

Aprendizaje automático

Redes neuronales convolucionales (ConvNet)

Contenidos

- Introducción
- Aprendizaje con redes neuronales
- Redes neuronales convolucionales



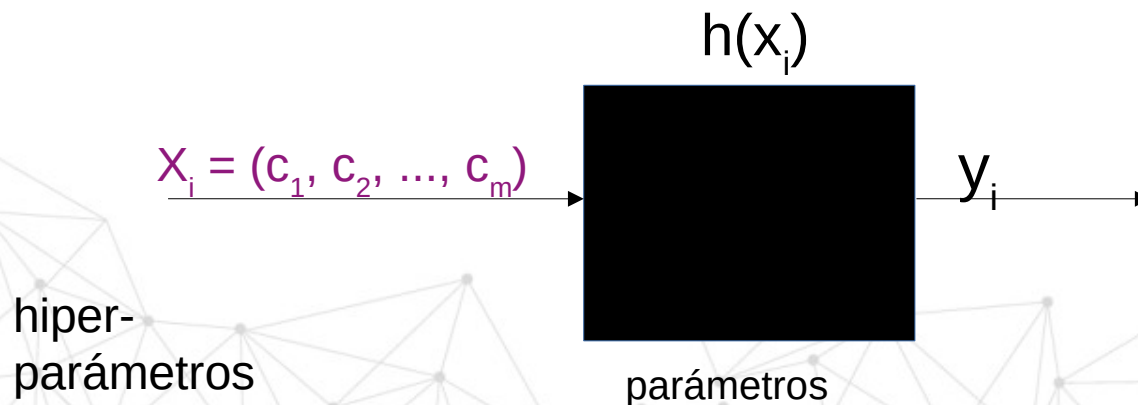
Aprendizaje supervisado

La tarea del aprendizaje supervisado es:

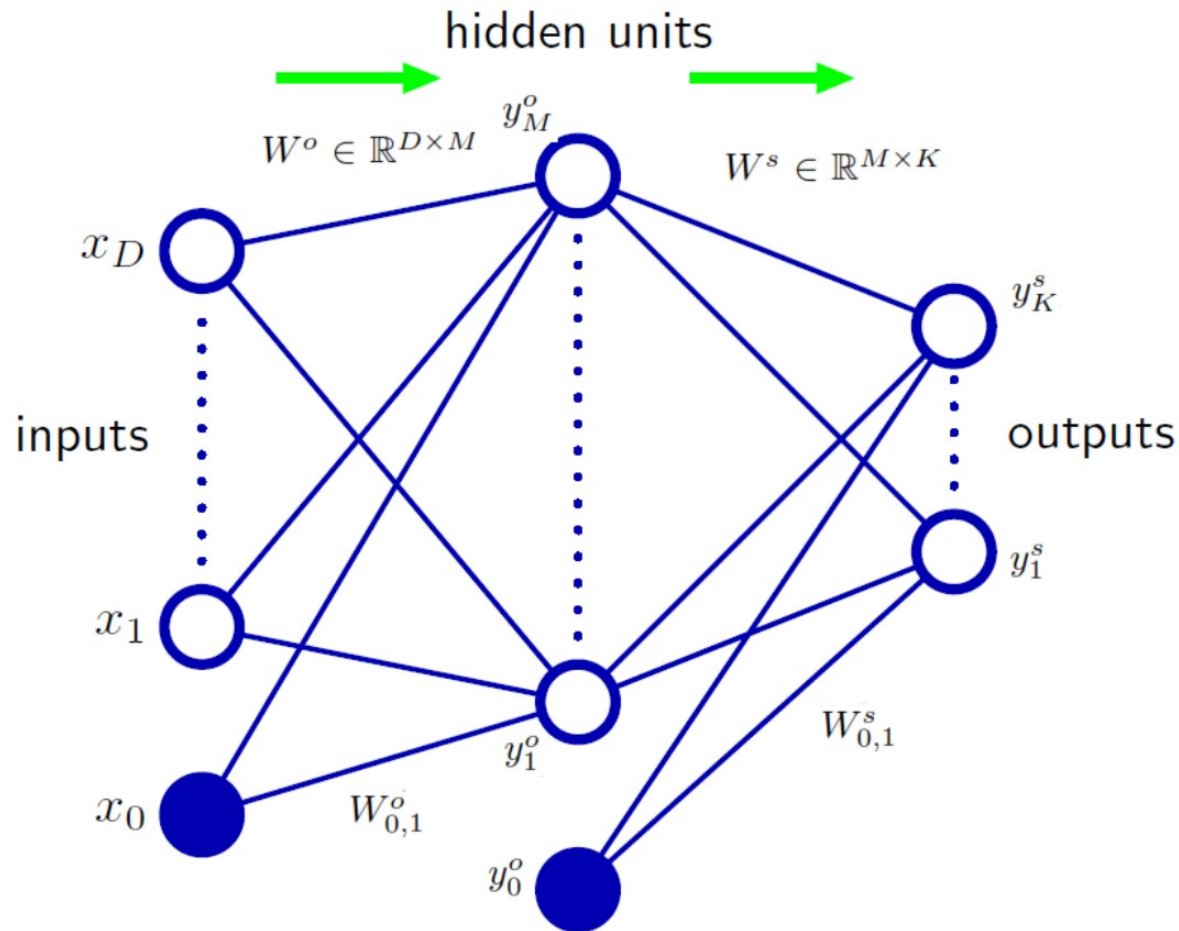
Dado un conjunto de entrenamiento con n ejemplos de la forma entrada/salida:

$$(x_1, y_1), (x_2, y_2), \dots (x_N, y_N)$$

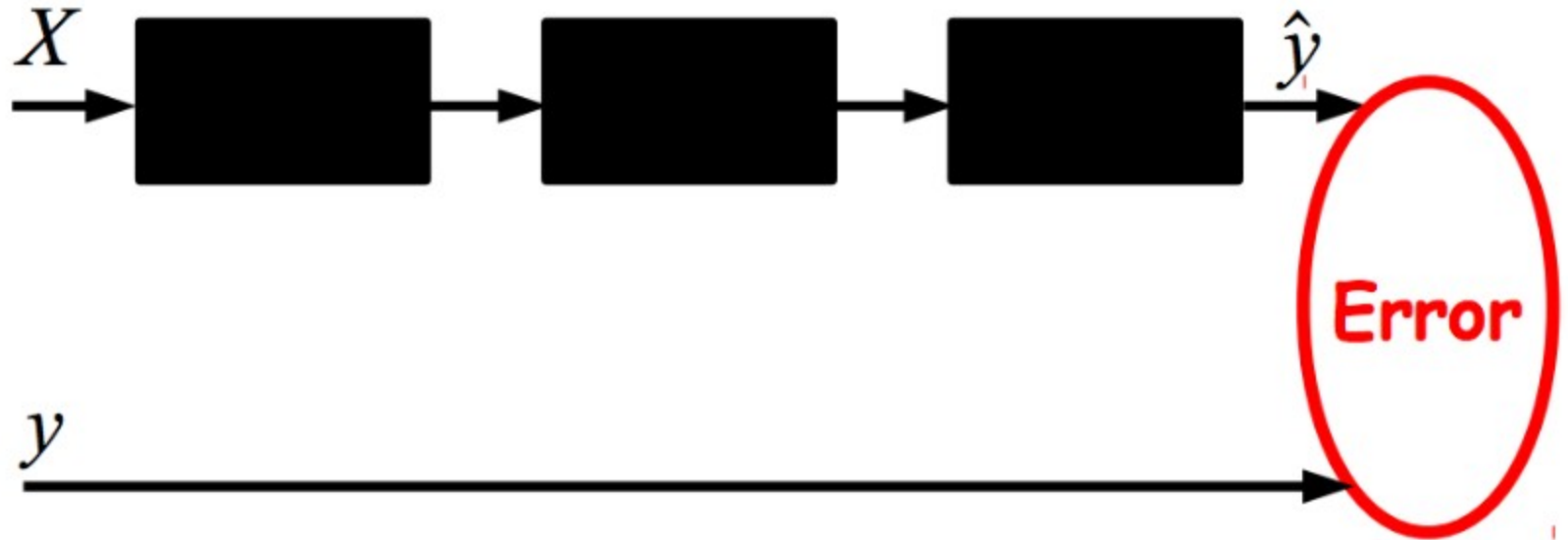
- Donde los y_i fueron generados por la función f (desconocida) con $f(x_i)=y_i$
- Hay que descubrir una función h que aproxime la función f .
- La función h se denomina **hipótesis**.



