The rise of AI

Eric Postma
CS&AI
Tilburg University

Cognitive Science & AI

- Department at Tilburg School of Humanities and Digital Sciences
- Responsible for Cognitive Science & AI program
- Research focusses on computational methods for text analysis, signal analysis, image analysis, and so forth

Deep learning algorithm does as well as dermatologists in identifying skin cancer

Date: January 25, 2017

Source: Stanford University

Summary: In hopes of creating better access to medical care, researchers have trained an algorithm to diagnose skin cancer.

Share:

RELATED TOPICS

Health & Medicine

- Skin Care
- Skin Cancer
- Psoriasis

Computers & Math

- Computer Programming
- Computers and Internet
- Artificial Intelligence

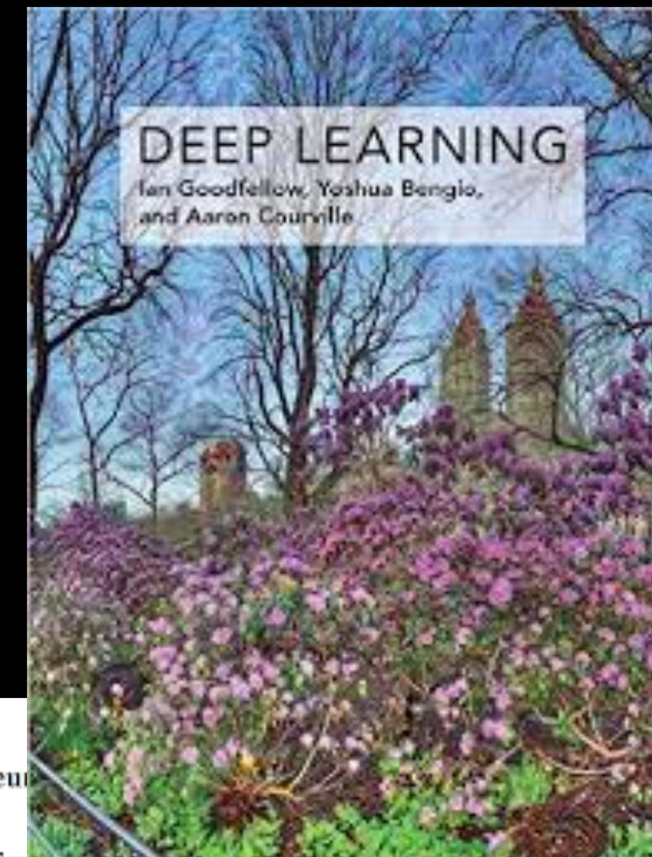
RELATED TERMS

- Stem cell treatments

FULL STORY

A dermatologist using a dermatoscope, a type of handheld microscope, to look at skin. Computer scientists at Stanford have created an artificially intelligent diagnosis algorithm for skin cancer that matched the performance of board-certified

Literature



- http://www.deeplearningbook.org/
- arXiv server (https://arxiv.org/)

Fast Algorithms for Convolutional Neu

Andrew Lavin
alavin@acm.org

Scott Gray
Nervana Systems
sgray@nervanasys.com

arXiv:1509.09308v2 [cs.NE] 10 Nov 2015

Abstract

Deep convolutional neural networks take GPU days of compute time to train on large data sets. Pedestrian detection for self driving cars requires very low latency. Image recognition for mobile phones is constrained by limited processing resources. The success of convolutional neural networks in these situations is limited by how fast we can compute them. Conventional FFT based convolution is fast for large filters, but state of the art convolutional neural networks use small, 3 × 3 filters. We introduce a new class of fast algorithms for convolutional neural networks using Winograd's minimal filtering algorithms. The algorithms compute minimal complexity convolution over small tiles, which makes them fast with small filters and small batch sizes. We benchmark a GPU implementation of our algorithm with the VGG network and show state of the art throughput at batch sizes from 1 to 64.

1. Introduction

Deep convolutional neural networks (convnets) achieve state of the art results on image recognition problems [11][7]. The networks take several days of GPU time to train and require significant compute resources during classification as well. Larger data sets and models lead to better accuracy but also increase computation time. Therefore progress in deep neural networks is limited by how fast

cause they achieve better accuracy with fewer weights than shallow networks with larger filters [11, 7].

Therefore there is a strong need for fast convnet algorithms for small batch sizes and small filters. However conventional convnet libraries require large batch sizes and large filters for fast operation.

This paper introduces a new class of fast algorithms for convolutional neural networks based on the minimal filtering algorithms pioneered by Winograd [13]. The algorithms can reduce the arithmetic complexity of a convnet layer by up to a factor of 4 compared to direct convolution. Almost all of the arithmetic is performed by dense matrix multiplies of sufficient dimensions to be computed efficiently, even when the batch size is very small. The memory requirements are also light compared to the conventional FFT convolution algorithm. These factors make practical implementations possible. Our implementation for NVIDIA Maxwell GPUs achieves state of the art throughput for all batch sizes measured, from 1 to 64, while using at most 16MB of workspace memory.

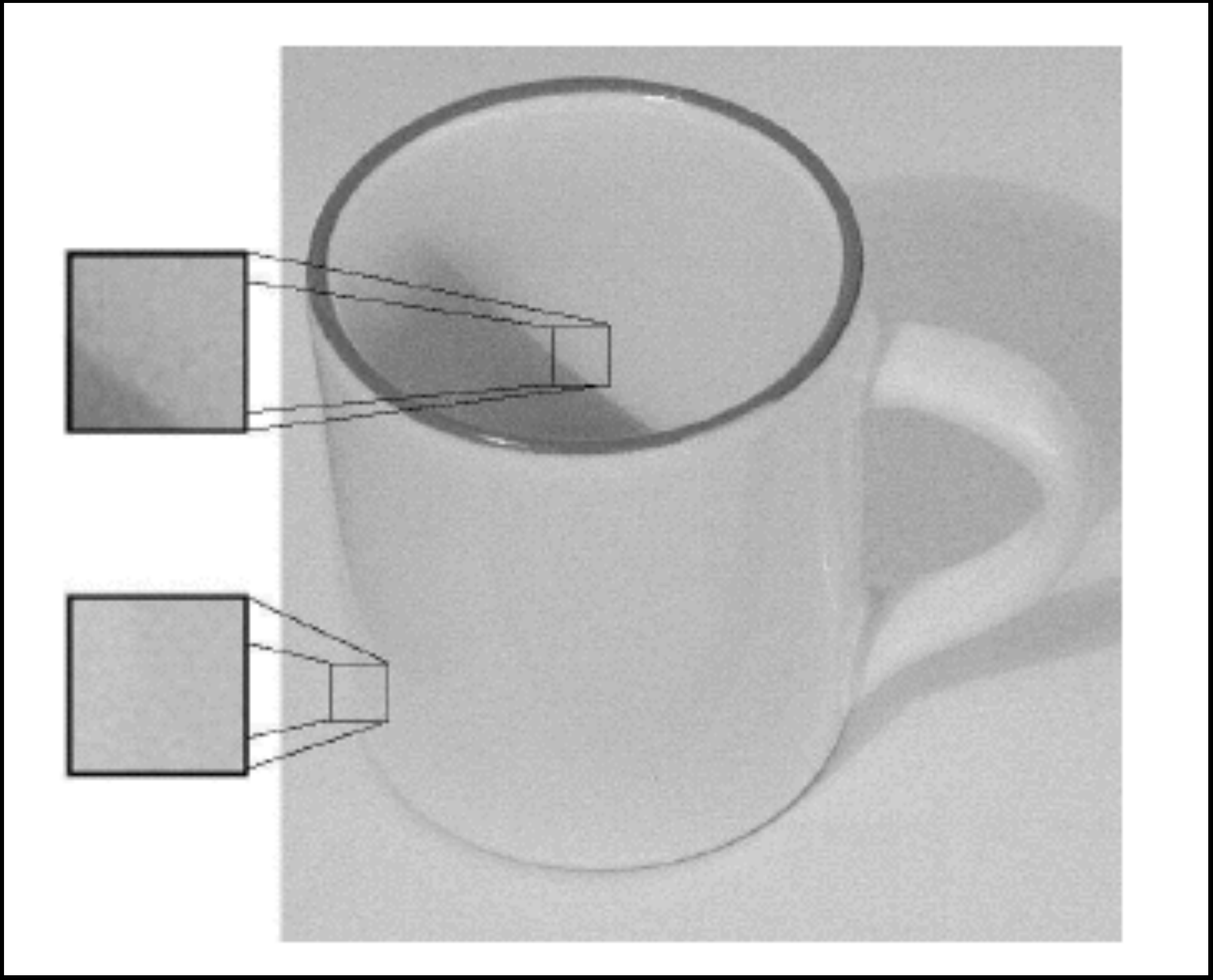
2. Related Work

The FFT and convolution theorem have been used to reduce the arithmetic complexity of convnet layers, first by Mathieu *et al.* [10], then refined by Visalache *et al.* [12], and then implemented in the NVIDIA cuDNN library [1].

Today

- Deep Learning essentials
- Acquiring intuition about convolutional neural networks
- Exploring deep neural networks
- Exploring convolution
- Exploring convolutional neural networks

Local Processing (visual "features")



Filters

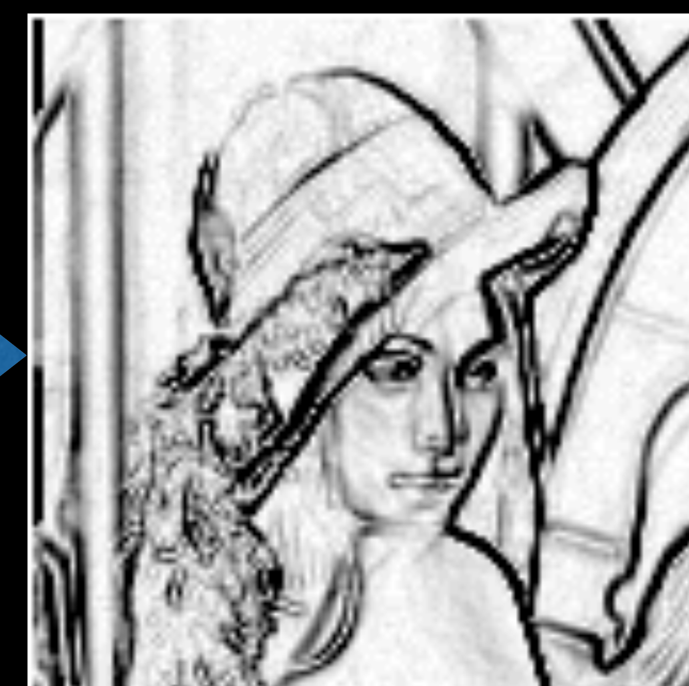
-1	0	+1
-2	0	+2
-1	0	+1

Gx

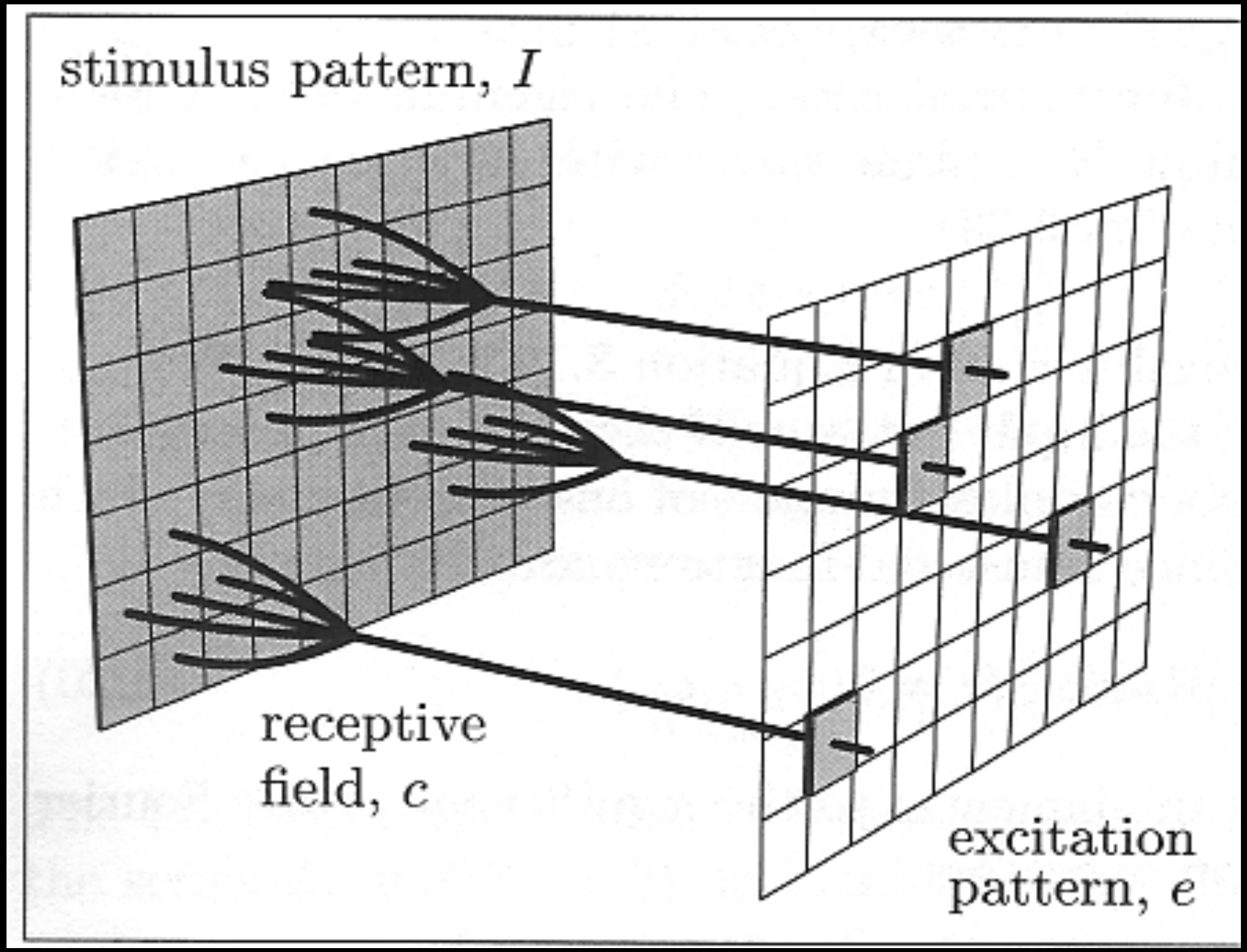
+1	+2	+1
0	0	0
-1	-2	-1

Gy

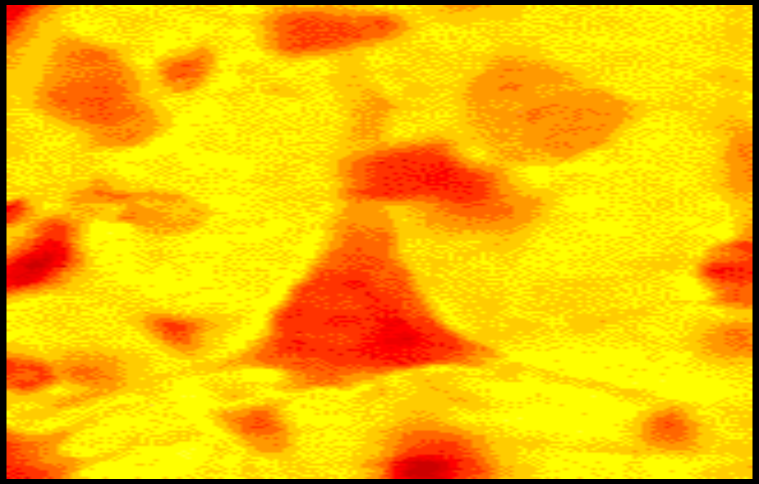
convolution

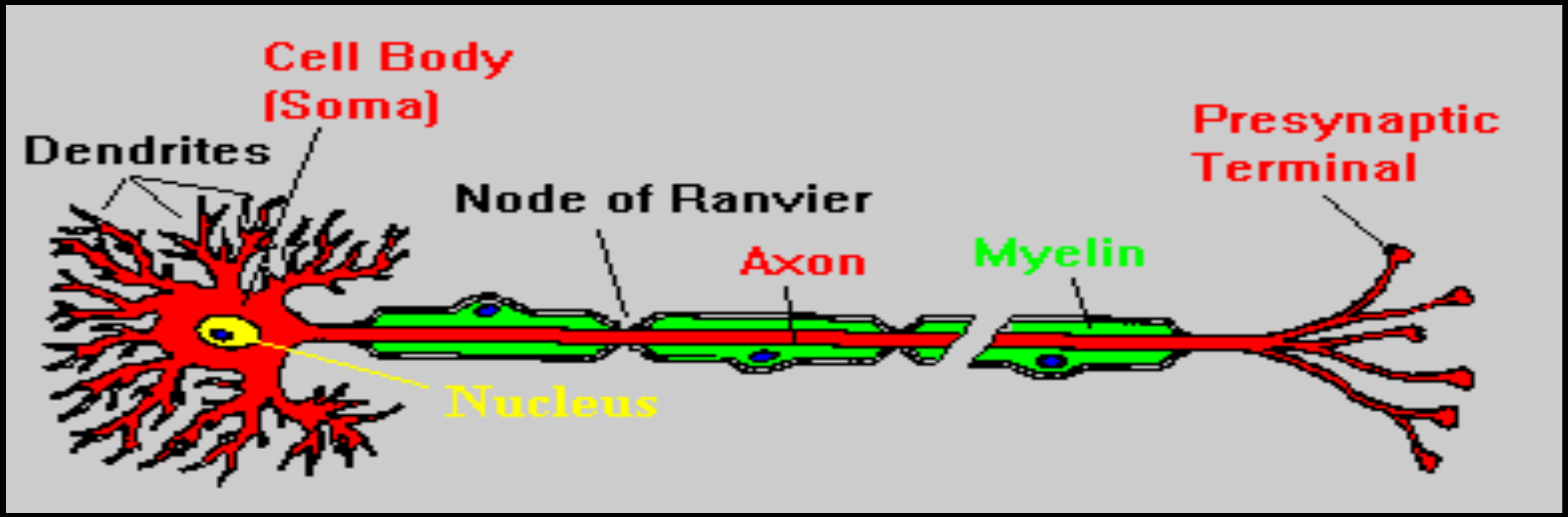


Filtering in Neural Networks

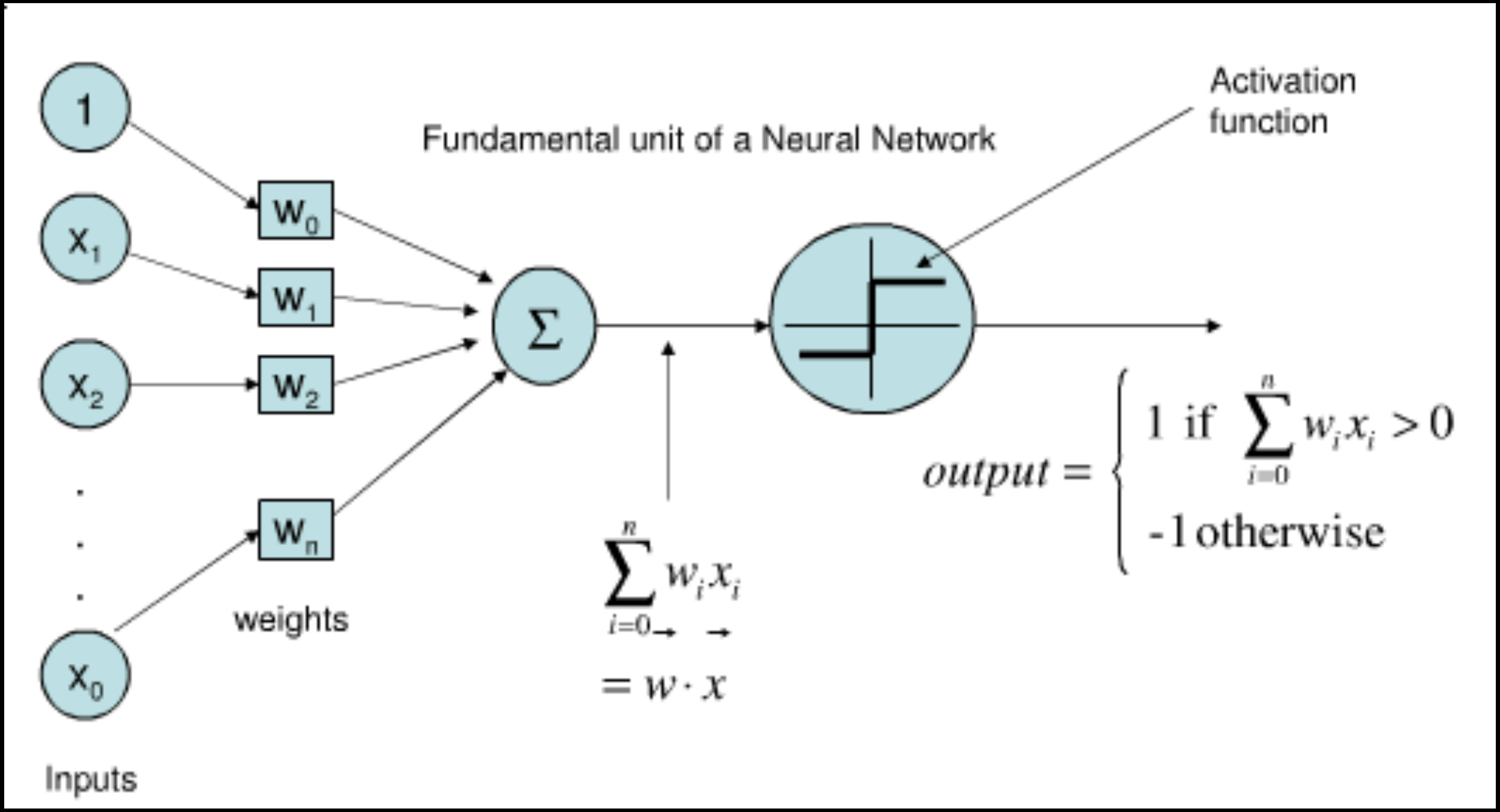


A brief history of neural networks



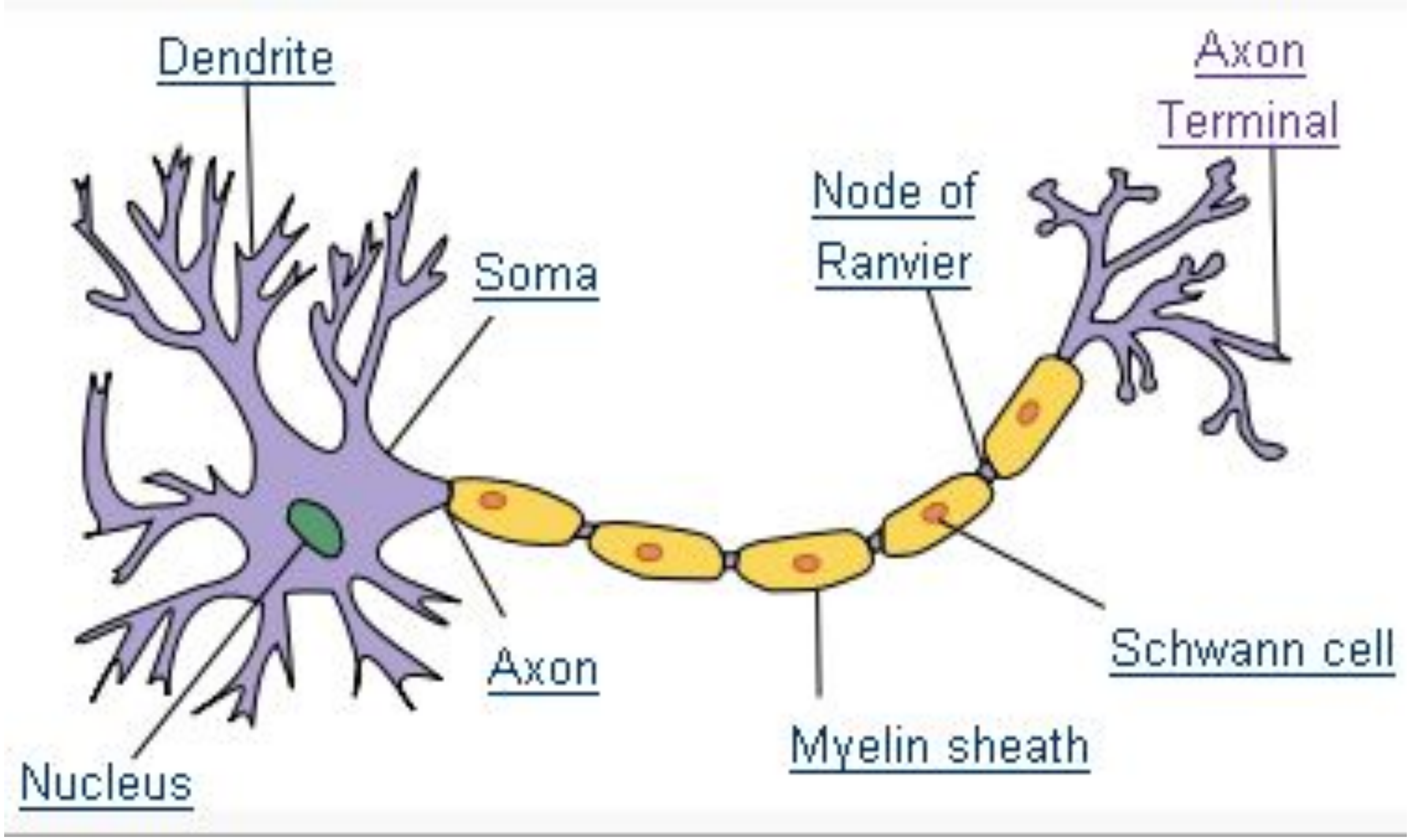


Perceptron (1957)

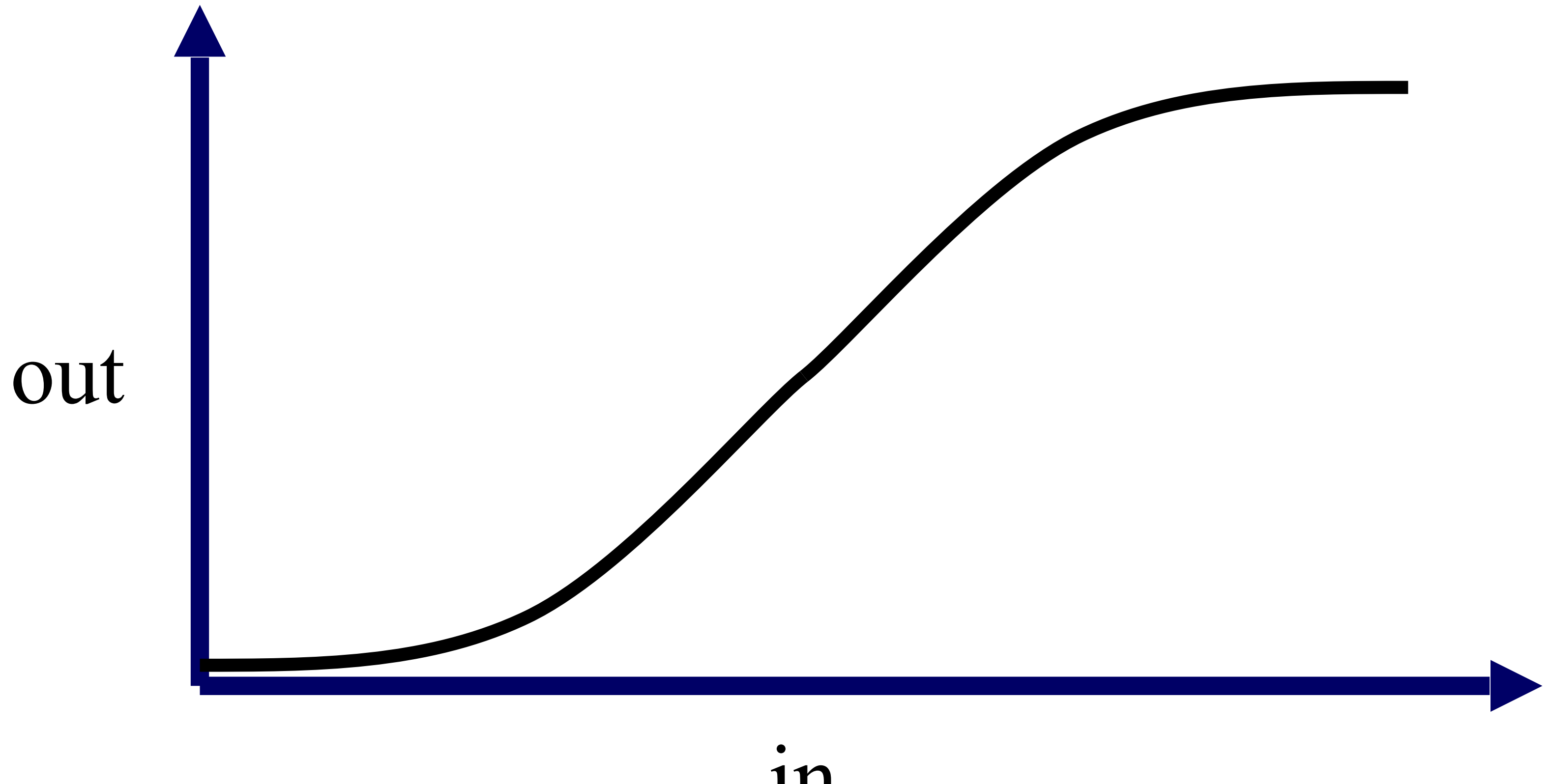


Automatically Learns Linear Mappings

Neurons, the building blocks of the brain

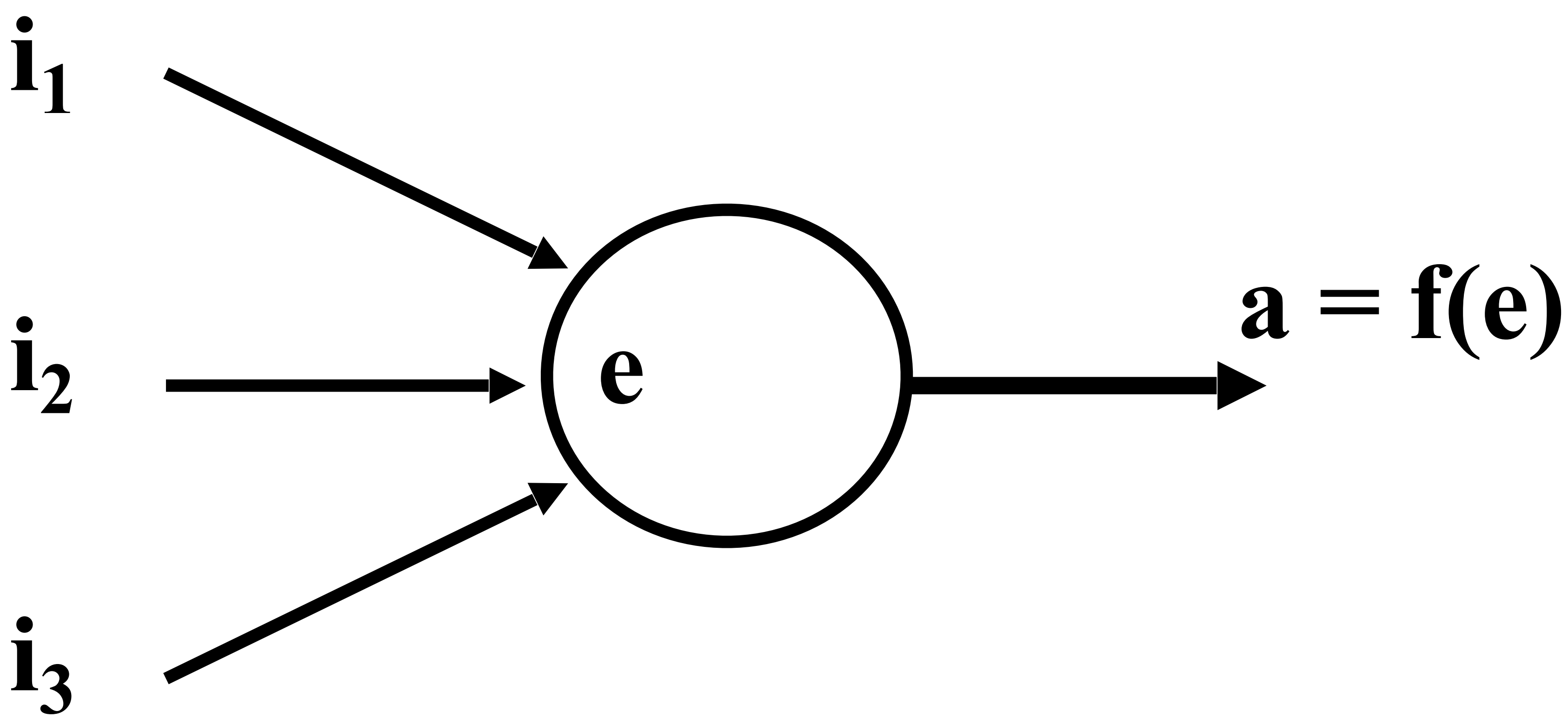


Neural activity



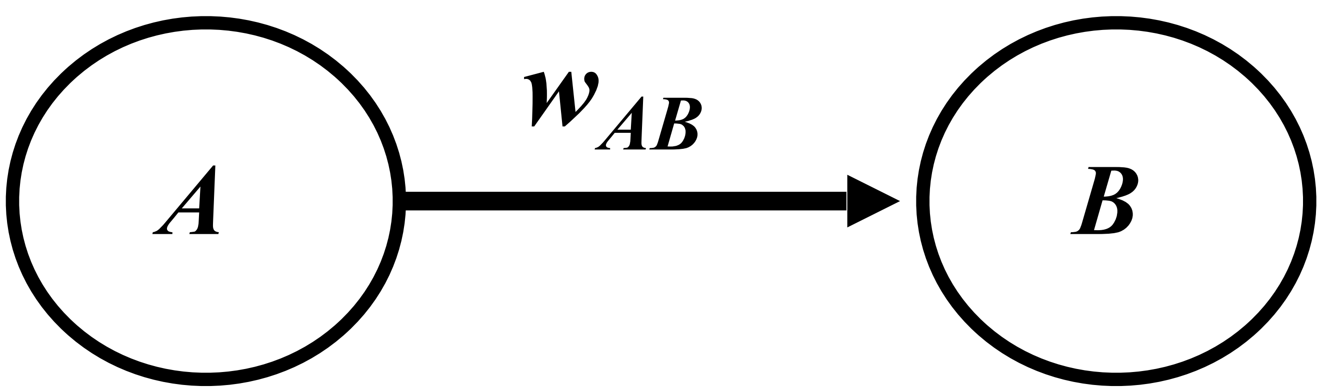
Artificial Neurons

- input (vectors)
- summation (excitation)
- output (activation)

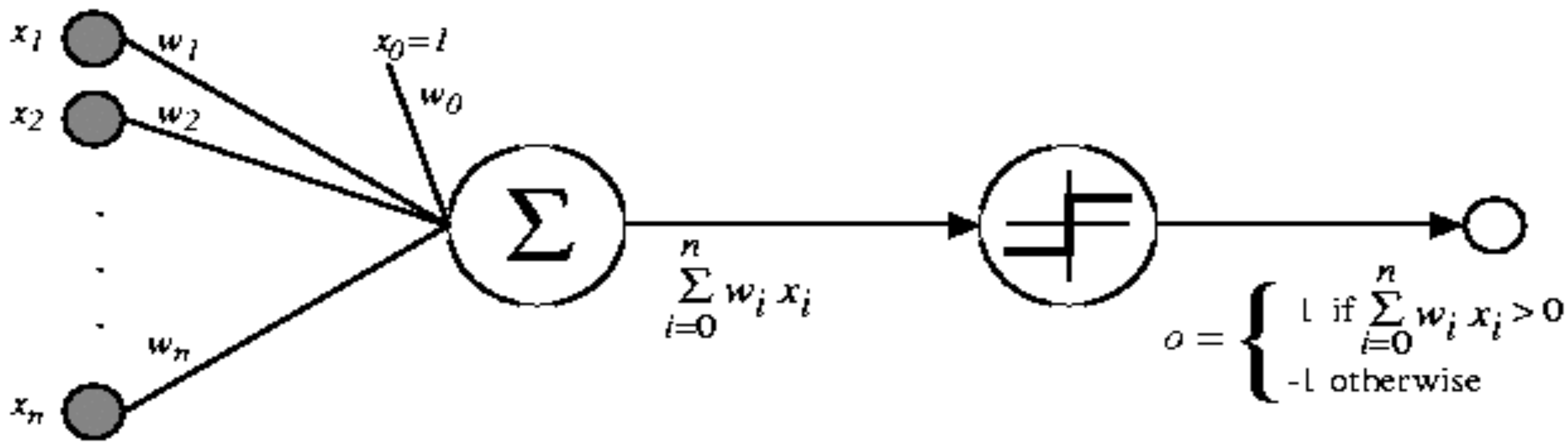


Artificial Connections (Synapses)

- w_{AB}
  - The weight of the connection from neuron *A* to neuron *B*



The Perceptron



$$o(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\ -1 & \text{otherwise.} \end{cases}$$

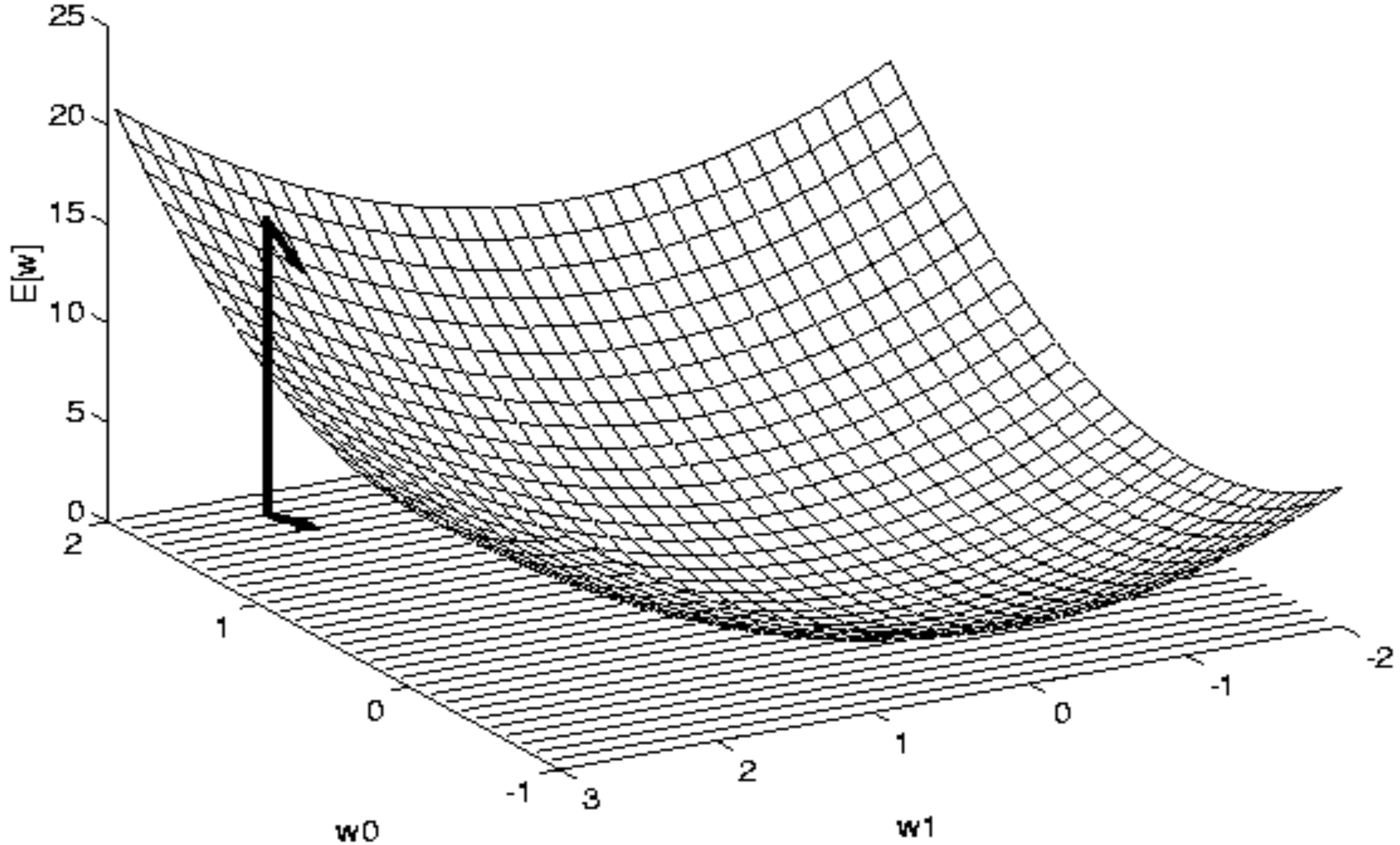
Learning in the Perceptron

- Delta learning rule
  - the difference between the desired output ***t*** and the actual output ***o*,** given input ***x***

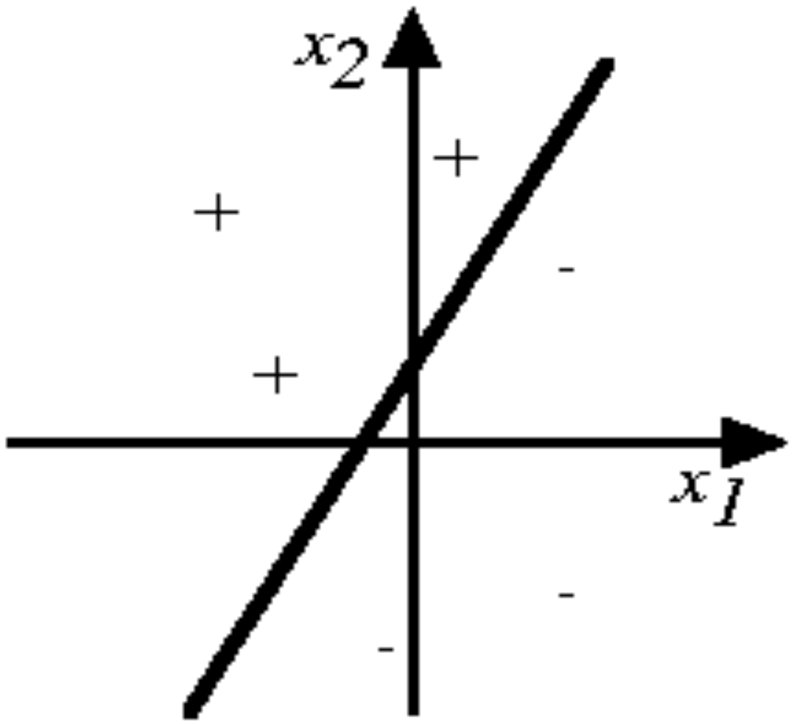
weights are adjusted to minimise the difference

- Global error E
  - is a function of the differences between the desired (t) and actual (o) outputs

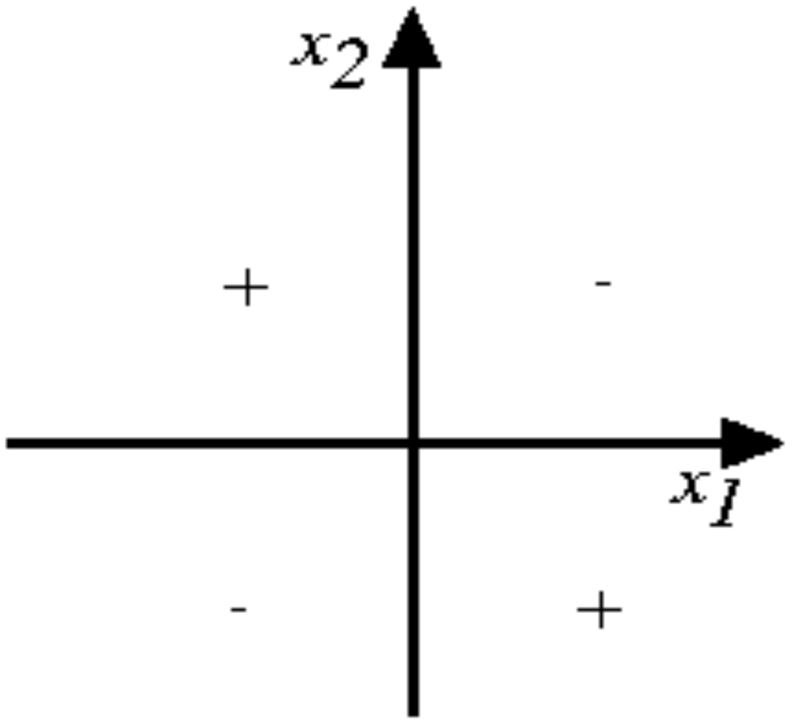
Gradient Descent



Can only learn linearly separable classification tasks



(a)



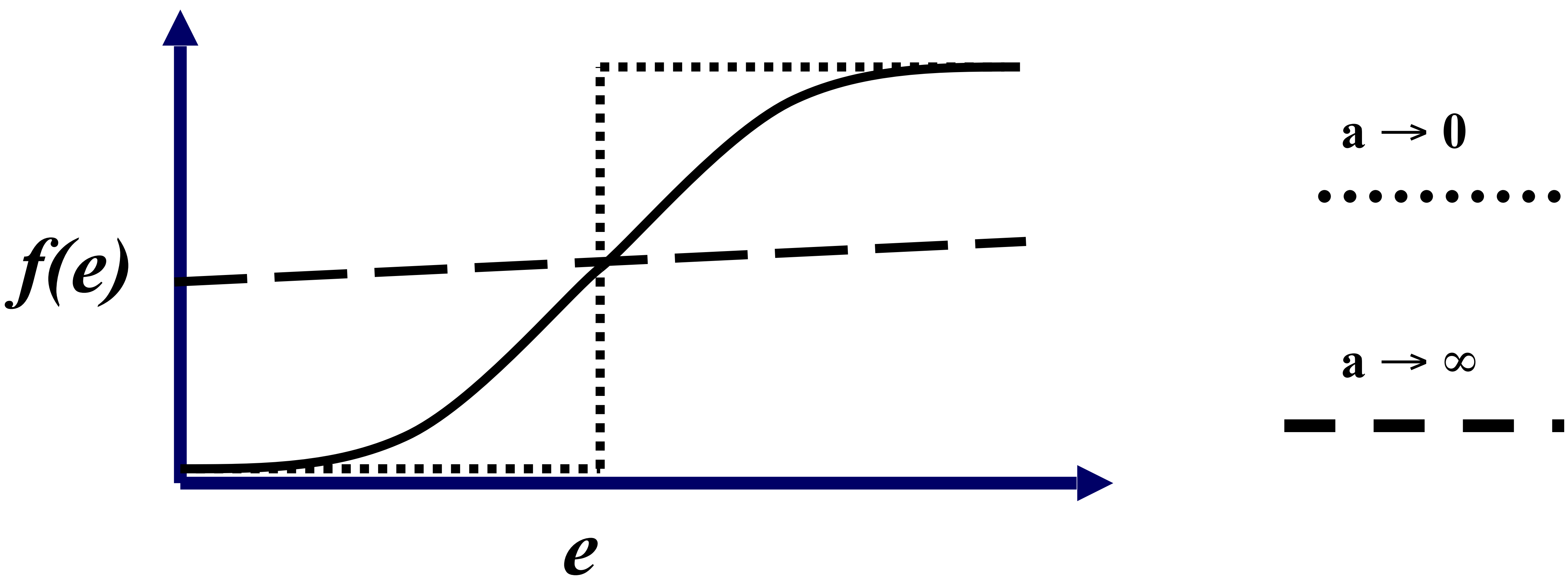
(b)

The history of the Perceptron

- Rosenblatt (1959)
- Minsky & Papert (1961)
- Rumelhart, McClelland & Hinton (1986)
- LeCun (1989), Hinton, LeCun (2016…)

