

Prácticas de Visión por Computador

Grupo 2

Repasso de Aprendizaje Profundo y
Redes Neuronales Convolucionales

Pablo Mesejo

Universidad de Granada

Departamento de Ciencias de la Computación e Inteligencia Artificial



UNIVERSIDAD
DE GRANADA



Nota preliminar

- Esto no deja de ser un **repaso rápido de algunos conceptos** fundamentales.
- **Bajo ningún concepto sustituye a la teoría.**
 - El **material explicado en la parte teórica**, tanto en fondo como en forma, será el objeto de estudio para el **segundo cuestionario**, y no estas diapositivas.

Bibliografía recomendada

Deep Learning

An MIT Press book

Ian Goodfellow and Yoshua Bengio and Aaron Courville

LeCun, Y., Bengio, Y., & Hinton, G. (2015). *Deep learning*. Nature, 521(7553), 436-444.

Schmidhuber, J. (2015). *Deep learning in neural networks: An overview*. Neural networks, 61, 85-117.

Bengio, Y.; Courville, A.; Vincent, P. (2013). *Representation Learning: A Review and New Perspectives*. IEEE Transactions on Pattern Analysis and Machine Intelligence. 35 (8): 1798–1828.

Bengio, Y. (2009). *Learning deep architectures for AI*. Foundations and Trends in Machine Learning, 2(1), 1-127.

Recursos online recomendados

“Neural Networks for Machine Learning” (curso impartido por Geoffrey Hinton, University of Toronto/Coursera):

<https://www.youtube.com/watch?v=cbeTc-Urqak>

“Deep Learning Specialization” (curso impartido por Andrew Ng, Stanford/deeplearning.ai):

https://www.youtube.com/watch?v=CS4cs9xVecg&list=PLkDaE6sCZn6Ec-XTbcX1uRg2_u4xOEky0

“CS231n: Convolutional Neural Networks for Visual Recognition” (Stanford):

<http://cs231n.stanford.edu/>

“Dive into Deep Learning. Interactive deep learning book with code, math, and discussions” (A. Zhang, Z.C. Lipton, M. Li, A.J. Smola): <https://d2l.ai/>

“CS230 Deep Learning” (Stanford): <https://cs230.stanford.edu/lecture/>

Andrej Karpathy’s blog: <https://karpathy.github.io/>

Christopher Olah’s blog: <https://colah.github.io/> (especialmente para RNNs)

<https://distill.pub/>

1. Introducción

Enfoques

Simbólica/clásica

SI

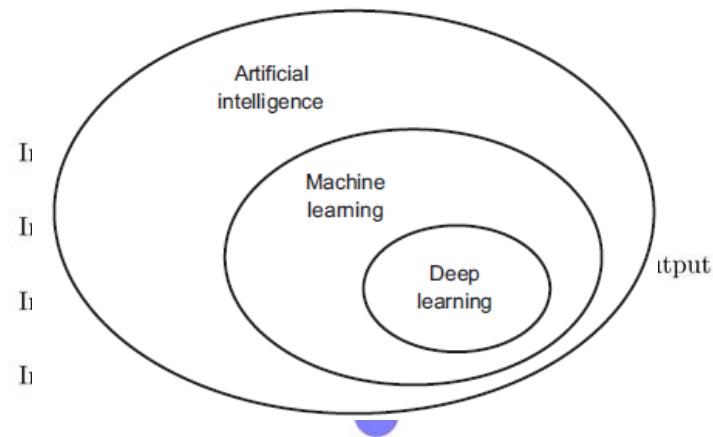
TARJETA	= verificada	y
FECHA	= no expirada	y
CÓDIGO	= correcto	y
INTENTOS	= no excedidos	y
BALANCE	= suficiente	y
LÍMITE	= no excedido	y
ENTONCES		
PAGO	= autorizado	

ENTONCES

Ejemplo: sistemas basados en reglas.

Se parte del conocimiento (información de alto nivel)

Sub-simbólica/conexionista

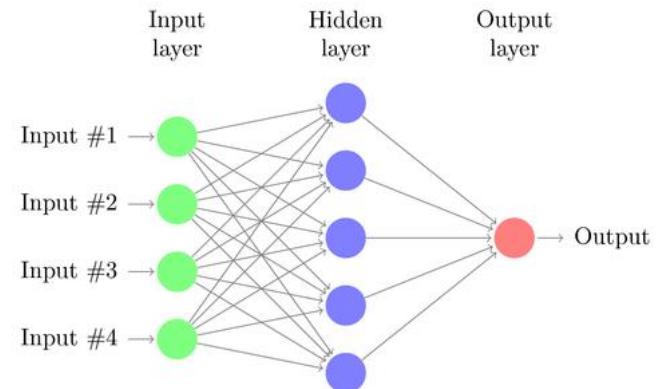


Ejemplo: red de neuronas artificial.

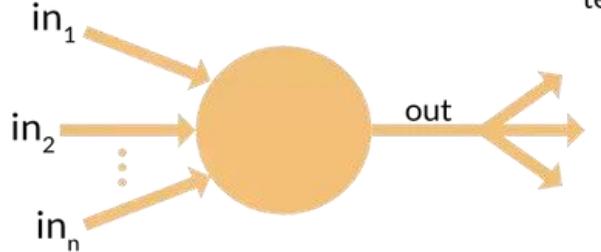
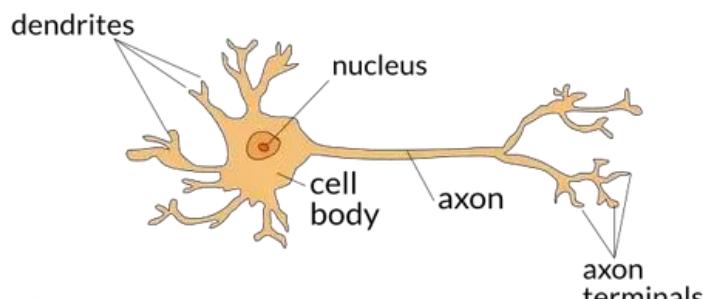
Se parte de los datos (información de bajo nivel)

Recordatorio de Redes Neuronales

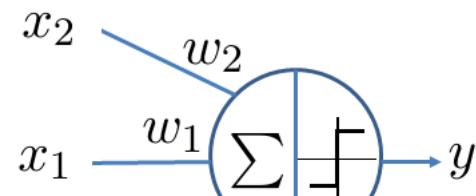
Conjunto de unidades simples de procesado altamente interconectadas y que, a través de un proceso de aprendizaje, resuelven una tarea concreta



¿Qué es una neurona artificial?



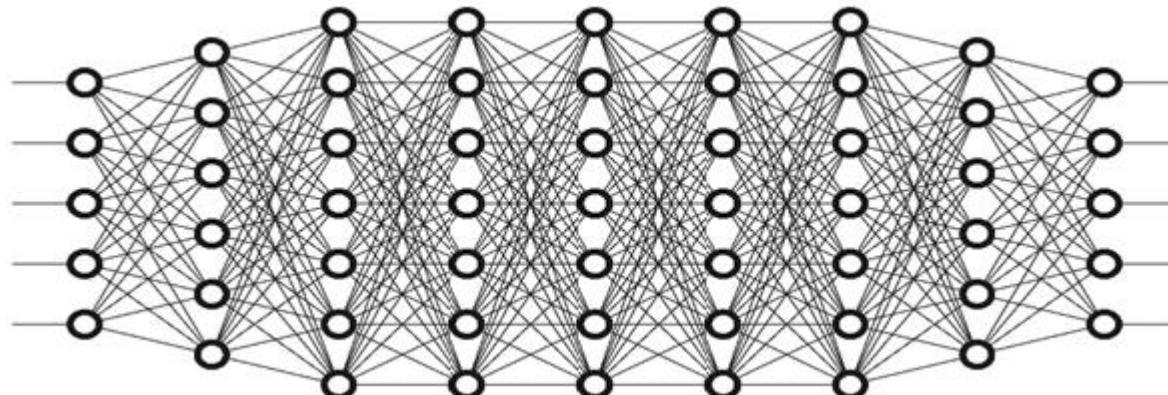
inputs weights activation function



$$x_2 w_2 + x_1 w_1 + w_0$$

¿Qué son las Redes Neuronales Profundas?

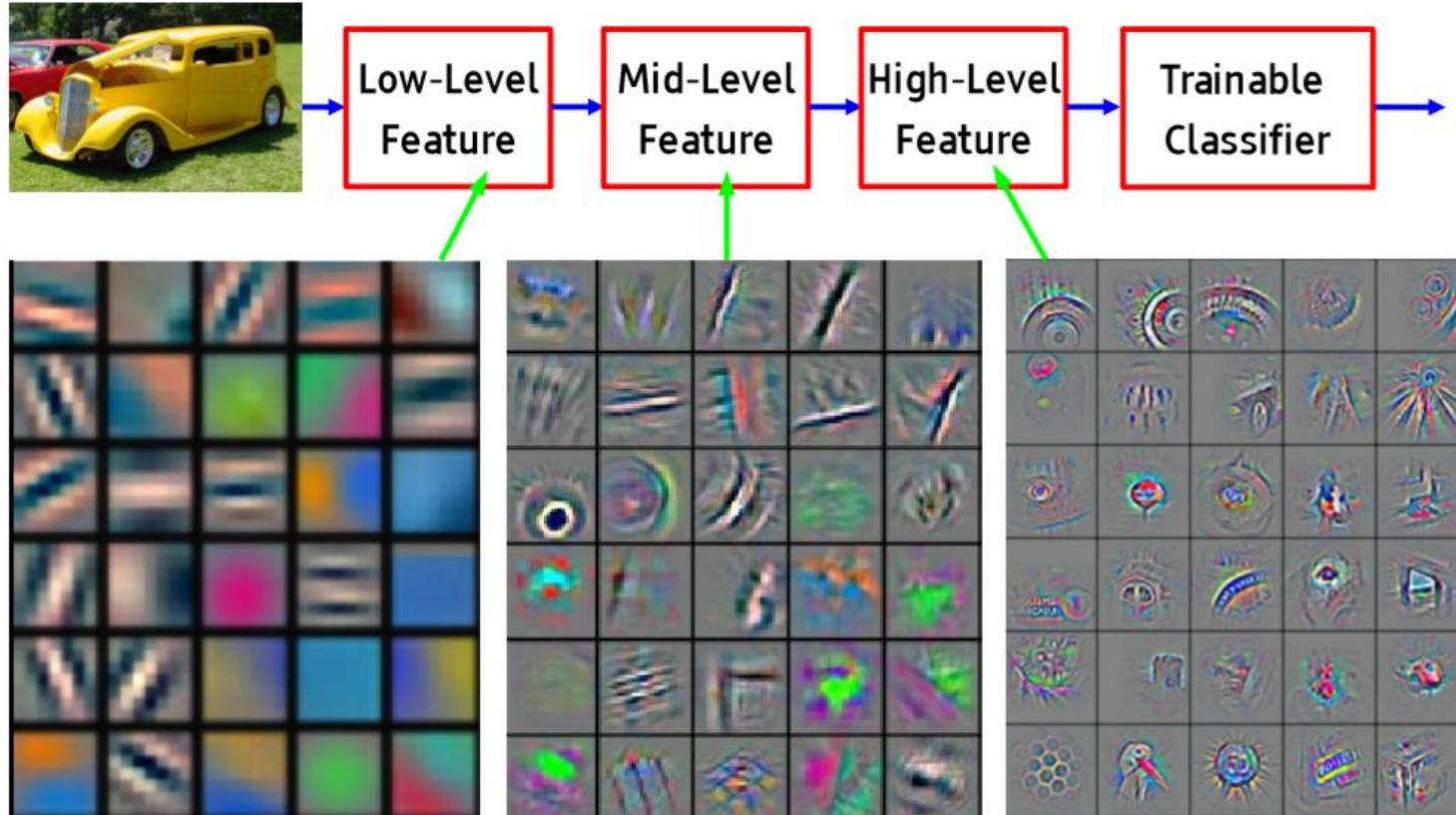
- Definición oficial
 - modelos computacionales, compuestos por numerosas capas de procesado, empleados para aprender representaciones con múltiples niveles de abstracción a partir de los datos de entrada
- En la práctica:
 - Redes de neuronas con “muchas” capas ocultas
 - Y que, por tanto, pueden aproximar funciones más complejas (es decir, resolver problemas más difíciles)



¿Qué son las Redes Neuronales Profundas?



■ It's deep if it has more than one stage of non-linear feature transformation



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

¿Qué son las Redes Neuronales Profundas?



François Chollet

@fchollet

Follow

"Neural networks" are a sad misnomer. They're neither neural nor even networks. They're chains of differentiable, parameterized geometric functions, trained with gradient descent (with gradients obtained via the chain rule). A small set of highschool-level ideas put together

11:58 AM - 12 Jan 2018

1,319 Retweets 3,423 Likes



114

1.3K

3.4K

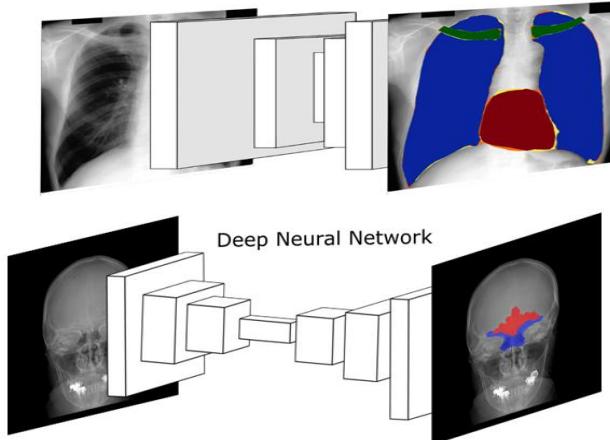


Aplicaciones (1)

Reconocimiento de Objetos en Imágenes

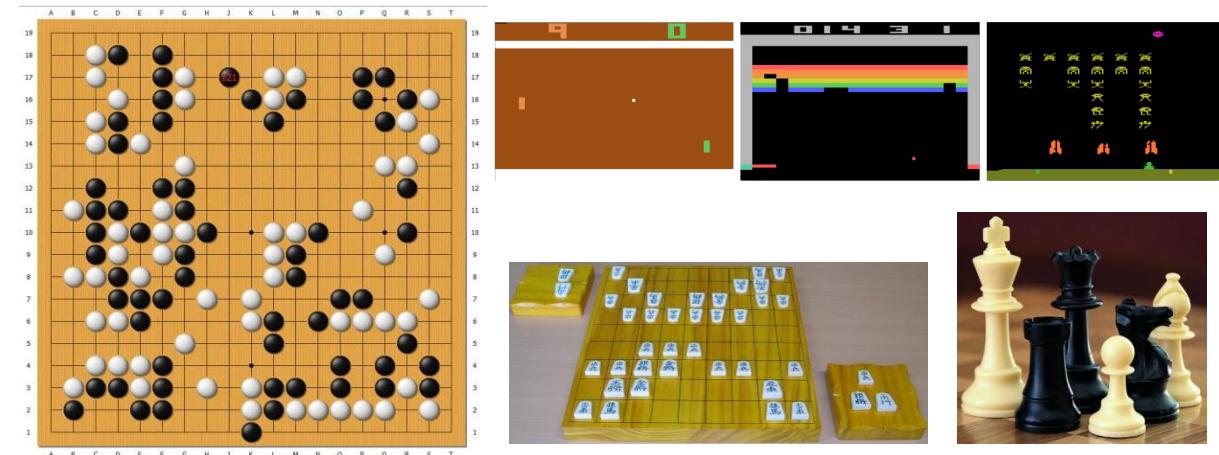


Mask R-CNN output from
<https://github.com/facebookresearch/Detectron>



"Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields" (CVPR'17)

Aprendizaje por Refuerzo en Juegos



Aplicaciones (2)

Reconocimiento Automático del Habla



Imagen extraída de <https://www.trzcacak.rs/imgb/iRoJhx/>

≡ Google Translate

Traducción Automática

The screenshot shows the Google Translate interface. At the top, there are tabs for "Text" and "Documents". Below that, the source language is set to "SPANISH" and the target language is "ENGLISH". The text to be translated is a political statement in Spanish:

El presidente de Estados Unidos, Donald Trump, "está usando el poder de su cargo para solicitar la interferencia de un país extranjero en las elecciones estadounidenses de 2020". Y la Casa Blanca trató de "bloquear" información para ocultarlo. Esas son las explosivas acusaciones que un denunciante anónimo, en base a información recibida de "múltiples oficiales del Gobierno de Estados Unidos", escribió en una queja dirigida a los presidentes de los Comités de Inteligencia de las dos cámaras del Congreso el pasado 12 de agosto.

The English translation is as follows:

The president of the United States, Donald Trump, "is using the power of his office to request interference from a foreign country in the 2020 US elections." And the White House tried to "block" information to hide it. Those are the explosive accusations that an anonymous whistleblower, based on information received from "multiple US government officials," wrote in a complaint addressed to the presidents of the Intelligence Committees of the two houses of Congress on August 12 .

At the bottom of the interface, there are icons for audio playback, a progress bar showing "531/5000", and sharing options.

Aplicaciones (3): transferencia de estilo



Imagen extraída de “A Neural Algorithm of Artistic Style” (Gatys et al., 2015)

Aplicaciones (4): colorización de imágenes

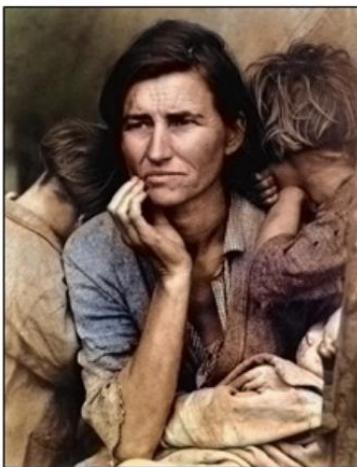
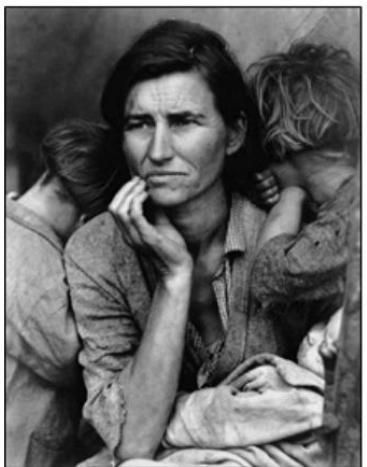


Imagen extraída de “Colorful Image Colorization” (Zhang et al., 2016)

Aplicaciones (5): reconstrucción de imágenes

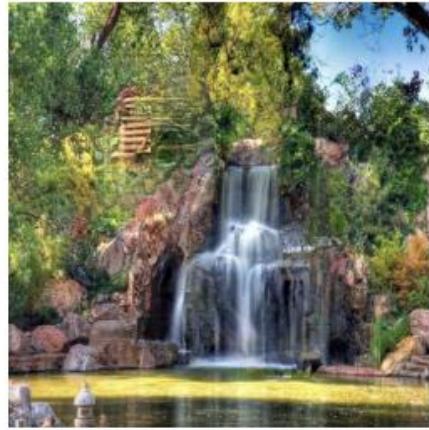
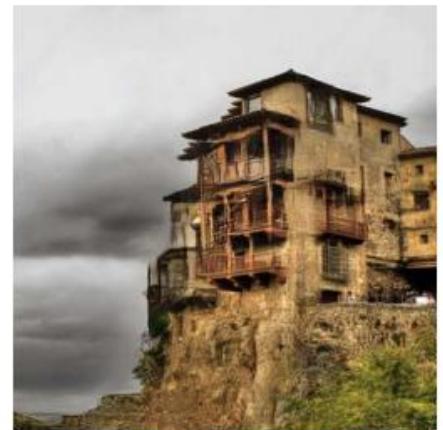
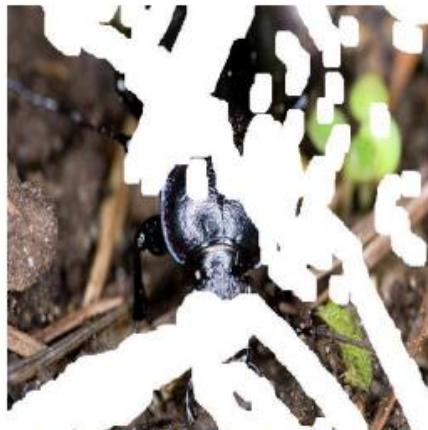


Imagen extraída de “Image Inpainting for Irregular Holes Using Partial Convolutions”
(Liu et al., 2018)

Aplicaciones (6): síntesis de imágenes



Imagen extraída de “Progressive Growing of GANs for Improved Quality, Stability, and Variation” (Karras et al., 2017)”

Aplicaciones (y 7): image captioning



"man in black shirt is playing guitar."



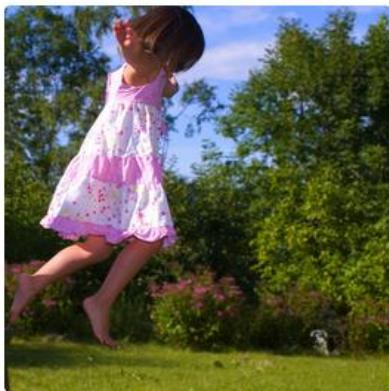
"construction worker in orange safety vest is working on road."



"two young girls are playing with lego toy."



