REDES NEURAIS ARTIFICIAIS APRENDIZADO PROFUNDO (2)

PROFA. ROSELI AP. FRANCELIN ROMERO

SCC – ICMC - USP

REDES NEURAIS ARTIFICIAIS

- PERCEPTRONS
- 1970 – 2000 – IA ⬅➡ SISTEMAS ESPECIALISTAS
- 1970 – 2010 – DESENVOLVIMENTO DA TEORIA DE APRENDIZADO DE MÁQUINA

➢McClelland & PDP Group (1986) – ALGORITMO BACK-PROPAGATION

➢Fukushima´s Neocognitron (1989)

➢LeCun´s LeNet (1998) – “nova primavera” surgiu

INÍCIO

NEW NAVY DEVICE LEARNS BY DOING

Psychologist Shows Embryo of Computer Designed to Read and Grow Wiser

WASHINGTON, July 7 (UPI) —The Navy revealed the embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.

The embryo—the Weather Bureau's $2,000,000 "704" computer—learned to differentiate between right and left after fifty attempts in the Navy's demonstration for newsmen.

The service said it would use this principle to build the first of its Perceptron thinking machines that will be able to read and write. It is expected to be finished in about a year at a cost of $100,000.

Dr. Frank Rosenblatt, designer of the Perceptron, conducted the demonstration. He said the machine would be the

ings, Perceptron will make mistakes at first, but will grow wiser as it gains experience, he said.

Dr. Rosenblatt, a research psychologist at the Cornell Aeronautical Laboratory, Buffalo, said Perceptrons might be fired to the planets as mechanical space explorers.

Without Human Controls

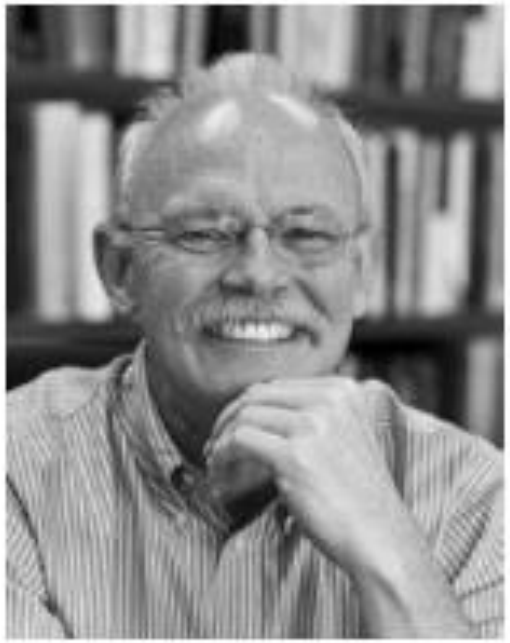
The Navy said the perceptron would be the first non-living mechanism "capable of receiving, recognizing and identifying its surroundings without any human training or control."

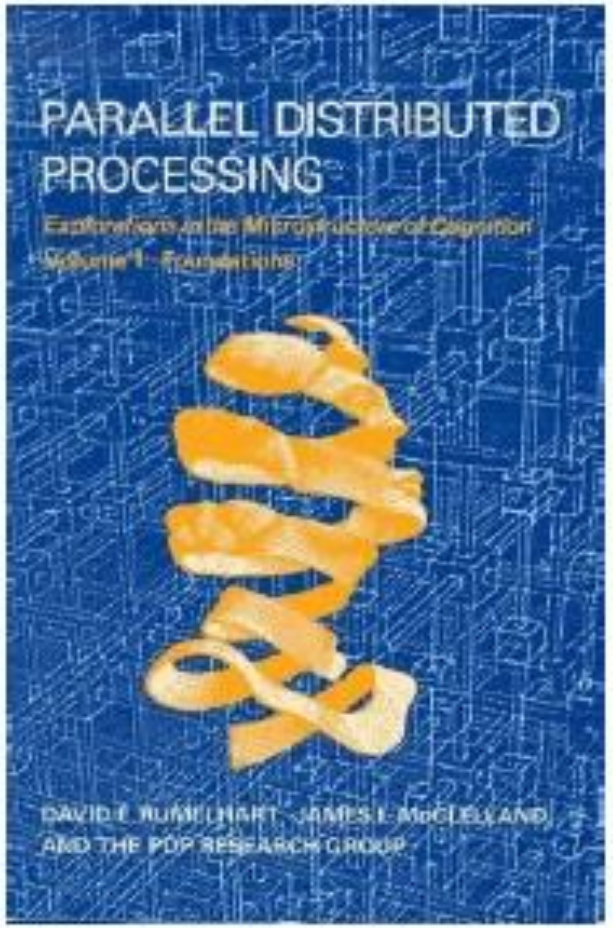
The "brain" is designed to remember images and information it has perceived itself. Ordinary computers remember only what is fed into them on punch cards or magnetic tape.

Later Perceptrons will be able to recognize people and call out their names and instantly translate speech in one language to speech or writing in another language, it was predicted.

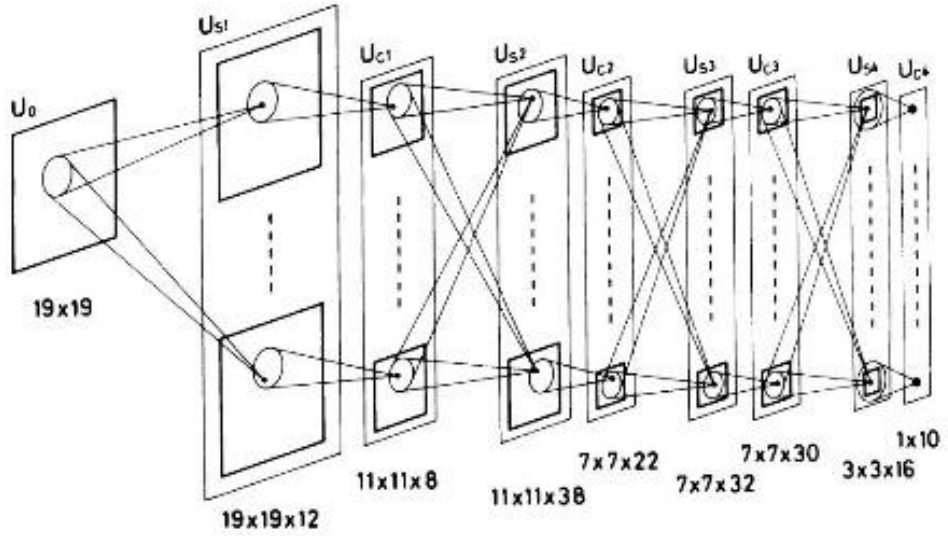
Mr. Rosenblatt said in principle it would be possible to build brains that could reproduce themselves on an assembly

BACK-PROPAGATION (1986)
REDES MULTI-CAMADAS (MLP)

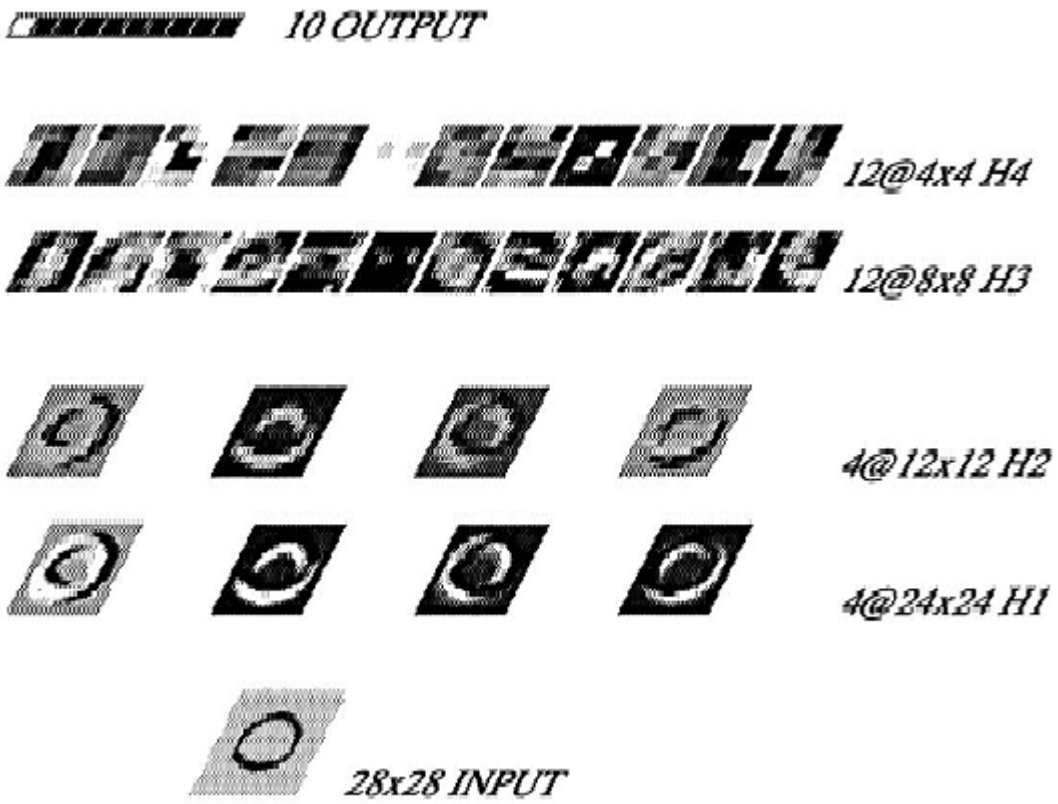




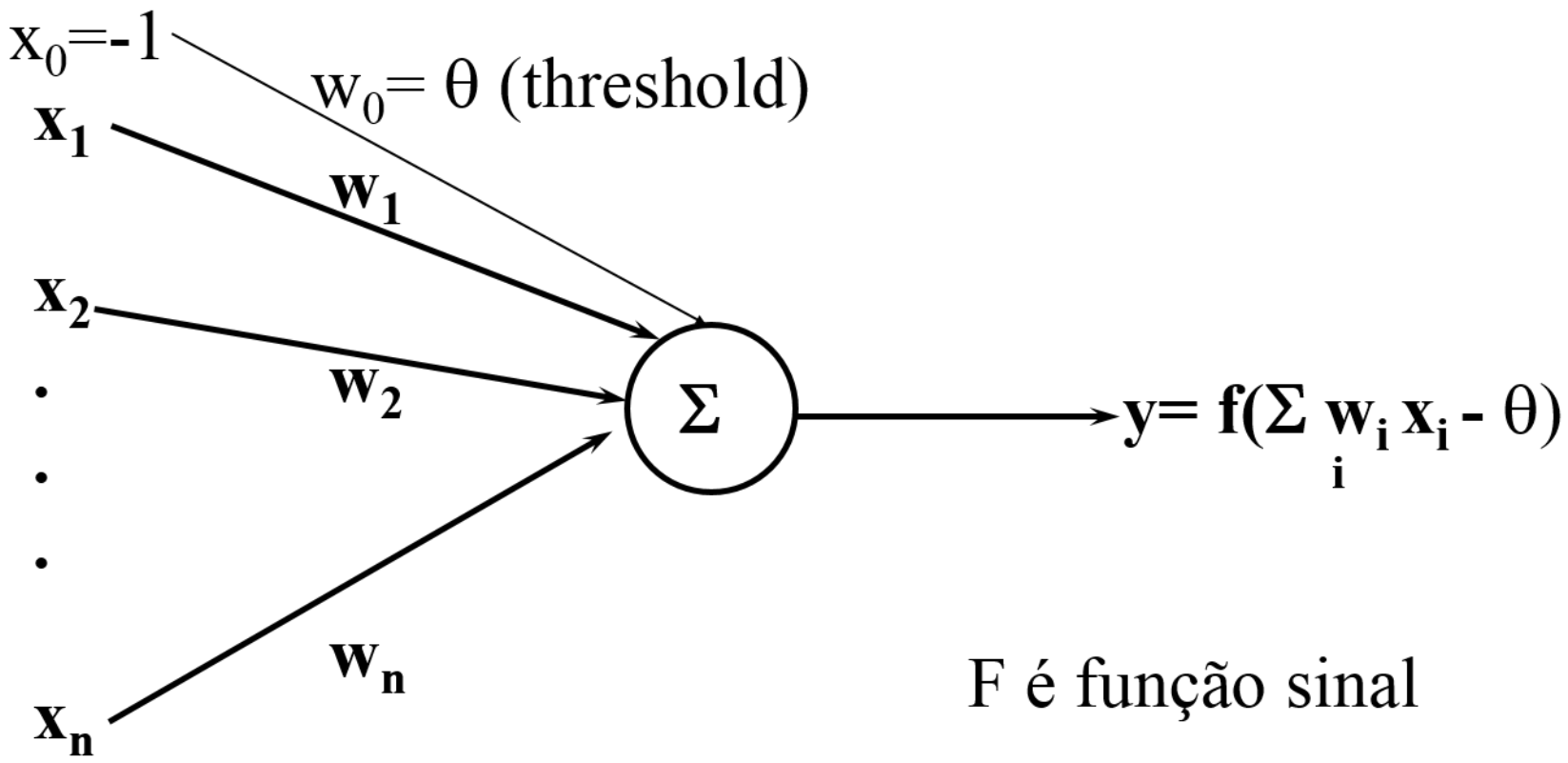
NEOCOGNITRON (FUKUSHIMA – 1989)



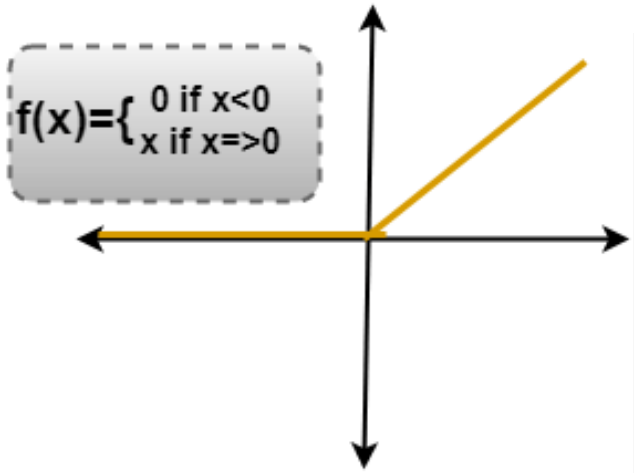
LENET – LeCun (1998)



Modelo de um neurônio simples



FUNÇÕES DE ATIVAÇÃO



Sigmoid

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

tanh

$\tanh(x)$

ReLU

$\max(0, x)$

Leaky ReLU

$\max(0.1x, x)$

Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

O que é Aprendizado?

"Aprendizado é um processo pelo qual os parâmetros livres de uma rede neural são adaptados através de um processo contínuo de estímulos pelo ambiente no qual a rede está incorporada".

W e B = ?

Aprendizado em RNA

Durante o processo de aprendizado:

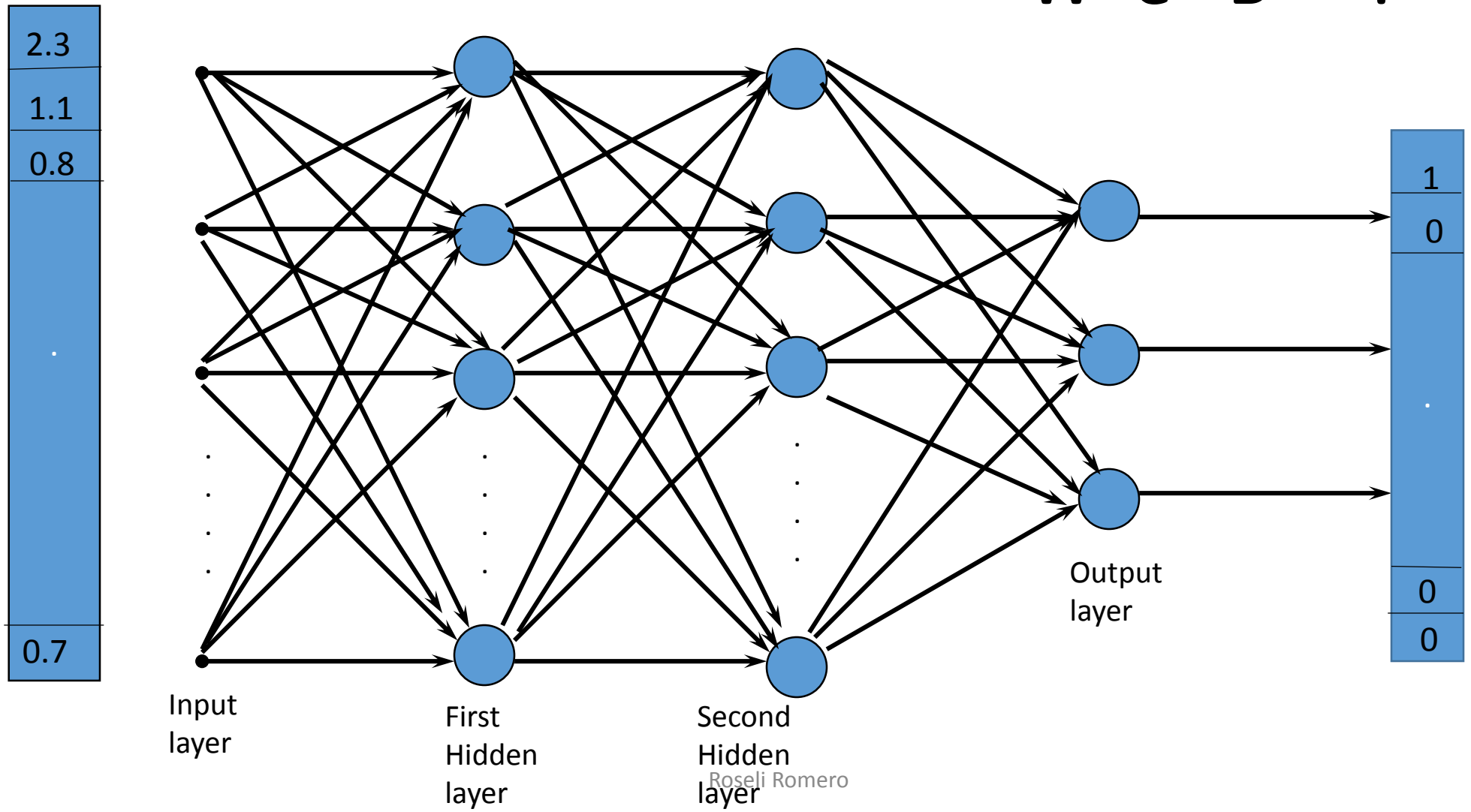
1- A rede neural é estimulada por um ambiente.

2 - A rede muda com o resultado deste estímulo

3 - A rede responde de um novo modo ao ambiente, em função das mudanças que ocorreram na sua estrutura interna.

Modelo de Rede Neural com Multiplas Camadas - MLP

W e B = ?



II - Algoritmo BackPropagation

OBJETIVO

- Encontrar um conjunto de pesos $\{W_J\},\{B_J\}$, para

  MINIMIZAR $\sum_i (y_i - \text{Out}(\underline{x}_i))^2$

pelo método do "gradiente descendente".

OBS: Convergência para um MINIMO global não é garantida.

Na prática: não é problema!!!

II - Algoritmo Back-Propagation

$$\Delta w_{jk} = -\eta \delta_j{}^p out_k{}^p$$

- Se o neurônio está na camada de saída

$$\delta_{pj} = (y_j{}^p - out_j{}^p) f'(net_j{}^p) \qquad net_j^p = \sum_k w_{jk} out_k{}^p$$

