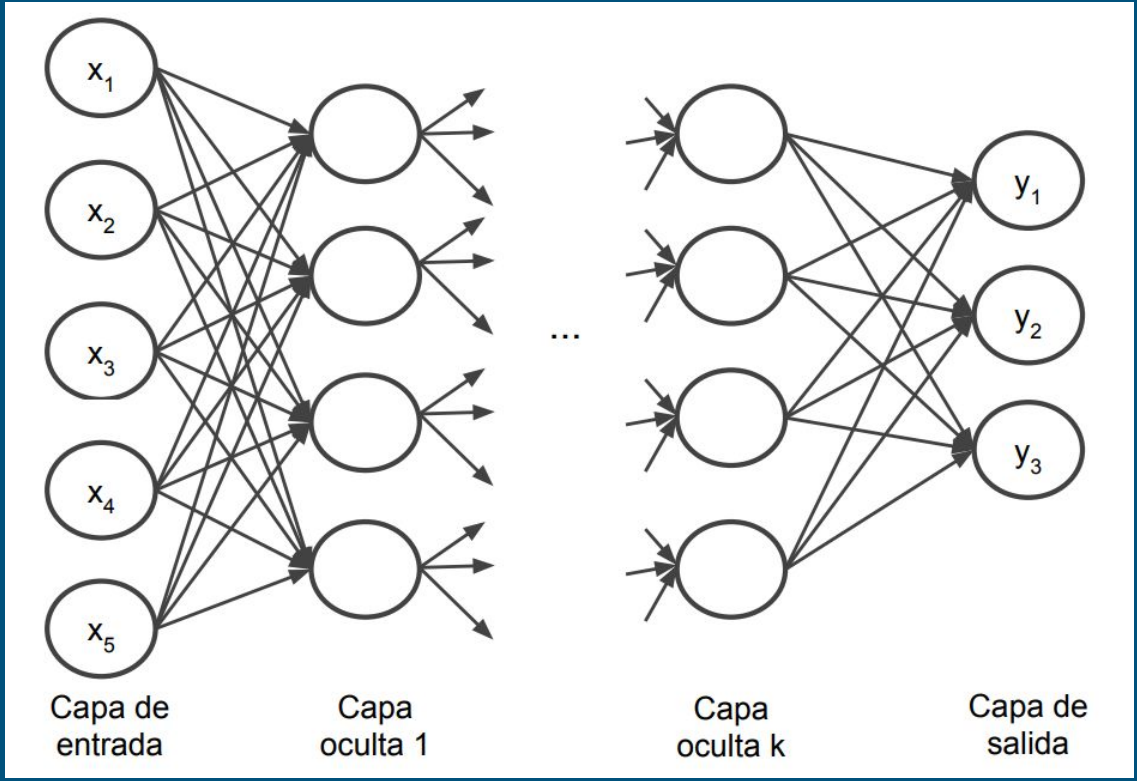
Machine Learning en Raspberry Pi

Introducción a las redes neuronales convolucionales

Lic. Alexis Luszczak
Lic. Lucas Finazzi

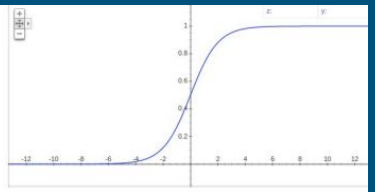
Repaso: Red Feedforward



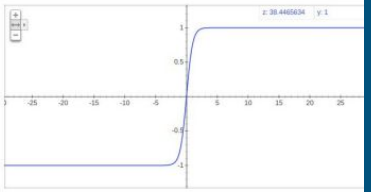
Función de activación

- **Función acotada y preferentemente derivable**

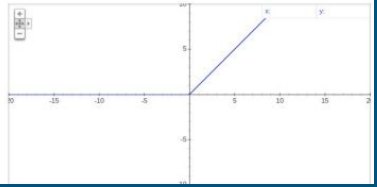
Función sigmoide o logística: $\sigma(z) = \dfrac{1}{1+e^{-z}}$



Tangente hiperbólica: $\tanh(z) = \dfrac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$



ReLU: $\text{relu}(z) = max(0, z)$



Problema en imagenes

- **La feedforward, no tiene información de quién es el vecino ya que está todo conectado. Tienes además una cantidad de pesos igual a el número de pixeles por el número de colores por el número de neuronas. Lo cual escala mucho.**

Una solución

- **Usar redes neuronales convolucionales (CNN)**
- **La cnn solo conecta unidades que se corresponden a píxeles vecinos, se fuerza la entrada de unidades espacialmente cercanas. Se reduce así el número de conexiones con los vecinos.**
- **Además se comparten los parámetros a ajustar (los núcleos son compartidos por varios pixeles), esto a su vez permite invarianza de traslación y evitar el sobreajuste.**
- **Menor necesidad de computo.**

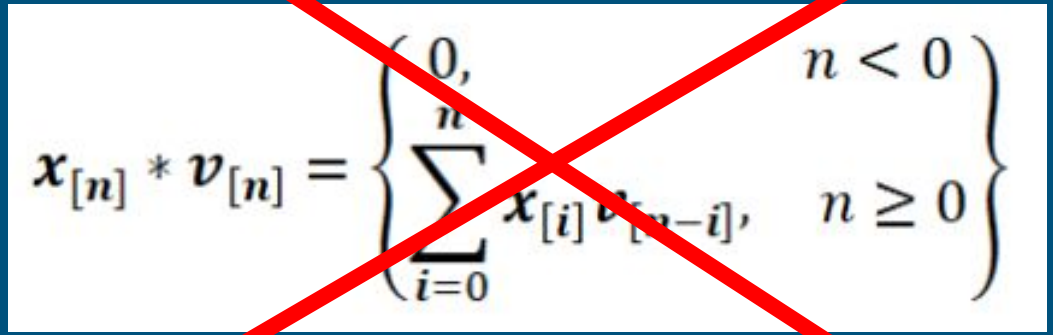
Convolutional Neural Network CNN

Matemáticamente la convolución es esto:

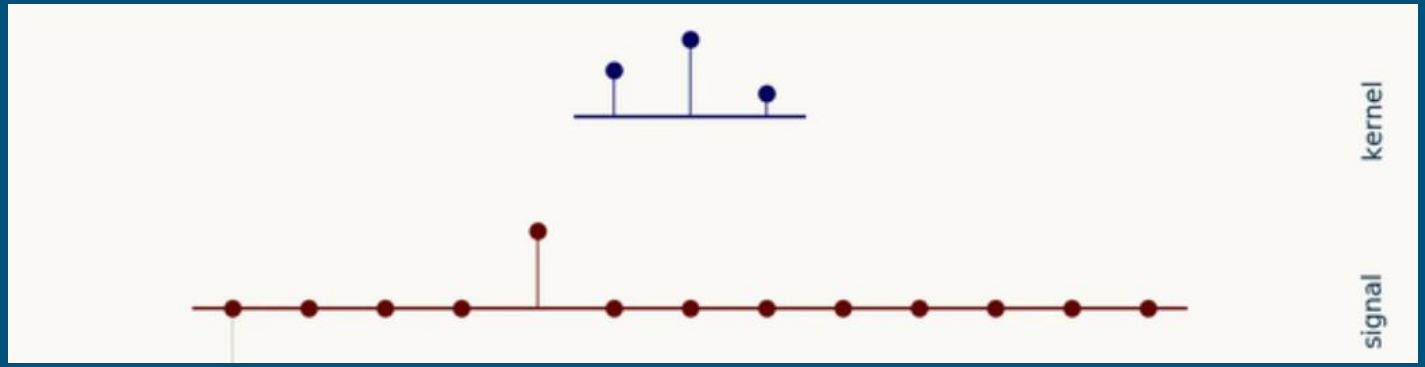
$$x_{[n]} * v_{[n]} = \begin{cases} 0, & n < 0 \\ \sum_{i=0}^{n} x_{[i]} v_{[n-i]}, & n \geq 0 \end{cases}$$

