Filtros Convolucionales

Resumen Problemas de Clasificación

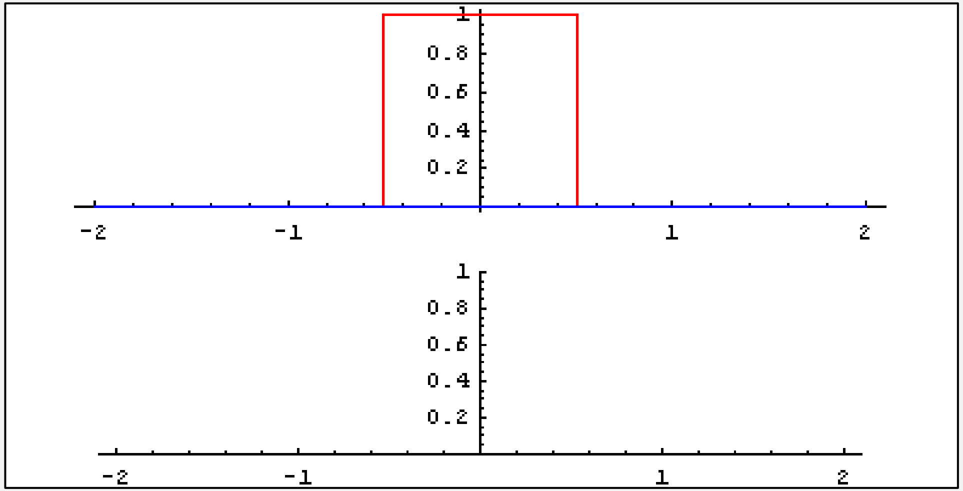
Hasta ahora...

- **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

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.



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

Convolución 1D discreta

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

x	5	0	2	-1	3	0	2
g	1	0	-1				
y							

Parámetros:

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

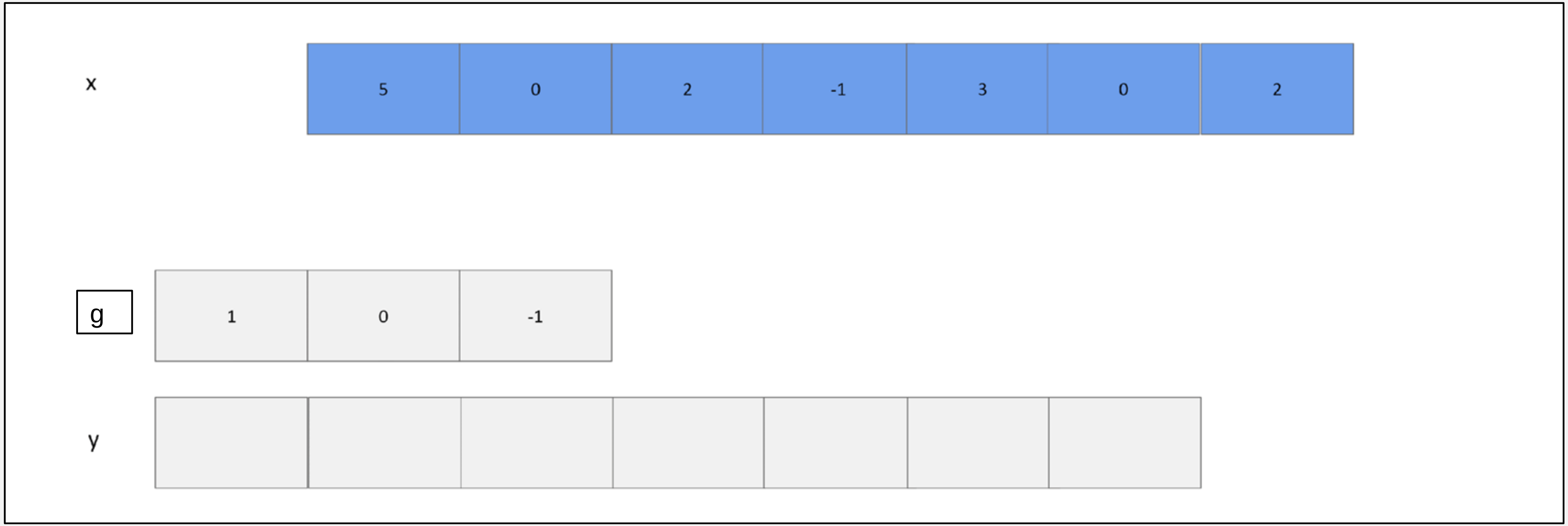
Stride: Es el número de saltos que da el filtro cada vez que se aplica. En este caso = 1.

