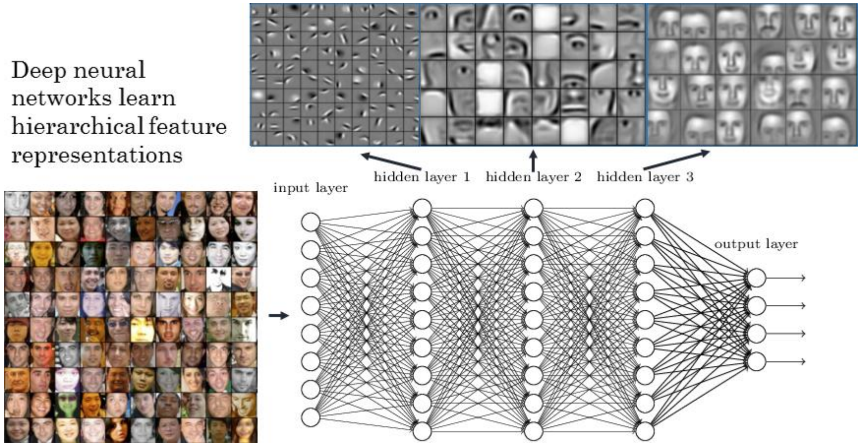
Redes Neurais e Deep Learning

André E. Lazzaretti

UTFPR/CPGEI



UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E INFORMÁTICA INDUSTRIAL
CPGEI

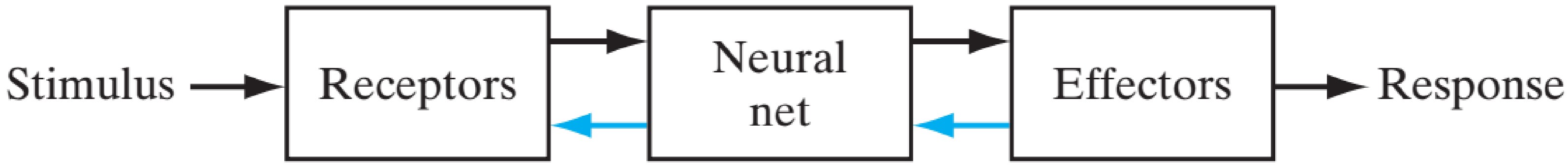
Introdução

- Definição formal de rede neural:

A neural network is a massively parallel distributed processor made up of simple processing units that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:

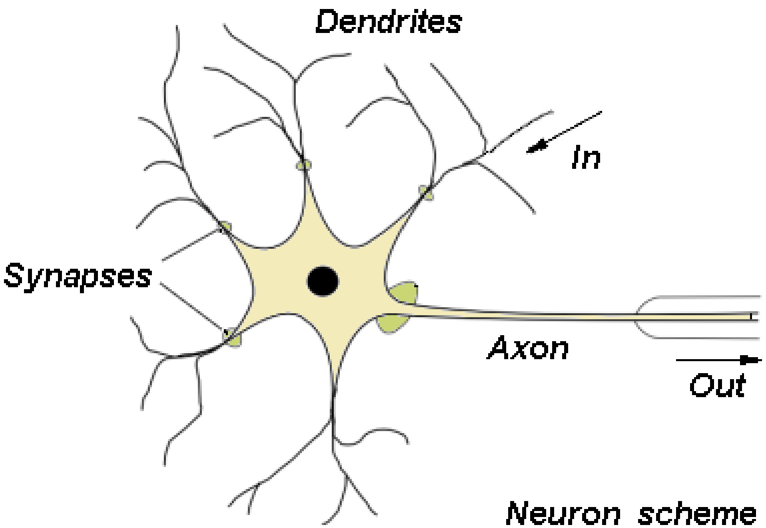
1. *Knowledge is acquired by the network from its environment through a learning process.*
2. *Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.*

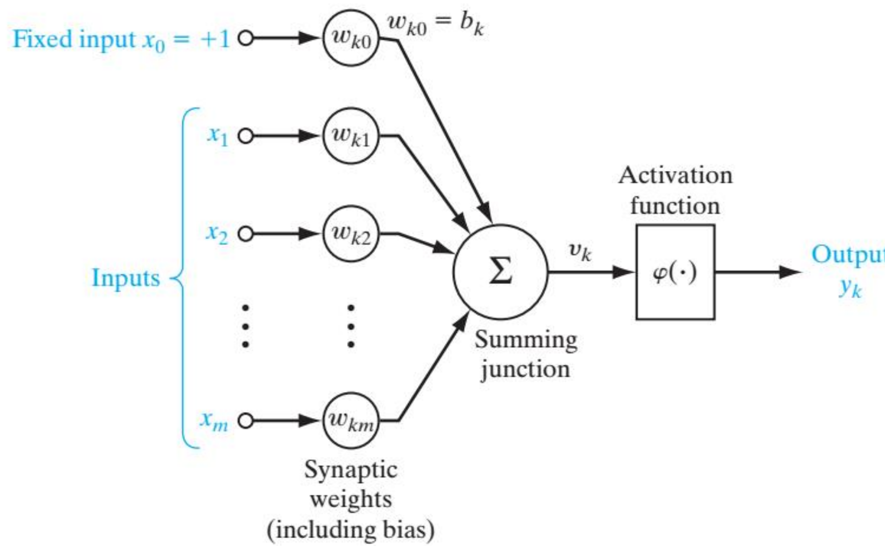
- Sistema nervoso:

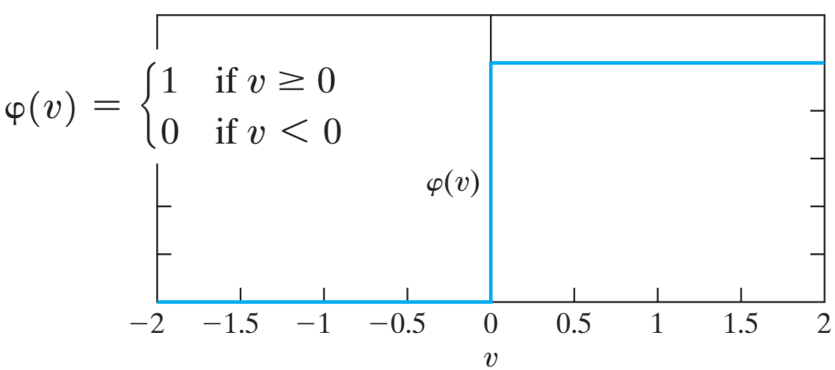


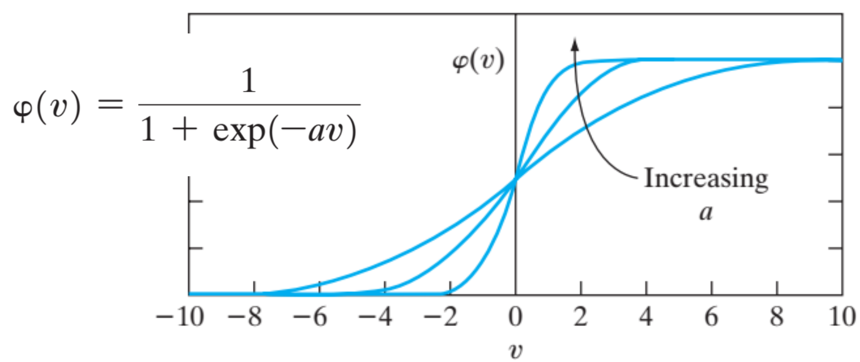
Introdução

- Modelo de neurônio artificial:



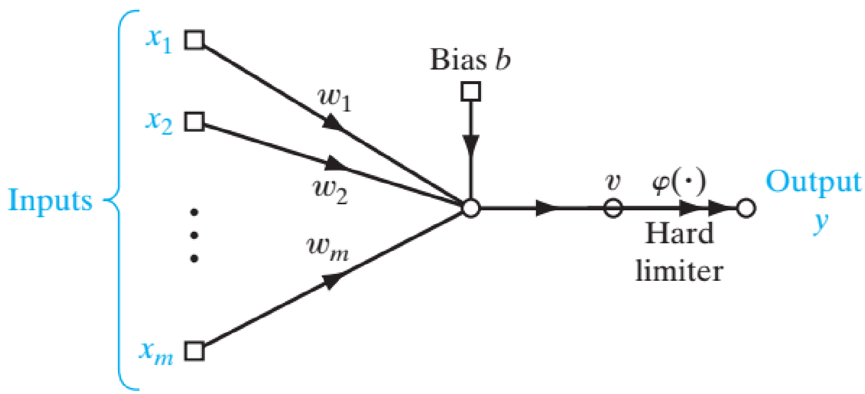


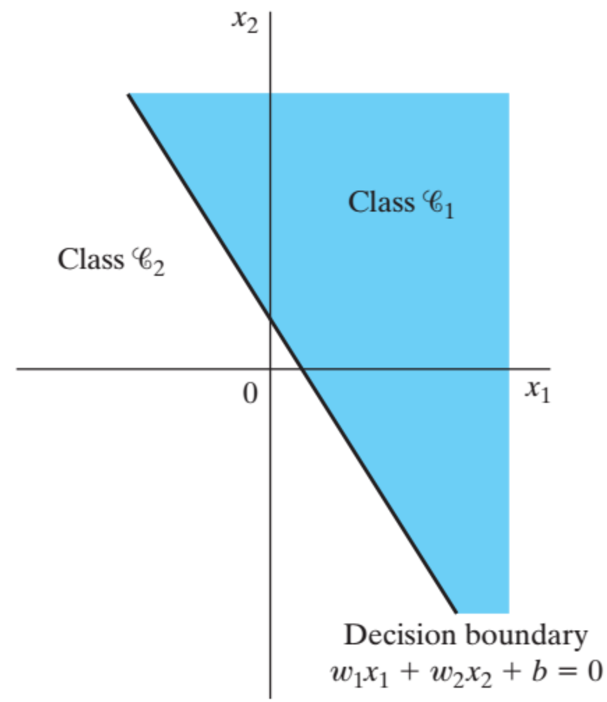




Perceptron

- Classificador linear: hiperplano de separação





$$\sum_{i=1}^{m} w_i x_i + b = 0$$

Perceptron

Avalia se o padrão foi corretamente classificado:

$$\boldsymbol{w}^{*T}\boldsymbol{x} > 0 \quad \forall \boldsymbol{x} \in \omega_1$$

$$\boldsymbol{w}^{*T}\boldsymbol{x} < 0 \quad \forall \boldsymbol{x} \in \omega_2$$

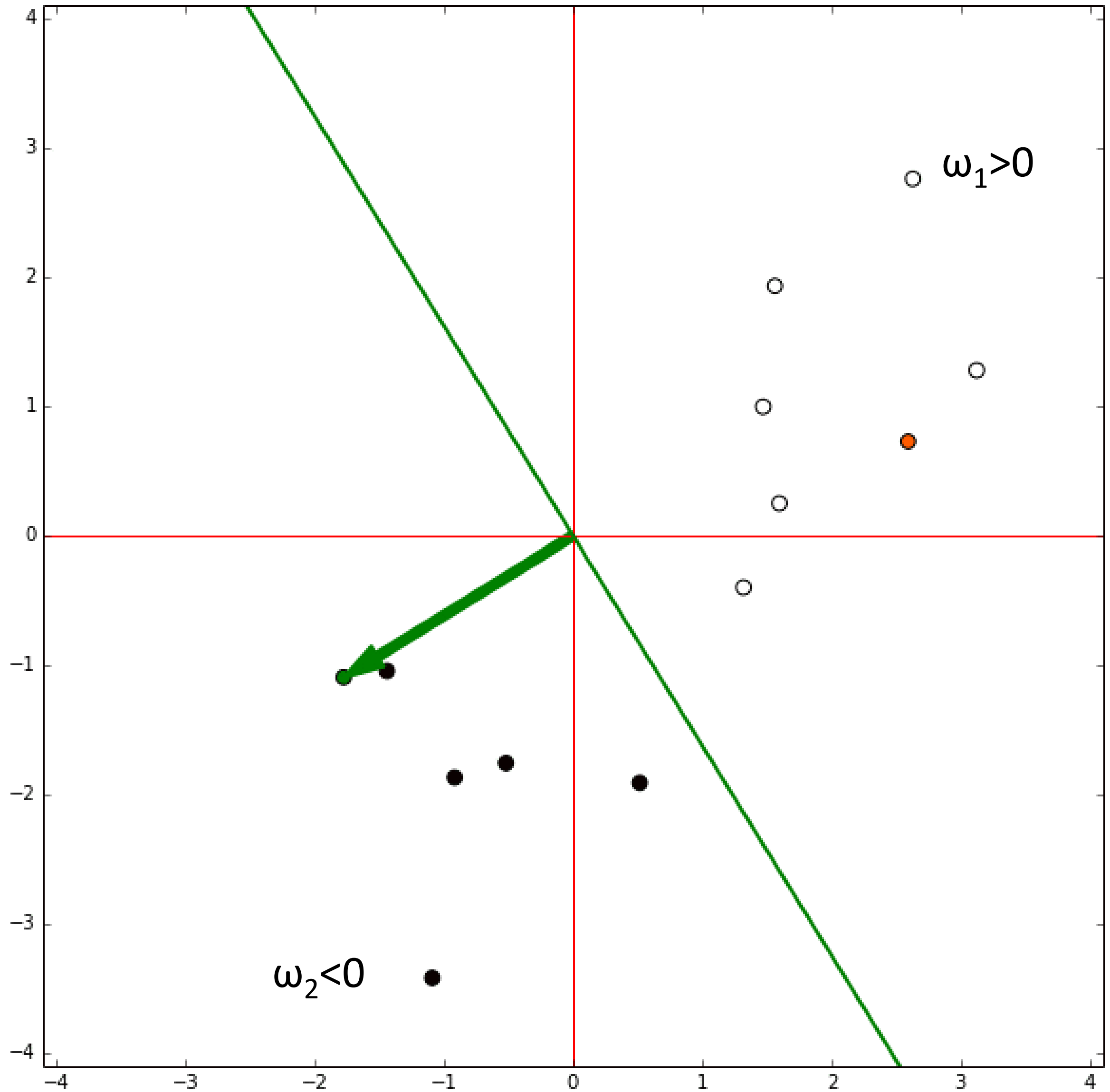
Se estiver incorreto, atualiza **w**:

$$\boldsymbol{w}(t+1) = \boldsymbol{w}(t) - \rho_t \sum_{\boldsymbol{x} \in Y} \delta_x \boldsymbol{x}$$