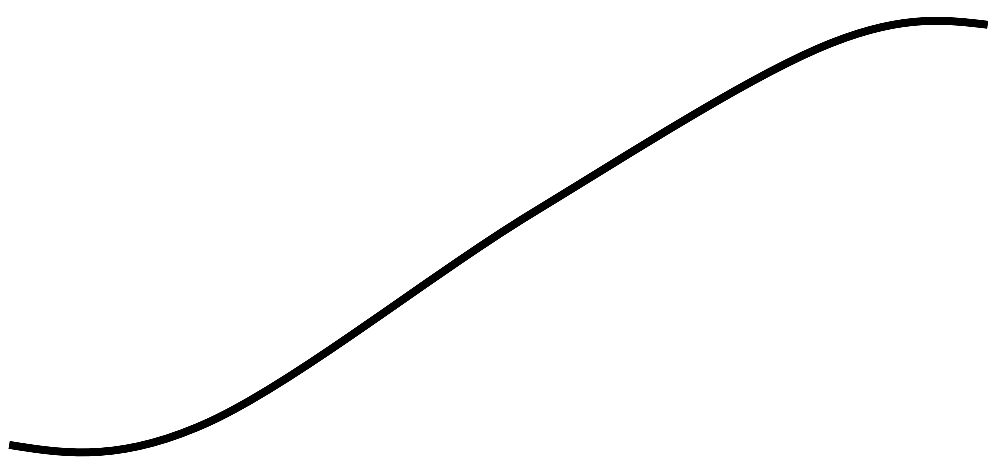
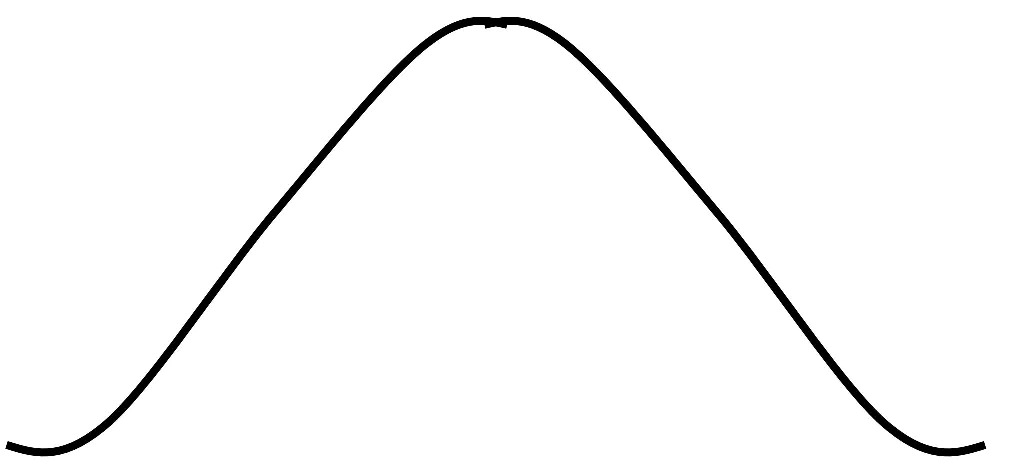
Sigmoid transfer (input-output) function

- nonlinear function: $f(x) = \dfrac{1}{1 + e^{-x/a}}$

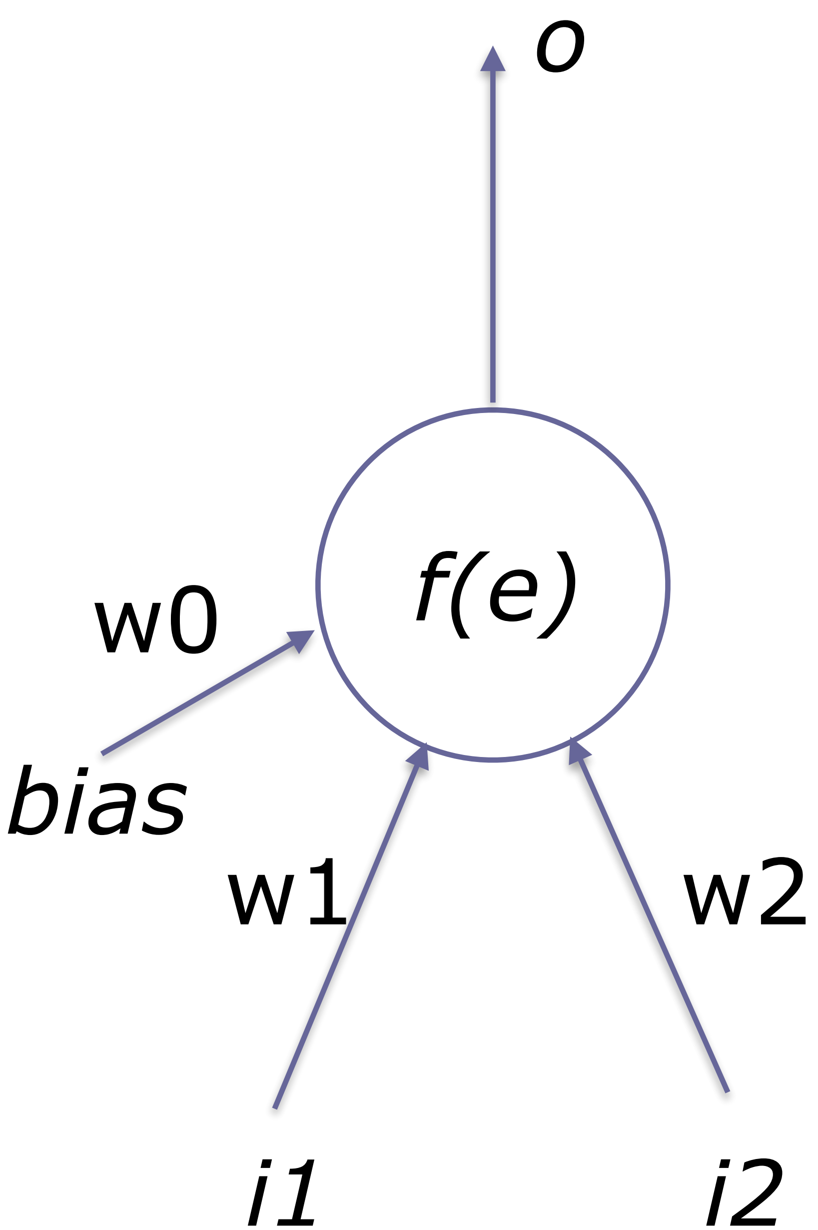


sigmoid transfer function

- May also be the **tanh** function (<-1,+1> instead of <0,1>)
- Derivative $f'(x) = f(x)\,[1 - f(x)]$

Perceptron



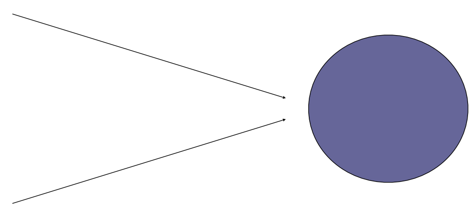
$$o = f\left(\sum_{i=0}^{I} w_i\, i_i\right) = f(e)$$

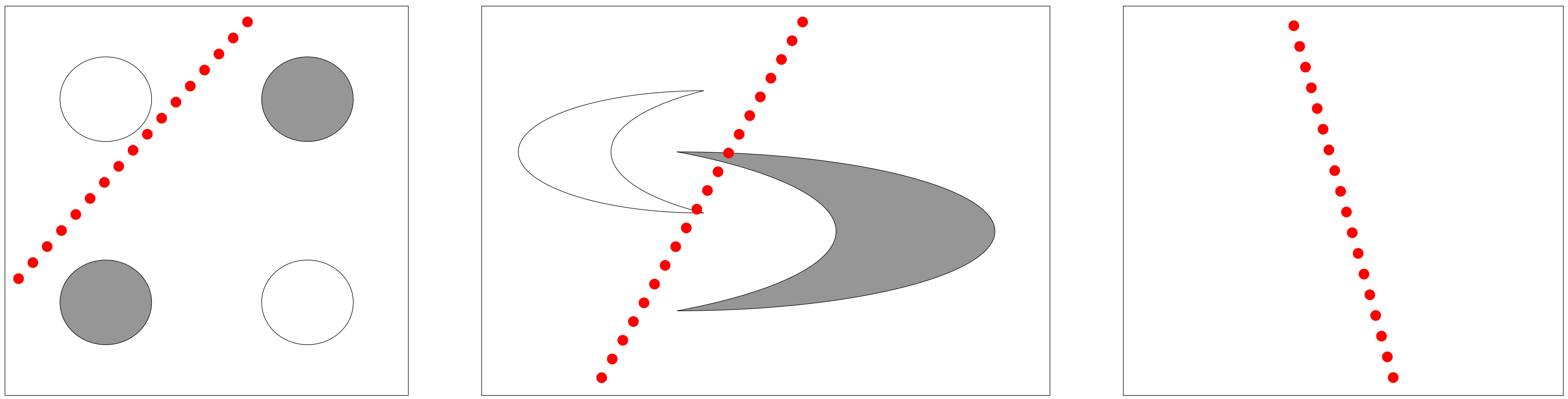
Derivation of the delta learning rule (for a sigmoid transfer function)

$$E = \sum_{p=1}^{P} \frac{1}{2}\left(t_p - o_p\right)^2$$

$$\Delta w_k = -\eta \frac{\partial E}{\partial w_k} = \eta \sum_{p=1}^{P} \left(t_p - o_p\right) f'\left(e_p\right) i_k = \eta \sum_{p=1}^{P} \sigma\, i_k$$

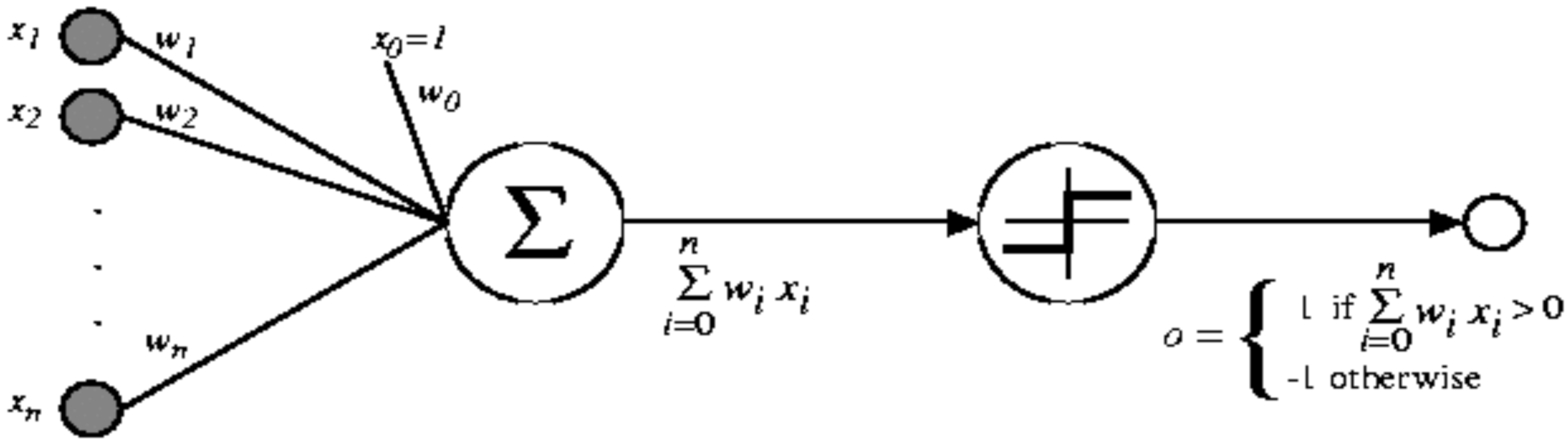
Decision boundaries of Perceptrons





Straight lines (surfaces), linear separable

Can you derive the equation of the decision boundary for a perceptron with 2 inputs (+ 1 bias input)?

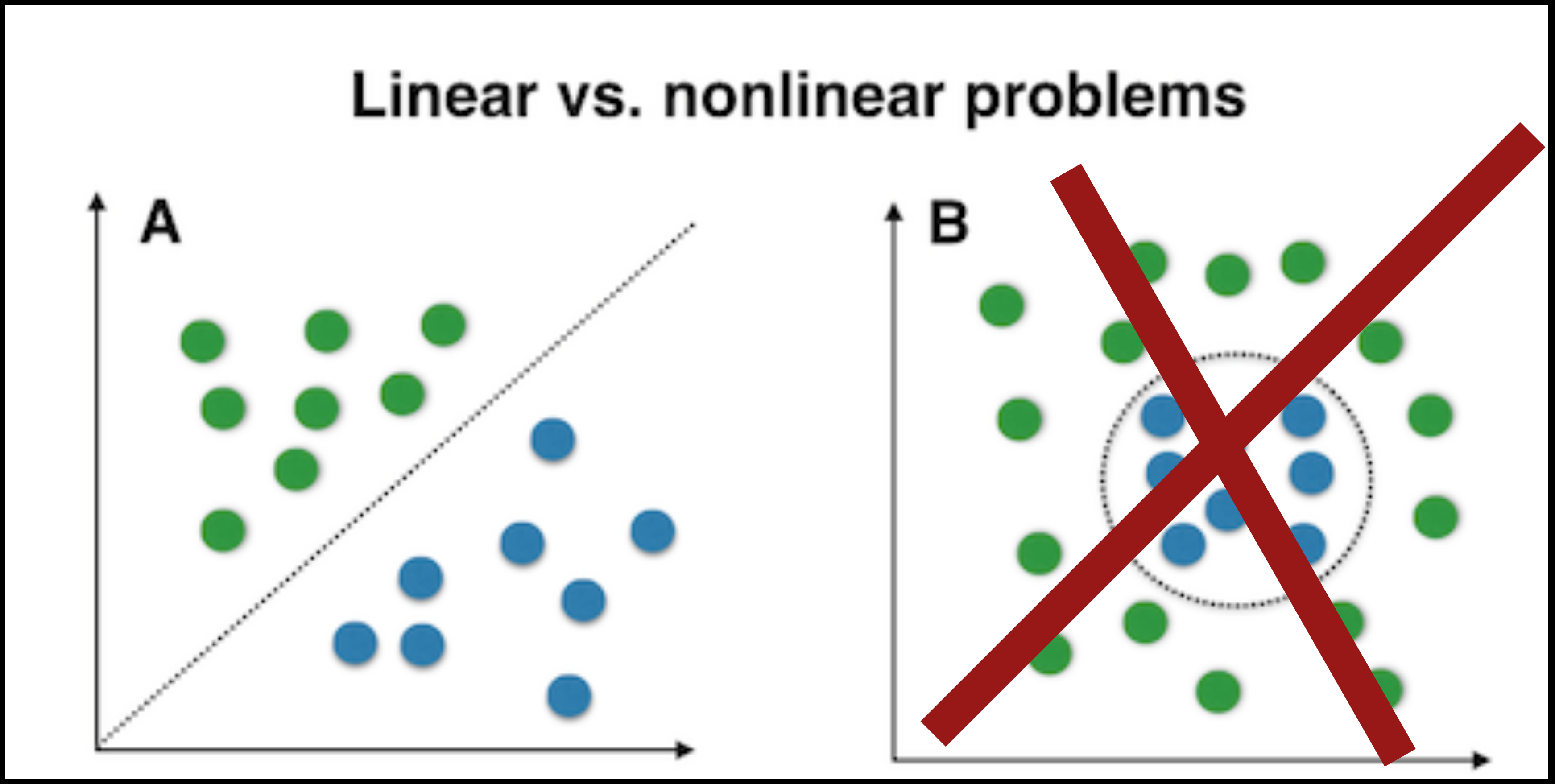


$$o(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\ -1 & \text{otherwise.} \end{cases}$$

Exercises

https://github.com/ericpostma/ep
KPMG1.pdf
KPMG2.pdf
KPMG3.pdf

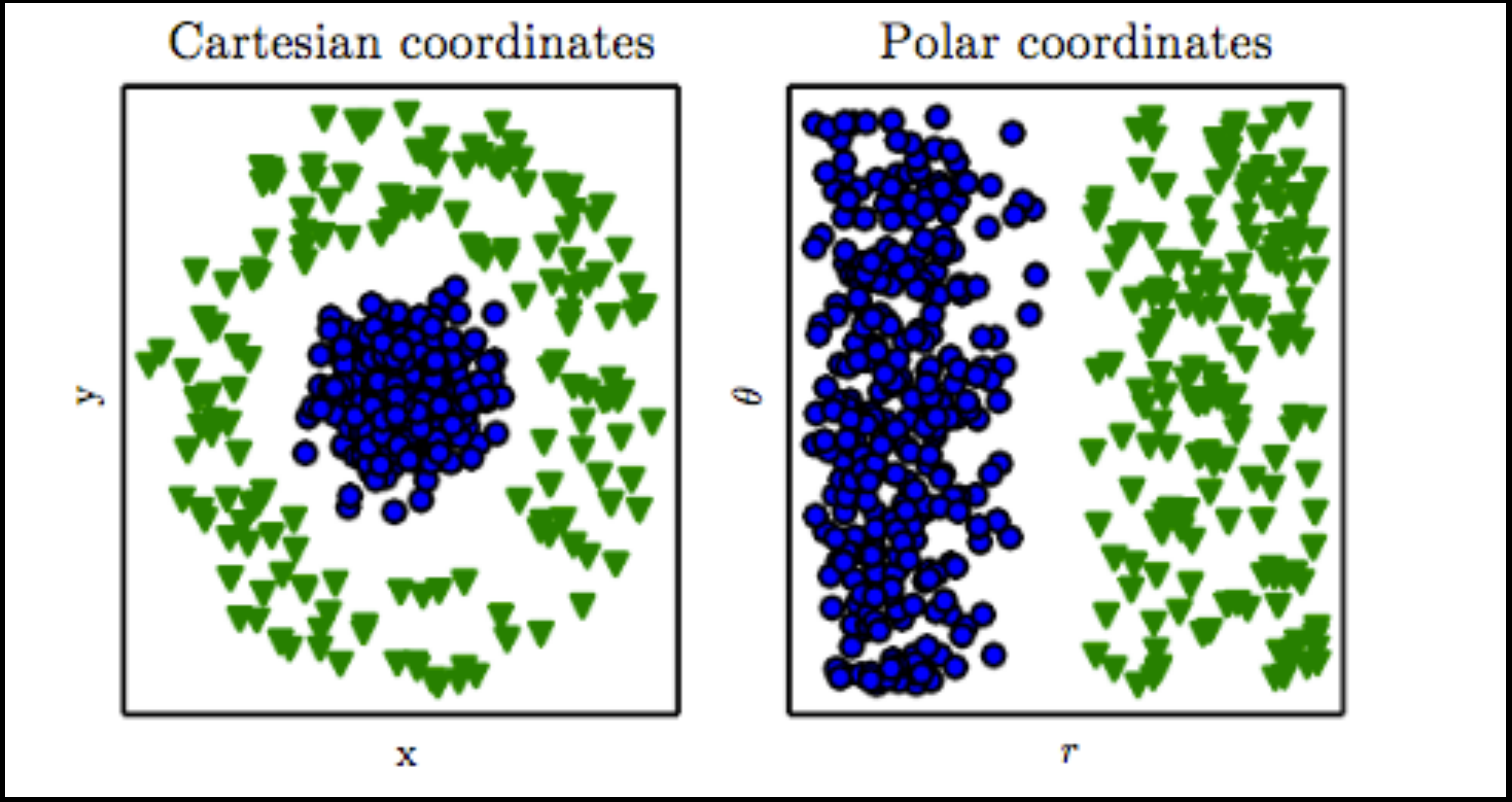
Linear separability



The importance of representation

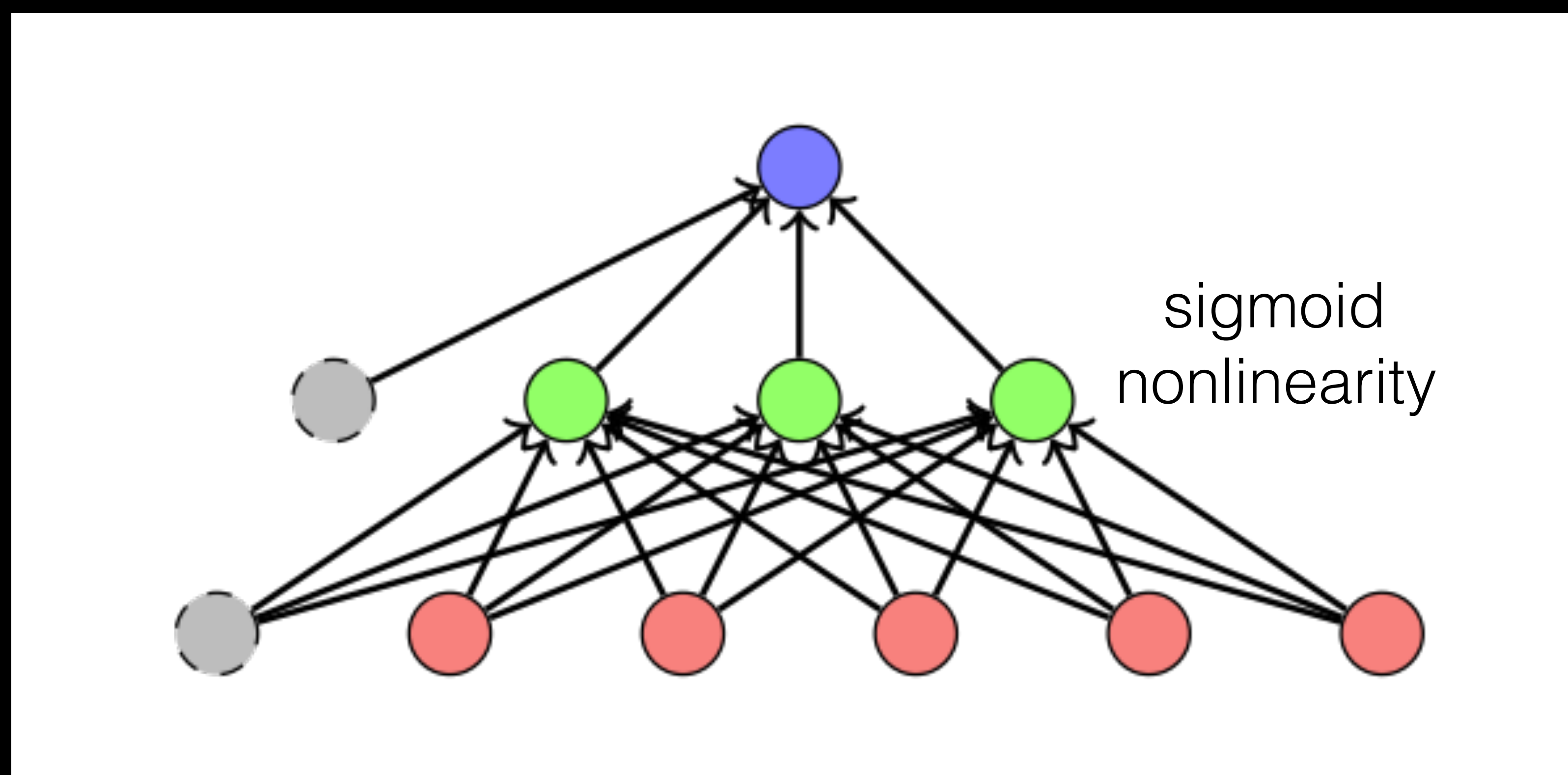
Perceptron can't classify this

Perceptron can classify this



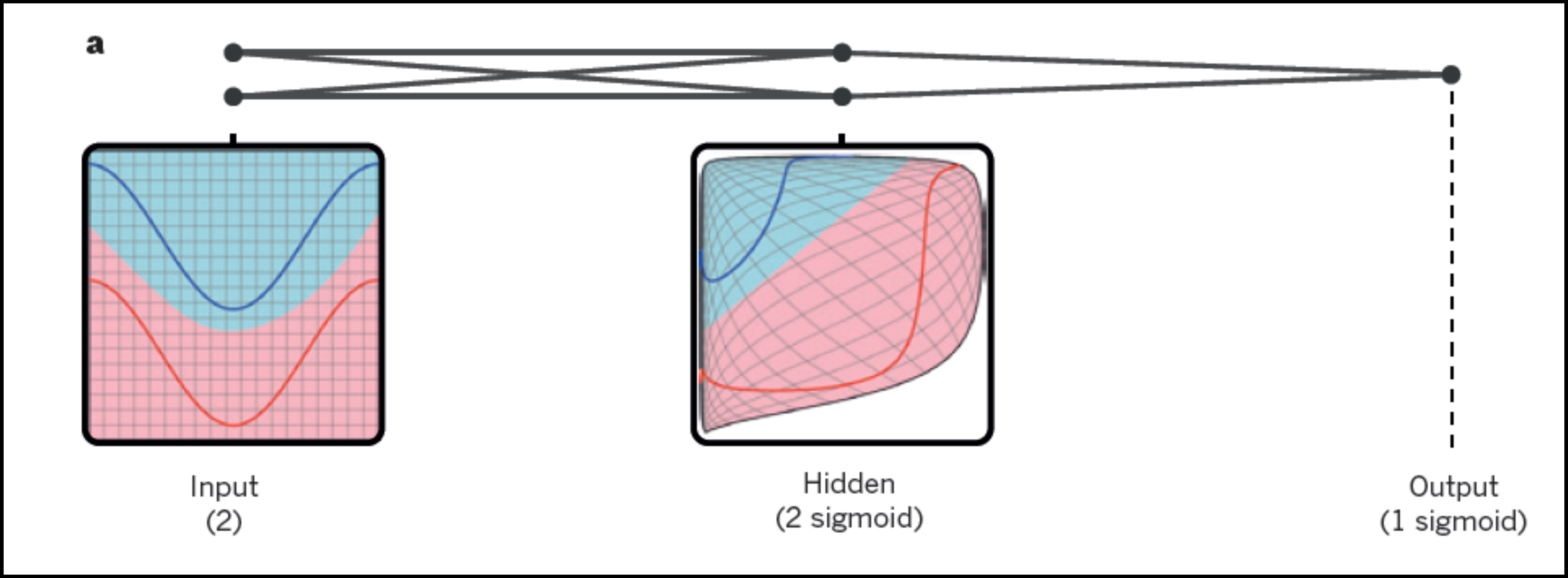
same task, different representation

Stacking Perceptrons



Main challenge (in the period 1960-1988):
How to automatically adapt the weights.

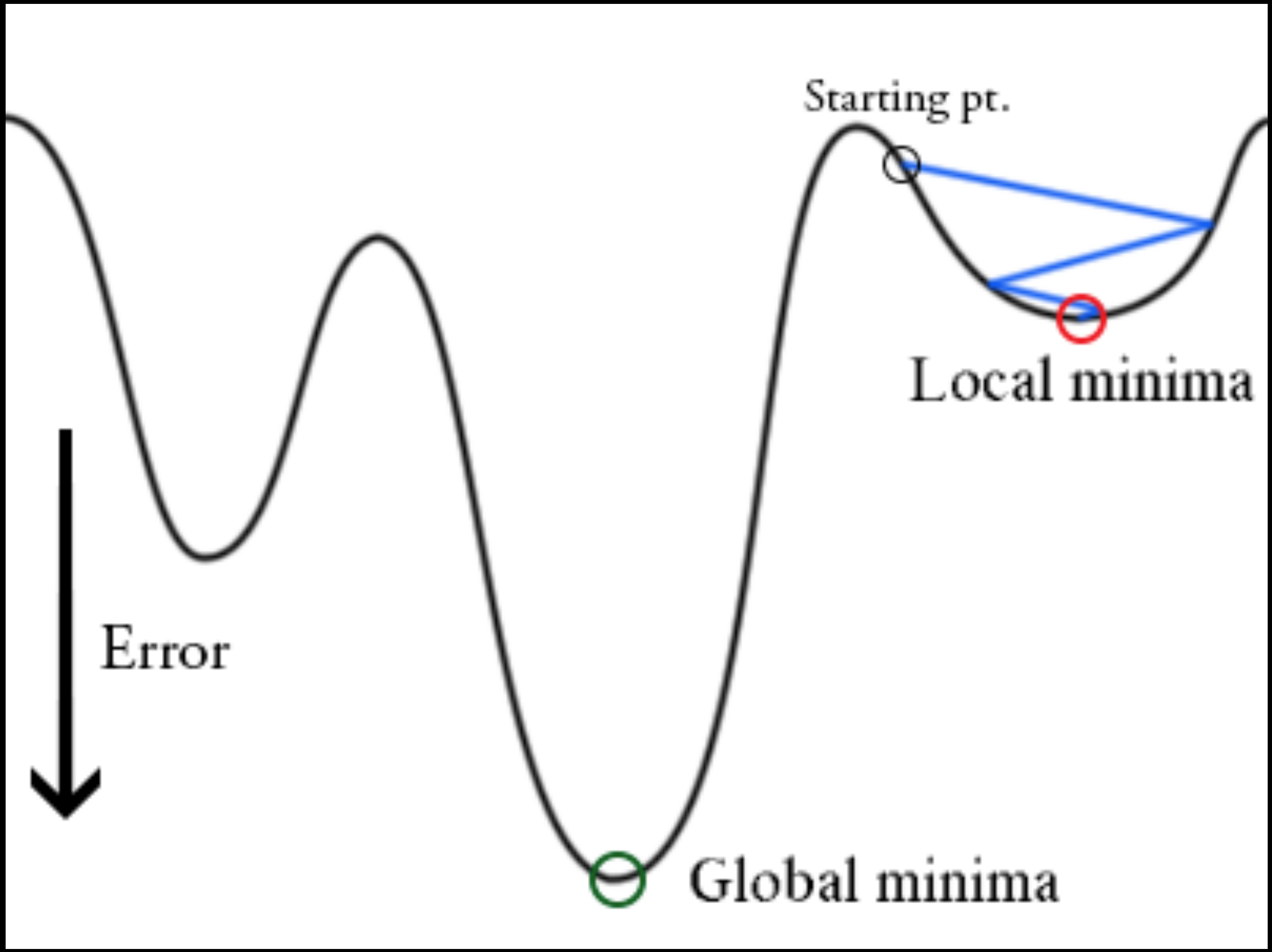
Multilayer Perceptron (1988)



Automatically Learns Non-Linear Mappings

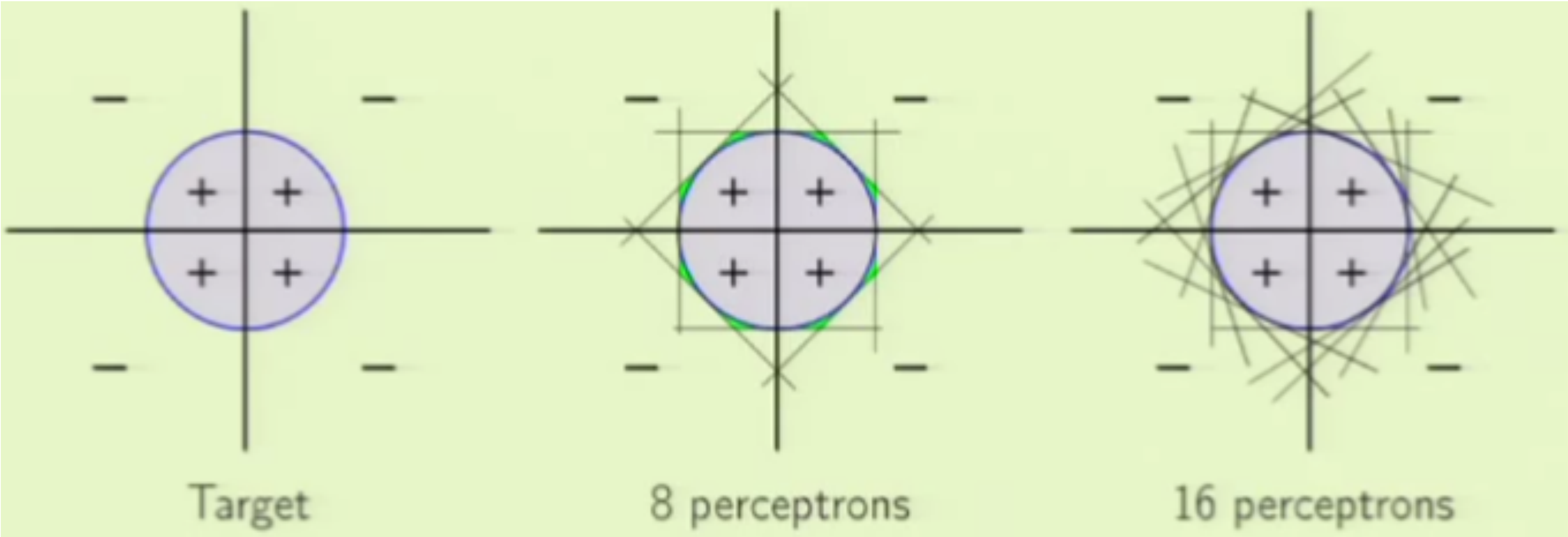
Gradient Descent

- Backpropagation (generalised delta rule)



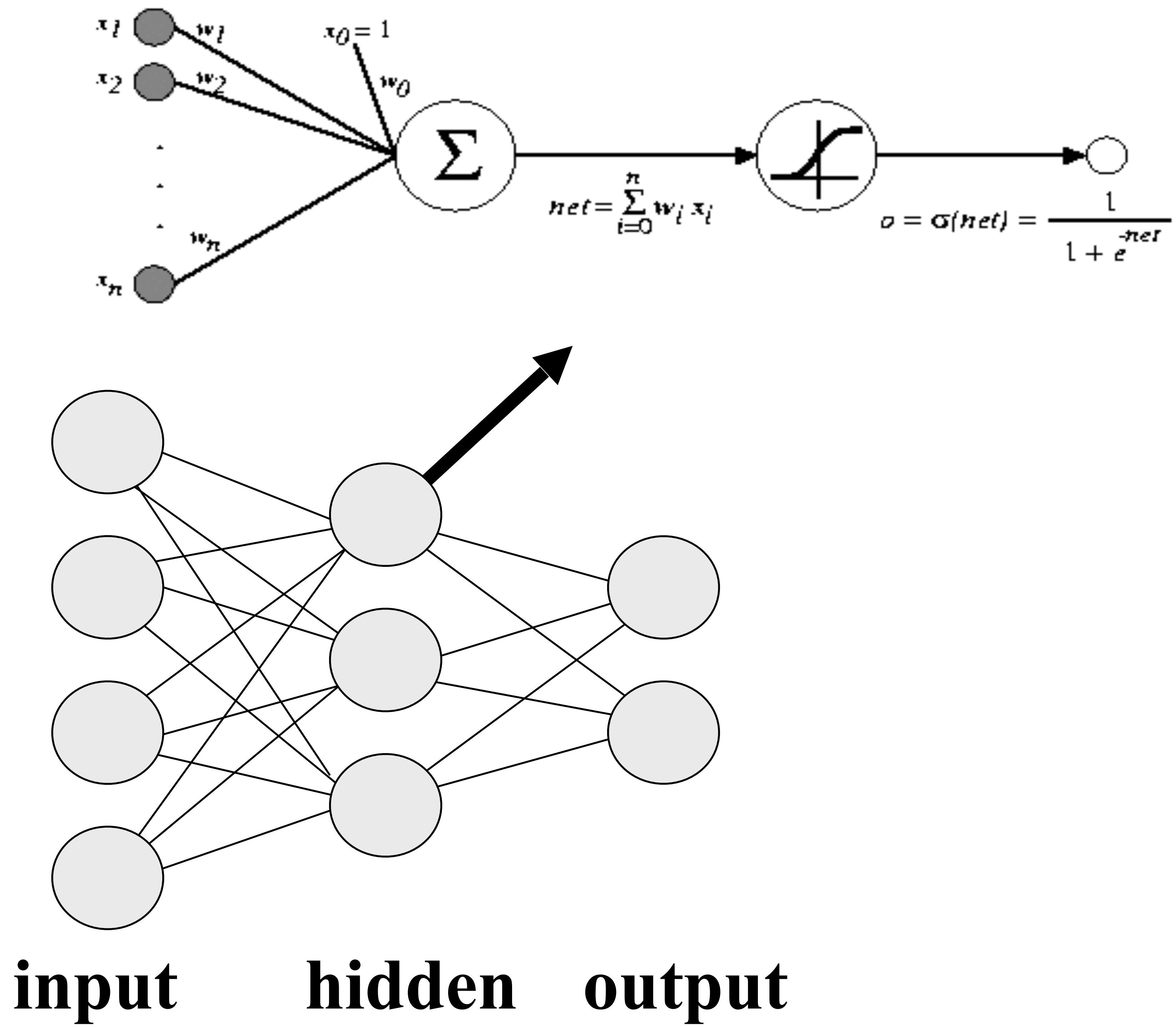
Coping with the limits of perceptrons

- Stacking perceptrons



what do these stacked
perceptrons look like?

The Multilayer Perceptron

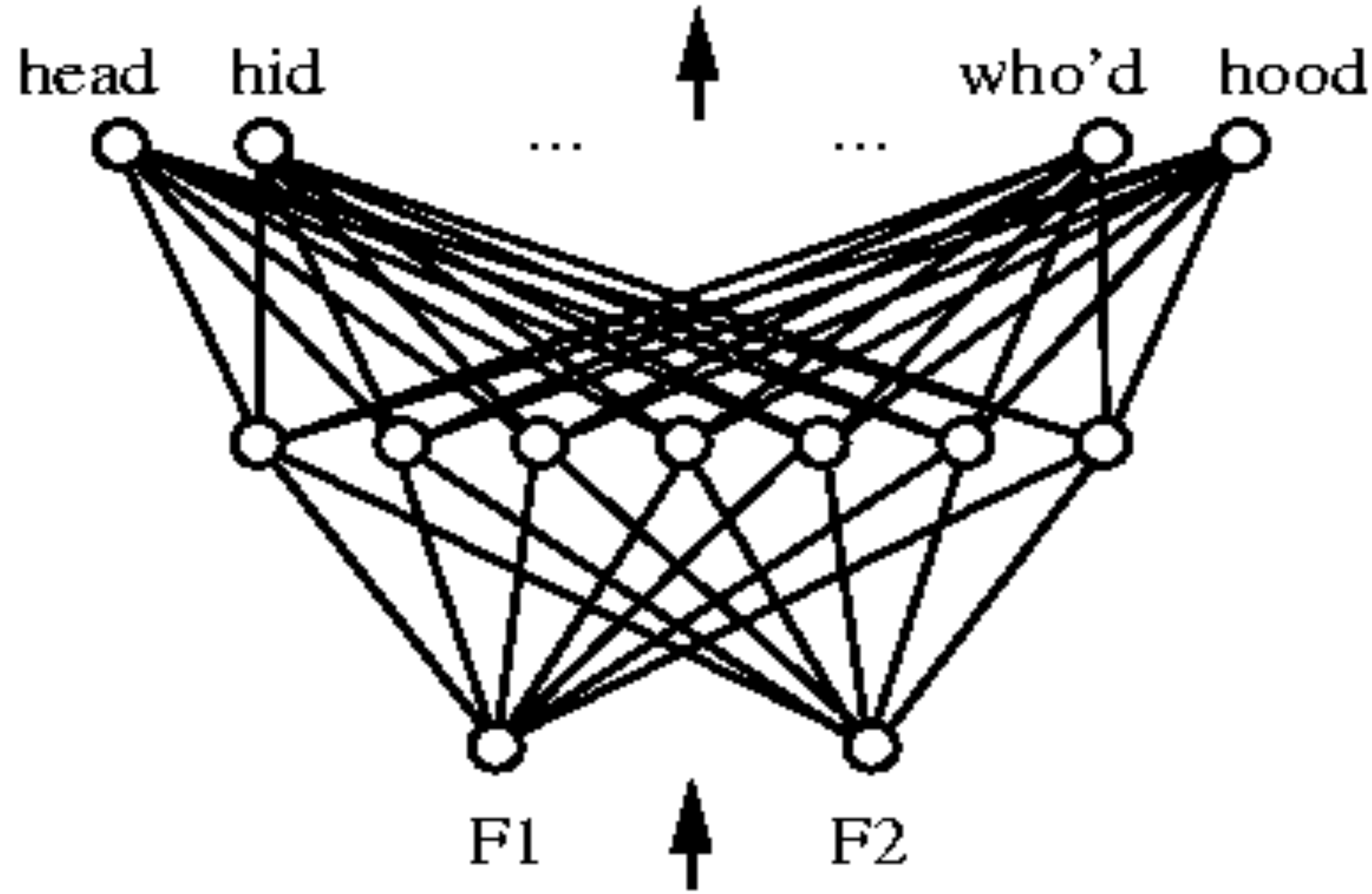


Training the MLP

- supervised learning
  - each training pattern: input I + desired output T
  - in each *epoch*: present all patterns
  - at each presentation: adapt weights
  - after many epochs convergence to a local minimum

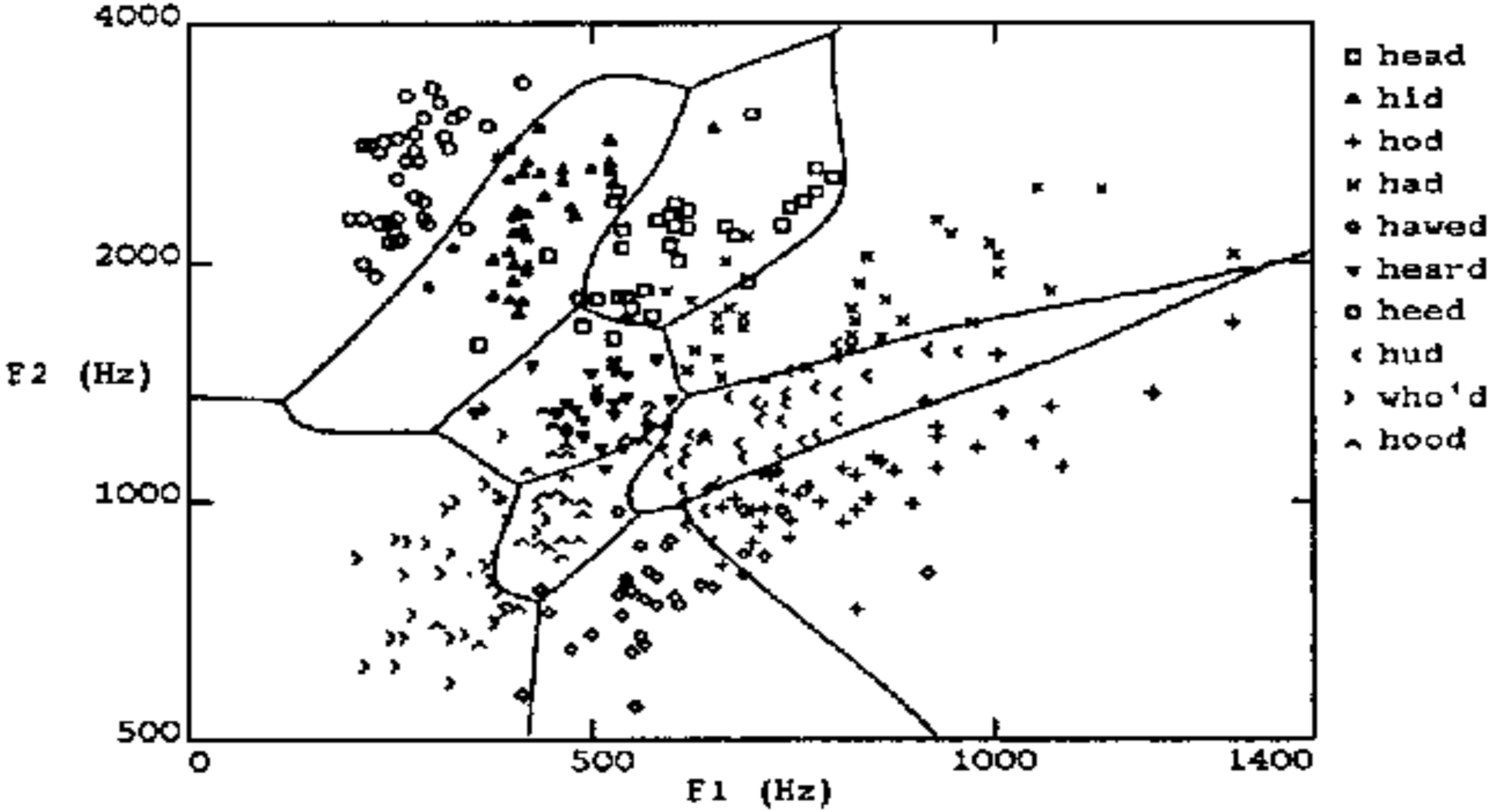
phoneme recognition with a MLP

Output:
pronunciation

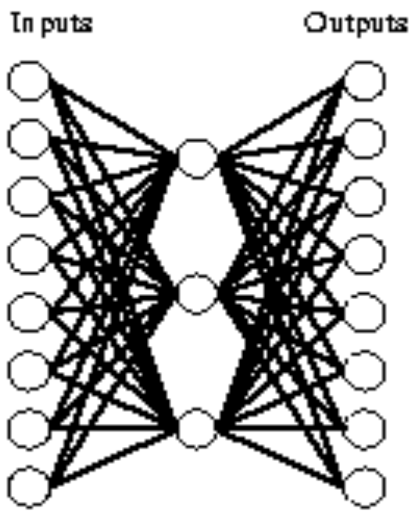


input:
frequencies

Non-linear decision boundaries



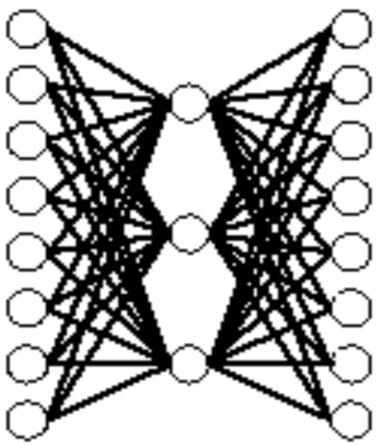
The *Autoencoder* *(Compression with an MLP)*



A target function:

Input		Output
10000000	$\rightarrow$	10000000
01000000	$\rightarrow$	01000000
00100000	$\rightarrow$	00100000
00010000	$\rightarrow$	00010000
00001000	$\rightarrow$	00001000
00000100	$\rightarrow$	00000100
00000010	$\rightarrow$	00000010
00000001	$\rightarrow$	00000001

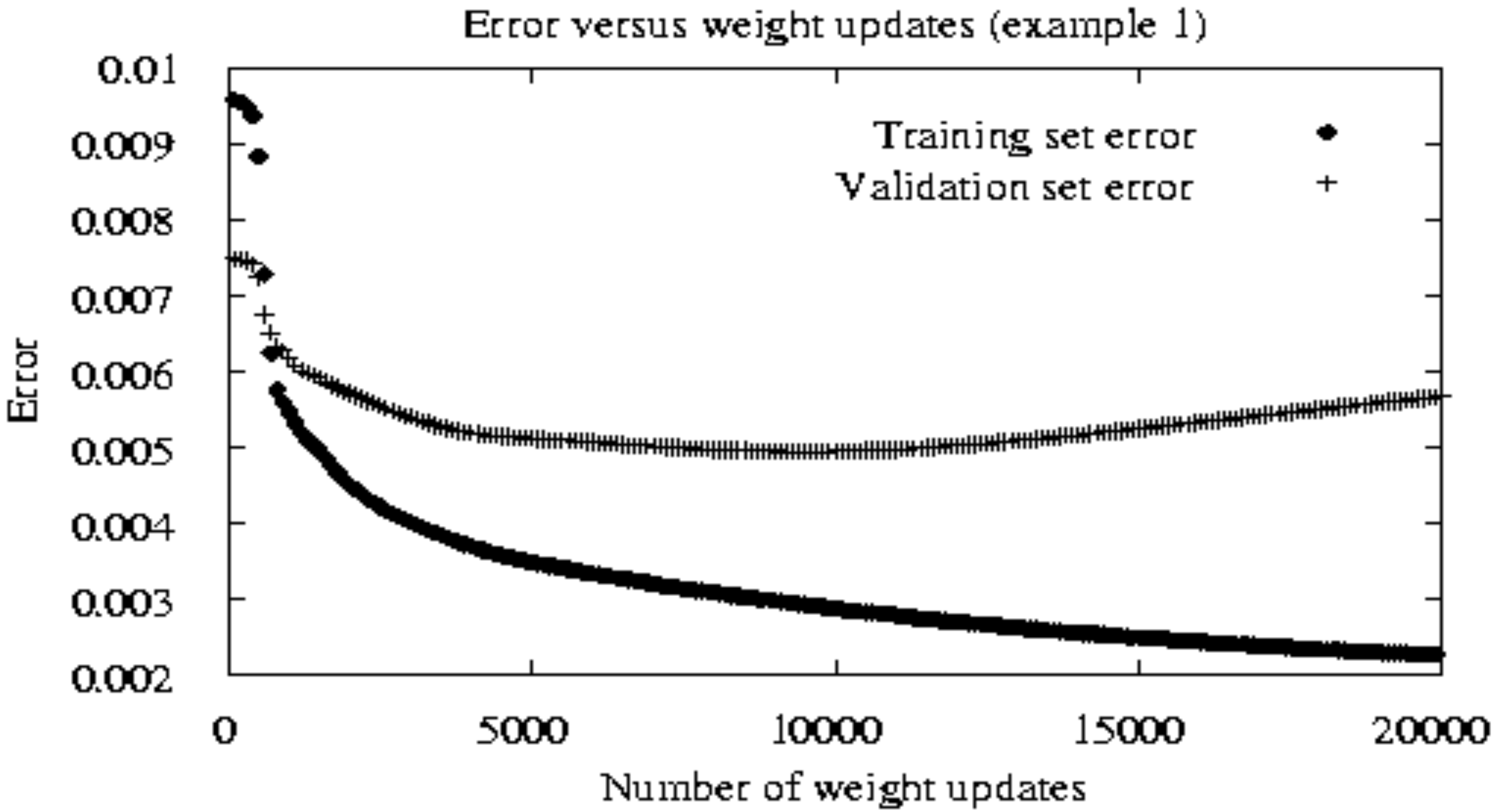
hidden representation



Learned hidden layer representation:

Input		Hidden Values				Output
10000000	→	.89	.04	.08	→	10000000
01000000	→	.01	.11	.88	→	01000000
00100000	→	.01	.97	.27	→	00100000
00010000	→	.99	.97	.71	→	00010000
00001000	→	.03	.05	.02	→	00001000
00000100	→	.22	.99	.99	→	00000100
00000010	→	.80	.01	.98	→	00000010
00000001	→	.60	.94	.01	→	00000001

Learning in the MLP



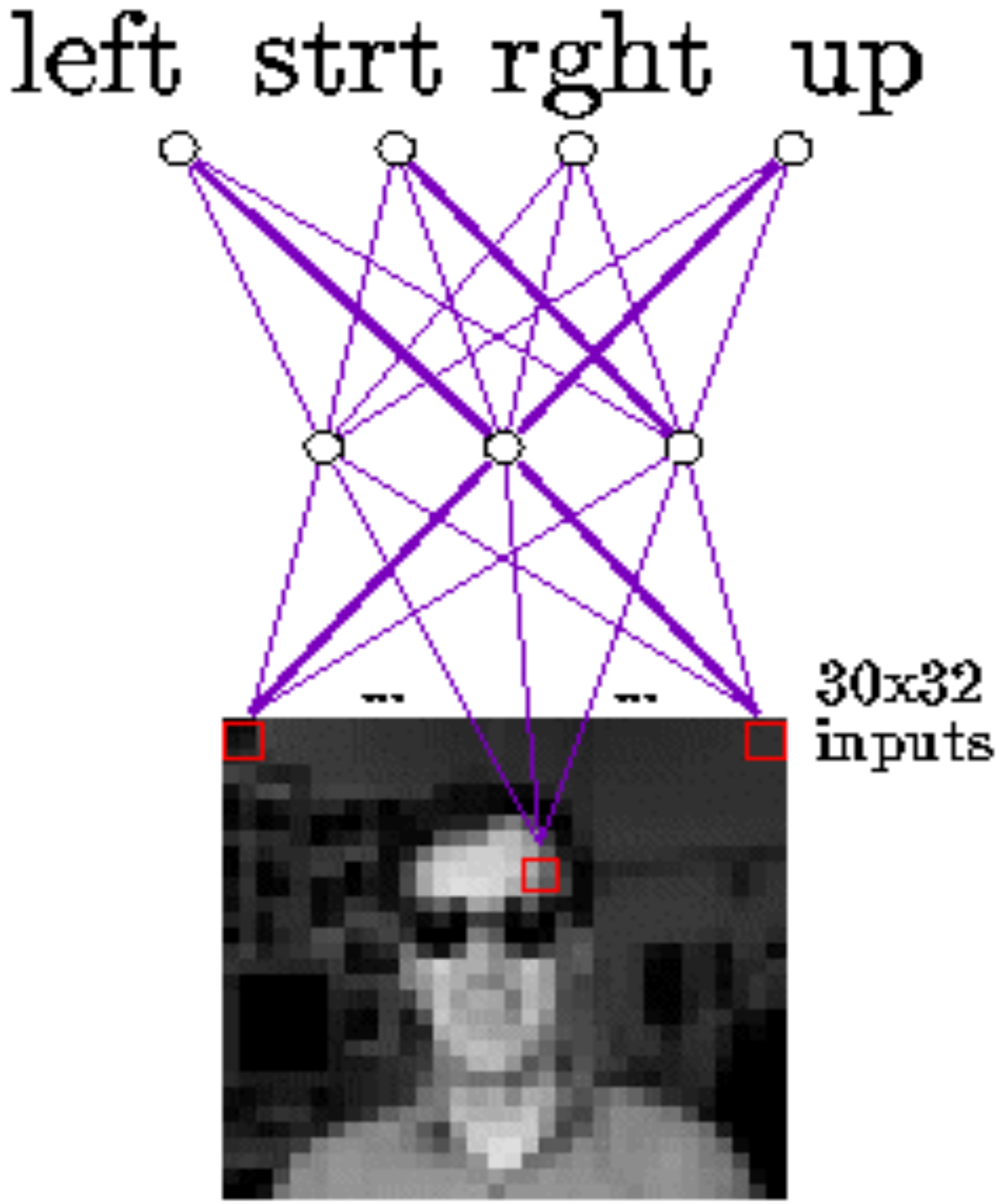
Preventing Overfitting

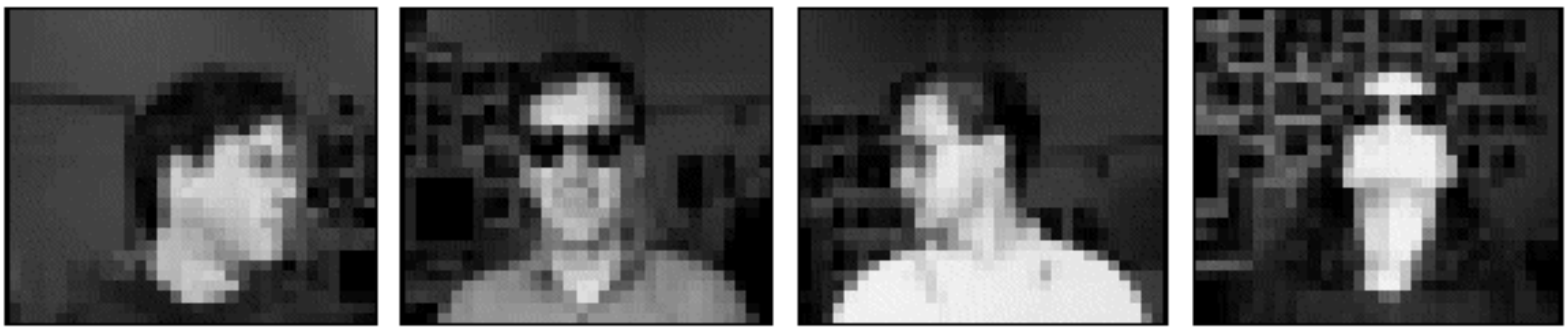
GENERALISATION

= performance on test set (unseen data)

- Early stopping
- Training, Validation and Test set
- *k*-fold cross validation
- Regularisation

Image Recognition with the MLP

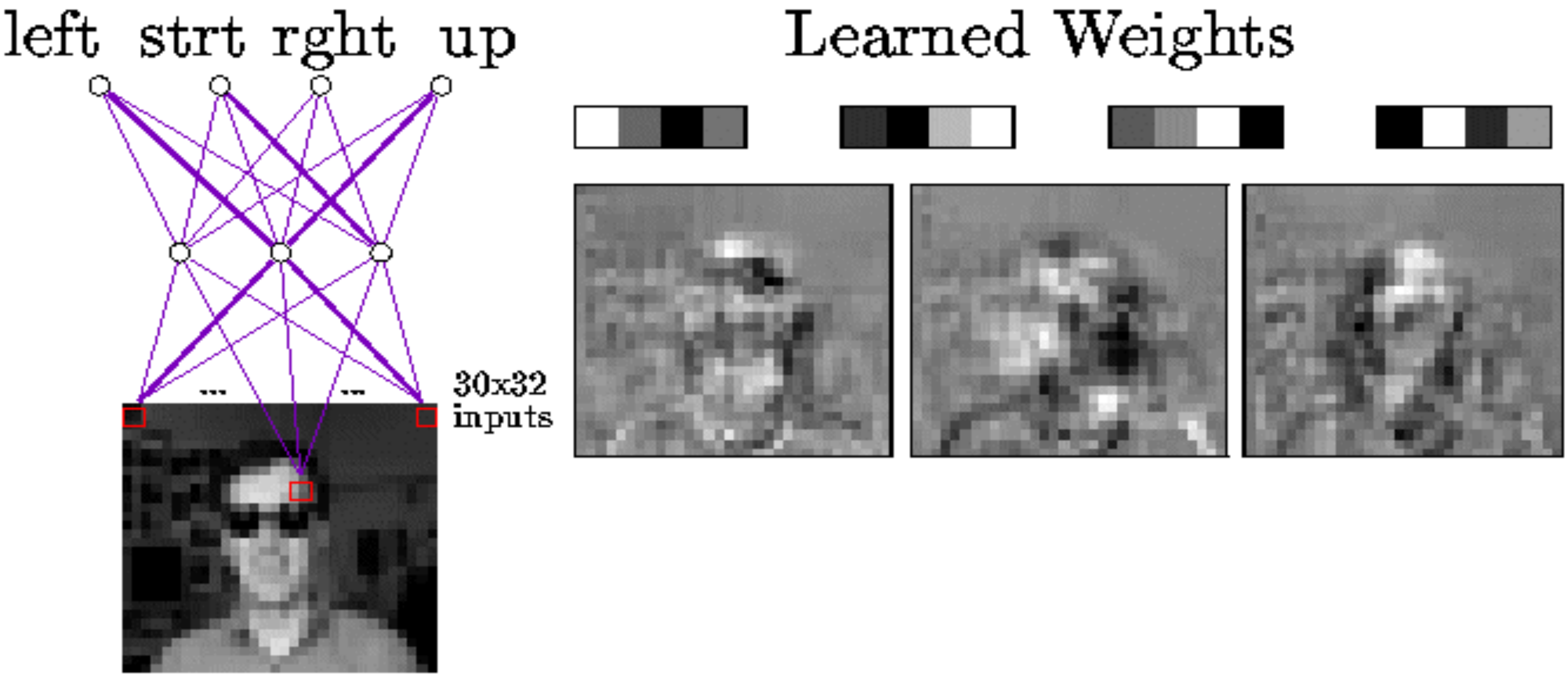




Typical input images

After training…

Hidden Representations



Questions

- The MLP consists of one (or more) hidden layers, i.e., hidden neurons with sigmoid transfer functions

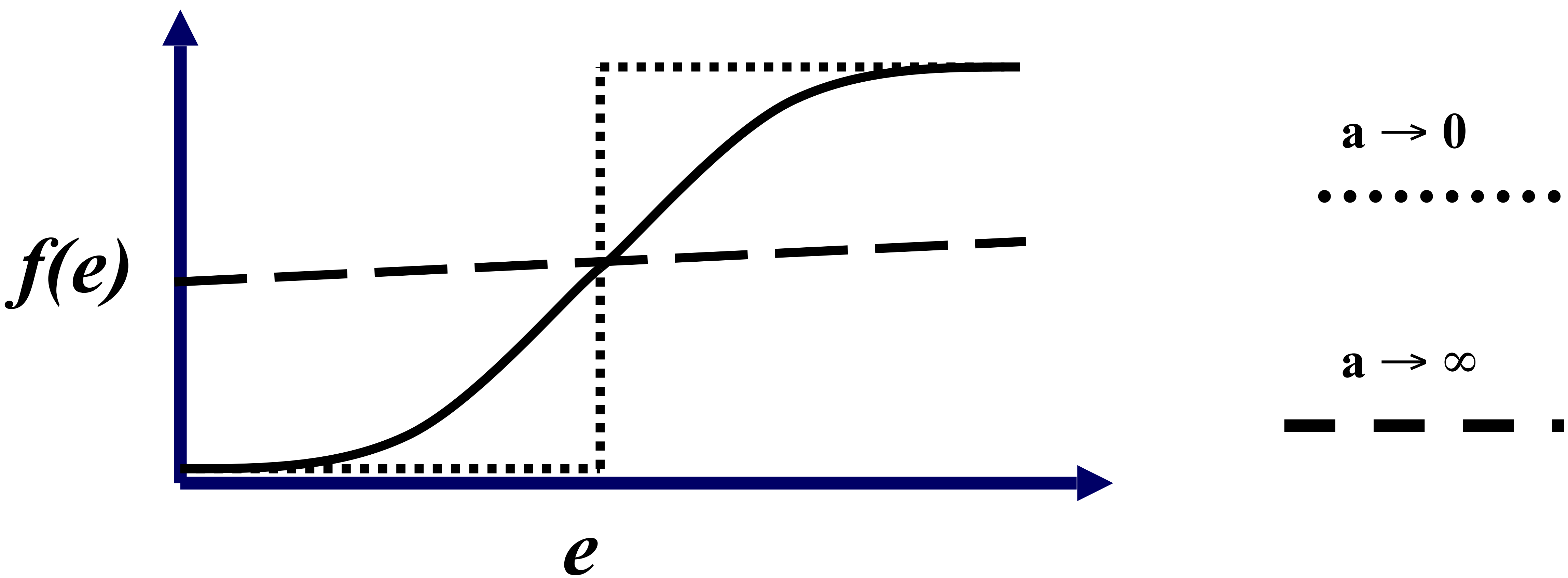
1. Can linear transfer functions be used? What would be the pro’s and con’s?

2. Can step transfer functions be used? What would be the pro’s and con’s?

RECAP PERCEPTRON…

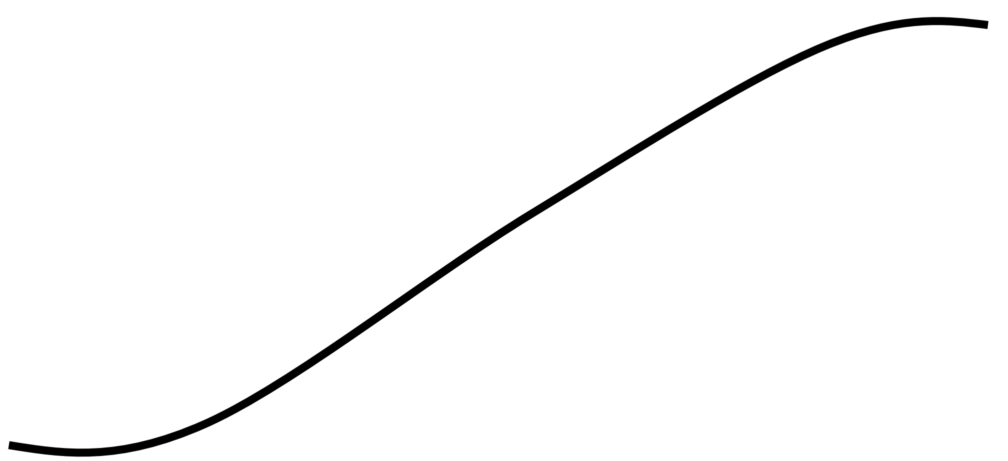
Sigmoid transfer (input-output) function

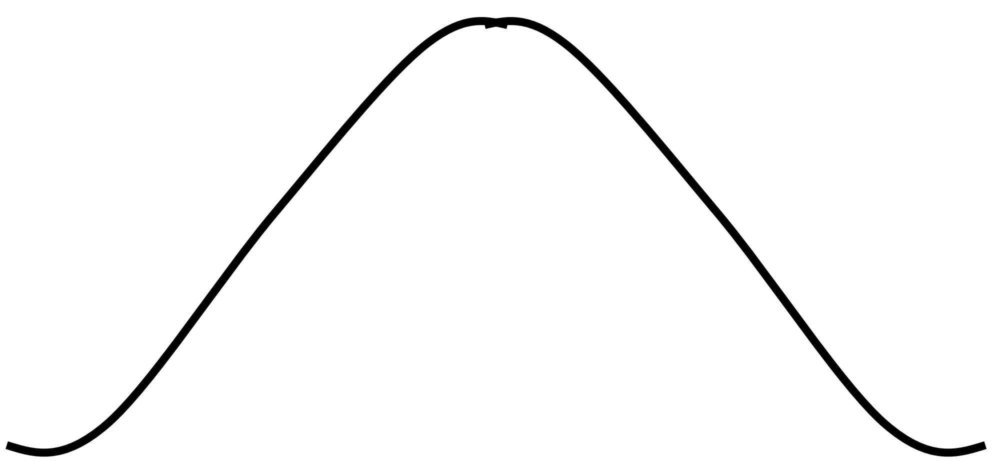
- nonlinear function: $$f(x) = \frac{1}{1 + e^{-x/a}}$$



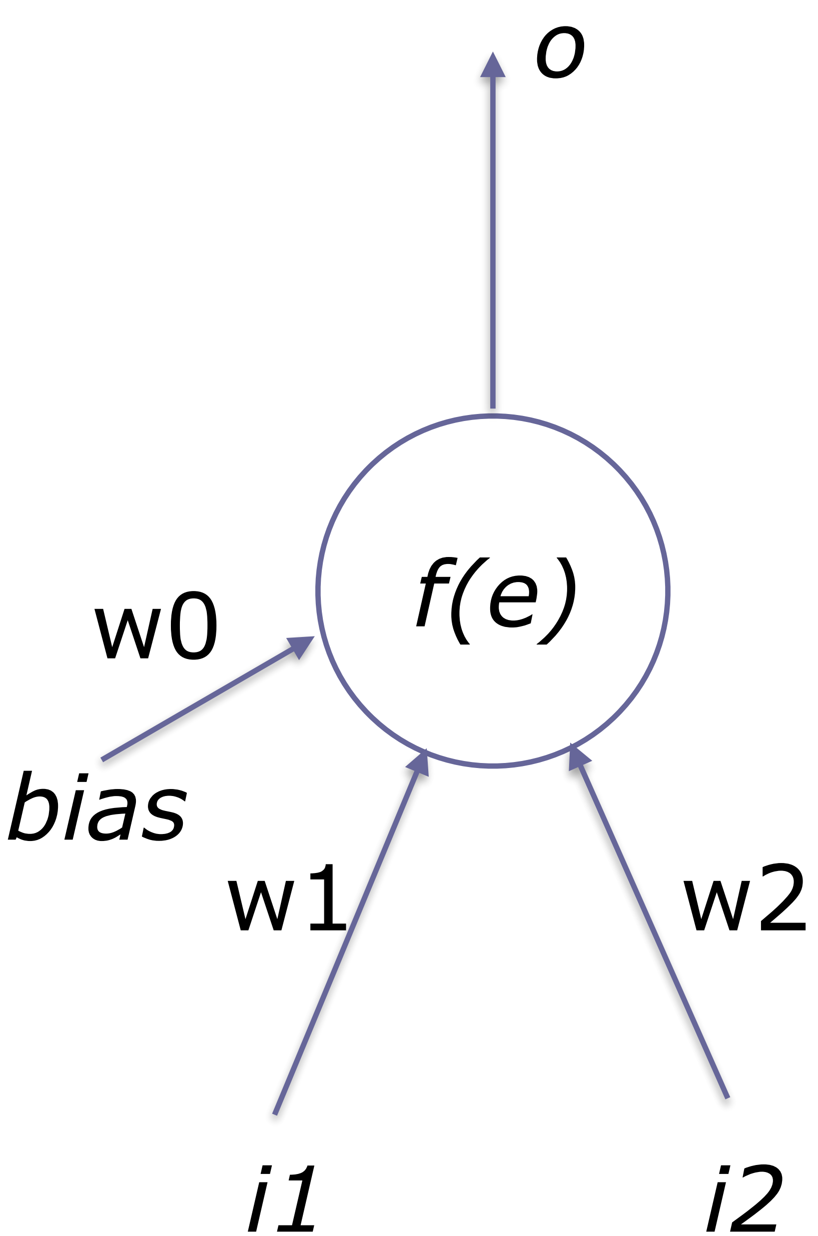
sigmoid transfer function

- May also be the **tanh** function (<-1,+1> instead of <0,1>)
- Derivative $f'(x) = f(x)\,[1 - f(x)]$





Perceptron



$$o = f\left(\sum_{i=0}^{I} w_i\, i_i\right) = f(e)$$

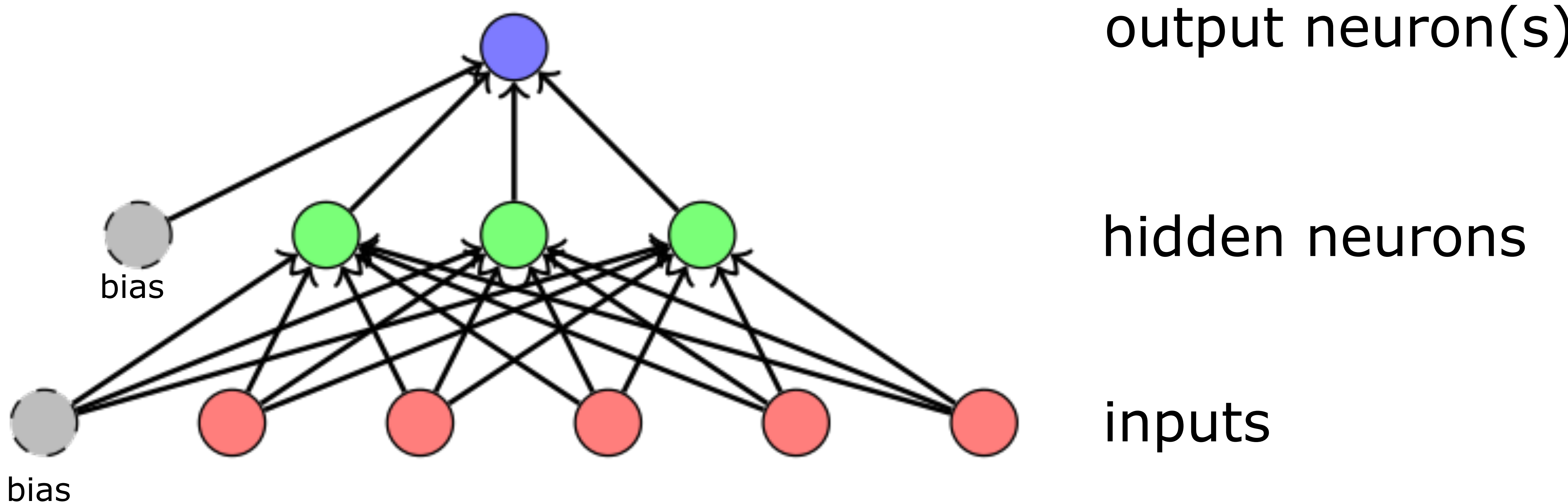
Derivation of the delta learning rule (for a sigmoid transfer function)

$$E = \sum_{p=1}^{P} \frac{1}{2}\left(t_p - o_p\right)^2$$

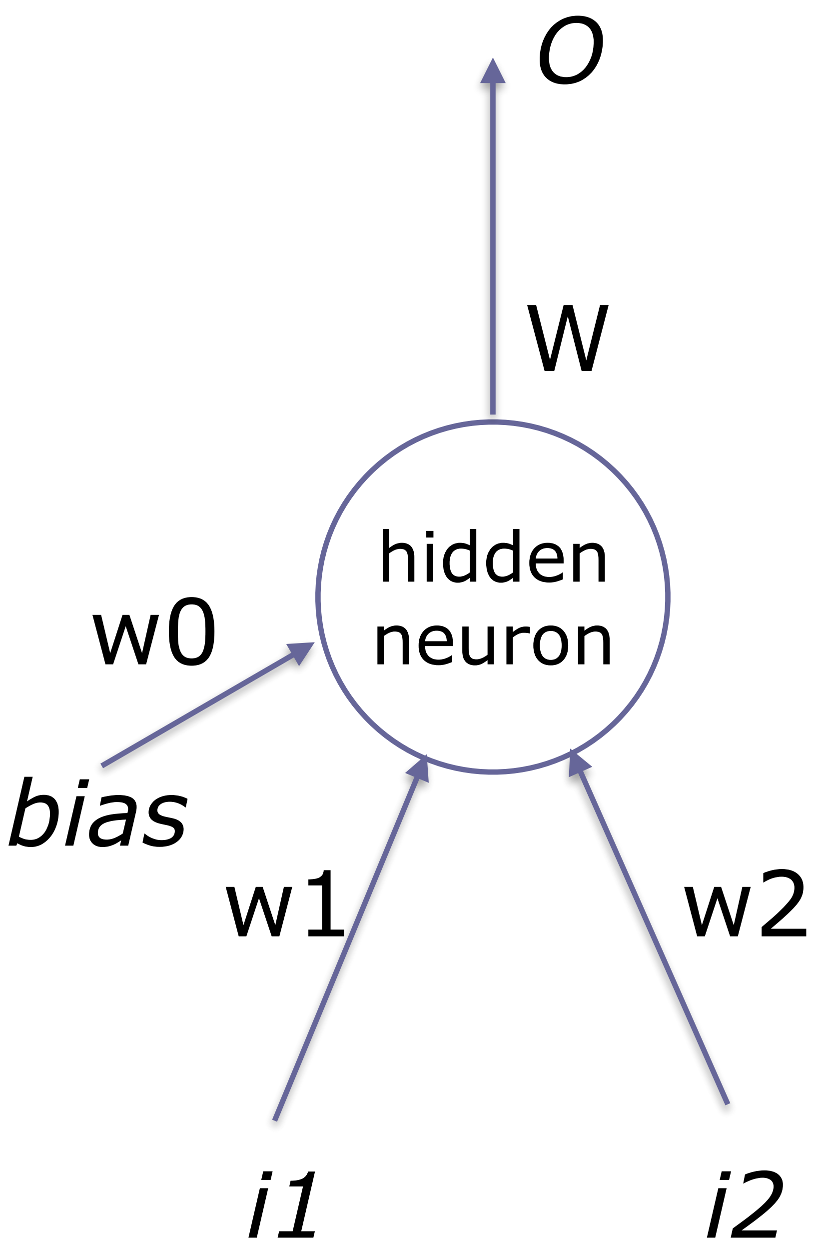
$$\Delta w_k = -\eta \frac{\partial E}{\partial w_k} = \eta \sum_{p=1}^{P} \left(t_p - o_p\right) f'\left(e_p\right) i_k = \eta \sum_{p=1}^{P} \delta\, i_k$$

Question

- How does the rate of weight change depend on the activation value of the neuron?


output neuron(s)
hidden neurons
bias
inputs
bias

Multilayer Perceptron



$$o_k = f\left(\sum_{j=0}^{H} W_{kj}\, h_j\right) = f(e_k)$$

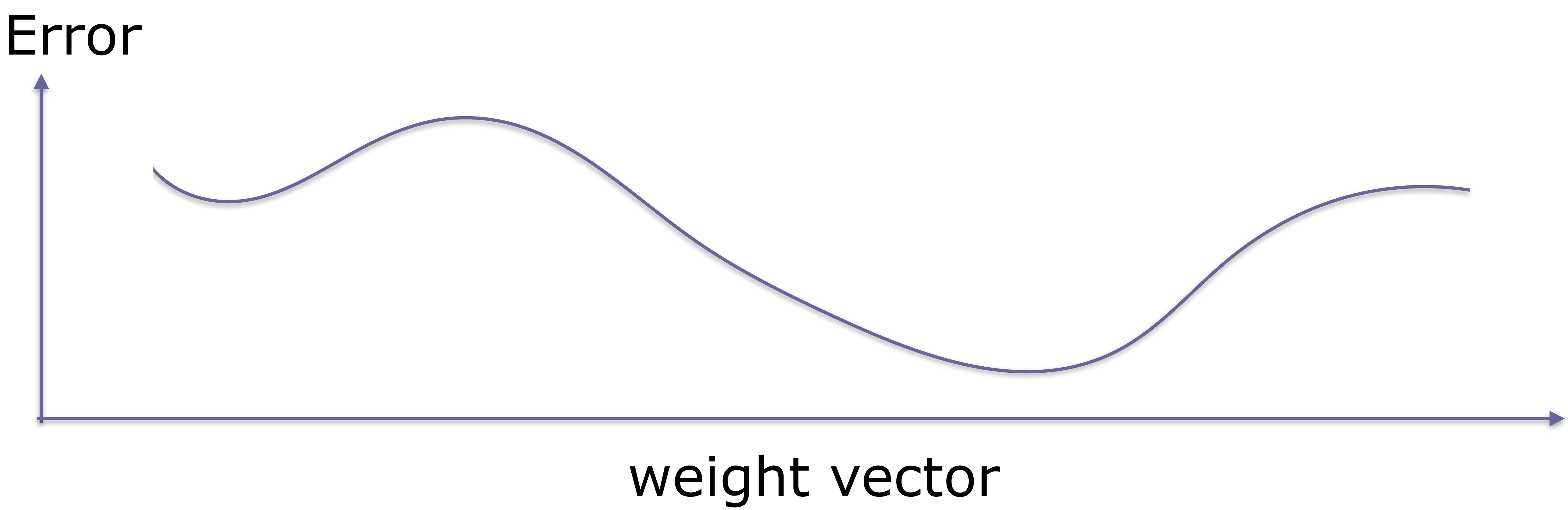
large W —> hidden to output weights

$$h_j = f\left(\sum_{i=0}^{I} w_{ji}\, i_i\right) = f(e_j)$$

small w —> input to hidden weights

Training the MLP

- Gradient Descent
- Error function is not convex
- Local minima



MLP with one hidden layer (mathematical equation)

O = number of output neurons
H = number of hidden neurons
I = number of inputs

$$o_k = f\left(\sum_{k=1,j=0}^{O,H} W_{kj}\, f\left(\sum_{j=1,i=0}^{H,I} w_{ji}\, i_i\right)\right)$$

output of k-th output neuron

hidden-to-output weights

input-to-hidden weights

Error function of MLP

$$E = \sum_{p=1,k=1}^{P,O} \frac{1}{2}\left(t_{pk} - o_{pk}\right)^2$$

$$E_{pk} = \frac{1}{2}\left(t_{pk} - o_{pk}\right)^2$$

$$E_{pk} = \frac{1}{2}\left(t_{pk} - f\left(\sum_{k=1,j=0}^{O,H} W_{kj}\, f\left(\sum_{j=1,i=0}^{H,I} w_{ji} i_{pi}\right)\right)\right)^2$$

Derivation of the generalised delta learning rule

(MLP with one hidden layer)

$$\Delta W_{kj} = -\eta \frac{\partial E}{\partial W_{kj}} = \eta \sum_{p=1}^{P} (t_{pk} - o_{pk}) f'(e_{pk}) h_{pj} = \eta \sum_{p=1}^{P} \delta_{pk} \, h_{pj}$$

the adaptation of the hidden to output weights is identica
to that of the perceptron weights (except for h <—> i)

Adaptation of input to hidden weights

- More complicated due to non-linearity of the hidden neuron transfer function

- Makes use of the chain rule

The chain rule may be written, in Leibniz's notation, in the following way. If a variable *z* depends on the variable *y*, which itself depends on the variable *x*, so that *y* and *z* are therefore dependent variables, then *z*, via the intermediate variable of *y*, depends on *x* as well. The chain rule then states,

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}.$$

https://en.wikipedia.org/wiki/Chain_rule chain rule

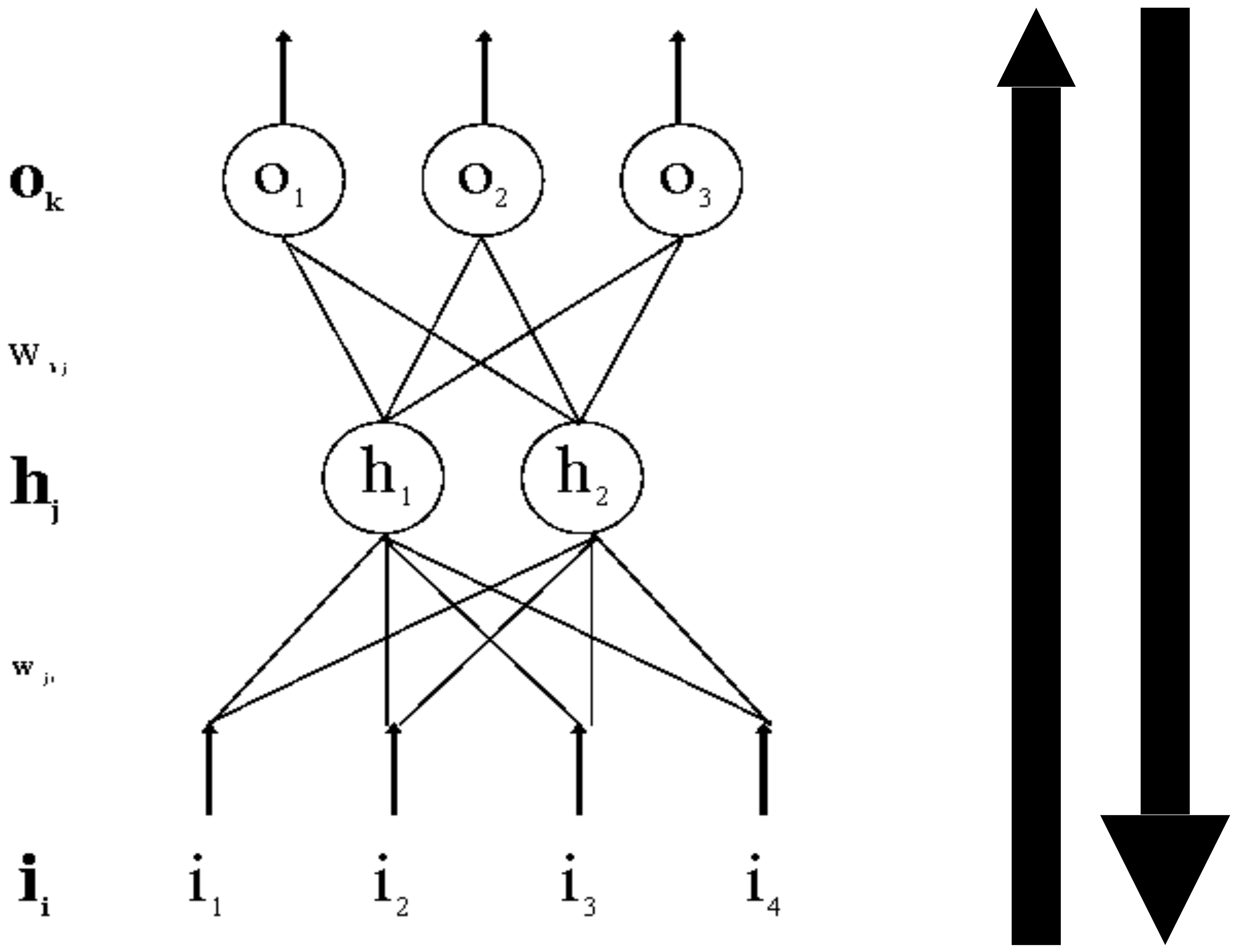
$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}} = -\eta \frac{\partial E}{\partial h_{pj}} \frac{\partial h_{pj}}{\partial w_{ji}}$$

$$= \eta \sum_{p=1,k=1}^{P,H} (t_{pk} - o_{pk}) f'(e_{pk}) W_{kj} f'(e_{pj}) i_{pi}$$

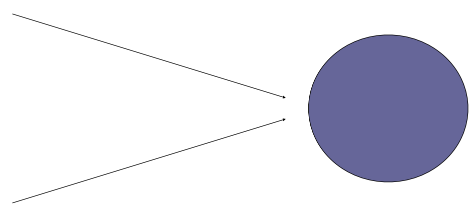
$$= \eta \sum_{p=1,k=1}^{P,H} \delta_{pk} W_{kj} f'(e_{pj}) i_{pi}$$

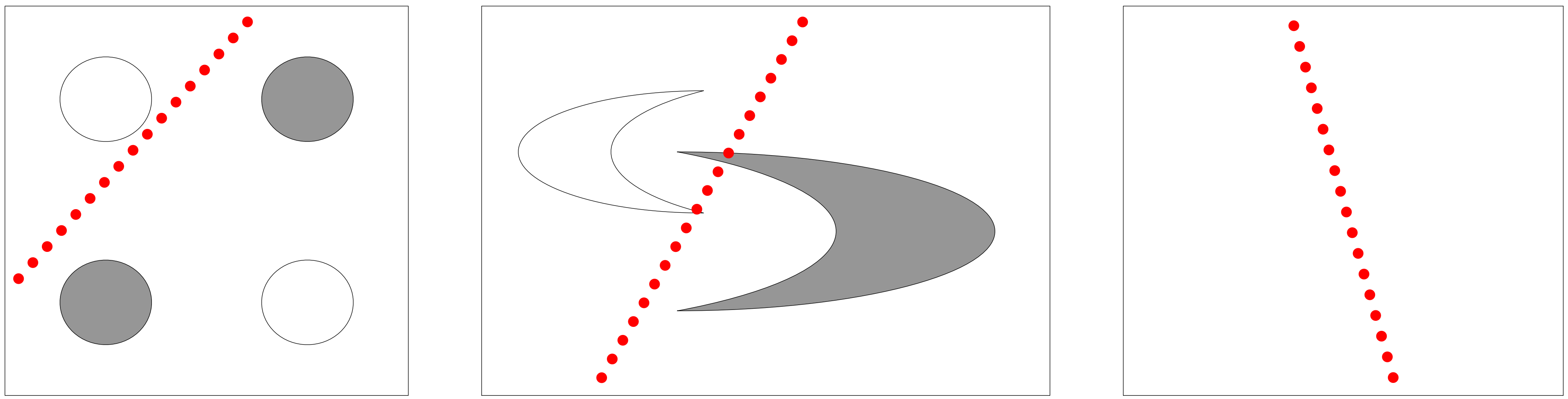
$$= \eta \sum_{p=1,k=1}^{P,H} \delta_{pj}\, i_{pi}$$

Forward and Backward Propagation



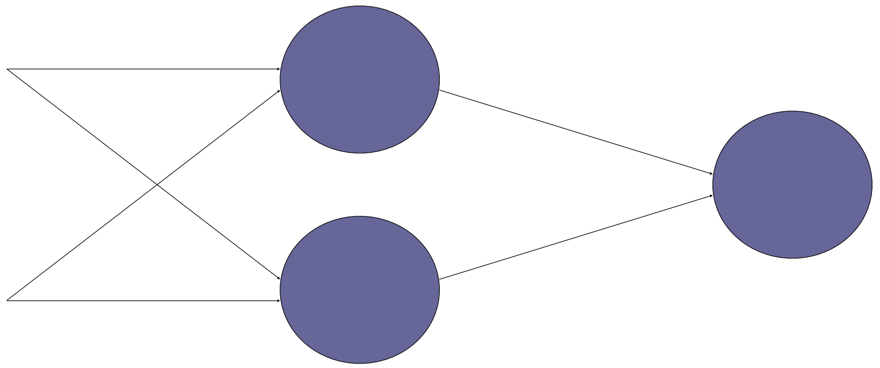
Decision boundaries of Perceptrons

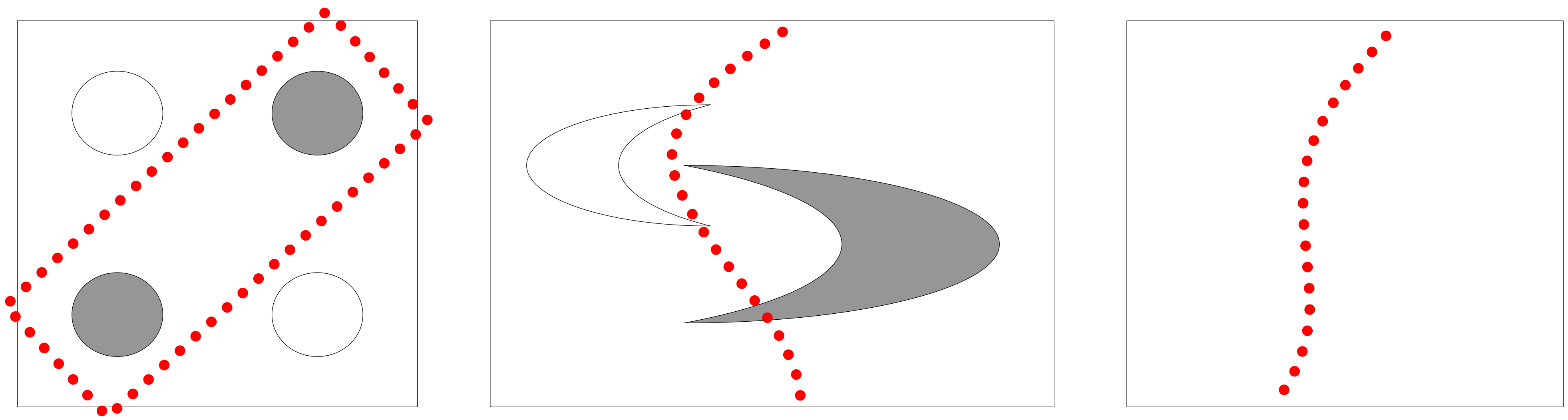




Straight lines (surfaces), linear separable

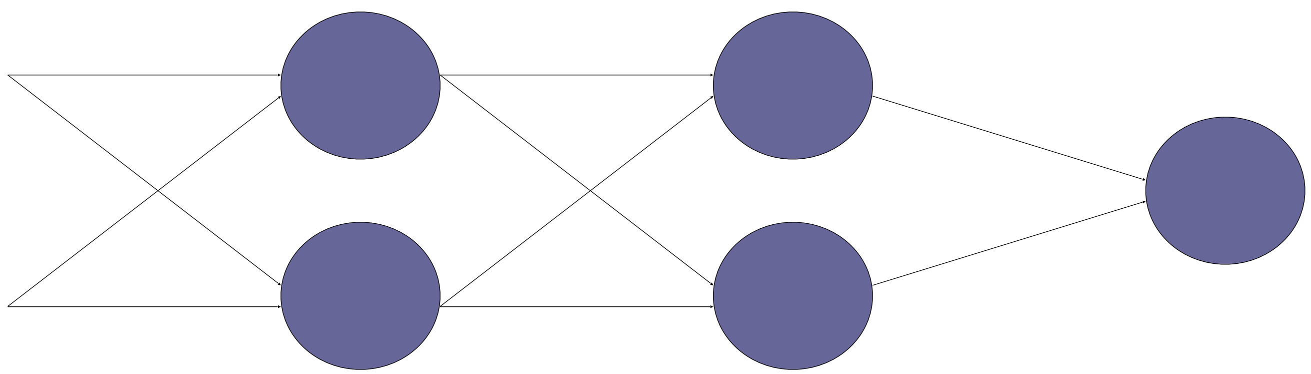
Decision boundaries of MLPs

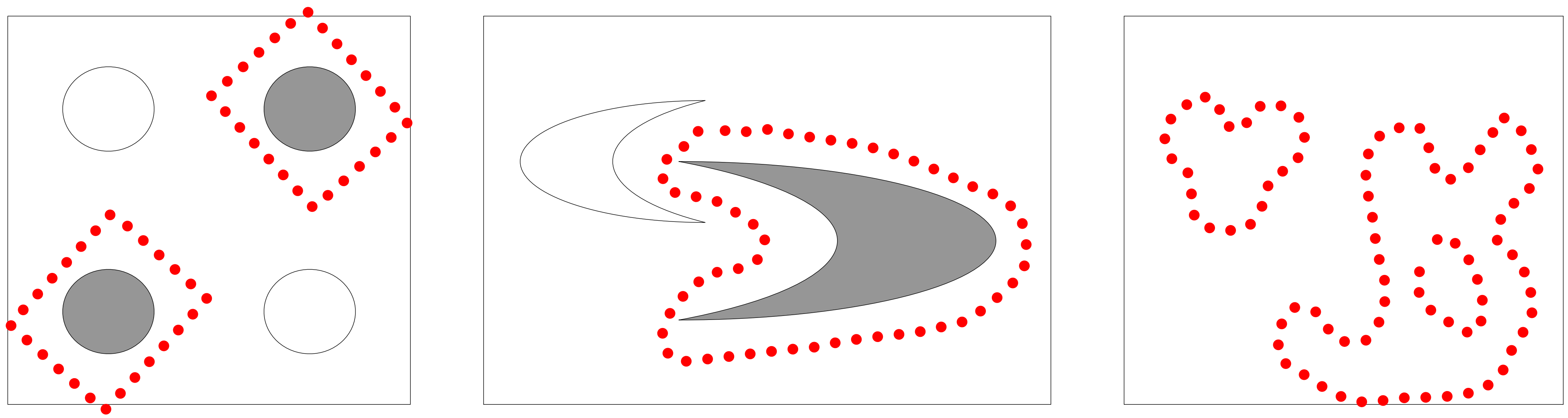




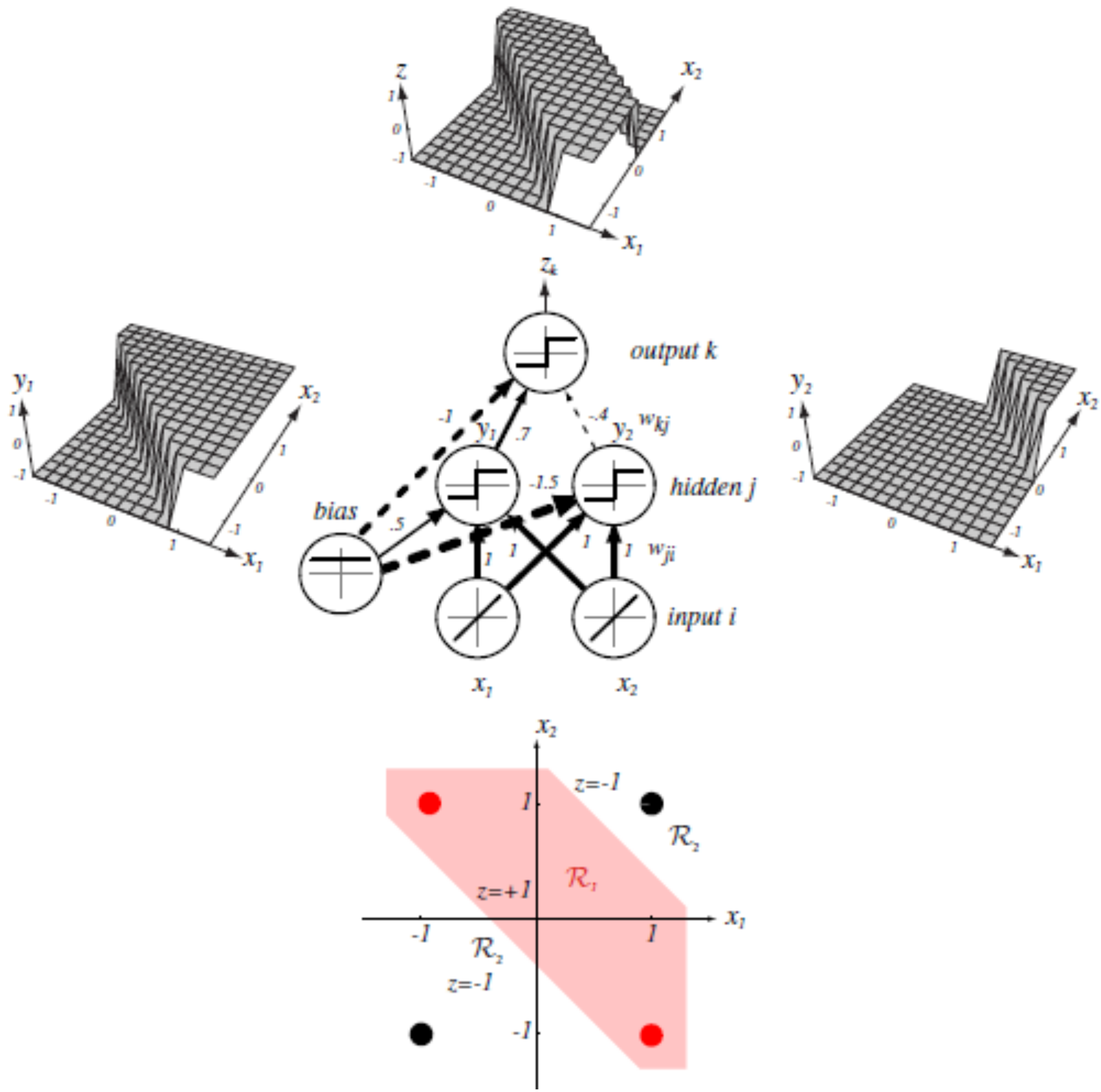
Convex areas (open or closed)

