Lecture 02 – Convolutional Neural Networks

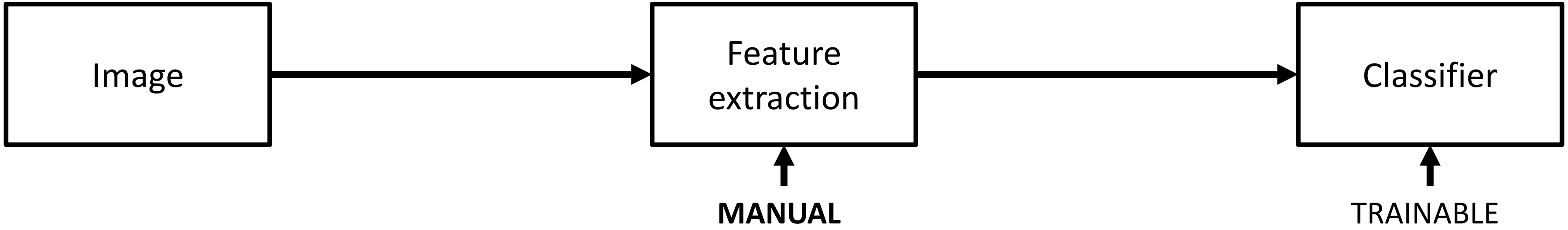
Prof. João Fernando Mari
joaofmari.github.io
joaof.mari@ufv.br

Agenda

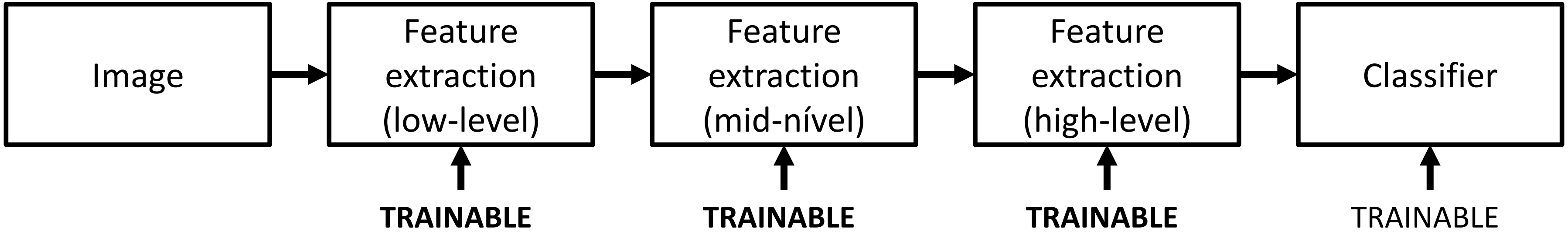
- Classification pipelines
- Multi-layer Perceptron (MLP)
- Convolutional Neural Networks (CNNs)
- Convolutional layer
- Pooling layer
- Activation function
- Fully connected layer
- Output layer - softmax
- Loss function
- Optimizers
- Architectures
- Development and libraries
- Image datasets

Classification pipelines

The classic image classification pipeline

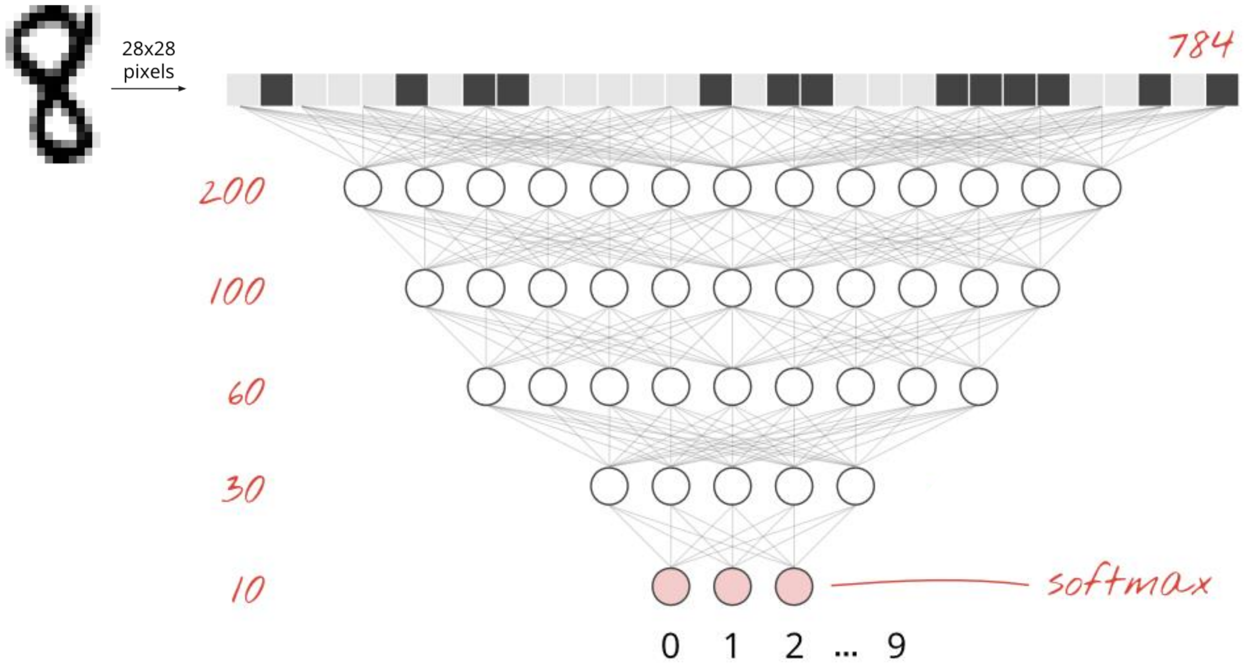


Deep Learning



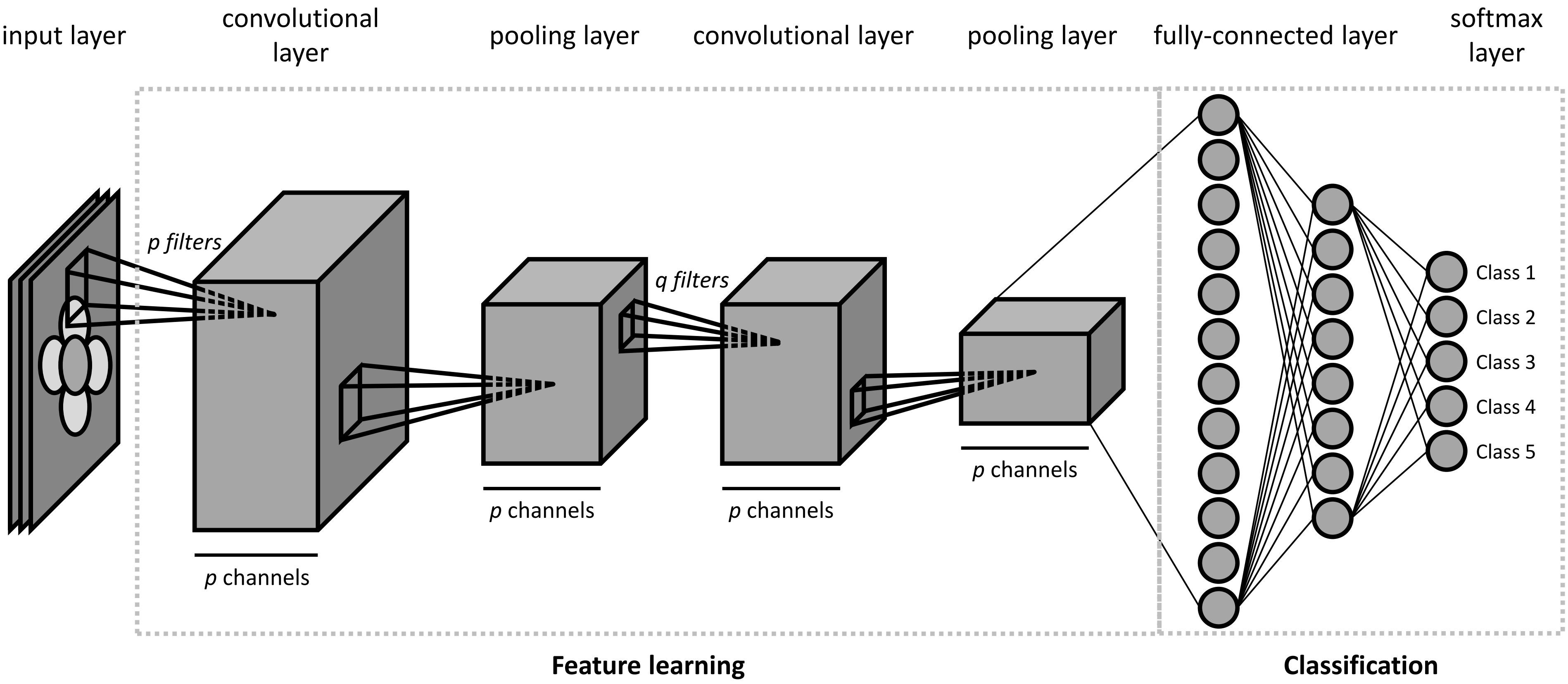
Yann LeCun's Deep Learning Course at CDS - SPRING 2021

Multi-layer Perceptron (MLP)



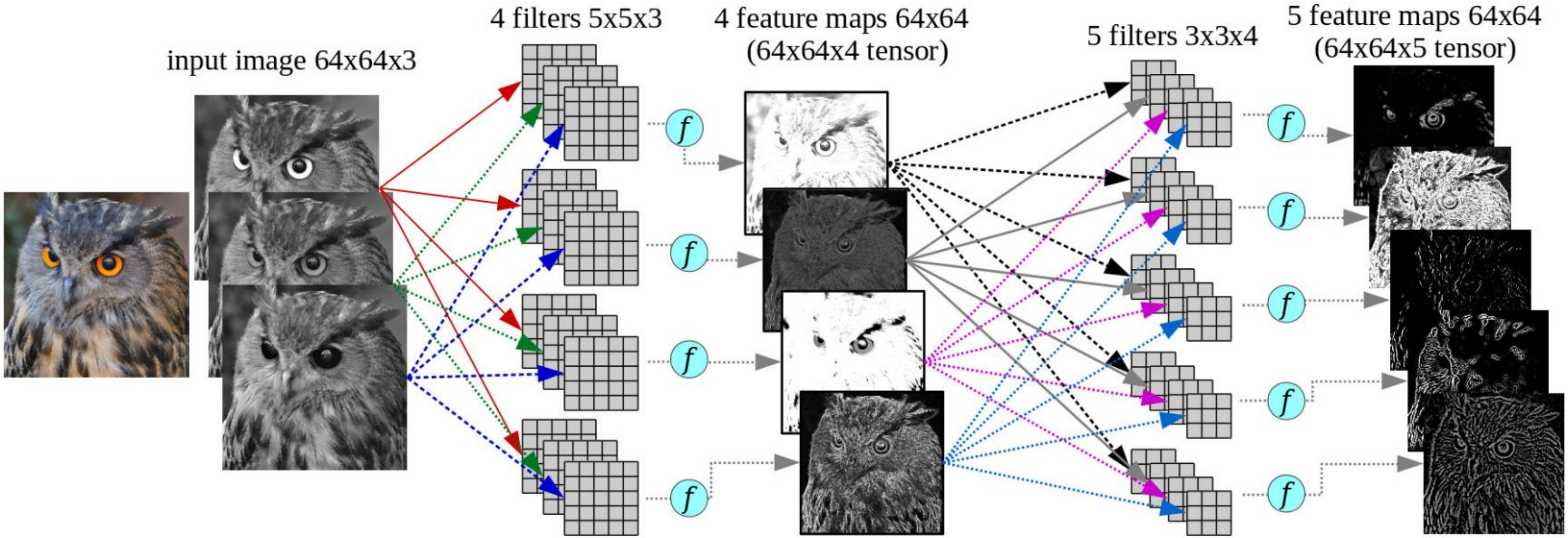
Learn TensorFlow and deep learning, without a Ph.D.

