

# Rețele neuronale conoluționale. Recapitulare.

Radu Ionescu

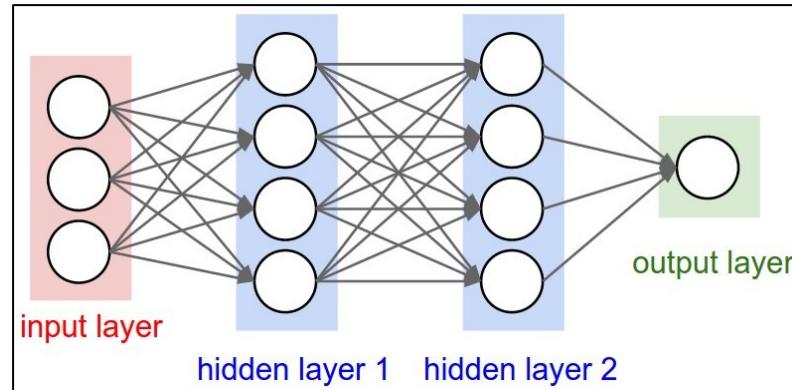
[raducu.ionescu@gmail.com](mailto:raducu.ionescu@gmail.com)

Facultatea de Matematică și Informatică  
Universitatea din București

# SGD cu mini-batch

Repetă:

1. Selectăm un mini-batch de exemple
2. Propagăm **înainte** prin graf pentru a obține pierdere
3. Propagăm **înapoi** pentru a calcula gradienții
4. Actualizăm ponderile pe baza gradienților calculați



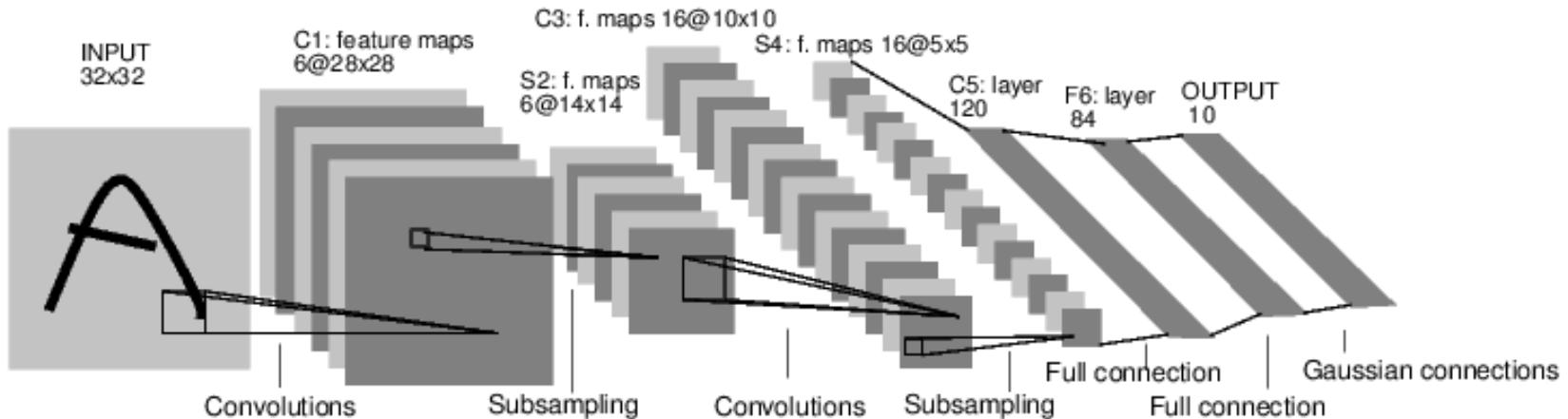
# Rețele neuronale sunt funcții universale de aproximare

- **Teorema Aproximării Universale:**

O rețea neuronală de tip feed-forward cu un strat ascuns având un număr finit de neuroni poate aproxima orice funcție continuă definită pe un subset compact din  $\mathbb{R}^n$ .

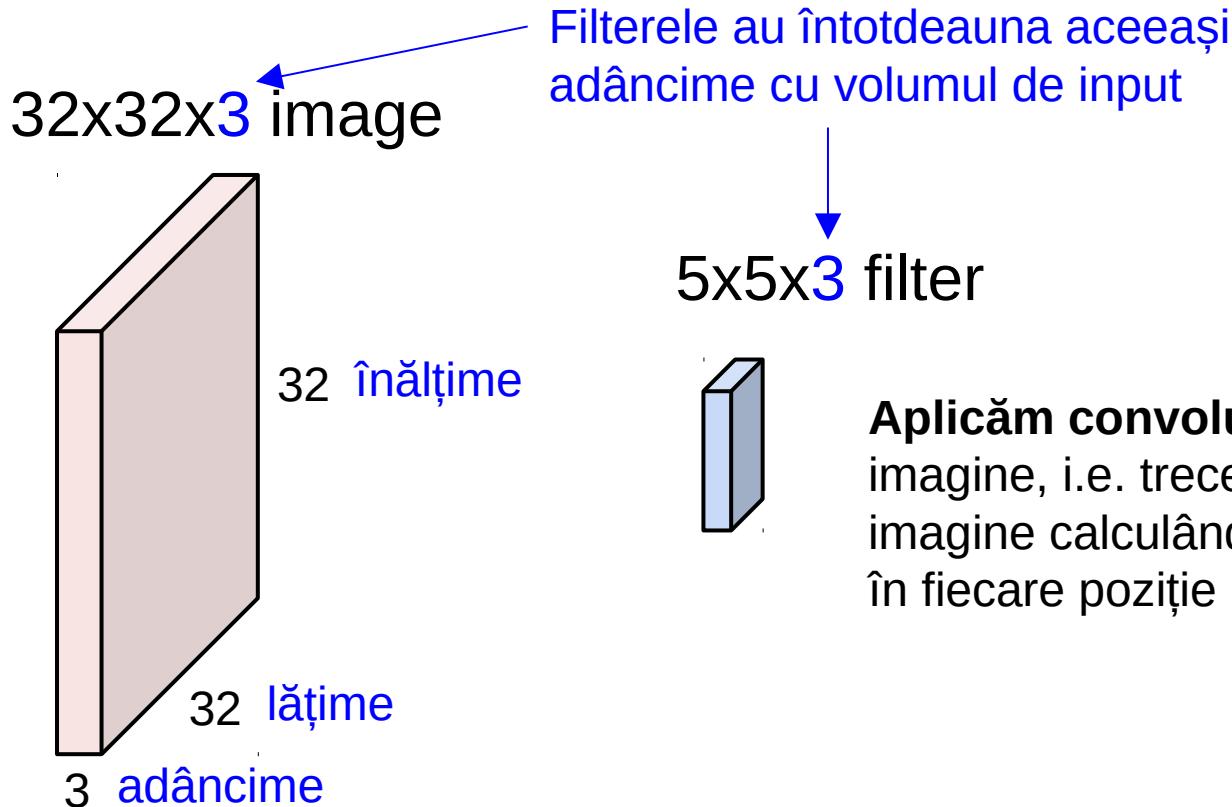
- Desi rețele neuronale cu două straturi (un strat ascuns) sunt funcții universale de aproximare, lățimea (numărul de perceptri) acestor rețele poate fi exponential de mare.
- În practică, preferăm rețele mai adânci (cu mai multe straturi)

# Rețele neuronale convolutionale



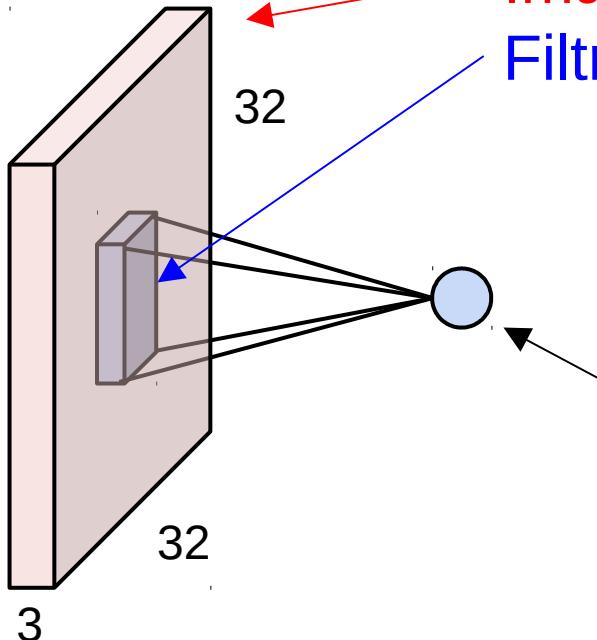
[LeNet-5, LeCun 1998]

# Stratul conoluțional



**Aplicăm conoluția** filtrului peste imagine, i.e. trecem filtrul peste imagine calculând produsul scalar în fiecare poziție

# Stratul conoluțional



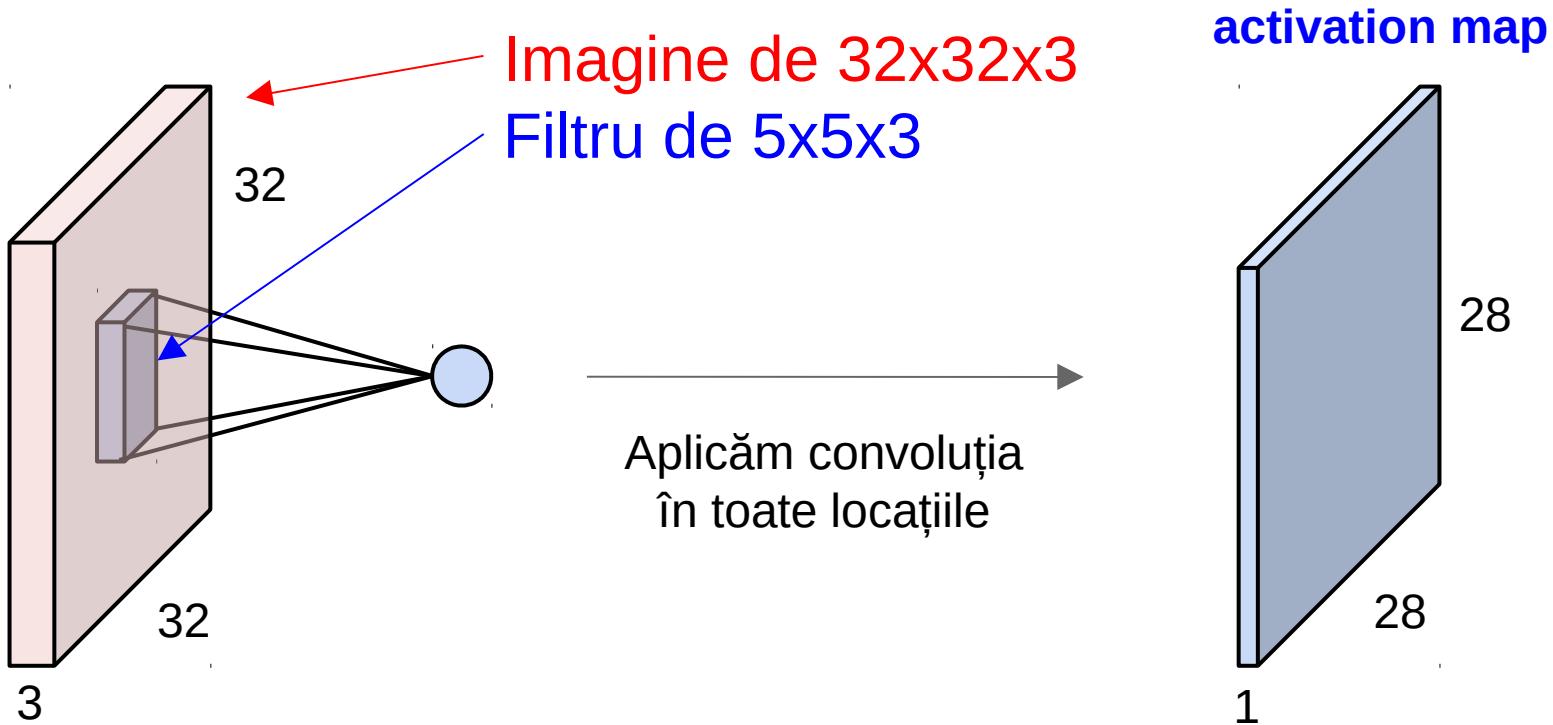
Imagine de  $32 \times 32 \times 3$   
Filtru de  $5 \times 5 \times 3$   $w$

un număr:

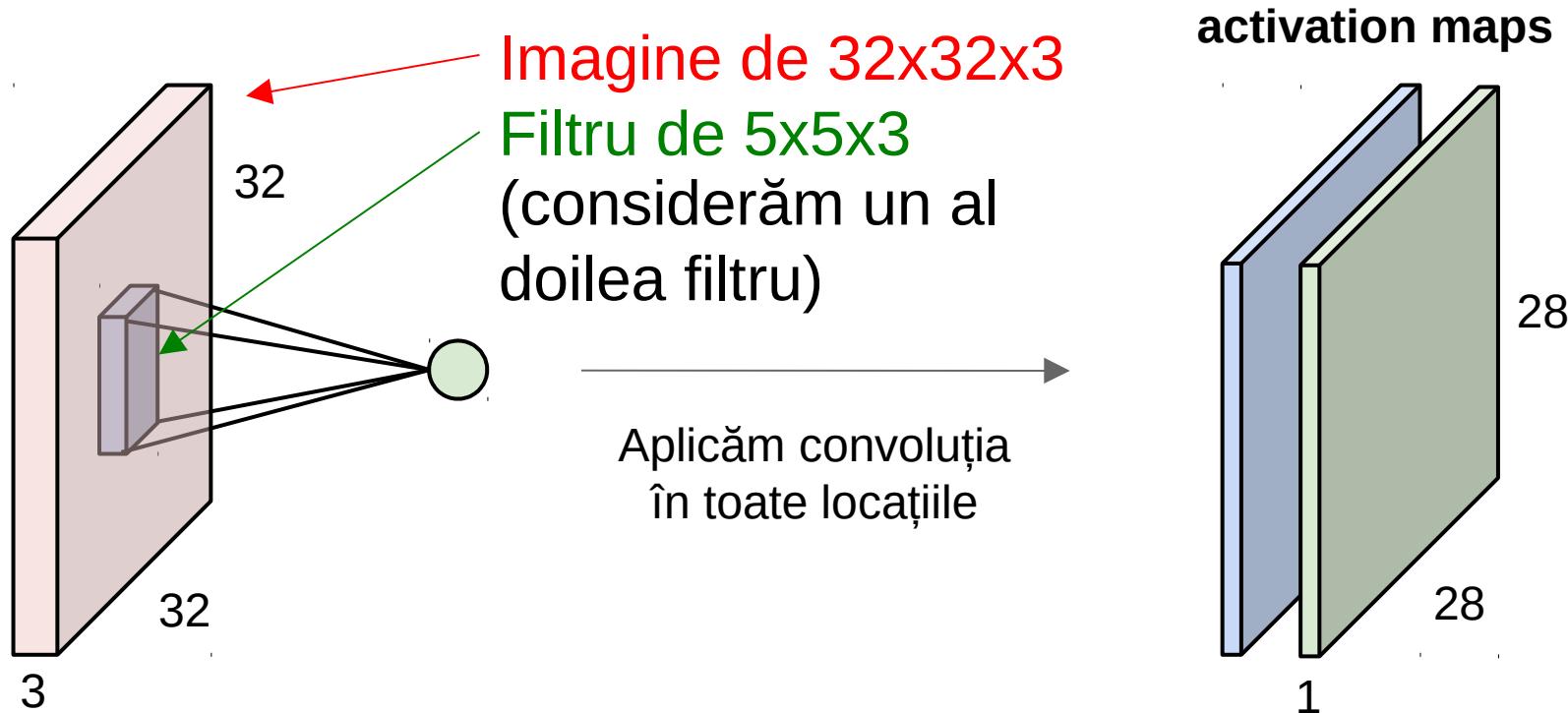
Rezultatul produsului scalar dintre filtru și o subimagine de  $5 \times 5 \times 3$  pixeli  
(i.e.  $5 \times 5 \times 3 =$  produs scalar pe 75 de componente + bias)

$$w^T x + b$$

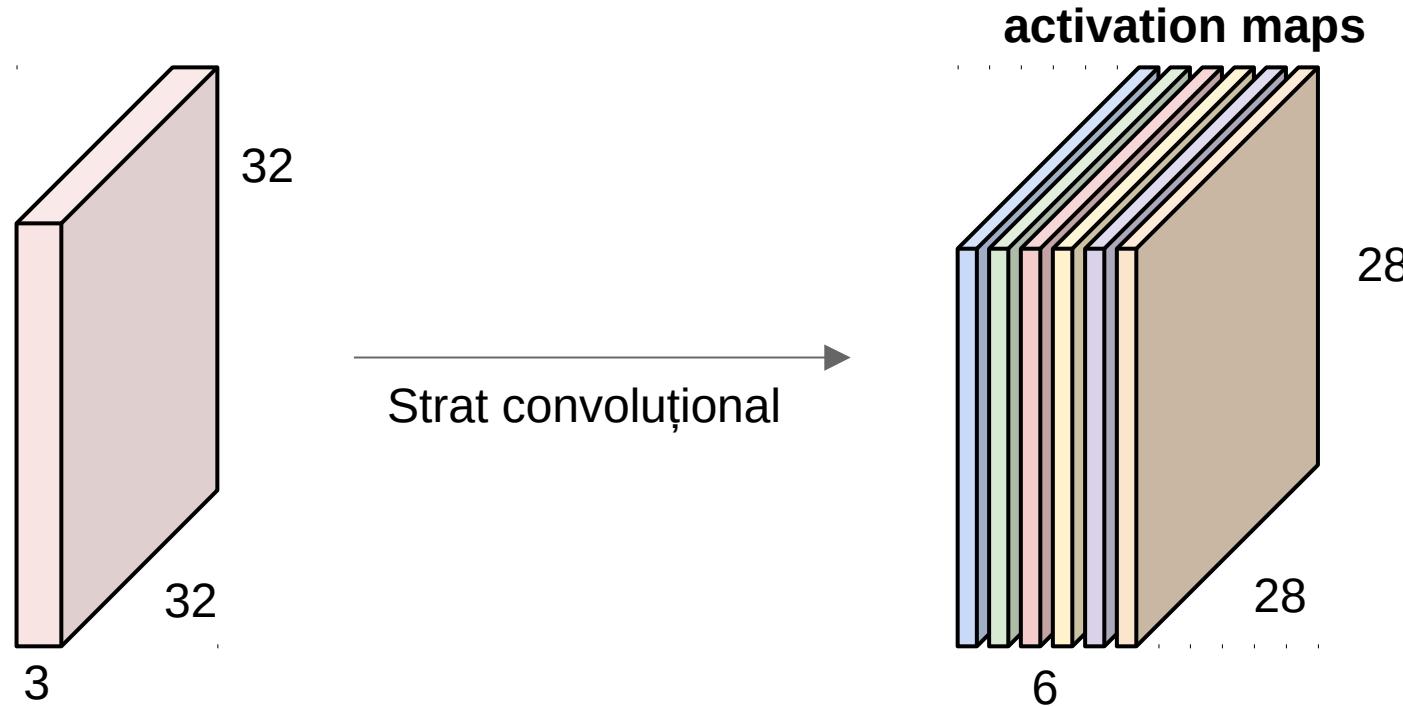
# Stratul convolutional



# Stratul convolutional

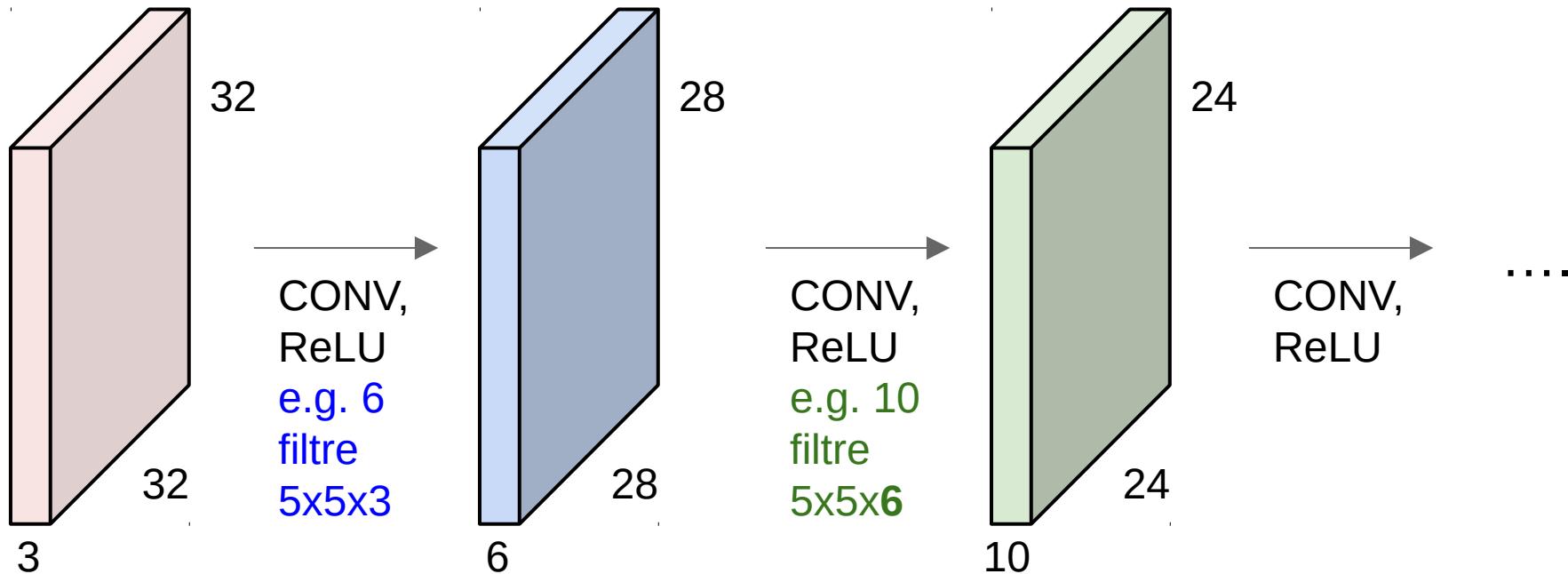


Un strat conoluțional este format din mai multe filtre (de exemplu 6)

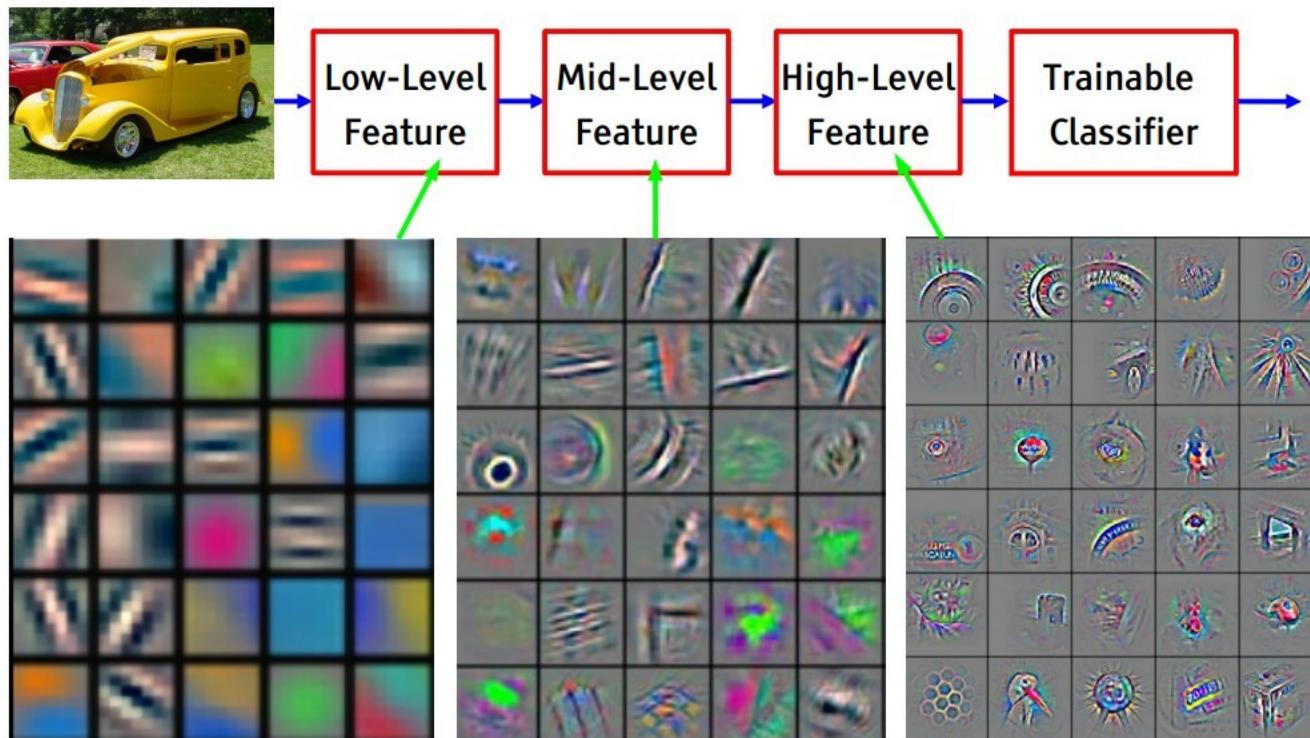


Concatenăm activările pentru a obține o nouă “imagine” de  $28 \times 28 \times 6$

În esență o rețea conoluțională (CNN sau ConvNet) este o formată dintr-o secvență de straturi conoluționale, între care interpunem funcții de activare



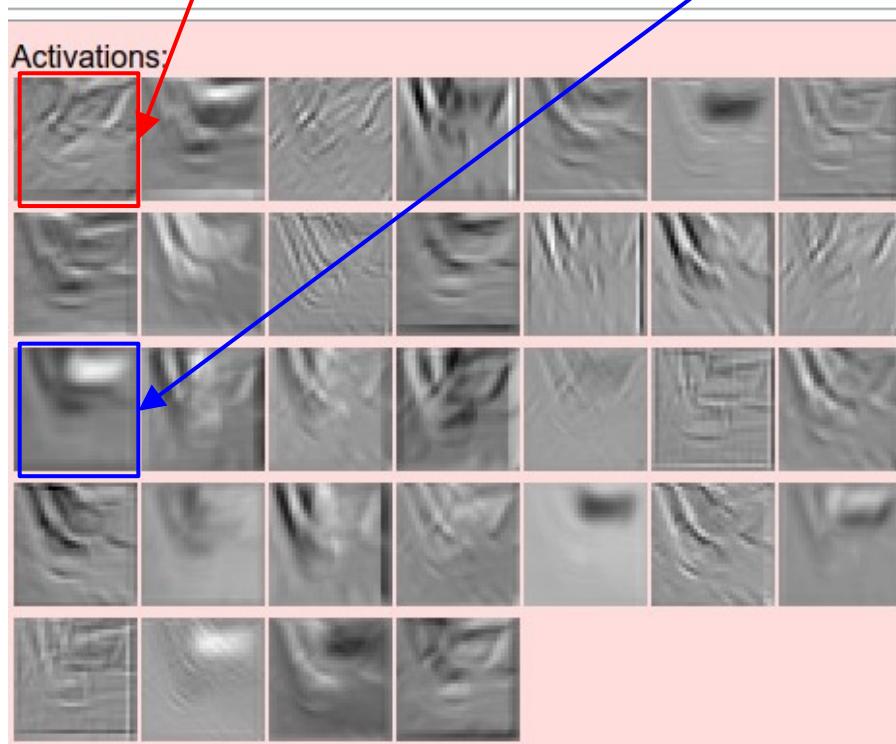
## Filtrele corespond unor trăsături ale obiectelor

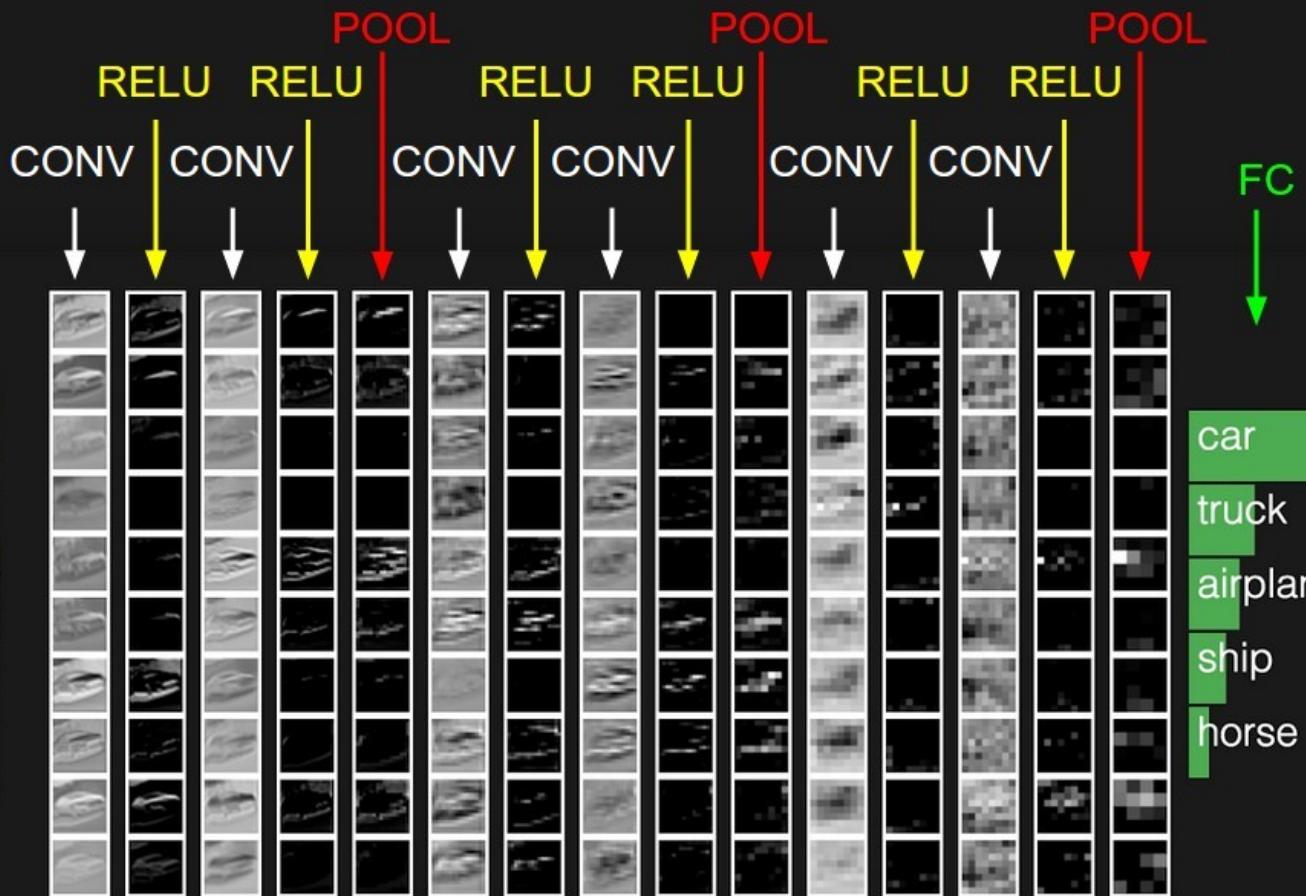


Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

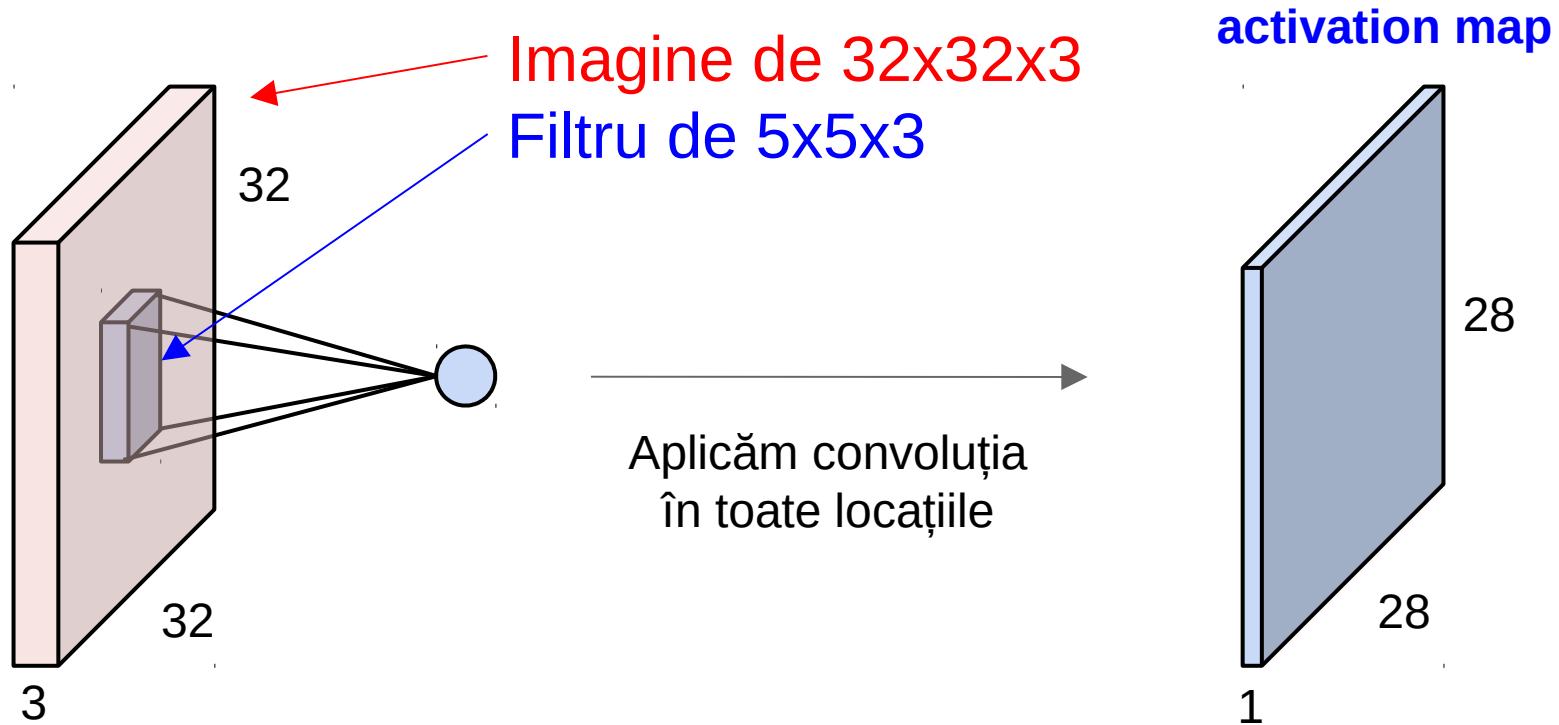


exemplu cu 32 de filtre 5x5



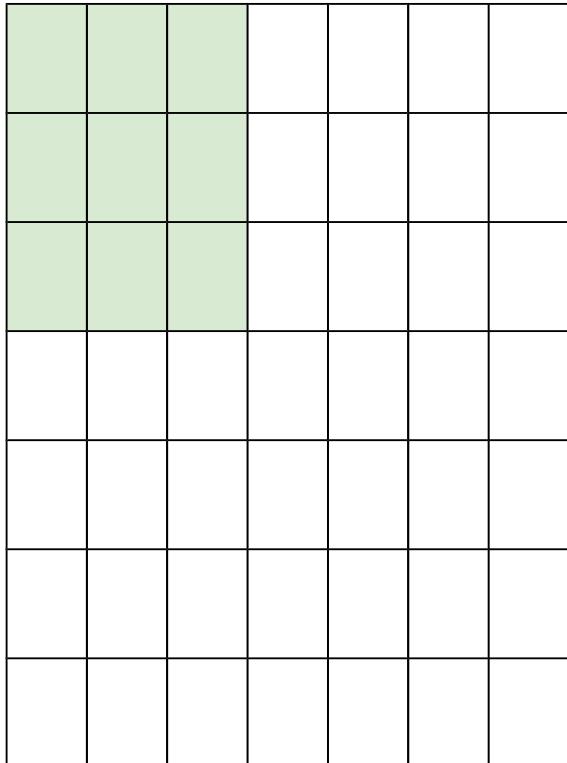


# Dimensiunea spațială a unei activări



# Dimensiunea spațială a unei activări

7

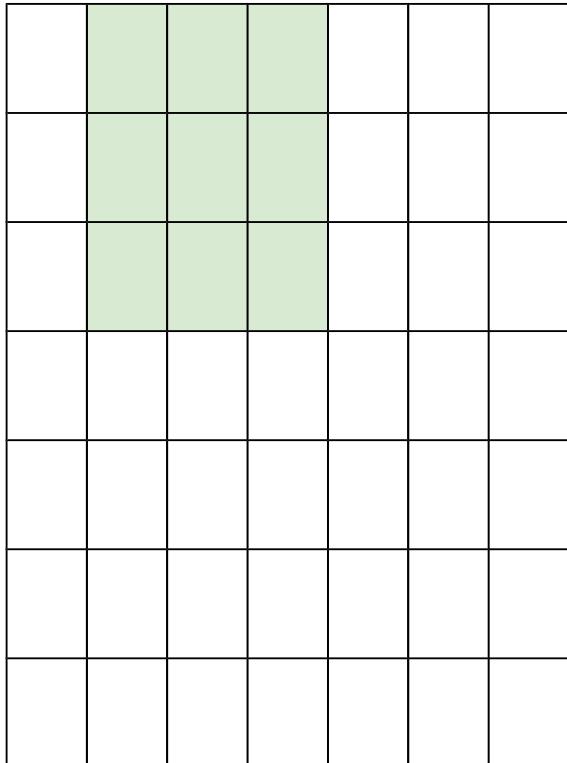


Imagine de  $7 \times 7$  (fără adâncime)  
Filtru de  $3 \times 3$

7

# Dimensiunea spațială a unei activări

7

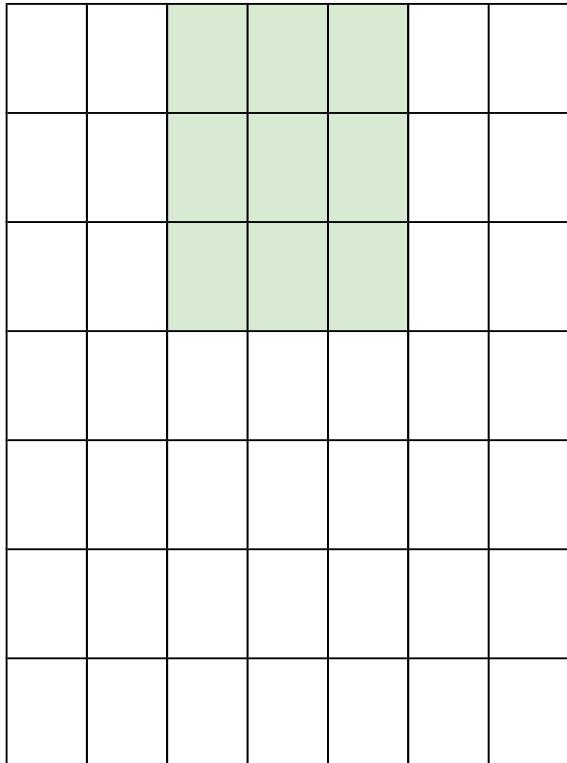


Imagine de  $7 \times 7$  (fără adâncime)  
Filtru de  $3 \times 3$

7

# Dimensiunea spațială a unei activări

7

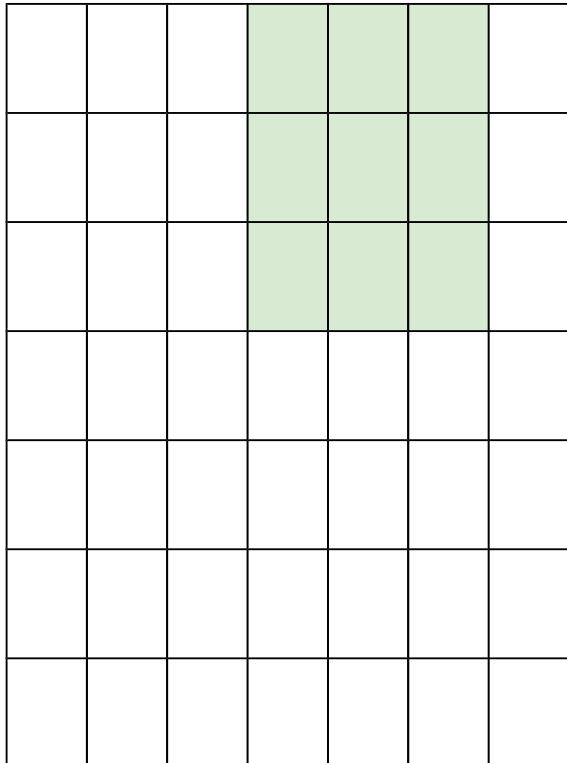


Imagine de  $7 \times 7$  (fără adâncime)  
Filtru de  $3 \times 3$

7

# Dimensiunea spațială a unei activări

7

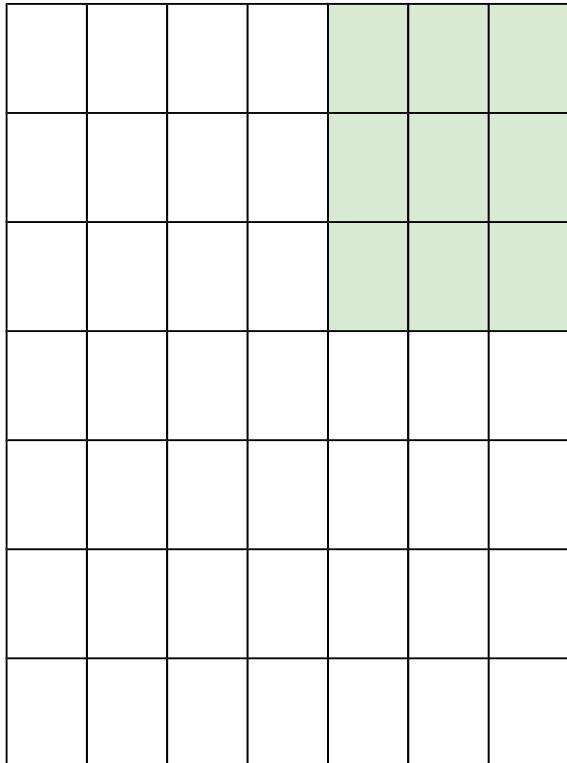


Imagine de  $7 \times 7$  (fără adâncime)  
Filtru de  $3 \times 3$

7

# Dimensiunea spațială a unei activări

7



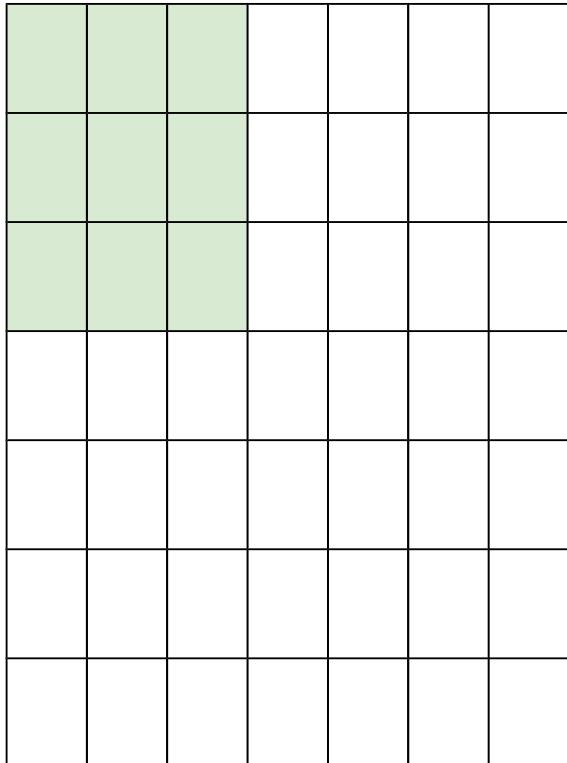
7

Imagine de  $7 \times 7$  (fără adâncime)  
Filtru de  $3 \times 3$

=> **Output de  $5 \times 5$**

# Dimensiunea spațială a unei activări

7

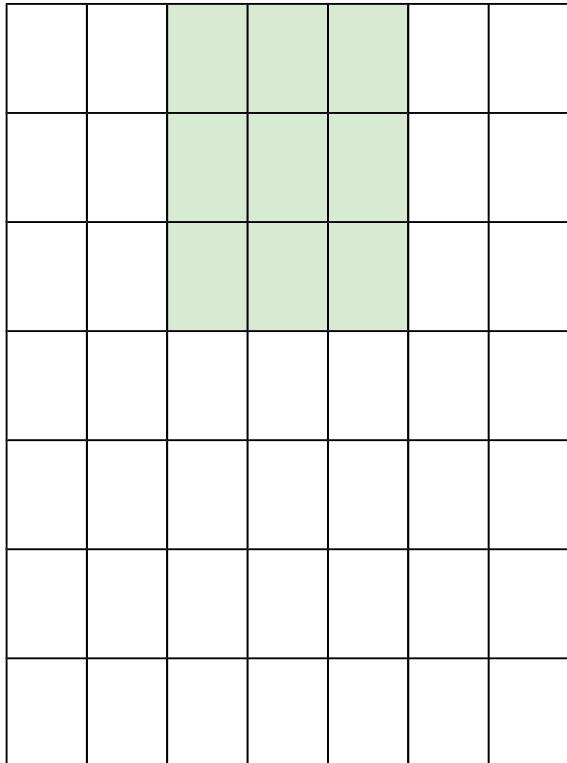


Imagine de  $7 \times 7$  (fără adâncime)  
Filtru de  $3 \times 3$  aplicat cu **stride 2**

7

# Dimensiunea spațială a unei activări

7

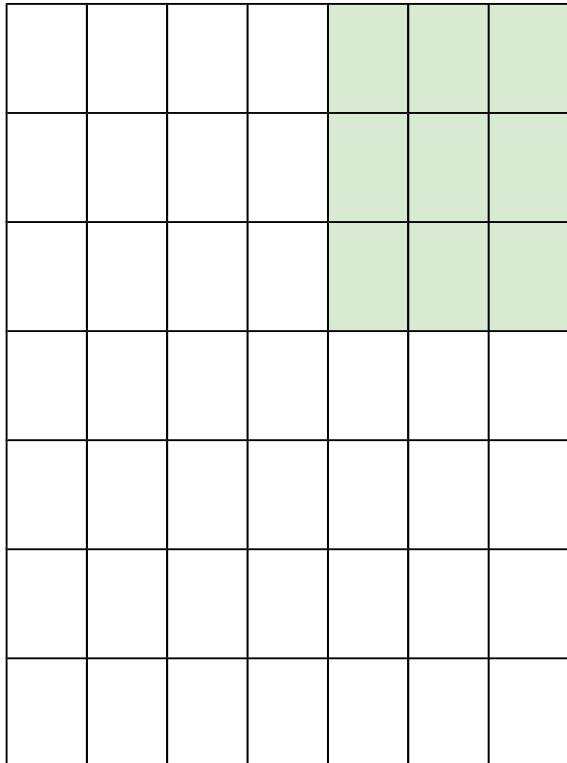


Imagine de  $7 \times 7$  (fără adâncime)  
Filtru de  $3 \times 3$  aplicat cu **stride 2**

7

# Dimensiunea spațială a unei activări

7

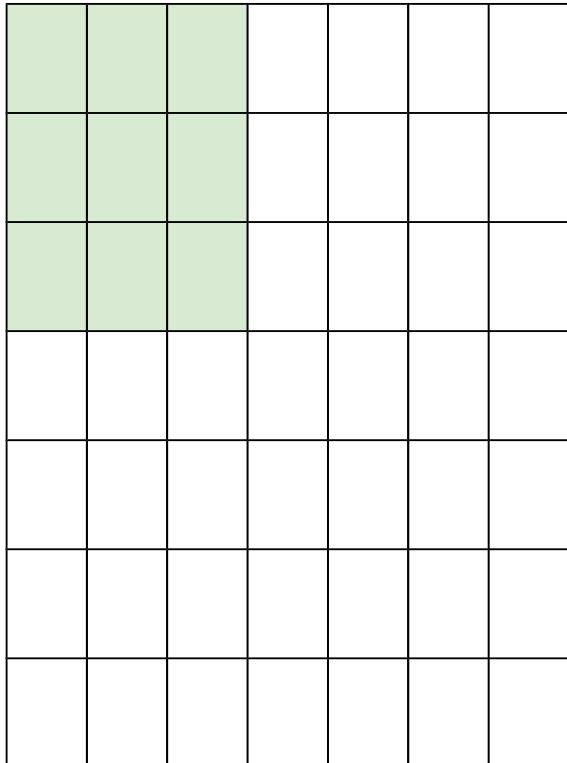


Imagine de  $7 \times 7$  (fără adâncime)  
Filtru de  $3 \times 3$  aplicat cu **stride 2**

=> Output de  $3 \times 3$

# Dimensiunea spațială a unei activări

7



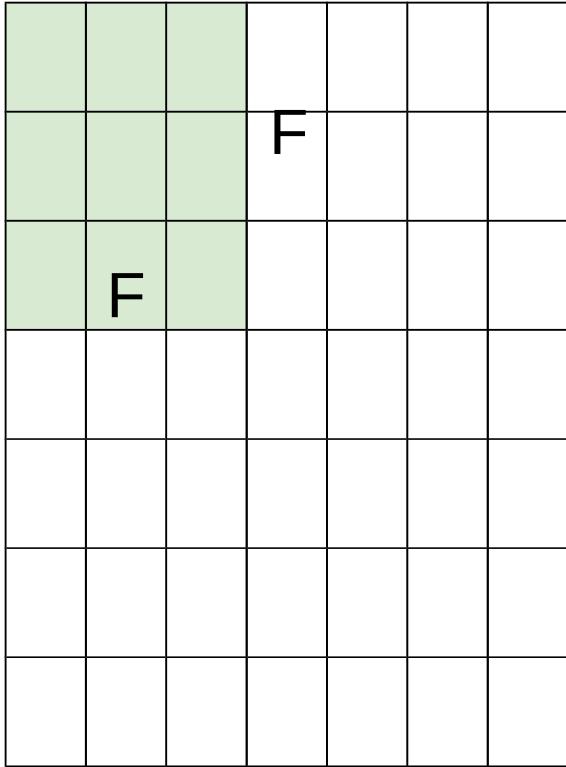
7

Imagine de  $7 \times 7$  (fără adâncime)  
Filtru de  $3 \times 3$  aplicat cu **stride 3**?

**Nu se potrivește!**

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

N



Mărimea activării:  
 **$(N - F) / \text{stride} + 1$**

N

e.g.  $N = 7, F = 3:$

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

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

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

# În practică: deobicei imaginea se bordează cu 0

0	0	0	0	0	0		
0							
0							
0							
0							

e.g. imagine de 7x7

filtru **3x3**, aplicat cu **stride 1**

**bordăm cu 1 pixel** de valoare 0

Q: Cum arată activarea?

=> **Output de 7x7**

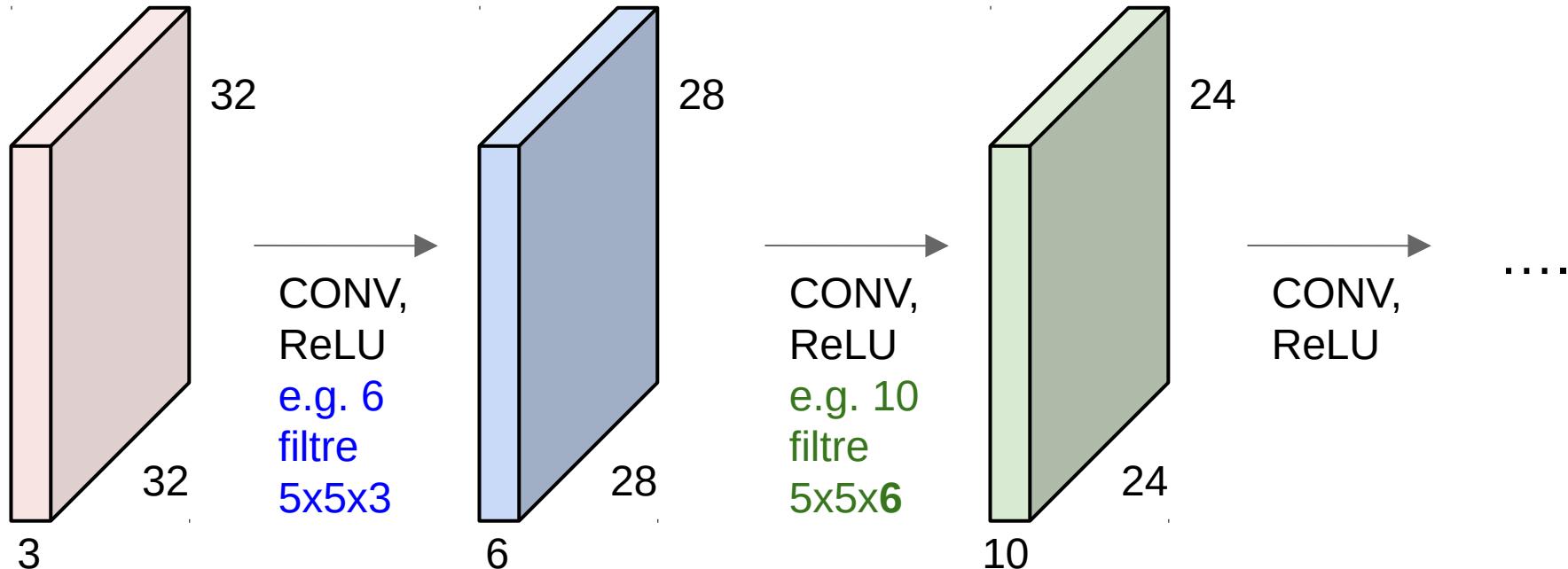
În general, se folosesc straturi conveționale cu stride 1, cu filtre de dimensiune  $F \times F$ , și bordură de  $(F-1)/2$ . (menține dimensiunea imaginii de input)

e.g.  $F = 3 \Rightarrow$  bordăm cu 1 pixel (valoare 0)

$F = 5 \Rightarrow$  bordăm cu 2 pixeli (valoare 0)

$F = 7 \Rightarrow$  bordăm cu 3 pixeli (valoare 0)

Aplicând convoluții cu filtre de  $5 \times 5$  în mod repetat pe un input de  $32 \times 32$  micșorează volumul ( $32 \Rightarrow 28 \Rightarrow 24 \dots$ )  
Micșorarea prea rapidă a volumului nu produce rezultate optime

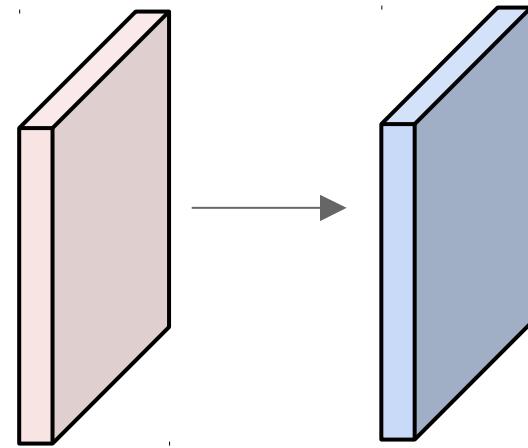


Exemplu:

Volum de input: **32x32x3**

10 filtre de 5x5 cu stride 1, bordură 2

Mărimea volumului de output: ?



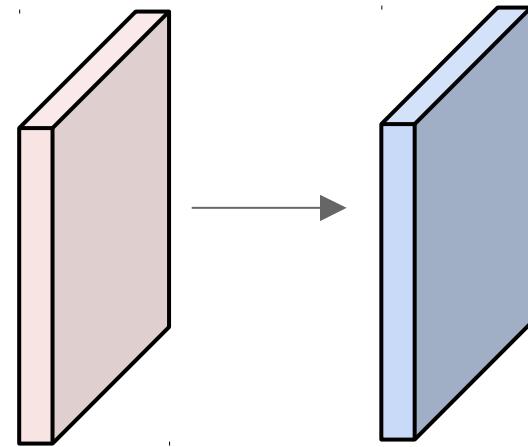
Exemplu:

Volum de input: **32x32x3**

**10** filtre de **5x5** cu stride **1**, bordură **2**

Mărimea volumului de output:

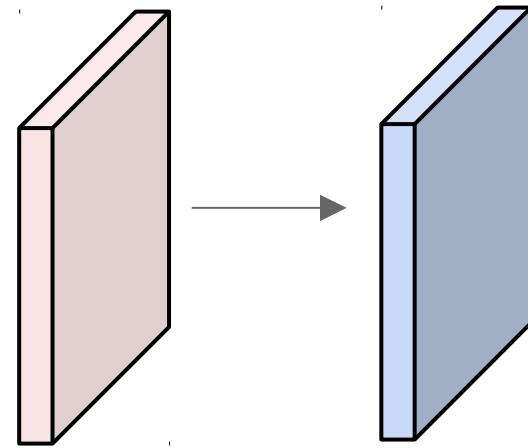
$(32+2*2-5)/1+1 = 32$ , deci volumul este **32x32x10**



Exemplu:

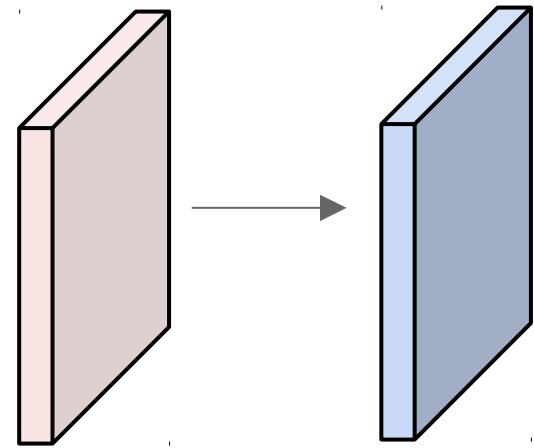
Volum de input: **32x32x3**

10 filtre de 5x5 cu stride 1, bordură 2



Numărul de parametrii al acestui strat: ?

## **Exemplu:**



Volum de input: **32x32x3**

**10 filtre de 5x5 cu stride 1, bordură 2**

Numărul de parametrii al acestui strat: ?

Fiecare filtru are  $5 \times 5 \times 3 + 1 = 76$

**parametrii** (+1 pentru bias)

$$\Rightarrow 76 * 10 = 760$$

## Setări comune:

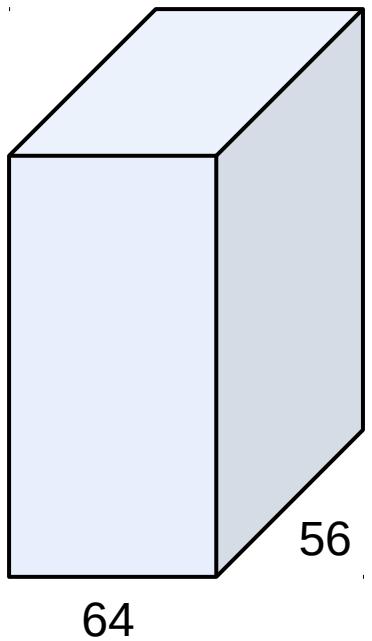
**Summary.** To summarize, the Conv Layer:

- Accepts a volume of size  $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters  $K$ ,
  - their spatial extent  $F$ ,
  - the stride  $S$ ,
  - the amount of zero padding  $P$ .
- Produces a volume of size  $W_2 \times H_2 \times D_2$  where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$  (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces  $F \cdot F \cdot D_1$  weights per filter, for a total of  $(F \cdot F \cdot D_1) \cdot K$  weights and  $K$  biases.
- In the output volume, the  $d$ -th depth slice (of size  $W_2 \times H_2$ ) is the result of performing a valid convolution of the  $d$ -th filter over the input volume with a stride of  $S$ , and then offset by  $d$ -th bias.

$K = (\text{puteri ale lui } 2, \text{ e.g. } 32, 64, 128, 512)$

- $F = 3, S = 1, P = 1$
- $F = 5, S = 1, P = 2$
- $F = 5, S = 2, P = ?$  (orice se încadrează)
- $F = 1, S = 1, P = 0$

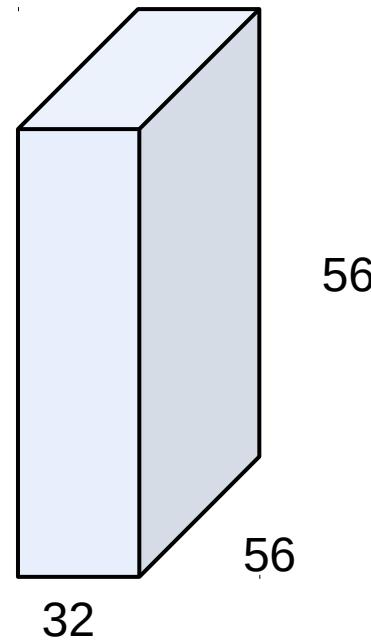
# Are sens să folosim chiar filtre de 1x1



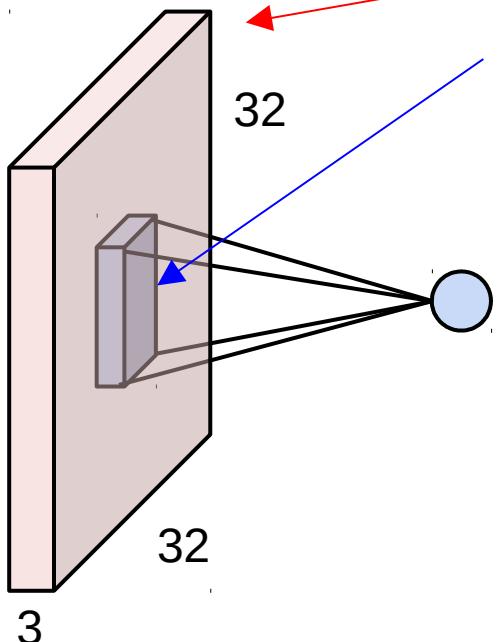
1x1 convoluție  
cu 32 de filtre

→

( fiecare filtru are  
dimensiunea  $1 \times 1 \times 64$  și  
calculează produsul  
scalar pe adâncime)



# Stratul convolutional din punct de vedere biologic

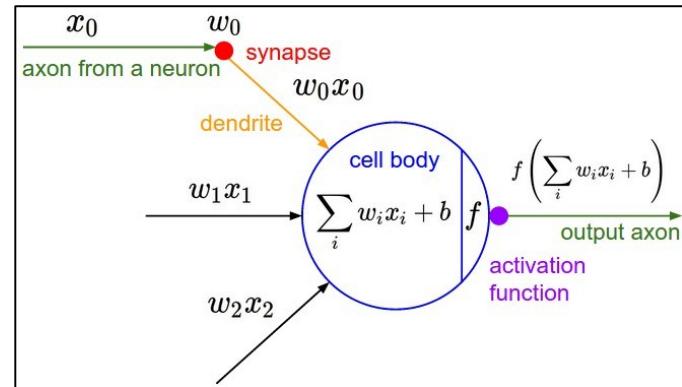


Imagine de  $32 \times 32 \times 3$   
Filtru de  $5 \times 5 \times 3$   $w$

un număr:

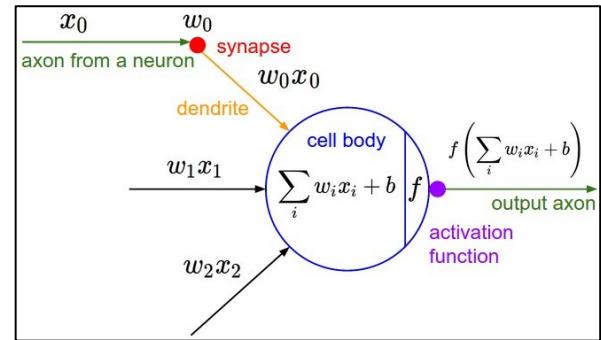
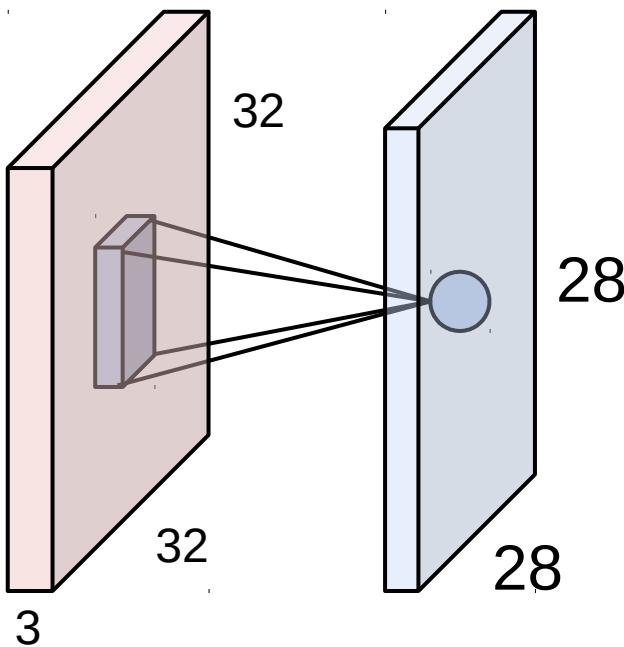
Rezultatul produsului scalar dintre filtru  
și o subimagine de  $5 \times 5 \times 3$  pixeli  
(i.e.  $5 \times 5 \times 3 =$  produs scalar pe 75 de  
componente + bias)

$$w^T x + b$$



...este doar un neuron cu  
conexiuni locale

# Stratul convecțional din punct de vedere biologic

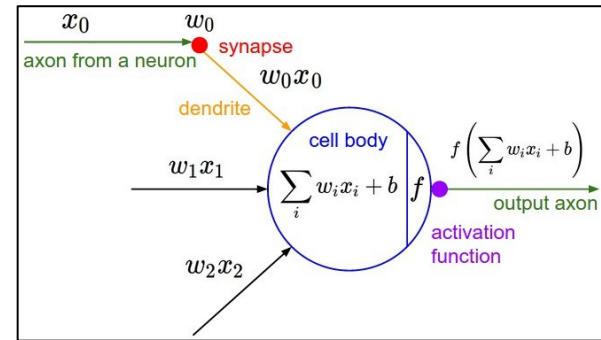
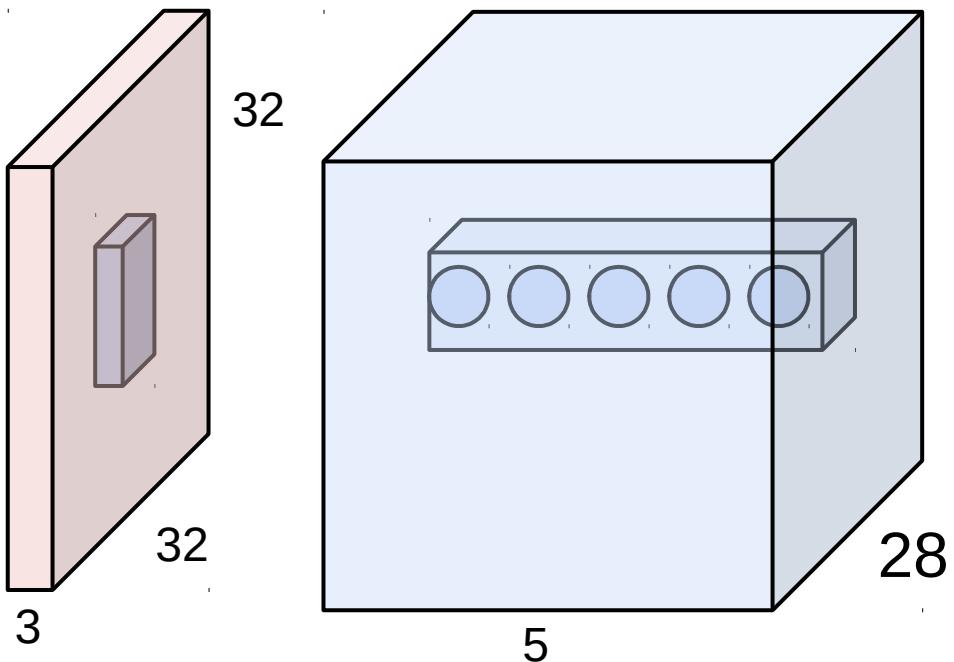


Un activation map este matrice cu output-urile a  $28 \times 28$  neuroni:

1. Fiecare este conectat la o regiune mică din input
2. Toții neuroni au aceeși parametrii

filtru  $5 \times 5$  = câmp receptiv de  $5 \times 5$  pentru fiecare neuron

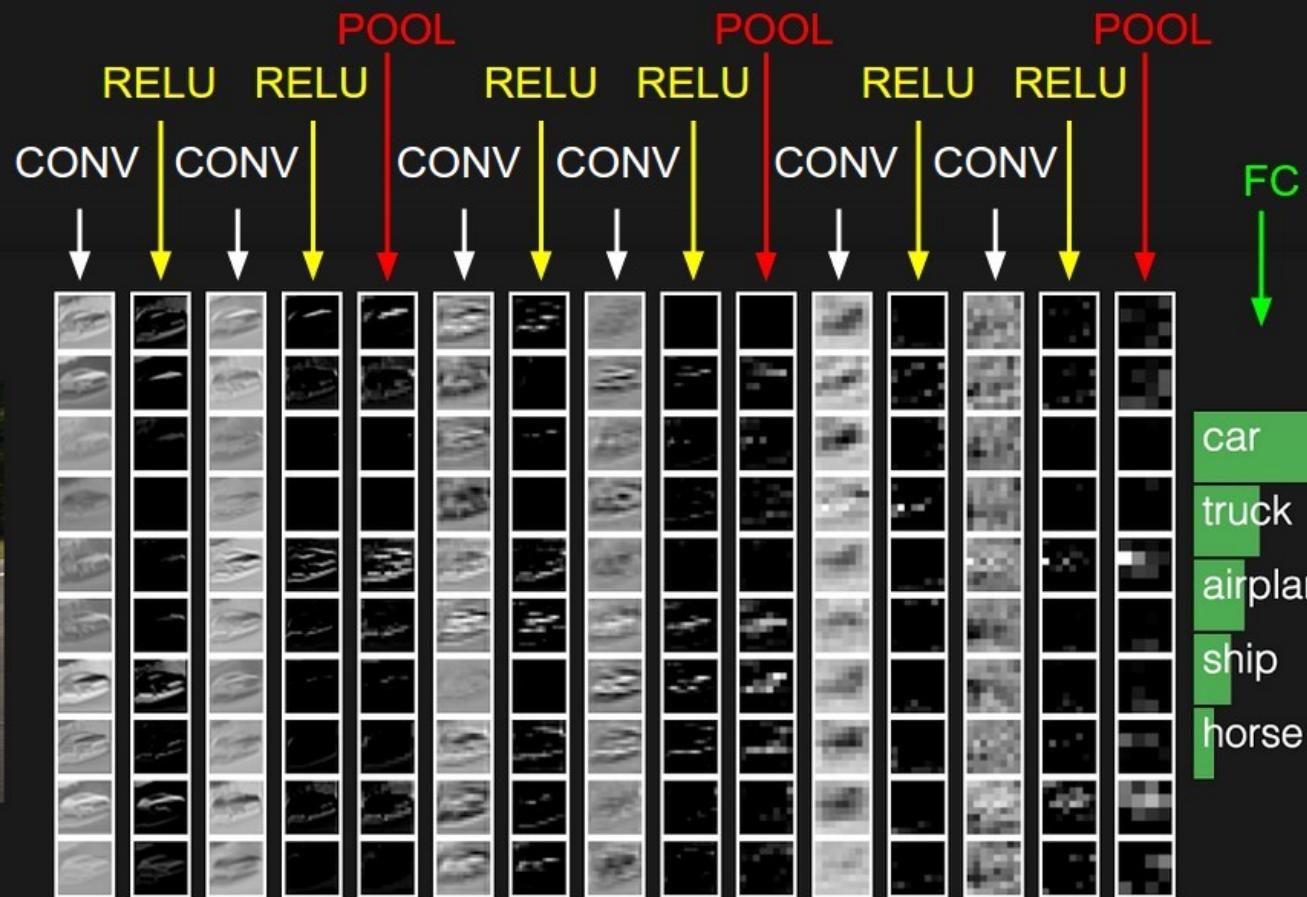
# Stratul convecțional din punct de vedere biologic



E.g. cu 5 filtre,  
stratul convecțional este format  
din neuroni aranjați într-un grid  
( $28 \times 28 \times 5$ )

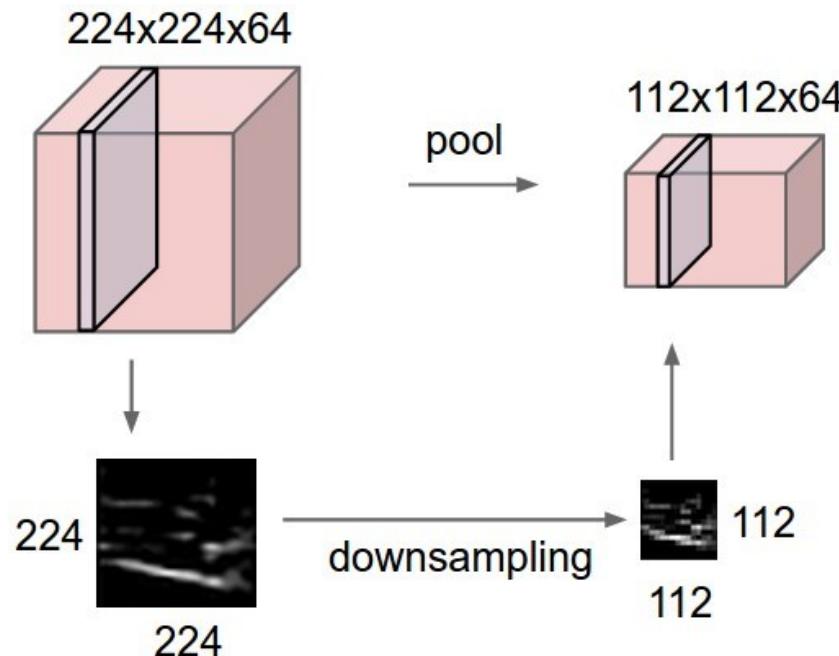
Pentru o regiune vor fi 5 neuroni  
diferiți (toți primesc același input)

Mai avem de discutat despre două tipuri de straturi: POOL, FC



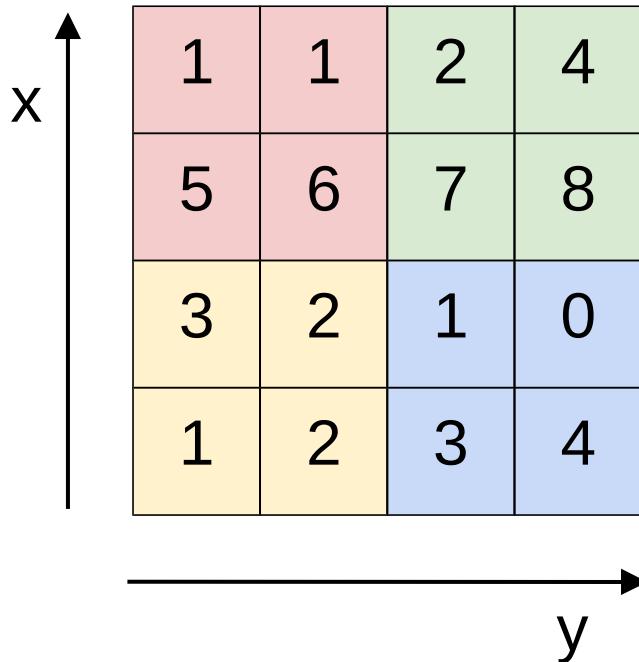
# Stratul de pooling

- Reduce reprezentarea, permite o utilizare mai ușoară
- Operează pe fiecare activation map în parte:



# MAX Pooling

Un activation map



max pooling cu filtru  
de 2x2 și stride 2

The result of the max pooling operation is a 2x2 matrix:

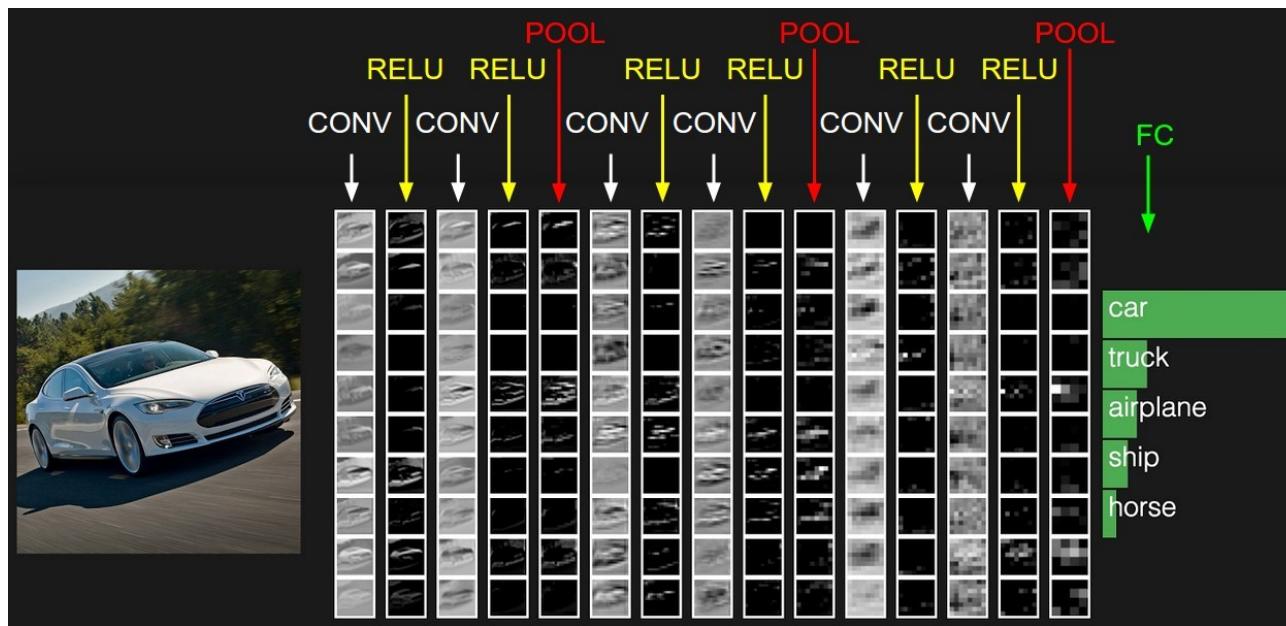
6	8
3	4

Setări comune:

- Accepts a volume of size  $W_1 \times H_1 \times D_1$   $F = 2, S = 2$
- Requires three hyperparameters:
  - their spatial extent  $F$ ,
  - the stride  $S$ , $F = 3, S = 2$
- Produces a volume of size  $W_2 \times H_2 \times D_2$  where:
  - $W_2 = (W_1 - F)/S + 1$
  - $H_2 = (H_1 - F)/S + 1$
  - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layers

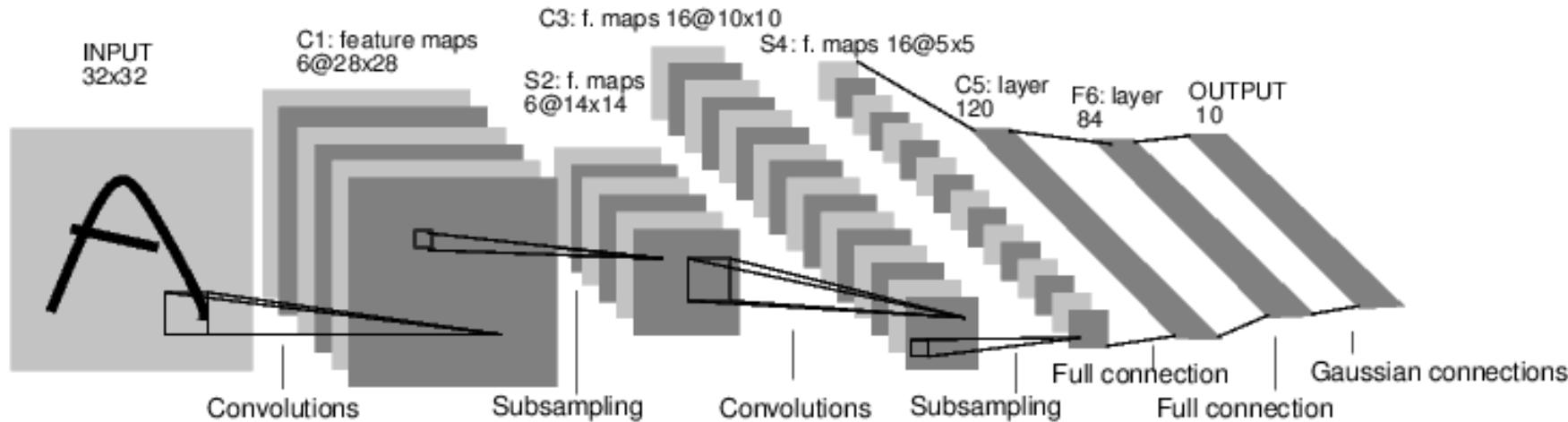
# Straturi cu conexiuni complete (FC layer)

- Contine neuroni conectați cu întregul volum de input, ca în rețelele neuronale obișnuite



# Studiu de caz: LeNet-5

[LeCun et al., 1998]



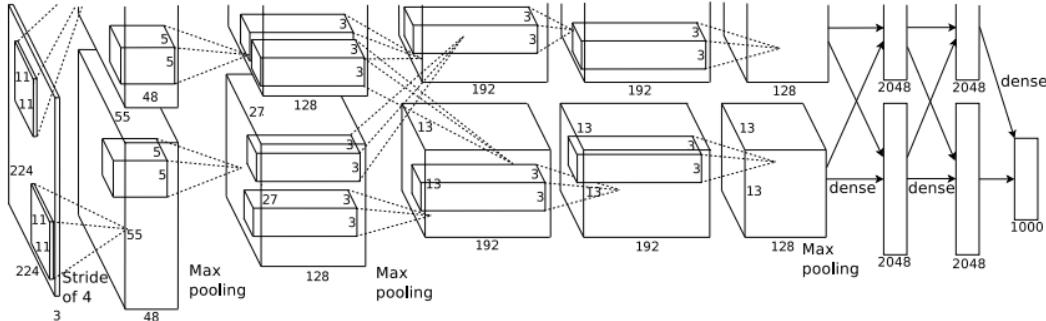
Straturi convełuonionale cu filtr de 5x5, aplicate cu stride 1

Straturi de pooling cu filtr de 2x2, aplicate cu stride 2

Arhitectura este [CONV-POOL-CONV-POOL-CONV-FC]

# Studiu de caz: AlexNet

[Krizhevsky et al. 2012]



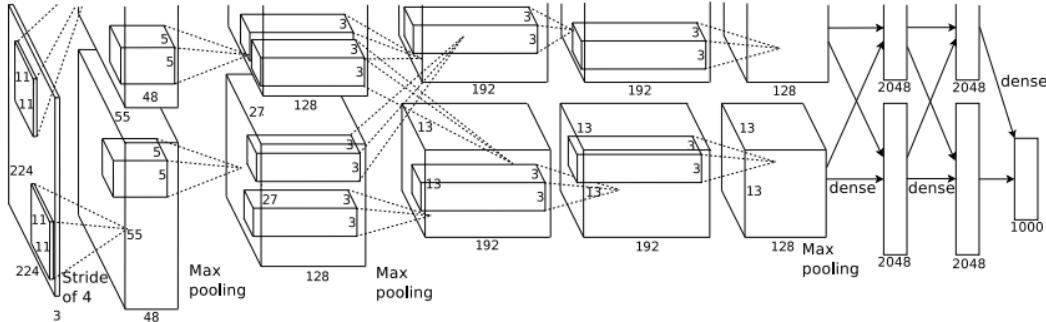
Input: imagini de 227x227x3

**Primul strat (CONV1): 96 filtre de 11x11, aplicate cu stride 4**

Q: Care este mărimea volumului de output? Hint:  $(227-11) / 4 + 1 = 55$

# Studiu de caz: AlexNet

[Krizhevsky et al. 2012]



Input: imagini de 227x227x3

**Primul strat (CONV1): 96 filtre de 11x11, aplicate cu stride 4**

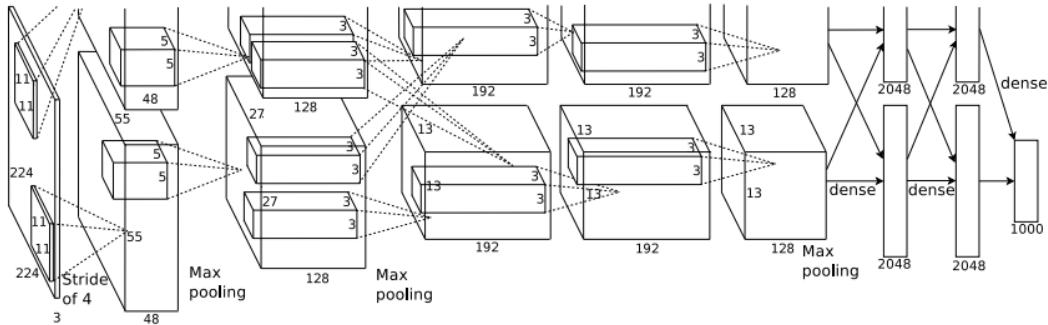
=>

Volumul de output este **[55x55x96]**

Q: Care este numărul parametrilor din acest strat?

# Studiu de caz: AlexNet

[Krizhevsky et al. 2012]



Input: imagini de 227x227x3

**Primul strat (CONV1): 96 filtre de 11x11, aplicate cu stride 4**

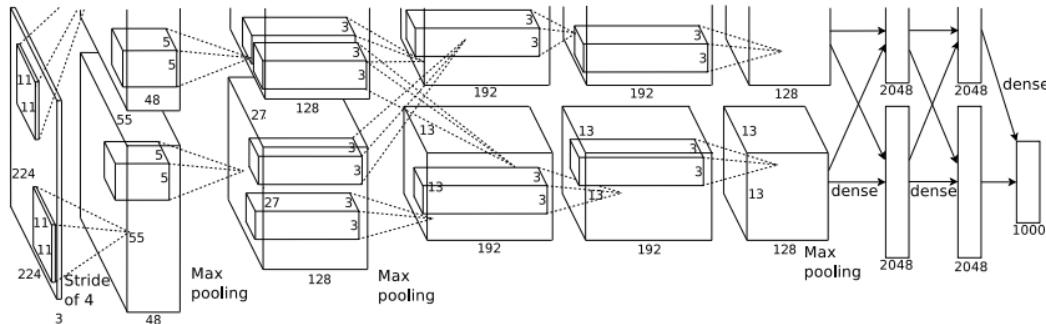
=>

Volumul de output este **[55x55x96]**

Parametrii:  $(11 \times 11 \times 3) \times 96 = 35K$

# Studiu de caz: AlexNet

[Krizhevsky et al. 2012]



Input: imagini de 227x227x3

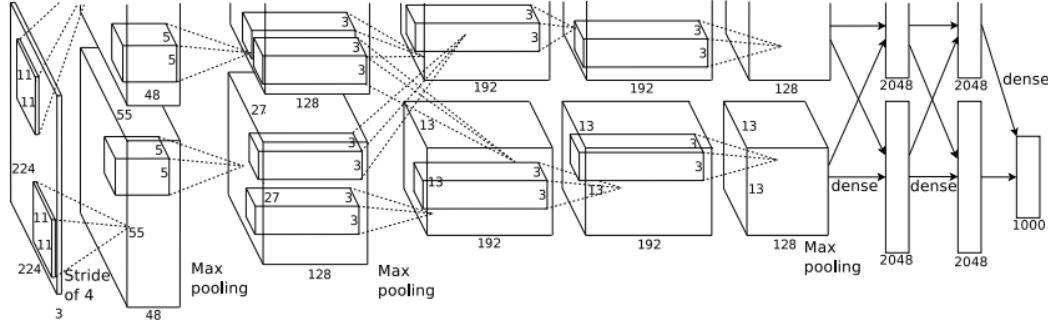
CONV1: 55x55x96

Al doilea strat (POOL1): filtru de 3x3 aplicate cu stride 2

Q: Care este mărimea volumului de output? Hint:  $(55-3) / 2 + 1 = 27$

# Studiu de caz: AlexNet

[Krizhevsky et al. 2012]



Input: imagini de 227x227x3

CONV1: 55x55x96

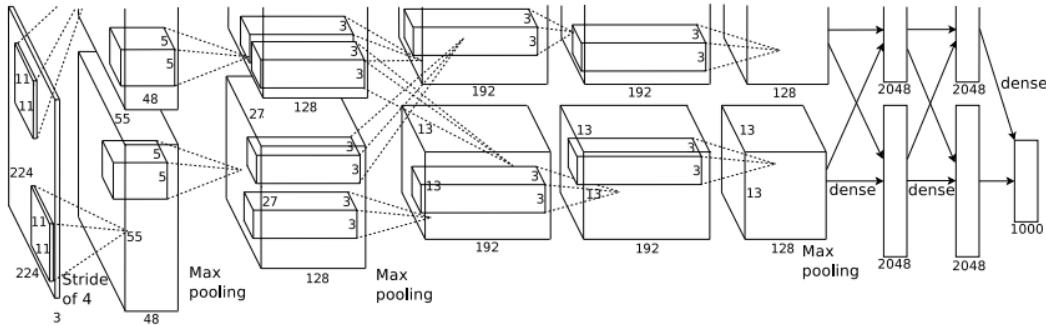
Al doilea strat (POOL1): filtre de 3x3 aplicate cu stride 2

Volumul de output este **[27x27x96]**

Q: Care este numărul parametrilor din acest strat?

# Studiu de caz: AlexNet

[Krizhevsky et al. 2012]



Input: imagini de 227x227x3

CONV1: 55x55x96

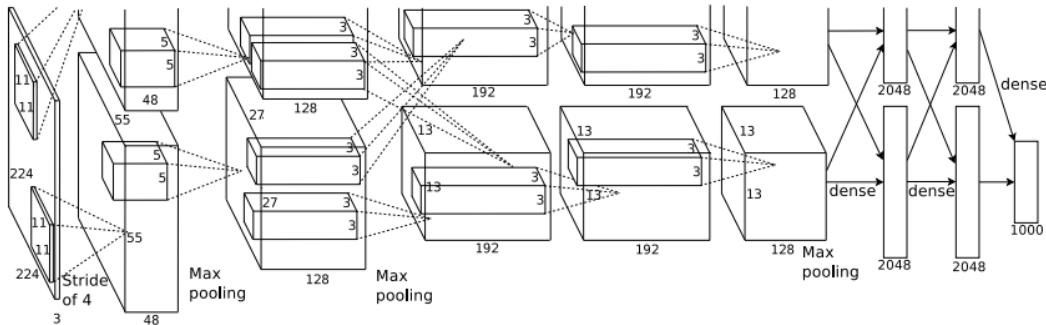
Al doilea strat (POOL1): filtru de 3x3 aplicate cu stride 2

Volumul de output este **[27x27x96]**

Parametrii: 0

# Studiu de caz: AlexNet

[Krizhevsky et al. 2012]



Input: imagini de 227x227x3

CONV1: 55x55x96

POOL1: 27x27x96

...

# Studiu de caz: AlexNet

[Krizhevsky et al. 2012]

Arhitectura completă AlexNet:

[227x227x3] INPUT

[55x55x96] CONV1: 96 filtre de 11x11 cu stride 4, fără bordură

[27x27x96] MAX POOL1: filtre de 3x3 cu stride 2

[27x27x96] NORM1: strat de normalizare

[27x27x256] CONV2: 256 filtre de 5x5 cu stride 1, bordură 2

[13x13x256] MAX POOL2: filtre de 3x3 cu stride 2

[13x13x256] NORM2: strat de normalizare

[13x13x384] CONV3: 384 filtre de 3x3 cu stride 1, bordură 1

[13x13x384] CONV4: 384 filtre de 3x3 cu stride 1, bordură 1

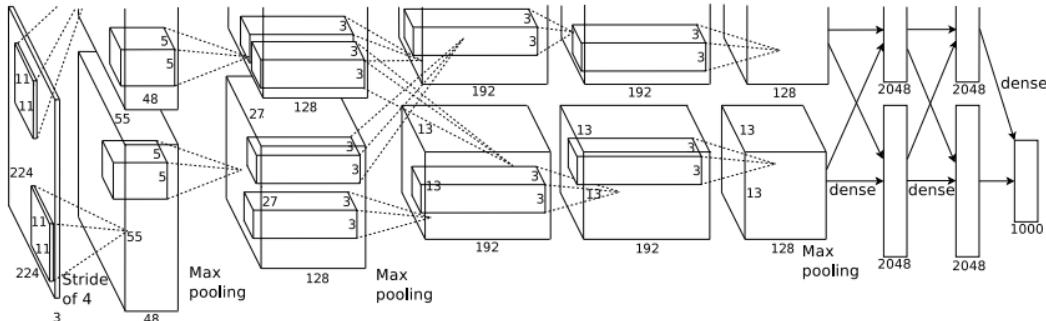
[13x13x256] CONV5: 256 filtre de 3x3 cu stride 1, bordură 1

[6x6x256] MAX POOL3: filtre de 3x3 cu stride 2

[4096] FC6: 4096 neuroni

[4096] FC7: 4096 neuroni

[1000] FC8: 1000 neuroni (scoruri pentru 1000 de clase)



## Detalii:

- Prima utilizare a ReLU
- Straturi de normalizare (nu se mai folosesc)
- Augmentarea datelor
- Dropout 0.5
- Mărimea batch-ului 128
- SGD cu moment 0.9
- Rata de învățare 0.01, împărțită la 10 atunci când acuratețea de validare atinge un platou
- Ansamblu de 7 CNN-uri: 18.2% => 15.4%

# Studiu de caz: VGGNet

[Simonyan and Zisserman, 2014]

Doar CONV de 3x3 cu stride 1, bordură 1  
și MAX POOL de 2x2 cu stride 2

Cel mai bun model

11.2% eroare top 5 la ILSVRC 2013

=>

7.3% eroare top 5

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 2: Number of parameters (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

(fără a număra bias-urile)

INPUT: [224x224x3] memorie:  $224 \times 224 \times 3 = 150K$  parametrii: 0

CONV3-64: [224x224x64] memorie:  $224 \times 224 \times 64 = 3.2M$  parametrii:  $(3 \times 3 \times 3) \times 64 = 1,728$

CONV3-64: [224x224x64] memorie:  $224 \times 224 \times 64 = 3.2M$  parametrii:  $(3 \times 3 \times 64) \times 64 = 36,864$

POOL2: [112x112x64] memorie:  $112 \times 112 \times 64 = 800K$  parametrii: 0

CONV3-128: [112x112x128] memorie:  $112 \times 112 \times 128 = 1.6M$  parametrii:  $(3 \times 3 \times 64) \times 128 = 73,728$

CONV3-128: [112x112x128] memorie:  $112 \times 112 \times 128 = 1.6M$  parametrii:  $(3 \times 3 \times 128) \times 128 = 147,456$

POOL2: [56x56x128] memorie:  $56 \times 56 \times 128 = 400K$  parametrii: 0

CONV3-256: [56x56x256] memorie:  $56 \times 56 \times 256 = 800K$  parametrii:  $(3 \times 3 \times 128) \times 256 = 294,912$

CONV3-256: [56x56x256] memorie:  $56 \times 56 \times 256 = 800K$  parametrii:  $(3 \times 3 \times 256) \times 256 = 589,824$

CONV3-256: [56x56x256] memorie:  $56 \times 56 \times 256 = 800K$  parametrii:  $(3 \times 3 \times 256) \times 256 = 589,824$

POOL2: [28x28x256] memorie:  $28 \times 28 \times 256 = 200K$  parametrii: 0

CONV3-512: [28x28x512] memorie:  $28 \times 28 \times 512 = 400K$  parametrii:  $(3 \times 3 \times 256) \times 512 = 1,179,648$

CONV3-512: [28x28x512] memorie:  $28 \times 28 \times 512 = 400K$  parametrii:  $(3 \times 3 \times 512) \times 512 = 2,359,296$

CONV3-512: [28x28x512] memorie:  $28 \times 28 \times 512 = 400K$  parametrii:  $(3 \times 3 \times 512) \times 512 = 2,359,296$

POOL2: [14x14x512] memorie:  $14 \times 14 \times 512 = 100K$  parametrii: 0

CONV3-512: [14x14x512] memorie:  $14 \times 14 \times 512 = 100K$  parametrii:  $(3 \times 3 \times 512) \times 512 = 2,359,296$

CONV3-512: [14x14x512] memorie:  $14 \times 14 \times 512 = 100K$  parametrii:  $(3 \times 3 \times 512) \times 512 = 2,359,296$

CONV3-512: [14x14x512] memorie:  $14 \times 14 \times 512 = 100K$  parametrii:  $(3 \times 3 \times 512) \times 512 = 2,359,296$

POOL2: [7x7x512] memorie:  $7 \times 7 \times 512 = 25K$  parametrii: 0

FC: [1x1x4096] memorie: 4096 parametrii:  $7 \times 7 \times 512 \times 4096 = 102,760,448$

FC: [1x1x4096] memorie: 4096 parametrii:  $4096 \times 4096 = 16,777,216$

FC: [1x1x1000] memorie: 1000 parametrii:  $4096 \times 1000 = 4,096,000$

**Memorie totală:  $24M * 4 \text{ bytes} \approx 93\text{MB} / \text{imagine}$**  (doar propagarea înainte,  $\sim 2$  înapoi)

**Numărul total de parametrii: 138M**

ConvNet Configuration			
B	C	D	E
13 weight layers	16 weight layers	16 weight layers	19 weight layers
put (224 x 224 RGB image)			
conv3-64 conv3-64 conv3-64 conv3-64			
<b>conv3-64</b>	conv3-64	conv3-64	cc
maxpool			
conv3-128 conv3-128 conv3-128 conv3-128			
<b>conv3-128</b>	conv3-128	conv3-128	co
maxpool			
conv3-256 conv3-256 conv3-256 conv3-256			
conv3-256	conv3-256	conv3-256	co
<b>conv1-256</b> conv3-256 conv3-256 conv3-256			
	<b>conv3-256</b>	conv3-256	co
maxpool			
conv3-512 conv3-512 conv3-512 conv3-512			
conv3-512	conv3-512	conv3-512	co
<b>conv1-512</b> conv3-512 conv3-512 conv3-512			
	<b>conv3-512</b>	conv3-512	co
maxpool			
conv3-512 conv3-512 conv3-512 conv3-512			
conv3-512	conv3-512	conv3-512	co
<b>conv1-512</b> conv3-512 conv3-512 conv3-512			
	<b>conv3-512</b>	conv3-512	co
maxpool			
FC-4096 FC-4096 FC-4096 FC-4096			
FC-4096	FC-4096	FC-4096	co
FC-4096	FC-4096	FC-4096	co
FC-1000	FC-1000	FC-1000	co
soft-max			

INPUT: [224x224x3] memorie:  $224 \times 224 \times 3 = 150\text{K}$  parametrii: 0  
 CONV3-64: [224x224x64] memorie:  $224 \times 224 \times 64 = 3.2\text{M}$  parametrii:  $(3 \times 3 \times 3) \times 64 = 1,728$   
 CONV3-64: [224x224x64] memorie:  $224 \times 224 \times 64 = 3.2\text{M}$  parametrii:  $(3 \times 3 \times 64) \times 64 = 36,864$   
 POOL2: [112x112x64] memorie:  $112 \times 112 \times 64 = 800\text{K}$  parametrii: 0  
 CONV3-128: [112x112x128] memorie:  $112 \times 112 \times 128 = 1.6\text{M}$  parametrii:  $(3 \times 3 \times 64) \times 128 = 73,728$   
 CONV3-128: [112x112x128] memorie:  $112 \times 112 \times 128 = 1.6\text{M}$  parametrii:  $(3 \times 3 \times 128) \times 128 = 147,456$   
 POOL2: [56x56x128] memorie:  $56 \times 56 \times 128 = 400\text{K}$  parametrii: 0  
 CONV3-256: [56x56x256] memorie:  $56 \times 56 \times 256 = 800\text{K}$  parametrii:  $(3 \times 3 \times 128) \times 256 = 294,912$   
 CONV3-256: [56x56x256] memorie:  $56 \times 56 \times 256 = 800\text{K}$  parametrii:  $(3 \times 3 \times 256) \times 256 = 589,824$   
 CONV3-256: [56x56x256] memorie:  $56 \times 56 \times 256 = 800\text{K}$  parametrii:  $(3 \times 3 \times 256) \times 256 = 589,824$   
 POOL2: [28x28x256] memorie:  $28 \times 28 \times 256 = 200\text{K}$  parametrii: 0  
 CONV3-512: [28x28x512] memorie:  $28 \times 28 \times 512 = 400\text{K}$  parametrii:  $(3 \times 3 \times 256) \times 512 = 1,179,648$   
 CONV3-512: [28x28x512] memorie:  $28 \times 28 \times 512 = 400\text{K}$  parametrii:  $(3 \times 3 \times 512) \times 512 = 2,359,296$   
 CONV3-512: [28x28x512] memorie:  $28 \times 28 \times 512 = 400\text{K}$  parametrii:  $(3 \times 3 \times 512) \times 512 = 2,359,296$   
 POOL2: [14x14x512] memorie:  $14 \times 14 \times 512 = 100\text{K}$  parametrii: 0  
 CONV3-512: [14x14x512] memorie:  $14 \times 14 \times 512 = 100\text{K}$  parametrii:  $(3 \times 3 \times 512) \times 512 = 2,359,296$   
 CONV3-512: [14x14x512] memorie:  $14 \times 14 \times 512 = 100\text{K}$  parametrii:  $(3 \times 3 \times 512) \times 512 = 2,359,296$   
 CONV3-512: [14x14x512] memorie:  $14 \times 14 \times 512 = 100\text{K}$  parametrii:  $(3 \times 3 \times 512) \times 512 = 2,359,296$   
 POOL2: [7x7x512] memorie:  $7 \times 7 \times 512 = 25\text{K}$  parametrii: 0  
 FC: [1x1x4096] memorie: 4096 parametrii:  $7 \times 7 \times 512 \times 4096 = 102,760,448$   
 FC: [1x1x4096] memorie: 4096 parametrii:  $4096 \times 4096 = 16,777,216$   
 FC: [1x1x1000] memorie: 1000 parametrii:  $4096 \times 1000 = 4,096,000$

Observații:

Cea mai multă memorie este consumată în primele straturi CONV

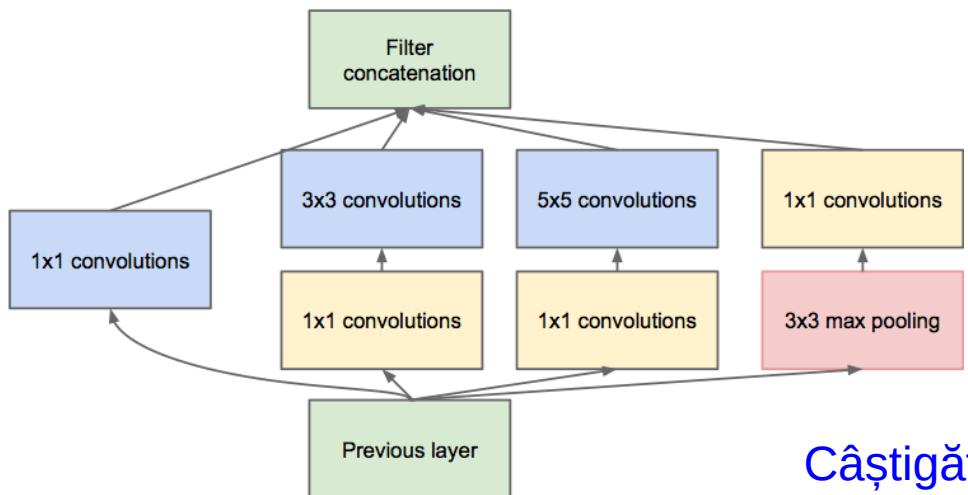
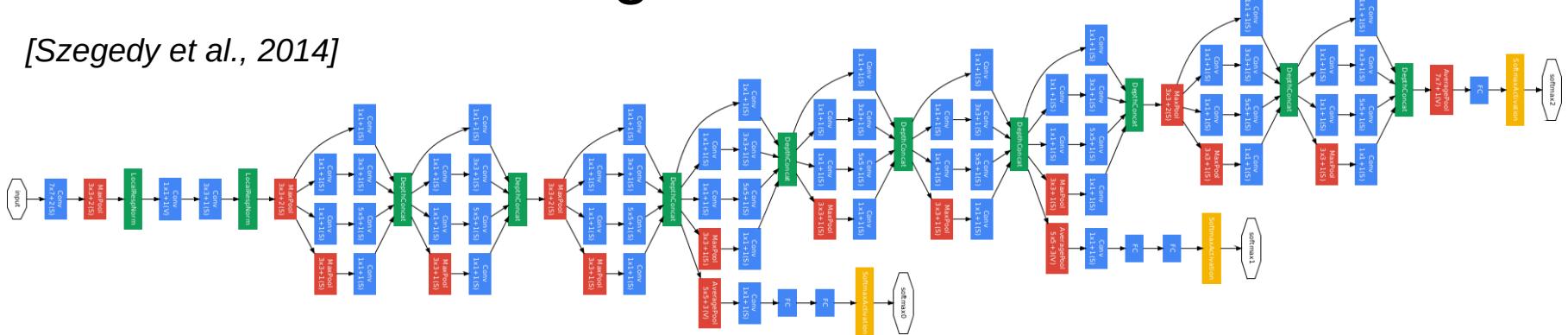
Cei mai mulți parametrii în ultimele straturi FC

Memorie totală:  $24\text{M} * 4 \text{ bytes} \approx 93\text{MB} / \text{ imagine}$  (doar propagarea înainte, ~x2 înapoi)

Numărul total de parametrii: 138M

# Studiu de caz: GoogLeNet

[Szegedy et al., 2014]



## Modulul Inception

Câștigătorul ILSVRC 2014 (6.7% eroarea top 5)

# Studiu de caz: GoogLeNet

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Avantaje interesante:

- Doar 5 milioane de parametrii! (elimină straturile FC)

Comparat cu AlexNet:

- 12x mai puțini parametrii
- 2x mai multe calcule
- 6.67% (vs. 16.4%)

# Studiu de caz: ResNet

[He et al., 2015]

Câștigătorul ILSVRC 2015 (3.6% eroare top 5)



## MSRA @ ILSVRC & COCO 2015 Competitions

- **1st places in all five main tracks**

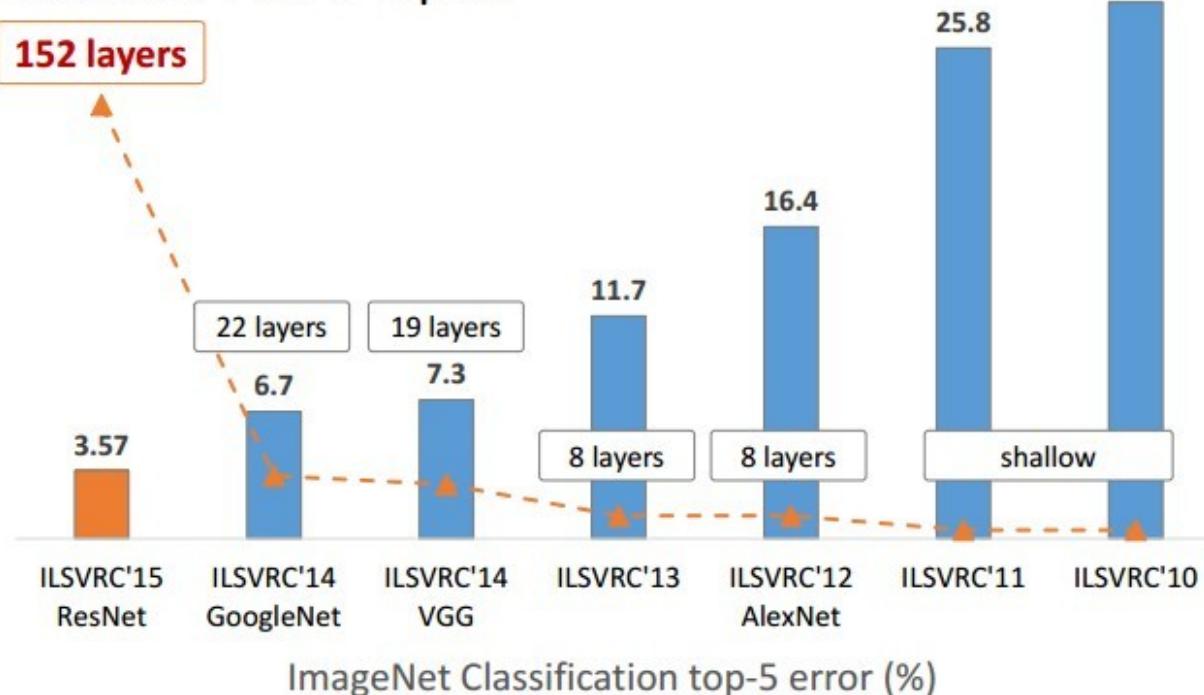
- ImageNet Classification: “Ultra-deep” (quote Yann) **152-layer** nets
- ImageNet Detection: **16%** better than 2nd
- ImageNet Localization: **27%** better than 2nd
- COCO Detection: **11%** better than 2nd
- COCO Segmentation: **12%** better than 2nd

\*improvements are relative numbers

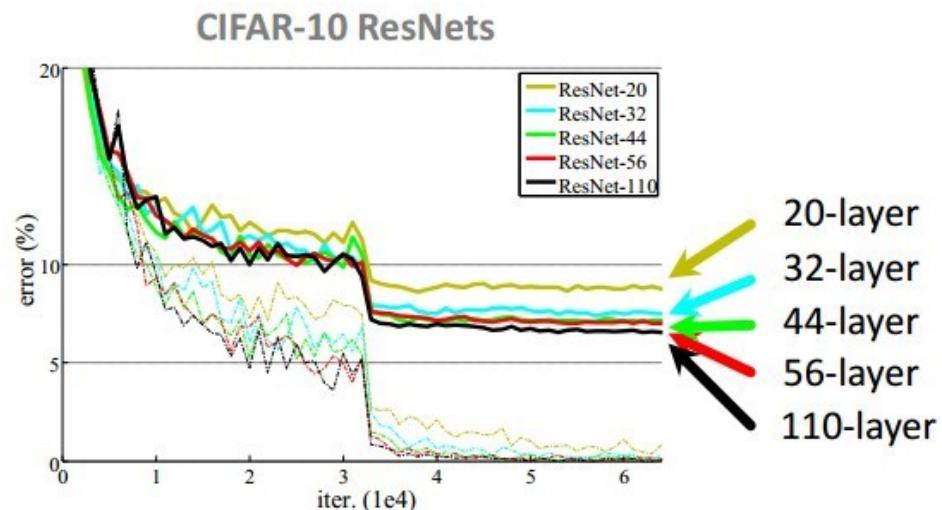
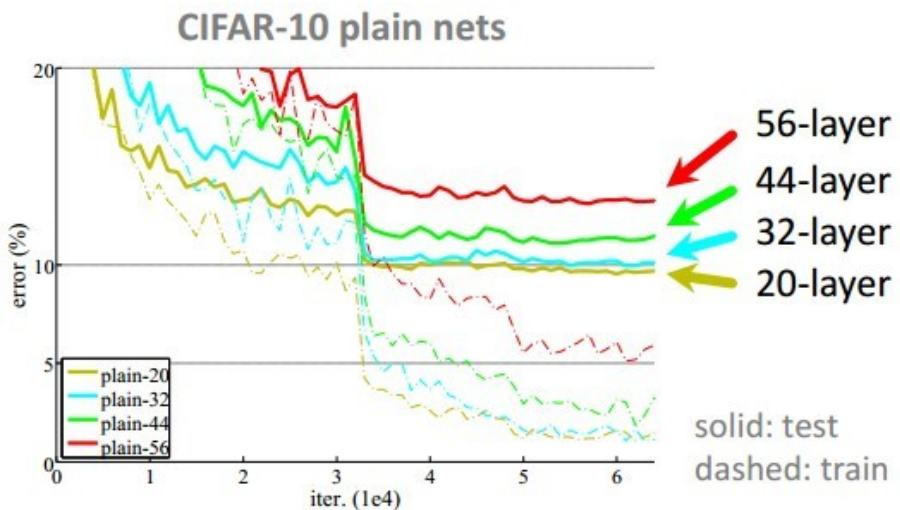


Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015.

# Revolution of Depth



# CIFAR-10 experiments



# Studiu de caz: ResNet

[He et al., 2015]

Câștigătorul ILSVRC 2015 (3.6% eroare top 5)

## Revolution of Depth

AlexNet, 8 layers  
(ILSVRC 2012)



VGG, 19 layers  
(ILSVRC 2014)



ResNet, 152 layers  
(ILSVRC 2015)

Microsoft  
Research

2-3 săptămâni pentru antrenare pe  
un server cu 8 plăci GPU

la runtime: mai rapid ca VGGNet!  
(deși are de 8x mai multe straturi)



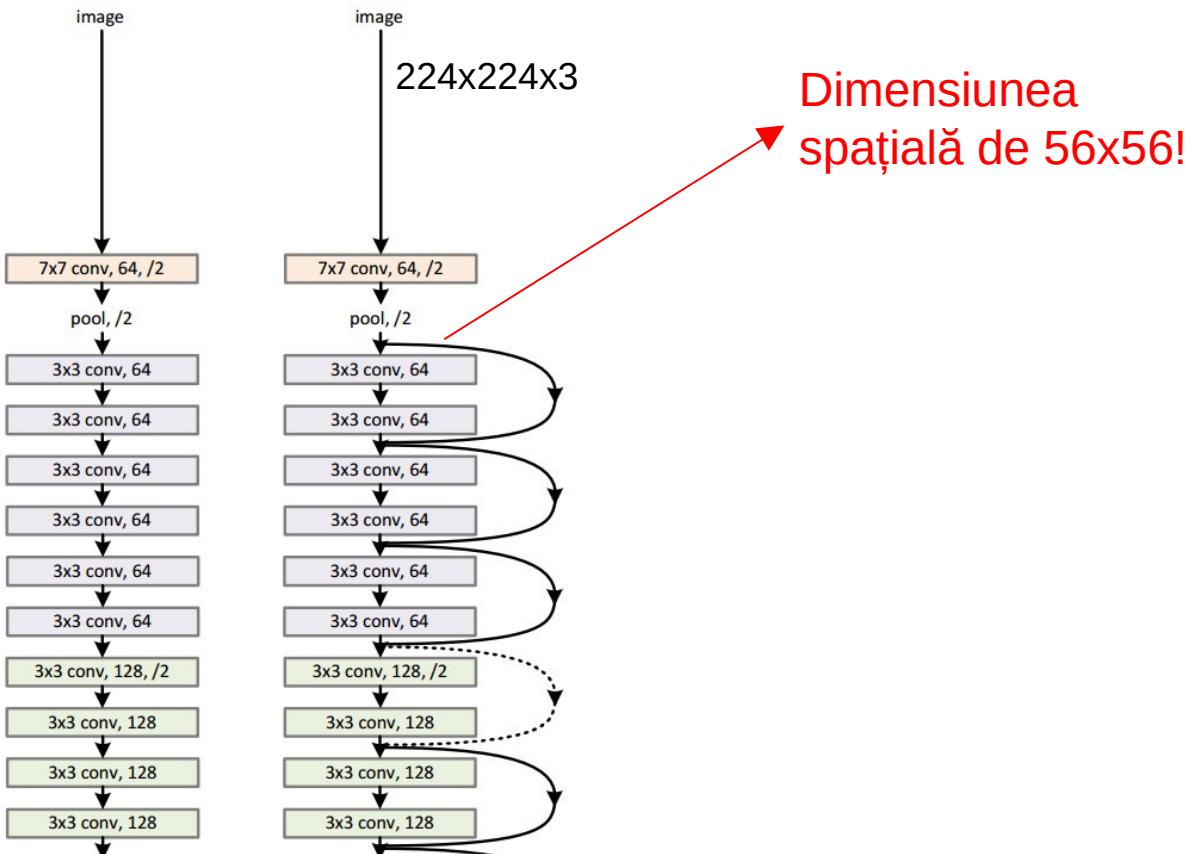
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. "Deep Residual Learning for Image Recognition". arXiv 2015.

Slide după Kaiming He

# Studiu de caz: ResNet

[He et al., 2015]

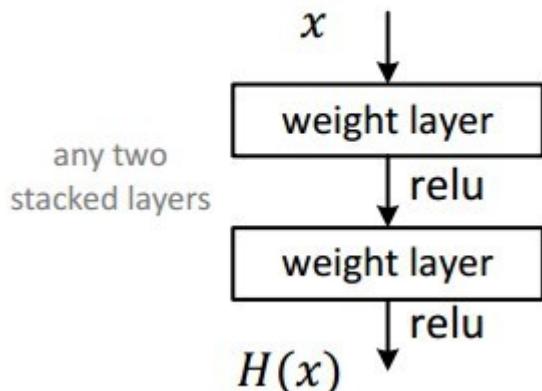
34-layer plain      34-layer residual



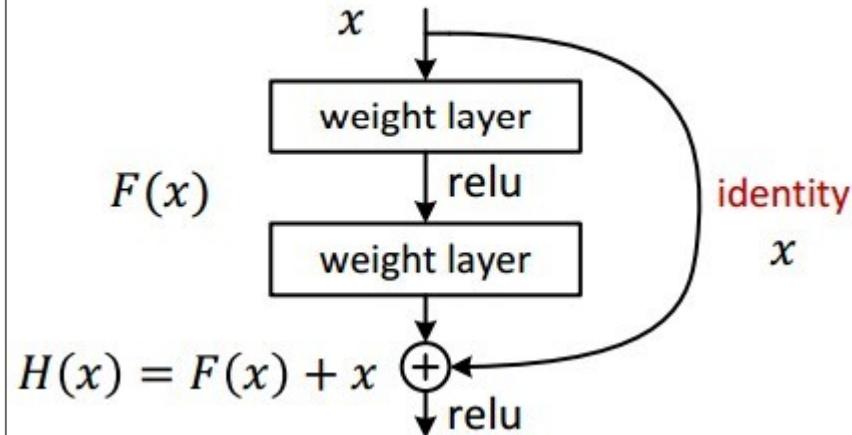
# Studiu de caz: ResNet

[He et al., 2015]

- Plain net



- Residual net



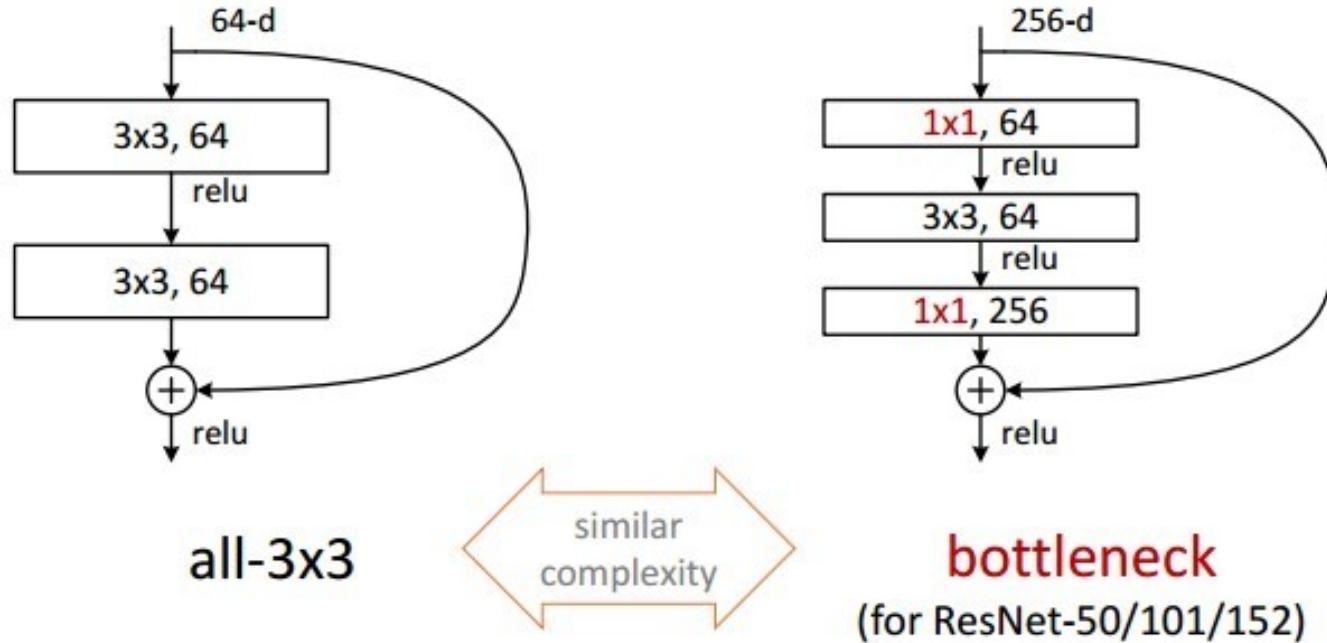
# Studiu de caz: ResNet

[He et al., 2015]

- Normalizare batch după fiecare strat CONV
- Inițializare Xavier / 2
- SGD cu moment 0.9
- Rată de învățare: 0.1, împărțită la 10 când eroare pe validare atinge un platou
- Mărimea unui mini-batch 256
- Degradarea ponderilor cu  $10^{-5}$
- Fără dropout!

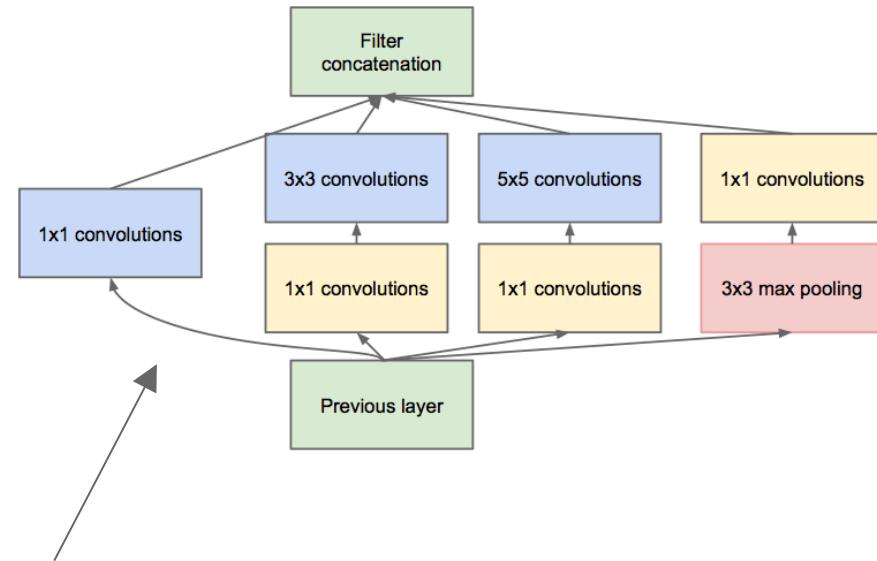
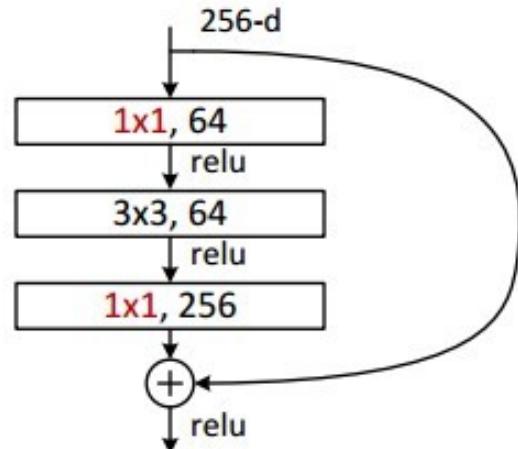
# Studiu de caz: ResNet

[He et al., 2015]



# Studiu de caz: ResNet

[He et al., 2015]

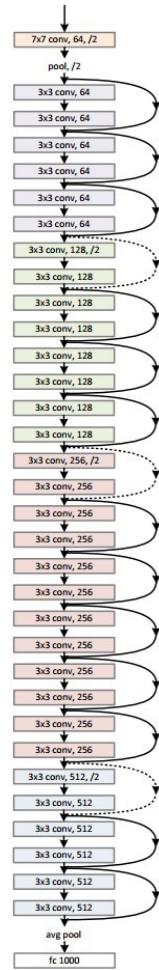


(truc utilizat și în GoogLeNet)

# Studiu de caz: ResNet

[He et al., 2015]

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112			7×7, 64, stride 2		
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		$1.8 \times 10^9$	$3.6 \times 10^9$	$3.8 \times 10^9$	$7.6 \times 10^9$	$11.3 \times 10^9$



# Concluzii

- ConvNets: folosim straturi CONV, POOL, FC
- Tendință către filtre mai mici și arhitecturi mai adânci
- Tendință către eliminarea straturilor POOL/FC (doar CONV)
- Arhitectura tipică arată astfel:

**$[(CONV-RELU)*N-POOL?] * M - (FC-RELU)*K, SOFTMAX$**

unde N este deobicei în jur de ~5, M este mare,  $0 \leq K \leq 2$

- Arhitecturile recente (ResNet / GoogLeNet) pun sub semnul întrebării această paradigmă

# Recapitulare

# Paradigma de învățare supervizată

Functions  $\mathcal{F}$

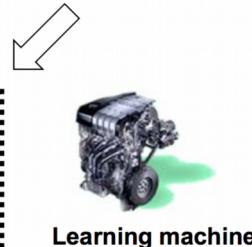
$$f : \mathcal{X} \rightarrow \mathcal{Y}$$

Training data

$$\{(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}\}$$

LEARNING

$$\begin{aligned} &\text{find } \hat{f} \in \mathcal{F} \\ &\text{s.t. } y_i \approx \hat{f}(x_i) \end{aligned}$$



PREDICTION

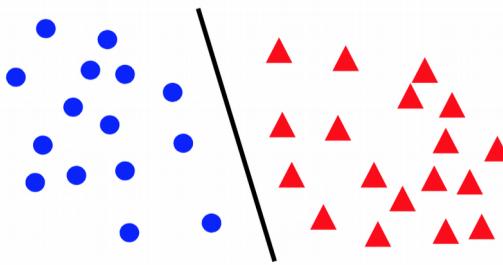
$$y = \hat{f}(x)$$

New data

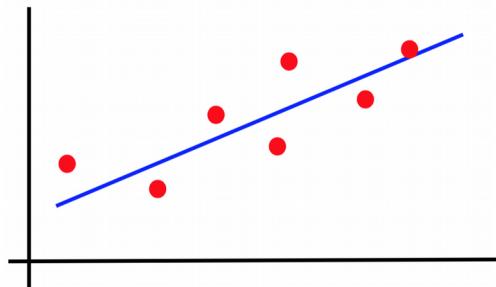
$$x$$

# Formele canonice ale problemelor de învățare supervizată

- Clasificare



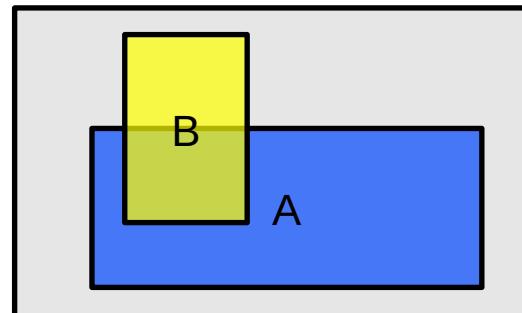
- Regresie



# Regula Bayes

$$P(B | A) = \frac{P(A \cap B)}{P(A)} = \frac{P(A | B) P(B)}{P(A)}$$

- Thomas Bayes "An Essay towards solving a Problem in the Doctrine of Chances" Royal Society, 1763.
- Thomas Bayes "An Essay towards solving a Problem in the Doctrine of Chances" Royal Society, 1763.
- Simplu de înțeles dacă vă gânditi la arii
- Simplu de înțeles dacă vă gândiți la arii



# Clasificatorul Naïve Bayes

- Presupunerea Naïve Bayes:

➤ Trăsăturile sunt independente:

- Presupunerea Naïve Bayes:

➤ Trăsăturile sunt independente:

➤ Mai general:

$$P(X_1, X_2 | Y) = P(X_1 | Y)P(X_2 | Y)$$

➤ Mai general:

$$P(X_1 \dots X_n | Y) = \prod_i P(X_i | Y)$$

# Clasificatorul Naïve Bayes

- Fiind date:
  - Probabilitatea apriori a claselor  $P(Y)$
  - ~~n trăsături independente  $X$  condiționate de  $Y$~~
  - Probabilitatea apriori probabilitatea  $P(X_i | Y)$
  - $n$  trăsături independente  $X$  condiționate de  $Y$
- ~~Regula de decizie Naïve Bayes este:~~

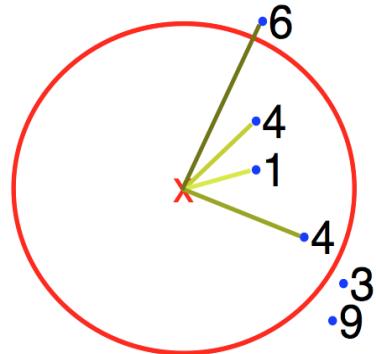
$$h_{NB}(x) = \operatorname{argmax} P(y) P(x_1, \dots, x_n | y)$$

- Regula de decizie Naïve Bayes<sup>y</sup> este:

$$h_{NB}(x) = \operatorname{argmax}_y P(y) \prod_i P(x_i | y)$$

- În practică folosim sumă de log!
- În practică folosim sumă de log!

# Metoda celor mai apropiati vecini



- Algoritmul k-NN:
  - 1) Pentru fiecare exemplu de test  $x$ , găsim cei mai apropiati  $k$  vecini
  - 2) Atribuim eticheta majoritară conform celor  $k$  vecini

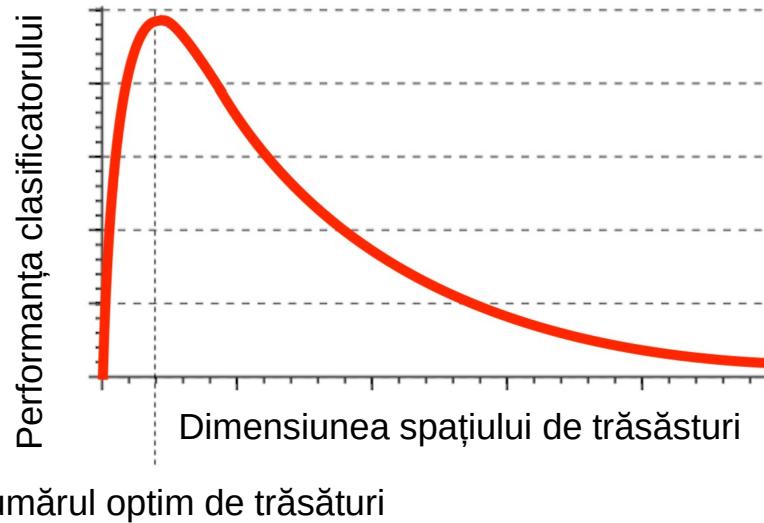
# Ce ne trebuie pentru un clasificator bazat pe memorie?

- O funcție de distanță
  - Distanța Euclideană
  - Distanța Edit (Levenshtein)
  - Distanța Hamming
- Câți vecini să luăm în considerare?
- Cum să antrenăm modelul pe exemplele din vecinătate?

# Blestemul dimensionalității

- Pentru a “umple” un spațiu nD (de exemplu  $\mathbb{R}^n$ ) avem nevoie de un număr exponential de puncte
- Dacă avem un număr mare de caracteristici care descriu datele, atunci sistemul are nevoie de foarte multe exemple de antrenare pentru învăță un model care să generalizeze
- De cele mai multe ori aceste date nu sunt disponibile în practică

# Fenomenul Hughes



- Fenomenul Hughes arată că, pe măsură ce numărul de caracteristici crește, performanța clasificatorului crește până când ajungem la numărul optim de trăsături
- Adăugarea mai multor caracteristici păstrând dimensiunea setului de antrenare degradează performanța clasificatorului

## Forma primală

Features:  $f_1, f_2, f_3, f_4, f_5, f_6, f_7$

Train samples:  
 $x_1, x_2, x_3, x_4$

	$f_1$	$f_2$	$f_3$	$f_4$	$f_5$	$f_6$	$f_7$
$x_1$	4	0	2	5	3	0	1
$x_2$	0	0	1	3	4	0	2
$x_3$	2	1	0	0	1	2	5
$x_4$	1	3	0	1	0	1	2

= X

$l_1$	1
$l_2$	1
$l_3$	-1
$l_4$	-1

= L



Linear classifier:  $C = (w_1, w_2, w_3, w_4, w_5, w_6, w_7, b)$  such that  $\text{sign}(X * W' + b) = L$



Test samples:  
 $y_1, y_2, y_3$

	$f_1$	$f_2$	$f_3$	$f_4$	$f_5$	$f_6$	$f_7$
$y_1$	1	0	2	4	2	0	2
$y_2$	1	2	0	1	2	2	1
$y_3$	3	1	0	0	4	1	1

= Y

$p_1$	?
$p_2$	?
$p_3$	?

= P

Apply C to obtain predictions:  $P = \text{sign}(Y * W' + b)$

## Forma duală

Kernel type: **linear**

Train samples:  
 $x_1, x_2, x_3, x_4$

	$x_1$	$x_2$	$x_3$	$x_4$
$x_1$	55	31	16	11
$x_2$	31	30	14	7
$x_3$	16	14	35	17
$x_4$	11	7	17	16

$$= \mathbf{X} * \mathbf{X}' = \mathbf{K}_X$$

$l_1$	1
$l_2$	1
$l_3$	-1
$l_4$	-1

$$= L$$



Linear classifier:  $C = (\alpha_1, \alpha_2, \alpha_3, \alpha_4, b)$  such that  $\text{sign}(\mathbf{K}_X * \alpha' + b) = L$



Test samples:  
 $y_1, y_2, y_3$

	$x_1$	$x_2$	$x_3$	$x_4$
$y_1$	36	26	14	9
$y_2$	16	13	15	12
$y_3$	25	18	18	9

$$= \mathbf{Y} * \mathbf{X}' = \mathbf{K}_Y$$

$p_1$	?
$p_2$	?
$p_3$	?

$$= P$$

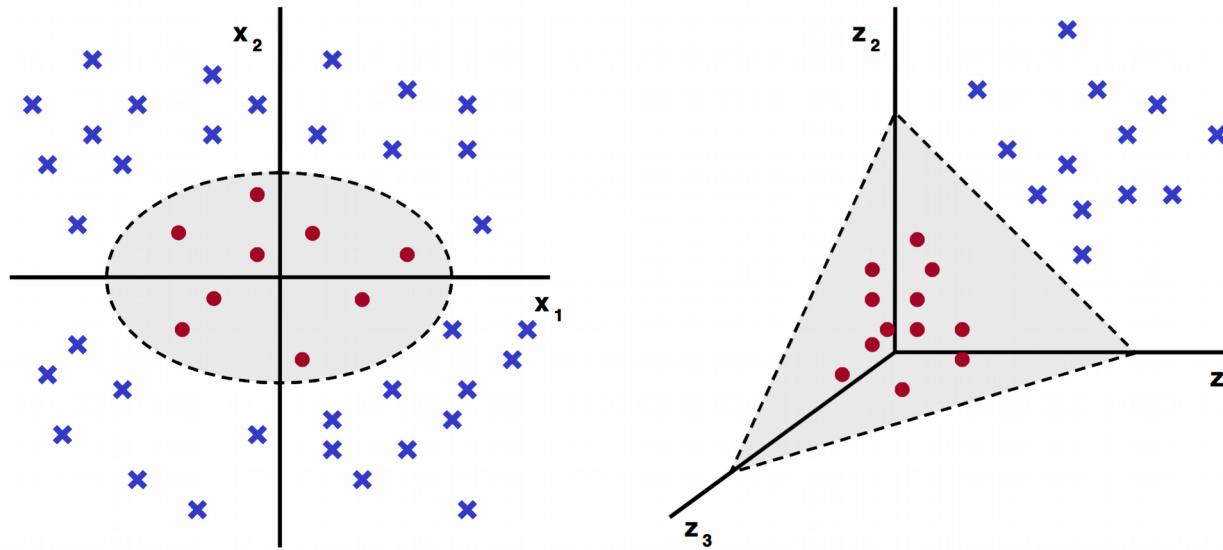
Apply C to obtain predictions:  $P = \text{sign}(\mathbf{K}_Y * \alpha' + b)$

# Exemple de funcții kernel

- Prin definirea explicită a funcției de scufundare

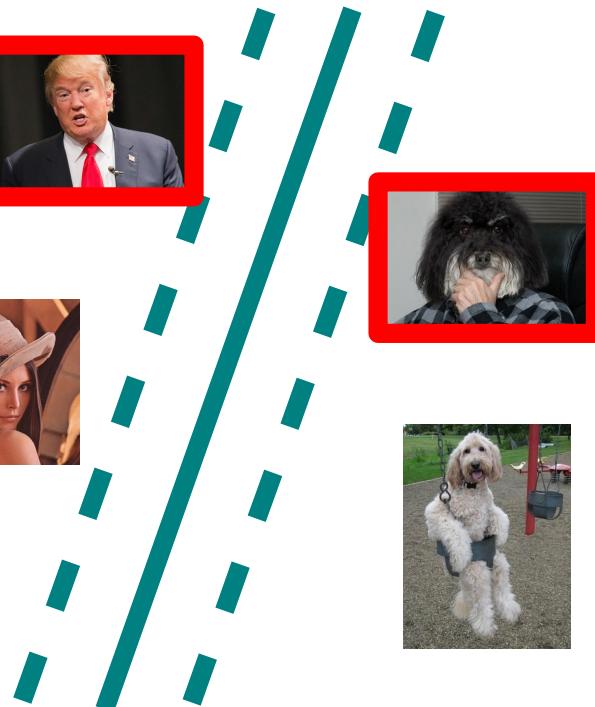
$$\phi : \mathbb{R}^2 \rightarrow \mathbb{R}^3$$

$$(x_1, x_2) \mapsto (z_1, z_2, z_3) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$



# Alegem hiperplanul de margine maximă

- Mașini cu **vectori suport** (SVM)



# SVM (Hard Margin)

$$S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots, (\mathbf{x}_\ell, y_\ell)\}$$

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$$

$$\max_{\mathbf{w}, b, \gamma} \gamma$$

subject to

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq \gamma \\ i = 1, \dots, \ell$$

$$\|\mathbf{w}\|^2 = 1$$



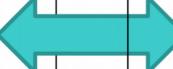
$$\min_{\mathbf{w}, b} \|\mathbf{w}\|^2$$

subject to

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \\ i = 1, \dots, \ell$$

# SVM (Soft Margin)

- În cazul în care exemple nu sunt liniar separabile:



$$\min_{\mathbf{w}, b, \gamma, \xi} -\gamma + C \sum_{i=1}^{\ell} \xi_i$$

subject to

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq \gamma - \xi_i$$
$$\xi_i \geq 0 \quad i = 1, \dots, \ell$$
$$\|\mathbf{w}\|^2 = 1$$

$$\min_{\mathbf{w}, b, \xi} \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{\ell} \xi_i$$

subject to

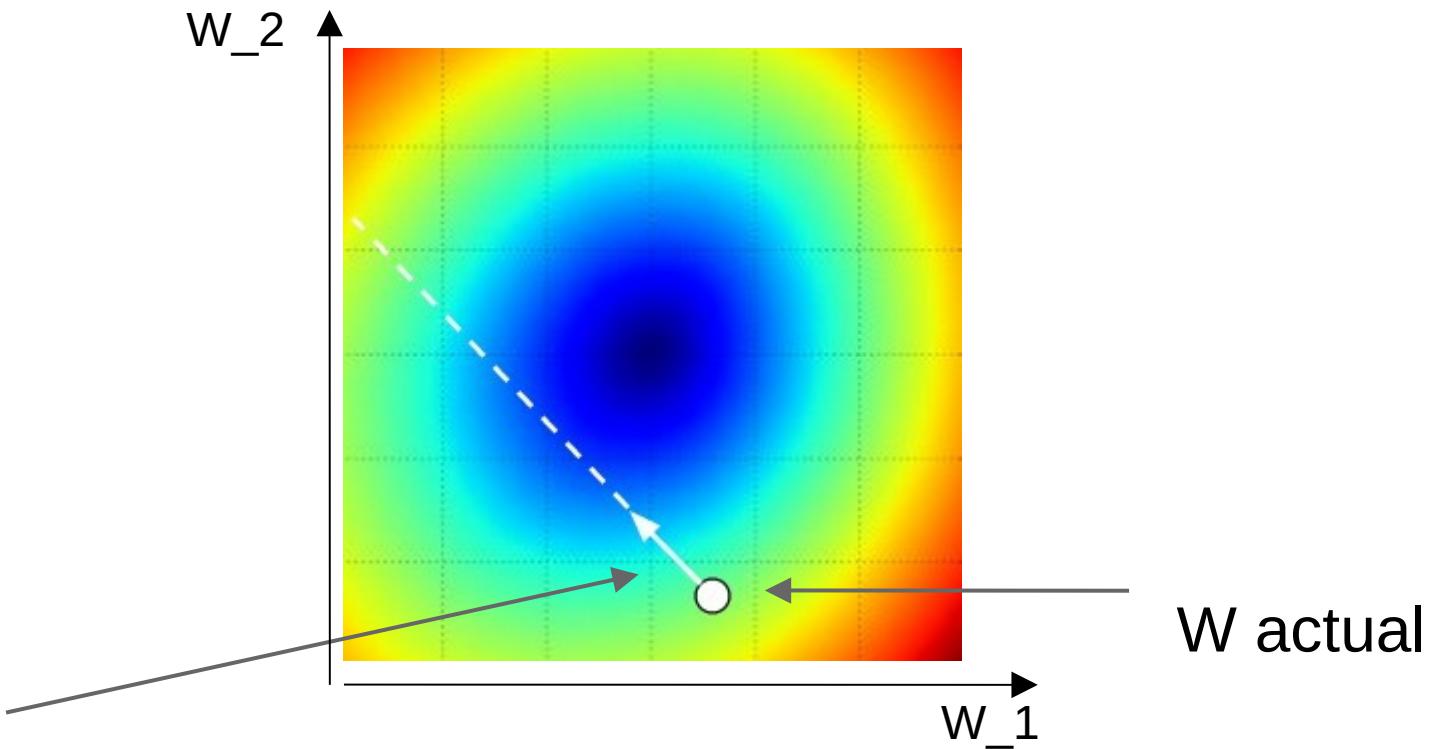
$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 - \xi_i$$
$$\xi_i \geq 0 \quad i = 1, \dots, \ell$$

## Algoritm: Coborârea pe gradient

Într-o singură dimensiune, derivata unei funcții este:

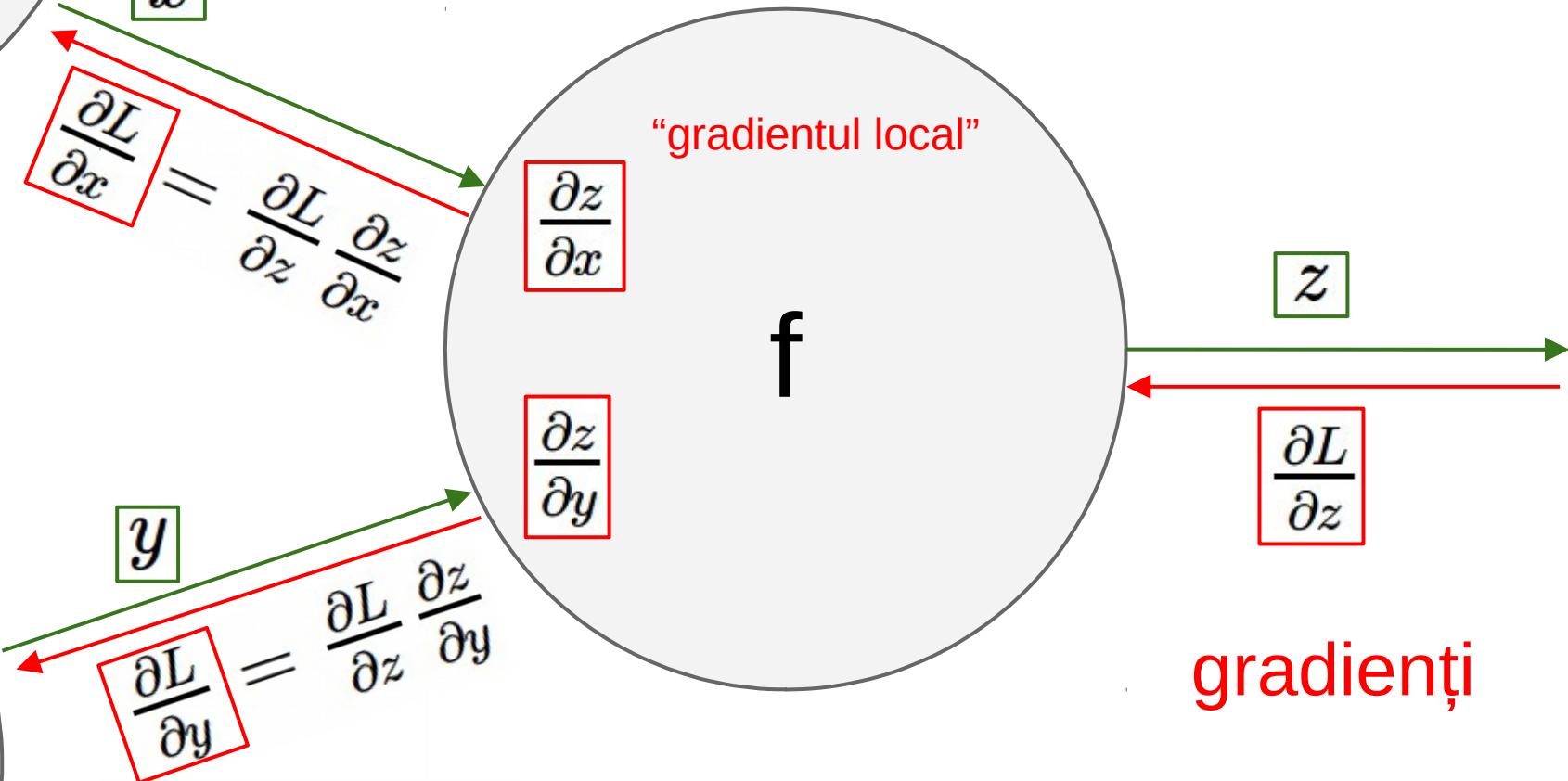
$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x + h) - f(x)}{h}$$

În mai multe dimensiuni, **gradientul** este un vector cu derivate parțiale.

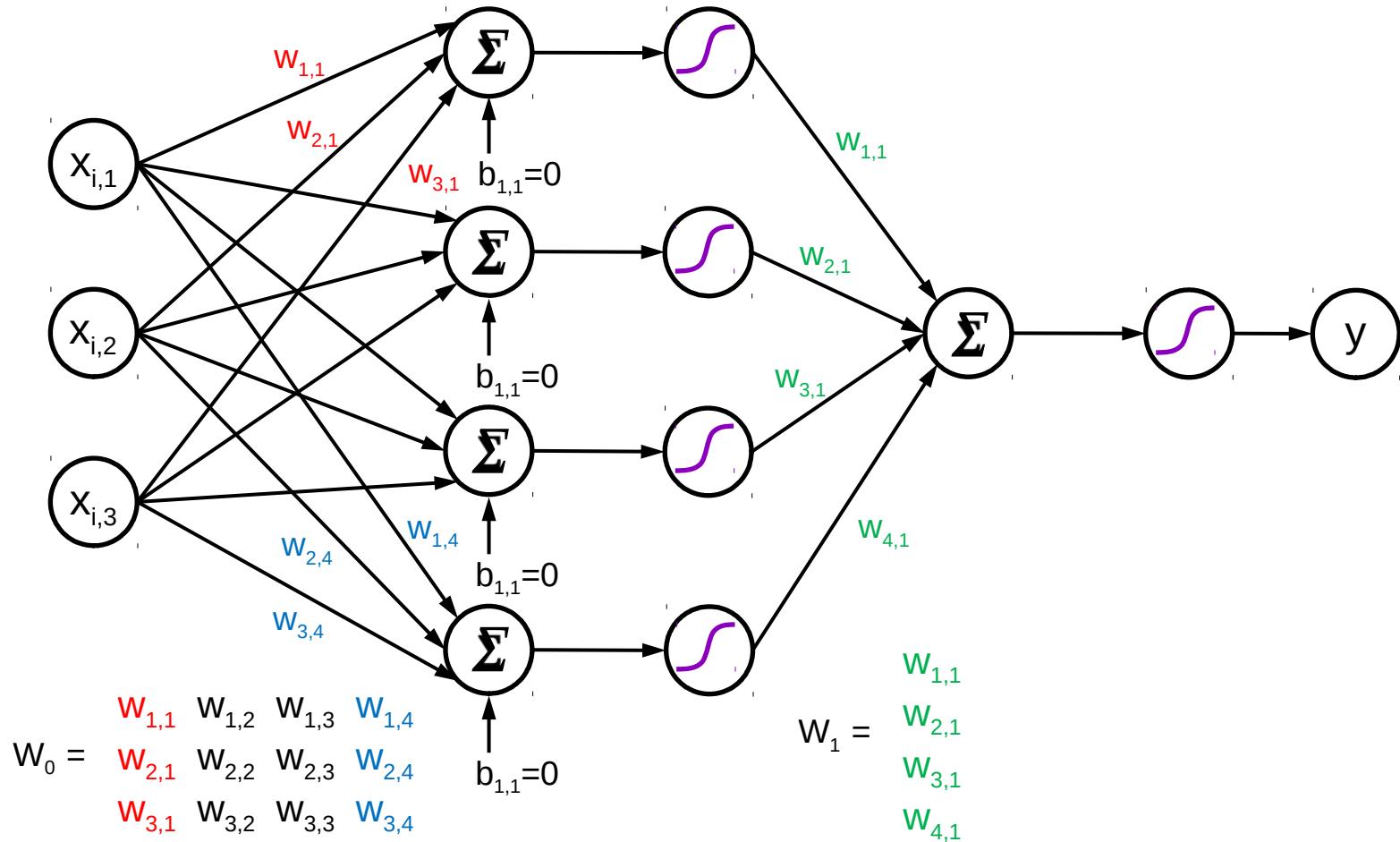


direcția negativă a gradientului

# Propagarea gradientului prin regula de înlățuire activări



# Arhitectura unei rețele neuronale cu două niveluri



# Rețele neuronale sunt funcții universale de aproximare

- **Teorema Aproximării Universale:**

O rețea neuronală de tip feed-forward cu un strat ascuns având un număr infinit de neuroni poate aproxima orice funcție continuă definită pe un subset compact din  $\mathbb{R}^n$ .

- Desi rețele neuronale cu două straturi (un strat ascuns) sunt funcții universale de aproximare, lățimea (numărul de perceptroni) acestor rețele poate fi exponential de mare.
- În practică, preferăm rețele mai adânci (cu mai multe straturi)

# Ce metodă de clasificare este cea mai bună?

- **Teorema “No free lunch”:**

Oricare doi algoritmi sunt echivalenți atunci când performanța lor este măsurată (în medie) pe toate problemele posibile

- Rezultă ca nu există nici o scurtătură în alegerea algoritmului potrivit pentru o anumită problemă
- Deobicei încercăm mai mulți algoritmi și vedem care obține rezultate mai bune