"boy is doing backflip on wakeboard."



"girl in pink dress is jumping in air."



"black and white dog jumps over bar."



"young girl in pink shirt is swinging on swing."



"man in blue wetsuit is surfing on wave."

Karpathy, A., & Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3128-3137).

Evidentes éxitos y avances

Tan evidentes, que tres de los principales investigadores en el campo recibieron en 2018 el **Turing Award** (“Premio Nobel de la Informática”)



Yoshua Bengio
University of Montreal



Geoffrey Hinton
University of Toronto & Google



Yann LeCun
New York University & Facebook

Evidentes éxitos y avances

MIT
Technology
Review

10 Breakthrough Technologies 2013

Deep Learning

With massive amounts of computational power, machines can now recognize objects and translate speech in real time. Artificial intelligence is finally getting smart.

¿Por qué ahora?

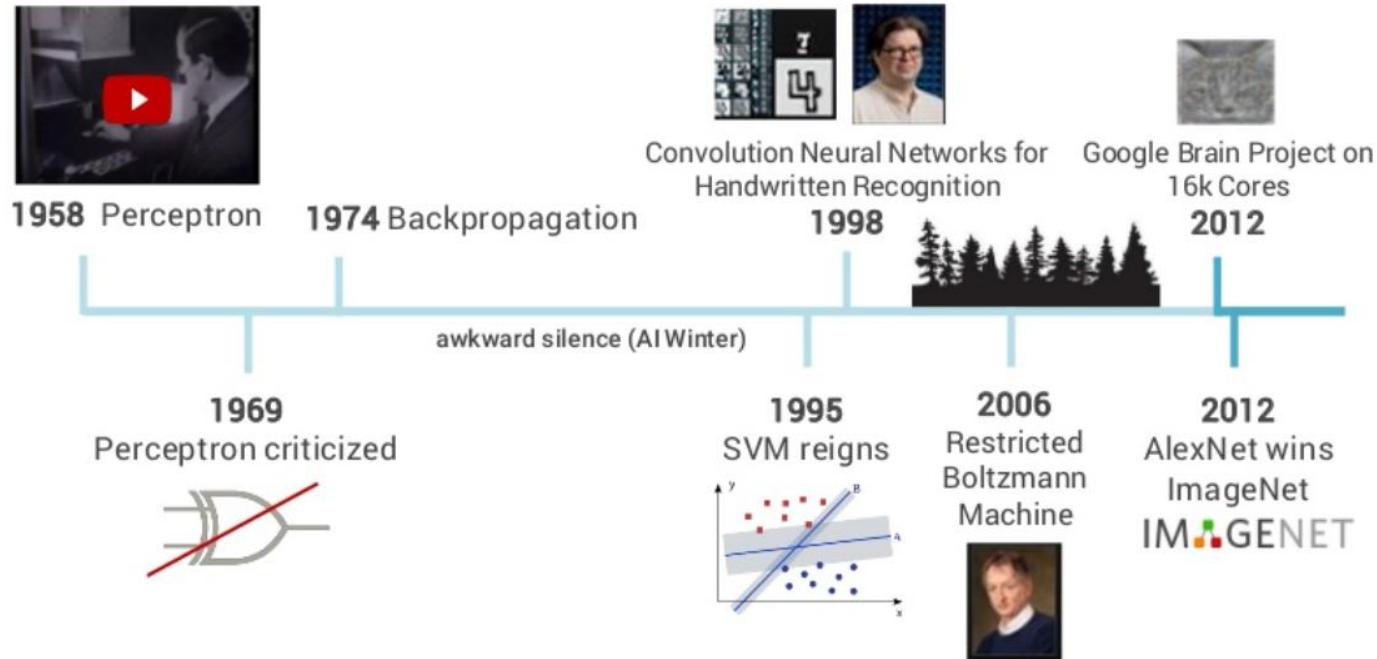


Imagen extraída de <https://www.slideshare.net/LuMa921/deep-learning-a-visual-introduction>

- Explosión desde 2012 debida a:
 - Enormes cantidades de datos etiquetados (p.ej. ImageNet)
 - Gran capacidad de cálculo (p.ej. GPU-computing)
 - Avances algorítmicos y arquitecturales (p.ej. ReLU, Dropout, Pretraining)

Principales nombres



Geoffrey Hinton: University of Toronto & Google



Yann LeCun: New York University & Facebook



Andrew Ng: Stanford & Baidu

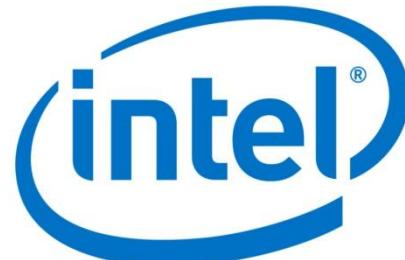


Yoshua Bengio: University of Montreal

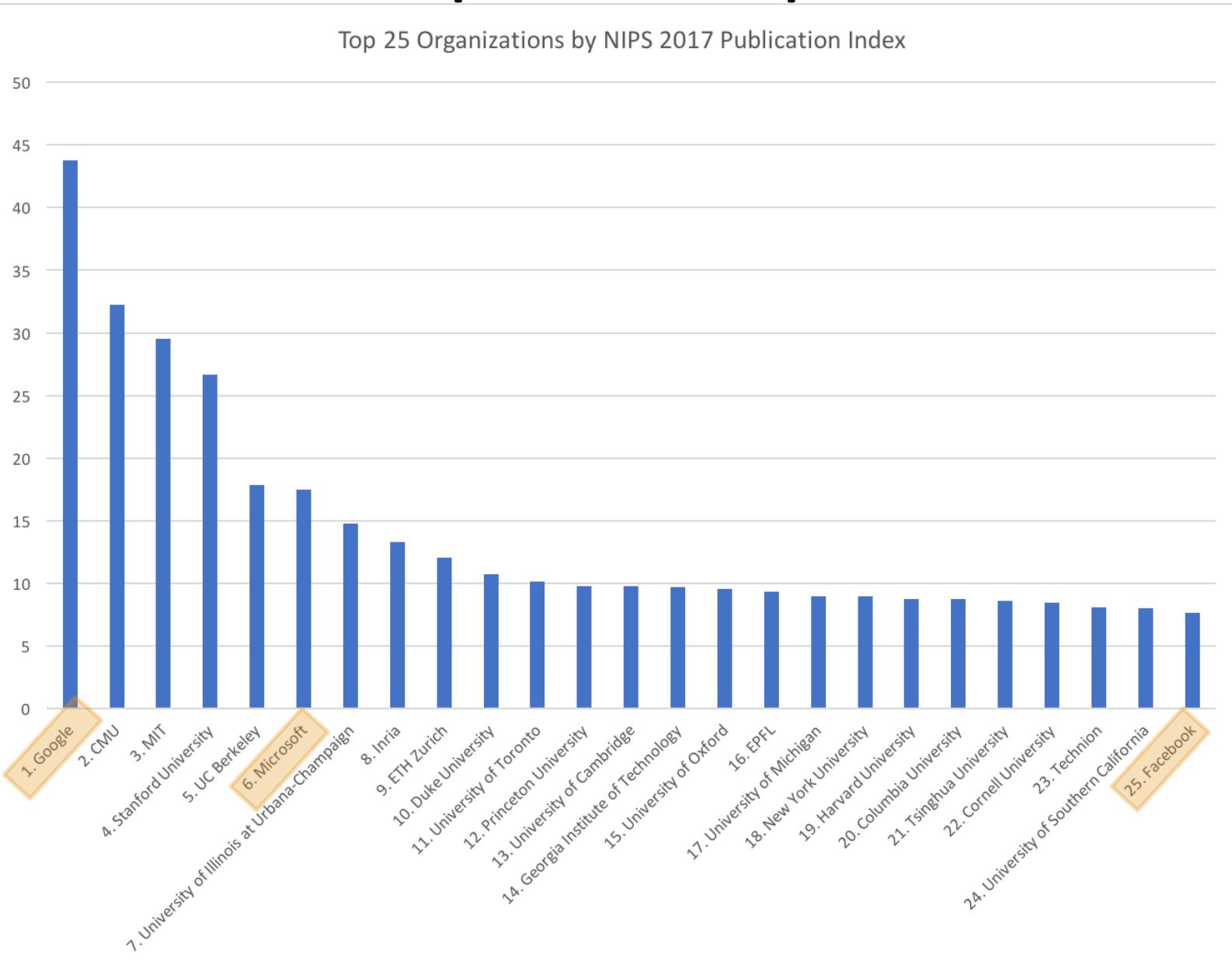


Jürgen Schmidhuber: Swiss AI Lab & NNAISENSE

Principales empresas



Principales empresas



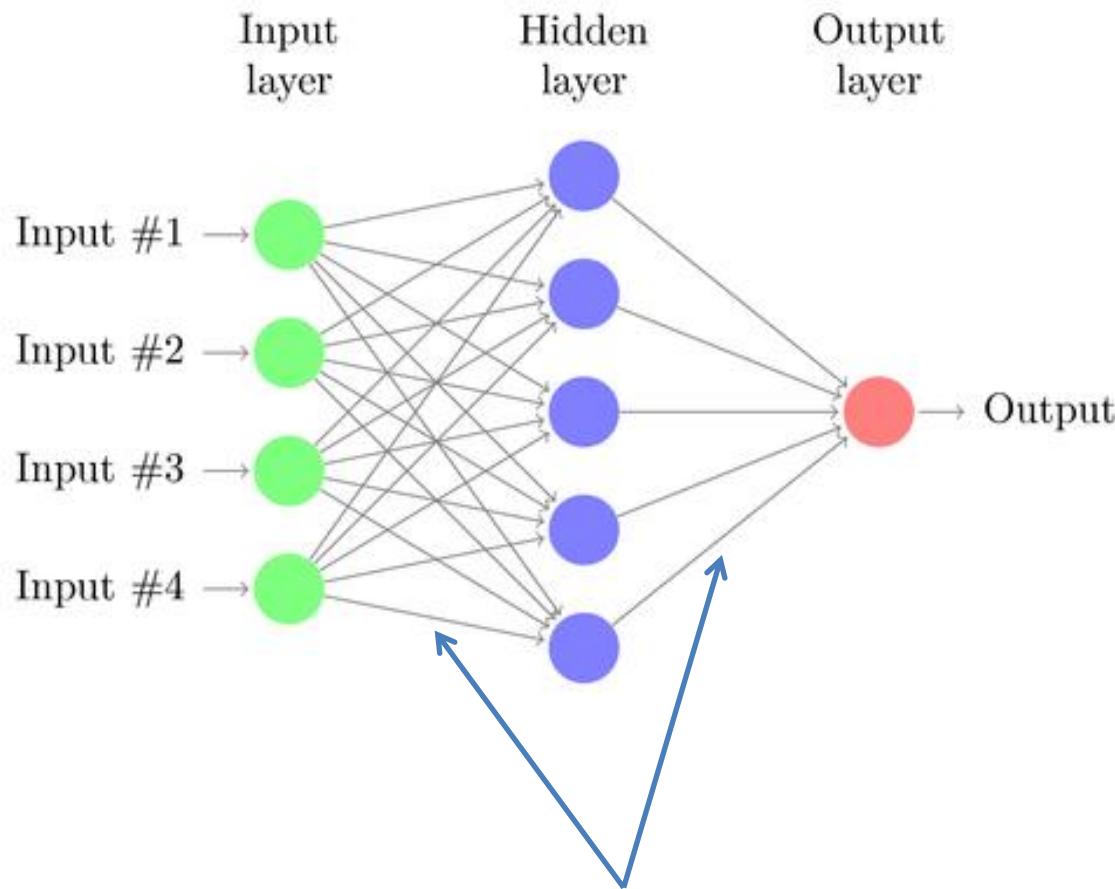
2. Redes Convolucionales

Recomendable lectura

(parte de estas diapositivas están extraídas de esta fuente)

- Muy recomendable consultar el curso de Stanford:
 - CS231n: Convolutional Neural Networks for Visual Recognition
(<http://cs231n.stanford.edu/2018/syllabus.html>)
- En concreto, sección sobre Convolutional Neural Networks (ConvNets):
http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture05.pdf
<https://cs231n.github.io/convolutional-networks/>

Hasta ahora...



Redes neuronales *shallow* (o superficiales): “pocas” capas de pesos. En el ejemplo de esta diapositiva, dos.

Había motivos teóricos para ello...

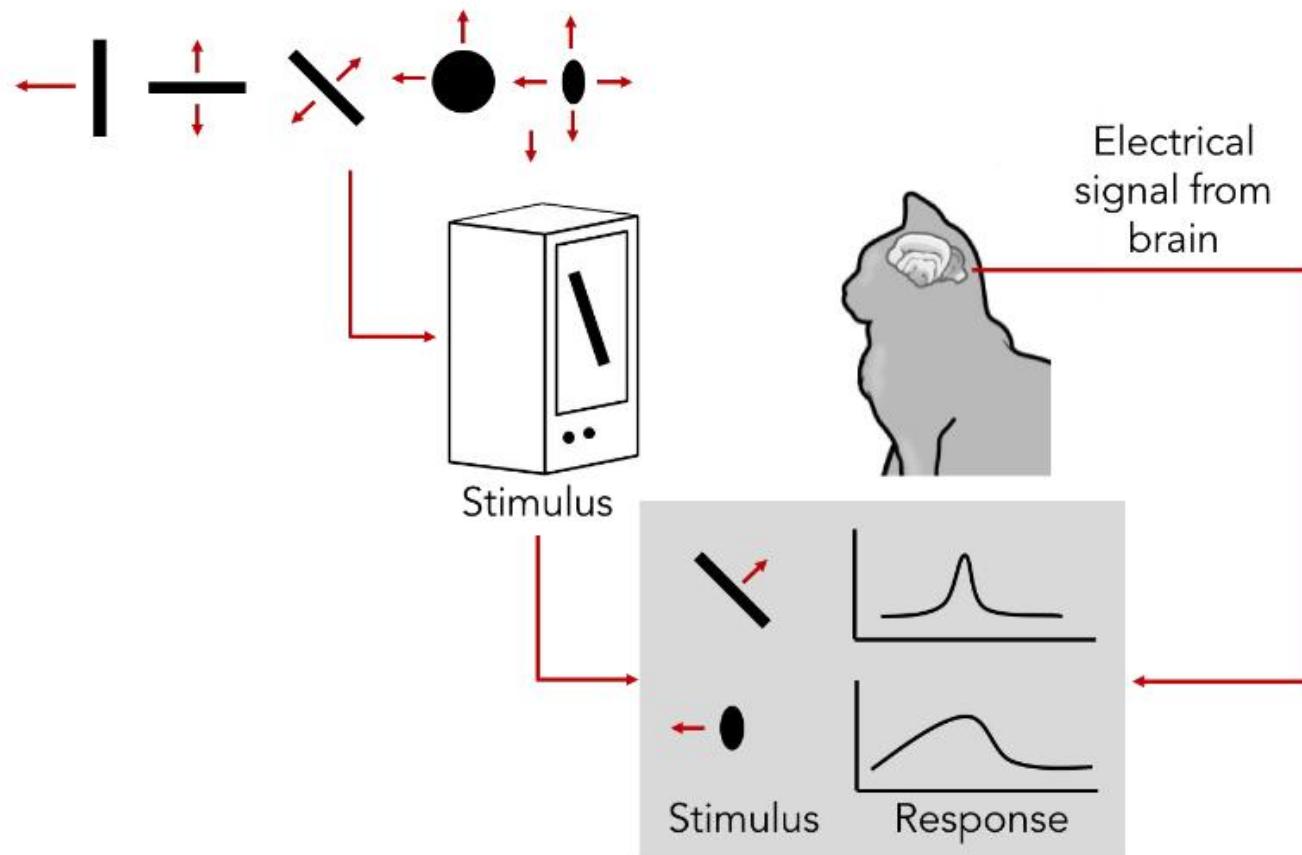
- ▶ Networks with two or more layers are **universal approximators**:
 - ▶ Any continuous function can be uniformly approximated to arbitrary accuracy, given enough hidden units.
 - ▶ This is true for many definitions of $h(\cdot)$, but excluding polynomials.
 - ▶ The key problem is how to find suitable parameter values given a set of training data.

<https://project.inria.fr/deeplearning/files/2016/05/session1.pdf>

Capítulo 5. Bishop, Christopher M. Pattern recognition and machine learning.
Springer, 2006.

Origen

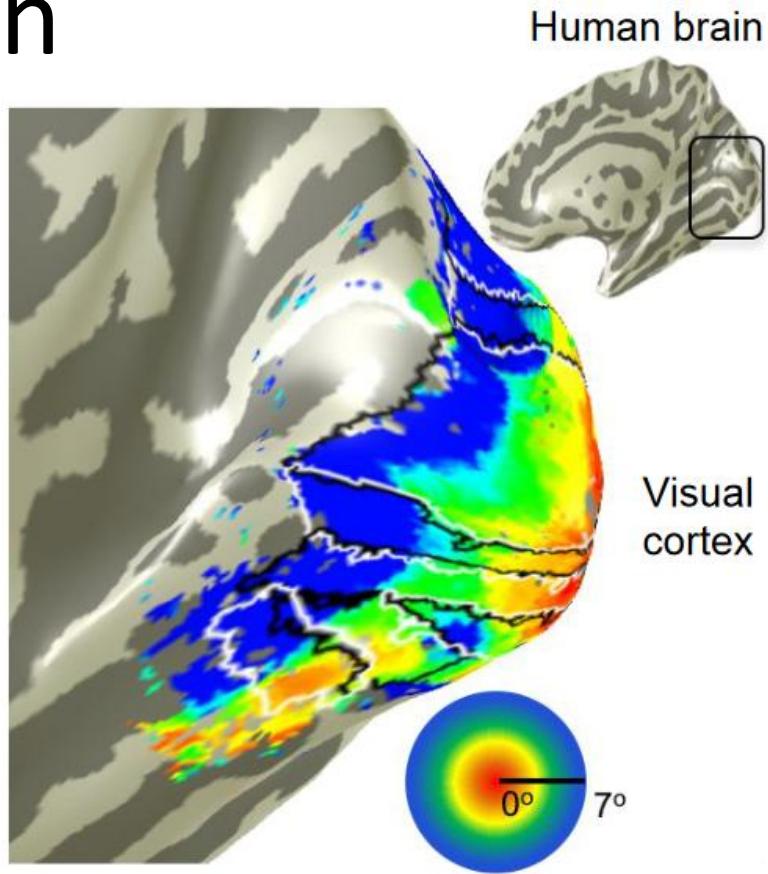
- El origen de las ConvNets puede encontrarse en el Neocognitron (Fukushima, 1980): modelo inspirado por los trabajos de Hubel y Wiesel (años 50 y 60) sobre la estructura y función del córtex visual.



¿De qué forma
los estímulos
visuales afectan
a la activación
de una
neurona?

Origen

- El córtex visual presenta un mapeado topográfico:
 - Células próximas en el córtex representan regiones próximas en el campo visual.
- Existe una **organización jerárquica**:
 - Células simples: responden a la orientación de la luz
 - Células complejas: responden a la orientación de la luz y al movimiento
 - Células hipercomplejas: responden al movimiento con un punto terminal.

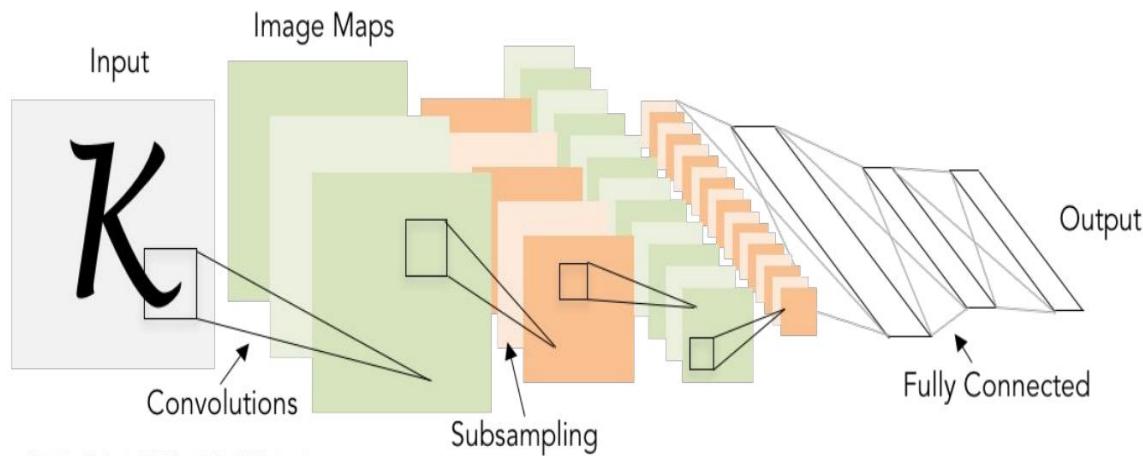


Retinotopy images courtesy of Jesse Gomez in the Stanford Vision & Perception Neuroscience Lab.



Definición

- ¿Qué es una Convolutional Neural Network (ConvNet o CNN)?
 - Red neuronal con una operación de convolución en, al menos, una de sus capas.
 - Utilizadas en problemas en donde la información se presenta en formato matricial/grid (p.ej. imágenes)
- Típica arquitectura de una ConvNet:



¿Qué es una convolución?