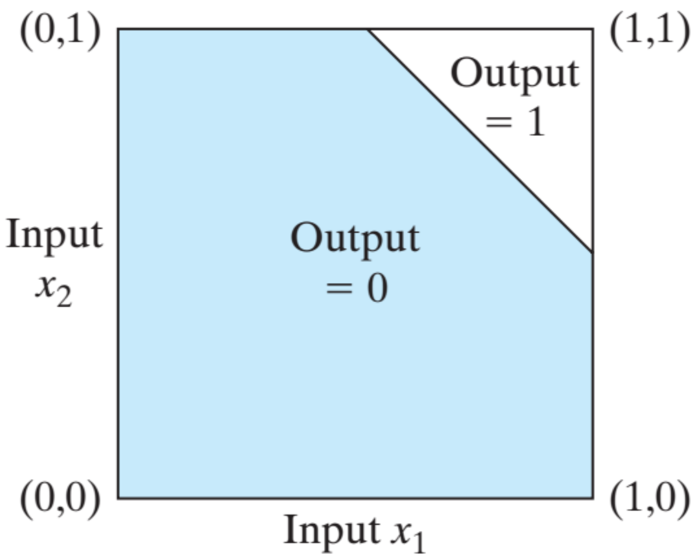
Learning rate

$$\delta_x = -1 \text{ if } \boldsymbol{x} \in \omega_1$$

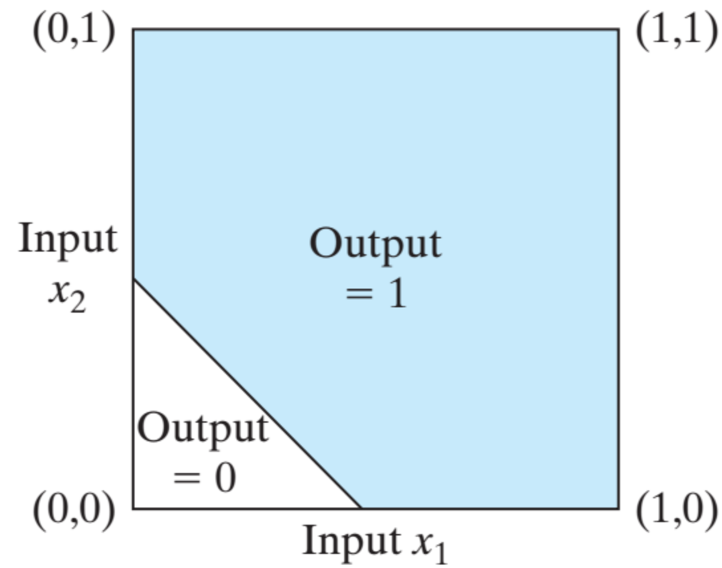
$$\delta_x = +1 \text{ if } \boldsymbol{x} \in \omega_2$$

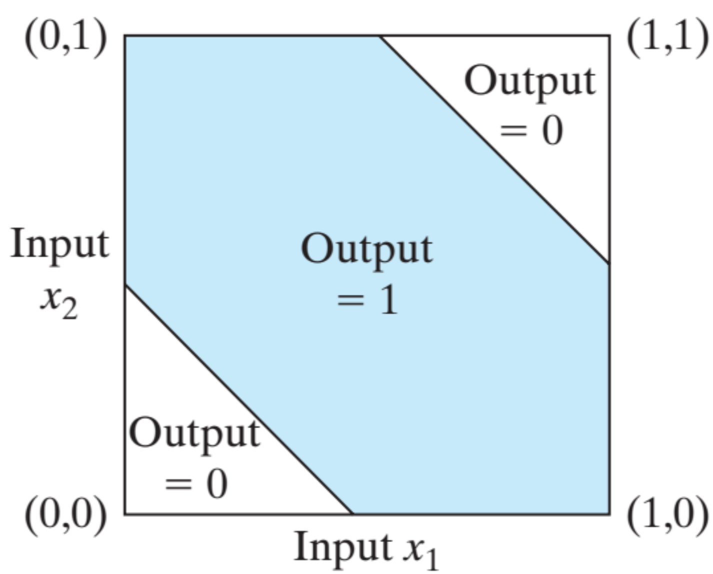


Problema: XOR

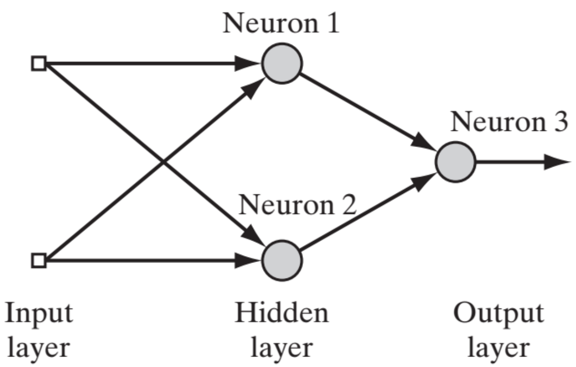


$$0 \oplus 0 = 0$$
$$1 \oplus 1 = 0$$
$$0 \oplus 1 = 1$$
$$1 \oplus 0 = 1$$

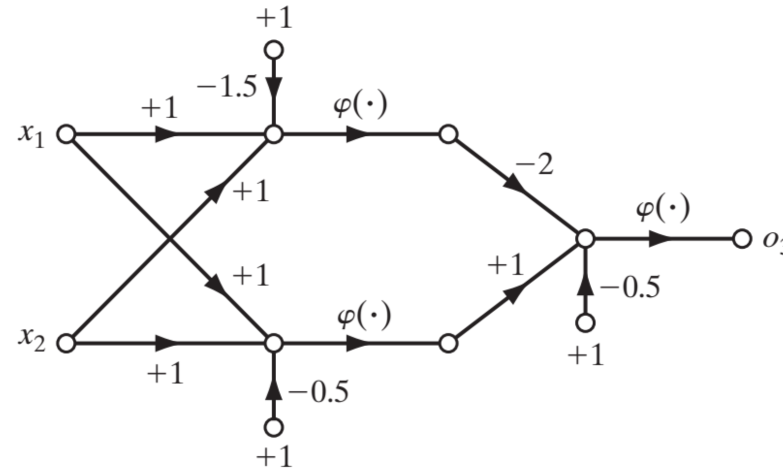




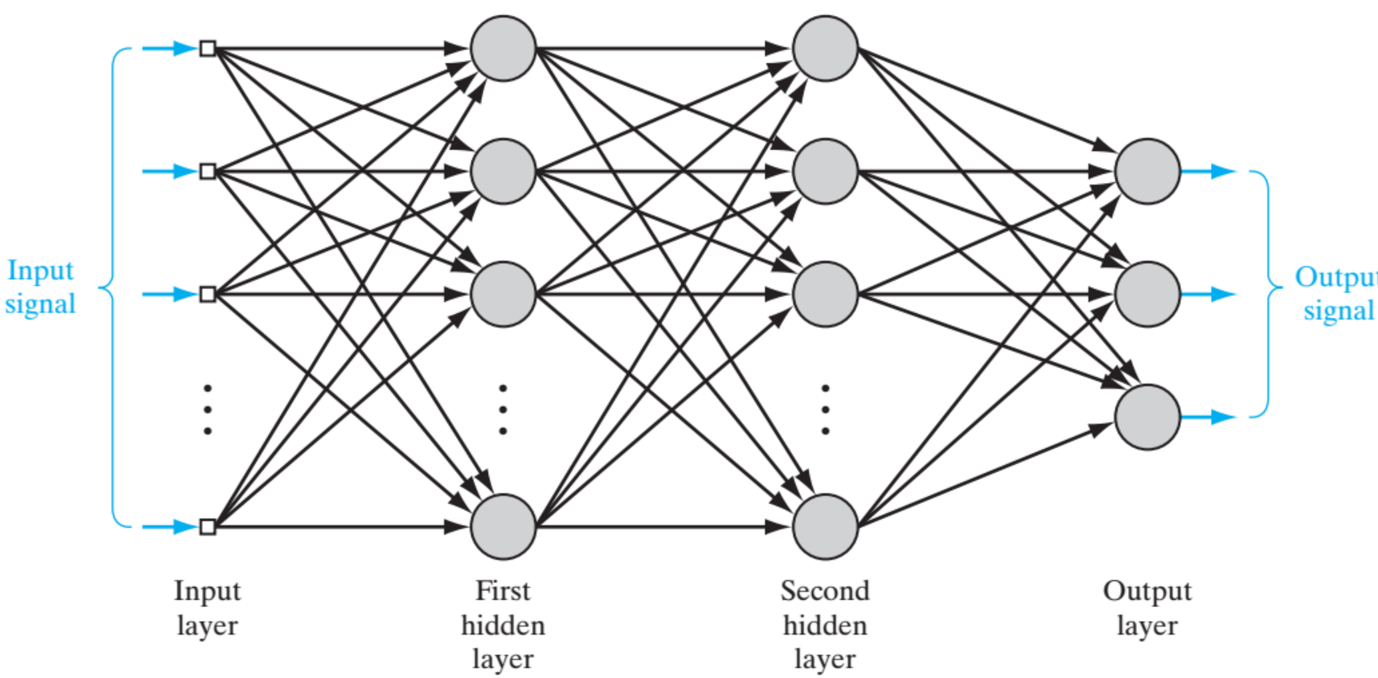
Alternativa: Aumentar a Arquitetura



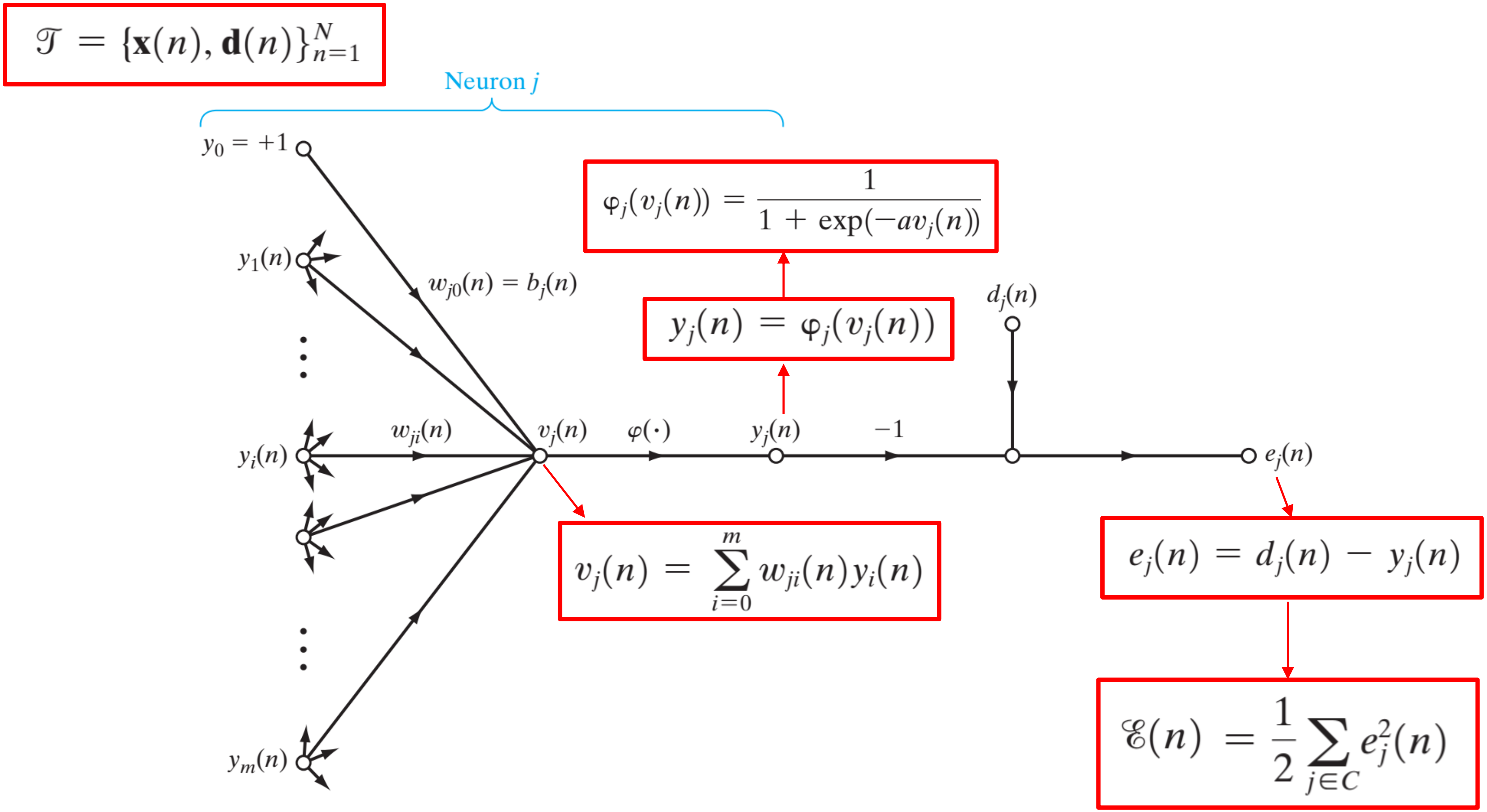
$$0 \oplus 0 = 0$$
$$1 \oplus 1 = 0$$
$$0 \oplus 1 = 1$$
$$1 \oplus 0 = 1$$



