# $A^3$ - Aprendizagem Automática Avançada

## An Introduction

## to

## Convolutional Neural Networks

## with Tensorflow-Keras

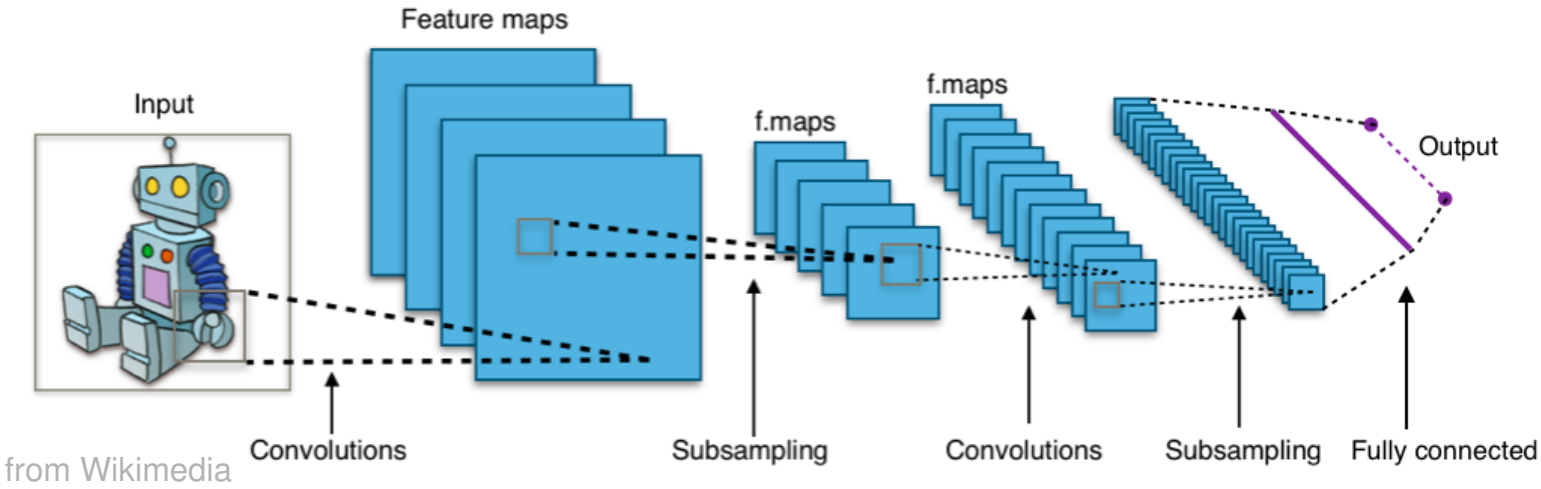
𝔾. Marques

# Convolutional Neural Networks

- Convolutional Neural Network (CNNs or ConvNets) have become very popular because they have achieved “superhuman” performances on complex visual tasks.
- The key building block is the convolutional layer, which applies local filtering operations to the input. The application of the same filter to an input (such as an image) results in a *feature map*. Specific filters can detect specific local features such as horizontal lines in an image, and combining a large number of filters can result in highly informative representations.
- CNNs were biologically inspired on studies of the visual cortex (Hubel and Wiesel experiments on cats in 1958 and monkeys 1968). Hubel and Wiesel showed that neurons have *receptive fields* which make them react to stimuli located only in their field of vision (composed of the surrounding neurons). Different receptive fields may overlap, and together they cover the whole field of view. Also, different neurons react to different stimuli (such as line orientations), and some neurons have large receptive fields that detect complex combinations of lower-level patterns.
- A major milestone was the 1998 paper by LeCun, Bottou, Bengio and Haffner, “Gradient-Based Learning Applied to Document Recognition”.
  This paper introduced two new building blocks for neural networks, the convolutional layer and the pooling layer (along with the popular LeNet-5 architecture).

# Convolutional Neural Networks

- A typical CNN architecture for image classification tasks:

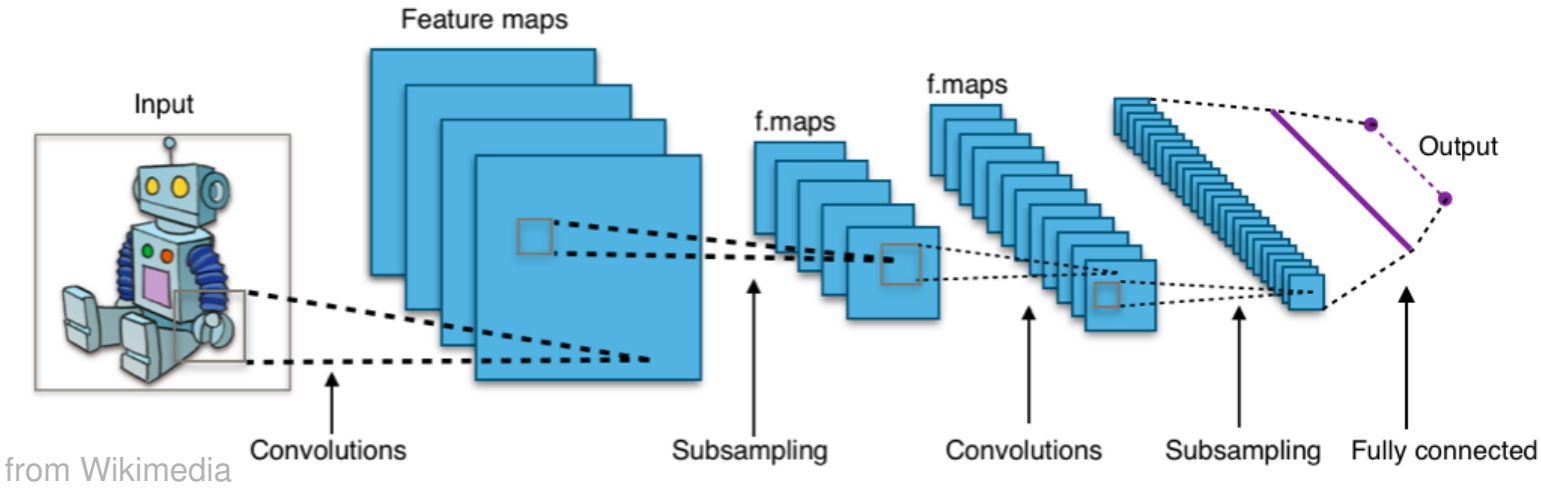



from Wikimedia

  - Neurons in the first layer are not connected to every pixel of the input image but only to pixels in their receptive fields.
  - In turn, neurons in the second convolutional layer are also connected to neurons within a small square (or rectangle) of the output of the sub-sampling layer.
  - Sub-sampling or **pooling** layer is an aggregation layer that reduces the size of the feature maps produce by convolutional layers, and therefore the computational and memory requirements to process the data.
  - The final part of the CNN is a stack fully connected layers (a classical MLP network).

# Convolutional Neural Networks

- A typical CNN architecture for image classification tasks:




from Wikimedia

  - Each feature map (one output of a convolutional layer) is obtained by convolving its input with a **kernel** (*i.e.* filter) whose weights are learned during the training phase.
  - Kernels detect local features in its input data. The first layer concentrates on low-level features, the next convolution layer assembles these features into higher-level ones, and so on.
  - The fact that all neurons share the same kernel (weights) in a feature map reduces drastically the number of parameters in a model (compared to a MLP network).
  - Once a kernel has learned to recognize a pattern in one location of the input data, it can recognize it in any other location.

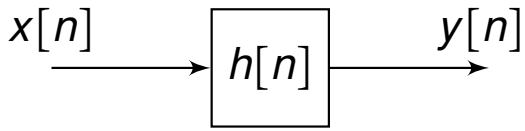
# Convolutional Neural Networks

## Convolution:

- Convolution is a mathematical operation that combines two functions and forms a third one. Formally, the convolution of two functions $g(t)$ and $h(t)$ is:

$$z(t) = g(t) * h(t) = \int_{-\infty}^{+\infty} g(s)h(t-s)ds = \int_{-\infty}^{+\infty} h(s)g(t-s)ds$$

- Convolution is a linear operation that enjoys several algebraic properties, such as associativity, distributivity, commutativity, and many others.
- Convolution is widely used in **digital signal processing**. Given a discrete systems with an impulse response, $h[n]$, the output of the system, $y[n]$, given the input signal $x[n]$, is the convolution between $x[n]$ and $h[n]$:




$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k] = \sum_{k=-\infty}^{+\infty} x[n-k]h[k]$$

Example:

Given a system, $h[n]$, with an impulse response $h[0] = 1$, $h[2] = -1$, and $h[n] = 0$ for $n \neq 0, 2$, and an input signal $x[n] = 2e^{-0.3n}$ for $n = 0, \ldots, 7$ and $x[n] = 0$ otherwise, find the output of the system.

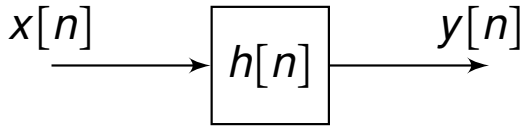
# Convolutional Neural Networks

## Convolution:

- Convolution is a mathematical operation that combines two functions and forms a third one. Formally, the convolution of two functions $g(t)$ and $h(t)$ is:

$$z(t) = g(t) * h(t) = \int_{-\infty}^{+\infty} g(s)h(t-s)ds = \int_{-\infty}^{+\infty} h(s)g(t-s)ds$$

- Convolution is a linear operation that enjoys several algebraic properties, such as associativity, distributivity, commutativity, and many others.
- Convolution is widely used in **digital signal processing**. Given a discrete systems with an impulse response, $h[n]$, the output of the system, $y[n]$, given the input signal $x[n]$, is the convolution between $x[n]$ and $h[n]$:




$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k] = \sum_{k=-\infty}^{+\infty} x[n-k]h[k]$$

Example:

$$y[n] = x[n] * h[n] = \sum_{k=n,n-2} h[n-k]x[k] \text{ for } n = 0,\ldots,9$$

Note that for $n < 0$ and $n > 9$, $y[n] = 0$.

