

Aula 02 – Redes Neurais Convolucionais

Prof. João Fernando Mari

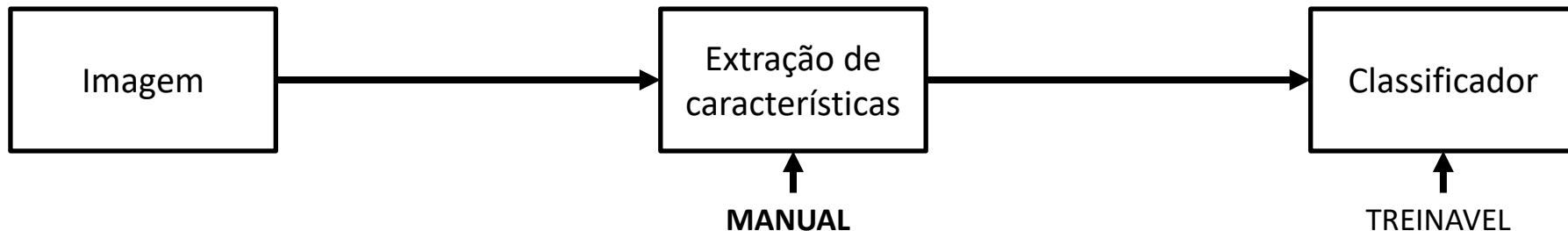
joaofmari.github.io

joaof.mari@ufv.br

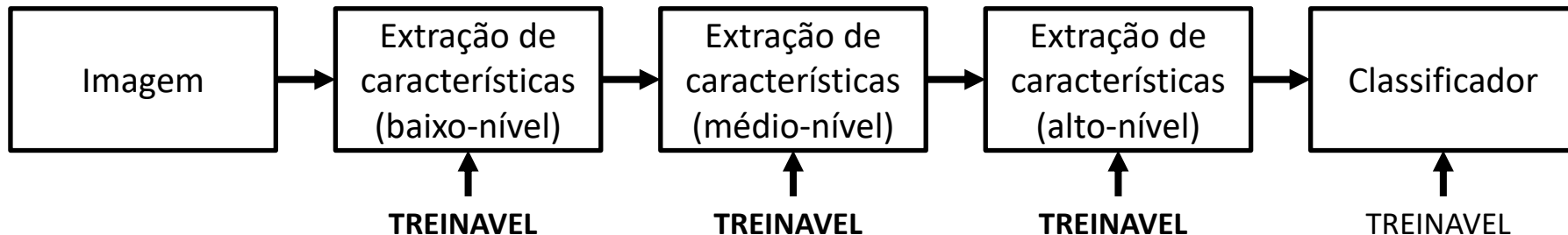
- Pipelines de classificação
- Perceptron de múltiplas camadas (MLP)
- Redes Neurais Convolucionais (CNNs)
- Camada convolucional
- Camada de pooling
- Função de ativação
- Camada completamente conectada
- Camada de saída – softmax
- Função de perda (loss)
- Otimizadores
- Arquiteturas
- Bibliotecas e desenvolvimento
- Conjuntos de imagens

Pipelines de classificação

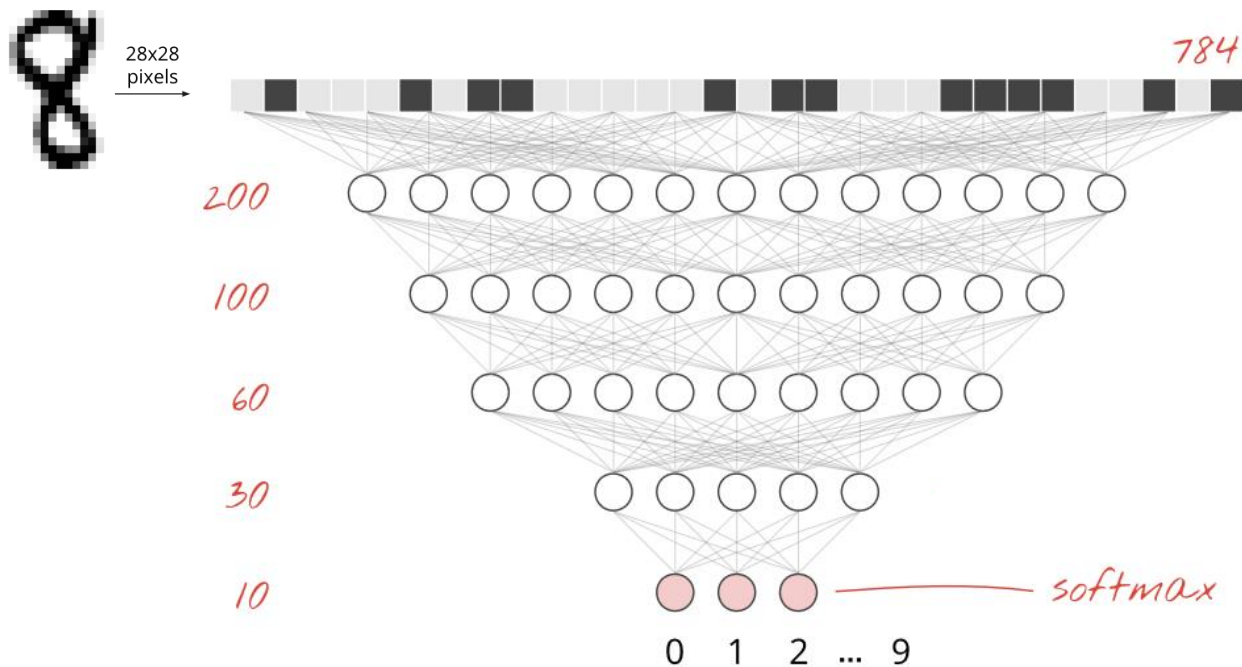
Pipeline clássico de classificação de imagens



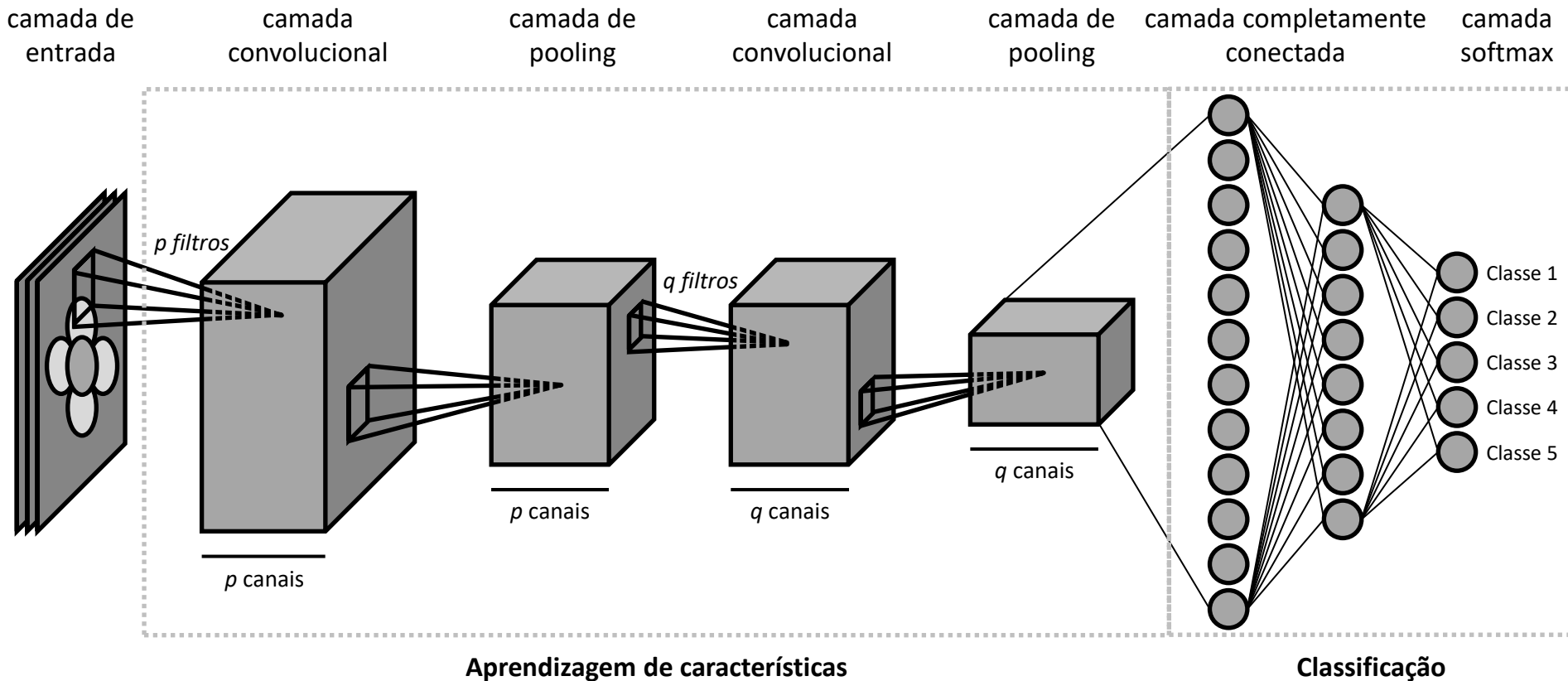
Deep Learning



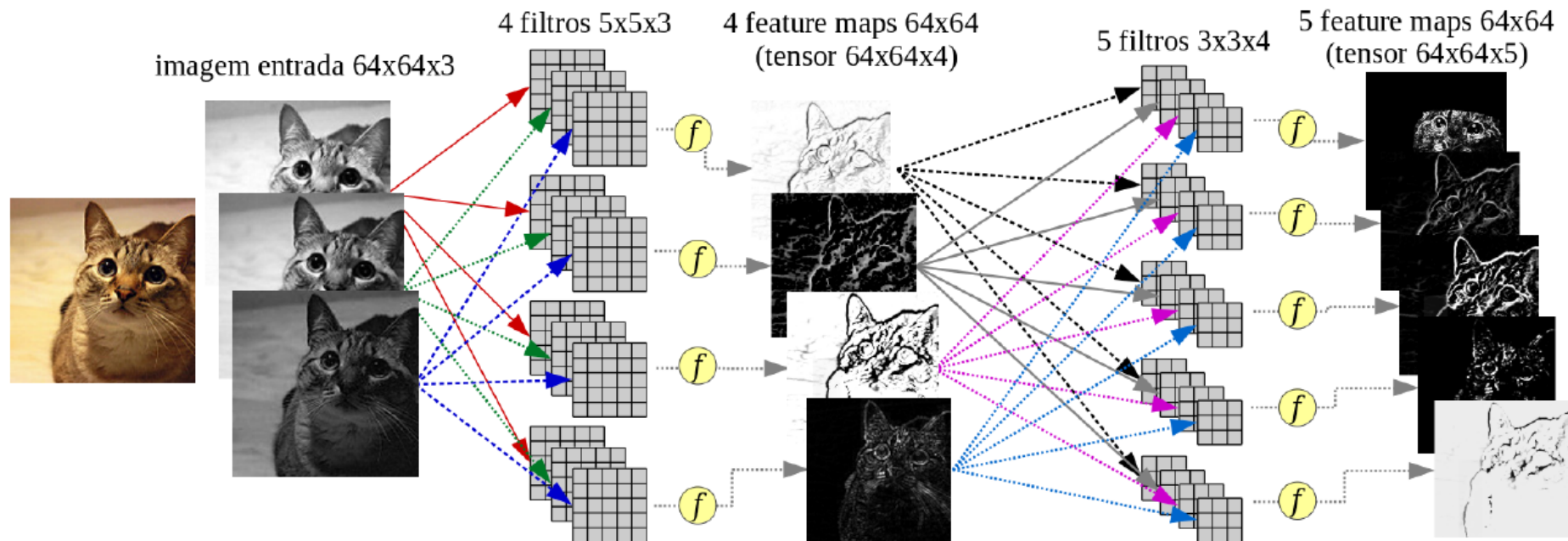
Perceptron de múltiplas camadas (MLP)

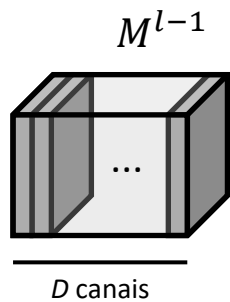


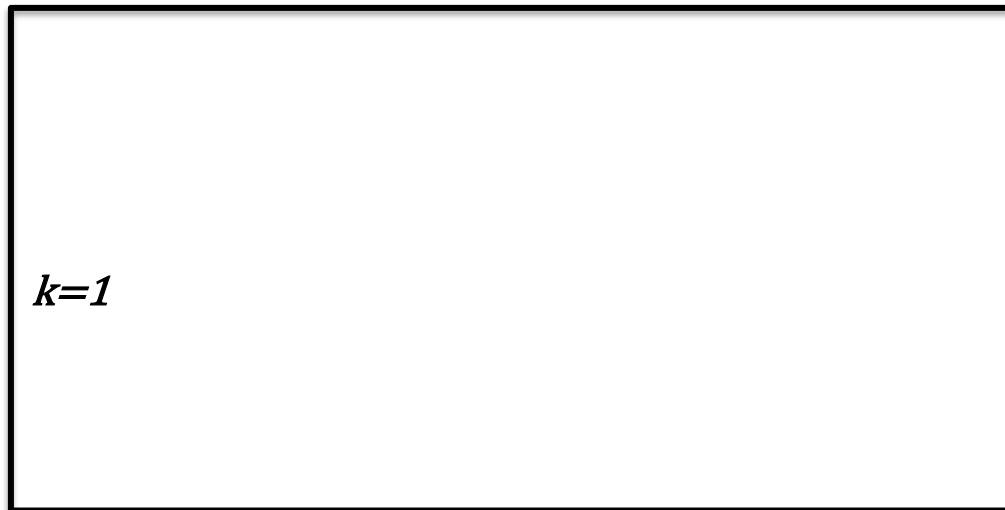
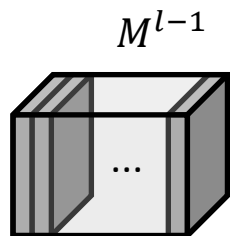
Redes Neurais Convolucionais (CNNs)



CAMADA CONVOLUCIONAL

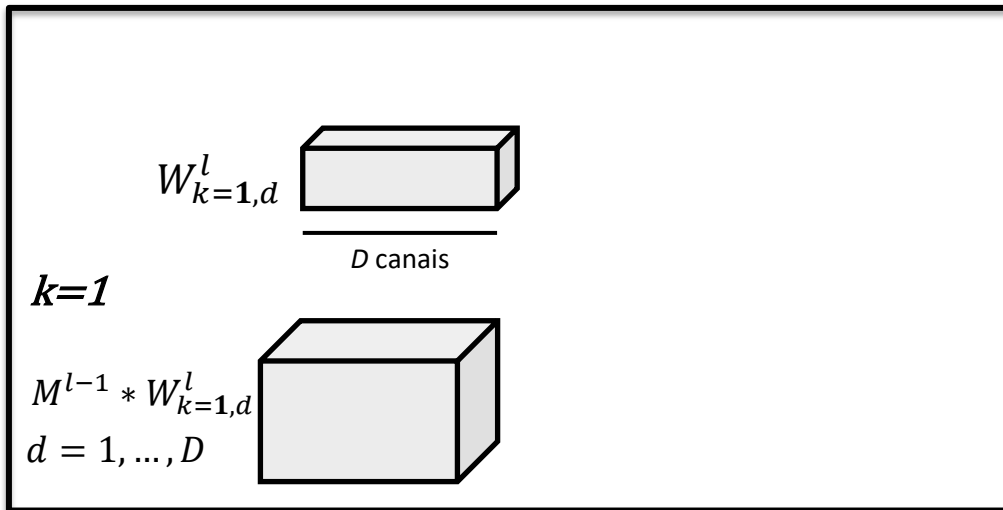
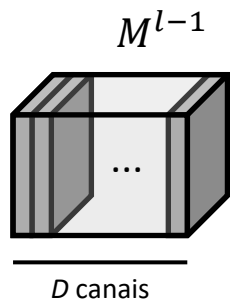