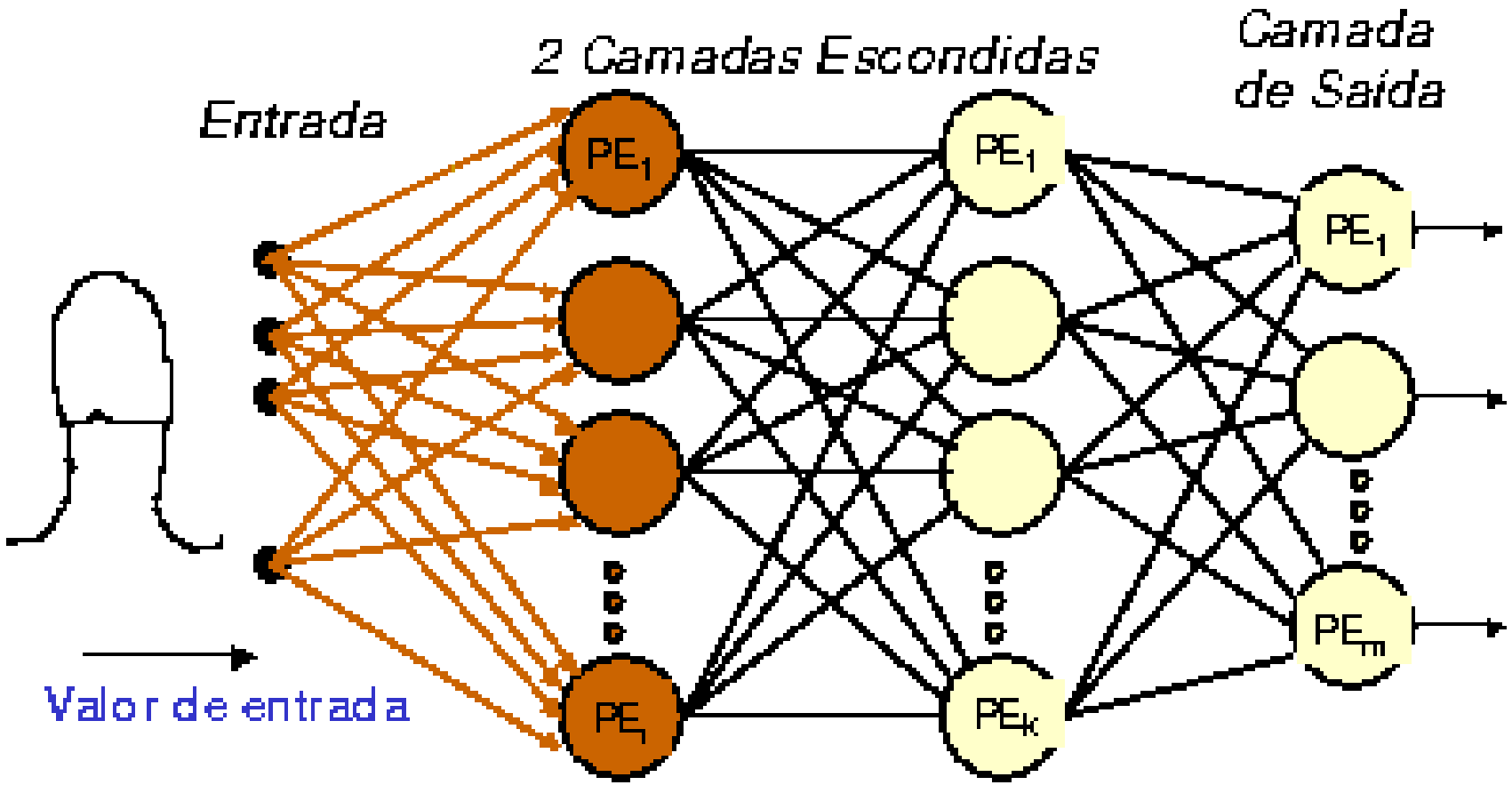
- Se o neurônio está na camada oculta

$$\delta_{pj} = f'(net_j{}^p) \sum_k \delta_k{}^p w_{kj}$$

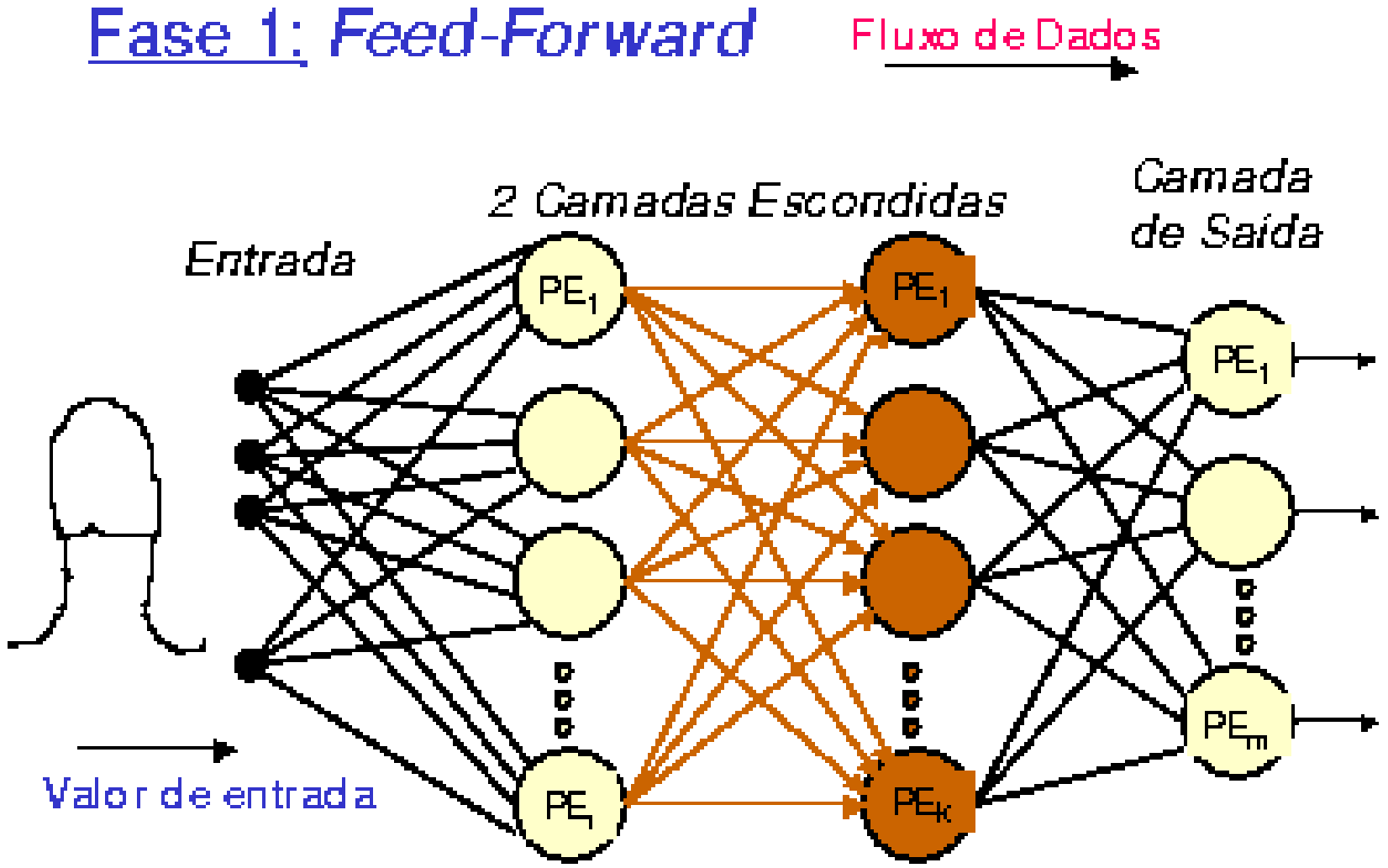
PROCESSO DE APRENDIZADO

Fase 1: *Feed-Forward*

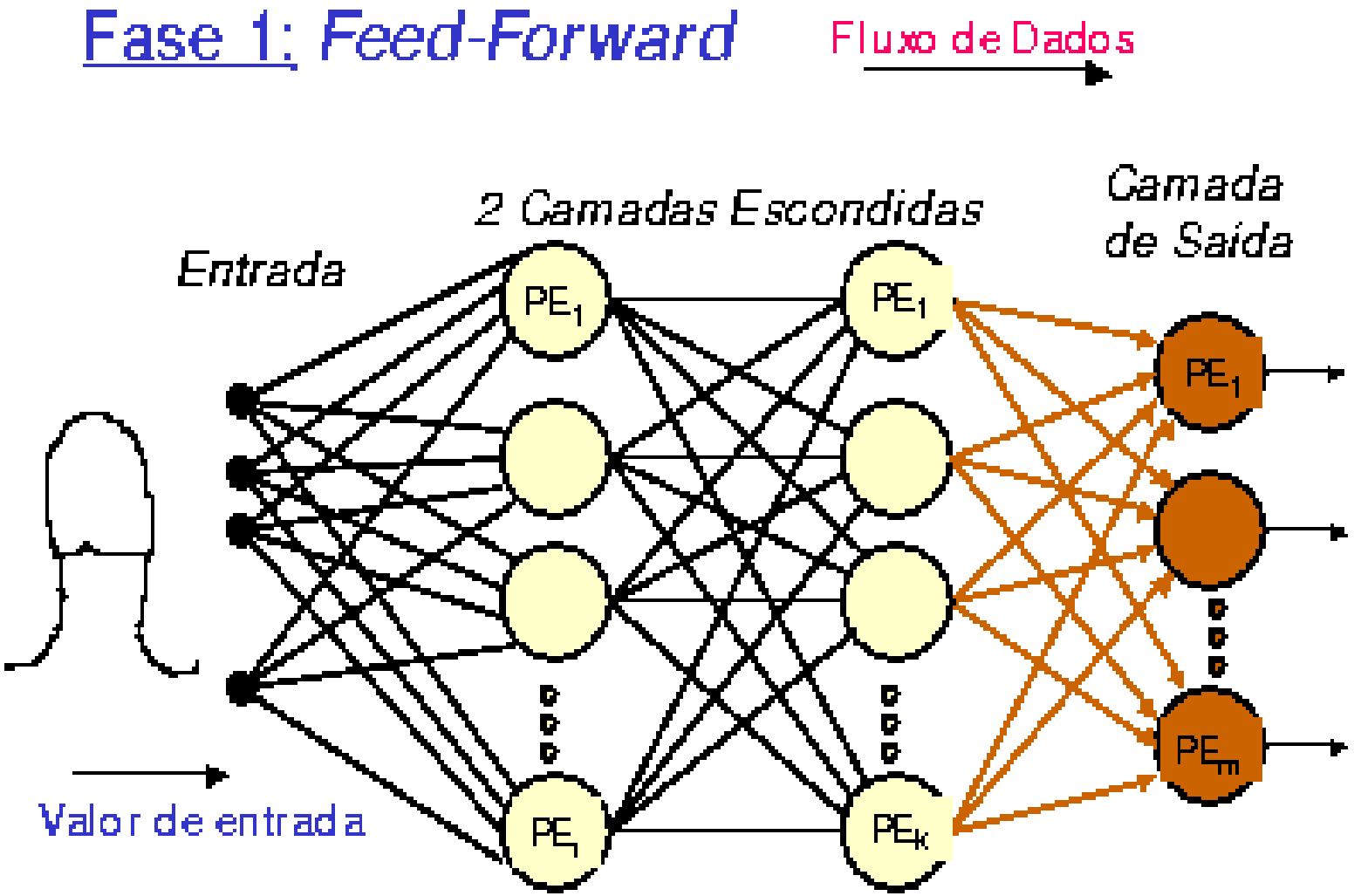
Fluxo de Dados



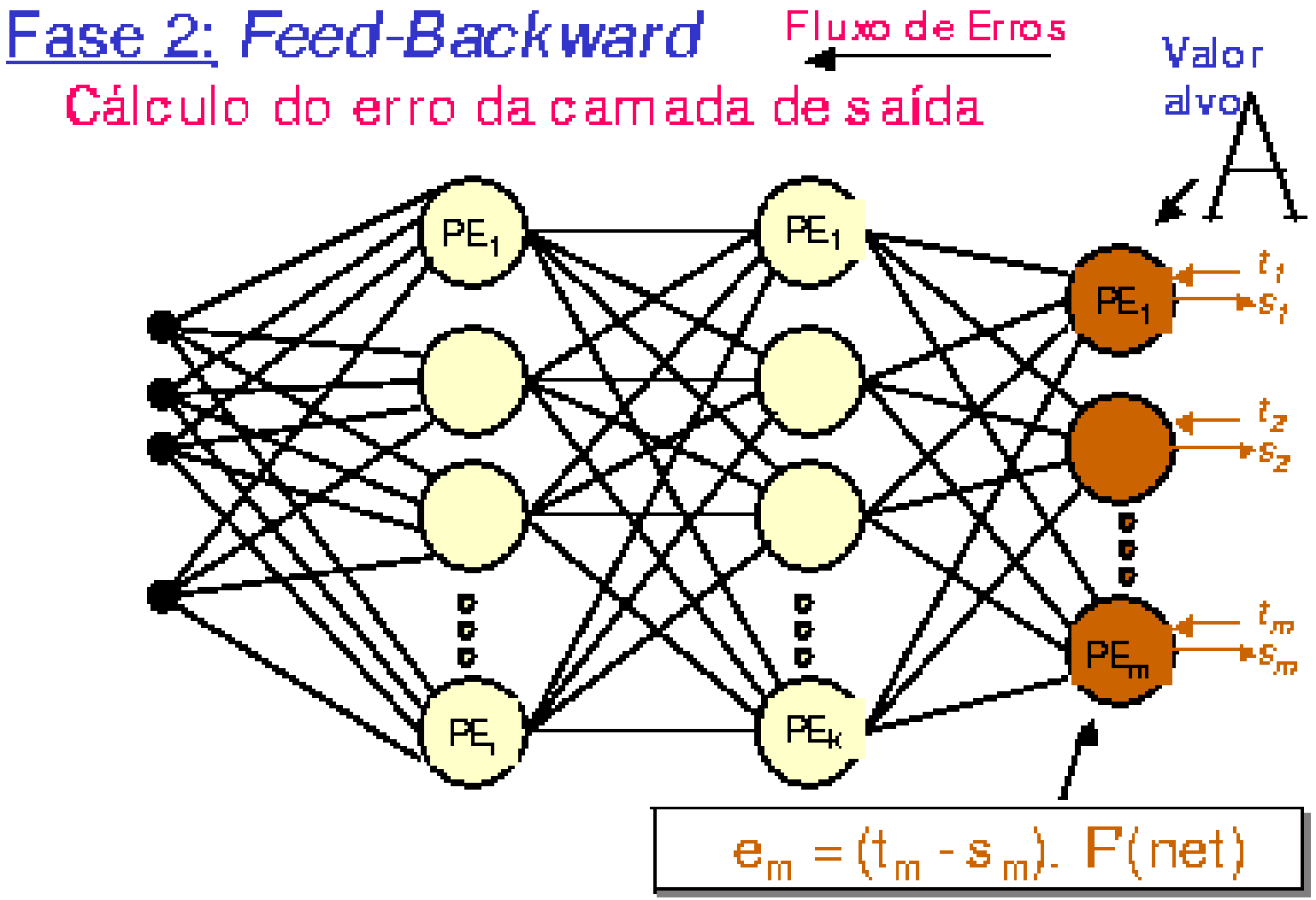
PROCESSO DE APRENDIZADO



PROCESSO DE APRENDIZADO



PROCESSO DE APRENDIZADO

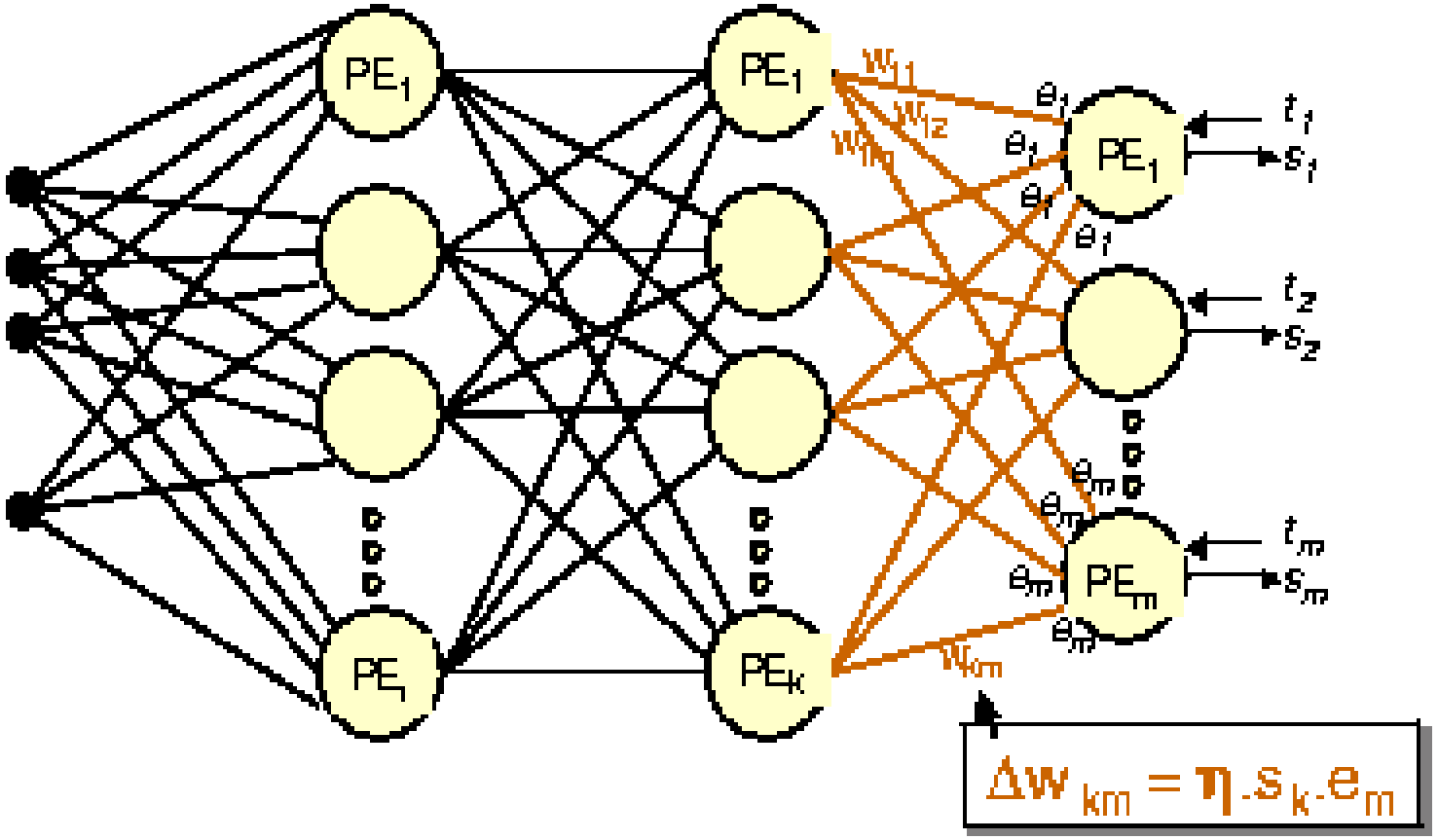


PROCESSO DE APRENDIZADO

Fase 2: *Feed-Backward*

Fluxo de Erros

Atualização dos pesos da camada de saída

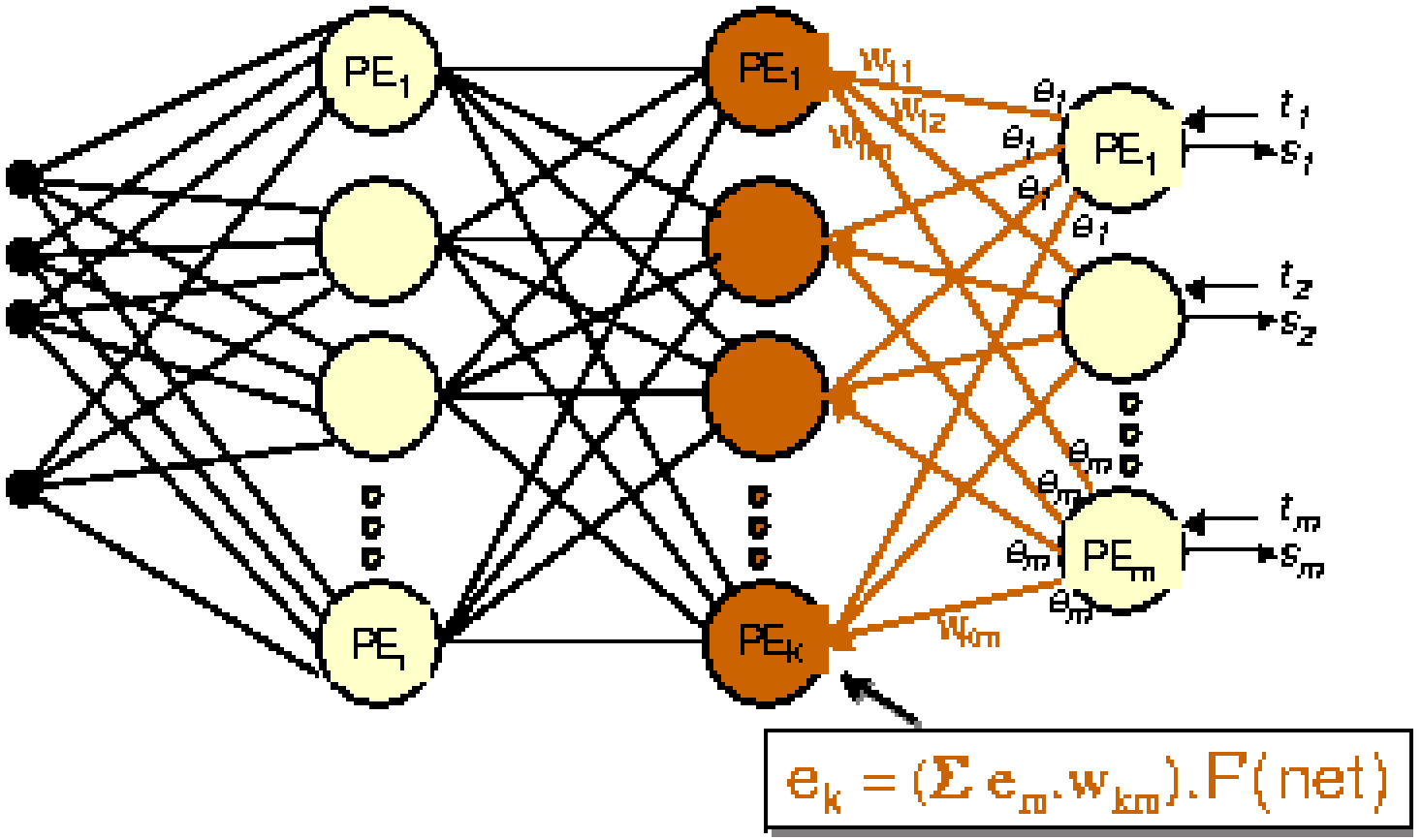


PROCESSO DE APRENDIZADO

Fase 2: *Feed-Backward*

Fluxo de Erros

Cálculo do erro da 2° camada escondida



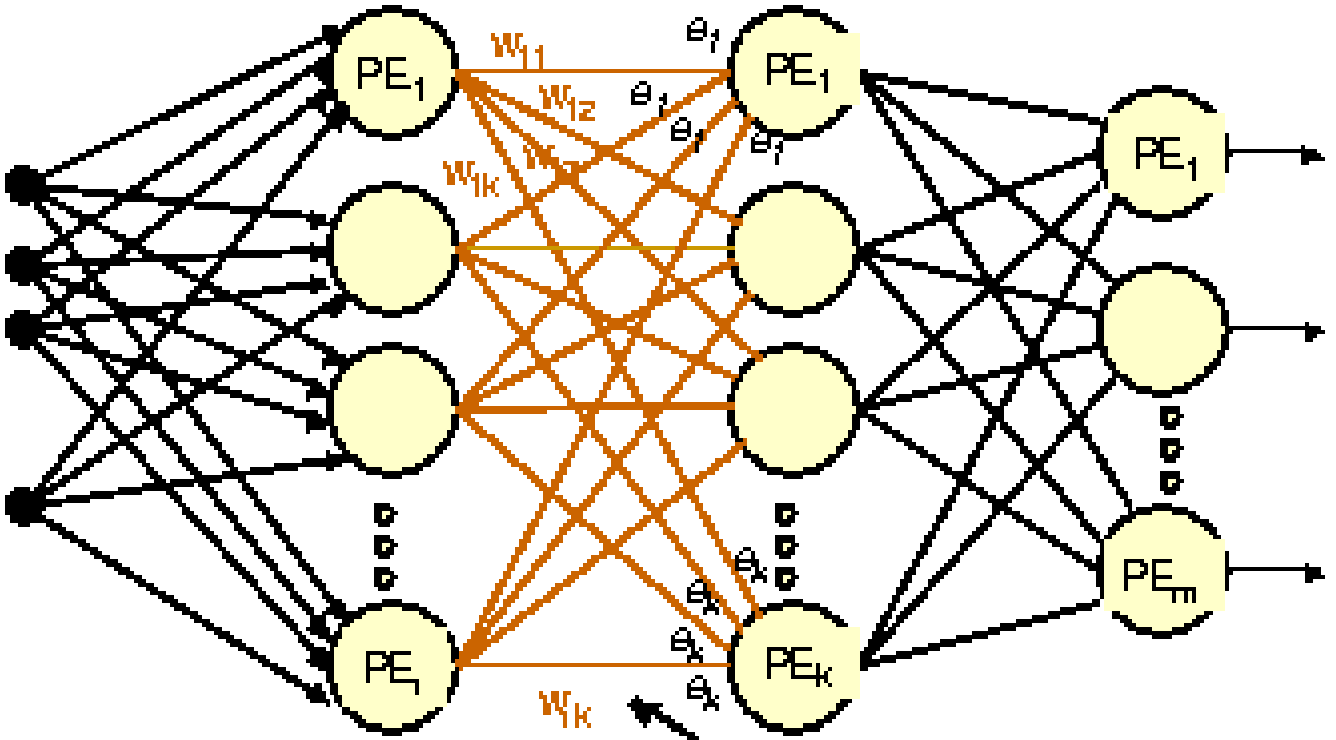
$$e_k = (\Sigma\, e_m . w_{km}) . F(net)$$

