Filtros Convolucionales

Resumen Problemas de Clasificación

Hasta ahora...

- **Regresor Logístico** (Lineal): 1 capa de salida
- **Redes Neuronales** (no lineal): al menos una capa oculta + 1 capa de salida
- Siempre tantas neuronas de salida como clases a clasificar.
- Tipos de problemas:
  - 1 o 2 Features: podemos graficar los datos y las fronteras de decisión.
  - Imágenes: es un caso particular de N-features donde podemos interpretar los datos visualmente.
- Métricas
  - **Train set**: para entrenar. **Test set**: para validar el modelo con nuevos datos.
  - **Accuracy**: nos dice como funciona el modelo de forma global.
  - **Precision**/**Recall**: lo usamos para clasificación binaria. Explica mejor cómo detecta los True Positives.

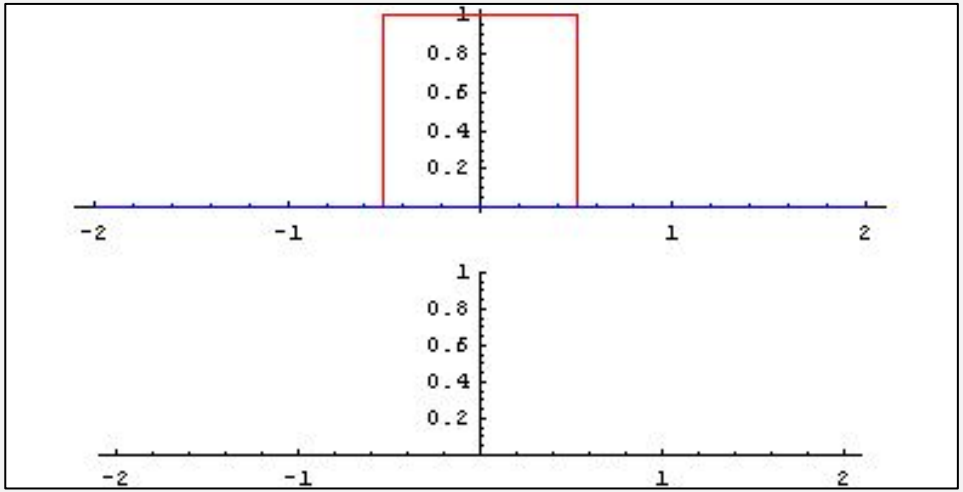
Filtros Convolucionales

- También llamada **salida** o **feature (característica)**

Convolución

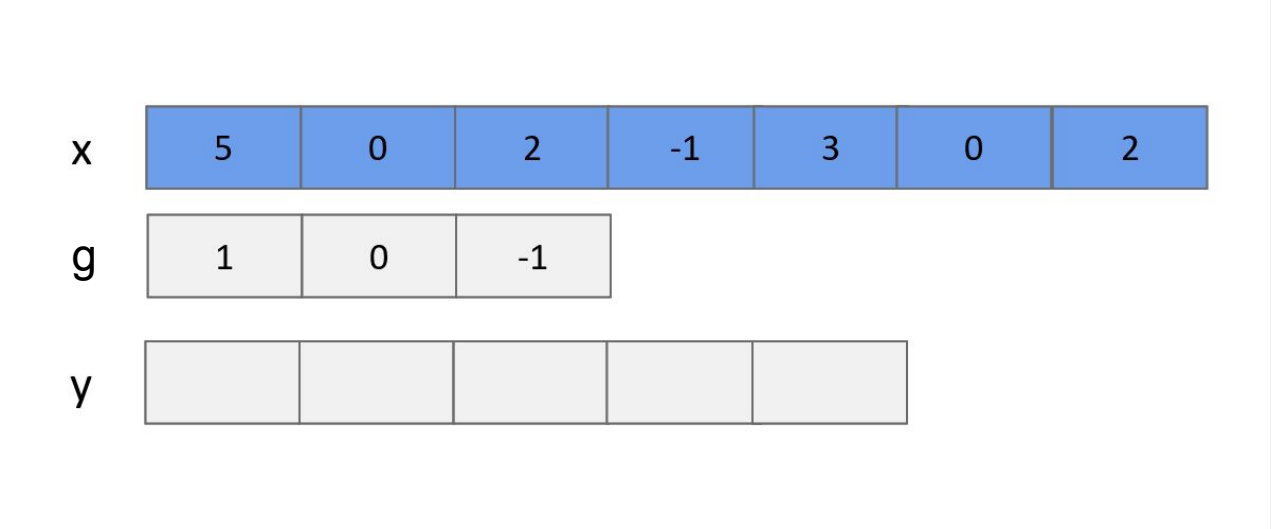
- Operación sobre dos funciones *f* y *g*, que produce una tercera función que puede ser interpretada como una versión “filtrada” de *f*.
- En funciones unidimensionales se utiliza para realizar diferentes filtros en señales o modelar estímulos en simulaciones.
- Si bien la convolución se define en forma continua, a nosotros nos interesa la versión discreta.

$$f[x] * g[x] = \sum_{k=-\infty}^{\infty} f[k] \cdot g[x-k]$$



Filtro discreto

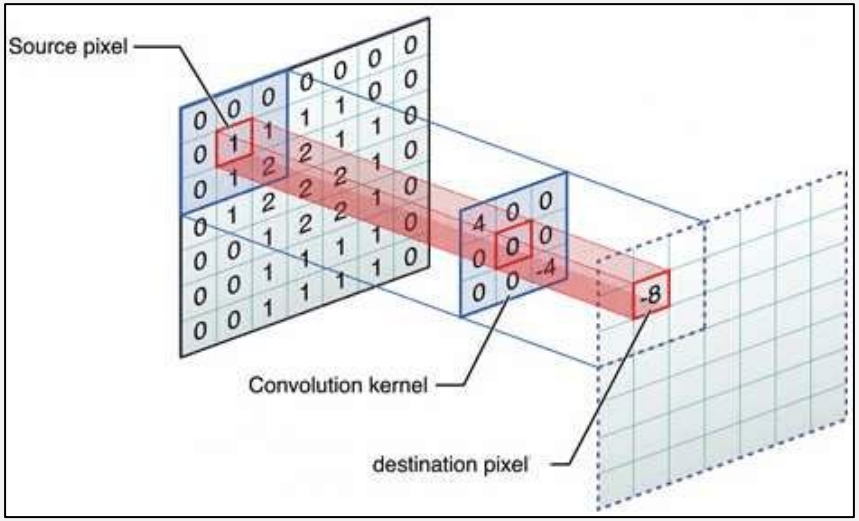
- Es un vector o matriz de valores
- Pesa los valores de la imagen para producir la salida
- También llamado Kernel, Núcleo, Filter
- Tamaño del kernel = Kernel size = Cantidad de valores = K
  - Ejemplo con K = 3



Convolución 2D

Siguiendo la misma idea, podemos extender el concepto de convolución sobre matrices. Es decir, una convolución en 2 dimensiones.

Esto nos sirve para imágenes en escala de grises.



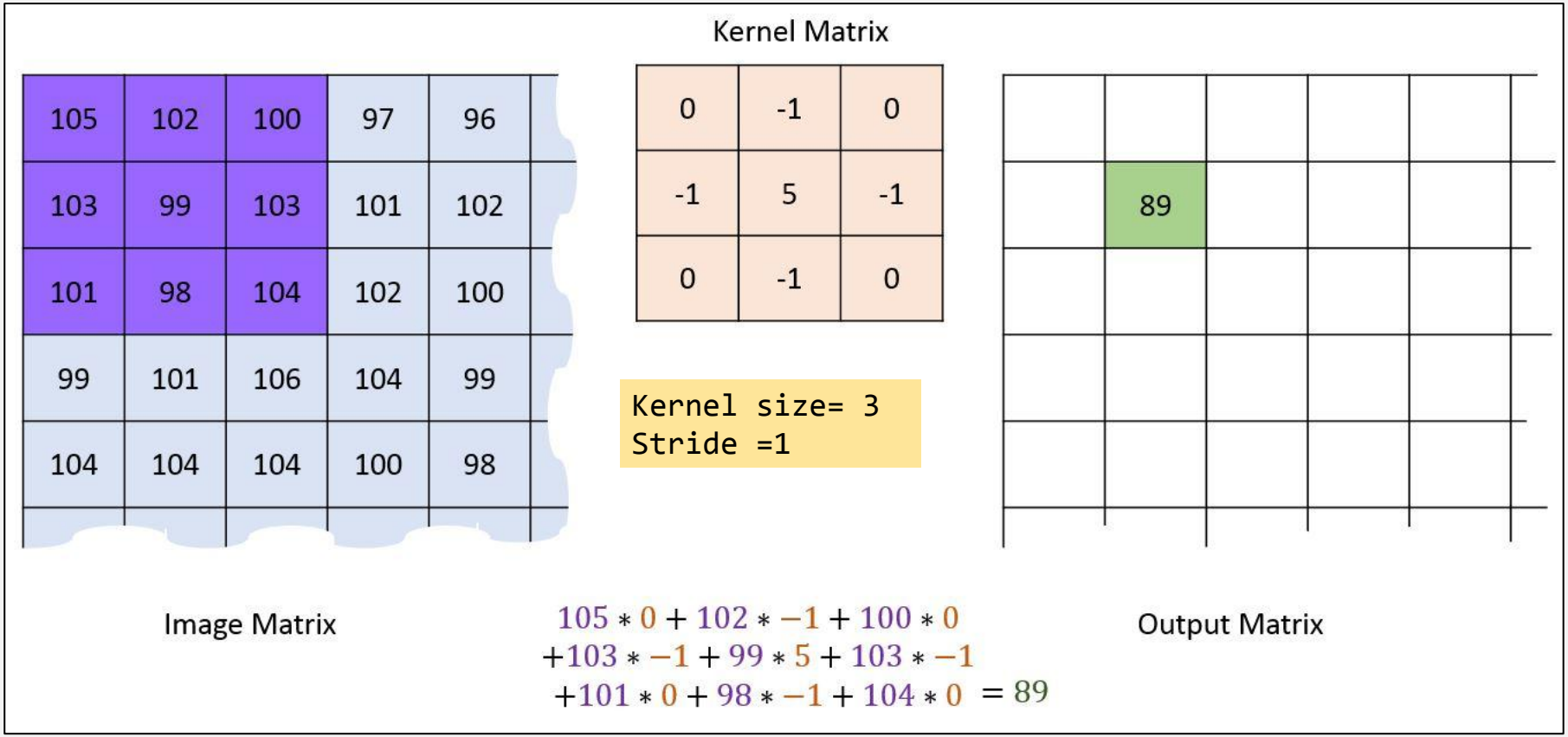
$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