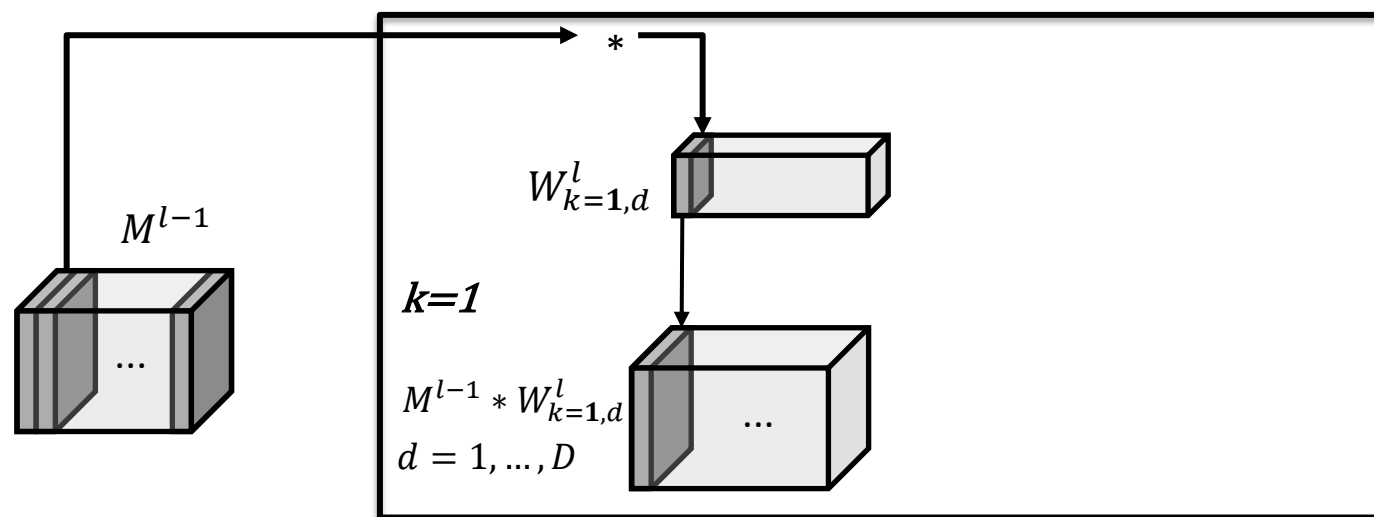


Camada convolucional C^l

Camada convolucional

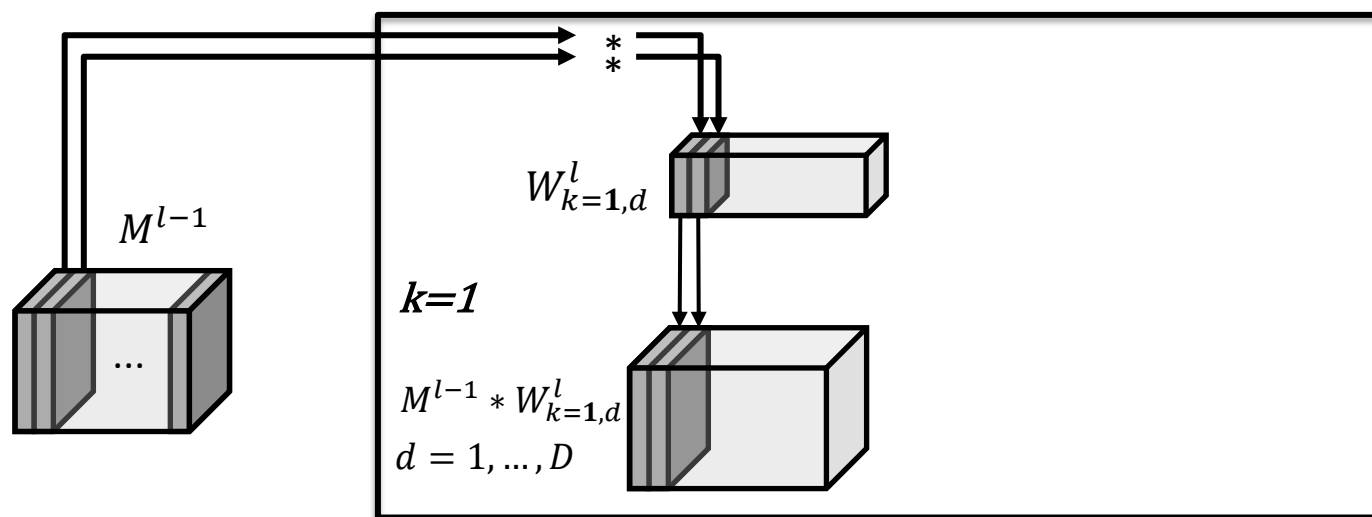


Camada convolucional C^l



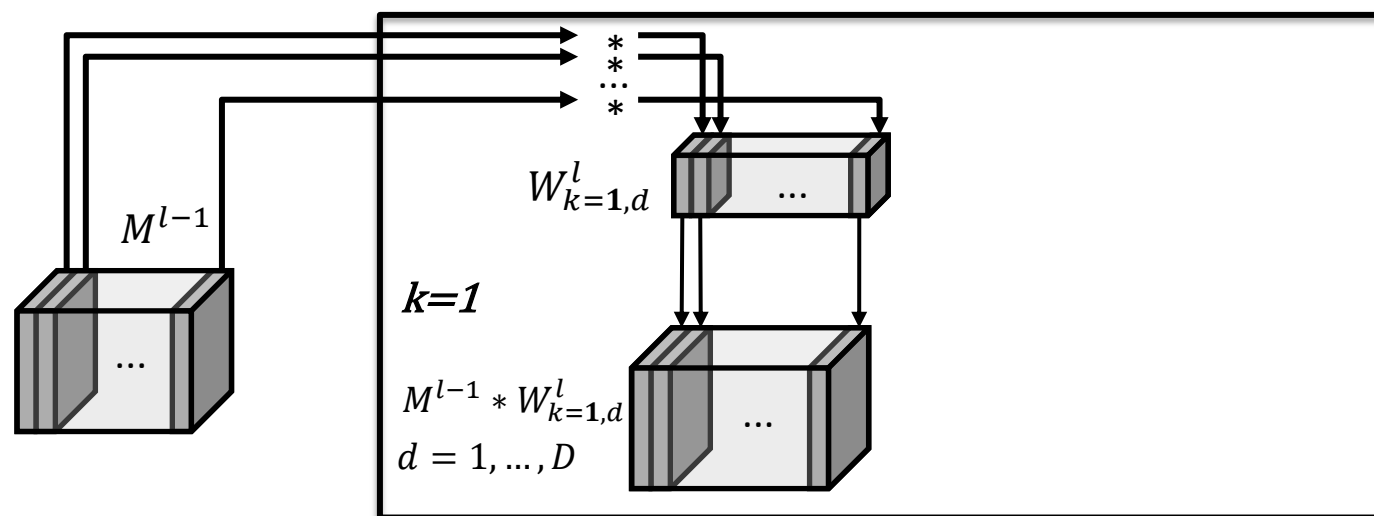
Camada convolucional C^l

Camada convolucional



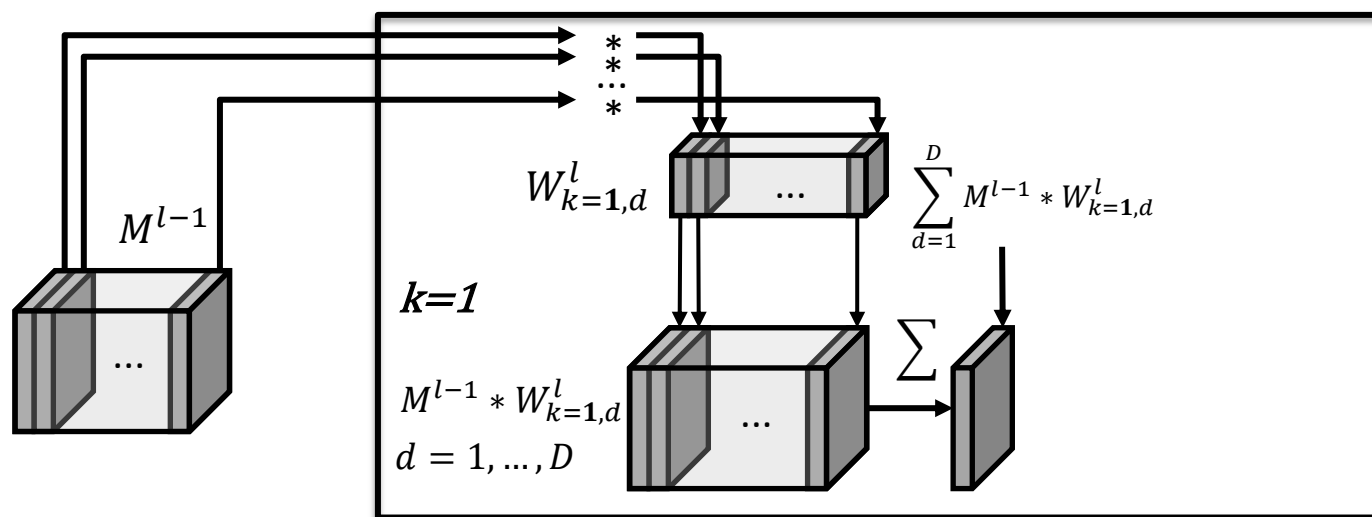
Camada convolucional C^l

Camada convolucional



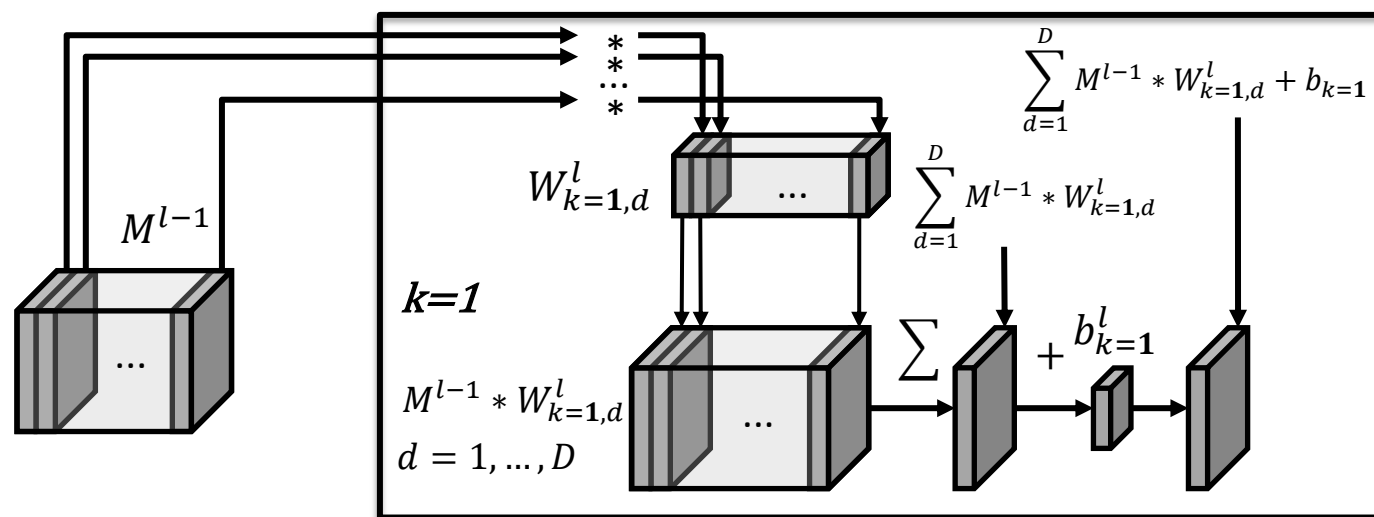
Camada convolucional C^l

Camada convolucional



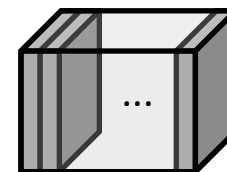
Camada convolucional C^l

Camada convolucional

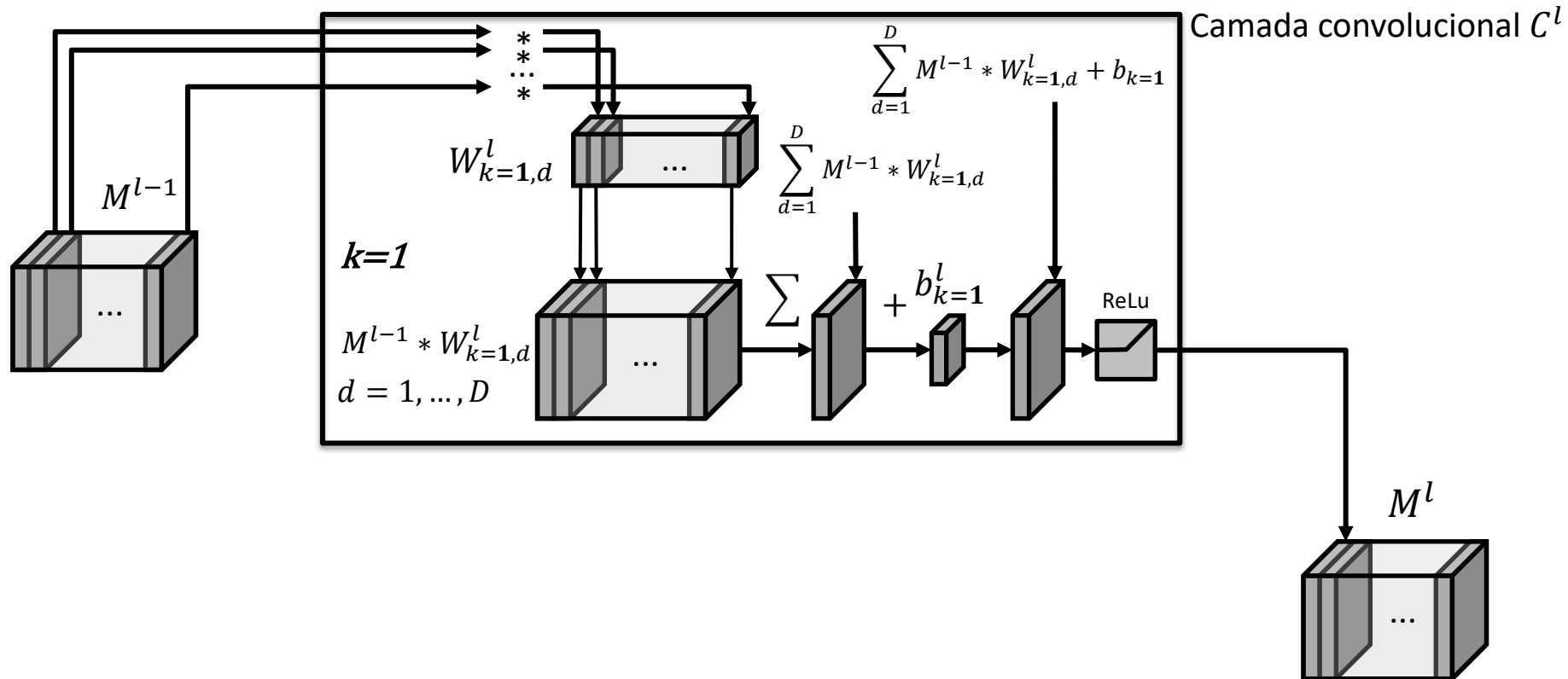


Camada convolucional C^l

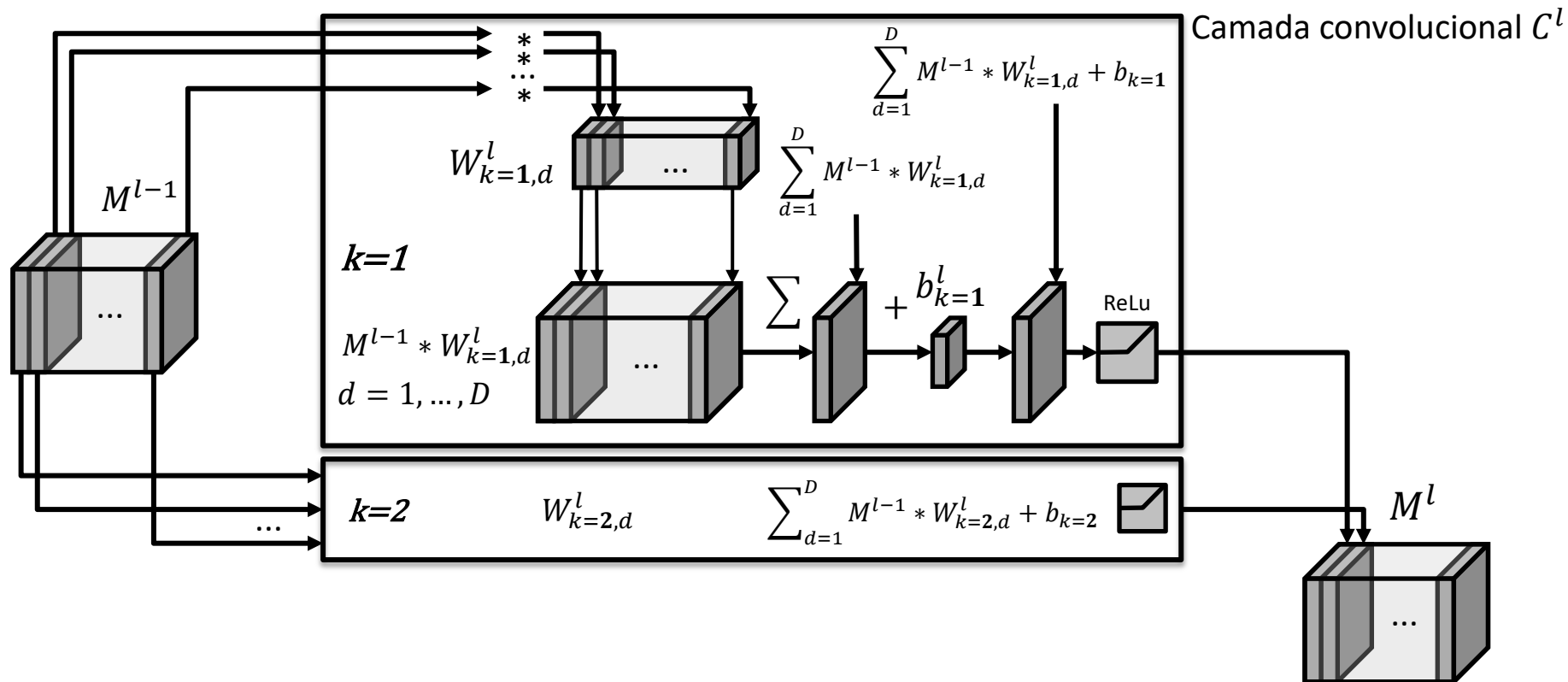
M^l



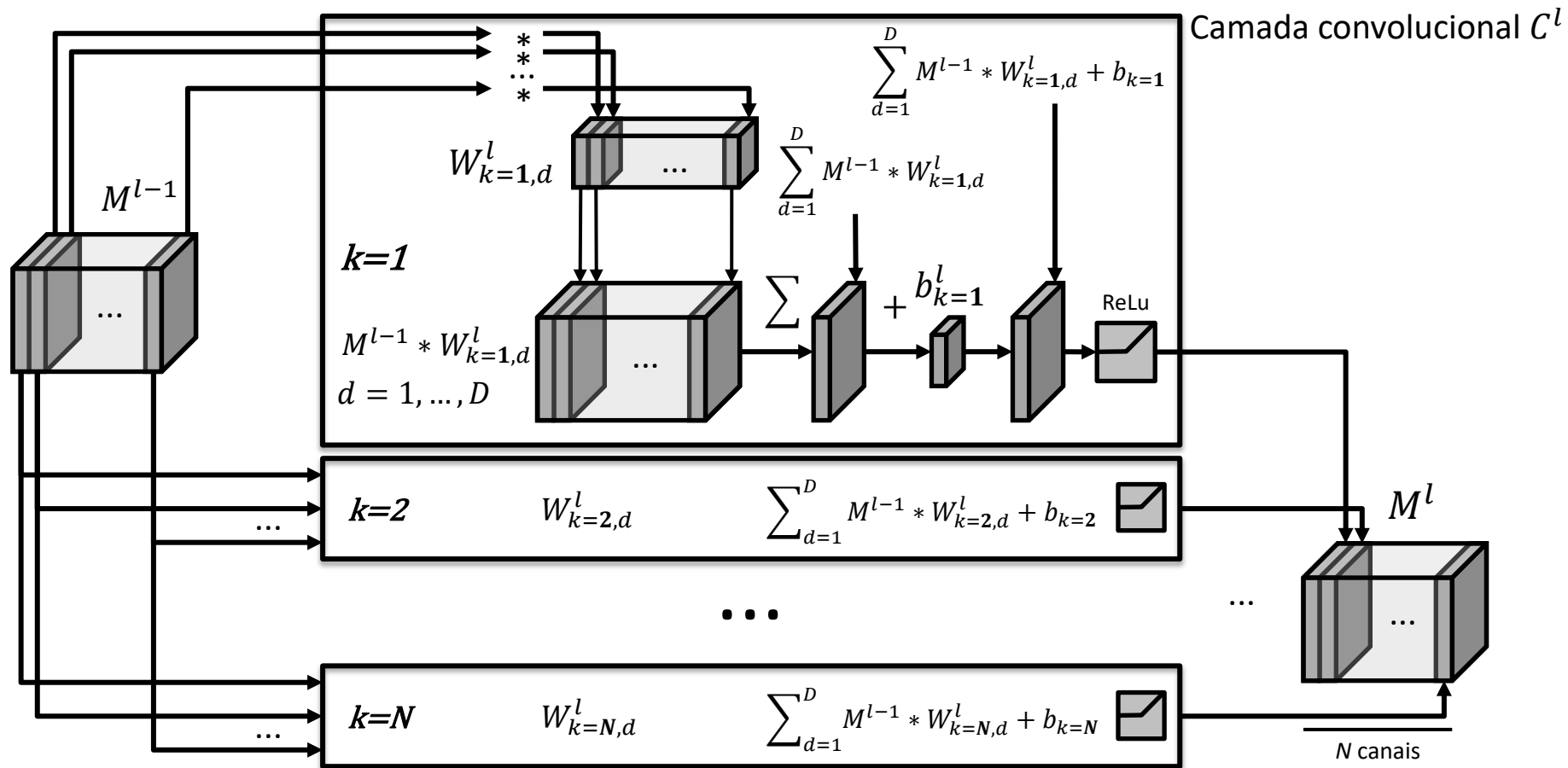
Camada convolucional



Camada convolucional



Camada convolucional



Camada convolucional

<https://cs231n.github.io/convolutional-networks/>

$x[:, :, 0]$ 5×5 + pad 1

0	0	0	0	0	0	0
0	6	3	4	6	2	0
0	7	4	4	6	1	0
0	2	6	2	2	7	0
0	4	3	7	7	2	0
0	5	4	1	7	3	0
0	0	0	0	0	0	0

$x[:, :, 1]$ 5×5 + pad 1

0	0	0	0	0	0	0
0	5	5	1	7	3	0
0	4	0	3	1	5	0
0	4	3	0	0	2	0
0	2	6	1	7	3	0
0	3	7	6	5	5	0
0	0	0	0	0	0	0

$x[:, :, 2]$ 5×5 + pad 1

0	0	0	0	0	0	0
0	6	5	2	3	6	0
0	3	7	0	2	4	0
0	2	6	4	0	6	0
0	1	3	0	3	5	0
0	1	1	0	1	4	0
0	0	0	0	0	0	0

$w0[:, :, 0]$

-1	1	1
1	1	2
0	-2	1

$w0[:, :, 1]$

-1	1	-1
-1	1	2
-1	-1	1

$w0[:, :, 2]$

-1	-1	1
1	-2	2
2	-1	2

$x[:, :, 0] * w0[:, :, 0]$

$x[:, :, 1] * w0[:, :, 1]$

$b0$

1

$x[:, :, 2] * w0[:, :, 2]$

$w1[:, :, 0]$

-1	-2	1
1	1	2
-2	2	2

$x[:, :, 0] * w1[:, :, 0]$

$x[:, :, 1] * w1[:, :, 1]$

$b1$

0

$x[:, :, 2] * w1[:, :, 2]$

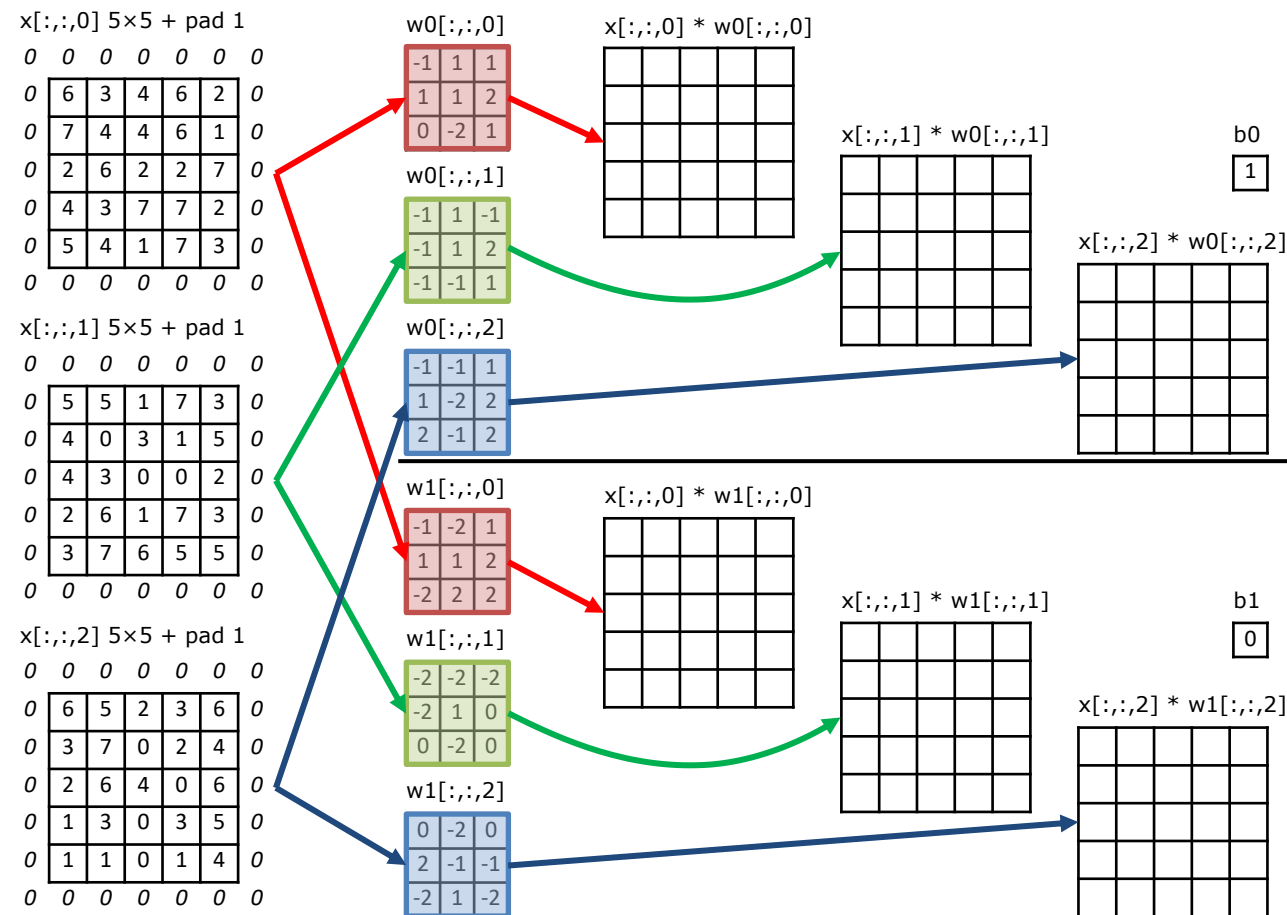
$w1[:, :, 1]$

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

$w1[:, :, 2]$

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

Camada convolucional



Camada convolucional

<https://cs231n.github.io/convolutional-networks/>

$x[:, :, 0]$ 5×5 + pad 1

0	0	0	0	0	0	0
0	6	3	4	6	2	0
0	7	4	4	6	1	0
0	2	6	2	2	7	0
0	4	3	7	7	2	0
0	5	4	1	7	3	0
0	0	0	0	0	0	0

$x[:, :, 1]$ 5×5 + pad 1

0	0	0	0	0	0	0
0	5	5	1	7	3	0
0	4	0	3	1	5	0
0	4	3	0	0	2	0
0	2	6	1	7	3	0
0	3	7	6	5	5	0
0	0	0	0	0	0	0

$x[:, :, 2]$ 5×5 + pad 1

0	0	0	0	0	0	0
0	6	5	2	3	6	0
0	3	7	0	2	4	0
0	2	6	4	0	6	0
0	1	3	0	3	5	0
0	1	1	0	1	4	0
0	0	0	0	0	0	0

$w0[:, :, 0]$

-1	1	1
1	1	2
0	-2	1

$w0[:, :, 1]$

-1	1	-1
-1	1	2
-1	-1	1

$w0[:, :, 2]$

-1	-1	1
1	-2	2
2	-1	2

$x[:, :, 0] * w0[:, :, 0]$

$x[:, :, 1] * w0[:, :, 1]$

$b0$

1

$x[:, :, 2] * w0[:, :, 2]$

$w1[:, :, 0]$

-1	-2	1
1	1	2
-2	2	2

$w1[:, :, 1]$

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

$w1[:, :, 2]$

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

$x[:, :, 0] * w1[:, :, 0]$

$x[:, :, 1] * w1[:, :, 1]$

$b1$

0

$x[:, :, 2] * w1[:, :, 2]$

$x[:, :, 0]$ 5×5 + pad 1

0	0	0	0	0	0	0
0	6	3	4	6	2	0
0	7	4	4	6	1	0
0	2	6	2	2	7	0
0	4	3	7	7	2	0
0	5	4	1	7	3	0
0	0	0	0	0	0	0

$x[:, :, 1]$ 5×5 + pad 1

0	0	0	0	0	0	0
0	5	5	1	7	3	0
0	4	0	3	1	5	0
0	4	3	0	0	2	0
0	2	6	1	7	3	0
0	3	7	6	5	5	0
0	0	0	0	0	0	0

$x[:, :, 2]$ 5×5 + pad 1

0	0	0	0	0	0	0
0	6	5	2	3	6	0
0	3	7	0	2	4	0
0	2	6	4	0	6	0
0	1	3	0	3	5	0
0	1	1	0	1	4	0
0	0	0	0	0	0	0

$w0[:, :, 0]$

-1	1	1
1	1	2
0	-2	1

$w0[:, :, 1]$

-1	1	-1
-1	1	2
-1	-1	1

$w0[:, :, 2]$

-1	-1	1