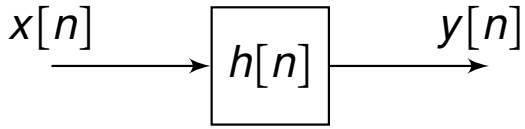
# Convolutional Neural Networks

## Convolution:

- Convolution is a mathematical operation that combines two functions and forms a third one. Formally, the convolution of two functions $g(t)$ and $h(t)$ is:

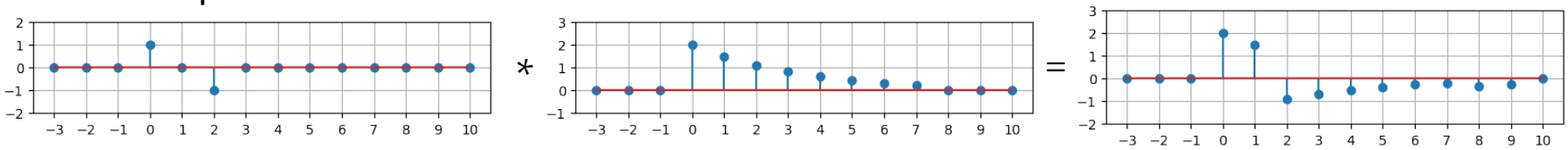
$$z(t) = g(t) * h(t) = \int_{-\infty}^{+\infty} g(s)h(t-s)ds = \int_{-\infty}^{+\infty} h(s)g(t-s)ds$$

- Convolution is a linear operation that enjoys several algebraic properties, such as associativity, distributivity, commutativity, and many others.
- Convolution is widely used in **digital signal processing**. Given a discrete systems with an impulse response, $h[n]$, the output of the system, $y[n]$, given the input signal $x[n]$, is the convolution between $x[n]$ and $h[n]$:




$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{+\infty} x[k]h[n-k] = \sum_{k=-\infty}^{+\infty} x[n-k]h[k]$$

Example:



# Convolutional Neural Networks

## Convolution:

- Convolution in **digital image processing**, involves filtering an image with a much smaller matrix, called the **convolutional kernel**.
- Formally, discrete convolution between two, 2D functions, $h[m,n]$ and $x[m,n]$ is:

  $$y[m,n] = h[m,n] * x[m,n] = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} x[i,j] * h[m-i, n-j]$$

  In image processing, $x[m,n]$ represents an image and $h[m,n]$ is the convolutional kernel (usually of dimensions $3\times3$ or $5\times5$).

  NOTE: Convolution involves flipping one of the signals and multiplying it with the other. Nevertheless, in image processing, many the kernels are symmetric and remain the same after flipping. In this case, convolution is equivalent to **cross-correlation**.
- There are two main types of kernels used in image processing: differential kernels, and smoothing kernels. Differential kernels are use to detect high-frequency contents in the image (*e.g.* edges), while smoothing kernels are used to low-pass filter the images (*e.g.* blurring, noise removal). Usually the sum of the coefficients in differential type kernels is zero, while for smoothing kernels is one.

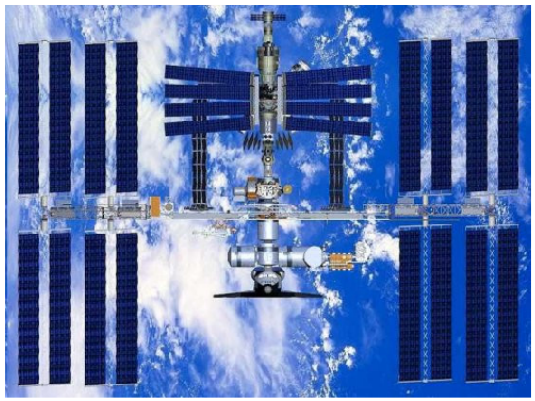
# Convolutional Neural Networks

## Convolution:

- Convolution in **digital image processing**, involves filtering an image with a much smaller matrix, called the **convolutional kernel**.
- Formally, discrete convolution between two, 2D functions, $h[m,n]$ and $x[m,n]$ is:

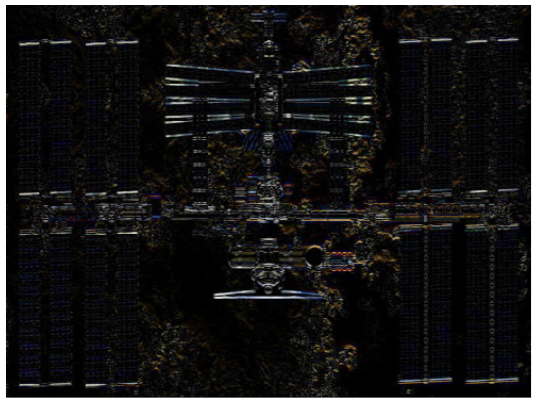
  $$y[m,n] = h[m,n] * x[m,n] = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} x[i,j] * h[m-i, n-j]$$

  In image processing, $x[m,n]$ represents an image and $h[m,n]$ is the convolutional kernel (usually of dimensions $3\times3$ or $5\times5$).
- Example:



$$* \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} =$$

(Sobel filter)



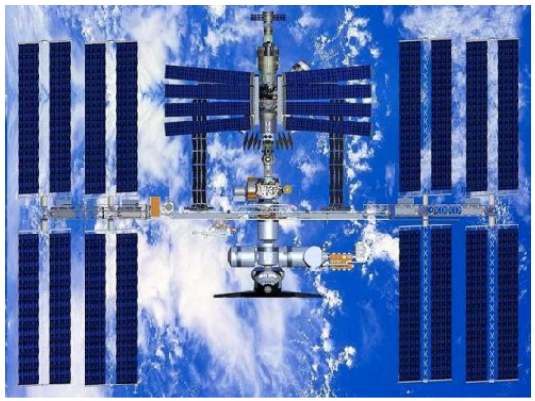
# Convolutional Neural Networks

## Convolution:

- Convolution in **digital image processing**, involves filtering an image with a much smaller matrix, called the **convolutional kernel**.
- Formally, discrete convolution between two, 2D functions, $h[m,n]$ and $x[m,n]$ is:

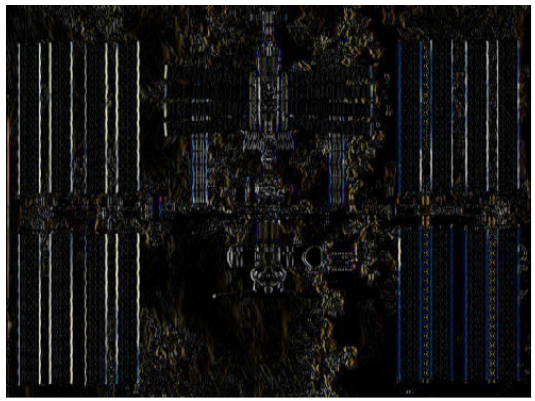
  $$y[m,n] = h[m,n] * x[m,n] = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} x[i,j] * h[m-i, n-j]$$

  In image processing, $x[m,n]$ represents an image and $h[m,n]$ is the convolutional kernel (usually of dimensions $3\times3$ or $5\times5$).
- Example:



$$* \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} =$$

(Sobel filter)



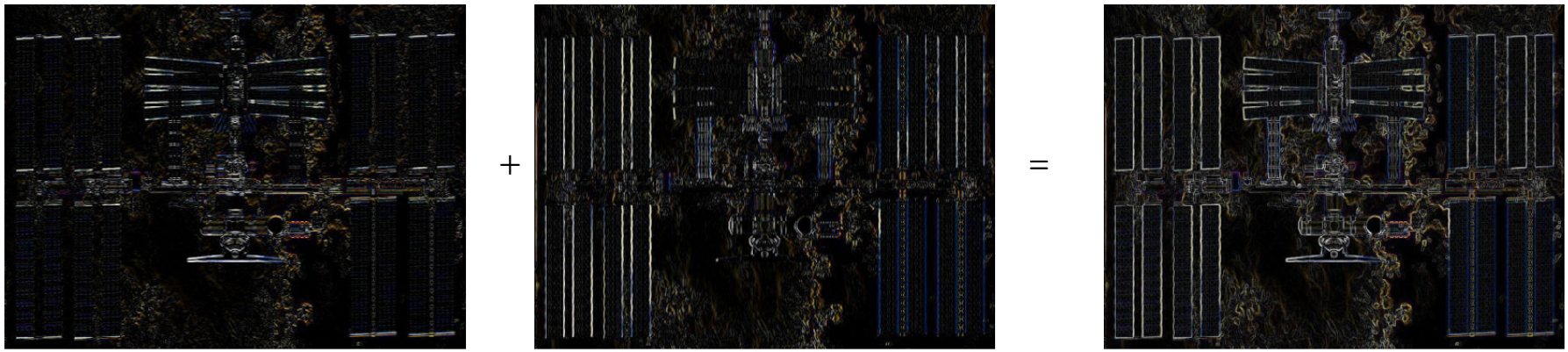
# Convolutional Neural Networks

## Convolution:

- Convolution in **digital image processing**, involves filtering an image with a much smaller matrix, called the **convolutional kernel**.
- Formally, discrete convolution between two, 2D functions, $h[m,n]$ and $x[m,n]$ is:

  $$y[m,n] = h[m,n] * x[m,n] = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} x[i,j] * h[m-i, n-j]$$

  In image processing, $x[m,n]$ represents an image and $h[m,n]$ is the convolutional kernel (usually of dimensions $3\times3$ or $5\times5$).
- Example:



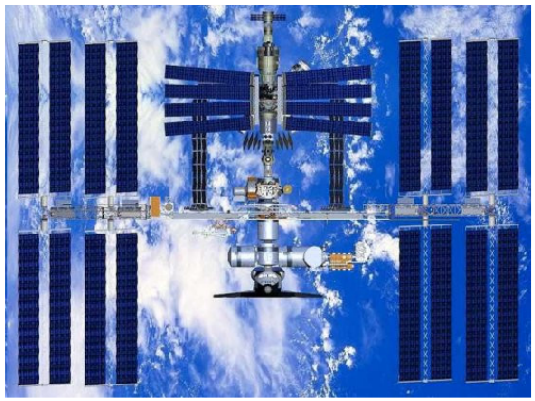
# Convolutional Neural Networks

## Convolution:

- Convolution in **digital image processing**, involves filtering an image with a much smaller matrix, called the **convolutional kernel**.
- Formally, discrete convolution between two, 2D functions, $h[m,n]$ and $x[m,n]$ is:

  $$y[m,n] = h[m,n] * x[m,n] = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} x[i,j] * h[m-i, n-j]$$