Convolución 2D

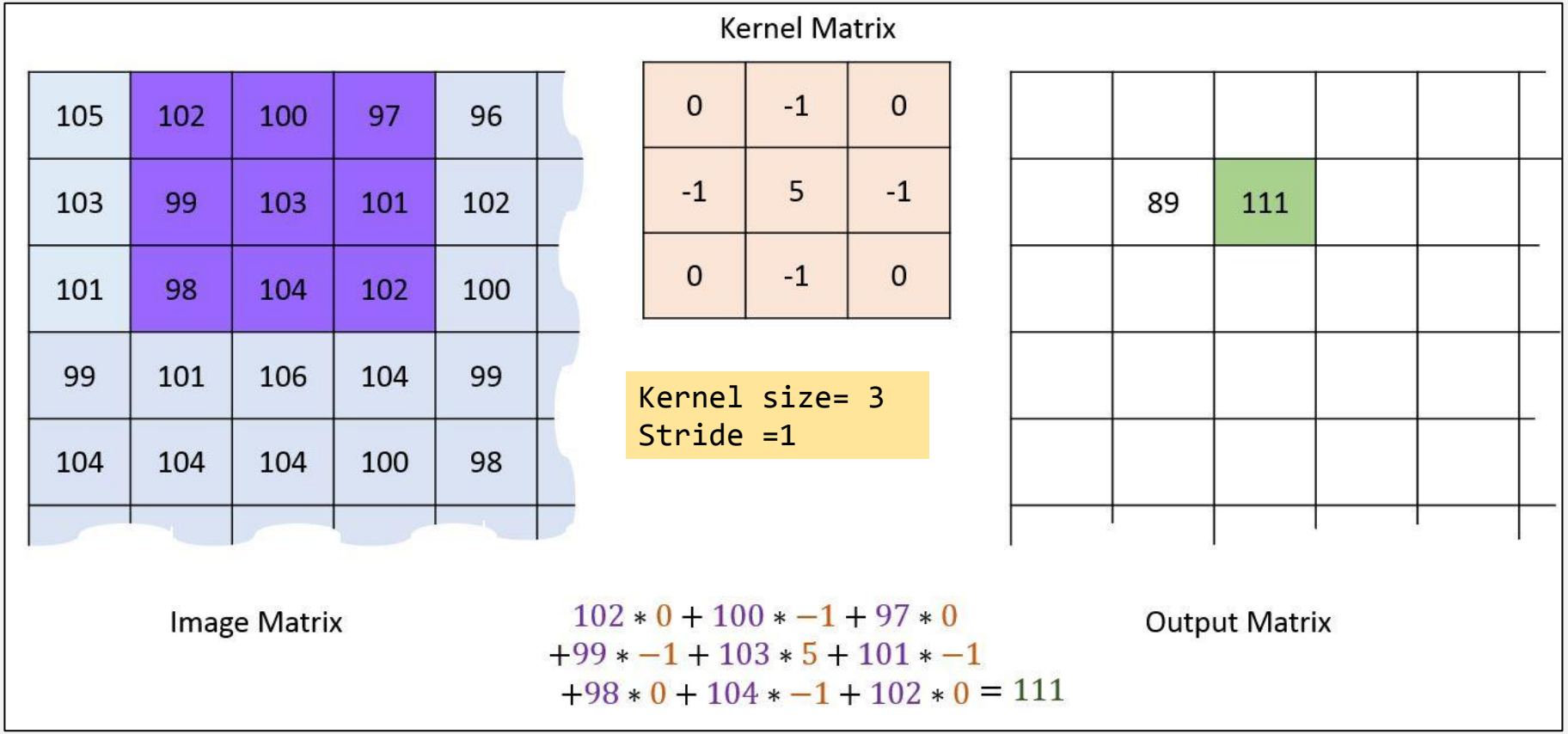
1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

4		

Convolución 2D



Convolución 2D



Tamaños de Kernel (y stride/padding)

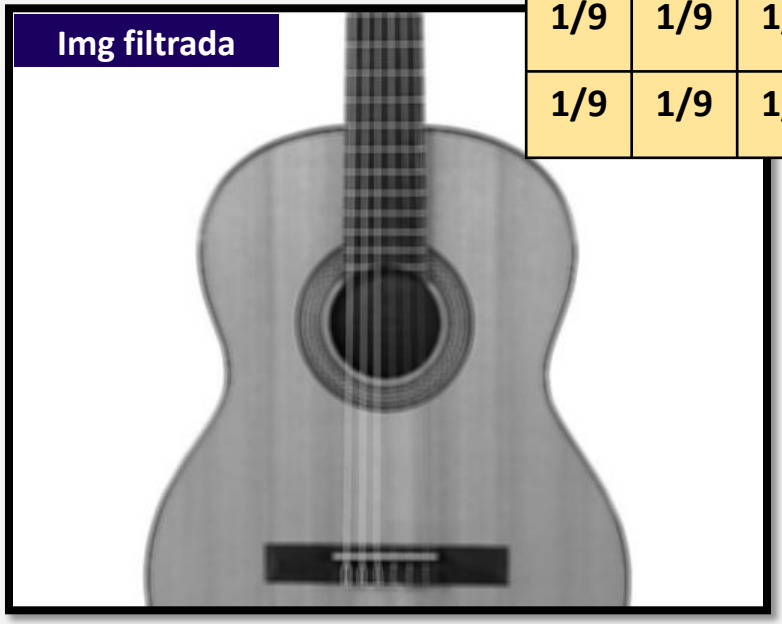
- Entrada 1D de tamaño N
  - Kernel de tamaño K
  - Ejemplo:
    - Vector de 5 elementos, K de 3 elementos
- Entrada 2D de tamaño HxW
  - Kernel de tamaño K_hxK_w
  - Ejemplo
    - Imagen de 7x9, Kernel de tamaño 3x4
  - El Kernel suele ser Cuadrado
    - Kernel de tamaño K significa que K_h=K_w = K
      - Idem para Padding/Stride (más adelante)
  - K suele ser impar (1, 3, 5, 7)
    - Simplifica cálculos

Efecto de la convolución

Veamos ahora cuál es el efecto de aplicar algunos *kernels* clásicos.

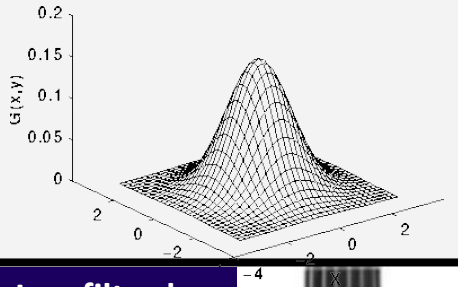
Simple promedio de todos los píxeles: borronea la imagen.

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

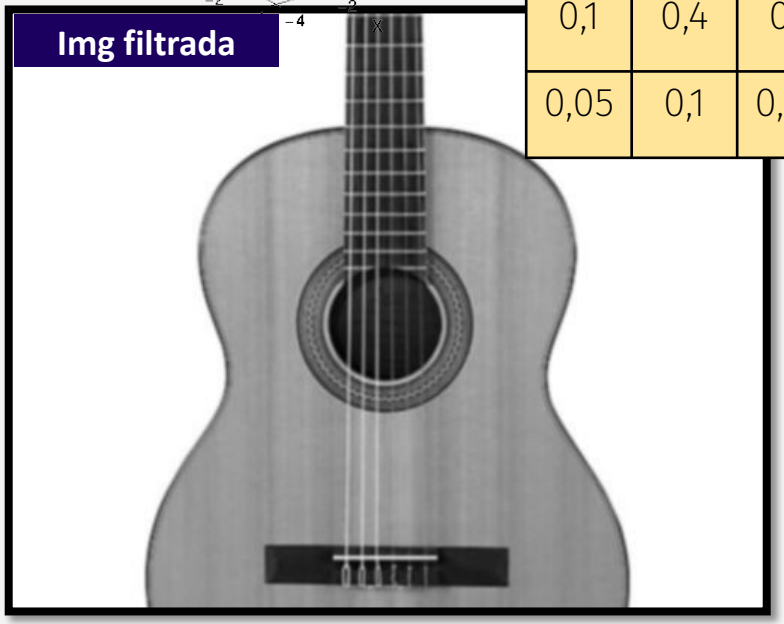


Efecto de la convolución

Filtro gaussiano
(filtro de pasa baja)

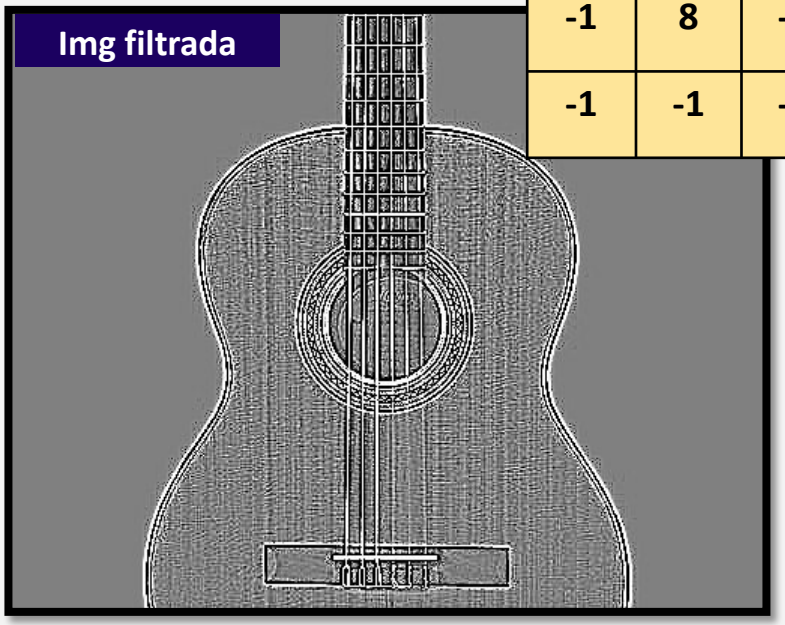


0,05	0,1	0,05
0,1	0,4	0,1
0,05	0,1	0,05



Efecto de la convolución

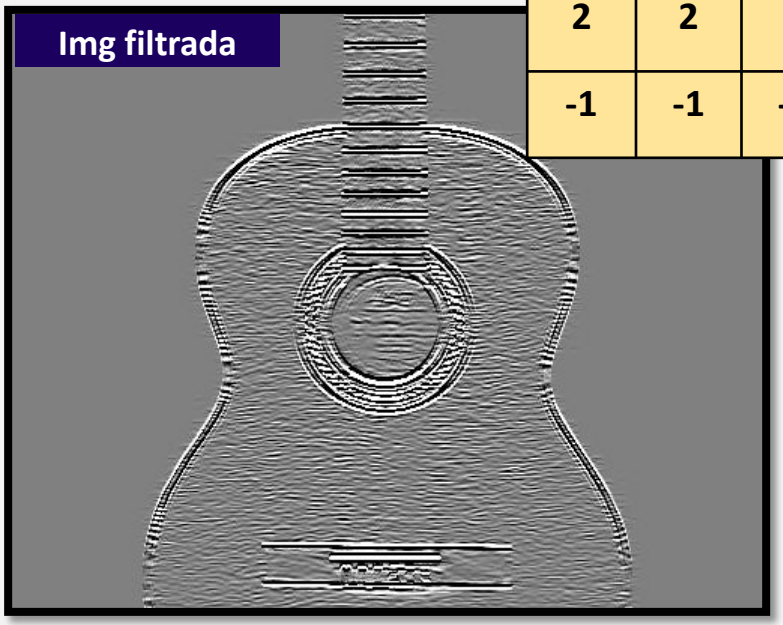
Detección de bordes (filtro de pasa alta)



