

# Concepte și aplicații în Vederea Artificială

Bogdan Alexe

[bogdan.alexe@fmi.unibuc.ro](mailto:bogdan.alexe@fmi.unibuc.ro)

Radu Ionescu

[radu.ionescu@fmi.unibuc.ro](mailto:radu.ionescu@fmi.unibuc.ro)

Curs optional  
semestrul I, 2023-2024

# Lista studenților care au trimis rezolvarea pentru Tema 1

Nr. crt.	Nume student	Grupa					
1	Lungu Elena Izabela	311	23	Pomparau Renato-Emil	341	46	Bora Dragos
2	Anton Marius Alexandru	331	24	Dumitrasche Flavian	342	47	Cretu Laurentiu Vasile
3	Ilie Andrei-Virgil	331	25	Vrinceanu Radu-Tudor	342	48	David Victor
4	Popovici Antonia	331	26	Besliu Radu Stefan	343	49	Iancu Florentina Mihhaela
5	Talpiga Andrei Vlad	331	27	Crivoi Carla	343	50	Sorete Rbert
6	Tindeche Alexandru	331	28	Cucos Maria Marianita	343	51	Spataru Catalin Gabriel
7	Corolevschi Mihai	332	29	Ghetoiu Gheorghe Laurentiu	343	52	Ghergu Nicolae Marius
8	Ivan Bogdan	332	30	Hutan Mihai Alexandru	343	53	Holmanu Antonio Marius
9	Patrascu Adrian Octavian	332	31	Militaru Mihai Alexandru	343	54	Licu Mihai George
10	Rincu Stefania	332	32	Nechita Maria Ilinca	343	55	Stanciu Alexandra Andreea
11	Ancuta Andrei	333	33	Tanasa Florin Petrisor	343	56	Statescu Relu
12	Isache Maria Catalina	333	34	State Giulia Antonia	344	57	Cucu Stefan catalin
13	Titiriga Tiberiu Nicolae	333	35	Tanase Victor Flavian	344	58	Dura Bogdan
14	Ionescu Radu Constantin	334	36	Birsan Vlad Ioan	351	59	Son Andreea Marina
15	Mura Victor	334	37	Gheorghe Radu Mihai	351	60	Udrea Iulia Maria
16	Petcu Mircea	334	38	Popa Stefan	351	61	Dinoiu Anastasia
17	Rotaru Iulia Maria	334	39	Rosu Vlad Andrei	351	62	Alpetri Iulita
18	Sabau Eduard	334	40	Sava Constantin Alin	351	63	Cioclov Maria-Simona
19	Stefanou Rares	334	41	Timbur Cristina	351	64	Moarcas Cosmin
20	Talpaliu Iulia Georgiana	334	42	Oprea Mihai Adrian	352	65	Putineanu Bogdan
21	Gontescu Maria Ruxandra	341	43	Pascu Ionut Alexandru	352	66	Stan Catalin Andrei
22	Mihai Dragos Vasile	341	44	Stan Catalin Andrei	352		
			45	Trifan Robert Gabriel	352		

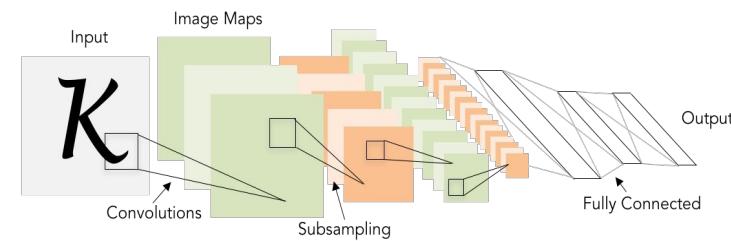
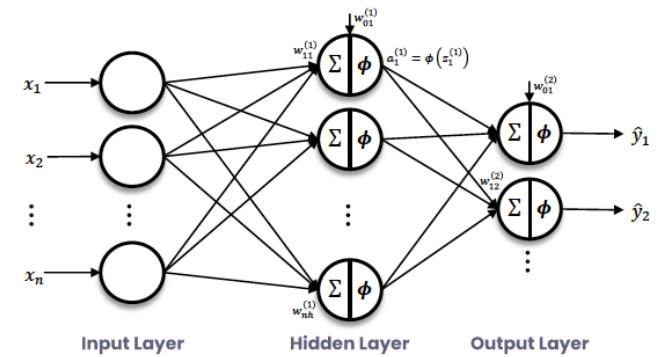
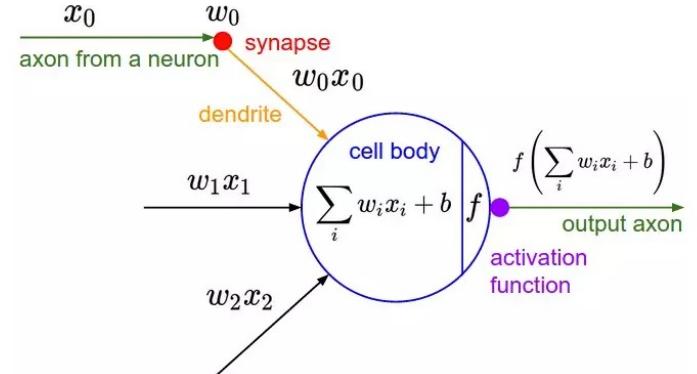
# Cursul trecut

- Modelul Bag of Visual Words
  - privire de ansamblu
  - descriptorul SIFT
  - clusterizarea k-means
  - piramida spațială



# Cursul de azi

- Perceptron
- Rețele neuronale feedforward multistrat de perceptroni
- Rețele neuronale convolutionale

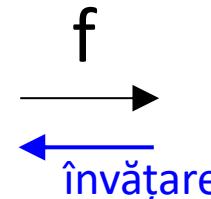


# De la extragere de caracteristici la învățare end-to-end



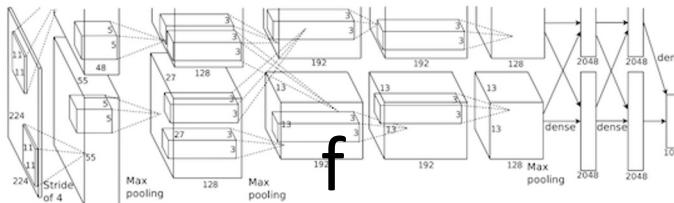
Extragere de  
caracteristici

Vectori ce descriu conținutul vizual al unei imagini



N numere, ce indică scorurile claselor

[32x32x3]



învățare

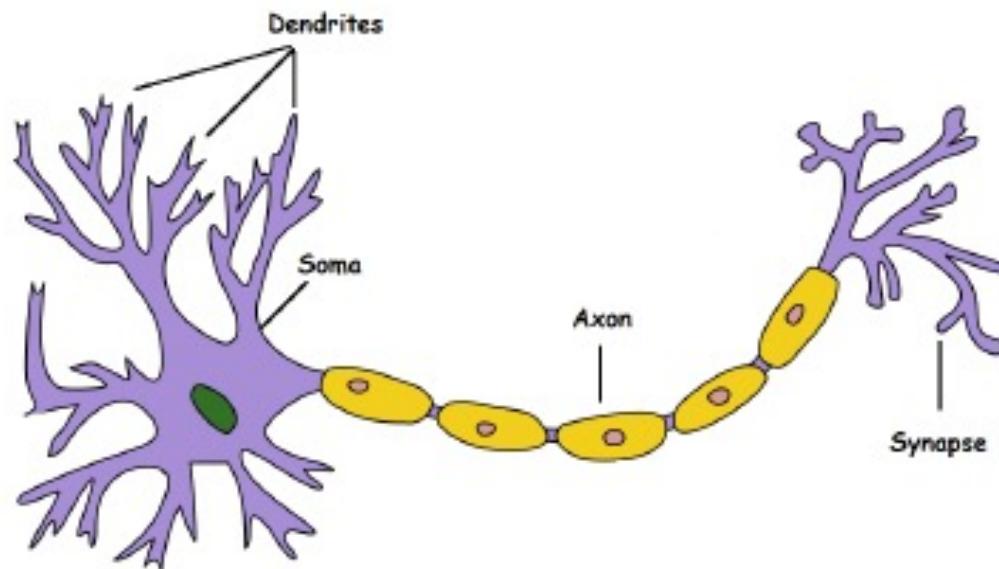
N numere, ce indică scorurile claselor

[32x32x3]

# Neuronul biologic

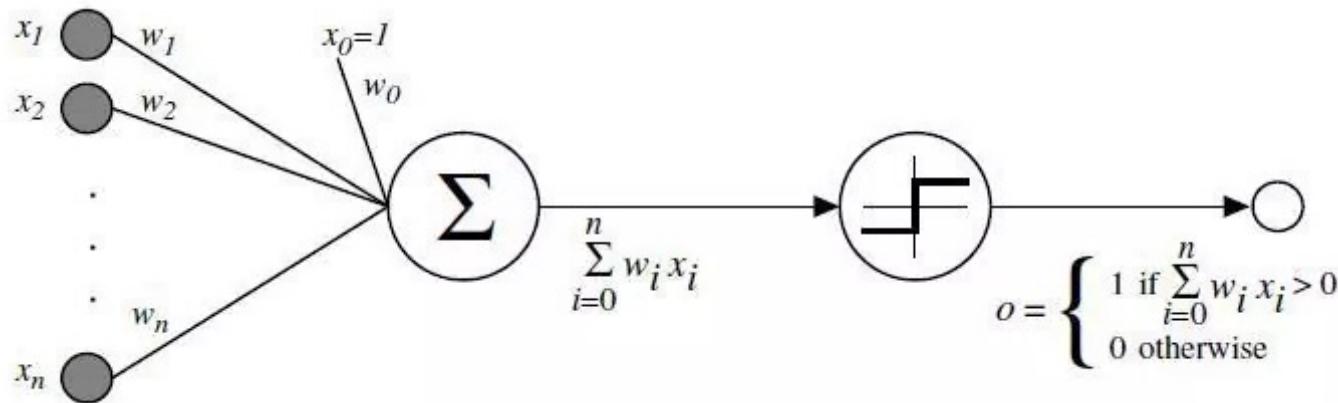
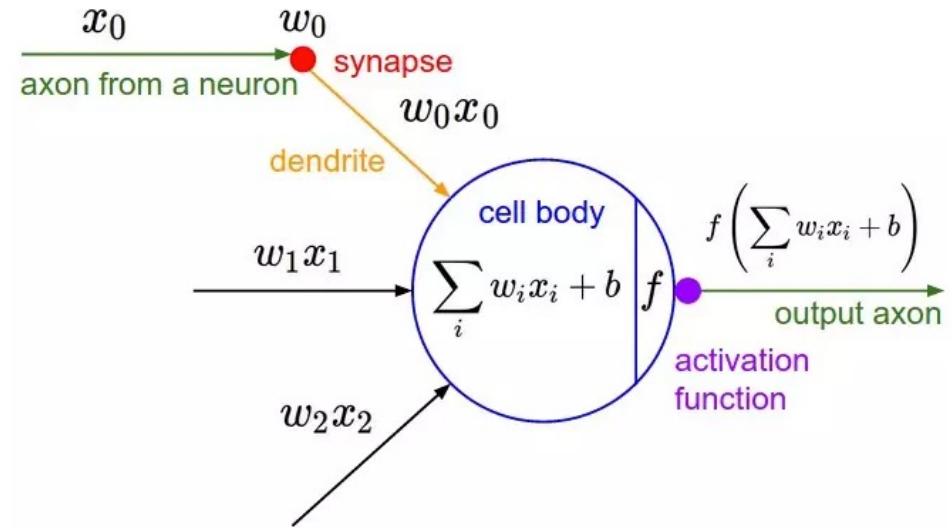
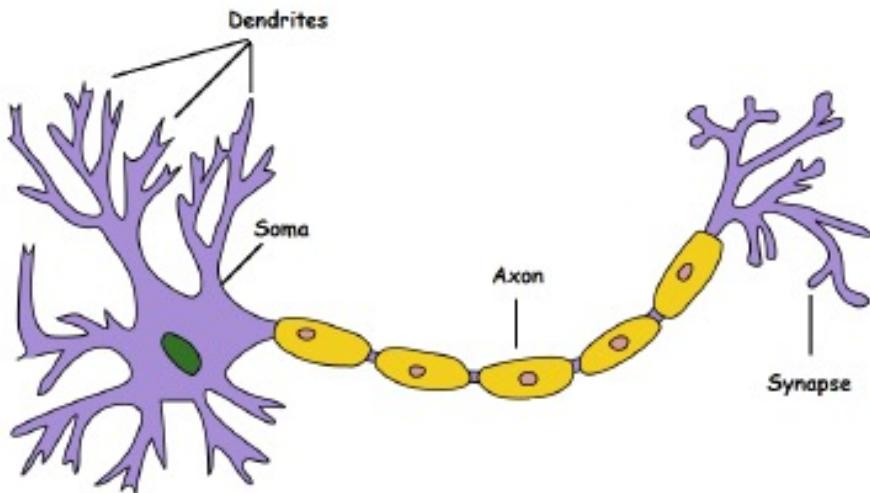
Cum funcționează un neuron:

- semnale electrice (intrări) sunt transmise prin **dendrite**
- aceste semnale duc la acumularea unui potențial electric în **corpul neuronului (soma)**
- când potențialul electric acumulat depășește un anumit prag (threshold) un semnal electric (ieșirea neuronului) este transmis prin intermediul unui **axon**
- conectarea axonului cu dendritele altor neuroni se realizează prin **sinapse**



# Perceptronul

Perceptronul este un clasificator liniar inspirat de neuronul uman.



# Algoritmul de învățare al perceptronului

- Mulțimea de exemple de antrenare:

$$E = \{(\vec{x}^{(1)}, y^{(1)}), \dots, (\vec{x}^{(m)}, y^{(m)})\}, \vec{x}^{(i)} \in \{0,1\}^{n+1} (x_0^{(i)} = 1, \forall i), y^{(i)} \in \{0,1\}$$

- Ponderile  $w_j$  sunt inițializate cu 0 (pe acestea le învățăm)
- Pentru fiecare exemplu de antrenare  $(\vec{x}^{(i)}, y^{(i)})$  din mulțimea E:
  - dacă eticheta prezisă = eticheta reală  $\hat{y}^{(i)} == y^{(i)}$  nu facem actualizare
  - dacă  $\hat{y}^{(i)} == 0$  și  $y^{(i)} == 1$  creștem ponderile tuturor intrărilor active
  - dacă  $\hat{y}^{(i)} == 1$  și  $y^{(i)} == 0$  scădem ponderile tuturor intrărilor active
- Regula de actualizare a perceptronului

$$w_j = w_j + \Delta w_j, \text{ unde}$$

$$\Delta w_j = \eta(y^{(i)} - \hat{y}^{(i)}) \cdot x_j^{(i)}$$

Cantitatea prin care schimbăm ponderea  
(rata de învățare)

Setează semnul actualizării

Selectează intrările active

$$\hat{y}^{(i)} = \begin{cases} 1 & \text{if } \sum_{j=0}^n w_j x_j^{(i)} \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

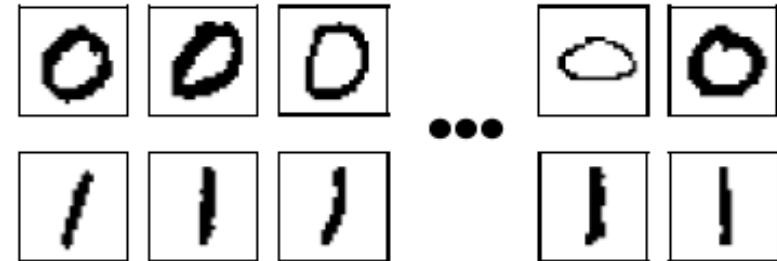
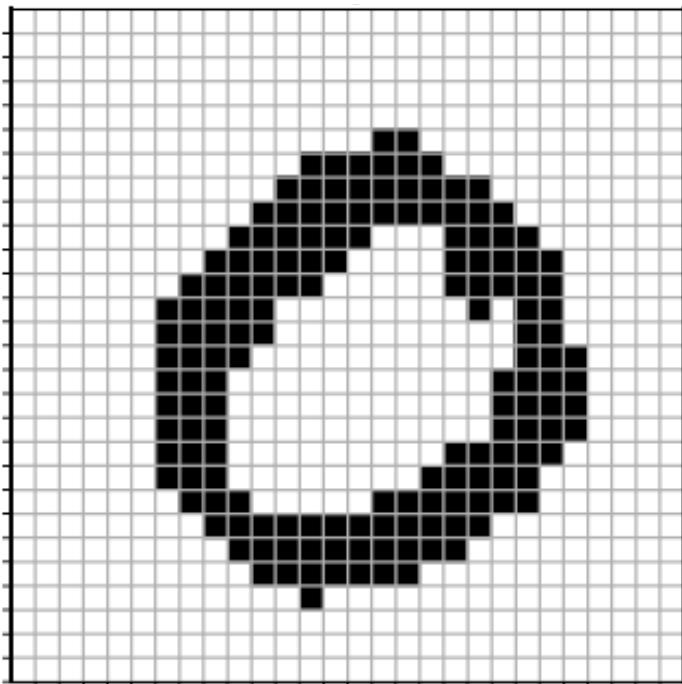
# Algoritmul de învățare al perceptronului

```
1 def perceptron(X, y, n_epochs, η):
2     m, n = X.shape # number of samples, number of inputs
3     for j in range(n):
4         wj = 0
5     for epoch in range(n_epochs): # an “epoch” is a run through all training data.
6         for i in range(m): # a “training step” is one update of the weights.
7             ŷ(i) = unit_step_function(Σj=0n wjxj(i))      ŷ(i) = {1 if Σj=0n wjxj(i) ≥ 0
8             for j in range(n):
9                 wj += η(y(i) - ŷ(i)) · xj(i)
```

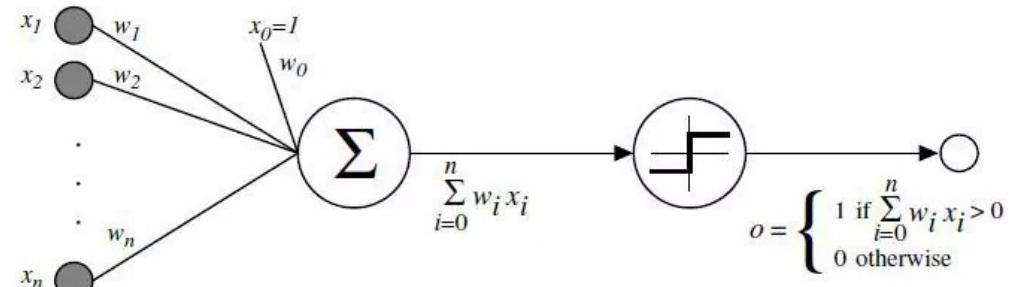
# Exemplu – Recunoașterea cifrelor

Putem antrena un perceptron care să discrimineze între imagini cu cifre scrise de mâna cu cifrele 0 și 1?

**$28 \times 28$  valori binare**



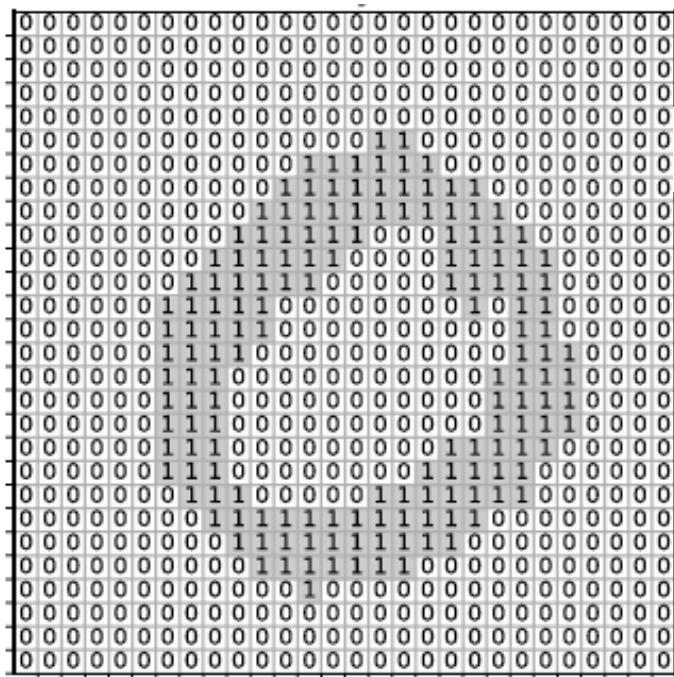
**100 de exemple pentru fiecare clasă**



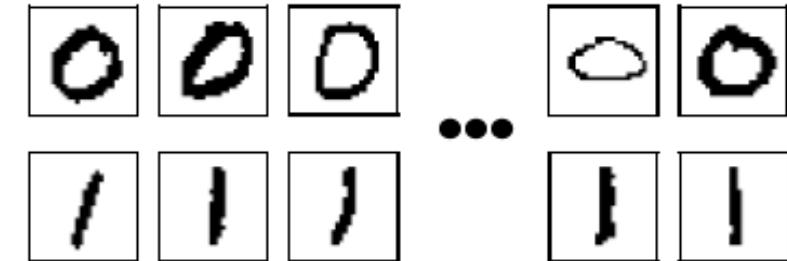
# Exemplu – Recunoașterea cifrelor

Putem antrena un perceptron care să discrimineze între imagini cu cifre scrise de mână cu cifrele 0 și 1?