PROCESSO DE APRENDIZADO

Fase 2: *Feed-Backward*

Fluxo de Erros

Atualização dos pesos da 2ª camada escondida



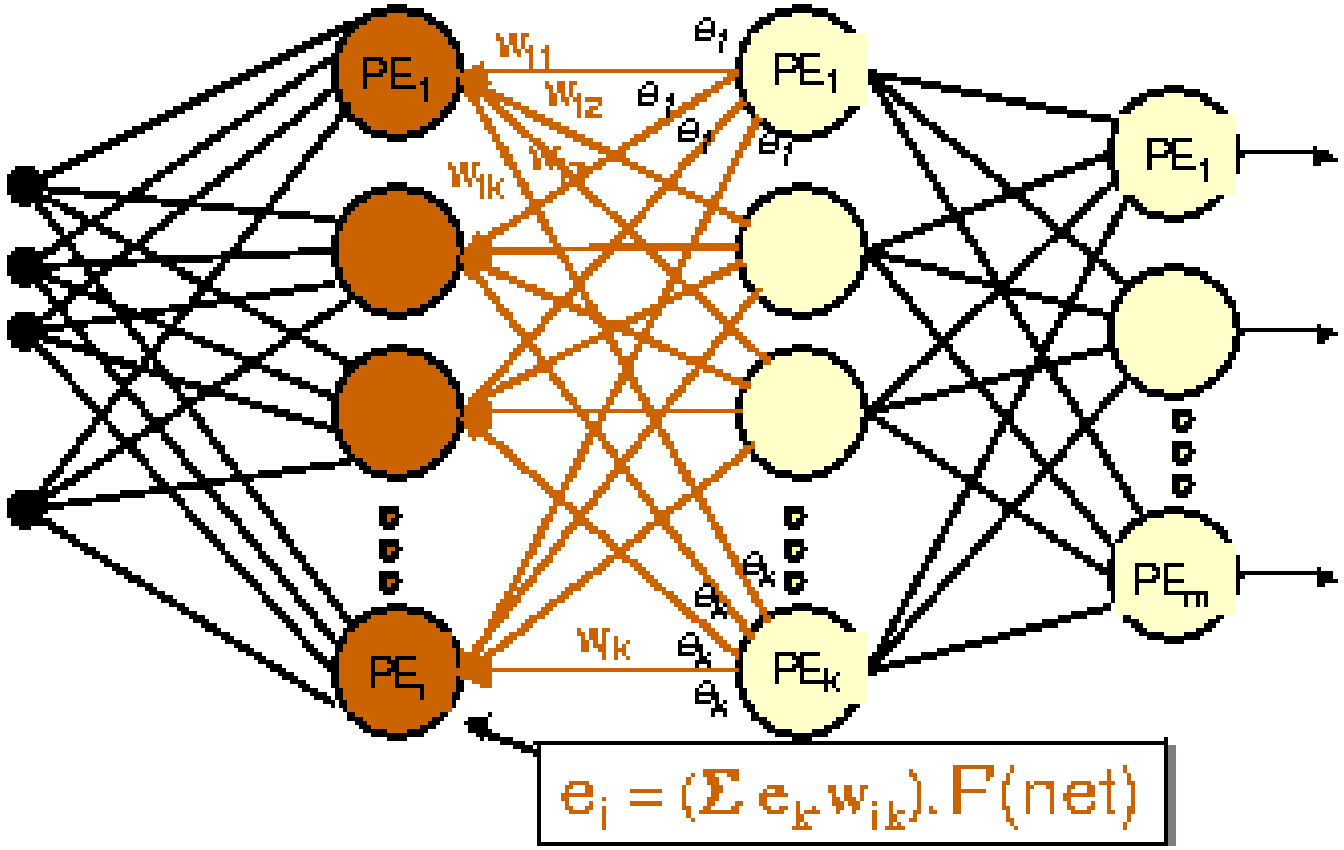
$$\Delta w_{ik} = \eta . s_i . e_k$$

PROCESSO DE APRENDIZADO

Fase 2: *Feed-Backward*

Fluxo de Erros

Cálculo do erro da 1° camada escondida

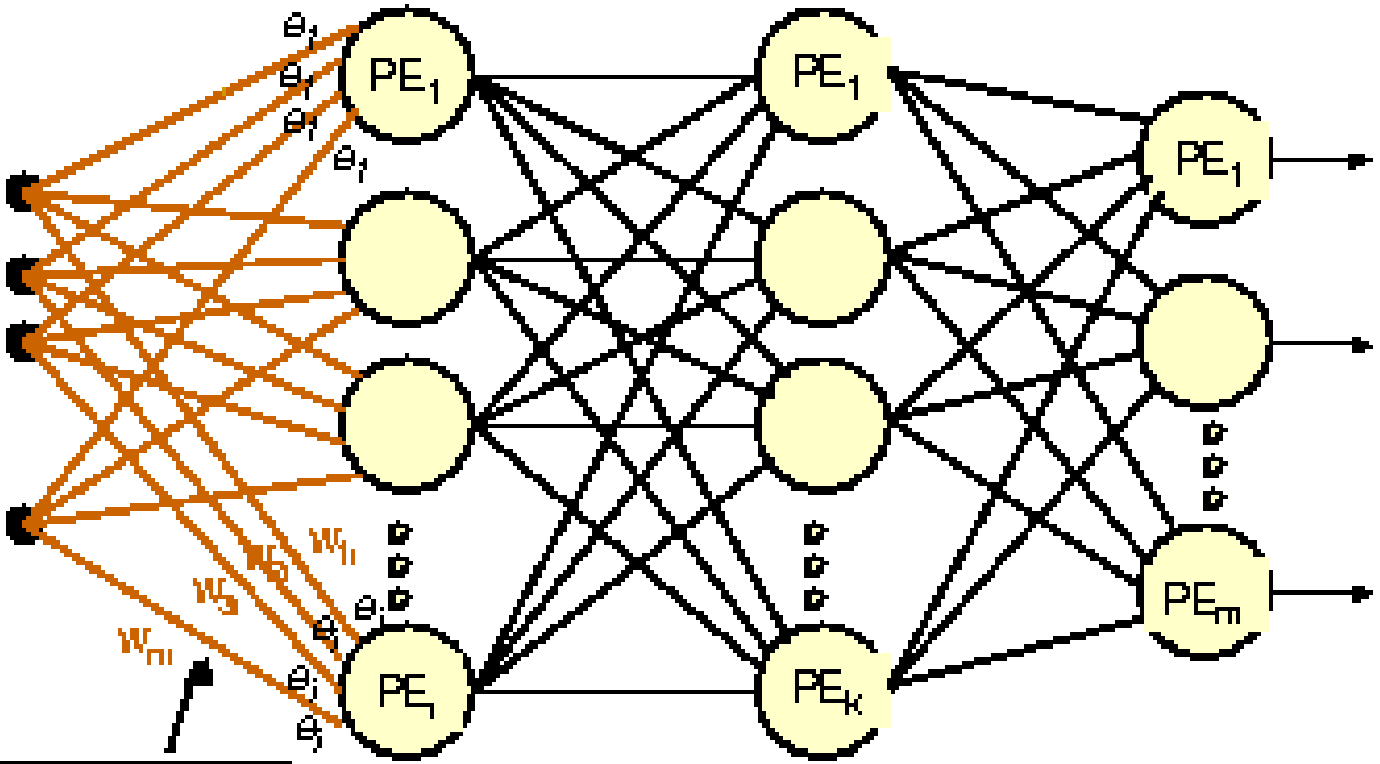


PROCESSO DE APRENDIZADO

Fase 2: *Feed-Backward*

Fluxo de Erros

Atualização dos pesos da 1ª camada escondida



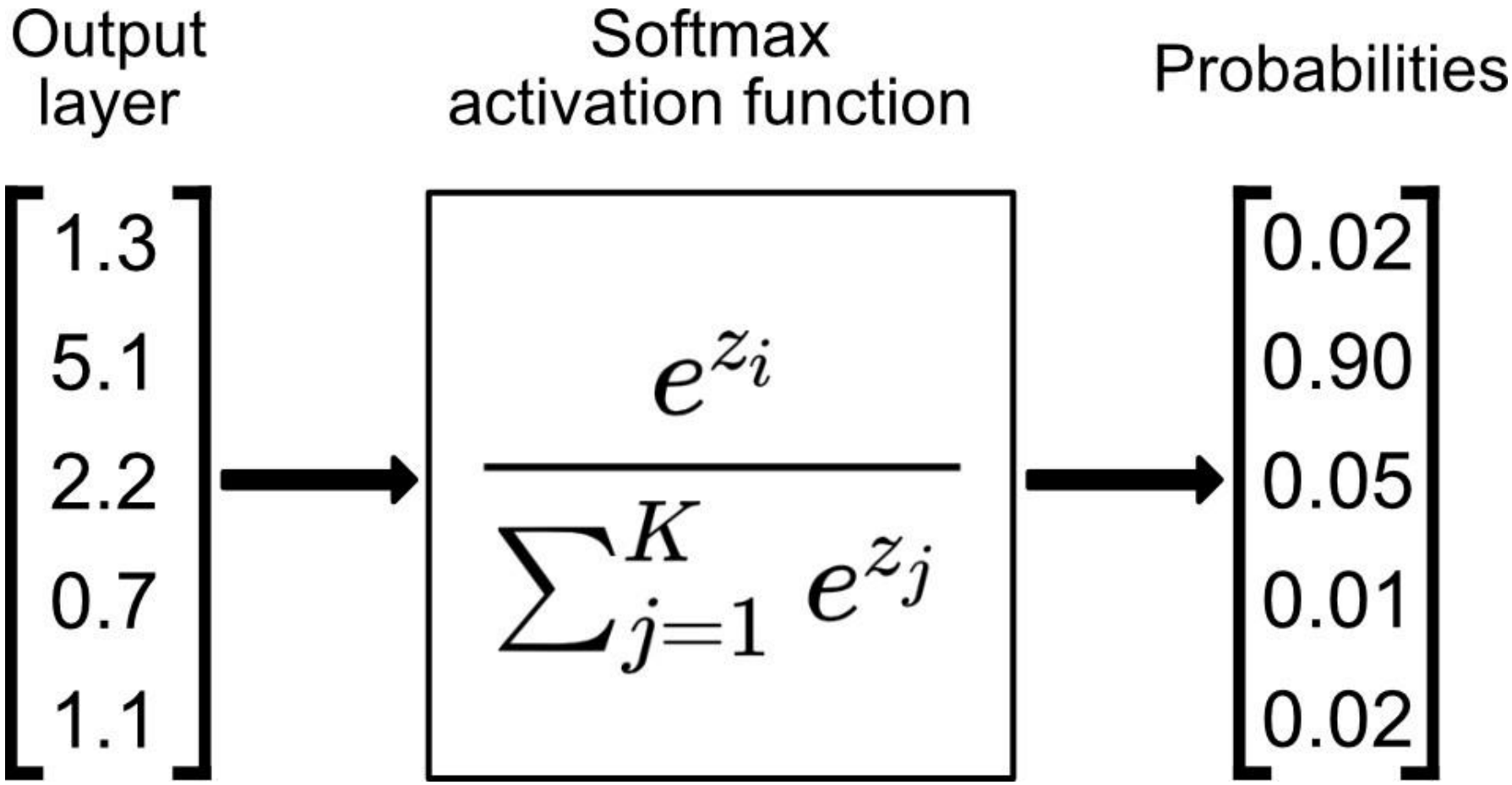
$$\Delta w_{ik} = \eta . s_i . e_k$$

MLP podem ser usadas:

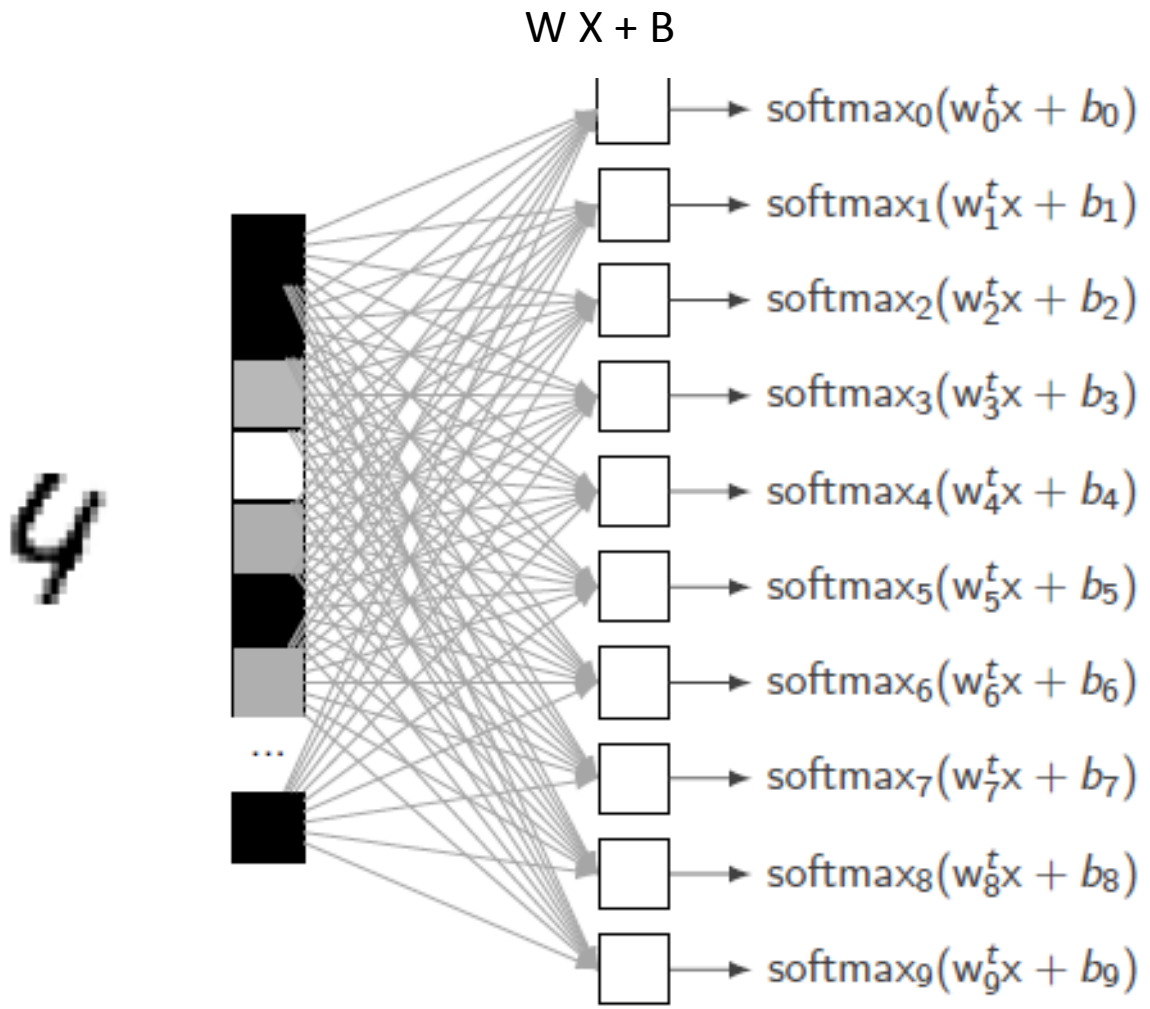
- **CLASSIFICAÇÃO**
- **REGRESSÃO**

FUNÇÃO SOFTMAX

Permite mais do que 2 classes de saída
Padroniza vetor de saída de forma a somar 1
Interpretação: valores são probabilidades
Saída: distr. de prob. das classes



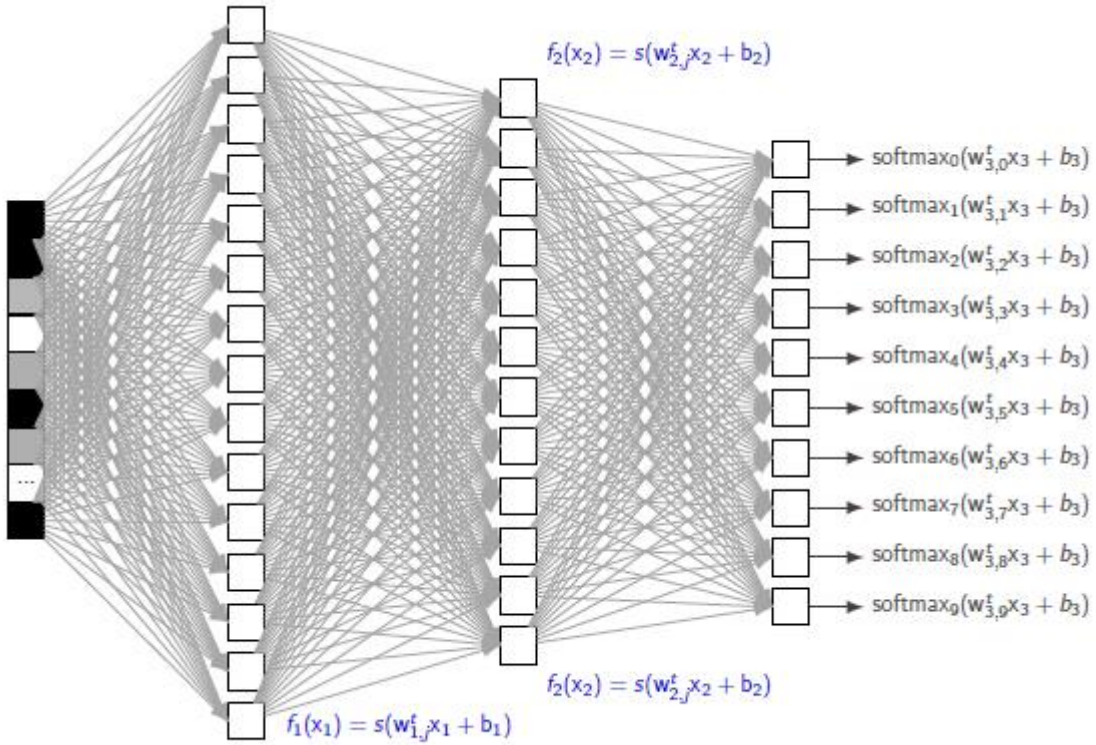
Questões importantes sobre MLP



1. Valores de entrada (atributos) são considerados independentes

2. Não são aproveitadas relações locais entre os dados

Questões importantes sobre MLP



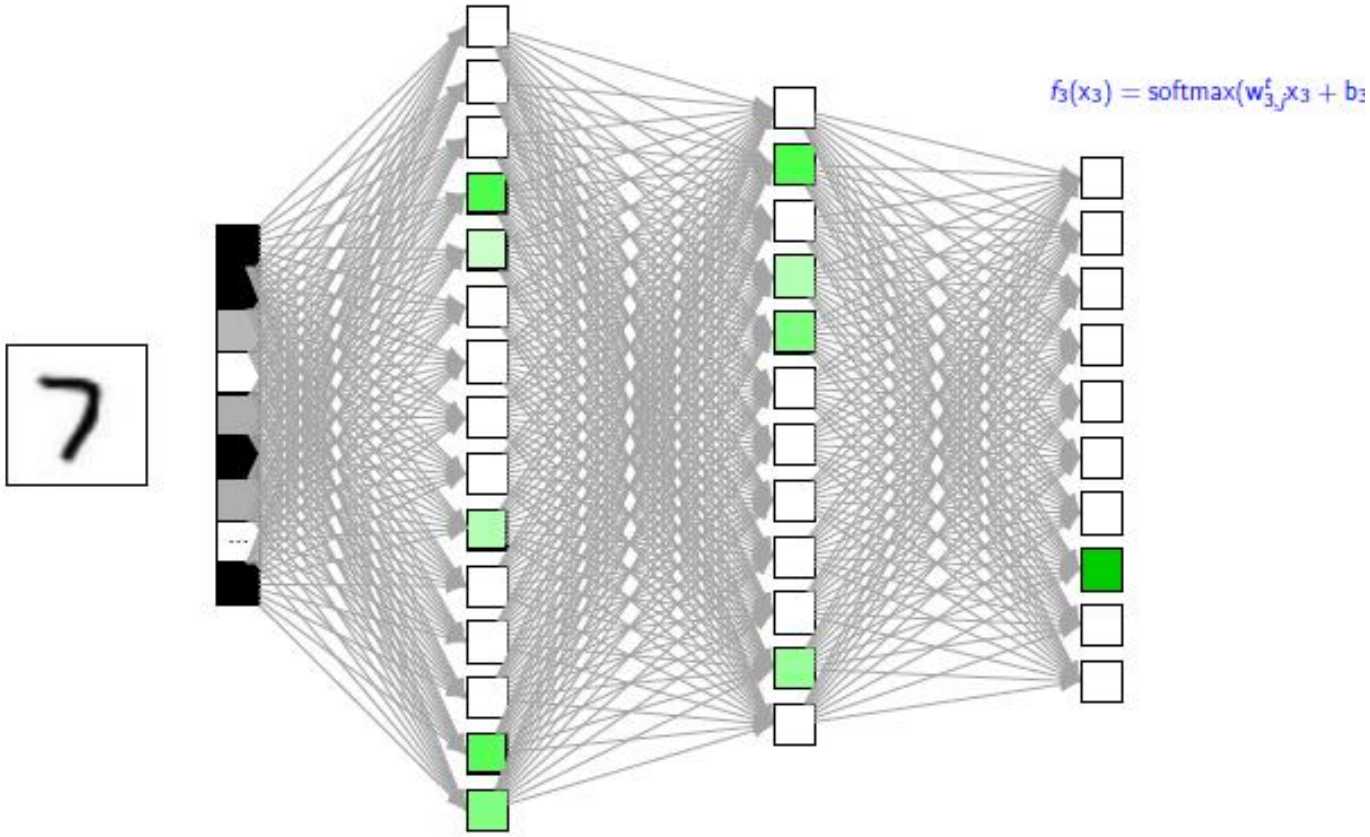
3. Número de parâmetros crescente impacta restrições de memória e processamento

=> Exemplo: entrada imagem de 28 x 28 = 784 pixels

Uma camada com 100 neurônios tem...