Convolutional Neural Network CNN

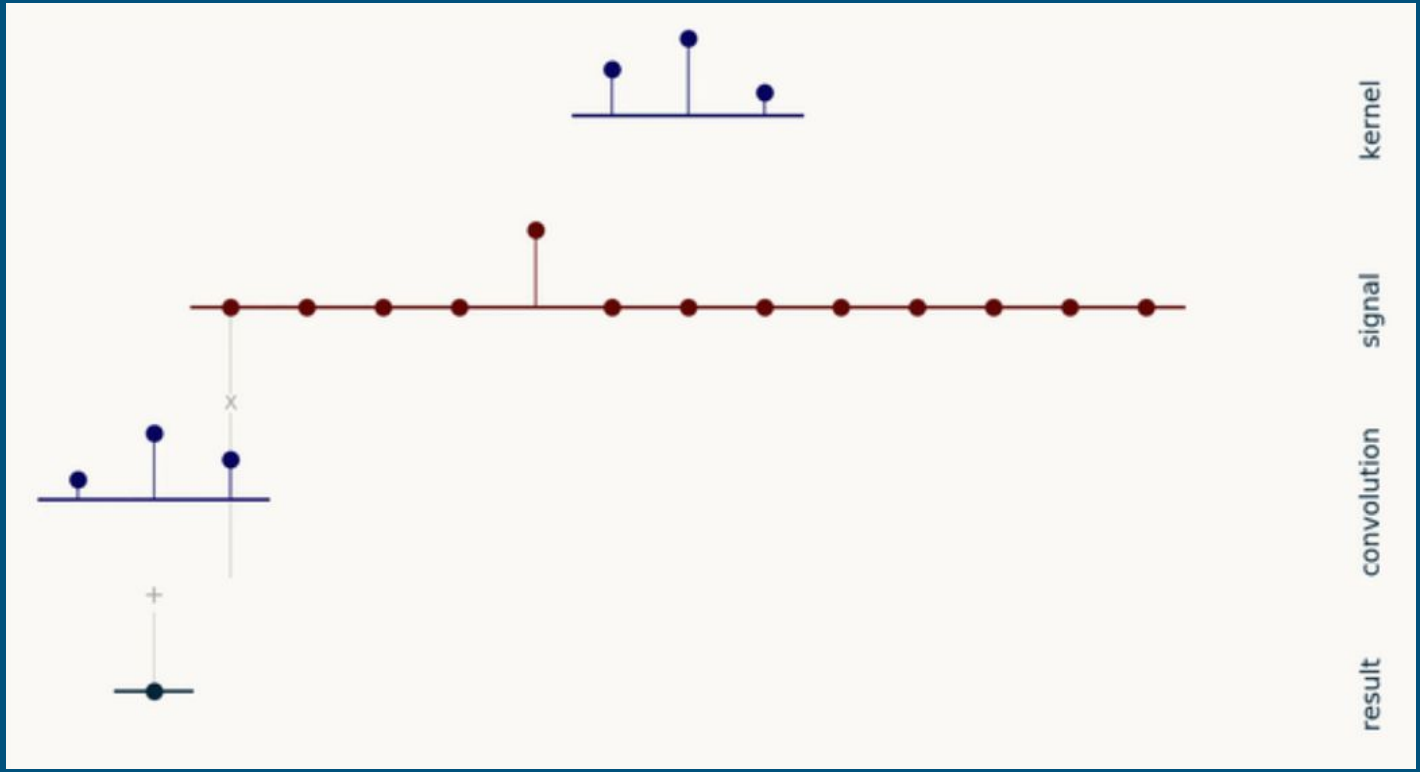
Matemáticamente la convolución es esto:



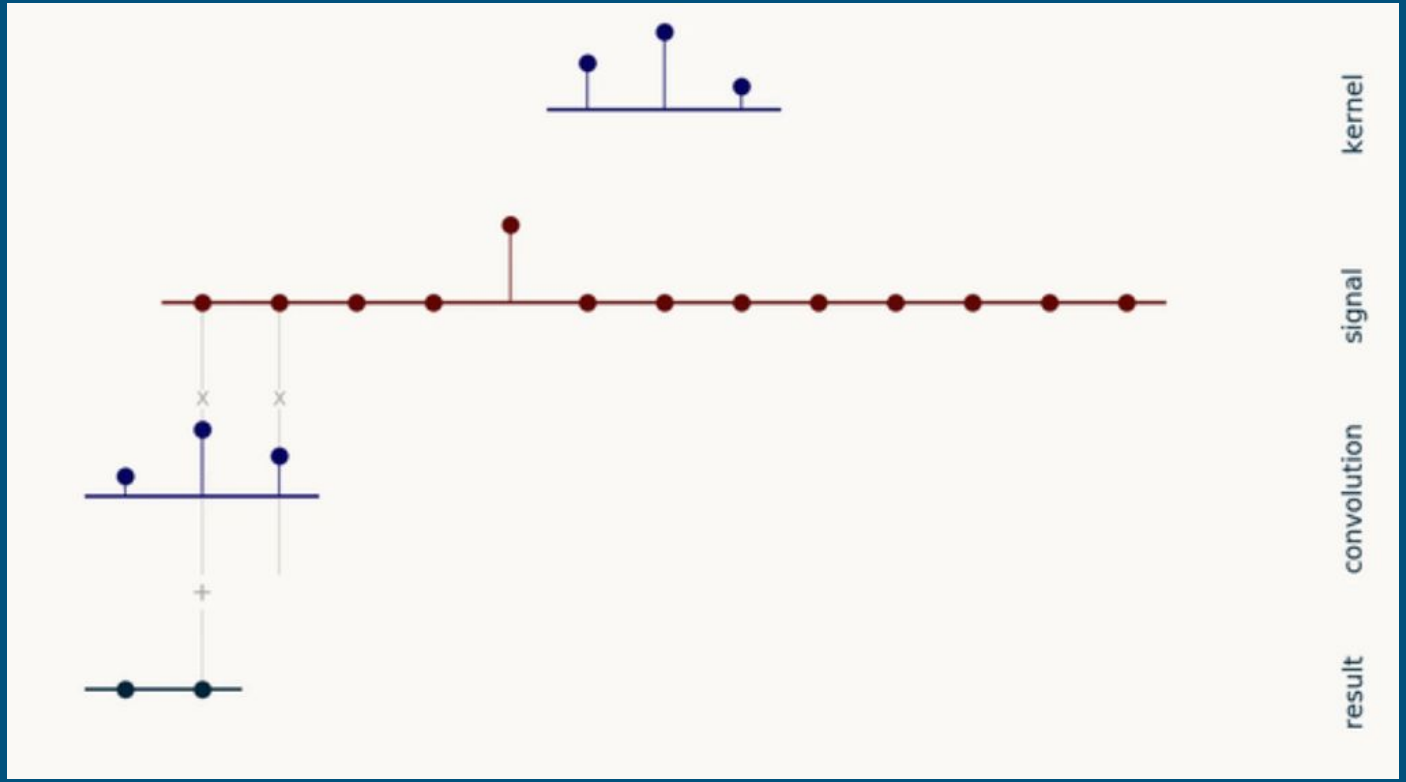
Convolutional Neural Network CNN



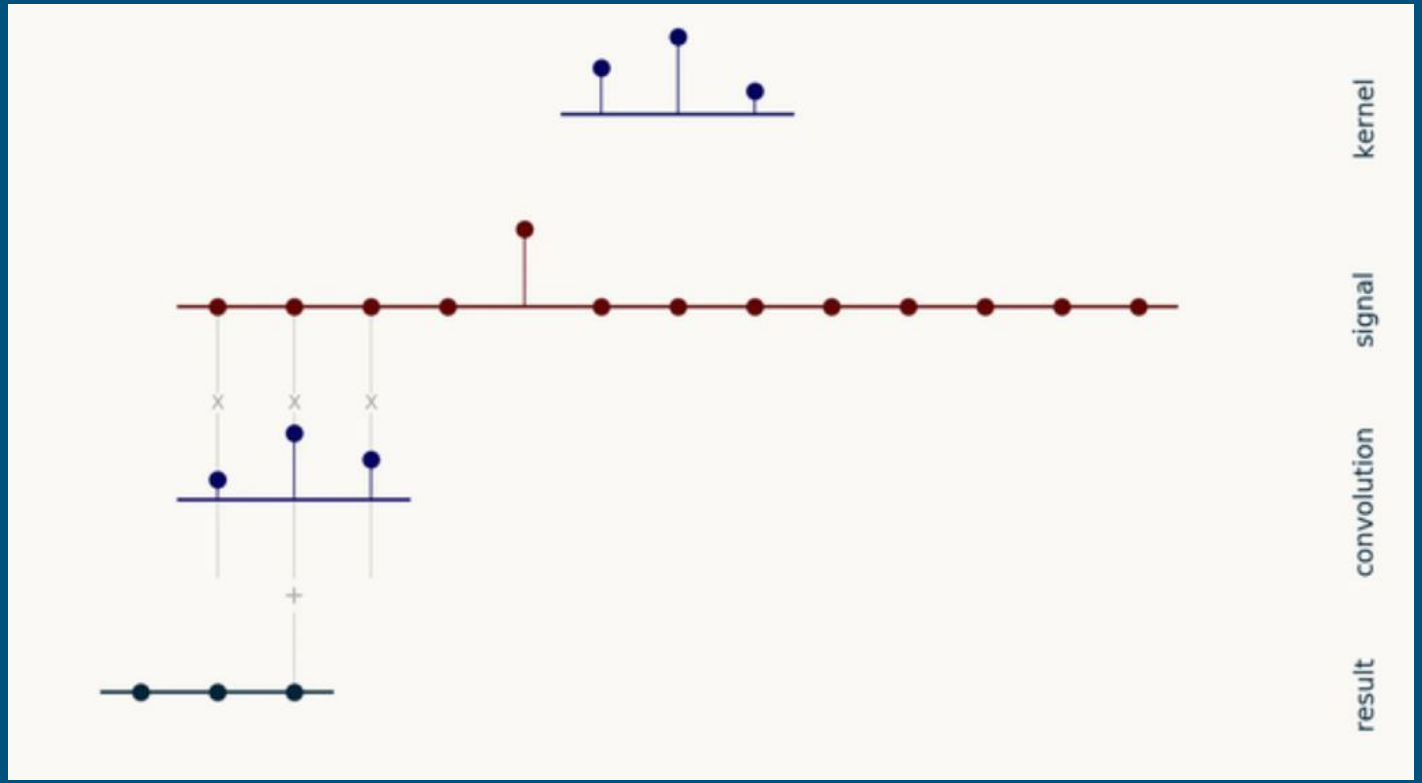
Convolutional Neural Network CNN



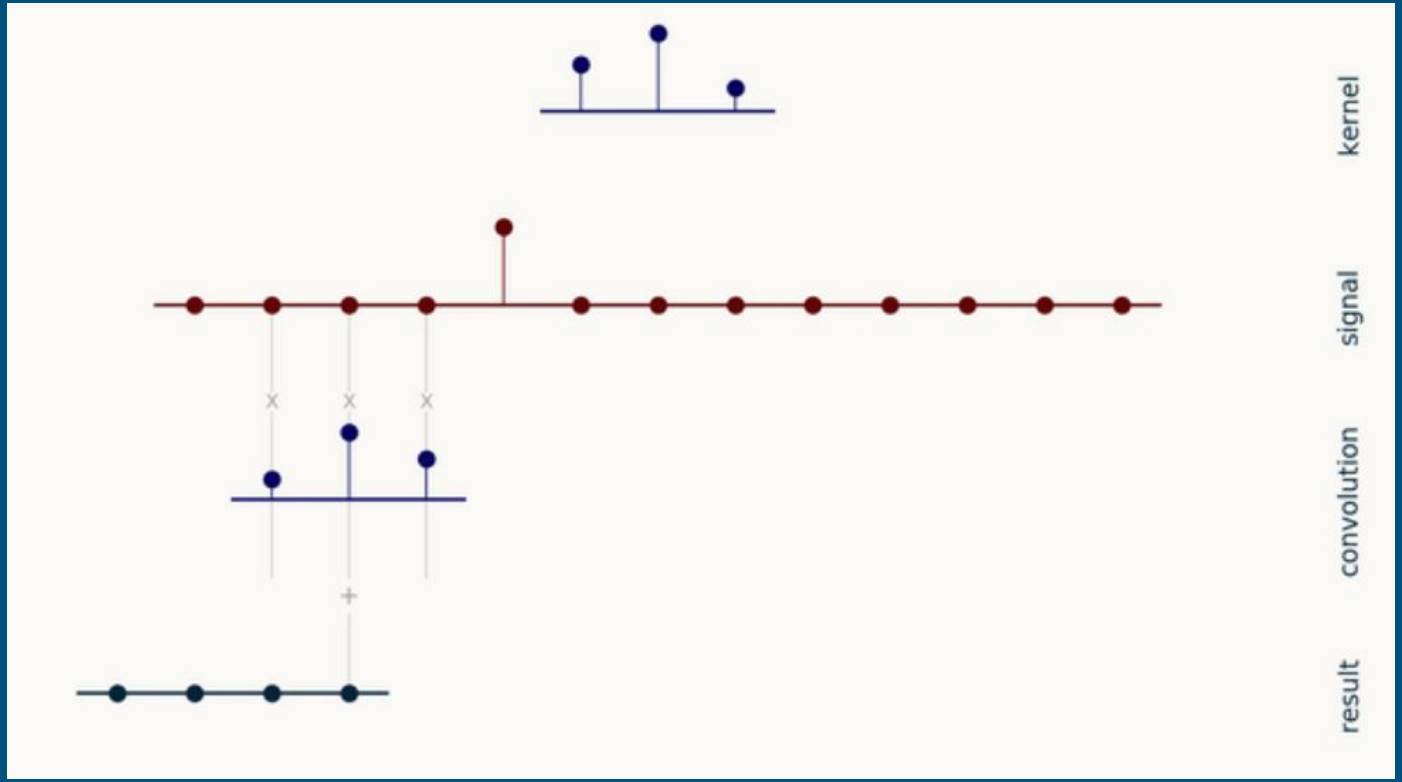
Convolutional Neural Network CNN



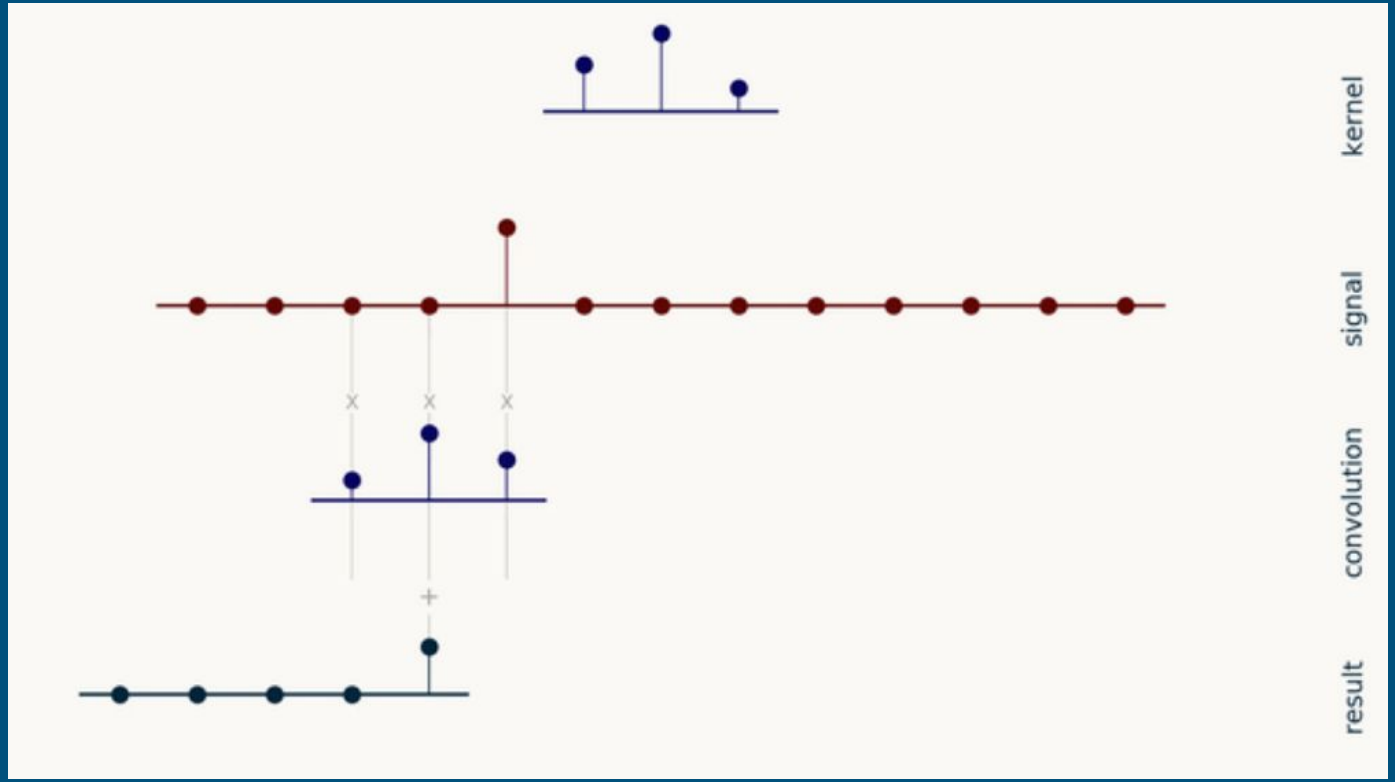
Convolutional Neural Network CNN



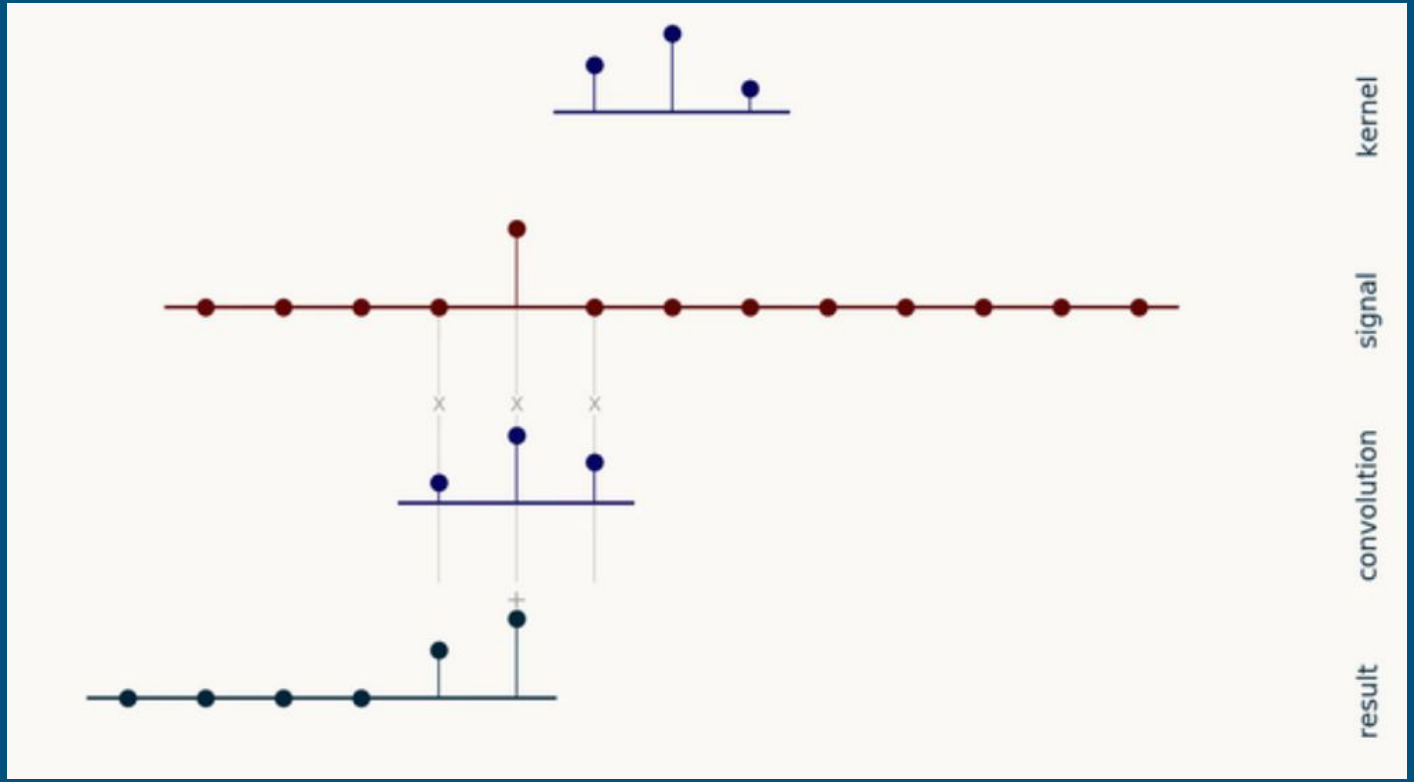
Convolutional Neural Network CNN