Multilayer Perceptron:



Treinamento: Backpropagation



Backpropagation: Última Camada

Descida em gradiente:

$$\boxed{\boldsymbol{w}_{NOVO} = \boldsymbol{w}_{ANTERIOR} - \eta \frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)}}$$

$$\frac{\partial \mathscr{E}(n)}{\partial e_j(n)} = e_j(n)$$

$$\mathscr{E}(n) = \frac{1}{2} \sum_{j \in C} e_j^2(n)$$

$$\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'(v_j(n))$$

$$y_j(n) = \varphi_j(v_j(n))$$

$$\frac{\partial \mathscr{E}(n)}{\partial w_{ji}(n)} = \frac{\partial \mathscr{E}(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)}$$

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1$$

$$e_j(n) = d_j(n) - y_j(n)$$

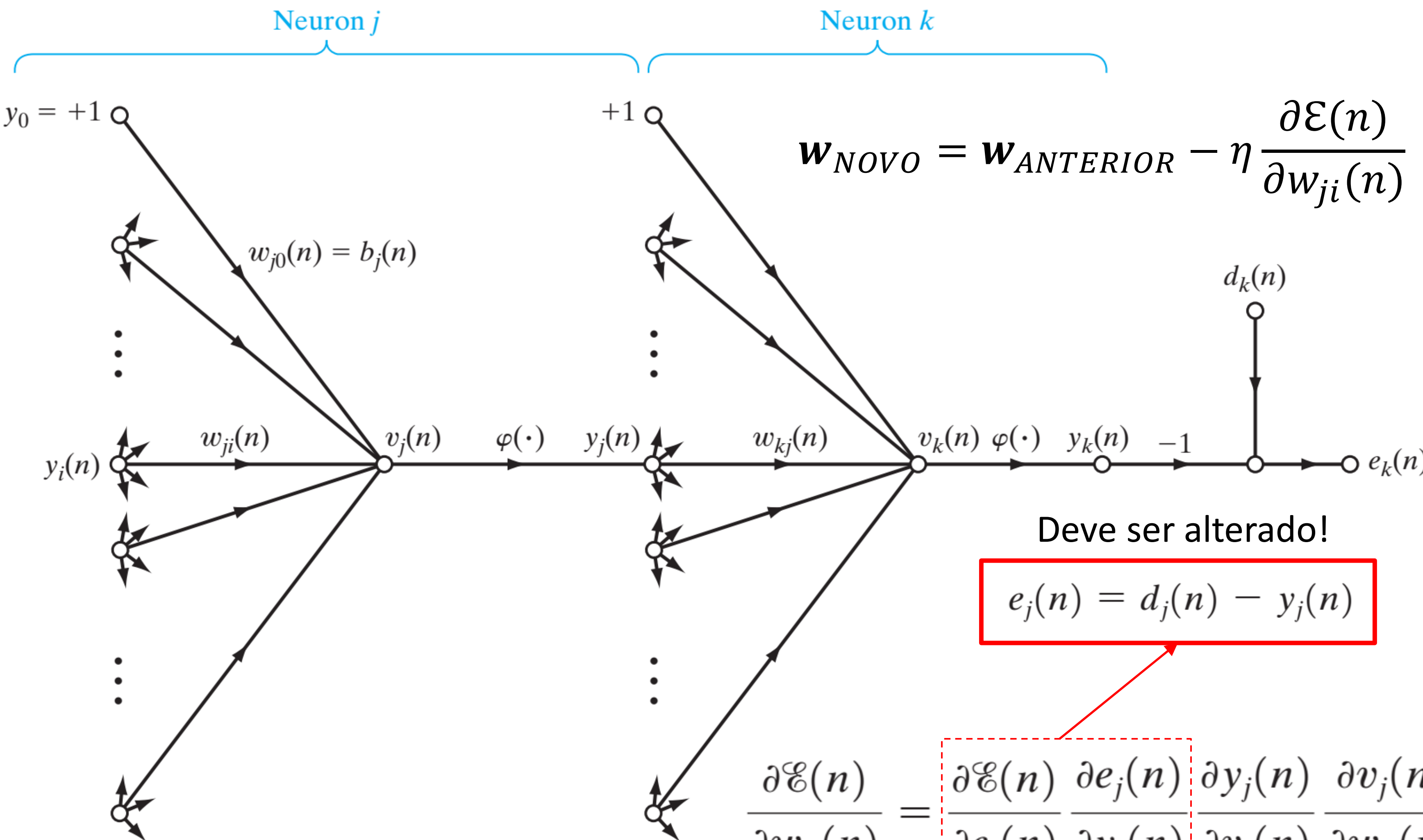
$$\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n)$$

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) y_i(n)$$

$$\boxed{\frac{\partial \mathscr{E}(n)}{\partial w_{ji}(n)} = -e_j(n) \varphi_j'(v_j(n)) y_i(n)}$$

https://en.wikipedia.org/wiki/Matrix_calculus

Backpropagation: Camada Oculta



$$\boldsymbol{w}_{NOVO} = \boldsymbol{w}_{ANTERIOR} - \eta \frac{\partial \mathcal{E}(n)}{\partial w_{ji}(n)}$$

Deve ser alterado!

$$\boxed{e_j(n) = d_j(n) - y_j(n)}$$

$$\frac{\partial \mathscr{E}(n)}{\partial w_{ji}(n)} = \frac{\partial \mathscr{E}(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)}$$

Alternativa: suprimir $e_j(n)$ e usar somente $e_k(n)$

Backpropagation: Última Camada

Permanece igual!

$$\frac{\partial \mathscr{E}(n)}{\partial w_{ji}(n)} = \frac{\partial \mathscr{E}(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)}$$

$$\boxed{\frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n)} \qquad \boxed{\frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'(v_j(n))}$$

I) $$\mathscr{E}(n) = \frac{1}{2}\sum_{k \in C} e_k^2(n)$$

II) $$\frac{\partial \mathscr{E}(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial y_j(n)}$$

III) $$\frac{\partial \mathscr{E}(n)}{\partial y_j(n)} = \sum_k e_k(n) \frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)}$$

IV) $$\begin{aligned} e_k(n) &= d_k(n) - y_k(n) \\ &= d_k(n) - \varphi_k(v_k(n)) \end{aligned}$$

V) $$\frac{\partial e_k(n)}{\partial v_k(n)} = -\varphi_k'(v_k(n))$$

VI) $$v_k(n) = \sum_{j=0}^{m} w_{kj}(n) y_j(n)$$

VII) $$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n)$$

VIII) $$\frac{\partial \mathscr{E}(n)}{\partial y_j(n)} = -\sum_k e_k(n)\varphi_k'(v_k(n)) w_{kj}(n)$$

$$\boxed{\frac{\partial \mathscr{E}(n)}{\partial w_{ji}(n)} = -\sum_k e_k(n)\varphi_k'(v_k(n)) w_{kj}(n)\ \varphi_j'(v_j(n))\ y_i(n)}$$

Backpropagation: Resumo

$$\begin{pmatrix}\textit{Weight} \\ \textit{correction} \\ \Delta w_{ji}(n)\end{pmatrix} = \begin{pmatrix}\textit{learning-} \\ \textit{rate parameter} \\ \eta\end{pmatrix} \times \begin{pmatrix}\textit{local} \\ \textit{gradient} \\ \delta_j(n)\end{pmatrix} \times \begin{pmatrix}\textit{input signal} \\ \textit{of neuron } j, \\ y_i(n)\end{pmatrix}$$

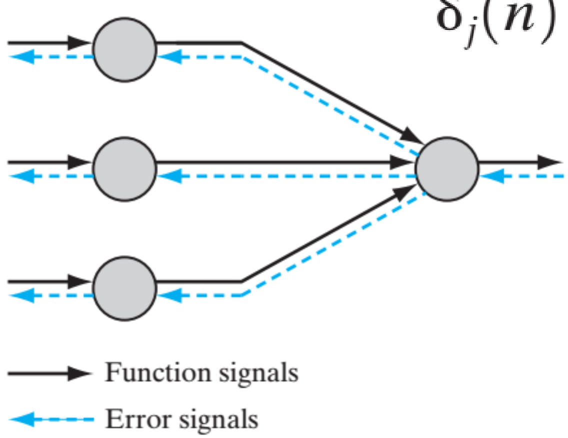
Second, the local gradient $\delta_j(n)$ depends on whether neuron j is an output node or a hidden node:

1. If neuron j is an output node, $\delta_j(n)$ equals the product of the derivative $\varphi_j'(v_j(n))$ and the error signal $e_j(n)$, both of which are associated with neuron j

$$\delta_j(n) = e_j(n)\varphi_j'(v_j(n))$$

2. If neuron j is a hidden node, $\delta_j(n)$ equals the product of the associated derivative $\varphi_j'(v_j(n))$ and the weighted sum of the δs computed for the neurons in the next hidden or output layer that are connected to neuron j

$$\delta_j(n) = \varphi_j'(v_j(n)) \sum_k \delta_k(n) w_{kj}(n),$$



Exemplo Matlab!

Deep Learning

PROC. OF THE IEEE, NOVEMBER 1998 1

Gradient-Based Learning Applied to Document Recognition

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca

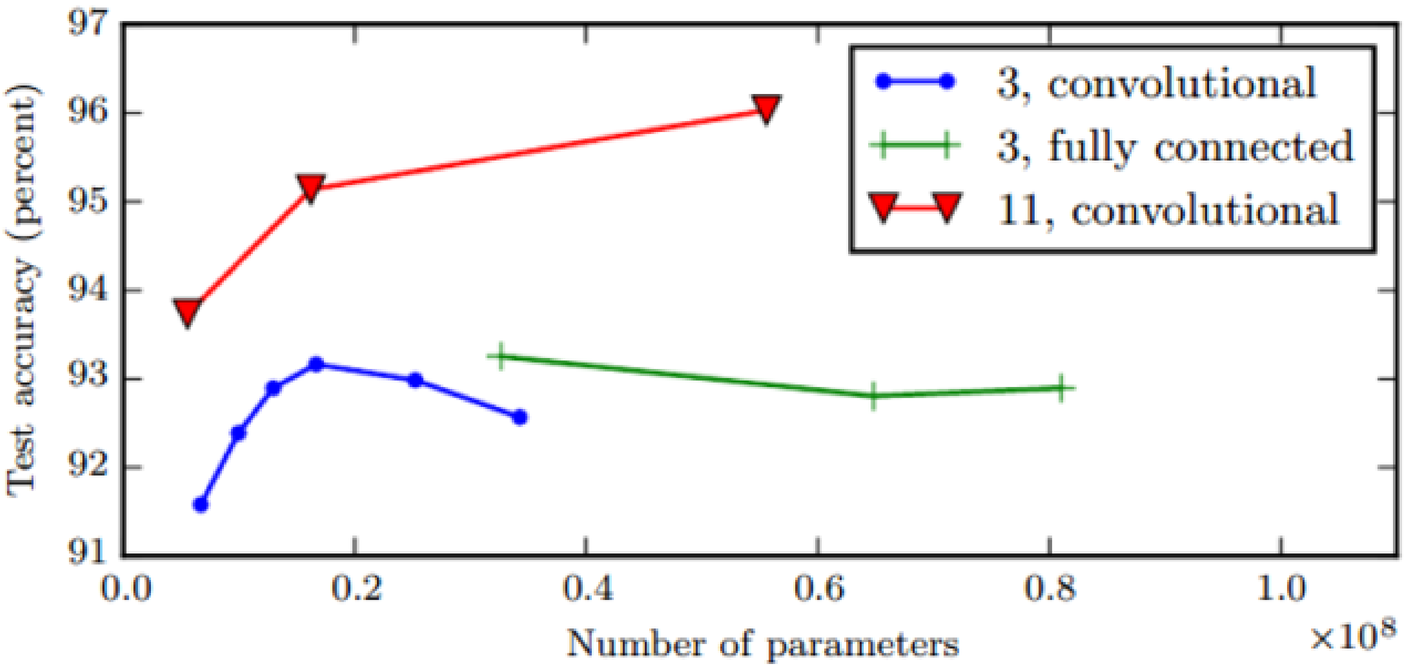
Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca

Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

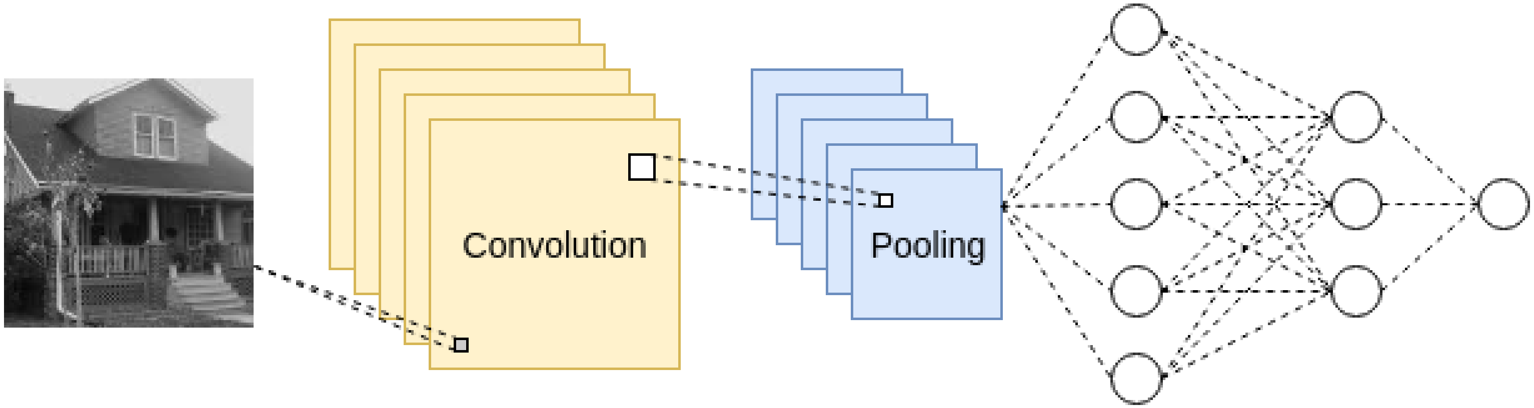
Por que *Deep Learning*?

- *Deeper models*: inserir camadas com diferentes características.

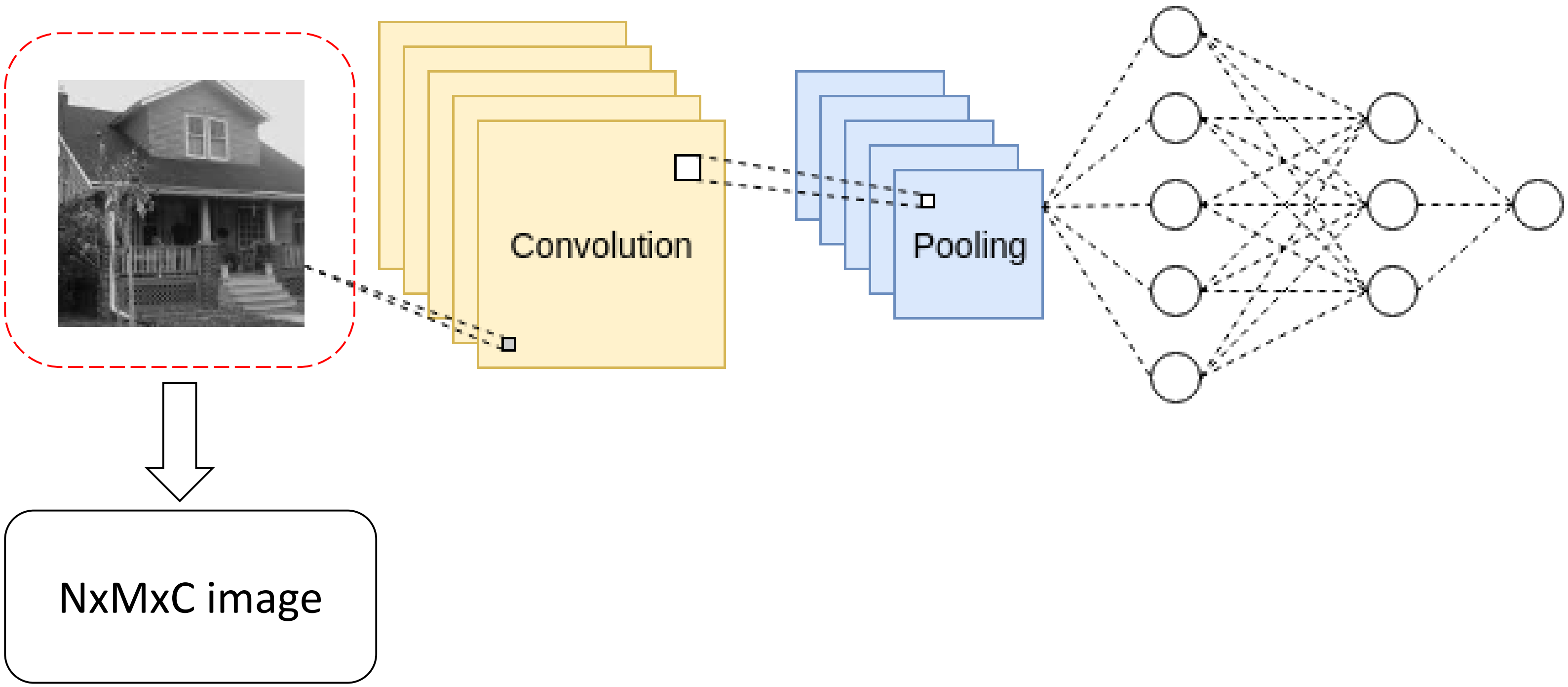


Convolutional Neural Network

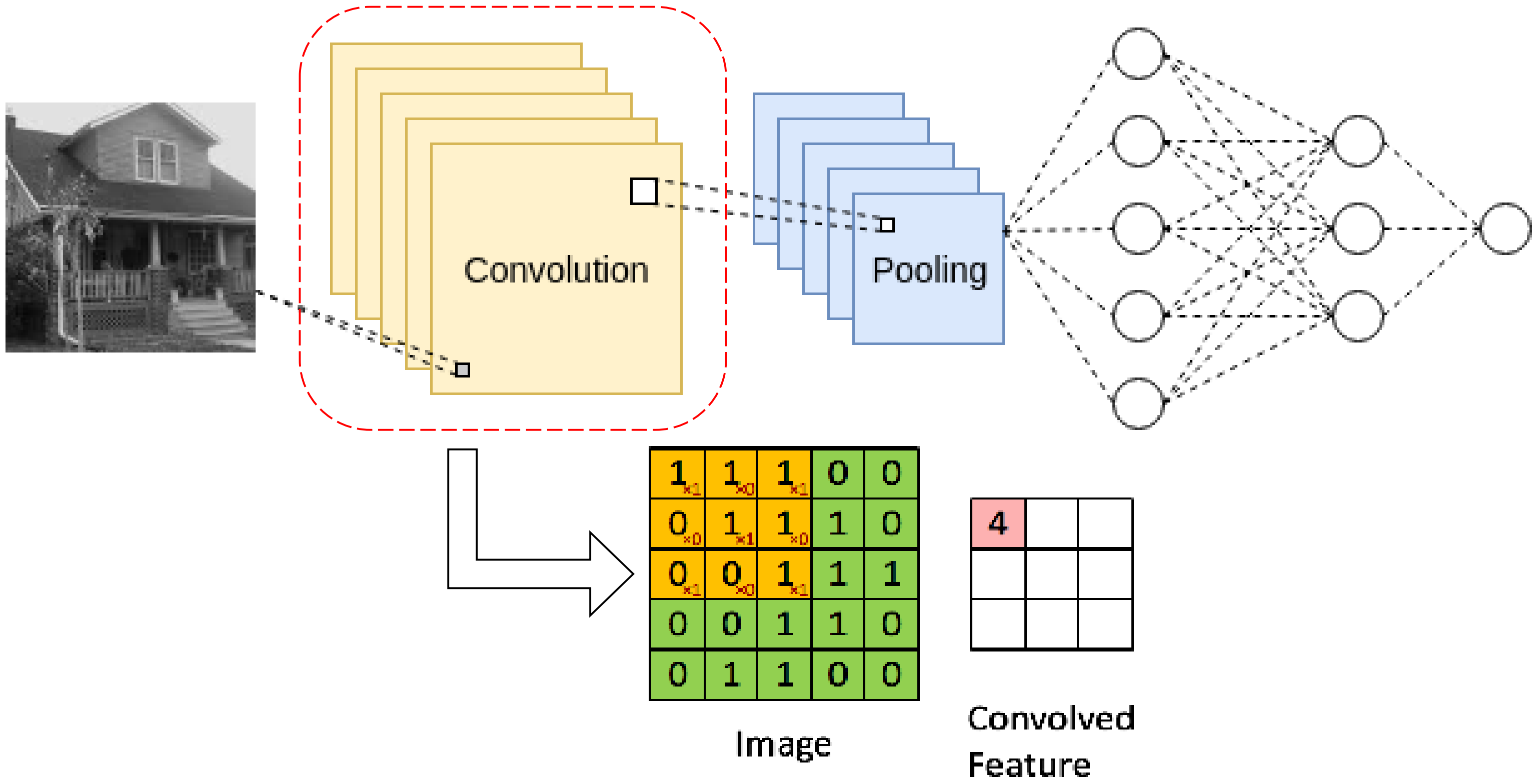
- **Definição formal:** *Convolutional networks are simply neural networks that use convolution in place of general matrix multiplication in at least one of their layers.*



