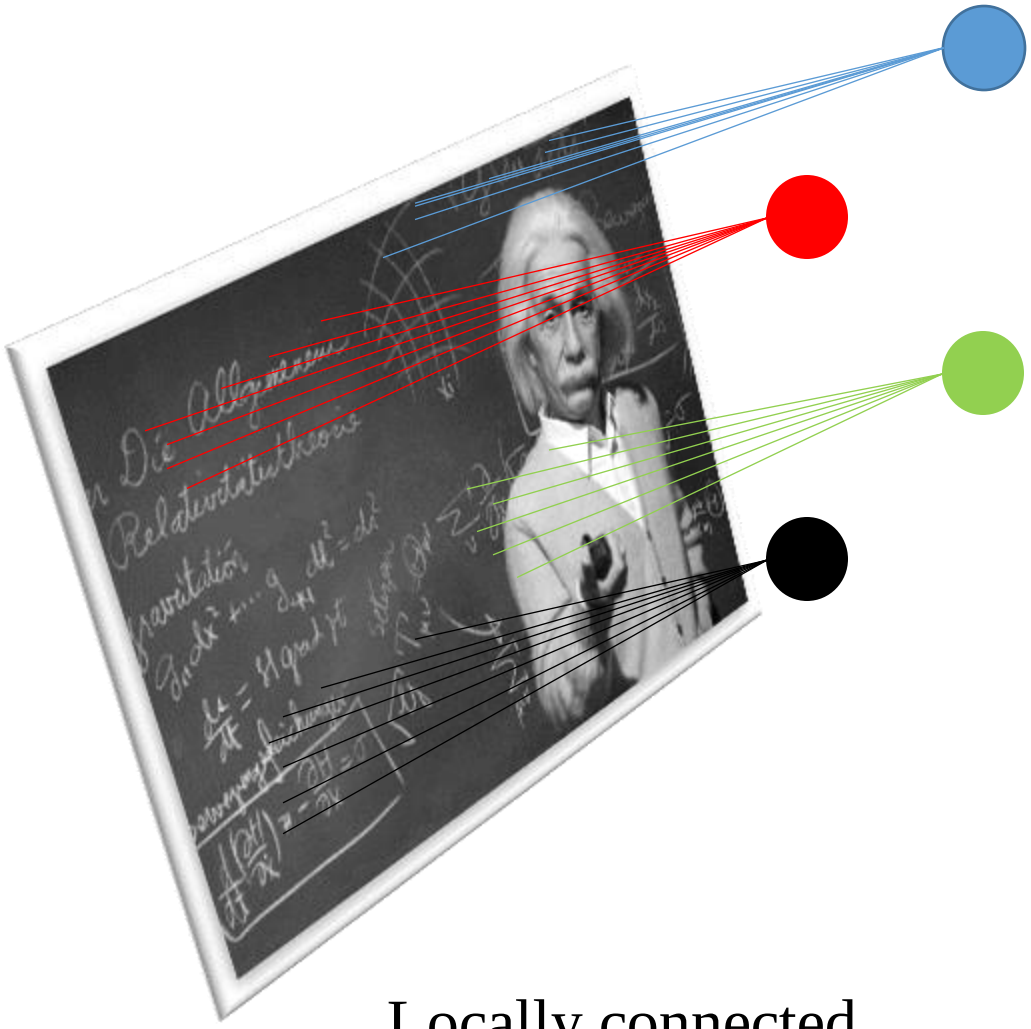
- Classical Architecture of CNN

64 * 64 image

Convolutional Neural Network

- Classical Architecture
  - Receptive Field

Fully connected



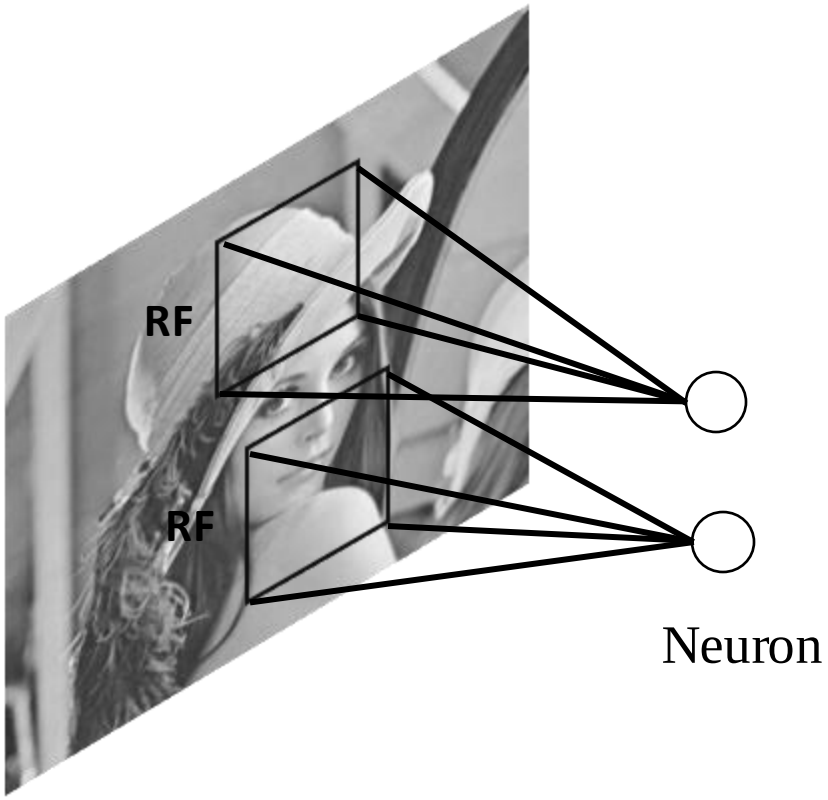
Locally connected

Convolutional Neural Network

Classical Architecture of CNN

➢ Receptive Field

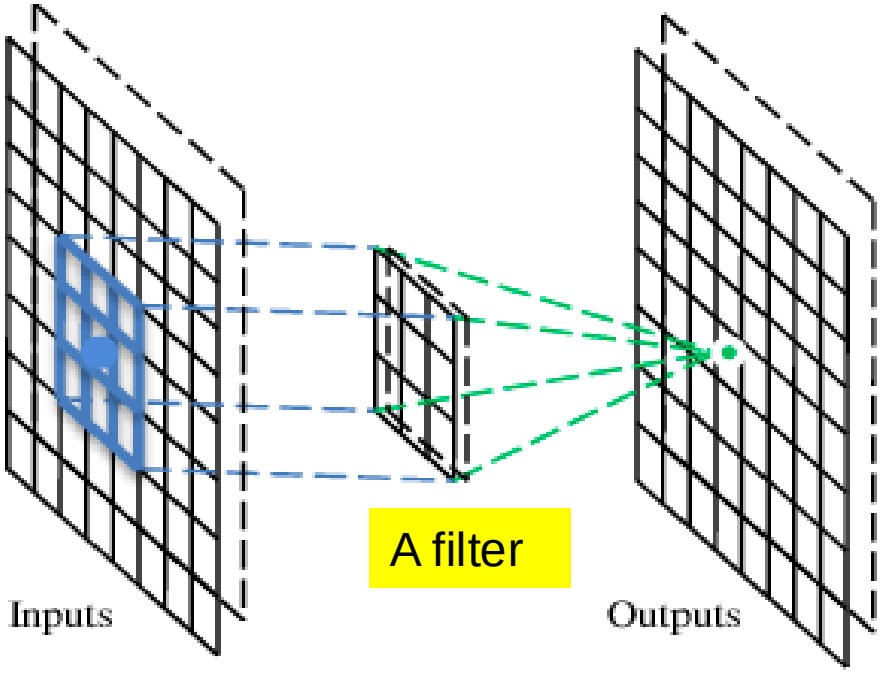
- **Receptive Field**
  - Receptive Field of a neuron is a ***continuous sub-region*** of the input space
- **Locally Connected**
  - Neuron is just connected to its own Receptive Field



Convolutional Neural Network

Classical Architecture of CNN

A CNN is a neural network with some convolutional layers (and some other layers). A convolutional layer has a number of filters that does convolutional operation.

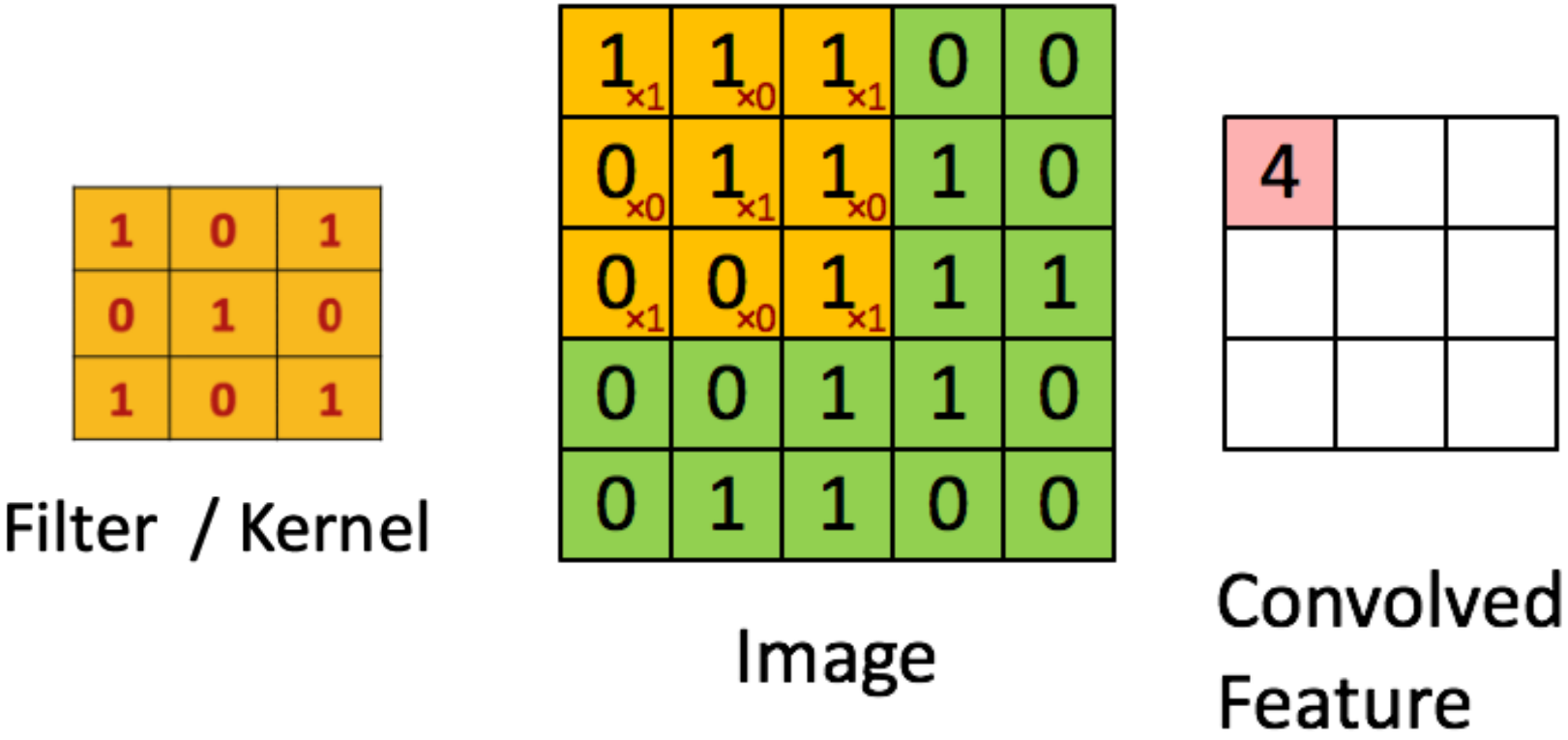


Convolutional Neural Network

- Classical Architecture
  - Three Main Concepts in CNNs
    - Receptive Field
    - Convolution (Simple Cell)
    - Pooling (Complex Cell)

Convolutional Neural Network

- Classical Architecture of CNN



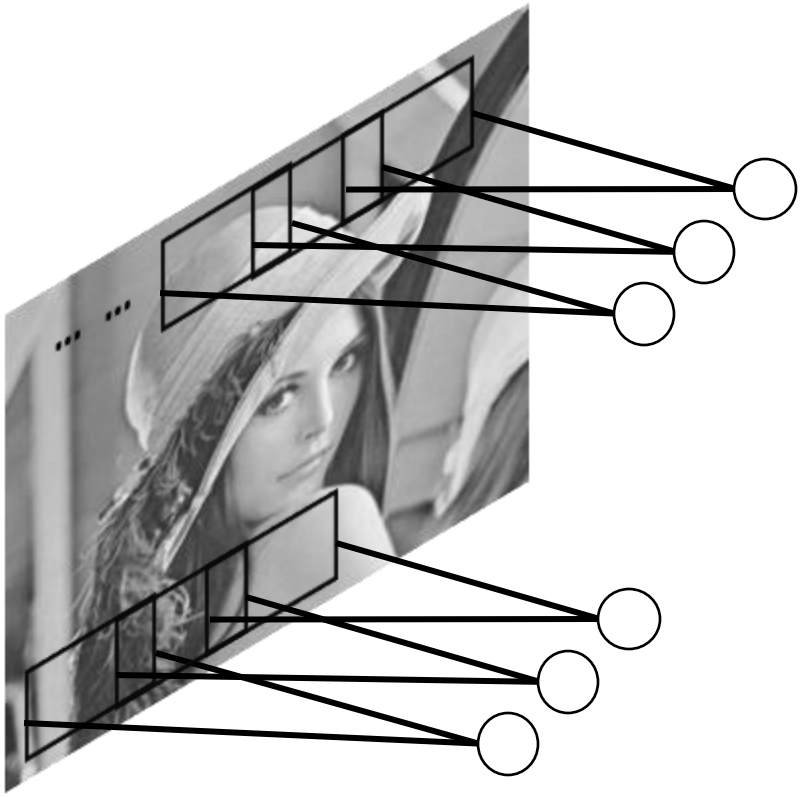
The sum of the dot product of the image pixel value and kernel pixel value gives the output matrix

Convolutional Neural Network

Classical Architecture

➢ Convolution (Simple Cell)

- **Shared Weight**
  - All units share the same set of weights
- **Why Shared Weight**
  - Features that are useful on one part of the image and probably useful elsewhere
  - Shared Weight can reduce the number of parameters



Convolutional Neural Network

- Classical Architecture - Convolution

0

1	0	1
1	0	1
1	0	1

1

1	1	1
0	0	0
1	1	1

Convolutional Neural Network

- Classical Architecture - Convolution

0

1	0	1
1	0	1
1	0	1

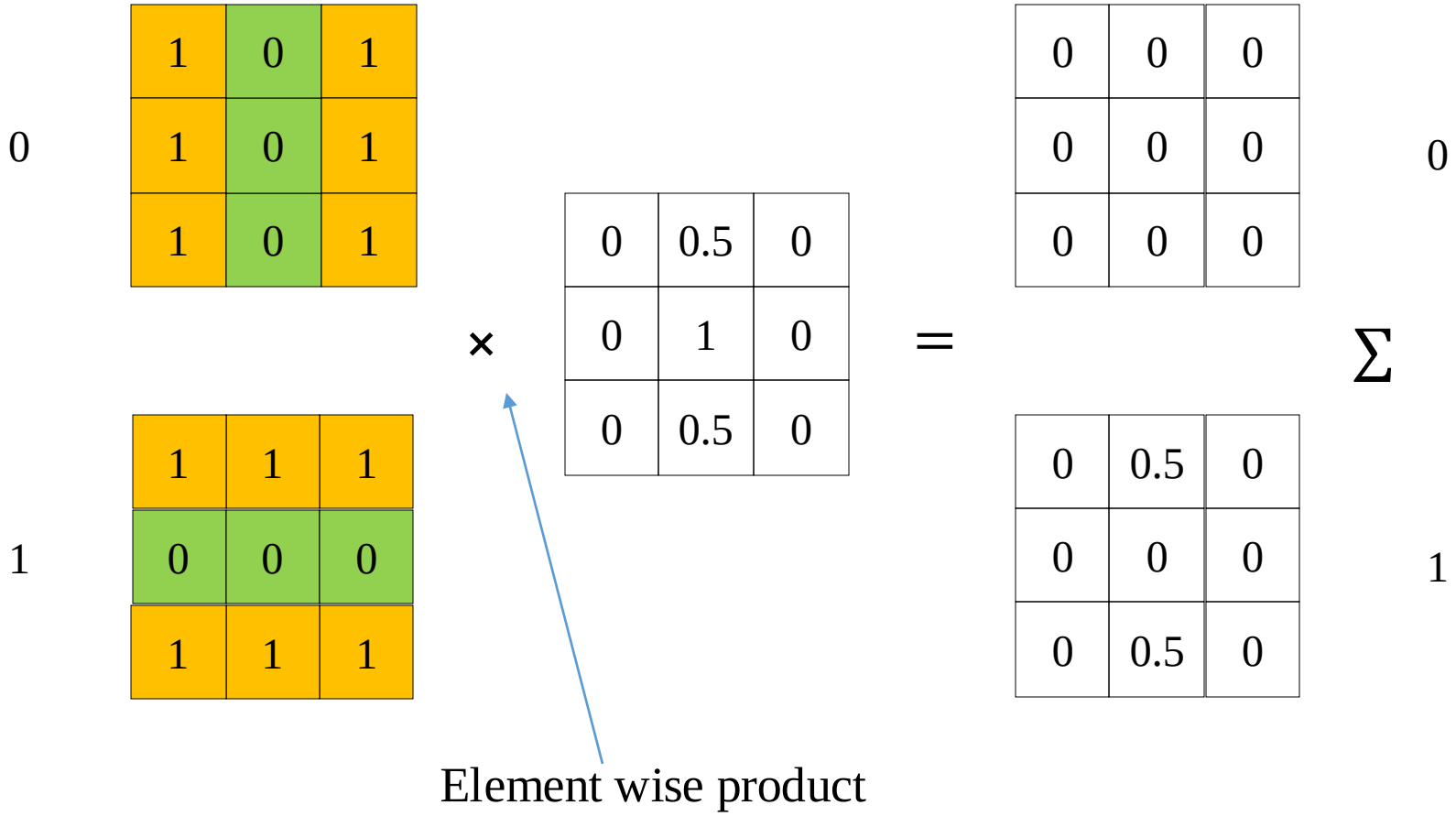
0	0.5	0
0	1	0
0	0.5	0

1

1	1	1
0	0	0
1	1	1

Convolutional Neural Network

- Classical Architecture - Convolution



Convolutional Neural Network

- Classical Architecture of CNN

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

These are the network parameters to be learned.

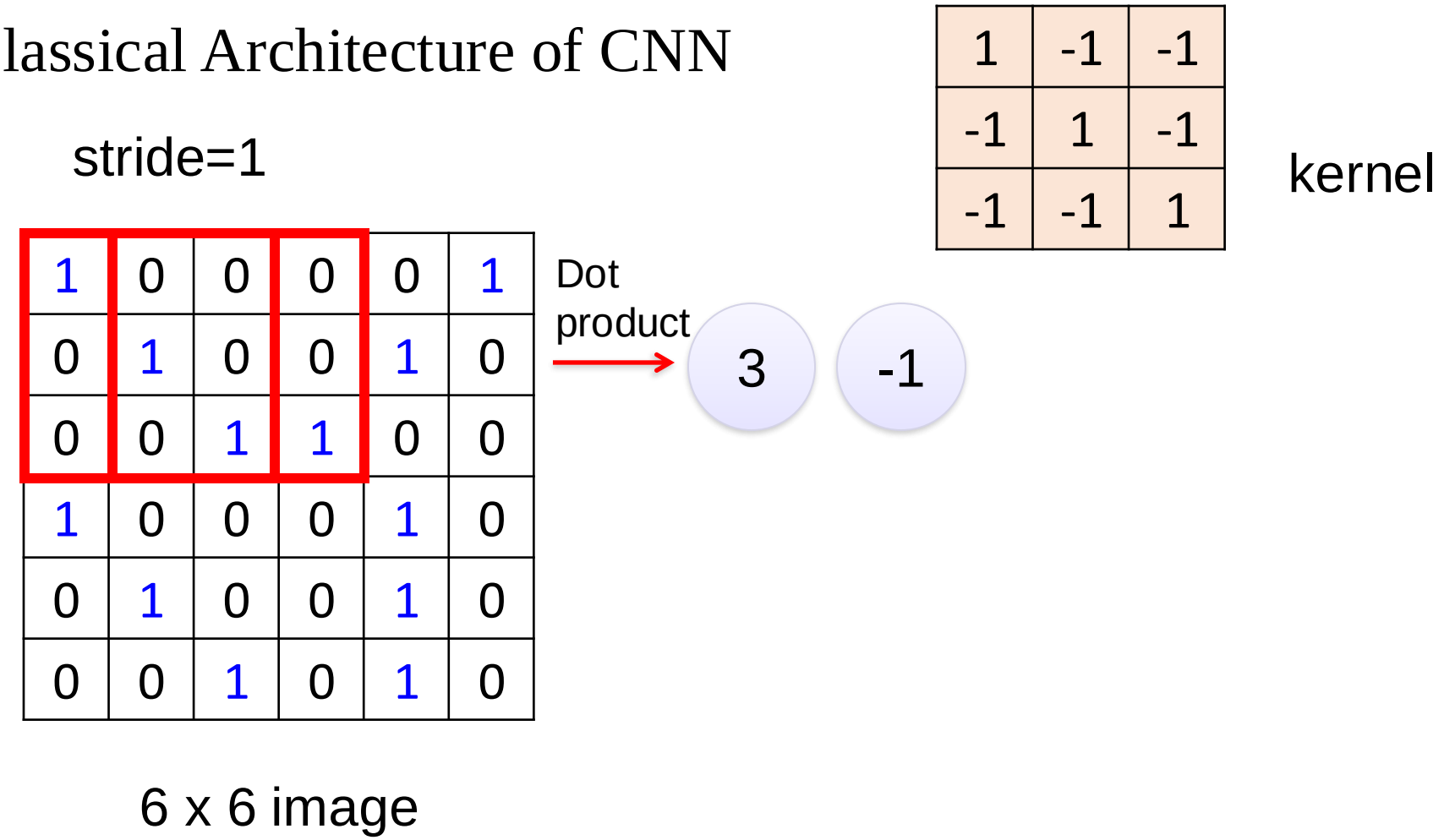
1	-1	-1
-1	1	-1
-1	-1	1

Kernel/Filter

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

Convolutional Neural Network

- Classical Architecture of CNN



Convolutional Neural Network

- Classical Architecture

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

kernel

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

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

Convolutional Neural Network

- Classical Architecture of CNN

If stride=2

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

kernel

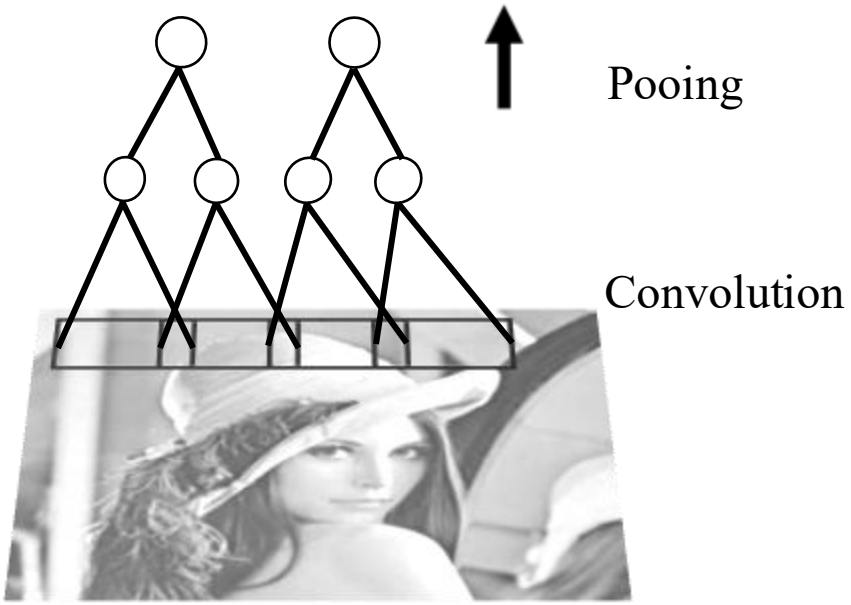
3 -3

Convolutional Neural Network

- Classical Architecture - Example

Input $X = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$

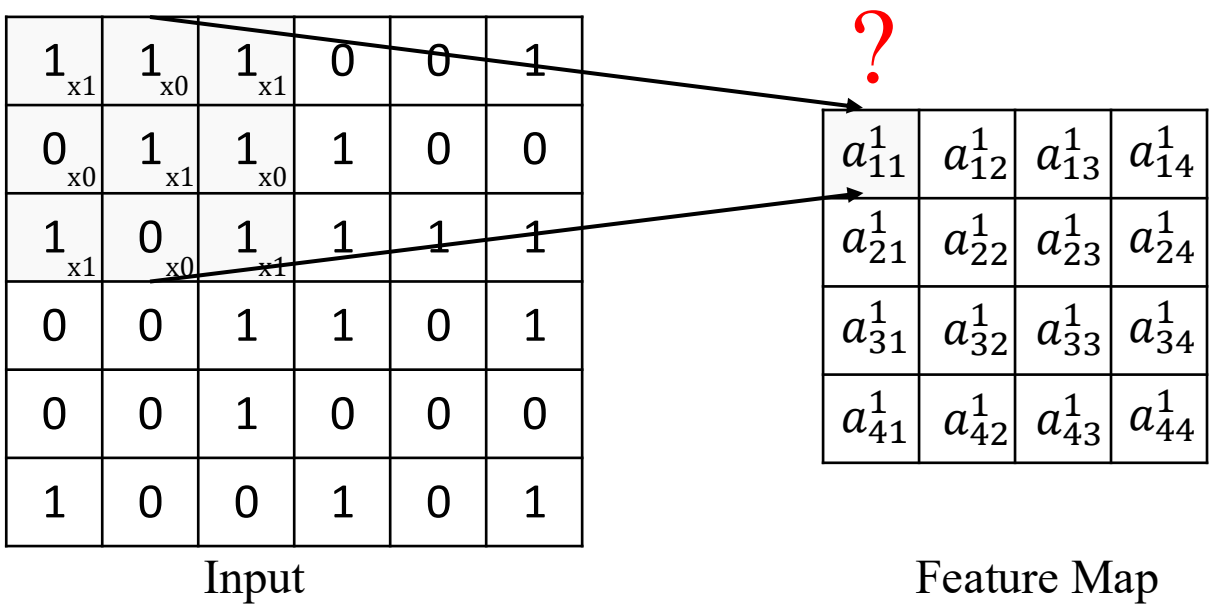
Filter $w = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$