Decision boundaries of MLPs

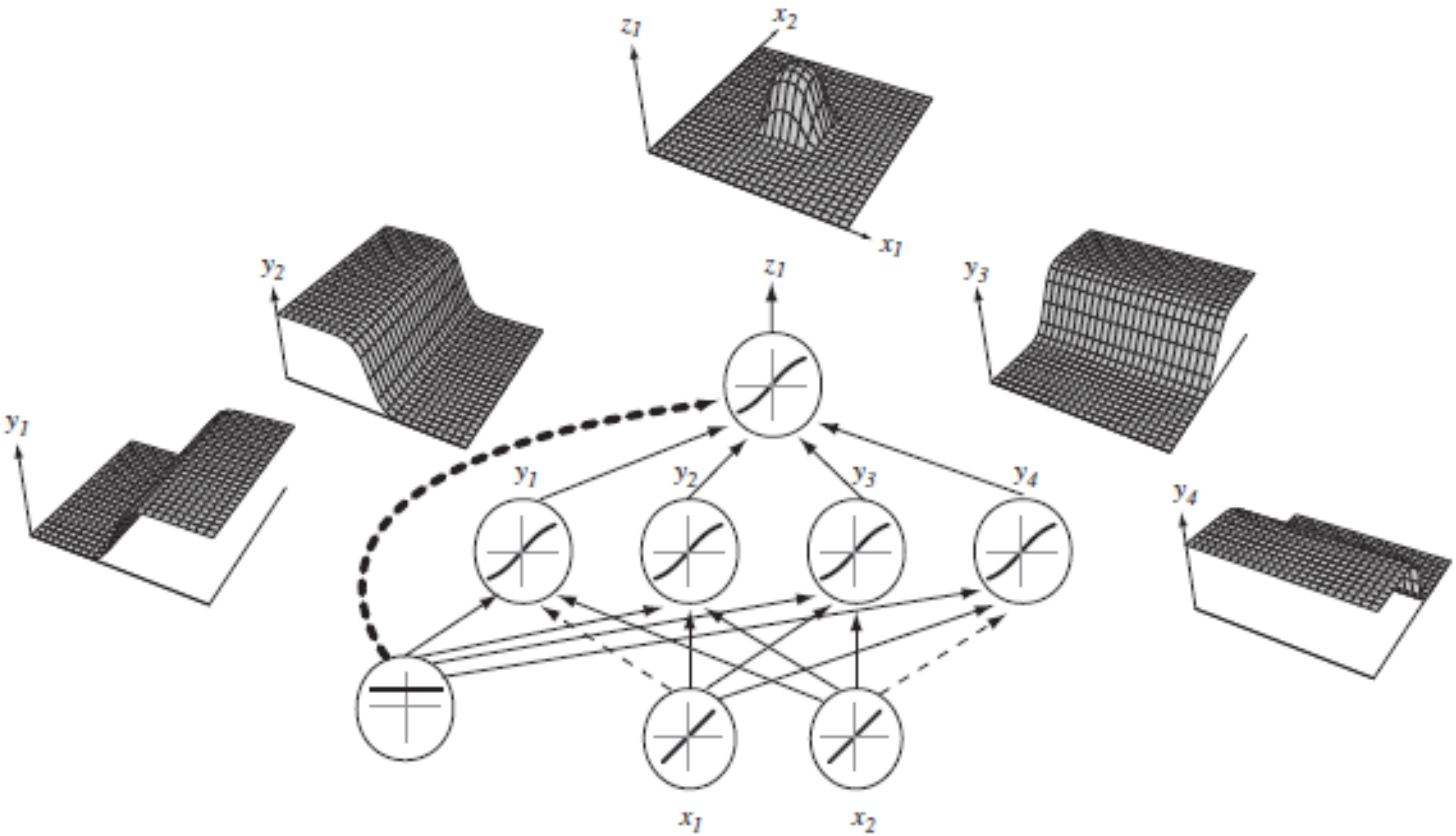




Combinations of convex areas



output k
hidden j
input i
bias
z_k
y_1
y_2
w_{kj}
w_{ji}
x_1
x_2
-1
.7
-.4
-1.5
.5
1
$z=-1$
$z=+1$
$\mathcal{R}_1$
$\mathcal{R}_2$



Soft thresholds (sigmoid functions) allow for smoother decision boundaries

Hands-on exercise 1

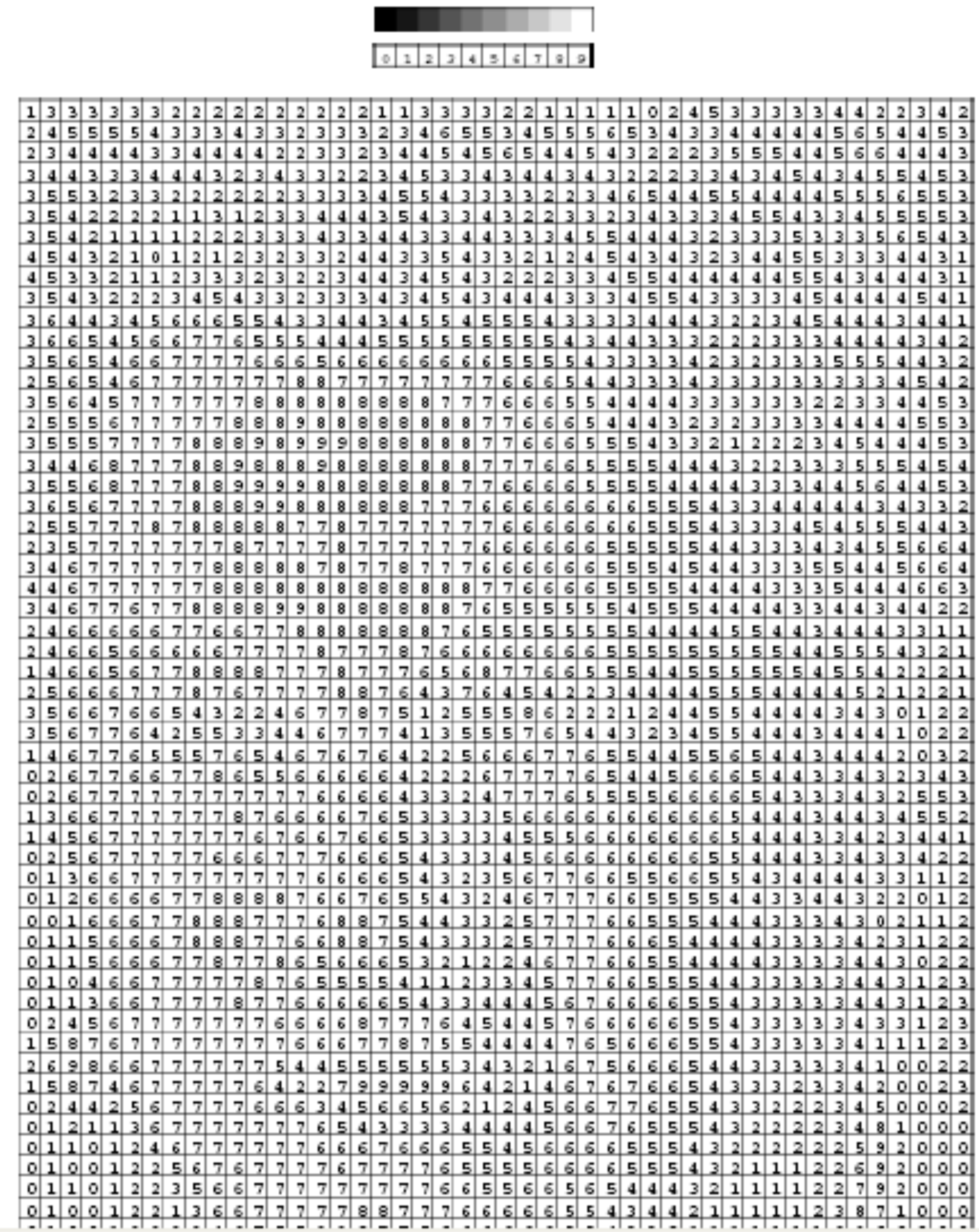
Preliminaries

About numbers and bits

- Computers represent numbers in terms of bits (zeros and ones)
- Binary versus decimal numbers
- Brains are not computers but they have (kind of) binary elements

Representing images by numbers

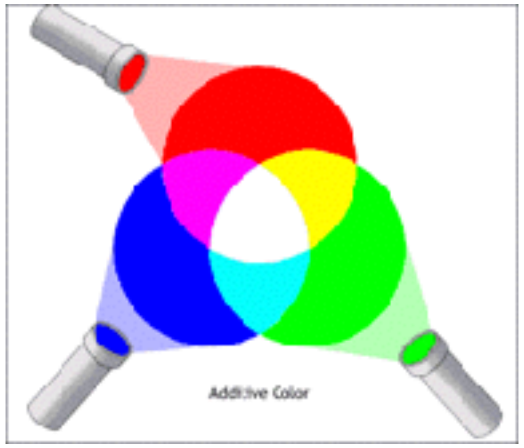
- In mathematics this is called a matrix (2D)
- If all numbers form a row or column it is called a vector (1D)



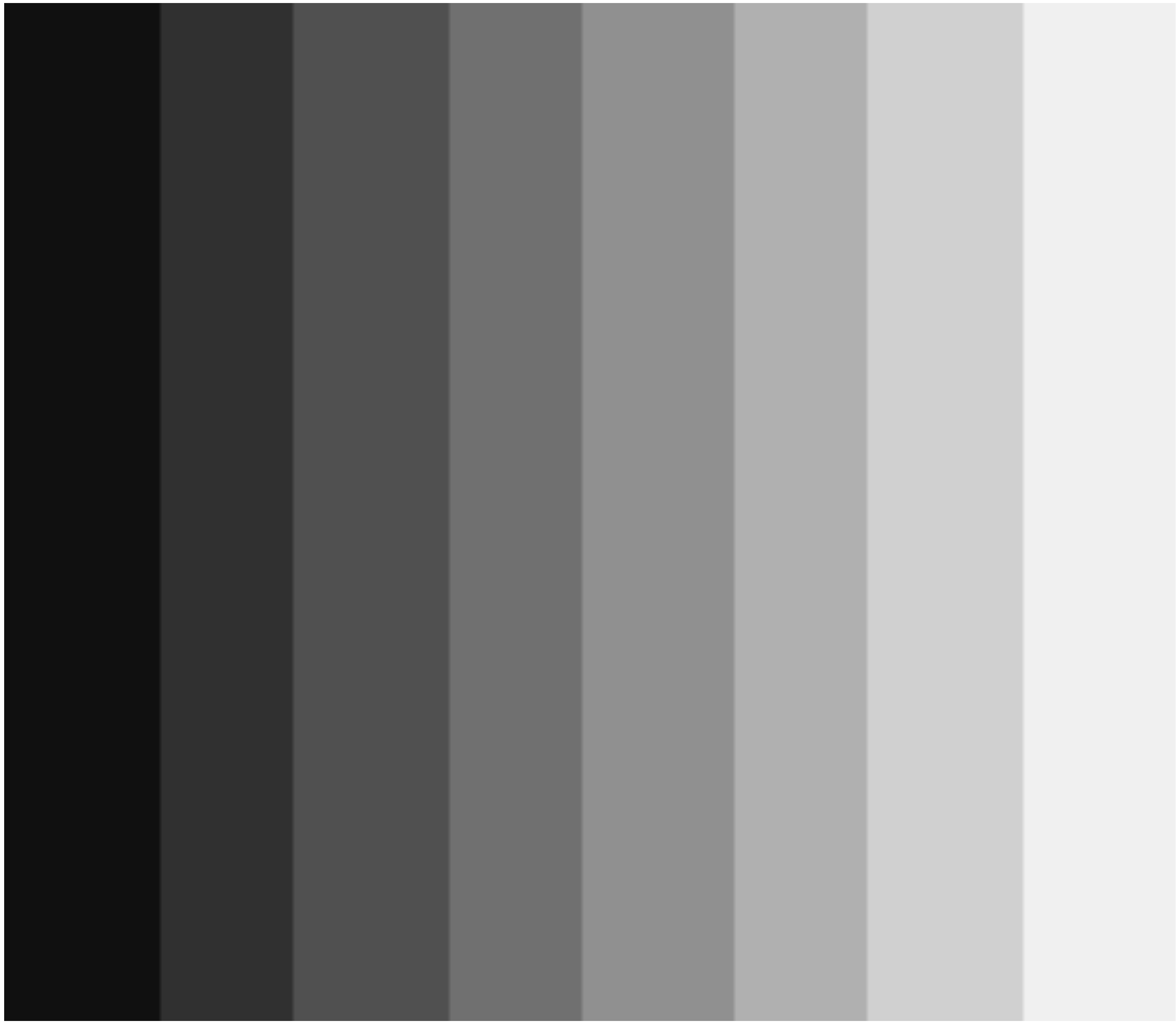
- The number of different gray levels depends on the number of bits
- 1 bits: 2 levels
  - black and white
- 2 bits: 4 levels
  - Black, gray, white
- 3 bits: 8 levels
  - Black, 6 grays, white
- 8 bits: 256 levels
  - Black, 254 grays, white
- N bits: 2^N levels
  - Black, 2^N-2 grays, white

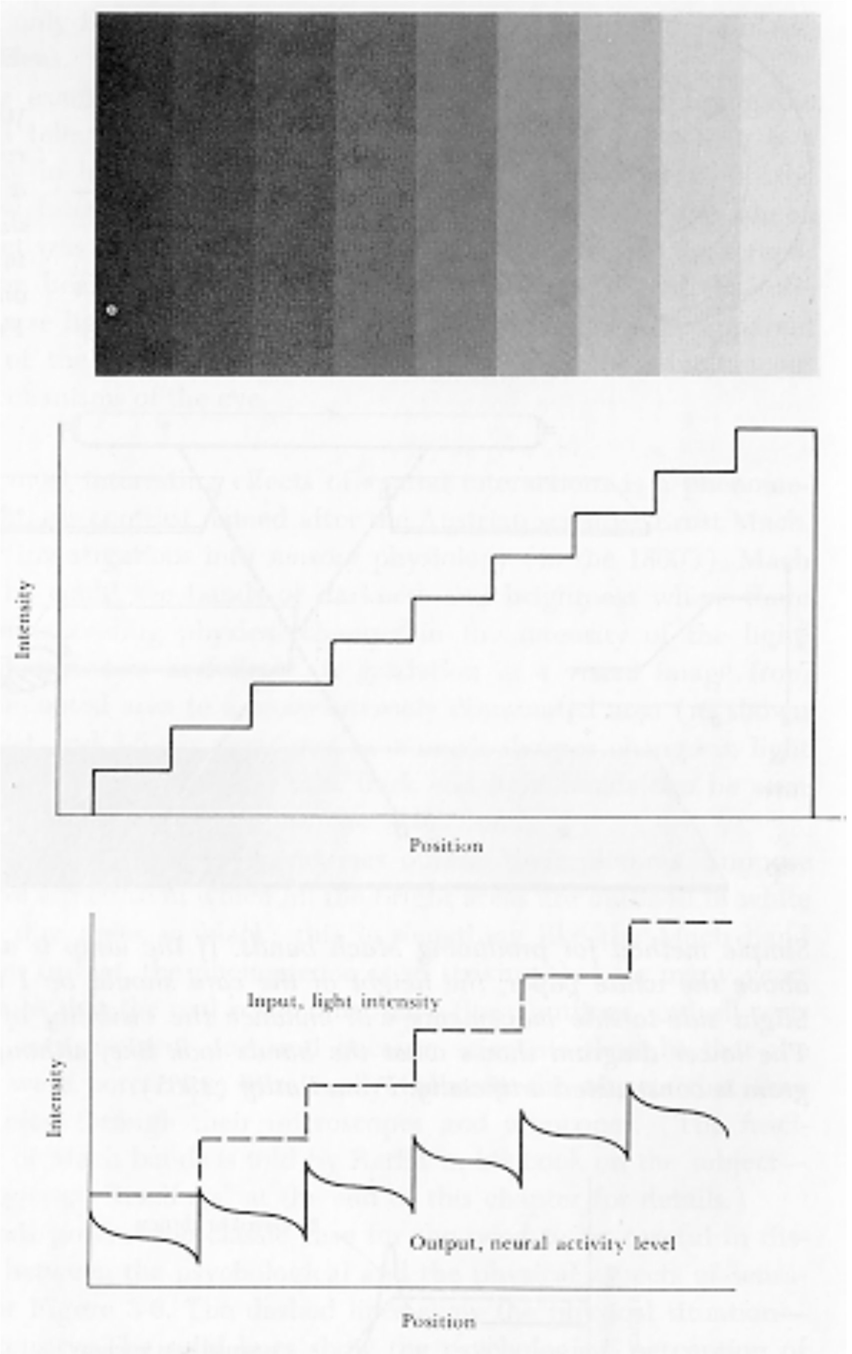
Color images

- Rows × columns × primary colors
- Number of primary colors is typically 3
- Red, Green, Blue
- Each represented by 8 bits
  - 256 × 256 × 256 = 16.777.216 colours

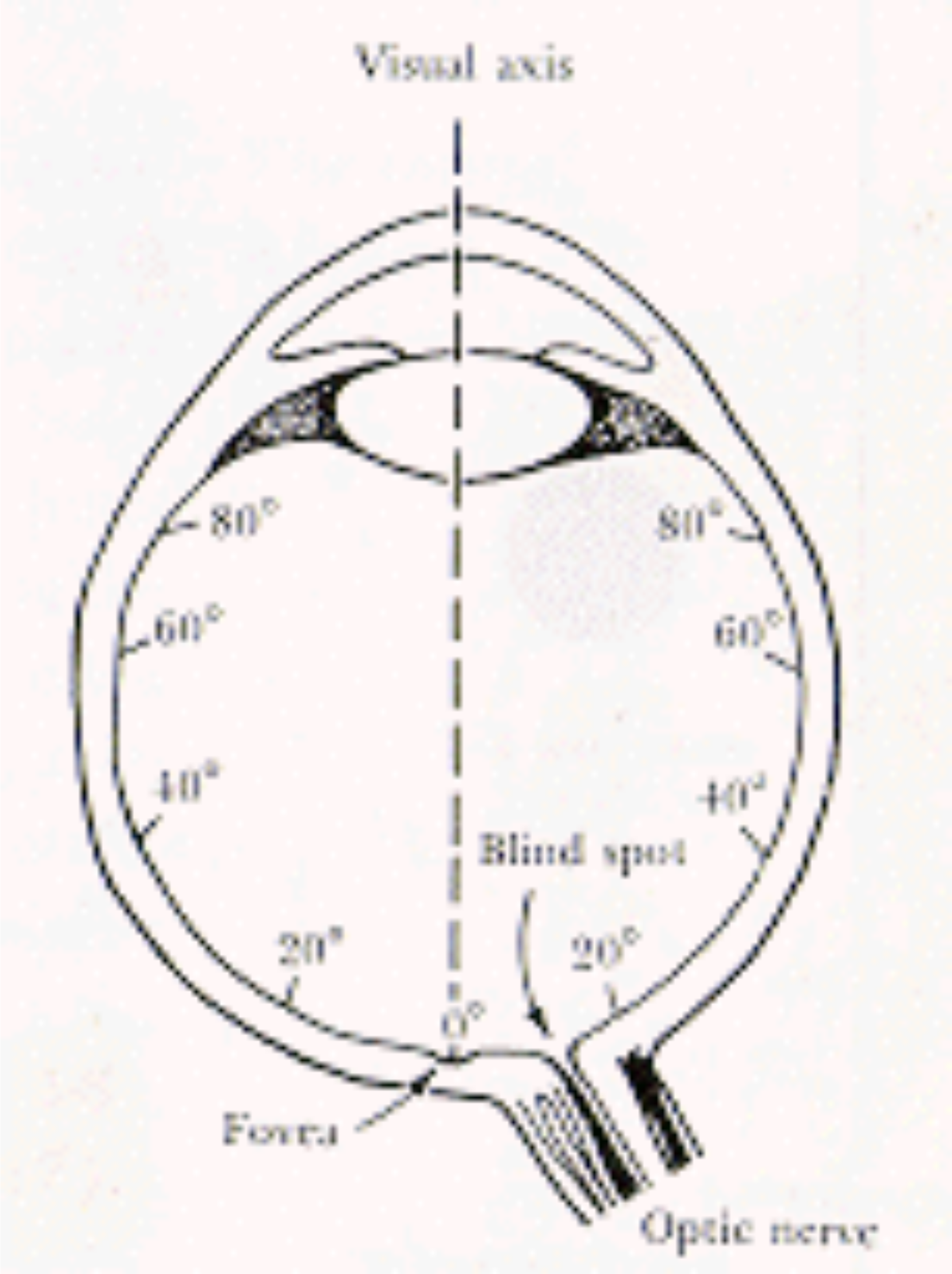


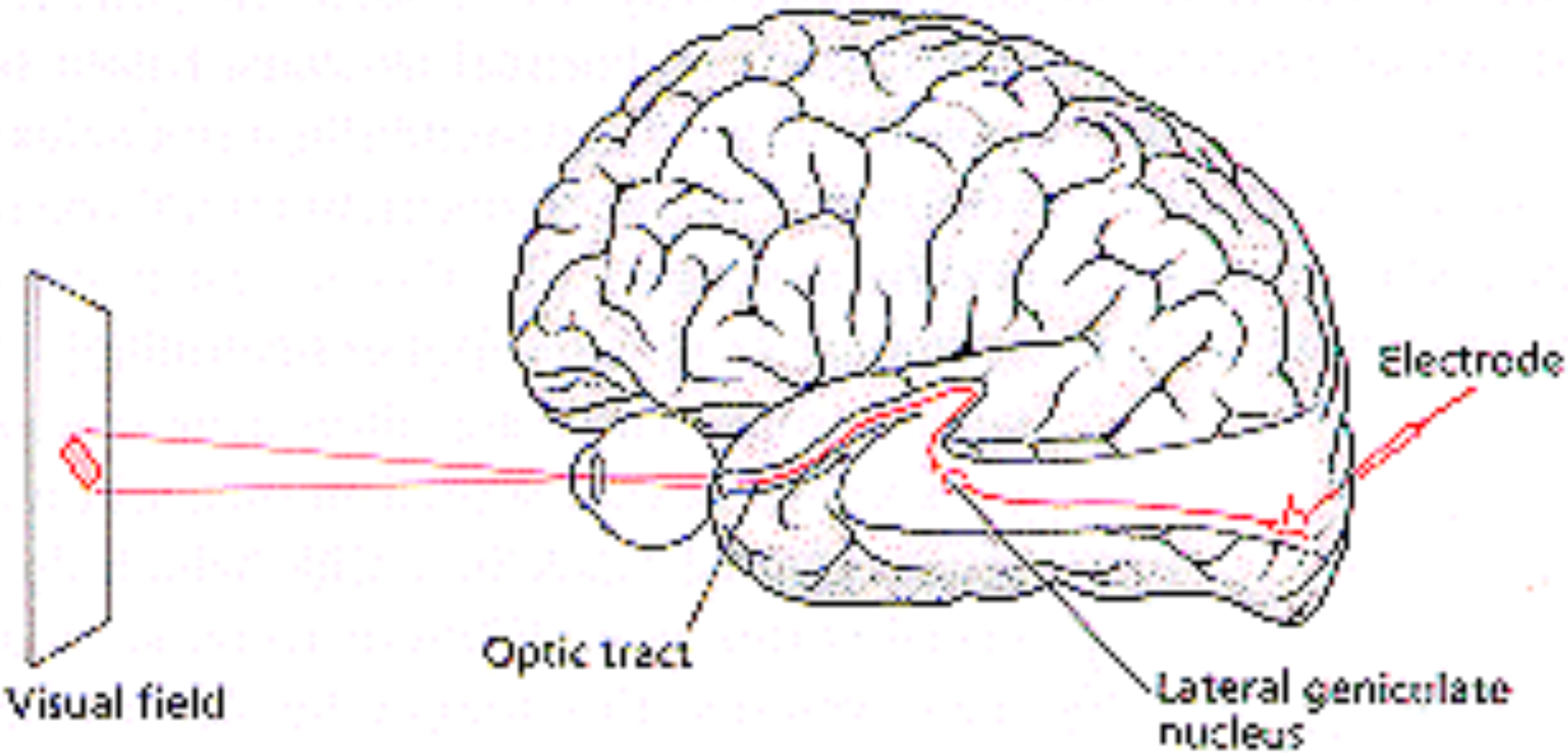
What you see is **not** what you get



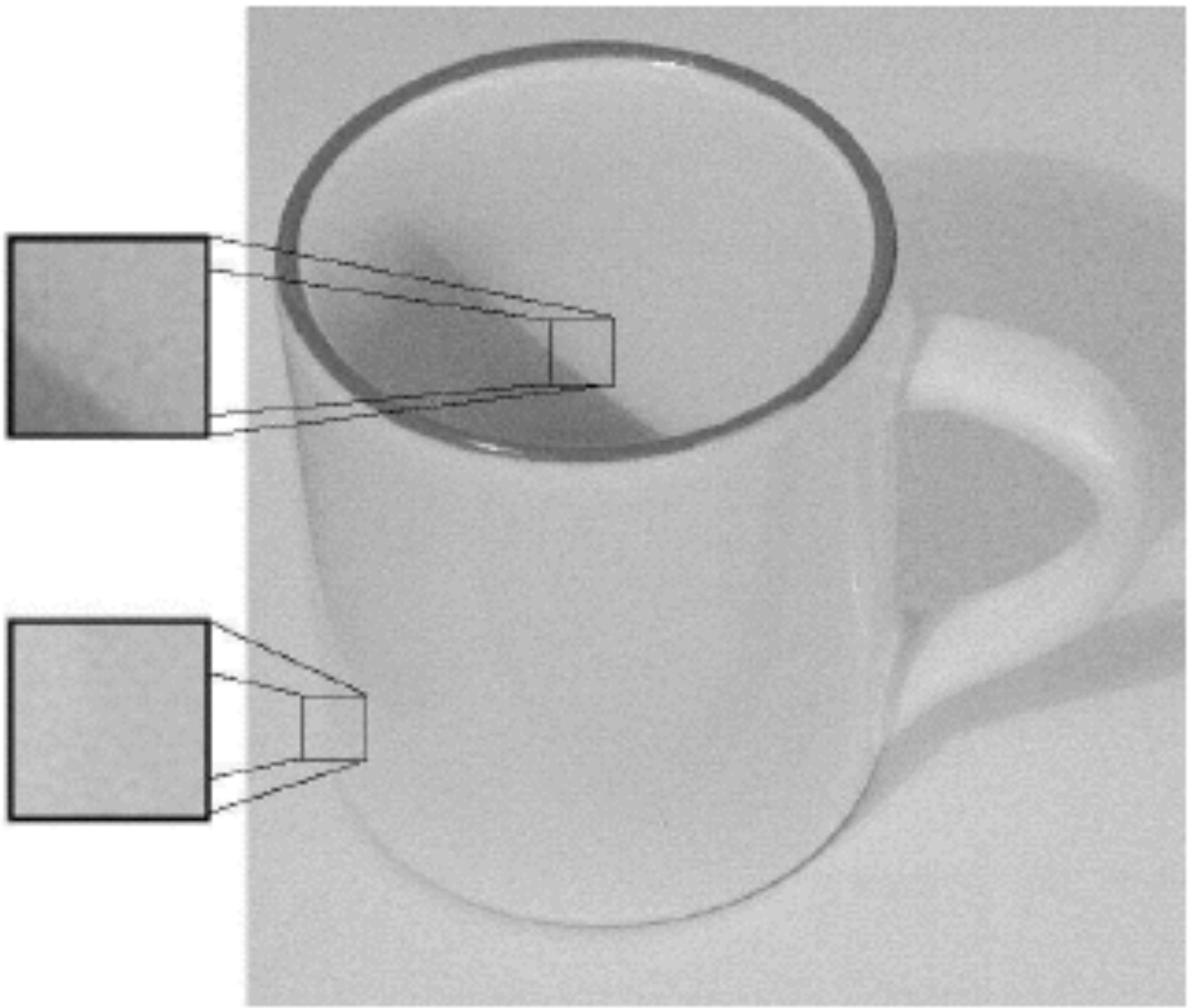


The eye and visual degrees

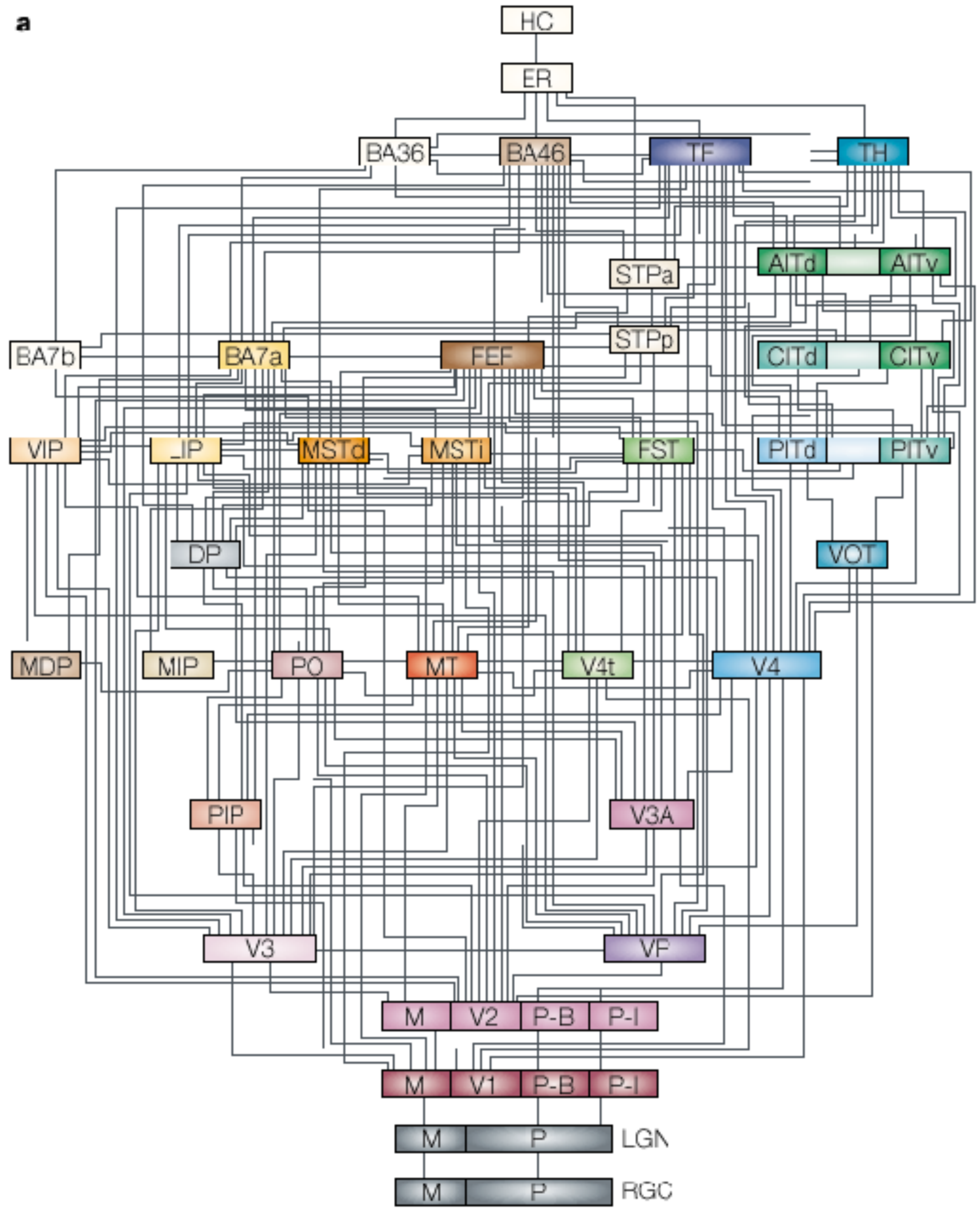



Electrode
Optic tract
Visual field
Lateral geniculate nucleus

Contours ("edges") are important

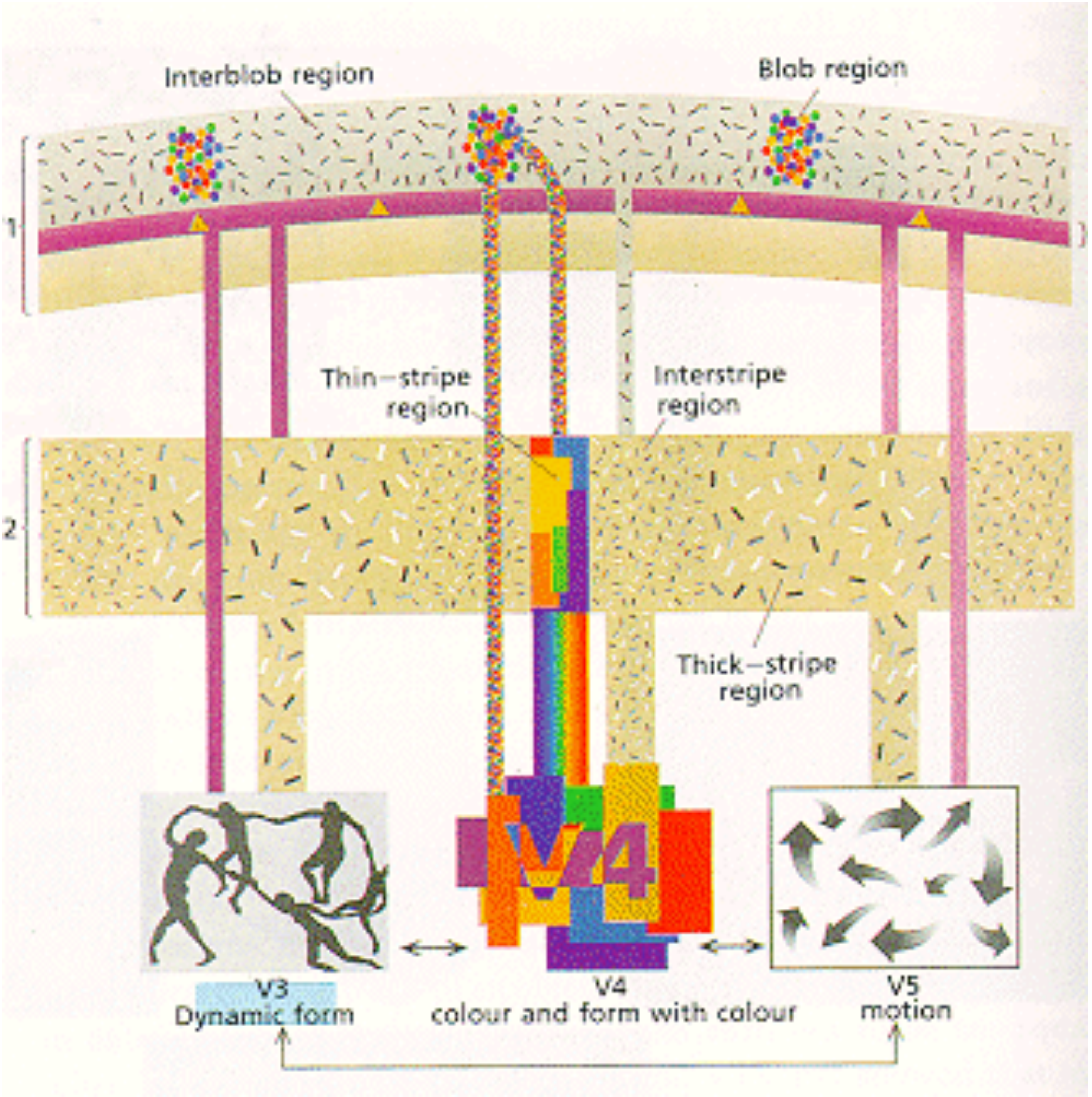


The Visual System



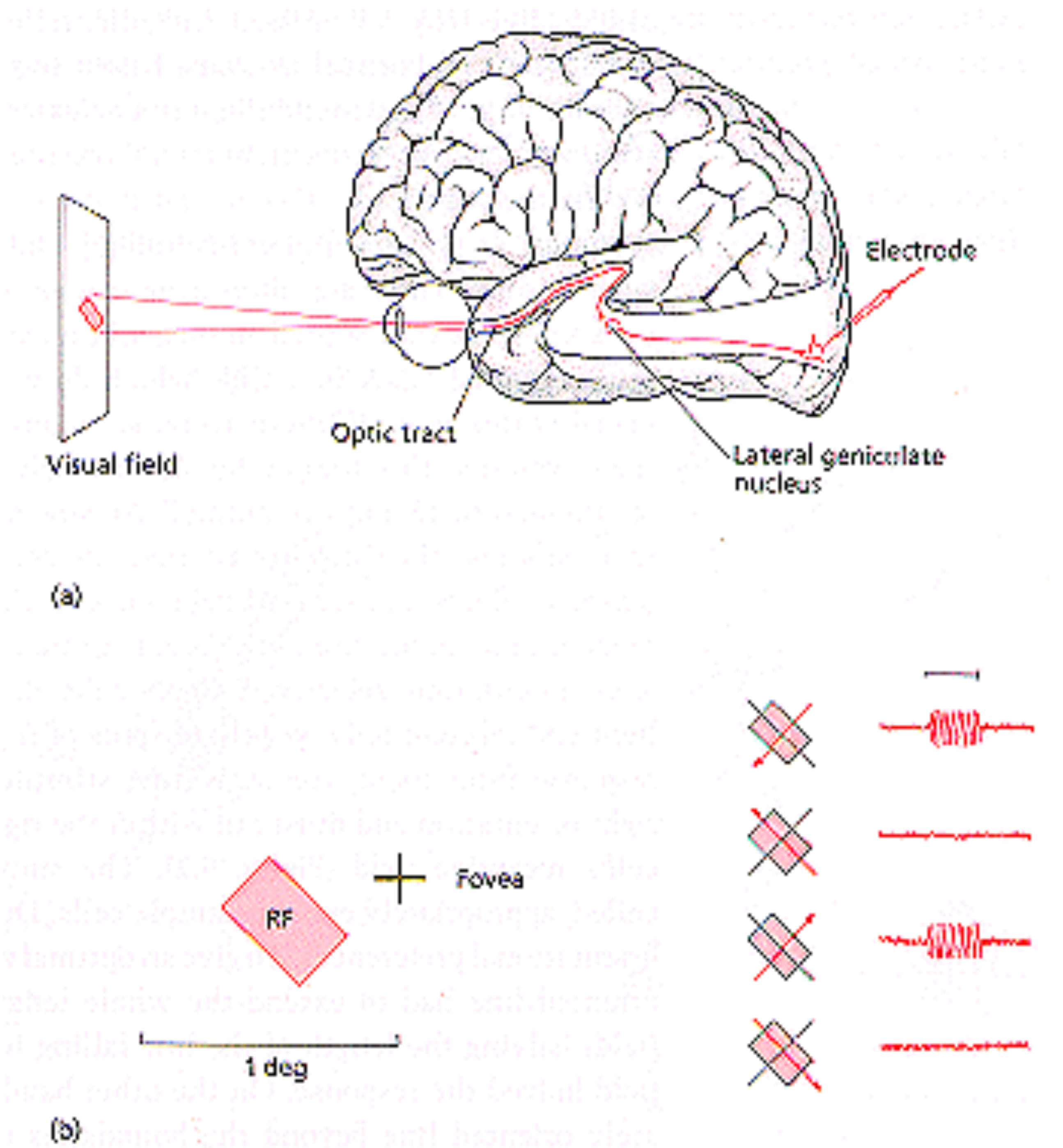
Multiple feature analysis

- V1
- V2



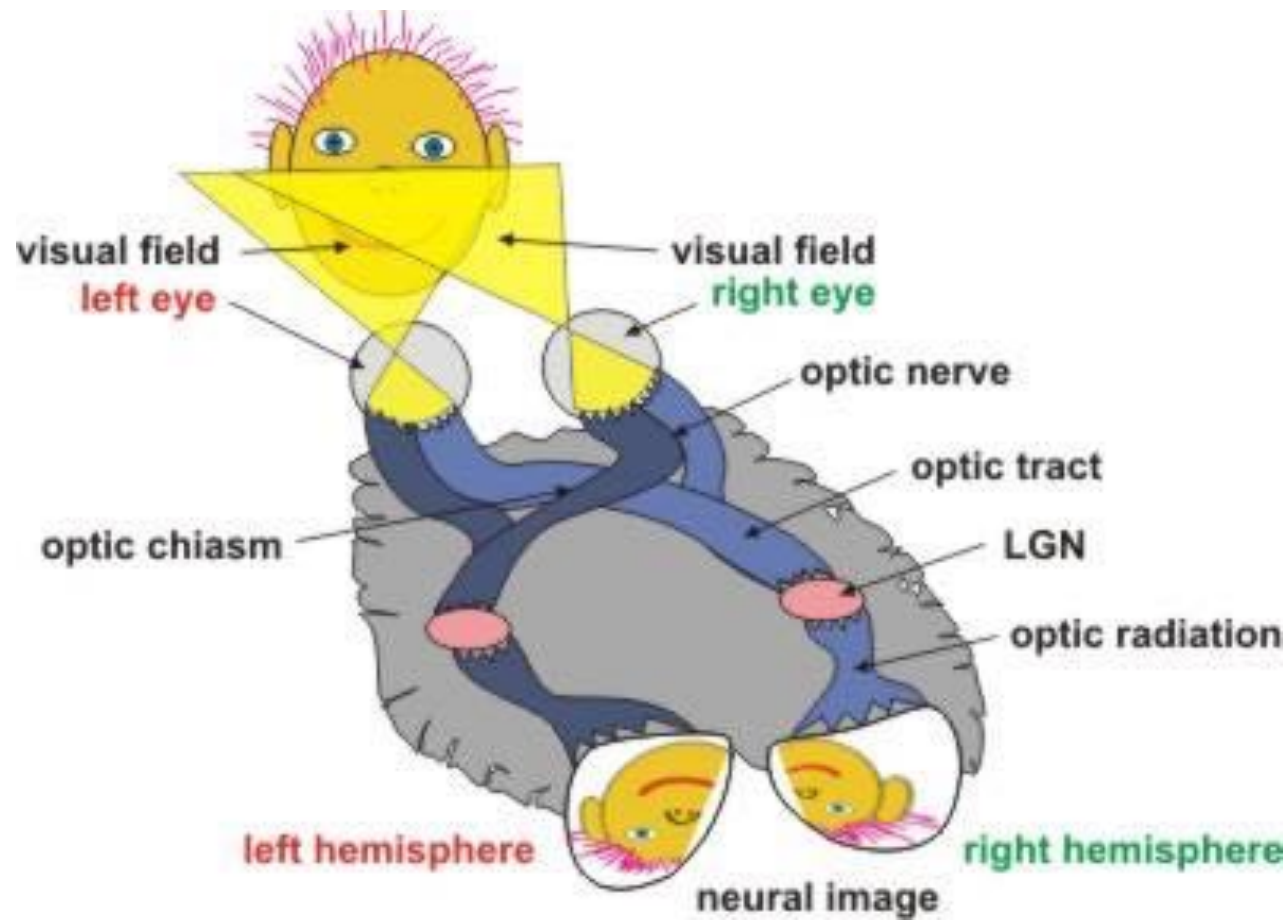
Feature analysis in V1 (and V2)

- orientation selectivity
  – Hubel & Wiesel
- color selectivity
- direction of motion selectivity
- size (spatial-frequency) selectivity
- texture selectivity



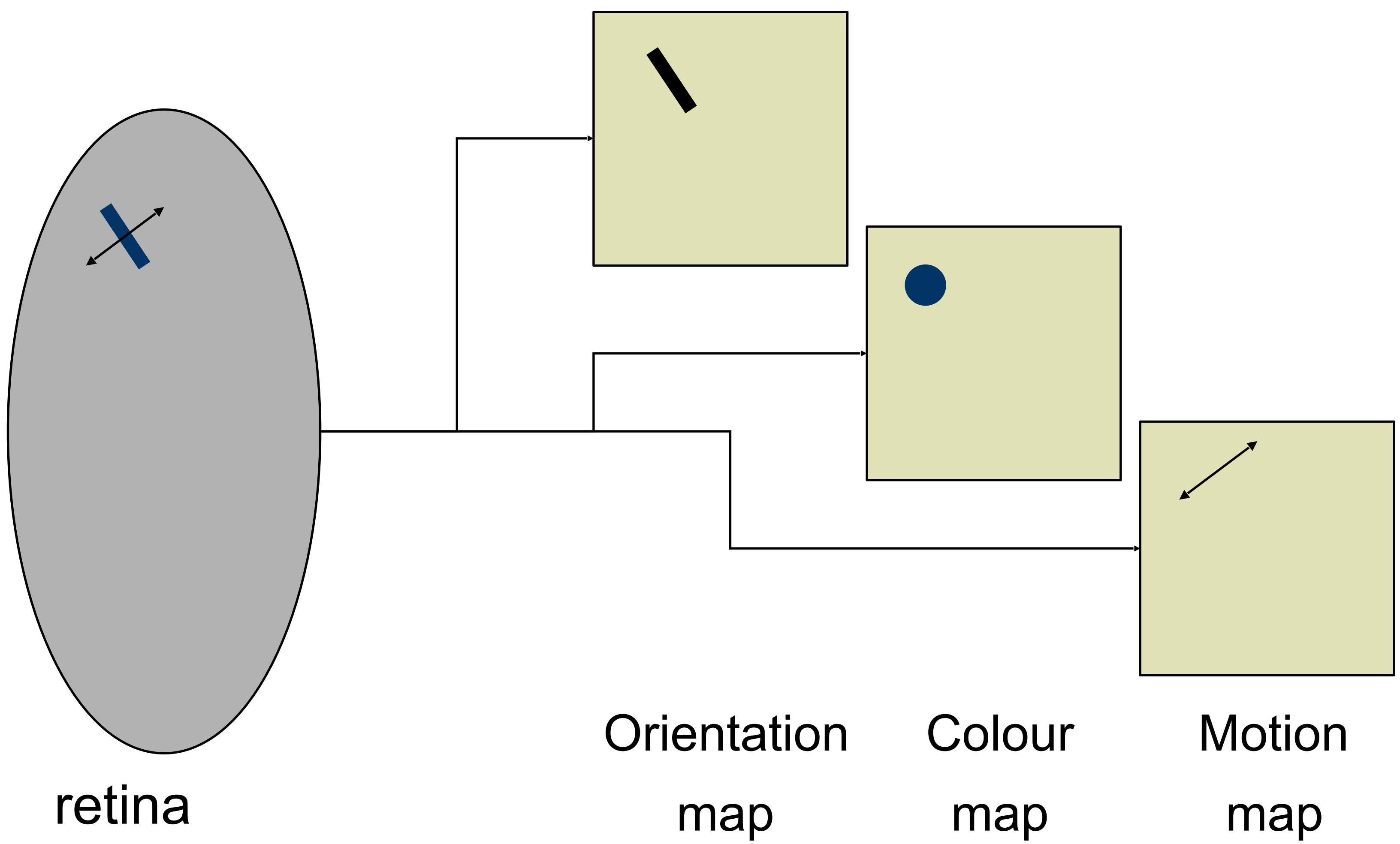
Feature maps in the visual system

- For each visual "point" (= retinal receptor)
  - "detector-cells" in all orientations
- Retinotopic organization
  - V1 neurons are spatially organized in a way that reflects the spatial organization of the retina



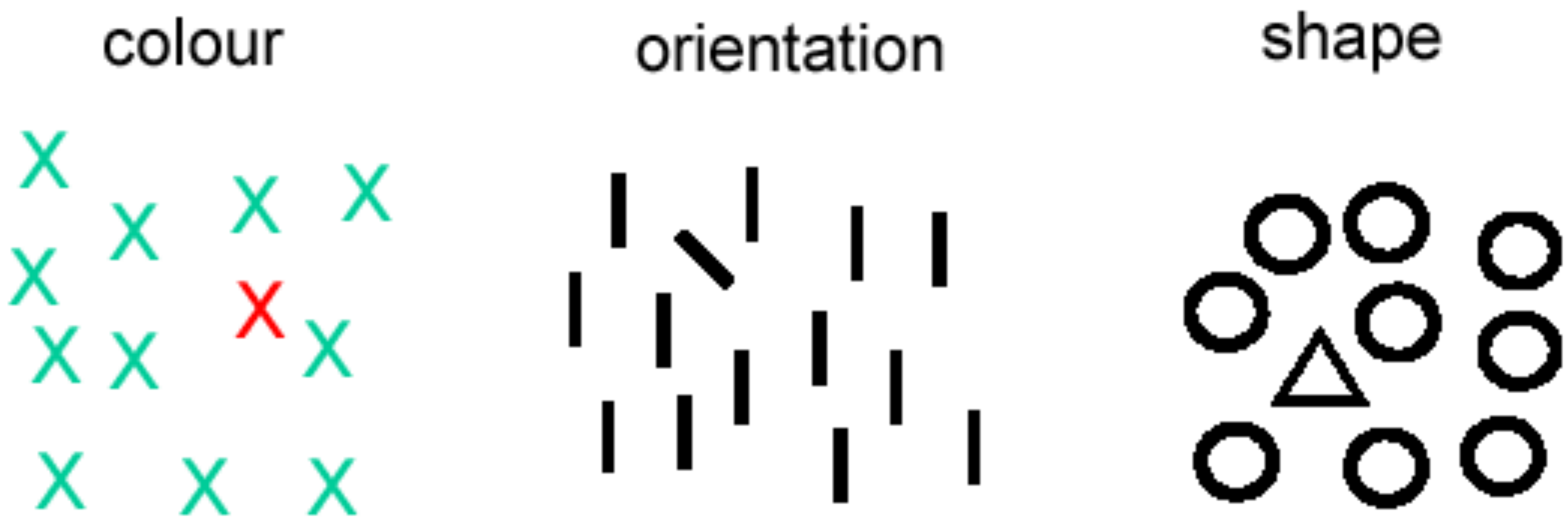
Retinotopic maps in the visual system

- The visual system has many retinotopically-organised feature maps



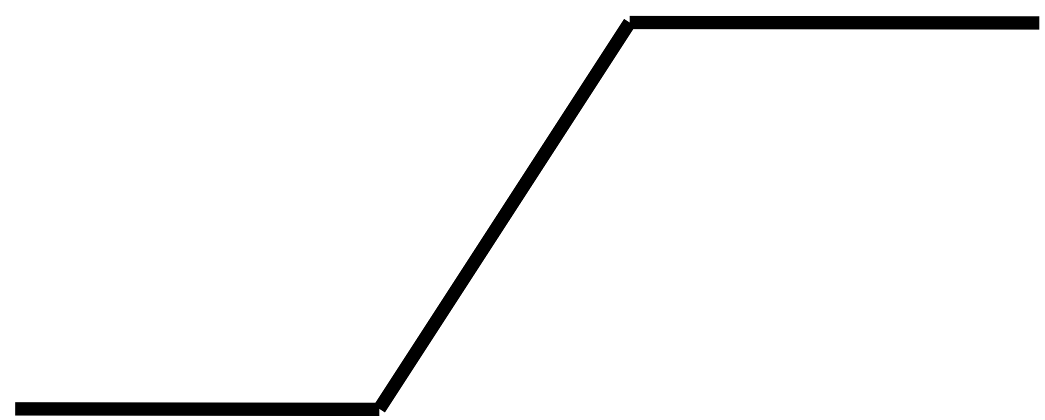
What happens in feature maps?