Perceptrón multicapa (MLP)



Entrenamiento de redes neuronales artificiales



Entrenamiento de redes neuronales artificiales

1) Se realizan corridas hacia adelante con pequeños grupos de ejemplos para calcular la pérdida.

F-PROP



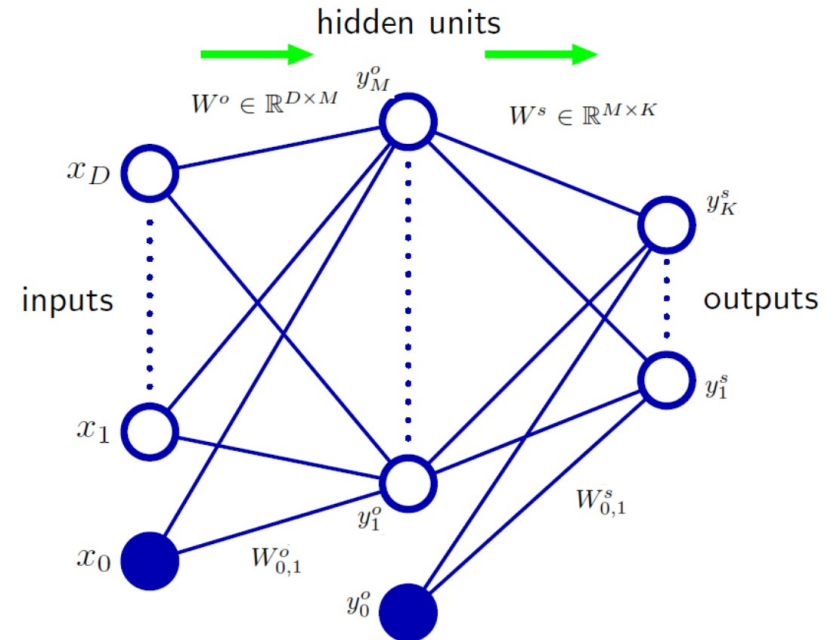
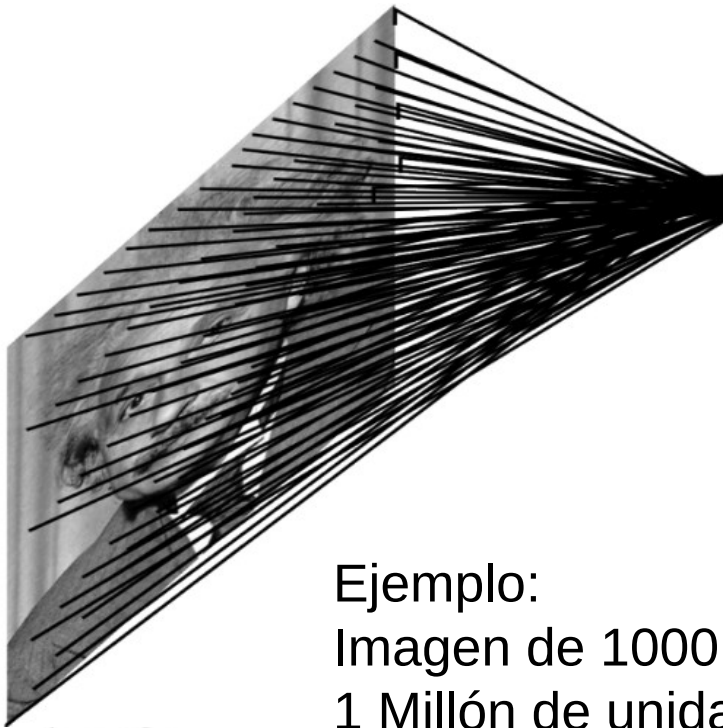
Entrenamiento de redes neuronales artificiales (ANN)

- 1) Calcula la pérdida utilizando pequeños grupos de ejemplos.
- 2) Se ajustan los pesos usando por ejemplo el algoritmo del descenso de gradiente.

B-PROP



Cuando la entrada es una imagen



Ejemplo:

Imagen de 1000 x 1000

1 Millón de unidades de entrada

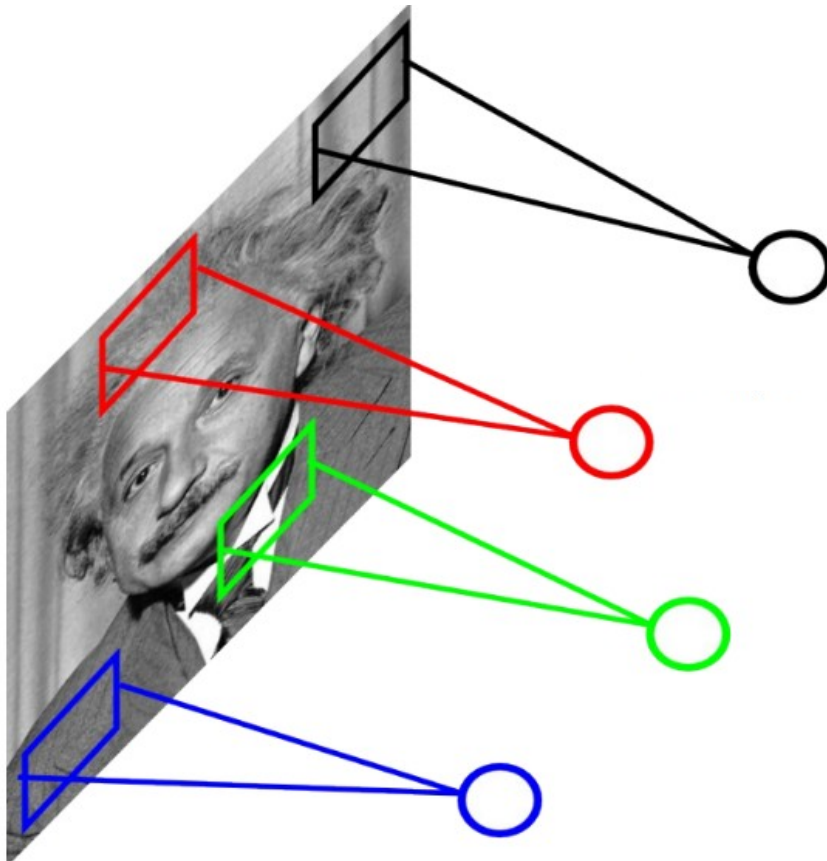
Con 100 unidades en la capa oculta

=> 100 millones de parámetros



Redes Neuronales Convolucionales (ConvNet)

Reducir el procesamiento a regiones locales



Ejemplo:

Imagen de 1000 x 1000

1 Millón de unidades ocultas



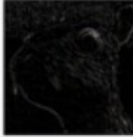



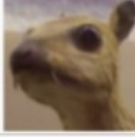
=> 1 billón de parámetros

Con filtro de 10x10 =>

Alrededor de 10 millones de
parámetros

Convolución para resaltar características en imágenes

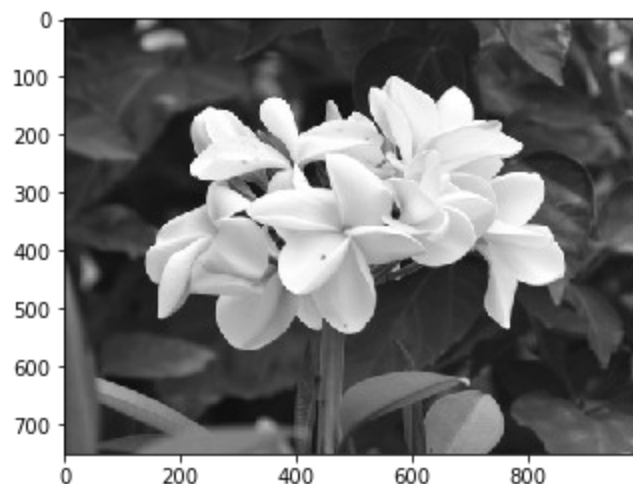
Ejemplos de filtros
o kernels

| Operation | Filter | Convolved Image |
|----------------------------------|--|---|
| Identity | $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ |  |
| Edge detection | $\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$ |  |
| | $\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ |  |
| | $\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$ |  |
| Sharpen | $\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$ |  |
| Box blur (normalized) | $\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ |  |
| Gaussian blur (approximation) | $\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$ |  |

