

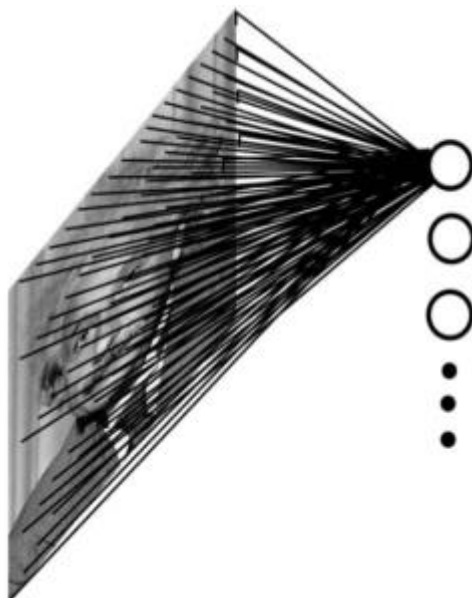
The background features abstract, overlapping geometric shapes in various shades of blue, ranging from light sky blue to deep navy blue. These shapes are primarily located on the left and right sides of the frame, creating a modern, dynamic feel.

CNN

Marcelo Piovan - piovan@heurys.com.br

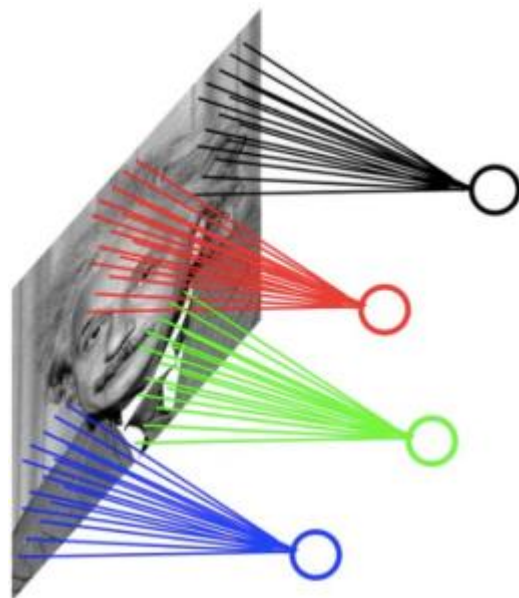
Problema da MLP

- ▶ A entrada pode ter uma dimensão muito alta, quando usamos a MLP, por causa disso, precisamos de uma quantidade muito grande de parâmetros.
- ▶ Exemplo:
Uma imagem 200x200 de entrada, na MLP teremos 40.000 valores de entrada e 1.6 bilhões de parâmetros.



Rede Neural Convolucional (CNN)

- ▶ As CNNs são um tipo especial de redes neurais que usam o conceito de campo receptivo local.
- ▶ Exemplo:
Uma imagem 200x200 de entrada, na CNN com 5x5 kernel, 100 campos receptivos haverá 2.500 parâmetros.



“Odin” da CNN

- ▶ Em 1995, Yann LeCun e Yoshua Bengio introduziram o conceito de redes neurais convolucionais.



CNN

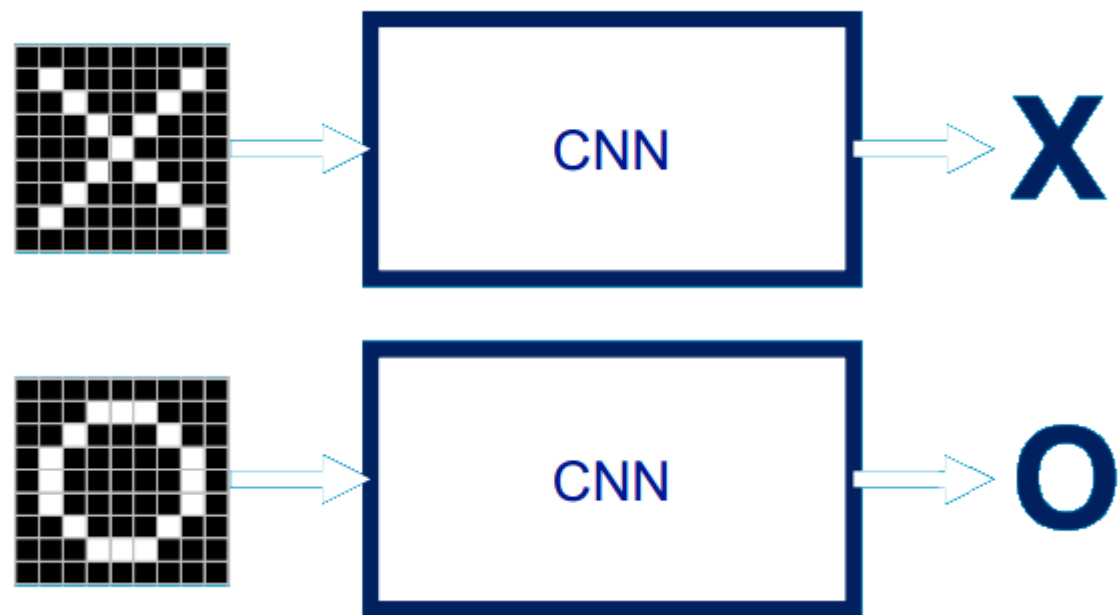
- ▶ Foram inspiradas pela sensibilidade local e orientação seletiva do cérebro.
- ▶ É uma rede neural que implicitamente extraem características relevantes da entrada.
- ▶ São um tipo especial de multi-layer neural networks (MLP);
- ▶ Como qualquer outra rede neural a arquitetura é treinada com alguma variação do algoritmo back-propagation;

CNN

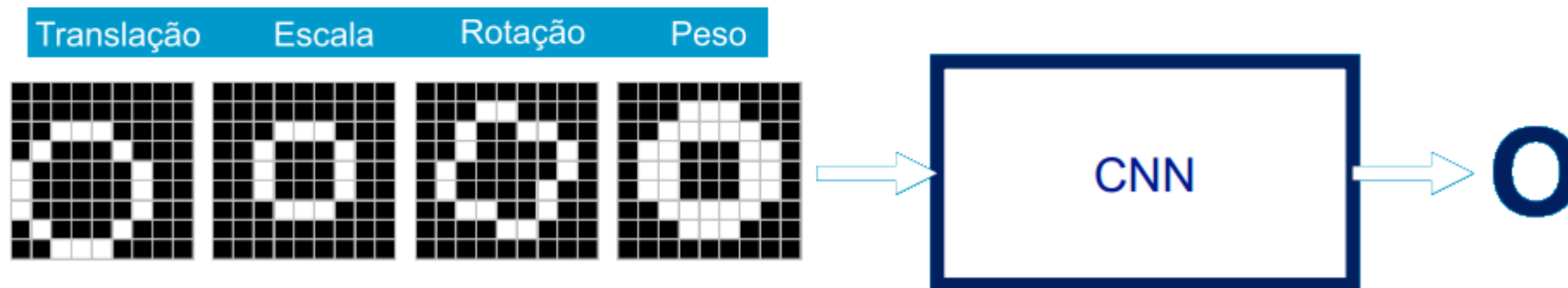
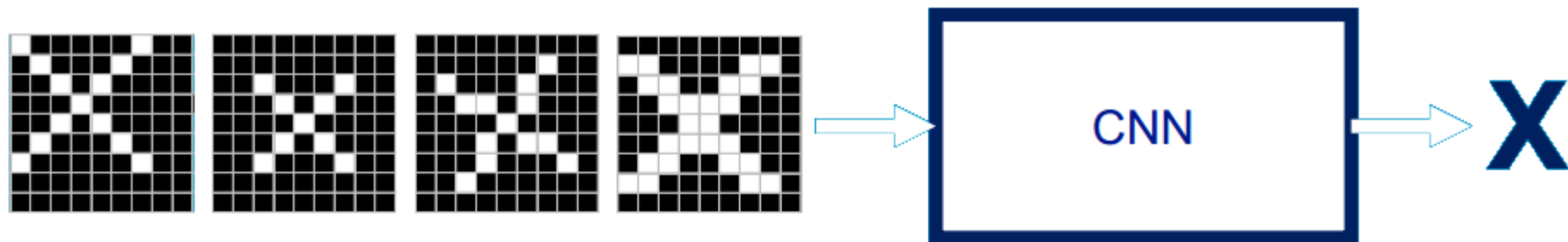
Decidir se uma figura contém X ou O



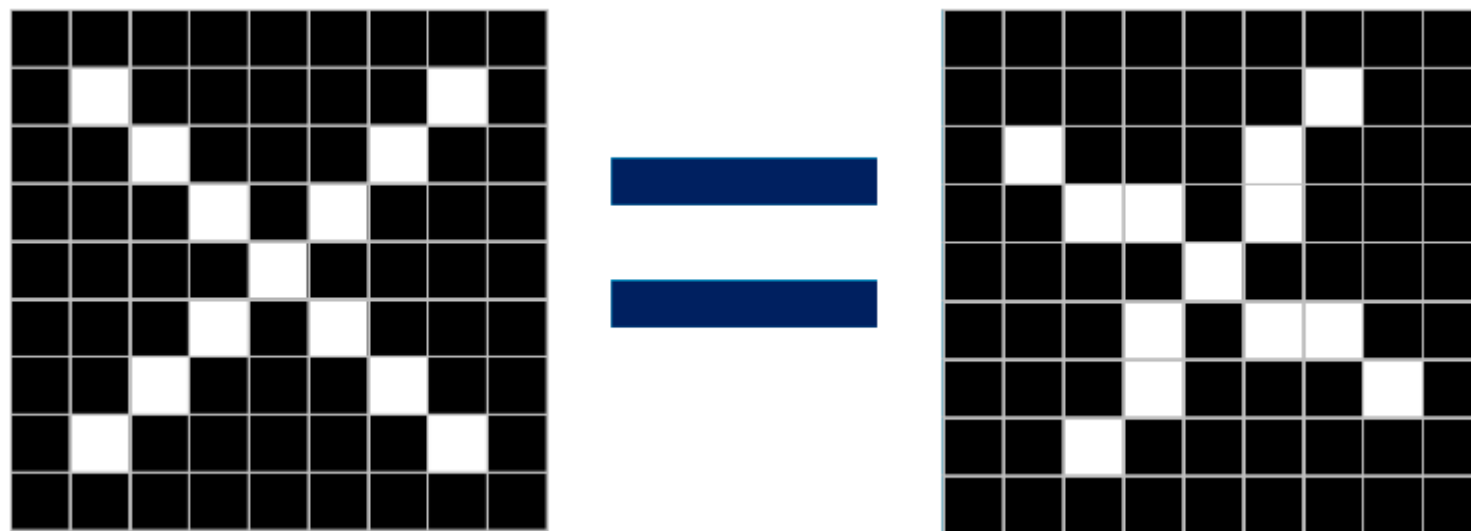
Exemplo básico



Vamos complicar !



Qual será a decisão do computador ?



Camadas da rede convolucional

- ▶ **Entrada (input):** Altura x Largura x Camadas, exemplo 16 x 16 x 3, onde:
 - ▶ Altura = 16 pixels
 - ▶ Largura = 16 pixels
 - ▶ Camadas = 3 (RGB)
- ▶ **Convolução:** É a camada que calculará a saída dos neurônios conectados as regiões locais da entrada.
- ▶ **RELU:** Função de ativação ponto a ponto, pode ser utilizada com max, min e avg.
- ▶ **Pool:** É uma camada downsampling, que visa corrigir os efeitos de escala.
- ▶ **Fully Connected (Classificador):** MLP, SVM, Softmax, etc.

Camada de convolução

- ▶ Fórmula para calcular

output	filter	input (image)
$G[i, j] = \sum_{u=-k}^k \sum_{v=-k}^k H[u, v] F[i + u, j + v]$		

Camada de convolução - Filtro

- ▶ Para calcular os filtros para termos a saída de nossa rede, precisamos:
 1. Multiplicar cada pixel da imagem de entrada pelo pixel correspondente do filtro.
 2. Adicionar ao somatório.
 3. Dividir pelo total de pixels.

► Exemplo:

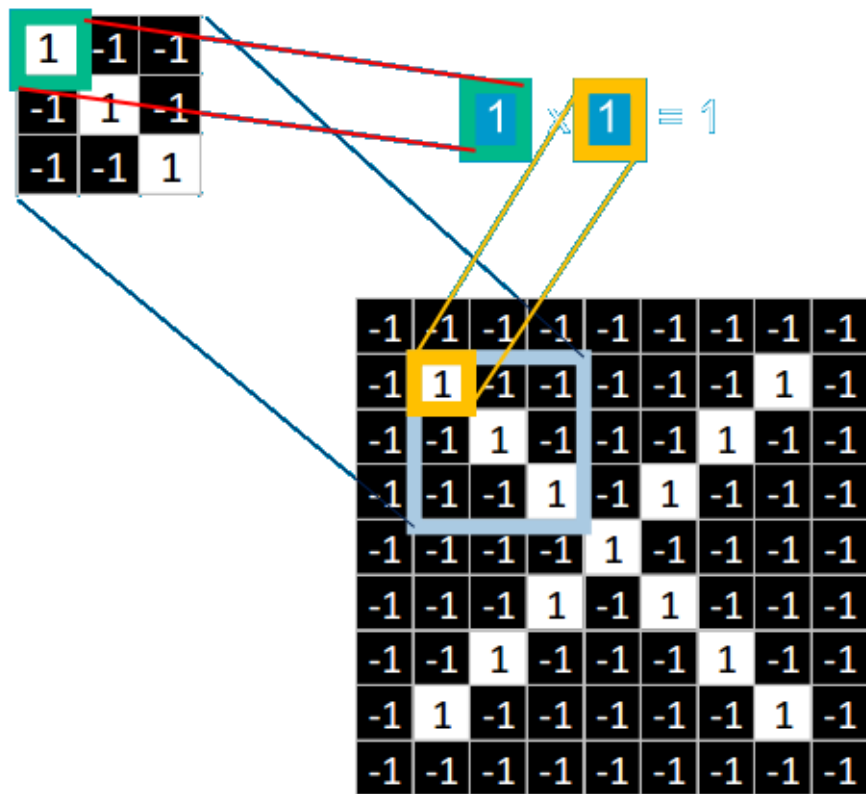
Filtro

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

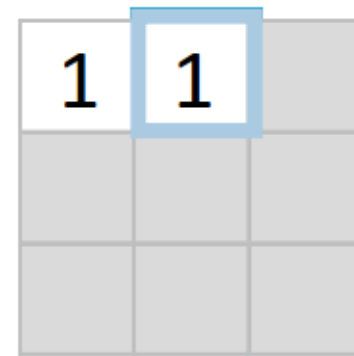
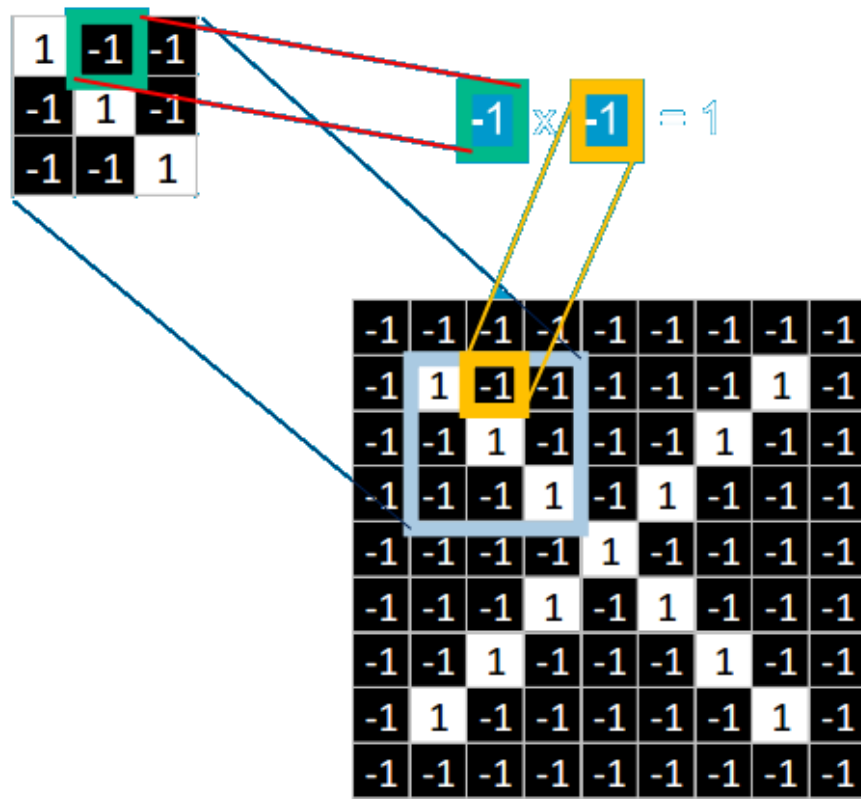
Imagem de entrada