-1	-1	-1
-1	8	-1
-1	-1	-1

Efecto de la convolución

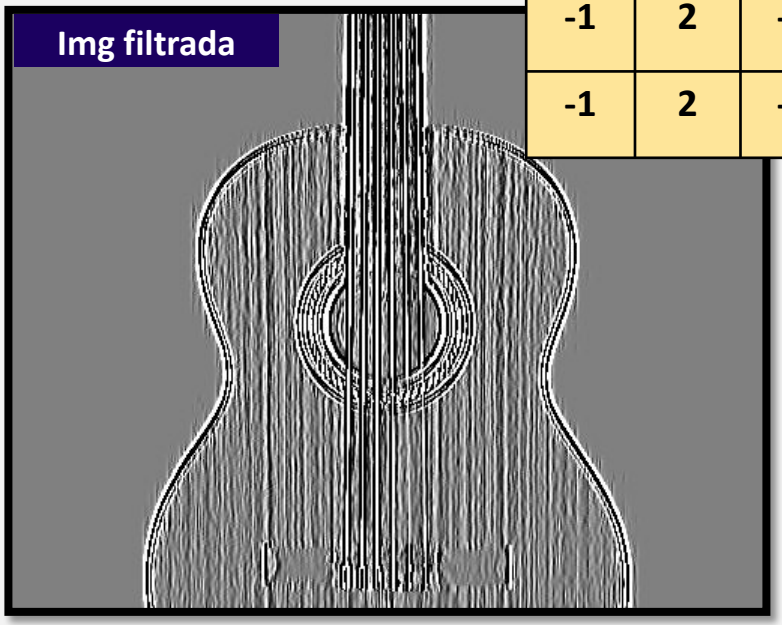
Bordes horizontales



-1	-1	-1
2	2	2
-1	-1	-1

Efecto de la convolución

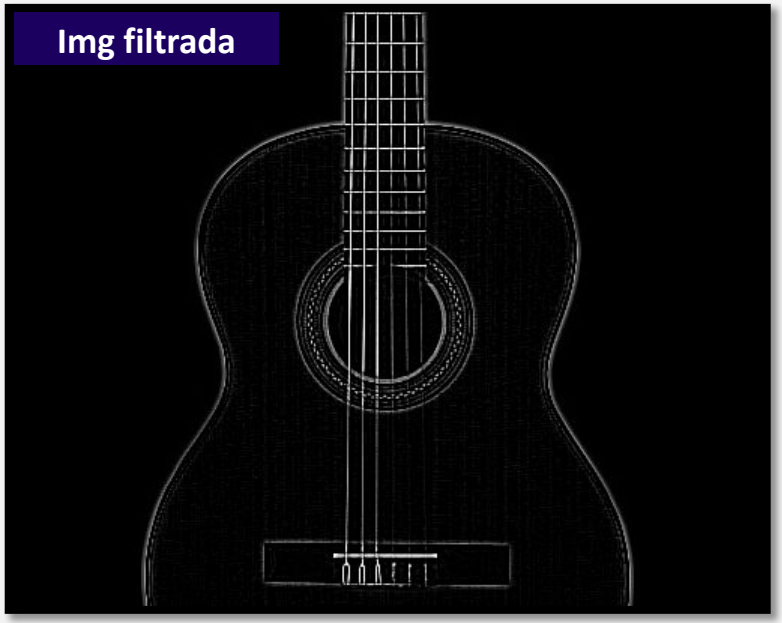
Bordes verticales



-1	2	-1
-1	2	-1
-1	2	-1

Efecto de la convolución

Kernel Gaussiano + Kernel Bordes



Tamaño del Kernel (suavizado)

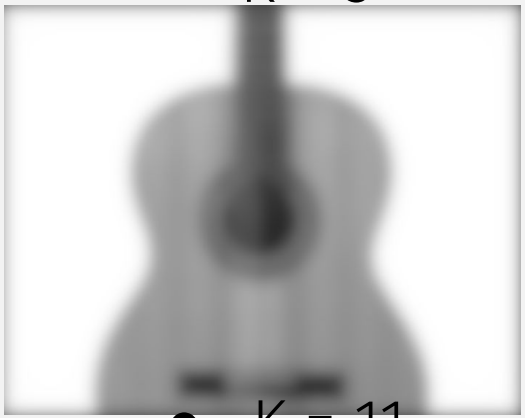
- K = 1

- K = 3

- K = 5

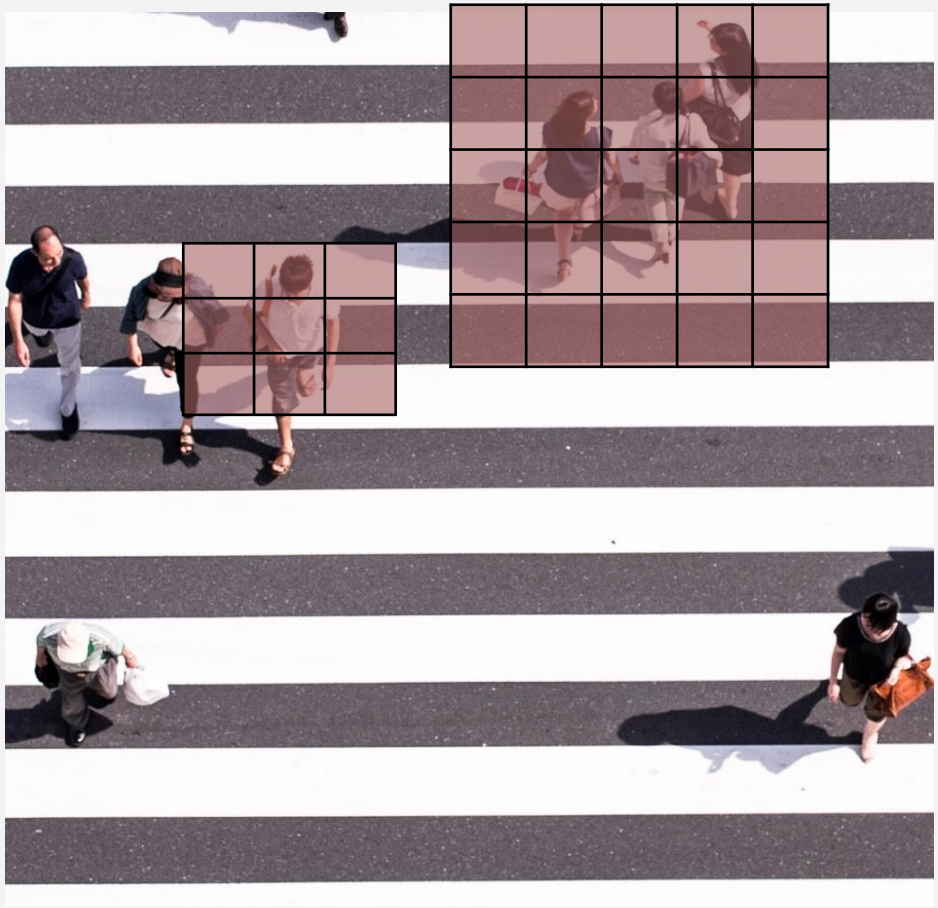
- K = 7

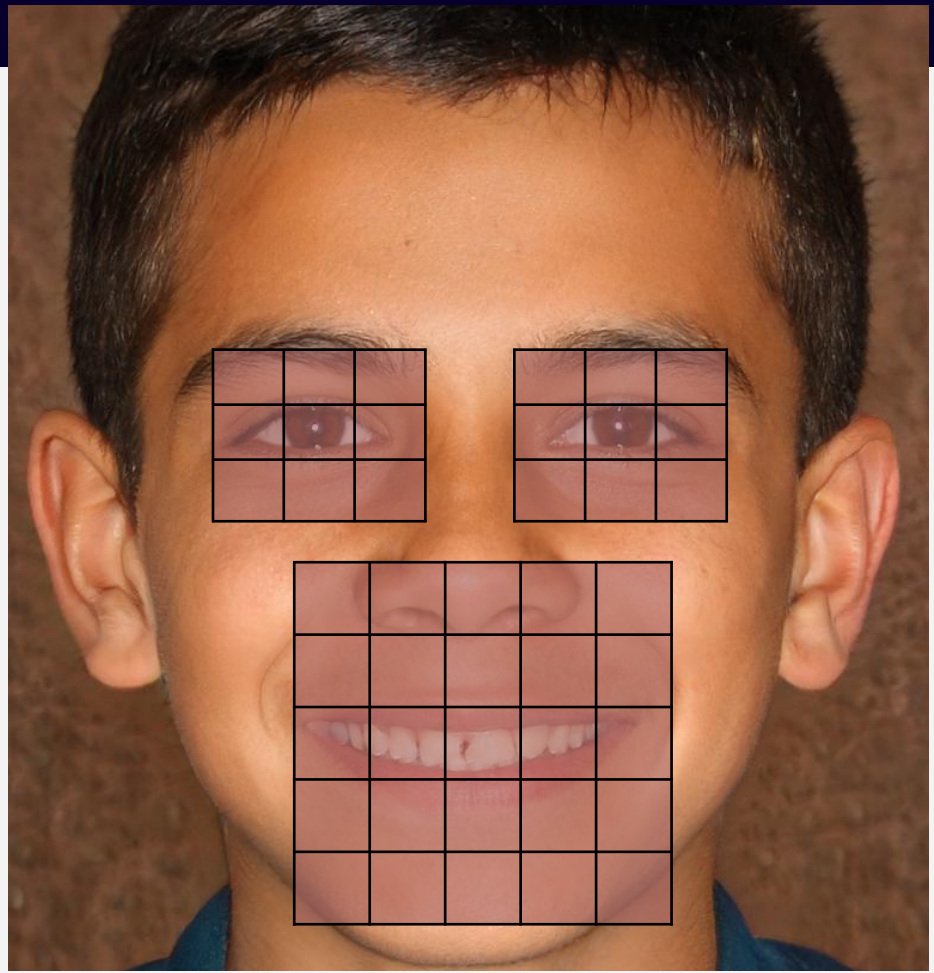
- K = 9



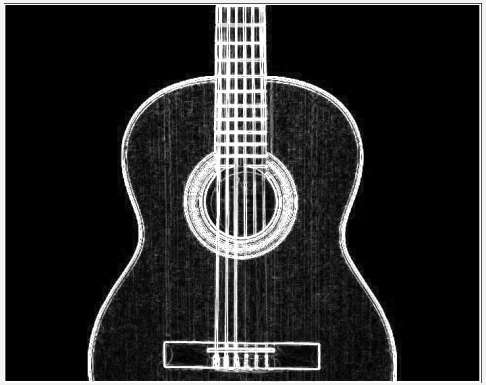
- K = 11

Tamaño del Kernel





Tamaño del Kernel (líneas)