Convolución para resaltar características en imágenes

Ejemplo de aplicación de un filtro

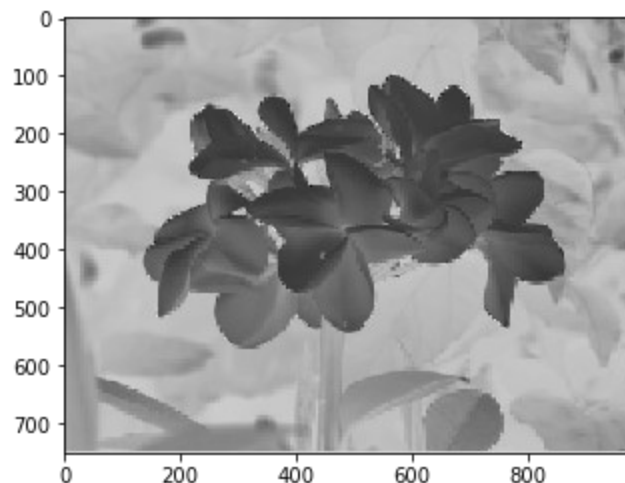
Mapa de características



Entrada



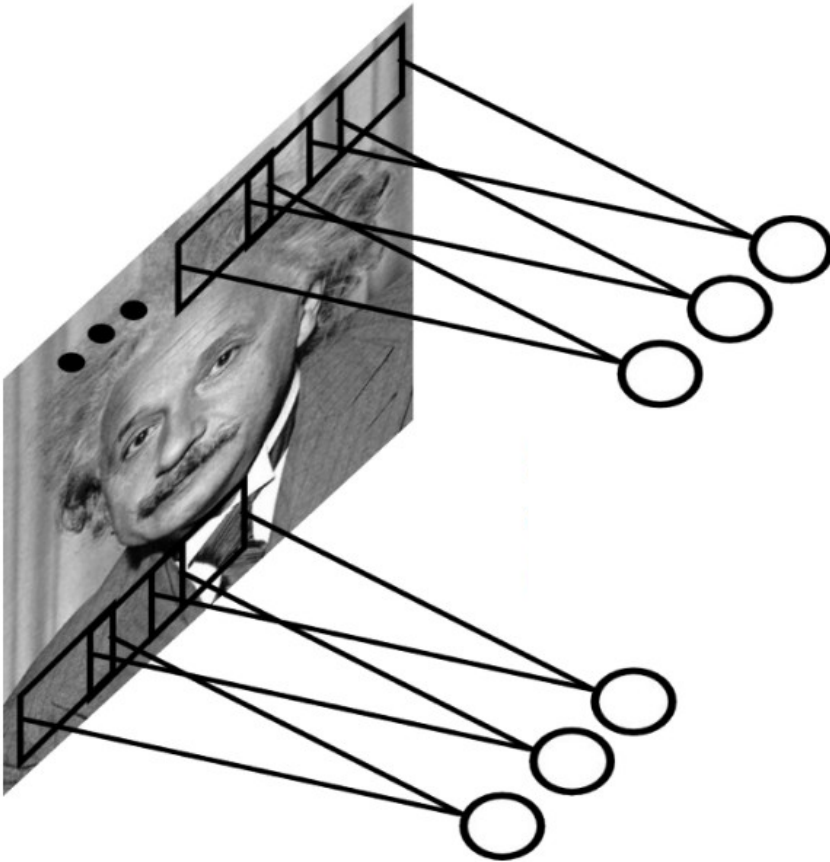
Filtro
[[-1,-1,-1],
[-1, 5,-1],
[-1,-1,-1]]



Filtro
[[-1,-1,-1],
[-1, 7,-1],
[-1,-1,-1]]



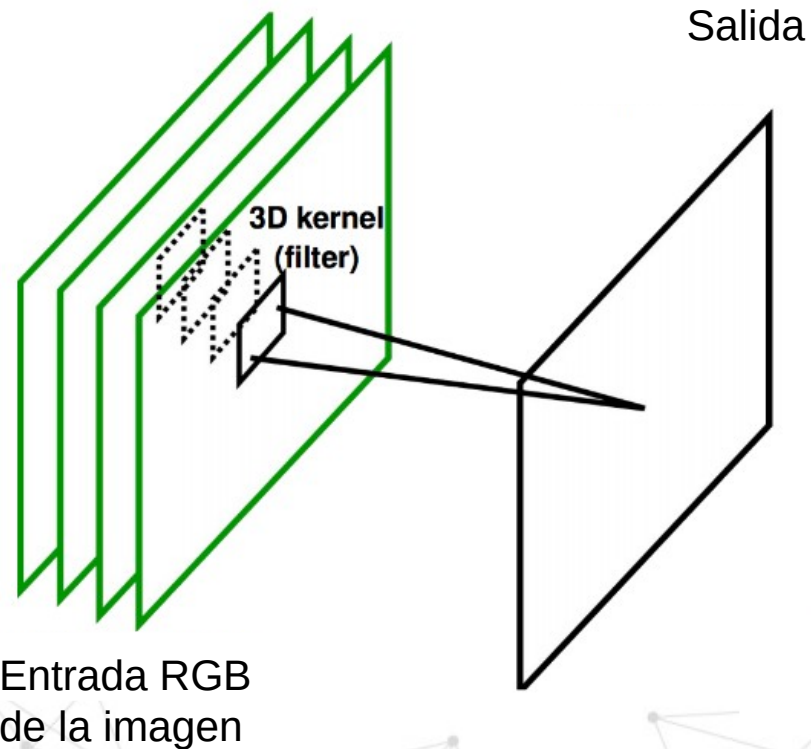
Redes Neuronales Convolucionales



- Caracteres interesantes, por ejemplo bordes, pueden estar en cualquier lugar de la imagen.
- Se comparten los mismos parámetros a través de toda la imagen