[illegible]

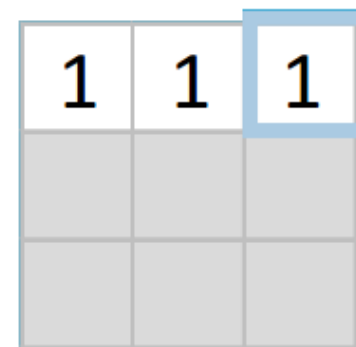
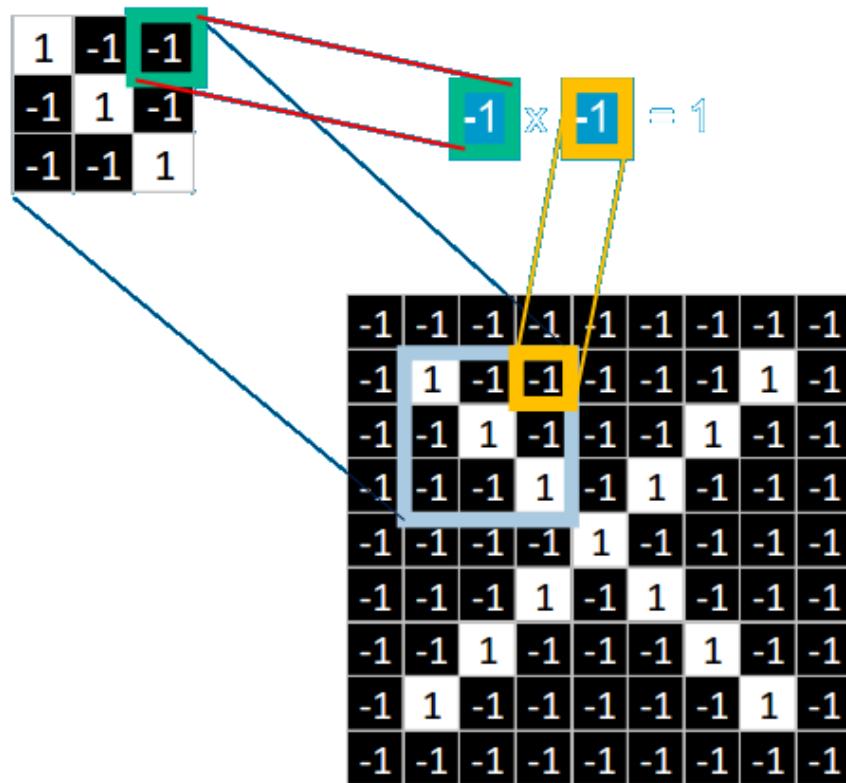
Camada de convolução - Filtro