**$28 \times 28$  valori binare**

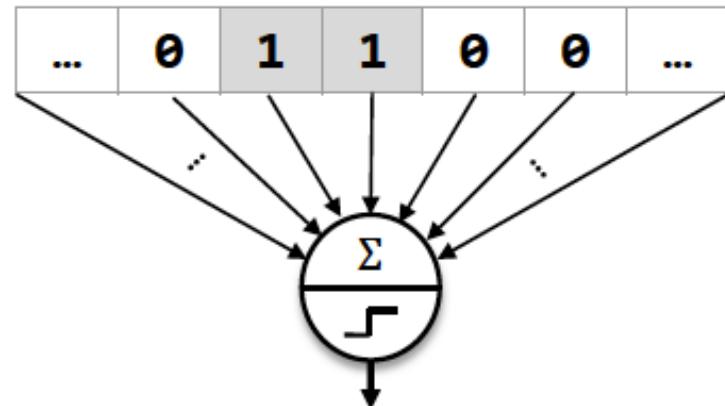


**liniarizare**



**100 de exemple pentru fiecare clasă**

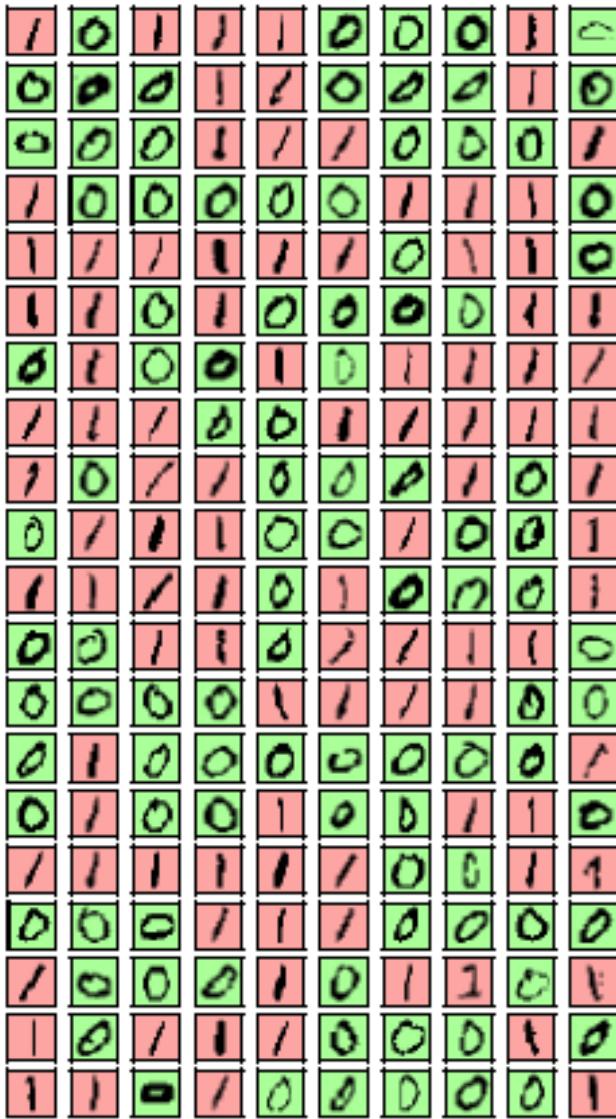
**784 de intrări**



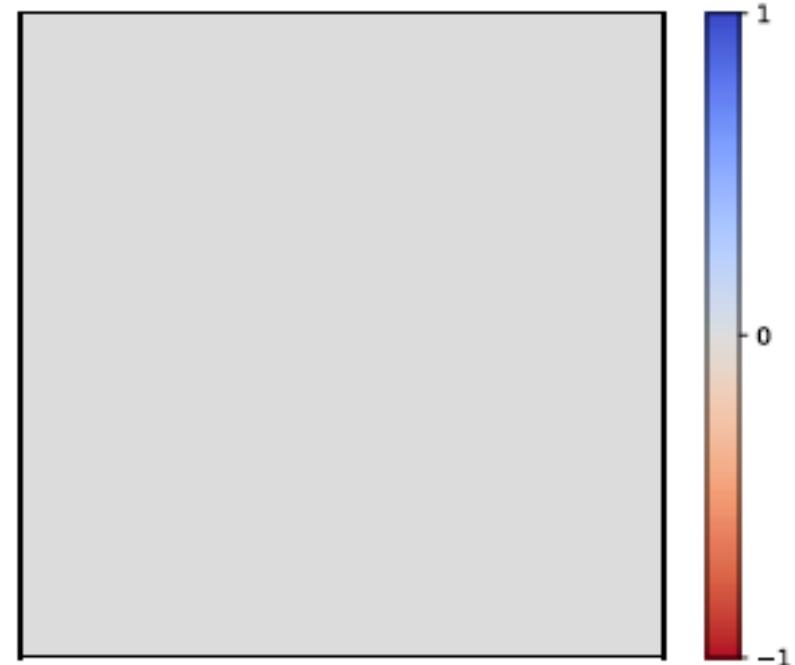
# Exemplu – Recunoașterea cifrelor

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

# Exemplu – Recunoașterea cifrelor

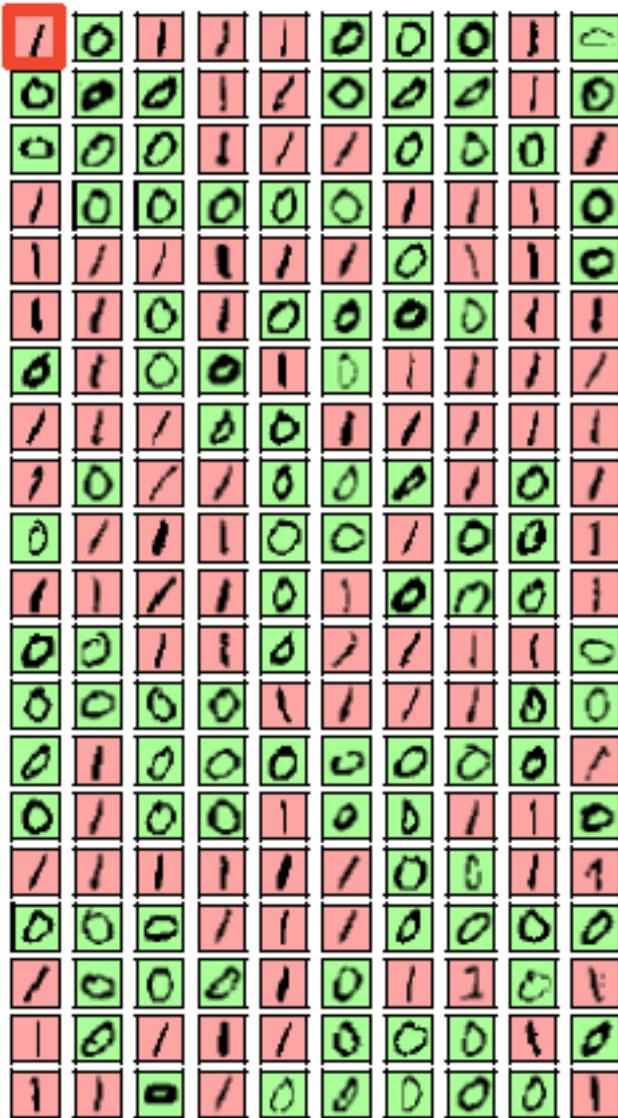


Valorile ponderilor (inițial 0)  
afișate ca o imagine  $28 \times 28$

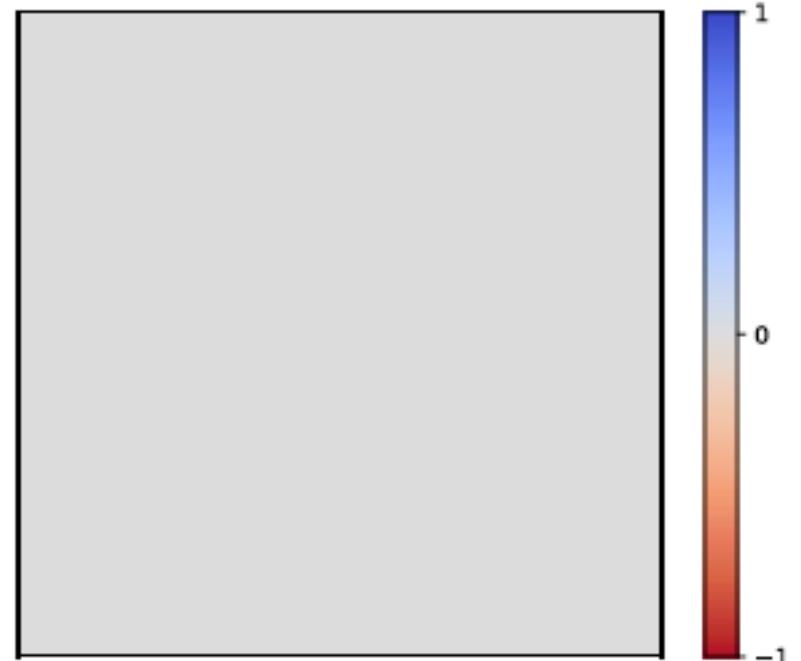


Pasul 0 de antrenare  
Acuratețe 50%

# Exemplu – Recunoașterea cifrelor



Valorile ponderilor afișate  
ca o imagine  $28 \times 28$



Pasul 1 de antrenare  
Acuratețe 50%

# Exemplu – Recunoașterea cifrelor

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

## Regula de actualizare a perceptronului

$$w_j = w_j + \Delta w_j, \text{ unde}$$

$$\Delta w_j = \eta(y^{(i)} - \hat{y}^{(i)}) \cdot x_j^{(i)}$$

Cantitatea prin care schimbăm ponderea (rata de învățare)

Setează semnul actualizării

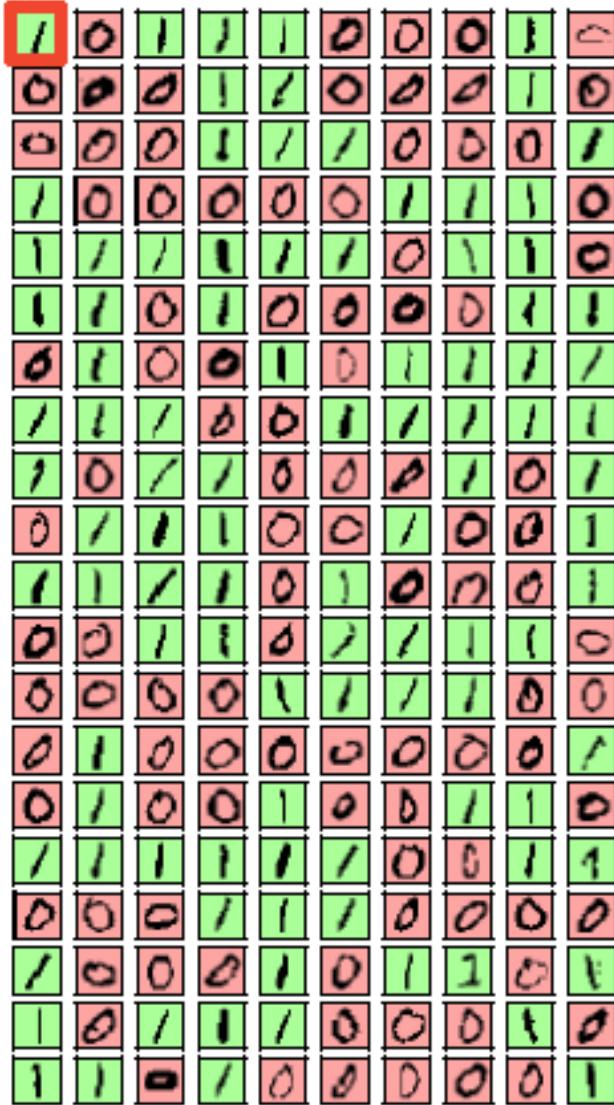
Selectează intrările active

Pentru exemple de antrenare misclasificate:

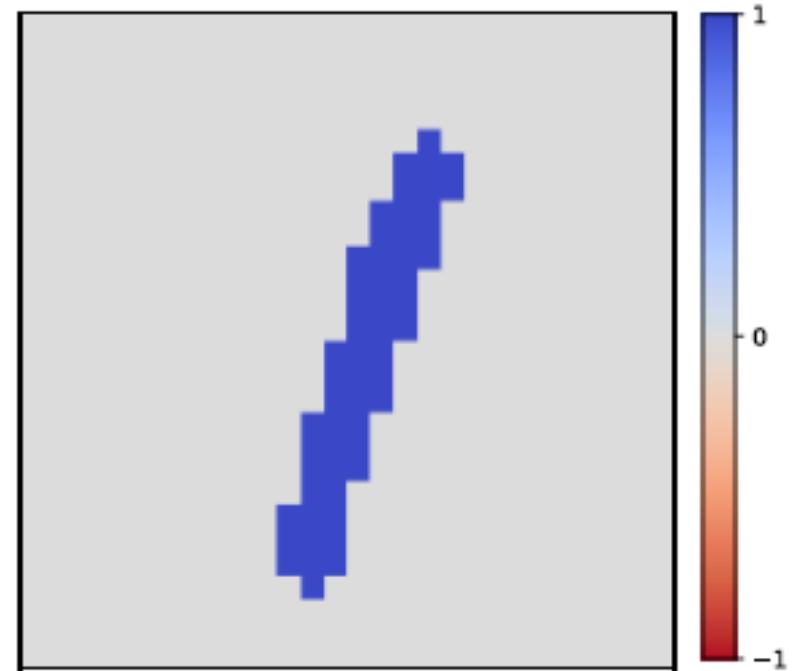
-daca  $y^{(i)}=1$  atunci adaug  $\eta*(1-0)*1$  pentru ponderile corespunzătoare pixelilor negri (cei din exemplul de antrenare curent cu cifra 1)

-daca  $y^{(i)}=0$  atunci adaug  $\eta*(0-1)*1$  pentru ponderile corespunzătoare pixelilor negri (cei din exemplul de antrenare curent cu cifra 0)

# Exemplu – Recunoașterea cifrelor



Valorile ponderilor afișate  
ca o imagine  $28 \times 28$

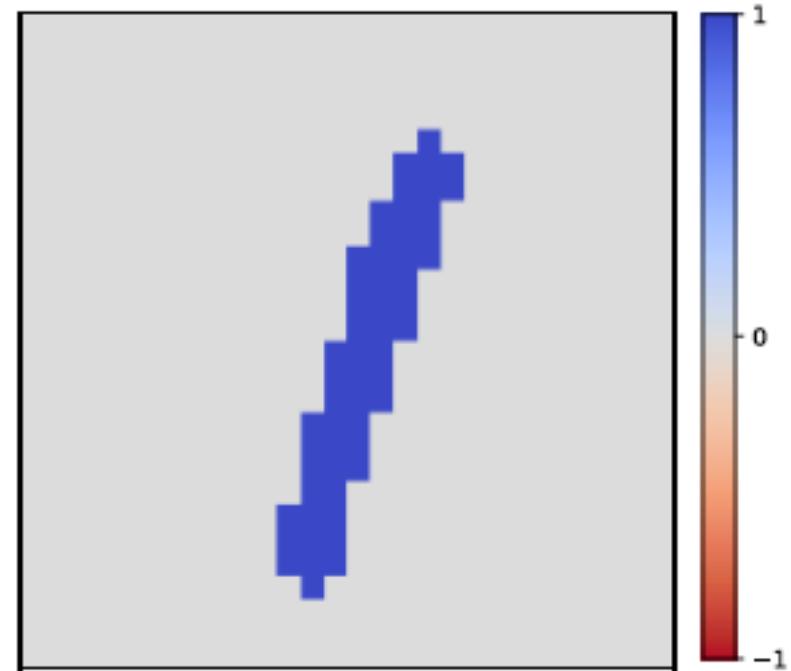


Pasul 1 de antrenare  
Acuratețe 50%

# Exemplu – Recunoașterea cifrelor

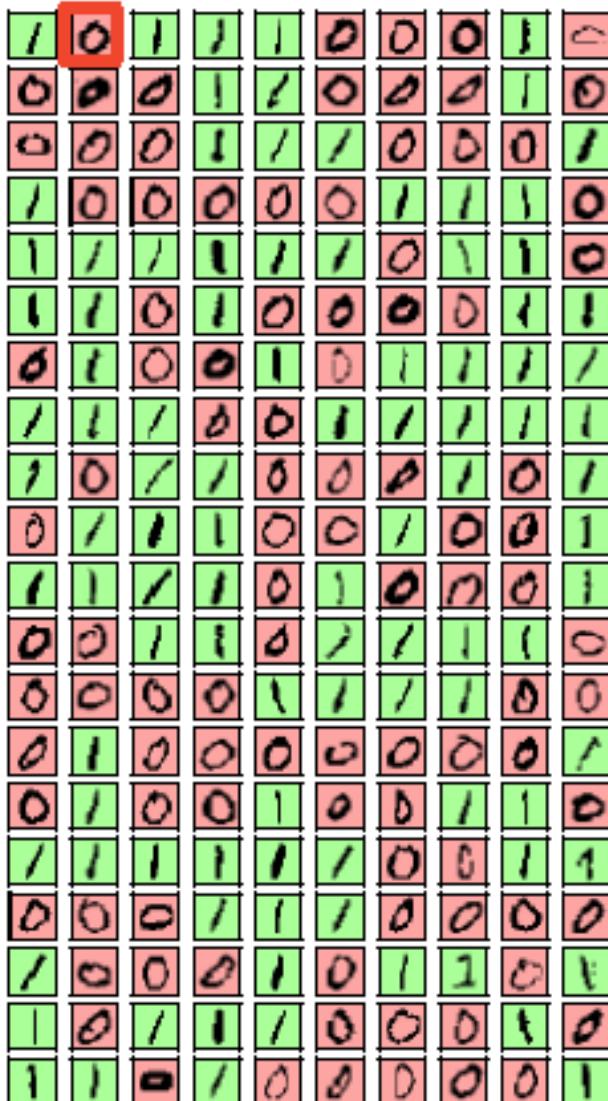
1	0	1	1	1	0	0	0	1	-1
0	0	0	1	1	0	0	0	1	0
0	0	0	1	1	1	0	0	0	1
1	0	0	0	0	0	1	1	1	0
1	1	1	1	1	1	0	1	1	0
1	1	0	1	0	0	0	1	1	1
0	1	0	0	1	0	1	1	1	1
1	1	1	0	0	1	1	1	1	1
1	0	1	1	0	0	0	1	0	1
0	1	1	1	0	0	1	0	0	1
1	1	1	1	0	1	0	0	0	1
0	0	1	1	0	1	1	1	0	0
0	1	0	0	0	0	0	0	0	1
0	1	0	0	1	0	0	1	1	0
1	1	1	1	1	1	0	0	1	1
0	0	1	1	1	0	0	0	0	0
1	0	0	0	1	0	1	1	0	1
1	0	1	1	0	0	0	1	0	0
1	1	0	1	0	0	0	0	0	1

Valorile ponderilor afișate ca o imagine  $28 \times 28$



Pasul 2 de antrenare  
Acuratețe 50%

# Exemplu – Recunoașterea cifrelor



## Regula de actualizare a perceptronului

$$w_j = w_j + \Delta w_j, \text{ unde}$$

$$\Delta w_j = \eta(y^{(i)} - \hat{y}^{(i)}) \cdot x_j^{(i)}$$

Cantitatea prin care  
schimbăm ponderea  
(rata de învățare)

Setează semnul actualizării

Selectează intrările active

Pentru exemple de antrenare misclasificate:

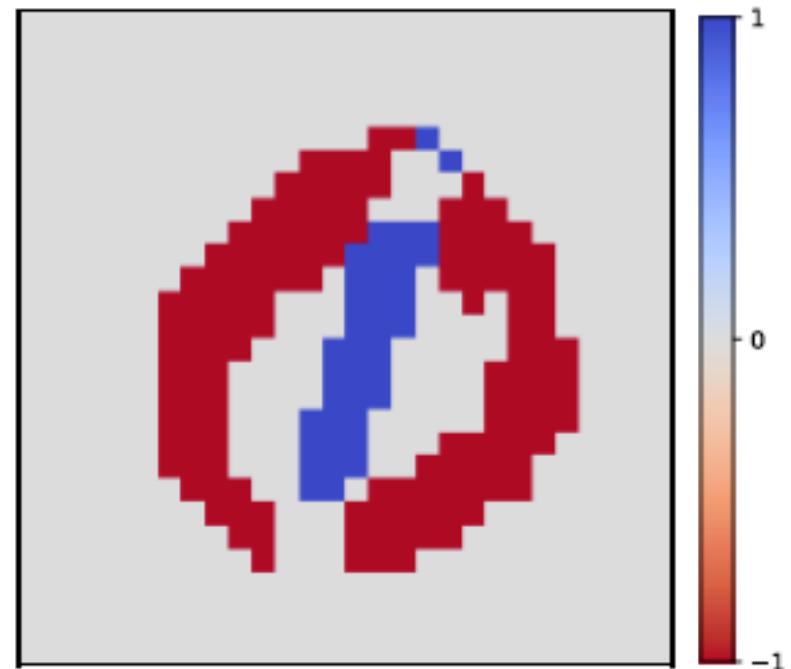
-daca  $y^{(i)}=1$  atunci adaug  $\eta*(1-0)*1$  pentru ponderile corespunzatoare pixelilor negri (cei din exemplul de antrenare curent cu cifra 1)

-daca  $y^{(i)}=0$  atunci adaug  $\eta*(0-1)*1$  pentru ponderile corespunzatoare pixelilor negri (cei din exemplul de antrenare curent cu cifra 0)

# Exemplu – Recunoașterea cifrelor

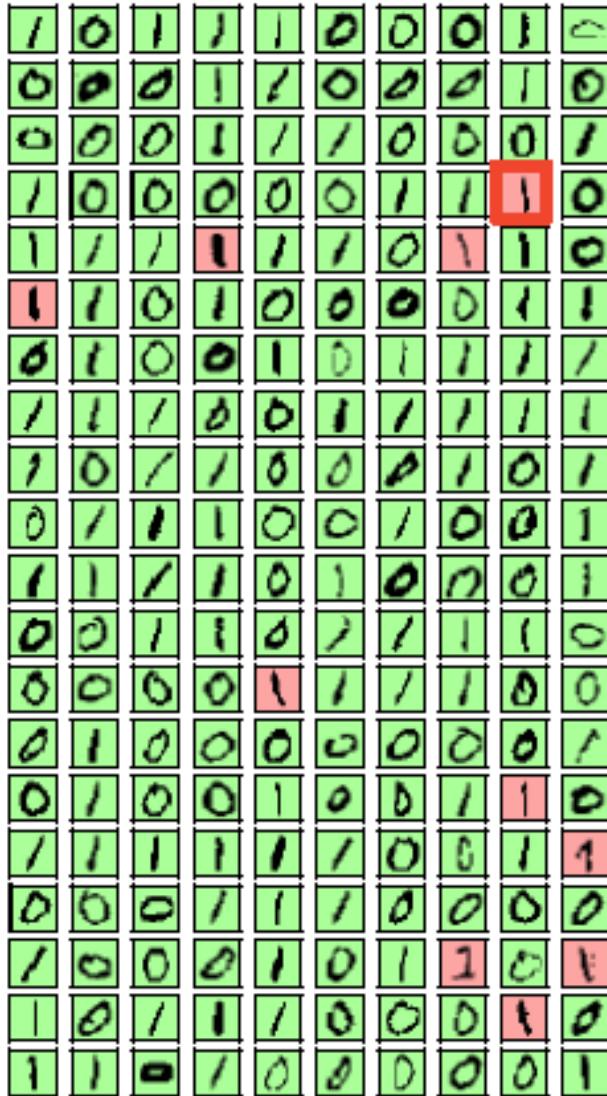
1	0	1	1	1	0	0	0	1	0
0	0	0	1	1	0	0	0	1	0
0	0	0	1	1	1	0	0	0	1
1	0	0	0	0	0	1	1	1	0
1	1	1	1	1	0	1	1	0	0
1	1	0	1	0	0	0	1	1	1
0	1	0	0	1	0	1	1	1	1
1	1	1	0	0	1	1	1	1	1
1	0	1	1	0	0	0	1	0	1
0	1	1	1	0	0	1	0	0	1
1	1	1	1	0	1	0	0	1	0
0	0	1	1	0	1	1	1	1	0
0	0	0	0	1	1	1	1	0	0
0	1	0	0	0	0	0	0	1	0
0	1	0	0	1	0	0	1	1	0
1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