1	-2	2
2	-1	2

$w1[:, :, 0]$

-1	-2	1
1	1	2
-2	2	2

$w1[:, :, 1]$

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

$w1[:, :, 2]$

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

$x[:, :, 0] * w0[:, :, 0]$

12						

$x[:, :, 1] * w0[:, :, 1]$

4						

$b0$

1

$x[:, :, 2] * w0[:, :, 2]$

-17						

$x[:, :, 0] * w1[:, :, 0]$

-9						

$x[:, :, 1] * w1[:, :, 1]$

-13						

$b1$

0

$x[:, :, 2] * w1[:, :, 2]$

-2						

Camada convolucional

<https://cs231n.github.io/convolutional-networks/>

$x[:, :, 0]$ 5×5 + pad 1

0	0	0	0	0	0
0	6	3	4	6	2
0	7	4	4	6	1
0	2	6	2	2	7
0	4	3	7	7	2
0	5	4	1	7	3
0	0	0	0	0	0

$w0[:, :, 0]$

-1	1	1
1	1	2
0	-2	1

$x[:, :, 0] * w0[:, :, 0]$

12	26			

$w0[:, :, 1]$

-1	1	-1
-1	1	2
-1	-1	1

$x[:, :, 1] * w0[:, :, 1]$

4	7			

$b0$

1

$x[:, :, 2] * w0[:, :, 2]$

-17	0			

$x[:, :, 1]$ 5×5 + pad 1

0	0	0	0	0	0
0	5	5	1	7	3
0	4	0	3	1	5
0	4	3	0	0	2
0	2	6	1	7	3
0	3	7	6	5	5
0	0	0	0	0	0

$w0[:, :, 2]$

-1	-1	1
1	-2	2
2	-1	2

$w1[:, :, 0]$

-1	-2	1
1	1	2
-2	2	2

$x[:, :, 0] * w1[:, :, 0]$

-9	14			

$x[:, :, 1] * w1[:, :, 1]$

-13	-11			

$b1$

0

$x[:, :, 2] * w1[:, :, 2]$

-2	-21			

$x[:, :, 2]$ 5×5 + pad 1

0	0	0	0	0	0
0	6	5	2	3	6
0	3	7	0	2	4
0	2	6	4	0	6
0	1	3	0	3	5
0	1	1	0	1	4
0	0	0	0	0	0

$w1[:, :, 1]$

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

$w1[:, :, 2]$

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

Camada convolucional

$x[:, :, 0]$ 5×5 + pad 1

0	0	0	0	0	0
0	6	3	4	6	2
0	7	4	4	6	1
0	2	6	2	2	7
0	4	3	7	7	2
0	5	4	1	7	3
0	0	0	0	0	0

$w0[:, :, 0]$

-1	1	1
1	1	2
0	-2	1

$x[:, :, 0] * w0[:, :, 0]$

12	26	18		

$w0[:, :, 1]$

-1	1	-1
-1	1	2
-1	-1	1

$x[:, :, 1] * w0[:, :, 1]$

4	7	6		

$b0$

1

$x[:, :, 2] * w0[:, :, 2]$

-17	0	14		

$x[:, :, 1]$ 5×5 + pad 1

0	0	0	0	0	0
0	5	5	1	7	3
0	4	0	3	1	5
0	4	3	0	0	2
0	2	6	1	7	3
0	3	7	6	5	5
0	0	0	0	0	0

$w0[:, :, 2]$

-1	-1	1
1	-2	2
2	-1	2

$w1[:, :, 0]$

-1	-2	1
1	1	2
-2	2	2

$x[:, :, 0] * w1[:, :, 0]$

-9	14	6		

$x[:, :, 1] * w1[:, :, 1]$

-13	-11	-21		

$b1$

0

$x[:, :, 2] * w1[:, :, 2]$

-2	-21	-1		

$x[:, :, 2]$ 5×5 + pad 1

0	0	0	0	0	0
0	6	5	2	3	6
0	3	7	0	2	4
0	2	6	4	0	6
0	1	3	0	3	5
0	1	1	0	1	4
0	0	0	0	0	0

$w1[:, :, 1]$

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

$w1[:, :, 2]$

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

$x[:, :, 0]$ 5×5 + pad 1

0	0	0	0	0	0
0	6	3	4	6	2
0	7	4	4	6	1
0	2	6	2	2	7
0	4	3	7	7	2
0	5	4	1	7	3
0	0	0	0	0	0

$x[:, :, 1]$ 5×5 + pad 1

0	0	0	0	0	0
0	5	5	1	7	3
0	4	0	3	1	5
0	4	3	0	0	2
0	2	6	1	7	3
0	3	7	6	5	5
0	0	0	0	0	0

$x[:, :, 2]$ 5×5 + pad 1

0	0	0	0	0	0
0	6	5	2	3	6
0	3	7	0	2	4
0	2	6	4	0	6
0	1	3	0	3	5
0	1	1	0	1	4
0	0	0	0	0	0

$w0[:, :, 0]$

-1	1	1
1	1	2
0	-2	1

$w0[:, :, 1]$

-1	1	-1
-1	1	2
-1	-1	1

$w0[:, :, 2]$

-1	-1	1
1	-2	2
2	-1	2

$x[:, :, 0] * w0[:, :, 0]$

12	26	18	25	

$x[:, :, 1] * w0[:, :, 1]$

4	7	6	-1	

$b0$

1

$x[:, :, 2] * w0[:, :, 2]$

-17	0	14	-2	

$w1[:, :, 0]$

-1	-2	1
1	1	2
-2	2	2

$x[:, :, 0] * w1[:, :, 0]$

-9	14	6	7	

$x[:, :, 1] * w1[:, :, 1]$

-13	-11	-21	-17	

$b1$

0

$x[:, :, 2] * w1[:, :, 2]$

-2	-21	-1	3	

$x[:, :, 0]$ 5×5 + pad 1

0	0	0	0	0	0
0	6	3	4	6	2
0	7	4	4	6	1
0	2	6	2	2	7
0	4	3	7	7	2
0	5	4	1	7	3
0	0	0	0	0	0