  In image processing, $x[m,n]$ represents an image and $h[m,n]$ is the convolutional kernel (usually of dimensions $3\times3$ or $5\times5$).
- Example:



$$* \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \frac{1}{9} =$$

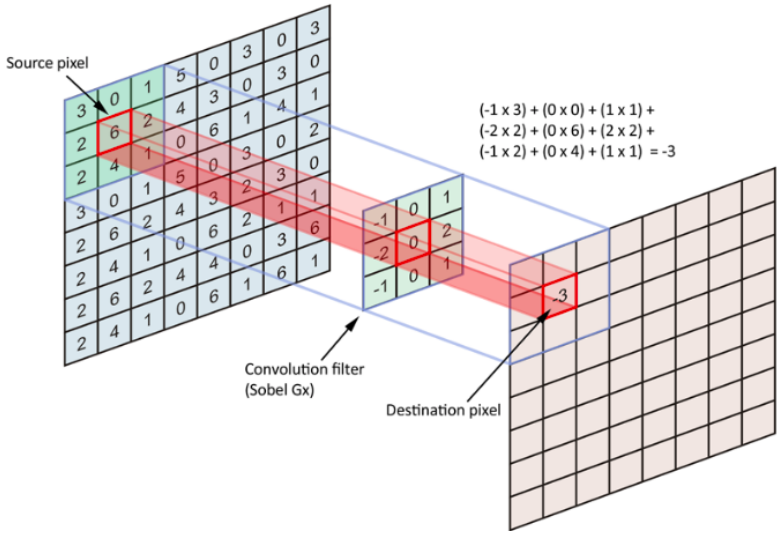
(Avg. filter)

# Convolutional Neural Networks

## Convolution in CNNs:

- The operations in a convolution layer of CNNs consists in multiply a kernel with the input of the layer, and applying a non-linearity to the result of the convolution.

  NOTE: This is not convolution but cross-correlation, but since the weights are trained, it does not matter.




by Arden Detat

- Formally, the output of a neuron in a CNN layer is the result of the convolution operation plus a bias term, passed through an activation function, $\varphi(\cdot)$:

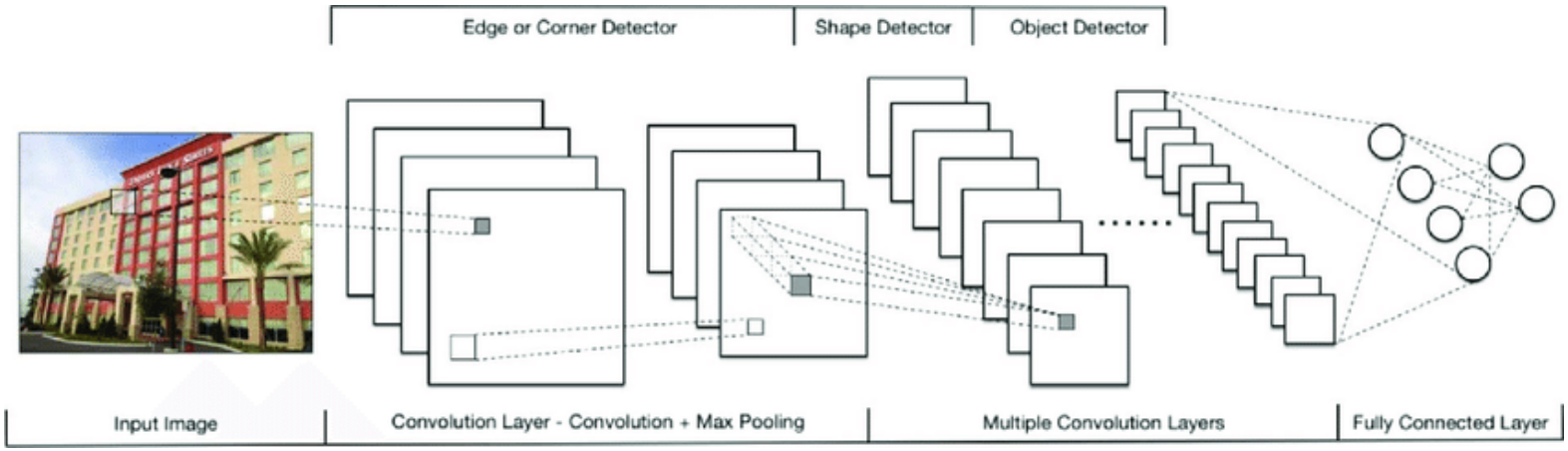
$$u[n,m] = \sum_{i,j=0}^{N} I[n+i, m+j] W[i,j] + b$$
$$v[n,m] = \varphi\big(u[n,m]\big)$$

- The filter weights are adapted during the training,
- CNNs are built by stacking several convolutional layers.

# Convolutional Neural Networks

## Convolution in CNNs:

- CNNs are built by stacking several convolutional layers.




by Yufeng Ma

- In the first layer, the kernels' input is the data (usually an image). The second layer receives as input the output of the first layer, and so on.
- The first layers extract low-level features of the data, while deeper layers extract more complex patterns resulting from combinations of low-level features from the first layers.

# Convolutional Neural Networks

## Convolution in CNNs:

- Example:

| | | | | |
|---|---|---|---|---|
| 5 | 8 | 0 | 4 | 8 |
| 3 | 9 | 2 | 9 | 2 |
| 4 | 9 | 2 | 3 | 6 |
| 5 | 4 | 8 | 2 | 0 |
| 5 | 8 | 0 | 0 | 8 |

$*$

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 4 | -1 |
| 0 | -1 | 0 |

$=$

| | | |
|---|---|---|
| | | |
| | | |
| | | |

# Convolutional Neural Networks

## Convolution in CNNs:

- Example:

| | | | | |
|---|---|---|---|---|
| 5 | 8 | 0 | 4 | 8 |
| 3 | 9 | 2 | 9 | 2 |
| 4 | 9 | 2 | 3 | 6 |
| 5 | 4 | 8 | 2 | 0 |
| 5 | 8 | 0 | 0 | 8 |

$*$

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 4 | -1 |
| 0 | -1 | 0 |

$=$

| | | |
|---|---|---|
| **14** | | |
| | | |
| | | |

# Convolutional Neural Networks

## Convolution in CNNs:

- Example:

| | | | | |
|---|---|---|---|---|
| 5 | 8 | 0 | 4 | 8 |
| 3 | 9 | 2 | 9 | 2 |
| 4 | 9 | 2 | 3 | 6 |
| 5 | 4 | 8 | 2 | 0 |
| 5 | 8 | 0 | 0 | 8 |

$*$

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 4 | -1 |
| 0 | -1 | 0 |

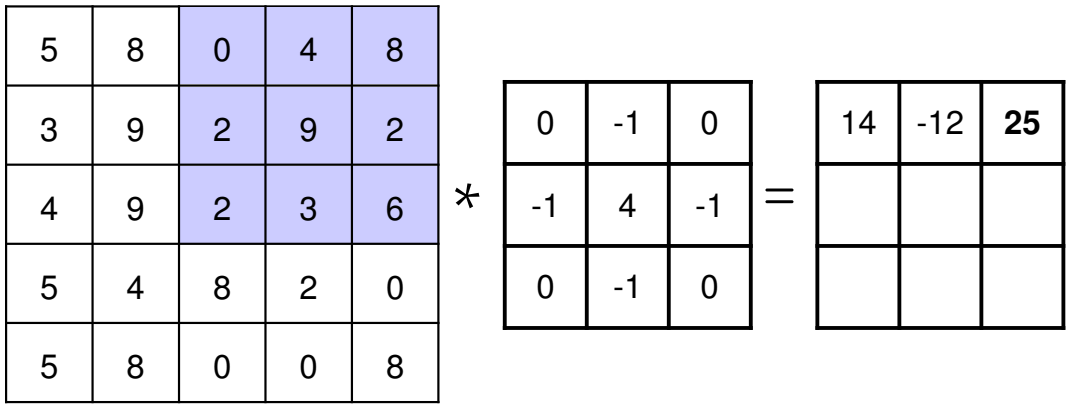
$=$

| | | |
|---|---|---|
| 14 | **-12** | |
| | | |
| | | |

# Convolutional Neural Networks

## Convolution in CNNs:

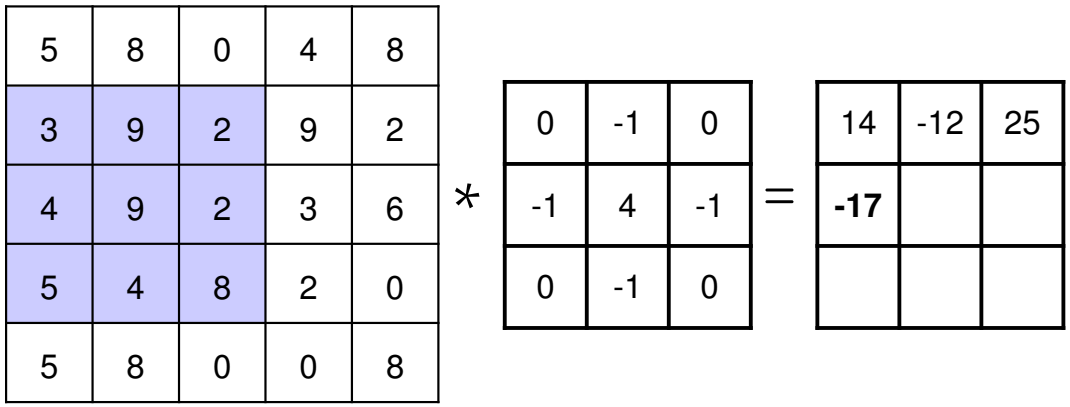
- Example:



# Convolutional Neural Networks

## Convolution in CNNs:

- Example:



# Convolutional Neural Networks

## Convolution in CNNs:

- Example:

| | | | | |
|---|---|---|---|---|
| 5 | 8 | 0 | 4 | 8 |
| 3 | 9 | 2 | 9 | 2 |
| 4 | 9 | 2 | 3 | 6 |
| 5 | 4 | 8 | 2 | 0 |
| 5 | 8 | 0 | 0 | 8 |