Convolución - Padding

Aplicar el filtro de forma discreta ocasiona dos problemas:

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

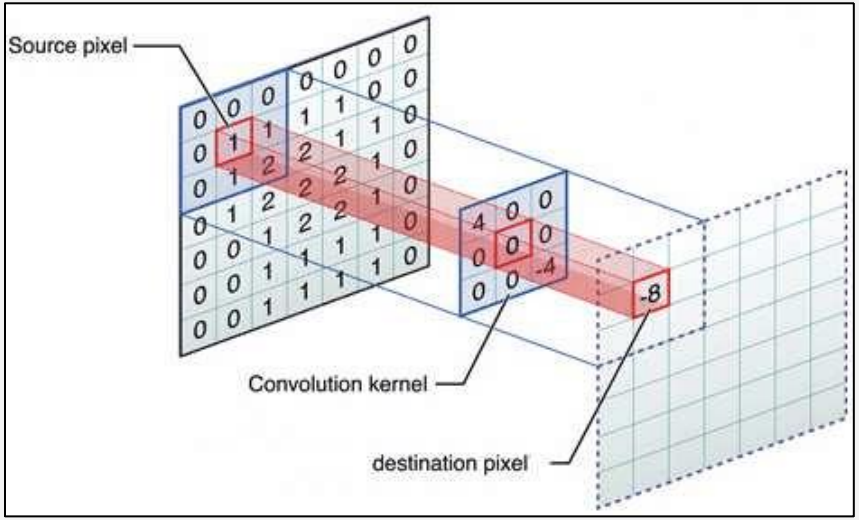
Para solucionar esto se suele utilizar la técnica de “padding”, generalmente rellenando con ceros.



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.



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

Convolución 2D

Kernel Matrix

0	-1	0
-1	5	-1
0	-1	0

105	102	100	97	96
103	99	103	101	102
101	98	104	102	100
99	101	106	104	99
104	104	104	100	98

Image Matrix

```
Kernel size= 3
Stride =1
```

	89			

Output Matrix

$$
\begin{aligned}
&105 * 0 + 102 * -1 + 100 * 0 \\
&+103 * -1 + 99 * 5 + 103 * -1 \\
&+101 * 0 + 98 * -1 + 104 * 0 \;=\; 89
\end{aligned}
$$

Convolución 2D

Kernel Matrix

0	-1	0
-1	5	-1
0	-1	0

Image Matrix

105	102	100	97	96
103	99	103	101	102
101	98	104	102	100
99	101	106	104	99
104	104	104	100	98

```
Kernel size= 3
Stride =1
```

Output Matrix

	89	111		

$$102 * 0 + 100 * -1 + 97 * 0$$
$$+99 * -1 + 103 * 5 + 101 * -1$$
$$+98 * 0 + 104 * -1 + 102 * 0 = 111$$

Convolución 2D

Para obtener una imagen resultante del mismo tamaño que la original se utiliza "padding" 2D.

0	0	0	0	0	0	0
0	60	113	56	139	85	0
0	73	121	54	84	128	0
0	131	99	70	129	127	0
0	80	57	115	69	134	0
0	104	126	123	95	130	0
0	0	0	0	0	0	0