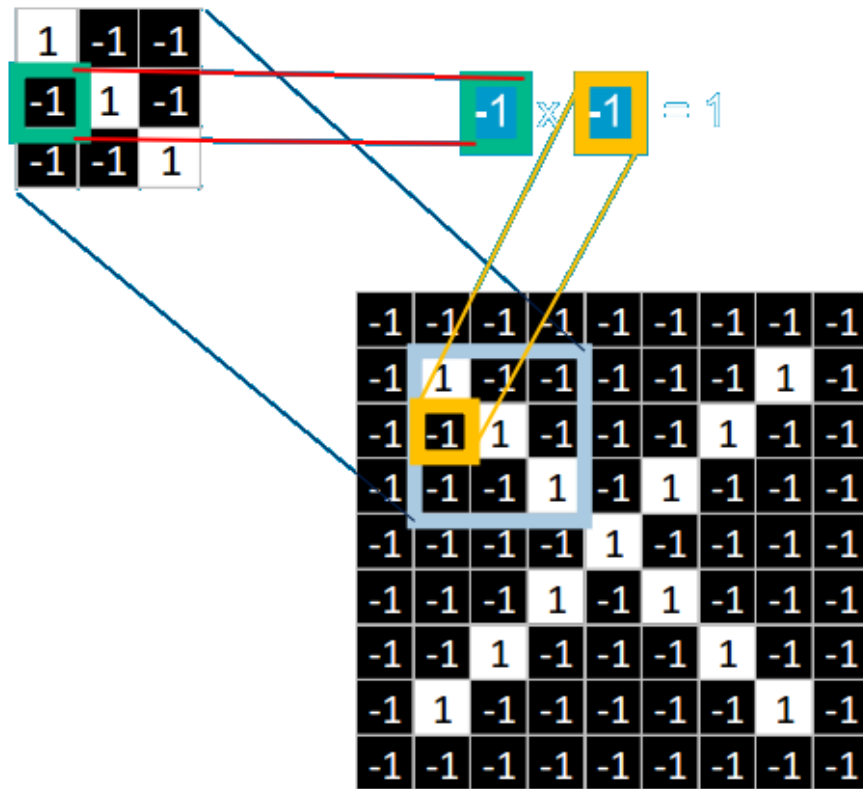
Camada de convolução - Filtro



Camada de convolução - Filtro

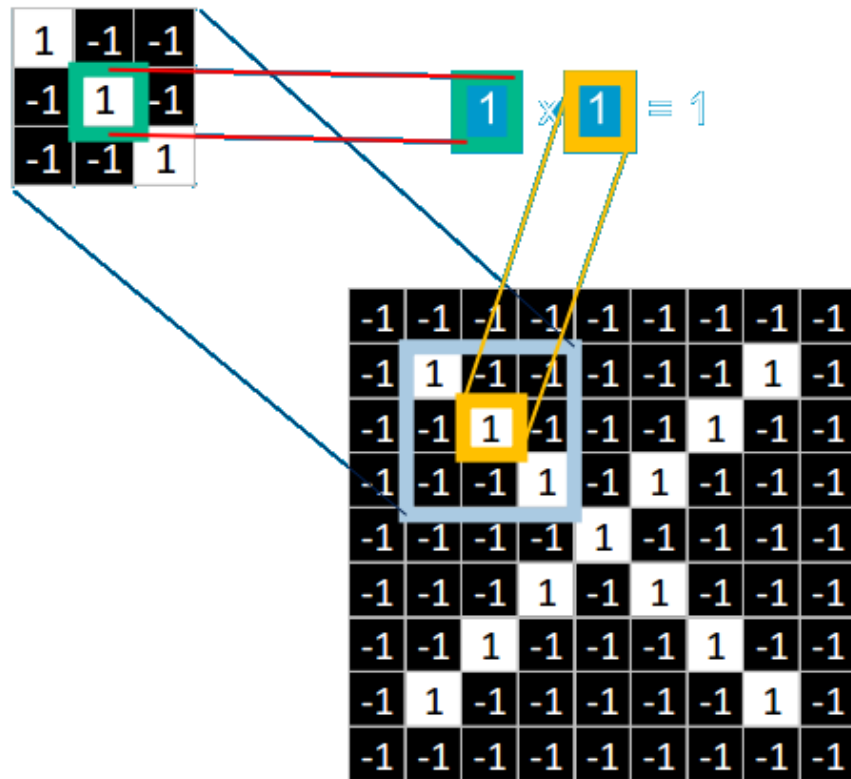


Camada de convolução - Filtro



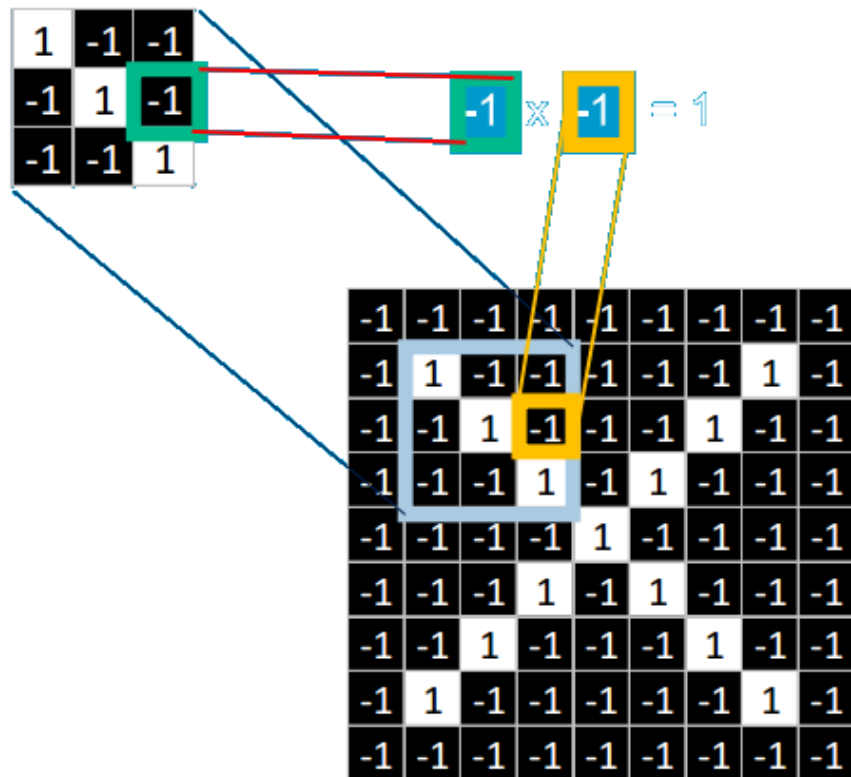
1	1	1
1		

Camada de convolução - Filtro



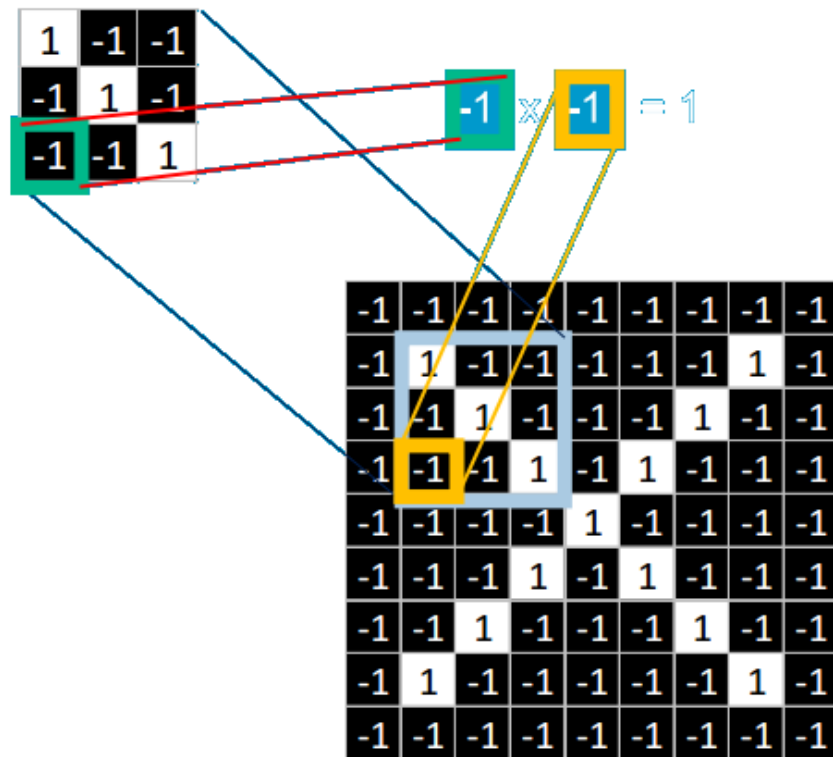
1	1	1
1	1	

Camada de convolução - Filtro



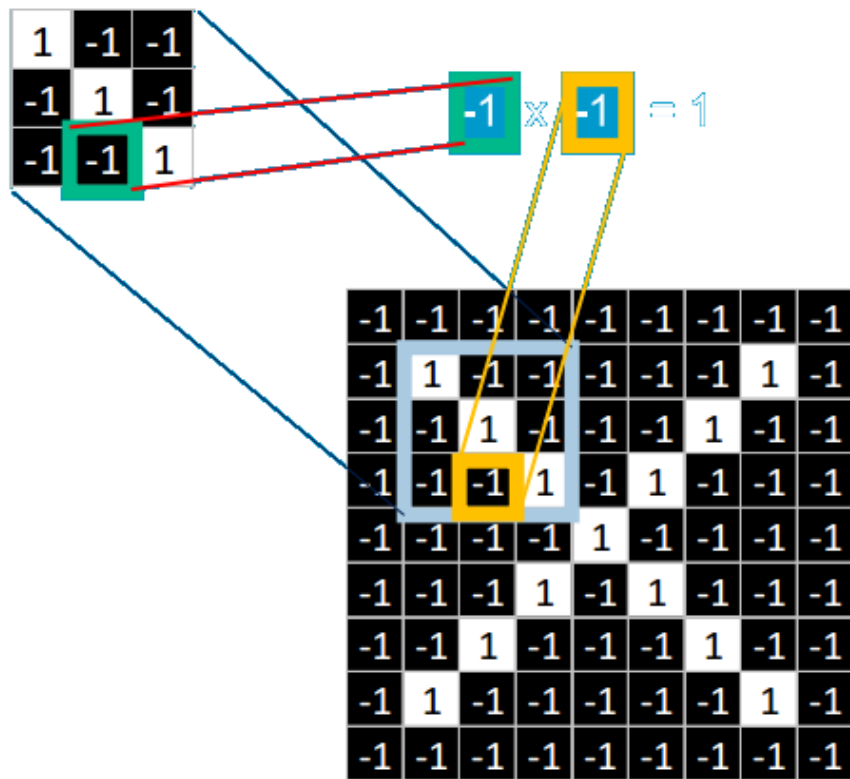
1	1	1
1	1	1

Camada de convolução - Filtro



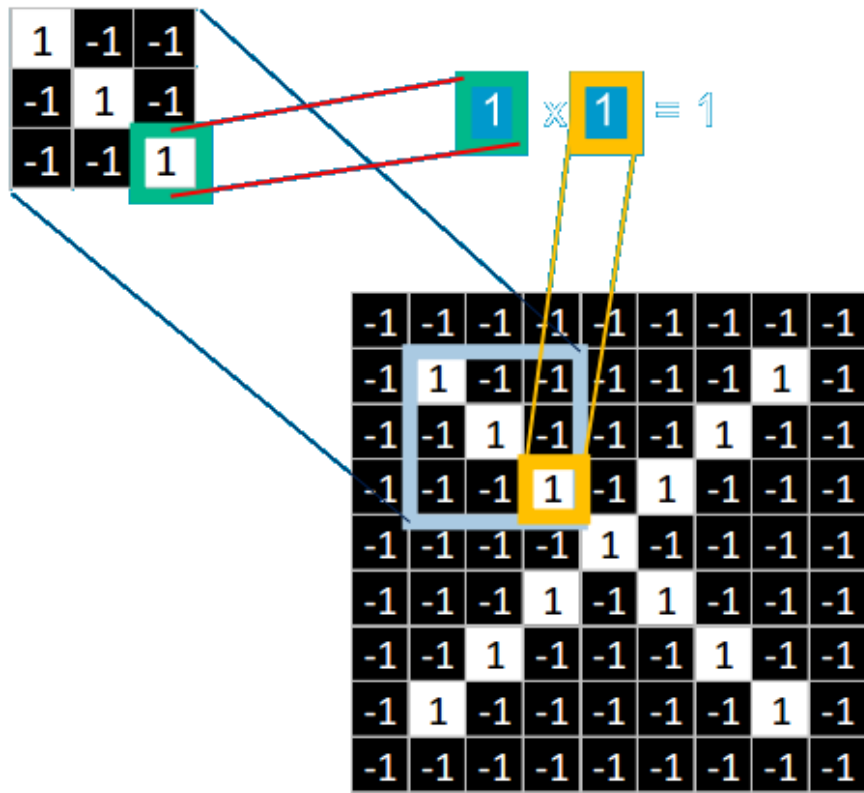
1	1	1
1	1	1
1		

Camada de convolução - Filtro



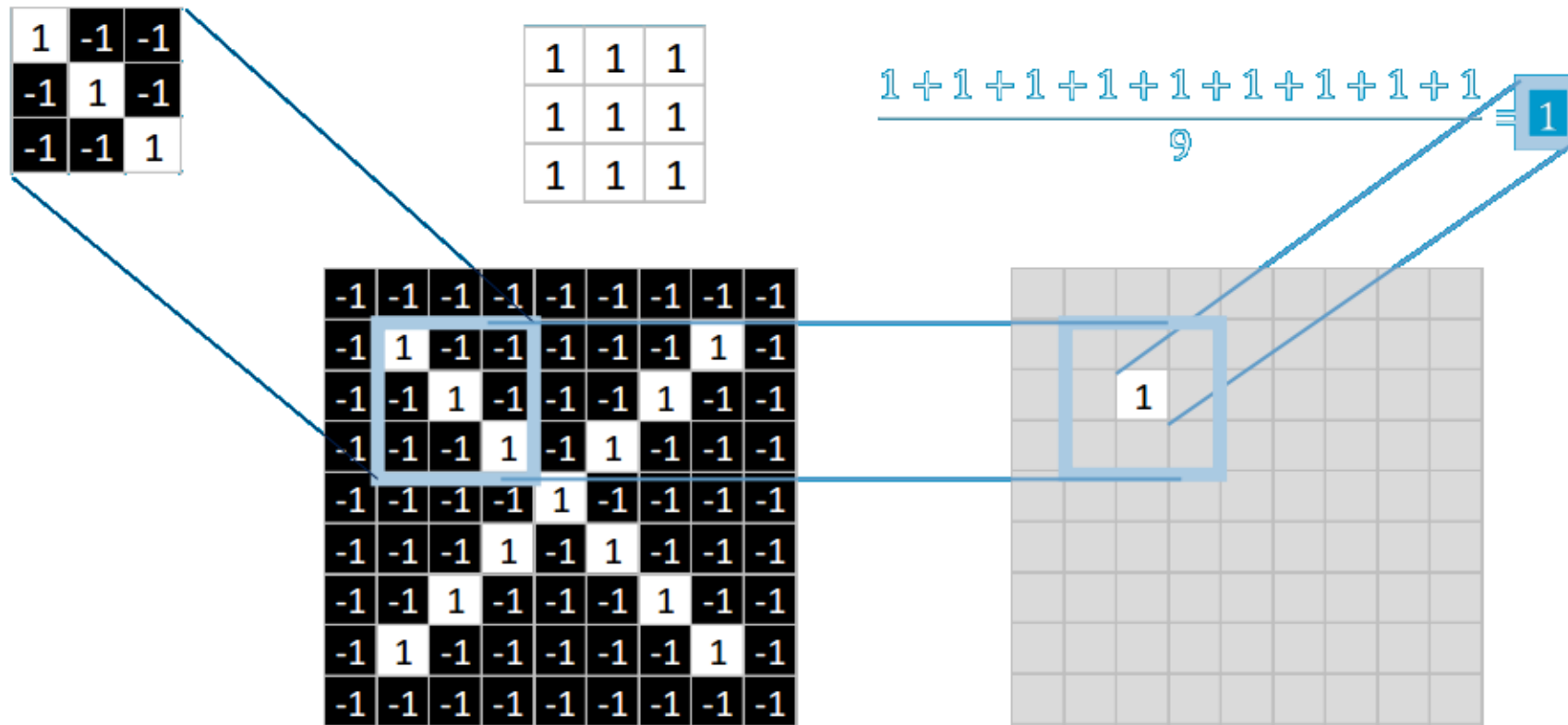
1	1	1
1	1	1
1	1	

Camada de convolução - Filtro



1	1	1
1	1	1
1	1	1

Camada de convolução - Filtro



Camada de convolução - Filtro

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

Camada de convolução - Filtro

Exemplos de filtros

0	0	0	0	0	1	0
0	0	0	0	1	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0

Matriz numérica do filtro convolucional



Table 3.2: Visualização do filtro convolucional



0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

*

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

Camada de convolução - Filtro

Exemplos de filtros



Figure 3.2: Imagem de entrada para aplicação do filtro de convolução

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

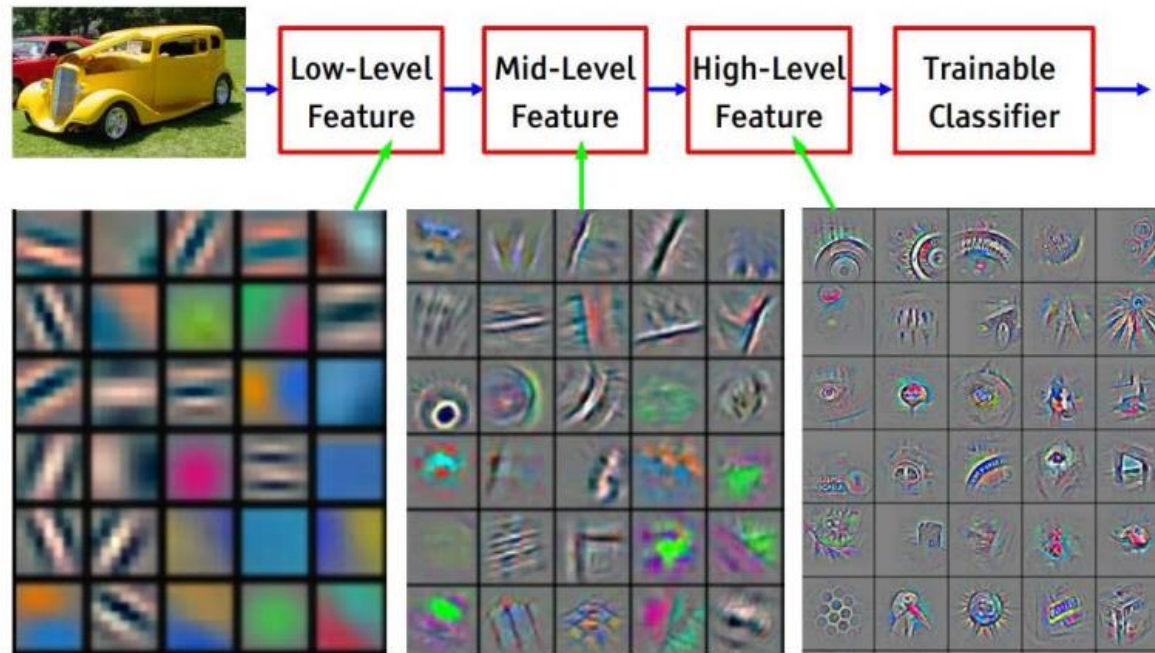
Imagem de entrada considerando a região do amarelo no canto superior esquerdo

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

Table 3.4: Filtro convolucional

Camada de convolução - Filtro

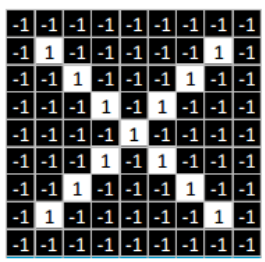


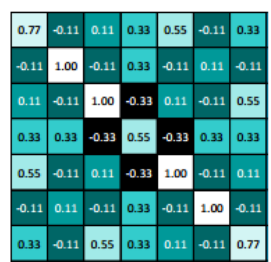
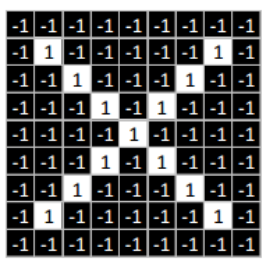

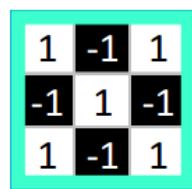

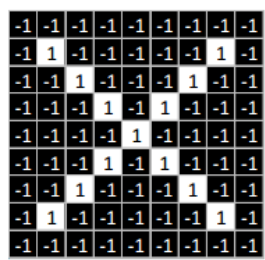

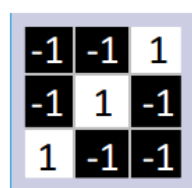
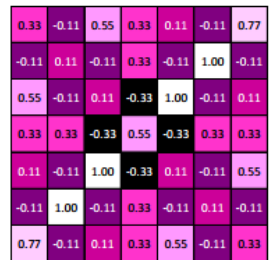
Outros exemplos



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

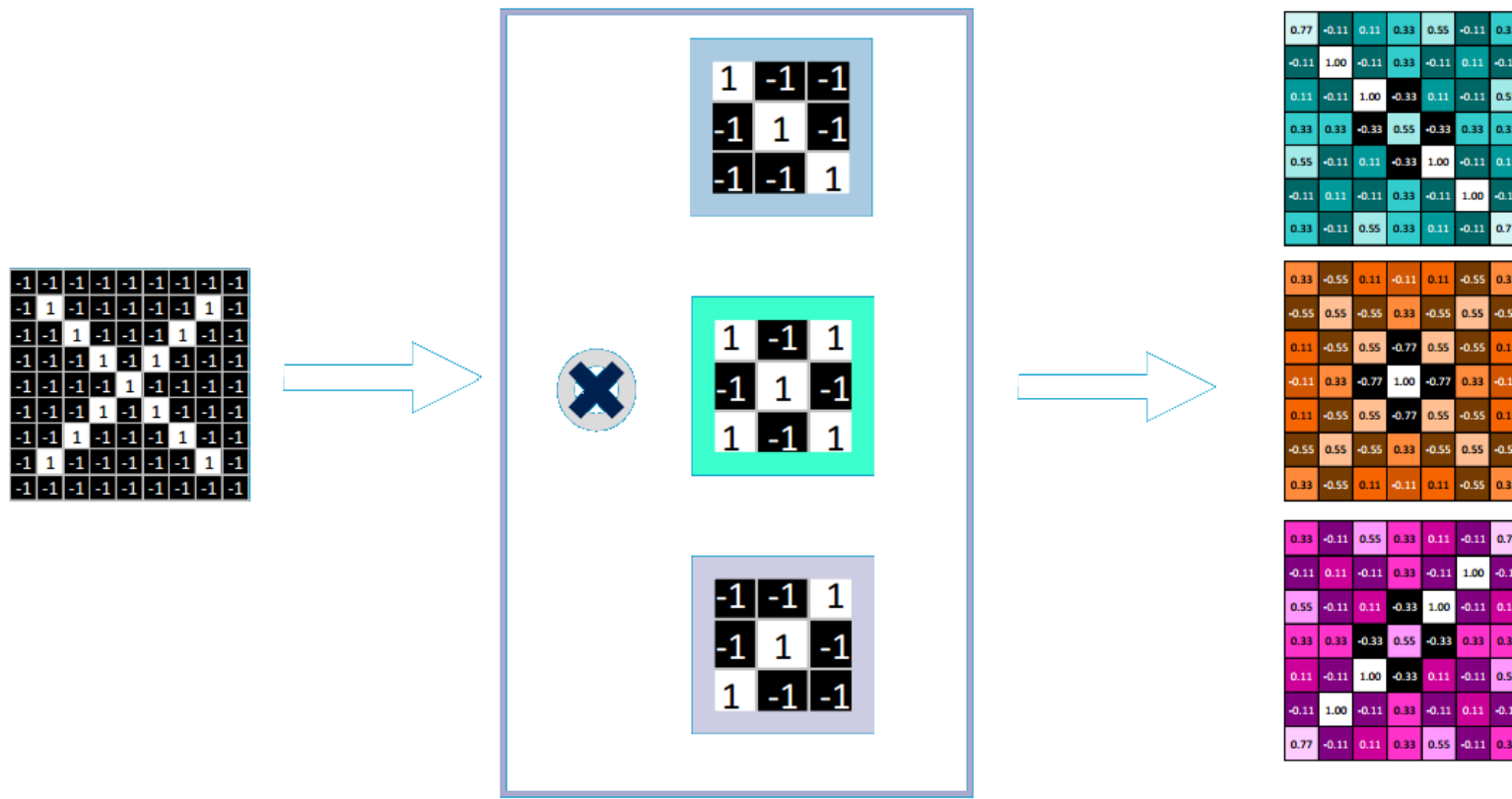
Camada de convolução - Filtro

Voltando para nosso caso X ou O

			=	
			=	
			=	

Camada de convolução - Filtro

Uma imagem gera um conjunto de imagens

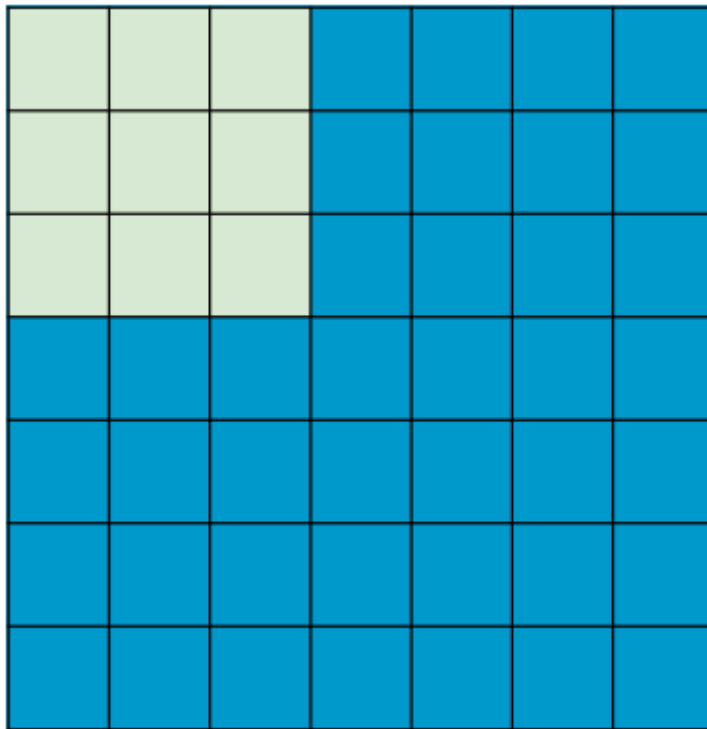


Camada de convolução - Filtro

- ▶ Podemos dizer se temos 3 filtros 3x3, teremos 3 mapas de ativação (3 imagens) e essas imagens são empilhadas.
- ▶ Para calcular o tamanho da nossa saída, utilizamos a seguinte fórmula.
 - ▶ $(N - F) / \text{stride} + 1$
 - ▶ Dado nosso exemplo imagem de entrada 9x9x1 e filtros 3x3
 - ▶ N (entrada) = 9
 - ▶ F (filtro) = 3
 - ▶ Sendo stride = 1: $(9 - 3) / 1 + 1 = 7$, output **7x7x3**