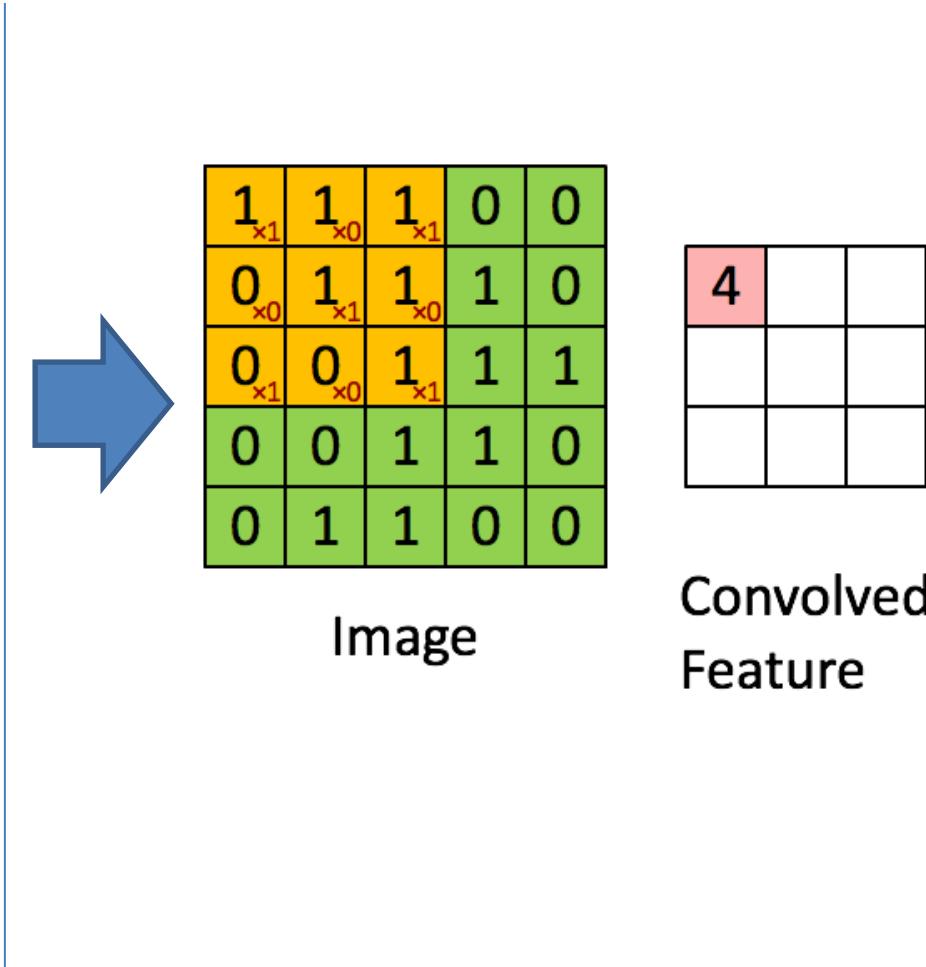
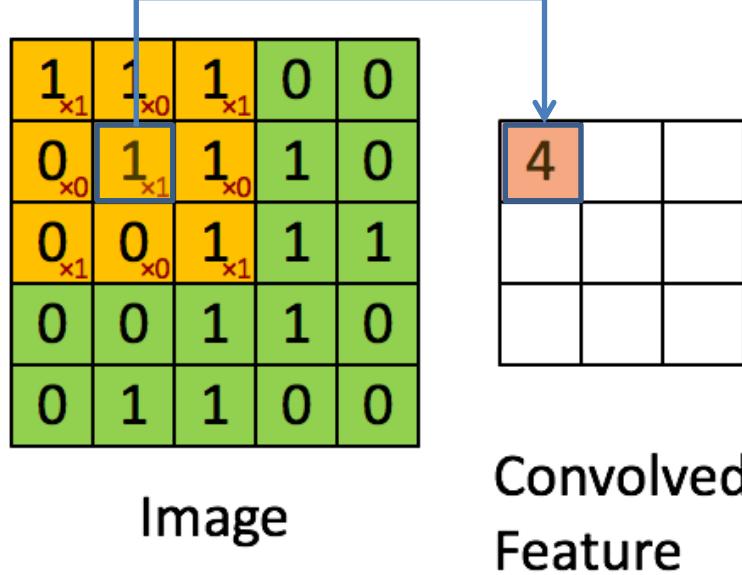
- **Operación lineal a nivel local** con una máscara.
 - Los coeficientes/valores de dicha máscara/filtro determinan la operación realizada.
- Técnicamente, una convolución es una correlación cruzada (*cross-correlation*) en donde el filtro/máscara se rota 180 grados.
 - A diferencia de la correlación, la convolución verifica (entre otras) la **propiedad asociativa**, lo que **nos permite construir filtros complejos a partir de filtros simples**.

¿Qué es una convolución?

- **Operación lineal a nivel local** con una máscara.

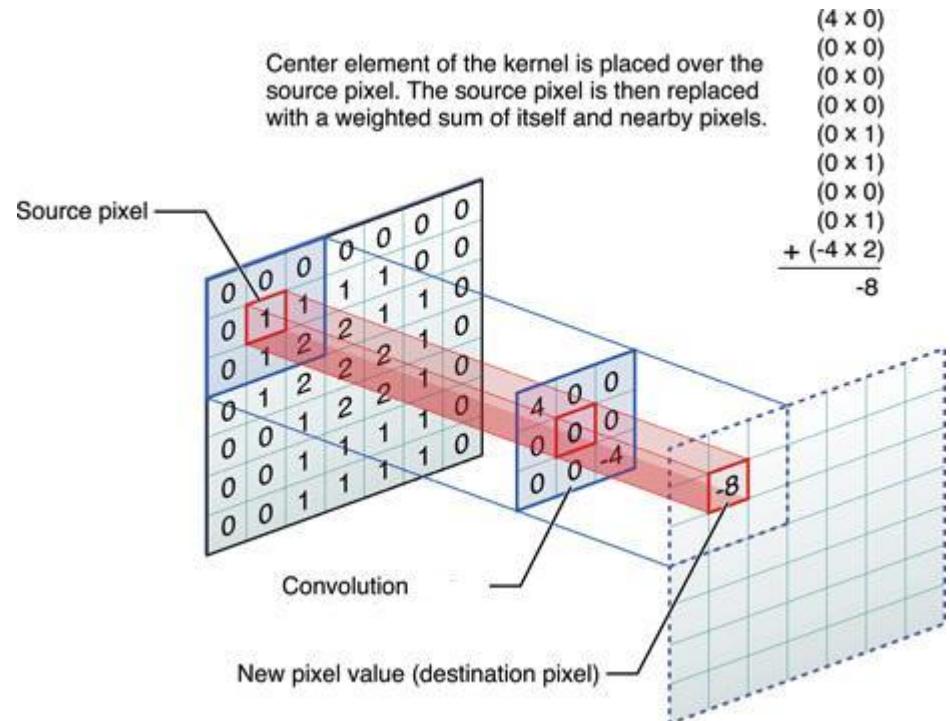
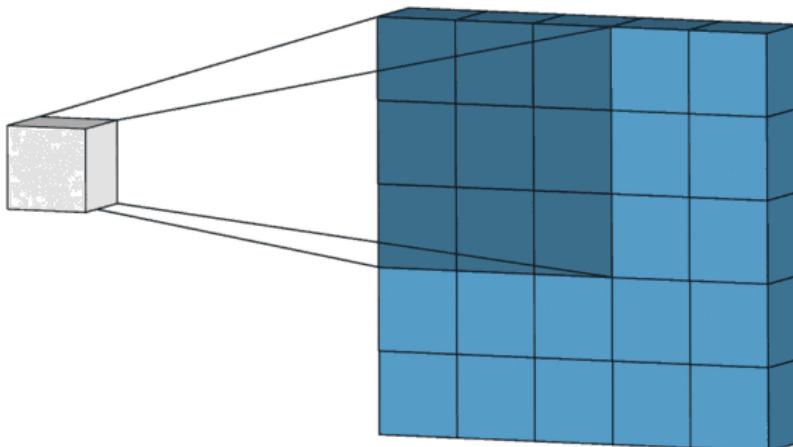
Los **números rojos** representan los valores/coeficientes del filtro/máscara.

Se multiplica elemento-a-elemento el filtro con la imagen, se suman los productos, y se sustituye la posición central del filtro en la imagen.



¿Qué es una convolución?

- Otra forma de visualizarlo...



<https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1>

<https://medium.com/@bdhumal/basic-things-to-know-about-convolution-dae5e1bc411>

¿Qué es una convolución?

- **Operación lineal a nivel local** con una máscara.
 - Los valores de la máscara determinan el resultado (características que se extraen)

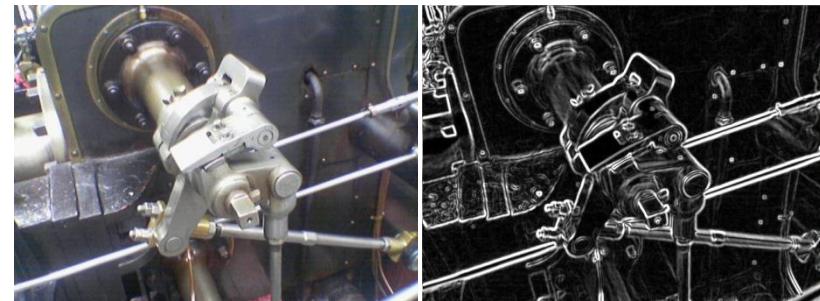
Filtro Gaussiano (elimina frecuencias altas → suaviza la imagen)

$$\frac{1}{273} \begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$



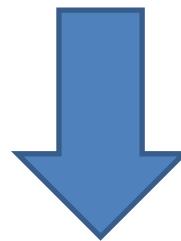
Filtro de Sobel (elimina bajas frecuencias → realza bordes)

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$



Idea clave

En ConvNets, ¡estos valores se aprenden! No vienen prefijados por un experto humano. Son parámetros libres de la red y se entranan como cualquier otro peso.



1989: LeCun et al. utilizaron back-propagation para aprender directamente los coeficientes de los filtros convolucionales a partir de imágenes de números manuscritos.

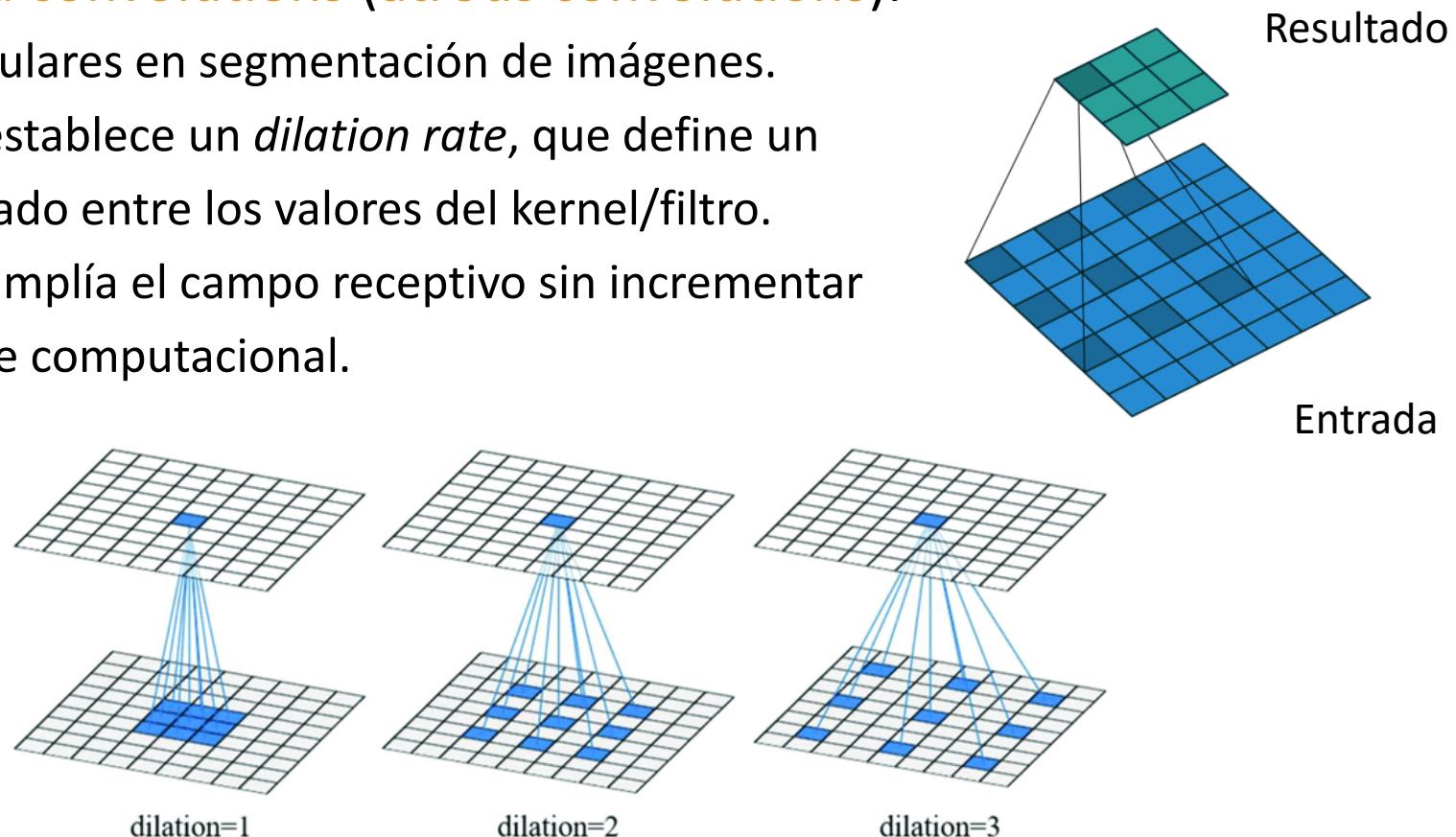
1998: LeCun et al. presentaron LeNet-5, ConvNet con 7 capas que permitía reconocer automáticamente números manuscritos en cheques, y demostraron que las ConvNets superan a todos los otros modelos en esta tarea.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4), 541-551.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278-2324.

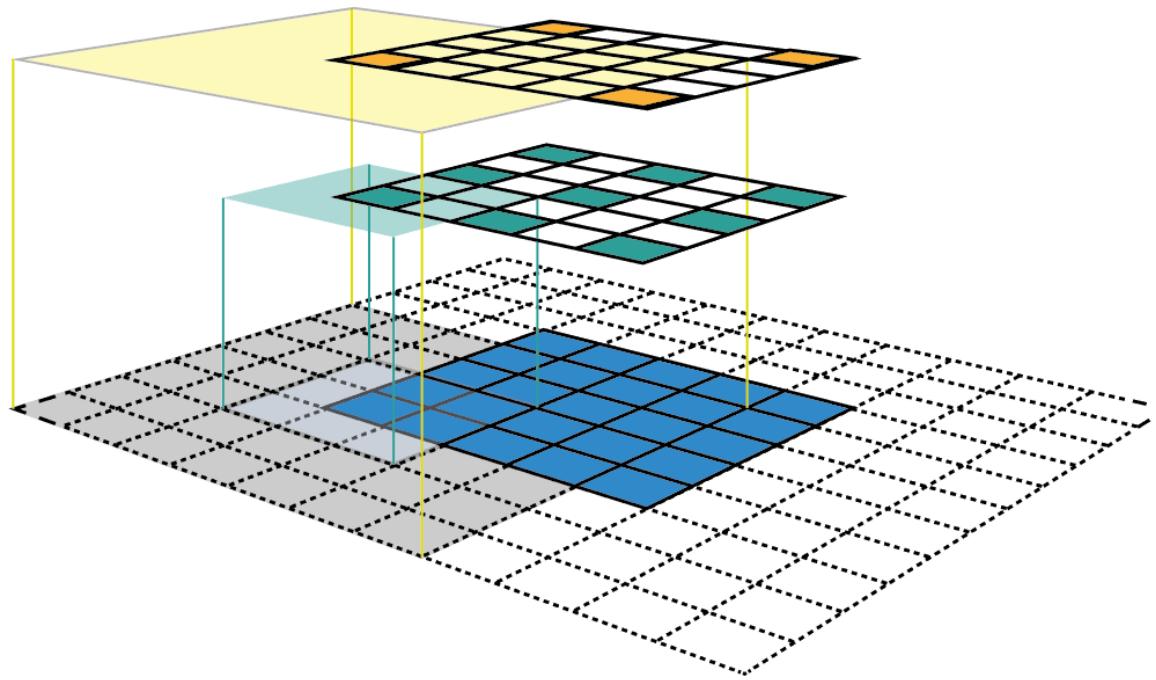
Otras formas de convolución

- Hasta ahora hemos visto la forma clásica/canónica de convolución, pero hay otras. Vale la pena, al menos, conocerlas.
- **Dilated convolutions (atrous convolutions).**
 - Populares en segmentación de imágenes.
 - Se establece un *dilation rate*, que define un espaciado entre los valores del kernel/filtro.
 - Se amplía el campo receptivo sin incrementar el coste computacional.



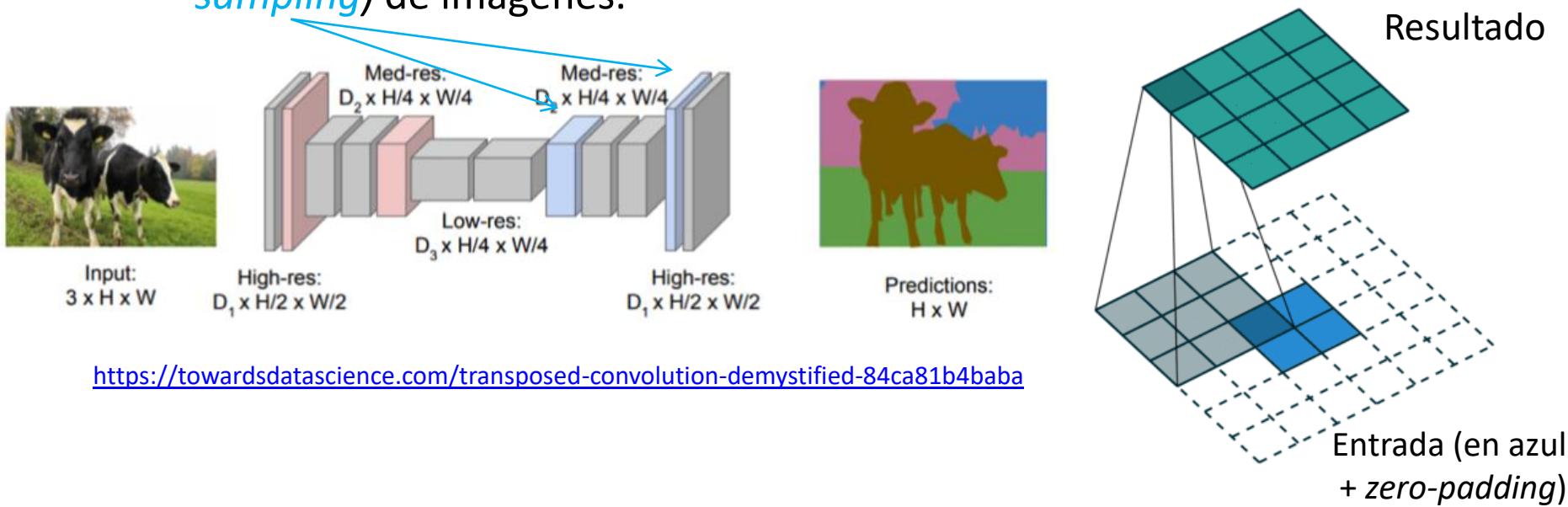
¿Qué es el campo receptivo?

- Región de la entrada que afecta a una unidad/*feature* particular de la red neuronal.



Otras formas de convolución

- **Transposed convolutions** (en ocasiones, erróneamente denominada *deconvolution*)
 - Populares en super-resolución y segmentación (en la parte de *upsampling*) de imágenes.