Convolutional Neural Networks (CNNs)



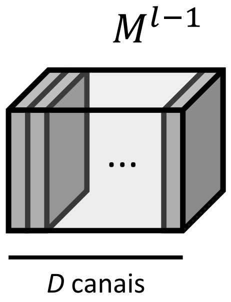
CONVOLUTIONAL LAYER

Convolutional layer



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

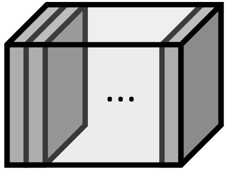
Convolutional layer



Convolutional layer

Convolutional layer C^l

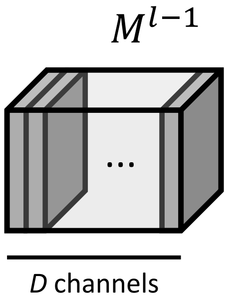
M^{l-1}

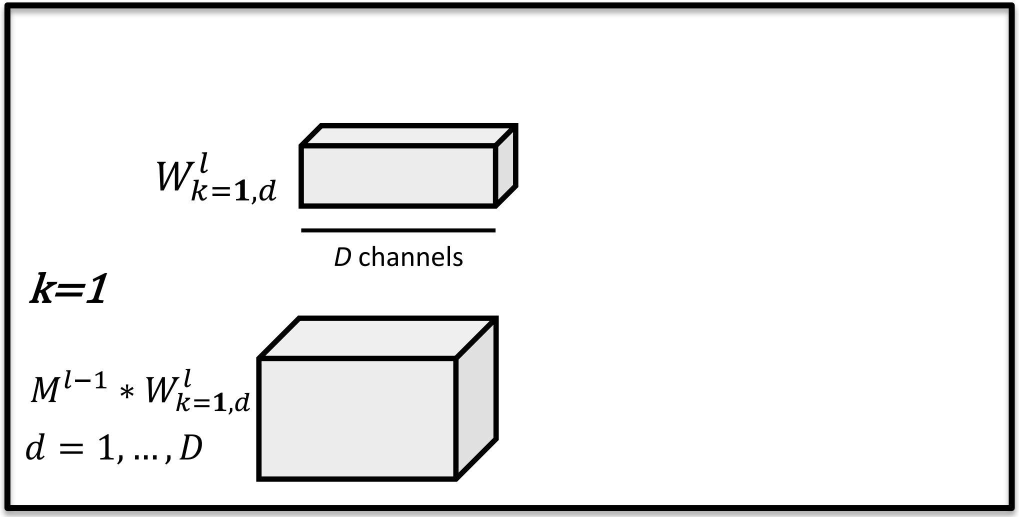


$k=1$

Convolutional layer

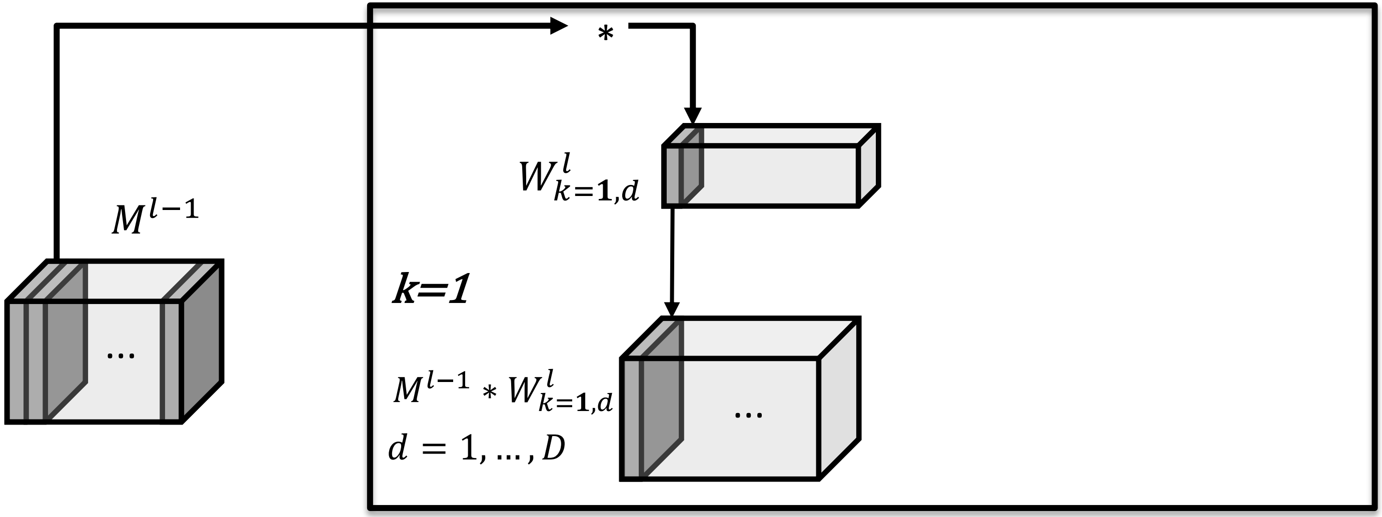
Convolutional layer C^l





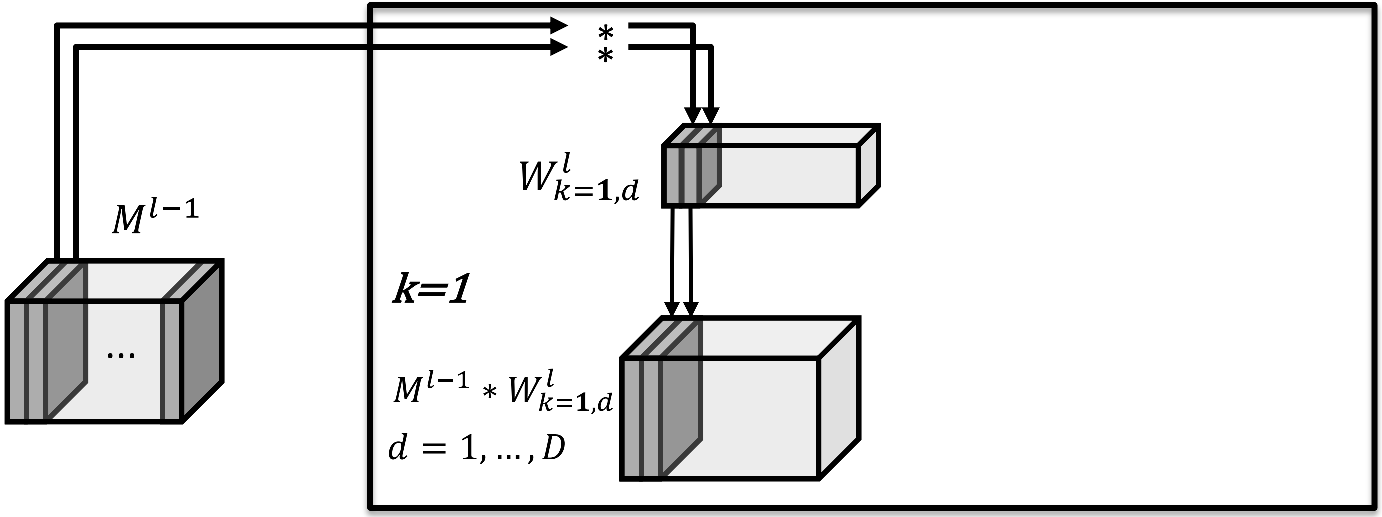
Convolutional layer

Convolutional layer C^l



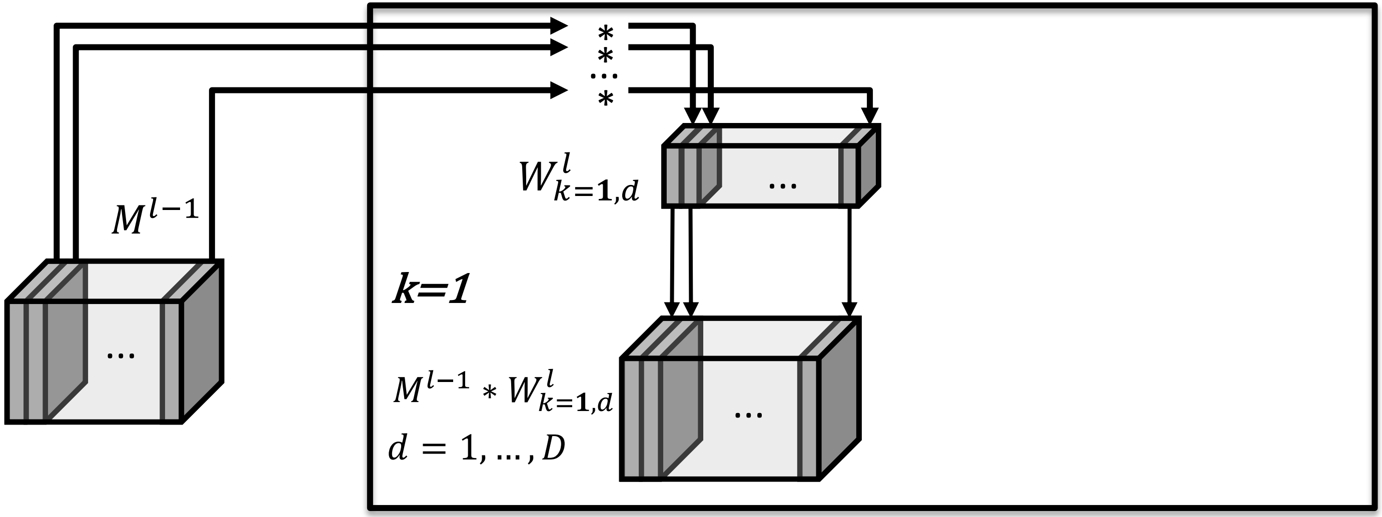
Convolutional layer

Convolutional layer C^l



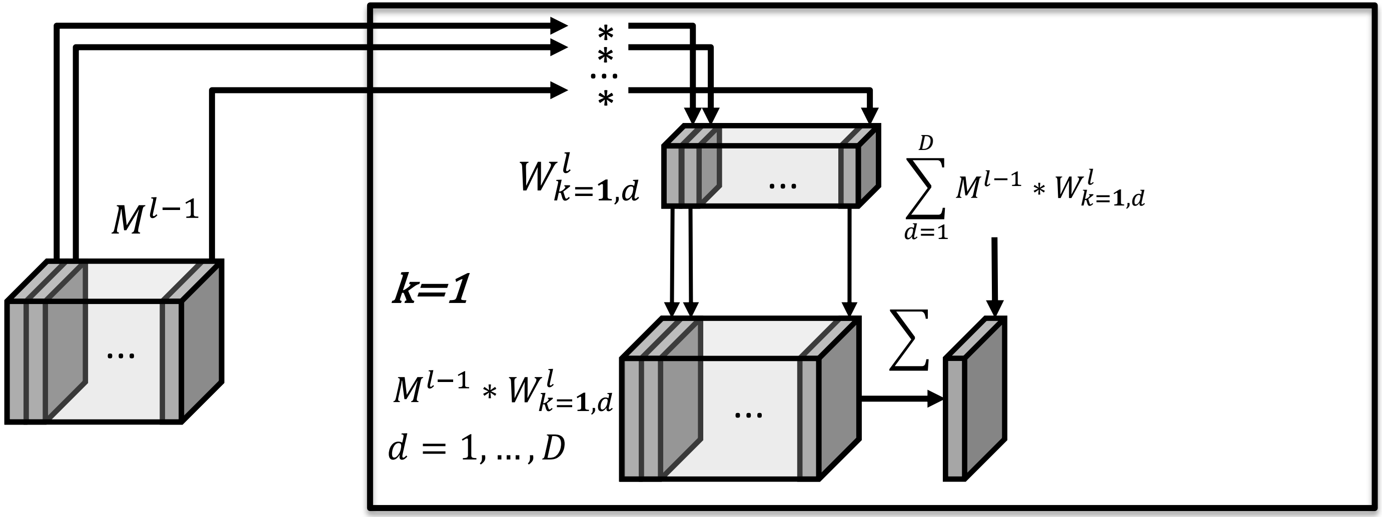
Convolutional layer

Convolutional layer C^l

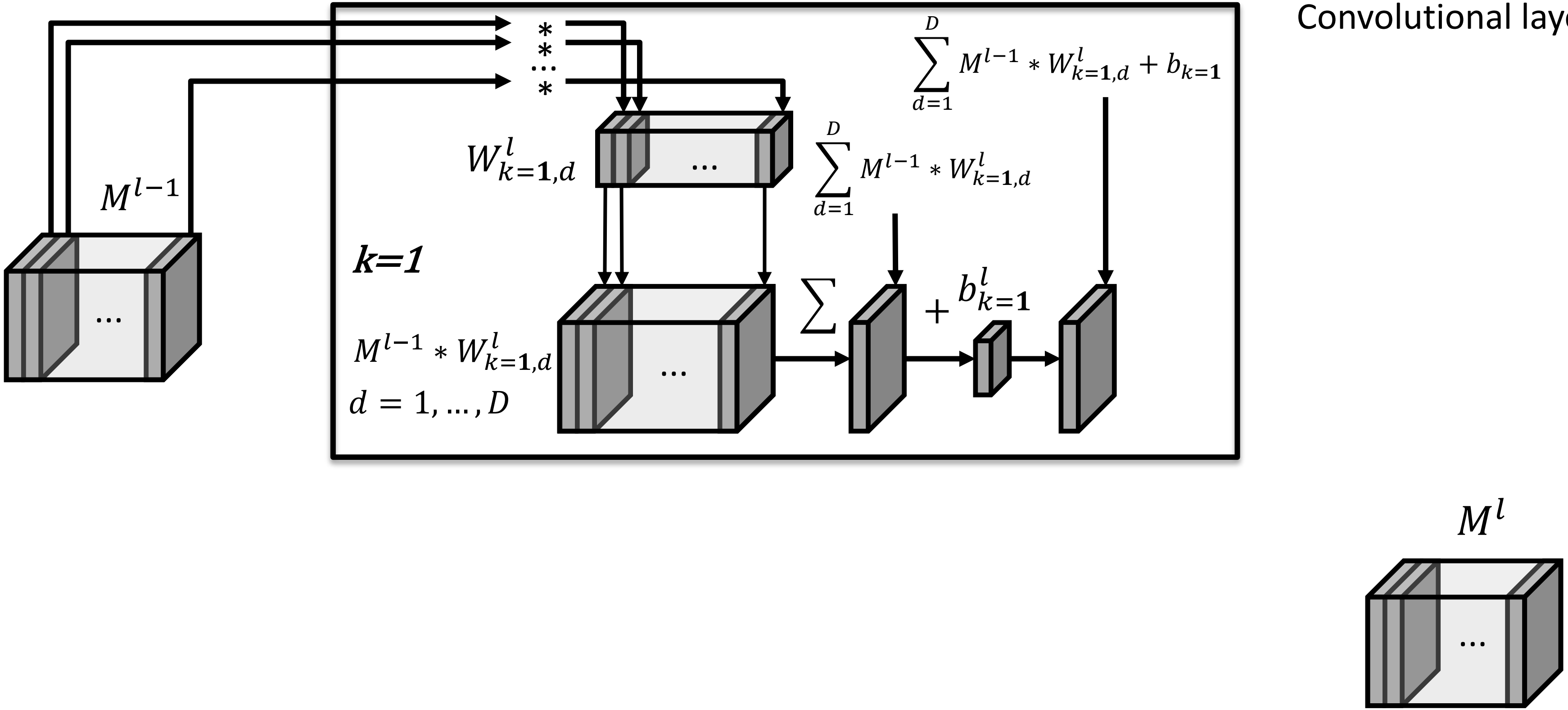


Convolutional layer

Convolutional layer C^l

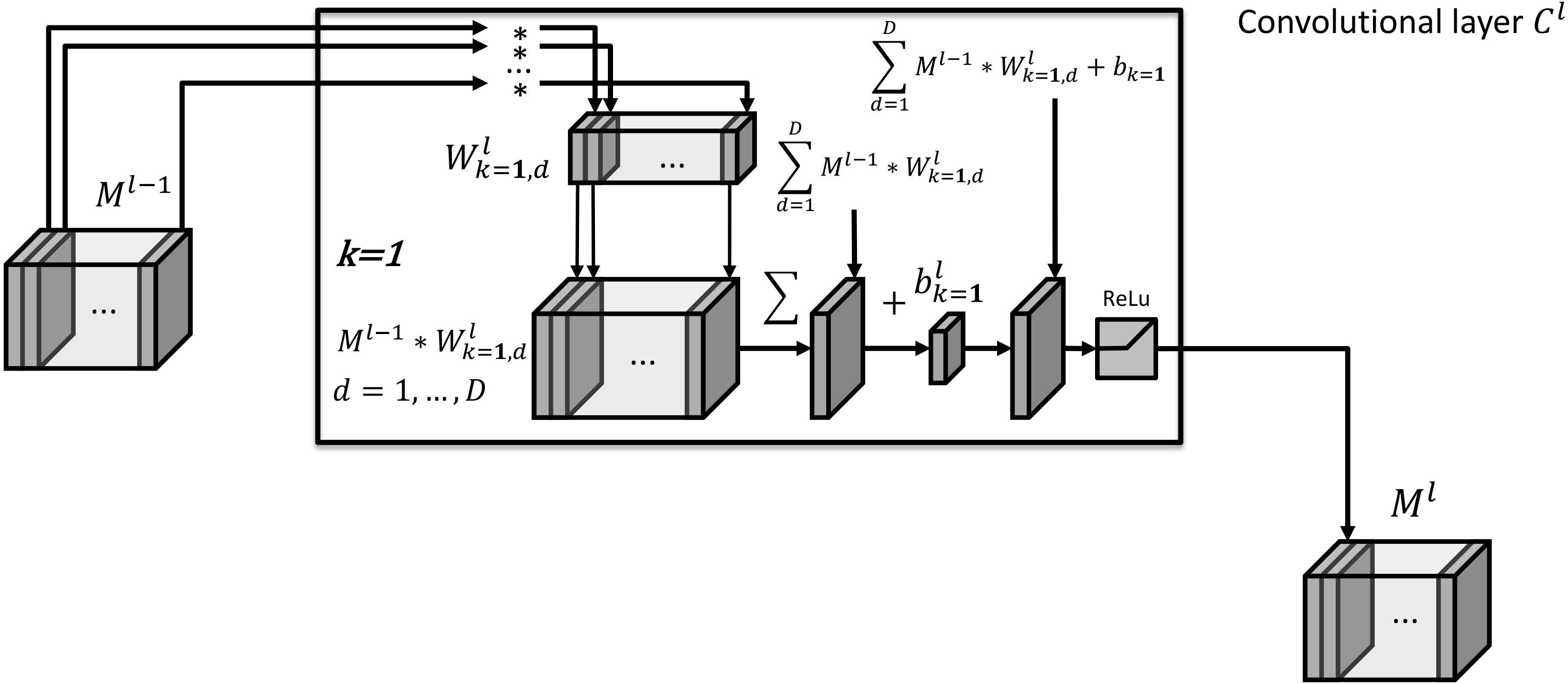


Convolutional layer

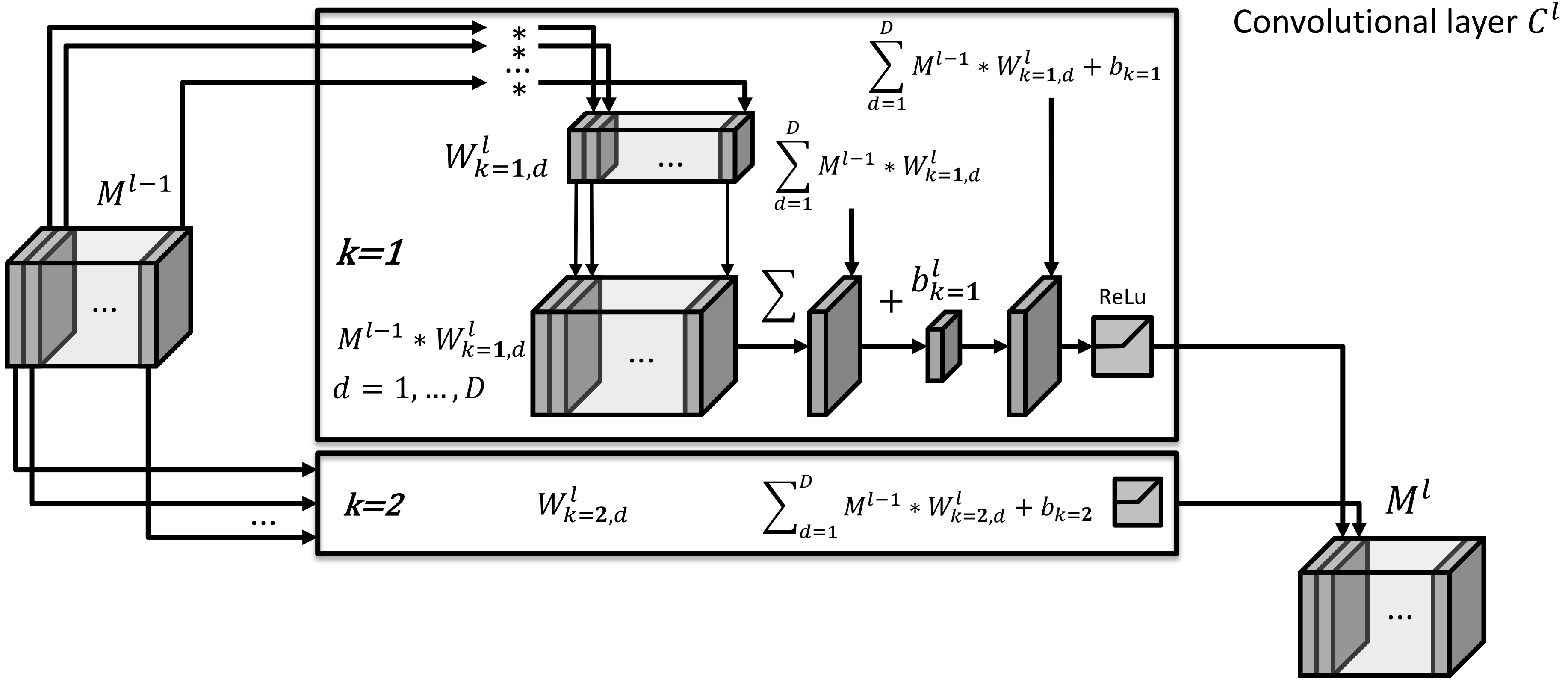


Convolutional layer C^{l}

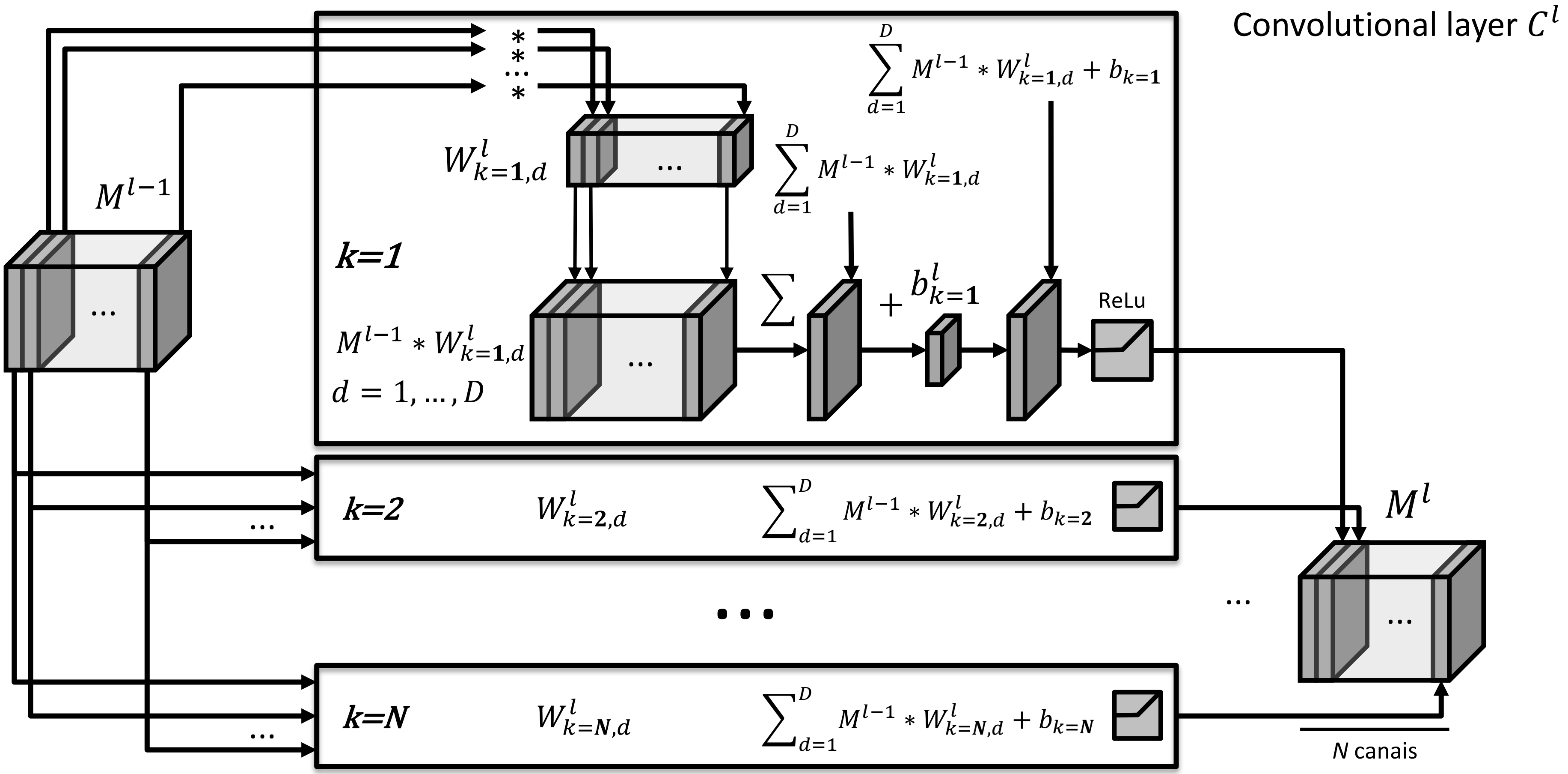
Convolutional layer



Convolutional layer



Convolutional layer



Convolutional layer

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]

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

b0

1

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]

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

b1

0

Convolutional layer

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]

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

b0

1

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]

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

b1

0

Convolutional layer

UFV Universidade Federal de Viçosa

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]

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

b0

1

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]

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

b1

0

Convolutional layer

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				

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