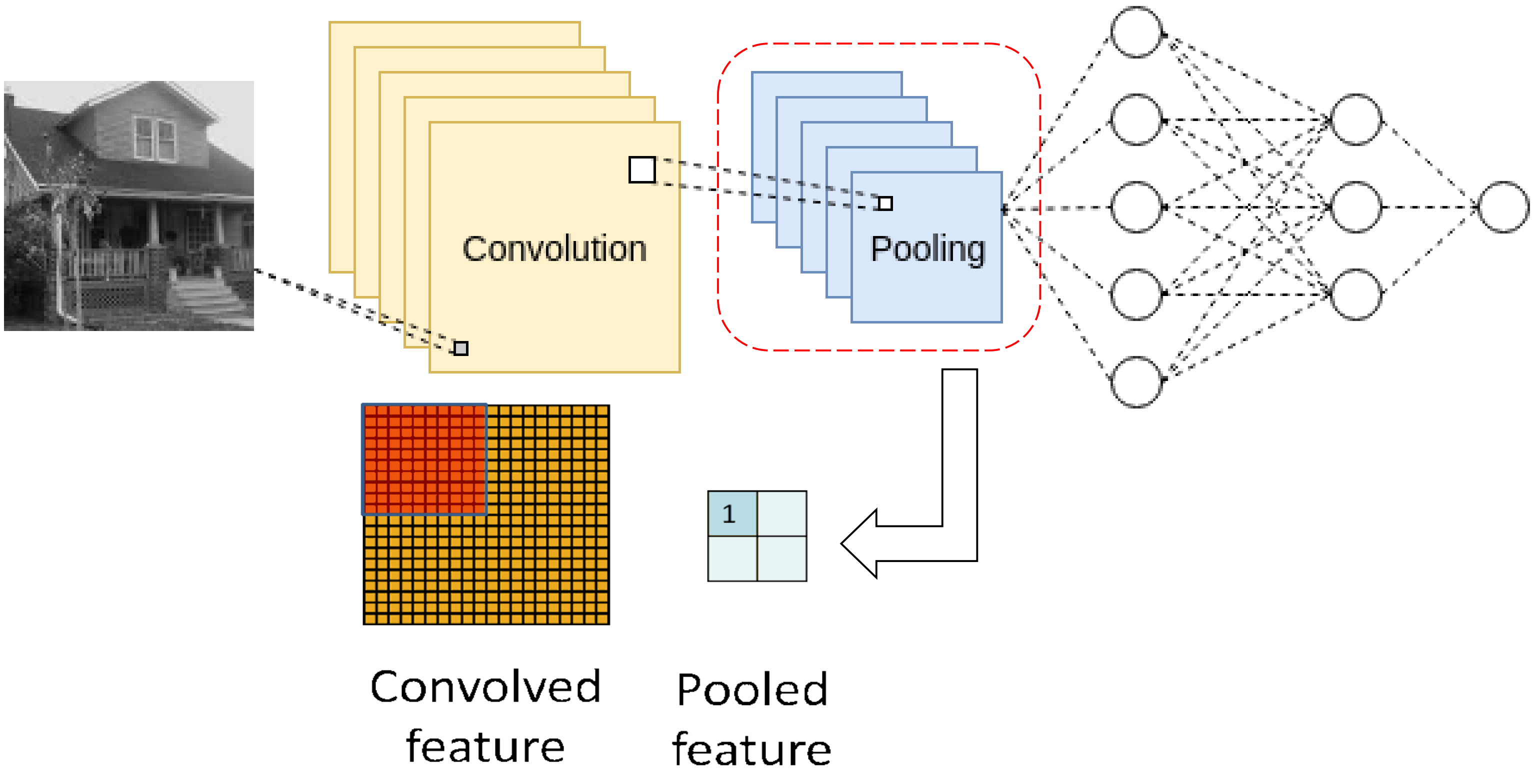
Convolutional Neural Network



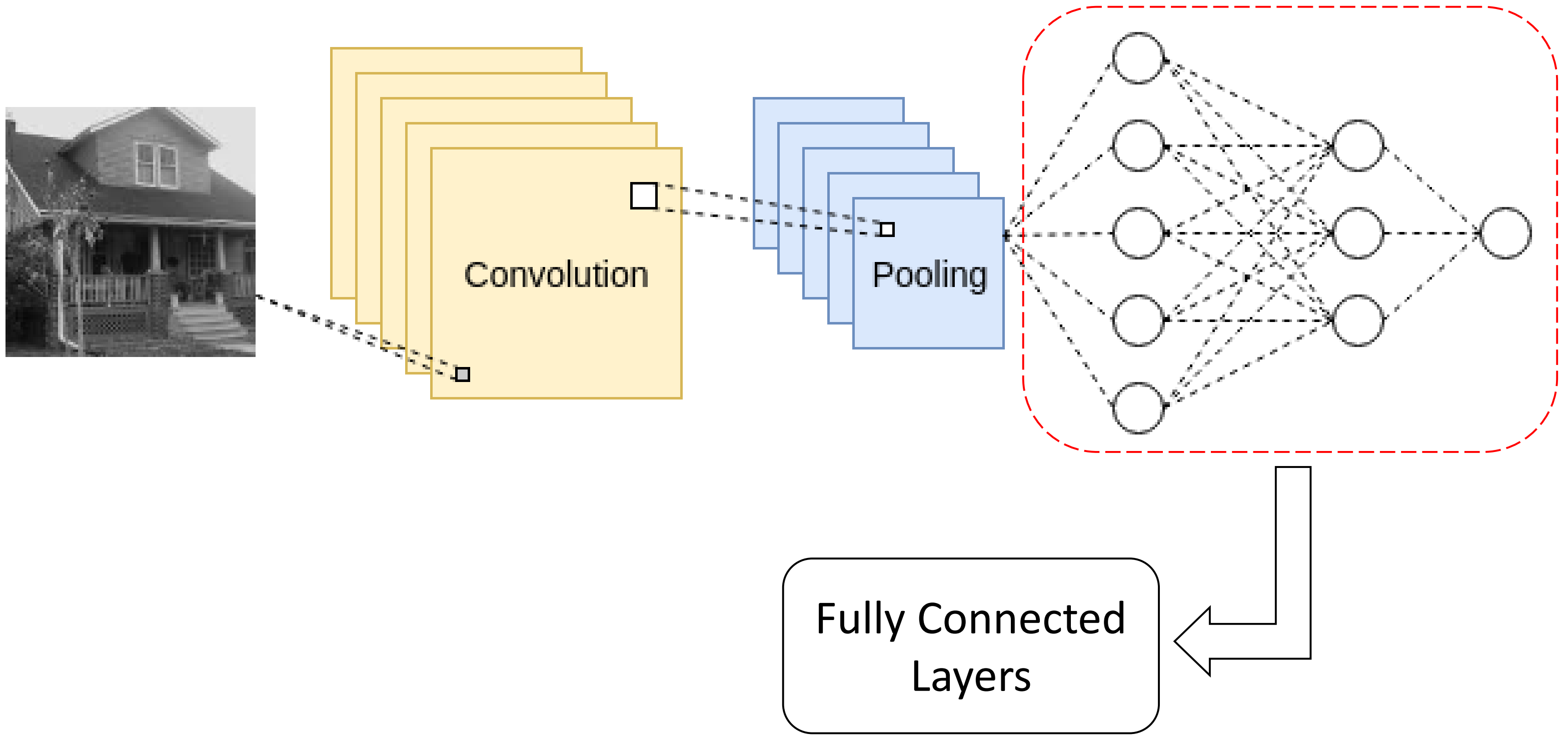
Convolutional Neural Network



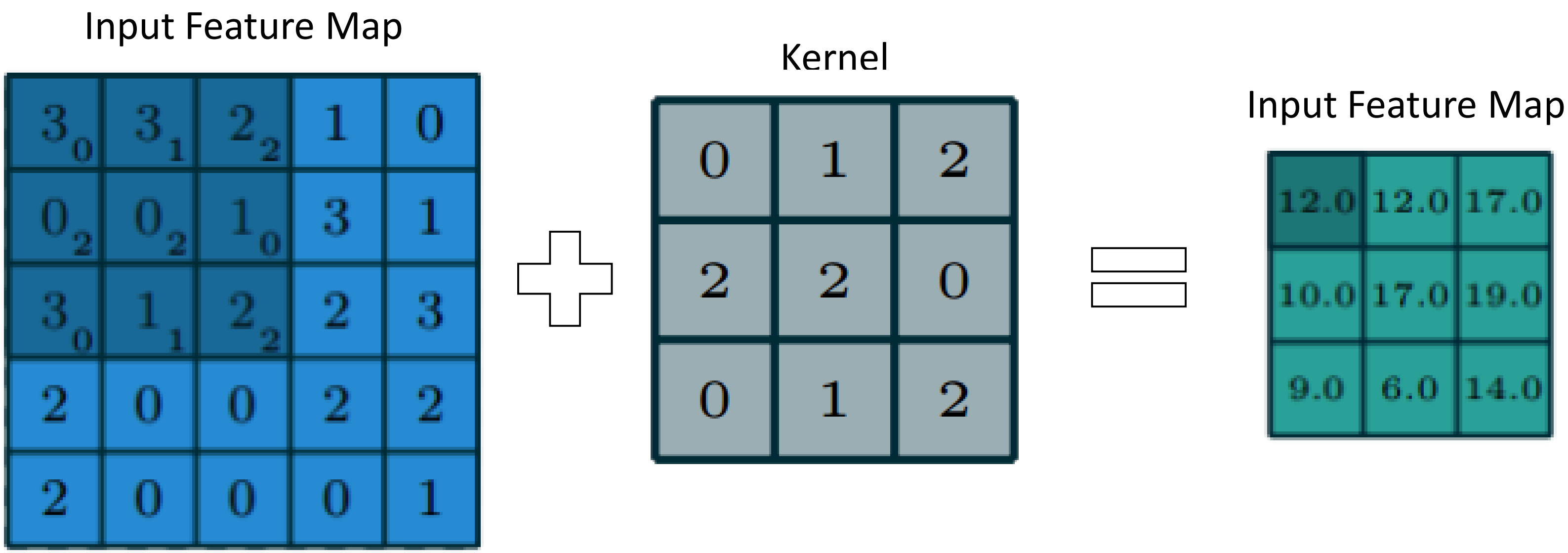
Convolutional Neural Network



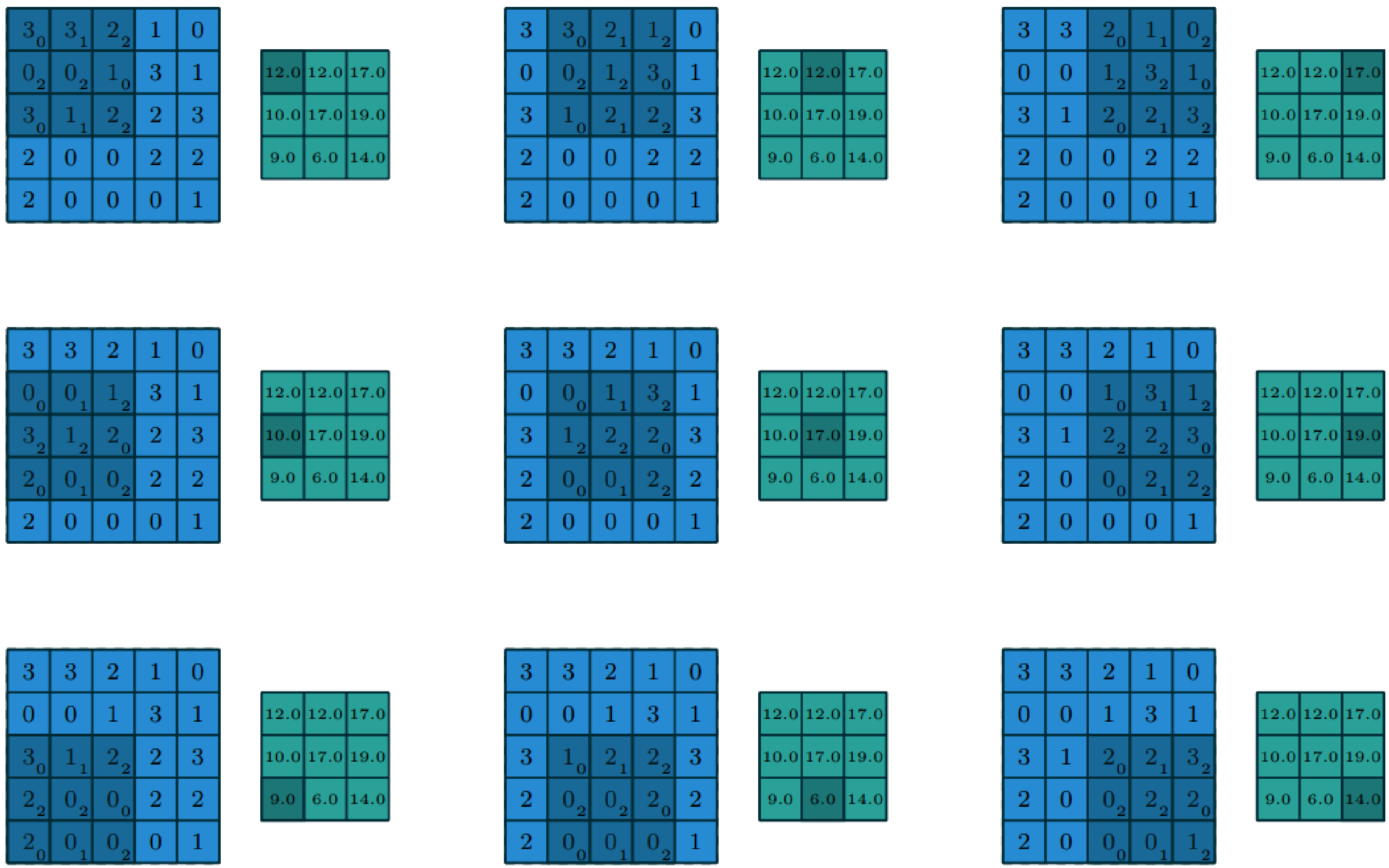
Convolutional Neural Network



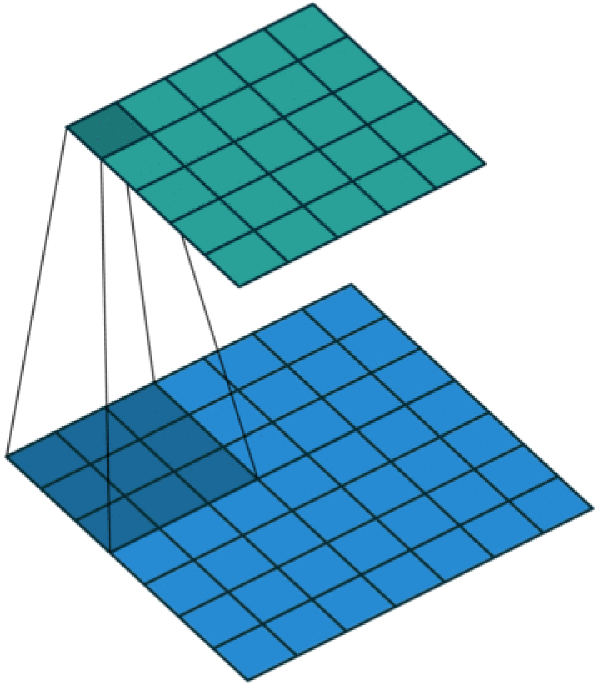
Discrete Convolution



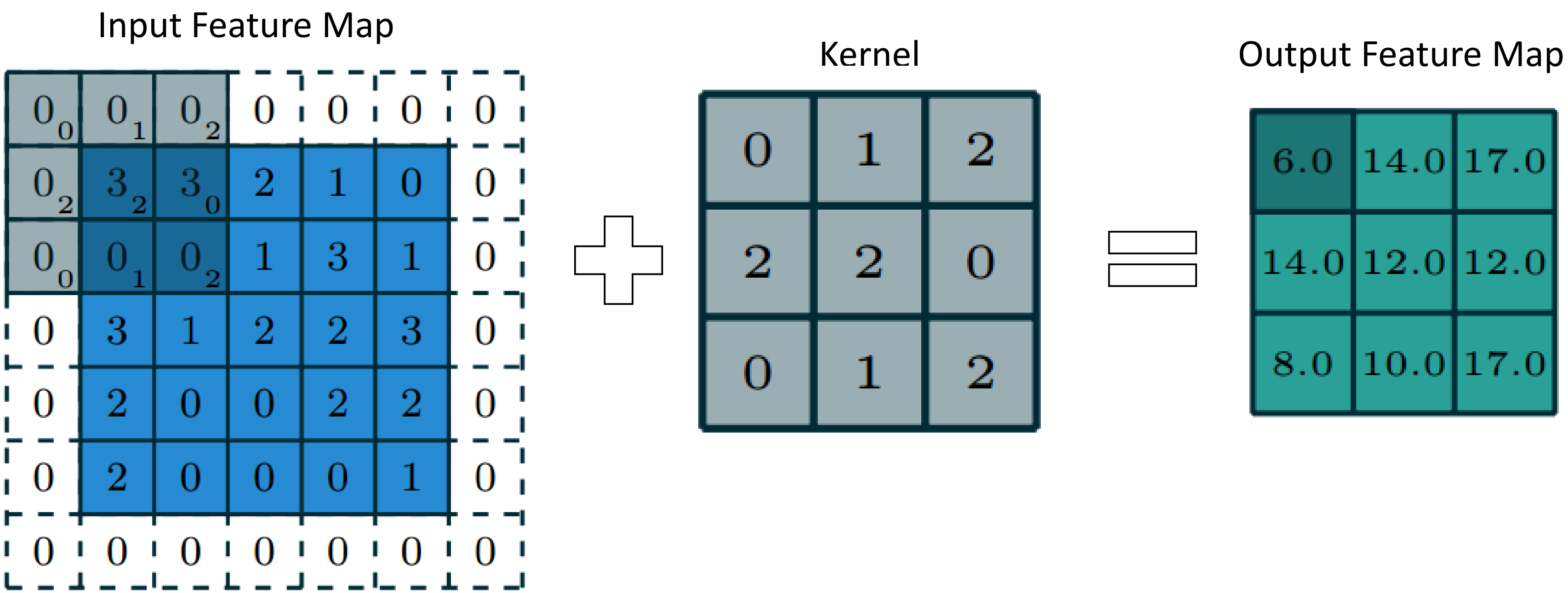
Discrete Convolution



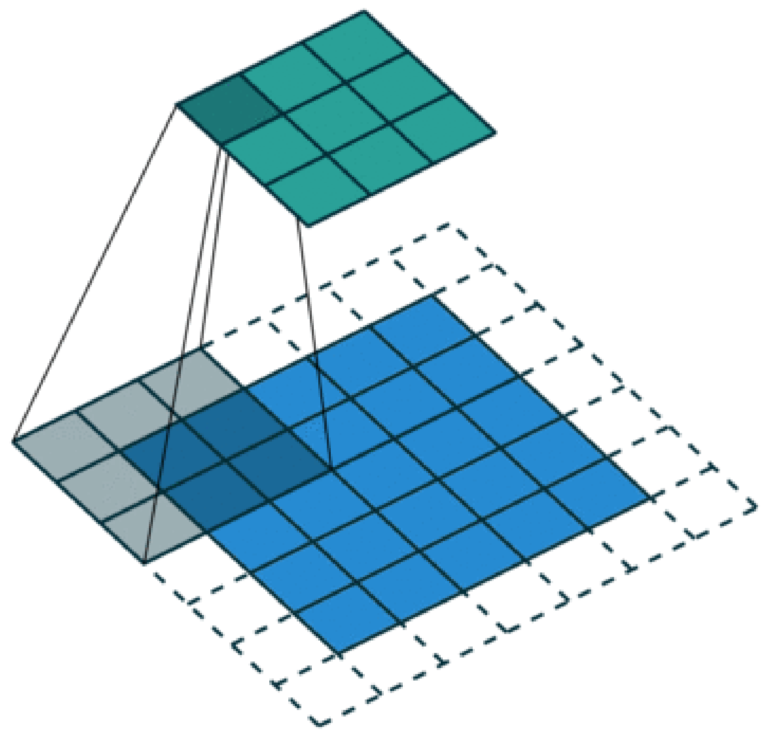
Discrete Convolution