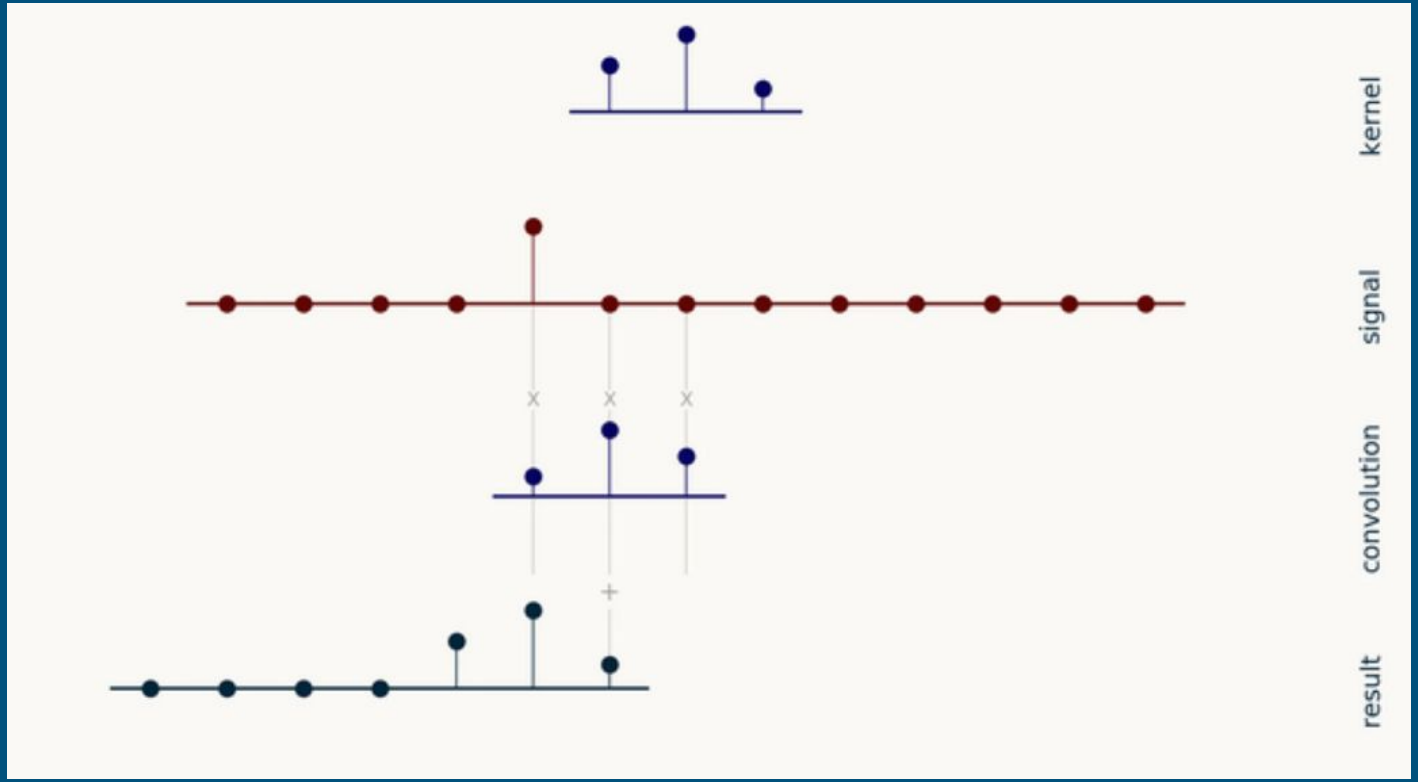
Convolutional Neural Network CNN



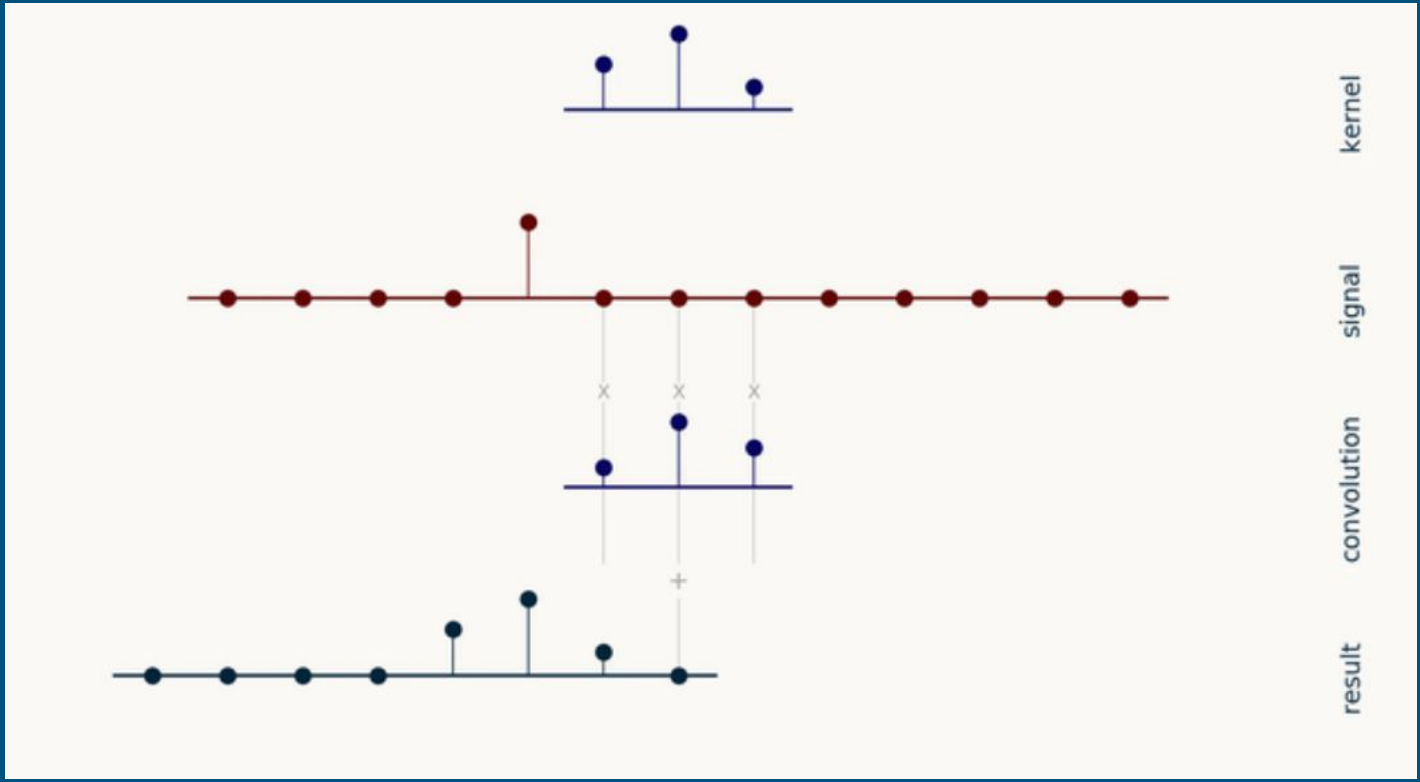
Convolutional Neural Network CNN



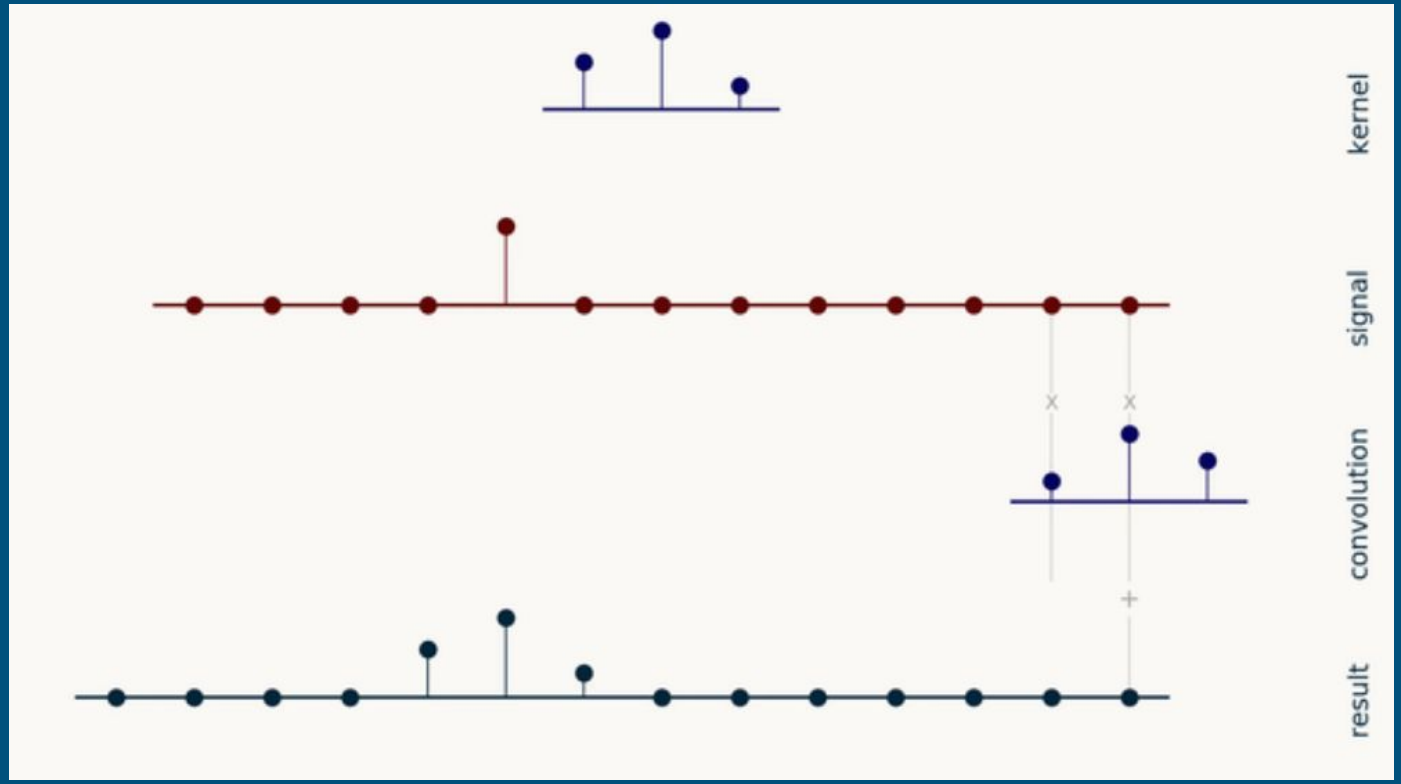
Convolutional Neural Network CNN



Convolutional Neural Network CNN



Convolutional Neural Network CNN



Ejemplo de aplicación de un núcleo

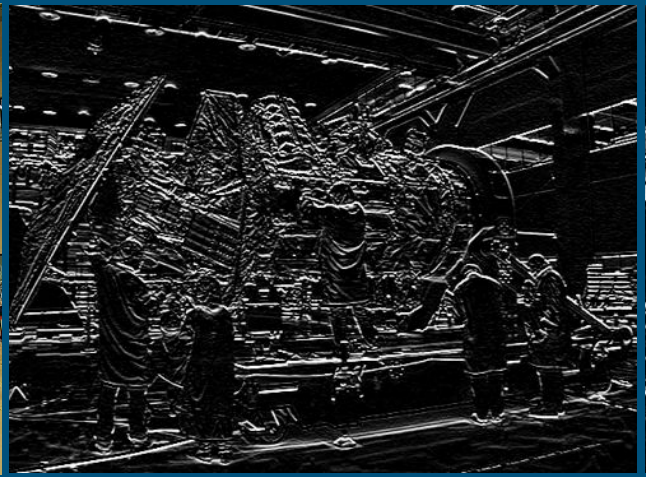