Valorile ponderilor afișate ca o imagine  $28 \times 28$

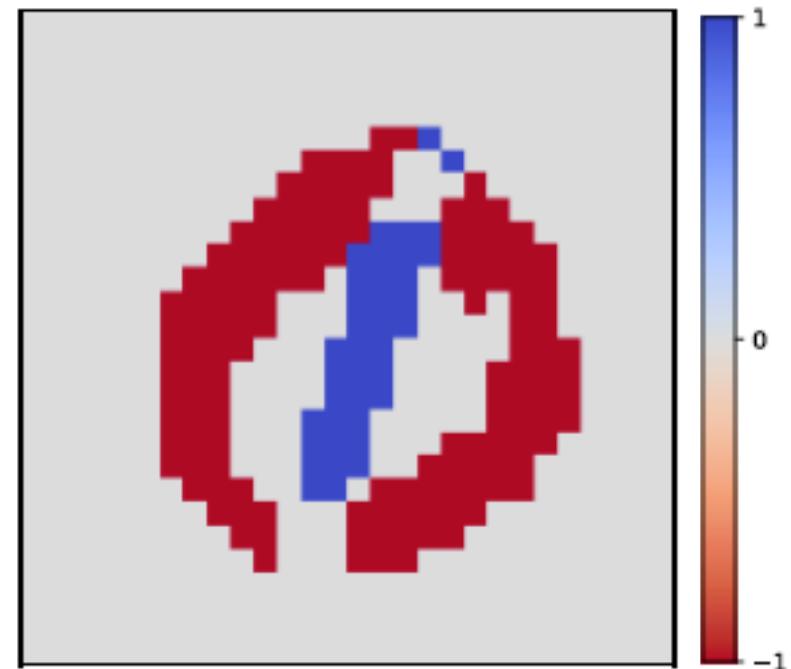


Pasul 2 de antrenare  
Acuratețe 95%

# Exemplu – Recunoașterea cifrelor

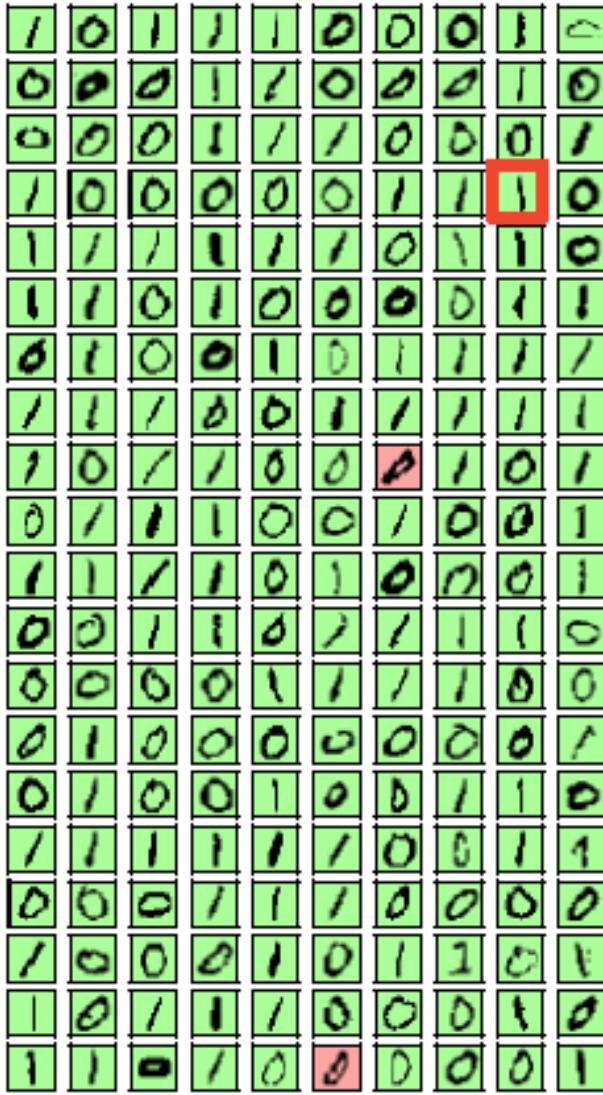


Valorile ponderilor afișate ca o imagine  $28 \times 28$

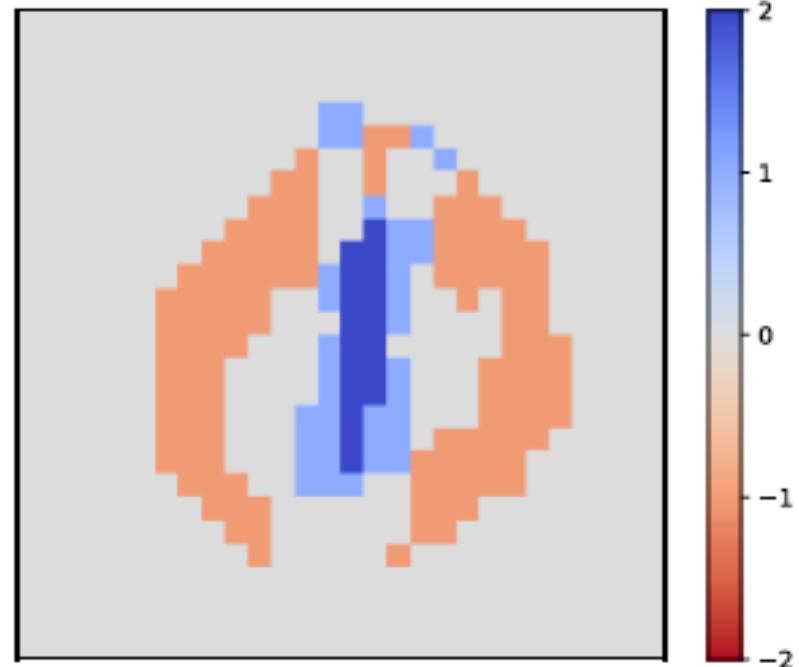


Pasul 39 de antrenare  
Acuratețe 95%

# Exemplu – Recunoașterea cifrelor

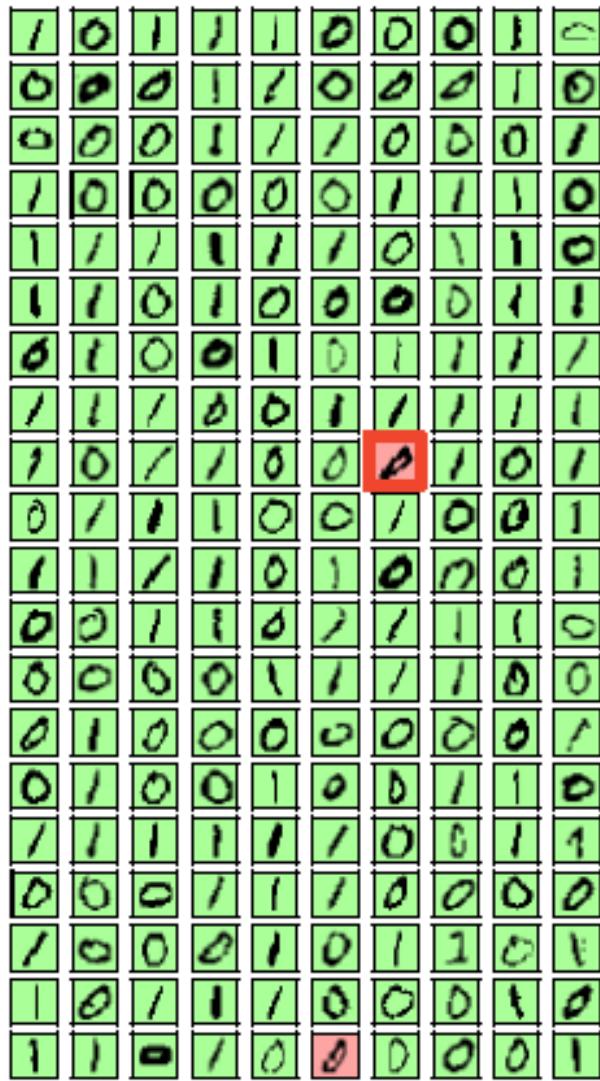


Valorile ponderilor afișate ca o imagine  $28 \times 28$

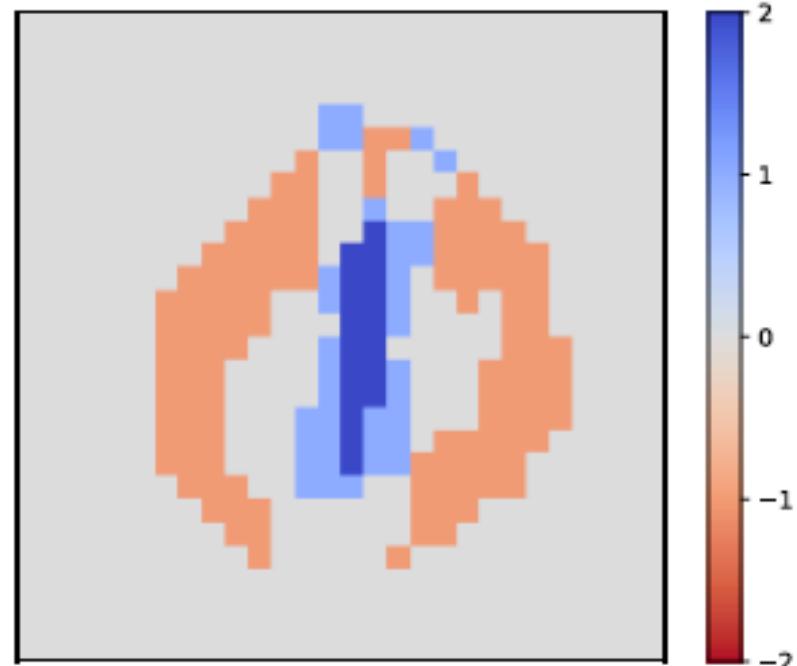


Pasul 39 de antrenare  
Acuratețe 99%

# Exemplu – Recunoașterea cifrelor



Valorile ponderilor afișate ca o imagine  $28 \times 28$

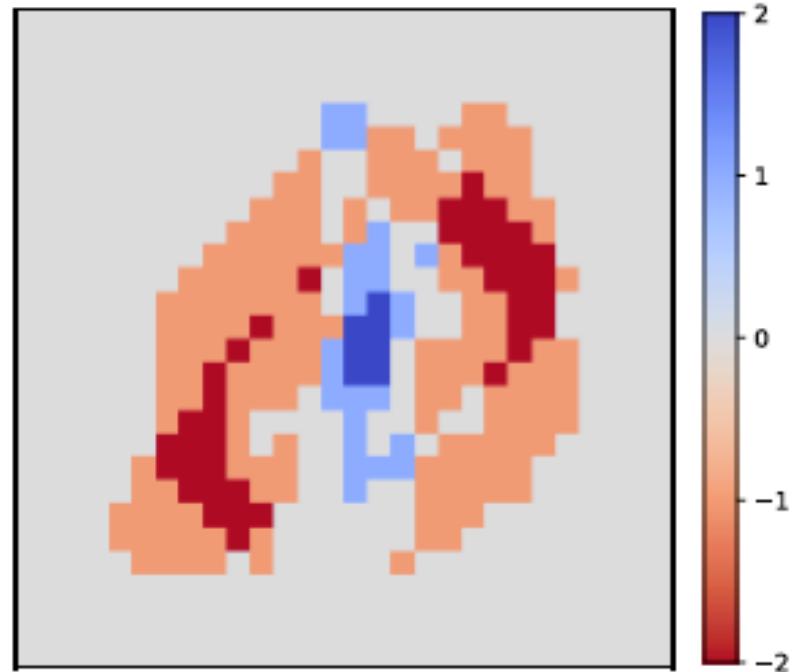


Pasul 87 de antrenare  
Acuratețe 99%

# Exemplu – Recunoașterea cifrelor

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

Valorile ponderilor afișate ca o imagine  $28 \times 28$

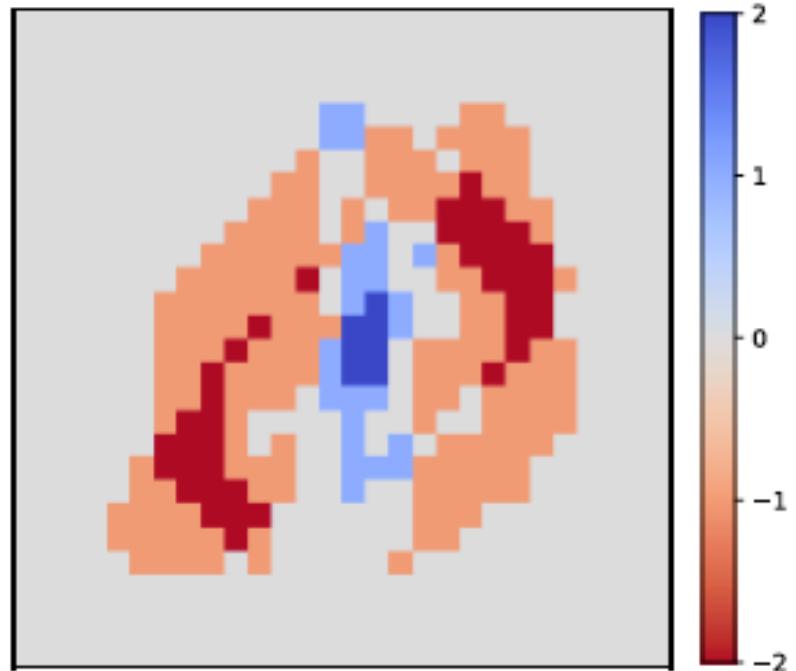


Pasul 87 de antrenare  
Acuratețe 88%

# Exemplu – Recunoașterea cifrelor

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

Valorile ponderilor afișate ca o imagine  $28 \times 28$

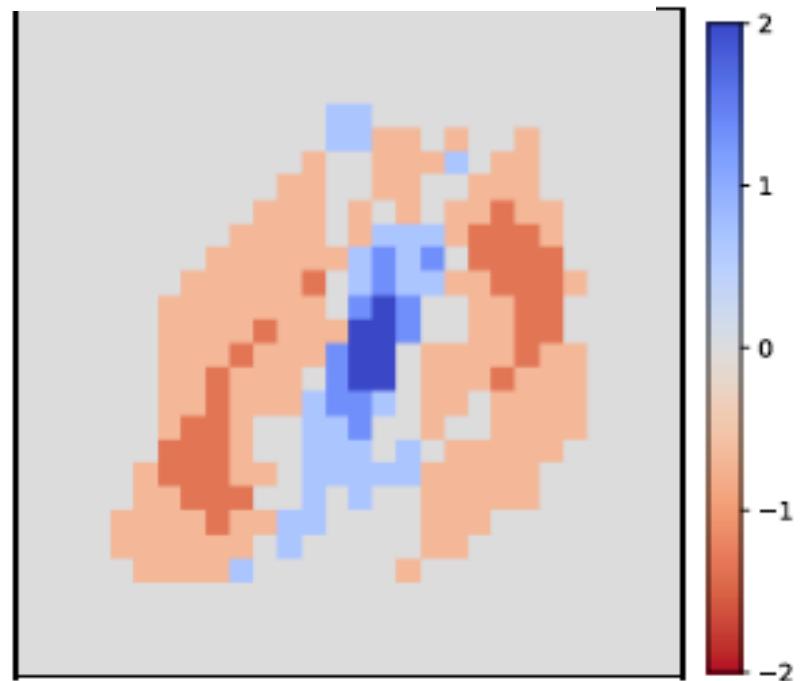


Pasul 92 de antrenare  
Acuratețe 88%

# Exemplu – Recunoașterea cifrelor

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

Valorile ponderilor afișate ca o imagine  $28 \times 28$

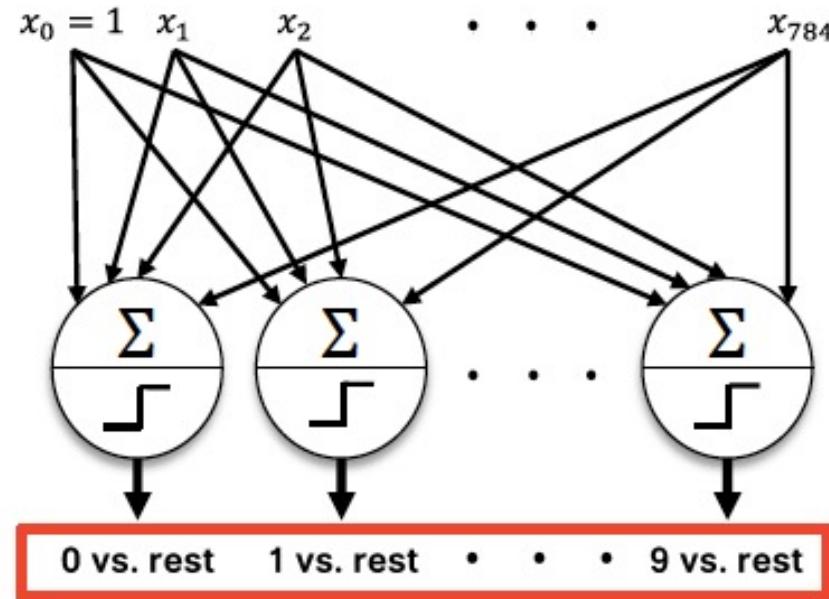


Pasul 92 de antrenare  
Acuratețe 100%

# Exemplu – Recunoașterea cifrelor

100 samples of each digit

8	0	3	6	1	0	0	4	1	7
3	2	4	9	5	1	1	9	2	8
1	5	8	1	2	0	5	8	3	0
3	5	4	0	6	9	7	5	5	2
2	7	0	0	8	8	6	1	8	4
9	8	6	1	3	3	3	2	6	4
6	5	7	7	8	7	3	3	6	5
4	6	7	7	4	4	6	2	3	6
5	5	5	4	3	1	5	5	1	2
2	3	9	7	5	0	6	2	4	2
4	6	7	4	5	0	8	7	3	2
0	2	9	4	7	1	6	3	2	1
6	9	8	7	9	2	6	3	8	8
3	4	4	9	6	5	1	9	3	7
5	8	4	0	5	3	8	3	1	1
8	2	0	3	5	5	6	7	4	1
3	7	6	1	7	9	6	2	5	1
8	6	1	7	9	5	2	2	5	4
7	9	5	0	3	1	4	5	2	4
5	2	4	8	8	5	8	✓	8	1



reprezentare one-hot  
(o componentă 1, restul 0)

O rețea poate prezice clase multiple folosind un neuron separat pentru fiecare clasă  
Pe problema dată atinge ușor acuratețe de ~99%

# Perceptronul în IA

Extrase din ziarul The New York Times, 8 iulie, 1958

## NEW NAVY DEVICE LEARNS BY DOING

Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

WASHINGTON, July 7 (UPI)—The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of \$100,000.

Dr. Frank Rosenblatt, designer of the Perceptron, conducted the demonstration. He said the machine would be the first device to think as the human brain. As do human beings, Perceptron will make mistakes at first, but will grow wiser as it gains experience, he said.

The Navy said the perceptron would be the first non-living mechanism "capable of receiving, recognizing and identifying its surroundings without any human training or control."

The "brain" is designed to remember images and information it has perceived itself. Ordinary computers remember only what is fed into them on punch cards or magnetic tape.

Later Perceptrons will be able to recognize people and call out their names and instantly translate speech in one language to speech or writing in another language, it was predicted.

- <http://jcblackmon.com/wp-content/uploads/2018/01/MBC-Rosenblatt-Perceptron-NYT-article.jpg.pdf>

# Limitările perceptronului

- Perceptronul învață funcțiile booleene AND/OR și reușește să rezolve probleme de recunoaștere a cifrelor. Probleme simple sau grele?
- Considerăm funcția XOR (exclusiv OR):

$x_1$	$x_2$	$y$
0	0	0
0	1	1
1	0	1
1	1	0

Pentru ca un Perceptron să învețe funcția XOR e nevoie ca:

$$w_0 + 0 \cdot w_1 + 0 \cdot w_2 < 0$$

$$w_0 + 0 \cdot w_1 + 1 \cdot w_2 \geq 0$$

$$w_0 + 1 \cdot w_1 + 0 \cdot w_2 \geq 0$$

$$w_0 + 1 \cdot w_1 + 1 \cdot w_2 < 0$$

# Limitările perceptronului

- Perceptronul învață funcțiile booleene AND/OR și reușește să rezolve probleme de recunoaștere a cifrelor. Probleme simple sau grele?
- Considerăm funcția XOR (exclusiv OR):

$x_1$	$x_2$	$y$
0	0	0
0	1	1
1	0	1
1	1	0

Pentru ca un Perceptron să învețe funcția XOR e nevoie ca:

$$w_0 + 0 \cdot w_1 + 0 \cdot w_2 < 0 \Rightarrow w_0 < 0$$

$$w_0 + 0 \cdot w_1 + 1 \cdot w_2 \geq 0 \Rightarrow w_0 \geq -w_2$$

$$w_0 + 1 \cdot w_1 + 0 \cdot w_2 \geq 0 \Rightarrow w_0 \geq -w_1$$

$$w_0 + 1 \cdot w_1 + 1 \cdot w_2 < 0 \Rightarrow w_0 < -w_1 - w_2$$

# Limitările perceptronului

- Perceptronul învață funcțiile booleene AND/OR și reușește să rezolve probleme de recunoaștere a cifrelor. Probleme simple sau grele?
- Considerăm funcția XOR (exclusiv OR):

$x_1$	$x_2$	$y$
0	0	0
0	1	1
1	0	1
1	1	0

Pentru ca un Perceptron să învețe funcția XOR e nevoie ca:

$$w_0 + 0 \cdot w_1 + 0 \cdot w_2 < 0 \Rightarrow w_0 < 0$$

$$w_0 + 0 \cdot w_1 + 1 \cdot w_2 \geq 0 \Rightarrow w_0 \geq -w_2$$

$$w_0 + 1 \cdot w_1 + 0 \cdot w_2 \geq 0 \Rightarrow w_0 \geq -w_1$$

$$w_0 + 1 \cdot w_1 + 1 \cdot w_2 < 0 \Rightarrow w_0 < -w_1 - w_2$$

$$\left. \begin{array}{l} 2w_0 \geq -w_1 - w_2 \\ 2w_0 > w_0 \end{array} \right\} \Rightarrow 2w_0 > w_0 \Rightarrow w_0 > 0$$

# Limitările perceptronului

- Perceptronul învață funcțiile booleene AND/OR și reușește să rezolve probleme de recunoaștere a cifrelor. Probleme simple sau grele?
- Considerăm funcția XOR (exclusiv OR):

$x_1$	$x_2$	$y$
0	0	0
0	1	1
1	0	1
1	1	0

Pentru ca un Perceptron să învețe funcția XOR e nevoie ca:

$$w_0 + 0 \cdot w_1 + 0 \cdot w_2 < 0 \Rightarrow w_0 < 0$$

$$w_0 + 0 \cdot w_1 + 1 \cdot w_2 \geq 0 \Rightarrow w_0 \geq -w_2$$

$$w_0 + 1 \cdot w_1 + 0 \cdot w_2 \geq 0 \Rightarrow w_0 \geq -w_1$$

$$w_0 + 1 \cdot w_1 + 1 \cdot w_2 < 0 \Rightarrow w_0 < -w_1 - w_2$$

$$\left. \begin{array}{l} 2w_0 \geq -w_1 - w_2 \\ 2w_0 > w_0 \end{array} \right\} \Rightarrow 2w_0 > w_0 \Rightarrow w_0 > 0$$



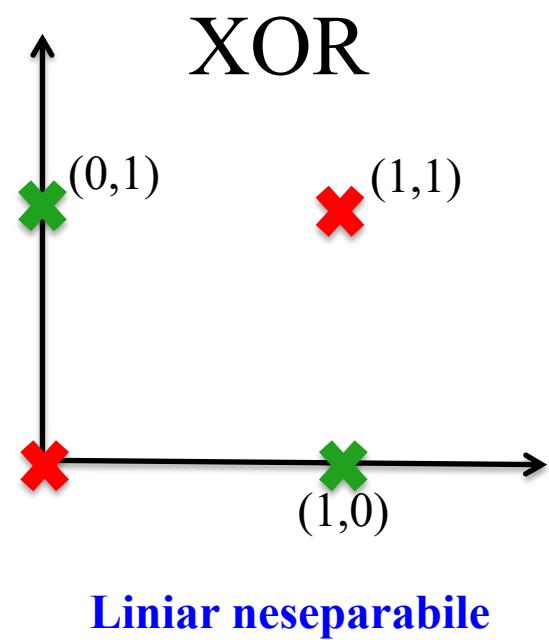
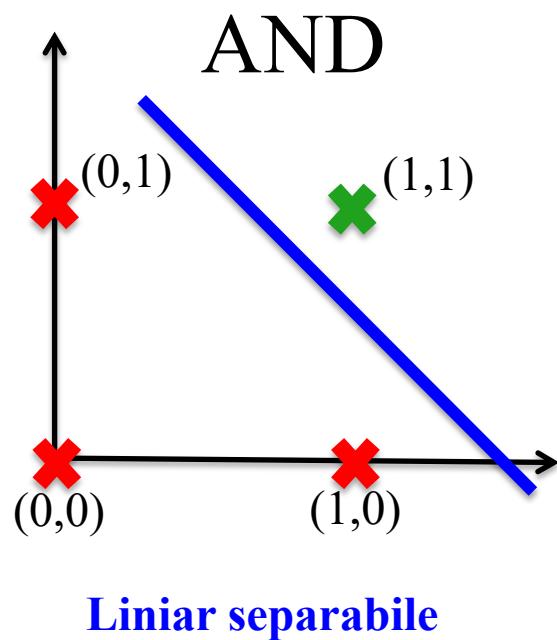
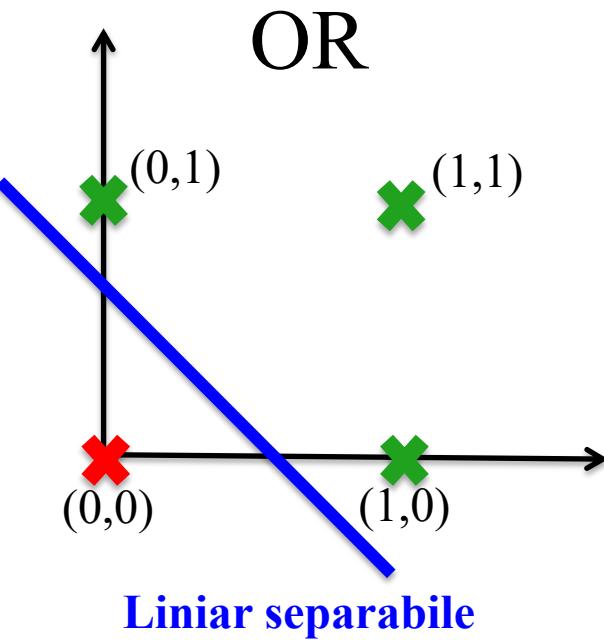
**Contradicție**

- Perceptronul nu poate învăța XOR oricât de mulți pași ar face
- Mai mult, Perceptronul poate învăță numai clase care sunt liniar separabile

# Teorema de convergență a perceptronului

- Dacă mulțimea de antrenare  $E$  este liniar separabilă cu margine  $\gamma$ , algoritmul de învățare a perceptronului este garantat că va converge într-un număr finit de pași către o soluție în care nu se fac greșeli pe mulțimea de antrenare
- Numărul de pași  $k$  satisfac relația  $k \leq \frac{R^2}{\gamma^2}$ , unde  $R$  este raza sferei din spațiul caracteristicilor care cuprinde toate exemplele de antrenare
- Teorema spune că va converge către o soluție, nu este necesar să fie o soluție bună (SVM oferă soluția de margine maximă). Dacă mulțimea de antrenare  $E$  nu este liniar separabilă algoritmul nu va converge către o soluție.

# Perspectiva geometrică



✗ Eticheta 0

✖ Eticheta 1

# Rețele feedforward multistrat de perceptroni

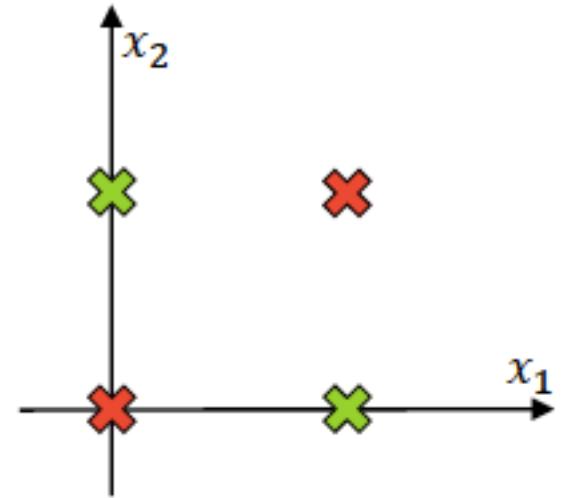
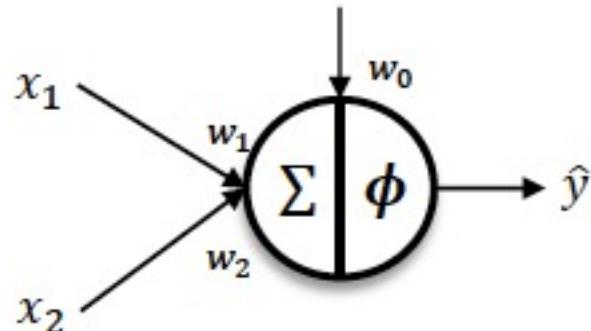
## (Multi-layer perceptrons = MLPs)

# Funcția XOR

- Un singur perceptron nu poate învăța funcția XOR încărcat nu este liniar separabilă

XOR

$x_1$	$x_2$	$y$
0	0	0
0	1	1
1	0	1
1	1	0



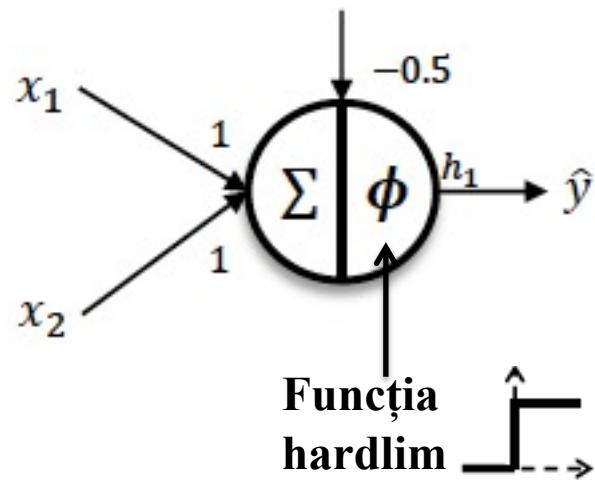
✗ Eticheta 0

✗ Eticheta 1

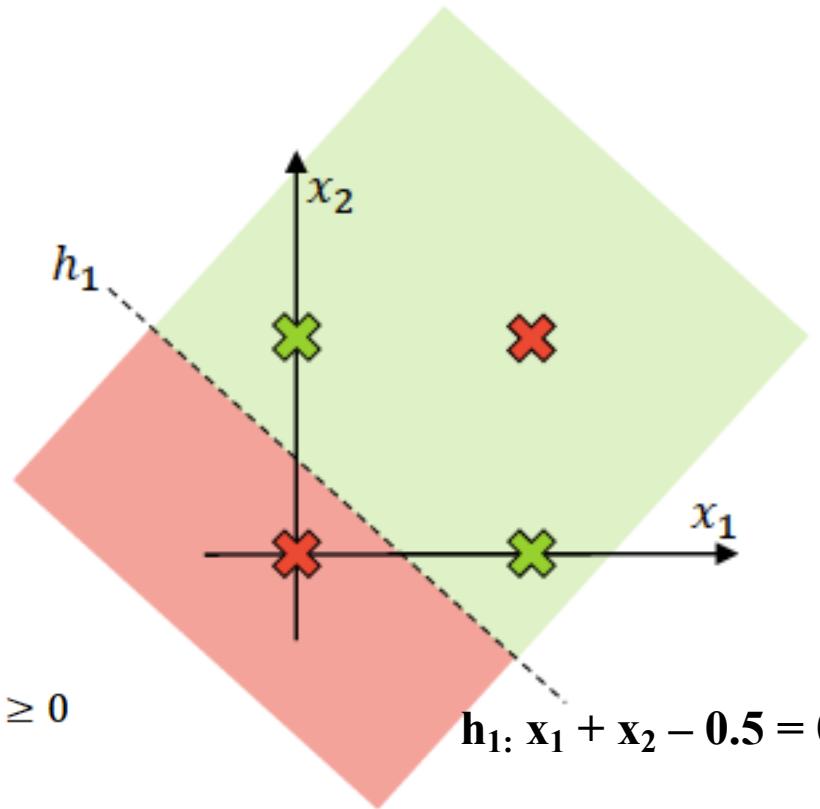
# Funcția XOR

- Un singur perceptron nu poate învăța funcția XOR încărcăt nu este liniar separabilă
- Putem depăși această limitare combinând ieșirile mai multor perceptri

