

REDE NEURAL CONVOLUCIONAL

Prof. Valmir Macário Filho

DC - UFRPE





IMAGENS



VISÃO GERAL

- Imagens
- Redes Neurais Completamente Conectadas
- Redes Neurais Convolucionais
- Exemplos de CNNs

IMAGENS SÃO NÚMEROS



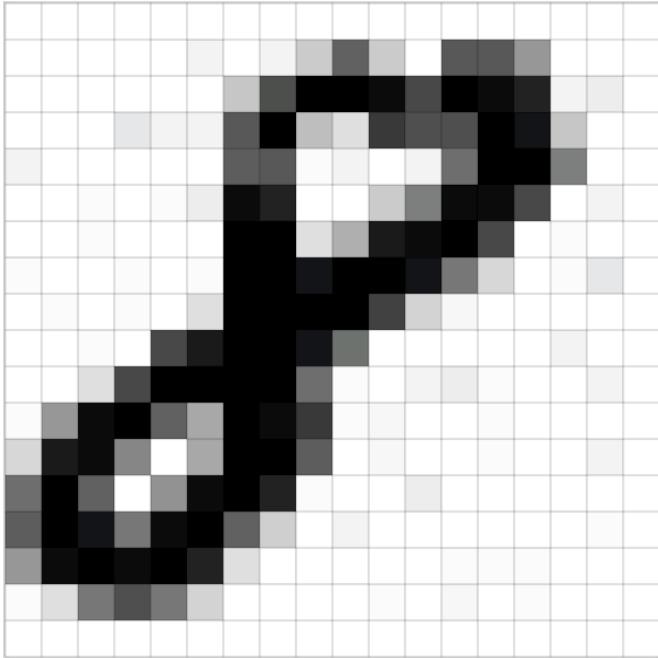
```
[[ 9  1 29 70 114 76  0  8  4  5  5  0 111 162 9  8 62 62]
[ 3  0 33 61 102 106 34  0  0  0  0 49 182 150 1 12 65 62]
[ 1  0 40 54 123 90 72 77 52 51 49 121 205 98 0 15 67 59]
[ 3  1 41 57 74 54 96 181 220 170 90 149 208 56 0 16 69 59]
[ 6  1 32 36 47 81 85 90 176 206 140 171 186 22 3 15 72 63]
[ 4  1 31 39 66 71 71 97 147 214 203 190 198 22 6 17 73 65]
[ 2  3 15 30 52 57 68 123 161 197 207 200 179 8 8 18 73 66]
[ 2  2 17 37 34 40 78 103 148 187 205 225 165 1 8 19 76 68]
[ 2  3 20 44 37 34 35 26 78 156 214 145 200 38 2 21 78 69]
[ 2  2 20 34 21 43 70 21 43 139 205 93 211 70 0 23 78 72]
[ 3  4 16 24 14 21 102 175 120 130 226 212 236 75 0 25 78 72]
[ 6  5 13 21 28 28 97 216 184 90 196 255 255 84 4 24 79 74]
[ 6  5 15 25 30 39 63 105 140 66 113 252 251 74 4 28 79 75]
[ 5  5 16 32 38 57 69 85 93 120 128 251 255 154 19 26 80 76]
[ 6  5 20 42 55 62 66 76 86 104 148 242 254 241 83 26 80 77]
[ 2  3 20 38 55 64 69 80 78 109 195 247 252 255 172 40 78 77]
[ 10 8 23 34 44 64 88 104 119 173 234 247 253 254 227 66 74 74]
[ 32 6 24 37 45 63 85 114 154 196 226 245 251 252 250 112 66 71]]
```

Toda imagem é uma matriz de valores de pixels

<https://www.commonlounge.com/discussion/244616b76d3d40f88e8f12103a22743d>



IMAGENS SÃO NÚMEROS



8

2



RGB



Red



Green

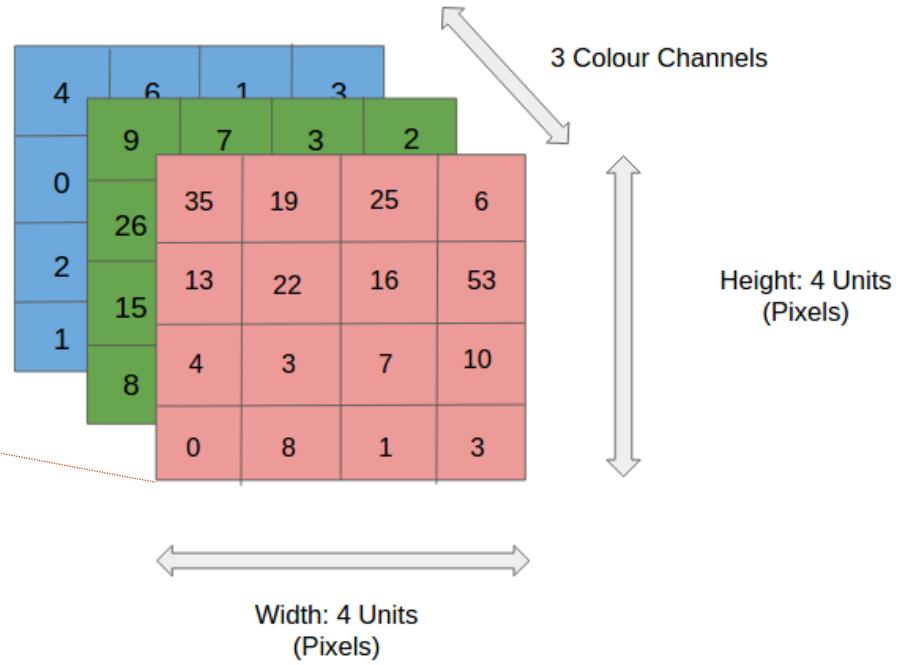


Blue



IMAGENS RGB

- Toda imagem RGB é uma **estrutura tridimensional**.



DATASETS DE IMAGENS

- CIFAR-100
- MNIST
- ImageNet

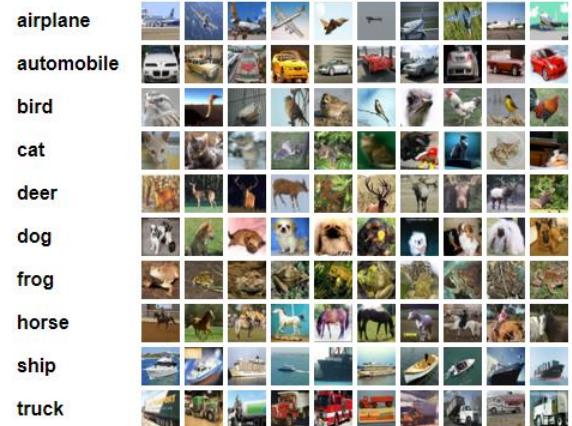
DATASETS: CIFAR

■ CIFAR-10

- 60K imagens RGB 32x32
- Total de 10 classes
 - 6000 imagens por classe.
- Treinamento/teste: 50K/10K.

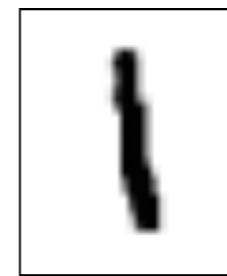
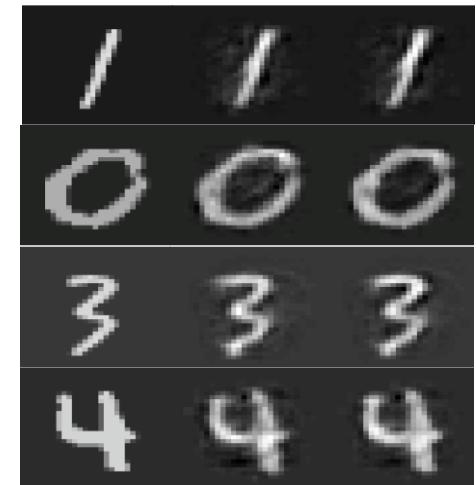
■ CIFAR-100

- similar ao CIFAR-10, mas com 100 classes com 600 imagens por classe.



DATASETS: MNIST (*HANDWRITTEN DIGITS DATASET*)

- Conjunto de dados com 70k exemplos
 - Treinamento: 60k
 - Teste: 10k
- Cada exemplo: 28x28 pixels
- Melhores desempenhos:
 - classificador linear: 12% error
 - (Gaussian) SVM 1.4% error
 - ConvNets <1% error



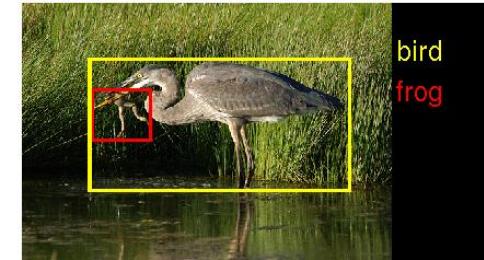
[A 28x28 matrix of binary values (0 or 1) representing the digit '1' in a 28x28 grid.]	\approx	[A 1x1 matrix containing the value 1, indicating the digit is a '1'.]
--	-----------	---

[<http://yann.lecun.com/exdb/mnist/>]



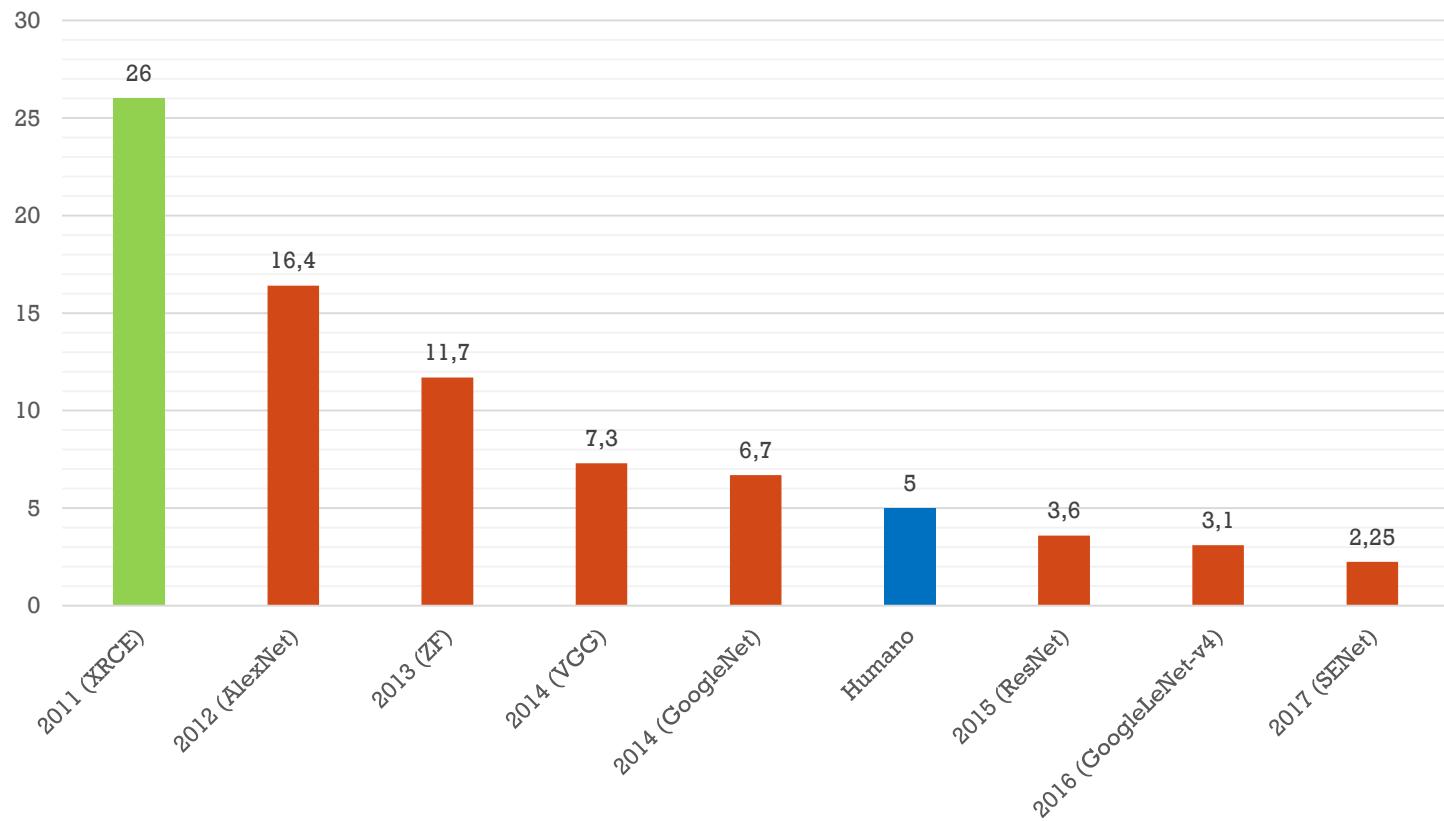
DATASETS: IMAGENET

- Tamanho médio das imagens: 482x415 pixels.
- 14M+ de imagens no total
- 1M+ de imagens com “bounding boxes” anotados
- Competições:
 - ILSVRC
 - ImageNet Object Detection Challenge (Kaggle)



IMAGENET

Competição Imagenet





REDES NEURAIS CONVOLUCIONAIS

(CONVOLUTIONAL NEURAL NETWORKS, CONVNET, CNN)



MLPS VS CNNS

MLPs

- Utiliza camadas completamente conectadas
- Aceita apenas vetor como entrada

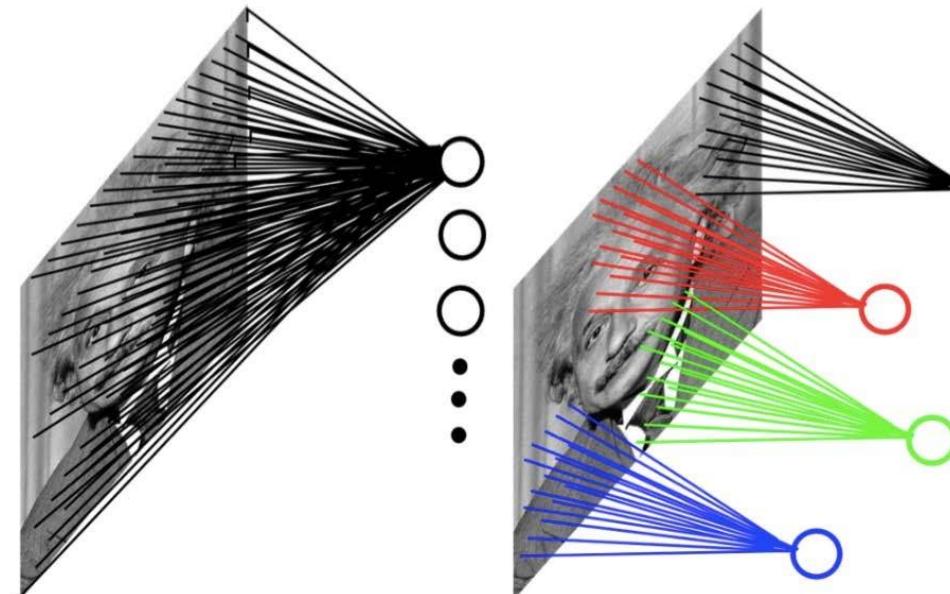
CNNs

- Também pode usar camadas esparsamente conectadas
- Também aceita matriz como entrada



Rede Neural Convolucional

- Problema: Entrada pode ter uma dimensão muito alta. Utilizando uma Multi-Layer Perceptron precisamos de uma quantidade muito grande de parâmetros.
- Inspirado pelos experimentos de neurofisiologia [Hubel & Wiesel 1962], CNNs são um tipo especial de redes neurais que usam o conceito de campo receptivo local.

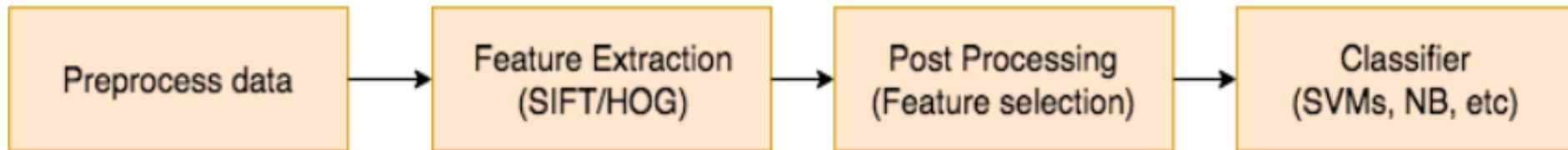


Exemplo: 200x200 imagem

- a) MLP: 40,000 hidden units
=> 1.6 bilhões de parâmetros
- b) CNN: 5x5 kernel, 100 campos receptivos => 2,500 parâmetros



ERA SEM DEEP LEARNING



- Cons:
 - Características Manuais difíceis de se construir
 - Processo que demanda tempo
 - Quais conjuntos de características maximizam acurácia?
 - Tende a causar overfitting.



COM DEEP LEARNING



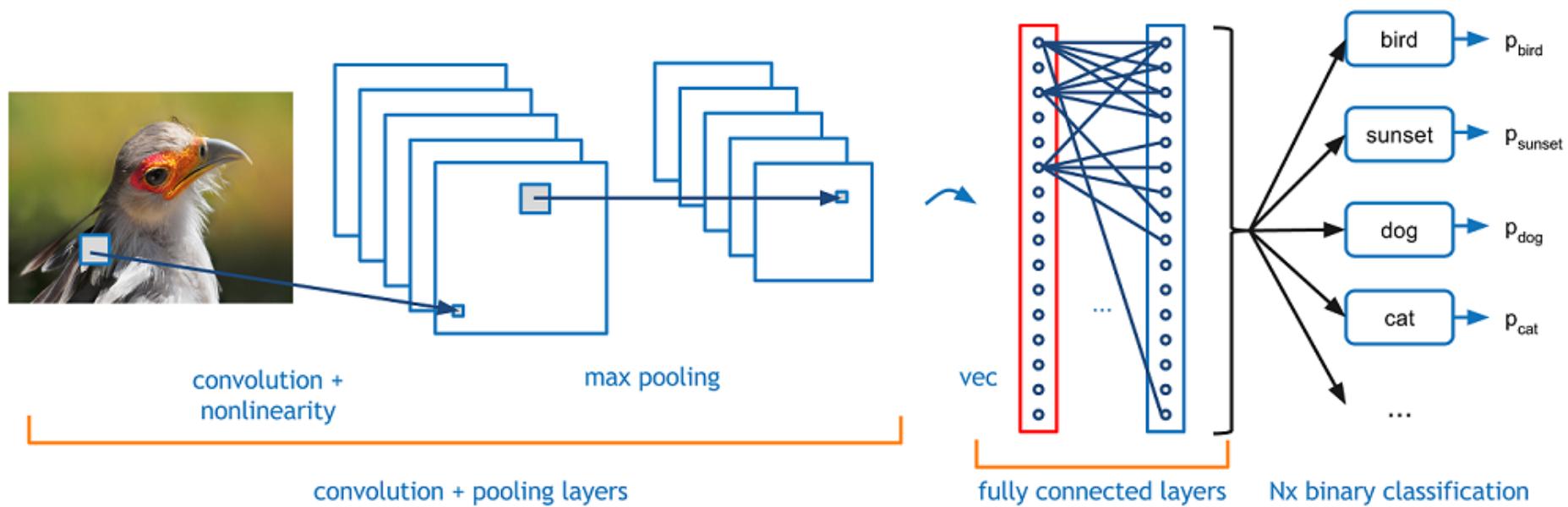
- CNN's foram inspiradas pela sensibilidade local e orientação seletiva do cérebro
- Projetaram uma rede neural que implicitamente extrai características relevantes da entrada
- CNN é uma rede de aprendizagem supervisionada que extrai propriedades topológicas a partir da sua entrada
- Redes Convolucionais são um tipo especial de MLP



- Como qualquer outra rede neural a arquitetura é treinada com alguma variação do algoritmo back-propagation
- Capazes de reconhecer padrões com muita variabilidade (caracteres escritos a mão)
- Desenvolvidas para dados de duas dimensões, como imagens e vídeos



Convolutional Neural Network (CNN)



CAMADAS DA REDE CONVOLUCIONAL

Entrada: $32 \times 32 \times 3 \rightarrow$ 32 pixels de altura, 32 de largura e 3 camadas de RBG

Convolução: Camada que calculará a saída dos neurônios conectados a regiões locais da entrada.

RELU: Função de ativação ponto-a-ponto como por exemplo max, min, avg, etc.

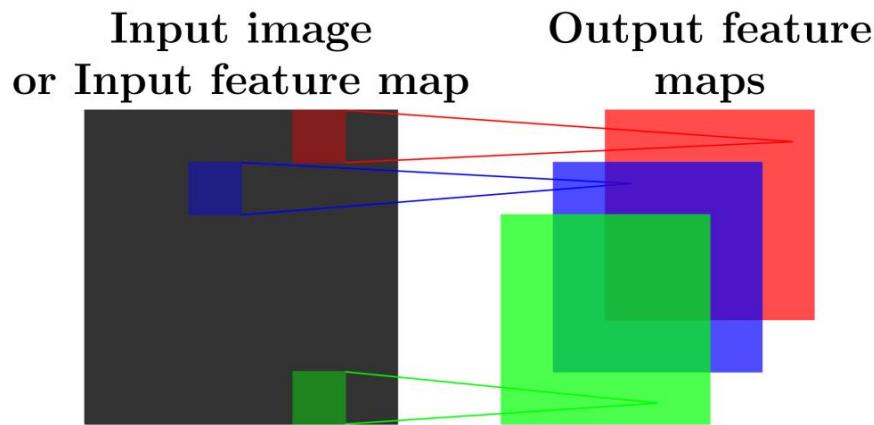
POOL: Camada de Downsampling

Classificador: MLP, SVM, Softmax, etc



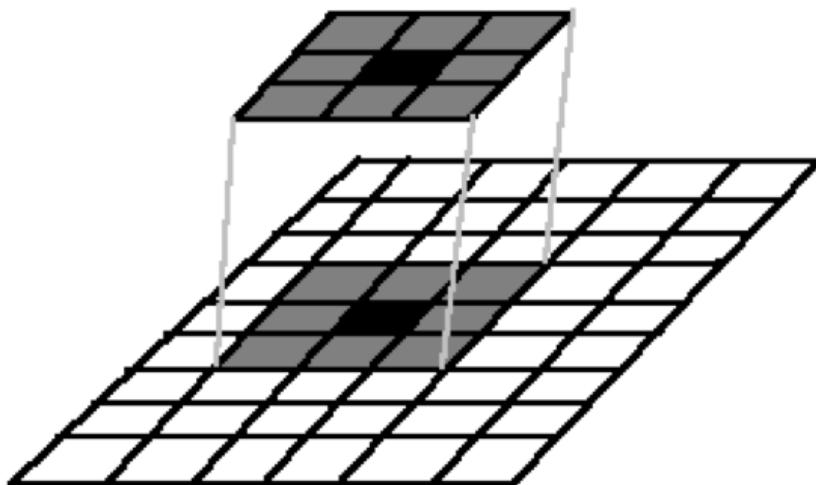
CAMADA DE CONVOLUÇÃO

- É a mais importante operação de uma CNN
- Atuam como extratores de características dos dados de entrada
- Detectam bordas, cantos, pontos de extremidade, etc. usando filtros



CONVOLUÇÃO

- Definição de kernel/ máscara



131	162	232	84	91	207	
104	-1	109	+1	237	109	
243	-2	202	+2	135	126	
185	-1	20	+1	61	225	
157	124	25	14	102	108	
5	155	116	218	232	249	



CONVOLUÇÃO

- Seja a máscara 3x3

W1	W2	W3
W4	W5	W6
W7	W8	W9

- e sejam z_1, z_2, \dots, z_9 a cor dos pixels sob a máscara
- O novo tom do pixel central será dado por
 - $R = w_1z_1 + w_2z_2 + \dots + w_9z_9$



FILTRAGEM

$$R = p_1 z_1 + p_2 z_2 + \cdots + p_n z_n$$

$$\begin{array}{|c|c|c|} \hline p_1 & p_2 & p_3 \\ \hline p_4 & p_5 & p_6 \\ \hline p_7 & p_8 & p_9 \\ \hline \end{array} \otimes
 \begin{array}{|c|c|c|} \hline z_1 & z_2 & z_3 \\ \hline z_4 & z_5 & z_6 \\ \hline z_7 & z_8 & z_9 \\ \hline \end{array}
 = R$$



FILTRAGEM NO DOMÍNIO ESPACIAL

$$\frac{1}{9} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} \dots$$

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

0									

FILTRAGEM NO DOMÍNIO ESPACIAL

$$\frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

0	10								

FILTRAGEM NO DOMÍNIO ESPACIAL

$$\frac{1}{9} \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \dots$$
$$F[x, y]$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$G[x, y]$$

			0	10	20				

FILTRAGEM NO DOMÍNIO ESPACIAL

$$\frac{1}{9}$$

1	1	1
1	1	1
1	1	1

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

0 10 20 30

FILTRAGEM NO DOMÍNIO ESPACIAL

$$\frac{1}{9} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

FILTRAGEM NO DOMÍNIO ESPACIAL

1	1	1
1	1	1
1	1	1

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

A MATEMÁTICA ATRÁS DA COMPARAÇÃO

1	-1	-1
-1	1	-1
-1	-1	1

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

1	1	-1
1	1	1
-1	1	1



A MATEMÁTICA ATRÁS DA COMPARAÇÃO

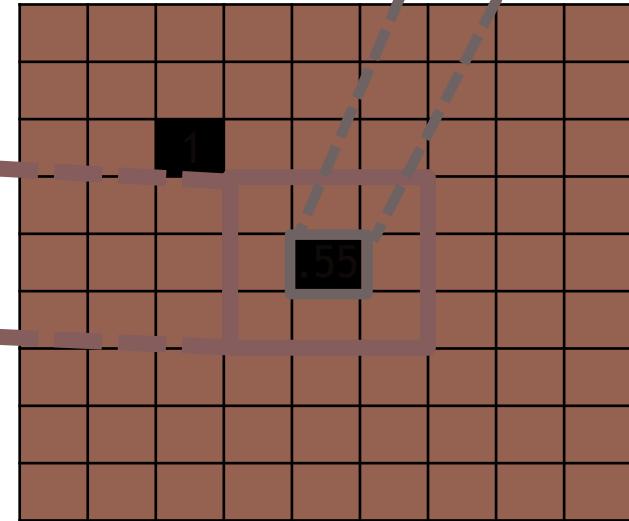
1	-1	-1
-1	1	-1
-1	-1	1

1	1	-1
1	1	1
-1	1	1

$$\frac{1 + 1 - 1 + 1 + 1 + 1 - 1 + 1 + 1}{9}$$

= .55

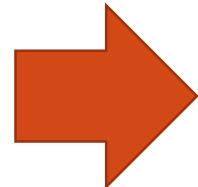
-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1