Camada de convolução - Filtro

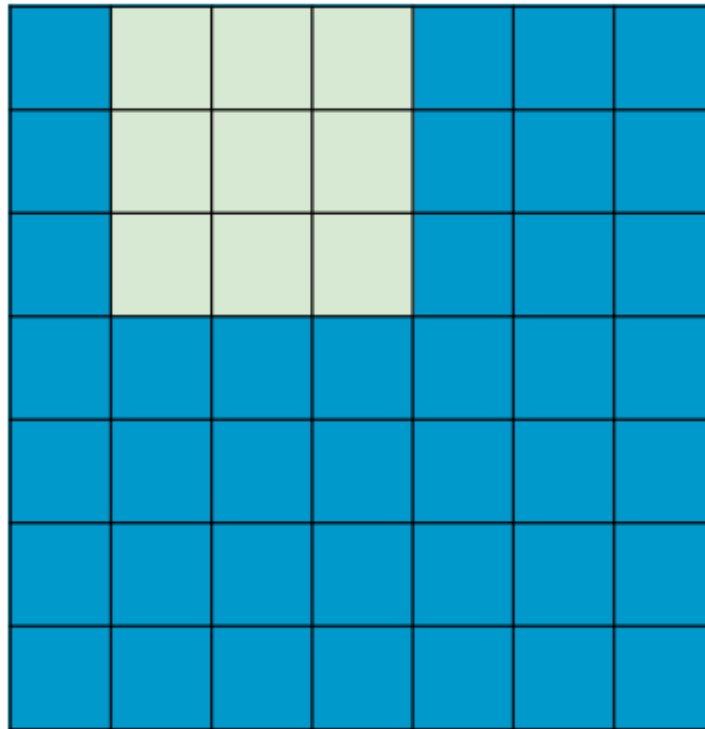
Slide e Stride



Entrada: 7 x 7
Filtro: 3 x 3
Stride: 1 x 1

Camada de convolução - Filtro

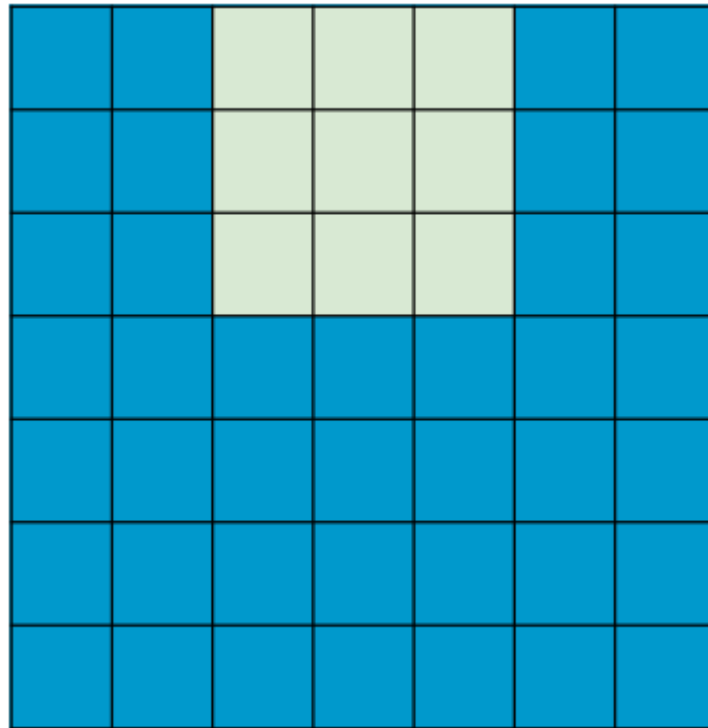
Slide e Stride



Entrada: 7 x 7
Filtro: 3 x 3
Stride: 1 x 1

Camada de convolução - Filtro

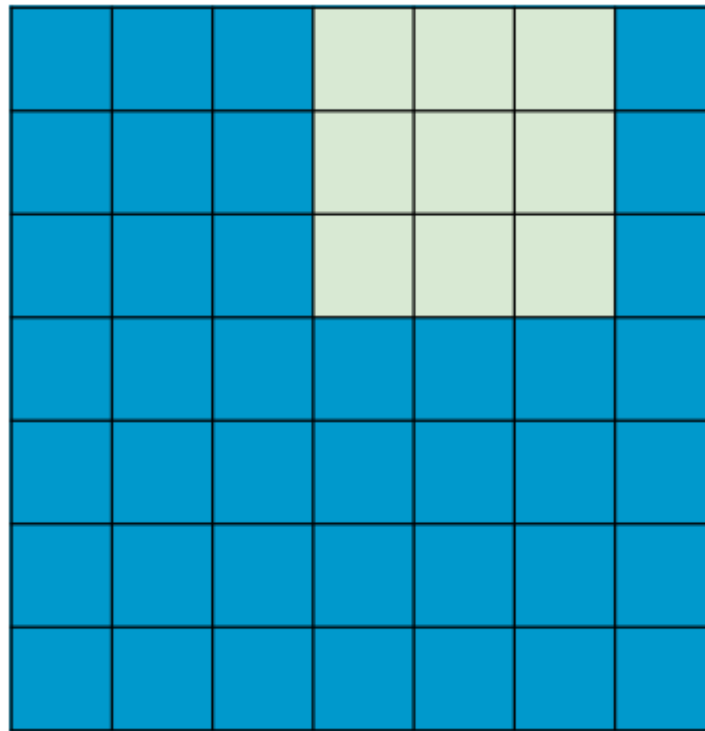
Slide e Stride



Entrada: 7 x 7
Filtro: 3 x 3
Stride: 1 x 1

Camada de convolução - Filtro

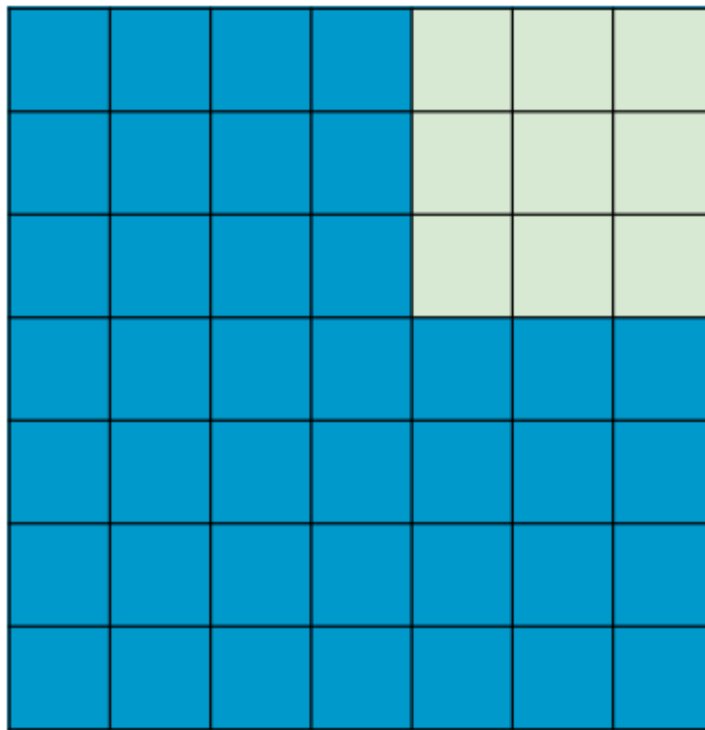
Slide e Stride



Entrada: 7 x 7
Filtro: 3 x 3
Stride: 1 x 1

Camada de convolução - Filtro

Slide e Stride



Entrada: 7 x 7

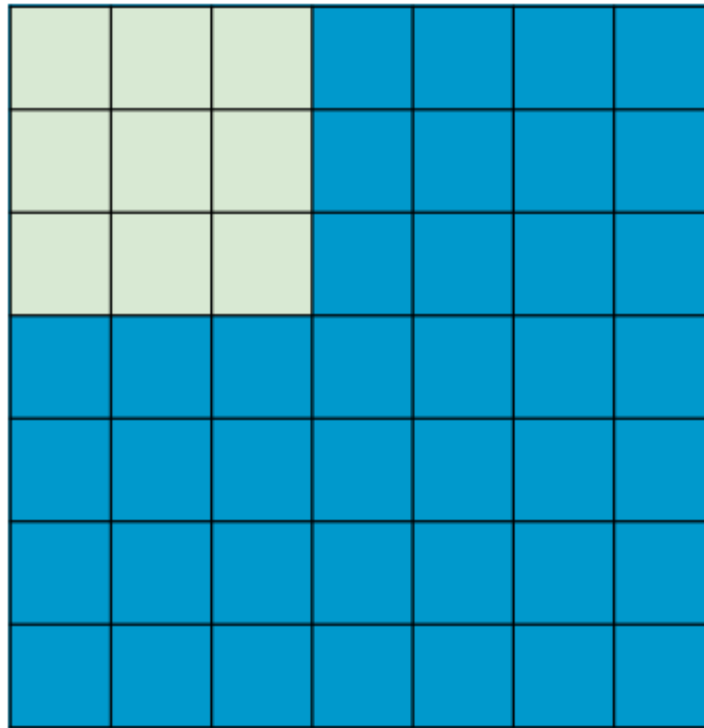
Filtro: 3 x 3

Stride: 1 x 1

Saída: 5 x 5

Camada de convolução - Filtro

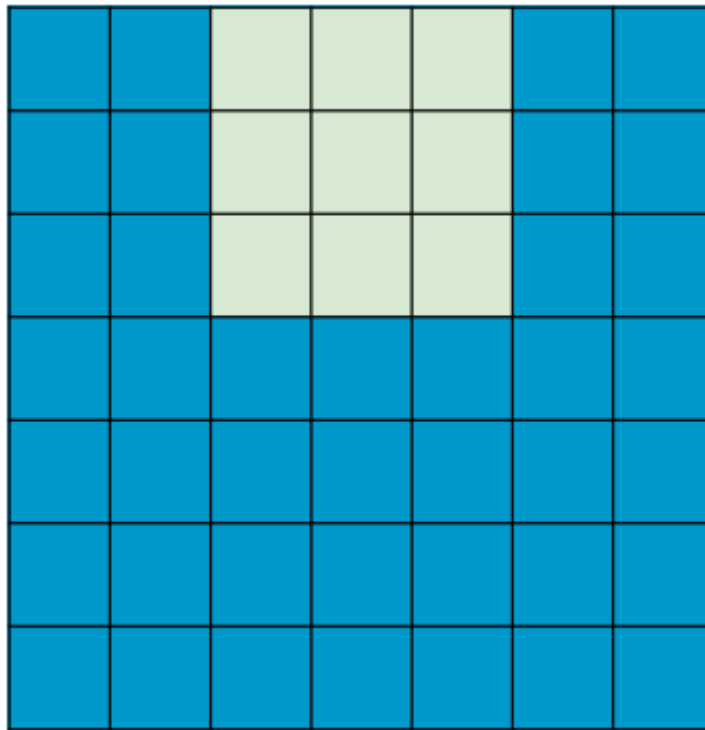
Slide e Stride



Entrada: 7 x 7
Filtro: 3 x 3
Stride: 2 x 2

Camada de convolução - Filtro

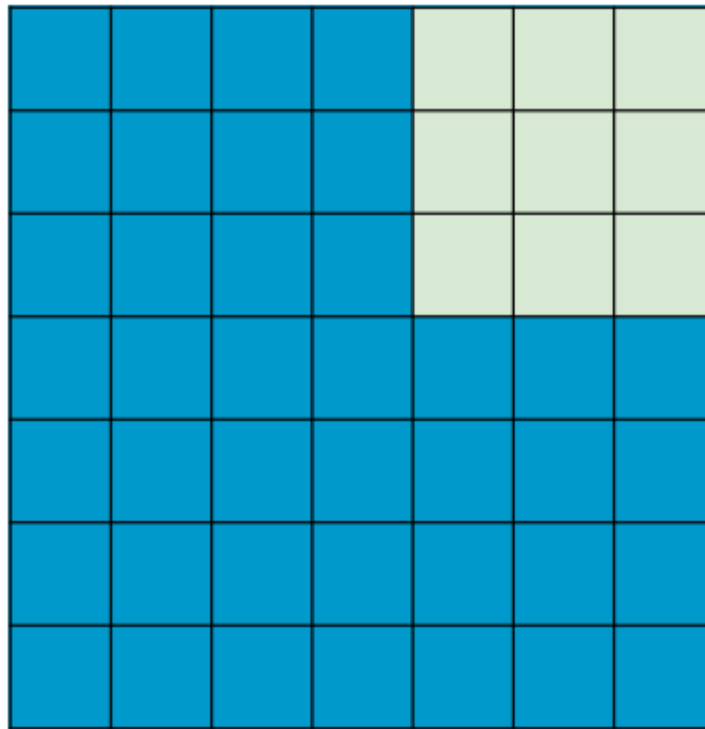
Slide e Stride



Entrada: 7 x 7
Filtro: 3 x 3
Stride: 2 x 2

Camada de convolução - Filtro

Slide e Stride



Entrada: 7 x 7

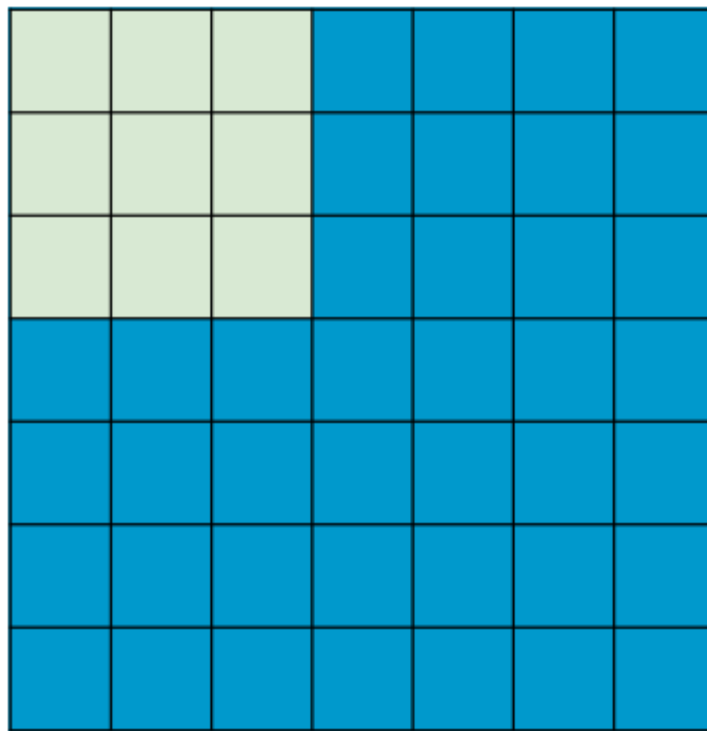
Filtro: 3 x 3

Stride: 2 x 2

Saída: 3 x 3

Camada de convolução - Filtro

Slide e Stride



Entrada: 7 x 7
Filtro: 3 x 3
Stride: 3 x 3

Camada de convolução - Filtro

Slide e Stride

- ▶ Para calcular o tamanho da nossa saída, utilizamos a seguinte fórmula.
 - ▶ $(N - F) / \text{stride} + 1$
 - ▶ Dado nosso exemplo imagem de entrada 7x7x1 e filtros 3x3
 - ▶ N (entrada) = 7
 - ▶ F (filtro) = 3
 - ▶ Sendo stride = 1: $(7 - 3) / 1 + 1 = 5$, output **7x7**
 - ▶ Sendo stride = 2: $(7 - 3) / 2 + 1 = 3$, output **3x3**
 - ▶ Sendo stride = 3: $(7 - 3) / 3 + 1 = 2.33$, output **Error!!!!!!!**

Camada de convolução - Filtro

- ▶ O que pode ser feito para resolver esse problema ?
- ▶ Entrada 7×7
- ▶ Filtro 3×3
- ▶ Stride 3×3



Camada de convolução - Filtro

Padding

0	0	0	0	0	0			
0								
0								
0								
0								

Fórmula para calcular o tamanho da saída:

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

Onde:

$$\text{Entrada} = 7 \times 7 = 7$$

$$\text{Filtro} = 3 \times 3 = 3$$

$$\text{Stride} = 3 \times 3 = 3$$

$$\text{Padding} = 1$$

$$\text{Saída} = (7 + 2 * 1 - 3) / 3 + 1 = 3 \times 3$$

Camada de convolução - Filtro Padding

- ▶ Entrada: 32 x 32 x 3
- ▶ 10 Filtros: 5 x 5
- ▶ Stride: 1 x 1
- ▶ Padding: 2 x 2

Logo:

$$(32 + 2 * 2 - 5) / 1 + 1 = 32$$

$$\text{Saída} = 32 \times 32 \times 10$$

Camadas da rede convolucional

- ▶ **Entrada (input):** Altura x Largura x Camadas, exemplo 16 x 16 x 3, onde:
 - ▶ Altura = 16 pixels
 - ▶ Largura = 16 pixels
 - ▶ Camadas = 3 (RGB)
- ▶ **Convolução:** É a camada que calculará a saída dos neurônios conectados as regiões locais da entrada.
- ▶ **RELU:** Função de ativação ponto a ponto, pode ser utilizada com max, min e avg.
- ▶ **Pool:** É uma camada downsampling, que visa corrigir os efeitos de escala.
- ▶ **Fully Connected (Classificador):** MLP, SVM, Softmax, etc.