- K = 1
- K = 3
- K = 5

Tamaño del Kernel y múltiples filtrados

- K = 1

- K = 3

- K = 5

- K = 7

- K = 9

- K = 1

- K = 3

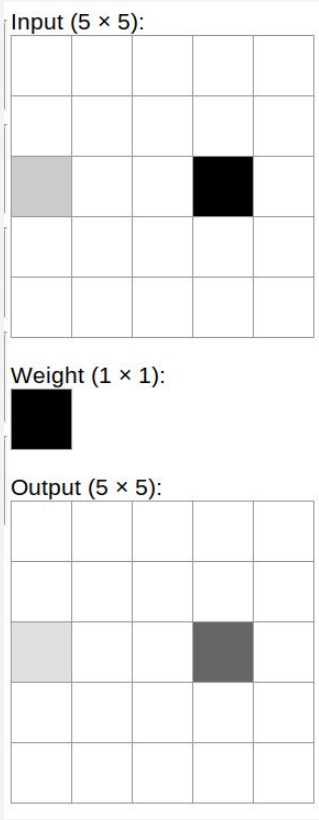
- K = 3
- 2 veces

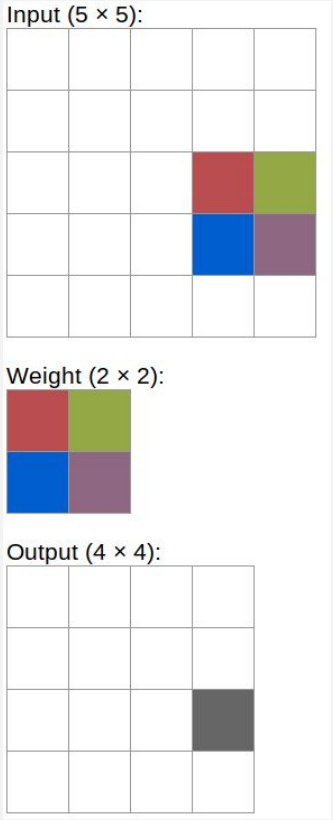
- K = 3
- 3 veces

- K = 3
- 4 veces

Tamaño de la salida vs tamaño kernel K

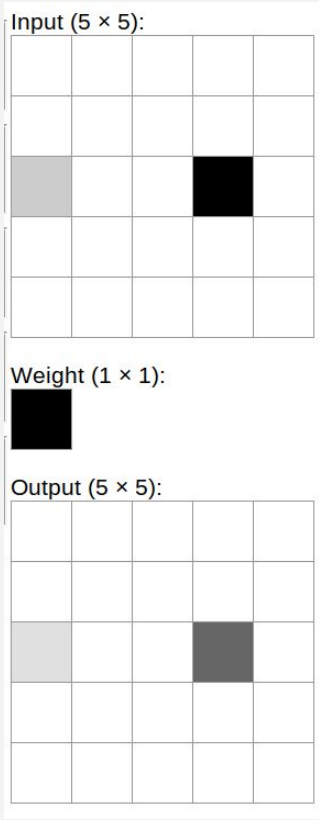
- Kernel KxK → Input HxW → Output H'xW' → **H' = H - (K-1)** y **W'=W- (K-1)**

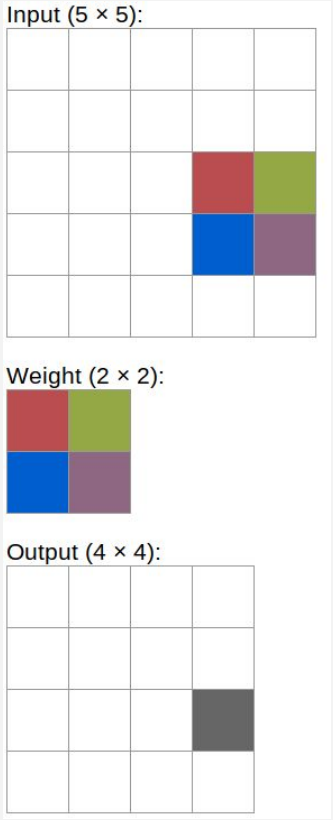




Kernel y Cantidad de parámetros

- Kernel de KxK → KxK parámetros. Ejemplo, Kernel de 5x5 → 25 parámetros

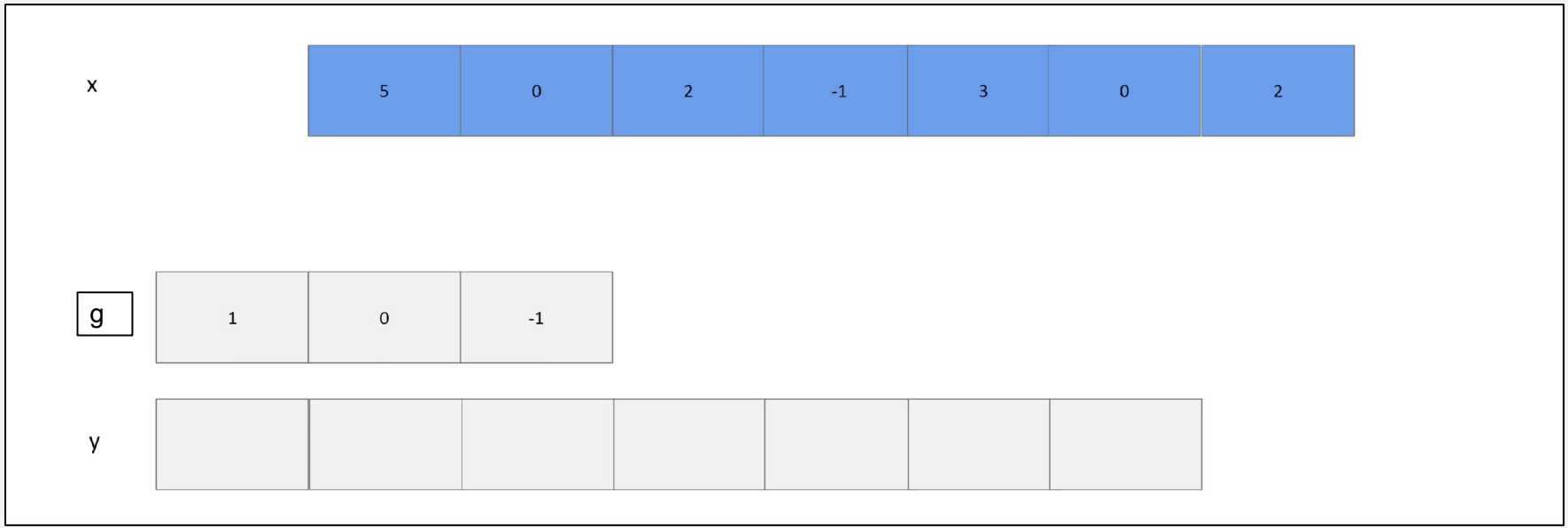




Convolución 1D con Padding

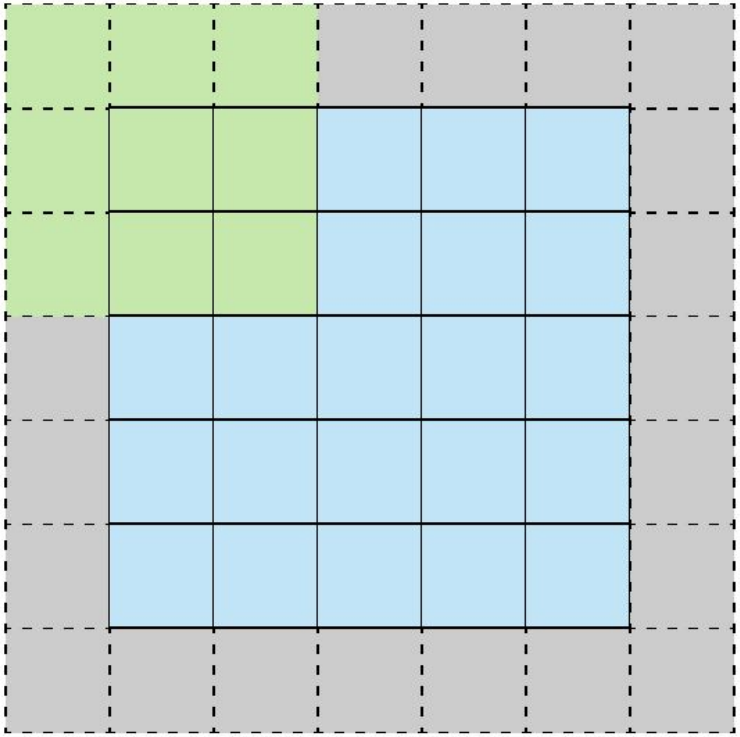
Aplicar el filtro de forma discreta ocasiona dos problemas:

- Pérdida de información en los bordes.
- Reducción del tamaño final del vector.
  - Padding con ceros