XOR		
$x_1$	$x_2$	$y$
0	0	0
0	1	1
1	0	1
1	1	0



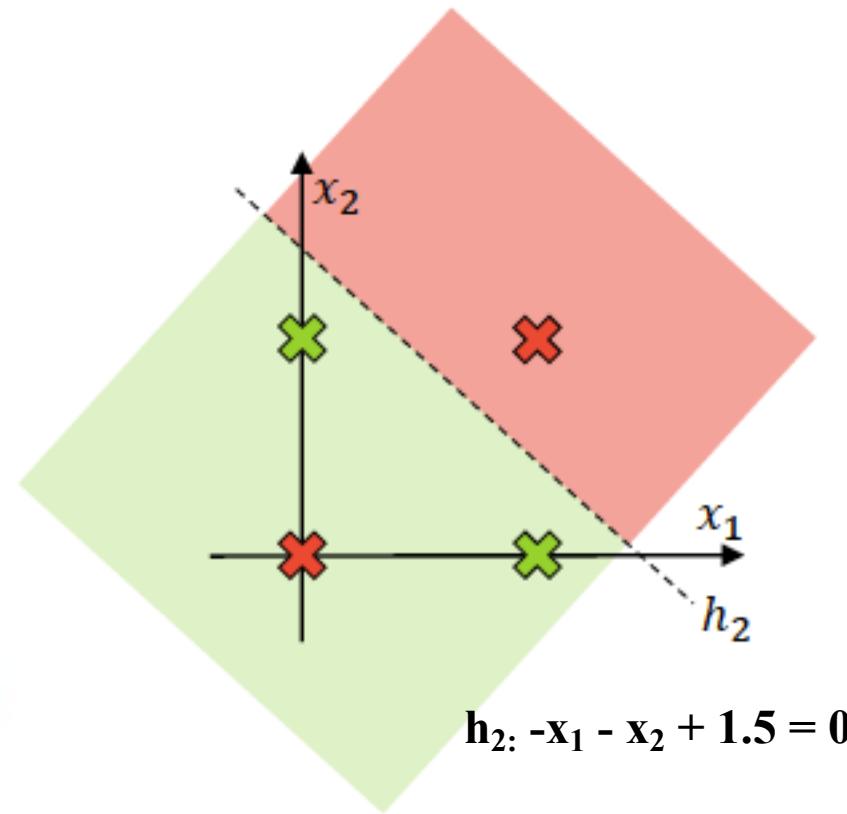
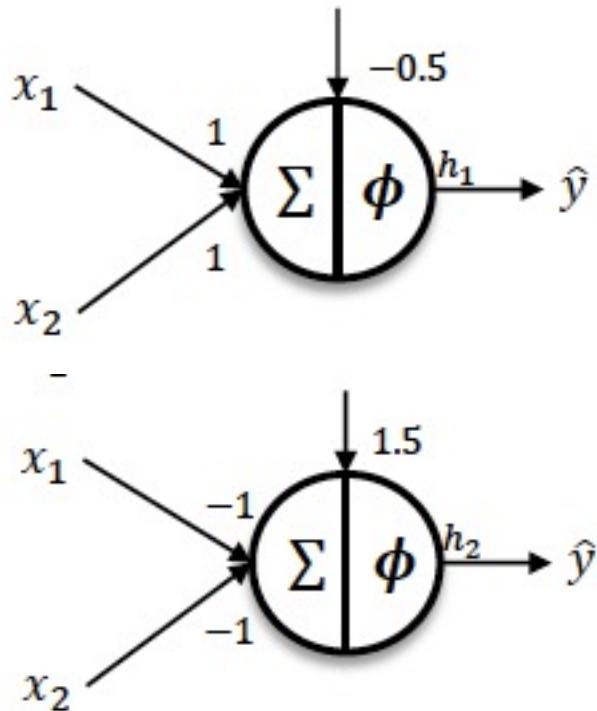
$$\hat{y} = \begin{cases} 1 & \text{if } \sum_{i=0}^n w_i x_i \geq 0 \\ 0 & \text{otherwise} \end{cases}$$



# Funcția XOR

- Un singur perceptron nu poate învăța funcția XOR încărcat nu este liniar separabilă
- Putem depăși această limitare combinând ieșirile mai multor perceptri

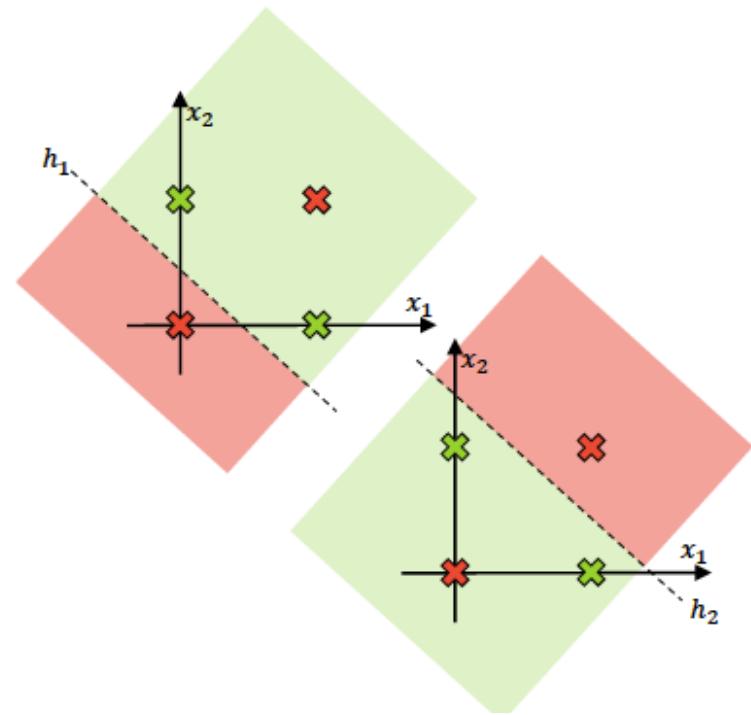
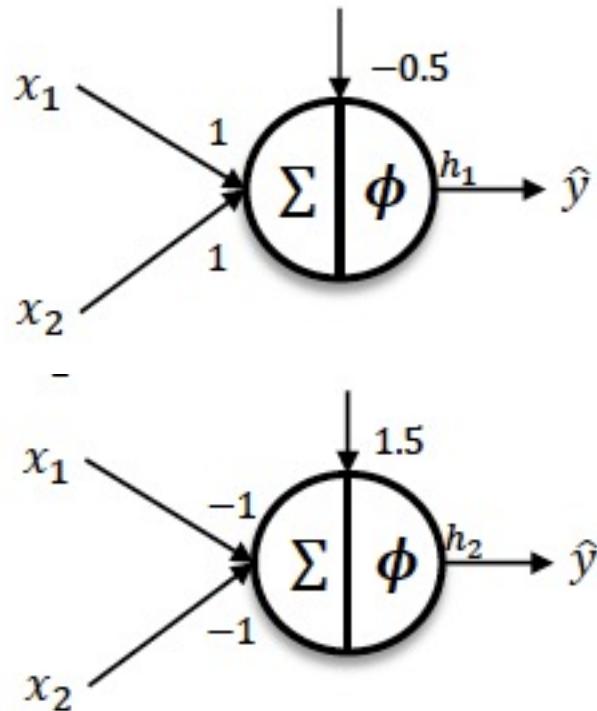
XOR		
$x_1$	$x_2$	$y$
0	0	0
0	1	1
1	0	1
1	1	0



# Funcția XOR

- Un singur perceptron nu poate învăța funcția XOR încărcat nu este liniar separabilă
- Putem depăși această limitare combinând ieșirile mai multor perceptri

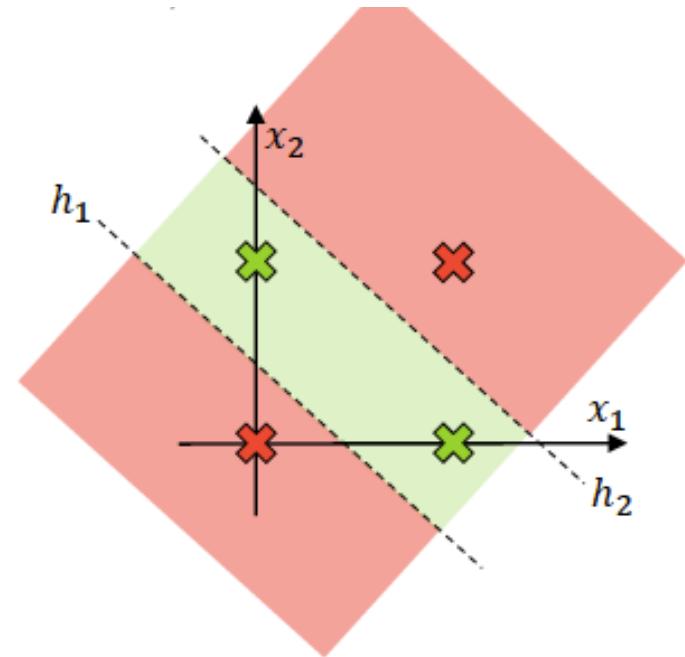
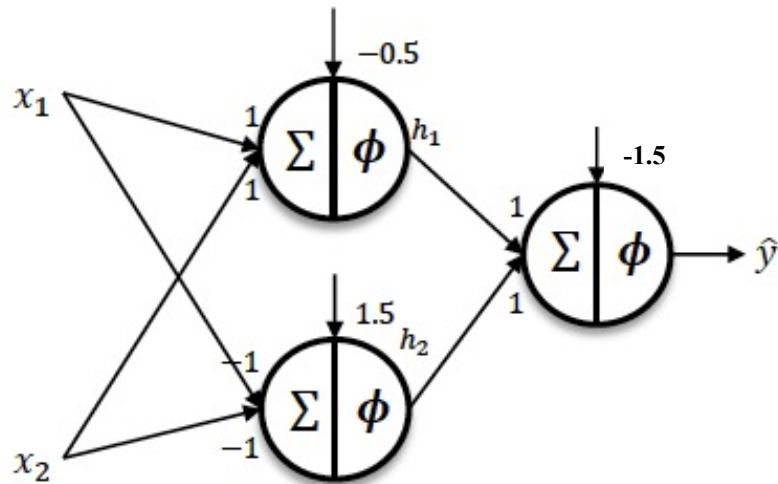
XOR		
$x_1$	$x_2$	$y$
0	0	0
0	1	1
1	0	1
1	1	0



# Funcția XOR

- Un singur perceptron nu poate învăța funcția XOR încărcat nu este liniar separabilă
- Putem depăși această limitare combinând ieșirile mai multor perceptri formând o rețea de perceptri

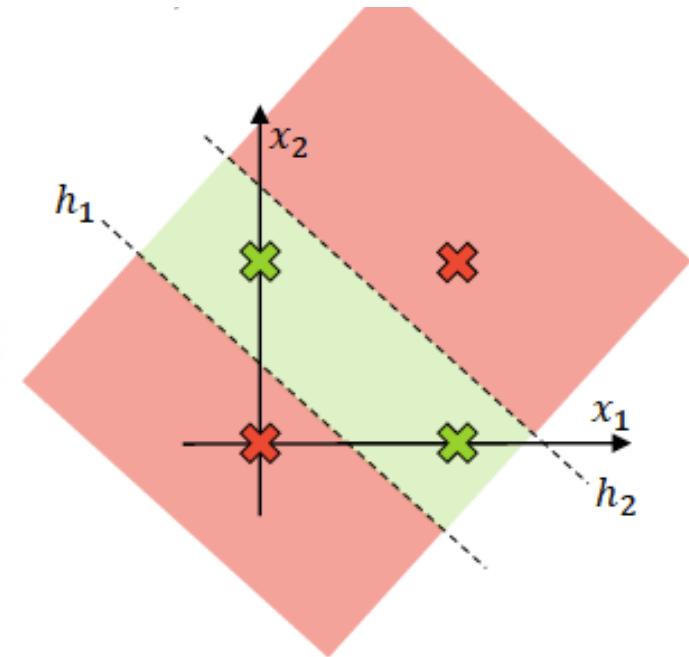
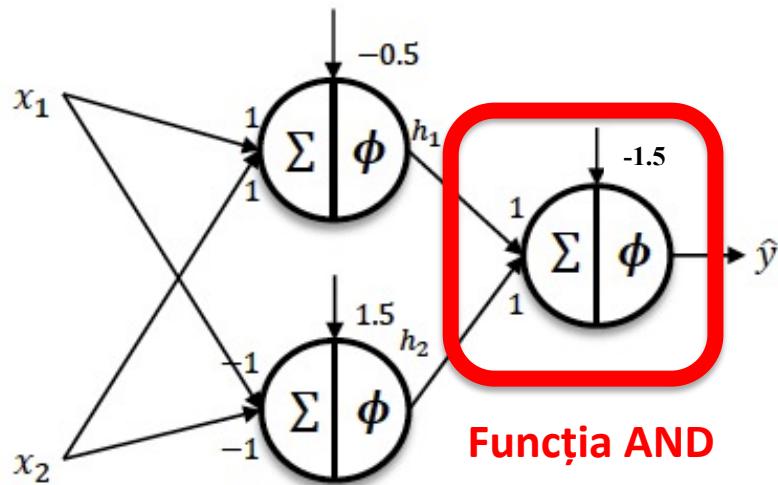
XOR		
$x_1$	$x_2$	$y$
0	0	0
0	1	1
1	0	1
1	1	0



# Funcția XOR

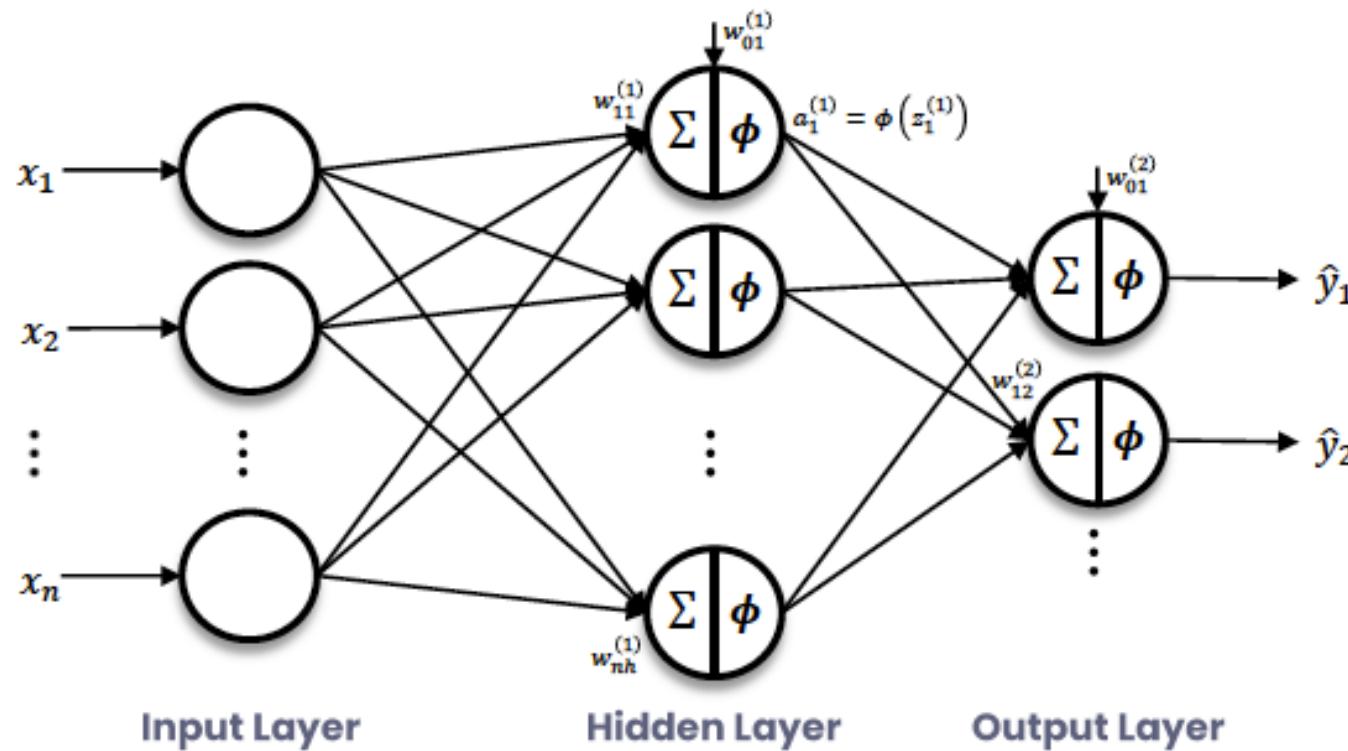
- Un singur perceptron nu poate învăța funcția XOR încărcat nu este liniar separabilă
- Putem depăși această limitare combinând ieșirile mai multor perceptri formând o rețea de perceptri

XOR		
$x_1$	$x_2$	$y$
0	0	0
0	1	1
1	0	1
1	1	0



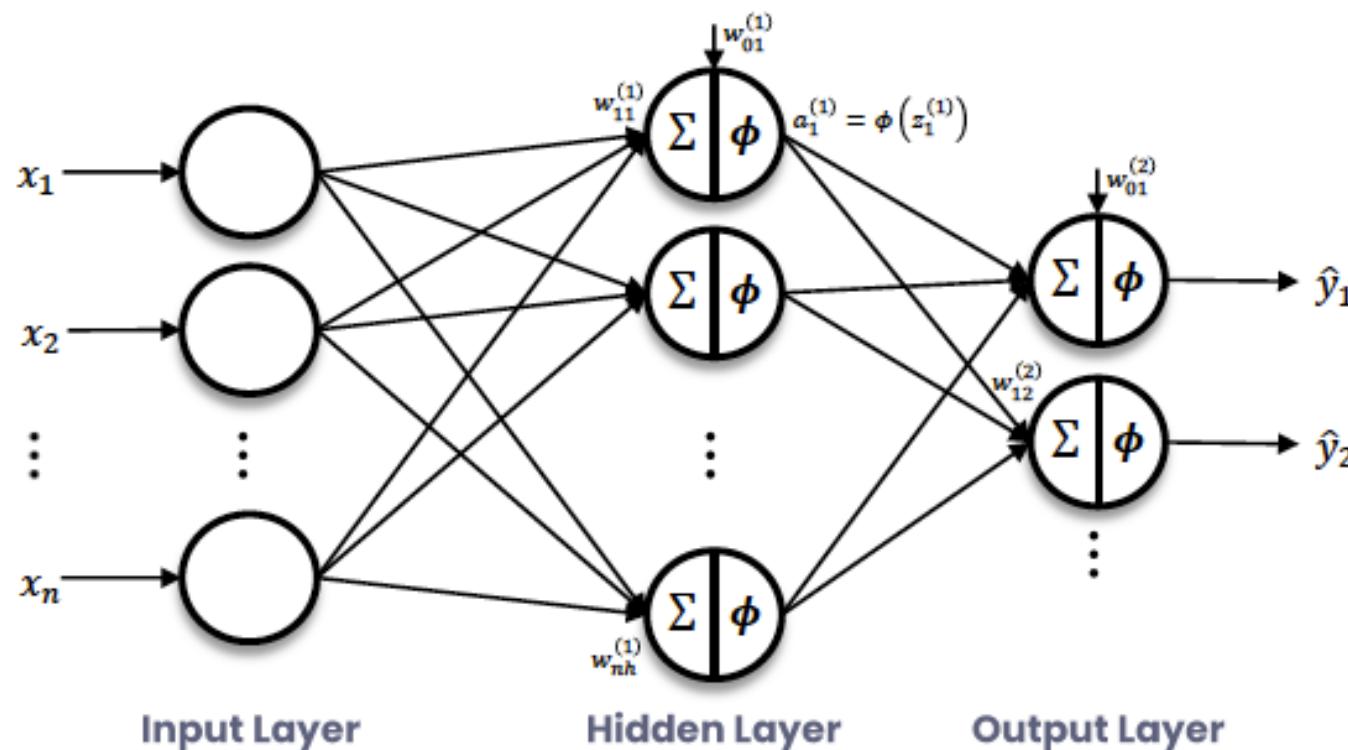
# Rețele feedforward multistrat de neuroni

- O rețea feedforward multistrat de neuroni (perceptroni) este o rețea de neuroni grupați pe straturi (layere), în care propagarea informației se realizează numai dinspre intrare spre ieșire (de la stânga la dreapta). Rețeaua are un strat de intrare (input layer), unul sau mai multe straturi ascunse (hidden layers) și un strat de ieșire (output layer).



# Rețele feedforward multistrat de neuroni

- Toți neuronii din rețea, cu excepția celor din stratul de intrare aplică o funcție de activare sumei ponderate ale intrărilor.
- Fiecare pereche de neuroni din două straturi consecutive are o pondere asociată



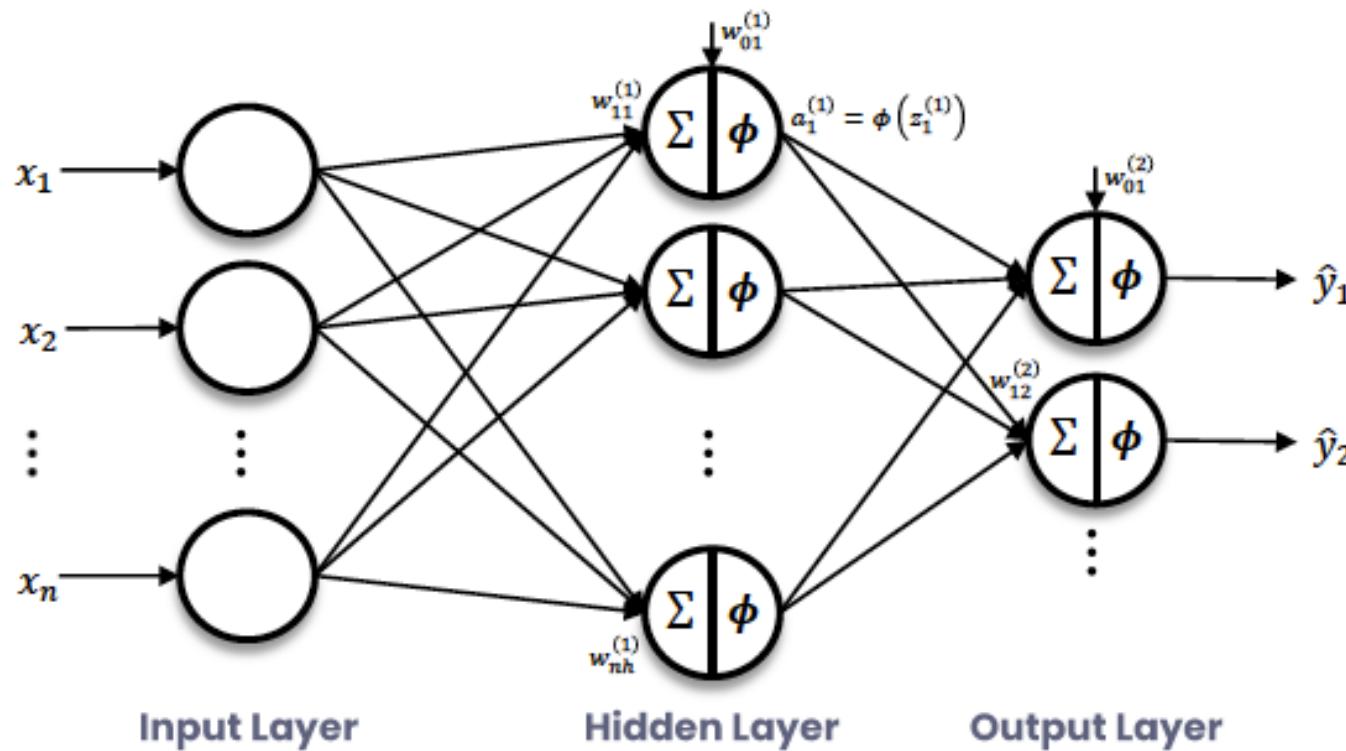
# Rețele feedforward multistrat de neuroni

$w_{ij}^{(l)}$  este ponderea neuronului  $i$  de pe stratul  $l - 1$  către neuronul  $j$  pe stratul  $l$ .

