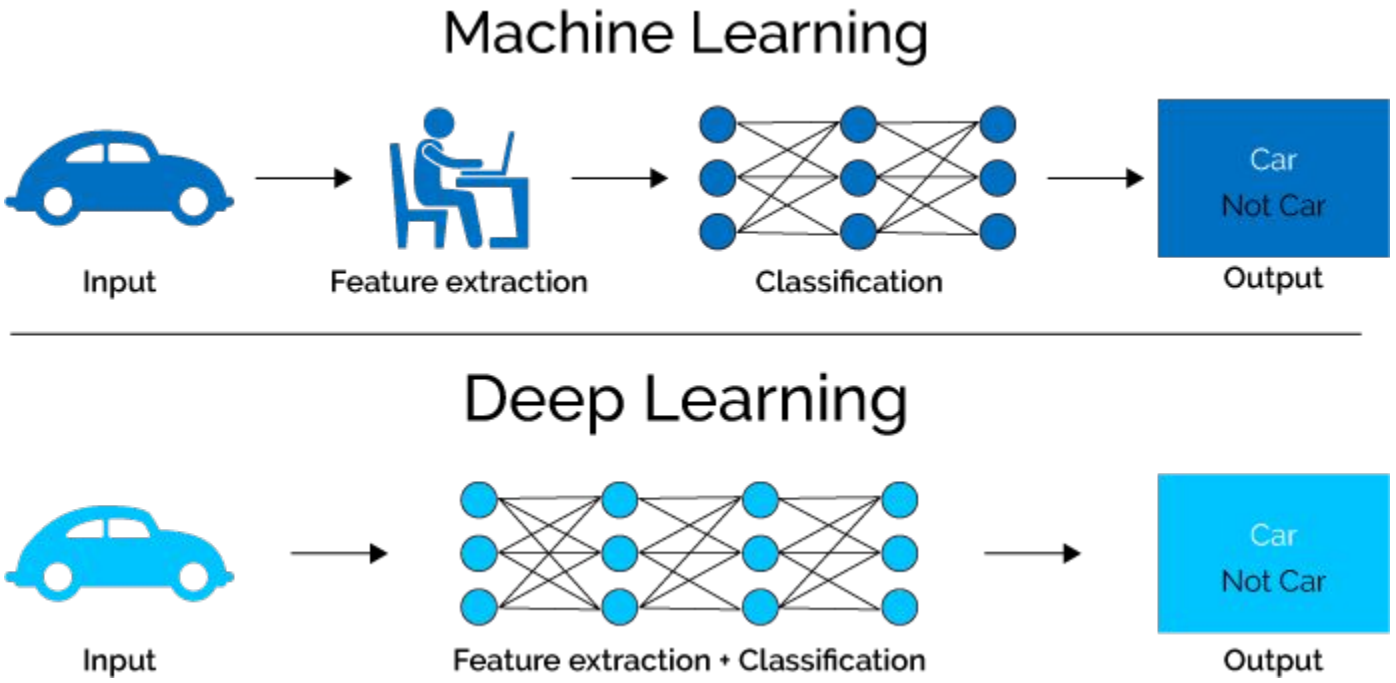
Redes Neurais Convolucionais

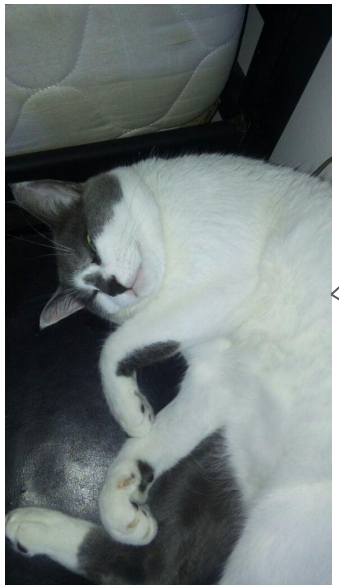
Thaís de Almeida Ratis Ramos

Baseado nas aulas de Thaís Gaudencio e Yuri Malheiros - UFPB

Machine learning x Deep Learning



Visão



- Gato
- Branco e cinza
- Dormindo

```
>>> a = cv2.imread('img.jpg
>>> a
array([[[ 87,  93,  82],
        [ 88,  94,  83],
        [ 89,  95,  84],
        ...,
        [ 13,  10,   5],
        [ 13,  10,   5],
        [ 13,  10,   5]],

       [[ 98, 104,  93],
        [101, 107,  96],
        [105, 111, 100],
        ...,
        [ 13,  10,   5],
        [ 13,  10,   5],
        [ 13,  10,   5]],

       [[113, 119, 108],
        [117, 123, 112],
        [122, 128, 117],
        ...,
```

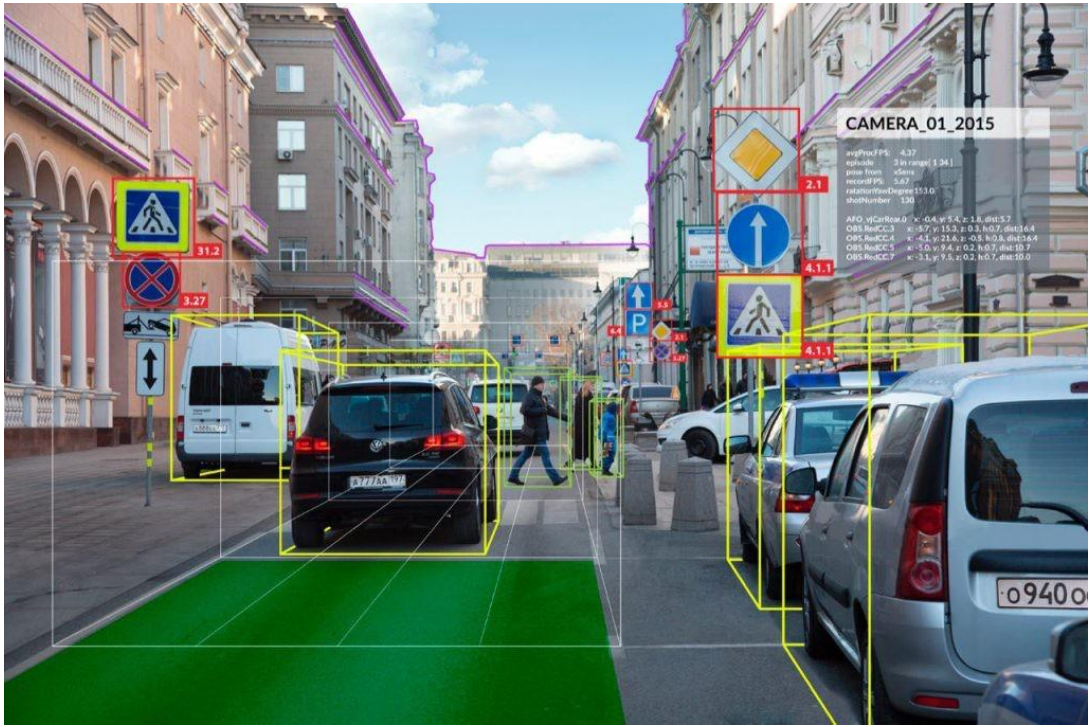
Visão Computacional

Visão computacional é o campo da ciência que estuda como os computadores podem enxergar e entender o conteúdo de imagens e vídeos

Isto inclui adquirir, processar e analisar imagens ou vídeos

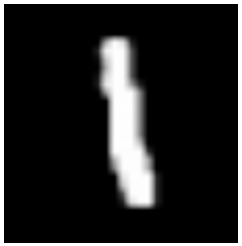
Problemas ligados à visão computacional muitas vezes são resolvidos facilmente por pessoas, mas não pelas máquinas

Visão Computacional



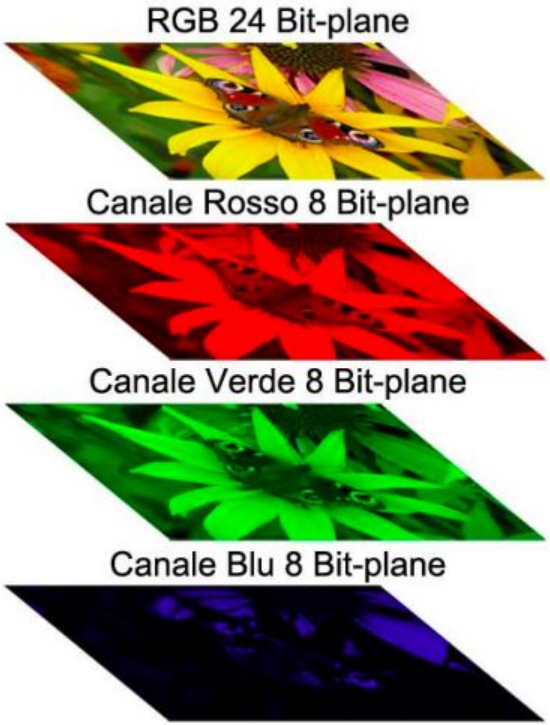
Visão Computacional

Uma imagem é representada como uma matriz



0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0,6	0,8	0	0	0	0	0	0
0	0	0	0	0	0	0,7	1	0	0	0	0	0	0
0	0	0	0	0	0	0,7	1	0	0	0	0	0	0
0	0	0	0	0	0	0,5	1	0,4	0	0	0	0	0
0	0	0	0	0	0	0	1	0,4	0	0	0	0	0
0	0	0	0	0	0	0	1	0,4	0	0	0	0	0
0	0	0	0	0	0	0	1	0,7	0	0	0	0	0
0	0	0	0	0	0	0	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0,9	1	0,1	0	0	0	0
0	0	0	0	0	0	0	0,3	1	0,1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

Imagens RGB



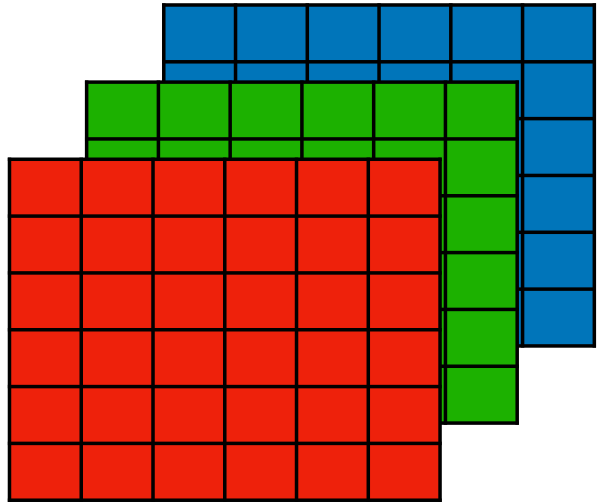
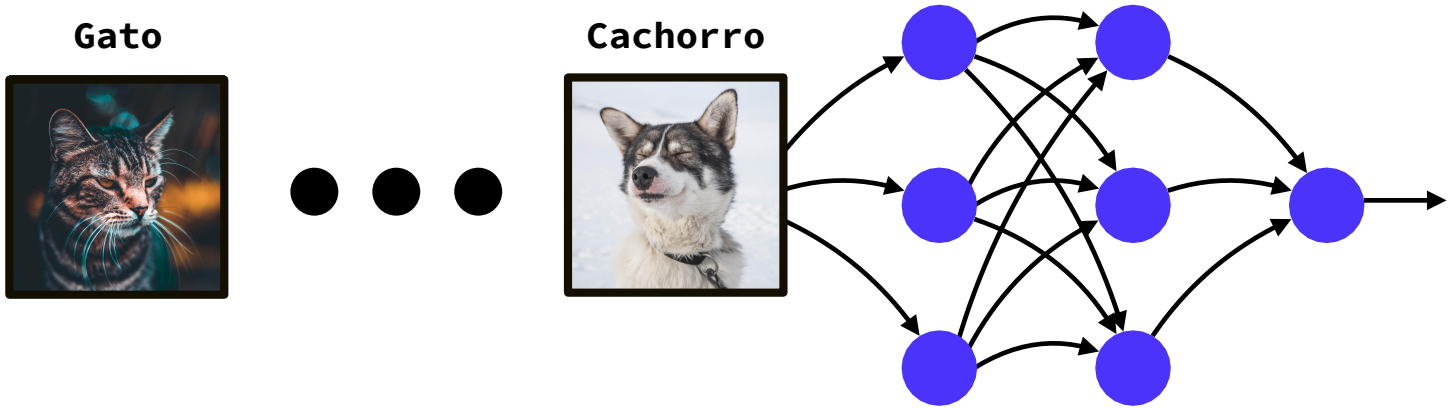