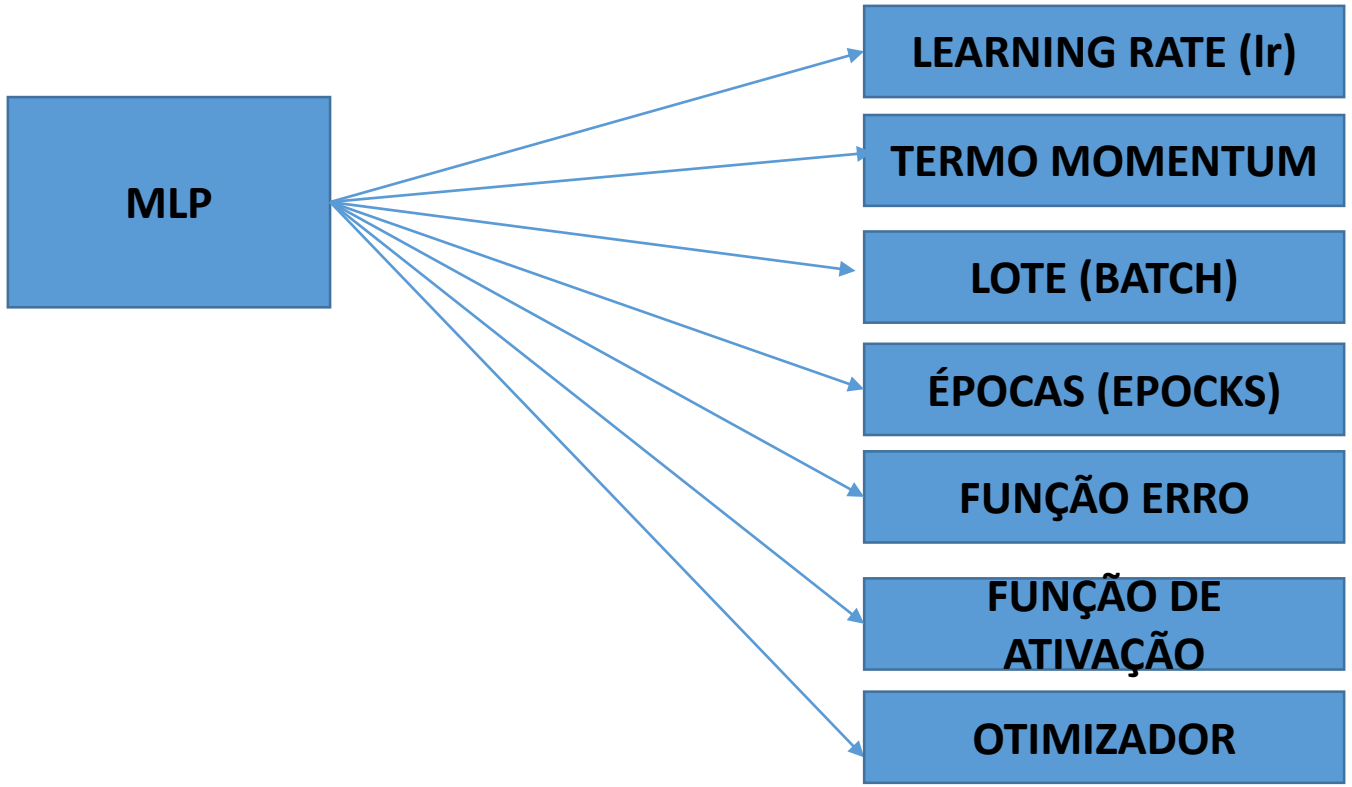
78400 + 100 = 78500 parâmetros a serem aprendidos e mantidos na memória durante o treinamento

MLP e RECONHECIMENTO



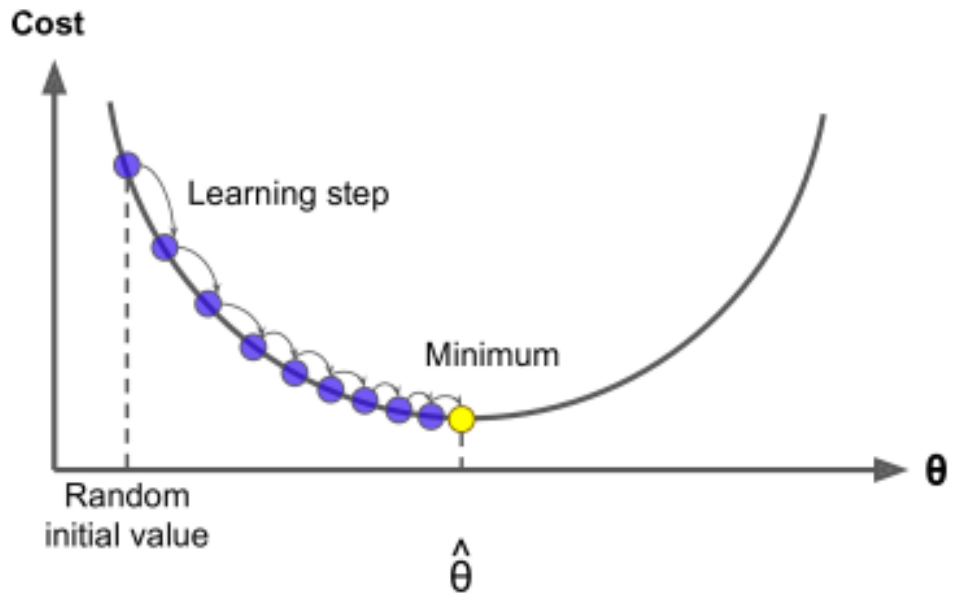
QUESTÕES DE IMPLEMENTAÇÃO DO MODELO MLP

IMPLEMENTAÇÃO DO MODELO MLP



IMPLEMENTAÇÃO DO MODELO MLP

LEARNING RATE – PARÂMETRO DE APRENDIZADO

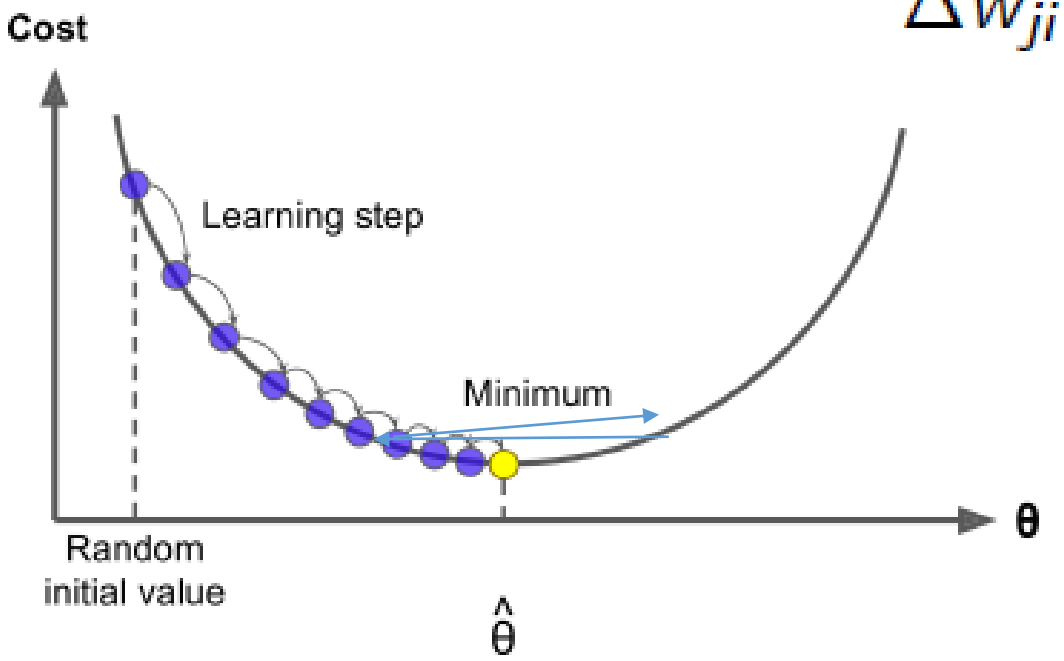


$$\Delta w_{jk} = -\eta \delta_j{}^p out_k{}^p$$

η - parâmetro de aprendizado

$0 < \eta < 1$

IMPLEMENTAÇÃO DO MODELO MLP
TERMO MOMENTUM



$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n) + \alpha \Delta w_{ji}(n-1)$$

α - parâmetro de aprendizado

$0 < \alpha < 1$

IMPLEMENTAÇÃO DO MODELO MLP

LOTE – SUBCONJUNTO A SER TREINADO

ÉPOCAS (early stopping) – TREINAMENTO COMPLETO DO CONJUNTO DE DADOS

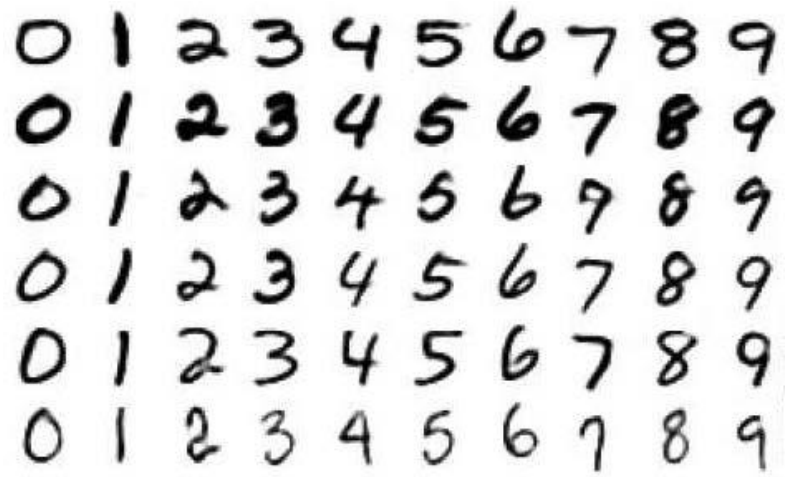
FUNÇÃO ERRO – MSE

FUNÇÃO DE ATIVAÇÃO – SIGMOID OU RELU

OTIMIZADOR - SGD

APRENDIZADO PROFUNDO x MLP

- EXEMPLO: 10 classes e bach size 32 e 784 características (pixels) por imagem



MNIST

TREINAMENTO: 50.000

TESTE: 10.000

BATCH = 32

$$\begin{bmatrix} x_{0,0} & x_{0,1} & x_{0,2} & \cdots & x_{0,783} \\ x_{1,0} & x_{0,1} & x_{1,2} & \cdots & x_{0,783} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{31,0} & x_{31,1} & x_{31,2} & \cdots & x_{31,783} \end{bmatrix} \begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,9} \\ w_{1,0} & w_{1,1} & \cdots & w_{1,9} \\ w_{2,0} & w_{2,1} & \cdots & w_{2,9} \\ \vdots & \vdots & \ddots & \vdots \\ w_{783,0} & w_{783,1} & \cdots & w_{783,9} \end{bmatrix} + \begin{bmatrix} b_0 & b_1 & b_2 & \ldots & b_9 \end{bmatrix}$$

$$\mathsf{Y} = \mathrm{softmax}(\mathsf{X} \cdot \mathsf{W} + \mathsf{b})$$

$$\mathsf{Y} = \begin{bmatrix} y_{0,0} & y_{0,1} & y_{0,2} & \cdots & y_{0,9} \\ y_{1,0} & y_{1,1} & y_{1,2} & \cdots & y_{1,9} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ y_{31,0} & y_{31,1} & y_{31,2} & \cdots & y_{31,9} \end{bmatrix}$$

EXEMPLO 1

MLP
USO DO KERAS

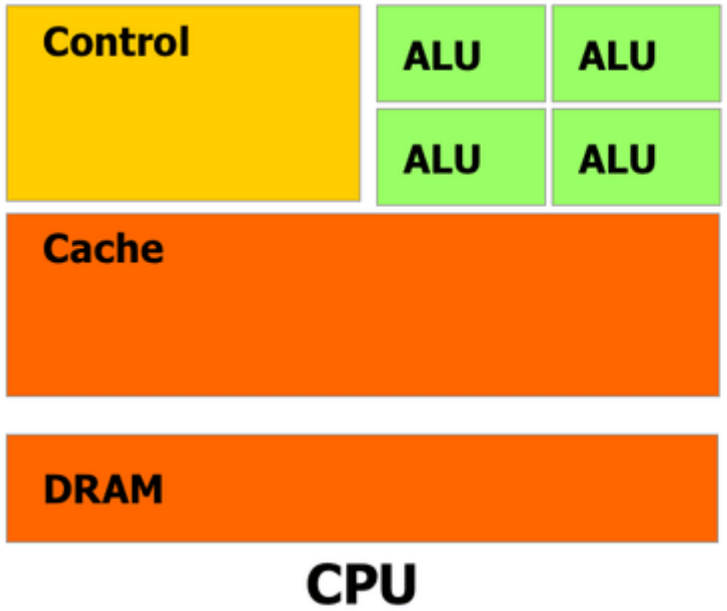
REDES NEURAIS PROFUNDAS

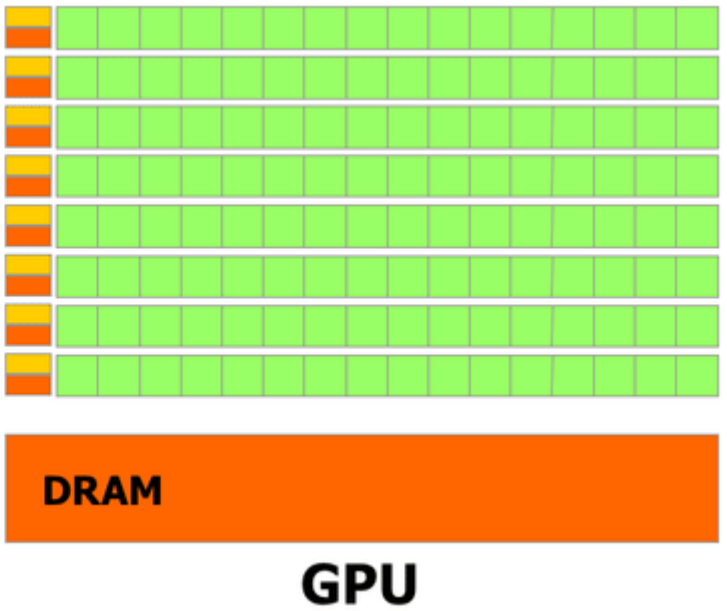
1º. MOTIVO

- IMAGENET – aproximadamente 1,4 milhões de imagens (2009)
- 1000 classes no total – categorizar essas imagens

2º. MOTIVO

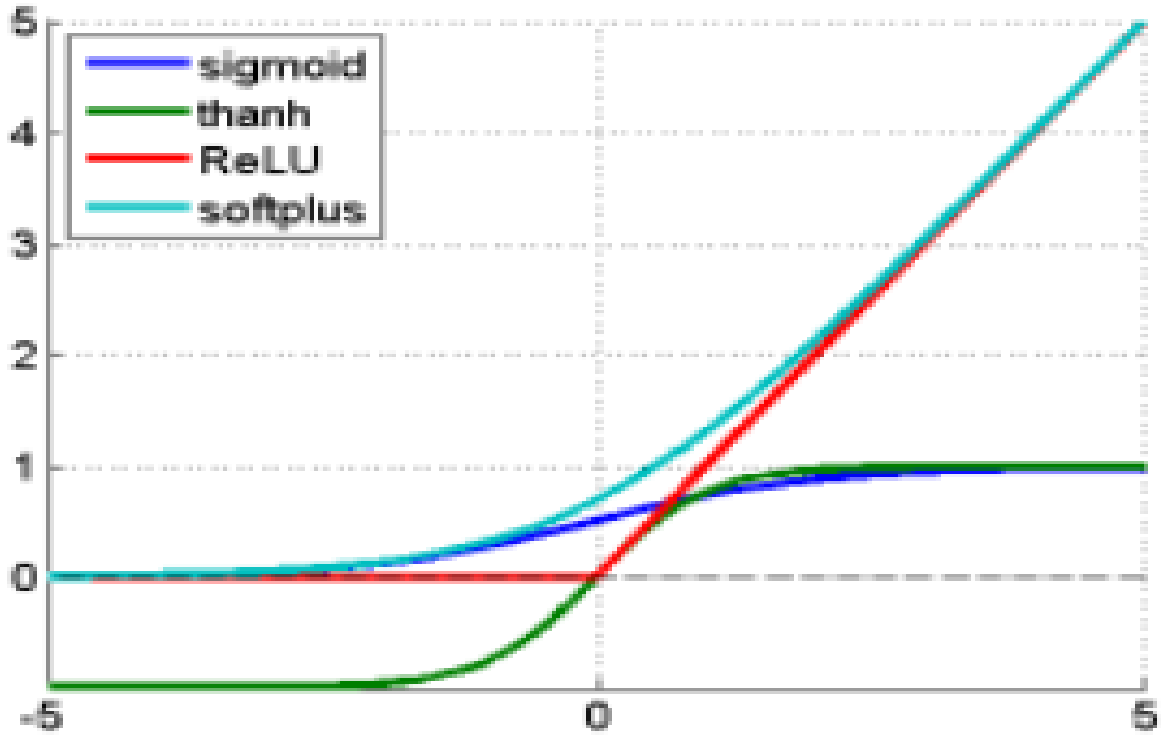
- CPU X GPU – Alta densidade de processamento
  - controle simples
  - cache pequeno
  - alta tolerância a latência





