

Arquitecturas de Deep Learning

Roberto Muñoz, PhD
Astrónomo y Data Scientist
MetricArts



METRICARTS



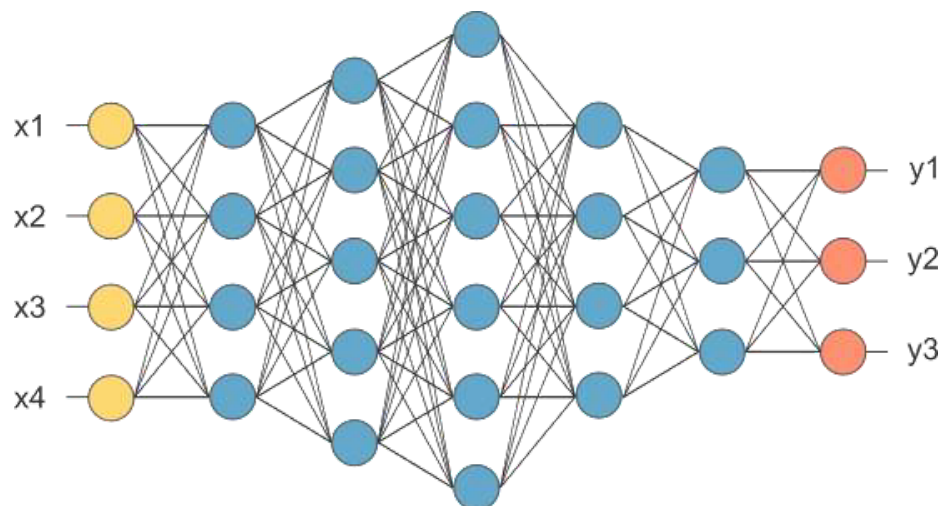
github.com/rpmunoz



[@RobertoKPax](https://twitter.com/RobertoKPax)

Temario

- Feed-forward network
- Arquitecturas más populares
 - Convolutional Neural Networks
 - Recurrent Neural Networks
 - Recursive Neural Networks

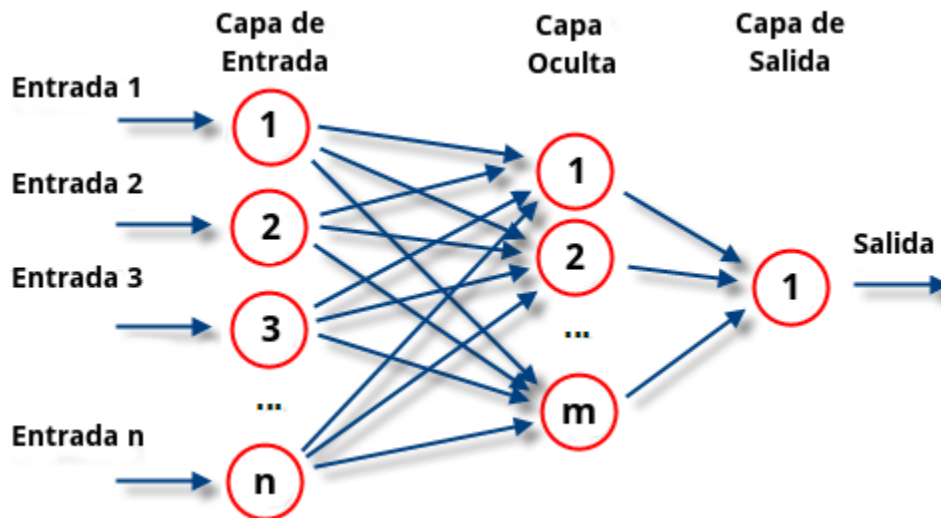


Feed-forward network

REDES NEURONALES PREALIMENTADAS

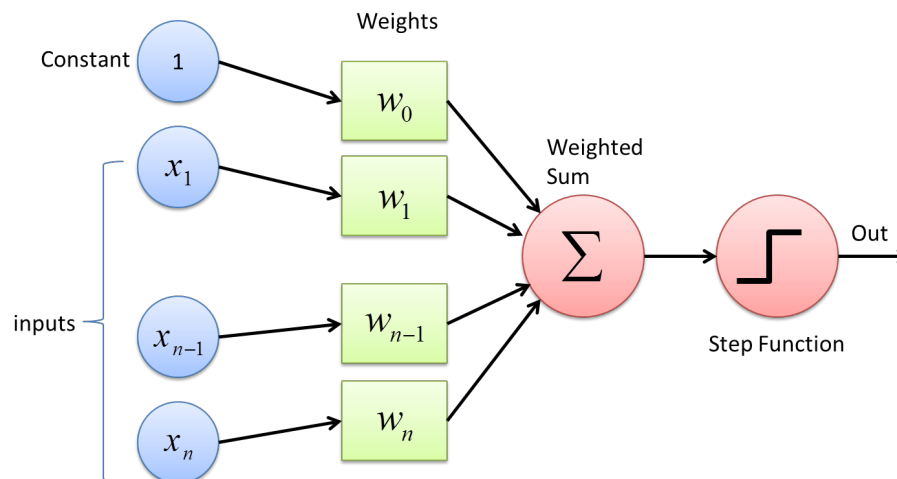
Feed-forward network

- Las Redes neuronales prealimentadas fueron las primeras que se desarrollaron y son el modelo más sencillo.
- En estas redes la información se mueve en una sola dirección: hacia adelante.
- Los principales exponentes de este tipo de arquitectura son el perceptrón y el perceptrón multicapa.