4				

b0

1

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

-17				

---

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				

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

-13				

b1

0

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

-2				

Convolutional layer

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			

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

4	7			

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

-17	0			

b0

1

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			

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

-13	-11			

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

-2	-21			

b1

0

Convolutional layer

UFV Universidade Federal de Viçosa

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		

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

4	7	6		

b0

1

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

-17	0	14		

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		

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

-13	-11	-21		

b1

0

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

-2	-21	-1		

Convolutional layer

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	

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

4	7	6	-1	

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

-17	0	14	-2	

b0

1

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	

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

-13	-11	-21	-17	

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

-2	-21	-1	3	

b1

0

Convolutional layer

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

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

4	7	6	-1	12

b0

1

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

-17	0	14	-2	-8

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

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

-13	-11	-21	-17	-9

b1

0

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

-2	-21	-1	3	-17

Convolutional layer

Universidade Federal de Viçosa

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				

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				

Convolutional layer

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
12				

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

4	7	6	-1	12
4				

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

-17	0	14	-2	-8
-17				

b0

1

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				

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

-2	-21	-1	3	-17
3				

b1

0

Convolutional layer

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

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

Convolutional layer

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

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

Convolutional layer

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

$\sum$

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

$\sum$

v[:,:,1]

Convolutional layer

Universidade Federal de Viçosa

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

∑

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

∑

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

Convolutional layer

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

$\sum$

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

f

ReLu

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

$\sum$

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

f

ReLu

y[:,:,1]

Convolutional layer

UFV
Universidade Federal de Viçosa

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

∑

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

f

ReLu

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

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

∑

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

f

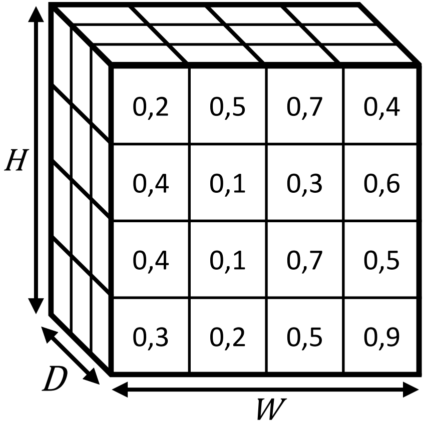
ReLu

y[:,:,1]

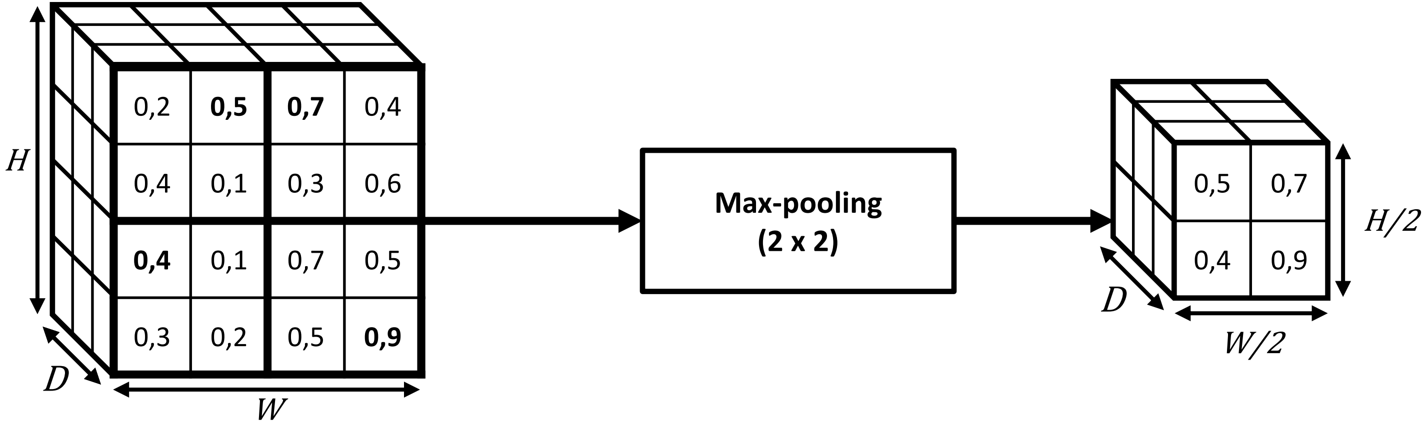
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	4
0	0	5	17	28

POOLING LAYER

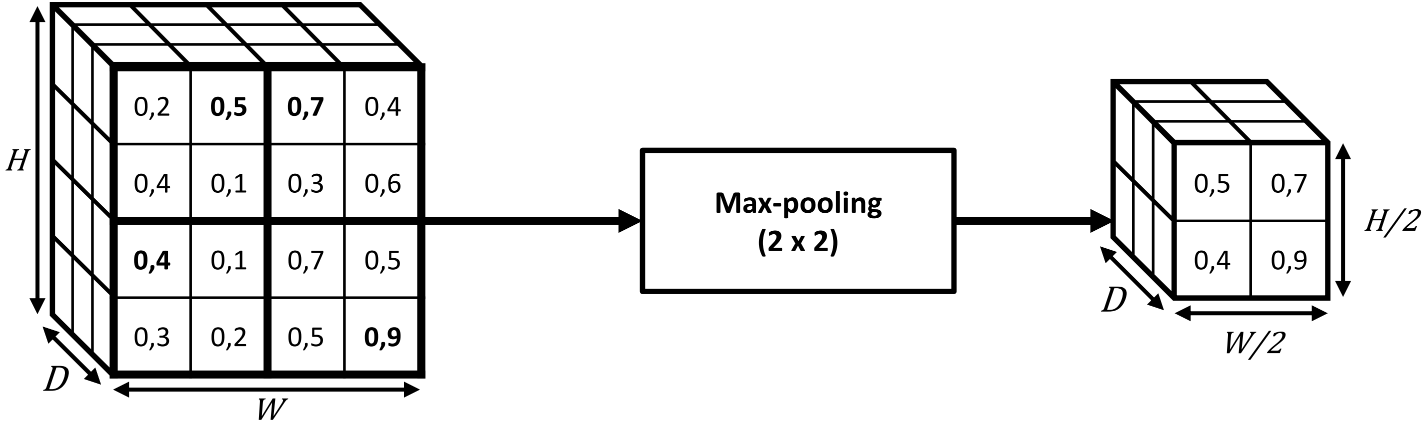
Pooling layer

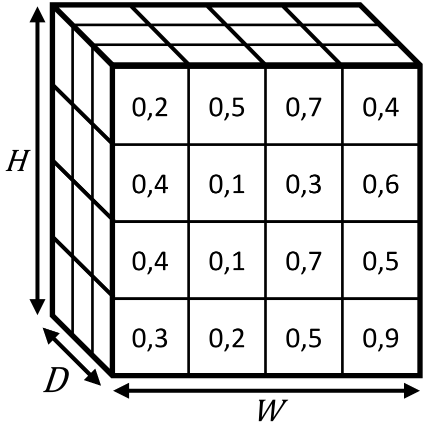


Pooling layer

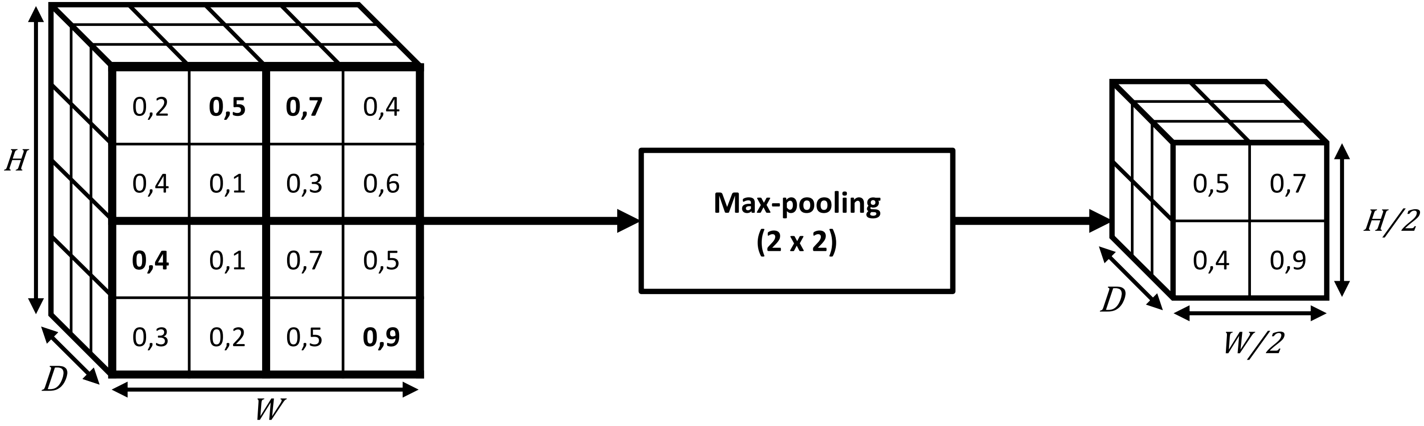


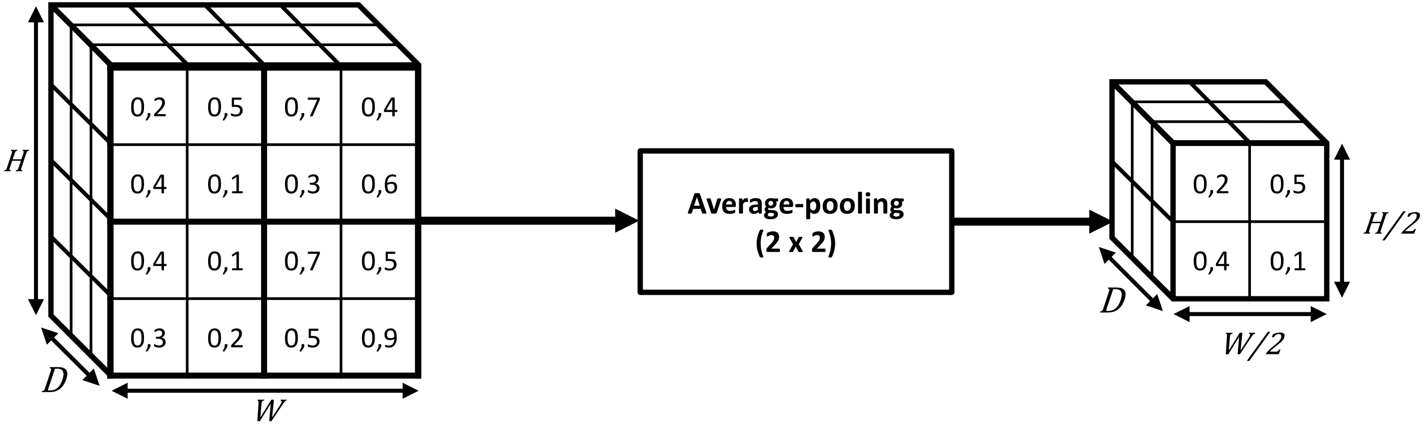
Pooling layer





Pooling layer

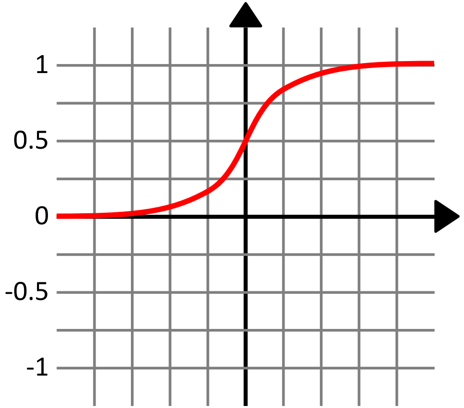




ACTIVATION FUNCTION

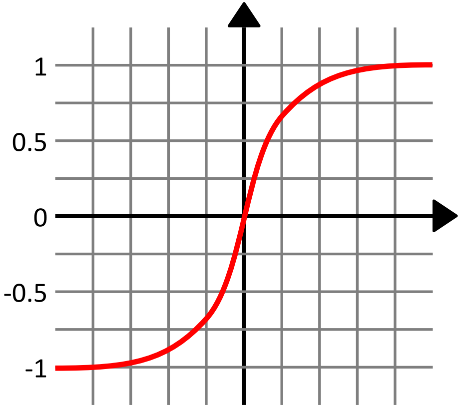
Activation function

Logistic



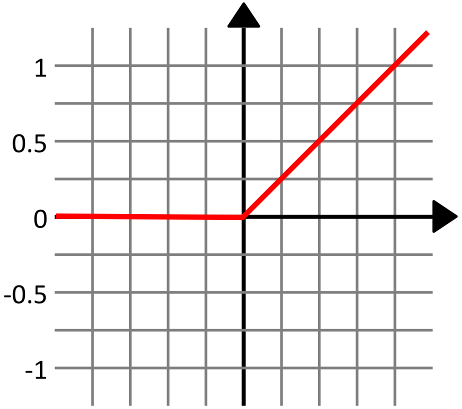
$$y = \frac{1}{1 + e^{-v}}$$

Hiperbolic tangent



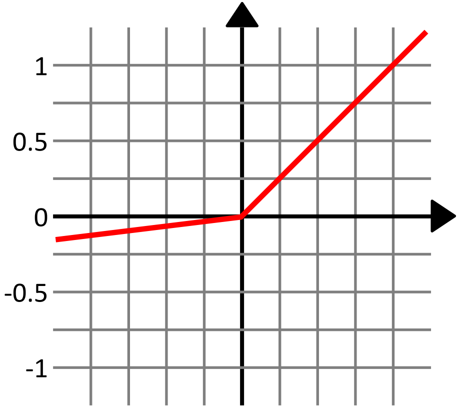
$$y = tanh(v) = \frac{sen(v)}{\cos(v)}$$

ReLu



$$y = max(0, v)$$

PReLu

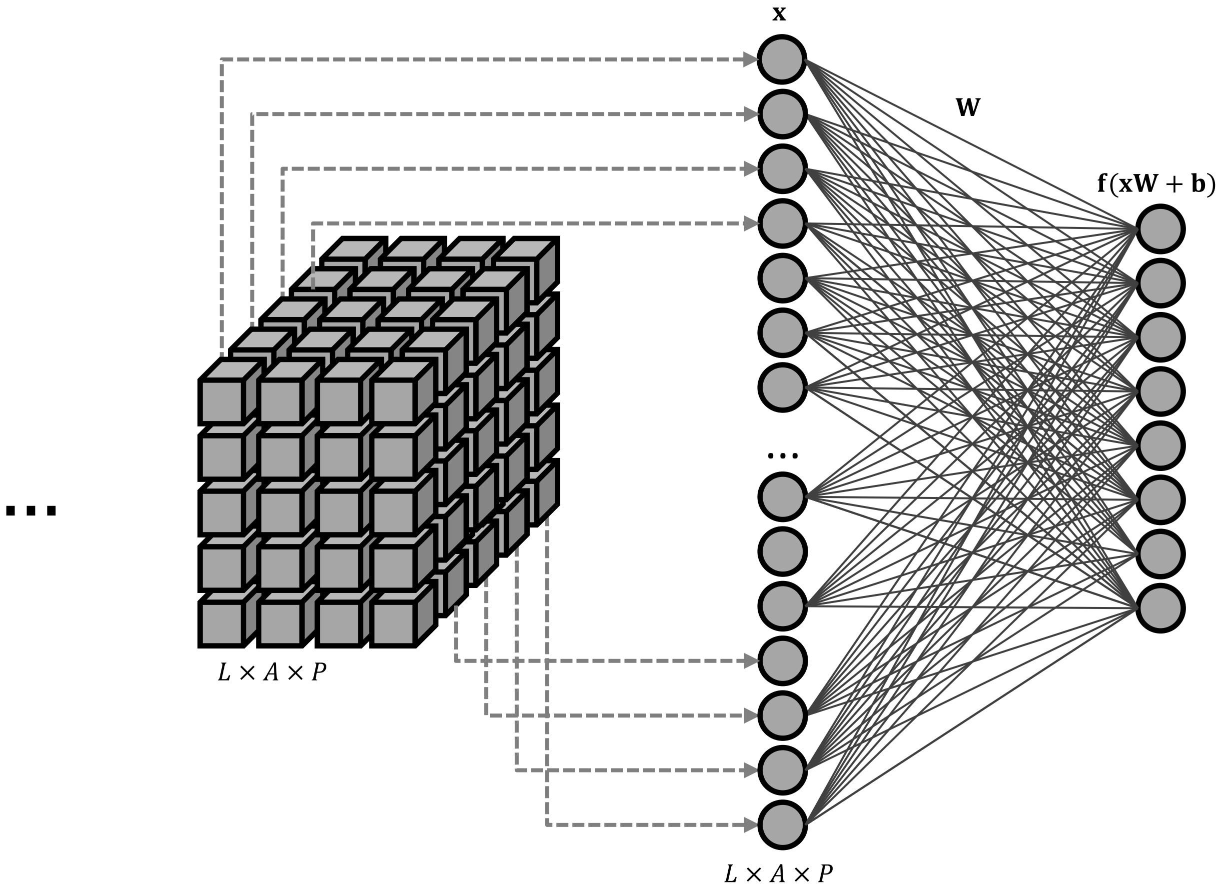


$$y = max(av, v)$$

- If *a*=0,01 → Leak ReLu

FULLY CONNECTED LAYER

Fully connected layer



OUTPUT LAYER - SOFTMAX

Output layer - softmax

- Softmax function for M classes:
  - $softmax(x_i) = \frac{e^{x_i}}{\sum_{j=0}^{M-1} e^{x_j}}$

- **Example:**
  - $\mathbf{x} = [-0.8 \quad 2.0 \quad 6.0 \quad -2.7 \quad 0.8]$
    - $\sum_{j=0}^{M-1} x_j = 5{,}3$
    - *Sum != 1.0. It cannot be interpreted as probabilities.*
  - $\sum_{j=0}^{M-1} e^{x_j} = 0.4493 + 7.3891 + 403.4288 + 0.0672 + 2.2255 = 413.5599$
  - $softmax(x_i) = [0.0011 \quad 0.0179 \quad 0.9755 \quad 0.0002 \quad 0.0054]$
    - $\sum_{j=0}^{M-1} softmax(x_i) = 1.0$
    - *The probability of the sample belonging to each class.*

LOSS FUNCTION

Cross-entropy loss

- Cross-entropy for more than 2 classes (M>2):
  - $L(\boldsymbol{y}, \widehat{\boldsymbol{y}}) = -\sum_{j=0}^{M-1} \boldsymbol{y}_j \cdot \log(\widehat{\boldsymbol{y}}_j)$

- Cross-entropy for 2 classes (M=2):
  - $L(\boldsymbol{y}, \widehat{\boldsymbol{y}}) = -(\boldsymbol{y} \cdot \log(\widehat{\boldsymbol{y}}) + (1 - \boldsymbol{y}) \log(1 - \widehat{\boldsymbol{y}}))$

Cross-entropy for M>2

- 5 classes, **correct** classification, with 72% probability:
  - $\boldsymbol{y} = [0 \quad 0 \quad 0 \quad 1 \quad 0]$
  - $\hat{\boldsymbol{y}} = [0.20 \quad 0.0 \quad 0.05 \quad 0.72 \quad 0.03]$
  - $L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -(0 \times \log 0.2 + 0 \times \log 0.0 + 0 \times \log 0.5 + 1 \times \log 0.72 + 0 \times \log 0.03)$
  - $L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -(\log 0.72) = 0.14267$

Cross-entropy for M>2

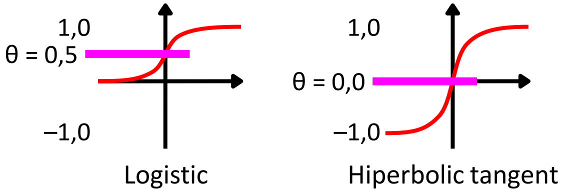
- 5 classes, **correct** classification, with 72% probability:
  - $\boldsymbol{y} = [0 \quad 0 \quad 0 \quad 1 \quad 0]$
  - $\widehat{\boldsymbol{y}} = [0.20 \quad 0.0 \quad 0.05 \quad 0.72 \quad 0.03]$
  - $L(\boldsymbol{y}, \widehat{\boldsymbol{y}}) = -(0 \times \log 0.2 + 0 \times \log 0.0 + 0 \times \log 0.5 + 1 \times \log 0.72 + 0 \times \log 0.03)$
  - $L(\boldsymbol{y}, \widehat{\boldsymbol{y}}) = -(\log 0.72) = 0.14267$
- 5 classes, **correct** classification, with 52% probability:
  - $\boldsymbol{y} = [0 \quad 0 \quad 0 \quad 1 \quad 0]$
  - $\widehat{\boldsymbol{y}} = [0.30 \quad 0.0 \quad 0.05 \quad 0.52 \quad 0.13]$
  - $L(\boldsymbol{y}, \widehat{\boldsymbol{y}}) = -(0 \times \log 0.3 + 0 \times \log 0.0 + 0 \times \log 0.5 + 1 \times \log 0.52 + 0 \times \log 0.13)$
  - $L(\boldsymbol{y}, \widehat{\boldsymbol{y}}) = -(\log 0.52) = 0.284$