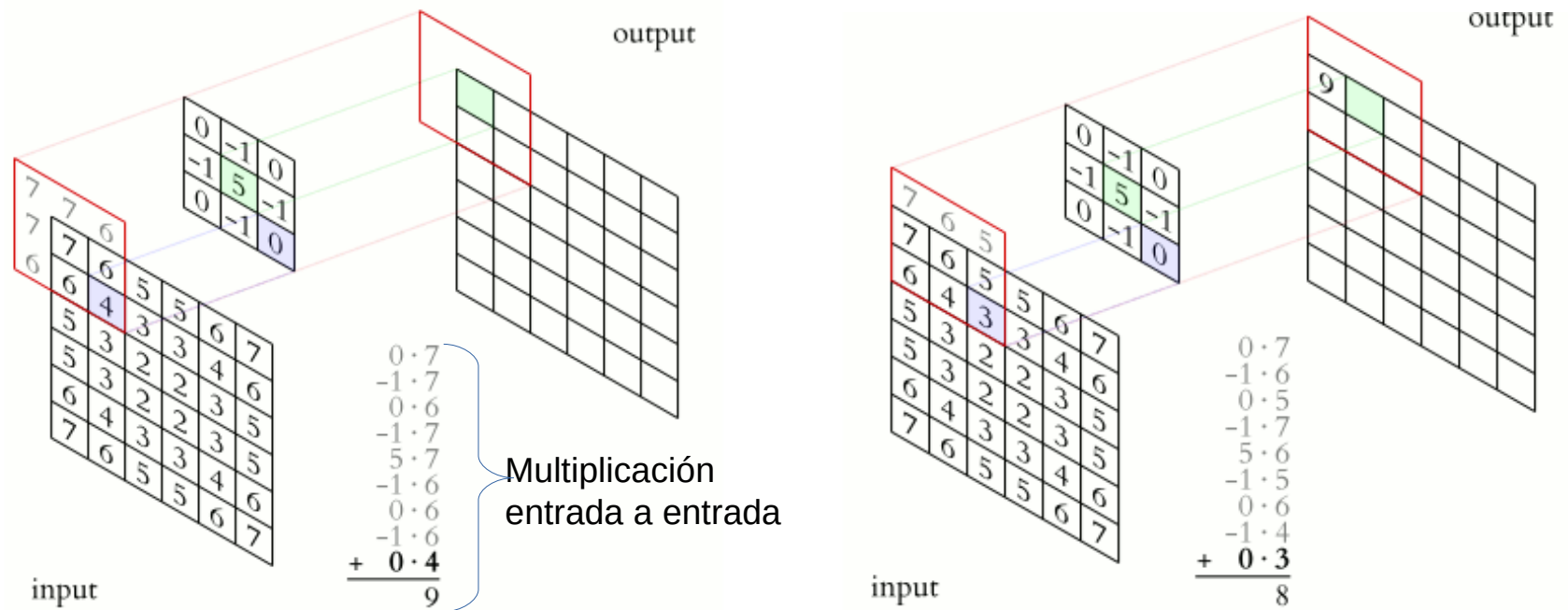
Redes Neuronales Convolucionales

Procesamiento de la imagen de entrada



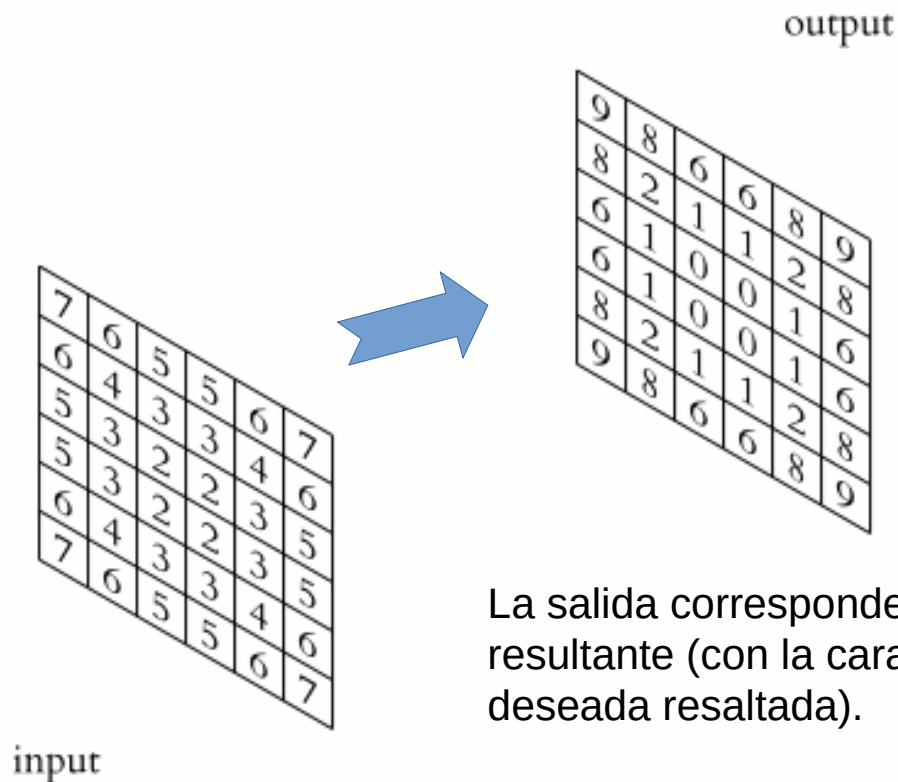
- Si la entrada tiene 3 canales (R, G, B), se aplican 3 filtros separados, de tamaño k por k , a cada canal.
- La salida se denomina mapa de características

Convolución en 2D



La operación de convolución realiza el **producto punto elemento a elemento** entre el filtro y las regiones de los datos de entrada.

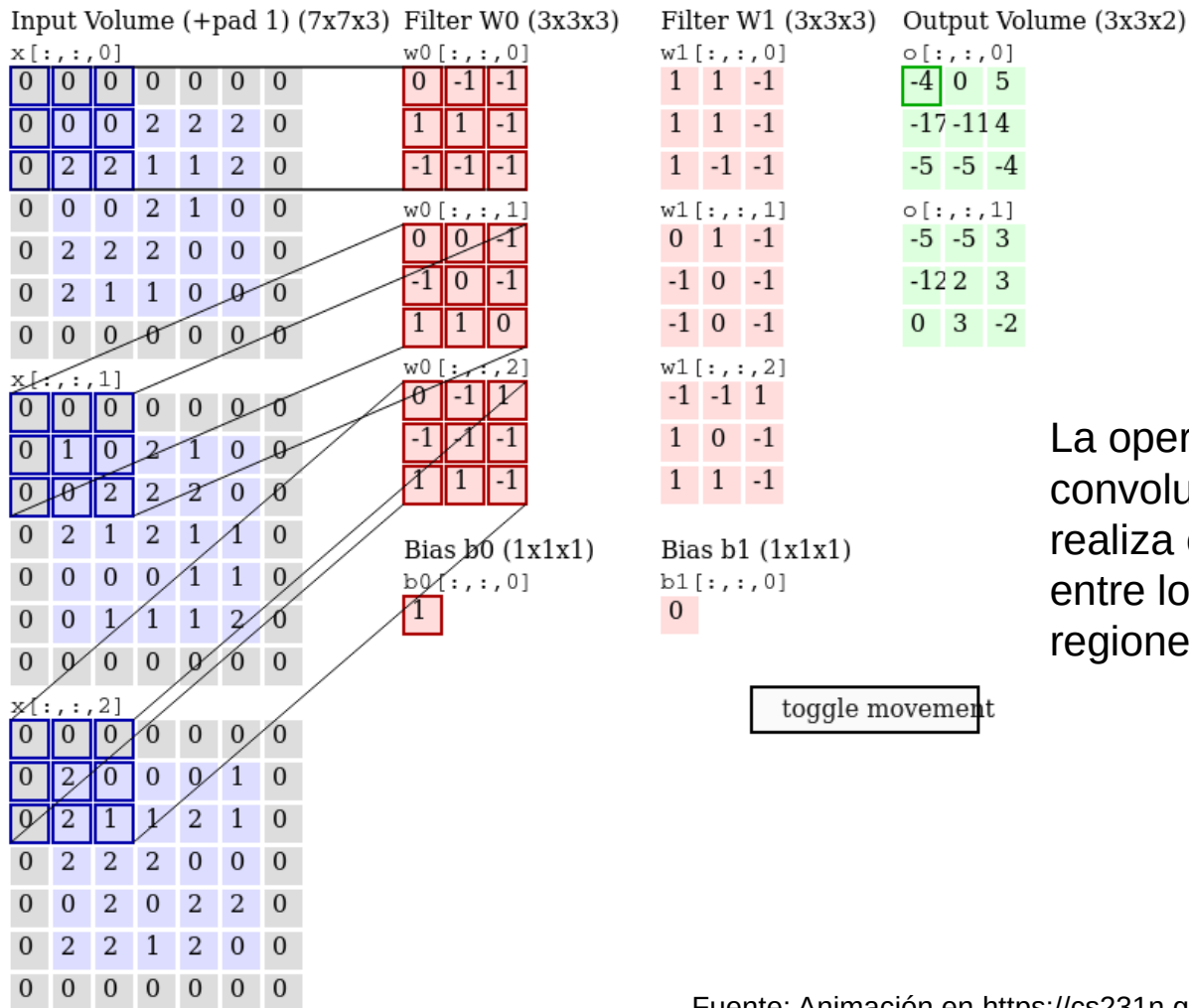
Convolución en 2D



La salida corresponde a la imagen resultante (con la característica deseada resaltada).