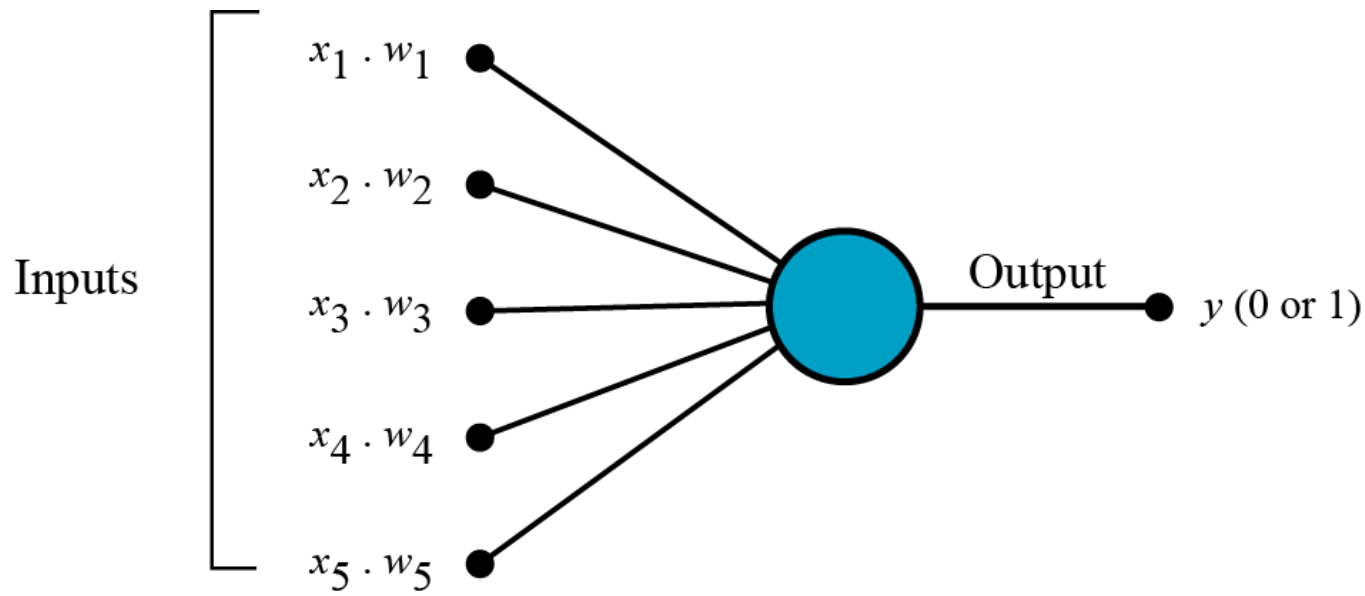
Perceptrón

- Perceptrón lo introduce Frank Rosenblatt (1958)
- Principales componentes
 - Valores de entrada o capa de input
 - Pesos (weights) y sesgo (bias)
 - Suma total
 - Función de activación



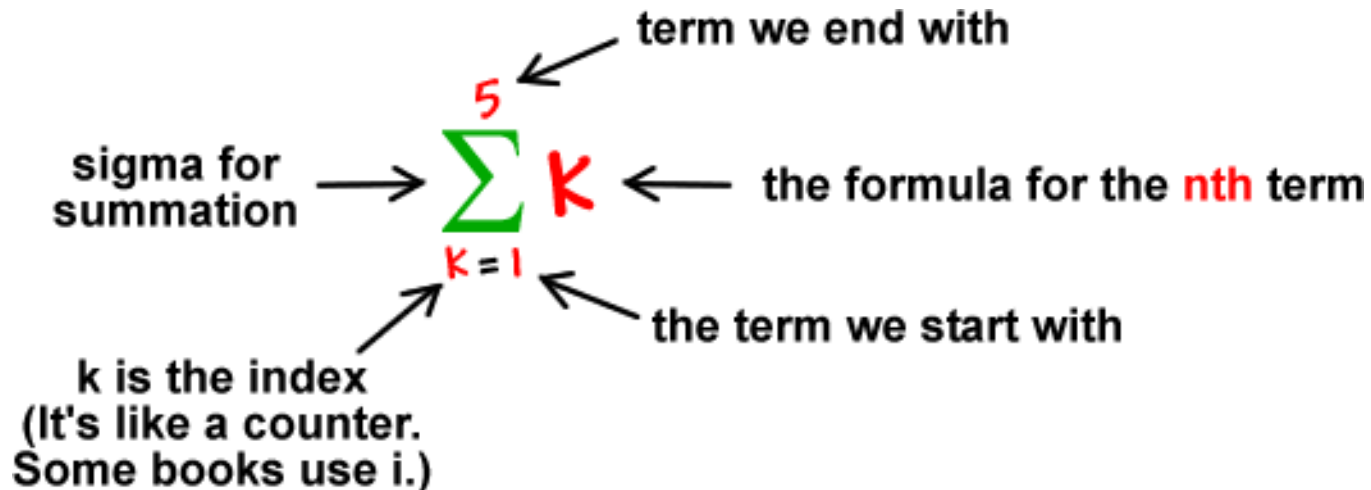
Paso 1

- Cada uno de los valores de entrada \mathbf{x} es multiplicado por un valor de peso \mathbf{w} .
- El resultado se llama $\mathbf{k} = \mathbf{x} \times \mathbf{w}$.











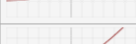
Paso 2

- Sumar todos los valores de **k**. Esa suma se llama suma pesada.



Paso 3

- Usar el valor de la suma pesada como valor de entrada para la función de activación.
- Funciones de activación: lineal, limitante, escalón, sigmoidal, tangente hiperbólica, Gaussiana, ReLU.