A MATEMÁTICA ATRÁS DA COMPARAÇÃO

1	-1	-1
-1	1	-1
-1	-1	1

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1



0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77



A MATEMÁTICA ATRÁS DA COMPARAÇÃO

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1



1			-1
-1	1		-1
-1	-1	1	

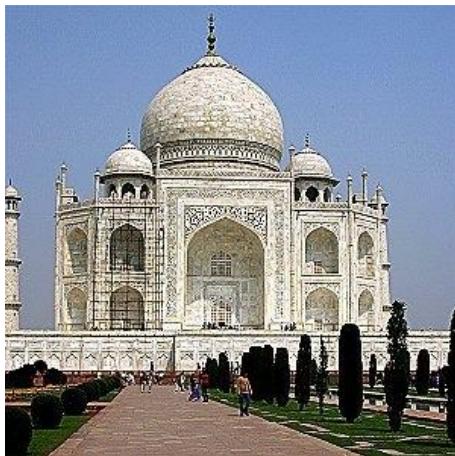
=

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

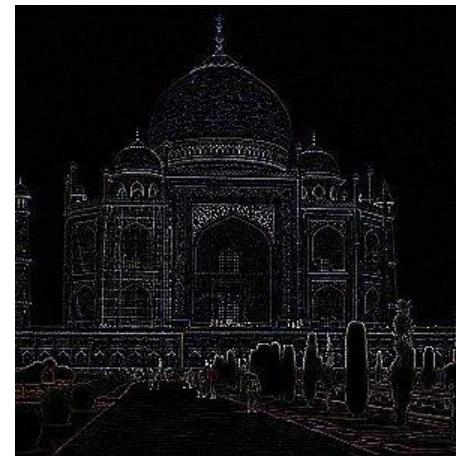


OPERAÇÃO DE CONVOLUÇÃO - EXEMPLO

- Um filtro comumente aprendido em uma CNN (nas CONVs iniciais) é o **detector de arestas**.



$$\begin{array}{|c|c|c|} \hline 0 & 1 & 0 \\ \hline 1 & -4 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array}$$



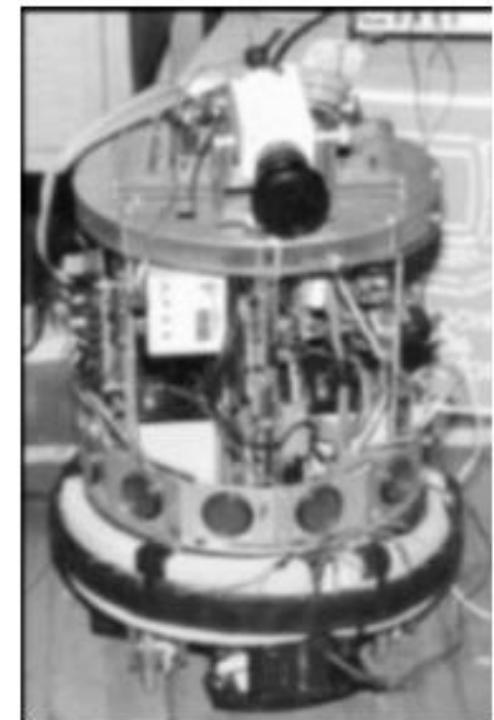
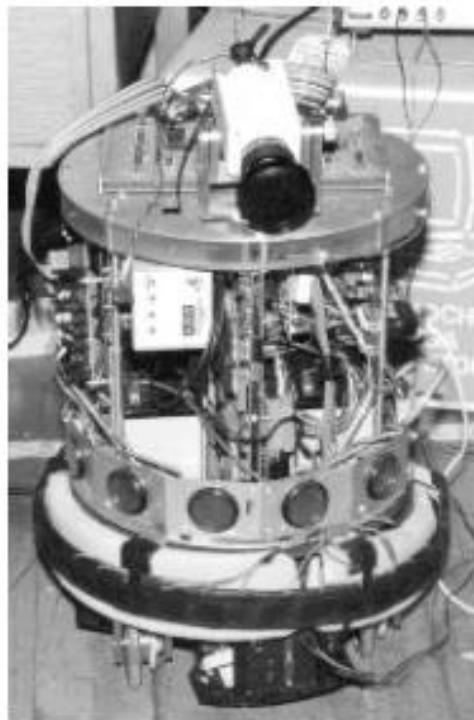
Fonte: [https://docs.gimp.org/en/plug-in-convmatrix.html'](https://docs.gimp.org/en/plug-in-convmatrix.html)

FILTRAGEM LINEAR

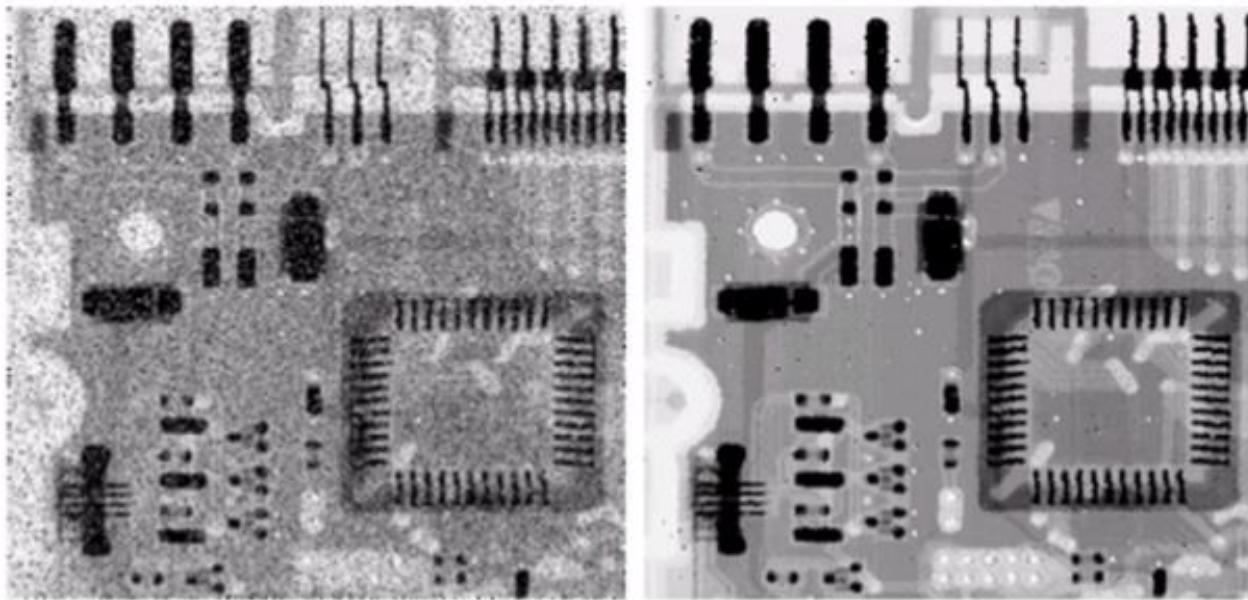
- Filtro de média ponderada
- Mais importância para pixels mais próximos do centro da máscara
 - Menos borramento

Exemplo: máscara 3×3

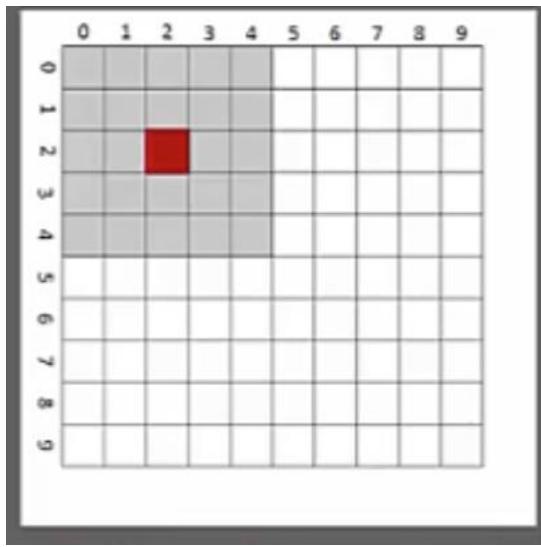
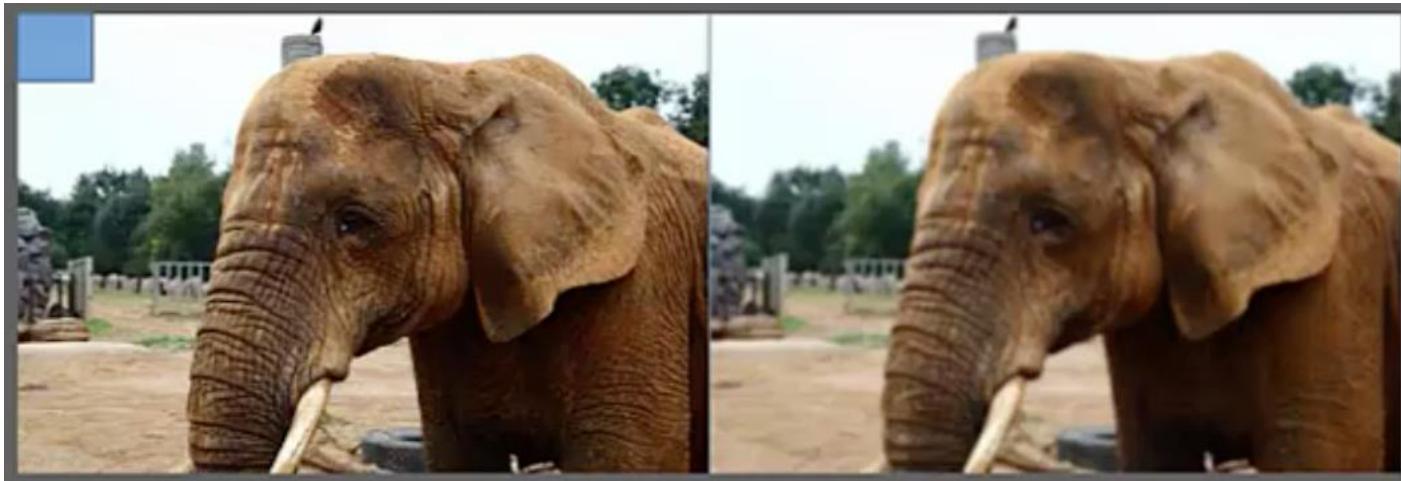
$$\frac{1}{100} \times \begin{array}{|c|c|c|} \hline 8 & 12 & 8 \\ \hline 12 & 20 & 12 \\ \hline 8 & 12 & 8 \\ \hline \end{array}$$



FILTROS LINEARES – SUAVIZAÇÃO



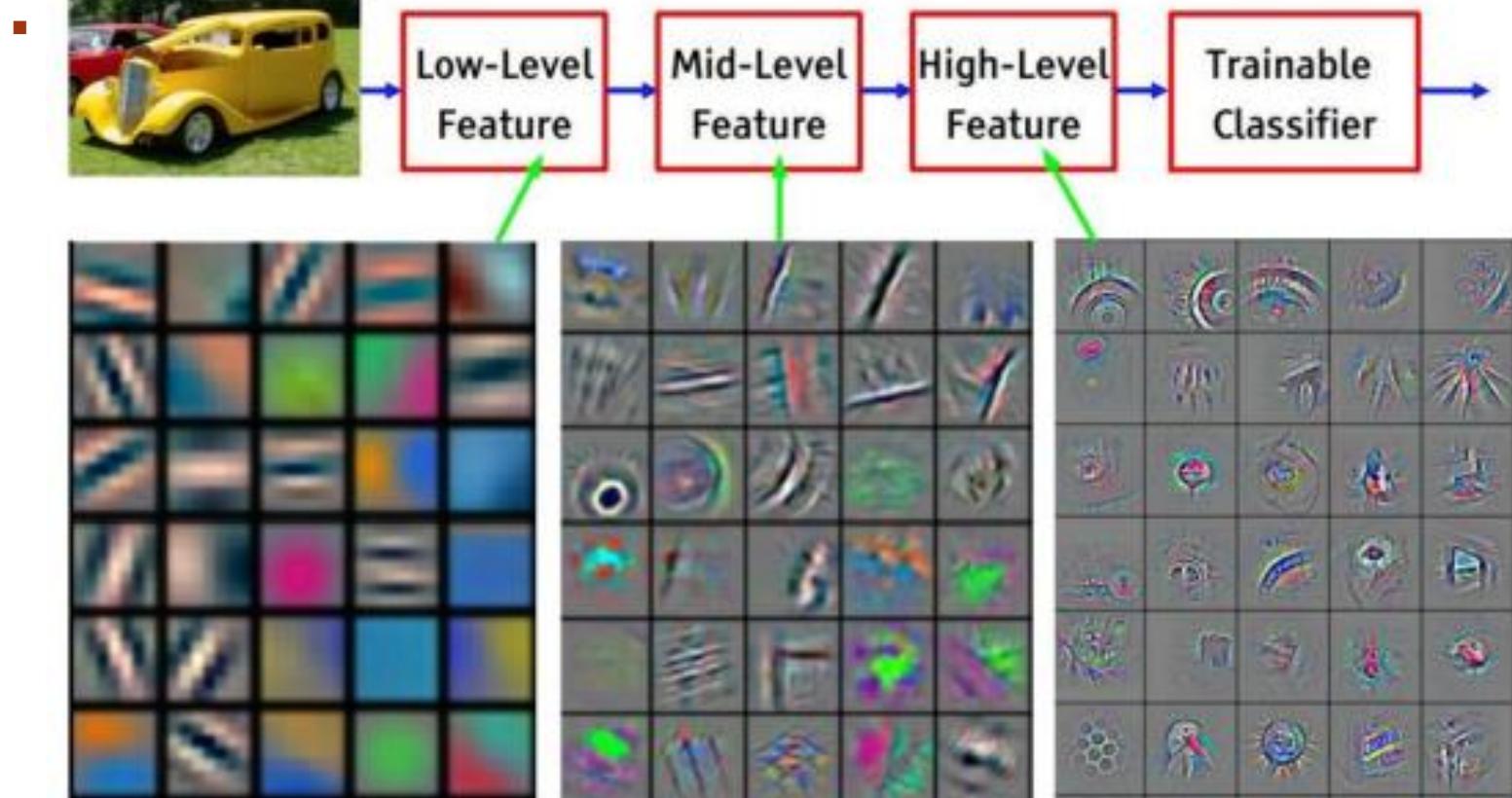
SUAVIZAÇÃO



$$Kernel_-= \frac{1}{25} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$



O QUE É DEEP LEARNING



UM POUCO DE HISTÓRIA..

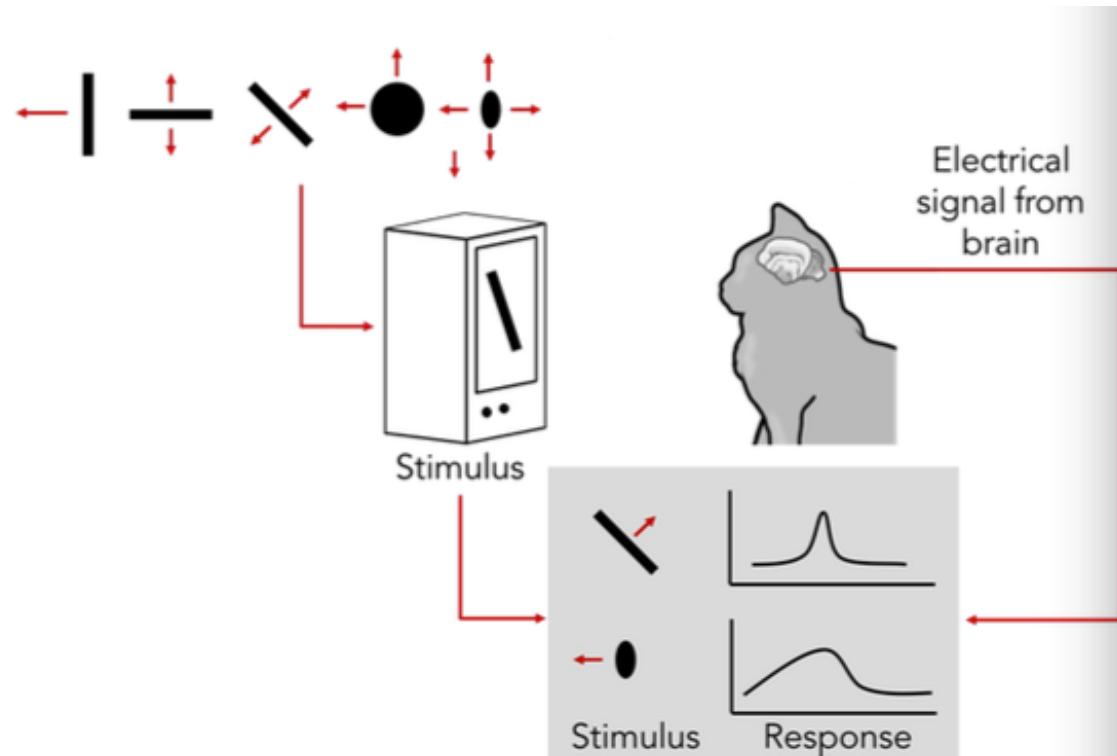
**Hubel & Wiesel,
1959**

RECEPTIVE FIELDS OF SINGLE
NEURONES IN
THE CAT'S STRIATE CORTEX

1962

RECEPTIVE FIELDS, BINOCULAR
INTERACTION
AND FUNCTIONAL ARCHITECTURE IN
THE CAT'S VISUAL CORTEX

1968...



Cat image by CNX OpenStax is licensed under CC BY 4.0; changes made



ORGANIZAÇÃO HIERÁRQUICA

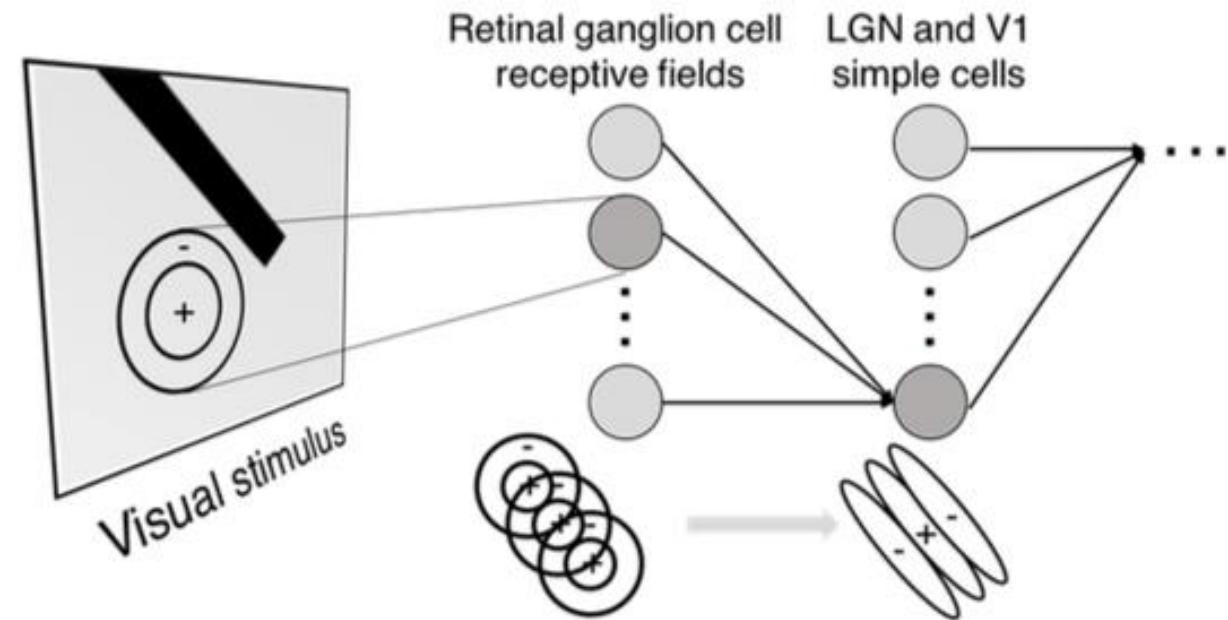


Illustration of hierarchical organization in early visual pathways by Lane McIntosh, copyright CS231n 2017

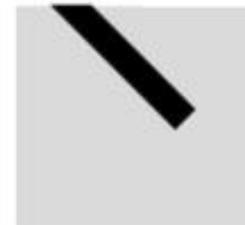
Simple cells:
Response to light orientation

Complex cells:
Response to light orientation and movement

Hypercomplex cells:
response to movement with an end point



No response



Response
(end point)



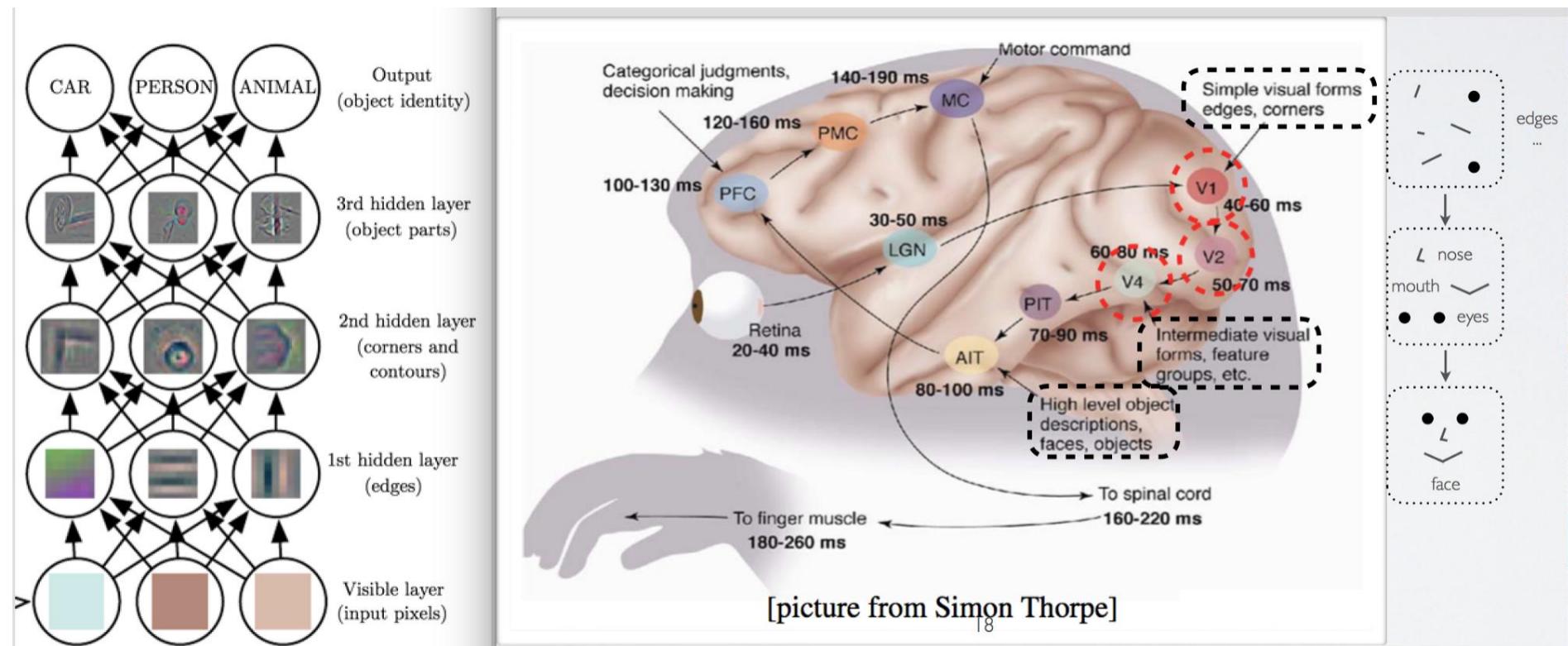
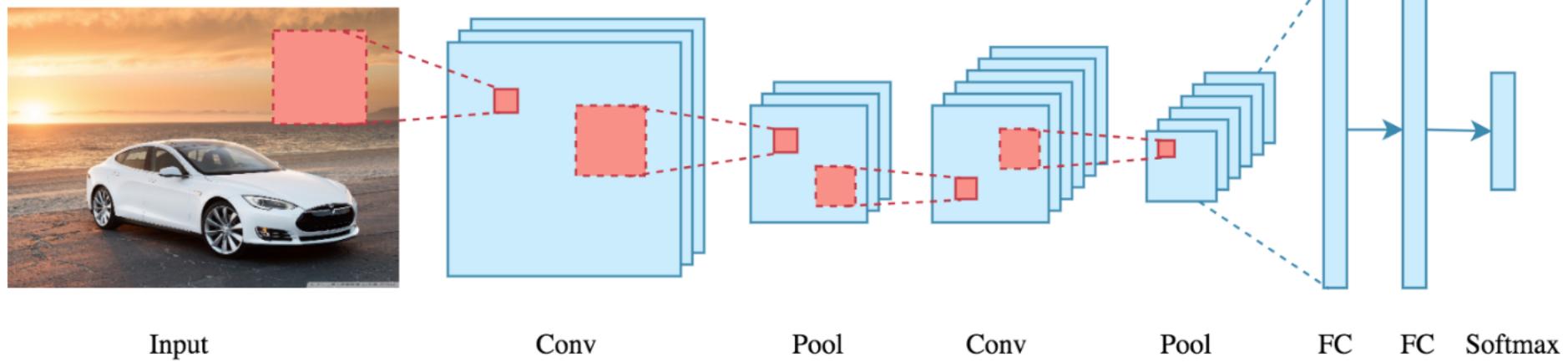


Fig 1. Image Courtesy: Hugo Larochelle Slides.