$w0[:, :, 0]$

-1	1	1
1	1	2
0	-2	1

$x[:, :, 0] * w0[:, :, 0]$

12	26	18	25	21

$w0[:, :, 1]$

-1	1	-1
-1	1	2
-1	-1	1

$x[:, :, 1] * w0[:, :, 1]$

4	7	6	-1	12

$b0$

1

$x[:, :, 2] * w0[:, :, 2]$

-17	0	14	-2	-8

$x[:, :, 1]$ 5×5 + pad 1

0	0	0	0	0	0
0	5	5	1	7	3
0	4	0	3	1	5
0	4	3	0	0	2
0	2	6	1	7	3
0	3	7	6	5	5
0	0	0	0	0	0

$w0[:, :, 2]$

-1	-1	1
1	-2	2
2	-1	2

$w1[:, :, 0]$

-1	-2	1
1	1	2
-2	2	2

$x[:, :, 0] * w1[:, :, 0]$

-9	14	6	7	18

$x[:, :, 1] * w1[:, :, 1]$

-13	-11	-21	-17	-9

$b1$

0

$x[:, :, 2] * w1[:, :, 2]$

-2	-21	-1	3	-17

$x[:, :, 2]$ 5×5 + pad 1

0	0	0	0	0	0
0	6	5	2	3	6
0	3	7	0	2	4
0	2	6	4	0	6
0	1	3	0	3	5
0	1	1	0	1	4
0	0	0	0	0	0

$w1[:, :, 1]$

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

$w1[:, :, 2]$

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

$x[:, :, 0] \ 5 \times 5 + \text{pad } 1$

0	0	0	0	0	0	0
0	6	3	4	6	2	0
0	7	4	4	6	1	0
0	2	6	2	2	7	0
0	4	3	7	7	2	0
0	5	4	1	7	3	0
0	0	0	0	0	0	0

$x[:, :, 1] \ 5 \times 5 + \text{pad } 1$

0	0	0	0	0	0	0
0	5	5	1	7	3	0
0	4	0	3	1	5	0
0	4	3	0	0	2	0
0	2	6	1	7	3	0
0	3	7	6	5	5	0
0	0	0	0	0	0	0

$x[:, :, 2] \ 5 \times 5 + \text{pad } 1$

0	0	0	0	0	0	0
0	6	5	2	3	6	0
0	3	7	0	2	4	0
0	2	6	4	0	6	0
0	1	3	0	3	5	0
0	1	1	0	1	4	0
0	0	0	0	0	0	0

$w0[:, :, 0]$

-1	1	1
1	1	2
0	-2	1

$w0[:, :, 1]$

-1	1	-1
-1	1	2
-1	-1	1

$w0[:, :, 2]$

-1	-1	1
1	-2	2
2	-1	2

$x[:, :, 0] * w0[:, :, 0]$

12	26	18	25	21
-5				

$x[:, :, 1] * w0[:, :, 1]$

4	7	6	-1	12
-5				

$b0$

1

$x[:, :, 2] * w0[:, :, 2]$

-17	0	14	-2	-8
-3				

$w1[:, :, 0]$

-1	-2	1
1	1	2
-2	2	2

$w1[:, :, 1]$

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

$w1[:, :, 2]$

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

$x[:, :, 0] * w1[:, :, 0]$

-9	14	6	7	18
7				

$x[:, :, 1] * w1[:, :, 1]$

-13	-11	-21	-17	-9
-20				

$b1$

0

$x[:, :, 2] * w1[:, :, 2]$

-2	-21	-1	3	-17
3				

$x[:, :, 0]$ 5×5 + pad 1

0	0	0	0	0	0	0
0	6	3	4	6	2	0
0	7	4	4	6	1	0
0	2	6	2	2	7	0
0	4	3	7	7	2	0
0	5	4	1	7	3	0
0	0	0	0	0	0	0

$x[:, :, 1]$ 5×5 + pad 1

0	0	0	0	0	0	0
0	5	5	1	7	3	0
0	4	0	3	1	5	0
0	4	3	0	0	2	0
0	2	6	1	7	3	0
0	3	7	6	5	5	0
0	0	0	0	0	0	0

$x[:, :, 2]$ 5×5 + pad 1

0	0	0	0	0	0	0
0	6	5	2	3	6	0
0	3	7	0	2	4	0
0	2	6	4	0	6	0
0	1	3	0	3	5	0
0	1	1	0	1	4	0
0	0	0	0	0	0	0

$w0[:, :, 0]$

-1	1	1
1	1	2
0	-2	1

$w0[:, :, 1]$

-1	1	-1
-1	1	2
-1	-1	1

$w0[:, :, 2]$

-1	-1	1
1	-2	2
2	-1	2

$w1[:, :, 0]$

-1	-2	1
1	1	2
-2	2	2

$w1[:, :, 1]$

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

$w1[:, :, 2]$

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

$x[:, :, 0] * w0[:, :, 0]$

12	26	18	25	21
12				

$x[:, :, 1] * w0[:, :, 1]$

4	7	6	-1	12
4				

$b0$

1

$x[:, :, 2] * w0[:, :, 2]$

-17	0	14	-2	-8
-17				

$x[:, :, 0] * w1[:, :, 0]$

-9	14	6	7	18
7				

$x[:, :, 1] * w1[:, :, 1]$

-13	-11	-21	-17	-9
-20				

$b1$

0

$x[:, :, 2] * w1[:, :, 2]$

-2	-21	-1	3	-17
3				

Camada convolucional

<https://cs231n.github.io/convolutional-networks/>

$x[:, :, 0]$ 5×5 + pad 1

0	0	0	0	0	0	0
0	6	3	4	6	2	0
0	7	4	4	6	1	0
0	2	6	2	2	7	0
0	4	3	7	7	2	0
0	5	4	1	7	3	0
0	0	0	0	0	0	0

$w0[:, :, 0]$

-1	1	1
1	1	2
0	-2	1

$x[:, :, 0] * w0[:, :, 0]$

12	26	18	25	21
-5	28	19	4	24
-5	11	15	17	24
4	16	20	26	14
1	16	5	5	20

$x[:, :, 1] * w0[:, :, 1]$

4	7	6	-1	12
-5	3	-4	-9	13
-7	15	-10	-2	-6
-15	8	3	-2	15
-12	2	13	3	19

$b0$

1

$x[:, :, 2] * w0[:, :, 2]$

-17	0	14	-2	-8
-3	-5	32	11	-10
9	-7	22	12	-14
9	2	17	14	-13
4	-1	15	9	-5

$x[:, :, 1]$ 5×5 + pad 1

0	0	0	0	0	0	0
0	5	5	1	7	3	0
0	4	0	3	1	5	0
0	4	3	0	0	2	0
0	2	6	1	7	3	0
0	3	7	6	5	5	0
0	0	0	0	0	0	0

$w0[:, :, 1]$

-1	1	-1
-1	1	2
-1	-1	1

$w1[:, :, 0]$

-1	-2	1
1	1	2
-2	2	2

$x[:, :, 0] * w1[:, :, 0]$

-9	14	6	7	18
7	20	20	22	17
3	17	2	22	28
-15	26	27	1	35
11	15	22	36	35

$x[:, :, 1] * w1[:, :, 1]$

-13	-11	-21	-17	-9
-20	-30	-7	-27	-5
-26	-15	-34	-28	-28
-38	-34	-49	-31	-21
-15	-17	-6	-19	-1

$b1$

0

$x[:, :, 2] * w1[:, :, 2]$

-2	-21	-1	3	-17
3	-33	-25	-7	-18
-3	-5	-28	-4	-16
-7	-12	-5	-15	-10
-4	-1	-11	0	-6

$x[:, :, 2]$ 5×5 + pad 1

0	0	0	0	0	0	0
0	6	5	2	3	6	0
0	3	7	0	2	4	0
0	2	6	4	0	6	0
0	1	3	0	3	5	0
0	1	1	0	1	4	0
0	0	0	0	0	0	0

$w1[:, :, 1]$

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

$w1[:, :, 2]$

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

Camada convolucional

<https://cs231n.github.io/convolutional-networks/>

$x[:, :, 0]$ 5×5 + pad 1

0	0	0	0	0	0
0	6	3	4	6	2
0	7	4	4	6	1
0	2	6	2	2	7
0	4	3	7	7	2
0	5	4	1	7	3
0	0	0	0	0	0

$w0[:, :, 0]$

-1	1	1
1	1	2
0	-2	1

$x[:, :, 0] * w0[:, :, 0]$

12	26	18	25	21
-5	28	19	4	24
-5	11	15	17	24
4	16	20	26	14
1	16	5	5	20

$w0[:, :, 1]$

-1	1	-1
-1	1	2
-1	-1	1

$x[:, :, 1] * w0[:, :, 1]$

4	7	6	-1	12
-5	3	-4	-9	13
-7	15	-10	-2	-6
-15	8	3	-2	15
-12	2	13	3	19

$b0$

1

$x[:, :, 2] * w0[:, :, 2]$

-17	0	14	-2	-8
-3	-5	32	11	-10
9	-7	22	12	-14
9	2	17	14	-13
4	-1	15	9	-5

$x[:, :, 1]$ 5×5 + pad 1

0	0	0	0	0	0
0	5	5	1	7	3
0	4	0	3	1	5
0	4	3	0	0	2
0	2	6	1	7	3
0	3	7	6	5	5
0	0	0	0	0	0

$w0[:, :, 2]$

-1	-1	1
1	-2	2
2	-1	2

$w1[:, :, 0]$

-1	-2	1
1	1	2
-2	2	2

$x[:, :, 0] * w1[:, :, 0]$

-9	14	6	7	18
7	20	20	22	17
3	17	2	22	28
-15	26	27	1	35
11	15	22	36	35

$x[:, :, 1] * w1[:, :, 1]$

-13	-11	-21	-17	-9
-20	-30	-7	-27	-5
-26	-15	-34	-28	-28
-38	-34	-49	-31	-21
-15	-17	-6	-19	-1

$b1$

0

$x[:, :, 2] * w1[:, :, 2]$

-2	-21	-1	3	-17
3	-33	-25	-7	-18
-3	-5	-28	-4	-16
-7	-12	-5	-15	-10
-4	-1	-11	0	-6

$x[:, :, 2]$ 5×5 + pad 1

0	0	0	0	0	0
0	6	5	2	3	6
0	3	7	0	2	4
0	2	6	4	0	6
0	1	3	0	3	5
0	1	1	0	1	4
0	0	0	0	0	0

$w1[:, :, 1]$

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

$w1[:, :, 2]$

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

Camada convolucional

<https://cs231n.github.io/convolutional-networks/>

$x[:, :, 0]$ 5×5 + pad 1

0	0	0	0	0	0	0
0	6	3	4	6	2	0
0	7	4	4	6	1	0
0	2	6	2	2	7	0
0	4	3	7	7	2	0
0	5	4	1	7	3	0
0	0	0	0	0	0	0

$x[:, :, 1]$ 5×5 + pad 1

0	0	0	0	0	0	0
0	5	5	1	7	3	0
0	4	0	3	1	5	0
0	4	3	0	0	2	0
0	2	6	1	7	3	0
0	3	7	6	5	5	0
0	0	0	0	0	0	0

$x[:, :, 2]$ 5×5 + pad 1

0	0	0	0	0	0	0
0	6	5	2	3	6	0
0	3	7	0	2	4	0
0	2	6	4	0	6	0
0	1	3	0	3	5	0
0	1	1	0	1	4	0
0	0	0	0	0	0	0

$w0[:, :, 0]$

-1	1	1
1	1	2
0	-2	1

$w0[:, :, 1]$

-1	1	-1
-1	1	2
-1	-1	1

$w0[:, :, 2]$

-1	-1	1
1	-2	2
2	-1	2

$x[:, :, 0] * w0[:, :, 0]$

12	26	18	25	21
-5	28	19	4	24
-5	11	15	17	24
4	16	20	26	14
1	16	5	5	20

$x[:, :, 1] * w0[:, :, 1]$

4	7	6	-1	12
-5	3	-4	-9	13
-7	15	-10	-2	-6
-15	8	3	-2	15
-12	2	13	3	19

$b0$

1

$x[:, :, 2] * w0[:, :, 2]$

-17	0	14	-2	-8
-3	-5	32	11	-10
9	-7	22	12	-14
9	2	17	14	-13
4	-1	15	9	-5

Σ

$v[:, :, 0]$

$w1[:, :, 0]$

-1	-2	1
1	1	2
-2	2	2

$w1[:, :, 1]$

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

$w1[:, :, 2]$

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

$x[:, :, 0] * w1[:, :, 0]$

-9	14	6	7	18
7	20	20	22	17
3	17	2	22	28
-15	26	27	1	35
11	15	22	36	35

$x[:, :, 1] * w1[:, :, 1]$

-13	-11	-21	-17	-9
-20	-30	-7	-27	-5
-26	-15	-34	-28	-28
-38	-34	-49	-31	-21
-15	-17	-6	-19	-1

$b1$

0

$x[:, :, 2] * w1[:, :, 2]$

-2	-21	-1	3	-17
3	-33	-25	-7	-18
-3	-5	-28	-4	-16
-7	-12	-5	-15	-10
-4	-1	-11	0	-6

Σ

$v[:, :, 1]$

Camada convolucional

<https://cs231n.github.io/convolutional-networks/>

$x[:, :, 0]$ 5×5 + pad 1

0	0	0	0	0	0	0
0	6	3	4	6	2	0
0	7	4	4	6	1	0
0	2	6	2	2	7	0
0	4	3	7	7	2	0
0	5	4	1	7	3	0
0	0	0	0	0	0	0

$x[:, :, 1]$ 5×5 + pad 1

0	0	0	0	0	0	0
0	5	5	1	7	3	0
0	4	0	3	1	5	0
0	4	3	0	0	2	0
0	2	6	1	7	3	0
0	3	7	6	5	5	0
0	0	0	0	0	0	0

$x[:, :, 2]$ 5×5 + pad 1

0	0	0	0	0	0	0
0	6	5	2	3	6	0
0	3	7	0	2	4	0
0	2	6	4	0	6	0
0	1	3	0	3	5	0
0	1	1	0	1	4	0
0	0	0	0	0	0	0

$w0[:, :, 0]$

-1	1	1
1	1	2
0	-2	1

$w0[:, :, 1]$

-1	1	-1
-1	1	2
-1	-1	1

$w0[:, :, 2]$

-1	-1	1
1	-2	2
2	-1	2

$x[:, :, 0] * w0[:, :, 0]$

12	26	18	25	21
-5	28	19	4	24
-5	11	15	17	24
4	16	20	26	14
1	16	5	5	20

$x[:, :, 1] * w0[:, :, 1]$

4	7	6	-1	12
-5	3	-4	-9	13
-7	15	-10	-2	-6
-15	8	3	-2	15
-12	2	13	3	19

$x[:, :, 2] * w0[:, :, 2]$

-17	0	14	-2	-8
-3	-5	32	11	-10
9	-7	22	12	-14
9	2	17	14	-13
4	-1	15	9	-5

$b0$

1

Σ

$v[:, :, 0]$

0	34	39	23	35
-12	27	48	7	28
-2	20	28	48	5
-1	27	41	39	17
-6	15	34	18	35

$w1[:, :, 0]$

-1	-2	1
1	1	2
-2	2	2

$w1[:, :, 1]$

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

$w1[:, :, 2]$

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

$x[:, :, 0] * w1[:, :, 0]$

-9	14	6	7	18
7	20	20	22	17
3	17	2	22	28
-15	26	27	1	35
11	15	22	36	35

$x[:, :, 1] * w1[:, :, 1]$

-13	-11	-21	-17	-9
-20	-30	-7	-27	-5
-26	-15	-34	-28	-28
-38	-34	-49	-31	-21
-15	-17	-6	-19	-1

$x[:, :, 2] * w1[:, :, 2]$

-2	-21	-1	3	-17
3	-33	-25	-7	-18
-3	-5	-28	-4	-16
-7	-12	-5	-15	-10
-4	-1	-11	0	-6

$b1$

0

Σ

$v[:, :, 1]$

-24	-18	-16	-7	-8
-10	-43	-12	-12	-6
-26	-3	-60	-10	-16
-60	-20	-27	-45	4
-8	-3	5	17	28

Camada convolucional

<https://cs231n.github.io/convolutional-networks/>

$x[:, :, 0]$ 5×5 + pad 1

0	0	0	0	0	0	0
0	6	3	4	6	2	0
0	7	4	4	6	1	0
0	2	6	2	2	7	0
0	4	3	7	7	2	0
0	5	4	1	7	3	0
0	0	0	0	0	0	0

$x[:, :, 1]$ 5×5 + pad 1

0	0	0	0	0	0	0
0	5	5	1	7	3	0
0	4	0	3	1	5	0
0	4	3	0	0	2	0
0	2	6	1	7	3	0
0	3	7	6	5	5	0
0	0	0	0	0	0	0

$x[:, :, 2]$ 5×5 + pad 1

0	0	0	0	0	0	0
0	6	5	2	3	6	0
0	3	7	0	2	4	0
0	2	6	4	0	6	0
0	1	3	0	3	5	0
0	1	1	0	1	4	0
0	0	0	0	0	0	0