Imagem A x L x 3
Imagens A x L x C
C: no. de canais, ou profundidade

Classificação de imagens

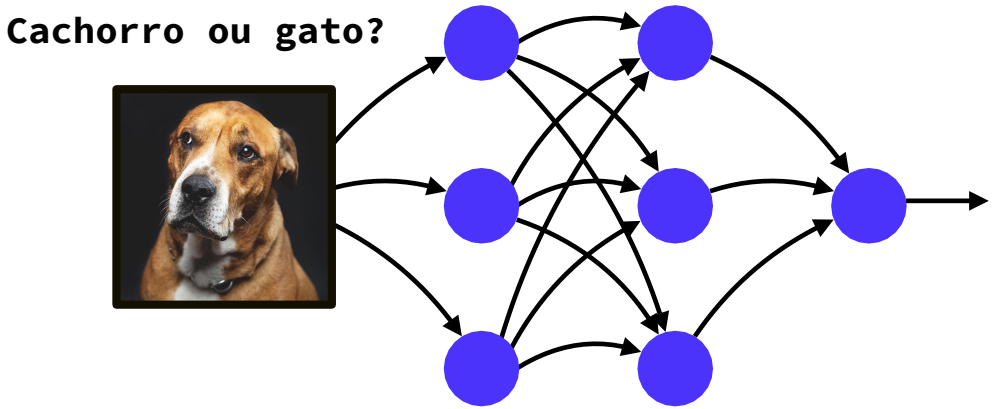
Classificação de imagens

Treinamos uma rede neural com várias imagens de cachorros e gatos



Classificação de imagens

Para que possamos classificar novas entradas



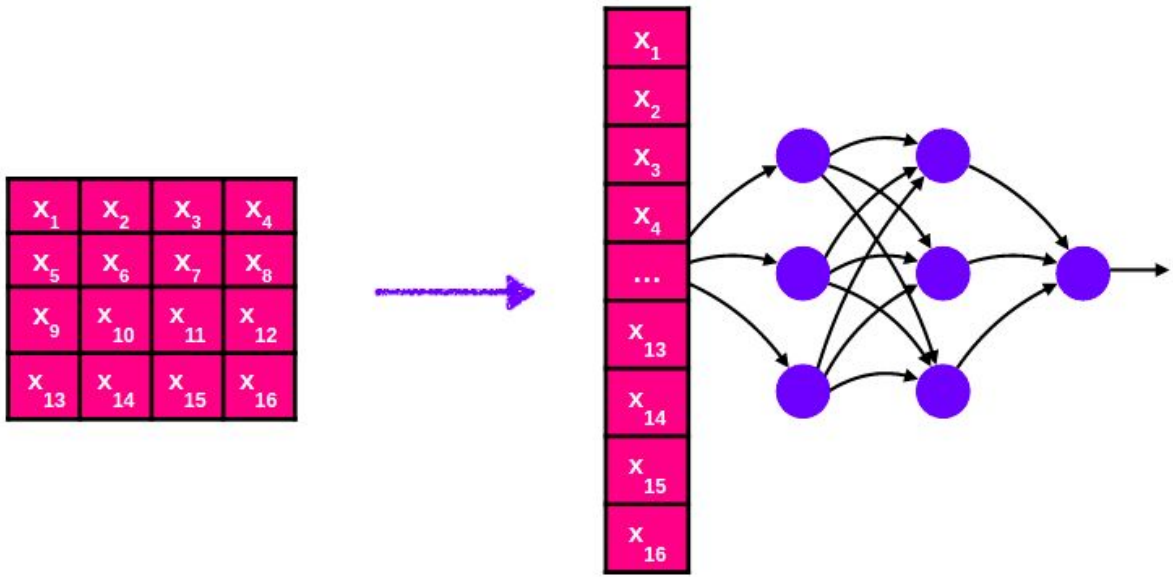
Classificação de imagens

Como passar uma imagem para uma rede neural?

A rede recebe entradas numéricas, ela não recebe uma matriz

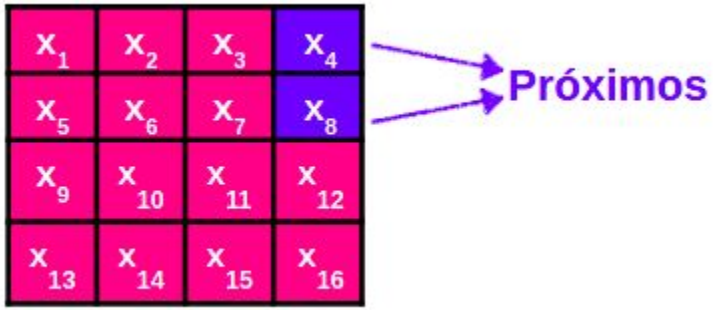
Classificação de imagens

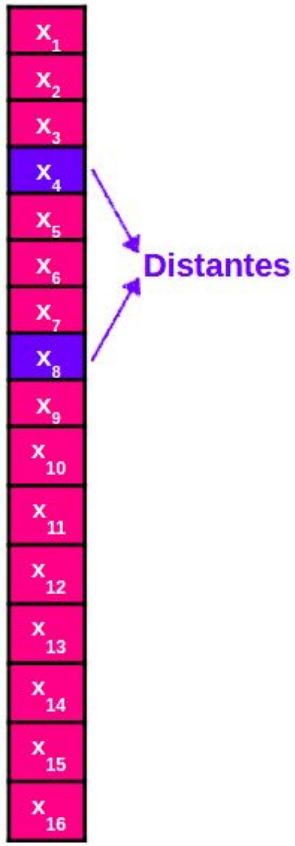
Podemos passar cada valor da matriz como uma entrada



Classificação de imagens

Uma limitação dessa abordagem é que perdemos a noção espacial da imagem





Classificação de imagens

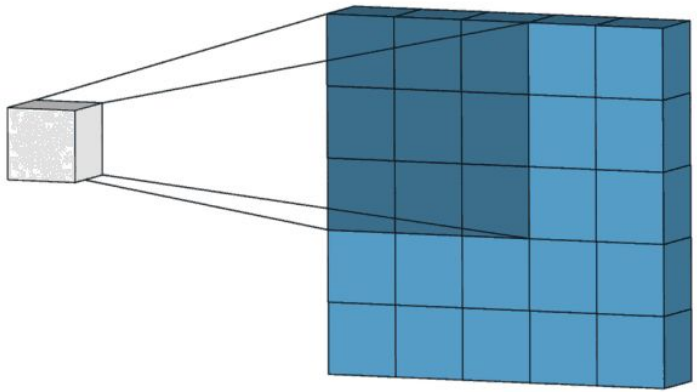
Outra limitação é que o número de entradas e pesos será muito grande

Uma imagem colorida 1000x1000 tem 3 milhões de valores (A x L x 3)

Se a primeira camada intermediária possuir 10 neurônios, teremos 30 milhões de pesos só na primeira camada!

Classificação de imagens - Visão Computacional

- Uma operação linear para reconhecer padrões
- Operação do filtro/kernel sobre a imagem/sinal
- Disposição espacial dos dados se transforma em informação

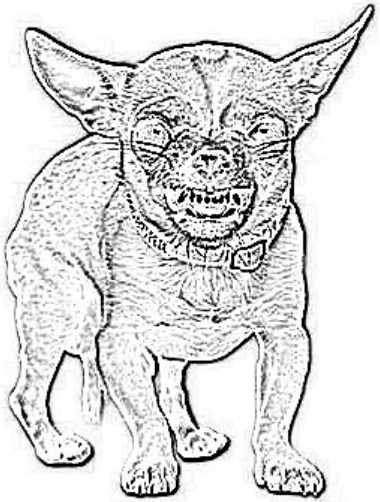


Classificação de imagens

Podemos processar previamente as imagens e depois passar para rede neural:

➔ Detectar arestas
➔ Detectar cantos
➔ Detectar regiões

Detecção de arestas

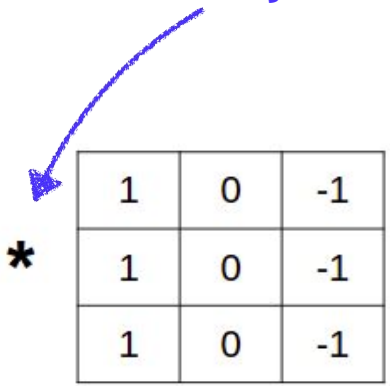


Detecção de arestas

Imagem

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

Convolução



Filtro

* =

Operação com filtros

O mesmo filtro é “aplicado” sobre o campo receptivo (CR) que desliza sobre a imagem

A cada iteração, são somados os produtos dos pares (sum(CRxfiltro)) gerando um novo valor para a matriz resultante

Detecção de arestas

Campo receptivo

Imagem

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

Convolução

*

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

Filtro

=

Detecção de arestas - Convolução 2D

3x1

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

*

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

=

Detecção de arestas - Convolução 2D

3x1 + 1x1

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

*

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

=

Detecção de arestas - Convolução 2D

3x1 + 1x1 + 2x1

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

*

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

=

Detecção de arestas - Convolução 2D

$$3x1 + 1x1 + 2x1 + 0x0$$

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

*

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

=

Detecção de arestas - Convolução 2D

3x1 + 1x1 + 2x1 + 0x0 + 5x0

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

*

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

=

Detecção de arestas - Convolução 2D

3x1 + 1x1 + 2x1 + 0x0 + 5x0 + 7x0

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

*

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

=

Detecção de arestas - Convolução 2D

3x1 + 1x1 + 2x1 + 0x0 + 5x0 + 7x0 + 1x-1

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

*

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

=

Detecção de arestas - Convolução 2D

3x1 + 1x1 + 2x1 + 0x0 + 5x0 + 7x0 + 1x-1 + 8x-1

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

*

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

=

Detecção de arestas - Convolução 2D

3x1 + 1x1 + 2x1 + 0x0 + 5x0 + 7x0 + 1x-1 + 8x-1 + 2x-1

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

*

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

=

Detecção de arestas - Convolução 2D

3x1 + 1x1 + 2x1 + 0x0 + 5x0 + 7x0 + 1x-1 + 8x-1 + 2x-1 = -5

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

*

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

=

-5			

Detecção de arestas - Convolução 2D

0x1 + 5x1 + 7x1 + 1x0 + 8x0 + 2x0 + 2x-1 + 9x-1 + 5x-1 = -4

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

*

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

=

-5	-4		

Detecção de arestas - Convolução 2D

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

*

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

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

Detecção de arestas - Convolução 2D

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

*

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

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

Detecção de arestas - Convolução 2D

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

*

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

=

Detecção de arestas - Convolução 2D

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

*

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

=

Detecção de arestas - Convolução 2D

10x1 + 10x1 + 10x1 + 10x0 + 10x0 + 10x0 + 10x-1 + 10x-1 + 10x-1 = 0

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

*

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

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

Detecção de arestas - Convolução 2D

10x1 + 10x1 + 10x1 + 10x0 + 10x0 + 10x0 + 10x0 + 10x0 + 10x0 = 30

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

*

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

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

Detecção de arestas - Convolução 2D

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

*

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

=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

Detecção de arestas

Filtros diferentes detectam características diferentes:

Arestas verticais

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

Arestas horizontais

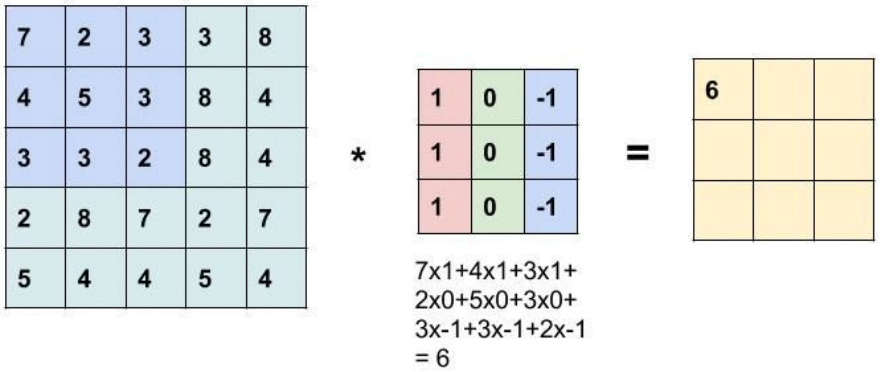
1	1	1
0	0	0
-1	-1	-1

Redes neurais convolucionais

$$D_{(0,0)} = (S_{(0,0)} * W_{(0,0)}) + (S_{(1,0)} * W_{(1,0)}) + (S_{(2,0)} * W_{(2,0)}) +$$

$$(S_{(0,1)} * W_{(0,1)}) + (S_{(1,1)} * W_{(1,1)}) + (S_{(2,1)} * W_{(2,1)}) +$$

$$(S_{(0,2)} * W_{(0,2)}) + (S_{(1,2)} * W_{(1,2)}) + (S_{(2,2)} * W_{(2,2)})$$

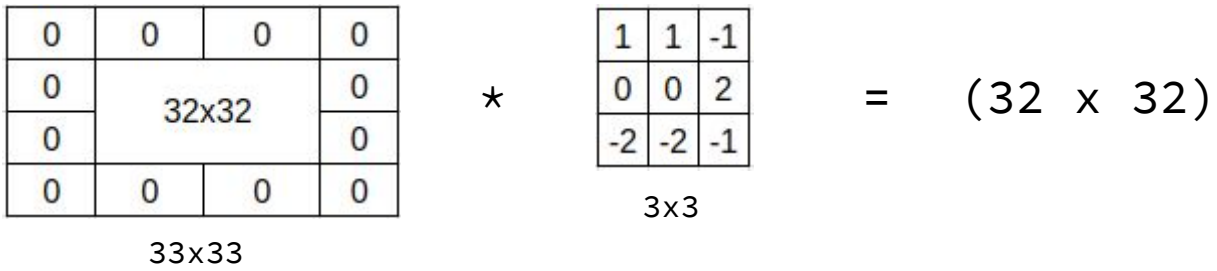


Tamanho Final = (Img - W) + 1 → (5 - 3) + 1 = 3

Filtro

Padding

Padding = 1; Imagem (32 x 32) -> (33 x 33)

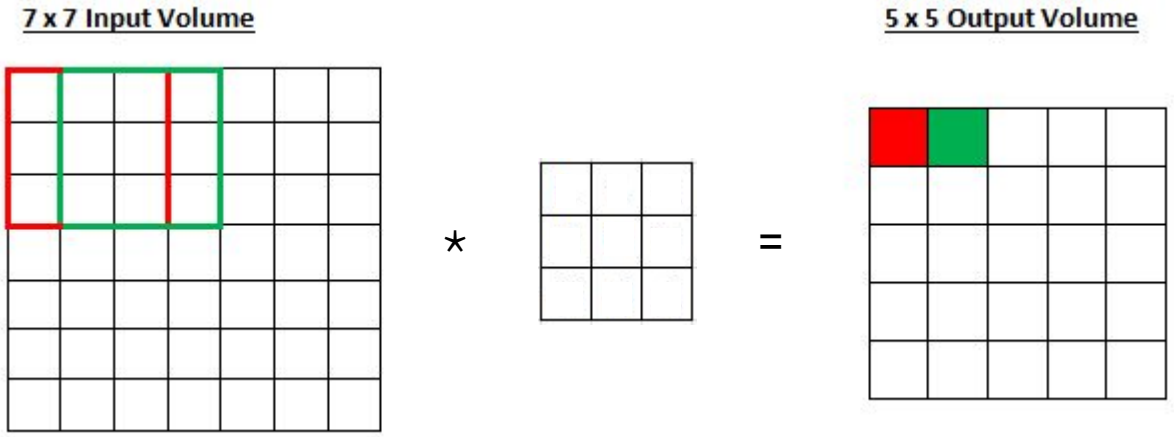


Tamanho Final = (Img - W + 2*P) + 1

Tamanho Final = (32 - 3 + 2*1) + 1 = 32

Stride

Stride = 1; Filtro 3x3

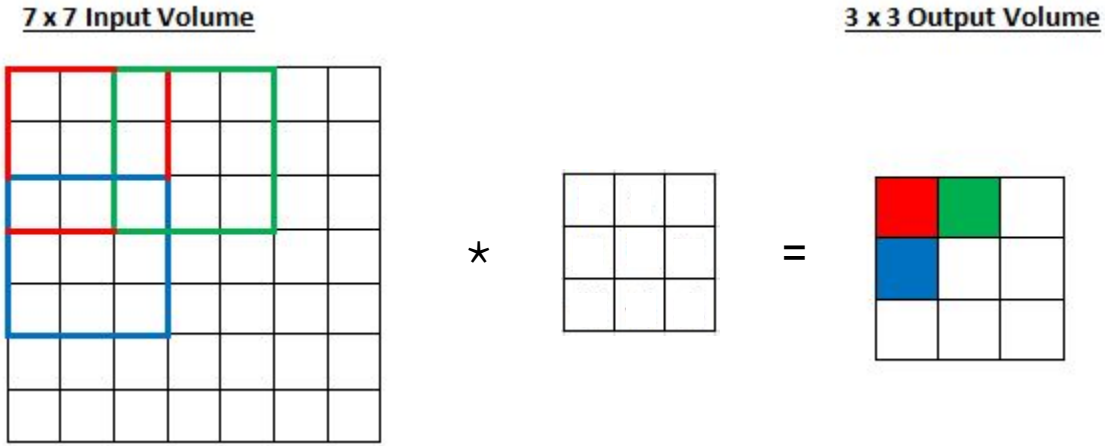


Tamanho Final = [(Img - W + 2*P) / S] + 1

Tamanho Final = [(7 - 3 + 2*0) / 1] + 1 = 5

Stride

Stride = 2; Filtro 3x3



Tamanho Final = [(Img - W + 2*P) / S] + 1

Tamanho Final = [(7 - 3 + 2*0) / 2] + 1 = 3

Redes neurais convolucionais

Podemos tentar criar filtros manualmente escolhendo valores

Quais filtros criar?
Como saber se os valores escolhidos são bons?

Não sabemos previamente!

Redes neurais convolucionais

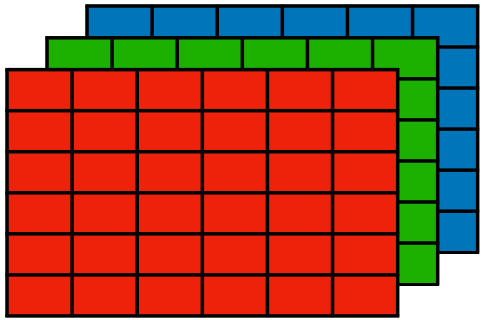
3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

*

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

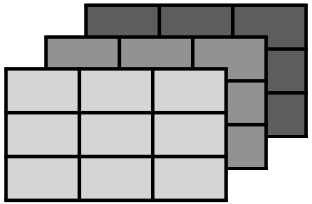
Vamos tratar os valores dos filtros como pesos que serão aprendidos pela rede neural através do backpropagation

