

# EE6222: Machine Vision

Lecturer: **Jiang Xudong, Fellow of IEEE**

 : [exdjiang@ntu.edu.sg](mailto:exdjiang@ntu.edu.sg)

 : 67905018

 : S1-B1c-105

<https://personal.ntu.edu.sg/exdjiang/>

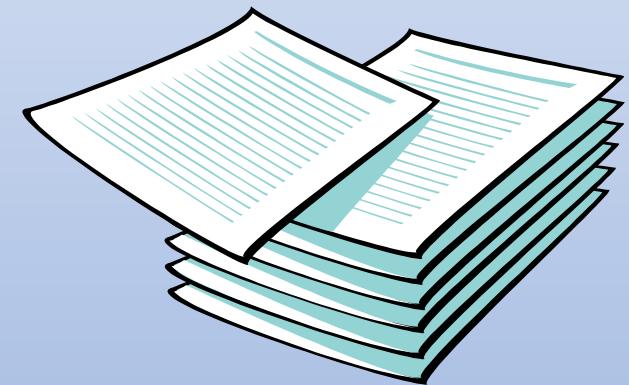
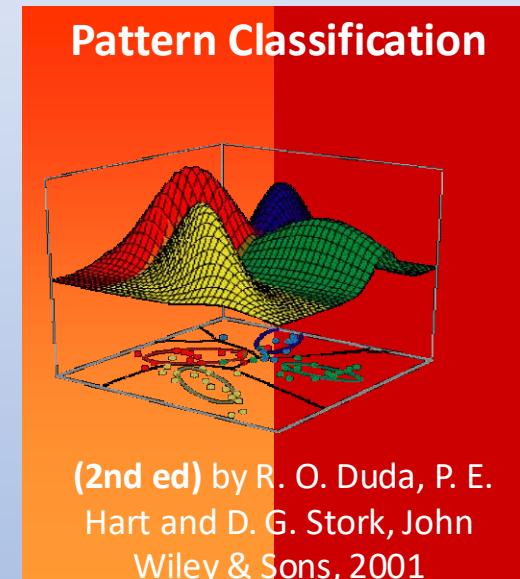
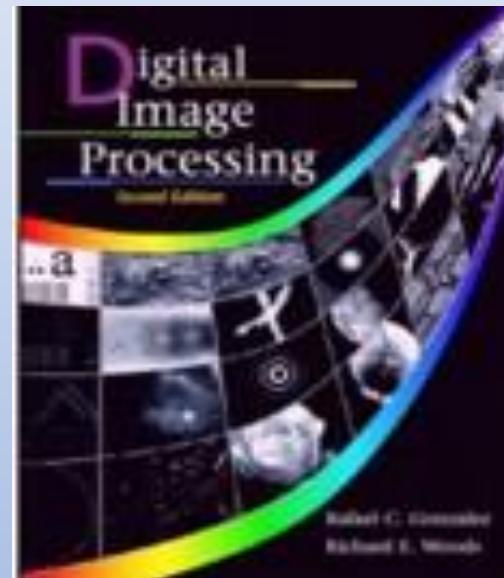
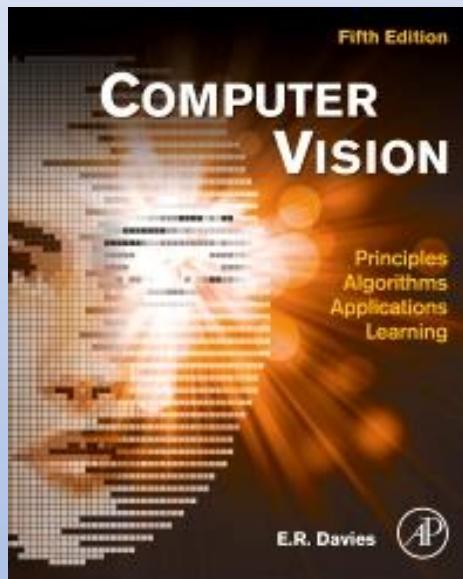
# Course Content

1. Image Fundamentals and Human Perception.
2. LSI Systems and Transforms
3. Image Denoising and Enhancement
4. Morphological Image Processing
5. Intuitive Understanding of Object Recognition: from Matching to Classification
6. MAP Decision and Classifiers
7. Statistical Estimation and Machine Learning
8. Handcrafted Feature Generation and Feature Selection
9. Visual Data Dimensionality Reduction as Feature Extraction
10. Neural Networks and Deep Machine Learning: from MLP to CNN
11. Deep Learning: from CNN to Transformer
12. Video Analysis
13. Video Recognition
14. Three-dimensional Machine Perception
15. Three-dimensional Machine Vision

Topics 12, 13, 14, 15 will be lectured by Dr. Cheng Jun from week 10 to week 13.

# Textbooks, Assessments & Prerequisites

- Davies E. R., Computer Vision: Principles, Algorithms, Applications, Learning, Elsevier Science, Academic Press, 2017.
- Gonzalez, R. C., Woods, R. E., *Digital Image Processing*, Prentice Hall
- R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, Wiley Inter-science



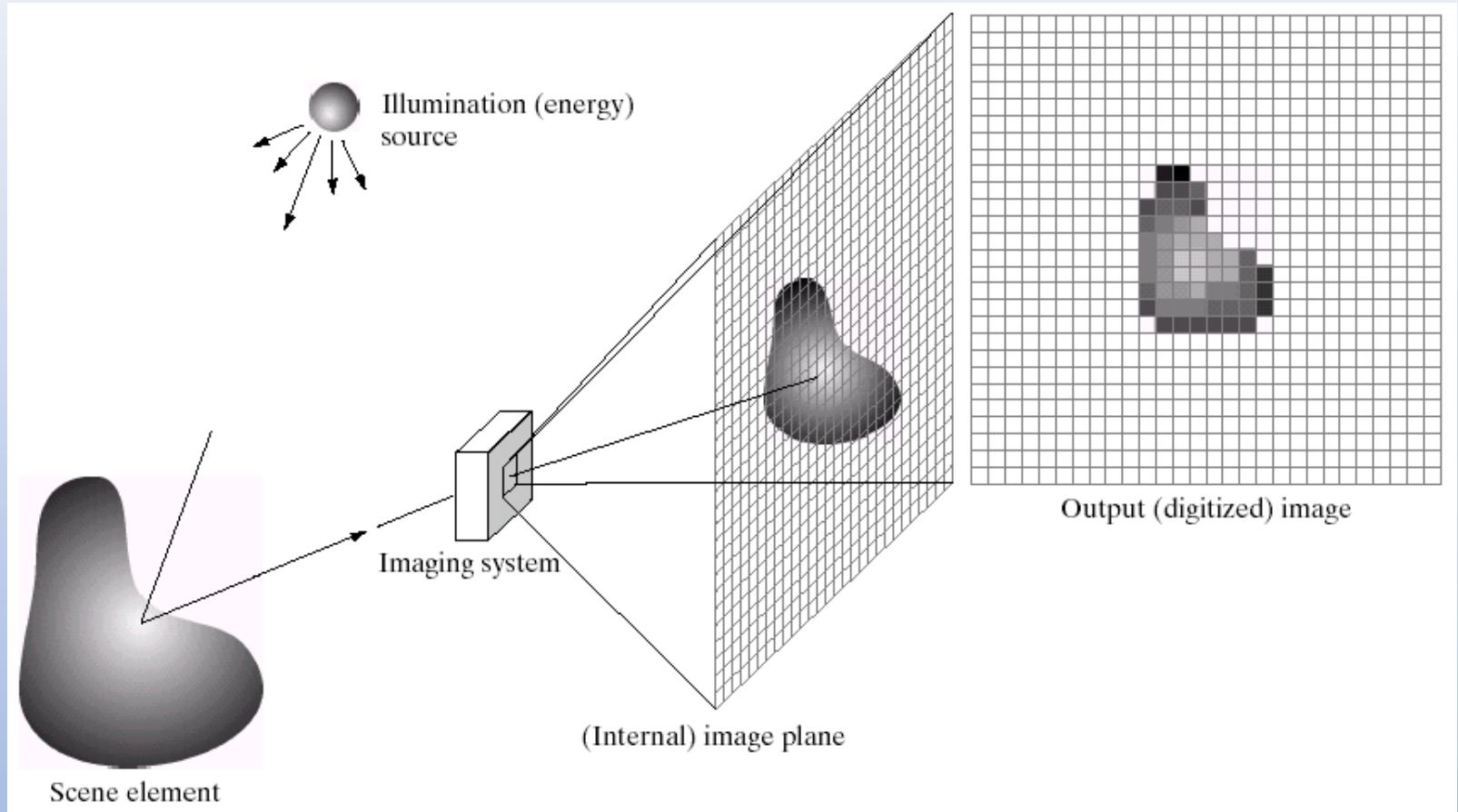
- **Assessment scheme:** Final Examination 60% and CA 40% by one quiz (10%, 30 minutes in class at week 7) and two take-home assignments (15 % each)
- **Prerequisites:** Probability Theory, Linear Algebra, Signals & Systems

# 1. Image Fundamentals & human Perception – Outline

- Image Formation
- Human Perception of Image and Color image
- Digital Image Representation
- **Image Histogram**
- Image Processing and Its Application

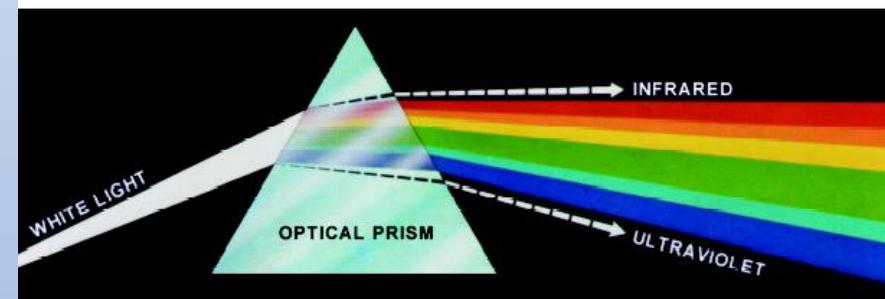
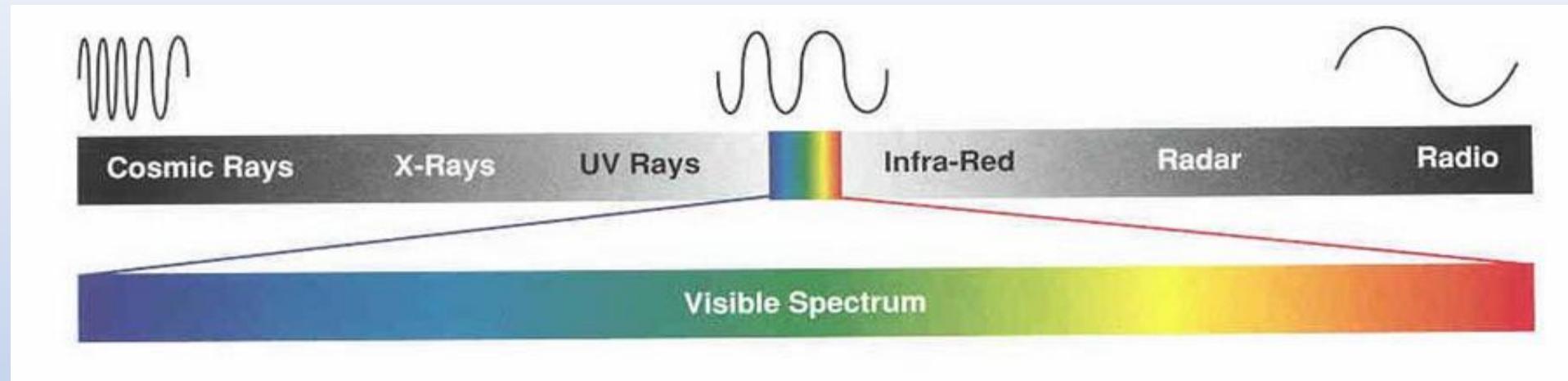
# 1. Image Fundamentals—image formation

What is an image?



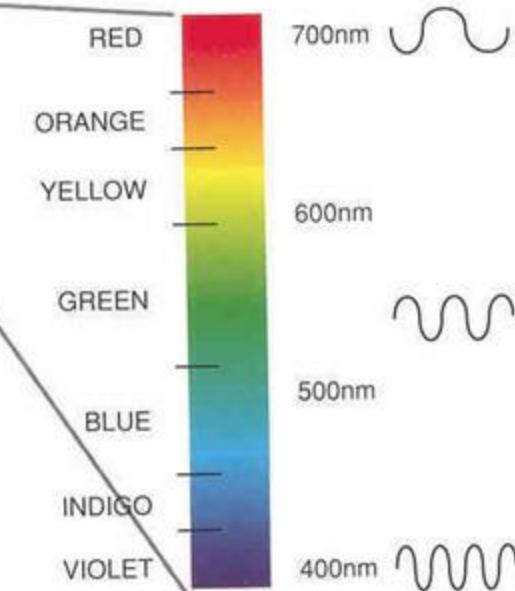
- Physically, an image is a two-dimensional (2-D) projection of a three-dimensional (3-D) scene, a visual representation, a vivid or graphic description of an object or scene.

# 1. Image Fundamentals—color image



The Visible Light

Why can we see color?



# 1. Image Fundamentals—Luminance and brightness

Light received/reflected from a point of an object is

$$I(\lambda) = \rho(\lambda)L(\lambda)$$

where  $\rho(\lambda)$  is the reflectivity of object,  $L(\lambda)$  is the spectral energy distribution of the light source,  $\lambda$  is the wavelength in the visible spectrum, 350nm to 780 nm.

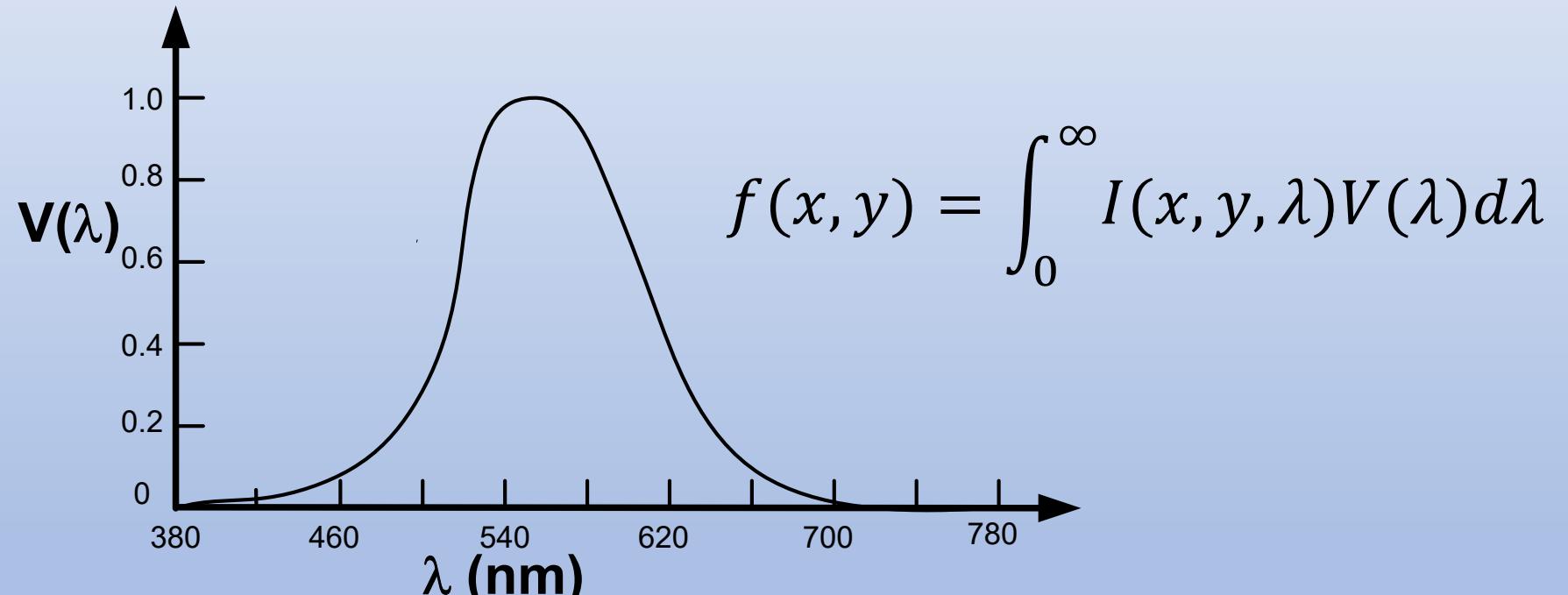
Luminance or intensity of a spatially distributed object with light distribution  $I(x,y,\lambda)$  received by a sensor is defined as

$$f(x, y) = \int_0^{\infty} I(x, y, \lambda)V(\lambda)d\lambda$$

where  $V(\lambda)$  is the relative spectral sensitivity function of the visual system

# 1. Image Fundamentals—Luminance and brightness

$V(\lambda)$  is a bell-shaped curve

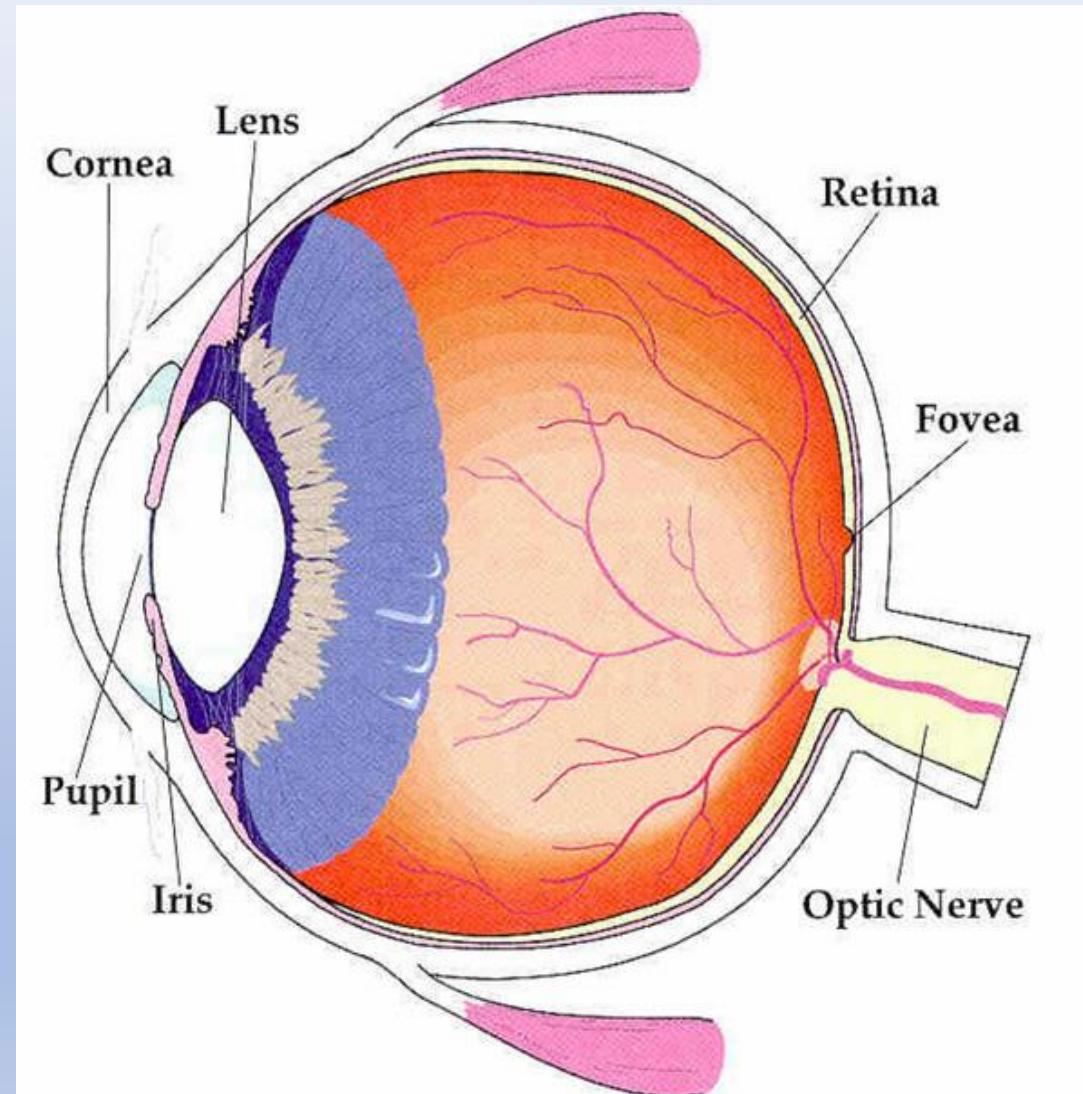


$V(\lambda)$  is some kind of frequency response

# 1. Image Fundamentals—Human Perception

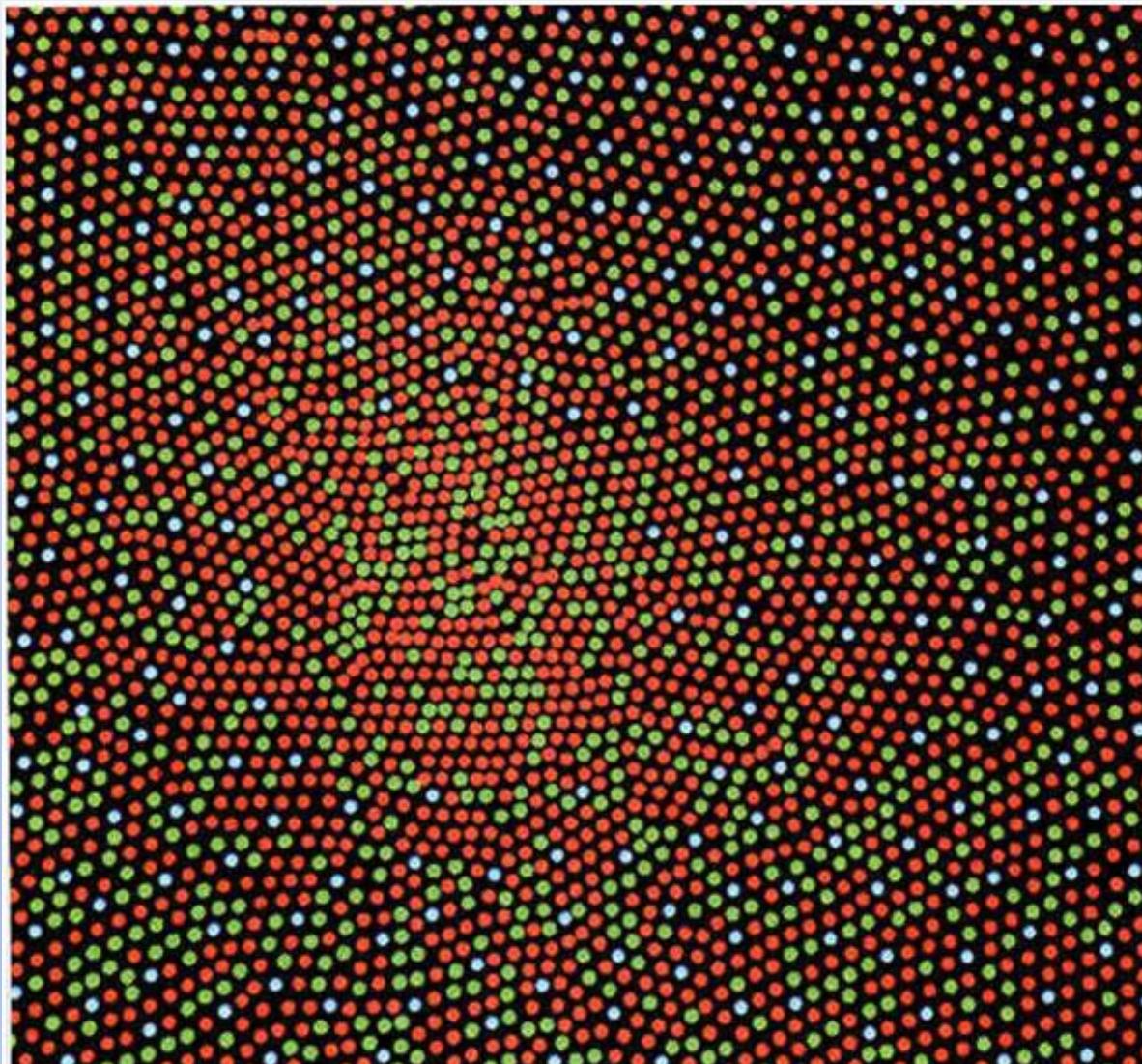
## The Human eye

- Human eye has many cells shaped like cones to perceive light.
- There are three different type of cones with different spectral sensitivity function  $V(\lambda)$ , which help perceive colour.



# 1. Image Fundamentals—Human Perception

Relative proportions of  
L (red),  
M (green), and  
S (blue) cones  
in the human retina.



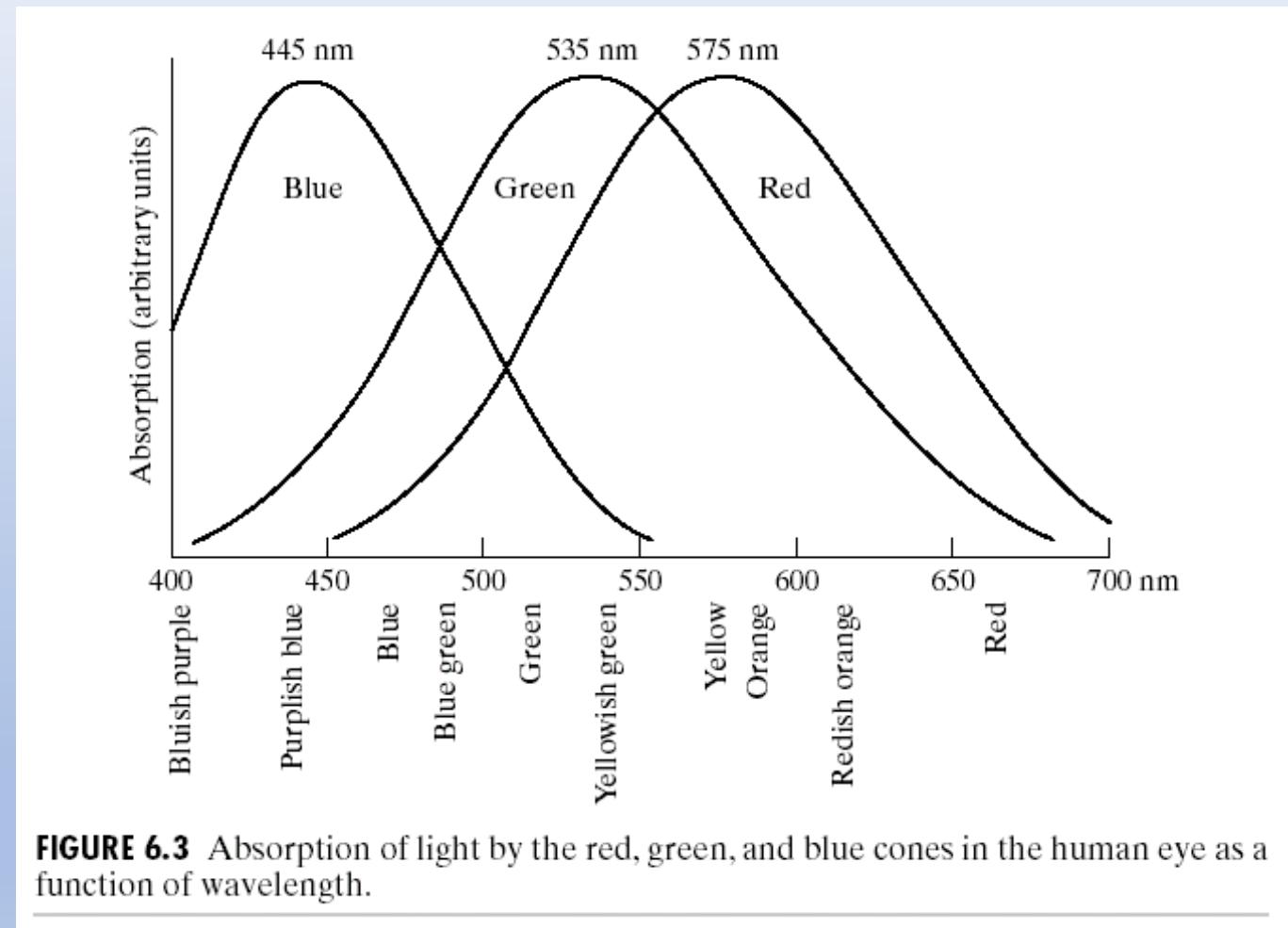
# 1. Image Fundamentals—Human Perception

- Color is a function of wavelength (frequency)
- Color Primaries: Red(R), Green(G), Blue(B)

$$f_1(x,y) = \int_0^{\infty} I(x,y,\lambda) V_1(\lambda) d\lambda$$

$$f_2(x,y) = \int_0^{\infty} I(x,y,\lambda) V_2(\lambda) d\lambda$$

$$f_3(x,y) = \int_0^{\infty} I(x,y,\lambda) V_3(\lambda) d\lambda$$



**FIGURE 6.3** Absorption of light by the red, green, and blue cones in the human eye as a function of wavelength.

# 1. Image Fundamentals—color image

Three values per sample (pixel) are required for a color image.



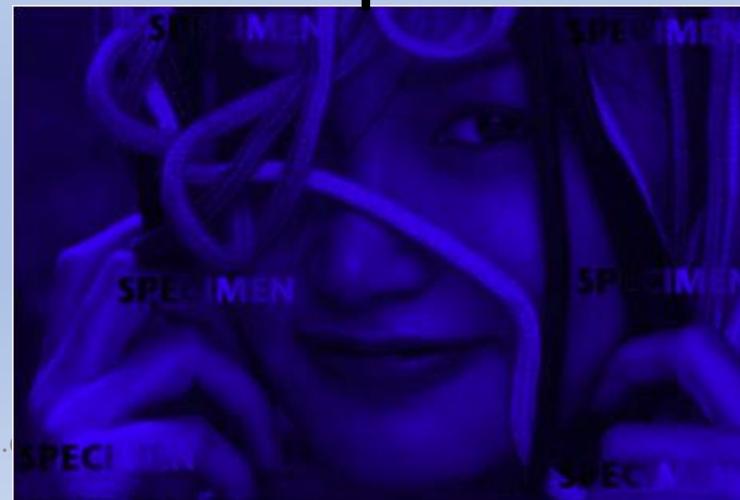
=



+

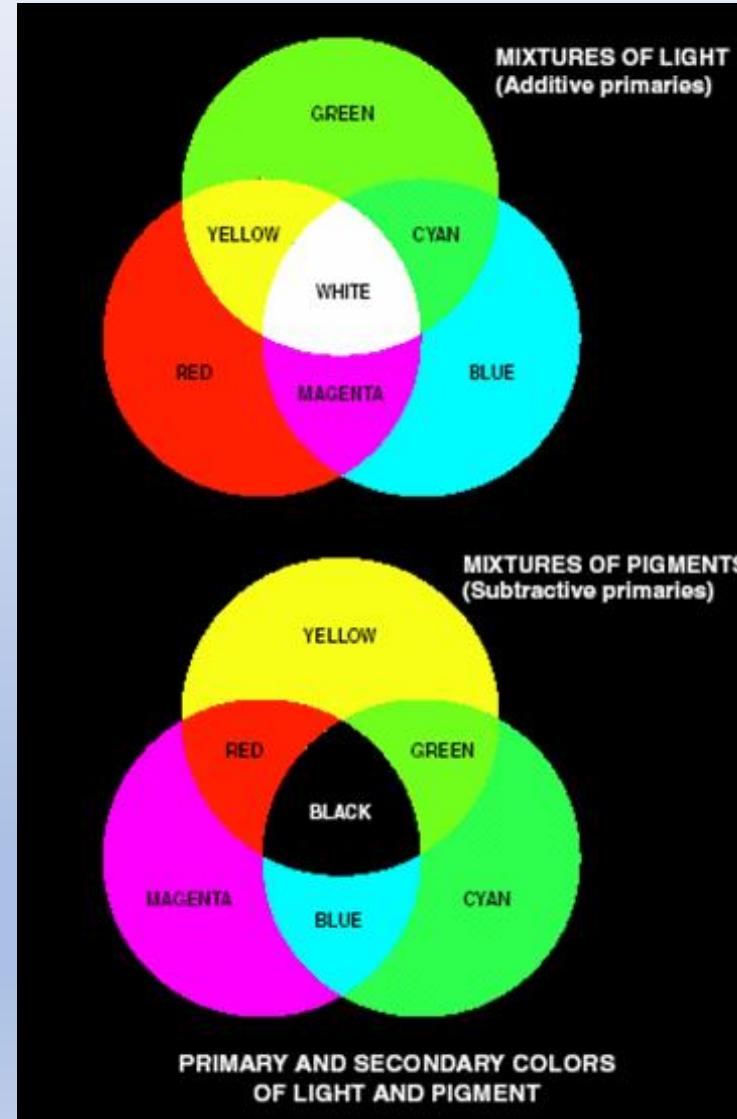


+



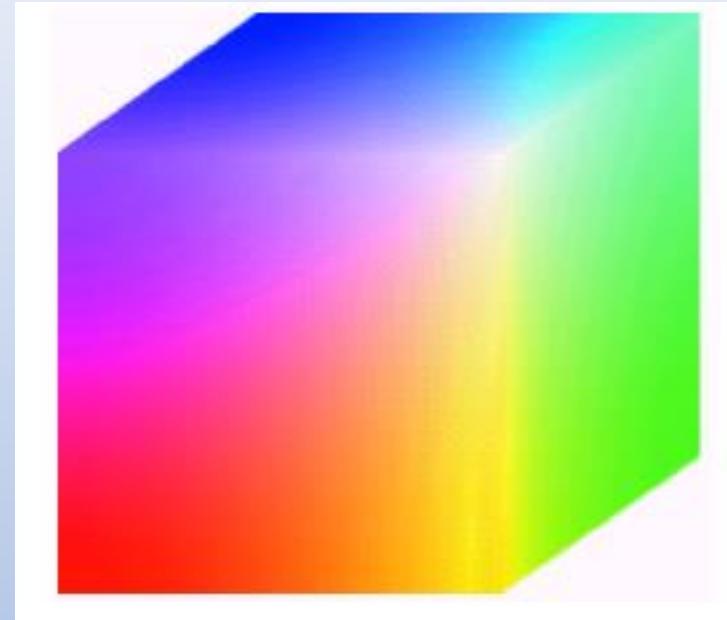
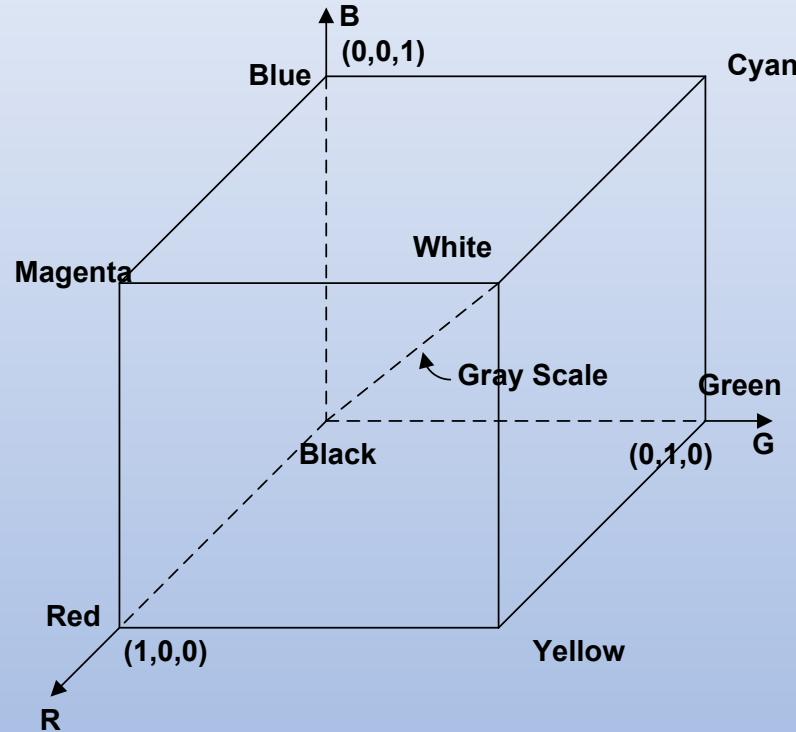
# 1. Image Fundamentals—color space

- Additive primaries: Red(R), Green(G), Blue(B)
- Subtractive primaries: Cyan, Magenta, Yellow
- A color can be specified in terms of the amounts of three primaries required:  $c = a \times p1 + b \times p2 + c \times p3$ , where  $(p1, p2, p3)$  is a particular set of primaries.
- A color space is a 3D space, defined to describe color in some standard way.



# 1. Image Fundamentals—color space

➤ RGB color spaces:



Z. Lu, X.D. Jiang and A. Kot, “[A Color Channel Fusion Approach for Face Recognition](#),” *IEEE Signal Processing letters*, vol. 22, no. 11, pp. 1839 - 1843, Nov. 2015.

# 1. Image Fundamentals—color space

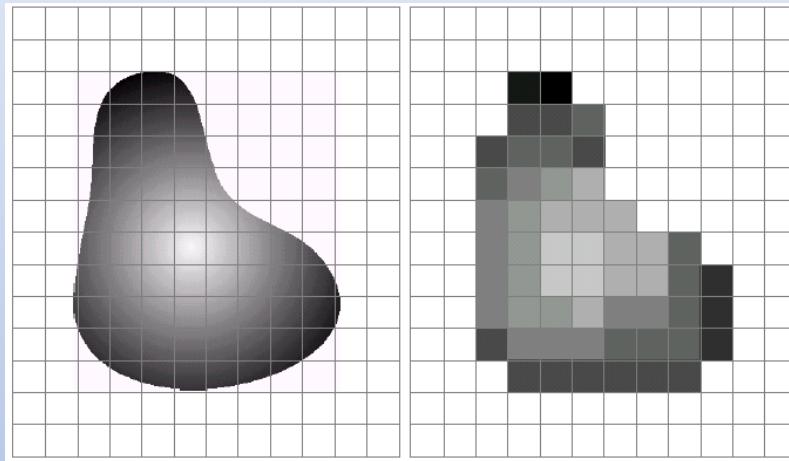
## ➤ Common color spaces:

- RGB - hardware oriented, used for monitors, video cameras
- rgb - Normalized RGB
- CMY (Cyan-Magenta-Yellow) - used for color printer
- YIQ (luminance, in-phase, quadrature. ) - color TV broadcast
- HSI (HSV) (Hue, saturation, intensive/value) - used for color manipulation
- CIE-Luv, CIE-Lab (lightness, red-green, yellow-blue) - used for color differentiation.
- sRGB – used for device independent digital image display.

Z. Lu, X. Jiang, A. Kot, “Color Space Construction by Optimizing Luminance and Chrominance Components for Face Recognition,” ***Pattern Recognition***, vol. 83, pp. 456-468, 2018.

# 1. Image Fundamentals—representation of image

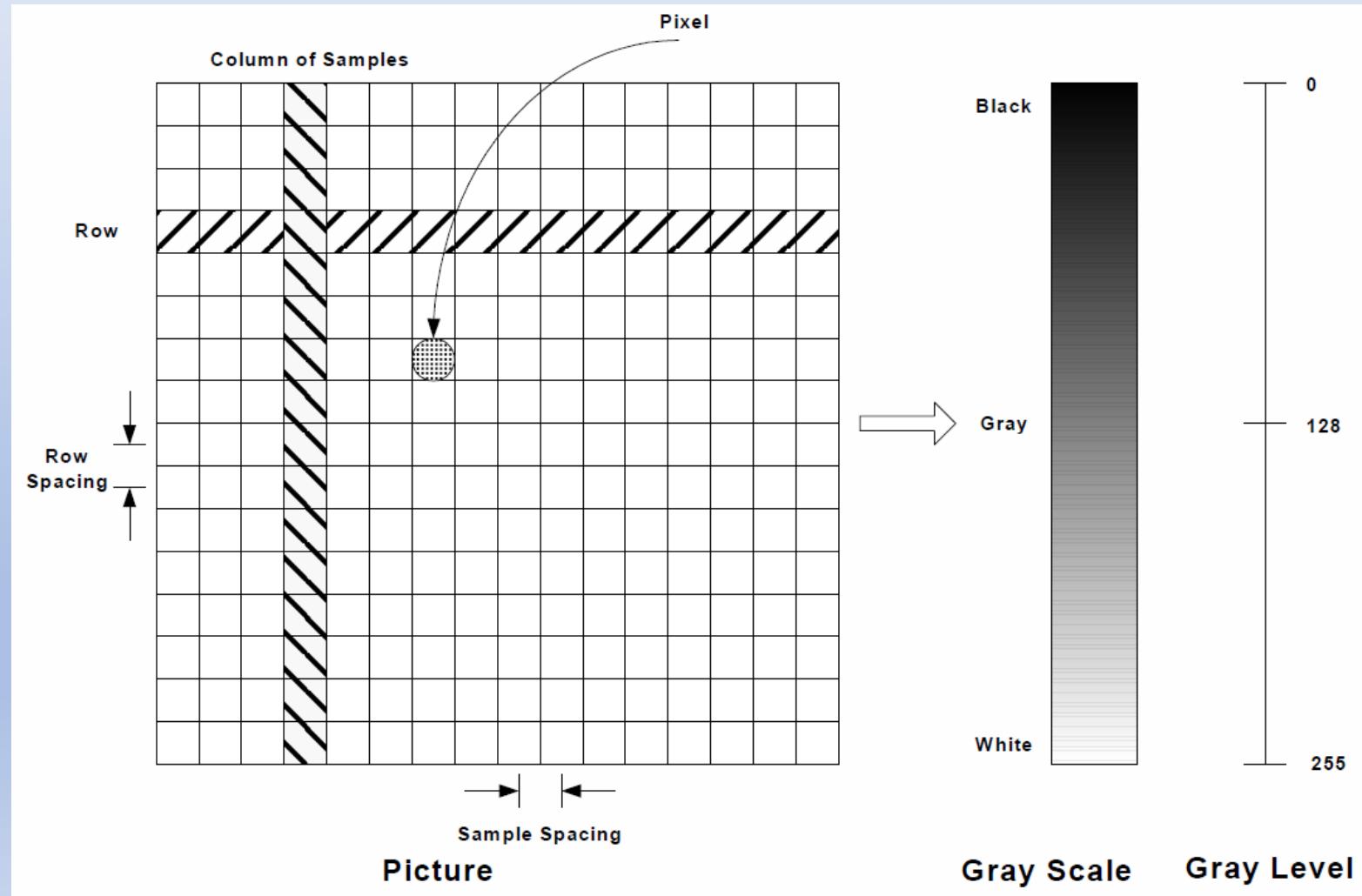
- Mathematically, an Image is a two-dimensional (2-D) function  $f(x,y)$ , a function of the two spatial coordinates.



- A digital image is a sampled, quantized version of a 2D light-intensity function generated by optical means.
- The function is usually sampled in an equally spaced rectangular grid pattern, with its amplitude quantized in equal intervals.
- A digital image is  $f(x,y)$  where  $x$ ,  $y$  and  $f$  are all finite and discrete quantities.

# 1. Image Fundamentals—image digitization

- Digitization of an image: **Spatial sampling** or discretization and Intensity or gray-level quantization



# 1. Image Fundamentals—representation of image

- Denote an image, a 2-D light-intensity function, as  $f(x,y)$ .
- The value or amplitude of  $f$  at spatial coordinates  $(x, y)$  indicates the intensity (brightness, grey level) of the image at that point.
- $f(x,y)$  must be digitized both spatially and in amplitude for computer processing.
- Digitization of spatial coordinates  $f(x,y)$  is referred to as spatial sampling or discretization.
- Digitization of intensity amplitude  $f(x,y)$  is referred to as intensity or gray-level quantization.

# 1. Image Fundamentals—representation of image

- The (spatial) resolution of a digital image refers to the size of the  $m \times n$  array of which the image is sampled.

$$f(x, y) = \text{e.g.: } \sin[2\pi(u \sin(\phi)x + v \sin(\phi)y)]$$

$$\begin{bmatrix} f(1,1) & f(1,2) & \cdots & f(1,n) \\ f(2,1) & f(2,2) & \cdots & f(2,n) \\ \vdots & \vdots & \ddots & \vdots \\ f(m,1) & f(m,2) & \cdots & f(m,n) \end{bmatrix} \text{ or } \mathbf{F} = \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1n} \\ f_{21} & f_{22} & \cdots & f_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ f_{m1} & f_{m2} & \cdots & f_{mn} \end{bmatrix}$$

- The gray-level resolution of a digital image refers to the number of gray levels (intensities)  $g=2^b$ , where  $b$  is the number of bits per sample.

# 1. Image Fundamentals—spatial resolution

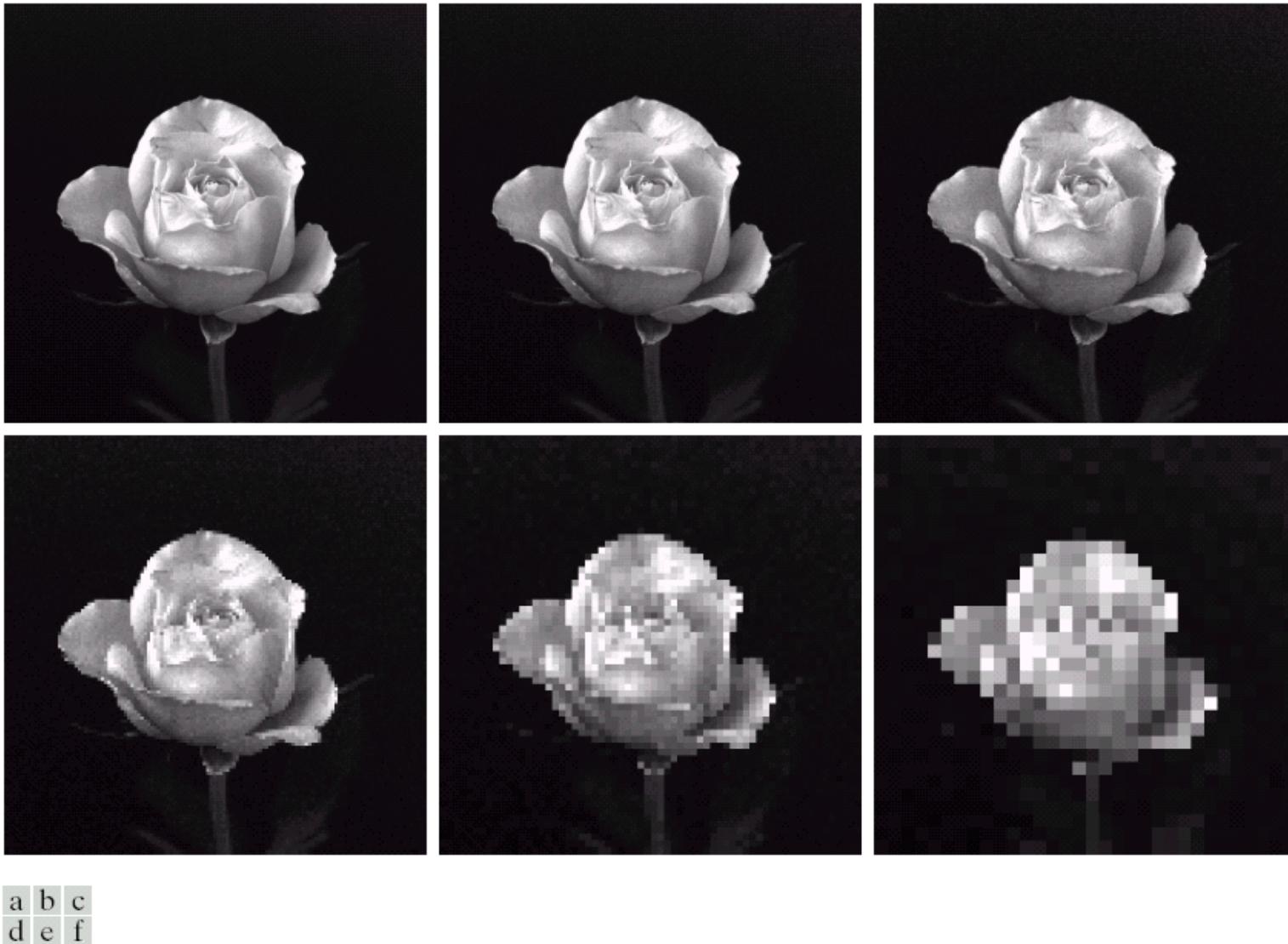


600 x 408 pixels

150 x 102 pixels



# 1. Image Fundamentals—spatial resolution



**FIGURE 2.20** (a)  $1024 \times 1024$ , 8-bit image. (b)  $512 \times 512$  image resampled into  $1024 \times 1024$  pixels by row and column duplication. (c) through (f)  $256 \times 256$ ,  $128 \times 128$ ,  $64 \times 64$ , and  $32 \times 32$  images resampled into  $1024 \times 1024$  pixels.

# 1. Image Fundamentals—gray-level resolution

256 levels



128 levels



64 levels



32 levels



16 levels



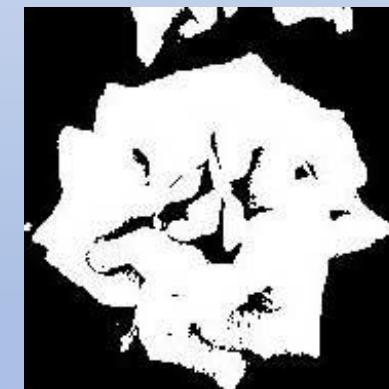
8 levels



4 levels



2 levels



# 1. Image Fundamentals—image histogram

- The **histogram** of a digital image  $f(x,y)$  with gray level range  $[0, L]$  is a discrete function

$$p_f(f) = \frac{n_f}{n}$$

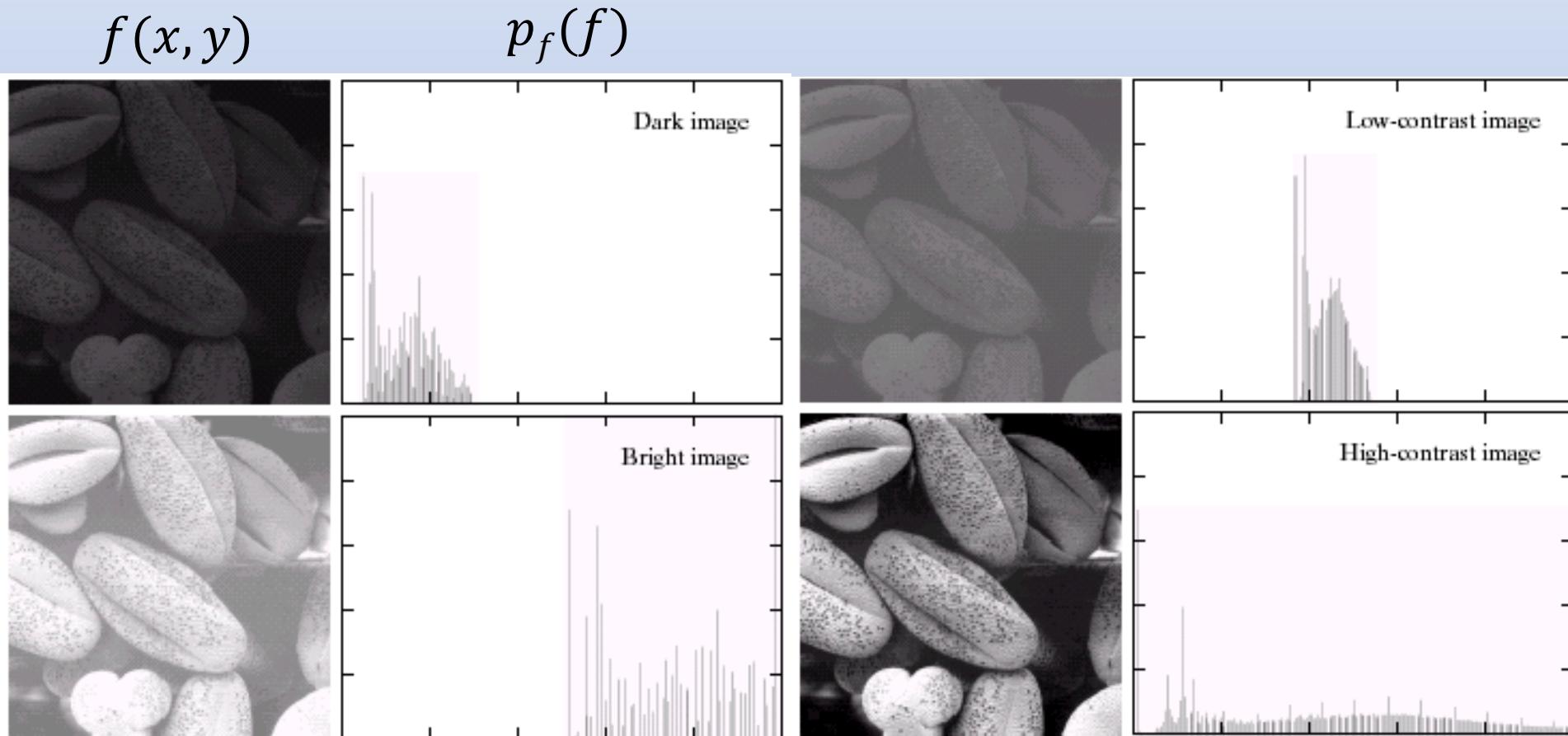
- where  $f$  is the gray level,  $f = 0, 1, 2, \dots, L$ .  $n_f$  is the number of pixels with that gray level.  $n$  is the total number of pixels in the region of the image being processed.
- It is clear that the histogram  $p_f(f)$  of a digital image is the **frequency of occurrence** of gray-level  $f$  in the image.
- Histogram shows the **frequency distribution** of gray-level  $f$ .
- Obviously,

$$p_f(f) \geq 0, \quad \text{and} \quad \sum_{f=0}^L p_f(f) = 1$$

- If we treat the pixel gray-level of an image as a random variable, the histogram  $p_f(f)$  is an **estimate** of the **probability of occurrence** of gray-level  $f$  over an image.

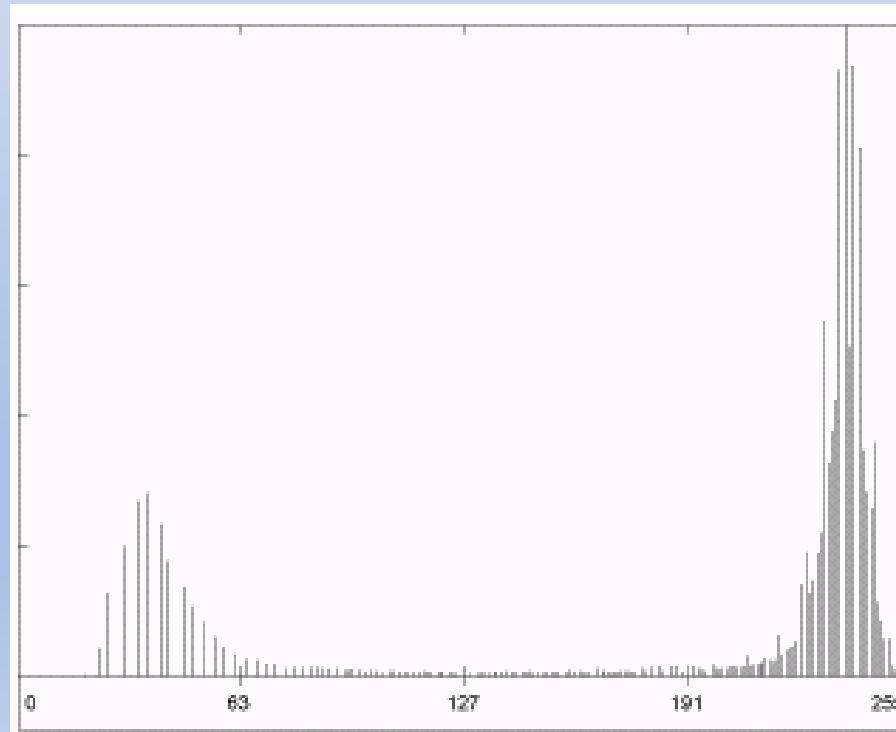
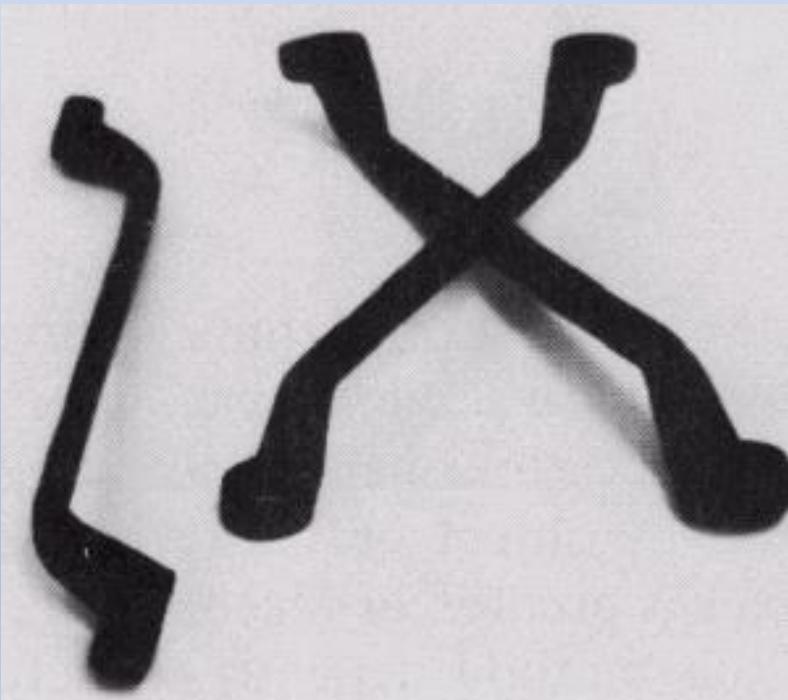
# 1. Image Fundamentals—image histogram

- **Understand the histogram:** The shape of image histogram provides many clues as to the characteristics of image. For example,
  - a narrowly distributed histogram indicates a low-contrast image



# 1. Image Fundamentals—image histogram

- **Understand the histogram:** The shape of image histogram provides many clues as to the characteristics of image. For example,
  - A bimodal histogram suggests that the image contains an object with a narrow amplitude range against a background of differing amplitude.



# 1. Image Fundamentals—image histogram

## ➤ Some Journal papers exploring histogram features:

- J. Ren, X. Jiang and J. Yuan, "[LBP Encoding Schemes Jointly Utilizing the Information of Current Bit and Other LBP Bits](#)," *IEEE Signal Processing letters*, vol. 22, no. 12, pp. 2373 - 2377, Dec. 2015.
- J. Ren, X. Jiang and J. Yuan, "[A Chi-Squared-Transformed Subspace of LBP Histogram for Visual Recognition](#)," *IEEE Trans. Image Processing*, vol. 24, no. 6, pp. 1893-1904, June, 2015.
- J. Ren, X. Jiang and J. Yuan, "[Learning LBP Structure by Maximizing the Conditional Mutual Information](#)," *Pattern Recognition*, vol. 48, no. 10, pp. 3180 - 3190, Oct. 2015.
- A. Satpathy, X. Jiang and H. Eng, "[LBP Based Edge-Texture Features for Object Recognition](#)," *IEEE Trans. Image Processing*, vol. 23, no. 5, pp. 1953-1964, May, 2014.
- J. Ren, X. Jiang, J. Yuan and W. Gang, "[Optimizing LBP Structure for Visual Recognition Using Binary Quadratic Programming](#)," *IEEE Signal Processing letters*, vol. 21, pp. 1346-1350, Nov. 2014.
- A. Satpathy, X. Jiang and H. Eng, "[Human Detection by Quadratic Classification on Subspace of Extended Histogram of Gradients](#)," *IEEE Trans. Image Processing*, vol. 23, pp. 287-297, Jan, 2014.
- J. Ren, X. Jiang and J. Yuan, "[Noise-Resistant Local Binary Pattern with an Embedded Error-Correction Mechanism](#)," *IEEE Trans. Image Processing*, vol. 22, no. 10, pp. 4049-4060, Oct, 2013.

# 1. Image Fundamentals—what is image processing?

- An digital image is a two dimensional numerical representation (a 2-D function  $f(x,y)$  or a  $mxn$  matrix) of a 3D scene or an object.

## What is digital image processing?

- Digital image processing is a series of machine or computer operations leading to some desired results.
- The operations could be, should be, and are desired to be described by mathematics.
- Digital image processing, starting with an image or a set of images, produces a modified version of the image(s) or extract more “meaningful” information (features) from the image(s) or understand (recognize) the meaning of the image content.

# 1. Image Fundamentals—Why need image processing?

## ➤ Why need image processing?

- Visualization :
  - Contrast enhancement, noise removal, visual quality improvement, pseudo colouring
- Image understanding
  - Extraction of image properties such as colour, shape, texture, edges, lines, curves, corners.
- Automated Guided vehicle
  - Identifying road, vehicle, pedestrian, traffic signs
- Visual servicing
  - Automated robot control
- Security
  - Intrusion detection, biometrics
- Information retrieval
  - Image content based search

# 1. Image Fundamentals—image processing examples

## ➤ Contrast Enhancement



# 1. Image Fundamentals—image processing examples

## ➤ Deblurring



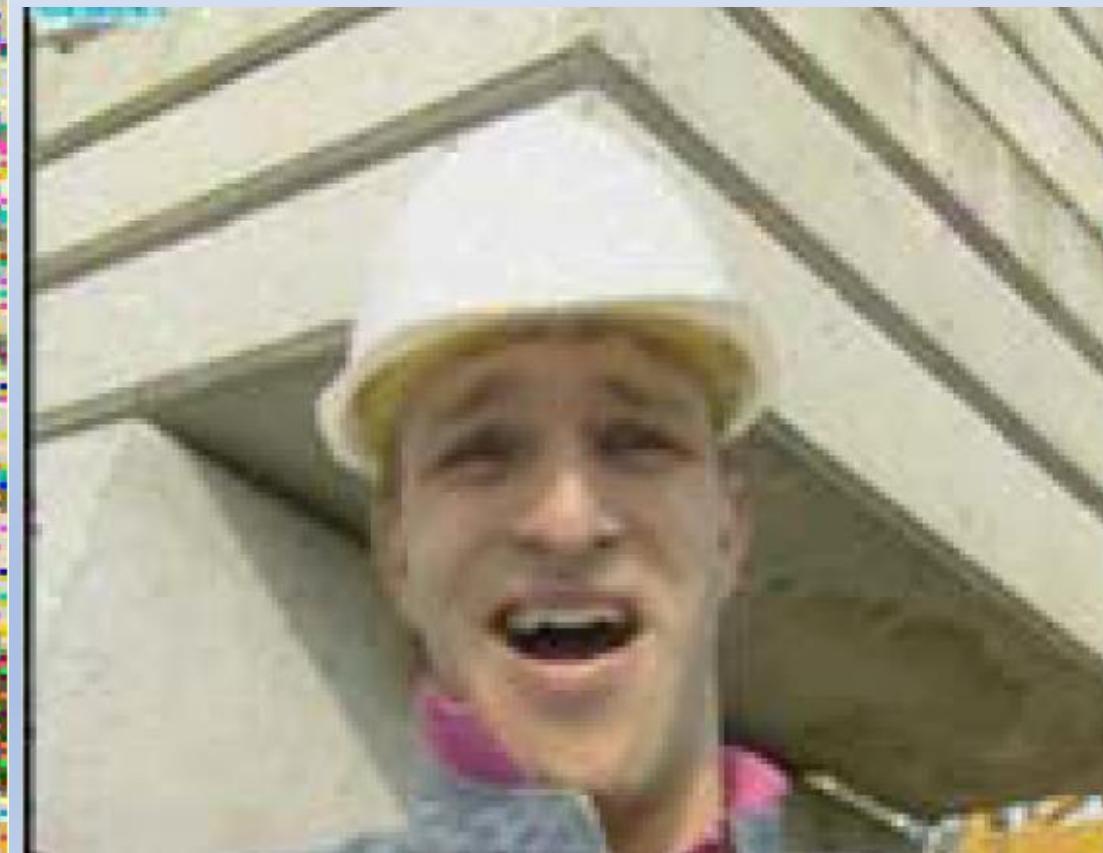
# 1. Image Fundamentals—image processing examples

## ➤ Denoising: noise attenuation

Noise corrupted image and the image **processed by DIP**



exdjiang@ntu.edu.sg



<https://personal.ntu.edu.sg/exdjiang/>

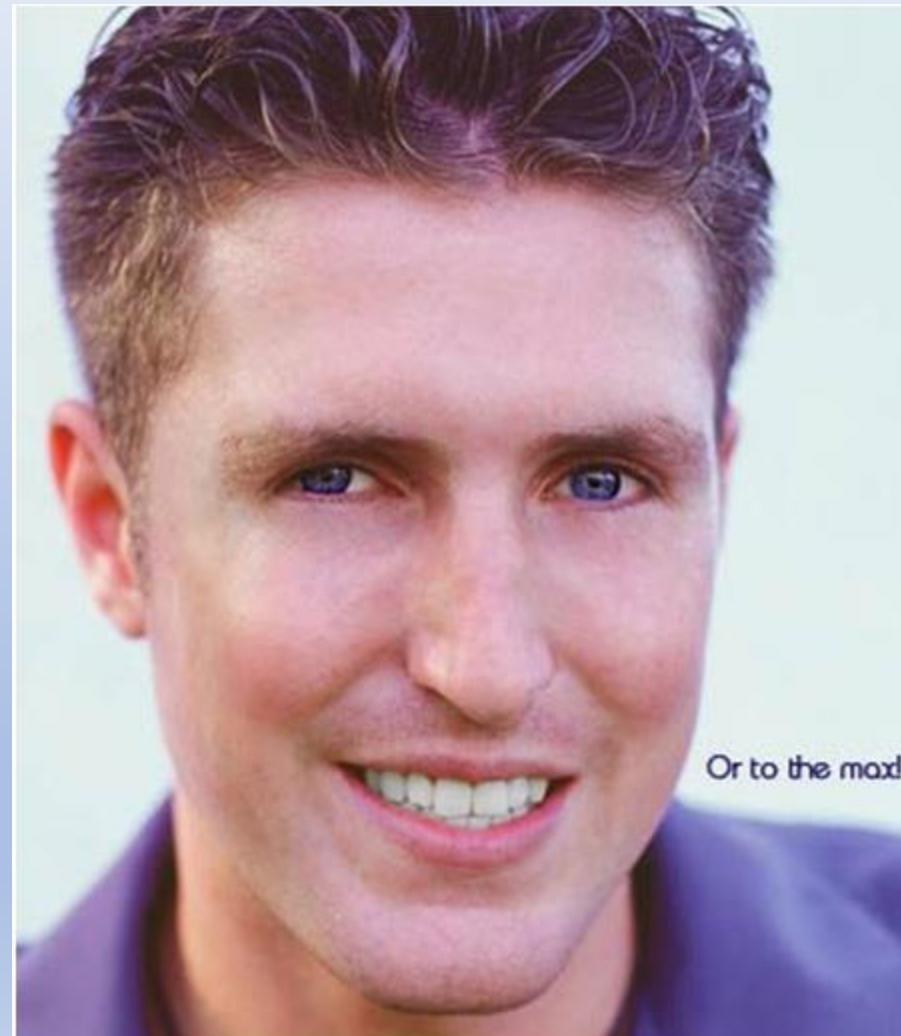
# 1. Image Fundamentals—image processing examples

## ➤ Beautifying



exdjiang@ntu.edu.sg

<https://personal.ntu.edu.sg/exdjiang/>



Or to the max!

32

# 1. Image Fundamentals—image processing examples

## ➤ Restoration and Retouching



exdjiang@ntu.edu.sg



<https://personal.ntu.edu.sg/exdjiang/>

33

# 1. Image Fundamentals—image processing examples

## ➤ Segmentation



# 1. Image Fundamentals—image processing examples

## ➤ Digital Watermarking

### ▪ Traditional Watermark (bank notes)

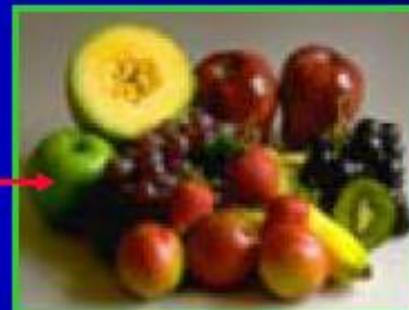
Watermark would appear  
when placed in the  
presence of ultra-violet light



### ▪ Digital Watermark (digital images)



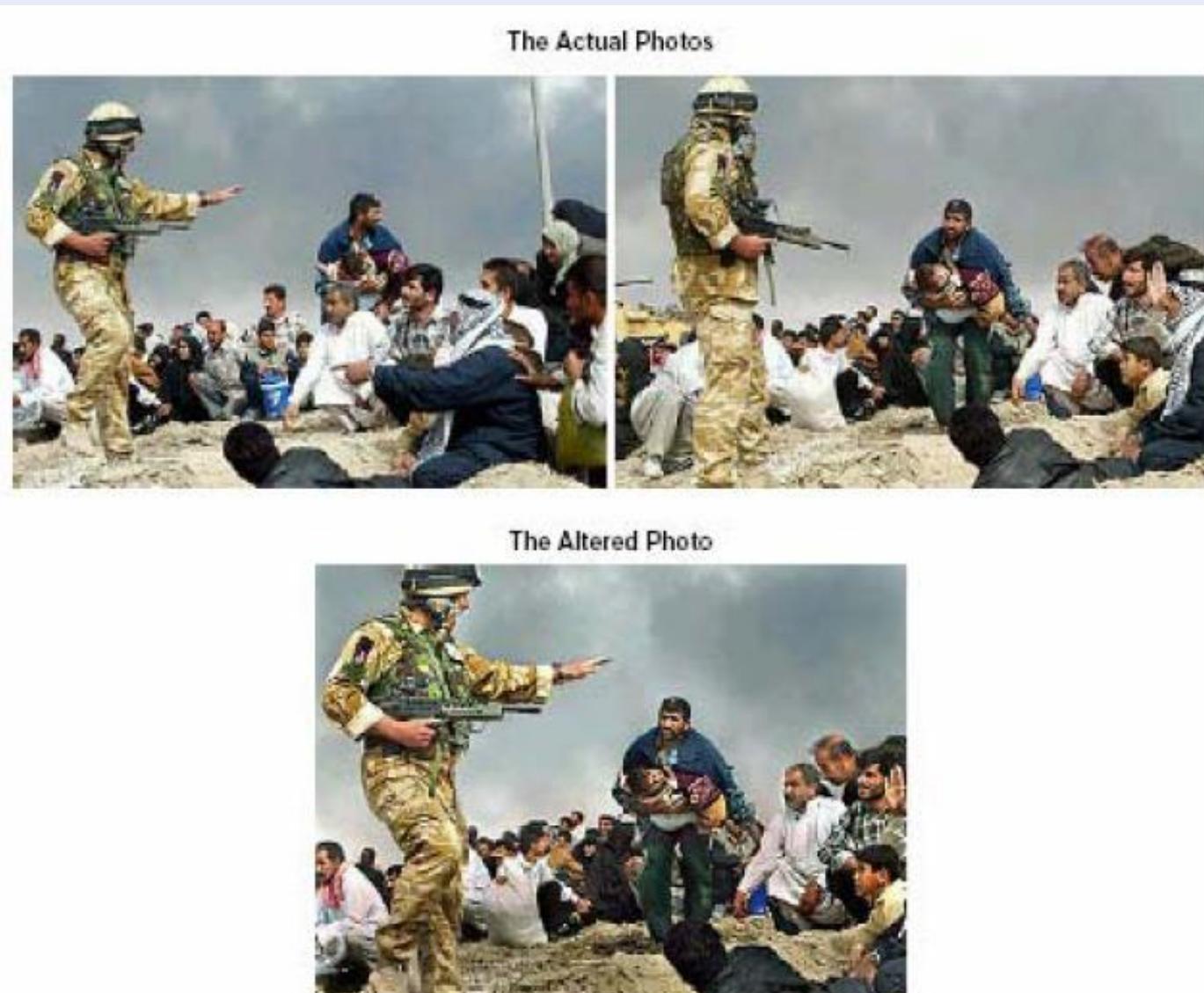
Digital watermark retrieved  
through an algorithm



# 1. Image Fundamentals—image processing examples

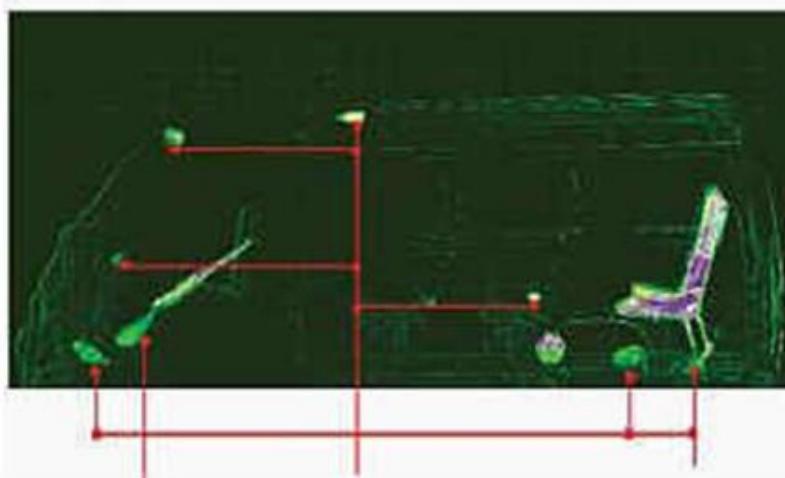
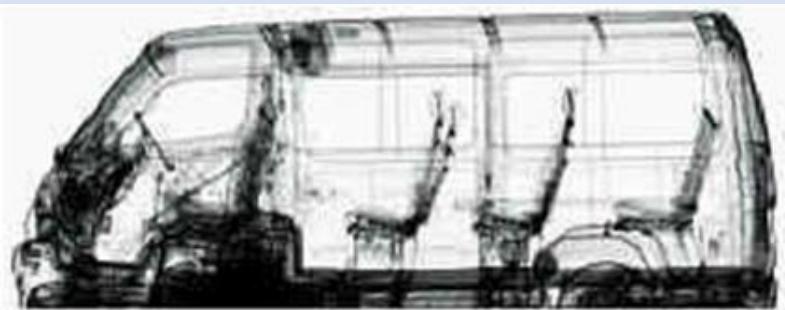
Digital content  
tampering  
detection

The photograph is a composite created by ex-composite LA Times photographer Brian Walski.  
He was dismissed when the photograph was found be altered.



# 1. Image Fundamentals—image processing examples

## ➤ Objection Detection



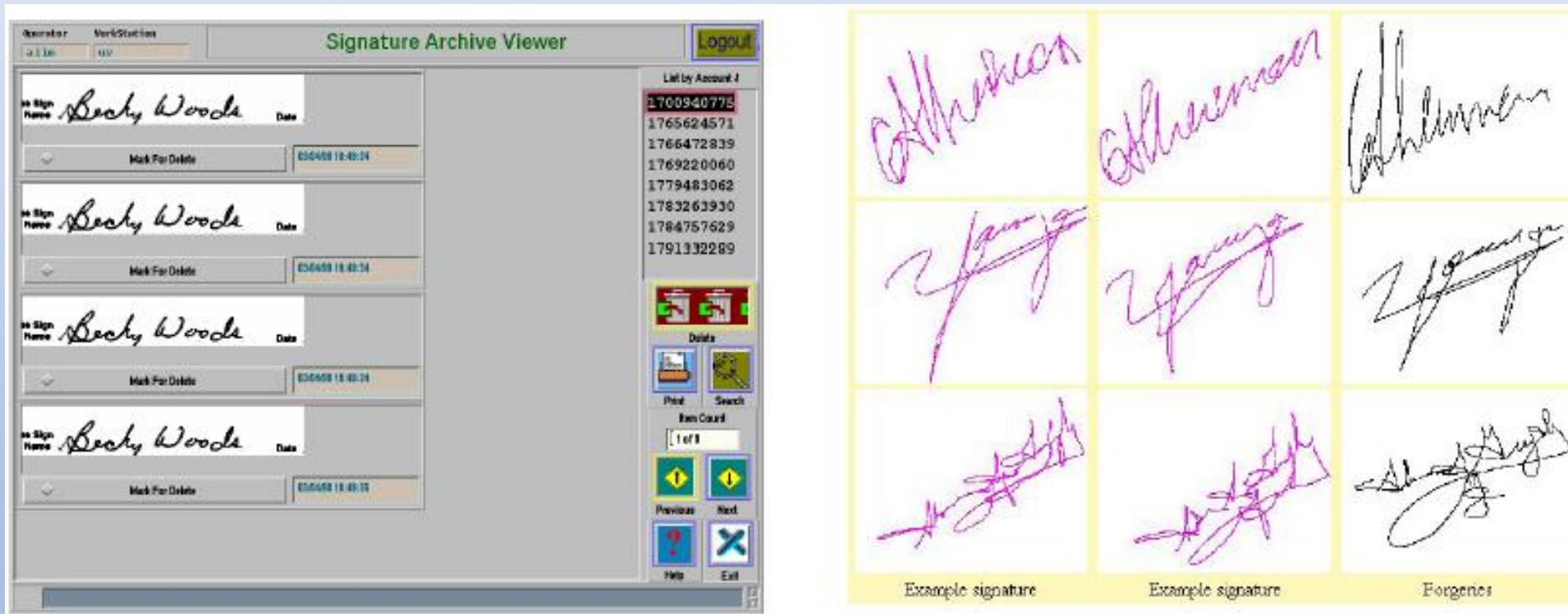
# 1. Image Fundamentals—image processing examples

## ➤ Face Detection



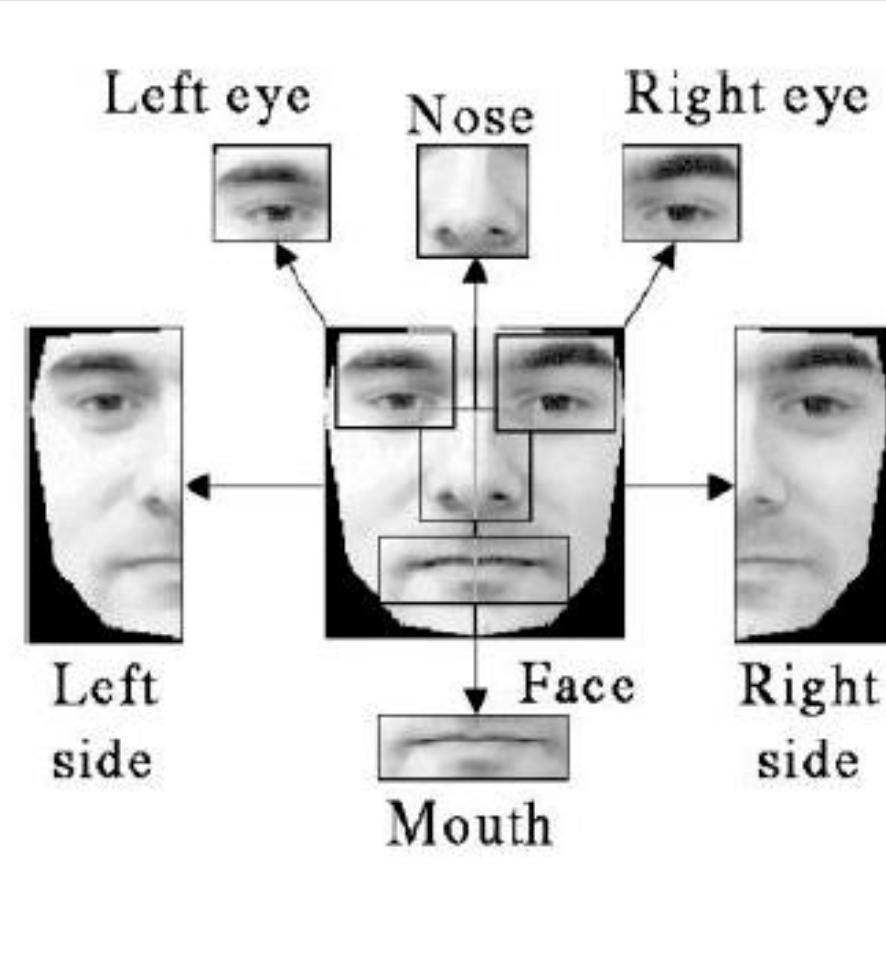
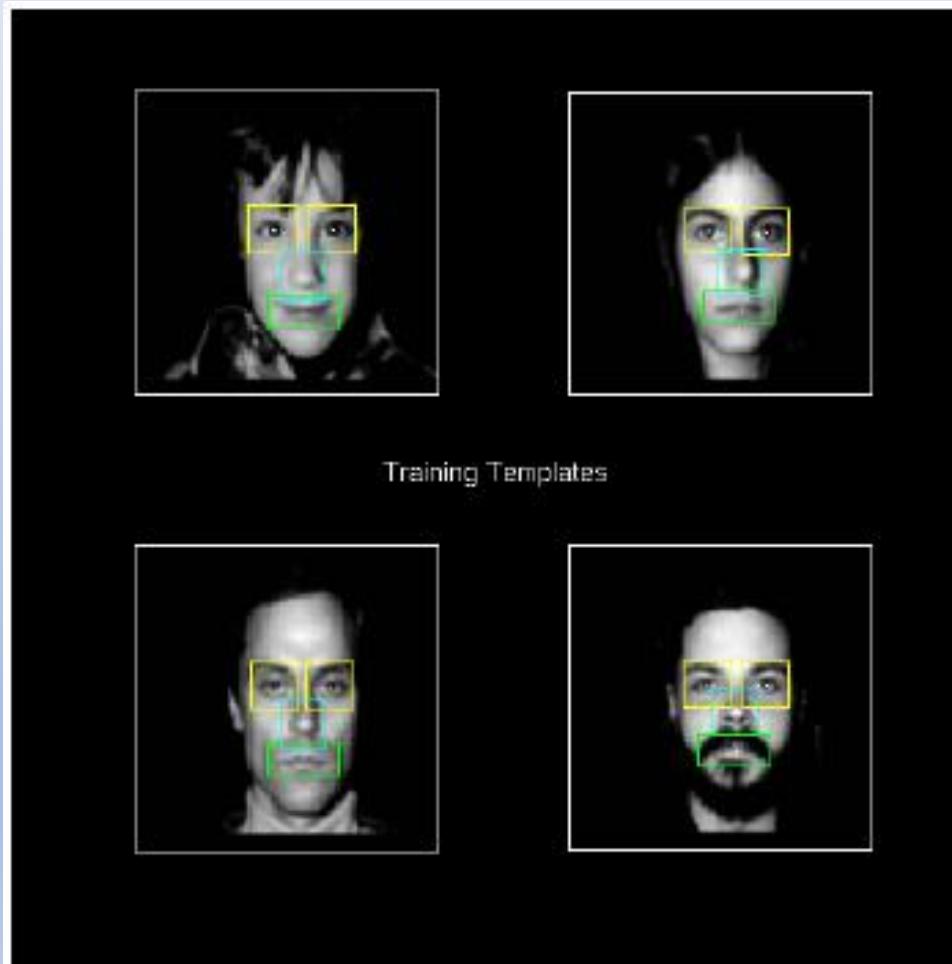
# 1. Image Fundamentals—image processing examples

## ➤ Automatic signature verification and identification

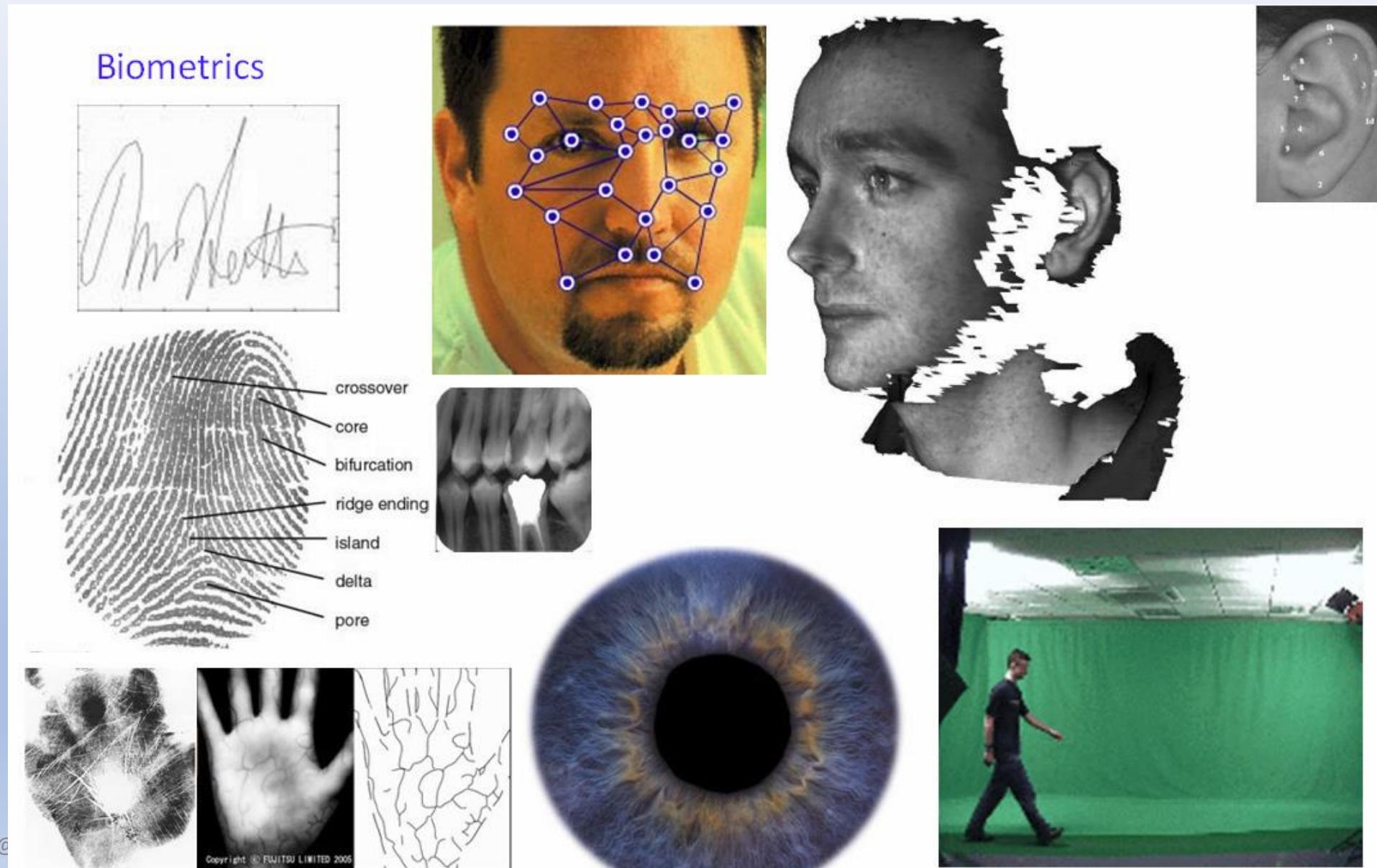


# 1. Image Fundamentals—image processing examples

Are these features effective for **Automatic Face Recognition?**



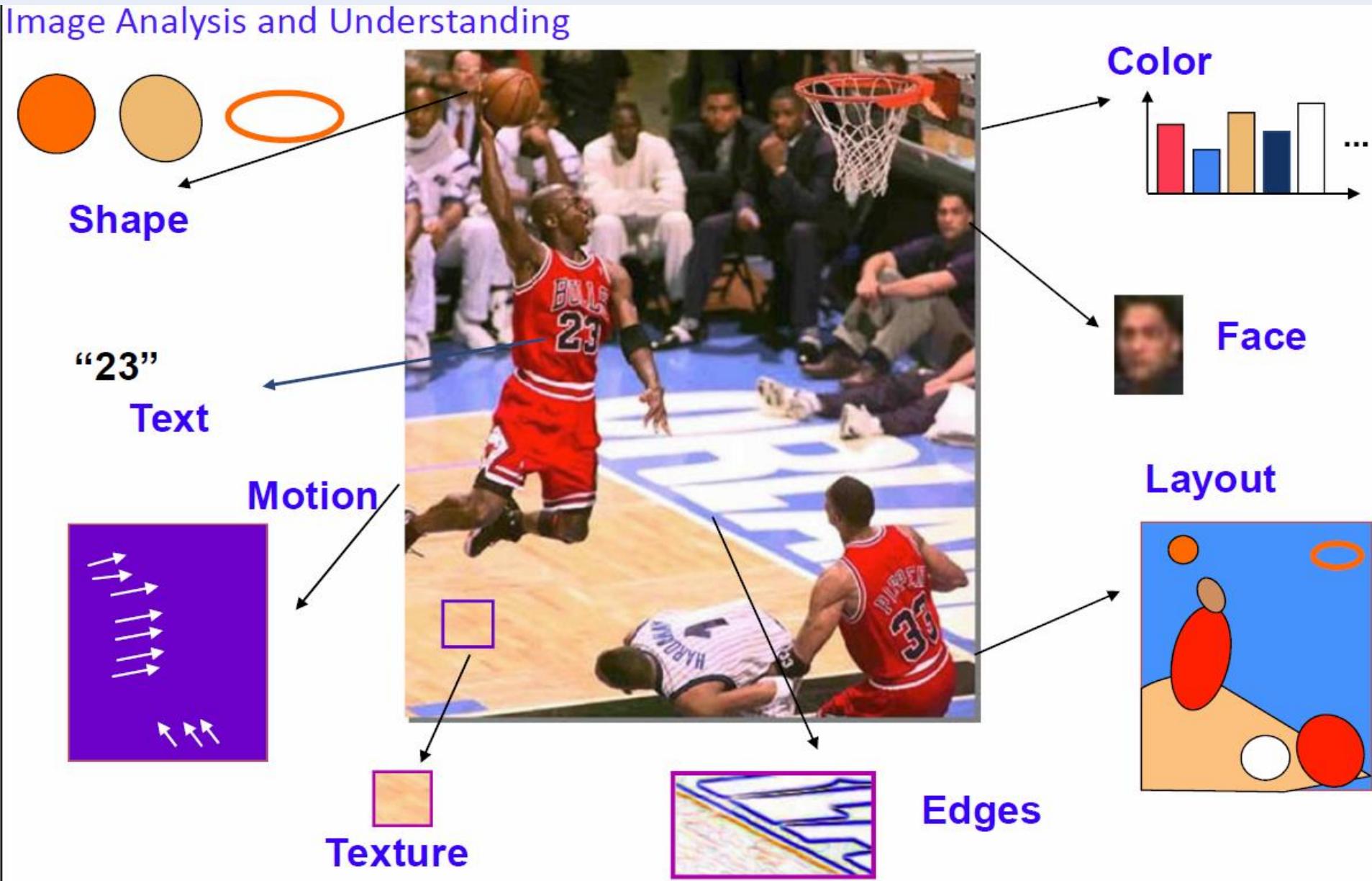
# 1. Image Fundamentals – image processing examples



# 1. Image Fundamentals – image processing examples



# 1. Image Fundamentals—image processing examples



## 2. LSI Systems & Transforms—Outline

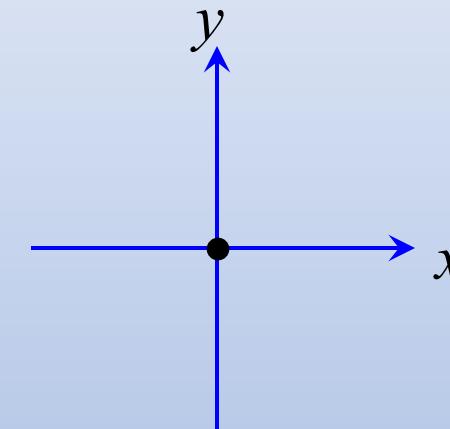
- Image Decomposition and Linear Shift Invariant Image Processing System
- Two-dimensional Convolution and its Properties
- Two-dimensional Fourier Transform and its Properties
- Image Sampling

## 2. LSI Systems & Transforms—basic image element

- A **digital image** can be represented by a 2-D function with two **integer** arguments, such as  $f(x,y)$  where  $x$  and  $y$  are integers

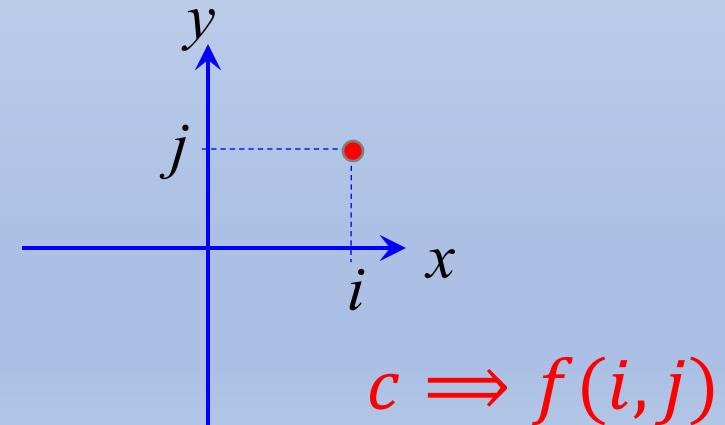
- The basic element image is the impulse

$$\delta(x,y) = \begin{cases} 1, & x = y = 0 \\ 0, & \text{otherwise} \end{cases}$$



$$f(x,y) = c\delta(x - i, y - j) = \begin{cases} c, & x = i, y = j \\ 0, & \text{otherwise} \end{cases}$$

- Shifted and scaled the impulse
- It describes any arbitrary pixel with all other pixels zero gray value.



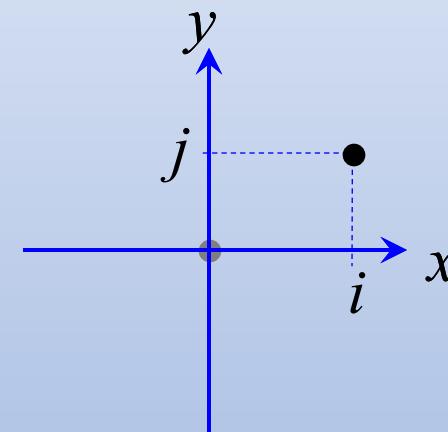
## 2. LSI Systems & Transforms—image decomposition

- Any image  $f(x,y)$  then can be represented by the sum of a number of shifted and scaled impulses

$$f(x,y) = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} f(i,j) \delta(x - i, y - j)$$

Or

$$f(x,y) = \sum_{j=-n}^{n} \sum_{i=-m}^{m} f(i,j) \delta(x - i, y - j)$$



- For example, a constant gray level 215 square of size 11X11 centered at (0,0) is:

$$f(x,y) = \sum_{j=-5}^{5} \sum_{i=-5}^{5} 215 \delta(x - i, y - j)$$

## 2. LSI Systems & Transforms—2-D convolution

- A processing system relates any input image  $f(x,y)$  to a unique output image  $g(x,y)$ , given by

$$g(x,y) = T\{f(x,y)\} = T \left\{ \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} f(i,j) \delta(x-i, y-j) \right\}$$

- If the processing system is linear, then

$$g(x,y) = T\{f(x,y)\} = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} f(i,j) T\{\delta(x-i, y-j)\}$$

- If define the output image of the input impulse image as impulse response of the system,

$h(x,y) \triangleq T\{\delta(x,y)\}$ . Then for shift invariant system:  
 $T\{\delta(x-i, y-j)\} = h(x-i, y-j)$

## 2. LSI Systems & Transforms—2-D convolution

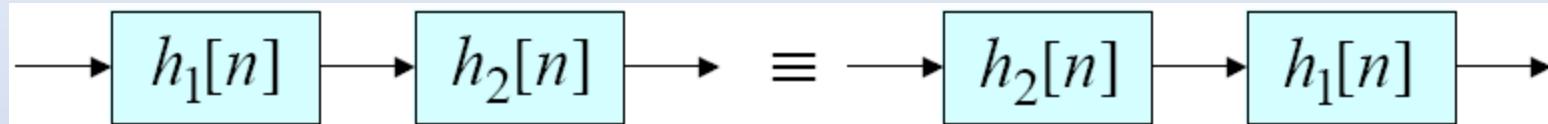
- Therefore, given an input image  $f(x,y)$ , a **linear and shift-invariant (LSI)** image processing system  $T$  produces the output image  $g(x,y)$  by

$$g(x,y) = T\{f(x,y)\} = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} f(i,j)h(x-i, y-j)$$
$$\triangleq f(x,y) * h(x,y) = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} h(i,j)f(x-i, y-j)$$

- A LSI system is **completely characterized by its impulse response  $h(x,y)$ .**  $*$  is the **convolution operator.**
- For any LSI image processing system, the output image equals to the input image convolving with the impulse response of the system.

