Taller Keras

Redes convolucionales para procesamiento de imágenes

Mariano Rivera Meraz
Alan G. Reyes Figueroa

Convolución

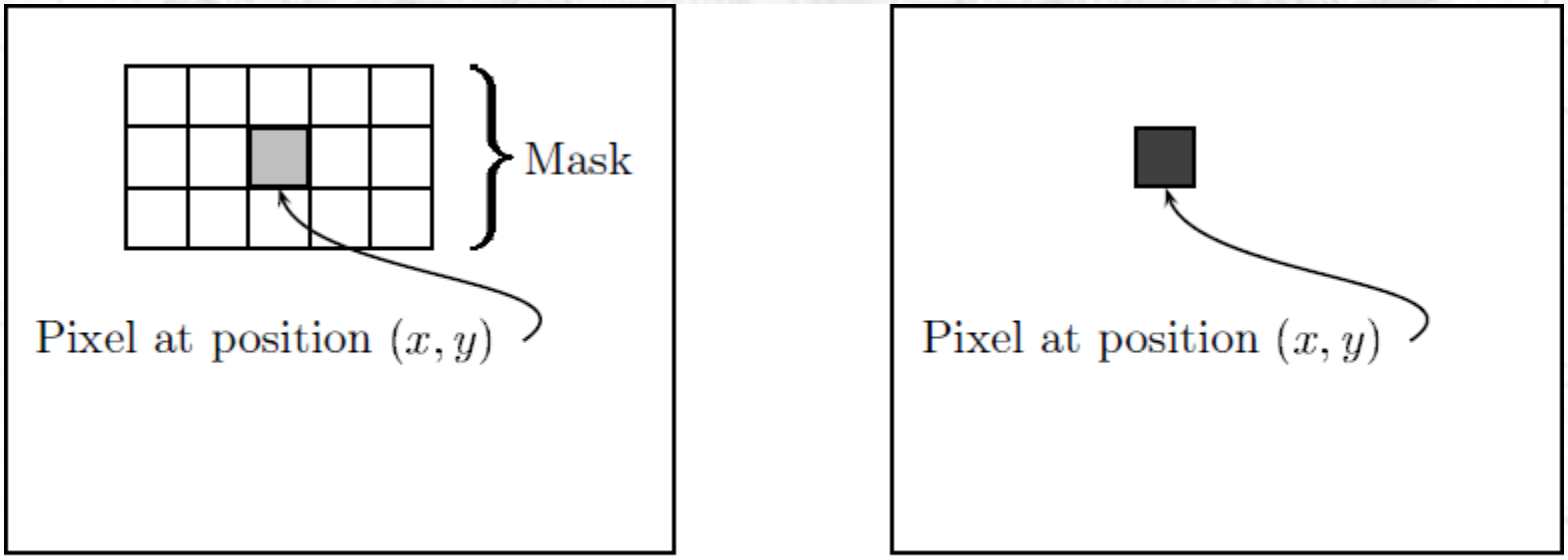
Procesamiento de imágenes

- El **procesamiento espacial** consiste de estimar el valor de un píxel en función de sus vecinos.

- La idea: mover una **máscara**: un rectángulo (con lados de tamaño impar) u otra forma. Los valores de los píxeles se calculan usando los valores de los píxeles en la máscara.

- La combinación de una máscara y una función se llama un **filtro**.

Convolución



Original image

Image after filtering

Convolución

$m(-1,-2)$	$m(-1,-1)$	$m(-1,0)$	$m(-1,1)$	$m(-1,2)$
$m(0,-2)$	$m(0,-1)$	$m(0,0)$	$m(0,1)$	$m(0,2)$
$m(1,-2)$	$m(1,-1)$	$m(1,0)$	$m(1,1)$	$m(1,2)$

and that corresponding pixel values are

$p(i-1,j-2)$	$p(i-1,j-1)$	$p(i-1,j)$	$p(i-1,j+1)$	$p(i-1,j+2)$
$p(i,j-2)$	$p(i,j-1)$	$p(i,j)$	$p(i,j+1)$	$p(i,j+2)$
$p(i+1,j-2)$	$p(i+1,j-1)$	$p(i+1,j)$	$p(i+1,j+1)$	$p(i+1,j+2)$

- A la máscara se asignan pesos $m_{ij} = m(i,j)$.
- Los píxeles de la imagens tienen valores $p_{kl} = p(k,l)$.
- El nuevo valor p_{ij} se calcula como una función de los m_{kl} y los p_{kl}.

Filtros y convolución

- Cuando esta función de de la forma.