Discrete Convolution with Padding



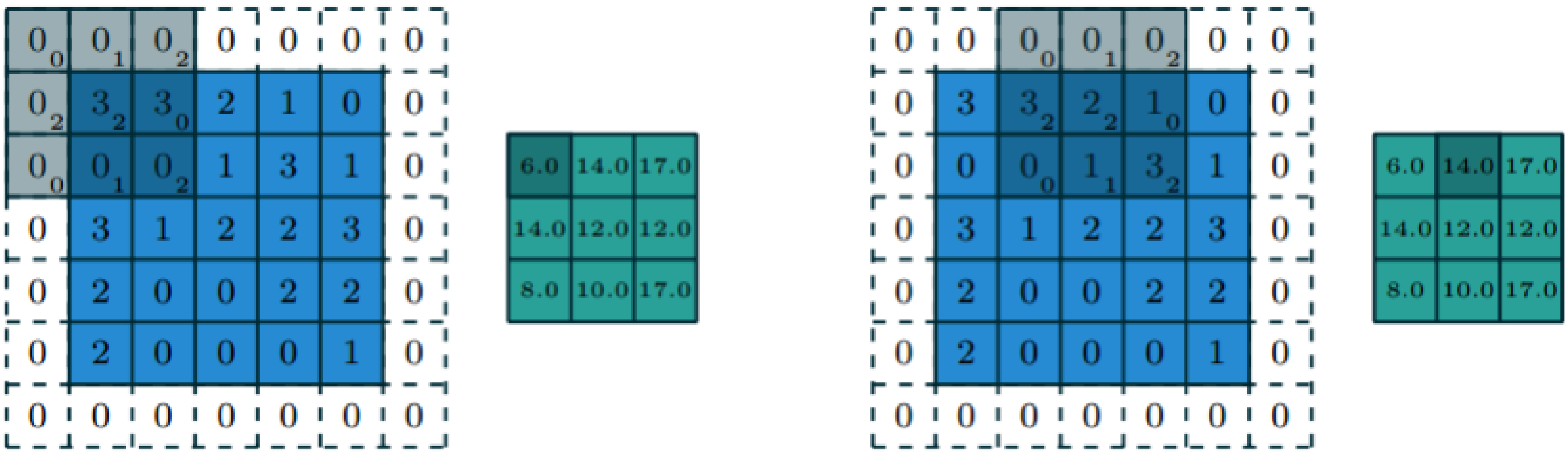
Discrete Convolution with Padding



Discrete Convolution

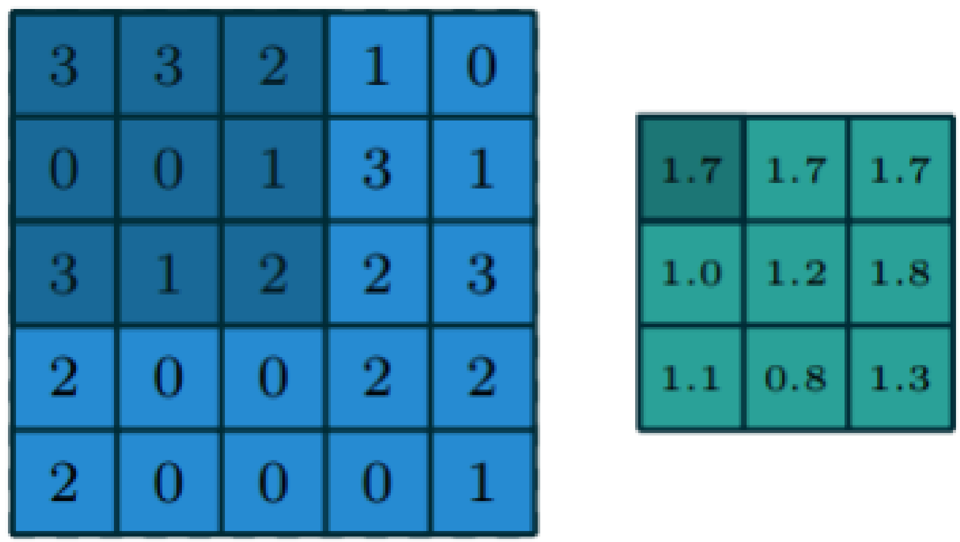
- The following properties affect the output size of a convolutional layer for an axis *j*:
  - i: input size
  - k: kernel size
  - s: stride (distance between two consecutive positions of the kernel)
  - p: zero padding (number of zeros concatenated at the beginning and at the end of an axis)