$w0[:, :, 0]$

-1	1	1
1	1	2
0	-2	1

$w0[:, :, 1]$

-1	1	-1
-1	1	2
-1	-1	1

$w0[:, :, 2]$

-1	-1	1
1	-2	2
2	-1	2

$x[:, :, 0] * w0[:, :, 0]$

12	26	18	25	21
-5	28	19	4	24
-5	11	15	17	24
4	16	20	26	14
1	16	5	5	20

$x[:, :, 1] * w0[:, :, 1]$

4	7	6	-1	12
-5	3	-4	-9	13
-7	15	-10	-2	-6
-15	8	3	-2	15
-12	2	13	3	19

$x[:, :, 2] * w0[:, :, 2]$

-17	0	14	-2	-8
-3	-5	32	11	-10
9	-7	22	12	-14
9	2	17	14	-13
4	-1	15	9	-5

$b0$

1

Σ

$v[:, :, 0]$

0	34	39	23	35
-12	27	48	7	28
-2	20	28	48	5
-1	27	41	39	17
-6	15	34	18	35

$y[:, :, 0]$



$w1[:, :, 0]$

-1	-2	1
1	1	2
-2	2	2

$w1[:, :, 1]$

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

$w1[:, :, 2]$

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

$x[:, :, 0] * w1[:, :, 0]$

-9	14	6	7	18
7	20	20	22	17
3	17	2	22	28
-15	26	27	1	35
11	15	22	36	35

$x[:, :, 1] * w1[:, :, 1]$

-13	-11	-21	-17	-9
-20	-30	-7	-27	-5
-26	-15	-34	-28	-28
-38	-34	-49	-31	-21
-15	-17	-6	-19	-1

$x[:, :, 2] * w1[:, :, 2]$

-2	-21	-1	3	-17
3	-33	-25	-7	-18
-3	-5	-28	-4	-16
-7	-12	-5	-15	-10
-4	-1	-11	0	-6

$b1$

0

Σ

$v[:, :, 1]$

-24	-18	-16	-7	-8
-10	-43	-12	-12	-6
-26	-3	-60	-10	-16
-60	-20	-27	-45	4
-8	-3	5	17	28

$y[:, :, 1]$



Camada convolucional

<https://cs231n.github.io/convolutional-networks/>

$x[:, :, 0]$ 5×5 + pad 1

0	0	0	0	0	0	0
0	6	3	4	6	2	0
0	7	4	4	6	1	0
0	2	6	2	2	7	0
0	4	3	7	7	2	0
0	5	4	1	7	3	0
0	0	0	0	0	0	0

$x[:, :, 1]$ 5×5 + pad 1

0	0	0	0	0	0	0
0	5	5	1	7	3	0
0	4	0	3	1	5	0
0	4	3	0	0	2	0
0	2	6	1	7	3	0
0	3	7	6	5	5	0
0	0	0	0	0	0	0

$x[:, :, 2]$ 5×5 + pad 1

0	0	0	0	0	0	0
0	6	5	2	3	6	0
0	3	7	0	2	4	0
0	2	6	4	0	6	0
0	1	3	0	3	5	0
0	1	1	0	1	4	0
0	0	0	0	0	0	0

$w0[:, :, 0]$

-1	1	1
1	1	2
0	-2	1

$w0[:, :, 1]$

-1	1	-1
-1	1	2
-1	-1	1

$w0[:, :, 2]$

-1	-1	1
1	-2	2
2	-1	2

$x[:, :, 0] * w0[:, :, 0]$

12	26	18	25	21
-5	28	19	4	24
-5	11	15	17	24
4	16	20	26	14
1	16	5	5	20

$x[:, :, 1] * w0[:, :, 1]$

4	7	6	-1	12
-5	3	-4	-9	13
-7	15	-10	-2	-6
-15	8	3	-2	15
-12	2	13	3	19

$x[:, :, 2] * w0[:, :, 2]$

-17	0	14	-2	-8
-3	-5	32	11	-10
9	-7	22	12	-14
9	2	17	14	-13
4	-1	15	9	-5

$b0$

1

Σ

$v[:, :, 0]$

0	34	39	23	35
-12	27	48	7	28
-2	20	28	48	5
-1	27	41	39	17
-6	15	34	18	35

$y[:, :, 0]$

0	34	39	23	35
0	27	48	7	28
0	20	28	48	5
0	27	41	39	17
0	15	34	18	35



ReLU



$w1[:, :, 0]$

-1	-2	1
1	1	2
-2	2	2

$w1[:, :, 1]$

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

$w1[:, :, 2]$

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

$x[:, :, 0] * w1[:, :, 0]$

-9	14	6	7	18
7	20	20	22	17
3	17	2	22	28
-15	26	27	1	35
11	15	22	36	35

$x[:, :, 1] * w1[:, :, 1]$

-13	-11	-21	-17	-9
-20	-30	-7	-27	-5
-26	-15	-34	-28	-28
-38	-34	-49	-31	-21
-15	-17	-6	-19	-1

$x[:, :, 2] * w1[:, :, 2]$

-2	-21	-1	3	-17
3	-33	-25	-7	-18
-3	-5	-28	-4	-16
-7	-12	-5	-15	-10
-4	-1	-11	0	-6

$b1$

0

Σ

$v[:, :, 1]$

-24	-18	-16	-7	-8
-10	-43	-12	-12	-6
-26	-3	-60	-10	-16
-60	-20	-27	-45	4
-8	-3	5	17	28

$y[:, :, 1]$

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	4
0	0	5	17	28



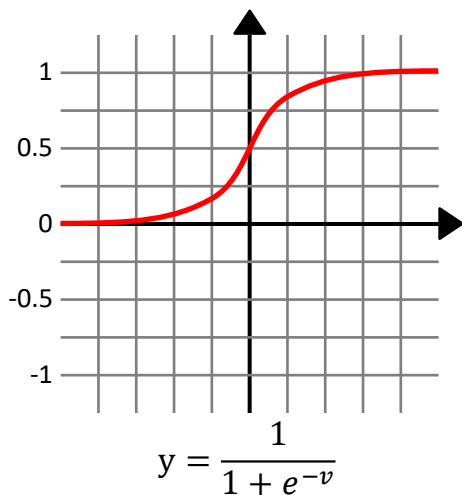
ReLU



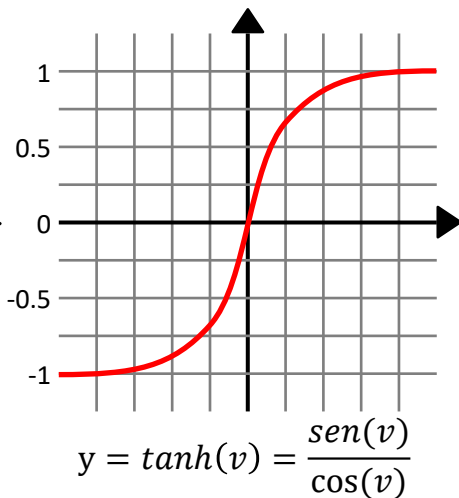
FUNÇÃO DE ATIVAÇÃO

Função de ativação

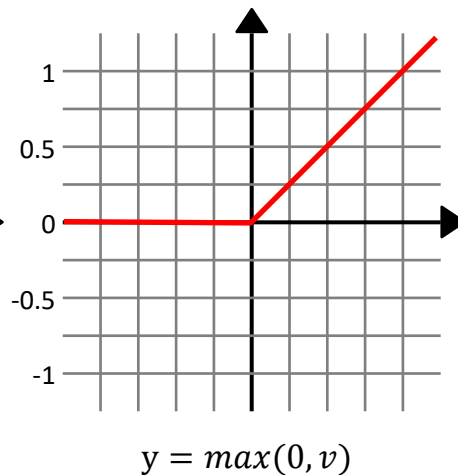
Logística



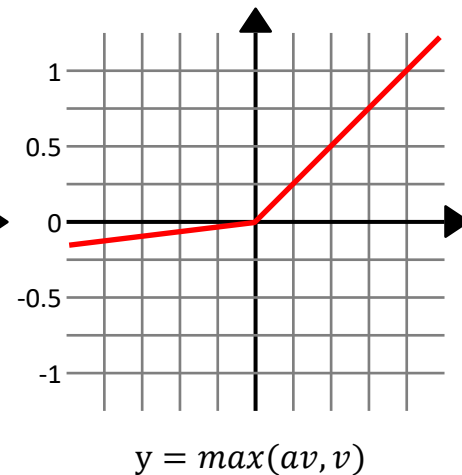
Tangente hiperbólica



ReLu

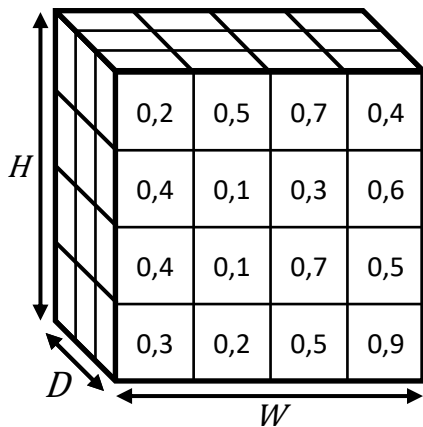


PReLU

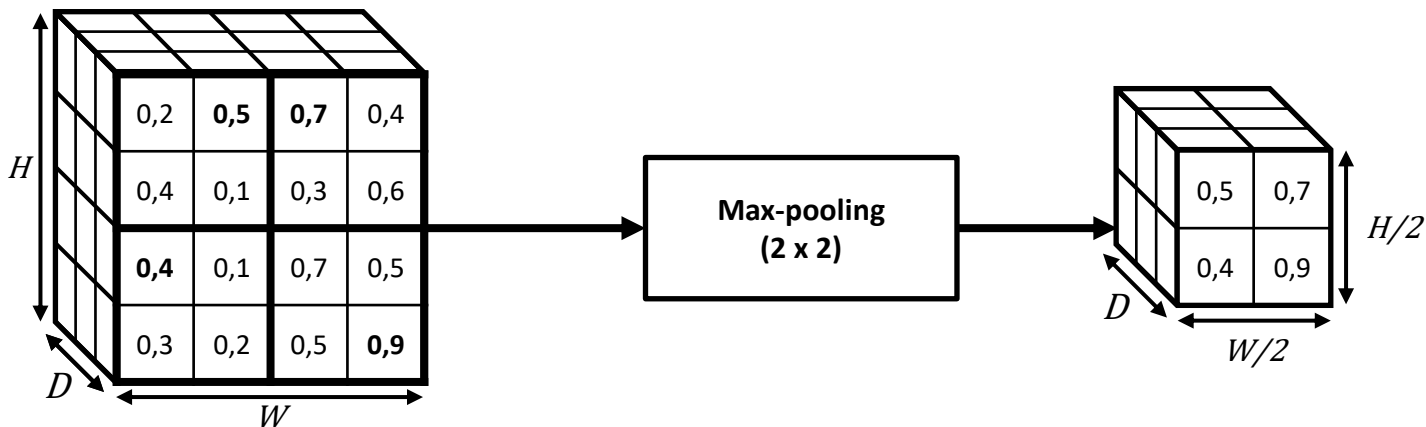


- Se $a=0,01 \rightarrow$ Leak ReLu

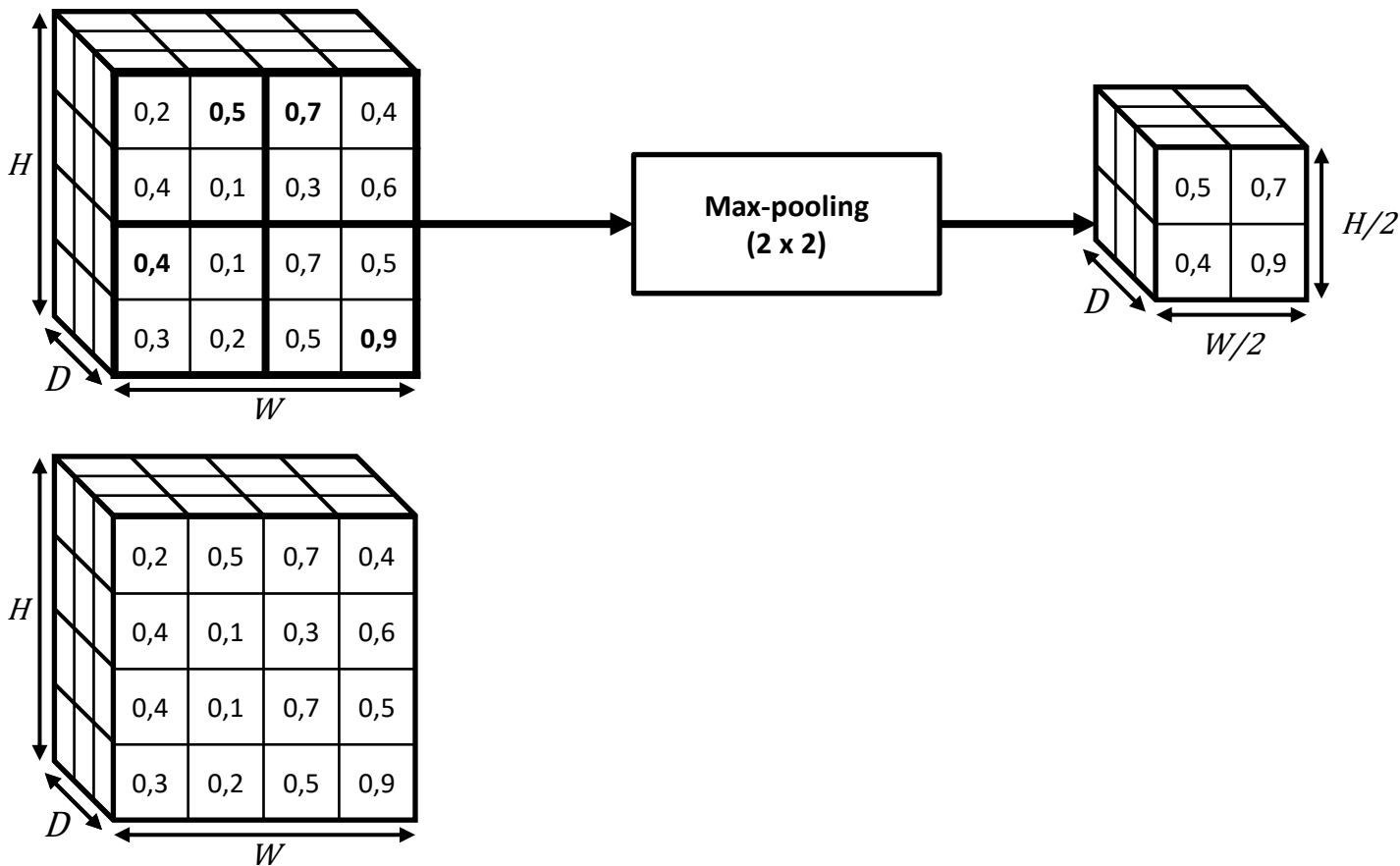
CAMADA DE POOLING



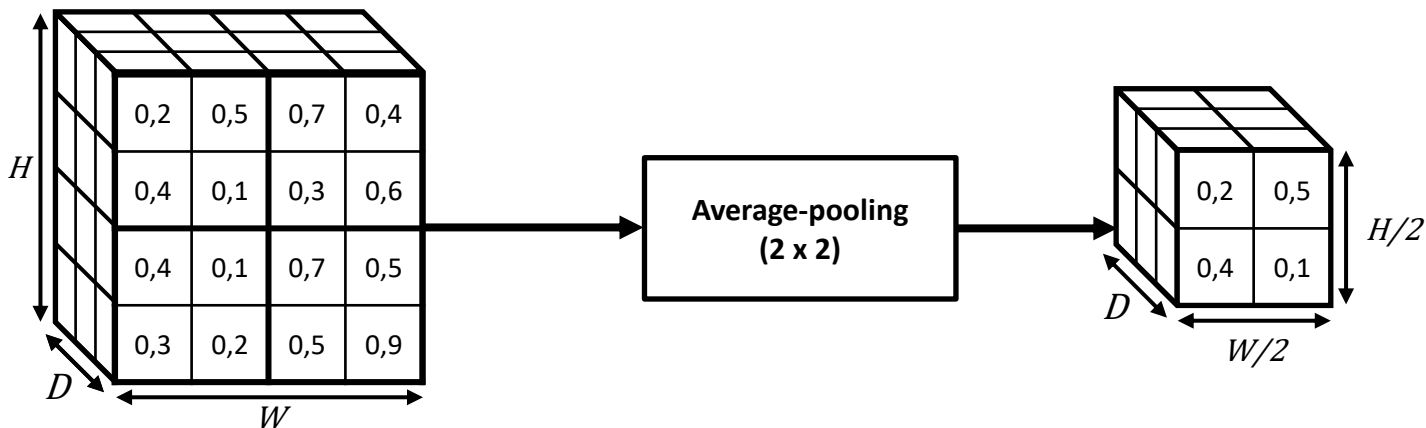
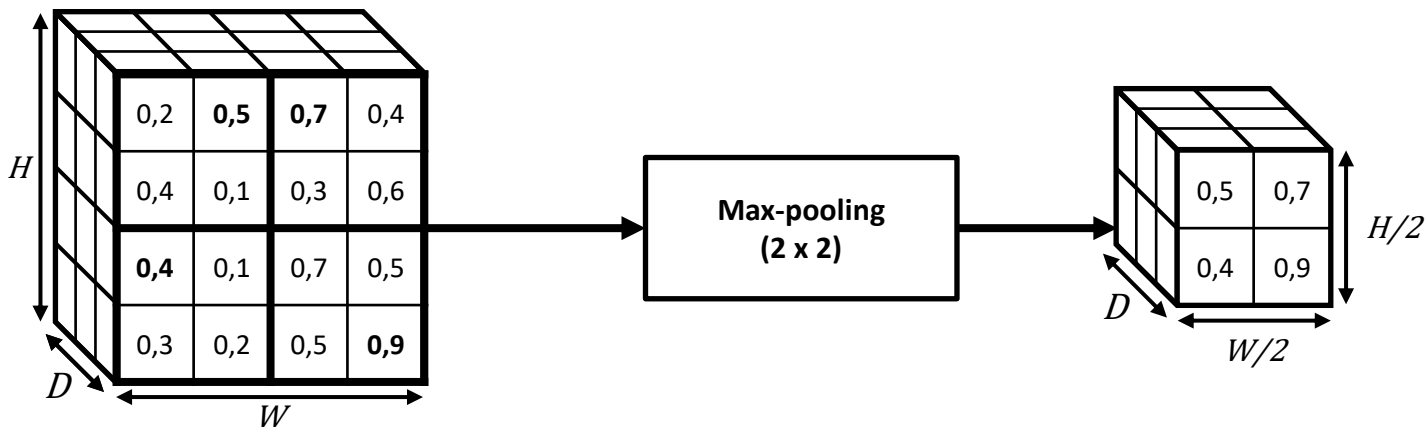
Camada de pooling