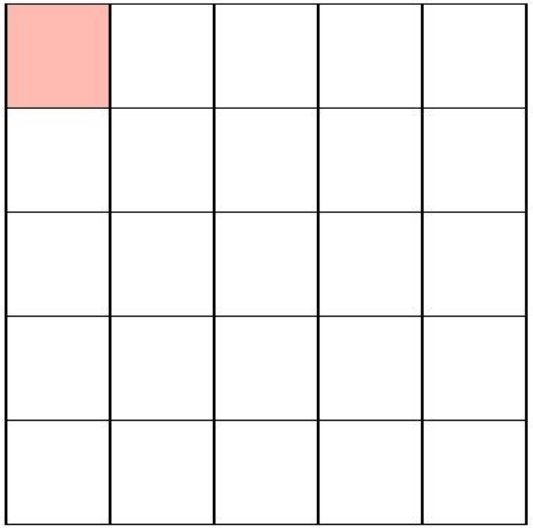


Kernel size =3, stride = 1, padding = 1

Convolución 2D con padding



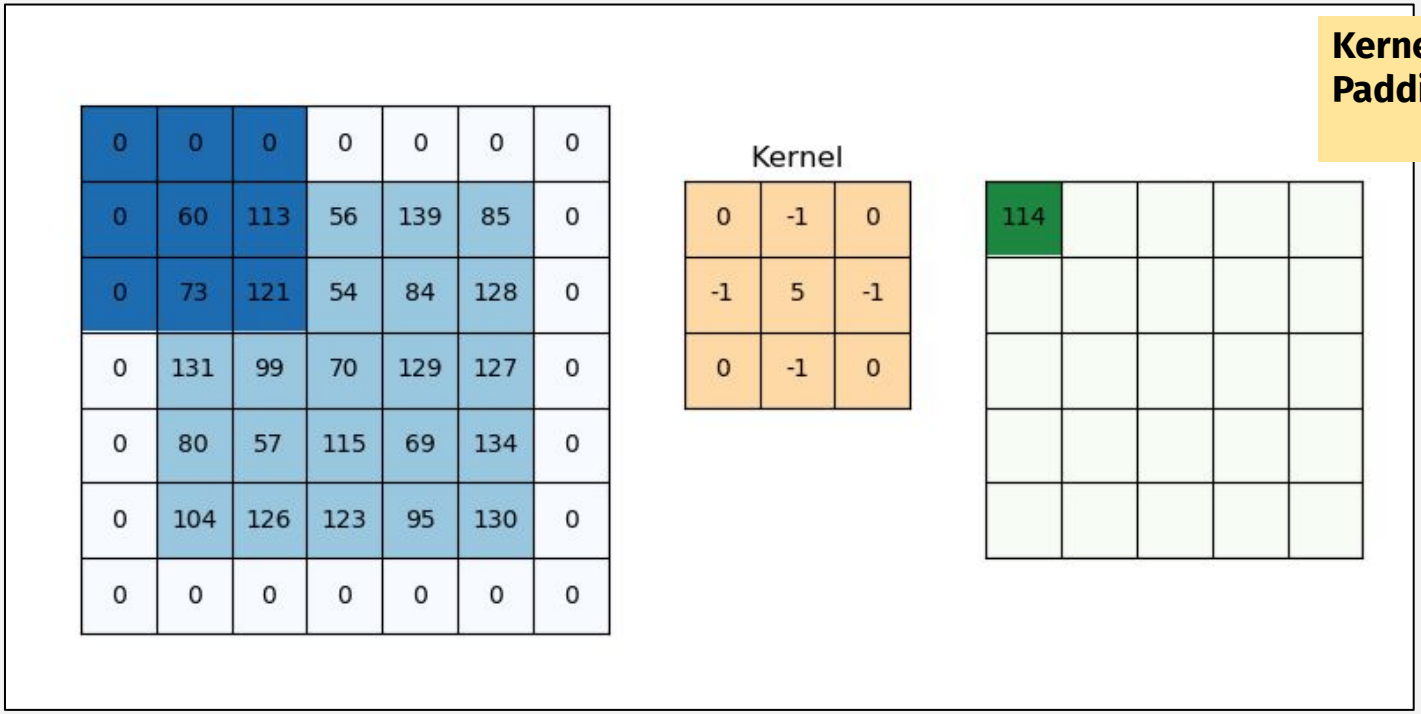
Stride 1 with Padding



Feature Map

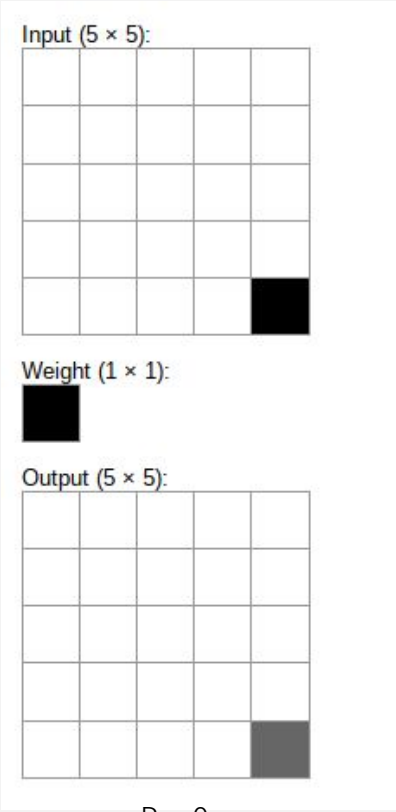
Convolución 2D con padding

- Puede usarse para que
  - tamaño de entrada = tamaño salida

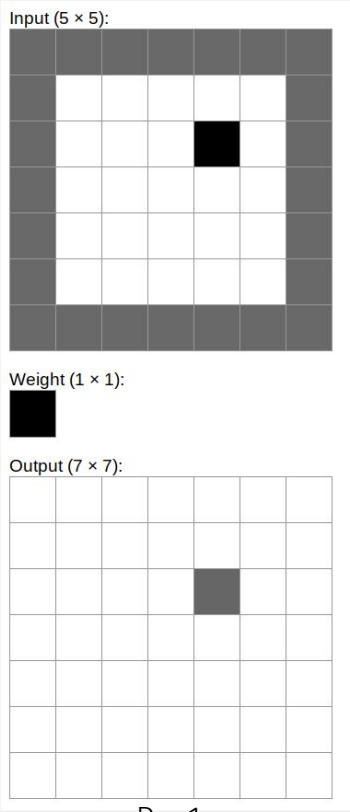


Tamaño de la salida vs padding P

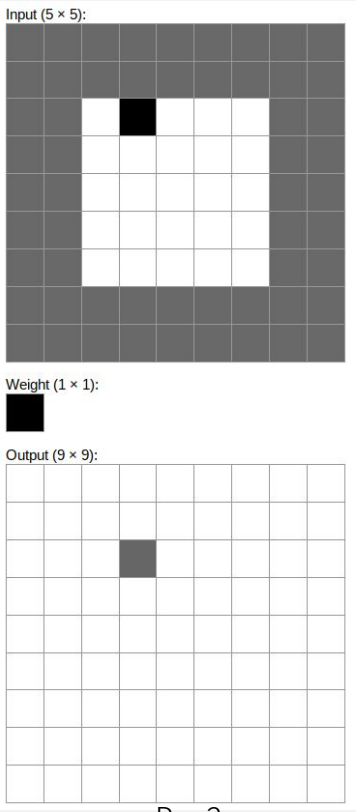
- Kernel KxK → Input HxW → Output H'xW' → **H' = H + 2P - (K-1)** y **W'=W + 2P - (K-1)**



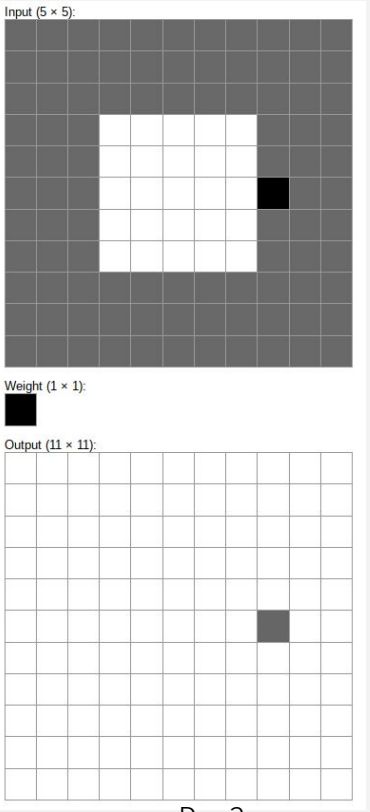
- P = 0



- P = 1



- P = 2



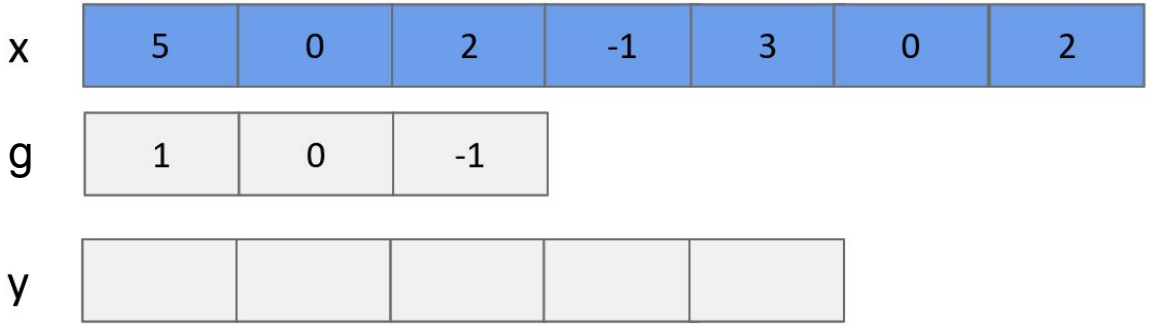
- P = 3

Salida vs Padding vs Kernel

- Kernel KxK → Input HxW → Output H'xW'
  - **H' = H + 2P - (K-1)**
  - **W'=W + 2P - (K-1)**
- **Ejemplo 1**
  - Entrada: 10x10
  - **K = 5**
  - **P = 1**
  - Salida
    - H' = 10 + 2***1** - **4** = 8
    - W' = 10 + 2***1** - **4** = 8
- **Ejemplo 2**
  - Entrada: 10x10
  - **K = 5**
  - **P = 2**
  - Salida
    - H' = 10 + 2***2** - **4** = 10
    - W' = 10 + 2***2** - **4** = 10

Stride o paso en la convolución

$$f[x] * g[x] = \sum_{k=-\infty}^{\infty} f[k] \cdot g[x-k]$$

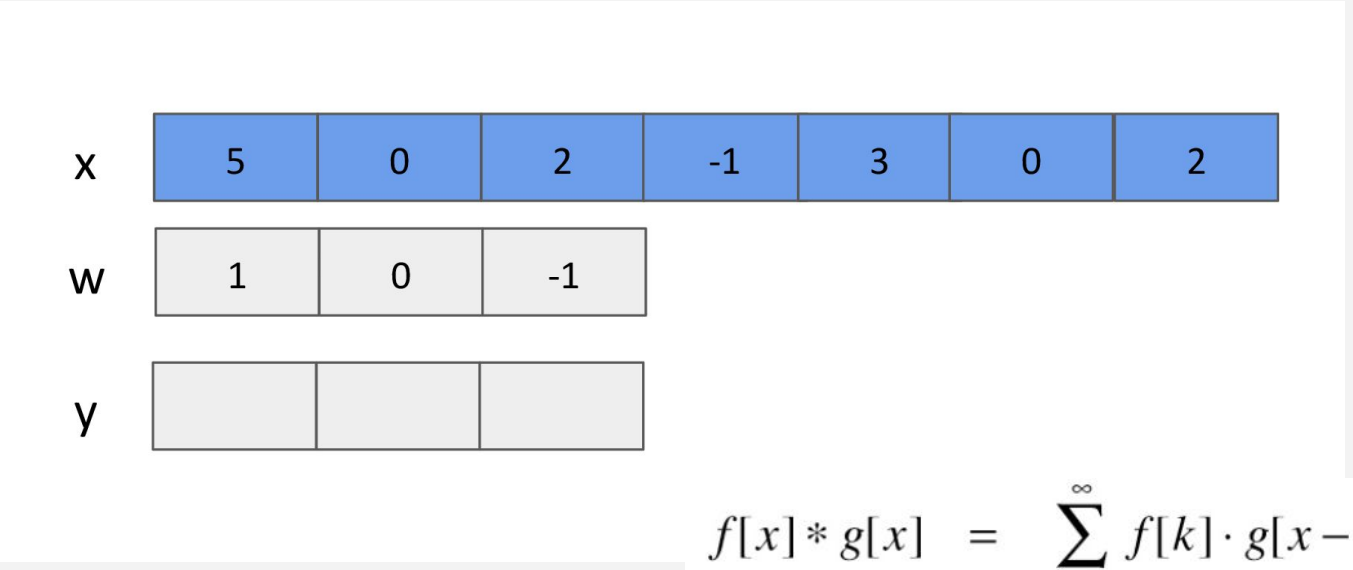


Parámetros:

Kernel_Size: Es el tamaño del filtro utilizado. En este caso = 3

Stride: Es el número de saltos que da el filtro cada vez que se aplica. En este caso = 1.