Redes Neuronales Convolucionales

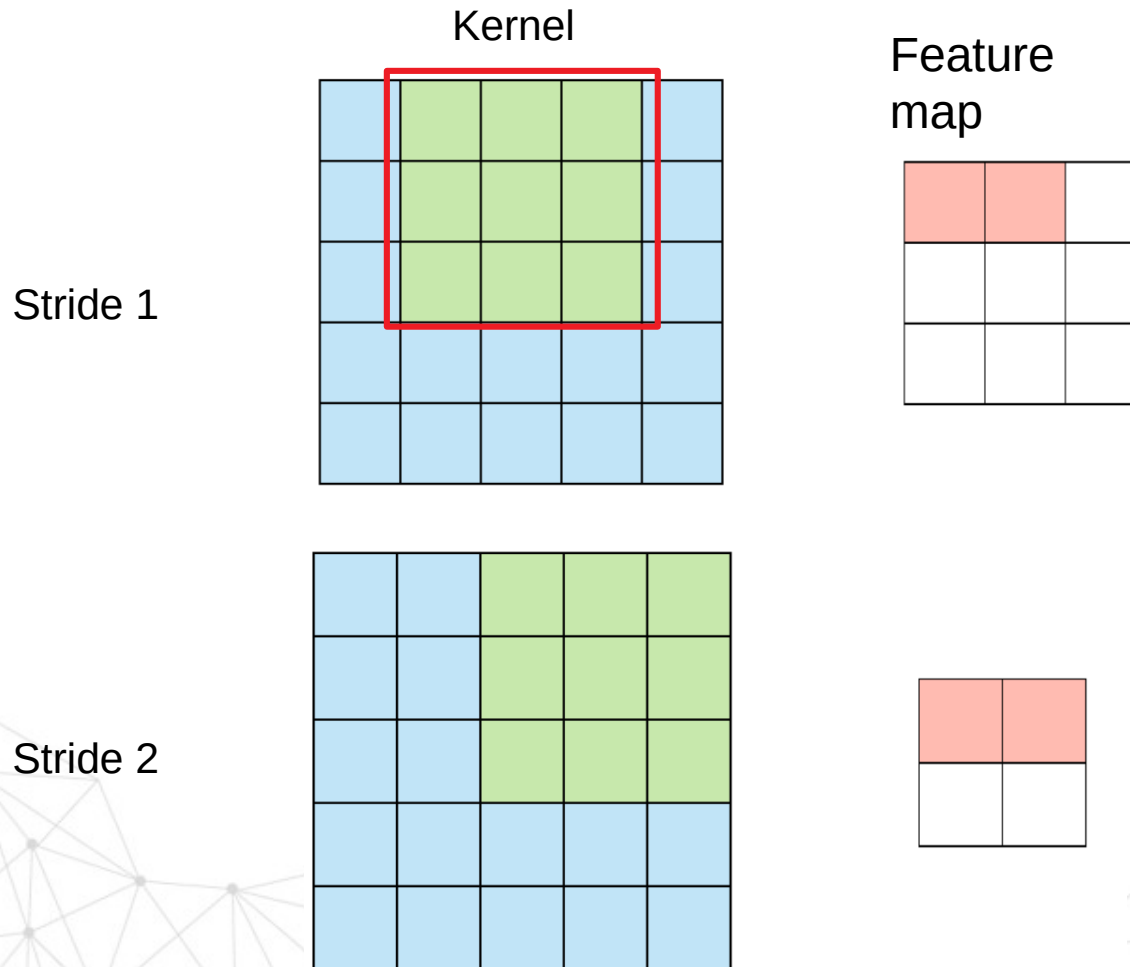
Procesamiento de la imagen de entrada



La operación de convolución esencialmente realiza el producto punto entre los filtros y las regiones de la entrada.

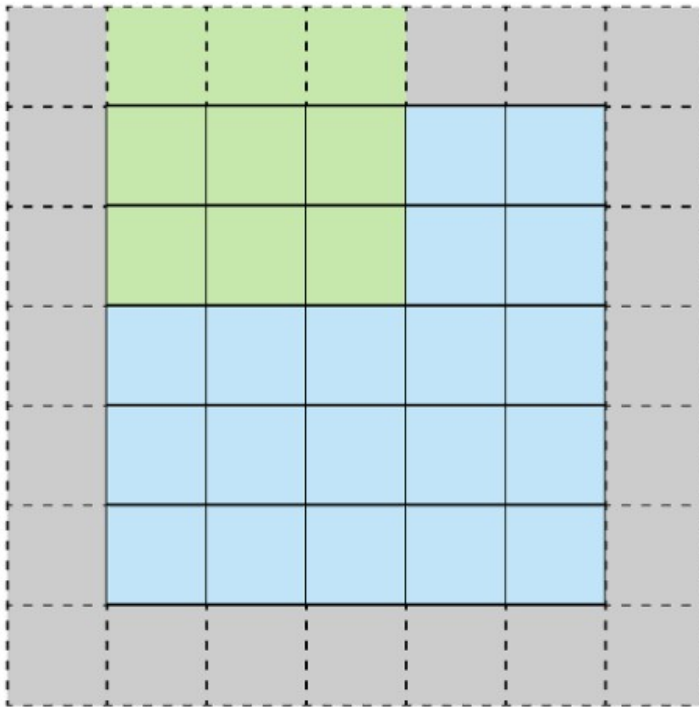
Redes Neuronales Convolucionales

Stride (paso, zancada)



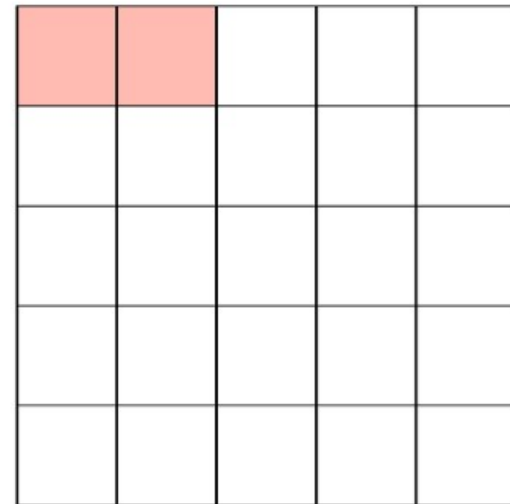
Redes Neuronales Convolucionales

Padding (relleno del borde)