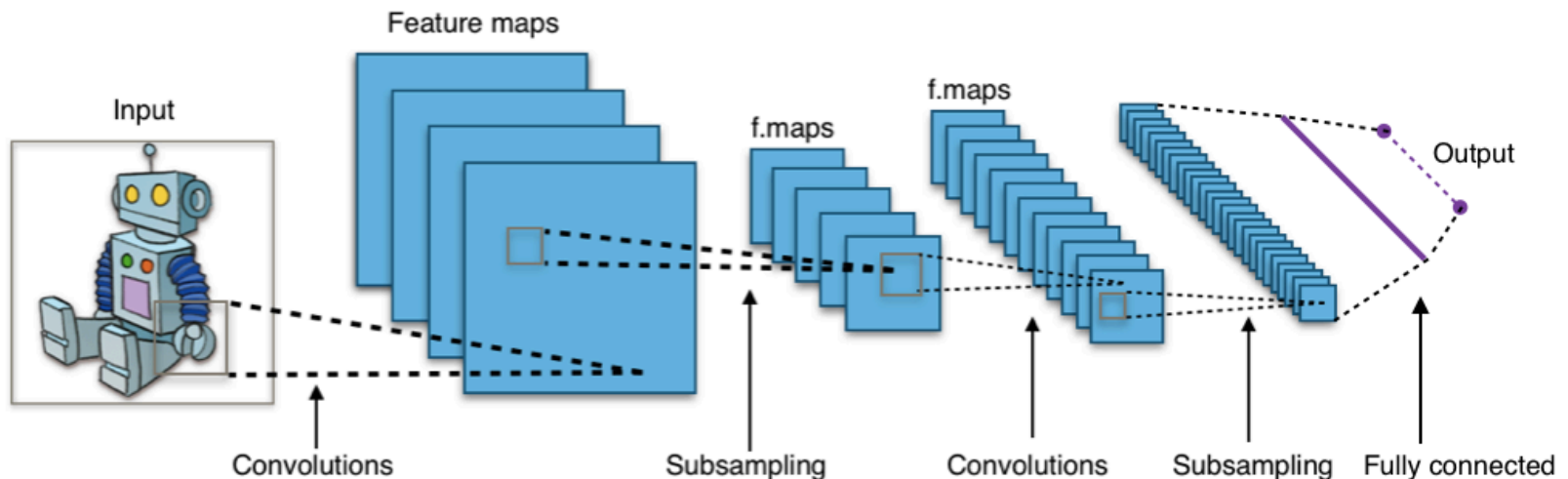
Name	Plot	Equation	Derivative
Identity		$f(x) = x$	$f'(x) = 1$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x \neq 0 \\ ? & \text{for } x = 0 \end{cases}$
Logistic (a.k.a Soft step)		$f(x) = \frac{1}{1 + e^{-x}}$	$f'(x) = f(x)(1 - f(x))$
TanH		$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$	$f'(x) = 1 - f(x)^2$
ArcTan		$f(x) = \tan^{-1}(x)$	$f'(x) = \frac{1}{x^2 + 1}$
Rectified Linear Unit (ReLU)		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Parametric Rectified Linear Unit (PReLU) [2]		$f(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Exponential Linear Unit (ELU) [3]		$f(x) = \begin{cases} \alpha(e^x - 1) & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$	$f'(x) = \begin{cases} f(x) + \alpha & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
SoftPlus		$f(x) = \log_e(1 + e^x)$	$f'(x) = \frac{1}{1 + e^{-x}}$

CNN

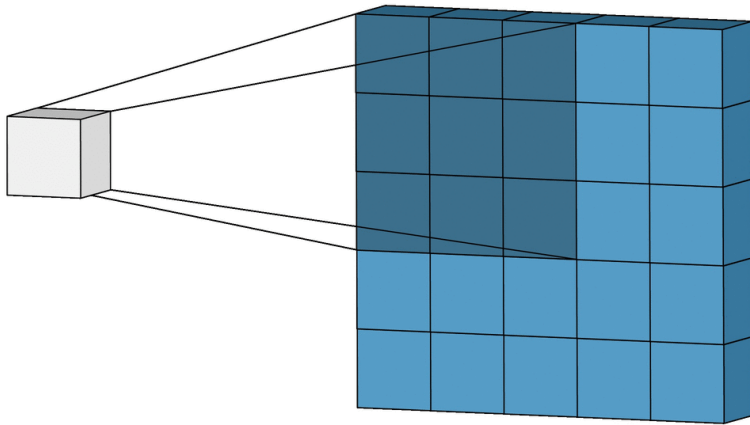
REDES CONVOLUCIONALES

Convolutional neural network

- Las redes neuronales convolucionales son similares a las redes neuronales ordinarias.
- Se componen de neuronas que tienen pesos y sesgos que pueden aprender.
- Cada neurona recibe algunas entradas, realiza un producto escalar y luego aplica una función de activación.