CNN

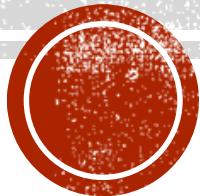


TIPOS DE CAMADAS

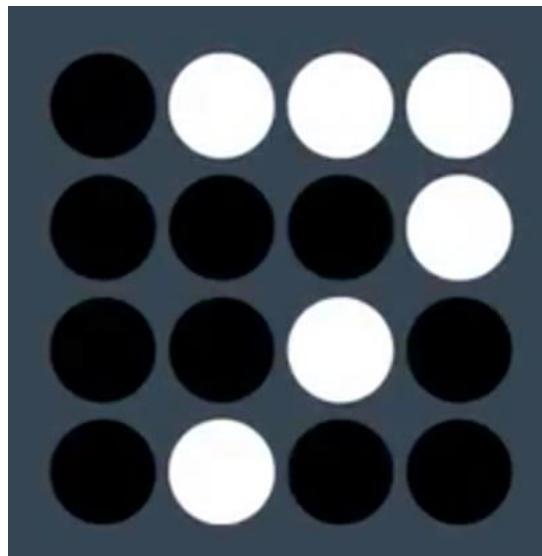
- Convolutiva (CONV)
- Pooling (POOL)
- Fully-conected (FC)
- Dropout(DO)
- Softmax



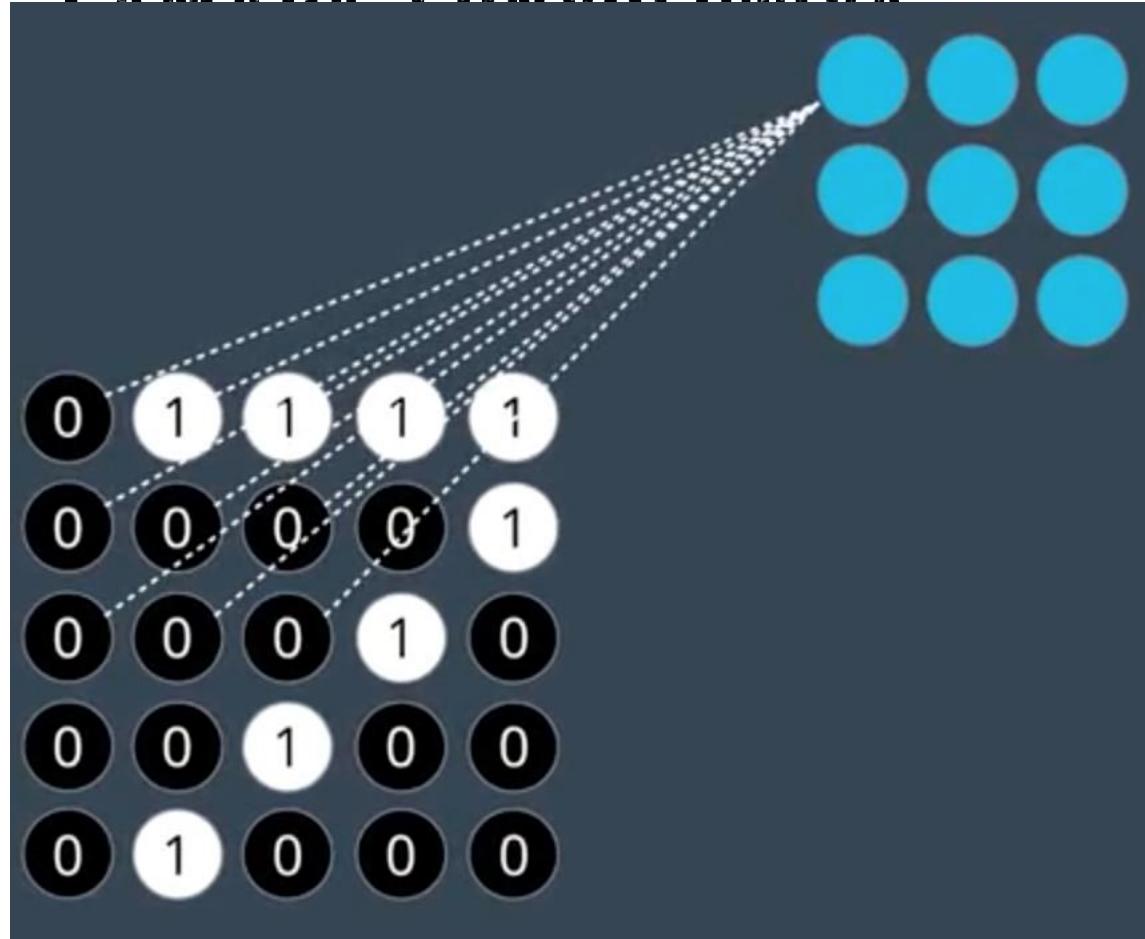
CAMADA CONVOLUTIVA



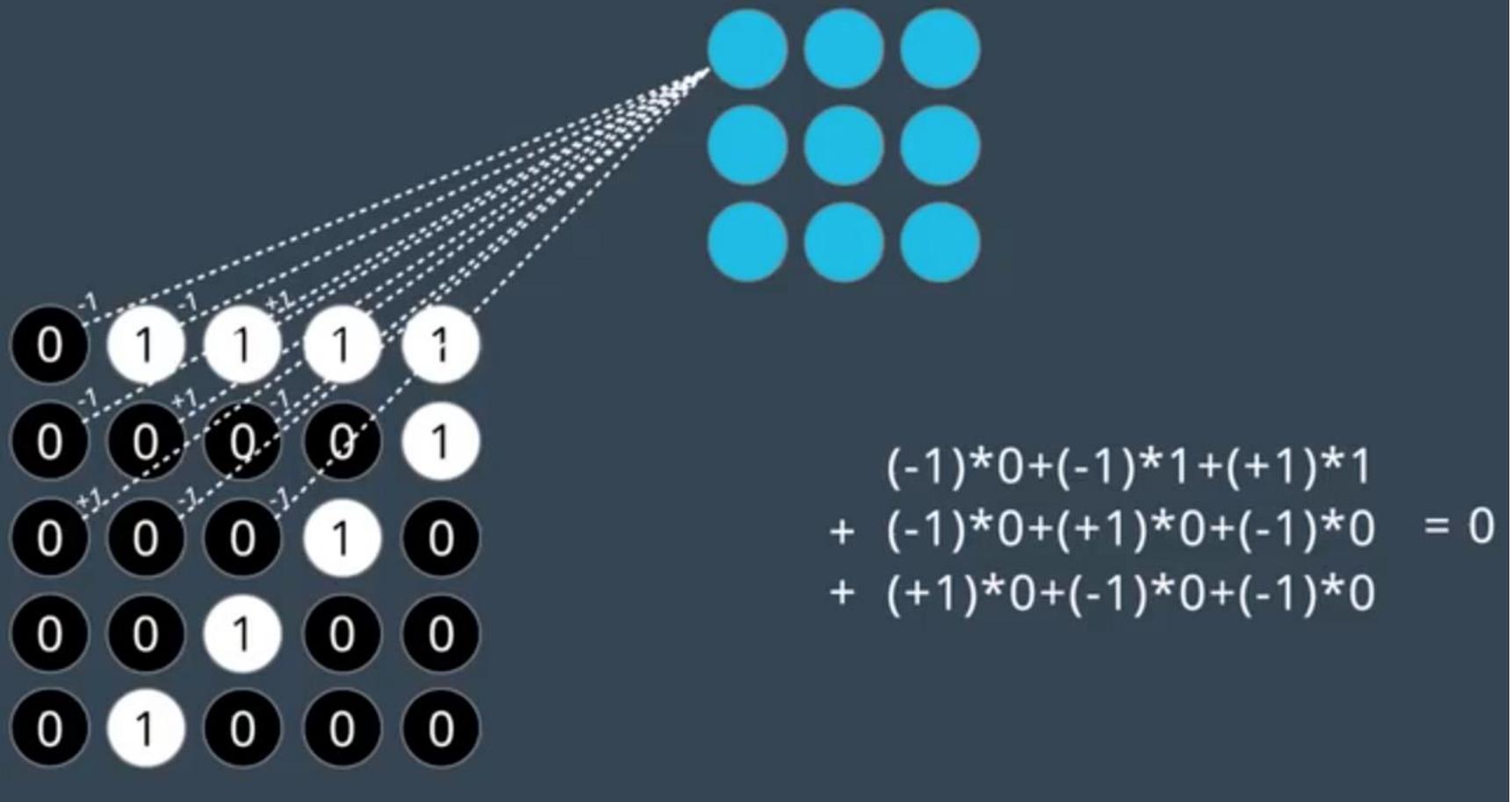
CAMADA CONVOLUTIVA



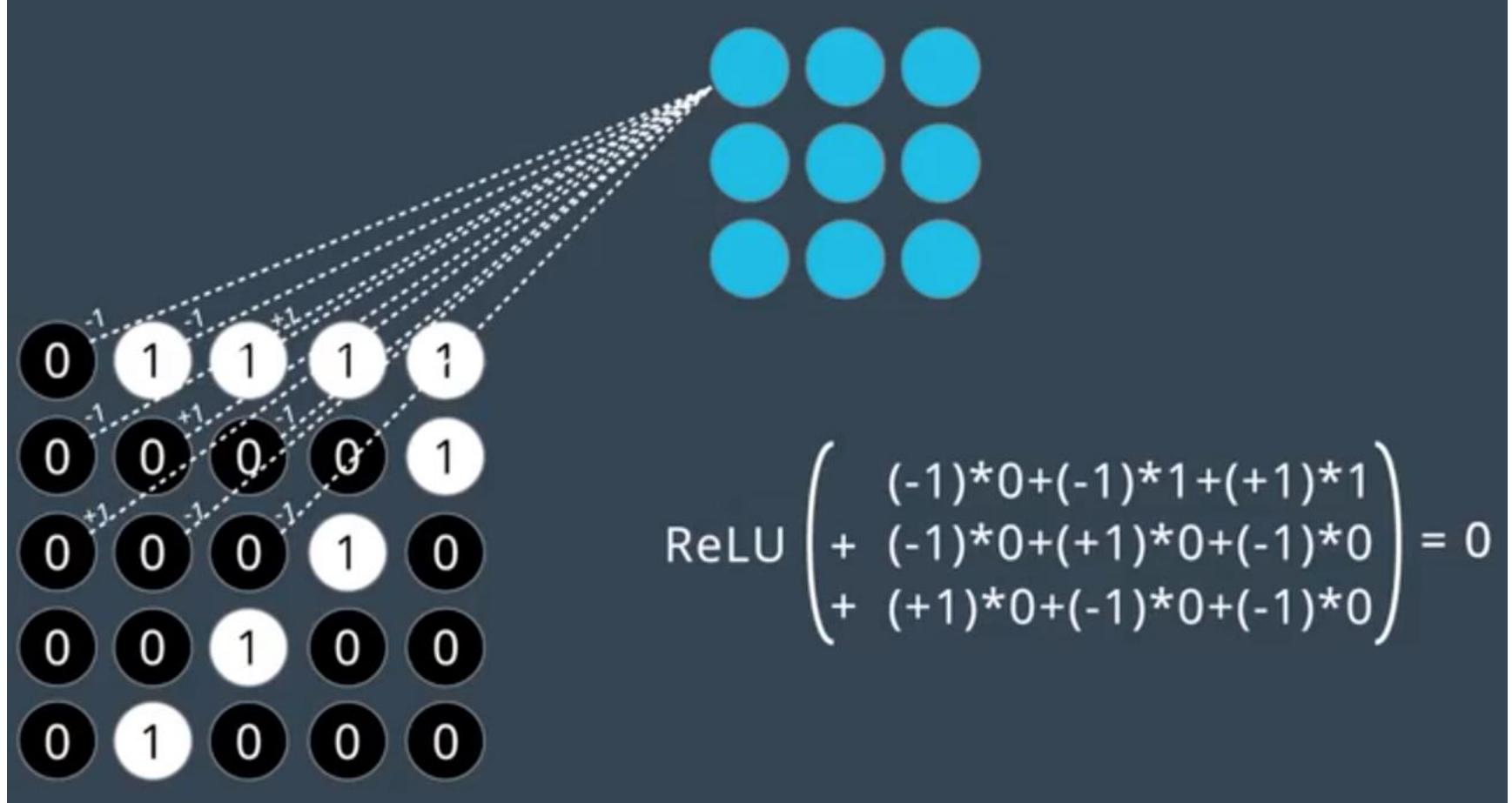
COMMAND CONTROL TRITIUM



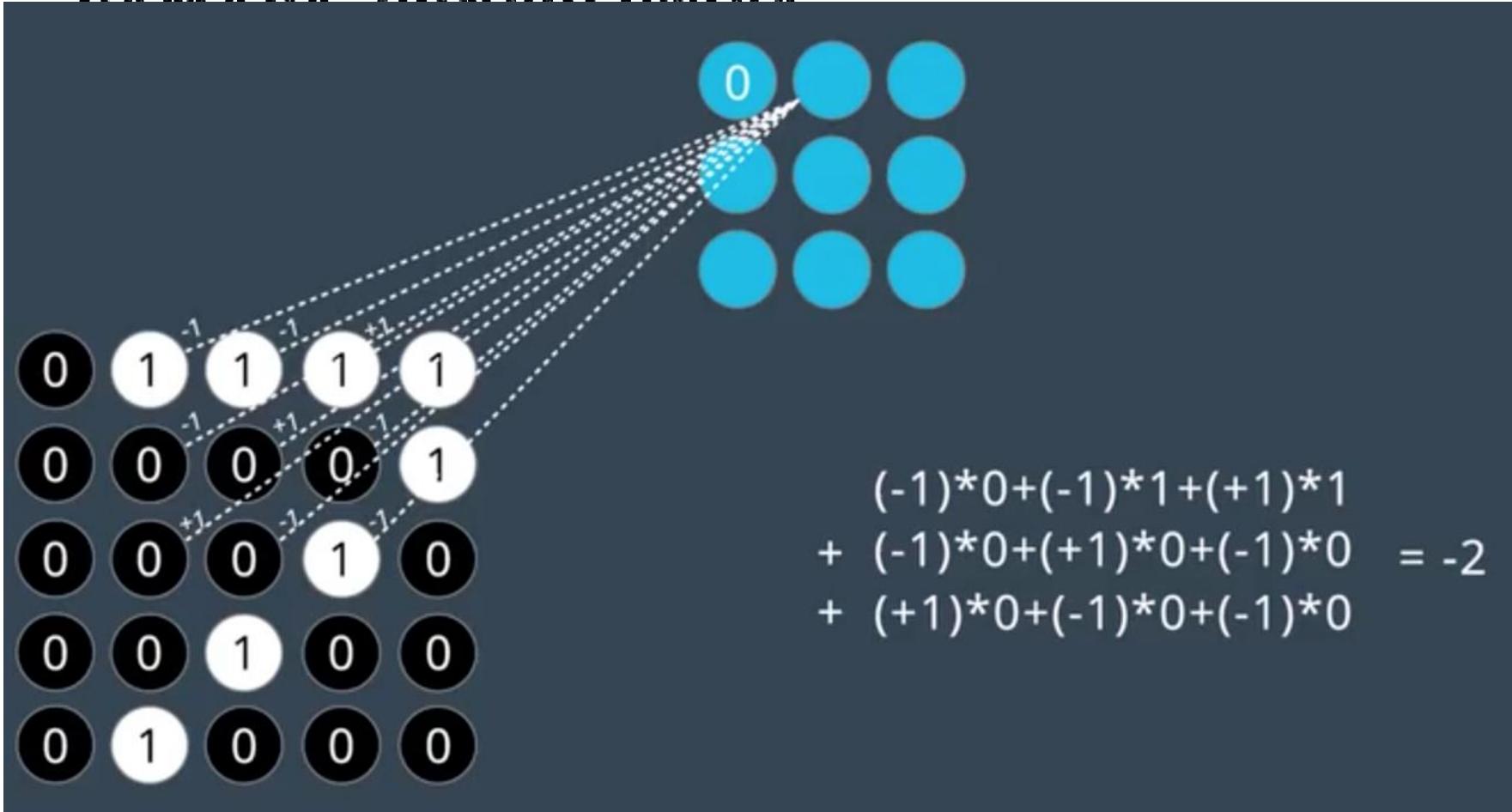
CLIFFORD ALGEBRA TUTORIAL



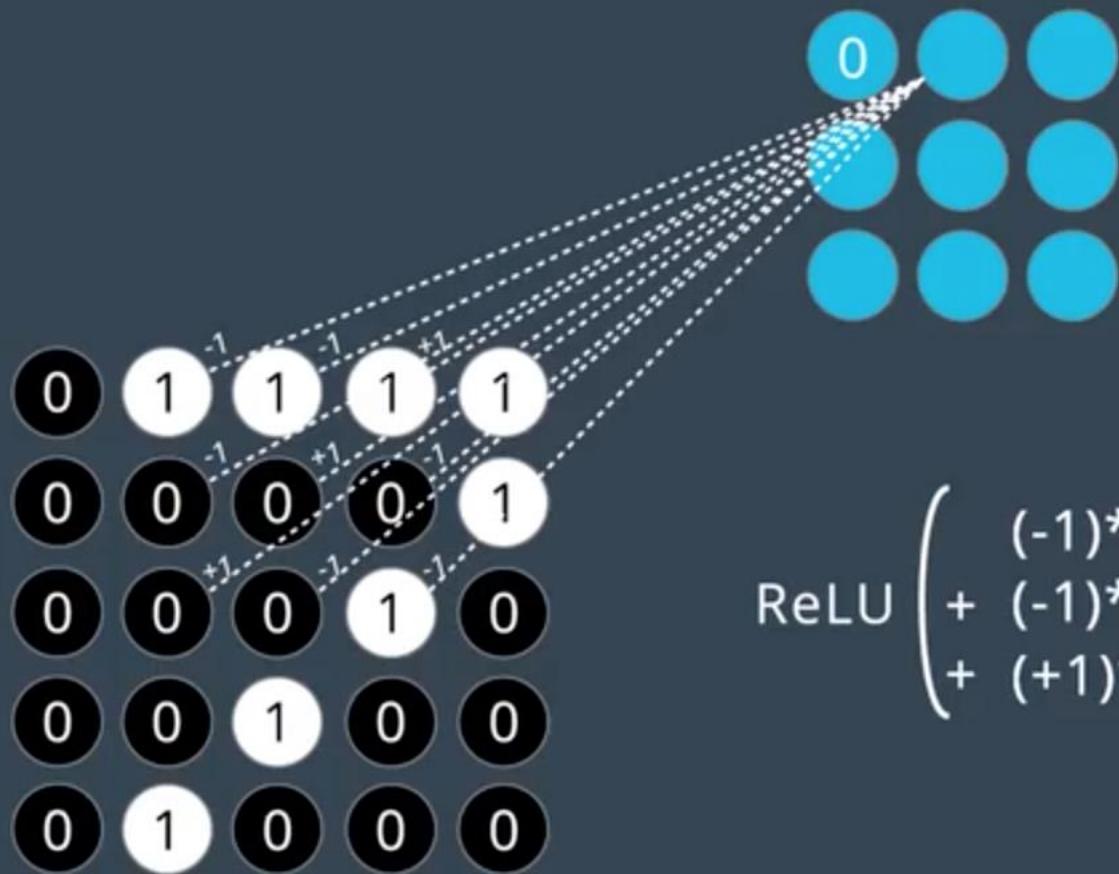
CAMADA CONVOLUTIVA



CLIMATE DATA CLASSIFICATION



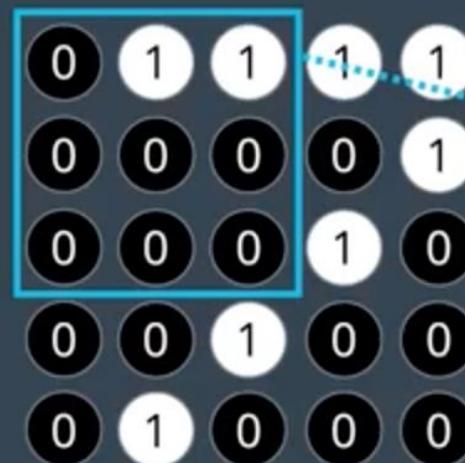
CAMADA CONVOLUTIVA



$$\text{ReLU} \left(\begin{array}{l} (-1)*0+(-1)*1+(+1)*1 \\ + (-1)*0+(+1)*0+(-1)*0 \\ + (+1)*0+(-1)*0+(-1)*0 \end{array} \right) = 0$$



Filter		
-1	-1	+1
-1	+1	-1
+1	-1	-1



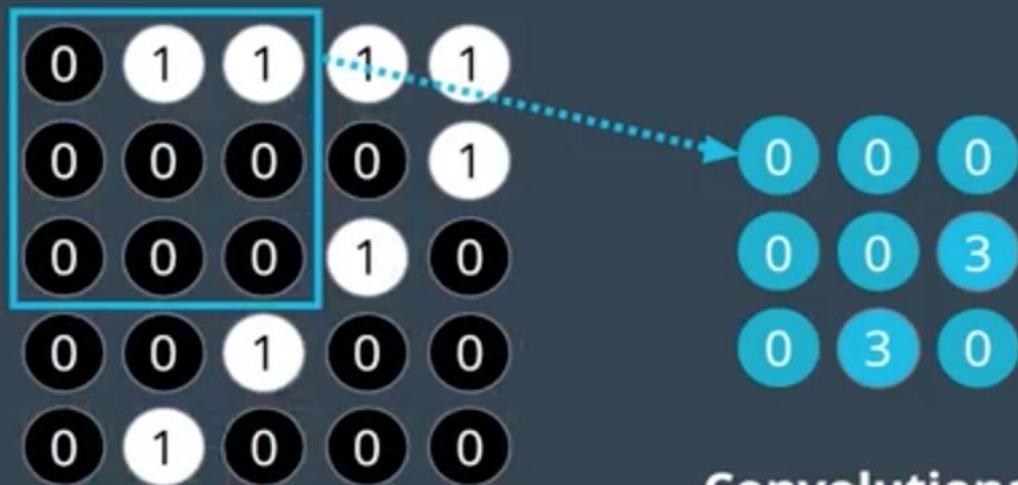
Convolutional
Layer

$$\text{ReLU} \left(\text{SUM} \left(\begin{array}{ccc} * & * & * \\ * & * & * \\ * & * & * \end{array} \right) \right) = 0$$



Filter

-1	-1	+1
-1	+1	-1
+1	-1	-1



Input Layer

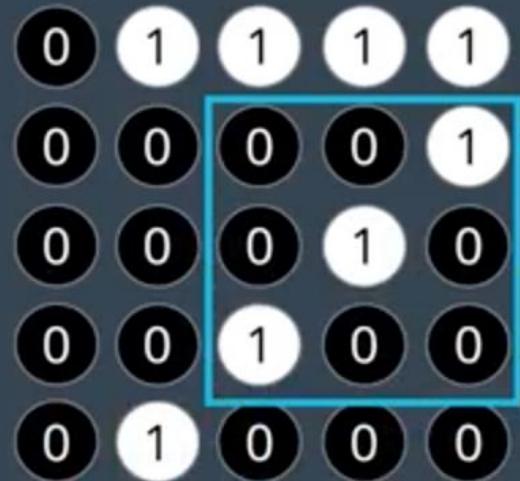
**Convolutional
Layer**

$$\text{ReLU} \left(\text{SUM} \left(\begin{array}{ccc} -1 * 0 & -1 * 1 & +1 * 1 \\ -1 * 0 & +1 * 0 & -1 * 0 \\ +1 * 0 & -1 * 0 & -1 * 0 \end{array} \right) \right) = 0$$



Filter

$$\begin{pmatrix} -1 & -1 & +1 \\ -1 & +1 & -1 \\ +1 & -1 & -1 \end{pmatrix}$$

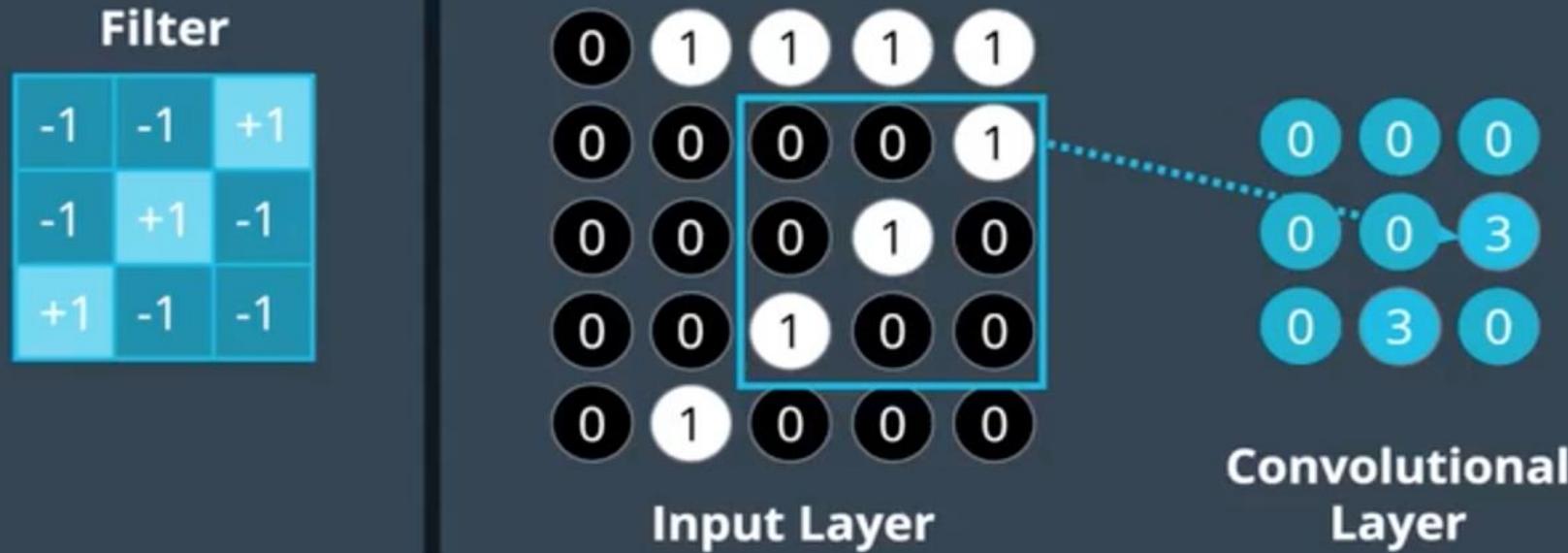


Convolutional Layer

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 3 \\ 0 & 3 & 0 \end{pmatrix}$$