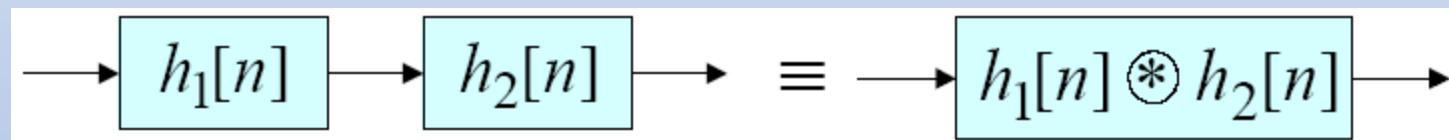
## 2. LSI Systems & Transforms—convolution property

➤ **Commutative:**  $f(x, y) * h(x, y) = h(x, y) * f(x, y)$



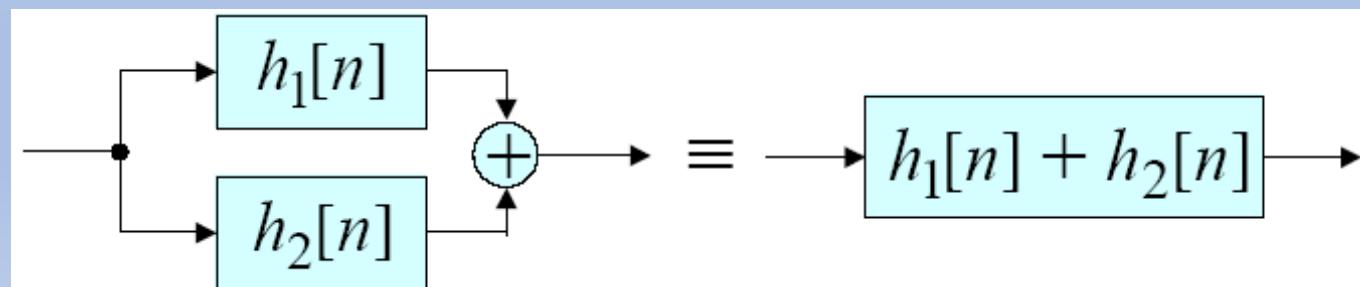
➤ **Associative:**

$$f(x, y) * (h_1(x, y) * h_2(x, y)) = (f(x, y) * h_1(x, y)) * h_2(x, y)$$



➤ **Distributive:**

$$f(x, y) * (h_1(x, y) + h_2(x, y)) = f(x, y) * h_1(x, y) + f(x, y) * h_2(x, y)$$



## 2. LSI Systems & Transforms—what is $h(x,y)$ ?

Understanding the impulse response:

- Impulse response  $h(x,y)$  of an image processing system is the output image when an impulse image is inputted to the system.
- Therefore, the impulse response  $h(x,y)$  is also an image, often called spatial representation of a filter or filter mask or filter coefficients or filter parameters.
- Although the impulse response  $h(x,y)$  is basically an image, to speed up the image process, it is often a small size of image with size such as 3X3, 5X5, ...11X11, comparing with a normal image of size 256X256.

$$\begin{aligned} g(x, y) &= f(x, y) * h(x, y) = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} h(i, j)f(x - i, y - j) \\ &= \sum_{j=-3}^{3} \sum_{i=-3}^{3} h(i, j)f(x - i, y - j), \quad \text{if } h(x, y) \neq 0 \text{ only when } -3 < x, y < 3 \end{aligned}$$

## 2. LSI Systems & Transforms—understand by example

- Given an input image  $f(x,y)$ , you want to suppress the pixel random noise or smooth the image so that the image looks “soft”. So you do some local average that a pixel in the output image is produced by sum up the corresponding pixel and its 4 neighbor pixels in the input image.
- What is the mathematical expression of this very simple process?

$$g(x, y) = f(x, y) + f(x - 1, y) + f(x + 1, y) + f(x, y - 1) + f(x, y + 1)$$

- What is the convolution like?

$$g(x, y) = \sum_{j=-1}^1 \sum_{i=-1}^1 h(i, j)f(x - i, y - j)$$

- What is the size of the impulse response? Is the impulse response a constant one within the filter window  $h(x,y)=1$ ?

$$g(x, y) = \sum_{j=-1}^1 \sum_{i=-1}^1 f(x - i, y - j) \times$$

- What is the impulse response analytically?

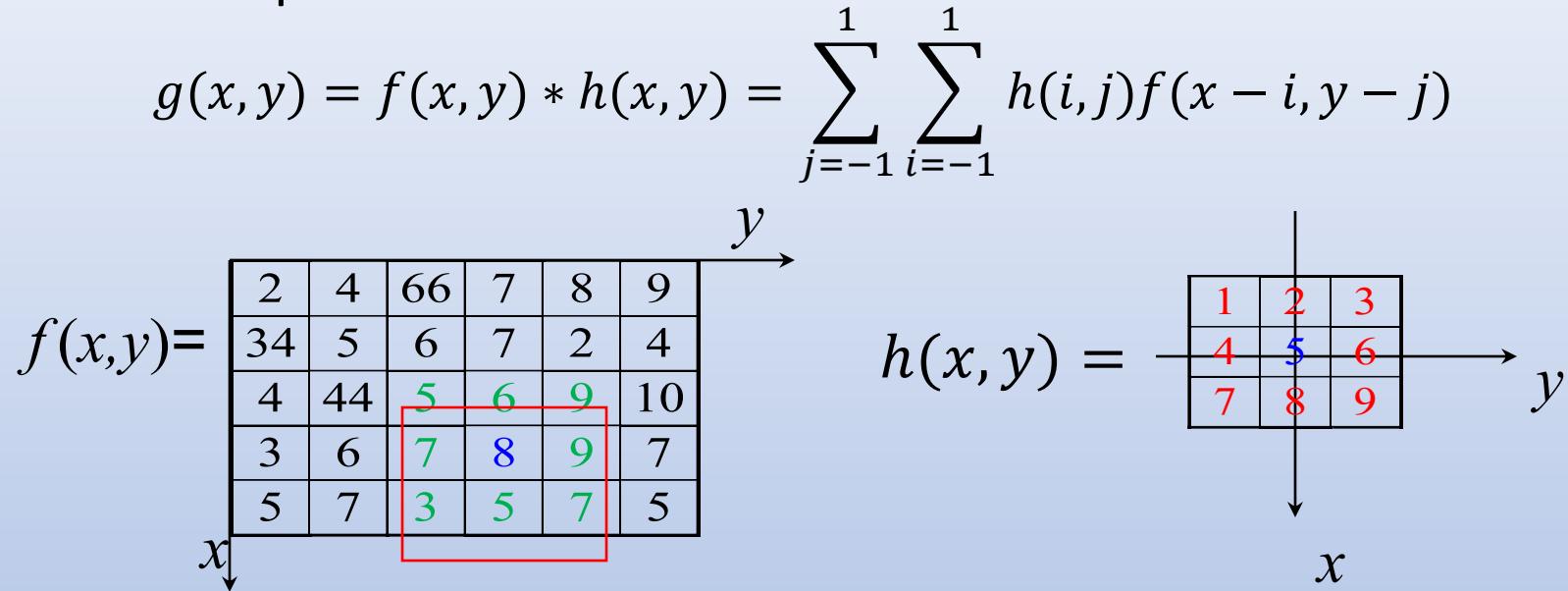
$$h(x, y) = \delta(x, y) + \delta(x - 1, y) + \delta(x + 1, y) + \delta(x, y - 1) + \delta(x, y + 1)$$

- How to represent the impulse response by image, or filter mask or filter coefficient?

0	1	0
1	1	1
0	1	0

## 2. LSI Systems & Transforms—understand by example

- A numerical example of convolution:



$$g(4,4) = \sum_{j=-1}^1 \sum_{i=-1}^1 h(i, j)f(4 - i, 4 - j)$$

- $g(4,4) = (7*1 + 5*2 + 3*3 + 9*4 + 8*5 + 7*6 + 9*7 + 6*8 + 5*9)$
- At each point  $(x, y)$  the response of the filter at that point is calculated as a sum of products of the filter coefficients and the corresponding image pixels in the area spanned by the filter mask.

## 2. LSI Systems & Transforms—Fourier transform

- Let  $f(x)$  be a continuous function of a single variable  $x$  and  $F(u)$  be its Fourier transform, then

$$F(u) = \mathfrak{F}\{f(x)\} = \int_{-\infty}^{\infty} f(x) \exp(-j2\pi ux) dx$$

$$\underline{f(x) = \mathfrak{F}^{-1}\{F(u)\} = \int_{-\infty}^{\infty} F(u) \exp(j2\pi ux) du}$$

$$\exp(j2\pi ux) = \cos(2\pi ux) + j \sin(2\pi ux)$$

where  $u$  is frequency variable and  $j = \sqrt{-1}$

- The two dimensional **(2-D) Fourier transform** and inverse Fourier transform are given by

$$F(u, v) = \mathfrak{F}\{f(x, y)\} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) \exp[-j2\pi(ux + vy)] dx dy$$

$$f(x, y) = \mathfrak{F}^{-1}\{F(u, v)\} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u, v) \exp[j2\pi(ux + vy)] du dv$$

where  $u$  and  $v$  are 2 frequency variables.

## 2. LSI Systems & Transforms—Fourier transform

- 2-D Fourier transform is separable

$$\begin{aligned} F(u, v) &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) \exp[-j2\pi(ux + vy)] dx dy \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) \exp(-j2\pi ux) \exp(-j2\pi vy) dx dy \\ &= \int_{-\infty}^{\infty} \left[ \int_{-\infty}^{\infty} f(x, y) \exp(-j2\pi ux) dx \right] \exp(-j2\pi vy) dy \\ &= \int_{-\infty}^{\infty} F_x(u, y) \exp(-j2\pi vy) dy \end{aligned}$$

$$? = F_x(u)F_y(v) \quad \text{only if } f(x, y) = f_1(x)f_2(y)$$

- Note

$$\begin{aligned} \exp[-j2\pi(ux + vy)] &= \cos[-2\pi(ux + vy)] + j \sin(-2\pi(ux + vy)) \\ &= \cos[2\pi(ux + vy)] - j \sin(2\pi(ux + vy)) \end{aligned}$$

## 2. LSI Systems & Transforms—Fourier transform

- Obviously,  $F(u, v)$  is in general a **complex function** that can be represented by

$$F(u, v) = R(u, v) + jI(u, v) = |F(u, v)| \exp[j\varphi(u, v)]$$

where

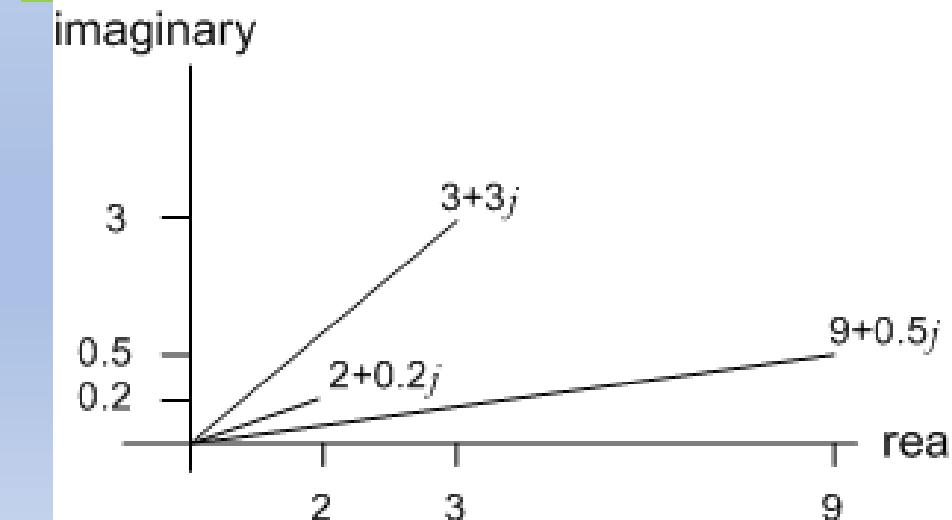
$$|F(u, v)| = \sqrt{R^2(u, v) + I^2(u, v)}$$

$$\varphi(u, v) = \tan^{-1} \left[ \frac{I(u, v)}{R(u, v)} \right]$$

$$R(u, v) = |F(u, v)| \cos \varphi(u, v)$$

$$I(u, v) = |F(u, v)| \sin \varphi(u, v)$$

$$c = a + jb = re^{j\varphi}$$
$$a = r\cos(\varphi), b = r\sin(\varphi)$$
$$r = \sqrt{a^2 + b^2}, \varphi = \tan^{-1}\left(\frac{b}{a}\right)$$



## 2. LSI Systems & Transforms—definition of DFT

- The discrete Fourier transform (**DFT**) of a 2-D discrete function (or image)  $f(x,y)$  of size  $m \times n$  is defined by:

$$F(u, v) = \sum_{x=0}^{m-1} \sum_{y=0}^{n-1} f(x, y) \exp[-j2\pi(ux/m + vy/n)]$$

$$f(x, y) = \frac{1}{mn} \sum_{u=0}^{m-1} \sum_{v=0}^{n-1} F(u, v) \exp[j2\pi(ux/m + vy/n)]$$

- where  $u$  and  $v$  are also discrete variable.
- Comparing with continuous Fourier transform:

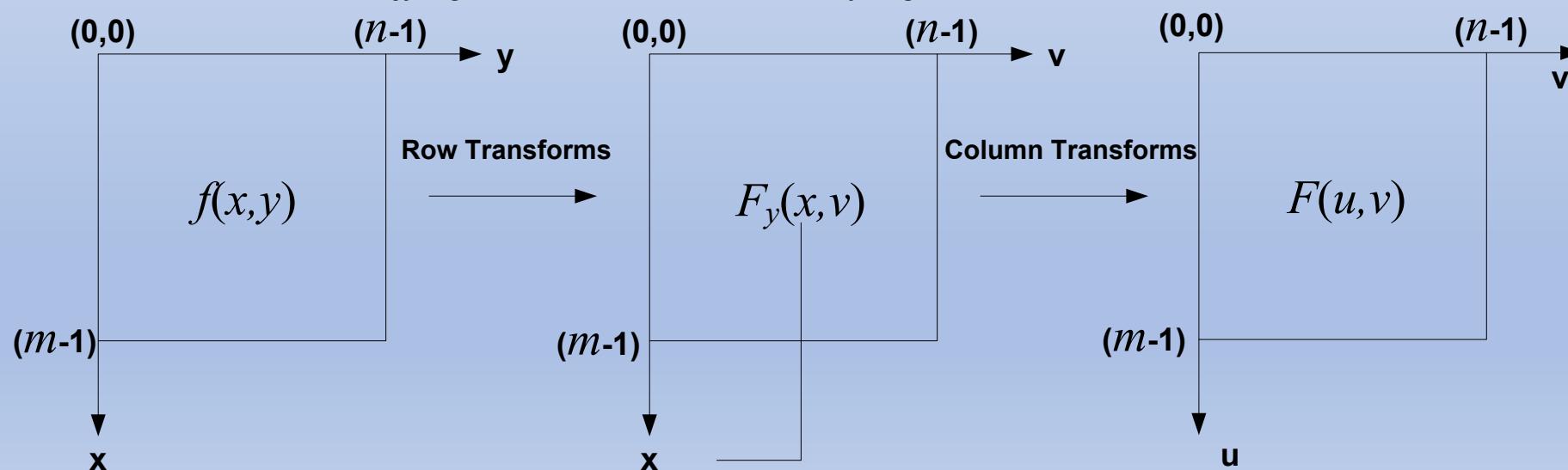
$$F(u, v) = \mathcal{F}\{f(x, y)\} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) \exp[-j2\pi(ux + vy)] dx dy$$

## 2. LSI Systems & Transforms—properties of DFT

➤ The 2-D DFT is also separable:

$$F(u, v) = \frac{1}{mn} \sum_{x=0}^{m-1} \exp[-j2\pi ux/m] \sum_{y=0}^{n-1} f(x, y) \exp[-j2\pi vy/n]$$

$$f(x, y) = \frac{1}{mn} \sum_{u=0}^{m-1} \exp[j2\pi ux/m] \sum_{v=0}^{n-1} F(u, v) \exp[j2\pi vy/n]$$



## 2. LSI Systems & Transforms—properties of DFT

➤ Periodicity:

$$F(u, v) = F(u + m, v) = F(u, v + n) = F(u + m, v + n)$$

➤ Conjugate symmetry for real image  $f(x, y)$

$$F(u, v) = F^*(-u, -v), \quad |F(u, v)| = |F(-u, -v)|$$

➤ Linearity and scaling:

$$\Im\{\alpha f_1(x, y) + \beta f_2(x, y) + \dots\} = \alpha F_1(u, y) + \beta F_2(u, y) + \dots$$

$$\Im\{f(\alpha x, \beta y)\} = \frac{1}{|\alpha\beta|} F(u/\alpha, y/\beta)$$

➤ Convolution theorem:

For continuous function:  $f(x, y) * g(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\alpha, \beta)g(x - \alpha, y - \beta)d\alpha d\beta$

$$f(x, y) * g(x, y) \Leftrightarrow F(u, v)G(u, v)$$

$$f(x, y)g(x, y) \Leftrightarrow F(u, v) * G(u, v)$$

➤ Be careful to apply this in digital image. Apply zero padding!

## 2. LSI Systems & Transforms—properties of DFT

### ➤ Translation

$$f(x - x_0, y - y_0) \Leftrightarrow F(u, v) \exp[-j2\pi(x_0 u/m + y_0 v/n)]$$

$$F(u - u_0, v - v_0) \Leftrightarrow f(x, y) \exp[j2\pi(u_0 x/m + v_0 y/n)]$$

### ➤ Rotation:

Let:  $x = r \cos \theta, y = r \sin \theta, u = \omega \cos \varphi, v = \omega \sin \varphi$

$$f(r, \theta + \theta_0) \Leftrightarrow F(\omega, \varphi + \theta_0)$$

### ➤ Rotation Invariant Transform:

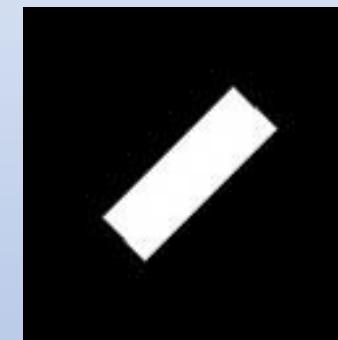
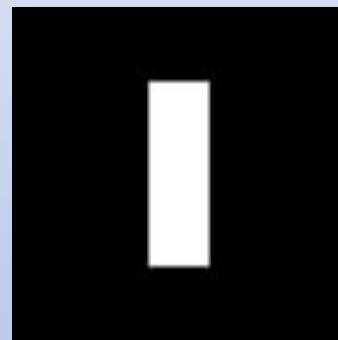
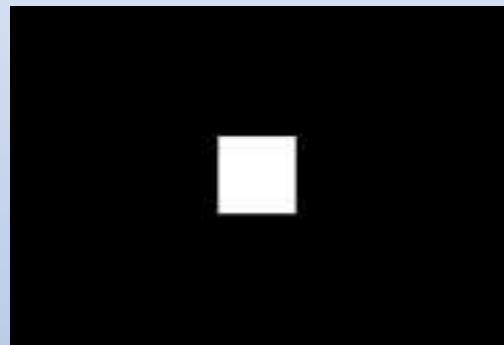
$$g(u, v) = \frac{1}{\pi} \int_0^{2\pi} \int_0^1 e^{-j(2\pi u r^2 + v\theta)} f(r, \theta) dr d\theta$$

P. Yap, X.D. Jiang and A. Kot, "[Two Dimensional Polar Harmonic Transforms for Invariant Image Representation](#)," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 7, pp. 1259-1270, July 2010.

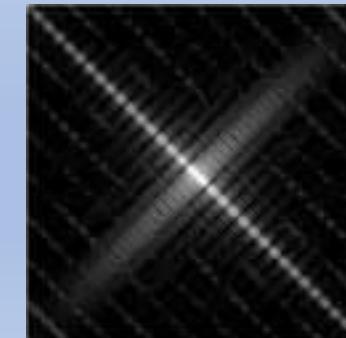
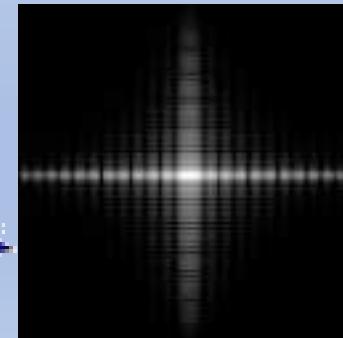
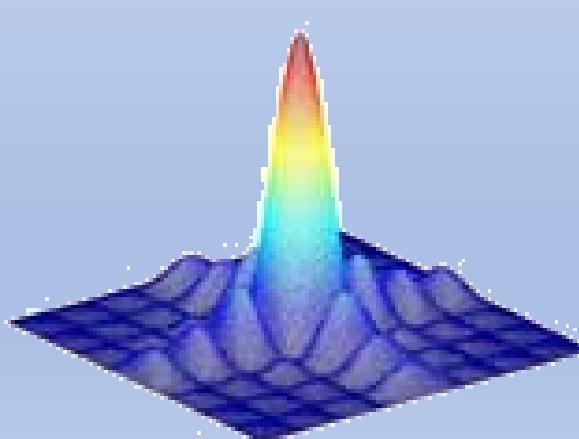
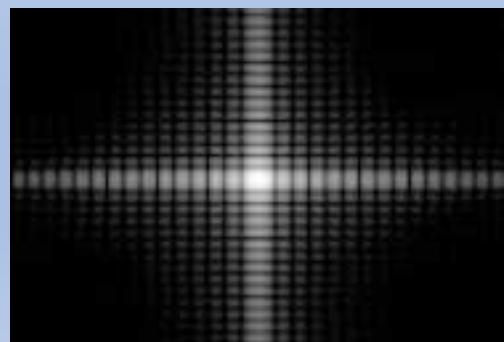
## 2. LSI Systems & Transforms—properties of DFT

➤ Examples:

$$f(x,y)$$



$$|F(u,v)|$$



## 2. LSI Systems & Transforms—properties of DFT

**TABLE 4.1**

Summary of some important properties of the 2-D Fourier transform.

Property	Expression(s)
Fourier transform	$F(u, v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-j2\pi(ux/M+vy/N)}$
Inverse Fourier transform	$f(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) e^{j2\pi(ux/M+vy/N)}$
Polar representation	$F(u, v) =  F(u, v)  e^{-j\phi(u, v)}$
Spectrum	$ F(u, v)  = [R^2(u, v) + I^2(u, v)]^{1/2}, \quad R = \text{Real}(F) \text{ and } I = \text{Imag}(F)$
Phase angle	$\phi(u, v) = \tan^{-1} \left[ \frac{I(u, v)}{R(u, v)} \right]$
Power spectrum	$P(u, v) =  F(u, v) ^2$
Average value	$\bar{f}(x, y) = F(0, 0) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y)$
Translation	$f(x, y) e^{j2\pi(u_0 x/M+v_0 y/N)} \Leftrightarrow F(u - u_0, v - v_0)$ $f(x - x_0, y - y_0) \Leftrightarrow F(u, v) e^{-j2\pi(ux_0/M+vy_0/N)}$ <p>When <math>x_0 = u_0 = M/2</math> and <math>y_0 = v_0 = N/2</math>, then</p> $f(x, y) (-1)^{x+y} \Leftrightarrow F(u - M/2, v - N/2)$ $f(x - M/2, y - N/2) \Leftrightarrow F(u, v) (-1)^{u+v}$

## 2. LSI Systems & Transforms—properties of DFT

Conjugate symmetry	$F(u, v) = F^*(-u, -v)$ $ F(u, v)  =  F(-u, -v) $
Differentiation	$\frac{\partial^n f(x, y)}{\partial x^n} \Leftrightarrow (ju)^n F(u, v)$ $(-jx)^n f(x, y) \Leftrightarrow \frac{\partial^n F(u, v)}{\partial u^n}$
Laplacian	$\nabla^2 f(x, y) \Leftrightarrow -(u^2 + v^2) F(u, v)$
Distributivity	$\Im[f_1(x, y) + f_2(x, y)] = \Im[f_1(x, y)] + \Im[f_2(x, y)]$ $\Im[f_1(x, y) \cdot f_2(x, y)] \neq \Im[f_1(x, y)] \cdot \Im[f_2(x, y)]$
Scaling	$af(x, y) \Leftrightarrow aF(u, v), f(ax, by) \Leftrightarrow \frac{1}{ ab } F(u/a, v/b)$
Rotation	$x = r \cos \theta \quad y = r \sin \theta \quad u = \omega \cos \varphi \quad v = \omega \sin \varphi$ $f(r, \theta + \theta_0) \Leftrightarrow F(\omega, \varphi + \theta_0)$
Periodicity	$F(u, v) = F(u + M, v) = F(u, v + N) = F(u + M, v + N)$ $f(x, y) = f(x + M, y) = f(x, y + N) = f(x + M, y + N)$
Separability	See Eqs. (4.6-14) and (4.6-15). Separability implies that we can compute the 2-D transform of an image by first computing 1-D transforms along each row of the image, and then computing a 1-D transform along each column of this intermediate result. The reverse, columns and then rows, yields the same result.

## 2. LSI Systems & Transforms—properties of DFT

Property	Expression(s)
Computation of the inverse Fourier transform using a forward transform algorithm	$\frac{1}{MN} f^*(x, y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F^*(u, v) e^{-j2\pi(ux/M+vy/N)}$ <p>This equation indicates that inputting the function <math>F^*(u, v)</math> into an algorithm designed to compute the forward transform (right side of the preceding equation) yields <math>f^*(x, y)/MN</math>. Taking the complex conjugate and multiplying this result by <math>MN</math> gives the desired inverse.</p>
Convolution <sup>†</sup>	$f(x, y) * h(x, y) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n)h(x - m, y - n)$
Correlation <sup>†</sup>	$f(x, y) \circ h(x, y) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f^*(m, n)h(x + m, y + n)$
Convolution theorem <sup>†</sup>	$f(x, y) * h(x, y) \Leftrightarrow F(u, v)H(u, v);$ $f(x, y)h(x, y) \Leftrightarrow F(u, v) * H(u, v)$
Correlation theorem <sup>†</sup>	$f(x, y) \circ h(x, y) \Leftrightarrow F^*(u, v)H(u, v);$ $f^*(x, y)h(x, y) \Leftrightarrow F(u, v) \circ H(u, v)$

## 2. LSI Systems & Transforms—some basic FT pairs

Some useful FT pairs:

$$\text{Impulse} \quad \delta(x, y) \Leftrightarrow 1$$

$$\text{Gaussian} \quad A\sqrt{2\pi}\sigma e^{-2\pi^2\sigma^2(x^2+y^2)} \Leftrightarrow Ae^{-(u^2+v^2)/2\sigma^2}$$

$$\text{Rectangle} \quad \text{rect}[a, b] \Leftrightarrow ab \frac{\sin(\pi ua)}{(\pi ua)} \frac{\sin(\pi vb)}{(\pi vb)} e^{-j\pi(ua+vb)}$$

$$\text{Cosine} \quad \cos(2\pi u_0 x + 2\pi v_0 y) \Leftrightarrow \frac{1}{2} [\delta(u + u_0, v + v_0) + \delta(u - u_0, v - v_0)]$$

$$\text{Sine} \quad \sin(2\pi u_0 x + 2\pi v_0 y) \Leftrightarrow j \frac{1}{2} [\delta(u + u_0, v + v_0) - \delta(u - u_0, v - v_0)]$$

<sup>†</sup> Assumes that functions have been extended by zero padding.

## 2. LSI Systems & Transforms—image sampling

- An digital image  $f_d(m,n)$  is obtained by sampling and quantizing a continuous analogy image  $f_c(x,y)$ . Here the sampling converts the continuous variable  $x, y$  into integer  $m, n$  and the quantization convert the continuous variable  $f_c$  into a finite set of numbers  $f_d$ . As the quantization has less impact than sampling to the image processing, we will focus more on the sampling process and theory.
- Given a continuous analogy image  $f_c(x,y)$ , it is **very simple** to get its discrete image  $f_d(m,n)$  mathematically by

$$f_d(m, n) = f_c(m\Delta x, n\Delta y) = f_c(x, y) \Big|_{\text{Let } x=m\Delta x, y=n\Delta y}$$

- Does  $f_d(m,n)$  contain the same information as  $f_c(x,y)$ ? Under what conditions is it yes? Mathematically analyzing this, however, needs abstract mathematical tools.

## 2. LSI Systems & Transforms—image sampling

- A 2-D function  $f_c(x,y)$  is band-limited if its Fourier transform  $F_c(u,v)$  is zero outside a bounded spatial frequency support; e.g.,

$$F_c(u, v) = 0, \quad \text{for } |u| > U_0, |v| > V_0$$

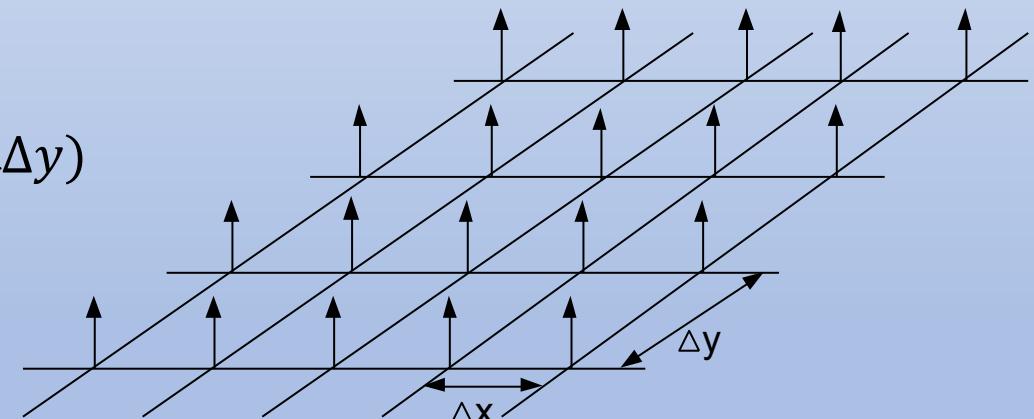
- where  $2U_0$  and  $2V_0$  are referred to as the  $x$  and  $y$  bandwidths of the 2-D function.  
**Why  $2U_0$  and  $2V_0$ ?**

- In practice, real-world images can be well **approximated** by band-limited signals.
- Define a 2-D **sampling function**

$$s(x, y) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \delta(x - m\Delta x, y - n\Delta y)$$

- It has the Fourier transform of

$$S(u, v) = \frac{1}{\Delta x \Delta y} \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \delta(u - m/\Delta x, v - n/\Delta y)$$



## 2. LSI Systems & Transforms—image sampling

- Multiply the continuous image  $f_c(x,y)$  with the **sampling image**  $s(x,y)$  yield

$$f_d(x,y) = f_c(x,y)s(x,y) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f_c(m\Delta x, n\Delta y) \delta(x - m\Delta x, y - n\Delta y)$$

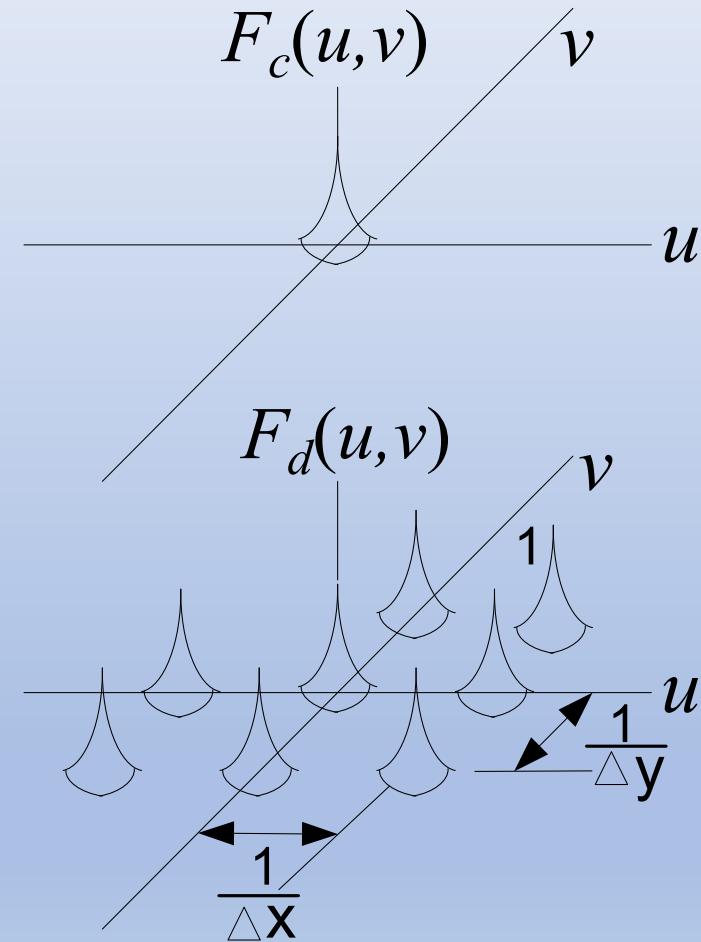
- The Fourier transform of the sampled image  $f_d(x,y)$  in continuous domain is:

$$\begin{aligned} F_d(u,v) &= F_c(u,v) * S(u,v) = \frac{1}{\Delta x \Delta y} \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} F_c(u,v) * \delta(u - m/\Delta x, v - n/\Delta y) \\ &= \frac{1}{\Delta x \Delta y} \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} F_c(u - m/\Delta x, v - n/\Delta y) = \frac{1}{\Delta x \Delta y} \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} F_c(u - mf_{xs}, v - nf_{ys}) \end{aligned}$$

- It is a periodic replication of  $F_c(u,v)$ , on a rectangular grid with spacing  $(1/\Delta x, 1/\Delta y)$ .

## 2. LSI Systems & Transforms—image sampling

- The spectrum of the sampled image  $f_d(x,y)$  consists of the spectrum of the continuous  $f_c(x,y)$  image (top) infinitely repeated over the frequency plane in a rectangular grid with spacing  $(1/\Delta x, 1/\Delta y)$ .
- It is a periodic replication of  $F_c(u,v)$ , on a rectangular grid with spacing  $(1/\Delta x, 1/\Delta y)$ .



## 2. LSI Systems & Transforms—image sampling

- If the  $x, y$  sampling frequencies are greater than the bandwidths, or if the sampling intervals are smaller than the reciprocal of bandwidths, i.e.,

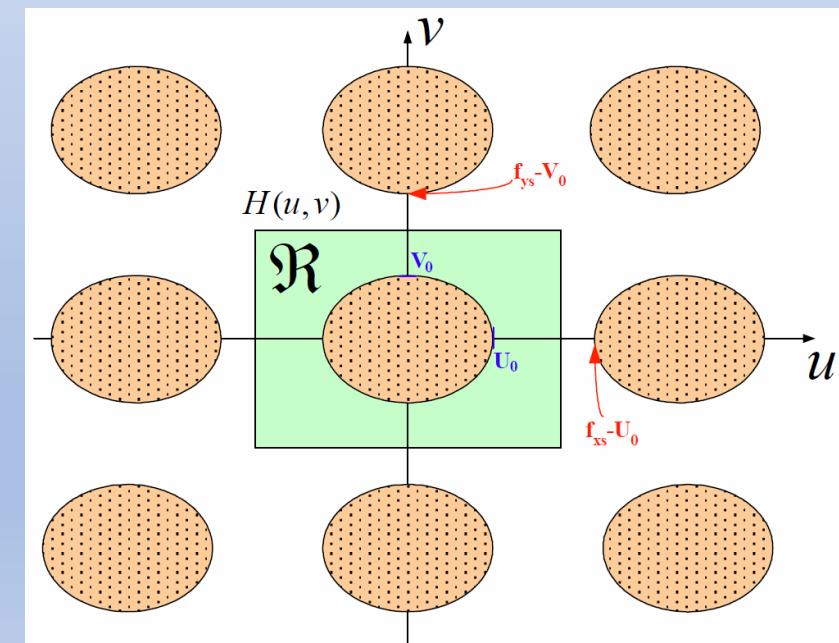
$$f_{xs} = \frac{1}{\Delta x} \geq 2U_0 \quad \& \quad f_{ys} = \frac{1}{\Delta y} \geq 2V_0 \quad \text{or} \quad \Delta x \leq \frac{1}{2U_0} \quad \& \quad \Delta y \leq \frac{1}{2V_0}$$

- Then  $F_c(u, v)$  can be recovered from  $F_d(u, v)$  by using a low-pass filter with frequency response

$$H(u, v) = \begin{cases} \Delta x \Delta y, & (u, v) \in \mathcal{R} \\ 0, & \text{otherwise} \end{cases}$$

- That is

$$\begin{aligned} F_c(u, v) &= F_d(u, v) H(u, v) \\ f_c(x, y) &= f_d(x, y) * h(x, y) \end{aligned}$$



## 2. LSI Systems & Transforms—image sampling

$$\begin{aligned}f_c(x, y) &= h(x, y) * f_d(x, y) \\&= h(x, y) * \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f_c(m\Delta x, n\Delta y) \delta(x - m\Delta x, y - n\Delta y) \\&= \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f_c(m\Delta x, n\Delta y) h(x, y) * \delta(x - m\Delta x, y - n\Delta y) \\&= \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f_c(m\Delta x, n\Delta y) h(x - m\Delta x, y - n\Delta y) \\&= \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f_d(m, n) h(x - m\Delta x, y - n\Delta y)\end{aligned}$$

- Therefore, all information of the continuous image  $f_c(x, y)$  can be recovered from the discrete image  $f_d(m, n)$ .

## 2. LSI Systems & Transforms—image sampling

### Sampling Theorem:

- A band-limited image  $f_c(x,y)$  with bandwidths  $(2U_0, 2V_0)$  sampled uniformly on a rectangular grid with spacing  $(\Delta x, \Delta y)$  can be recovered without error from the sampled values  $f_c(m\Delta x, n\Delta y) = f_d(m, n)$  provided that the sampling rates  $(f_{xs}, f_{ys})$  are greater than the Nyquist rates.
- The lower bounds of the required sampling rates, the band width, are known as the Nyquist rates or Nyquist frequencies. Their reciprocals are known as the Nyquist intervals.
- Sampling below the Nyquist rates will cause the periodic replications of  $F_c(u,v)$  to overlap, resulting in a distorted spectrum  $F_d(u,v)$ , in which  $F_c(u,v)$  is irrevocably lost—a phenomenon that is known as aliasing.
- Aliasing can be avoided or reduced by low-pass filtering the image  $f_c(x,y)$  before sampling so that its bandwidth is less than the sampling frequency. (at the expense of what?)

### 3. Image Enhancement—outline

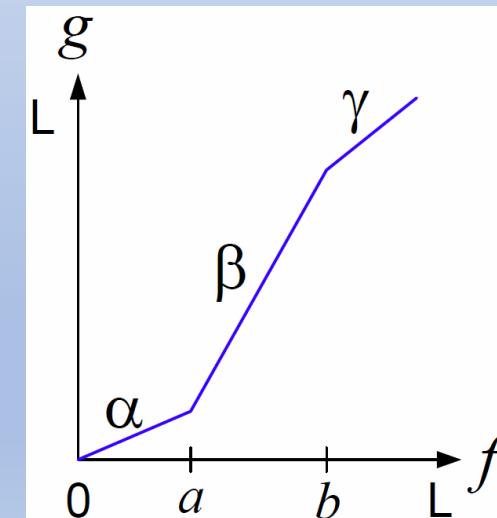
- Simple Point Processing Approaches
- Histogram Equalization
- Image Smoothing
- Image Sharpening
- Nonlinear Image Processing

### 3. Image Enhancement—point processing

- Image processing operations are designed to enhance image content or features so that they are more suitable for display or analysis.
- Many image enhancement processes are **point and memoryless** operations which map input image gray-level to output gray-level according to a transformation  $g=T(f)$ . For example:
  - Power Transformation (gamma correction):  $g=c f^\gamma$
  - Log Transformation:  $g=c \log(1+f)$
  - Piecewise Linear Transformation

$$g = T(f) = \begin{cases} \alpha f, & 0 \leq f < a \\ \beta(f - a) + T(a), & a \leq f < b \\ \gamma(f - b) + T(b), & b \leq f < L \end{cases}$$

- Histogram equalization



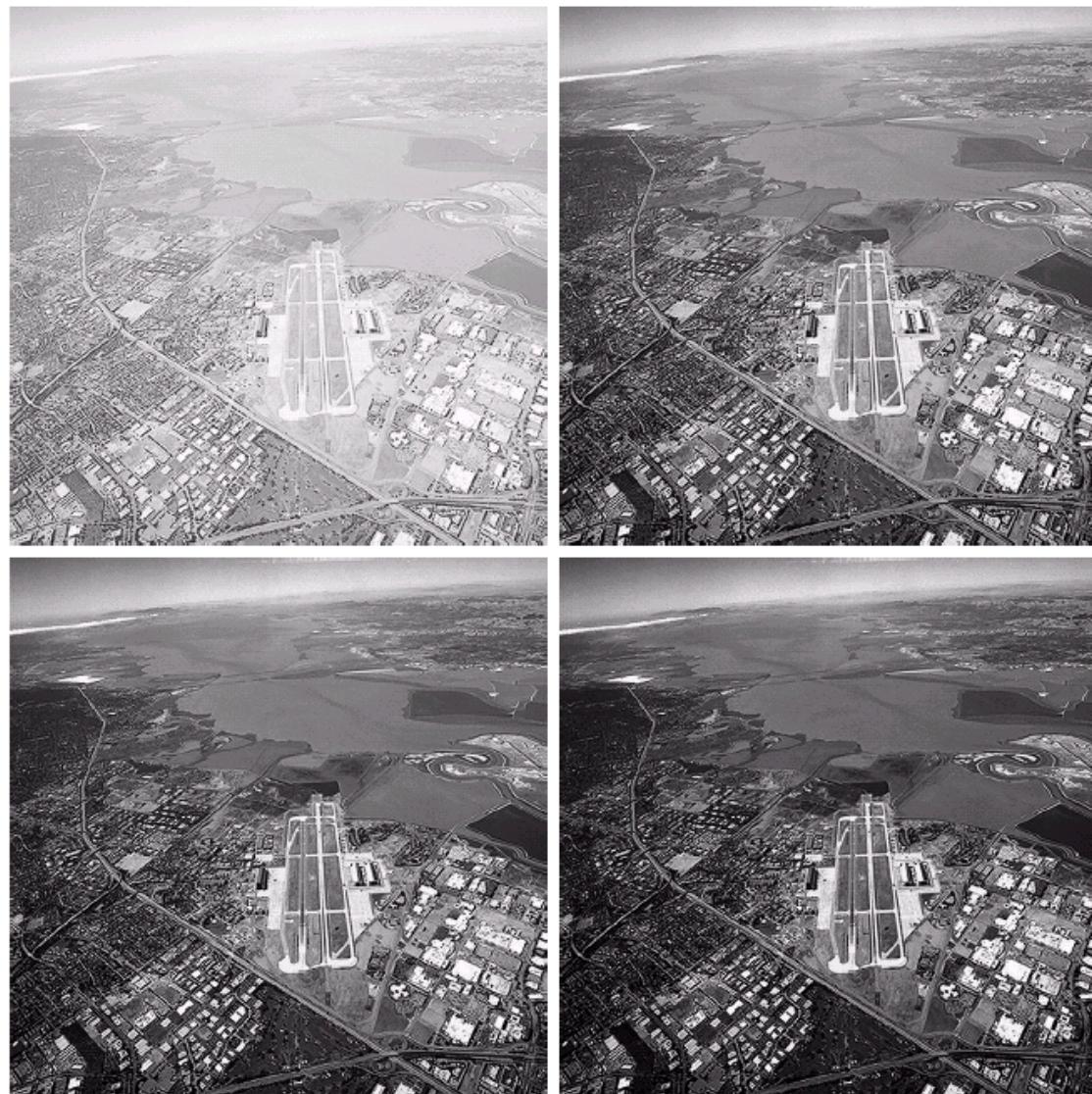
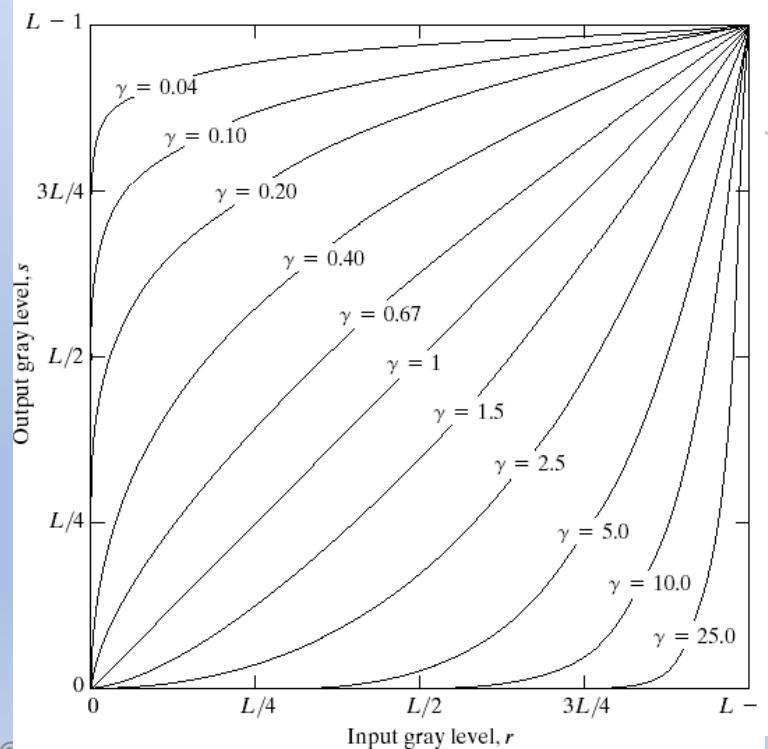
# 3. Image Enhancement—simple point processing

Power  
Transformation  
gamma correction  
 $g = c f^\gamma$

a b  
c d

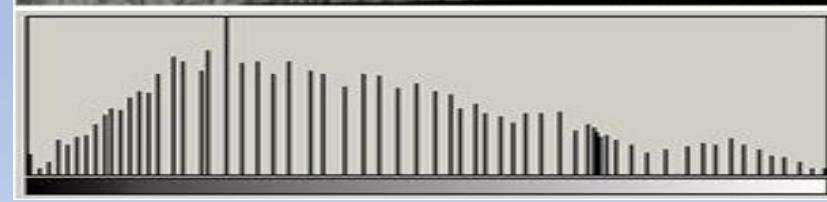
FIGURE 3.9

(a) Aerial image.  
(b)–(d) Results of applying the transformation in Eq. (3.2-3) with  $c = 1$  and  $\gamma = 3.0, 4.0$ , and  $5.0$ , respectively.



# 3. Image Enhancement—simple point processing

Log Transformation:  $g=c\log(1+f)$

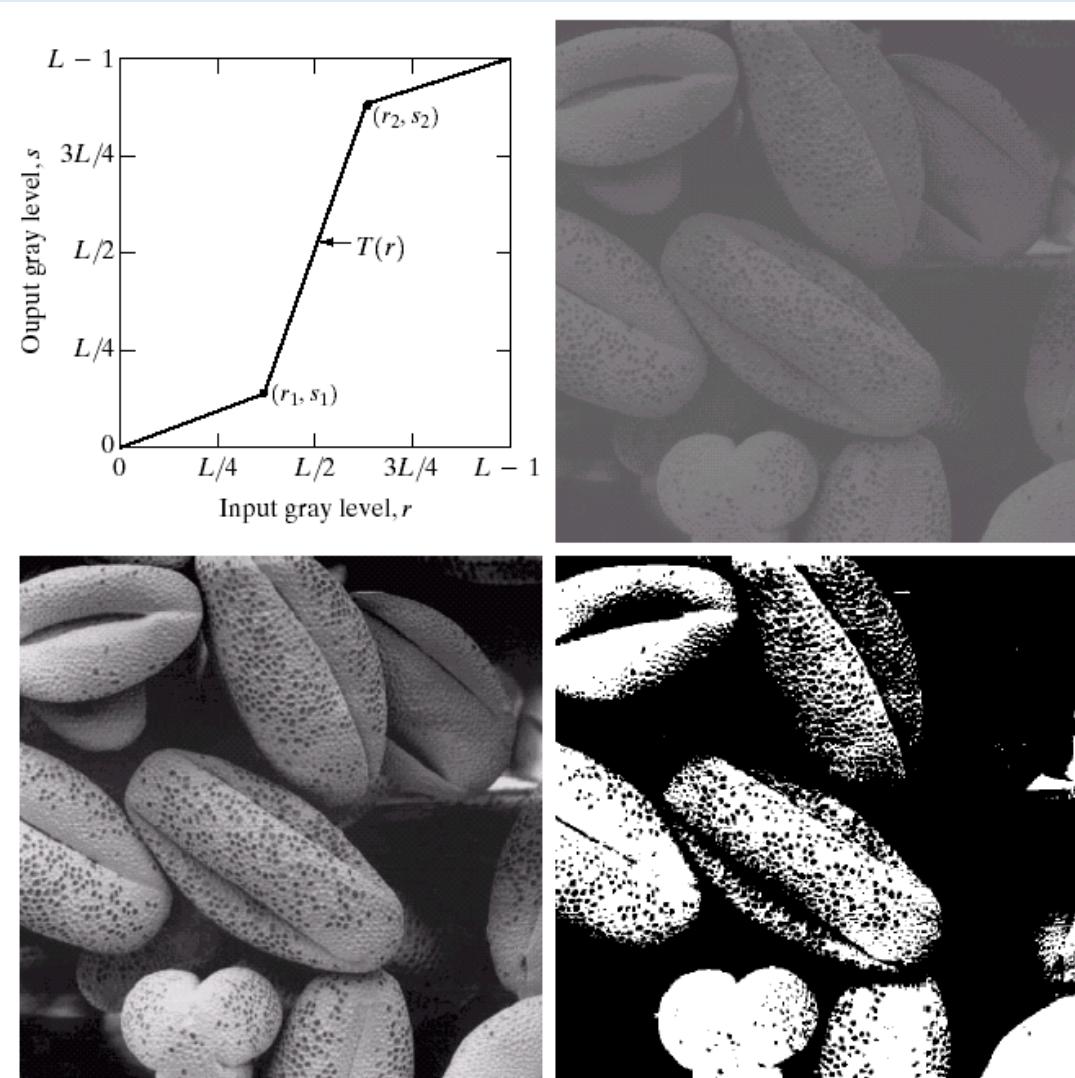


# 3. Image Enhancement—simple point processing

Piecewise Linear Transformation:

Contrast stretching:

$$g = T(f)$$
$$= \begin{cases} \alpha f, & 0 \leq f < a \\ \beta(f - a) + T(a), & a \leq f < b \\ \gamma(f - b) + T(b), & b \leq f < L \end{cases}$$



### 3. Image Enhancement—histogram equalization

- Histogram equalization aims to obtain a uniform histogram for the output image  $g(x,y)$  by transforming the gray-level  $f$  of the input image  $f(x,y)$  into  $g$
- Histogram equalization algorithm:  $g = T(f)$

$$c(f) = \sum_{t=0}^f p_f(t) = \sum_{t=0}^f \frac{n_t}{n},$$

$$g = T(f) = \text{round} \left[ \frac{c(f) - c_{\min}}{1 - c_{\min}} L \right],$$

- where  $t$  is a dummy variable of the summation.  $c_{\min}$  is the smallest value of all  $c(f)$ , `round`[ ] rounds a real number to an integer.  $g$  is approximately uniformly distributed in  $[0, L]$ .

### 3. Image Enhancement—histogram equalization

- Theoretical analysis of the histogram equalization can only be done for continuous variable.
- Let  $f$  be the continuous gray value normalize to  $[0,1]$
- Let the transform  $g=T(f)$  single-valued, monotonically increasing in  $0 \leq g=T(f) \leq 1$ .
- The inverse transform is  $f=T^{-1}(g)$  should also be single-valued and monotonically increasing.
- From **probability theory**, if original gray level pdf  $p_f(f)$  and  $T(f)$  are known,  $T^{-1}(g)$  satisfies the above condition, then the transformed gray level pdf  $p_g(g)$  is

$$p_g(g) = p_f(f) \frac{df}{dg}$$

### 3. Image Enhancement—histogram equalization

- Consider that  $g = T(f) = \int_0^f p_f(t)dt$
- The above function is the cumulative distribution function (cdf) of  $f$ . cdf is single-valued and monotonically increasing.

$$\because \frac{dg}{df} = p_f(f) \quad \therefore p_g(g) = p_f(f) \frac{df}{dg} = p_f(f) \frac{1}{p_f(f)} = 1$$

- Therefore, the transformed gray value has **uniform distribution**.
- The histogram equalization
- is the discrete version of

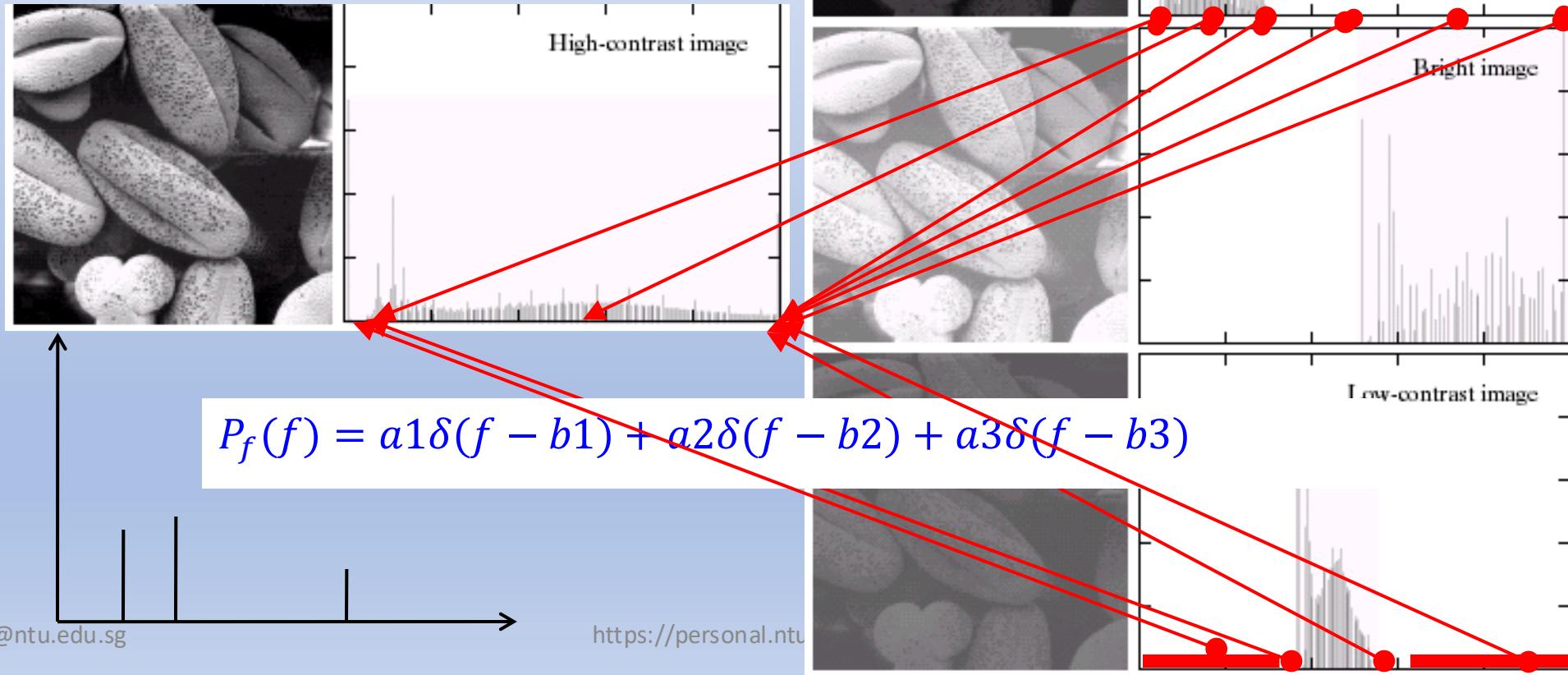
$$c(f) = \sum_{t=0}^f p_f(t) = \sum_{t=0}^f \frac{n_t}{n}$$

$$g = T(f) = \int_0^f p_f(t)dt$$

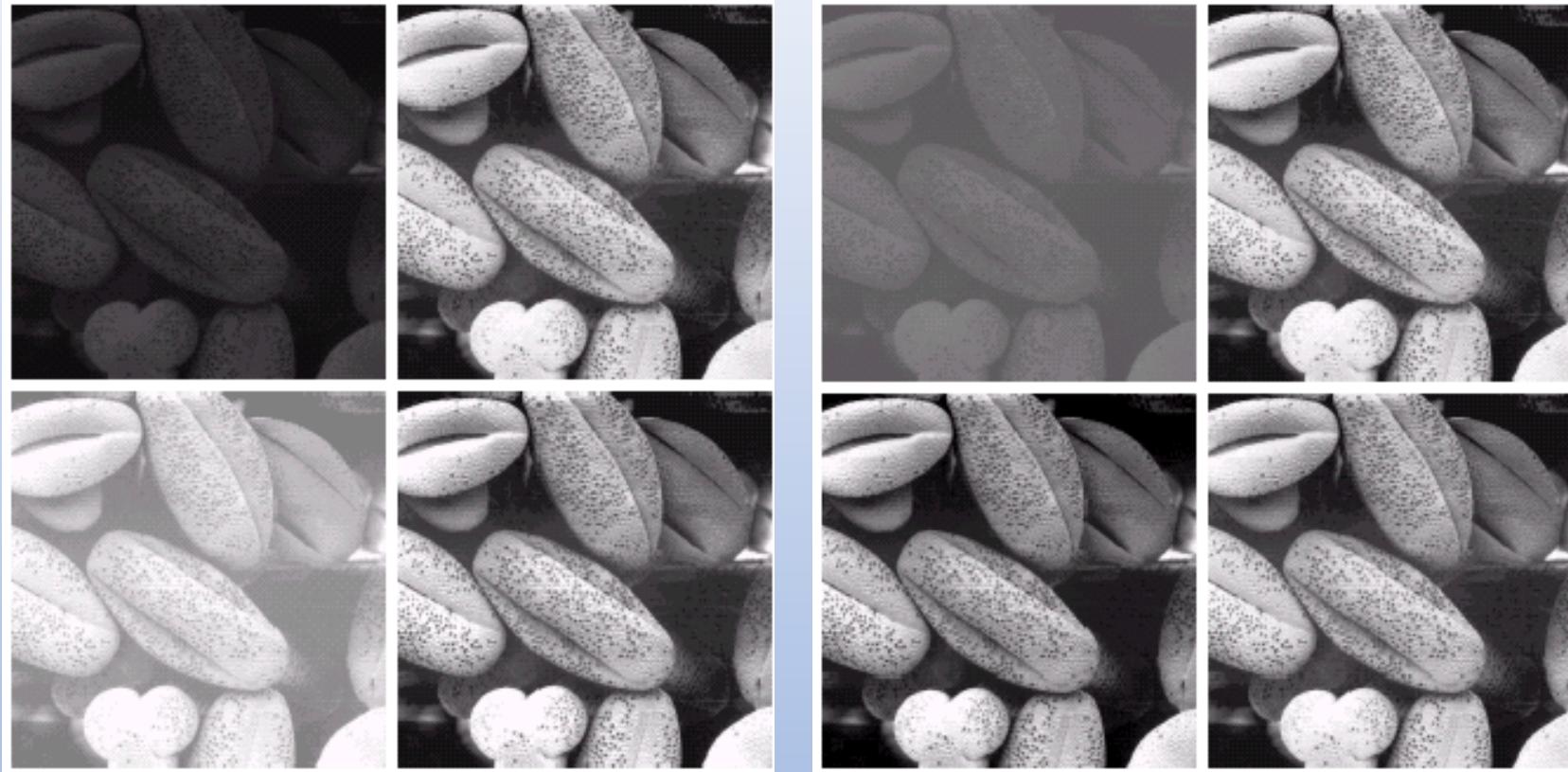
### 3. Image Enhancement—histogram equalization

Understand the histogram equalization and its effect on image.

$$c(f) = \sum_{t=0}^f p_f(t) = \sum_{t=0}^f \frac{n_t}{n}$$



### 3. Image Enhancement—histogram equalization

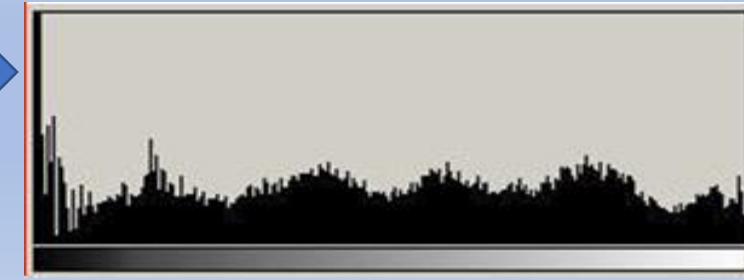


J. Ren, X. Jiang and J. Yuan, “[A Chi-Squared-Transformed Subspace of LBP Histogram for Visual Recognition](#),” *IEEE Trans. Image Processing*, vol. 24, no. 6, pp. 1893-1904, June, 2015.

### 3. Image Enhancement—histogram equalization

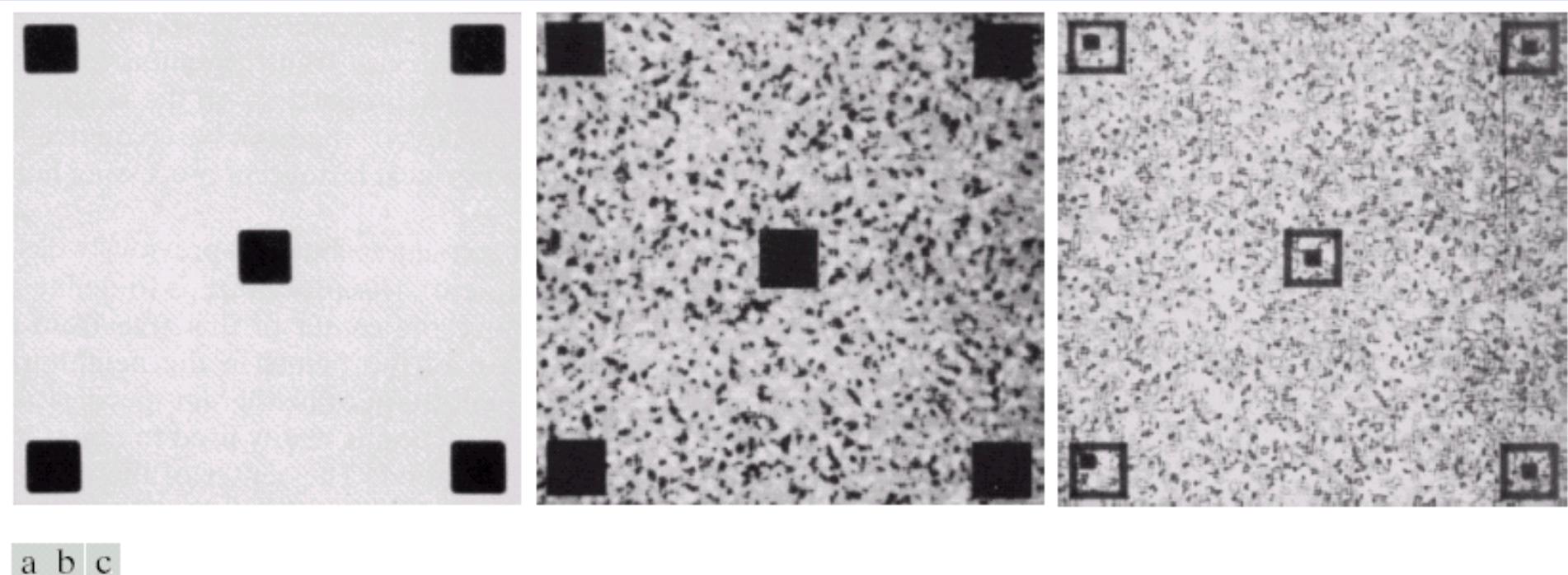


Histogram  
Equalization



# 3. Image Enhancement—histogram equalization

## ➤ Local histogram equalization



**FIGURE 3.23** (a) Original image. (b) Result of global histogram equalization. (c) Result of local histogram equalization using a  $7 \times 7$  neighborhood about each pixel.

### 3. Image Enhancement—image smoothing

- The basic form of linear image filtering in the spatial and frequency domain can be given as

$$\begin{aligned} g(x, y) &= f(x, y) * h(x, y) \\ &\Updownarrow \quad \Updownarrow \quad \Updownarrow \\ G(u, v) &= F(u, v)H(u, v) \end{aligned}$$

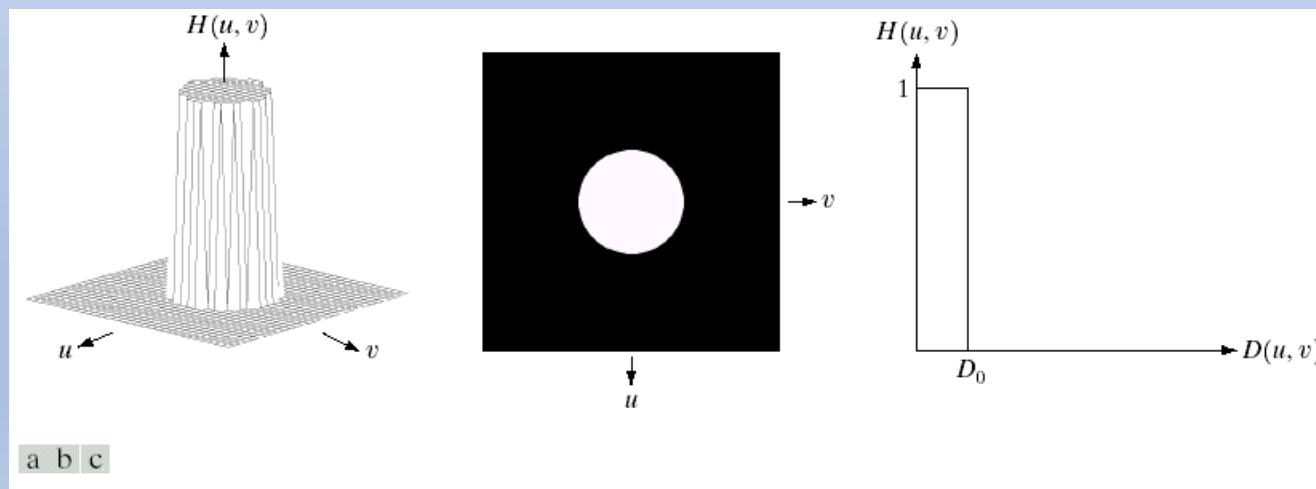
$$\begin{aligned} g(x, y) &= f(x, y) * h(x, y) = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} h(i, j)f(x - i, y - j) \\ &= \sum_{j=-3}^{3} \sum_{i=-3}^{3} h(i, j)f(x - i, y - j), \quad \text{if } h(x, y) = 0 \text{ for } -3 < x, y < 3 \end{aligned}$$

- At each point  $(x, y)$  the response of the filter at that point is calculated as a sum of products of the filter coefficients and the corresponding image pixels in the area spanned by the filter mask centered at  $(x, y)$ .

### 3. Image Enhancement—image smoothing

- Image smoothing filters are used for blurring and for noise reduction.
- These filters are also known as averaging or low pass filters.
- Ideal low-pass filter

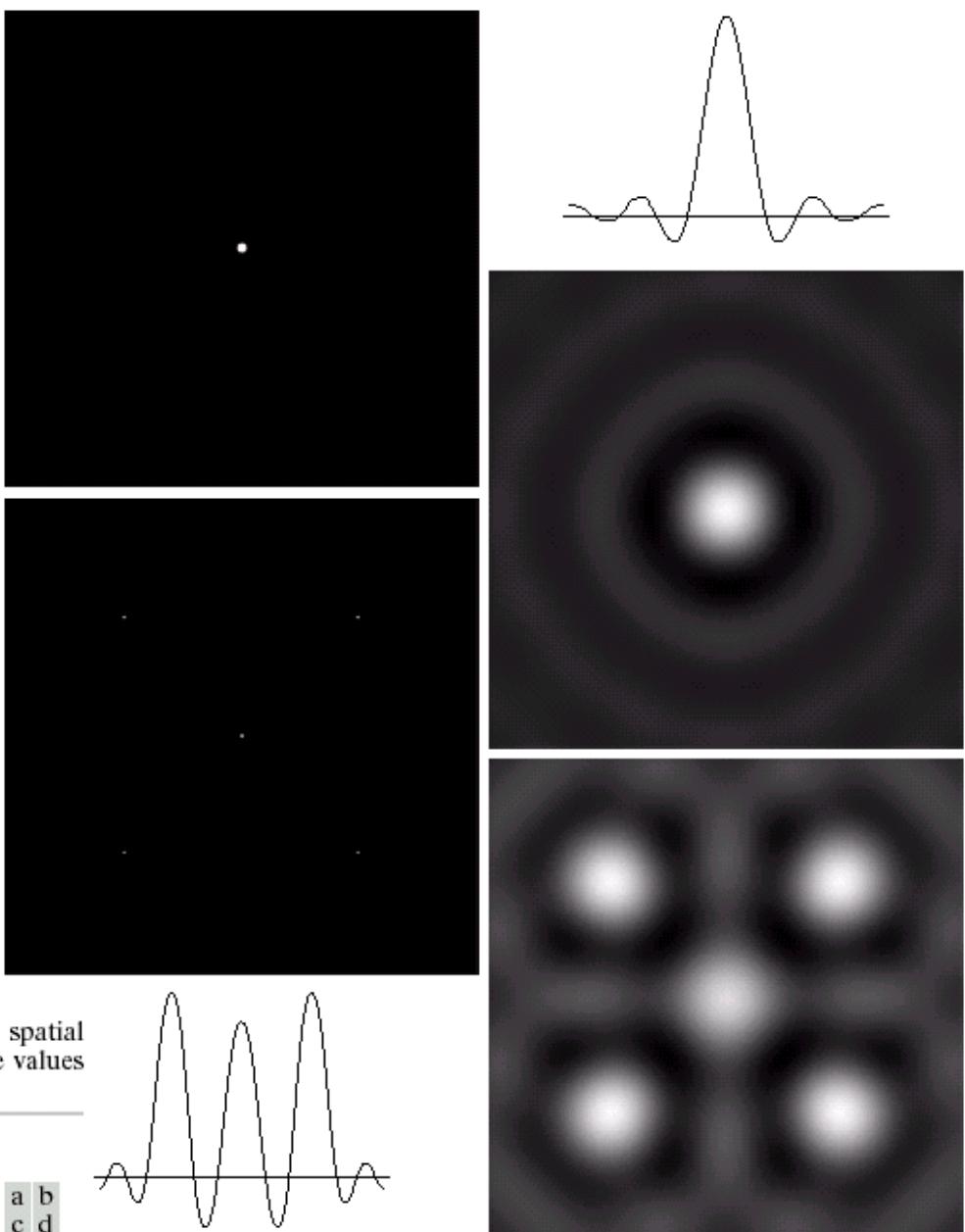
$$H(u, v) = \begin{cases} 1, & \text{if } D(u, v) \leq D_0 \\ 0, & \text{if } D(u, v) > D_0 \end{cases} \quad D(u, v) = \sqrt{u^2 + v^2}$$



**FIGURE 4.10** (a) Perspective plot of an ideal lowpass filter transfer function. (b) Filter displayed as an image. (c) Filter radial cross section.

### 3. Image Enhancement—image smoothing

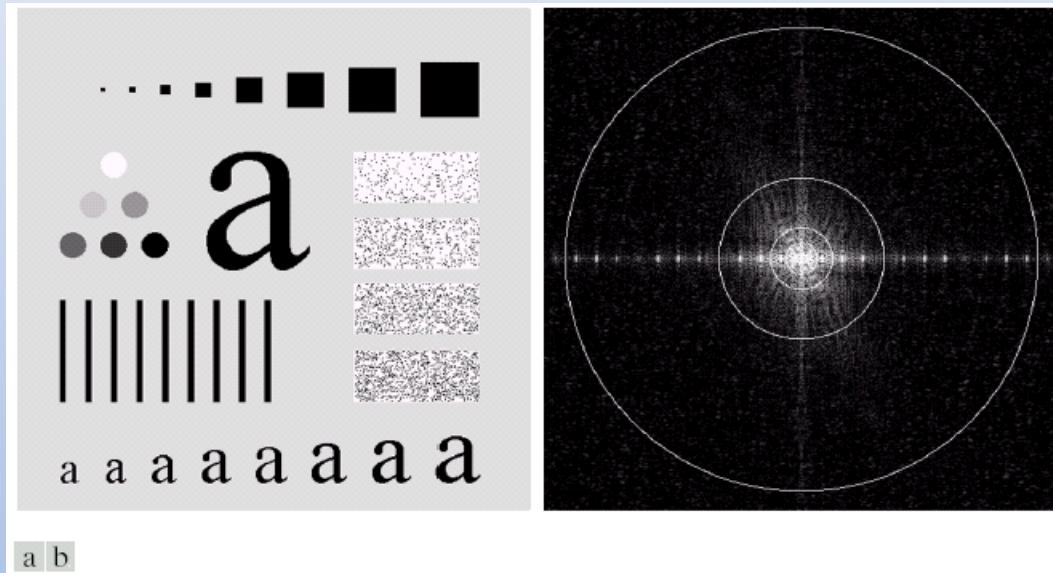
- Problems of Ideal low-pass filtering



**FIGURE 4.13** (a) A frequency-domain ILPF of radius 5. (b) Corresponding spatial filter (note the ringing). (c) Five impulses in the spatial domain, simulating the values of five pixels. (d) Convolution of (b) and (c) in the spatial domain.

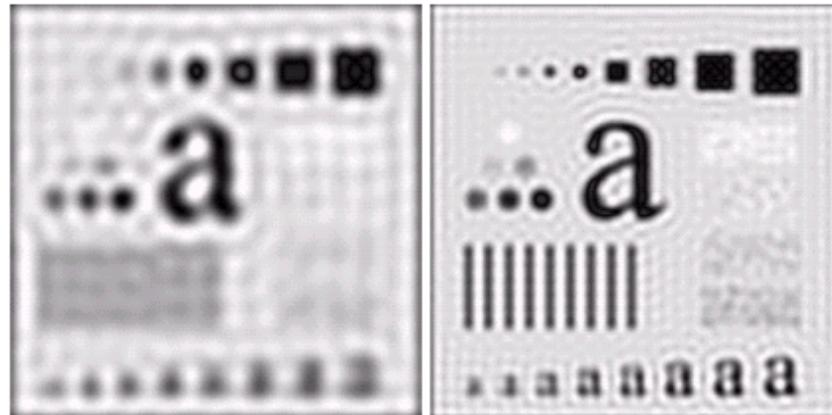
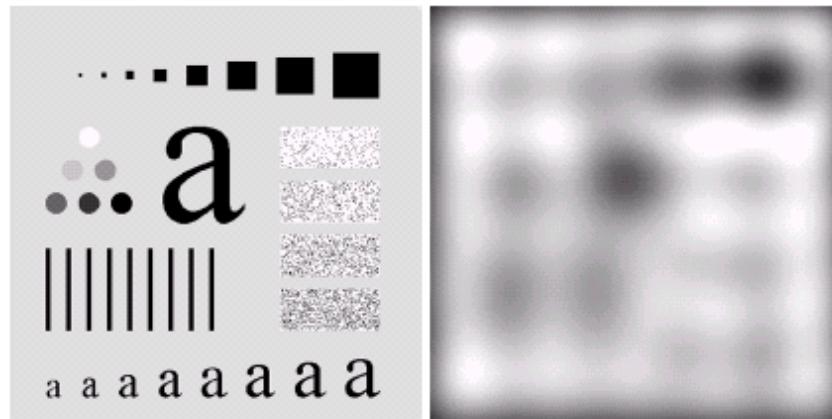
# 3. Image Enhancement—image smoothing

- Ideal low-pass filtering



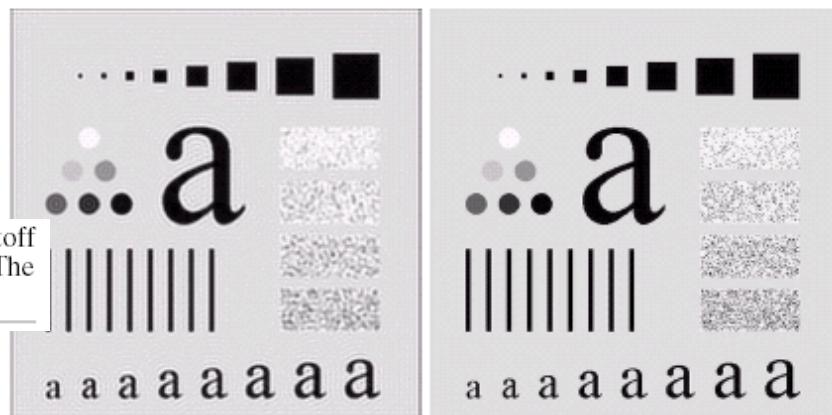
a b

**FIGURE 4.11** (a) An image of size  $500 \times 500$  pixels and (b) its Fourier spectrum. The superimposed circles have radii values of 5, 15, 30, 80, and 230, which enclose 92.0, 94.6, 96.4, 98.0, and 99.5% of the image power, respectively.



a b  
c d  
e f

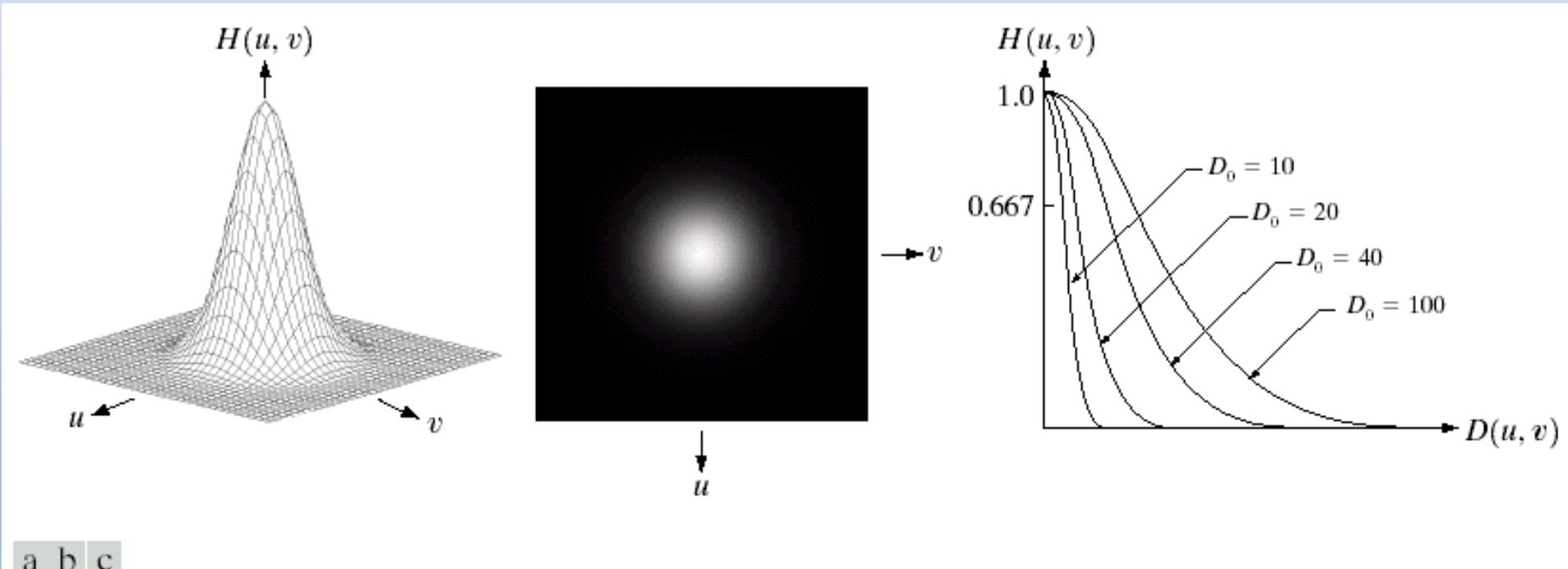
**FIGURE 4.12** (a) Original image. (b)–(f) Results of ideal lowpass filtering with cutoff frequencies set at radii values of 5, 15, 30, 80, and 230, as shown in Fig. 4.11(b). The power removed by these filters was 8, 5.4, 3.6, 2, and 0.5% of the total, respectively.



### 3. Image Enhancement—image smoothing

- Gaussian low-pass filtering (GLPF)

$$G(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2D_0}}$$



**FIGURE 4.17** (a) Perspective plot of a GLPF transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections for various values of  $D_0$ .

### 3. Image Enhancement—image smoothing

- Gaussian low-pass filtering  
(GLPF)

$$G(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2D_0}}$$

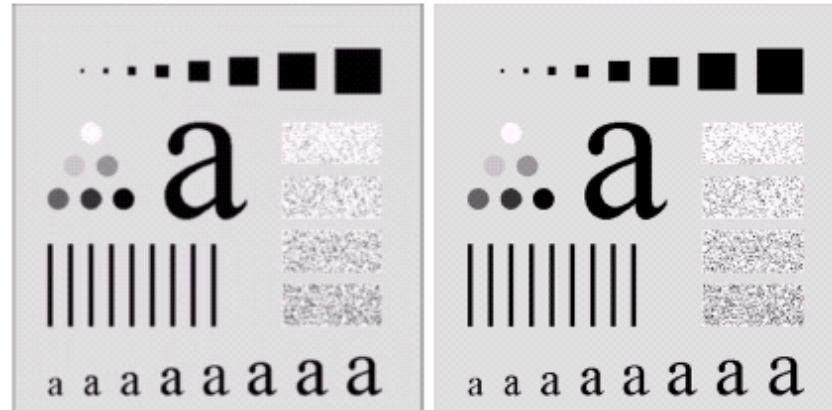
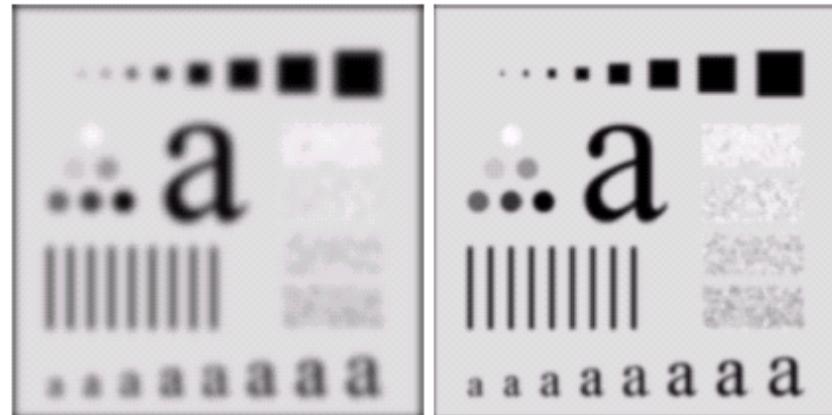


FIGURE 4.18 (a) Original image. (b)–(f) Results of filtering with Gaussian lowpass filters with cutoff frequencies set at radii values of 5, 15, 30, 80, and 230, as shown in Fig. 4.11(b). Compare with Figs. 4.12 and 4.15.

a b  
c d  
e f

### 3. Image Enhancement—image smoothing



a b c

**FIGURE 4.21** (a) Image showing prominent scan lines. (b) Result of using a GLPF with  $D_0 = 30$ . (c) Result of using a GLPF with  $D_0 = 10$ . (Original image courtesy of NOAA.)

### 3. Image Enhancement—image smoothing



**FIGURE 4.20** (a) Original image ( $1028 \times 732$  pixels). (b) Result of filtering with a GLPF with  $D_0 = 100$ . (c) Result of filtering with a GLPF with  $D_0 = 80$ . Note reduction in skin fine lines in the magnified sections of (b) and (c).

# 3. Image Enhancement—image sharpening

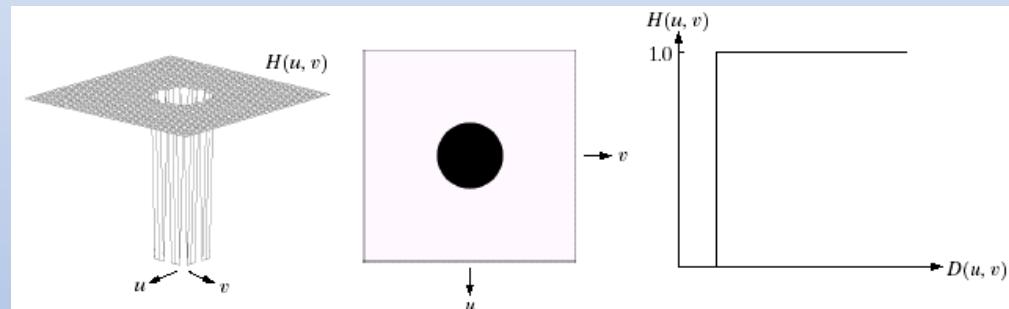
- High-pass filtering

$$H_{hp}(u, v) = 1 - H_{lr}(u, v)$$

- Ideal high-pass filter

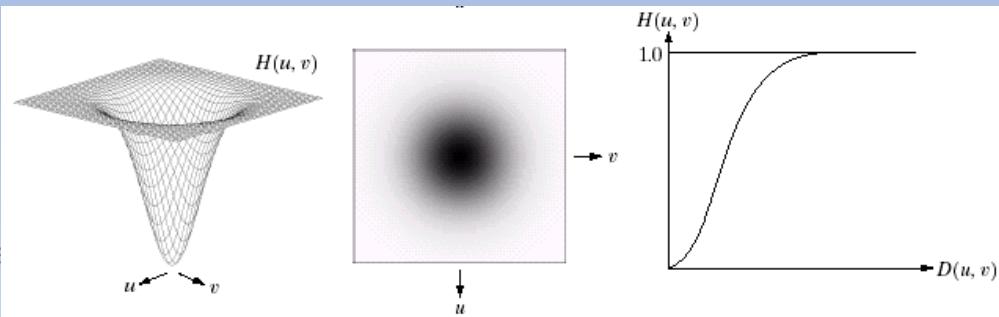
$$H(u, v) = \begin{cases} 0, & \text{if } D(u, v) \leq D_0 \\ 1, & \text{if } D(u, v) > D_0 \end{cases}$$

$$D(u, v) = \sqrt{u^2 + v^2}$$



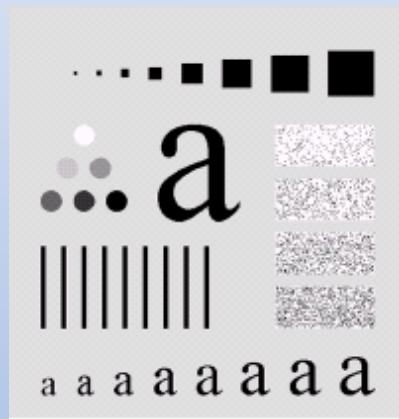
- Gaussian high-pass filter (GHPF)

$$G(u, v) = 1 - e^{-\frac{u^2+v^2}{2D_0}}$$

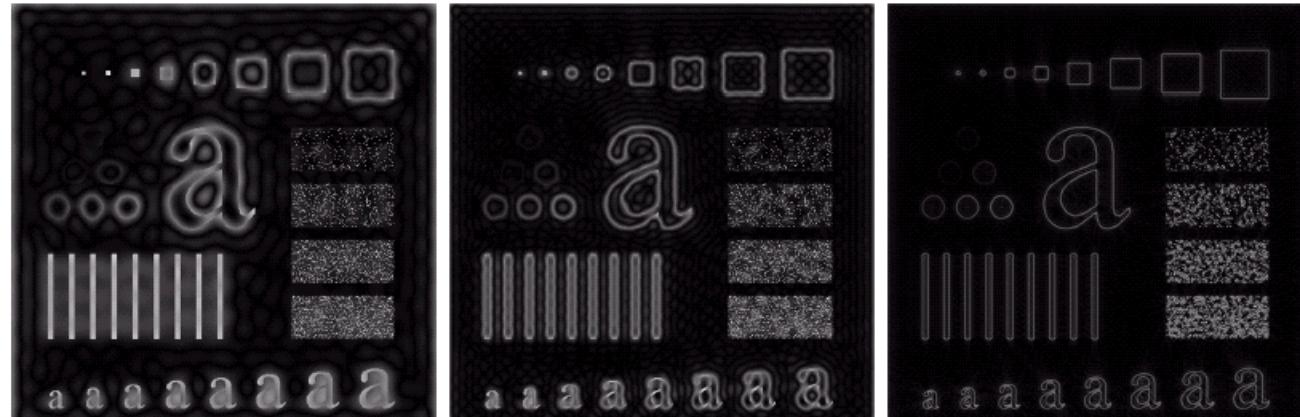


# 3. Image Enhancement—image sharpening

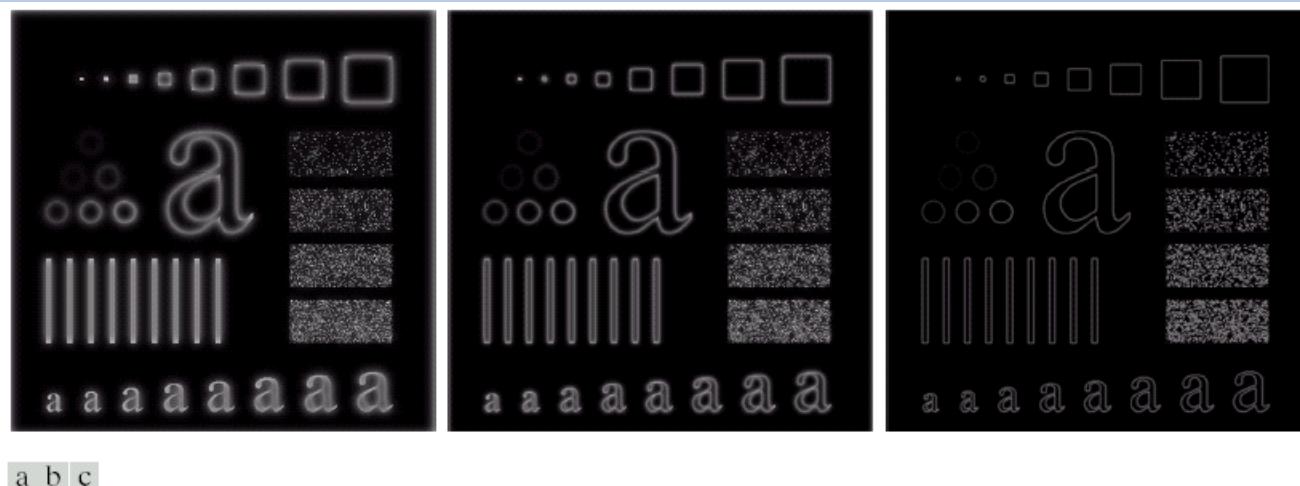
Ideal high-pass filter



Gaussian High-pass filt



**FIGURE 4.24** Results of ideal highpass filtering the image in Fig. 4.11(a) with  $D_0 = 15, 30$ , and  $80$ , respectively. Problems with ringing are quite evident in (a) and (b).



**FIGURE 4.26** Results of highpass filtering the image of Fig. 4.11(a) using a GHPF of order 2 with  $D_0 = 15, 30$ , and  $80$ , respectively. Compare with Figs. 4.24 and 4.25.