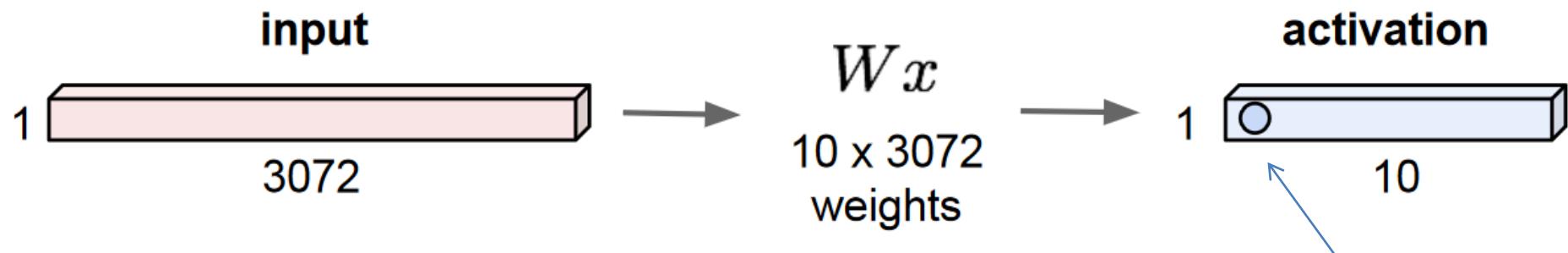
Recomendable lectura:

Vincent Dumoulin, Francesco Visin - [A guide to convolution arithmetic for deep learning](#)

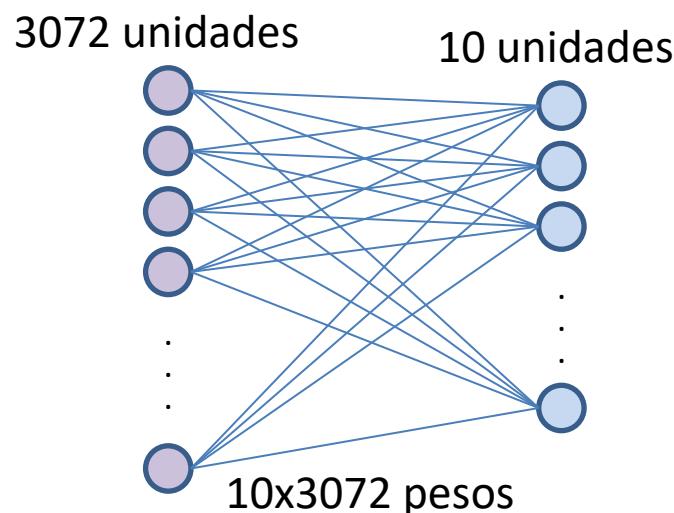
Aquí más visualizaciones: https://github.com/vdumoulin/conv_arithmetic

Capa Totalmente Conectada

Todas las unidades/neuronas de una capa están conectadas (pesos) con todas las de la capa siguiente.

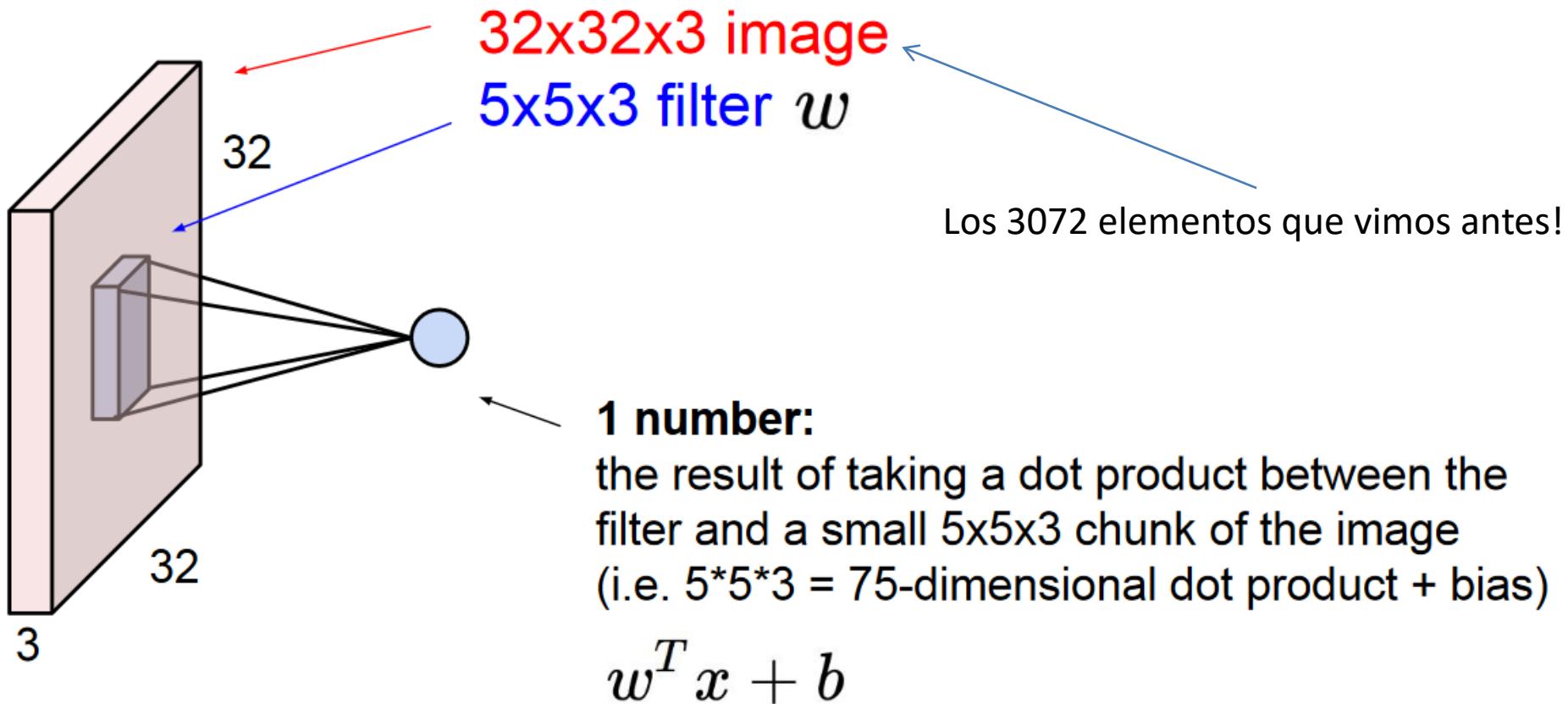


Otra forma de representarlo:

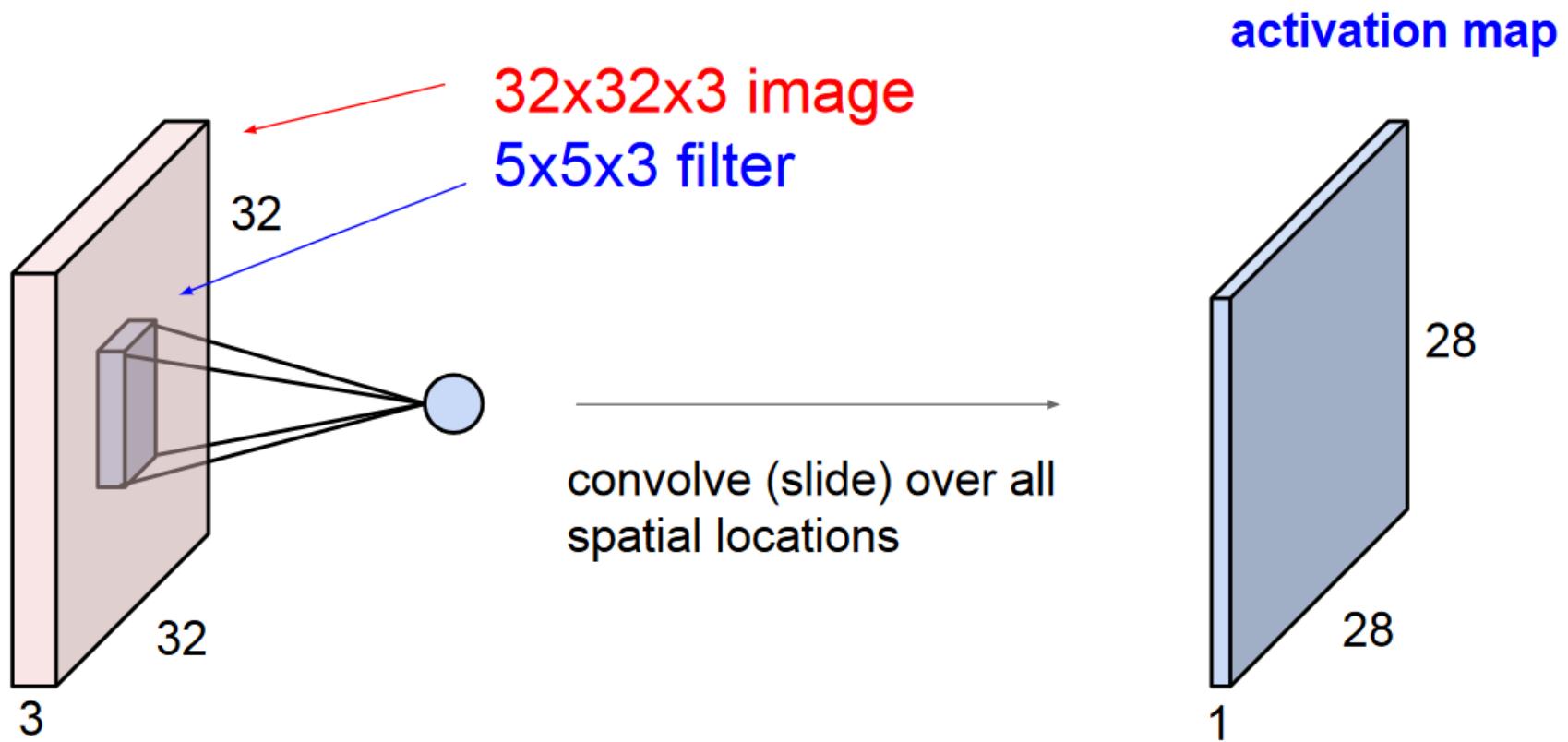


1 número:
resultado de
aplicar el
producto
escalar (*dot
product*) entre
una fila de W y
la entrada

Capa Convolucional

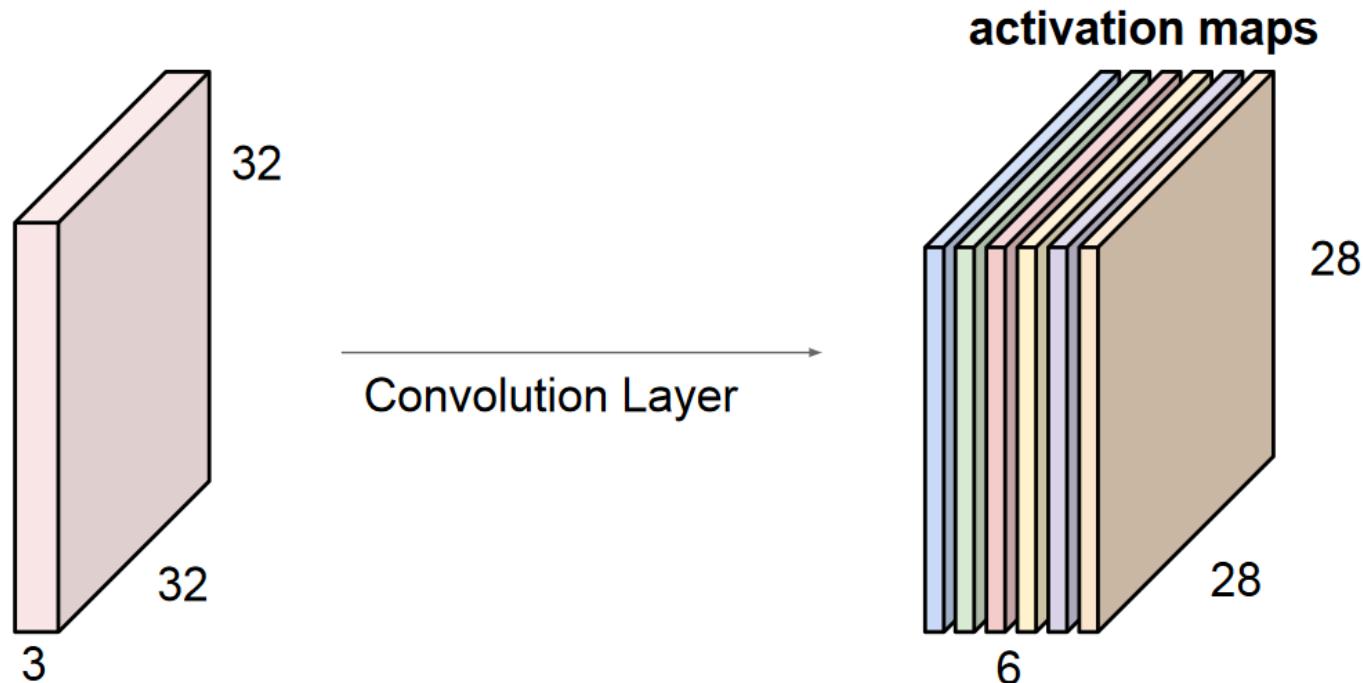


Capa Convolucional



Capa Convolucional

For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:

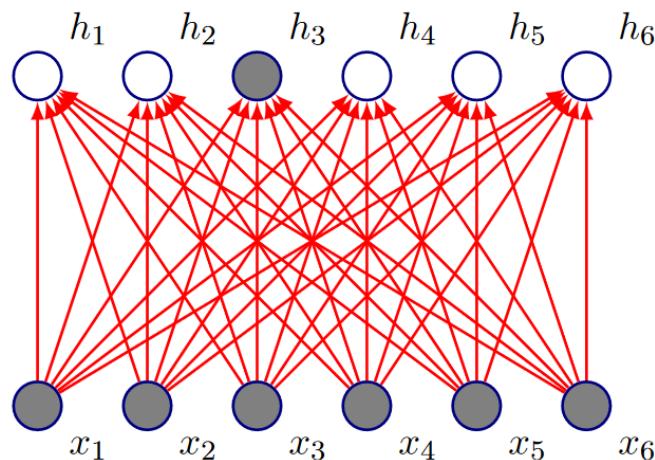


We stack these up to get a “new image” of size 28x28x6!

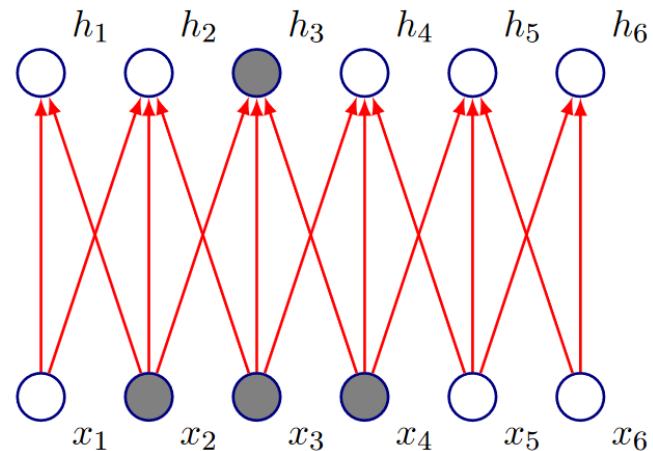
¿Qué ventajas presenta usar convoluciones?

- **Conejividad dispersa**

- Las *fully-connected layers* operan a nivel global (cada neurona ve la entrada completa), mientras que las *convolutional layers* operan a nivel local → **Menos operaciones!**



vs



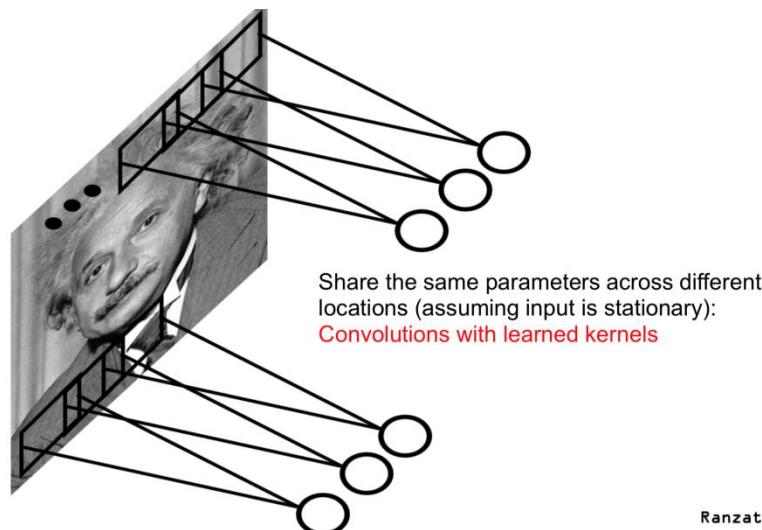
- Fully connected network: h_3 is computed by full matrix multiplication with no sparse connectivity

- Kernel of size 3, moved with stride of 1
- h_3 only depends on x_2, x_3, x_4

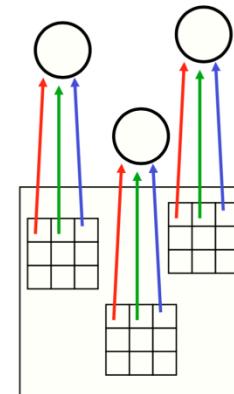
¿Qué ventajas presenta usar convoluciones?

- **Compartición de parámetros**

- El mismo filtro/ es aplicado en toda la imagen.
 - En lugar de aprender un parámetro para cada localización de la imagen, solo se aprenden un conjunto reducido de ellos (los correspondientes al filtro).
 - Se reduce mucho el número de parámetros a aprender y almacenar! → Regularización!



The red connections all have the same weight.



https://www.cs.toronto.edu/~lczhang/aps360_2019/1/lec/w03/convnet.html

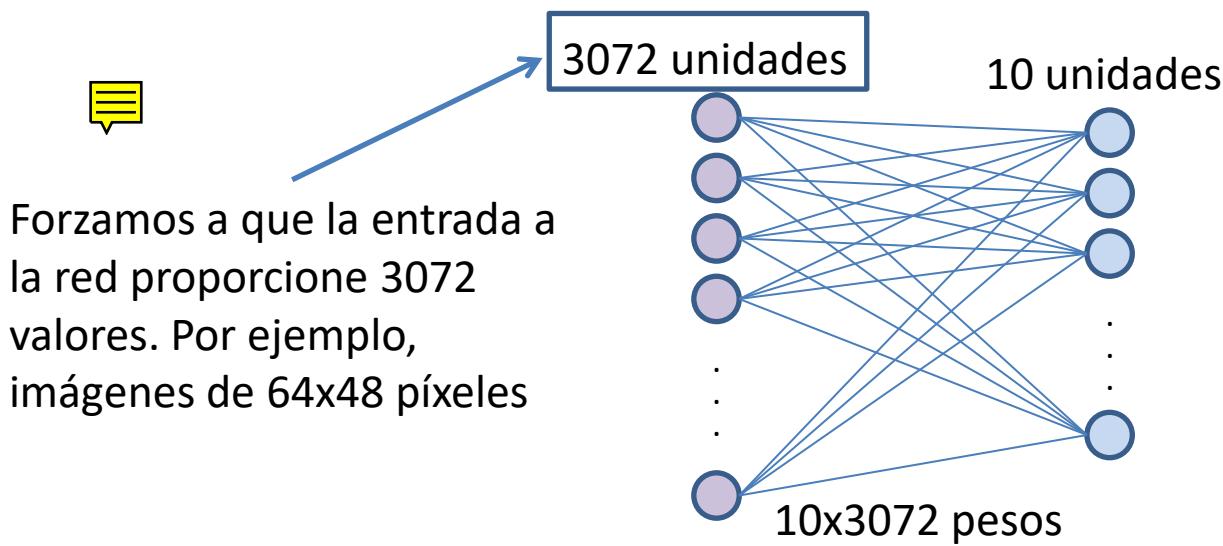
¿Qué ventajas presenta usar convoluciones?

- **Representaciones equivariantes**
 - Debido a lo anterior, cada capa convolucional es equivariante con respecto a la translación.

f es equivariante con respecto g si $f(g(\mathbf{x})) = g(f(\mathbf{x}))$
 - Si trasladamos un objeto en la imagen, su representación se trasladará la misma distancia en la salida.
 - Permite generalizar la detección de bordes, texturas y formas en distintas localizaciones de la imagen!
 - La convolución no es equivariante con respecto a escala o rotación.

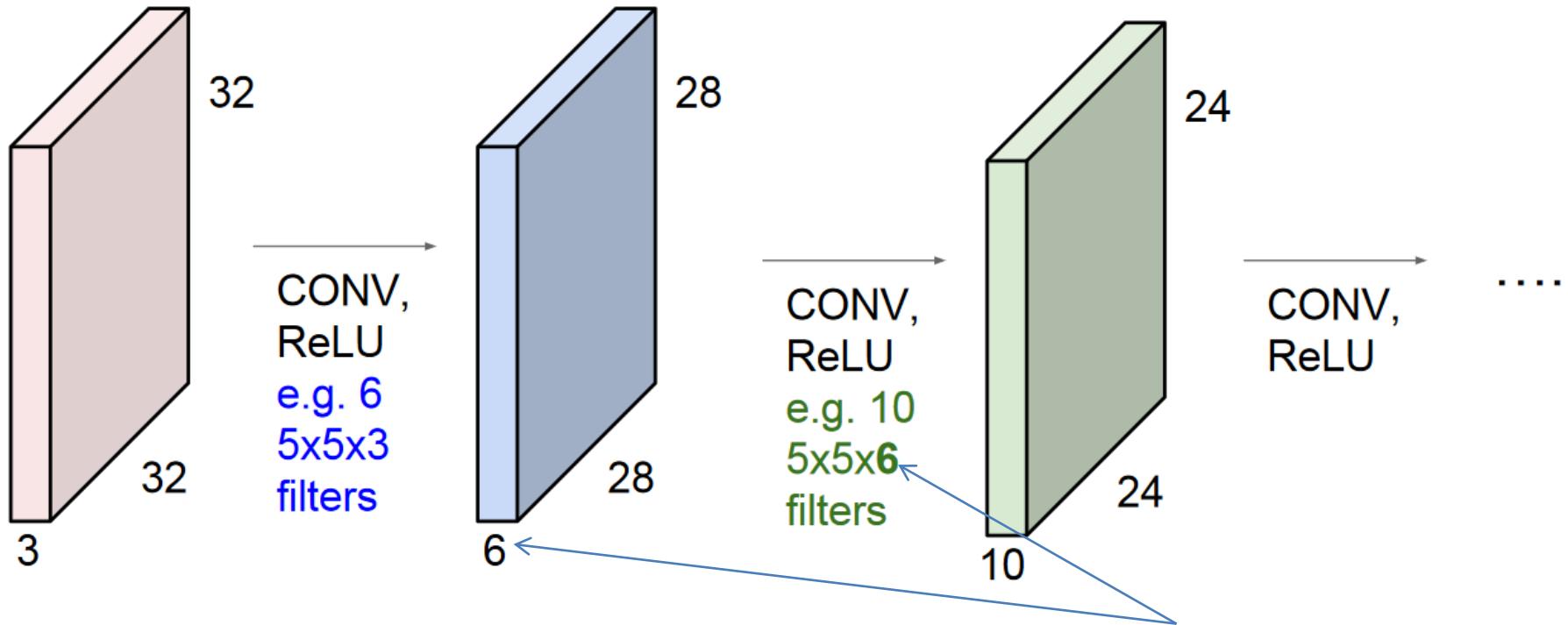
¿Qué ventajas presenta usar convoluciones?