ReLU SUM $\left(\begin{array}{ccc} -1 * 0 & -1 * 0 & +1 * 1 \\ -1 * 0 & +1 * 1 & -1 * 0 \\ +1 * 1 & -1 * 0 & -1 * 0 \end{array} \right) = 3$





$$\text{ReLU} \left(\text{SUM} \left(\begin{array}{ccc} -1 * 0 & -1 * 0 & +1 * 1 \\ -1 * 0 & +1 * 1 & -1 * 0 \\ +1 * 1 & -1 * 0 & -1 * 0 \end{array} \right) \right) = 3$$



Filter

$$\begin{bmatrix} -1 & -1 & +1 \\ -1 & +1 & -1 \\ +1 & -1 & -1 \end{bmatrix}$$



Convolutional Layer

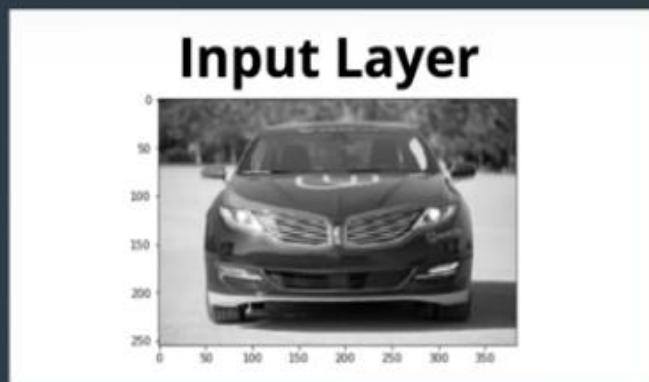
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 3 \\ 0 & 3 & 0 \end{bmatrix}$$

ReLU SUM

$$\text{ReLU} \left(\text{SUM} \left(\begin{array}{ccc} -1 * 0 & -1 * 0 & +1 * 1 \\ -1 * 0 & +1 * 1 & -1 * 0 \\ +1 * 1 & -1 * 0 & -1 * 0 \end{array} \right) \right) = 3$$



COLLEÇÃO DE FILTROS



Filter 1

-1	-1	+1	+1
-1	-1	+1	+1
-1	-1	+1	+1
-1	-1	+1	+1

Filter 2

+1	+1	-1	-1
+1	+1	-1	-1
+1	+1	-1	-1
+1	+1	-1	-1

Filter 3

-1	-1	-1	-1
-1	-1	-1	-1
+1	+1	+1	+1
+1	+1	+1	+1

Filter 4

+1	+1	+1	+1
+1	+1	+1	+1
-1	-1	-1	-1
-1	-1	-1	-1



Input Layer



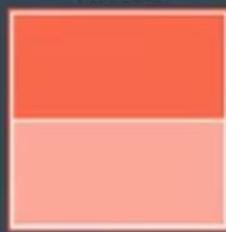
Filter 1



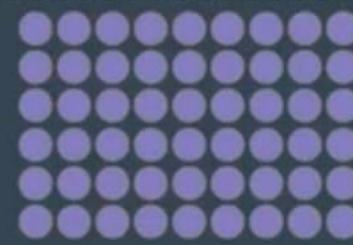
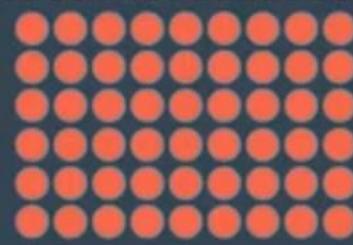
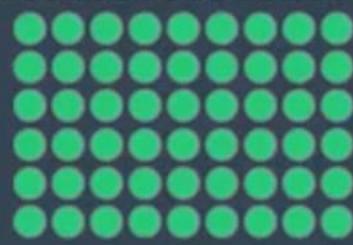
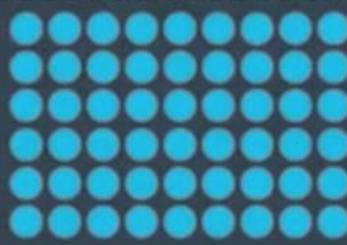
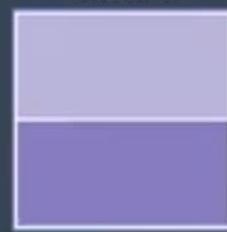
Filter 2



Filter 3



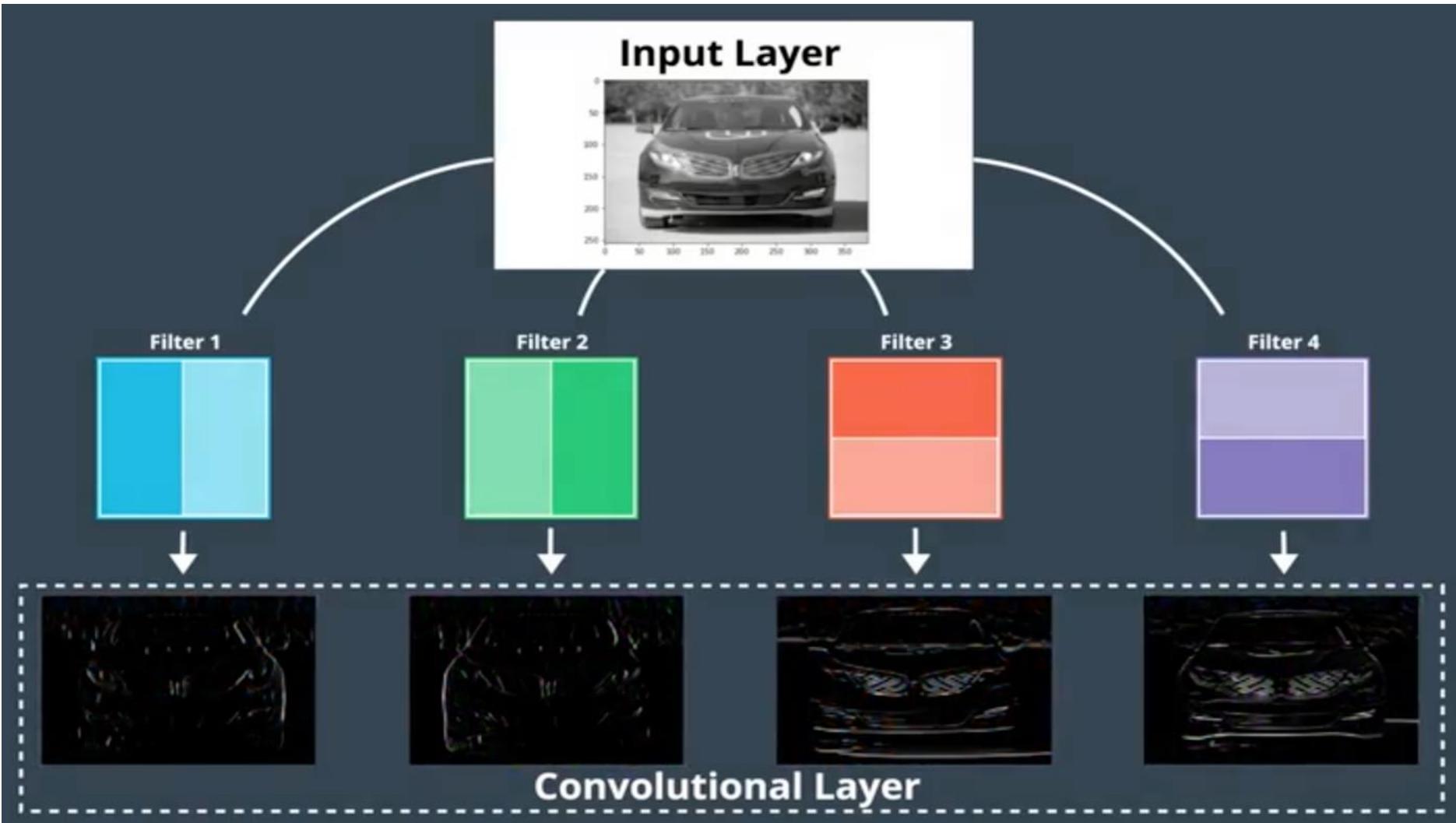
Filter 4



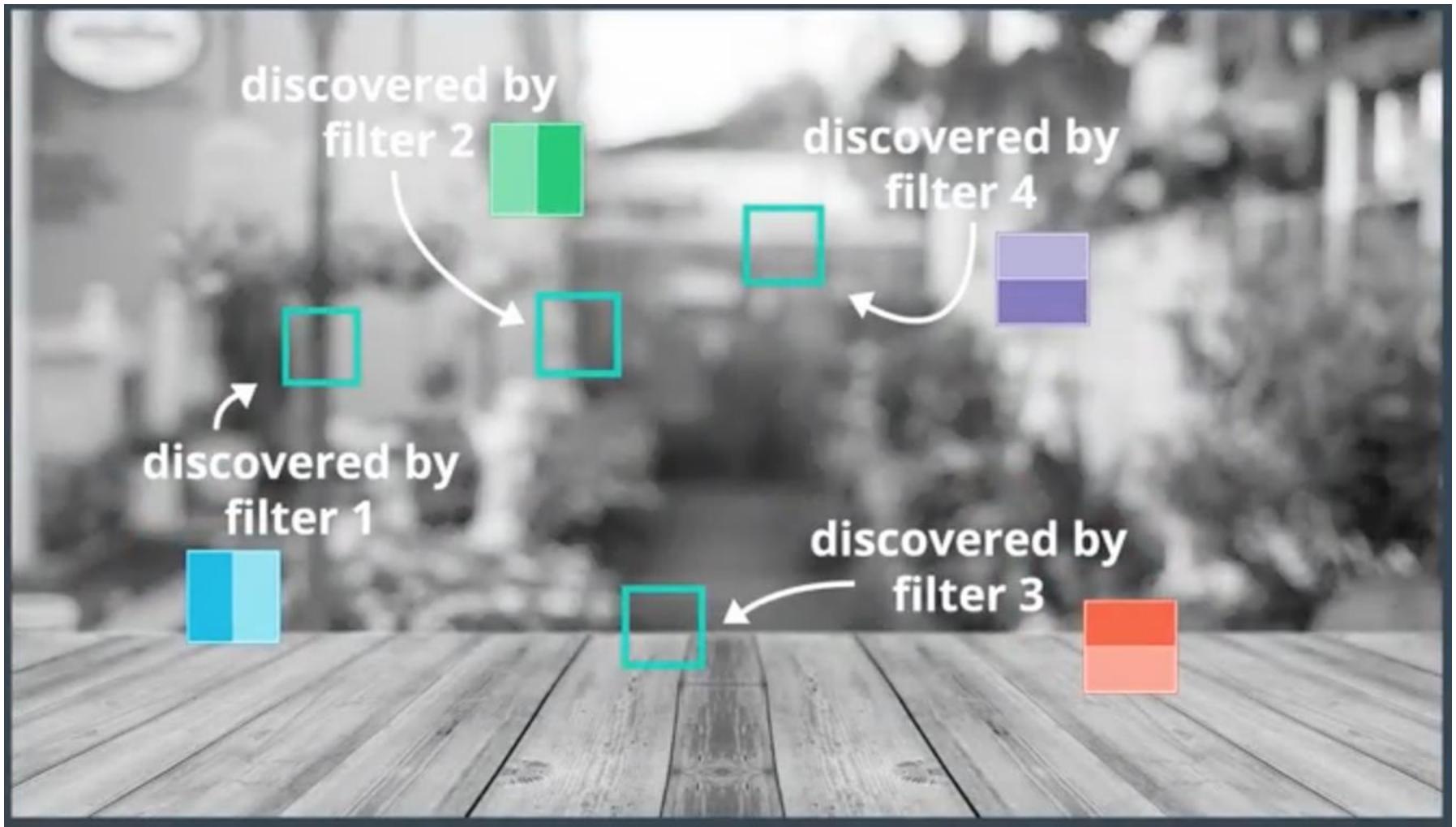
Convolutional Layer



IMAGENS FILTRADAS

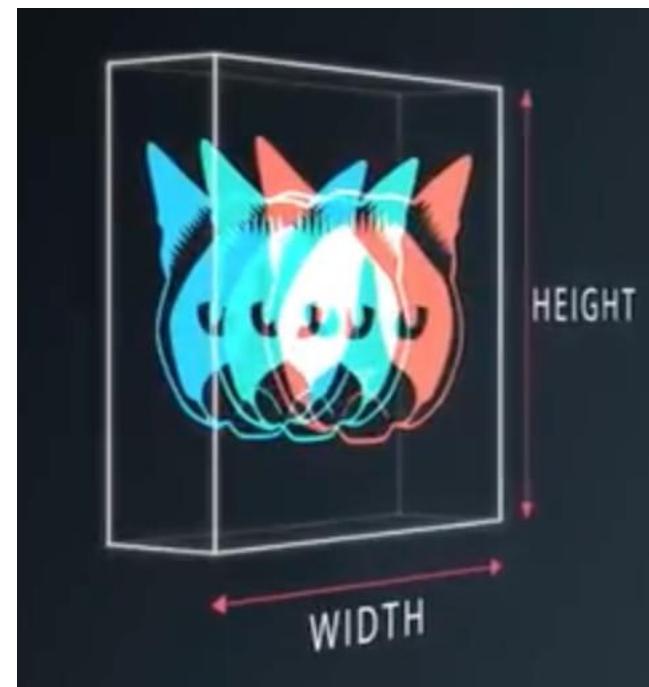


VÁRIAS REGIÕES DETECTADAS

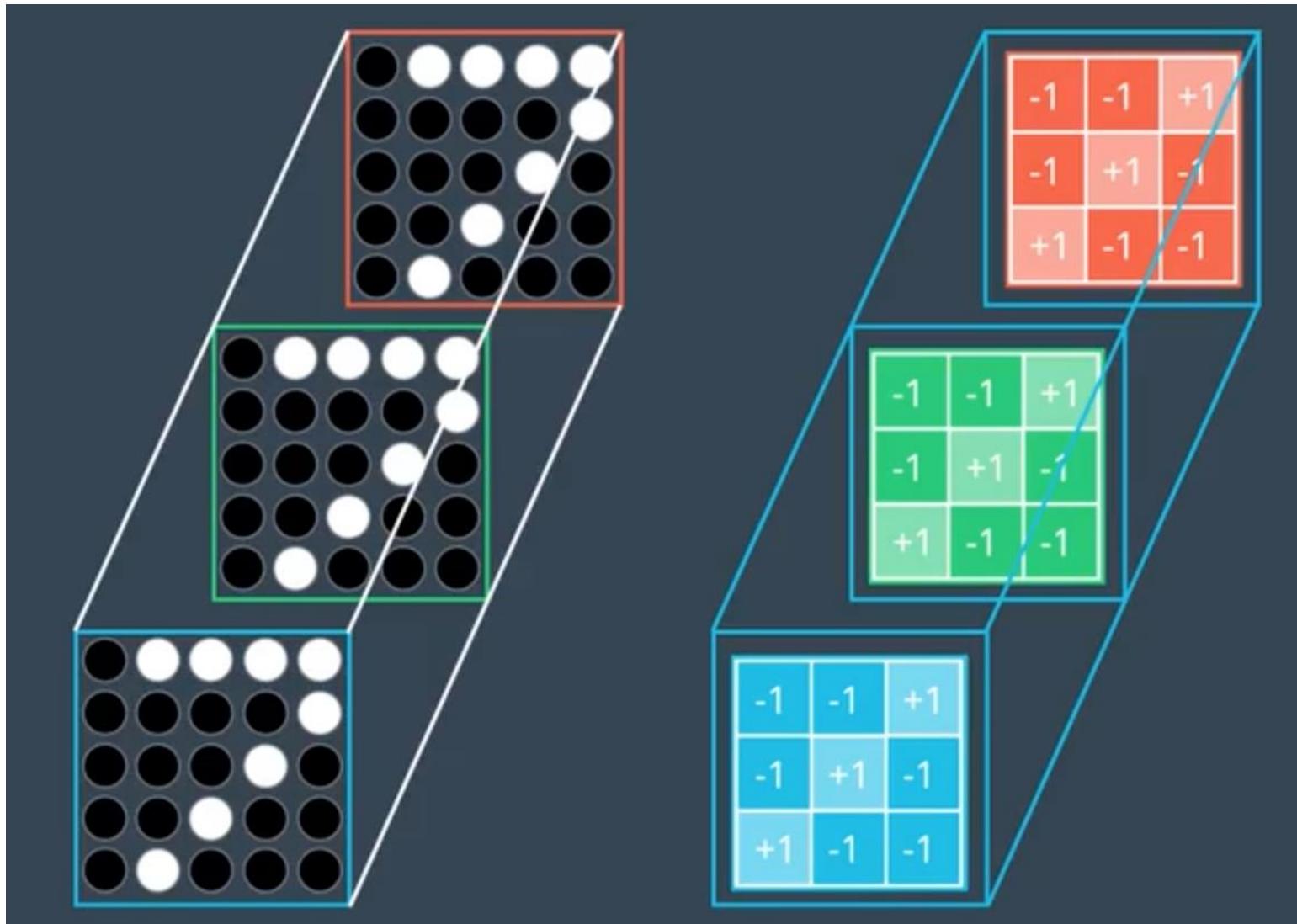


CONVOLUÇÃO EM IMAGENS COLORIDAS

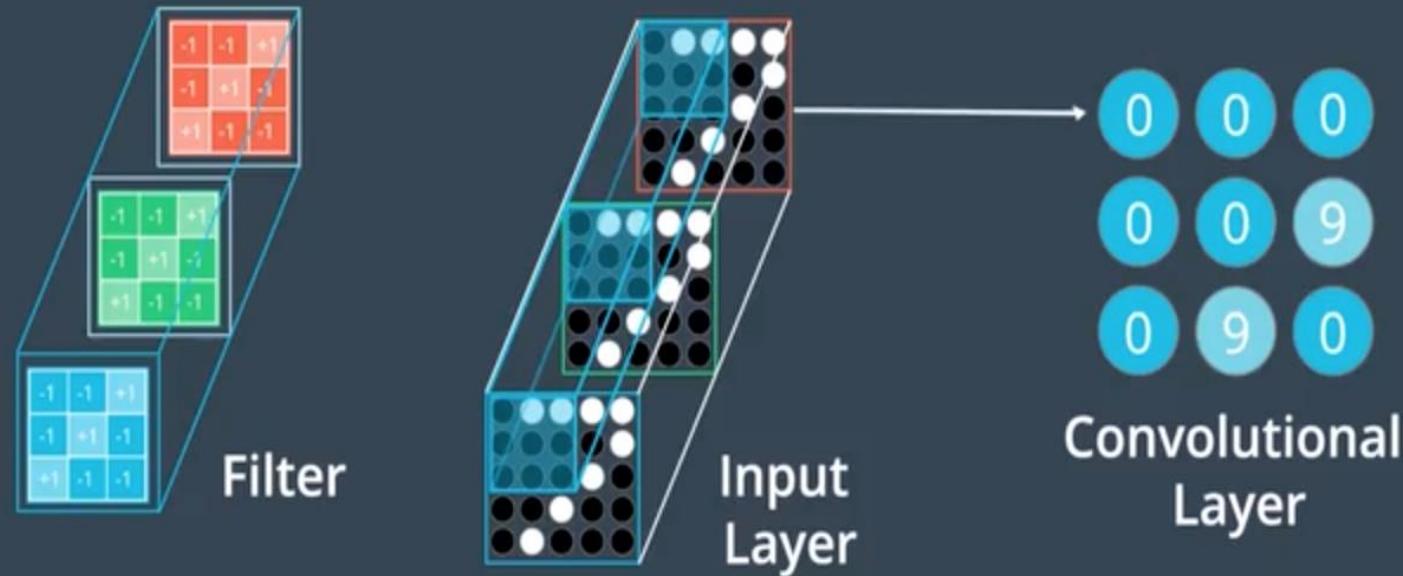
- Imagens RGB (3 dimensões)



CONVOLUÇÃO EM IMAGENS COLORIDAS

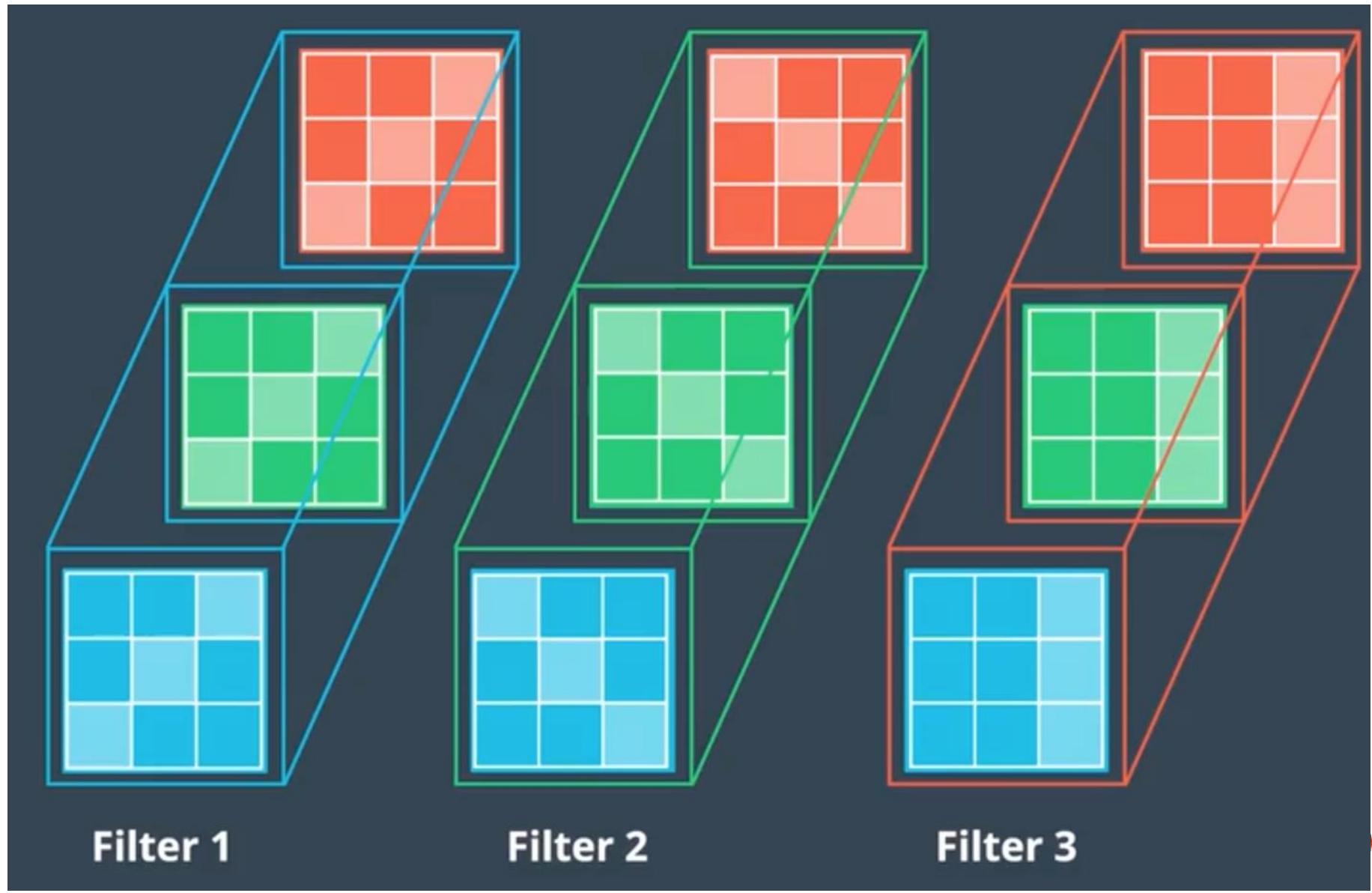


1 FILTRO COLORIDO

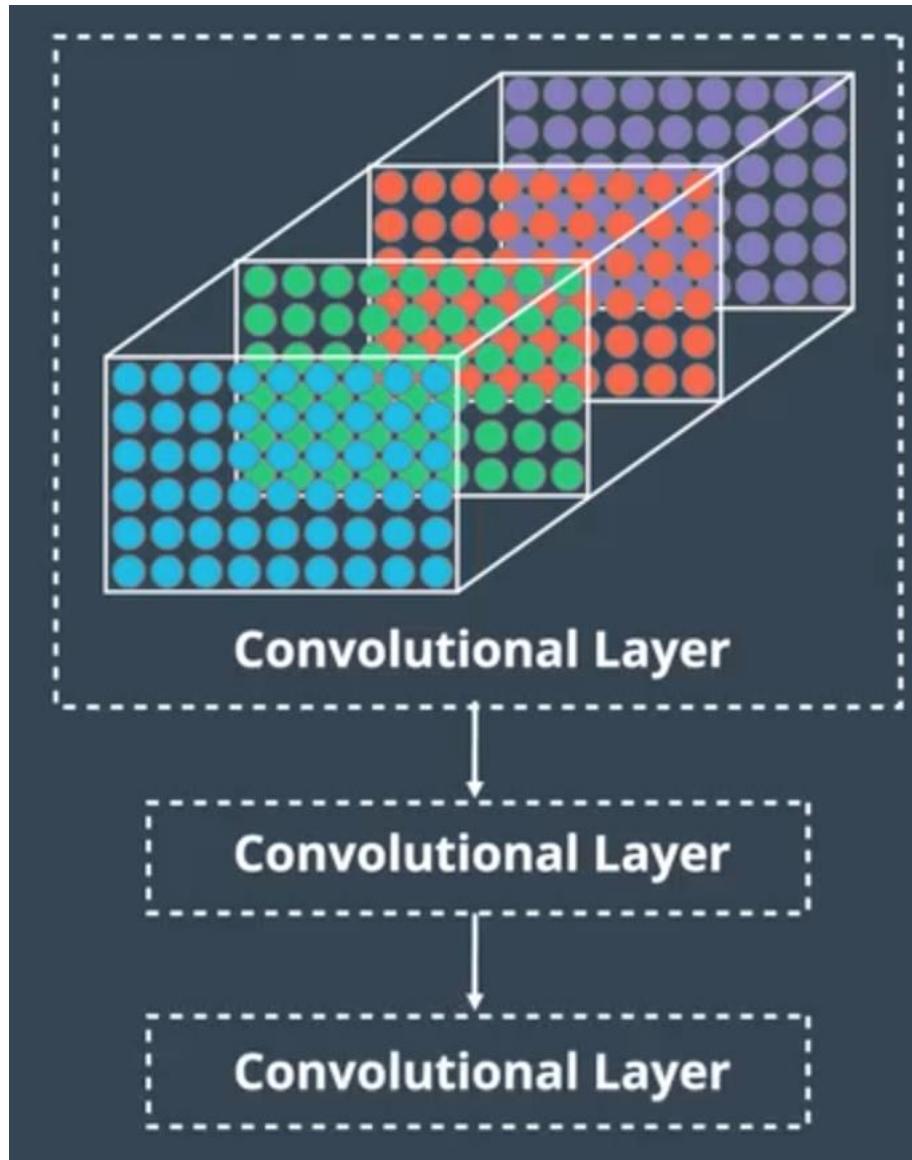


$$\text{ReLU} \left(\text{SUM} \left(\begin{array}{ccccccccc} \text{■} \times \text{○} & \text{■} \times \text{●} & \text{■} \times \text{●} & \text{■} \times \text{○} & \text{■} \times \text{●} & \text{■} \times \text{●} & \text{■} \times \text{○} & \text{■} \times \text{●} & \text{■} \times \text{●} \\ \text{■} \times \text{○} & \text{■} \times \text{○} & \text{■} \times \text{●} & \text{■} \times \text{○} & \text{■} \times \text{●} & \text{■} \times \text{●} & \text{■} \times \text{○} & \text{■} \times \text{●} & \text{■} \times \text{●} \\ \text{■} \times \text{○} & \text{■} \times \text{○} & \text{■} \times \text{●} & \text{■} \times \text{○} & \text{■} \times \text{●} & \text{■} \times \text{●} & \text{■} \times \text{○} & \text{■} \times \text{●} & \text{■} \times \text{●} \end{array} \right) \right)$$