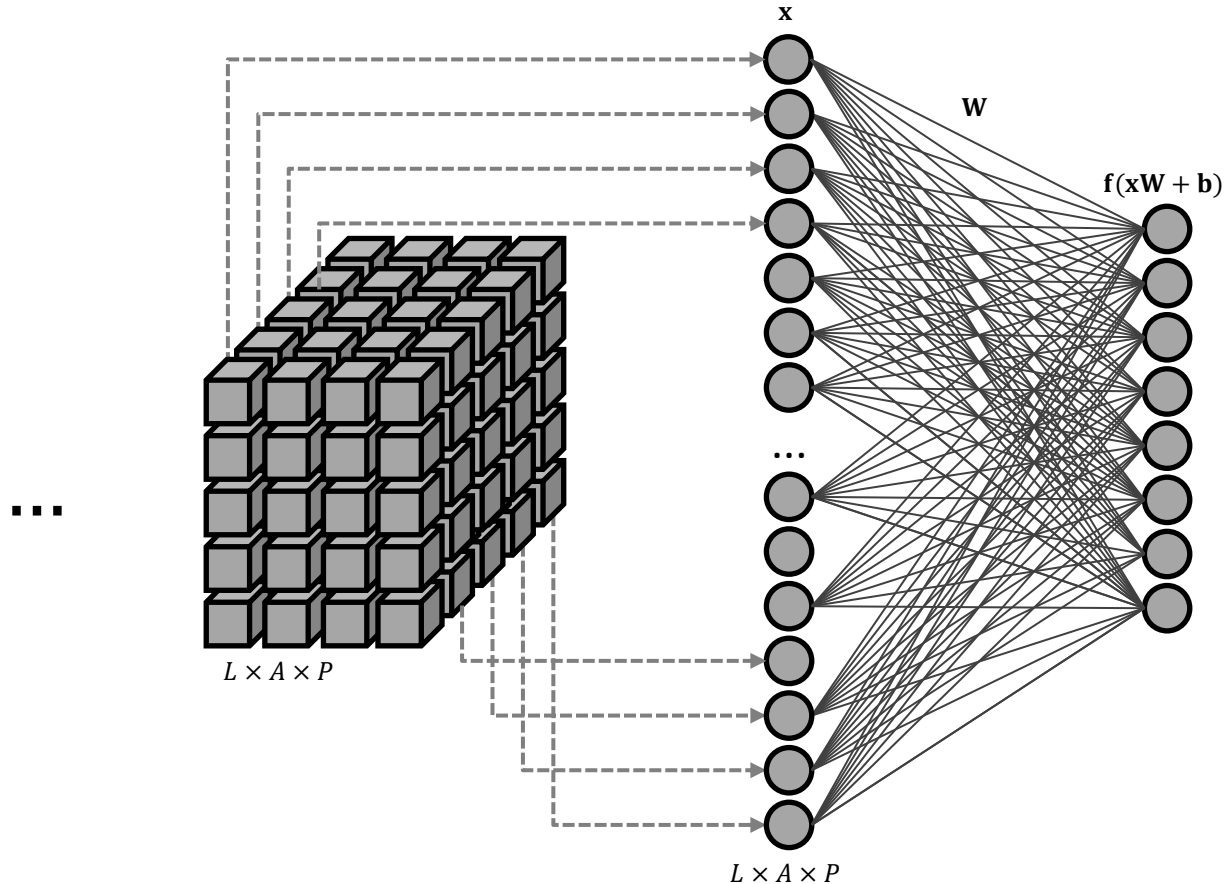
Camada de pooling



Camada de pooling



CAMADA COMPLETAMENTE CONECTADA



CAMADA DE SAÍDA - SOFTMAX

Camada de saída – softmax

- Função softmax para M classes:

$$- \text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=0}^{M-1} e^{x_j}}$$

- **Exemplo:**

- $\mathbf{x} = [-0,8 \quad 2,0 \quad 6,0 \quad -2,7 \quad 0,8]$

- $\sum_{j=0}^{M-1} x_j = 5,3$

- *Soma != de 1,0. Não pode ser interpretado como probabilidades.*

- $\sum_{j=0}^{M-1} e^{x_j} = 0,4493 + 7,3891 + 403,4288 + 0,0672 + 2,2255 = 413,5599$

- $\text{softmax}(x_i) = [0,0011 \quad 0,0179 \quad 0,9755 \quad 0,0002 \quad 0,0054]$

- $\sum_{j=0}^{M-1} \text{softmax}(x_i) = 1,0$

- *Representa a probabilidade da amostra pertencer a cada classe.*

FUNÇÃO DE PERDA (LOSS)

Função de perda (loss)

- Entropia cruzada para mais de 2 classes ($M > 2$):
 - $L(\mathbf{y}, \hat{\mathbf{y}}) = - \sum_{j=0}^{M-1} \mathbf{y}_j \cdot \log(\hat{\mathbf{y}}_j)$
- Entropia cruzada para 2 classes ($M=2$):
 - $L(\mathbf{y}, \hat{\mathbf{y}}) = -(\mathbf{y} \cdot \log(\hat{\mathbf{y}}) + (1 - \mathbf{y}) \log(1 - \hat{\mathbf{y}}))$

Entropia cruzada para $M > 2$

- 5 classes, classificação **correta**, com 72% de probabilidade:
 - $\mathbf{y} = [0 \quad 0 \quad 0 \quad 1 \quad 0]$
 - $\hat{\mathbf{y}} = [0,20 \quad 0,0 \quad 0,05, \quad 0,72 \quad 0,03]$
 - $L(\mathbf{y}, \hat{\mathbf{y}}) = -(0 \times \log 0,2 + 0 \times \log 0,0 + 0 \times \log 0,5 + 1 \times \log 0,72 + 0 \times \log 0,03)$
 - $L(\mathbf{y}, \hat{\mathbf{y}}) = -(\log 0,72) = 0,14267$

Entropia cruzada para $M > 2$

- 5 classes, classificação **correta**, com 72% de probabilidade:
 - $\mathbf{y} = [0 \quad 0 \quad 0 \quad 1 \quad 0]$
 - $\hat{\mathbf{y}} = [0,20 \quad 0,0 \quad 0,05 \quad 0,72 \quad 0,03]$
 - $L(\mathbf{y}, \hat{\mathbf{y}}) = -(0 \times \log 0,2 + 0 \times \log 0,0 + 0 \times \log 0,5 + 1 \times \log 0,72 + 0 \times \log 0,03)$
 - $L(\mathbf{y}, \hat{\mathbf{y}}) = -(\log 0,72) = 0,14267$
- 5 classes, classificação **correta**, com 52% de probabilidade:
 - $\mathbf{y} = [0 \quad 0 \quad 0 \quad 1 \quad 0]$
 - $\hat{\mathbf{y}} = [0,30 \quad 0,0 \quad 0,05 \quad 0,52 \quad 0,13]$
 - $L(\mathbf{y}, \hat{\mathbf{y}}) = -(0 \times \log 0,3 + 0 \times \log 0,0 + 0 \times \log 0,5 + 1 \times \log 0,52 + 0 \times \log 0,13)$
 - $L(\mathbf{y}, \hat{\mathbf{y}}) = -(\log 0,52) = 0,284$

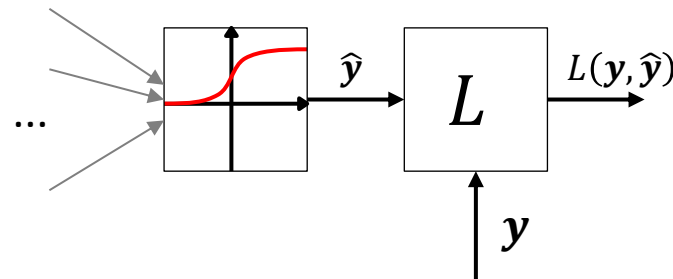
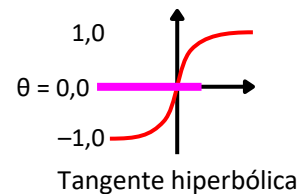
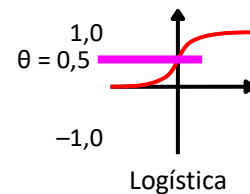
Entropia cruzada para $M > 2$

- 5 classes, classificação **correta**, com 72% de probabilidade:
 - $\mathbf{y} = [0 \quad 0 \quad 0 \quad 1 \quad 0]$
 - $\hat{\mathbf{y}} = [0,20 \quad 0,0 \quad 0,05 \quad 0,72 \quad 0,03]$
 - $L(\mathbf{y}, \hat{\mathbf{y}}) = -(0 \times \log 0,2 + 0 \times \log 0,0 + 0 \times \log 0,5 + 1 \times \log 0,72 + 0 \times \log 0,03)$
 - $L(\mathbf{y}, \hat{\mathbf{y}}) = -(\log 0,72) = 0,14267$
- 5 classes, classificação **correta**, com 52% de probabilidade:
 - $\mathbf{y} = [0 \quad 0 \quad 0 \quad 1 \quad 0]$
 - $\hat{\mathbf{y}} = [0,30 \quad 0,0 \quad 0,05 \quad 0,52 \quad 0,13]$
 - $L(\mathbf{y}, \hat{\mathbf{y}}) = -(0 \times \log 0,3 + 0 \times \log 0,0 + 0 \times \log 0,5 + 1 \times \log 0,52 + 0 \times \log 0,13)$
 - $L(\mathbf{y}, \hat{\mathbf{y}}) = -(\log 0,52) = 0,284$
- 5 classes, classificação **incorreta**:
 - $\mathbf{y} = [0 \quad 0 \quad 0 \quad 1 \quad 0]$
 - $\hat{\mathbf{y}} = [0,60 \quad 0,0 \quad 0,07 \quad 0,30 \quad 0,03]$
 - $L(\mathbf{y}, \hat{\mathbf{y}}) = -(0 \times \log 0,6 + 0 \times \log 0,0 + 0 \times \log 0,07 + 1 \times \log 0,3 + 0 \times \log 0,03)$
 - $L(\mathbf{y}, \hat{\mathbf{y}}) = -(\log 0,3) = 0,5229$

Entropia cruzada para M=2

- 2 classes, classificação correta:

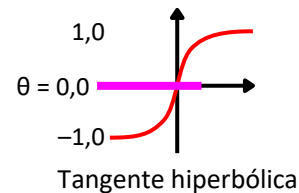
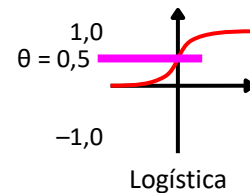
- $\mathbf{y} = [0]$
- $\hat{\mathbf{y}} = [0,20]$
- $L(\mathbf{y}, \hat{\mathbf{y}}) = -(0 \times \log 0,2 + (1 - 0) \times \log(1 - 0,2))$
- $L(\mathbf{y}, \hat{\mathbf{y}}) = -(0 \times \log 0,2 + (1) \times \log(0,8)) = -(\log(0,8)) = 0,09691$



Entropia cruzada para M=2

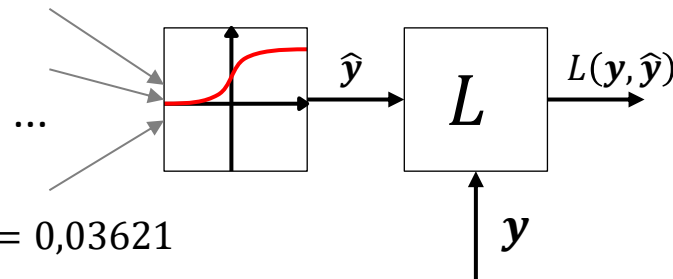
- 2 classes, classificação correta:

- $\mathbf{y} = [0]$
- $\hat{\mathbf{y}} = [0,20]$
- $L(\mathbf{y}, \hat{\mathbf{y}}) = -(0 \times \log 0,2 + (1 - 0) \times \log(1 - 0,2))$
- $L(\mathbf{y}, \hat{\mathbf{y}}) = -(0 \times \log 0,2 + (1) \times \log(0,8)) = -(\log(0,8)) = 0,09691$



- 2 classes, classificação correta:

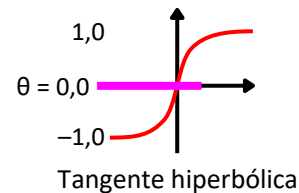
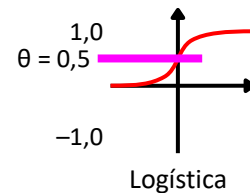
- $\mathbf{y} = [1]$
- $\hat{\mathbf{y}} = [0,92]$
- $L(\mathbf{y}, \hat{\mathbf{y}}) = -(1 \times \log 0,92 + (1 - 1) \times \log(1 - 0,92))$
- $L(\mathbf{y}, \hat{\mathbf{y}}) = -(1 \times \log 0,92 + (0) \times \log(0,08)) = -(\log(0,92)) = 0,03621$



Entropia cruzada para M=2

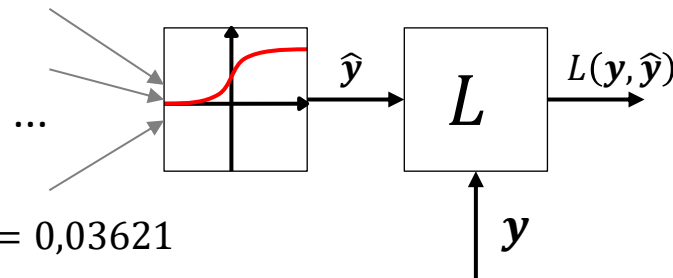
- 2 classes, classificação correta:

- $\mathbf{y} = [0]$
- $\hat{\mathbf{y}} = [0,20]$
- $L(\mathbf{y}, \hat{\mathbf{y}}) = -(0 \times \log 0,2 + (1 - 0) \times \log(1 - 0,2))$
- $L(\mathbf{y}, \hat{\mathbf{y}}) = -(0 \times \log 0,2 + (1) \times \log(0,8)) = -(\log(0,8)) = 0,09691$



- 2 classes, classificação correta:

- $\mathbf{y} = [1]$
- $\hat{\mathbf{y}} = [0,92]$
- $L(\mathbf{y}, \hat{\mathbf{y}}) = -(1 \times \log 0,92 + (1 - 1) \times \log(1 - 0,92))$
- $L(\mathbf{y}, \hat{\mathbf{y}}) = -(1 \times \log 0,92 + (0) \times \log(0,08)) = -(\log(0,92)) = 0,03621$



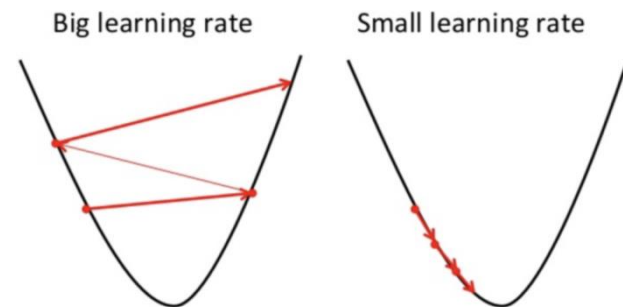
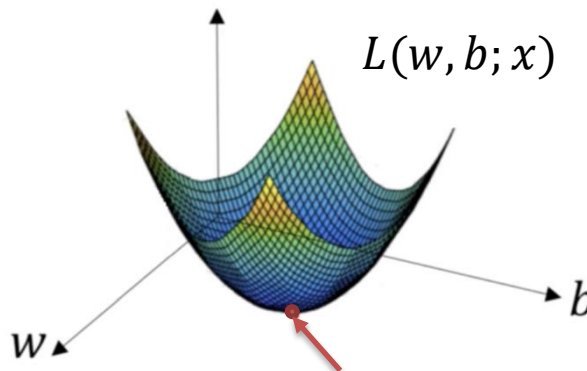
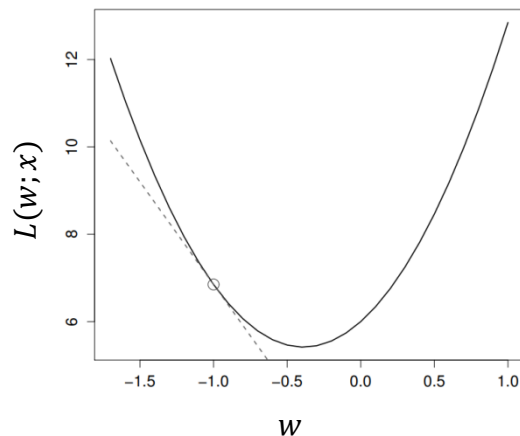
- 2 classes, classificação incorreta:

- $\mathbf{y} = [0]$
- $\hat{\mathbf{y}} = [0,65]$
- $L(\mathbf{y}, \hat{\mathbf{y}}) = -(0 \times \log 0,65 + (1 - 0) \times \log(1 - 0,65))$
- $L(\mathbf{y}, \hat{\mathbf{y}}) = -(0 \times \log 0,65 + (1) \times \log(0,35)) = -(\log(0,35)) = 0,45593$