Camada RELU

Função de ativação

$$f(x) = x^+ = \max(0, x)$$

or

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$



Camada RELU

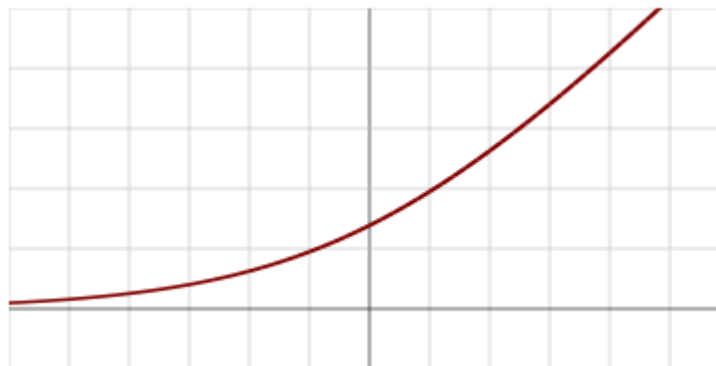
Outras funções de ativação

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



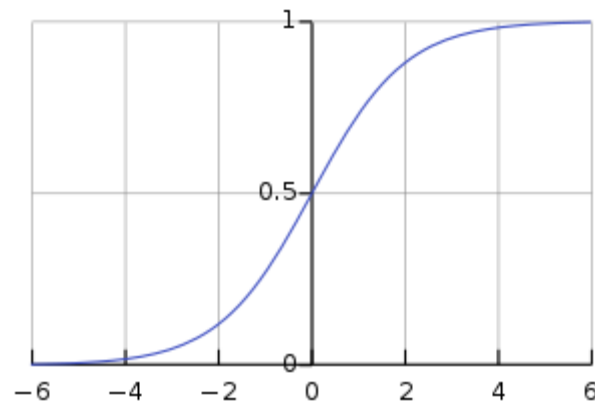
Softplus

$$f(x) = \log(1 + e^x)$$



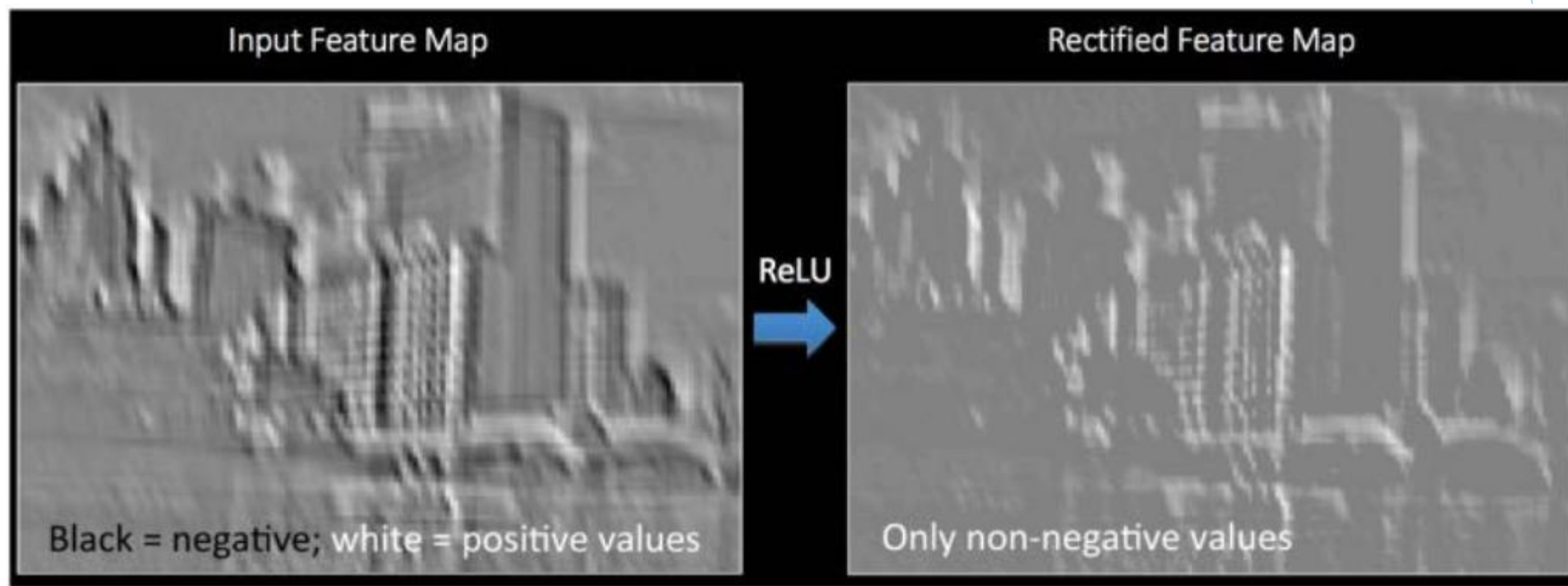
$$f(x) = \frac{1}{(1 + e^{-x})}$$

Sigmoid



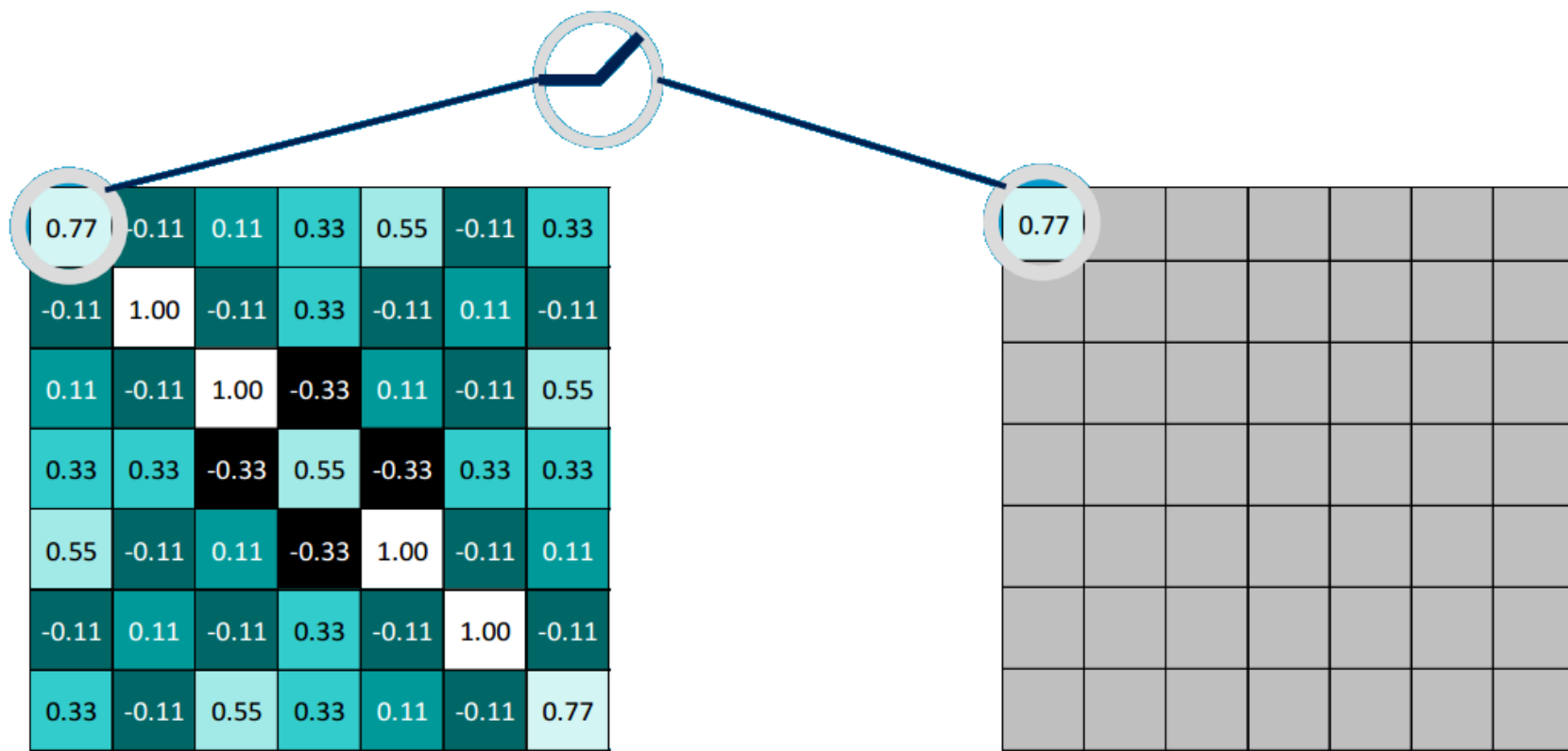
Camada RELU

Função de ativação - Exemplo



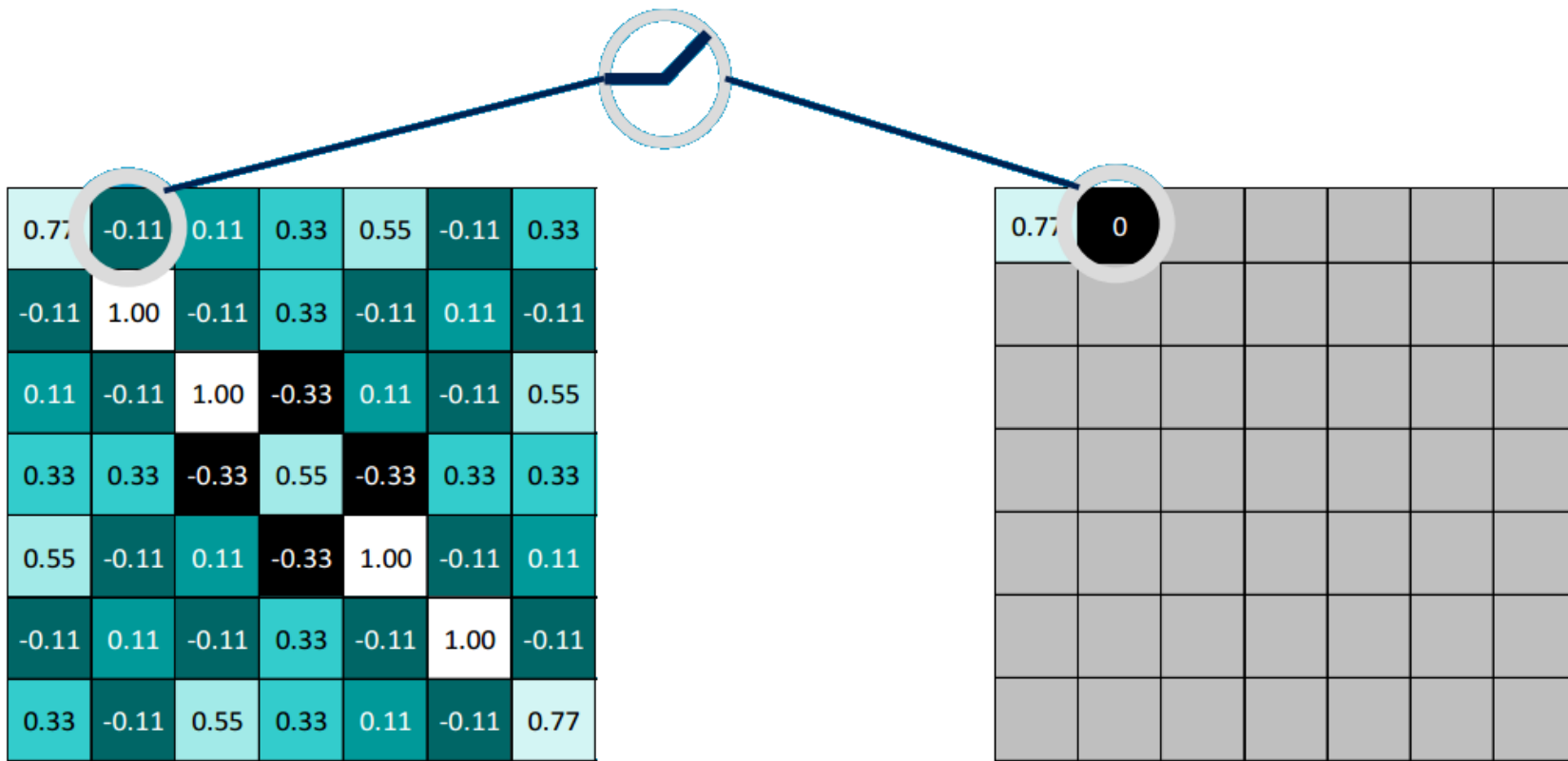
Camada RELU

Função de ativação

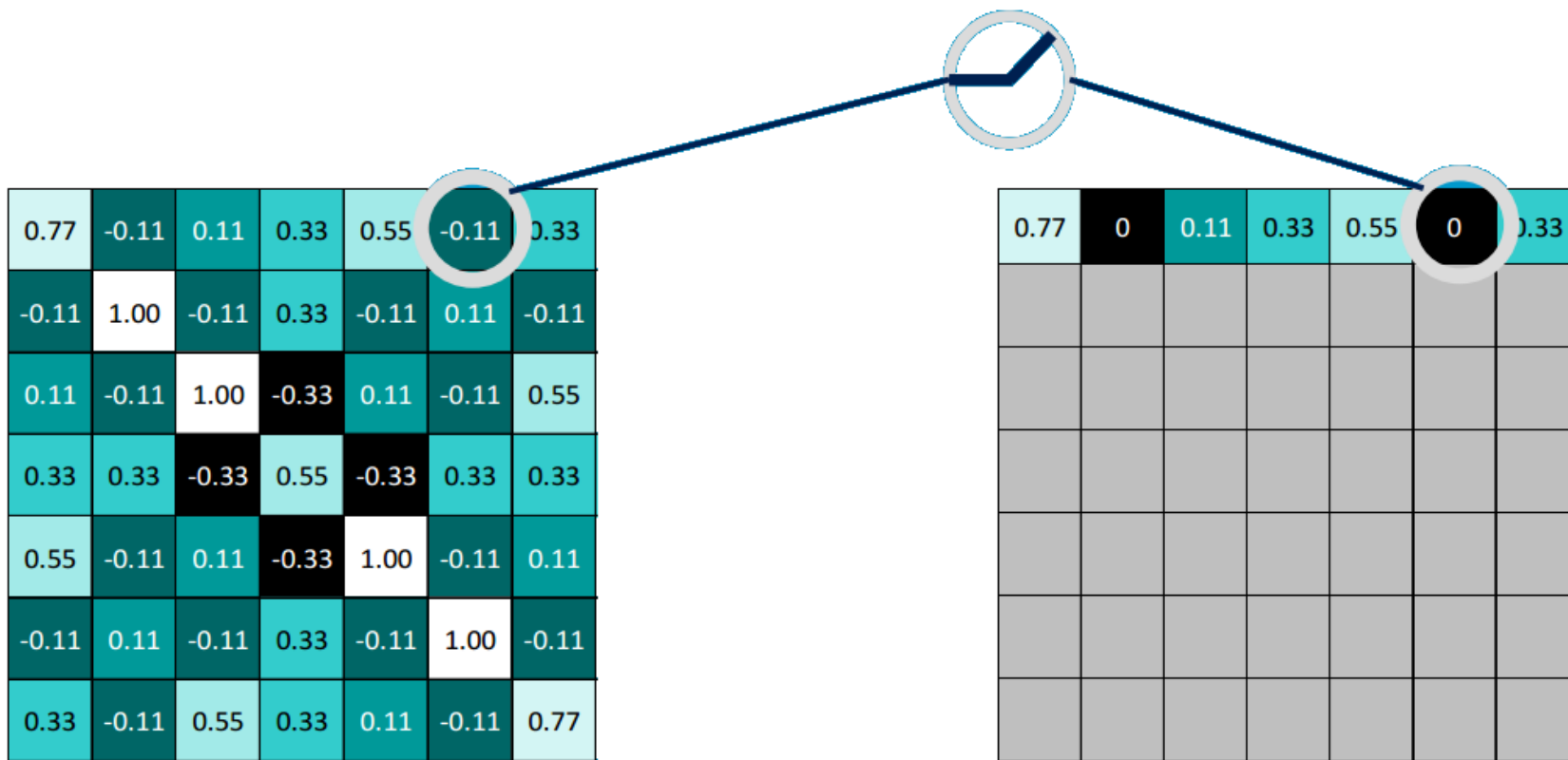


Camada RELU

Função de ativação



Função de ativação



Camada RELU

Função de ativação

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

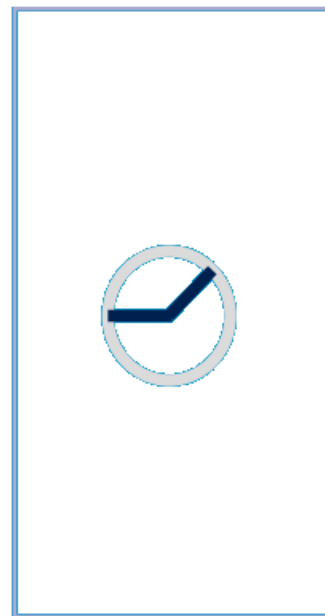


0.77	0	0.11	0.33	0.55	0	0.33
0	1.00	0	0.33	0	0.11	0
0.11	0	1.00	0	0.11	0	0.55
0.33	0.33	0	0.55	0	0.33	0.33
0.55	0	0.11	0	1.00	0	0.11
0	0.11	0	0.33	0	1.00	0
0.33	0	0.55	0.33	0.11	0	0.77

Camada RELU

Função de ativação

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



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

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

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

Camadas da rede convolucional

- ▶ **Entrada (input):** Altura x Largura x Camadas, exemplo 16 x 16 x 3, onde:
 - ▶ Altura = 16 pixels
 - ▶ Largura = 16 pixels
 - ▶ Camadas = 3 (RGB)
- ▶ **Convolução:** É a camada que calculará a saída dos neurônios conectados as regiões locais da entrada.
- ▶ **RELU:** Função de ativação ponto a ponto, pode ser utilizada com max, min e avg.
- ▶ **Pool:** É uma camada downsampling, que visa corrigir os efeitos de escala.
- ▶ **Classificador:** MLP, SVM, Softmax, etc.

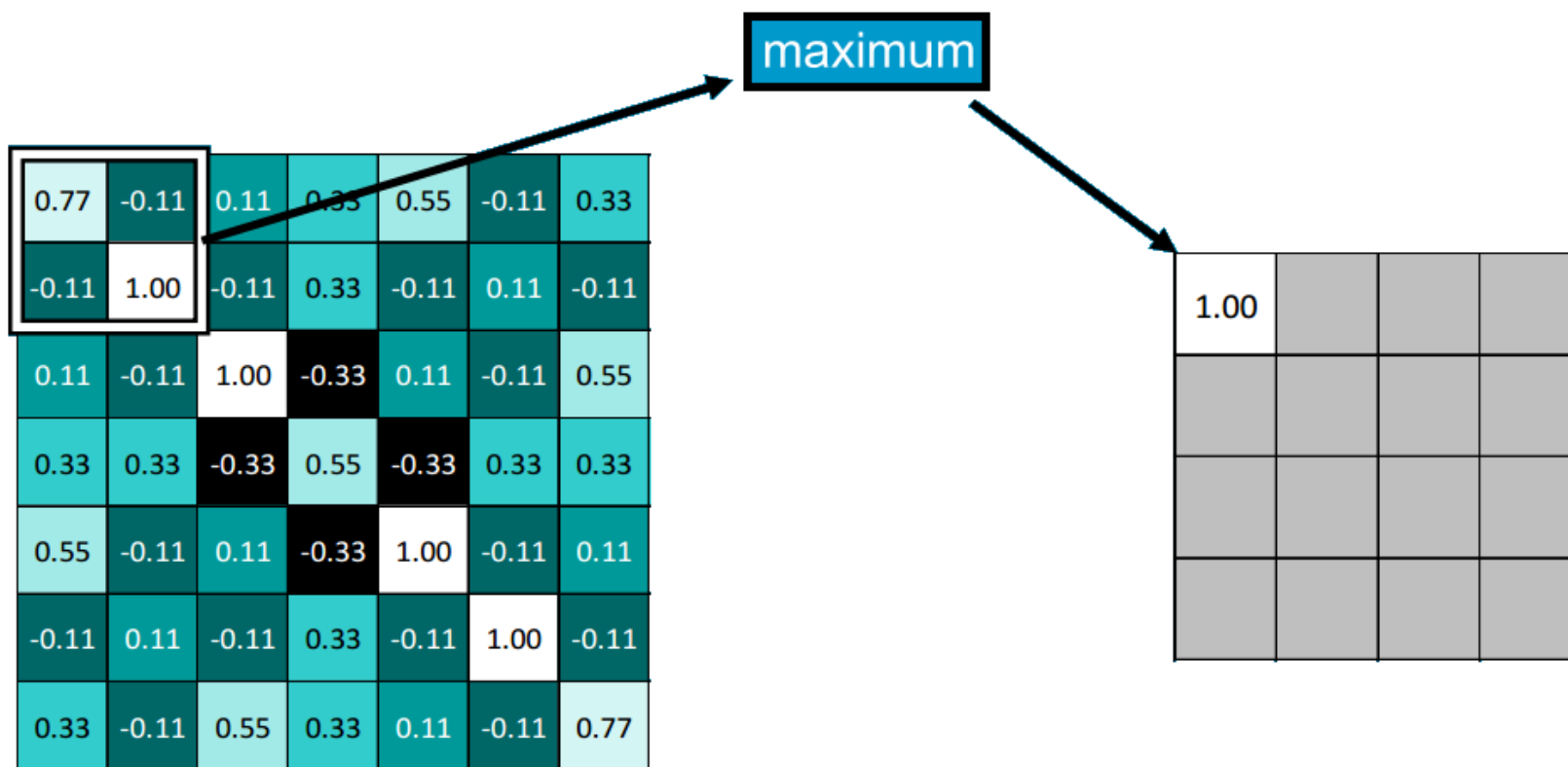
Camada Pool

- ▶ A camada pooling torna a convolução invariante a translação, rotação e janelamento.
- ▶ Pode ser usada com as funções Min, Max, Avg;
- ▶ A opção mais usada é o max-pooling;
- ▶ O objetivo dessa camada é destacar a maior ativação para propagar a região de interesse do campo receptivo;

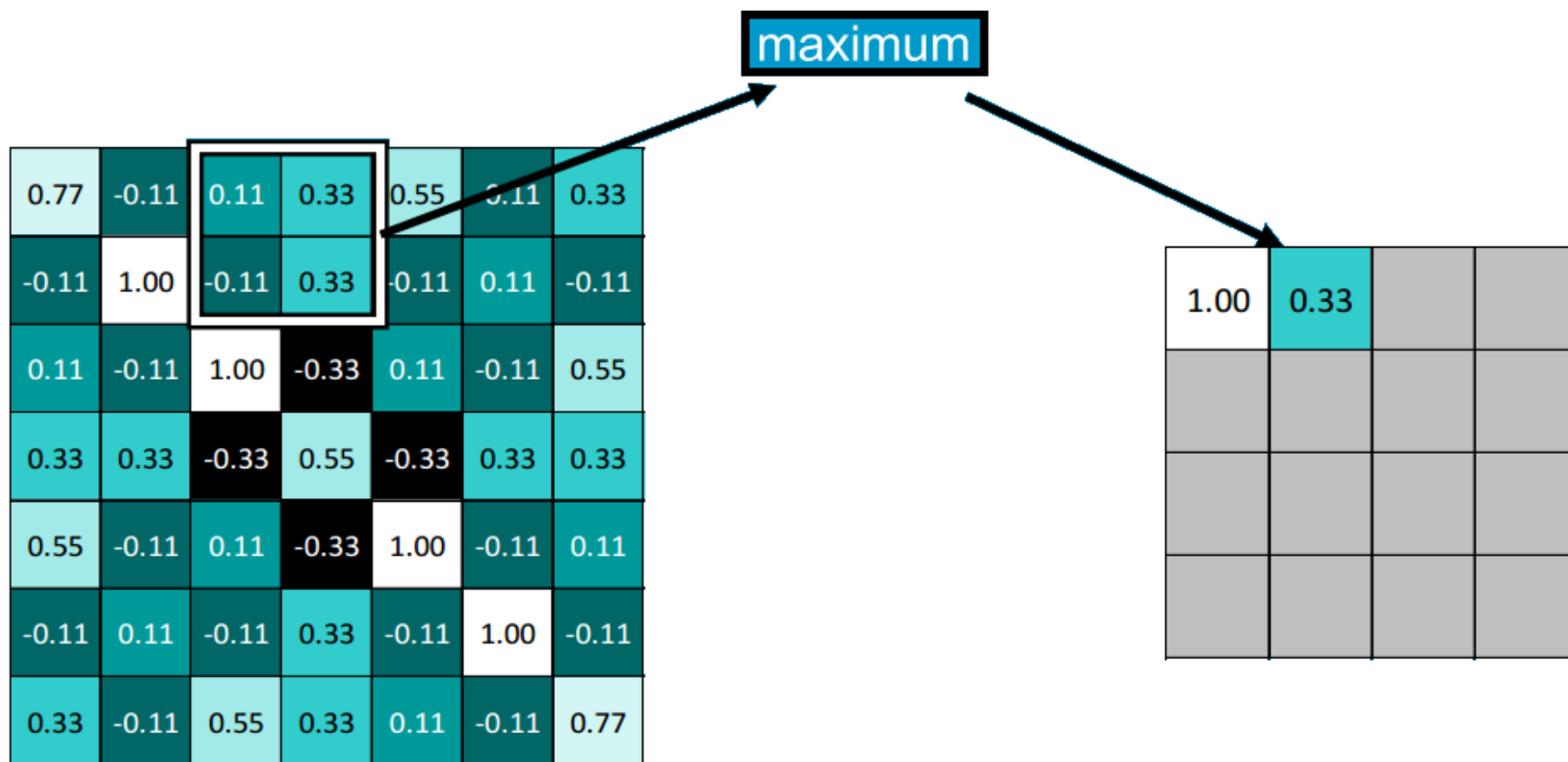
Camada Pool

- ▶ Passos para calcular a camada pooling.
 - ▶ Escolher o tamanho da janela.
 - ▶ Fazer o slide sobre a entrada.
 - ▶ Para cada janela pool, pegar o valor máximo (max-pooling)

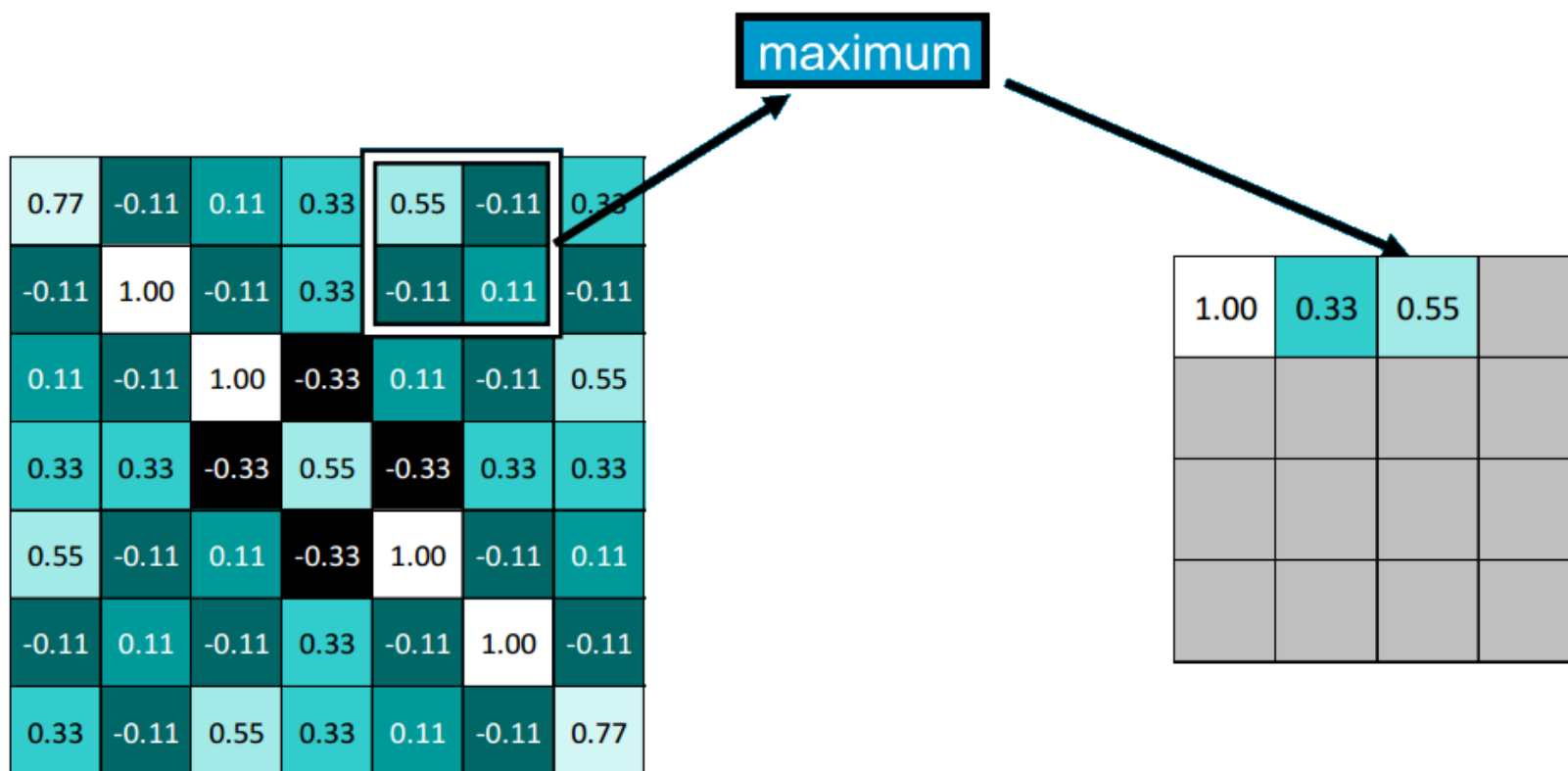
Camada Pool



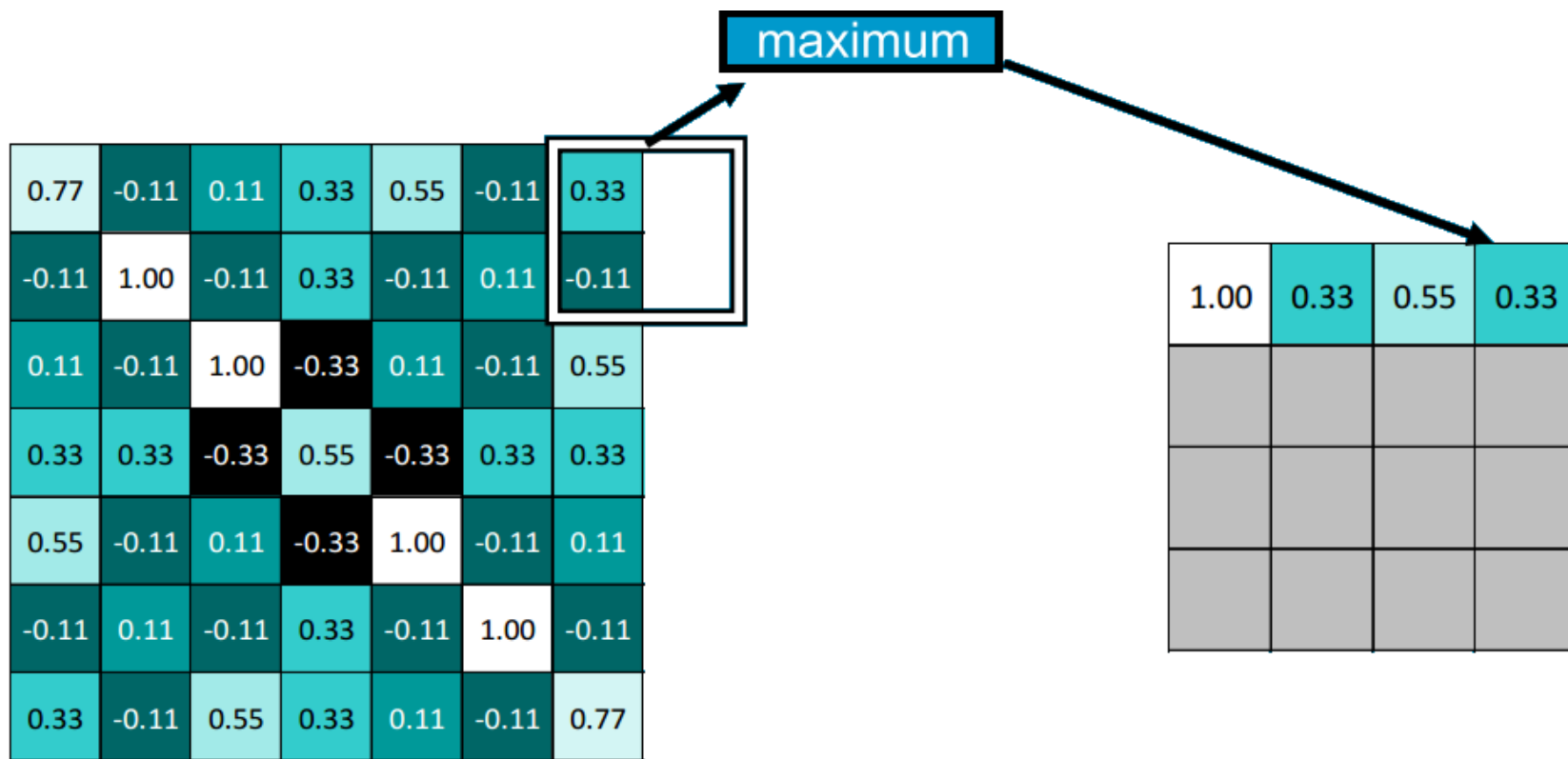
Camada Pool



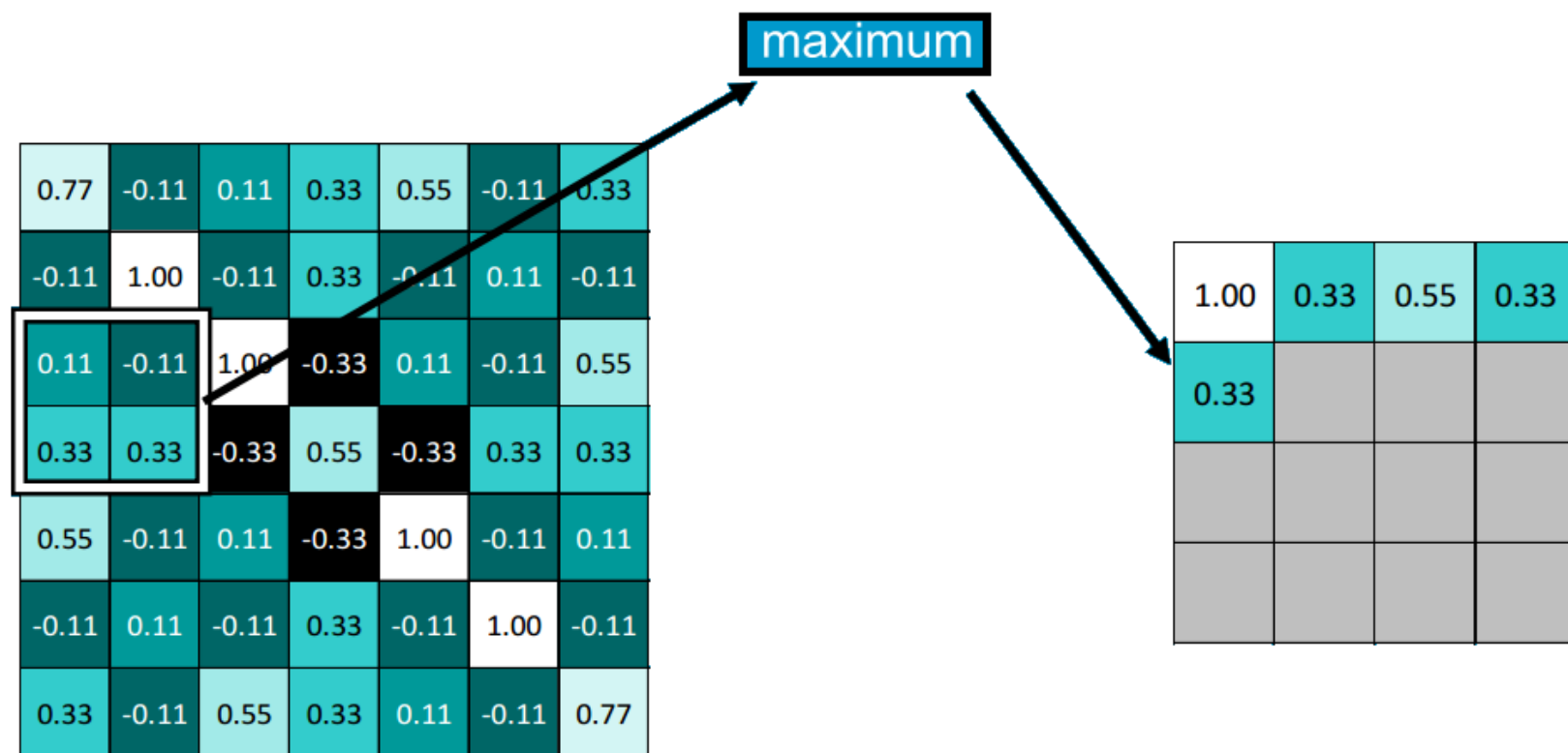
Camada Pool



Camada Pool



Camada Pool



Camada Pool

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

max pooling

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

Camada Pool

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

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

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



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

0.55	0.33	0.55	0.33
0.33	1.00	0.55	0.11
0.55	0.55	0.55	0.11
0.33	0.11	0.11	0.33

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

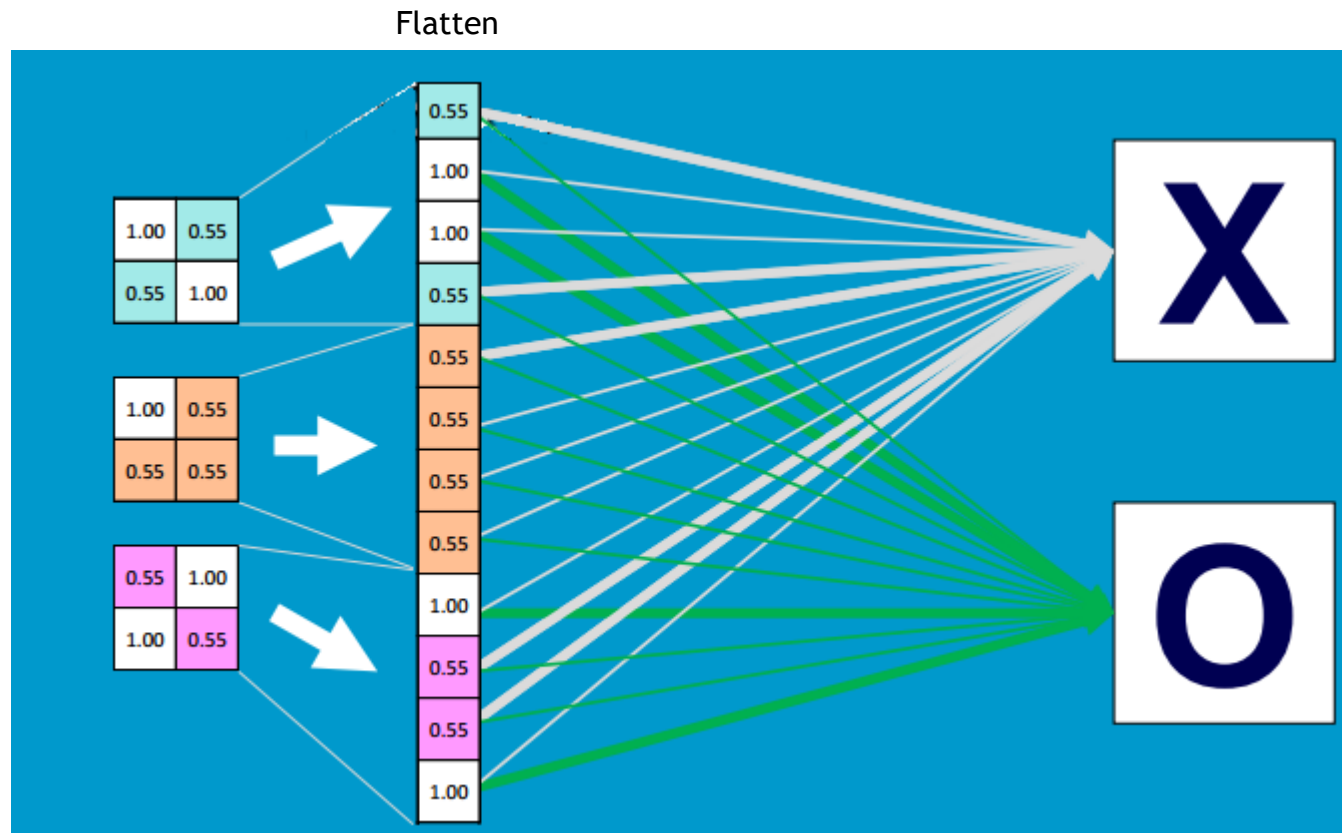
Camadas da rede convolucional

- ▶ **Entrada (input):** Altura x Largura x Camadas, exemplo 16 x 16 x 3, onde:
 - ▶ Altura = 16 pixels
 - ▶ Largura = 16 pixels
 - ▶ Camadas = 3 (RGB)
- ▶ **Convolução:** É a camada que calculará a saída dos neurônios conectados as regiões locais da entrada.
- ▶ **RELU:** Função de ativação ponto a ponto, pode ser utilizada com max, min e avg.
- ▶ **Pool:** É uma camada downsampling, que visa corrigir os efeitos de escala.
- ▶ **Fully Connected (Classificador):** MLP, SVM, Softmax, etc.

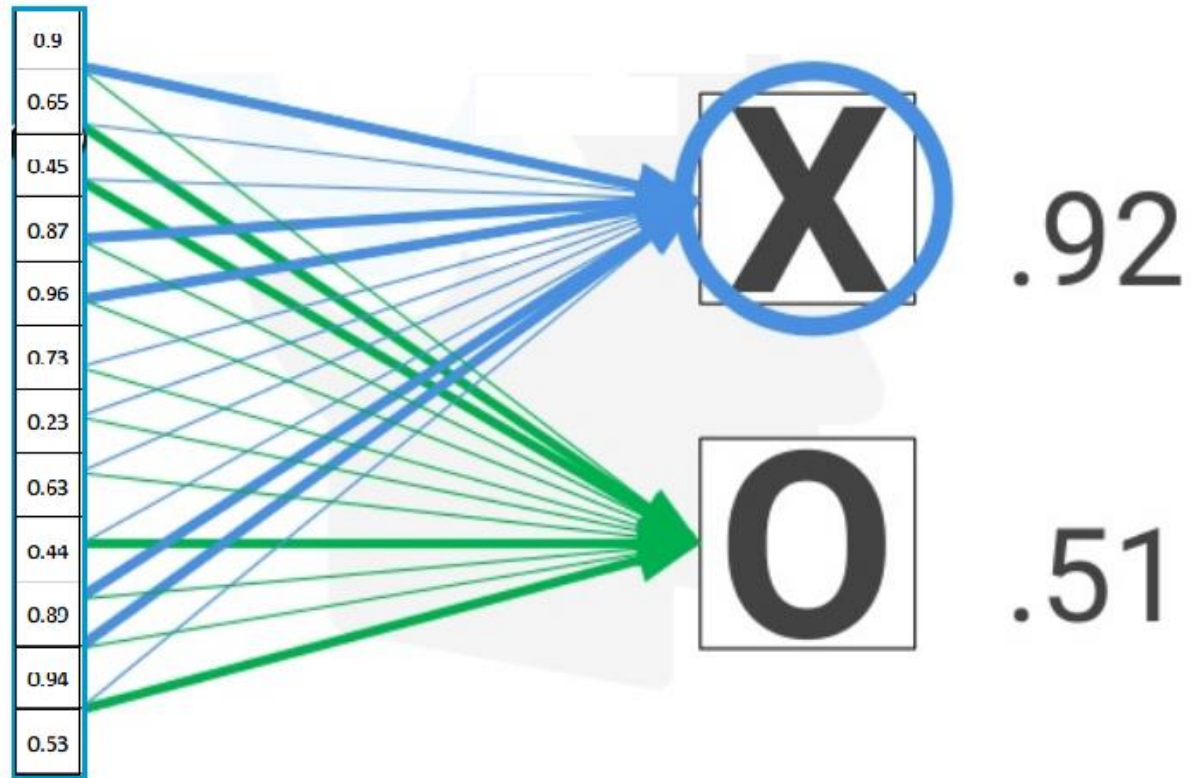
Camada Fully Connected (FC)

- ▶ Cada valor recebido da camada pooling expressa um valor.
- ▶ Todas as entradas são passadas para todos os neurônios.

Camada Fully Connected (FC)



Camada Fully Connected (FC)



Camada Fully Connected (FC)

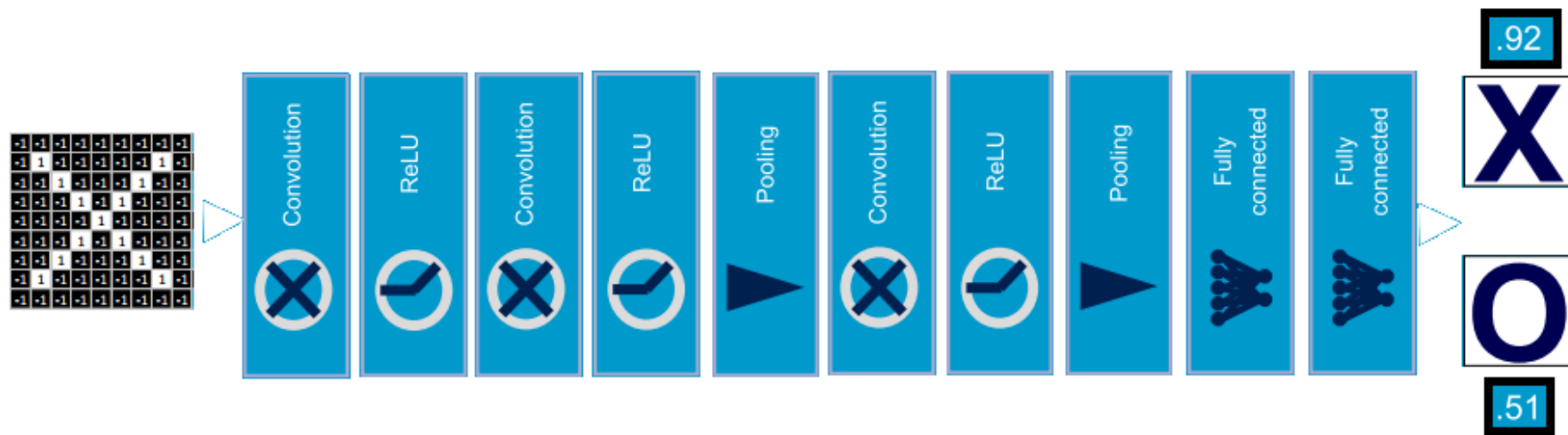
0.9
0.65
0.45
0.87
0.96
0.73
0.23
0.63
0.44
0.89
0.94
0.53



X

O

Rede Exemplo

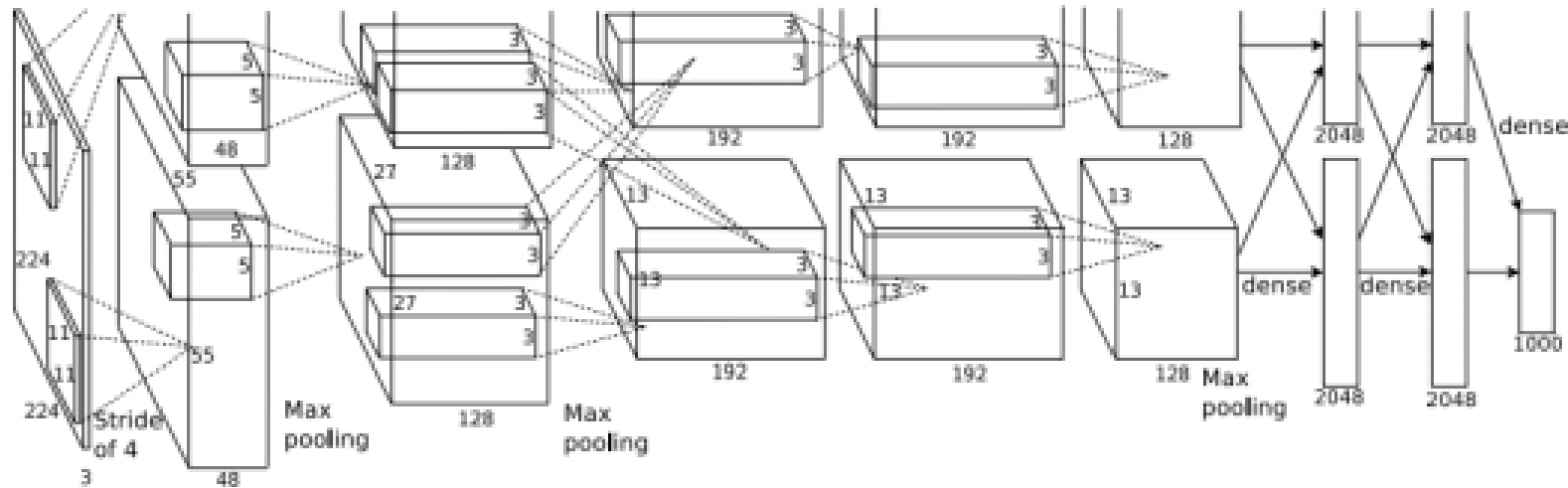


CNN

- ▶ Quantas camadas e de quais tipos precisamos ?
- ▶ Em que ordem ?
- ▶ Qual o tipo de filtro ?