Stride 1 con padding

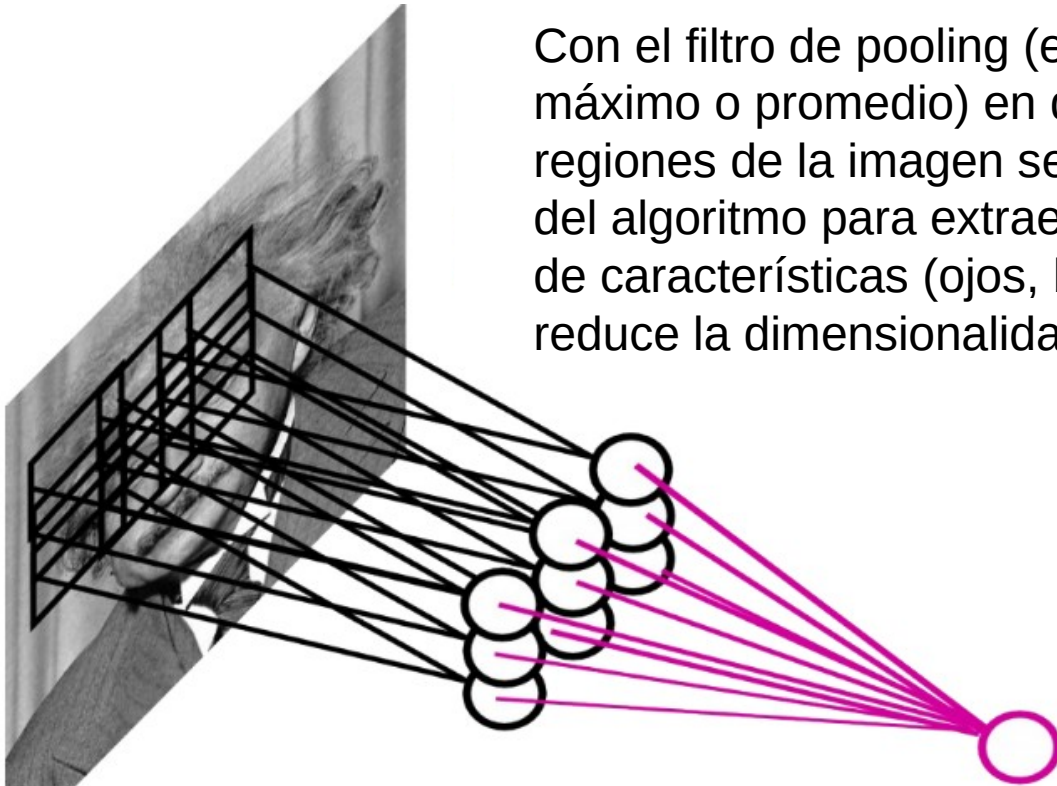
Feature map



Redes Neuronales Convolucionales

Pooling

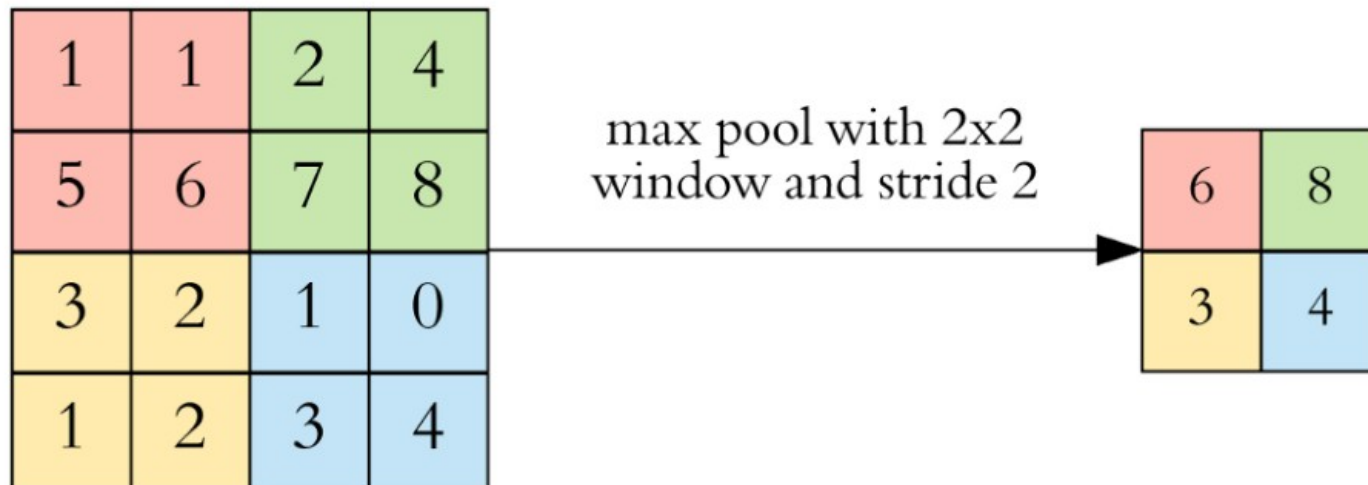
Con el filtro de pooling (ej. calculo del máximo o promedio) en diferentes regiones de la imagen se gana robustez del algoritmo para extraer la ubicación de características (ojos, boca, etc) y se reduce la dimensionalidad.



Redes Neuronales Convolucionales

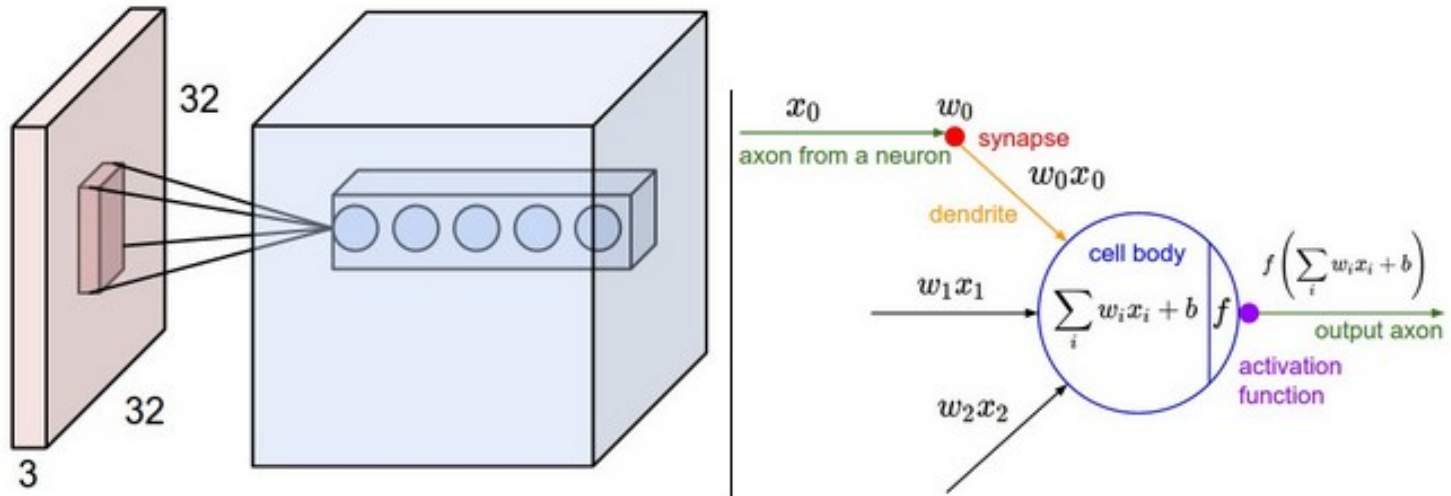
Pooling para reducir dimensionalidad

Max pooling



Redes Neuronales Convolucionales

Procesamiento de la imagen de entrada

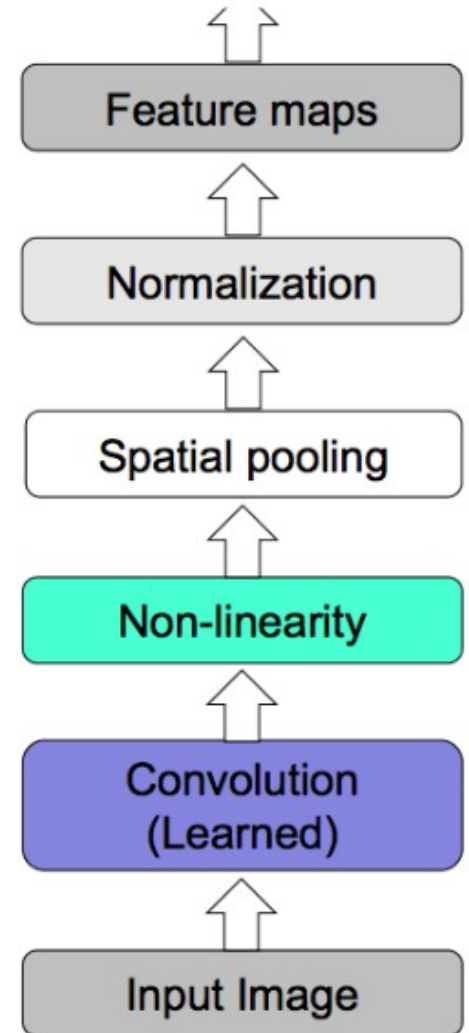


- La operación de convolución esencialmente realiza el producto punto entre los filtros y las regiones de la entrada.
- Un patrón de implementación común de la capa CONV es aprovechar este hecho y formular el *forward pass* de una capa convolucional como una gran matriz.