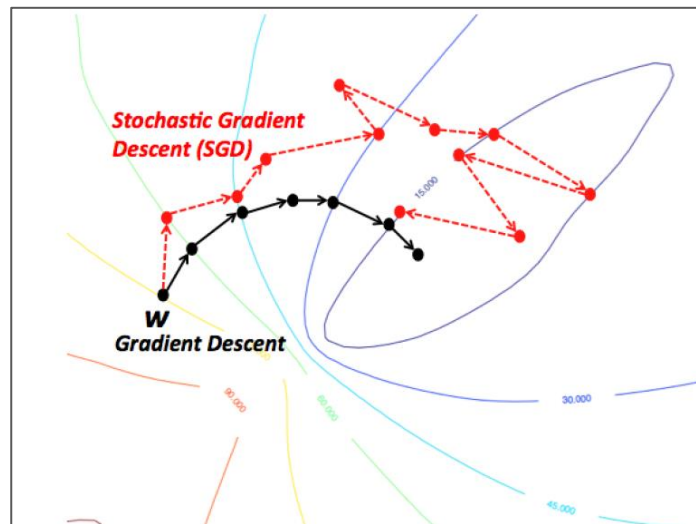
OTIMIZADORES

Otimizadores

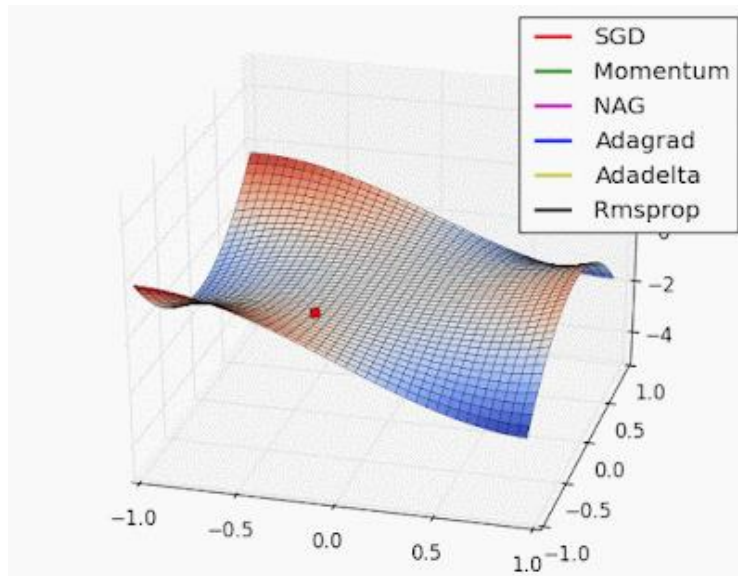
- Gradiente descendente (GD - *Gradient descent*):
 - $W_{t+1} = W_t - \eta \sum_{j=1}^N \nabla L(W; x_j)$
 - N é o tamanho do conjunto de treinamento



- Gradiente descendente estocástico (SGD – *Stochastic gradient descent*):
 - $W_{t+1} = W_t - \eta \sum_{j=1}^B \nabla L(W; x_j^B)$
 - B é o tamanho do mini-lote (*mini-batch*)



- SGD com momentum:
 - $W_{t+1} = W_t - \eta \sum_{j=1}^B \nabla L(W; x_j^B)$
 - B é o tamanho do mini-lote (*mini-batch*)
 - $W_{t+1} = W_t + \alpha(W_t - W_{t-1}) + (1 - \alpha)[- \eta \sum_{j=1}^B \nabla L(W; x_j^B)]$



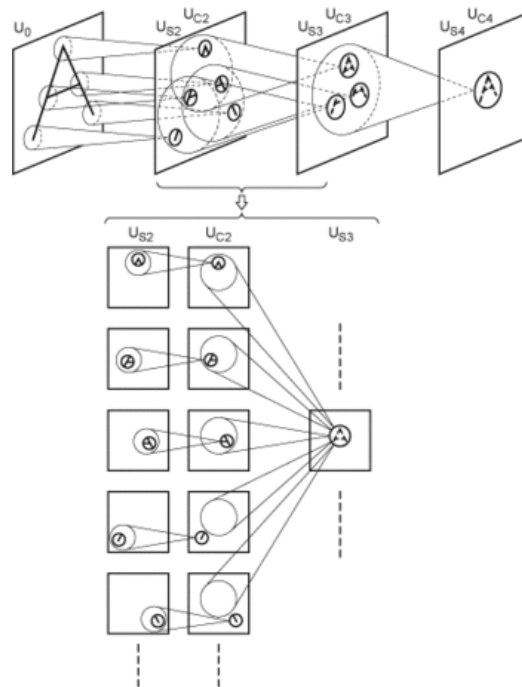
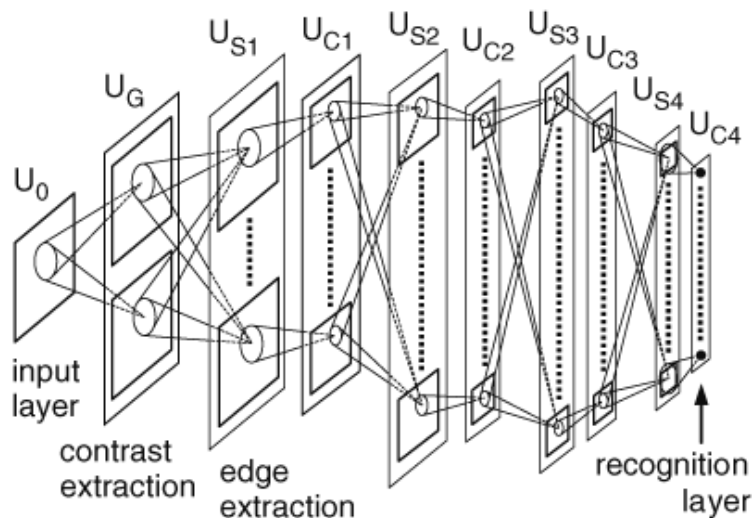
<http://www.denizyuret.com/2015/03/alec-radfords-animations-for.html>

- Outros otimizadores:
 - AdaGrad - *Adaptive Gradient*
 - AdaDelta - *Adaptive learning rate*
 - RMSProp - *Root Mean Squared Propagation*
 - Adam - *Adaptive moment estimation*
 - ...

ARQUITETURAS

Arquiteturas

- Neocognitron (1979)

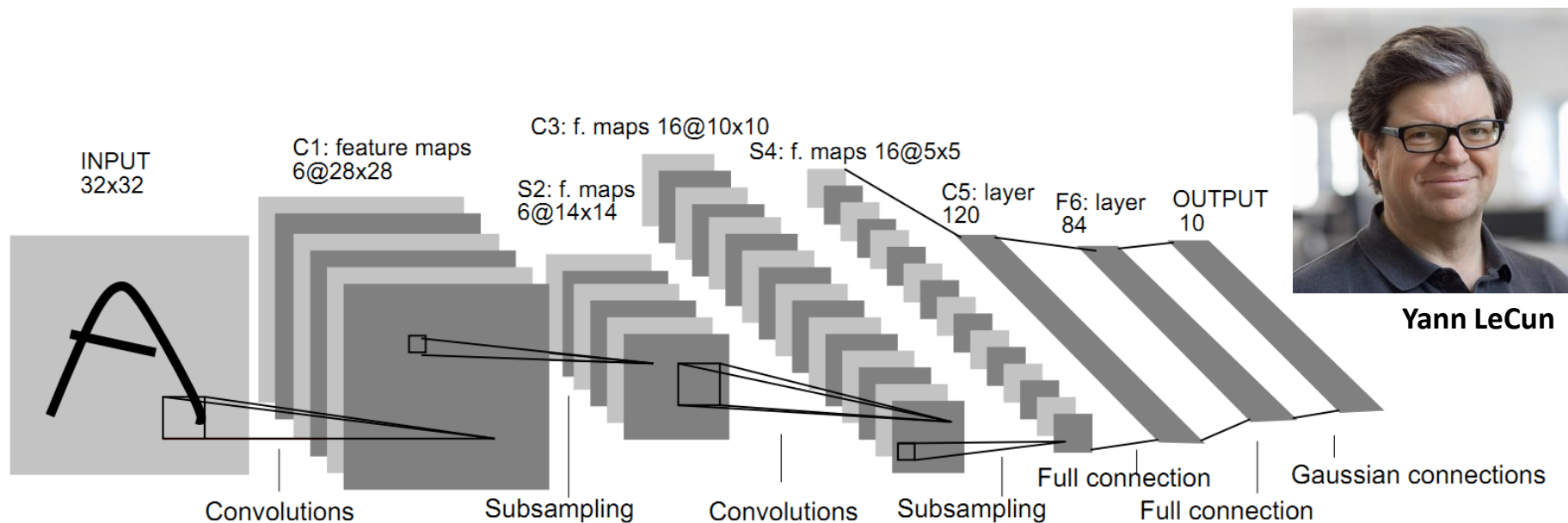


Kunihiro Fukushima

Fukushima, K. (1980). "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position". *Biological Cybernetics*. 36 (4)

Arquiteturas

- LeNet-5 (1998)

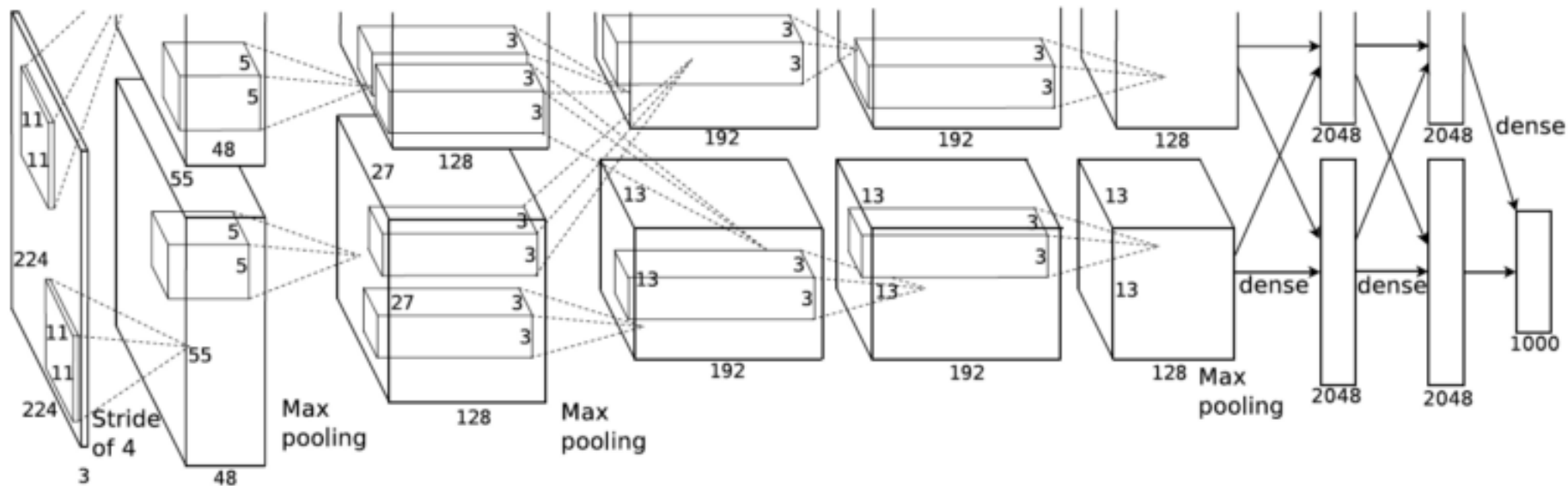


Yann LeCun

Lecun, Y. et al. (1998). "Gradient-based learning applied to document recognition". *Proceedings of the IEEE*. 86 (11): 2278–2324.

Arquiteturas

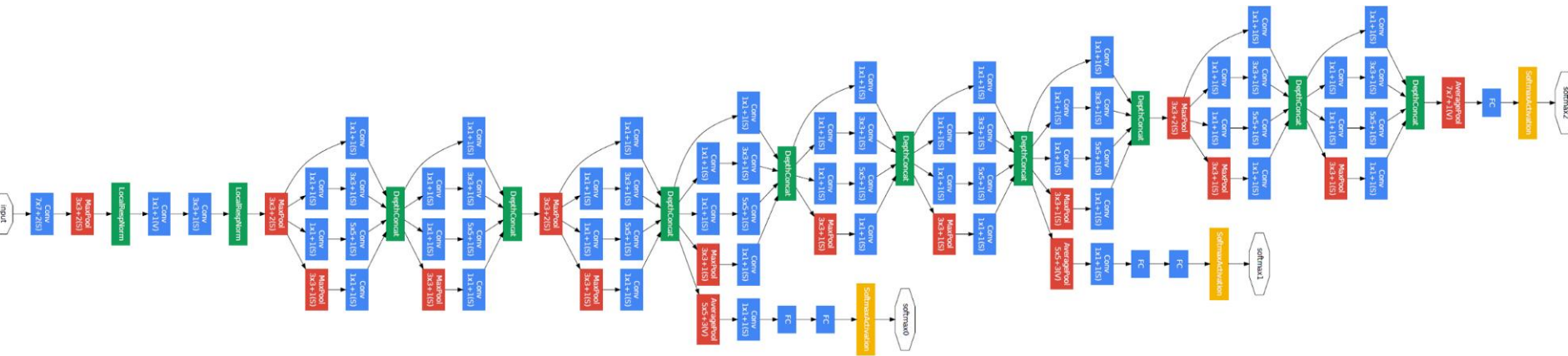
- AlexNet (2012)



Krizhevsky, Sutskever e Hinton. ImageNet Classification with Deep Convolutional Neural Networks. NeurIPS 2012

Arquiteturas

- Inception (GoogLeNet) (2015)



Szegedy, Christian (2015). "Going deeper with convolutions". CVPR2015.