- Very similar to what current computer vision systems do
- Look for “features” at all locations of the retinal image
- Features: edges, oriented contours, colours, textures, …



Receptive fields

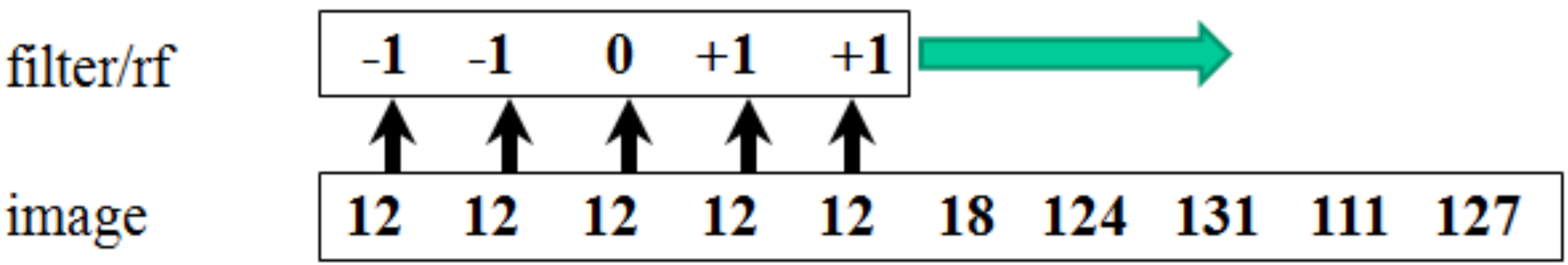
- “Edge detector” (dark to light)



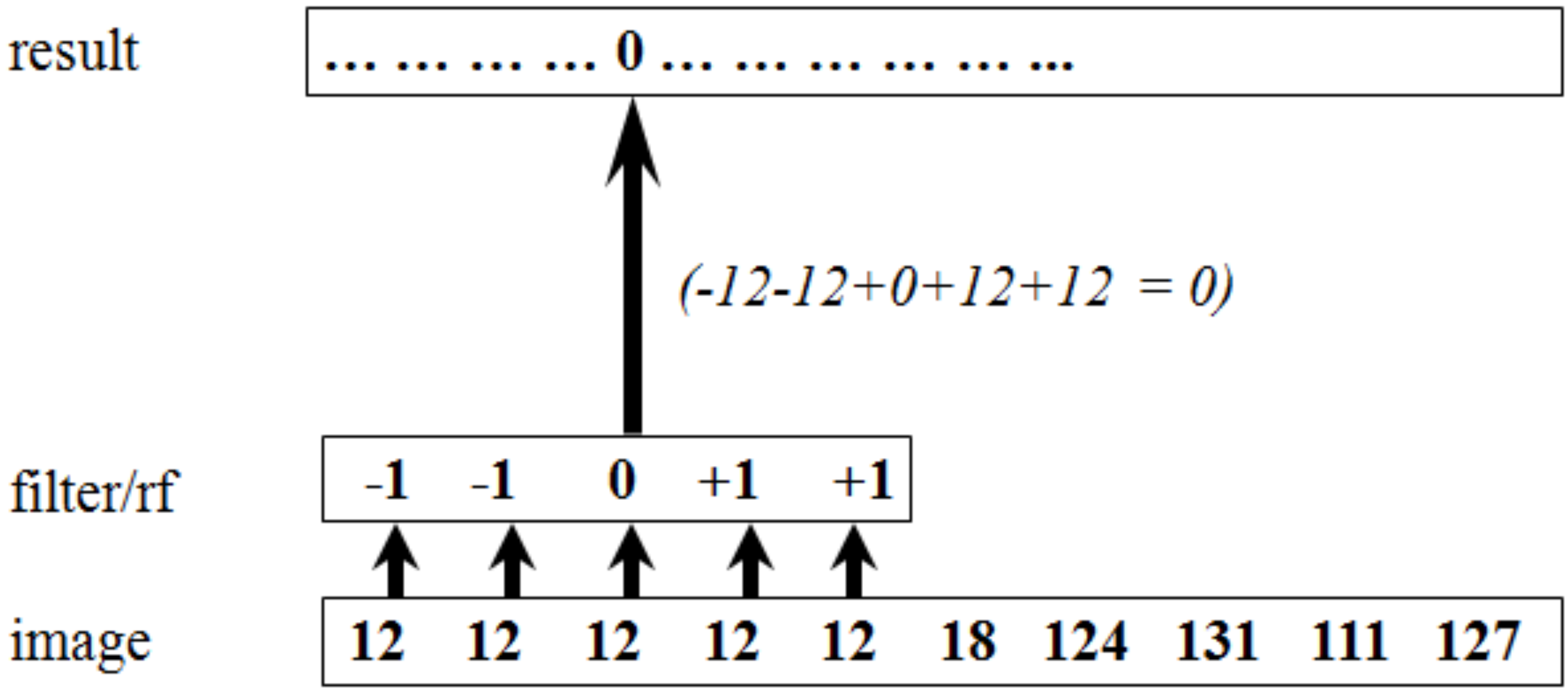
-1	-1	0	+1	+1

Filter/receptive field

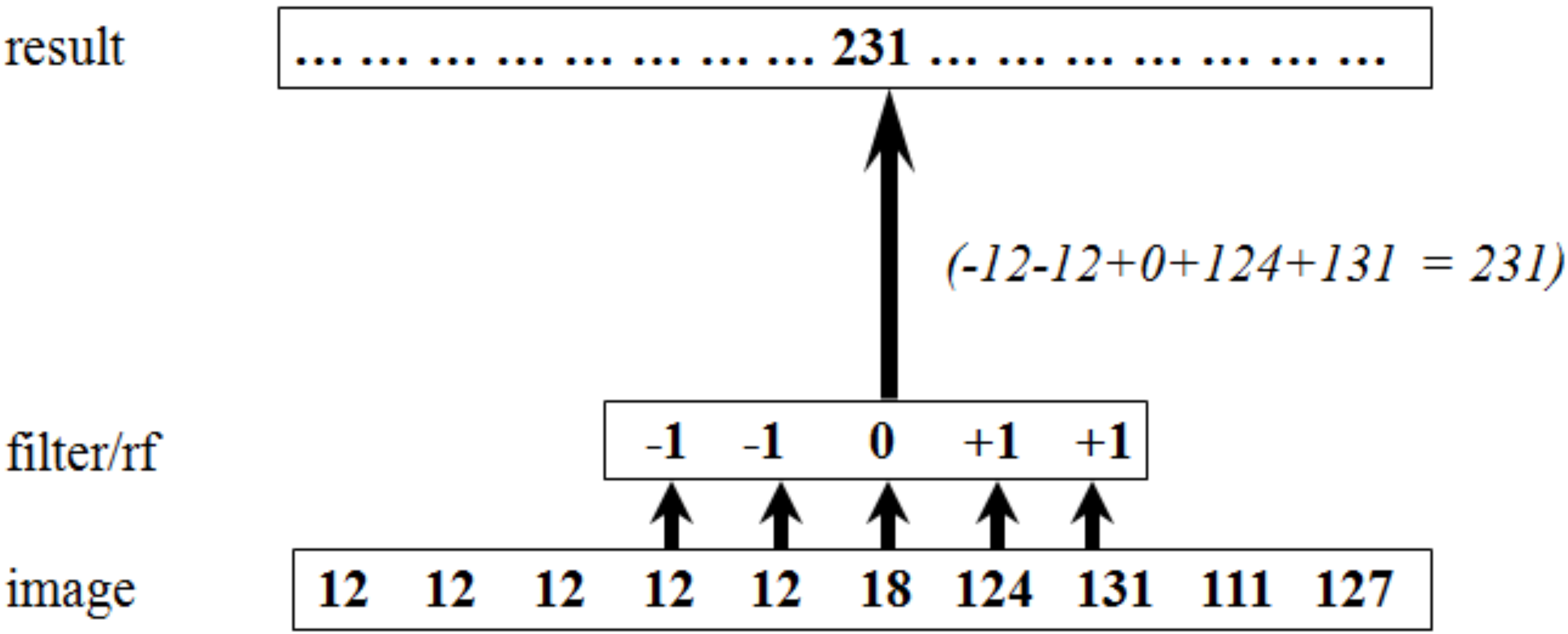
- A filter contains values (coefficients) each of which is multiplied with the corresponding pixel value in the image



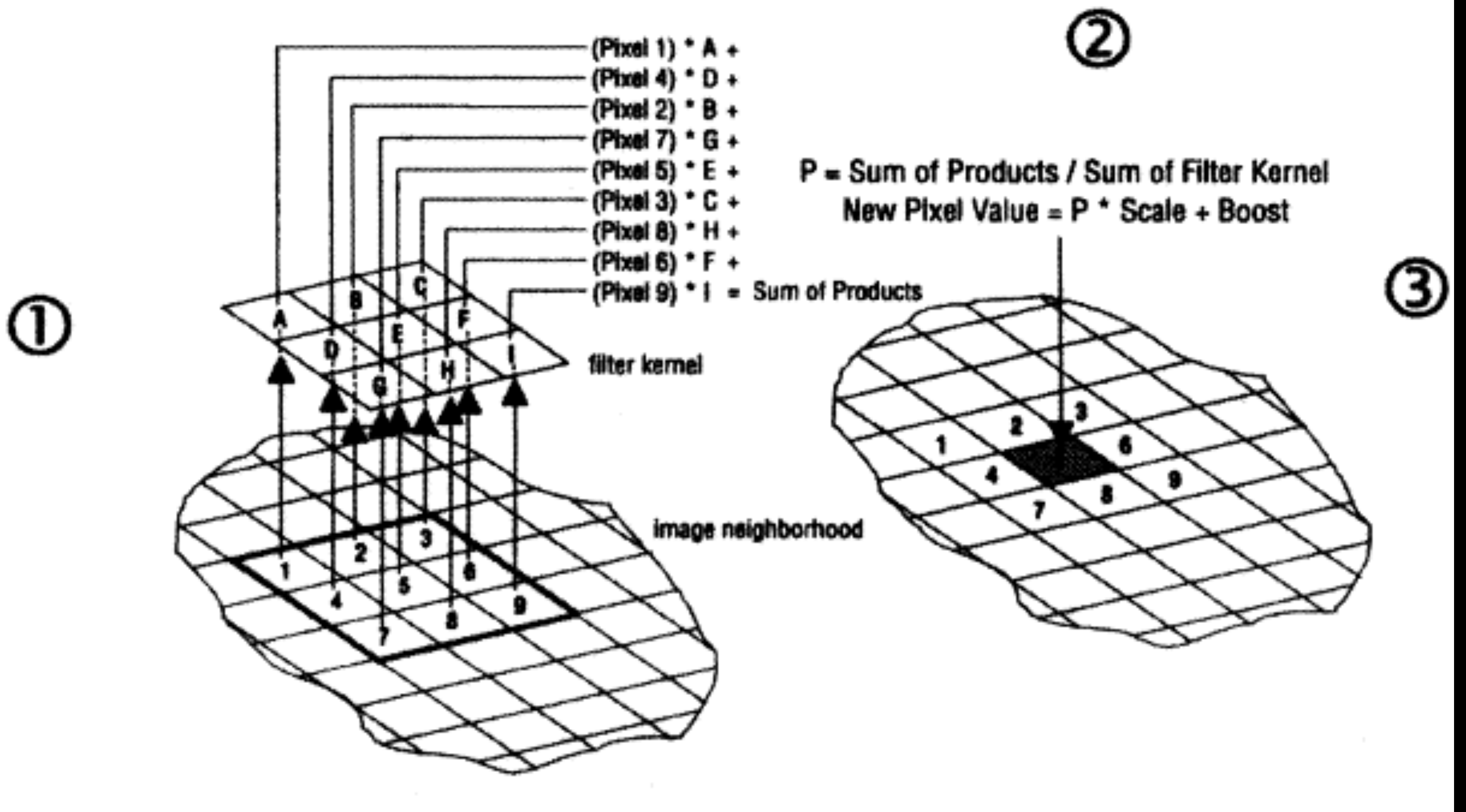
- Homogeneous image region: no edge detected



- Inhomogeneous image region: edge detected



Convolution



The Convolution Filtering Process

Computer Vision: Sobel edge detector

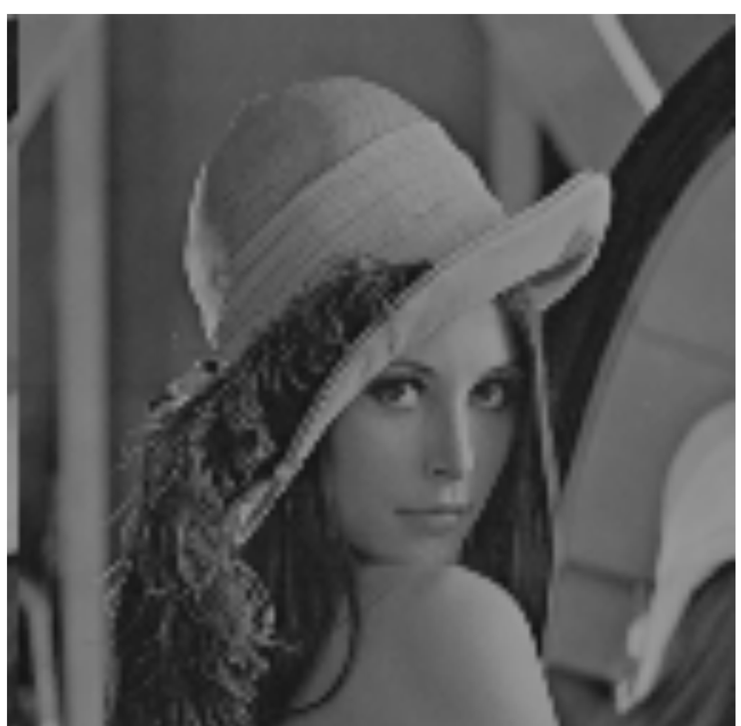
- Applies 2 filters (1 vertical and 1 horizontal)

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy



Laplace edge detector

- Applies a 5 x 5 filter to each image location

-1	-1	-1	-1	-1
-1	-1	-1	-1	-1
-1	-1	24	-1	-1
-1	-1	-1	-1	-1
-1	-1	-1	-1	-1

Canny edge detector

1. Gaussian blurring

2. Sobel edge detector

Steps 3-6: tracing edges

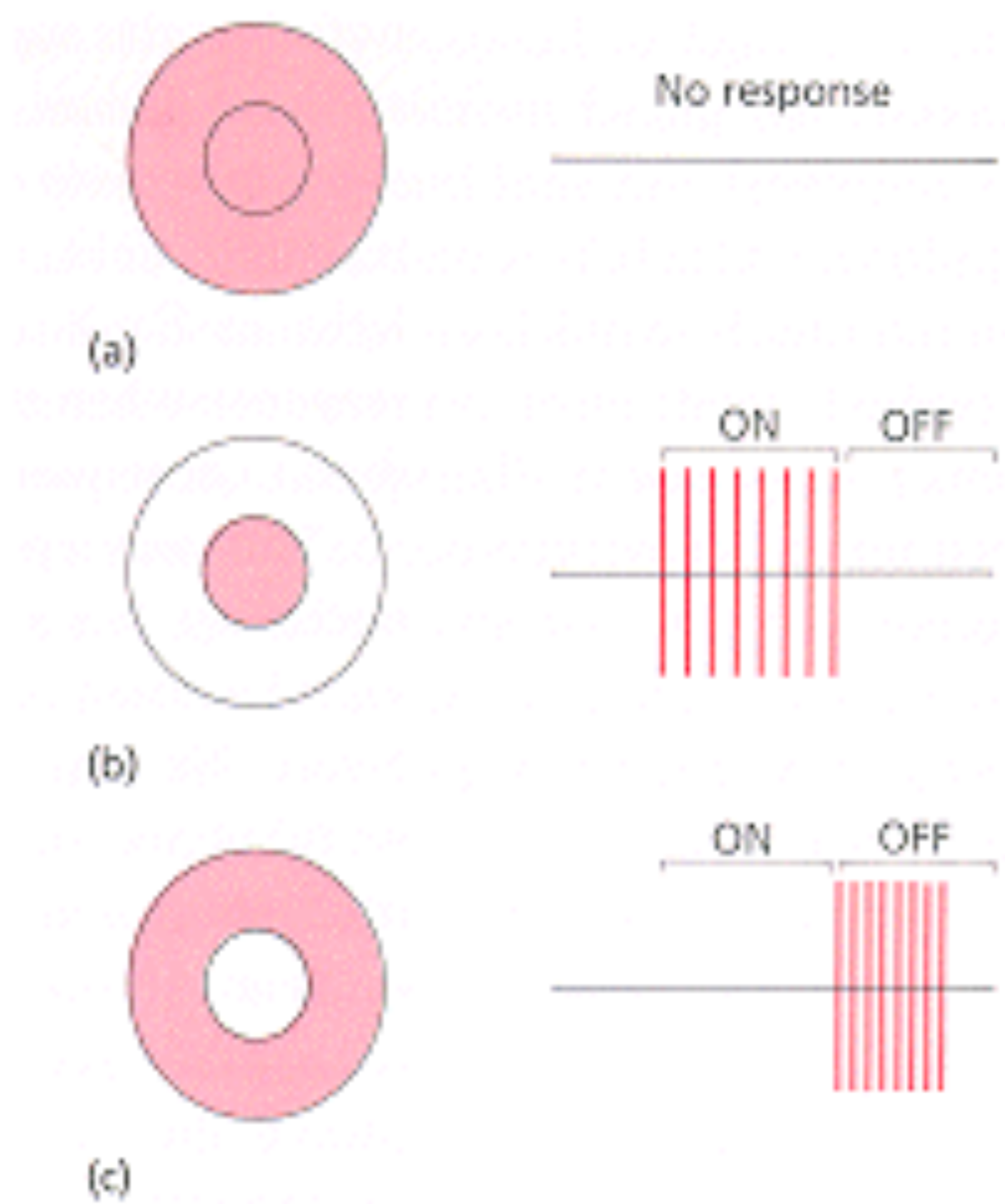
$$\frac{1}{115}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

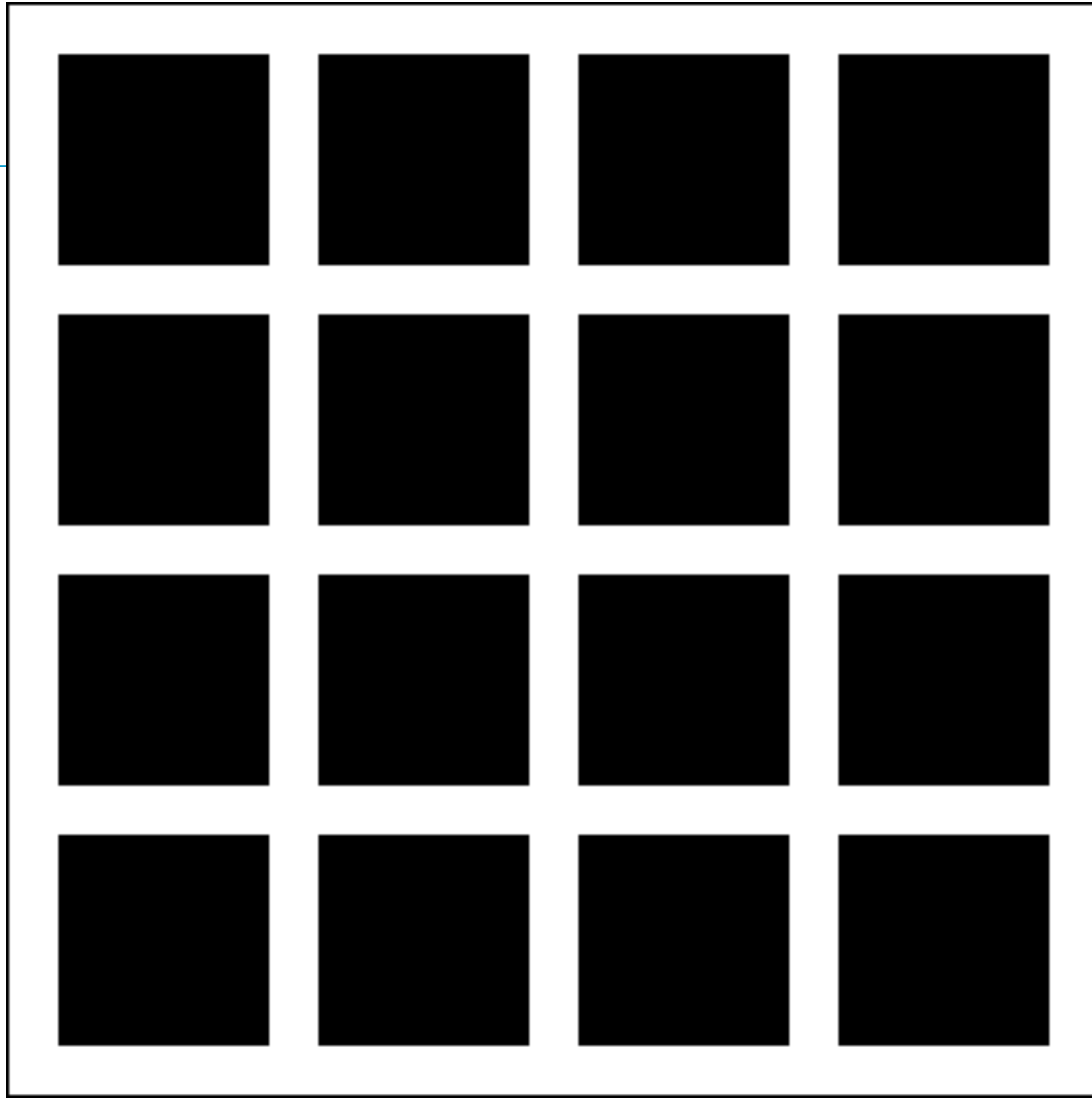
Figure 3 Discrete approximation to Gaussian function with σ=1.4

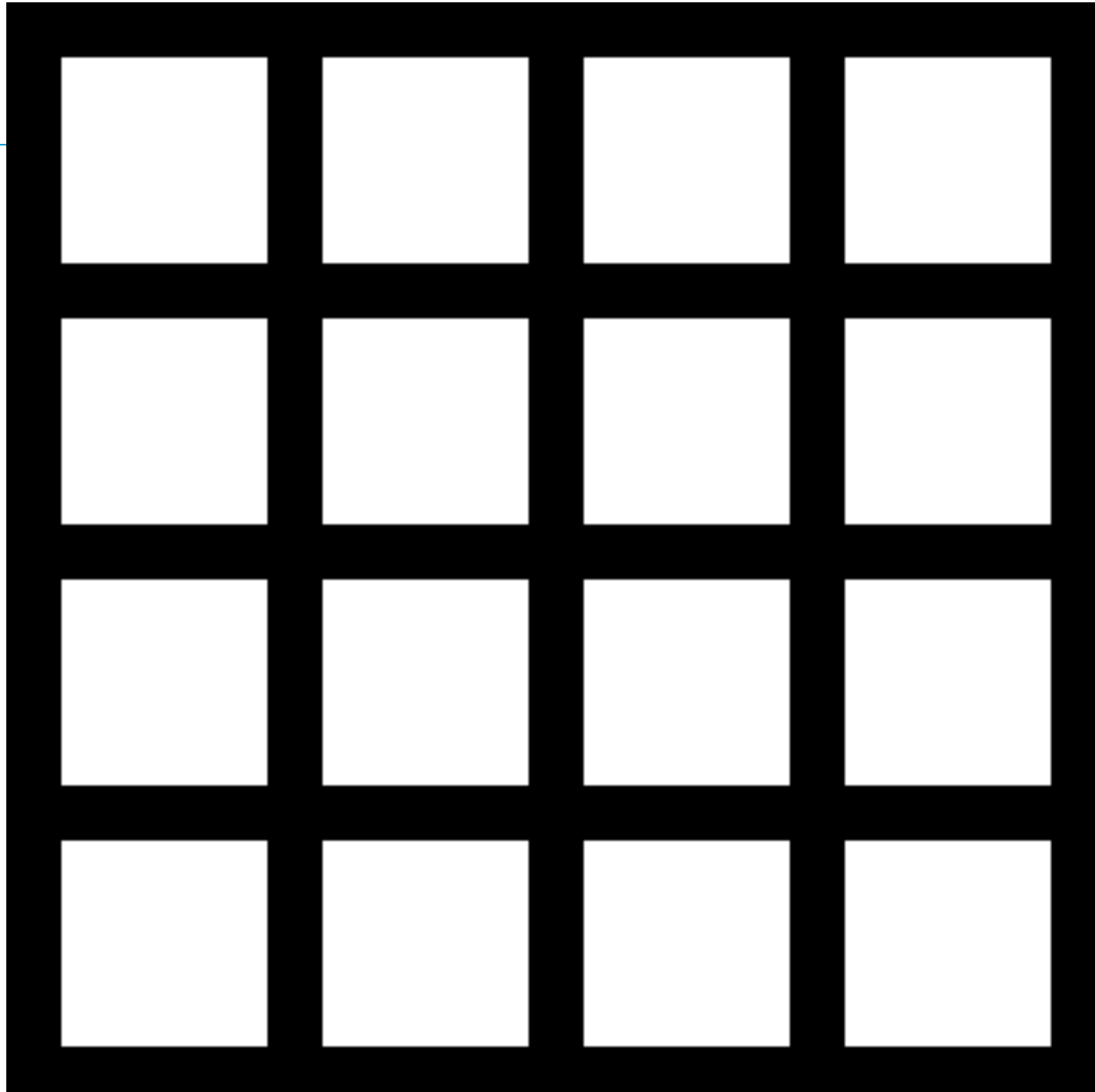
Hands-on exercise 2

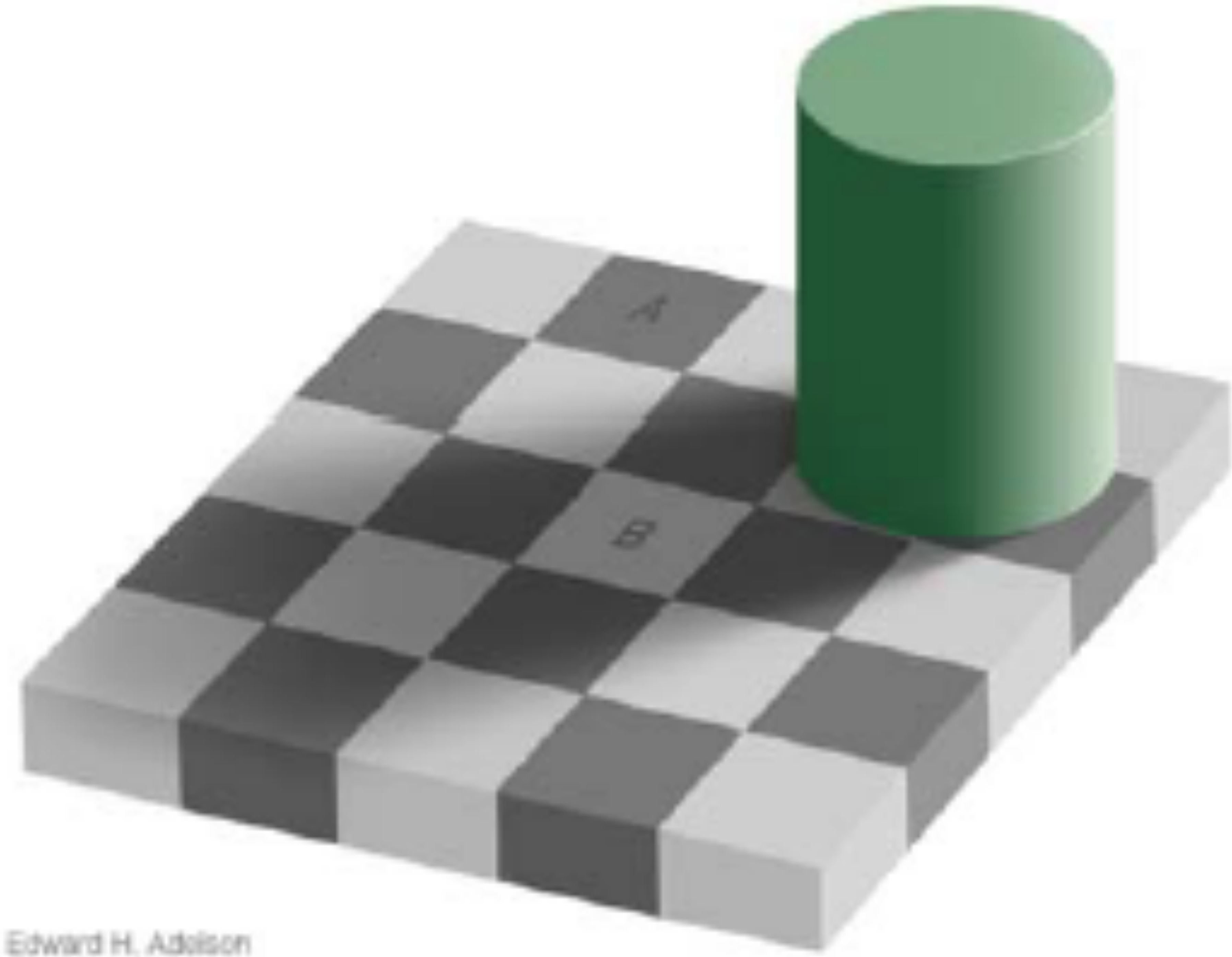
On-center off-surround cell

- a Diffuse light does not activate the cell
- b Light focused on the centre does activate the cell
- b Light focused on the surround generates an “off” response





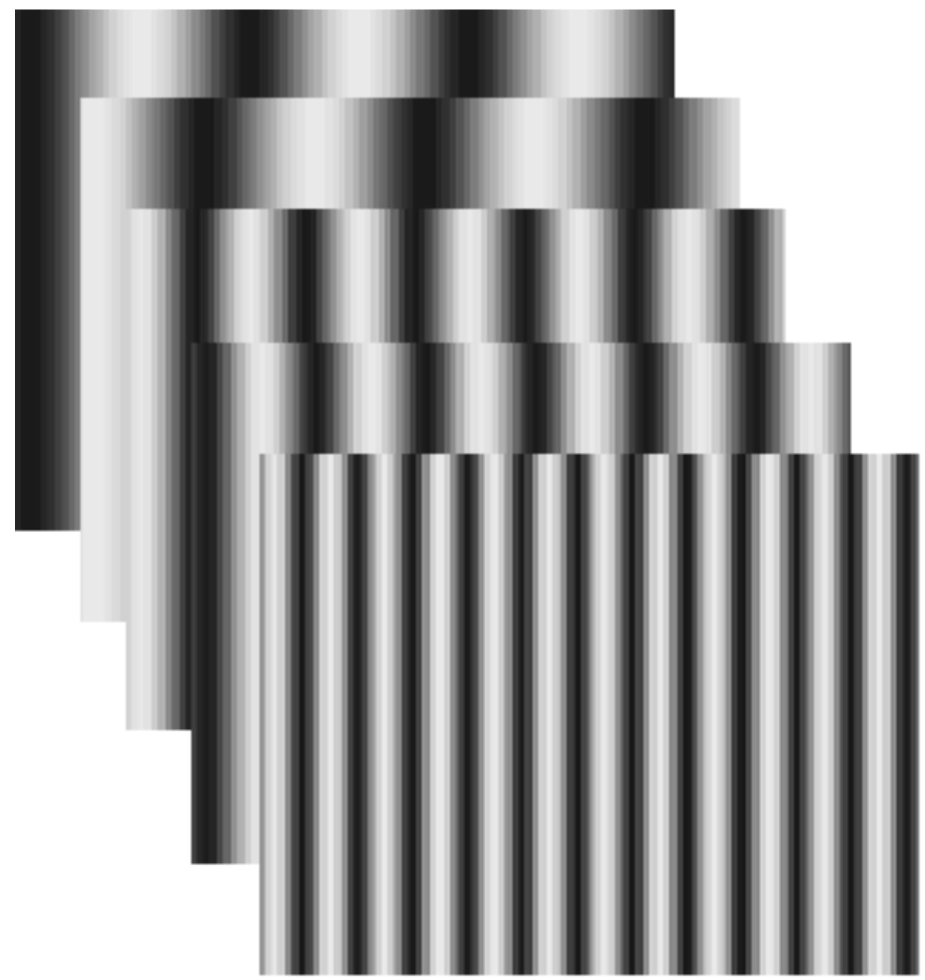



A
B
Edward H. Adelson

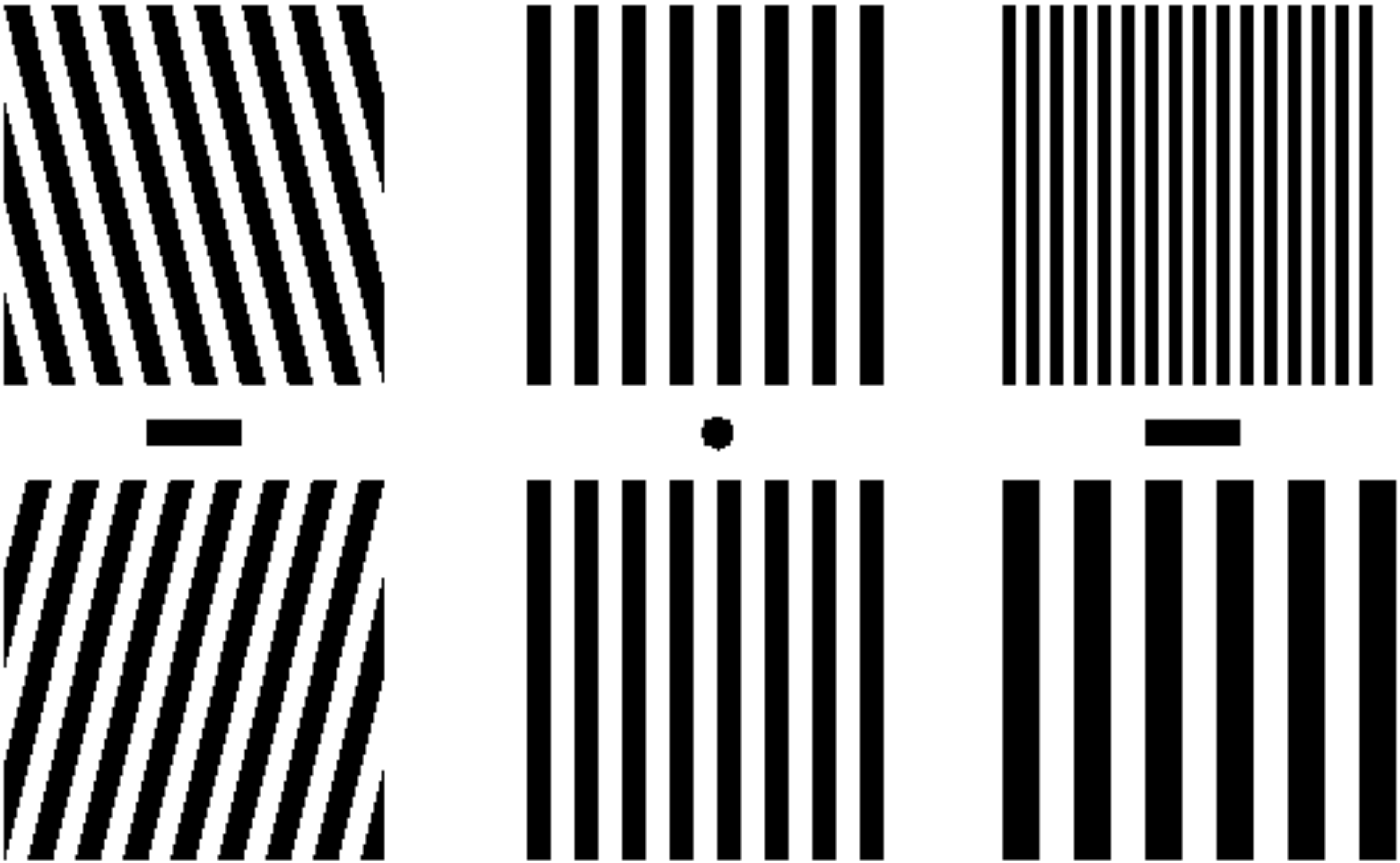
Convolution, Fourier and Gabor

- Fourier transform operates globally on images
  - extracts the global spatial frequencies and orientations of visual images
  - e.g., details of the face + face + other faces + bodies + background
- Gabor transform operates locally on images
  - extracts the local spatial frequencies and orientations of visual images
  - e.g., details of the face
- Convolution
  - comparing a template (or filter) with the local contents of an image at each image position
  - i.e., local comparison/filtering, but global operation
- Convolution can be performed in the spatial domain or in the Fourier domain

Feature extraction: example in vision



TILBURG UNIVERSITY



Jean Baptiste Joseph Fourier

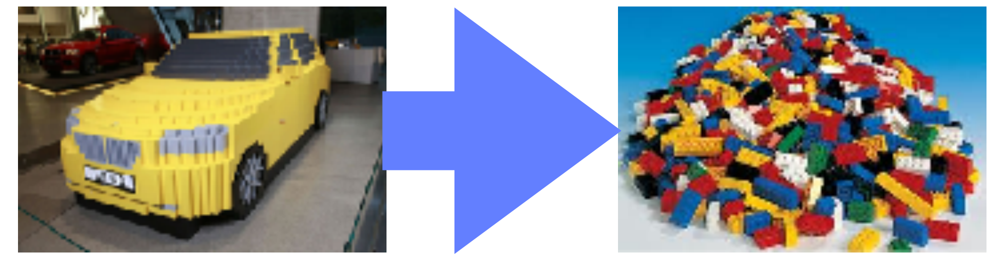
March 21, 1768 - May 16, 1830

Born Auxerre, France.

Died Paris, France.

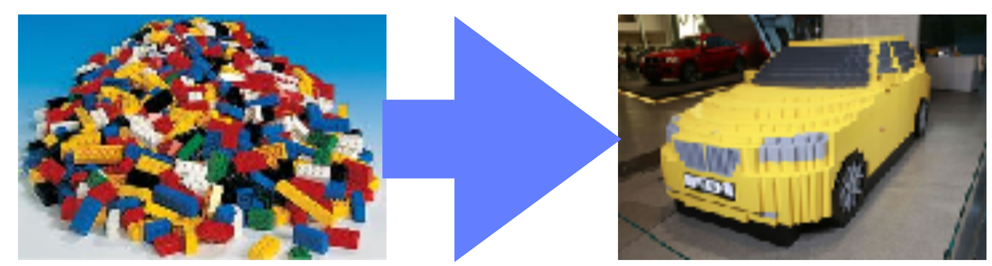
Analysis & synthesis

- FOURIER ANALYSIS



  – The representation of a periodic sound or waveform as a sum of Fourier components
  (i.e. pure sinusoidal waves).

- FOURIER SYNTHESIS



  – The construction of a periodic signal on the basis of Fourier coefficients which give the amplitude and phase of each component sine wave.