Discrete Convolution



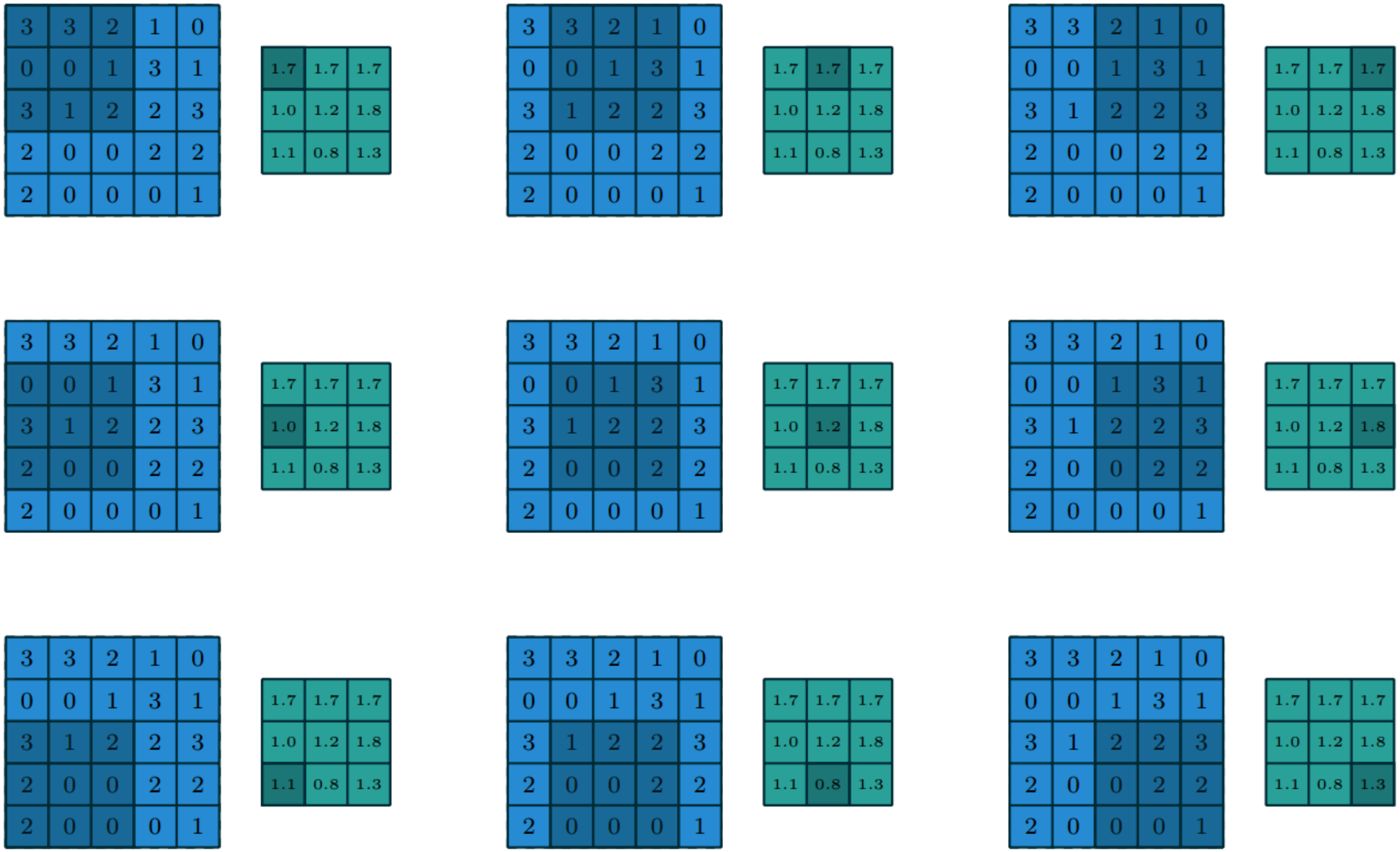
i = 5x5 k = 3x3 p = 1x1 s = 2x2

Average Pooling

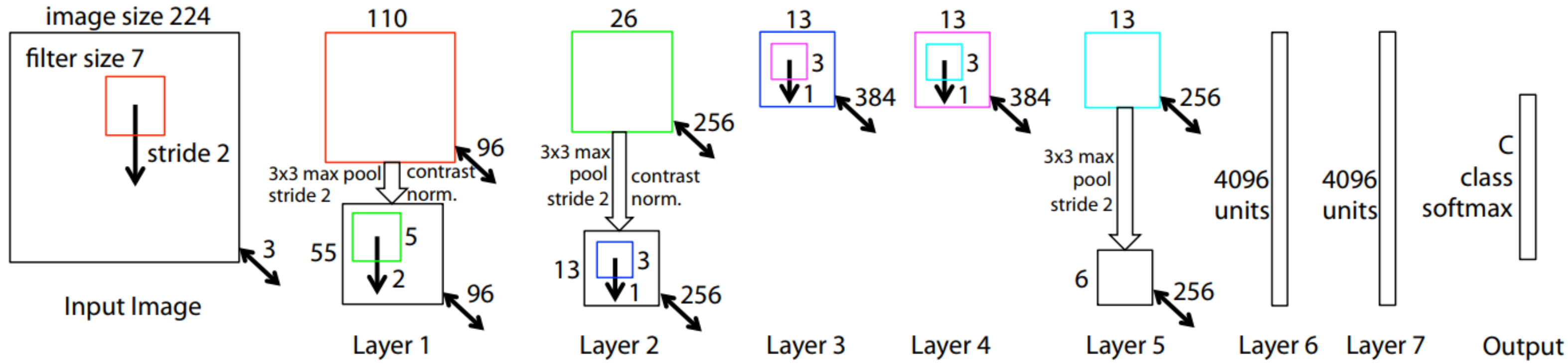


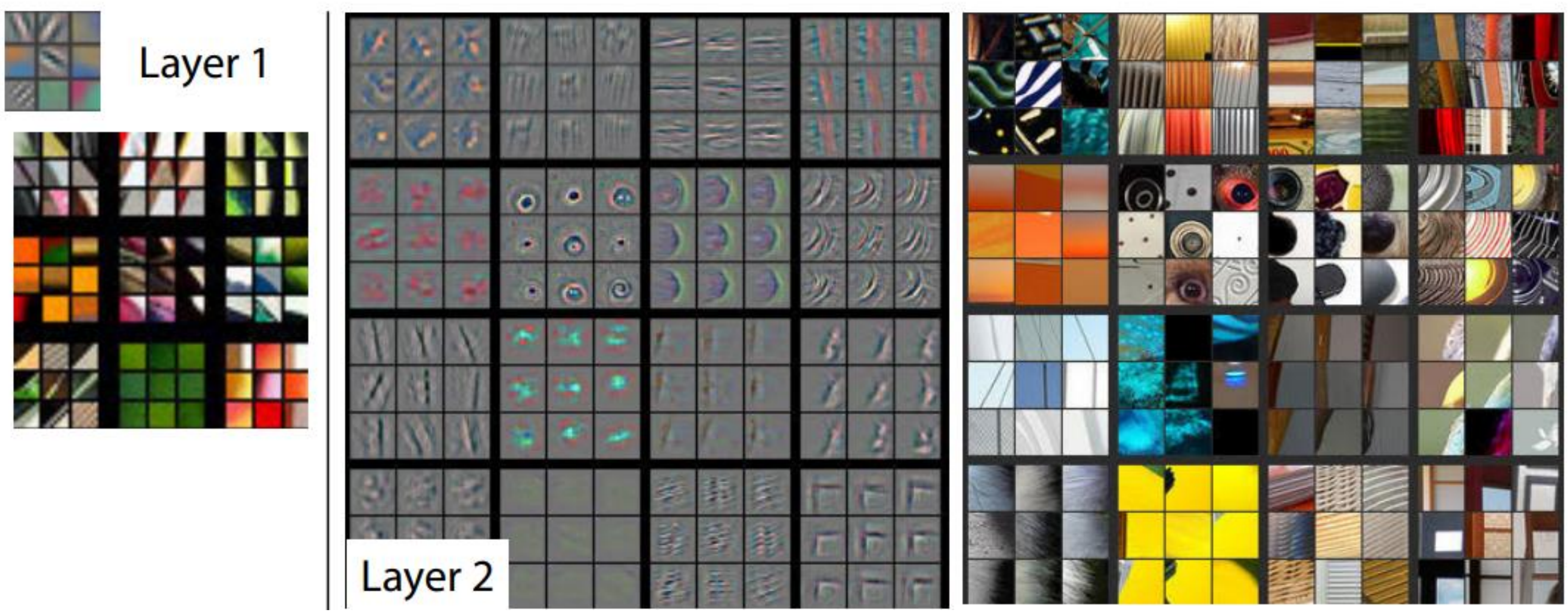
i = 5x5 k = 3x3 s = 1x1

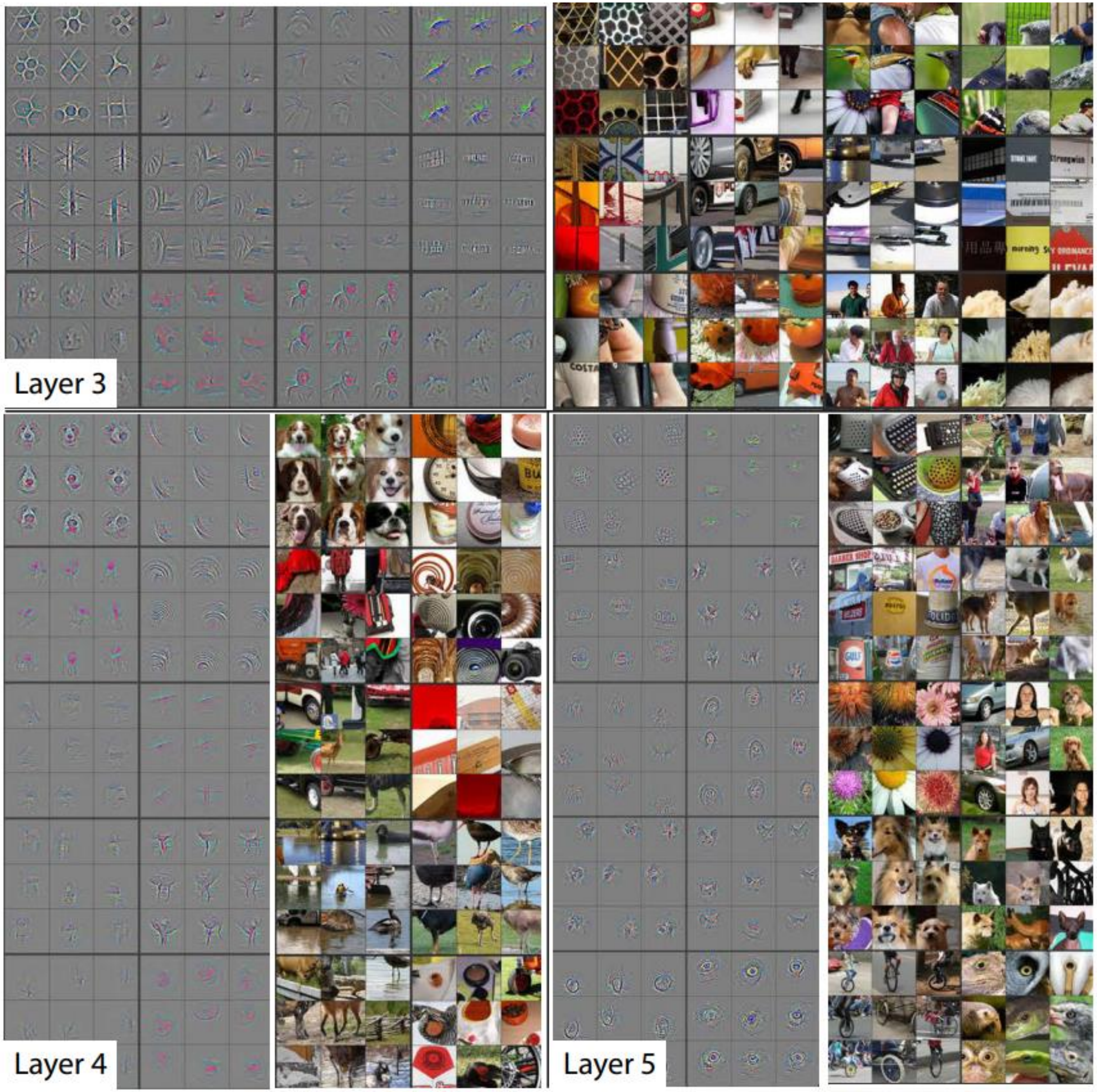
Average Pooling



Convolutional Neural Network






Layer 3
Layer 4
Layer 5

Relação Biológica

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

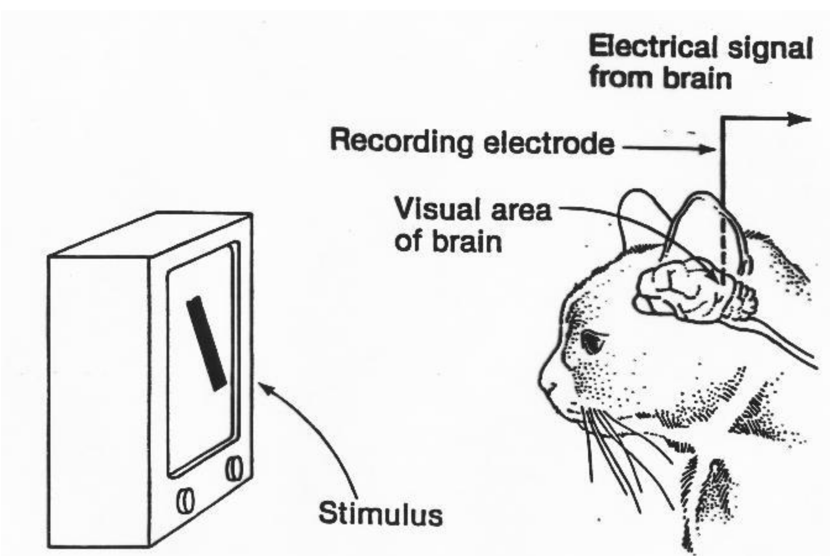
BY D. H. HUBEL* AND T. N. WIESEL*
From the Wilmer Institute, The Johns Hopkins Hospital and University, Baltimore, Maryland, U.S.A.

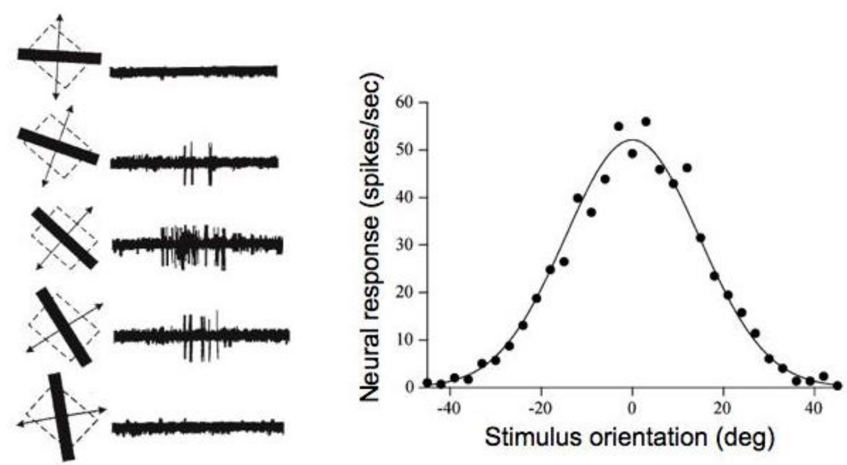
(*Received* 22 *April* 1959)