Convolutional Neural Network

Classical Architecture - Example

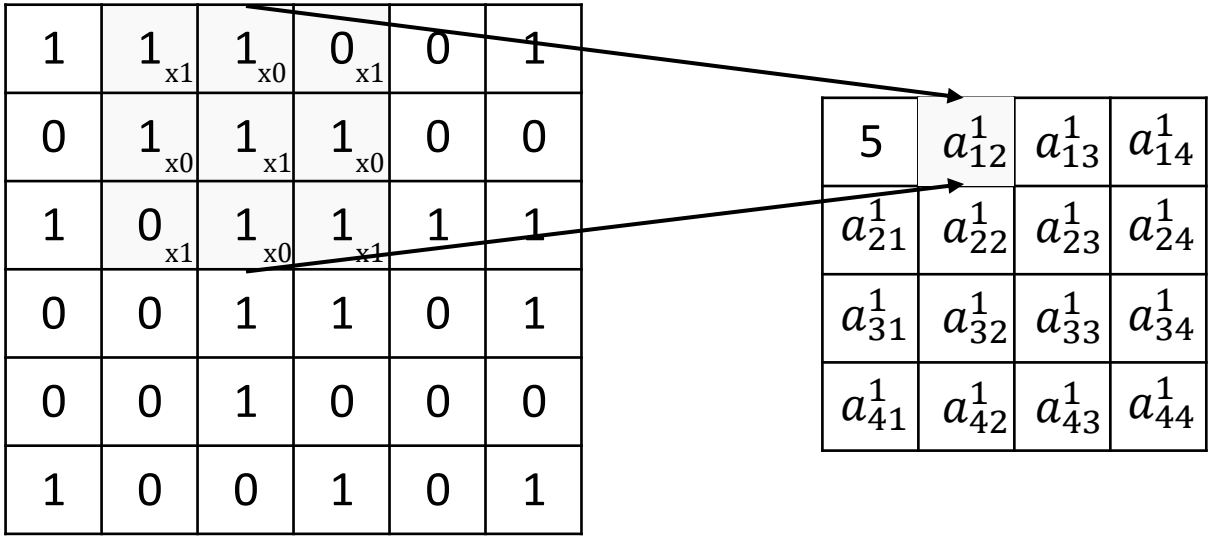
$$\begin{cases} z_{11}^1 = 1\times1+1\times0+1\times1+0\times0+1\times1+1\times0+1\times1+0\times0+1\times1 \\ \quad = 5 \\ a_{11}^1 = f(z_{11}^1) = 5 \end{cases}$$



Convolutional Neural Network

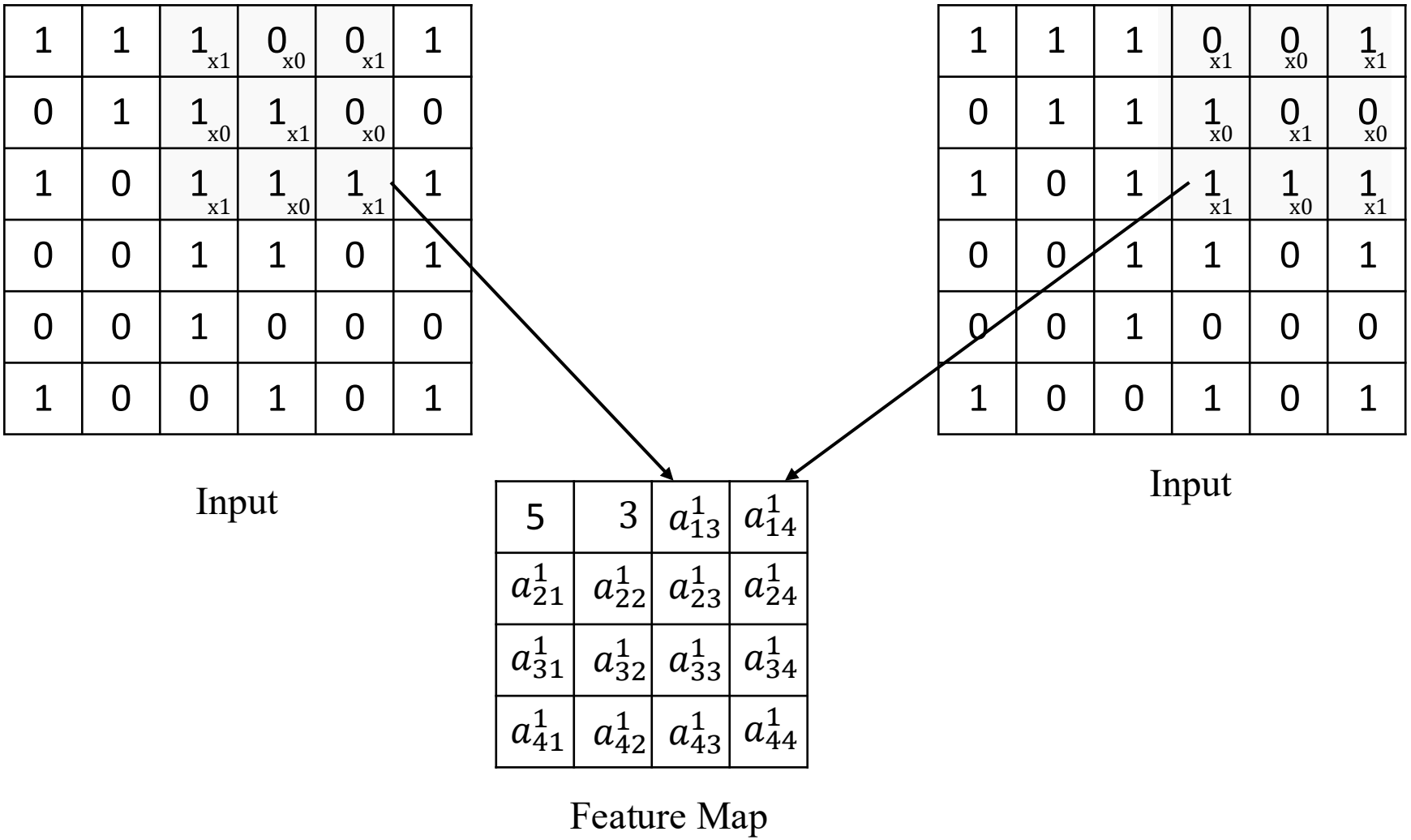
Classical Architecture - Example

$$\begin{cases} z_{12}^1 = 1\times1+1\times0+0\times1+1\times0+1\times1+1\times0+0\times1+1\times0+1\times1 \\ \quad\quad = 3 \\ a_{12}^1 = f(z_{12}^1) = 3 \end{cases}$$



Convolutional Neural Network

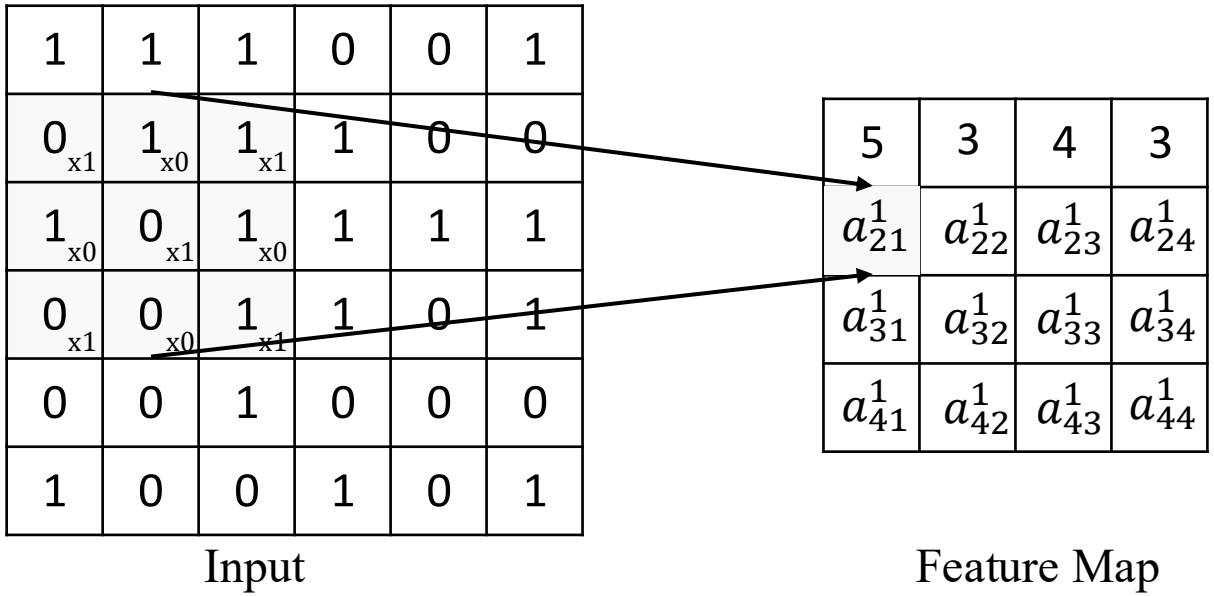
- Classical Architecture - Example



Convolutional Neural Network

Classical Architecture - Example

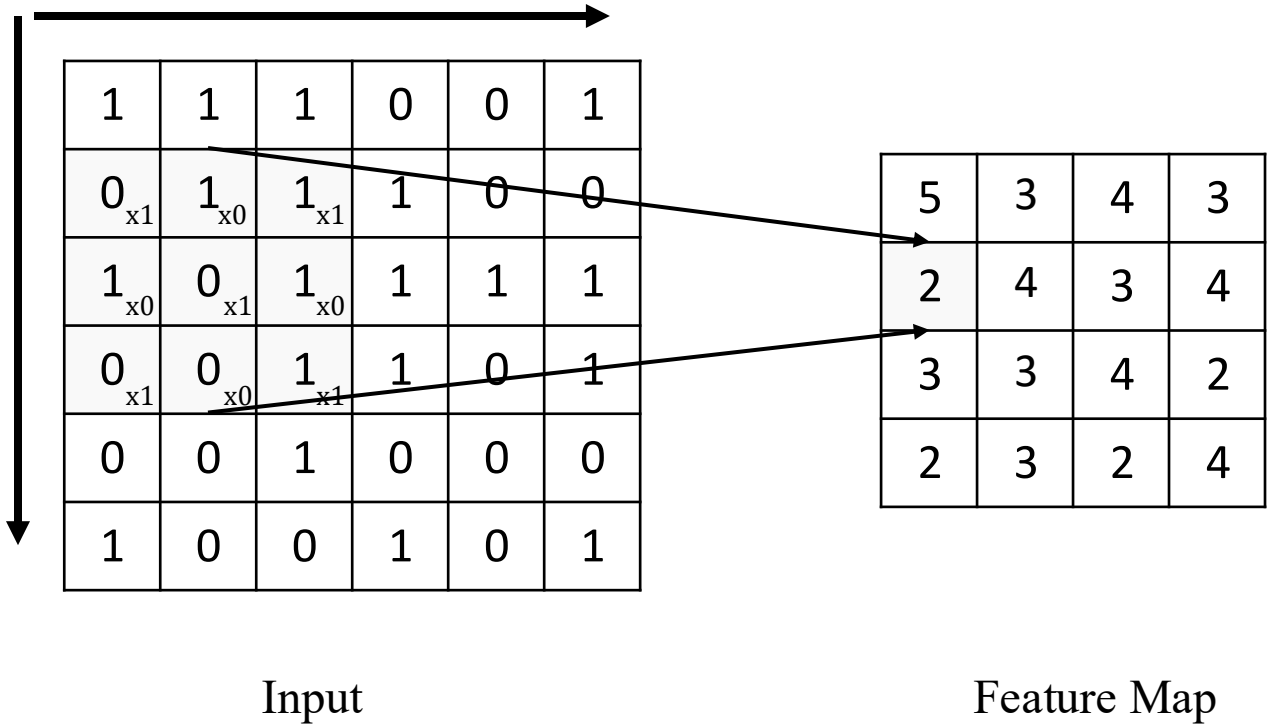
$$\begin{cases} z_{21}^1 = 0 \times 1 + 1 \times 0 + 1 \times 1 + 1 \times 0 + 0 \times 1 + 1 \times 0 + 0 \times 1 + 0 \times 0 + 1 \times 1 \\ \quad\quad = 2 \\ a_{21}^1 = f(z_{21}^1) = 2 \end{cases}$$