Redes Neuronales Convolucionales

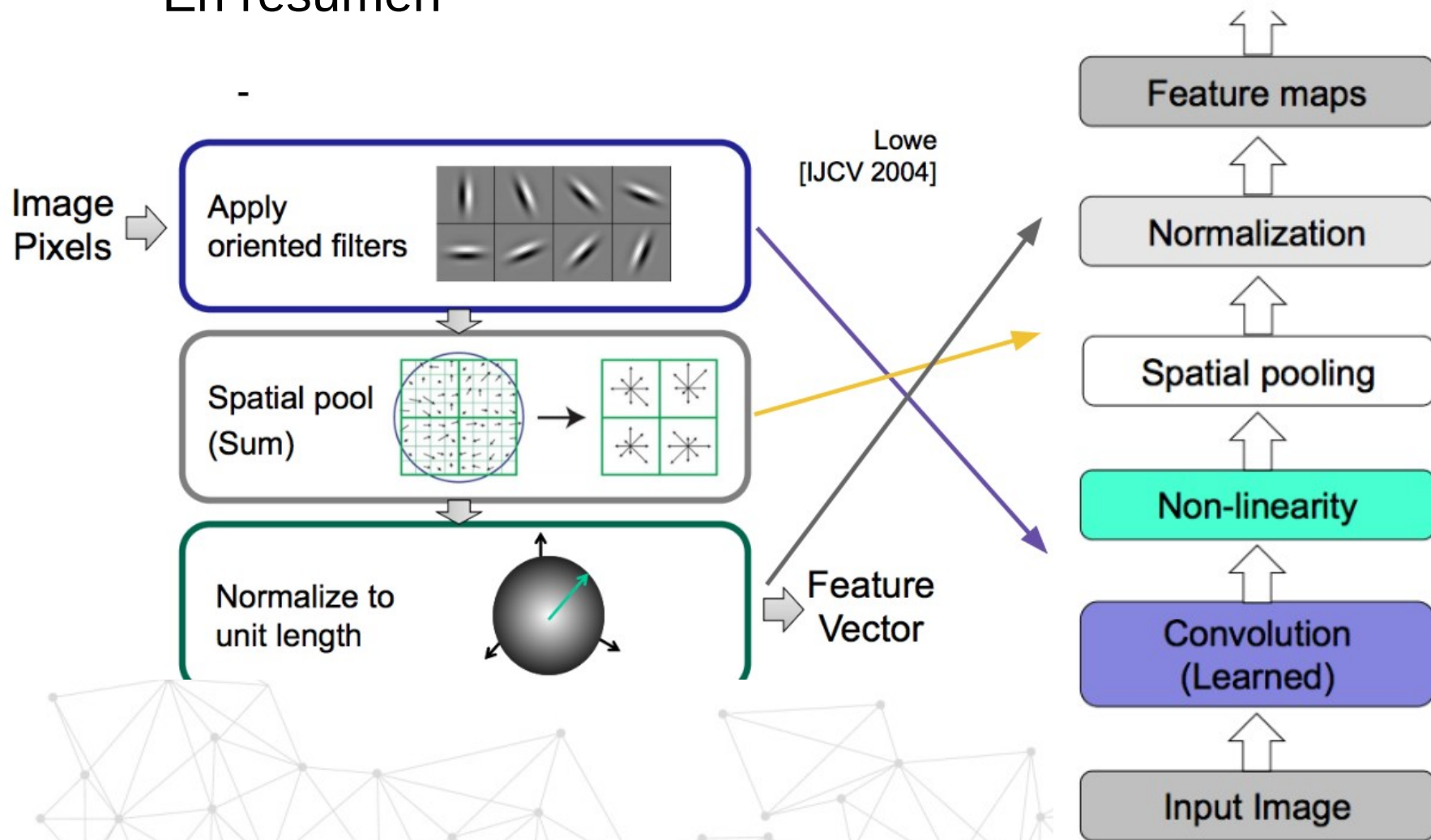
En resumen

- Todo lo anterior se realiza a la capa de entrada:
- Pooling y normalización es opcional.
- Los procesos se realizan en secuencia y se entrena la red como un perceptrón multicapa (MLP).
- Usualmente la última capa es un MLP cuya salida es igual al número de clases a separar.



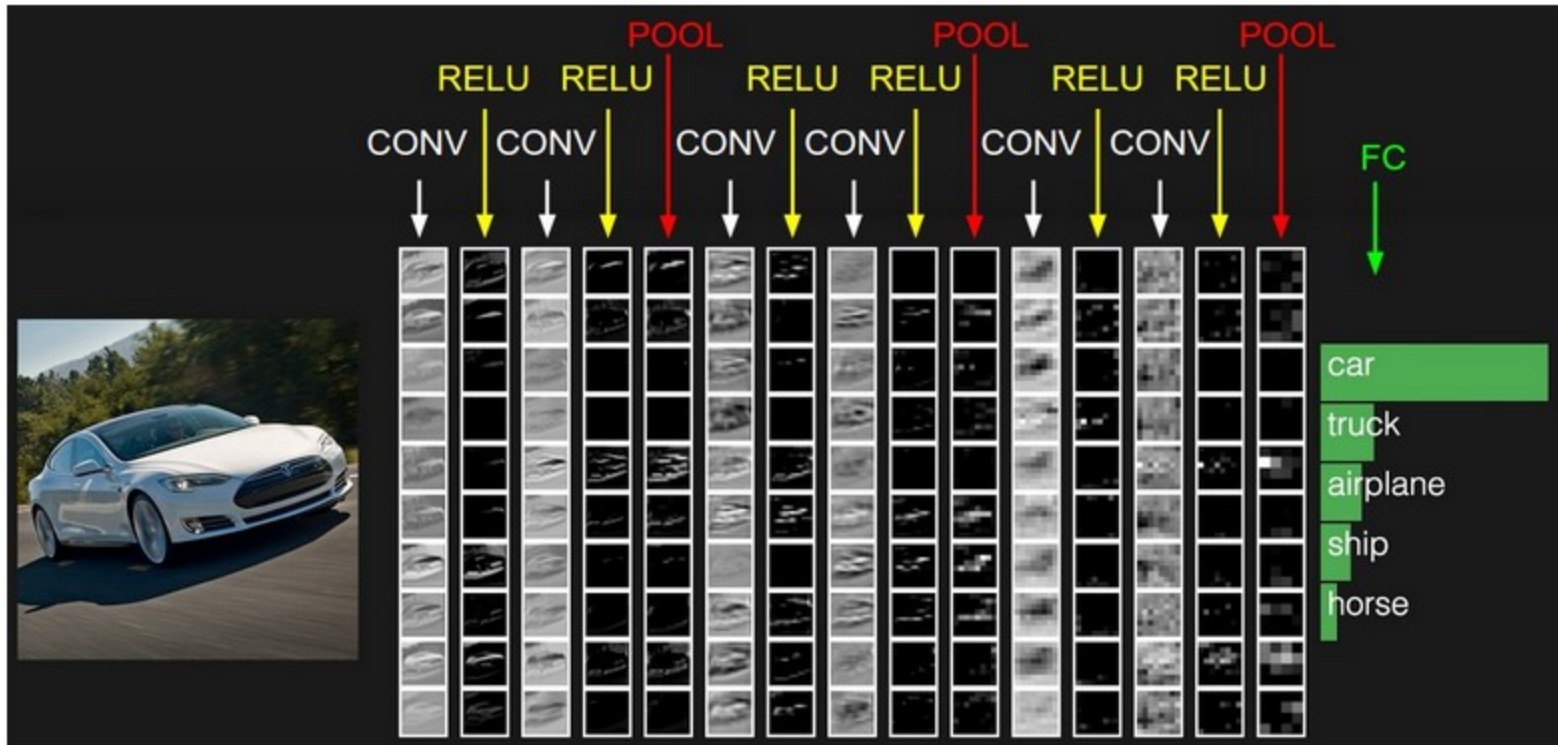
Redes Neuronales Convolucionales

En resumen



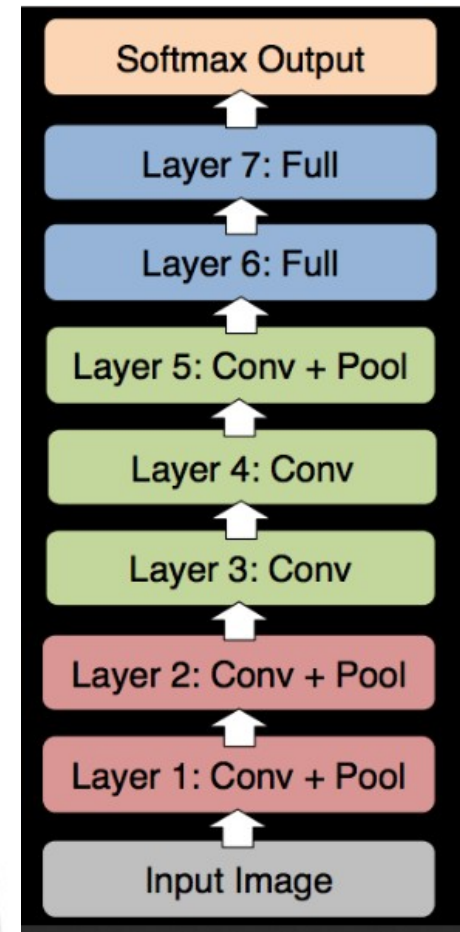
Redes Neuronales Convolucionales

- Cada capa aprende a detectar una característica combinando la salida de la capa anterior.
- Cada vez se aprenden más características abstractas a medida que apilamos capas.