$*$

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 4 | -1 |
| 0 | -1 | 0 |

$=$

| | | |
|---|---|---|
| 14 | -12 | 25 |
| -17 | **-14** | **7** |
| | | |

# Convolutional Neural Networks

## Convolution in CNNs:

- Example:

| | | | | |
|---|---|---|---|---|
| 5 | 8 | 0 | 4 | 8 |
| 3 | 9 | 2 | 9 | 2 |
| 4 | 9 | 2 | 3 | 6 |
| 5 | 4 | 8 | 2 | 0 |
| 5 | 8 | 0 | 0 | 8 |

$*$

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 4 | -1 |
| 0 | -1 | 0 |

$=$

| | | |
|---|---|---|
| 14 | -12 | 25 |
| -17 | -14 | 7 |
| **-14** | **24** | **-3** |

# Convolutional Neural Networks

## Convolution in CNNs:

- Example:

| | | | | |
|---|---|---|---|---|
| 5 | 8 | 0 | 4 | 8 |
| 3 | 9 | 2 | 9 | 2 |
| 4 | 9 | 2 | 3 | 6 |
| 5 | 4 | 8 | 2 | 0 |
| 5 | 8 | 0 | 0 | 8 |

$*$

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 4 | -1 |
| 0 | -1 | 0 |

$=$

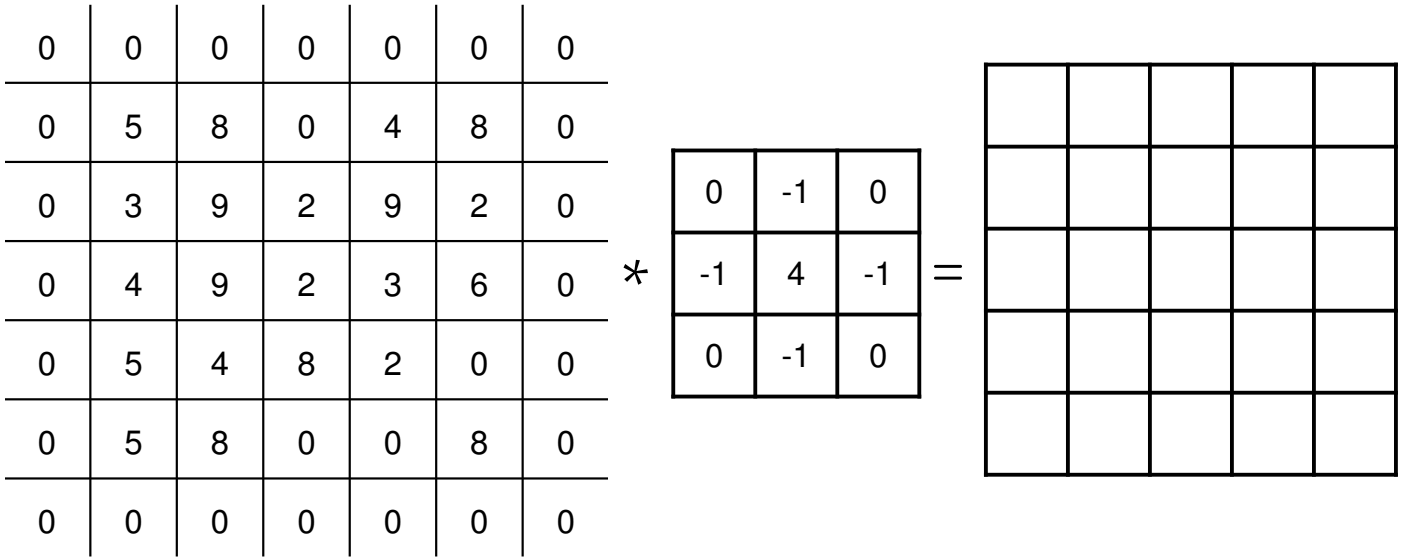
| | | |
|---|---|---|
| 14 | -12 | 25 |
| -17 | -14 | 7 |
| -14 | 24 | -3 |

- This operation results in an output that is smaller than the input.
  This is also known as *valid convolution*.
- Consider an image of $I_h \times I_w$ pixels and a filter (kernel) of dimensions $f_h \times f_w$.
  The output image will have dimensions: $(I_h - f_h + 1) \times (I_w - f_w + 1)$
- In order to obtain an output with the same dimensions as the input,
  one can use **zero padding**.

# Convolutional Neural Networks

## Convolution in CNNs:

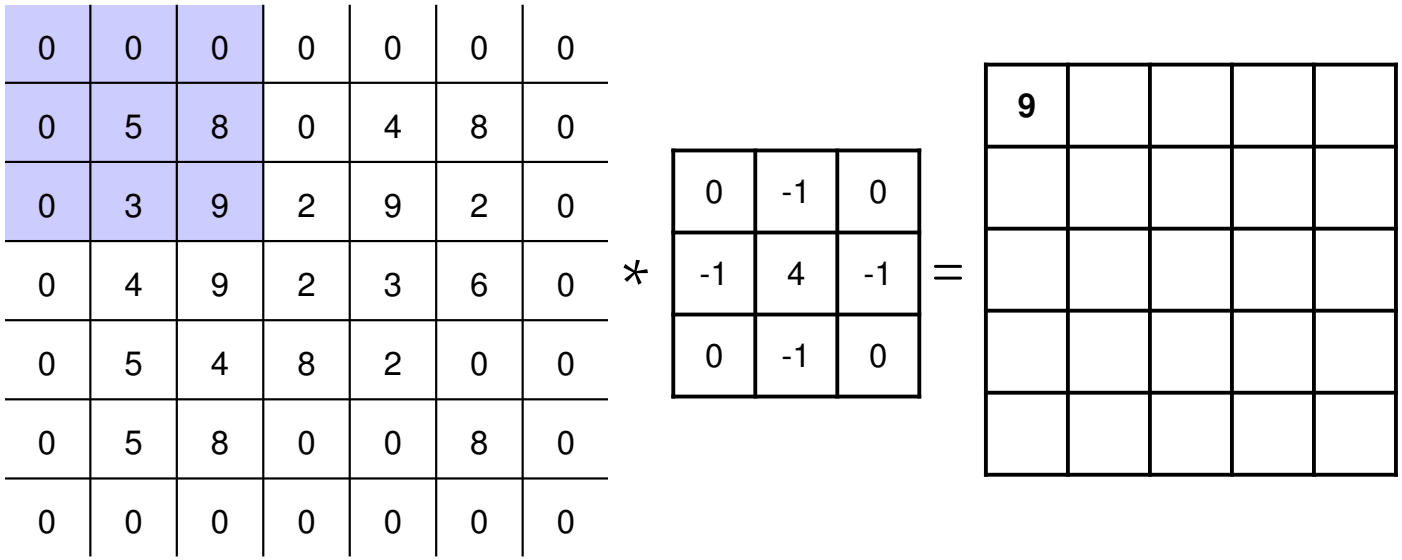
- Example (with zero-padding):



# Convolutional Neural Networks

## Convolution in CNNs:

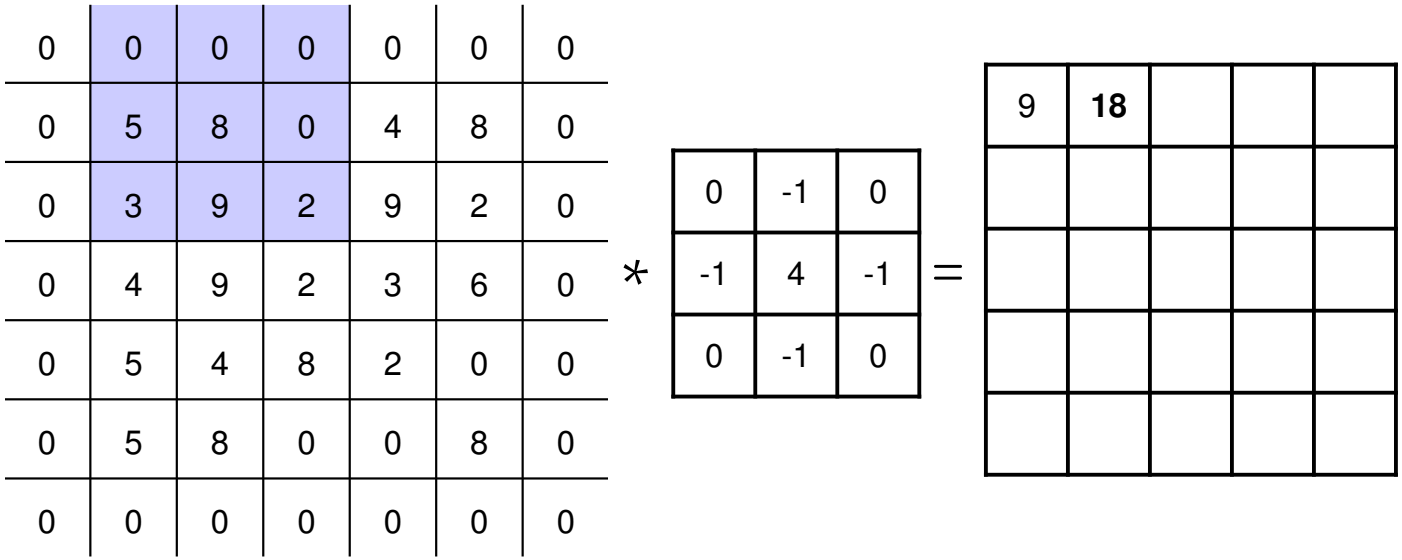
- Example (with zero-padding):



# Convolutional Neural Networks

## Convolution in CNNs:

- Example (with zero-padding):



# Convolutional Neural Networks

## Convolution in CNNs:

- Example (with zero-padding):

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 5 | 8 | 0 | 4 | 8 | 0 |
| 0 | 3 | 9 | 2 | 9 | 2 | 0 |
| 0 | 4 | 9 | 2 | 3 | 6 | 0 |
| 0 | 5 | 4 | 8 | 2 | 0 | 0 |
| 0 | 5 | 8 | 0 | 0 | 8 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$*$

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 4 | -1 |
| 0 | -1 | 0 |

$=$

| | | | | |
|---|---|---|---|---|
| 9 | 18 | **-14** | **-1** | **26** |
| **-6** | | | | |
| | | | | |
| | | | | |
| | | | | |

# Convolutional Neural Networks

## Convolution in CNNs:

- Example (with zero-padding):