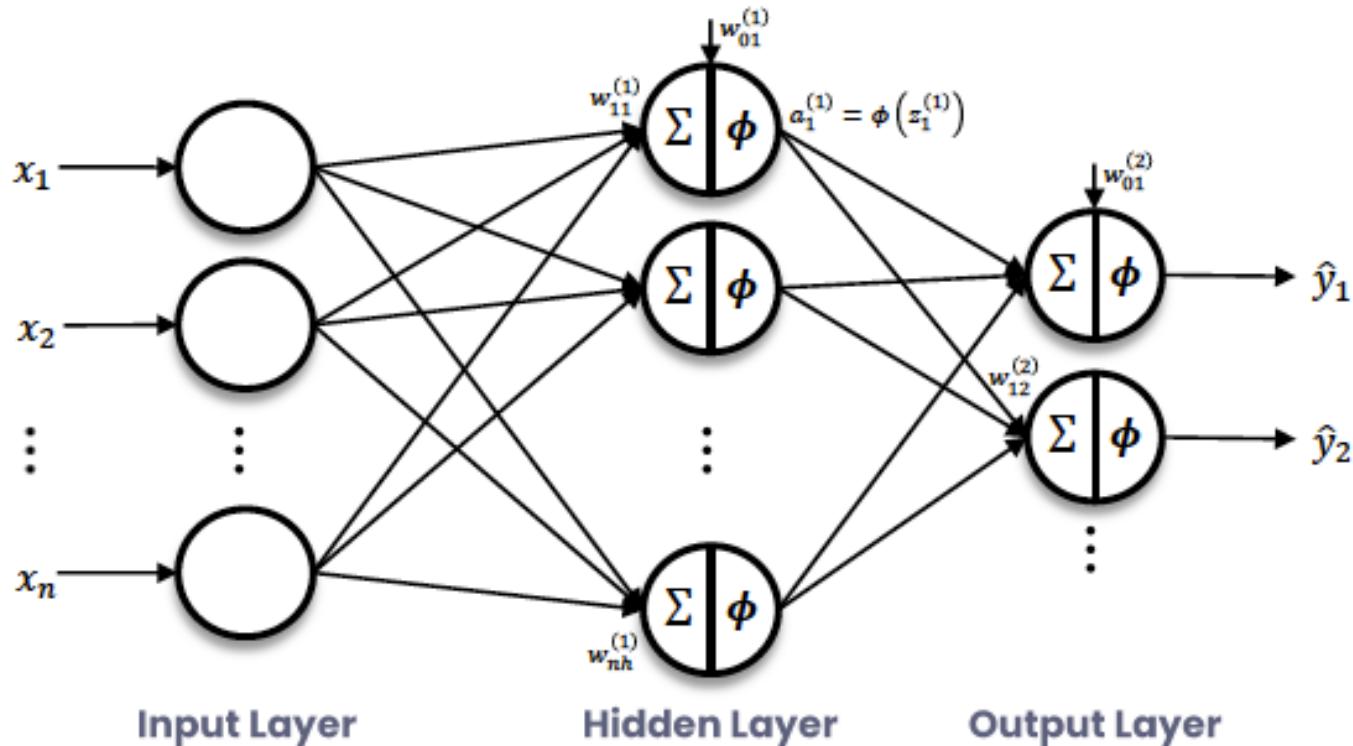
$w_{0j}^{(l)}$  este bias-ul (deplasarea) neuronului  $j$  de pe stratul  $l$ .

$z_j^{(l)}$  este ieșirea neuronului  $j$  de pe stratul  $l$  după însumarea intrărilor ponderate de la toți ceilalți neuroni.

$a_j^{(l)}$  este ieșirea neuronului  $j$  de pe stratul  $l$  după aplicarea funcției de activare  $\phi$  ieșirii  $z_j^{(l)}$ .



# Rețele feedforward multistrat de neuroni reprezentate în format matriceal



# Rețele feedforward multistrat de neuroni reprezentate în format matriceal

$$\vec{x} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad W^{(1)} = \begin{bmatrix} w_{01}^{(1)} & w_{11}^{(1)} & \cdots & w_{n1}^{(1)} \\ w_{02}^{(1)} & w_{12}^{(1)} & \cdots & w_{n2}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ w_{0h}^{(1)} & w_{1h}^{(1)} & \cdots & w_{nh}^{(1)} \end{bmatrix}_{h \times n+1} \quad W^{(1)}\vec{x} = \begin{bmatrix} \sum_{i=0}^n x_i w_{i1}^{(1)} \\ \sum_{i=0}^n x_i w_{i2}^{(1)} \\ \vdots \\ \sum_{i=0}^n x_i w_{ih}^{(1)} \end{bmatrix} = \begin{bmatrix} z_1^{(1)} \\ z_2^{(1)} \\ \vdots \\ z_h^{(1)} \end{bmatrix} = \vec{z}^{(1)}$$

$$\phi(\vec{z}^{(1)}) = \begin{bmatrix} a_1^{(1)} \\ a_2^{(1)} \\ \vdots \\ a_h^{(1)} \end{bmatrix} \quad \vec{a}^{(1)} = \begin{bmatrix} 1 \\ a_1^{(1)} \\ \vdots \\ a_h^{(1)} \end{bmatrix} \quad \hat{y} = \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_o \end{bmatrix} = \phi(W^{(2)}\phi(W^{(1)}x))$$



# Antrenarea unei rețele feedforward multistrat de neuroni

- O rețea se antrenează folosind algoritmul de backpropagation = propagarea erorii înapoi
- Varianta cea mai folosită este utilizarea unui algoritm stochastic de coborâre pe gradient (stochastic gradient descent)
  - “stochastic” întrucât gradientul se calculează în funcție de un exemplu sau o mulțime redusă de exemple (batch) și nu în raport cu întreaga mulțime
- Trebuie să calculăm gradientul funcției de eroare  $E$  în raport cu fiecare pondere din rețea și să facem actualizările corespunzătoare:

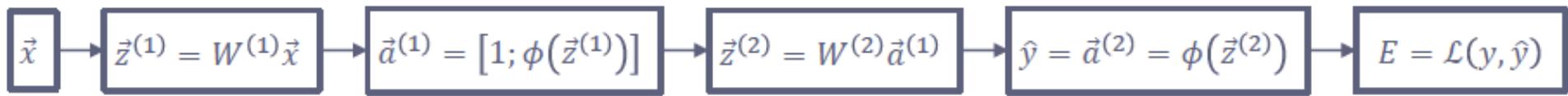
$$E(\vec{x}) = \mathcal{L}(y, \hat{y})$$

$$\Delta w_{ij}^{(l)} = -\eta \frac{\partial E}{\partial w_{ij}^{(l)}}$$

Funcția de eroare  $E$  ia valori mici când cele două etichete iau valori apropiate și ia valori mari altfel

- În format matriceal putem scrie:  $\Delta W^{(l)} = -\eta \frac{\partial E}{\partial W^{(l)}}$

# Regula de înlăntuire a derivatelor



- Trebuie să calculăm gradientul funcției de eroare E în raport cu fiecare pondere din rețea și să facem actualizările corespunzătoare
- Funcția eroare E depinde de toate ponderile din rețea:

$$E(\vec{x}) = \mathcal{L}\left(y, \phi\left(W^{(2)}\phi\left(W^{(1)}\vec{x}\right)\right)\right)$$

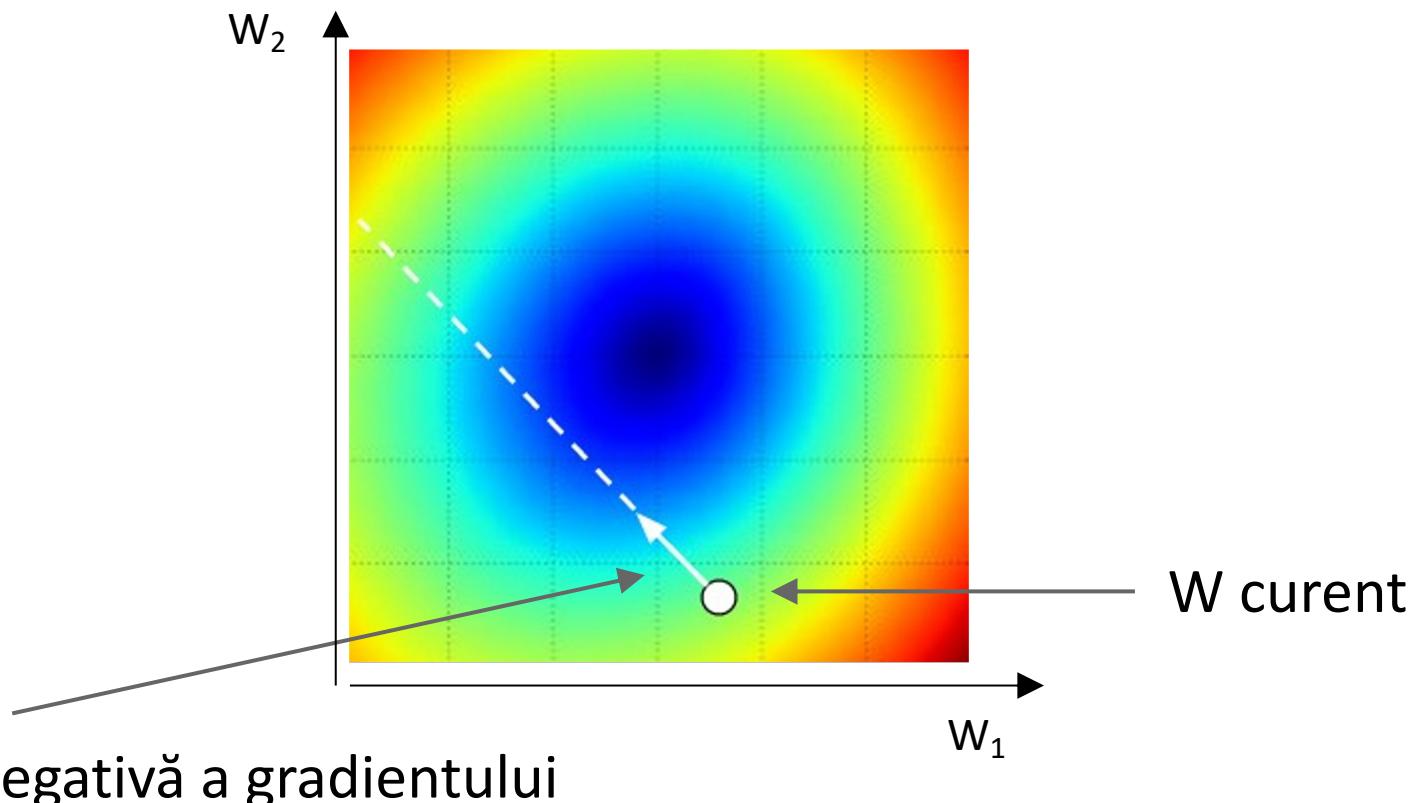
- Putem alege orice pondere  $w_{ij}^{(l)}$  și calcula prin regula de înlăntuire derivata ei corepunzătoare  $\frac{\partial E}{\partial w_{ij}^{(l)}}$ .
- În format matriceal putem scrie:

$$\frac{\partial E}{\partial W^{(1)}} = \frac{\partial E}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \vec{z}^{(2)}} \frac{\partial \vec{z}^{(2)}}{\partial \vec{a}^{(1)}} \frac{\partial \vec{a}^{(1)}}{\partial \vec{z}^{(1)}} \frac{\partial \vec{z}^{(1)}}{\partial W^{(1)}}$$

$$\frac{\partial E}{\partial W^{(2)}} = \frac{\partial E}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial \vec{z}^{(2)}} \frac{\partial \vec{z}^{(2)}}{\partial W^{(2)}}$$

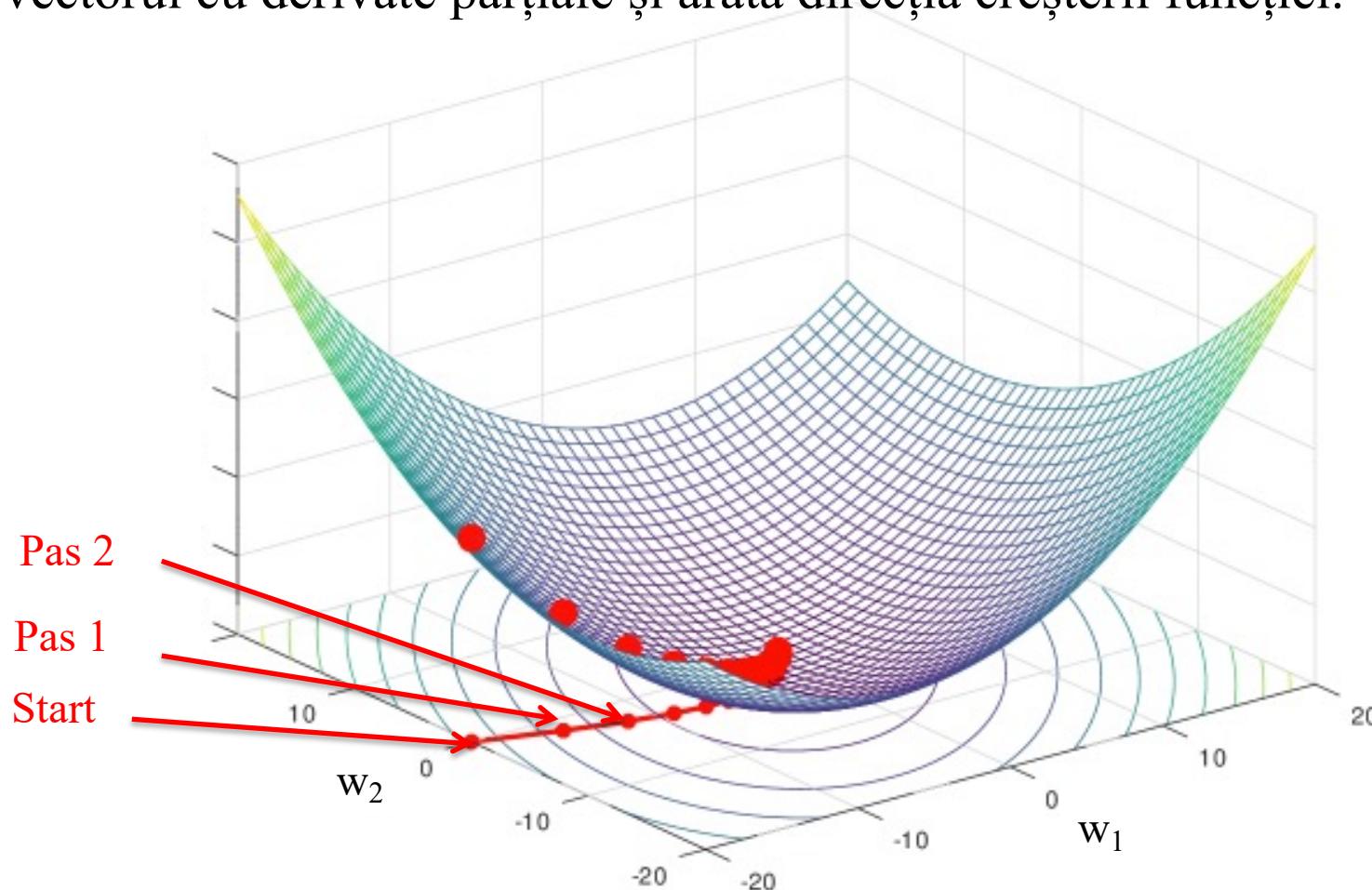
# Gradient Descent (Python)

```
def GD(w0, x, goal, learningRate):  
    perfGoalNotMet = true  
    w = w0  
  
    while perfGoalNotMet:  
        gradient = eval_gradient(x, w)  
        w_old = w  
        w = w - learningRate * gradient  
        perfGoalNotMet = sum(abs(w - w_old)) > goal
```



# Algoritmul de coborâre pe gradient

Algoritm iterativ, la fiecare pas o ia în direcția inversă a gradientului pentru minimizarea valorii funcției E. Gradientul unei funcții într-un punct este vectorul cu derivate parțiale și arată direcția creșterii funcției.



# Mini-batch Gradient Descent

- cunoscut și sub numele de **Stochastic Gradient Descent (SGD)**
- folosește doar o mică parte din mulțimea de antrenare pentru a calcula gradientul:

• • •

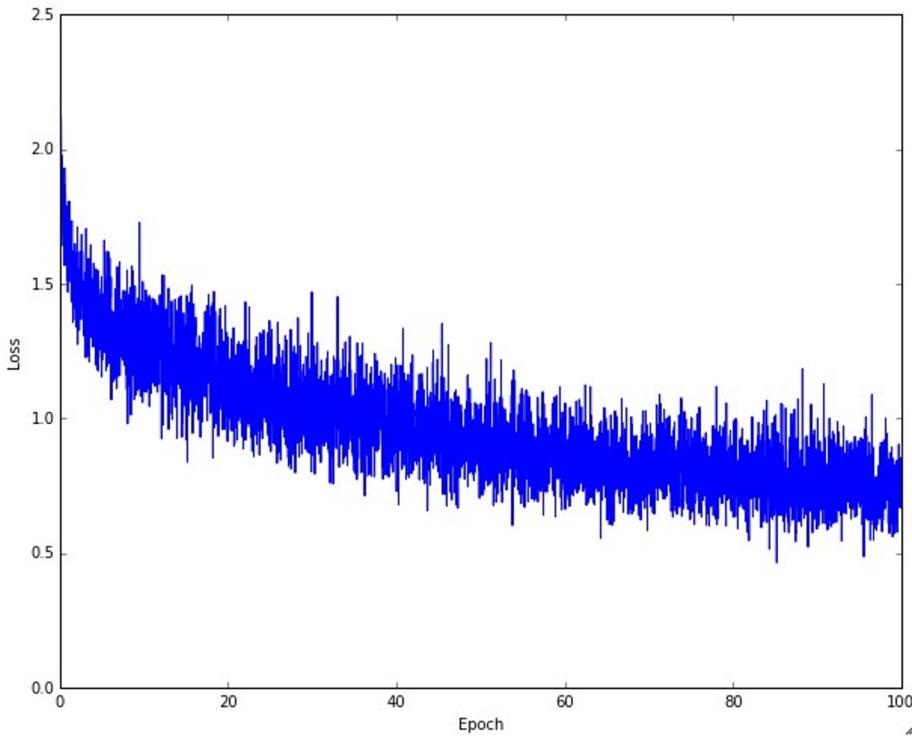
```
while perfGoalNotMet:
```

```
    x_batch = select_random_subsample(X)
    gradient = eval_gradient(@loss, x_batch, w)
```

• • •

- de regulă, dimensiunea unui mini-batch este de 32/64/128/256 de exemple
- de exemplu, AlexNet (2013) a folosit 128 de exemple

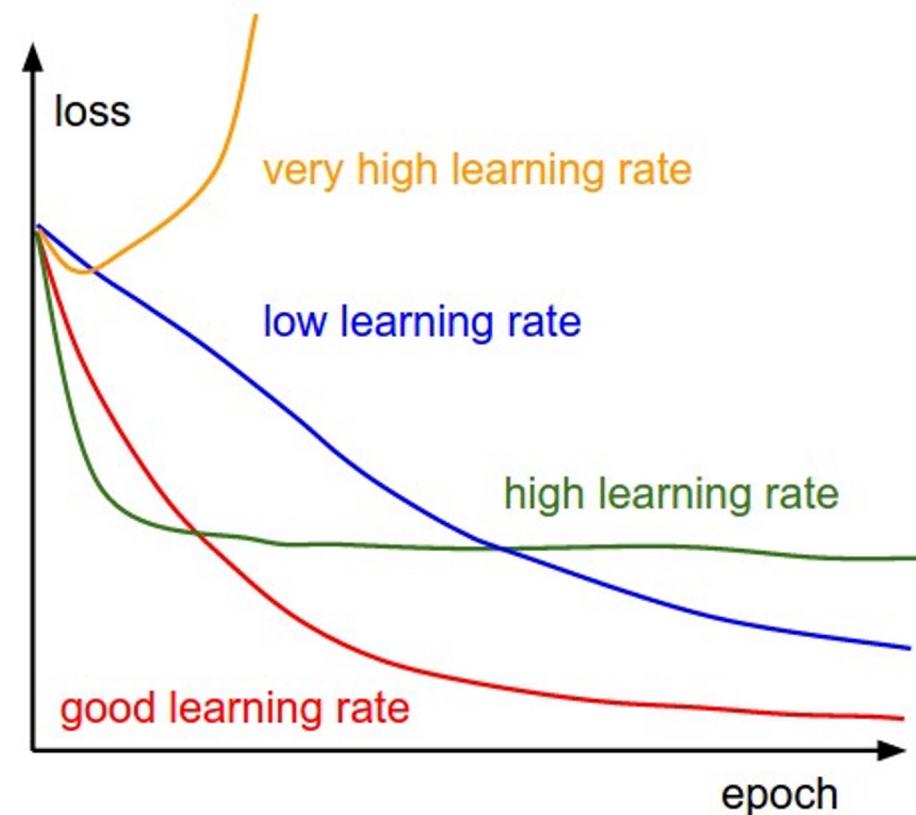
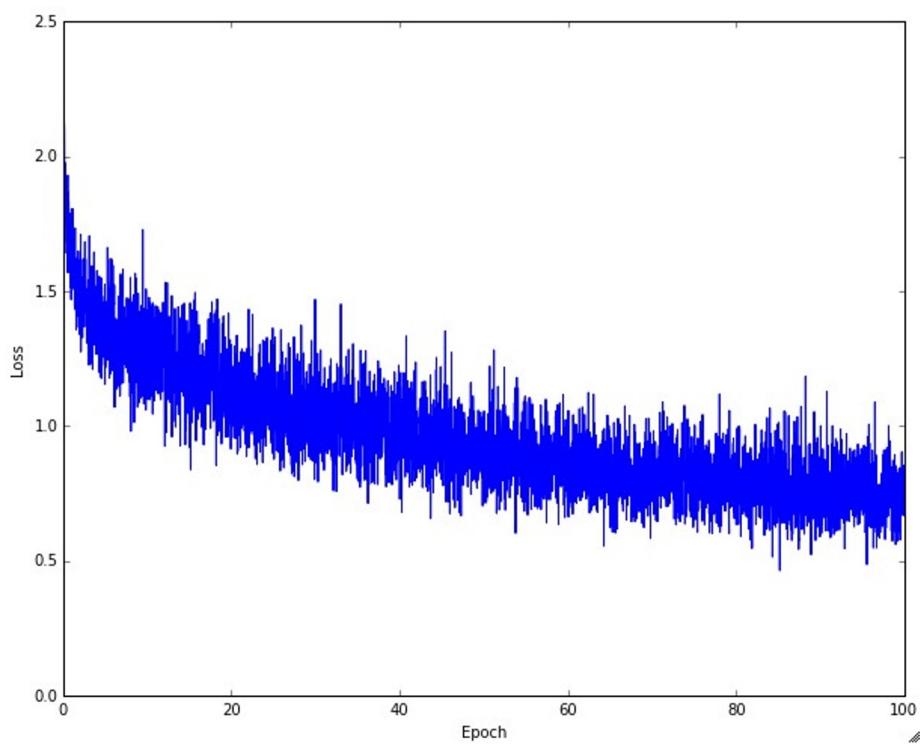
# Evoluția antrenării cu SGD



Evoluția funcției de pierdere (loss) pe multimea de antrenare folosind SGD pentru antrenarea rețelei

(funcția de pierdere (loss) calculată pe mini-batch-uri descrește pe parcursul epocilor)

# Influența ratei de învățare



# Alegerea funcției de activare

- Funcția hardlim (funcția de activare a perceptronului) nu este derivabilă în 0 iar în toate celelalte puncte derivata sa este nulă, deci orice actualizare este nulă.
- Funcții uzuale de activare:
  - funcția identitate  $f(x) = x$ ;
  - funcția logistică  $f(x) = 1/(1+e^{-x})$ ;
  - funcția tangentă hiperbolică  $f(x) = \tanh(x) = (e^x - e^{-x})/(e^x + e^{-x})$ ;
  - funcția relu  $f(x) = \max(0, x)$ .

Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) \underset{x=0}{\begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
Tanh		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
Arctan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU) [2]		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parameteric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

# Alegerea funcției de activare

- Funcția hardlim (funcția de activare a perceptronului) nu este derivabilă în 0 iar în toate celelalte puncte derivata sa este nulă, deci orice actualizare este nulă.
- Funcții uzuale de activare:
  - funcția identitate  $f(x) = x$ ;
  - funcția logistică  $f(x) = 1/(1+e^{-x})$ ;
  - funcția tangentă hiperbolică  $f(x) = \tanh(x) = (e^x - e^{-x})/(e^x + e^{-x})$ ;
  - funcția relu  $f(x) = \max(0, x)$ .
- Funcții de activare predispuse la fenomenul de "vanishing gradients"
  - logistică
  - tangentă hiperbolică
  - arcTan

# Alegerea funcției de eroare E

- Funcția pătratică de eroare

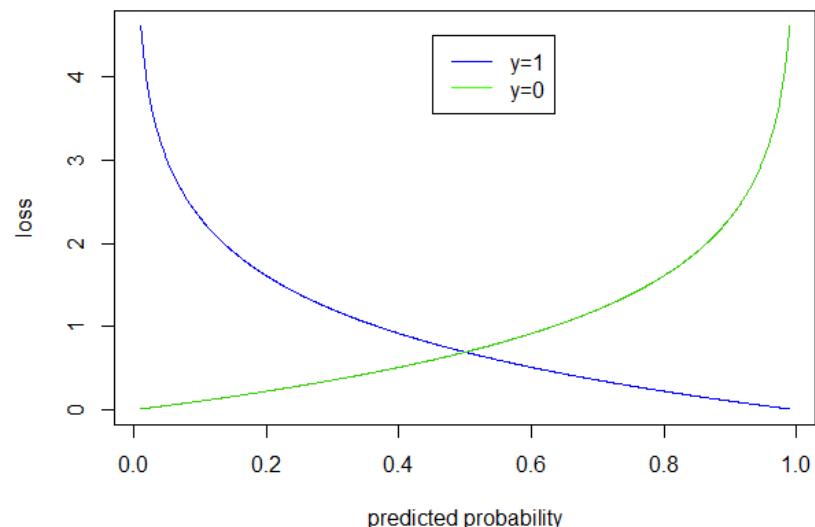
$$\mathcal{L}(y, \hat{y}) = \frac{1}{2} (y - \hat{y})^2$$

$$\mathcal{L}'(y, \hat{y}) = -(y - \hat{y})$$

- În practică, se folosește funcția de eroare bazată pe cross-entropie:

$$\mathcal{L}(y, \hat{y}) = -y \log \hat{y} - (1 - y) \log(1 - \hat{y})$$

$$\mathcal{L}'(y, \hat{y}) = -\frac{y}{\hat{y}} + \frac{1 - y}{1 - \hat{y}}$$



# Teorema de aproximare universală

George Cybenko, 1989, Kurt Hornik, 1991:

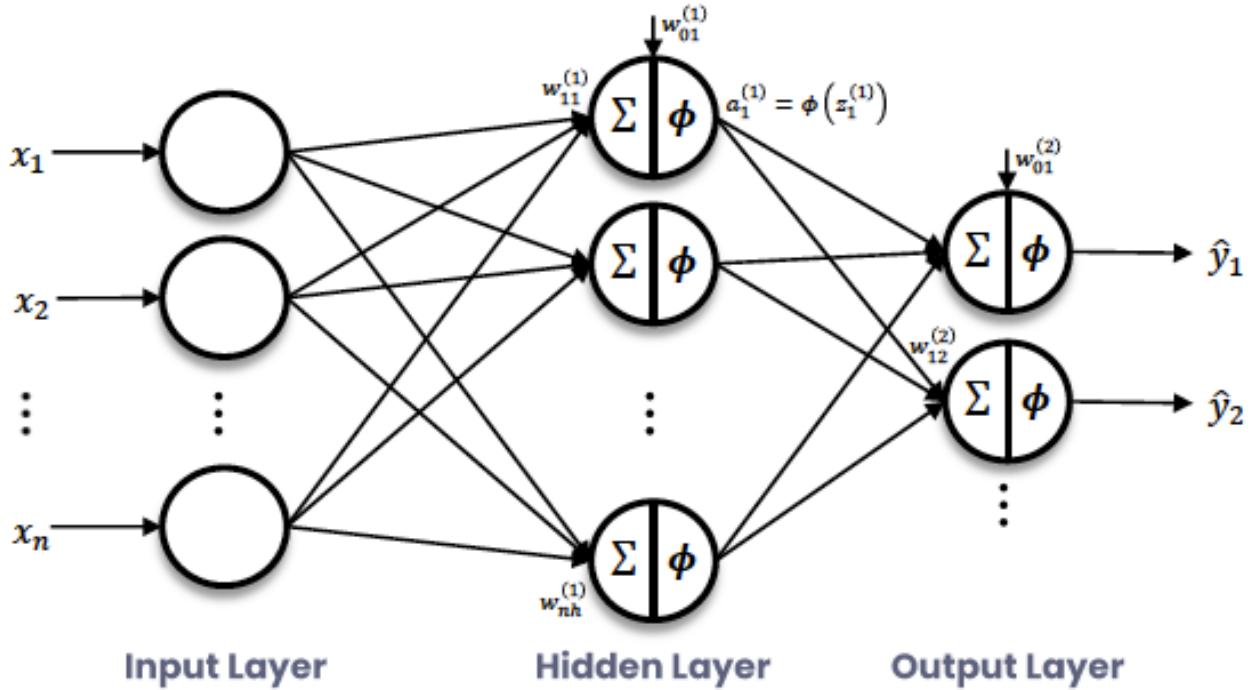
*O rețea neuronală cu un (singur) strat ascuns și un număr finit de neuroni poate modela orice funcție continuă pe subseturi compacte din  $\mathbb{R}^n$  oricât de complexă, cu o precizie oricât de bună dacă sunt utilizati suficienți neuroni pe stratul ascuns și funcții de activare potrivite.*

- numărul de neuroni de pe stratul ascuns poate fi exponential de mare
- **în practică, preferăm rețelele mai adânci (cu mai multe straturi) dar mai puțini neuroni pe strat**

# Limitări ale rețelelor feedforward multistrat de perceptroni

100 samples of each digit

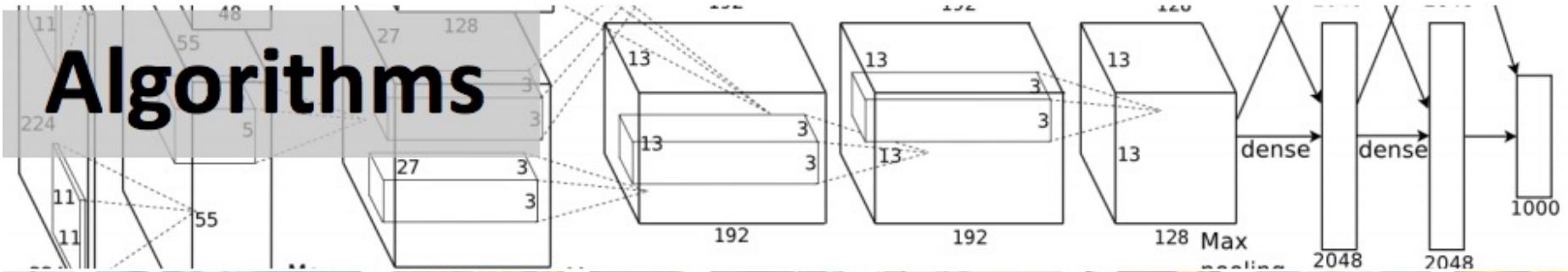
8	0	3	6	1	0	0	4	1	7
3	2	4	9	5	1	1	9	2	8
1	5	8	1	2	0	5	8	3	0
3	5	4	0	6	9	7	5	5	2
2	7	0	0	8	8	6	1	8	4
9	8	6	1	3	3	3	2	6	4
6	5	7	7	8	7	3	3	6	5
4	6	7	7	4	4	6	2	3	6
5	5	5	4	3	1	5	5	1	2
2	3	9	7	5	0	6	2	4	2
4	6	7	4	5	0	8	7	3	2
0	2	9	4	7	1	6	3	2	1
6	9	8	7	9	2	6	3	8	8
3	4	4	9	6	5	1	9	3	7
5	8	4	0	5	3	8	3	1	1
8	2	0	3	5	5	6	7	4	1
3	7	6	1	7	9	6	2	5	1
8	6	1	7	9	5	2	2	5	4
7	9	5	0	3	1	4	5	2	4
5	2	4	8	8	5	8	✓	8	1



O rețea feedforward antrenată pentru a recunoaște cifre (obiecte) într-o imagine nu este robustă la translații mici în imagine ale cifrelor (obiectelor)

# Ingrediente pentru succesul învățării automate în procesarea imaginilor

**Algorithms**



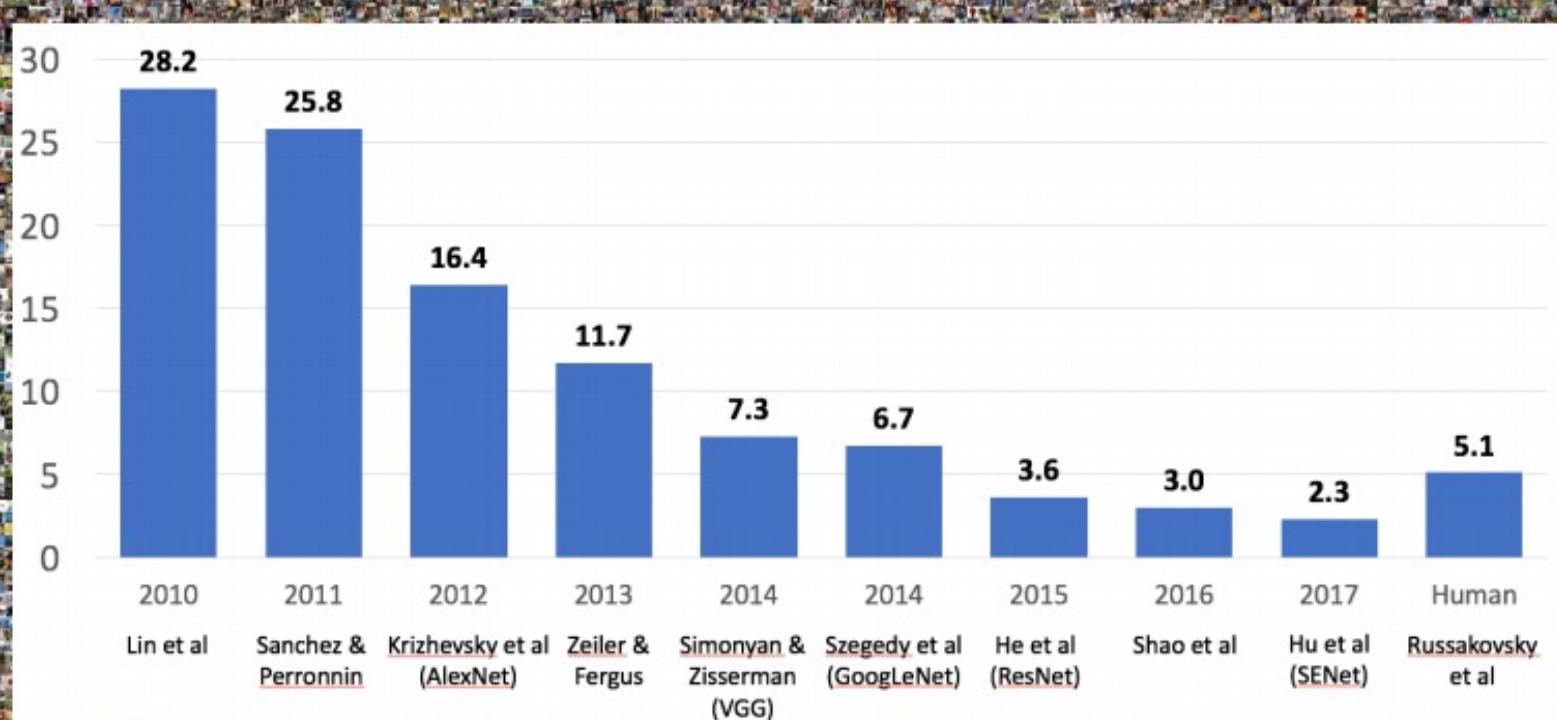
**Data**



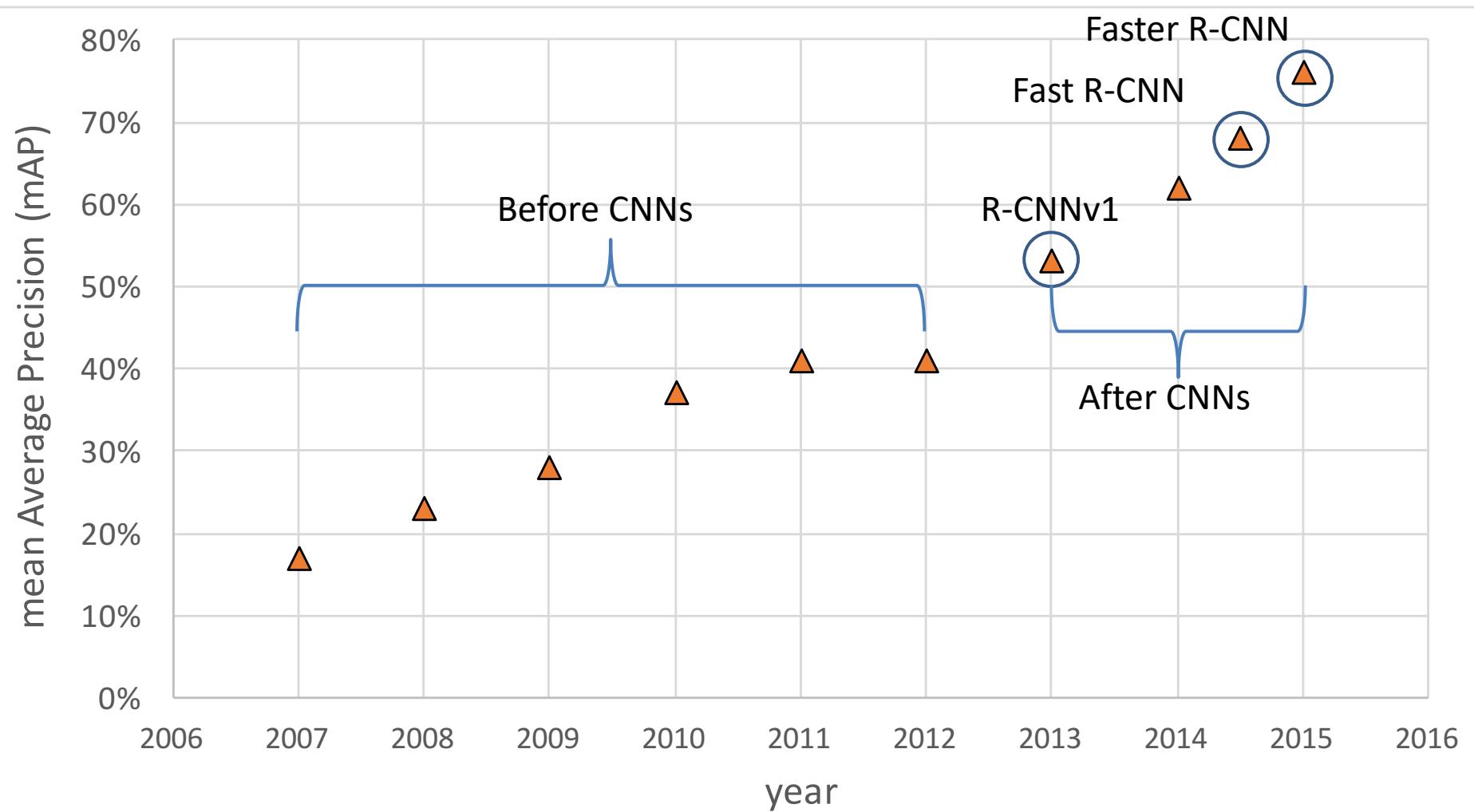
**Computation**



## Clasificarea imaginilor 1,431,167 imagini adnotate cu 1000 clase de obiecte

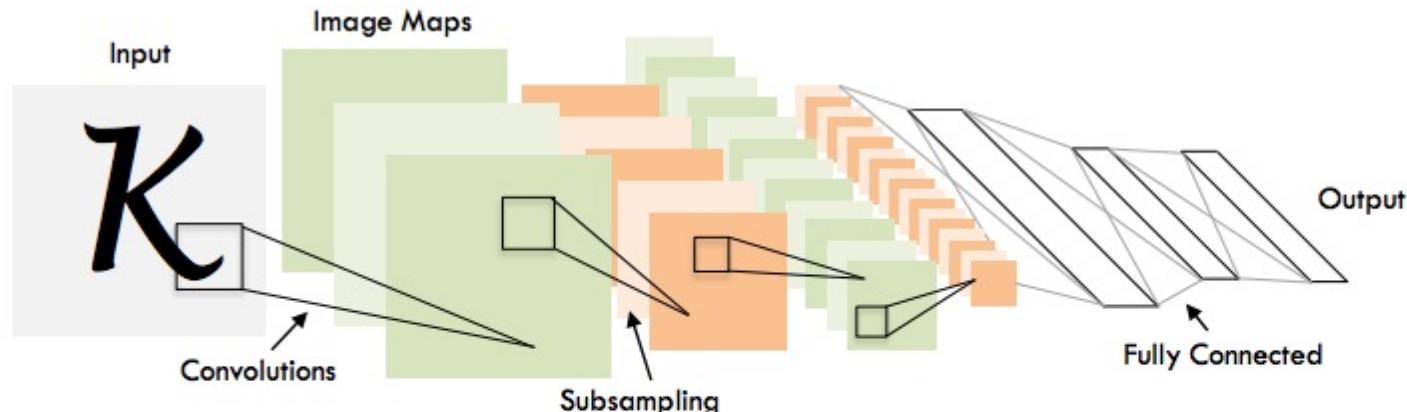


# Detectarea de obiecte pe setul VOC PASCAL 2007



1998

LeCun et al.



# of transistors



$10^6$

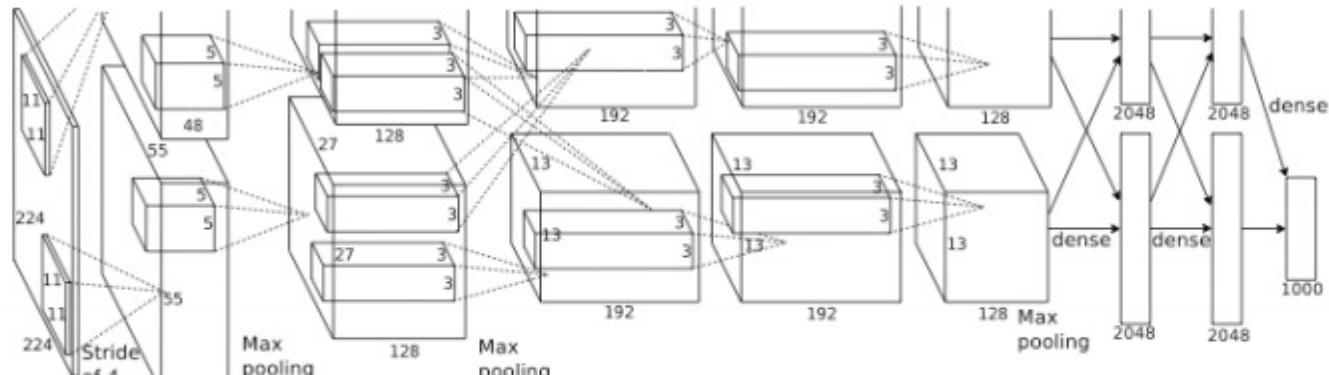
pentium® II

# of pixels used in training

$10^7$  NIST

2012

Krizhevsky et al.



# of transistors



$10^9$

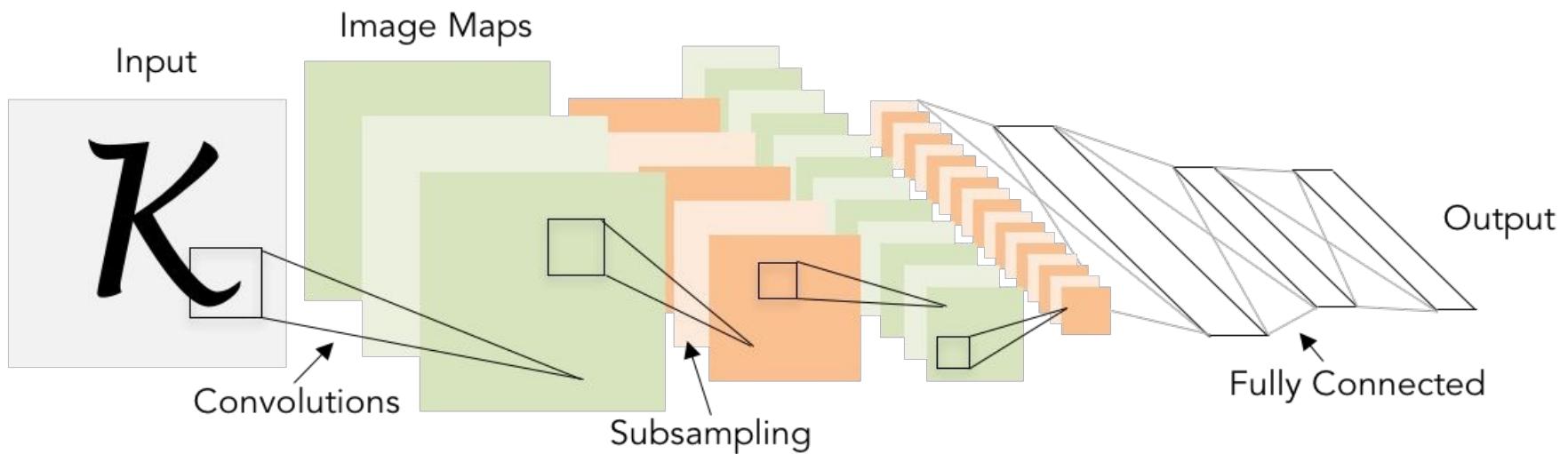
GPUs



# of pixels used in training

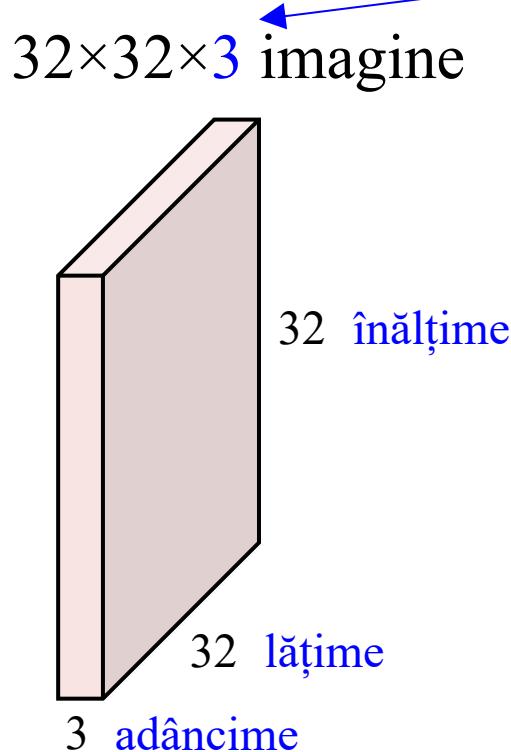
$10^{14}$  IMAGENET

# Rețele neuronale convecționale



[LeNet-5, LeCun 1998]

# Strat convolutional



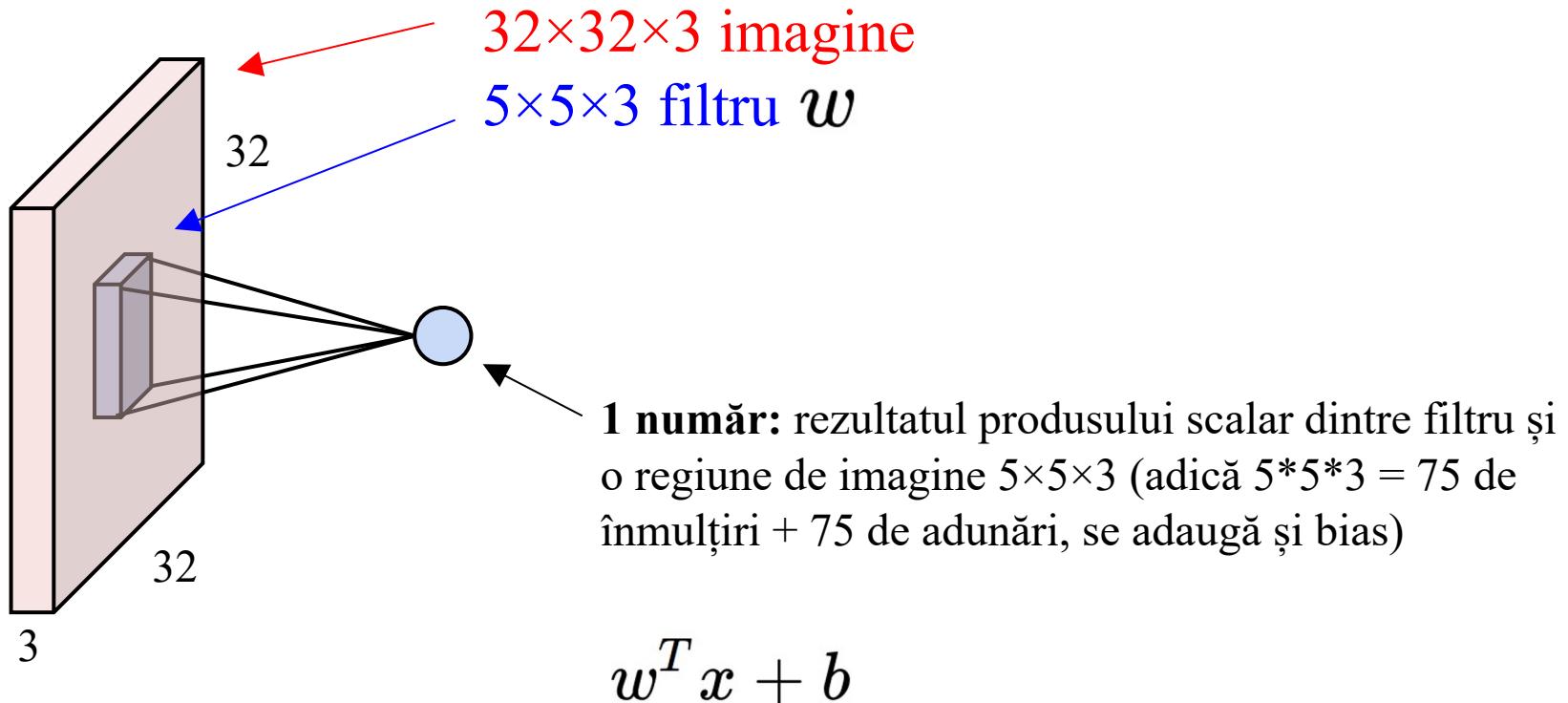
Filtrele au aceeași dimensiune în adâncime cu imaginea dată

5×5×3 filtru

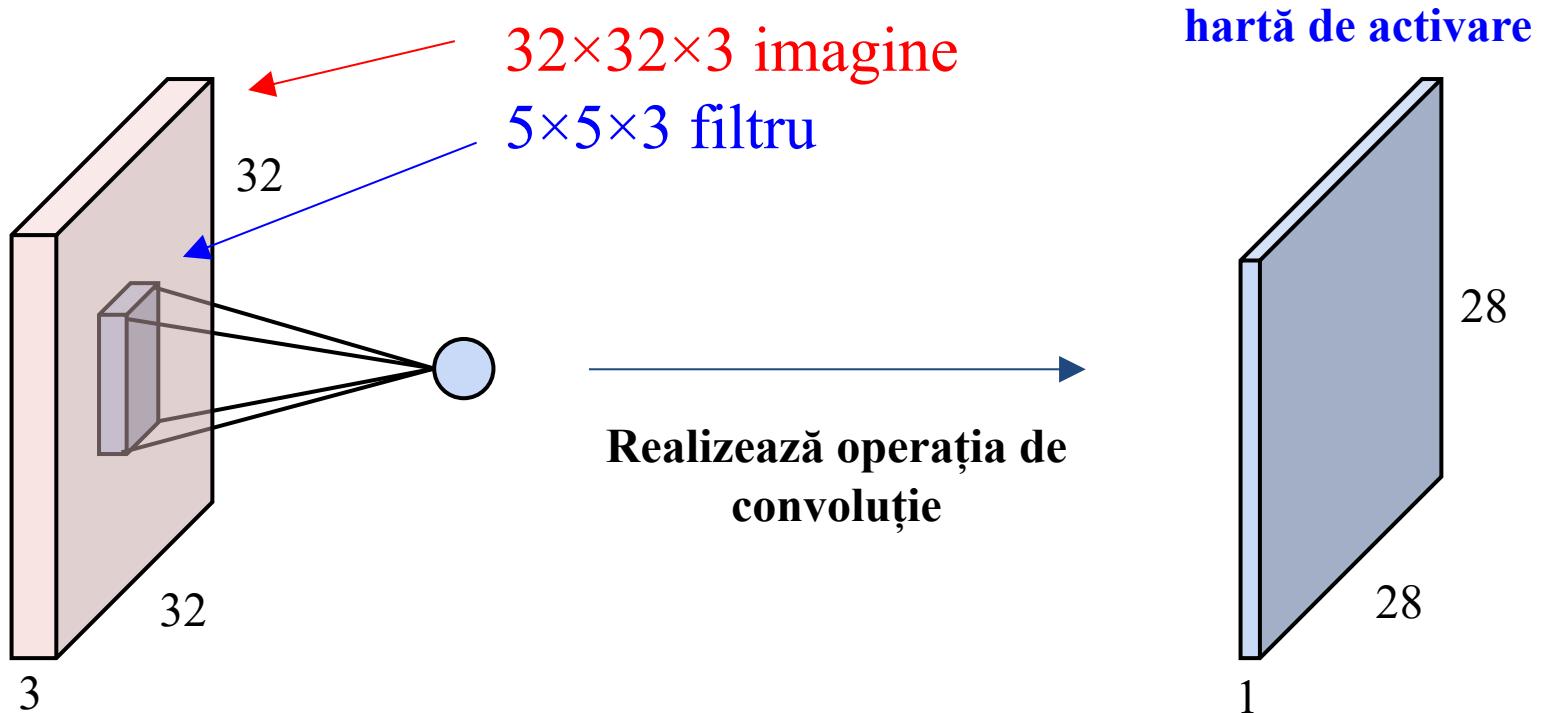


**Realizează operația de conoluție** între un filtru și o imagine, adică glisează de la stânga la dreapta și de sus în jos cu filtrul pe imagine și calculează la fiecare poziție produsul scalar dintre elementele filtrului și valorile din imagine.