- **Abran el recurso ejemplo conv_example.py o ipynb**

Ejemplo de aplicación de un núcleo

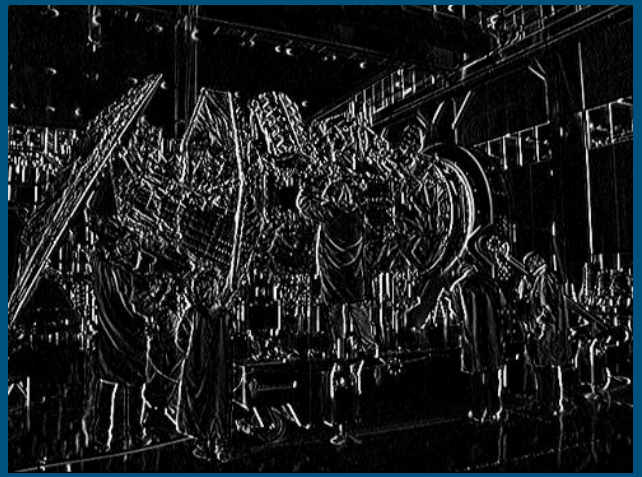
- **Abran el recurso ejemplo conv_example.py o ipynb**

Original

Núcleo horizontal



Núcleo vertical



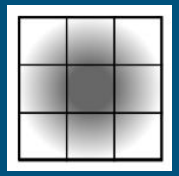
Stride

Es la distancia que nos movemos para aplicar la convolución. Mientras mayor sea, vamos a detectar relaciones más lejanas.