3 FILTROS COLORIDOS



VÁRIAS CAMADAS CONVOLUCIONAIS



PROPRIEDADES DA CAMADA CONVOLUCIONAL

- Funciona da mesma forma que as camadas densas da MLP tradicional
- Os Filtros são iniciados de forma aleatória
- Seus pesos são aprendidos pela rede neural minimizando a função de custo
- Se uma imagem possui um cachorro, a rede neural aprenderá por si só, filtros que se parecem com um cachorro.
- Hiper Parâmetros:
 - Número de filtros
 - Largura x Altura desses filtros



OUTROS PARÂMETROS:

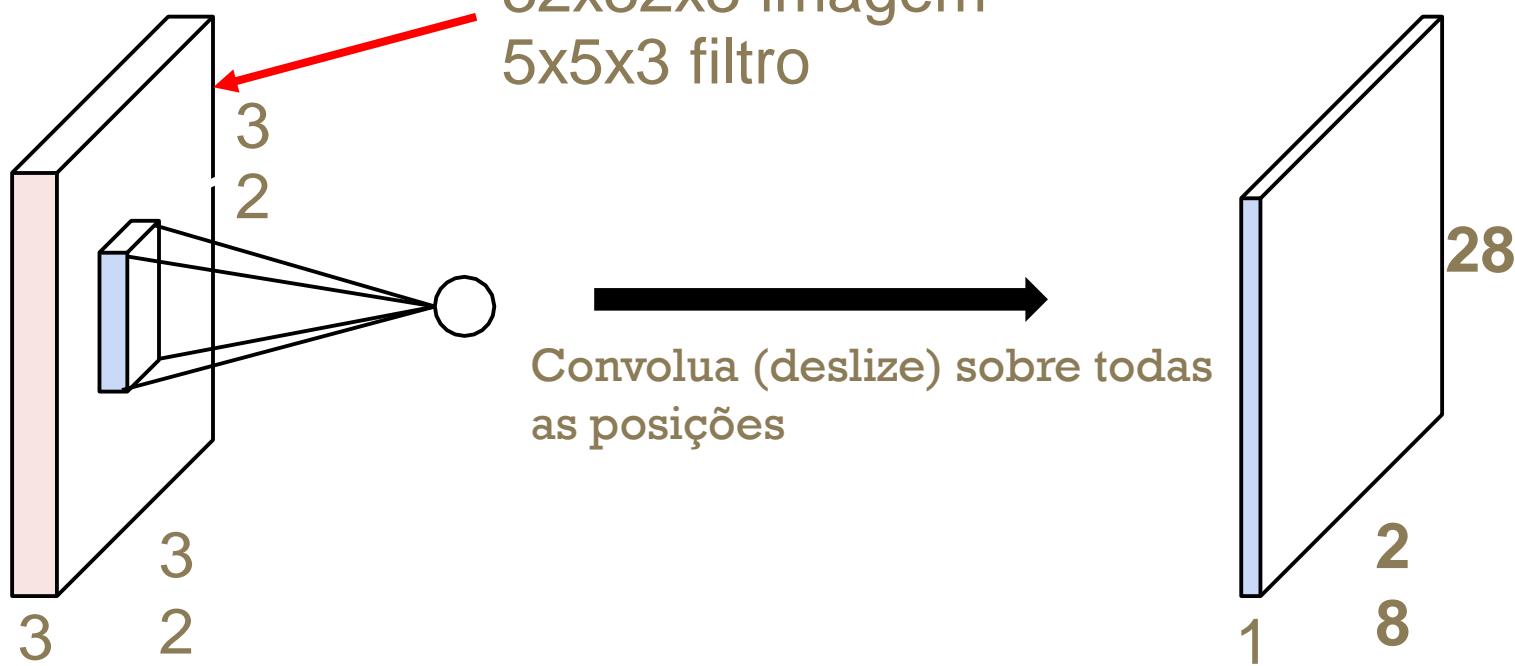
- Stride
- Padding
- Função de Ativação



STRIDE

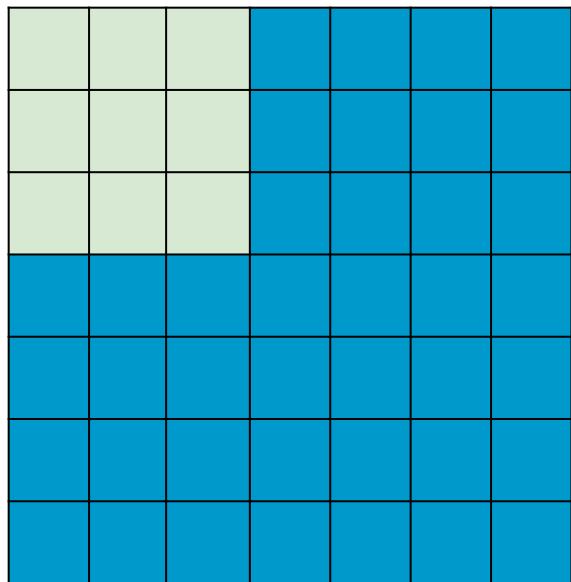
Olhando mais de perto as dimensões

Mapa de ativação



STRIDE

- Olhando mais de perto as dimensões:

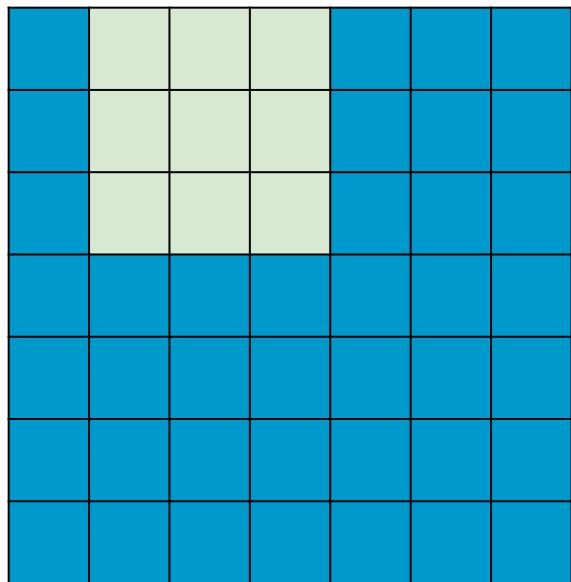


Entrada 7x7
Filtro 3x3



STRIDE

- Olhando mais de perto as dimensões:

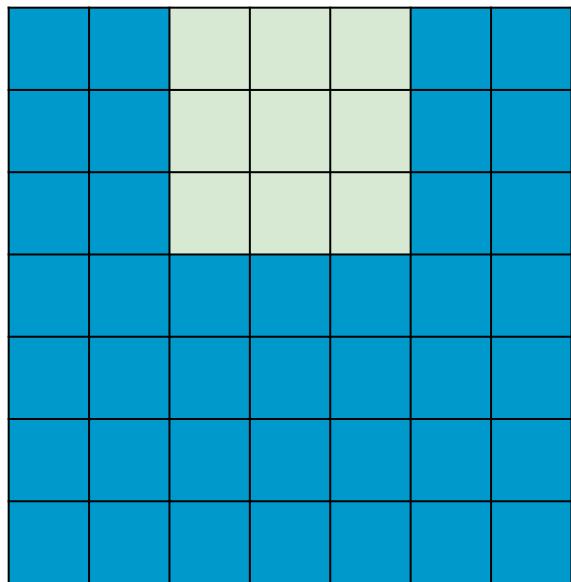


Entrada 7x7
Filtro 3x3



STRIDE

- Olhando mais de perto as dimensões:

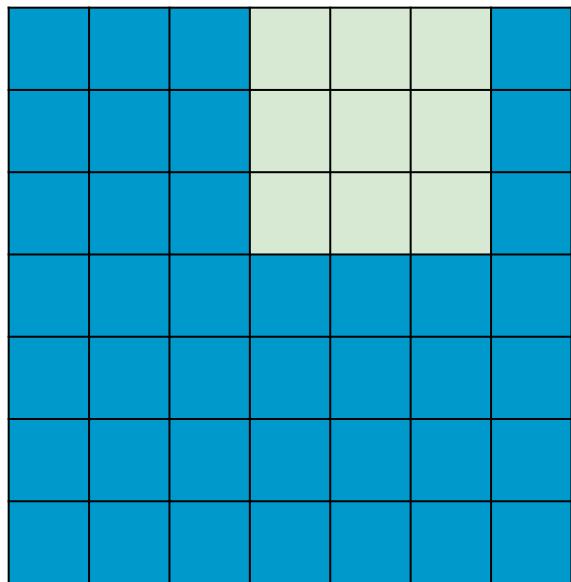


Entrada 7x7
Filtro 3x3



STRIDE

- Olhando mais de perto as dimensões:

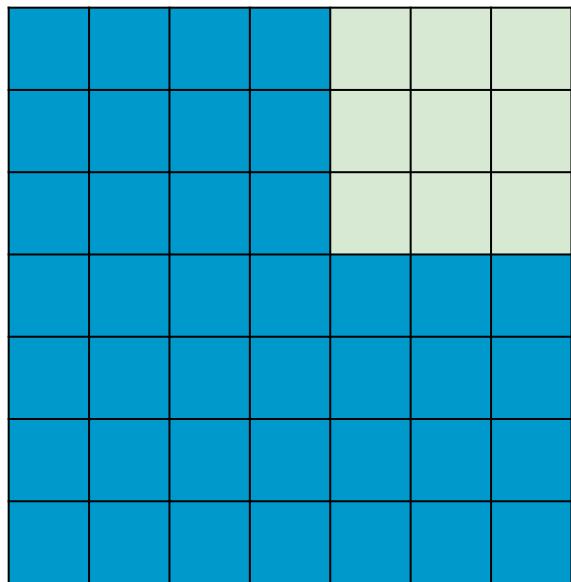


Entrada 7x7
Filtro 3x3



STRIDE

- Olhando mais de perto as dimensões:



Entrada 7×7

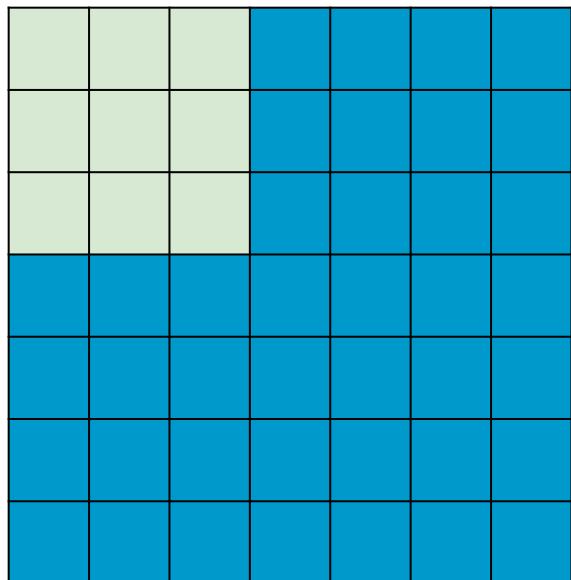
Filtro 3×3

Saída 5×5



STRIDE

- Olhando mais de perto as dimensões:



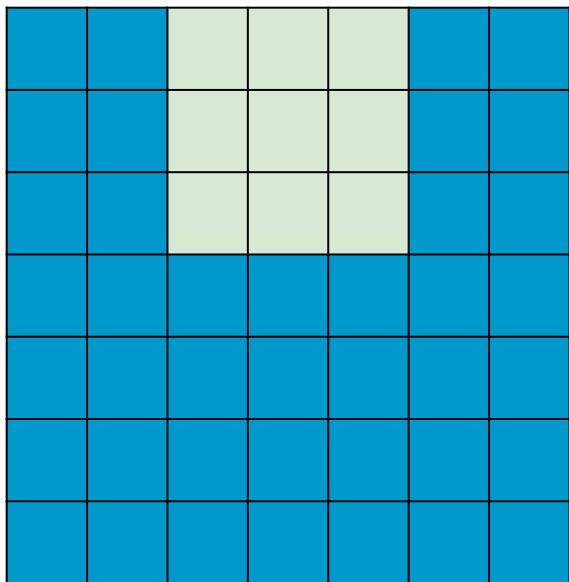
Entrada 7×7
Filtro 3×3

com stride 2 (stride = passo)



STRIDE

- Olhando mais de perto as dimensões:



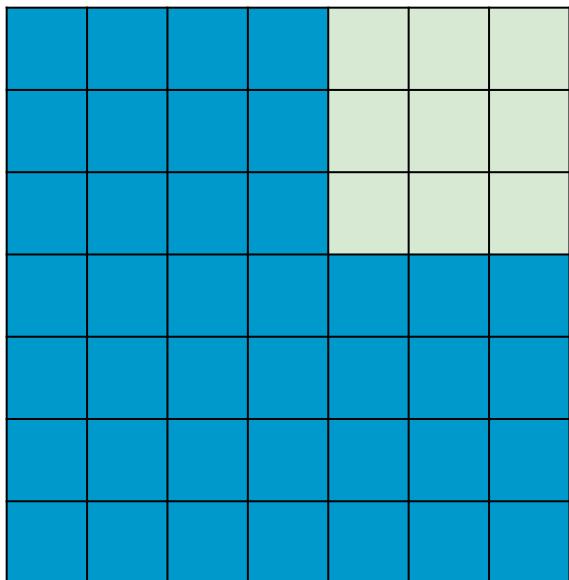
Entrada 7×7
Filtro 3×3

com stride 2 (stride = passo)



STRIDE

- Olhando mais de perto as dimensões:



Entrada 7×7
Filtro 3×3

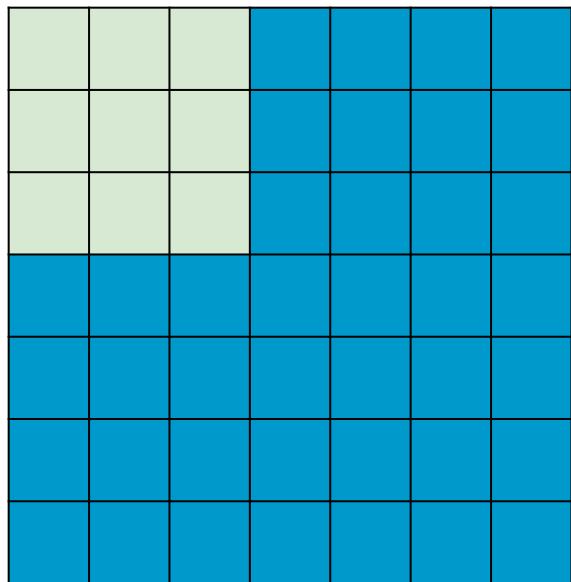
com stride 2 (stride = passo)

Saída 3×3



STRIDE

- Olhando mais de perto as dimensões:



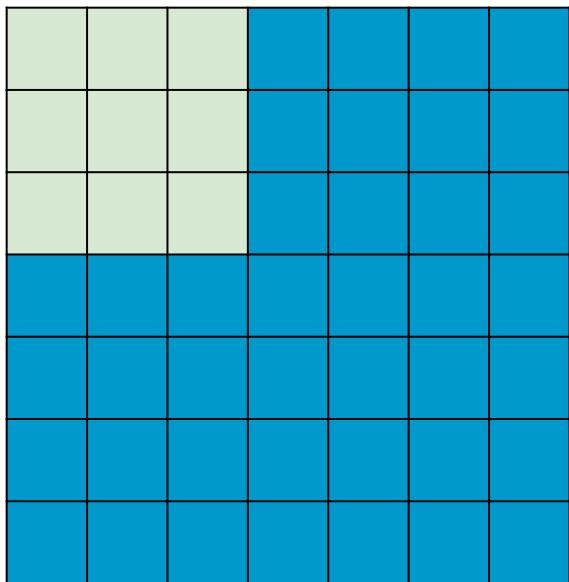
Entrada 7×7
Filtro 3×3

com stride 3 (stride = passo)



STRIDE

- Olhando mais de perto as dimensões:



Entrada 7×7

Filtro 3×3

com stride 3 (stride = passo)

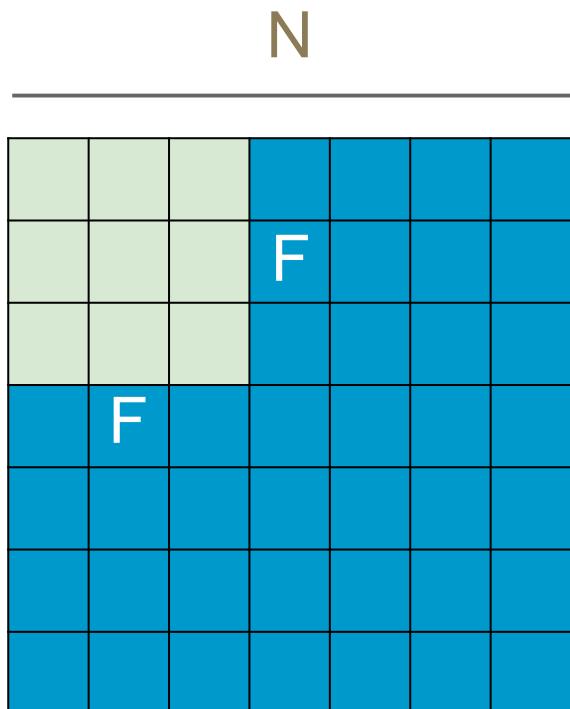
doesn't fit!

Você não consegue usar um filtro 3×3 em uma imagem 7×7 com stride 3.



STRIDE

- Olhando mais de perto as dimensões:



Calculando a saída:
(N - F) / stride + 1

N e.g. $N = 7, F = 3$:
stride 1 => $(7 - 3)/1 + 1 = 5$
stride 2 => $(7 - 3)/2 + 1 = 3$
stride 3 => $(7 - 3)/3 + 1 = 2.33$:\



PADDING

Na prática, costuma-se usar zero na borda

0	0	0	0	0	0		
0							
0							
0							
0							

e.g. Entrada 7x7

Filtro 3x3 com stride 1 pad com 1 pixel de borda

→ Qual é a saída?

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

Também chamado de padding



PADDING

Na prática, costuma-se usar zero na borda

0	0	0	0	0	0		
0							
0							
0							
0							

e.g. Entrada 7x7
Filtro 3x3 com stride 1
pad com 1 pixel de borda

→ Qual é a saída?

Saída 7 x 7

$$(N - F) / \text{stride} + 1 \rightarrow (9 - 3) / 1 + 1 = 7$$

$$(N + 2 * \text{padding} - F) / \text{stride} + 1$$



CAMADA CONVOLUCIONAL

- Para um volume de entrada $W \times H \times D$, um filtro de tamanho F , com passo S e *pad* P , a quantidade de neurônios em uma fatia do volume de saída é
$$(W - F + 2P)/S + 1$$
- Exemplos:
 - Com entrada 7×7 e filtro 3×3 com passo 1 e *pad* 0, a saída teria 5×5 neurônios por fatia.
 - Com $S=2$, teríamos saída com 3×3 neurônios por fatia.