Kernel

0	-1	0
-1	5	-1
0	-1	0

114				

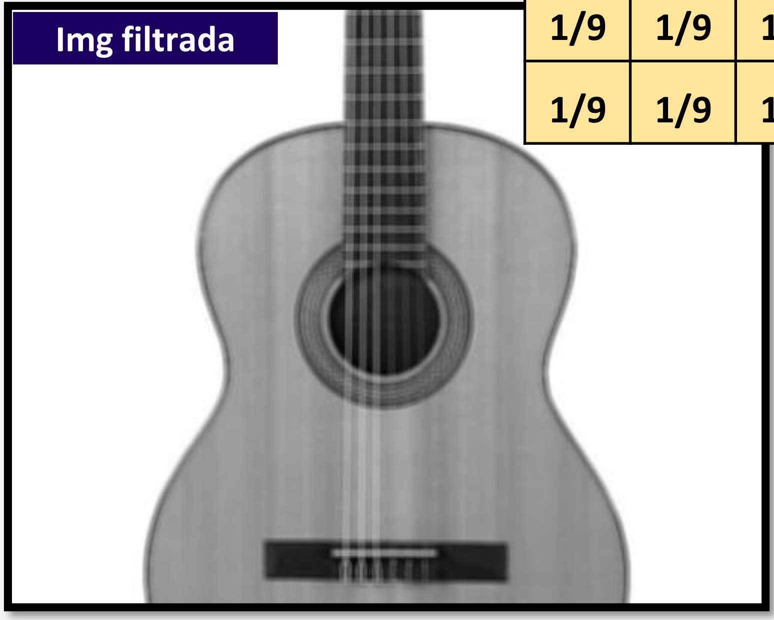
Kernel size= 3
Stride =1
Zero Padding

Convolución 2D

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

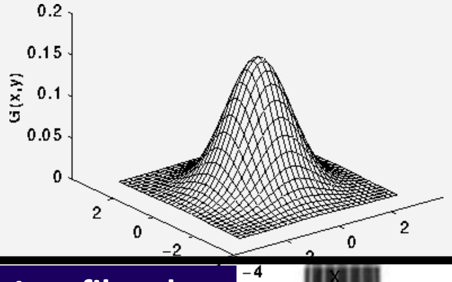
Simple promedio de todos los píxeles: borronea la imagen.

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

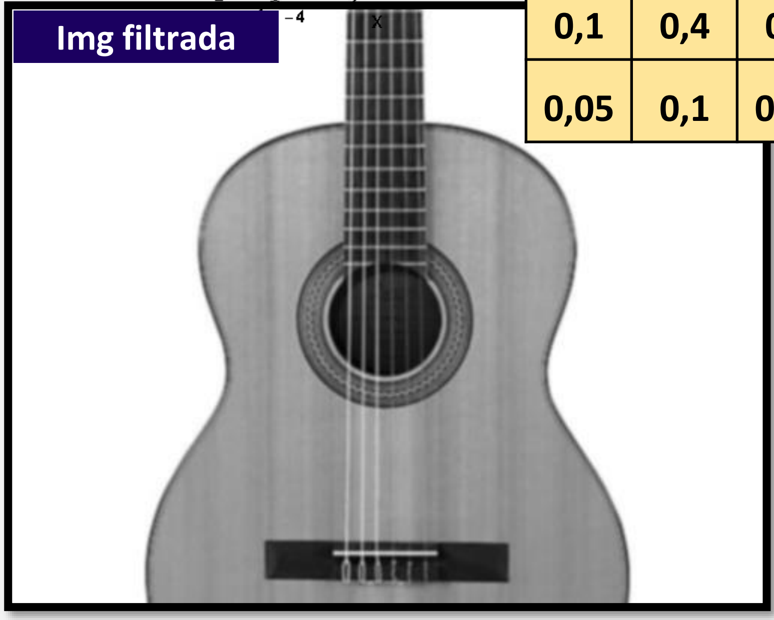


Convolución 2D

Filtro gaussiano
(filtro de pasa baja)