Convolutional Neural Network

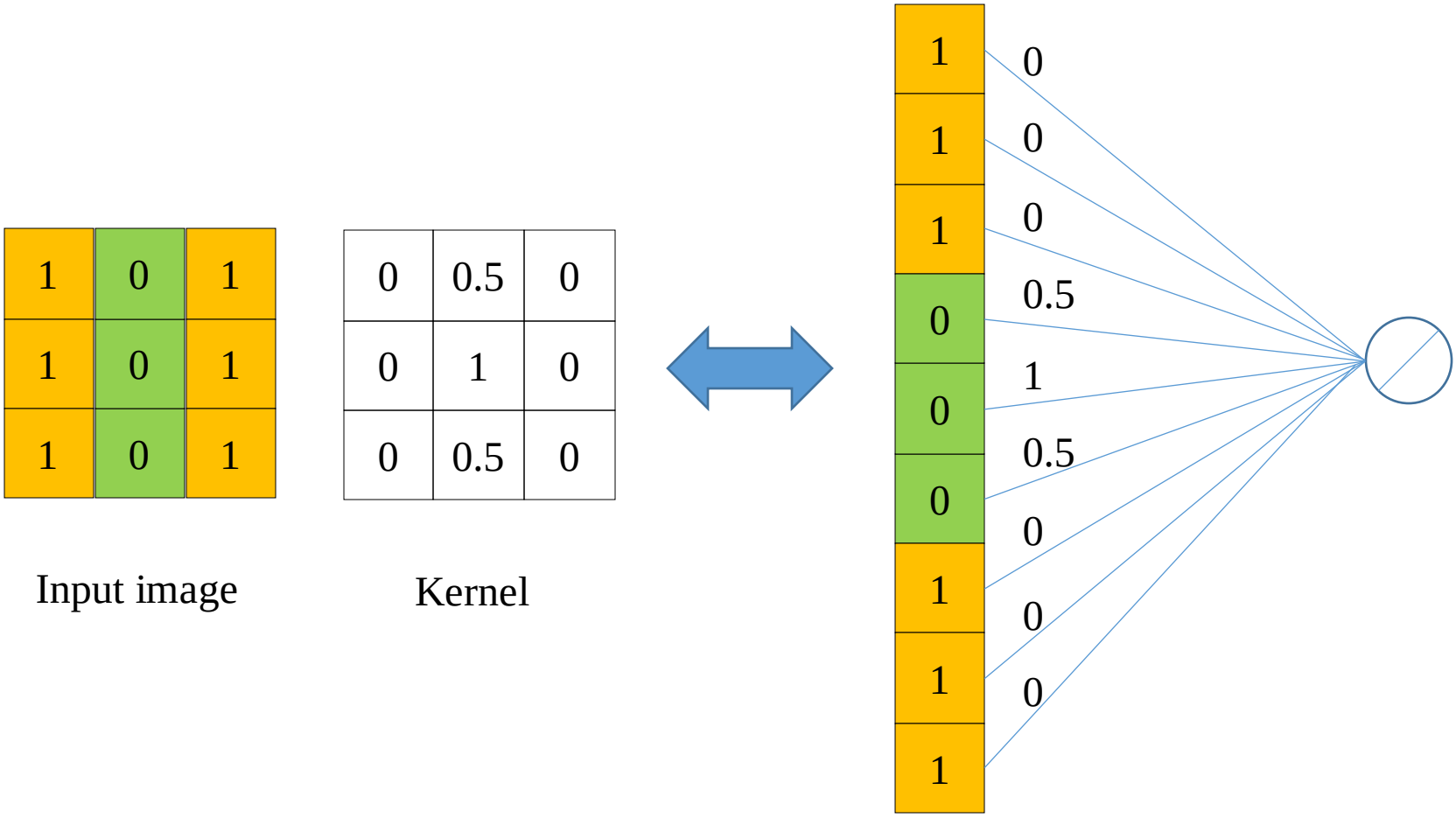
- Classical Architecture - Example

$$\begin{cases} z^1 = W * X \\ a^1 = f(z^1) \end{cases}$$



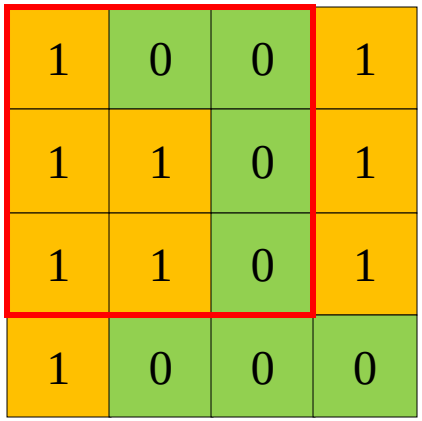
Convolutional Neural Network

- Classical Architecture - Convolution



Convolutional Neural Network

- Classical Architecture - Convolution



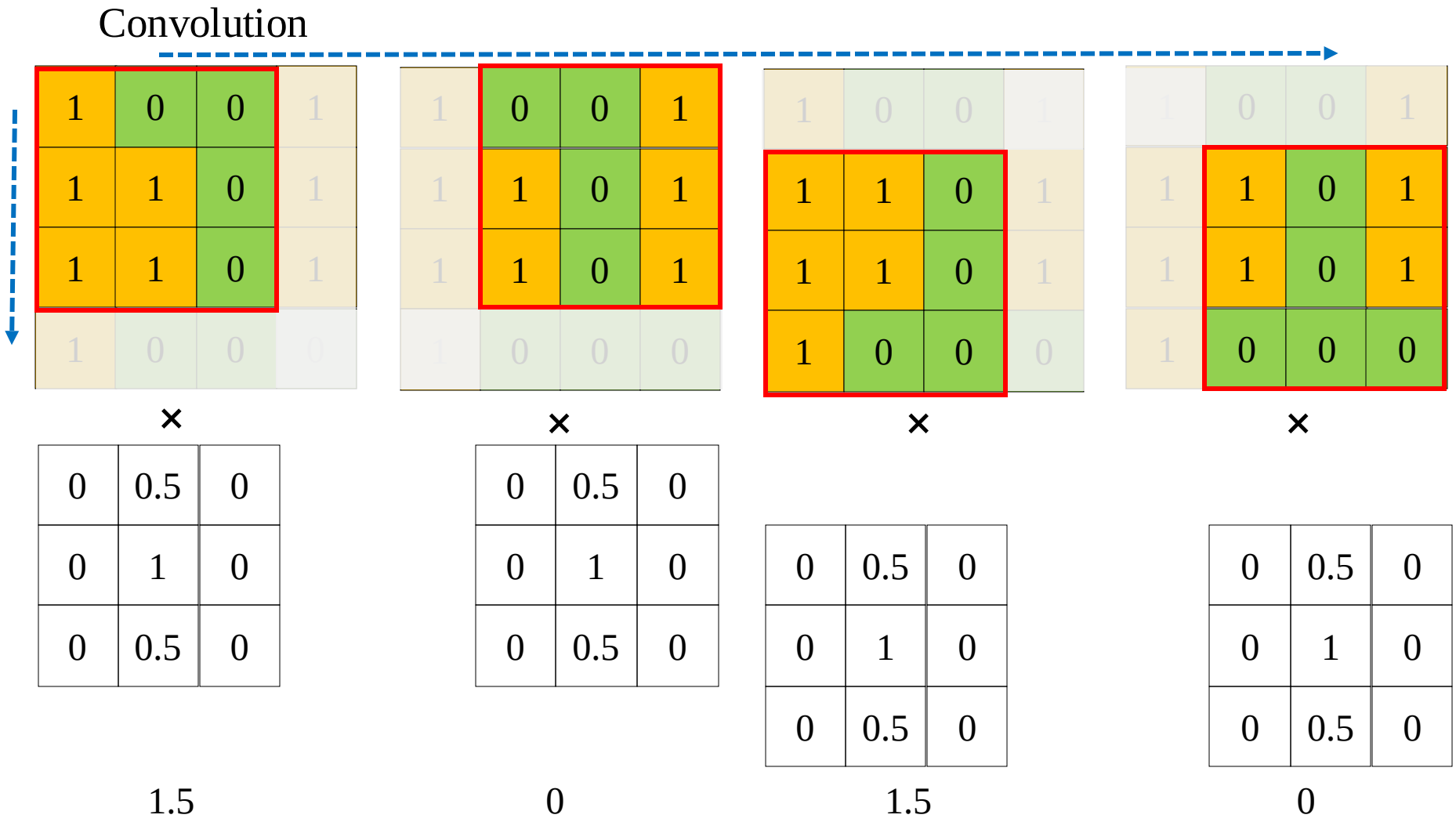
Input image

0	0.5	0
0	1	0
0	0.5	0

Kernel

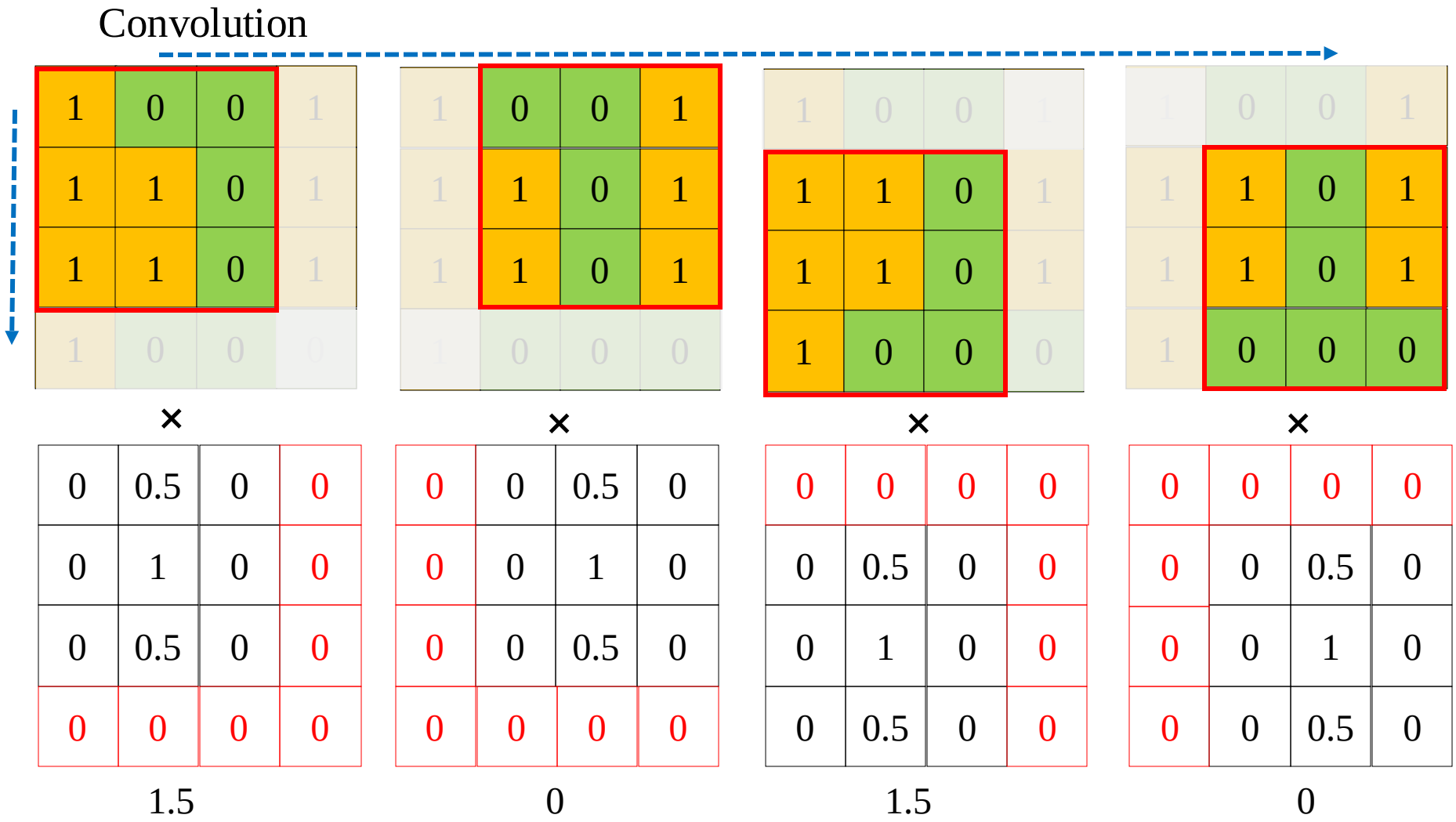
Convolutional Neural Network

Classical Architecture - Convolution



Convolutional Neural Network

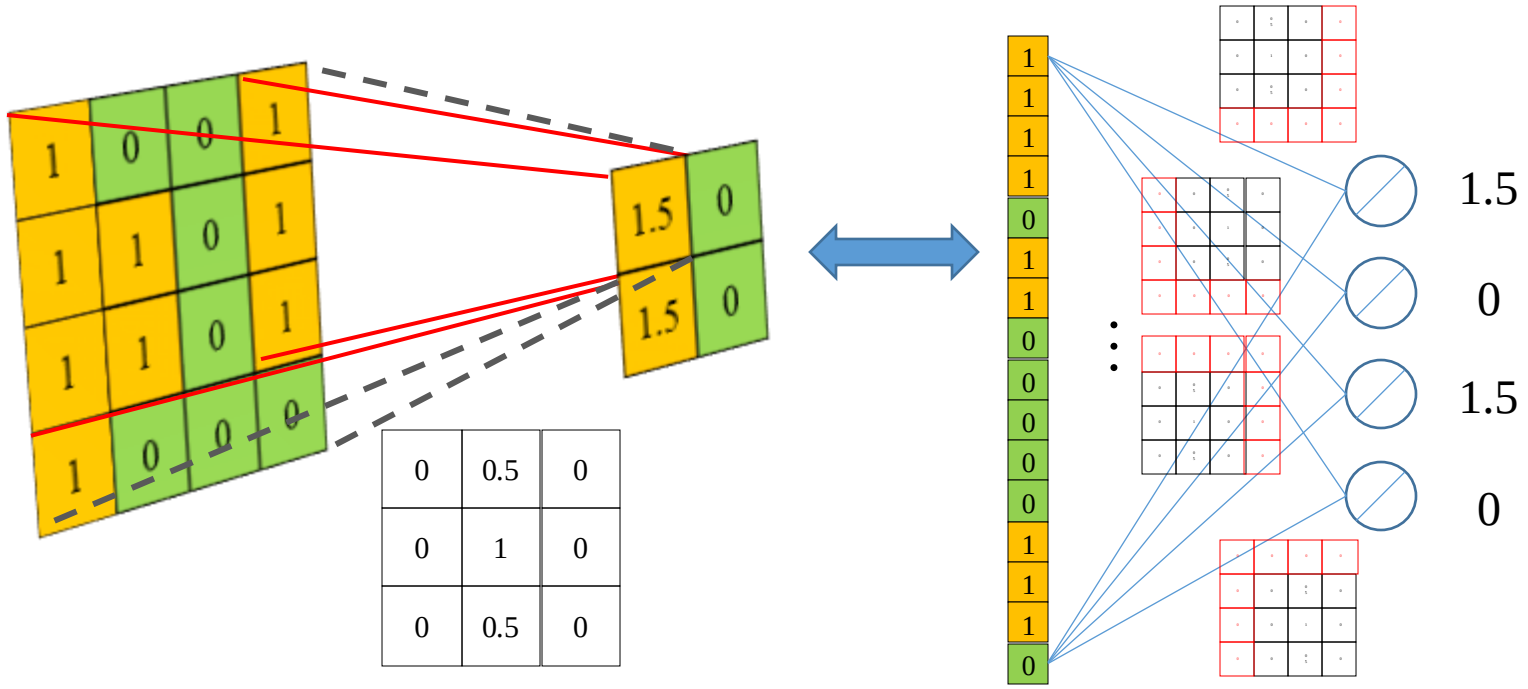
Classical Architecture - Convolution



Convolutional Neural Network

Classical Architecture - Convolution

Each convolutional kernel is equivalent to several spatially constrained FC neurons



1 *convolution kernel* $\Leftrightarrow$ $(W - W_k + 1) \times (H - H_k + 1)$ *neurons*

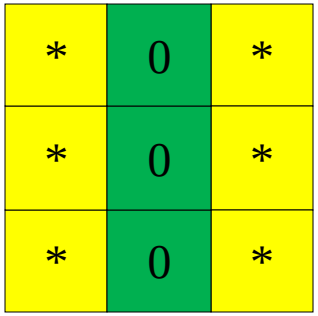
Convolutional Neural Network

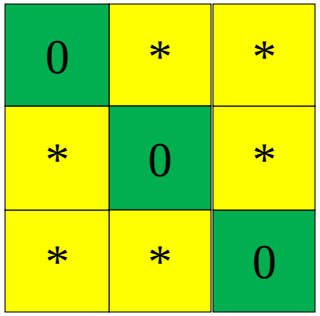
Classical Architecture - Convolution

Convolutional Kernels are special masks(filters) that respond to specific patterns in the input image.

0	0.5	0
0	1	0
0	0.5	0

0.5	0	0
0	1	0
0	0	0.5

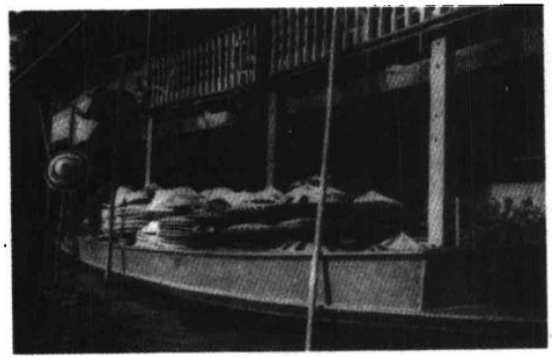




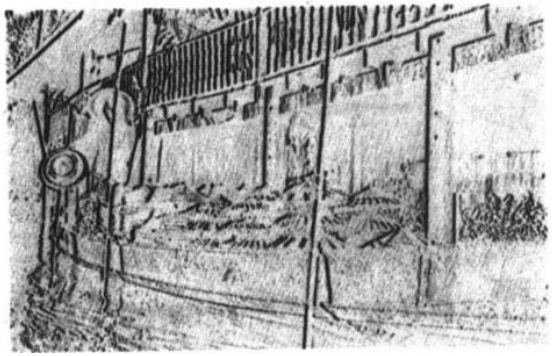
Convolutional Neural Network

Classical Architecture - Example

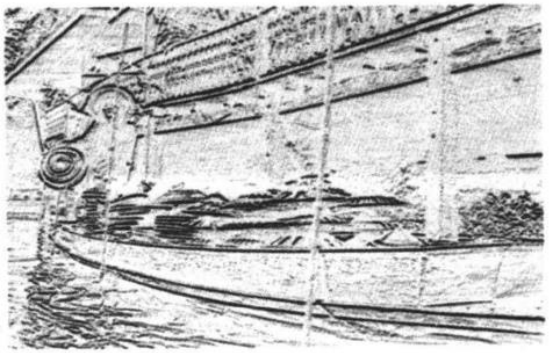
- Multiple Convolutions with Different Filters
  - Different filters detect different features



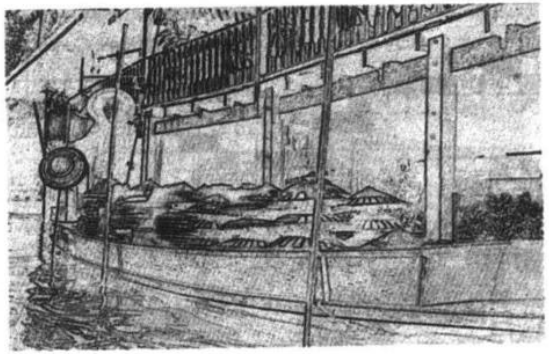
Raw Image



Vertical Edge Detect by w1

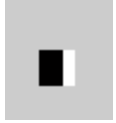


Horizontal Edge Detect by w2



Horizontal & Vertical

$$w1 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

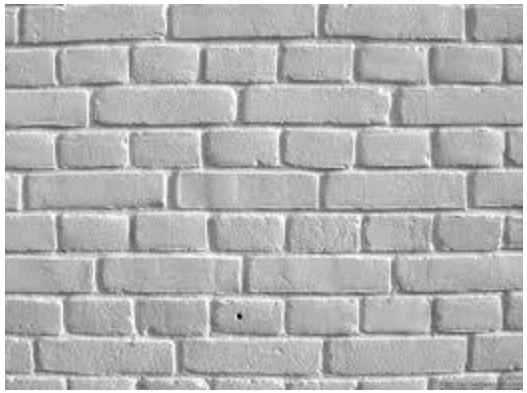


$$w2 = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

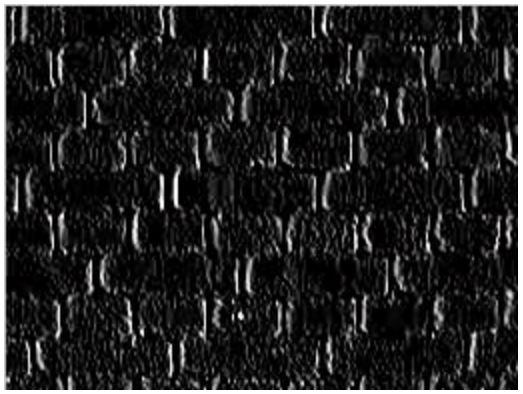
Convolutional Neural Network

Classical Architecture - Example

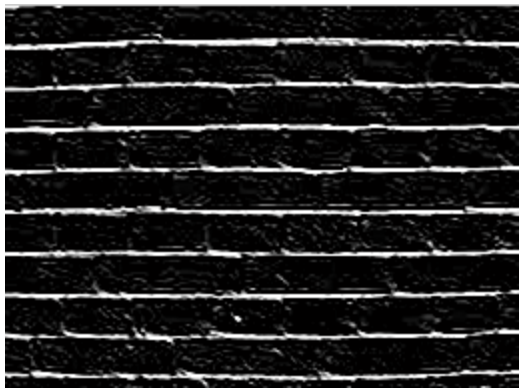
- Multiple Convolutions with Different Filters
  - Different filters detect different features



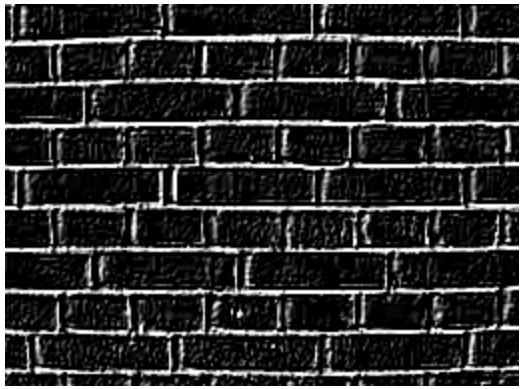
Raw Image



Vertical Edge Detect by w1

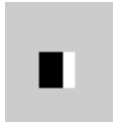


Horizontal Edge Detect by w2



Horizontal & Vertical

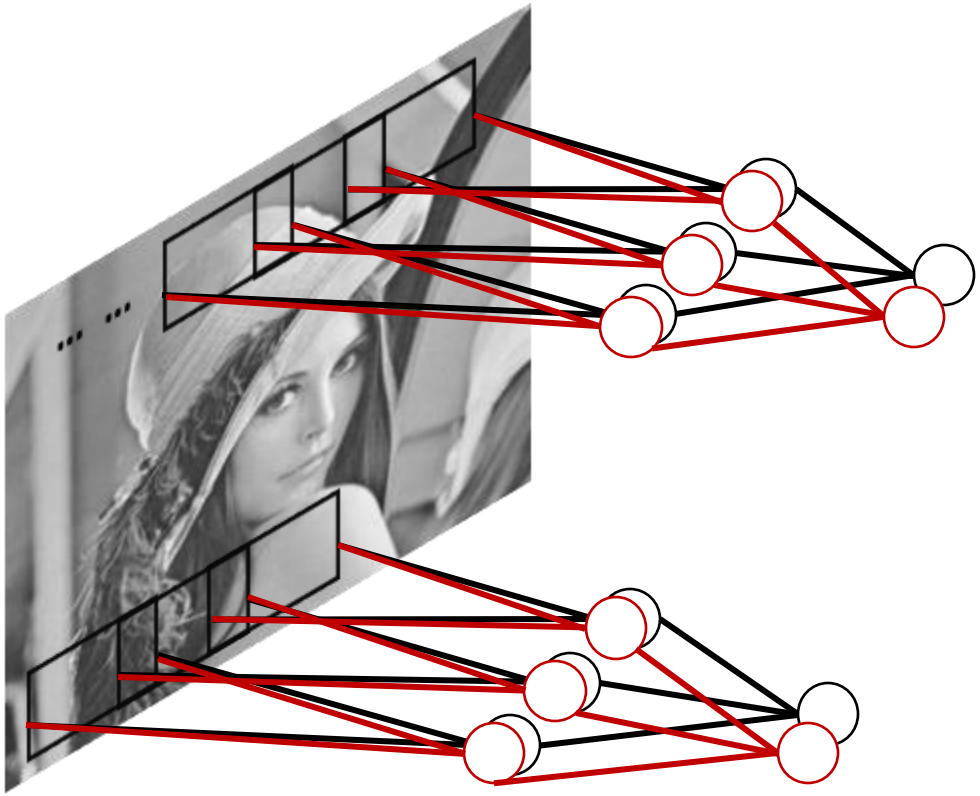
$$w1 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$



$$w2 = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Convolutional Neural Network

- Classical Architecture
  - Multiple Convolutions with Different Filters
    - Detect multiple features at each receptive field



Convolutional Neural Network

- Classical Architecture
  - Multiple Convolutions with Different Filters

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

⋮ ⋮

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

Convolutional Neural Network

- Classical Architecture
  - Multiple Convolutions with Different Filters

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

⋮ ⋮

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

Convolutional Neural Netw

- Classical Architecture

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

filter1

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

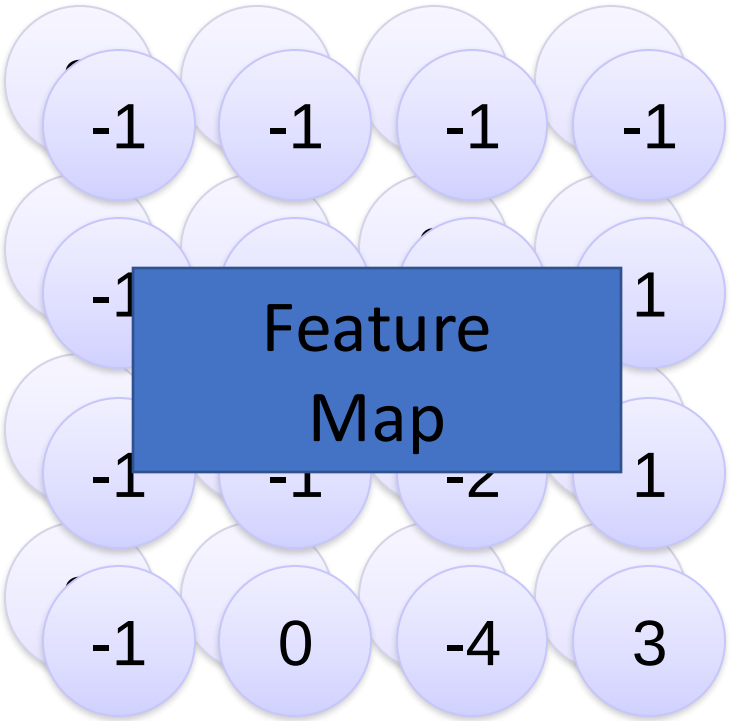
filter 2

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

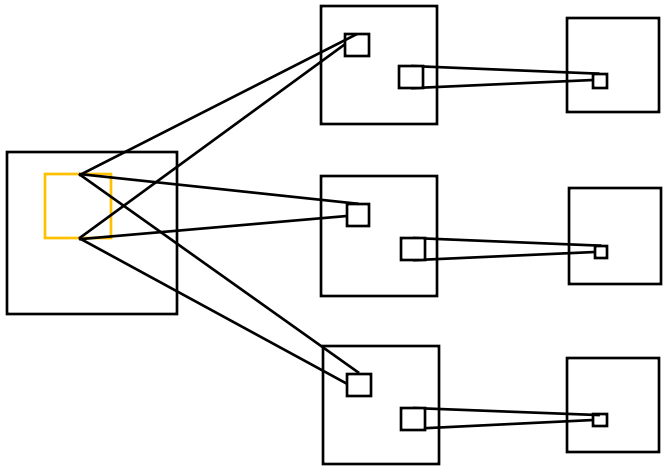
Repeat this for each filter

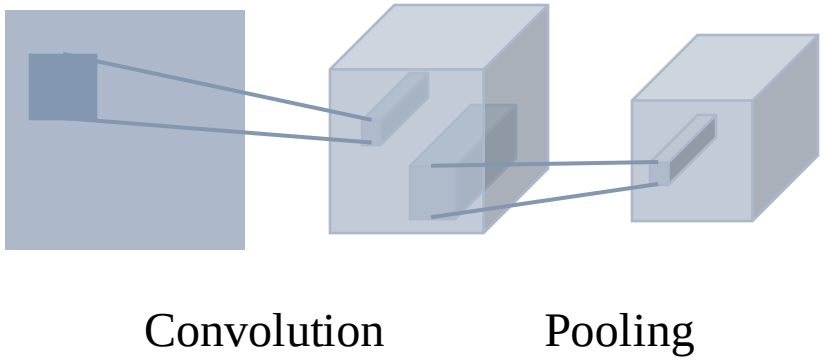


Two 4 x 4 images
Forming 4 x 4 x 2matrix

Convolutional Neural Network

- Classical Architecture
  - Multiple Convolutions with Different Filters
    - There results a 3D array, where each slice is a feature map.
    - Then pooling each feature map individually

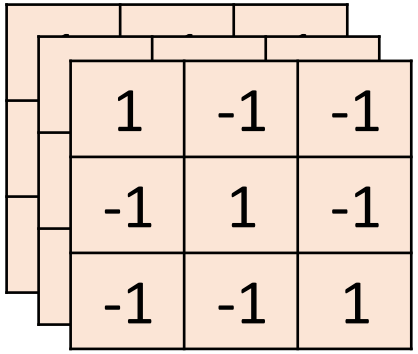




Convolutional Neural Network

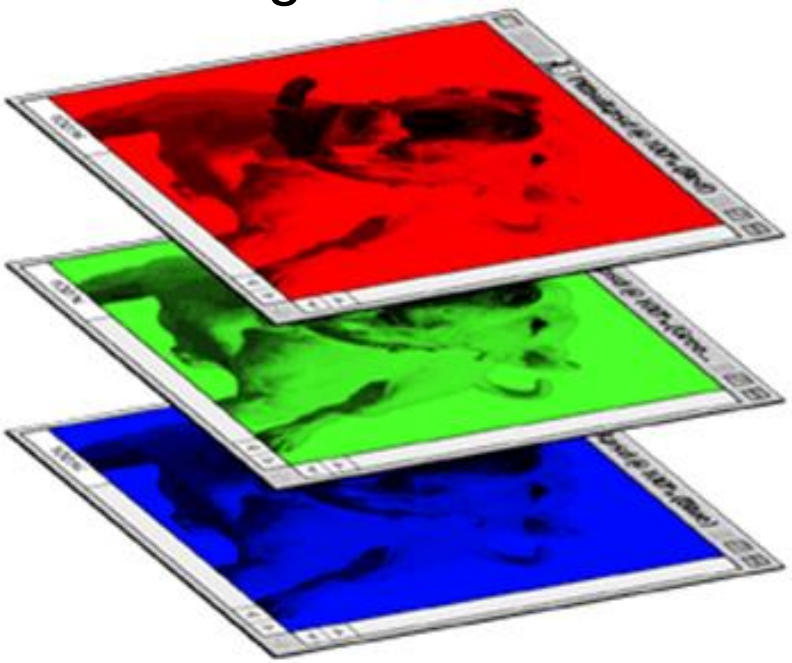
- Classical Architecture
  - Color image: RGB 3 channels

(use 1 filter with 3 kernels/channels)

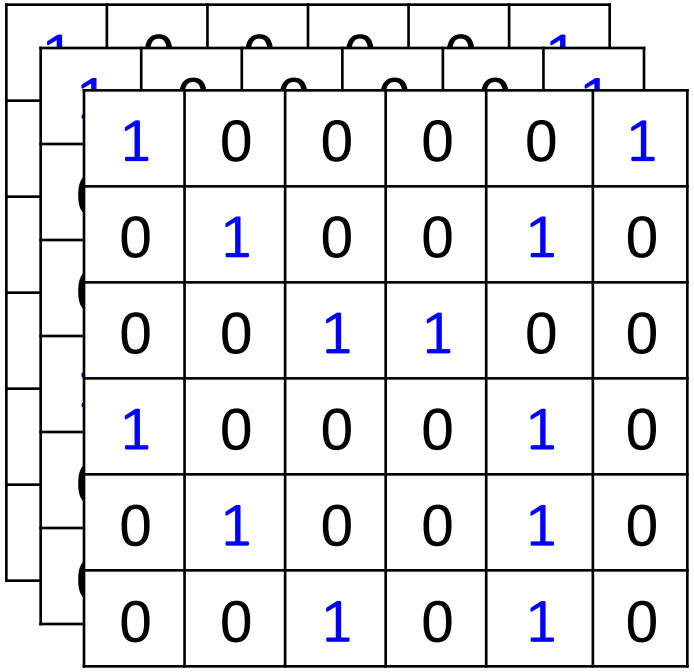


Filter

Color image

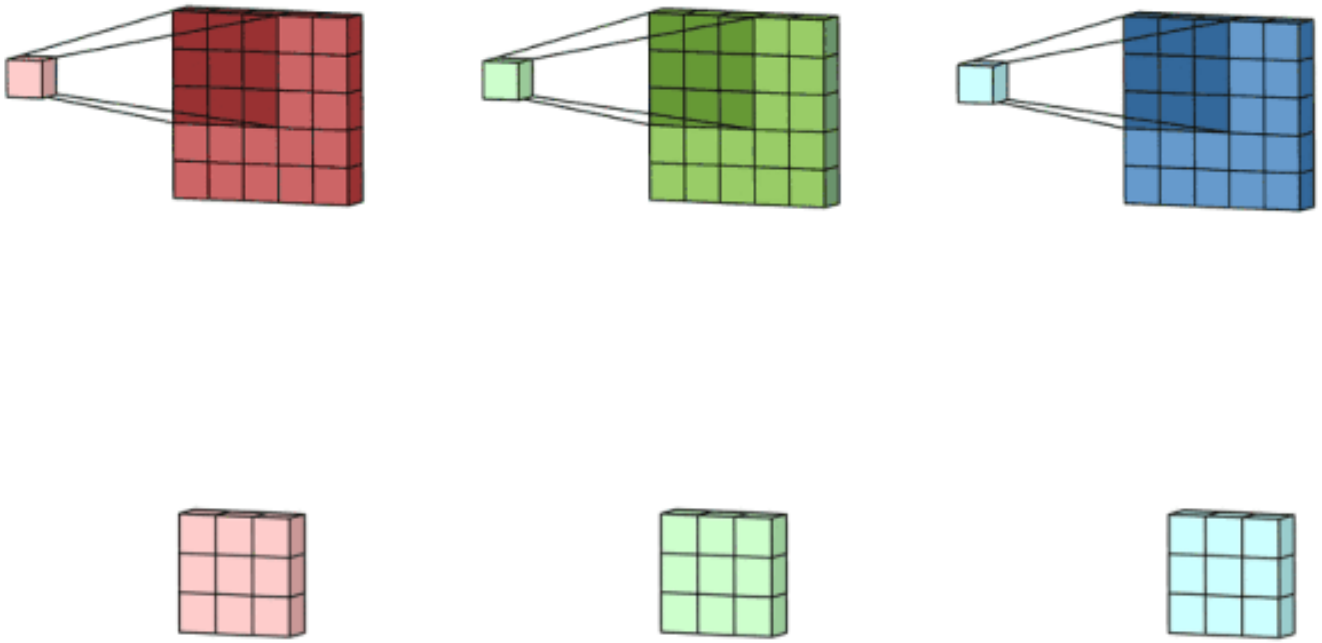


=



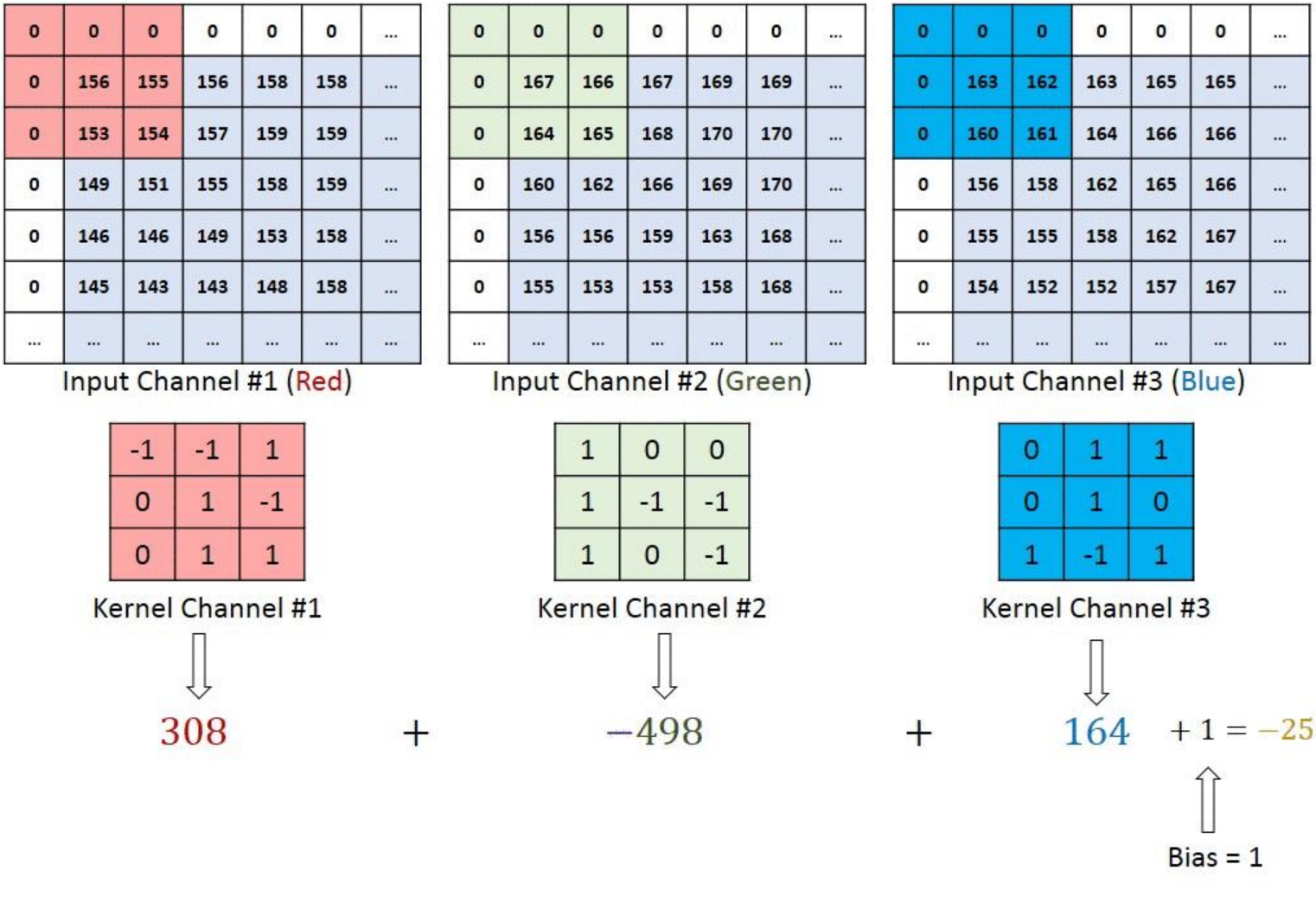
Convolutional Neural Network

- Classical Architecture



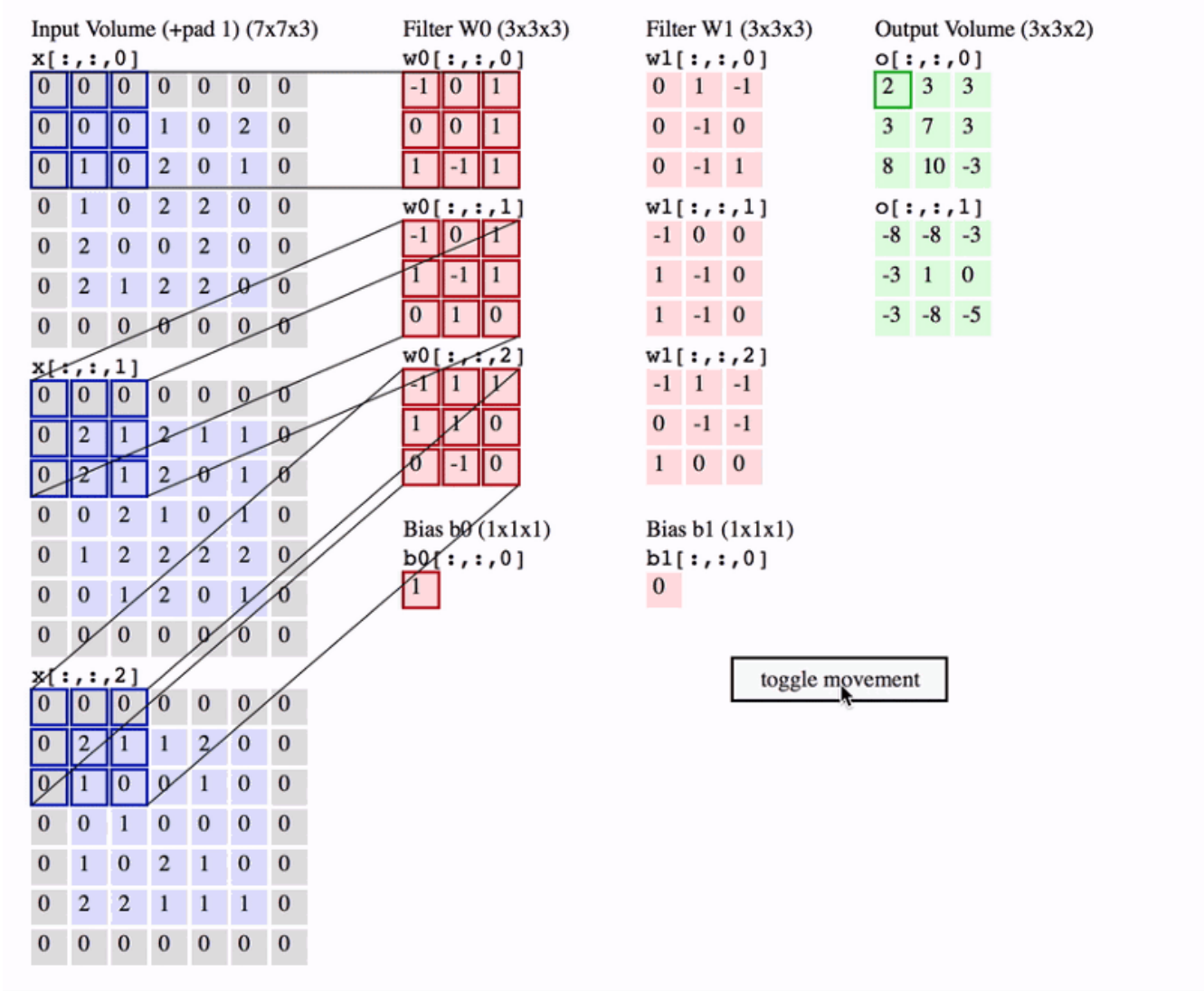
Convolutional Neural Network

Classical Architecture



Convolutional Neural Network

- Classical Architecture (use 2 filters with 3 kernels/channels respectively)



Convolutional Neural Network

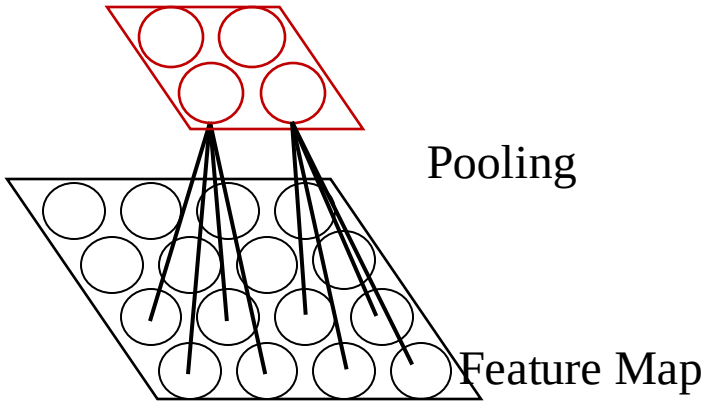
- Classical Architecture
  - Three Main Concepts in CNNs
    - Receptive Field
    - Convolution (Simple Cell)
    - Pooling (Complex Cell)

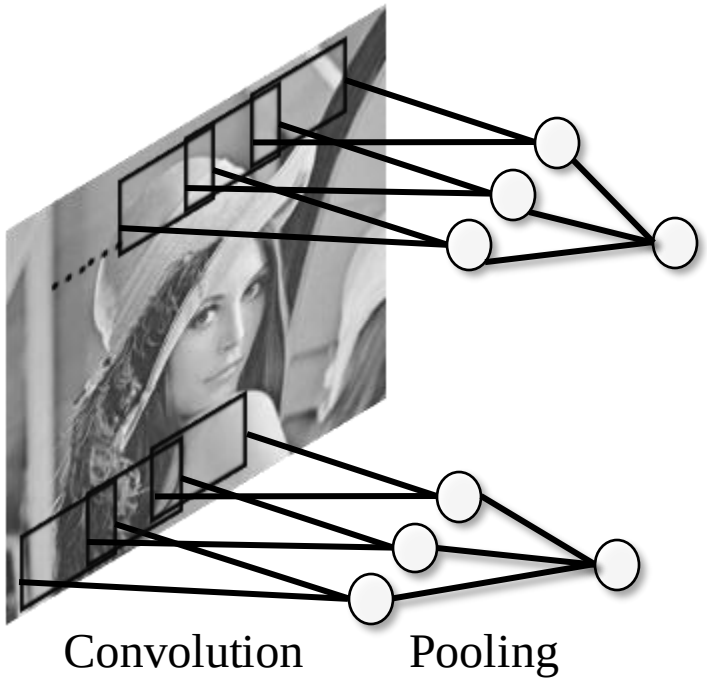
Convolutional Neural Network

Classical Architecture

➢ Pooling (Complex Cell)

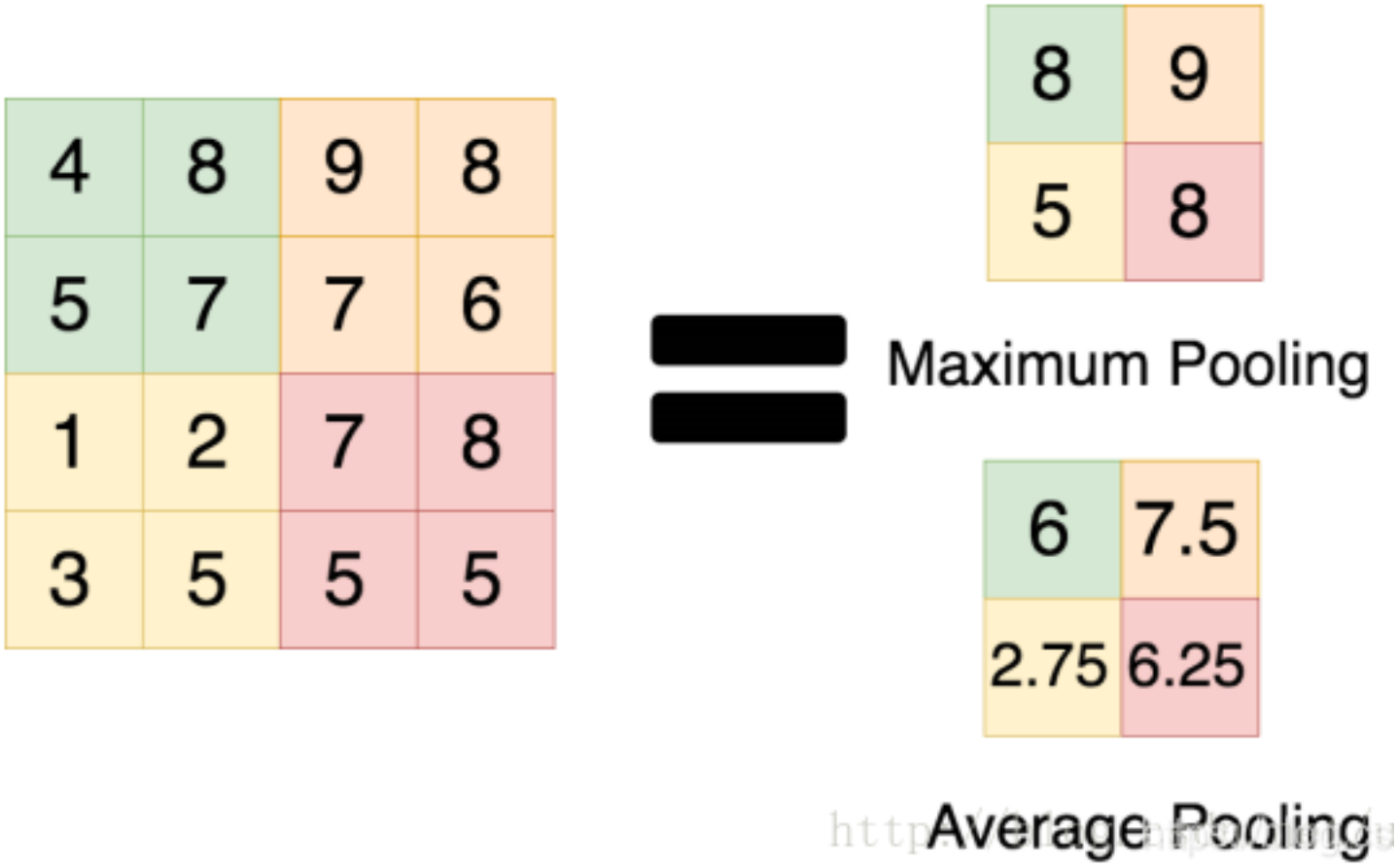
- **Pooling**
  - Pooling is an aggregation operation applied to receptive field on feature map
  - Sometimes are max/mean pooling





Convolutional Neural Network

- Classical Architecture
  - Pooling (Complex Cell)



Convolutional Neural Network

Classical Architecture

➢ Why Pooling

- Subsampling pixels will not change the object

bird

Subsampling

bird

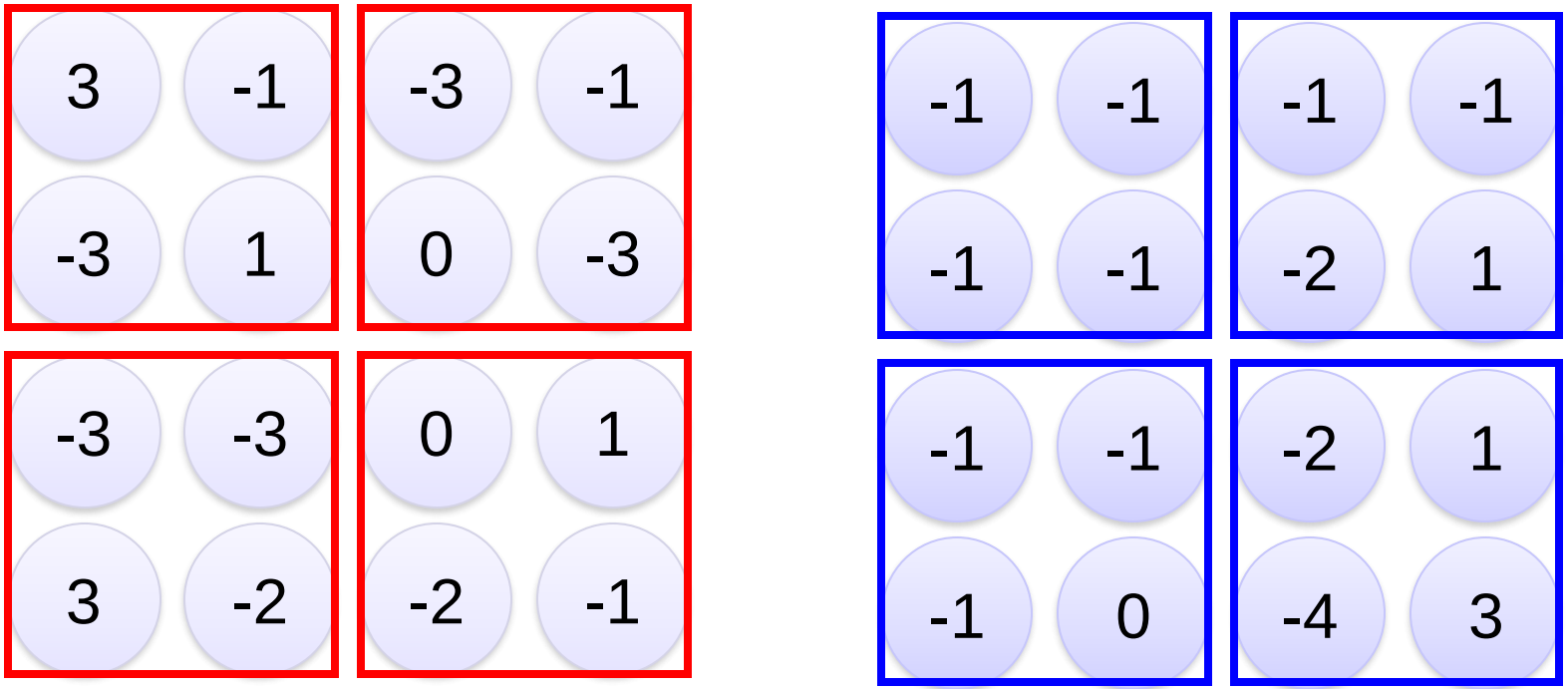
We can subsample the pixels to make image smaller

→ fewer parameters to characterize the image

Convolutional Neural Network

- Classical Architecture
  - Max Pooling

Pooling window 2x2



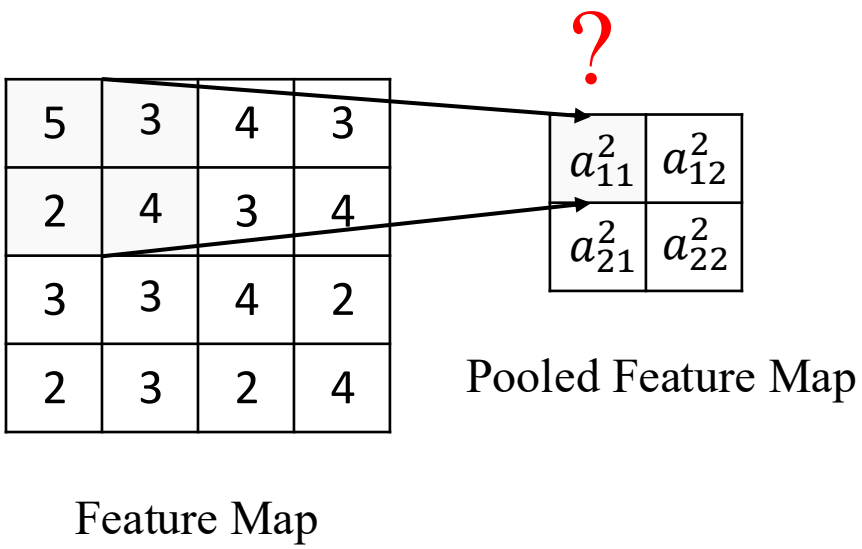
Convolutional Neural Network

- Classical Architecture - Example

Max Pooing

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

Input



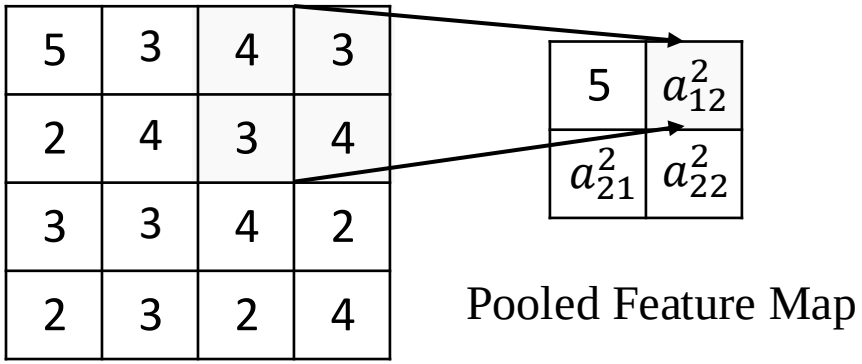
Convolutional Neural Network

- Classical Architecture - Example

Max Pooing

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

Input



Feature Map

Pooled Feature Map

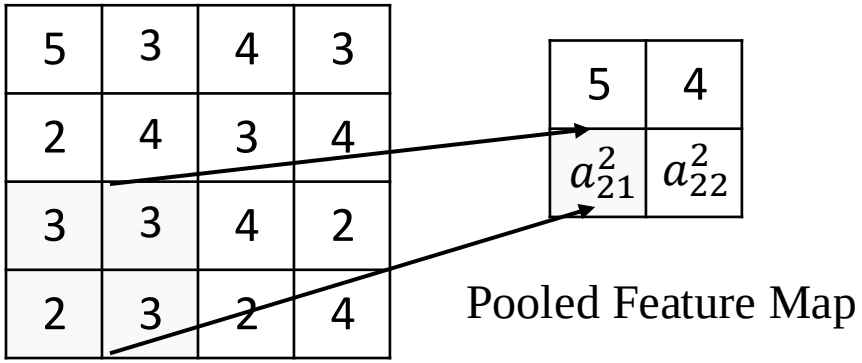
Convolutional Neural Network

- Classical Architecture - Example

Max Pooing

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

Input



Feature Map

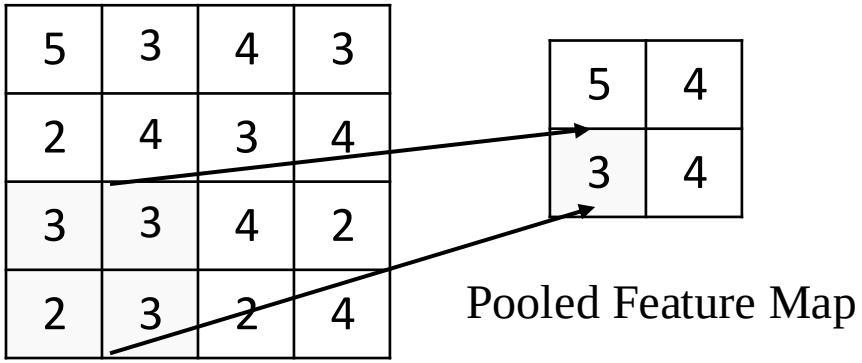
Convolutional Neural Network

- Classical Architecture - Example

Max Pooing

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

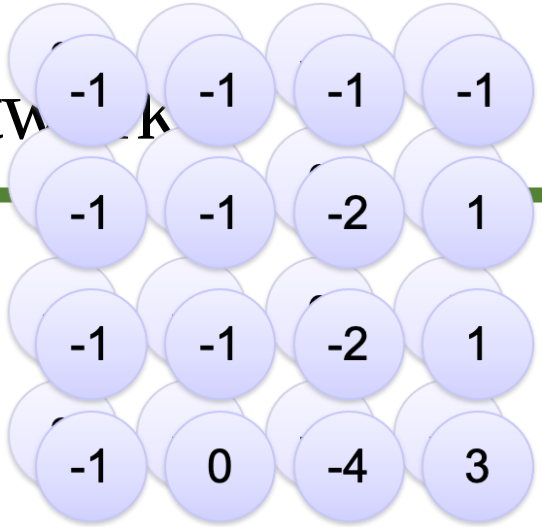
Input



Feature Map

Convolutional Neural Network

- Classical Architecture
  - Pooling



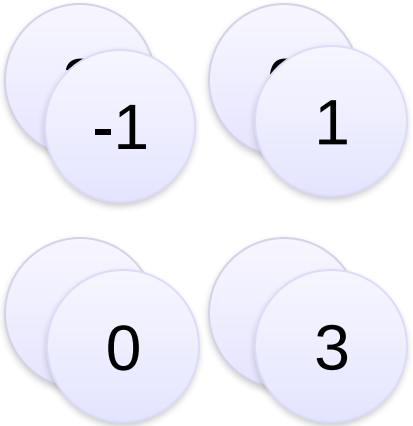
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

6 x 6 image

Conv

Max Pooling

New feature map but smaller

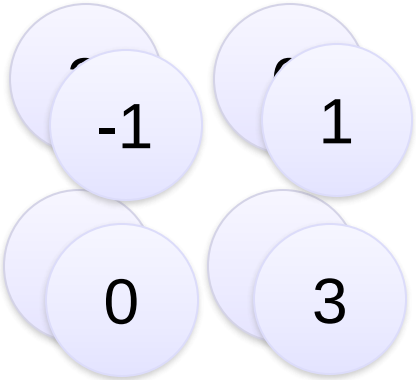


2 x 2 feature map

n filters,
output n channels

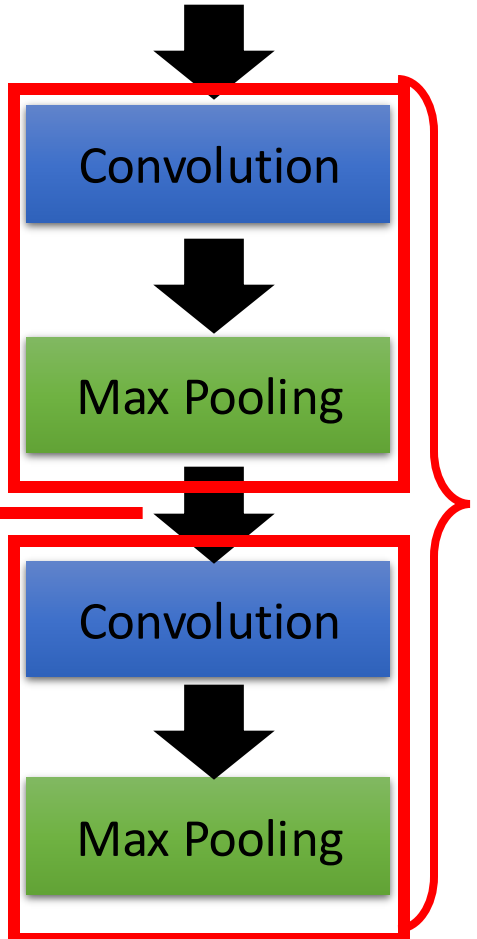
Convolutional Neural Network

- Classical Architecture
  - Pooling



A new feature map

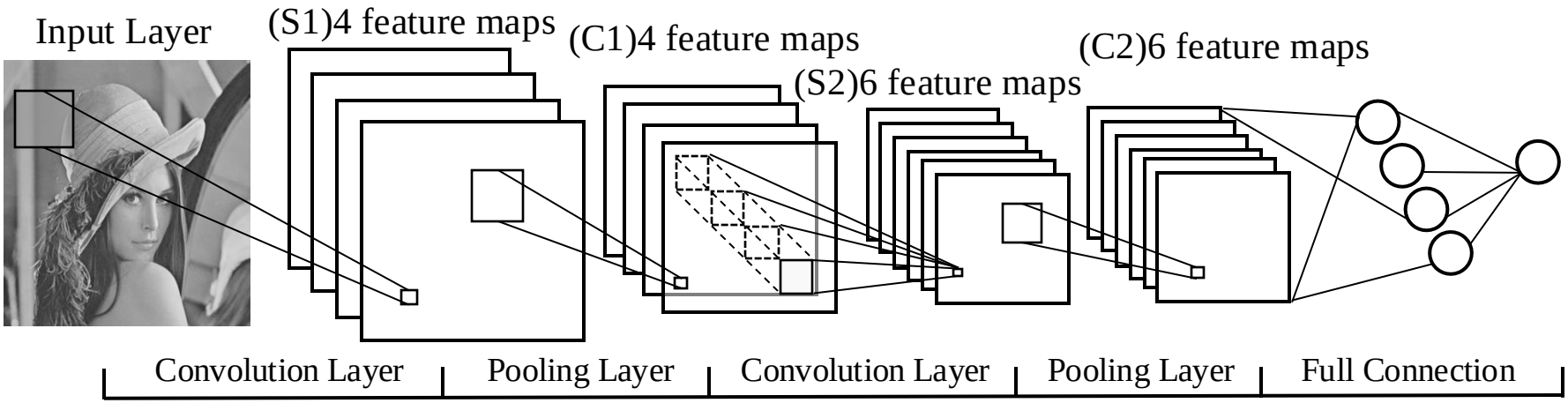
Smaller than the original image



Can repeat many times

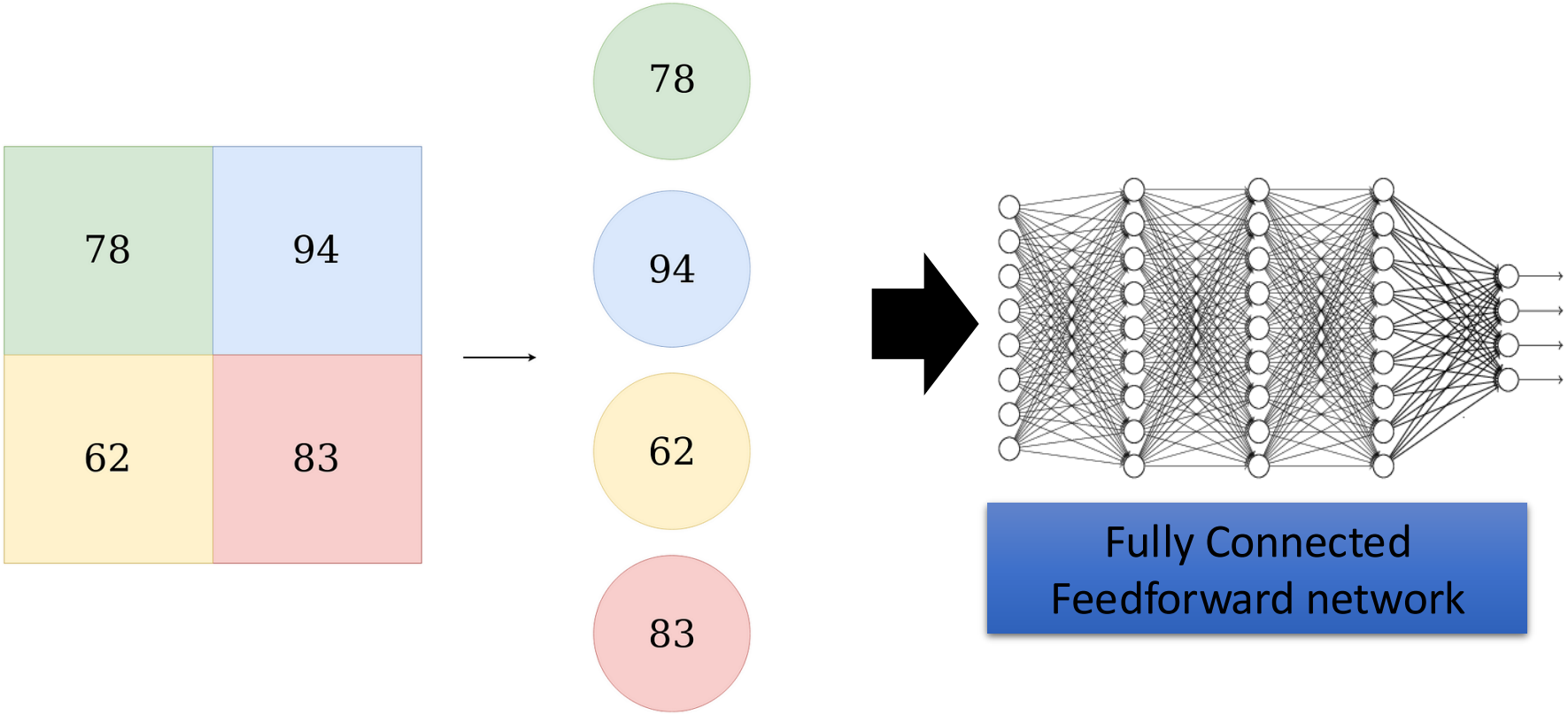
Convolutional Neural Network

- Classical Architecture
  - Added by Full Connection Layers



Convolutional Neural Network

- Classical Architecture
  - Flattening - Full Connection Layers



Neural Networks

- Brief review
- Convolutional Neural Network

  Classical Architecture

  Gradient-Based Learning Algorithm

  Achievements and Applications

Backpropagation

Conclusion: BP for FNN

Forward computing: $y = f(\sum_{i=1}^{n} w_i x_i)$

Define cost function: $J = J(w^1, \cdots, w^{L-1})$

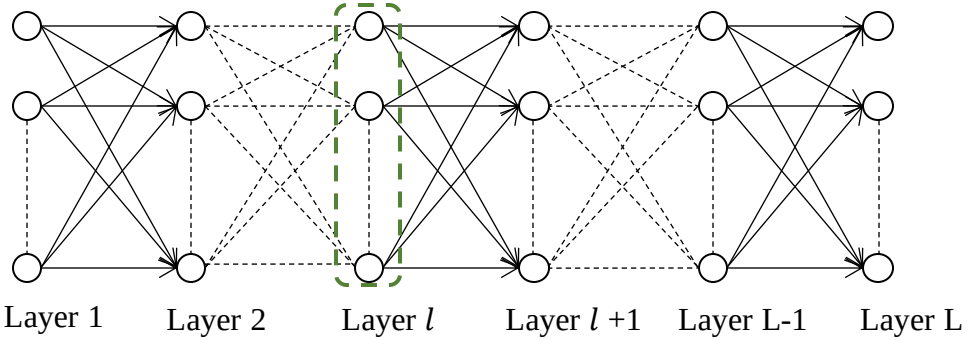
Updating rule: $w_{ji}^l \leftarrow w_{ji}^l - \alpha \cdot \frac{\partial J}{\partial w_{ji}^l}$

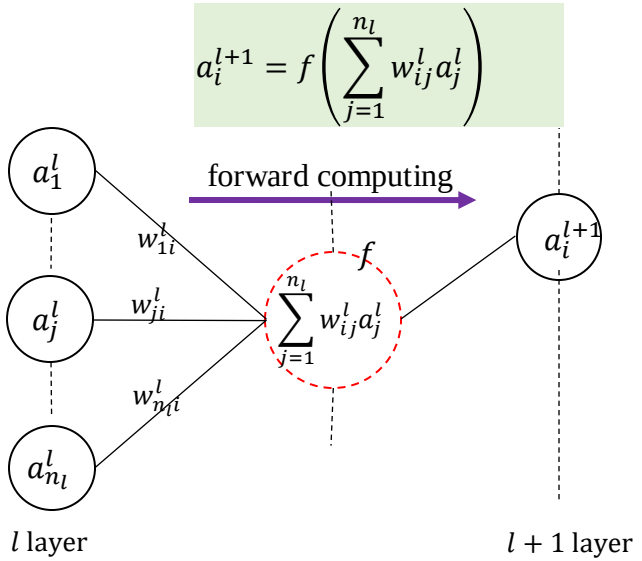
Define δ: $\delta_i^l = \frac{\partial J}{\partial z_i^l}$

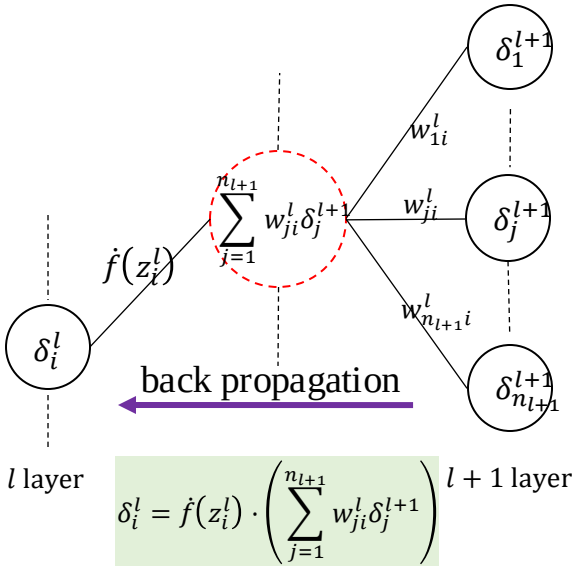
Find the relation: $\frac{\partial J}{\partial w_{ji}^l} = \delta_j^{l+1} \cdot a_i^l$

Back propagation: $\delta_i^L = \frac{\partial J}{\partial z_i^L} = (a_i^L - y_i^L) \cdot \dot{f}(z_i^L)$

$$\delta_i^l = \dot{f}(z_i^l) \cdot \left(\sum_{j=1}^{n_{l+1}} \delta_j^{l+1} \cdot w_{ji}^l \right)$$



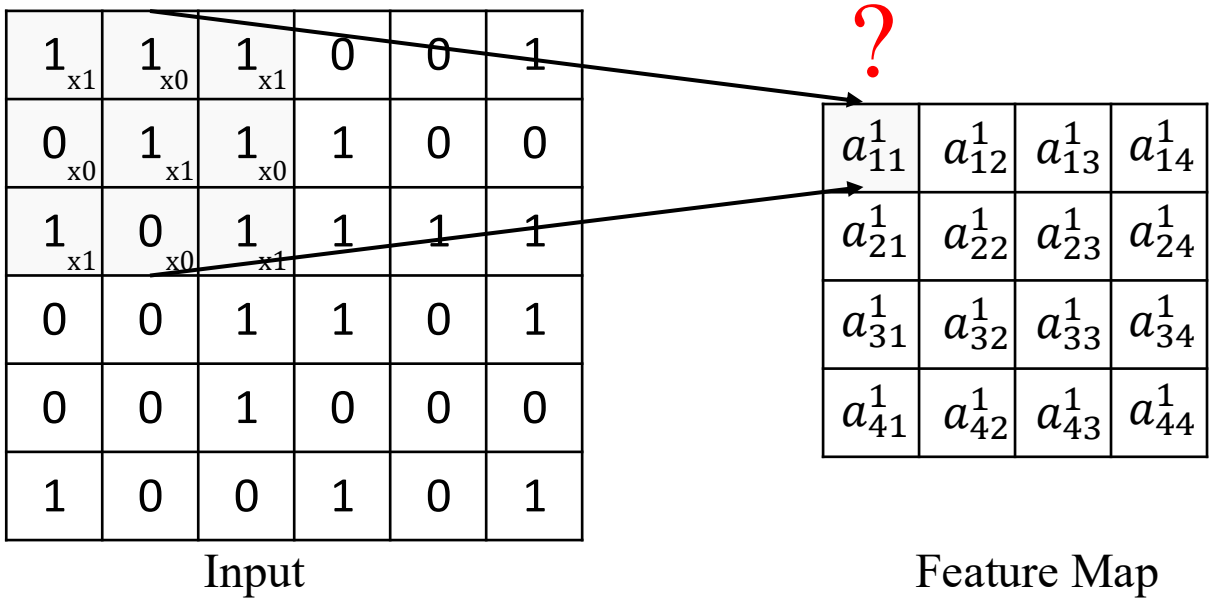




Convolutional Neural Network

Classical Architecture – Conv Example

$$\begin{cases} z_{11}^1 = 1\times1+1\times0+1\times1+0\times0+1\times1+1\times0+1\times1+0\times0+1\times1 \\ \quad = 5 \\ a_{11}^1 = f(z_{11}^1) = 5 \end{cases}$$



Convolutional Neural Network

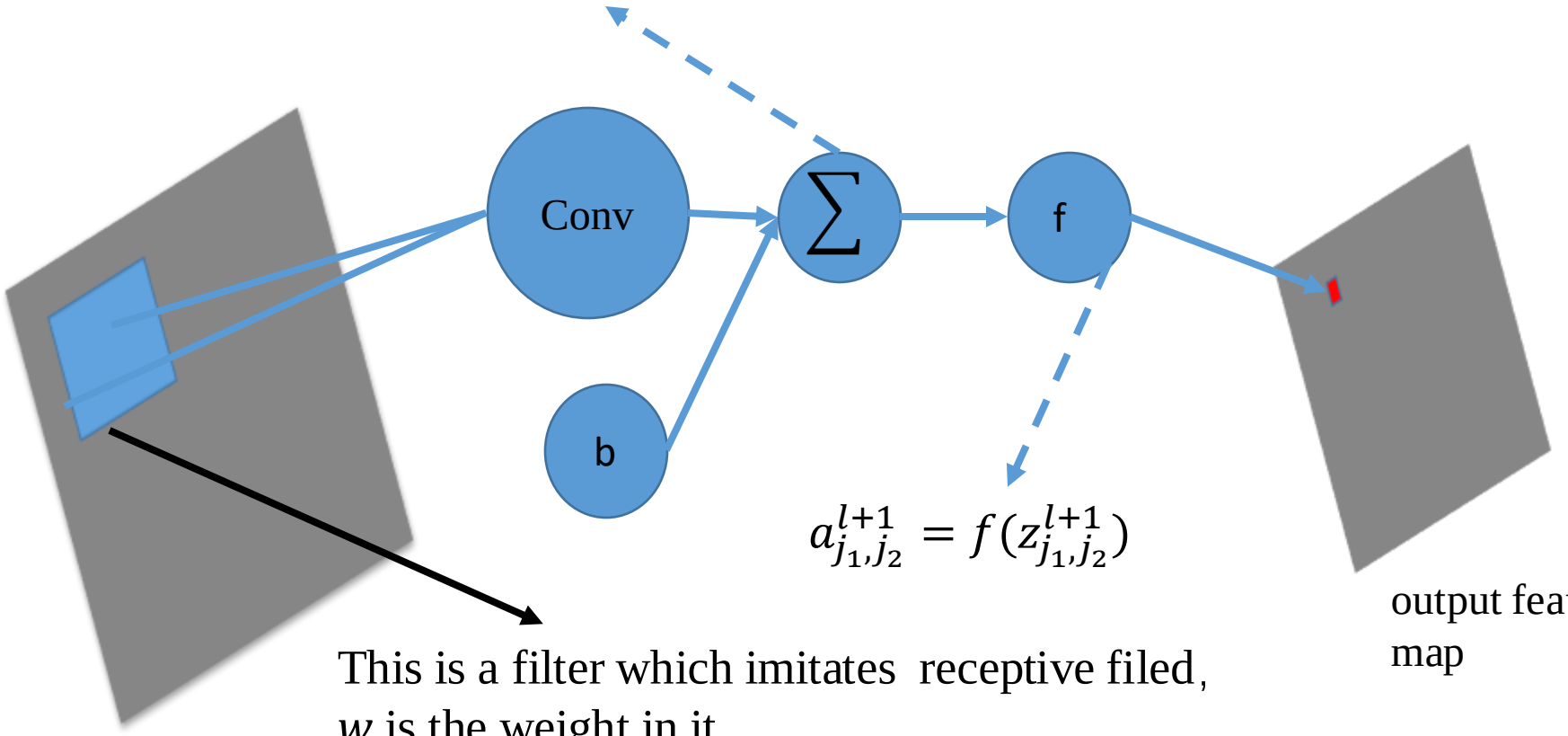
Gradient-Based Learning Algorithm

- Feedforward – Convolution (Conv)

$$z_{j_1,j_2}^{l+1} = \sum_{k_2=1}^{n_{k_2}} \sum_{k_1=1}^{n_{k_1}} a_{j_1+k_1-1,j_2+k_2-1}^{l} * w_{k_1,k_2}^{l} + b$$

Vector form

$$\begin{cases} z^{l+1} = W^{l} * a^{l} + b \\ a^{l+1} = f(z^{l+1}) \end{cases}$$



$$a_{j_1,j_2}^{l+1} = f(z_{j_1,j_2}^{l+1})$$

output feature map

This is a filter which imitates receptive filed, w is the weight in it

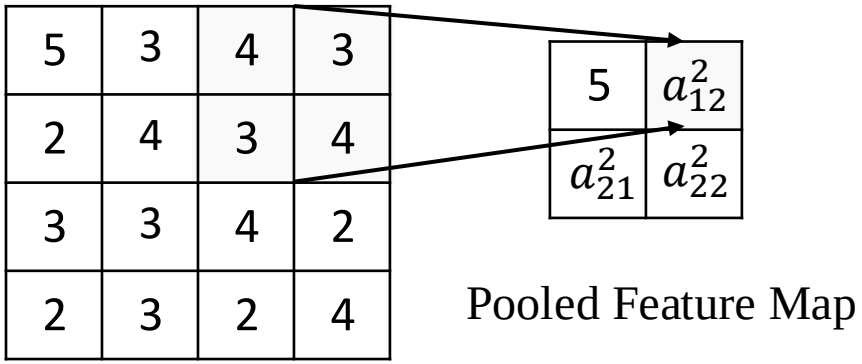
Convolutional Neural Network

- Classical Architecture - Example

Max Pooing

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

Input



Feature Map

Pooled Feature Map

Convolutional Neural Network

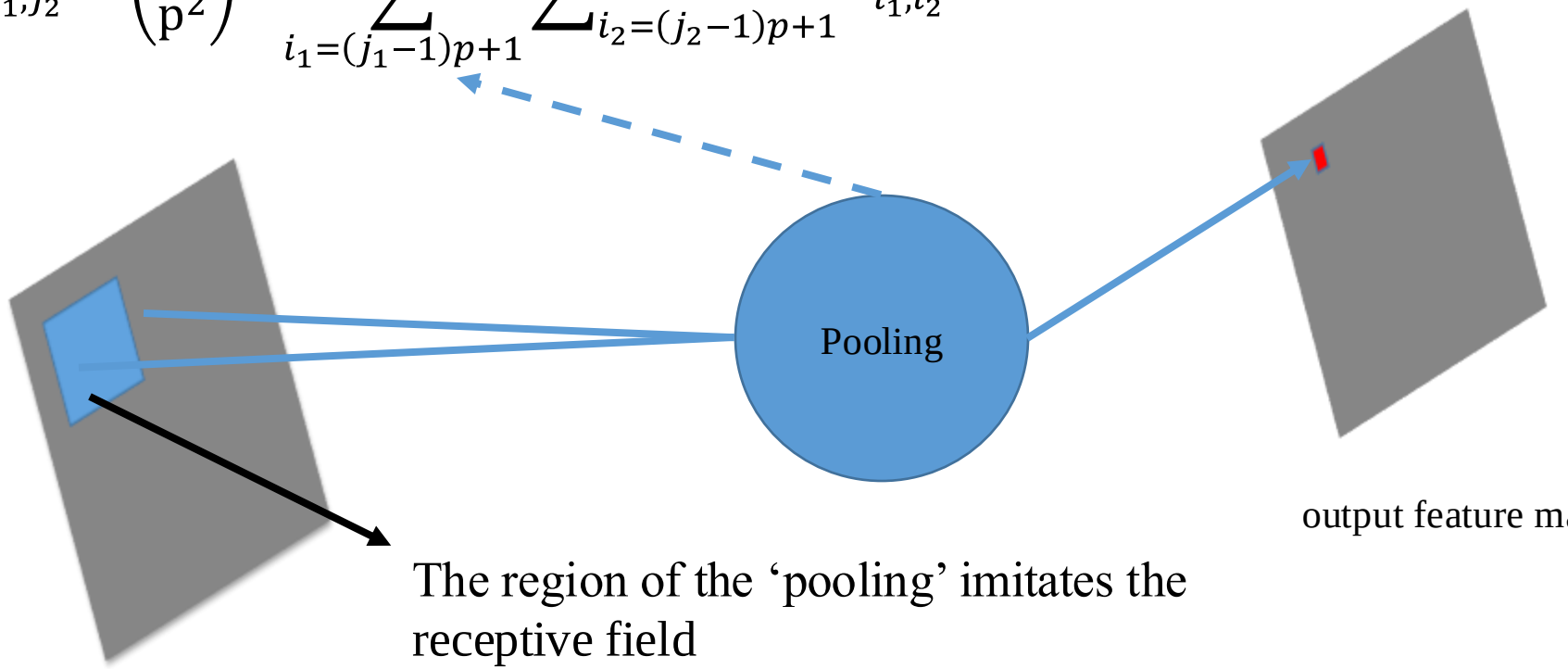
Gradient-Based Learning Algorithm

- Feedforward --Mean Pooling

$$z_{j_1,j_2}^{l+1} = \left(\frac{1}{\mathrm{p}^2}\right) * \sum_{i_1=(j_1-1)p+1}^{j_1 p} \sum_{i_2=(j_2-1)p+1}^{j_2 p} a_{i_1,i_2}^{l}$$

Vector form

- $\begin{cases} z^{l+1} = downSample(a^l) \\ a^{l+1} = z^{l+1} \end{cases}$



Convolutional Neural Network

Gradient-Based Learning Algorithm

Feedforward – vector form

- Convolution Layer

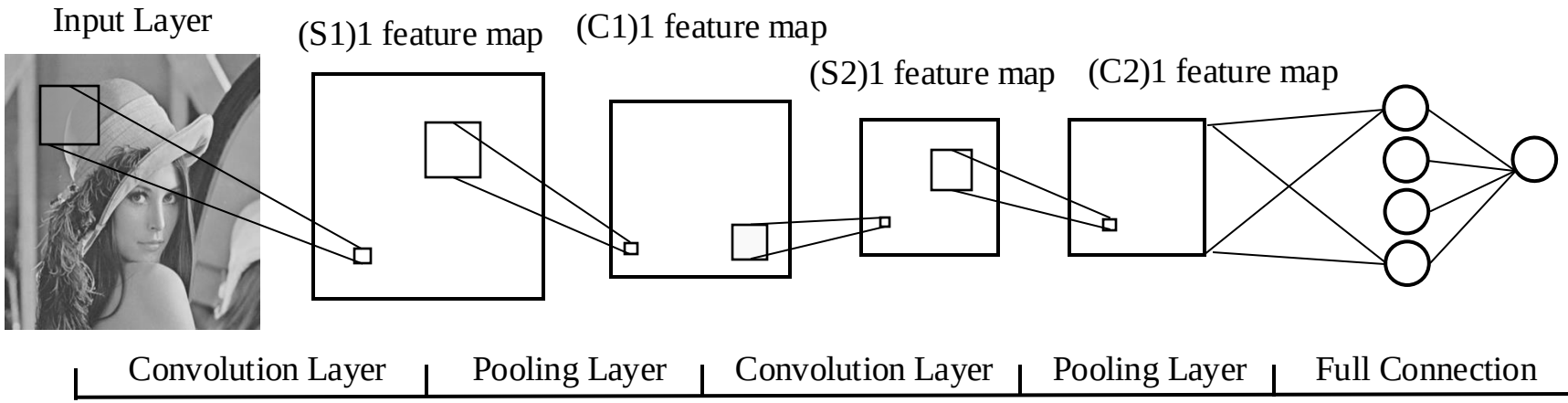
$$\begin{cases} z^{l+1} = W^l * a^l \\ a^{l+1} = f\left(z^{l+1}\right) \end{cases}$$

- Pooling Layer

$$\begin{cases} z^{l+1} = downSample(a^l) \\ a^{l+1} = z^{l+1} \end{cases}$$

- Full Connection Layer

$$\begin{cases} z^{l+1} = W^l a^l \\ a^{l+1} = f\left(z^{l+1}\right) \end{cases}$$



Convolutional Neural Network

Gradient-Based Learning Algorithm

- Backpropagation

Through BP :

$\delta^l = \frac{\partial J}{\partial z^l}$ (J is cost function of the convolution neural network)

$$\delta^l = \frac{\partial J}{\partial z^{l+1}} \frac{\partial z^{l+1}}{\partial z^l} = \delta^{l+1} w^l \frac{\partial a^l}{\partial z^l} = \delta^{l+1} w^l f'(z^l)$$

Which elements in the $(L+1)_{th}$ layer are related to the element z in the L_{th} layer ?

Convolutional Neural Network

Gradient-Based Learning Algorithm

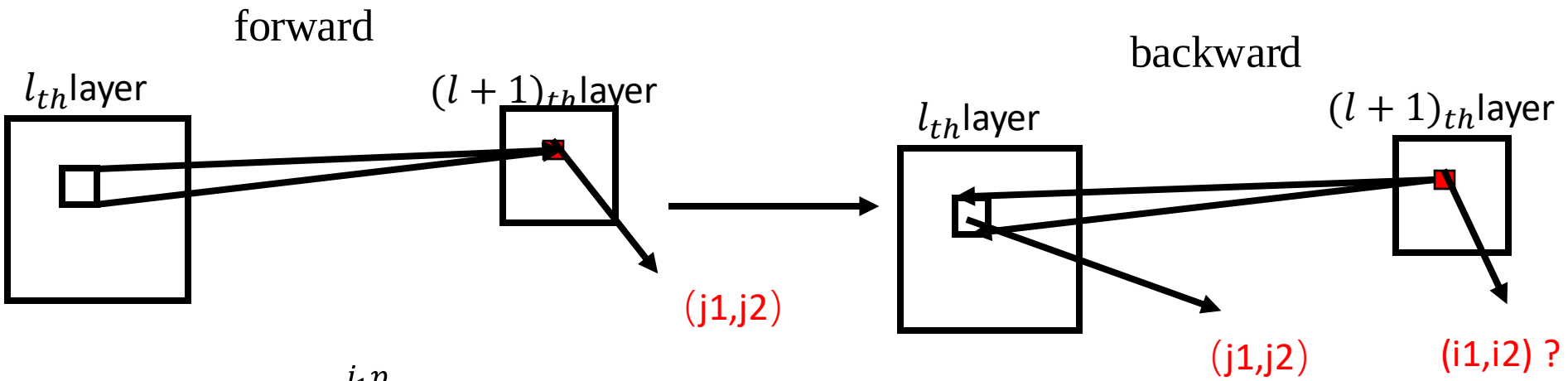
Backpropagation

- *NL-th* layer is output layer
  - $\delta^{NL} = \frac{\partial J}{\partial z^{NL}}$, where J is the cost function
- *l*-th layer is a Full Connection Layer
  - $$\begin{cases} \delta^{l} = \left((W^{l})^{T}\delta^{l+1}\right) \cdot f'(z^{l}) \\ \frac{\partial J}{\partial W^{l}} = \delta^{l+1}(a^{l})^{T} \end{cases}$$

Convolutional Neural Network

Gradient-Based Learning Algorithm

- Backpropagation
  - *l*-th layer is a **Convolution Layer,** *l*-th layer is pooling



$$z_{j_1,j_2}^{l+1} = \left(\frac{1}{\mathrm{p}^2}\right) * \sum_{i_1=(j_1-1)p+1}^{j_1 p} \sum_{i_2=(j_2-1)p+1}^{j_2 p} a_{i_1,i_2}^{l}$$

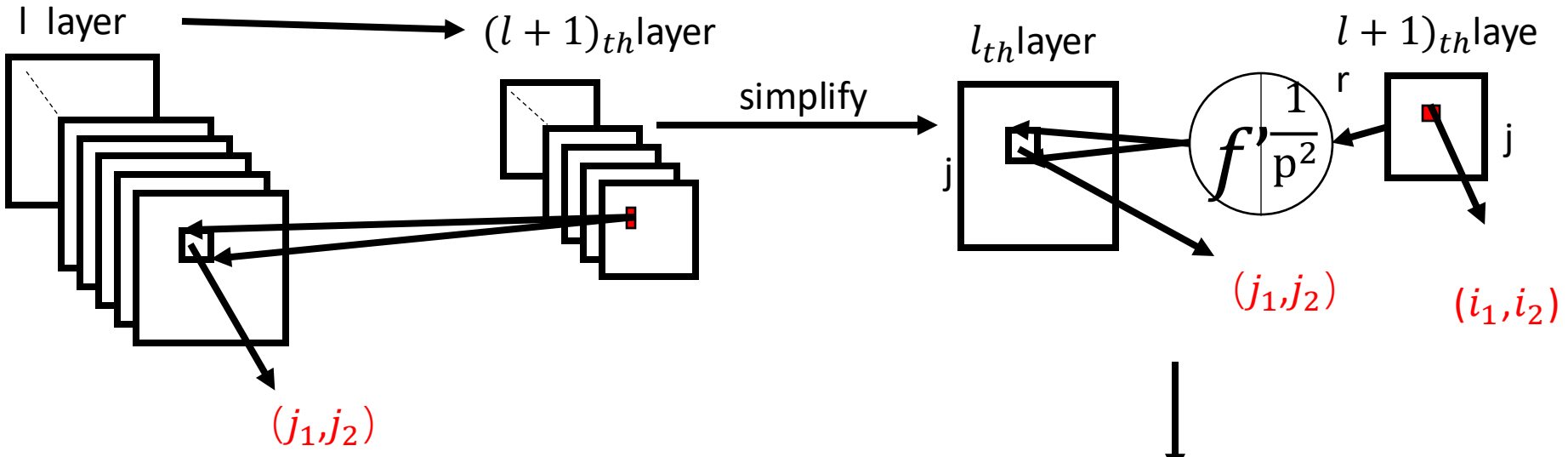
$i_1 = j_1/\mathrm{p} + 1$, $i_2 = j_2 / \mathrm{p} + 1$; ('/'denotes 'divided with no remainder')

$$a_{j_1,j_2}^{l+1} = z_{j_1,j_2}^{l+1} \longrightarrow \delta_{j_1,j_2}^{l} = \frac{\partial J}{\partial z_{i_1,i_2}^{l+1}} \frac{\partial z_{i_1,i_2}^{l+1}}{\partial z_{j_1,j_2}^{l}} = \delta_{i_1,i_2}^{l+1} * \frac{1}{\mathrm{p}^2} * f'(z_{j_1,j_2}^{l})$$

Convolutional Neural Network

Gradient-Based Learning Algorithm

- Backpropagation
  - *l*-th layer is a **Convolution Layer,** *l*-th layer is pooling



$$\delta^{l}_{j_1,j_2,j} = \frac{\partial J}{\partial z^{l+1}_{i_1,i_2,j}} \frac{\partial z^{l+1}_{i_1,i_2,j}}{\partial z^{l}_{j_1,j_2,j}} = \delta^{l+1}_{i_1,i_2,j} * \frac{1}{p^2} * f'(z^{l}_{j_1,j_2,j})$$

$$\delta^{l}_{j} = 1/(p^2)*\text{up}\,(\delta^{l+1}_{j})*f'(z^{l}_{j}) \text{ (matrix form)}$$

Convolutional Neural Network

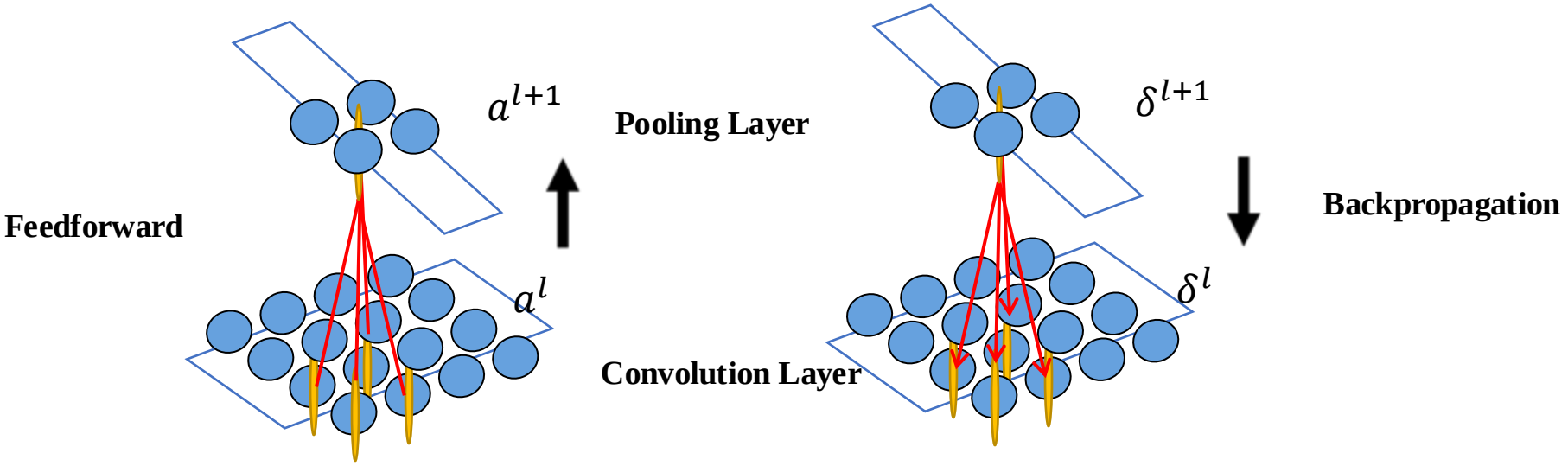
Gradient-Based Learning Algorithm

Backpropagation

- l-th layer is a **Convolution Layer**

  - $\delta^l = upSample(\delta^{l+1}) \cdot f'(z^l)$

$$\begin{cases} z^{l+1} = downSample(a^l) \\ a^{l+1} = z^{l+1} \end{cases}$$



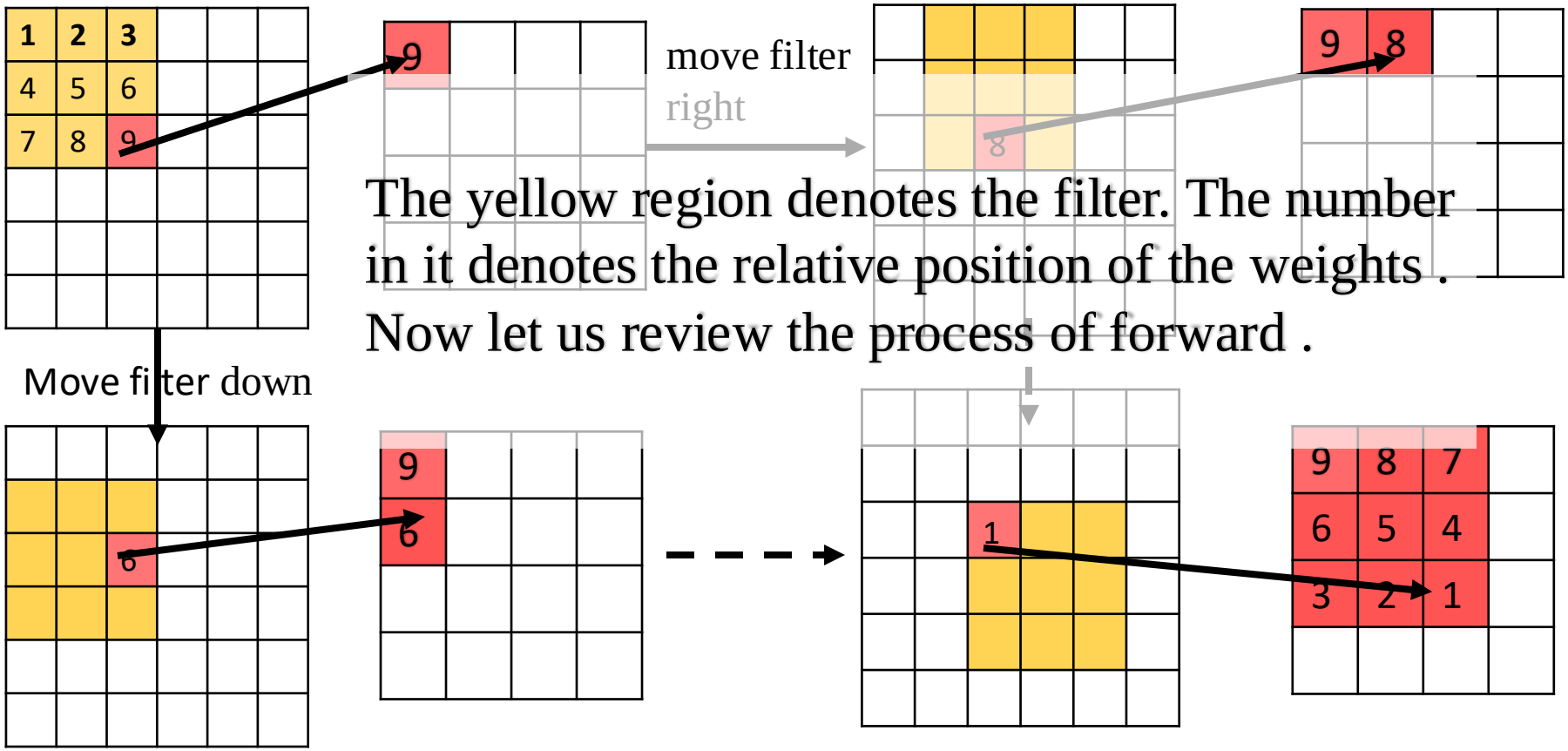
Convolution Layer is followed by a Pooling Layer

Convolutional Neural Network

Gradient-Based Learning Algorithm

Backpropagation

- *l*-th layer is a **Pooling Layer,** *l+1*-th layer is a Conv layer



Convolutional Neural Network

Gradient-Based Learning Algorithm

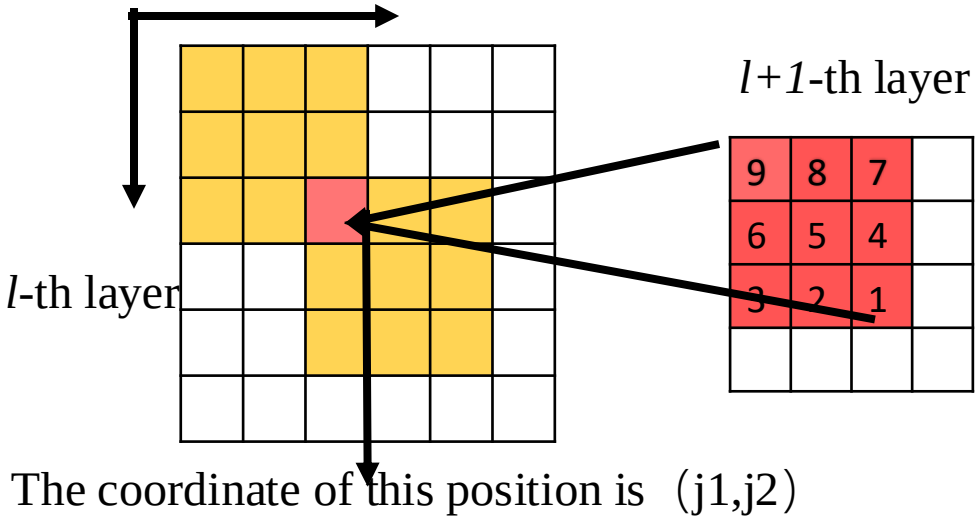
Backpropagation

- *l*-th layer is a **Pooling Layer,** *l+1*-th layer is a Conv layer

Hence: $\delta^{l}_{j_1,j_2} = \sum_{k_1=n_{k_1}}^{1} \sum_{k_2=n_{k_2}}^{1} \frac{\partial J}{\partial z^{l+1}_{j_1-k_1+1,j_2-k_2+1}} \frac{\partial z^{l+1}_{j_1-k_1+1,j_2-k_2+1}}{\partial z^{l}_{j_1,j_2}}$

$\delta^{l}_{j_1,j_2} = \sum_{k_1=n_{k_1}}^{1} \sum_{k_2=n_{k_2}}^{1} \delta^{l+1}_{j_1-k_1+1,j_2-k_2+1} w^{l}_{k_1,k_2} * f'(z^{l}_{j_1,j_2})$

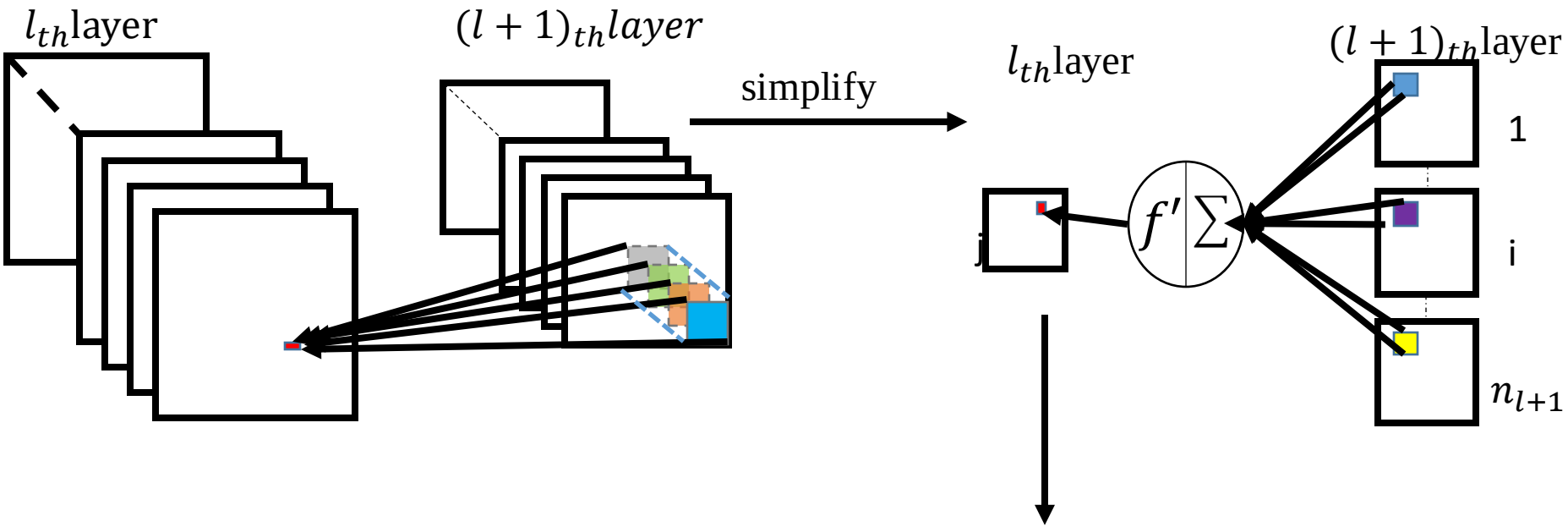
$f'(z^{l}_{j_1,j_2})=1$, so



Convolutional Neural Network

Gradient-Based Learning Algorithm

- Backpropagation
  - *l*-th layer is a **Pooling Layer,** *l+1*-th layer is a Conv layer



$$\delta^{l}_{j_1,j_2,j} = \sum_{k_1=n_{k_1}}^{1} \sum_{k_2=n_{k_2}}^{1} \delta^{l+1}_{j_1-k_1+1,j_2-k_2+1,j}\, w^{l}_{k_1,k_2,j,i}$$

$$\delta^{l}_{j} = \delta^{l+1}_{i} \times rot180(w^{l}_{j,i}) \text{ (matrix form)}$$

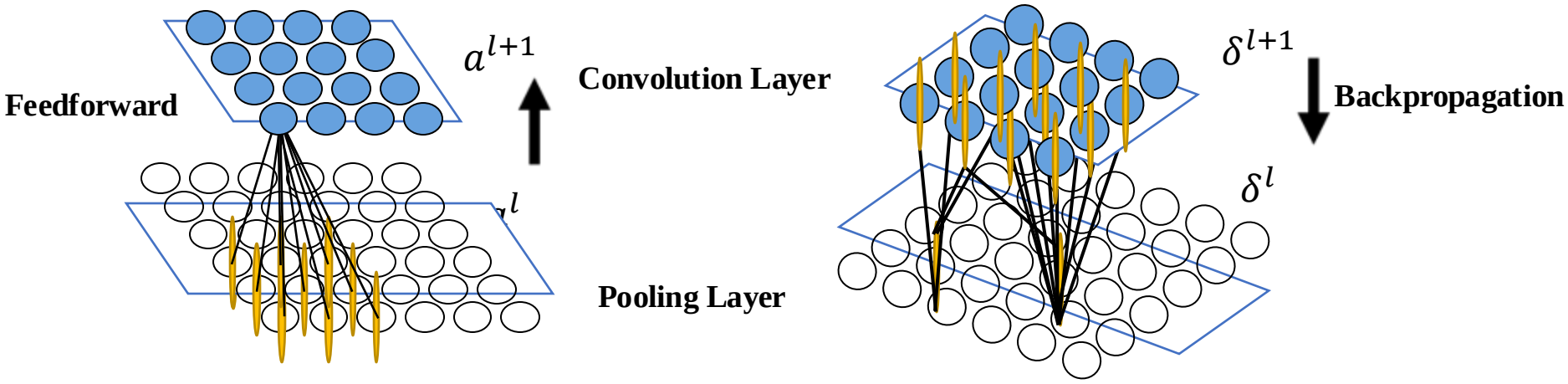
Convolutional Neural Network

Gradient-Based Learning Algorithm

Backpropagation

- l-th layer is a **Pooling Layer**
  - $\delta^l = (rot180(\delta^{l+1}) * w^l)$
  - $\frac{\partial J}{\partial W^l} = a^l * \delta^{l+1}$

$$\begin{cases} z^{l+1} = W^l * a^l \\ a^{l+1} = f(z^{l+1}) \end{cases}$$



Pooling Layer is followed by a Convolution Layer

Neural Networks

- Brief review
- Convolutional Neural Network

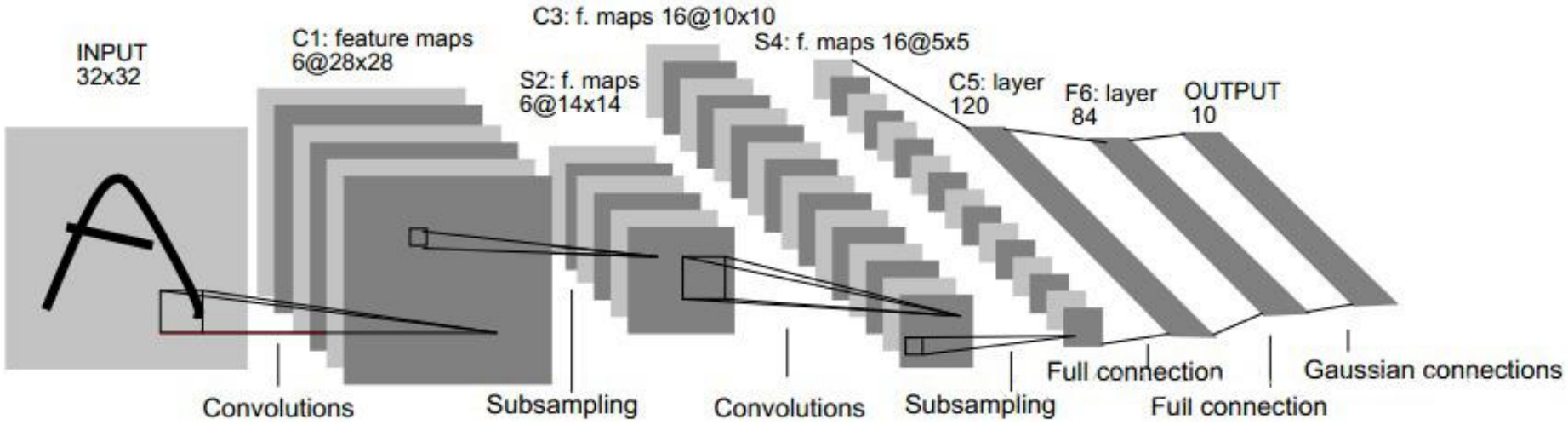
  Classical Architecture

  Gradient-Based Learning Algorithm

  Achievements

Convolutional Neural Network

- Achievements and Applications
  - LeNet – 1998 年



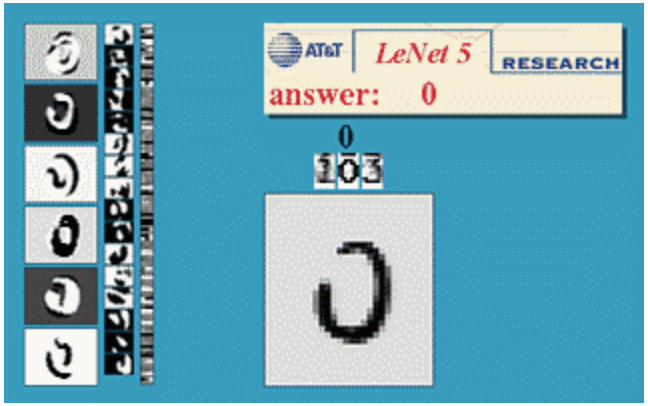
Kernel size	5*5 (conv)	2*2 (pooling)	5*5 (conv)	2*2 (pooling)

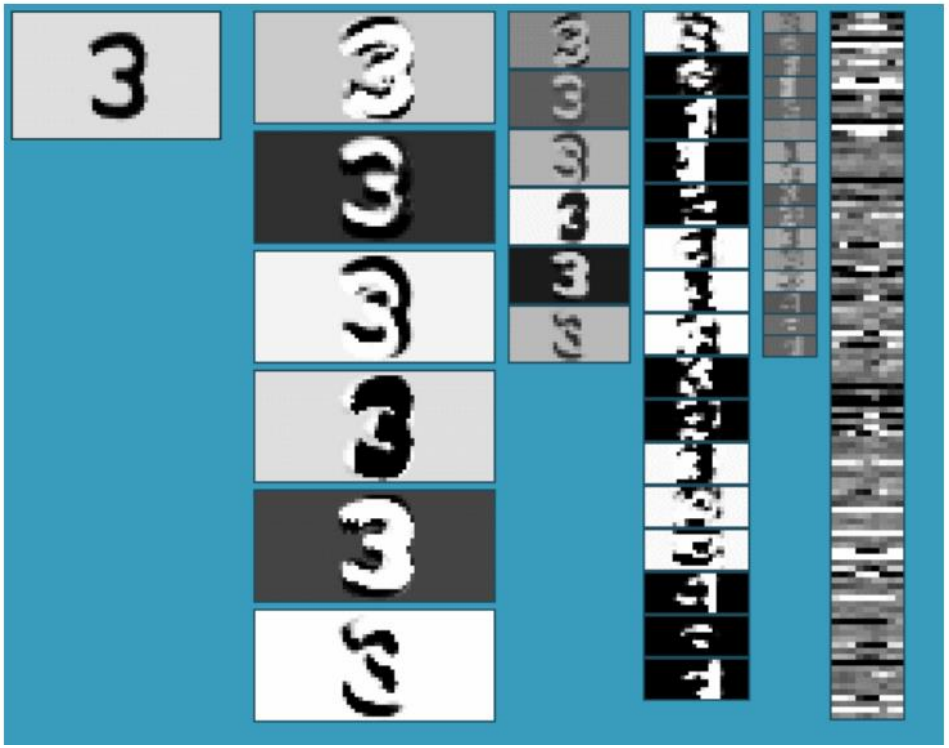
Convolutional Neural Network

- Achievements and Applications
  - Handwriting Recognition

filters → tanh → average-tanh → filters → tanh → average-tanh → filters → tanh

- LeNet LeCun

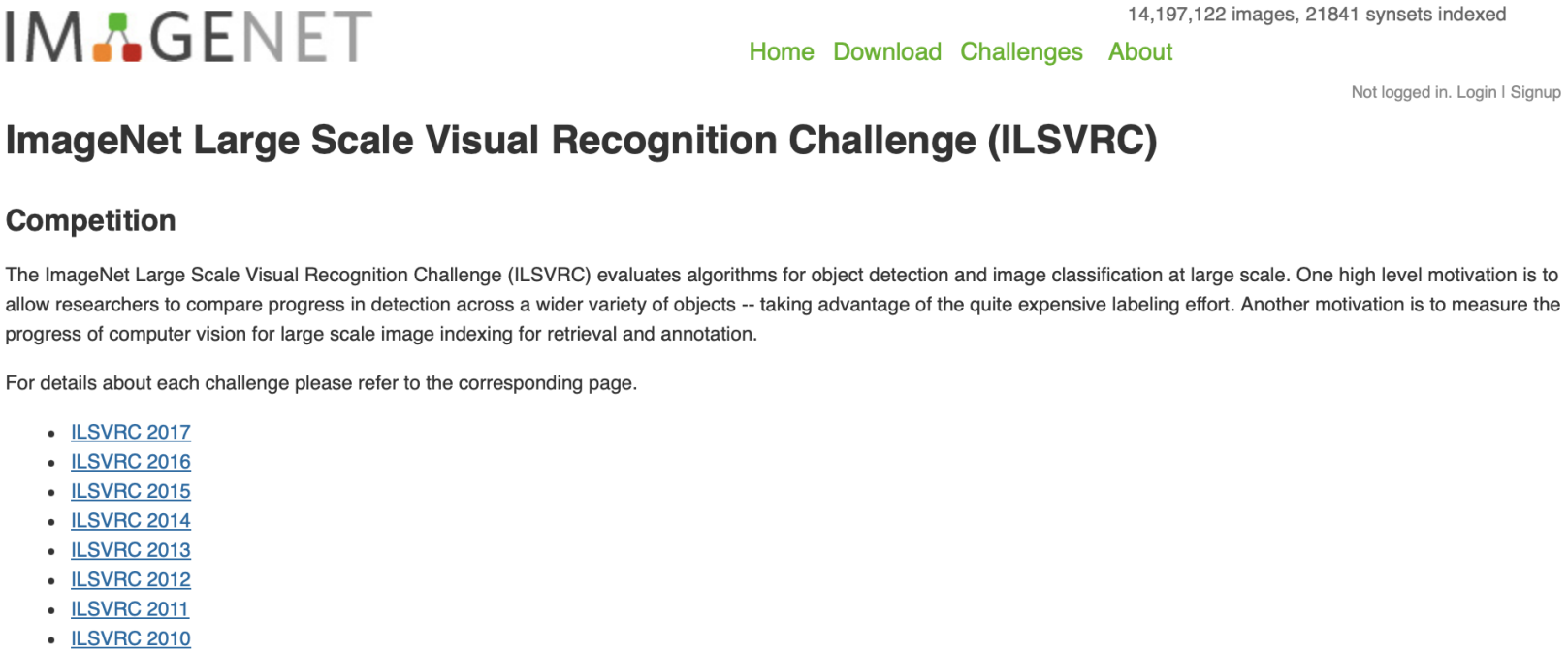




http://yann.lecun.com/exdb/lenet/index.html

Convolutional Neural Network

Achievements -- ILSVRC



ImageNet数据集是ILSVRC竞赛使用的是数据集，由斯坦福大学李飞飞教授主导，包含了超过1400万张全尺寸的有标记图片。ILSVRC比赛会每年从ImageNet数据集中抽出部分样本，以2012年为例，比赛的训练集包含1281167张图片，验证集包含50000张图片，测试集为100000张图片。

Convolutional Neural Network

Achievements -- ILSVRC

- 图像分类与目标定位

  图像分类的任务是要判断图片中物体在1000个分类中所属的类别，主要采用top-5错误率的评估方式，即对于每张图给出5次猜测结果，只要5次中有一次命中真实类别就算正确分类，最后统计没有命中的错误率。

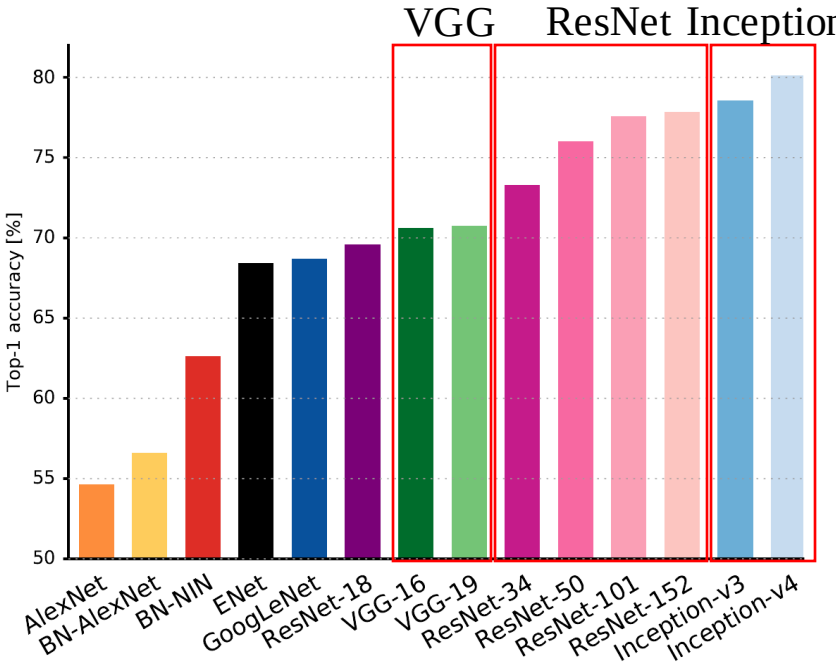
- 目标检测

  目标检测是在定位的基础上更进一步，在图片中同时检测并定位多个类别的物体。具体来说，是要在每一张测试图片中找到属于200个类别中的所有物体，如人、勺子、水杯等。评判方式是看模型在每一个单独类别中的识别准确率，在多数类别中都获得最高准确率的队伍获胜。

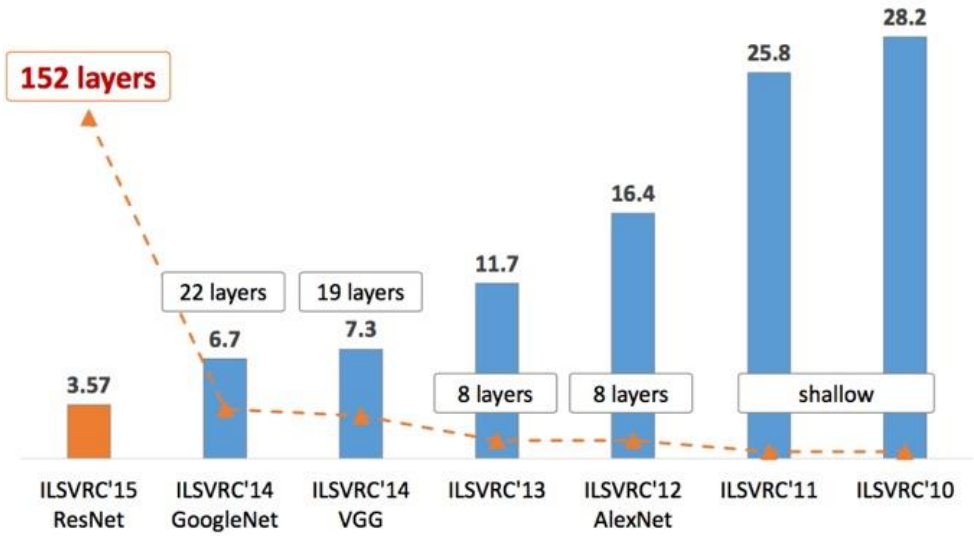
- 视频目标检测
- 场景分类

Convolutional Neural Network

- Achievements -- ImageNet



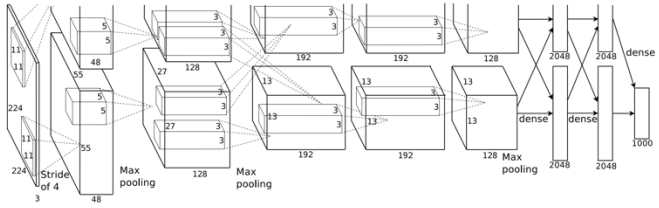
Single-crop Top-1 validation accuracies on ImageNet



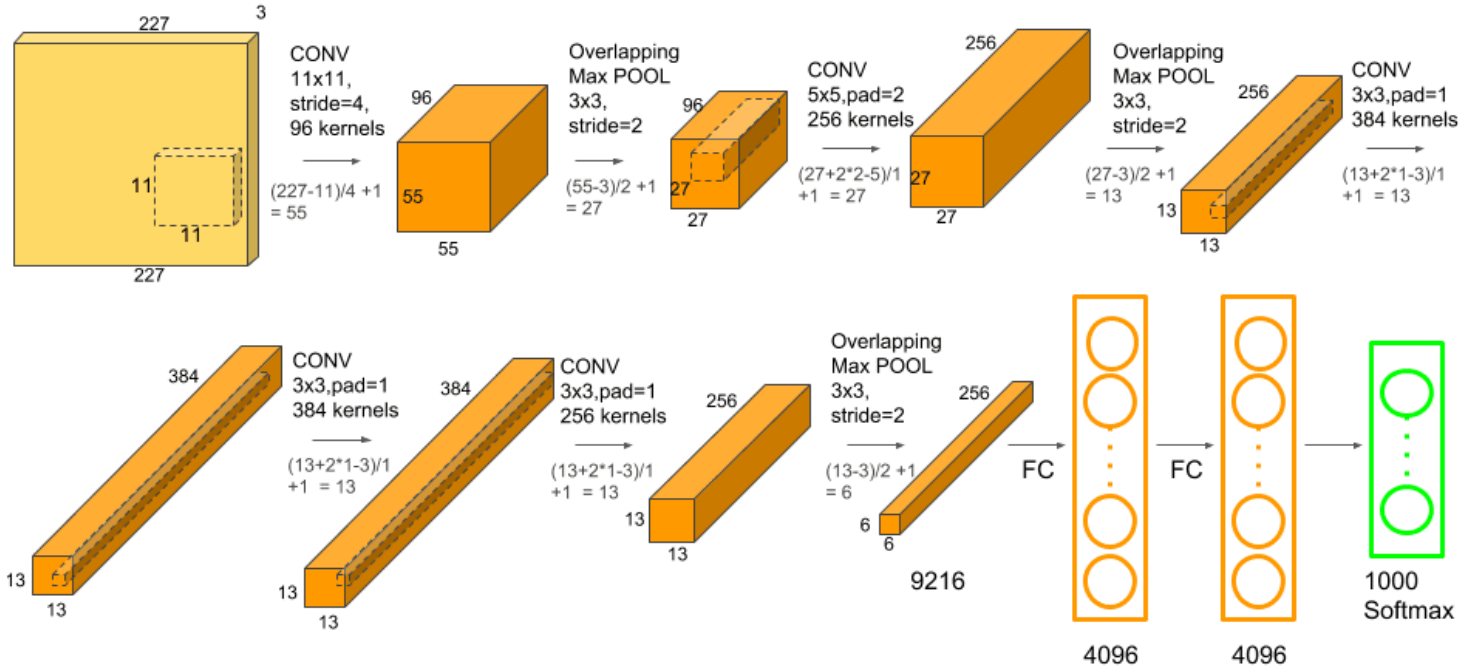
Reported Top-5 error on ImageNet

Convolutional Neural Network

- Achievements and Applications
  - AlexNet



ı of responsibilities
runs the layer-parts
8-dimensional, and
6–64,896–43,264–

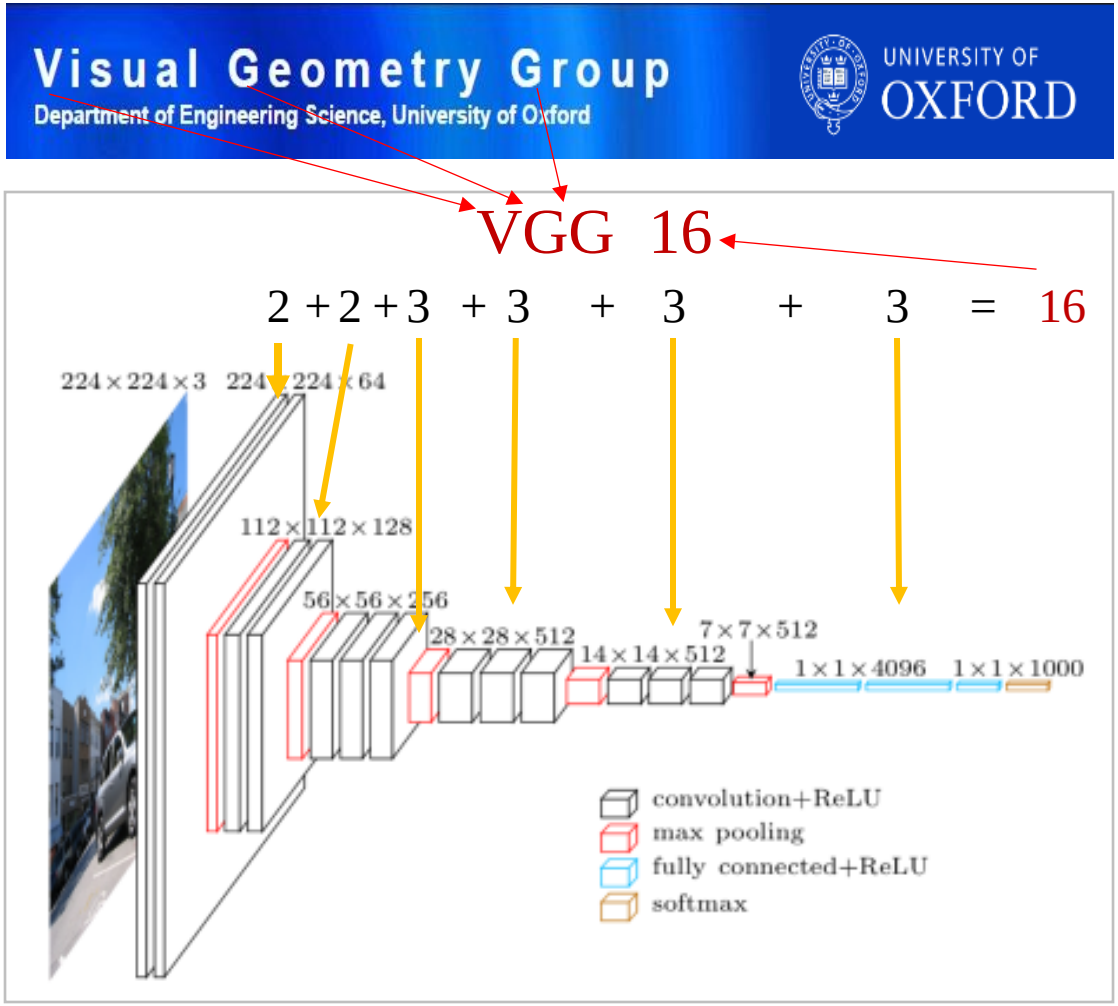


AlexNet是2012年ImageNet竞赛冠军获得者Hinton和他的学生Alex Krizhevsky设计的。

Convolutional Neural Network

Achievements

VGG Networks
- VGG11
- VGG13
- VGG16
- VGG19



VGG模型是2014年ILSVRC竞赛的第二名，第一名是GoogLeNet。

Convolutional Neural Network

Achievements

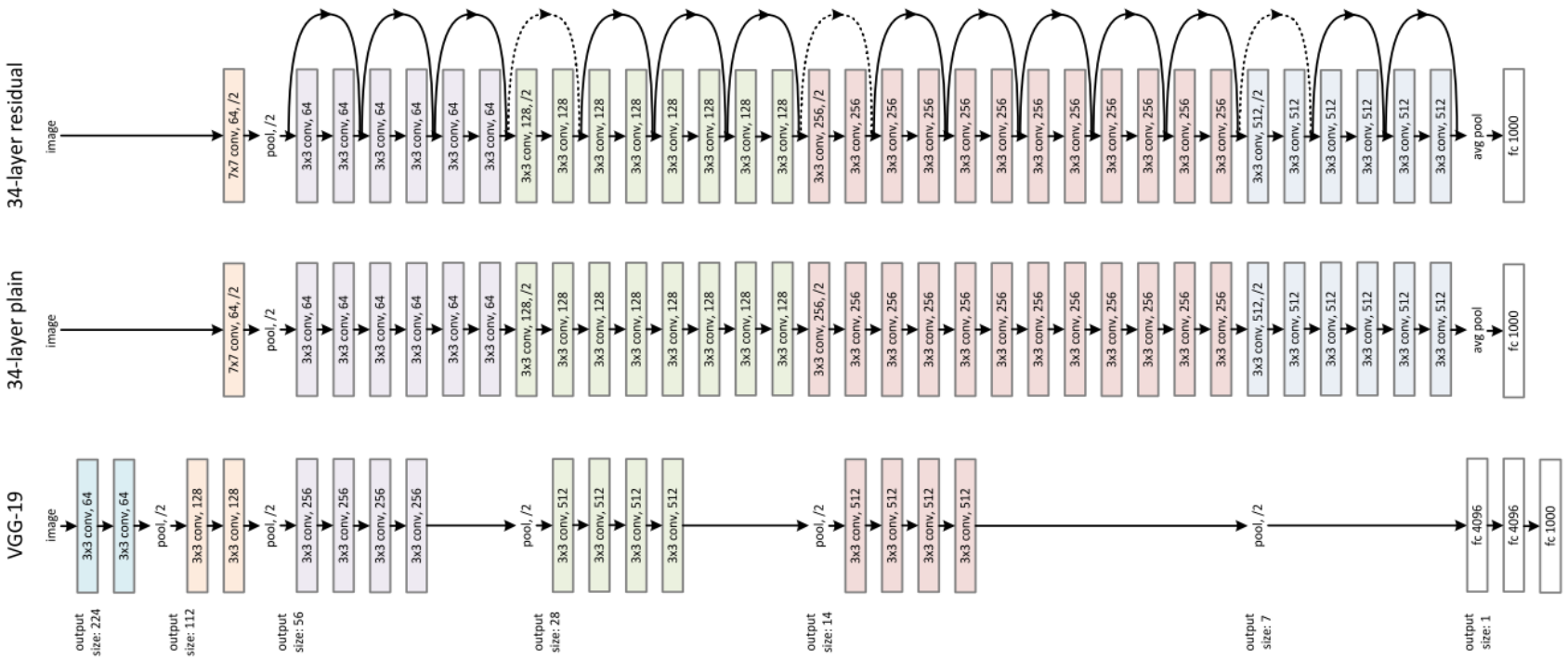
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	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					

VGG16 - Features:

- 11-19 layers
- Same input, 3×224×224
- 3 fully-connected layers
- Softmax output
- 5 stages, maxpool in between
- (most)3×3 kernels
- Increasing kernel number

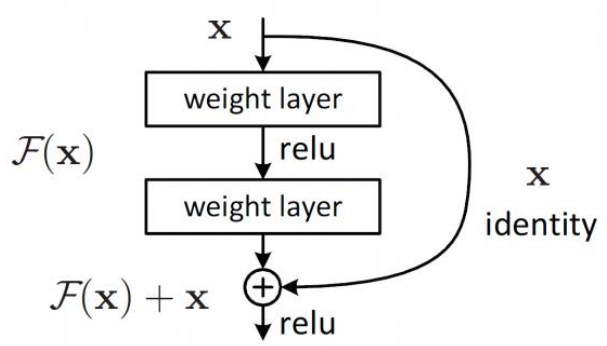
Convolutional Neural Network

- Achievements and Applications
  - ResNet --2015



Convolutional Neural Network

- Achievements and Applications
  - ResNet

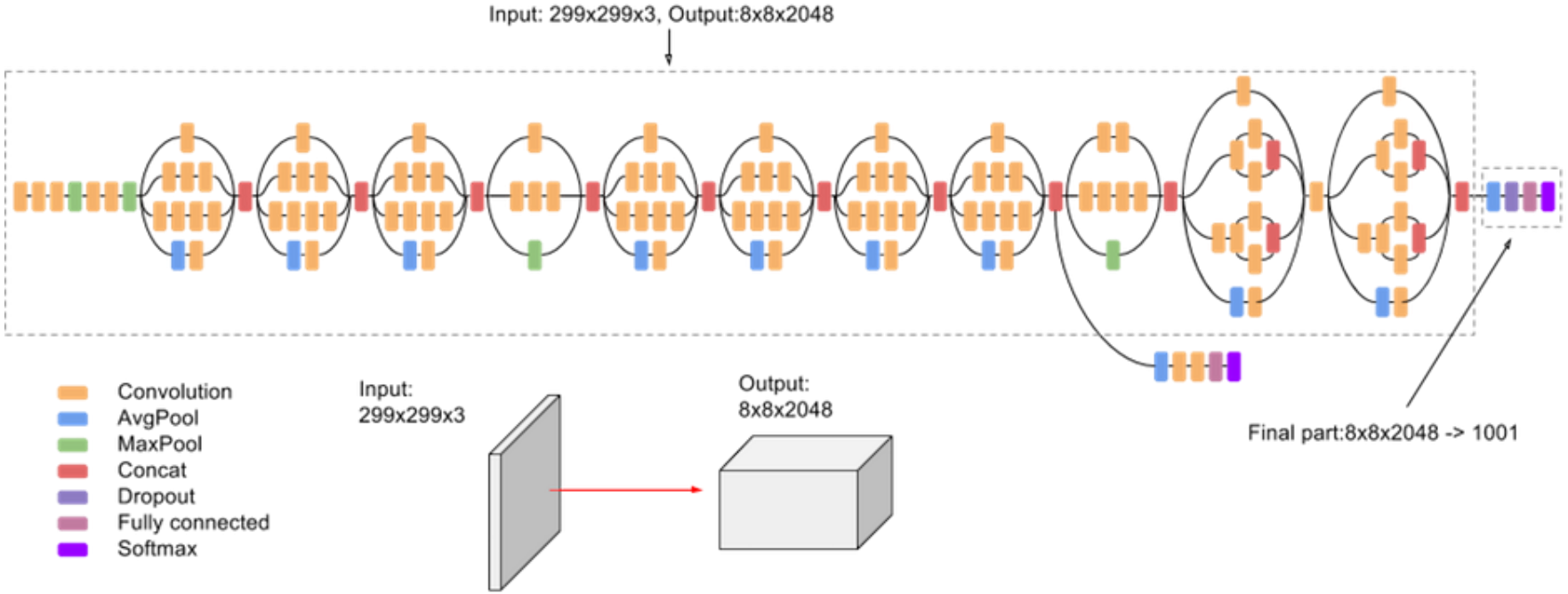


layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Convolutional Neural Network

Achievements and Applications

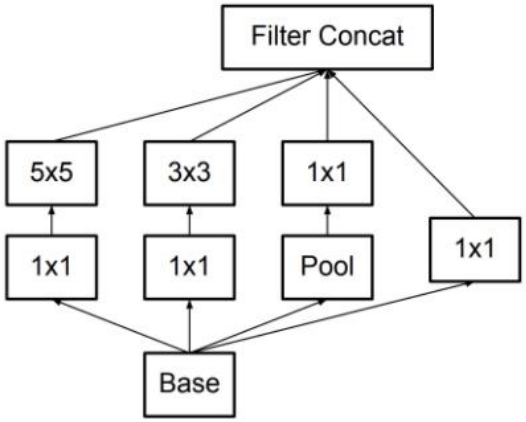
➢ Inception-V3

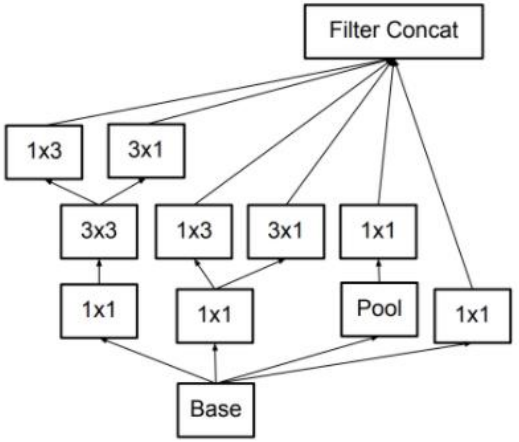


•42层深，但计算成本只比inception v1高2.5倍左右，而且比VGGNet高效得多.
•用卷积和池化并行的方法降低 Inception 模块的大小

Convolutional Neural Network

- Achievements and Applications
  - Inception V3





V1：分支卷积
V3：改为小卷积和非对称卷积

Convolutional Neural Network

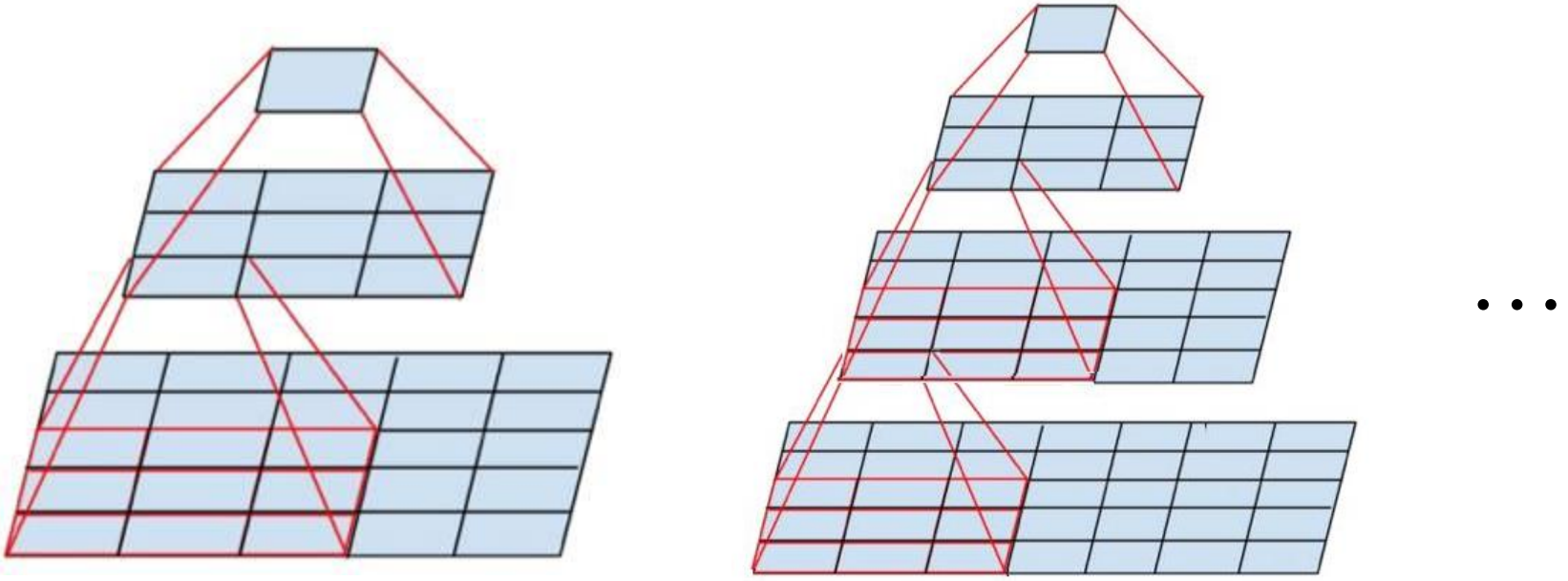
- Thinking--Why that size of kernels?

Network	Kernels Used	Deepth
Lenet	(5,5)	4
Alexnet	(11,11), (5,5), (3,3)	8
VGGs	(3,3)	11-19
ResNet	(3,3), (1,1)	
Inception V3	(1,3),(3,1),(3,3),(1,1)	

Convolutional Neural Network

- Thinking– why 3×3 kernels?

a.) 3×3 kernels can simulate larger kernels like 5×5 or 7×7

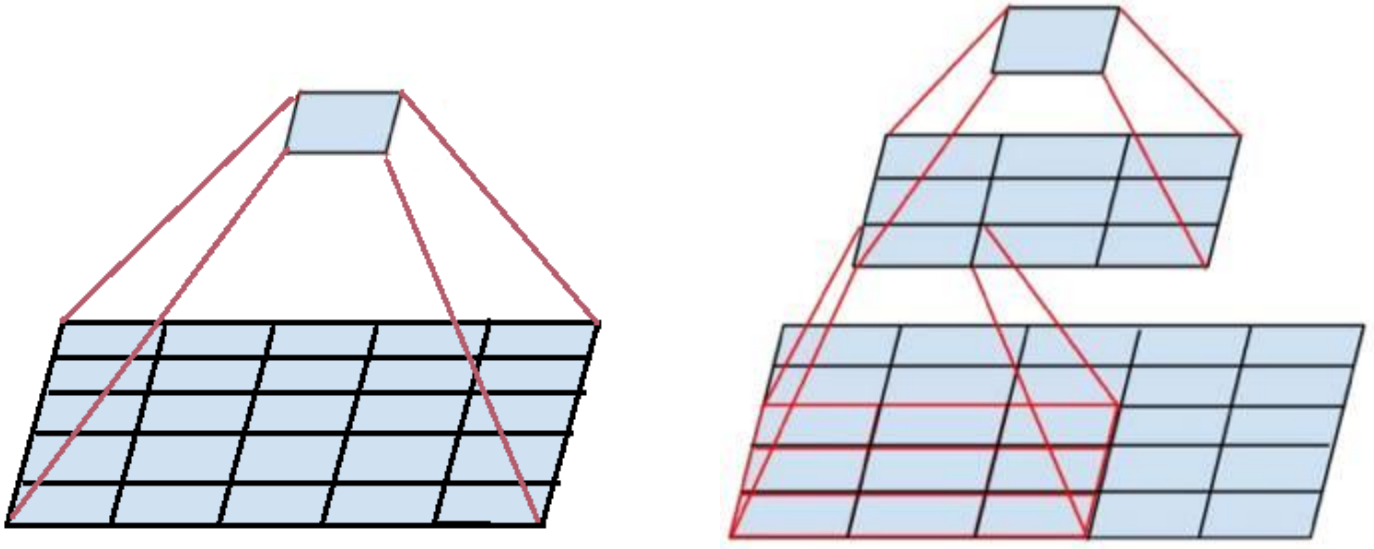


Convolutional Neural Network

- Thinking– why 3×3 kernels?

  a.) 3×3 kernels can simulate larger kernels like 5×5 or 7×7

  b.) by doing so, more nonlinearity is involved, thus more expressive ability



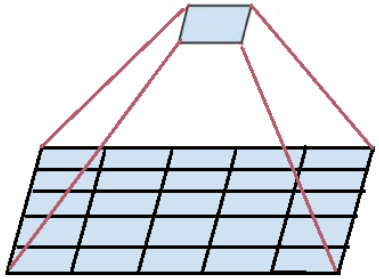
Convolutional Neural Network

Thinking– why 3×3 kernels?

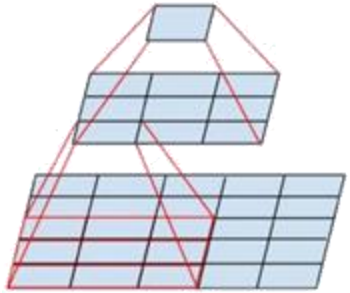
a.) 3×3 kernels can simulate larger kernels like 5×5 or 7×7

b.) by doing so, more nonlinearity is involved, thus more expressive ability

c.) 3×3 kernels can save a lot of parameters



# of weights: 5×5 = 25



# of weights: 2×(3×3) = 18

$\frac{18}{25} = 72\%$

General case:

5×5 ×input channel ×output channel

(3×3 ×input channel ×hidden channel) +(3×3 ×hidden channel ×output channel)