- Capacidad para **operar sobre entradas de tamaño variable**
 - Una *fully-connected layer* fuerza la entrada a tener un cierto tamaño.



Redes Convolucionales

- Se intercalan capas convolucionales con funciones de activación.



Fijateos que los filtros siempre tienen la misma profundidad que el bloque convolucional anterior, es decir, extienden la profundidad completa del volumen/tensor de entrada.

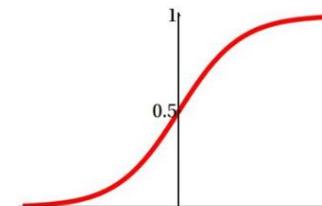


Redes Convolucionales

- Se intercalan capas convolucionales con **funciones de activación (no lineales)**.

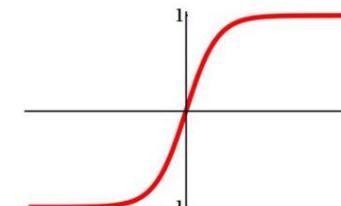
- Tradicionalmente, las funciones de activación eran **sigmoidales**, como la *logistic sigmoid* o la hiperbólica tangente.

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



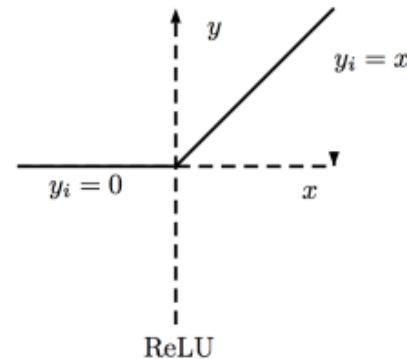
logistic (sigmoid, unipolar)

$$\tanh(\Sigma) = \frac{e^{\Sigma} - e^{-\Sigma}}{e^{\Sigma} + e^{-\Sigma}}$$

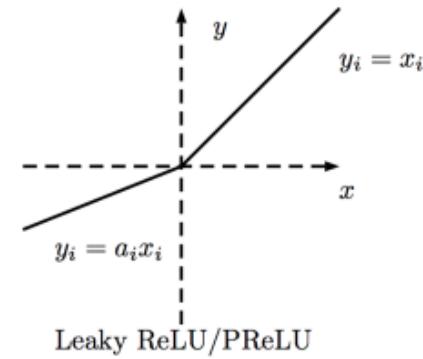


tanh (bipolar)

- En redes profundas se suele recomendar utilizar la unidad lineal rectificada (**ReLU**), entre otros motivos porque permiten entrenar redes profundas más rápido.



ReLU



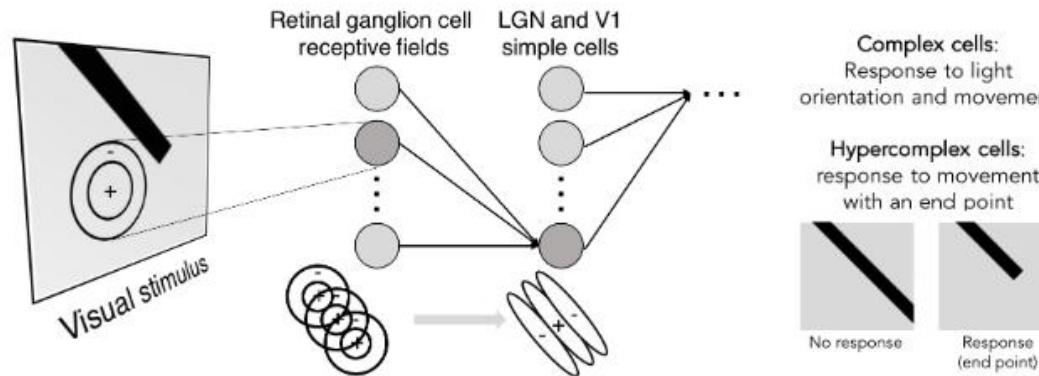
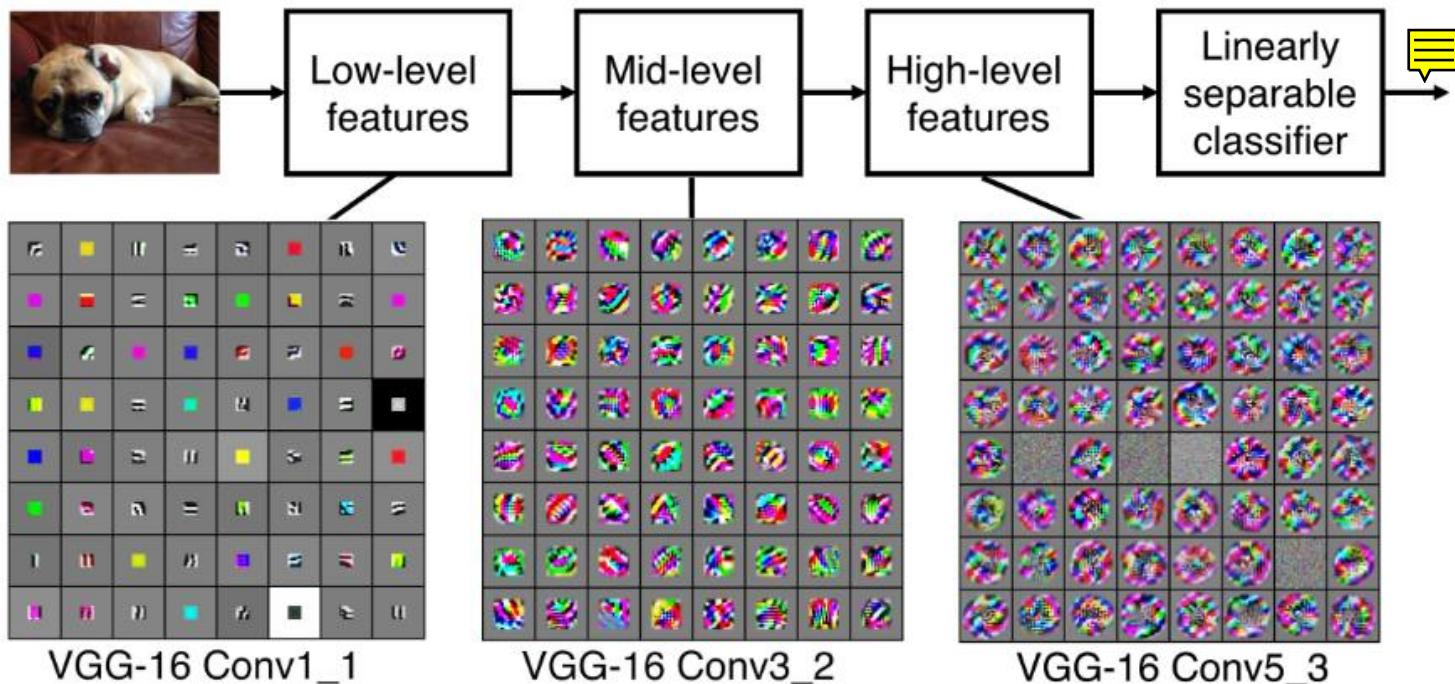
Leaky ReLU/PReLU

Redes Convolucionales

- Se intercalan capas convolucionales con **funciones de activación (no lineales)**.
 - Esta no linealidad es **necesaria para aprender complejas representaciones de los datos de entrada**.
 - De lo contrario, si se usasen solamente funciones de activación lineales, la red neuronal se comportaría como una función lineal.

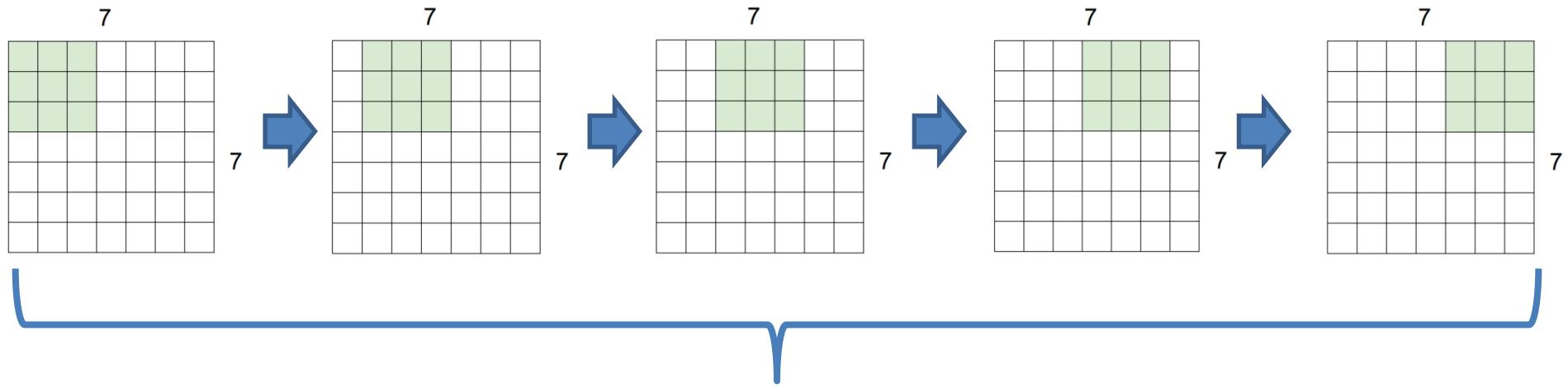
Si todas las unidades ocultas tienen activaciones lineales, entonces siempre se puede encontrar una red equivalente sin unidades ocultas.

Redes Convolucionales



Redes Convolucionales

- Prestemos ahora atención a las dimensiones
 - Ejemplo imagen 7×7 y filtro 3×3

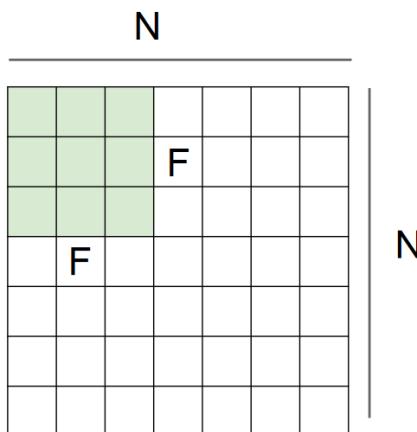
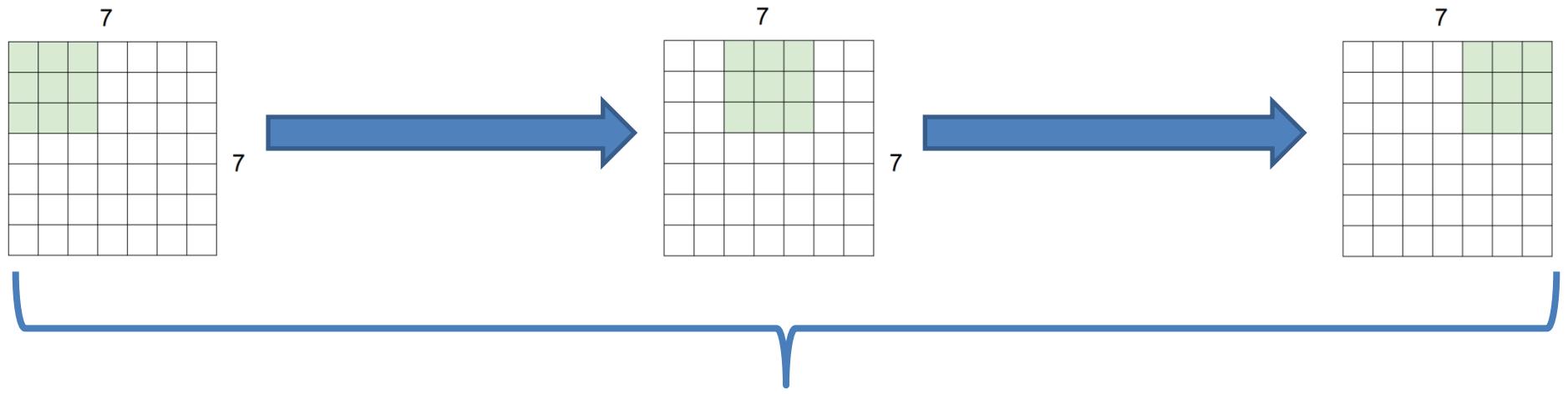


La salida es una imagen de 5×5
(fijaos que no llegamos a “sustituir” los bordes)

- Aquí aparecen los conceptos de **stride** y **padding**.

Redes Convolucionales

- Prestemos ahora atención a las dimensiones
 - Ejemplo imagen 7x7 y filtro 3x3, con **stride** 2



Output size:
 $(N - F) / \text{stride} + 1$

e.g. $N = 7, F = 3$:
stride 1 => $(7 - 3)/1 + 1 = 5$
stride 2 => $(7 - 3)/2 + 1 = 3$
stride 3 => $(7 - 3)/3 + 1 = 2.33$

Error! A priori, no se puede aplicar un filtro 3x3 en una entrada de 7x7 con stride 3!

Redes Convolucionales

- Vamos progresivamente reduciendo el tamaño de los mapas/volúmenes.
 - Si queremos conservar el tamaño → **padding**
 - Ejemplo *zero-padding*:

0	0	0	0	0	0			
0								
0								
0								
0								

e.g. input 7x7

3x3 filter, applied with **stride 1**

pad with 1 pixel border => what is the output?

7x7 output!

En muchas ocasiones es recomendable emplear *padding*, dado que la evidencia empírica aconseja no reducir dimensionalidad demasiado deprisa!

Redes Convolucionales

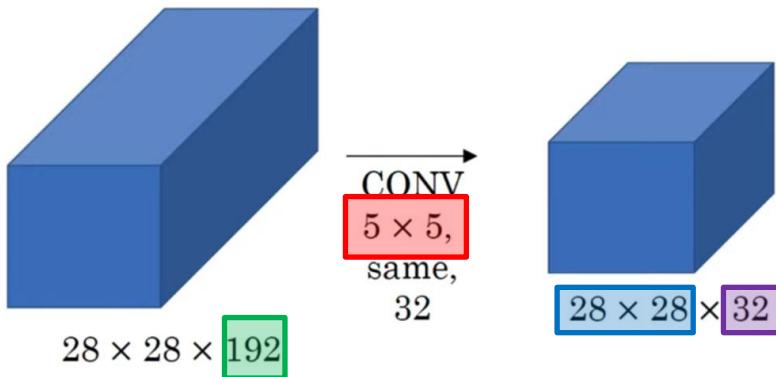
- Ejemplo completo:
 - Volumen de entrada: **16x16x3**
 - Filtros empleados: **10** de **3x3** con stride **1** y padding **1**

$\text{Tamaño_salida} = ((\text{N}+2*\text{P}-\text{F})/\text{stride}) + 1$
 - ¿Volumen de salida?
 - $(16+2*1-3)/1 + 1 = 16 \rightarrow 16x16x10$
 - ¿Número de parámetros en esa capa?
 - Cada filtro tiene $3x3x3 + 1 = 27$ parámetros (+1 por el *bias*)
 $\rightarrow 27x10 = 270$ parámetros

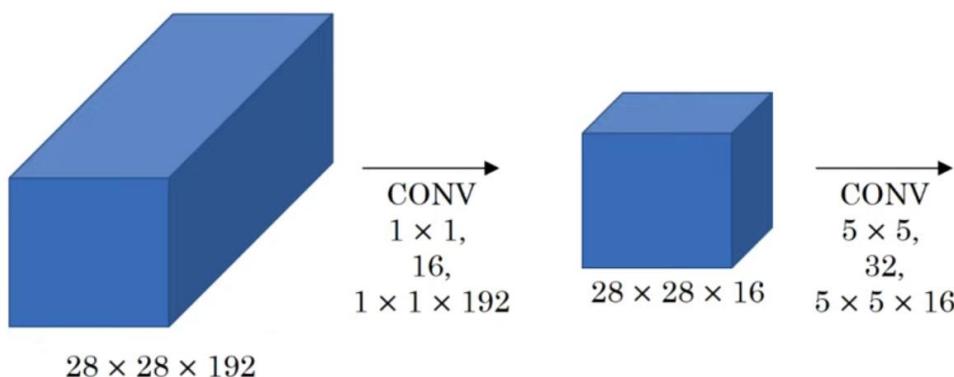
Redes Convolucionales

• Convoluciones 1x1

- Se emplean para combinar linealmente información de distintos canales y reducir (o ampliar) el número de filtros (*feature maps*).
- Reduce dimensionalidad y coste computacional.



Para obtener cada elemento del volumen de salida tenemos que realizar **5x5x192** operaciones (productos) → **5x5x192x28x28x32** = $\sim 120M$ operaciones/multiplicaciones



Ahora, tenemos que realizar:
a) $1 \times 1 \times 192 \times 28 \times 28 \times 16 =$

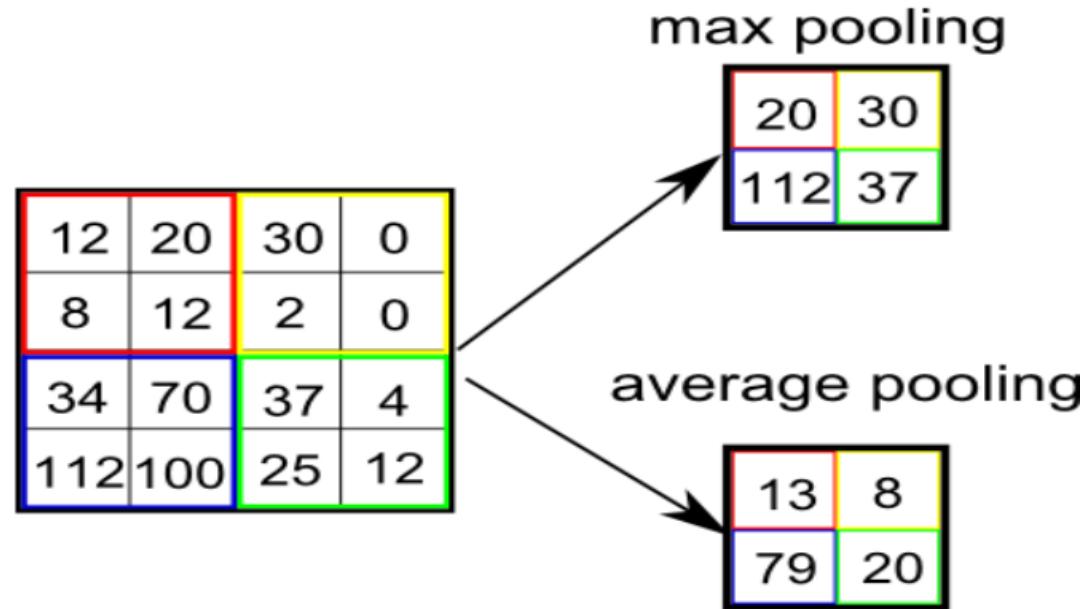
$\sim 2.4M$ mult.
b) $5 \times 5 \times 16 \times 28 \times 28 \times 32 = \sim 10M$ mult.