### 3. Image Enhancement—image sharpening

- High-boost filtering

$$f_{hb}(x, y) = Af(x, y) - f_{lp}(x, y)$$

where  $f_{lp}(x, y)$  is a smoothed version of  $f(x, y)$

by a lowpass filter,

$$A \geq 1$$

$$f_{hb}(x, y) = (A - 1)f(x, y) + f(x, y) - f_{lp}(x, y)$$

$$f_{hb}(x, y) = (A - 1)f(x, y) + f_{hp}(x, y)$$

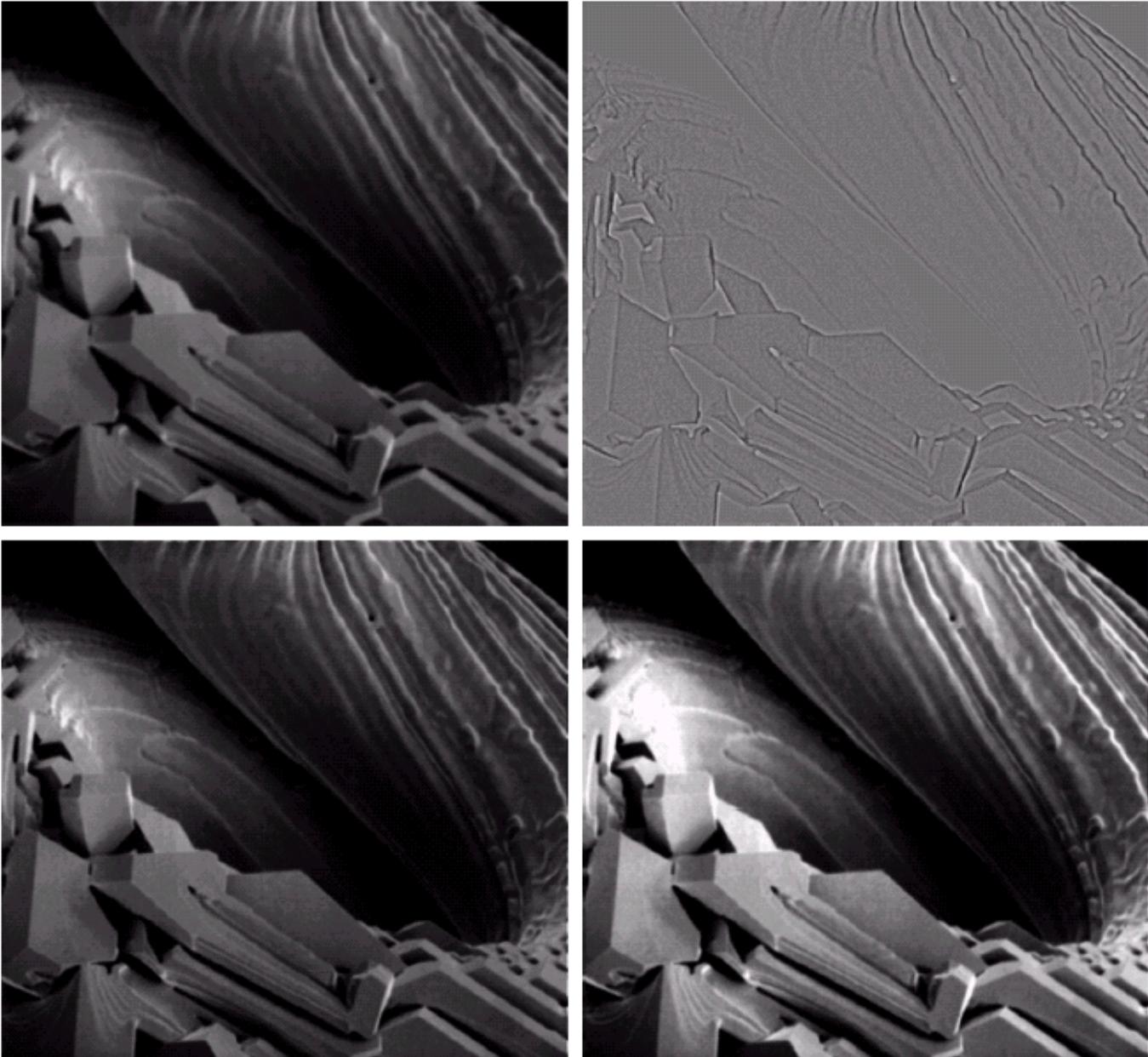
### 3. Image Enhancement—image sharpening

#### High-boost filtering

a b  
c d

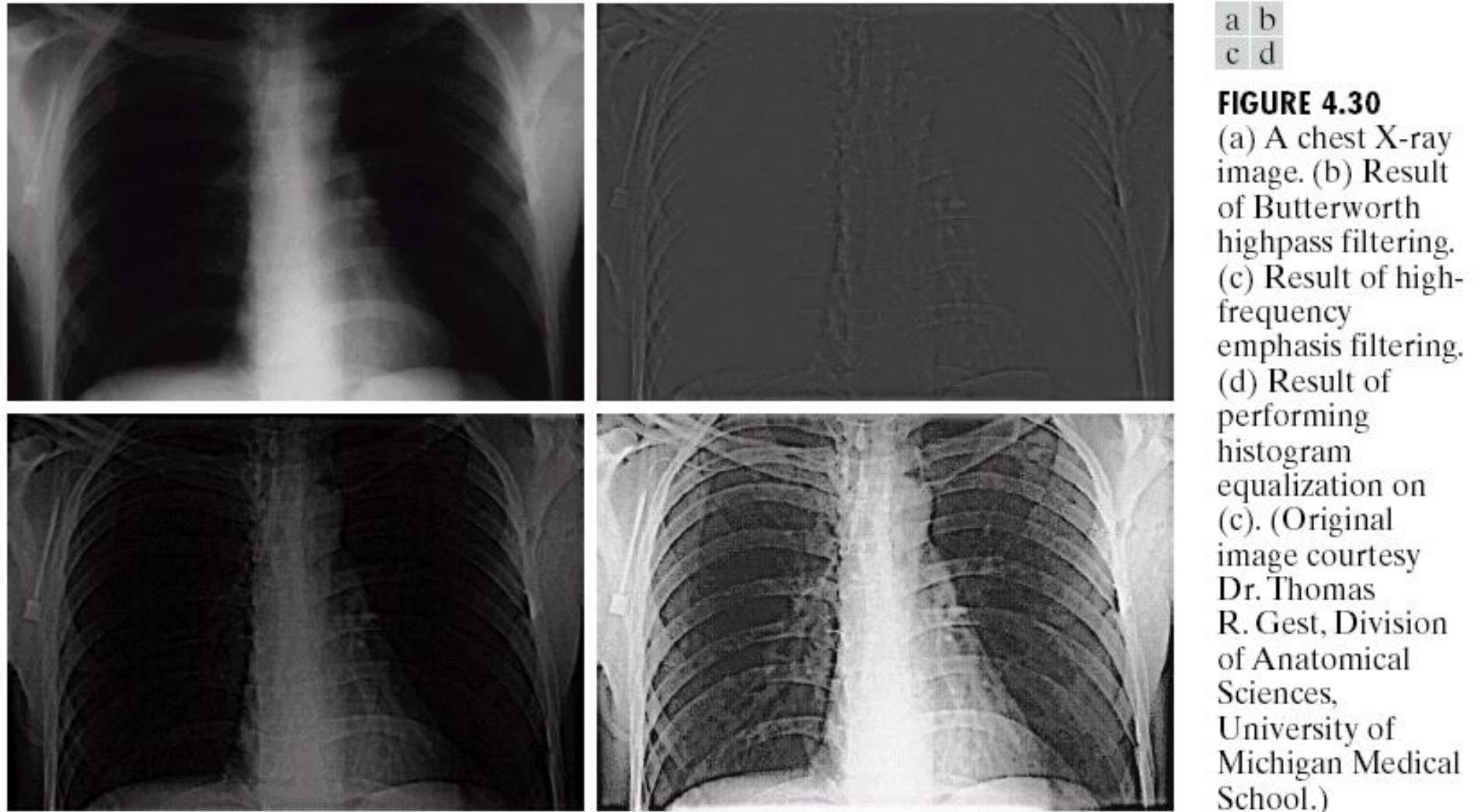
**FIGURE 4.29**

Same as Fig. 3.43, but using frequency domain filtering. (a) Input image.  
(b) Laplacian of (a). (c) Image obtained using Eq. (4.4-17) with  $A = 2$ . (d) Same as (c), but with  $A = 2.7$ . (Original image courtesy of Mr. Michael Shaffer, Department of Geological Sciences, University of Oregon, Eugene.)



# 3. Image Enhancement—image sharpening

## Examples of combining image enhancement

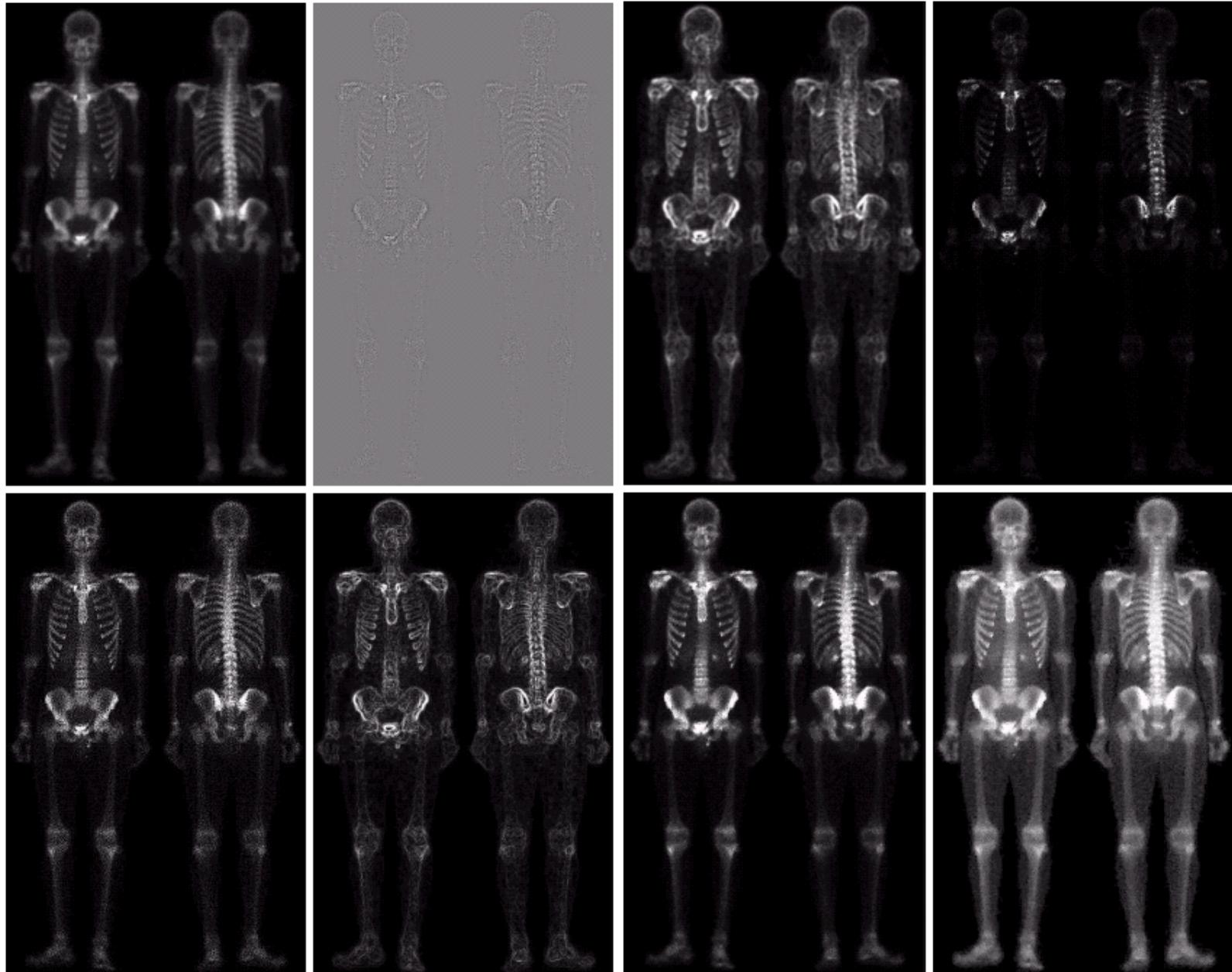


a  
b  
c  
d

**FIGURE 4.30**  
(a) A chest X-ray image. (b) Result of Butterworth highpass filtering. (c) Result of high-frequency emphasis filtering. (d) Result of performing histogram equalization on (c). (Original image courtesy Dr. Thomas R. Gest, Division of Anatomical Sciences, University of Michigan Medical School.)

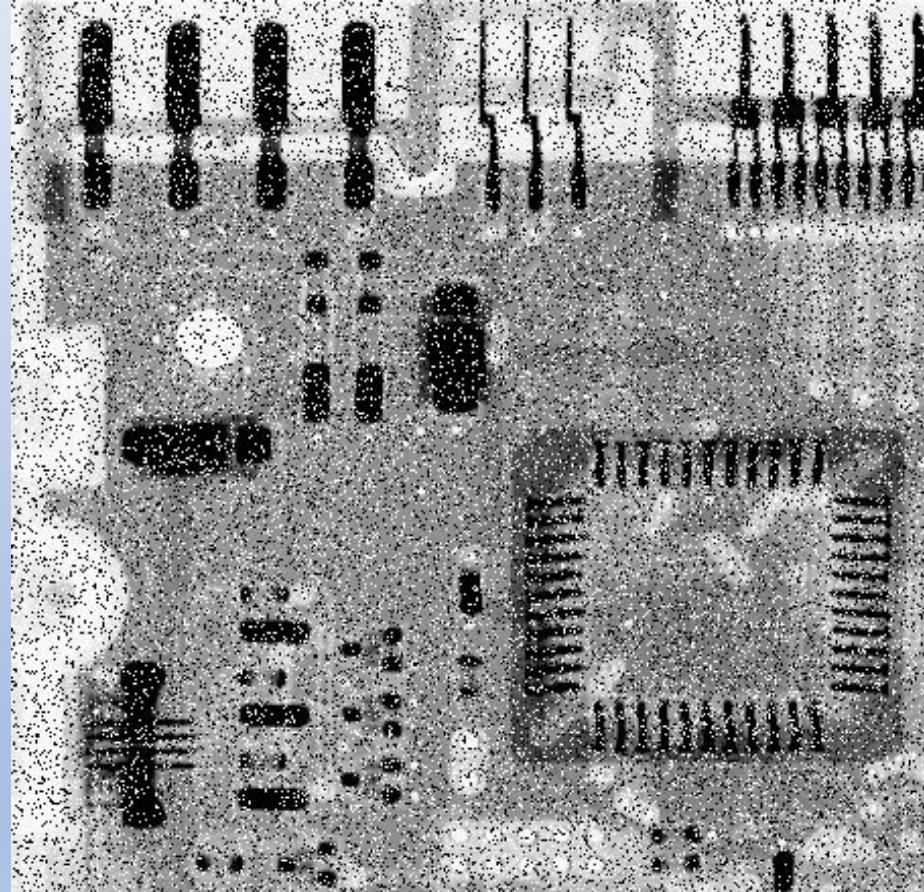
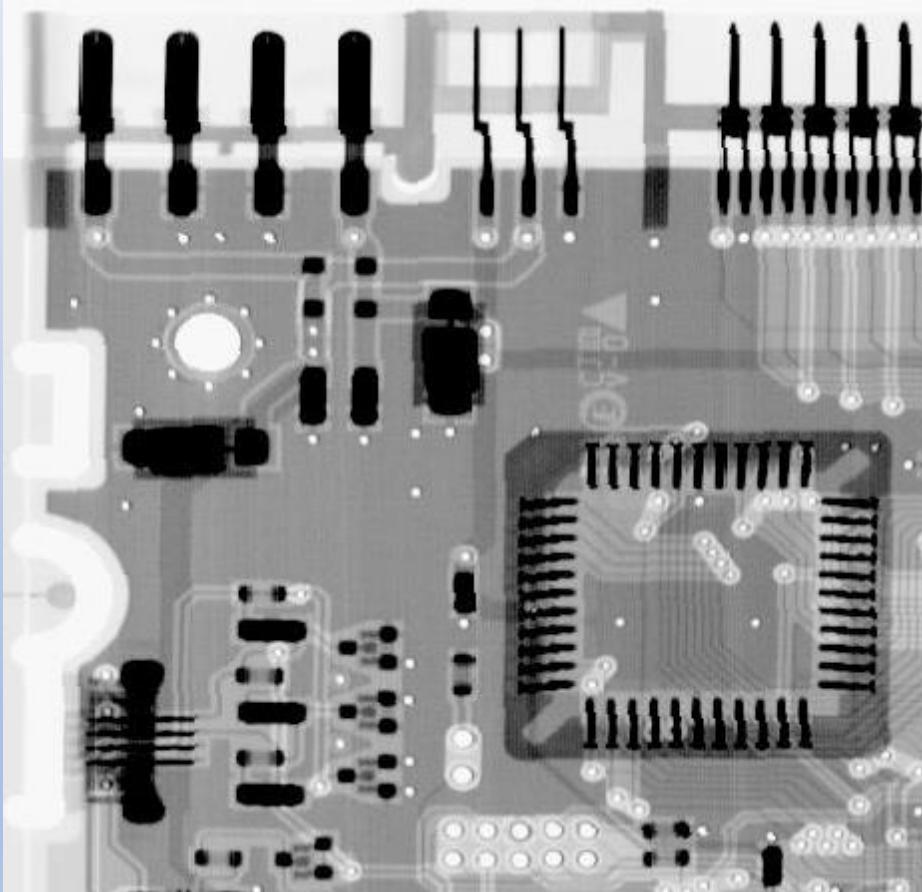
### 3. Image Enhancement—image sharpening

Examples of combining image enhancement



### 3. Image Enhancement—nonlinear processing

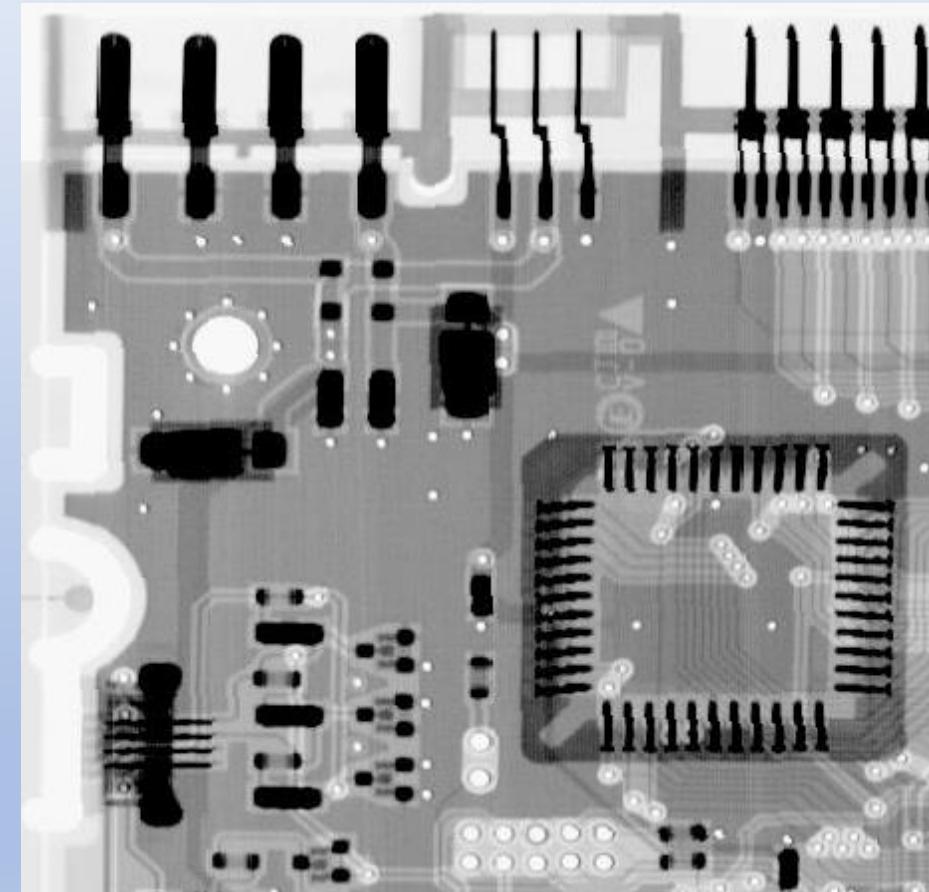
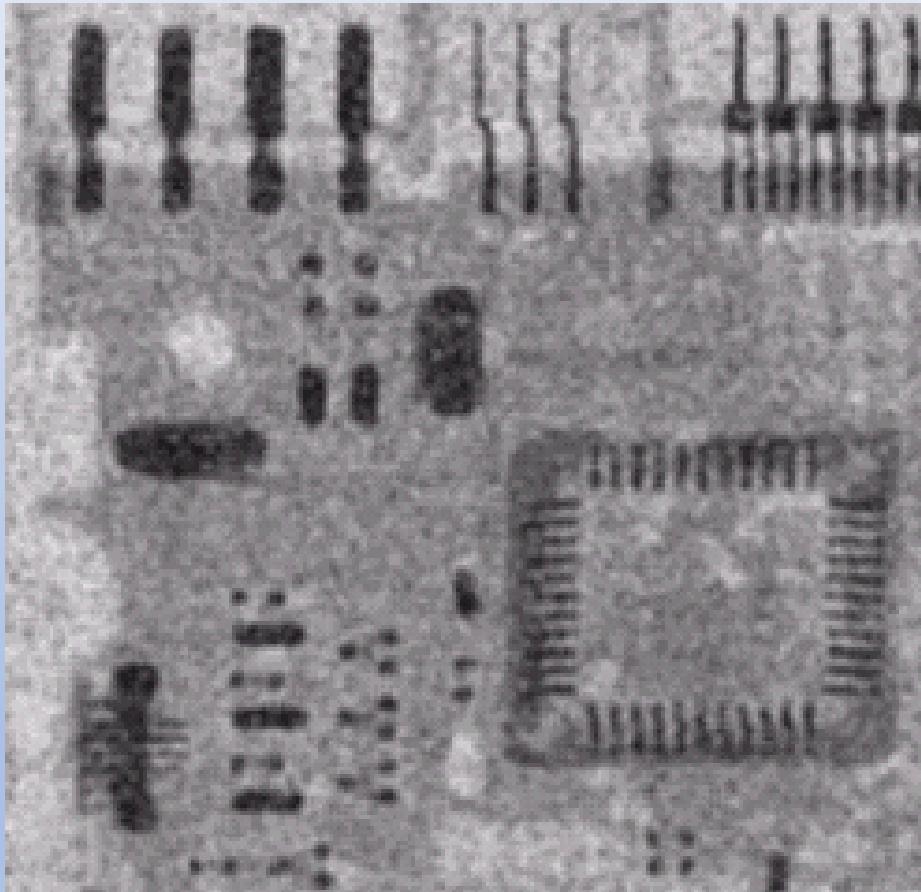
Observe an image and its noise contaminated version



What are the noise characteristics? How to remove such noise?

### 3. Image Enhancement—Problems of Linear Filter

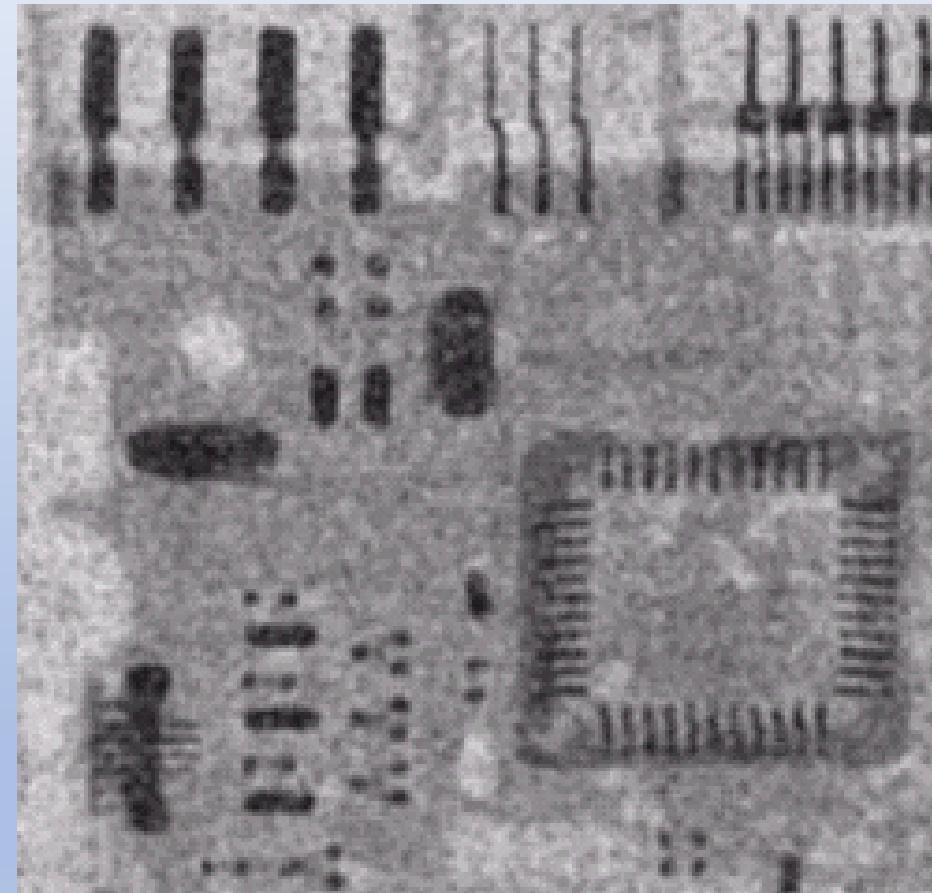
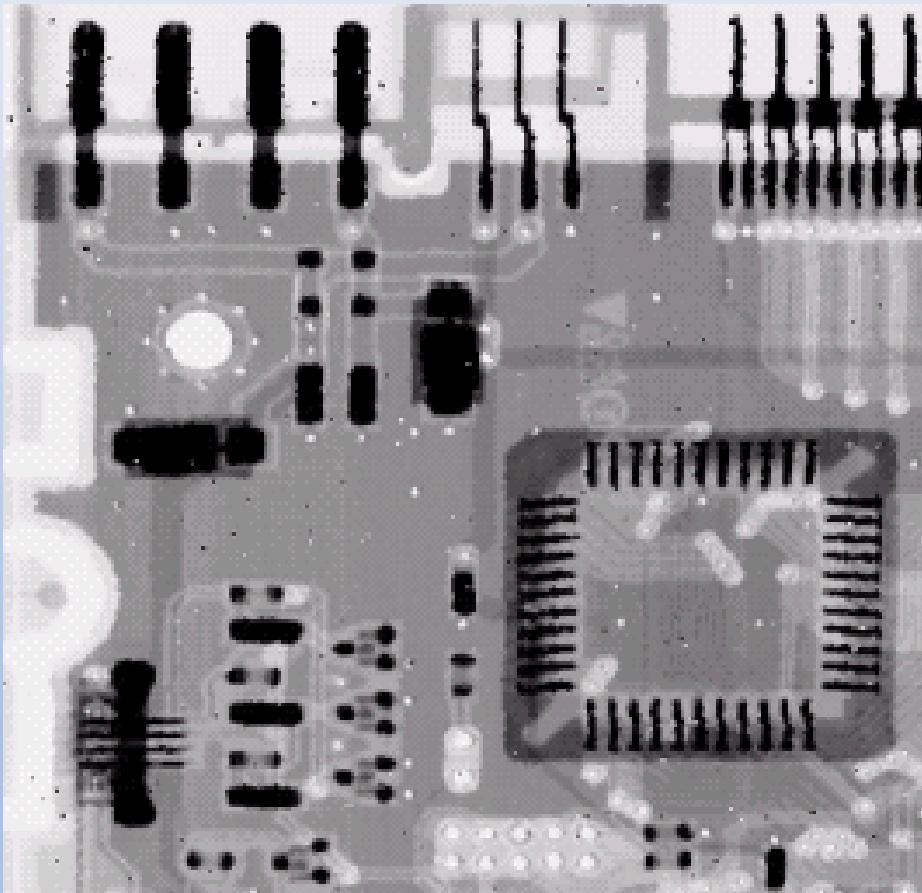
Observe the Image smoothed by a linear low pass filter



What are its problems comparing to the original image? Why?

### 3. Image Enhancement—Problems of Linear Filter

See another smoothed image comparing to the previous one



How is this smoothed image much better than the previous one?

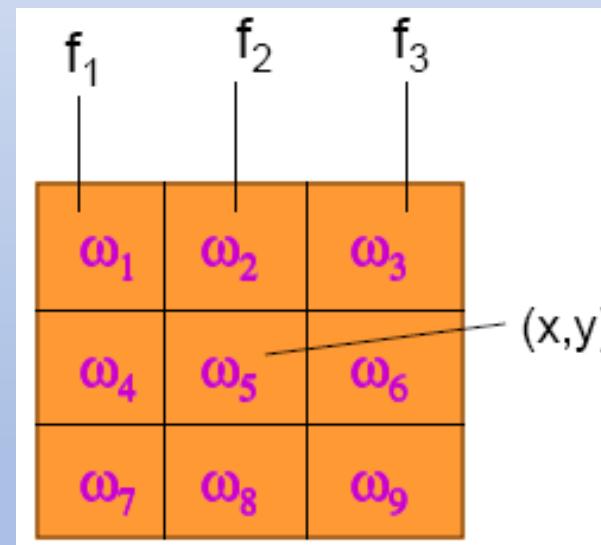
### 3. Image Enhancement—Problems of Linear Filter

- Any linear filter output is a weighted average of the input pixels

$$\begin{aligned}\hat{f}(x, y) &= h(x, y) * f(x, y) = \sum_{i=-a}^a \sum_{j=-b}^b h(i, j)f(x - i, y - j) \\ &= \sum_{(s,t) \in S_{xy}} \omega(s, t)f(s, t)\end{aligned}$$

- What are problems of the average of pixel grey values?

image blurring, sharpness details are lost, difficult to smooth strong noise



### 3. Image Enhancement—Order-Statistic Filters

- The response is based on ordering (ranking) the pixels contained in the image area encompassed by the filter.
- The best-known example is median filter, which replaces the value of a pixel by the median of the gray levels in the neighborhood of that pixel.

$$\hat{f}(x, y) = \underset{(s,t) \in S_{xy}}{\text{median}}\{f(s, t)\}$$

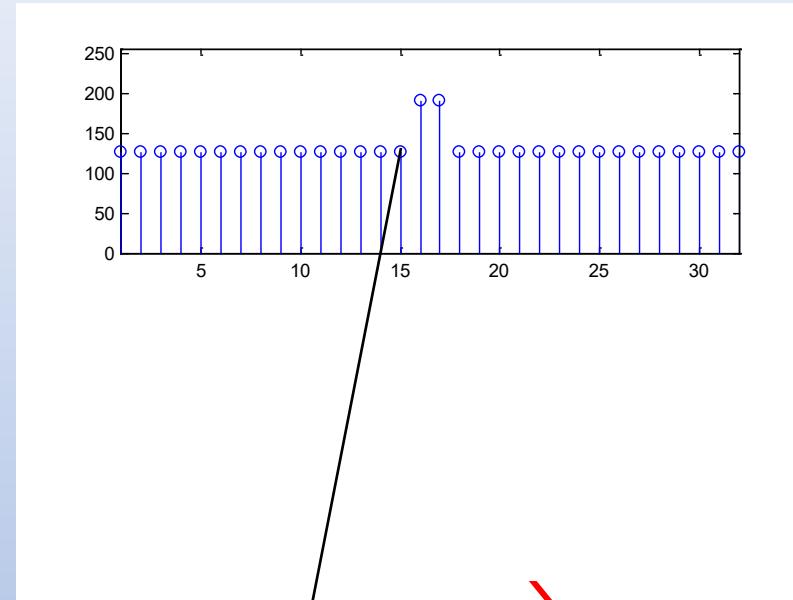
10	20	20
20	15	20
25	20	100



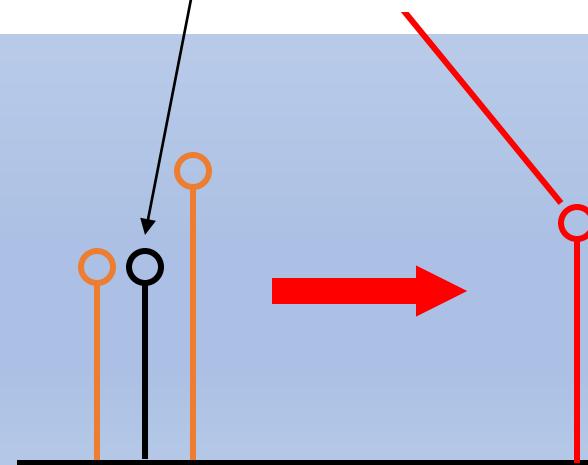
(10,15,20,20,20,20,20,25,100)  
Median=20  
So replace (15) with (20)

### 3. Image Enhancement—Mean vs. Median Filter

- Consider a uniform 1-D image with a pulse function.
- Pulse function corresponds to fine image detail such as lines and curves.

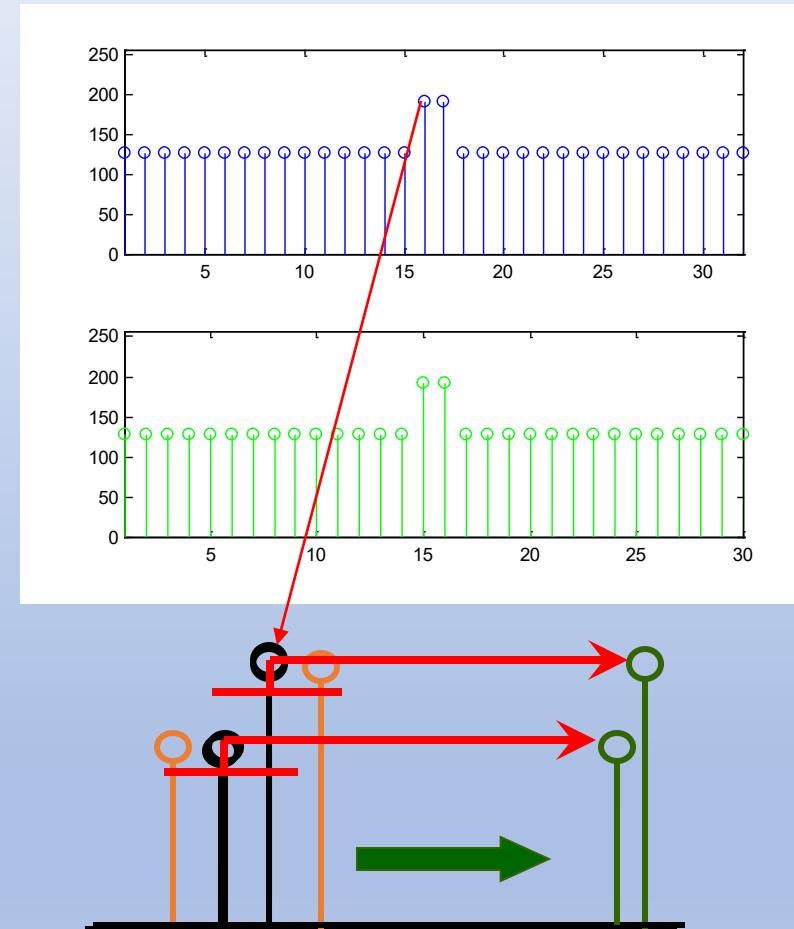


- Mean filter ‘blurs’ the image details.
- If the pulse is noise, mean filter suppress it **only for some extent** but **spread** the noise.



### 3. Image Enhancement—Mean vs. Median Filter

- Consider a uniform 1-D image with a pulse function.
  - Pulse function corresponds to fine image detail such as lines and curves.
  - Median filter does not ‘blur’ the edge.
- If the pulse is noise, 5X5 median filter totally remove such noise.



### 3. Image Enhancement—Median Filter

- Edge is a basic and significant structure of an image.
- Mean filter blurs the step edge. Median filter preserves the step edge.

What is the outputs of a mean filter?

$$\text{mean}\{0,0,0, \overleftarrow{1}, 1,1,1\} = 0.57$$

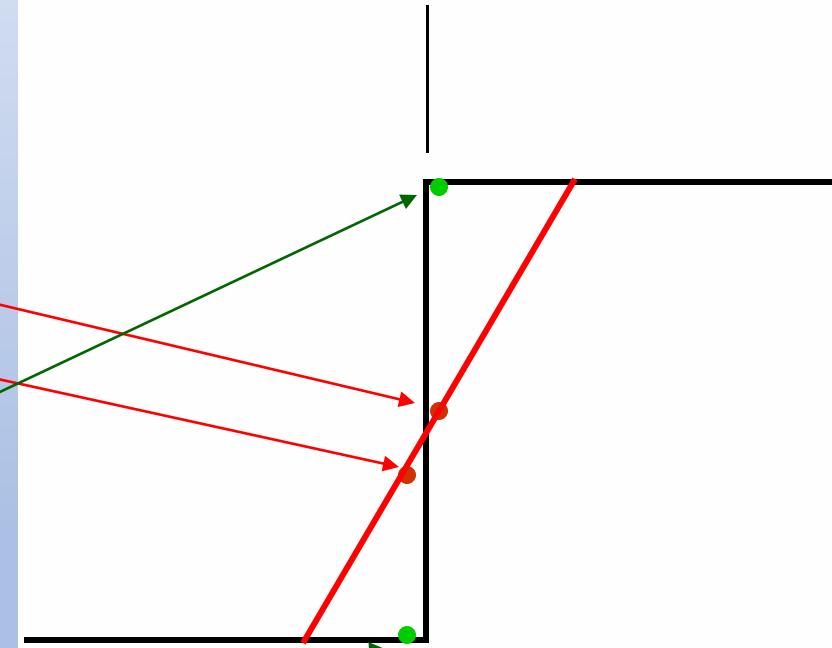
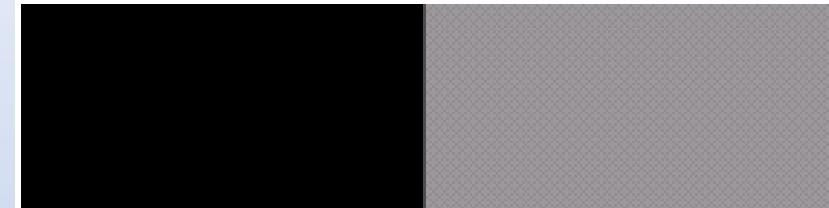
$$\text{mean}\{0,0,0, \overleftarrow{0}, 1,1,1, \} = 0.43$$

What is the outputs of a median filter?

$$\text{median}\{0,0,0, \overleftarrow{1}, 1,1,1\} = 1$$

$$\text{median}\{0,0,0, \overleftarrow{0}, 1,1,1, \} = 0$$

Model of an ideal digital edge



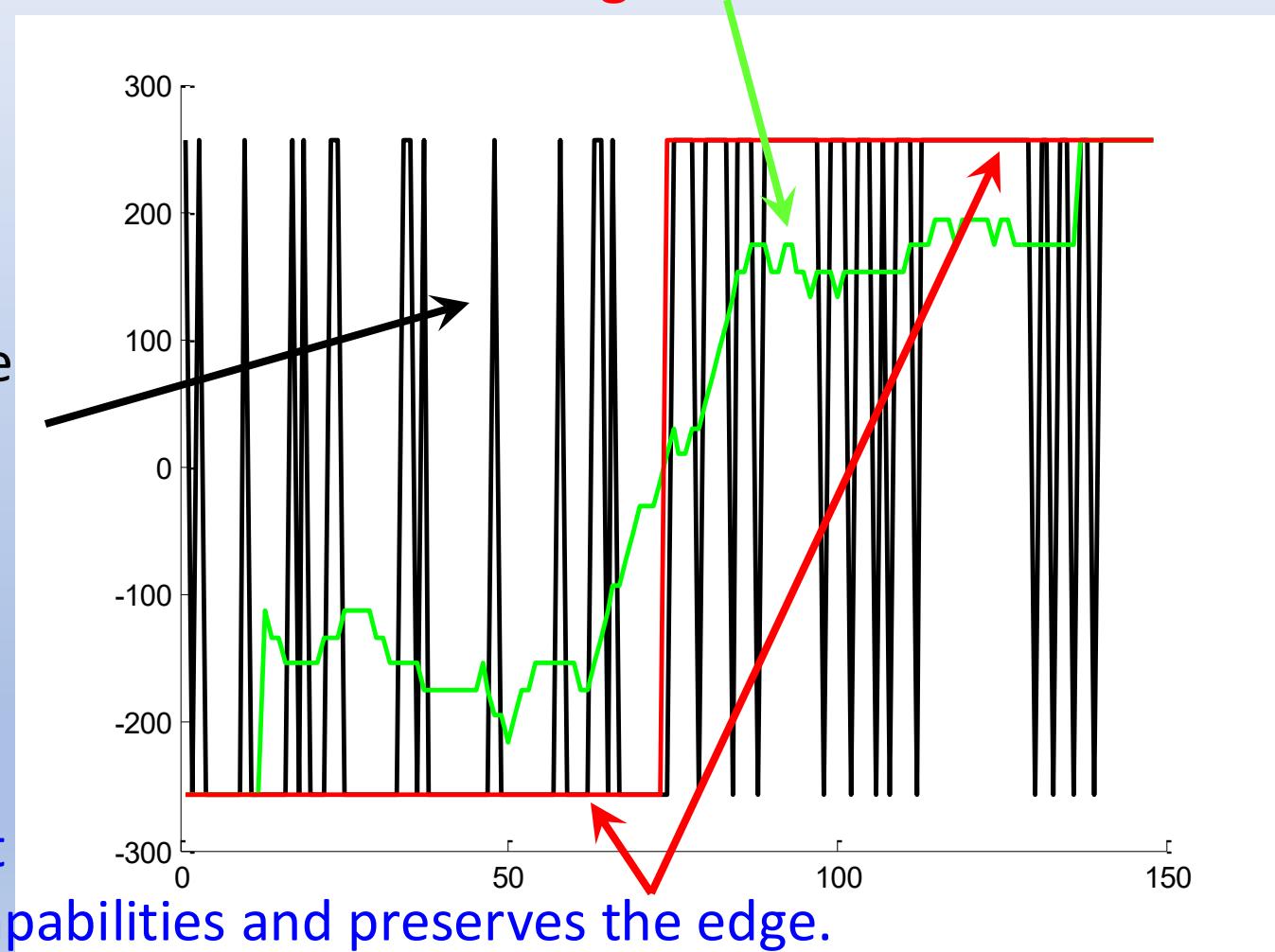
Gray-level profile  
of a horizontal line  
through the image

### 3. Image Enhancement—Mean vs. Median Filter

A simple MATLAB program can show: Mean filter is ineffective to attenuate impulsive noise and blurs the edge.

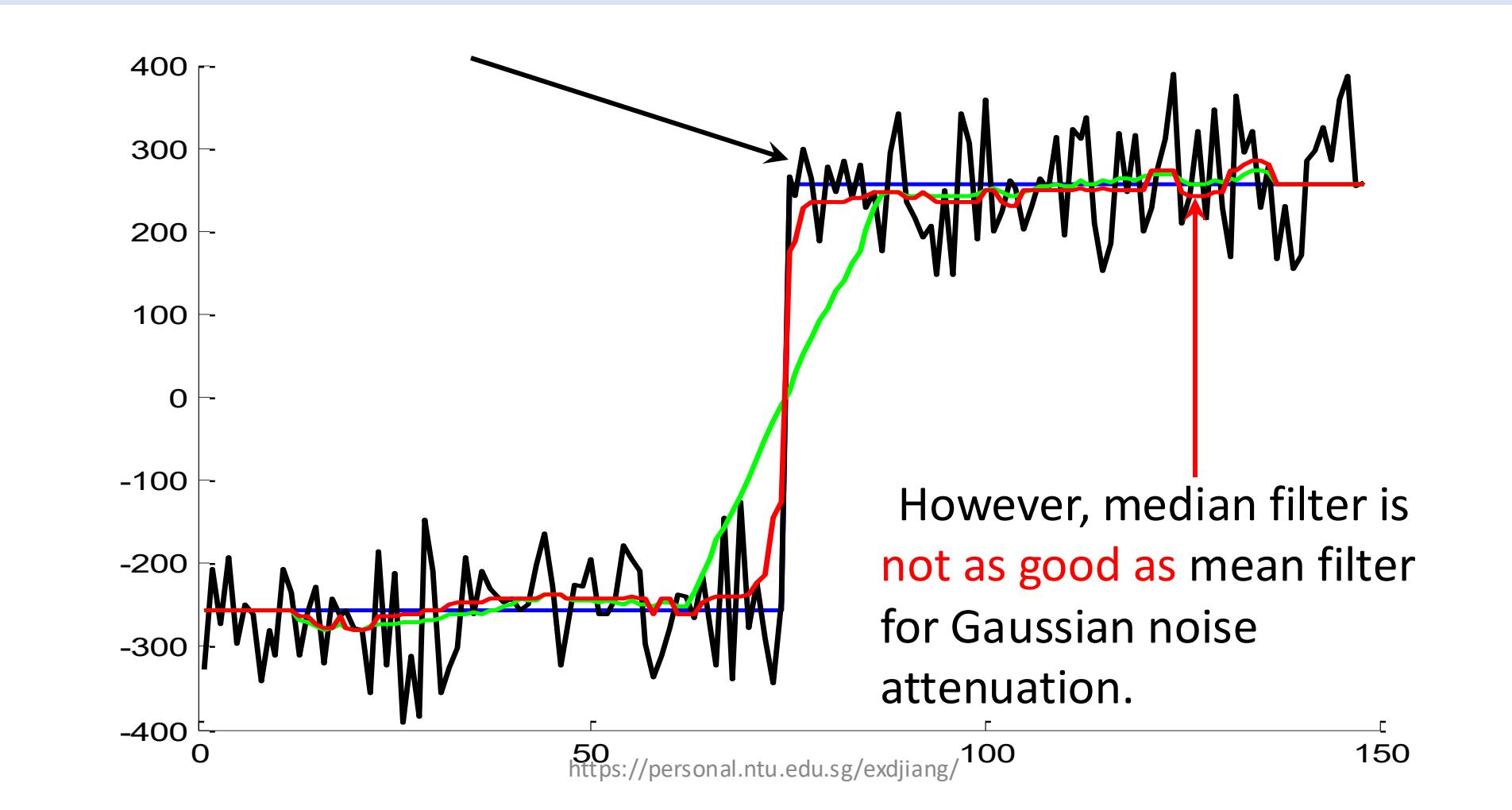
Impulsive noise is strong in amplitude and spatial sparse

Median filter provides excellent noise-reduction capabilities and preserves the edge.



### 3. Image Enhancement—Mean vs. Median Filter

A simple MATLAB program can show: Mean filter attenuates additive Gaussian noise but blurs the edge. Median filter attenuates Gaussian noise and preserves the edge.



### 3. Image Enhancement—Median Filter

$$\hat{f}(x, y) = \underset{(s,t) \in S_{xy}}{\operatorname{median}}\{f(s, t)\}$$

- Median filter forces the points with distinct gray levels to be more like their neighbors.
- Isolated clusters of pixels that are lighter or darker with respect to their neighbors, and whose area is less than  $n^2/2$  (one-half the filter area), are eliminated by an  $n \times n$  median filter.
- eliminated = forced to have the value equal the median intensity of the neighbors.
- Larger clusters are affected considerably less.

### 3. Image Enhancement—Mean vs. Median Filter

Original and noise corrupted images

impulse noise  $\Rightarrow$  salt and pepper noise.



### 3. Image Enhancement—Mean vs. Median Filter

Example outputs of



mean filter



and

median filter.

### 3 Nonlinear Image Smoothing—Med. Filter Properties

- Linear filter has established theory to analyze its properties, especially in the frequency domain.
- However, It is **difficult to analyze** Median filter and other order-statistic filters due to their nonlinearity.

- Repeated applications of median filter to a signal result in an invariant signal called the “**root signal**”. A root signal is invariant to further application of the median filter.
- Example: 1-D signal: Median filter length = 3

0 0 0 1 2 1 2 1 2 1 0 0 0

0 0 0 1 1 2 1 2 1 1 0 0 0

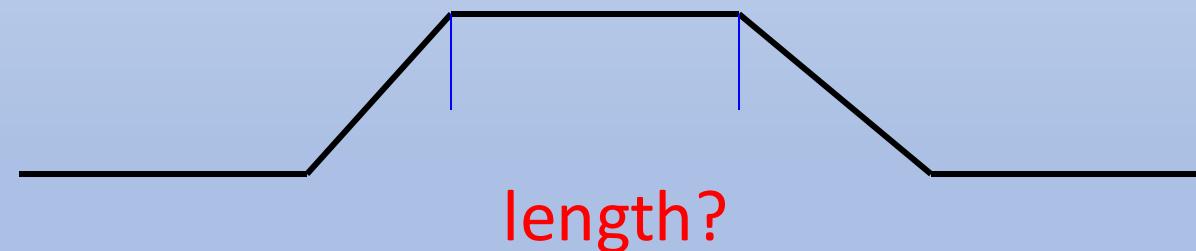
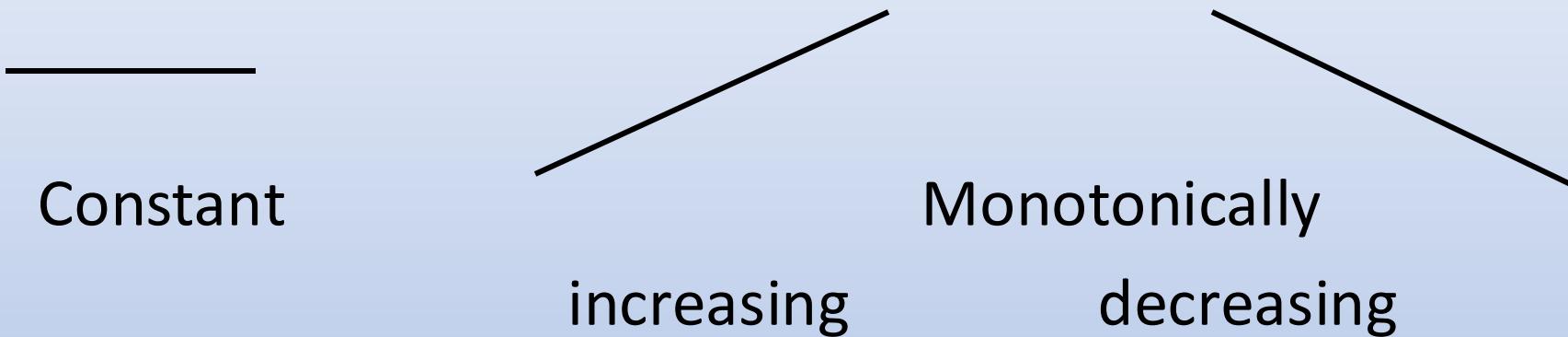
0 0 0 1 1 1 2 1 1 1 0 0 0

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

~      **root signal**

### 3. Image Enhancement—Med. Filter Properties

- Invariant signals to a median filter:



### 3. Image Enhancement—Other Order-stati. Filter

- Simple extension of the median filter

- Max filter

$$\hat{f}(x, y) = \max_{(s,t) \in S_{xy}} \{f(s, t)\}$$

- Min filter

$$\hat{f}(x, y) = \min_{(s,t) \in S_{xy}} \{f(s, t)\}$$

- Midpoint filter

$$\hat{f}(x, y) = \frac{1}{2} \left[ \max_{(s,t) \in S_{xy}} \{f(s, t)\} + \min_{(s,t) \in S_{xy}} \{f(s, t)\} \right]$$

### 3. Image Enhancement—Limitation and Solution

- Although Median filter preserves image edges, it **removes image details** such as corner, thin lines / curves and other fine details.
- How to design a rank order filter that can effectively removes impulsive noise and preserves these image details at the same time?
- The research work on this topic can be found in the research publication:

X.D. Jiang, “[Image Detail-Preserving Filter for Impulsive Noise Attenuation](#),” *IEE Proceedings: Vision, Image and Signal Processing*, Vol. 150, No. 3, pp. 179-185, June 2003.

### 3. Image Enhancement—Other Order-stati. Filter

- As median filter underperforms mean filter in attenuating short-tailed noise, e.g. Gaussian noise, filters that own merits of the both mean and median filters have been developed:
- Alpha-trimmed mean filter
- Iterative Truncated Arithmetic Mean Filter

$$\hat{f}(x, y) = \frac{1}{mn - d} \sum_{(s,t) \in S_{xy}} f_r(s, t)$$

where  $f_r(s, t)$  are the remaining  $mn - d$  pixels around median.

X.D. Jiang, “[Iterative Truncated Arithmetic Mean Filter And Its Properties](#),” *IEEE Transactions on Image Processing*, vol. 21, no. 4, PP. 1537-1547, April 2012.

Z. Miao and X.D. Jiang, “[Further Properties and a Fast Realization of the Iterative Truncated Arithmetic Mean Filter](#)” *IEEE Transactions on Circuits and Systems-II*, vol. 59, no. 11, pp. 810-814, Nov 2012.

Z. Miao and X.D. Jiang, “[Weighted Iterative Truncated Mean Filter](#),” *IEEE Transactions on Signal Processing*, Vol. 61, no. 16, pp. 4149-4160, Aug 2013.

Z. Miao and X.D. Jiang, “[Additive and exclusive noise suppression by iterative trimmed and truncated mean algorithm](#),” *Signal Processing*, vol. 99, pp. 147-158, June 2014.

# Iterative Truncated Arithmetic Mean Filter

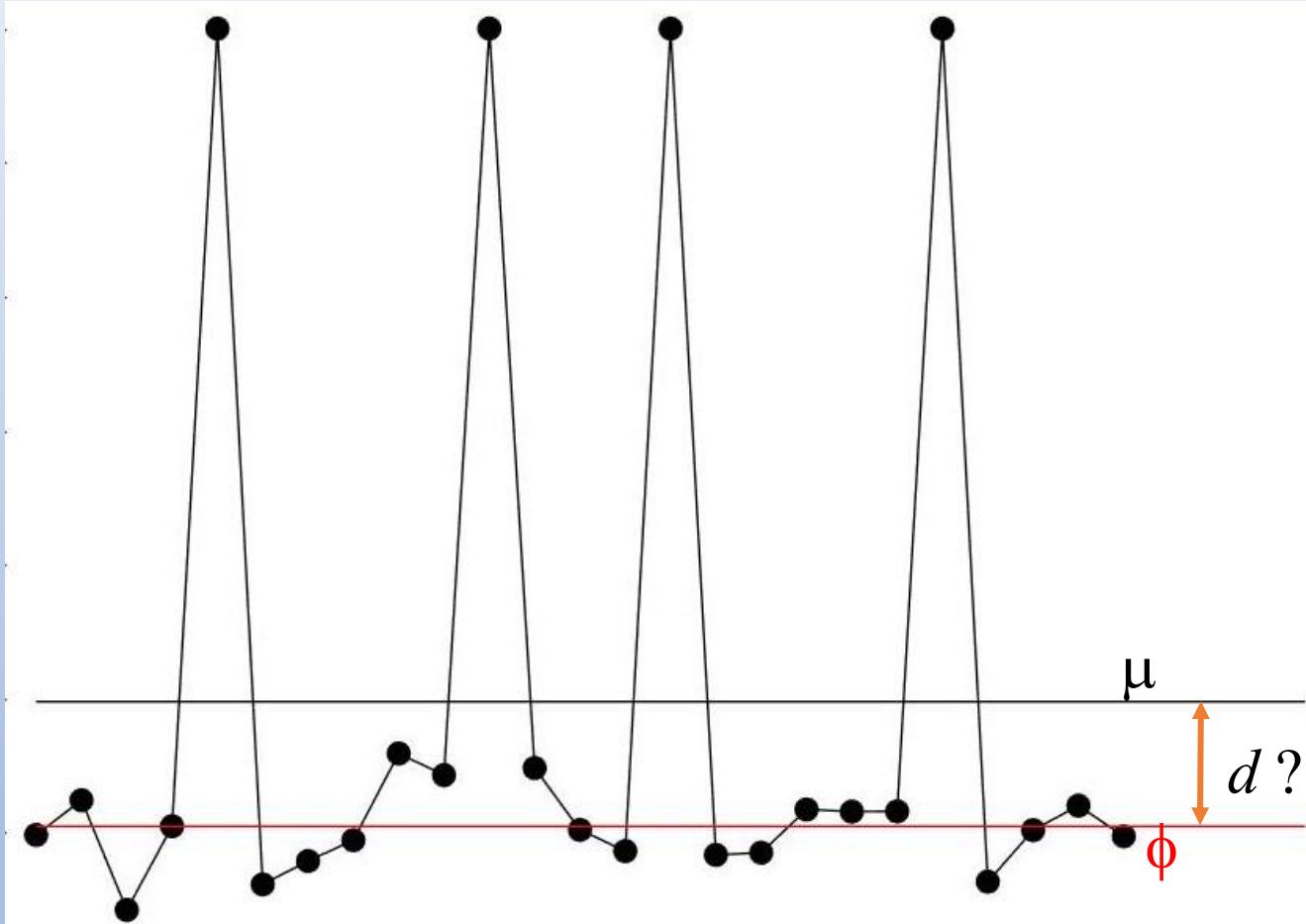
$$\mu = \arg \min_{\varphi} \sum_{i=1}^n (x_i - \varphi)^2$$

$$\phi = \arg \min_{\varphi} \sum_{i=1}^n |x_i - \varphi|$$

- As both mean and median have their own merits and limitations, how to find a solution between them that inherits the merits of the both operations?
- As the computation of median is inefficient, how to use the simple arithmetic mean to approach the order statistic median?
- To achieve these, we need first explore the relation between arithmetic mean and order statistic median

# Iterative Truncated Arithmetic Mean Filter

- Relation between arithmetic mean and order statistic median



$$\tau_1 = \frac{1}{2}(\mu_h - \mu_l)$$

$$\tau_2 = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2}$$

$$\tau_3 = \frac{1}{n} \sum_{i=1}^n |x_i - \mu|$$

$$d = |\phi - \mu|?$$

- For some data distribution, mean and median are close to each other while for some other data distribution, they are apart very far away.

# Iterative Truncated Arithmetic Mean Filter

- **Theorem 1:** The distance between the median and the mean of any finite data set is never greater than one sample standard deviation,  $\tau_2$ , never greater than the mean absolute deviation of the data from the mean,  $\tau_3$ , and never greater than the half distance between the upper mean and lower mean  $\tau_1$ .

$$|\phi - \mu| \leq \tau_1, \quad |\phi - \mu| \leq \tau_2, \quad |\phi - \mu| \leq \tau_3$$

- **Theorem 2:** The mean absolute deviation of the data from the mean is the tightest bound of the distance between the median and the mean of any finite data set.

$$\tau_3 \leq \tau_1, \quad \tau_3 \leq \tau_2$$

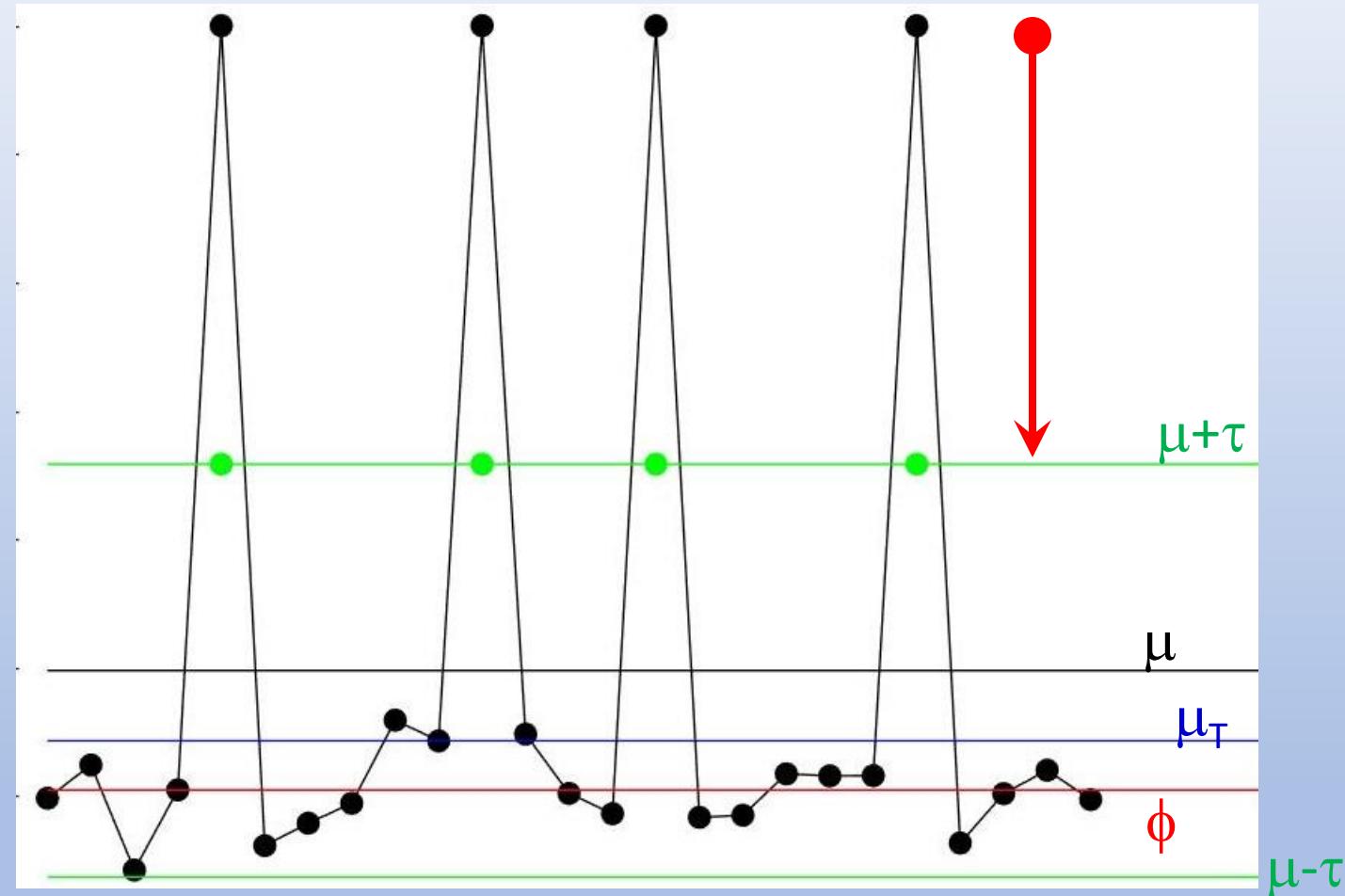
# Iterative Truncated Arithmetic Mean Filter

## Outline of the ITM algorithm:

- 1) Compute the mean
- 2) Compute threshold and truncate input data

$$x_i = \begin{cases} \mu + \tau, & \text{if } x_i > \mu + \tau \\ \mu - \tau, & \text{if } x_i < \mu - \tau \end{cases}$$

- 3) Return to step 1) if stopping criterion is violated. Otherwise, terminate iteration.



# Iterative Truncated Arithmetic Mean Filter

- **Theorem 3:** For any finite data set, there exists at least one sample whose distance from the sample mean is greater than the mean absolute deviation of the samples from the mean if the sample median deviates from the sample mean, i.e., letting

$$\exists x_i, x_i \in \mathbf{x}, \quad \text{that } |x_i - \mu| > \tau, \text{ if } \phi \neq \mu$$

- **Theorem 4:** The ITM algorithm decreases truncation threshold monotonically to zero if the mean deviates from the median.

$$\tau(k) < \tau(k - 1), \quad \lim_{k \rightarrow \infty} \tau(k) = 0, \text{ if } \phi \neq \mu$$

- **Theorem 5:** The truncated mean of the ITM algorithm approaches to median arbitrarily close.

# Iterative Truncated Arithmetic Mean Filter

- Mean and median are two **fundamental data operations** that have **different** characteristics. It's desirable to have merits of the both.
- Comparing with the arithmetic operation, data sorting required by computing median is a **complex process** and is **intractable**.
- This work discovers the **relation between the two fundamental statistics**, the arithmetic mean and the order statistical median.
- Based on this discovery, ITM filter is developed that **circumvents the data-sorting process** to approach the median.
- Proper termination of the proposed ITM algorithm enables the filters to own **merits of the both mean and median** and, hence, **outperform both the filters** in many image denoising applications.
- Although it is an iterative algorithm, **only few iterations** are required to achieve good results, It is faster than the median computation.

# 4 Morphological Image Processing –Outline

- Introduction
- Set Theory and Logic Operation
- Dilation and Erosion
- Opening and Closing
- Morphological Algorithm and applications

# 4 Morphological Image Processing –Introduction

Looking at these images.....

What is interesting, important or useful information we care about?

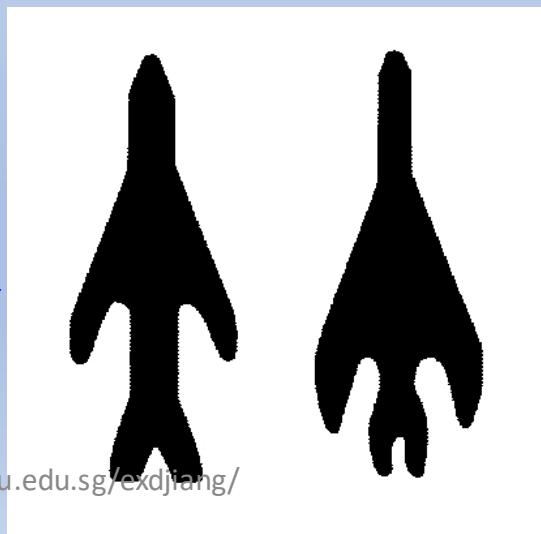
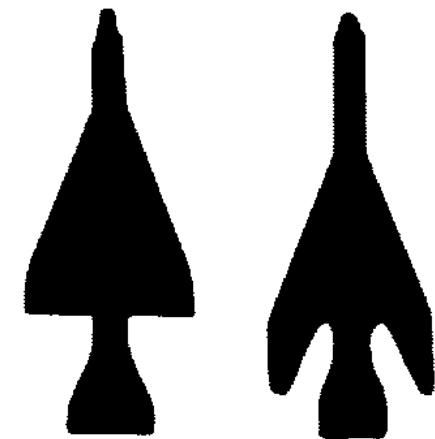
The grey value of the image is not important as there are only two different grey values.

➤ Region shape and boundaries of object are important.

➤ A binary image can be represented by object pixel set.

$$A = \{a \mid a=(x,y), f(x,y)=1\}$$

$$f(x,y)$$



A B C

D E F



# 4 Morphological Image Processing –Introduction

- Morphology deals with form and structure
- Mathematical morphology is a tool for extracting image components useful in:
  - representation and description of region shape (e.g. boundaries)
  - pre- or post-processing (filtering, thinning, etc.)
- Morphological operations are powerful tools in image analysis. They usually operate on binary images and thus often follow a segmentation task or an edge detection task.
- Based on set theory and logic operations

# 4 Morphology – Summary 1

Operation	Equation	Comments
Translation	$(A)_z = \{w \mid w = a + z, \text{ for } a \in A\}$	Translates the origin of $A$ to point $z$ .
Reflection	$\hat{B} = \{w \mid w = -b, \text{ for } b \in B\}$	Reflects all elements of $B$ about the origin of this set.
Complement	$A^c = \{w \mid w \notin A\}$	Set of points not in $A$ .
Difference	$A - B = \{w \mid w \in A, w \notin B\}$ $= A \cap B^c$	Set of points that belong to $A$ but not to $B$ .
Dilation	$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\}$	“Expands” the boundary of $A$ . (I)
Erosion	$A \ominus B = \{z \mid (B)_z \subseteq A\}$	“Contracts” the boundary of $A$ . (I)

## 4 Morphology –Dilation

- Dilation of  $A$  by  $B$ , denoted by  $A \oplus B$  , is defined as:

$$A \oplus B = \{z | [(\hat{B})_z \cap A] \neq \emptyset\}$$

- Interpretation:  
Obtaining the reflection of  $B$  about its origin and then shifting this reflection by  $z$ . Dilation of  $A$  by  $B$  then is the set of all  $z$  displacements such that the shifted  $\hat{B}$  and  $A$  overlap by at least one nonzero element.
- $B$  is called the structuring element in Dilation.

# 4 Morphology –Dilation

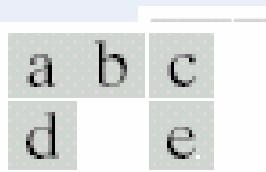
- Dilation of  $A$  by  $B$  can also be expressed as:

$$A \oplus B = \{z | [(\hat{B})_z \cap A] \subseteq A\}$$

- Further Interpretation:

Set  $B$  can be viewed as a convolution mask. The basic process of “flipping”  $B$  and then successively displace it so that it slides over set (image)  $A$  is analogous to the convolution.

# 4 Morphology –Dilation



(a) Set  $A$ .

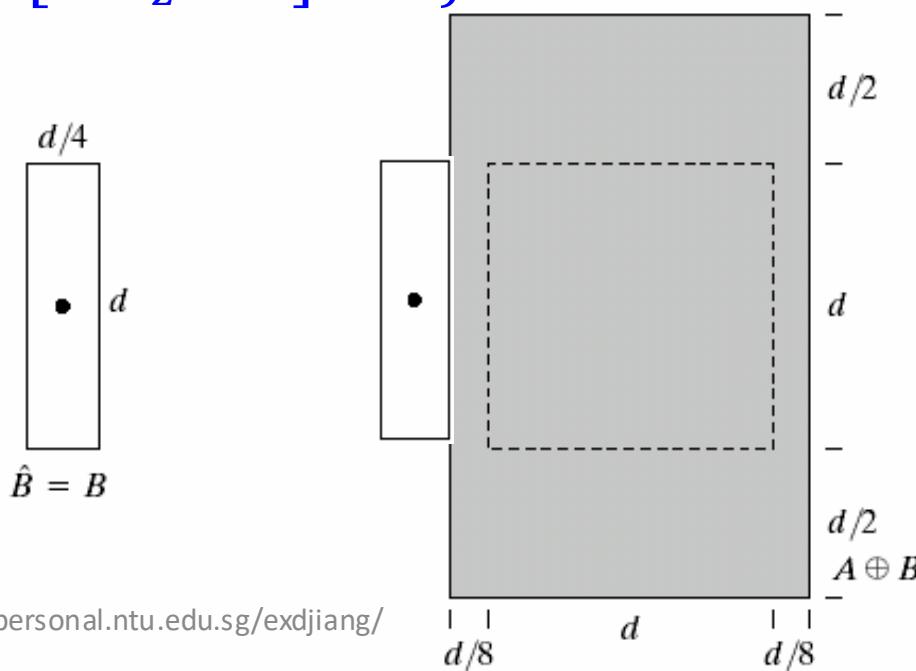
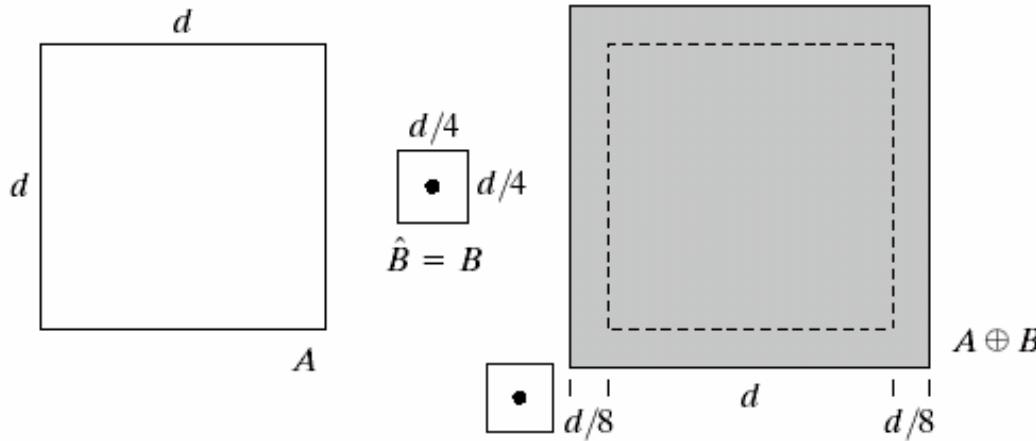
(b) Square structuring element (dot is the center).

$$A \oplus B = \{z | [(\hat{B})_z \cap A] \neq \emptyset\}$$

(c) Dilation of  $A$  by  $B$ , shown shaded.

(d) Elongated structuring element.

(e) Dilation of  $A$  using this element.

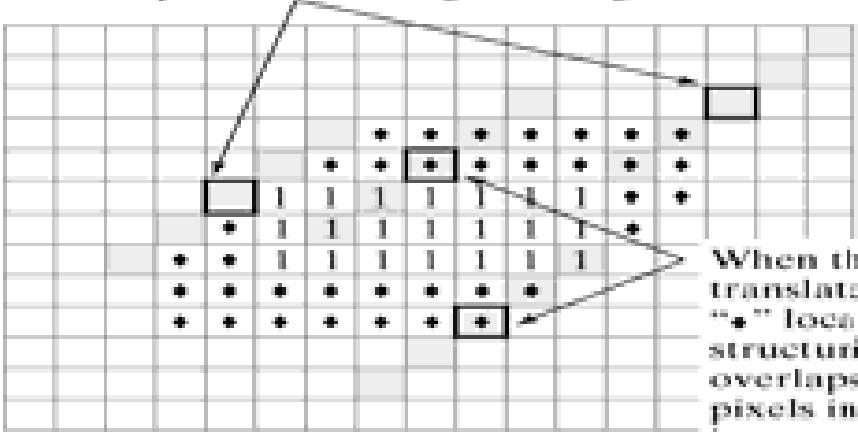


```

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

```

The structuring element translated to these locations does not overlap any 1-valued pixels in the original image.



```

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

```

1 1 1  
1 1 1  
1 1 1

a	b
c	
d	

## Dilation

Illustration of dilation.

(a) Original image with rectangular object.

(b) Structuring element with five pixels arranged in a diagonal line. The origin of the structuring element is shown with a dark border.

(c) Structuring element translated to several locations on the image.

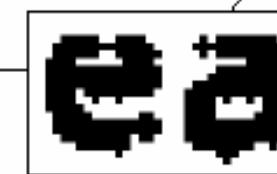
(d) Output image.

# 4 Morphology –Dilation

Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.



Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.



a  
b  
c

- (a) Sample text of poor resolution with broken characters (magnified view).  
(b) Structuring element.  
(c) Dilation of (a) by (b). Broken segments were joined.

0	1	0
1	1	1
0	1	0

## 4 Morphology –Erosion

- Erosion of  $A$  by  $B$ , denoted  $A \ominus B$ , is defined as:

$$A \ominus B = \{z | (B)_z \subseteq A\}$$

- Erosion of  $A$  by  $B$  is the set of all points  $z$  such that  $B$ , translated by  $z$ , is contained in  $A$ .
- Comparing with the Dilation:

$$A \oplus B = \left\{ z \mid [(\hat{B})_z \cap A] \subseteq A \right\}$$

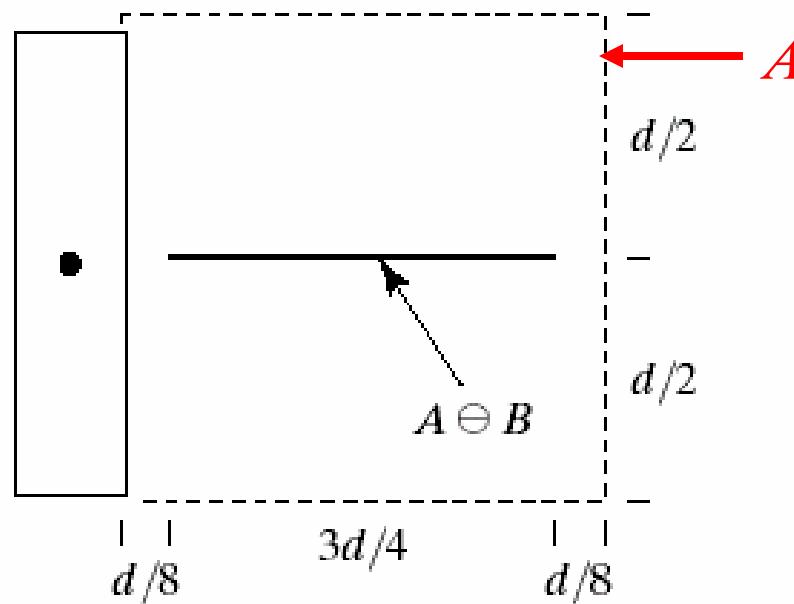
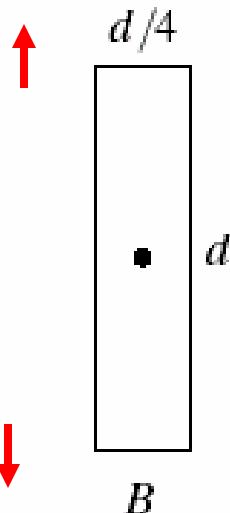
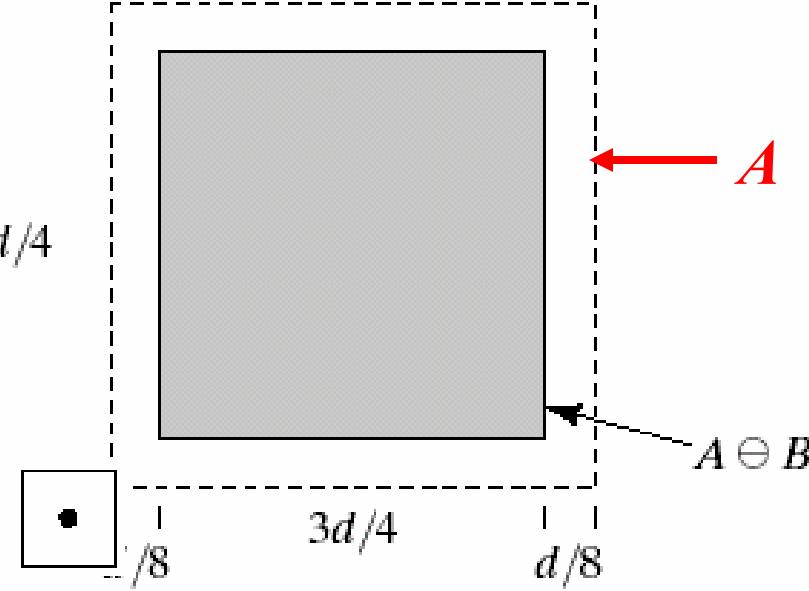
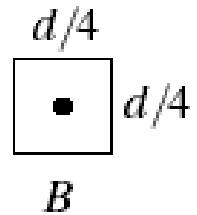
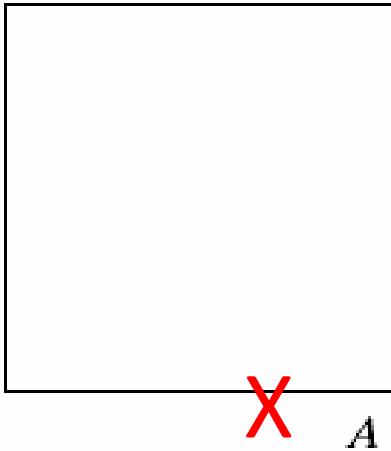
- Dilation and erosion are duals of each other with respect to set complementation and reflection. That is,

$$(A \ominus B)^c = A^c \oplus \hat{B}$$

# 4 Morphology

$$A \ominus B = \{z | (B)_z \subseteq A\}$$

a	b	c
d	e	



(a) Set A. (b) Square structuring element. (c) Erosion of  $A$  by  $B$ , shown shaded. (d) Elongated structuring element. (e) Erosion of  $A$  using this element.

# 4 Morphology –Erosion



a b c

(a) Image of squares of size 1, 3, 5, 7, 9, and 15 pixels on the side. (b) Erosion of (a) with a square structuring element of 1's, 13 pixels on the side. (c) Dilation of (b) with the same structuring element.

## 4 Morphology –Opening

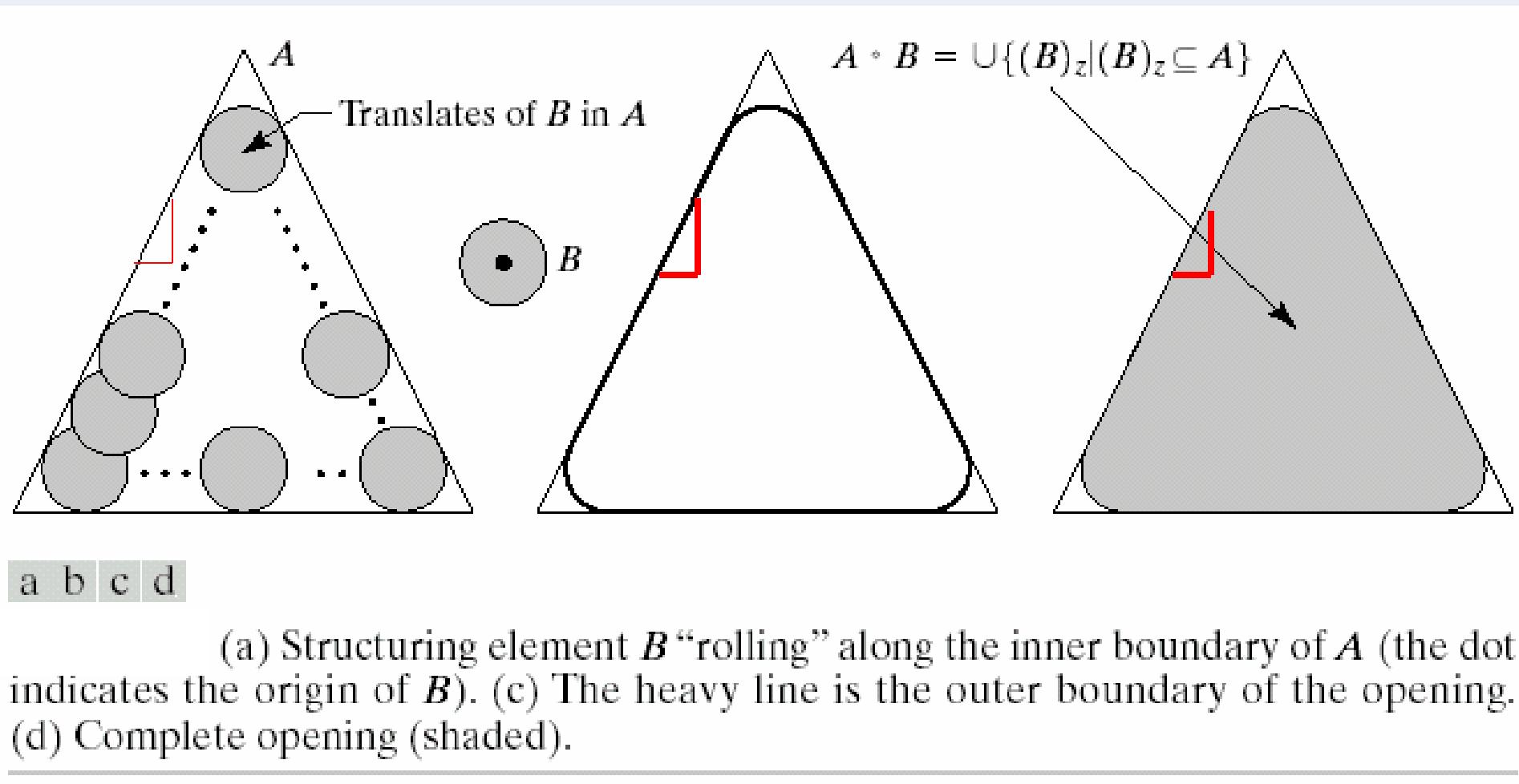
- Compound operations – Opening
- A compound operation is when two or more morphological operations are performed in succession. A common example is opening which is an erosion followed by a dilation:

$$A \circ B = (A \ominus B) \oplus B$$

- The opening  $A$  by  $B$  is obtained by taking the union of all translates of  $B$  that fit into  $A$ . This can be expressed as a fitting processing such that:

$$A \circ B = \cup \{(B)_z | (B)_z \subseteq A\} \quad A \ominus B = \{z | (B)_z \subseteq A\}$$

## 4 Morphology –Opening



➤ Note that the outward pointing corners are rounded, where the inward pointing corners remain unchanged.

## 4 Morphology –Opening

$$A \circ B = (A \ominus B) \oplus B \quad A \circ B = \cup \{(B)_z | (B)_z \subseteq A\}$$

- Opening is often performed to clear an image of noise whilst retaining the original object size. Care must be taken that the operation does not distort the shape size of the object if this is significant.
- The opening operation tends to flatten the sharp peninsular projections on the object.
- A useful way to see the effects of an opening operation is to look for differences between the original image and the image after opening by projecting these differences onto the original image.

## 4 Morphology –Closing

- Compound operations – Closing
- Closing is the complementary operation of opening, defined as dilation followed by erosion.

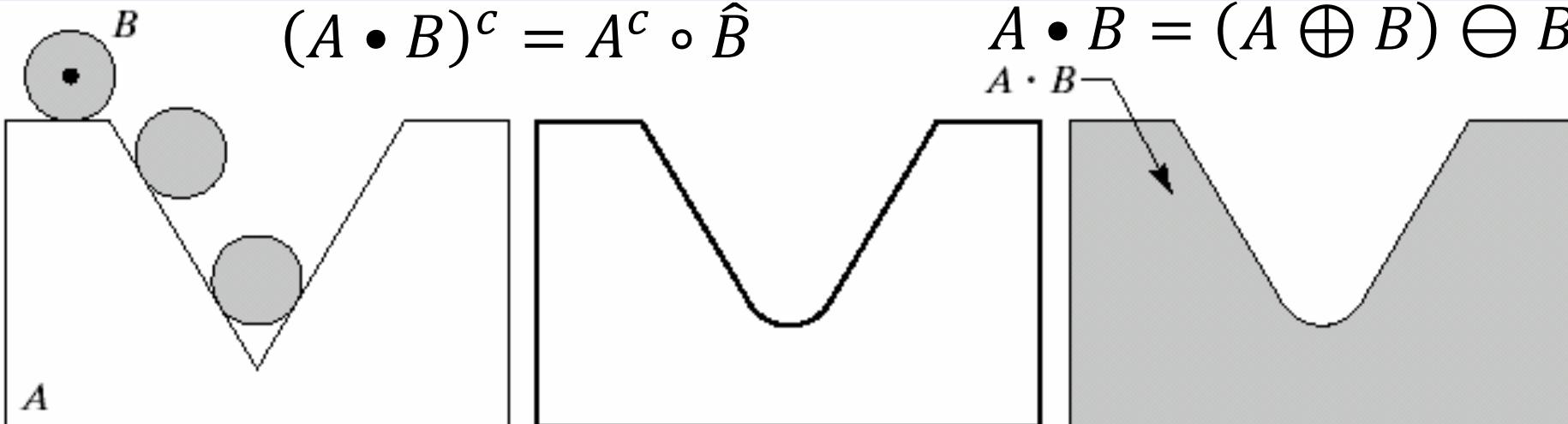
$$A \bullet B = (A \oplus B) \ominus B$$

➤ Opening and closing are duals of each other as:

$$(A \bullet B)^c = A^c \circ \hat{B}$$

Or:  $A \bullet B = (A^c \circ \hat{B})^c$

## 4 Morphology –Closing



a | b | c

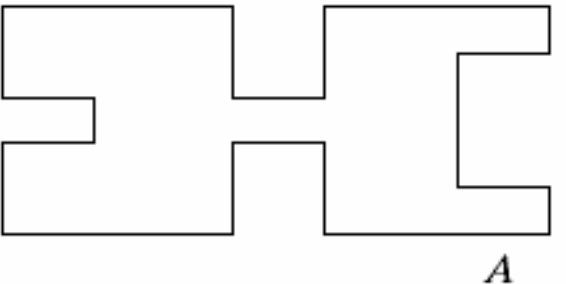
(a) Structuring element  $B$  “rolling” on the outer boundary of set  $A$ . (b) Heavy line is the outer boundary of the closing. (c) Complete closing (shaded).

- Note that the inward pointing corners are rounded, where the outward pointing corners remain unchanged.

## 4 Morphology –Closing

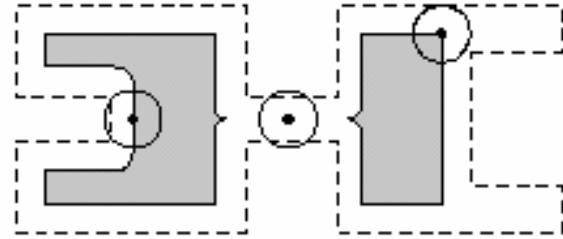
- The classic application of closing is to **fill holes** in a region **whilst retaining the original object size**.
- Dilation fills the holes and erosion restores the original region size.
- In addition to filling holes the closing operation tends to fill the ‘bays’ on the edge of a region.

## 4 Morpho

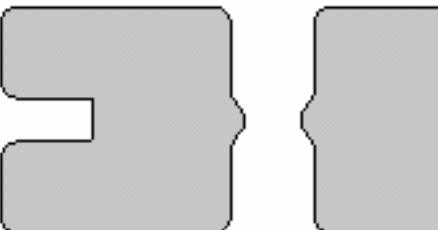
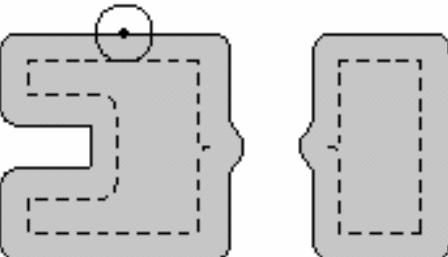


*A*

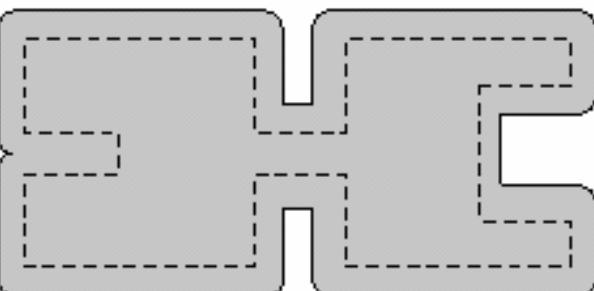
Examples and  
Interpretation of  
erosion, dilation,  
opening and closing



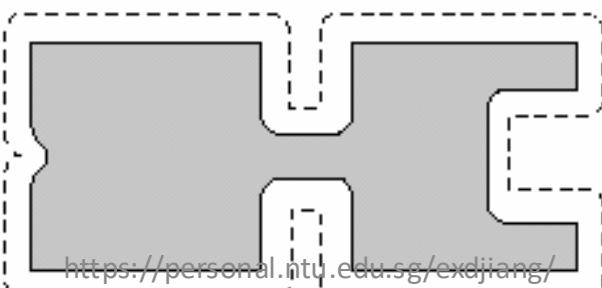
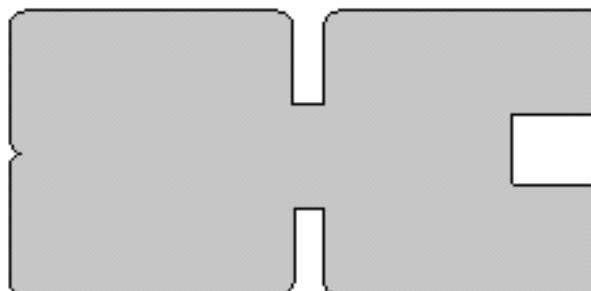
$$A \ominus B$$



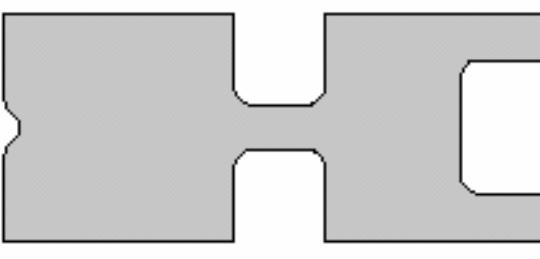
$$A \circ B = (A \ominus B) \oplus B$$



$$A \oplus B$$



<https://personal.ntu.edu.sg/exdjiang/>



$$A \cdot B = (A \oplus B) \ominus B$$

## 4 Morphology –Opening and Closing

- The opening operation satisfies the following properties:
  - $A \circ B$  is a subset (subimage) of  $A$
  - If  $C$  is a subset of  $D$ , then  $C \circ B$  is a subset of  $D \circ B$
  - $(A \circ B) \circ B = A \circ B$

➤ Similarly, the closing operation satisfies the following properties:

$A$  is a subset (subimage) of  $A \bullet B$

If  $C$  is a subset of  $D$ , then  $C \bullet B$  is a subset of  $D \bullet B$

$$(A \bullet B) \bullet B = A \bullet B$$

## 4 Morphology –Algorithms and Applications

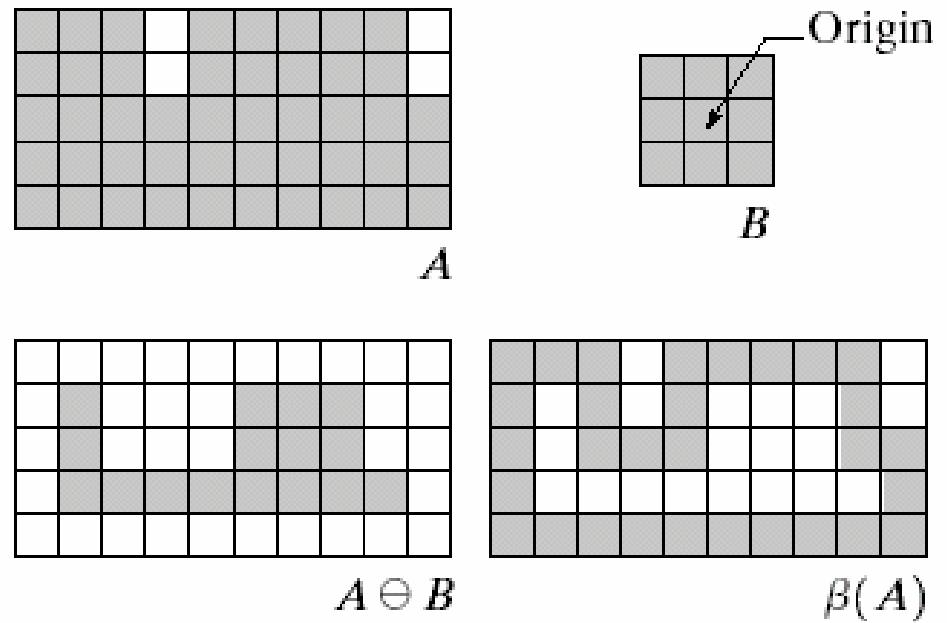
- **Boundary Extraction:** The boundary of a set  $A$ , denoted by  $\beta(A)$ , can be obtained by:

$$\beta(A) = A - (A \ominus B)$$

Two pixels are called 4-connected if one is a 4-neighbor of the other.  
Two pixels are called 8-connected if one is a 8-neighbor of the other.

- a b  
c d
- (a) Set  $A$ . (b) Structuring element  $B$ . (c)  $A$  eroded by  $B$ .  
(d) Boundary, given by the set difference between  $A$  and its erosion.

A 4-connected boundary.

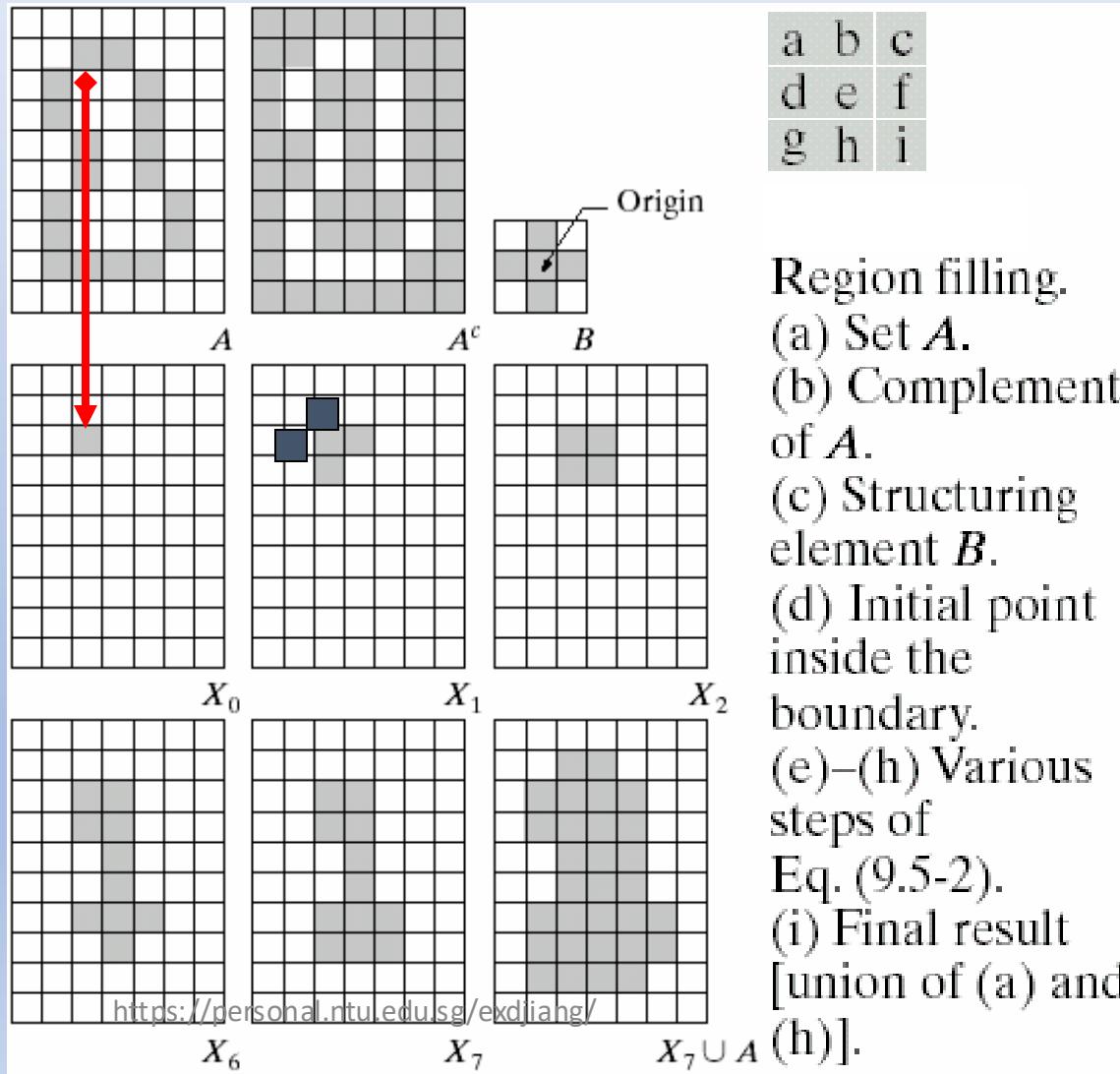


# 4 Morphology –Algorithms and Applications

- Region Filling:  $X_k = (X_{k-1} \oplus B) \cap A^c$ ,  $k = 1, 2, 3, \dots$

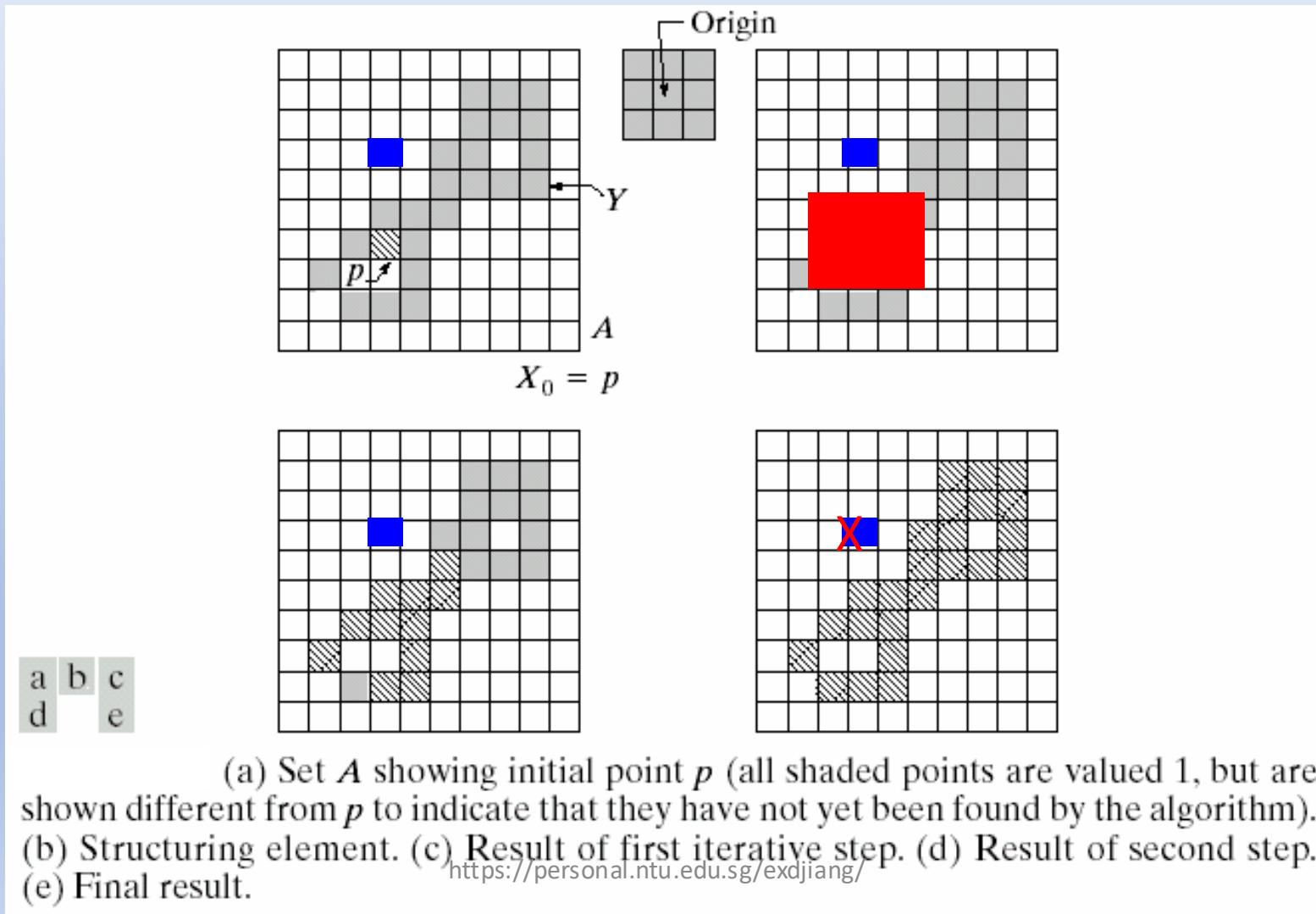
$$A^F = X_k \cup A$$

Beginning with a point  $X_0$  inside the boundary, the entire region inside the boundary is filled by the above procedure.