$$\hat{p}(i,j) = \sum_{s=-1}^{1} \sum_{t=-2}^{2} m(s,t)\, p(i+s, j+t)$$

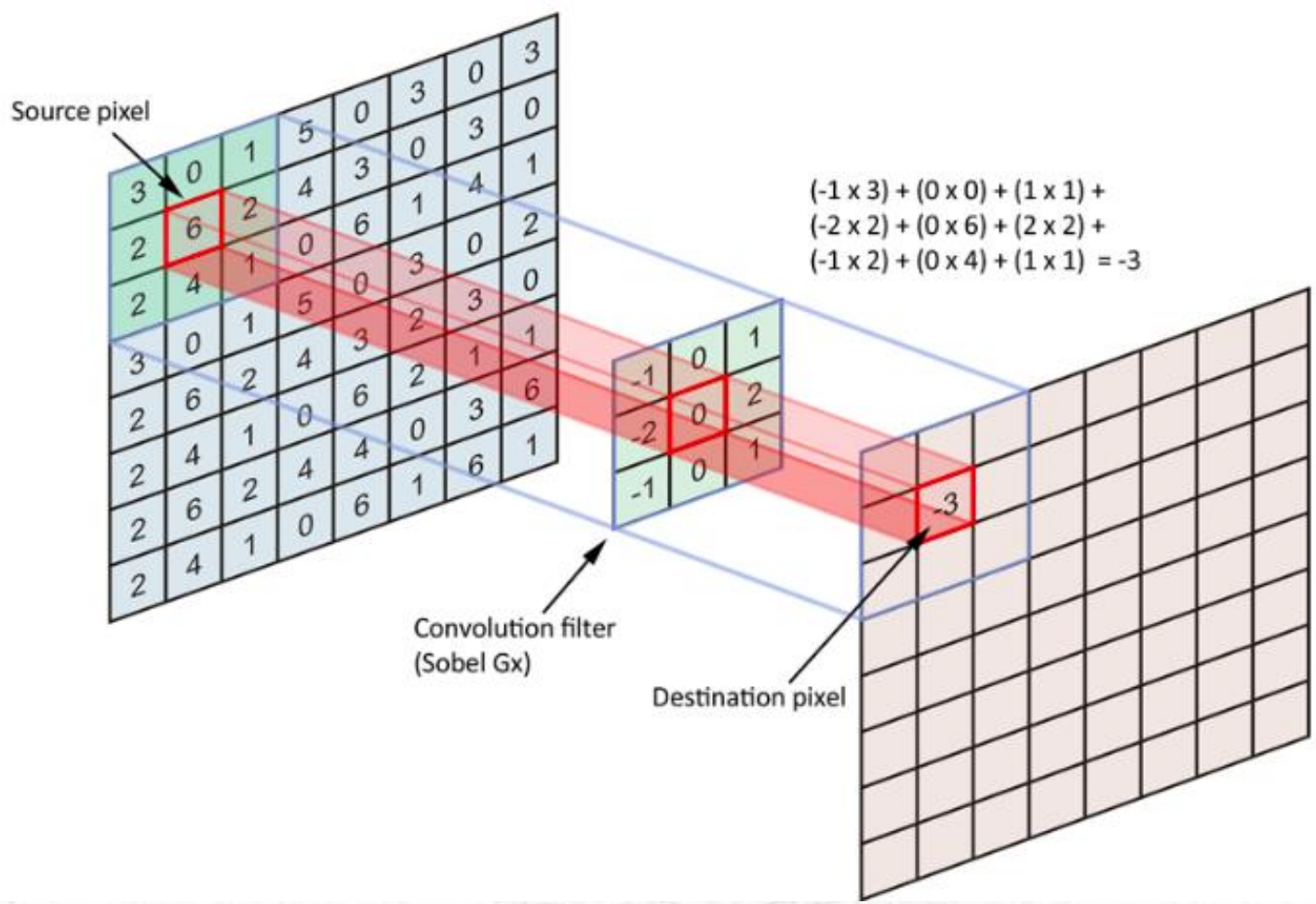
el filtro se llama **filtro lineal.** **(correlación)**

- Junto con este filtro espacial, tenemos la **convolución discreta**:

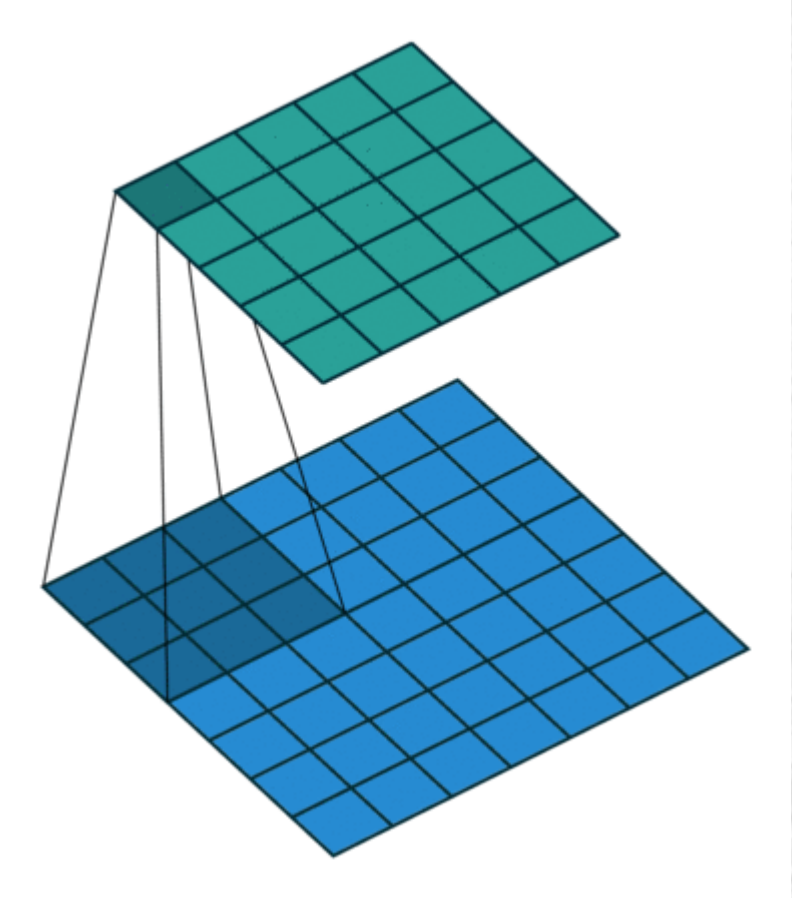
$$\hat{p}(i,j) = \sum_{s=-1}^{1} \sum_{t=-2}^{2} m(-s,-t)\, p(i+s, j+t)$$

$$= \sum_{s=-1}^{1} \sum_{t=-2}^{2} m(s,t)\, p(i-s, j-t)$$

Filtros y convolución



Filtros y convolución



Ejemplos

$$H = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

Filtro de medias.

Imagen original

Imagen filtrada

Ejemplos

$$H = \frac{1}{k^2}$$

1	1	…	1
1	1	…	1
…	…		…
1	1	…	1

$k \times k$

Filtro de medias.

Imagen original

Imagen filtrada (25 x 25)

Ejemplos

Filtros de gradiente.

$H_x =$

1	0	-1
1	0	-1
1	0	-1

$H_y =$

1	1	1
0	0	0
-1	-1	-1

Imagen original

Imagen filtrada H_x

Imagen filtrada H_y

Ejemplos

$$H_x = \frac{1}{4}$$

0	-1	0
-1	8	-1
0	-1	0

Filtro de sharpening.

Imagen original

Imagen filtrada

Ejemplos

$H =$

0	-1	0
-1	4	-1
0	-1	0

Filtro laplaciano.
(detección de bordes)

Imagen original

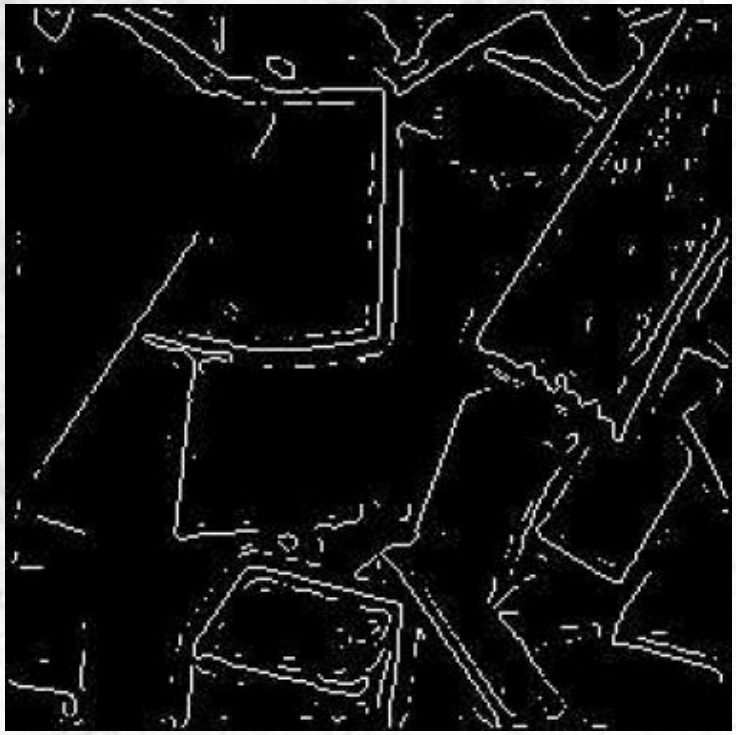
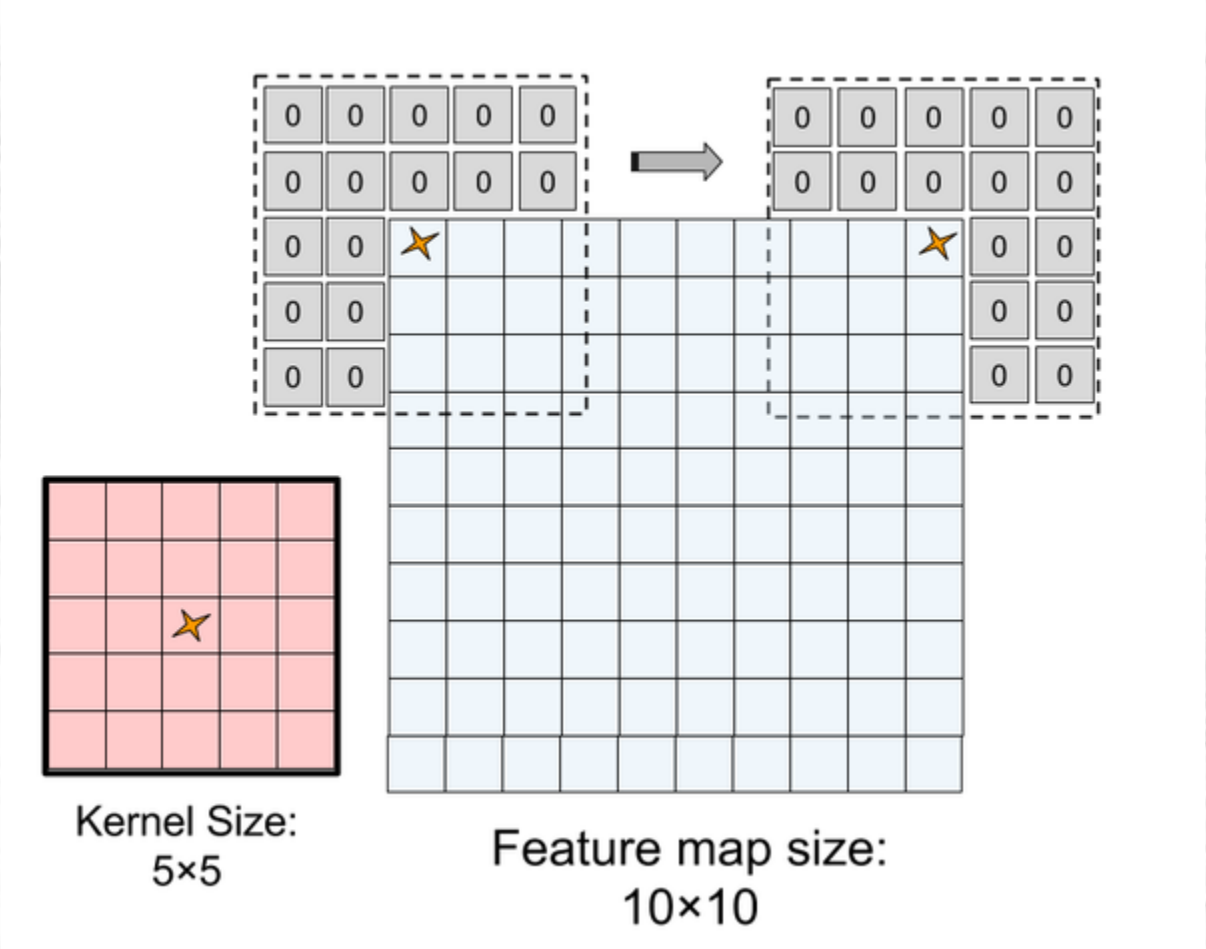


Imagen filtrada

Padding


0 0 0 0 0
0 0 0 0 0
0 0
0 0
0 0
0 0 0 0 0
0 0 0 0 0
0 0
0 0
0 0
Kernel Size:
5×5
Feature map size:
10×10

Padding

En Keras, hay dos valores para el parámetro **padding**:

H filtro de tamaño $(2k+1) \times (2k+1)$.

I imagen de tamaño $m \times n$.

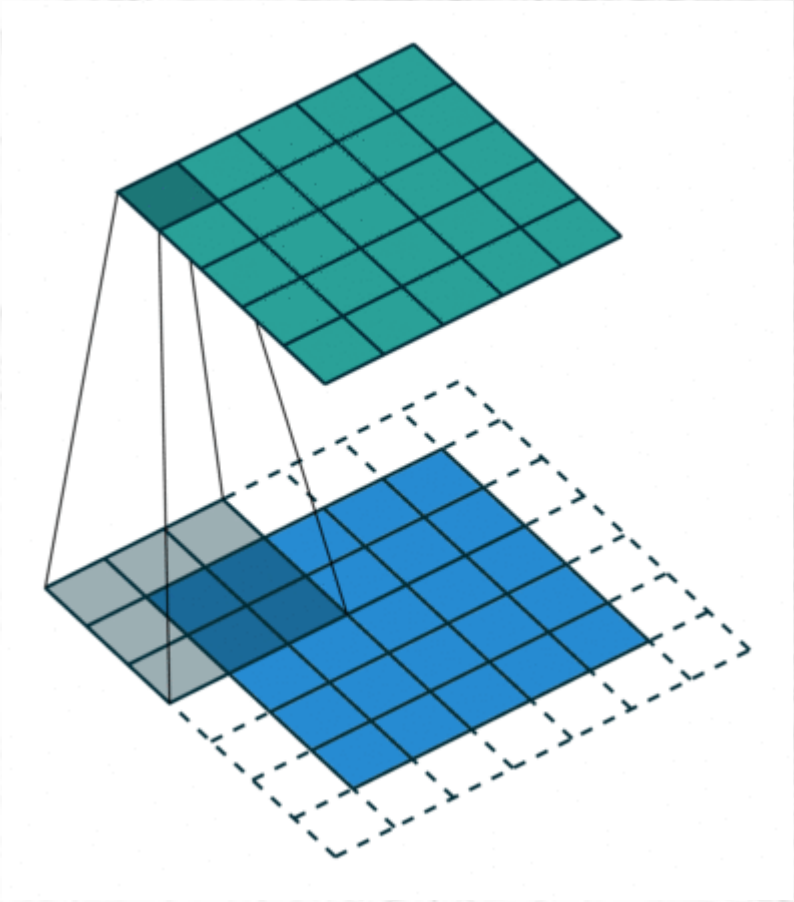
- padding = 'valid'. Sólo recorre el filtro en las posiciones válidas, donde no se sale del dominio de la imagen.

  La imagen de salida es más reducida $(m-2k) \times (n-2k)$.

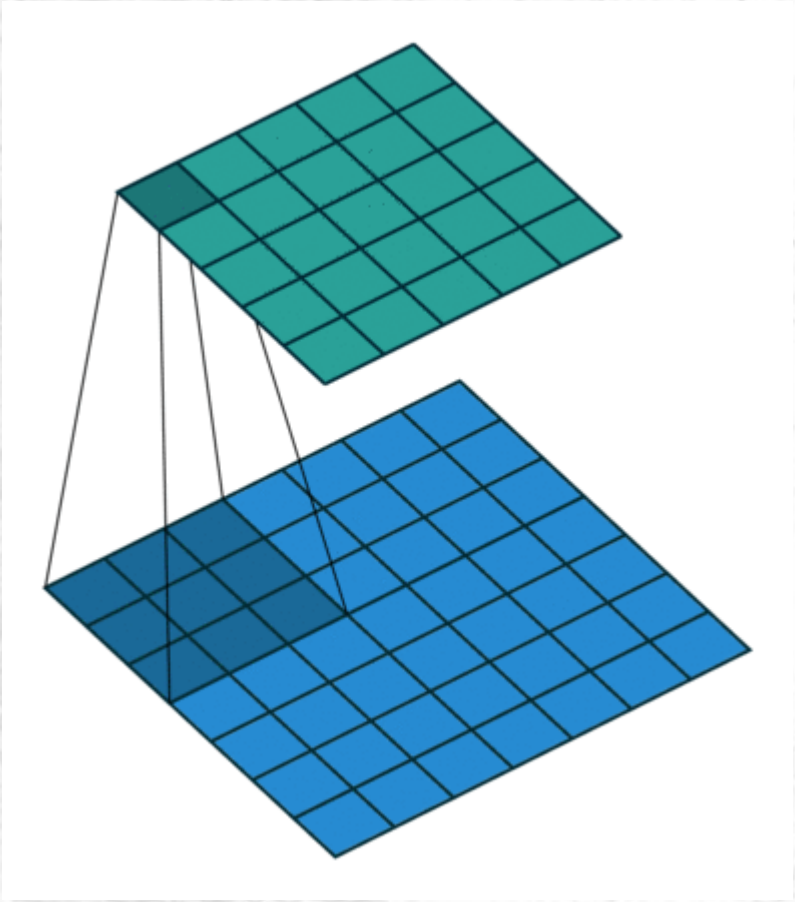
- padding = 'same'. Crea una región que extiende la imagen, para recorrer la máscara sobre todos los píxeles.

  La imagen de salida es del mismo tamaño $m \times n$.

Padding



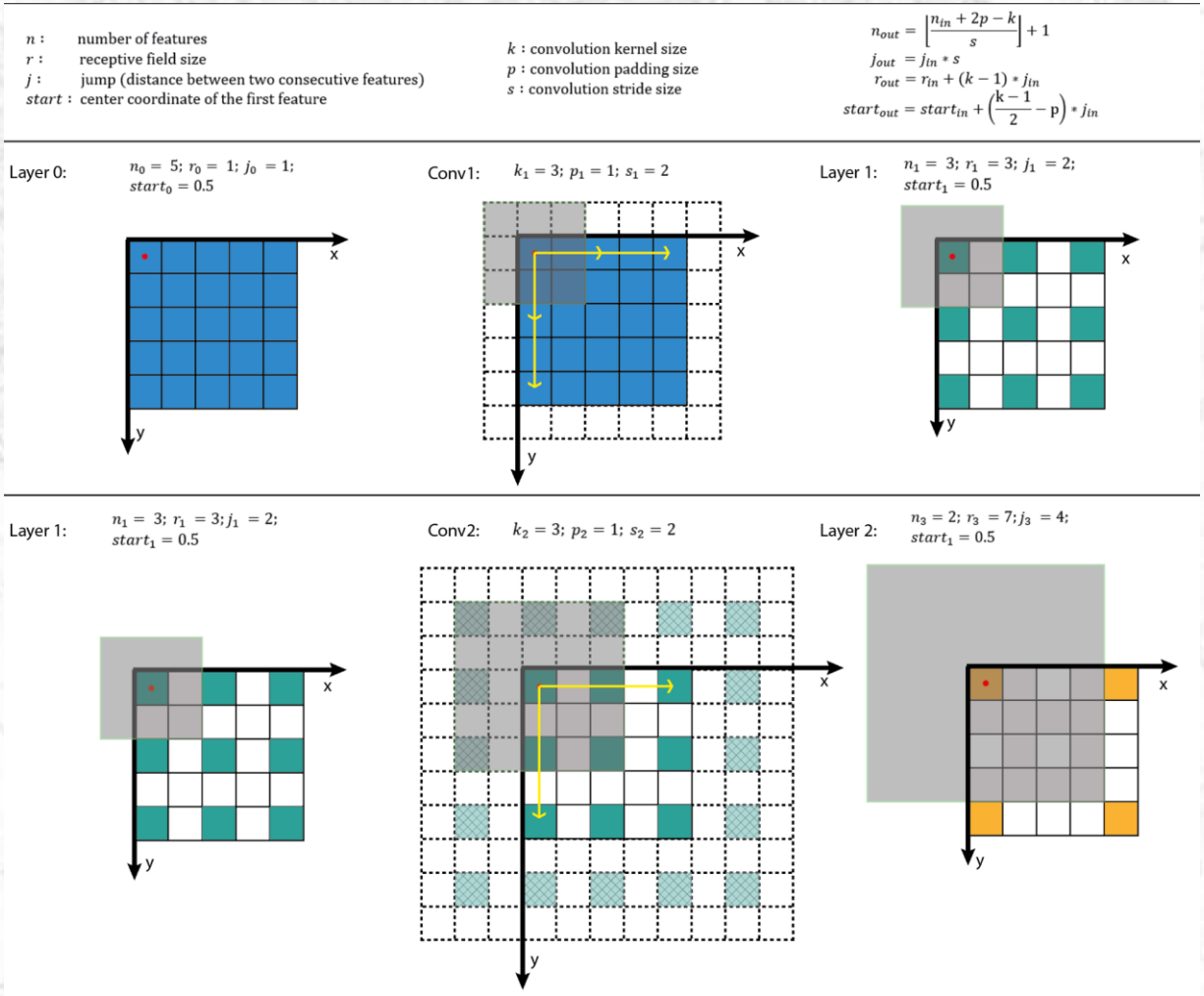
padding = ‘same’



padding = ‘valid’

Stride

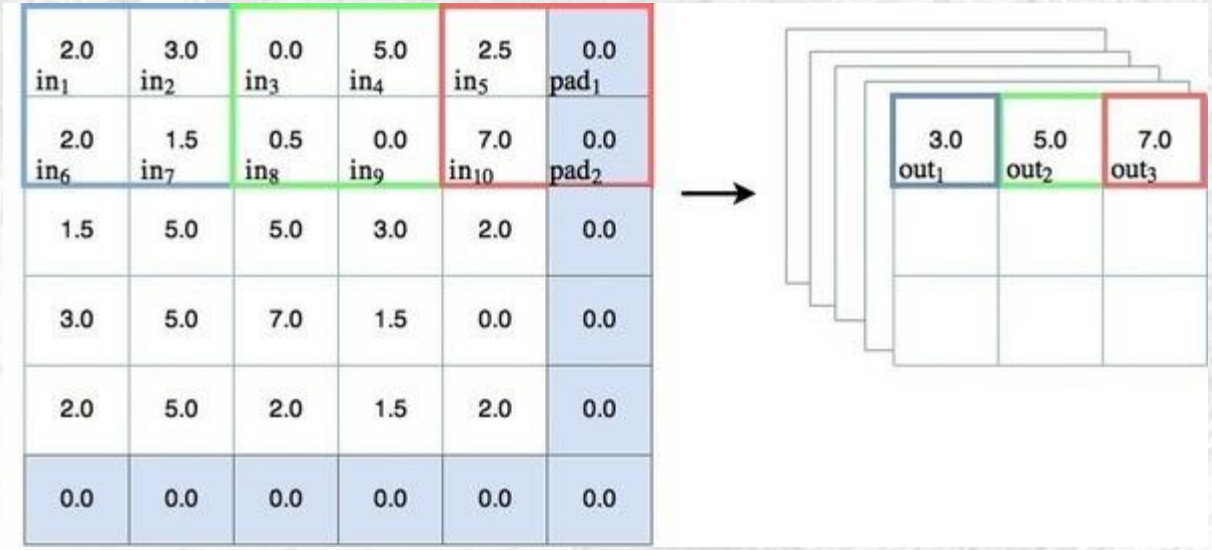
Tamaño de salto en la convolución. En Keras, el default es (1,1).



Pooling (sub-muestreo)

Las capas de Pooling, hacen un muestreo sistemático de los píxeles en la imagen.

En Keras, usa el parámetro Pool_size = (2,2).

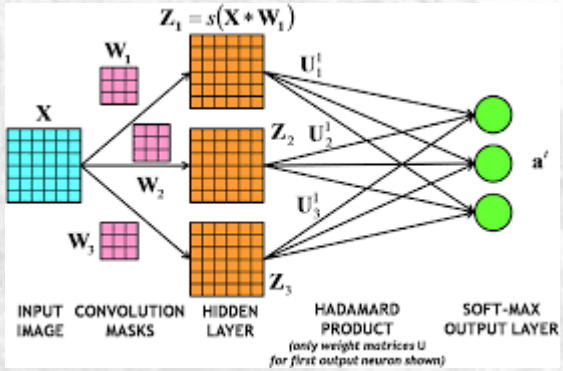


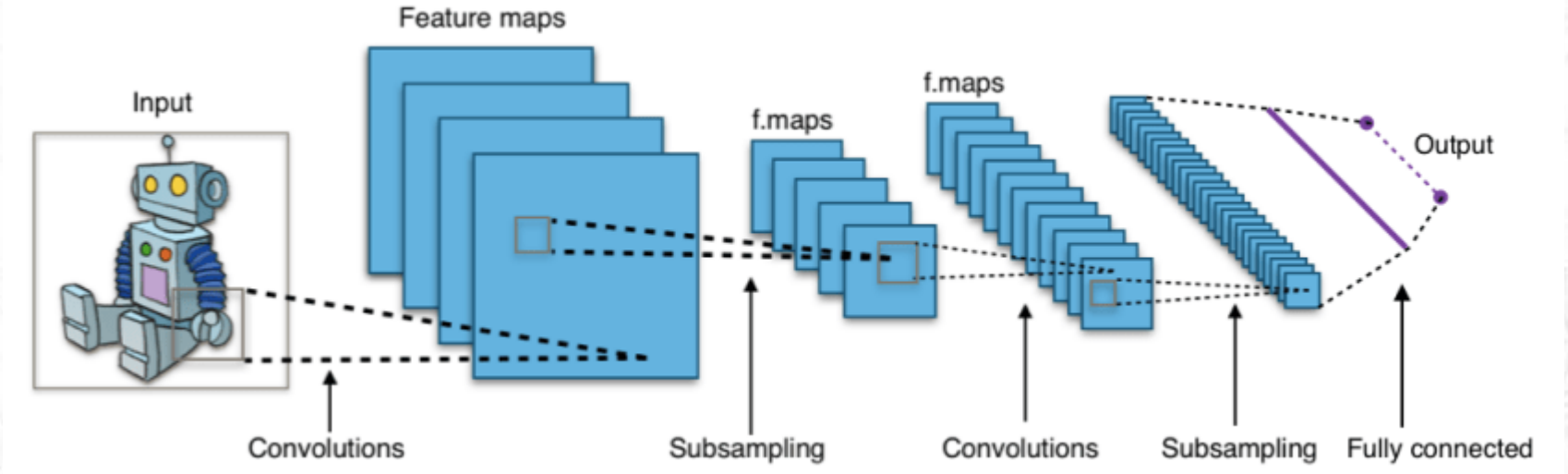
La imagen de salida se reduce en un factor de Pool_size.

También parámetro stride. En Keras, stride = pool_size.

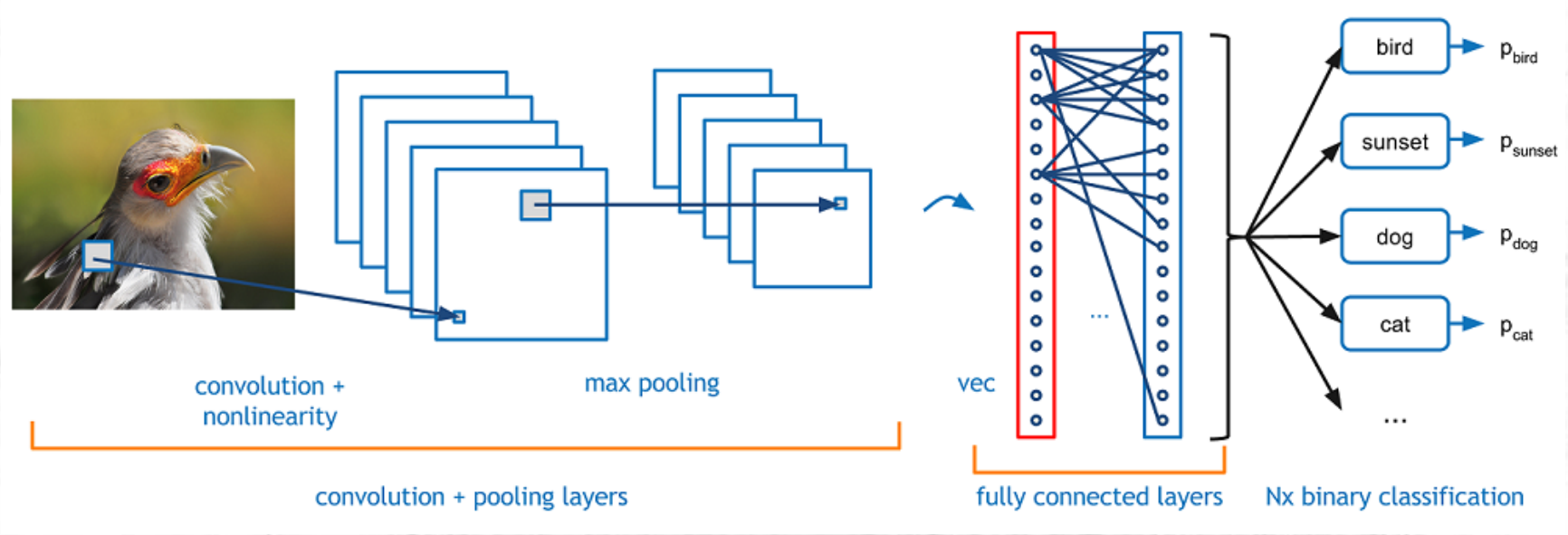
Redes neuronales convolucionales

- Neurona = filtro.
- Pesos = pesos de cada filtro.





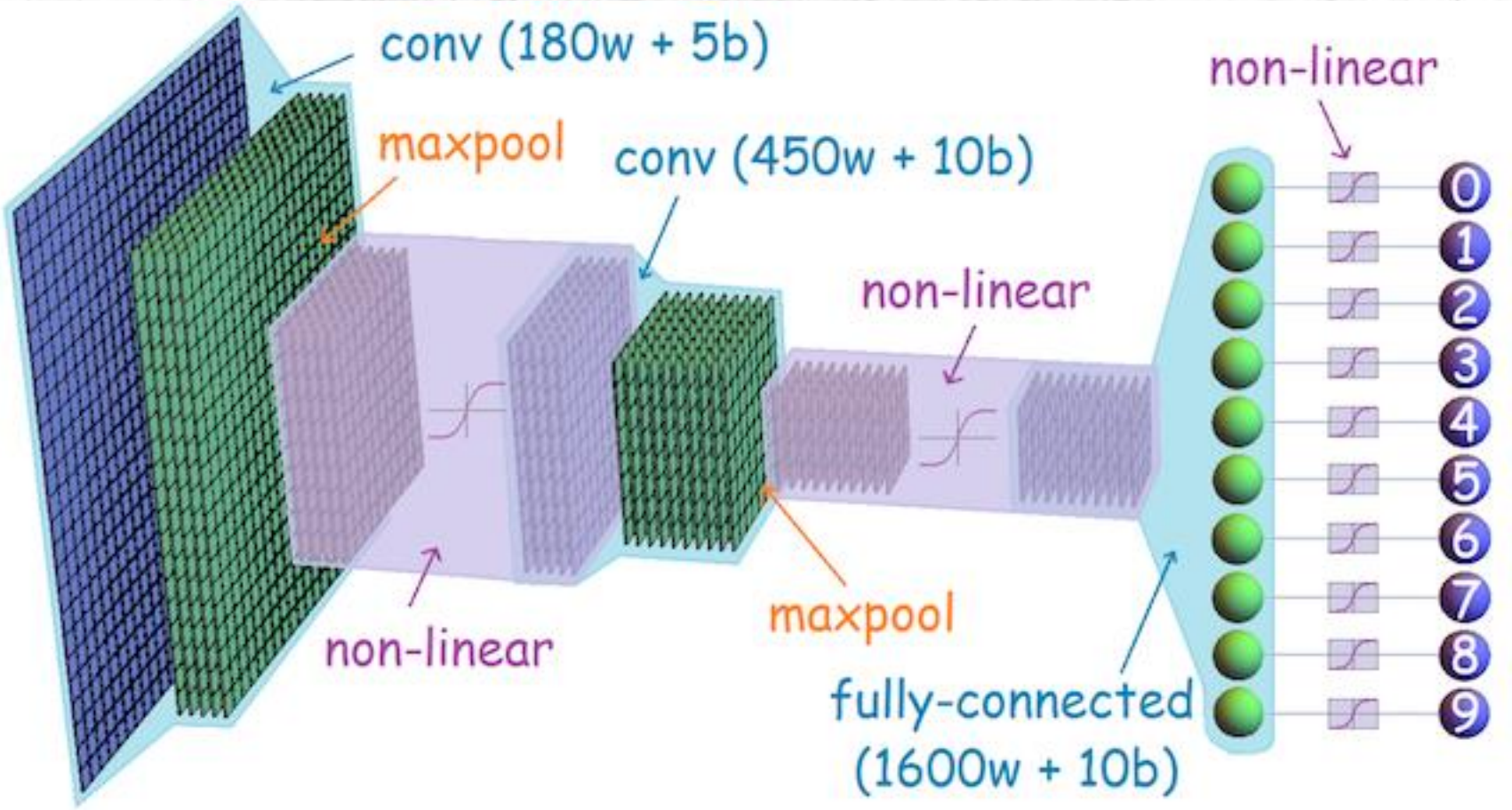
Redes neuronales convolucionales



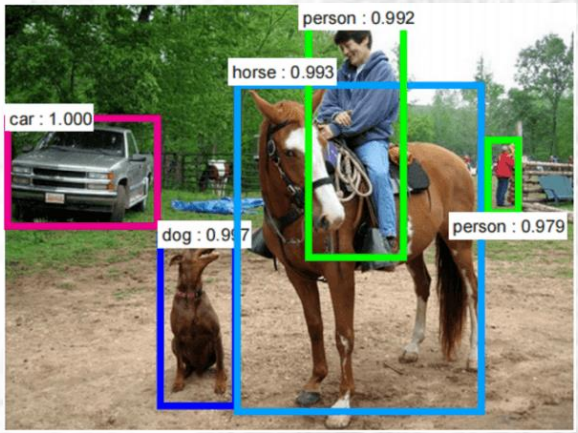
Una red convolucional integra dos partes

- Capas convolucionales + Capas Pooling: extractor de rasgos.
- Capas densas: clasificador.

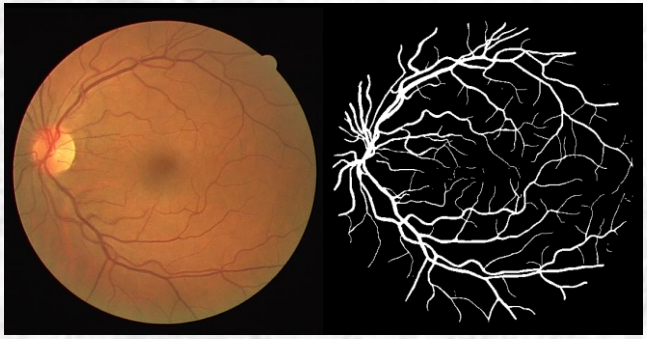
Redes neuronales convolucionales



Aplicaciones:

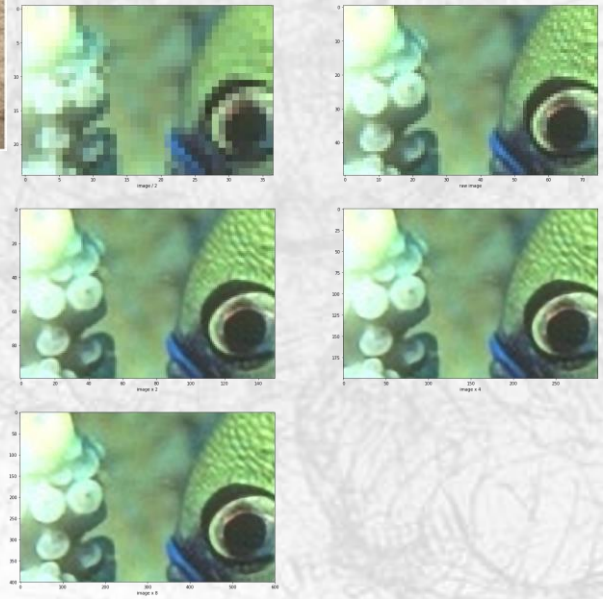


Object Detection



Segmentation

Style transfer



Super-resolution

Filtering

Todas usan redes convolucionales !

Objetivo de hoy

Base de datos MNIST:

70,000 imágenes (de tamaño 28 x 28).
(de ellos, 60,000 son de entrenamiento y 10,000 son de prueba).

Vamos a implementar una **red convolucional** que aprendan a clasificar los dígitos de MNIST.