Convolución



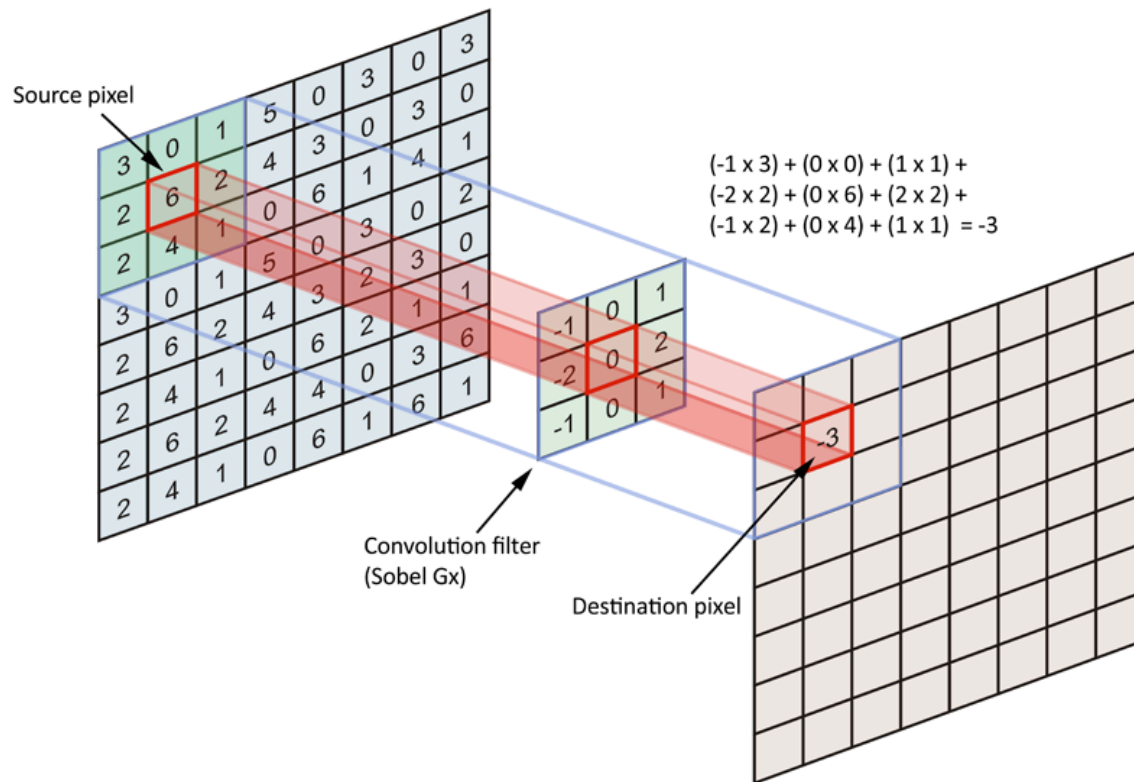
0	1	2
2	2	0
0	1	2

Kernel 3x3

3_0	3_1	2_2	1	0
0_2	0_2	1_0	3	1
3_0	1_1	2_2	2	3
2	0	0	2	2
2	0	0	0	1

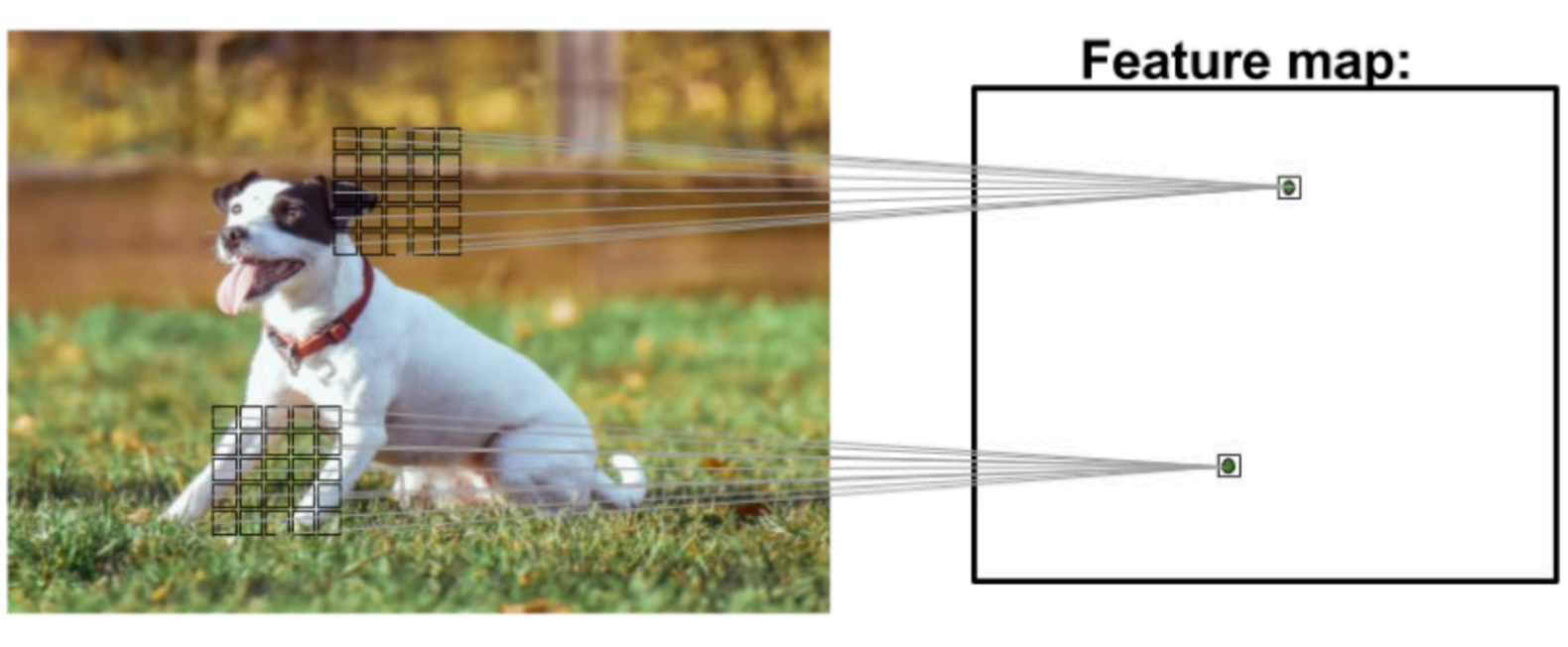
12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

Convolución



Convolutional neural network

- CNN calcula mapas de features a partir de una imagen de entrada. Cada elemento proviene de un región local de píxels.
- La región local de píxels se denomina local receptive field o campos receptivos.



Convolutional neural network

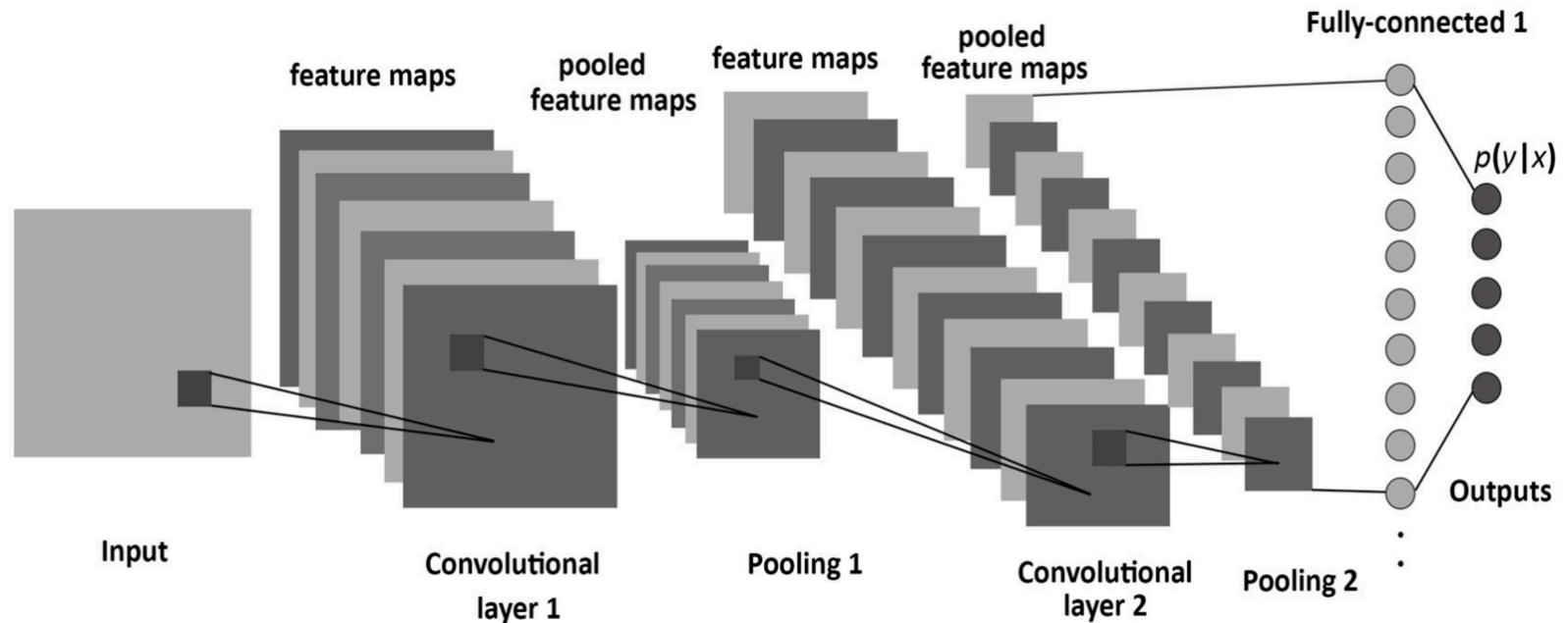
- ¿Porqué funcionan tan bien en imágenes?
 - **Conectividad local:** Cada elemento del mapa de features está conectado con una pequeña región de píxels/patronos. Función de respuesta alta de los filtros aprendidos. Se parte con representaciones de pequeñas regiones y luego se escala a regiones de mayor tamaño.
 - **Pesos compartidos:** Cada filtro se replica a lo largo del campo de visión completo. Invariancia respecto a translación.

Convolutional neural network

- Las redes neuronales convolucionales van a estar construidas por 3 tipos de capas,
 - Capa convolucional
 - Capa de reducción o pooling. Reducir cantidad de parámetros, características más comunes.
 - Capa clasificadora totalmente conectada. Suelen ser las últimas capas de la red.
- Algunas arquitecturas CNN
 - LeNet, AlexNet, VGGNet, GoogleNet, ResNet, etc.

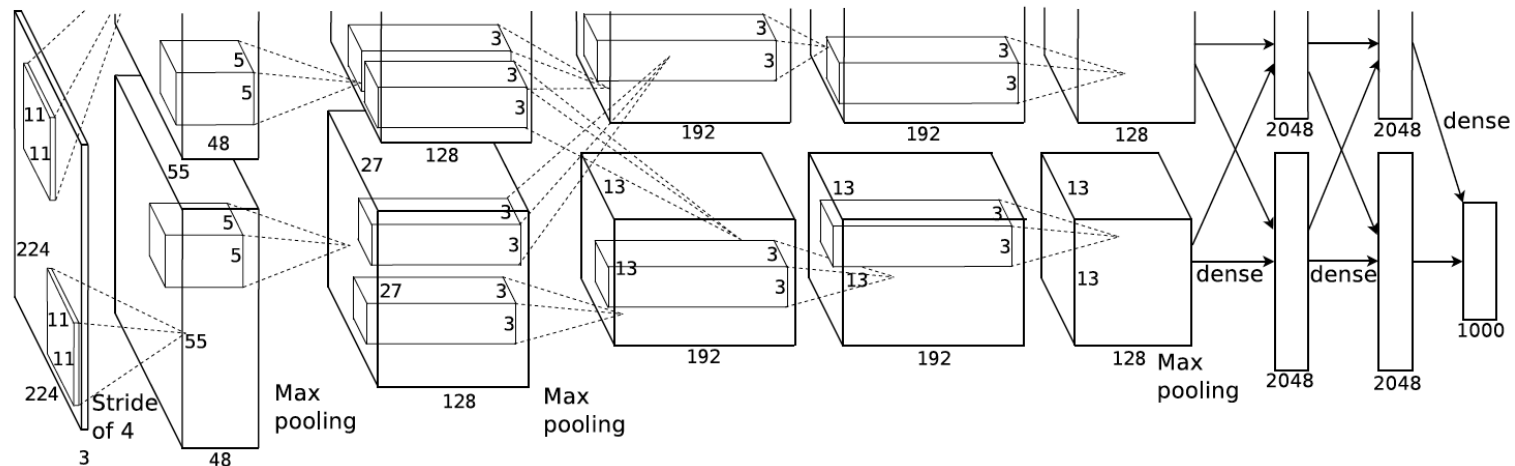
LeNet-5

- Propuesta por Yan LeCun et al. (1998)
- Usada por bancos para reconocer números escritos a mano en cheques digitalizados.



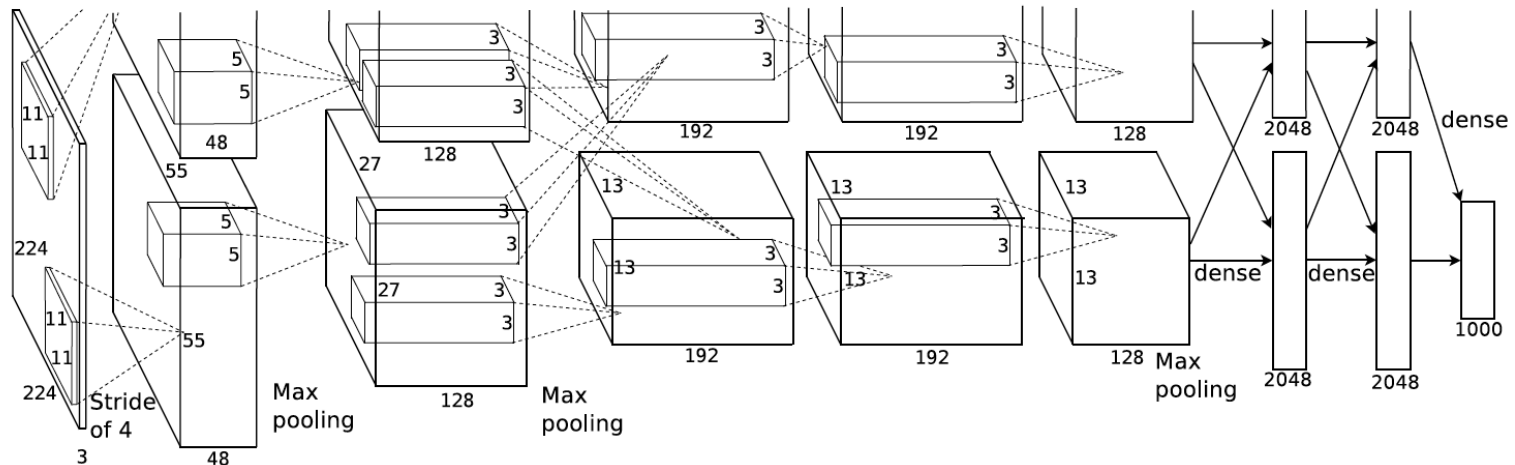
AlexNet

- Propuesta por Krizhevsky et al. (2012) como solución al challenge de ImageNet 2012.
- Clasificación de pequeñas imágenes.
- Top-5 score: Etiqueta anotada es una de las 5 predicciones con mayor probabilidad.
- Redujo error top-5 de 26,2% a 15,3%.



AlexNet

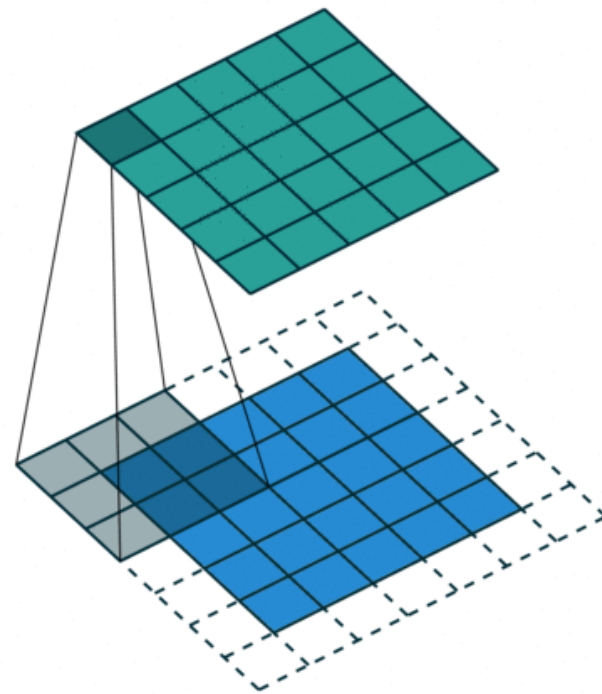
- Contiene 5 capas convolucionales y 3 capas densas (fully-connected).
- Se aplica ReLU después de cada capa convolucional y capa densa.
- Se aplica dropout antes de la primera y segunda capa densa.



Padding

- Agregar pixels extras en los bordes de la imagen

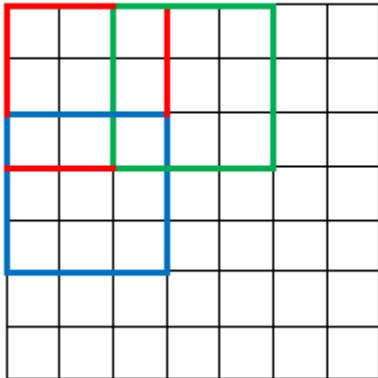
0	0	0	0	0	0
0	35	19	25	6	0
0	13	22	16	53	0
0	4	3	7	10	0
0	9	8	1	3	0
0	0	0	0	0	0



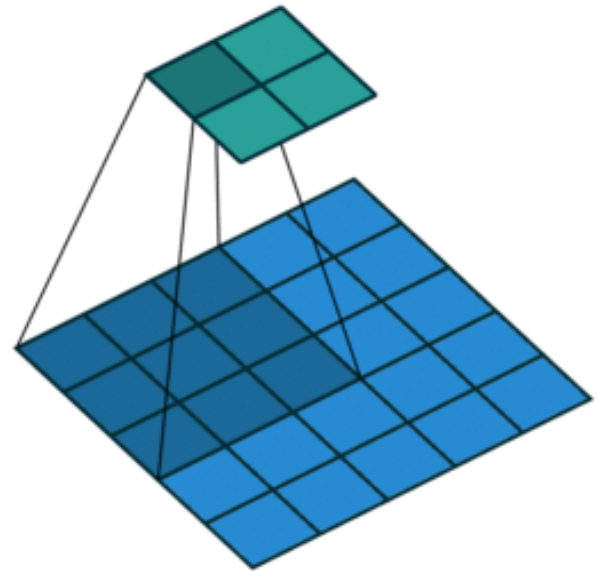
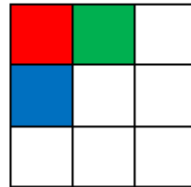
Striding

- Saltarse algunas posiciones en la imagen.
Stride=2 es dar pasos de 2 pixels

7 x 7 Input Volume

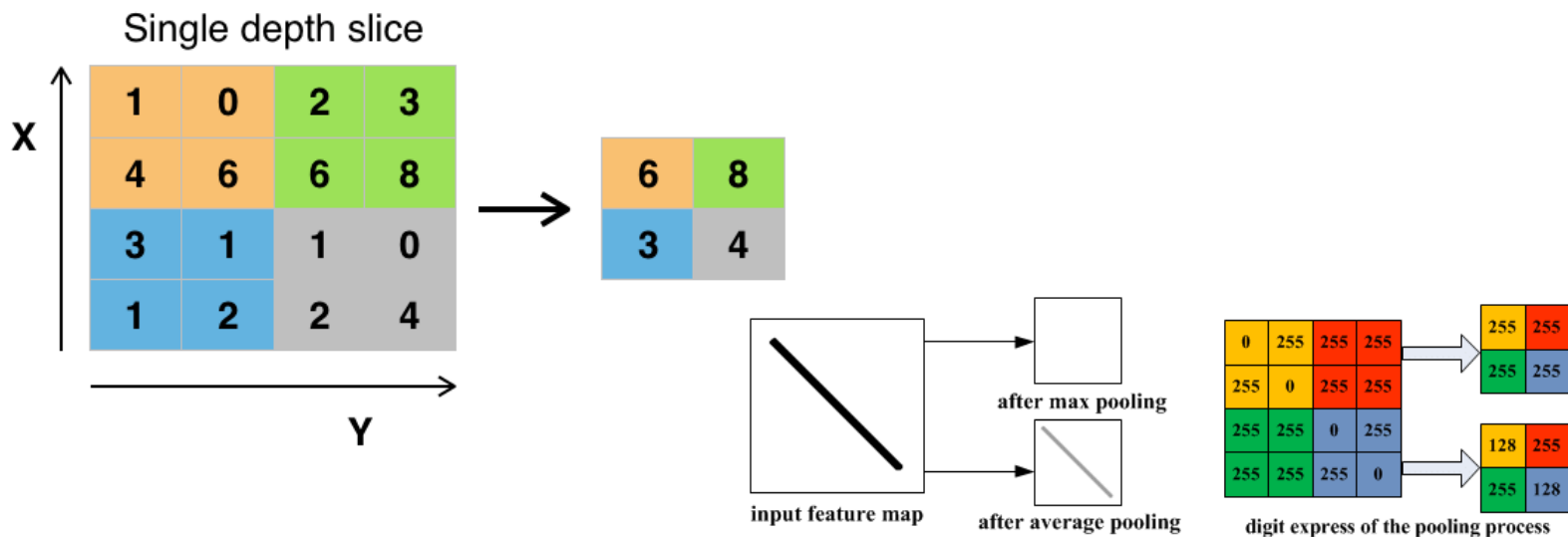


3 x 3 Output Volume

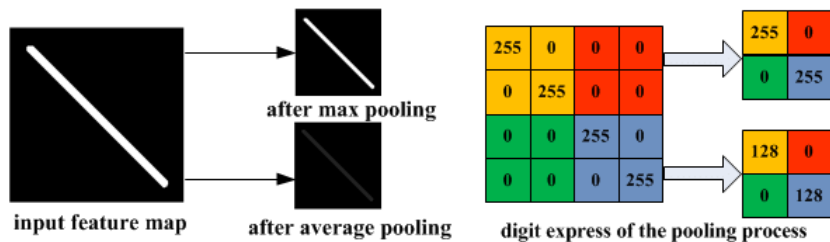


Pooling

- Reducir dimensionalidad de los features calculando el valor máximo o promedio agrupando pixels