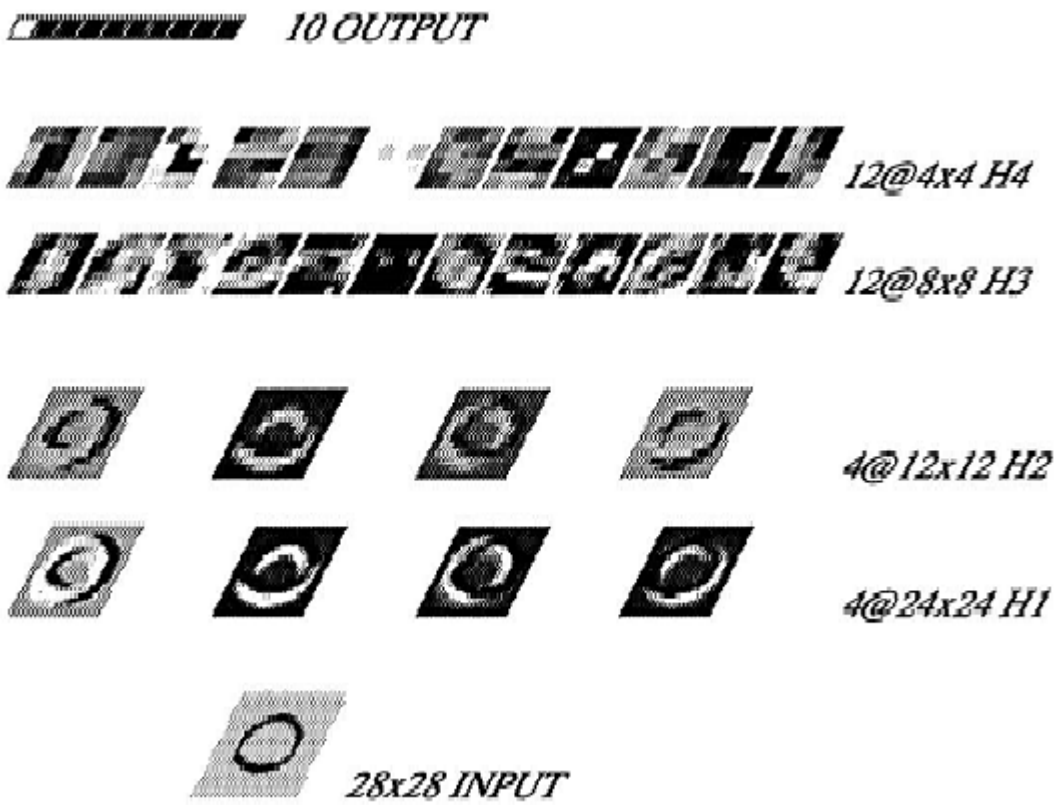
3º. MOTIVO

Não-linearidade: Funções de Ativação



4º. MOTIVO

- REVISITAÇÃO DO MODELO LE CUN



SUB-AMOSTRAGEM

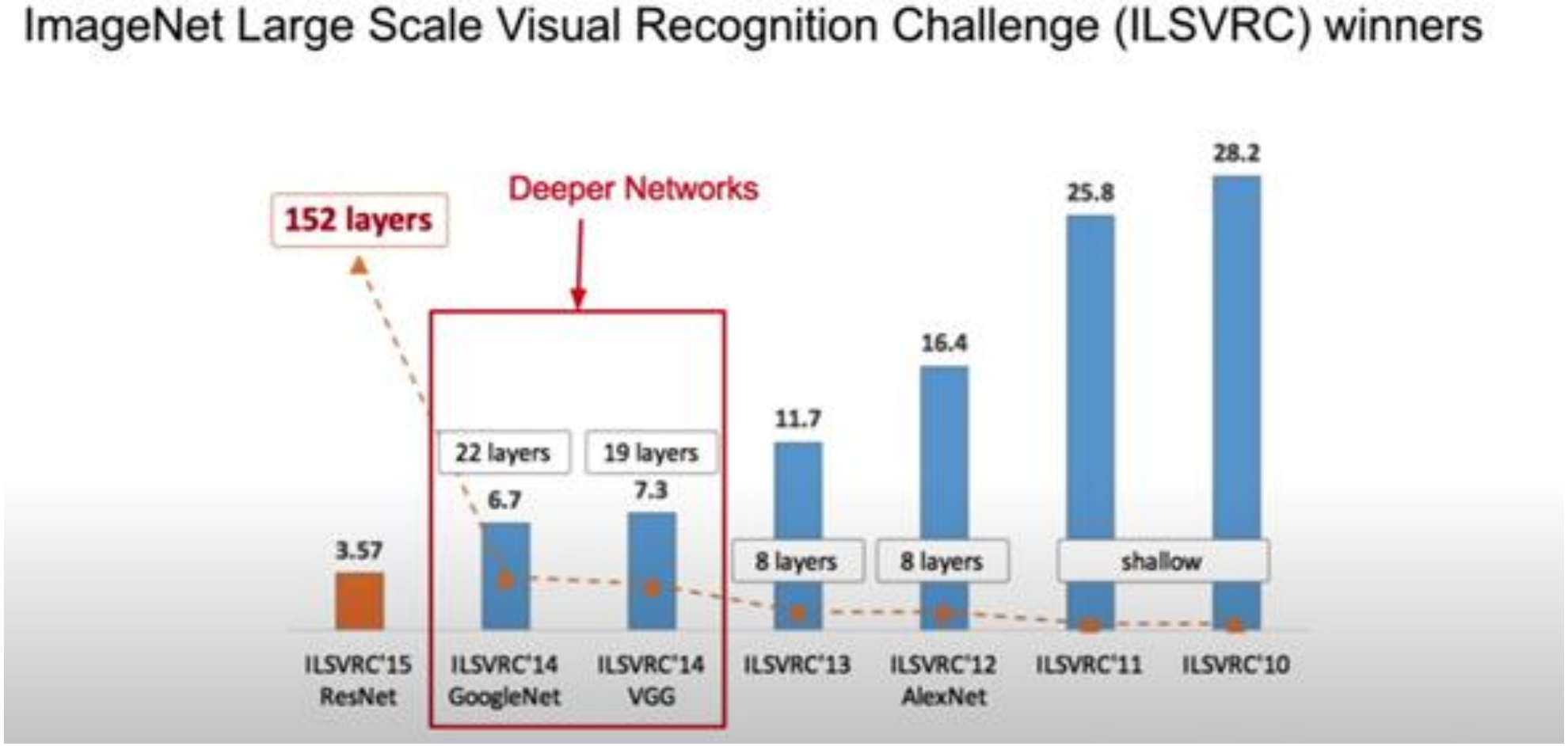
CONVOLUÇÃO

SUB-AMOSTRAGEM

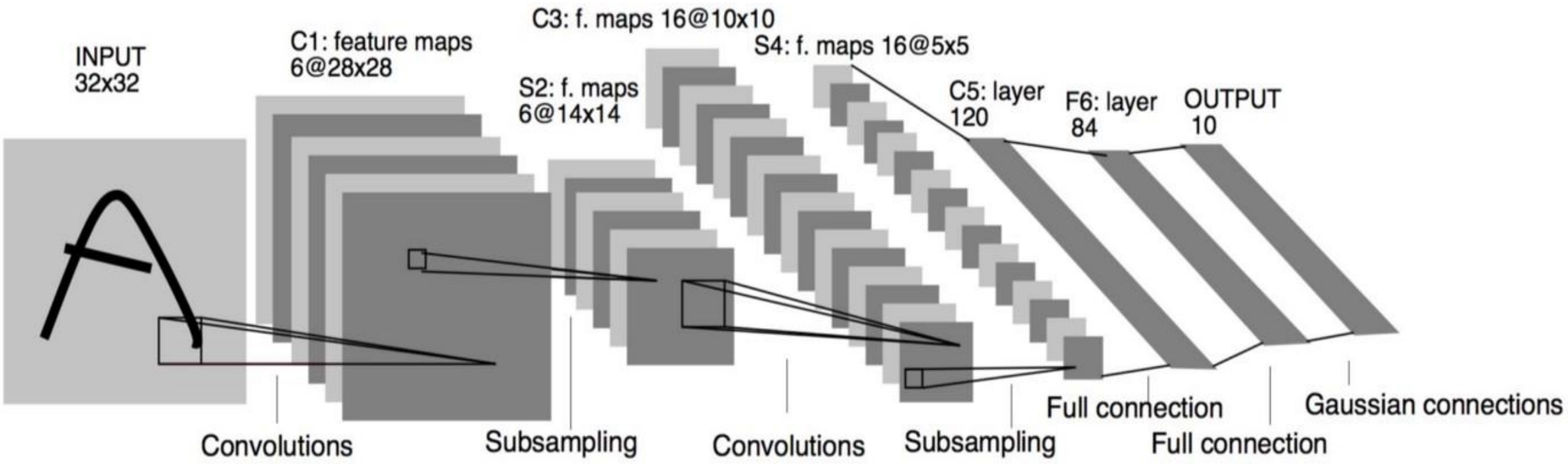
CONVOLUÇÃO

PRINCIPAIS MODELOS DE REDES PROFUNDAS

ALEX NET (9) INCEPTION (22) VGGNET (16/19) RESNET(34-1000)

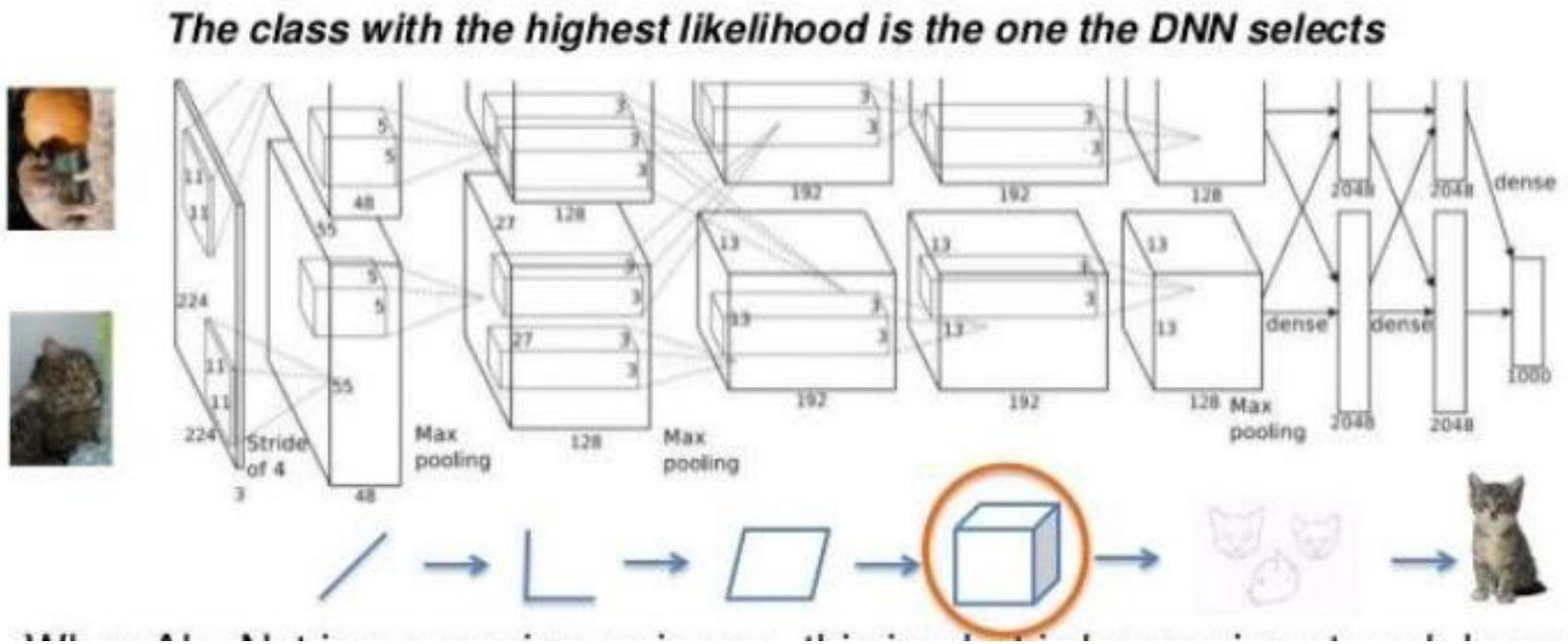


REDES NEURAIS CONVOLUCIONAIS



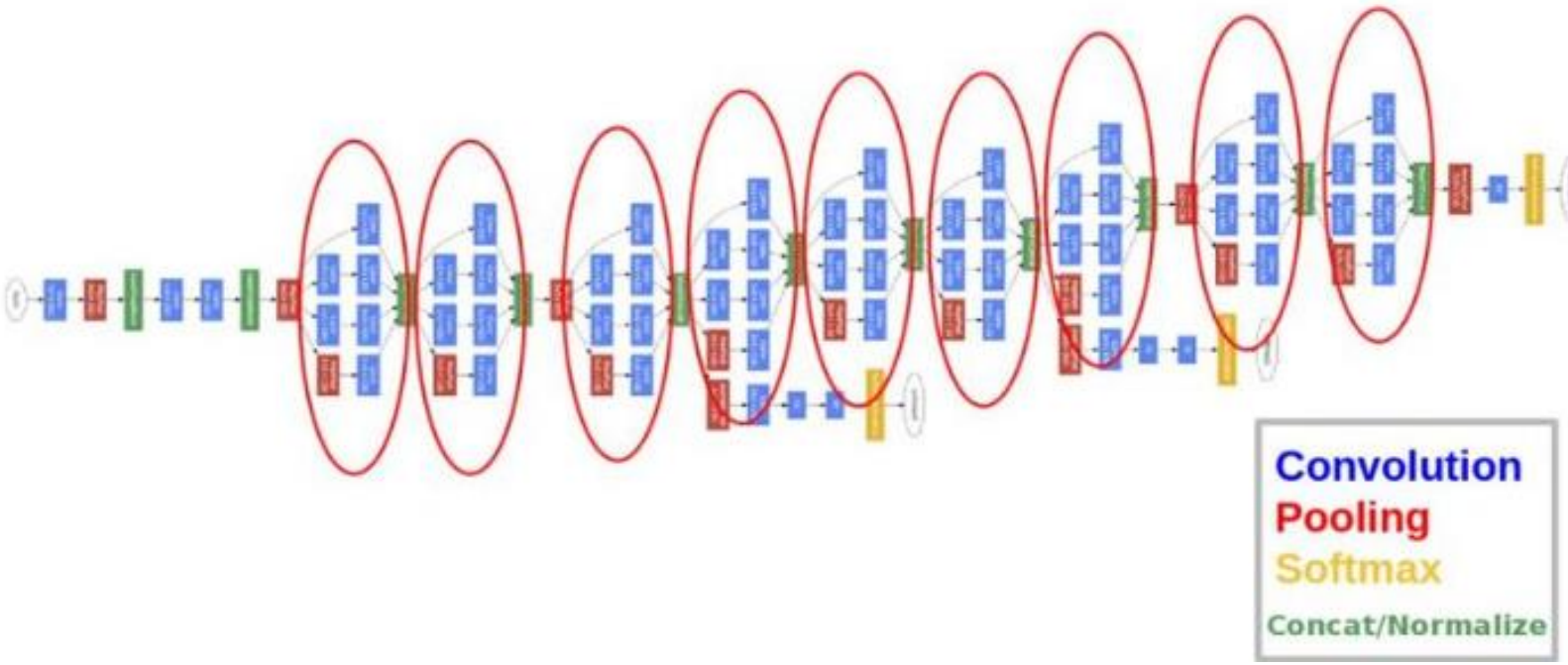
ALEX NET (2012)

9 CAM.



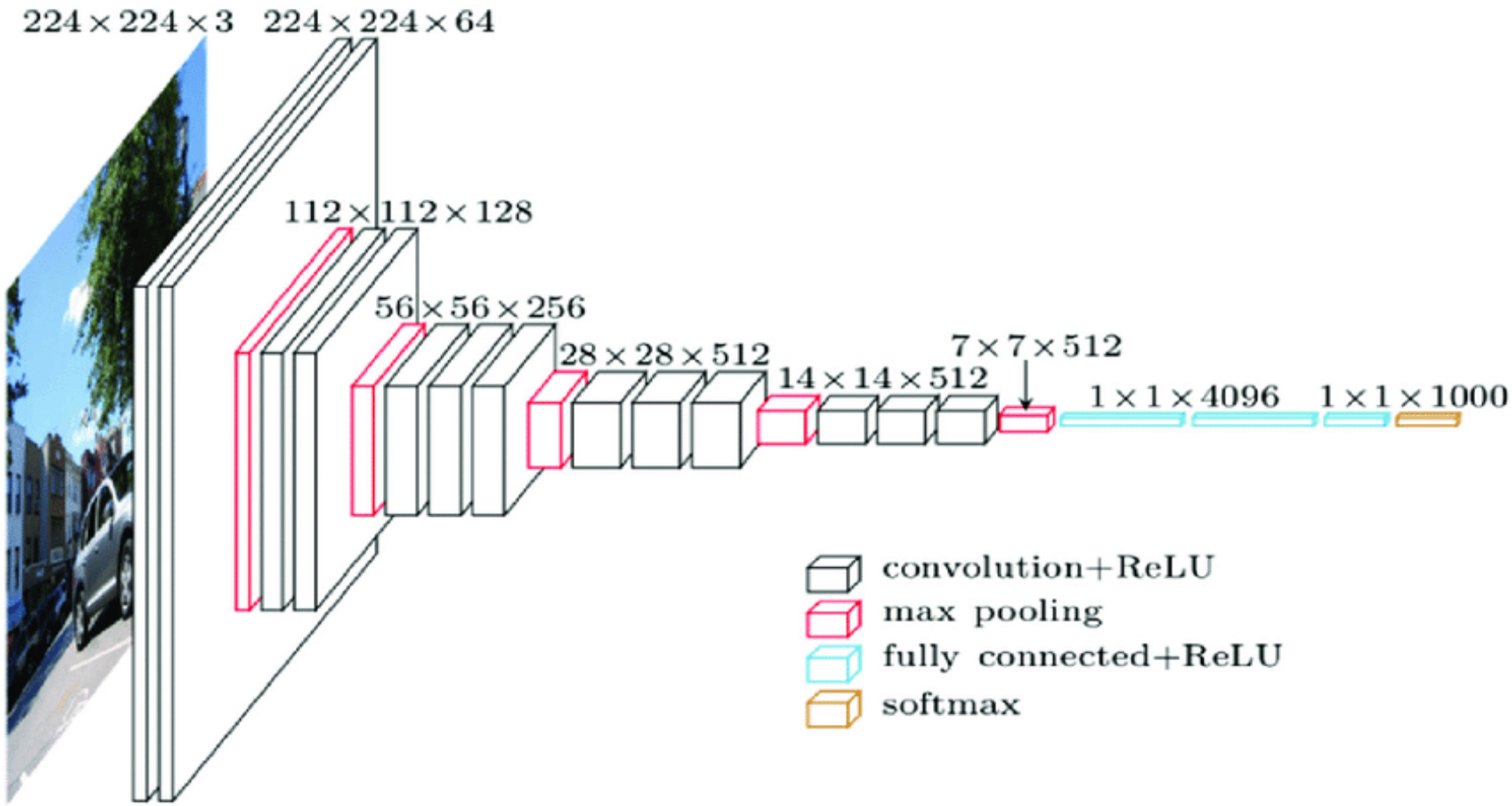
INCEPTION / GOOGLENET (2014)

22 CAM.



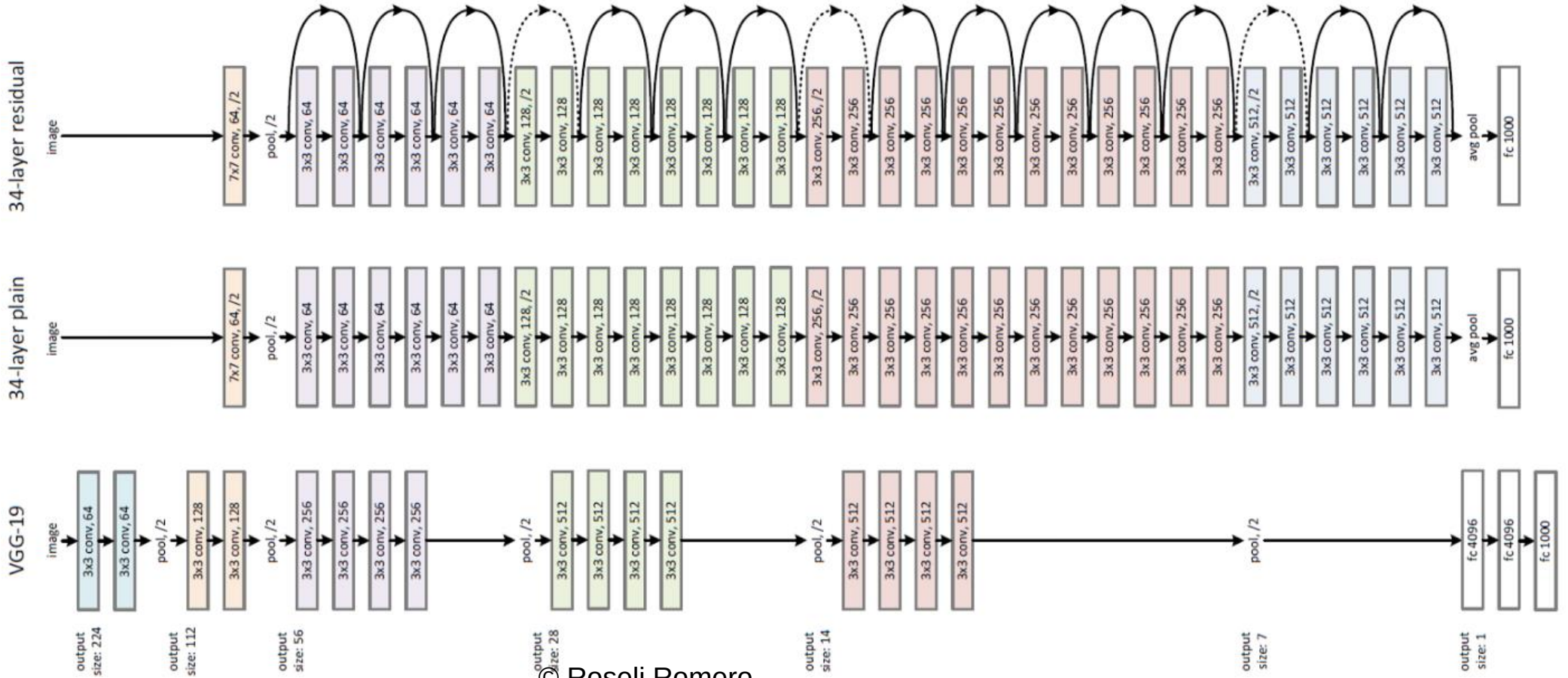
VGGNET (2014)

16 CAM.



RESNET (2015)

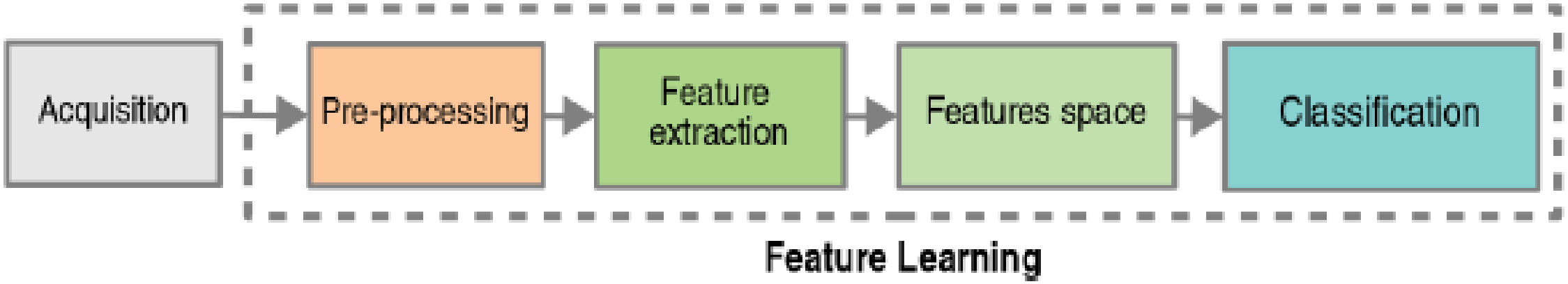
34 A 1000 CAM.