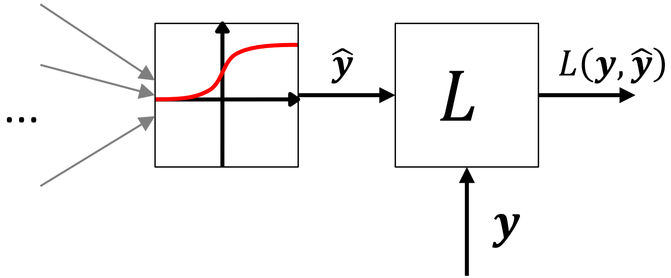
Cross-entropy for M>2

- 5 classes, **correct** classification, with 72% probability:
  - $\boldsymbol{y} = [0 \quad 0 \quad 0 \quad 1 \quad 0]$
  - $\hat{\boldsymbol{y}} = [0.20 \quad 0.0 \quad 0.05 \quad 0.72 \quad 0.03]$
  - $L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -(0 \times \log 0.2 + 0 \times \log 0.0 + 0 \times \log 0.5 + 1 \times \log 0.72 + 0 \times \log 0.03)$
  - $L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -(\log 0.72) = 0.14267$
- 5 classes, **correct** classification, with 52% probability:
  - $\boldsymbol{y} = [0 \quad 0 \quad 0 \quad 1 \quad 0]$
  - $\hat{\boldsymbol{y}} = [0.30 \quad 0.0 \quad 0.05 \quad 0.52 \quad 0.13]$
  - $L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -(0 \times \log 0.3 + 0 \times \log 0.0 + 0 \times \log 0.5 + 1 \times \log 0.52 + 0 \times \log 0.13)$
  - $L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -(\log 0.52) = 0.284$
- 5 classes, **incorrect** classification:
  - $\boldsymbol{y} = [0 \quad 0 \quad 0 \quad 1 \quad 0]$
  - $\hat{\boldsymbol{y}} = [0.60 \quad 0.0 \quad 0.07 \quad 0.30 \quad 0.03]$
  - $L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -(0 \times \log 0.6 + 0 \times \log 0.0 + 0 \times \log 0.07 + 1 \times \log 0.3 + 0 \times \log 0.03)$
  - $L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -(\log 0.3) = 0.5229$

Cross-entropy for M=2

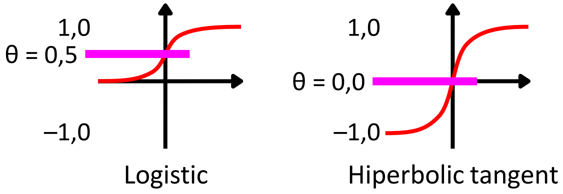
- 2 classes, correct classification:
  - $\boldsymbol{y} = [0]$
  - $\hat{\boldsymbol{y}} = [0.20]$
  - $L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -(0 \times \log 0.2 + (1 - 0) \times \log(1 - 0.2))$
  - $L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -(0 \times \log 0.2 + (1) \times \log(0.8)) = -(\log(0.8)) = 0.09691$



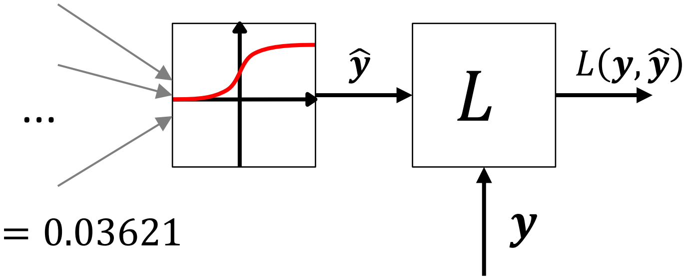


Cross-entropy for M=2

- 2 classes, correct classification:
  - $\boldsymbol{y} = [0]$
  - $\widehat{\boldsymbol{y}} = [0.20]$
  - $L(\boldsymbol{y}, \widehat{\boldsymbol{y}}) = -(0 \times \log 0.2 + (1-0) \times \log(1-0.2))$
  - $L(\boldsymbol{y}, \widehat{\boldsymbol{y}}) = -(0 \times \log 0.2 + (1) \times \log(0.8)) = -(\log(0.8)) = 0.09691$

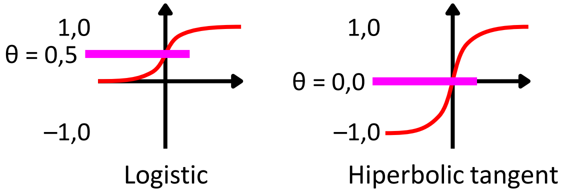


- 2 classes, correct classification:
  - $\boldsymbol{y} = [1]$
  - $\widehat{\boldsymbol{y}} = [0.92]$
  - $L(\boldsymbol{y}, \widehat{\boldsymbol{y}}) = -(1 \times \log 0.92 + (1-1) \times \log(1-0.92))$
  - $L(\boldsymbol{y}, \widehat{\boldsymbol{y}}) = -(1 \times \log 0.92 + (0) \times \log(0.08)) = -(\log(0.92)) = 0.03621$

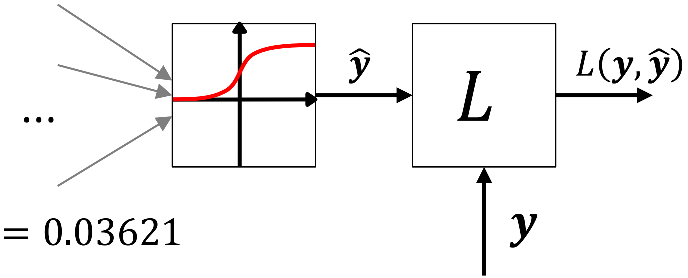


Cross-entropy for M=2

- 2 classes, correct classification:
  - $\boldsymbol{y} = [0]$
  - $\hat{\boldsymbol{y}} = [0.20]$
  - $L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -(0 \times \log 0.2 + (1-0) \times \log(1-0.2))$
  - $L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -(0 \times \log 0.2 + (1) \times \log(0.8)) = -(\log(0.8)) = 0.09691$



- 2 classes, correct classification:
  - $\boldsymbol{y} = [1]$
  - $\hat{\boldsymbol{y}} = [0.92]$
  - $L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -(1 \times \log 0.92 + (1-1) \times \log(1-0.92))$
  - $L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -(1 \times \log 0.92 + (0) \times \log(0.08)) = -(\log(0.92)) = 0.03621$

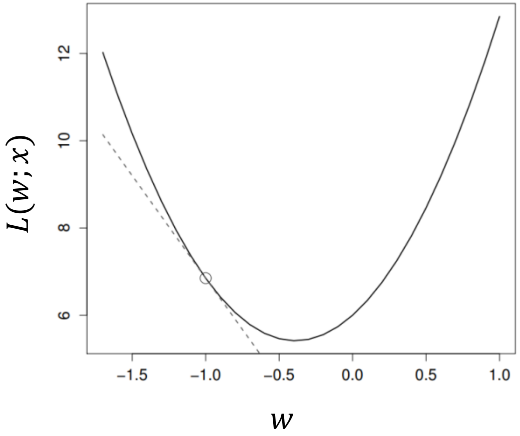


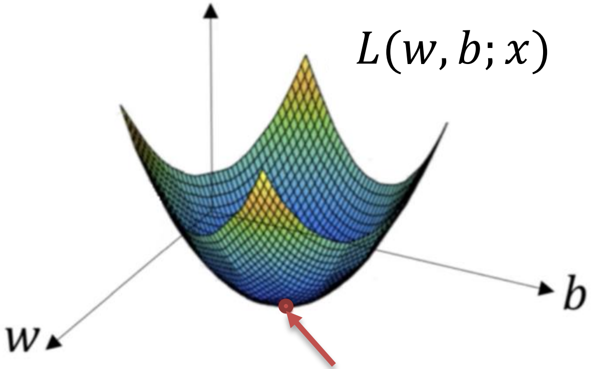
- 2 classes, incorrect classification:
  - $\boldsymbol{y} = [0]$
  - $\hat{\boldsymbol{y}} = [0.65]$
  - $L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -(0 \times \log 0.65 + (1-0) \times \log(1-0.65))$
  - $L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -(0 \times \log 0.65 + (1) \times \log(0.35)) = -(\log(0.35)) = 0.45593$

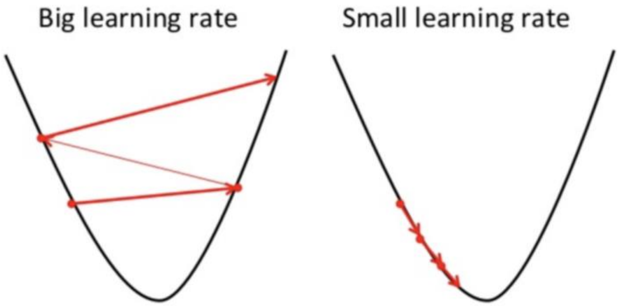
OPTIMIZERS

Optimizers

- Gradient descent (GD):
  - $W_{t+1} = W_t - \eta \sum_{j=1}^{N} \nabla L(W; x_j)$
  - N is the size of the training set





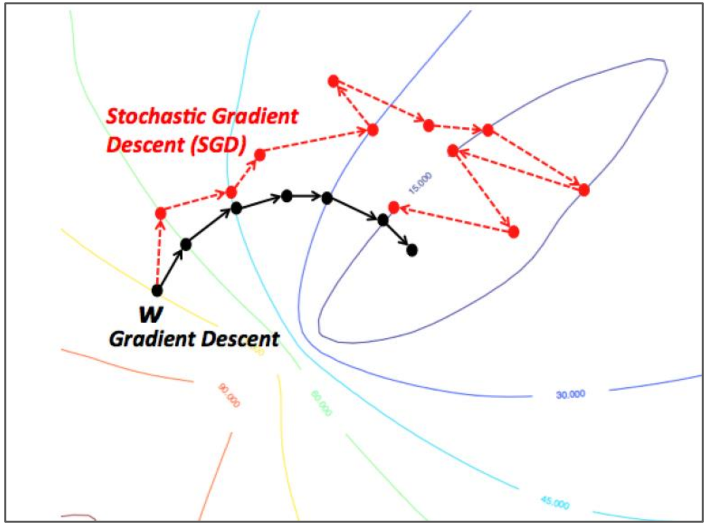


Moacir Ponti. http://conteudo.icmc.usp.br/pessoas/moacir/p17sibgrapi-tutorial/

Donges. Gradient Descent in Machine Learning: A Basic Introduction. https://builtin.com/data-science/gradient-descent

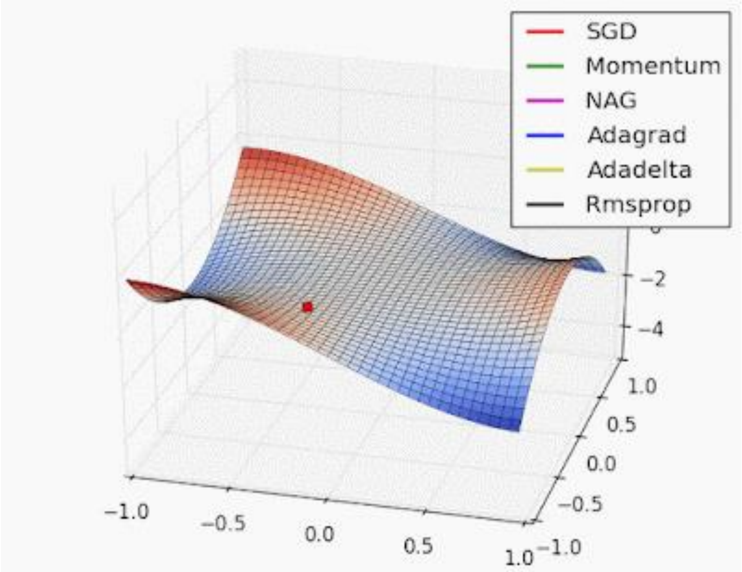
Optimizers

- Stochastic gradient descent (SGD):
  - $W_{t+1} = W_t - \eta \sum_{j=1}^{B} \nabla L(W; x_j^B)$
  - B is the size of the mini-batch.



https://wikidocs.net/3413

- SGD with momentum:
  - $W_{t+1} = W_t - \eta \sum_{j=1}^{B} \nabla L(W; x_j^B)$
    - B is the size of the 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

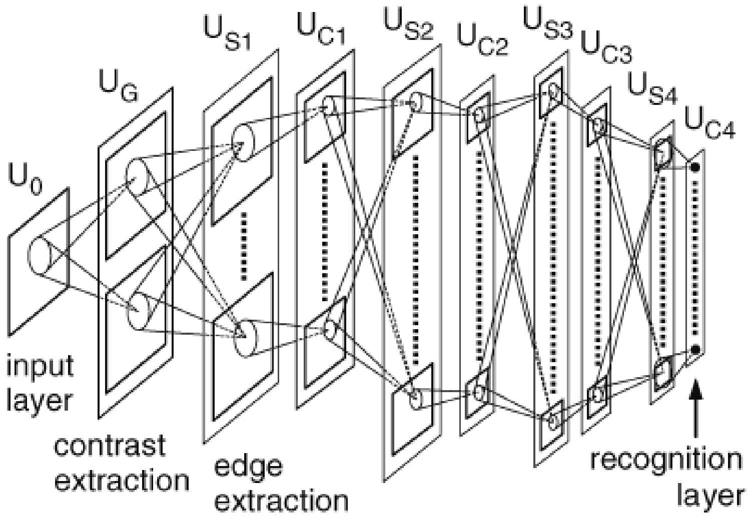
Optimizers

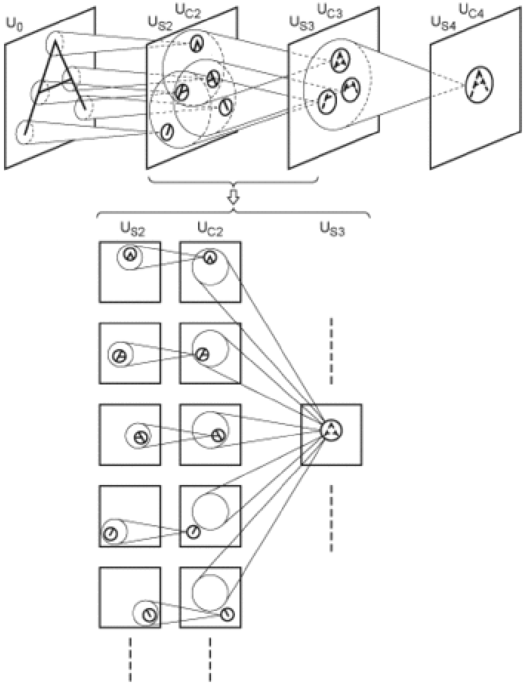
- Other optimizers:
  - AdaGrad - *Adaptive Gradient*
  - AdaDelta - *Adaptive learning rate*
  - RMSProp - *Root Mean Squared Propagation*
  - Adam - *Adaptive moment estimation*
  - ...

ARCHITECTURES

Architectures

- Neocognitron (1979)

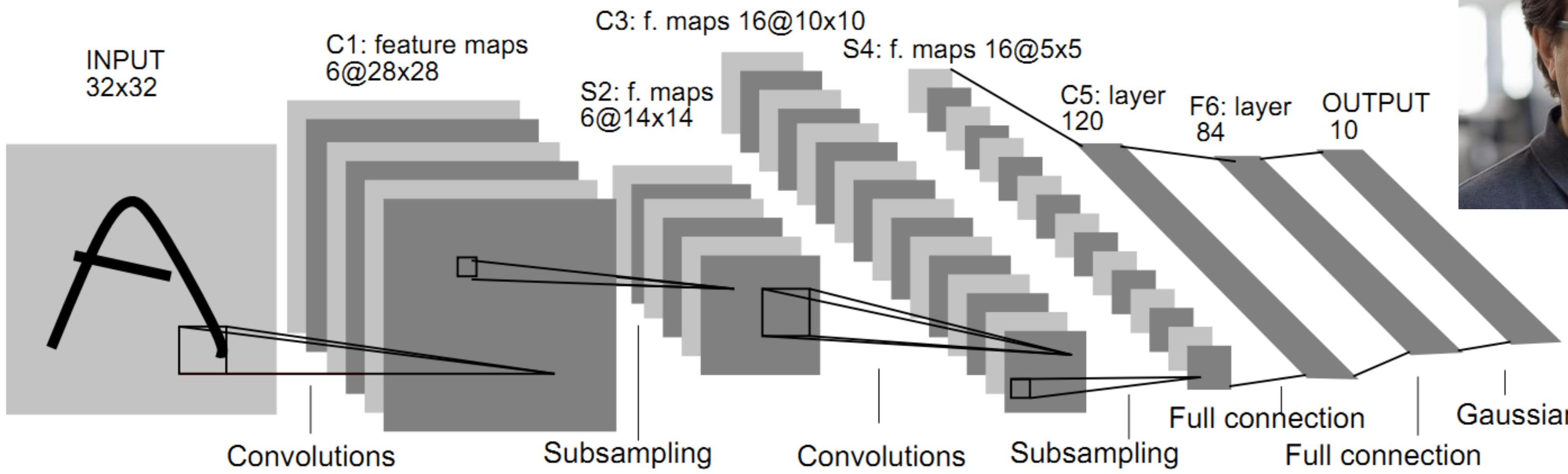




Kunihiko Fukushima

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)

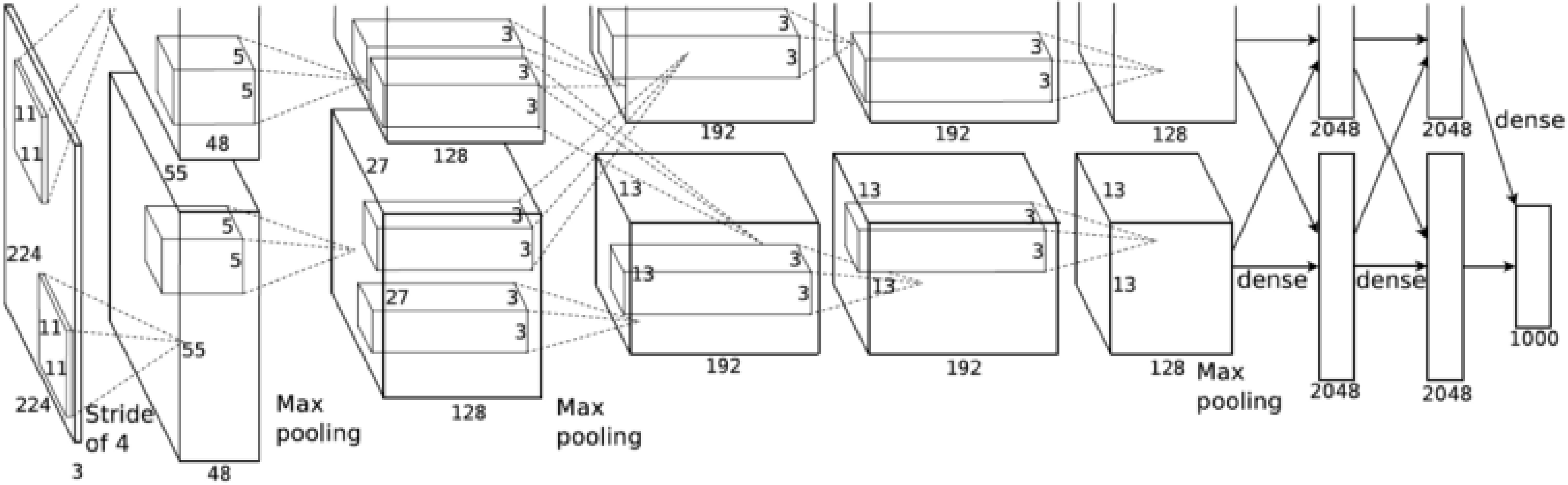
- LeNet-5 (1998)



Yann LeCun

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

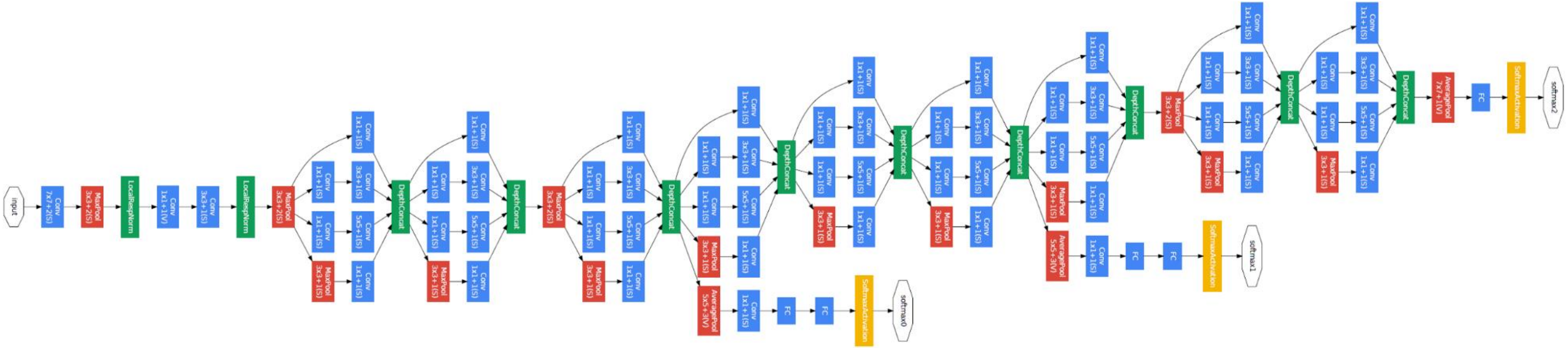
- AlexNet (2012)



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

Architectures

- Inception (GoogLeNet) (2015)

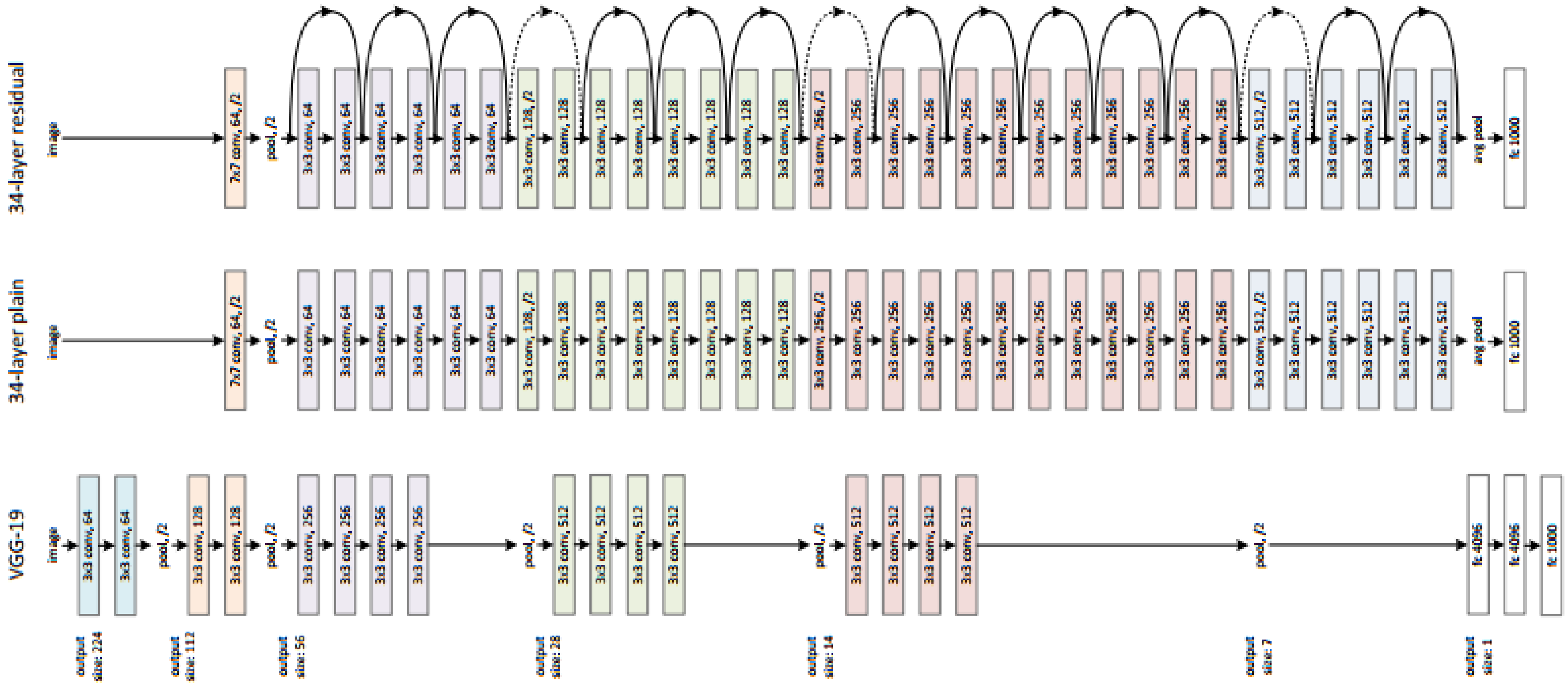


Szegedy, Christian (2015). "Going deeper with convolutions". CVPR2015.

