



Argentina
programa
4.0



Universidad
Nacional
de San Martín

Módulo 3

Aprendizaje Automático



Argentina
programa
4.0



Universidad
Nacional
de San Martín

Módulo 3

Aprendizaje Automático

Semana X Convolutional Neural Networks (CNN)

Contenidos del módulo

ML Clásico

- Árboles de Decisión
- Métodos de Ensemble
 - Bagging / Pasting → Random Forests
 - Boosting
- Support Vector Machines

Deep Learning

- Redes Neuronales
- Redes Neuronales Convolucionales
- Auto-Encoders / Auto-Encoders Variacionales
- Redes Neuronales Recurrentes (LSTM, otras)
- Extras:
 - Generative Adversarial Networks (GAN)
 - Reinforcement Learning

Contenidos del módulo

ML Clásico

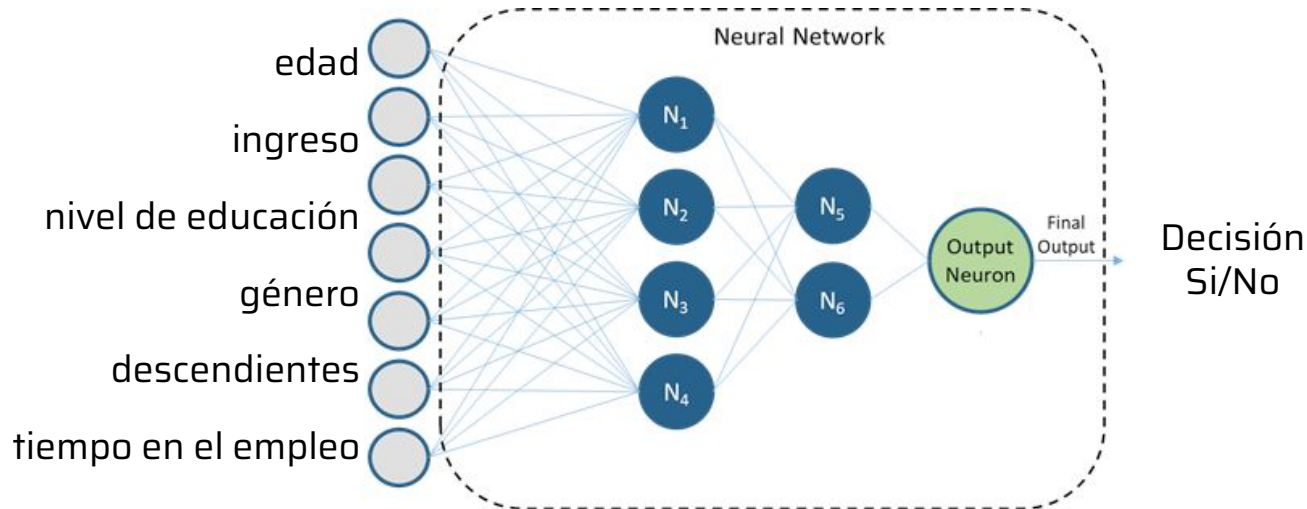
- Árboles de Decisión
- Métodos de Ensemble
 - Bagging / Pasting → Random Forests
 - Boosting
- Support Vector Machines

Deep Learning

- Redes Neuronales
- Redes Neuronales Convolucionales
- Auto-Encoders / Auto-Encoders Variacionales
- Redes Neuronales Recurrentes (LSTM, otras)
- Extras:
 - Generative Adversarial Networks (GAN)
 - Reinforcement Learning

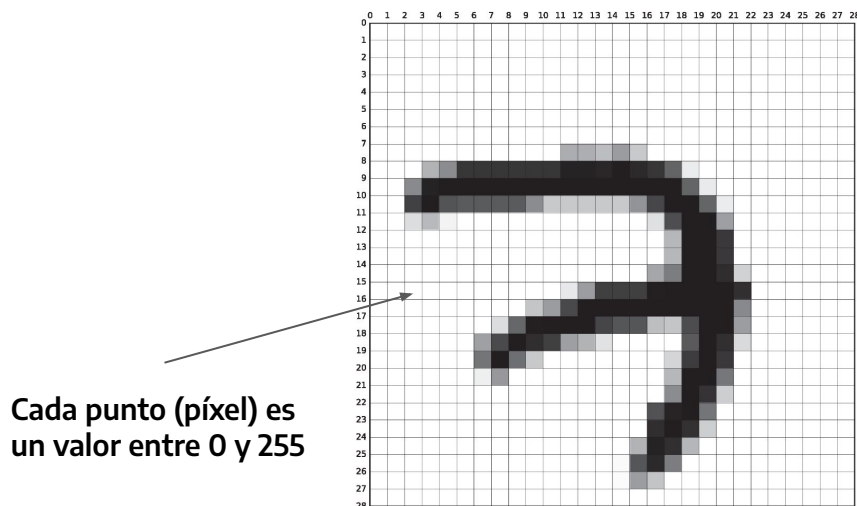
Redes totalmente conectadas

- Vimos la semana pasada las redes totalmente conectadas
- Funcionan bien para características independientes, pero mencionamos que para imágenes no eran lo ideal
- Vamos a introducir las redes convolucionales y vamos a ver como se usan para datos con orden espacial, primero repasemos MNIST e introduzcamos CIFAR-10



MNIST

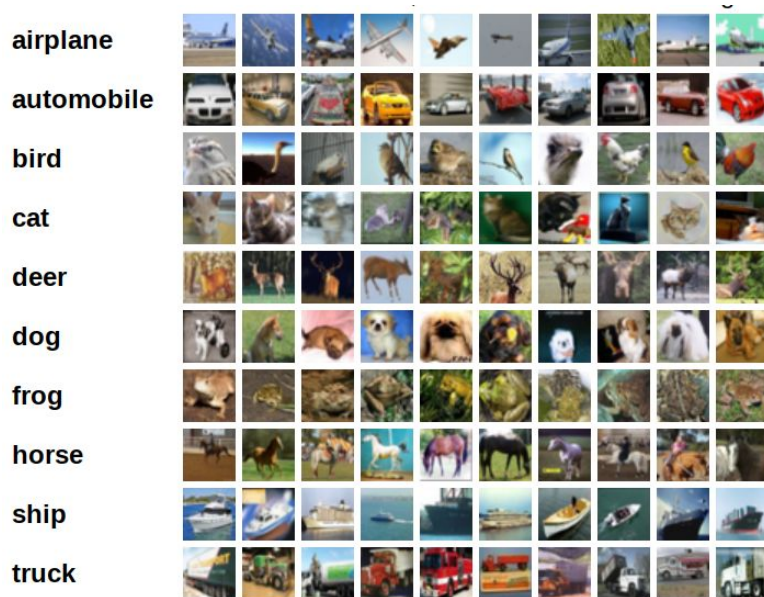
- MNIST (Modified NIST) (*)
- Es una serie de imágenes de dígitos escritos a mano alzada, entre 0 y 9.
- Cada dígito está representado por una imagen en blanco y negro y tiene su etiqueta correspondiente



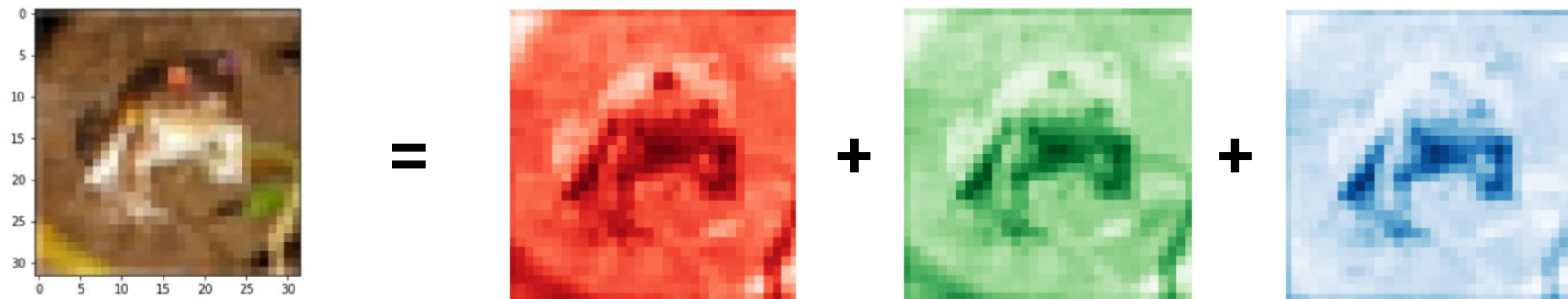
(*) Historia y resultados con diferentes algoritmos [link](#)

CIFAR-10

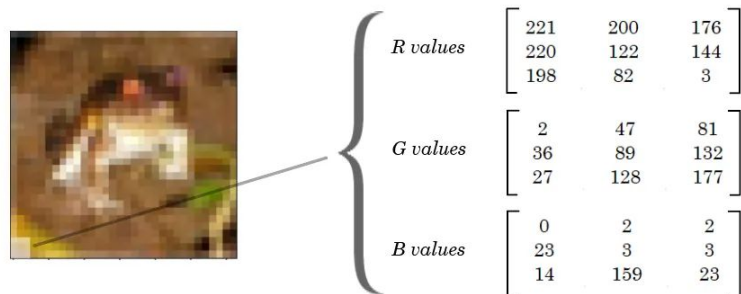
- CIFAR-10 tiene 60000 imágenes a color de 32x32 píxeles, separadas en 10 clases
- Las imágenes a color están representadas igual que antes, cada píxel es un valor entre 0 y 255, pero hay un “canal” para el rojo, el verde y el azul. O sea que por cada punto hay tres valores.



CIFAR-10



how to express a 3 x 3 square of pixels using matrices

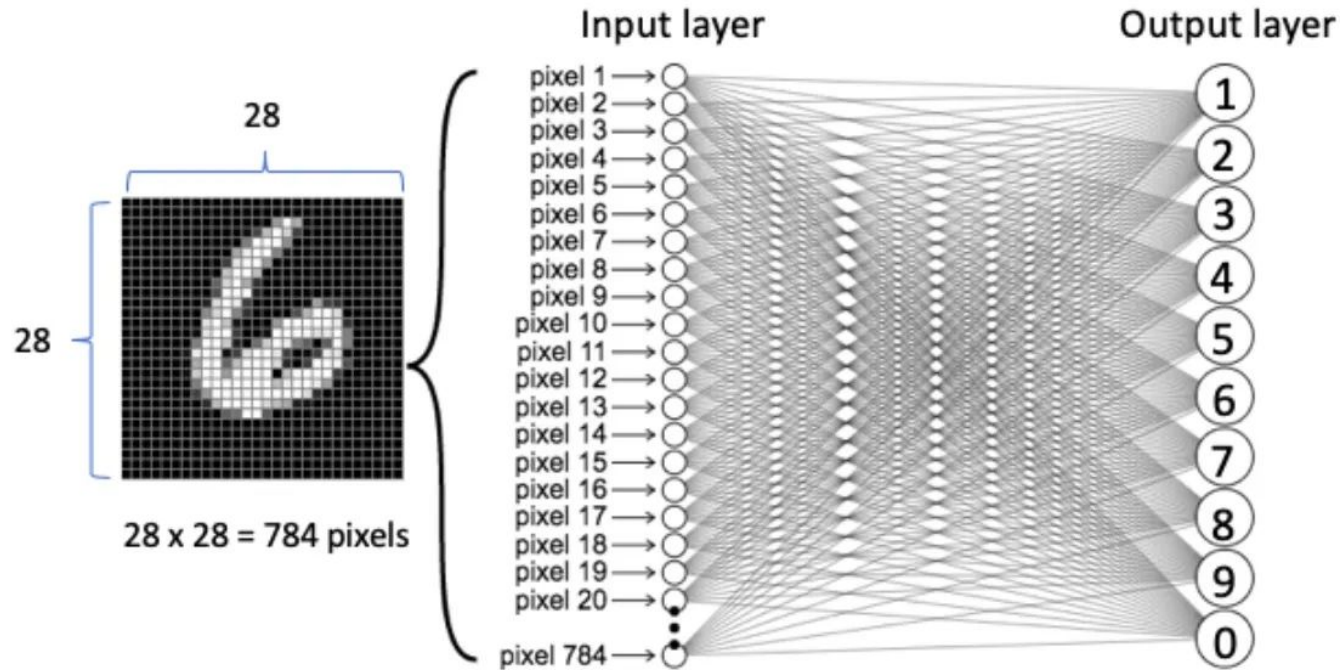


[Fuente](#)

Red *fully connected* para MNIST

- ¿Cómo era una red para MNIST?
- La salida es facil, es clasificación, una neurona para cada clase (10) y una función de salida softmax
- La matriz de la imagen se aplanan (*flatten*) y se transforma en un vector de 784 posiciones ($28 \times 28 = 784$)
- Quedaba algo así...

Red *fully connected* para MNIST



[Fuente](#)

Red *fully connected* para MNIST

- Sorprendentemente (al menos para mi) hay resultados muy buenos atacando este *data set* con algunas de las herramientas que ya conocen (datos del link anterior).
- Pensemos que pasa con una imagen más grande

Clasificador	Detalles	Pre-procesamiento	Error
Clasificador Lineal	<i>Linear Discriminant Analysis</i>	Enderezado (<i>deskewing</i>)	7.6%
K-Nearest Neighbors	K-NN con deformación no lineal (P2DHMDM)	Bordes desplazables	0.52%
Boosted Stumps (Variación de Random Forest)	Producto de <i>stumps</i> sobre Haar Cascades	Haar features	0.87%
Clasificador No Lineal	40 PCA + clasificador cuadrático	Ninguno	3.3%
Support Vector Machine	Virtual SVM, grado polinómico 9.	Enderezado	0.56%
Red Neuronal Profunda (DNN)	2 capas: 784-800-10	Ninguno.	1.6%
Red Neuronal Profunda (DNN)	2 capas: 784-800-10	Ninguno.	0.7%
Red Neuronal Profunda (DNN)	6 capas: 784-2500-2000-1500-1000-500-10	Ninguno.	0.35%

Red *fully connected* para imágenes

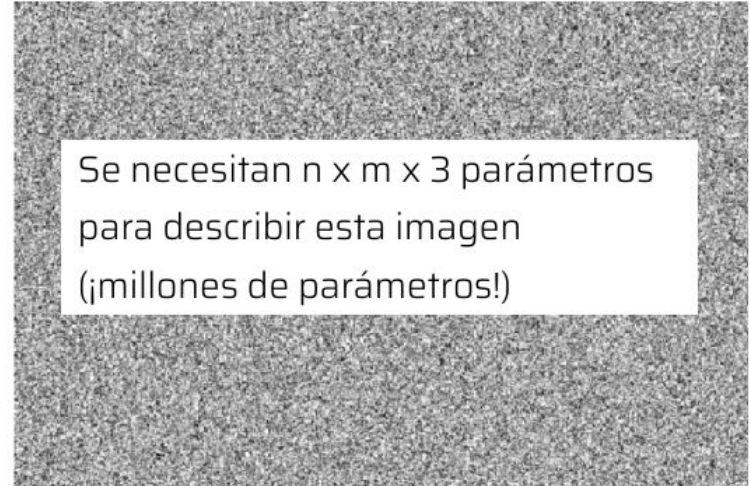


Red *fully connected* para imágenes



Se necesitan $n \times m \times 3$ parámetros
para describir esta imagen
(¡millones de parámetros!)

Red *fully connected* para imágenes



- Una red totalmente conectada para este tipo de entrada es enorme y además no tiene en cuenta que cada pixel en realidad está correlacionado con sus vecinos, no son independientes. Es un tipo de información que es útil al trabajar con imágenes y la estamos perdiendo.
- Así nacieron las **Redes Convolucionales**

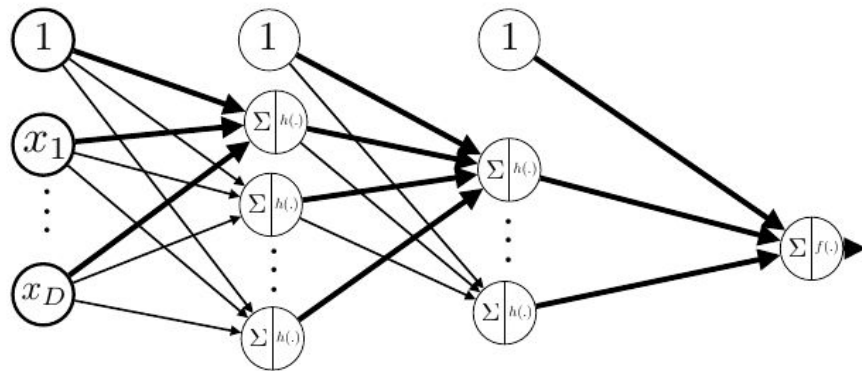
Redes Convolucionales (CNN)

- **1958 - 1968.** Hubel & Wiesel (Nobel 1981). Estudio de la corteza visual en gatos y monos.
 - a. Muchas neuronas de la corteza visual tienen un **pequeño campo receptivo local**.
 - b. Algunas neuronas reaccionan sólo a imágenes de líneas horizontales, mientras que otras reaccionan sólo a líneas con orientaciones diferentes (dos neuronas pueden tener el mismo campo receptivo pero reaccionan a orientaciones de línea diferentes).
 - c. Algunas neuronas tienen **campos receptivos más amplios** y reaccionan a **patrones más complejos**, que son combinaciones de los patrones de nivel inferior.
- **1980.** Primeros intentos de RNA basadas en este tipo de arquitectura.
- **1998,** LeCun. LeNet-5. Capas convolucionales y capas de agrupamiento.
- Años **2000** y posteriores.
 - a. rendimiento sobrehumano en algunas tareas visuales complejas.
 - b. Utilizado en otras tareas: reconocimiento de voz y procesamiento del lenguaje natural

Redes Convolucionales (CNN)

Arquitectura *Fully connected*:

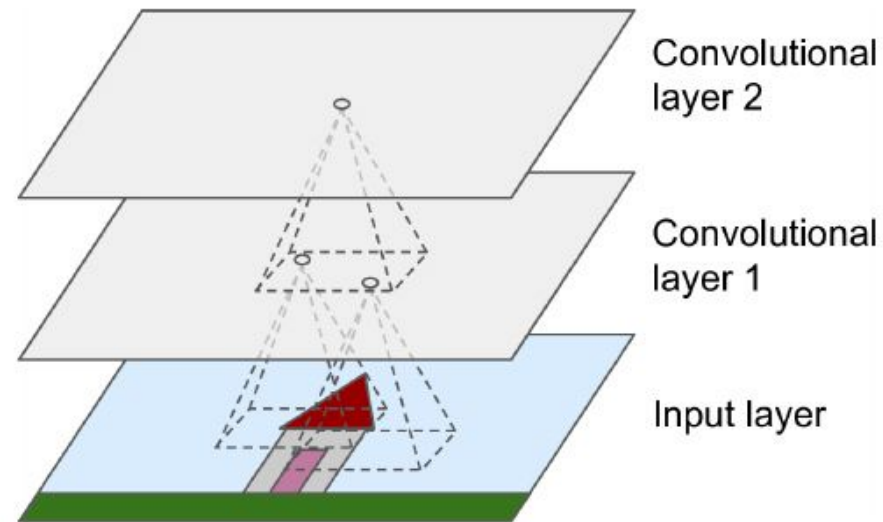
Cada unidad está conectada con todas las anteriores y las próximas



Entrada aplanada

Arquitectura convolucional:

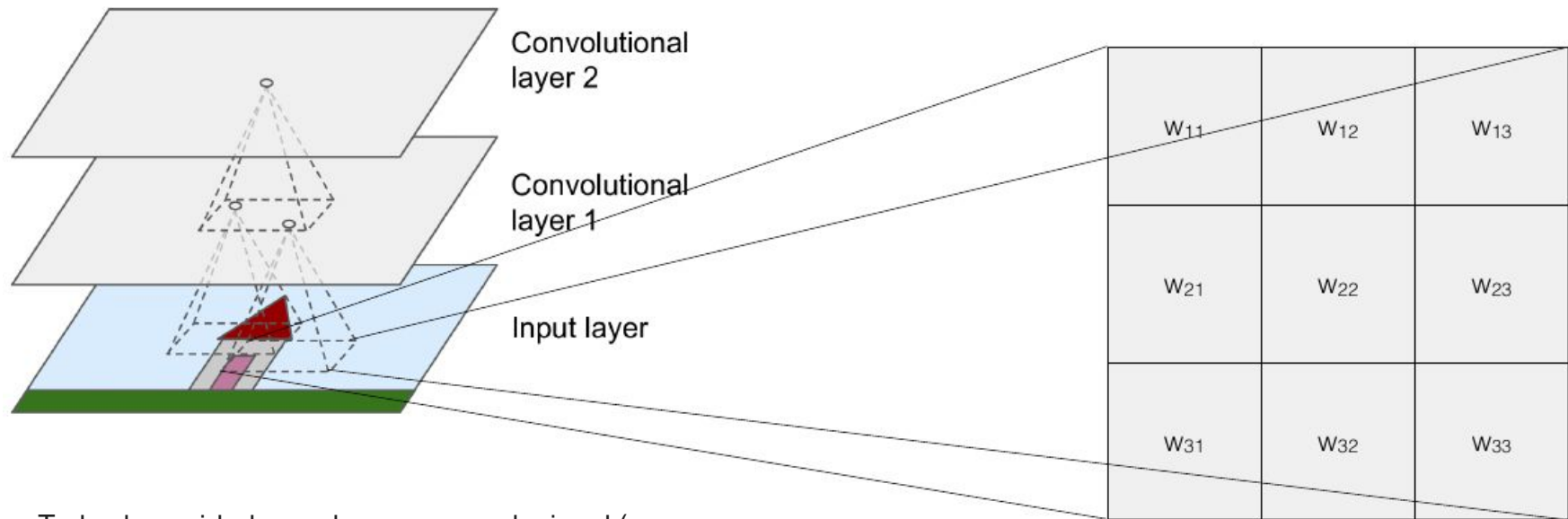
Las unidades en una capa son sensibles a un pequeño campo receptivo en la capa anterior



Se mantiene la estructura 2D

Imágenes del Géron

Redes Convolucionales (CNN)



Todas las unidades en la capa convolucional (en realidad del filtro) comparten pesos (i.e., hay relativamente un número pequeño de pesos por cada capa).

Pesos ajustables

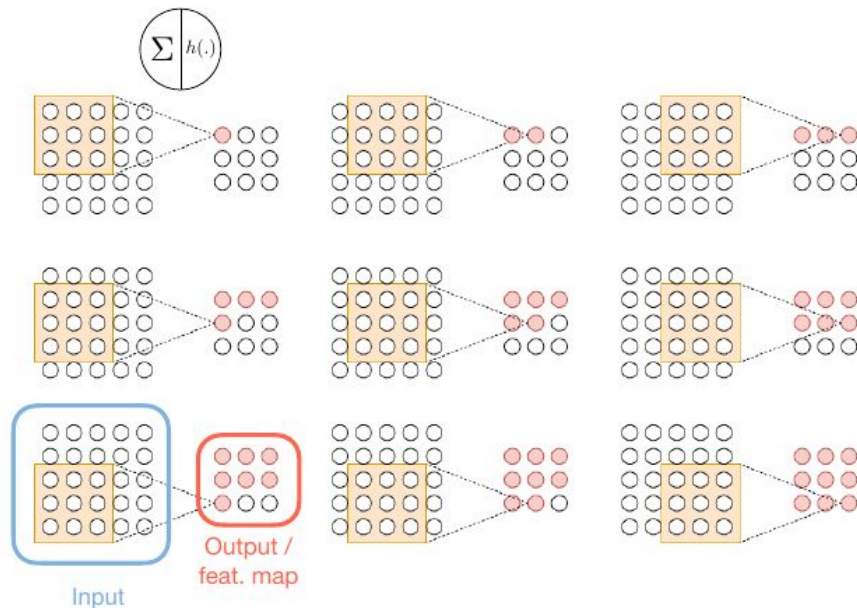
Redes Convolucionales (CNN)

Todas las unidades en la capa convolucional (en realidad del filtro) comparten pesos (i.e., hay relativamente un número pequeño de pesos por cada capa).

W_{11}	W_{12}	W_{13}
W_{21}	W_{22}	W_{23}
W_{31}	W_{32}	W_{33}

Pesos ajustables

La salida es calculada pasando el kernel a través de la imagen (o de la capa anterior) para producir un mapa de características o *Feature Map*



Convolución

1	0	1	1	0	1	1	0	1	0
0	1	0	0	1	1	0	0	1	0
1	0	1	1	0	0	0	1	1	0
1	1	0	1	1	1	1	0	1	0
1	0	1	1	0	1	0	0	0	1
1	1	0	1	0	1	0	1	0	0
0	0	0	0	1	1	0	0	1	0
0	0	0	1	1	1	1	0	1	1
0	0	1	1	0	1	0	1	1	1
0	1	1	1	0	1	1	0	1	1

tomemos una imagen

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

vamos a convoluir un pedazo

Convolución

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

*



1	-1	1
-1	1	-1
1	-1	1

=

?	?	?
?	?	?
?	?	?

elegimos un filtro
(núcleo/Kernel) que
extrae formas de
cruz

Convolución

(no es un producto de matrices)

Convolución

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

*



1	-1	1
-1	1	-1
1	-1	1

=

?		

elegimos un filtro
(núcleo/Kernel) que
extrae formas de
cruz

$$1 \times 1 + 0 \times (-1) + 1 \times 1 + \\ 0 \times (-1) + 1 \times 1 + 0 \times (-1) + \\ 1 \times 1 + 0 \times (-1) + 1 \times 1 = 5$$

Convolución

(no es un producto de matrices)

Convolución

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

*



Convolution

(no es un producto de matrices)

1	-1	1
-1	1	-1
1	-1	1

=

5		

elegimos un filtro
(núcleo/Kernel) que
extrae formas de
cruz

Convolución

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

*



1	-1	1
-1	1	-1
1	-1	1

=

5	?	

elegimos un filtro
(núcleo/Kernel) que
extrae formas de
cruz

Convolución

(no es un producto de matrices)

Convolución

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

*



1	-1	1
-1	1	-1
1	-1	1

elegimos un filtro
(núcleo/Kernel) que
extrae formas de
cruz

=

5	?	

$$\begin{aligned} &0 \times 1 + 1 \times (-1) + 1 \times 1 + \\ &1 \times (-1) + 0 \times 1 + 0 \times (-1) + \\ &0 \times 1 + 1 \times (-1) + 1 \times 1 = -1 \end{aligned}$$

Convolution

(no es un producto de matrices)

Convolución

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

*



1	-1	1
-1	1	-1
1	-1	1

=

5	-1	

elegimos un filtro
(núcleo/Kernel) que
extrae formas de
cruz

Convolución

(no es un producto de matrices)

Convolución

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

*



1	-1	1
-1	1	-1
1	-1	1

=

5	-1	-1

elegimos un filtro
(núcleo/Kernel) que
extrae formas de
cruz

Convolución

(no es un producto de matrices)

Convolución

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

*



Convolución

(no es un producto de matrices)

1	-1	1
-1	1	-1
1	-1	1

elegimos un filtro
(núcleo/Kernel) que
extrae formas de
cruz

=

5	-1	-1
-3		

Convolución

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

*



1	-1	1
-1	1	-1
1	-1	1

=

5	-1	-1
-3	3	

elegimos un filtro
(núcleo/Kernel) que
extrae formas de
cruz

Convolución

(no es un producto de matrices)

Convolución

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

*



1	-1	1
-1	1	-1
1	-1	1

=

5	-1	-1
-3	3	1

elegimos un filtro
(núcleo/Kernel) que
extrae formas de
cruz

Convolución

(no es un producto de matrices)

Convolución

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

*



1	-1	1
-1	1	-1
1	-1	1

=

5	-1	-1
-3	3	1
4		

elegimos un filtro
(núcleo/Kernel) que
extrae formas de
cruz

Convolución

(no es un producto de matrices)

Convolución

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

*



1	-1	1
-1	1	-1
1	-1	1

=

5	-1	-1
-3	3	1
4	-2	

elegimos un filtro
(núcleo/Kernel) que
extrae formas de
cruz

Convolución

(no es un producto de matrices)

Convolución

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

*



1	-1	1
-1	1	-1
1	-1	1

=

5	-1	-1
-3	3	1
4	-2	0

elegimos un filtro
(núcleo/Kernel) que
extrae formas de
cruz

Convolución

(no es un producto de matrices)

Convolución

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

*



Convolución

1	-1	1
-1	1	-1
1	-1	1

elegimos un filtro
(núcleo/Kernel) que
extrae formas de
cruz

=

5	-1	-1
-3	3	1
4	-2	0

Reconoce una
cruz

(no es un producto de matrices)

Convolución

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

*



1	-1	1
-1	1	-1
1	-1	1

=

5	-1	-1
-3	3	1
4	-2	0

elegimos un filtro
(núcleo/Kernel) que
extrae formas de
cruz







Convolución

(no es un producto de matrices)

También si es
similar a una cruz

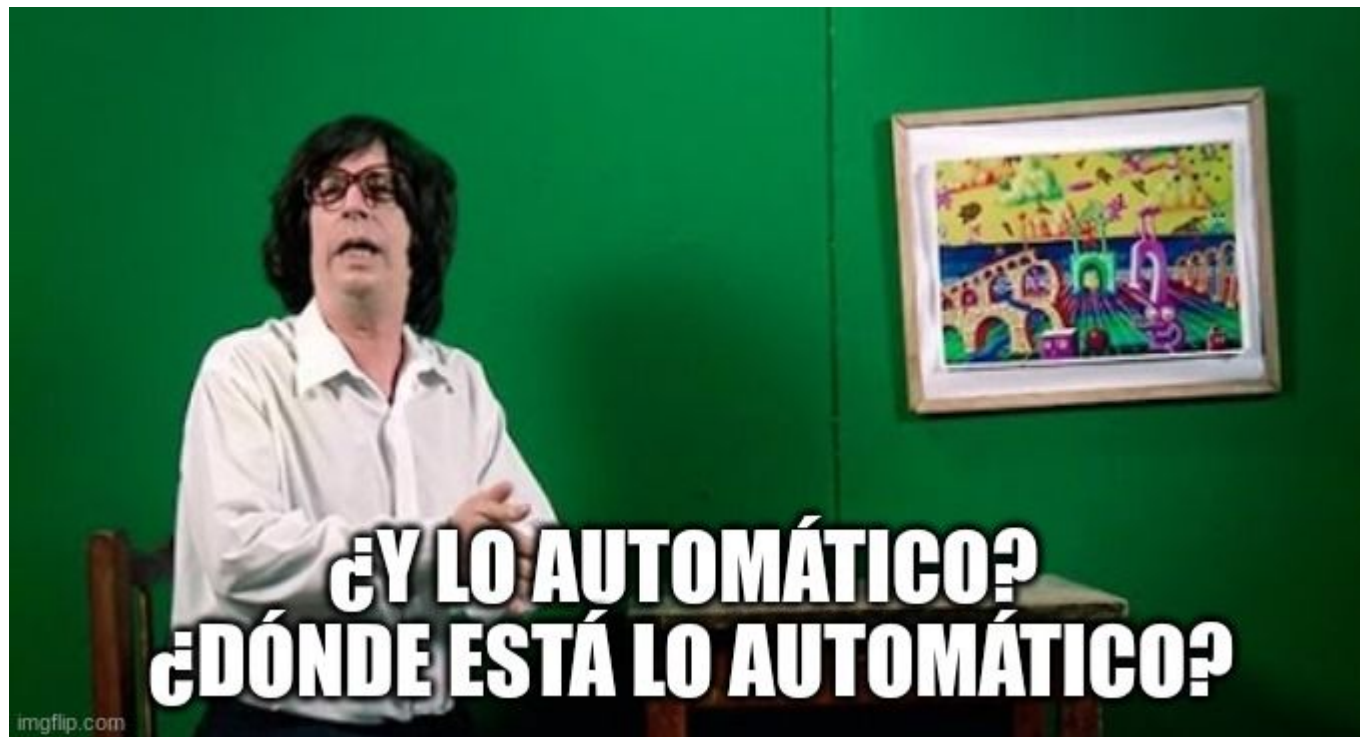
Convolución

- Resumiendo: un filtro busca un patrón particular en la imagen y también puede detectar cosas similares
- Tiene invarianza traslacional, lo puede encontrar en cualquier parte
- Ese fue un filtro específico para buscar cruces, hay muchos para hacer diferentes cosas.
- Muy lindo pero...

Operación	Núcleo	Imagen resultante
Identidad	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Detección de bordes	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Enfocar	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Desenfoque de cuadro (normalizado)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

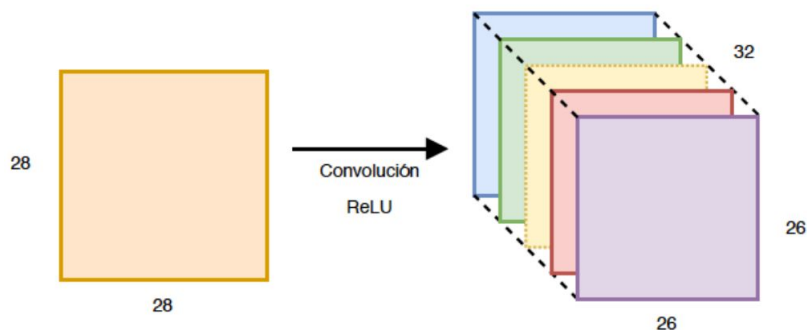
[https://es.wikipedia.org/wiki/Núcleo_\(procesamiento_digital_de_imágenes\)](https://es.wikipedia.org/wiki/Núcleo_(procesamiento_digital_de_imágenes))

Convolución

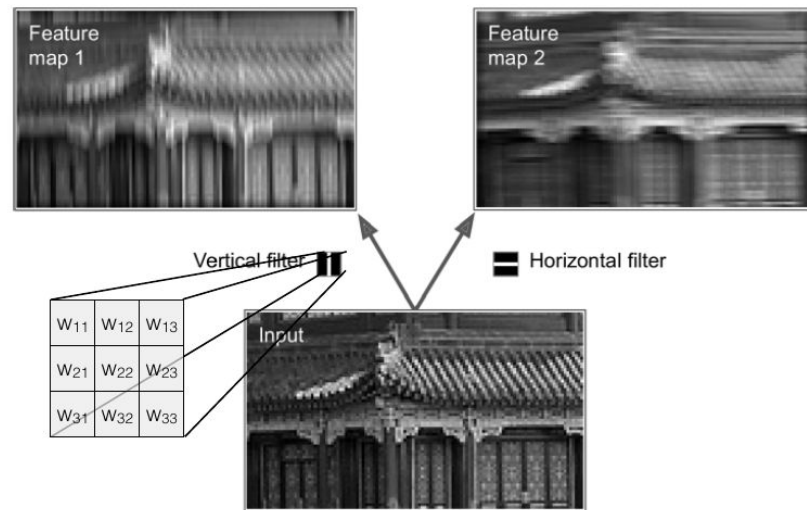


Convolución

- El chiste es que **no le vamos a dar los valores que van en los filtros**
- Le vamos a definir cosas como el tamaño o la cantidad, pero los valores que van dentro de cada uno los va a aprender la red

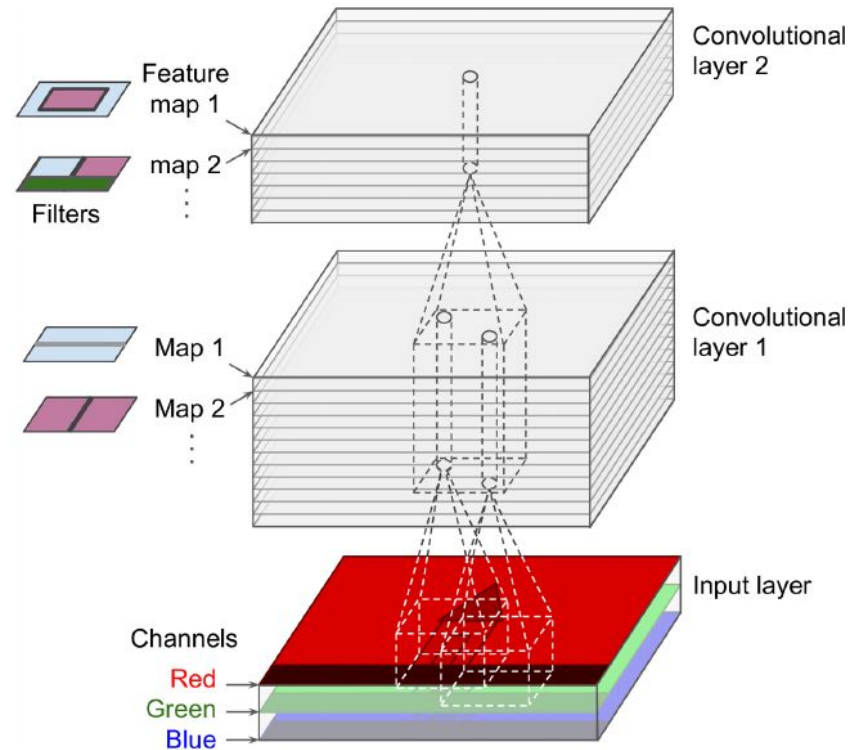
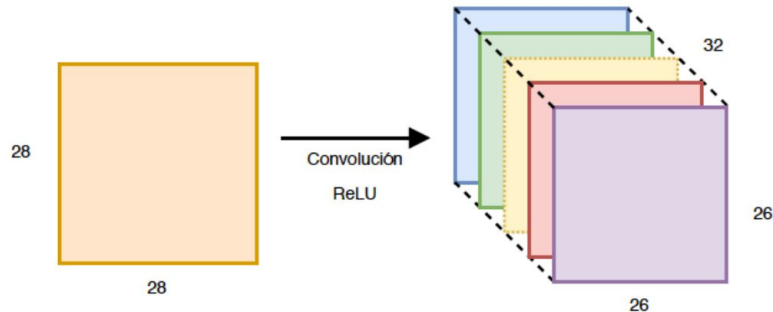


Por ejemplo acá se usó un kernel de 3x3 y se le dijo que genere 32 filtros



Redes Convolucionales (CNN)

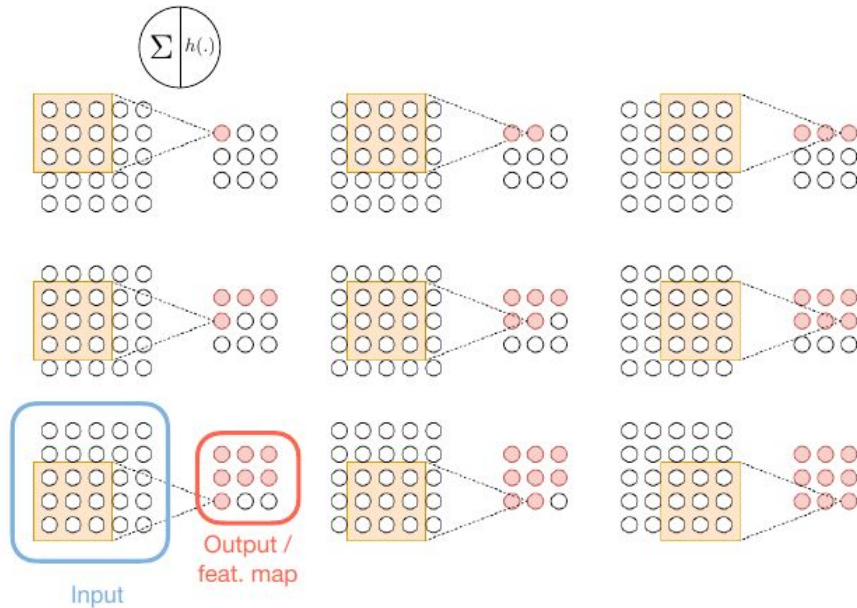
- Se busca que los diferentes filtros en las diferentes capas se combinen para alcanzar características de más alto nivel.
- En las primeras capas se pueden detectar patrones simples como líneas rectas o inclinadas, pero a medida que se profundiza se combinan y detecten cosas más complejas.



Redes Convolucionales (CNN)

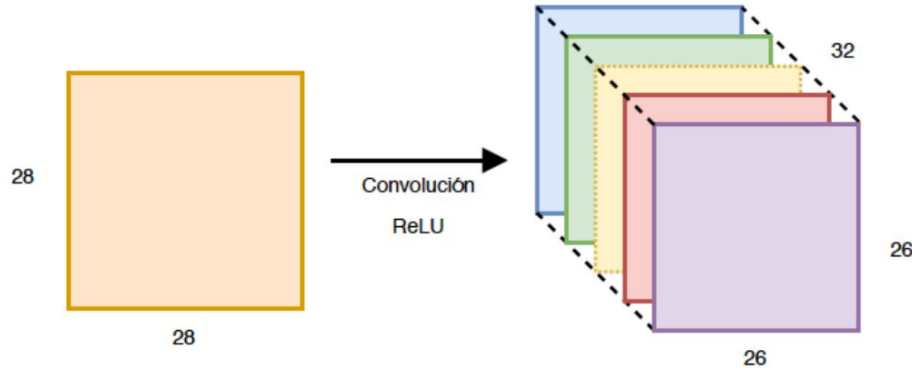
Recap

Redes Convolucionales (CNN)



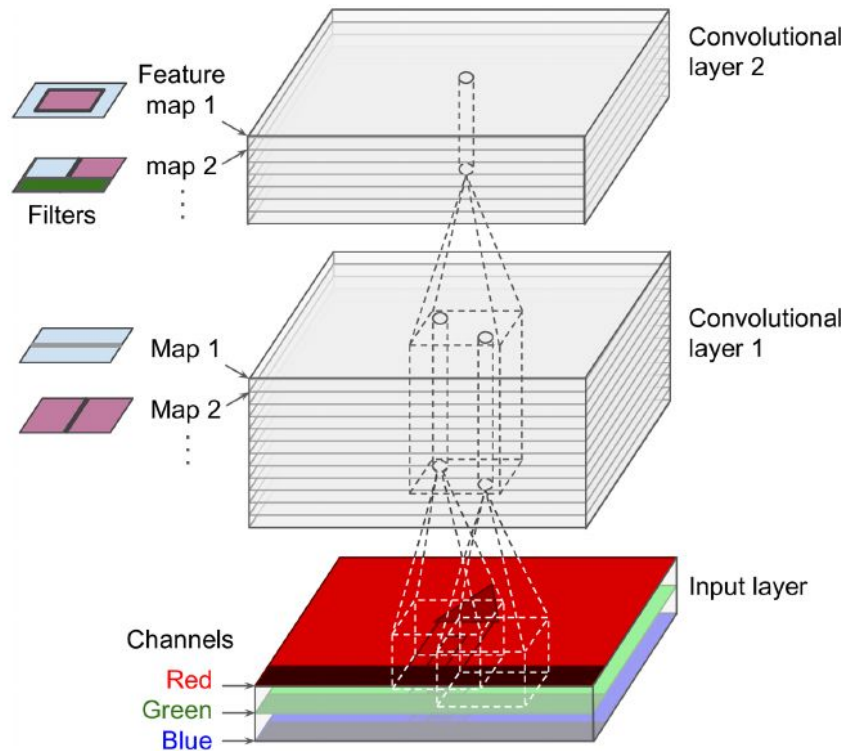
- Se pasa un **kernel** por la imagen (o la capa anterior).
- Los pesos que tenga esa matriz definen un **filtro**.
- El resultado de aplicar esa convolución sobre toda la capa anterior generan un **mapa de características**.
- El tamaño del kernel lo definimos nosotros y es el mismo para toda la capa

Redes Convolucionales (CNN)



- Cada uno de esos mapas de características son el resultado de un filtro
- Todos esos mapas de características juntos son una **capa**
- La cantidad de filtros de la capa (ergo, de mapas de características) es otro de los hiperparámetros definidos por nosotros

Redes Convolucionales (CNN)



- La convolución que se realiza sobre la capa anterior se hace sobre toda la *profundidad*
 - Si es sobre la entrada se hace sobre todos los canales
 - Si es sobre una capa convolucional sobre todos los mapas de características

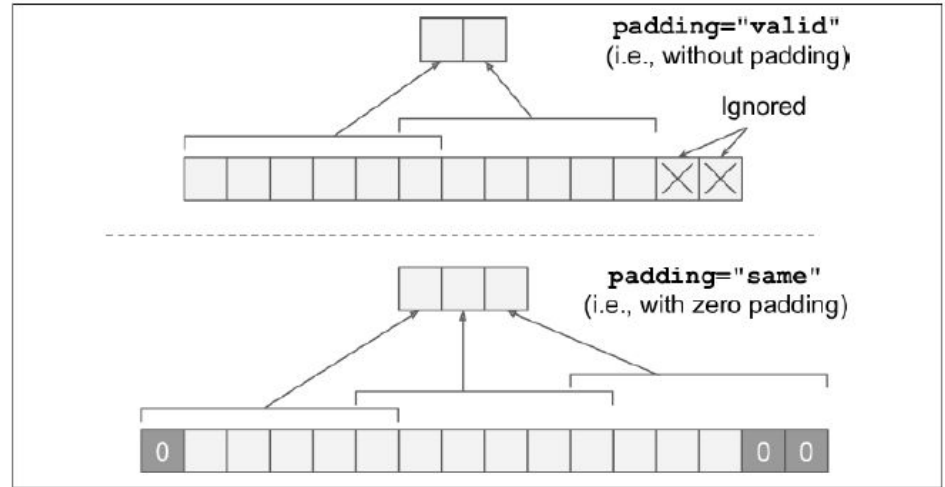
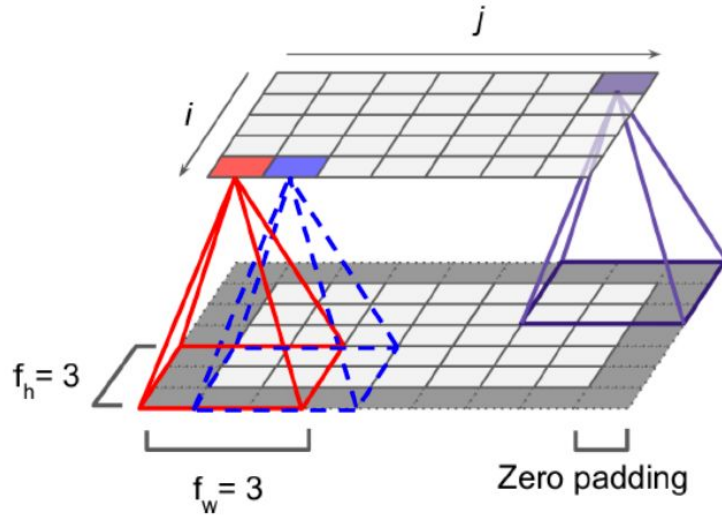
Piensen en todo lo que “ve” una neurona en la capa 2 vs. una en la capa 1

Redes Convolucionales (CNN)

Fin del recap

Padding

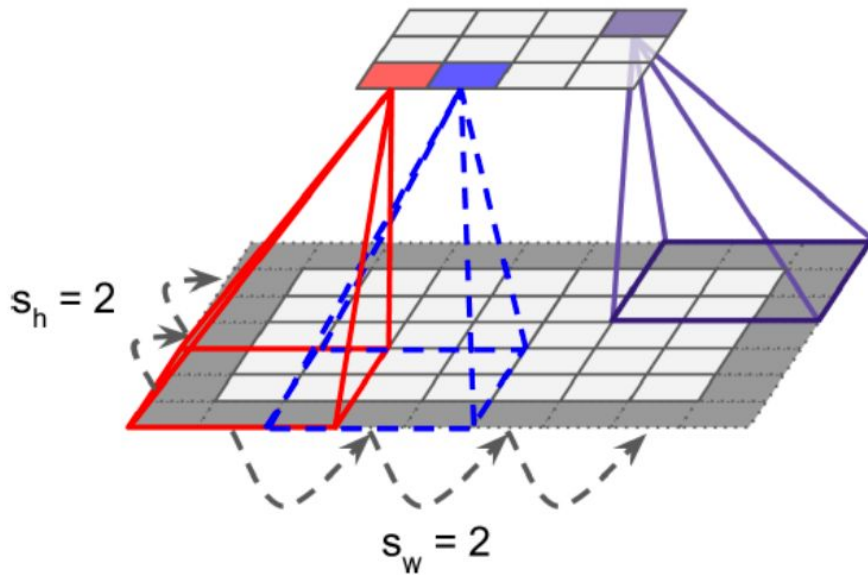
- Si se quiere mantener la dimensión de la entrada se puede rellenar (*padding*)
- Puede ser útil para no perder detalles en los bordes
- *Valid* y *same*



Imágenes del Géron

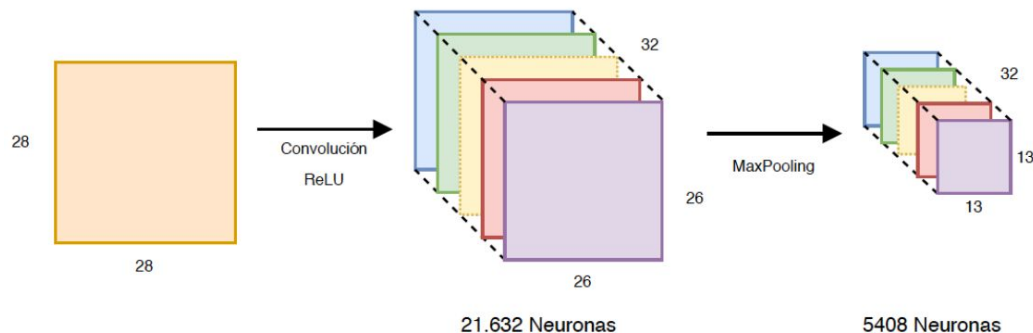
Striding

- Otra forma de controlar la dimensión es controlar cuanto queremos que se mueva el kernel por la imagen con el parámetro *stride*



Pooling

- Las CNNs son bastante intensivas en el uso de la memoria, una forma de mitigar esto es utilizar capas de reducción (*pooling layers*) que ayudan a reducir el tamaño de los mapas de características.
- También ayudan a reducir el overfitting y aumentan el nivel de invarianza traslacional (*) de la red
- *Max pooling* y *average pooling*



(*) Permite reconocer el mismo objeto en un lugar diferente de la imagen

Pooling (usando el ejemplo anterior)

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

*



Convolución

(no es un producto de matrices)

1	-1	1
-1	1	-1
1	-1	1

elegimos un filtro
(núcleo/Kernel) que
extrae formas de
cruz

=

5	-1	-1
-3	3	1
4	-2	0

Pooling (usando el ejemplo anterior)

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

*



Convolución

(no es un producto de matrices)

1	-1	1
-1	1	-1
1	-1	1

elegimos un filtro
(núcleo/Kernel) que
extrae formas de
cruz

=

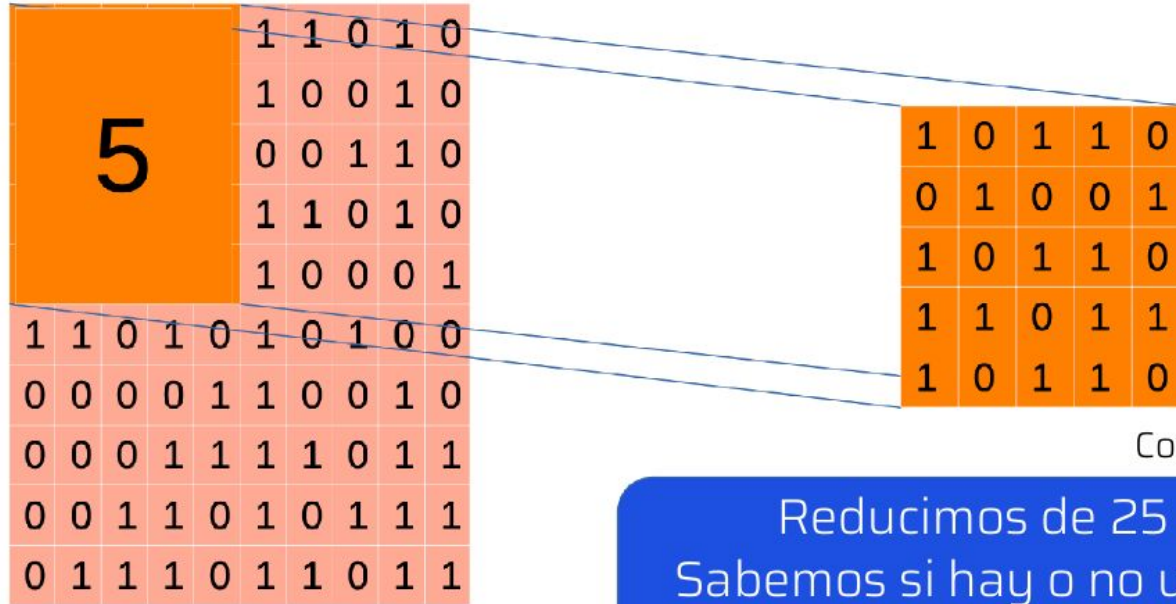
5	-1	-1
-3	3	1
4	-2	0



Si tomamos el valor
máximo con MaxPool
estamos quedándonos
con la cruz

5

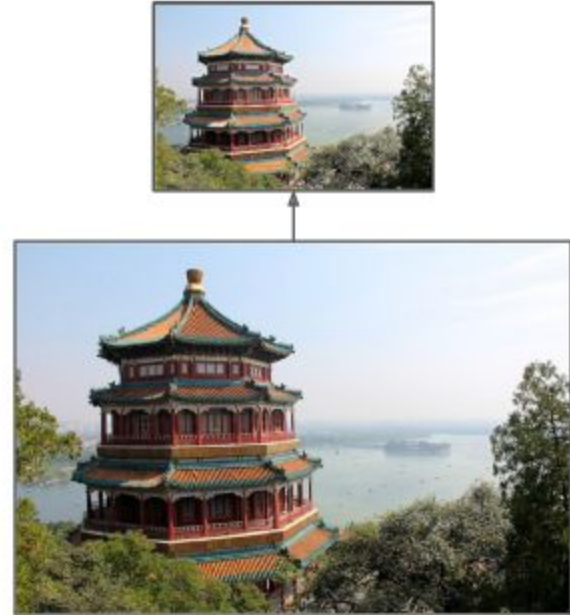
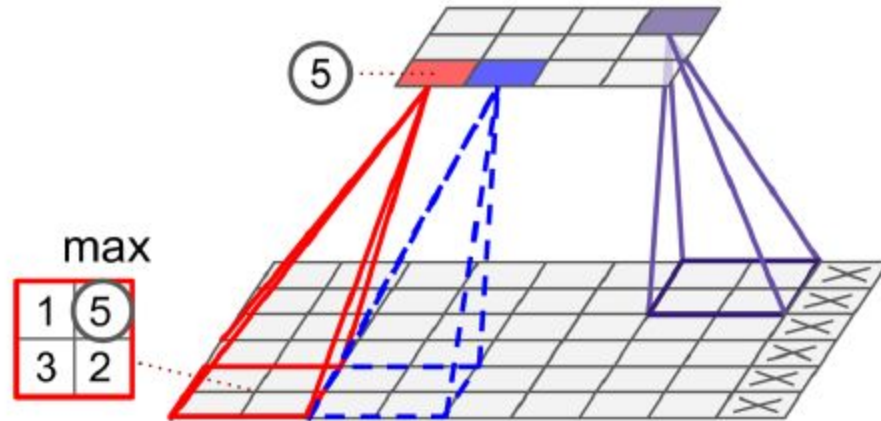
Pooling (usando el ejemplo anterior)



Computación paralela!

Reducimos de 25 pixeles a 1
Sabemos si hay o no una cruz en esa
región

Pooling

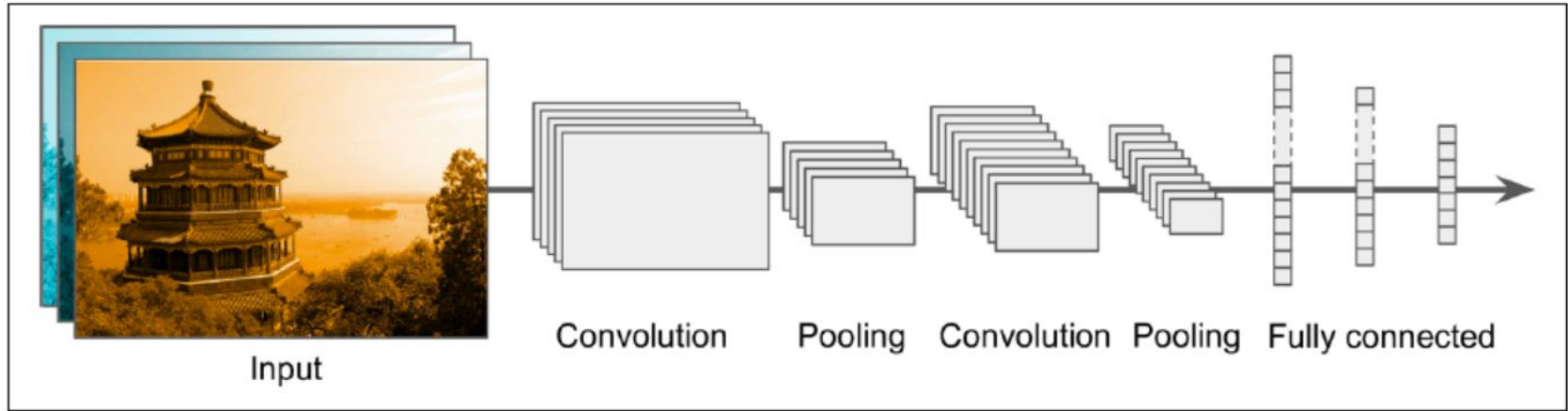


Arquitectura típica

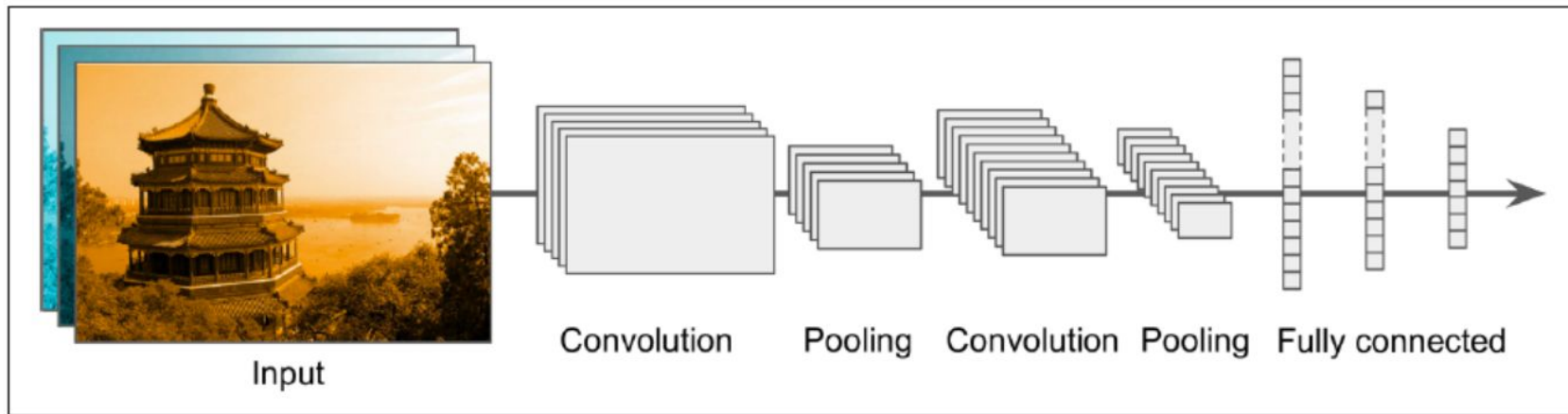
- La arquitectura va a ser, entonces, una combinación de capas convolucionales con capas de *pooling*.
- ¿Cuántas? ¿de qué características?, depende...(*).
- El detalle que nos falta es el final, si nos creyeron que a medida que la red se hace más profunda extrae características más generales también nos pueden creer que podemos usar una red totalmente conectada para trabajar con estas características, ¿**NOSIERTO?**.
- Entonces, al final de todo se agrega una red de las tradicionales para que trabaje con estas características y una salida acorde al problema con el que estamos trabajando (en el caso de MNIST o CIFAR-10 sería una capa de clasificación).

(*) Si, depende, hay algunas reglas a ojo pero hay que probar

Arquitectura típica



Arquitectura típica



- A medida que se avanza aumenta el número de mapas de características (*feature maps*)
- Disminuye las dimensiones de las imágenes (sobre todo por el pooling)
- la información espacial se va “transformando” en información sobre las características de las imágenes (se vuelve más *semántica*). Por ahora confien, la semana que viene le vamos a mostrar pruebas
- después de un cierto número de capas la información está lista para ser transmitida a una red neuronal totalmente conectada

Notebook

Vamos al notebook!

Notebook_Semana_7_CNN.ipynb