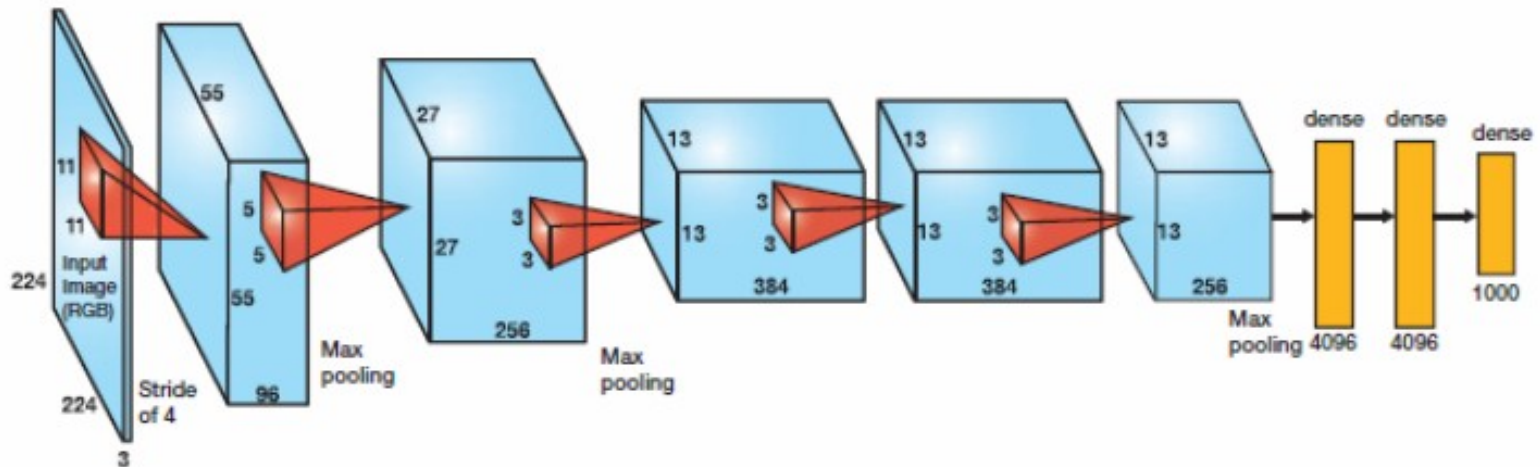
Ejemplo arquitectura de Alex Krizhevsky et al.

- El primer trabajo que popularizó las redes convolucionales en visión artificial.
- 8 capas en total
- Datos de entrenamiento: 1000 categorías, 1.2 millones de imágenes, 150 mil imágenes de prueba.
- Obtuvo el 18.2% de error y fue el ganador en el ILSVRC-2012*.

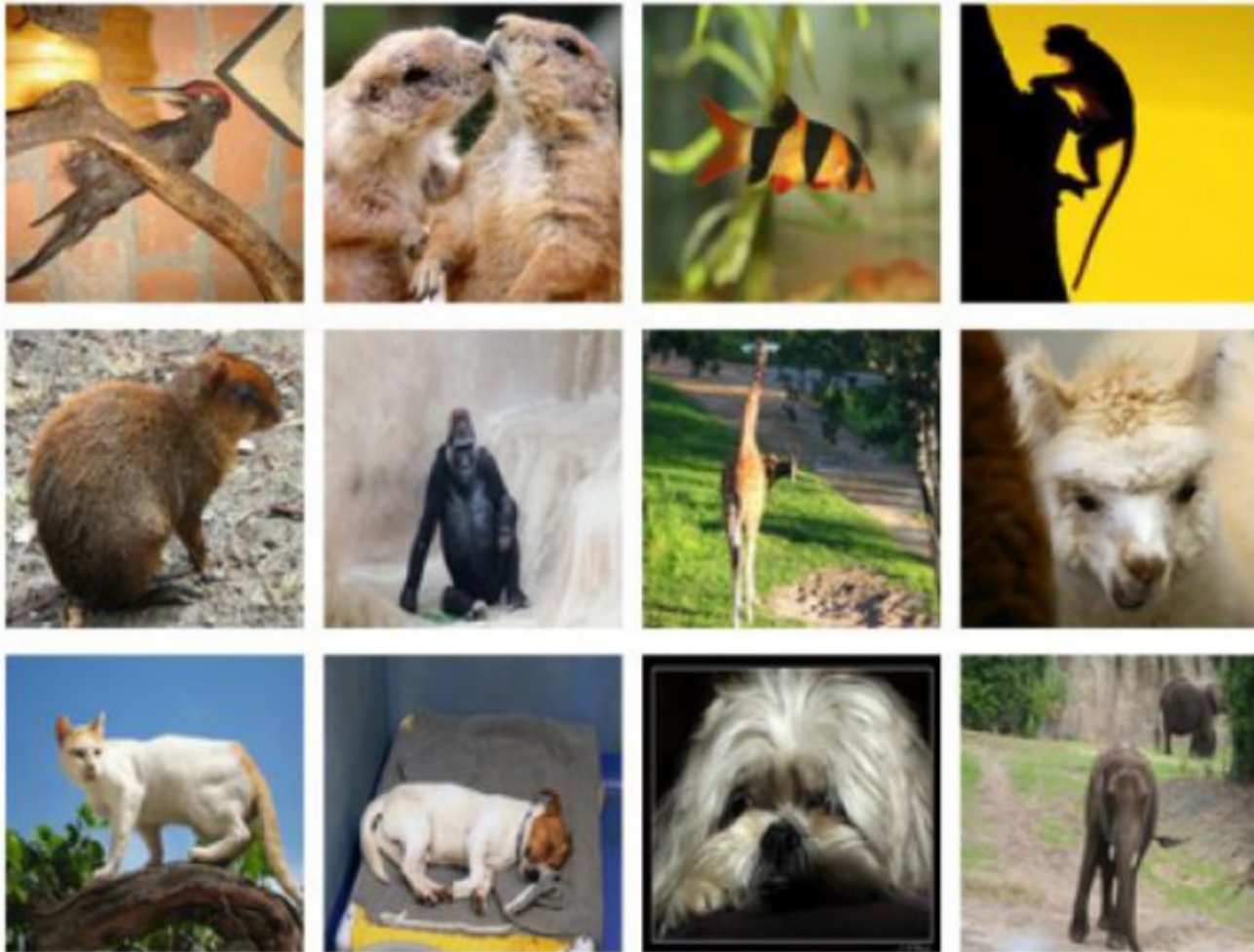


* ILSVRC: Large Scale Visual Recognition Competition

Ejemplo arquitectura de Alex Krizhevsky et al.



Ejemplo imágenes en la base de datos CIFAR utilizadas en Alex Krizhevsky et al.



Referencias

- Kanazawa, A(2015). Convolutional Neural Networks (Presentation). Recuperado de <http://www.cs.umd.edu/~djjacobs/CMSC733/CNN.pdf>
- Stanford University (2020). Convolutional Neural Networks for Visual Recognition. Recuperado de <https://cs231n.github.io/convolutional-networks/>
- Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. Nature, 521(7553):436, 2015.
- Christopher M Bishop (2006). Pattern Recognition and Machine Learning. (S. Calderón, Trans.). Springer.
- Krizhevsky, Alex, IlyaSutskever, and Hilton Geoffrey (2012). Imagenet Classification with Deep Convolutional Neural Networks. Advanced in Neural Information Processing Systems. Recuperado de <https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>
- Bishop, C (2006). Pattern Recognition and Machine Learning. (S. Calderón, Trans.). Springer.