Convolução 3D



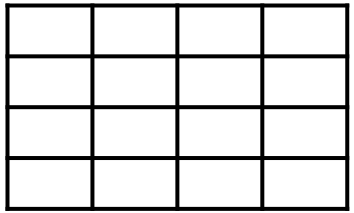
6 x 6 x 3

*



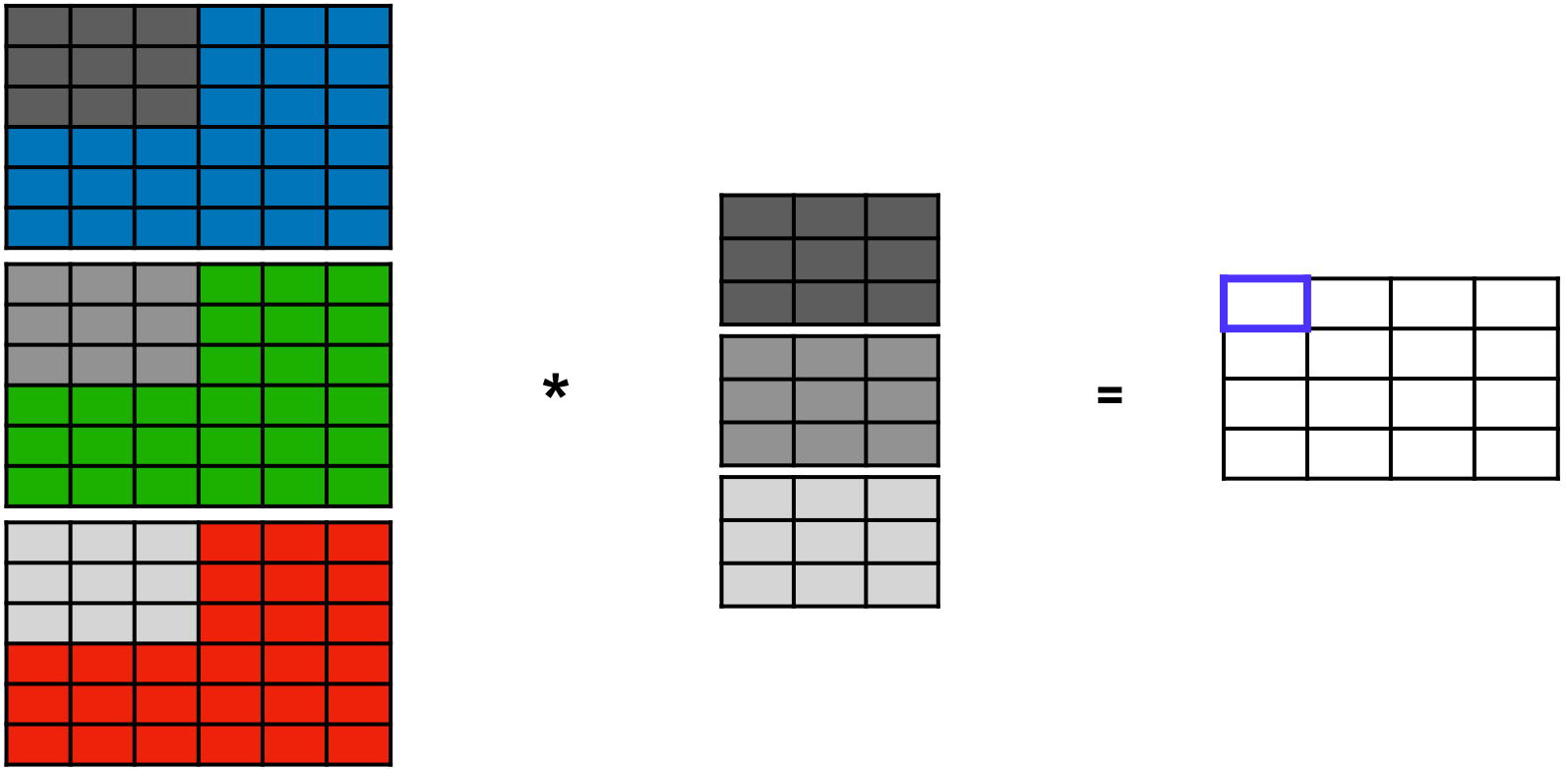
3 x 3 x 3

=

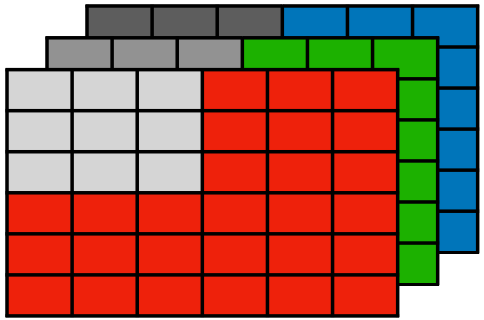


4 x 4

Convolução 3D

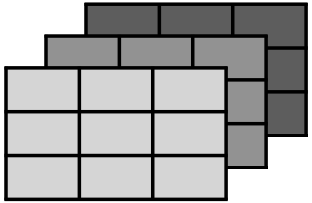


Convolução 3D



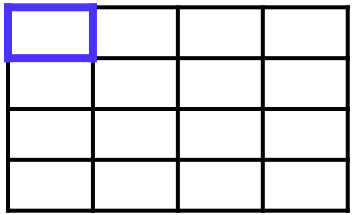
6 x 6 x 3

*



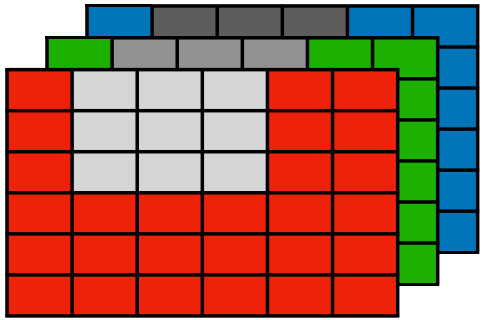
3 x 3 x 3

=



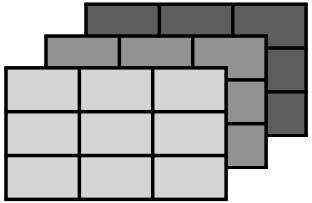
4 x 4

Convolução 3D



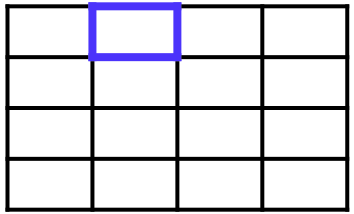
6 x 6 x 3

*



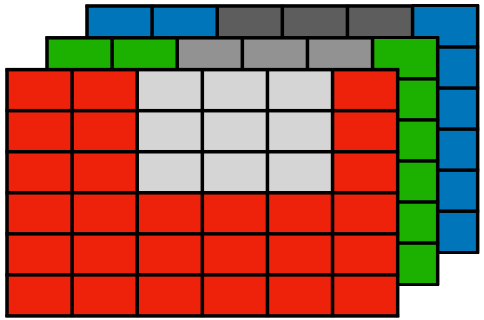
3 x 3 x 3

=



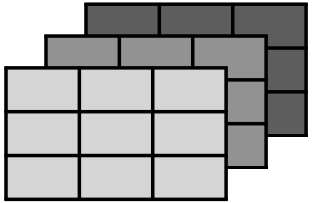
4 x 4

Convolução 3D



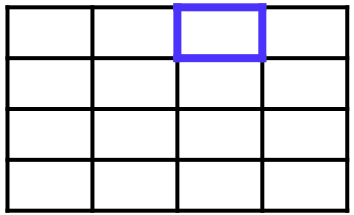
6 x 6 x 3

*



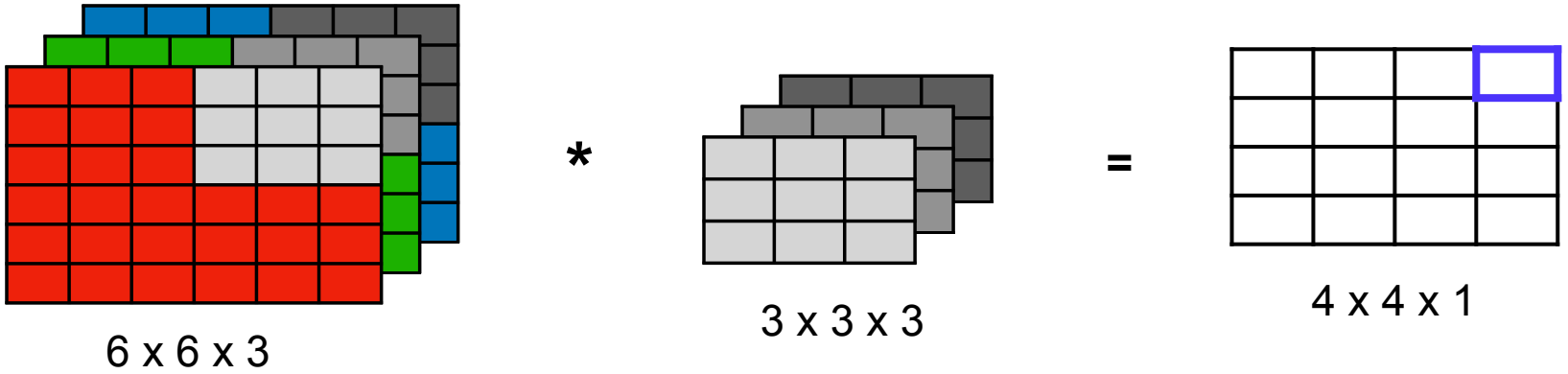
3 x 3 x 3

=



4 x 4

Convolução 3D



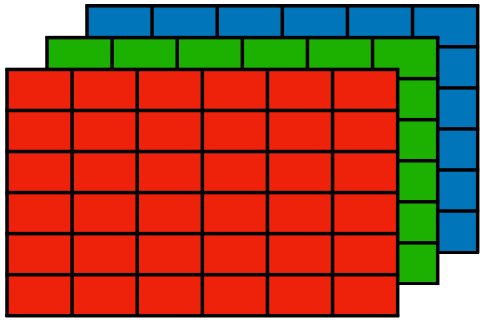
Tamanho Final 3D: $Imagem_{(x,y,c)}$ o $N*Filtros_{(x',y',c)}$ = $Saída_{(x",y",N)}$

Importante: x" e y" são calculados como se fosse 2D: (P=0; S=1)

TF = [(Img - W + 2P) / S] + 1 ⟹ [(6 - 3 + 0) / 1] + 1 = 4

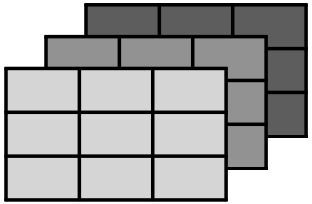
Tamanho Final 3D: $Img_{(6,\ 6,\ 3)}$ o $1*W_{(3,\ 3,\ 3)}$ = $Saída_{(4,\ 4,\ 1)}$

Convolução 3D



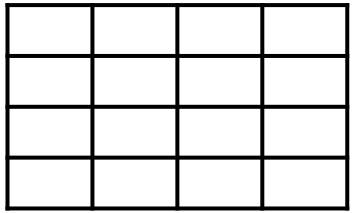
6 x 6 x 3

*



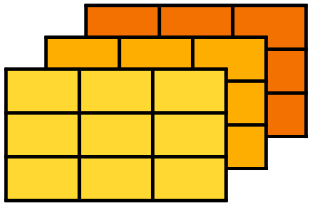
3 x 3 x 3

=



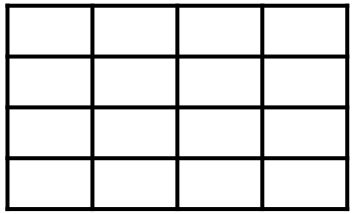
4 x 4 x 1

*



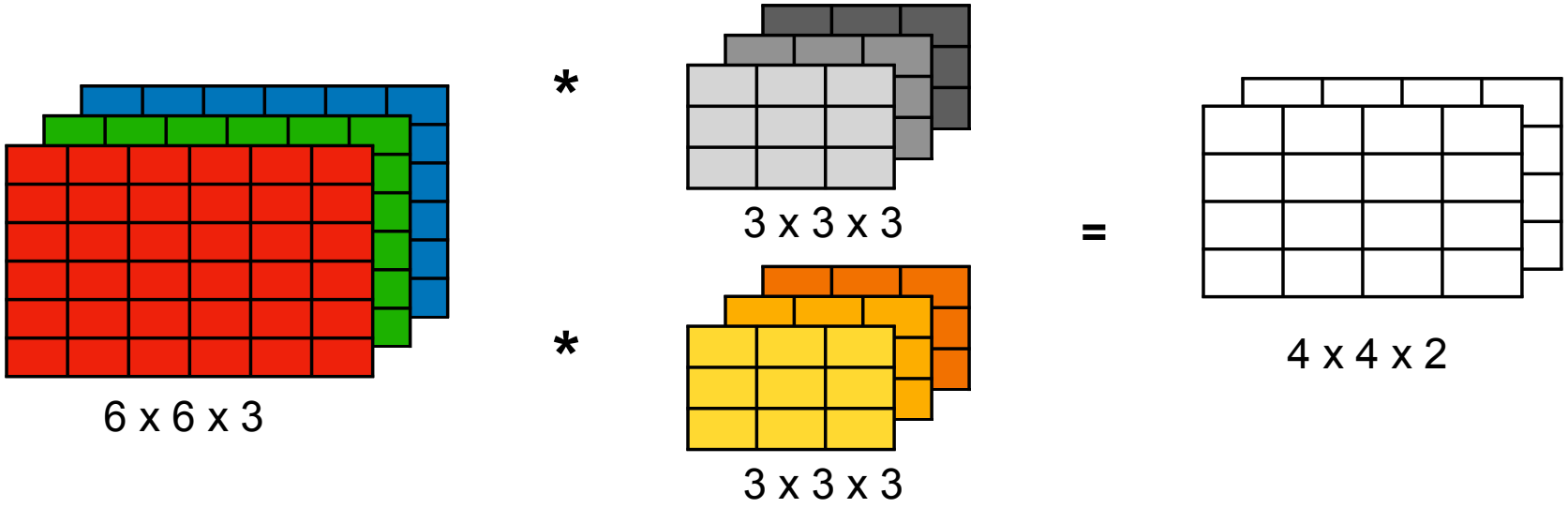
3 x 3 x 3

=



4 x 4 x 1

Convolução 3D



Tamanho Final 3D: $\text{Imagem}_{(x,y,c)}$ o $N*\text{Filtros}_{(x',y',c)}$ = $\text{Saída}_{(x'',y'',N)}$

x" e y" são definidos pelo mesmo cálculo anterior cujo resultado foi 4

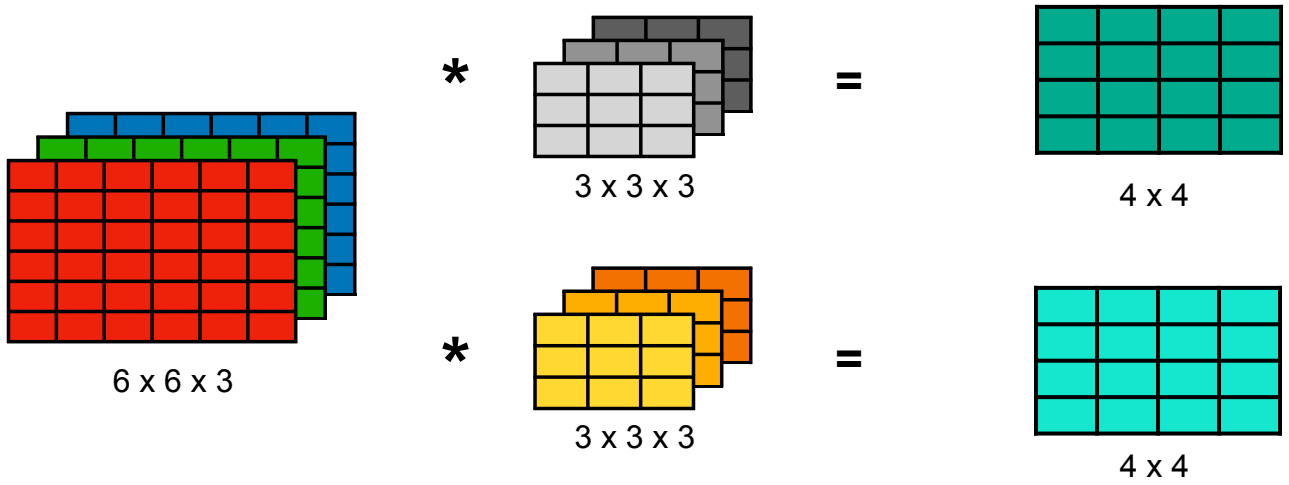
Tamanho Final 3D: $\text{Img}_{(6,\ 6,\ 3)}$ o $2*W_{(3,\ 3,\ 3)}$ = $\text{Saída}_{(4,\ 4,\ 2)}$