-
- The diagram illustrates the VGG-19 architecture, showing the flow from input image to output layers, including convolutional and pooling layers, and the final classification layers.
- VGG-19 Architecture:**
- Input:** Image (size: 224).
 - Layer 1:** 3x3 conv, 64 (output size: 112).
 - Layer 2:** 3x3 conv, 64 (output size: 112).
 - Layer 3:** pool, /2 (output size: 56).
 - Layer 4:** 3x3 conv, 128 (output size: 56).
 - Layer 5:** 3x3 conv, 128 (output size: 56).
 - Layer 6:** pool, /2 (output size: 28).
 - Layer 7:** 3x3 conv, 256 (output size: 28).
 - Layer 8:** 3x3 conv, 256 (output size: 28).
 - Layer 9:** 3x3 conv, 256 (output size: 28).
 - Layer 10:** 3x3 conv, 256 (output size: 28).
 - Layer 11:** pool, /2 (output size: 14).
 - Layer 12:** 3x3 conv, 512 (output size: 14).
 - Layer 13:** 3x3 conv, 512 (output size: 14).
 - Layer 14:** 3x3 conv, 512 (output size: 14).
 - Layer 15:** 3x3 conv, 512 (output size: 14).
 - Layer 16:** pool, /2 (output size: 7).
 - Layer 17:** 3x3 conv, 512 (output size: 7).
 - Layer 18:** 3x3 conv, 512 (output size: 7).
 - Layer 19:** 3x3 conv, 512 (output size: 7).
 - Layer 20:** 3x3 conv, 512 (output size: 7).
 - Layer 21:** pool, /2 (output size: 1).
 - Layer 22:** fc 4096 (output size: 1).
 - Layer 23:** fc 4096 (output size: 1).
 - Layer 24:** fc 1000 (output size: 1).
- 34-layer plain:**
- Input:** Image.
 - Layer 1:** 7x7 conv, 64, /2.
 - Layer 2:** pool, /2.
 - Layer 3:** 3x3 conv, 64.
 - Layer 4:** 3x3 conv, 64.
 - Layer 5:** 3x3 conv, 64.
 - Layer 6:** 3x3 conv, 64.
 - Layer 7:** 3x3 conv, 64.
 - Layer 8:** 3x3 conv, 64.
 - Layer 9:** 3x3 conv, 128, /2.
 - Layer 10:** 3x3 conv, 128.
 - Layer 11:** 3x3 conv, 128.
 - Layer 12:** 3x3 conv, 128.
 - Layer 13:** 3x3 conv, 128.
 - Layer 14:** 3x3 conv, 128.
 - Layer 15:** 3x3 conv, 128.
 - Layer 16:** 3x3 conv, 256, /2.
 - Layer 17:** 3x3 conv, 256.
 - Layer 18:** 3x3 conv, 256.
 - Layer 19:** 3x3 conv, 256.
 - Layer 20:** 3x3 conv, 256.
 - Layer 21:** 3x3 conv, 256.
 - Layer 22:** 3x3 conv, 256.
 - Layer 23:** 3x3 conv, 256.
 - Layer 24:** 3x3 conv, 256.
 - Layer 25:** 3x3 conv, 256.
 - Layer 26:** 3x3 conv, 256.
 - Layer 27:** 3x3 conv, 256.
 - Layer 28:** 3x3 conv, 256.
 - Layer 29:** 3x3 conv, 256.
 - Layer 30:** 3x3 conv, 256.
 - Layer 31:** 3x3 conv, 256.
 - Layer 32:** 3x3 conv, 256.
 - Layer 33:** 3x3 conv, 256.
 - Layer 34:** 3x3 conv, 256.
 - Layer 35:** 3x3 conv, 512, /2.
 - Layer 36:** 3x3 conv, 512.
 - Layer 37:** 3x3 conv, 512.
 - Layer 38:** 3x3 conv, 512.
 - Layer 39:** 3x3 conv, 512.
 - Layer 40:** 3x3 conv, 512.
 - Layer 41:** 3x3 conv, 512.
 - Layer 42:** 3x3 conv, 512.
 - Layer 43:** 3x3 conv, 512.
 - Layer 44:** 3x3 conv, 512.
 - Layer 45:** 3x3 conv, 512.
 - Layer 46:** 3x3 conv, 512.
 - Layer 47:** 3x3 conv, 512.
 - Layer 48:** 3x3 conv, 512.
 - Layer 49:** 3x3 conv, 512.
 - Layer 50:** 3x3 conv, 512.
 - Layer 51:** 3x3 conv, 512.
 - Layer 52:** 3x3 conv, 512.
 - Layer 53:** 3x3 conv, 512.
 - Layer 54:** 3x3 conv, 512.
 - Layer 55:** 3x3 conv, 512.
 - Layer 56:** 3x3 conv, 512.
 - Layer 57:** 3x3 conv, 512.
 - Layer 58:** 3x3 conv, 512.
 - Layer 59:** 3x3 conv, 512.
 - Layer 60:** 3x3 conv, 512.
 - Layer 61:** 3x3 conv, 512.
 - Layer 62:** 3x3 conv, 512.
 - Layer 63:** 3x3 conv, 512.
 - Layer 64:** 3x3 conv, 512.
 - Layer 65:** 3x3 conv, 512.
 - Layer 66:** 3x3 conv, 512.
 - Layer 67:** 3x3 conv, 512.
 - Layer 68:** 3x3 conv, 512.
 - Layer 69:** 3x3 conv, 512.
 - Layer 70:** 3x3 conv, 512.
 - Layer 71:** 3x3 conv, 512.
 - Layer 72:** 3x3 conv, 512.
 - Layer 73:** 3x3 conv, 512.
 - Layer 74:** 3x3 conv, 512.
 - Layer 75:** 3x3 conv, 512.
 - Layer 76:** 3x3 conv, 512.
 - Layer 77:** 3x3 conv, 512.
 - Layer 78:** 3x3 conv, 512.
 - Layer 79:** 3x3 conv, 512.
 - Layer 80:** 3x3 conv, 512.
 - Layer 81:** 3x3 conv, 512.
 - Layer 82:** 3x3 conv, 512.
 - Layer 83:** 3x3 conv, 512.
 - Layer 84:** 3x3 conv, 512.
 - Layer 85:** 3x3 conv, 512.
 - Layer 86:** 3x3 conv, 512.
 - Layer 87:** 3x3 conv, 512.
 - Layer 88:** 3x3 conv, 512.
 - Layer 89:** 3x3 conv, 512.
 - Layer 90:** 3x3 conv, 512.
 - Layer 91:** 3x3 conv, 512.
 - Layer 92:** 3x3 conv, 512.
 - Layer 93:** 3x3 conv, 512.
 - Layer 94:** 3x3 conv, 512.
 - Layer 95:** 3x3 conv, 512.
 - Layer 96:** 3x3 conv, 512.
 - Layer 97:** 3x3 conv, 512.
 - Layer 98:** 3x3 conv, 512.
 - Layer 99:** 3x3 conv, 512.
 - Layer 100:** 3x3 conv, 512.
 - Layer 101:** 3x3 conv, 512.
 - Layer 102:** 3x3 conv, 512.
 - Layer 103:** 3x3 conv, 512.
 - Layer 104:** 3x3 conv, 512.
 - Layer 105:** 3x3 conv, 512.
 - Layer 106:** 3x3 conv, 512.
 - Layer 107:** 3x3 conv, 512.
 - Layer 108:** 3x3 conv, 512.
 - Layer 109:** 3x3 conv, 512.
 - Layer 110:** 3x3 conv, 512.
 - Layer 111:** 3x3 conv, 512.
 - Layer 112:** 3x3 conv, 512.
 - Layer 113:** 3x3 conv, 512.
 - Layer 114:** 3x3 conv, 512.
 - Layer 115:** 3x3 conv, 512.
 - Layer 116:** 3x3 conv, 512.
 - Layer 117:** 3x3 conv, 512.
 - Layer 118:** 3x3 conv, 512.
 - Layer 119:** 3x3 conv, 512.
 - Layer 120:** 3x3 conv, 512.
 - Layer 121:** 3x3 conv, 512.
 - Layer 122:** 3x3 conv, 512.
 - Layer 123:** 3x3 conv, 512.
 - Layer 124:** 3x3 conv, 512.
 - Layer 125:** 3x3 conv, 512.
 - Layer 126:** 3x3 conv, 512.
 - Layer 127:** 3x3 conv, 512.
 - Layer 128:** 3x3 conv, 512.
 - Layer 129:** 3x3 conv, 51

He et al. Deep Residual Learning for Image Recognition. 2015.

Arquiteturas

- DenseNet (2017)

Huang et al. *Densely Connected Convolutional Networks*. CVPR 2017.

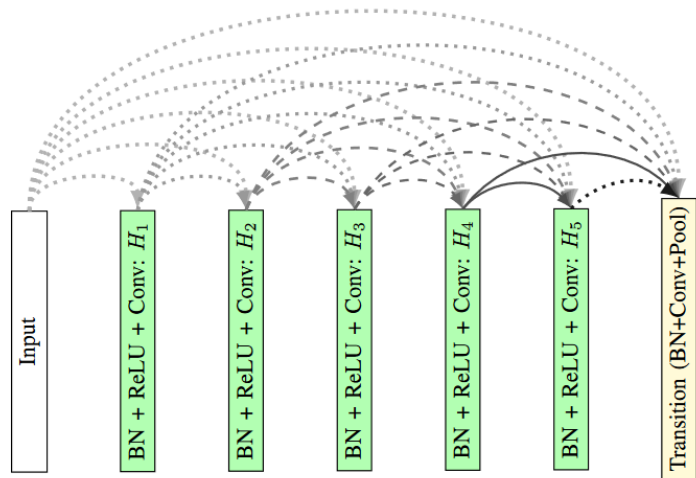


Figure 10. Illustration of a DenseBlock with 5 functions H_i and a Transition Layer.

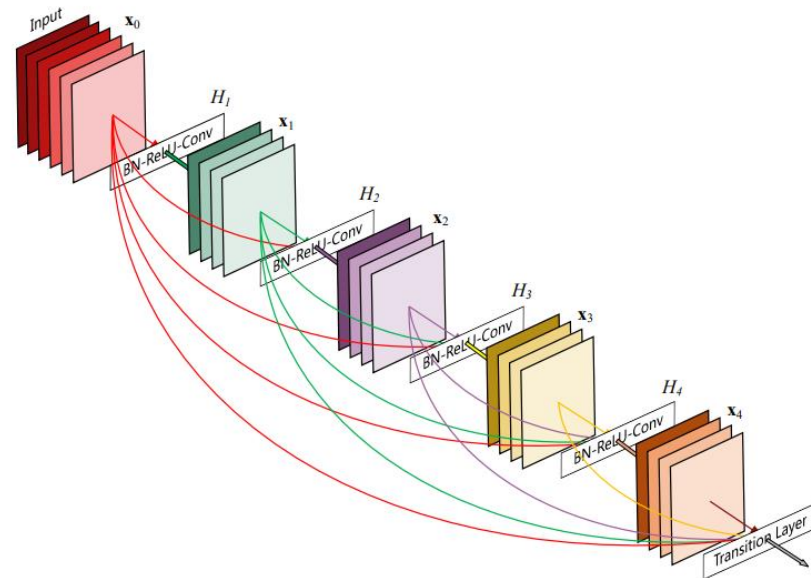
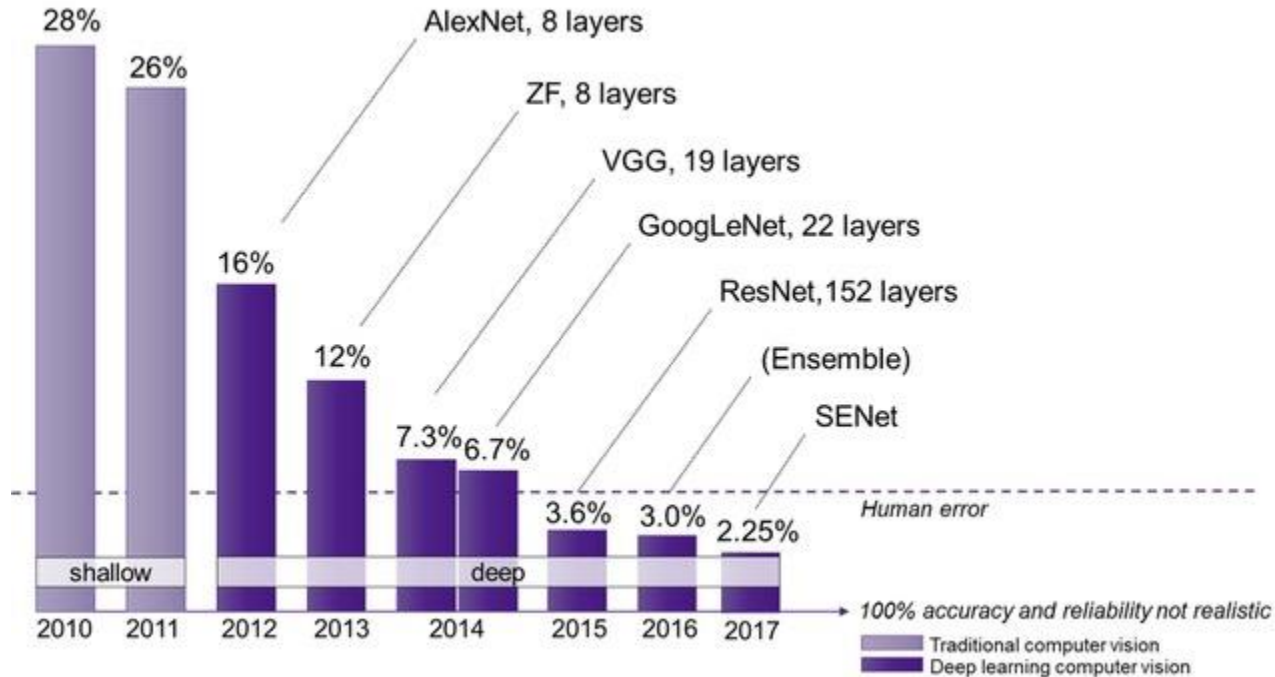


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

Ponti et al. *Everything You Wanted to Know about Deep Learning for Computer Vision but Were Afraid to Ask*. Sibgrapi 2017.

Arquiteturas

- ImageNet Large Scale Visual Recognition Challenge
 - <https://image-net.org/challenges/LSVRC/>



<https://semiengineering.com/new-vision-technologies-for-real-world-applications/>

BIBLIOTECAS E DESENVOLVIMENTO

- O treinamento de CNNs possui alto custo computacional.
 - Recomenda-se que sejam treinados usando GPUs.
 - O Google Colab fornece acesso à GPUs (com algumas restrições).



Bibliotecas e desenvolvimento

- Principais bibliotecas para Deep Learning e Redes Neurais Convolucionais
 - PyTorch
 - <https://pytorch.org/>
 - Tensorflow
 - <https://www.tensorflow.org/>



Bibliotecas e desenvolvimento

- **Anaconda Distribution:**
 - Distribuição Python com suporte às principais bibliotecas
 - <https://www.anaconda.com/products/distribution>
- **Google Colab:**
 - Ambiente de execução em nuvem com GPUs.
 - <https://colab.research.google.com>



CONJUNTOS DE IMAGENS

Conjuntos de imagens

- MNIST

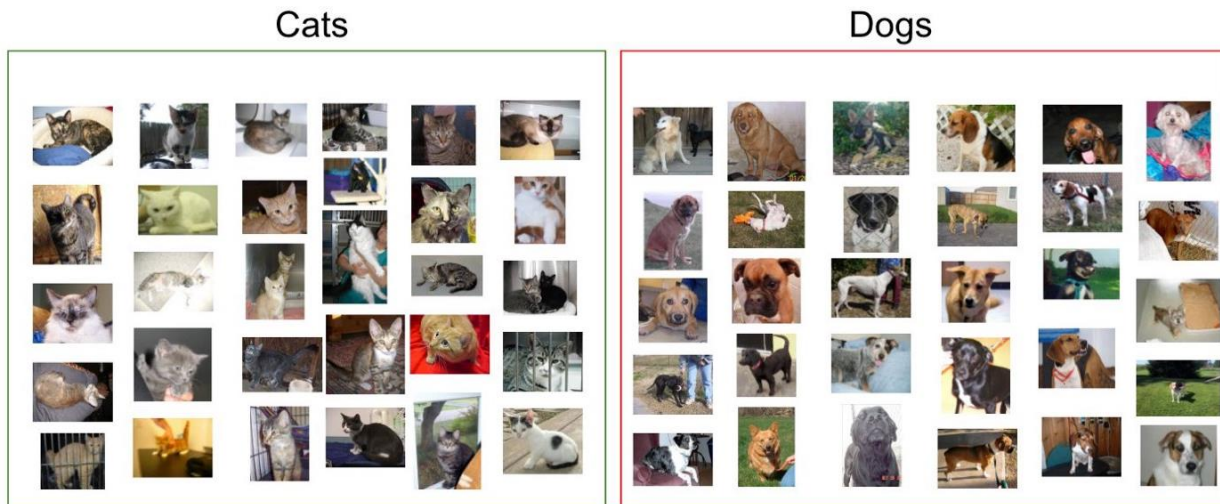
- <http://yann.lecun.com/exdb/mnist/>
- 60,000 training images
- 10,000 testing images
- 28 x 28 pixels
- Níveis de cinza



Conjuntos de imagens

- **Cats vs. Dogs:**

- <https://www.kaggle.com/c/dogs-vs-cats>
- 25,000 images de treinamento
- 12,500 imagens de teste
- 2 classes
- Diversos tamanhos
- RGB



Sample of cats & dogs images from Kaggle Dataset

Conjuntos de imagens

- **CIFAR10:**

- <https://www.cs.toronto.edu/~kriz/cifar.html>
- 50,000 training images
- 10,000 testing images
- 10 classes
- 32 x 32 pixels
- RGB

airplane

automobile

bird

cat

deer

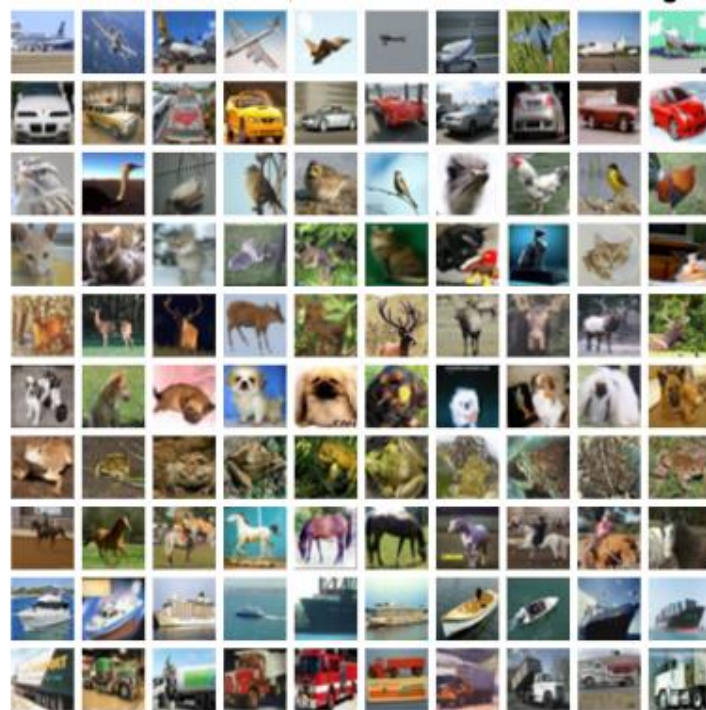
dog

frog

horse

ship

truck



- **ImageNet:**

- <https://www.image-net.org/>
- ~1,000,000 imagens
- 1,000 classes
- RGB

IMGENET



- Ponti et al. **Everything You Wanted to Know about Deep Learning for Computer Vision but Were Afraid to Ask**. Sibgrapi 2017.
- Moacir Ponti (ICMC-USP). **Material para o minicurso *Deep Learning***
 - https://github.com/maponti/deeplearning_intro_datascience
- **Learn TensorFlow and deep learning, without a Ph.D.**
 - <https://cloud.google.com/blog/products/gcp/learn-tensorflow-and-deep-learning-without-a-phd>
- CS231n: Convolutional Neural Networks for Visual Recognition
 - <http://cs231n.github.io/>
- Goodfellow, Bengio e Courville. **Deep Learning**. MIT Press, 2016
 - <https://www.deeplearningbook.org/>
- The MathWorks, Inc. **What is a Convolutional Neural Network? 3 things you need to know.**
 - <https://www.mathworks.com/discovery/convolutional-neural-network-matlab.html>

Bibliografia

- Fukushima, K. (1980). **Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position**. Biological Cybernetics. 36 (4): 193–202.
– [10.1007/bf00344251](https://doi.org/10.1007/bf00344251)
- Lecun, Y. et al. (1998). **Gradient-based learning applied to document recognition**. Proceedings of the IEEE. 86 (11): 2278–2324.
– [10.1109/5.726791](https://doi.org/10.1109/5.726791)
- Krizhevsky, Sutskever e Hinton. **ImageNet Classification with Deep Convolutional Neural Networks**. NeurIPS 2012.
- Szegedy, Christian (2015). **Going deeper with convolutions**. CVPR2015.
- Simonyan e Zisserman. **Very Deep Convolutional Networks for Large-Scale Image Recognition**. 2014.
- He et al. **Deep Residual Learning for Image Recognition**. 2015.
- Huang et al. **Densely Connected Convolutional Networks**. CVPR 2017.
- Rodrigues, L. F.; Naldi M. C., Mari, J. F. **Comparing convolutional neural networks and preprocessing techniques for HEP-2 cell classification in immunofluorescence images**. Computers in Biology and Medicine, 2019.
– <https://doi.org/10.1016/j.combiomed.2019.103542>

FIM