Stride

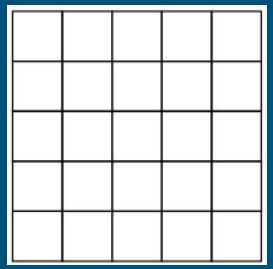
Es la distancia que nos movemos para aplicar la convolución. Mientras mayor sea, vamos a detectar relaciones más lejanas.

Núcleo

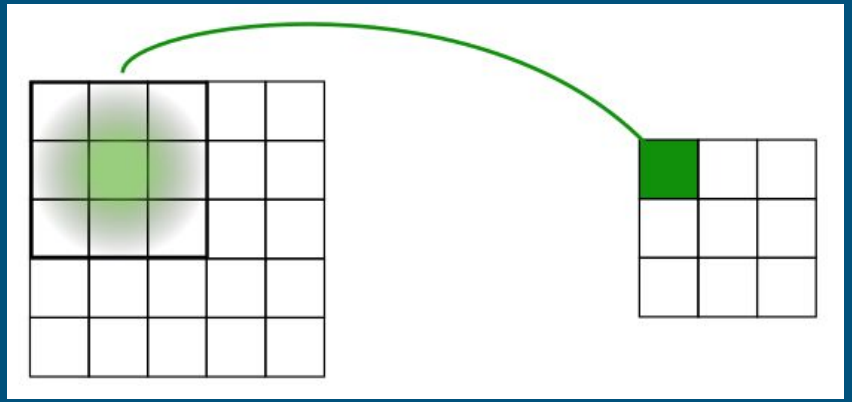


Ejemplo núcleo 3x3 y stride de 1

Imágen

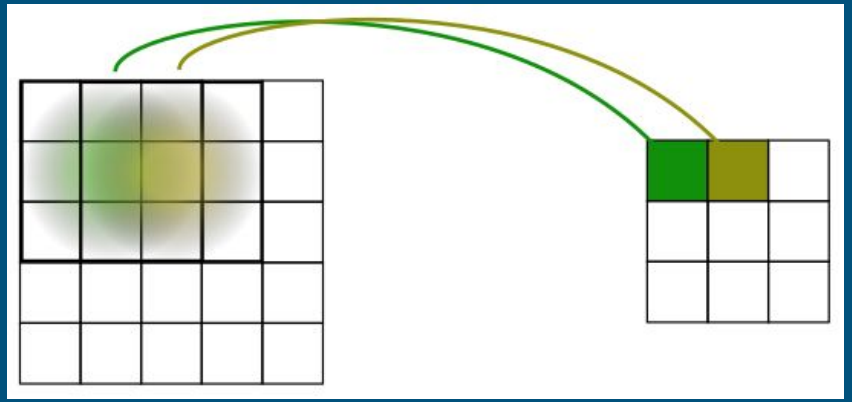


Stride



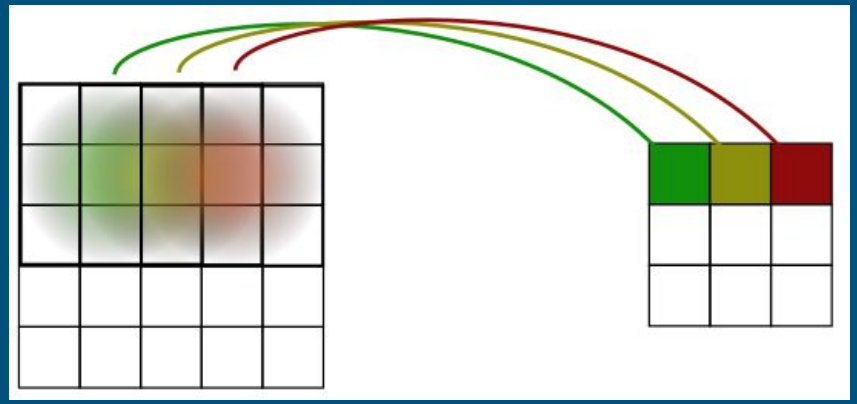
Ejemplo núcleo 3x3 y stride de 1

Stride



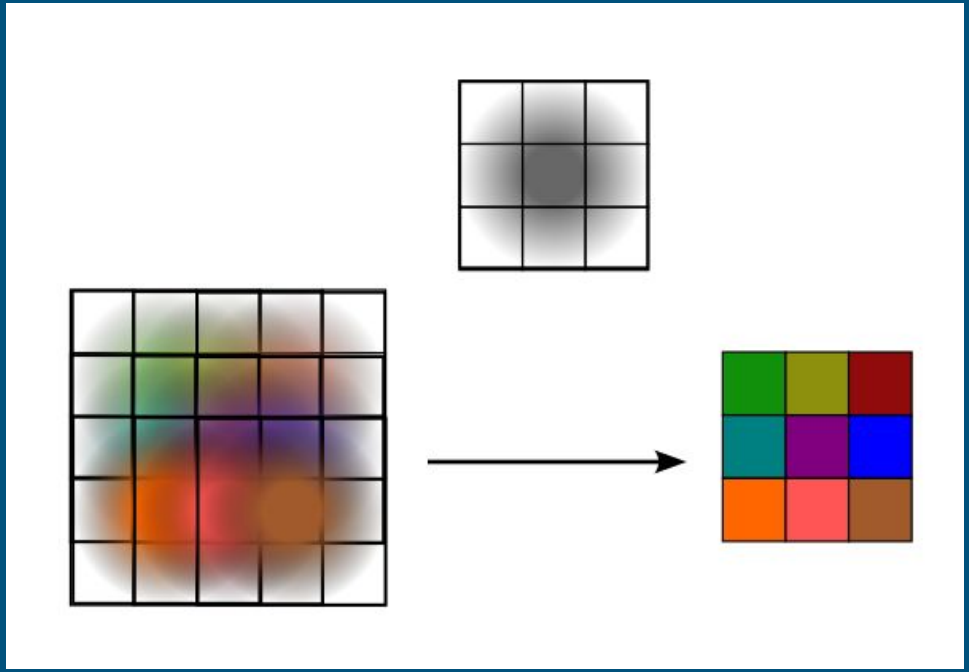
Ejemplo núcleo 3x3 y stride de 1

Stride



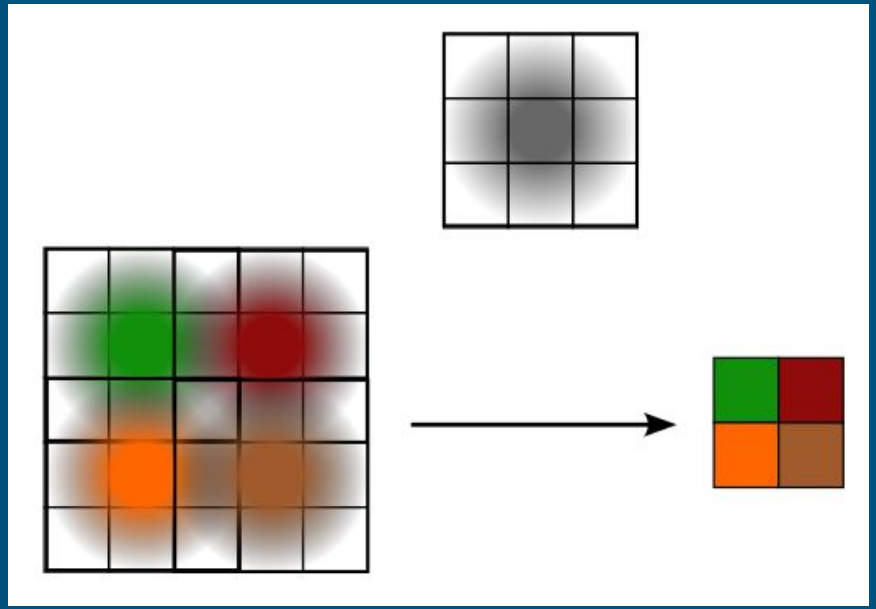
Ejemplo núcleo 3x3 y stride de 1

Stride