Fonte: <http://cs231n.github.io/convolutional-networks/>

RESUMO: CAMADA CONVOLUCIONAL

- Aceita volume do tamanho $W_1 \times H_1 \times D_1$
- Requer 4 hiperparâmetros:
 - Número de filtros **K**,
 - Tamanho do Filtro **F**,
 - Stride **S**,
 - Quantidade de zero padding **P**.
- Produz um volume $W_2 \times H_2 \times D_2$ onde:
 - $W_2 = \frac{W_1 - F + 2P}{S} + 1$
 - $H_2 = \frac{H_1 - F + 2P}{S} + 1$
 - $D_2 = K$
- $F \cdot F \cdot D_1$ pesos por filtro. Total de $(F \cdot F \cdot D_1)$ pesos e **K** biases;

Configurações comuns:

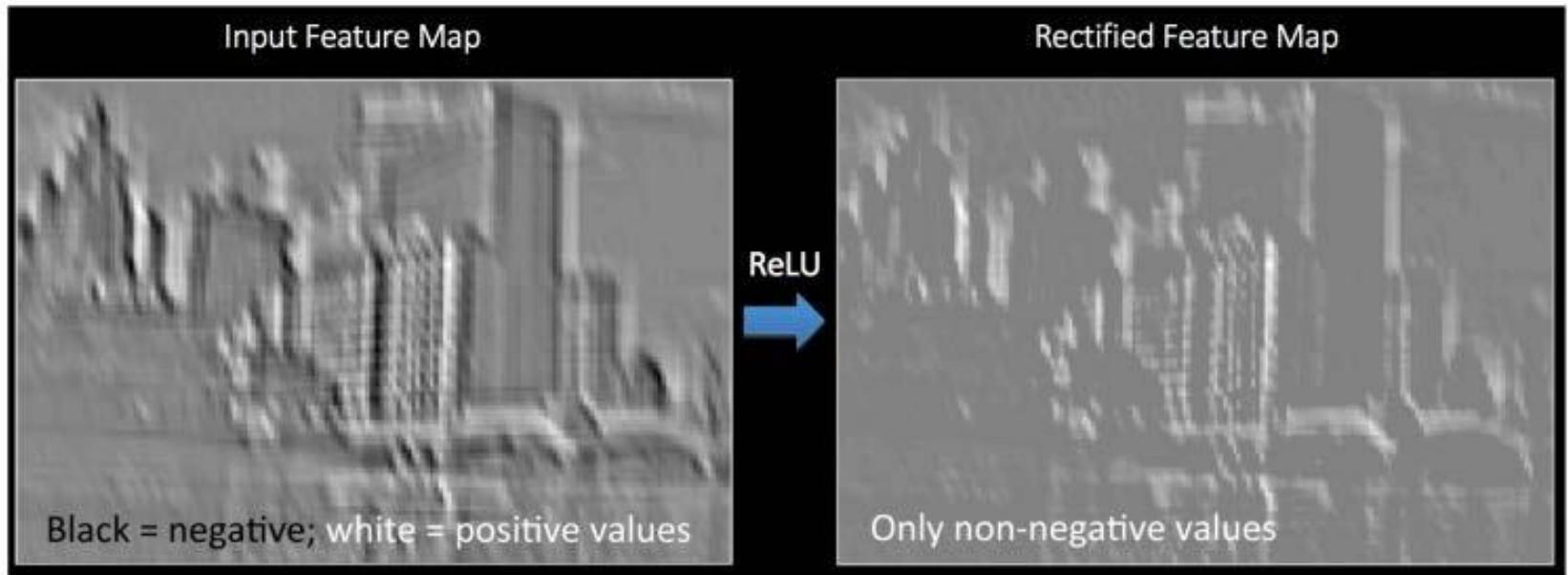
K= (potência de 2. Ex: 32,64,128,512)
- $F=3, S=1, P=1$
- $F=5, S=1, P=2$
- $F=1, S=1, P=0$



RELU (RECTIFIED LINEAR UNITS)

- Função de ativação comumente utilizada após uma camada de convolução
- Objetiva introduzir não linearidade
- Utilizada no lugar de “tanh” e “sigmoid” pois treina mais rápido e não interfere muito na acurácia
- Aplica a função $f(x) = \max(0, x)$
 - Transforma valores negativos em zero

ReLU



Somente valores positivos



RELU

0.77	-0.11	0.11	0.33	0.55	-0.11	0.33
-0.11	1.00	-0.11	0.33	-0.11	0.11	-0.11
0.11	-0.11	1.00	-0.33	0.11	-0.11	0.55
0.33	0.33	-0.33	0.55	-0.33	0.33	0.33
0.55	-0.11	0.11	-0.33	1.00	-0.11	0.11
-0.11	0.11	-0.11	0.33	-0.11	1.00	-0.11
0.33	-0.11	0.55	0.33	0.11	-0.11	0.77

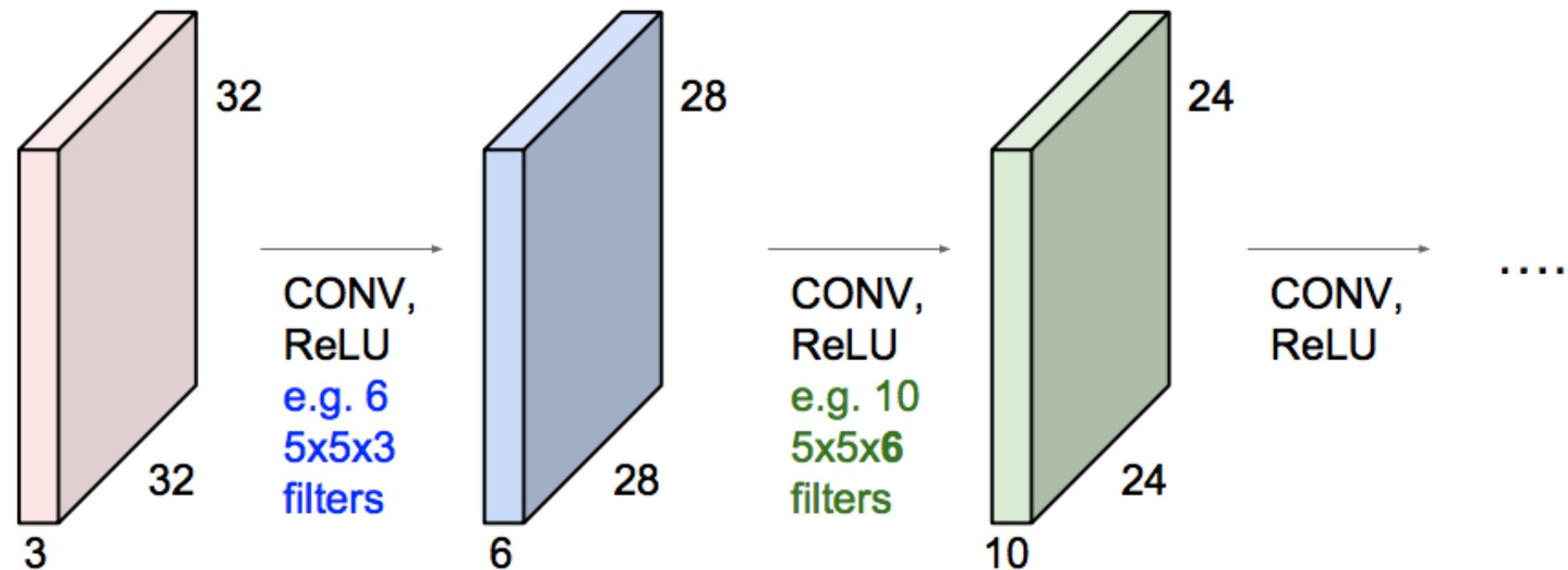


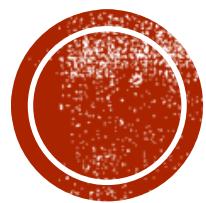
0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	0.77



CAMADAS CONVOLUTIVAS

- ConvNet é uma sequencia de Camadas Convolutivas, intercaladas com funções de ativação





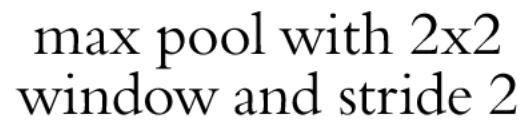
POOLING



CAMADA DE POOLING

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

max pool with 2x2
window and stride 2



6	8
3	4



CAMADA DE POOLING – OUTRO EXEMPLO

$$(S = 2, F = 2, W_1 = 4, H_1 = 4)$$

21	8	8	12
12	19	9	7
8	10	4	3
18	12	9	10

$$W_2 = 2, H_2 = 2$$

avg pooling

15	9
12	7

max pooling

21	12
18	10

Repare que 75% das ativações são descartadas!

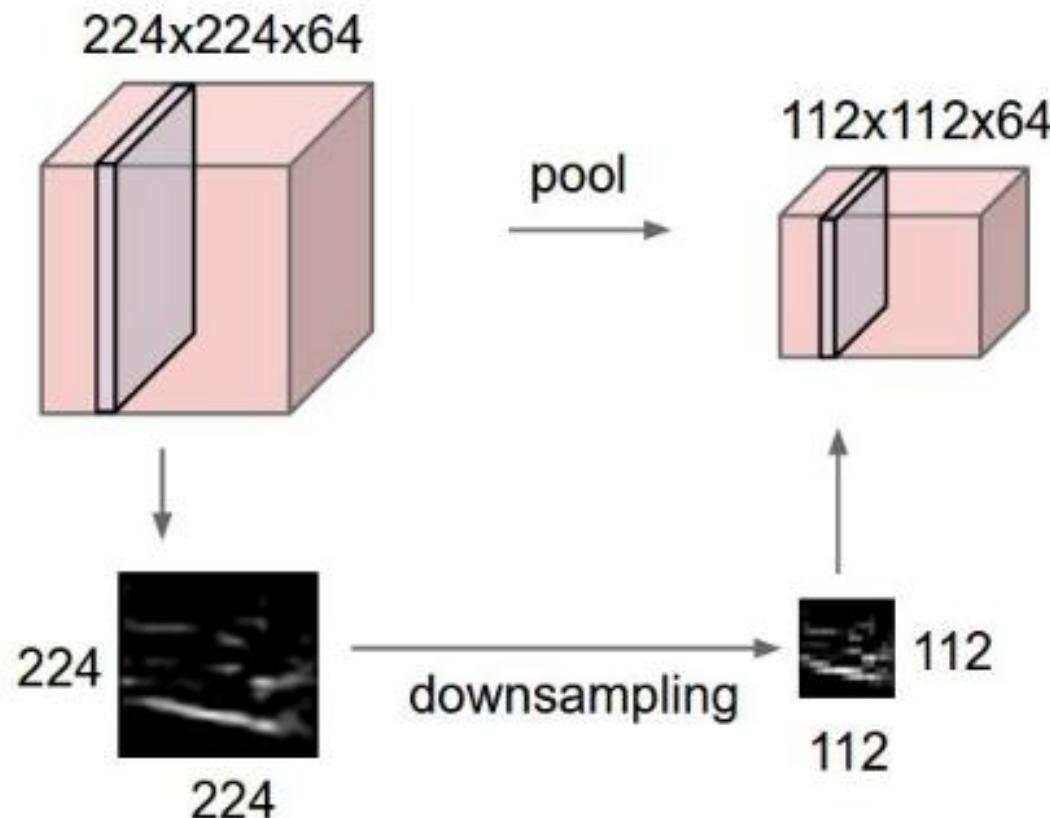
$$W_2 = (W_1 - F)/S + 1$$

$$H_2 = (H_1 - F)/S + 1$$

Imagen reproduzida do material disponibilizado na disciplina cs231n (<http://cs231n.stanford.edu/>)

CAMADA DE POOLING

- Torna a representação menor
- Opera sobre cada mapa de ativação independentemente.



RESUMO DE POOLING

Configurações comuns:
 $F=2, S=2$
 $F=3, S=2$

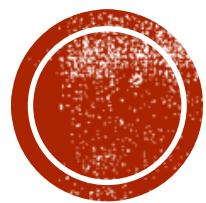
- Aceita um volume de tamanho $W_1 \times H_1 \times D_1$
- Requer três hiperparâmetros:
 - Tamanho da janela \mathbf{F} ,
 - Stride \mathbf{S}
- Produz um volume de tamanho $W_2 \times H_2 \times D_2$, onde:
 - $W_2 = \frac{W_1 - F}{S} + 1$
 - $H_2 = \frac{H_1 - F}{S} + 1$
 - $D_2 = D_1$
- Introduz zero parâmetros
- Não é comum usar zero-padding para camadas de pooling.



POR QUÊ MAX POOLING?

- Reduz tamanho especial da representação;
- Reduz número de parâmetros drasticamente;
- Filtro 2x2 com stride=2 descarta 75% das ativações
- Controla overfitting.
- Provê invariância à translação.





FULLY-CONNECTED (FC)



FULLY CONNECTED

- Atua como um classificador
- Recebe uma entrada da camada anterior (Conv ou ReLu ou Pooling) e fornece um vetor N dimensional onde C é o número de classes de prováveis saídas
- Exemplo: reconhecimento de dígitos
 - C = 10, pois são dígitos
 - Saída
[0 .1 .1 .75 0 0 0 0 0 .05]
10% ser o digito 1
10 % ser o digito 2
75% ser o digito 3
5% ser o digito 9



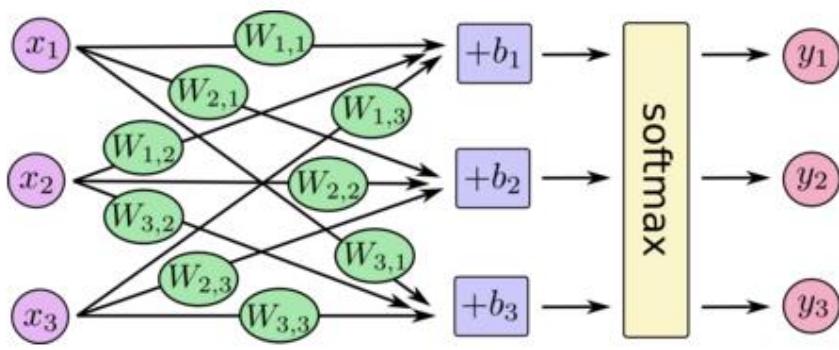
softmax approach



SOFTMAX

$$z_i = f(\text{net}_i) = \frac{e^{\text{net}_i}}{\sum_{c=1}^C e^{\text{net}_c}}$$

- Quando o propósito de uma RNA é realizar a tarefa de classificação, é comum usar a função **softmax**.



$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax} \begin{pmatrix} W_{1,1}x_1 + W_{1,2}x_2 + W_{1,3}x_3 + b_1 \\ W_{2,1}x_1 + W_{2,2}x_2 + W_{2,3}x_3 + b_2 \\ W_{3,1}x_1 + W_{3,2}x_2 + W_{3,3}x_3 + b_3 \end{pmatrix}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax} \left(\begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} \\ W_{2,1} & W_{2,2} & W_{2,3} \\ W_{3,1} & W_{3,2} & W_{3,3} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right)$$

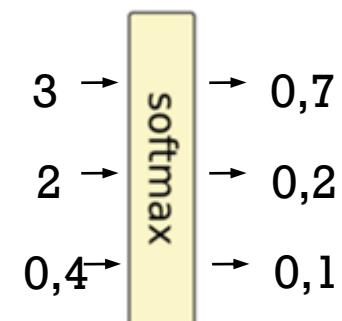
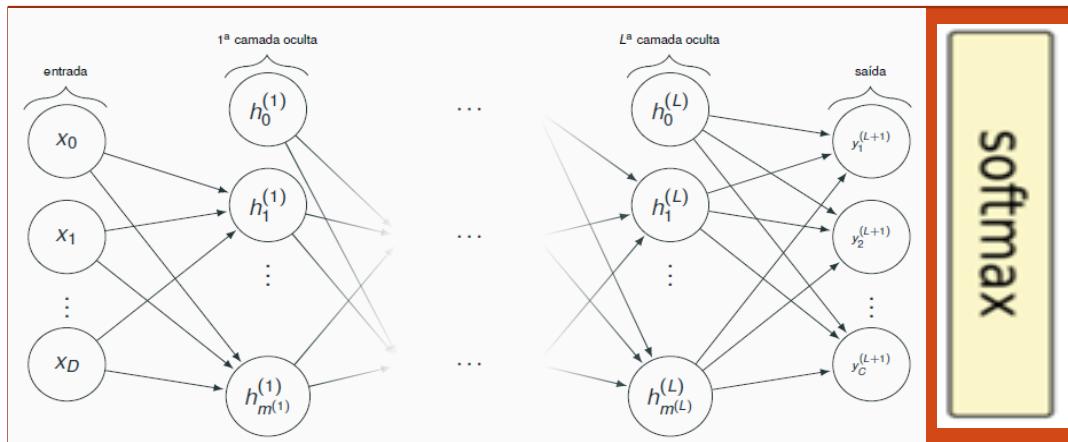
Fonte: <https://www.tensorflow.org/versions/r0.9/tutorials/mnist/beginners/index.html>



SOFTMAX

$$z_i = f(\text{net}_i) = \frac{e^{\text{net}_i}}{\sum_{c=1}^C e^{\text{net}_c}}$$

- Permite interpretar os valores da camada de saída como uma **distribuição de probabilidades**.



FULLY CONNECTED

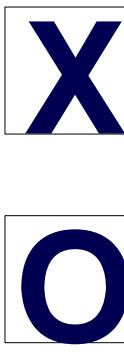
- Basicamente a camada utiliza o mapa de ativação da camada anterior com os maiores valores das características
- Exemplo:
 - se o programa visa identificar um cachorro, os mapas de ativação com maiores valores serão aqueles que representam as características de 4 patas, orelhas, etc.
 - se for de um pássaro os mapas de ativação serão aqueles com características que representam asas, bico, etc



JUNTANDO TUDO...

Um conjunto de pixels que se torna um conjunto de votos

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	-1	1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1



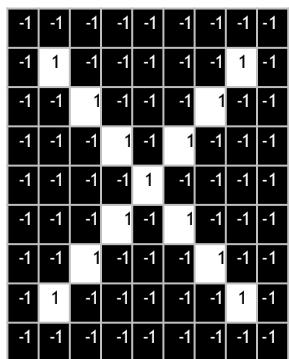
.92

.51



JUNTANDO TUDO...

Algoritmo de backpropagation utilizado para fazer os ajustes



.92

.51

	Right answer	Actual answer	Error
X	1	0.92	0.08
O	0	0.51	0.49
		Total	0.57

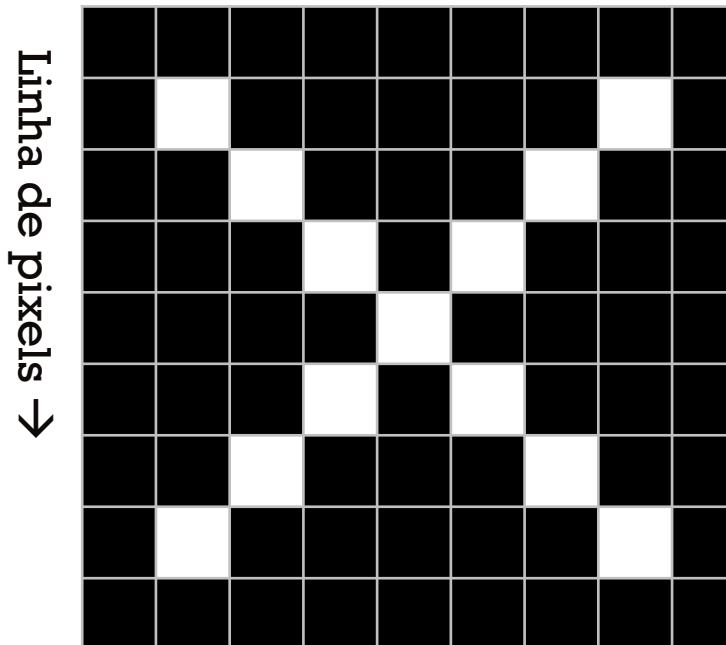


POSSIBILIDADES DE APLICAÇÕES



IMAGENS

Coluna de pixels →

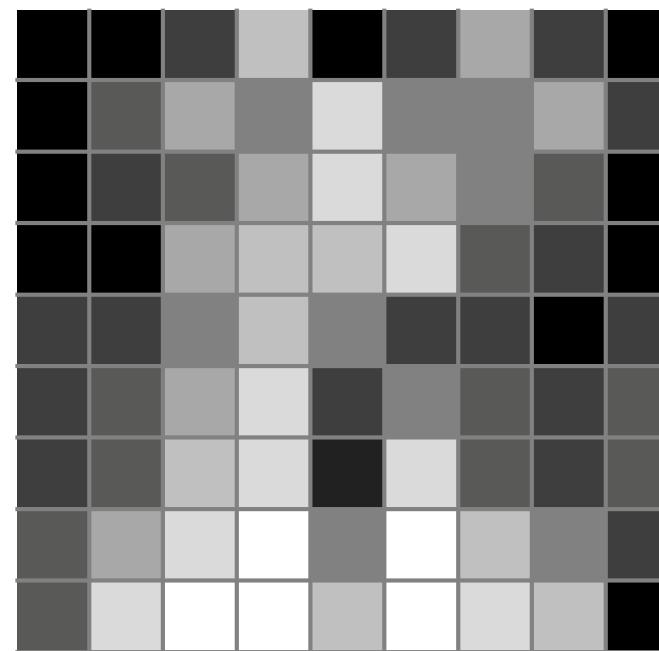


Linha de pixels →



SONS

Intervalos de tempo →

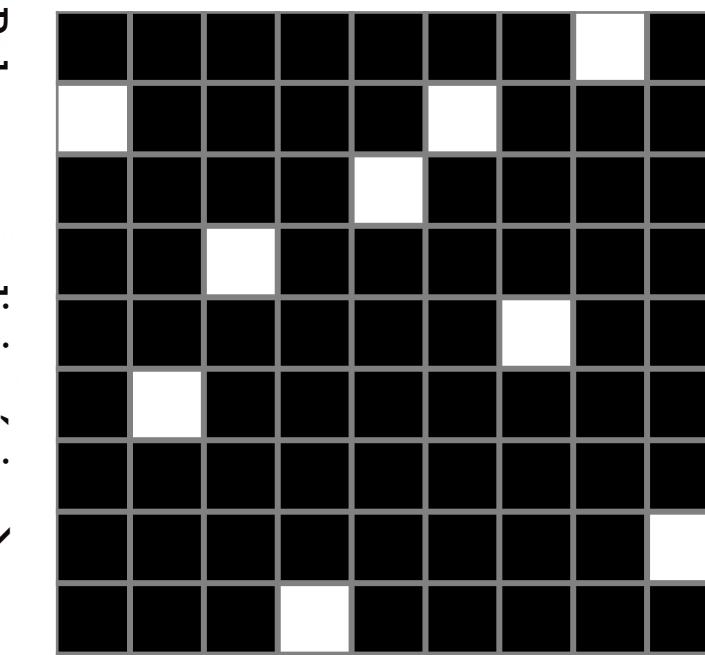


Intensidade em cada frequência de banda →



TEXTO

Posição na sentença →



Palavras no dicionário →



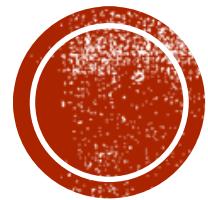
DADOS DE CLIENTES

Nome, idade, endereço, email,
Financeiro, atividade na internet, ... →

Clientes →

A	22	1A	<u>a@a</u>	1	aa	a1.a	123	aa1
B	33	2B	<u>b@b</u>	2	bb	b2.b	234	bb2
C	44	3C	<u>c@c</u>	3	cc	c3.c	345	cc3
D	55	4D	<u>d@d</u>	4	dd	d4.d	456	dd4
E	66	5E	<u>e@e</u>	5	ee	e5.e	567	ee5
F	77	6F	<u>f@f</u>	6	ff	f6.f	678	ff6
G	88	7G	<u>g@g</u>	7	gg	g7.g	789	gg7
H	99	8H	<u>h@h</u>	8	hh	h8.h	890	hh8
I	111	9I	<u>i@i</u>	9	ii	i9.i	901	ii9





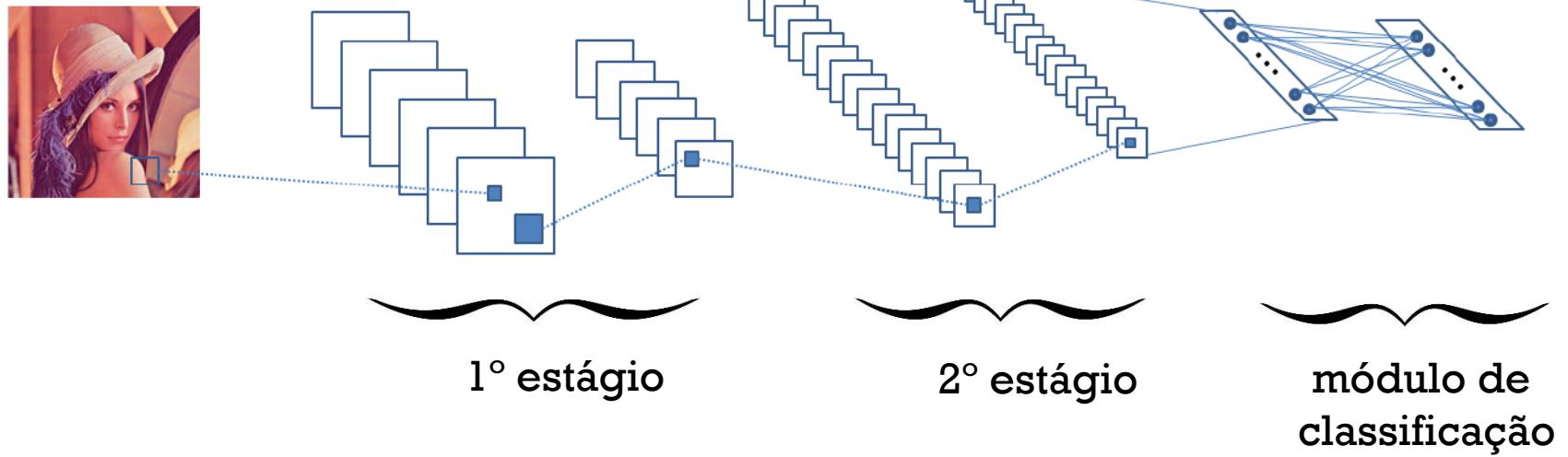
EXEMPLOS DE CNNS



ARQUITETURA TÍPICA

- Em uma CNN, encontramos um ou mais **estágios**, cada qual composto por camadas:
 - de convolução (CONV),
 - de subamostragem (POOL),
 - de normalização de contraste,
- Ao final, é comum encontrar uma ou mais camadas completamente conectadas seguida de camada softmax (módulo de classificação).