Redes neurais convolucionais

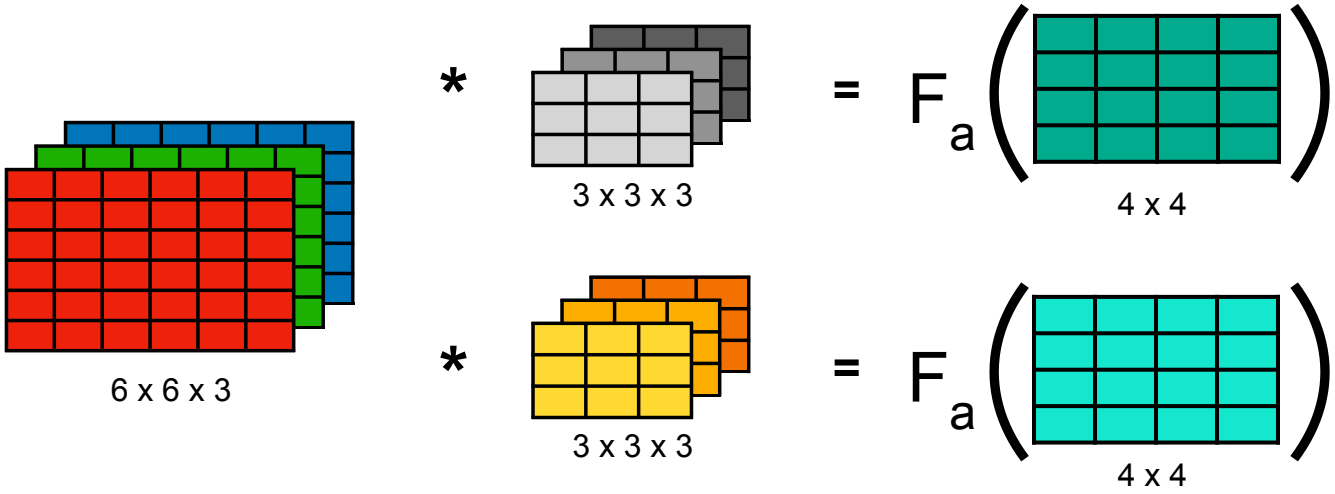
Temos todas as peças para entender e construir uma rede neural convolucional

Iniciaremos com uma camada

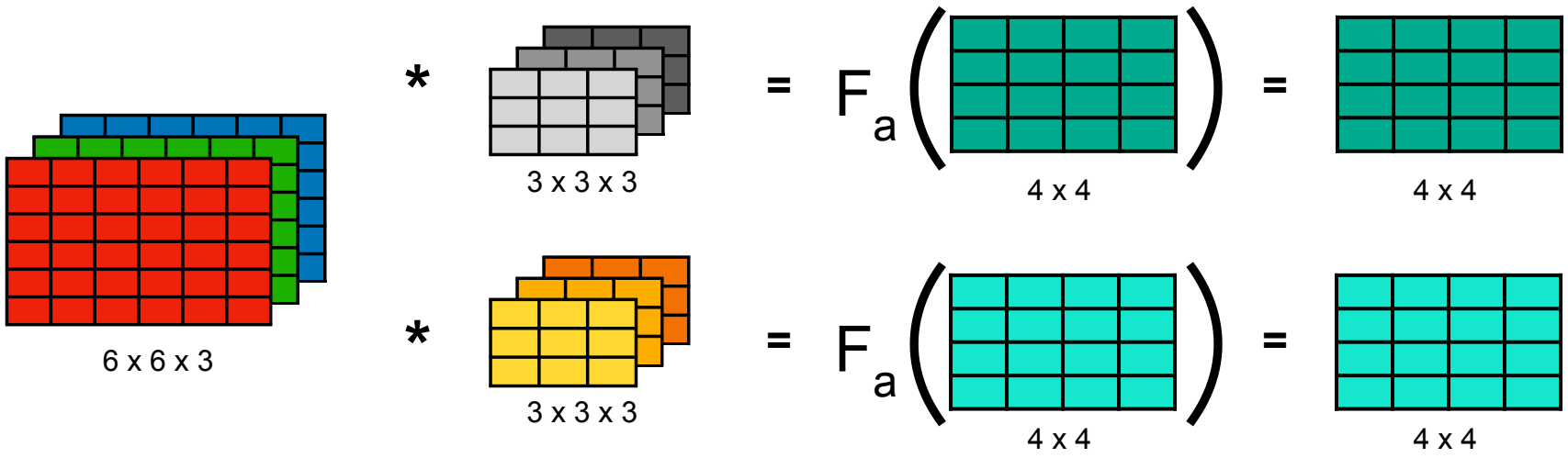
Redes neurais convolucionais



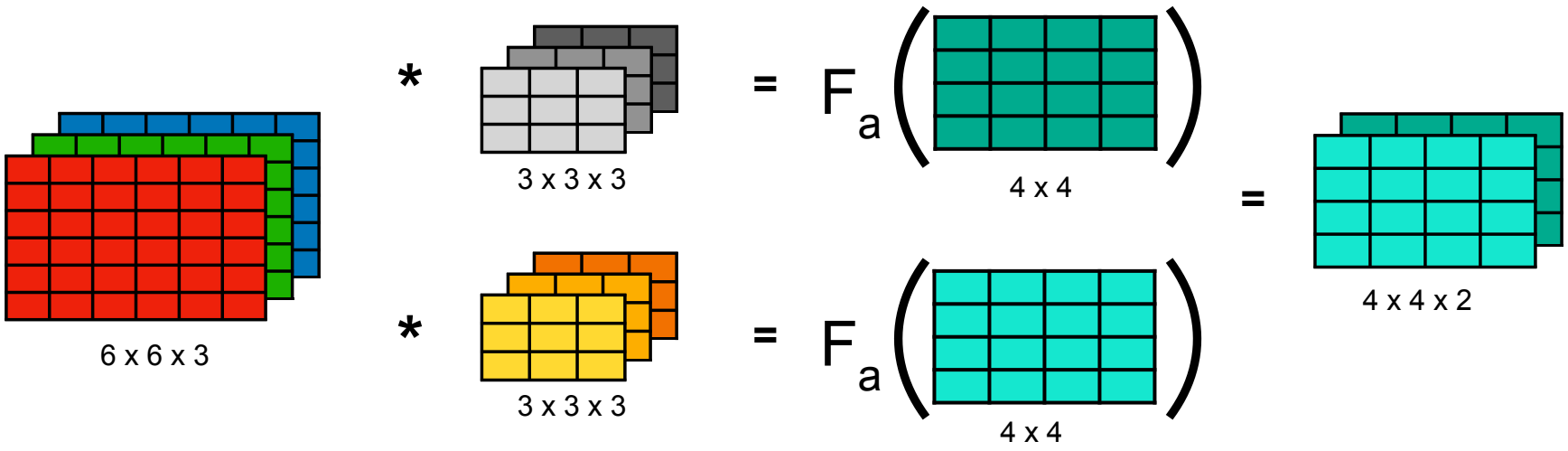
Redes neurais convolucionais



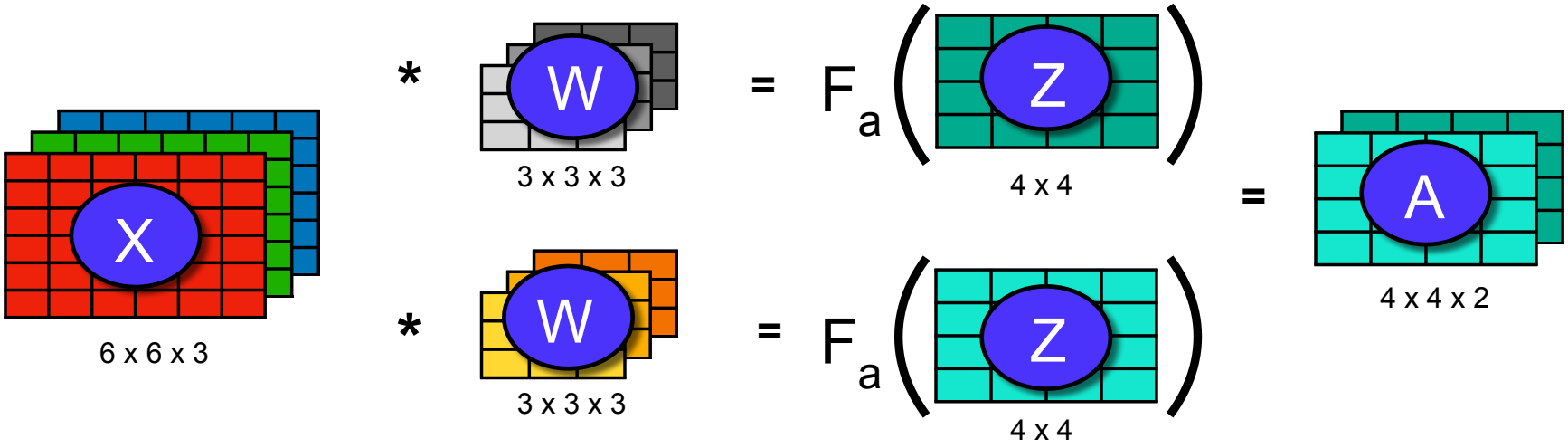
Redes neurais convolucionais



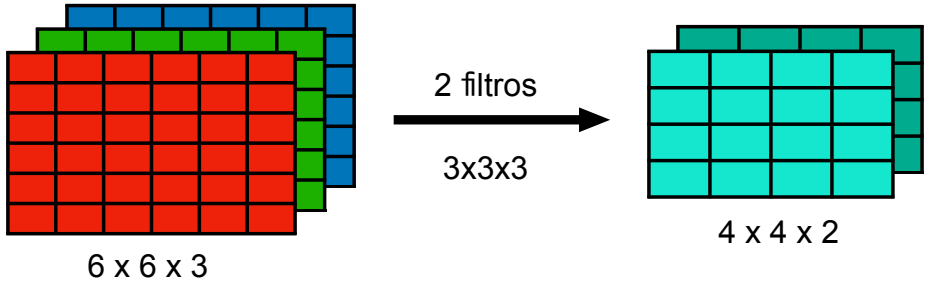
Redes neurais convolucionais



Redes neurais convolucionais

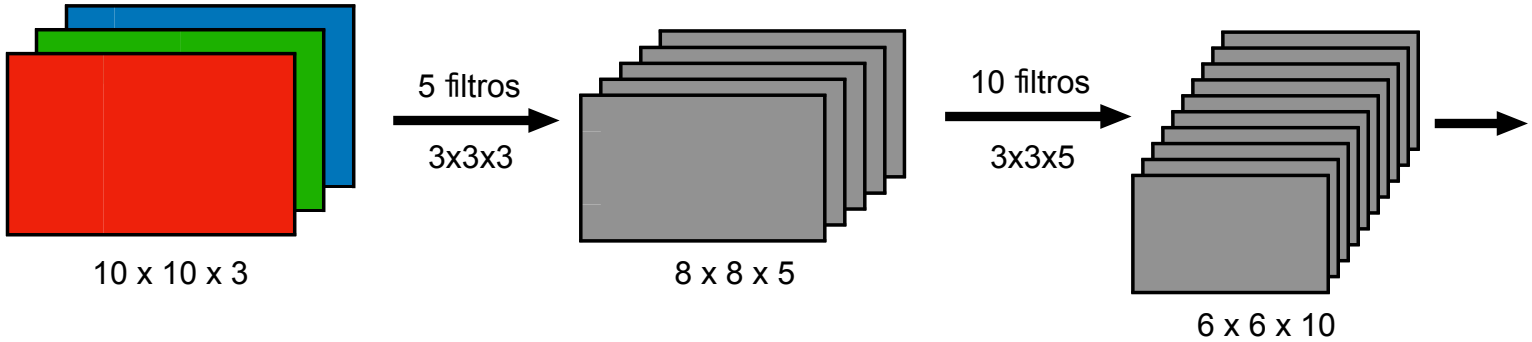


Redes neurais convolucionais



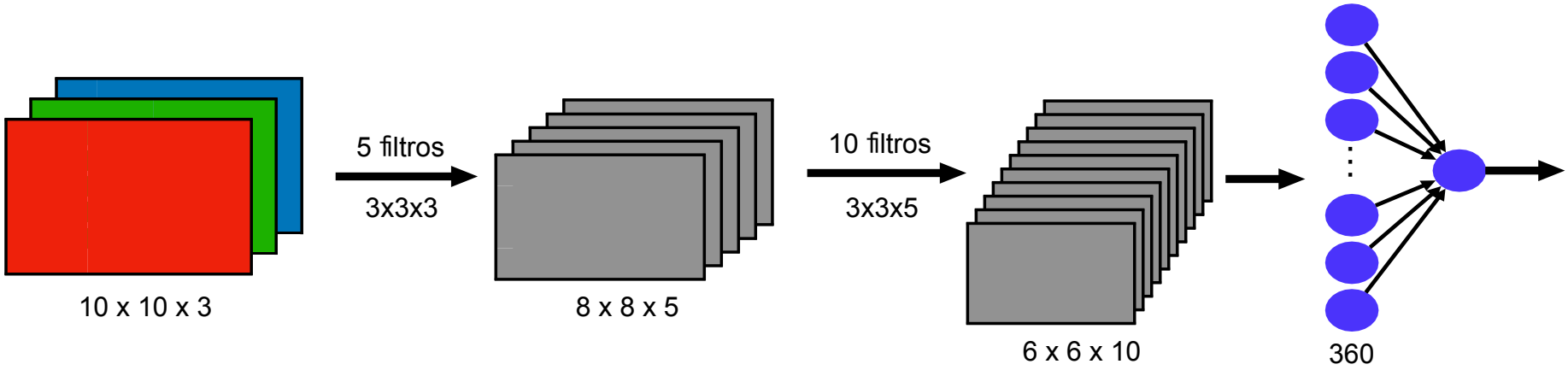
Redes neurais convolucionais

Uma rede convolucional com duas camadas:



Redes neurais convolucionais

No final, usamos uma camada totalmente conectada:



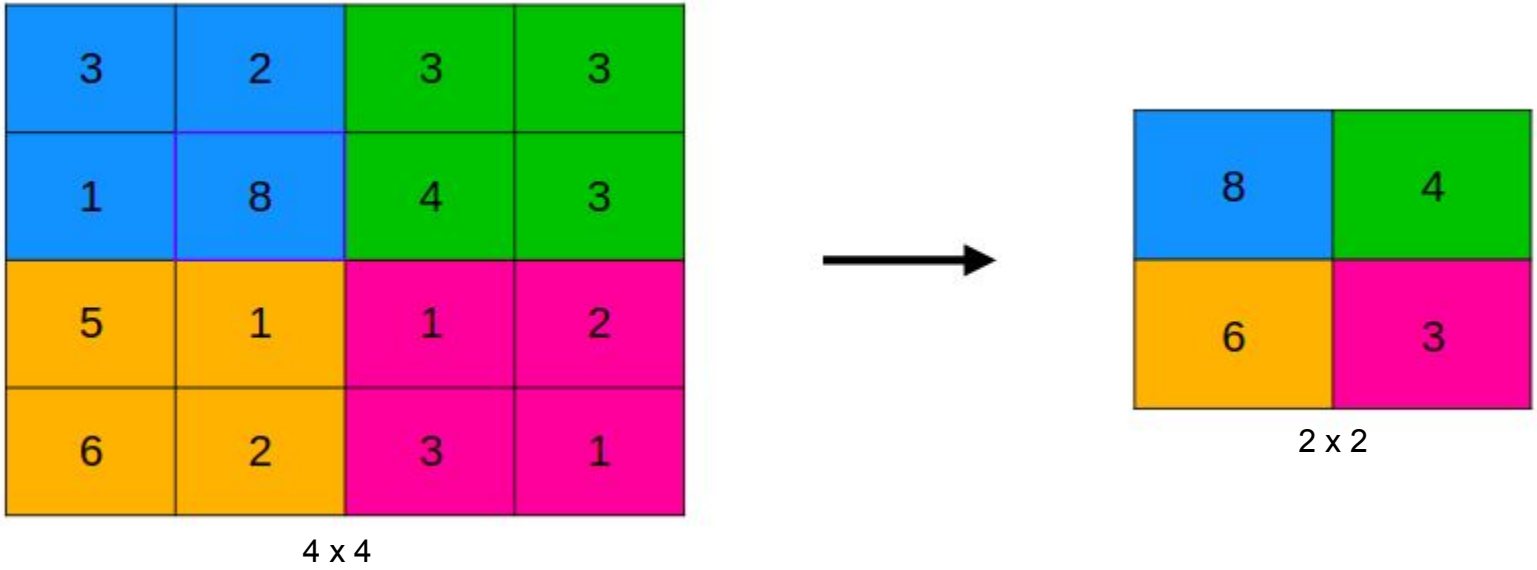
Pooling

A operação de pooling é usada para aumentar a velocidade e diminuir a quantidade de memória usada.

Ela também realça características encontradas pelos filtros.

Max Pooling

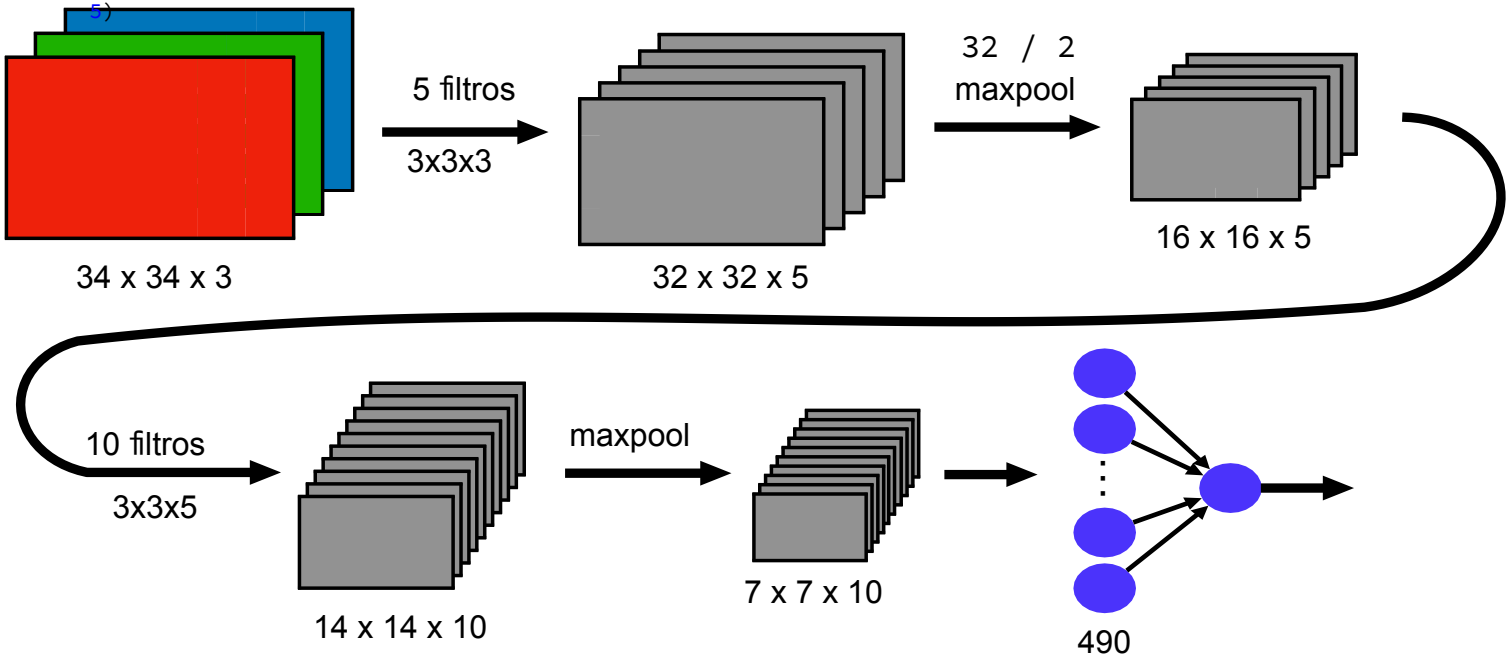
Stride = 2
Filtro 2x2
Tamanho Final = Img / S



Redes neurais convolucionais

TF 2D: [(34 - 3 + 0) / 1] + 1 = 32

TF 3D: $Img_{(34,34,\ 3)}$ o $5*W_{(3,\ 3,\ 3)}$ = $Saída_{(32,\ 32,\ 5)}$

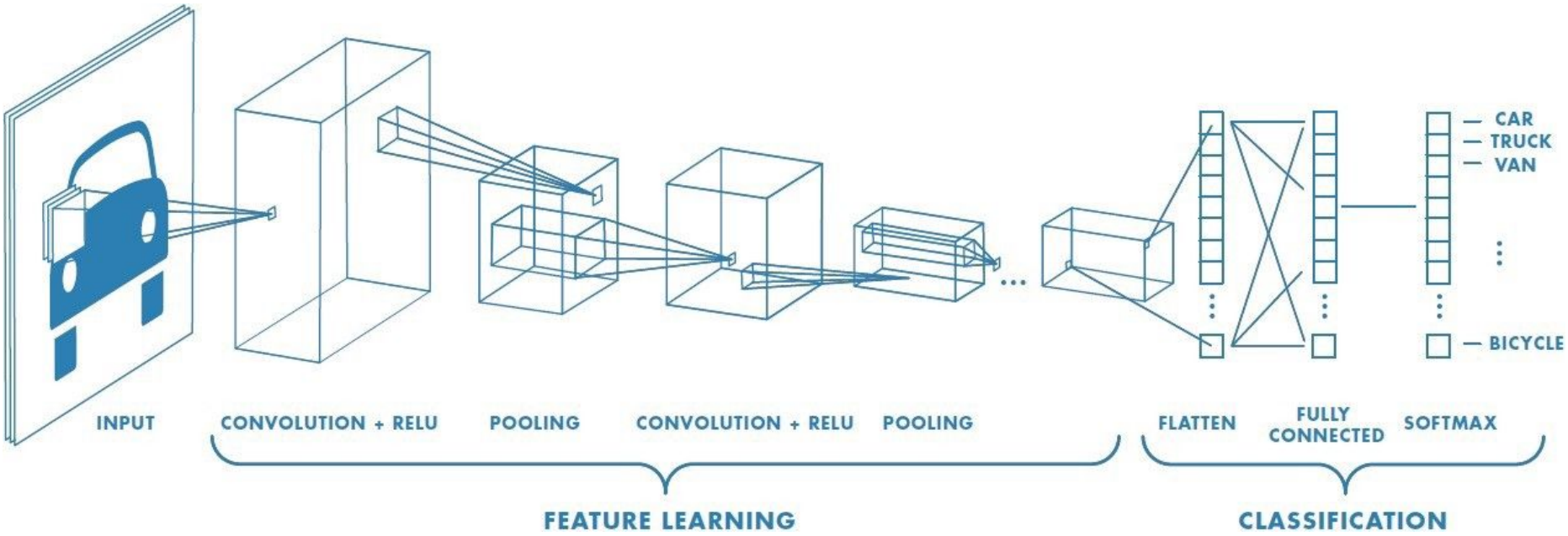


Redes neurais convolucionais

Nas Redes Neurais Convolucionais é muito comum usarmos o padrão:

Convolução ➡ pooling ➡ convolução ➡ pooling ➡ …
➡ convolução ➡ pooling ➡ camada totalmente conectada

Redes neurais convolucionais



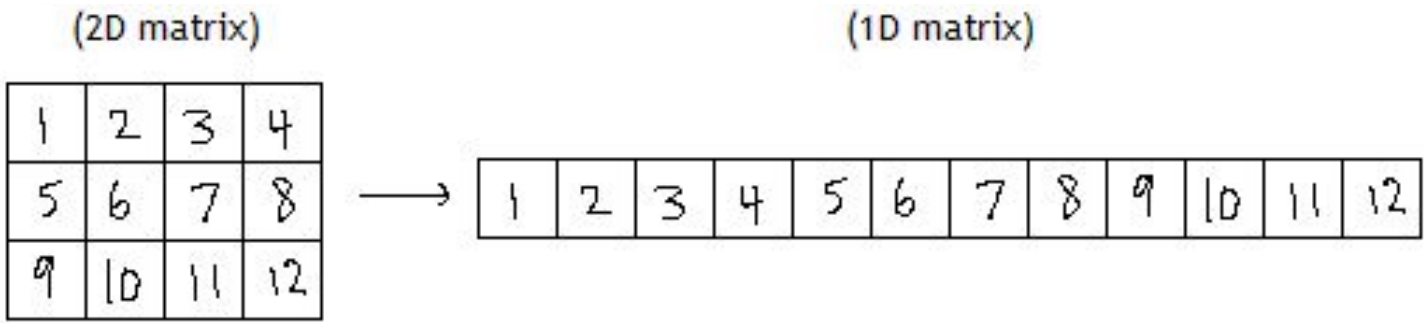
Redes neurais convolucionais

As Redes Neurais Convolucionais, ao longo de suas camadas, vão aprendendo a detectar padrões cada vez mais complexos.

É possível visualizar esses padrões através da criação de imagens artificialmente para maximizar uma das saídas de uma determinada camada.

Redes neurais convolucionais em dados categóricos

Redes neurais convolucionais 1D



Redes neurais convolucionais 1D

4X3 + 1X7 + 3X5 = 34

Matriz 1D	4	1	3	8	4	0	3	8	0	7	7	7	1	2
Filtro	3	7	5											
Saída		34												

Redes neurais convolucionais 1D

1X3 + 3X7 + 8X5 = 64

Matriz 1D	4	1	3	8	4	0	3	8	0	7	7	7	1	2
Filtro		3	7	5										
Saída		34	?											