DEEP LEARNING



$$\mathbf{x} \in \mathbb{R}^m$$



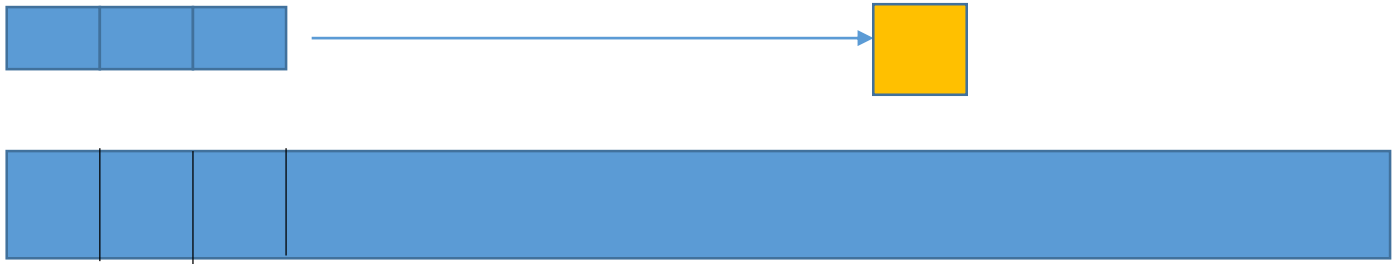
Novo Pipeline para Tratamento de Imagens

NOVA TERMINOLOGIA

- CAMADA CONVOLUCIONAL
- MAPAS DE ATIVAÇÃO (ACTIVATION TEXTURE MAPS)
- CAMADA DENSA (DENSE OR FULLY CONNECTED)
- SUBAMOSTRAGEM (POOLING)

CONVOLUÇÃO

- OPERADOR QUE VISA REALIZAR UMA COMBINAÇÃO LINEAR DE VALORES LOCAIS DA ENTRADA
- CENTRADO EM UMA POSIÇÃO, isto é, (x,y), gera como saída um único valor de saída
- CONVOLUÇÃO 1D



APRENDIZADO PROFUNDO

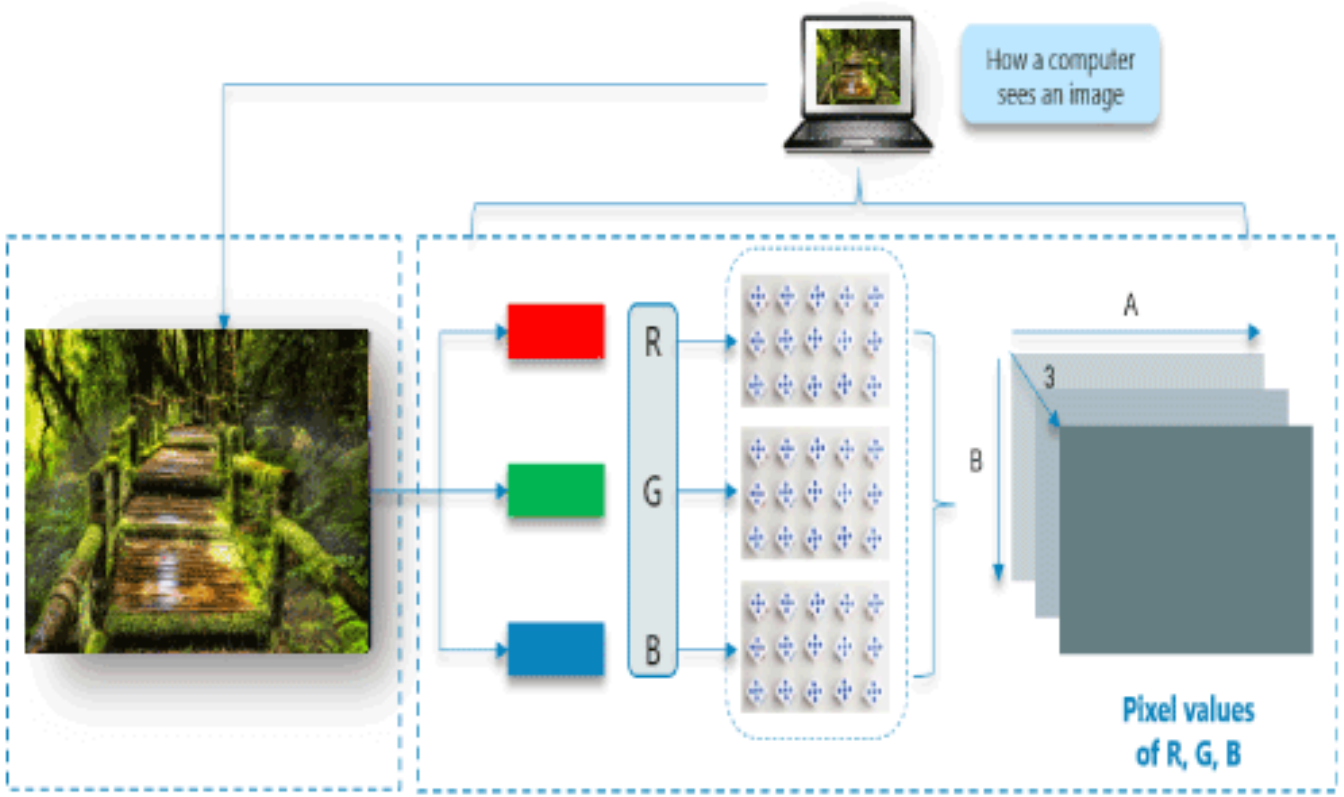
3	**0**	**1**	**2**	**7**	4
1	**5**	**8**	**9**	**3**	**1**
2	**7**	**2**	**5**	**1**	**3**
0	**1**	**3**	**1**	**7**	**8**
4	**2**	**1**	**6**	**2**	**8**
2	**4**	**5**	**2**	**3**	**9**

*

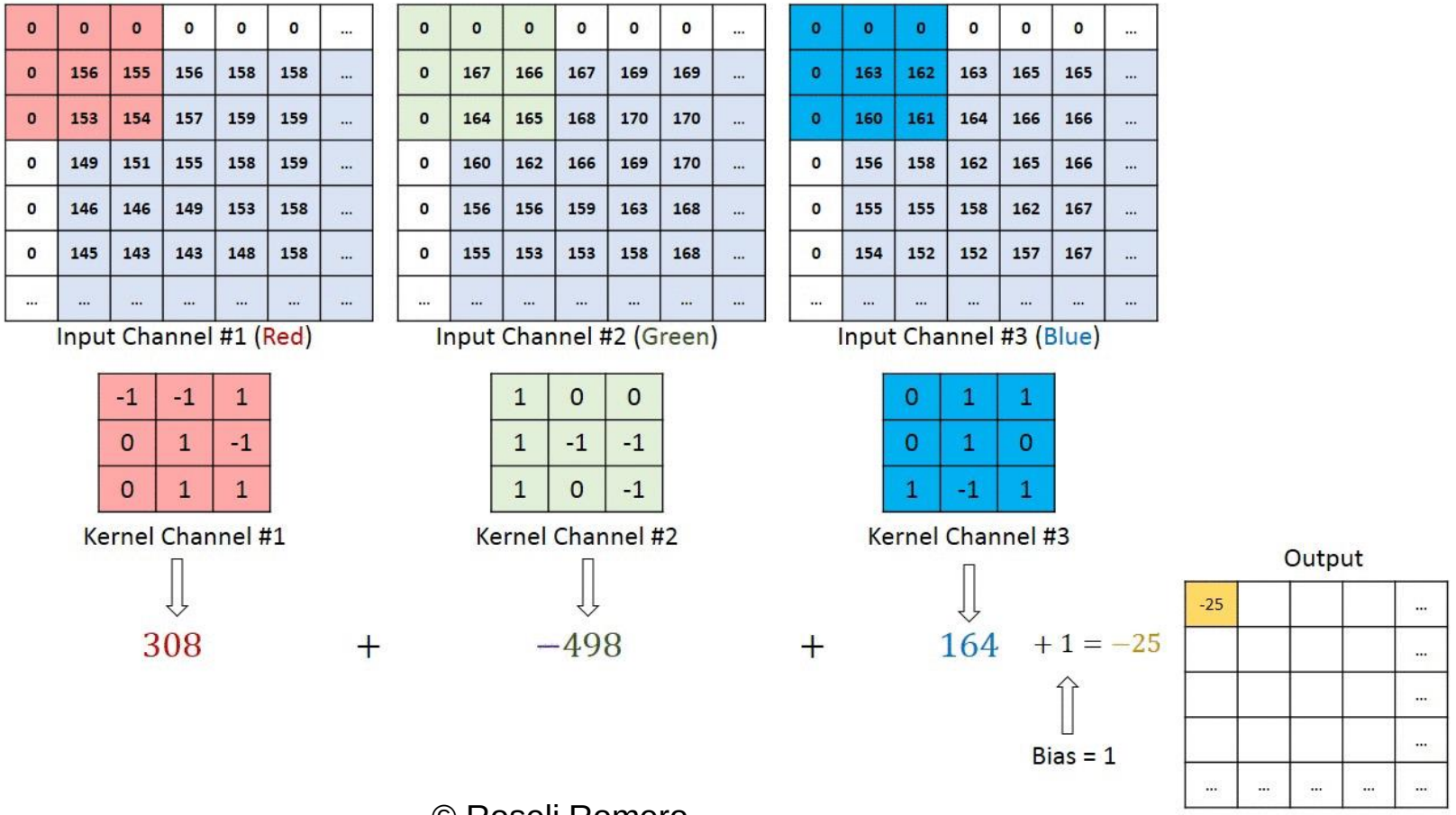
1	0	-1
1	0	-1
1	0	-1

-5	-4	0	8

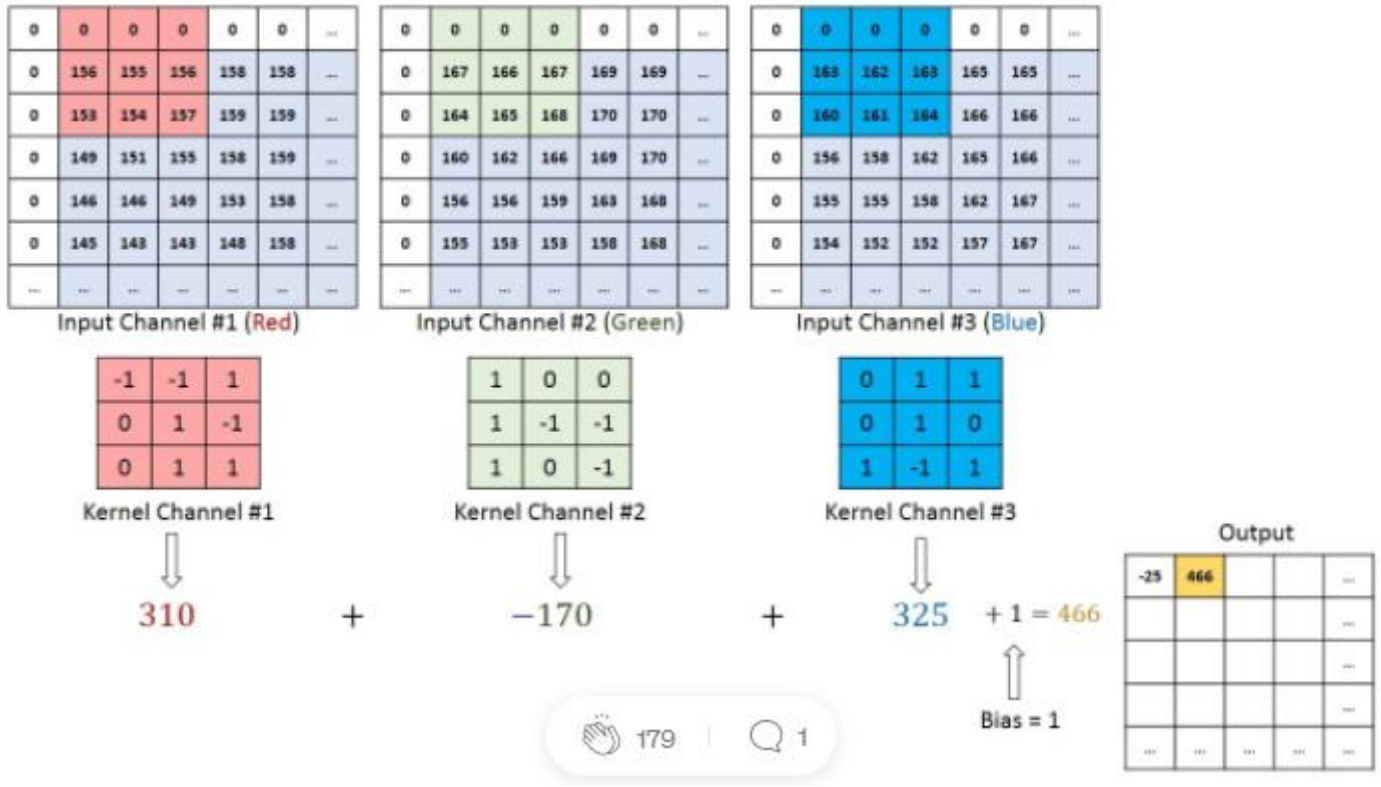
APRENDIZADO PROFUNDO



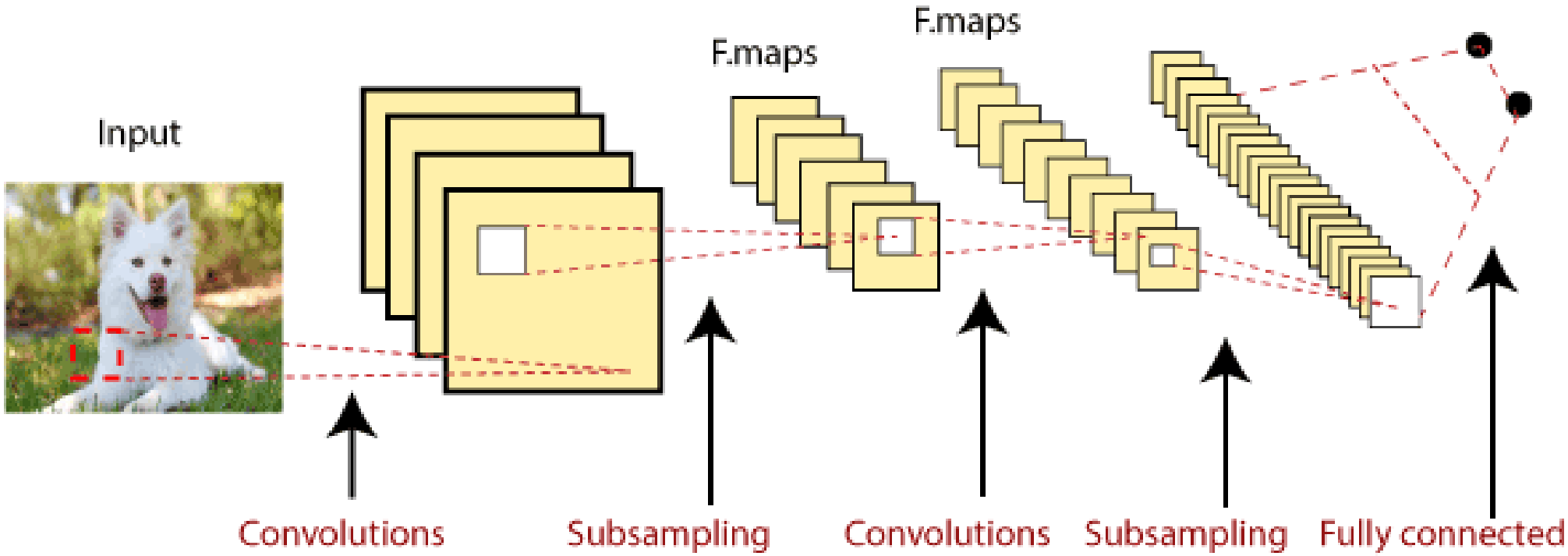
Convolução 3D



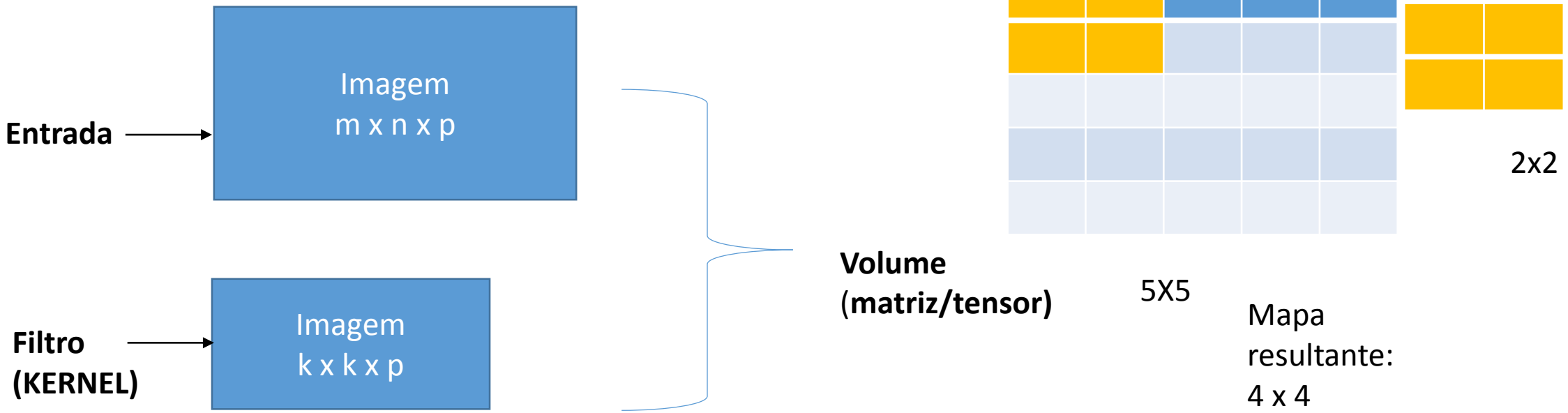
CONVOLUÇÃO 3D



APRENDIZADO PROFUNDO



TAMANHO DOS FEATURES MAPS



STRIDE → PASSO

PROFUNDIDADE DOS FEATURES MAPS

(5 – 2)/1 + 1 = 4

Stride 2: (5-2)/2 + 1 => 2x2 Mapa resultante:

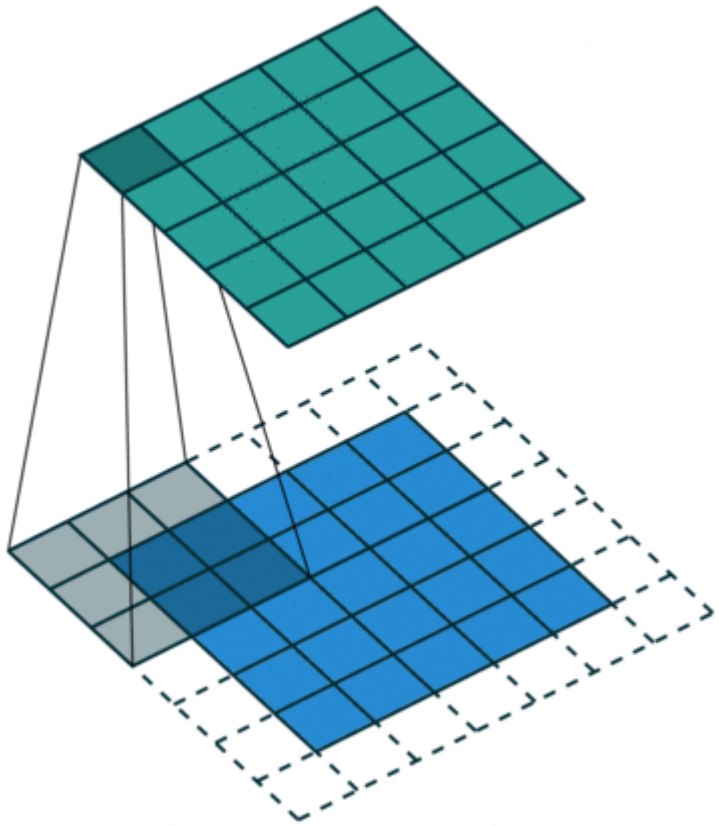
NÚMERO DE PARÂMETROS

$$[(K \times K \times P) + 1] \times D$$

Entrada: 32 x 32 x 3 e 3 Feature Maps

CONV1: K=5 D = 8 #param_1: [(5x5x3) + 1] x 8 = 608
CONV2: K=3 D = 16 #param_2: [(3x3x8) + 1] x 16 = 1168
CONV3: K=1 D = 32 #param_3: [(1x1x16) + 1] x 32 = 544

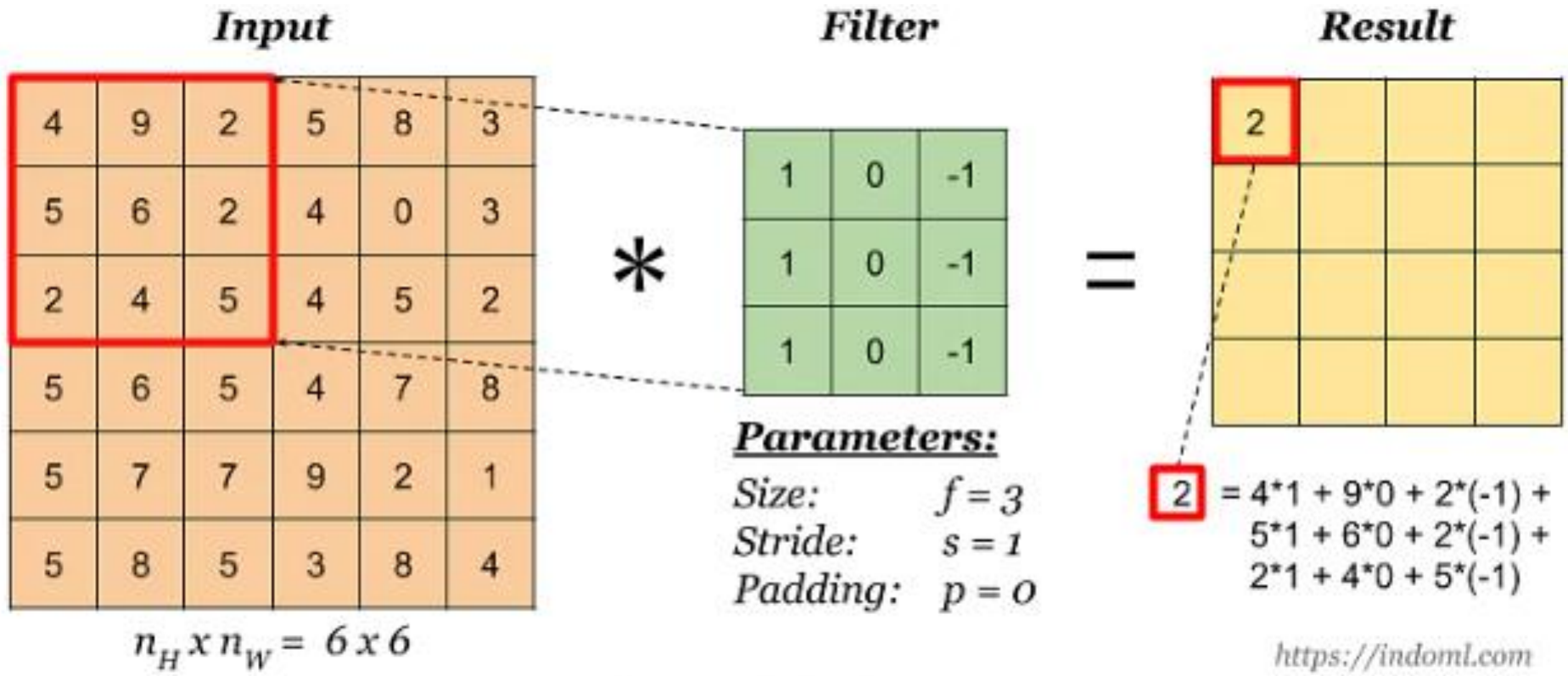
PADDING



Zero-padding: para compensar a impossibilidade de computar todos os valores

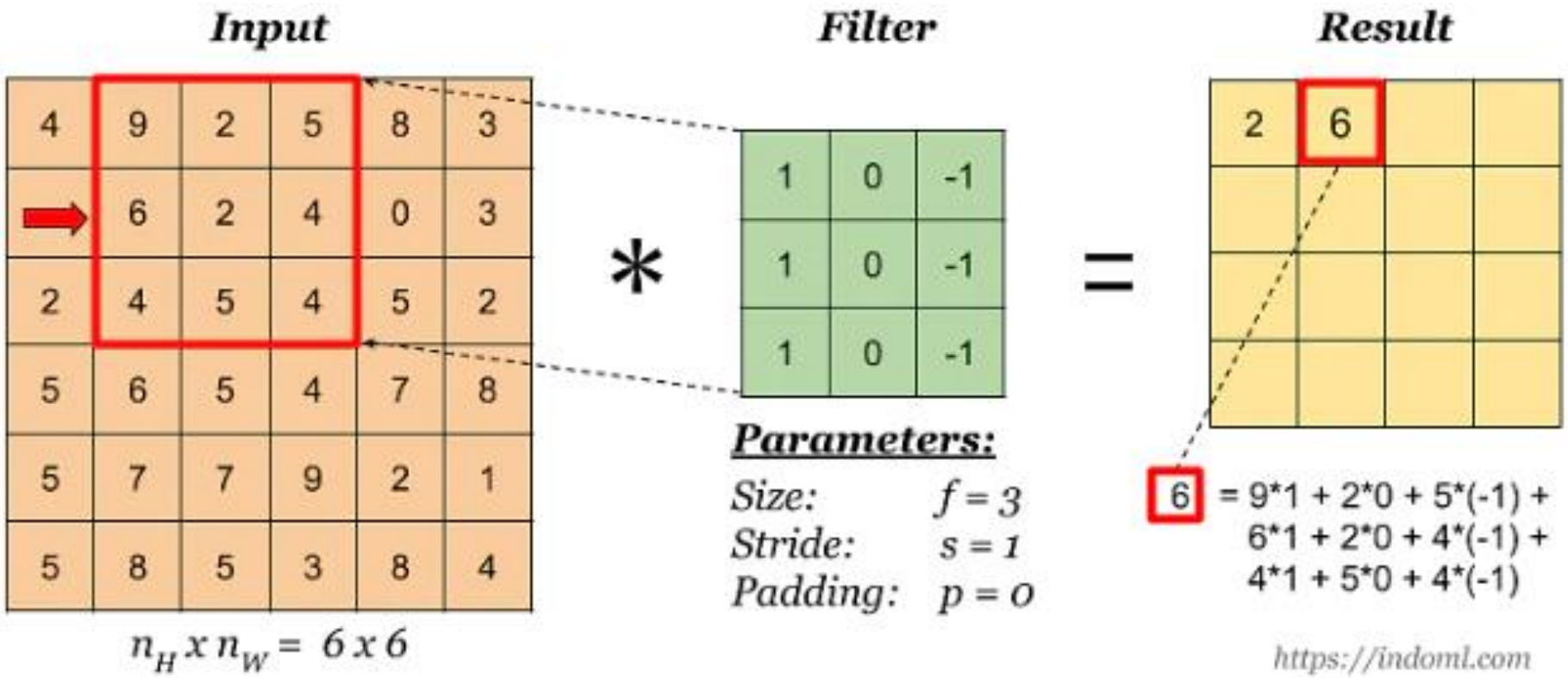
Amplia-se a entrada de forma que o volume de saída seja igual ao da entrada