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 5 | 8 | 0 | 4 | 8 | 0 |
| 0 | 3 | 9 | 2 | 9 | 2 | 0 |
| 0 | 4 | 9 | 2 | 3 | 6 | 0 |
| 0 | 5 | 4 | 8 | 2 | 0 | 0 |
| 0 | 5 | 8 | 0 | 0 | 8 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$*$

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 4 | -1 |
| 0 | -1 | 0 |

$=$

| | | | | |
|---|---|---|---|---|
| 9 | 18 | -14 | -1 | 26 |
| -6 | **14** | **-12** | **25** | **-15** |
| **-1** | **17** | **-14** | **-7** | **19** |
| **7** | **-14** | **24** | **-3** | **-16** |
| **7** | **23** | **-16** | **-10** | **32** |

# Convolutional Neural Networks

## Convolution in CNNs:

- Example (with zero-padding):

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 5 | 8 | 0 | 4 | 8 | 0 |
| 0 | 3 | 9 | 2 | 9 | 2 | 0 |
| 0 | 4 | 9 | 2 | 3 | 6 | 0 |
| 0 | 5 | 4 | 8 | 2 | 0 | 0 |
| 0 | 5 | 8 | 0 | 0 | 8 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$*$

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 4 | -1 |
| 0 | -1 | 0 |

$=$

| | | | | |
|---|---|---|---|---|
| 9 | 18 | -14 | -1 | 26 |
| -6 | 14 | -12 | 25 | -15 |
| -1 | 17 | -14 | -7 | 19 |
| 7 | -14 | 24 | -3 | -16 |
| 7 | 23 | -16 | -10 | 32 |

- This operation results in an output with the same dimensions as the input.
- Another type of padding commonly used in image processing is to copy the values of the adjacent pixels, instead of using zeros.

# Convolutional Neural Networks

## Convolution in CNNs:

- Strides:
  - The stride is the distance the kernel progresses between two consecutive convolutions.
  - Large strides reduce significantly the size of the layer output (and subsequent processing in higher layers).
  - A stride can be defined by two values, the height $s_h$ and the width $s_w$.
- The previous example with $s_h = s_w = 2$:

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 5 | 8 | 0 | 4 | 8 | 0 |
| 0 | 3 | 9 | 2 | 9 | 2 | 0 |
| 0 | 4 | 9 | 2 | 3 | 6 | 0 |
| 0 | 5 | 4 | 8 | 2 | 0 | 0 |
| 0 | 5 | 8 | 0 | 0 | 8 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$*$

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 4 | -1 |
| 0 | -1 | 0 |

$=$

| | | |
|---|---|---|
| | | |
| | | |
| | | |

# Convolutional Neural Networks

## Convolution in CNNs:

- Strides:
  - The stride is the distance the kernel progresses between two consecutive convolutions.
  - Large strides reduce significantly the size of the layer output (and subsequent processing in higher layers).
  - A stride can be defined by two values, the height $s_h$ and the width $s_w$.
- The previous example with $s_h = s_w = 2$:

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 5 | 8 | 0 | 4 | 8 | 0 |
| 0 | 3 | 9 | 2 | 9 | 2 | 0 |
| 0 | 4 | 9 | 2 | 3 | 6 | 0 |
| 0 | 5 | 4 | 8 | 2 | 0 | 0 |
| 0 | 5 | 8 | 0 | 0 | 8 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$*$

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 4 | -1 |
| 0 | -1 | 0 |

$=$

| | | |
|---|---|---|
| **9** | | |
| | | |
| | | |

# Convolutional Neural Networks

## Convolution in CNNs:

- Strides:
  - The stride is the distance the kernel progresses between two consecutive convolutions.
  - Large strides reduce significantly the size of the layer output (and subsequent processing in higher layers).
  - A stride can be defined by two values, the height $s_h$ and the width $s_w$.
- The previous example with $s_h = s_w = 2$:

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 5 | 8 | 0 | 4 | 8 | 0 |
| 0 | 3 | 9 | 2 | 9 | 2 | 0 |
| 0 | 4 | 9 | 2 | 3 | 6 | 0 |
| 0 | 5 | 4 | 8 | 2 | 0 | 0 |
| 0 | 5 | 8 | 0 | 0 | 8 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$*$

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 4 | -1 |
| 0 | -1 | 0 |

$=$

| | | |
|---|---|---|
| 9 | **-14** | |
| | | |
| | | |

# Convolutional Neural Networks

## Convolution in CNNs:

- Strides:
  - The stride is the distance the kernel progresses between two consecutive convolutions.
  - Large strides reduce significantly the size of the layer output (and subsequent processing in higher layers).
  - A stride can be defined by two values, the height $s_h$ and the width $s_w$.
- The previous example with $s_h = s_w = 2$:

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 5 | 8 | 0 | 4 | 8 | 0 |
| 0 | 3 | 9 | 2 | 9 | 2 | 0 |
| 0 | 4 | 9 | 2 | 3 | 6 | 0 |
| 0 | 5 | 4 | 8 | 2 | 0 | 0 |
| 0 | 5 | 8 | 0 | 0 | 8 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$*$

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 4 | -1 |
| 0 | -1 | 0 |

$=$

| | | |
|---|---|---|
| 9 | -14 | **26** |
| | | |
| | | |

# Convolutional Neural Networks

## Convolution in CNNs:

- Strides:
  - The stride is the distance the kernel progresses between two consecutive convolutions.
  - Large strides reduce significantly the size of the layer output (and subsequent processing in higher layers).
  - A stride can be defined by two values, the height $s_h$ and the width $s_w$.
- The previous example with $s_h = s_w = 2$:

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 5 | 8 | 0 | 4 | 8 | 0 |
| 0 | 3 | 9 | 2 | 9 | 2 | 0 |
| 0 | 4 | 9 | 2 | 3 | 6 | 0 |
| 0 | 5 | 4 | 8 | 2 | 0 | 0 |
| 0 | 5 | 8 | 0 | 0 | 8 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$*$

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 4 | -1 |
| 0 | -1 | 0 |

$=$

| | | |
|---|---|---|
| 9 | -14 | 26 |
| **-1** | | |
| | | |

# Convolutional Neural Networks

## Convolution in CNNs:

- Strides:
  - The stride is the distance the kernel progresses between two consecutive convolutions.
  - Large strides reduce significantly the size of the layer output (and subsequent processing in higher layers).
  - A stride can be defined by two values, the height $s_h$ and the width $s_w$.
- The previous example with $s_h = s_w = 2$:

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 5 | 8 | 0 | 4 | 8 | 0 |
| 0 | 3 | 9 | 2 | 9 | 2 | 0 |
| 0 | 4 | 9 | 2 | 3 | 6 | 0 |
| 0 | 5 | 4 | 8 | 2 | 0 | 0 |
| 0 | 5 | 8 | 0 | 0 | 8 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$*$

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 4 | -1 |
| 0 | -1 | 0 |

$=$

| | | |
|---|---|---|
| 9 | -14 | 26 |
| -1 | **-14** | |
| | | |

# Convolutional Neural Networks

## Convolution in CNNs:

- Strides:
  - The stride is the distance the kernel progresses between two consecutive convolutions.
  - Large strides reduce significantly the size of the layer output (and subsequent processing in higher layers).
  - A stride can be defined by two values, the height $s_h$ and the width $s_w$.
- The previous example with $s_h = s_w = 2$:

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 5 | 8 | 0 | 4 | 8 | 0 |
| 0 | 3 | 9 | 2 | 9 | 2 | 0 |
| 0 | 4 | 9 | 2 | 3 | 6 | 0 |
| 0 | 5 | 4 | 8 | 2 | 0 | 0 |
| 0 | 5 | 8 | 0 | 0 | 8 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$*$

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 4 | -1 |
| 0 | -1 | 0 |

$=$

| | | |
|---|---|---|
| 9 | -14 | 26 |
| -1 | -14 | **19** |
| | | |

# Convolutional Neural Networks

## Convolution in CNNs:

- Strides:
  - The stride is the distance the kernel progresses between two consecutive convolutions.
  - Large strides reduce significantly the size of the layer output (and subsequent processing in higher layers).
  - A stride can be defined by two values, the height $s_h$ and the width $s_w$.
- The previous example with $s_h = s_w = 2$:

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 5 | 8 | 0 | 4 | 8 | 0 |
| 0 | 3 | 9 | 2 | 9 | 2 | 0 |
| 0 | 4 | 9 | 2 | 3 | 6 | 0 |
| 0 | 5 | 4 | 8 | 2 | 0 | 0 |
| 0 | 5 | 8 | 0 | 0 | 8 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$*$

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 4 | -1 |
| 0 | -1 | 0 |

$=$

| | | |
|---|---|---|
| 9 | -14 | 26 |
| -1 | -14 | 19 |
| **7** | | |

# Convolutional Neural Networks

## Convolution in CNNs:

- Strides:
  - The stride is the distance the kernel progresses between two consecutive convolutions.
  - Large strides reduce significantly the size of the layer output (and subsequent processing in higher layers).
  - A stride can be defined by two values, the height $s_h$ and the width $s_w$.
- The previous example with $s_h = s_w = 2$:

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 5 | 8 | 0 | 4 | 8 | 0 |
| 0 | 3 | 9 | 2 | 9 | 2 | 0 |
| 0 | 4 | 9 | 2 | 3 | 6 | 0 |
| 0 | 5 | 4 | 8 | 2 | 0 | 0 |
| 0 | 5 | 8 | 0 | 0 | 8 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$*$

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 4 | -1 |
| 0 | -1 | 0 |

$=$

| | | |
|---|---|---|
| 9 | -14 | 26 |
| -1 | -14 | 19 |
| 7 | **-16** | |

# Convolutional Neural Networks

## Convolution in CNNs:

- Strides:
  - The stride is the distance the kernel progresses between two consecutive convolutions.
  - Large strides reduce significantly the size of the layer output (and subsequent processing in higher layers).
  - A stride can be defined by two values, the height $s_h$ and the width $s_w$.
- The previous example with $s_h = s_w = 2$:

| | | | | | | |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 5 | 8 | 0 | 4 | 8 | 0 |
| 0 | 3 | 9 | 2 | 9 | 2 | 0 |
| 0 | 4 | 9 | 2 | 3 | 6 | 0 |
| 0 | 5 | 4 | 8 | 2 | 0 | 0 |
| 0 | 5 | 8 | 0 | 0 | 8 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |

$*$

| | | |
|---|---|---|
| 0 | -1 | 0 |
| -1 | 4 | -1 |
| 0 | -1 | 0 |

$=$

| | | |
|---|---|---|
| 9 | -14 | 26 |
| -1 | -14 | 19 |
| 7 | -16 | **32** |

# Convolutional Neural Networks

## Convolution in CNNs:

- **Feature Maps:**
  Convolving an input image with a given kernel results in an output image called a **feature map**.
- The height and width of a feature map, $F$, are:

$$F_{\mathrm{h}} = \left\lfloor \frac{I_{\mathrm{h}} - W_{\mathrm{h}} + 2 \times p_{\mathrm{h}}}{s_{\mathrm{h}}} + 1 \right\rfloor$$

$$F_{\mathrm{w}} = \left\lfloor \frac{I_{\mathrm{w}} - W_{\mathrm{w}} + 2 \times p_{\mathrm{w}}}{s_{\mathrm{w}}} + 1 \right\rfloor$$

with

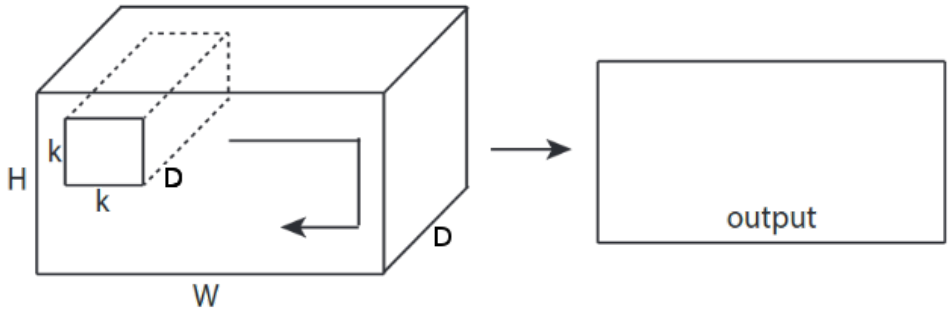
| | |
|---|---|
| input: | $I_{\mathrm{h}} \times I_{\mathrm{w}}$ |
| kernel: | $W_{\mathrm{h}} \times W_{\mathrm{w}}$ |
| padding: | $p_{\mathrm{h}}$ and $p_{\mathrm{w}}$ |
| strides: | $s_{\mathrm{h}}$ and $s_{\mathrm{w}}$ |

- Until now, for simplicity, we have represented the input of a convolution layer as a two-dimensional signal (*e.g.* image). In reality, images can have three or more channels, and in CNNs, the input of a convolution layer can be composed of several feature maps from the previous layer of the same height and width.

# Convolutional Neural Networks

## Convolution in CNNs:

- The dimensions of the inputs of a convolutional layer can be described by three numbers: height, width, and depth.
- The kernels also have a depth that matches the depth of their inputs.

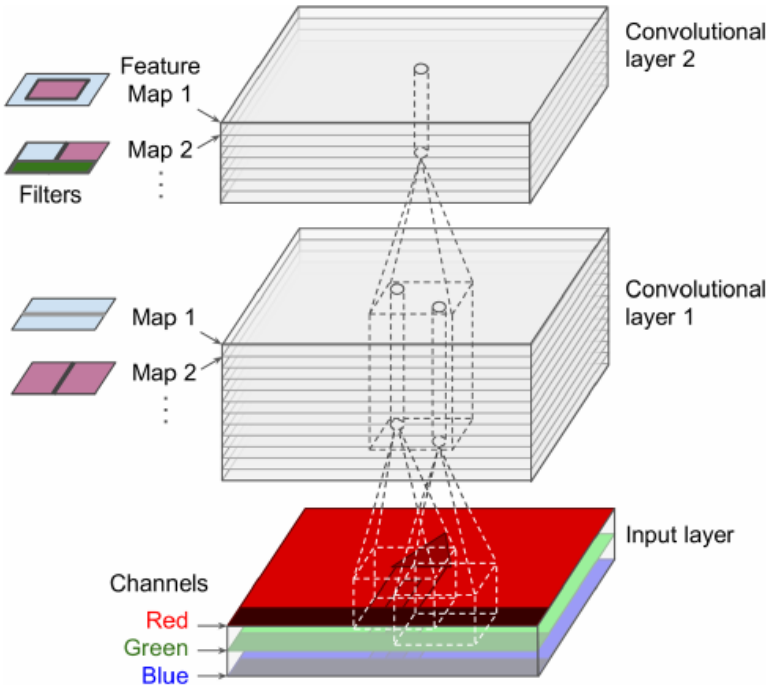



- In a convolution layer, a single kernel outputs a 2-dimensional feature map (of depth one). Applying multiple filters results in an output with a depth equal to the number of filters (kernels) used.
- The values of the feature maps are obtained convolving the inputs with the kernel, adding a bias term, and passing the outcome through an activation function.

# Convolutional Neural Networks

## Multiple Feature Maps:

- A single feature map shares the same parameters (the kernel weights and the bias term), which drastically reduces the total number of parameters in the model.
- Stacking multiple feature maps enables the detection of complex patterns in the data.
- Once the CNN has learned a pattern in one location, it can recognize it in any other location.




by Aurélien Géron

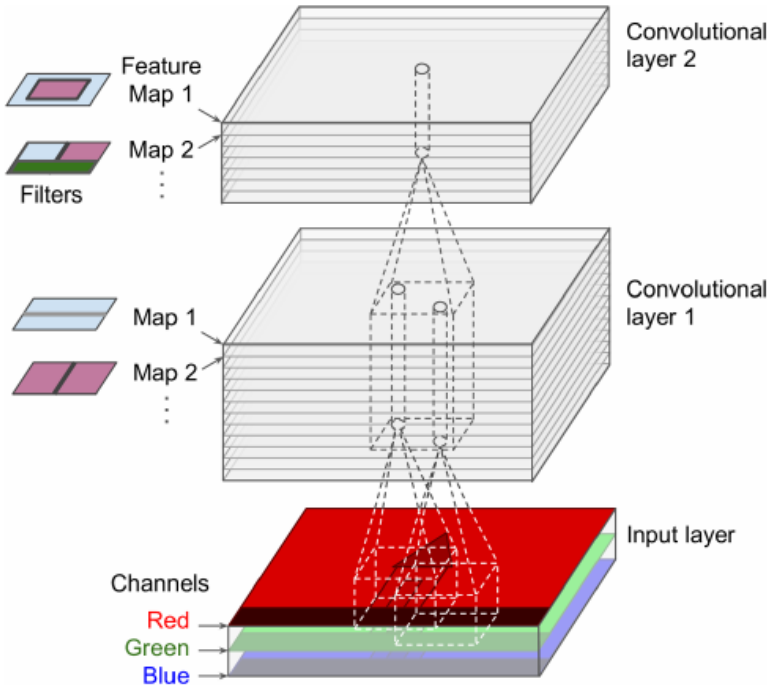
### Simple Example:

Consider a CNN with an input convolution layer with 100, $5\times5$ kernels, on a RGB, $512\times512$ image. the number of parameters in that layer is: $(3\times5\times5+1)\times100=7600$.
For comparison, consider a MLP network with a fully connect input layer with 100 output neurons. The number of parameters in that layer is: $(3\times512\times512+1)\times100=78643300$.

# Convolutional Neural Networks

## Multiple Feature Maps:

- A single feature map shares the same parameters (the kernel weights and the bias term), which drastically reduces the total number of parameters in the model.
- Stacking multiple feature maps enables the detection of complex patterns in the data.
- Once the CNN has learned a pattern in one location, it can recognize it in any other location.




by Aurélien Géron

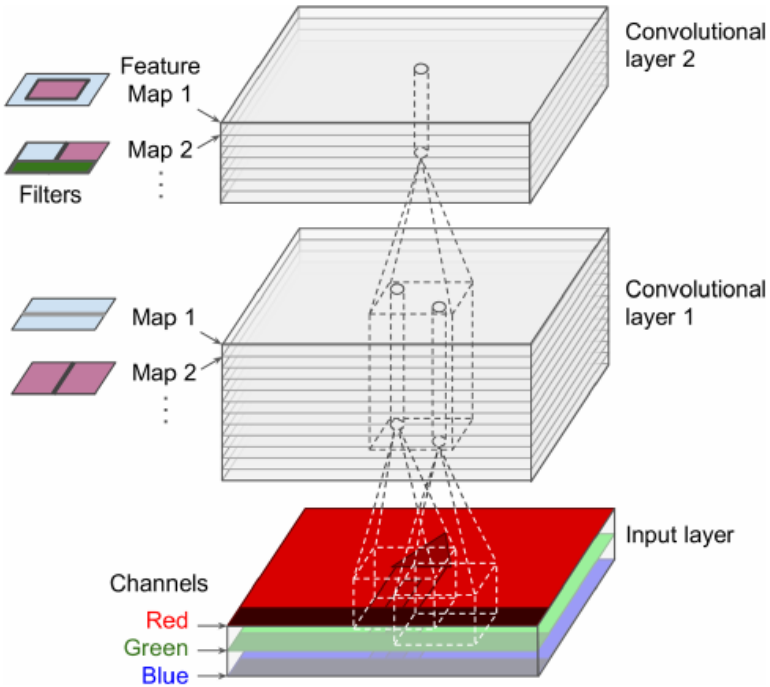
### Memory Requirements:

One drawback with CNNs is that they require a lot of RAM memory, especially during training. For instance, an input convolution layer with 100 filters, for an RGB image of $512 \times 512$, with stride 1 and zero padding, (and admitting the values are stored in 32-bit floats), would occupy: $3 \times 512 \times 512 \times 100 \times 32 = 2516582400$ bits (exactly 300 MB). This is for only one example! If the batch size is comprised of 100 instances, this layer will use over 29 GB of RAM.