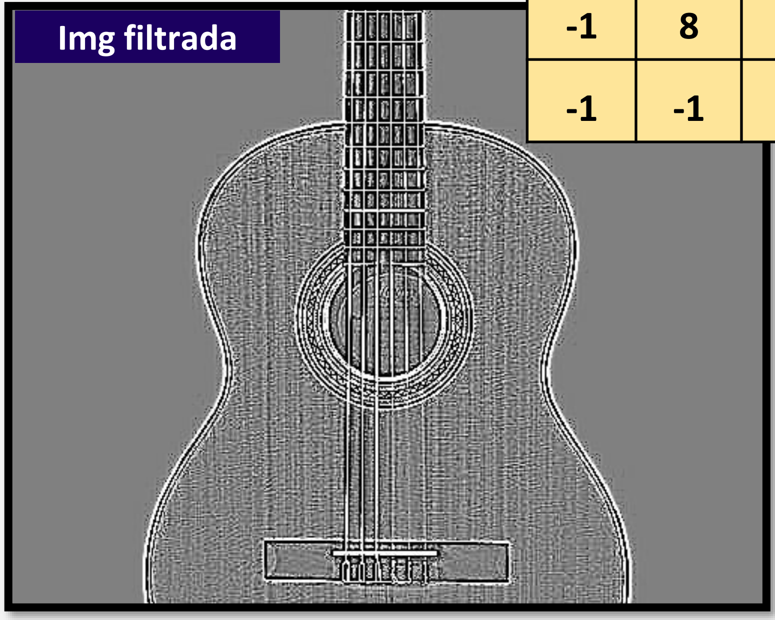


0,05	0,1	0,05
0,1	0,4	0,1
0,05	0,1	0,05



Convolución 2D

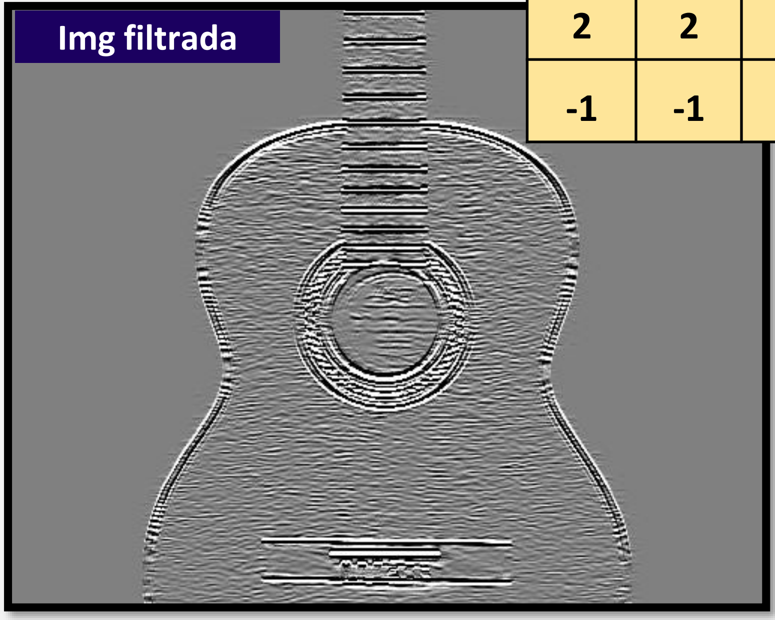
Detección de bordes (filtro de pasa alta)