ARQUITETURA TÍPICA



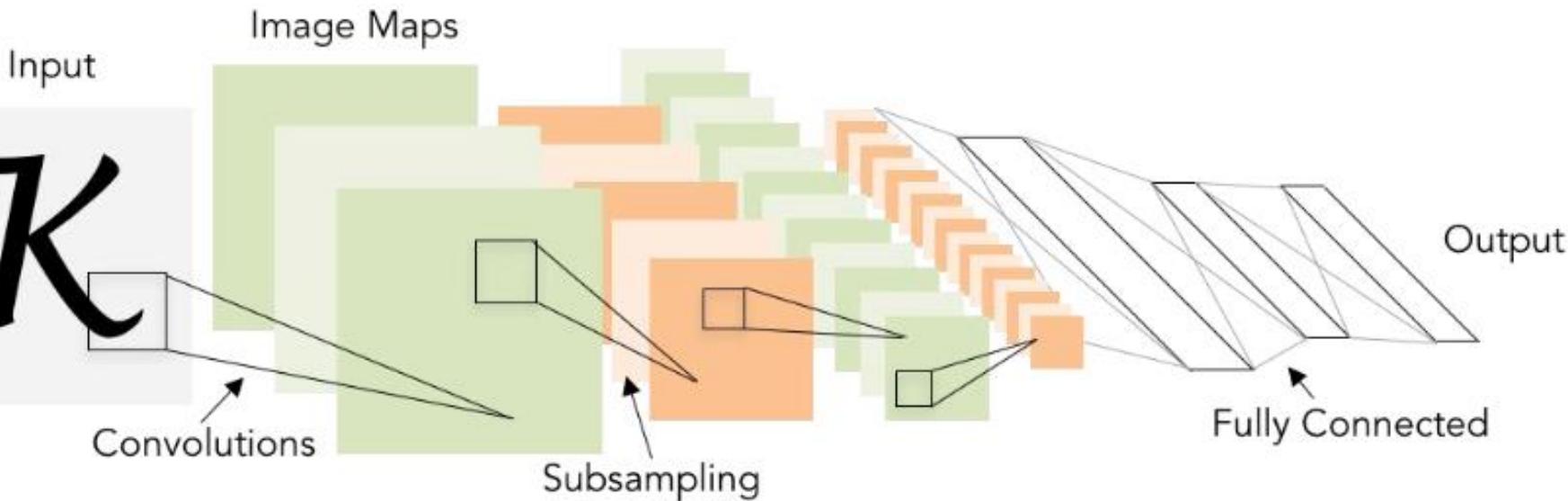
EXEMPLOS DE CNNS

- Estudos de Caso
 - Lenet
 - AlexNet
 - VGG
 - GoogLeNet
 - ResNet
- Existe Também
 - NiN (Network in Network)
 - Wide ResNet
 - ResNetXT
 - Stochastic Depth
 - DenseNet
 - FractalNet
 - SqueezeNet



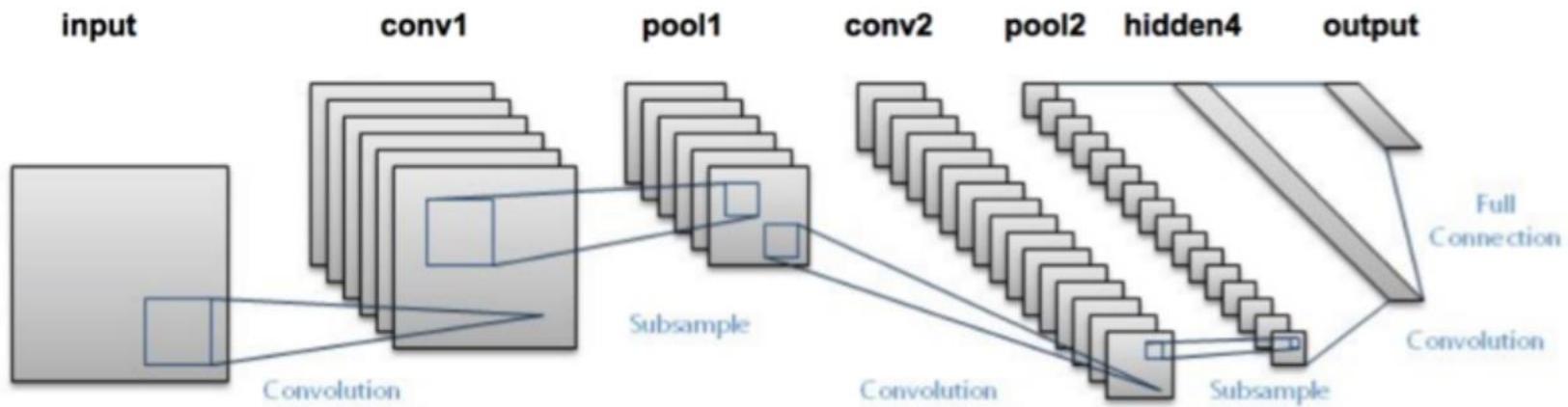
LENET-5

- Yan Le Cun et al, Convolutional Networks and Applications in Vision (2010)



- Filtros convolutivos 5×5 , aplicados com stride 1
- Camadas de pooling com filtro 2×2 aplicado com stride 2
- [CONV-POOL-CONV-POOL-FC-FC]





Layer Type	Output Size	Filter Size / Stride
INPUT IMAGE	$28 \times 28 \times 1$	
CONV	$28 \times 28 \times 20$	$5 \times 5, K = 20$
ACT	$28 \times 28 \times 20$	
POOL	$14 \times 14 \times 20$	2×2
CONV	$14 \times 14 \times 50$	$5 \times 5, K = 50$
ACT	$14 \times 14 \times 50$	
POOL	$7 \times 7 \times 50$	2×2
FC	500	
ACT	500	
FC	10	
SOFTMAX	10	

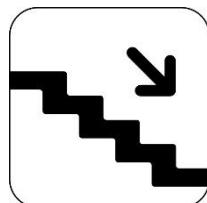
INPUT => CONV => TANH => POOL => CONV => TANH => POOL =>
FC => TANH => FC



ALEXNET, 2012



- Competição ILSVRC (*ImageNet Challenge*)
 - ~1,2M de imagens de alta resolução para treinamento;
 - ~1000 imagens por classe; 1000 classes!
 - Ganhou a edição 2012
 - Até então, soluções incluiam características produzidas manualmente, estudadas e melhoradas por décadas.



9%

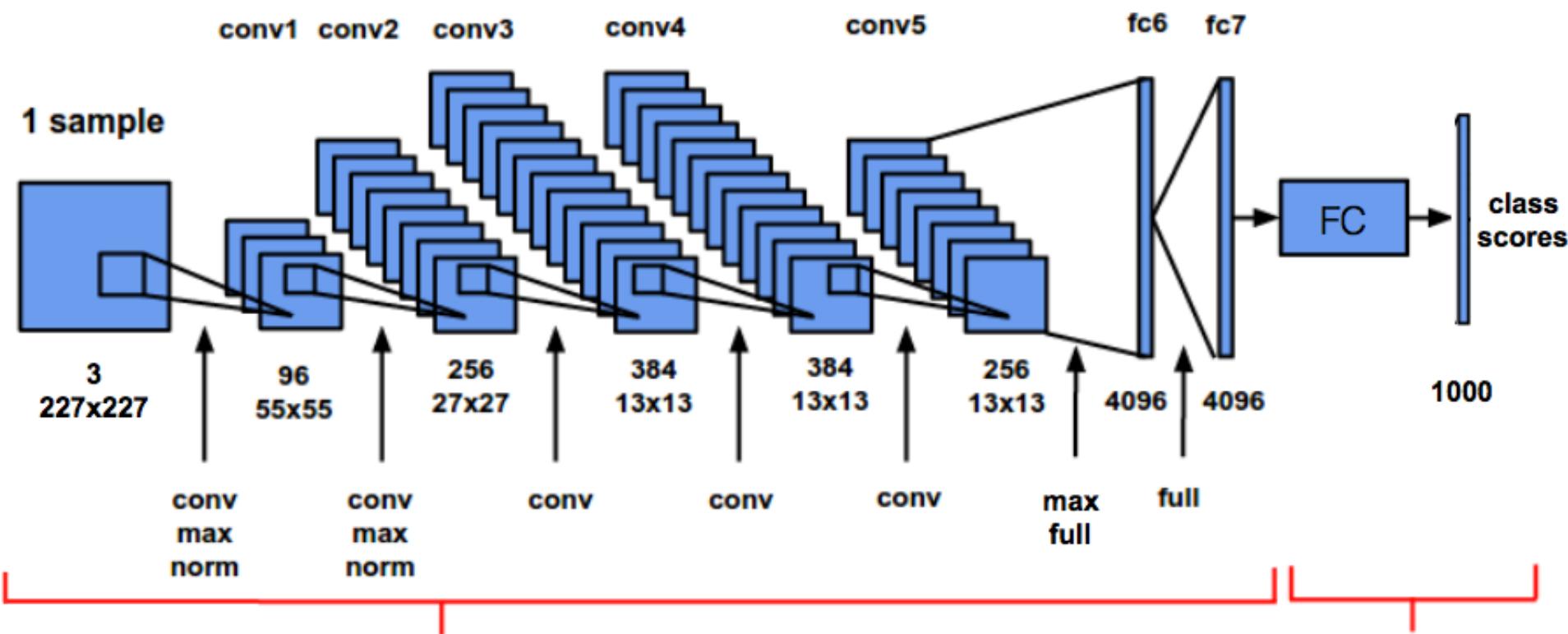
ImageNet Classification with L

Out/2016

[ImageNet Classification with Deep Convolutional Neural Networks](https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf)
by A Krizhevsky - 2012 - Cited by 12168 - Related

Out/2017

Example: AlexNet [Krizhevsky 2012]



Extract high level features

Classify each sample

“max”: max pooling

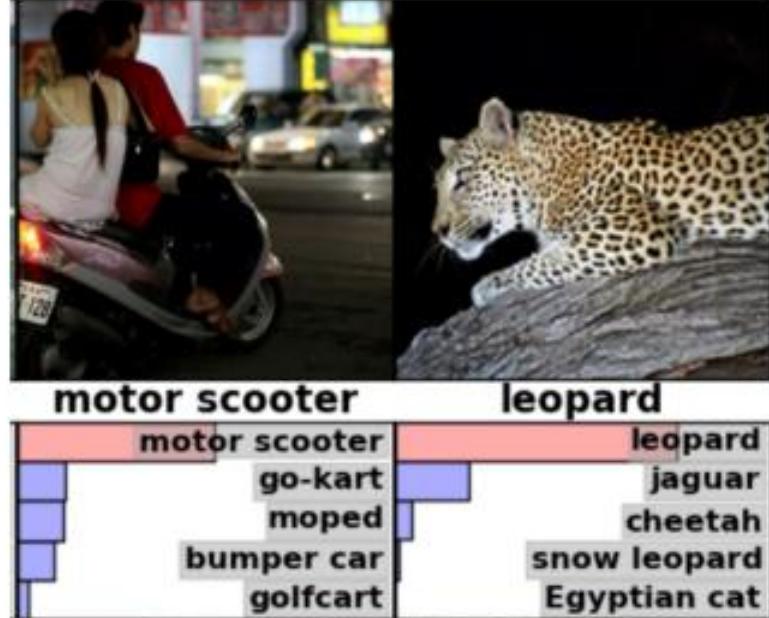
“norm”: local response normalization

“full”: fully connected

Figure: [Karnowski 2015] (with corrections)

ALEXNET- ESPECIFICAÇÃO

- 1000 categorias do ImageNet
- 1.2 milhões de imagens de treino
- 50.000 imagens de validação
- 150.000 imagens de teste



- 60M Parâmetros
- Treinado em duas GPUs GTX entre 5 e 6 dias



ALEXNET – PRINCIPAIS IDEIAS

- Usava **ReLU** – tornava convergência mais rápida
- Usava técnicas de **data augmentation**
- Implementou **dropout** para combater overfitting
- Treinou modelo usando **batch stochastic gradient descent**
- Usou **momentum** e **weight decay**



VGG Net - 2014

"Simple and deep"

Top-5 error rate of 7.3% on ImageNet

16 layer CNN - Best result - Conf. D

138 M parameters

Trained on 4 Nvidia Titan Black GPUs

for two to three weeks.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 conv1-256	conv3-256 conv3-256 conv3-256	conv3-256 conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512 conv3-512	conv3-512 conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

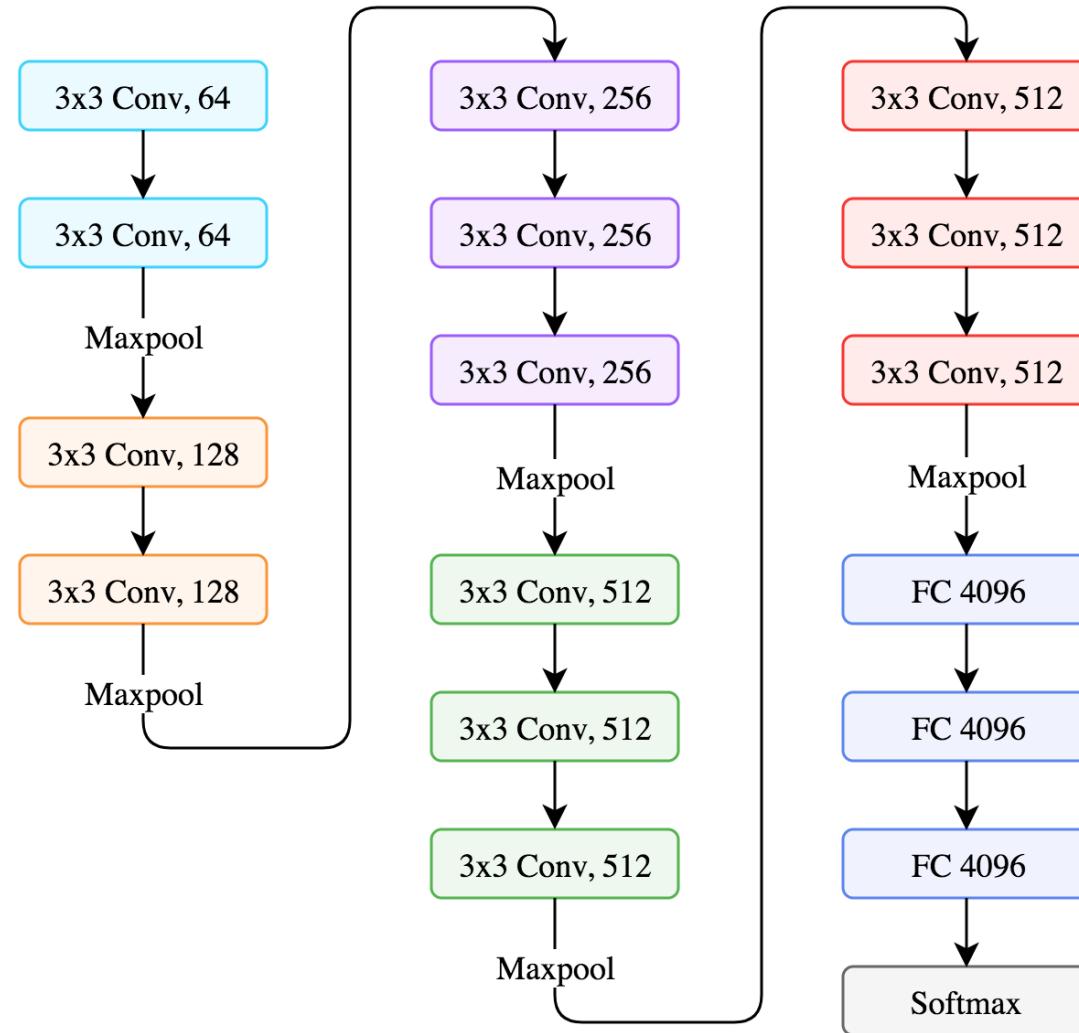


VGG – IDEIAS PRINCIPAIS

- Usa apenas filtros 3x3
 - Usar várias vezes = maior campo receptivo
- Diminui a dimensão espacial e aumenta profundidade da rede
- Data augmentation com variação de escala
- Uso de Relu depois de cada camada convolutiva
- Número de parâmetros reduzidos
 - 3×3^2 , ao invés de 7^2



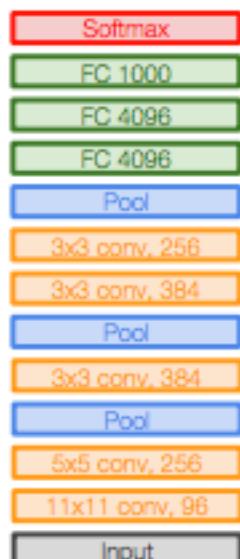
VGG-16



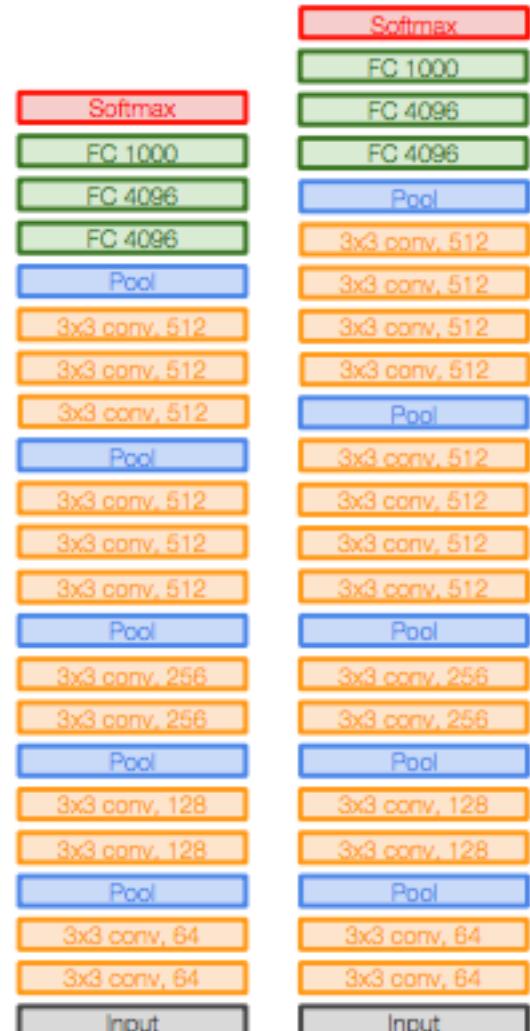
VGGNET

[Simonyan and Zisserman, 2014]

- Usa filtros menores
- Duas versões?
 - 16 Camadas
 - 19 Camadas



AlexNet



VGG16

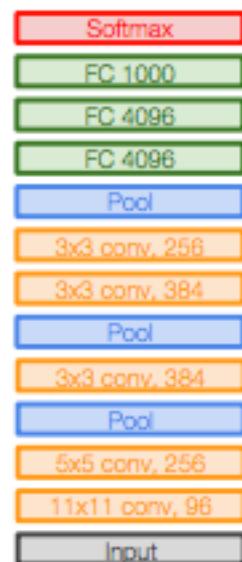
VGG19

VGGNET

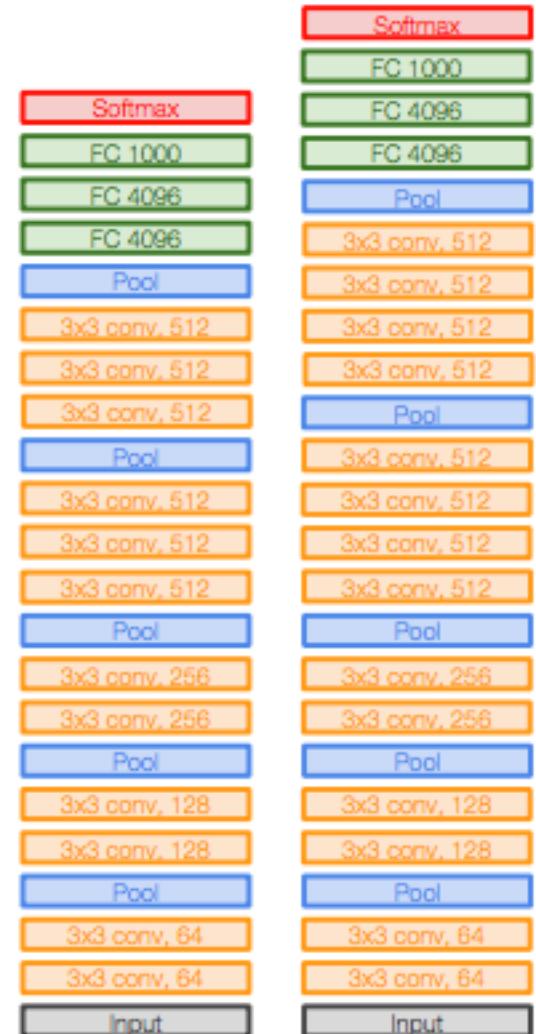
[Simonyan and Zisserman, 2014]

Por que usar filtros menores? (3x3 conv)

Enfileirar três camadas 3x3 conv
(stride 1) tem o mesmo **campo
receptivo efetivo** de uma camada
7x7 conv



AlexNet

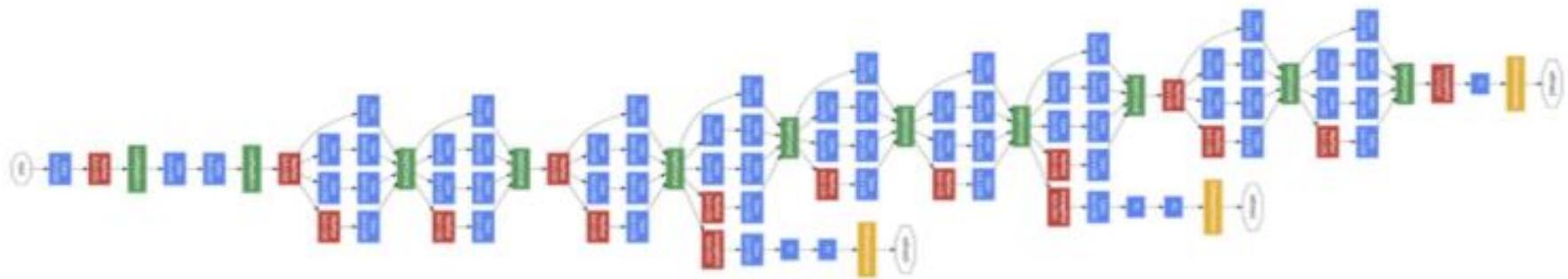


VGG16

VGG19

GOOGLENET / INCEPTION (2014)

- Vencedora da ILSVRC 2014 com erro de 6.7% (4M parâmetros comparado com 60M do AlexNet)
- Treinado em uma semana (algumas GPUs high-end)



Convolution
Pooling
Softmax
Other

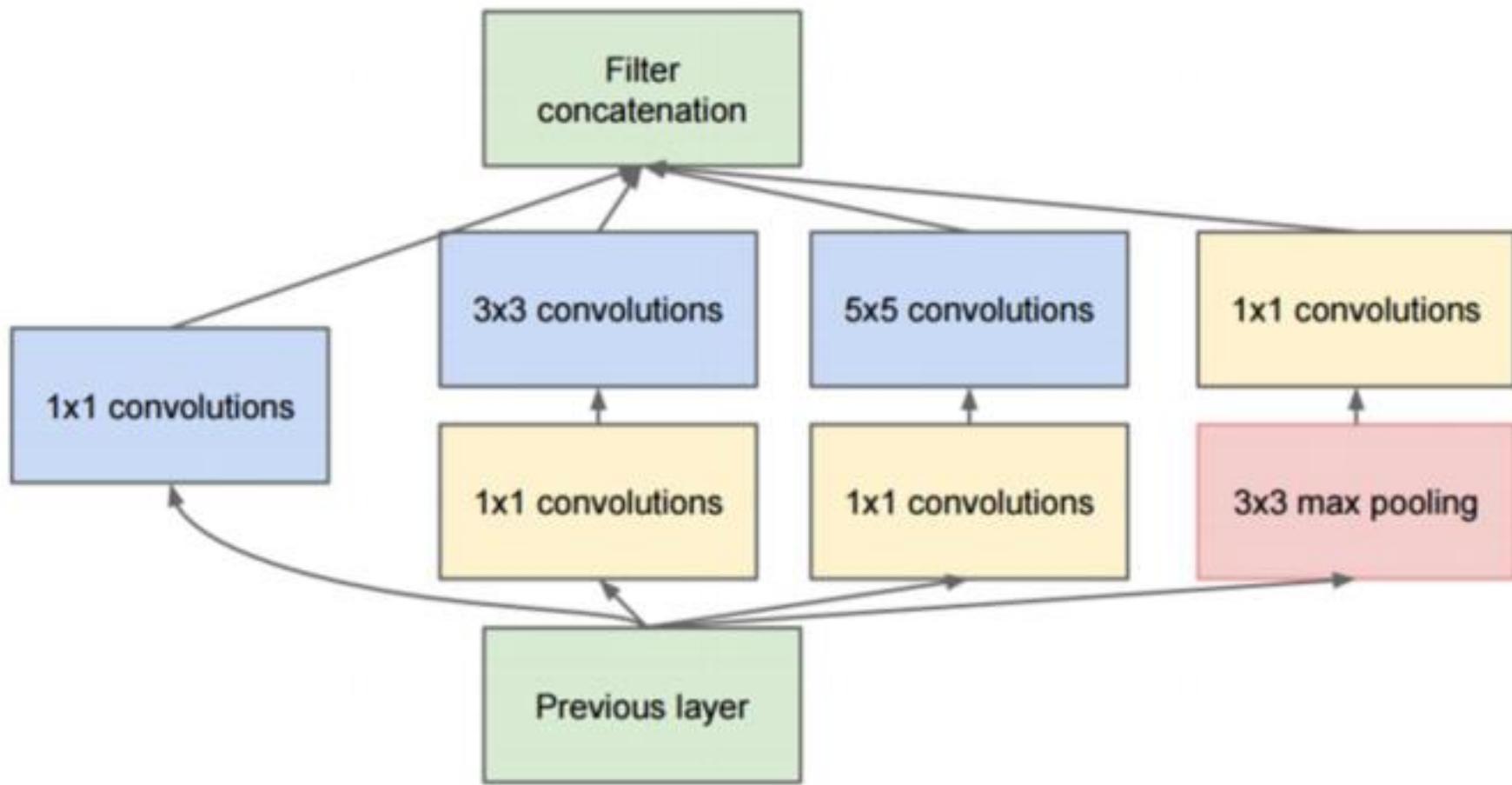


GOOGLENET – PRINCIPAIS IDEIAS

- Usa 9 módulos inceptions
- Não usa camadas FC
 - Usa average pool para transformar $7 \times 7 \times 1024$ em $1 \times 1 \times 1024$
- Usa 12x menos parâmetros que AlexNet



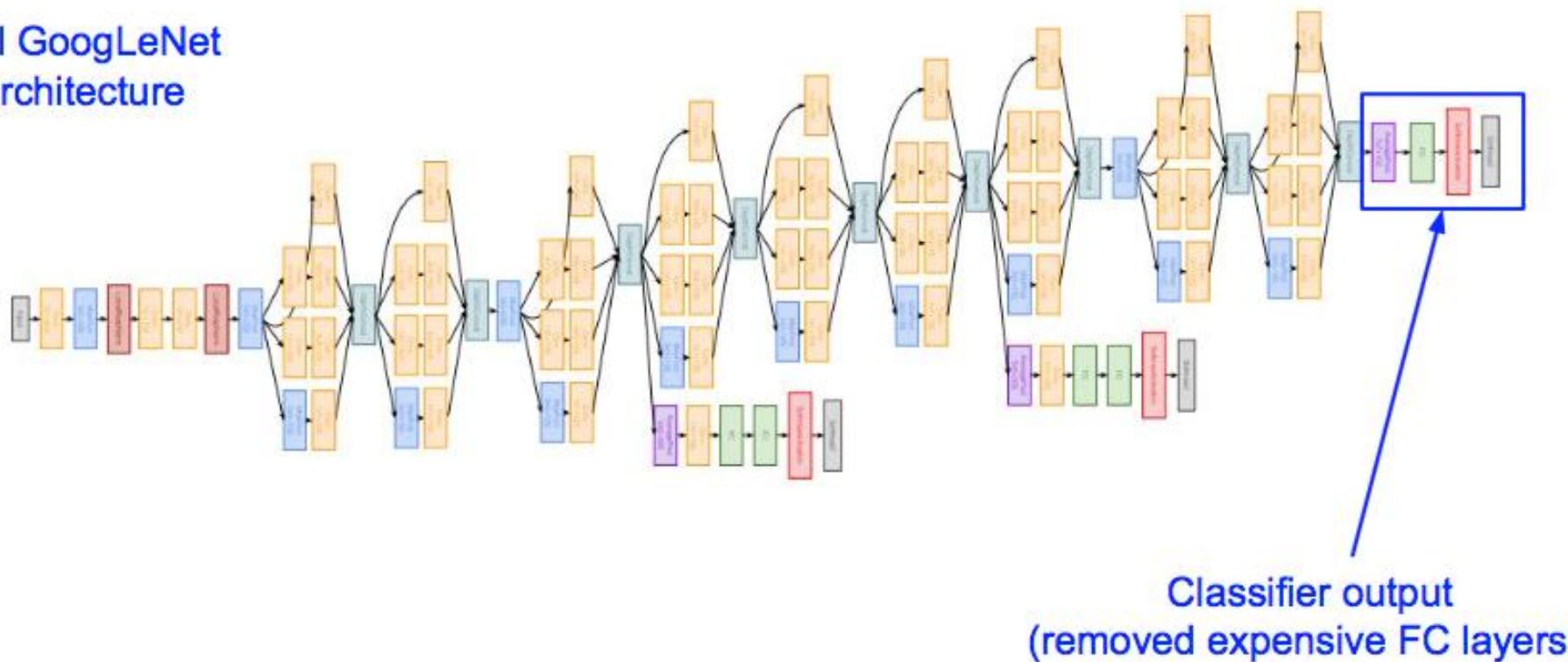
The Inception Module - A closer look



Case Study: GoogLeNet

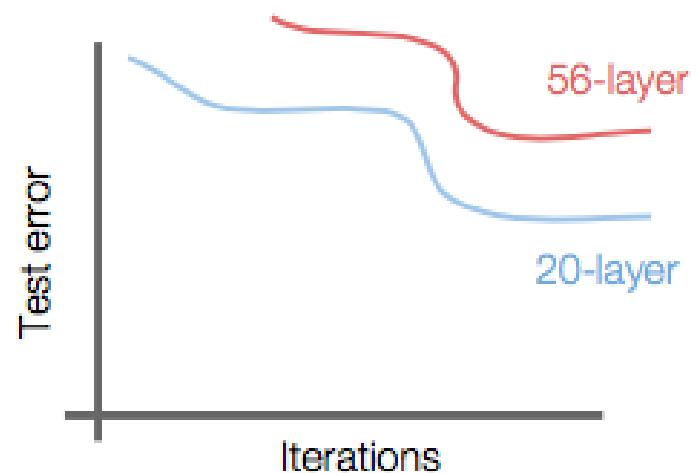
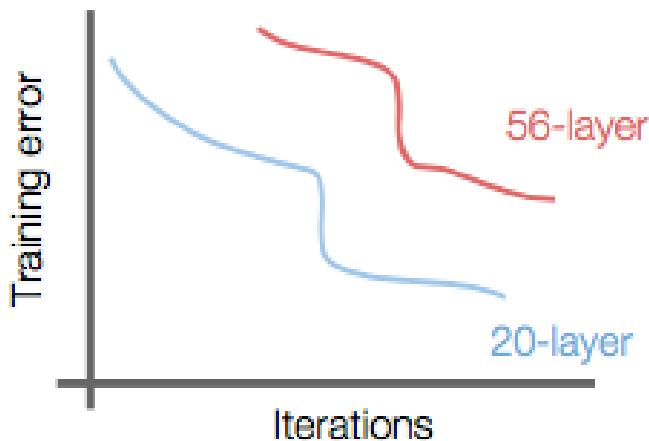
[Szegedy et al., 2014]

Full GoogLeNet
architecture



RESNET (MICROSOFT) [HE ET AL., 2015]

- O que acontece quando continuamos a empilhar mais camadas profundas na CNN?