Ejemplo núcleo 3x3 y stride de 1

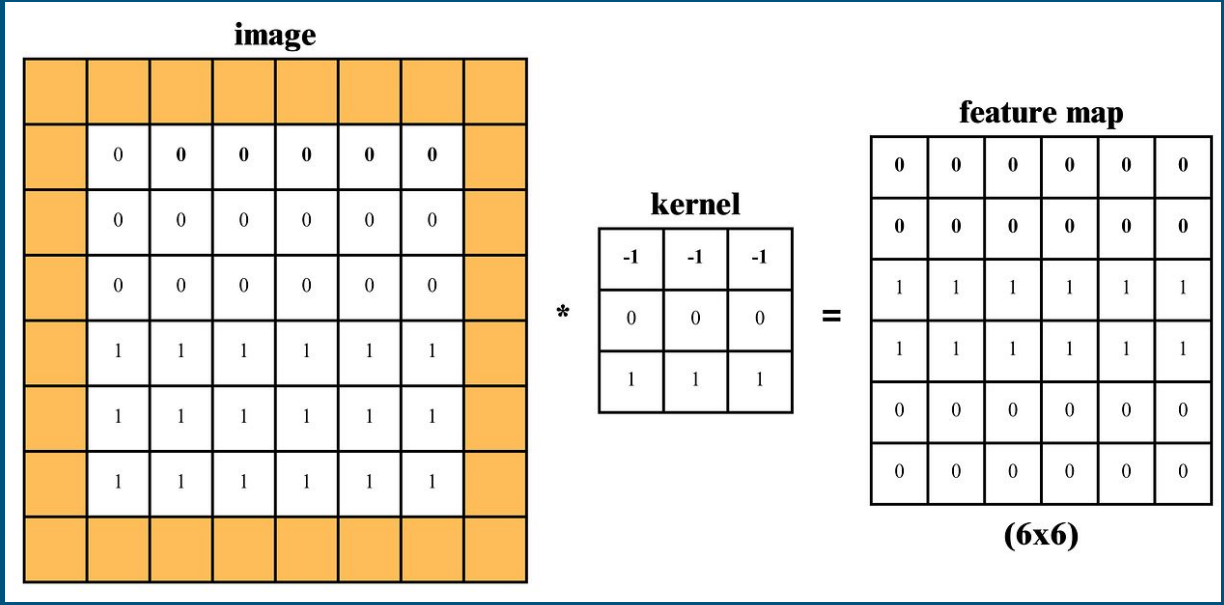
Stride



Ejemplo núcleo 3x3 y stride de 2

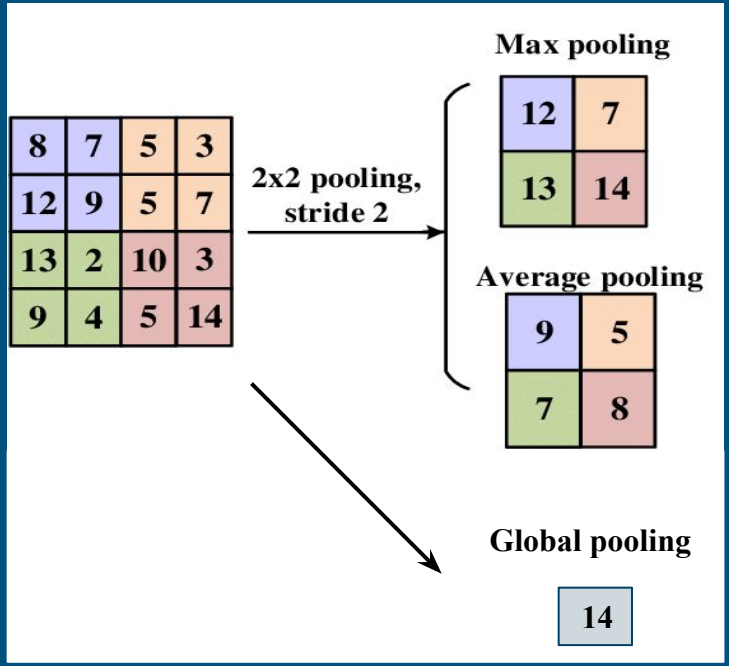
Padding

Consiste en agregar bordes a la imagen original para compensar la dimensión de la salida al aplicar la convolución.



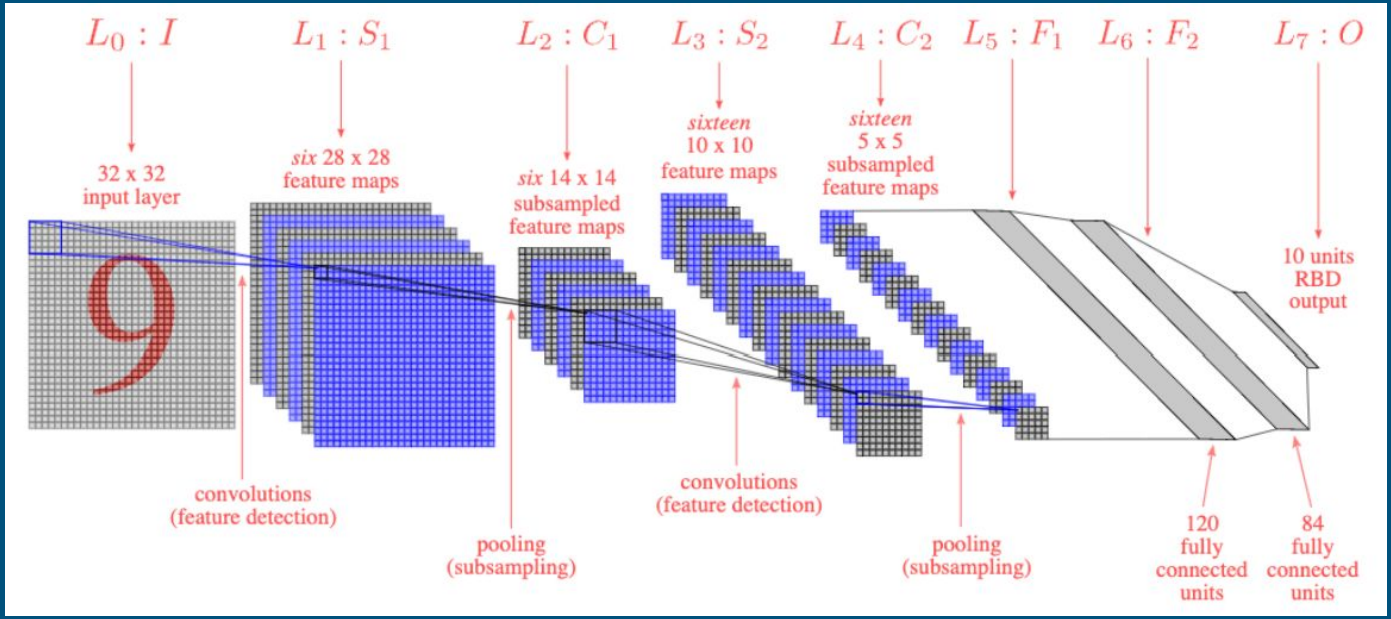
Pooling

Se utiliza para aumentar el campo receptivo. Tipos:

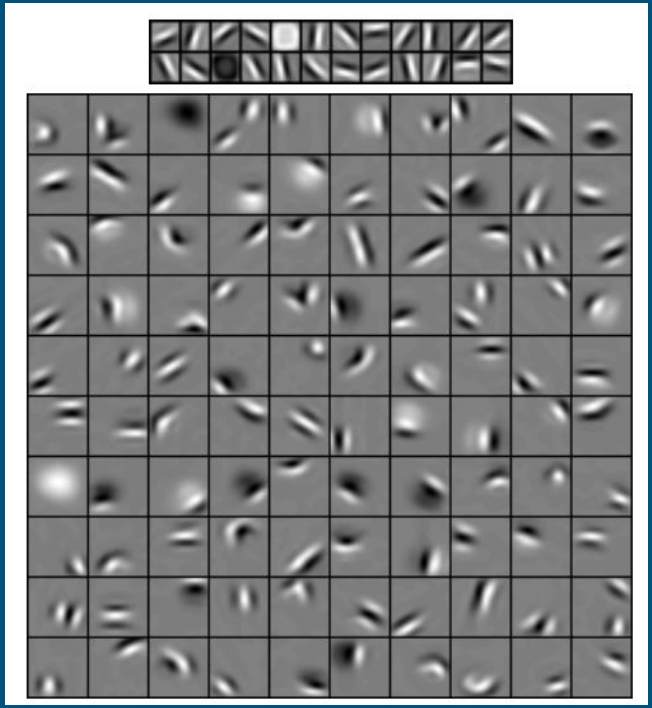


LeNet 1989

- Definido por la operación de convolución
- Entrenado por back propagation
- LeNet-5 (1998) 95% de precisión sobre MNIST (igual a SVM)



¿Qué ven las capas?