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

Convolución 2D

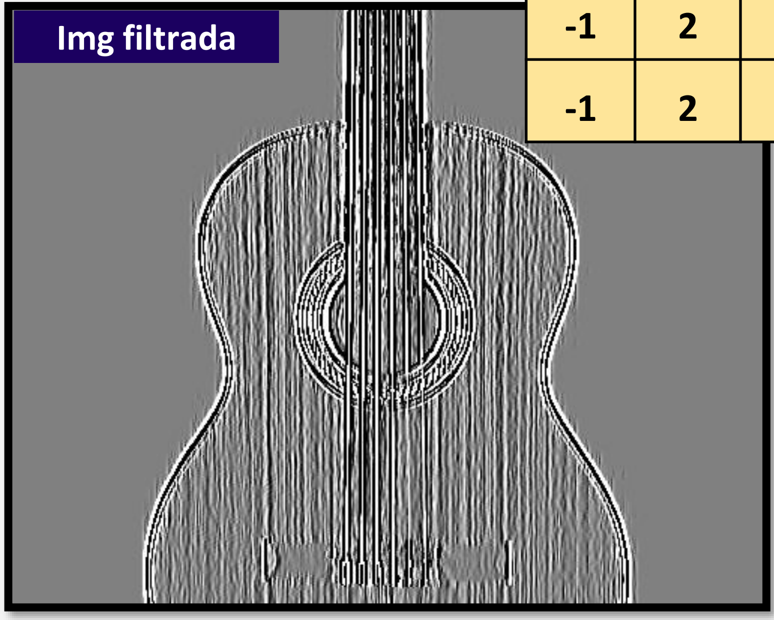
Bordes horizontales



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

Convolución 2D

Bordes verticales



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

Convolución 2D

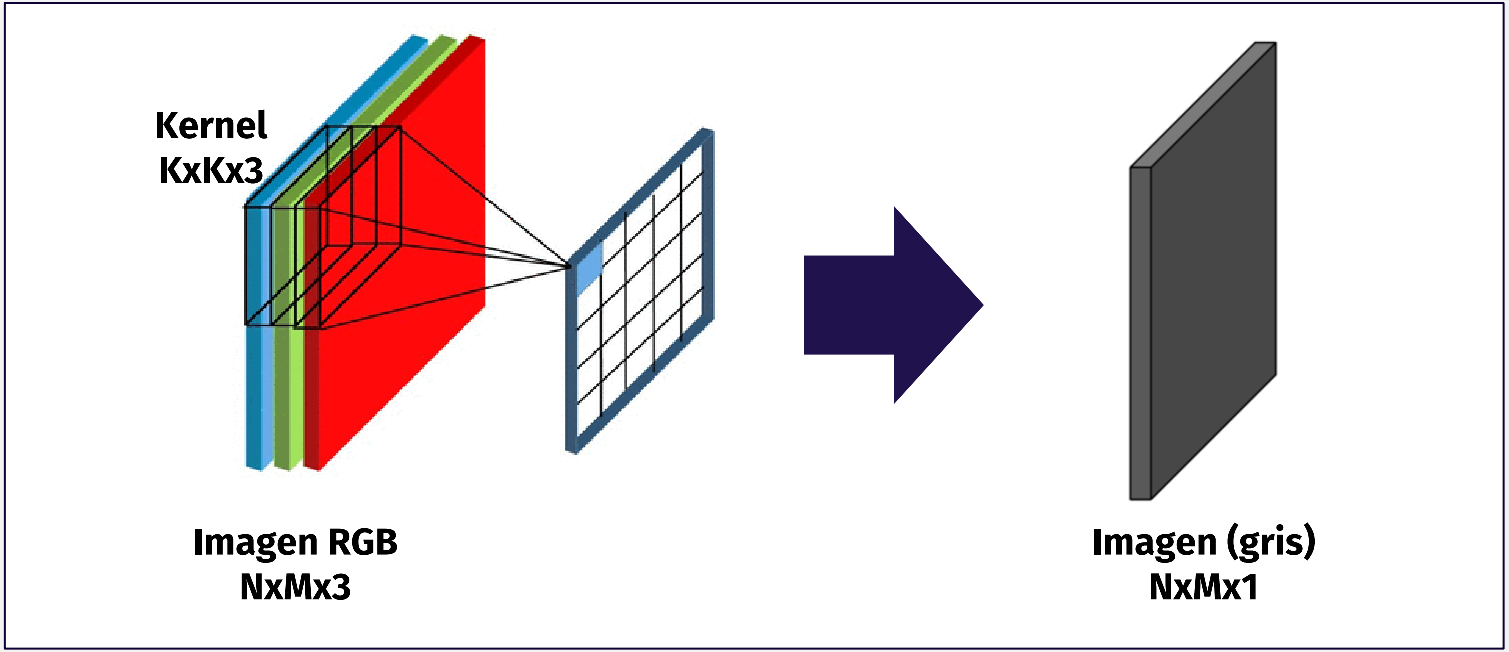