(a) Illustration of max pooling drawback



(b) Illustration of average pooling drawback

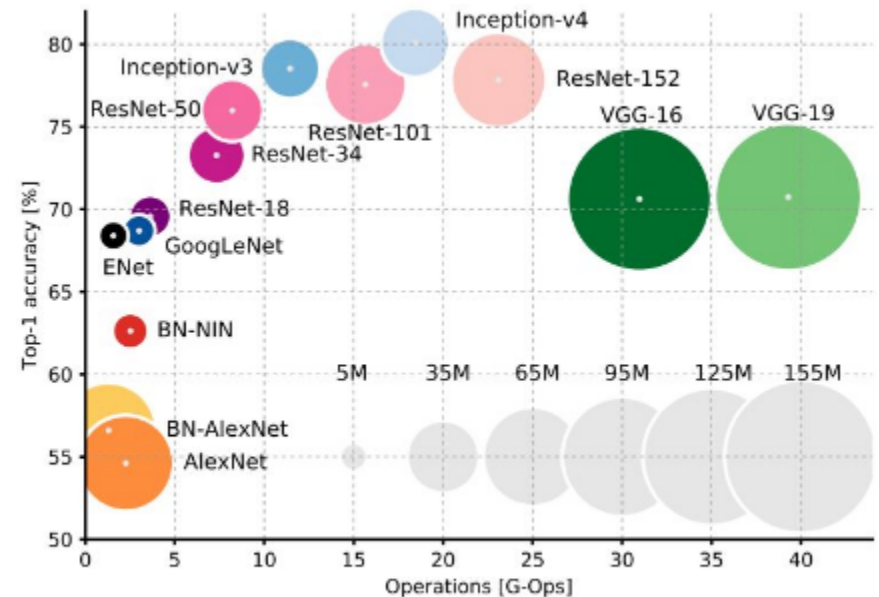
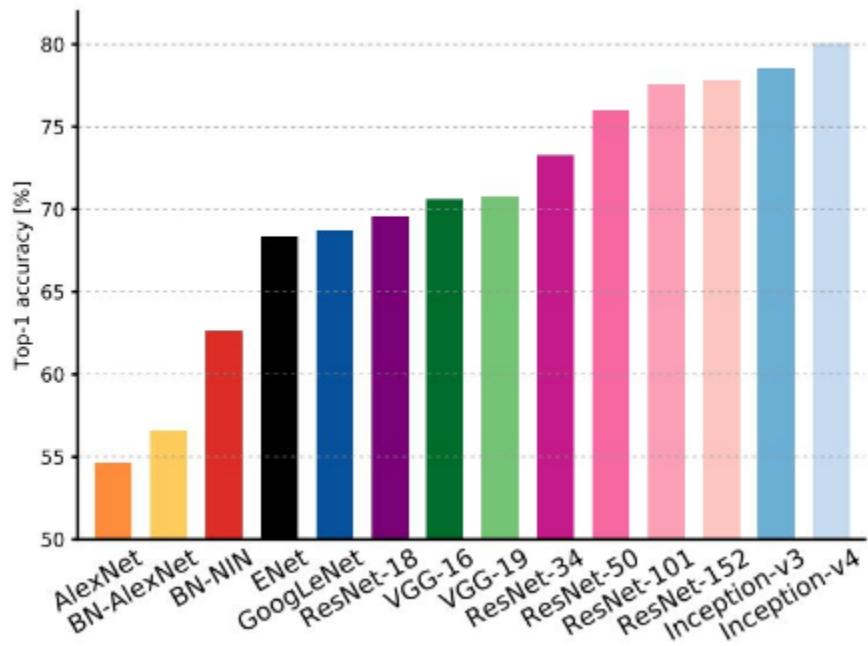
AlexNet – Parámetros

Size / Operation	Filter	Depth	Stride	Padding	Number of Parameters	Forward Computation
3 * 227 * 227						
Conv1 + Relu	11 * 11	96	4		$(11 \cdot 11 \cdot 3 + 1) \cdot 96 = 34944$	$(11 \cdot 11 \cdot 3 + 1) \cdot 96 \cdot 55 \cdot 55 = 105705600$
96 * 55 * 55						
Max Pooling	3 * 3		2			
96 * 27 * 27						
Norm						
Conv2 + Relu	5 * 5	256	1		$2(5 \cdot 5 \cdot 96 + 1) \cdot 256 = 614656$	$(5 \cdot 5 \cdot 96 + 1) \cdot 256 \cdot 27 \cdot 27 = 448084224$
256 * 27 * 27						
Max Pooling	3 * 3		2			
256 * 13 * 13						
Norm						
Conv3 + Relu	3 * 3	384	1		$1(3 \cdot 3 \cdot 256 + 1) \cdot 384 = 885120$	$(3 \cdot 3 \cdot 256 + 1) \cdot 384 \cdot 13 \cdot 13 = 149585280$
384 * 13 * 13						
Conv4 + Relu	3 * 3	384	1		$1(3 \cdot 3 \cdot 384 + 1) \cdot 384 = 1327488$	$(3 \cdot 3 \cdot 384 + 1) \cdot 384 \cdot 13 \cdot 13 = 224345472$
384 * 13 * 13						
Conv5 + Relu	3 * 3	256	1		$1(3 \cdot 3 \cdot 384 + 1) \cdot 256 = 884992$	$(3 \cdot 3 \cdot 384 + 1) \cdot 256 \cdot 13 \cdot 13 = 149563648$
256 * 13 * 13						
Max Pooling	3 * 3		2			
256 * 6 * 6						
Dropout (rate 0.5)						
FC6 + Relu					$256 \cdot 6 \cdot 6 \cdot 4096 = 37748736$	$256 \cdot 6 \cdot 6 \cdot 4096 = 37748736$
4096						
Dropout (rate 0.5)						
FC7 + Relu					$4096 \cdot 4096 = 16777216$	$4096 \cdot 4096 = 16777216$
4096						
FC8 + Relu					$4096 \cdot 1000 = 4096000$	$4096 \cdot 1000 = 4096000$
1000 classes						
Overall					$62369152 = 62.3 \text{ million}$	$1135906176 = 1.1 \text{ billion}$
Conv VS FC					Conv: 3.7 million (6%) , FC: 58.6 million (94%) Conv: 1.08 billion (95%) , FC: 58.6 million (5%)	

AlexNet

- La red tiene 62,3 millones de parámetros
- Necesita de 1.100 millones de unidades cómputo en un solo forward pass.
- Las capas convolucionales representan el 6% de todos los parámetros. Consumen cerca del 95% del cómputo.
- Capas convolucionales tienen pocos parámetros y requieren un elevado número de cálculos.
- Capas densas o fully-connected tienen muchos parámetros y requiere pocos cálculos.

CNN architectures



An Analysis of Deep Neural Network Models for Practical Applications, 2017.