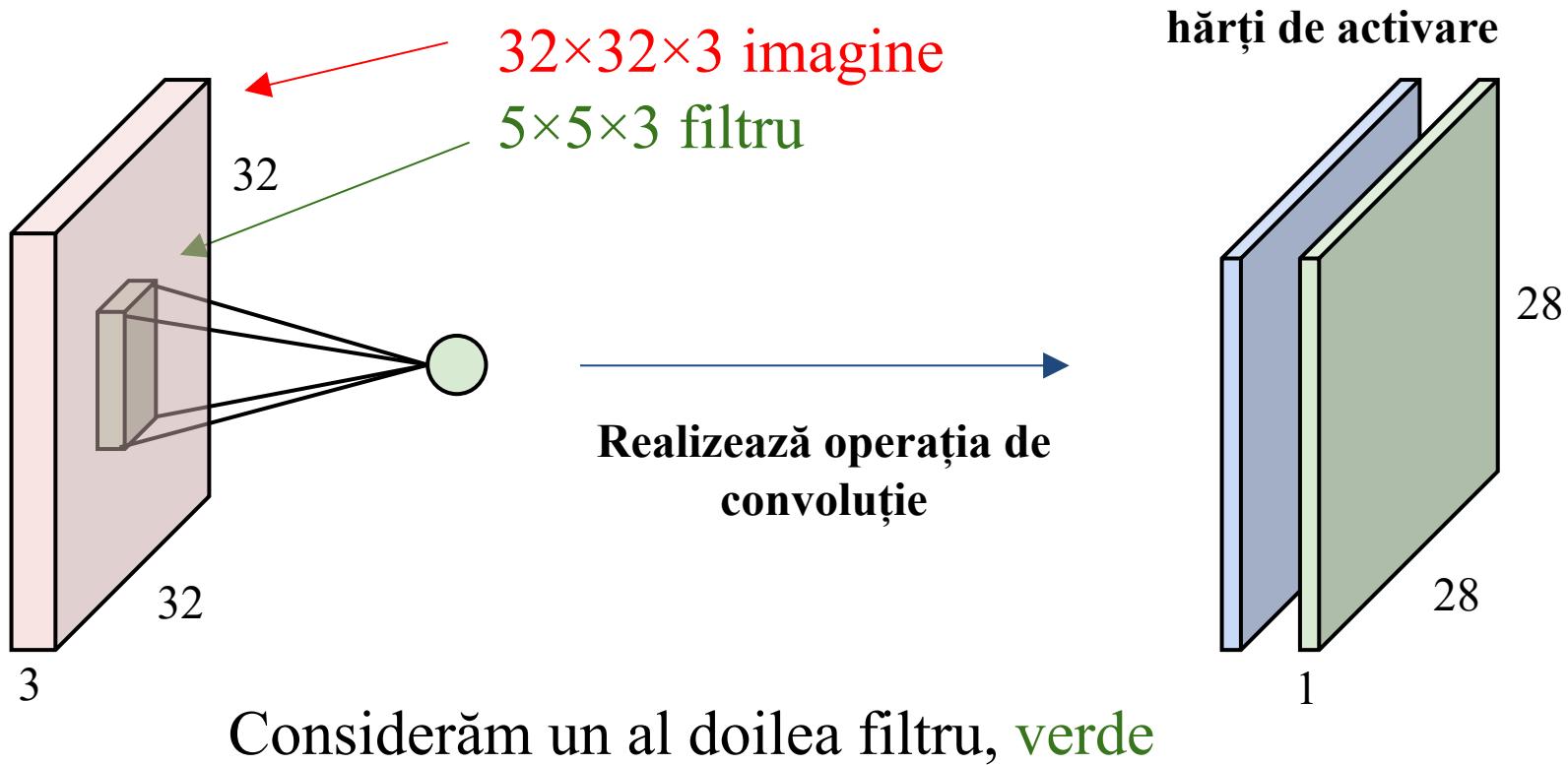
# Strat convolutional



# Strat convolutional

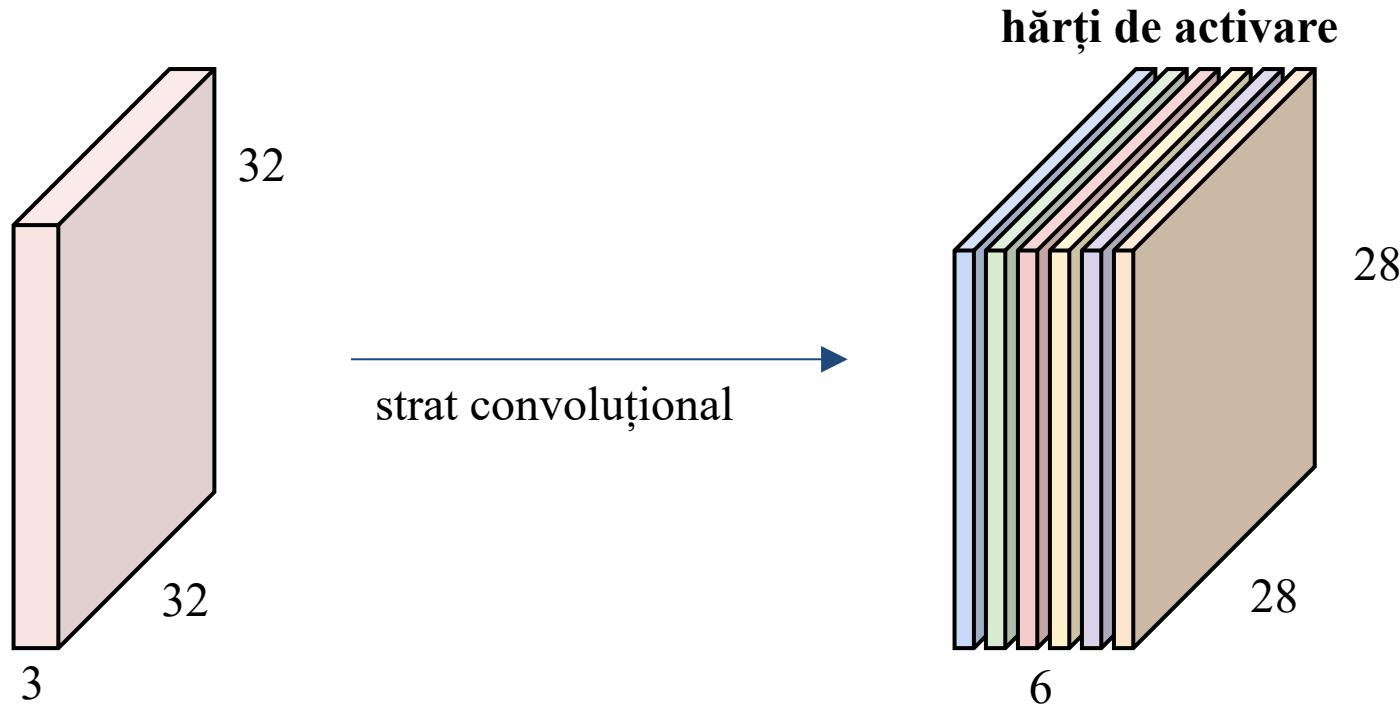


# Strat convolutional



# Strat conoluțional

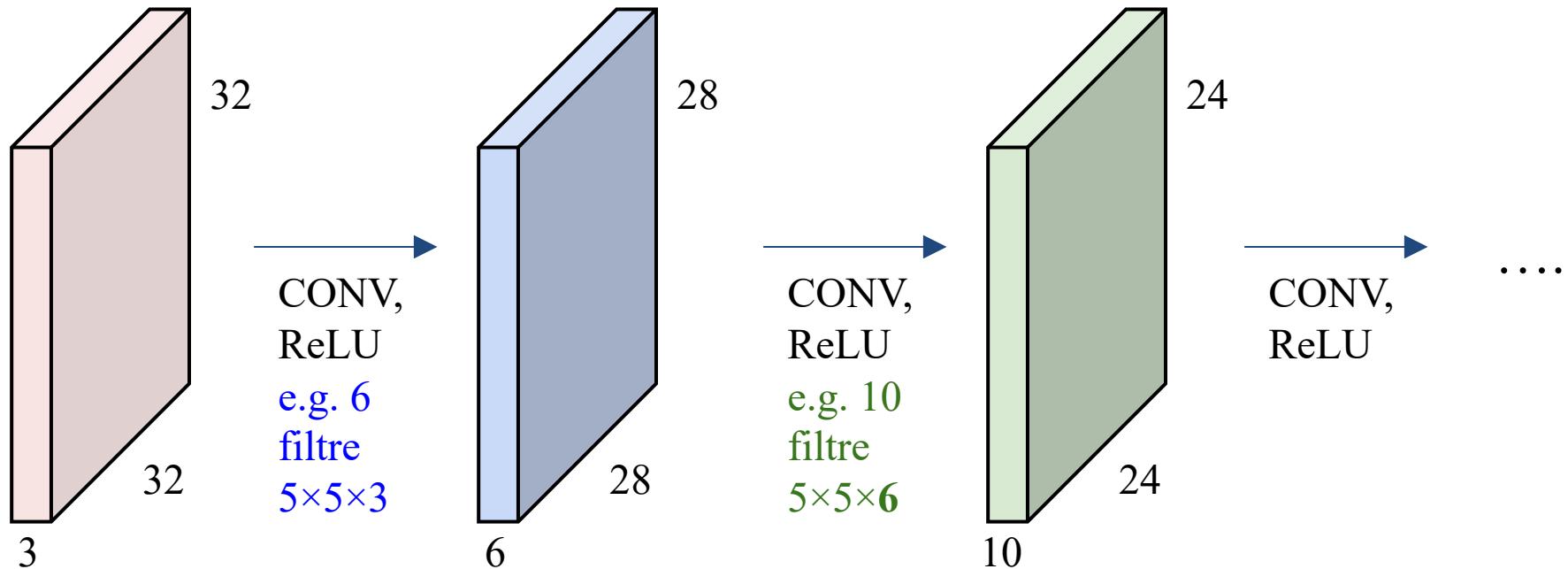
De exemplu, dacă avem 6 filtre  $5 \times 5$ , am obține 6 hărți de activare:



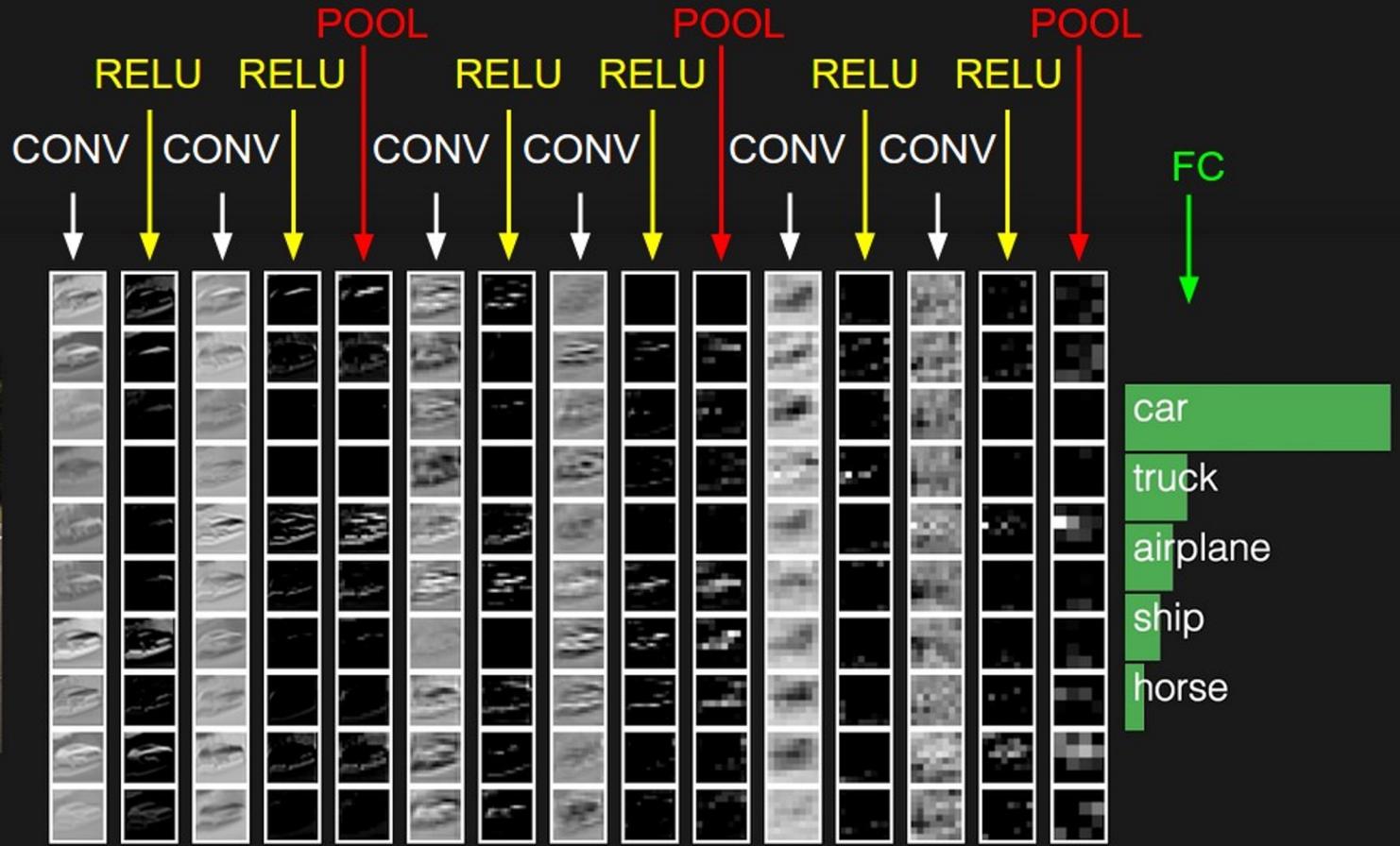
Acumulăm aceste hărți de activare și obținem o “nouă imagine” de dimensiuni  $28 \times 28 \times 6$ !

# Rețea conoluțională

**Preview:** O rețea conoluțională este o secvență de straturi conoluționale, la care adaugăm funcții de activare (și straturi de pooling)

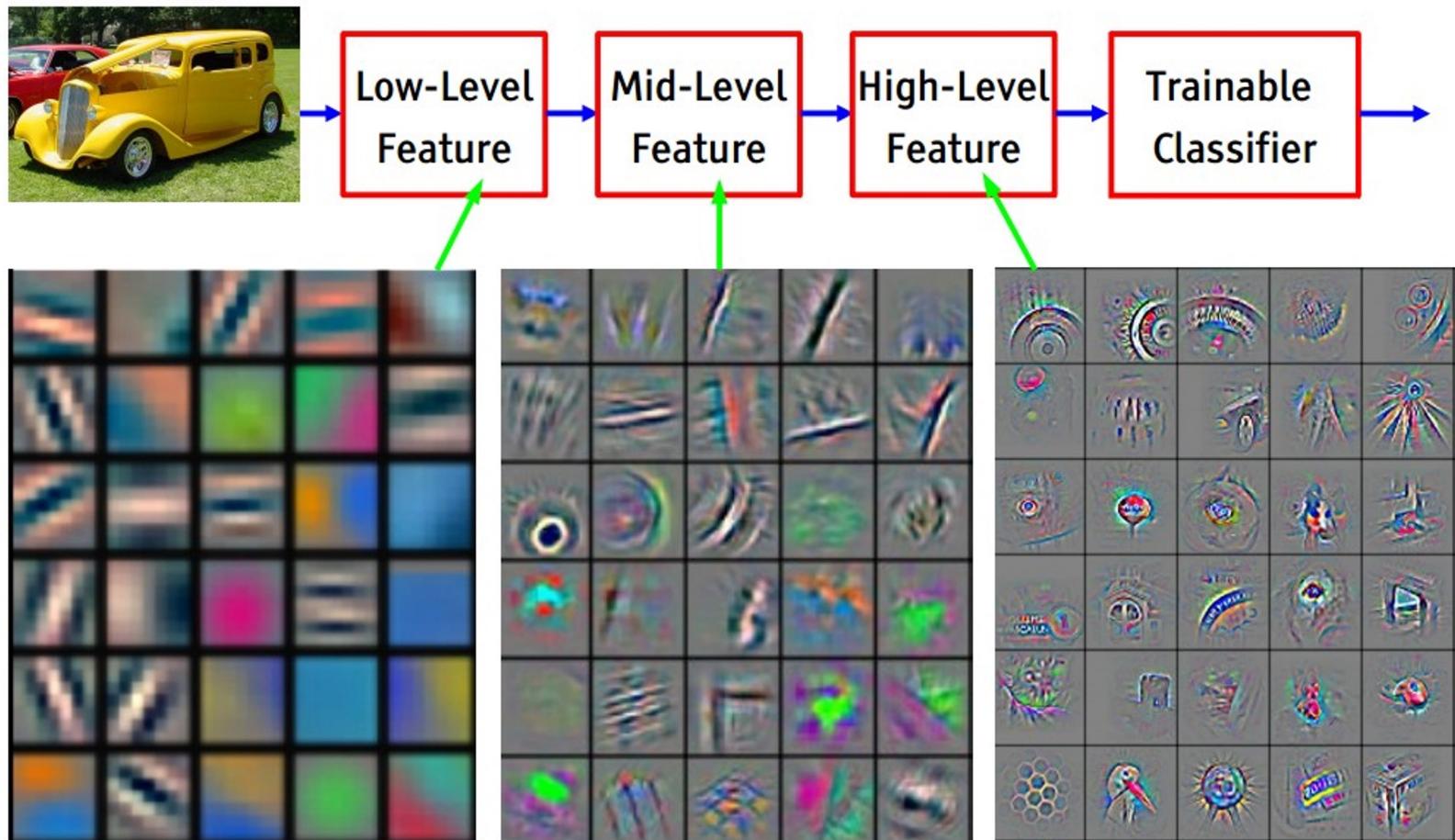


# Rețea convețională



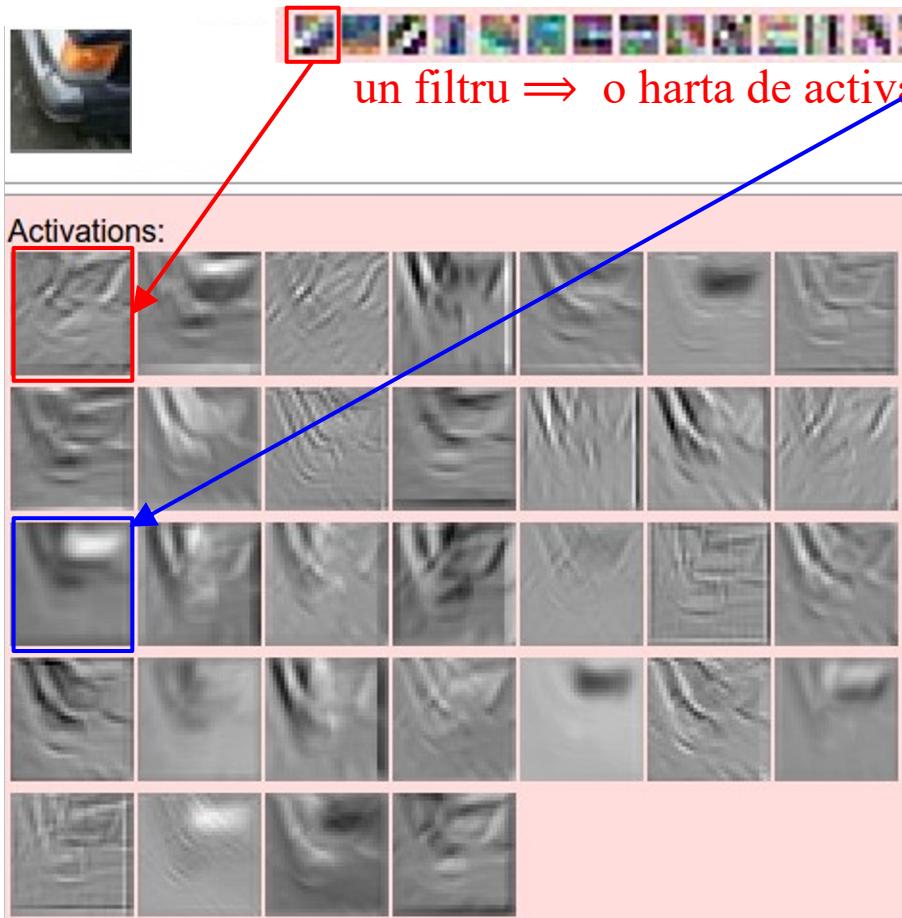
# Vizualizarea caracteristicilor învățate de filtre

Filtrele învățate corespund unor caracteristici / părți de obiecte



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

# Vizualizarea caracteristicilor învățate de filtre



un filtru  $\Rightarrow$  o harta de activare

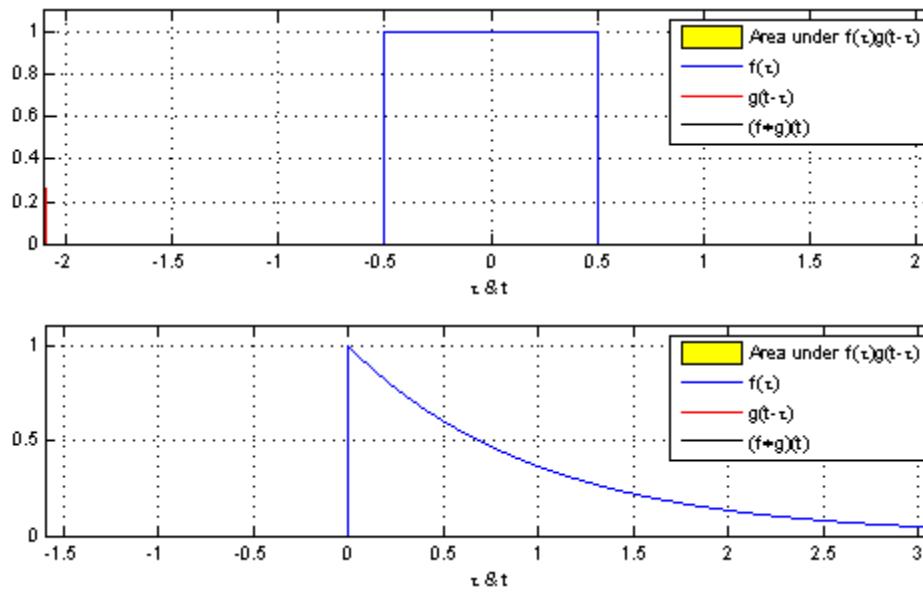
exemple filtre 5x5  
(32 în total)

numim stratul convecțional întrucât este legat de convecția a două semnale:

$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1, n_2] \cdot g[x - n_1, y - n_2]$$