Architectures

UFV
Universidade Federal de Viçosa

- VGG (2014) and ResNet (2015)



Simonyan e Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. 2014

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

- DenseNet (2017)

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

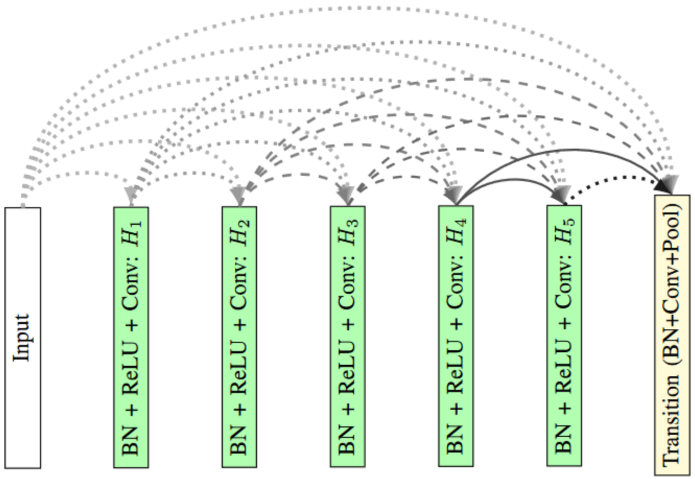


Figure 10. Illustration of a DenseBlock with 5 functions H_l and a Transition Layer.

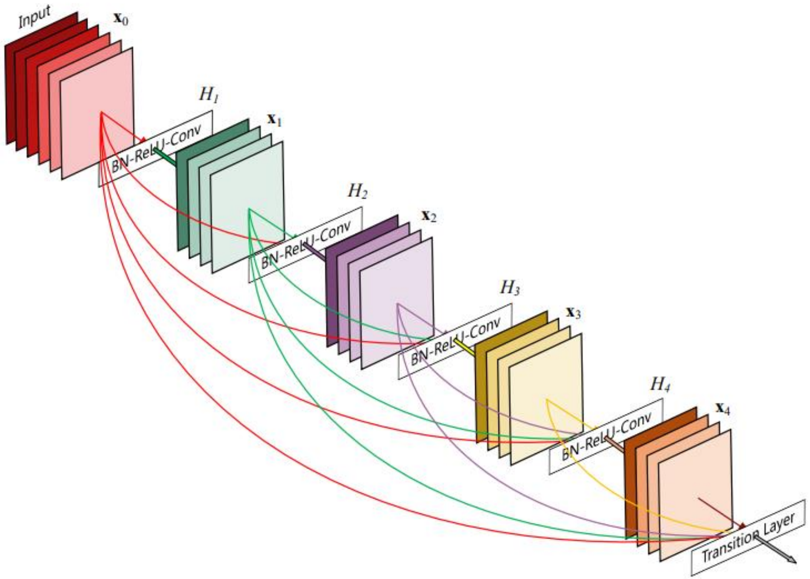
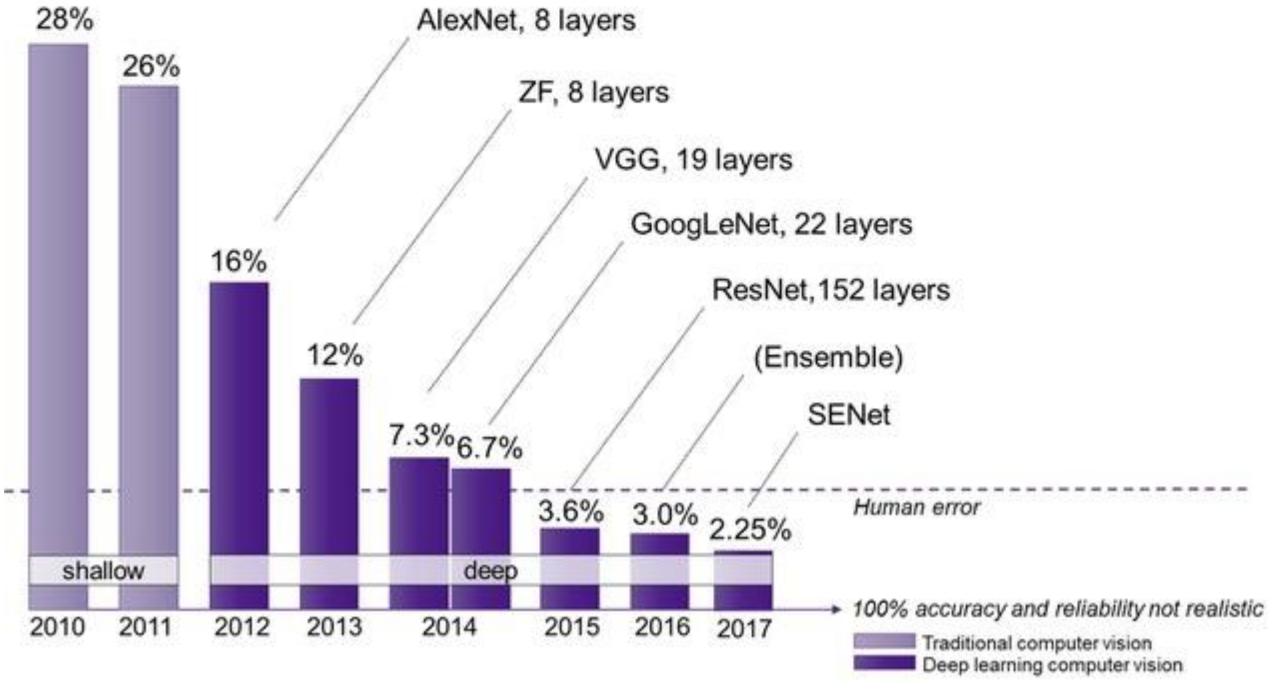


Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

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

Architectures

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



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

DEVELOPMENT AND LIBRARIES

Development and libraries

- Training CNNs has a high computational cost.
  - These are recommended to be trained using GPUs.
  - Google Colab provides access to GPUs (with some restrictions).

Development and libraries

- Top libraries for Deep Learning and Convolutional Neural Networks
  - PyTorch
    - https://pytorch.org/
  - Tensorflow
    - https://www.tensorflow.org/

Development and libraries

- **Anaconda Distribution:**
  - Python distribution with support for major libraries
  - https://www.anaconda.com/products/distribution
- **Google Colab:**
  - Cloud execution environment with GPUs
  - https://colab.research.google.com

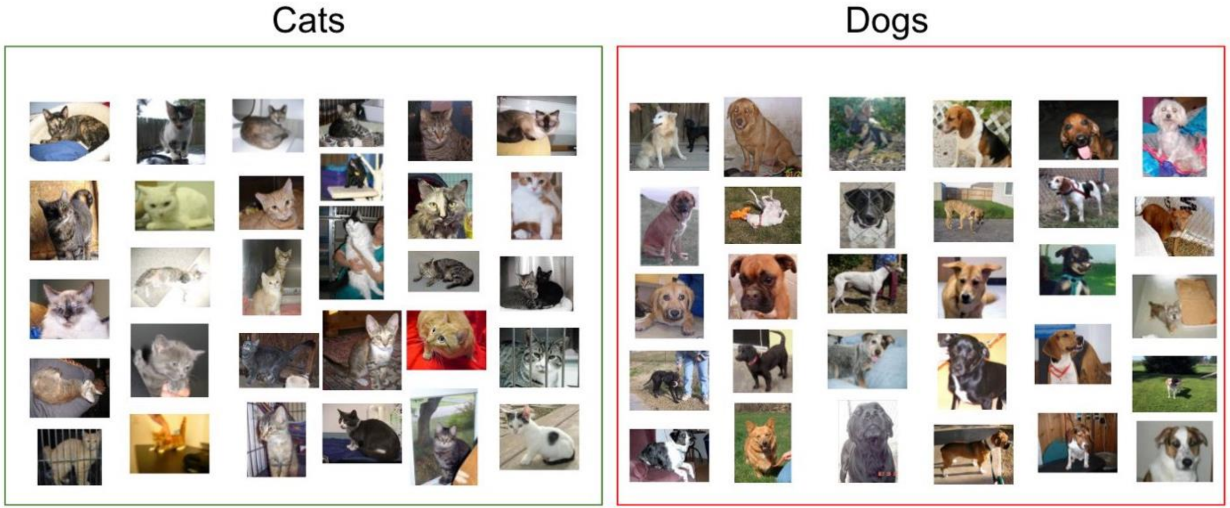
IMAGE DATASETS

UFV
Universidade Federal de Viçosa

- MNIST
  - http://yann.lecun.com/exdb/mnist/
  - 60,000 training images
  - 10,000 testing images
  - 28 x 28 pixels
  - Gray level

Image datasets

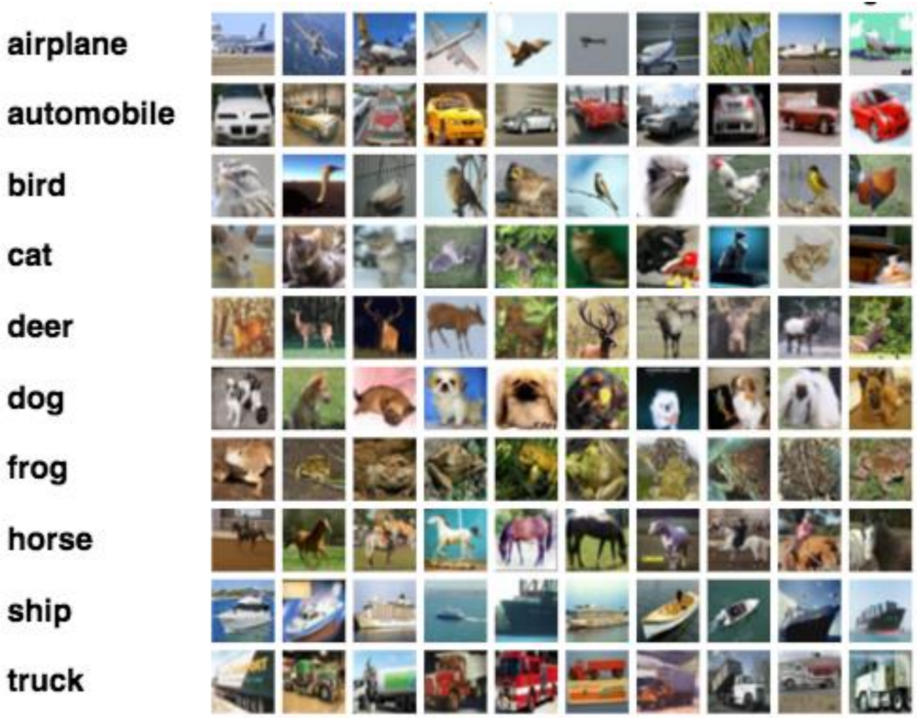
- **Cats vs. Dogs:**
  - https://www.kaggle.com/c/dogs-vs-cats
  - 25,000 training images
  - 12,500 testing images
  - 2 classes
  - Various sizes
  - RGB images



Sample of cats & dogs images from Kaggle Dataset

Image datasets

- **CIFAR10:**
  - https://www.cs.toronto.edu/~kriz/cifar.html
  - 50,000 training images
  - 10,000 testing images
  - 10 classes
  - 32 x 32 pixels
  - RGB



- **ImageNet:**
  - https://www.image-net.org/
  - ~1,000,000 images
  - 1,000 classes
  - RGB

Bibliography

- 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

Bibliography

Universidade Federal de Viçosa

- 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*
- Lecun, Y. et al. (1998). **Gradient-based learning applied to document recognition**. Proceedings of the IEEE. 86 (11): 2278–2324.
  - *10.1109/5.726791*
- Krizhevsky, Sutskever e Hinton. **ImageNet Classification with Deep Convolutional Neural Networks**. NeuripIPS 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.compbiomed.2019.103542

THE END