Matriz 1D	4	1	3	8	4	0	3	8	0	7	7	7	1	2
Filtro		3	7	5										
Saída		34	64											

Redes neurais convolucionais 1D

Tamanho Final = [(Matriz - W + 2*P) / S] + 1 (Mesma fórmula da 2D)
Tamanho Final = [(6 - 3 + 2*0) / 1] + 1 = 4

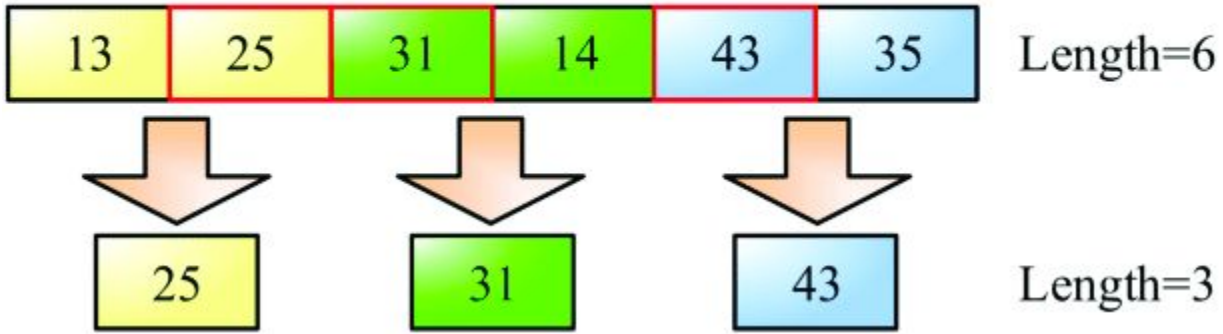
Cálculos:

1º Elemento: 1X2 + 3X0 + 3X1 = 5
2º Elemento: 3X2 + 3X0 + 0X1 = 6
3º Elemento: 3X2 + 0X0 + 1X1 = 7
4º Elemento: 0X2 + 1X0 + 2X1 = 2

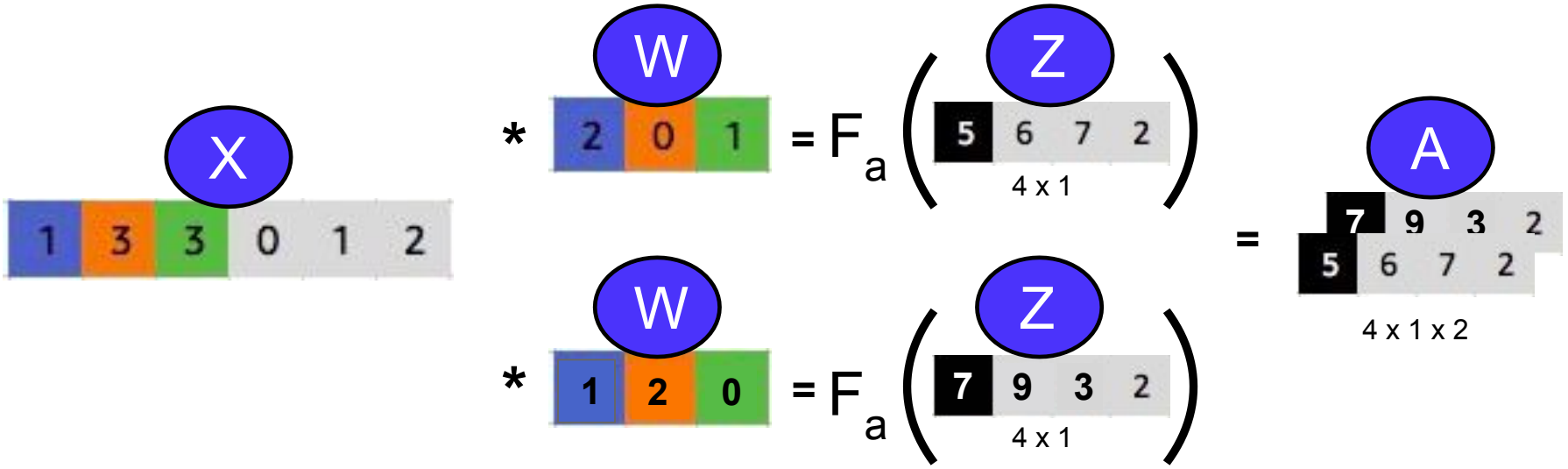
Max Pooling 1D

Stride = 2
Filtro 2x1
Tamanho Final = Matriz / S

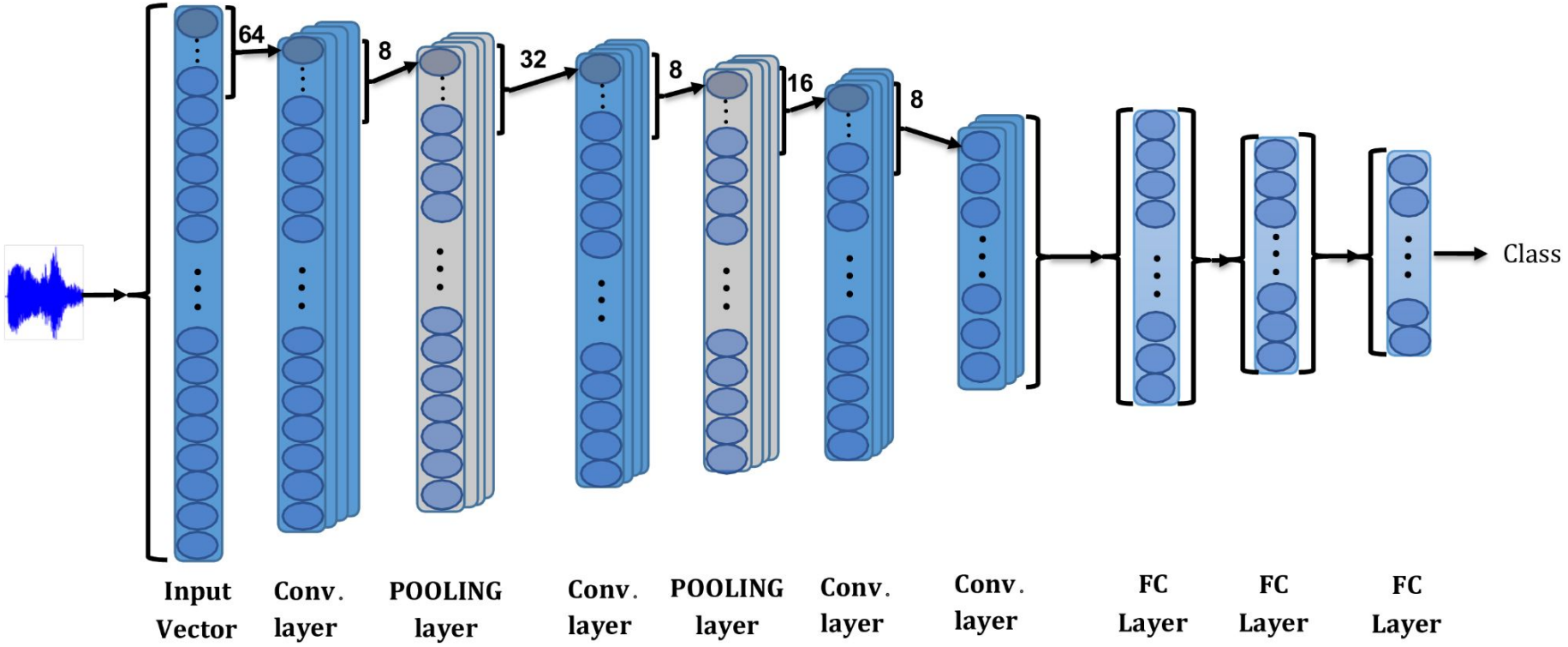
Tamanho Final = 6 / 2 = 3



Redes neurais convolucionais 1D



Redes neurais convolucionais 1D



Aplicações na bioinformática e biomedicina

AlphaFold

nature > articles > article

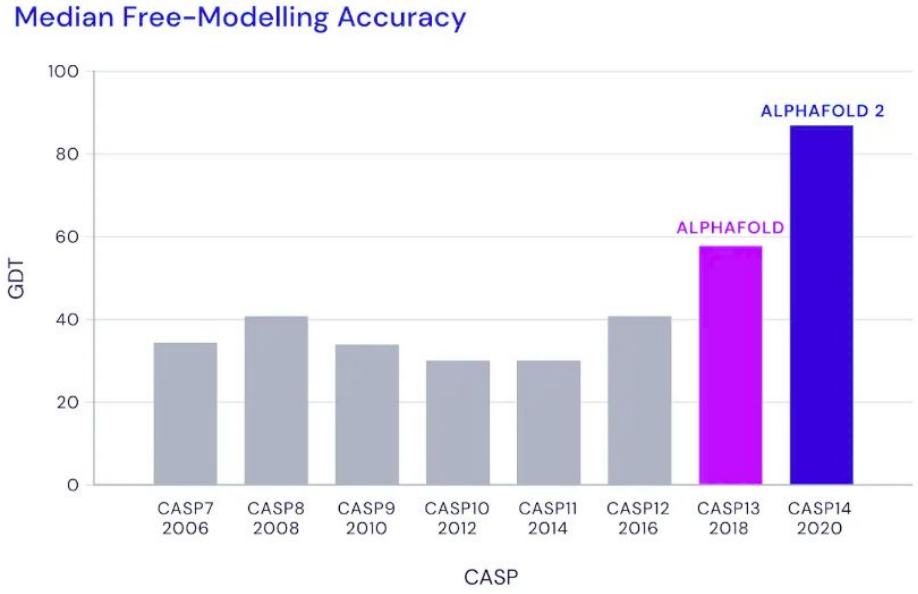
Article | Published: 15 January 2020

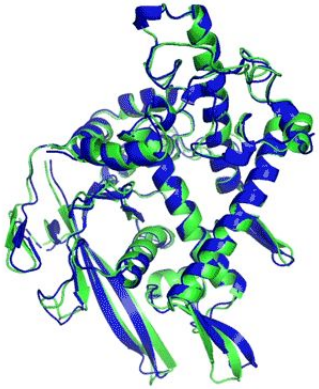
Improved protein structure prediction using potentials from deep learning

Andrew W. Senior ✉, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander W. R. Nelson, Alex Bridgland, Hugo Penedones, Stig Petersen, Karen Simonyan, Steve Crossan, Pushmeet Kohli, David T. Jones, David Silver, Koray Kavukcuoglu & Demis Hassabis

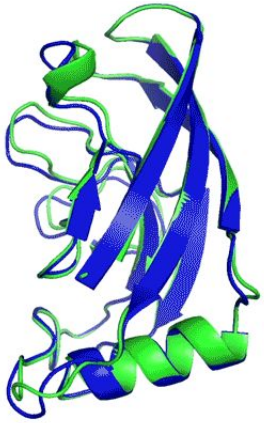
Nature **577**, 706–710(2020) | Cite this article

AlphaFold





T1037 / 6vr4
90.7 GDT
(RNA polymerase domain)



T1049 / 6y4f
93.3 GDT
(adhesin tip)

Experimental result
Computational prediction

RNAsamba

Published online 13 January 2020

NAR Genomics and Bioinformatics, 2020, Vol. 2, No. 1 **1**
doi: 10.1093/nargab/lqz024

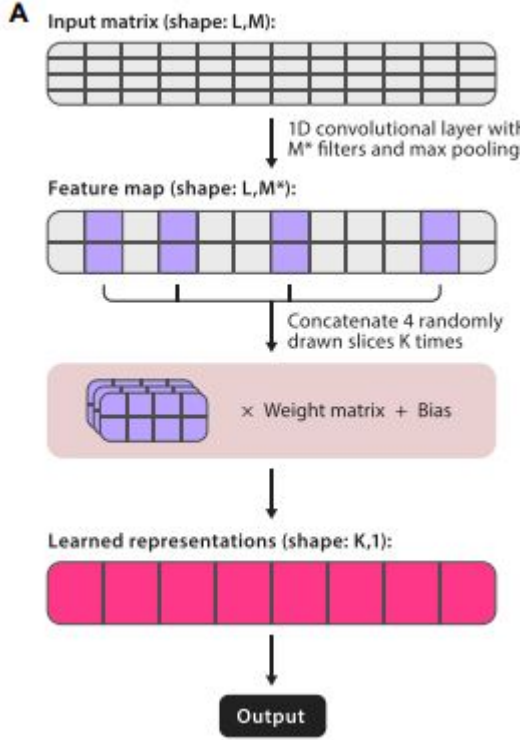
RNAsamba: neural network-based assessment of the protein-coding potential of RNA sequences

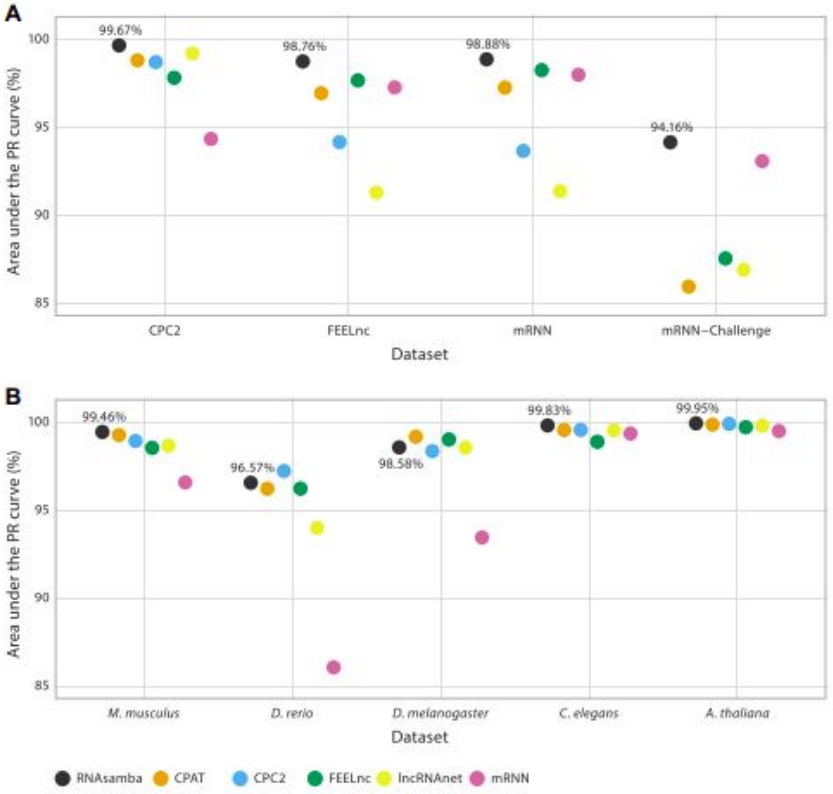
Antonio P. Camargo[1], Vsevolod Sourkov[2], Gonçalo A. G. Pereira[1] and Marcelo F. Carazzolle[1,*]

[1]Department of Genetics, Evolution, Microbiology and Immunology, Institute of Biology, University of Campinas, Campinas, SP, 13083-862, Brazil and [2]Department of Computer Science, ReDNA Labs, Pattaya, Chonburi, 20150, Thailand

Received August 09, 2019; Revised November 15, 2019; Editorial Decision December 10, 2019; Accepted December 17, 2019

RNAsamba


A
Input matrix (shape: L,M):
1D convolutional layer with
M* filters and max pooling
Feature map (shape: L,M*):
Concatenate 4 randomly
drawn slices K times
× Weight matrix + Bias
Learned representations (shape: K,1):
Output


A
Area under the PR curve (%)
100
95
90
85
99.67%
98.76%
98.88%
94.16%
CPC2
FEELnc
mRNN
mRNN−Challenge
Dataset
B
Area under the PR curve (%)
100
95
90
85
99.46%
96.57%
98.58%
99.83%
99.95%
M. musculus
D. rerio
D. melanogaster
C. elegans
A. thaliana
Dataset
RNAsamba
CPAT
CPC2
FEELnc
lncRNAnet
mRNN

CovXNet

Computers in Biology and Medicine 122 (2020) 103869

Contents lists available at ScienceDirect

Computers in Biology and Medicine

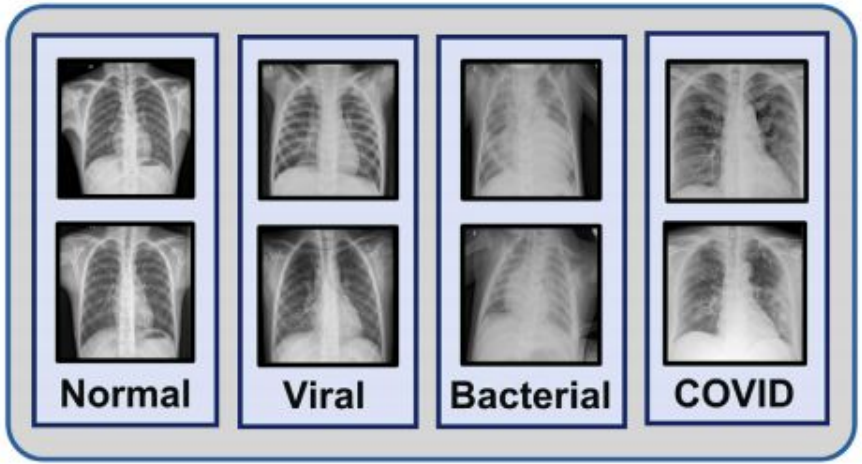
journal homepage: www.elsevier.com/locate/compbiomed

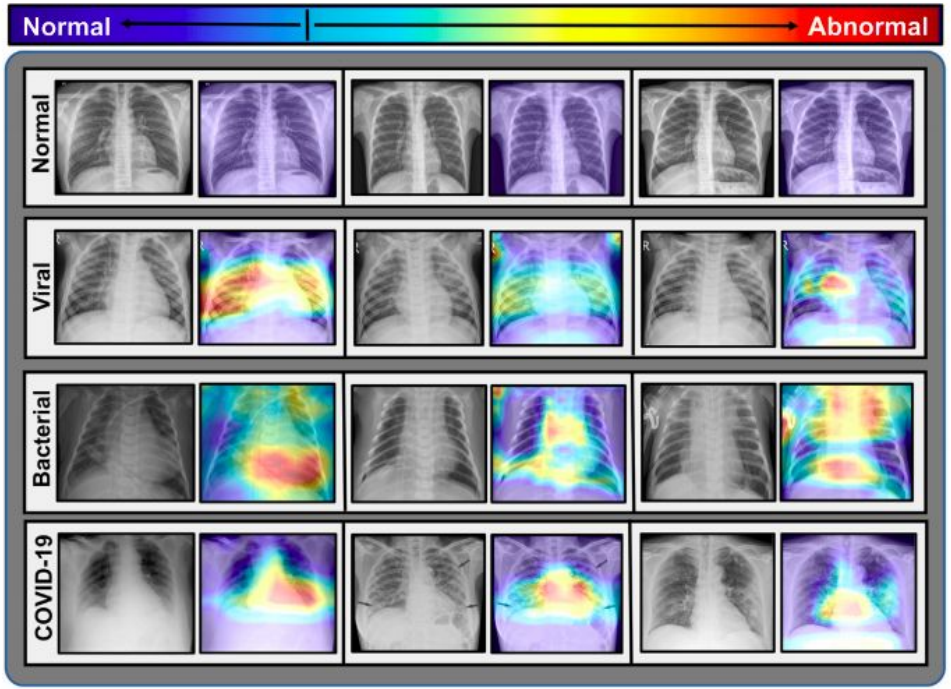
CovXNet: A multi-dilation convolutional neural network for automatic COVID-19 and other pneumonia detection from chest X-ray images with transferable multi-receptive feature optimization ☆

Tanvir Mahmud, Md Awsafur Rahman, Shaikh Anowarul Fattah *

Department of EEE, BUET, ECE Building, West Palashi, Dhaka 1205, Bangladesh

CovXNet


Normal
Viral
Bacterial
COVID


Normal
Abnormal
Normal
Viral
Bacterial
COVID-19

Tipos de câncer através da expressão gênica

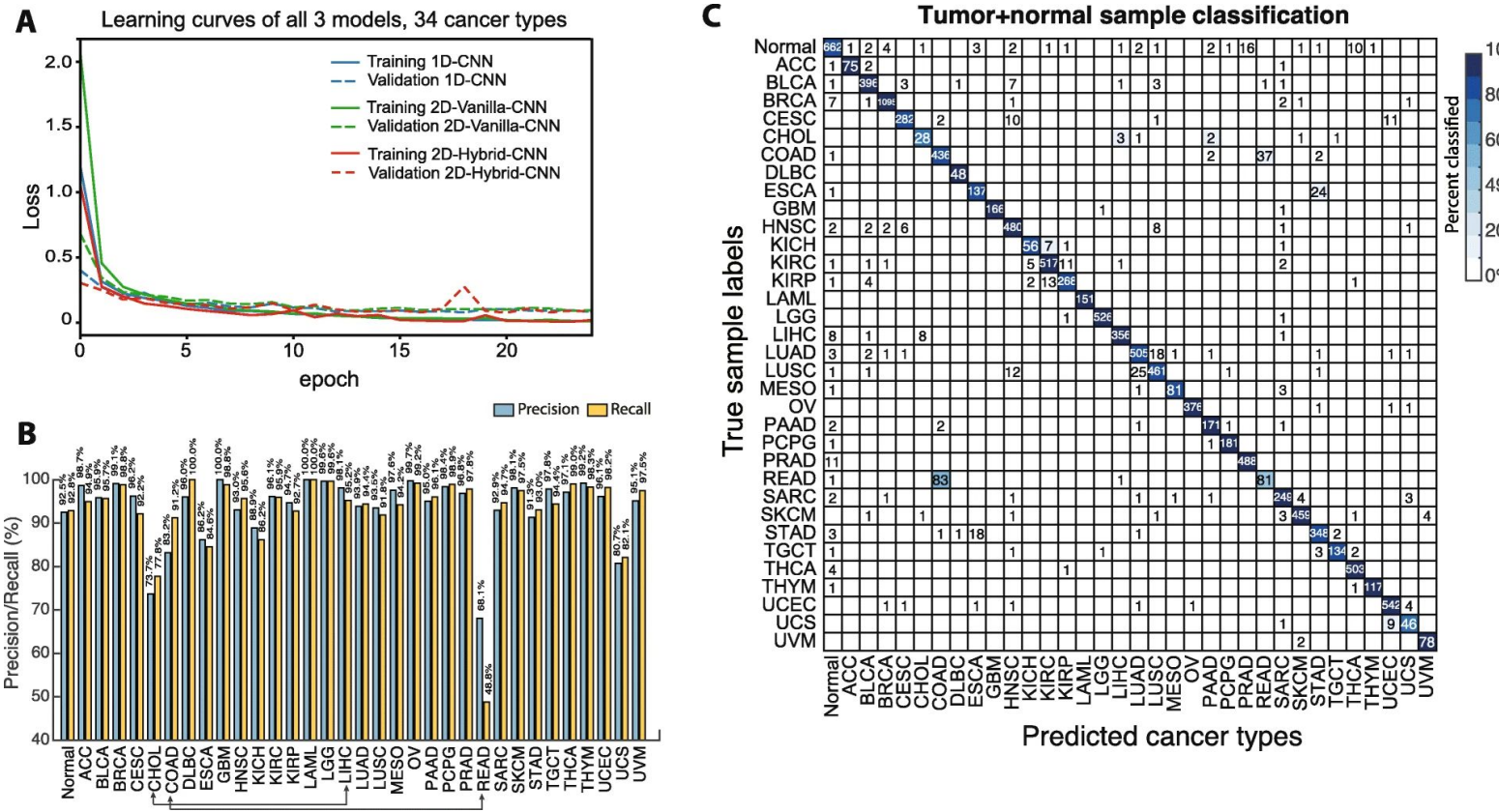
Research | Open Access | Published: 03 April 2020

Convolutional neural network models for cancer type prediction based on gene expression

Milad Mostavi, Yu-Chiao Chiu, Yufei Huang & Yidong Chen

BMC Medical Genomics **13**, Article number: 44 (2020) | Cite this article

Tipos de câncer através da expressão gênica



Câncer de mama

S.I. : Healthcare Analytics | Published: 09 October 2020

Convolutional neural network-based models for diagnosis of breast cancer

Mehedi Masud, Amr E. Eldin Rashed & M. Shamim Hossain

Neural Computing and Applications (2020) | Cite this article

Câncer de mama