Kernel Gaussiano + Kernel Bordes

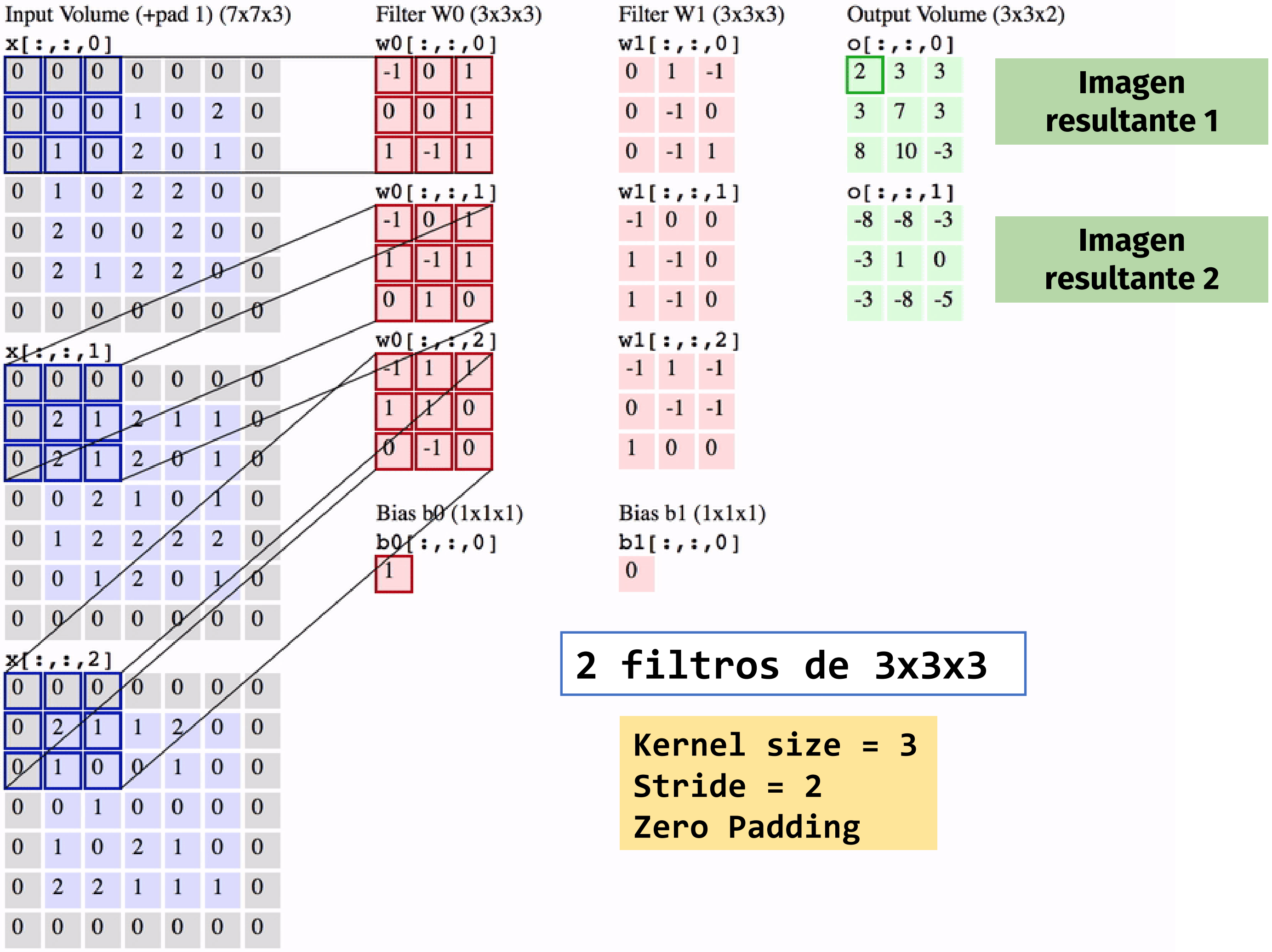
Convolución 2D sobre datos 3D - ND

En imágenes RGB, nuestro kernel deberá tener una dimensión más:

Kernel_Size= KxKx3

La convolución sigue siendo 2D pero se realiza sobre los 3 canales a la vez.





2 filtros de 3x3x3

Kernel size = 3
Stride = 2
Zero Padding