Frequency

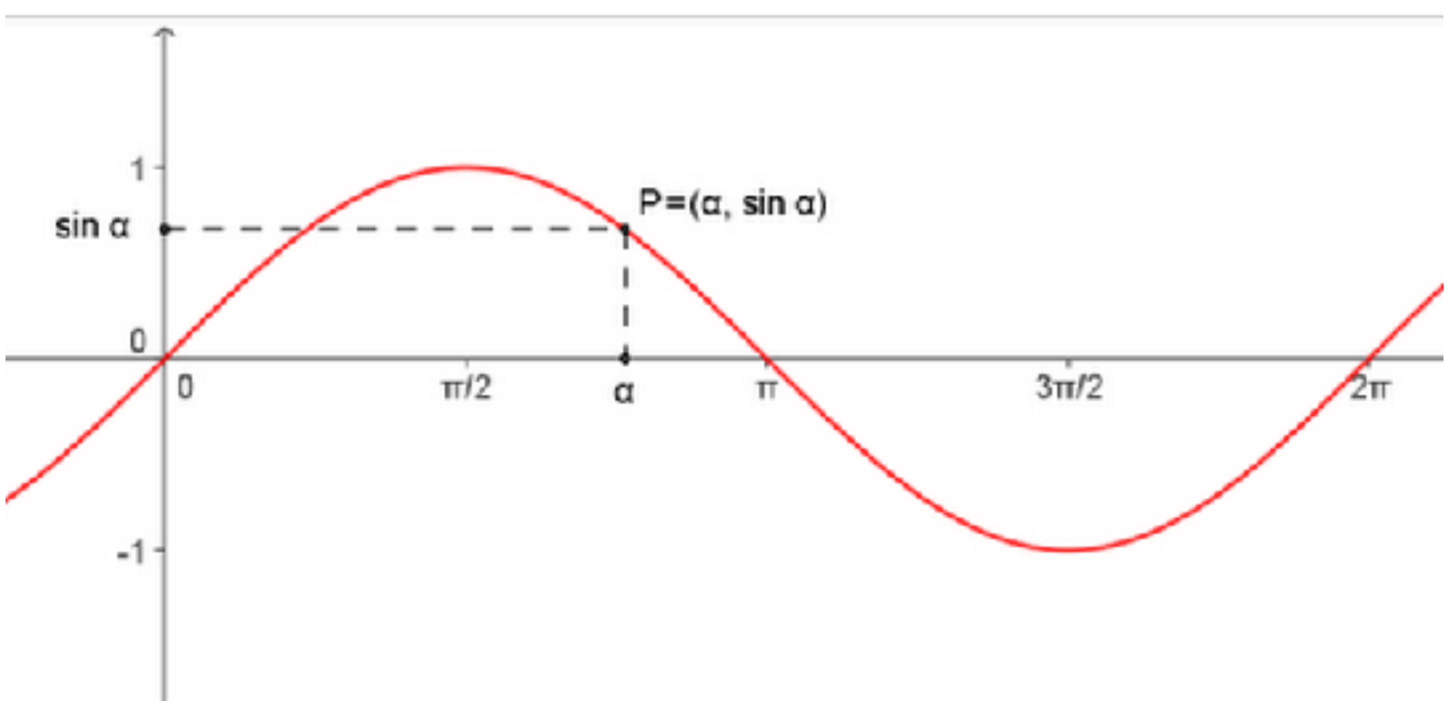
- 1-dimensional (e.g., speech)
  - For instance, auditory signals
  - High frequency: high pitch
  - Low frequency: low pitch

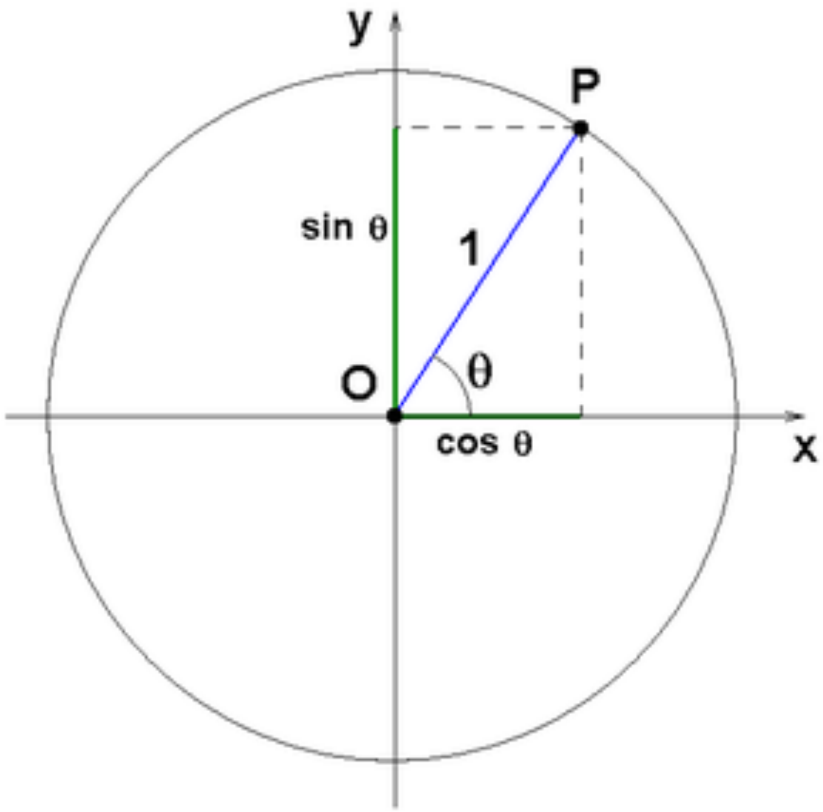
- 2-dimensional (e.g., vision)
  - Visual images
  - High spatial frequency: small objects (details)
  - Low spatial frequency: large objects

TILBURG UNIVERSITY

Basic building blocks

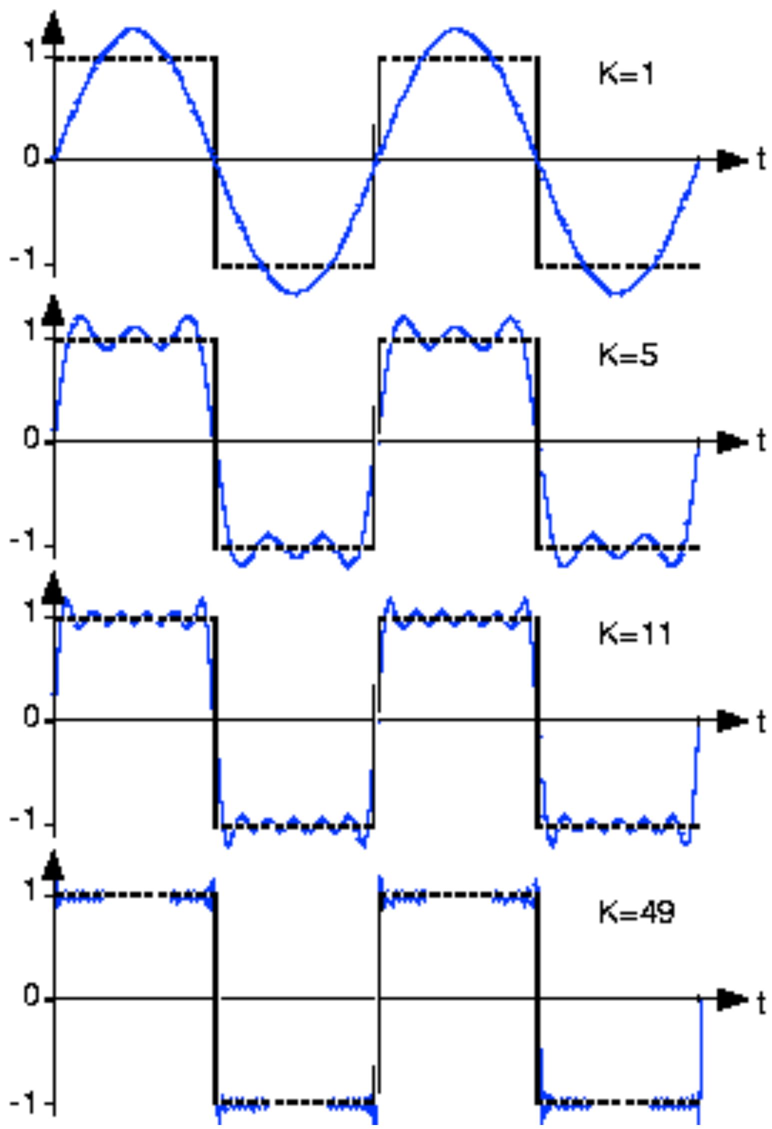
- SIN and COS
- COS is just a shifted SIN

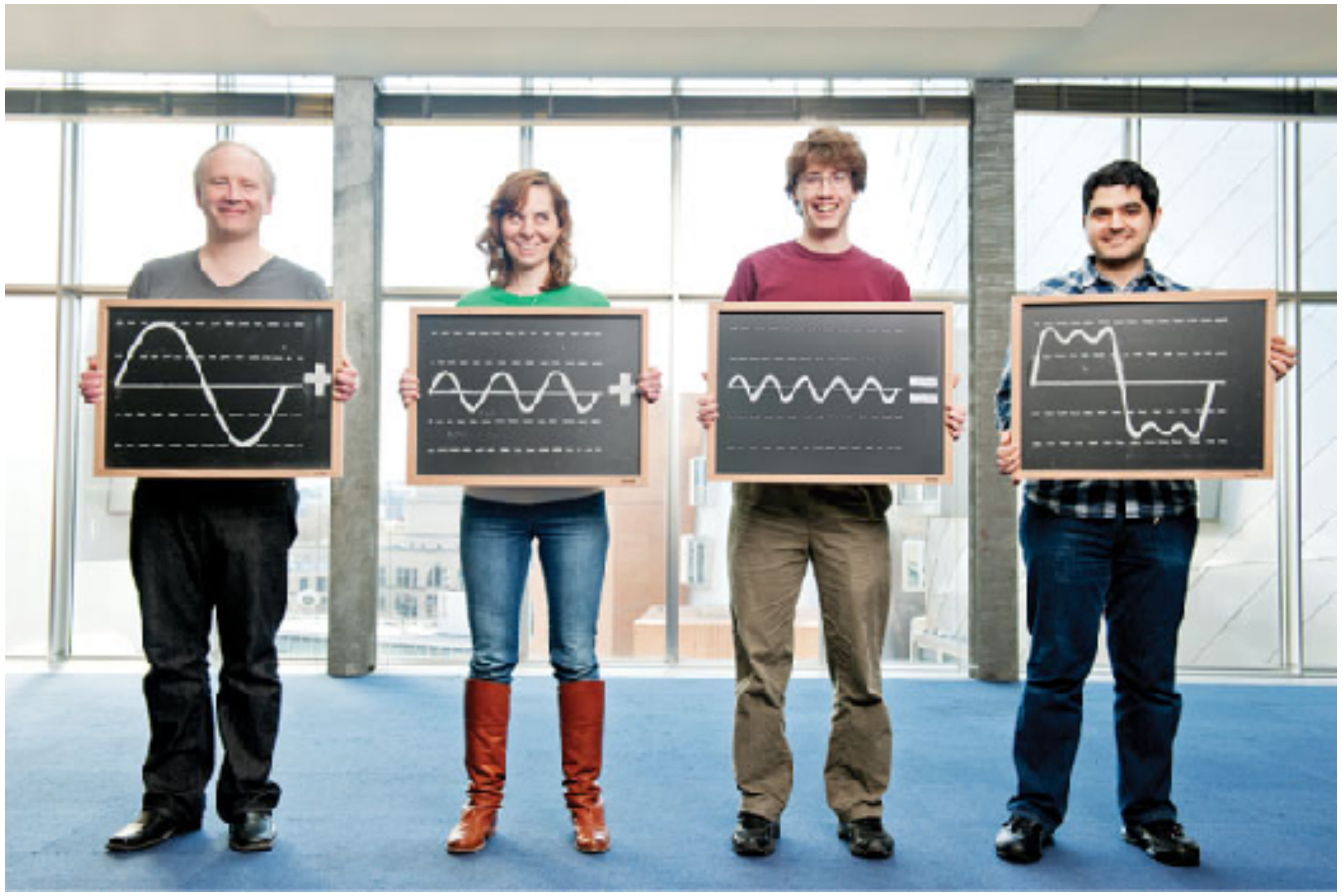


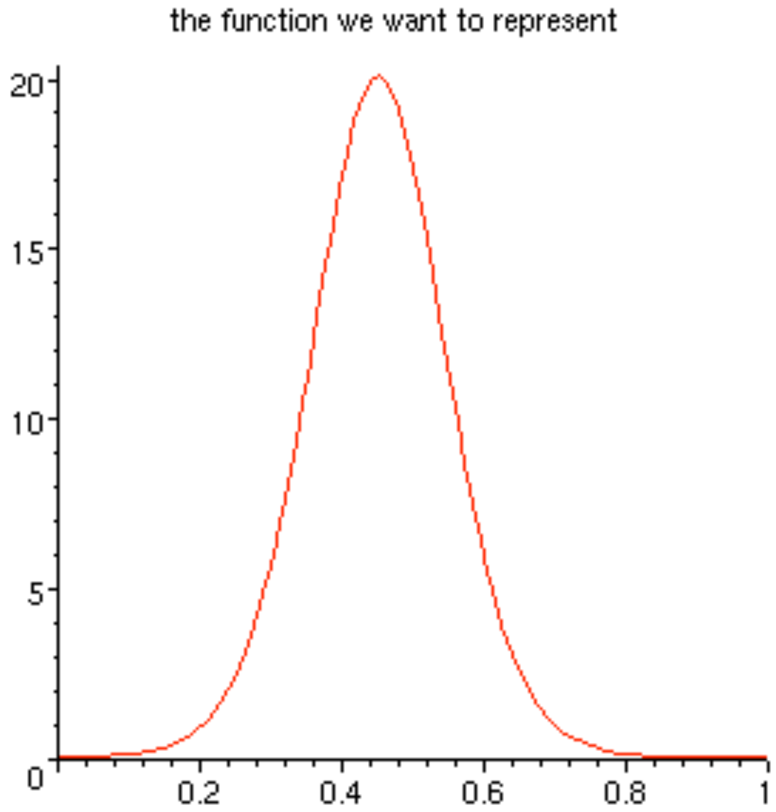


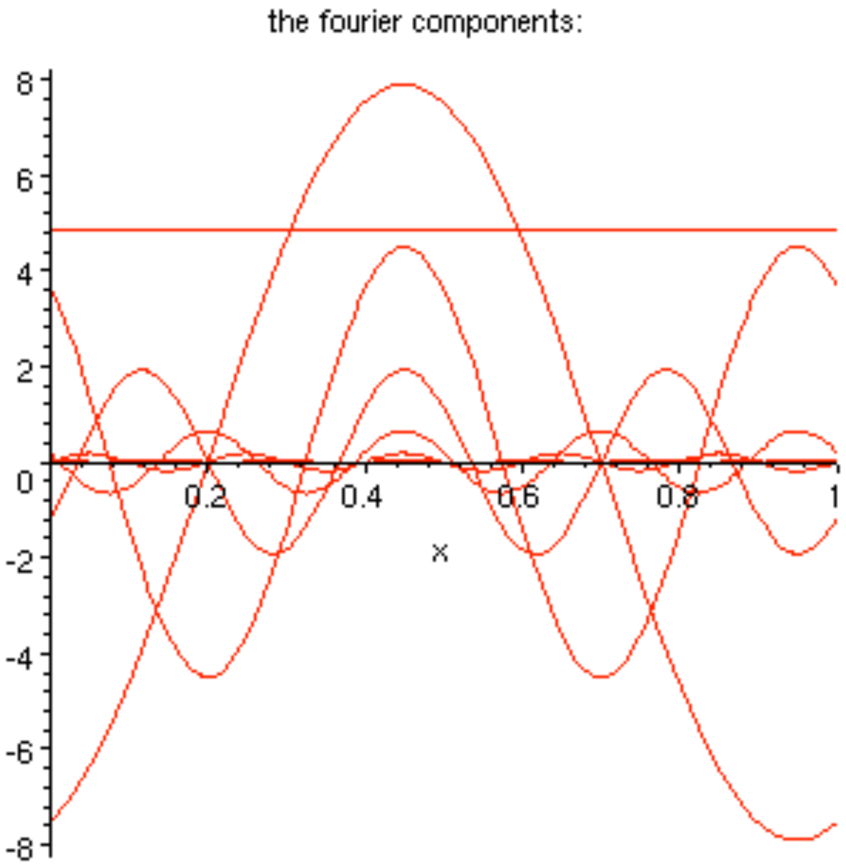
TILBURG ◆ UNIVERSITY

Approximating a square wave by adding many sine waves

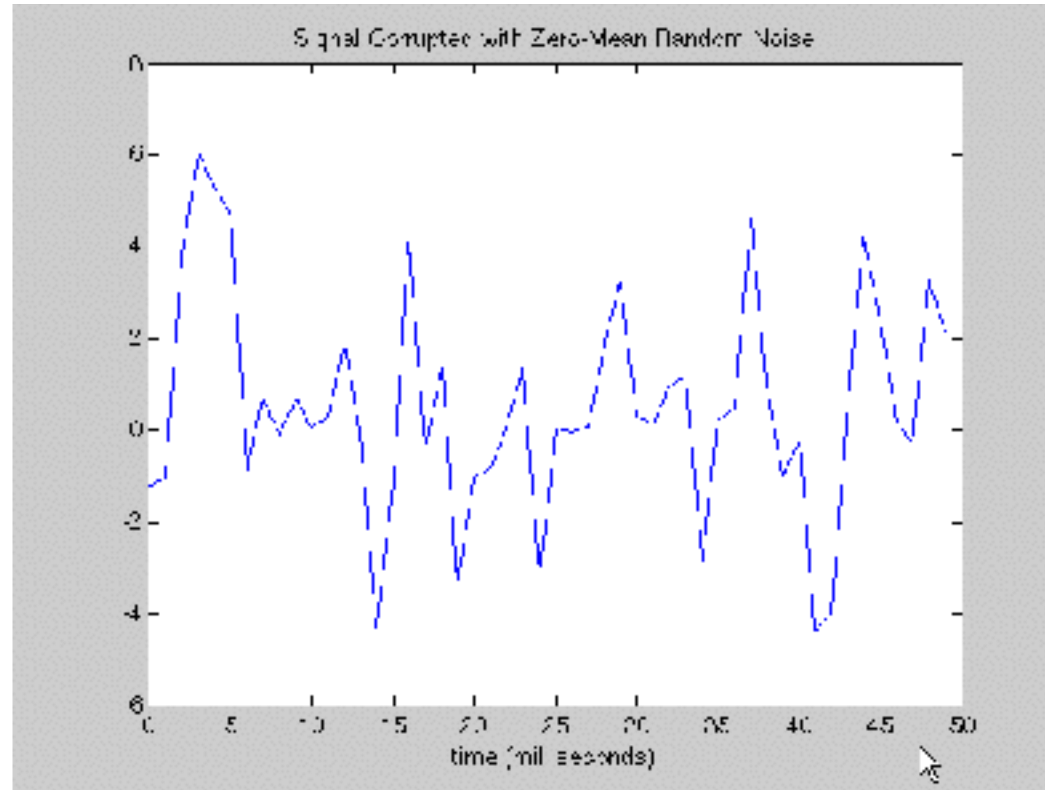




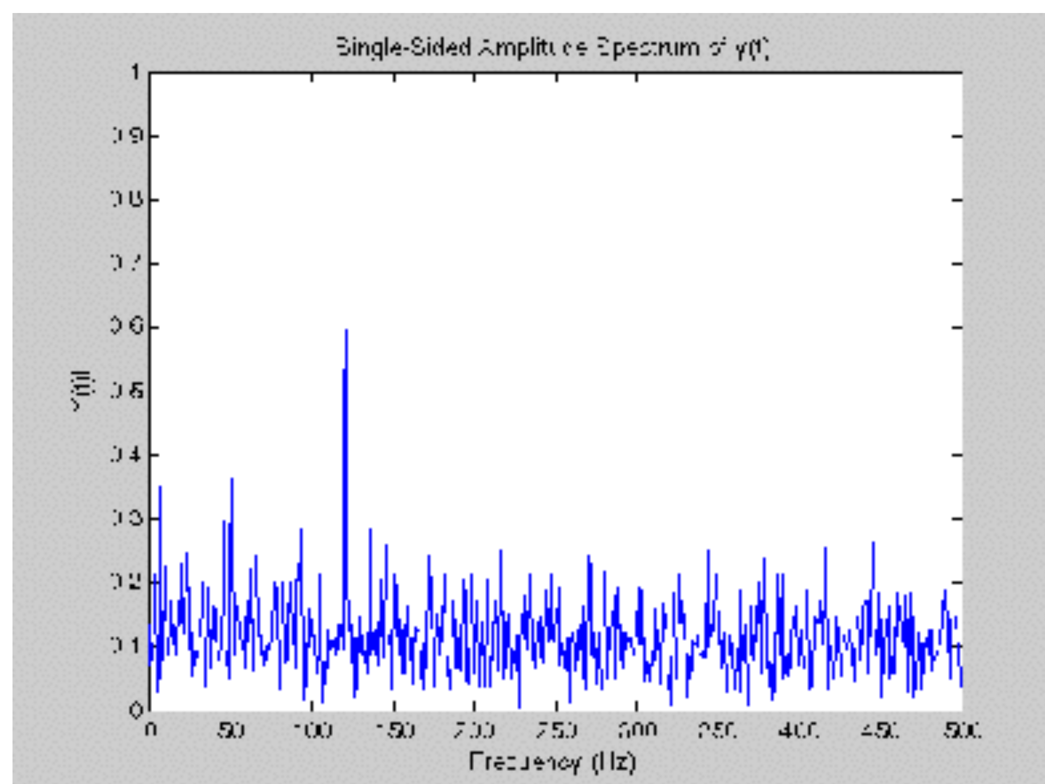

the function we want to represent
20
15
10
5
0
0.2
0.4
0.6
0.8
1


the fourier components:
8
6
4
2
0
-2
-4
-6
-8
0.2
0.4
0.6
0.8
1
x

- temporal signal
  - E.g. a sound

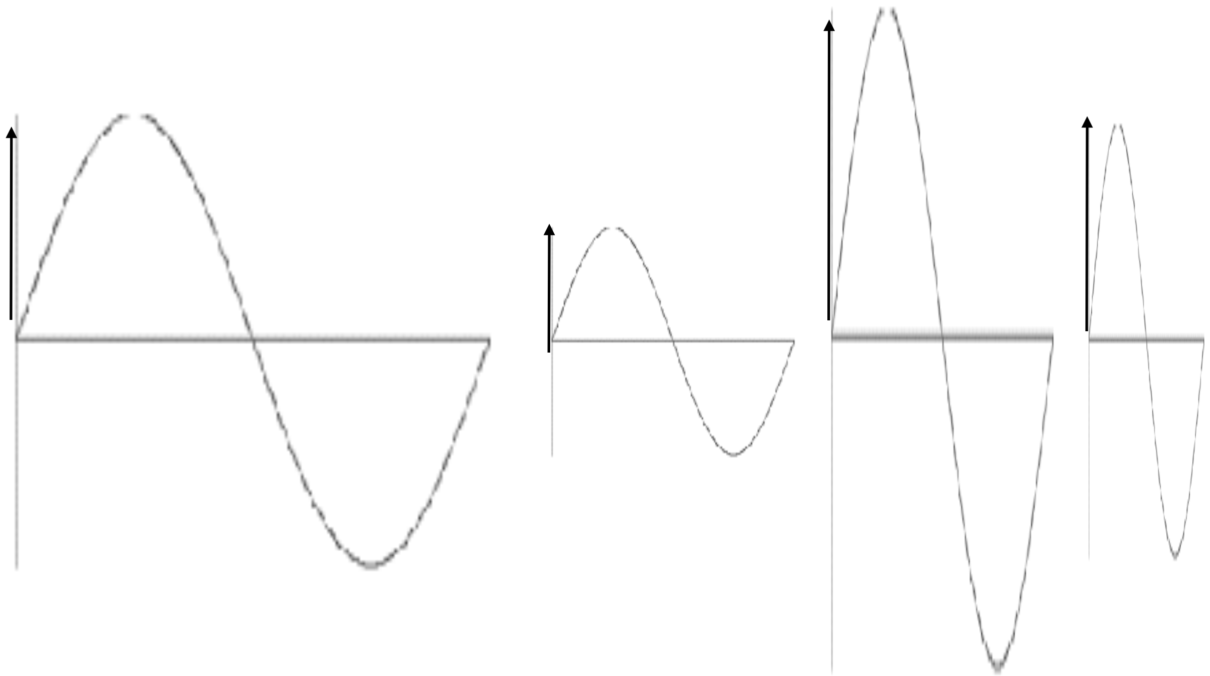


- spectral signal
  - Showing the strength of the constituent frequencies

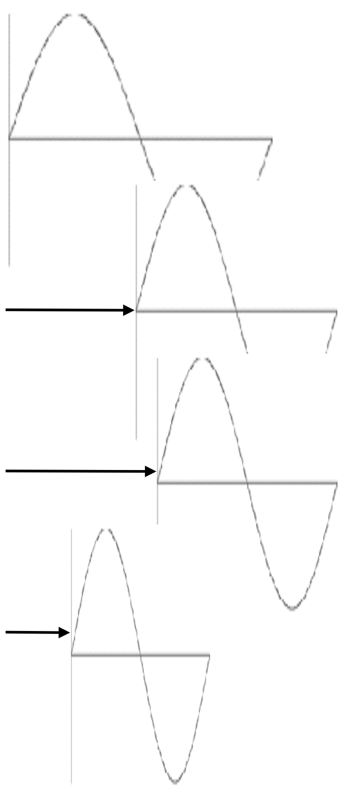


Tilburg University

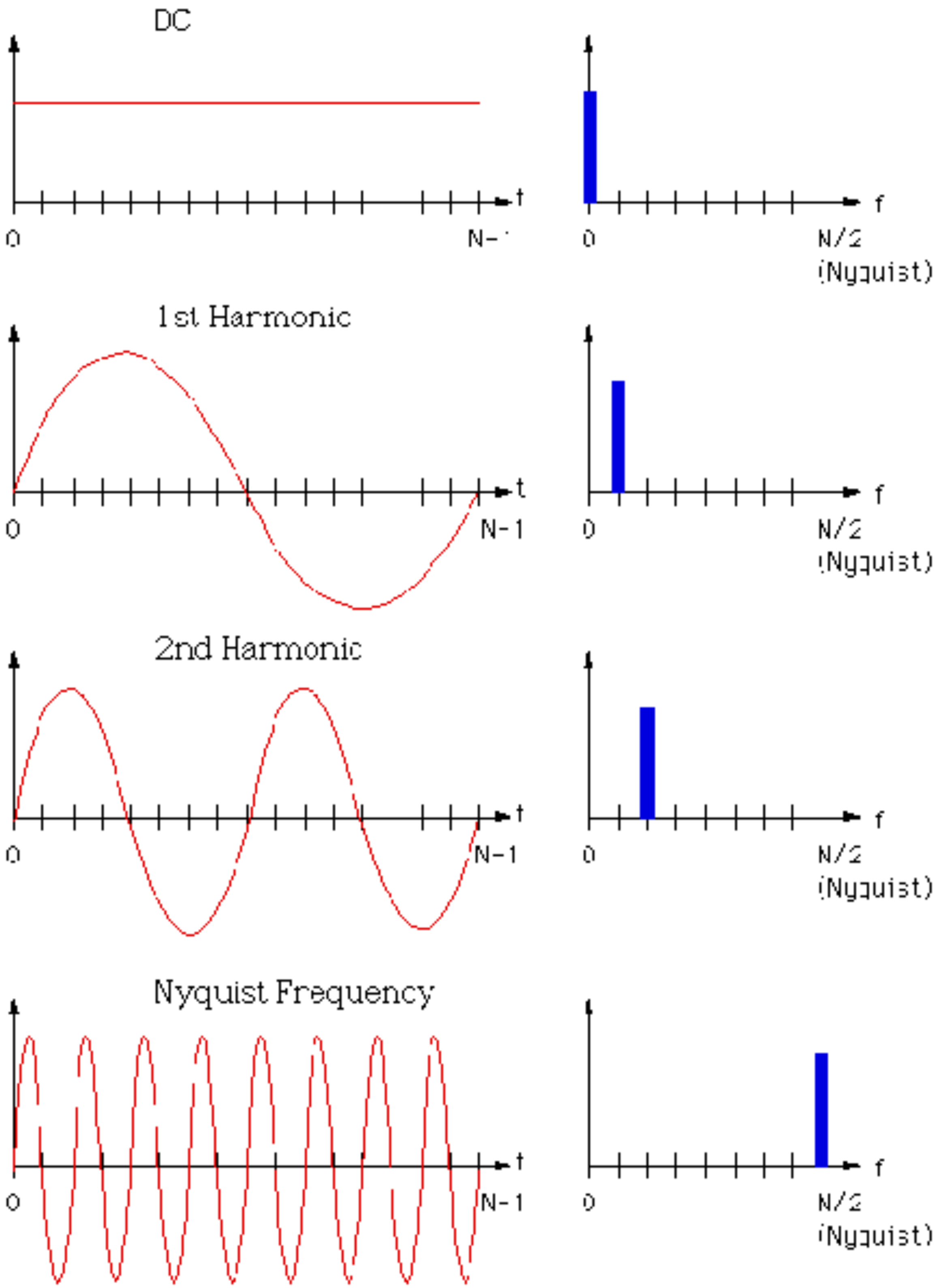
Magnitude and phase



- Magnitude
  - The strength of the individual frequencies (vertical enlargements)

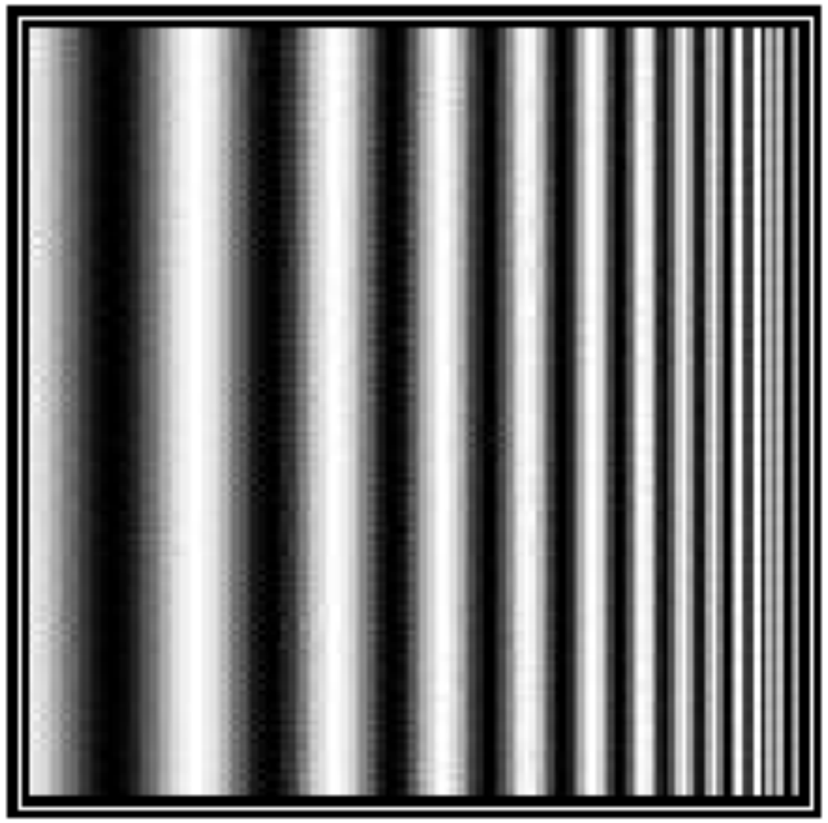


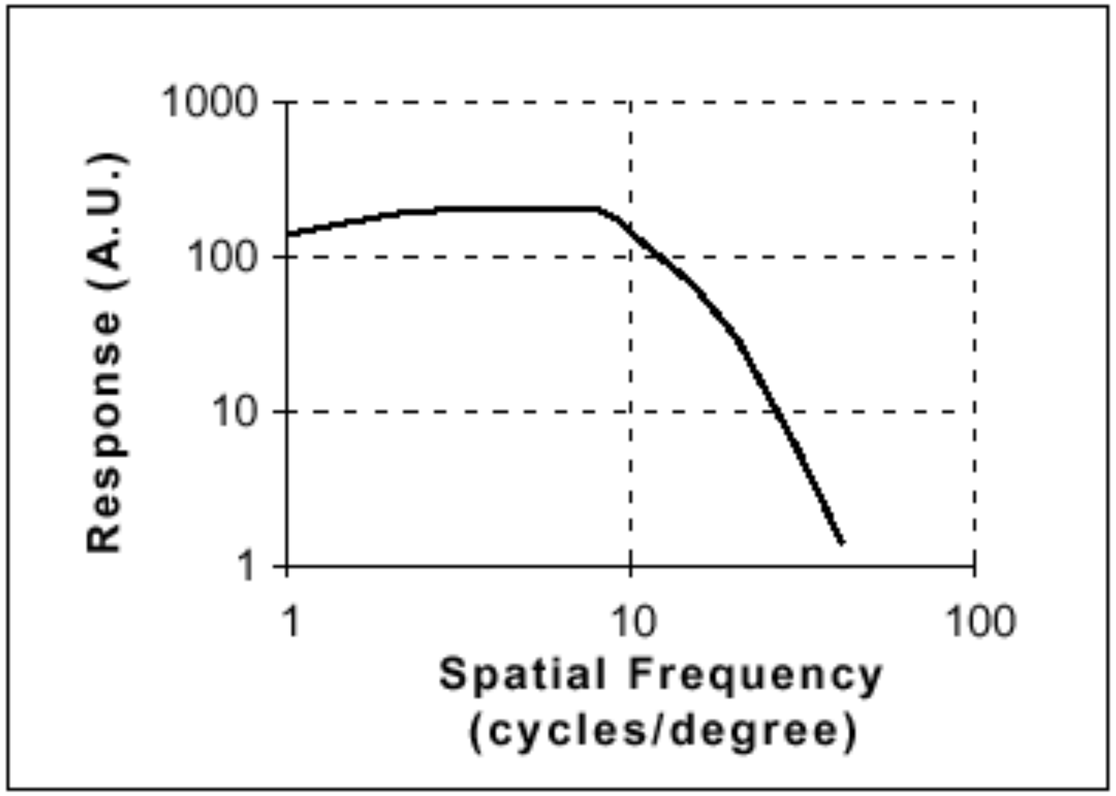
- Phase
  - The position of the individual frequencies (horizontal shifts)



- The smallest frequency that can be represented
- The largest frequency that can be represented

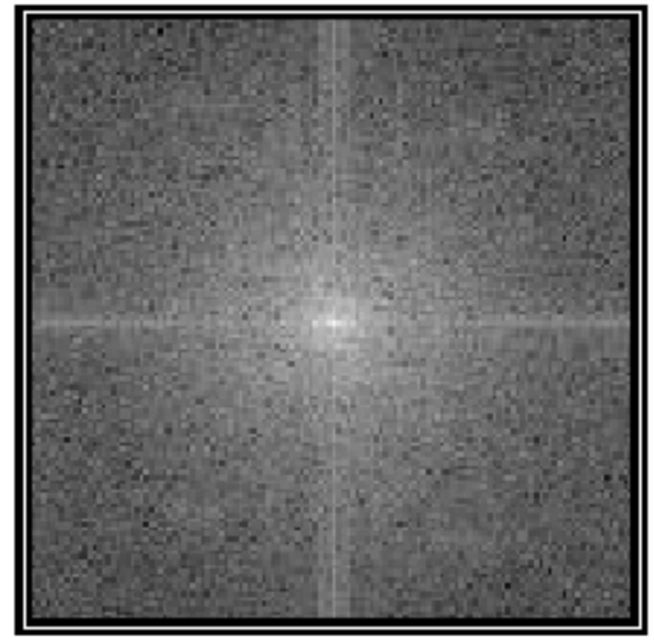
Spatial frequencies



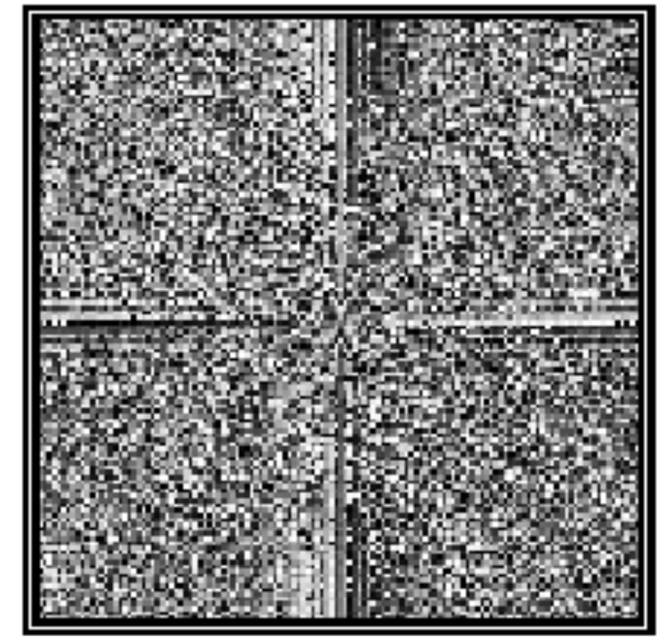


Fourier

image

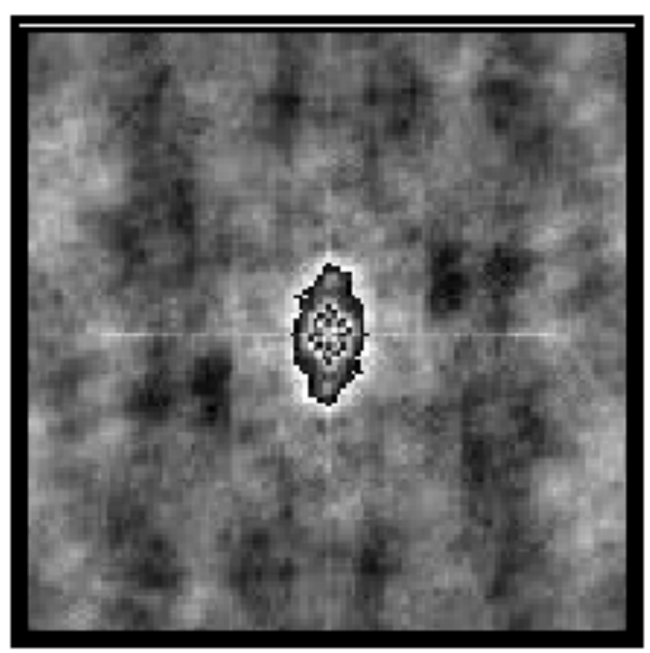


magnitude



phase

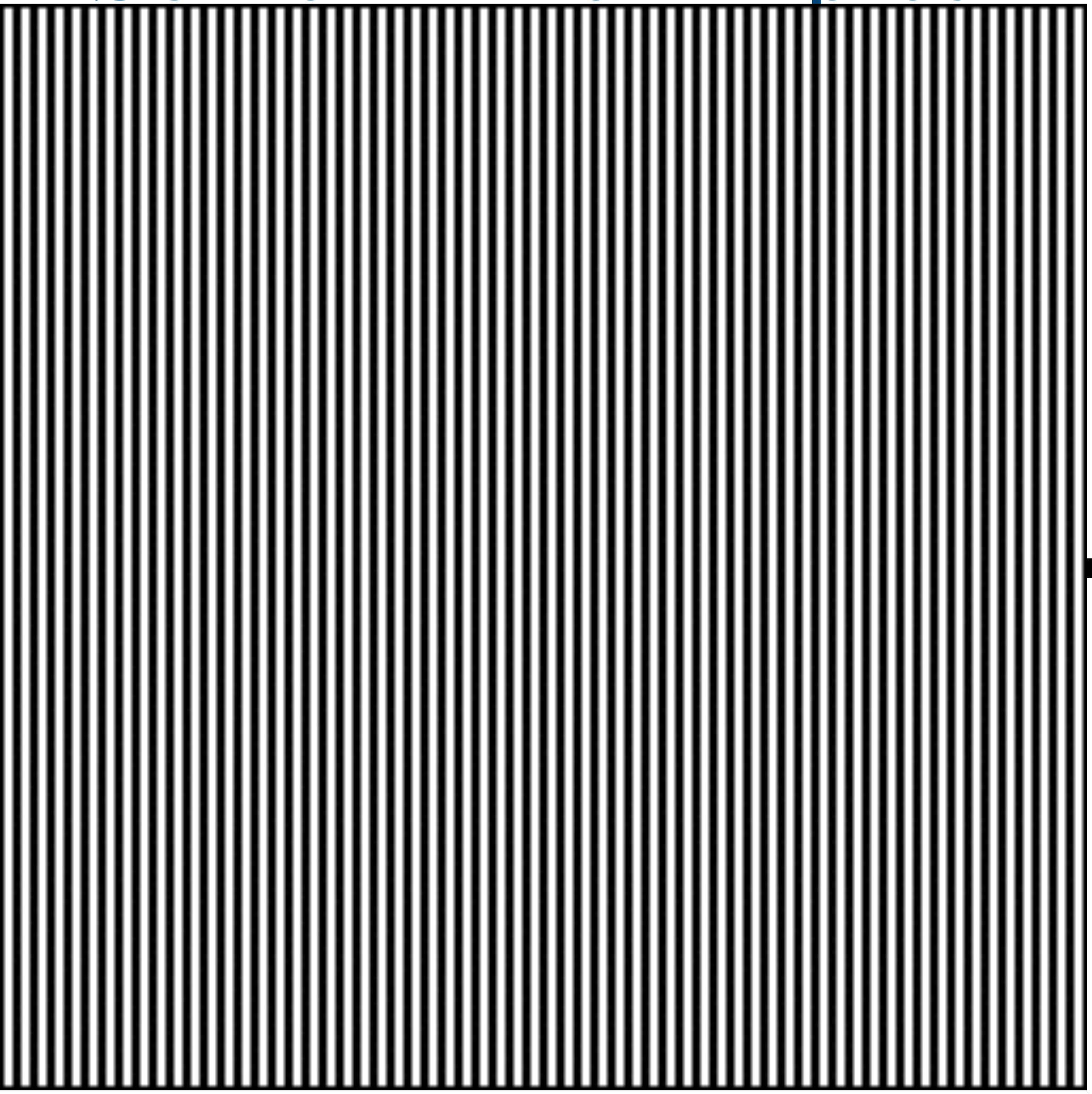
Inverse Fourier



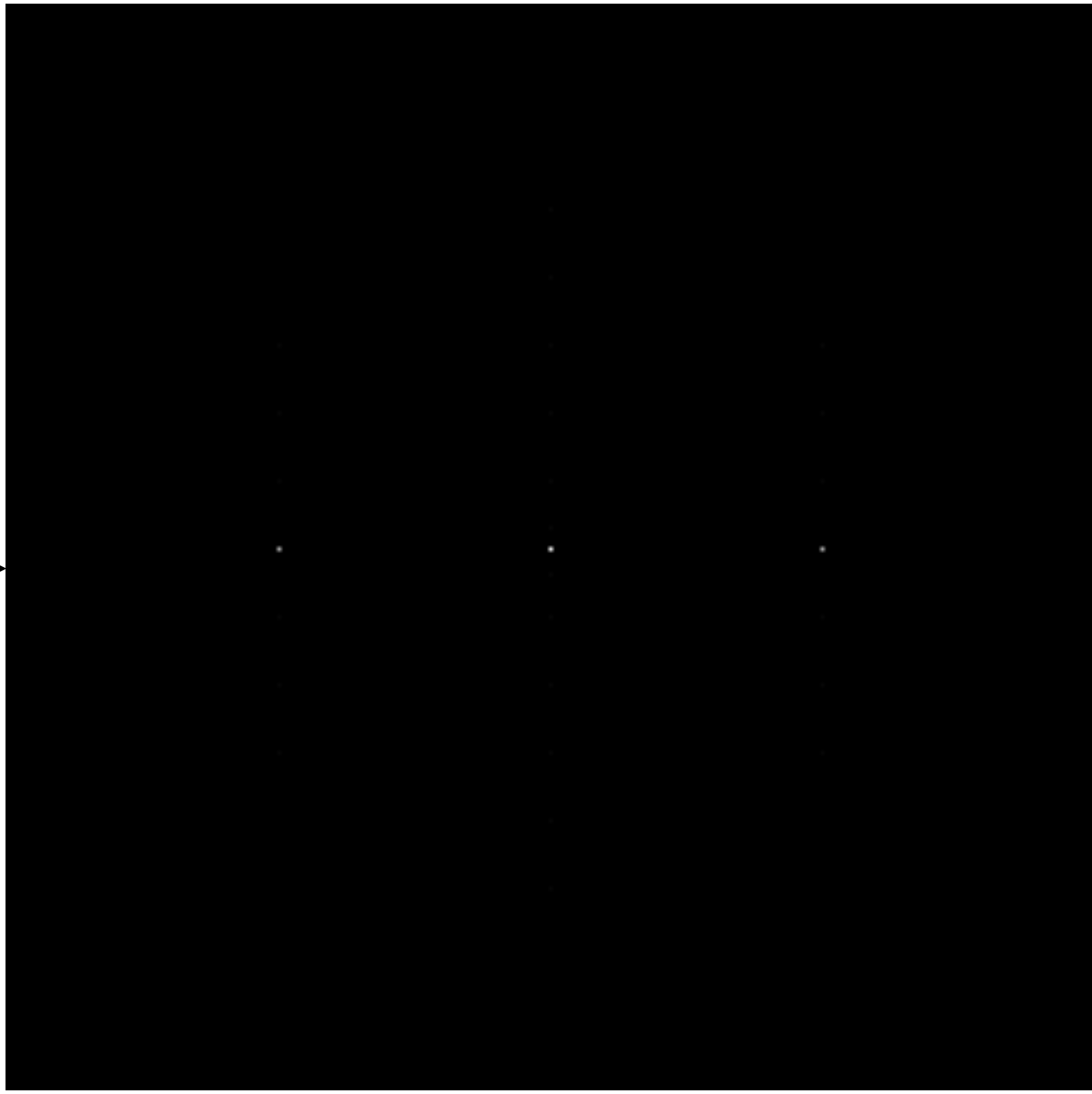
magnitude only

phase only

Some FFT examples

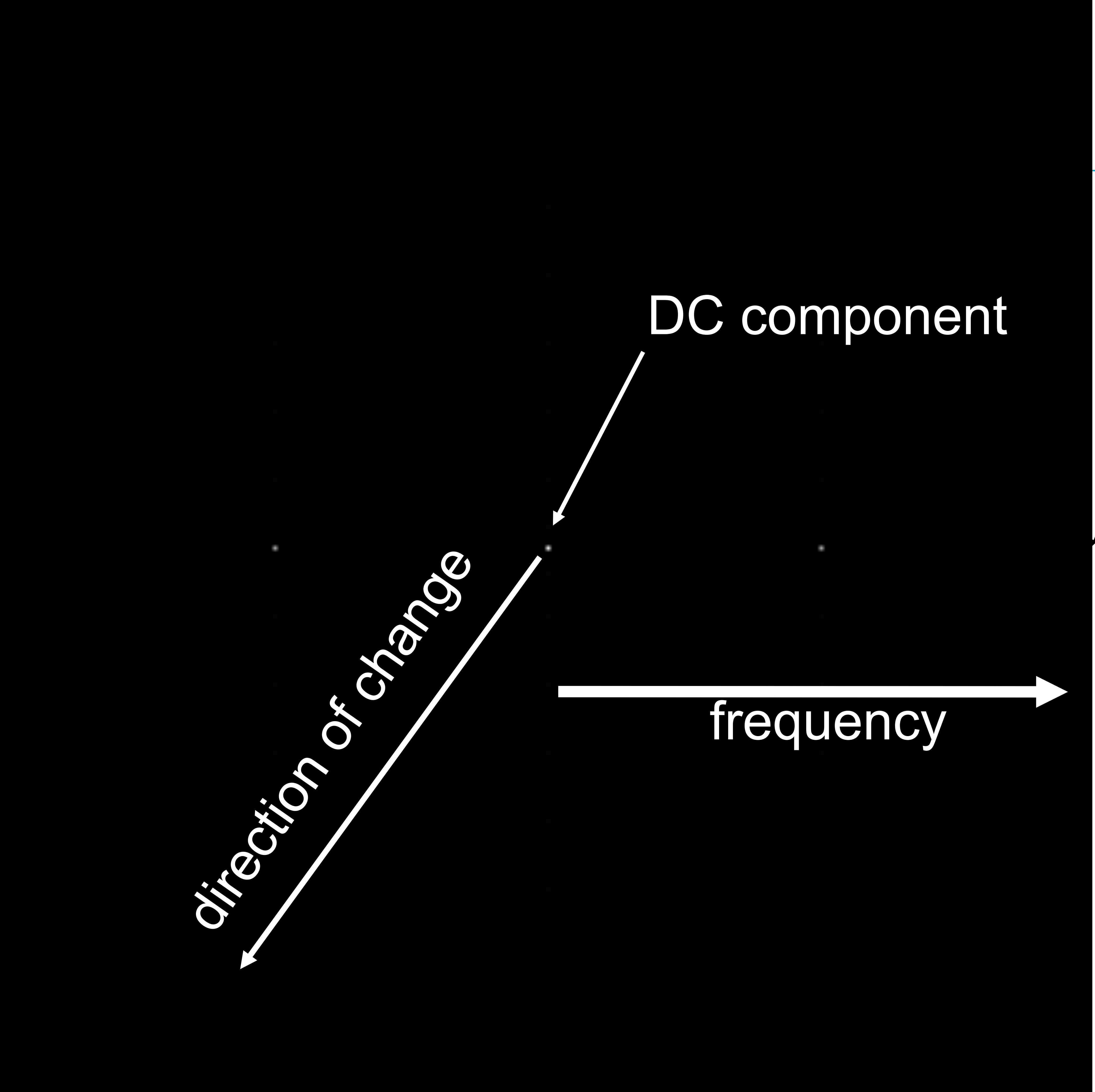


FFT



Individual lines are 2 pixels wide

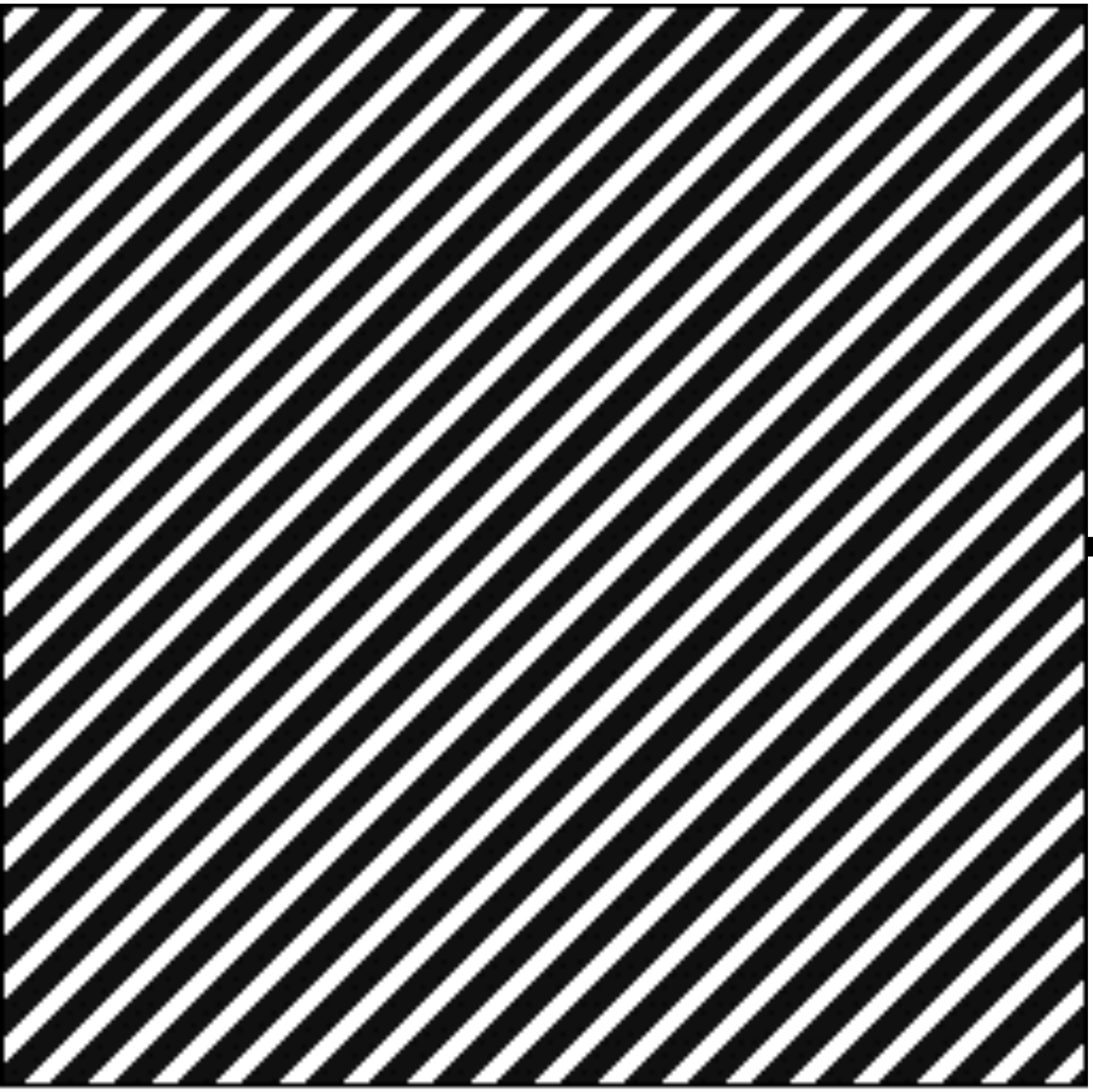
magnitude

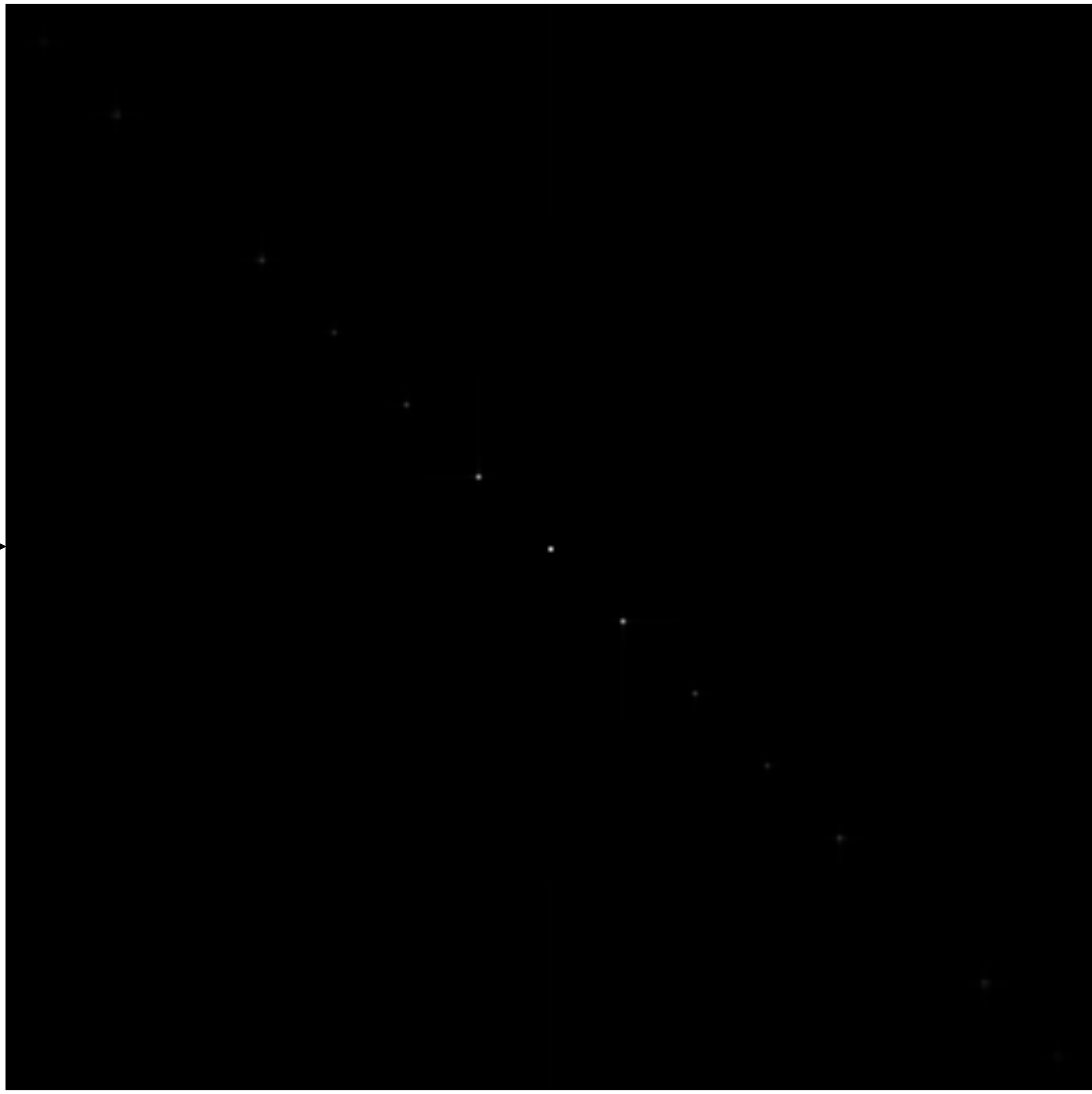


Max frequency

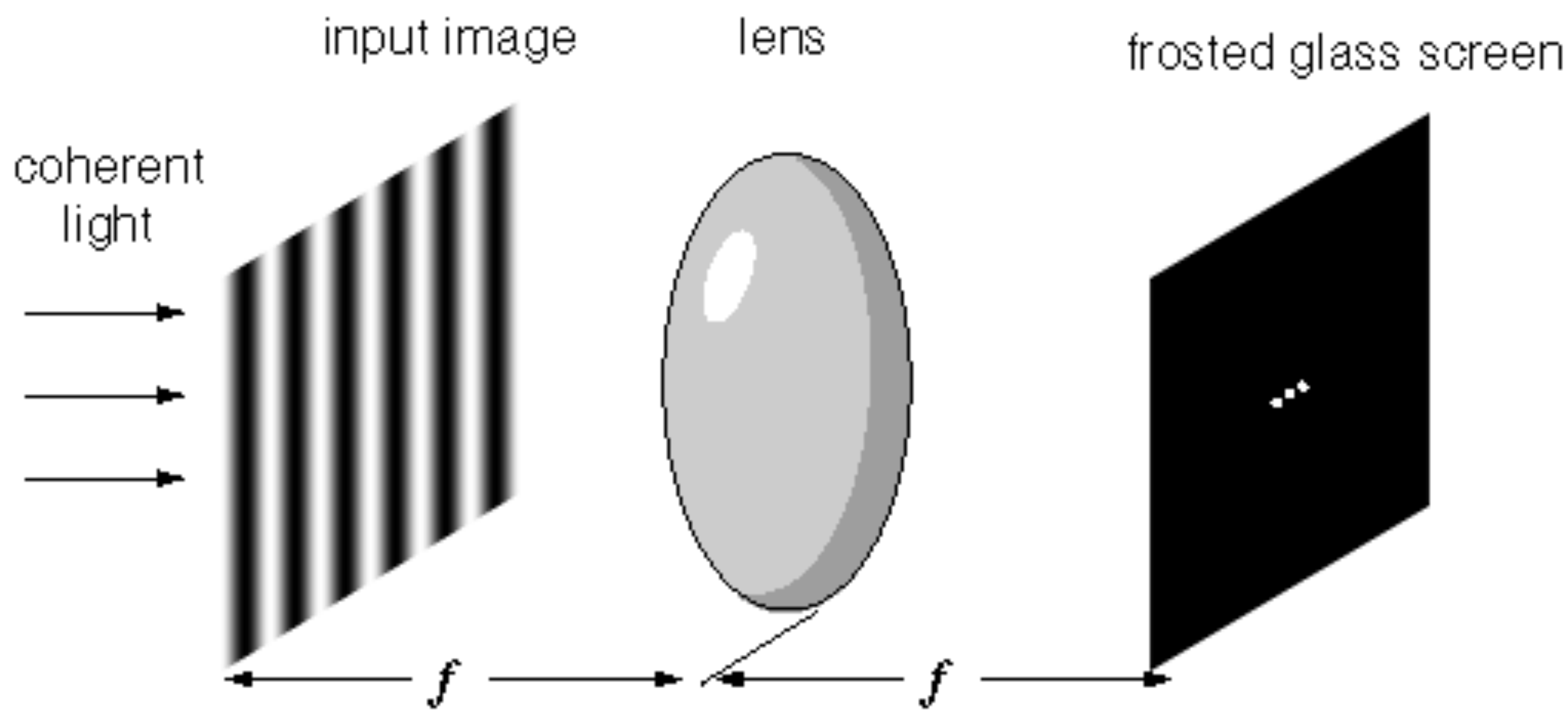
1-pixel wide line

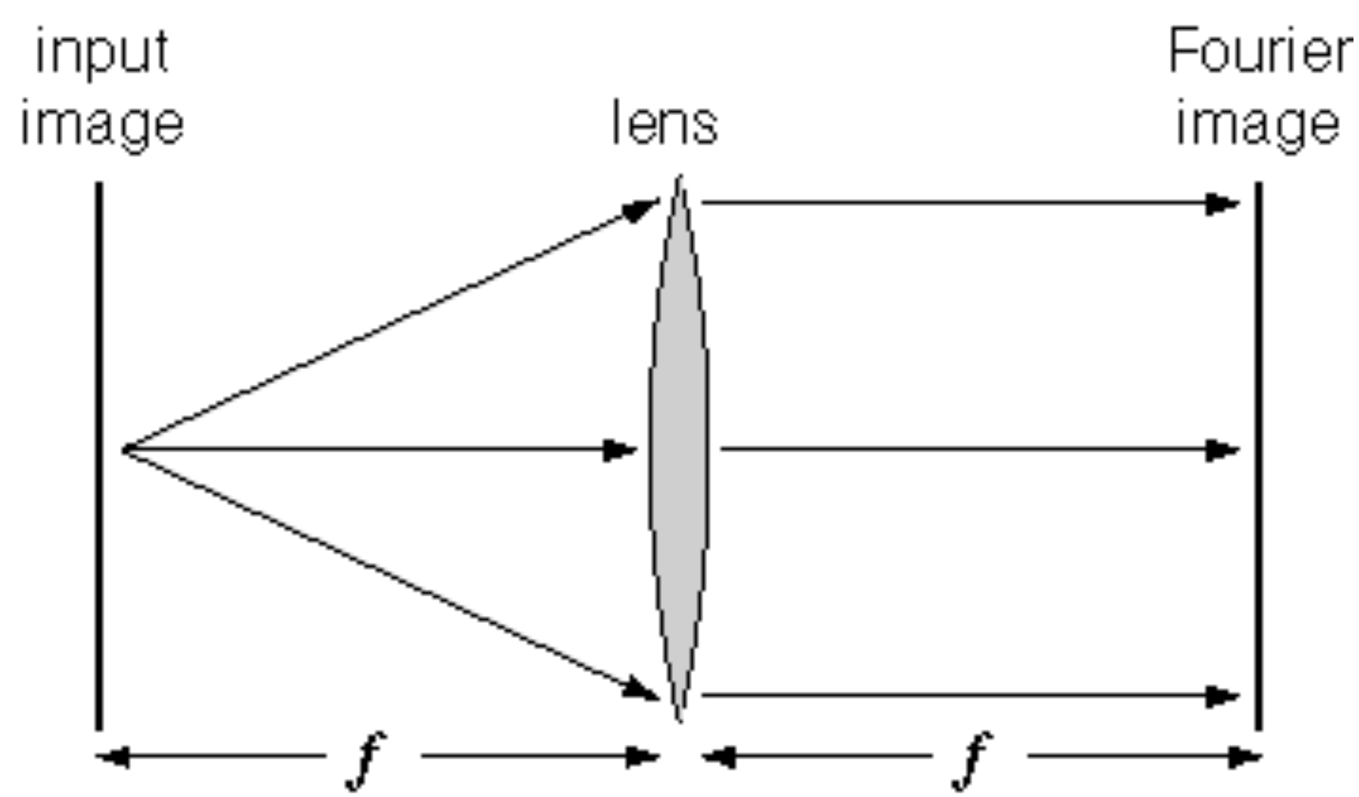
Some FFT examples





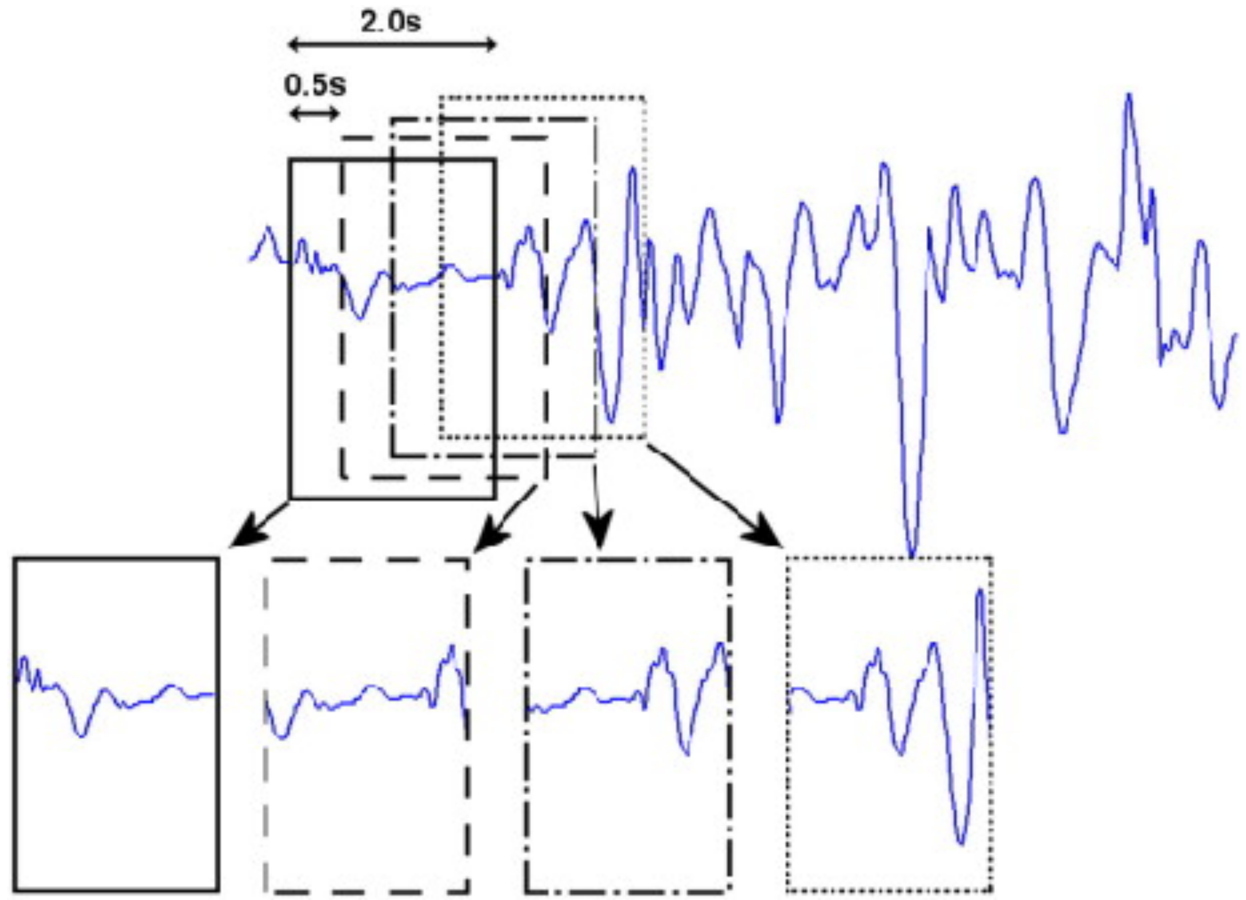
magnitude

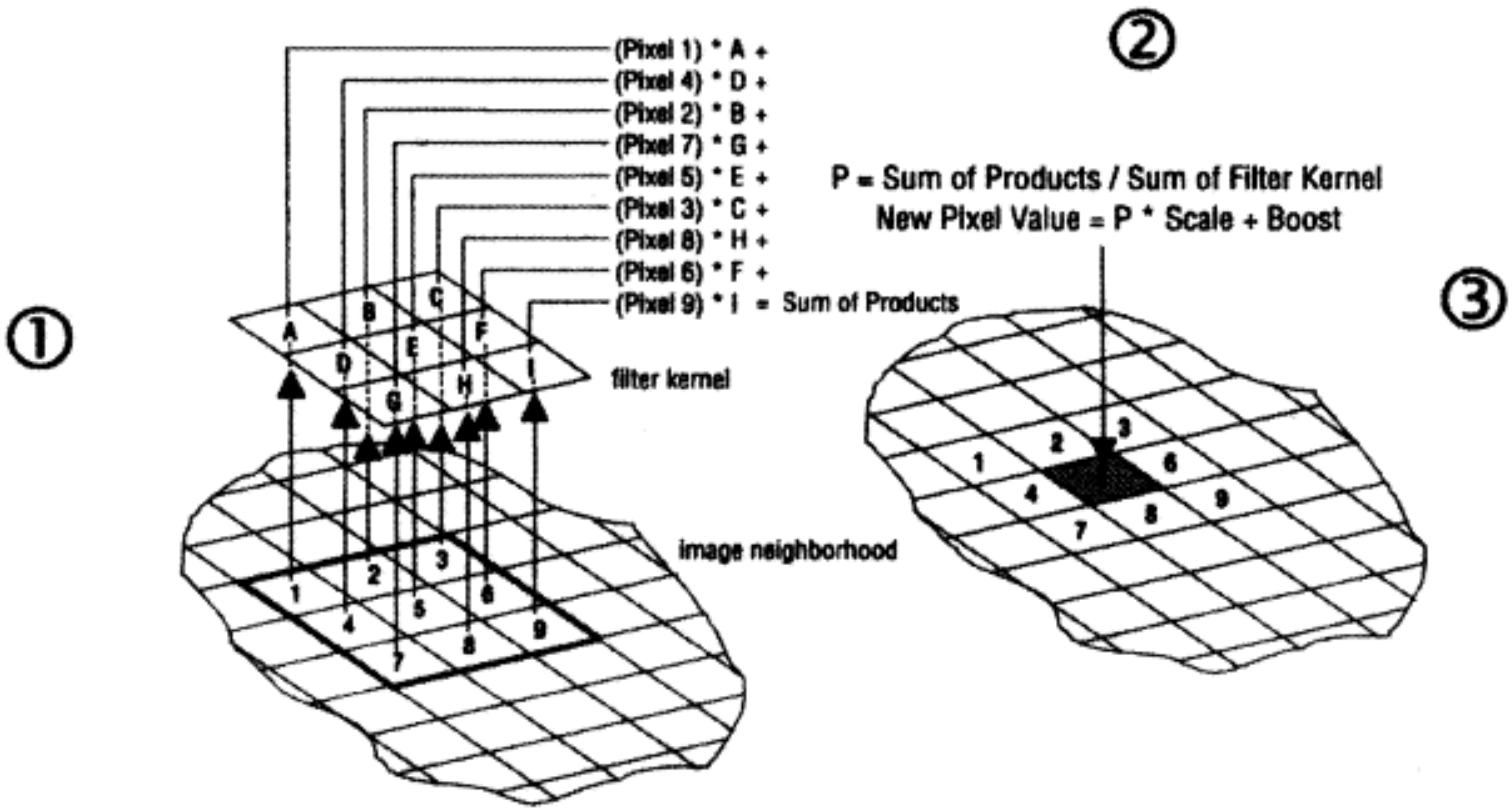

input image
lens
frosted glass screen
coherent
light
f
f


input
image
lens
Fourier
image
f
f

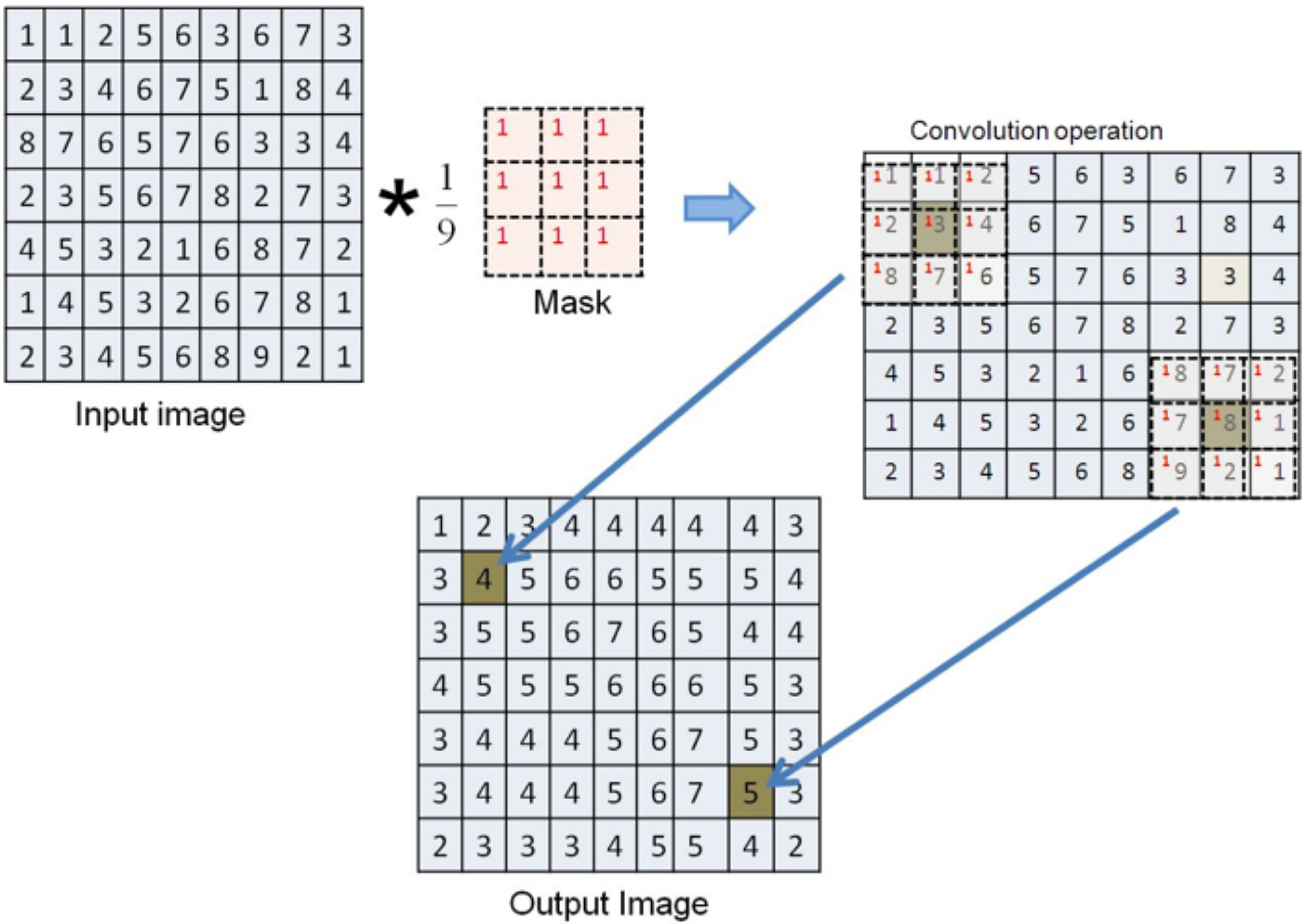
Filtering in the temporal or spatial (image) domain

- Filtering can be done directly in the image domain by applying a filter (matrix of weights) to regions of the image
- This is the so-called “sliding window” method





The Convolution Filtering Process


Input image
1/9
Mask
Convolution operation
Output Image

Alternative way of filtering: spectral filtering

- Image → convolve with filter/receptive field

→ filtered image

-1	0	+1
-2	0	+2
-1	0	+1

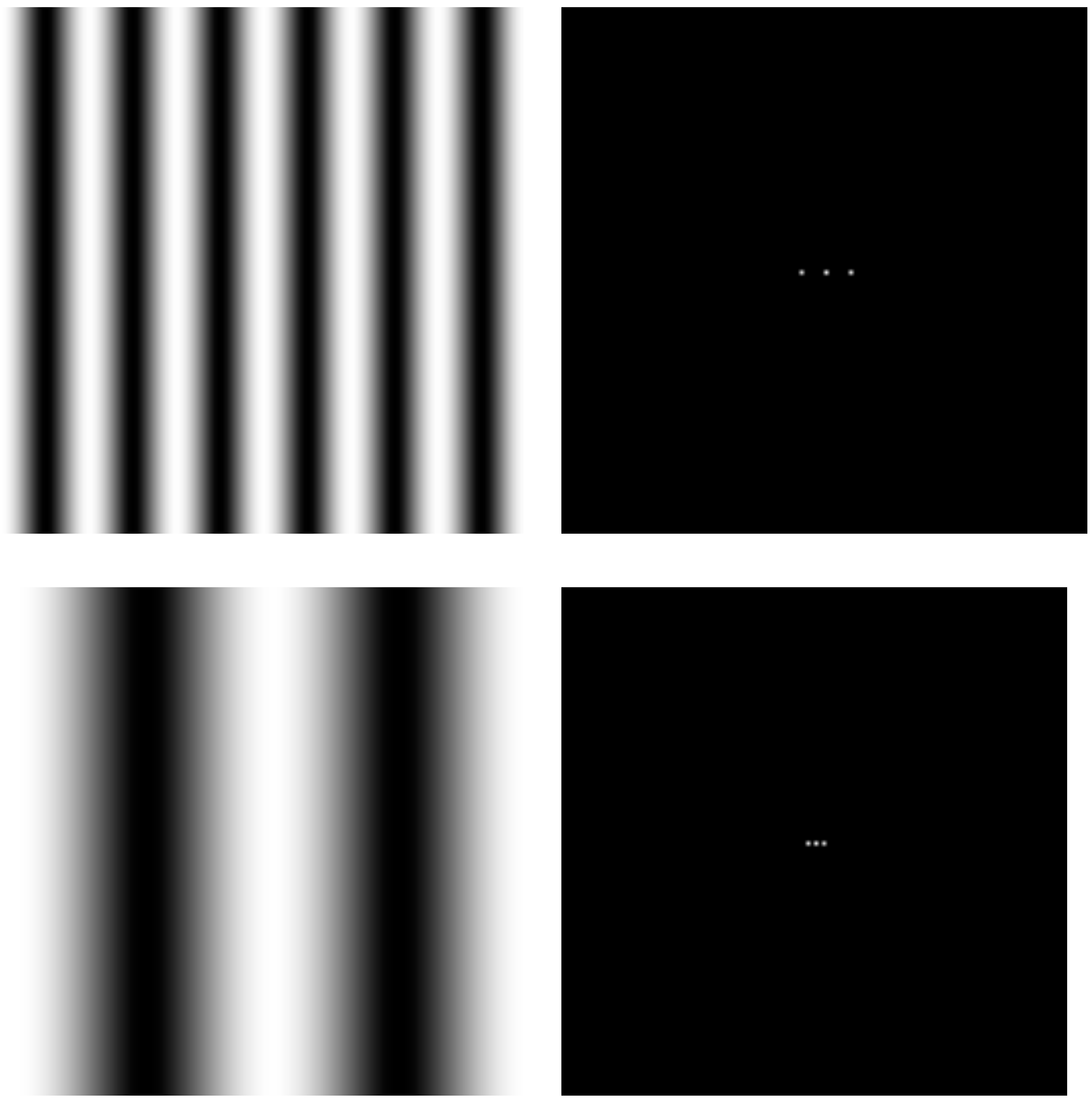
Gx

+1	+2	+1
0	0	0
-1	-2	-1

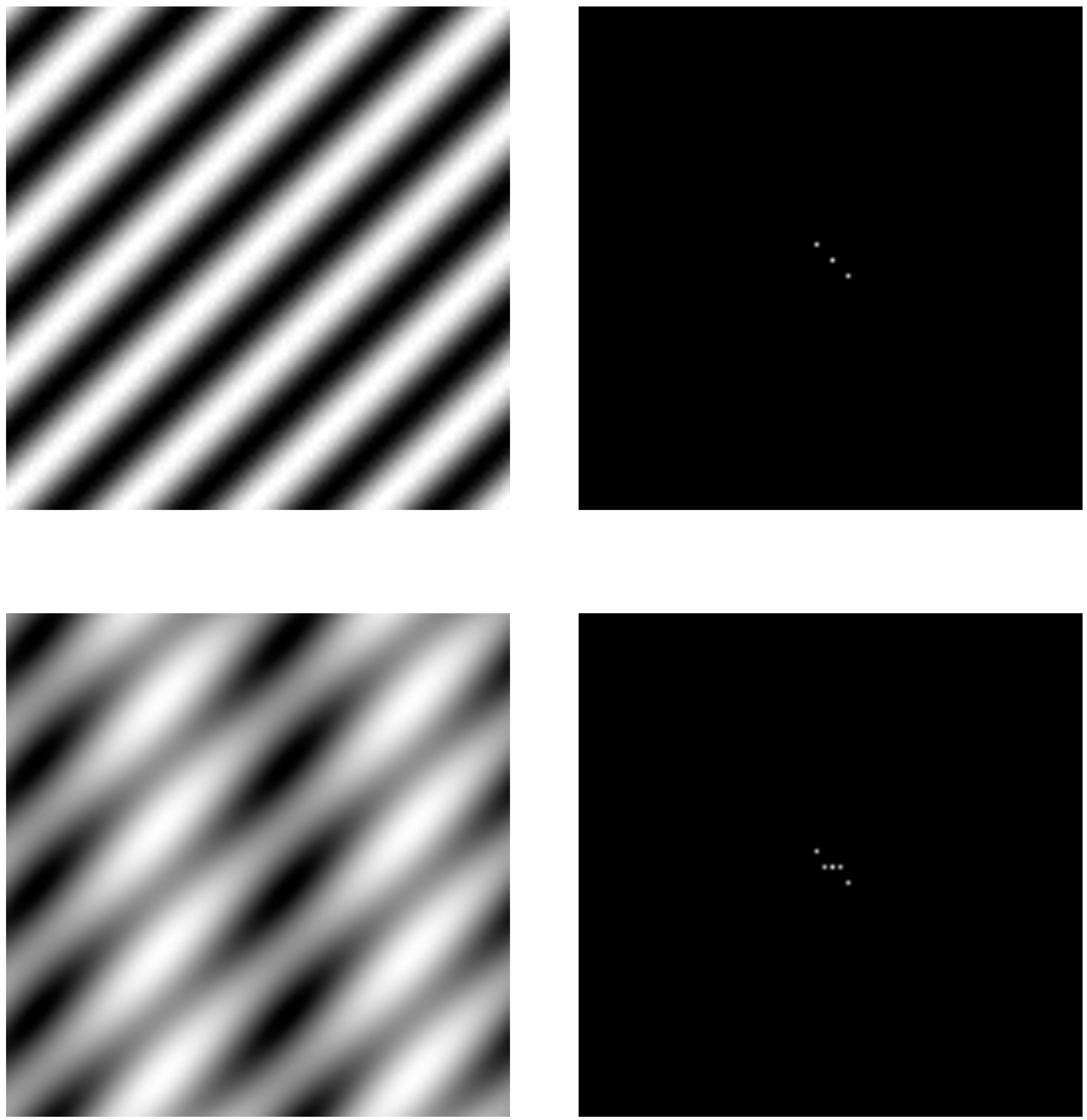
Gy

- Image → Image to Spectrum → apply filter → Spectrum to Image = filtered image
  - "Image to Spectrum": Fast Fourier Transform
  - "Spectrum to Image": Inverse Fast Fourier Transform

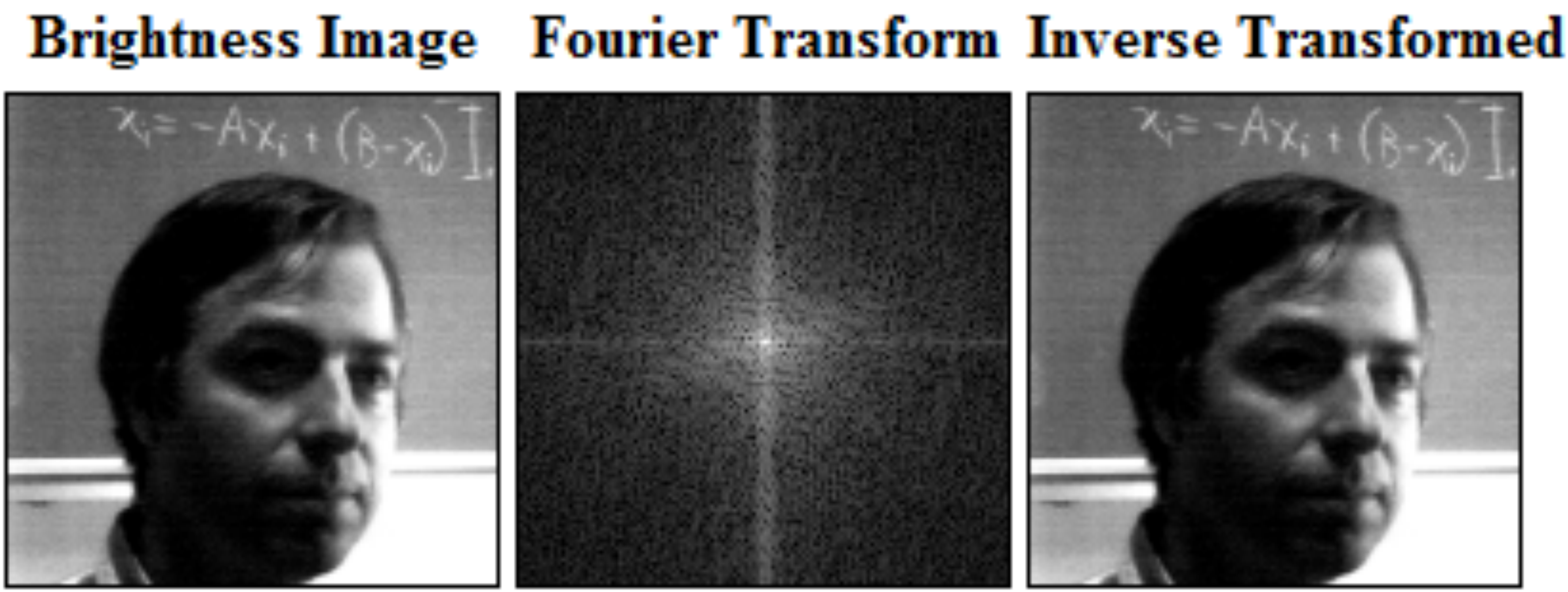
Brightness image – Fourier transform



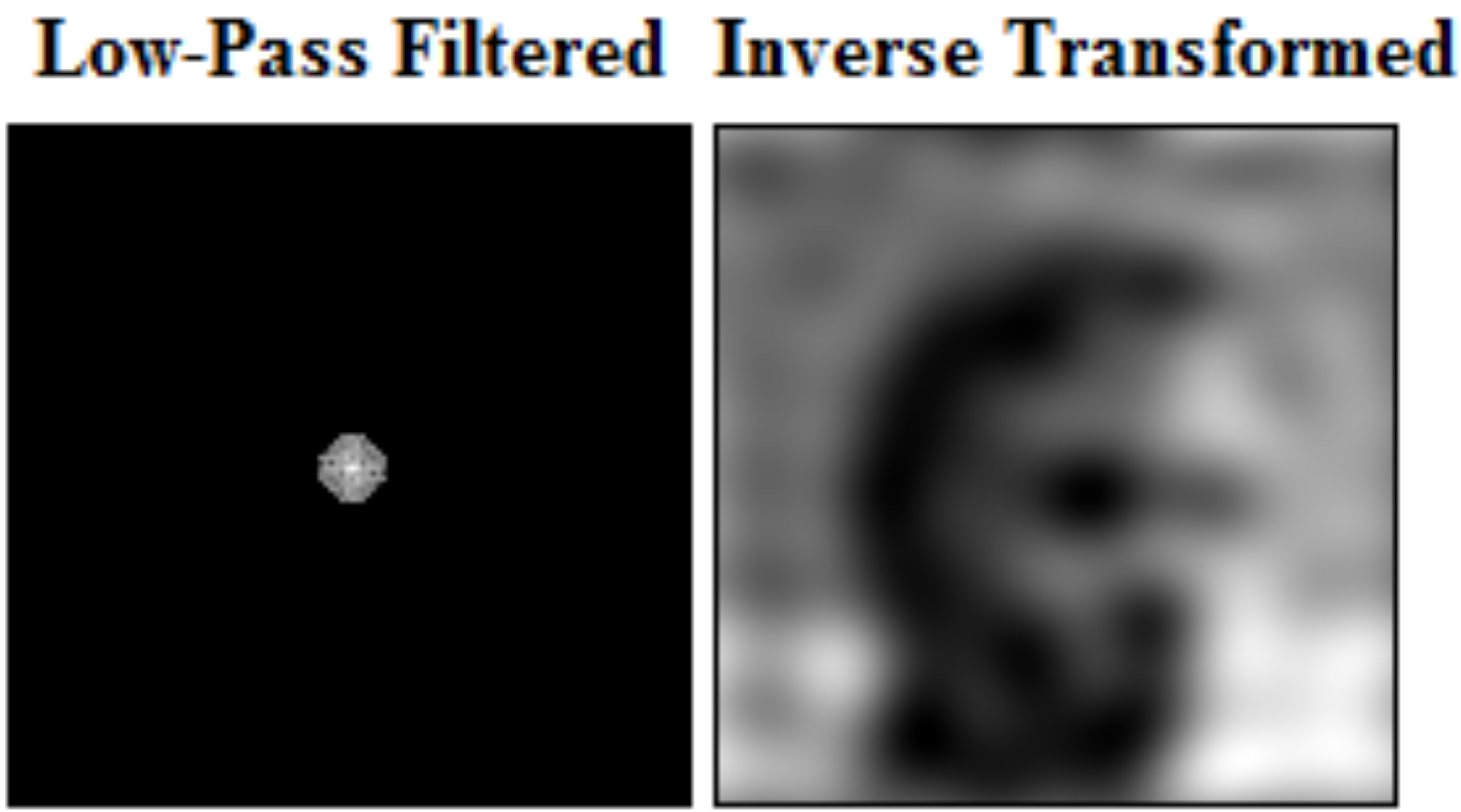
Brightness image – Fourier transform



Brightness image – Fourier transform – Brightness image

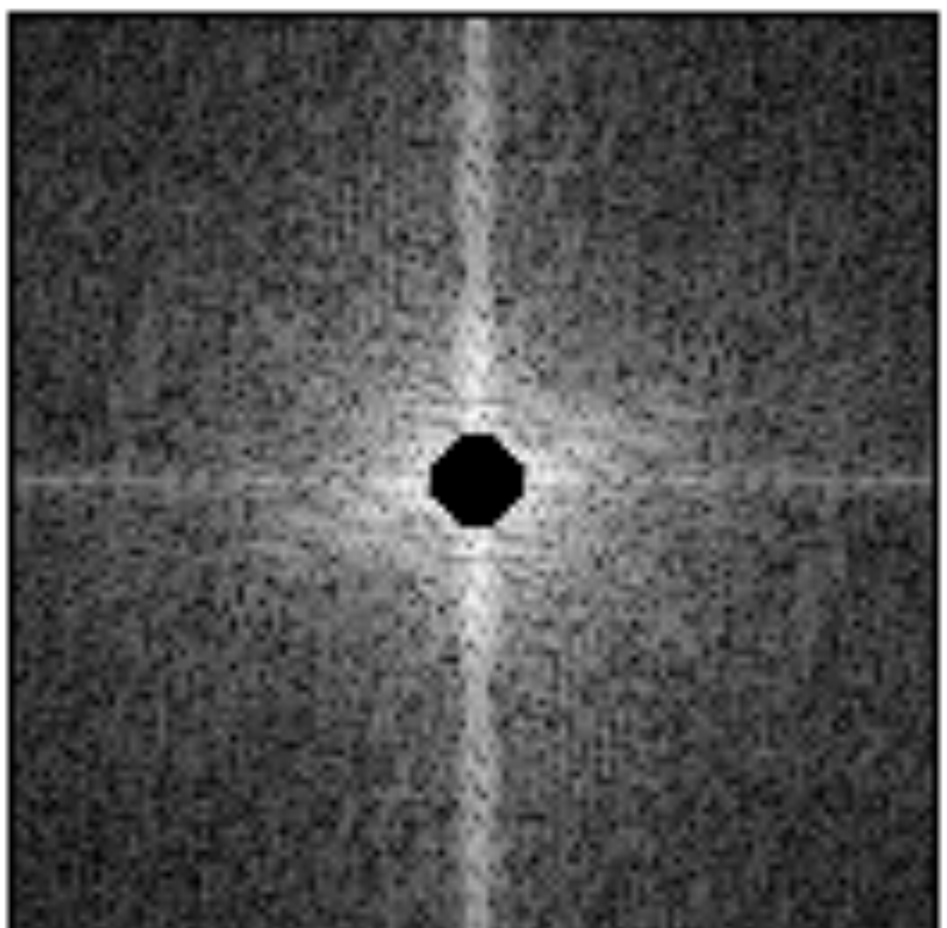


Filtering in the Fourier domain: keeping the low spatial frequencies only

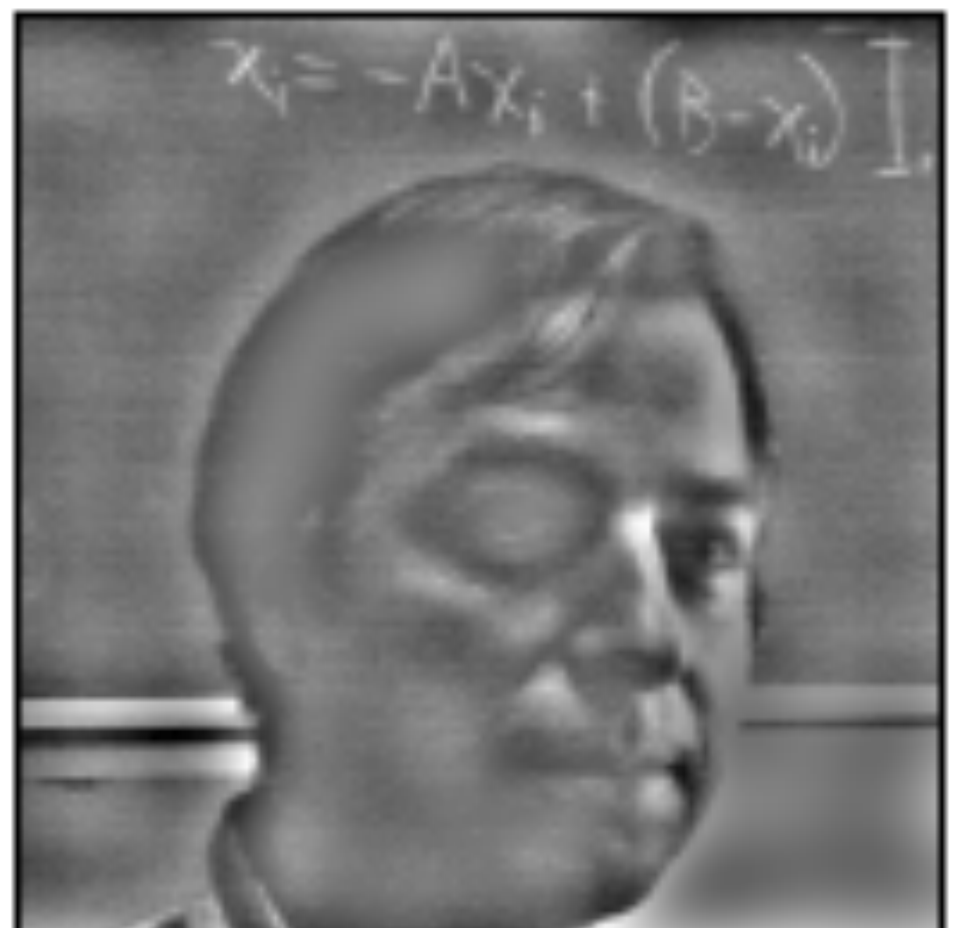


Filtering in the Fourier domain: keeping the high spatial frequencies only

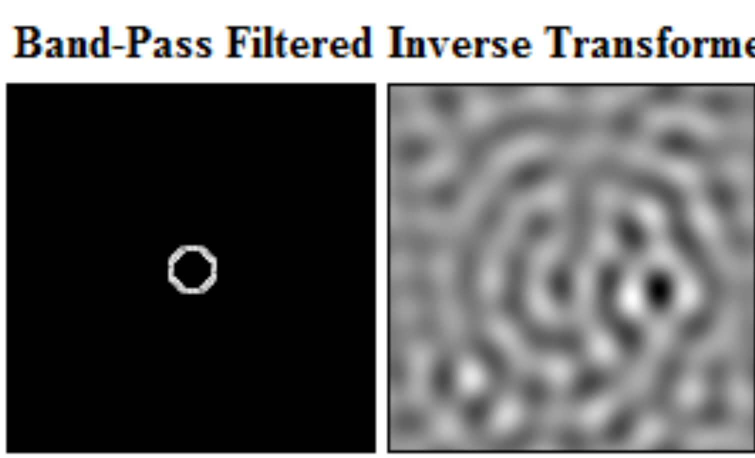
High-Pass Filtered

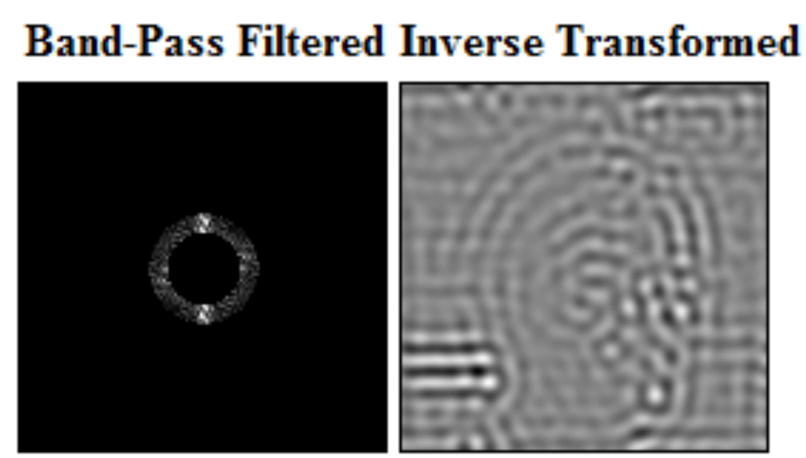


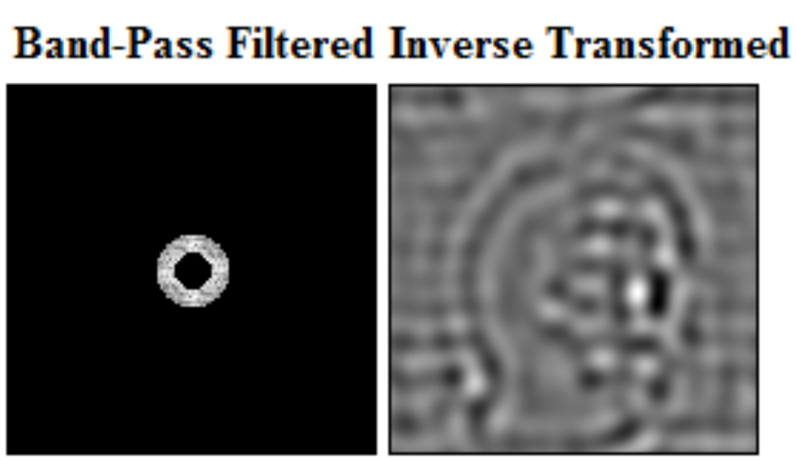
Inverse Transformed

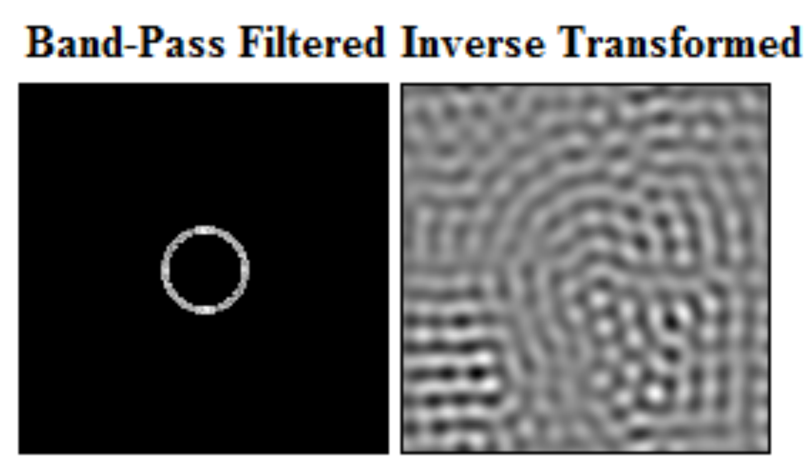


Filtering in the Fourier domain: keeping the intermediate spatial frequencies only



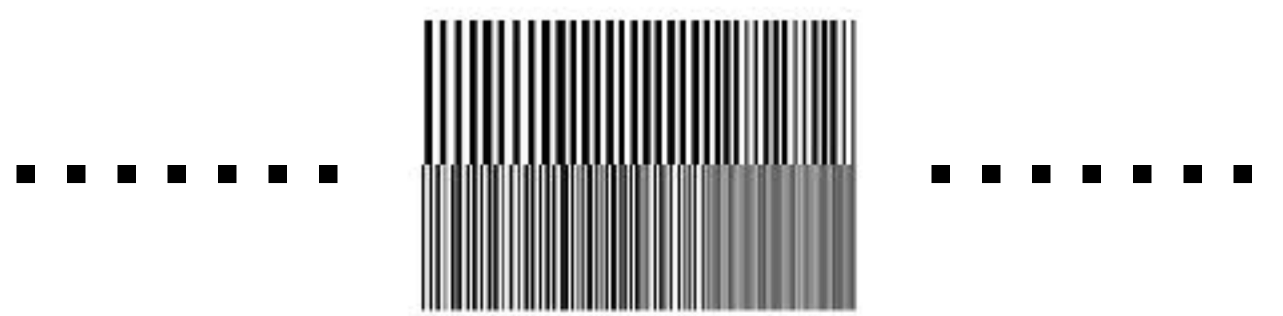






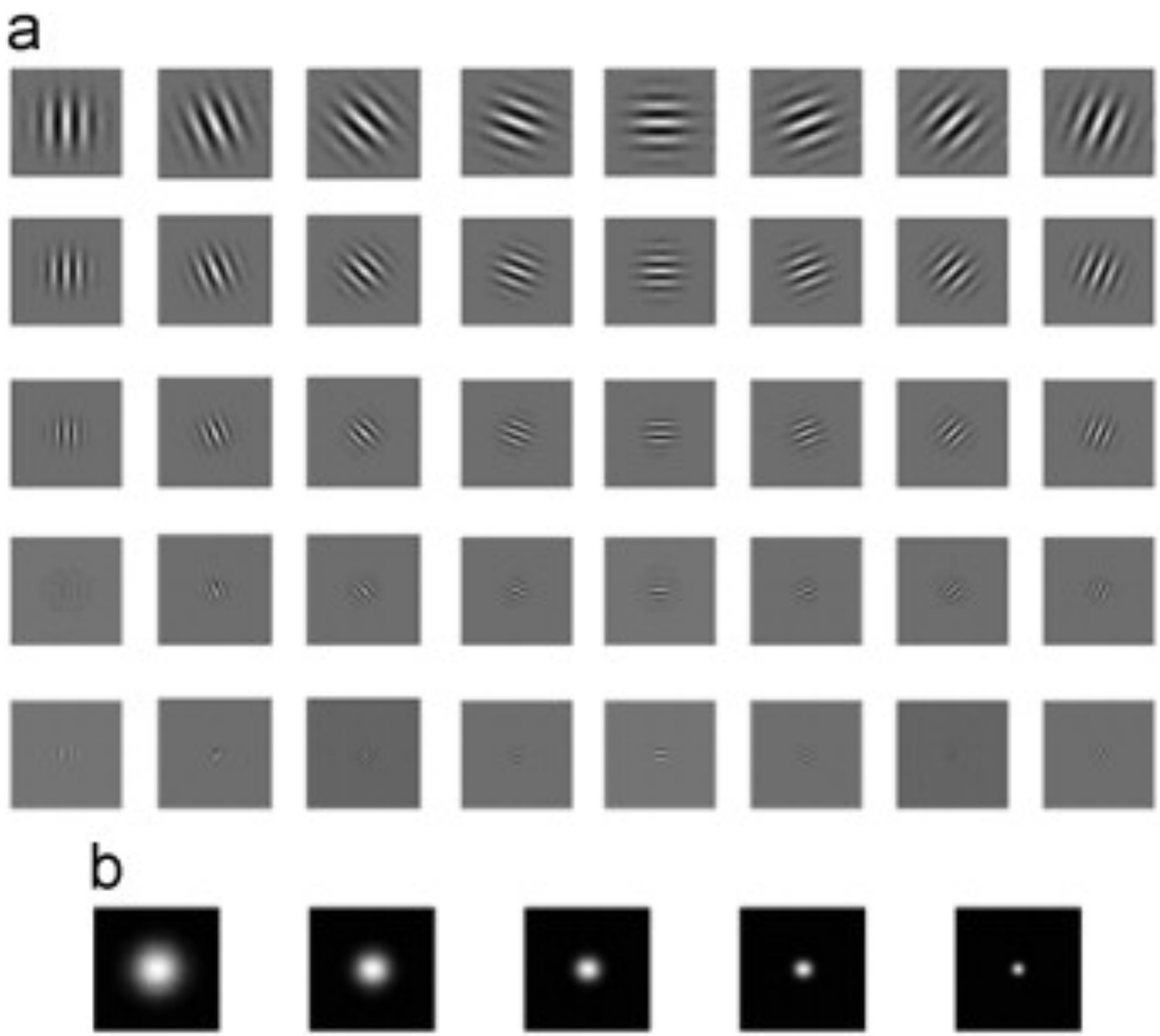
Fourier transform versus Gabor transform

- Fourier analysis decomposes images in terms of global spatial frequencies (they continue forever)



- Gabor analysis decomposes image in terms of LOCAL spatial frequencies (they do not continue forever)

Convolution with Gabor filters (see this week's paper)



TILBURG ◆ UNIVERSITY

Facial expression coding with local Fourier analysis

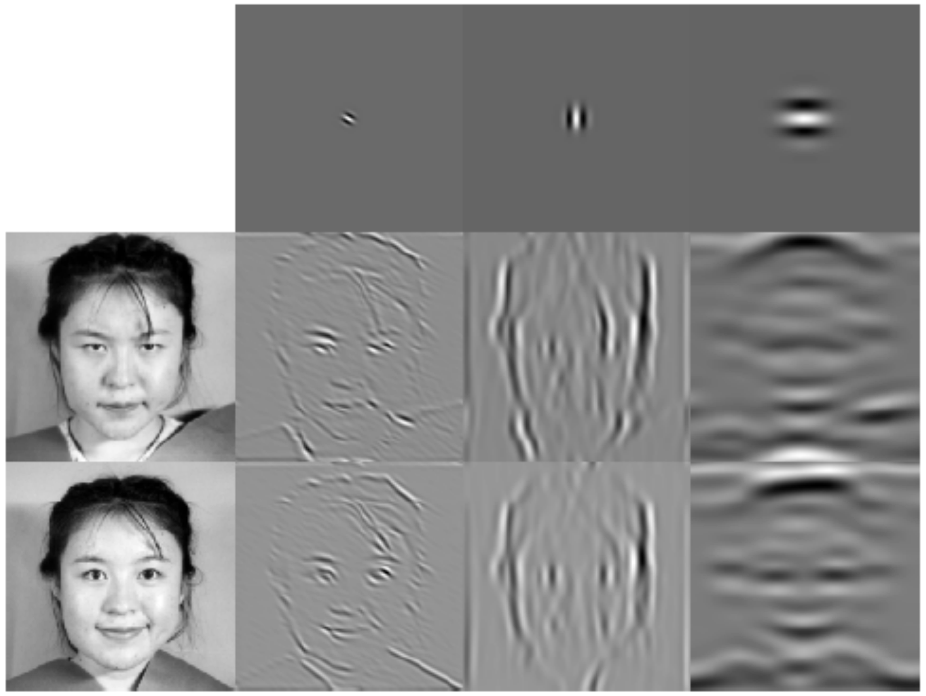


Figure 1. Examples of Gabor filter responses to two facial expression images for three of the filters used.

Correspondence with human ratings

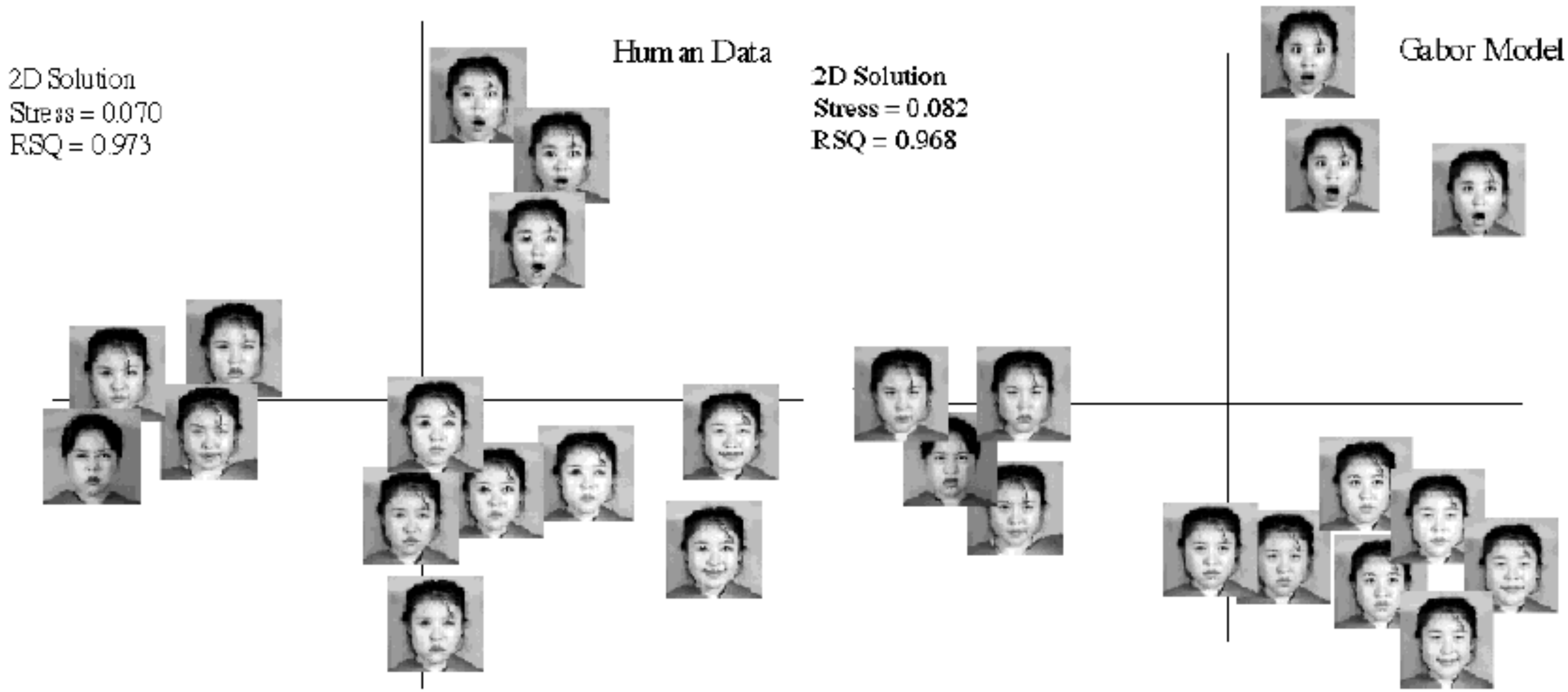
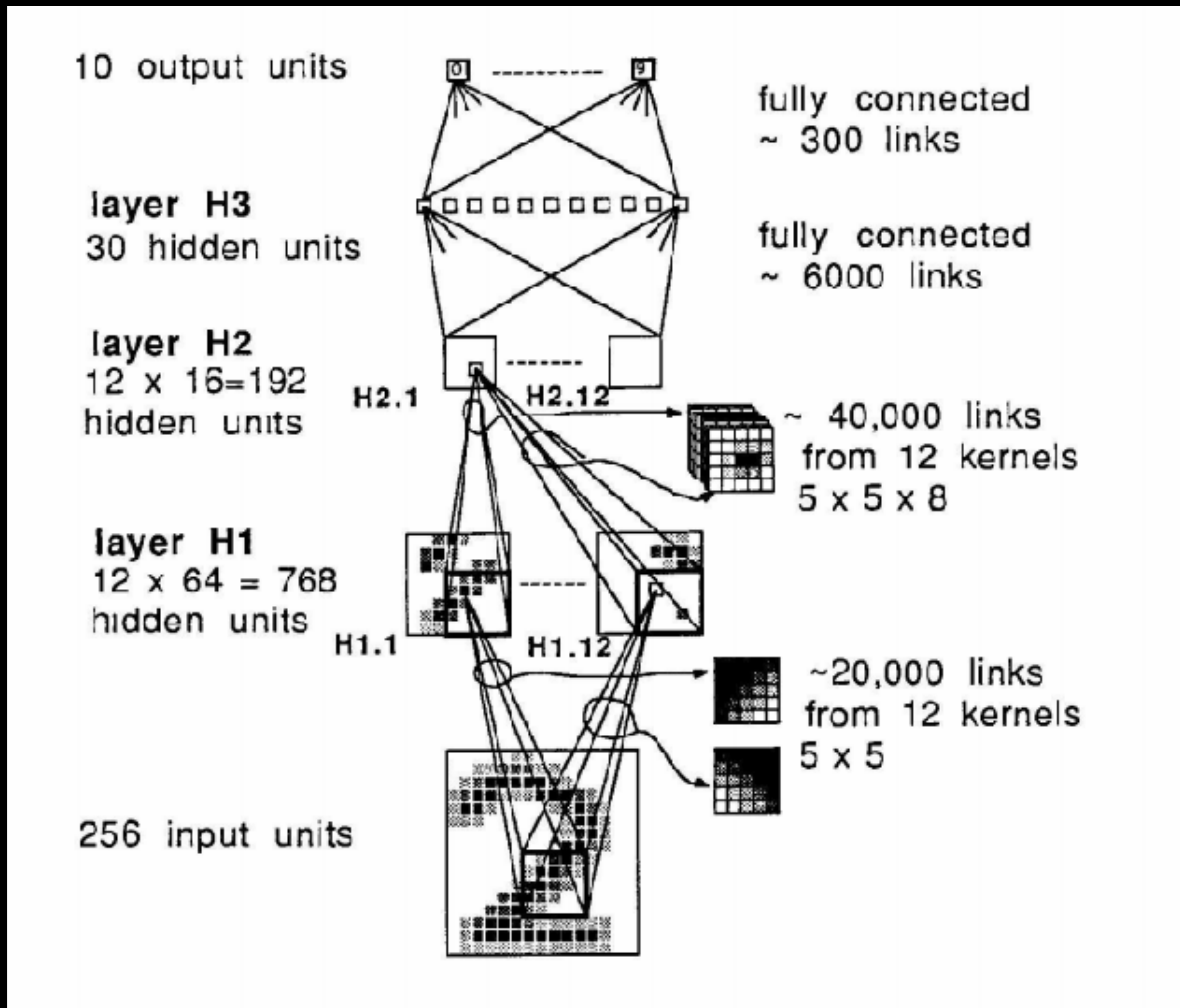


Figure 5. nMDS solutions for Gabor and semantic rating similarities (data from pilot study).

Exercise with convolution

LeCun et al. (1989)



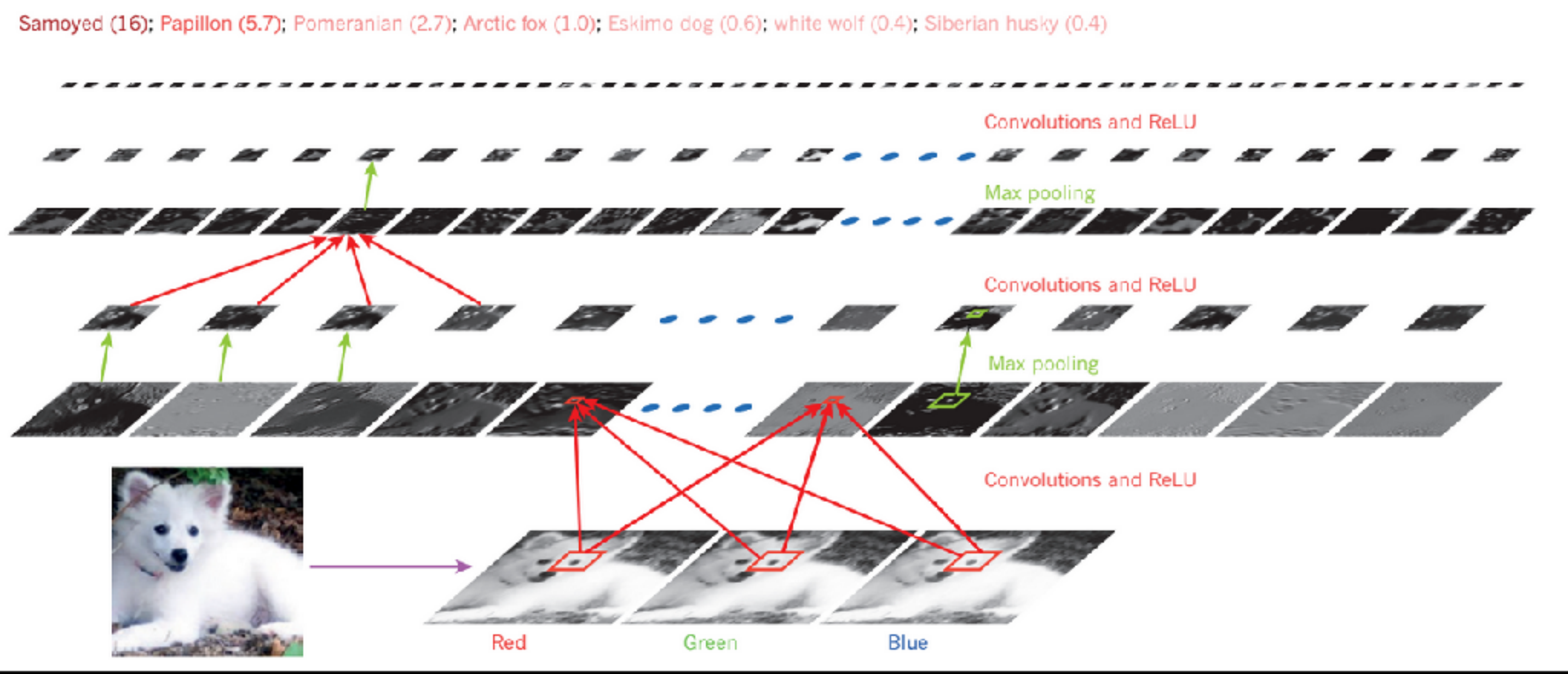
opinions1990-2012

- “It is very hard to train multilayer networks with more than 1 hidden layer”
- “This is a fundamental limit of multilayer perceptrons and (stochastic) gradient descent learning”

  Shift towards other machine learning methods (SVM, graphical models, …)

Deep Learning (2015)

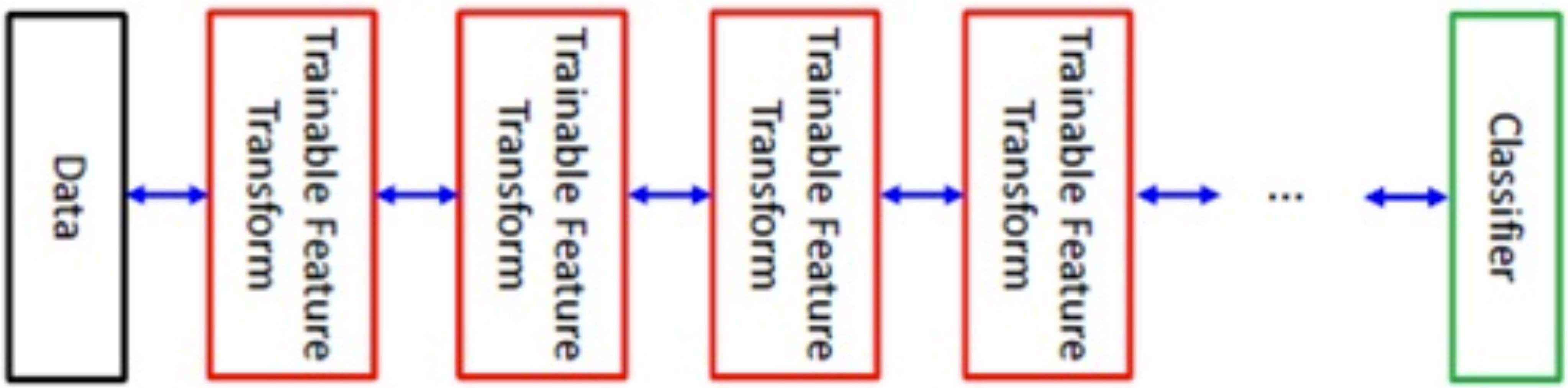
Convolutional Neural Network (LeCun)



Automatically Learns Complex Non-Linear Mappings

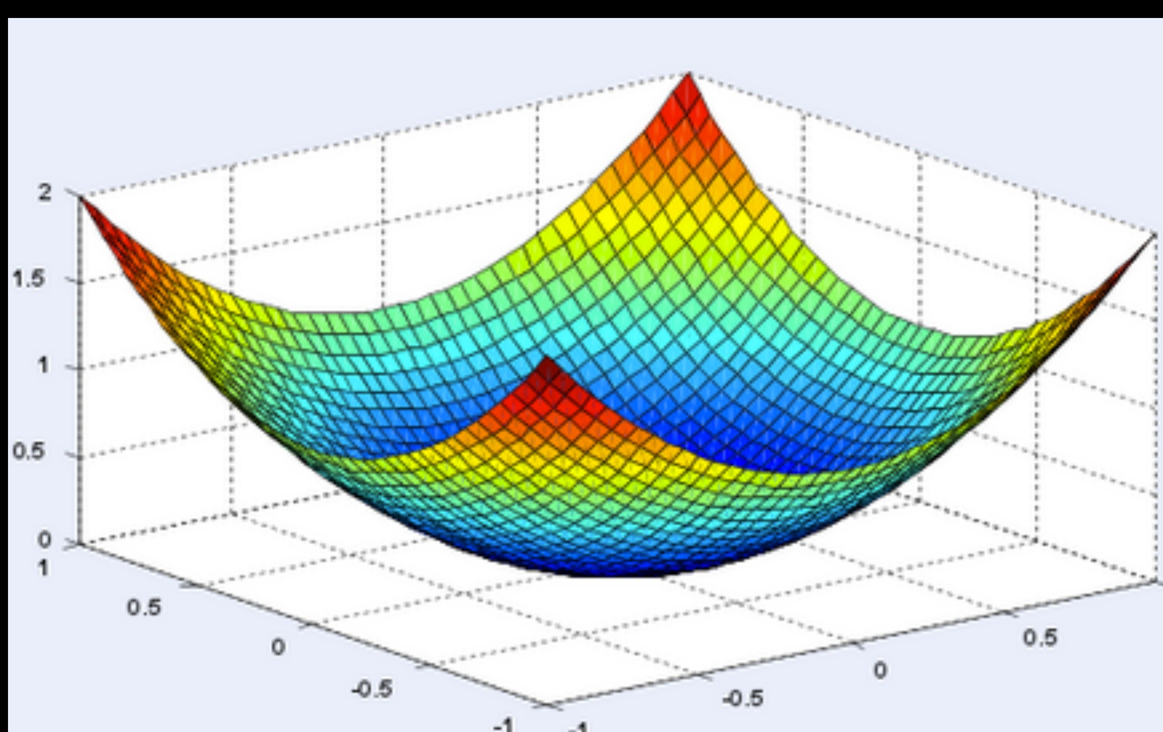
Stochastic Gradient Descent

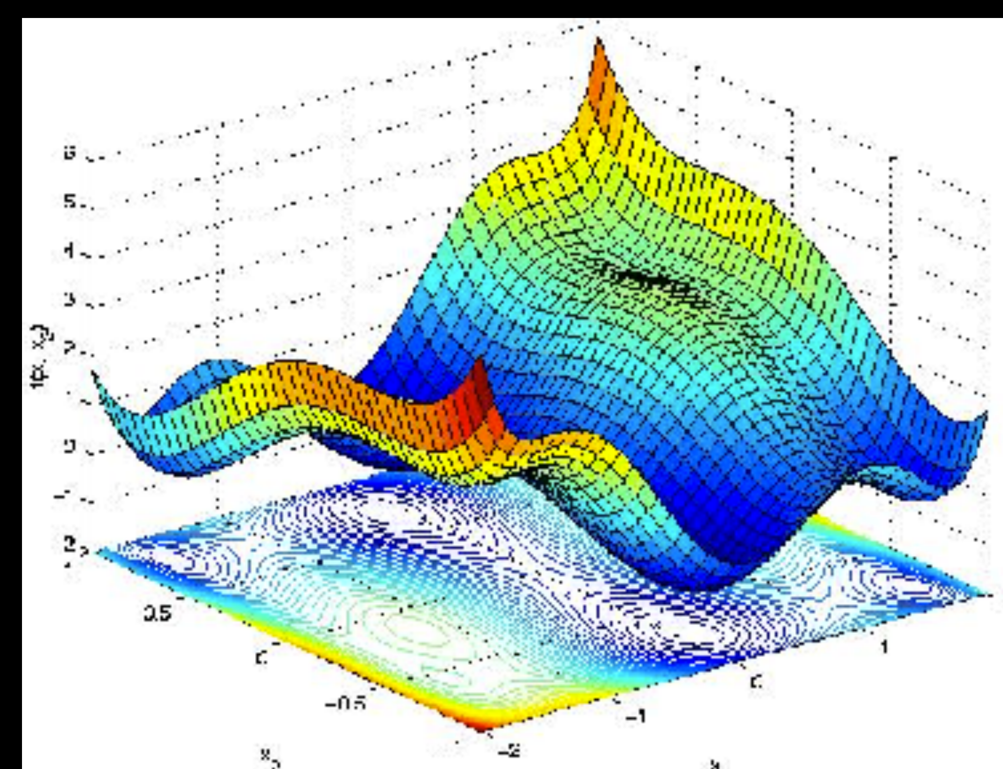
$$\mathbf{y} = F(\mathbf{W}^k \cdot F(\mathbf{W}^{k-1} \cdot F(\ldots F(\mathbf{W}^0 \cdot \mathbf{x}))$$

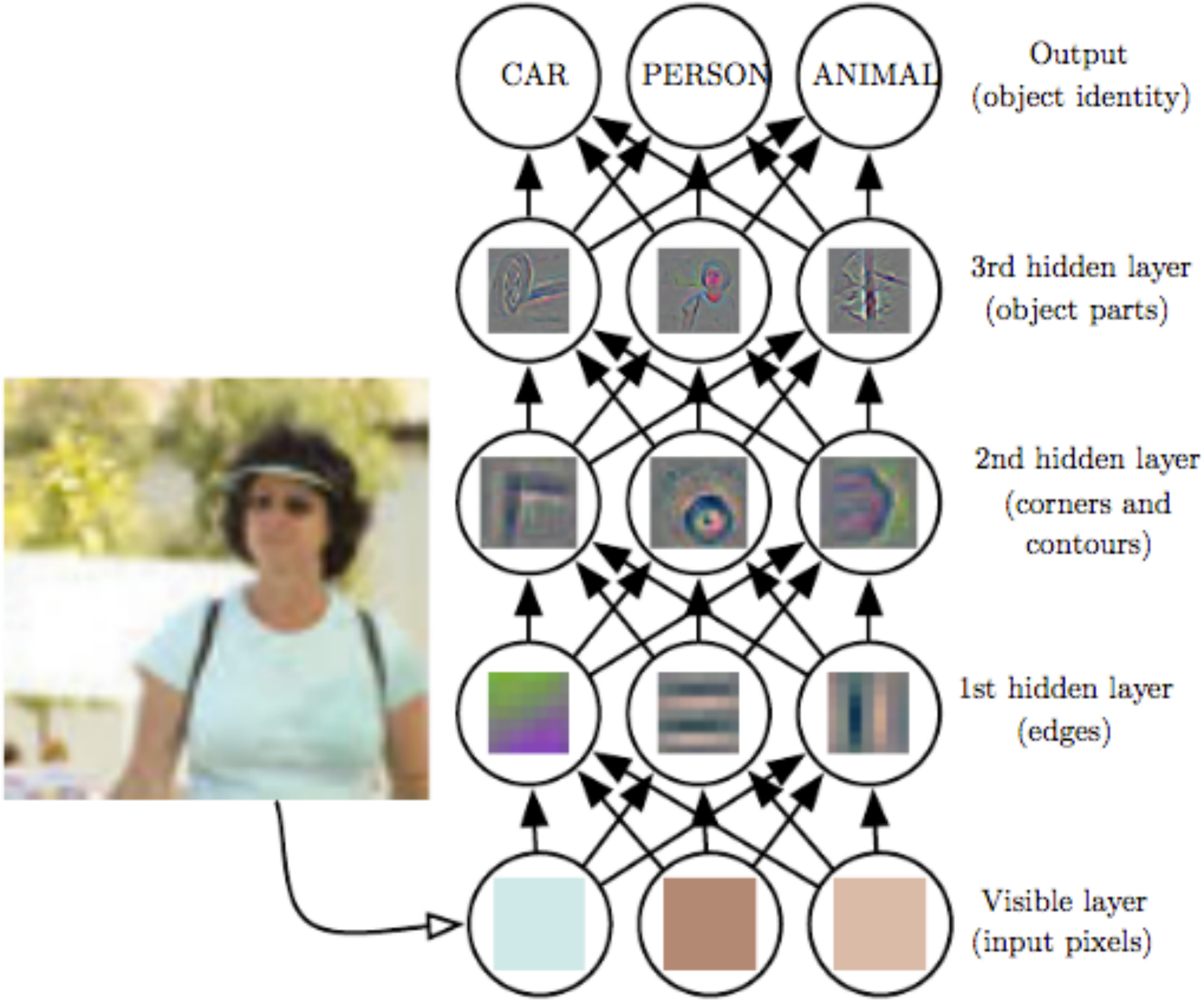


Training

- Training a Deep Convolutional Neural Network
- Non-convex optimisation problem
- Convex versus non-convex optimisation

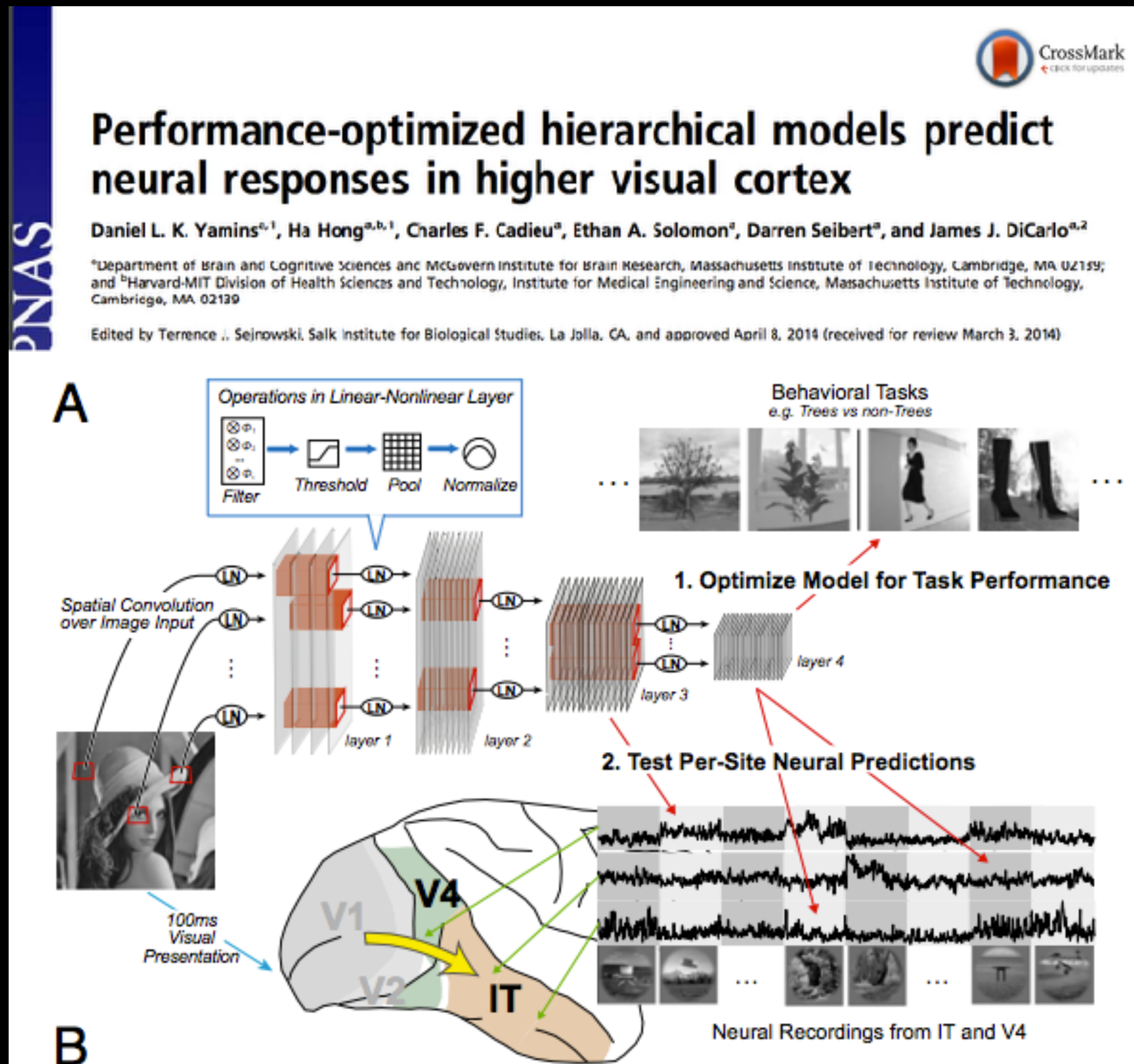






http://www.deeplearningbook.org/

Biological Plausibility


PNAS

CrossMark

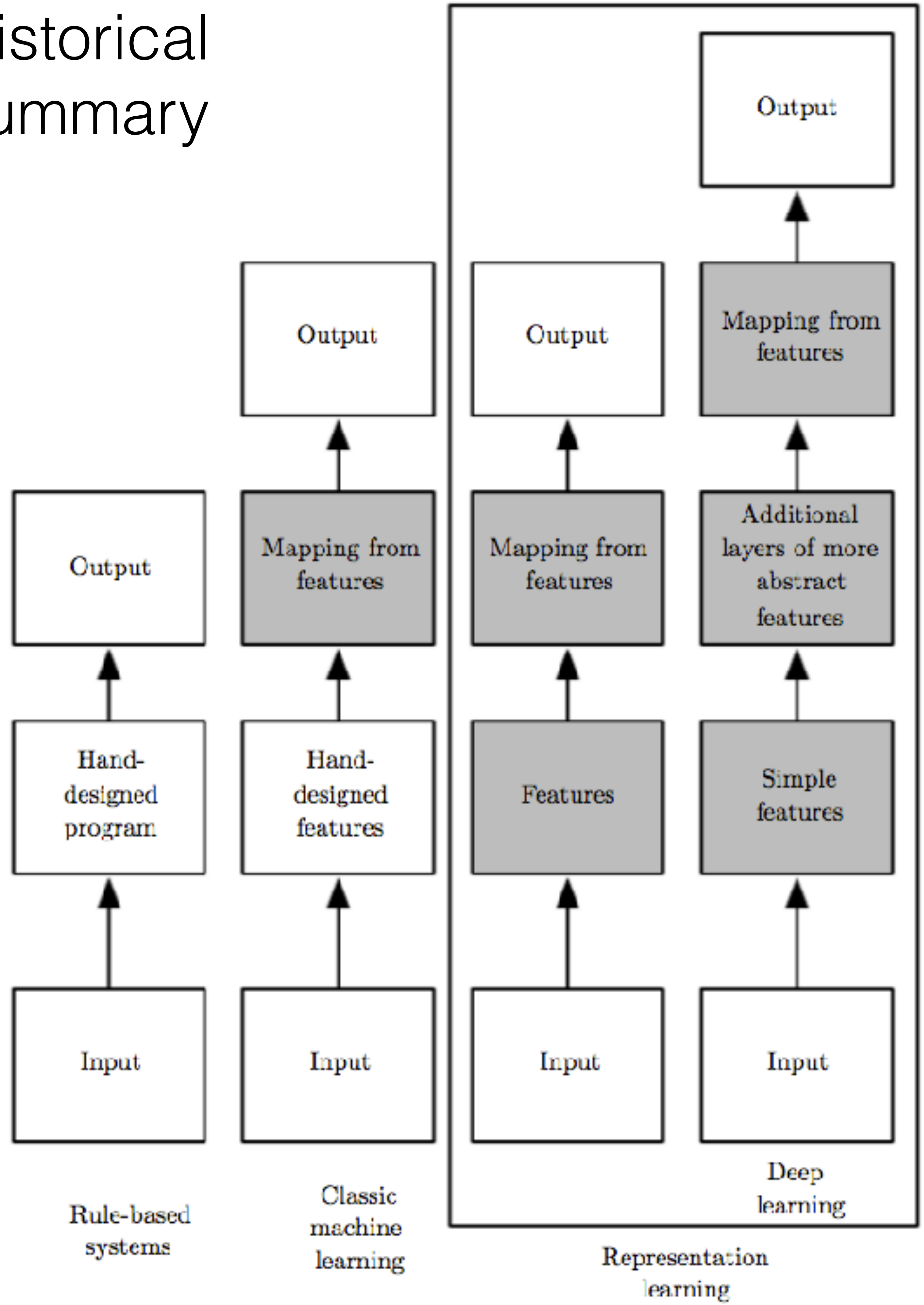
Performance-optimized hierarchical models predict neural responses in higher visual cortex

Daniel L. K. Yamins[a,1], Ha Hong[a,b,1], Charles F. Cadieu[a], Ethan A. Solomon[a], Darren Seibert[a], and James J. DiCarlo[a,2]

[a]Department of Brain and Cognitive Sciences and McGovern Institute for Brain Research, Massachusetts Institute of Technology, Cambridge, MA 02139; and [b]Harvard-MIT Division of Health Sciences and Technology, Institute for Medical Engineering and Science, Massachusetts Institute of Technology, Cambridge, MA 02139

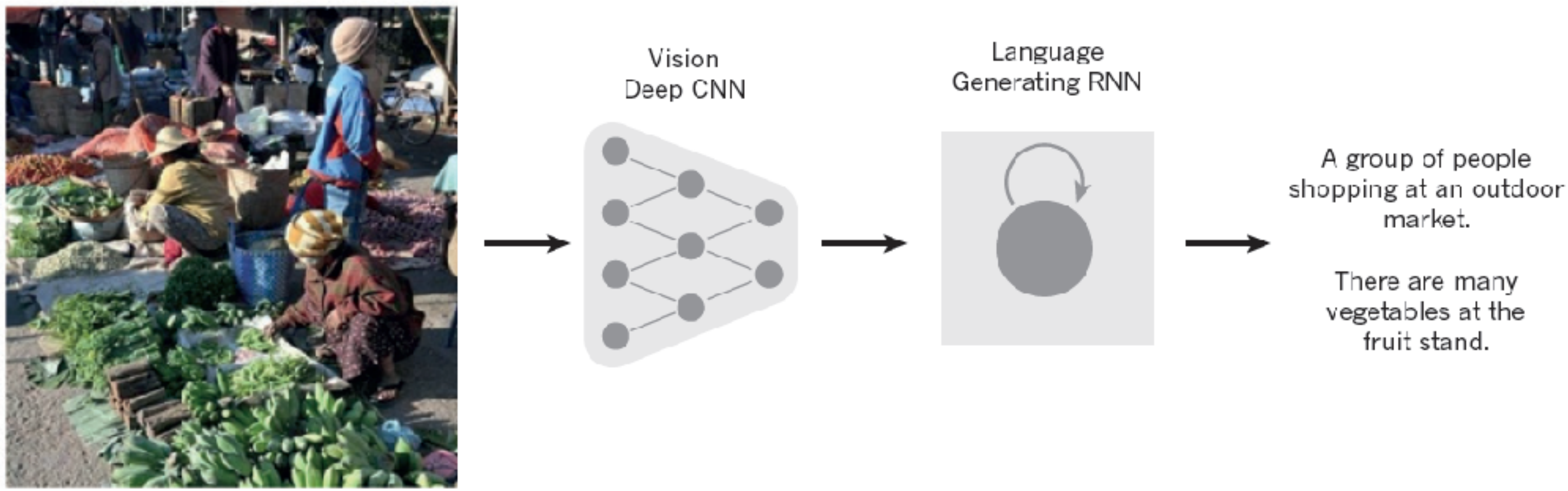
Edited by Terrence J. Sejnowski, Salk Institute for Biological Studies, La Jolla, CA, and approved April 8, 2014 (received for review March 3, 2014)

Historical summary



http://www.deeplearningbook.org/

Reading Images (FaceBook)



A woman is throwing a **frisbee** in a park.

A **dog** is standing on a hardwood floor.

A **stop** sign is on a road with a mountain in the background

A group of **people** sitting on a boat in the water.

Applications of Deep Learning

(Feedforward networks, recurrent networks, reinforcement learning, …)

- Speech recognition/translation
- Image and object recognition
- Control tasks (playing video games)
- Language generation

COMPUTING NEWS

3 COMMENTS

Deep Learning Catches On in New Industries, from Fashion to Finance

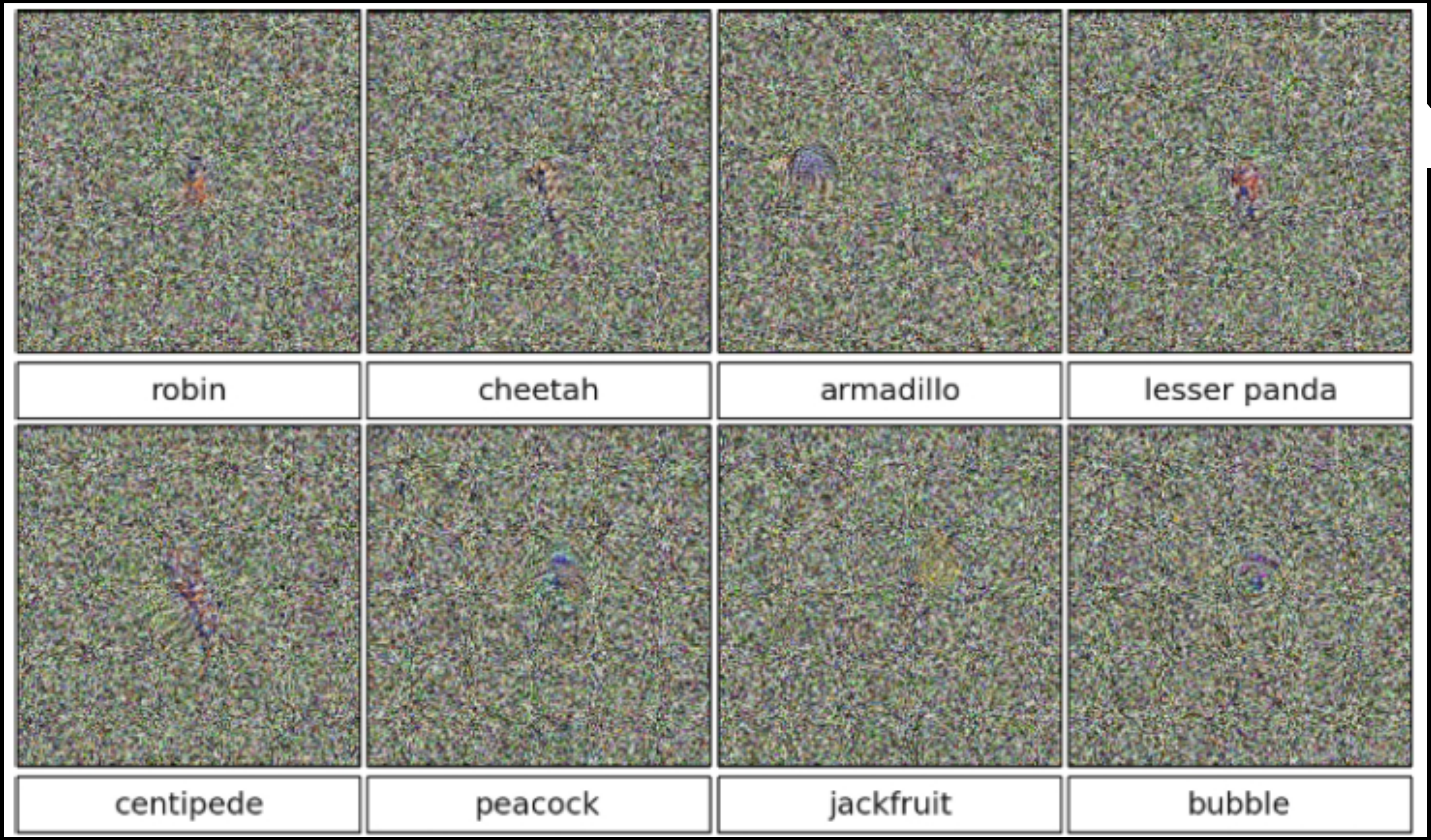
The machine-learning technique known as deep learning, which has shown impressive results in voice and image recognition, is finding new applications.

By Will Knight on May 31, 2015

Google's AI beats world Go champion in first of five matches

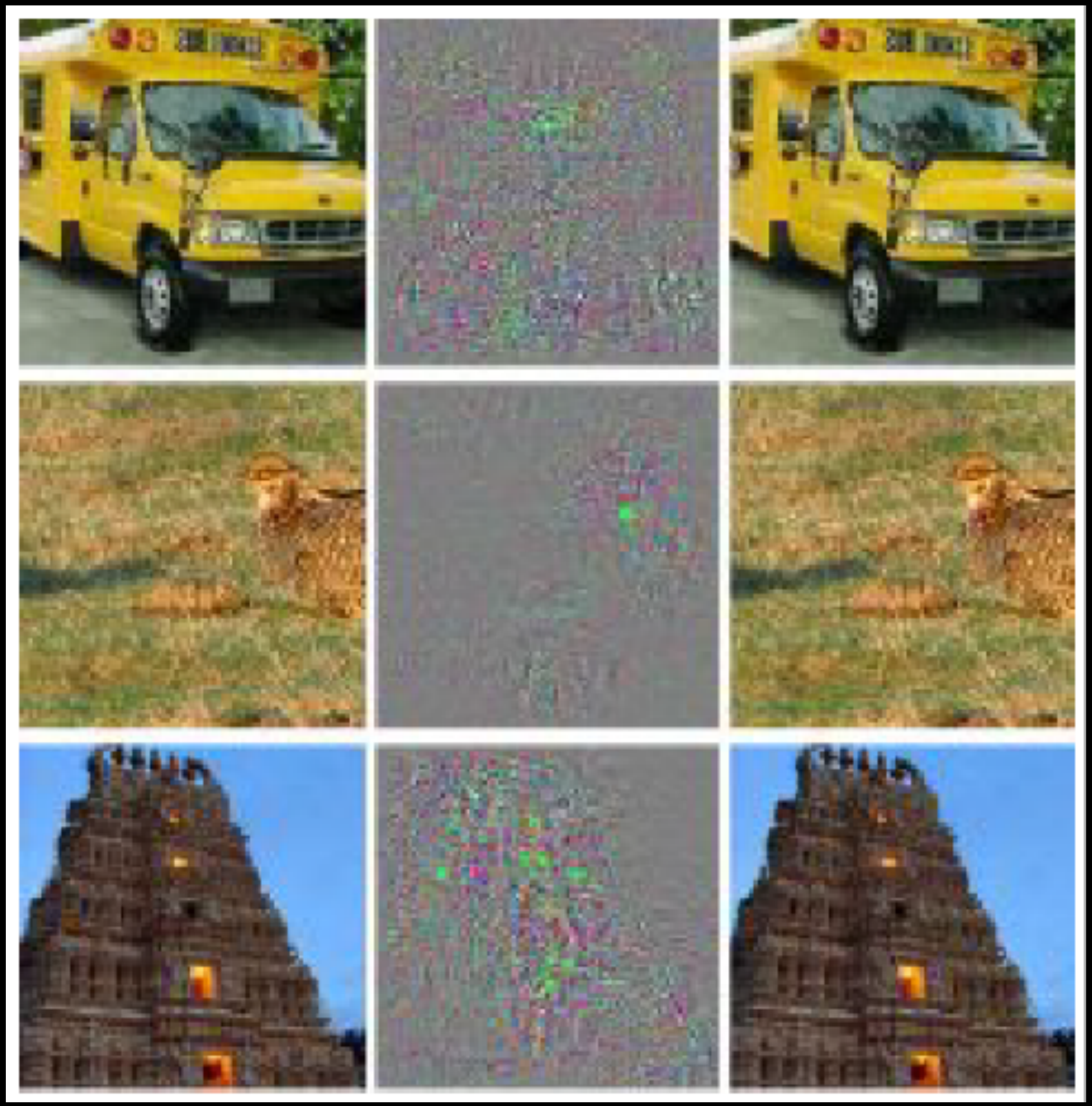
9 March 2016 | Technology

Hands-on exercise 3



Nguyen, Yosinski & Clune (1994)

Deep Learning Flaws



correctly classified

incorrectly classified

Szegedy et al. (2014)

Deep Learning versus Statistics

- Deep Learning:
  prediction - opaque
- Statistics:
  inference - transparant

representations are
like latent factors

CNNs representations offer some transparancy