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

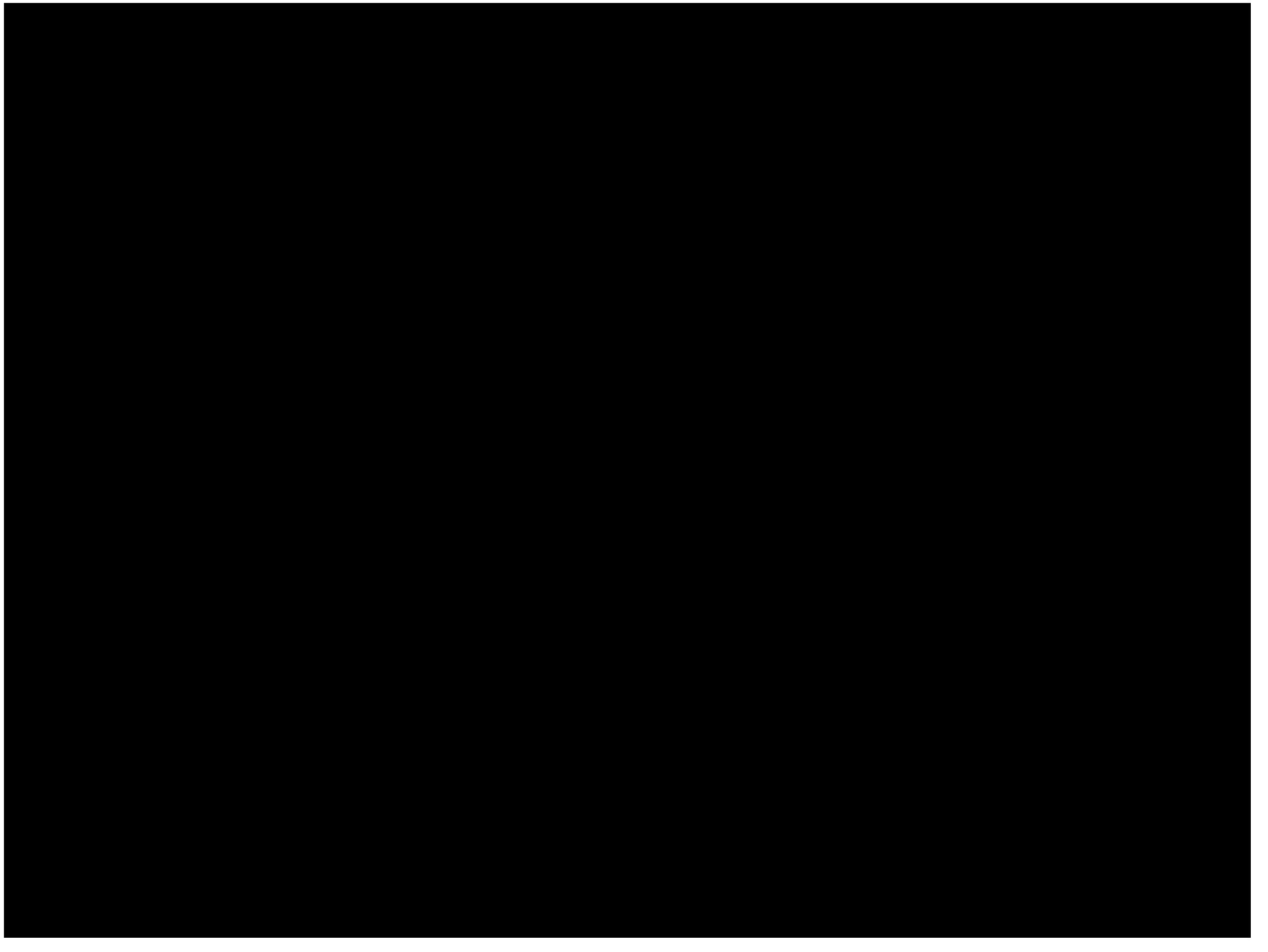
BY D. H. HUBEL AND T. N. WIESEL
From the Neurophysiolojy Laboratory, Department of Pharmacology Harvard Medical School, Boston, Massachusetts, U.S.A.

(*Received* 31 *July* 1961)

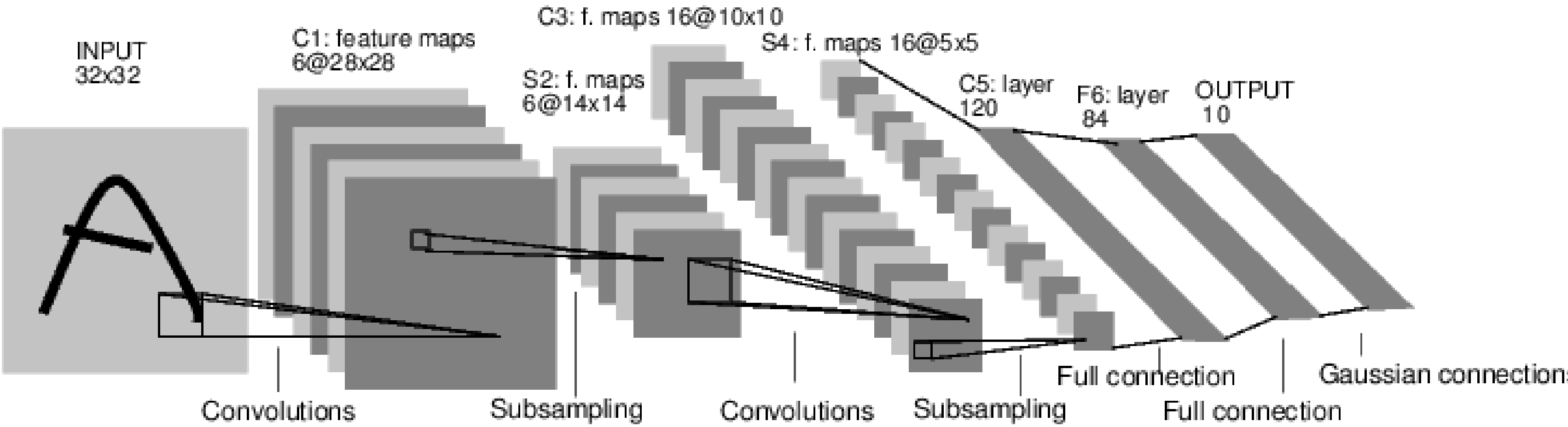


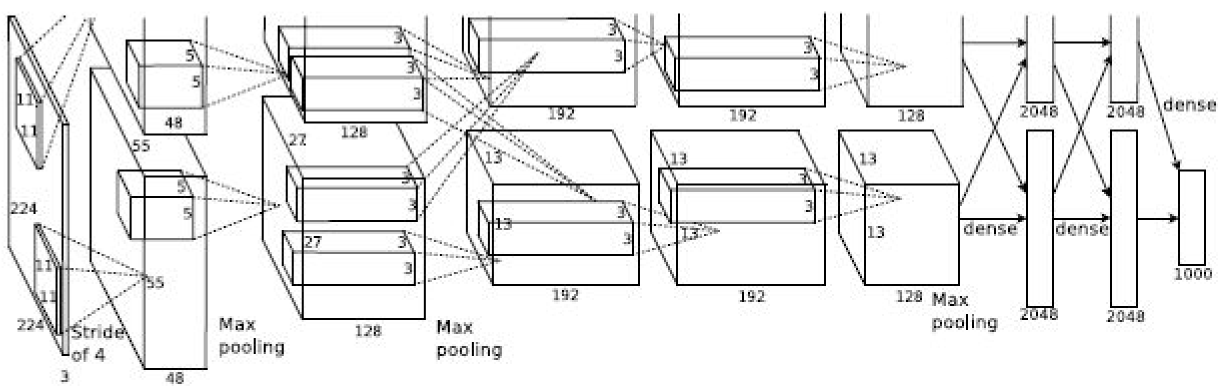


Experimento de Hubel & Wiesel



Lenet-5 e AlexNet (Model Zoo)





Frameworks

Caffe

APACHE Spark™

torch

K

neon

Lasagne

theano

DL4J DEEPLEARNING4J

TensorFlow

Chainer

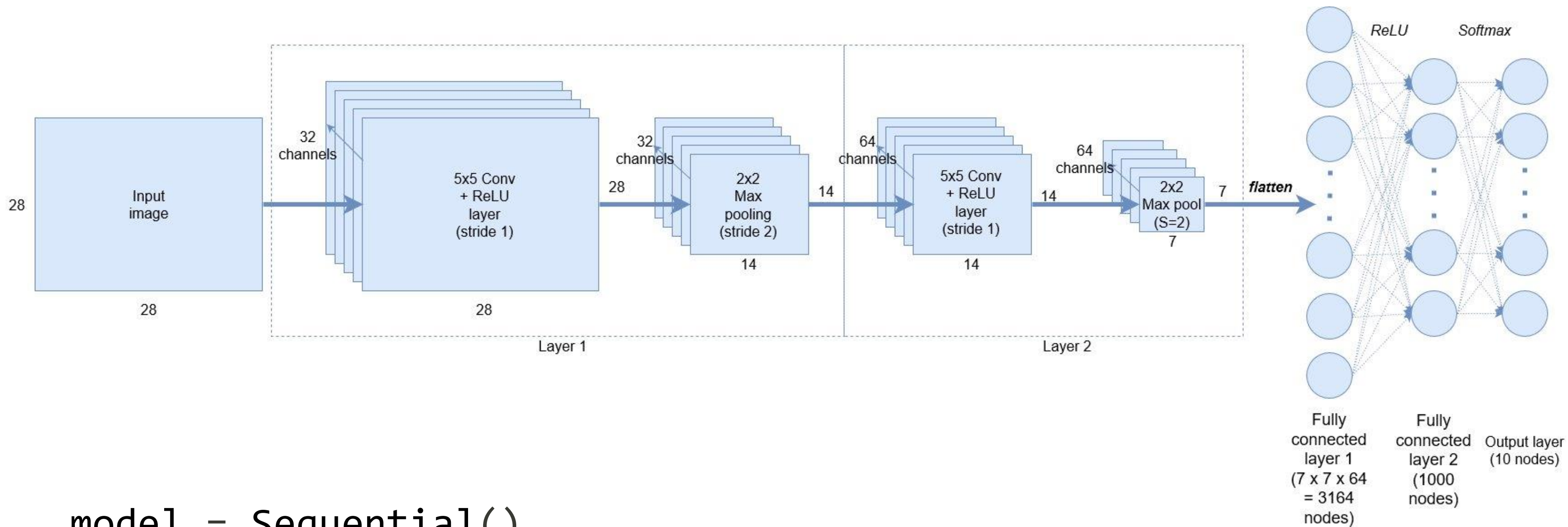
Microsoft CNTK

OpenDeep

MatConvNet

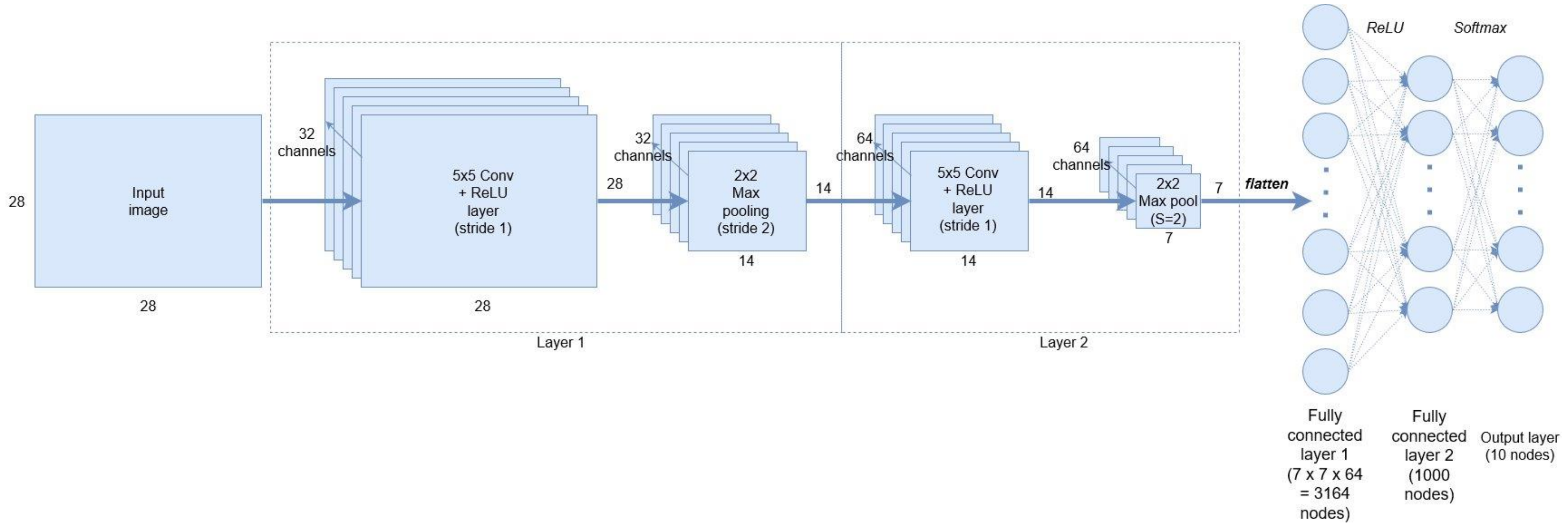
mxnet

Exemplo Keras + TensorFlow




```
model = Sequential()
model.add(Conv2D(32, kernel_size=(5, 5), strides=(1, 1),
activation='relu', input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Conv2D(64, (5, 5), activation='relu’))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(1000, activation='relu’))
model.add(Dense(num_classes, activation='softmax'))
```

Exemplo Keras + TensorFlow




```
model.compile(loss=keras.losses.categorical_crossentropy,
optimizer=keras.optimizers.SGD(lr=0.01), metrics=['accuracy'])

model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs,
verbose=1, validation_data=(x_test, y_test), callbacks=[history])

score = model.evaluate(x_test, y_test, verbose=0)
```

Referências

- Livro Redes Neurais S. Haykin (Capítulo 4);
- Backpropagation In Convolutional Neural Networks: http://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/
- Udacity Deep Learning: https://www.udacity.com/course/deep-learning--ud730
- ImageNet Classification with Deep Convolutional Neural Networks: https://www.nvidia.cn/content/tesla/pdf/machine-learning/imagenet-classification-with-deep-convolutional-nn.pdf
- VINCENT, Pascal et al. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion.