

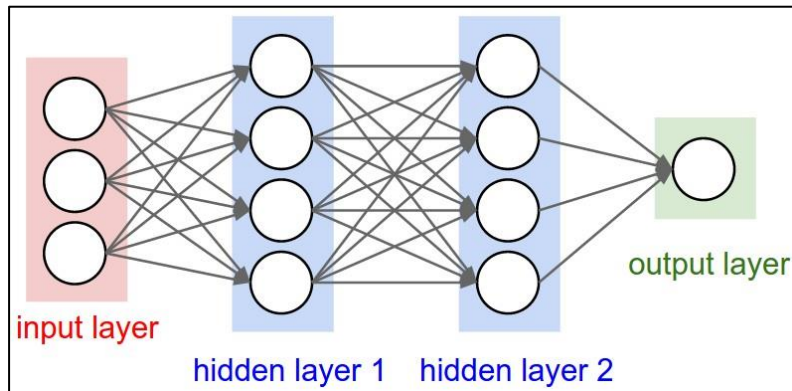
Rețele neuronale convoluționale. Straturi speciale și arhitecturi.

Prof. Dr. Radu Ionescu
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 pierderea
3. Propagăm **înapoi** pentru a calcula gradientii
4. Actualizăm ponderile pe baza gradientilor calculați



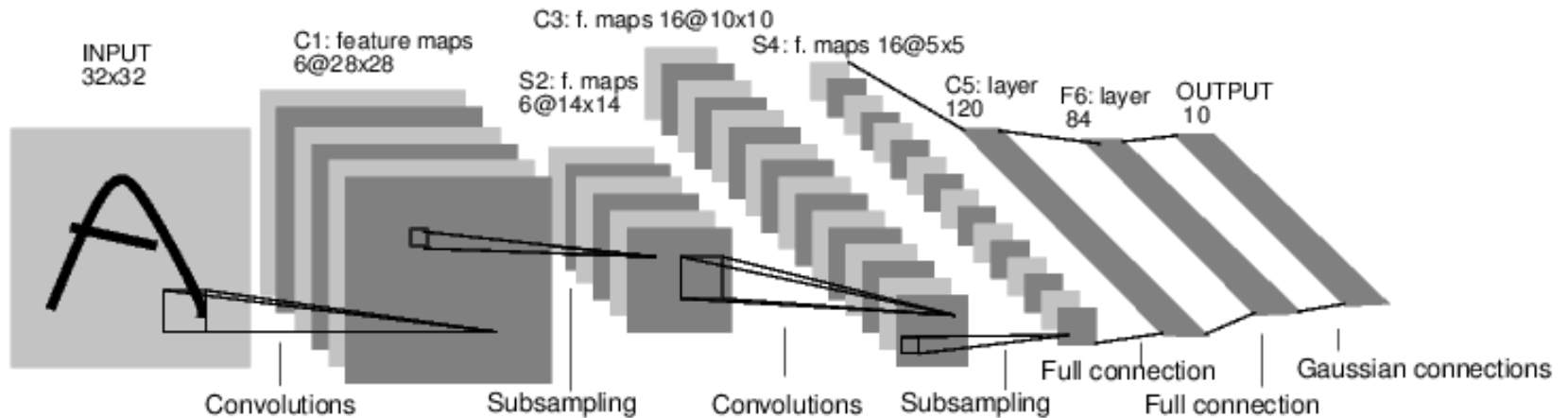
Rețele neuronale sunt funcții universale de aproximare

- **Teorema Aproximării Universale:**

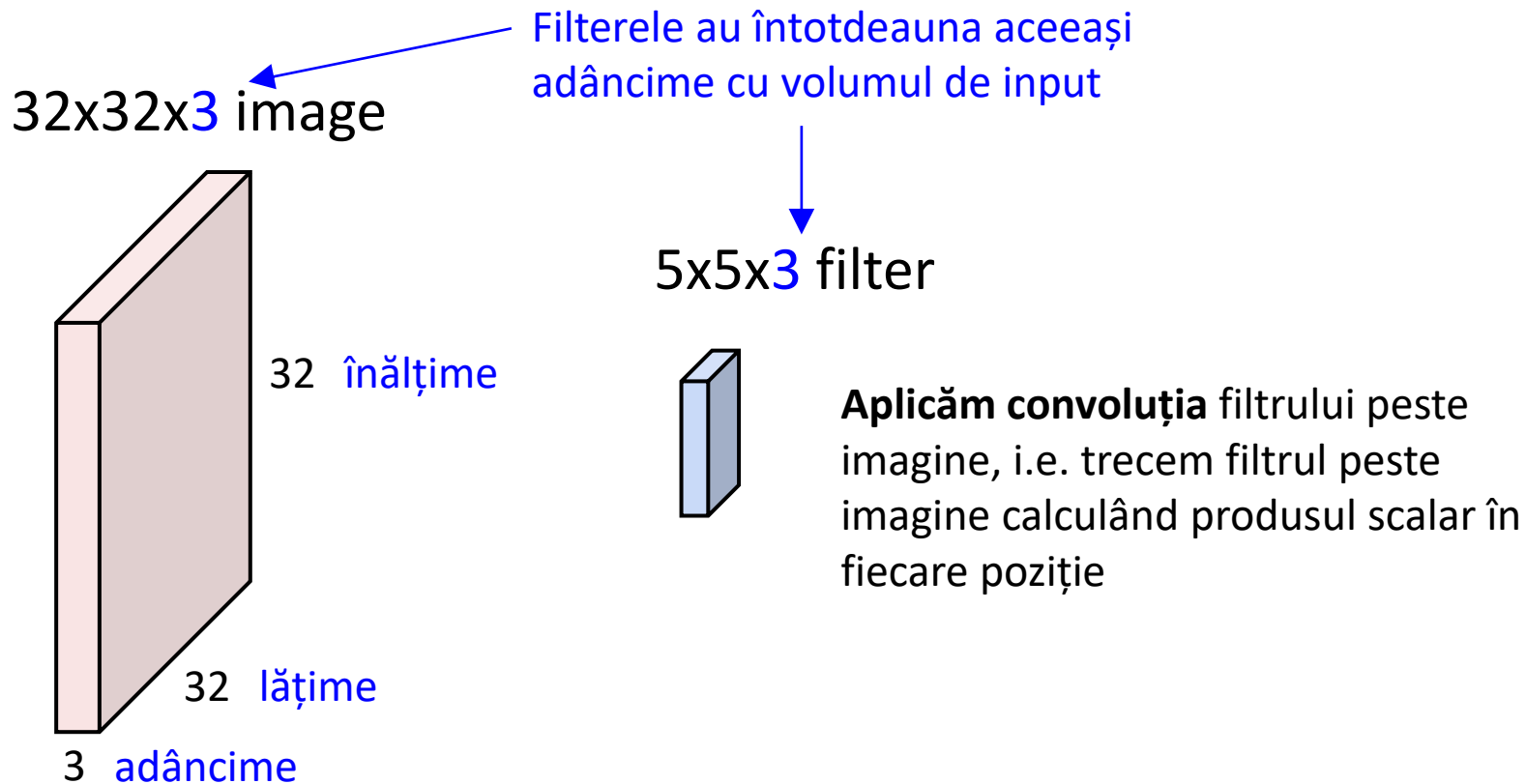
O rețea neuronală de tip feed-forward cu un strat ascuns având un număr finit de neuroni poate aproxima orice funcție continuă definită pe un subset compact din \mathbb{R}^n .

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

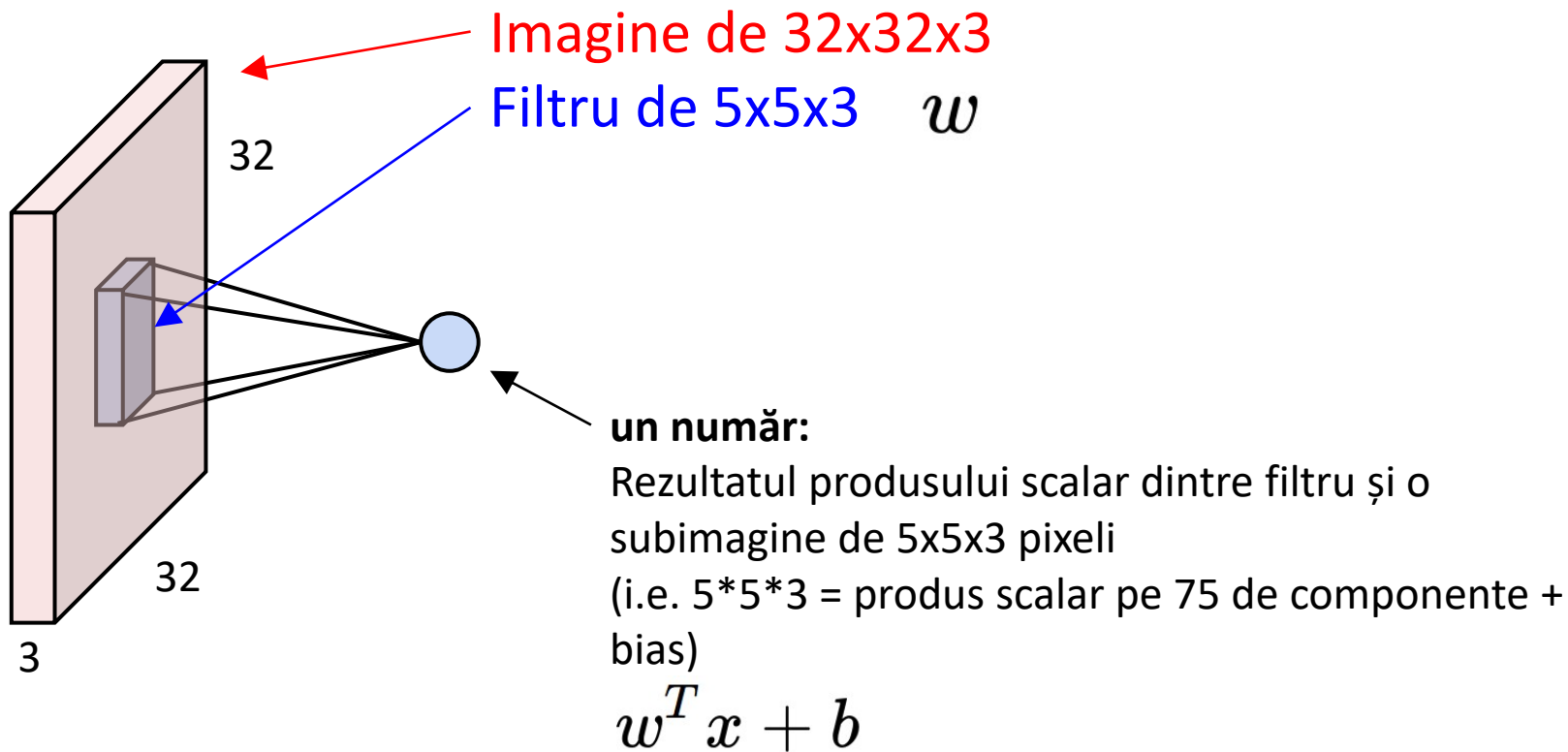
Rețele neuronale convoluționale



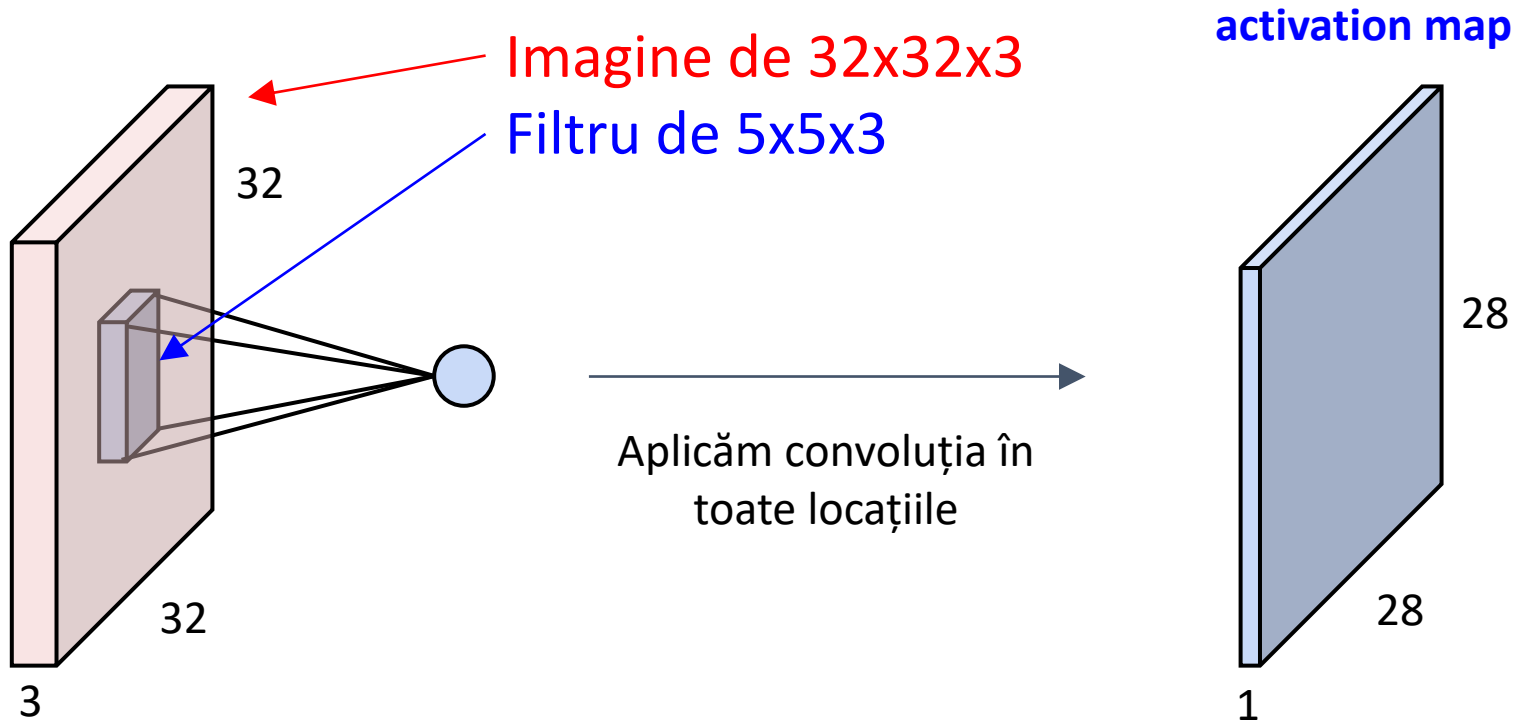
Stratul convoluțional



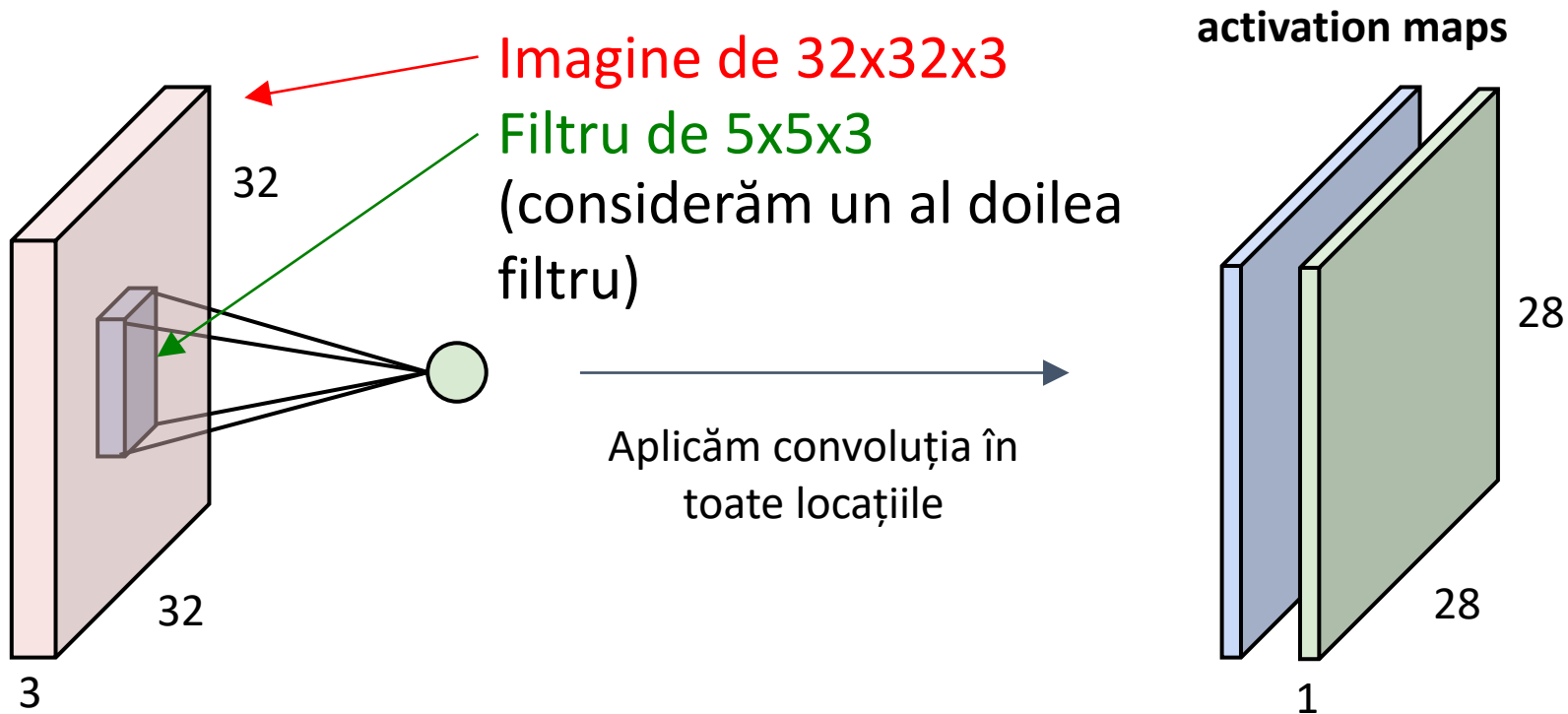
Stratul convoluțional



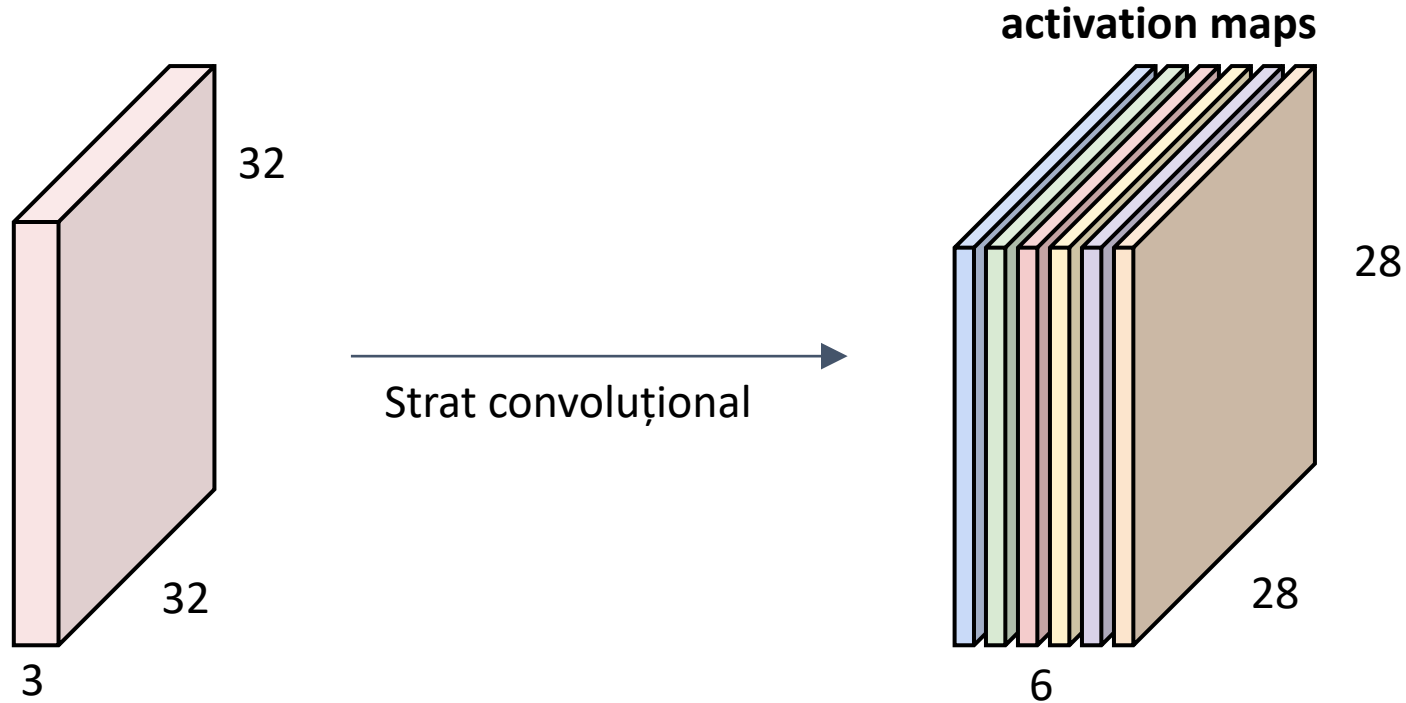
Stratul convoluțional



Stratul convoluțional

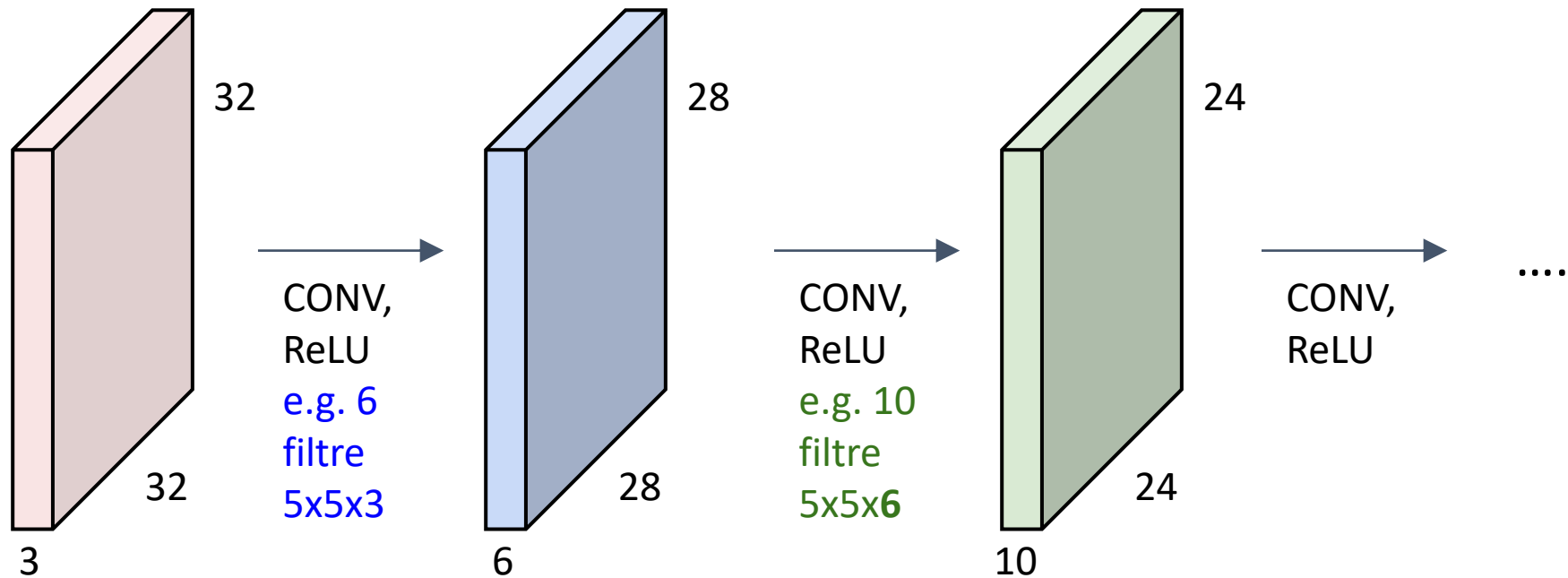


Un strat convoluțional este format din mai multe filtre (de exemplu 6)

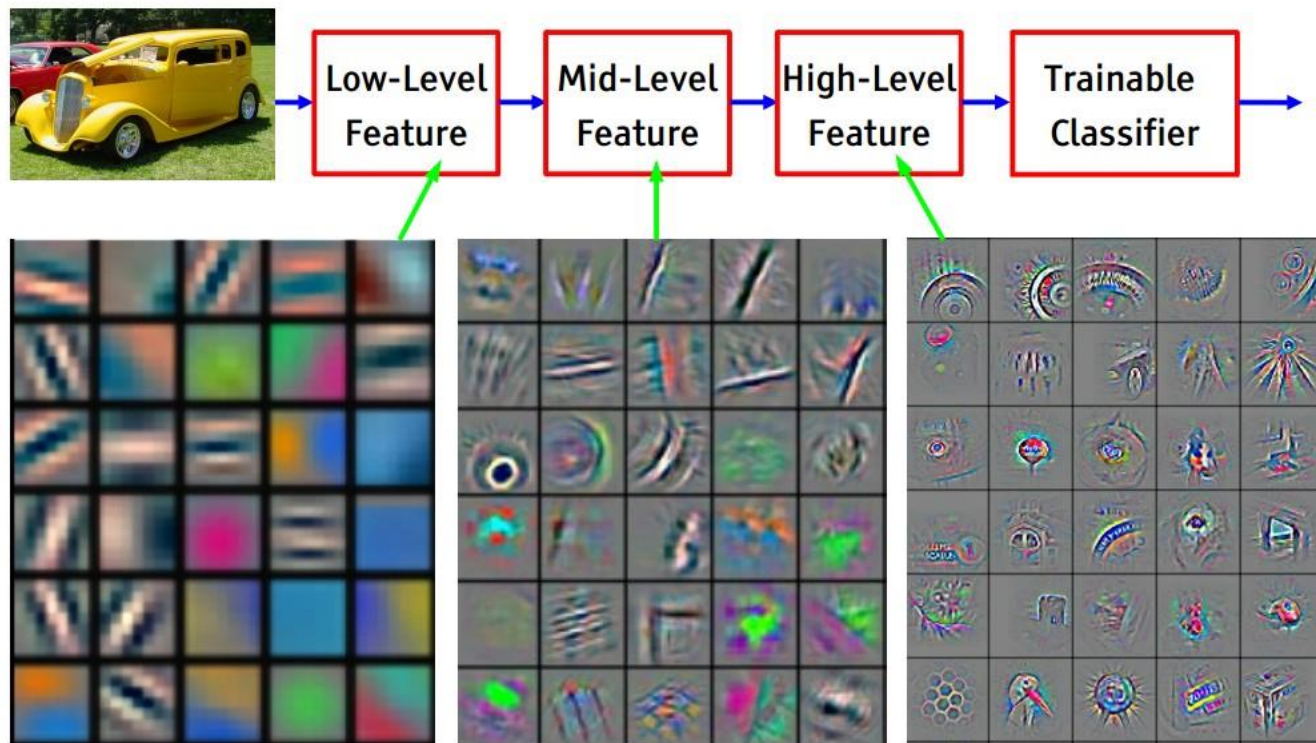


Concatenăm activările pentru a obține o nouă "image" de 28x28x6

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

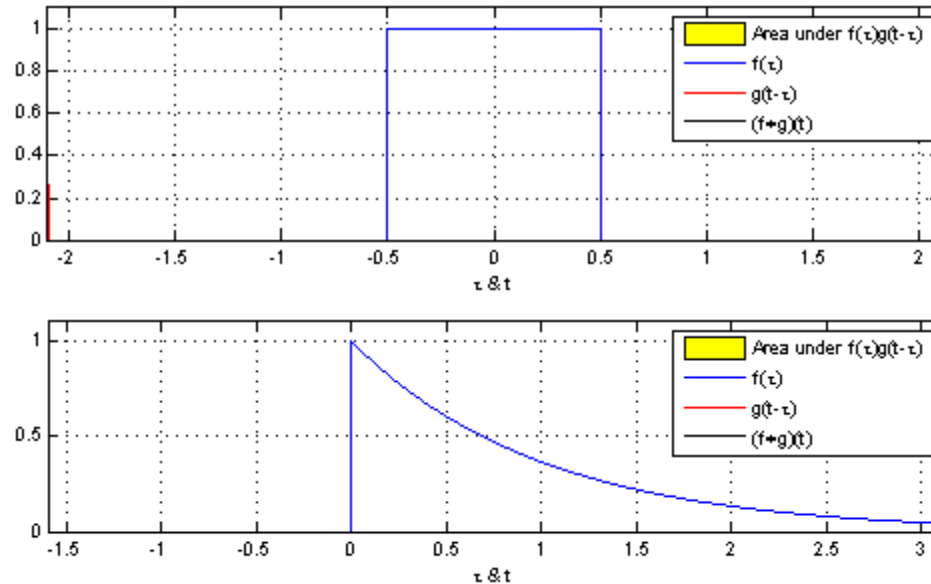


Filtrele corespund unor un trăsături ale obiectelor

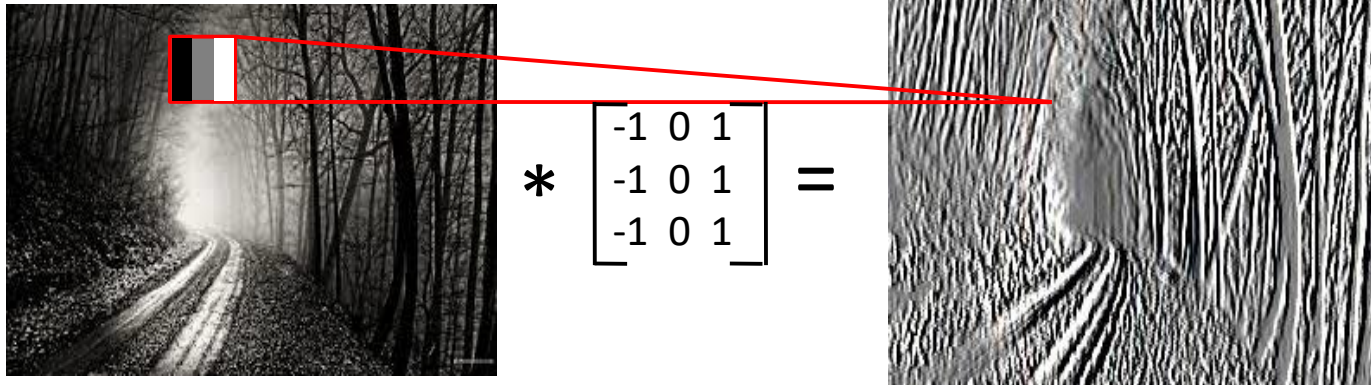


Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Activare maximă = răspunsul cel mai mare



Activare maximă = răspunsul cel mai mare

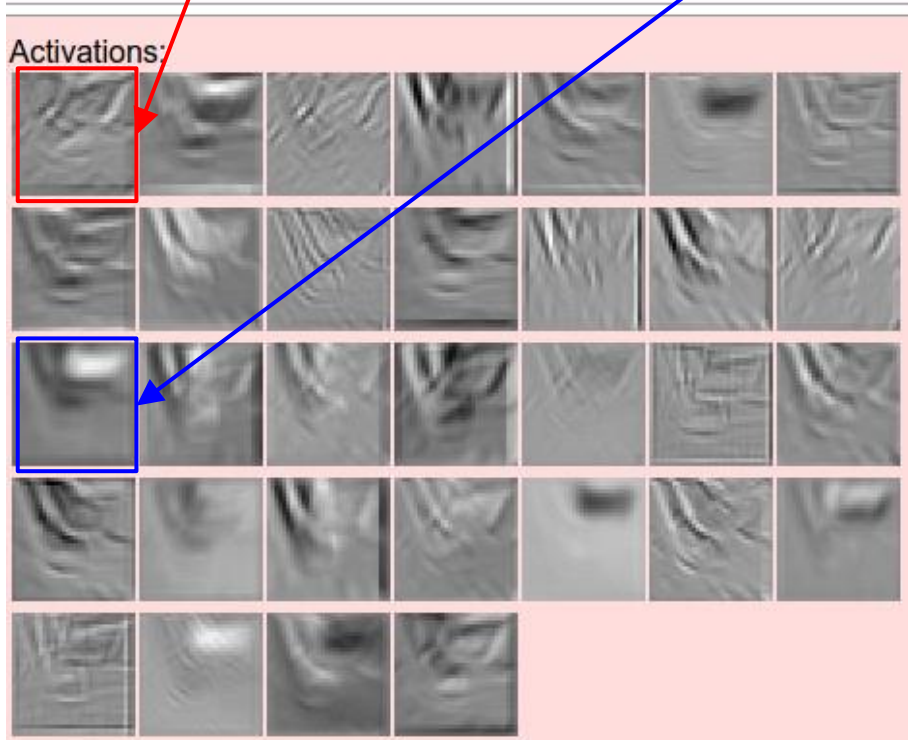




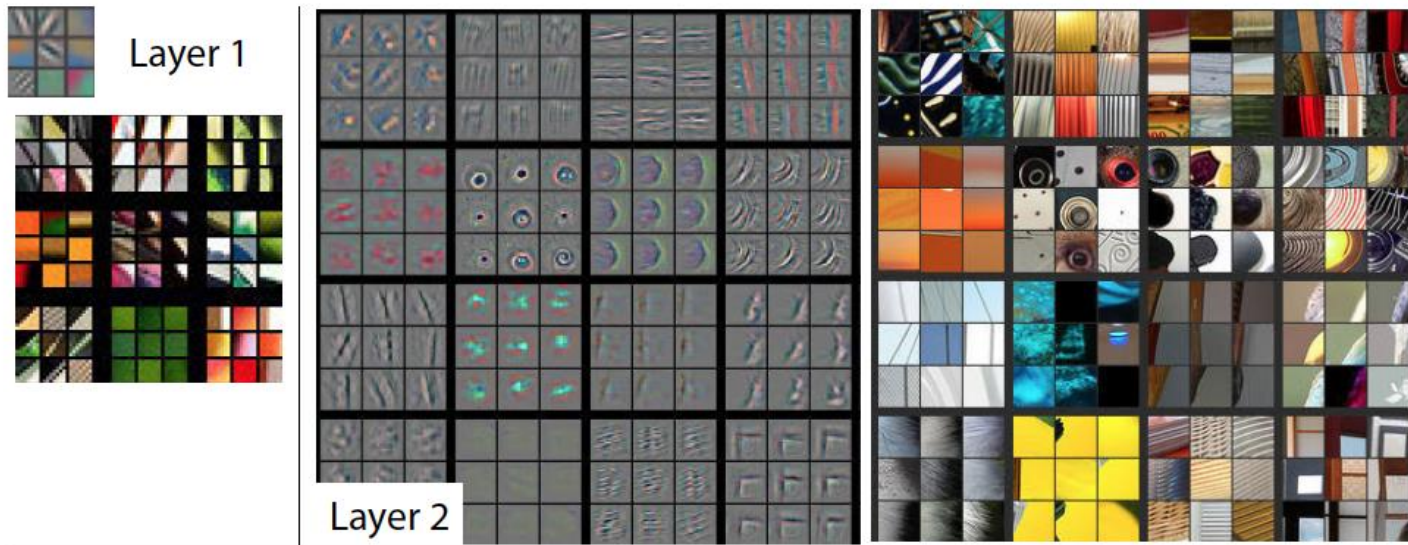
un filtru =>
un activation map



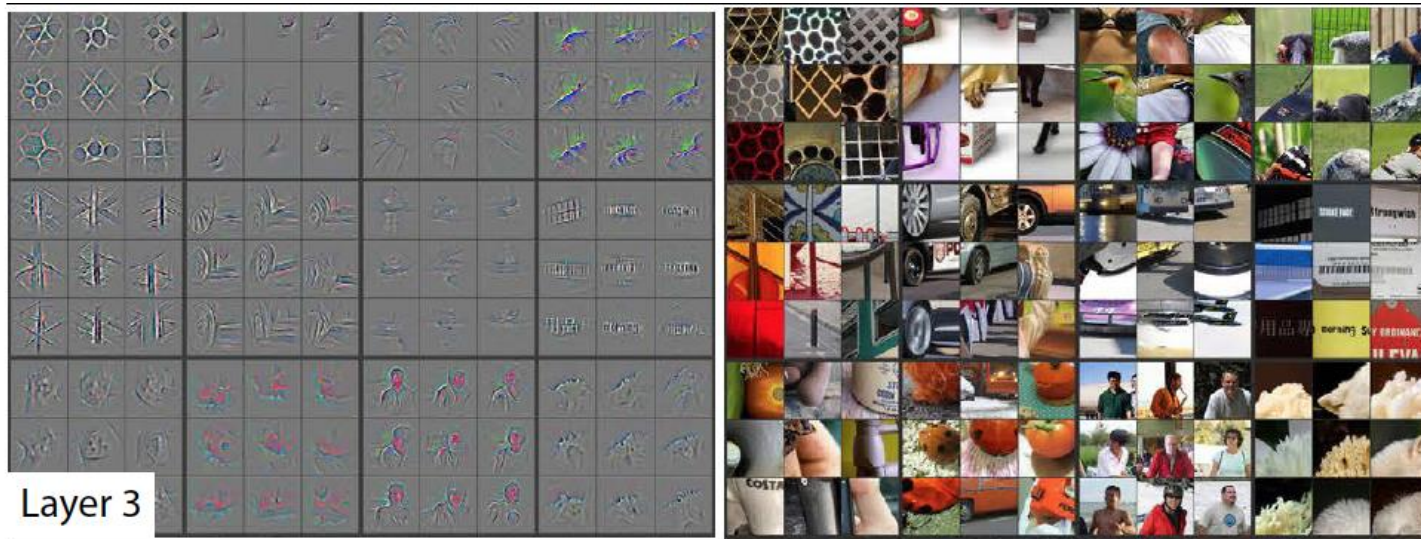
exemplu cu 32 de filtre 5x5



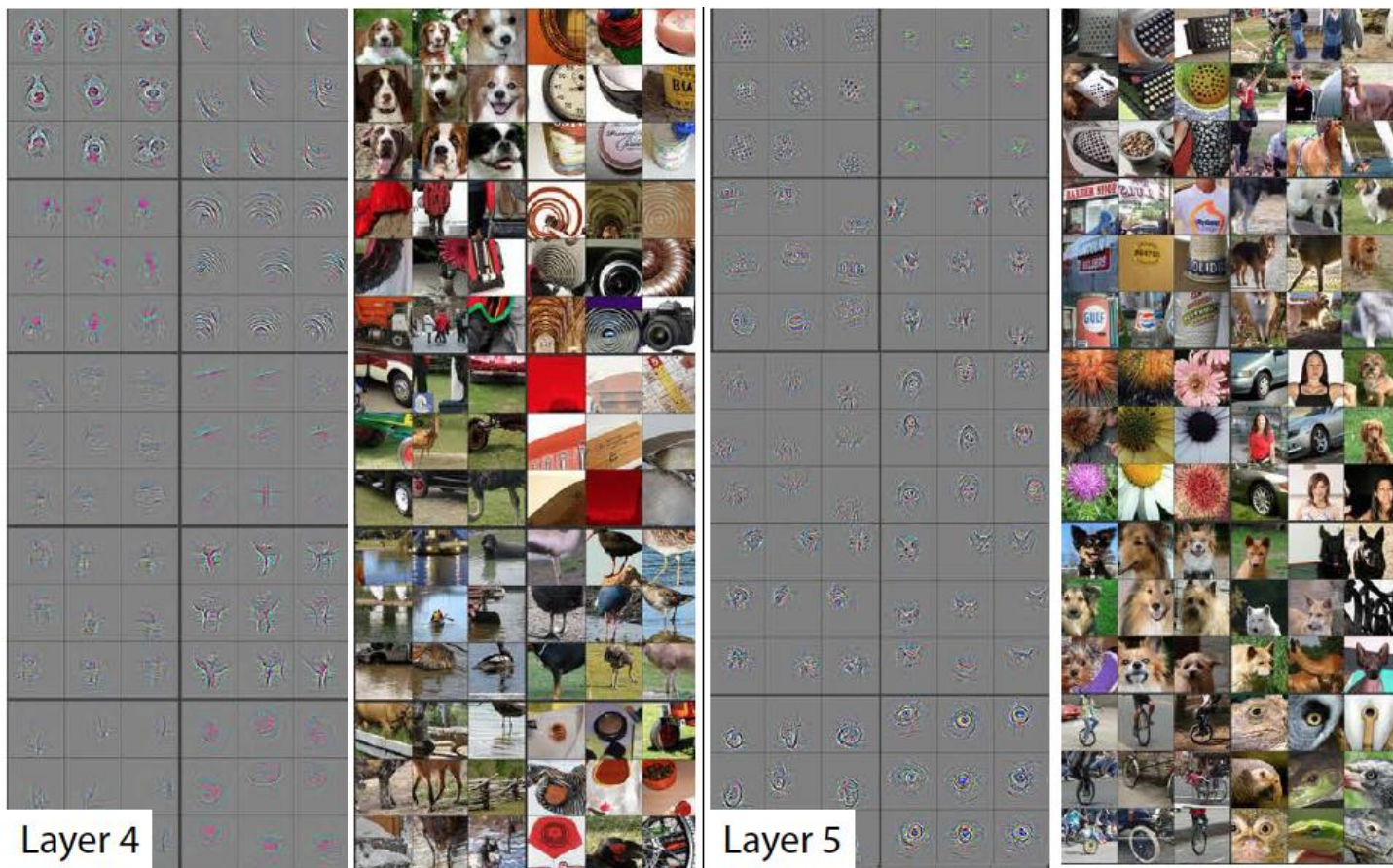
Vizualizarea filterelor învățate

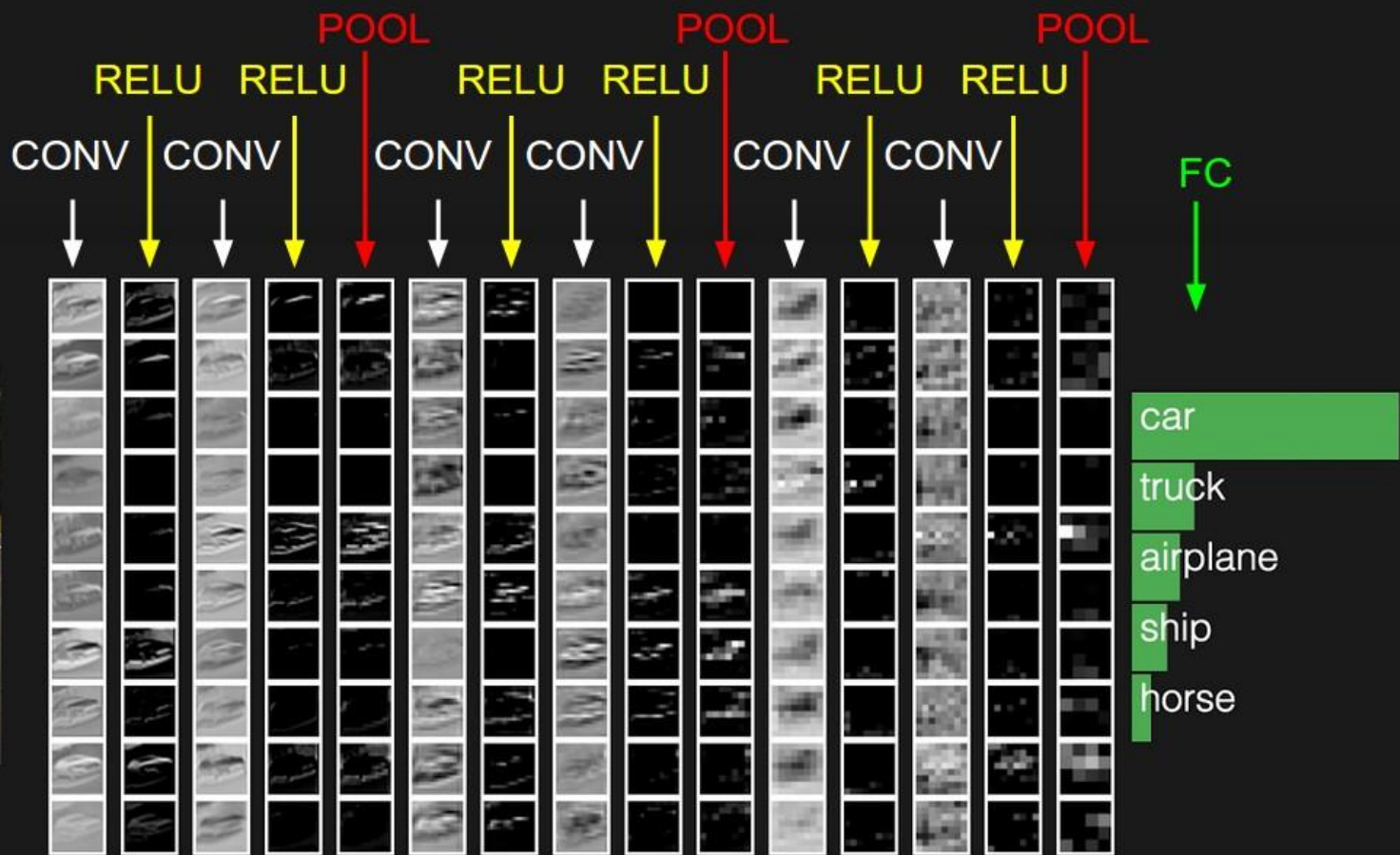


Vizualizarea filterelor învățate

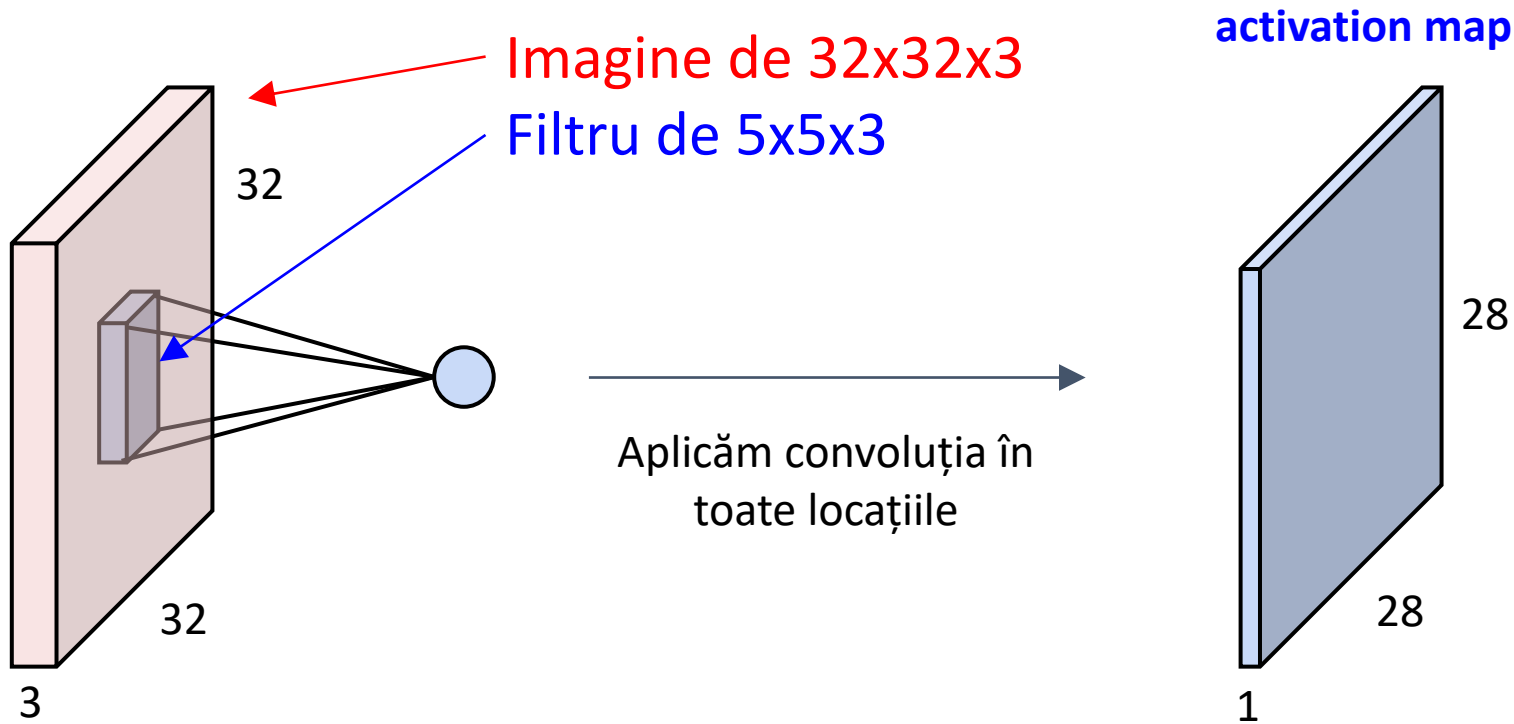


Vizualizarea filterelor învățate



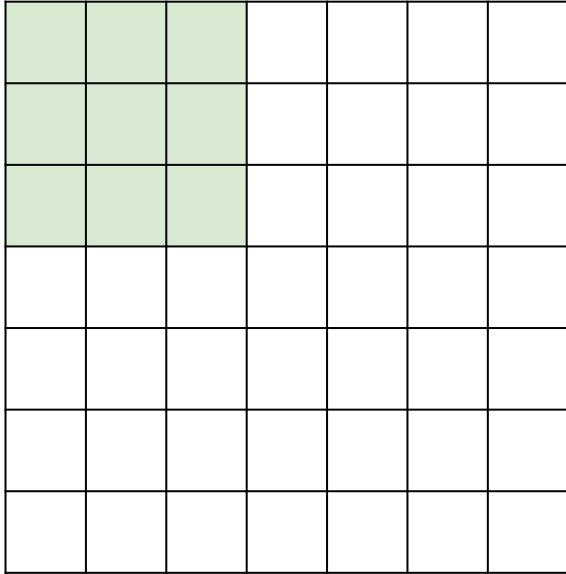


Dimensiunea spațială a unei activări



Dimensiunea spațială a unei activări

7

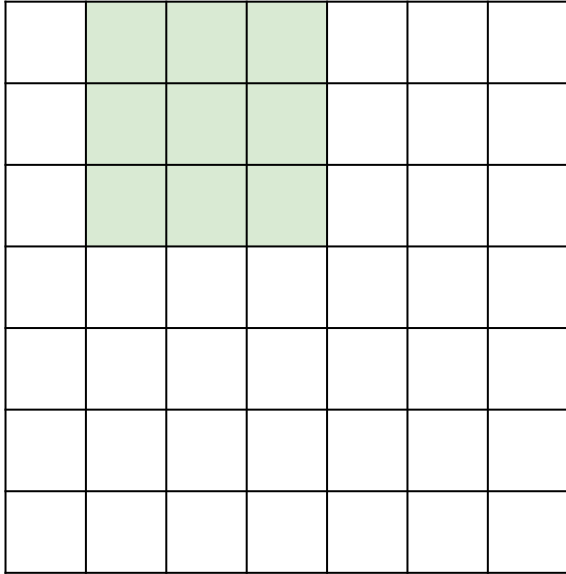


7

Imagine de 7x7 (fără adâncime)
Filtru de 3x3

Dimensiunea spațială a unei activări

7

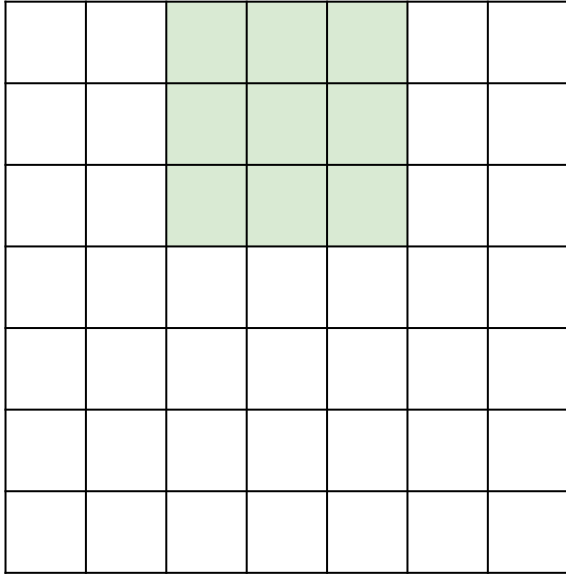


7

Imagine de 7x7 (fără adâncime)
Filtru de 3x3

Dimensiunea spațială a unei activări

7

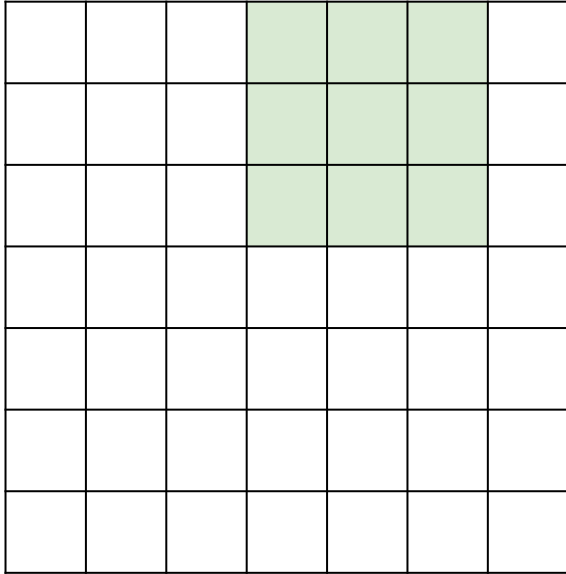


7

Imagine de 7x7 (fără adâncime)
Filtru de 3x3

Dimensiunea spațială a unei activări

7



7

Imagine de 7x7 (fără adâncime)
Filtru de 3x3

Dimensiunea spațială a unei activări

7

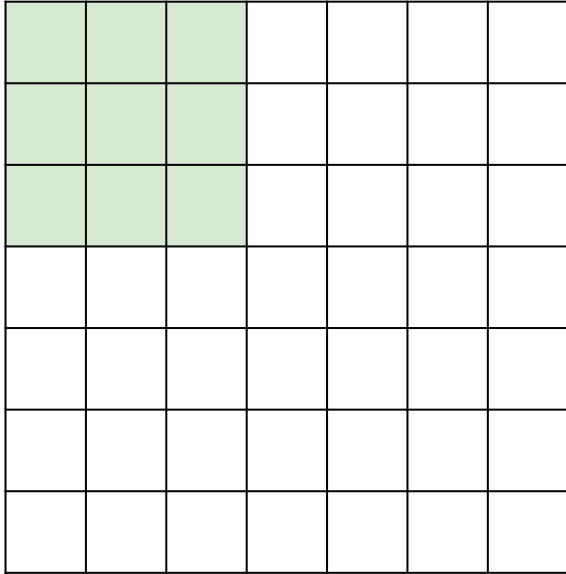
7

Imagine de 7x7 (fără adâncime)
Filtru de 3x3

=> **Output de 5x5**

Dimensiunea spațială a unei activări

7

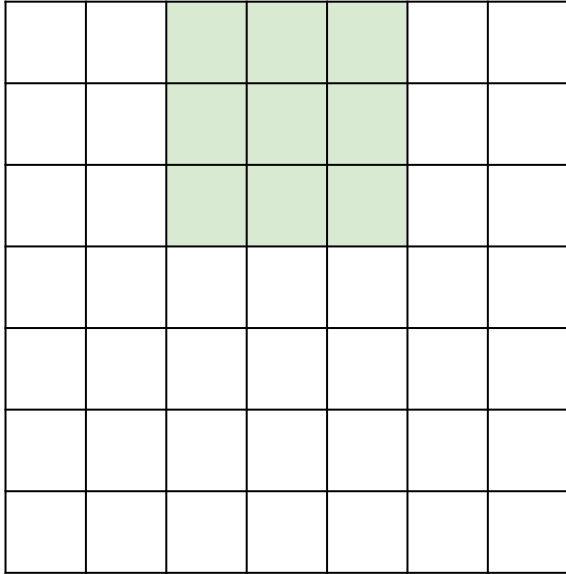


7

Imagine de 7x7 (fără adâncime)
Filtru de 3x3 aplicat cu **stride 2**

Dimensiunea spațială a unei activări

7



7

Imagine de 7x7 (fără adâncime)
Filtru de 3x3 aplicat cu **stride 2**

Dimensiunea spațială a unei activări

7

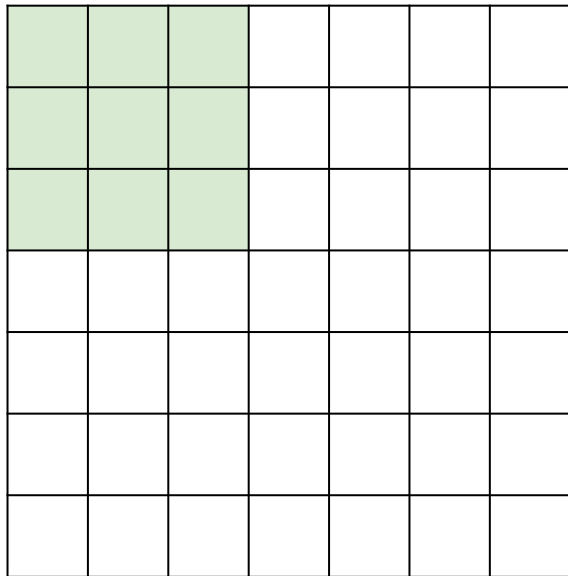
7

Imagine de 7x7 (fără adâncime)
Filtru de 3x3 aplicat cu **stride 2**

=> **Output de 3x3**

Dimensiunea spațială a unei activări

7



7

Imagine de 7x7 (fără adâncime)

Filtru de 3x3 aplicat cu **stride 3**?

Nu se potrivește!

Nu putem aplica un filtru de 3x3
pe o imagine de 7x7 folosind stride
3

N

			F			
	F					

N

Mărimea activării:

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

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: de obicei 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 convoluționale cu stride 1, cu filtre de dimensiune $F \times F$, și bordură de $(F-1)/2$. (menține dimensiunea imaginii de input)

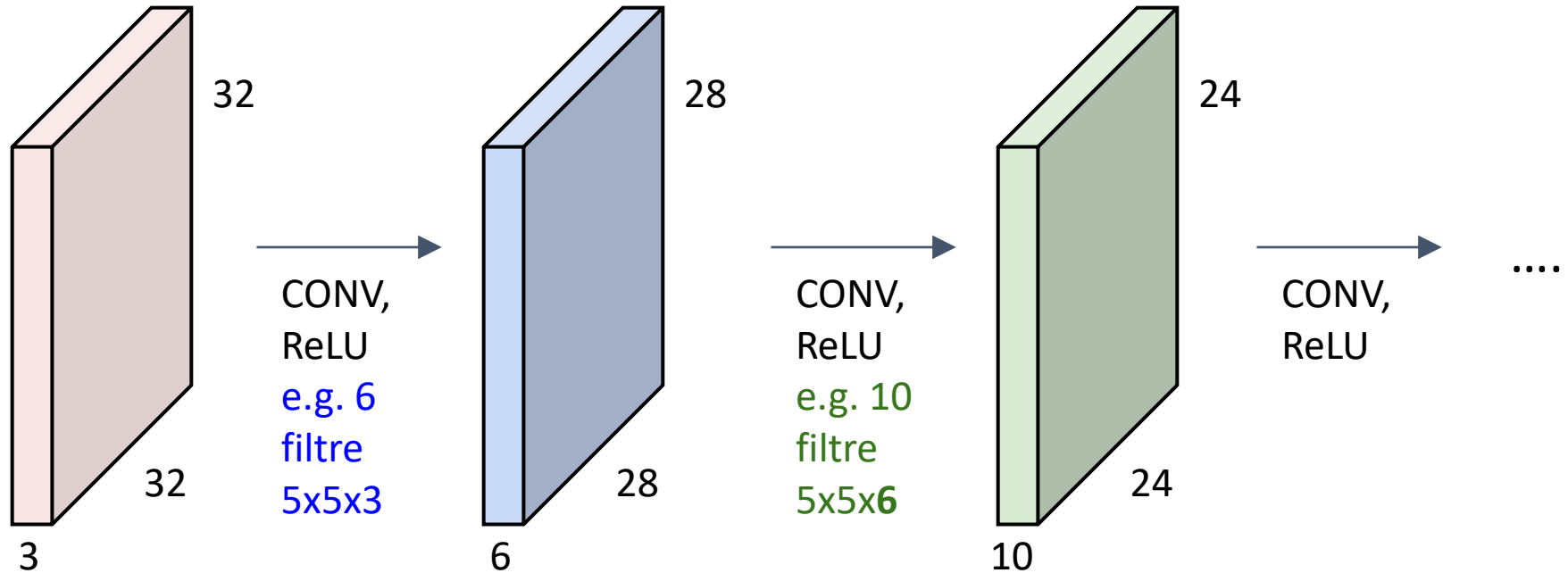
e.g. $F = 3 \Rightarrow$ bordăm cu 1 pixel (valoare 0)

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

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

Aplicând convoluții cu filtre de 5x5 în mod repetat pe un input de 32x32 micșorează volumul (32 => 28 => 24 ...)

Micșorarea prea rapidă a volumului nu produce rezultate optime

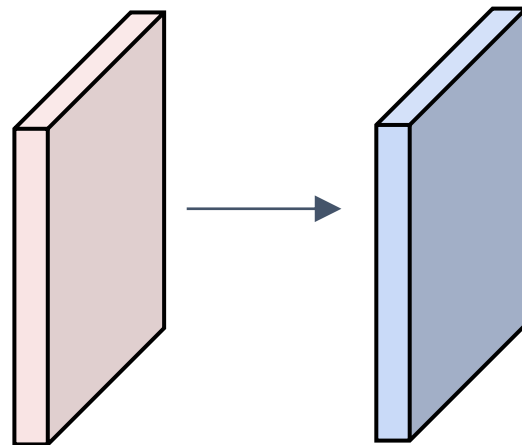


Exemplu:

Volum de input: **32x32x3**

10 filtre de 5x5 cu stride 1, bordură 2

Mărimea volumului de output: ?



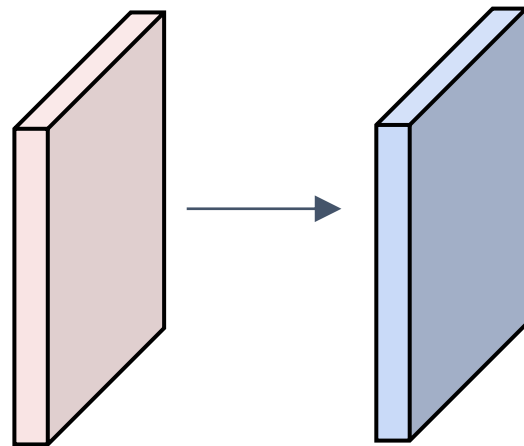
Exemplu:

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**

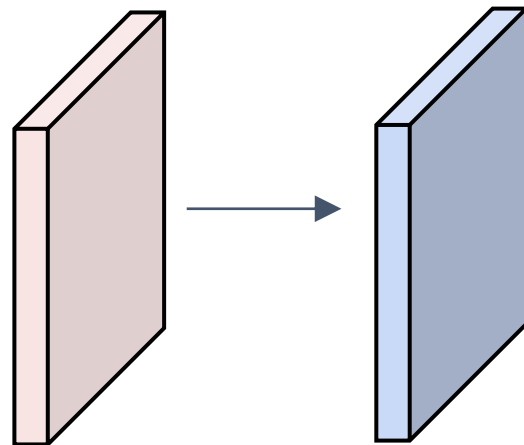


Exemplu:

Volum de input: **32x32x3**

10 filtre de 5x5 cu stride 1, bordură 2

Numărul de parametri al acestui strat: ?



Exemplu:

Volum de input: **32x32x3**

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

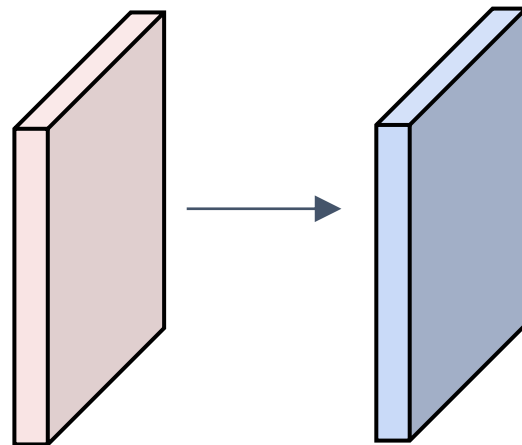
Numărul de parametri al acestui strat: ?

Fiecare filtru are **5*5*3** + 1 = **76**

parametrii

(+1 pentru bias)

=> **76*10** = **760**



Setări comune:

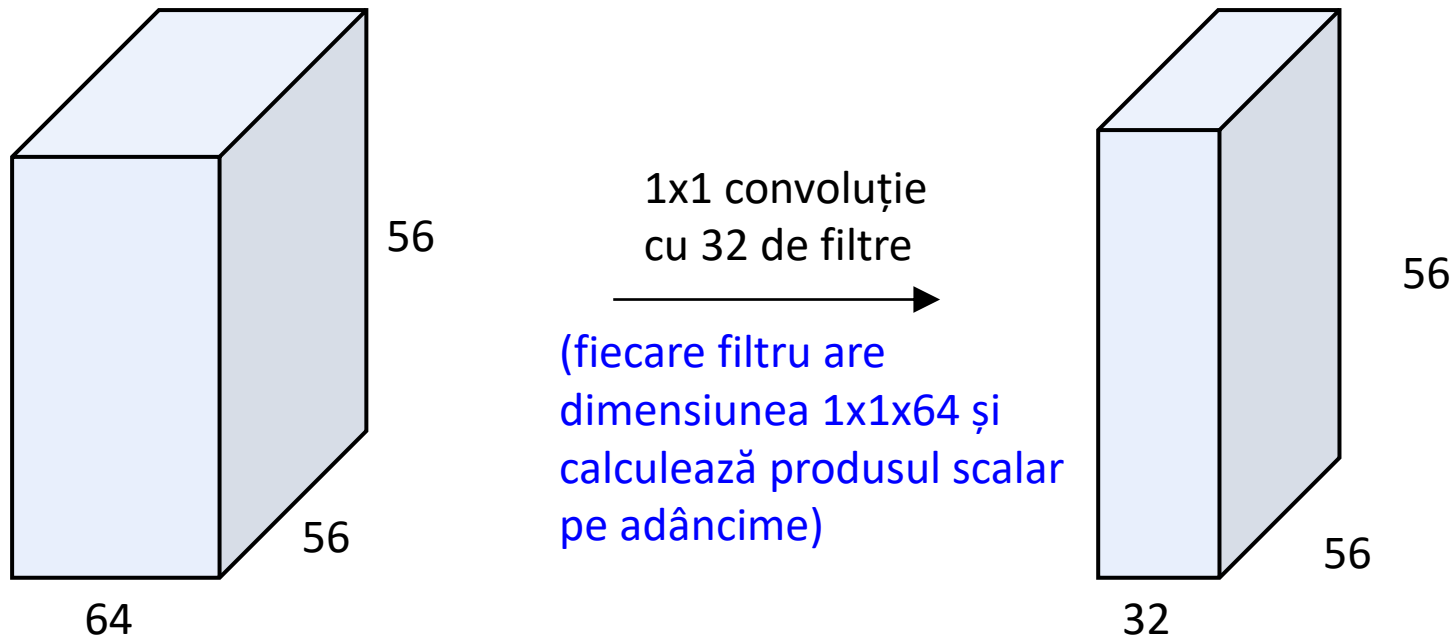
Summary. To summarize, the Conv Layer:

- 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.

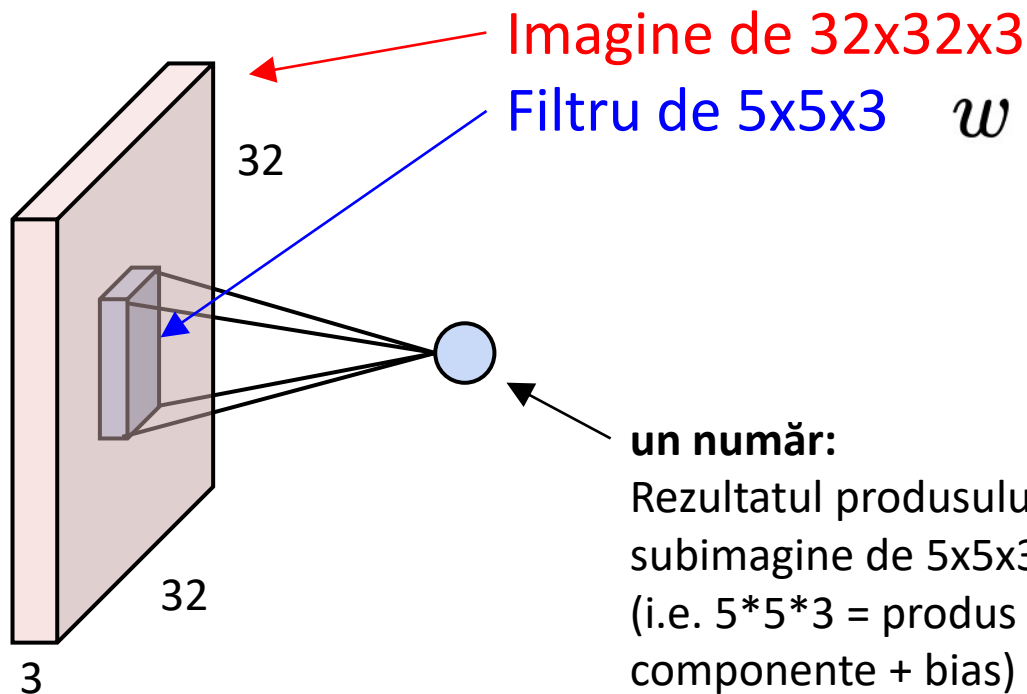
$K =$ (puteri ale lui 2, e.g. 32, 64, 128, 512)

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

Are sens să folosim chiar filtre de 1x1



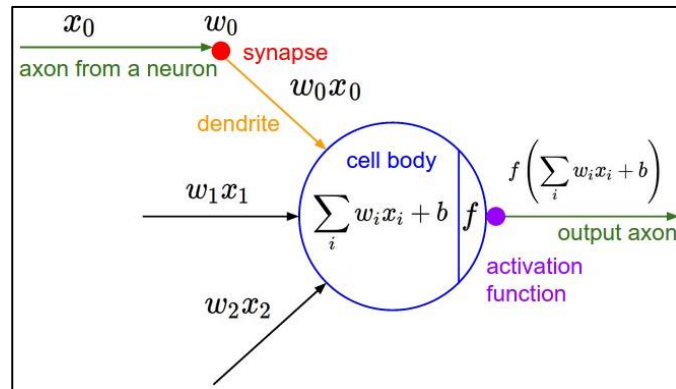
Stratul convoluțional din punct de vedere biologic



un număr:

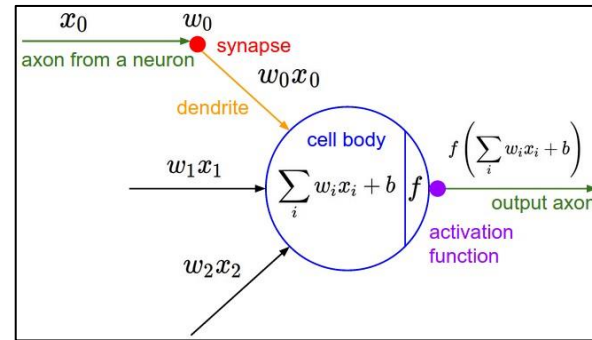
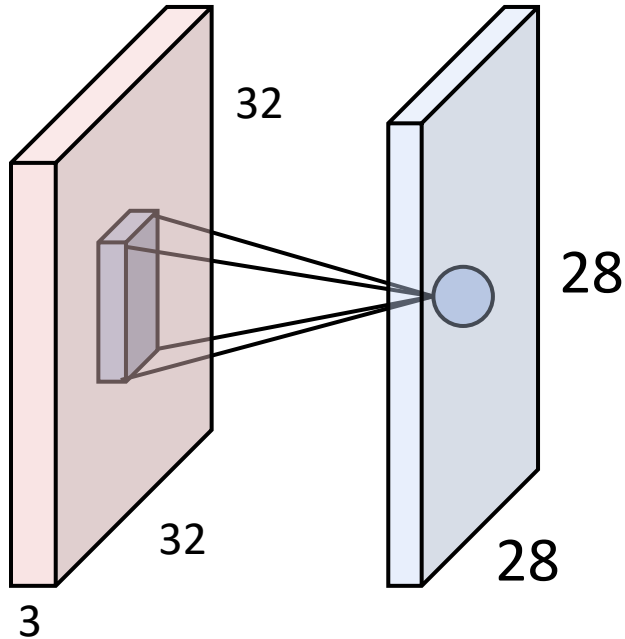
Rezultatul produsului scalar dintre filtru și o subimagine de 5x5x3 pixeli
(i.e. $5*5*3 =$ produs scalar pe 75 de componente + bias)

$$w^T x + b$$



...este doar un neuron cu conexiuni locale

Stratul convoluțional din punct de vedere biologic

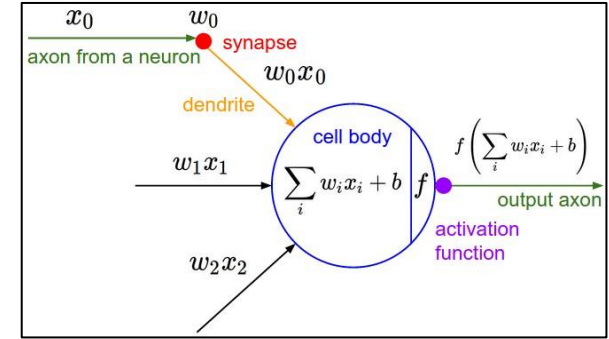
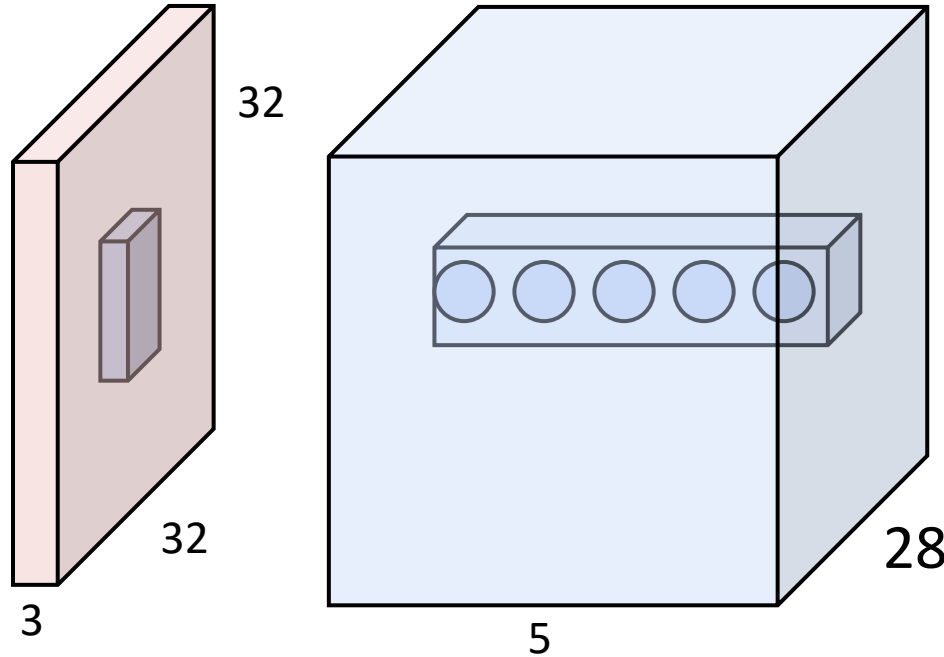


Un activation map este matrice cu output-urile a 28x28 neuroni:

1. Fiecare este conectat la o regiune mică din input
2. Toții neuronii au aceeași parametrii

filtru 5x5 = câmp receptiv de 5x5 pentru fiecare neuron

Stratul convoluțional din punct de vedere biologic



28

E.g. cu 5 filtre,
stratul convoluțional este format din
neuroni aranjați într-un grid
(28x28x5)

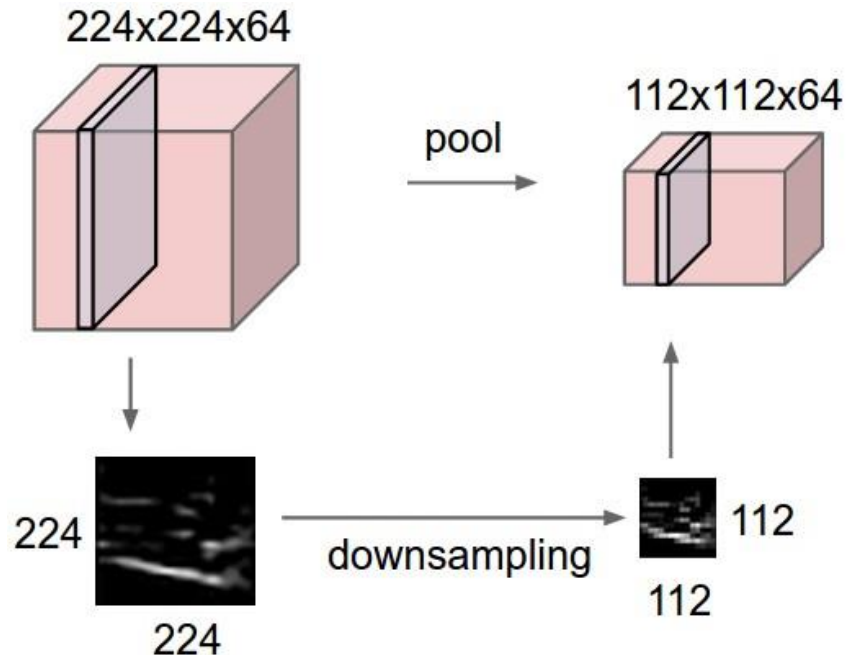
Pentru o regiune vor fi 5 neuroni
diferiți (toți primesc același input)

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



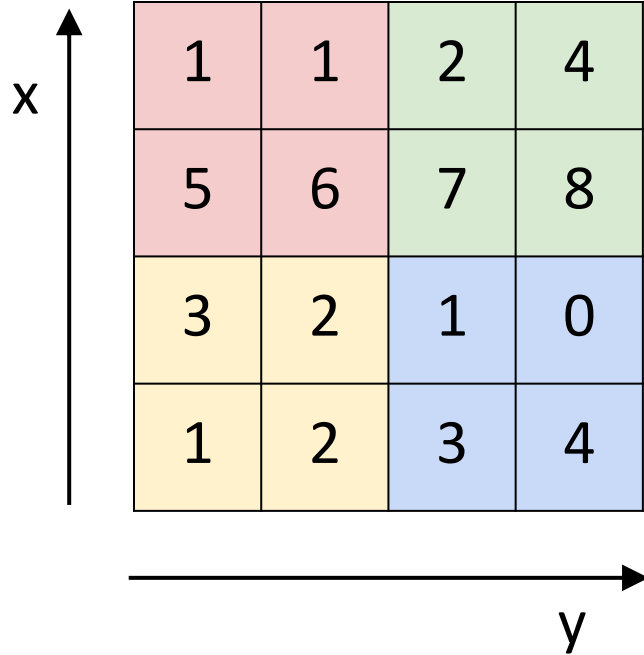
Stratul de pooling

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

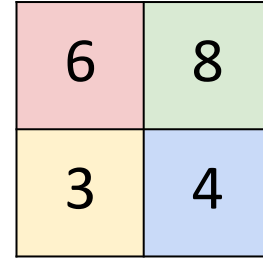


MAX Pooling

Un activation map



max pooling cu filtre
de 2x2 și stride 2



Setări comune:

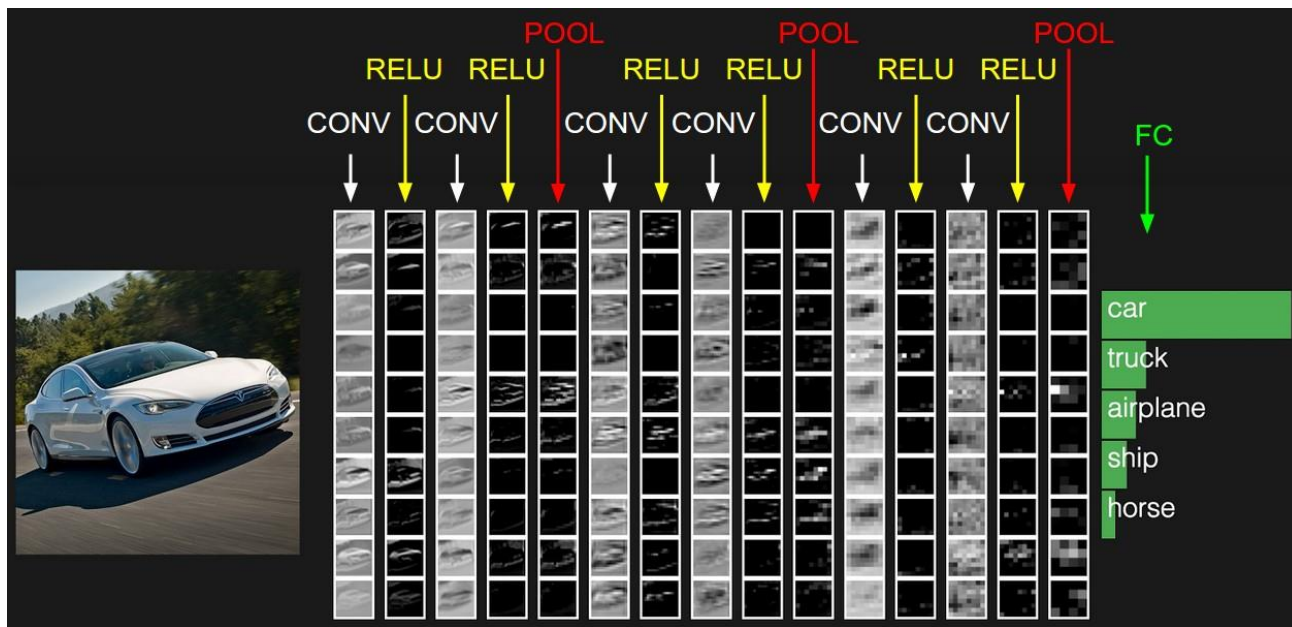
- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires three hyperparameters:
 - their spatial extent F ,
 - the stride S ,
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
 - $W_2 = (W_1 - F)/S + 1$
 - $H_2 = (H_1 - F)/S + 1$
 - $D_2 = D_1$
- Introduces zero parameters since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

$F = 2, S = 2$

$F = 3, S = 2$

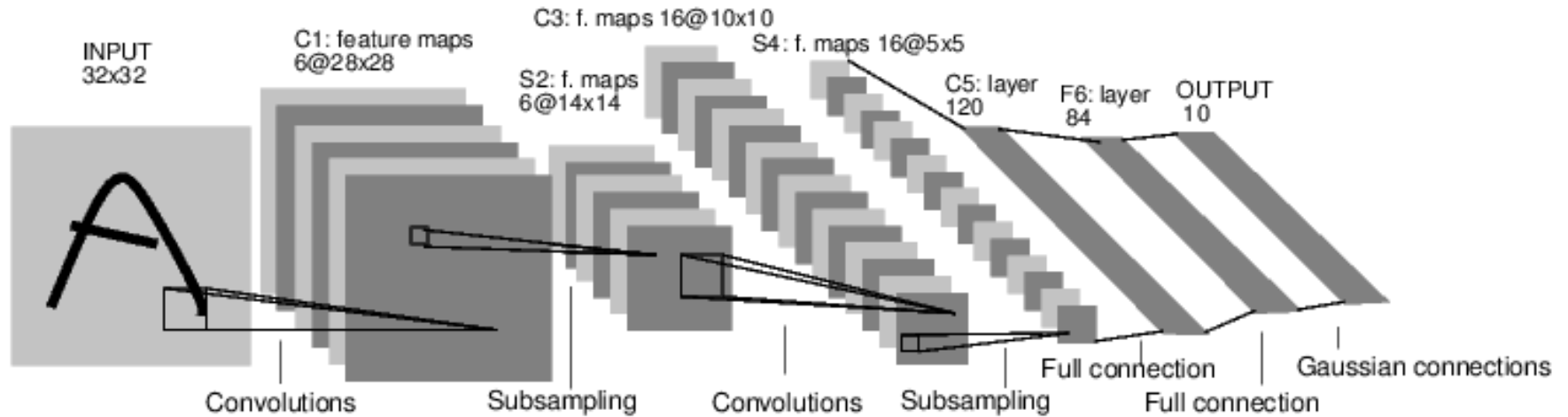
Straturi cu conexiuni complete (FC layer)

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



Studiu de caz: LeNet-5

[LeCun et al., 1998]



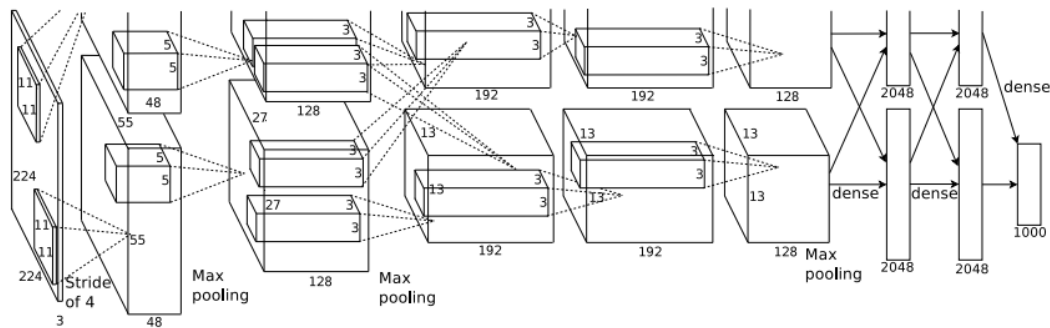
Straturi convoluționale cu filtre de 5x5, aplicate cu stride 1

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

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

Studiu de caz: AlexNet

[Krizhevsky et al. 2012]



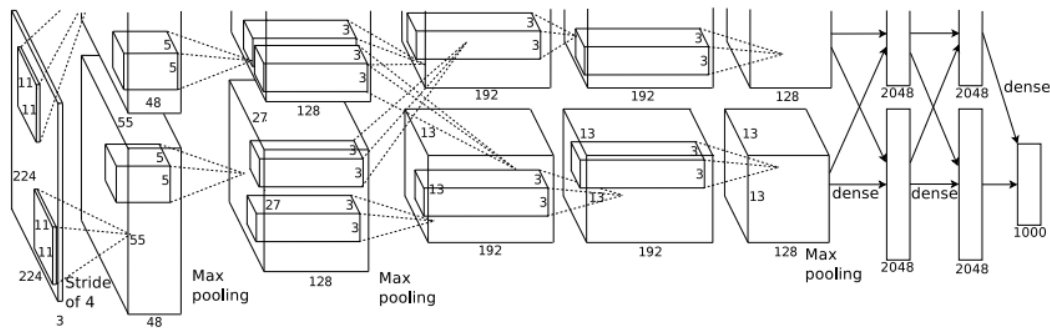
Input: imagini de 227x227x3

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

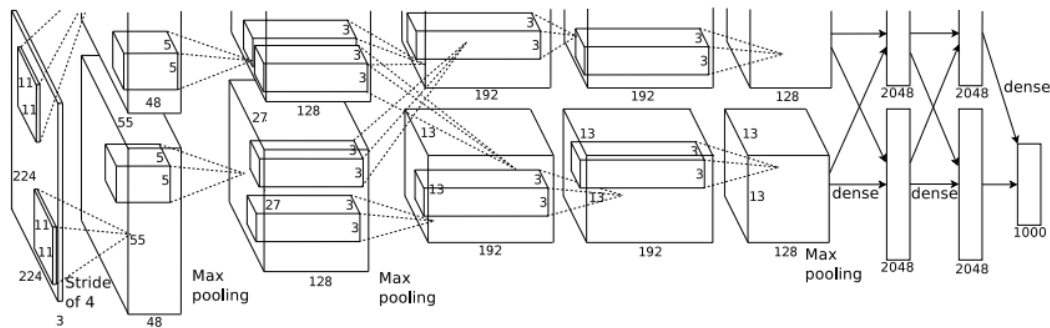
=>

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

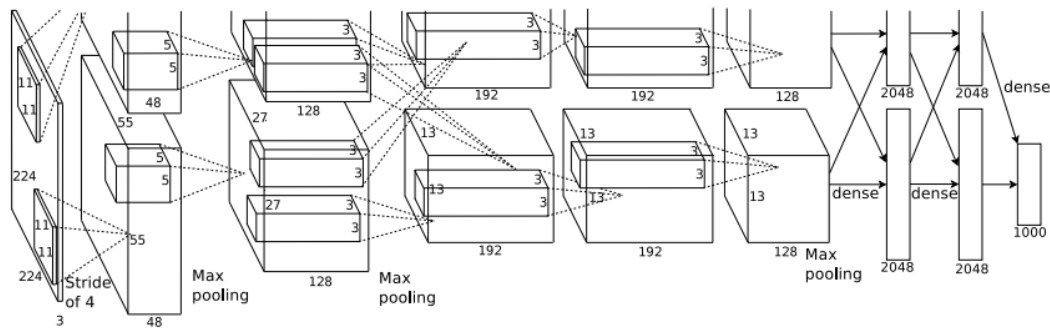
=>

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

Parametrii: $(11*11*3)*96 = \mathbf{35K}$

Studiu de caz: AlexNet

[Krizhevsky et al. 2012]



Input: imagini de 227x227x3

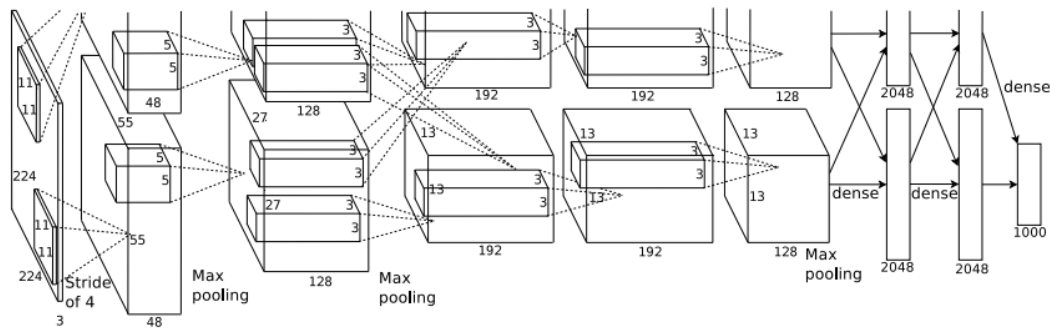
CONV1: 55x55x96

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

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

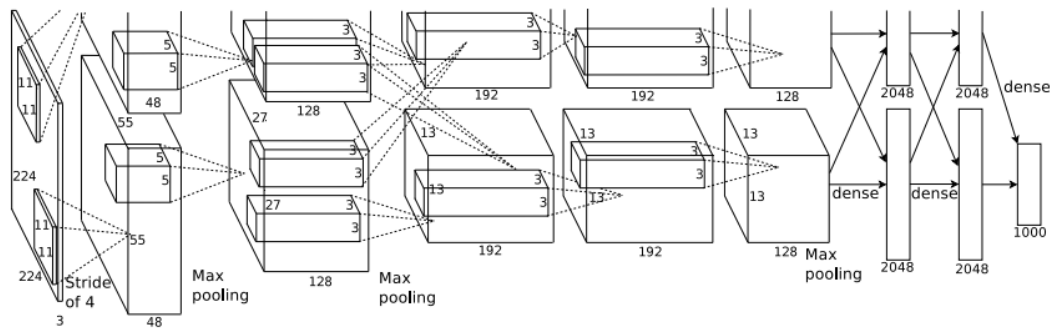
Al doilea strat (POOL1): filtre de 3x3 aplicate cu stride 2

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

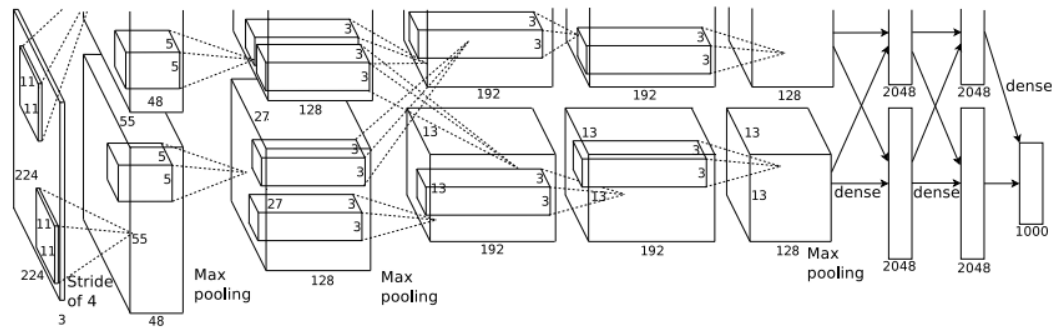
Al doilea strat (POOL1): filtre de 3x3 aplicate cu stride 2

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

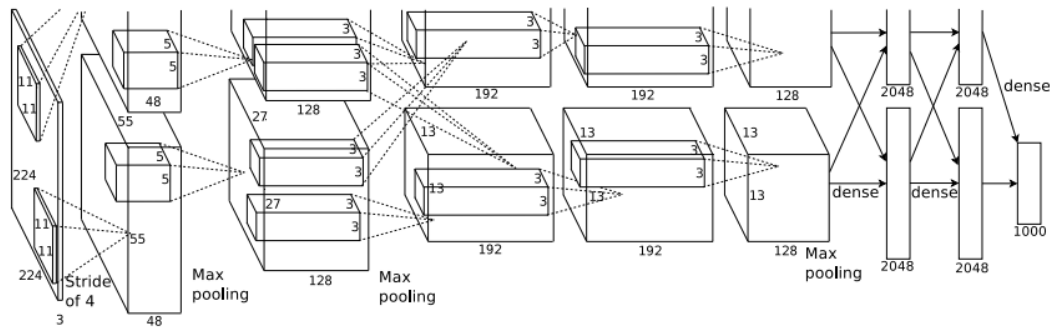
[13x13x256] **CONV5**: 256 filtre de 3x3 cu stride 1, bordură 1

[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 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

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: $224*224*3=150K$ parametrii: 0

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Memorie totală: 24M * 4 bytes ~ = 93MB / imagine (doar propagarea înainte, ~x2 înapoi)

Numărul total de parametrii: 138M

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

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

Observații:

Cea mai multă
memorie este
consumată în primele
straturi CONV

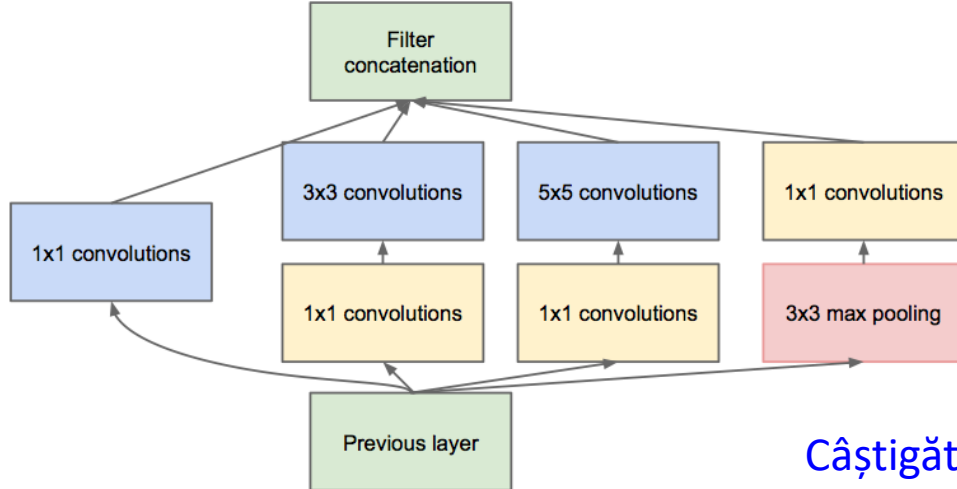
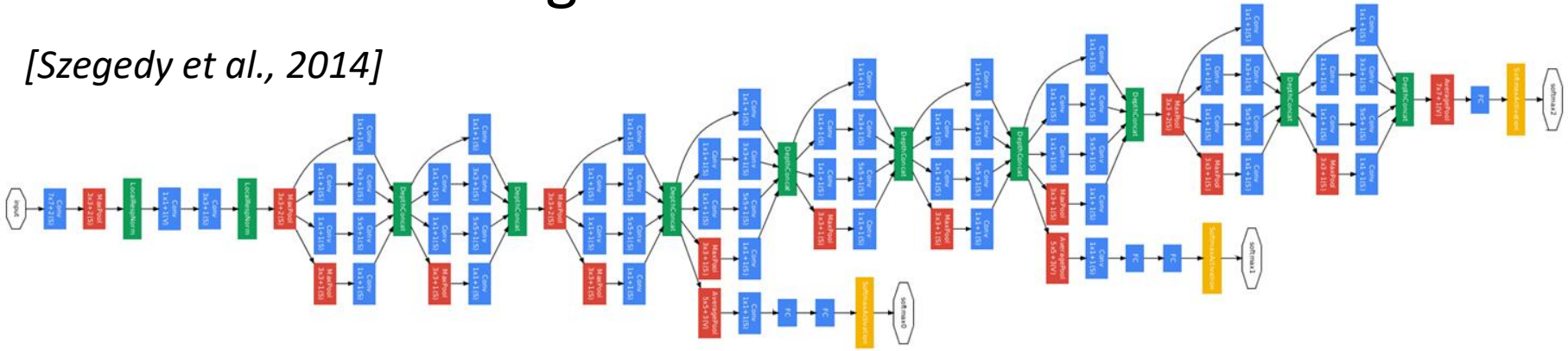
Cei mai mulți
parametrii în ultimele
straturi FC

Memorie totală: 24M * 4 bytes ~ = 93MB / 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 parametri! (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

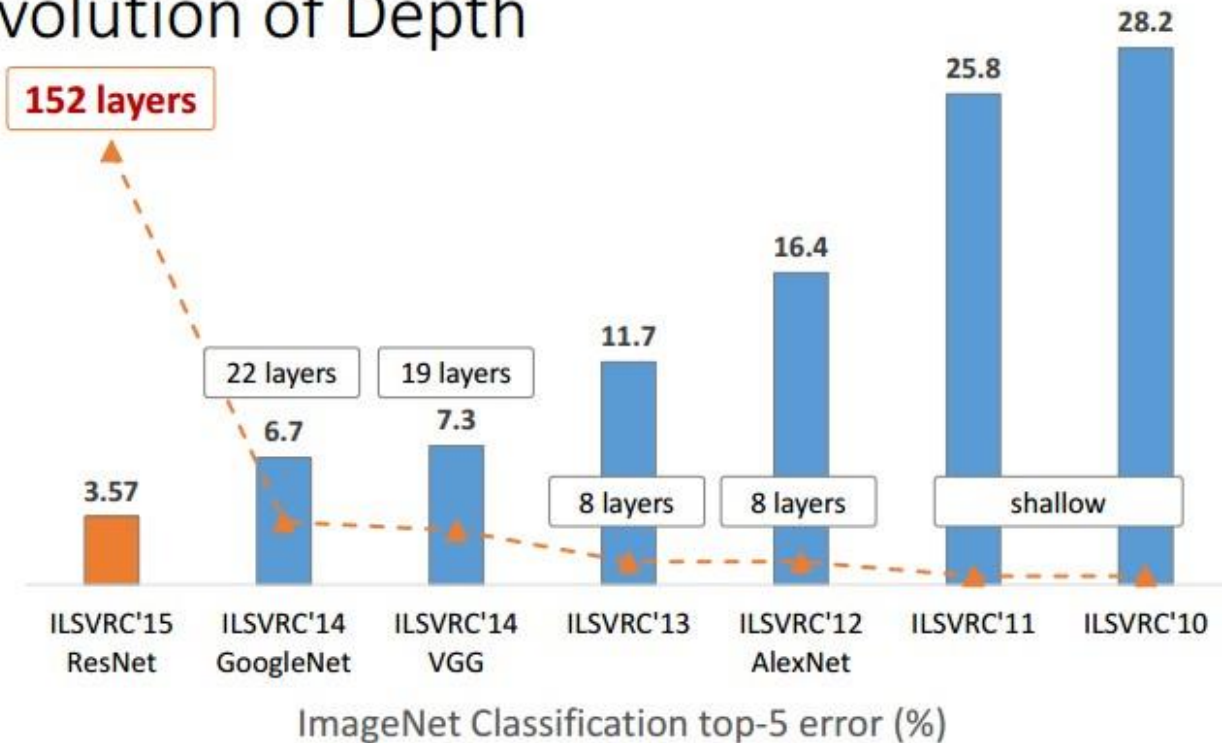
- **1st places** in all five main tracks
 - ImageNet Classification: “Ultra-deep” (quote Yann) **152-layer** nets
 - ImageNet Detection: **16%** better than 2nd
 - ImageNet Localization: **27%** better than 2nd
 - COCO Detection: **11%** better than 2nd
 - COCO Segmentation: **12%** better than 2nd

*improvements are relative numbers



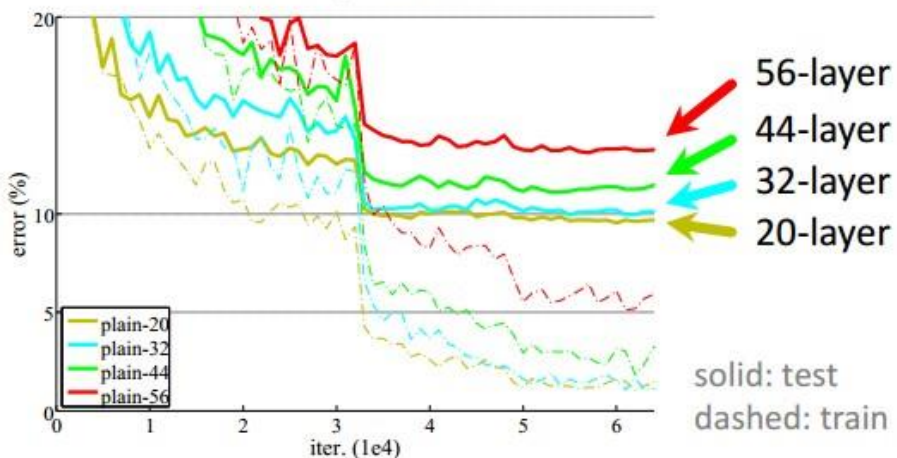
Kaiming He, Xiangyu Zhang, Shaoqing Ren, & Jian Sun. “Deep Residual Learning for Image Recognition”. arXiv 2015.

Revolution of Depth

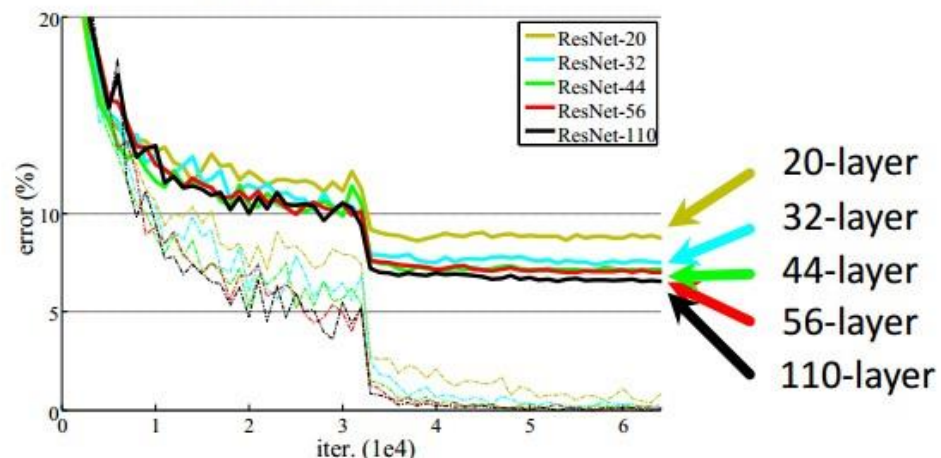


CIFAR-10 experiments

CIFAR-10 plain nets



CIFAR-10 ResNets





Studiu de caz: ResNet

[He et al., 2015]

Câștigătorul ILSVRC 2015 (3.6% eroare top 5)

Microsoft Research

Revolution of Depth

AlexNet, 8 layers (ILSVRC 2012)		VGG, 19 layers (ILSVRC 2014)		ResNet, 152 layers (ILSVRC 2015)
------------------------------------	---	---------------------------------	---	-------------------------------------

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)

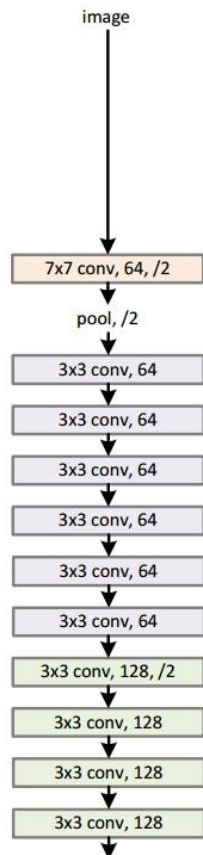
ICCV15
International Conference on Computer Vision

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

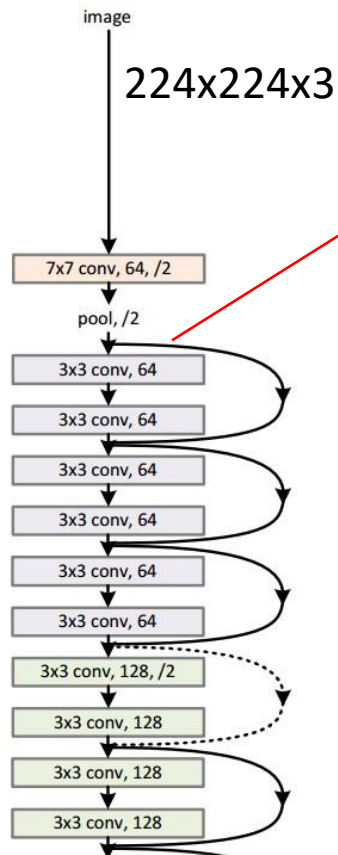
Studiu de caz: ResNet

[He et al., 2015]

34-layer plain



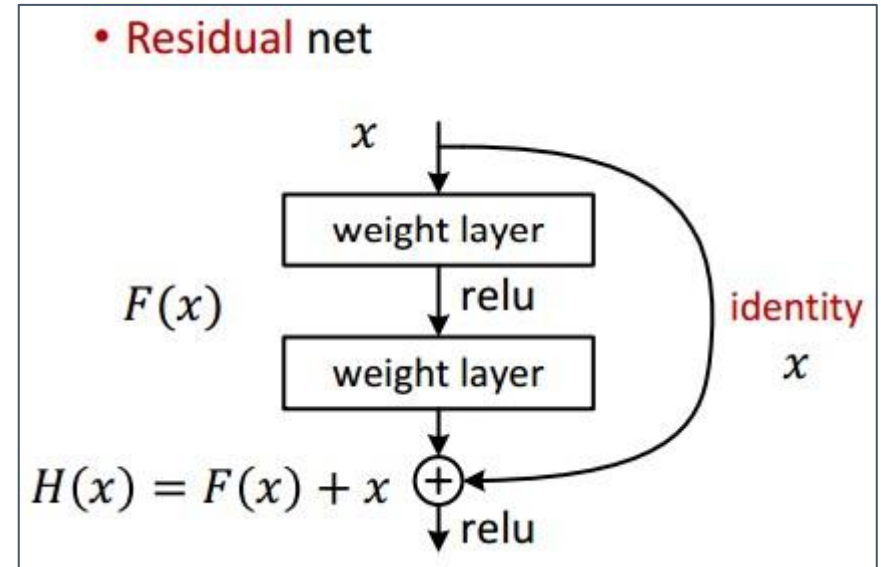
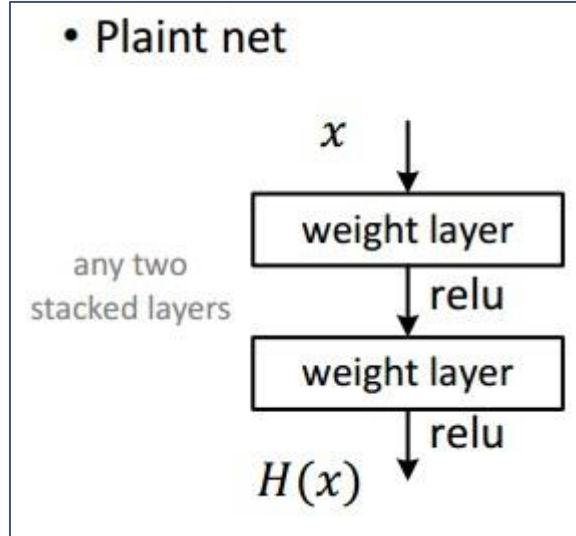
34-layer residual



Dimensiunea spațială
de 56x56!

Studiu de caz: ResNet

[He et al., 2015]



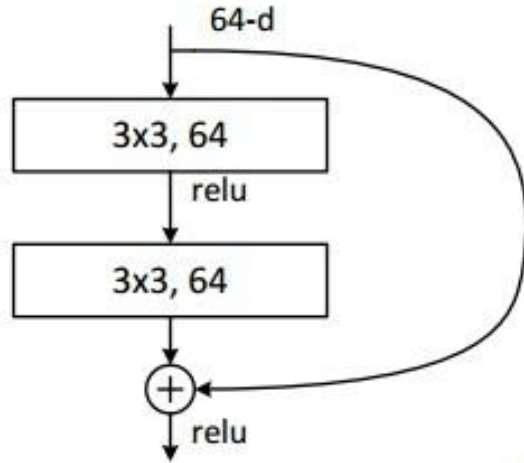
Studiu de caz: ResNet

[He et al., 2015]

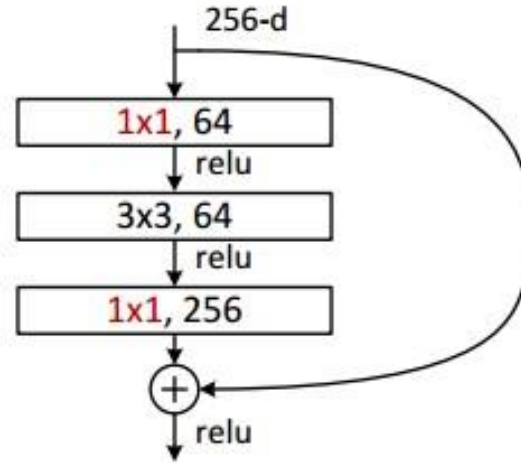
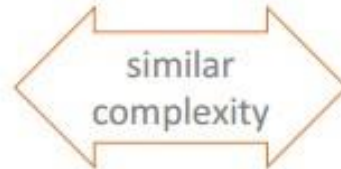
- Normalizare batch după fiecare strat CONV
- Inițializare Xavier / 2
- SGD cu moment 0.9
- Rată de învățare: 0.1, împărțită la 10 când eroare pe validare atinge un platou
- Mărimea unui mini-batch 256
- Degradarea ponderilor cu 10^{-5}
- Fără dropout!

Studiu de caz: ResNet

[He et al., 2015]



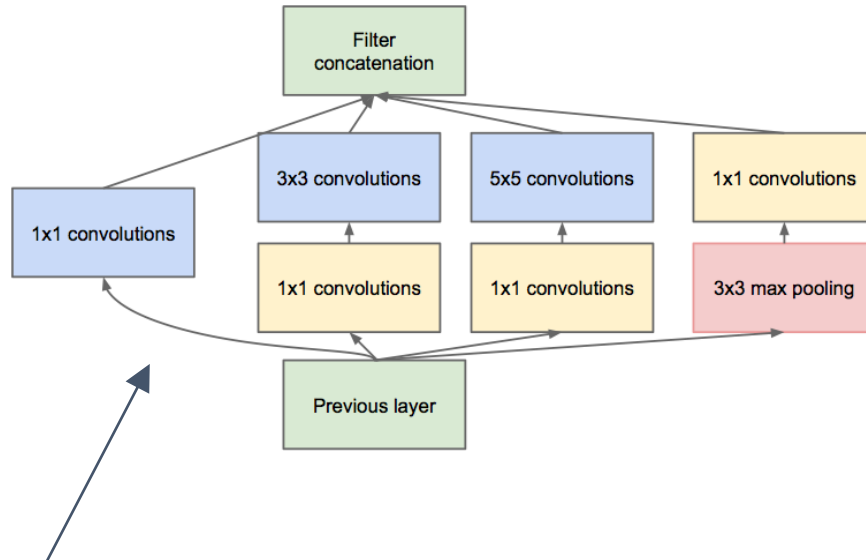
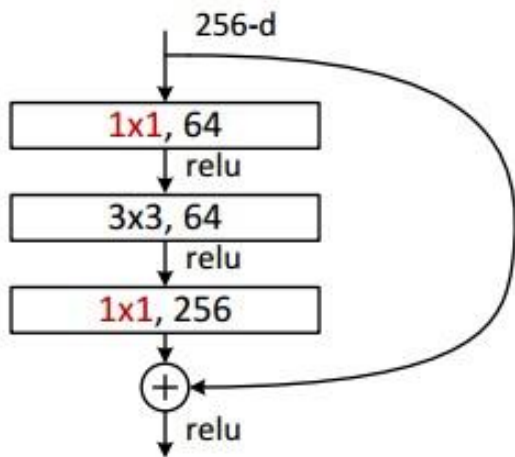
all-3x3



bottleneck
(for ResNet-50/101/152)

Studiu de caz: ResNet

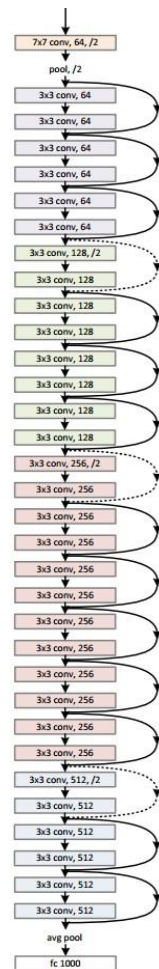
[He et al., 2015]



(truc utilizat și în GoogLeNet)

[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	3×3 max pool, stride 2				
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9



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:

$[(\text{CONV-RELU})^*N\text{-POOL?}]^*M\text{-(FC-RELU)}^*K, \text{SOFTMAX}$

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

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