Total (a+b): $\sim 12.4M$ operaciones

Redes Convolucionales

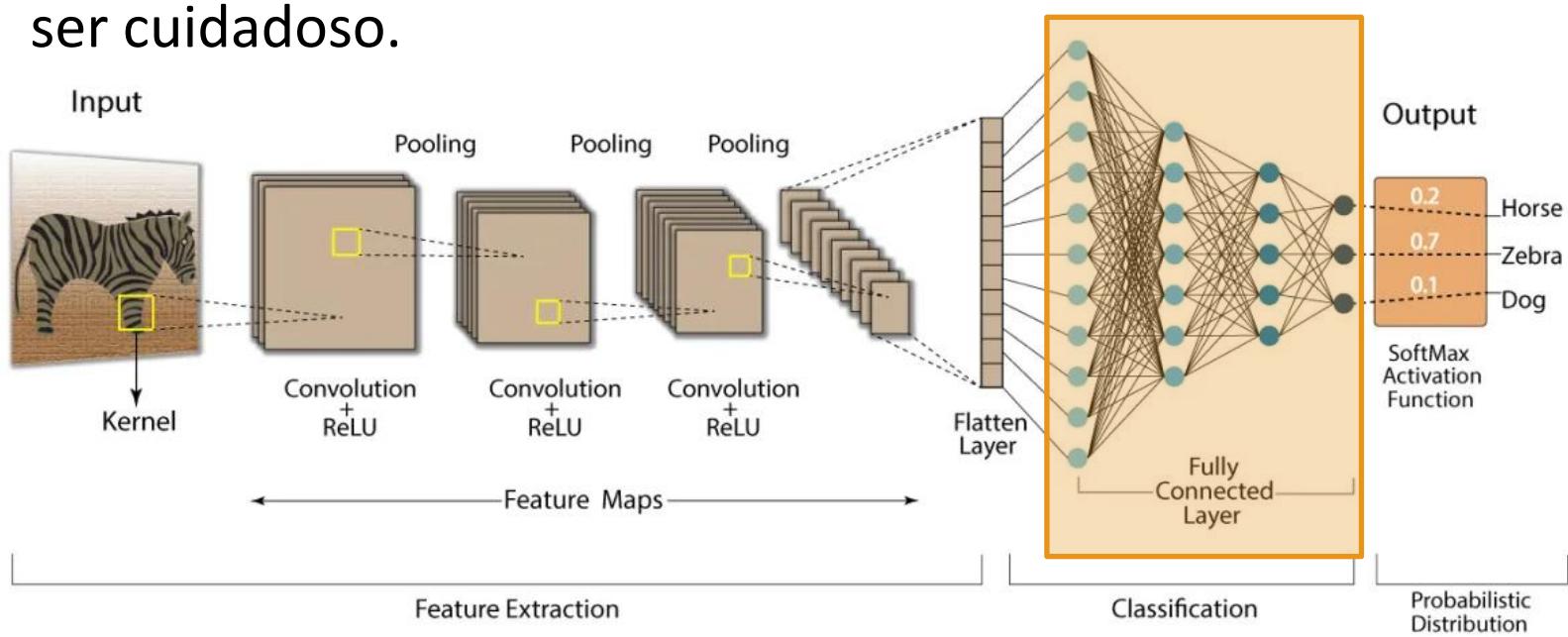
- **Pooling.** Reduce dimensionalidad e introduce cierta invarianza a (pequeñas) traslaciones de la entrada
 - Si la entrada es desplazada por una pequeña cantidad, la mayor parte de los valores de salida del *pooling* no cambian.

Ejemplo de *max* y *average* pooling con filtros de 2x2 y stride 2

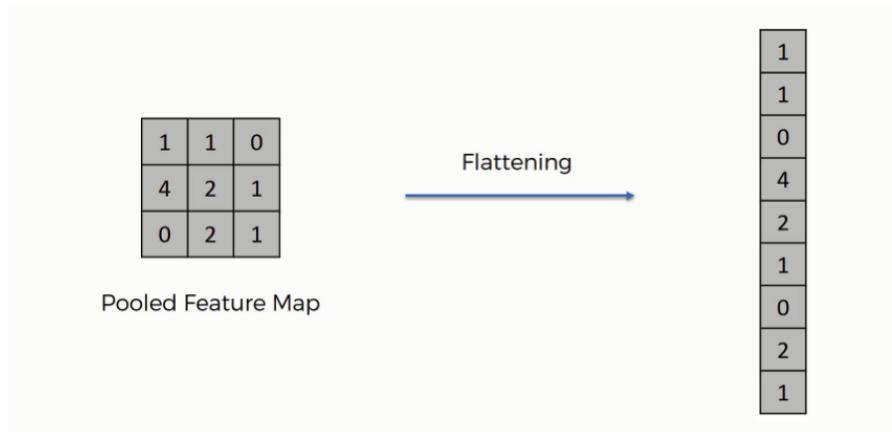


Redes Convolucionales

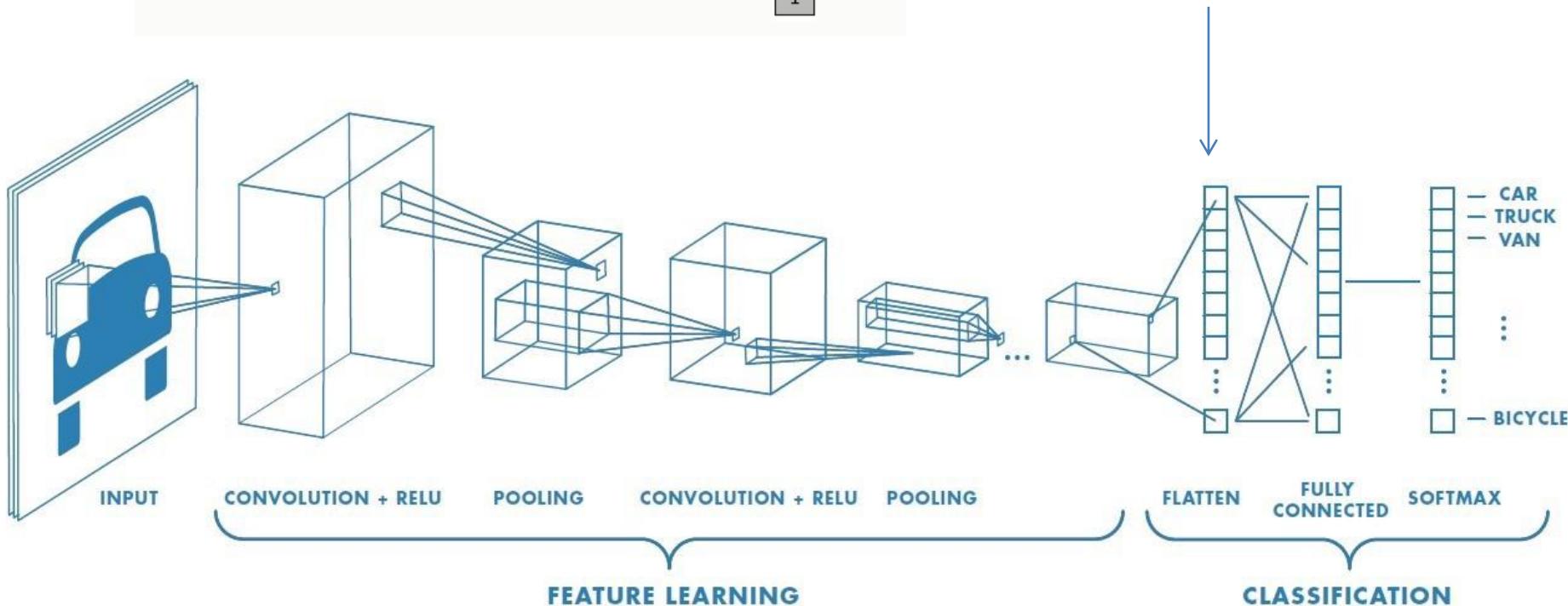
- **Fully-connected layers (capas totalmente conectadas).**
 - Las redes convolucionales pueden, o no, tener capas totalmente conectadas (generalmente, al final de la red).
 - Todas las unidades de una capa están conectadas con todas las unidades de la siguiente.
 - Suelen contener muchos parámetros, de modo que su uso debe ser cuidadoso.



Combinándolo todo



Flatten: Capa que “aplana” las *features*, permitiendo pasar de un volumen (conjunto de mapas de características) a un vector que puede ser procesado por una capa totalmente conectada



Combinándolo todo

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}$$

σ = softmax

\vec{z} = input vector

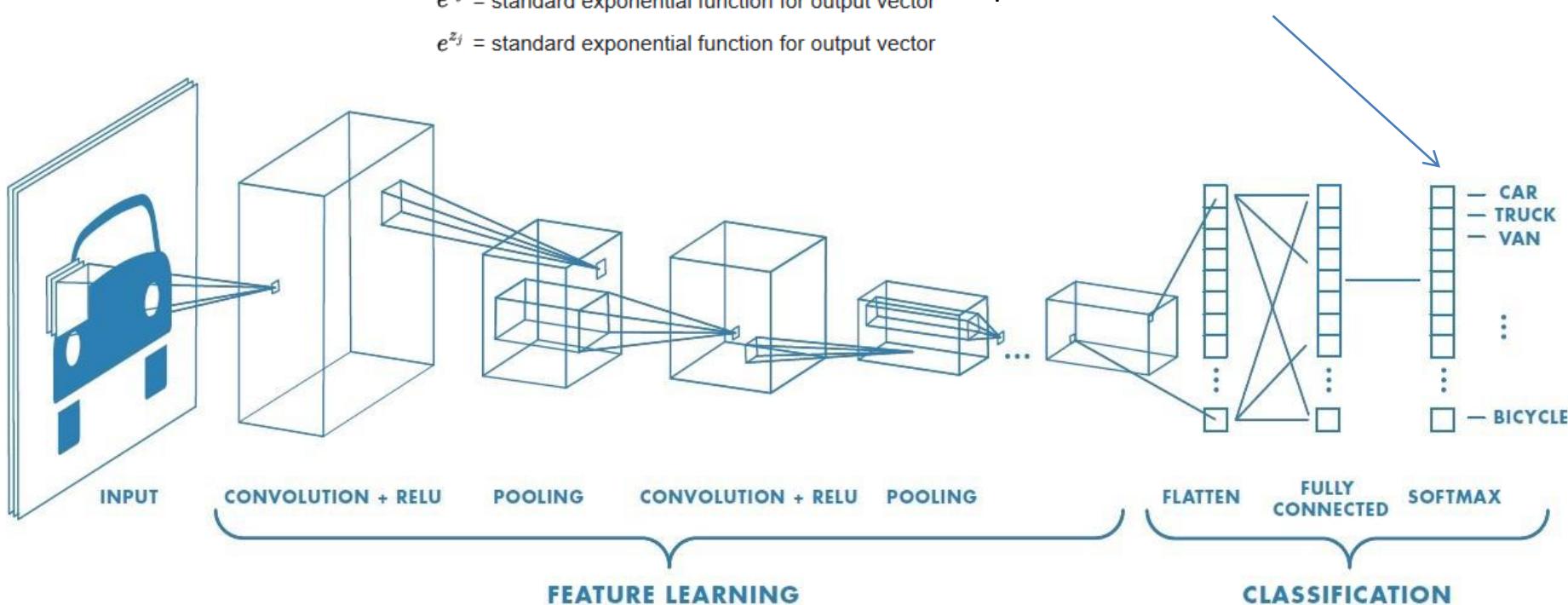
e^{z_i} = standard exponential function for input vector

K = number of classes in the multi-class classifier

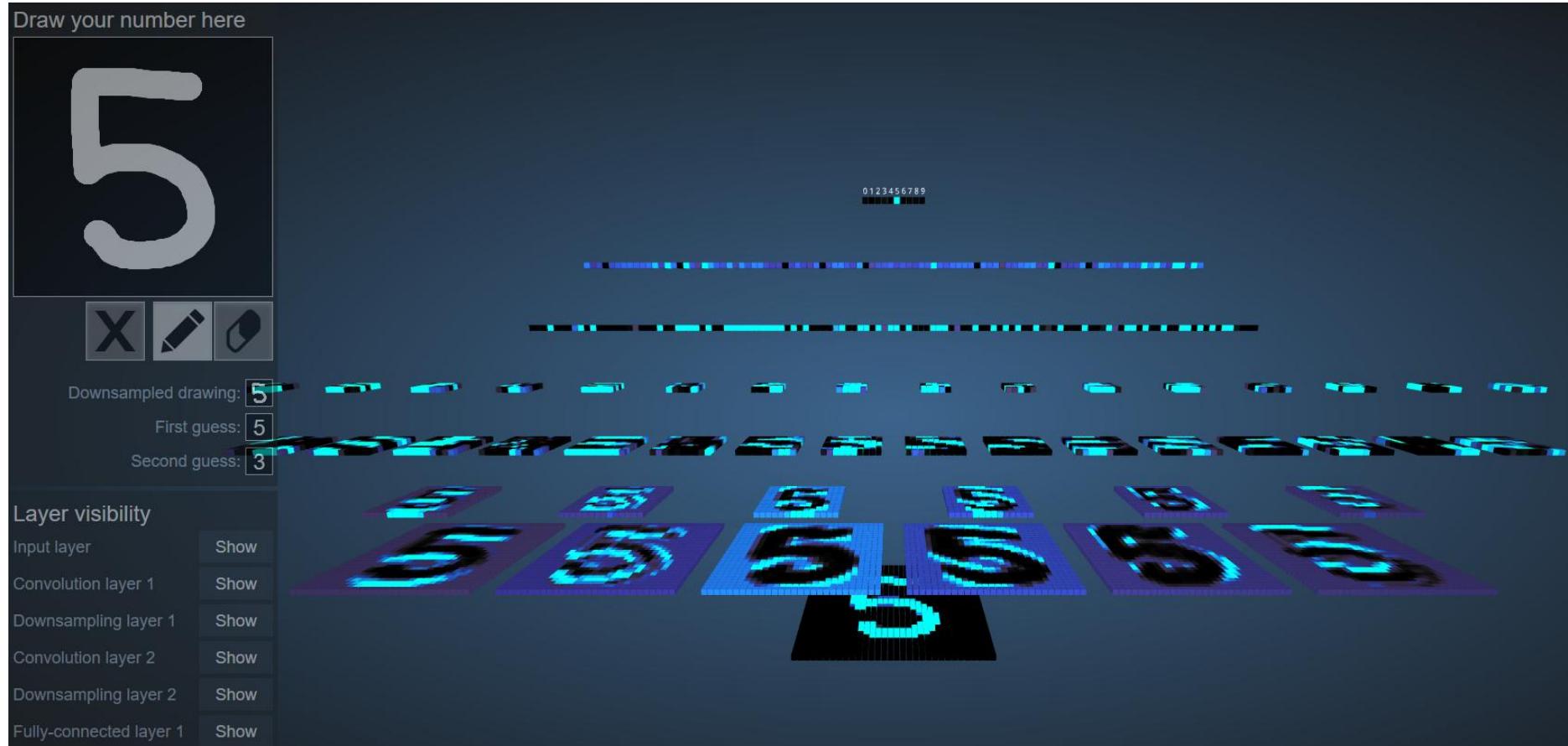
e^{z_j} = standard exponential function for output vector

e^{z_j} = standard exponential function for output vector

Softmax: función de activación que generaliza la función sigmoide a múltiples clases. Normaliza la salida de la red, de modo que cada predicción se corresponde con la probabilidad de que la entrada pertenezca a dicha clase.



Visualizando el funcionamiento



<https://www.cs.ryerson.ca/~aharley/vis/conv/>

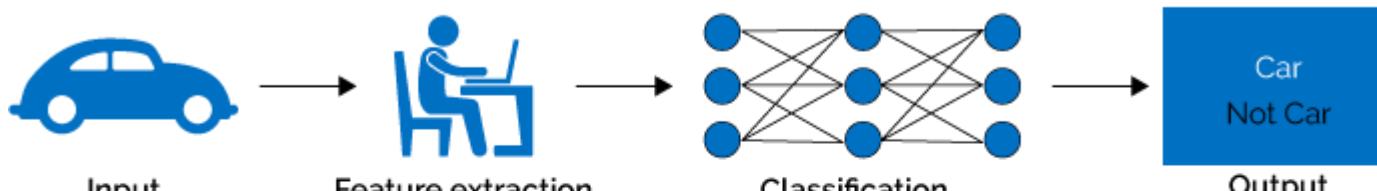
Otra herramienta interesante: <https://cs.stanford.edu/people/karpathy/convnetjs/>

Y otra: <https://playground.tensorflow.org/>

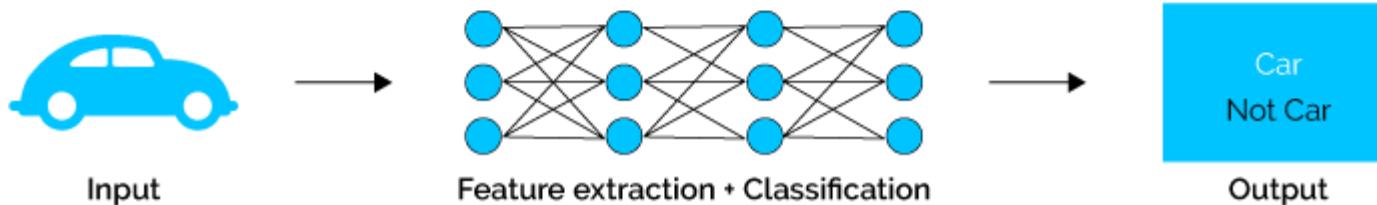
Ventaja de las ConvNets

- A nivel metodológico: permiten aprender características (*feature learning*) en lugar de diseñarlas a mano (*engineered feature*).
- A nivel empírico: proporcionan resultados superiores en numerosas tareas de visión (clasificación, regresión, segmentación, etc.)

Aproximación Clásica de Aprendizaje Automático



Deep Learning

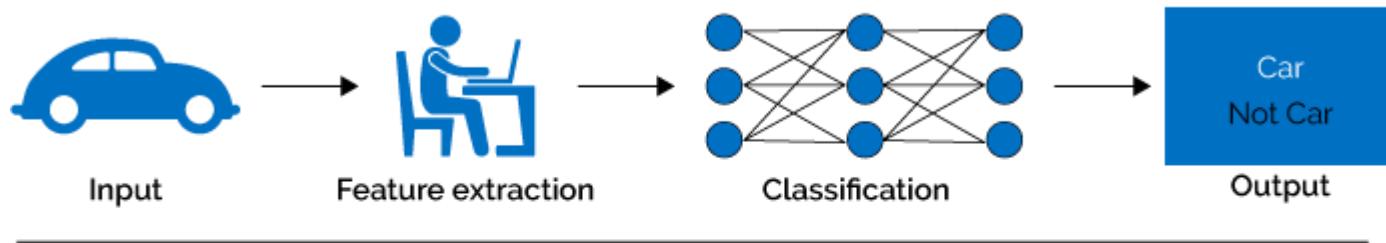


<https://towardsdatascience.com/cnn-application-on-structured-data-automated-feature-extraction-8f2cd28d9a7e>

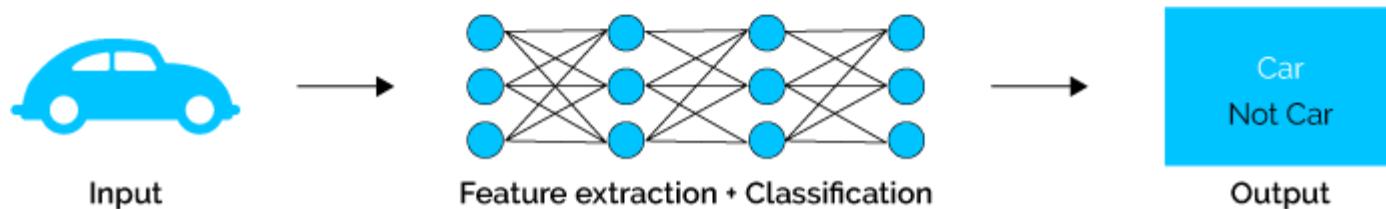
Ventaja de las ConvNets

- Deep Learning difumina los límites entre extracción de características y clasificación.
- La clave de todo es tener una buena representación interna de los datos de entrada.

Aproximación Clásica de Aprendizaje Automático



Deep Learning



¿Cómo se entrenan las ConvNets?

- Se emplea *backpropagation*, como con el perceptrón multicapa.
 - El aprendizaje de los parámetros se convierte en un problema de optimización.
- No confundir *backprop* y descenso de gradiente!
 - El **descenso de gradiente** usa **backprop**!

Método de optimización para minimizar una función de pérdida usando el gradiente

$$\mathbf{a}_{n+1} = \mathbf{a}_n - \gamma \nabla F(\mathbf{a}_n)$$

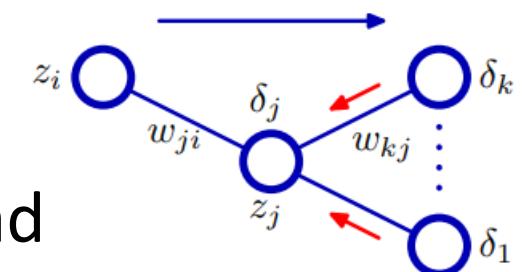
Método para calcular el gradiente (es decir, las derivadas parciales de la función de pérdida con respecto a los pesos)

- *Backprop* es la regla de la cadena!

¿Cómo se entrena las ConvNets?

Dado un ejemplo de entrenamiento

1. Propágalo hacia adelante (*forward pass*), calcula activaciones y el error de salida
2. Vamos hacia atrás (*backward pass*) calculando el “error” (δ_j) de cada unidad j en cada capa
3. Actualizamos los parámetros/pesos usando el gradiente calculado.



$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w}^{(\tau)})$$

Batch gradient descent: empleamos la media de los gradientes de **todos los ejemplos de entrenamiento** para actualizar los pesos

Mini-batch gradient descent: empleamos la media de los gradientes de un **subconjunto de los ejemplos de entrenamiento** para actualizar los pesos

Stochastic gradient descent: empleamos el gradiente de un **único ejemplo de entrenamiento** para actualizar los pesos

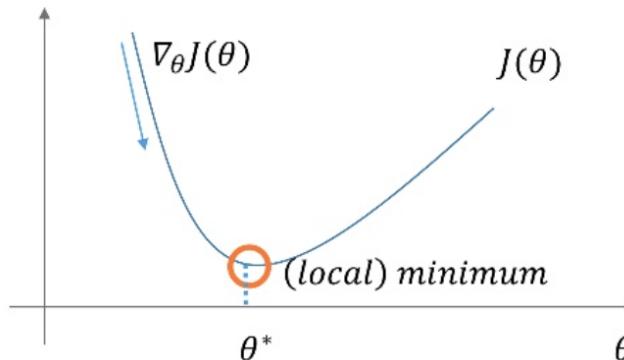
¿Cómo se entrena las ConvNets?

- **Batch**: conjunto de ejemplos en que se divide el conjunto de entrenamiento para entrenar y ajustar los pesos.
 - Batch Gradient Descent. Batch Size = Size of Training Set
 - Stochastic Gradient Descent. Batch Size = 1
 - Mini-Batch Gradient Descent. $1 < \text{Batch Size} < \text{Size of Training Set}$
- **Epoch**: cada vez que la red, durante el entrenamiento, ve el conjunto de entrenamiento al completo.
- **Iteración**: número de batches necesarios para completar un *epoch*.

¿Cómo se entrena las ConvNets?

- El descenso de gradiente tiene un único learning rate para todos los pesos y no cambia en todo el entrenamiento.

$$\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta)$$



[https://www.slideshare.net/
SebastianRuder/optimization
-for-deep-learning](https://www.slideshare.net/SebastianRuder/optimization-for-deep-learning)

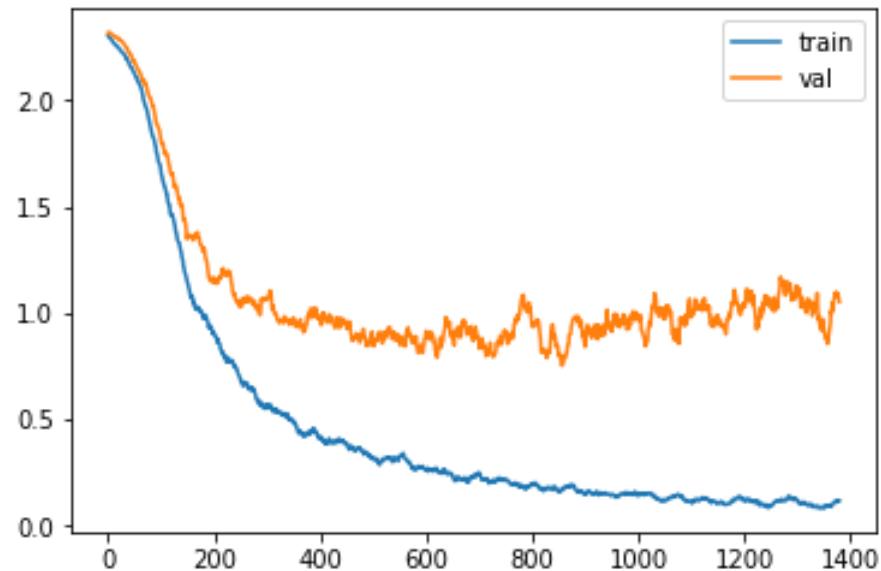
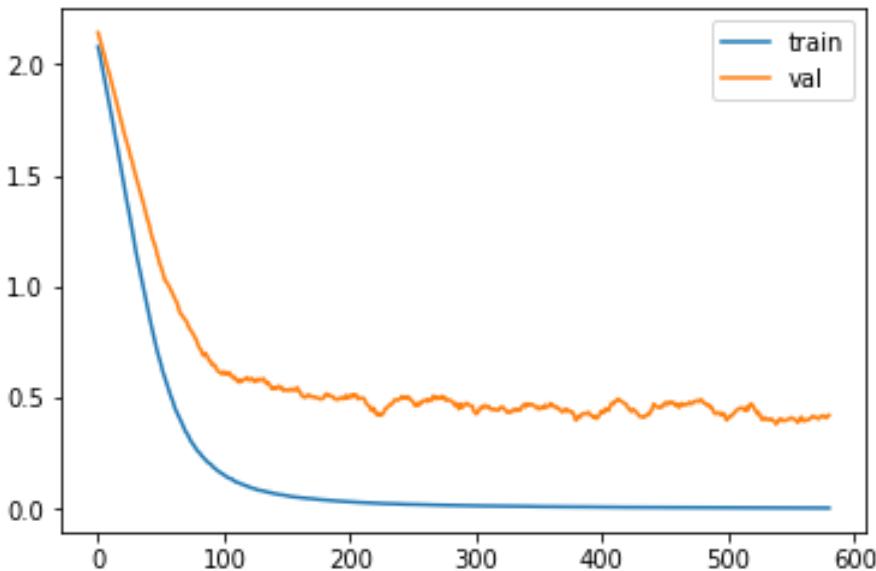
- Hay muchos otros algoritmos de optimización (<https://ruder.io/optimizing-gradient-descent/>). Aquí destacamos uno:
 - Adam [Kingma and Ba, 2015]:
 - Learning rate para cada parámetro
 - Este learning rate es adaptativo.

Formas de mejorar el entrenamiento

- Las redes profundas tienen muchos parámetros
→ necesitan grandes cantidades de datos para entrenar y tienen un alto riesgo de sobreentrenar
- Introduciremos ciertas técnicas para acelerar y/o regularizar el entrenamiento:
 - Early Stopping
 - Aumento de datos
 - Batch normalization y otras formas de normalization
 - Dropout

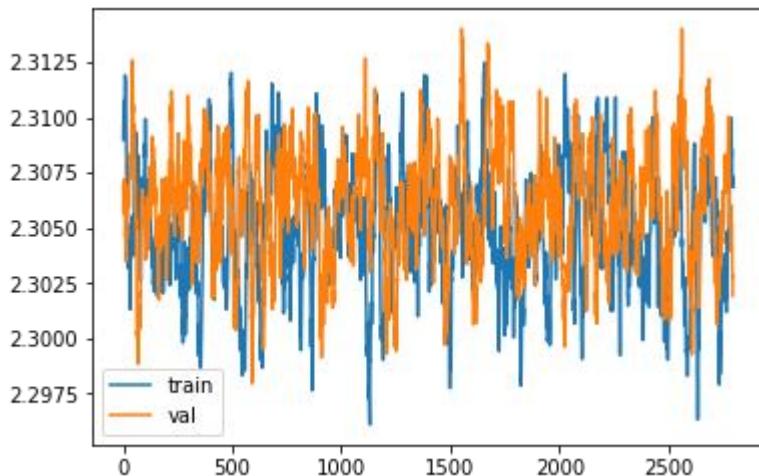
Formas de mejorar el entrenamiento

- Sobreentrenamiento:
 - El modelo no generaliza adecuadamente
 - El modelo es demasiado complejo / tenemos pocos datos

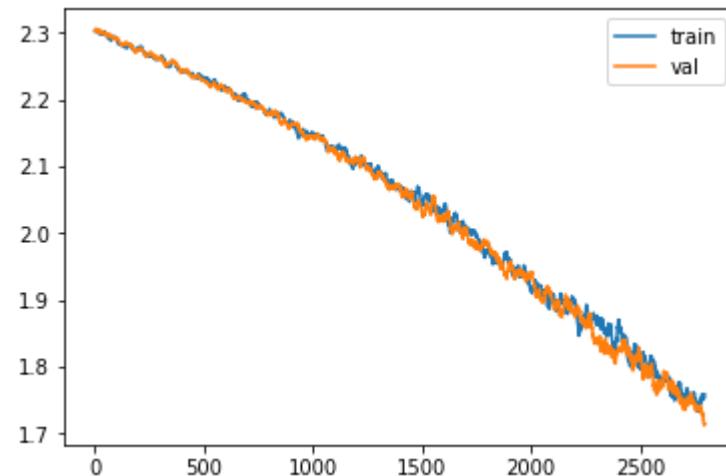


Fíjate en las **curvas de entrenamiento** porque os dan mucha información sobre lo que os puede estar pasando

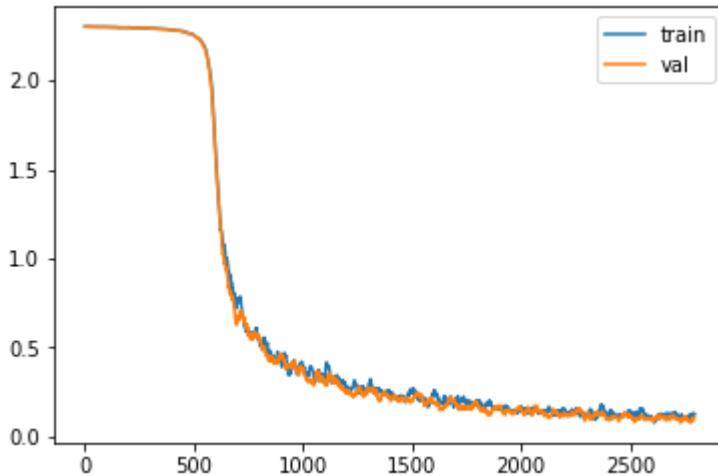
Formas de mejorar el entrenamiento



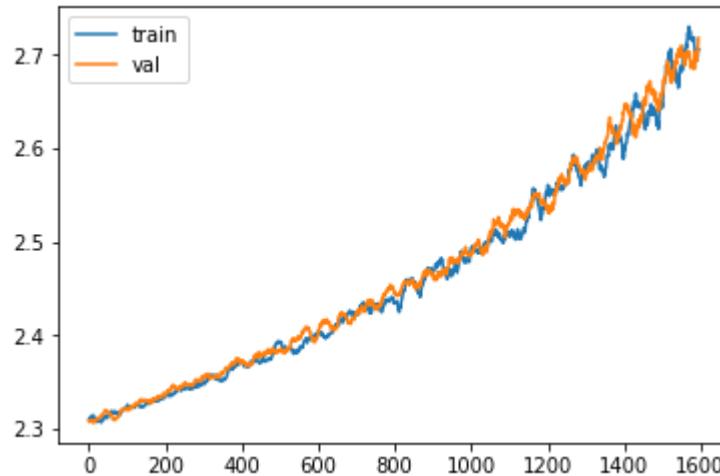
No hay aprendizaje: gradientes no aplicados a los pesos



Todavía no convergió: necesita más tiempo



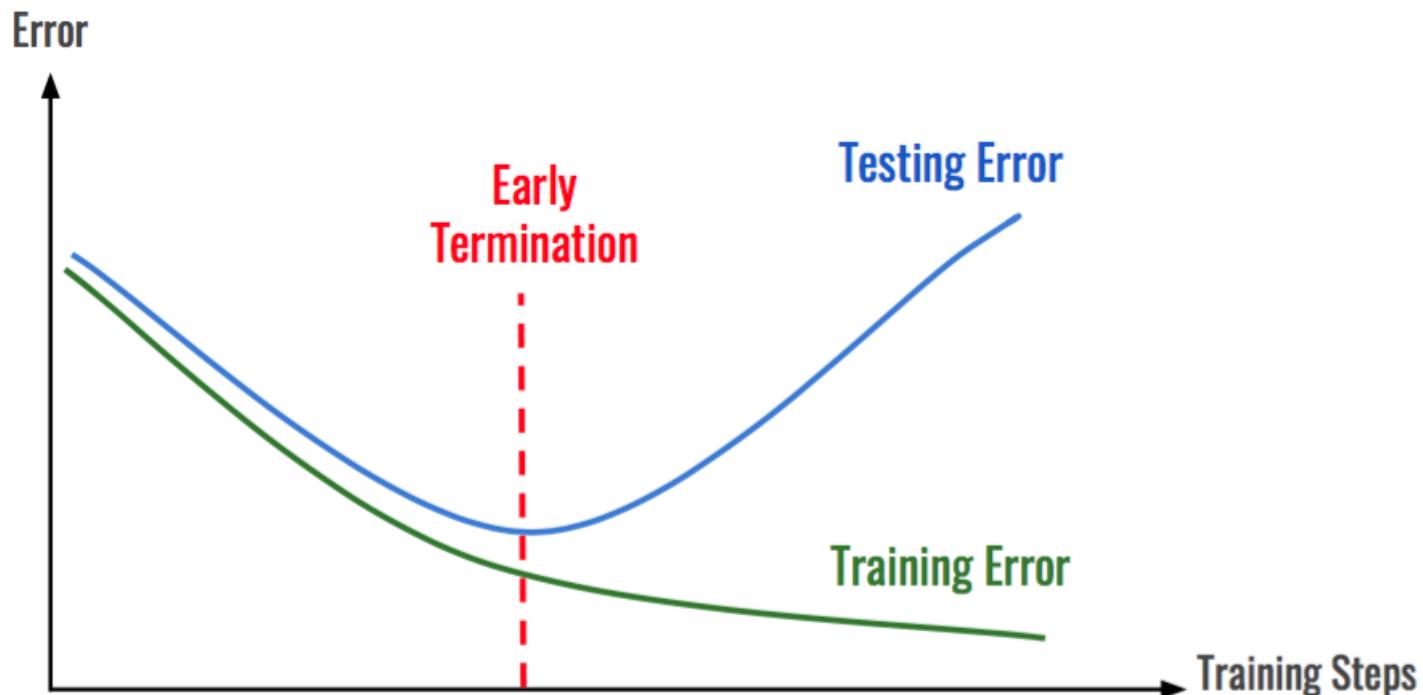
Arranque lento: inicialización de pesos demasiado pequeña



Seguramente estéis aplicando el negativo de los gradientes

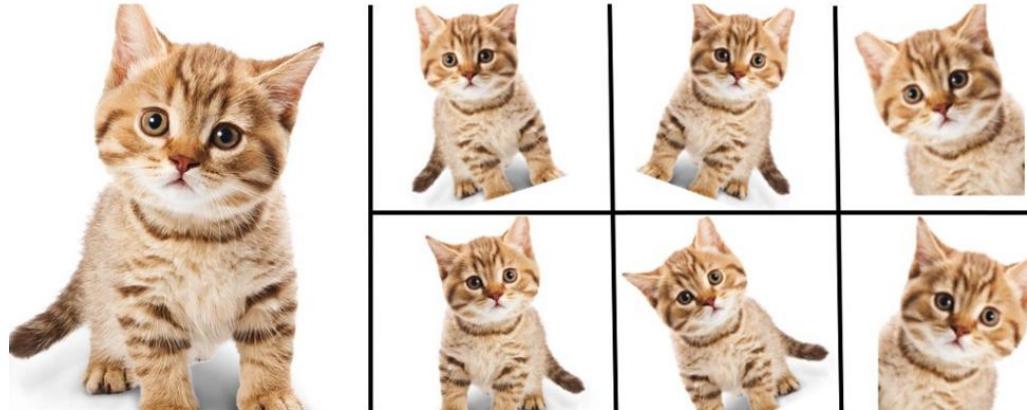
Formas de mejorar el entrenamiento

- Early Stopping



Formas de mejorar el entrenamiento

- Aumento de datos
 - Aplicamos distintas **transformaciones a los datos de entrada** → aumentamos el conjunto de entrenamiento
 - P.ej. en imágenes: rotaciones, cambios de contraste, inserción de ruido, *mirroring*,...
 - Dependiendo del problema, unas transformaciones serán válidas y otras no. P.ej. si quieres clasificar dígitos y rotas mucho un 6, se podría convertir en un 9...



Formas de mejorar el entrenamiento

- Batch normalization [Ioffe & Szegedy, 2015]

- Se sabe que **normalizar las entradas** acelera el **entrenamiento**



- La idea es **normalizar las entradas a las capas ocultas** (generalmente justo antes de aplicar la func. de activ.).

γ, β son parámetros a aprender.
Las estadísticas se calculan para
cada batch y cada canal.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_1 \dots m\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Formas de mejorar el entrenamiento

- Otras formas de normalización

Normalizamos cada batch (entero), tomando cada canal por separado

Normalizamos cada imagen de un batch por separado, incluyendo todos los canales

Normalizamos cada canal de cada imagen (de cada batch) por separado

Normalizamos un conjunto de canales de cada imagen (de cada batch)

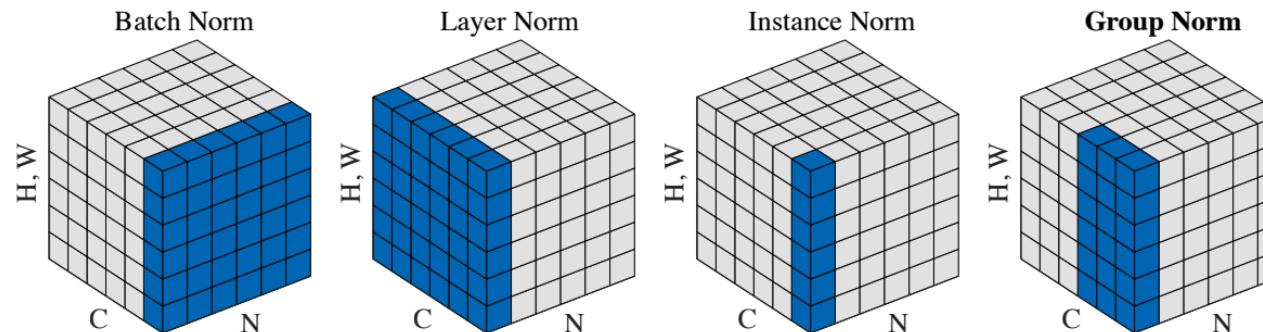
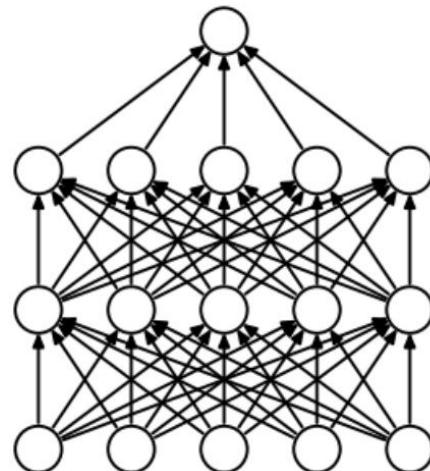


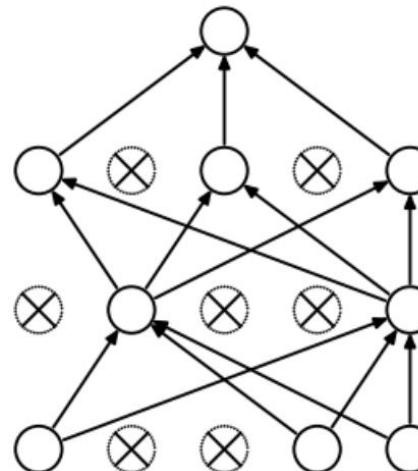
Figure 2. **Normalization methods.** Each subplot shows a feature map tensor, with N as the batch axis, C as the channel axis, and (H, W) as the spatial axes. The pixels in blue are normalized by the same mean and variance, computed by aggregating the values of these pixels.

Formas de mejorar el entrenamiento

- Dropout [Hinton et al., 2012]
 - Eliminamos, de modo aleatorio, unidades ocultas durante el entrenamiento.
 - Fuerza la generalización de las unidades, desacoplándolas unas de otras.
 - Generalmente, se aplica en fully-connected layers, y con diferentes unidades en cada iteración.



(a) Standard Neural Net

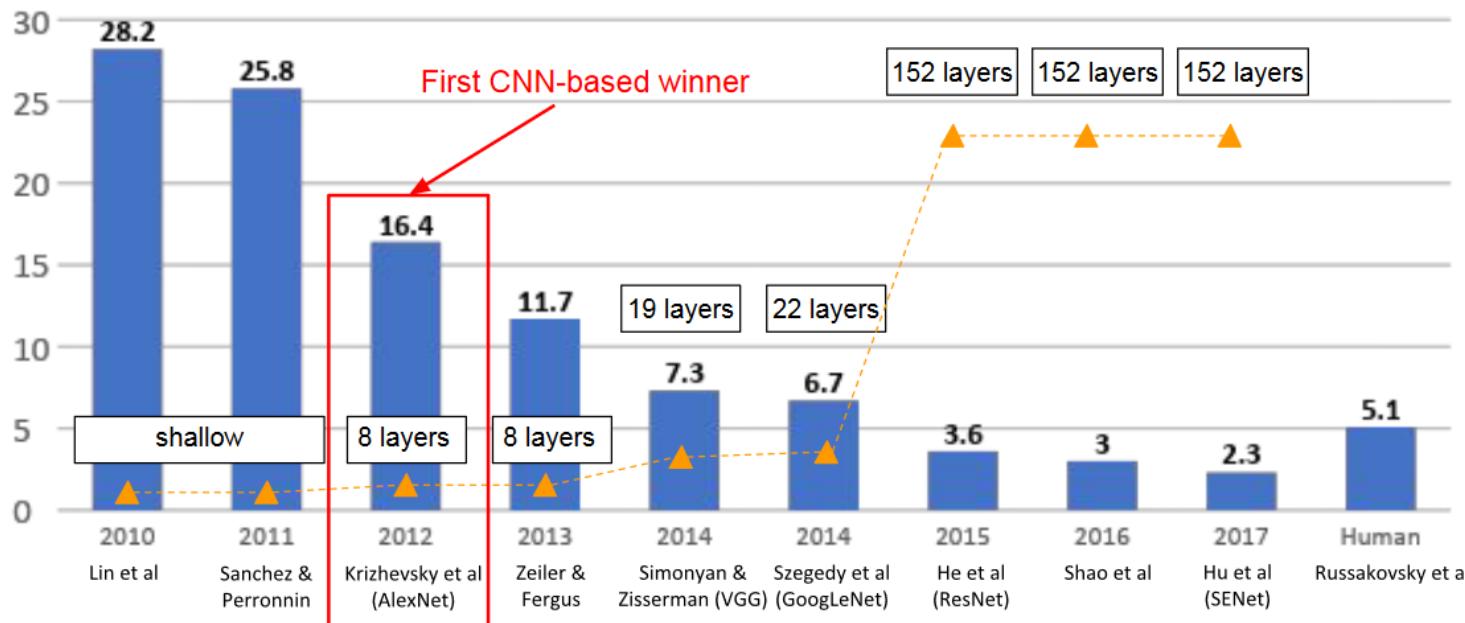


(b) After applying dropout.