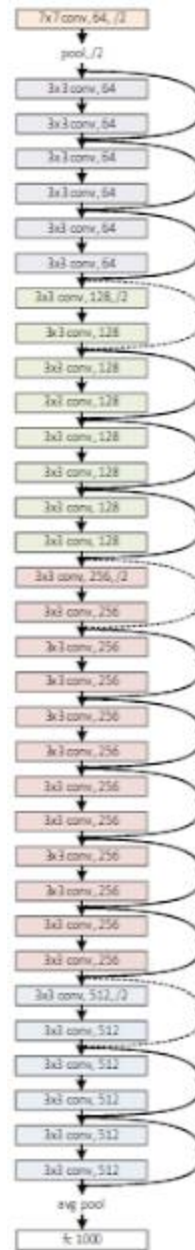
Redes conhecidas

Alexnet 2012



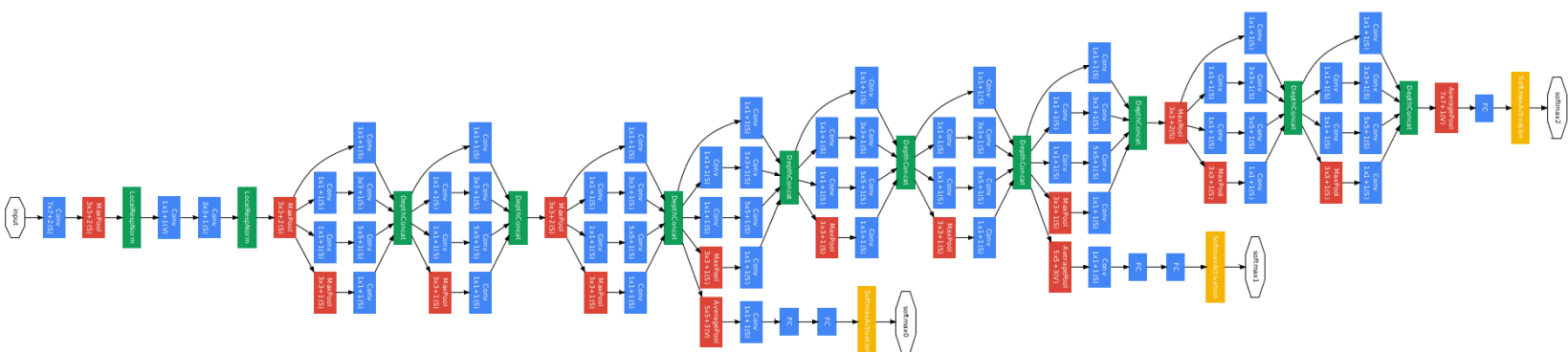
Redes conhecidas

ResNet



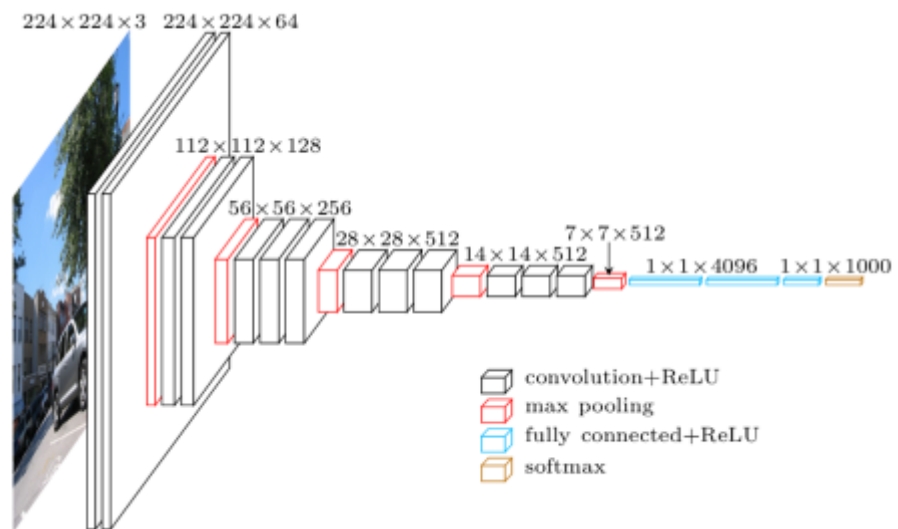
Redes conhecidas

Inception-v4



Redes conhecidas

Vgg16



A yellow circular emoji with a thinking expression. It has furrowed brows, large oval eyes, and a small frown. A hand is shown with the index finger pointing up to the chin, indicating deep thought or contemplation.[illegible]

Exercício

- ▶ Dado um input $15 \times 15 \times 1$
 - ▶ Calcular a convolução a partir dos filtros 3×3
 - ▶ Calcular a função de ativação RELU
 - ▶ Calcular a camada max-pooling
 - ▶ Fazer o flatten para o classificador.
-
- ▶ Outras informações
 - ▶ Não usar padding
 - ▶ Utilizar o stride 3×3
 - ▶ Utilizar a janela do pooling 2×2

Exercício

Input															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
2	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
3	-1	-1	-1	-1	-1	1	-1	-1	-1	1	-1	-1	-1	-1	-1
4	-1	-1	-1	-1	-1	1	1	-1	1	1	-1	-1	-1	-1	-1
5	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	1	1	-1	1	1	-1	-1	-1	-1	-1
7	-1	-1	-1	-1	-1	1	-1	-1	-1	1	-1	-1	-1	-1	-1
8	-1	-1	-1	-1	1	-1	-1	-1	-1	-1	1	-1	-1	-1	-1
9	-1	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1	-1
10	-1	-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	-1
11	-1	1	1	1	1	1	1	1	1	1	1	1	1	1	-1
12	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
13	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
14	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
15	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

Filtro - Reta

	1	2	3
1	-1	-1	-1
2	1	1	1
3	-1	-1	-1

Filtro - Diag. Dir

	1	2	3
1	1	-1	-1
2	-1	1	-1
3	-1	-1	1

Filtro - Diag. Esq

	1	2	3
1	-1	-1	1
2	-1	1	-1
3	1	-1	-1

Obrigado pela atenção !!!



THANK YOU

ESKERIK ASKO
DANKIE
MERCI
GRACIAS
DANKE
GRATIAS AGIMUS TIBI
TAKKGRÀCIES
DIOLCH DZIĘKUJĘ
TAK FALEMINDERIT

謝謝
OBRIGADO
THANK YOU
DANKIE
GRACIAS
MERCI
GRACIAS
DANKE
GRATIAS AGIMUS TIBI
TAKKGRÀCIES
DIOLCH DZIĘKUJĘ
TAK FALEMINDERIT

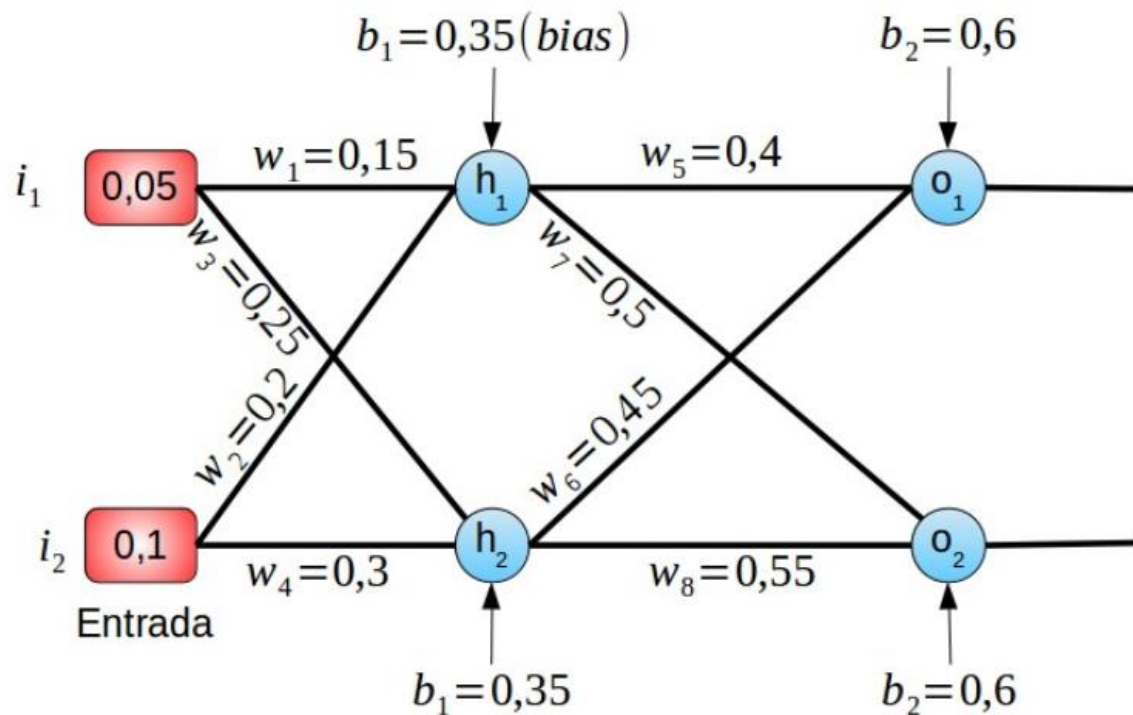
ДЗЯКУЙ
ESKERIK ASKO
DANKIE
MERCI
GRACIAS
DANKE
GRATIAS AGIMUS TIBI
TAKKGRÀCIES
DIOLCH DZIĘKUJĘ
TAK FALEMINDERIT

پاک کا پیر
DěKUJI
شکرا
ありがとう
спасибо

வினியாசம்
வினியாசம்

Exercício MLP

- ▶ Criar um programa python para calcular a MLP abaixo. Somente poderá ser utilizado python e numpy.



Exercício MLP

Fórmulas

Calculando a camada oculta (H)

Parâmetros

x_i = entradas

w_i = pesos

b_i = bias

h = camada oculta

NET_{hi} = propagação dos pesos da camada

$g(h_i)$ = saídas da camada H

Fórmula de propagação das entradas

$$NET_{h1} = x_1 \cdot w_1 + x_2 \cdot w_2 + b_1 \cdot 1$$

$$NET_{h2} = x_1 \cdot w_3 + x_2 \cdot w_4 + b_1 \cdot 1$$

Aplicando a função de ativação

$$g(h_1) = \frac{1}{(1 + e^{-NET_{h1}})}$$

$$g(h_2) = \frac{1}{(1 + e^{-NET_{h2}})}$$

Exercício MLP

Fórmulas

Calculando a camada de saída (O)

Parâmetros

$g(h_i)$ = entradas

w_i = pesos

b_i = bias

o = camada de saída

NET_{oi} = propagação dos pesos da camada

$g(o_i)$ = saídas da camada O

Fórmula de propagação das entradas

$$NET_{o1} = g(h_1) \cdot w_5 + g(h_2) \cdot w_6 + b_2 \cdot 1$$

$$NET_{o2} = g(h_1) \cdot w_7 + g(h_2) \cdot w_8 + b_2 \cdot 1$$

Aplicando a função de ativação

$$g(h_1) = \frac{1}{(1 + e^{-NET_{o1}})}$$

$$g(h_2) = \frac{1}{(1 + e^{-NET_{o2}})}$$

Exercício MLP

Fórmulas

Calculando os erros

Parâmetros

O = camada de saída

$g(o_i)$ = saídas da camada O

d_i = valor desejado

Erro Total

$$E_{Total} = \frac{1}{2} \cdot \sum_{k=1} (d_k - g(o_k))^2$$

Erro da saída o_1

$$E_{o1} = \frac{1}{2} \cdot (d_1 - g(o_1))^2$$

Erro da saída o_2

$$E_{o2} = \frac{1}{2} \cdot (d_2 - g(o_2))^2$$

Exercício MLP

Fórmulas

Atualizando os pesos da camada de saída(Back Propagation)

Parâmetros

o = camada de saída

n = taxa de aprendizado

$g(o_i)$ = saídas da camada O

$g(h_i)$ = saídas da camada H

d_i = valor desejado

Atualizando o w_5 e w_6

$$\frac{\delta E_{Total}}{\delta w_{5,6}} = \frac{\delta E_{Total}}{\delta g(o_1)} \cdot \frac{\delta g(o_1)}{\delta NET_{o1}} \cdot \frac{\delta NET_{o1}}{\delta w_{5,6}}$$

Derivada da primeira parcela

$$\frac{\delta E_{Total}}{\delta g(o_1)} = -(d_1 - g(o_1))$$

Derivada da segunda parcela

$$\frac{\delta g(o_1)}{\delta NET_{o1}} = g(o_1) \cdot (1 - g(o_1))$$

Derivada da terceira parcela

$$\frac{\delta NET_{o1}}{\delta w_{5,6}} = g(h_1)$$

Logo

$$\frac{\delta E_{Total}}{\delta w_{5,6}} = -(d_1 - g(o_1)) \cdot g(o_1) \cdot (1 - g(o_1)) \cdot g(h_1)$$

Atualiza os pesos w_5 e w_6

$$w_{5,6} = w_{5,6} - \frac{n \cdot \delta E_{Total}}{\delta w_{5,6}}$$

Para calcular w_7 e w_8 basta trocar:

$$g(o_1) > g(o_2)$$

$$NET_{o1} > NET_{o2}$$

$$w_{5,6} > w_{7,8}$$



Exercício MLP

Fórmulas

Logo

$$\frac{\delta E_{Total}}{\delta w_{1,2}} = (((g(o_1) - d_1)) \cdot g(o_1) \cdot (1 - g(o_1))) \cdot w_5 +$$
$$(((g(o_2) - d_2)) \cdot g(o_2) \cdot (1 - g(o_2))) \cdot w_7).$$
$$g(h_1) \cdot (1 - g(h_1)) \cdot x_1$$

Atualiza os pesos w_1 e w_2

$$w_{1,2} = w_{1,2} - \frac{n \cdot \delta E_{Total}}{\delta w_{1,2}}$$

Para calcular w_3 e w_4 basta trocar

$$g(h_1) > g(h_2)$$
$$NET_{h1} > NET_{h2}$$
$$w_{1,2} > w_{3,4}$$
$$w_5 > w_6$$
$$w_7 > w_8$$
$$x_1 > x_2$$



Atualizando os pesos da camada oculta (Back Propagation)

Parâmetros

o = camada de saída

n = taxa de aprendizado

x_i = entrada da camada H

$g(h_i)$ = saídas da camada H

d_i = valor desejado

w_i = pesos

Atualizando o w_1 e w_2

$$\frac{\delta E_{Total}}{\delta w_{1,2}} = \frac{\delta E_{Total}}{\delta g(h_1)} \cdot \frac{\delta g(h_1)}{\delta NET_{h1}} \cdot \frac{\delta NET_{h1}}{\delta w_{1,2}}$$

Derivada da primeira parcela

$$\frac{\delta E_{Total}}{\delta g(h_1)} = (((g(o_1) - d_1)) \cdot g(o_1) \cdot (1 - g(o_1))) \cdot w_5 + (((g(o_2) - d_2)) \cdot g(o_2) \cdot (1 - g(o_2))) \cdot w_7$$

Derivada da segunda parcela

$$\frac{\delta g(h_1)}{\delta NET_{h1}} = g(h_1) \cdot (1 - g(h_1))$$

Derivada da terceira parcela

$$\frac{\delta NET_{h1}}{\delta w_{1,2}} = x_1$$



Obrigado

[illegible]