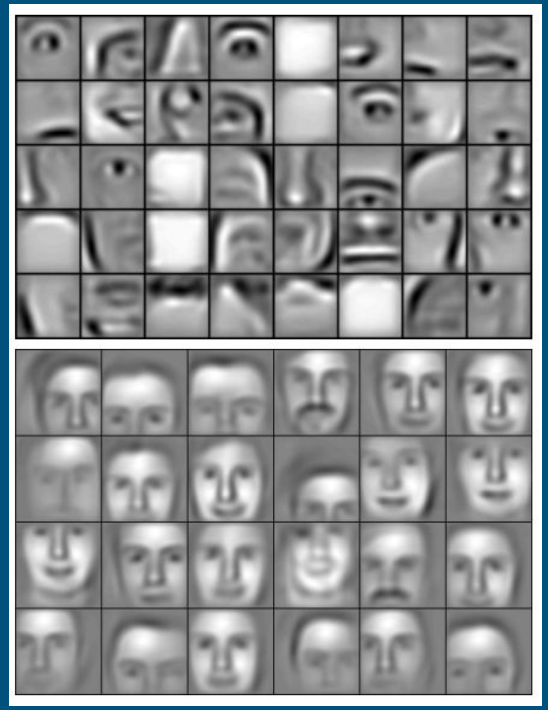


En la primer capa se detectan bordes .

En la segunda capa aparecen figuras mas complejas como puntos o esquinas.

Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations

¿Qué ven las capas?

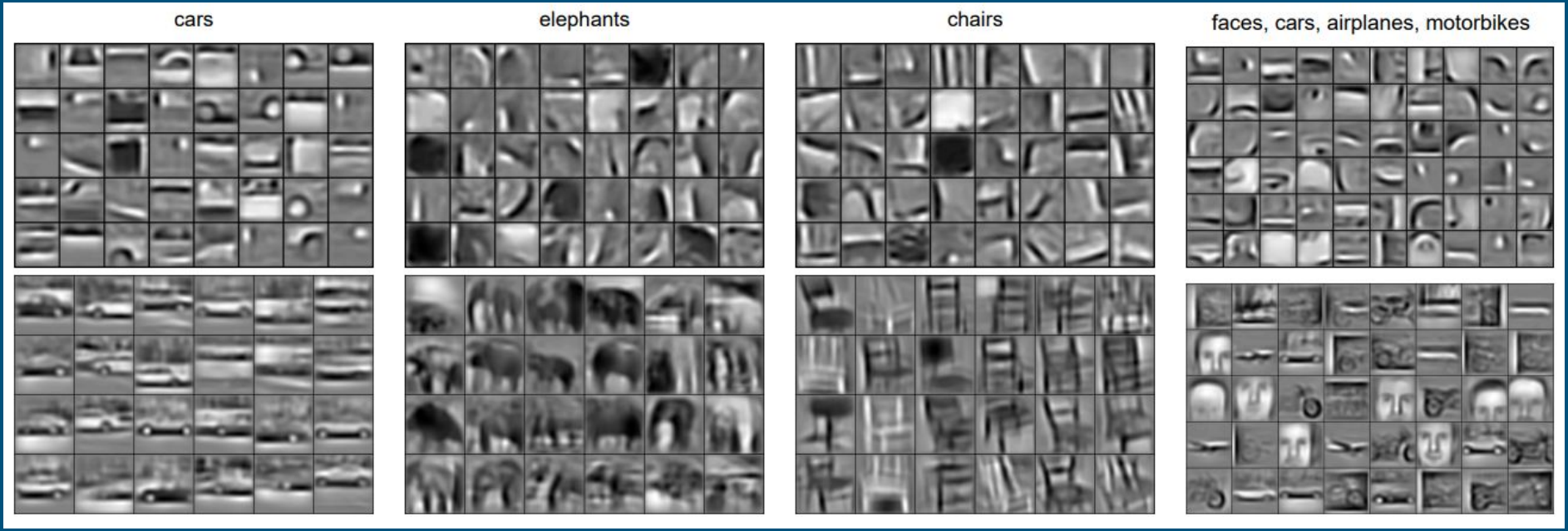


En la segunda capa se detectan partes de la cara.

En la tercer capa aparecen formas de caras enteras.

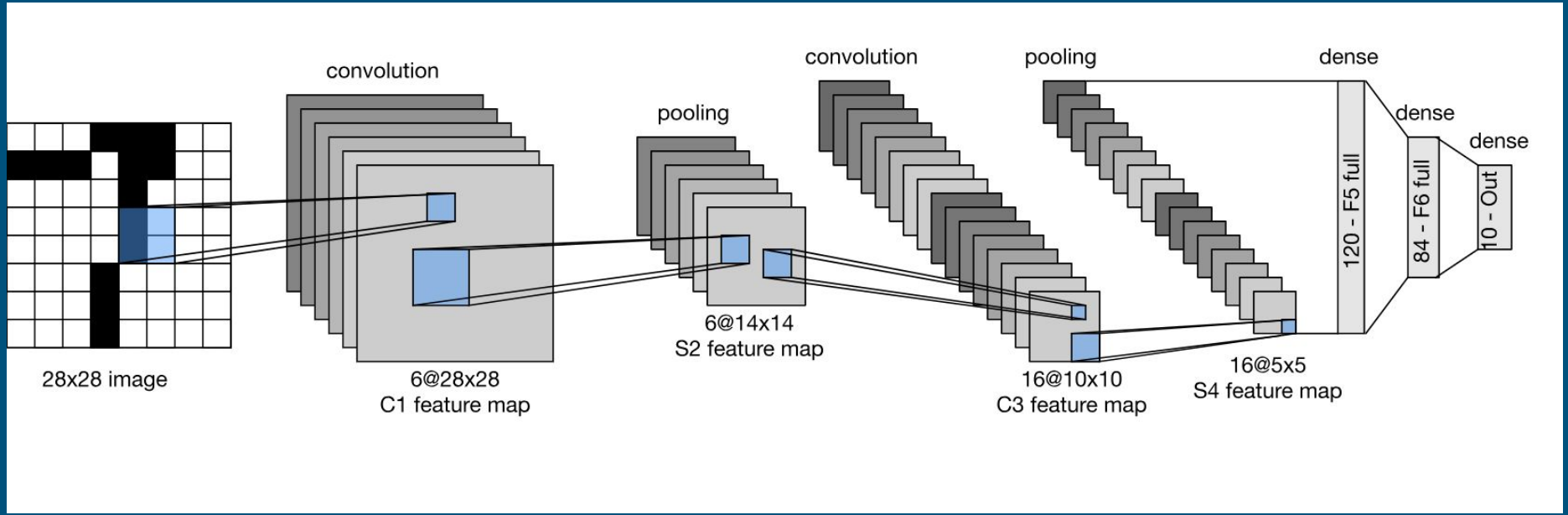
Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations

¿Qué ven las capas?



Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations

RED


convolution
pooling
convolution
pooling
dense
dense
dense
120 - F5 full
84 - F6 full
10 - Out
28x28 image
6@28x28
C1 feature map
6@14x14
S2 feature map
16@10x10
C3 feature map
16@5x5
S4 feature map