Algunos modelos populares

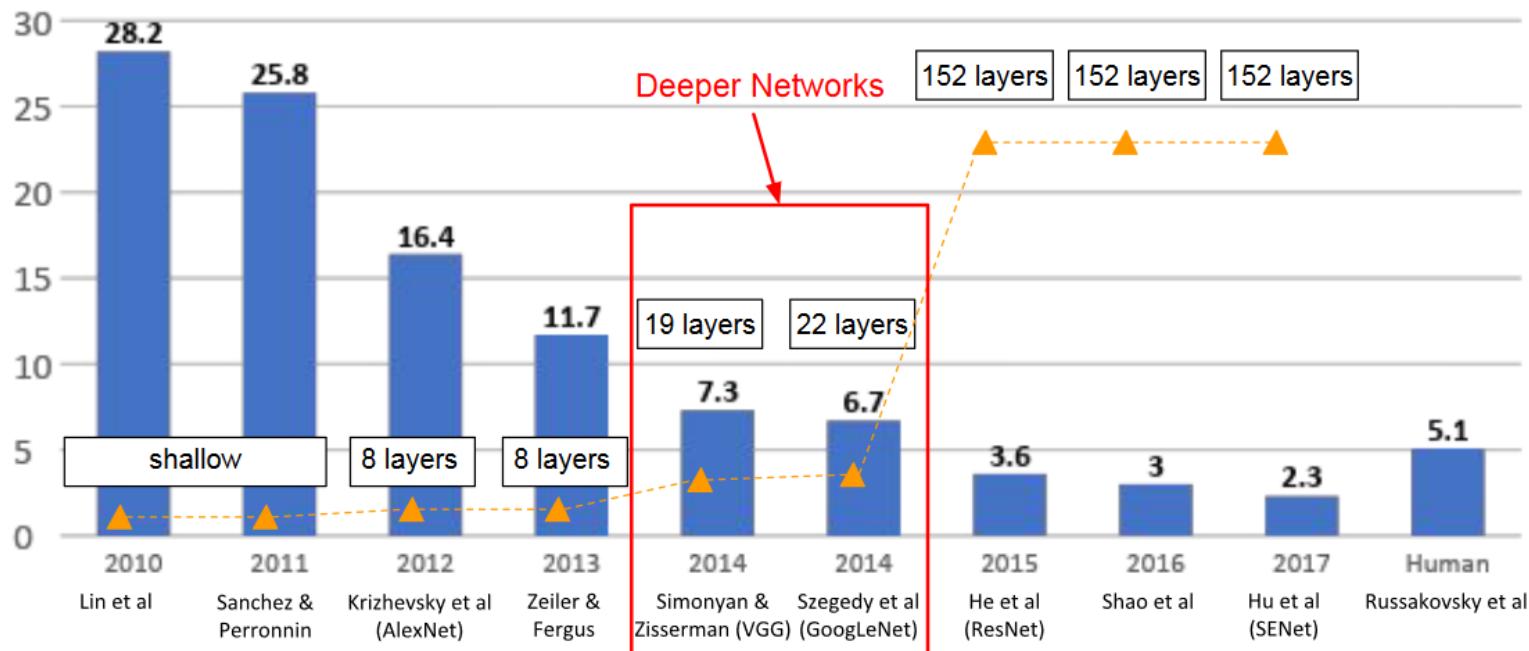
- LeNet-5 [LeCun et al., 1998]
- AlexNet [Krizhevsky et al., 2012]
 - Primer uso de ReLU
 - Supuso el despegue de deep learning

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



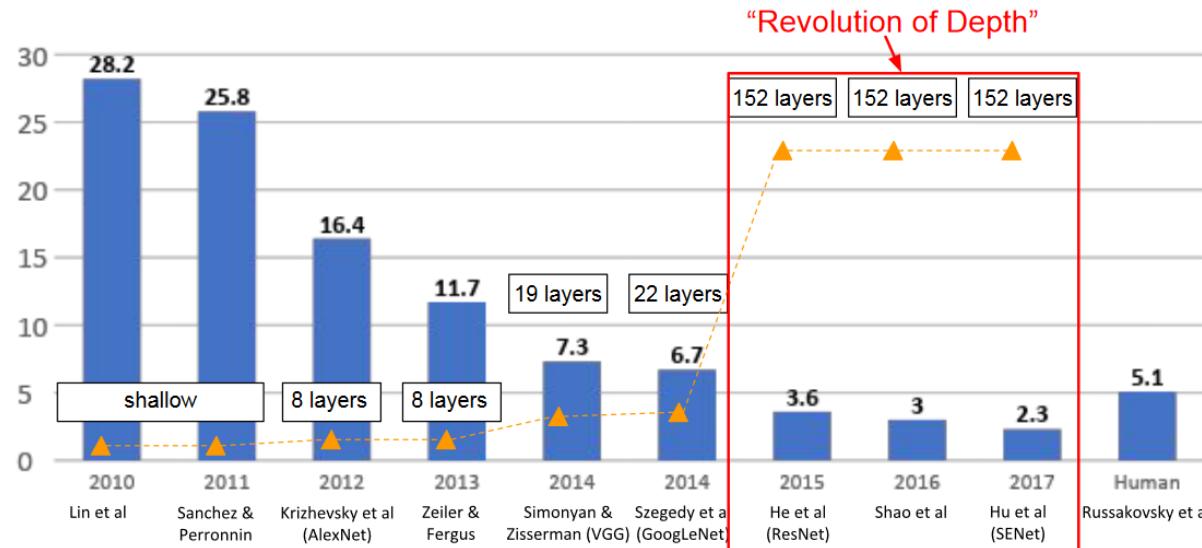
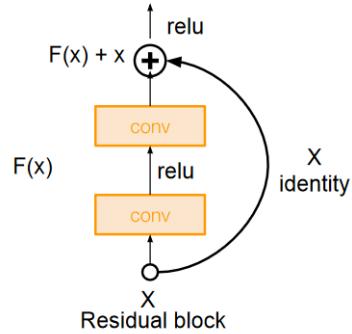
Algunos modelos populares

- VGGNet [Simonyan and Zisserman, 2014]
 - Filtros más pequeños (3x3), redes más profundas (más no-linealidad)
- GoogLeNet [Szegedy et al., 2014]
 - Combina distintos campos receptivos (módulos Inception)
 - Usa convoluciones 1x1 para reducir coste computacional
 - Salidas intermedias auxiliares para “inyectar” gradiente adicional en capas inferiores

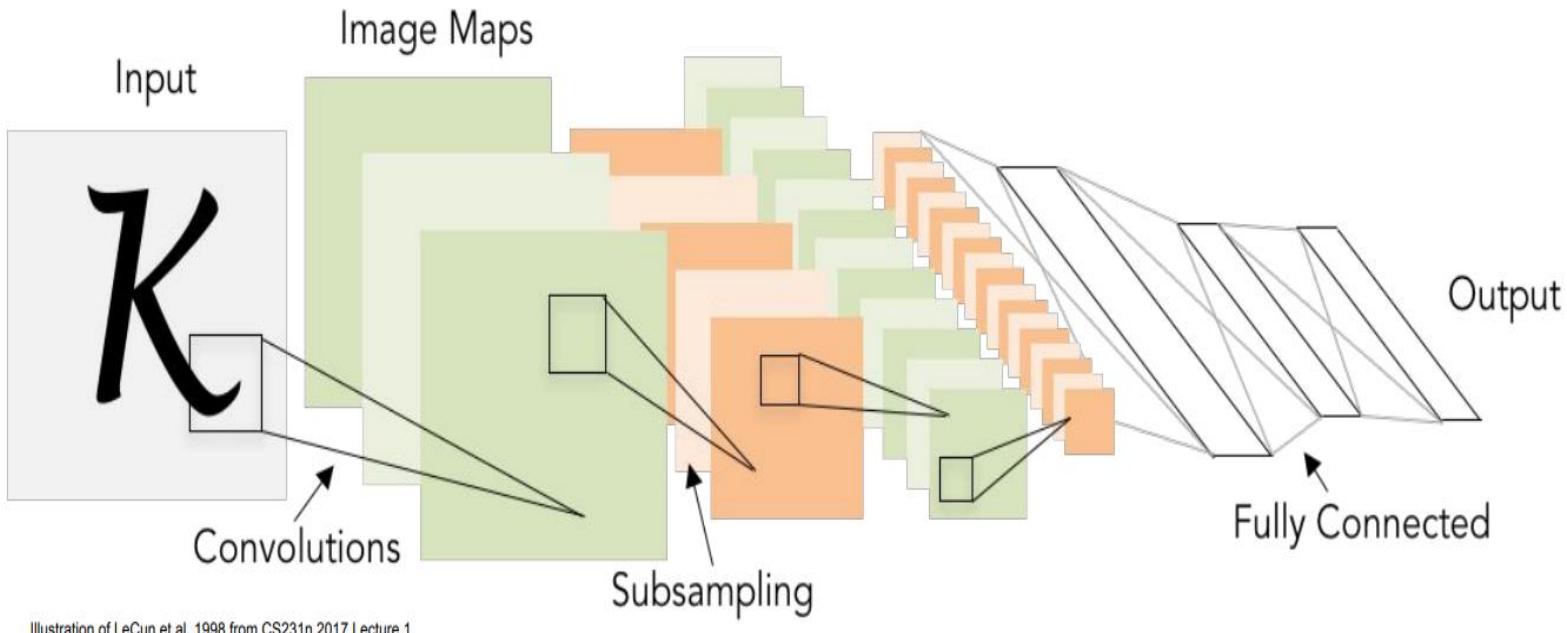


Algunos modelos populares

- ResNet [He et al., 2015]
 - Redes muy profundas usando conexiones residuales (*skip connections* junto con *batch norm* y convoluciones 1x1)
- ResNeXt [Xie et al., 2016]
 - ResNet + estrategia de rutas paralelas inspirada en módulo Inception
- [Shao et al., 2016]: combinación de modelos (Inception, ResNet,...)
- SENet [Hu et al., 2017]: módulo de recalibración de características para aprender a re-pesar los *feature maps*.

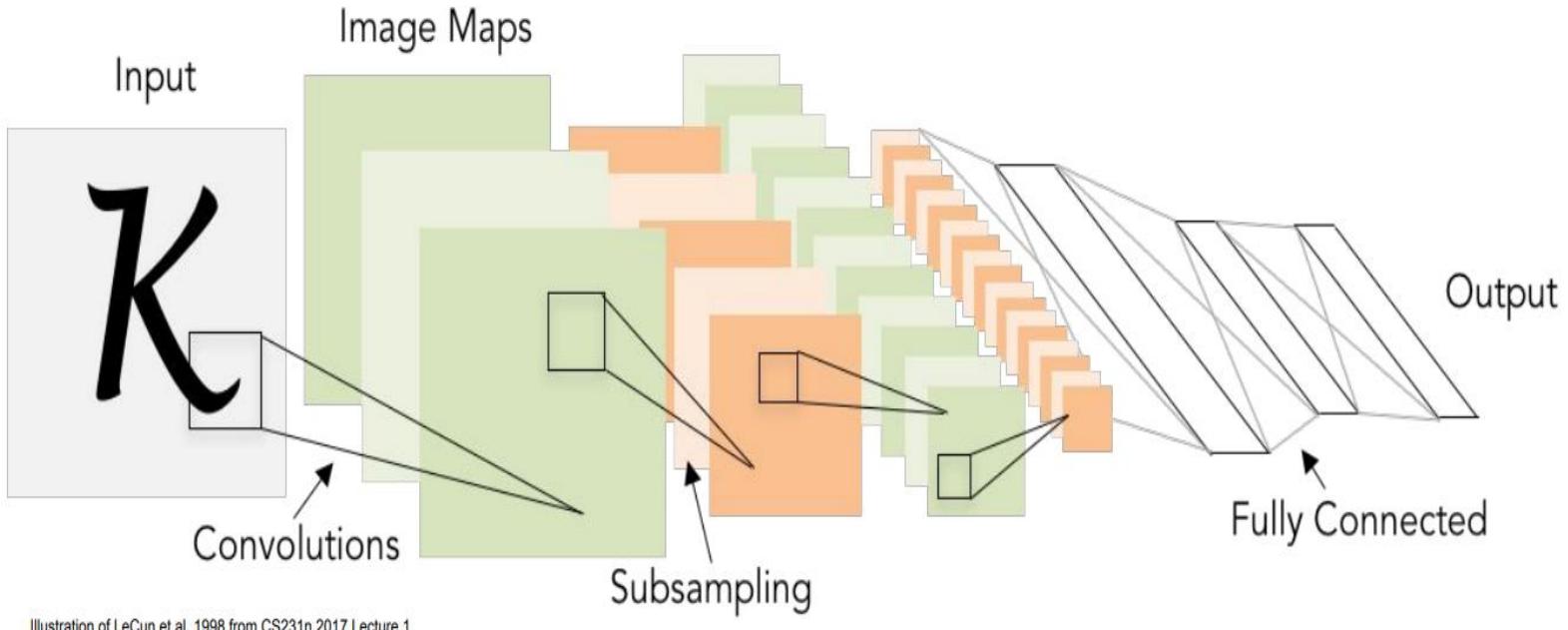


Principales conclusiones



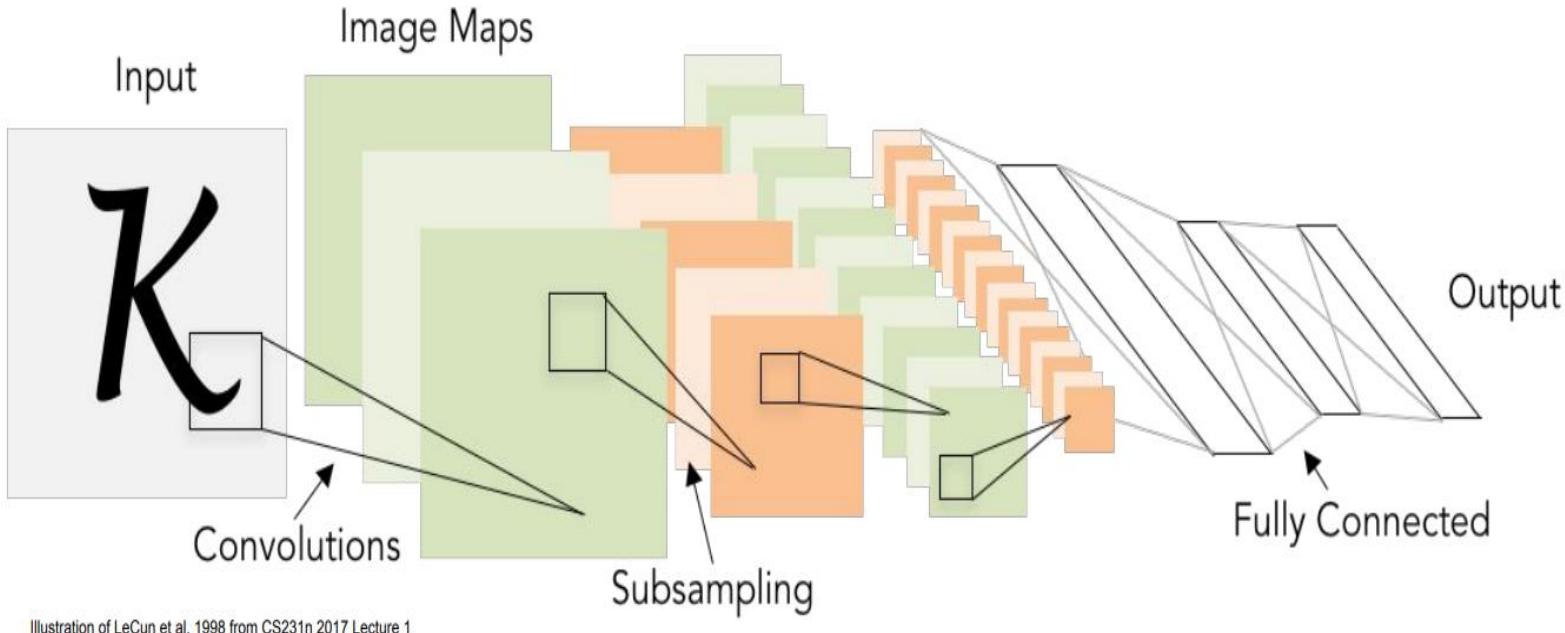
- Las redes convolucionales (ConvNets) son redes neuronales aplicadas comúnmente al análisis de imágenes.
- Emplean convoluciones para extraer características (*feature maps*).
- Las máscaras de convolución se aprenden (pesos de la red).

Principales conclusiones (2)



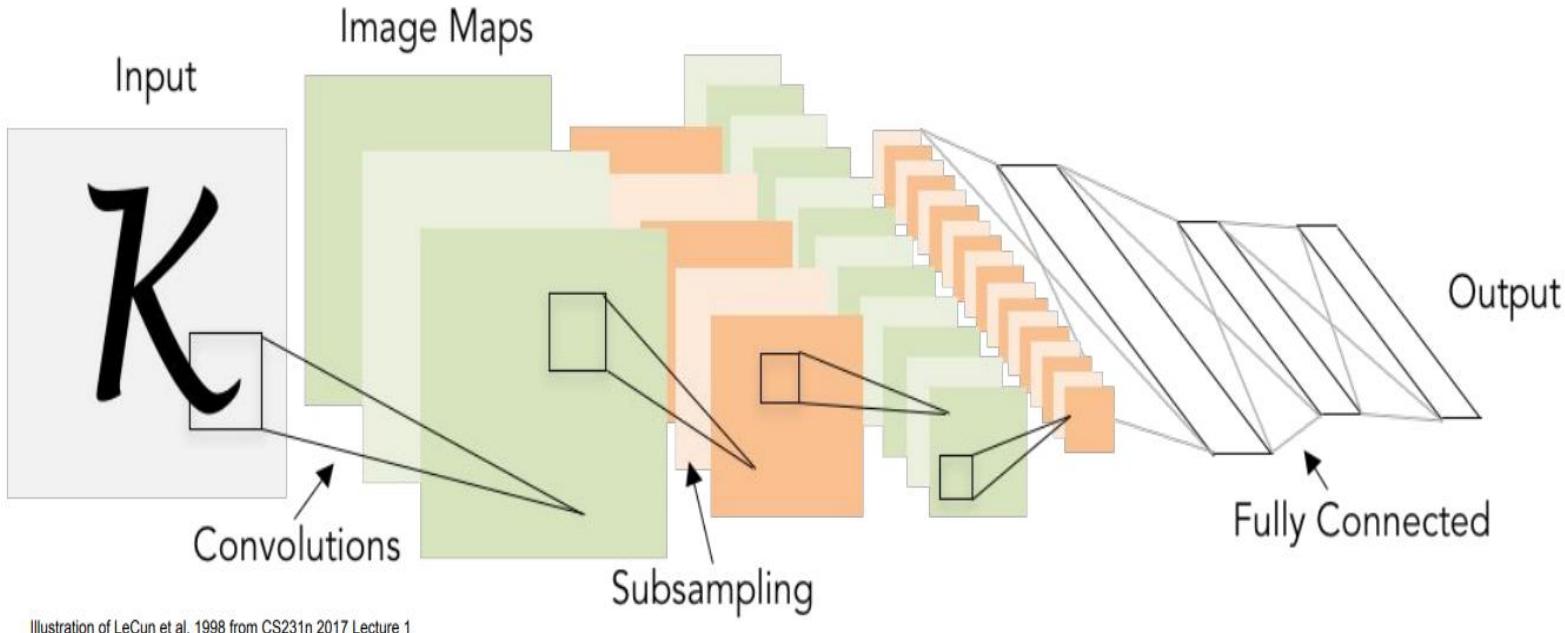
- Pueden contener, o no, capas totalmente conectadas (*fully connected layers*).
- Incluyen capas de *pooling* para reducir dimensionalidad (*subsampling*).

Principales conclusiones (3)



- Emplean como entrada la propia imagen (píxeles directamente).
- Se enlazan operaciones lineales (convolución) con no lineales (función de activación).
- Se aprenden representaciones progresivamente más abstractas.

Principales conclusiones (y 4)



- El entrenamiento es *end-to-end*, extremo a extremo, minimizando una única función de pérdida (*loss*).
- Cuando el aprendizaje es supervisado, generalmente, necesita muchos datos.

Prácticas de Visión por Computador

Grupo 2

Repasso de Aprendizaje Profundo y
Redes Neuronales Convolucionales

Pablo Mesejo

Universidad de Granada

Departamento de Ciencias de la Computación e Inteligencia Artificial



UNIVERSIDAD
DE GRANADA