# Convolutional Neural Networks

## Multiple Feature Maps:

- A single feature map shares the same parameters (the kernel weights and the bias term), which drastically reduces the total number of parameters in the model.
- Stacking multiple feature maps enables the detection of complex patterns in the data.
- Once the CNN has learned a pattern in one location, it can recognize it in any other location.




by Aurélien Géron

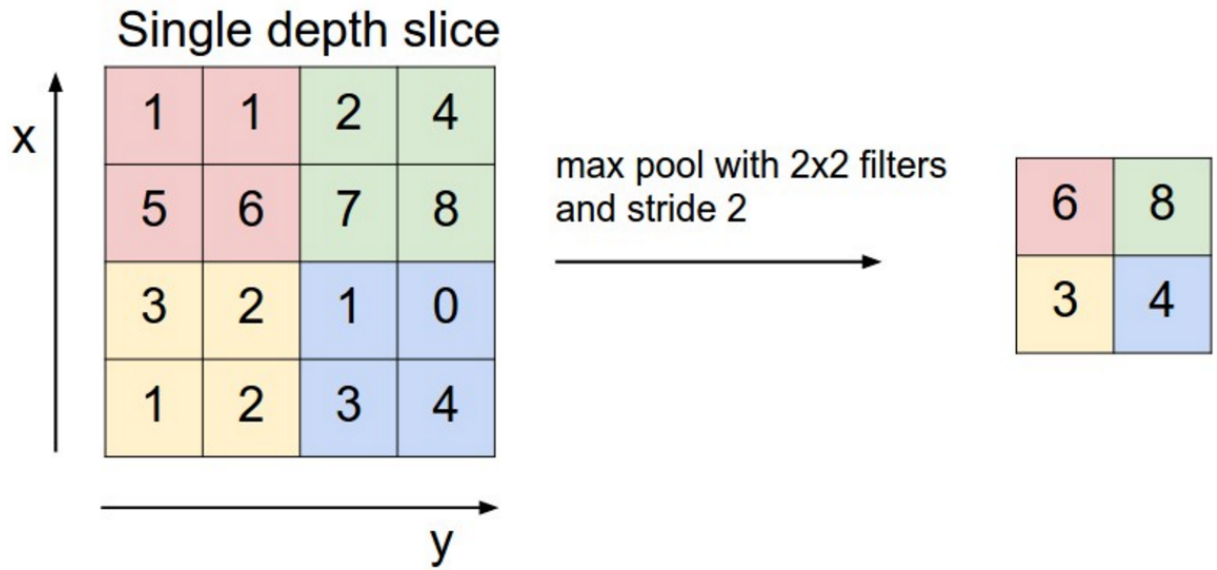
### Memory Requirements:

This memory requirements can be lowered by reducing the batch-size, using 16-bit floats, or using large strides, but these methods are often not as effective as using **pooling layers** after the convolution layers.

# Convolutional Neural Networks

## Pooling Layer:

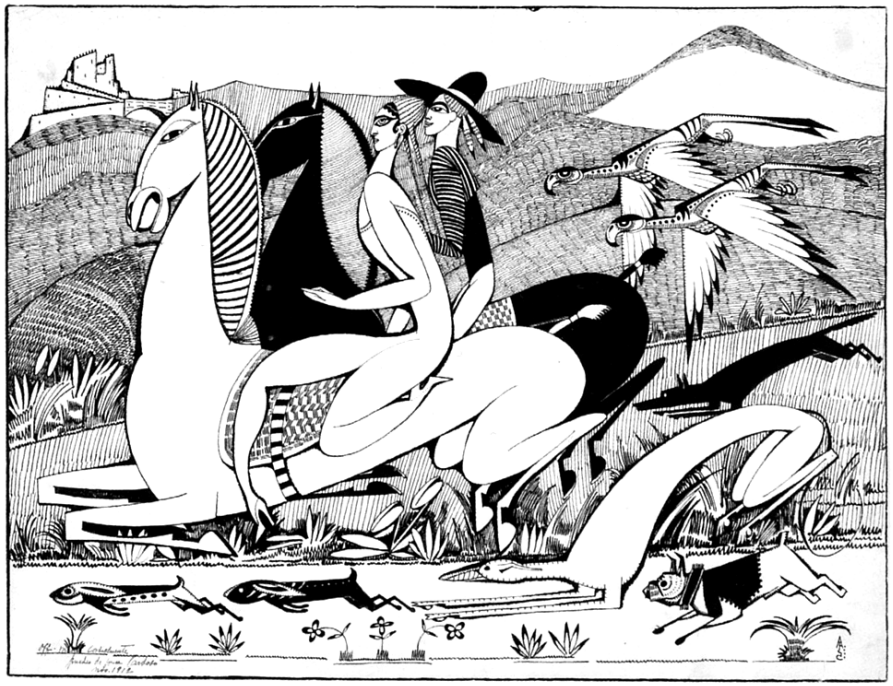
- Pooling layers are used to reduce a feature map resolution (height and width) through sub-sampling in order to lower the computational and memory loads.
- The pooling operation consists in calculating some statistical value, like the mean or the maximum, of a small rectangular patch of the feature map. Like the kernel in the convolutional layer, on must define the size of the patch, its stride and the padding type.



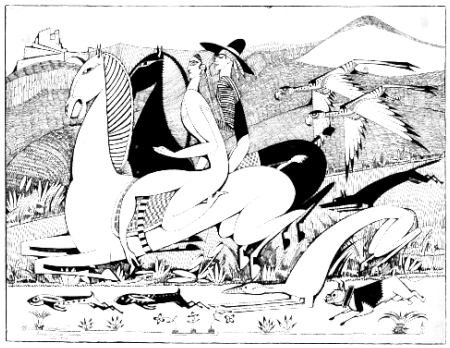
# Convolutional Neural Networks

## Pooling Layer:

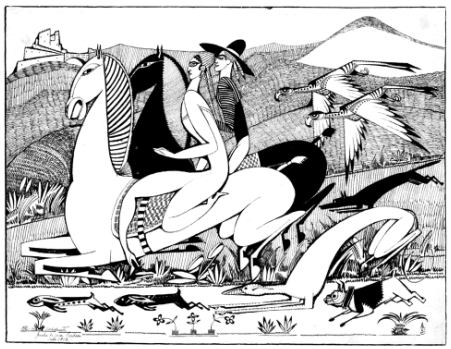
- Pooling layers are used to reduce a feature map resolution (height and width) through sub-sampling in order to lower the computational and memory loads.
- The pooling operation consists in calculating some statistical value, like the mean or the maximum, of a small rectangular patch of the feature map. Like the kernel in the convolutional layer, on must define the size of the patch, its stride and the padding type.



by Amadeo Souza-Cardoso



Max Pooling 2×2, stride 2

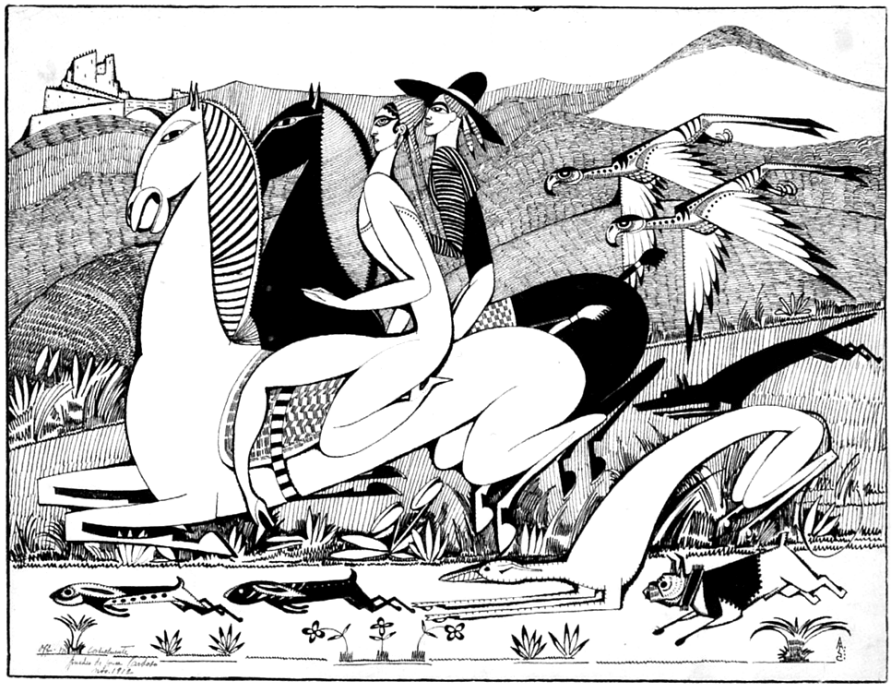


Av. Pooling 2×2, stride 2

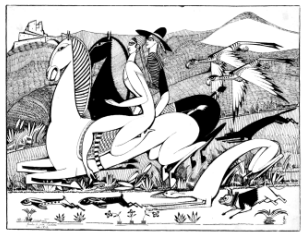
# Convolutional Neural Networks

## Pooling Layer:

- Pooling layers are used to reduce a feature map resolution (height and width) through sub-sampling in order to lower the computational and memory loads.
- The pooling operation consists in calculating some statistical value, like the mean or the maximum, of a small rectangular patch of the feature map. Like the kernel in the convolutional layer, on must define the size of the patch, its stride and the padding type.



Max Pooling 3×3, stride 3

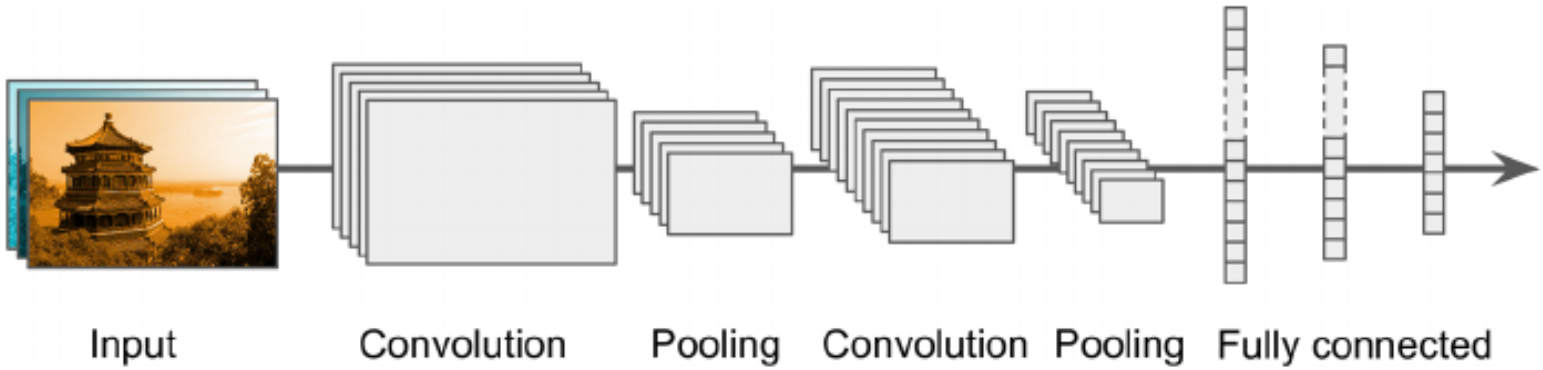


Av. Pooling 3×3, stride 3

# Convolutional Neural Networks

## CNNs Architectures:

- A standard processing block in CNNs consists of one or more convolutional layers followed by a pooling layer. Typical CNN architectures stack several of these blocks followed by a MLP network composed of a few fully connected layers and a final layer that outputs the predictions (*e.g.* softmax for class probabilities).