# 4 Morphology –Algorithms and Applications

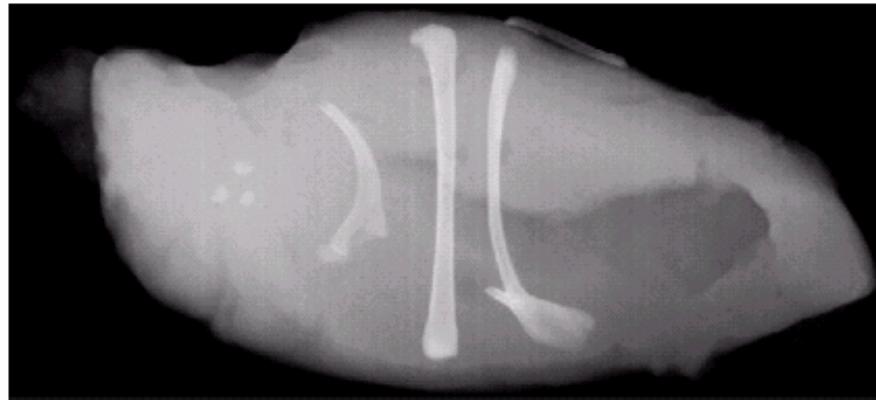
- Extract connected components:  $X_k = (X_{k-1} \oplus B) \cap A, k = 1, 2, 3, \dots$



# 4 Morphology –Algorithms and Applications

a  
b  
c d

- (a) X-ray image of chicken filet with bone fragments.  
(b) Thresholded image. (c) Image eroded with a  $5 \times 5$  structuring element of 1's.  
(d) Number of pixels in the connected components of (c). (Image courtesy of NTB Elektronische Geraete GmbH, Diepholz, Germany, [www.ntbxray.com.](http://www.ntbxray.com/))



Connected component	No. of pixels in connected comp
01	11
02	9
03	9
04	39
05	133
06	1
07	1
08	743
09	7
10	11
11	11
12	9
13	9
14	674
15	85

## 4 Morphology –Algorithms and Applications

- Denoising:

$$(A \circ B) \bullet B$$

- Or

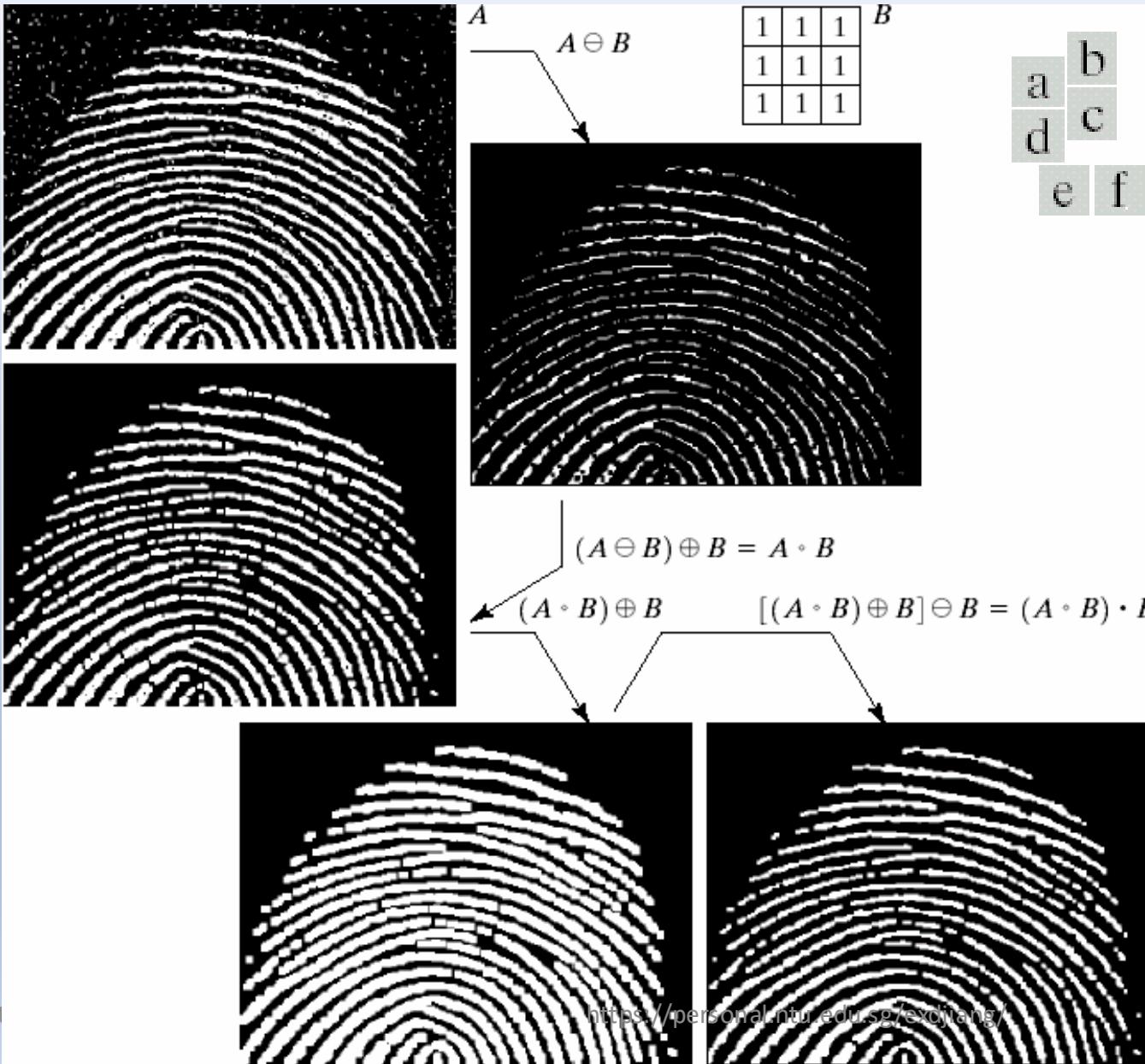
$$(A \bullet B) \circ B$$

Can be used to eliminate noise and its effect on the object.

- Noise pixels outside the object area are removed by opening with  $B$  while noise pixels inside the object area are removed by closing with  $B$ .

See example in the next slide

## 4 Morphology –Algorithms and Applications



(a) Noisy image.  
(c) Eroded image.  
(d) Opening of A.  
(d) Dilation of the opening.  
(e) Closing of the opening. (Original image for this example courtesy of the National Institute of Standards and Technology.)

# 4 Morphology –Summary

Operation	Equation	Comments
Translation	$(A)_z = \{w \mid w = a + z, \text{ for } a \in A\}$	Translates the origin of $A$ to point $z$ .
Reflection	$\hat{B} = \{w \mid w = -b, \text{ for } b \in B\}$	Reflects all elements of $B$ about the origin of this set.
Complement	$A^c = \{w \mid w \notin A\}$	Set of points not in $A$ .
Difference	$A - B = \{w \mid w \in A, w \notin B\}$ $= A \cap B^c$	Set of points that belong to $A$ but not to $B$ .
Dilation	$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \emptyset\}$	“Expands” the boundary of $A$ . (I)
Erosion	$A \ominus B = \{z \mid (B)_z \subseteq A\}$	“Contracts” the boundary of $A$ . (I)

# 4 Morphology –Summary

Opening	$A \circ B = (A \ominus B) \oplus B$	Smoothes contours, breaks narrow isthmuses, and eliminates small islands and sharp peaks. (I)
Closing	$A \bullet B = (A \oplus B) \ominus B$	Smoothes contours, fuses narrow breaks and long thin gulfs, and eliminates small holes. (I)
Boundary extraction	$\beta(A) = A - (A \ominus B)$	Set of points on the boundary of set $A$ . (I)
Region filling	$X_k = (X_{k-1} \oplus B) \cap A^c; X_0 = p$ and $k = 1, 2, 3, \dots$	Fills a region in $A$ , given a point $p$ in the region. (II)
Connected components	$X_k = (X_{k-1} \oplus B) \cap A; X_0 = p$ and $k = 1, 2, 3, \dots$	Finds a connected component $Y$ in $A$ , given a point $p$ in $Y$ . (I)

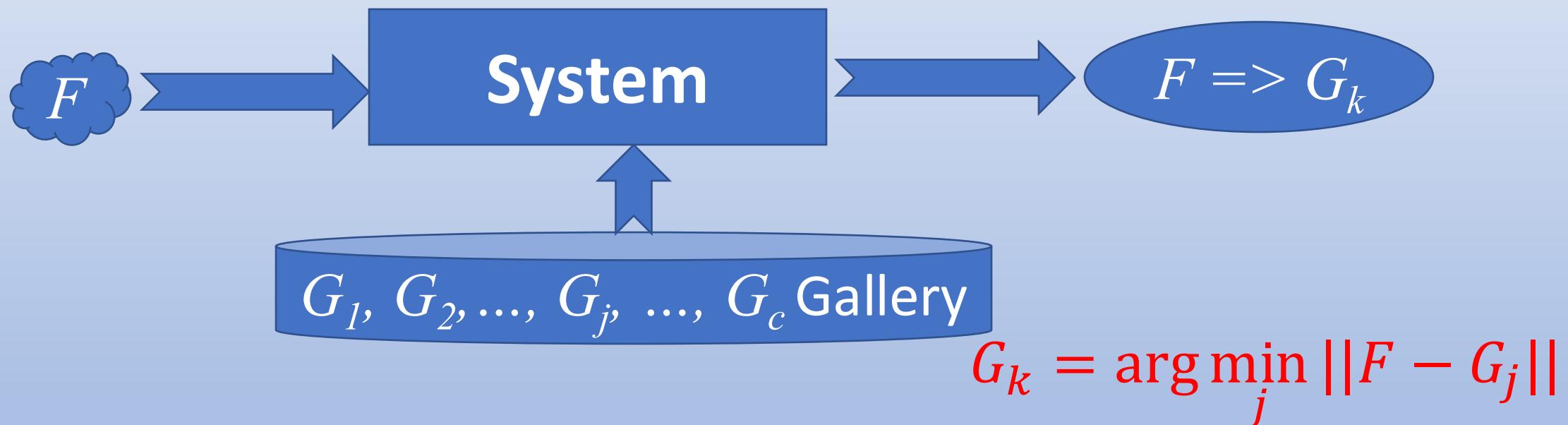
# Topic 5

**Intuitive Understanding of  
Visual Object Recognition:**

**from Matching to Classification  
to Machine Learning**

# Template Matching, Distance and Similarity

- How a system recognize/identify a query object  $F$  ?
- A system has  $c$  different objects  $G_j, j=1, 2, \dots, c$  in its gallery/database.



- Given an unknown query object  $F$ , the system needs to recognize/identify it originating from the same source of which gallery object  $G_j, j = 1, 2, \dots, c$ .

# Template Matching, Distance and Similarity

$$G_k = \arg \min_j ||F - G_j||$$

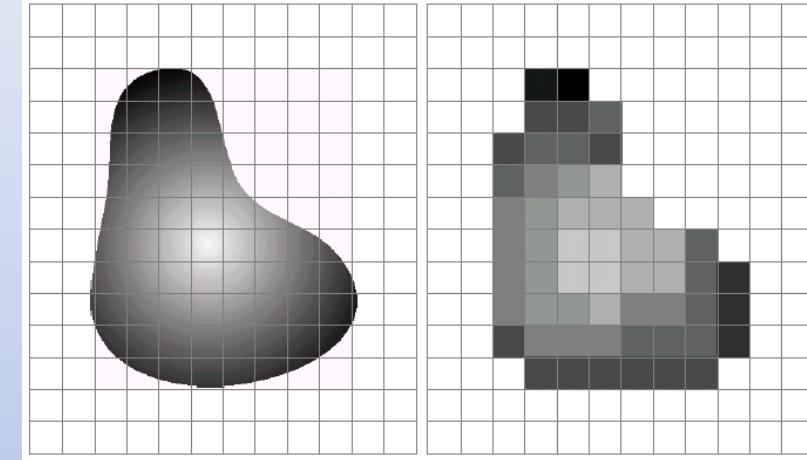
is called template matching that compares the unknown query object with each of the known gallery objects or template objects to find the matched gallery object.

- This needs to compute distance or the similarity between two objects

$$d_j = ||F - G_j||, \quad \text{for } j = 1, 2, \dots, c$$

# Template Matching, Distance and Similarity

A visual object represented by a gray level image is specified by the different brightness at different  $x$ - and  $y$ -positions.



A gray level image is thus a function of  $x$ , and  $y$ ,  $f(x, y)$ , which can be represented by a matrix.

$$f(x, y) = \text{e.g.: } \sin[2\pi(u \sin(\phi)x + v \sin(\phi)y)]$$
$$\begin{bmatrix} f(1,1) & f(1,2) & \cdots & f(1,n) \\ f(2,1) & f(2,2) & \cdots & f(2,n) \\ \vdots & \vdots & \ddots & \vdots \\ f(m,1) & f(m,2) & \cdots & f(m,n) \end{bmatrix} \text{ or } \mathbf{F} = \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1n} \\ f_{21} & f_{22} & \cdots & f_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ f_{m1} & f_{m2} & \cdots & f_{mn} \end{bmatrix}$$

# Template Matching, Distance and Similarity

- The Euclidian distance between two visual objects  $F$  and  $G$  represented by two images  $f(x, y)$  and  $g(x, y)$  of size  $p$  by  $q$  is defined as (*Note that the square root is omitted for symbolic simplicity.*)

$$d = ||F - G|| = \sum_{y=1}^q \sum_{x=1}^p [f(x, y) - g(x, y)]^2$$

- For convenience, we convert the  $p$  by  $q$  matrices  $F$  and  $G$  into  $n$ -dimensional column vectors  $\mathbf{f}$  and  $\mathbf{g}$ ,  $n=pq$ .

for example,  $p = q = 3$ , then  $n = 3 \times 3 = 9$ :

$$\mathbf{F} = \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 0 & 1 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \quad \longrightarrow \quad \mathbf{f} = (1, 1, 1, 1, 0, 1, 0, 0, 0)^T$$

# Template Matching, Distance and Similarity

- Then we can express the (squared) Euclidian distance convenient as:  $d = ||F - G|| = ||\mathbf{f} - \mathbf{g}||^2 = (\mathbf{f} - \mathbf{g})^T(\mathbf{f} - \mathbf{g})$
- This is the convenient vector difference, transpose and multiplication.
- However, does the Euclidian distance between two images really show the similarity of the two visual objects?
- Compute the Euclidian distance of the following two images

1	1	1
1	0	1
0	0	0

100	100	100
100	10	100
10	10	10

# Template Matching, Distance and Similarity

- From the two images:

1	1	1
1	0	1
0	0	0

100	100	100
100	10	100
10	10	10

- We have

$$\mathbf{f} = (1, 1, 1, 1, 0, 1, 0, 0, 0)^T$$

$$\mathbf{g} = (100, 100, 100, 100, 10, 100, 10, 10, 10)^T$$

- The Euclidian distance  $d = (\mathbf{f} - \mathbf{g})^T(\mathbf{f} - \mathbf{g})$  is huge though the both images show the same triangle pattern, only with different brightness and contrast.
- How to solve this problem?

# Template Matching, Distance and Similarity

- We normalize the images into zero mean and unit variance.
- From  $\mathbf{f} = (1, 1, 1, 1, 0, 1, 0, 0, 0)^T$
- We have its mean  $m_f = 5/9$ .
  
- So we have centralized vector

$$\begin{aligned}\mathbf{f}_c &= \mathbf{f} - m_f \\ &= (1 - 5/9, 1 - 5/9, 1 - 5/9, 1 - 5/9, 0 - 5/9, 1 - 5/9, 0 - 5/9, 0 - 5/9, 0 - 5/9)^T \\ &= \left( \frac{4}{9}, \frac{4}{9}, \frac{4}{9}, \frac{4}{9}, -\frac{5}{9}, \frac{4}{9}, -\frac{5}{9}, -\frac{5}{9}, -\frac{5}{9} \right)^T\end{aligned}$$

# Template Matching, Distance and Similarity

- The standard deviation of the image or the length of the vector is

$$\|\mathbf{f}_c\| = \sqrt{\frac{1}{9} \mathbf{f}_c^T \mathbf{f}_c} = \sqrt{\frac{1}{9} \left( \left(\frac{4}{9}\right)^2 + \left(\frac{4}{9}\right)^2 + \dots \right)} = \frac{\sqrt{20}}{9}$$

- We normalize the images into zero mean and unit variance/length.

$$\mathbf{f}_n = \frac{\mathbf{f}_c}{\|\mathbf{f}_c\|} = \frac{9}{\sqrt{20}} \left( \frac{4}{9}, \frac{4}{9}, \dots \right)^T = \frac{1}{\sqrt{20}} (4, 4, 4, 4, -5, \dots)^T$$

- Note that  $\mathbf{f}_n$  has zero mean and unit variance.

# Template Matching, Distance and Similarity

- Repeat the same process for

$$\mathbf{g} = (100, 100, 100, 100, 10, 100, 10, 10, 10)^T$$

- We have its mean  $m_g = 60$ . So we have centralized

$$\mathbf{g}_c = \mathbf{g} - m_g = (100 - 60, 100 - 60, \dots, 10 - 60)^T = (40, 40, \dots, -50)^T$$

- The standard deviation or the length of the vector is then

$$\|\mathbf{g}_c\| = \sqrt{\frac{1}{9}(\mathbf{g}_c^T \mathbf{g}_c)} = \sqrt{\frac{1}{9}(40^2 + 40^2 + \dots)} = \sqrt{2000}$$

- We normalize the images into zero mean and unit variance.

$$\mathbf{g}_n = \frac{\mathbf{g}_c}{\|\mathbf{g}_c\|} = \frac{1}{\sqrt{20}} (4, 4, 4, 4, -5, \dots)^T = \mathbf{f}_n$$

# Template Matching, Distance and Similarity

- From the two images:

1	1	1
1	0	1
0	0	0

100	100	100
100	10	100
10	10	10

$$\mathbf{g} = (100, 100, 100, 100, 10, 100, 10, 10, 10)^T$$

$$\mathbf{f} = (1, 1, 1, 1, 0, 1, 0, 0, 0)^T$$

- The Euclidian distance of the normalized images

$$d_n = (\mathbf{f}_n - \mathbf{g}_n)^T (\mathbf{f}_n - \mathbf{g}_n) = 0$$

1	2	3
3	2	1
2	3	4

- Try another image,  $H$ :
- Thus,  $d_n$  is much better than  $d$  as distance for template matching.

$$d_n = (\mathbf{f}_n - \mathbf{h}_n)^T (\mathbf{f}_n - \mathbf{h}_n) \gg 0$$

# Template Matching, Distance and Similarity

- We define the correlation coefficient:

$$\gamma = \mathbf{f}_n^T \mathbf{g}_n = \frac{(\mathbf{f} - m_f)^T (\mathbf{g} - m_g)}{\|\mathbf{f}_c\| \|\mathbf{g}_c\|}$$

- For these two images  $F$  and  $G$ , we have  $\gamma = 1$ .

1	1	1
1	0	1
0	0	0

100	100	100
100	10	100
10	10	10

- The correlation coefficient of either  $F$  or  $G$  with  $H$  will produce  $-1 < \gamma < 1$ . In fact, for all images,  $-1 \leq \gamma \leq 1$ .

1	2	3
3	2	1
2	3	4

- Thus,  $\gamma$  can be used as similarity for template matching.

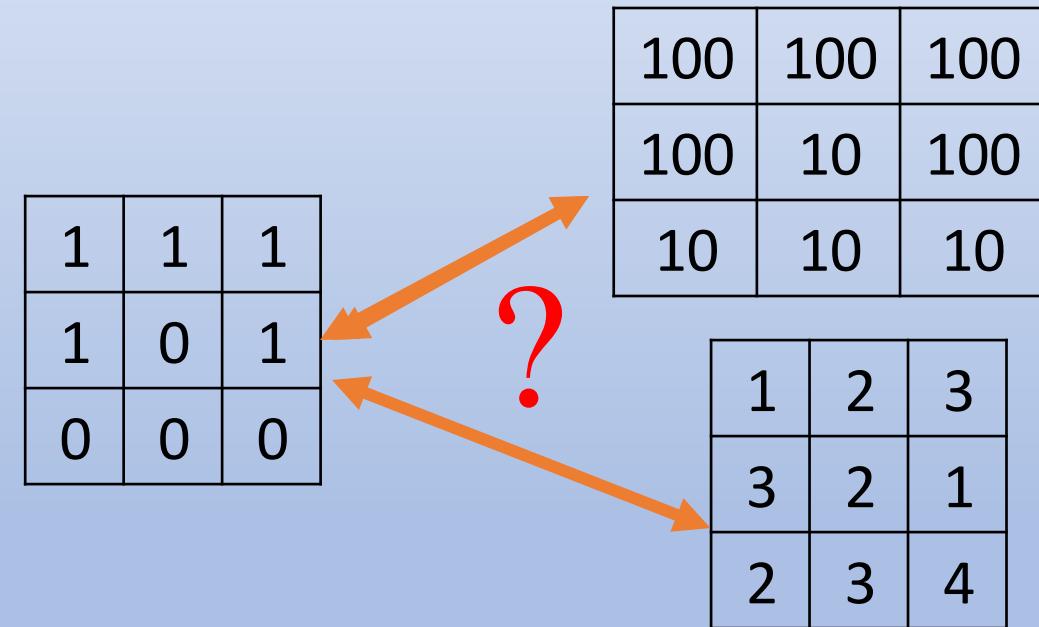
# Template Matching, Distance and Similarity

- We have learnt how to properly compare two visual objects through proper normalization to recognize the unknown query image.
- We can use Euclidian distance of normalized images

$$d_n = (\mathbf{f}_n - \mathbf{g}_n)^T (\mathbf{f}_n - \mathbf{g}_n)$$

- Or correlation coefficient

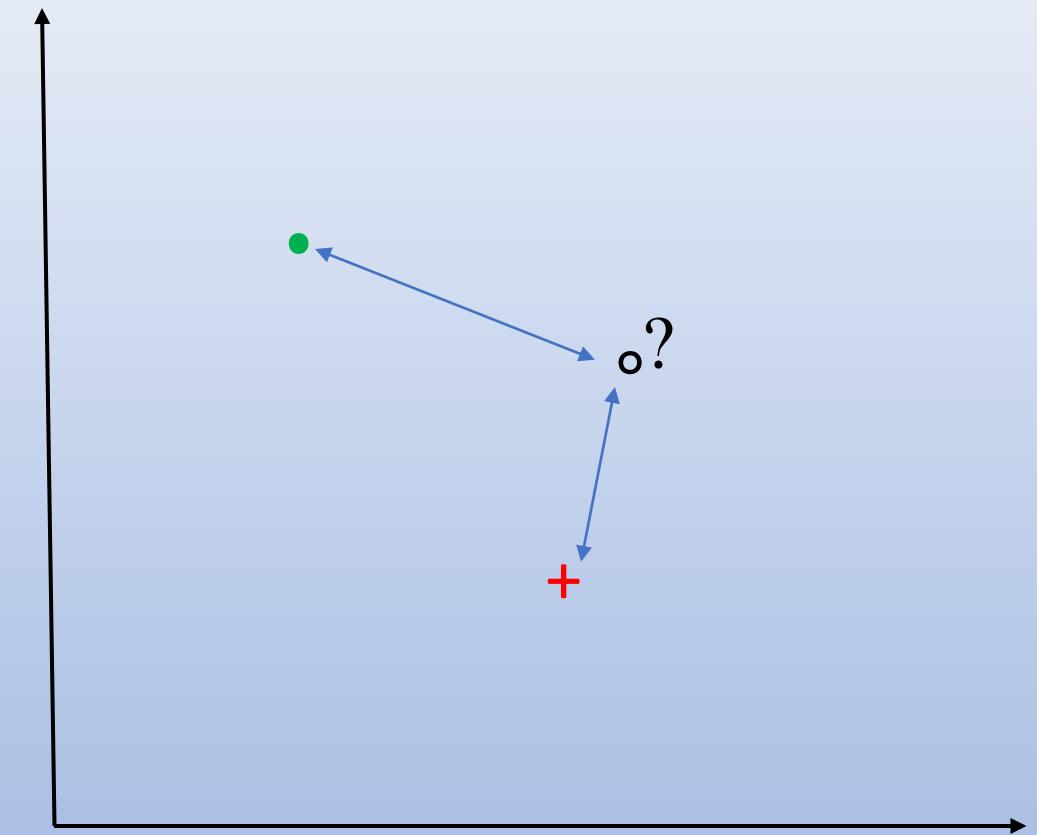
$$\gamma = \mathbf{f}_n^T \mathbf{g}_n = \frac{(\mathbf{f} - m_f)^T (\mathbf{g} - m_g)}{\|\mathbf{f}_c\| \|\mathbf{g}_c\|}$$



- Other normalizations? Scale, position, rotation variation.

# From Template Matching to Classification

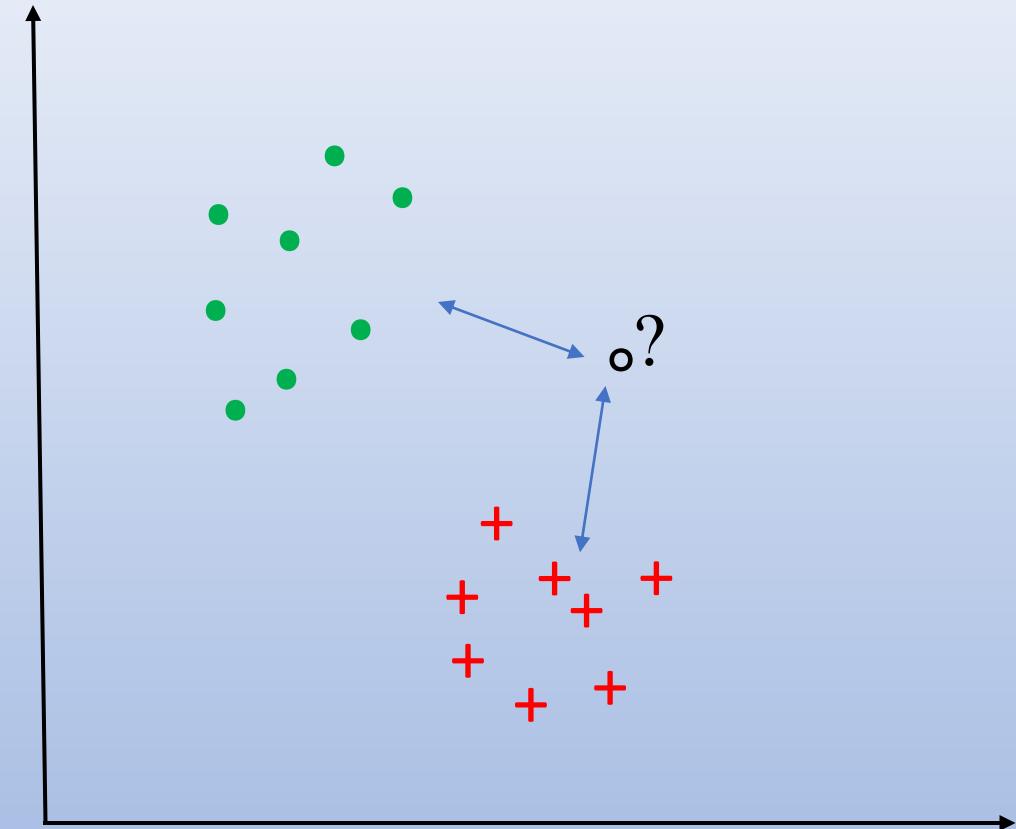
- We have represented an image object by an  $n$ -dimensional column vector that contains all pixel gray values, one vector element for one pixel, from a  $p$  by  $q$  matrix to an  $n$ -dimensional vector,  $n=pq$ .
- An  $n$ -dimensional vector is a point in an  $n$ -dimensional space.
- To visualize the process, take  $n=c=2$  and use distance, as shown in Figure.



$$G_k = \arg \min_j \|\mathbf{f} - \mathbf{g}_j\|$$

# From Template Matching to Classification

- However, an object has many variants.  
It is not sufficient to represent an object by just a single image.
- Each object should be represented by many images.
- Each object is thus called a class, each image is called a sample of the class.
- Recognizing/identifying an unknown query object as one of the  $c$  known gallery objects is thus converted to **classifying** it into one of the  $c$  classes.



$$G_k = \arg \min_j ||\mathbf{f} - \mathbf{g}_j|| ?$$

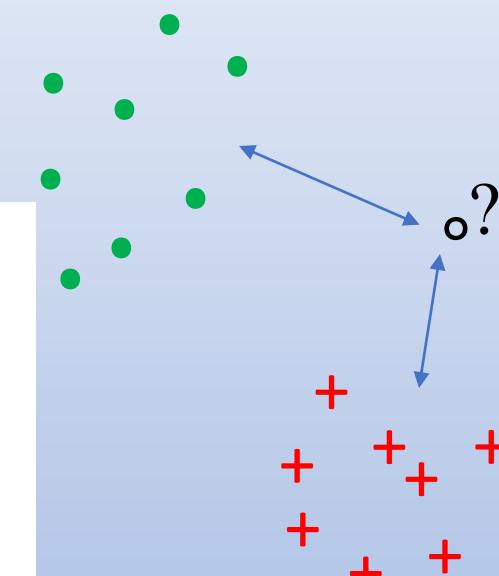
# From Template Matching to Classification

- How to Classify an unknown query object as one of the  $c$  known classes?
- 1<sup>st</sup> Nearest Neighbor (NN) Classifier:

- Denote by  $\mathcal{D}^n = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$  a set of labeled **prototypes** or training samples.
- Let  $\mathbf{x}^*$  be the prototype nearest to  $\mathbf{x}$ .
- Then the **nearest-neighbor (NN)** rule for classifying  $\mathbf{x}$  is to assign it the label associated with  $\mathbf{x}^*$ .
- More formally if we have a set of labeled training samples  $\{(\mathbf{x}_1, \theta_1), \dots, (\mathbf{x}_n, \theta_n)\}$ , where each  $\theta_i$  is one of the labels  $\omega_1, \dots, \omega_c$ , then the NN decision rule is

$$\alpha_{nn}(\mathbf{x}) = \theta_k : k = \arg \min_i \|\mathbf{x} - \mathbf{x}_i\|.$$

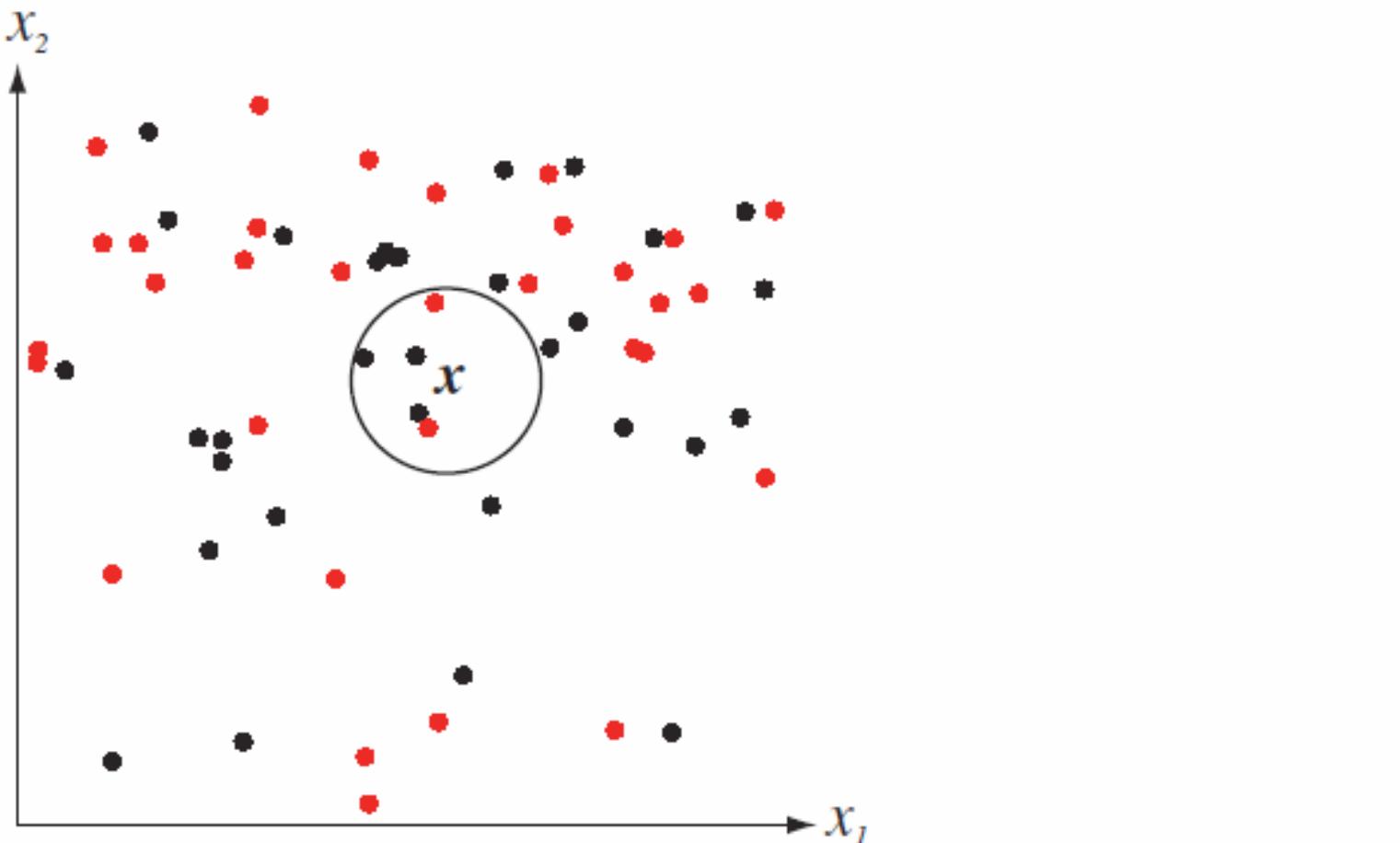
$$G_k = \arg \min_j \|\mathbf{f} - \mathbf{g}_j\| ?$$



# Nearest Neighbor (NN) Classifier

- Generalization of the NN rule.
- The  $k_n$  nearest neighbors rule: Given a set of training samples  $\{x_1, \dots, x_n\}$  and a test point  $x$ , find  $k$  training points closest to  $x$ ,  $x_1^*, \dots, x_k^*$ . Collect the labels associated  $\theta_1^*, \dots, \theta_k^*$  and classify  $x$  to the class which has the greatest number of representatives in  $\theta_1^*, \dots, \theta_k^*$ .
- In other words, the classification is performed by taking the majority vote among  $k$  nearest neighbors of  $x$ .

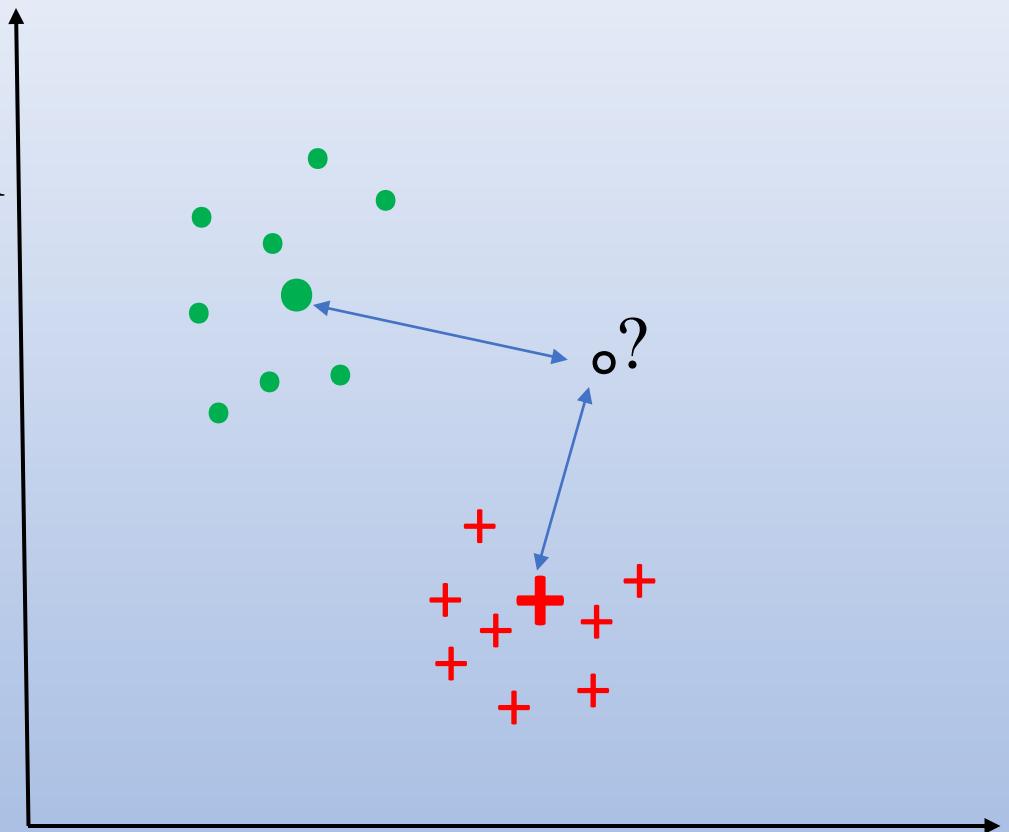
# Nearest Neighbor (NN) Classifier



**FIGURE 4.15.** The  $k$ -nearest-neighbor query starts at the test point  $x$  and grows a spherical region until it encloses  $k$  training samples, and it labels the test point by a majority vote of these samples. In this  $k = 5$  case, the test point  $x$  would be labeled the category of the black points. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern*

# From Sample Comparison to Machine Learning

- However, some class may have thousands or millions samples. It is tedious to compare query sample with every training samples. NN classifier also over-fits the training data or has poor generalization.
- How to solve the problems?
- One solution is the minimum distance classifier that compares the distances of query sample to each class mean vector  $\mu_j$ .



$$G_k = \arg \min_j \| \mathbf{f} - \boldsymbol{\mu}_j \|$$

# From Sample Comparison to Machine Learning

- Given  $q$   $n$ -dimensional training samples of  $c$  classes

$$\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_i, \dots, \mathbf{x}_q,$$

- The number of training samples of class  $\omega_j$  is  $q_j$ ,  $j = 1, 2, \dots, c$ .
- The mean vector of class  $\omega_j$  is computed as

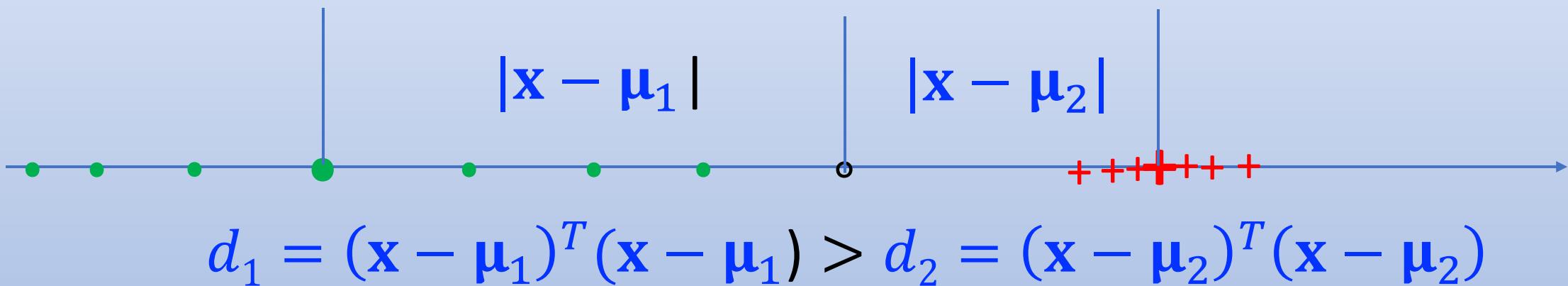
$$\boldsymbol{\mu}_j = \frac{1}{q_j} \sum_{\mathbf{x}_i \in \omega_j} \mathbf{x}_i$$

- Before receiving query sample  $\mathbf{x}$ , the machine compute some parameters from training data. This is **Machine Learning**.
- The minimum Euclidian distance classifier classifies unknown query sample  $\mathbf{x}$  by:

$$\omega_k = \arg \min_j (\mathbf{x} - \boldsymbol{\mu}_j)^T (\mathbf{x} - \boldsymbol{\mu}_j)$$

# From Sample Comparison to Machine Learning

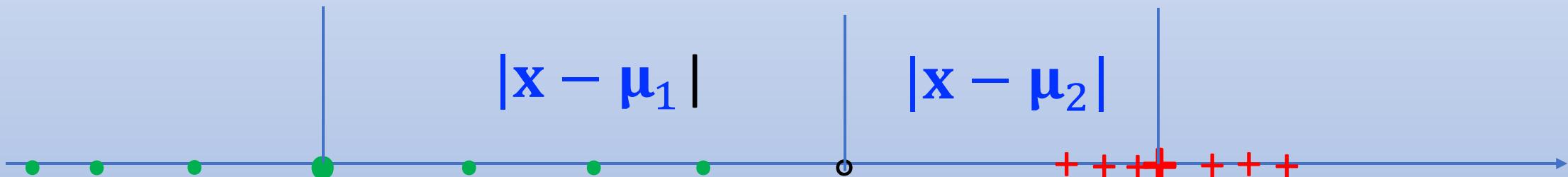
- However, only use the class mean vector to represent a class is not sophisticated.
- Visualize the problem by a 1-D example:



- Should  $\mathbf{x}$  definitely be classified into class 2 (red)?
- Most likely wrong! What is the problem?

# From Sample Comparison to Machine Learning

- Taken the 1-D data as an example, besides the class mean, we should also consider the class standard deviation or variance.
- A solution is to normalize the Euclidian distance by the class standard deviation or normalize the squared Euclidian distance by the class variance.



$$d_1 = (\mathbf{x} - \boldsymbol{\mu}_1)^T (\mathbf{x} - \boldsymbol{\mu}_1) > d_2 = (\mathbf{x} - \boldsymbol{\mu}_2)^T (\mathbf{x} - \boldsymbol{\mu}_2)$$

$$d_{Ma1} = \frac{(\mathbf{x} - \boldsymbol{\mu}_1)^T (\mathbf{x} - \boldsymbol{\mu}_1)}{\sigma_1^2} < d_{Ma2} = \frac{(\mathbf{x} - \boldsymbol{\mu}_2)^T (\mathbf{x} - \boldsymbol{\mu}_2)}{\sigma_2^2}$$

# From Sample Comparison to Machine Learning

- For multiple dimensional data, however, besides mean and variance of each element of the data vector, we have covariance between different elements of the data vector.
- Given  $q$   $n$ -dimensional training samples of  $c$  classes

$$\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_i, \dots, \mathbf{x}_q,$$

- The number of training samples of class  $\omega_j$  is  $q_j$ ,  $j = 1, 2, \dots, c$ .
- The mean vector and covariance matrix of class  $\omega_j$  is computed as

$$\boldsymbol{\mu}_j = \frac{1}{q_j} \sum_{x_i \in \omega_j} \mathbf{x}_i$$

$$\begin{aligned}\boldsymbol{\Sigma}_j &= \frac{1}{q_j} \sum_{x_i \in \omega_j} (\mathbf{x}_i - \boldsymbol{\mu}_j)(\mathbf{x}_i - \boldsymbol{\mu}_j)^T \\ &= \frac{1}{q_j} \sum_{x_i \in \omega_j} \mathbf{x}_i \mathbf{x}_i^T - \boldsymbol{\mu}_j \boldsymbol{\mu}_j^T\end{aligned}$$

# From Sample Comparison to Machine Learning

- To have the distance normalized by the data variation, the square of Euclidian distance:

$$d_{Eu_j} = \|\mathbf{x} - \boldsymbol{\mu}_j\|^2 = (\mathbf{x} - \boldsymbol{\mu}_j)^T (\mathbf{x} - \boldsymbol{\mu}_j)$$

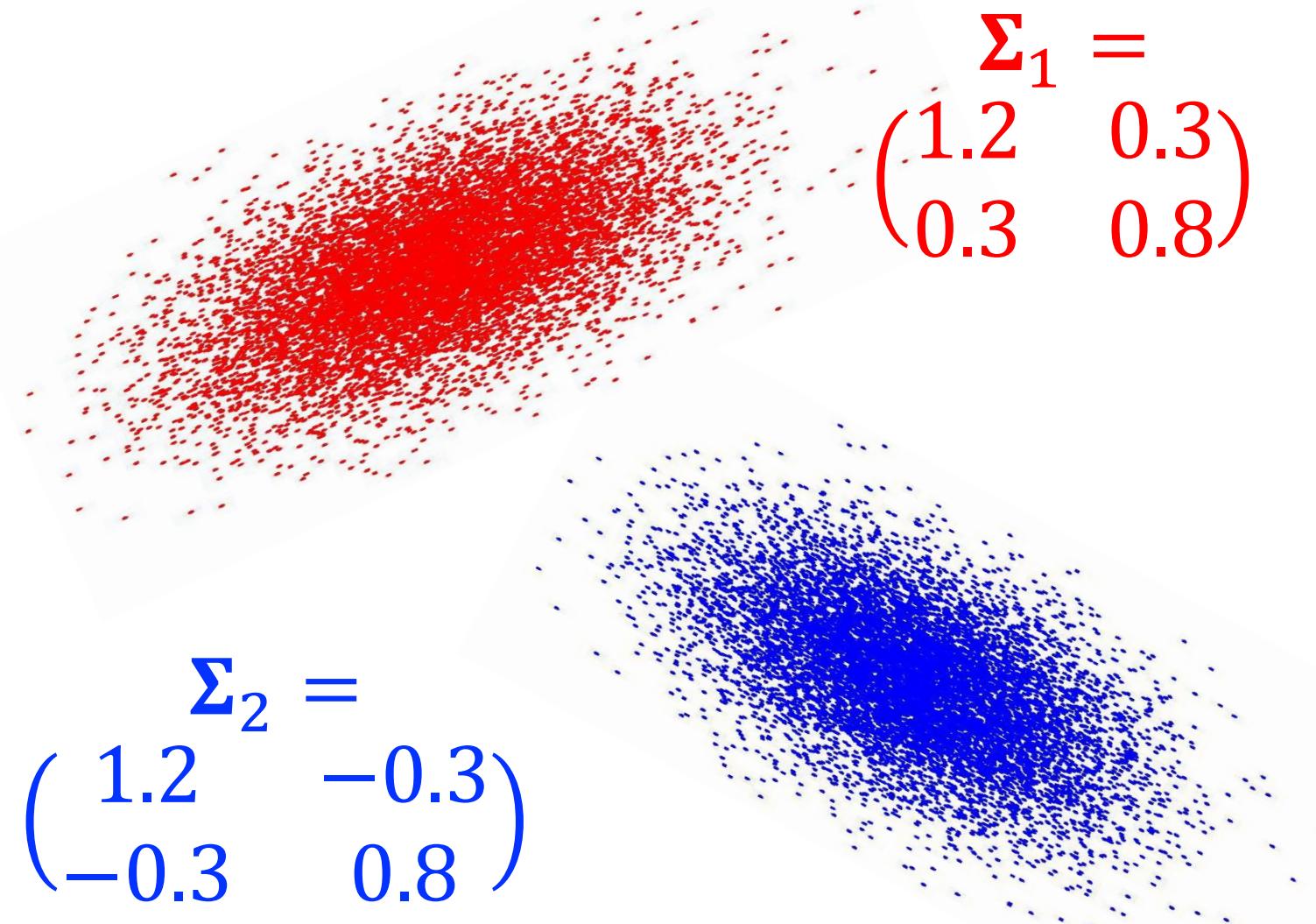
is weighted by the inverse of the covariance matrix by

$$d_{Maj} = (\mathbf{x} - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} (\mathbf{x} - \boldsymbol{\mu}_j)$$

- Called (squared) Mahalanobis distance.
- Minimum Mahalanobis distance classifier is the optimal classifier if the data of each class obeys Gaussian PDF.

# From Sample Comparison to Machine Learning

- The figure shows the data variation or distribution of a 'red' class and a 'blue' class in 2-dimension case.
- The shape and size of the data variation or distribution is determined by the covariance matrix.



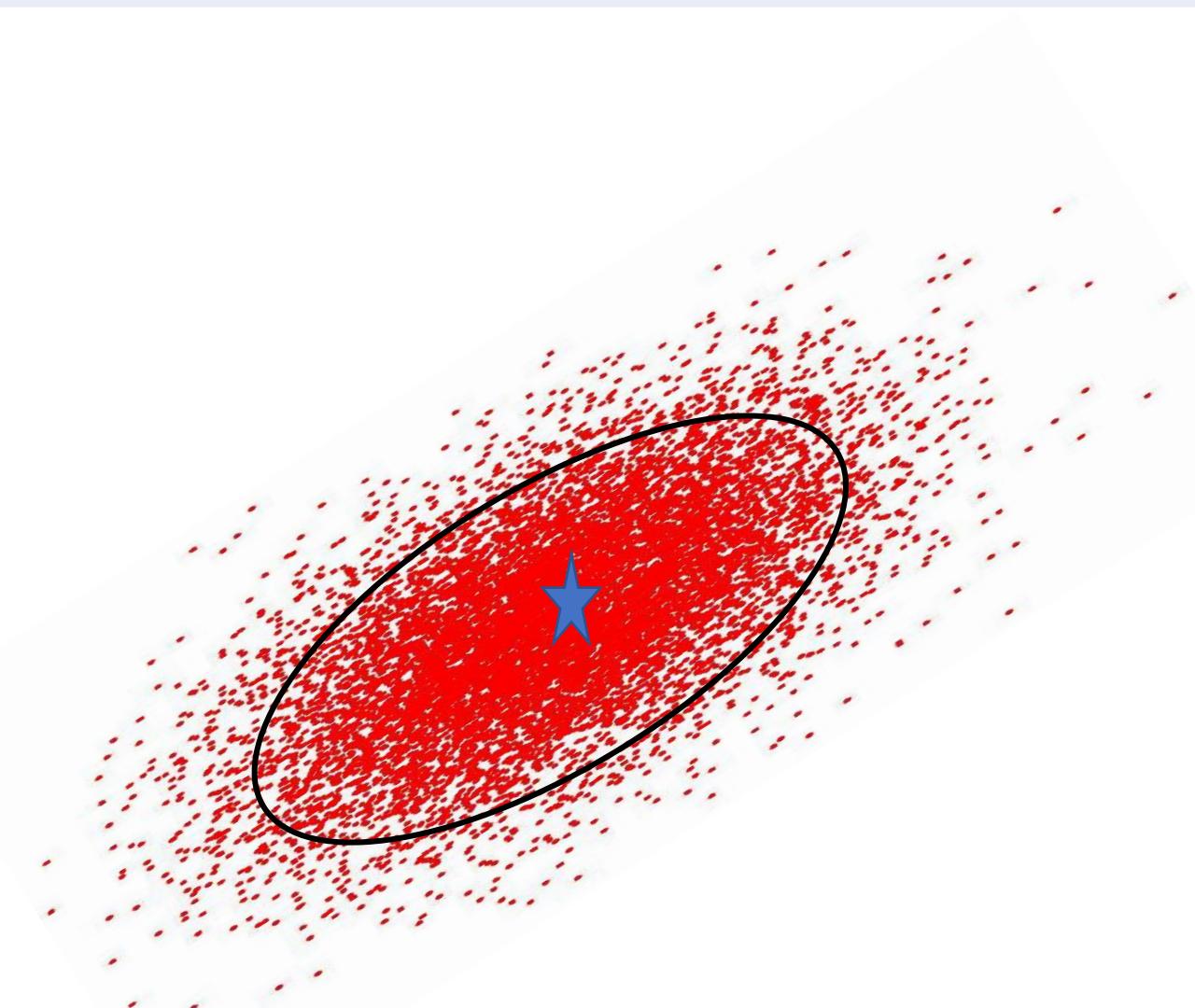
# From Sample Comparison to Machine Learning

- All points along the ellipse shaped by the covariance matrix has the same distance to its center. It agrees to the contour of Gaussian PDF

$$d_{Maj} = (\mathbf{x} - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}_j^{-1} (\mathbf{x} - \boldsymbol{\mu}_j)$$

How about Euclidian distance?

$$\begin{aligned} d_{Euj} &= \|\mathbf{x} - \boldsymbol{\mu}_j\|^2 \\ &= (\mathbf{x} - \boldsymbol{\mu}_j)^T (\mathbf{x} - \boldsymbol{\mu}_j) \end{aligned}$$



# From Sample Comparison to Machine Learning

- If we denote the centralized training samples by

$$\tilde{\mathbf{x}}_i = \mathbf{x}_i - \boldsymbol{\mu}_j$$

- And put all column vector of training samples of a class in a matrix:

$$\mathbf{X}_j = (\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \dots, \tilde{\mathbf{x}}_i, \dots, \tilde{\mathbf{x}}_{q_j})$$

- Then we can express:

$$\boldsymbol{\Sigma}_j = \frac{1}{q_j} \sum_{x_i \in \omega_j} (\mathbf{x}_i - \boldsymbol{\mu}_j)(\mathbf{x}_i - \boldsymbol{\mu}_j)^T = \frac{1}{q_j} \mathbf{X}_j \mathbf{X}_j^T$$

# From Sample Comparison to Machine Learning

- For some classification problem, we may have large number of classes but small number of training samples per class. The covariance matrix computed by few training samples of single class is extremely unreliable. So we average covariance matrices over all classes, called pooled class covariance matrix, also called within-class scatter matrix.

$$\Sigma^w = \frac{1}{q} \sum_{j=1}^c q_j \Sigma_j$$

# From Sample Comparison to Machine Learning

- We have seen that the covariance matrix represents a rich information of the data variation or distribution.

- Besides

$$\Sigma_j = \frac{1}{q_j} \mathbf{X}_j \mathbf{X}_j^T \quad \Sigma^w = \frac{1}{q} \sum_{j=1}^c q_j \Sigma_j = \mathbf{S}^w$$

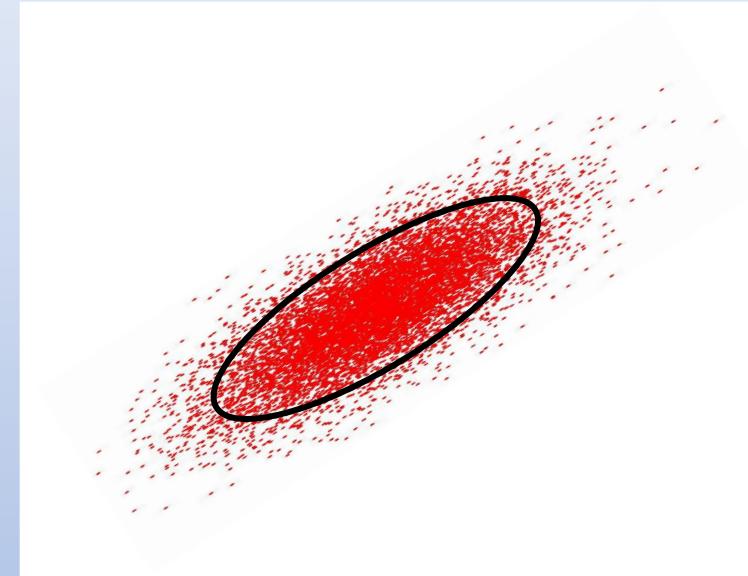
- We define more covariance matrices as

$$\mathbf{S}^b = \frac{1}{q} \sum_{j=1}^c q_j (\boldsymbol{\mu}_j - \boldsymbol{\mu})(\boldsymbol{\mu}_j - \boldsymbol{\mu})^T$$

$$\mathbf{S}^t = \frac{1}{q} \sum_{i=1}^q (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T = \frac{1}{q} \mathbf{X} \mathbf{X}^T$$

$$\boldsymbol{\mu} = \frac{1}{q} \sum_{i=1}^q \mathbf{x}_i$$

- $\mathbf{X}$  is the data matrix of all samples centralized to  $\boldsymbol{\mu}$ .

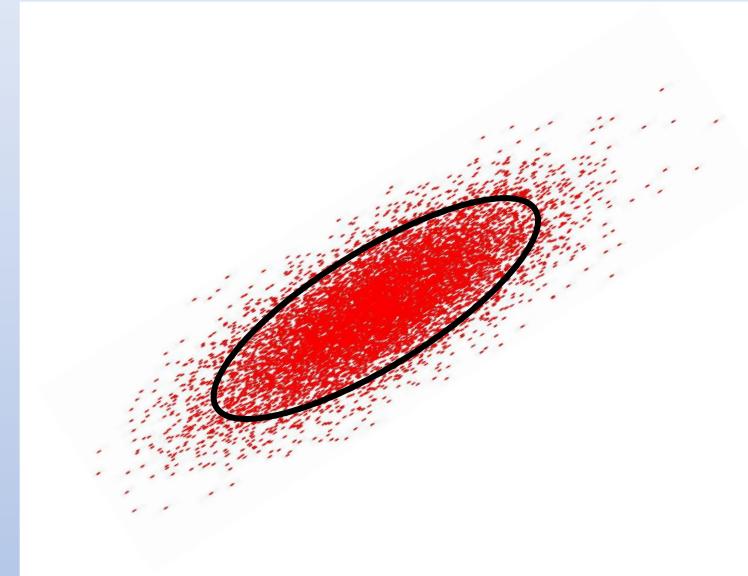


# From Sample Comparison to Machine Learning

- These covariance matrices have following relationship:

$$\mathbf{S}^t = \mathbf{S}^w + \mathbf{S}^b = \frac{1}{q} \sum_{j=1}^c q_j \boldsymbol{\Sigma}_j + \mathbf{S}^b$$

- $\mathbf{S}^t$  is called data total scatter matrix,  $\mathbf{S}^w$  called within class scatter matrix and  $\mathbf{S}^b$  called between class scatter matrix.



# **Topic 6**

## **MAP Decision and Classifiers**

# MAP Decision and Classifiers

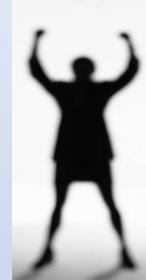
## Outline

- Understand how the decision theory is developed from an intuitive process that everybody understands to an abstract mathematical theory
- The optimal decision rule: MAP and Bayesian decision rules
- Evaluation a pattern recognition system: Recognition accuracy or error rate
- Generalize the classification process to the discriminant function evaluation
- Discriminant function for multivariate Gaussian PDF
- Mahalanobis distance and Euclidean distance
- Linear classifier

# How to make the best decision in uncertainty?

Let's see a simple example:

- You totally unclearly see a student in a classroom of school of EEE. You need decide/judge if this student is a male or a female. What is your decision?
- if you are not silly you will judge that the student is a male based on the fact that the student population in the school of EEE is 70% for male and 30% for female.
- Is your decision the same in a classroom of school of accounting, supposing the student population in the school of accounting is 30% for male and 70% for female?
- How do you work out your decision or describe your decision rule or justify it is a good decision scientifically or theoretically or mathematically?
- Mathematically, this is the **maximum probability** decision based on our **knowledge** that  $p(\omega_1) = 0.7$  and  $p(\omega_2) = 0.3$  for a male and female in the school of EEE, and that  $p(\omega_1) = 0.3$  and  $p(\omega_2) = 0.7$  for a male and female in the school of accounting, respectively.
- So a wise decision rule is to decide  $\omega_1$  if  $p(\omega_1) > p(\omega_2)$ , and vice versa.
- Is your decision definitely correct? No. But is there any other better decision?
- No. The probability of the wrong decision (error rate) is  $p(e) = 0.3$ , which is the minimum.
- What is the error rate for the opposite decision?



# MAP Decision and Classifiers

- Now if you know the student in the classroom of school of accounting is of height 1.75, how do you make the best decision?
- No difference. The best decision is still based on  $p(\omega_1)$  and  $p(\omega_2)$  to see which is larger.
- So simple? Yes! There is no magic in our world. But what is  $p(\omega_i)$ ,  $i = 1, 2$ , now?
- Same as that  $p(\omega_i)$  of the school of accounting are different from those of EEE,  $p(\omega_i)$  including the knowledge of height of 1.75 are different from those without this knowledge.
- For convenience, they are mathematically denoted by  $p(\omega_1|x)$  and  $p(\omega_2|x)$ ,  $x = 1.75$ , called **a posterior probability**. It means **the probability of a class  $\omega_i$  after knowing the data value  $x$**  in a particular scenario.  $p(\omega_1)$  and  $p(\omega_2)$  without knowing the value of  $x$  are then renamed as **the prior probability**.
- The name of decision rule is modified from the **maximum probability** decision to the **maximum a posterior probability** decision, **MAP decision** in short.
- Thus, in general, we have **MAP decision rule**:  
Decide  $\omega_k$ : if  $p(\omega_k|x) > p(\omega_i|x), i \neq k$   
or more general:  $\omega_k = \arg \max_{\omega_i} [p(\omega_i|x)]$

# MAP Decision and Classifiers

- As you decide  $\omega_k$ , based on your observed value  $x$ , the probability that you make a correct decision is hence  $p(\omega_k|x)$ . Therefore, the probability that you make a wrong decision or error rate is:

$$p(e_k|x) = 1 - p(\omega_k|x)$$

- If your decision is  $\omega_k$ , because  $p(\omega_k|x)$  is maximum, consequently, the probability of the decision error will be minimum.
- Therefore, this decision rule is optimal in the sense of minimizing the probability of the wrong decision.
- The maximum a posterior (MAP) decision rule minimizes the probability of the decision error.

Decide:  $\omega_k = \arg \max_{\omega_i} [p(\omega_i|x)] = \arg \min_{\omega_i} [p(e_i|x)]$

# MAP Decision and Classifiers

- However, it is not very strait forward to get the value of  $p(\omega_i|x)$ . It is much easier and more convenient to get the values of  $p(\omega_i)$  and  $p(x|\omega_i)$ , called the class-conditional probability.
- From the basic rule of probability theory:

$$p(x, \omega_i) = p(x)p(\omega_i|x) = p(\omega_i)p(x|\omega_i)$$

$$p(\omega_i|x) = \frac{p(\omega_i)p(x|\omega_i)}{p(x)}$$

$p(A, B)$   
 $= p(A)p(B|A)$   
 $= p(B)p(A|B)$

We have

where

$$p(x) = \sum_{i=1}^c p(x|\omega_i)p(\omega_i)$$

- $p(x)$  is called mixture PDF or PMF, the probability (density) of a person's height at  $x$ . We can remove it from the decision as it has the same value for all classes.

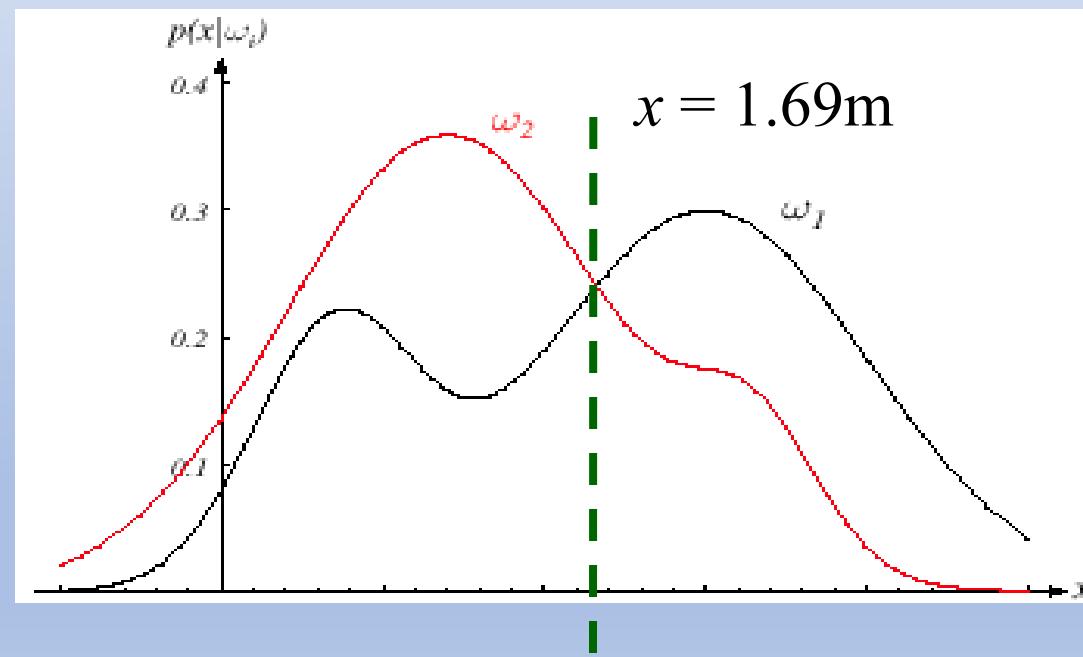
# MAP Decision and Classifiers

- Now we know how to make the optimal decision/recognition at a specific value,  $x=1.75\text{m}$ , and compute the error rate. The concept is intuitive and easy to understand.
- However, to fully automatically recognize a person's gender for all value of  $x$ , we need the class conditional probability  $p(x|\omega_i)$ , of all possible value of  $x$ . This brings big challenge.

$$\sum_{x=0}^3 p(x|\omega_1) = 1, \text{ and } \sum_{x=0}^3 p(x|\omega_2) = 1 \text{ for discrete } x$$

$$\int_0^3 p(x|\omega_1) dx = 1, \text{ and } \int_0^3 p(x|\omega_2) dx = 1 \text{ for continuous } x$$

Decide:  $\omega_k = \arg \max_{\omega_i} [p(\omega_i|x)]$



- Is it possible to simplify the computation of the system? Convert decision into classification.

## MAP Decision and Classifiers

- How good is the system? Or what is the performance of the system? Or how to evaluate your designed system?

Decide:  $\omega_k = \arg \max_{\omega_i} [p(\omega_i|x)]$

- We have understand the formula to compute the error probability for a specific value of  $x$ .

$$p(e_k|x) = 1 - p(\omega_k|x) = \sum_{\substack{i=1 \\ i \neq k}}^c p(\omega_i|x)$$

- It is not difficult to understand that the performance of a recognition system can be measured by the average of  $p(e_k|x)$  over all possible value of  $x$ .

# MAP Decision and Classifiers

- How to average  $p(e_k|x)$  over all possible value of  $x$ ?

$$p(e)? = \frac{1}{n_x} \sum_{x=0}^3 p(e_k|x) \text{ for discrete } x$$

$$p(e)? = \frac{1}{3} \int_0^3 p(e_k|x) dx, \quad \text{for continuous } x$$

- The above is not a proper way because  $x$  is a random variable with different probability of occurrence at different  $x$  value.
- The right way for a random variable should be:

$$p(e) = \sum_{x=-\infty}^{\infty} p(e_k|x) p(x) \text{ for discrete } x$$

$$p(e) = \int_{-\infty}^{\infty} p(e_k|x) p(x) dx, \quad \text{for continuous } x$$

# MAP Decision and Classifiers

- Let's take the continuous  $x$  for further working:

$$\begin{aligned} p(e) &= \int_{-\infty}^{\infty} p(e_k|x)p(x)dx = \int_{-\infty}^{\infty} [1 - p(\omega_k|x)]p(x)dx \\ &= \int_{-\infty}^{\infty} \left[1 - \frac{p(\omega_k)p(x|\omega_k)}{p(x)}\right]p(x)dx = 1 - \int_{-\infty}^{\infty} p(\omega_k)p(x|\omega_k)dx \end{aligned}$$

- Note that for different region of  $x$ , the pattern recognition system has different decision  $\omega_k$ . Decision/classification is to partition the whole space of  $x$  into  $c$  decision regions  $\mathcal{R}_i$ . Therefore,

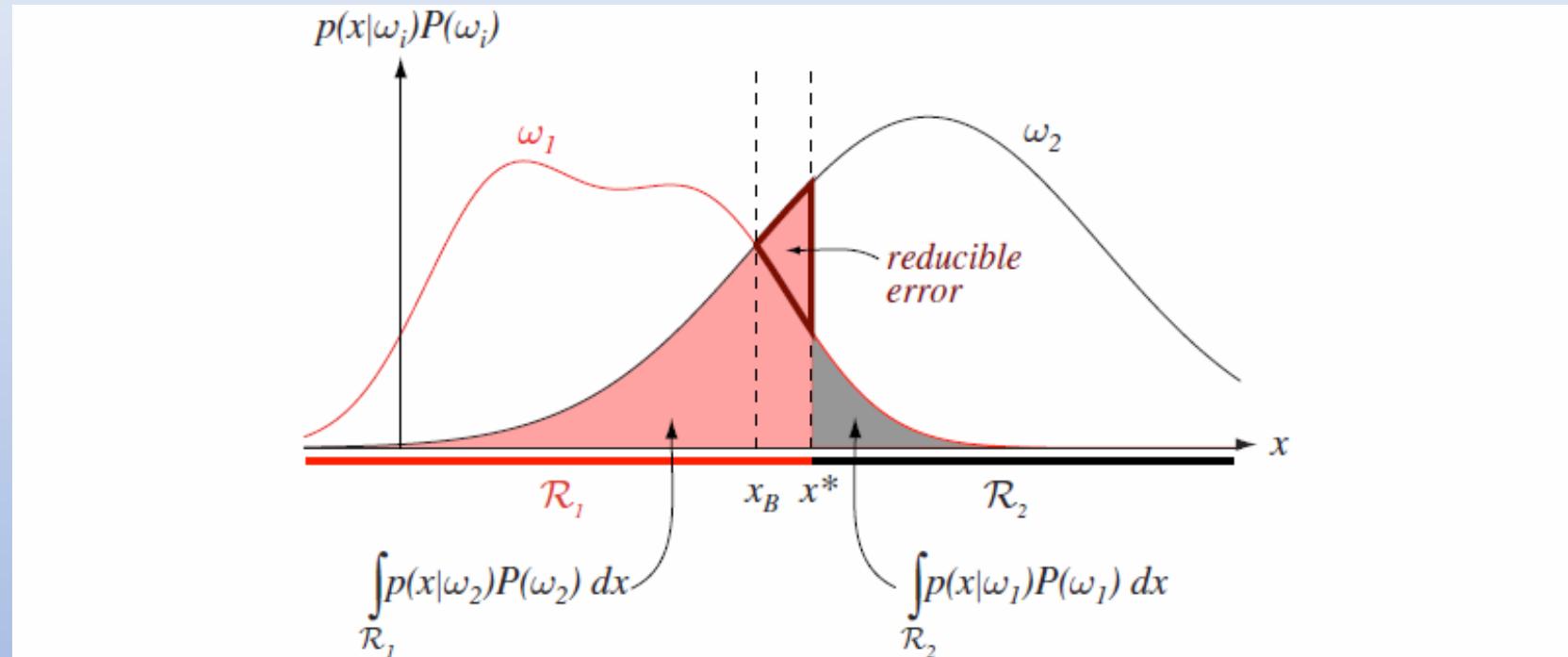
$$p(e) = 1 - \sum_{i=1}^c \int_{\mathcal{R}_i} p(\omega_i)p(x|\omega_i)dx = 1 - \sum_{i=1}^c p(\omega_i) \int_{\mathcal{R}_i} p(x|\omega_i)dx$$

- Obviously, the probability of the correct decision is

$$p(correct) = 1 - p(e) = \sum_{i=1}^c p(\omega_i) \int_{\mathcal{R}_i} p(x|\omega_i)dx$$

- For the 2-class problem: 
$$p(e) = 1 - p(\omega_1) \int_{\mathcal{R}_1} p(x|\omega_1)dx - p(\omega_2) \int_{\mathcal{R}_2} p(x|\omega_2)dx$$
  

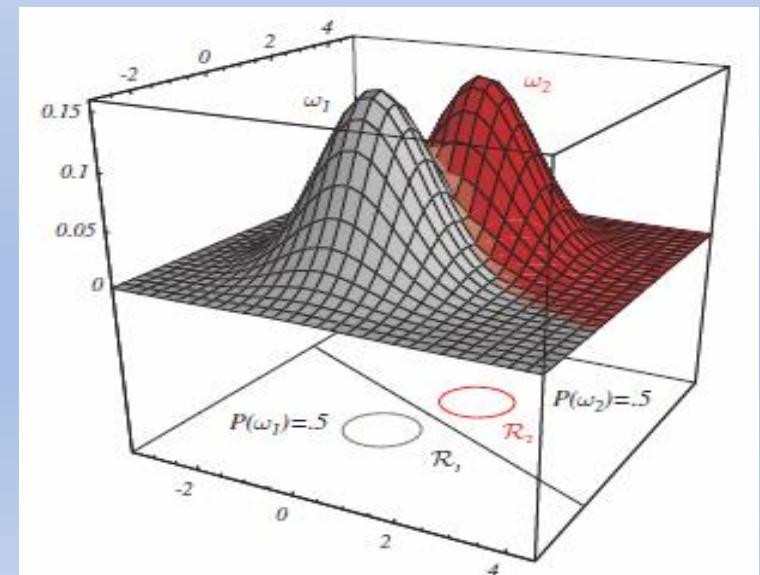
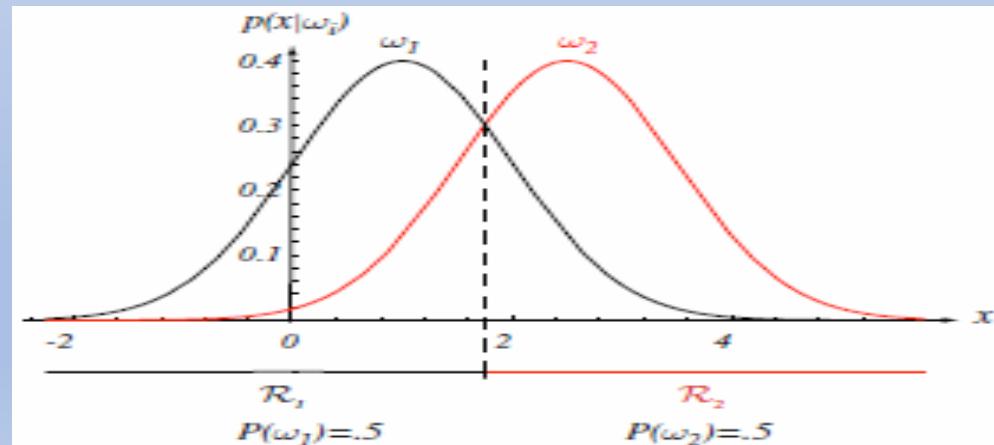
$$= p(\omega_1) \int_{\mathcal{R}_2} p(x|\omega_1)dx + p(\omega_2) \int_{\mathcal{R}_1} p(x|\omega_2)dx$$



**FIGURE 2.17.** Components of the probability of error for equal priors and (nonoptimal) decision point  $x^*$ . The pink area corresponds to the probability of errors for deciding  $\omega_1$  when the state of nature is in fact  $\omega_2$ ; the gray area represents the converse, as given in Eq. 70. If the decision boundary is instead at the point of equal posterior probabilities,  $x_B$ , then this reducible error is eliminated and the total shaded area is the minimum possible; this is the Bayes decision and gives the Bayes error rate. From: Richard O.

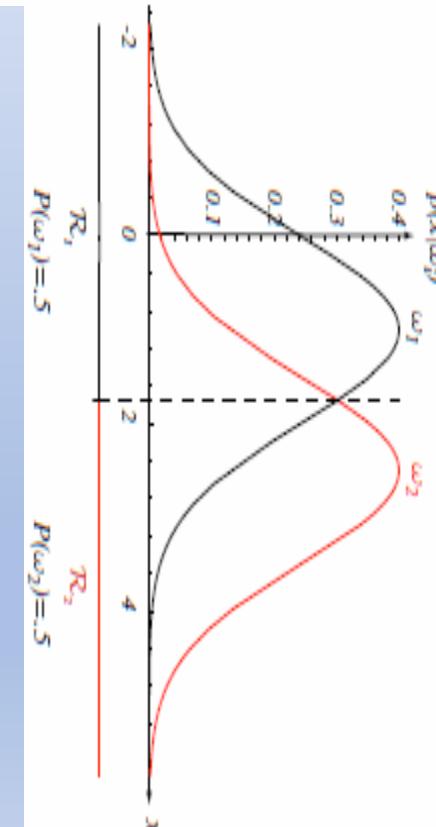
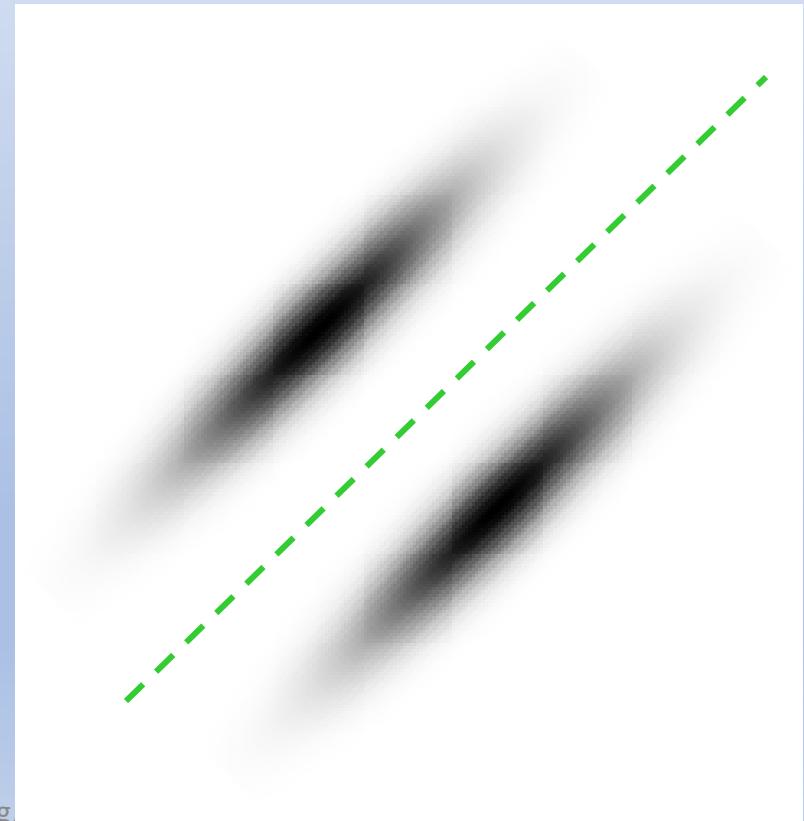
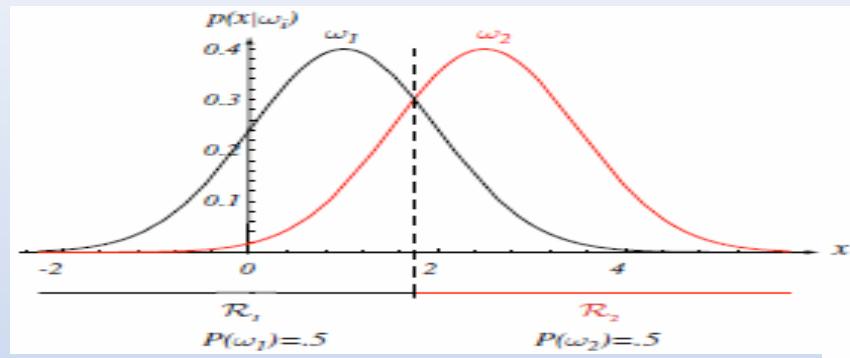
# MAP Decision and Classifiers

- Now if we not only know the person's height  $x_1$ , but also the length of the hair  $x_2$ , it is not difficult to image that we will make better decision, i.e. have lower error probability. The recognition principle will be the same as before as long as now you use a vector  $\mathbf{x} = [x_1, x_2]^T$  to replace the scalar  $x$  before. The probability distribution is now two dimensional. A scalar threshold to separate the two classes in 1D case becomes a curve in the plane spanned by  $\mathbf{x}$ .



# MAP Decision and Classifiers

- The increase of the information or data or feature dimension in general increases the probability of the correct decision or reduce the probability of the decision error.



# MAP Decision and Classifiers

- We understand that the optimal classification maximizes a posterior probability and hence minimizes the classification error by:

$$\text{Decide: } \omega_k = \arg \max_{\omega_i} [p(\omega_i | \mathbf{x})] = \arg \min_{\omega_i} [p(e_i | \mathbf{x})]$$

- Since  $p(\omega_i | \mathbf{x}) = p(\omega_i)p(\mathbf{x}|\omega_i)p^{-1}(\mathbf{x})$

and  $p(\mathbf{x})$  is not a function of  $\omega_i$ , the Bayesian (optimal) classification is to evaluate the called **discriminant functions** that can be defined as

$g_i(\mathbf{x}) = \ln p(\mathbf{x}|\omega_i) + \ln p(\omega_i)$  and find the class  $\omega_i$  that has **the maximum value of the discriminant function** for a given pattern  $\mathbf{x}$ .

- Here, a natural logarithm  $\ln$  is applied as it is a monotonically increasing function that does not affect the decision result but will simplify its evaluation if  $p(\mathbf{x} | \omega_i)$  is an exponential function.

# MAP Decision and Classifiers

- We will derive classifiers for the case that the class conditional PDF is multivariate Gaussian of  $p(\mathbf{x}|\omega_i) = \mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)$

$$p(\mathbf{x}|\omega_i) = \frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}_i|^{1/2}} \exp \left[ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i) \right]$$

- The discriminant function becomes a quadratic function of  $\mathbf{x}$ :

$$\begin{aligned} g_i(\mathbf{x}) &= -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i) + \ln p(\omega_i) - \frac{1}{2} \ln |\boldsymbol{\Sigma}_i| - \frac{d}{2} \ln 2\pi \\ &= -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i) + b_i \\ &= -\frac{1}{2} d_{\Sigma}(\mathbf{x}, \boldsymbol{\mu}_i) + b_i \end{aligned}$$

- where  $d_{\Sigma_i}(\mathbf{x}, \boldsymbol{\mu}_i) = (\mathbf{x} - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i)$ ,  $b_i = \ln p(\omega_i) - \frac{1}{2} \ln |\boldsymbol{\Sigma}_i|$

# MAP Decision and Classifiers

$$d_{\Sigma_i}(\mathbf{x}, \boldsymbol{\mu}_i) = (\mathbf{x} - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1} (\mathbf{x} - \boldsymbol{\mu}_i)$$

is called the square of **Mahalanobis distance** between  $\mathbf{x}$  and  $\boldsymbol{\mu}_i$ .

- Compare to the square of Euclidean distance

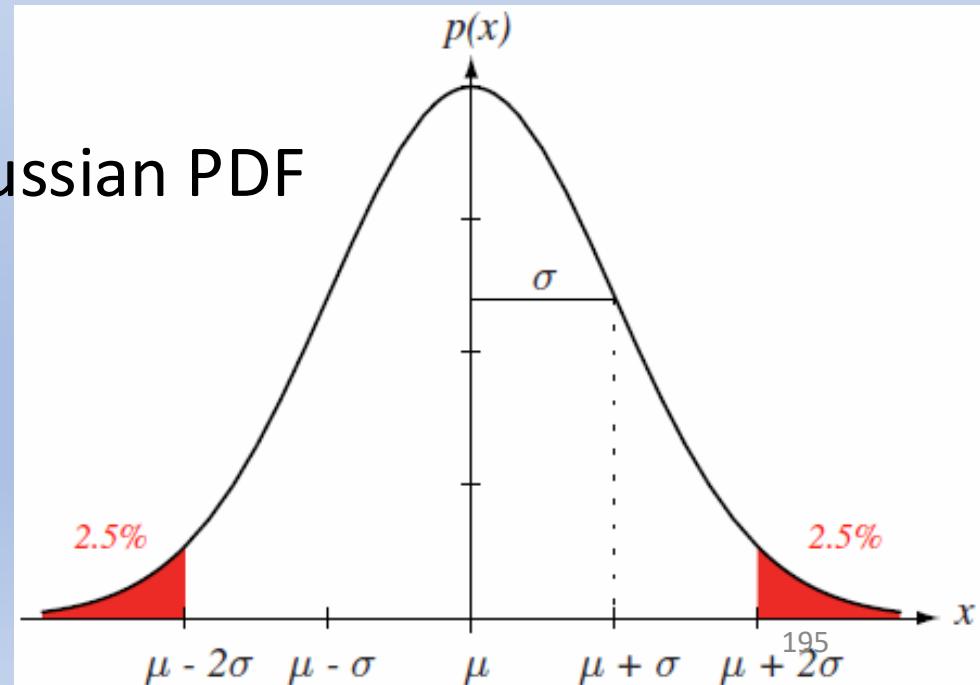
$$d_{Eu}(\mathbf{x}, \boldsymbol{\mu}_i) = (\mathbf{x} - \boldsymbol{\mu}_i)^T (\mathbf{x} - \boldsymbol{\mu}_i)$$

- In 1D case:

$$d_{Eu}(x, \boldsymbol{\mu}_i) = (x - \boldsymbol{\mu}_i)^2$$

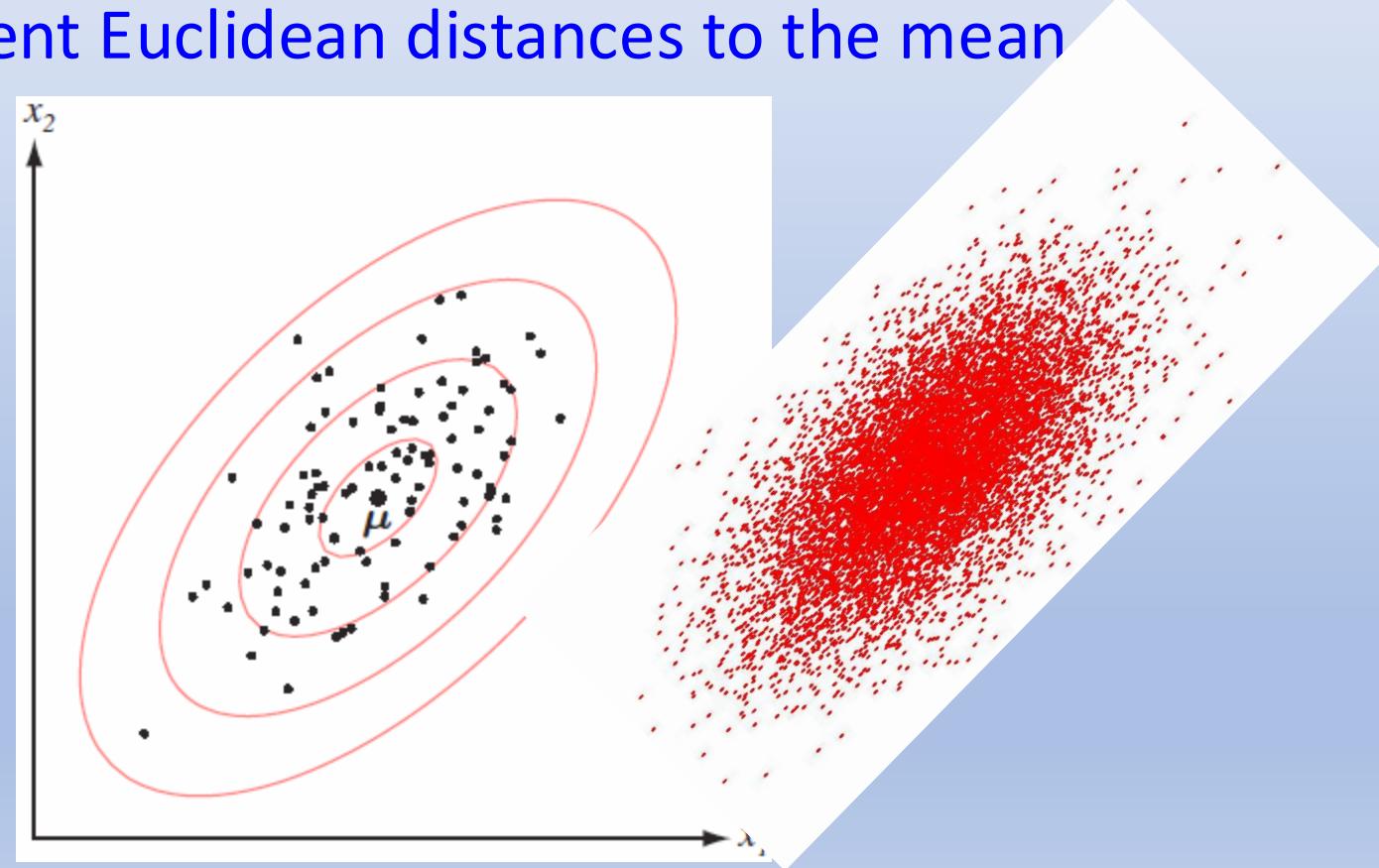
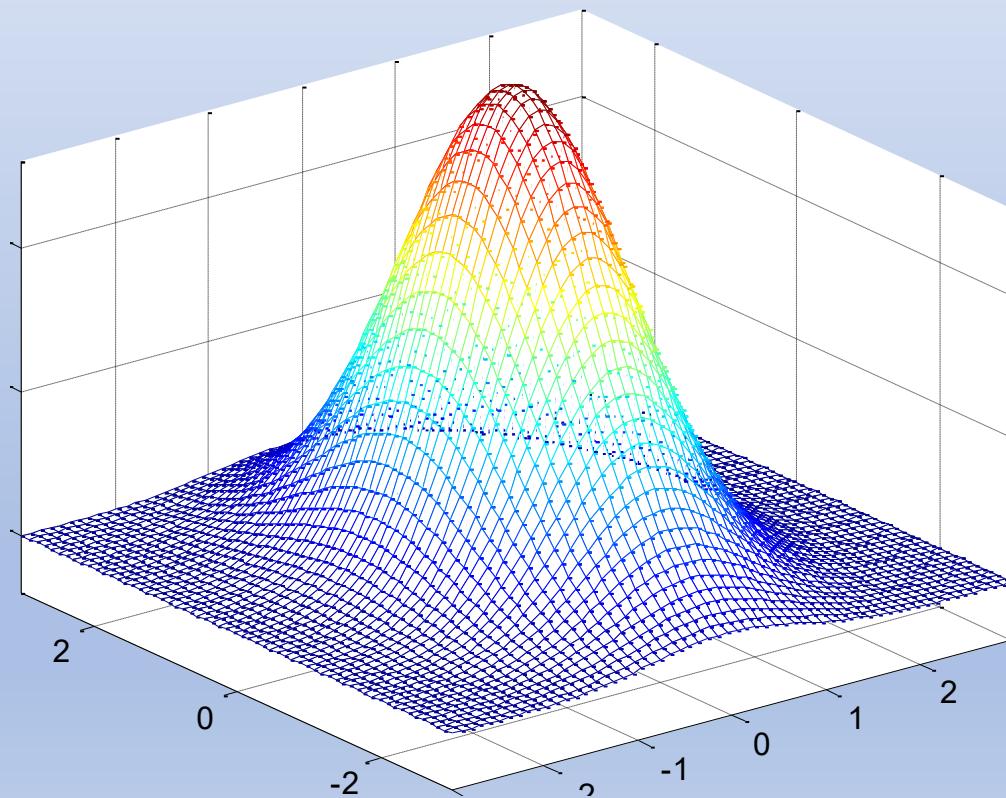
$$d_{\Sigma}(x, \boldsymbol{\mu}_i) = \frac{(x - \boldsymbol{\mu}_i)^2}{\sigma^2}$$

1D Gaussian PDF



# MAP Decision and Classifiers

- For 2D Gaussian PDF Samples drawn from a two-dimensional Gaussian lie in a cloud centered on the mean. The ellipses show lines of equal probability density of the Gaussian. All points on a ellipse has the **same Mahalanobis distance** but different Euclidean distances to the mean



# MAP Decision and Classifiers

- Quadratic Classifier: recall that the discriminant functions for class conditional PDF of multivariate Gaussian of  $p(\mathbf{x}|\omega_i) = N(\boldsymbol{\mu}_i, \Sigma_i)$  is

$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}_i^{-1}(\mathbf{x} - \boldsymbol{\mu}_i) + \ln p(\omega_i) - \frac{1}{2} \ln |\boldsymbol{\Sigma}_i|$$

- This is the general multivariate normal case where the covariance matrices are different for each category. The only term that can be dropped is the  $(d/2) \ln 2\pi$  term, and the resulting discriminant functions are inherently quadratic:

$$g_i(\mathbf{x}) = \mathbf{x}^t \mathbf{W}_i \mathbf{x} + \mathbf{w}_i^t \mathbf{x} + \omega_{i0}$$

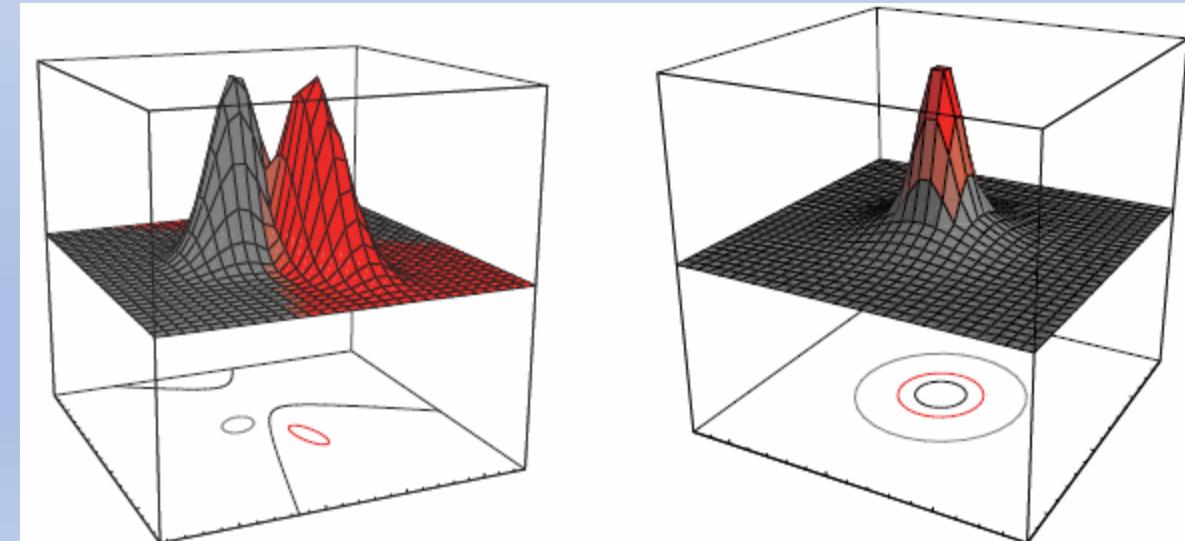
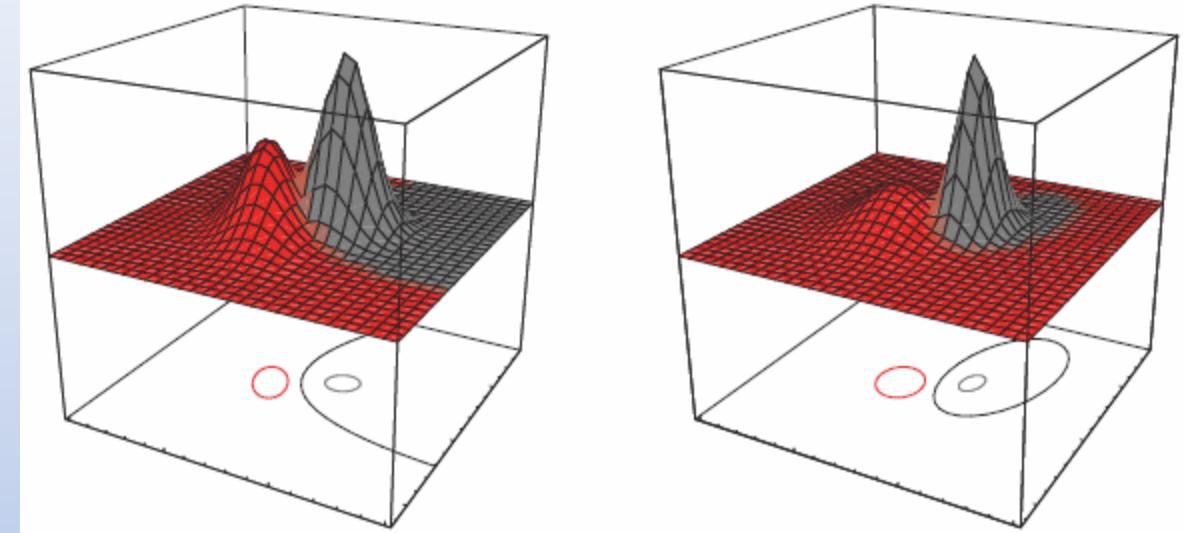
where  $\mathbf{W}_i = -\frac{1}{2}\boldsymbol{\Sigma}_i^{-1}$  and  $\mathbf{w}_i = \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i$  and

$$\omega_{i0} = -\frac{1}{2}\boldsymbol{\mu}_i^t \boldsymbol{\Sigma}_i^{-1} \boldsymbol{\mu}_i - \frac{1}{2} \ln |\boldsymbol{\Sigma}_i| + \ln P(\omega_i)$$

- The decision surfaces are *hyperquadrics*, and can assume any of the general forms hyperplanes, pairs of hyperplanes, hyperspheres, hyperellipsoids, hyperparaboloids, and hyperhyperboloids of various types.