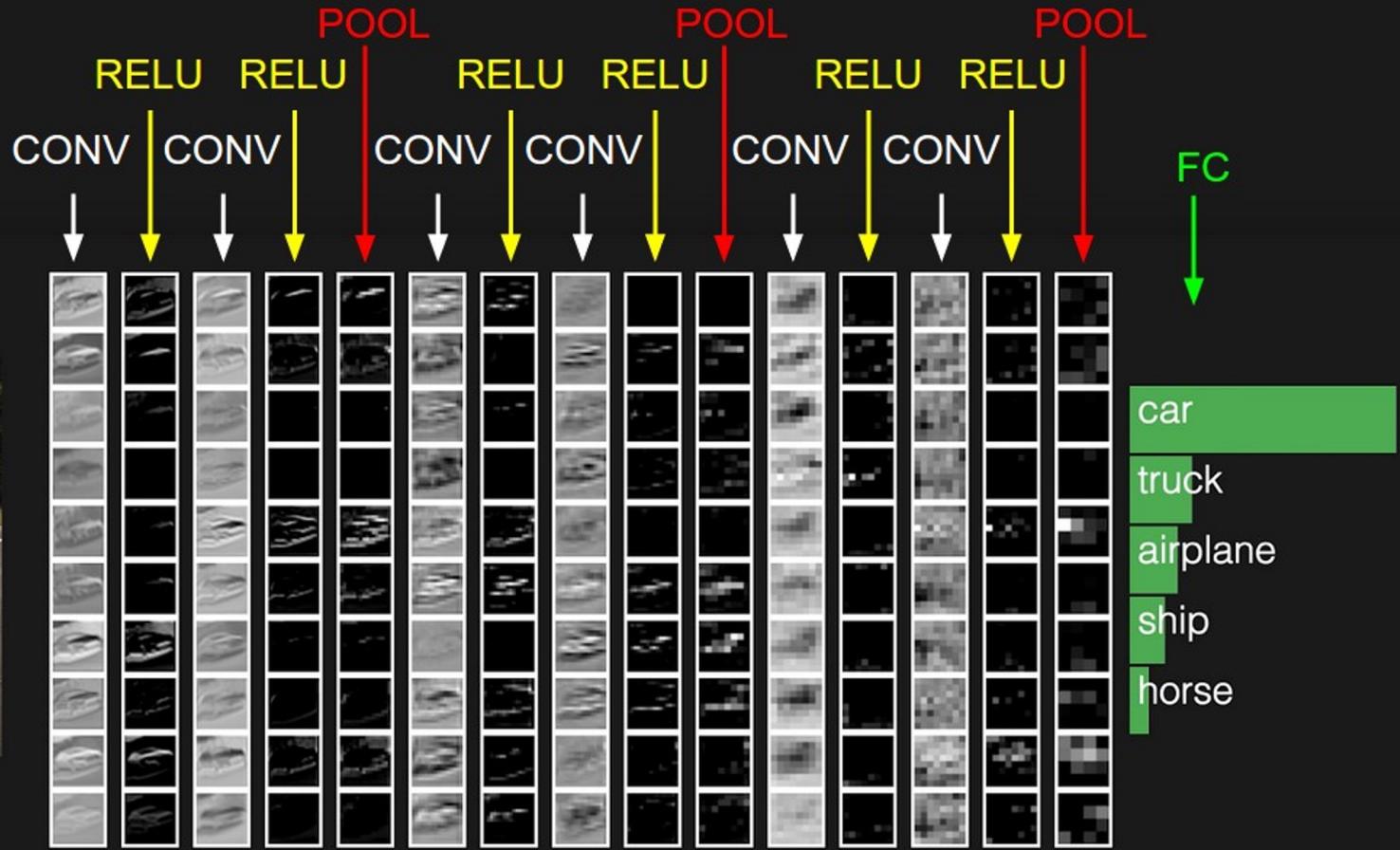
operăția de convecție dintre un filtru și un semnal (imagină)

Activare maximă = răspuns cel mai mare

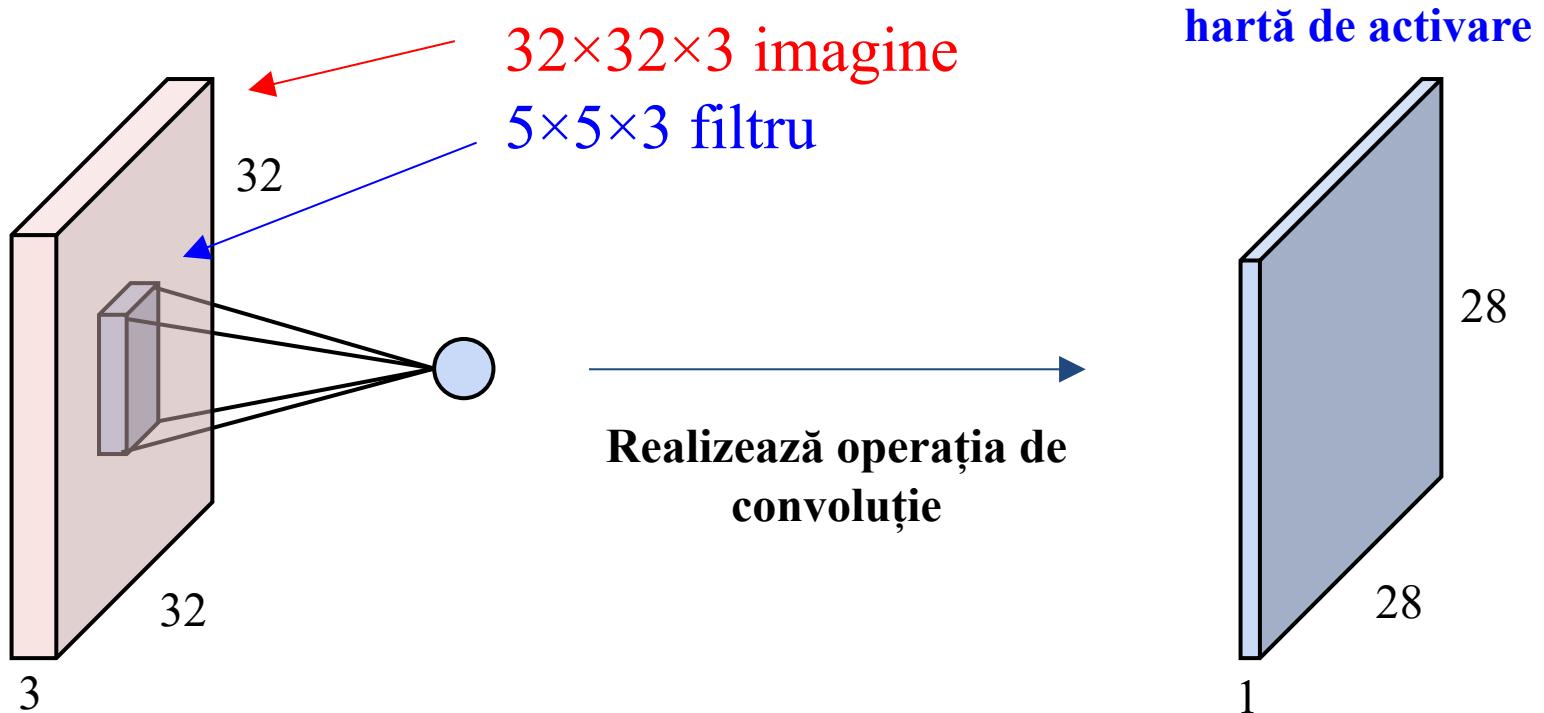


[https://commons.wikimedia.org/wiki/File:Convolution\\_of\\_box\\_signal\\_with\\_itself.gif](https://commons.wikimedia.org/wiki/File:Convolution_of_box_signal_with_itself.gif)

# Rețea convețională

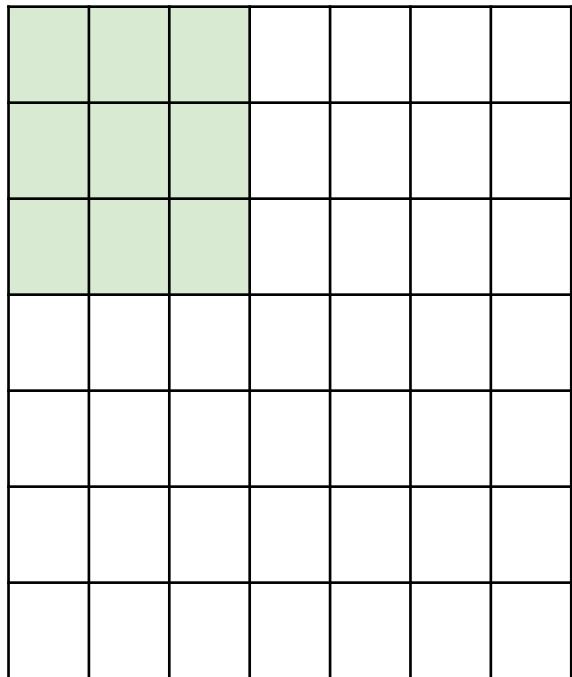


# Dimensiunea unei hărți de activare



# Dimensiunea unei hărți de activare

7

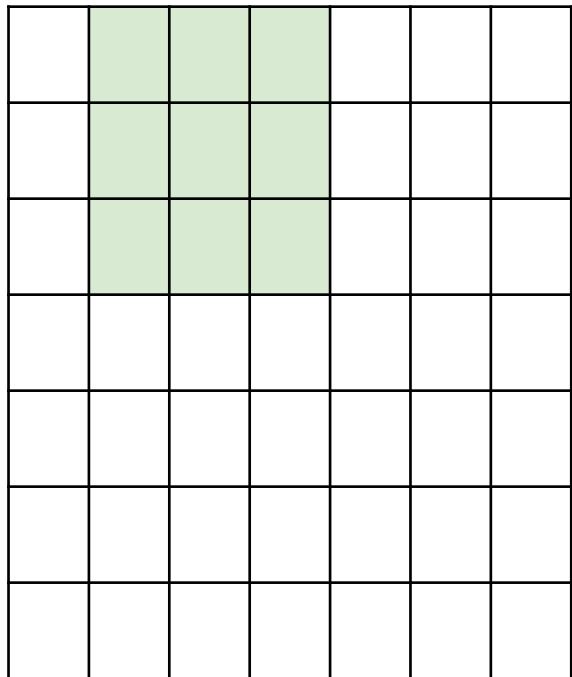


$7 \times 7$  input (numai 2D, fără a treia dimensiune)  
considerăm un filtru  $3 \times 3$

7

# Dimensiunea unei hărți de activare

7

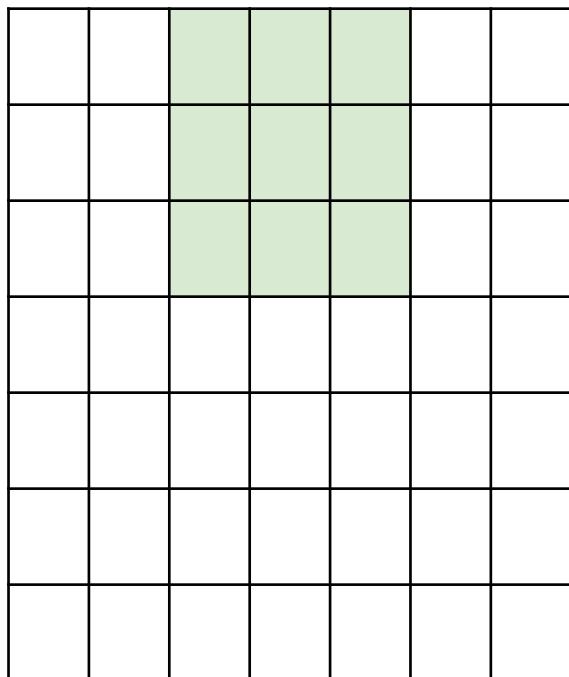


$7 \times 7$  input (numai 2D, fără a treia dimensiune)  
considerăm un filtru  $3 \times 3$

7

# Dimensiunea unei hărți de activare

7

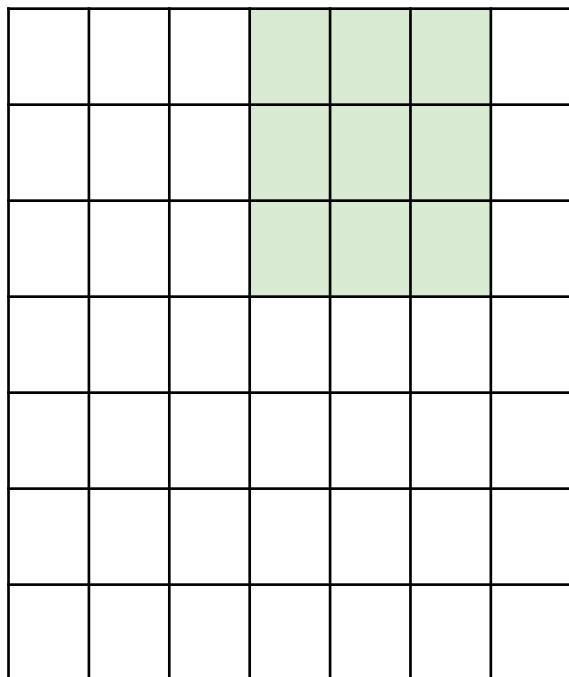


$7 \times 7$  input (numai 2D, fără a treia dimensiune)  
considerăm un filtru  $3 \times 3$

7

# Dimensiunea unei hărți de activare

7

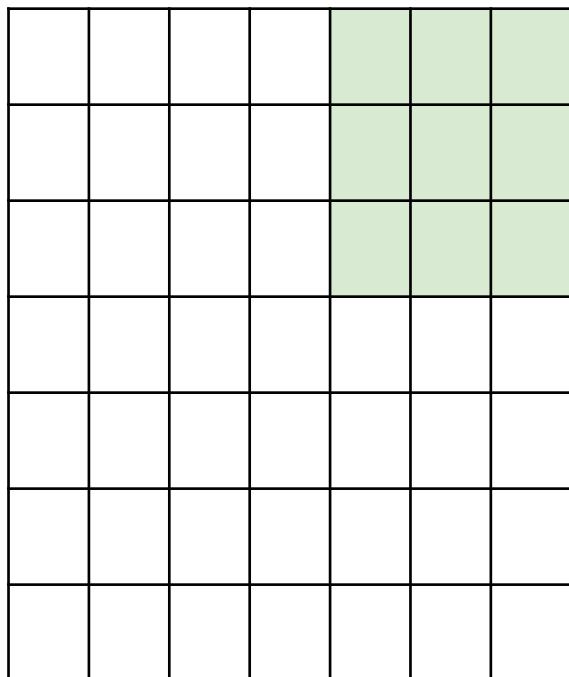


$7 \times 7$  input (numai 2D, fără a treia dimensiune)  
considerăm un filtru  $3 \times 3$

7

# Dimensiunea unei hărți de activare

7



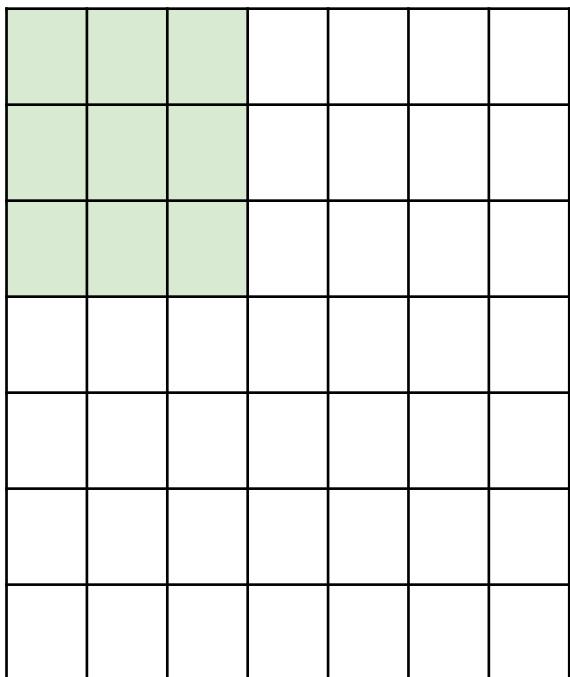
$7 \times 7$  input (numai 2D, fără a treia dimensiune)  
considerăm un filtru  $3 \times 3$

$\Rightarrow$  **output  $5 \times 5$**

7

# Dimensiunea unei hărți de activare

7

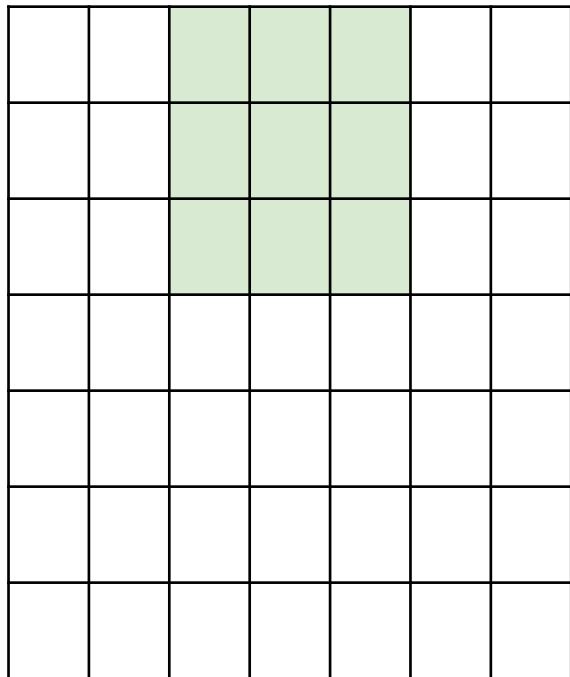


$7 \times 7$  input (numai 2D, fără a treia dimensiune)  
considerăm un filtru  $3 \times 3$   
aplicat cu **stride 2**

7

# Dimensiunea unei hărți de activare

7

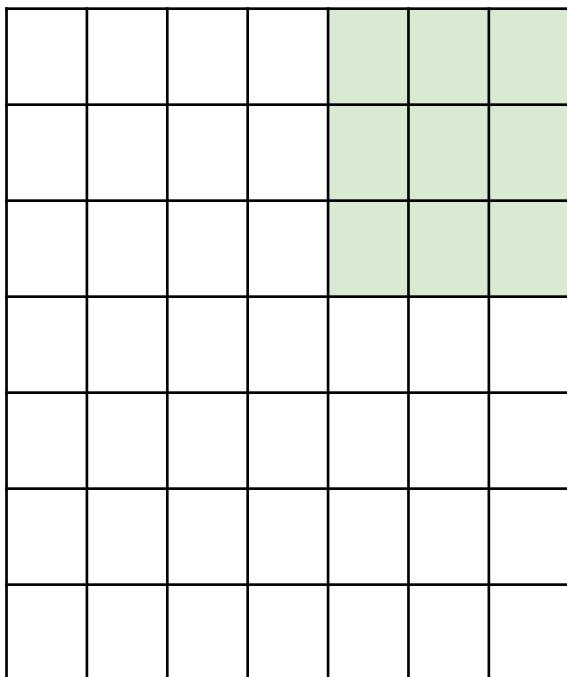


$7 \times 7$  input (numai 2D, fără a treia dimensiune)  
considerăm un filtru  $3 \times 3$   
aplicat cu **stride 2**

7

# Dimensiunea unei hărți de activare

7

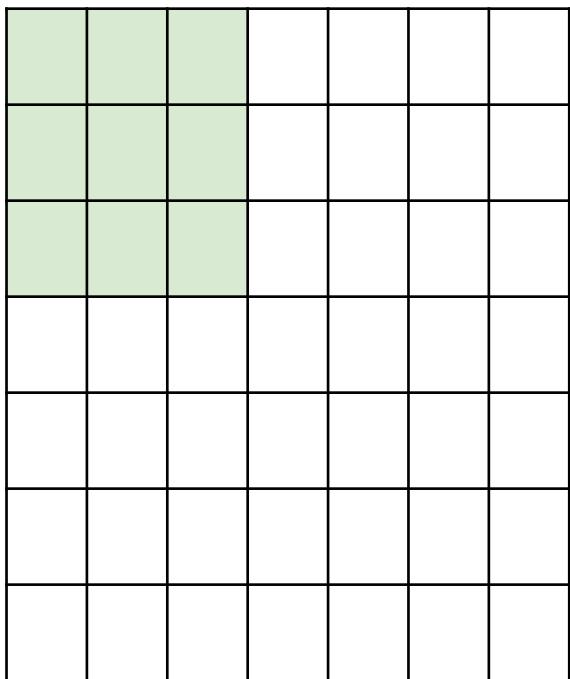


$7 \times 7$  input (numai 2D, fără a treia dimensiune)  
considerăm un filtru  $3 \times 3$   
aplicat cu **stride 2**

$7 \Rightarrow \text{output } 3 \times 3$

# Dimensiunea unei hărți de activare

7

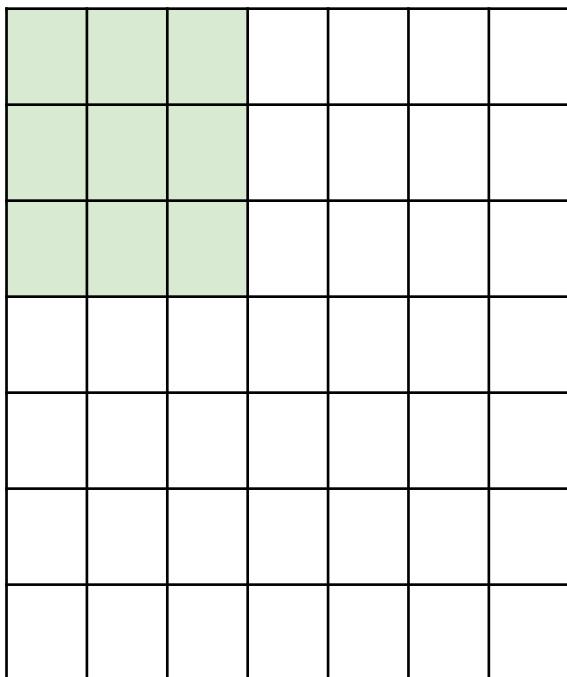


$7 \times 7$  input (numai 2D, fără a treia dimensiune)  
considerăm un filtru  $3 \times 3$   
aplicat cu **stride 3**?

7

# Dimensiunea unei hărți de activare

7



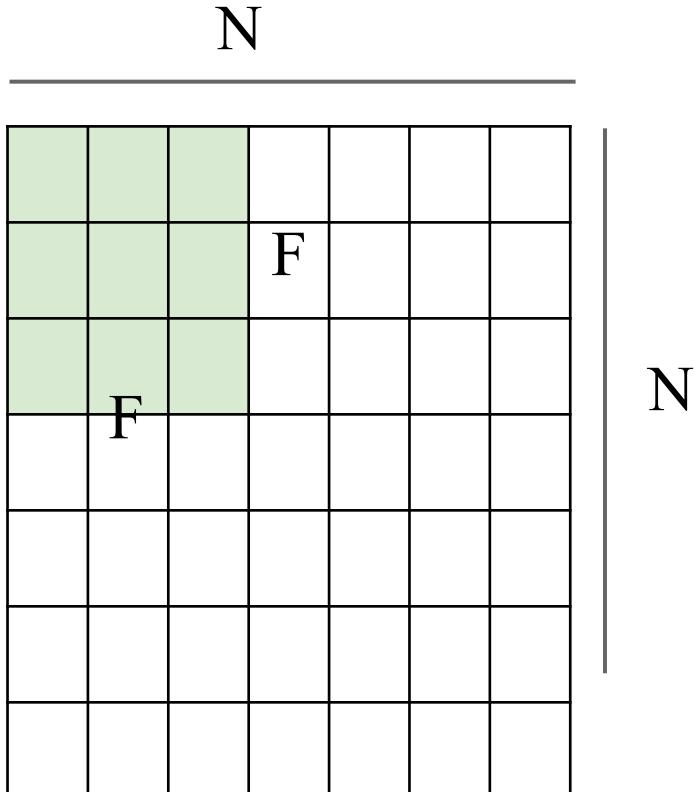
7×7 input (numai 2D, fără a treia dimensiune)  
considerăm un filtru  $3\times 3$   
aplicat cu **stride 3**?

7

**Nu se potrivește!**

Nu putem aplica un filtru  $3\times 3$  pe o  
imagine  $7\times 7$  folosind un stride de 3.

# Dimensiunea unei hărți de activare



Dimensiune output:  
 $(N - F) / \text{stride} + 1$

pentru  $N = 7$ ,  $F = 3$ :

$$\text{stride } 1 \Rightarrow (7 - 3)/1 + 1 = 5$$

$$\text{stride } 2 \Rightarrow (7 - 3)/2 + 1 = 3$$

$$\text{stride } 3 \Rightarrow (7 - 3)/3 + 1 = 2.33 :($$

# Dimensiunea unei hărți de activare

În practică deseori se adaugă (padding) 0 pe margine

0	0	0	0	0	0			
0								
0								
0								
0								

Pentru  $7 \times 7$  input  
filtru  $3 \times 3$ , aplicat cu **stride 1**  
**padding 1 pixel** la margine

⇒ **output = ?**

formula fără padding este:  
 $(N - F) / \text{stride} + 1$

# Dimensiunea unei hărți de activare

În practică deseori se adaugă (padding) 0 pe margine

0	0	0	0	0	0			
0								
0								
0								
0								

Pentru  $7 \times 7$  input  
filtru  $3 \times 3$ , aplicat cu **stride 1**  
**padding 1 pixel** la margine

⇒ **output = ?**

formula cu padding de dimensiune P este:  
$$(N - F + 2P) / \text{stride} + 1$$

# Dimensiunea unei hărți de activare

În practică deseori se adaugă (padding) 0 pe margine

0	0	0	0	0	0			
0								
0								
0								
0								

Pentru  $7 \times 7$  input  
filtru  $3 \times 3$ , aplicat cu **stride 1**  
**padding 1 pixel** la margine

⇒ **output**  $7 \times 7$

# Dimensiunea unei hărți de activare

În practică deseori se adaugă (padding) 0 pe margine

0	0	0	0	0	0			
0								
0								
0								
0								

Pentru  $7 \times 7$  input  
filtru  $3 \times 3$ , aplicat cu **stride 1**  
**padding 1 pixel** la margine

⇒ **output**  $7 \times 7$

În general, e destul de întâlnit să avem straturi convoluționale cu stride de 1, filtre  $F \times F$  și 0-padding de dimensiune  $(F-1)/2$ . Astfel se păstrează dimensiunea imaginii input / harta de activare.

e.g.  $F = 3 \Rightarrow$  0-padding de dimensiune 1  
 $F = 5 \Rightarrow$  0-padding de dimensiune 2  
 $F = 7 \Rightarrow$  0-padding de dimensiune 3