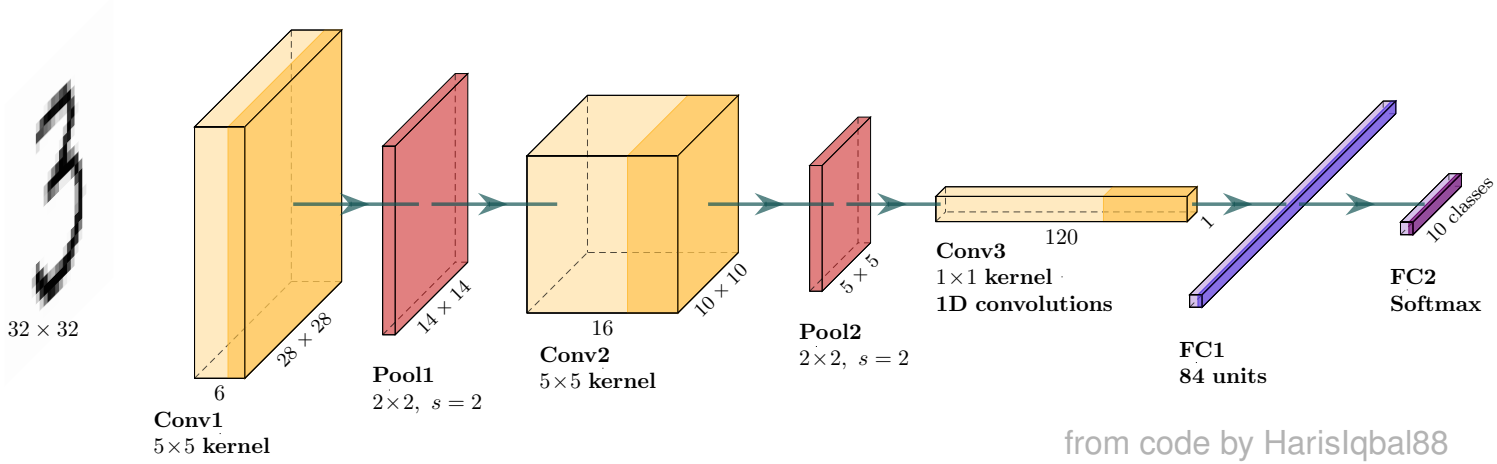



by Aurélien Géron

# Convolutional Neural Networks with Tensorflow-Keras

# CNNs with Tensorflow-Keras

Here is popular CNN architecture called LeNet-5[1] used to tackle the MNIST dataset. The LeNet-5 architecture is the following:




from code by HarisIqbal88

- MNIST digit images are 28×28 pixels, but they are zero-padded and normalized before been fed to the network.
- The pooling layers perform average pooling with a learned scaling coefficient and bias term (one per feature map).
- The activation functions in all layers consist of hyperbolic tangents, except for the last layer which is a Radial-Basis Function layer.

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, november 1998.

# CNNs with Tensorflow-Keras

The following code implements a CNN similar to LeNet-5, using `tf-keras`.

```
from tensorflow.keras import models
from tensorflow.keras import layers
lenet5 = models.Sequential()
lenet5.add(layers.Conv2D(6, (5, 5), activation='relu',\
input_shape=(28, 28, 1),padding="same"))
lenet5.add(layers.AveragePooling2D((2, 2)))
lenet5.add(layers.Conv2D(16, (5, 5), activation='relu'))
lenet5.add(layers.AveragePooling2D((2, 2)))
lenet5.add(layers.Conv2D(120, (1, 1), activation='relu'))
lenet5.add(layers.Flatten())
lenet5.add(layers.Dense(84, activation='relu'))
lenet5.add(layers.Dense(10, activation='softmax'))
```

# CNNs with Tensorflow-Keras

```
lenet5.summary()
```

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_1 (Conv2D) | (None, 28, 28, 6) | 156 |
| average_pooling2d_1 (Average | (None, 14, 14, 6) | 0 |
| conv2d_2 (Conv2D) | (None, 10, 10, 16) | 2416 |
| average_pooling2d_2 (Average | (None, 5, 5, 16) | 0 |
| conv2d_3 (Conv2D) | (None, 5, 5, 120) | 2040 |
| flatten_1 (Flatten) | (None, 3000) | 0 |
| dense_1 (Dense) | (None, 84) | 252084 |
| dense_2 (Dense) | (None, 10) | 850 |

```
Total params:  257,546
Trainable params:  257,546
Non-trainable params:  0
```

- Note that more than 98% of the network weights belong to the fully connected layers.

# CNNs with Tensorflow-Keras

- Now let us load and prepare the MNIST data,
  - Load the data:

    ```
    (X,y),(Xtest,ytest)=mnist.load_data()
    ```
  - Scale the data and divide it into training, validation and test sets

    ```
    Ntrain=40000
    Xtrain=X[:Ntrain]/255.
    ytrain=y[:Ntrain]
    Xvalid=X[Ntrain:]/255.
    yvalid=y[Ntrain:]
    Xtest=Xtest/255.
    ```
  - Reshape the data into a 4D tensor

    ```
    Xtrain=Xtrain.reshape(-1,28,28,1)
    Xvalid=Xvalid.reshape(-1,28,28,1)
    Xtest=Xtest.reshape(-1,28,28,1)
    ```
  - One hot encode the labels:

    ```
    ytrainB=keras.utils.to_categorical(ytrain)
    yvalidB=keras.utils.to_categorical(yvalid)
    ytestB=keras.utils.to_categorical(ytest)
    ```
- and compile the model.

```
lenet5.compile(optimizer="nadam",loss="categorical_crossentropy",metrics=["accuracy"])
```

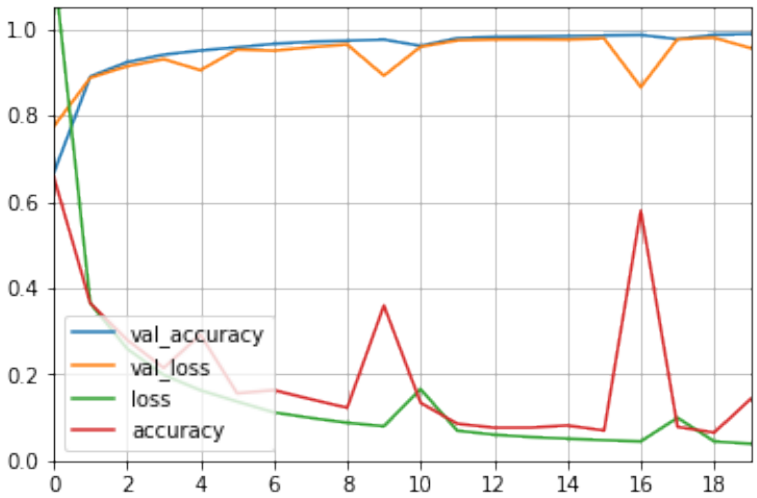
# CNNs with Tensorflow-Keras

- Training,

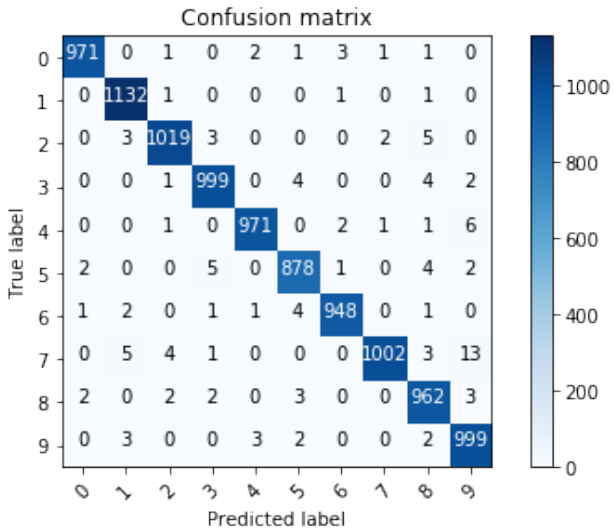
```
tLog = lenet5.fit(Xtrain, ytrainB, epochs=20,batch_size=1024,validation_data=(Xvalid, yvalidB))

    Train on 40000 samples, validate on 20000 samples
    Epoch 1/20
    40000/40000 [==============================] - 14s 346us/sample - loss:  1.1475 -
    accuracy:  0.6818 - val_loss:  0.4906 - val_accuracy:  0.8418
           ⋮
    Epoch 20/20 40000/40000 [==============================] - 13s 317us/sample - loss:
    0.0465 - accuracy:  0.9854 - val_loss:  0.0745 - val_accuracy:  0.9779
```

- and evaluation.




Evolution (training and validation loss+accuracy)




Test acuracy = 98.81%

# CNNs with Tensorflow-Keras

- Training,

```
tLog = lenet5.fit(Xtrain, ytrainB, epochs=20,batch_size=1024,validation_data=(Xvalid, yvalidB))

    Train on 40000 samples, validate on 20000 samples
    Epoch 1/20
    40000/40000 [==============================] - 14s 346us/sample - loss:  1.1475 -
    accuracy:  0.6818 - val_loss:  0.4906 - val_accuracy:  0.8418
           ⋮
    Epoch 20/20 40000/40000 [==============================] - 13s 317us/sample - loss:
    0.0465 - accuracy:  0.9854 - val_loss:  0.0745 - val_accuracy:  0.9779
```

- and evaluation.
  Some of the errors (total of 119 in 10000)