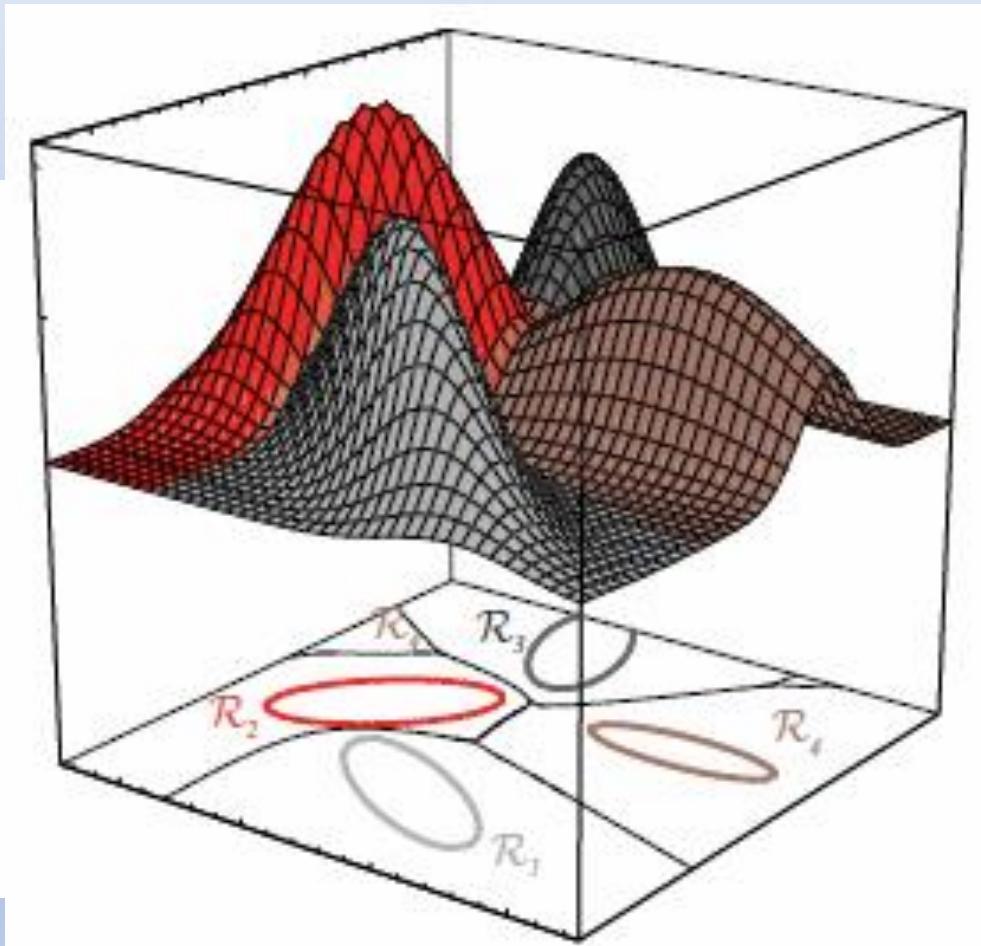
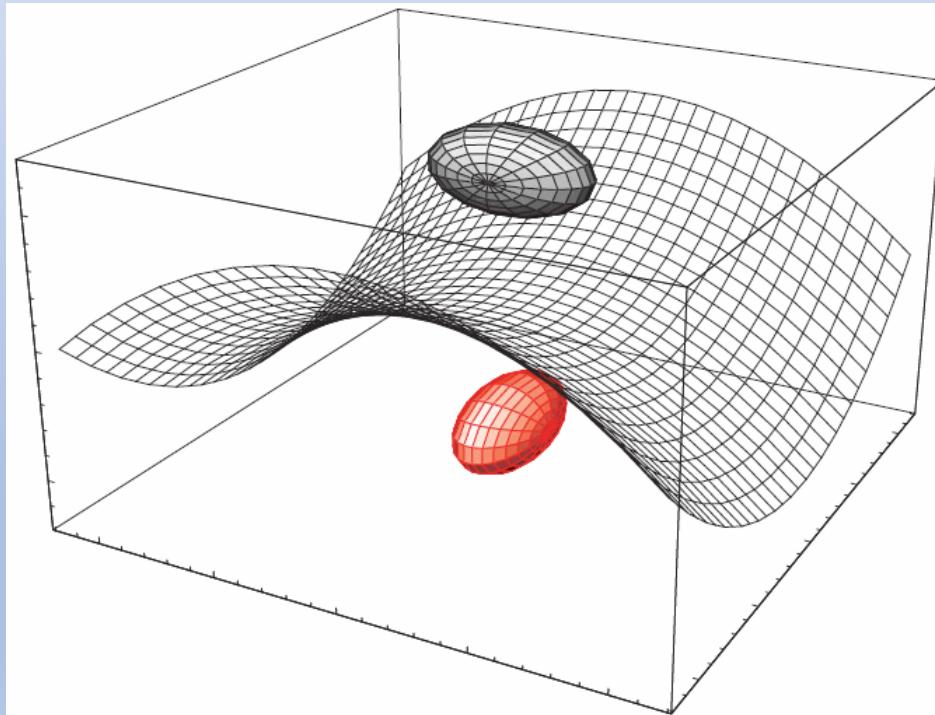
# MAP Decision and Classifiers

- Decision boundary  
examples of quadratic classifier in  
2-dimensional, 2-class cases.



# MAP Decision and Classifiers

- Complex decision boundary for 3-dimensional, 2-class case and 2-dimensional, multiclass problem



# MAP Decision and Classifiers

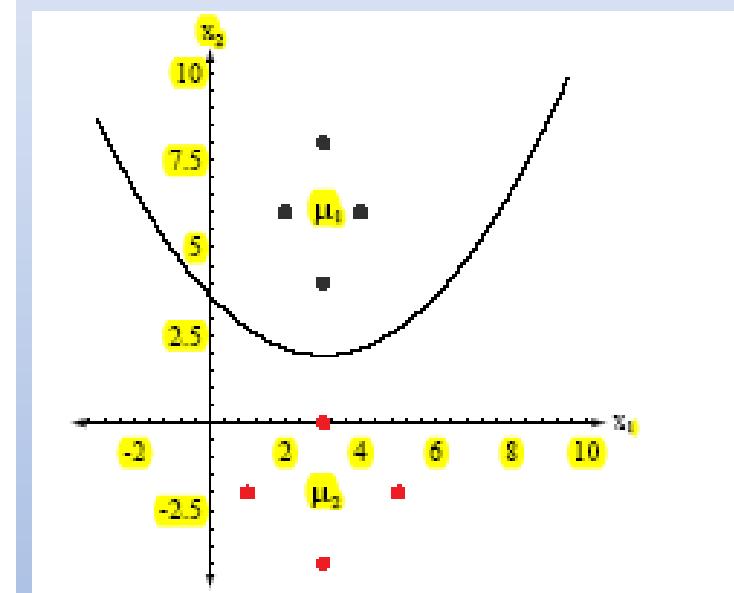
Example: Decision regions for two-dimensional Gaussian data

- Let  $\omega_1$  be the set of the four black points, and  $\omega_2$  the red points.

$$\mu_1 = \begin{bmatrix} 3 \\ 6 \end{bmatrix}; \Sigma_1 = \begin{pmatrix} 1/2 & 0 \\ 0 & 2 \end{pmatrix} \text{ and } \mu_2 = \begin{bmatrix} 3 \\ -2 \end{bmatrix}; \Sigma_2 = \begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$$

- The inverse matrices are then,

$$\Sigma_1^{-1} = \begin{pmatrix} 2 & 0 \\ 0 & 1/2 \end{pmatrix} \text{ and } \Sigma_2^{-1} = \begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix}$$



The computed Bayes decision boundary for two Gaussian distributions, each based on four data points.

# MAP Decision and Classifiers

- Special case 1: all classes own the same covariance matrix  $\Sigma_i = \Sigma$ .
- The discriminant functions are simplified as

$$\begin{aligned}g_i(\mathbf{x}) &= -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_i) + \ln p(\omega_i) \\&= -\frac{1}{2}\mathbf{x}^T \boldsymbol{\Sigma}^{-1}\mathbf{x} + \boldsymbol{\mu}_i^T \boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_i^T \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_i + \ln p(\omega_i)\end{aligned}$$

- and drop terms that are independent to the class label  $i$  we have:

$$\begin{aligned}g_i(\mathbf{x}) &= \boldsymbol{\mu}_i^T \boldsymbol{\Sigma}^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_i^T \boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_i + \ln p(\omega_i) \\&= \mathbf{w}_i^T \mathbf{x} + w_{i0}\end{aligned}$$

- The discriminant function is a linear function of  $\mathbf{x}$ . This results in a linear classifier as we will show that the decision or classification boundary between any two classes is a hyperplane. We call  $w_{i0}$  the threshold or bias for the class  $\omega_i$ .

# MAP Decision and Classifiers

- The decision or classification boundary between any two classes is the solution of the equation.

$$g_i(\mathbf{x}) = g_j(\mathbf{x}) \Rightarrow (\mathbf{w}_i - \mathbf{w}_j)^T \mathbf{x} + w_{i0} - w_{j0} = 0$$

$$(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}^{-1} \mathbf{x} - \frac{1}{2} (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{\mu}_i + \boldsymbol{\mu}_j) + \ln[p(\omega_i)/p(\omega_j)] = 0$$

- It is a straight line in 2D space  $\mathbf{x}$ , a plane in 3D space  $\mathbf{x}$  and a hyperplane in higher dimensional space  $\mathbf{x}$ .
- We see that the decision hyper plane that separates two classes is generally not orthogonal to the line between the means. (in what case is it orthogonal?)

# MAP Decision and Classifiers

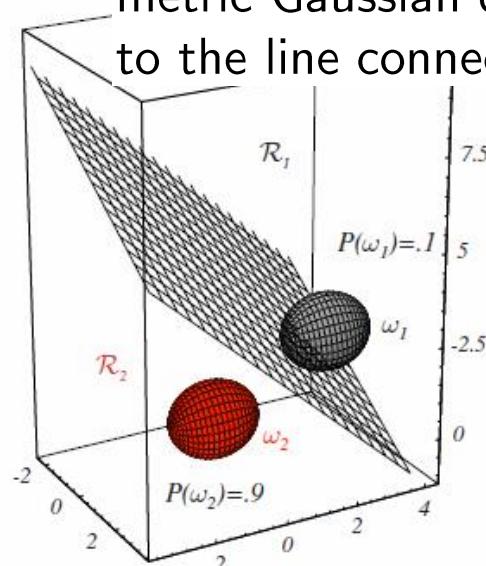
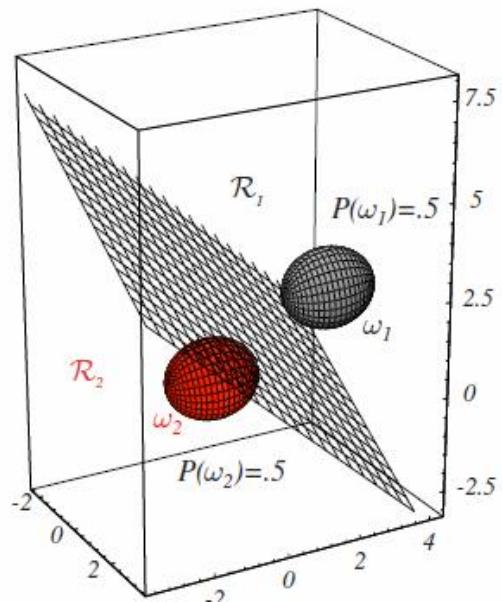
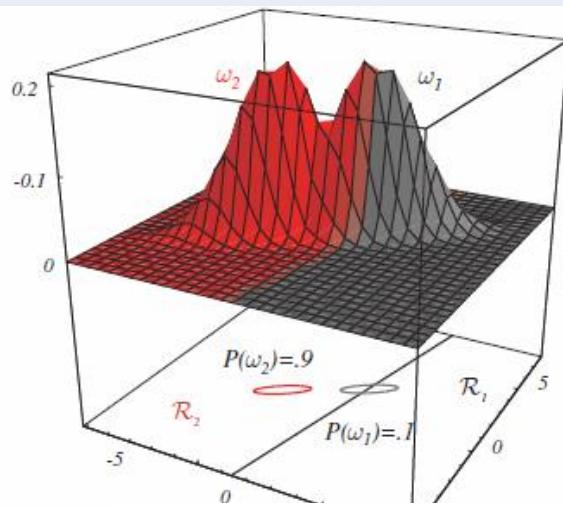
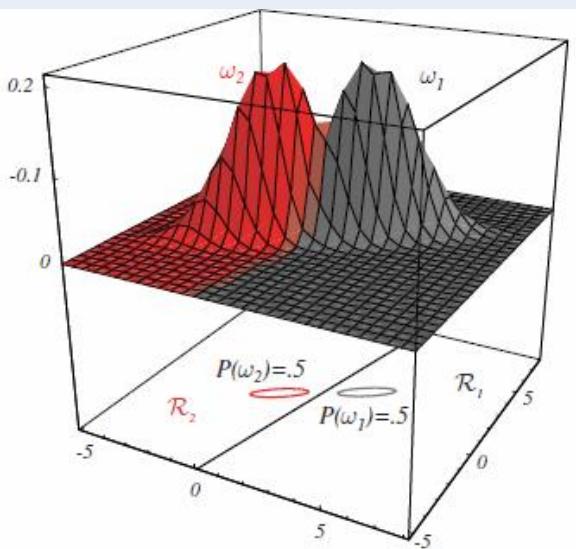


Figure 2.12: Probability densities (indicated by the surfaces in two dimensions and ellipsoidal surfaces in three dimensions) and decision regions for equal but asymmetric Gaussian distributions. The decision hyperplanes need not be perpendicular to the line connecting the means.

# MAP Decision and Classifiers

Example:

Given pattern vectors  $(1, 2)^t, (2, 2)^t, (3, 1)^t, (3, 2)^t$ , and  $(2, 3)^t$ , are known to be in class  $\omega_1$ . Another set of vectors,  $(7, 9)^t, (8, 9)^t, (9, 8)^t, (9, 9)^t$ , and  $(8, 10)^t$ , are known to be in class  $\omega_2$ . Find the Bayes classifier and the decision boundary. (assume all classes are equally likely to occur.)

Solution:

$$c = 2, j = 1, 2.$$

$$\mu_1 = \frac{1}{5} \left[ \begin{pmatrix} 1 \\ 2 \end{pmatrix} + \begin{pmatrix} 2 \\ 2 \end{pmatrix} + \begin{pmatrix} 3 \\ 1 \end{pmatrix} + \begin{pmatrix} 3 \\ 2 \end{pmatrix} + \begin{pmatrix} 2 \\ 3 \end{pmatrix} \right] = \frac{1}{5} \begin{bmatrix} 11 \\ 10 \end{bmatrix}$$

$$\mu_2 = \frac{1}{5} \left[ \begin{pmatrix} 7 \\ 9 \end{pmatrix} + \begin{pmatrix} 8 \\ 9 \end{pmatrix} + \begin{pmatrix} 9 \\ 8 \end{pmatrix} + \begin{pmatrix} 9 \\ 9 \end{pmatrix} + \begin{pmatrix} 8 \\ 10 \end{pmatrix} \right] = \frac{1}{5} \begin{bmatrix} 41 \\ 45 \end{bmatrix}$$

# MAP Decision and Classifiers

Solution: (cont)

$$\Sigma_j = \left( \frac{1}{N_j} \sum_{\mathbf{x} \in \omega_j} \mathbf{x}\mathbf{x}^t \right) - \mu_j \mu_j^t$$

$$\Sigma_1 = \frac{1}{5} \left[ \begin{pmatrix} 1 \\ 2 \end{pmatrix} (1 \ 2) + \begin{pmatrix} 2 \\ 2 \end{pmatrix} (2 \ 2) + \begin{pmatrix} 3 \\ 1 \end{pmatrix} (3 \ 1) + \begin{pmatrix} 3 \\ 2 \end{pmatrix} (3 \ 2) + \begin{pmatrix} 2 \\ 3 \end{pmatrix} (2 \ 3) \right]$$

$$-\frac{1}{25} \begin{pmatrix} 11 \\ 10 \end{pmatrix} (11 \ 10)$$

$$\Sigma_1 = \frac{1}{25} \begin{pmatrix} 14 & -5 \\ -5 & 10 \end{pmatrix}$$

Similarly, we find  $\Sigma_2 = \Sigma_1 = \Sigma$

$$\Sigma^{-1} = \frac{5}{23} \begin{pmatrix} 10 & 5 \\ 5 & 14 \end{pmatrix}$$

# MAP Decision and Classifiers

**Solution: (cont)**

All classes are equally likely to occur means  $P(\omega_1) = P(\omega_2)$ . Also,  $\Sigma_1 = \Sigma_2 = \Sigma$ , the decision function is simplified to

$$d_j(\mathbf{x}) = \mathbf{x}^t \Sigma^{-1} \boldsymbol{\mu}_j - \frac{1}{2} \boldsymbol{\mu}_j^t \Sigma^{-1} \boldsymbol{\mu}_j$$

$$\Sigma^{-1} \boldsymbol{\mu}_1 = \frac{1}{23} \begin{pmatrix} 160 \\ 195 \end{pmatrix} \text{ and } \boldsymbol{\mu}_1^t \Sigma^{-1} \boldsymbol{\mu}_1 = \frac{742}{23} \quad g_i(\mathbf{x}) = -(\mathbf{x} - \mathbf{\mu}_i)^T \mathbf{\Sigma}^{-1} (\mathbf{x} - \mathbf{\mu}_i)$$

$$d_1(\mathbf{x}) = \frac{160}{23}x_1 + \frac{195}{23}x_2 - 16.13$$

$$\text{Similarly, } \Sigma^{-1} \boldsymbol{\mu}_2 = \frac{1}{23} \begin{pmatrix} 635 \\ 835 \end{pmatrix} \text{ and } \boldsymbol{\mu}_2^t \Sigma^{-1} \boldsymbol{\mu}_2 = 553.12$$

$$d_2(\mathbf{x}) = \frac{635}{23}x_1 + \frac{835}{23}x_2 - 276.56$$

# MAP Decision and Classifiers

**Solution:** (cont) The decision matrix is

$$\begin{pmatrix} d_1 \\ d_2 \end{pmatrix} = \begin{pmatrix} \frac{160}{23} & \frac{195}{23} & -16.13 \\ \frac{635}{23} & \frac{835}{23} & -276.56 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix}$$

- To find the decision boundary,  $d_{12}(\mathbf{x}) = d_1(\mathbf{x}) - d_2(\mathbf{x})$
- The decision rule is : decide  $\omega_1$  if  $d_{12} > 0$ , otherwise, decide  $\omega_2$ .

# MAP Decision and Classifiers

- Special case 2: all classes own the same diagonal, scalar covariance matrix  $\Sigma_i = \sigma^2 \mathbf{I}$ .  $\mathbf{I}$  is the identity matrix.
- This case occurs when all features (components of  $\mathbf{x}$ ) are statistically uncorrelated and each feature has the same variance,  $\sigma^2$ .
- Note that  $\Sigma_i = \sigma^2 \mathbf{I} \Rightarrow \Sigma_i^{-1} = \mathbf{I}/\sigma^2$ ,  $|\Sigma_i| = \sigma^{2d}$
- The discriminant functions are simplified as

$$\begin{aligned}g_i(\mathbf{x}) &= -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_i) + \ln p(\omega_i) \\&= -\frac{1}{2\sigma^2}(\mathbf{x} - \boldsymbol{\mu}_i)^T(\mathbf{x} - \boldsymbol{\mu}_i) + \ln p(\omega_i) \\&= -\frac{1}{2\sigma^2}(\mathbf{x}^T \mathbf{x} - 2\boldsymbol{\mu}_i^T \mathbf{x} + \boldsymbol{\mu}_i^T \boldsymbol{\mu}_i) + \ln p(\omega_i)\end{aligned}$$

# MAP Decision and Classifiers

- The quadratic term  $\mathbf{x}^T \mathbf{x}$  is the same for all class  $i$  and hence can be dropped.

Multiplying the constant  $\sigma^2$ , the discriminant functions are simplified as

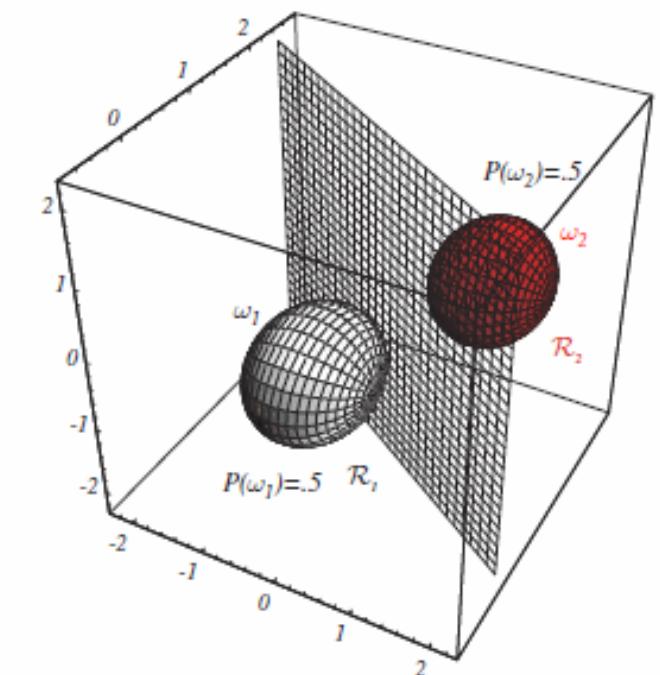
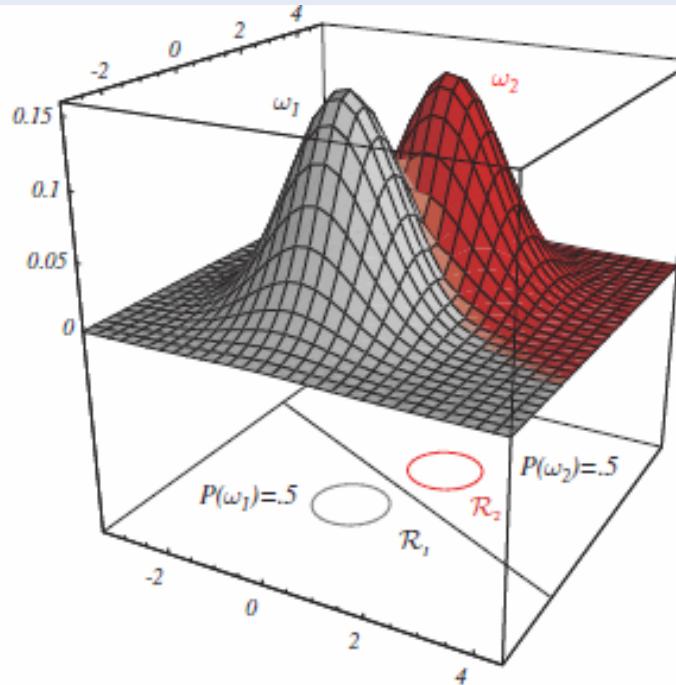
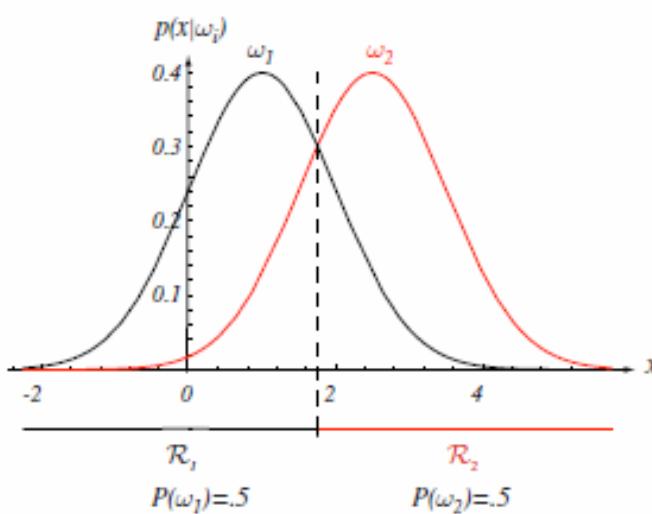
$$\begin{aligned}g_i(\mathbf{x}) &= \boldsymbol{\mu}_i^T \mathbf{x} - \boldsymbol{\mu}_i^T \boldsymbol{\mu}_i / 2 + \sigma^2 \ln p(\omega_i) \\&= \mathbf{w}_i^T \mathbf{x} + w_{i0}\end{aligned}$$

- We call  $w_{i0}$  the threshold or bias for the class  $\omega_i$ . It is a linear function of  $\mathbf{x}$ . This results a linear classifier.
- The decision or classification boundary between any two classes is the solution of the equation.

$$\begin{aligned}g_i(\mathbf{x}) = g_j(\mathbf{x}) &\Rightarrow (\mathbf{w}_i - \mathbf{w}_j)^T \mathbf{x} + w_{i0} - w_{j0} = 0 \\(\boldsymbol{\mu}_i - \boldsymbol{\mu}_j)^T \mathbf{x} - (\boldsymbol{\mu}_i^T \boldsymbol{\mu}_i - \boldsymbol{\mu}_j^T \boldsymbol{\mu}_j) / 2 + \sigma^2 \ln [p(\omega_i)/p(\omega_j)] &= 0\end{aligned}$$

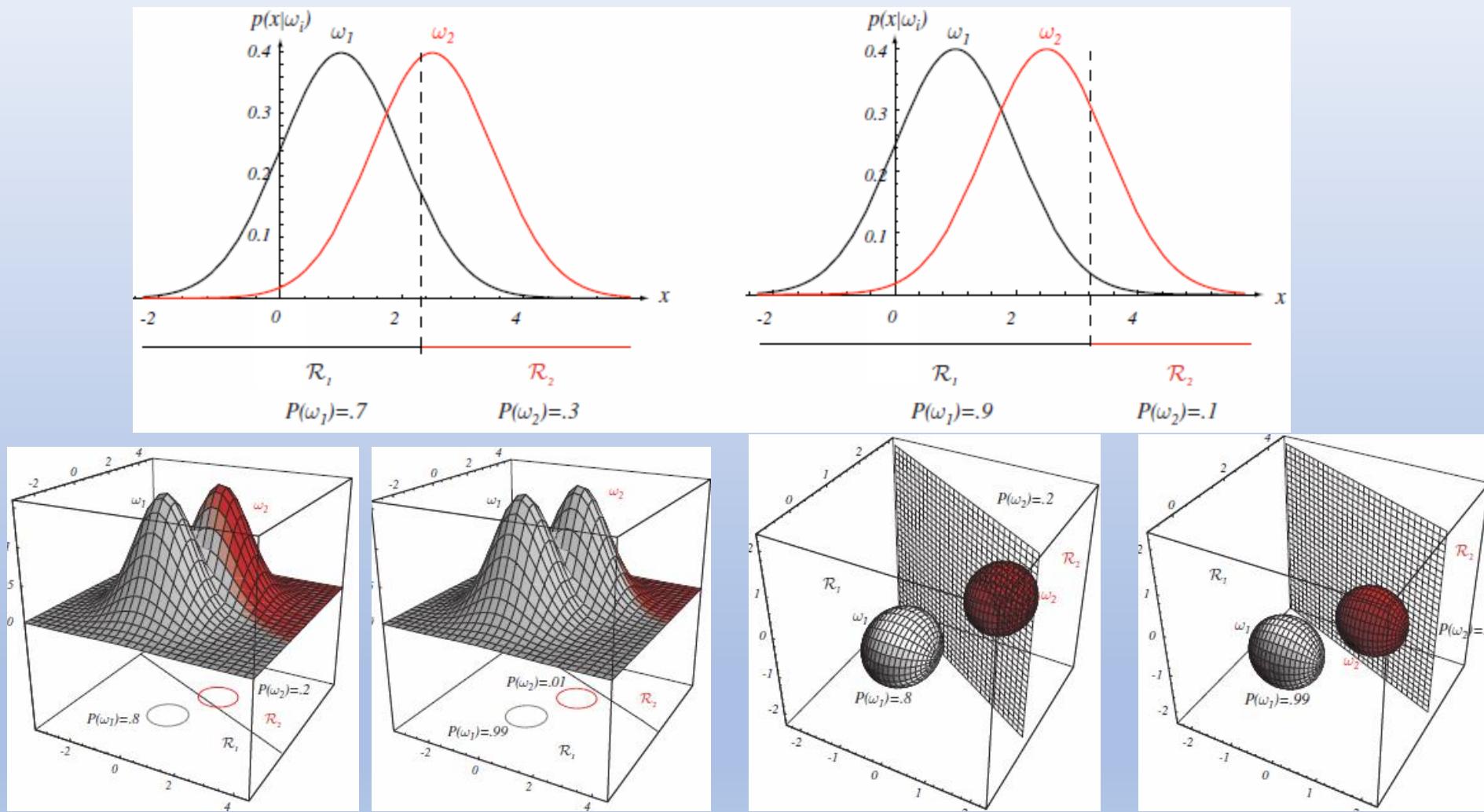
- This equation defines a hyperplane orthogonal to the line linking the two means.

# MAP Decision and Classifiers



**FIGURE 2.10.** If the covariance matrices for two distributions are equal and proportional to the identity matrix, then the distributions are spherical in  $d$  dimensions, and the boundary is a generalized hyperplane of  $d - 1$  dimensions, perpendicular to the line separating the means. In these one-, two-, and three-dimensional examples, we indicate  $p(x|\omega_i)$  and the boundaries for the case  $P(\omega_1) = P(\omega_2)$ . In the three-dimensional case, the grid plane separates  $\mathcal{R}_1$  from  $\mathcal{R}_2$ . From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern*

# MAP Decision and Classifiers



**FIGURE 2.11.** As the priors are changed, the decision boundary shifts; for sufficiently disparate priors the boundary will not lie between the means of these one-, two- and three-dimensional spherical Gaussian distributions. From: Richard O. Duda, Peter E.

# MAP Decision and Classifiers

- Recall the discriminant function

$$g_i(\mathbf{x}) = -\frac{1}{2\sigma^2} (\mathbf{x} - \boldsymbol{\mu}_i)^T (\mathbf{x} - \boldsymbol{\mu}_i) + \ln p(\omega_i)$$

- If the prior probabilities are the same for all classes, then

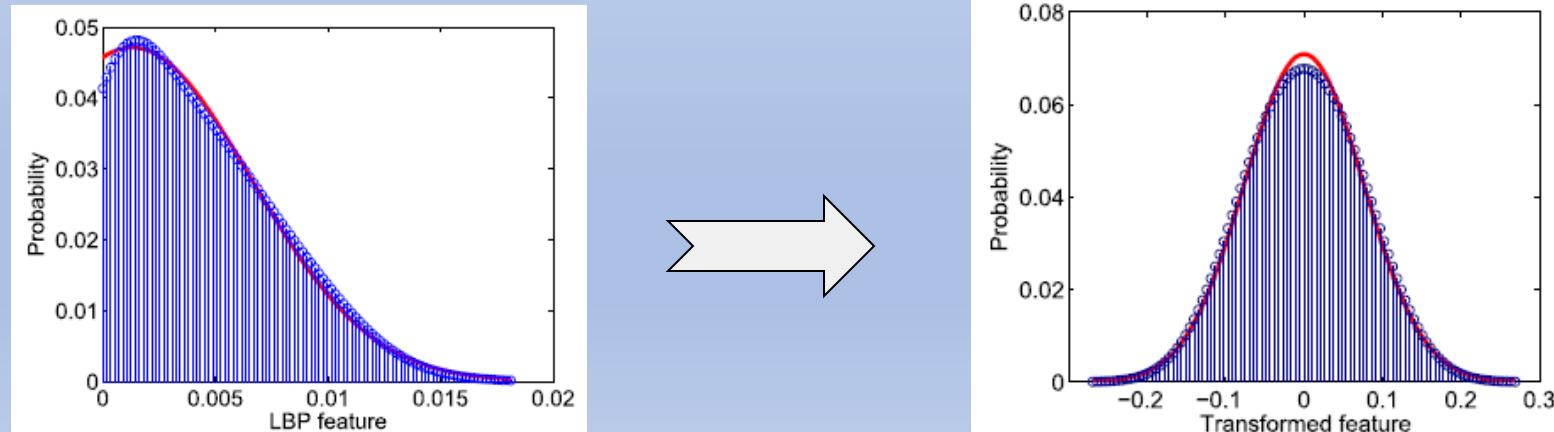
$$g_i(\mathbf{x}) = -(\mathbf{x} - \boldsymbol{\mu}_i)^T (\mathbf{x} - \boldsymbol{\mu}_i) = -d_{Eu}(\mathbf{x}, \boldsymbol{\mu}_i) = \|\mathbf{x} - \boldsymbol{\mu}_i\|$$

- When this happens, the optimum decision rule can be stated very simply: to classify a feature vector  $\mathbf{x}$ , measure the Euclidean distance from  $\mathbf{x}$  to each of the  $c$  mean vectors, and assign  $\mathbf{x}$  to the category of the nearest mean. Such a classifier is called a **minimum distance classifier**. If each mean vector is thought of as being an ideal prototype or template for patterns in its class, then this is essentially a **template matching** procedure.

# MAP Decision and Classifiers

- We have derived several classifiers under Gaussian assumption.
- Gaussian PDF is the most natural and most common distribution.
- For Non-Gaussian distribution, it is very difficult if not impossible to derive a theoretical optimal classifier.
- One way to solve non-Gaussian data distribution is to perform some proper feature transform to convert it into Gaussian PDF. Example:

J. Ren, X.D. Jiang and J. Yuan, “[A Chi-Squared-Transformed Subspace of LBP Histogram for Visual Recognition](#),” *IEEE Transactions on Image Processing*, vol. 24, no. 6, pp. 1893-1904, June, 2015.



# **Topic 7**

## **Statistical Estimation and Machine Learning**

# Statistical Estimation & Machine Learning

## Outline

- Nonparametric approach to estimate the probability distribution density
  - Parzen-window approach
  - $k$ -nearest-neighbor  $k$ NN rule
- Parametric approach to estimate the probability distribution density
  - Maximum likelihood (ML) estimation
  - Multivariate Gaussian probability distribution density

# Statistical Estimation & Machine Learning

- We understand that the best decision or optimal classification is:

$$\begin{aligned}\text{Decide } \omega_k &= \arg \min_{\omega_i} [p(e_i | \mathbf{x})] = \arg \min_{\omega_i} [1 - p(\omega_i | \mathbf{x})] \\ &= \arg \max_{\omega_i} [p(\omega_i | \mathbf{x})] = \arg \max_{\omega_i} [p(\omega_i) p(\mathbf{x} | \omega_i)]\end{aligned}$$

- To design an automatic pattern recognition system, we need know the probability distribution/density function (PDF) for all values of  $\mathbf{x}$  so that the system can make decision for any value of received data  $\mathbf{x}$ . In practice, the PDF is often unknown and can only be estimated by collected examples/samples  $\{\mathbf{x}_i\} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n]$  for the design of the pattern recognition system.
- Determining some rules or some deterministic values on a random variable  $\mathbf{x}$  based on a set of training samples  $D = \{\mathbf{x}_i\}$  is the task of statistical estimation or machine learning.

# Statistical Estimation & Machine Learning

- The probability distribution/density function (PDF) is the complete information about a random variable.
- It is difficult to estimate the posterior probability  $p(\omega_k|\mathbf{x})$  function directly. So we learn the prior probability  $p(\omega_k)$  and the class-conditional probability function  $p(\mathbf{x}|\omega_k)$ .
- The  $c$  prior probabilities can be easily estimated by

$$\hat{p}(\omega_k) = n_k/n$$

where  $n_k$  and  $n$  are the number of training samples of class  $\omega_k$  and the total number of training samples.

As the estimation method is the same for all classes, we simplifying the notation of  $p(\mathbf{x}|\omega_k)$  to  $p(\mathbf{x})$  and suppose we have  $n$  samples  $\{\mathbf{x}_i\}=[\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n]$  drawn independently and identically distributed (i.i.d.) according to the probability law  $p(\mathbf{x})$ .

# Statistical Estimation & Machine Learning

- According to the definition of probability density function, probability  $P$  that  $\mathbf{x}$  falls in a region  $R$  is:

$$P(\mathbf{x}) = \int_{R_{\mathbf{x}}} p(\mathbf{t}) d\mathbf{t} \approx p(\mathbf{x})V$$

where  $V$  is the volume of the region  $R_{\mathbf{x}}$ .

- If out of  $n$  training samples of this class,  $k$  samples fall in the region  $R$ , the estimate of  $P$  is then.  $\hat{P}(\mathbf{x}) = \frac{k}{n}$
- Therefore, the estimate of the PDF  $p(\mathbf{x})$  is  $\hat{p}(\mathbf{x}) = \frac{k}{nV}$
- This is called **nonparametric approach** to the estimation of the probability density function because no assumption of the model is applied.

# Statistical Estimation & Machine Learning

- The procedure to estimate the PDF based on  $n$  training samples is: Given a value of  $\mathbf{x}$ , select a region/cell of size (volume)  $V$  centered at  $\mathbf{x}$ , count the number of samples  $k$  fall in the region/cell. The probability density  $p(\mathbf{x})$  at  $\mathbf{x}$  is then estimated as :

$$\hat{p}(\mathbf{x}) = \frac{k}{nV}$$

- Obviously, different shape and size of the region/cell lead to different estimate of PDF.
- In general,  $\mathbf{x}$  is multiple dimensional  $\mathbf{x}=[x_1, x_2, \dots, x_d]$ . If we choose a  $d$ -dimensional hypercube of the side length  $h$  as the region, a sample  $\mathbf{x}_i=[x_{i1}, x_{i2}, \dots, x_{id}]$  will fall into the hypercube if

$$\frac{|x_j - x_{ij}|}{h} < \frac{1}{2} \quad \text{for } \forall j = 1, 2, \dots, d$$

# Statistical Estimation & Machine Learning

- Therefore, we can express the number of samples falling into the cell  $k$

mathematically as

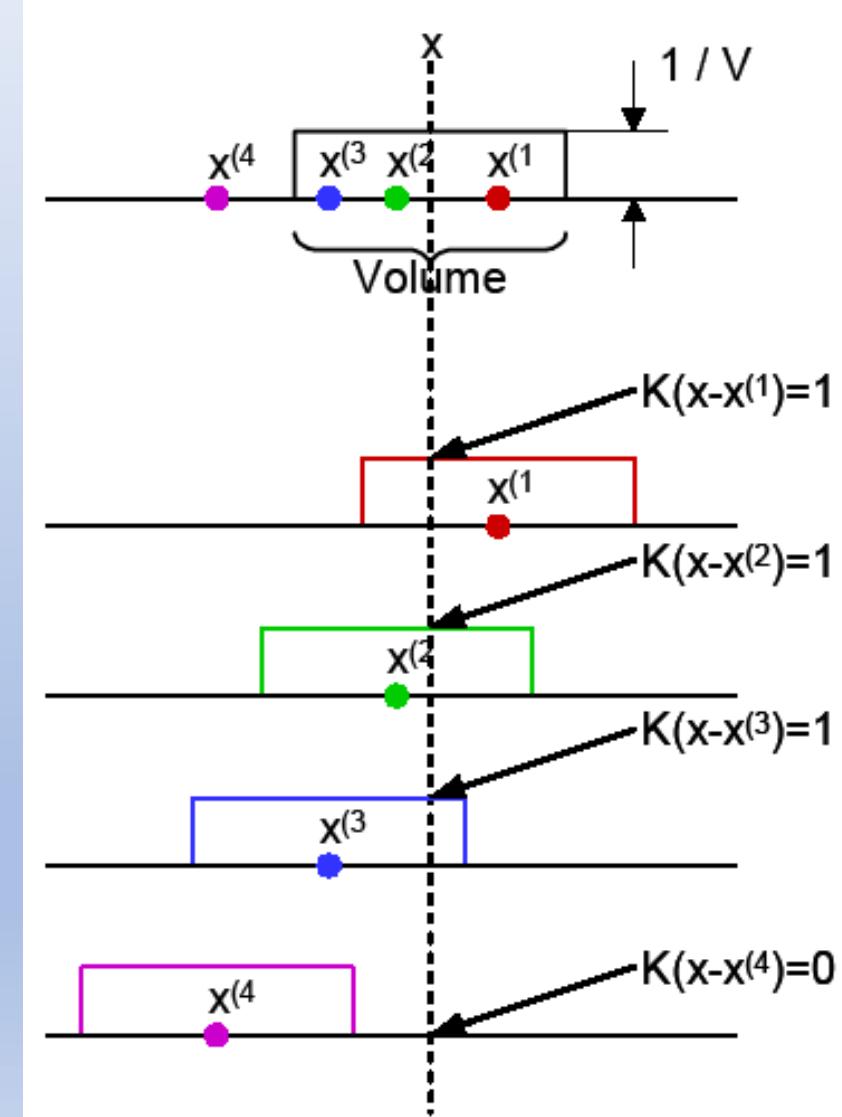
$$k = \sum_{i=1}^n K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right)$$

- Where the kernel function called Parzen-window is defined as

$$K(\mathbf{u}) = \begin{cases} 1 & \text{if } |u_j| < 1/2 \text{ for } \forall j = 1, 2, \dots, d \\ 0 & \text{otherwise} \end{cases}$$

- The probability density  $p(\mathbf{x})$  at  $\mathbf{x}$  is then estimated as :

$$\hat{p}(\mathbf{x}) = \frac{1}{nh^d} \sum_{i=1}^n K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right)$$



# Statistical Estimation & Machine Learning

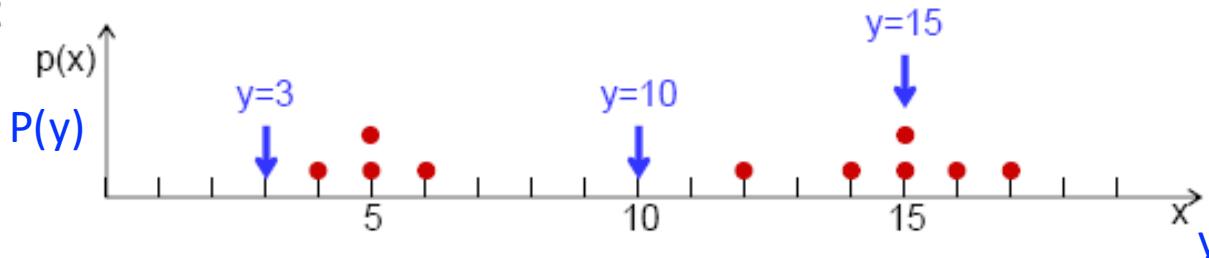
- Given the dataset below, use Parzen windows to estimate the density  $p(x)$  at  $y=3, 10, 15$ . Use a bandwidth of  $h=4$

Example:

- $X = \{x^{(1)}, x^{(2)}, \dots, x^{(N)}\} = \{4, 5, 5, 6, 12, 14, 15, 15, 16, 17\}$

- Solution

- Let's first draw the dataset to get an idea of what numerical results we should expect



- Let's now estimate  $p(y=3)$ :

$$p_{\text{KDE}}(y=3) = \frac{1}{Nh^D} \sum_{n=1}^N K\left(\frac{y-x^{(n)}}{h}\right) = \frac{1}{10 \times 4^1} \left[ K\left(\frac{3-4}{4}\right) + K\left(\frac{3-5}{4}\right) + K\left(\frac{3-5}{4}\right) + K\left(\frac{3-6}{4}\right) + \dots + K\left(\frac{3-17}{4}\right) \right] = \\ = \frac{1}{10 \times 4^1} [1+0+0+0+0+0+0+0+0+0] = \frac{1}{10 \times 4} = 0.025$$

- Similarly

$$p_{\text{KDE}}(y=10) = \frac{1}{10 \times 4^1} [0+0+0+0+0+0+0+0+0+0] = \frac{0}{10 \times 4} = 0 \quad \text{Note here: } K(\mathbf{u}) = \begin{cases} 1 & \text{if } |\mathbf{u}_j| < 1/2 \\ 0 & \text{otherwise} \end{cases}$$
$$p_{\text{KDE}}(y=15) = \frac{1}{10 \times 4^1} [0+0+0+0+0+1+1+1+1+0] = \frac{4}{10 \times 4} = 0.1$$

# Statistical Estimation & Machine Learning

- The rectangular kernel function produces **unsmoothed PDF** due to the unsmooth rectangular kernel function. In fact, we can choose any function as kernel so long as:

$$K(\mathbf{u}) \geq 0$$

$$\int_{-\infty}^{\infty} K(\mathbf{u}) d\mathbf{u} = 1 \text{ i.e. } \int_{-\infty}^{\infty} \frac{1}{h^d} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right) d\mathbf{x} = 1$$

$$\therefore \int_{-\infty}^{\infty} K(\mathbf{u}) d\mathbf{u} = \int_{-\infty}^{\infty} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right) d\frac{\mathbf{x} - \mathbf{x}_i}{h} = \int_{-\infty}^{\infty} \frac{1}{h^d} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right) d\mathbf{x}$$

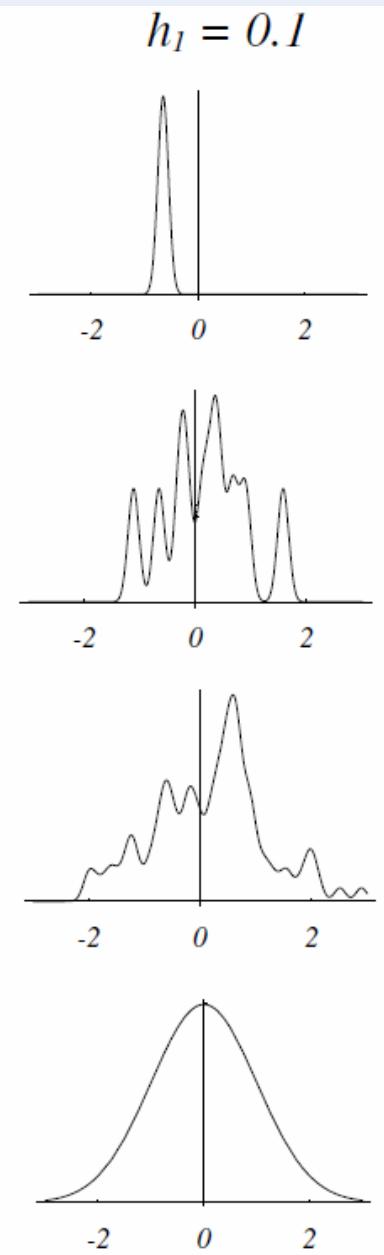
- Therefore, any DPF function can be served as the kernel function. This approach is called **Parzen-window approach**, it in fact interpolates the discrete points  $\{\mathbf{x}_i\} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n]$  into a continuous function PDF  $p(\mathbf{x})$ .

# Statistical Estimation & Machine Learning

A simple 1-D example.

$$K(\mathbf{u}) = N(0, \mathbf{I})$$
$$= \frac{1}{(2\pi)^{d/2}} \exp \left[ -\frac{1}{2} \mathbf{u}^T \mathbf{u} \right]$$

$$\hat{p}(\mathbf{x}) = \frac{1}{nh^d} \sum_{i=1}^n K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right)$$



# Statistical Estimation & Machine Learning

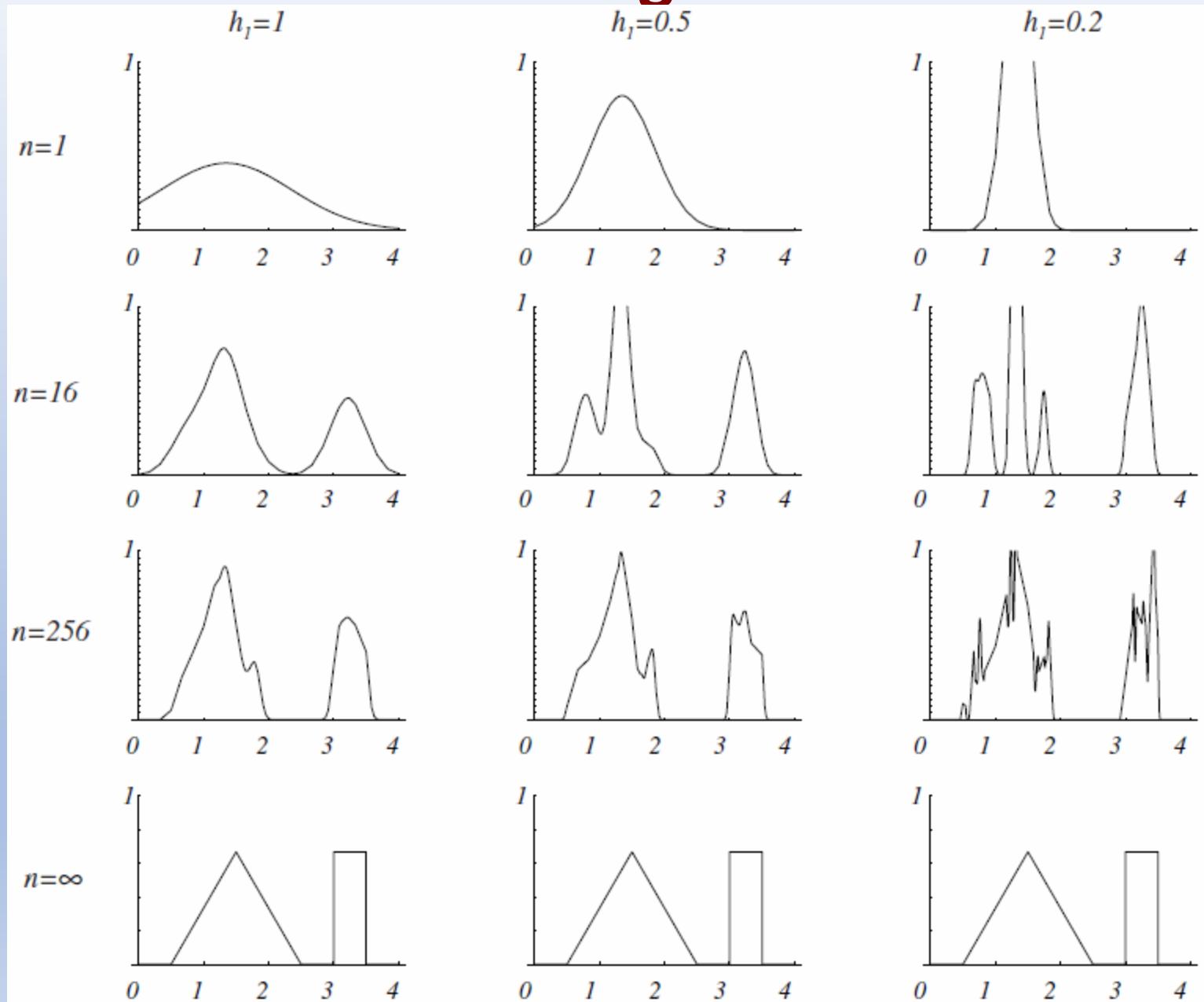
another

1-D example.

Bimodal distribution

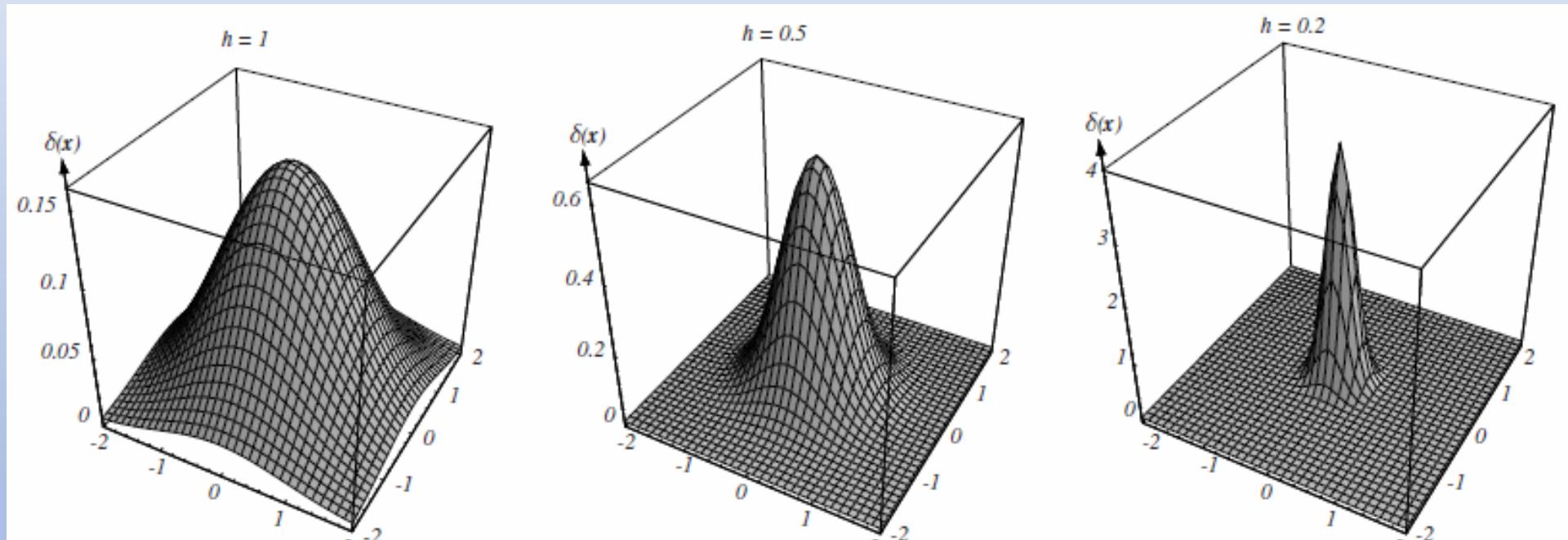
$$K(\mathbf{u}) = N(0, \mathbf{I})$$
$$= \frac{1}{(2\pi)^{d/2}} \exp \left[ -\frac{1}{2} \mathbf{u}^T \mathbf{u} \right]$$

$$\hat{p}(\mathbf{x}) = \frac{1}{nh^d} \sum_{i=1}^n K \left( \frac{\mathbf{x} - \mathbf{x}_i}{h} \right)$$



# Statistical Estimation & Machine Learning

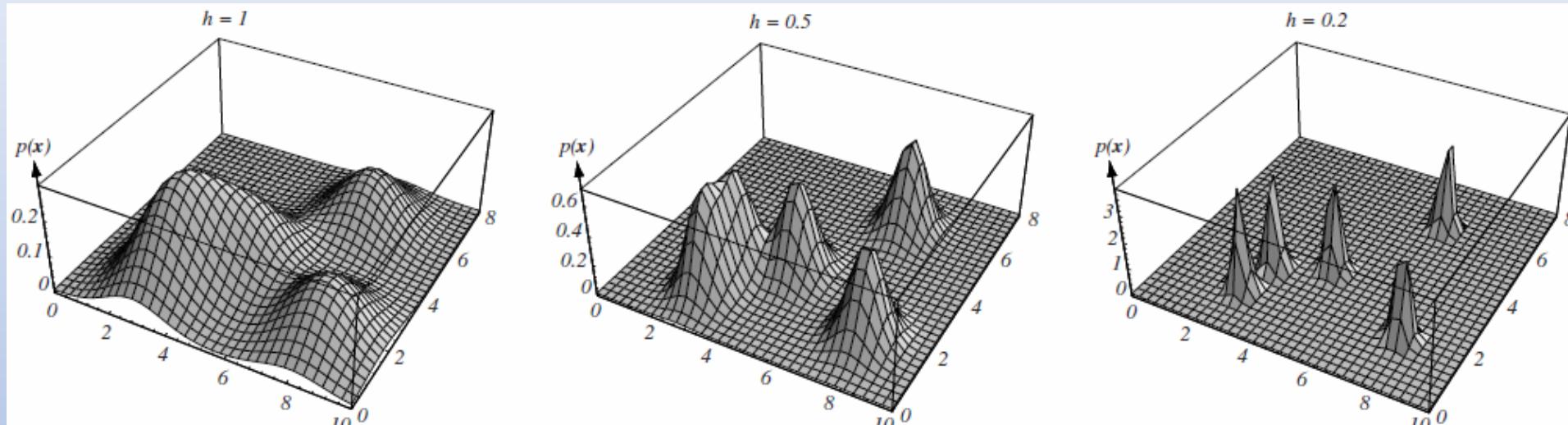
- A single 2-D training sample with Gaussian kernel function of different width  $h$ .



**FIGURE 4.3.** Examples of two-dimensional circularly symmetric normal Parzen windows for three different values of  $h$ . Note that because the  $\delta(x)$  are normalized, different

# Statistical Estimation & Machine Learning

- 5 2-D training samples with Gaussian kernel function of different width  $h$ .



**FIGURE 4.4.** Three Parzen-window density estimates based on the same set of five samples, using the window functions in Fig. 4.3. As before, the vertical axes have been scaled to show the structure of each distribution.

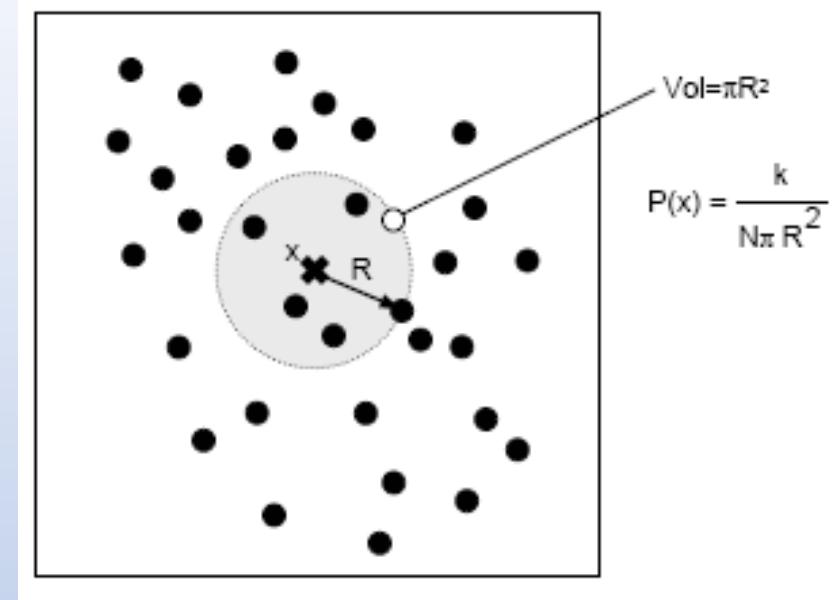
- The probabilistic neural networks and the RBF neural networks are almost equivalent to the Parzen-window approach.

$$\hat{p}(\mathbf{x}) = \frac{1}{nh^d} \sum_{i=1}^n K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right)$$

# Statistical Estimation & Machine Learning

- To estimate the PDF by

$$\hat{p}(\mathbf{x}) = \frac{k}{nV}$$



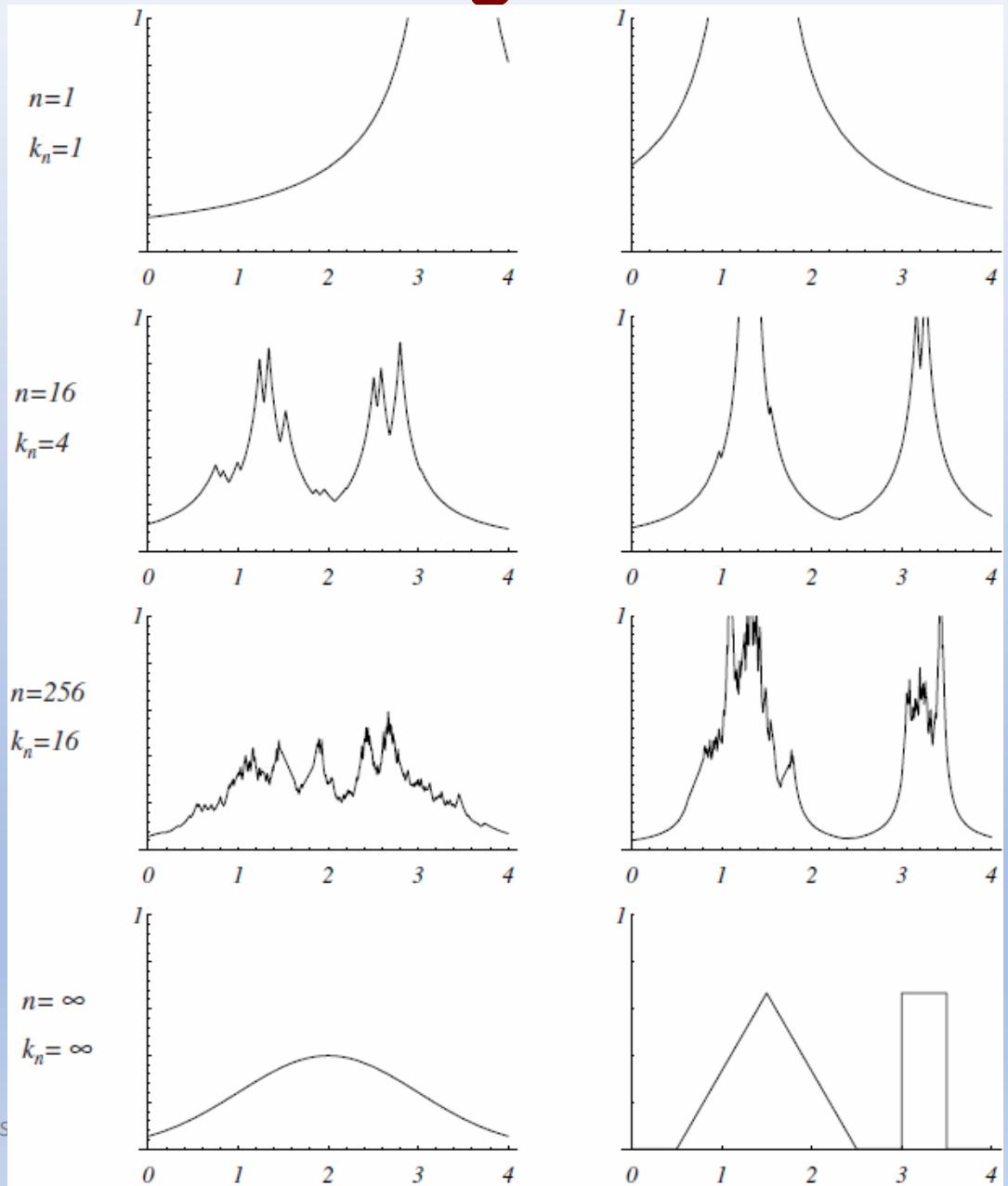
the Parzen-window approach selects a region/cell of fixed size (volume)  $V$  centered at  $\mathbf{x}$  and counts the number of samples  $k$  fall in the region/cell for the estimation of PDF.

- We can also select the fixed number of samples  $k$  and compute the size/volume that just encloses  $k$  samples to estimate the PDF by
- This approach is called  **$k$ -nearest-neighbor  $kNN$  estimation**.

# Statistical Estimation & Machine Learning

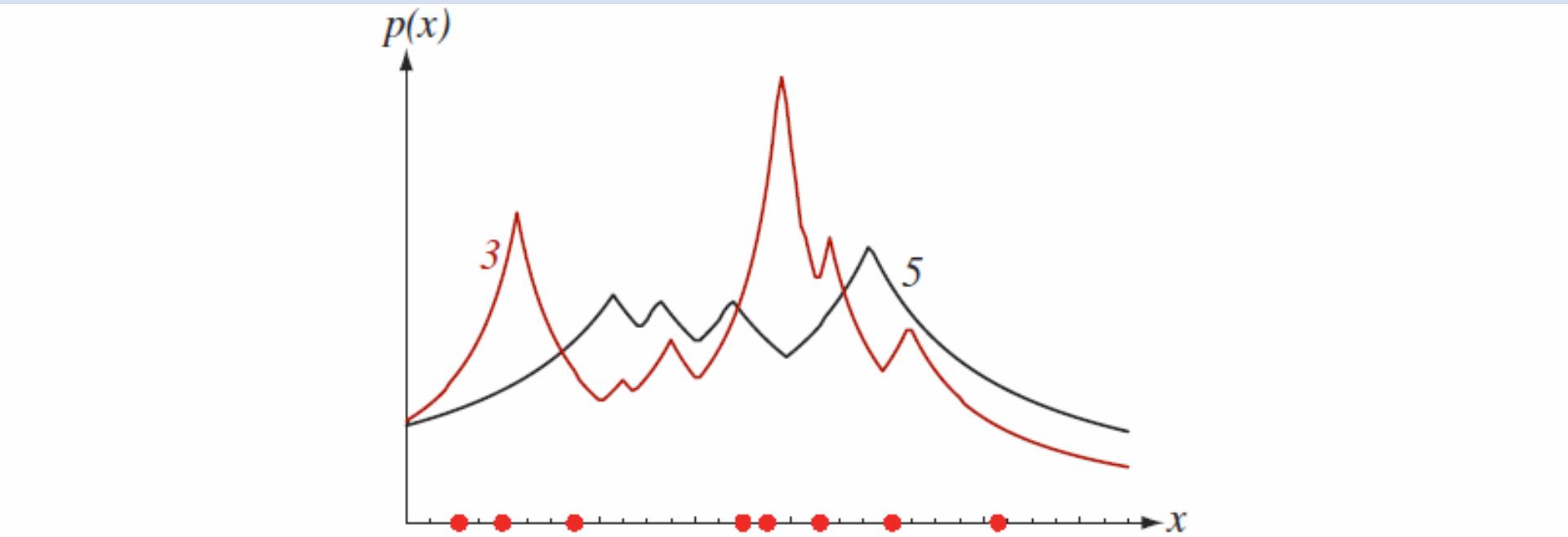
- Several k-nearest-neighbor estimates of two one-dimensional densities:

$$\hat{p}(\mathbf{x}) = \frac{k}{nV(k)}$$



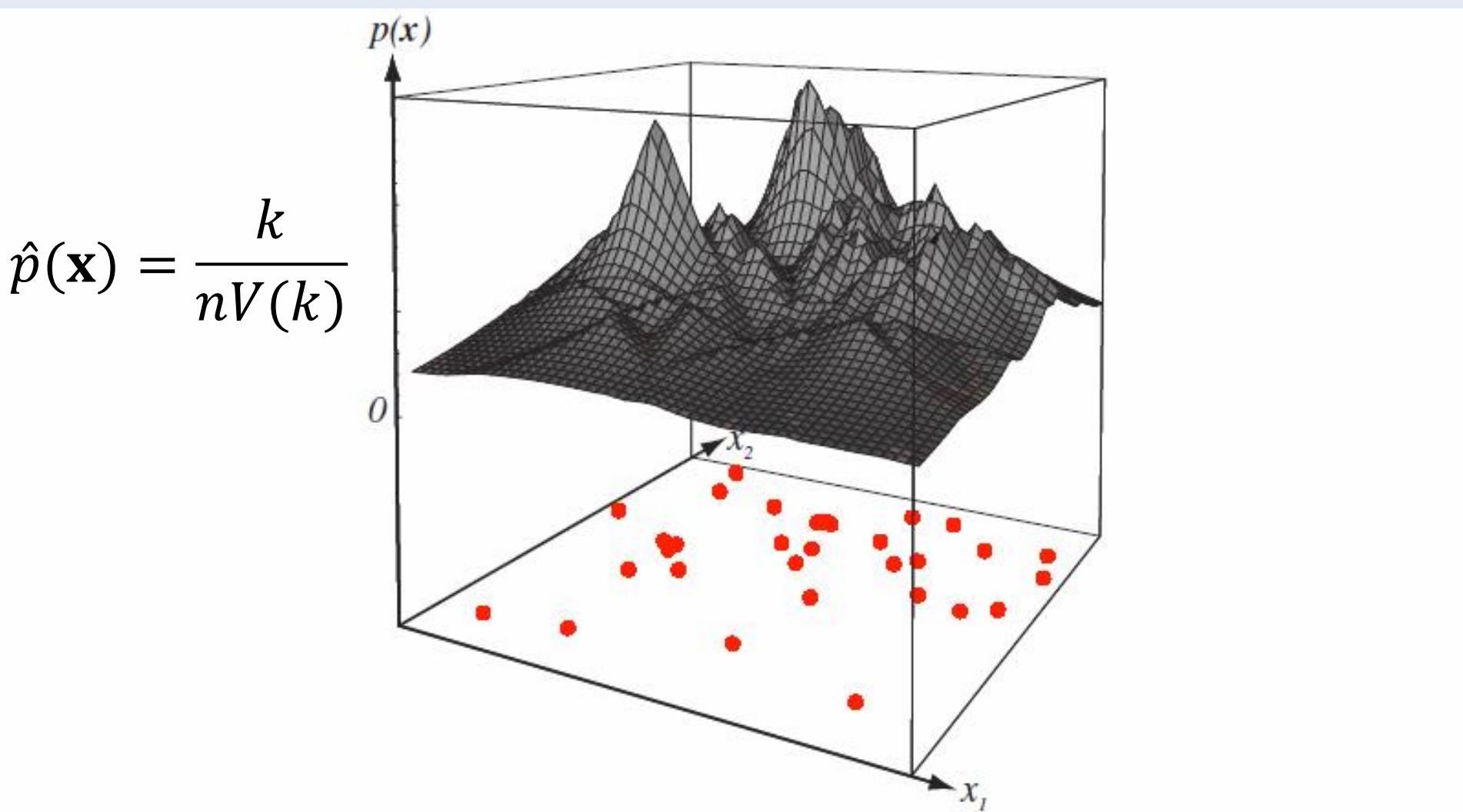
# Statistical Estimation & Machine Learning

$$\hat{p}(\mathbf{x}) = \frac{k}{nV(k)}$$



**FIGURE 4.10.** Eight points in one dimension and the  $k$ -nearest-neighbor density estimates, for  $k = 3$  and 5. Note especially that the discontinuities in the slopes in the estimates generally lie away from the positions of the prototype points. From: Richard

# Statistical Estimation & Machine Learning



# Statistical Estimation & Machine Learning

- The KNN technique can also be used for estimation of a-posteriori probabilities  $P(\omega_i|\mathbf{x})$  from a set of  $n$  labeled samples. (Compare to PNNs and Parzen window estimates.)
- Suppose that we place a cell of volume  $V$  around  $\mathbf{x}$  and capture  $k$  samples,  $k_i$  of which turn out to be labeled  $\omega_i$ .
- An estimate for the joint probability is  $p_n(\mathbf{x}, \omega_i) = \frac{k_i}{nV}$ .
- Hence, we can estimate  $P(\omega_i|\mathbf{x})$  by

$$\hat{p}(\mathbf{x}) = \frac{k}{nV(k)}$$

$$\begin{aligned}\hat{p}(\omega_i|\mathbf{x}) &= \hat{P}(\omega_i)\hat{p}(\mathbf{x}|\omega_i)/\hat{p}(\mathbf{x}) \\ &= \frac{n_i}{n} \frac{k_i}{n_iV(k)} \frac{nV(k)}{k} = \frac{k_i}{k}\end{aligned}$$

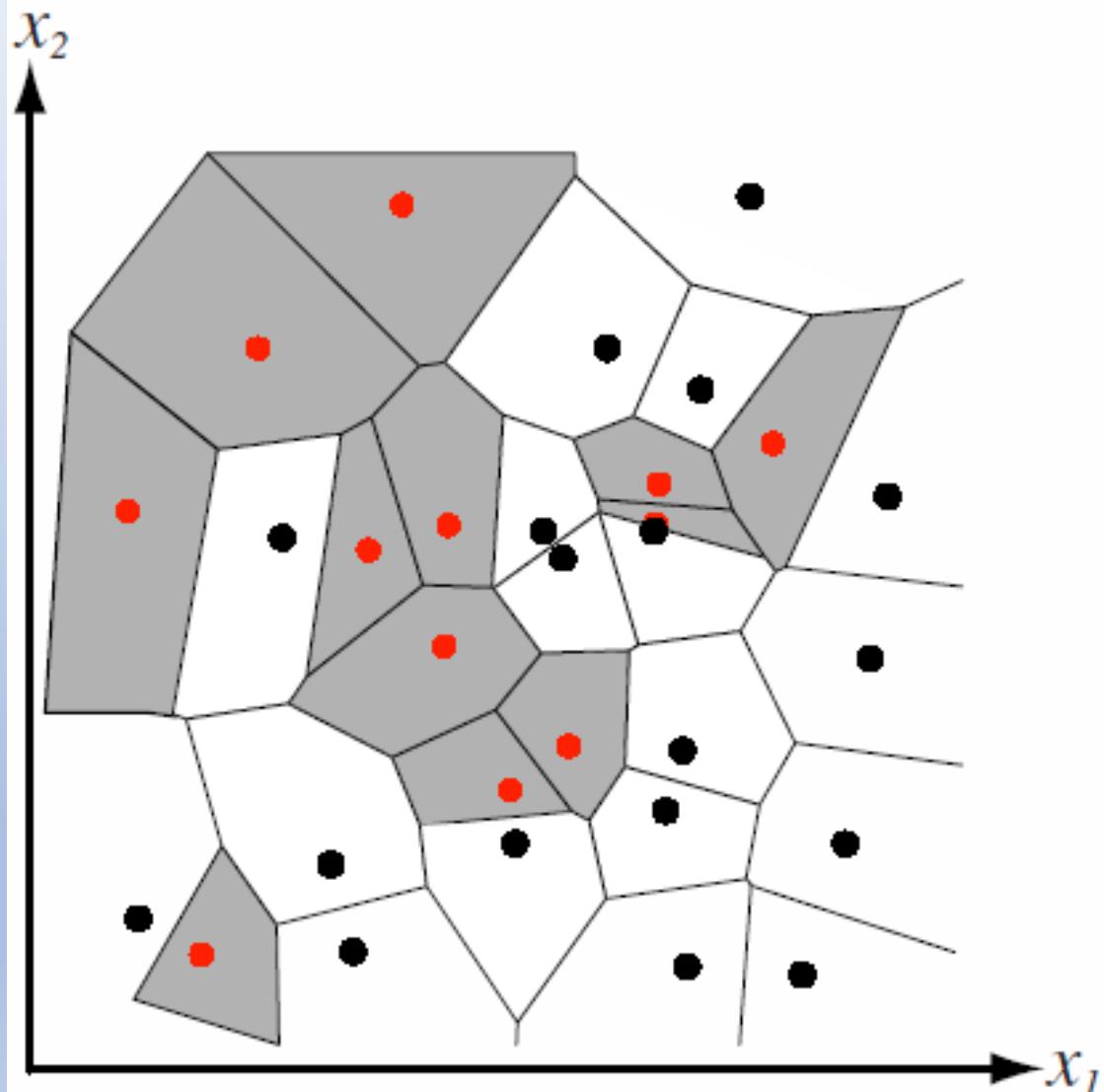
$$P_n(\omega_i|\mathbf{x}) = \frac{p_n(\mathbf{x}, \omega_i)}{\sum_{j=1}^c p_n(\mathbf{x}, \omega_j)} = \frac{k_i}{k}.$$

$$\hat{P}(\omega_i) = \frac{n_i}{n}$$

*k*-nearest-neighbor classifier,     $\hat{p}(\mathbf{x}|\omega_i) = \frac{k_i}{n_iV(k)}$   
*kNN classifier*

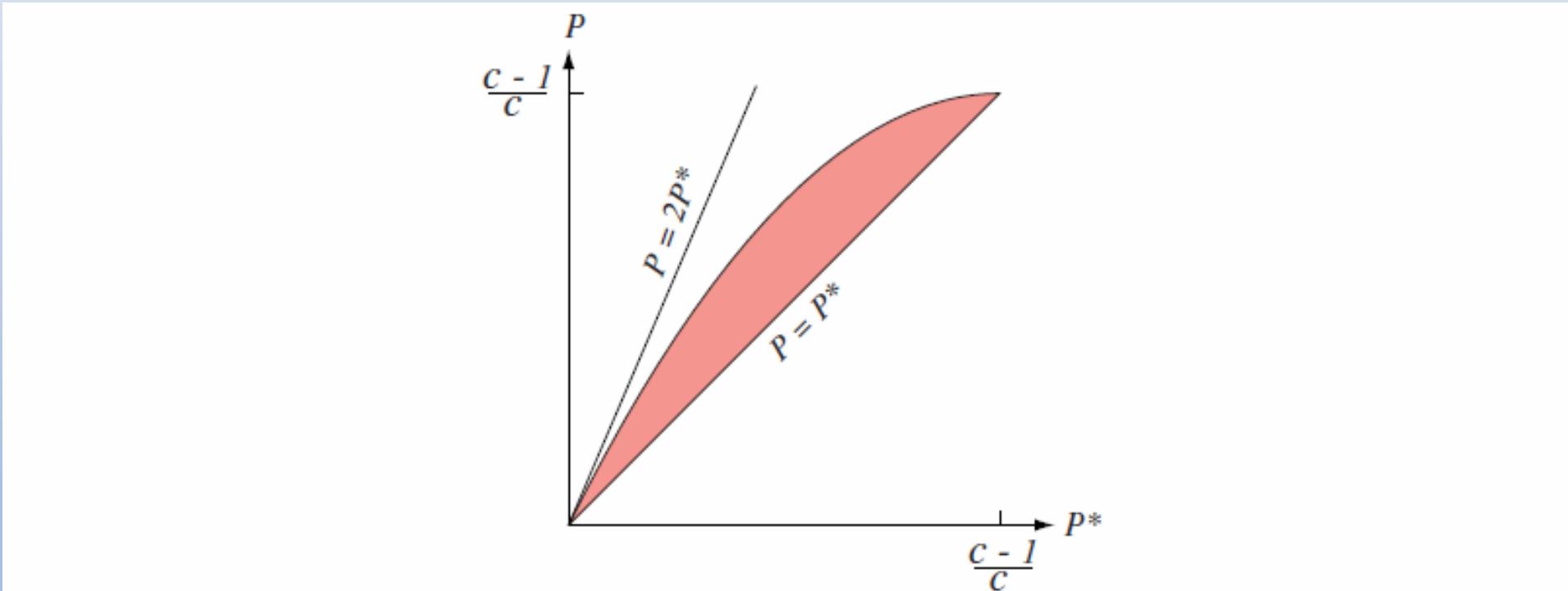
# Statistical Estimation & Machine Learning

- Classification boundary of 1<sup>st</sup>-NN classifier, also simply called NN classifier.



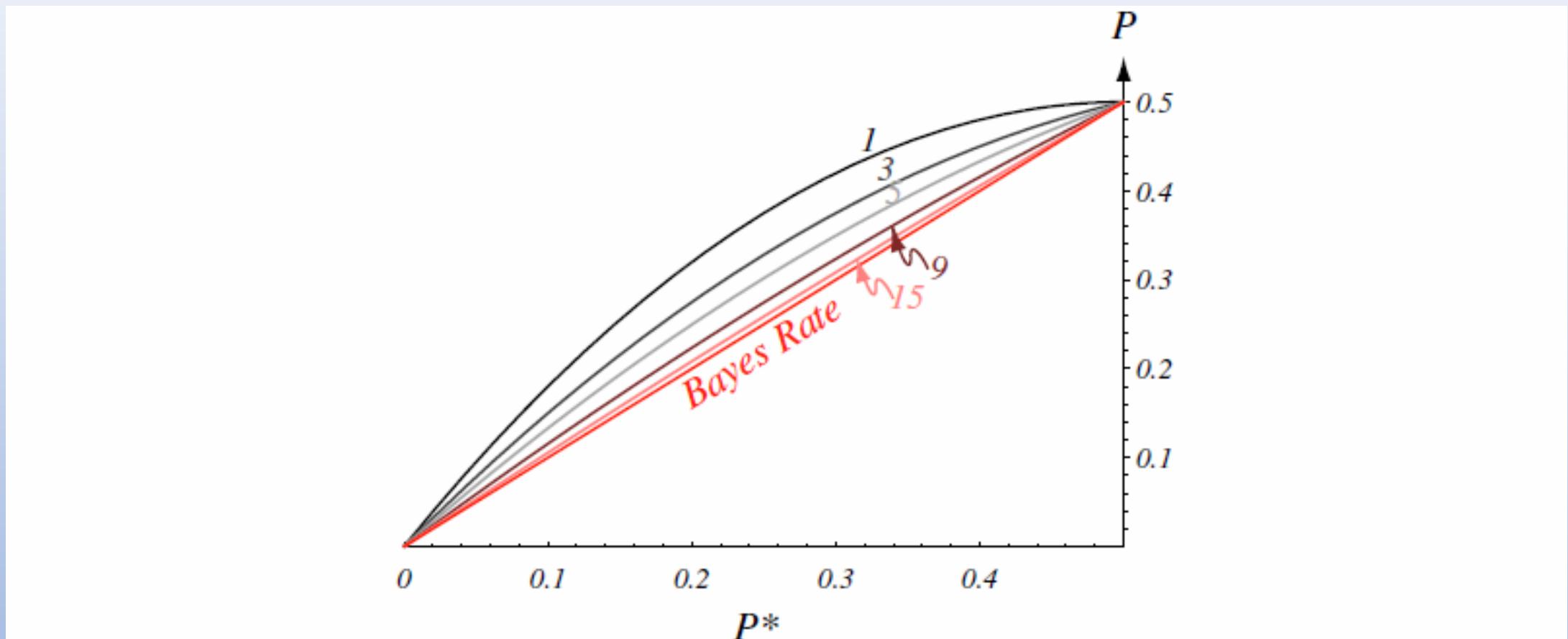
# Statistical Estimation & Machine Learning

- The error rate of NN classifier  $P$  for very large number of training samples are bounded as  $P^* \leq P \leq P^* \left( 2 - \frac{c}{c-1} P^* \right)$



**FIGURE 4.14.** Bounds on the nearest-neighbor error rate  $P$  in a  $c$ -category problem given infinite training data, where  $P^*$  is the Bayes error (Eq. 52). At low error rates, the nearest-neighbor error rate is bounded above by twice the Bayes rate. From: Richard O.

# Statistical Estimation & Machine Learning



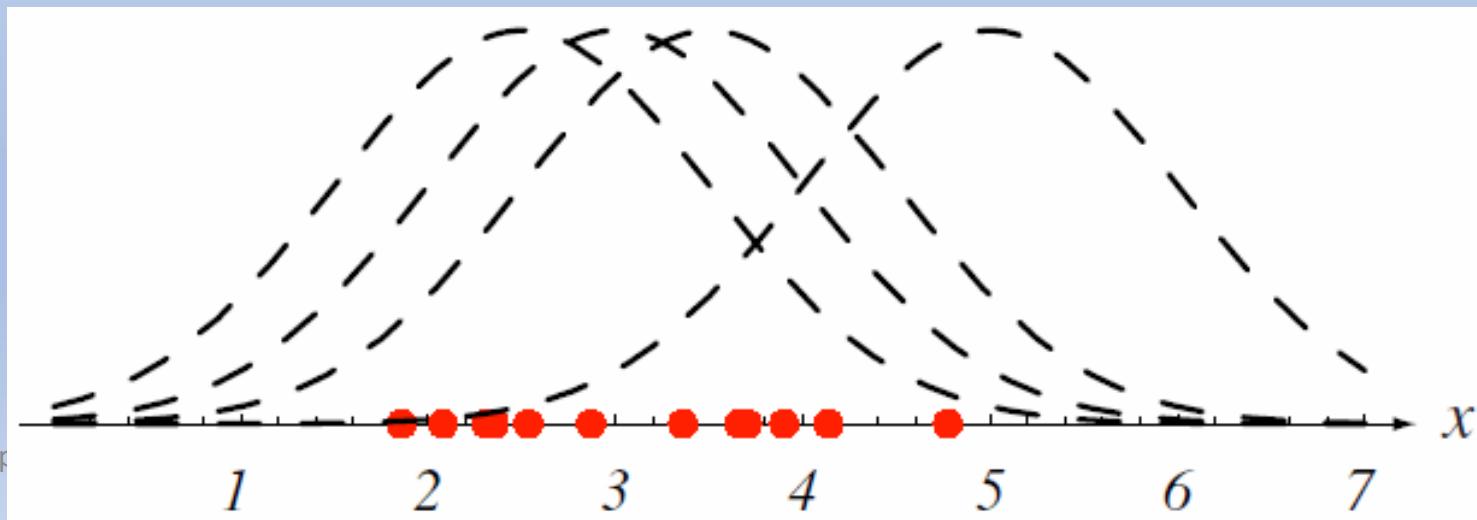
**FIGURE 4.16.** The error rate for the  $k$ -nearest-neighbor rule for a two-category problem is bounded by  $C_k(P^*)$  in Eq. 54. Each curve is labeled by  $k$ ; when  $k = \infty$ , the estimated probabilities match the true probabilities and thus the error rate is equal to the Bayes rate, that is,  $P = P^*$ . From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern*

# Statistical Estimation & Machine Learning

- We see that the learned conditional PDF from training samples could greatly deviate from the true PDF of the population, especially in case of small number of training samples.
- If our general knowledge about the problem permits us to model the conditional PDF, i.e. using a mathematical analytical function to represent the PDF with unknown parameter. The severity of these problems can be reduced significantly. Here we parameterize the conditional PDF, which is called parametric method.
- Neural networks and deep learning are also parametric method.
- But we start from the simplest. Suppose that  $p(\mathbf{x}|\omega_k)$  is a Gaussian density with mean  $\mu_k$  and covariance matrix  $\Sigma_k$ . Although we do not know their values, this knowledge simplifies the problem from estimating an unknown function  $p(\mathbf{x}|\omega_k)$  to estimating the unknown parameters  $\mu_k$  and  $\Sigma_k$  only.

# Statistical Estimation & Machine Learning

- Now we will use a set of training samples  $D=\{\mathbf{x}_i\}=[\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n]$  drawn independently from the probability density  $p(\mathbf{x}|\theta)$  to estimate the unknown parameter vector  $\theta$ .
- Lets see an example to generate idea how to estimate the parameter of a given probability density  $p(\mathbf{x}|\theta)$  reasonably based on the training data D.
- The graph shows several training points in one dimension, known or assumed to be drawn from a Gaussian of a particular variance, but unknown mean. Four PDF with 4 different means are shown in dashed lines.
- Which PDF you should choose?



# Statistical Estimation & Machine Learning

- Now we formulate our idea mathematically.
- Obviously, the probability that a sample  $\mathbf{x}_k$  occurs is  $p(\mathbf{x}_k|\theta)$ . As all samples in the training set are independently collected (occur), the probability that all samples occur is

$$p(D|\theta) = \prod_{k=1}^n p(\mathbf{x}_k|\theta)$$

- Intuitively, we should select the parameter so that the probability density  $p(\mathbf{x}|\theta)$  best supports the actually observed training samples, i.e. to make the probability of all training data occur  $p(D|\theta)$  maximal. Note that  $p(D|\theta)$  is called the likelihood of  $\theta$  with respect to the set of samples  $D$ . Thus, this method is called the maximum likelihood (ML) estimation.

$$\hat{\theta} = \arg \max_{\theta} p(D|\theta) = \arg \max_{\theta} \prod_{k=1}^n p(\mathbf{x}_k|\theta)$$

# Statistical Estimation & Machine Learning

- It is often not easy to get an analytical solution of

$$\hat{\theta} = \arg \max_{\theta} p(D|\theta) = \arg \max_{\theta} \prod_{k=1}^n p(\mathbf{x}_k|\theta)$$

due to the multiplication of the functions of  $\theta$  and  $p(\mathbf{x}_k|\theta)$  is often nonlinear function of  $\theta$ .

- Since the logarithm is monotonically increasing, **maximizing the logarithm of a function also maximizes the function itself**. The logarithm has nice property that converts the multiplication into summation and simplifies the exponential function.
- Thus, we maximize the log-likelihood instead of maximizing the likelihood

$$\hat{\theta} = \arg \max_{\theta} \ln p(D|\theta) = \arg \max_{\theta} \sum_{k=1}^n \ln p(\mathbf{x}_k|\theta)$$

# Statistical Estimation & Machine Learning

- The solution can be found by the standard methods of differential calculus:  
Solving the equation that the gradient is zero.

$$\nabla_{\boldsymbol{\theta}} \ln p(D|\boldsymbol{\theta}) = 0$$
$$\sum_{k=1}^n \nabla_{\boldsymbol{\theta}} \ln p(\mathbf{x}_k|\boldsymbol{\theta}) = 0$$

- If the number of parameters to be estimated is  $q$ , then  $\boldsymbol{\theta}$  is a  $q$ -component vector  $\boldsymbol{\theta}=(\theta_1, \theta_2, \dots, \theta_q)^T$ . The gradient is a vector that contains partial differentiation against all components of  $\boldsymbol{\theta}$ .

$$\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\theta}) \triangleq \begin{pmatrix} \frac{\partial f(\boldsymbol{\theta})}{\partial \theta_1} \\ \vdots \\ \frac{\partial f(\boldsymbol{\theta})}{\partial \theta_q} \end{pmatrix}$$

# Statistical Estimation & Machine Learning

- To see how maximum likelihood methods results apply to a specific case, suppose that the samples are drawn from a **multivariate Gaussian** population with unknown **mean  $\mu$**  and **covariance matrix  $\Sigma$** .

$$p(\mathbf{x}|\boldsymbol{\theta}) = \frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2}} \exp \left[ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right]$$

- The log-likelihood of a single sample is

$$\ln p(\mathbf{x}_k|\boldsymbol{\theta}) = -\frac{1}{2} \ln[(2\pi)^d |\boldsymbol{\Sigma}|] - \frac{1}{2} (\mathbf{x}_k - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_k - \boldsymbol{\mu})$$

- Consider first the univariate case with  $\boldsymbol{\theta} = (\theta_1, \theta_2)^T = (\mu, \sigma^2)^T$ .

$$p(x|\boldsymbol{\theta}) = \frac{1}{\sqrt{2\pi}\sigma} \exp \left[ -\frac{1}{2} \left( \frac{x - \mu}{\sigma} \right)^2 \right]$$

# Statistical Estimation & Machine Learning

- Here the log-likelihood of a single sample is simplified as

$$\ln p(x_k | \boldsymbol{\theta}) = -\frac{1}{2} \ln[2\pi\sigma^2] - \frac{1}{2\sigma^2} (x_k - \mu)^2$$

- Its derivative is

$$\nabla_{\boldsymbol{\theta}} \ln p(x_k | \boldsymbol{\theta}) = \begin{pmatrix} \frac{1}{\sigma^2} (x_k - \mu) \\ -\frac{1}{2\sigma^2} + \frac{(x_k - \mu)^2}{2\sigma^4} \end{pmatrix}$$

# Statistical Estimation & Machine Learning

- Applying ML

$$\nabla_{\boldsymbol{\theta}} \ln p(D|\boldsymbol{\theta}) = \sum_{k=1}^n \nabla_{\boldsymbol{\theta}} \ln p(\mathbf{x}_k|\boldsymbol{\theta}) = 0$$

- We have

$$\sum_{k=1}^n \frac{1}{\sigma^2} (x_k - \mu) = 0$$

$$-\sum_{k=1}^n \frac{1}{2\sigma^2} + \sum_{k=1}^n \frac{(x_k - \mu)^2}{2\sigma^4} = 0$$

- Solve these two equations we have the following  
**maximum likelihood estimates**

$$\hat{\mu} = \frac{1}{n} \sum_{k=1}^n x_k .$$

$$\hat{\sigma}^2 = \frac{1}{n} \sum_{k=1}^n (x_k - \hat{\mu})^2$$

# Statistical Estimation & Machine Learning

- While the analysis of the multivariate case is basically very similar, considerably more manipulations are involved. The result is that the maximum likelihood estimates for **mean vector  $\mu$  and covariance matrix  $\Sigma$**  of multivariate Gaussian PDF

$$p(\mathbf{x}|\boldsymbol{\theta}) = \frac{1}{(2\pi)^{d/2} |\boldsymbol{\Sigma}|^{1/2}} \exp \left[ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right]$$

are given by

$$\hat{\boldsymbol{\mu}} = \frac{1}{n} \sum_{k=1}^n \mathbf{x}_k$$

$$\hat{\boldsymbol{\Sigma}} = \frac{1}{n} \sum_{k=1}^n (\mathbf{x}_k - \hat{\boldsymbol{\mu}})(\mathbf{x}_k - \hat{\boldsymbol{\mu}})^T$$

# **Topic 8**

## **Handcrafted Feature Generation and Feature Selection**

# Visual Object Detection

Given an image, visual object is only part of it at unknown position and size.