CONVOLUÇÃO 2D



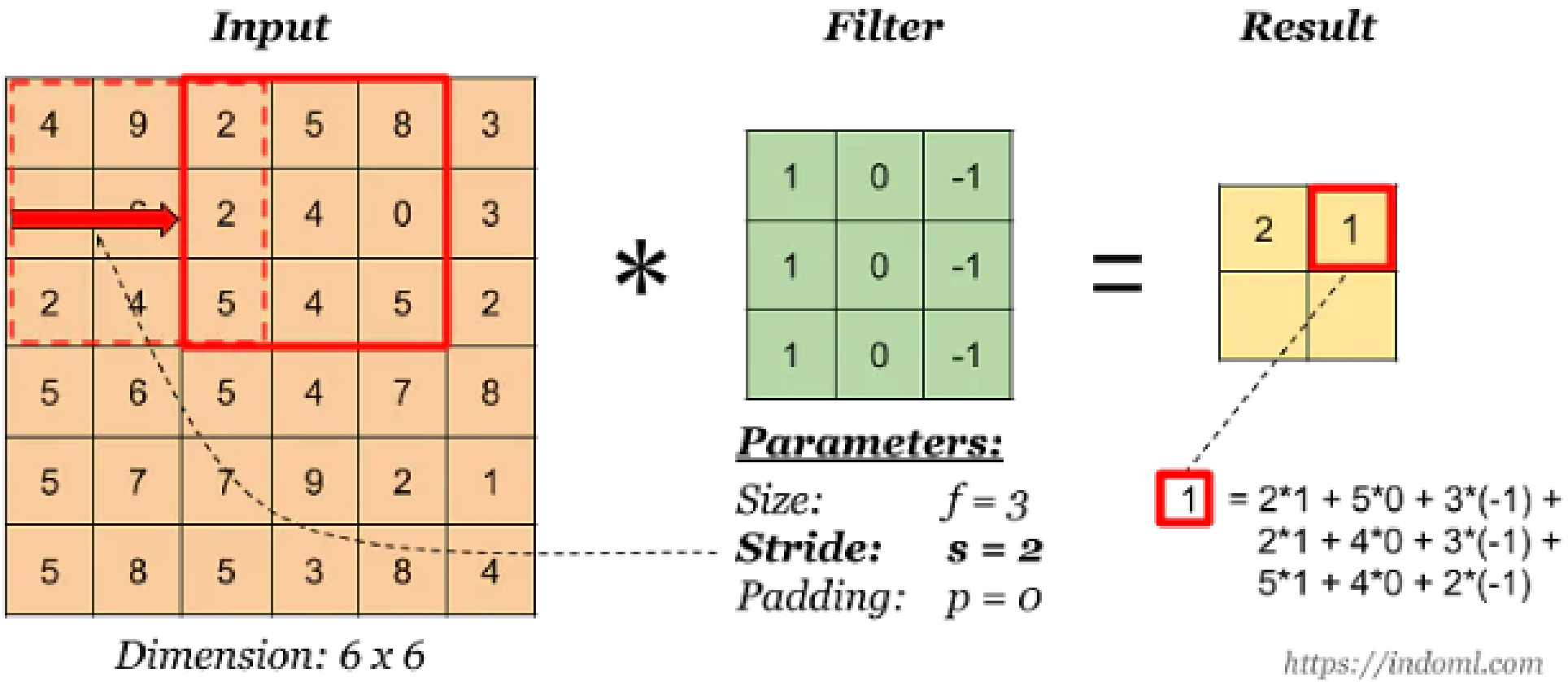
Source: indoml.com

CONVOLUÇÃO 2D



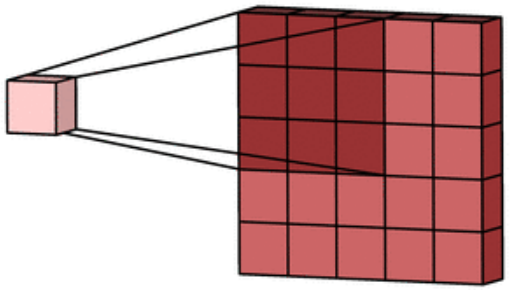
Source: indoml.com

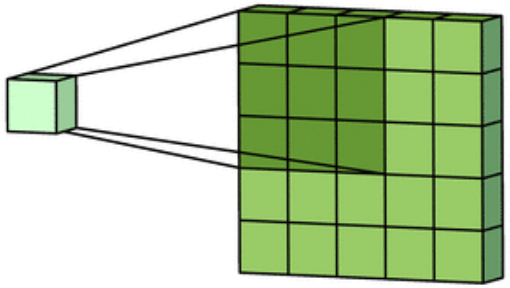
CONVOLUÇÃO 2D

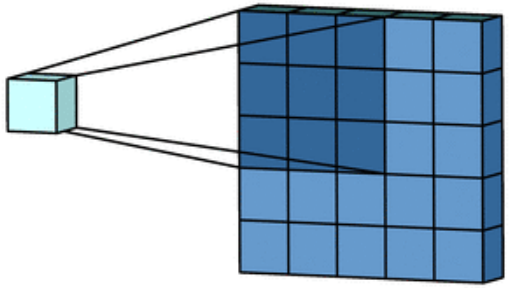


Source: indoml.com

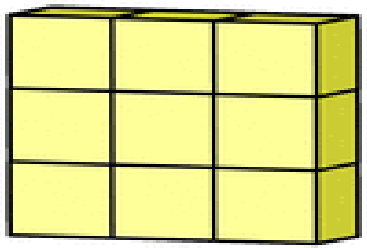
CONVOLUÇÃO 3D







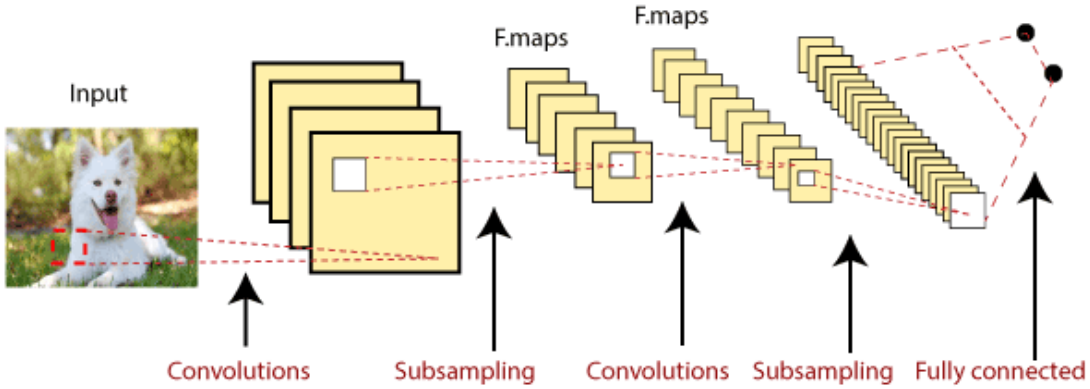
CONVOLUÇÃO 3D



CONVOLUÇÃO 3D

- ENTRADA (mxnxp)

  (32 x 32 x 3)



FILTRO (kxkxp)

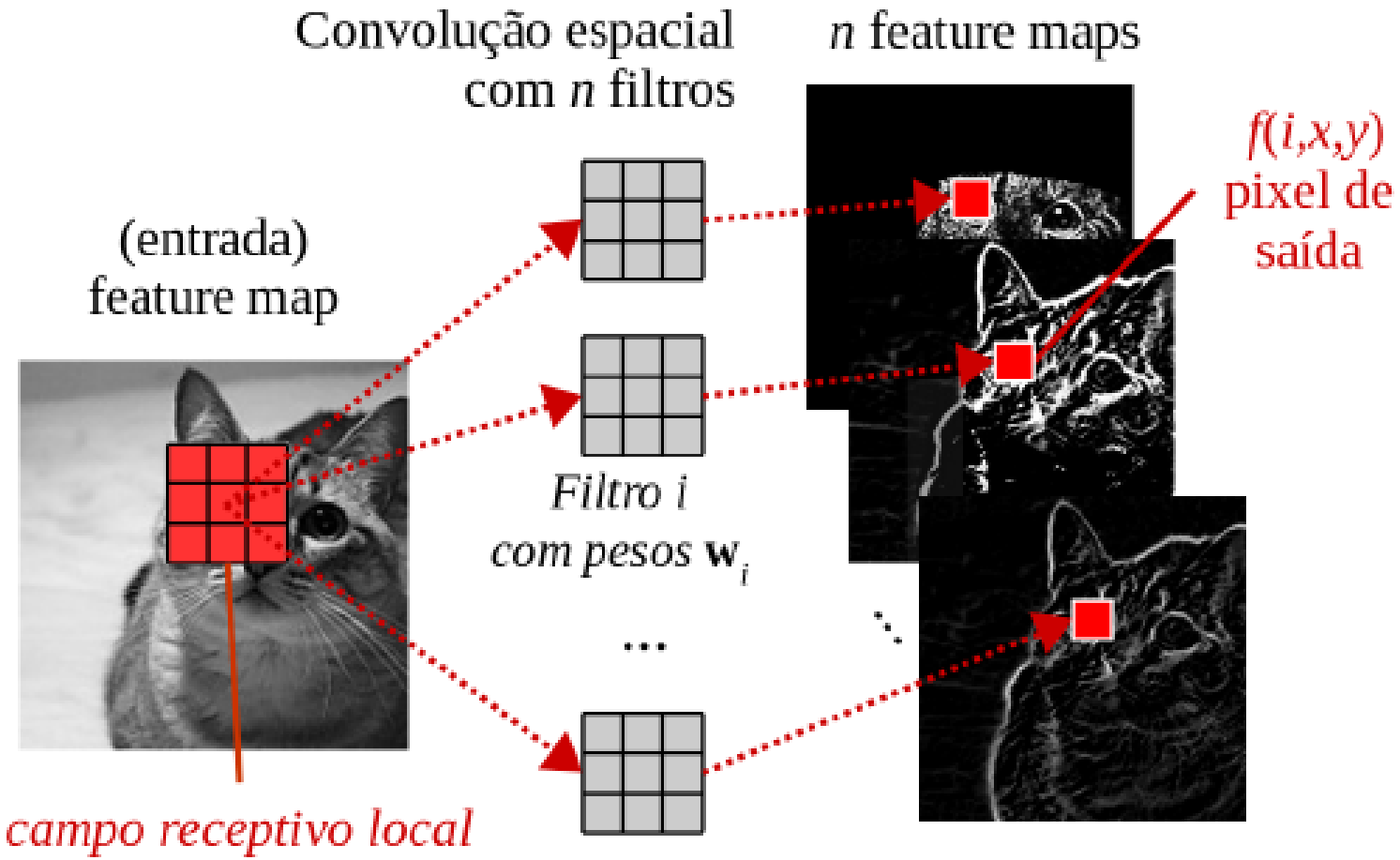
(5x5x3)

1. Cada neurônio realiza a convolução da entrada e gera um volume (matriz/tensor) de saída ➔ w x + b
2. Mapas de Ativação: são obtidos após a convolução e função de ativação (RELU)

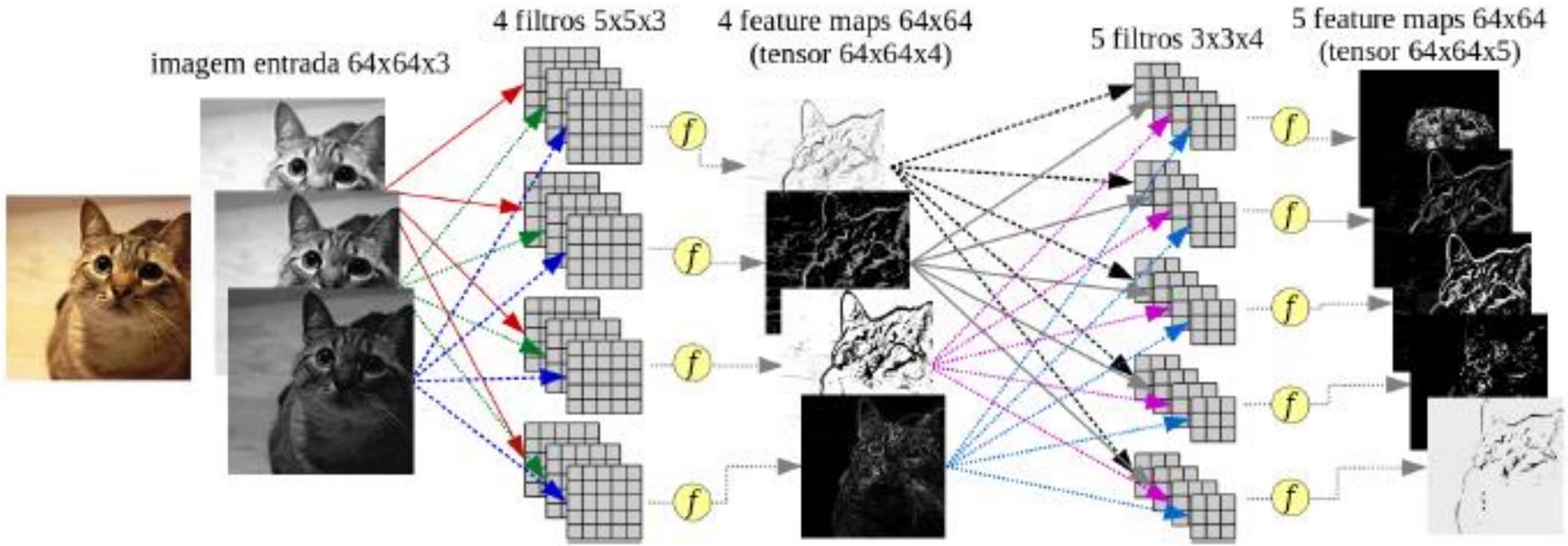
Camada Convolucional

- Feature Maps: Empilhados formam um tensor que será a entrada da próxima camada.

Camada Convolucional



Camada Convolucional



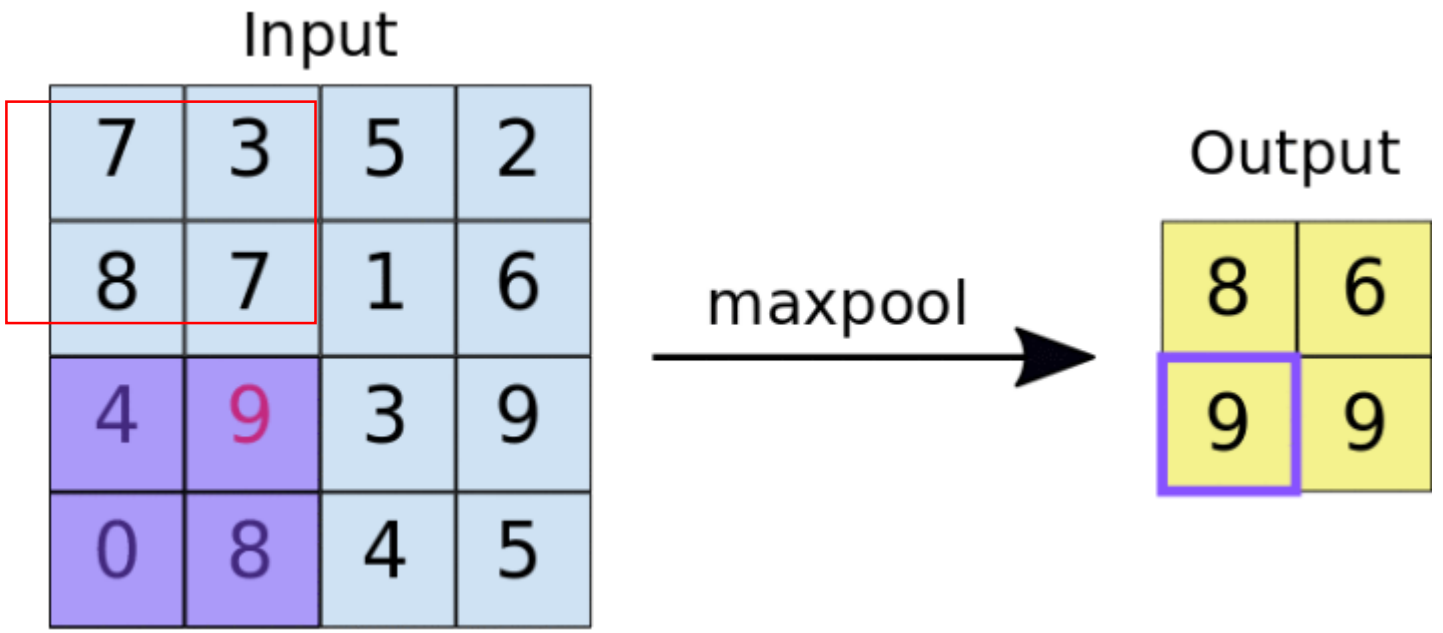
Camada Convolucional

A camada convolucional tem que levar em conta:

- o tamanho da entrada (largura, altura, profundidade)
- o tamanho do filtro
  - a profundidade deve ser igual à da entrada
  - a altura e largura afetam o campo receptivo local
- stride (passo)
  - = 1 : todos os pixels são filtrados pelo neurônio
  - > 1 : - salta um número de pixels em determinada direção, a cada convolução.
    - volume de saída tem tamanho reduzido

SUBAMOSTRAGEM: POOLING LAYER

1- Operação MAX pooling 2x2



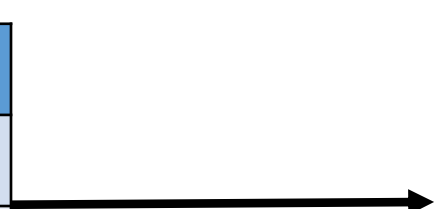
Reduzir o tamanho da entrada permite que o filtro opere em regiões maiores da imagem

SUBAMOSTRAGEM: POOLING LAYER

2 - Operação AVERAGE pooling 2x2

Input

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



Saída

6	**3**
5	5

CAMPO RECEPTIVO

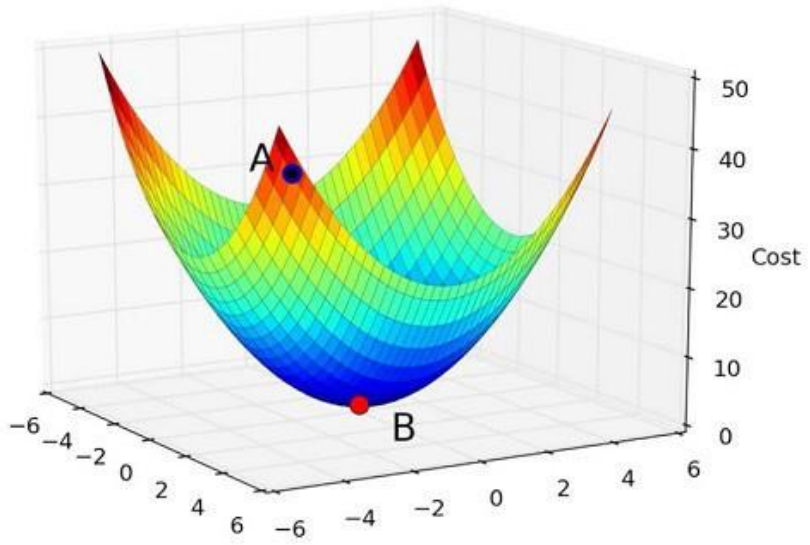
- EMPILHAMENTO DE CAMADAS CONVOLUCIONAIS AUMENTA O CAMPO RECEPTIVO LOCAL NÃO NECESSITANDO MANTER A RESOLUÇÃO DA ENTRADA

- filtro de mesmo tamanho
- imagens progressivamente menores

GLOBAL POOLING

- Obtém um valor por canal, como se o tamanho de pool fosse igual às dimensões laterais
- Ex. numa entrada com 40 x 40 x 100, a saída será 100 dimensões.

FUNÇÃO BINARY CROSS ENTROPY



OUTRA FUNÇÃO DE CUSTO

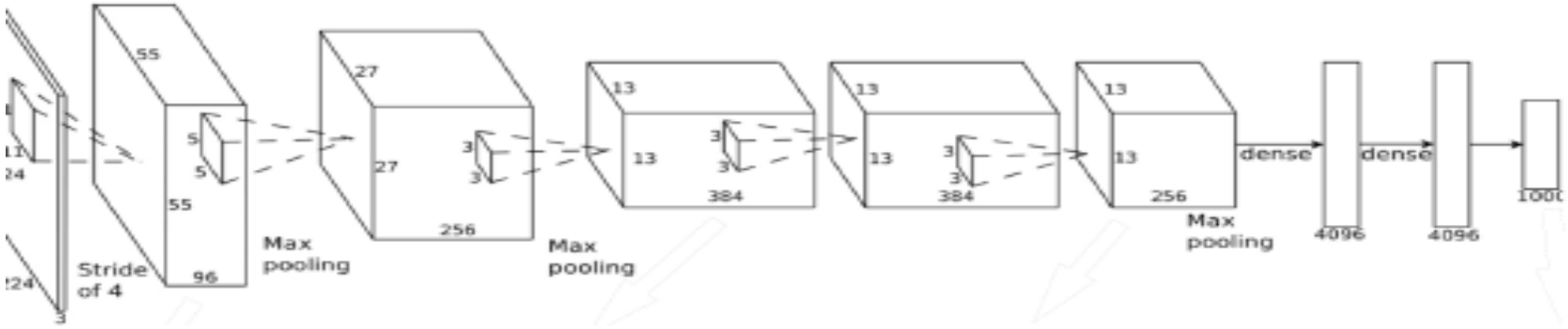
- Usando probabilidade condicional:
  h(x) = P(y = 1 , x)
  1 - h(x) = P(y = 0 , x) (pela regra de probabilidade complementar)
- Combinando as 2 equações, obtemos:
  $P(y,x) = (h(x)^y) * ((1 - h(x))^{(1-y)})$

$$log(P(y/x)) = \sum (y(i)log(h(x)) + (1 - y(i))log((1 - h(x))))$$

ARQUITETURAS TÍPICAS DE CNN

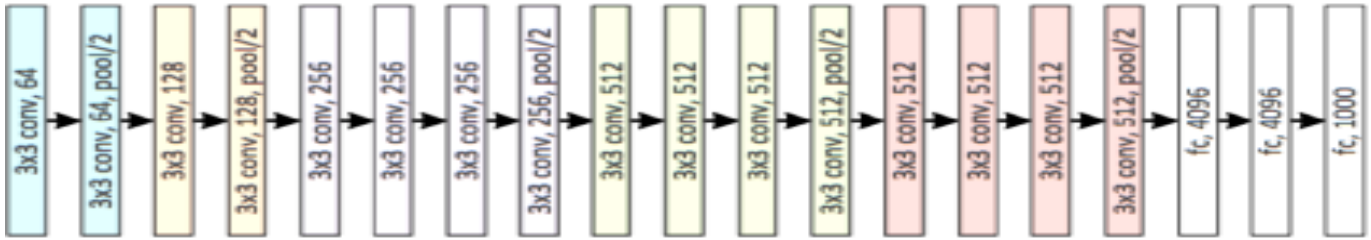
AlexNet (Krizhevsky, 2012)

- entrada 224×224
- conv1: $K = 96$ filters with $11 \times 11 \times 3$, stride 4,
- conv2: $K = 256$ filters with $5 \times 5 \times 96$,
- conv3: $K = 384$ filters with $3 \times 3 \times 256$,
- conv4: $K = 384$ filters with $3 \times 3 \times 384$,
- conv5: $K = 256$ filters with $3 \times 3 \times 384$,
- densas1, 2: $K = 4096$.



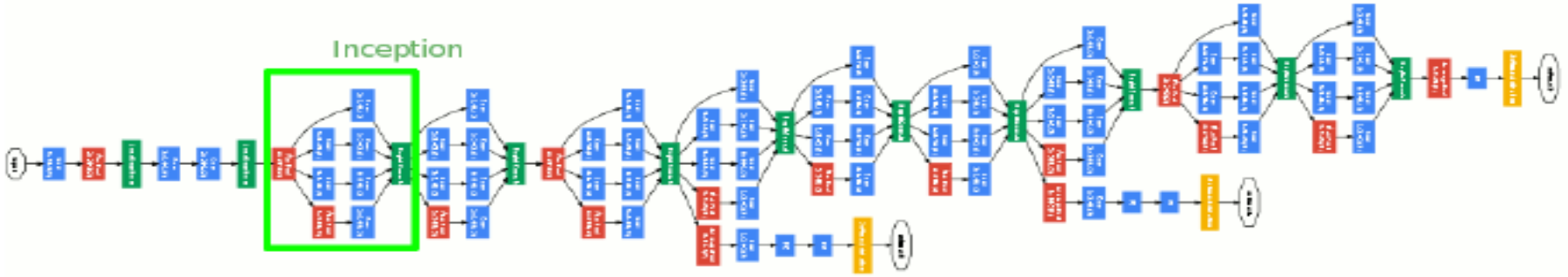
VGGNet (Simonyan, 2014)

- entrada 224×224,
- filtros: todos 3×3,
- conv 1-2: $K = 64$ + maxpool
- conv 3-4: $K = 128$ + maxpool
- conv 5-6-7-8: $K = 256$ + maxpool
- conv 9-10-11-12: $K = 512$ + maxpool
- conv 13-14-15-16: $K = 512$ + maxpool
- densas1, 2: $K = 4096$

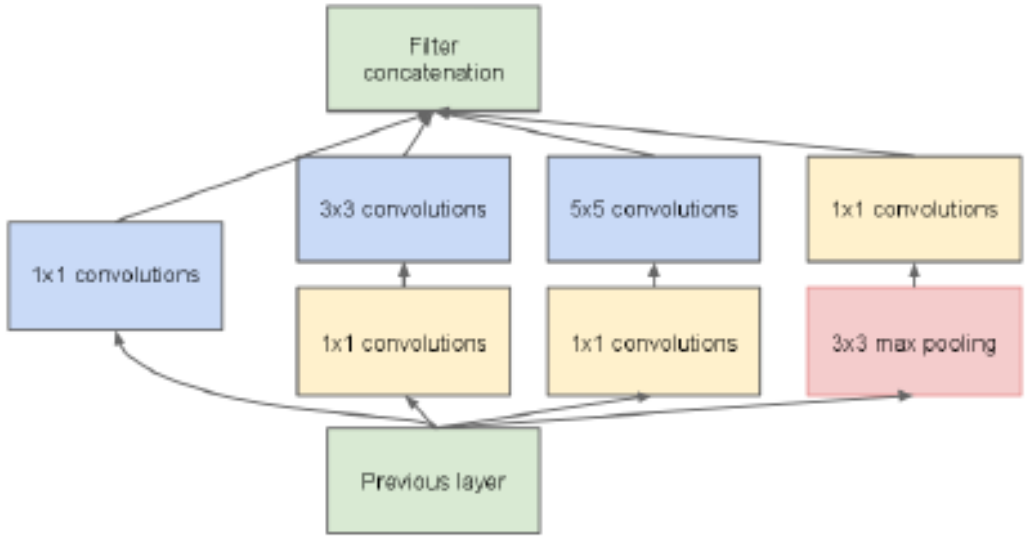


GoogLeNet / Inception (Szegedy, 2014)

- 22 layers (v1)
- *Inception layer* (banco de filtros):
  - filtros 1×1, 3×3, 5×5 + max pooling 3×3;
  - controla dimensionalidade usando filtros 1×1.
  - 3 classificadores (não sequenciais)
- Azul = conv.; Vermelho = pool.; Amarelo = densa+softmax; Verde = concatenação.

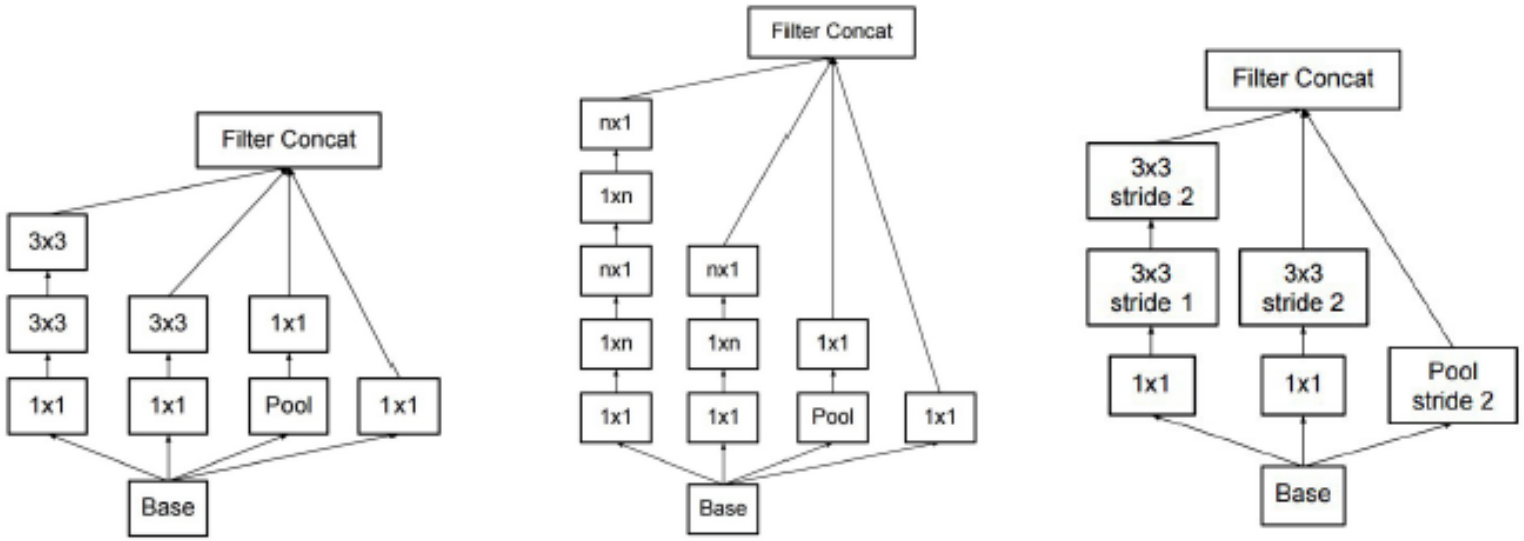


GoogLeNet: módulo inception v1

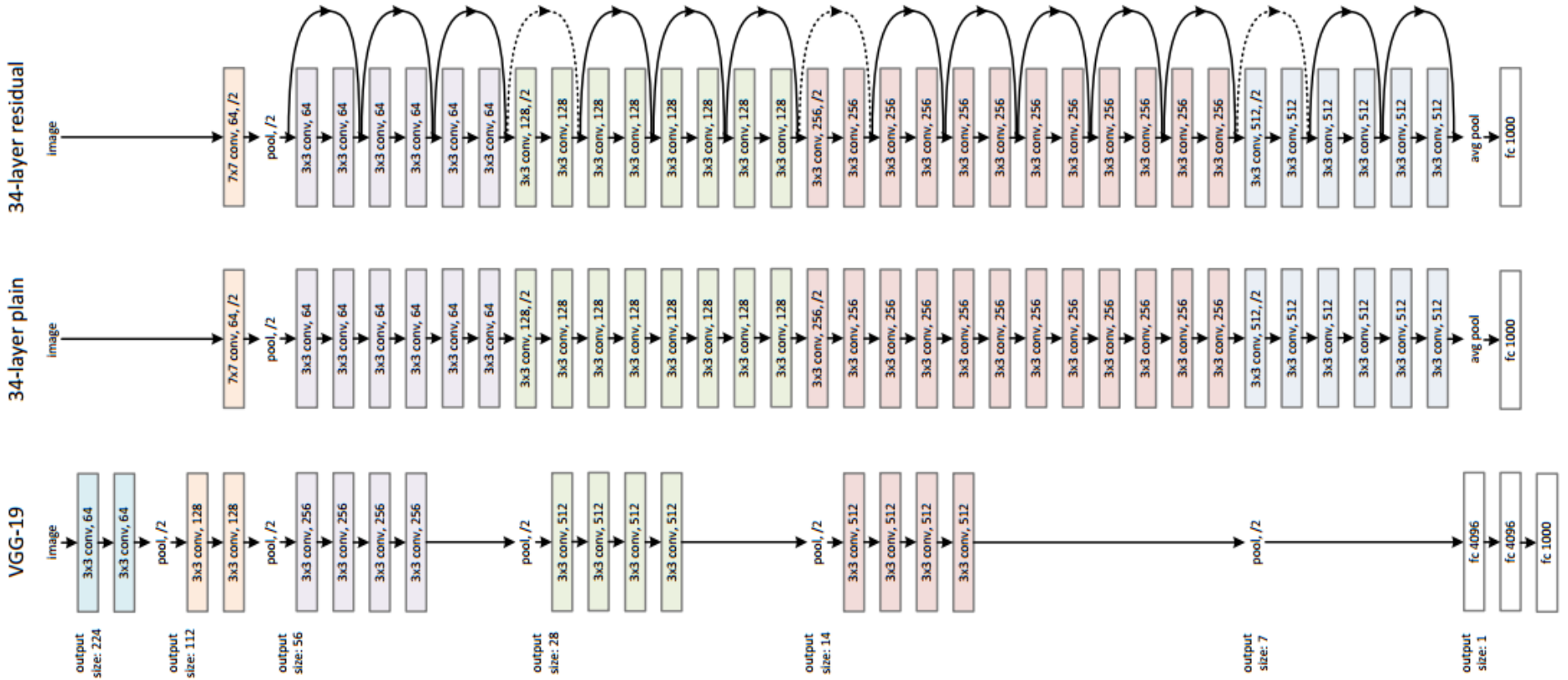


- filtro 1×1 reduz profundidade da entrada
- concatena ativação de 3 filtros + maxpooling

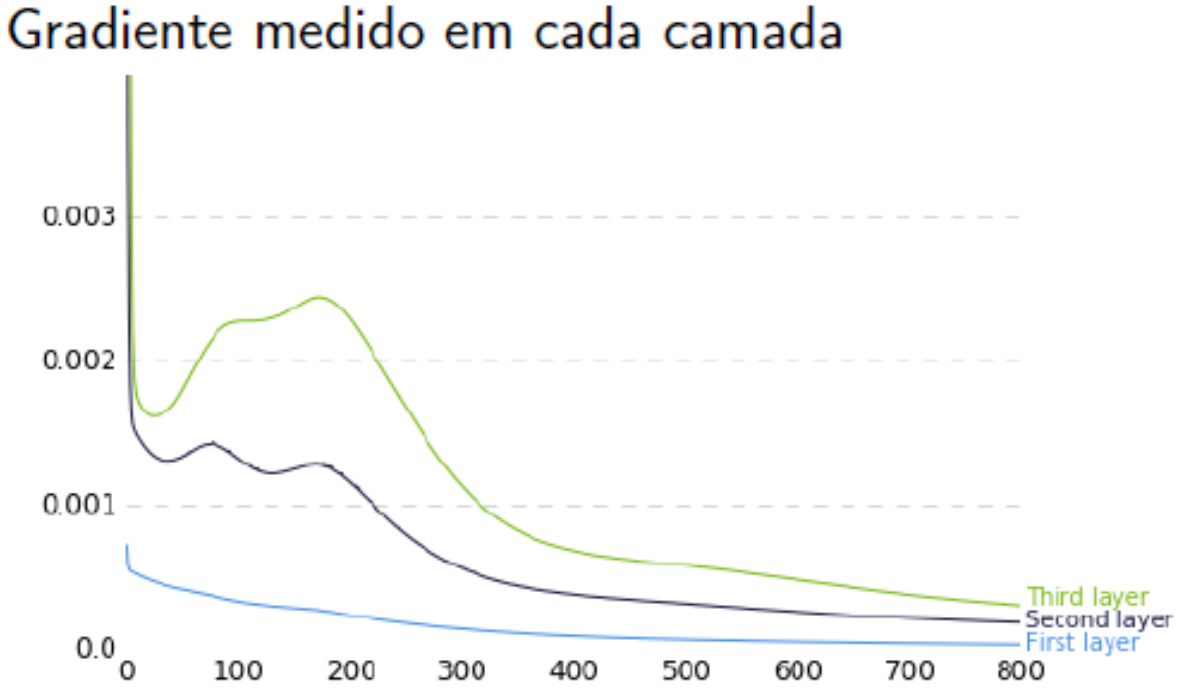
Módulos inception (V2 and V3)



Residual Network (He et al, 2015)



O gradiente não se comporta igual em todas as camadas

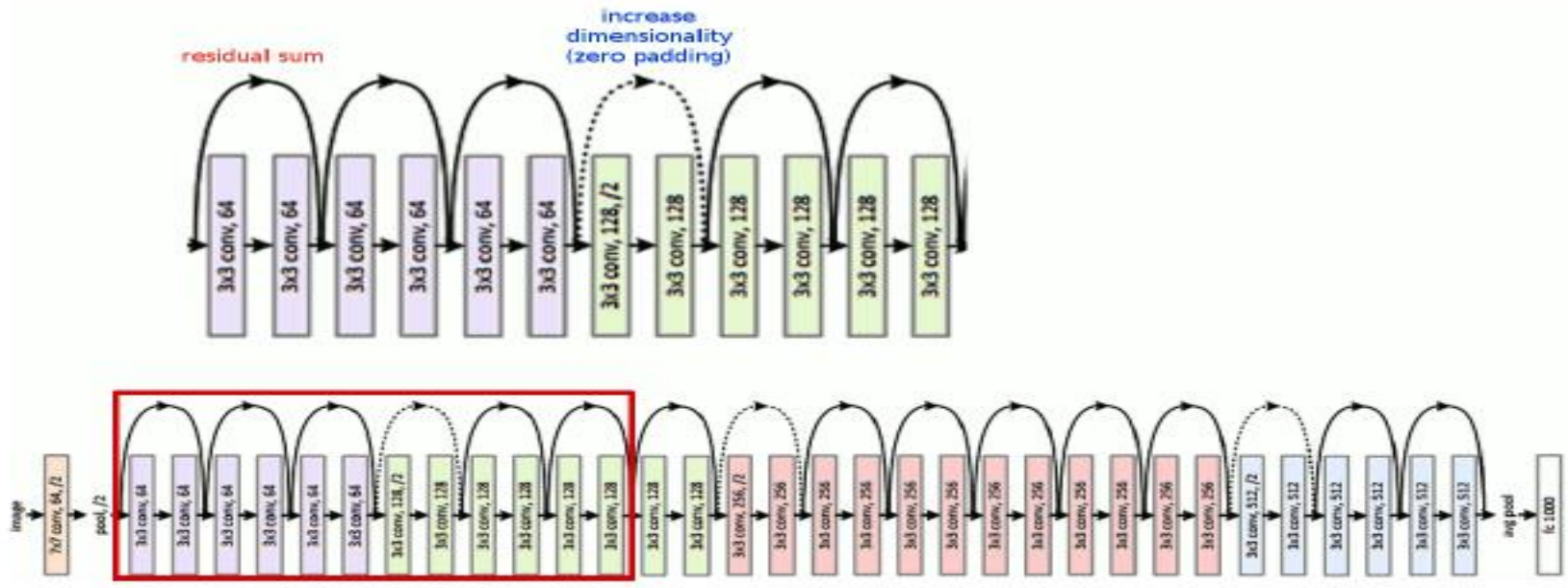


Agradecimentos a Harini suresh (http://harinisuresh.com) pelos gráficos

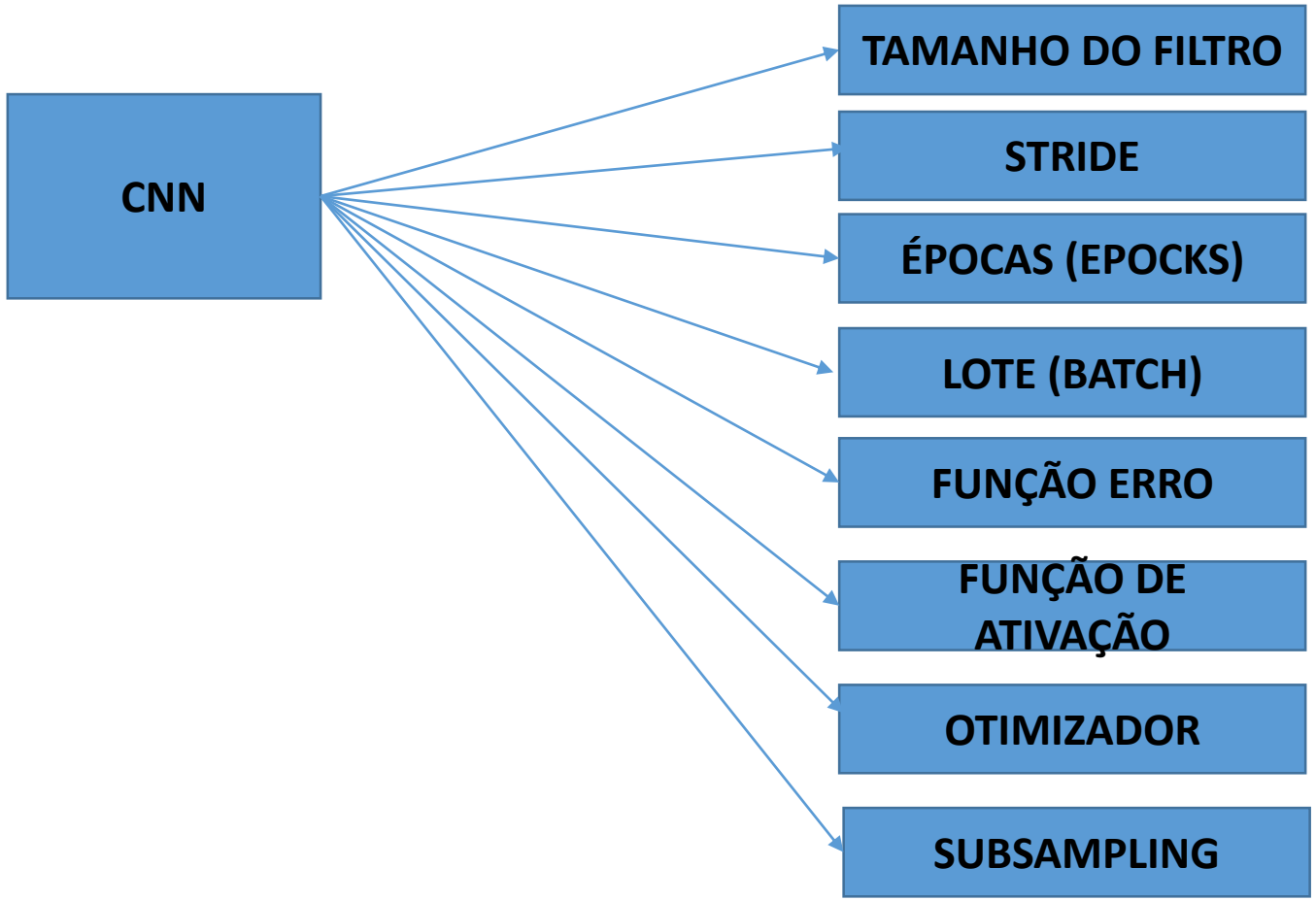
Residual Network — ResNet (He et al, 2015)

Pular camadas (skip layers) permite:
– empilhar mais camadas (de 34 a $\sim$ 1000).

Residual: adiciona resultado anterior preservando gradiente.



IMPLEMENTAÇÃO DA CNN



EXEMPLO 2

CNN
USO DO KERAS

EXEMPLO 3

FUNÇÃO CROSS-ENTROPY