O modelo mais profundo é pior, mas não é por causa de overfitting



RESNET (MICROSOFT) [HE ET AL., 2015]

■ Resultados

- Capaz de treinar uma *very deep network* (152 camadas) sem perder desempenho
- Primeiro lugar em todas as competições do ILSVRC e COCO
- 3.6% de erro no ILSVRC (melhor que desempenho humano)

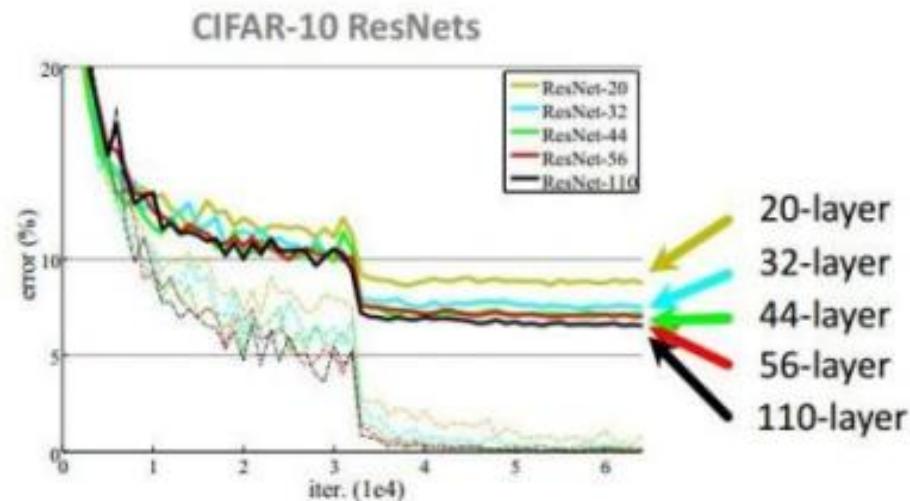
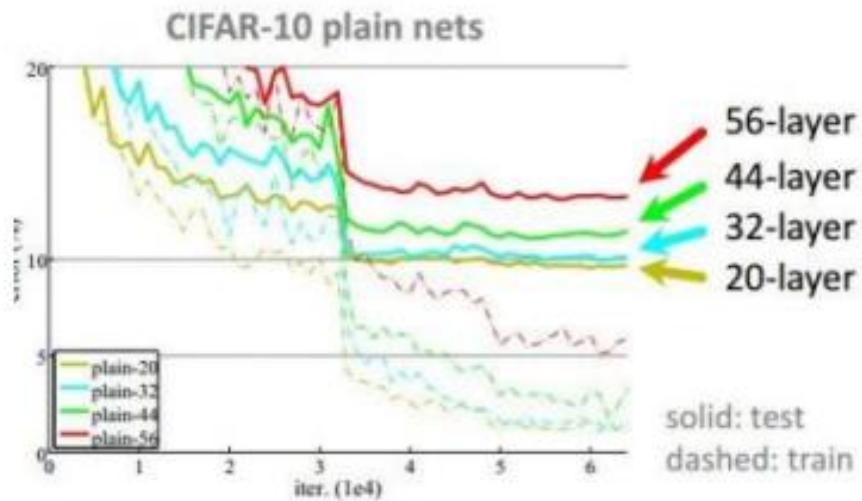
MSRA @ ILSVRC & COCO 2015 Competitions

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



Going deeper

Performance of ResNets versus plain-nets as depth is increased



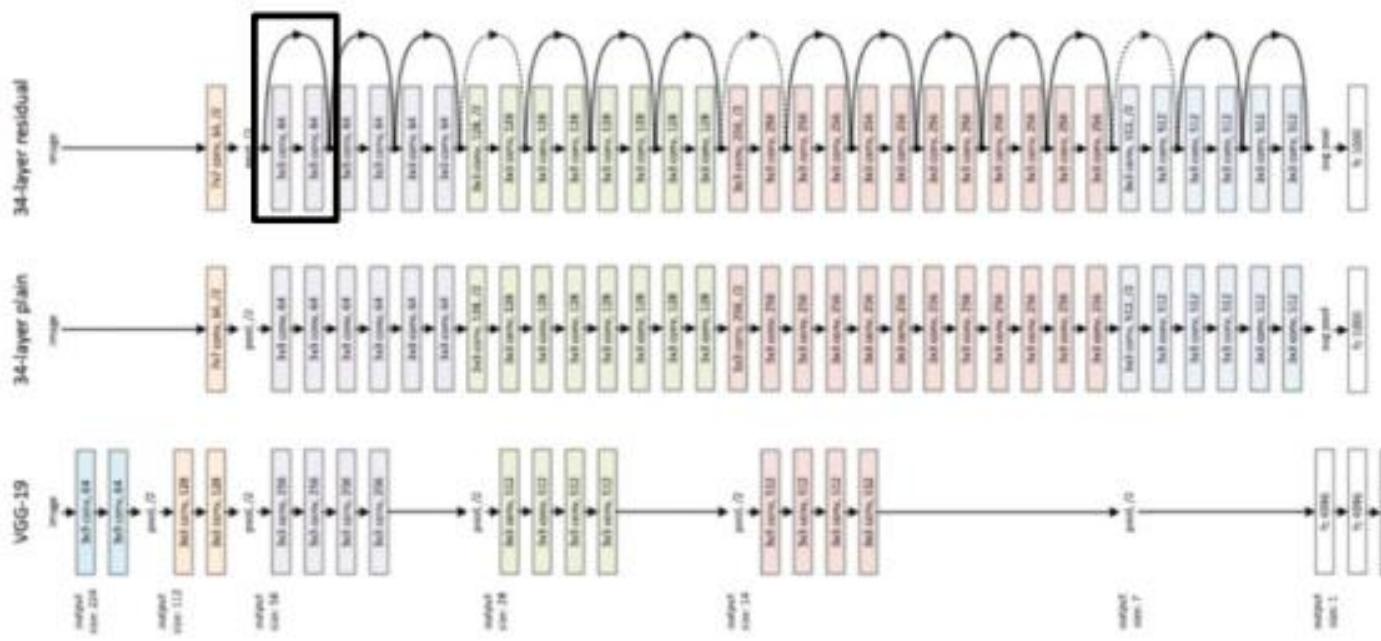
Microsoft ResNet 2015

ResNet won ILSVRC 2015 with an incredible error rate of 3.6%

Humans usually hover around 5-10%

Trained on an 8 GPU machine for two to three weeks.

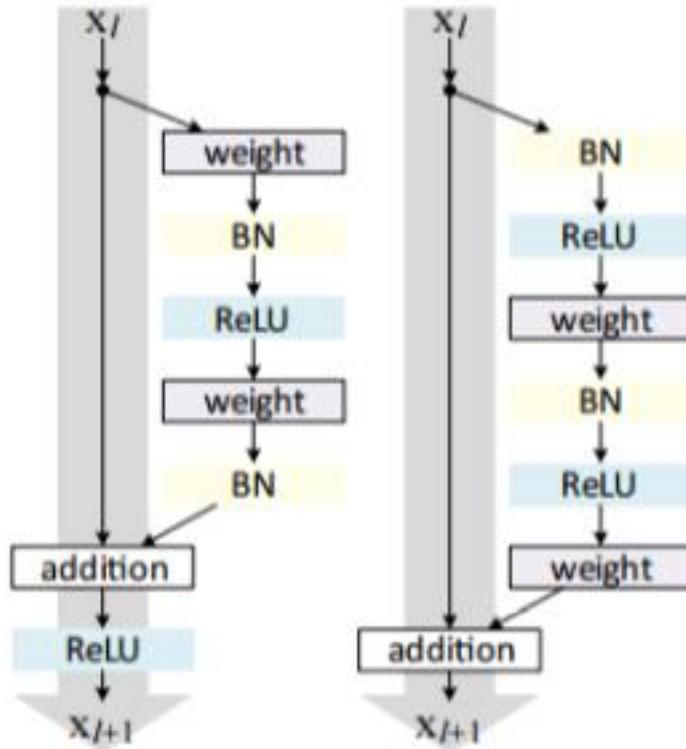
34-residual



VGG



RESNET

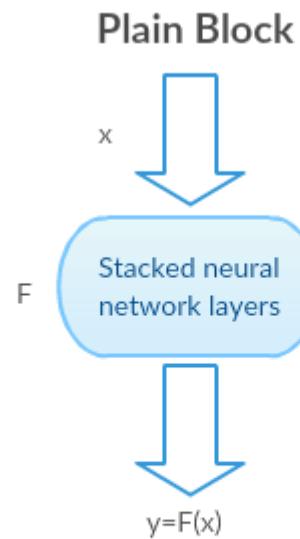


(a) original

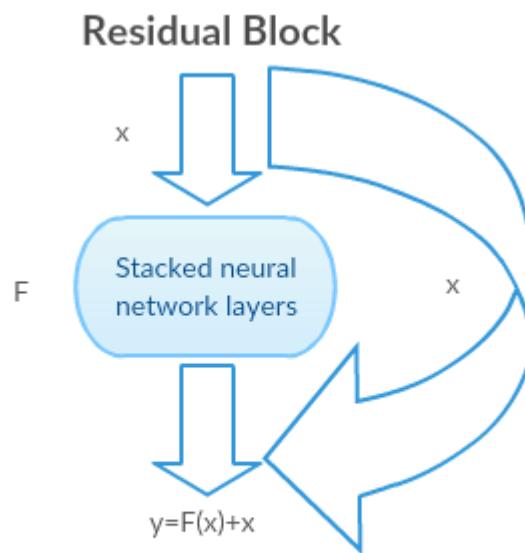
(b) proposed



RESNET



Hard to get $F(x)=x$ and make $y=x$
an identity mapping



Easy to get $F(x)=0$ and make $y=x$
an identity mapping



RESNET

- Treinamento

- Batch Normalization depois de toda camada convolutiva
- Inicialização usando algoritmo de Xavier (He et al.)
- SGD + Momentum (0.9)
- Taxa de aprendizado: 0.1
- Mini-batch size 256
- Weight decay de 1e-5
- Sem dropout



REVOLUÇÃO DA PROFUNDIDADE

AlexNet, 8 layers
(ILSVRC 2012)



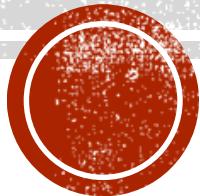
VGG, 19 layers
(ILSVRC 2014)



ResNet, 152 layers
(ILSVRC 2015)



TRANSFER LEARNING

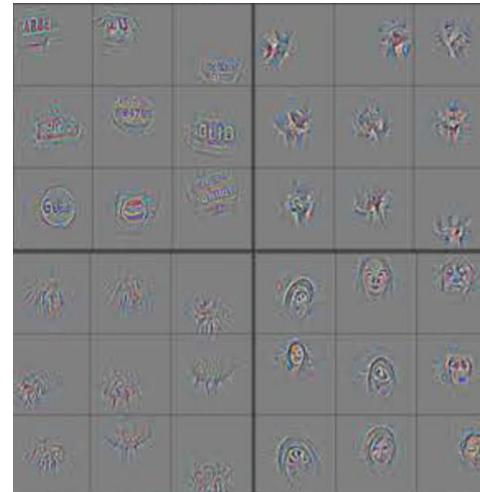
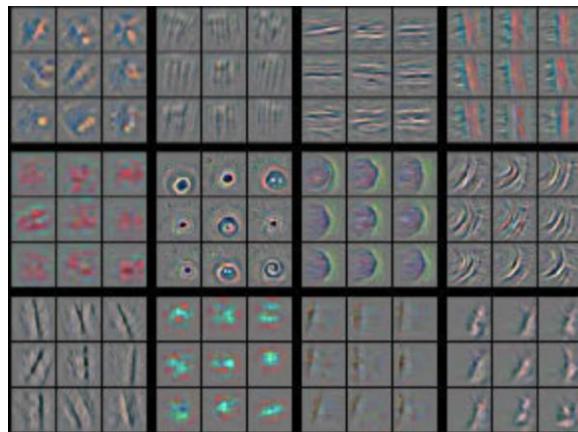
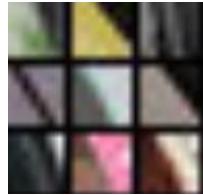
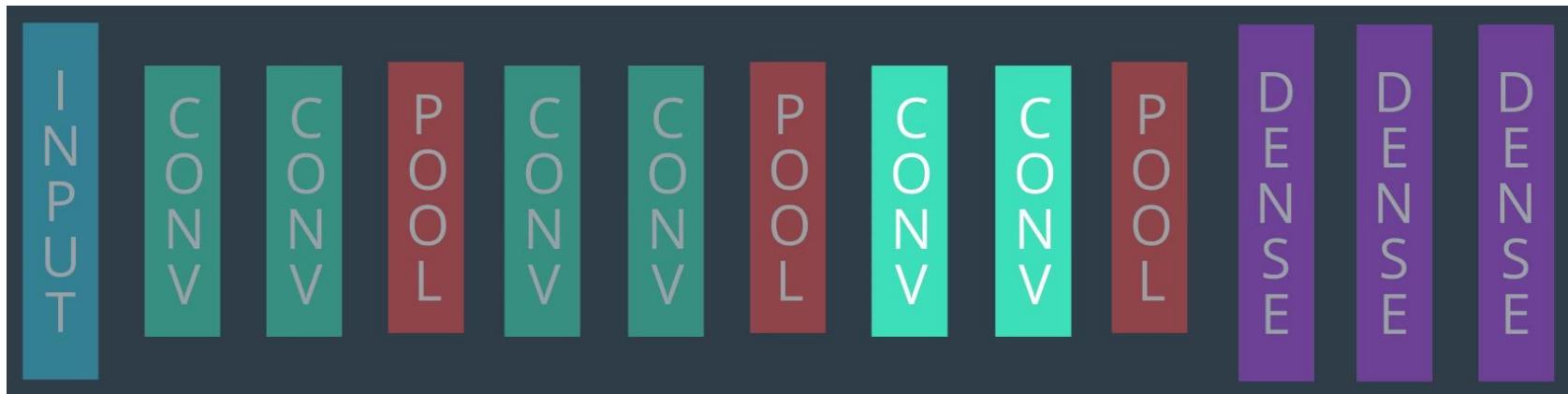


É PRECISO TREINAR A REDE TODA VEZ?

- Transfer learning!



TRANSFER LEARNING



COMO USAR TRANSFER LEARNING

	Base Similar	Base diferente
Base pequena	Usar modelo treinado	Treinar mais camadas
Base grande	Treinar camadas FC	treinar próprio modelo



TRANSFER LEARNING

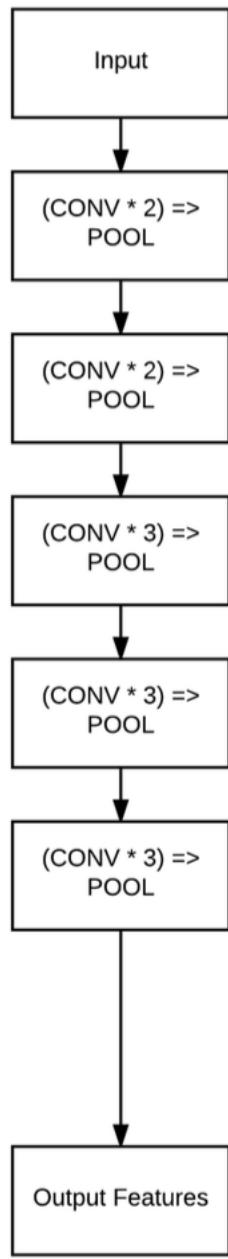
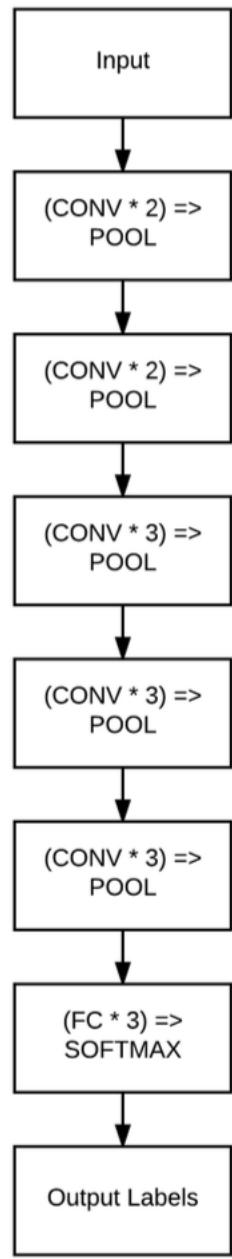
- Escolha um modelo pré-treinado para utilizar em seu problema.
- Geralmente, mantemos as primeiras camadas e retreinamos apenas as últimas camadas
- A imagem de entrada precisa ser reescalonada para o mesmo tamanho de imagem que a rede original foi treinada.



KERAS

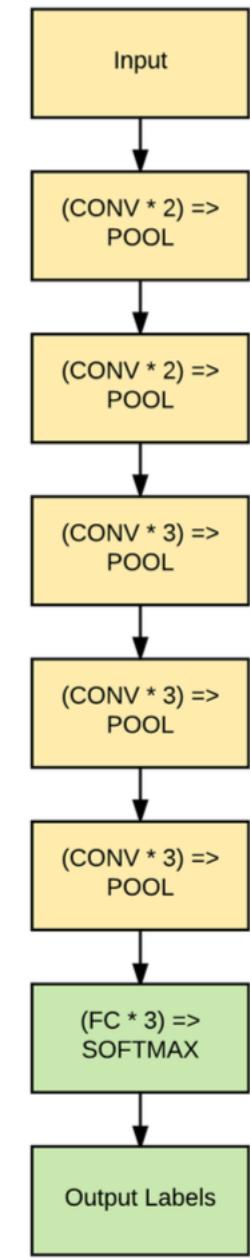
Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.715	0.901	138,357,544	23
VGG19	549 MB	0.727	0.910	143,667,240	26
ResNet50	99 MB	0.759	0.929	25,636,712	168
InceptionV3	92 MB	0.788	0.944	23,851,784	159
InceptionResNetV2	215 MB	0.804	0.953	55,873,736	572
MobileNet	17 MB	0.665	0.871	4,253,864	88





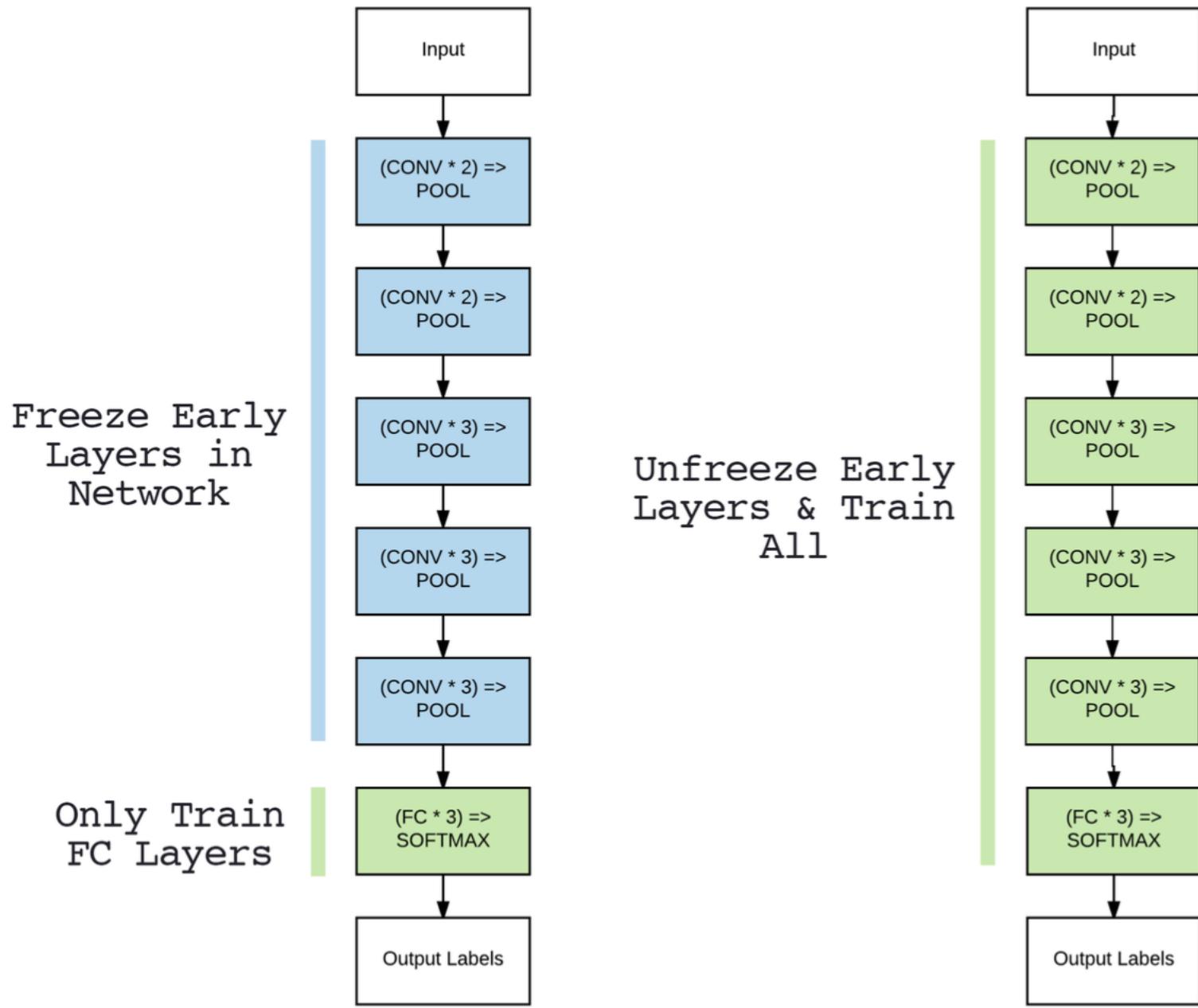
Original
Layers

New FC
Layers



Old FC
Layers





TRANSFER LEARNING

- Locais com Modelos de Redes da Literatura

TF-Slim: (<https://github.com/tensorflow/models/tree/master/slim/nets>)

Keras: (<https://github.com/fchollet/deep-learning-models>)