Given an image  $F$ , the detection process is:

1. Use a window of size  $m$  by  $n$  to sample a region from image  $F$ ,  $W = F(i+1:i+m, j+1:j+n)$ .
2. Classify  $W$  to positive or negative class.
3. Iteratively do steps 1 and 2 by increasing  $i, j$ .
4. Iteratively do steps 1, 2, 3 by increasing  $m, n$ .

We see that the detection needs numerous times of binary classification. So we need fast feature extraction and classification.



# Visual Object Detection

## Three challenges for feature extraction

How can we compute features quickly?

How can we refrain from repeated computation in different feature?

How do we obtain the good representative features?

## One solution, Adaboost using Harr-like features

Reference:

Viola, P., Jones, M. (2001). Robust Real-time Object Detection. 2. *Int. Workshop on Statistical and Computational Theories on Vision – Modeling, Learning, Computing and Sampling*. <https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-IJCV-01.pdf>.



Paul Viola

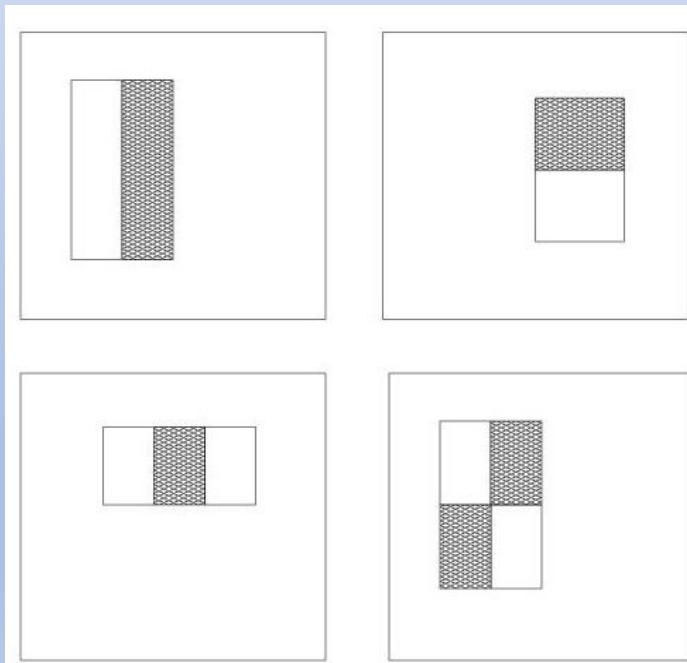


Michael Jones

# Feature Generation

Harr-like features can be extracted/computed rapidly

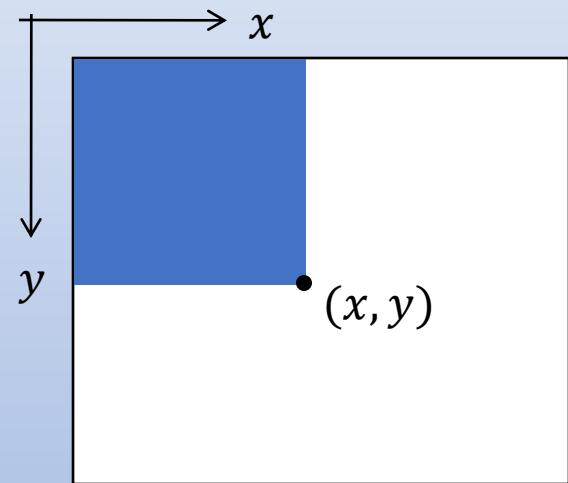
Given an image window, the feature is the brightness differences between different rectangle regions. The feature value is the difference between the sum of the pixels in the two rectangular regions.



# Feature Generation

Given an image  $i(x, y)$ , we compute an integral image  $ii(x, y)$ , whose value at location  $(x, y)$  is the sum of all pixel values above and left of  $(x, y)$ .

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y')$$



Recursive computation:

$s(x, y)$  is the cumulative column sum

$$\begin{aligned}s(x, y) &= s(x, y - 1) + i(x, y) \\ ii(x, y) &= ii(x - 1, y) + s(x, y)\end{aligned}$$

# Feature Generation

Using the integral image, we can compute the value of any rectangular sum in short and constant time.

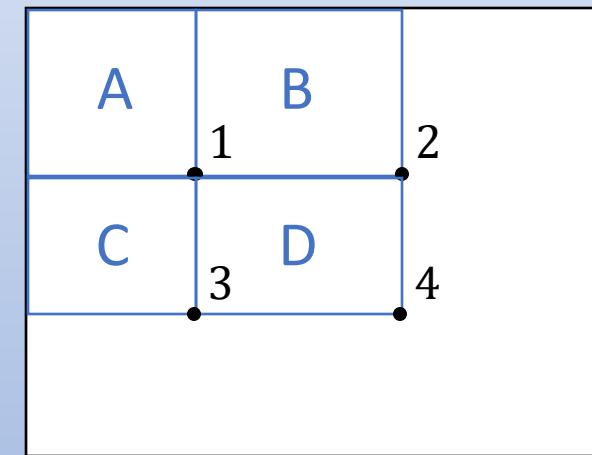
For example:

The integral sum inside rectangle D can be computed as:

$$ii(4) + ii(1) - ii(2) - ii(3)$$

As a result: two-, three-, and four-rectangular features can be computed with 6, 8 and 9 array references respectively.

Now that's fast!



# Feature Generation

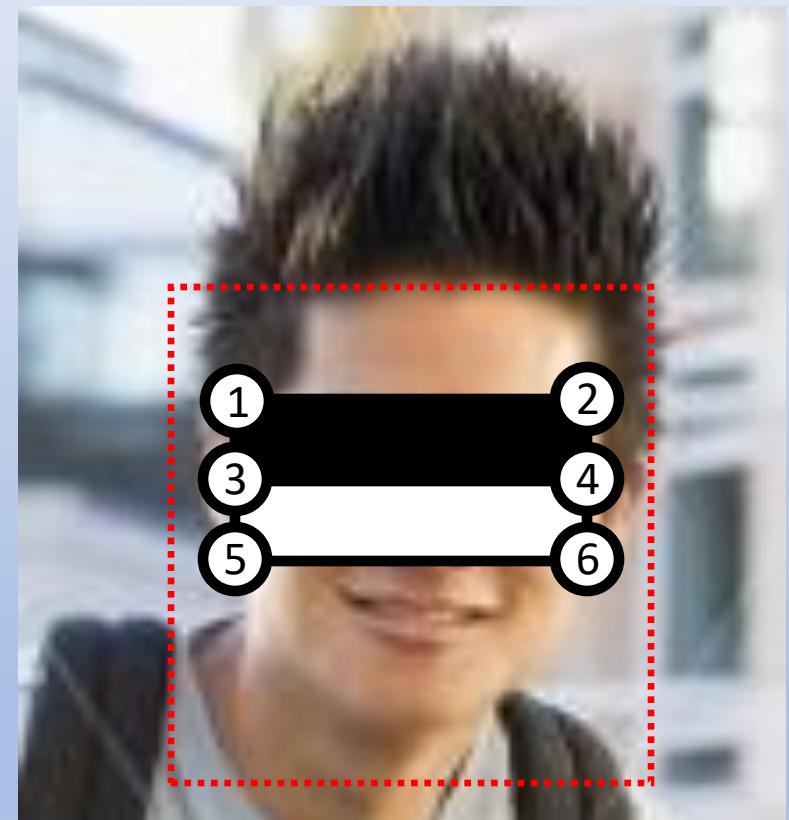
Integral image enables us to compute features of all rectangle sizes in short and constant time.



Therefore, no image scaling is necessary.



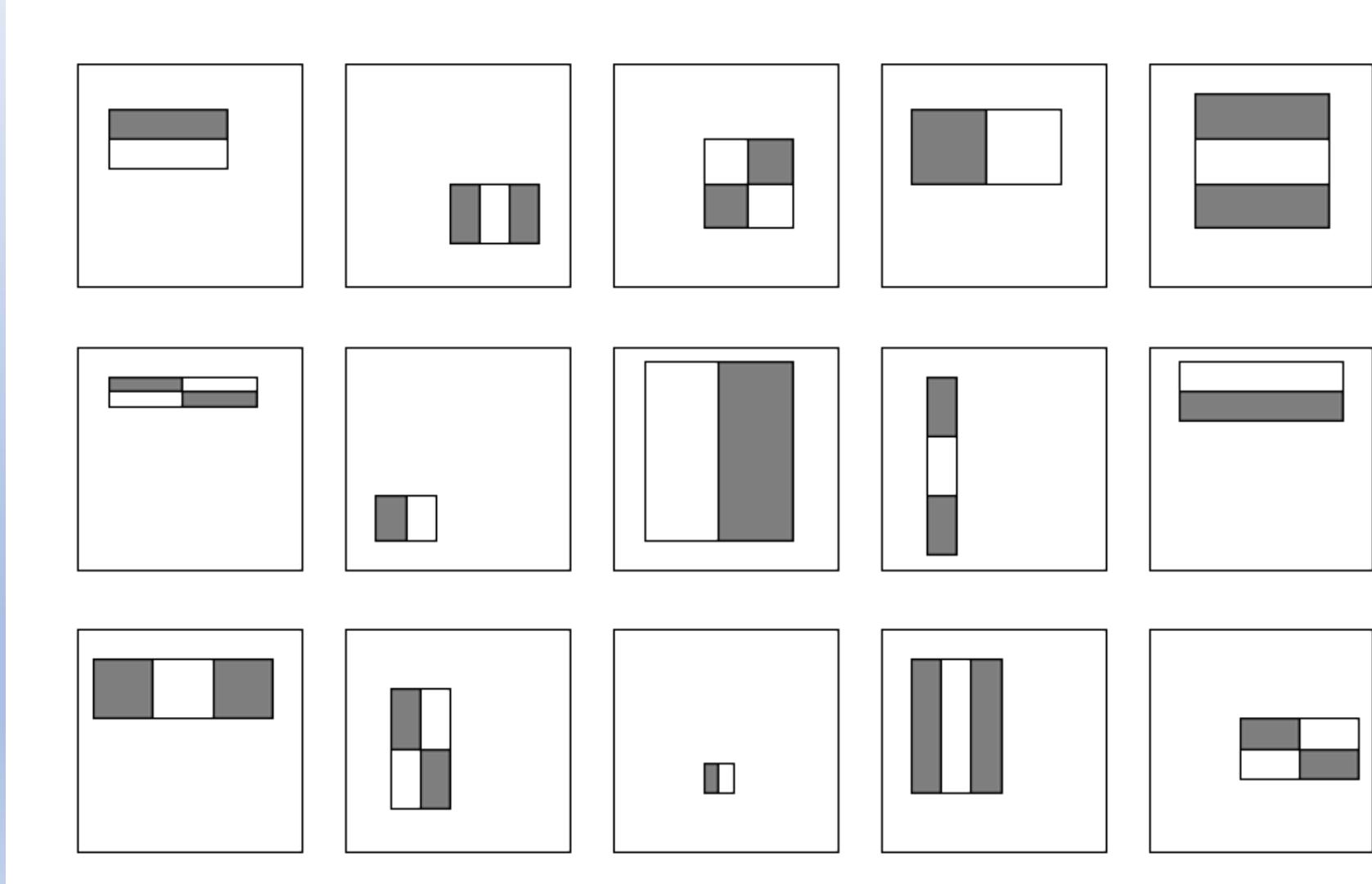
Scale the rectangular features instead!



# Feature Generation

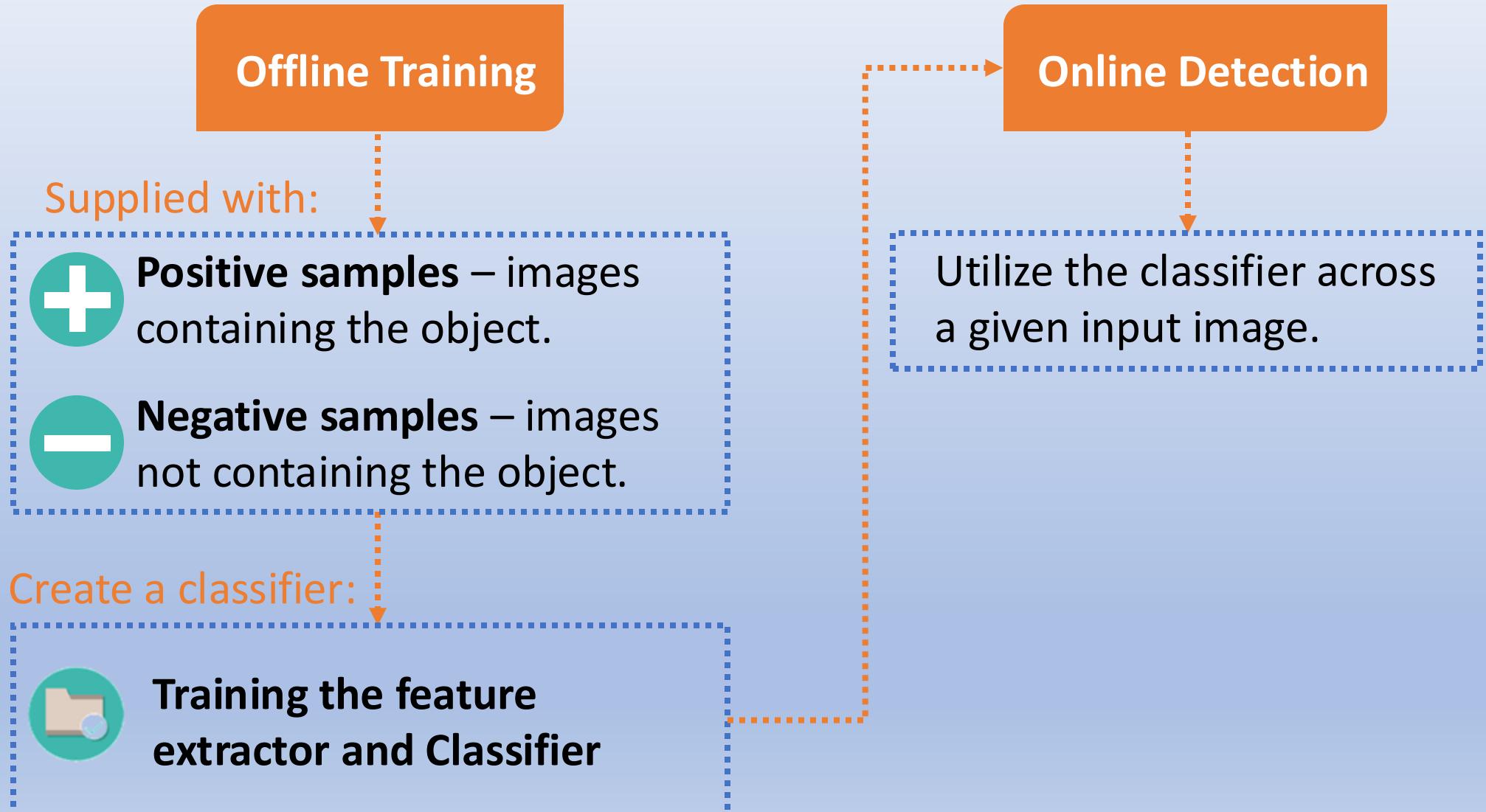
For a  $24 \times 24$  detection region, the number of possible rectangle features is ~180,000!

We need feature selection



Machine  
learns the  
features  
and  
classifier

# Feature Selection



# Feature Selection

4,916 positive training examples were hand picked, aligned, normalised, and scaled to a base resolution of  $24 \times 24$ .



10,000 negative examples were selected by randomly picking sub-windows from 9500 images which did not contain faces.

# Feature Selection

Boosting is a machine learning meta-algorithm for performing supervised learning.



Creates a “strong” classifier from a set of “weak” classifiers.

## Definitions:

### “weak” classifier

Has an error rate  $< 0.5$   
(i.e. a better than average advice).

### “strong” classifier

Has an error rate of  $\epsilon$   
(i.e. our final classifier).

# Feature Selection

AdaBoost for aggressive feature selection.



Training set  
consists of positive  
and negative  
**images**.

Simple classifier  
uses only a single  
**feature**.

**Variety** is the key here – if we want a small number of features – we must make sure they compensate each other's flaws.

# Feature Selection

AdaBoost is a boosting algorithm for searching out a small number of good classifiers/features which have **significant variety**.

AdaBoost accomplishes this by endowing misclassified training examples with more weight (thus enhancing their chances to be classified correctly by next classifier).

The weights tell the learning algorithm the importance of the example.

Stands for  
“Adaptive boost”.

# Feature Selection

Design a simple classifier that uses only a single feature.

Hence, there are 180k classifiers to choose from.

For each classifier, we search an **optimal threshold** such that the minimum number of training samples are misclassified.

$$h_j = \begin{cases} -1, & f_j(x) < \theta_j \\ +1, & \text{else;} \end{cases}$$

value of one rectangle feature

$h_j$  : simple classifier

$f_j$  : feature

$\theta_j$  : threshold, selected optimally from training samples

# Feature Selection

Given  $m$  example images  $(x_1, y_1), \dots, (x_m, y_m)$  where  $y_1 = \{-1, +1\}$

For example:  $(x_1, -1) \rightarrow$  negative example,  $(x_9, +1) \rightarrow$  positive example

- Initialise  $D_1(i) = 1/m, i = 1, \dots, m$
- For  $t = 1, \dots, T$ : ( $T$  is the total number of weak classifiers)
  - Find the classifier  $h_t$  that minimizes the error with respect to the distribution  $D_t$ :

$$h_t = \arg \min_j \varepsilon_j \quad \text{where} \quad \varepsilon_j = \sum_{i=1}^m D_t(i)[y_i \neq h_j(x_i)]$$

( $\varepsilon_j$  is the sum of  $D_t(i)$  for all mismatched  $x_i$ ;  $h_t$  is selected with the smallest  $\varepsilon_j$ .)

- If  $\varepsilon_j \geq 0.5$  then stop;

# Feature Selection

- Assign the weight to the classifier  $\alpha_t = \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}$

where  $\varepsilon_t$  is the weighted error rate of classifier  $h_t$ .

- Update the weights of training samples

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t \cdot y_i \cdot h_t(x_i))}{Z_t}$$

where  $Z_t$  is a normalization factor (chosen so that  $D_{t+1}$  will be a probability distribution, i.e. sum one over all  $x$ ).

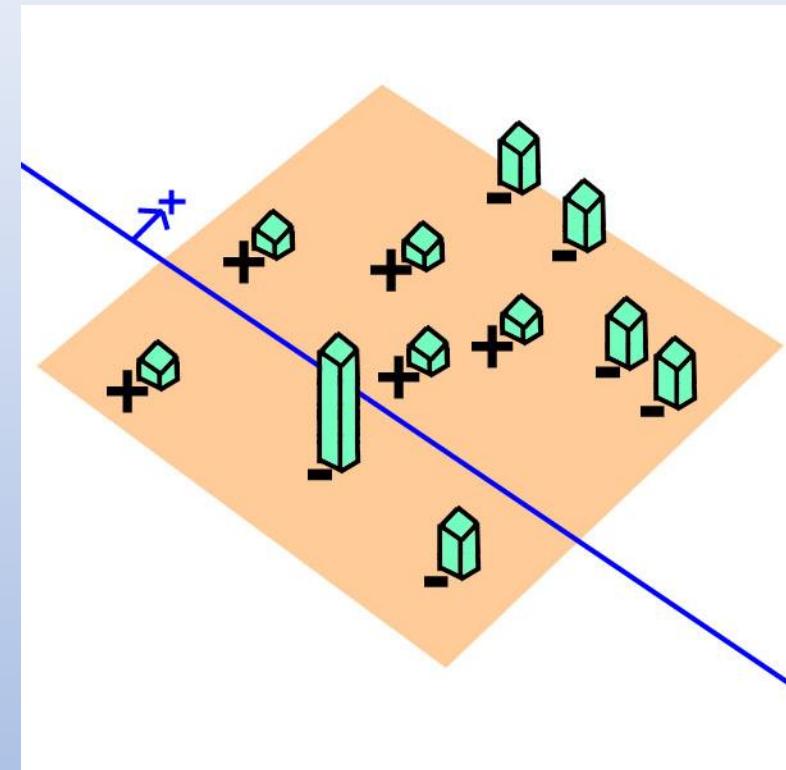
- Output the final classifier

$$h(x) = \begin{cases} 1, & \sum_{t=1}^T \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \alpha_t \\ 0, & \text{otherwise} \end{cases}$$

# Feature Selection

- Adaboost starts with a uniform distribution of “weights” over training examples.
- Select the classifier with the lowest weighted error (i.e. a “weak” classifier)
- Increase the weights on the training examples that were misclassified.
- (Repeat)
- At the end, carefully make a linear combination of the weak classifiers obtained at all iterations.

$$h_{\text{strong}}(\mathbf{x}) = \begin{cases} 1 & \alpha_1 h_1(\mathbf{x}) + K + \alpha_n h_n(\mathbf{x}) \geq \frac{1}{2}(\alpha_1 + K + \alpha_n) \\ 0 & \text{otherwise} \end{cases}$$



# Feature Selection

We can now train a classifier as accurate as we desire **on the training data**.

By increasing the number of features, we:

- Increase detection accuracy.
- Decrease detection speed.

Experiments showed that a 200 feature classifier makes a good face detector:

- Takes 0.7 seconds to scan an 384 by 288 pixel image.

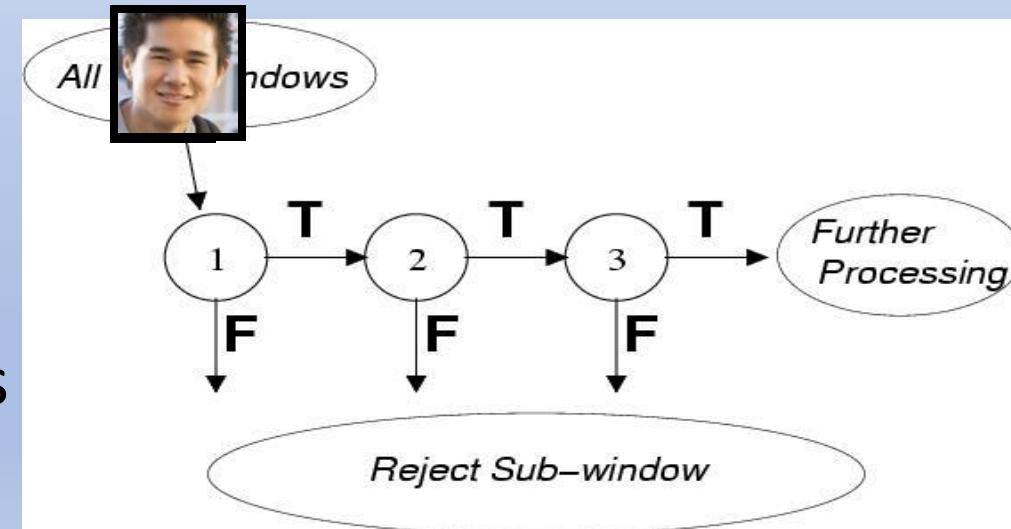
## Problem:

Not real time for video! (At most 0.067 seconds needed).

# Feature Selection

## The attentional cascade Classifier

- Overwhelming majority of windows are in fact negative.
- Simpler, boosted classifiers can reject many of negative sub-windows while detecting all positive instances.
- A cascade of gradually more complex classifiers achieves good detection rates.
- Consequently, on **average**, much fewer features are calculated per window.



# Visual Object Detection

Key points discussed in this topic:

- The face detection is a typical task of general object detection.
- Using the integral image representation and simple rectangular features eliminates the need of expensive calculation of multi-scale image pyramid.
- AdaBoost gives a general technique for efficient feature selection.
- A general technique for constructing a cascade of homogeneous classifiers is presented, which can reject most of the negative examples at early stages of processing thereby significantly reducing computation time.

# Visual Object Detection

Viola & Jones was the first real-time face detection system.

Was widely adopted and re-implemented.

Intel distributes this algorithm in a computer vision toolkit (OpenCV)



Actual output of Intel's implementation

# Topic 9

## Visual Data Dimensionality Reduction as Feature Extraction

# Feature Extraction / Dimensionality Reduction

- In some applications such as visual object detection and recognition, bioinformatics and data mining, high data dimensionality imposes great burdens on the robust and accurate recognition.
- Feature extraction or dimensionality reduction thus becomes a separate and maybe the most critical module of such recognition systems.
- Extracting the **discriminative** and **reliable** features or dimensions is always the **key step** for intelligent system.
- Feature extraction and dimensionality reduction can be based on
  - Human expert knowledge
  - Image local structures
  - Image global structure
  - Machine learning from training database

# Feature Extraction/Dimensionality Reduction by Machine Learning

Take all pixels of the whole image as initial features and derive the effective features based on machine learning.



$$[300 \times 200] \Rightarrow \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{60000} \end{pmatrix} \Rightarrow \mathbf{f} = \begin{bmatrix} f \\ \vdots \\ f_{60} \end{bmatrix} = \Phi^T \mathbf{x}$$

Feature Scaling,  
Dimension Reduction  
Feature Extraction

The transform parameter is determined by machine (computer) learning (training) from an image database.



# PCA: Principal Component Analysis

Given a data set of  $q$   $n$ -dimensional training samples

$$\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_q$$

Each sample is a point in an  $n$ -dimensional space.

How to use one point,  $\mathbf{x}_0$  to best represent all training data?

i.e.  $\varepsilon^2 = \sum_{i=1}^q \|\mathbf{x}_0 - \mathbf{x}_i\|^2 = \sum_{i=1}^q (\mathbf{x}_0 - \mathbf{x}_i)^T (\mathbf{x}_0 - \mathbf{x}_i) \Rightarrow \text{minimum}$

It is very easy to prove that **the solution is the sample mean**

$$\mathbf{x}_0 = \boldsymbol{\mu} = \frac{1}{q} \sum_{i=1}^q \mathbf{x}_i$$

Just for symbolic simplicity, we **centralize training samples** and define a training **data matrix** by  $\tilde{\mathbf{x}}_i = \mathbf{x}_i - \boldsymbol{\mu}$ ,  $\mathbf{X} = [\tilde{\mathbf{x}}_1 \quad \tilde{\mathbf{x}}_2 \quad \dots \quad \tilde{\mathbf{x}}_q]$

# PCA: Principal Component Analysis

Now we want to just use one dimension  $\phi$ ,  $||\phi||^2 = \phi^T \phi = 1$   
to best represent all samples,  $\tilde{\mathbf{x}}_i$ .

The  $n$ -dimensional data  $\tilde{\mathbf{x}}_i$  are reduced to one-dimensional data.

$$a_i = \phi^T \tilde{\mathbf{x}}_i$$

The best dimension  $\phi$  makes reconstruction error minimum.

$$\varepsilon^2 = \sum_{i=1}^q ||\tilde{\mathbf{x}}_i - a_i \phi||^2 \Rightarrow \text{minimum}$$

$$\begin{aligned} \varepsilon^2 &= \sum_{i=1}^q ||\tilde{\mathbf{x}}_i - a_i \phi||^2 = \sum_{i=1}^q (\tilde{\mathbf{x}}_i - a_i \phi)^T (\tilde{\mathbf{x}}_i - a_i \phi) \\ &= \sum_{i=1}^q ||\tilde{\mathbf{x}}_i||^2 - \sum_{i=1}^q a_i^2 \quad (\text{note: } a_i \phi^T \tilde{\mathbf{x}}_i = a_i^2, \quad a_i \phi^T a_i \phi = a_i^2 \phi^T \phi = a_i^2) \\ &= \sum_{i=1}^q ||\tilde{\mathbf{x}}_i||^2 - \sum_{i=1}^q \phi^T \tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^T \phi = \sum_{i=1}^q ||\tilde{\mathbf{x}}_i||^2 - \phi^T \left( \sum_{i=1}^q \tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^T \right) \phi \end{aligned}$$

# PCA: Principal Component Analysis

- The sample covariance matrix of all training data or total scatter matrix:

$$\mathbf{S}^t = \frac{1}{q} \sum_{i=1}^q (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T = \frac{1}{q} \sum_{i=1}^q \tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^T$$

- To minimize  $\varepsilon^2 = \sum_{i=1}^q \|\tilde{\mathbf{x}}_i\|^2 - q\phi^T \mathbf{S}^t \phi$   
is to maximize  $\phi^T \mathbf{S}^t \phi$  with the constraint of  $\|\phi\|^2 = \phi^T \phi = 1$
- Use Lagrange optimization method

$$f(\phi, \lambda) = \phi^T \mathbf{S}^t \phi - \lambda(\phi^T \phi - 1) \Rightarrow \max i \text{ mum}$$
$$\frac{\partial f}{\partial \phi} = 2\mathbf{S}^t \phi - 2\lambda \phi = 0$$
$$\mathbf{S}^t \phi = \lambda \phi$$

- The solution is eigenvalue and eigenvector of matrix  $\mathbf{S}^t$

# PCA: Principal Component Analysis

- We see that if  $\phi$  is the eigenvector of the covariance matrix  $\mathbf{S}^t$ ,  $\phi^T \mathbf{S}^t \phi$  is maximum and the reconstruction error is minimum.
- The variance of the data projected onto the dimension spanned by the eigenvector is maximum.

$$\phi^T \mathbf{S}^t \phi = \frac{1}{q} \sum_{i=1}^q \phi^T (\mathbf{x}_i - \boldsymbol{\mu}) [\phi^T (\mathbf{x}_i - \boldsymbol{\mu})]^T = \frac{1}{q} \sum_{i=1}^q a_i^2$$

- Eigenvalue of a covariance matrix is the variance of the data projected onto the eigenvector.

$$\because \mathbf{S}^t \phi = \lambda \phi \quad \therefore \phi^T \mathbf{S}^t \phi = \phi^T \lambda \phi = \lambda \phi^T \phi = \lambda$$

# PCA: Principal Component Analysis

- Reducing the  $n$ -dimensional data  $\mathbf{x}_i$  into lower  $m$ -dimensional data  $\mathbf{y}_i$  by

$$\mathbf{y}_i = \Phi^T(\mathbf{x}_i - \boldsymbol{\mu})$$

where  $\Phi = [\phi_1 \ \phi_2 \dots \ \phi_m]$ ,  $m < n$

consists of  $m$  eigenvectors corresponding to the  $m$  largest eigen values of the total scatter matrix  $\mathbf{S}^t$ .

- We can reconstruct the original  $n$ -dimensional data  $\mathbf{x}_i$  using the lower  $m$ -dimensional data  $\mathbf{y}_i$  using  $\hat{\mathbf{x}}_i = \Phi\mathbf{y}_i + \boldsymbol{\mu}$
- The reconstruction error is minimum.

$$\varepsilon^2 = \sum_{i=1}^q ||\mathbf{x}_i - \hat{\mathbf{x}}_i||^2 \Rightarrow \text{minimum}$$

# PCA: Principal Component Analysis

Therefore, the famous principal component analysis PCA transforms the  $n$ -dimensional  $\mathbf{x}$  into  $m$ -dimensional  $\mathbf{y}$  by:

$$\mathbf{y} = \Phi^T(\mathbf{x} - \boldsymbol{\mu})$$

where  $\Phi = [\phi_1 \ \phi_2 \ \dots \ \phi_m]$ ,  $m < n$

consists of  $m$  eigenvectors corresponding to the  $m$  largest eigenvalues of the total scatter matrix  $\mathbf{S}^t$ .

The reconstruction error is:  $\Delta = \mathbf{x} - \hat{\mathbf{x}} = \mathbf{x} - \Phi\mathbf{y} - \boldsymbol{\mu}$

and the mean squared reconstruction error

$$E[||\Delta||^2] = E[\Delta^T \Delta] = \sum_{k=m+1}^n \lambda_k$$

where  $\lambda_k$  are eigenvalues sorted in descending order.

# PCA: Principal Component Analysis

- Note that:  $\mathbf{S}^t = \frac{1}{q} \sum_{i=1}^q (\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^T = \frac{1}{q} \sum_{i=1}^q \tilde{\mathbf{x}}_i \tilde{\mathbf{x}}_i^T = \frac{1}{q} \mathbf{X} \mathbf{X}^T$
- The rank of the total scatter matrix  $\mathbf{S}^t$  is  $\min(q-1, n)$  at most. Therefore, it has  $q-1$  nonzero eigenvalues at most.
- Thus, PCA with  $q-1$  dimensional  $\mathbf{y}$  will fully represent the original  $n$ -dimensional data  $\mathbf{x}$ ,  $q-1 \leq n$ , because the reconstruction error is zero.

$$E[||\Delta||^2] = \sum_{k=q}^n \lambda_k = 0$$

- Obviously, the scatter matrix or covariance matrix is a  $n \times n$  symmetry matrix.

$$\mathbf{S}^t = (\mathbf{S}^t)^T$$

# PCA: Principal Component Analysis

- Eigenvalue and eigenvector of a  $n \times n$  matrix  $\Sigma$  is defined mathematically by:

$$\Sigma \phi_i = \lambda_i \phi_i, \quad i = 1, 2, \dots, n$$

- If  $\Sigma$  is a symmetry matrix, eigenvectors corresponding to the distinct eigenvalues  $\lambda_1, \lambda_2, \dots, \lambda_n$  are orthogonal. Take the unit length of  $\phi_1, \phi_2, \dots, \phi_n$

$$\phi_i^T \phi_j = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

- Prove it ! (orthogonal+unit length is called orthonormal)

# PCA: Principal Component Analysis

- Let  $\Phi$  be the orthonormal matrix formed by the eigenvectors

$$\Phi = [\phi_1 \ \phi_2 \ \dots \ \phi_n]$$

Obviously:  $\Phi^T \Phi = I$        $\therefore \Phi^{-1} = \Phi^T$ ,       $\therefore \Phi \Phi^T = I$

Let  $\Lambda$  be a diagonal matrix:

$$\Lambda = diag(\lambda_1 \ \lambda_2 \ \dots \ \lambda_n) = \begin{bmatrix} \lambda_1 & & & 0 \\ & \ddots & & \\ & & \lambda_2 & \\ 0 & & & \lambda_n \end{bmatrix}$$

From

$$\Sigma \varphi_i = \lambda_i \varphi_i, \quad i = 1, 2, \dots, n$$

- We have

$$\Sigma \Phi = \Phi \Lambda \quad \therefore \Sigma = \Phi \Lambda \Phi^T, \text{ or } \Lambda = \Phi^T \Sigma \Phi$$

# PCA: Principal Component Analysis

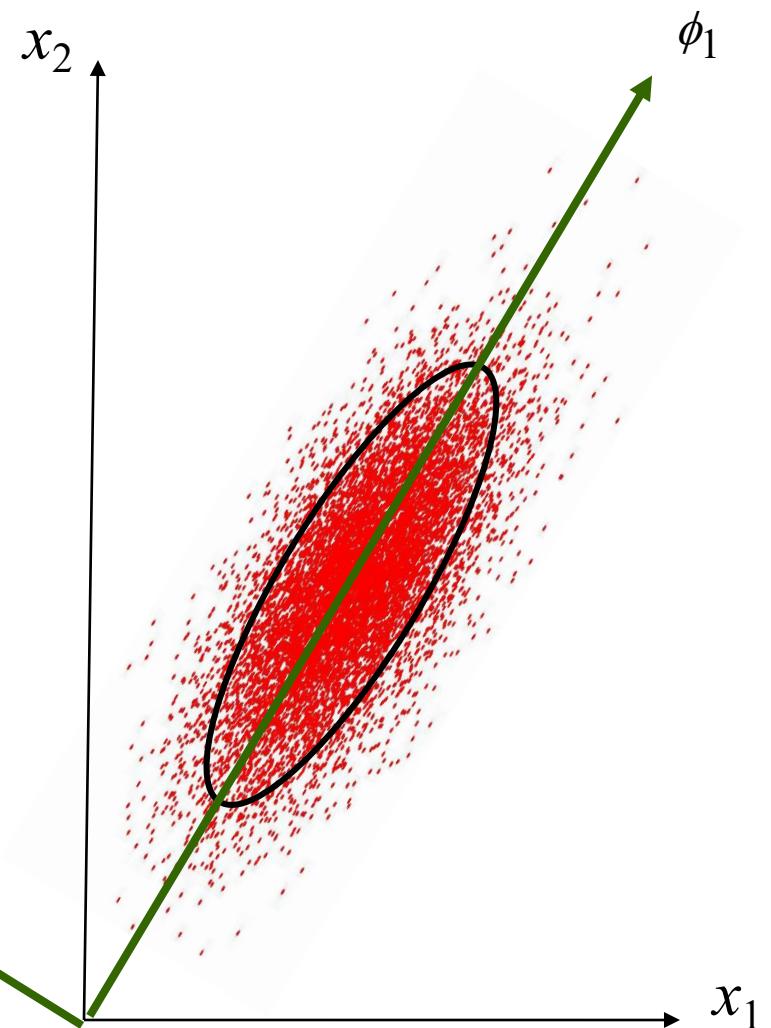
$$\Sigma_x = \begin{bmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \end{bmatrix}$$

$$= [\phi_1 \quad \phi_2] \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} [\phi_1 \quad \phi_2]^T$$
$$= \Phi \Lambda \Phi^T$$

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} = \Phi^T \Sigma_x \Phi$$

$$= \Phi^T \mathbf{X} \mathbf{X}^T \Phi = \Phi^T \mathbf{X} (\Phi^T \mathbf{X})^T = \mathbf{Y} \mathbf{Y}^T = \Sigma_y = \Lambda$$

where:  $\mathbf{Y} = \Phi^T \mathbf{X} = \mathbf{W}^T \mathbf{X}$

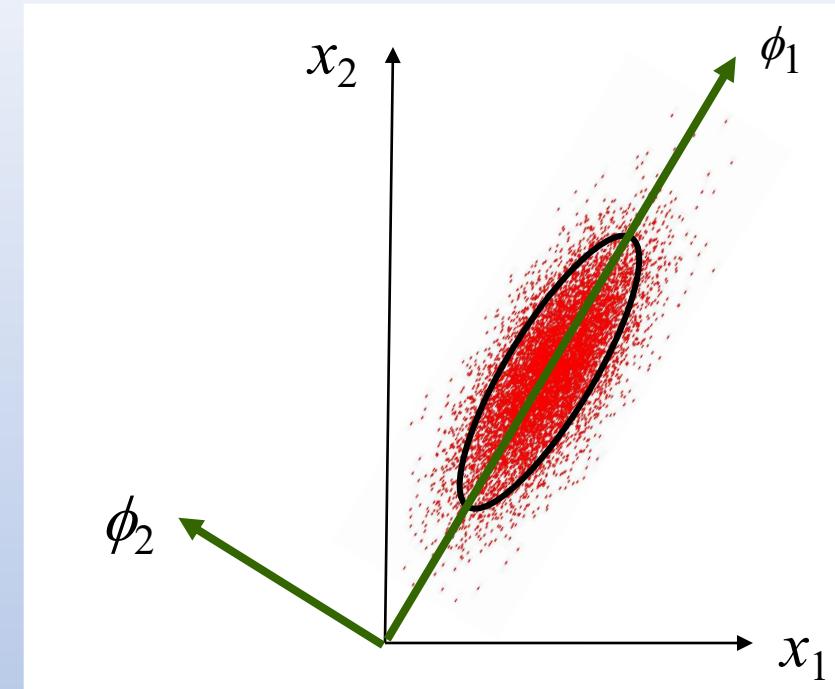


# PCA: Principal Component Analysis

PCA:  $\max t \text{trace}[\Phi^T S^t \Phi] = \sum_{k=1}^m \lambda_k^t$   
 $\Phi$ :  $[n \times m]$ , e.g.  $=[60000 \times 80]$

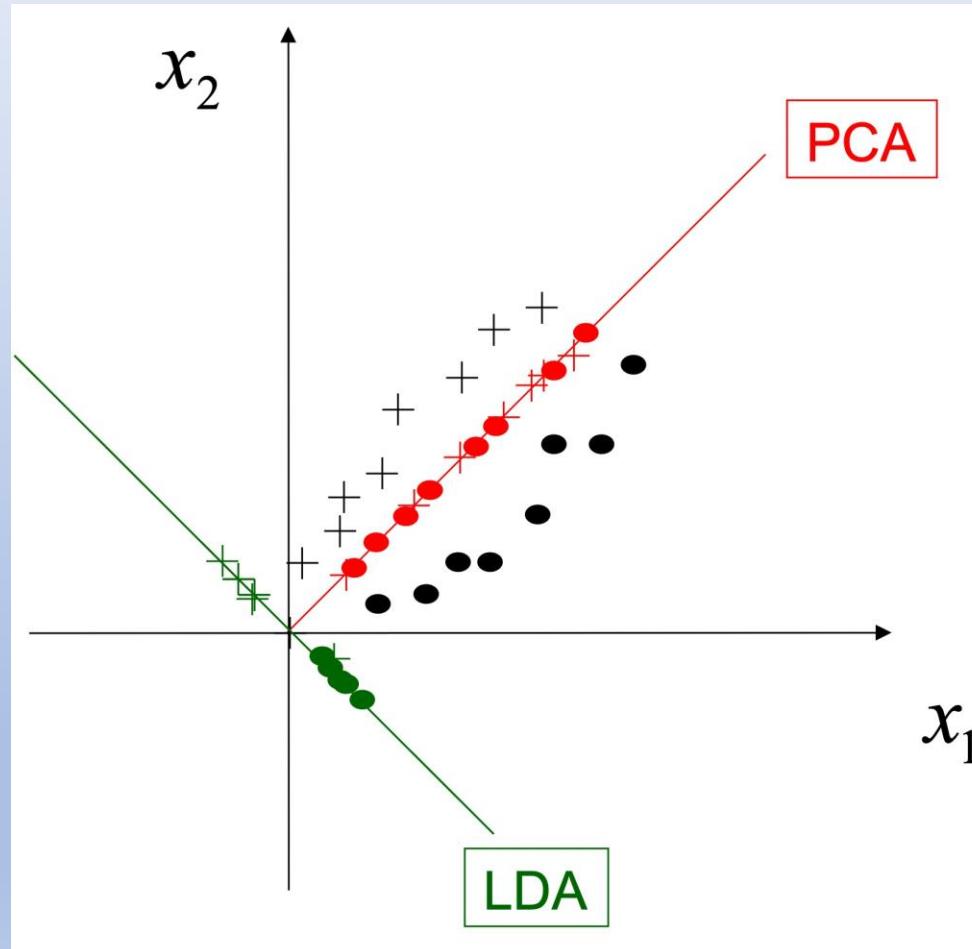
$$\mathbf{y} = \Phi^T (\mathbf{x} - \boldsymbol{\mu})$$

- PCA is an unsupervised method that minimizes the reconstruction error or maximize the variation (information carried by the data) in the reduced sub-space. This dimension reduction will reduce the computational complexity of the subsequent processing.
- However, any information lost may reduces the accuracy of the subsequent processing. Further more, the lost information, though less important for data representation, could be critical for discriminating the data class membership.



# LDA: Linear Discriminant Analysis

- Problem of PCA in classification?



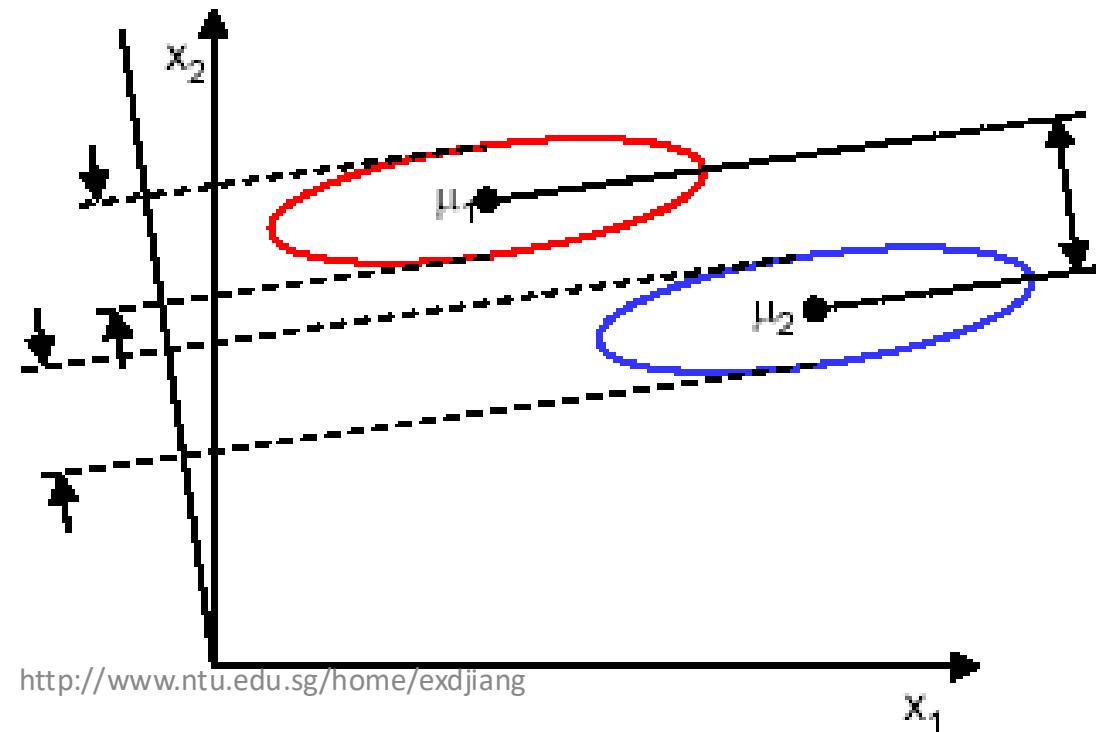
# LDA: Linear Discriminant Analysis

Desired? properties to determine the projection vector for classification

Maximize separation between projected class means

Minimize projected within-class scatter (variance)

$$\mathbf{y} = \Phi^T(\mathbf{x} - \boldsymbol{\mu})$$



# LDA: Linear Discriminant Analysis

- Given  $q$   $n$ -dimensional training samples of  $c$  classes

$$\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_q$$

- The number of training samples of class  $\omega_j$  is  $q_j$ ,  $j = 1, 2, \dots, c$ .
- The covariance matrix of class  $\omega_j$  is computed as

$$\Sigma_j = \frac{1}{q_j} \sum_{X_i \in \omega_j} (\mathbf{x}_i - \boldsymbol{\mu}_j)(\mathbf{x}_i - \boldsymbol{\mu}_j)^T, \quad \text{where } \boldsymbol{\mu}_j = \frac{1}{q_j} \sum_{X_i \in \omega_j} \mathbf{x}_i$$

- The within class scatter matrix of the  $c$  classes is defined as

$$\mathbf{S}^w = \sum_{j=1}^c \frac{q_j}{q} \Sigma_j$$

- The between class scatter matrix of  $c$  classes is defined as

$$\mathbf{S}^b = \sum_{j=1}^c \frac{q_j}{q} (\boldsymbol{\mu}_j - \boldsymbol{\mu})(\boldsymbol{\mu}_j - \boldsymbol{\mu})^T, \text{ obviously: } \boldsymbol{\mu} = \frac{1}{q} \sum_{i=1}^q \mathbf{x}_i = \sum_{j=1}^c \frac{q_j}{q} \boldsymbol{\mu}_j$$

# LDA: Linear Discriminant Analysis

➤ Instead of     $PCA: \max trace[\Phi^T S^t \Phi] = \sum_{k=1}^m \lambda_k^t$

$LDA: \min trace[\Phi^T S^w \Phi] \text{ & } \max trace[\Phi^T S^b \Phi]$

$\Rightarrow LDA: \max trace[\Phi^T S^{w-1} S^b \Phi] = \sum_{k=1}^m \lambda_k^{b/w}$

➤ Note that:

$trace(S^t) \Rightarrow$  total variation amount of all samples

$trace(S^w) \Rightarrow$  total variation amount of samples within classes

$trace(S^b) \Rightarrow$  total variation amount of samples between classes

➤ It is easy to prove:

$$S^t = S^w + S^b$$

# LDA: Linear Discriminant Analysis

- As pattern recognition is not to represent the original data, but to discriminate the class membership of the data.
- obviously, LDA extracts much more discriminative features than PCA.
- Therefore, the vast majority of researchers prefer LDA to PCA for feature extraction and dimensionality reduction for pattern recognition.

$$LDA: \max t \text{ trace}[\Phi^T \mathbf{S}^{w-1} \mathbf{S}^b \Phi] = \sum_{k=1}^m \lambda_k^{b/w}$$

# LDA: Linear Discriminant Analysis

$$LDA: \max t \operatorname{trace}[\Phi^T \mathbf{S}^{w^{-1}} \mathbf{S}^b \Phi] = \sum_{k=1}^m \lambda_k^{b/w}$$

- The rank of  $\mathbf{S}^w$  is  $\min(q-c, n)$  at most. For many applications  $q-c \ll n$ . So, we have **problem** because  $\mathbf{S}^w$  is singular and its inverse does not exist.
- **Thousands** of research papers proposed various LDA variants trying to solve this **problem**!
- One example is PCA + LDA or called Fisher Face.

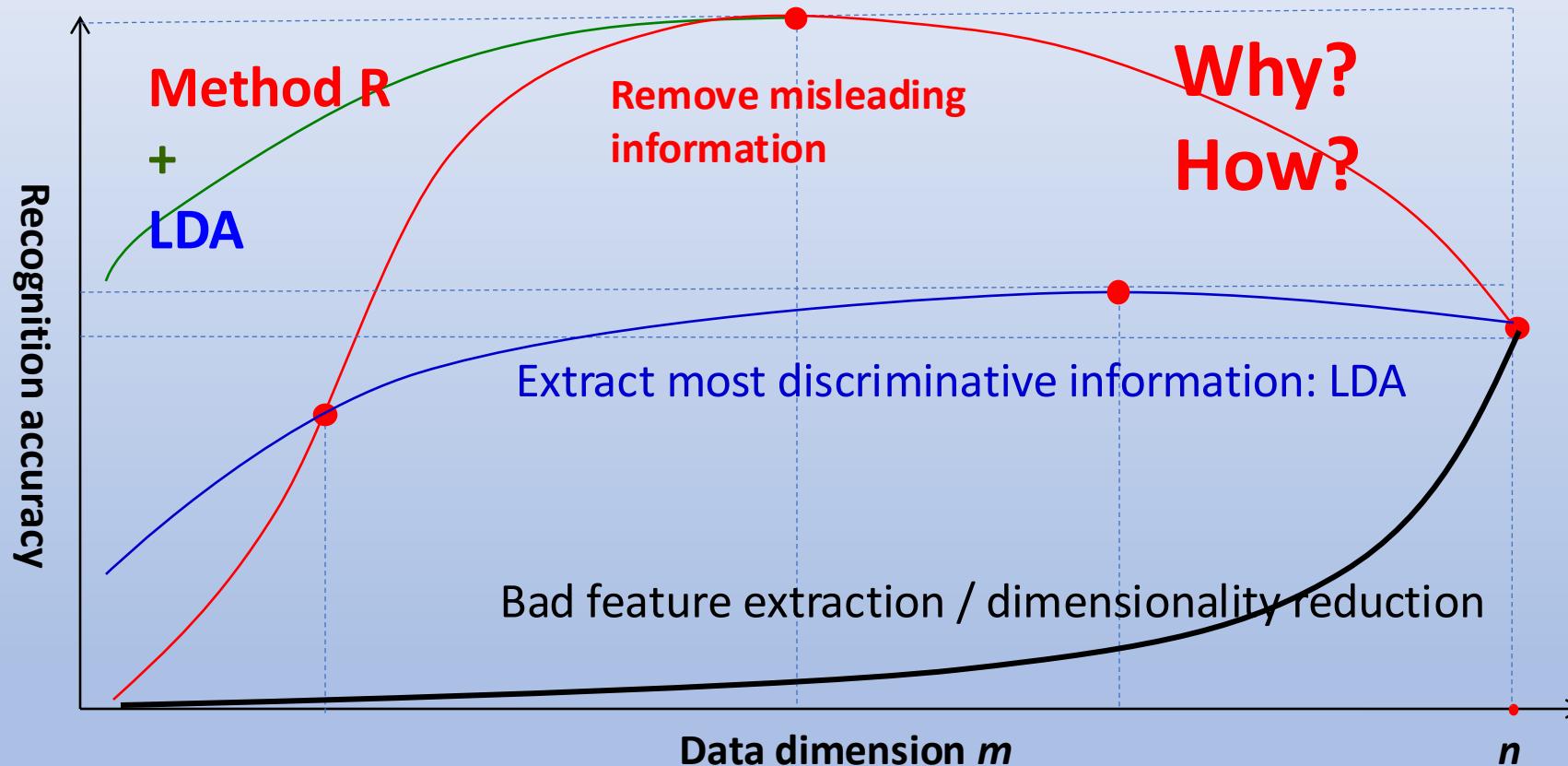
# LDA: Linear Discriminant Analysis

- Any dimensionality reduction loses information. Extracting the most discriminative information just means losing least information. Original (raw) data has the most information.
- If higher accuracy can be achieved with less discriminative information, **why the criterion of DR is set to extract the most discriminative information?**
- If more discriminative information leads to higher accuracy, **why we need dimensionality reduction, not directly use the raw data?**
- Superficial study will cause misunderstanding and hence insignificant (trivial) or even false research.
- You are encouraged to compare PCA and LDA in your project to verify your understanding.

Recognition accuracy decreases with decreasing dimensionality

Information = discriminative information + redundant information

➤ Minimizing the lost doesn't mean we can get any gain!



Where can the recognition accuracy gain come from?

**Information = reliable information + misleading information**

# PCA or LDA and their roles for pattern recognition

➤ The study these issues has been partly published in:

X.D. Jiang, “Linear Subspace Learning-Based Dimensionality Reduction,” *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 16-26, March 2011.

X.D. Jiang, B. Mandal and A. Kot, “Eigenfeature Regularization and Extraction in Face Recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 3, pp. 383-394, March 2008.

X.D. Jiang, “Asymmetric Principal Component and Discriminant Analyses for Pattern Classification,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 931-937, May 2009.

These issues are discussed courses EE7403: Image Analysis & Pattern Recognition.

# Classification in Subspace / Feature Space

- Using  $m$  eigenvectors corresponding to  $m$  largest eigenvalues of the  $n$  by  $n$  data scatter matrix, we can reduce the  $n$ -dimensionality data vector  $\mathbf{x}$  to  $m$ -dimensional feature vector  $\mathbf{f}$  by:

$$\mathbf{f} = \Phi^T \mathbf{x}$$

where  $\Phi$  is an  $n$  by  $m$  matrix containing  $m$  eigenvectors.



$$[300 \times 200] \Rightarrow \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{60000} \end{pmatrix} \Rightarrow \mathbf{f} = \begin{bmatrix} f \\ \vdots \\ f_{60} \end{bmatrix} = \Phi^T \mathbf{x}$$

Feature Scaling,  
Dimension Reduction  
Feature Extraction

# Classification in Subspace / Feature Space

- We can perform the classification in the  $m$ -dimensional space, called subspace or feature space spanned by  $\mathbf{f}$ :

$$\mathbf{f} = \Phi^T \mathbf{x}$$

$$d_{fMaj} = (\mathbf{f} - \boldsymbol{\mu}_{fj})^T \boldsymbol{\Sigma}_{fj}^{-1} (\mathbf{f} - \boldsymbol{\mu}_{fj})$$

$$\omega_k = \arg \min_j d_{fMaj} = \arg \min_j (\mathbf{f} - \boldsymbol{\mu}_{fj})^T \boldsymbol{\Sigma}_{fj}^{-1} (\mathbf{f} - \boldsymbol{\mu}_{fj})$$

- Although we can use  $\mathbf{f}$  of training data to compute  $\boldsymbol{\mu}_{fj}$  and  $\boldsymbol{\Sigma}_{fj}$ , we have:

$$\boldsymbol{\mu}_{fj} = \Phi^T \boldsymbol{\mu}_j \quad \boldsymbol{\Sigma}_{fj} = \Phi^T \boldsymbol{\Sigma}_j \Phi$$

# Classification in Subspace / Feature Space

- For face recognition, each person is a class. So we often have large number of classes and small number of training samples per class. Thus, we better use the pooled covariance matrix instead of class covariance matrices to compute the Mahanolobis distance.

$$\Sigma_f^w = \frac{1}{q} \sum_{j=1}^c q_j \Sigma_{fj}$$

$$d_{fMaj} = (\mathbf{f} - \boldsymbol{\mu}_{fj})^T \Sigma_f^{w-1} (\mathbf{f} - \boldsymbol{\mu}_{fj})$$

- Recognize the face by minimum Mahanolobis distance classifier:

$$\omega_k = \arg \min_j d_{fMaj} = \arg \min_j (\mathbf{f} - \boldsymbol{\mu}_{fj})^T \Sigma_f^{w-1} (\mathbf{f} - \boldsymbol{\mu}_{fj})$$

# **Topic 10**

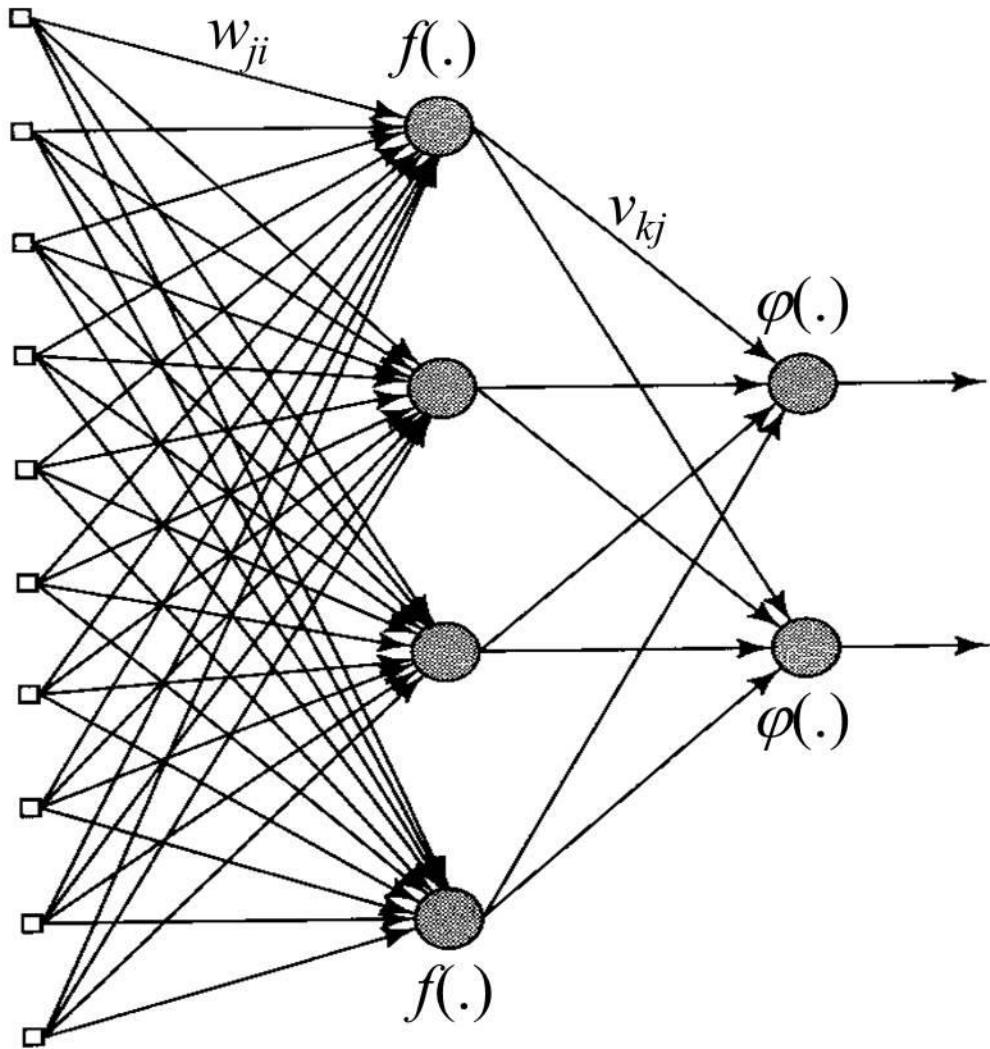
## **Neural Networks and Deep Machine Learning: from MLP to CNN**

# Neural Networks and Deep CNN

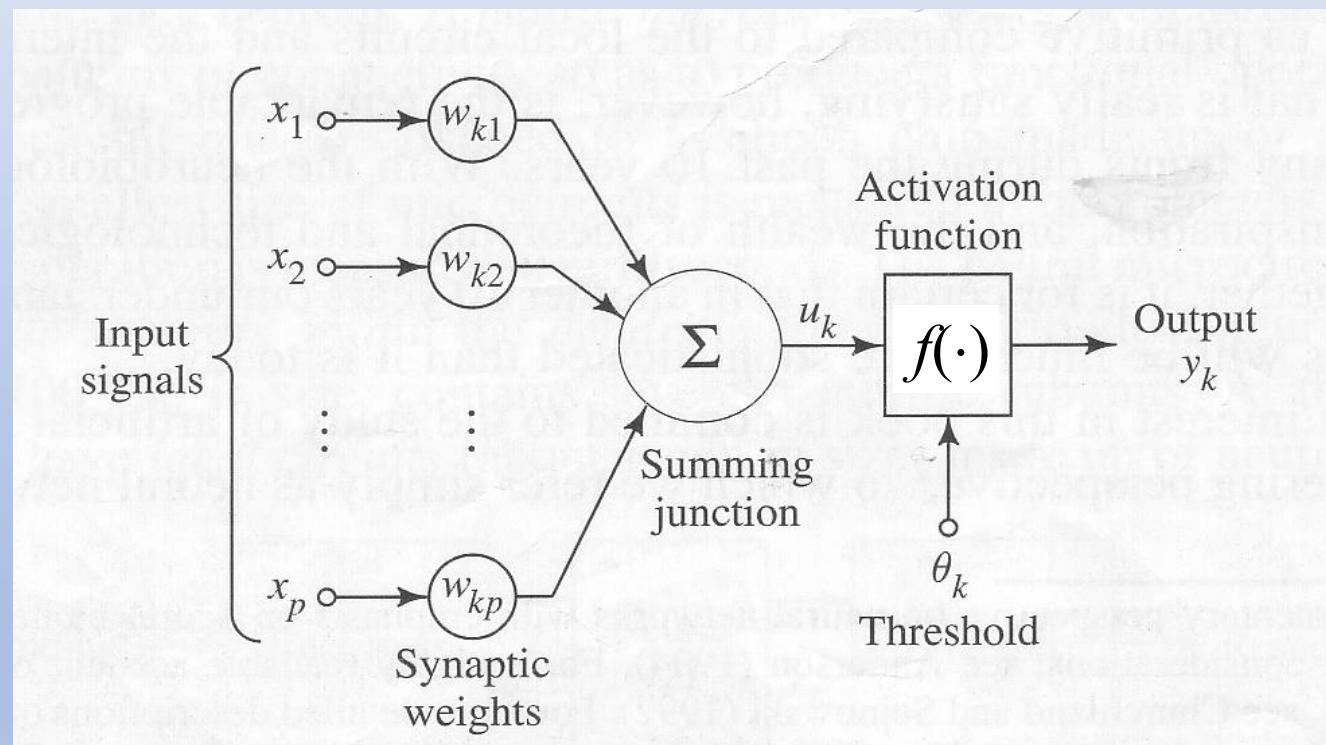
## Outline

- Network Architecture and Neuron Model
- Optimization and Iterative (Error Correction) Learning
- Multilayer Perceptron (MLP) and Backpropagation Learning Algorithm
- Understanding Deep Learning and Convolutional neural Networks (CNN)
- **Deep Learning: from CNN to Transformer**

# Neural Networks and Deep CNN



Network architecture of artificial neural networks (ANN), or simply, neural networks (NN)



Neuron Model.

# Neural Networks and Deep CNN -- Neuron Model

In mathematical terms, we can describe the neuron as:

$$u_k = \sum_{j=1}^p w_{kj}x_j \quad y_k = f(u_k - \theta_k)$$

Where  $x_1, x_2, \dots, x_p$  are the input signals,  $w_{k1}, w_{k2}, \dots, w_{kp}$  are the synaptic weights of neuron  $k$ ,  $u_k$  is the linear combiner output,  $\theta_k$  is the threshold,  $f(\cdot)$  is the activation function and  $y_k$  is the neuron's output.

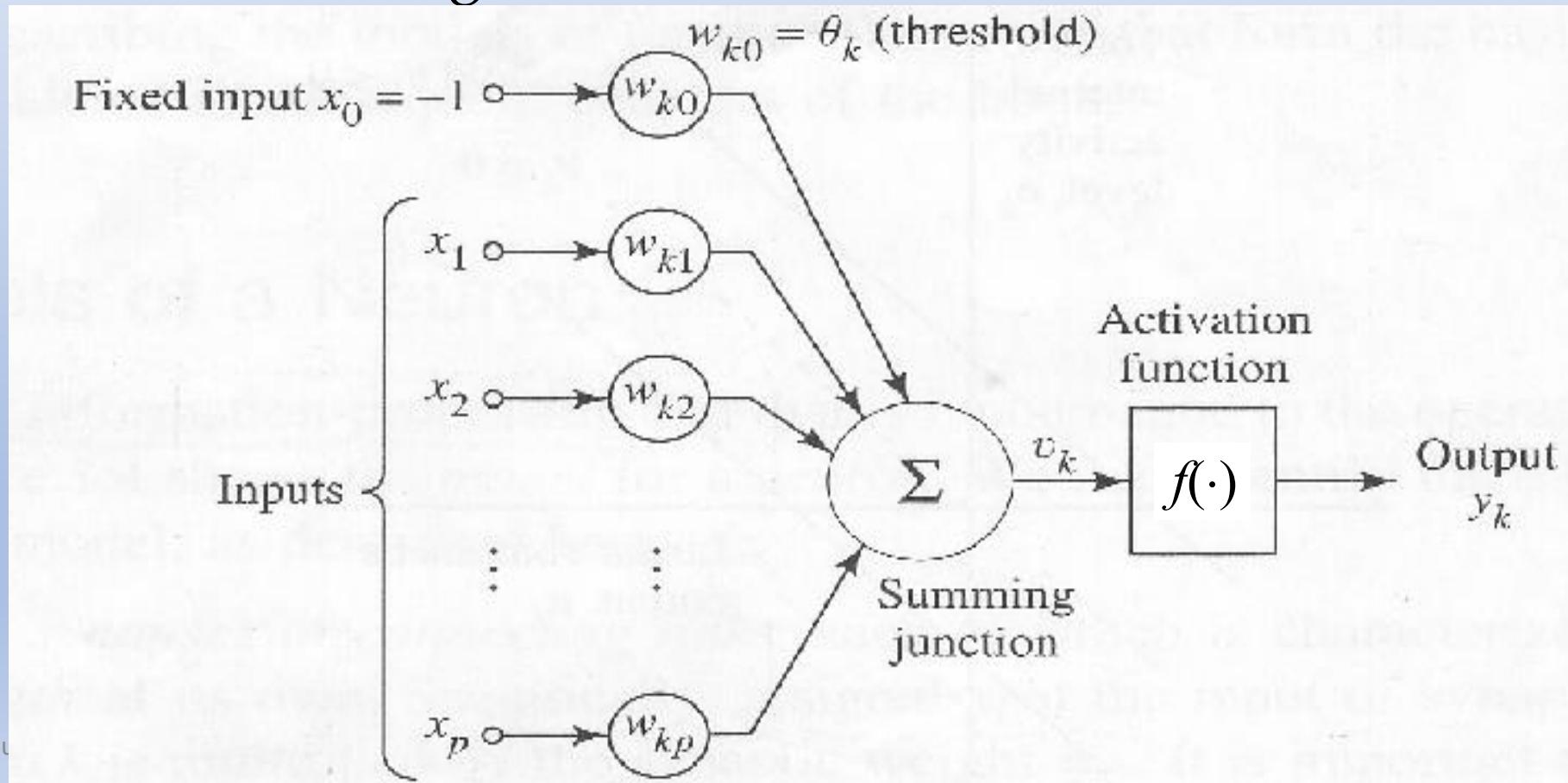
$\theta_k$  is an external parameter, we can consider this parameter as an input variable:

Then we have:  $x_0 = 1, \quad w_{k0} = -\theta_k$

# Neural Networks and Deep CNN -- Neuron Model

$$v_k = \sum_{j=0}^p w_{kj} x_j \quad y_k = f(v_k)$$

Thus, the diagram of the model becomes:



# Neural Networks and Deep CNN -- Activation Functions

## Types of Activation Functions

Binary Function:

$$f(v) = \begin{cases} 1 & v \geq 0 \\ 0 & v < 0 \end{cases}$$

Piecewise-Linear Function

$$f(v) = \begin{cases} 1 & v \geq 0.5 \\ v & -0.5 < v < 0.5 \\ 0 & v \leq -0.5 \end{cases}$$

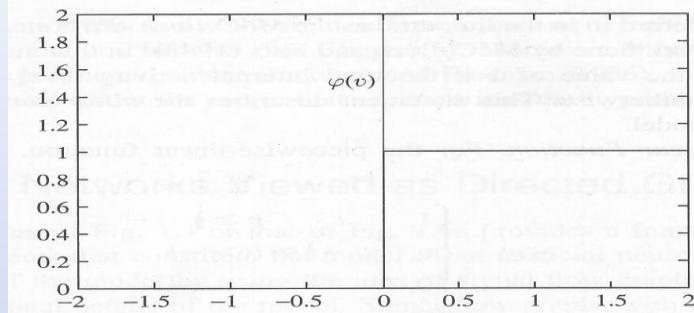
Sigmoid Function

$$f(v) = \frac{1}{1 + \exp(-av)}$$

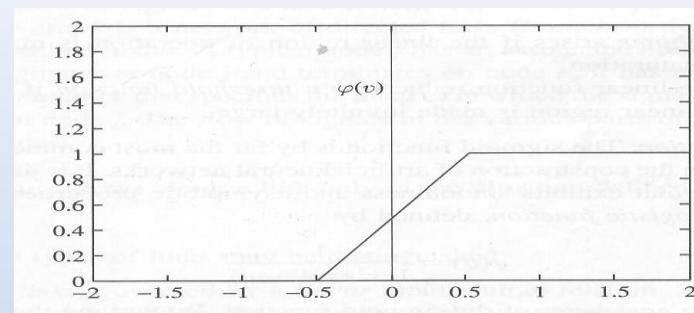
ReLU Function

$$f(v) = \max(0, v) = \begin{cases} v, & \text{for } v \geq 0 \\ 0, & \text{for } v < 0 \end{cases}$$

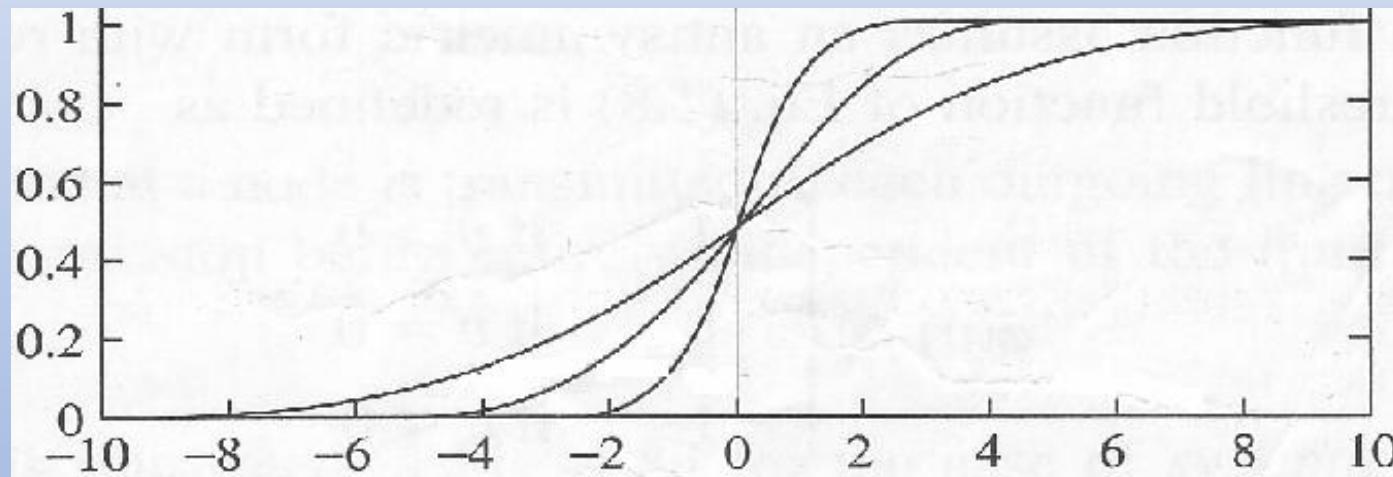
# Neural Networks and Deep CNN -- Activation Functions



binary

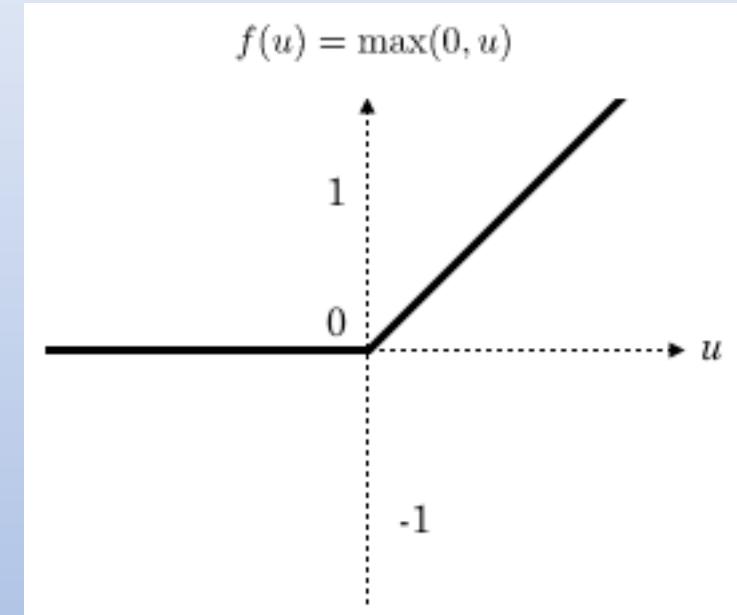


Piecewise linear



sigmoid

$$f(v) = \frac{1}{1 + \exp(-av)}$$



ReLU

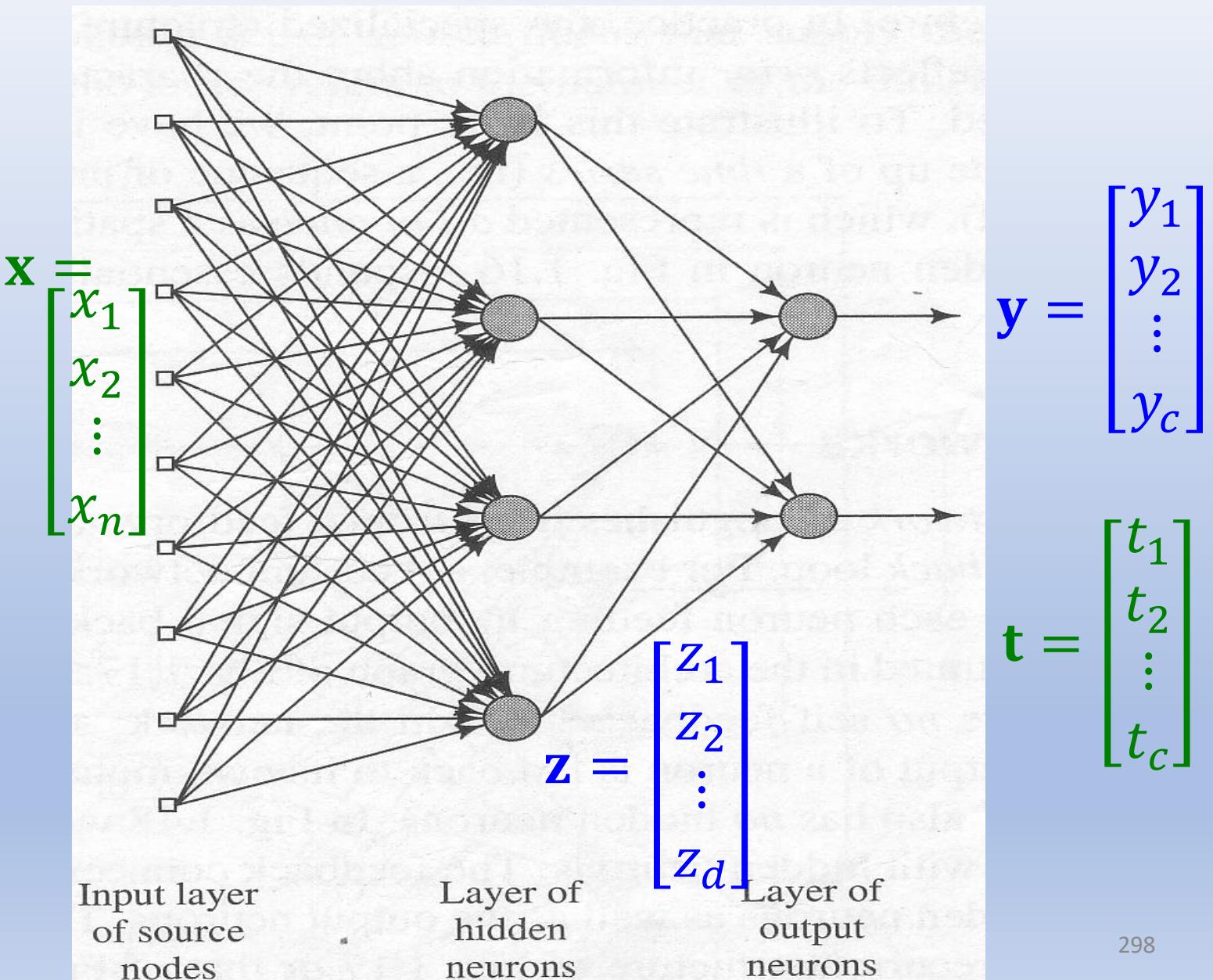
$$f(v) = \max(0, v) = \begin{cases} v, & \text{for } v \geq 0 \\ 0, & \text{for } v < 0 \end{cases}$$

# ANN – Multilayer Perceptron (MLP)

$$\mathbf{w}_j = \begin{bmatrix} w_{j1} \\ w_{j2} \\ \vdots \\ w_{jn} \end{bmatrix}, \mathbf{v}_k = \begin{bmatrix} v_{k1} \\ v_{k2} \\ \vdots \\ v_{kd} \end{bmatrix}$$

$$\mathbf{W} = [\mathbf{w}_1 \quad \mathbf{w}_2 \quad \cdots \quad \mathbf{w}_d]$$

$$\mathbf{V} = [\mathbf{v}_1 \quad \mathbf{v}_2 \quad \cdots \quad \mathbf{v}_c]$$



## ANN – Multilayer Perceptron (MLP)

Mathematical expression of outputs from inputs  
of neural networks in scalar, vector and matrix forms

$$z_j = f \left( \sum_{i=1}^n w_{ji} x_i \right) = f(\mathbf{w}_j^T \mathbf{x})$$

$$y_k = f \left( \sum_{j=1}^d v_{kj} z_j \right) = f(\mathbf{v}_k^T \mathbf{z}) = f \left[ \sum_{j=1}^d v_{kj} f \left( \sum_{i=1}^n w_{ji} x_i \right) \right]$$

↓

$$\mathbf{y} = f(\mathbf{V}^T \mathbf{z}) = f[\mathbf{V}^T f(\mathbf{W}^T \mathbf{x})]$$

## ANN – Multilayer Perceptron

- If the activation function  $f$  is a linear function,

$$f(r) = r$$

$\Downarrow$

$$\mathbf{y} = f[\mathbf{V}^T f(\mathbf{W}^T \mathbf{x})] = \mathbf{V}^T \mathbf{W}^T \mathbf{x} = (\mathbf{W}\mathbf{V})^T \mathbf{x} = \mathbf{U}^T \mathbf{x}$$

where  $\mathbf{U} = \mathbf{W}\mathbf{V}$

- Multilayer network will be just the same as a single layer network.
- To design a multi-layer neural network, it is thus very important that the activation function should be a nonlinear function, at least for hidden neurons.

# ANN – Single Layer Neural Network

- Let's first study the single layer network of single output

$$y = \mathbf{w}^T \mathbf{x}$$

- If we have a target output  $t$ , the error of the output is:

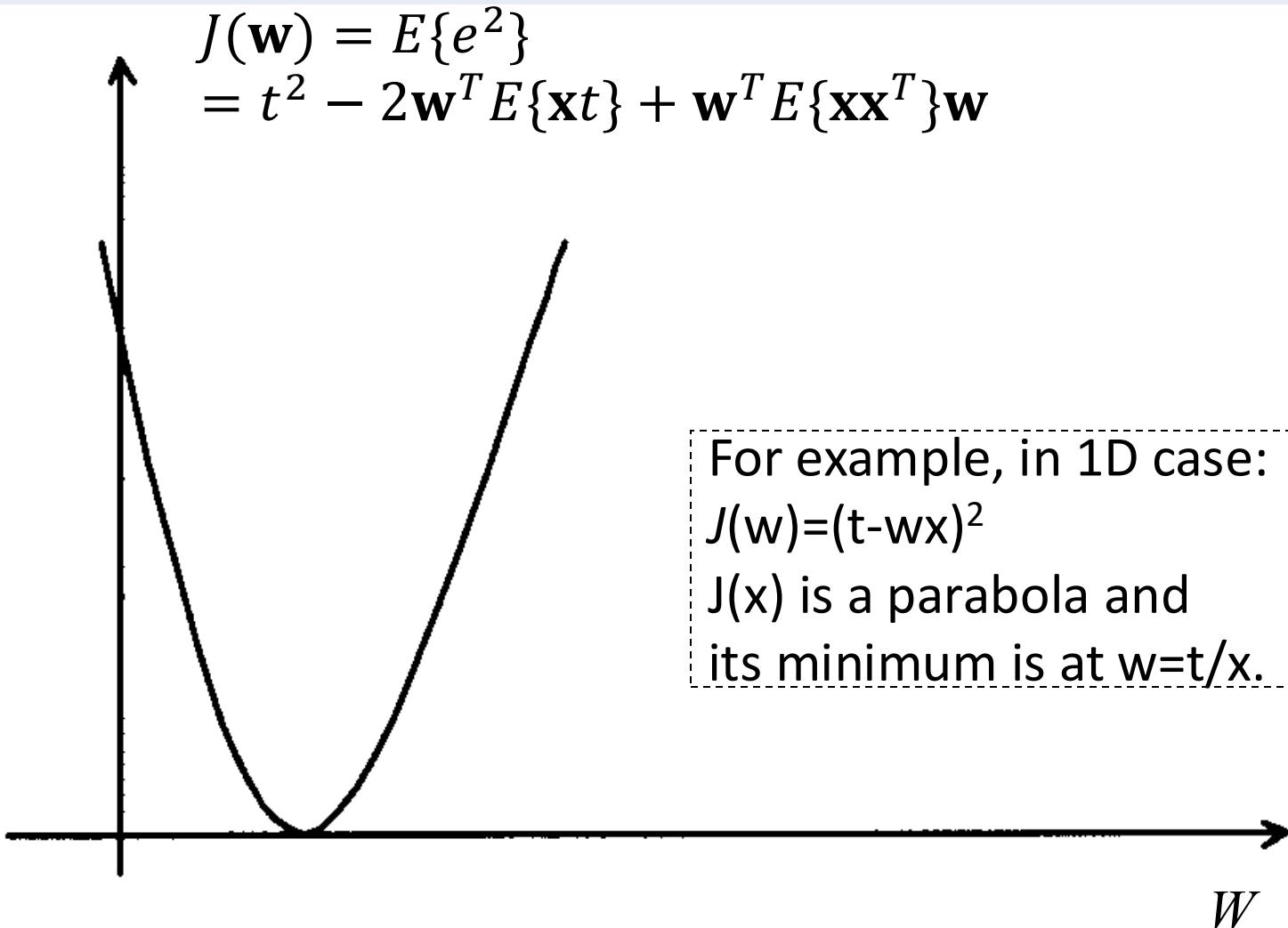
$$e = t - y = t - \mathbf{w}^T \mathbf{x}$$

- The mean square error is:

$$\begin{aligned} J(\mathbf{w}) &= E\{e^2\} = E\{(t - y)^2\} = E\{(t - \mathbf{w}^T \mathbf{x})^2\} \\ &= t^2 - 2\mathbf{w}^T E\{\mathbf{x}t\} + \mathbf{w}^T E\{\mathbf{x}\mathbf{x}^T\}\mathbf{w} \end{aligned}$$

- Obviously, the error function is a quadratic function (second order) of  $\mathbf{w}$ . It has only one minimum point.

# ANN – Single Layer Neural Network



## ANN – Single Layer Neural Network

- The optimal network parameter is obtained by minimizing the mean square error. This leads to the least mean square solution of  $\mathbf{w}$ :

$$\frac{\partial E\{e^2\}}{\partial \mathbf{w}} = \frac{\partial(t^2 - 2\mathbf{w}^T E\{\mathbf{x}\mathbf{t}\} + \mathbf{w}^T E\{\mathbf{x}\mathbf{x}^T\}\mathbf{w})}{\partial \mathbf{w}} = 0$$

↓

$$E\{\mathbf{x}\mathbf{x}^T\}\mathbf{w} = E\{\mathbf{x}\mathbf{t}\}$$

- Suppose we have  $q$  samples  $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_L$ ,

$$E\{\mathbf{x}\mathbf{x}^T\} \cong \frac{1}{q} \sum_{i=1}^q \mathbf{x}_i \mathbf{x}_i^T, \quad E\{\mathbf{x}\mathbf{t}\} \cong \frac{1}{q} \sum_{i=1}^q \mathbf{x}_i t_i$$

# ANN – Single Layer Neural Network

- Let  $\mathbf{X} = [\mathbf{x}_1 \quad \mathbf{x}_2 \quad \cdots \quad \mathbf{x}_q], \quad \mathbf{t} = [t_1 \quad t_2 \quad \cdots \quad t_q]$

- Then:  $q \cdot E\{\mathbf{xx}^T\} \cong \sum_{i=1}^q \mathbf{x}_i \mathbf{x}_i^T = \mathbf{XX}^T$

$$q \cdot E\{\mathbf{xt}\} \cong \sum_{i=1}^q \mathbf{x}_i t_i = \mathbf{Xt}^T$$

$$\begin{aligned} E\{\mathbf{xx}^T\}\mathbf{w} &= E\{\mathbf{xt}\} \\ &\approx \Downarrow \\ \mathbf{XX}^T \mathbf{w} &= \mathbf{Xt}^T \quad \Rightarrow \quad \mathbf{w} = (\mathbf{XX}^T)^{-1} \mathbf{Xt}^T \end{aligned}$$

- It is also called least square method.

# ANN – Single Layer Neural Network

- We can also obtain the optimal weights  $\mathbf{w}$  by iteratively adjust/update the value of  $\mathbf{w}$  using gradient descent method:

$$\mathbf{w} \leftarrow \mathbf{w} + \Delta\mathbf{w}$$

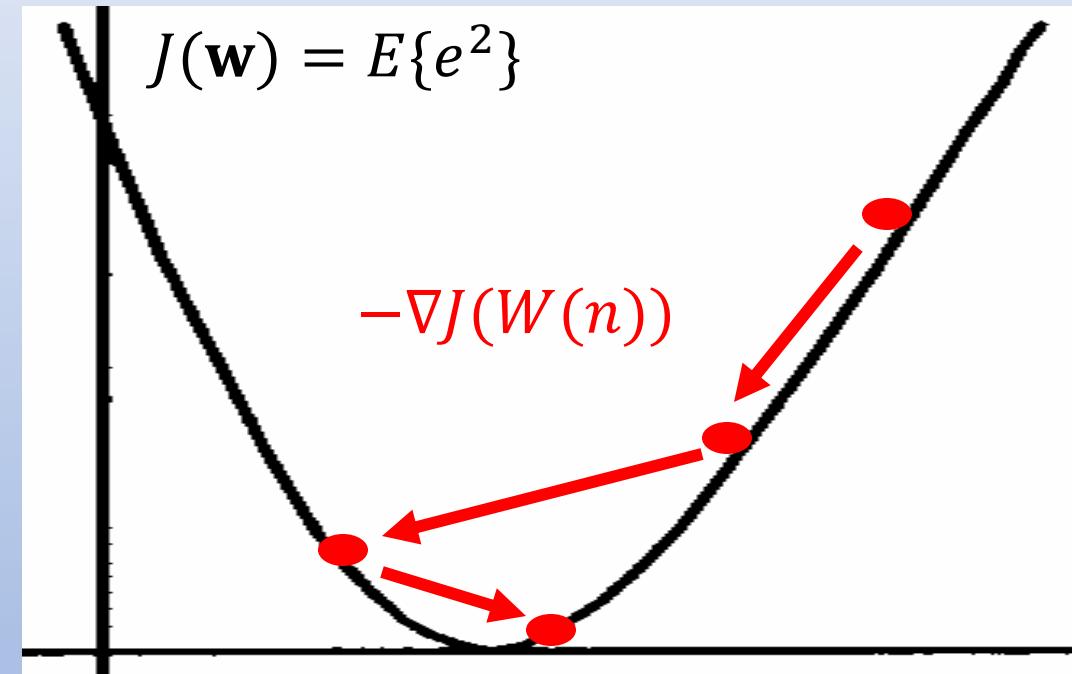
$$\Delta\mathbf{w} = -\eta \nabla J(\mathbf{w})$$

$$\nabla J(\mathbf{w}) = \frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}$$

$$= \frac{\partial E\{(t - \mathbf{w}^T \mathbf{x})^2\}}{\partial \mathbf{w}}$$

$$= -2E\{(t - \mathbf{w}^T \mathbf{x})\mathbf{x}\}$$

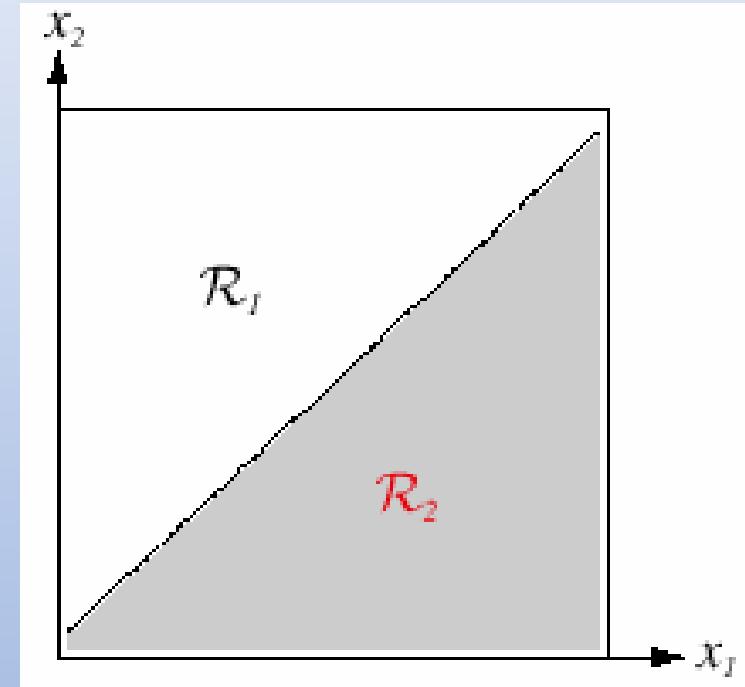
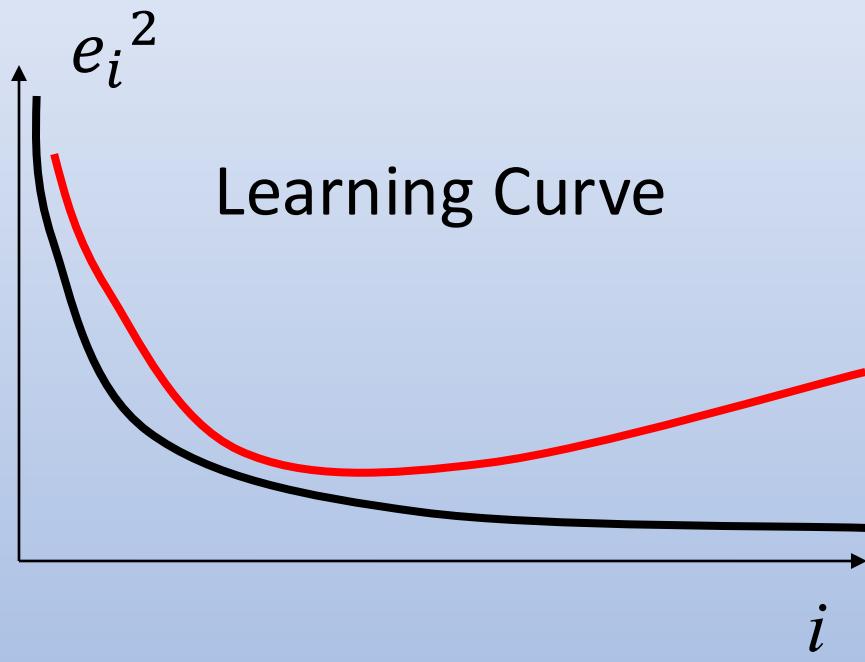
$$= -2E\{e\mathbf{x}\} \cong -2e_i \mathbf{x}_i$$



$$\begin{aligned}\mathbf{w} &\leftarrow \mathbf{w} + \eta e_i \mathbf{x}_i = \mathbf{w} + \eta(t_i - y_i)\mathbf{x}_i \\ \rightarrow \mathbf{w} &= (\mathbf{X}\mathbf{X}^T)^{-1}\mathbf{X}\mathbf{t}^T\end{aligned}$$

# ANN – Learning Curve and Decision Boundary

$$\mathbf{w} \leftarrow \mathbf{w} + \eta(t_i - y_i)\mathbf{x}_i$$



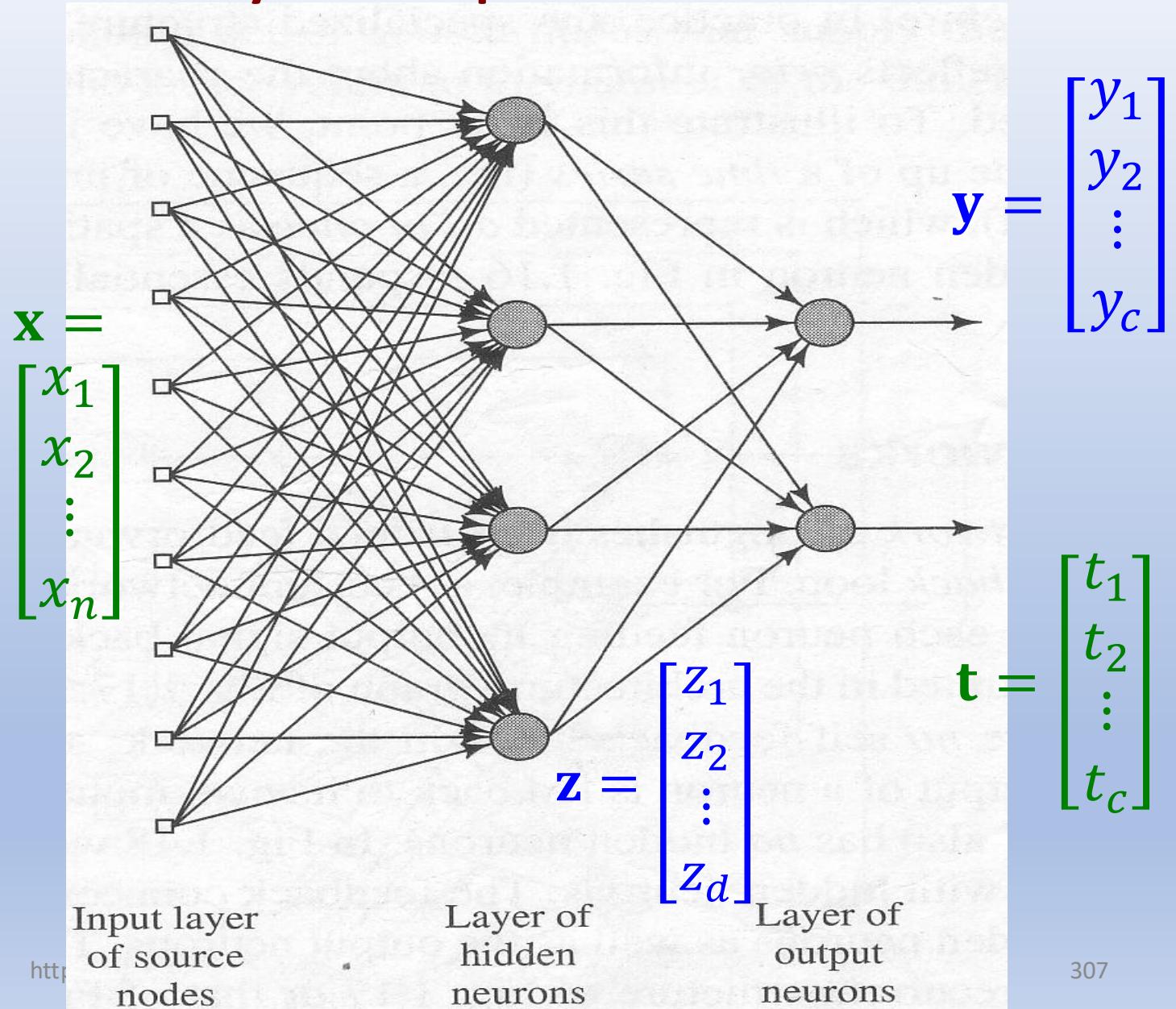
Decision boundary is a strait line or plane or hyper-plane.

$$\mathbf{w}^T \mathbf{x} = 0$$

# ANN – Multilayer Perceptron

$$\mathbf{y} = f(\mathbf{V}^T \mathbf{z}) \\ = f[\mathbf{V}^T f(\mathbf{W}^T \mathbf{x})]$$

Any decision can be implemented by a two-layer network.  
Any function from input to output can be implemented in a two-layer net, given sufficient number of hidden units, proper nonlinearities, and weights.