Stride o paso en la convolución

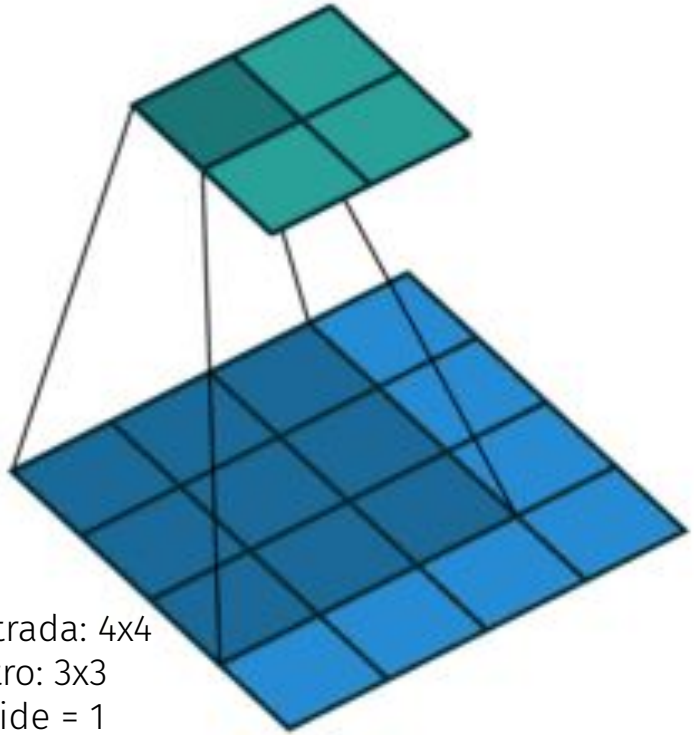


$$f[x] * g[x] = \sum_{k=-\infty}^{\infty} f[k] \cdot g[x-k]$$

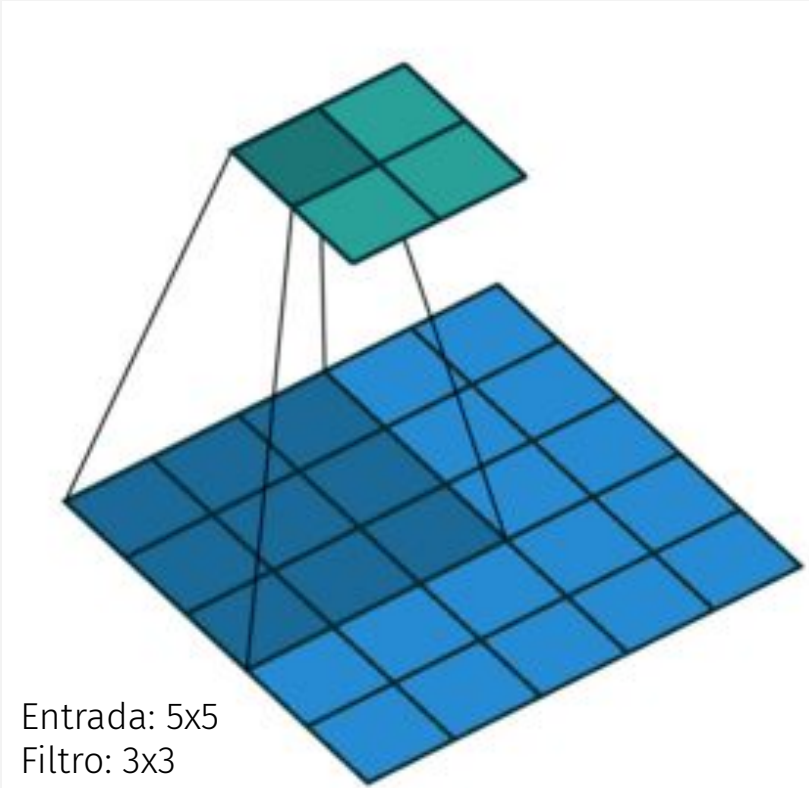
- Kernel Size =3, Stride = 2
  - Disminuye el tamaño de la salida
  - Representación de menor dimensionalidad

Convolución 2D con Stride =2

Convolución 2D con Stride

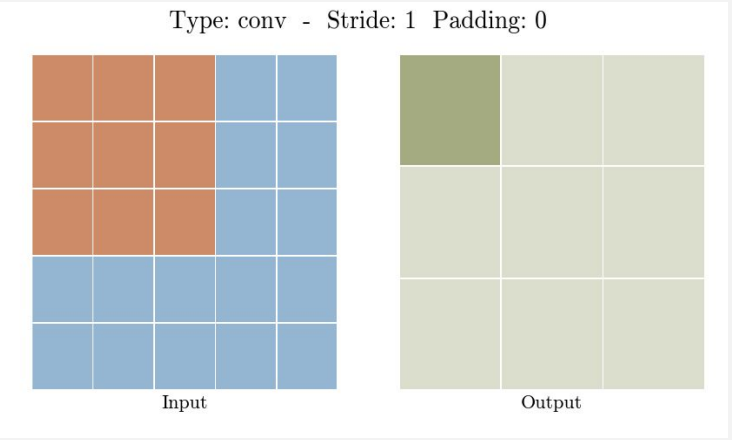


Entrada: 4x4
Filtro: 3x3
Stride = 1
Salida = 2x2



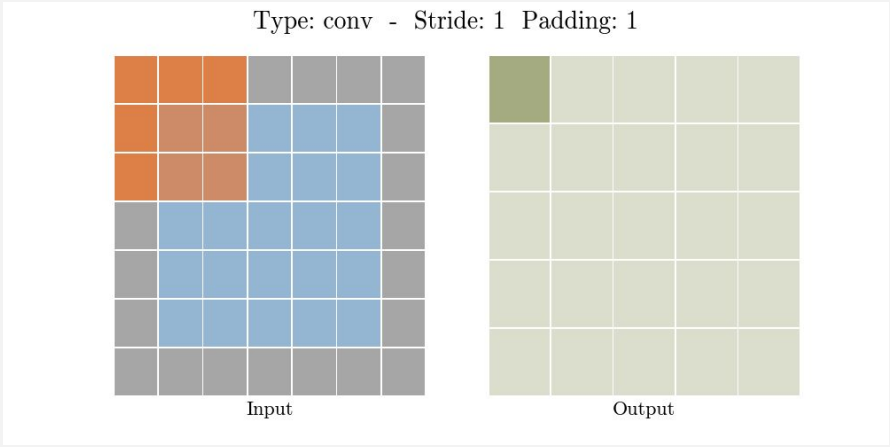
Entrada: 5x5
Filtro: 3x3
Stride = 2
Salida = 2x2

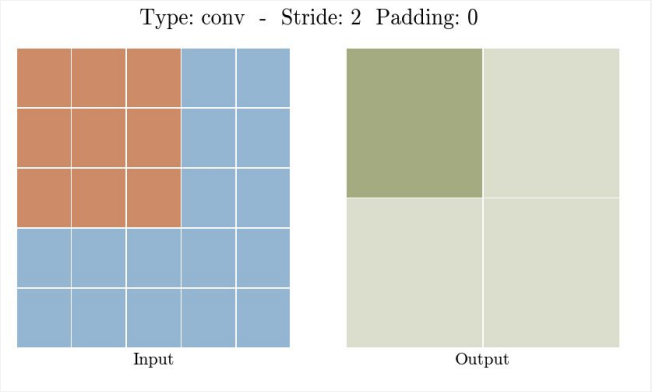
Convolución 2D con Stride o Padding



Entrada: 5x5
Filtro: 3x3
Stride = 1
Padding = 0
Salida = 3x3

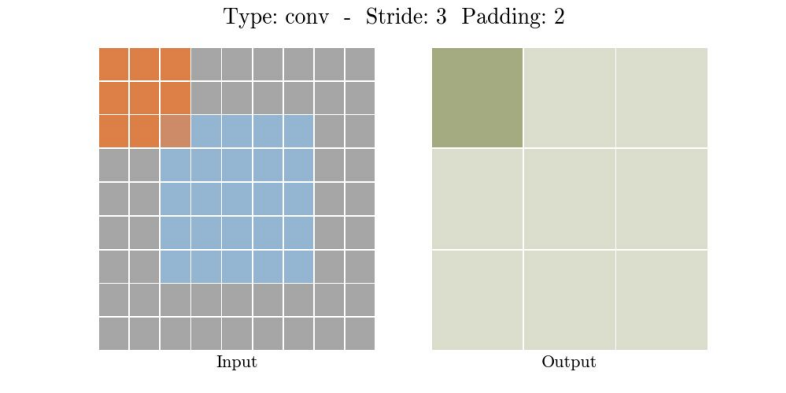
Entrada: 5x5
Filtro: 3x3
Stride = 1
Padding = 1
Salida = 5x5





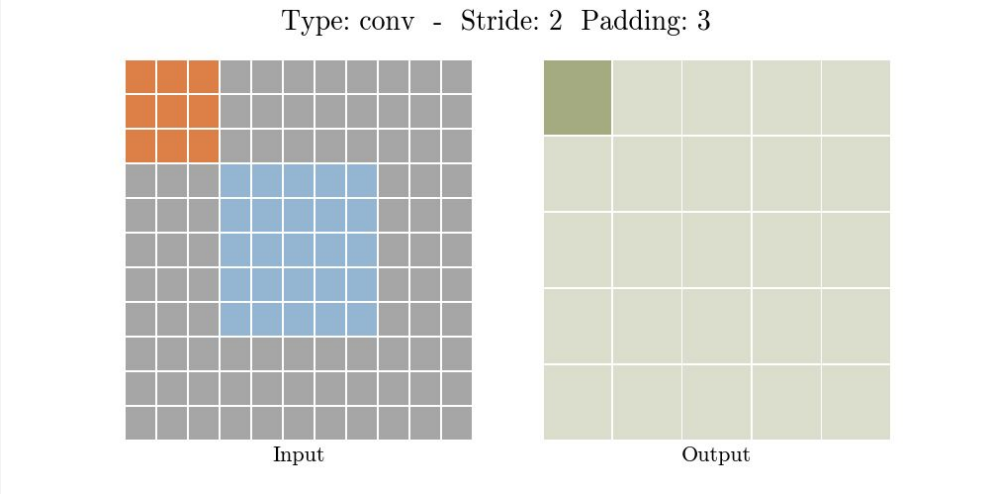
Entrada: 5x5
Filtro: 3x3
Stride = 2
Padding = 0
Salida = 2x2

Convolución 2D con Stride y Padding



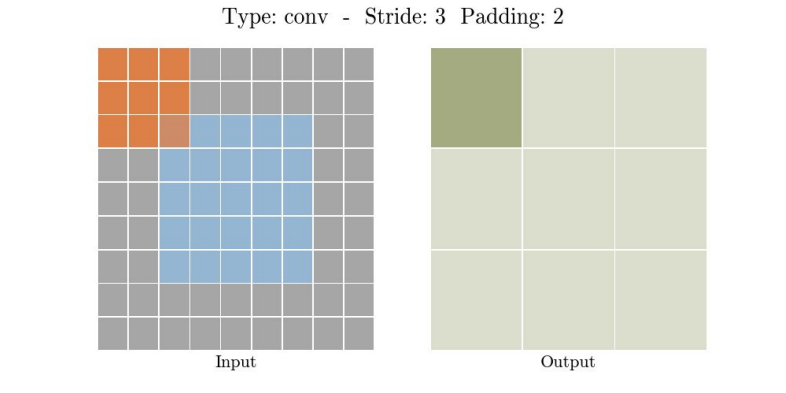
Entrada: 5x5
Filtro: 3x3
Stride = 3
Padding = 2
Salida = 3x3

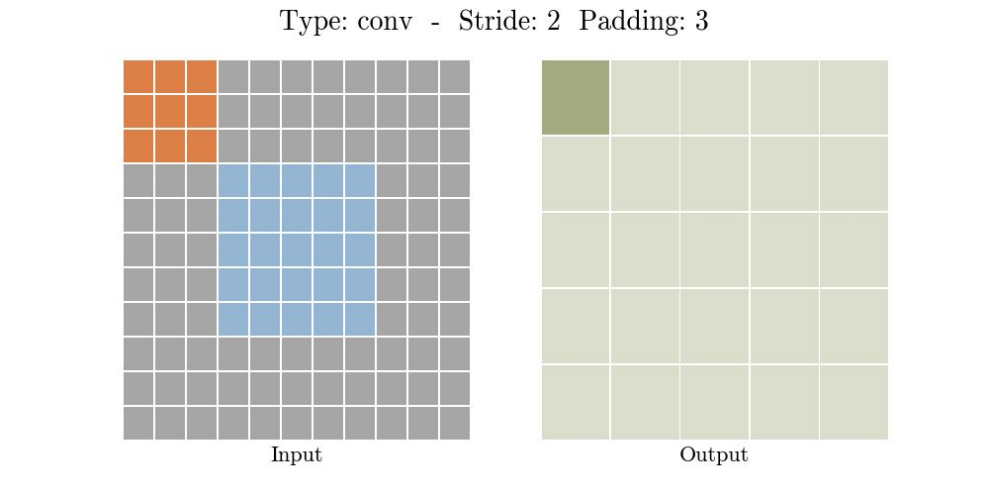
Entrada: 5x5
Filtro: 3x3
Stride = 2
Padding = 3
Salida = 5x5



Cantidad de parámetros con padding y stride

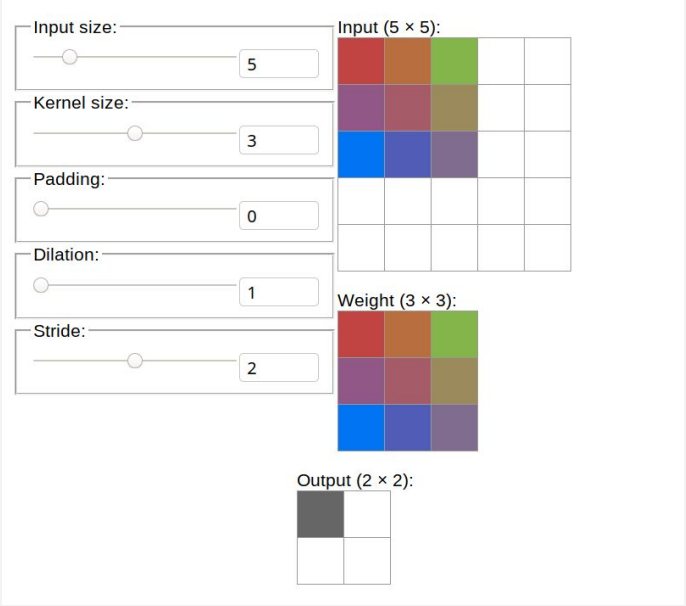
- Padding y Stride afectan cómo se calcula la salida
  - Pero no el tamaño del kernel
    - Kernel de KxK → KxK parámetros.
    - Ejemplo, Kernel de 5x5 → 25 parámetros

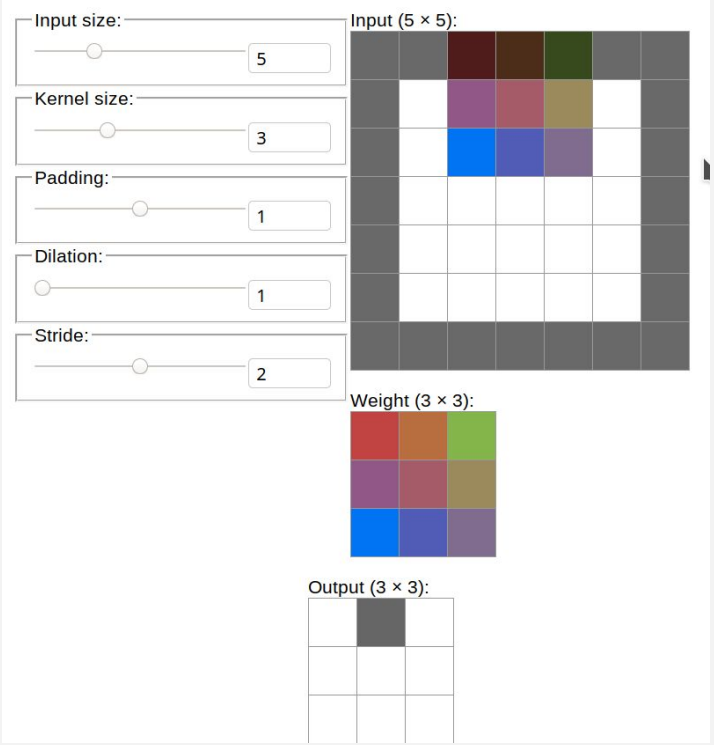




Visualizador de convoluciones

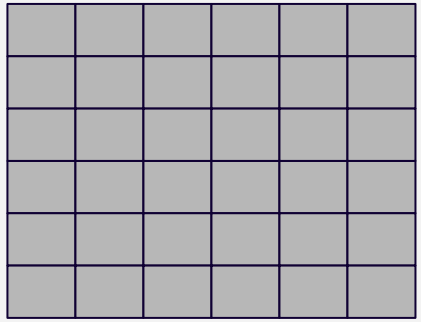
https://ezyang.github.io/convolution-visualizer/index.html

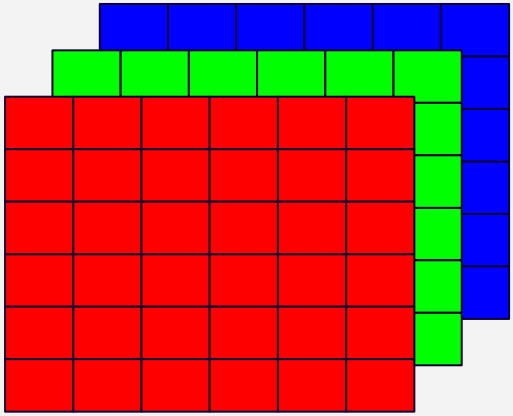


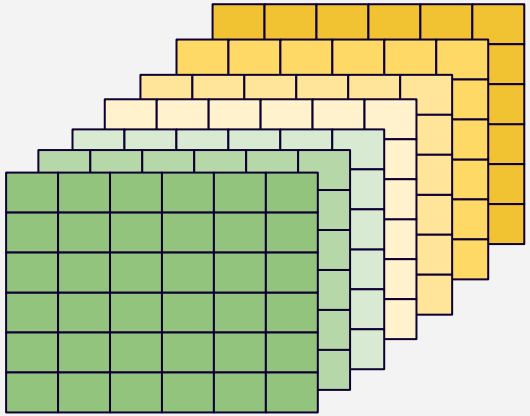


Convolución 2D sobre varios canales

- Imagen escala de grises
  - Tamaño HxW
  - Tamaño HxWx1
- Imagen RGB
  - Tamaño HxWx3
- Imagen multicanal
  - Tamaño HxWxC
- H y W
  - Dimensiones espaciales
- C
  - Dimensión de canales o *features*



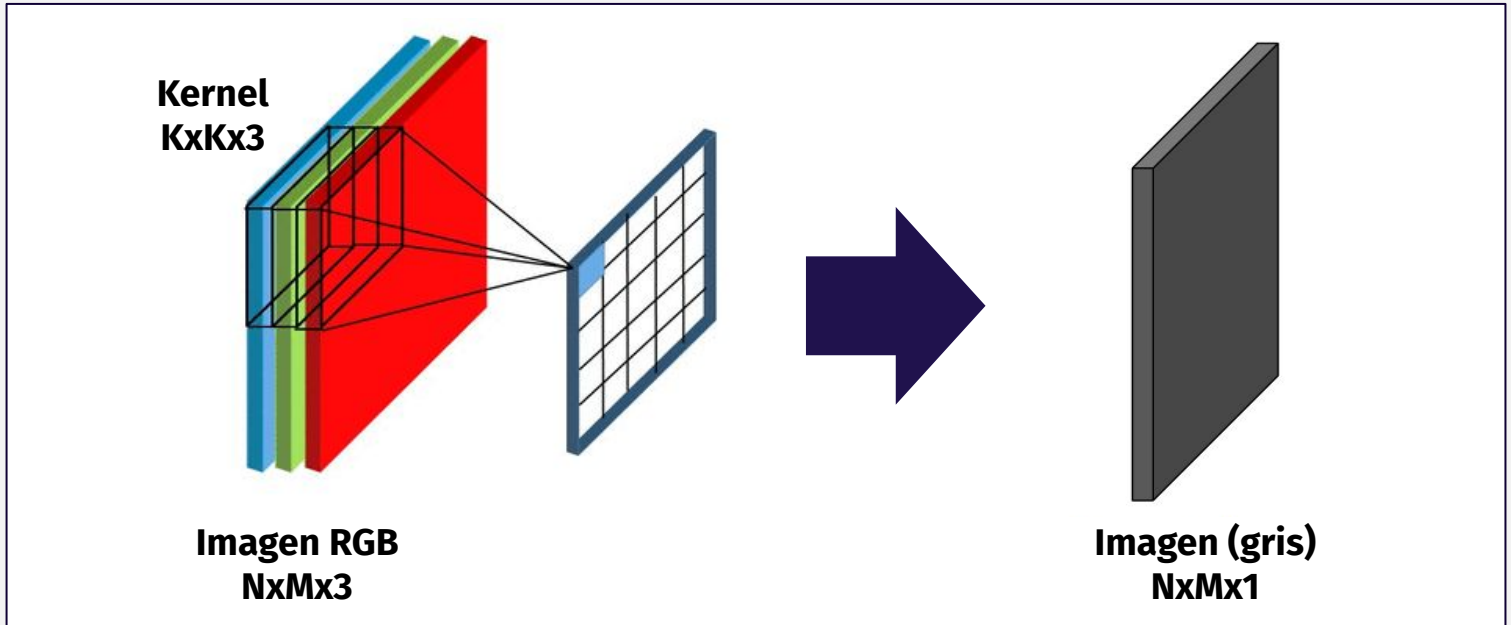




- Ejemplo:
  - Imágenes astronómicas o satelitales

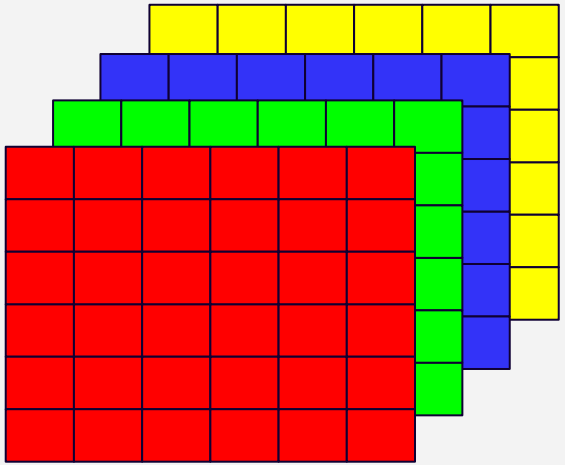
Convolución 2D sobre varios canales

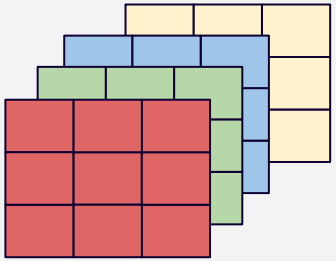
- Se realiza sobre los C canales a la vez
- Convierte una imagen HxWxC en una imagen HxWx1
  - Kernel_Size= KxKxC
- Ejemplo con C=3 (imágenes RGB)

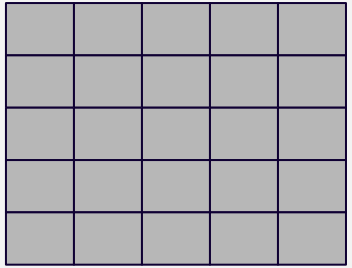


Convolución 2D sobre varios canales

- Se realiza sobre los C canales a la vez
- Convierte una imagen HxWxC en una imagen HxWx1
  - Kernel_Size= KxKxC
- Ejemplo con C=4 y K=3
  - ~~4 filtros de 3x3~~ 1 filtro de 3x3x4

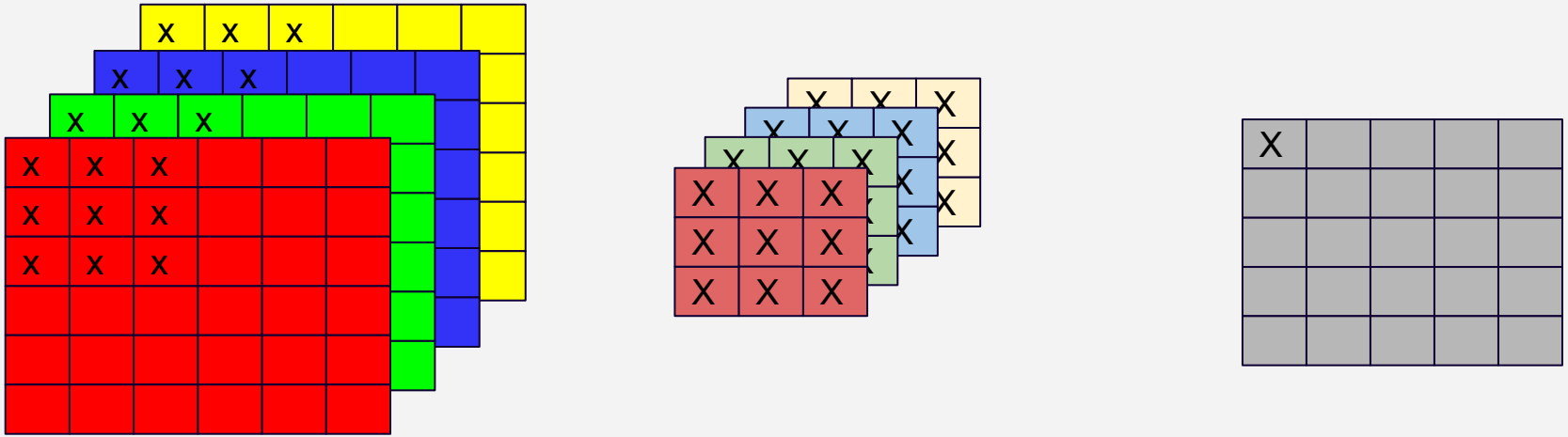






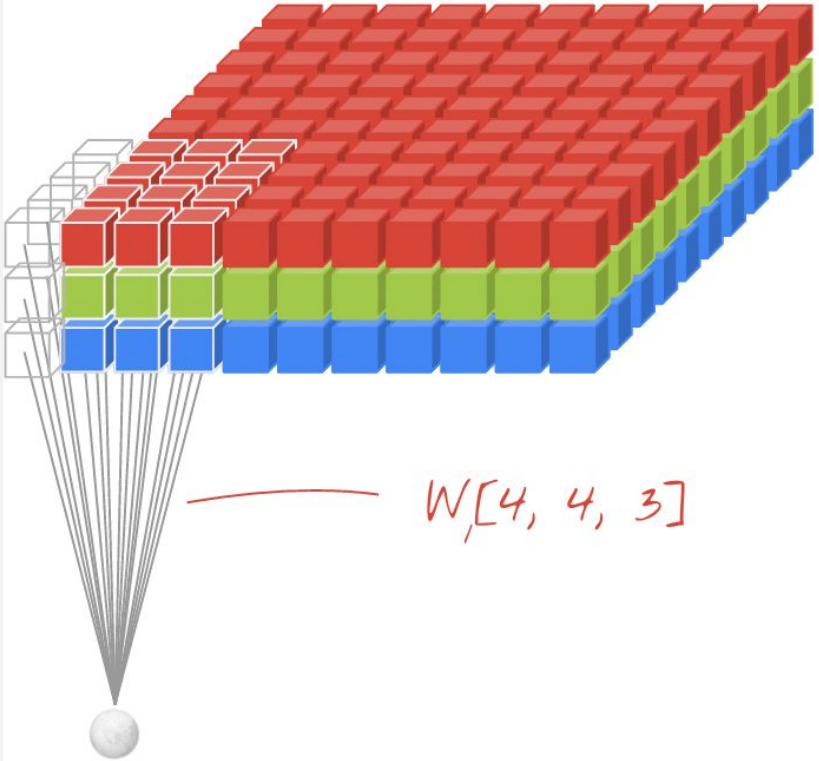
Convolución 2D sobre varios canales

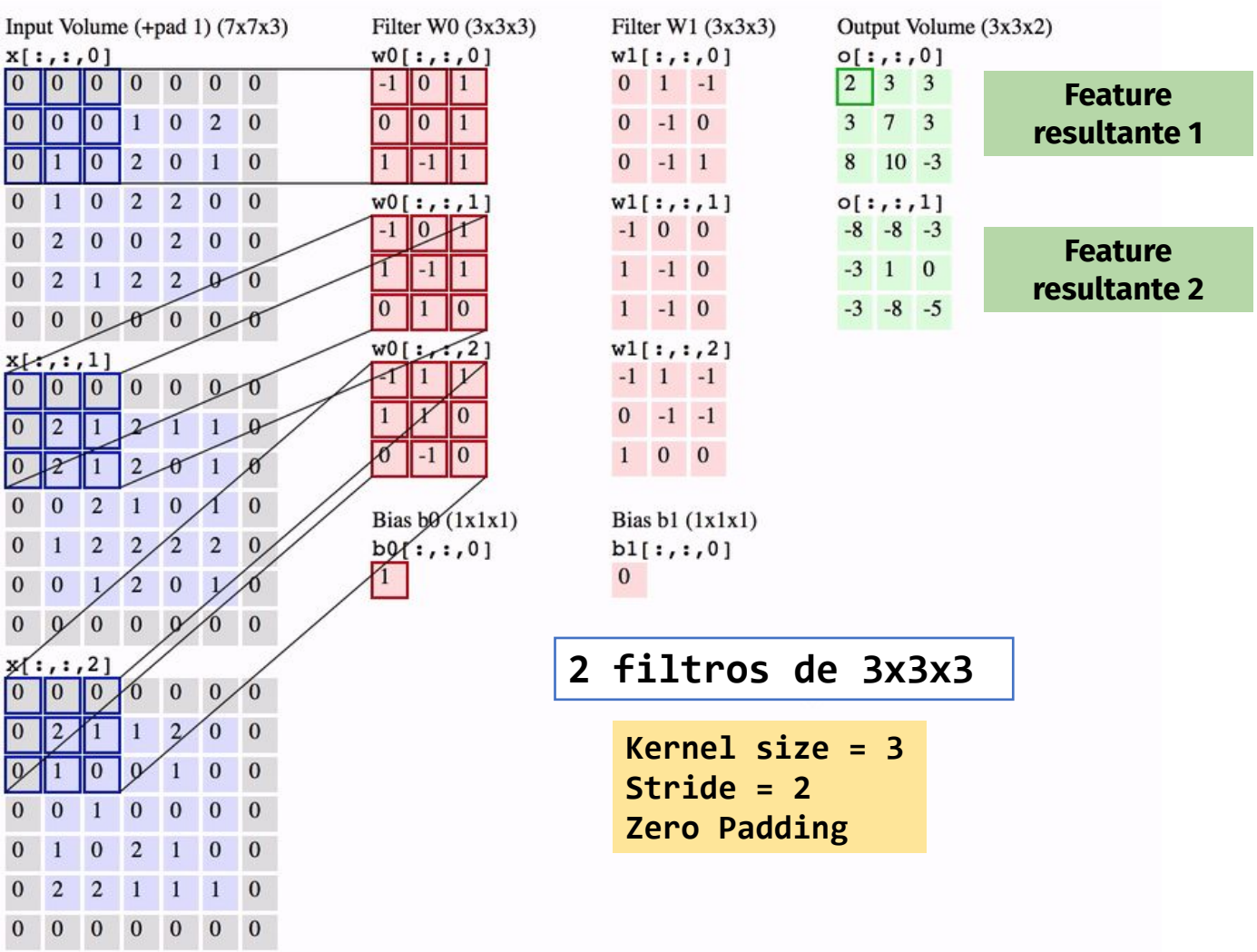
- Se realiza sobre los C canales a la vez
- Convierte una imagen HxWxC en una imagen HxWx1
  - Kernel_Size= KxKxC
- Ejemplo con C=4 y K=3
  - Pixel de salida depende de **todos** los canales



Convoluciónes 2D sobre varios canales

- Dados kernels W1 y W2
  - 2 kernels
  - 2 Convoluciones
- W1 genera **feature** 1
- W2 genera **feature** 2
- Entrada
  - **Feature map** con 3 canales
- Salida
  - **Feature map** con 2 canales




Input Volume (+pad 1) (7x7x3)
x[:,:,0]
0 0 0 0 0 0 0
0 0 0 1 0 2 0
0 1 0 2 0 1 0
0 1 0 2 2 0 0
0 2 0 0 2 0 0
0 2 1 2 2 0 0
0 0 0 0 0 0 0
x[:,:,1]
0 0 0 0 0 0 0
0 2 1 2 1 1 0
0 2 1 2 0 1 0
0 0 2 1 0 1 0
0 1 2 2 2 2 0
0 0 1 2 0 1 0
0 0 0 0 0 0 0
x[:,:,2]
0 0 0 0 0 0 0
0 2 1 1 2 0 0
0 1 0 0 1 0 0
0 0 1 0 0 0 0
0 1 0 2 1 0 0
0 2 2 1 1 1 0
0 0 0 0 0 0 0
Filter W0 (3x3x3)
w0[:,:,0]
-1 0 1
0 0 1
1 -1 1
w0[:,:,1]
-1 0 1
1 -1 1
0 1 0
w0[:,:,2]
-1 1 1
1 1 0
0 -1 0
Bias b0 (1x1x1)
b0[:,:,0]
1
Filter W1 (3x3x3)
w1[:,:,0]
0 1 -1
0 -1 0
0 -1 1
w1[:,:,1]
-1 0 0
1 -1 0
1 -1 0
w1[:,:,2]
-1 1 -1
0 -1 -1
1 0 0
Bias b1 (1x1x1)
b1[:,:,0]
0
Output Volume (3x3x2)
o[:,:,0]
2 3 3
3 7 3
8 10 -3
o[:,:,1]
-8 -8 -3
-3 1 0
-3 -8 -5
Feature resultante 1
Feature resultante 2
2 filtros de 3x3x3
Kernel size = 3
Stride = 2
Zero Padding