# ANN – Multilayer Perceptron

- These results are of greater theoretical interest than practical, since the construction of such a network requires sufficient number of hidden units and the weight values are to be learnt!
- How to find the nonlinear function based on data is the central problem in network-based pattern recognition.
- Our goal now is to set the interconnection weights based on the training patterns and the desired outputs.
- It is a straightforward matter to understand how the output, and thus the error, depend on the hidden-to-output layer weights.
- The power of **backpropagation** is that it enables us to compute an effective error for each hidden unit, and thus derive a learning rule for the weights.

## ANN – Multilayer Perceptron/backpropagation

$$\mathbf{y} = f(\mathbf{V}^T \mathbf{z}) = f[\mathbf{V}^T f(\mathbf{W}^T \mathbf{x})]$$

$$J(\mathbf{w}, \mathbf{v}) = E\{e^2\} = E\{\|\mathbf{t} - \mathbf{y}\|^2\} = E\{\|\mathbf{t} - f[\mathbf{V}^T f(\mathbf{W}^T \mathbf{x})]\|^2\}$$

To minimize  $J(\mathbf{w}, \mathbf{v})$ , let  $\nabla J(\mathbf{w}, \mathbf{v}) = 0$

There is no analytical solution to this equation due to the nonlinear function  $f$ .

However, we can use gradient decent method so long as the gradient is computable for a given sample  $\mathbf{x}_i$ .

$$\begin{aligned}\mathbf{w} &\leftarrow \mathbf{w} - \eta \nabla J(\mathbf{w}) \\ \mathbf{v} &\leftarrow \mathbf{v} - \eta \nabla J(\mathbf{v})\end{aligned}$$

## ANN – Multilayer Perceptron/backpropagation

- We use scalar form of weights to derive the learning formula.

$$J(\mathbf{w}, \mathbf{v}) = E\{e^2\} = E\{\|\mathbf{t} - \mathbf{y}\|^2\} = \frac{1}{2} \sum_{k=1}^c (t_k - y_k)^2$$

$$= \frac{1}{2} \sum_{k=1}^c [t_k - f(q_k)]^2 = \frac{1}{2} \sum_{k=1}^c \left[ t_k - f\left(\sum_{j=1}^d v_{kj} z_j\right) \right]^2$$

$$\frac{\partial J(\mathbf{w}, \mathbf{v})}{\partial v_{kj}} = \frac{\partial J}{\partial q_k} \cdot \frac{\partial q_k}{\partial v_{kj}} = -(t_k - y_k) f'(q_k) z_j$$

## ANN – Multilayer Perceptron/backpropagation

$$J(\mathbf{w}, \mathbf{v}) = \frac{1}{2} \sum_{k=1}^c \left[ t_k - f \left( \sum_{j=1}^d v_{kj} z_j \right) \right]^2 = \frac{1}{2} \sum_{k=1}^c \left[ t_k - f \left[ \sum_{j=1}^d v_{kj} f(q_j) \right] \right]^2$$
$$= \frac{1}{2} \sum_{k=1}^c \left[ t_k - f \left[ \sum_{j=1}^d v_{kj} f \left( \sum_{i=1}^n w_{ji} x_i \right) \right] \right]^2$$

$$\frac{\partial J(\mathbf{w}, \mathbf{v})}{\partial w_{ji}} = \frac{\partial J}{\partial z_j} \cdot \frac{\partial z_j}{\partial q_j} \cdot \frac{\partial q_j}{\partial w_{ji}} = - \left[ \sum_{k=1}^c (t_k - y_k) f'(q_k) v_{kj} \right] f'(q_j) x_i$$

# ANN – Multilayer Perceptron/backpropagation

$$v_{kj} \leftarrow v_{kj} - \eta \frac{\partial J(\mathbf{w}, \mathbf{v})}{\partial v_{kj}}$$

$$w_{ji} \leftarrow w_{ji} - \eta \frac{\partial J(\mathbf{w}, \mathbf{v})}{\partial w_{ji}}$$

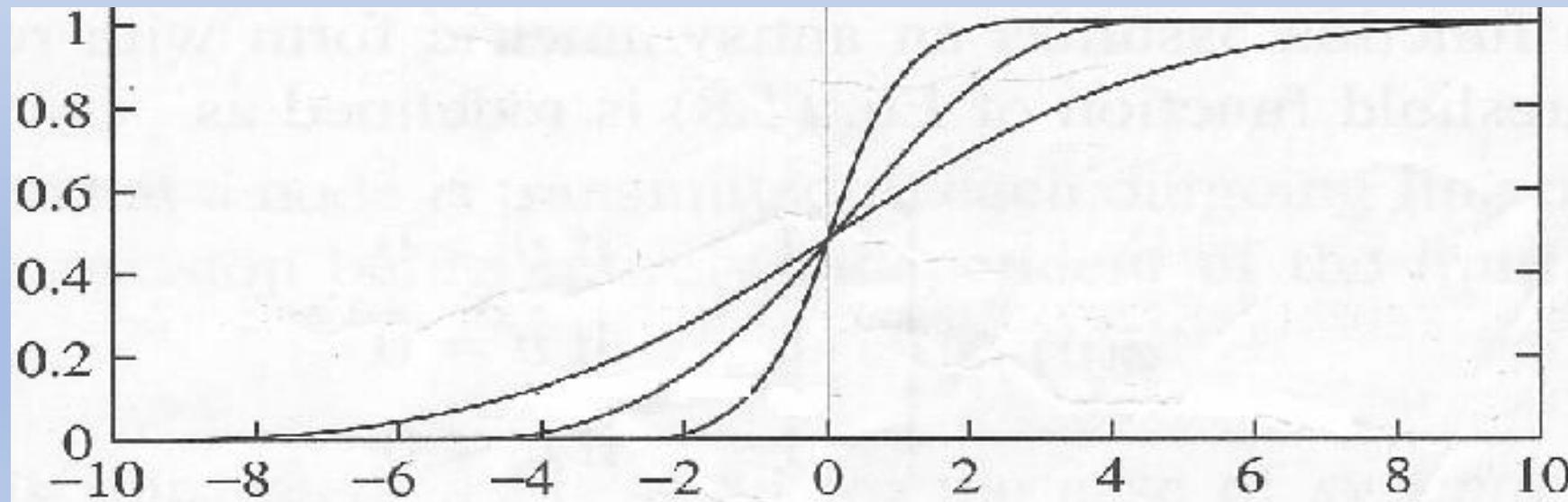
$$v_{kj} \leftarrow v_{kj} + \eta(t_k - y_k)f'(q_k)z_j$$

$$w_{ji} \leftarrow w_{ji} + \eta \left[ \sum_{k=1}^c v_{kj}(t_k - y_k)f'(q_k) \right] f'(q_j)x_i$$

## ANN – Multilayer Perceptron/backpropagation

$$f(q) = \frac{1}{1 + \exp(-aq)}, \quad 0 \leq f(q) \leq 1$$

$$f'(q) = \frac{a \exp(-aq)}{[1 + \exp(-aq)]^2} = af(q)[1 - f(q)]$$



# ANN – Multilayer Perceptron/backpropagation

- Such network is also called multilayer perceptron (MLP) having two modes of operation:

- **Feedforward**

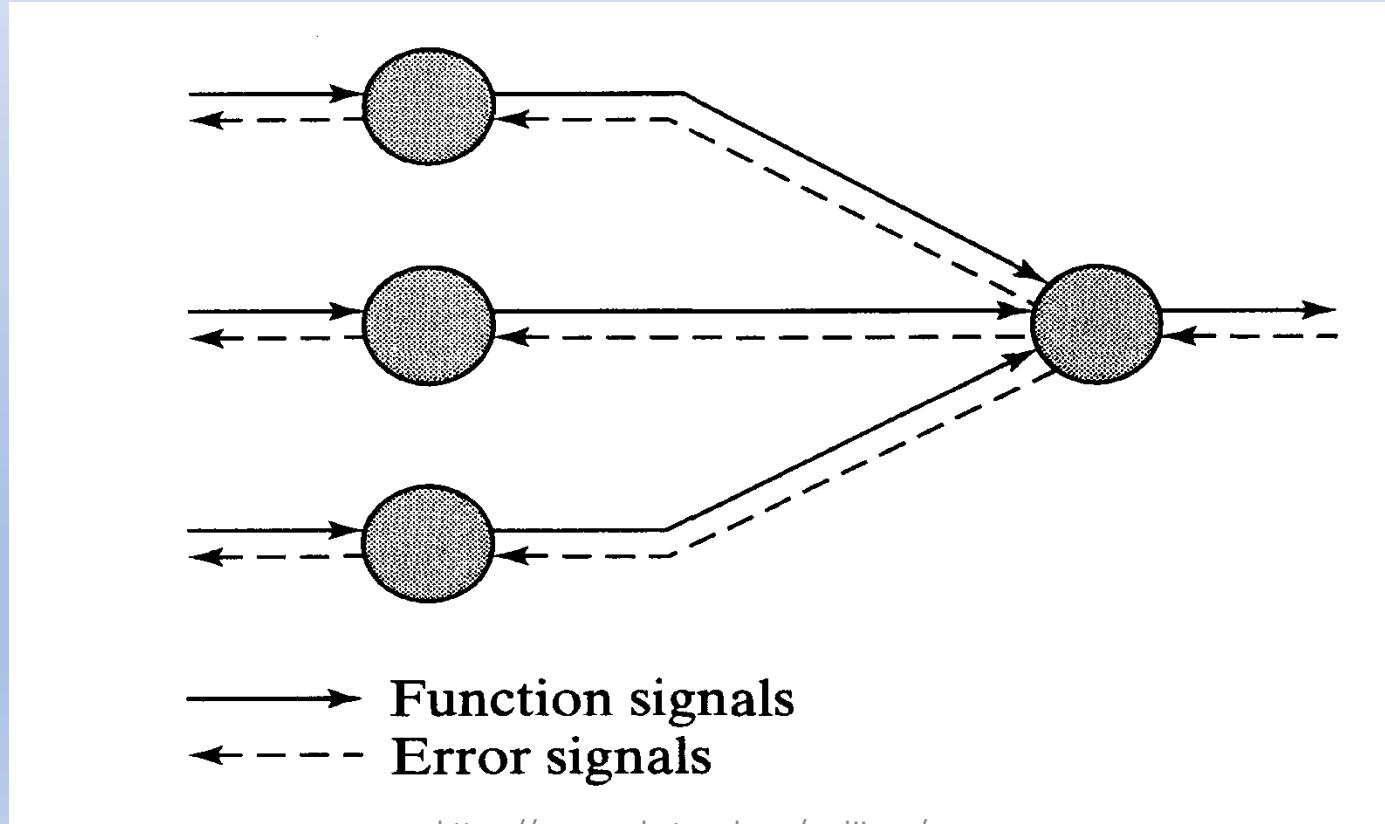
The feedforward operations consists of presenting a pattern to the input units and passing (or feeding) the signals through the network in order to get outputs units (no cycles!)

- **Learning**

The supervised learning consists of presenting an input pattern and modifying the network parameters (weights) to reduce distances between the computed output and the desired output

# ANN – Multilayer Perceptron/backpropagation

- This learning algorithm is called **backpropagation**:  
The following figure shows a portion of the MLP. Two kinds of signals are identified in this network.



# ANN – Multilayer Perceptron/backpropagation

Backpropagation training procedure:

Assume there are  $n$  the training samples:

$$\{\mathbf{x}(1), \mathbf{t}(1)\}, \{\mathbf{x}(2), \mathbf{t}(2)\}, \dots, \{\mathbf{x}(n), \mathbf{t}(n)\}$$

## (1) Initialization.

Assuming that no prior information is available, pick the synaptic weights from a uniform distribution whose mean is zero and whose variance is chosen to make the standard deviation of the activation signals lie at the transition between linear and saturated parts of the sigmoidal activation function.

## (2) Presentation of training samples

Present the network with an epoch of training samples. For each sample in the set, perform the sequence of forward and backward computations.

# ANN – Multilayer Perceptron/backpropagation

## (3) Forward computation

Let a training samples in the epoch be denoted by  $\{\mathbf{x}(i), \mathbf{t}(i)\}$ , with the  $\mathbf{x}(i)$  applied to the input layer and the desired response  $\mathbf{t}(i)$  presented to the output layer. Compute the activation signals and function signals of the network by proceeding through the network, layer by layer.

## (4) Backward computation

Compute the local gradient of the network, and adjust the synaptic weights of network.

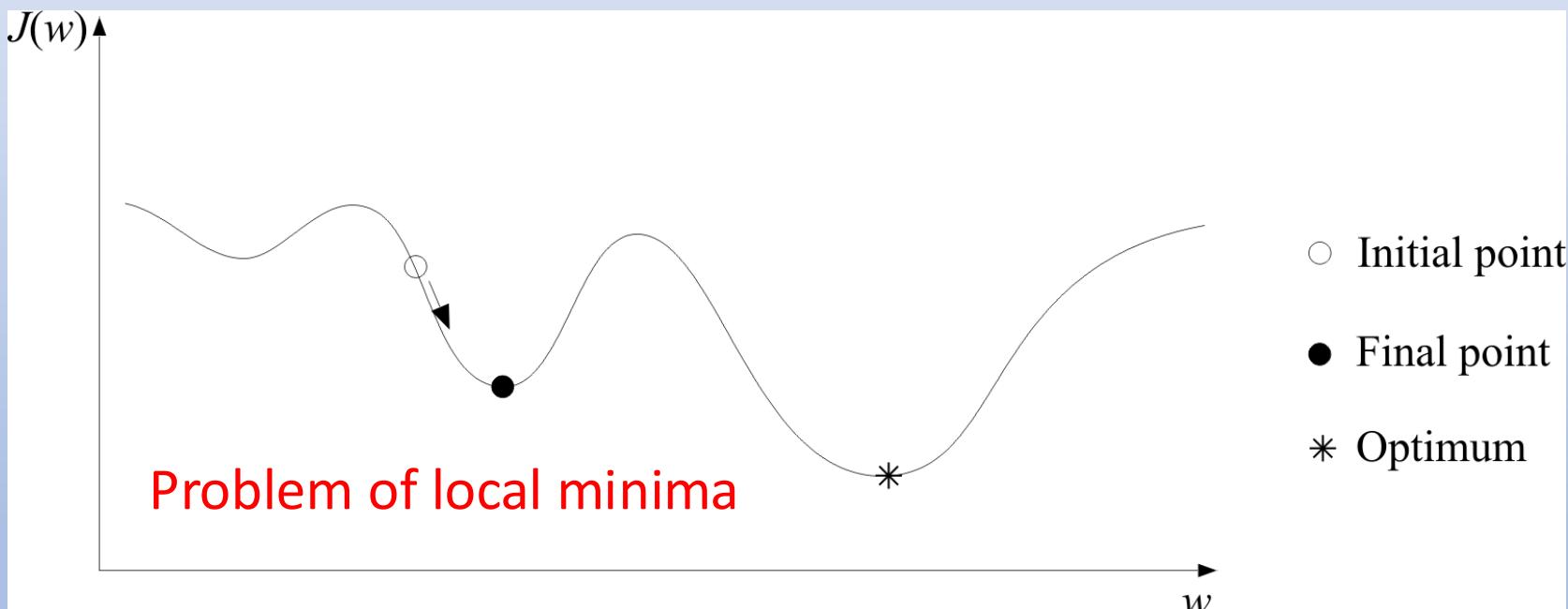
## (5) Iteration

Iterate the forward and backward computations in steps (3)-(4) by representing new epochs of training samples to the network until the stopping criterion is met.

## ANN – Problem of Local Minima

Note, the order of presentation of training samples should be randomized from epoch to epoch.

The stopping criterion can be the number of iterations, or the rate of change of the average error small enough.



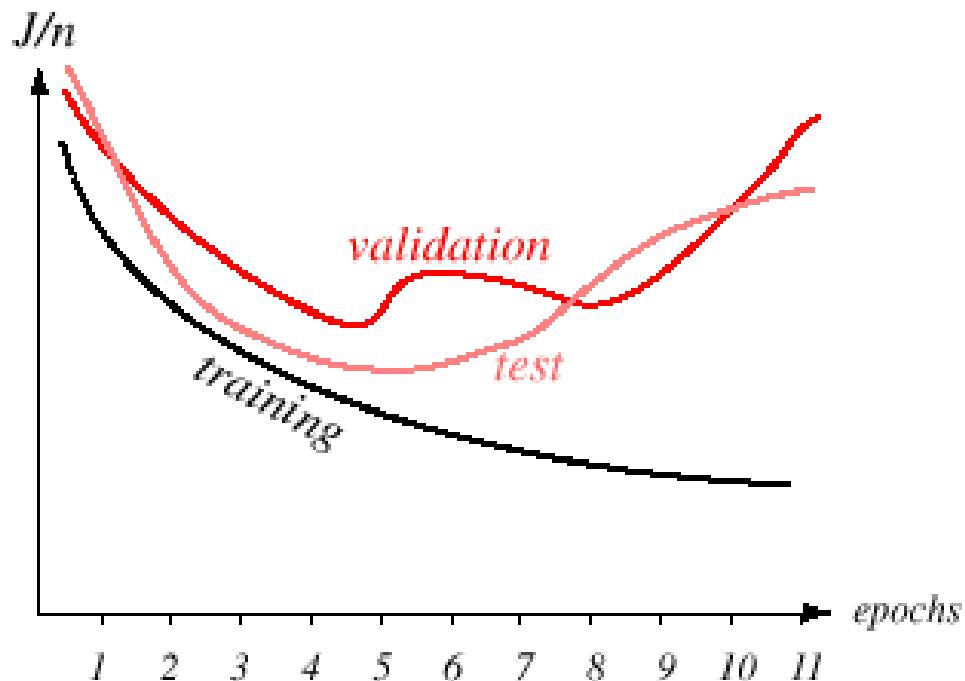
X.D. Jiang and A. Kam, [Constructing and Training Feed-Forward Neural Networks for Pattern Classification](#), *Pattern Recognition*, vol. 36, no. 4, pp. 853-867, April 2003.

## ANN – Learning Curve

- Before training starts, the error on the training set is high; through the learning process, the error becomes smaller
- The error per pattern depends on the amount of training data and the expressive power (such as the number of weights) in the network
- The average error on an independent test set is always higher than on the training set, and it can decrease as well as increase. (**Over-fitting/generalization problem**)
- A validation set is used in order to decide when to stop training ; we do not want to over fit the network and decrease the power of the classifier generalization

“We stop training at a minimum of the error on the validation set”

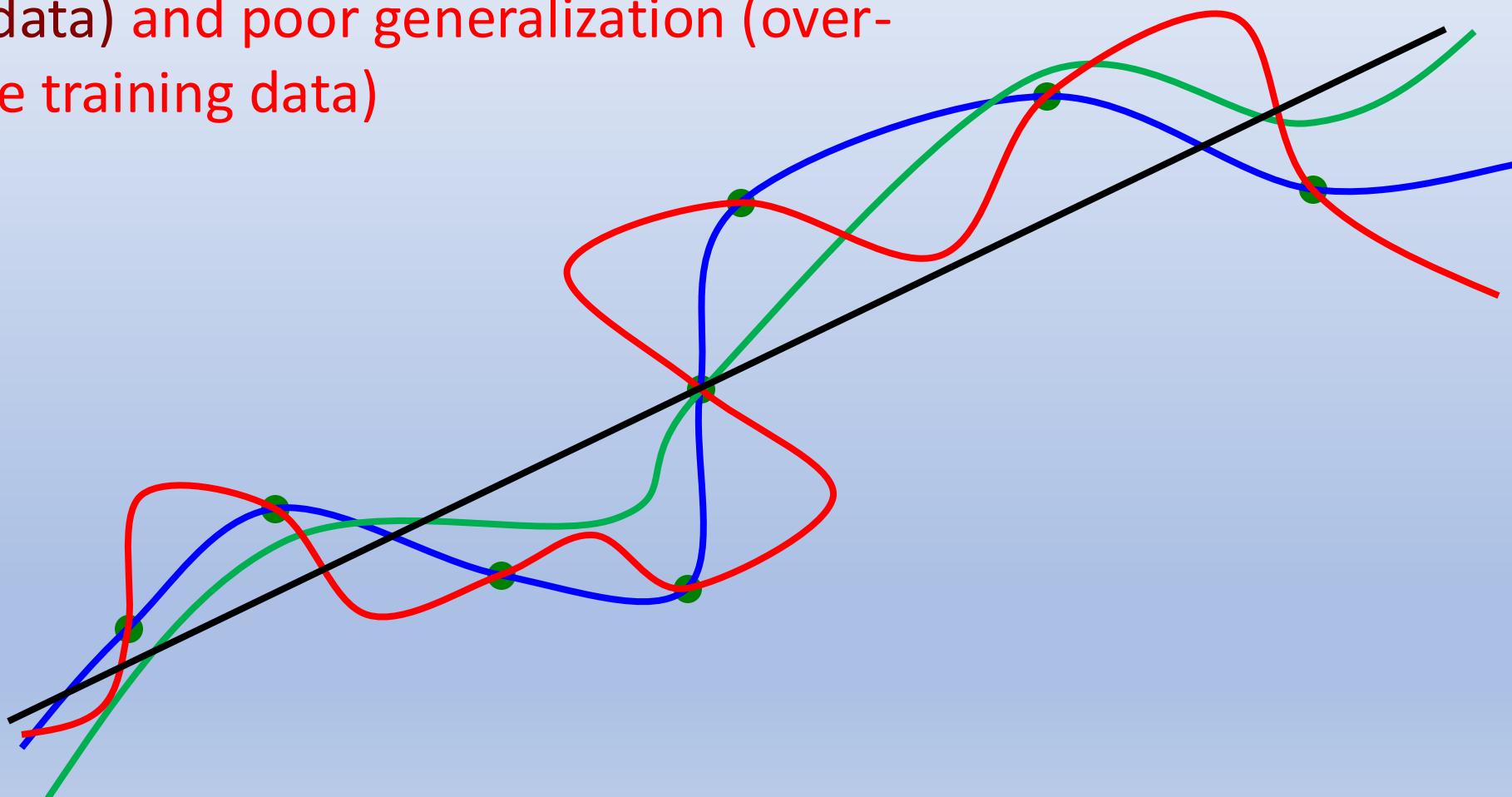
# ANN – Learning Curve



**FIGURE 6.6.** A learning curve shows the criterion function as a function of the amount of training, typically indicated by the number of epochs or presentations of the full training set. We plot the average error per pattern, that is,  $1/n \sum_{p=1}^n J_p$ . The validation error and the test or generalization error per pattern are virtually always higher than the training error. In some protocols, training is stopped at the first minimum of the validation set. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

# ANN – Understand Under- and Over-fitting

Problems of local minima (under-fitting the training data) and poor generalization (over-fitting the training data)



# ANN – Conclusions of Neural Networks

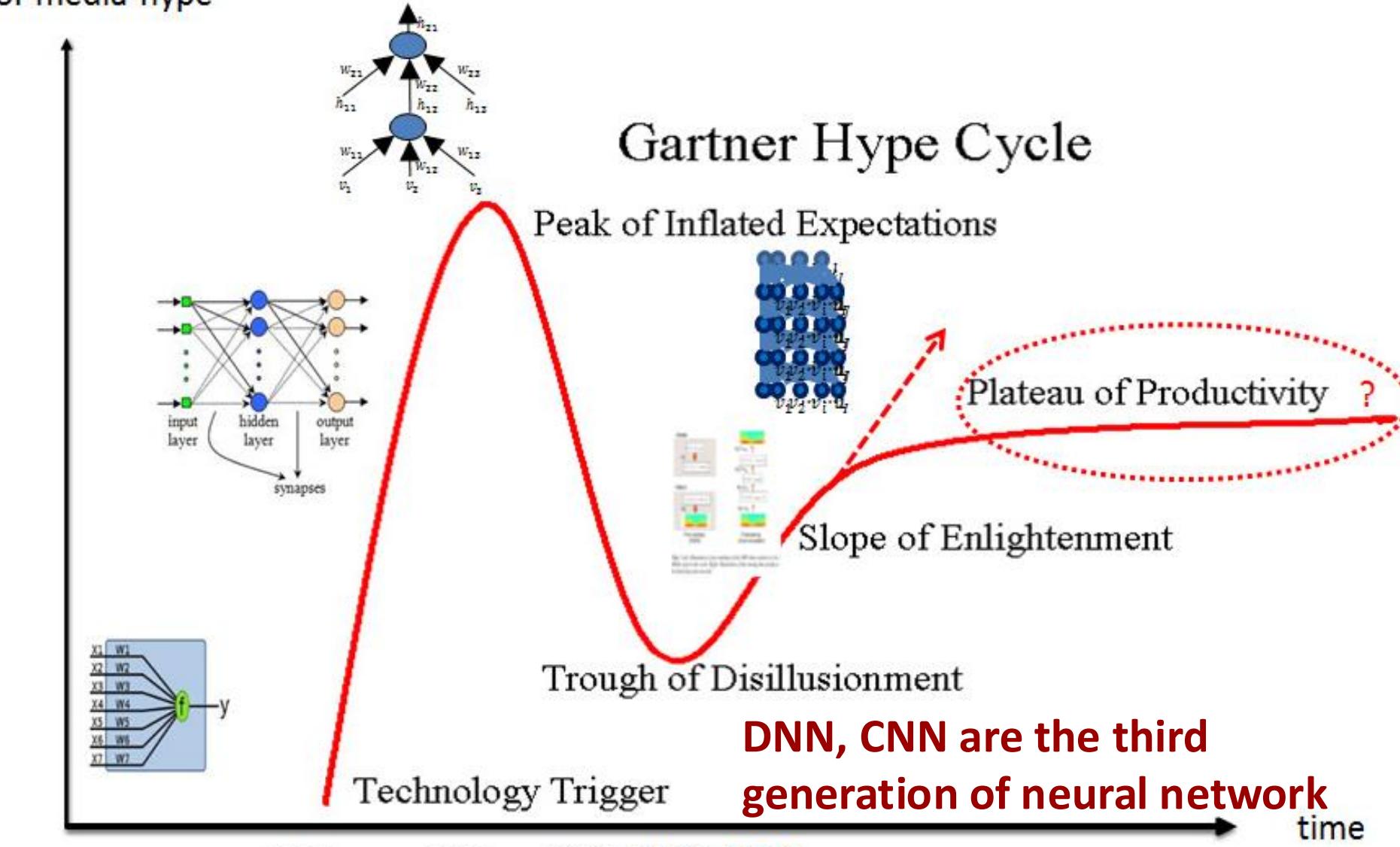
## ➤ Neural networks are Universal Approximators

It has been shown that any nonlinear continuous function can be approximated **arbitrarily close**, both, by a two-layer perceptron, with sigmoid activations, provided a **large enough** number of nodes are used.

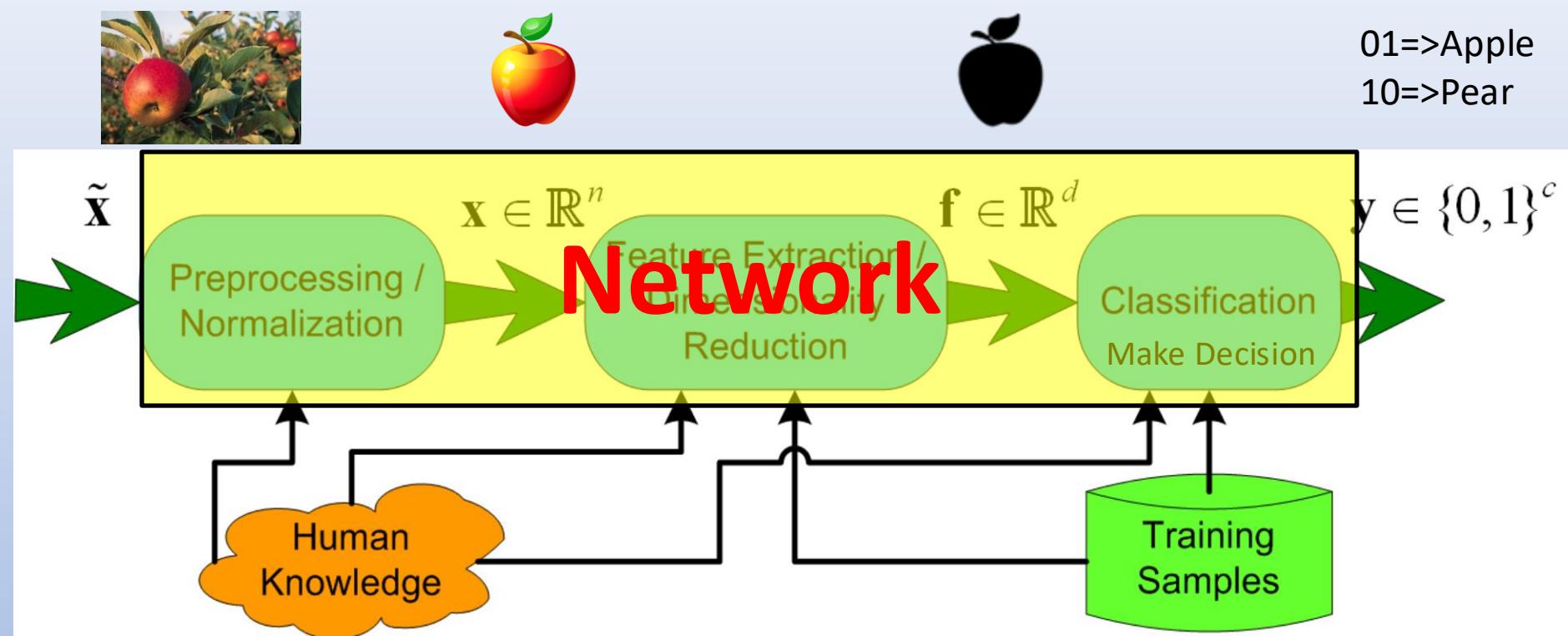
**However**, these results are of **greater theoretical interest than practical**, since the construction of such a network requires a large enough number of nodes and the weight values are to be learnt! It is still an **open question** how to find the nonlinear functions based on that training data. **Problems of local minima (under-fitting the training data) and poor generalization (over-fitting the training data)** are central issues of pattern recognition and machine learning.

# Neural Network History

Expectations  
or media hype



# Functionalities of Image Recognition Modules



- Now the machine learning goes from classification to feature extraction/ dimensionality reduction and even goes to all other steps through DNN, CNN.
- Is human knowledge useless?

Supervised learning: find an unknown mapping or **continuous** function

$$f(\bullet): \mathbf{y} = f(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^n, \quad \mathbf{y} \in \mathbb{Z}^c$$

using **finite discrete** known training samples:

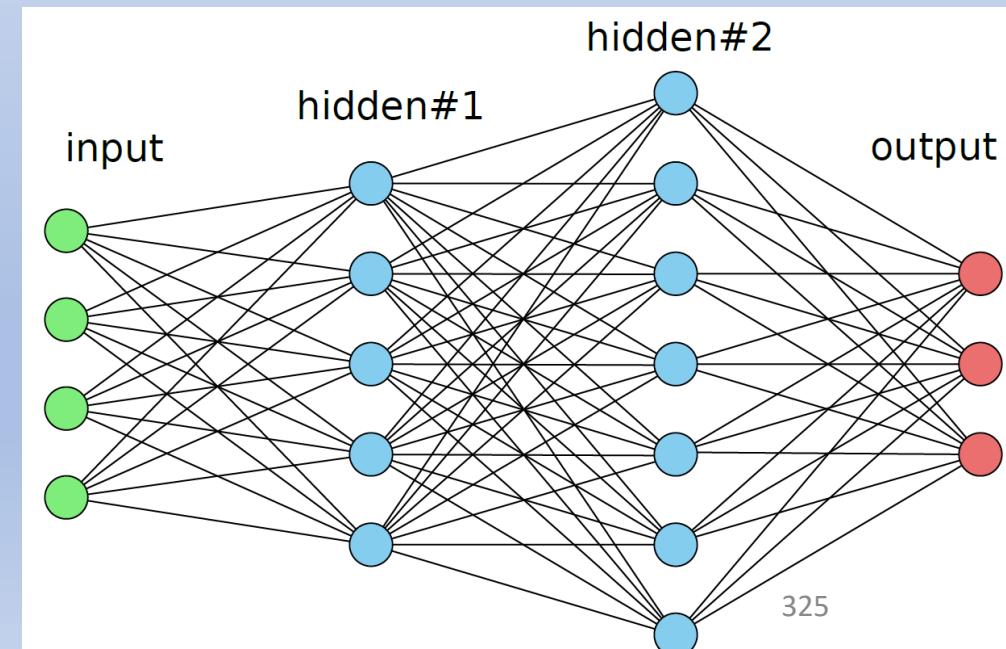
$$\{\mathbf{x}_i, \mathbf{y}_i\}, \mathbf{y}_i = f(\mathbf{x}_i), \text{ for } i=1, 2, \dots, l.$$

This is to find:  $\hat{f}(\bullet)$  by  $\min_{\hat{f}} e^2 = \min_{\hat{f}} \sum_{\forall i} \|\mathbf{y}_i - \hat{f}(\mathbf{x}_i)\|_2^2$

➤ Neural network (NN or MLP) solution:

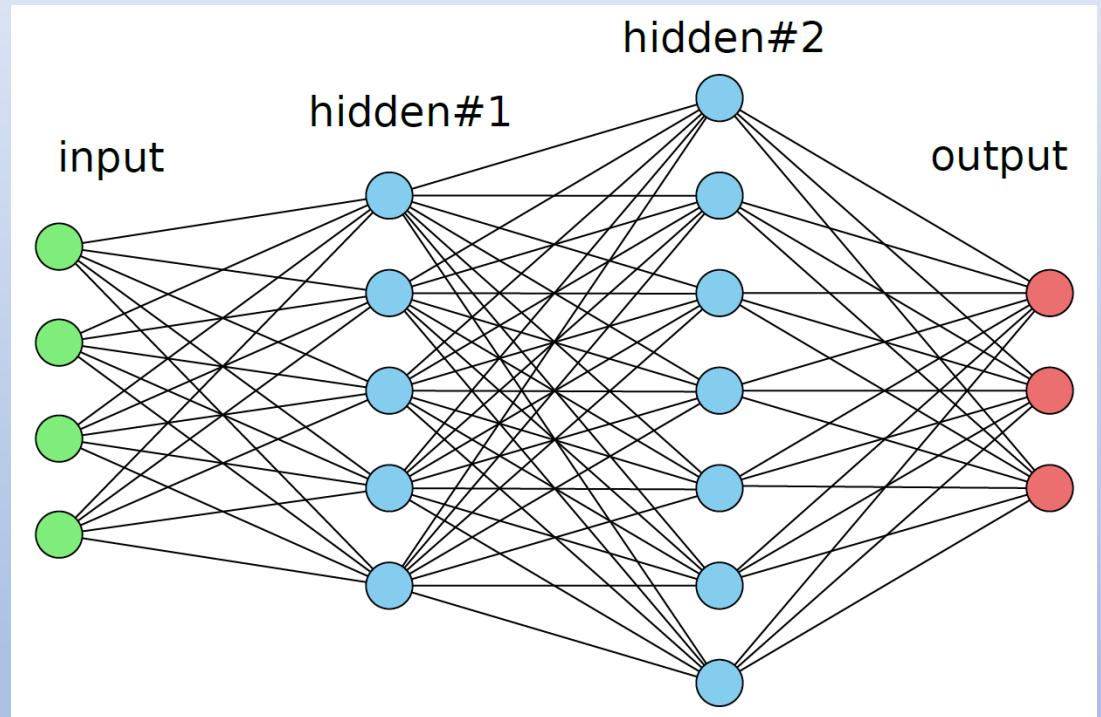
$$\mathbf{y} = h\left(\mathbf{W}_m^T \cdots h\left(\mathbf{W}_2^T h(\mathbf{W}_1^T \mathbf{x})\right)\right)$$

$h(\bullet)$ : simple nonlinear function



$$\mathbf{y} = h\left(\mathbf{W}_m^T \cdots h\left(\mathbf{W}_2^T h(\mathbf{W}_1^T \mathbf{x})\right)\right) \Rightarrow \mathbf{y} = f(\mathbf{x})$$

- Theoretically,  $m=2$  is sufficient to approximate any highly nonlinear function, i.e.  $e \Rightarrow 0$
- What further can we do?
- Why NN became dead in 2000s?
  
- Deep Learning  
 $m \gg 2$ .
- Motivation?  
difficult to find solution using  $m=2$ ?  
may be easier to find solution using  $m>2$ ?



## Problems of machine learning:

Example of MLP  
to solve highly  
nonlinear  
double spiral  
problem

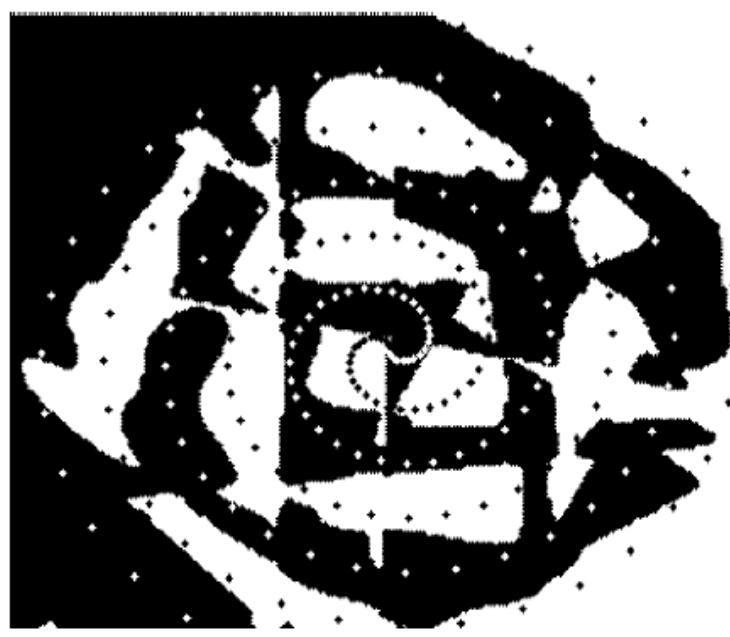
$$\min_{\hat{f}} e^2 = \min_{\hat{f}} \sum_{\forall i} \left\| \mathbf{y}_i - \hat{f}(\mathbf{x}_i) \right\|_2^2 \Rightarrow 0$$

can only find:  $\mathbf{y}_i = \hat{f}(\mathbf{x}_i)$  for  $i=1, 2, \dots, l$ .

not  $\mathbf{y} = f(\mathbf{x})$ , for the whole population  $\mathbf{x} \in \mathbb{R}^n$ ,  $\mathbf{y} \in \mathbb{Z}^c$

Problem of overfitting or poor generalization is serious

Compare to the continuous, uncountable infinite different values, a set of finite discrete samples, whatever large in size, is always negligible!



- Problems of machine learning:
- How to make
$$\hat{f}(\mathbf{x}) \Rightarrow f(\mathbf{x}), \quad \text{for the whole population } \mathbf{x} \in \mathbb{R}^n, \quad \mathbf{y} \in \mathbb{Z}^c?$$
- **Regularization!** Using human knowledge to restrict or constrain  $\hat{f}$ , so that
$$\hat{f}(\mathbf{x}) \Rightarrow f(\mathbf{x}), \quad \text{for the whole population } \mathbf{x} \in \mathbb{R}^n, \quad \mathbf{y} \in \mathbb{Z}^c$$

➤ Regularization! Using human knowledge to restrict or constrain  $\hat{f}$ ,

How to regularize  $\hat{f}$  ?

$$\min_{\hat{f}} e^2 = \min_{\hat{f}} \sum_{\forall i} \|\mathbf{y}_i - \hat{f}(\mathbf{x}_i)\|_2^2$$

↓

$$\min_{\hat{f} \in \Omega} e^2 = \min_{\hat{f} \in \Omega} \sum_{\forall i} \|\mathbf{y}_i - \hat{f}(\mathbf{x}_i)\|_2^2$$

$$\text{or } \min_{\hat{f}} \sum_{\forall i} \|\mathbf{y}_i - \hat{f}(\mathbf{W}\mathbf{x}_i)\|_2^2$$

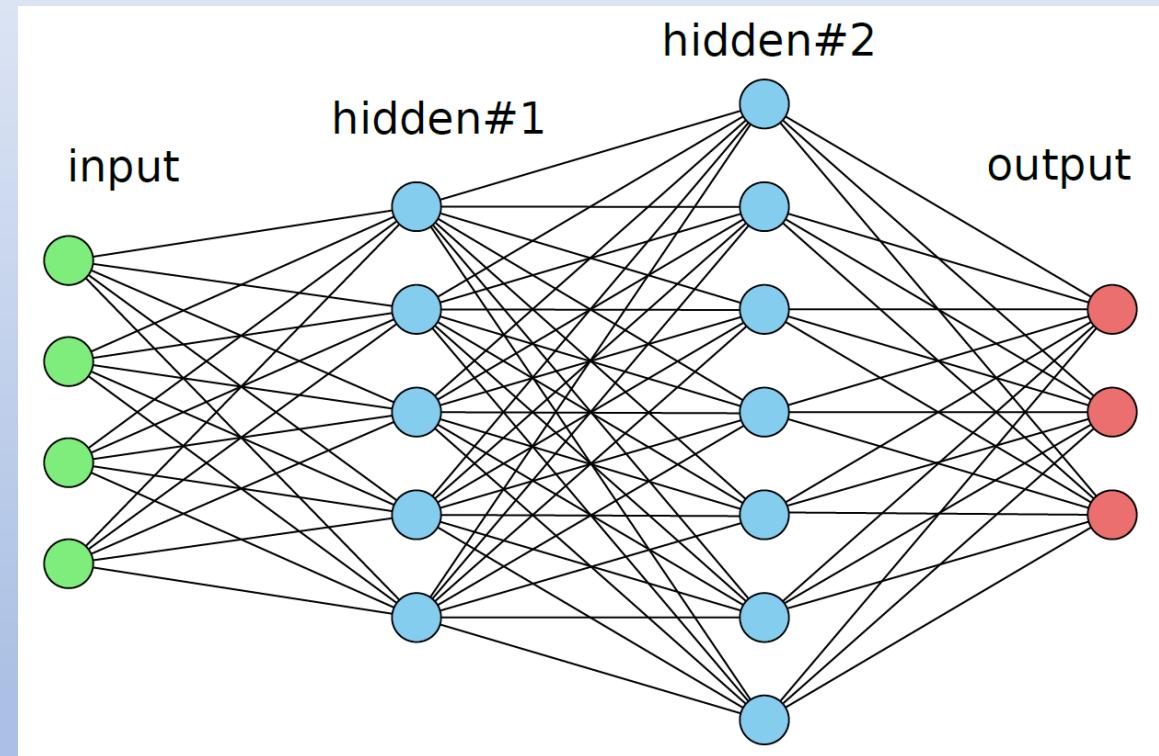
$$\min_{\hat{f}} \left[ \sum_{\forall i} \|\mathbf{y}_i - \hat{f}(\mathbf{x}_i)\|_2^2 + \lambda \Phi(\hat{f}) \right]$$

or all of the above

# Understanding Deep Learning and Convolutional Neural Networks, CNN

$$\mathbf{y} = h\left(\mathbf{W}_m^T \cdots h\left(\mathbf{W}_2^T h(\mathbf{W}_1^T \mathbf{x})\right)\right) \Rightarrow \mathbf{y} = f(\mathbf{x})$$

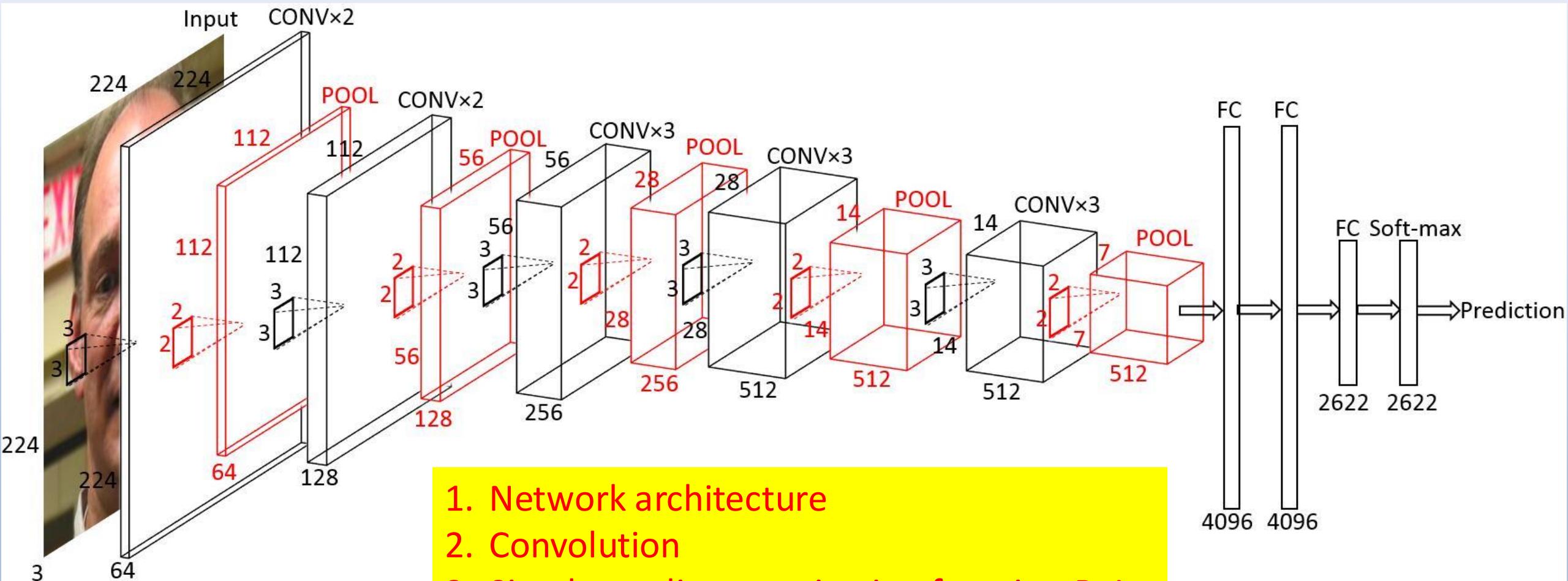
- Theoretically,  $m=2$  is sufficient  
But difficult to find solution.
- Deep Learning,  $m \gg 2$ .
- Why do we need Deep Learning,  $m \gg 2$  ?



$$\mathbf{x}^{k+1} = h(\mathbf{W}_k^T \mathbf{x}^k), \quad k = 1, 2, \dots, m$$

$$\mathbf{x}^1 = \mathbf{x}, \quad \mathbf{y} = \mathbf{x}^{m+1}, \quad \mathbf{o}^k = \mathbf{W}_k^T \mathbf{x}^k \Rightarrow \mathbf{o} = \mathbf{W}^T \mathbf{x}$$

# Convolutional network CNN appears to be quite different from MLP



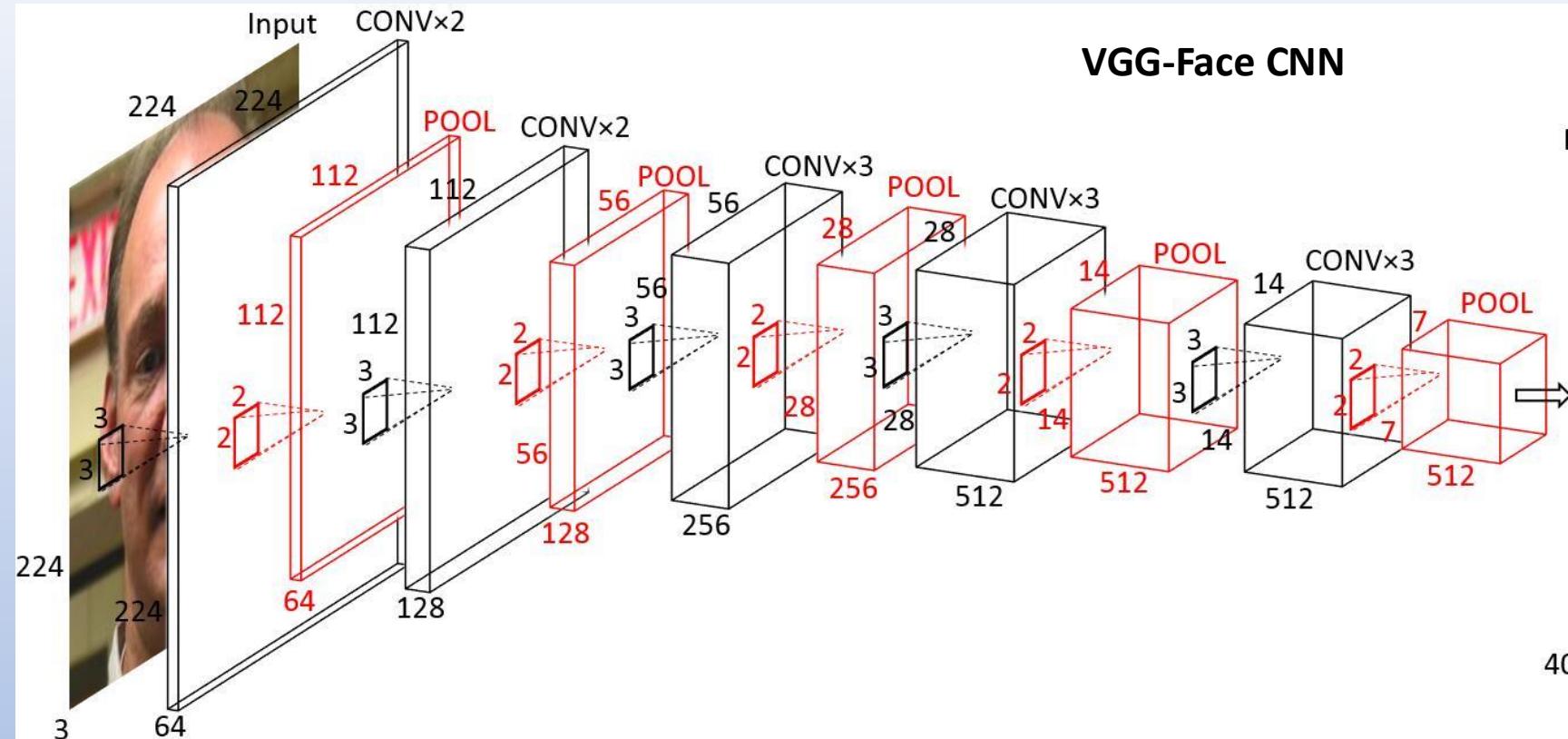
VGG-Face CNN

1. Network architecture
2. Convolution
3. Simple nonlinear activation function ReLu
4. Pooling
5. Deep, large number of layers

# Characteristics of Convolutional Network CNN

1. Network architecture
2. Convolution

Input feature maps of spatial size  $P \times Q$  and  $C$  channels and output feature maps of spatial size  $P \times Q$  and  $D$  channels are expressed



$$x_{i,j,k}, 1 \leq i \leq P, 1 \leq j \leq Q, 1 \leq k \leq C, \text{ and } y_{i,j,k}, 1 \leq i \leq P, 1 \leq j \leq Q, 1 \leq k \leq D.$$

?

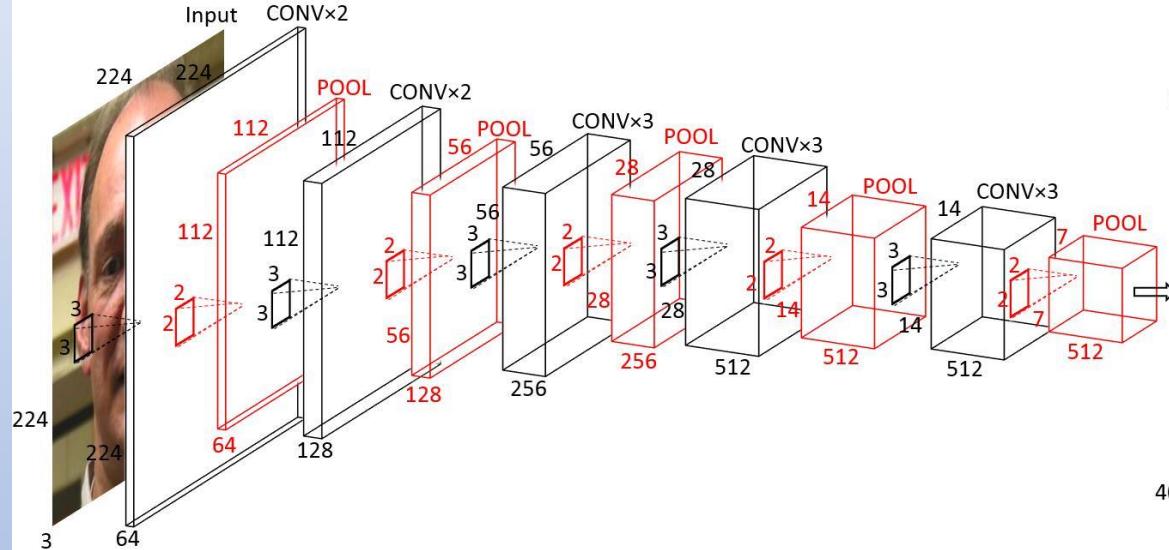
?

$$y_{i,j,k} = \sum_{n=1}^c \sum_{m=-1}^1 \sum_{l=-1}^1 w_{l,m,n,k} x_{i-l, j-m, n} + b_k, 1 \leq i \leq P, 1 \leq j \leq Q, 1 \leq k \leq D$$

# Characteristics of Convolutional Network CNN



$$y_{i,j,k} = \sum_{n=1}^c \sum_{m=-1}^1 \sum_{l=-1}^1 w_{l,m,n,k} x_{i-l, j-m, n} + b_k, \quad 1 \leq i \leq P, 1 \leq j \leq Q, 1 \leq k \leq D$$



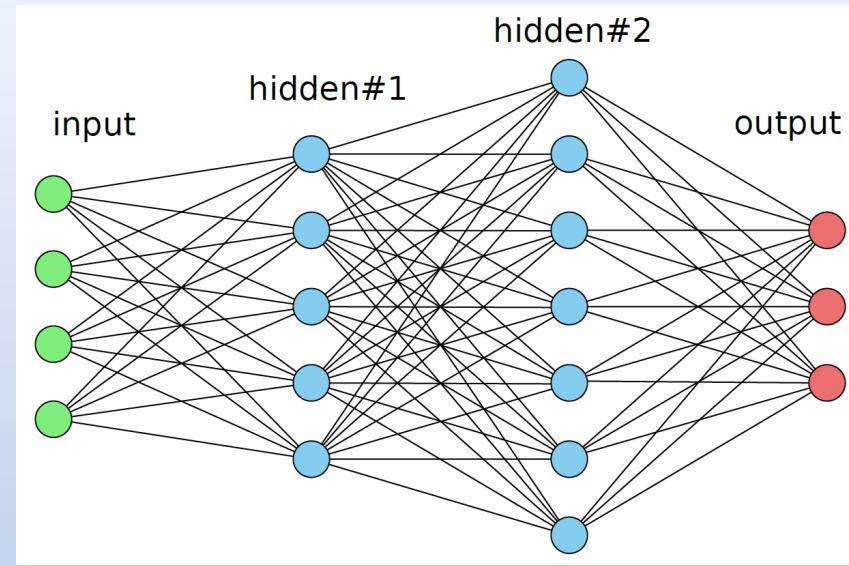
**Convolution in CNN includes a bias term.**

**Along the channel dimension, all inputs of different channels are fully connected, same as MLP.**

**So 1X1 convolution makes sense in CNN.**

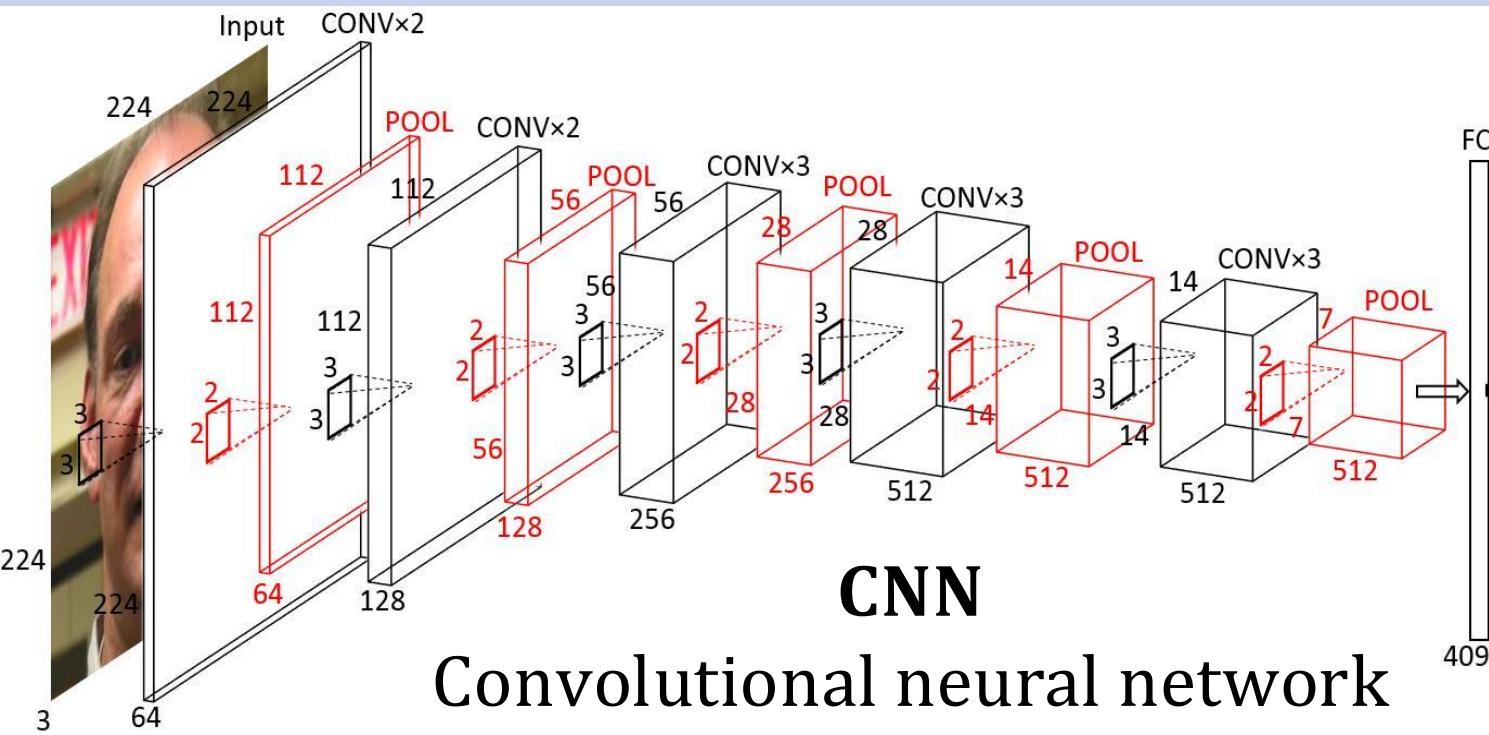
**What is the key characteristics of convolution?**

# Compare CNN with MLP in 1D: Network architecture



$$o_j = \sum_{\forall i} w_{ji} x_i = \mathbf{w}_j \mathbf{x}$$

$$\mathbf{o} = \mathbf{Wx}$$



$$o_j^q = w_j^q * x_j = \sum_{\forall i} w_i^q x_{j-i}$$

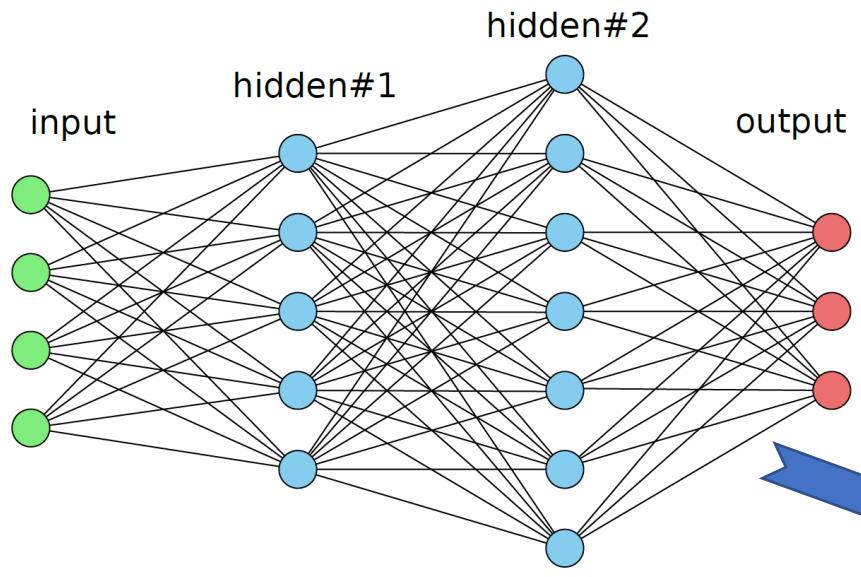
$$= \mathbf{w}^q \mathbf{x}_j = \mathbf{w}_j^q \mathbf{x}$$

$$\mathbf{o}^q = \mathbf{W}^q \mathbf{x}$$

$$\mathbf{o} = (\mathbf{o}^1, \dots, \mathbf{o}^q, \dots, \mathbf{o}^p)^T$$

$$\mathbf{W} = (\mathbf{W}^1, \dots, \mathbf{W}^q, \dots, \mathbf{W}^p)$$

$$\mathbf{o} = \mathbf{Wx}$$



$$o_j = \sum_{\forall i} w_{ji} x_i$$

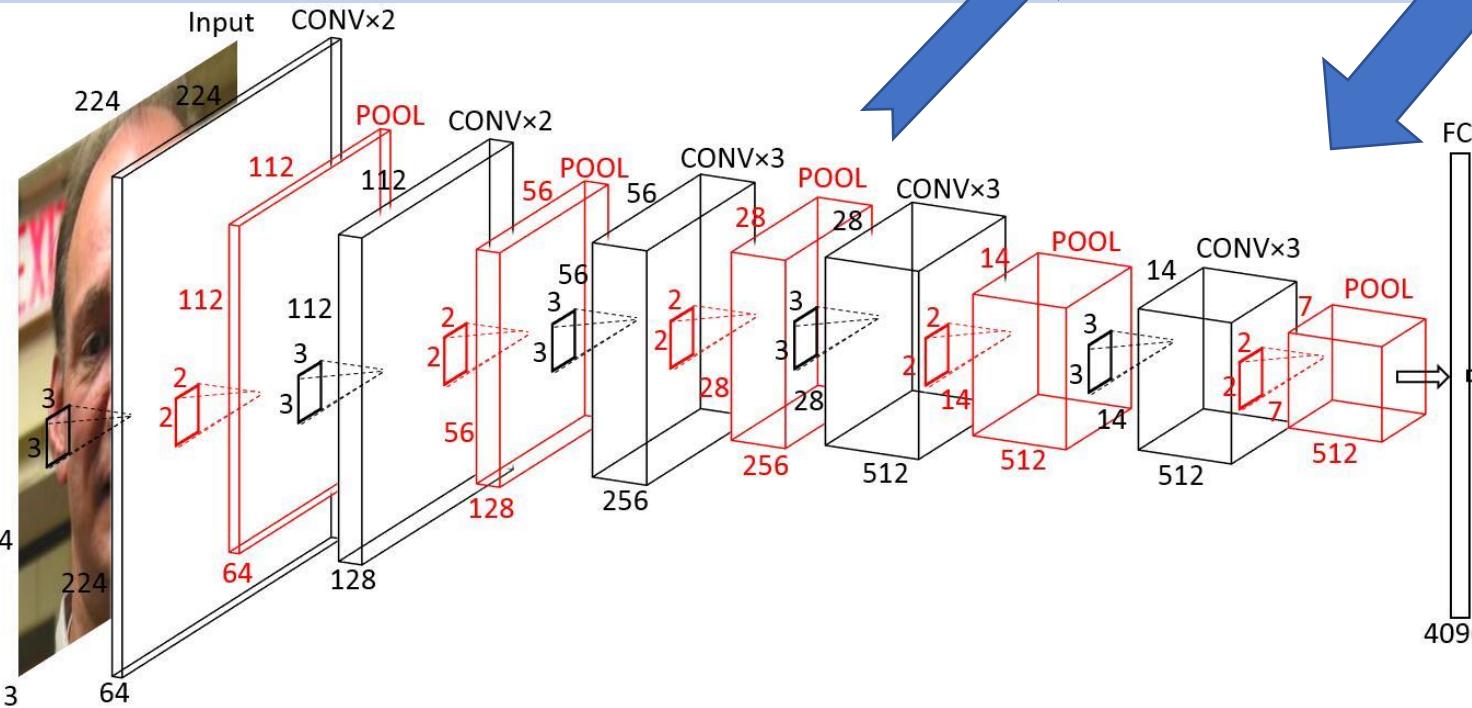
$$= \mathbf{w}_j \mathbf{x}$$

$$\mathbf{o} = \mathbf{Wx}$$

$$\mathbf{W} = (\mathbf{W}^1, \dots, \mathbf{W}^q, \dots, \mathbf{W}^p)$$

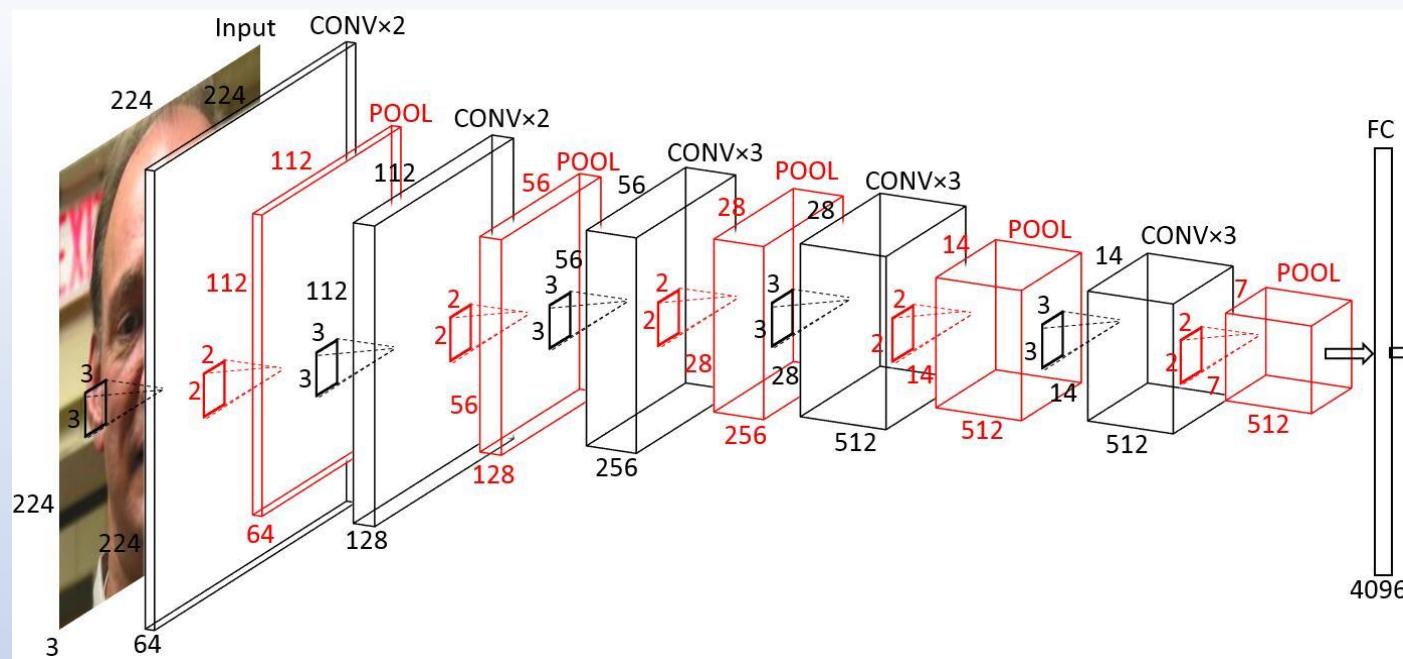
$$= \begin{pmatrix} \mathbf{g}^1 & 0 & 0 & \mathbf{g}^p & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \dots & \mathbf{g}^1 & \dots & 0 & \dots & \mathbf{g}^p & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \mathbf{g}^1 & 0 & 0 & 0 & \mathbf{g}^p \end{pmatrix}$$

**CNN is a regularized MLP**



$$o_j^q = \sum_{\forall i} w_i^q x_{j-i}$$

$$= \mathbf{w}^q \mathbf{x}_j = \mathbf{w}_j^q \mathbf{x}$$



$$\mathbf{W} = (\mathbf{W}^1, \dots, \mathbf{W}^q, \dots, \mathbf{W}^p)$$

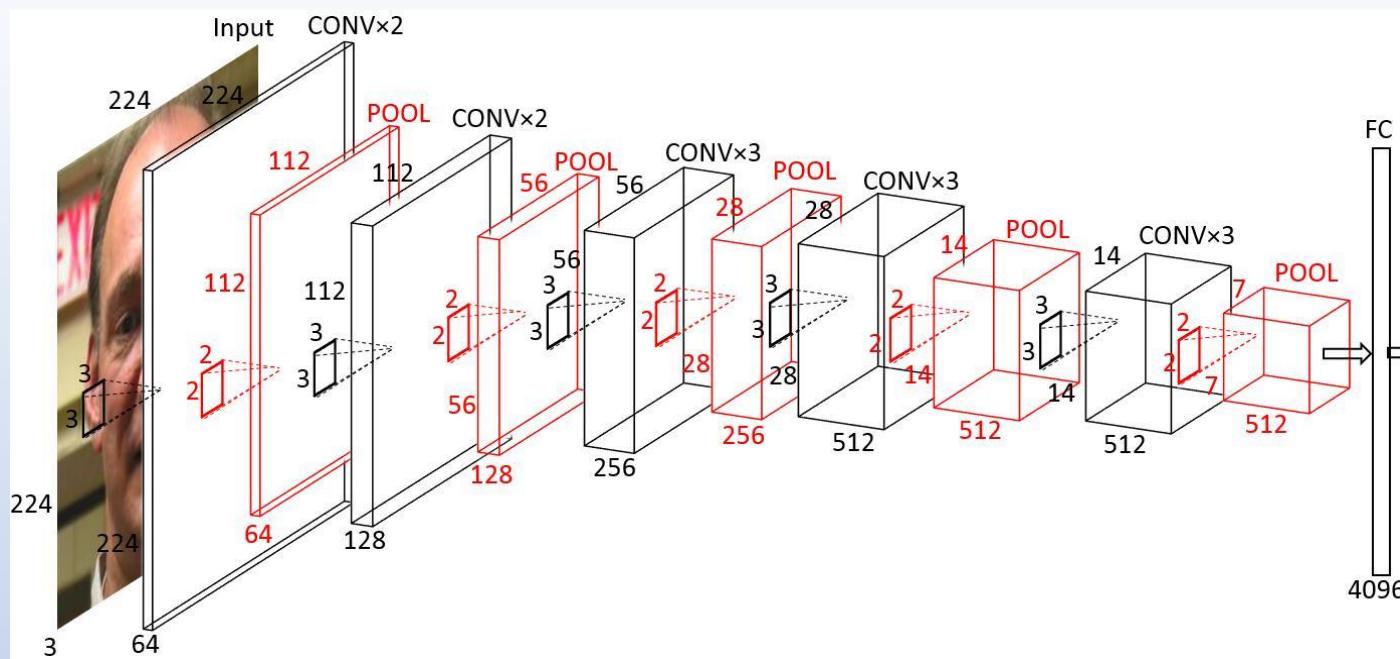
$$= \begin{pmatrix} \mathbf{g}^1 & 0 & 0 & \mathbf{g}^p & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \dots & \mathbf{g}^1 & \dots & 0 & \dots & \mathbf{g}^p & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \mathbf{g}^1 & 0 & 0 & 0 & \mathbf{g}^p \end{pmatrix}$$

How deep CNN overcome overfitting problem to make the extracted features robust to novel unseen data?

Each learnable parameter goes through all pixels. **Convert one image as a training sample to every pixel as a training sample?**

Why CNN uses smallest filter size 3X3 or 1X1?

To ensure each learnable parameter go through every input point/pixel. So each parameter **learns general rule over the whole input**, not over-fit to specific component.



$$\mathbf{W} = (\mathbf{W}^1, \dots, \mathbf{W}^q, \dots, \mathbf{W}^p) \\ = \begin{pmatrix} \mathbf{g}^1 & 0 & 0 & \mathbf{g}^p & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \dots & \mathbf{g}^1 & \dots & 0 & \dots & \mathbf{g}^p & \dots & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \mathbf{g}^1 & 0 & 0 & 0 & \mathbf{g}^p \end{pmatrix}$$

Why CNN needs deep network with many layers?

To capture large area of information.

The simplest ReLu activation function facilitates the deep layers' learning.

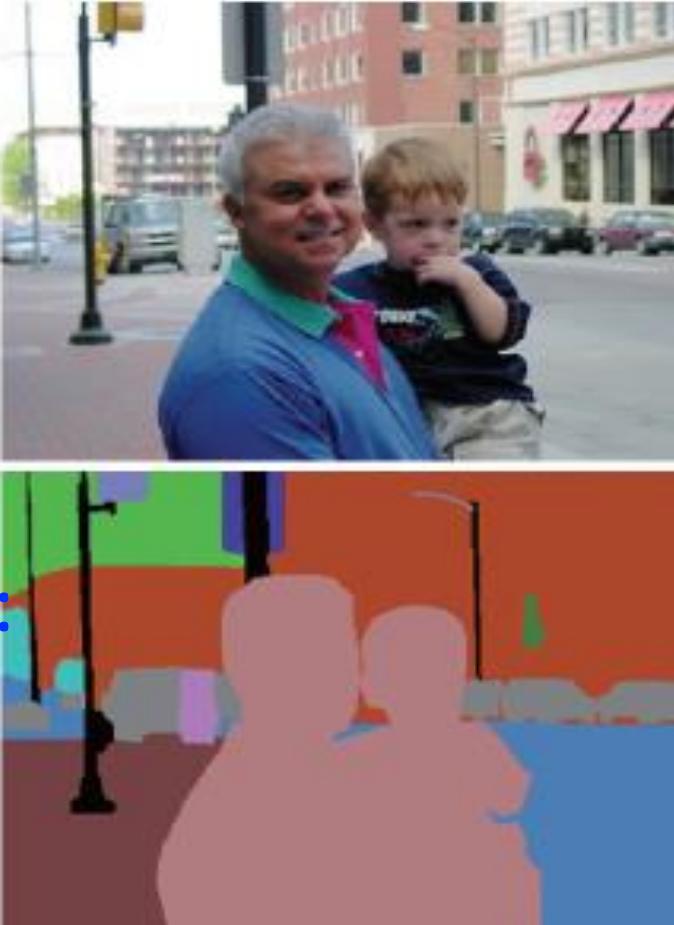


Deep learning is in fact progressive learning

Complex nonlinear input-output relation is learnt gradually, progressively, layer by layer.

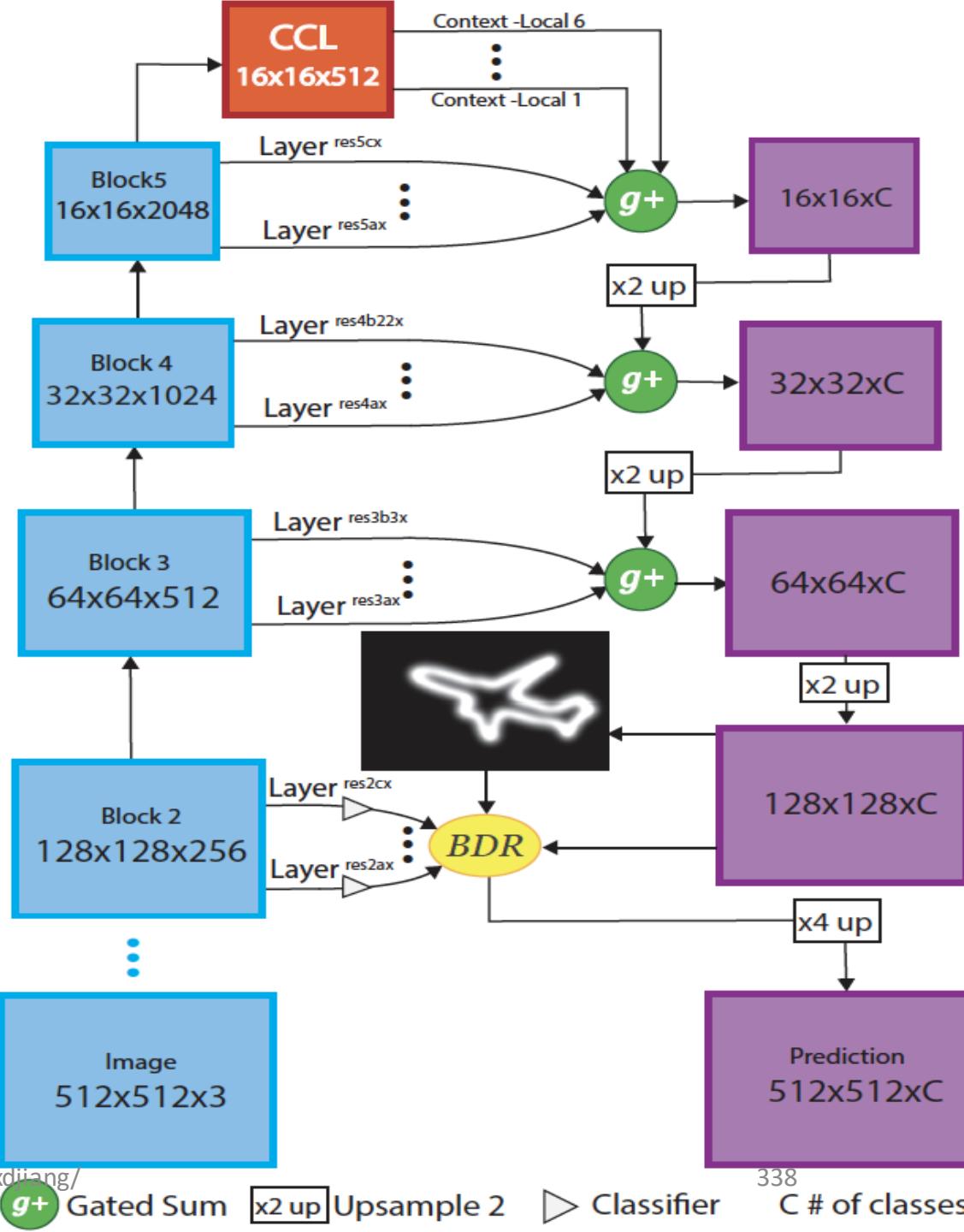
# Further Development of CNN

Image Segmentation:  
-- Pixel-wise Scene Understanding

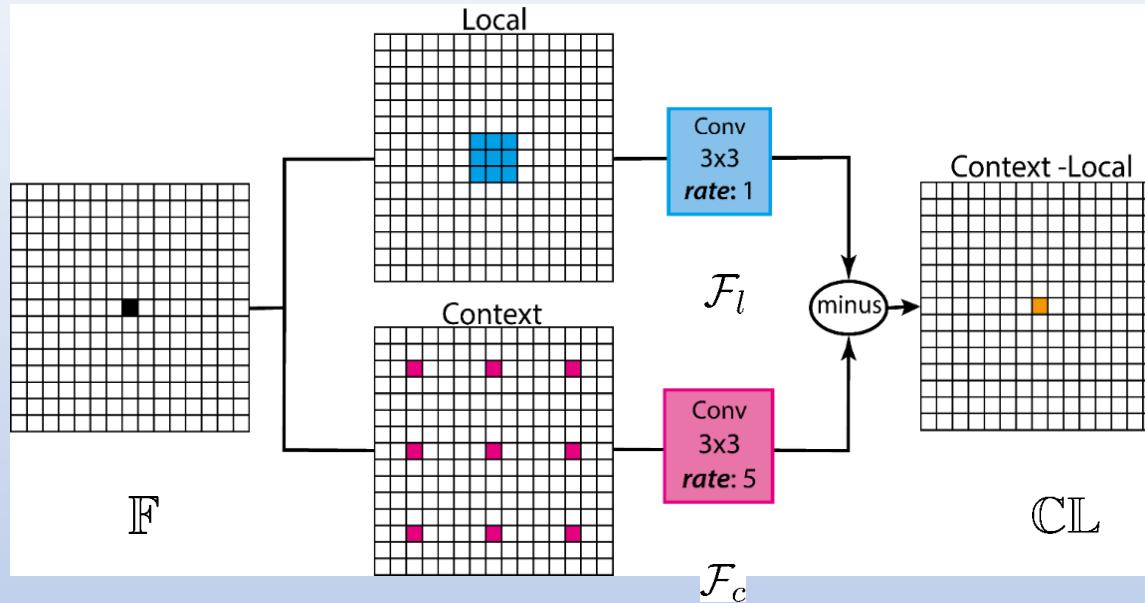


H. Ding, X.D. Jiang, B. Shuai, A. Liu, and G. Wang,  
“Context contrasted feature and gated multi-scale aggregation  
for scene segmentation,” *CVPR Oral*, 2018.

H. Ding, X. Jiang, B. Shuai, A. Liu, G. Wang, “Semantic  
Segmentation with Context Encoding and Multi-Path  
Decoding,” *IEEE Transactions on Image Processing*, 2020.



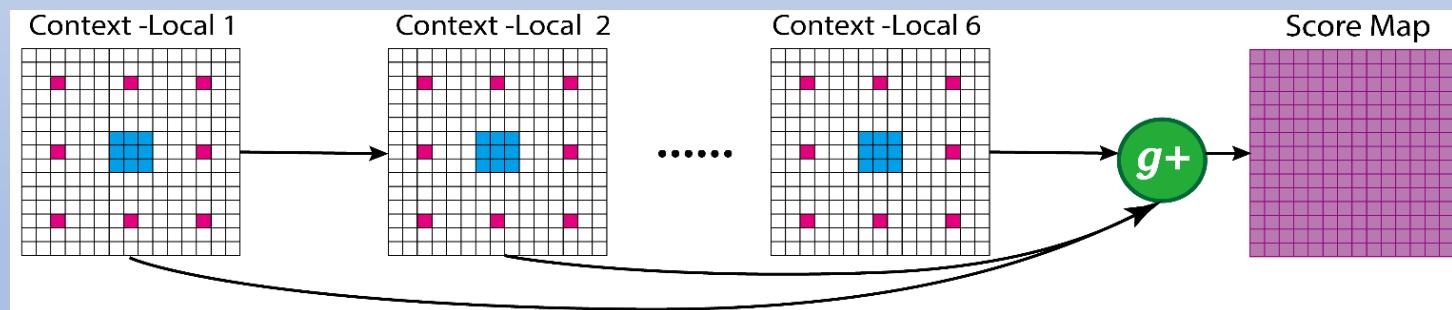
# Context Contrasted Local Features



*Context Contrasted Local Features:*

The context contrasted local features are obtained via making a contrast between the local information and its context (surroundings).

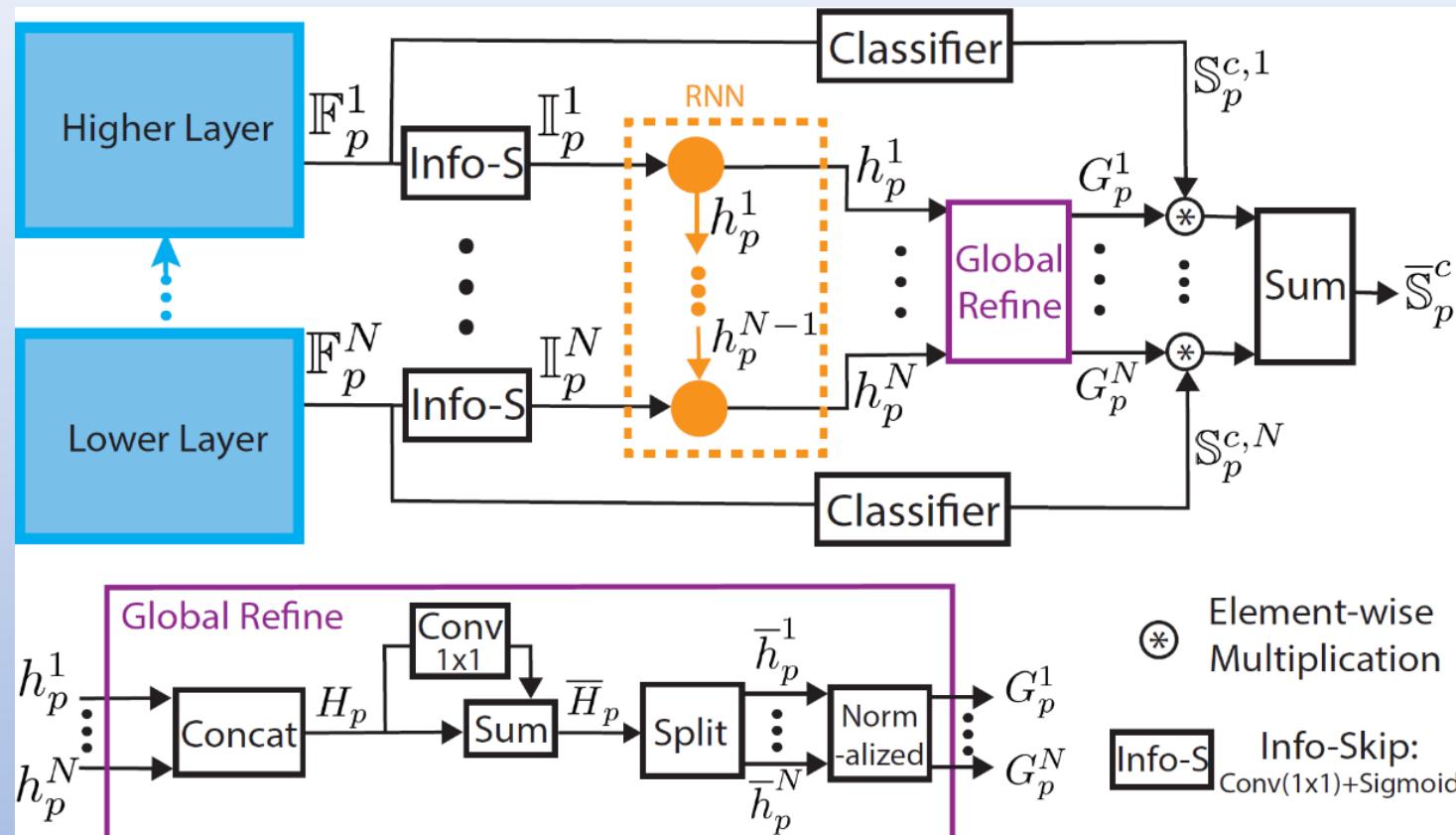
$$\text{CL} = \mathcal{F}_l(\mathbb{F}, \Theta_l) - \mathcal{F}_c(\mathbb{F}, \Theta_c)$$



*Context Contrasted Local (CCL) Model:*

Several blocks are chained to make multi-level context contrasted local features.

# Gated Multi-scale Aggregation



**Gated Sum**

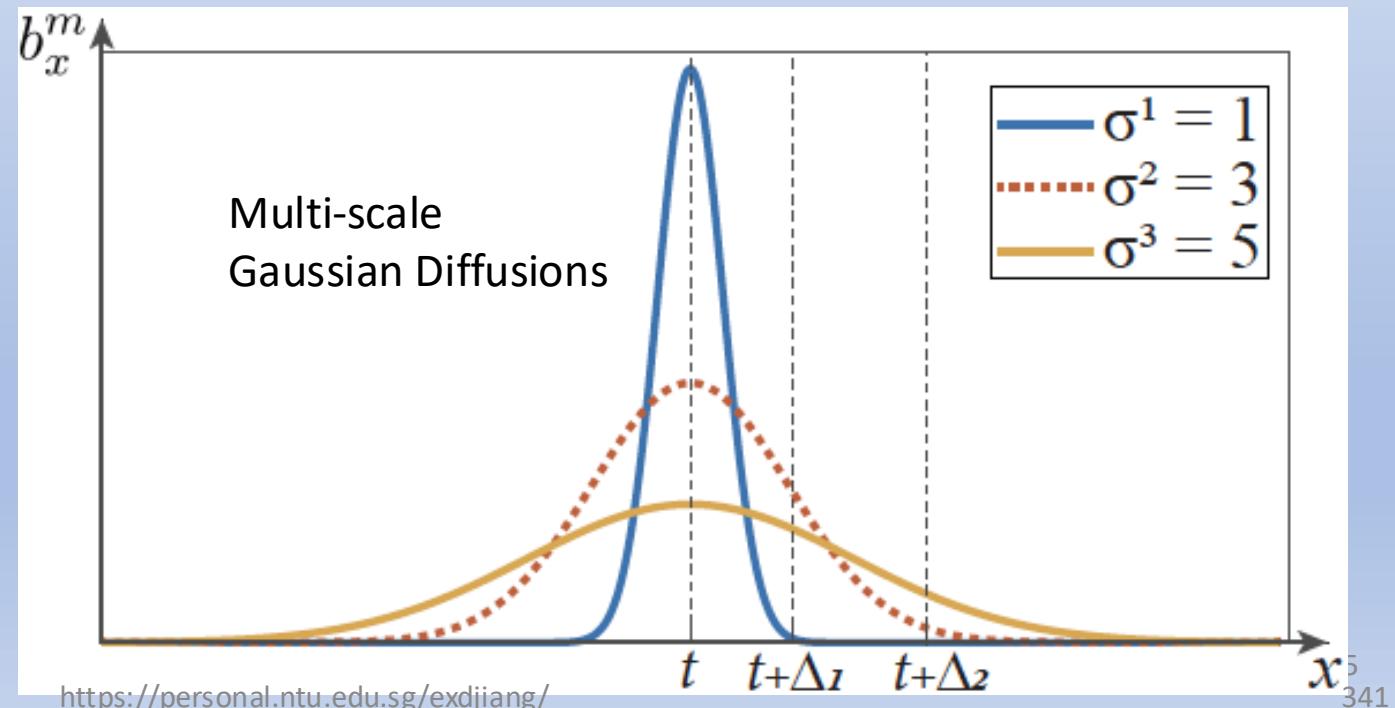
$$\bar{\mathbb{S}}_p^c = \sum_{n=1}^N G_p^n \mathbb{S}_{p,n}^{c,n}$$

The outputs of skip layers are selectively fused

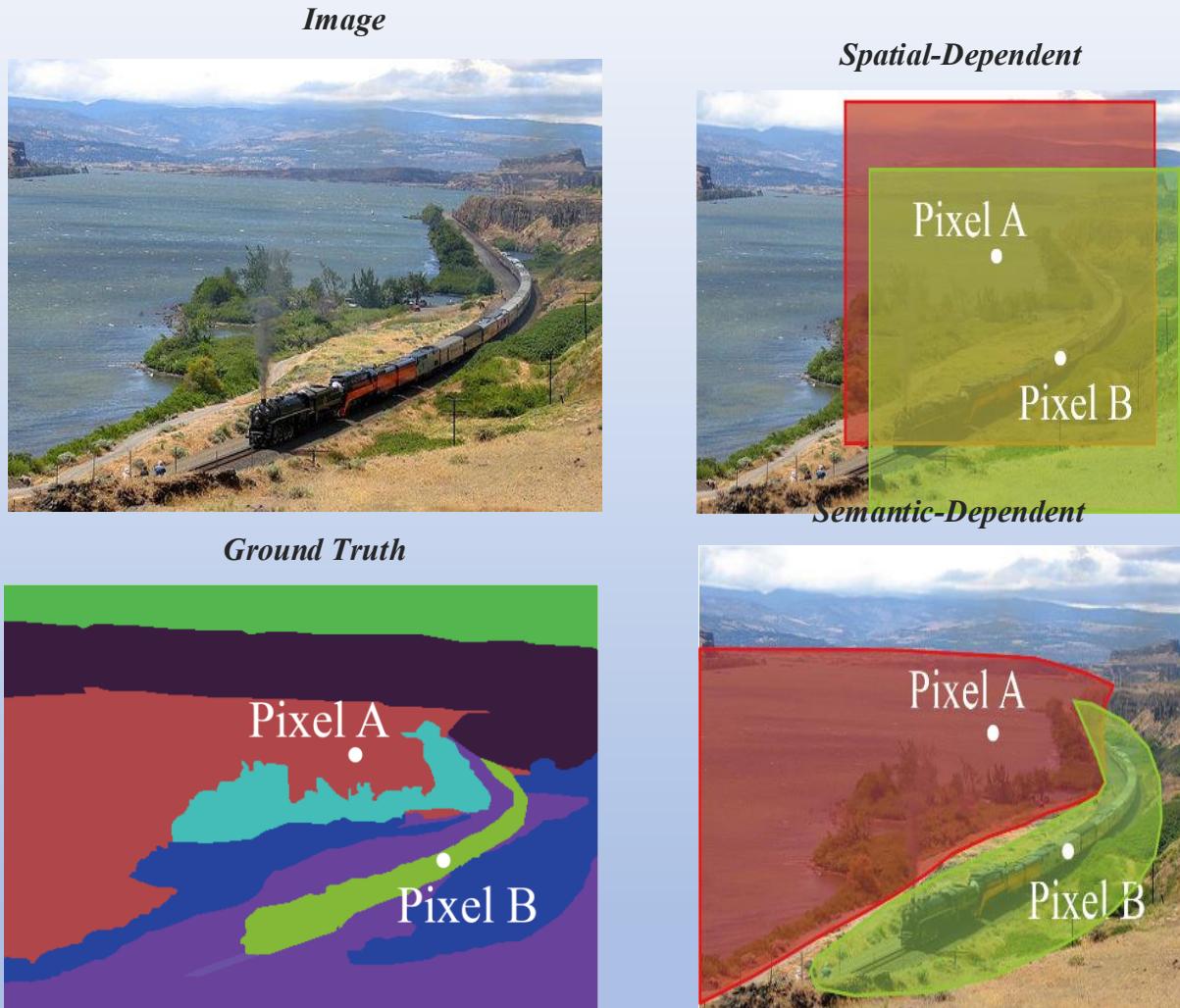
# Boundary Refinement



$$\tilde{\mathbb{S}}_p^c = \sum_{m=1}^M \mathcal{B}_p^m \circledast \dot{\mathbb{S}}_p^{c,m} + \overline{\mathbb{S}}_p^c$$



# Shape Variant Convolution



**Spatial-Dependent Context:** with pre-defined windows (e.g., the red rectangle region for pixel A in the image)

**Semantic-Dependent Context:** with variant shapes according to the semantic correlation

H. Ding, X. Jiang, B. Shuai, A. Liu, G. Wang, “[Semantic Correlation Promoted Shape-Variant Context for Segmentation](#),” **CVPR’19 Oral**, 2019.

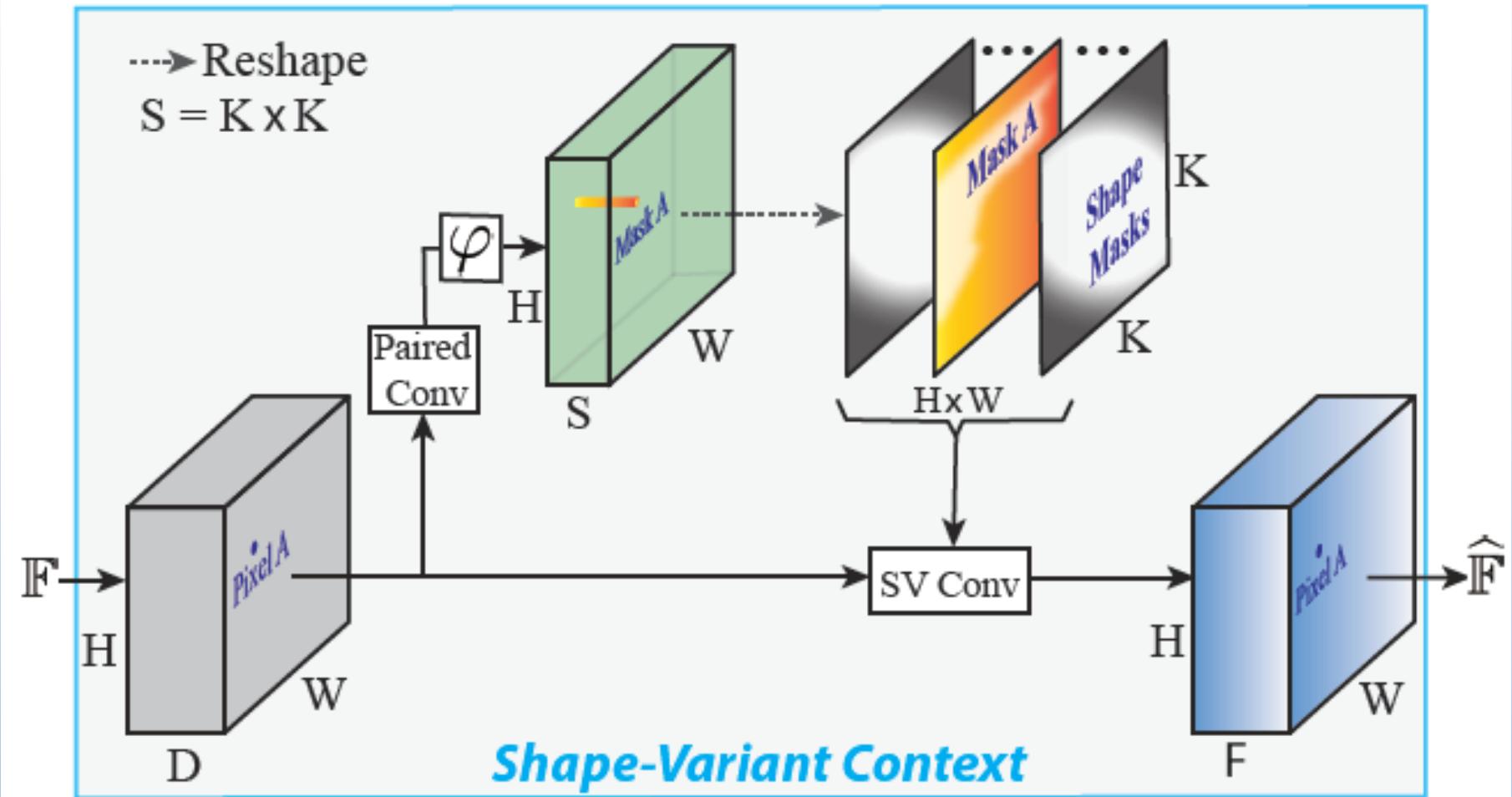
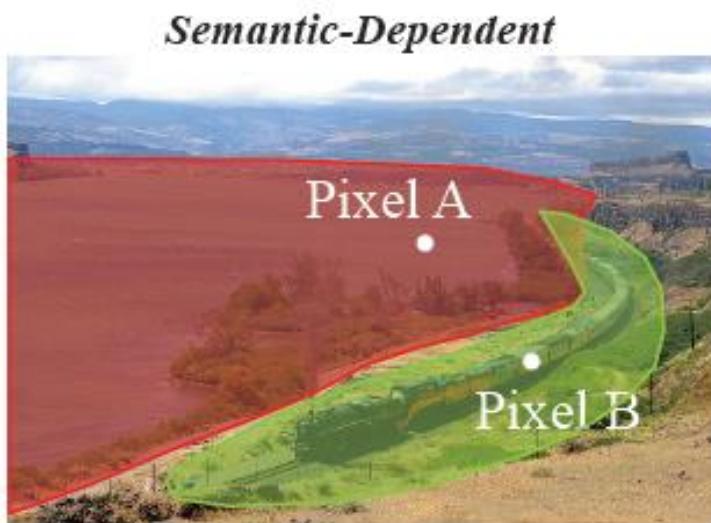
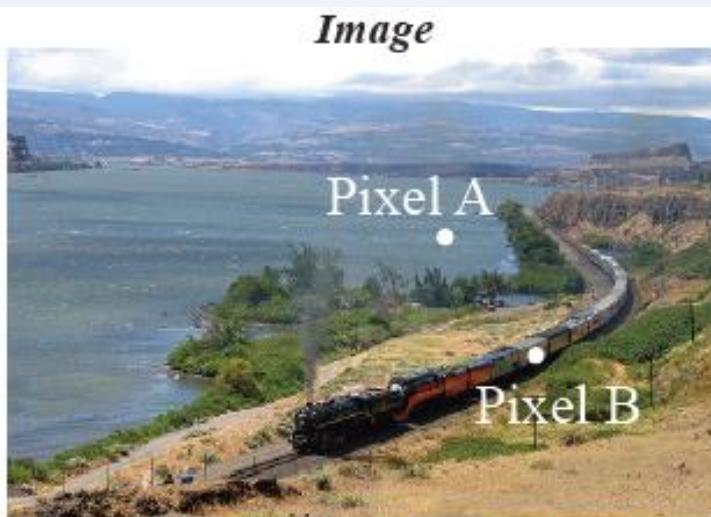
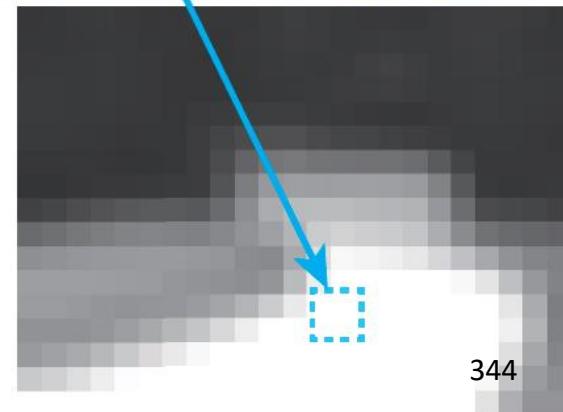
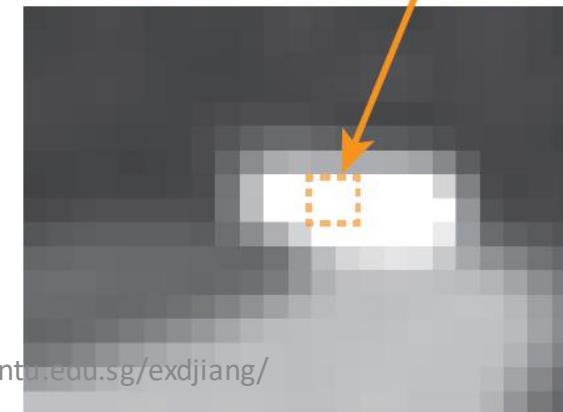
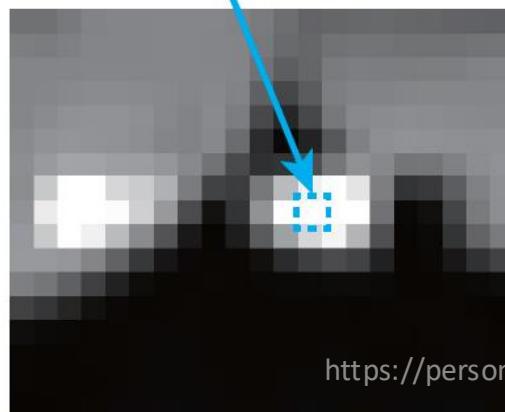
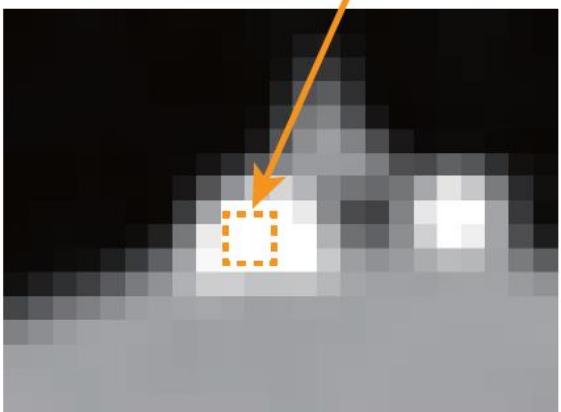
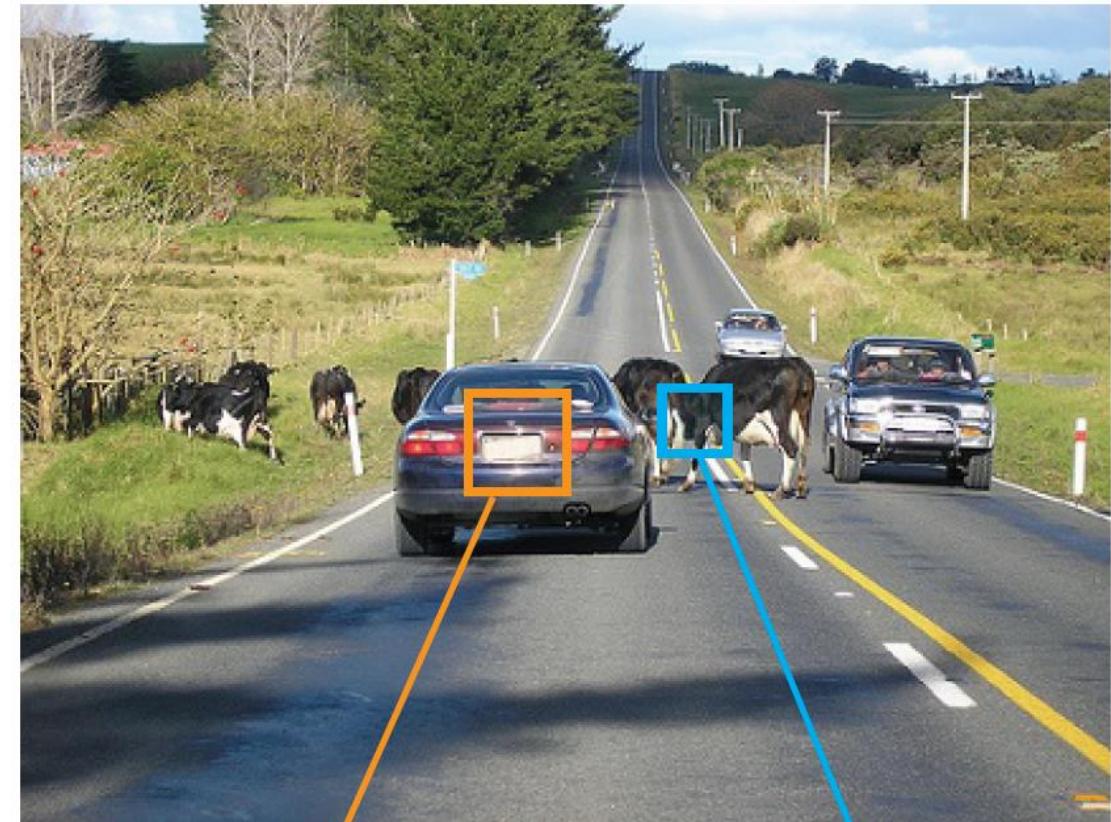


Figure 4. Semantic correlation-dependent shape-variant context aggregates surrounding information according to the semantic correlation and hence customizes an effective contextual region. It helps control the information flow within network via deciding what information to be passed or suppressed.

# Four Visual Examples Shape Mask



# Labeling De-noising in Decoding Process

Error of higher level : more on inaccurate location

Error of lower level : more on noisy classes

$$\mathcal{E}_k = F_g(F_{sf}(\mathcal{S}_k))$$

# training classes: 500

# classes in an input image: 5

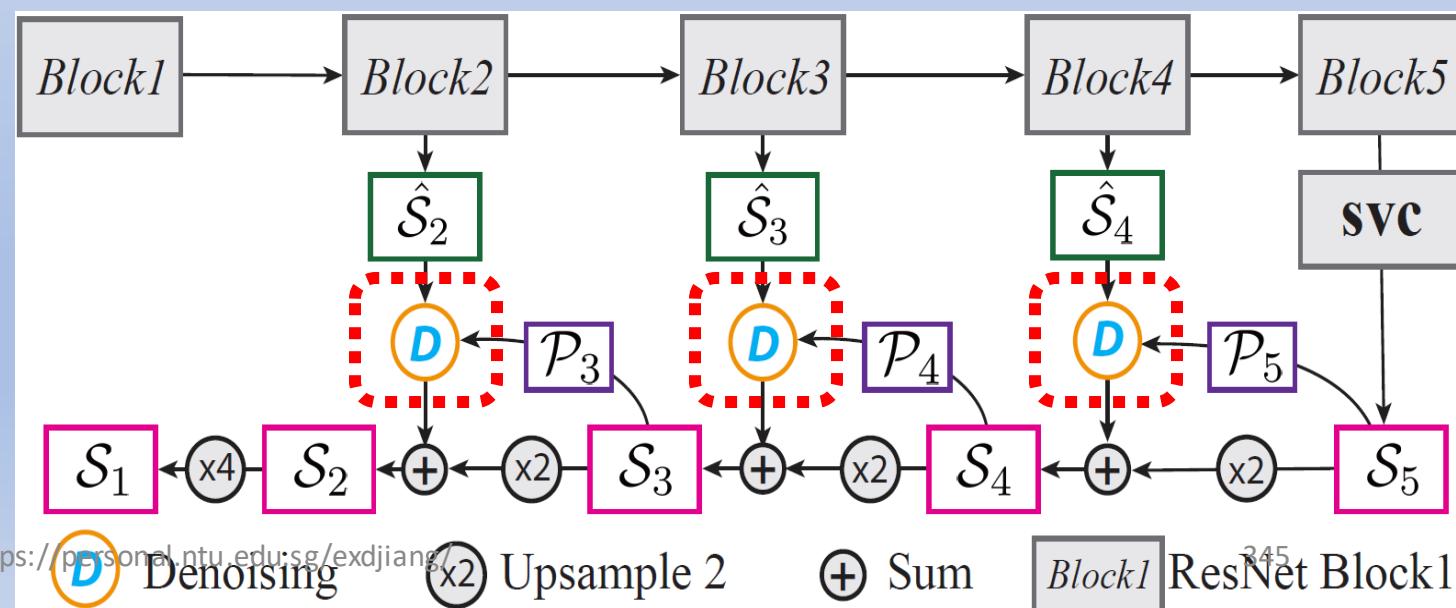
# of classified classes of layers

90<sup>th</sup>, 70<sup>th</sup>, 50<sup>th</sup>, 30<sup>th</sup>

5, 8, 14, 22

$$\mathcal{P}_k^c = \text{ReLU}(\mathcal{T} - e_k^c) \Delta_k^c$$

$$\mathcal{S}_{k-1}^c = \text{ReLU}(\hat{\mathcal{S}}_{k-1}^c - \mathcal{P}_k^c) + \mathcal{S}_k^c$$

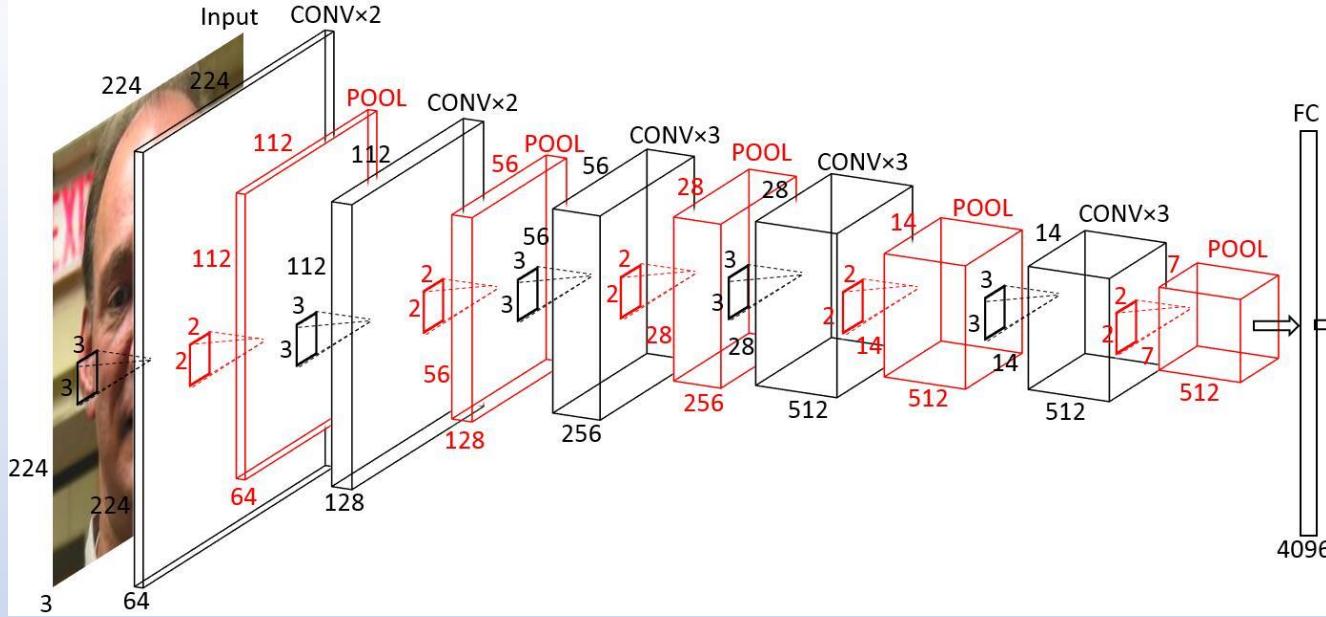


# **Topic 11**

## **Deep Learning: from CNN to Transformer**

# Understand Transformer

- The Transformer has a neural network architecture that transforms an input sequence to an output sequence such as speech, text and time series.
- Recurrent neural networks (RNNs) and its further development, Long-Short Term Memory (LSTM), are clearly related to sequences and lists. Transformer outperforms them by parallelization of their sequential operations.
- It is developed originally for natural language processing (NLP). Now it becomes a powerful neural network architecture also for computer vision, showing more powerful than CNN.
- The Transformer gets its powers thanks to its Attention module. It captures the relationships between **each** word/token in a sequence with **every** other word/token. The original paper of transformer: "**Attention Is All You Need**".
- How does the Transformer work? Is attention really everything? Any relation to CNN? Can we simply understand Transformer superior to CNN because CNN only captures local information while transformer captures global information?



Convolution captures local information so locality is the limitation of CNN?

Transformer outperforms CNN because it captures global information?

MLP is fully connected that captures the global information. Why MLP is problematic that performs badly?

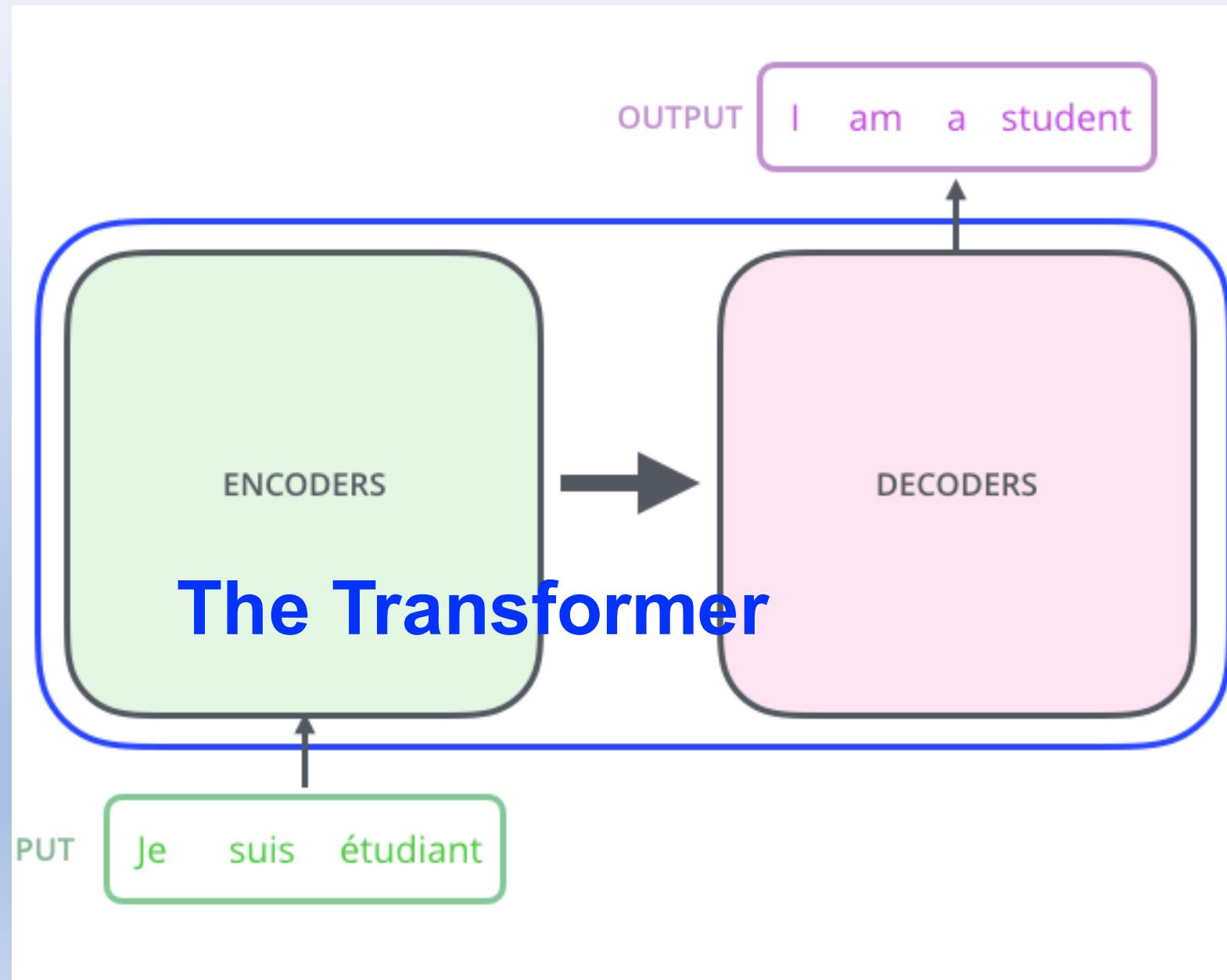
**Global information learnt by CNN is on the whole training dataset.**

# Transformer consists of Encoders and Decoders

A machine translation application, it takes a sentence in one language, and outputs its translation in another.

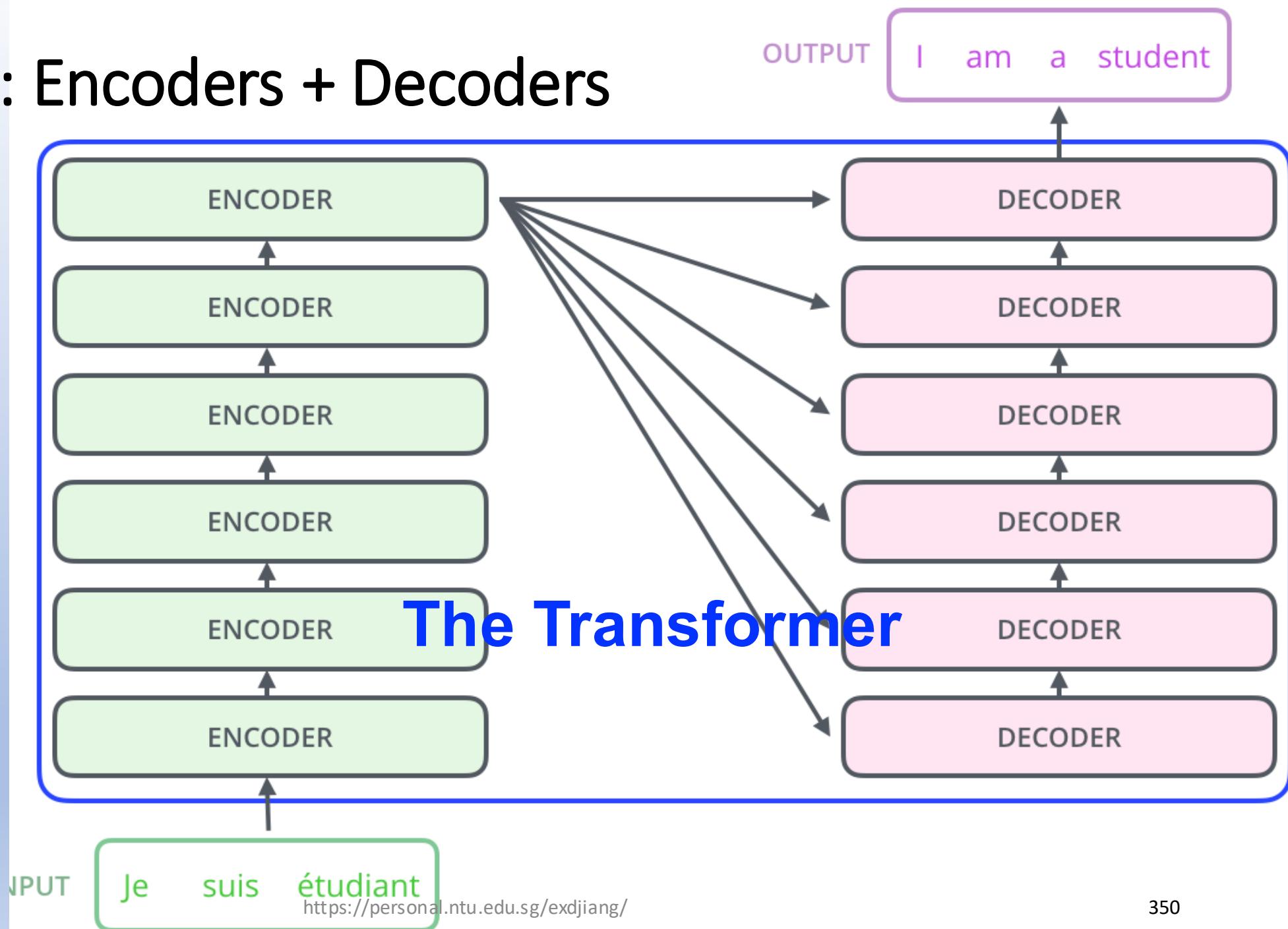
All words/tokens of a sentence/image are parallelly inputted into the transformer.

Output is a linear classification of the decoder output.



# Transformer: Encoders + Decoders

A stack of encoders of the same structure  
and  
a stack of decoders of the same structure

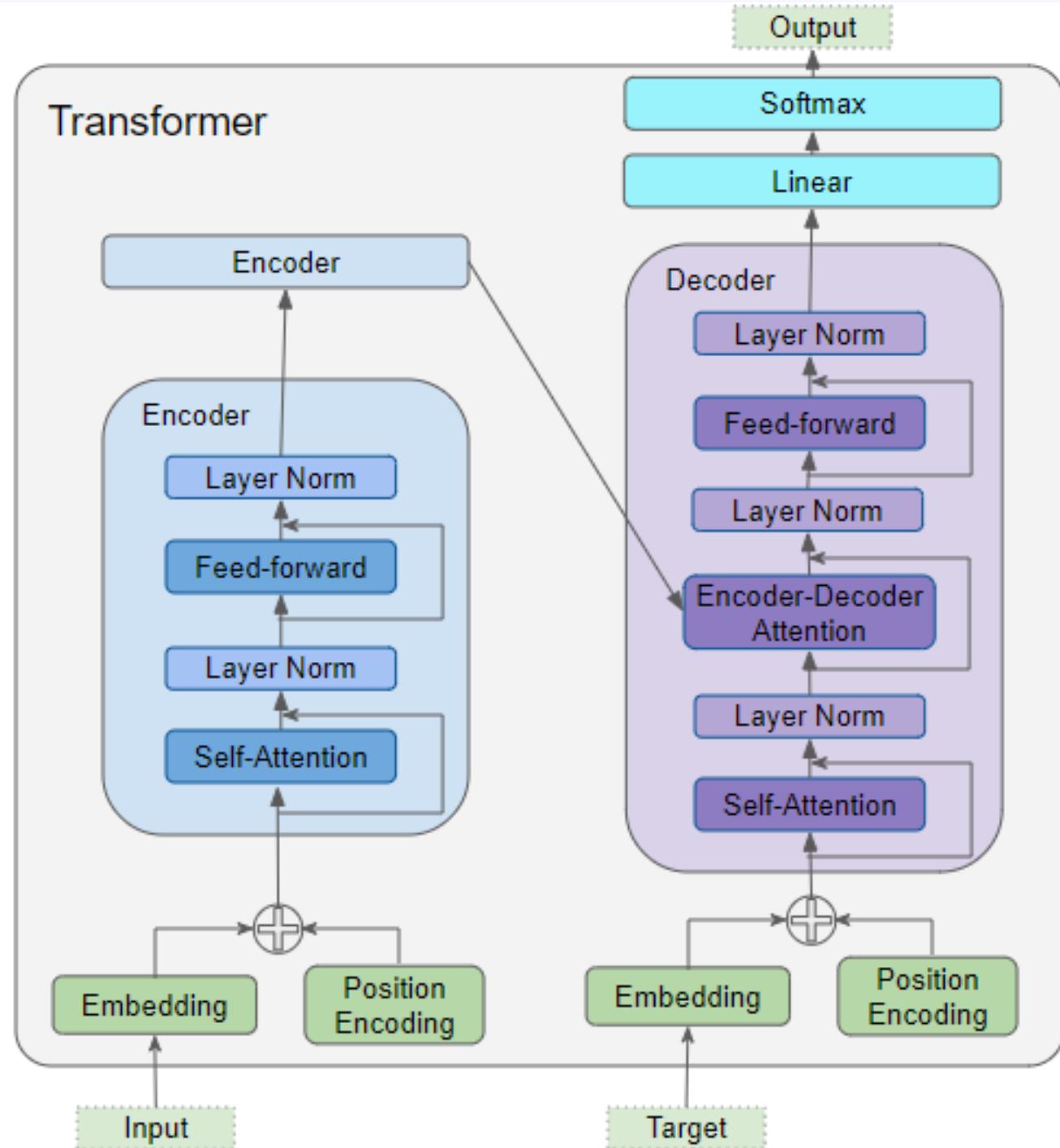


# Transformer: Encoders + Decoders

Structures of an  
Encoder and a  
Decoder

Commonality and  
Difference of encoder  
and decoder

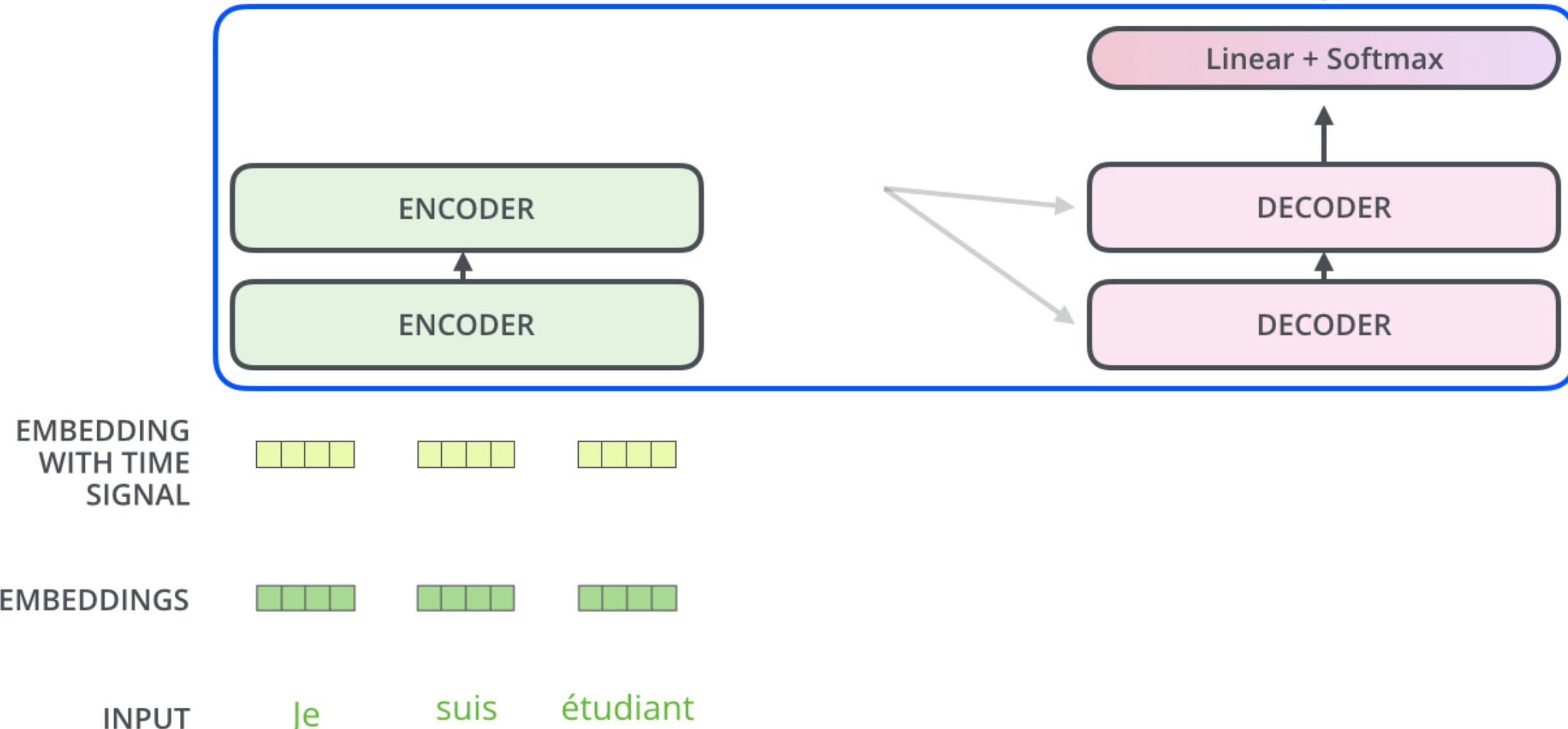
[https:/](https://)



Decoding time step: 1 2 3 4 5 6

OUTPUT

# How the transformer works

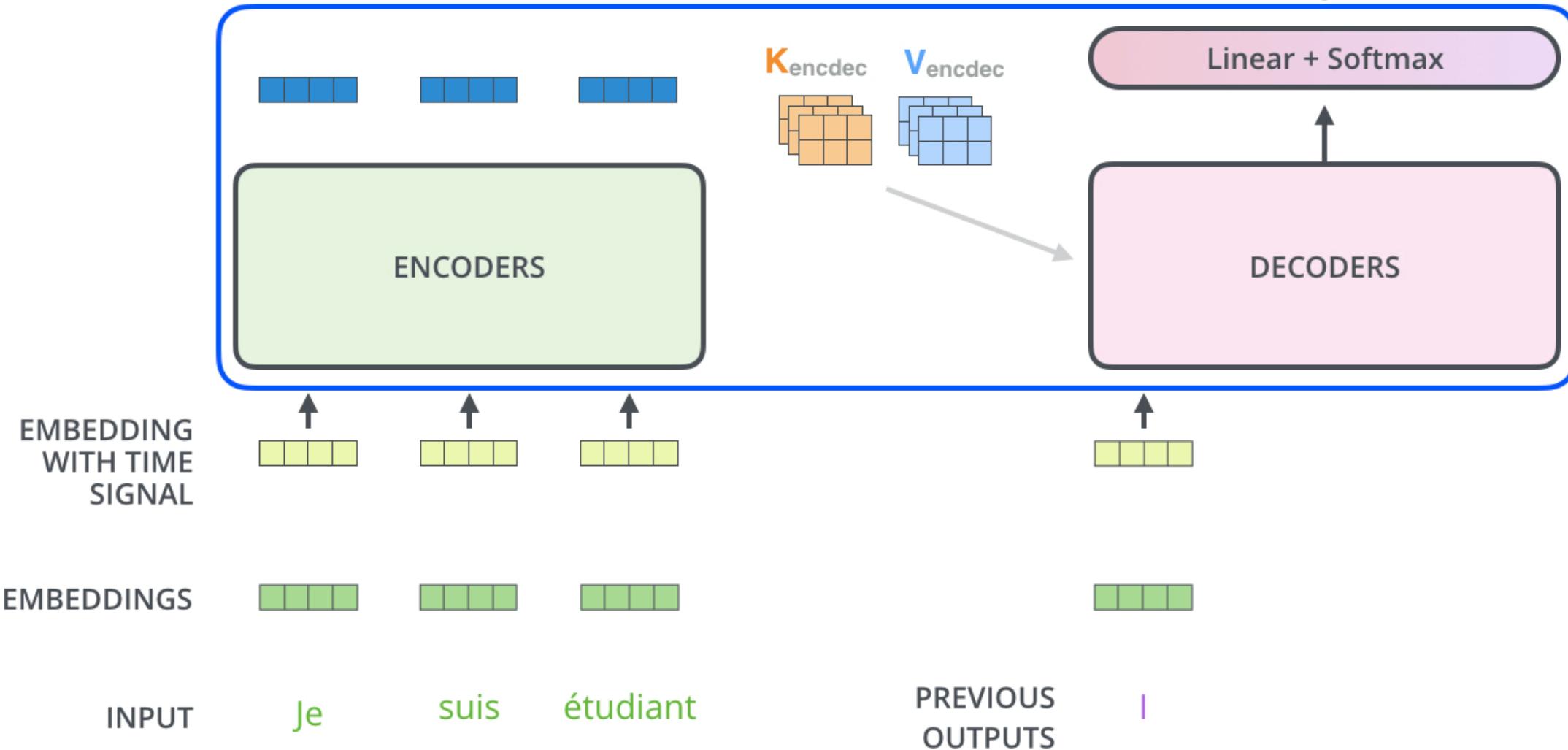


Decoding time step: 1 2 3 4 5 6

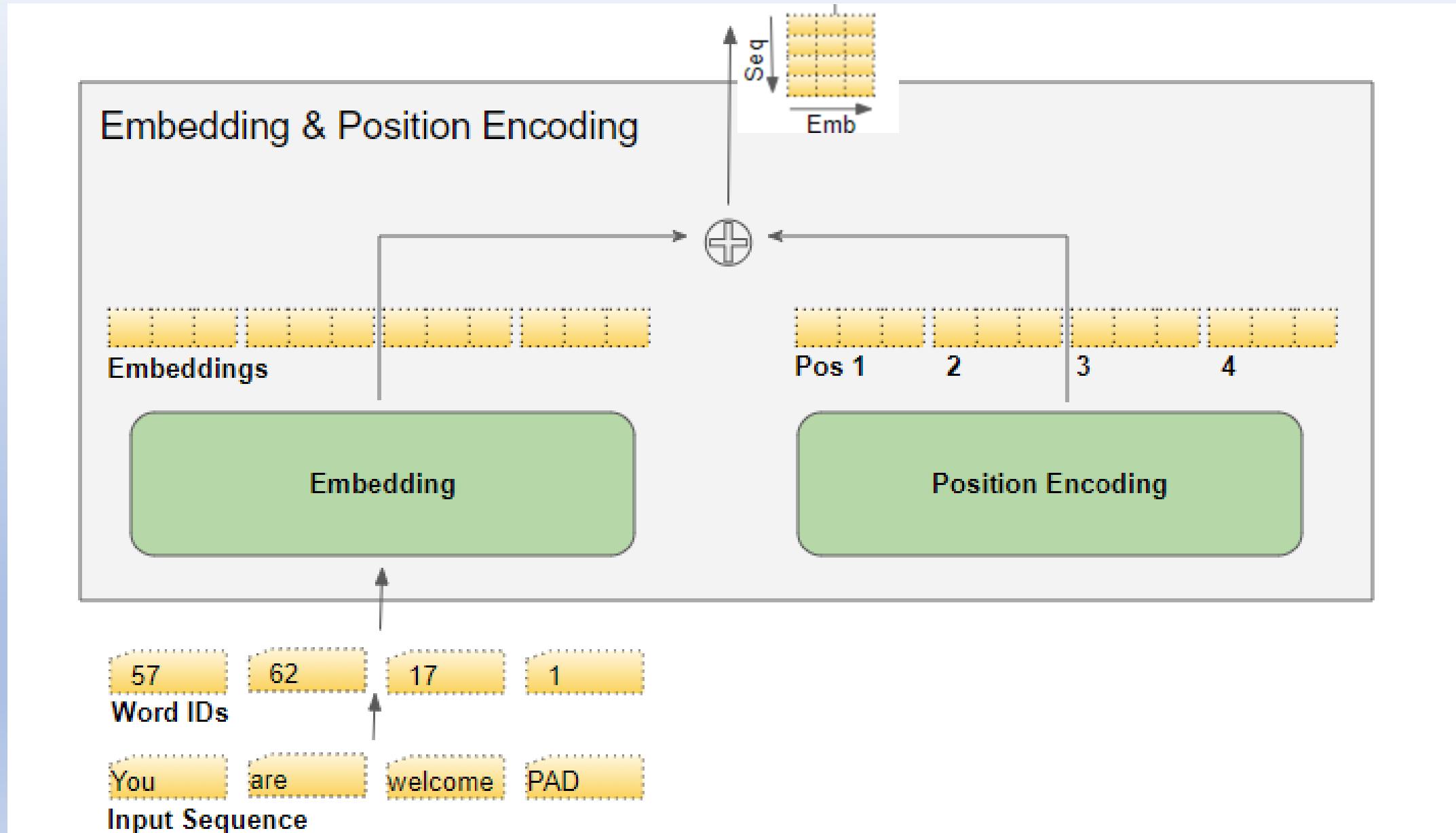
OUTPUT

|

# How the transformer works



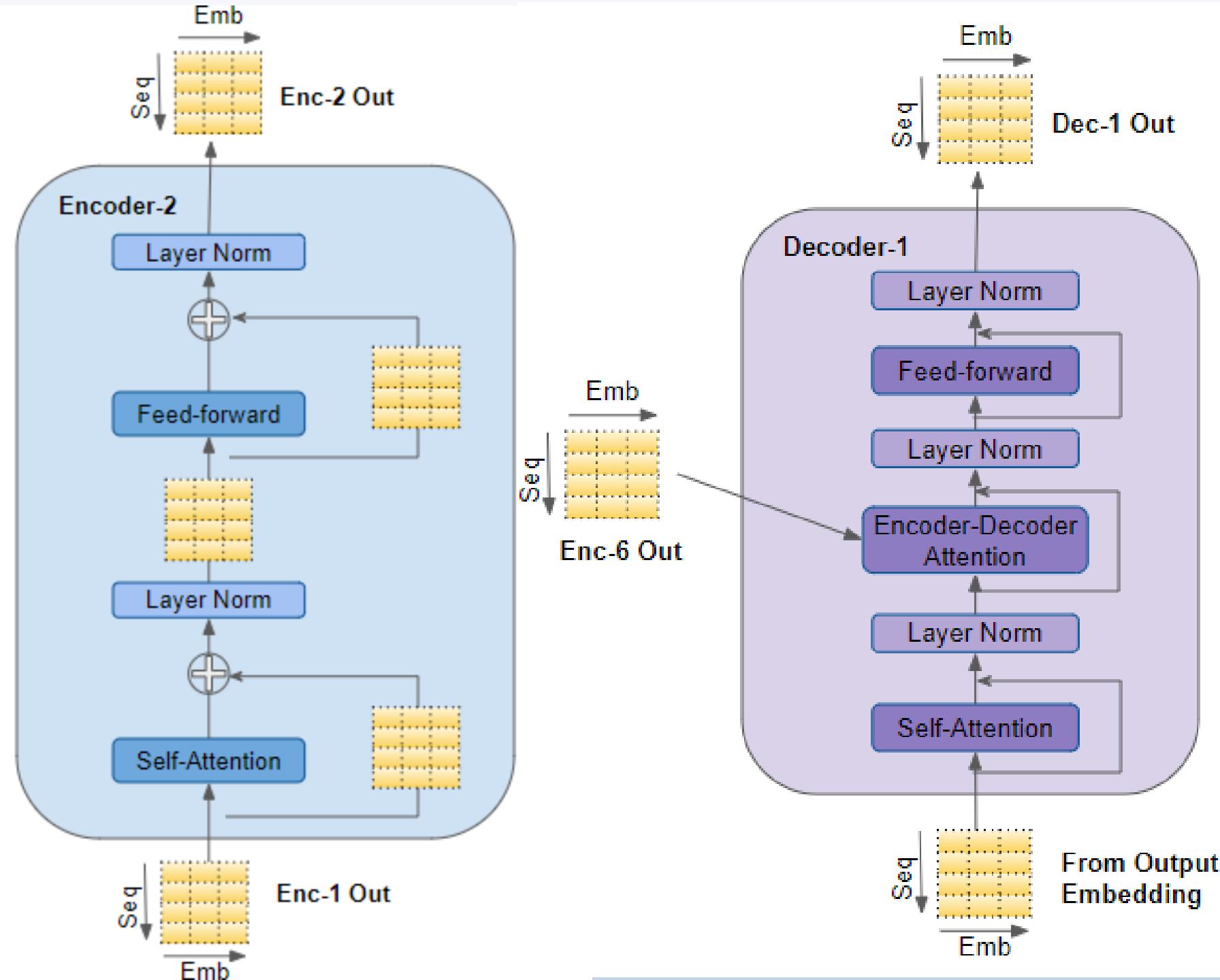
# Embedding each input word/token into a feature vector



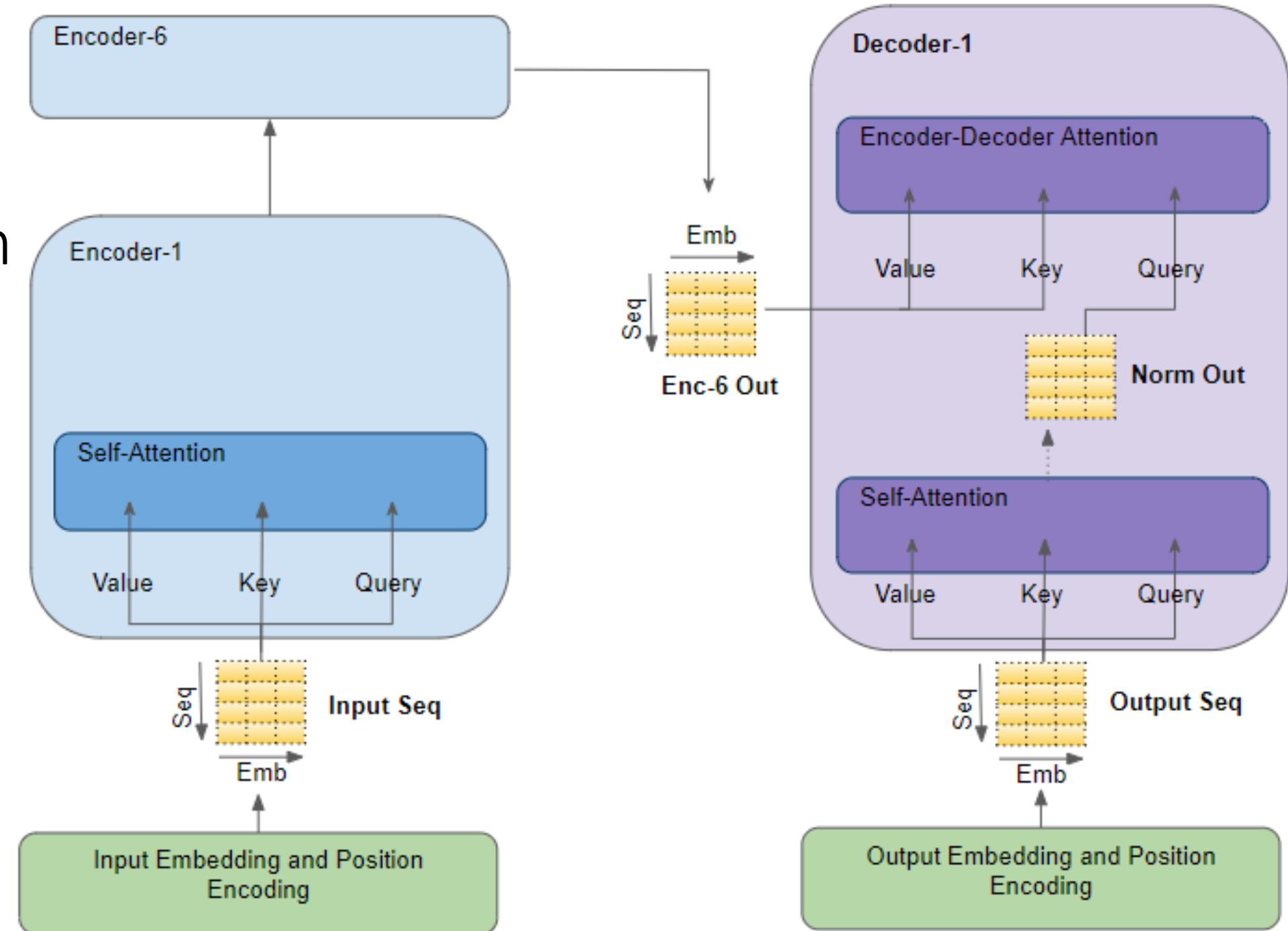
Both the Attention and Feed-forward layers, have a residual skip-connection.

The **pointwise** feed-forward network is a couple of linear layers with a ReLU activation in between.

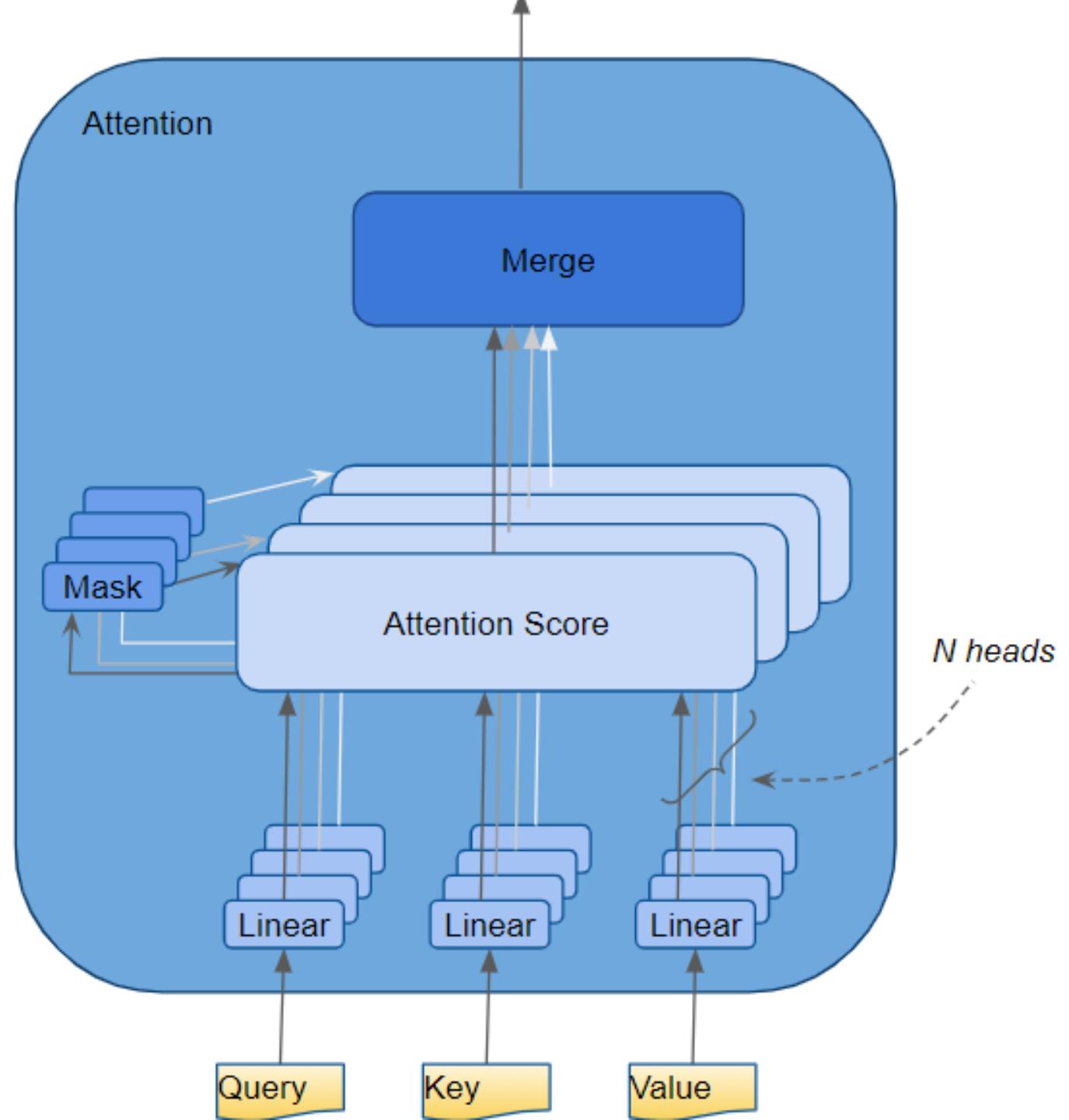
Pointwise?



# Self-attention and Encoder-decoder attention



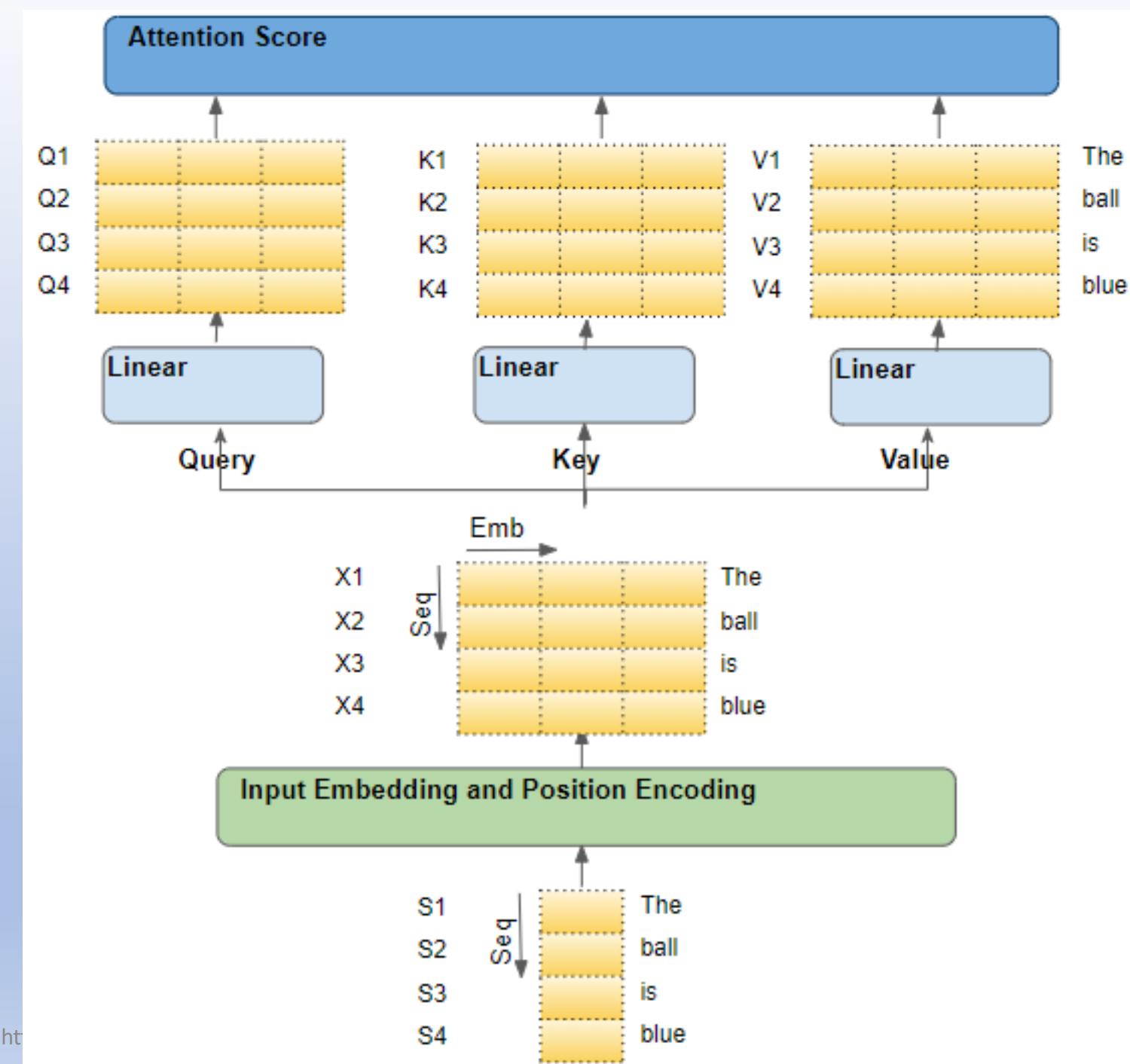
Multiple  
attention heads  
in each encoder  
and decoder.



# Prepare for Attention

An example of English-to-Spanish translation problem, where one sample source sequence is “The ball is blue”. The target sequence is “La bola es azul”.

**Three copies of each word/token are generated for self-attention by linear projection.**



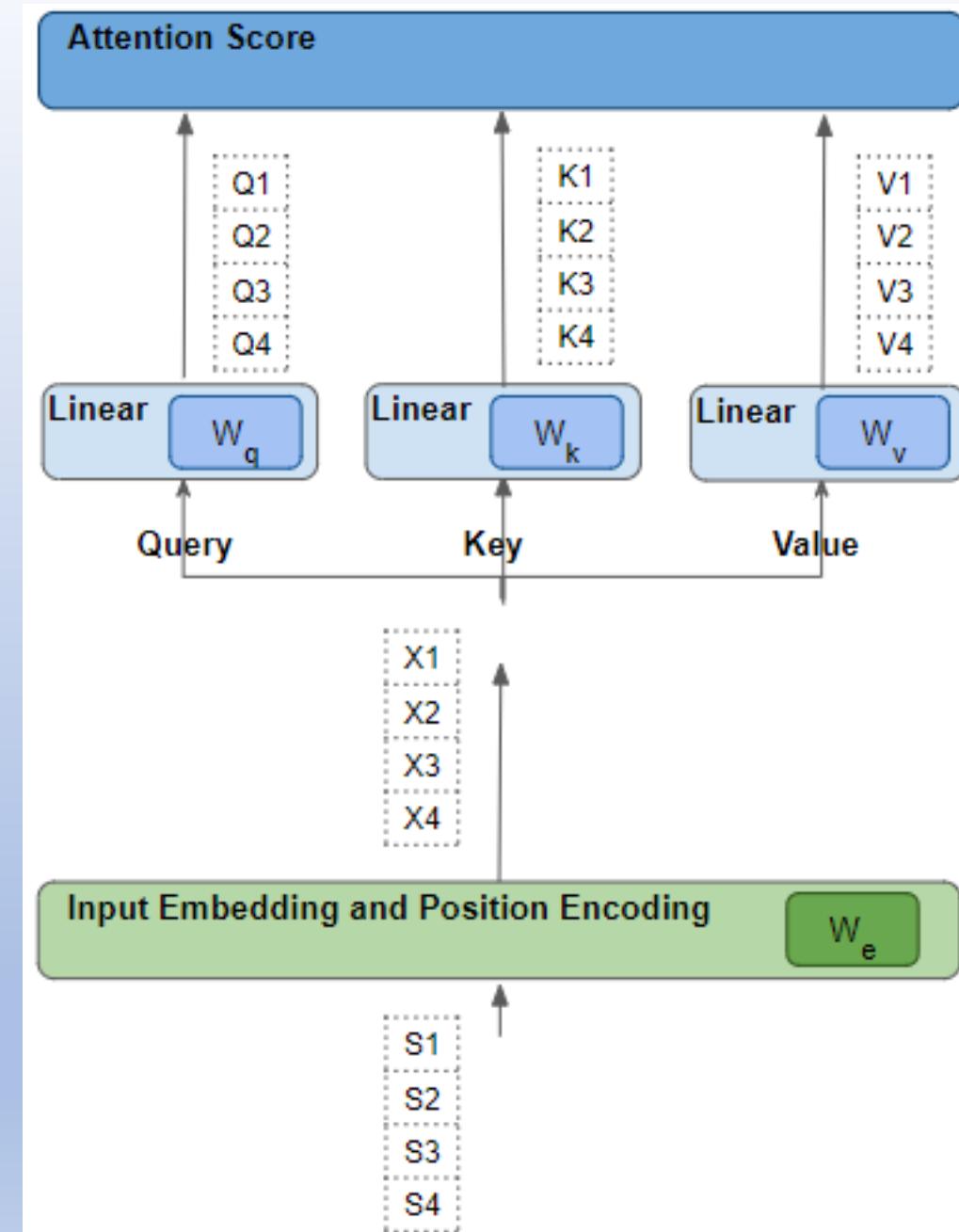
# Learnable linear Projections:

The input sequence (a matrix) is passed through three **trainable** linear layers which produce three separate matrices — known as the Query, Key, and Value.

$$\text{Let } \mathbf{X} = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{pmatrix}, \mathbf{Q} = \begin{pmatrix} Q_1 \\ Q_2 \\ Q_3 \\ Q_4 \end{pmatrix}, \mathbf{K} = \begin{pmatrix} K_1 \\ K_2 \\ K_3 \\ K_4 \end{pmatrix}, \mathbf{V} = \begin{pmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{pmatrix}$$

$$\text{Then: } \mathbf{Q} = \mathbf{X}W_q, \quad \mathbf{K} = \mathbf{X}W_k, \quad \mathbf{V} = \mathbf{X}W_v$$

The important thing to keep in mind is that each ‘row’ of these matrices corresponds to one word (token) in the source sequence.



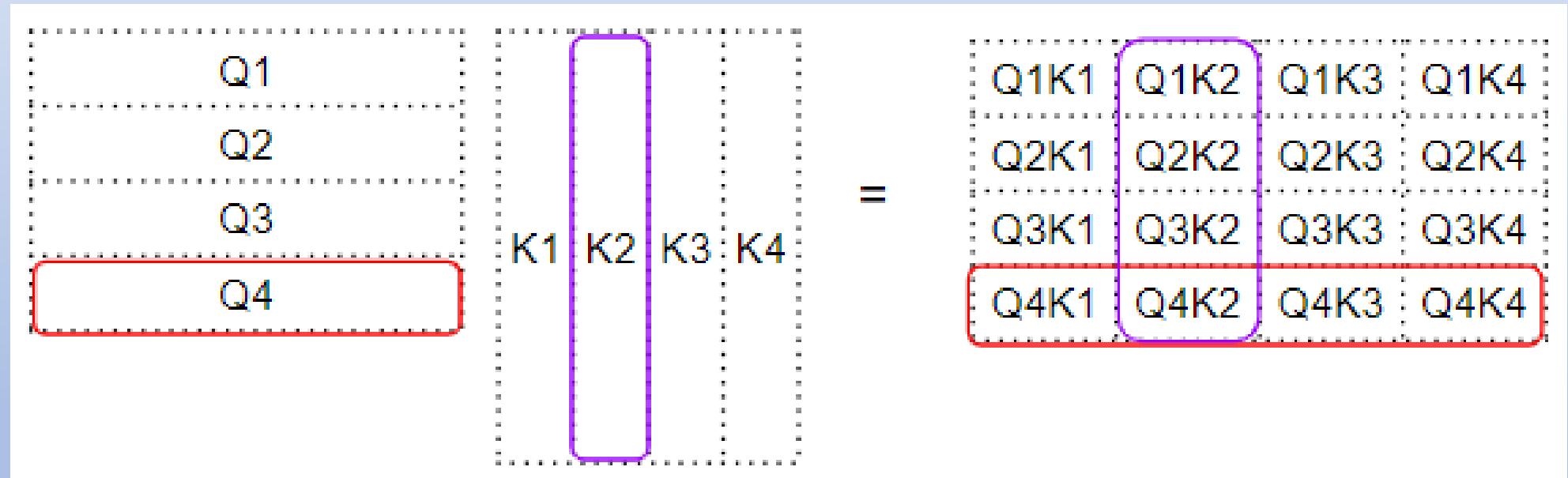
# Attention Score — Dot Product between Q and K words

- The first step of Attention is to do a matrix multiplication (ie. dot product) between the Query (Q) matrix and a transpose of the Key (K) matrix.

$$R = QK^T$$

- Watch what happens to each word. **Dot product generates similarity between words**

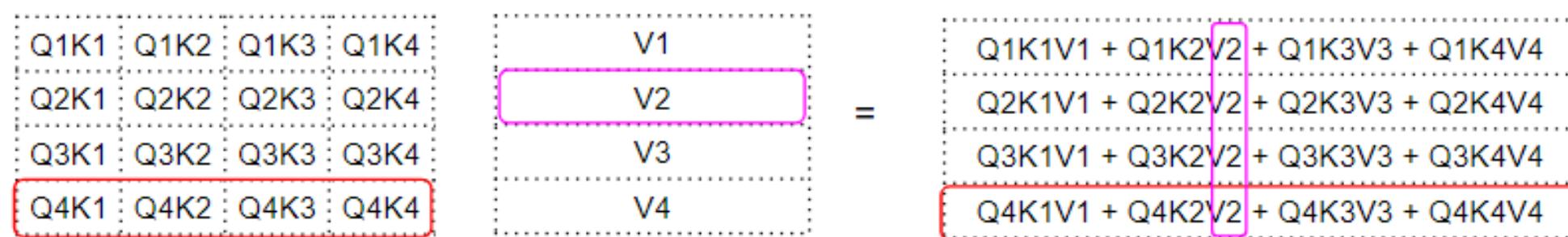
$$R = QK^T$$



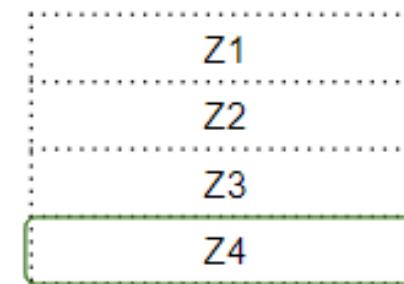
- We produce an intermediate matrix (let's call it a 'factor' matrix) where each cell is a matrix multiplication between two words. **Covariance Matrix!!!**

# Attention: Attended (weighted) combination of Values

The dot product between the Query and Key computes the relevance between each pair of words. This relevance is then used as a “factor” to compute a weighted sum of all the Value words. That weighted sum is output as the Attention module.



$$Z = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V =$$



The diagram illustrates the calculation of the fourth word score (Z<sub>4</sub>) as a weighted sum of values. It shows two paths: one path labeled "Fourth word Score" leading to Z<sub>4</sub>, and another path labeled "Fourth Query word \* first Key word" and "Fourth Query word \* second Key word" both leading to the same term (Q<sub>4</sub>K<sub>1</sub>) in the equation. The equation is:

$$Z_4 = (Q_4 K_1) V_1 + (Q_4 K_2) V_2 + (Q_4 K_3) V_3 + (Q_4 K_4) V_4$$

Self-attention in Encoder – the source sequence pays attention to itself.

Fourth Query word \* second Key word

Attentions is mutual correlation of two tokens

$$Z = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad Q, K, V, Z \in \mathbb{R}^{N \times K}$$

Attention:  $W = QK^T$ :  $(N \times K)(K \times N)$

$\Rightarrow (N \times N)$   
Output:  $Z = WV$ :  $(N \times N)(N \times K)$   
 $\Rightarrow (N \times K)$

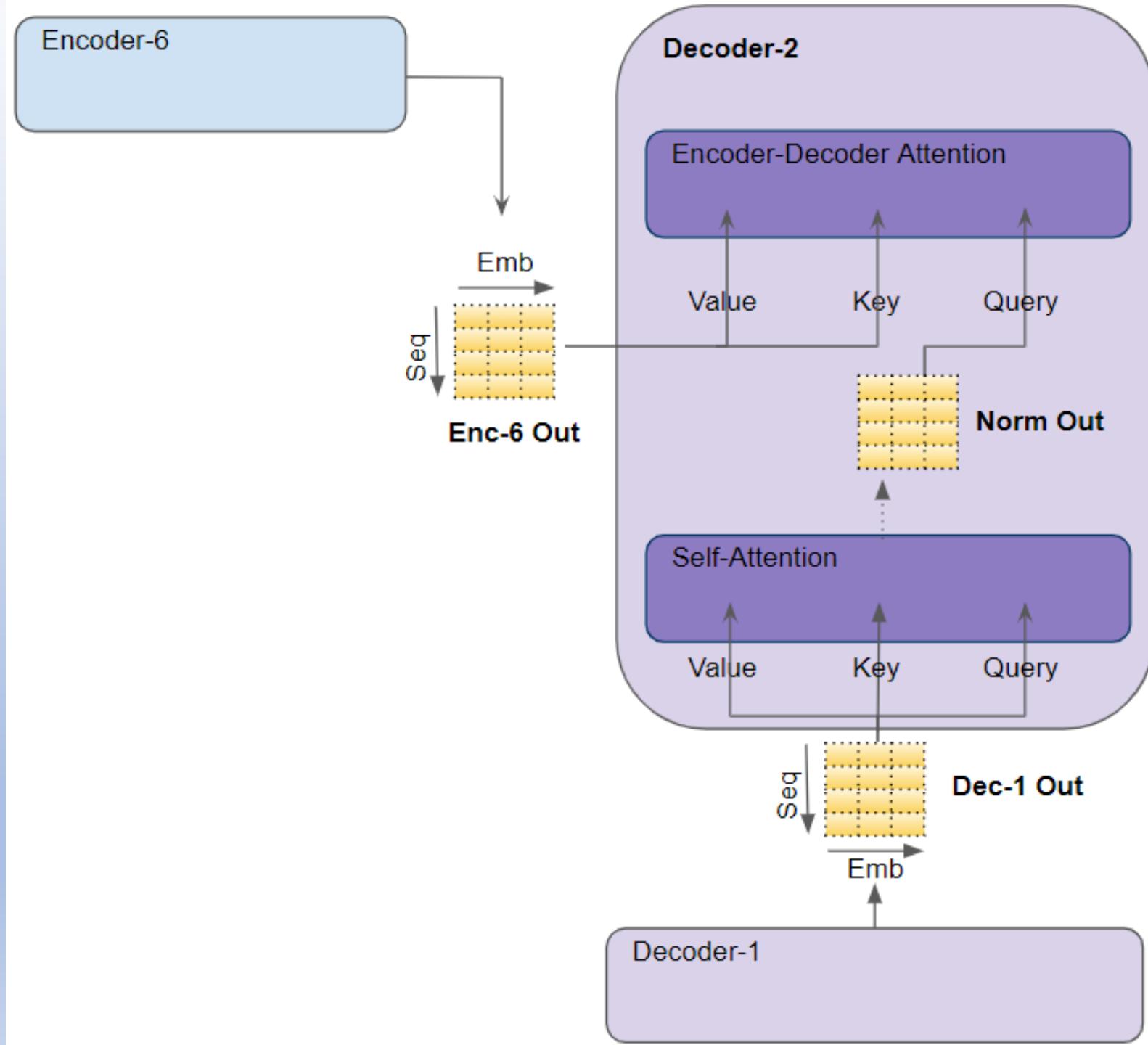
Each output token is **weighted** sum of **all input tokens**, capturing **global** information.

$$Z_4 = (Q_4 K_1) V_1 + (Q_4 K_2) V_2 + (Q_4 K_3) V_3 + (Q_4 K_4) V_4$$

Different from MLP and CNN, weights  $W$  are **not learnt** by **training** data. They are **flexibly** generated by **specific test input**. Attention itself is **not learnable**.

# Encoder-Decoder attention

In the Decoder's Encoder-Decoder attention, the output of the final Encoder in the stack is passed to the Value and Key parameters. The output of the Self-attention module below it is passed to the Query parameter.



La                              bola                              es                              azul

La	$Q1K1V1 + Q1K2V2 + Q1K3V3 + Q1K4V4$
bola	$Q2K1V1 + Q2K2V2 + Q2K3V3 + Q2K4V4$
es	$Q3K1V1 + Q3K2V2 + Q3K3V3 + Q3K4V4$
azul	$Q4K1V1 + Q4K2V2 + Q4K3V3 + Q4K4V4$

### Decoder Self Attention

Target sentence paying attention to itself

Mask is used in decoder  
to exclude words/tokens  
that haven't appeared in  
the target.

Self-attention in the Decoder – the target sequence pays attention to itself.

Encoder-Decoder-attention in the Decoder – the target sequence pays attention to the source sequence

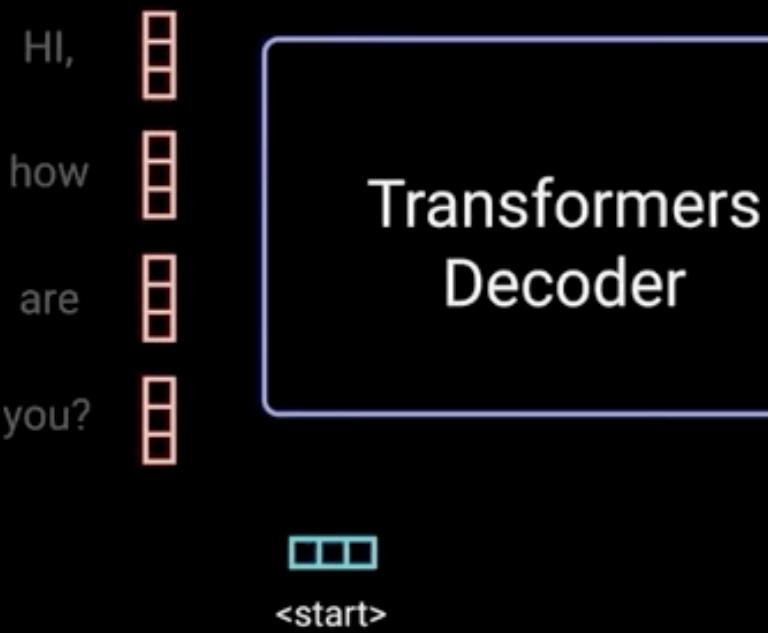
The	ball	is	blue
La	$Q1K1V1 + Q1K2V2 + Q1K3V3 + Q1K4V4$		
bola	$Q2K1V1 + Q2K2V2 + Q2K3V3 + Q2K4V4$		
es	$Q3K1V1 + Q3K2V2 + Q3K3V3 + Q3K4V4$		
azul	$Q4K1V1 + Q4K2V2 + Q4K3V3 + Q4K4V4$		

Query word "azul" that  
is paying attention

### Encoder-Decoder Attention

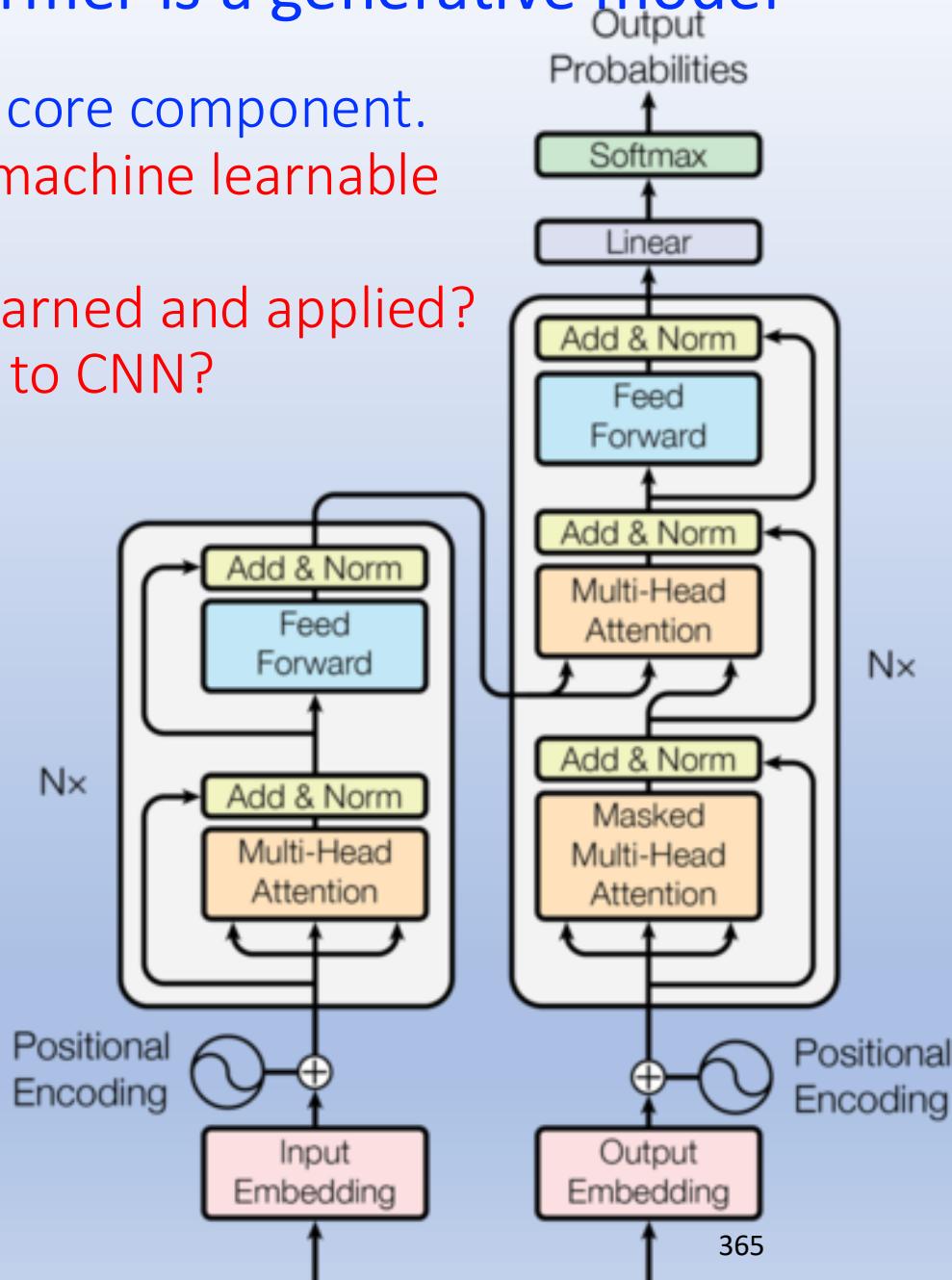
Target sentence paying attention to source  
sentence

Inputs: Hi, How are you? Output: I am fine. Transformer is a generative model



$$\mathbf{X} = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{pmatrix}, \quad \mathbf{Q} = \mathbf{XW}_q, \mathbf{K} = \mathbf{XW}_k, \mathbf{V} = \mathbf{XW}_v$$
$$\mathbf{Y} = f(f(f(\mathbf{XW}_1)\mathbf{W}_2)\mathbf{W}_3).$$

Attention is the core component.  
Where are the machine learnable parameters?  
How are they learned and applied?  
Any connection to CNN?



# Learnable Linear Projections

Input (a matrix) is passed through **trainable** linear layers to produce 3 separate matrices — Query, Key, and Value.

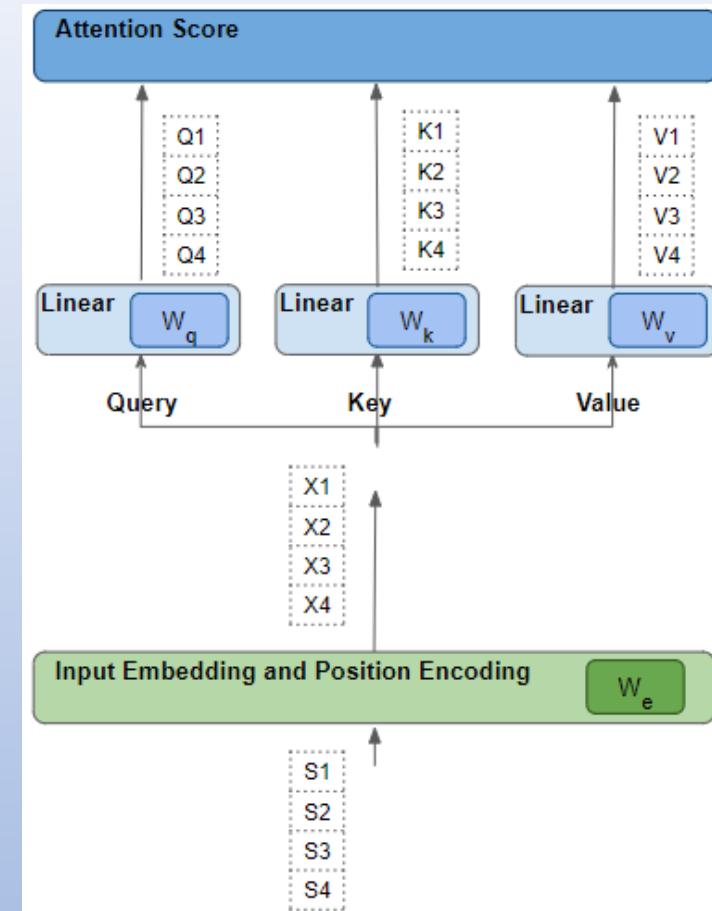
$$\mathbf{Q} = \mathbf{X}\mathbf{W}_q, \quad \mathbf{K} = \mathbf{X}\mathbf{W}_k, \quad \mathbf{V} = \mathbf{X}\mathbf{W}_v$$

$$\mathbf{X} = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{pmatrix}, \quad \mathbf{Q} = \begin{pmatrix} Q_1 \\ Q_2 \\ Q_3 \\ Q_4 \end{pmatrix} \xrightarrow{(N \times K)(K \times K)} \begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{pmatrix}, \quad \mathbf{K} = \begin{pmatrix} K_1 \\ K_2 \\ K_3 \\ K_4 \end{pmatrix}, \quad \mathbf{V} = \begin{pmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{pmatrix}$$

Each ‘row’  $X_i$  of these matrices represents one token.

$$Q_i = X_i \mathbf{W}_q, \quad K_i = X_i \mathbf{W}_k, \quad V_i = X_i \mathbf{W}_v, \quad \text{for } i = 1, 2, 3, \dots, N$$

All tokens share the same learnable parameter: any single learnable parameter goes through all tokens or is learned from all tokens. **Convolution of CNN!**



# Learnable feed forward networks

Similar to linear projection  $\mathbf{Q} = \mathbf{X}\mathbf{W}_q$ ,  $\mathbf{K} = \mathbf{X}\mathbf{W}_k$ ,  $\mathbf{V} = \mathbf{X}\mathbf{W}_v$ , feed-forward network:

$$\mathbf{Y} = h(h(h(\mathbf{X}\mathbf{W}_1)\mathbf{W}_2)\mathbf{W}_3).$$

$$(N \times K)(K \times K) \Rightarrow (N \times K)$$

$$\mathbf{X} = \begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ X_4 \end{pmatrix}, \mathbf{Y} = \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{pmatrix}$$

Each ‘row’  $X_i$  and  $Y_i$  of matrices corresponds to one token.

$$Y_i = h(h(h(X_i\mathbf{W}_1)\mathbf{W}_2)\mathbf{W}_3), \quad \text{for } i = 1, 2, 3, \dots, N$$

All tokens in inputs/outputs share same learnable parameter.

Feed-forward network of the transformer is Pointwise, which is not MLP.

Pointwise means that each single learnable parameter goes through all inputs. It comes again from the key concept, convolution of CNN.

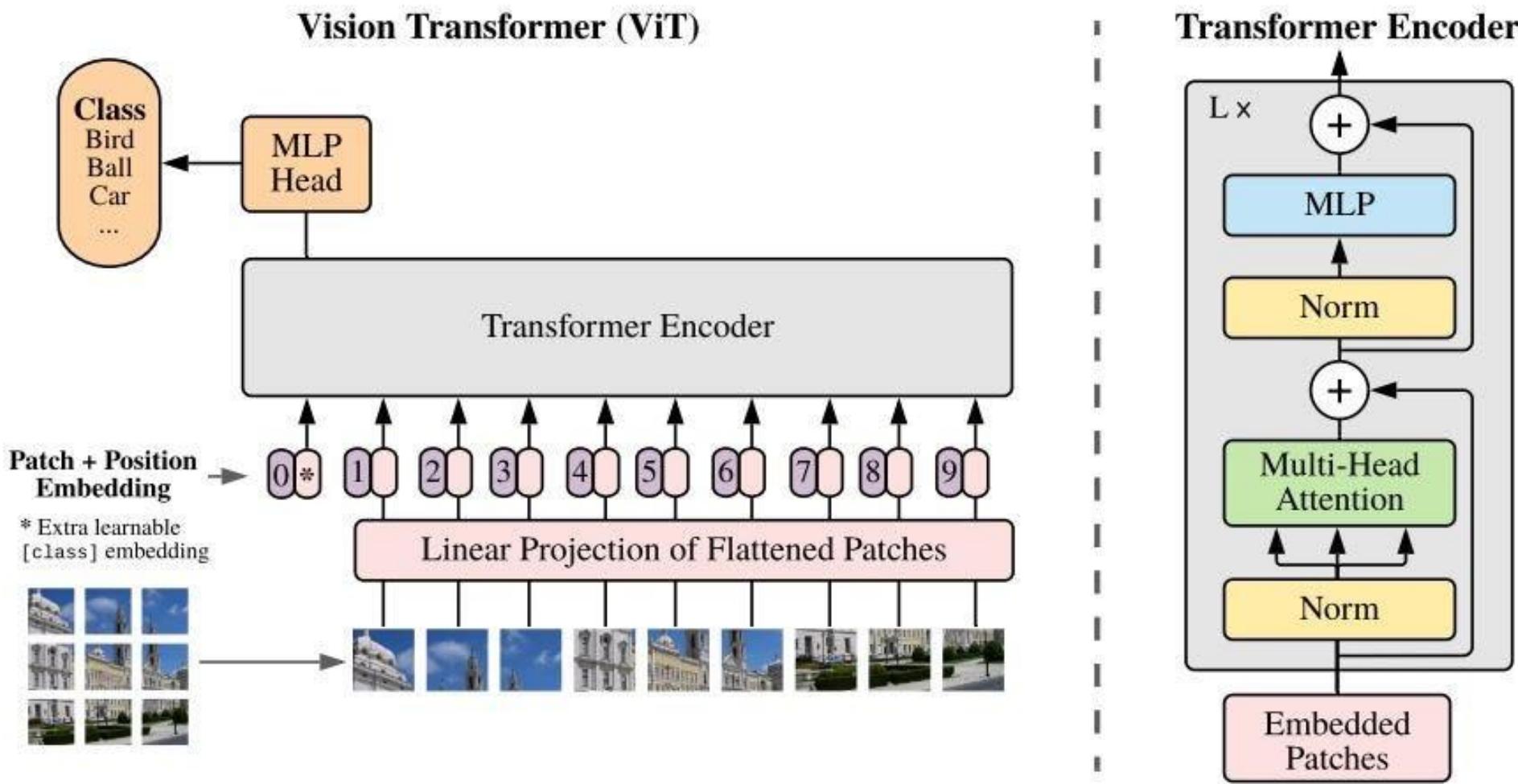


Figure 1: Model overview. We split an image into fixed-size patches, linearly embed each of them, add position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder. In order to perform classification, we use the standard approach of adding an extra learnable “classification token” to the sequence. The illustration of the Transformer encoder was inspired by Vaswani et al. (2017).

# Vision Transformer vs CNN in Computer Vision

- The Vision Transformer (ViT) outperforms state-of-the-art convolutional networks in multiple benchmarks of computer vision while requiring fewer computational resources to train, after being pre-trained on **large amounts of data**.
- While CNNs have a proven track record in various computer vision tasks and handle large-scale datasets efficiently, Vision Transformers offer advantages in scenarios where global(?) dependencies and contextual understanding are crucial.

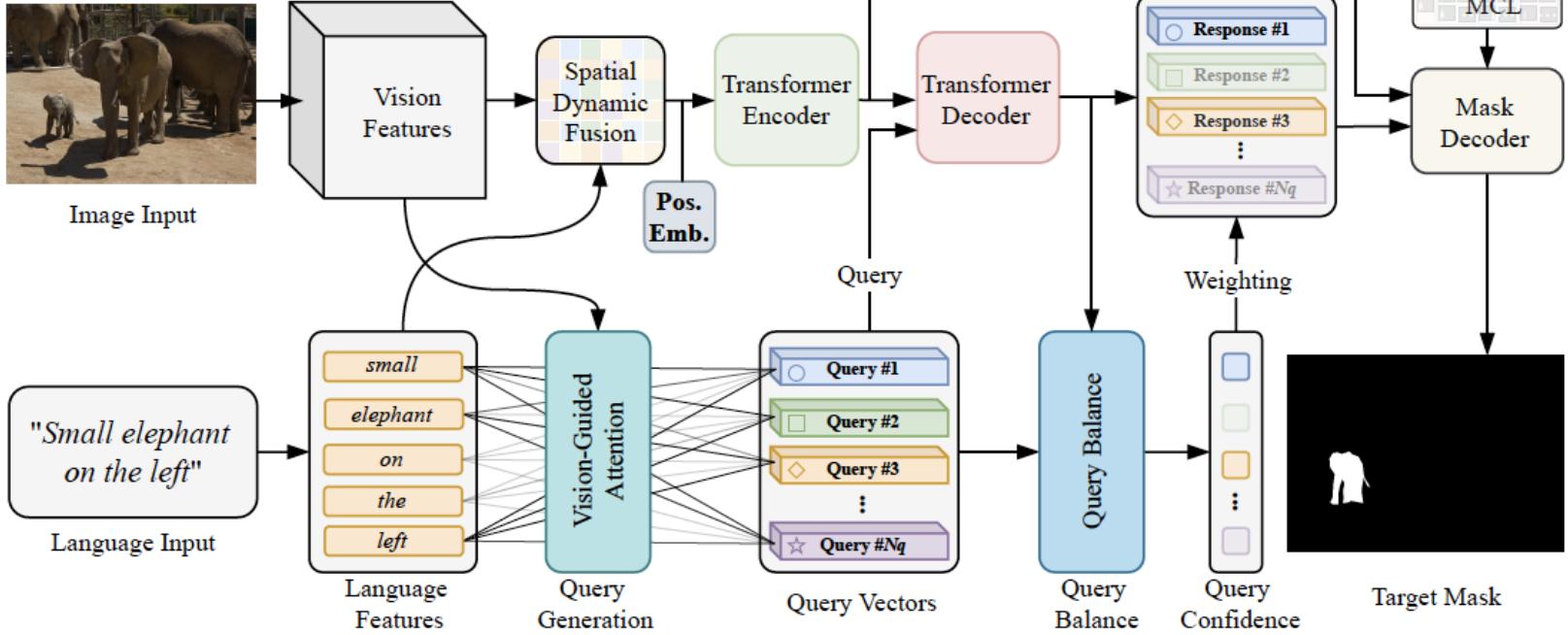


Fig. 2: The overview architecture of the proposed Vision-Language Transformer (VLT). Firstly, the given image and language expression are projected into visual and linguistic feature spaces, respectively. A Spatial Dynamic Fusion module is then employed to fuse vision and language features, generating multi-modal feature inputted to the transformer encoder. The proposed Query Generation Module generates a set of input-specific queries according to the vision and language features. These input-specific queries are sent to the decoder, producing corresponding query responses. These resulting responses are selected by the Query Balance Module and then decoded to output the target mask by a Mask Decoder. “Pos. Emb.”: Positional Embeddings. “MCL”: Masked Contrastive Learning.

H. Ding, C. Liu, S. Wang, X. Jiang, “[Vision-Language Transformer and Query Generation for Referring Segmentation](#),” *International Conference on Computer Vision (ICCV’21)*, Oct 2021.

H. Ding, C. Liu, S. Wang, X. Jiang, “[VLT: Vision-Language Transformer and Query Generation for Referring Segmentation](#),” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 6, 7900–7916, Jun. 2023.  
<https://personal.ntu.edu.sg/exdjiang/>

# Vision-Language Transformer

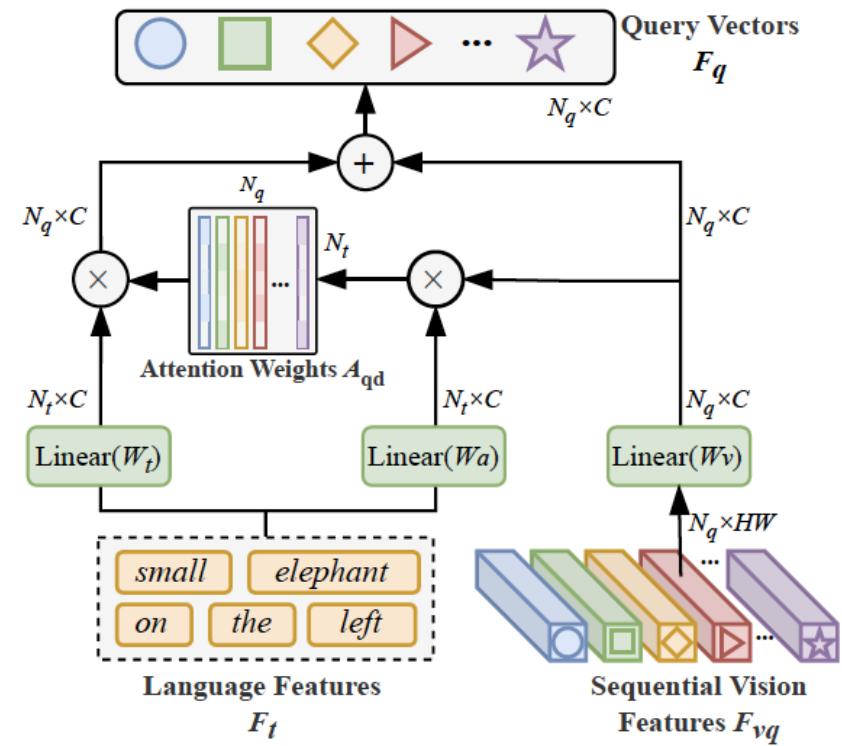


Fig. 6: Query Generation Module (QGM). The QGM takes sequential vision feature  $F_{vq}$  and language features  $F_t$  as inputs and generates a group of input-specific query vectors  $F_q$ , which are then sent to the transformer decoder of our VLT.

# Conclusion of Deep Learning

- The success of AI lies in Machine Learning
- The success of Machine Learning lies in learning general rules, not overfitting to specific details of training data.
- This is realized by learning in convolution with minimum filter size => CNN
- Hence, many convolutional layers are required by capturing large area of information => Deep learning
- Deep learning is facilitated by the simplest nonlinear activation function ReLu.
- Every single learned parameter of CNN going through all inputs **greatly** enhances the learning **generalization**. It captures information by weights learned from training data, however, **limits its capability**.
- **Transformer** captures global information by **non-learnable** attention modules. Different from MLP and CNN, the way to capture the global information is not learned from the training data set. It is specified by the specific test input.
- All learnable modules of transformer exploits the concept of **convolution!**