

# Redes Convolucionales

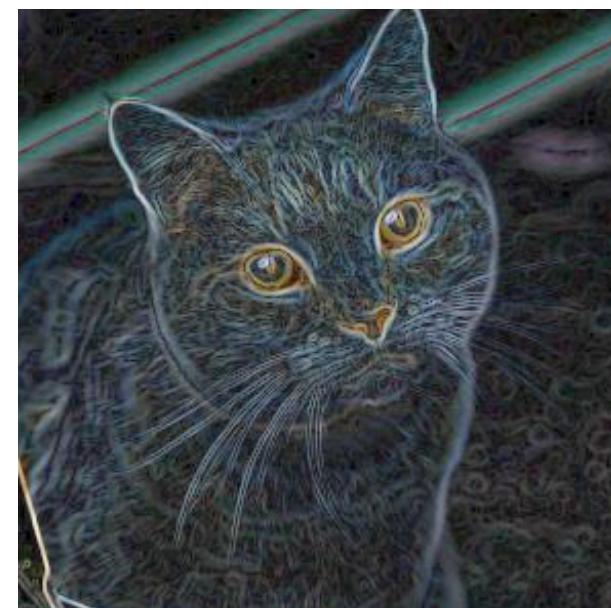
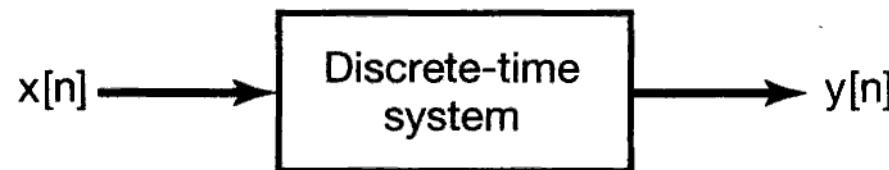
Capas Convolucionales vía Convoluciones



Prof. Ricardo Ñanculef - Departamento de Informática UTFSM 2022

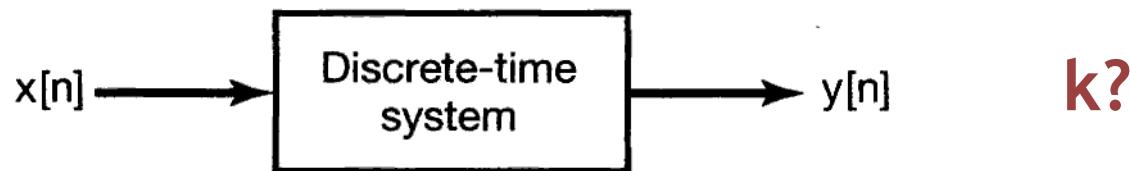
# Sistemas

- Es posible modelar una señal  $y$  como la acción de un cierto **sistema desconocido** sobre una señal de entrada  $x$ . Por ejemplo:



# Sistemas

- Es posible modelar una señal  $y$  como la acción de un cierto **sistema desconocido** sobre una señal de entrada  $x$ . Por ejemplo:



# Representación de Sistemas LTI

- Sea  $h_0[n]$  la respuesta de un sistema LTI ante un impulso unitario. Entonces la acción del sistema sobre cualquier señal  $x$  se obtiene como la convolución de  $x$  con  $h_0[n]$ :

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h_0[n-k] \quad \forall n$$

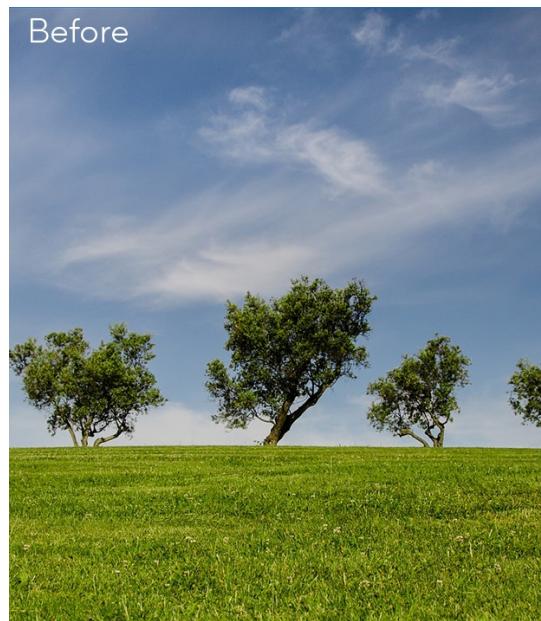
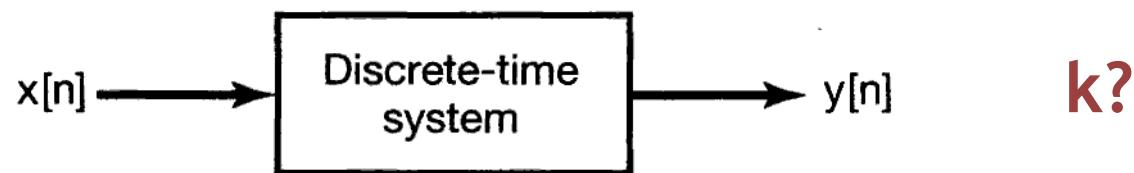
En lo que sigue, llamaremos a esta función **el kernel** del sistema  $k[n] = h_0[n]$ .

**La consecuencia de este resultado es que podemos “aprender” el sistema sólo aprendiendo su kernel.**



# Representación de Sistemas LTI

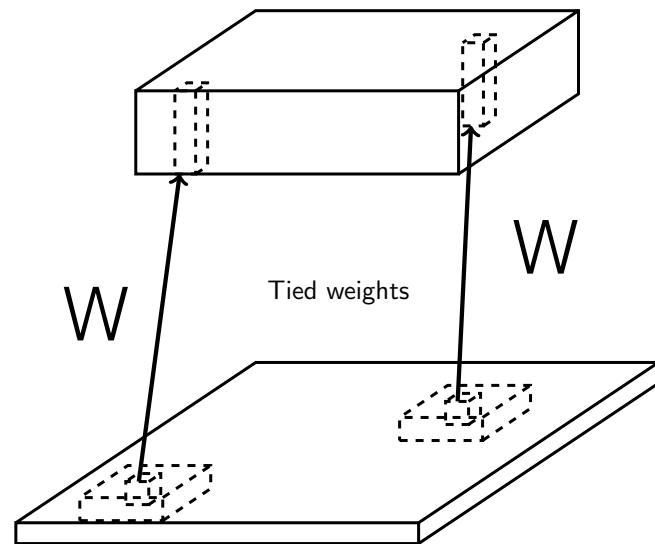
- Aprendiendo el kernel aprendemos la acción del sistema sobre cualquier señal:



# Capas Convolucionales

- Dada una señal  $x$  una capa convolucional es una capa compuesta de **O** canales, capa uno de los cuales computa como salida:

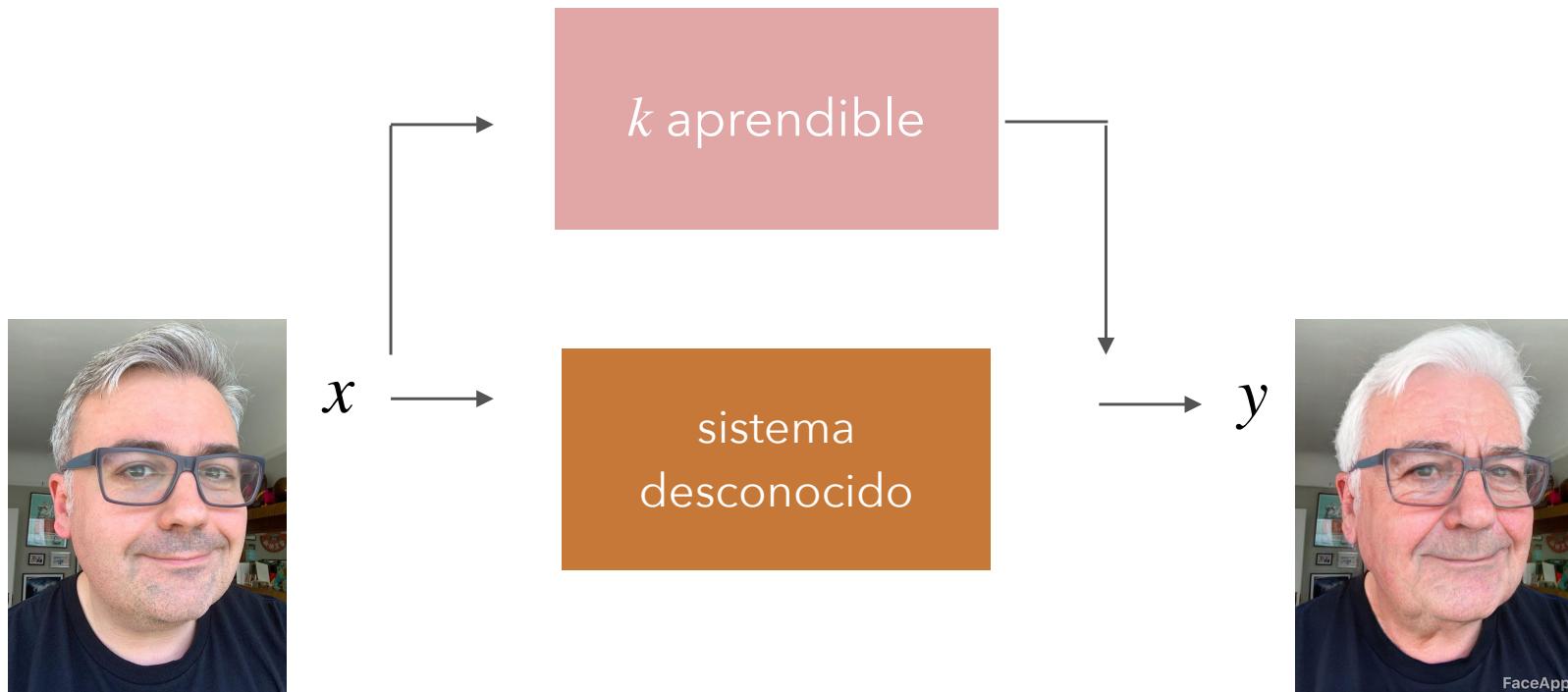
$$y = g(w * x - b)$$



- La señal  $w$  se denomina el filtro o el kernel correspondiente al canal y es aprendible desde datos.

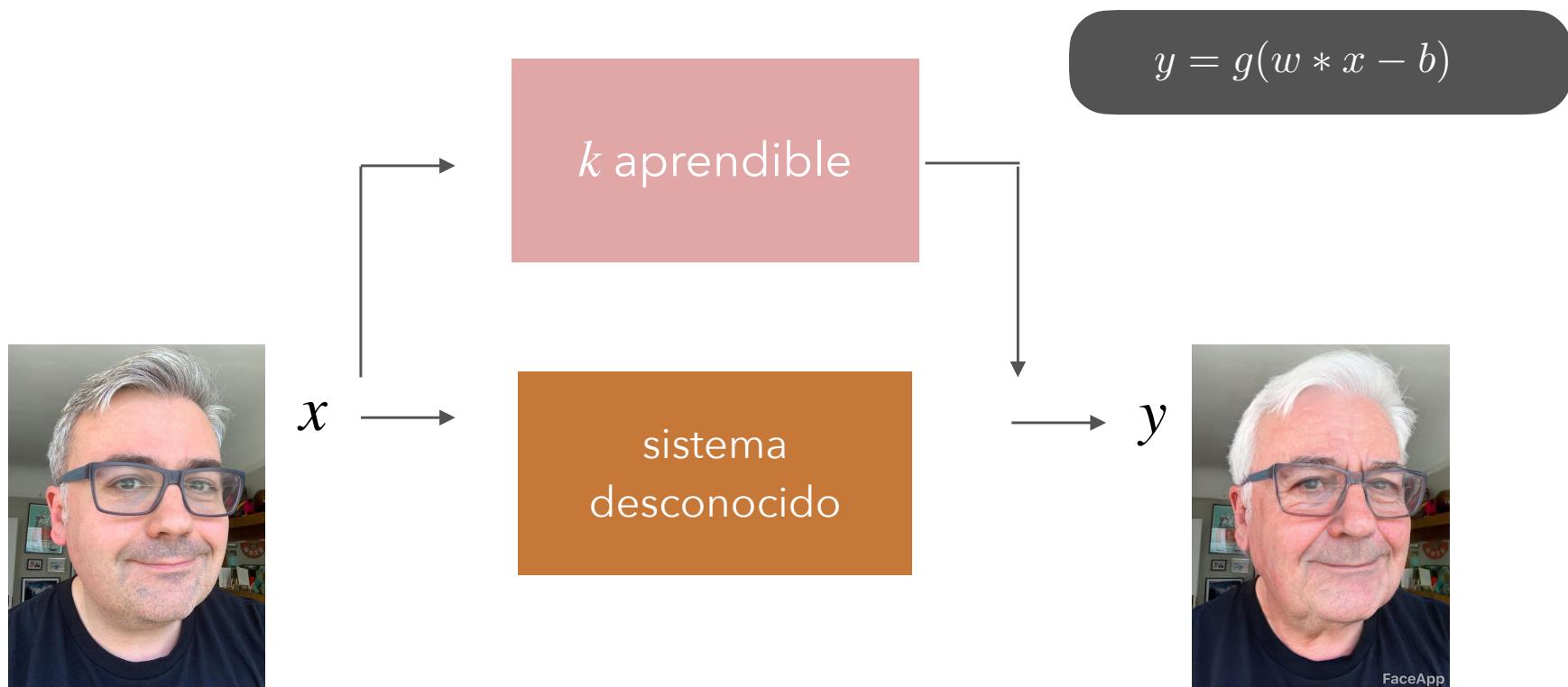
# Capas Convolucionales

- La señal  $w$  se denomina el filtro o el kernel de la capa y es aprendible desde datos.



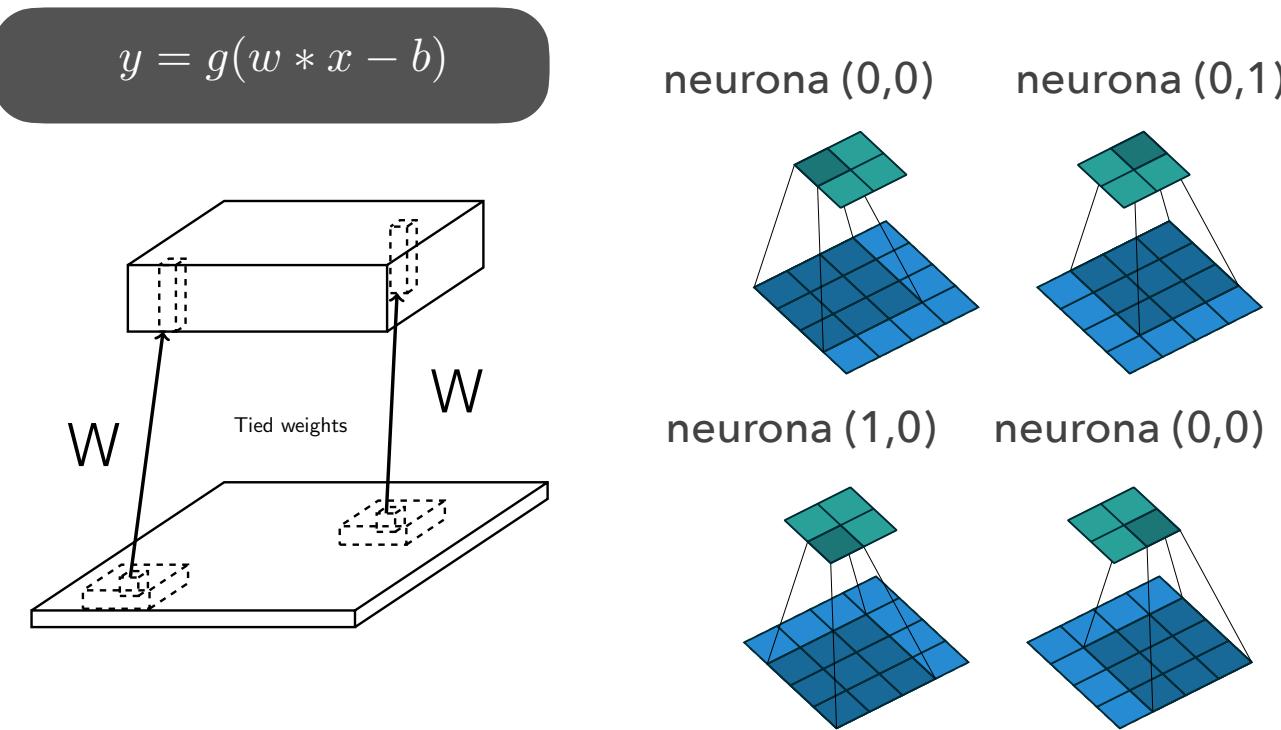
# Capas Convolucionales

- La inclusión de la no-linealidad y la composición en profundidad de varias capas convolucionales nos permitirá aproximar sistemas más complejos.



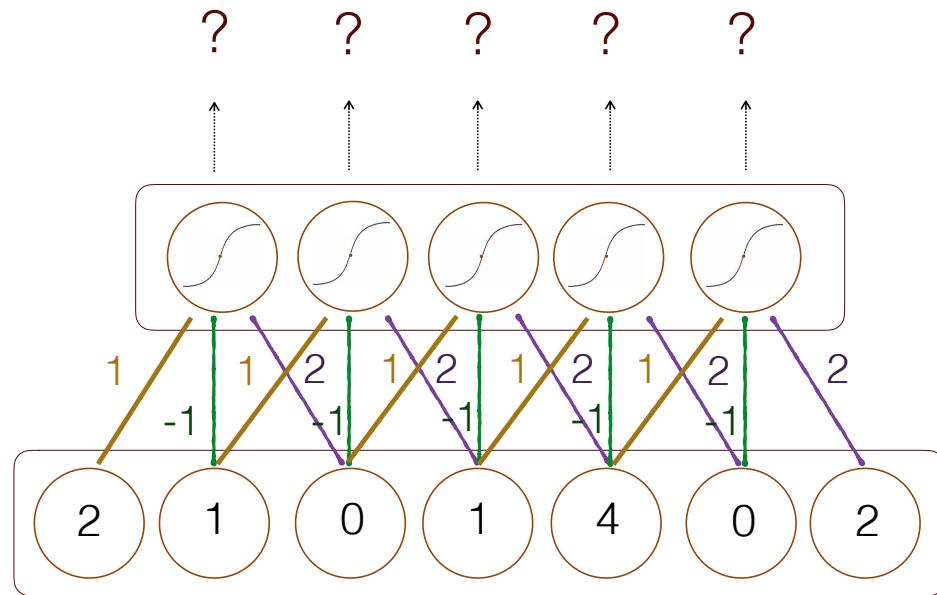
# Capas Convolucionales

- La descripción informal que habíamos hecho con la idea de los pesos compartidos y los campos receptivos coincide “casi” plenamente con la que acabamos de definir.



# FC vía Convolución (Caso 1D)

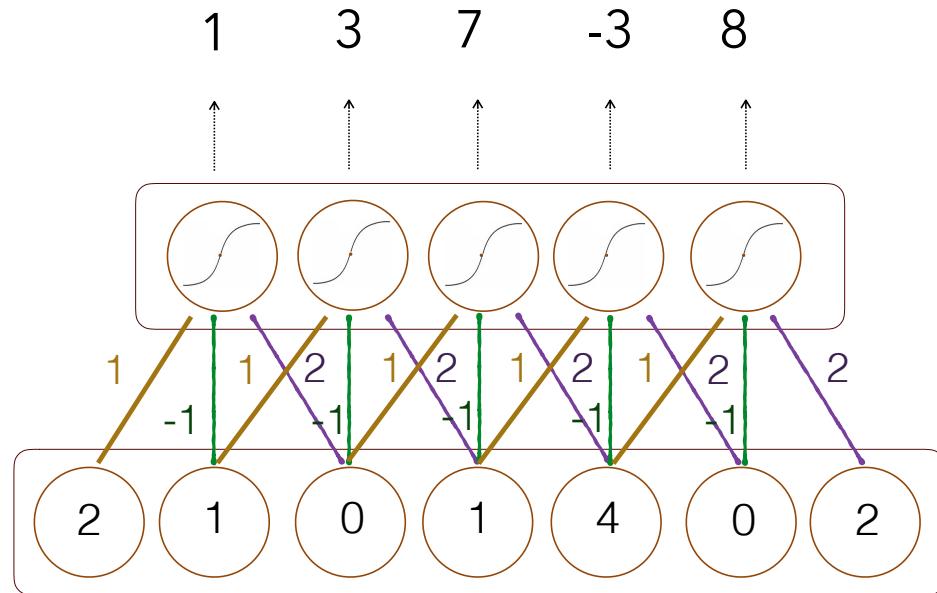
- Recordemos este ejercicio de la sección anterior:



(\*) Asumimos umbrales de activación 0 y funciones de activación lineales.

# FC vía Convolución (Caso 1D)

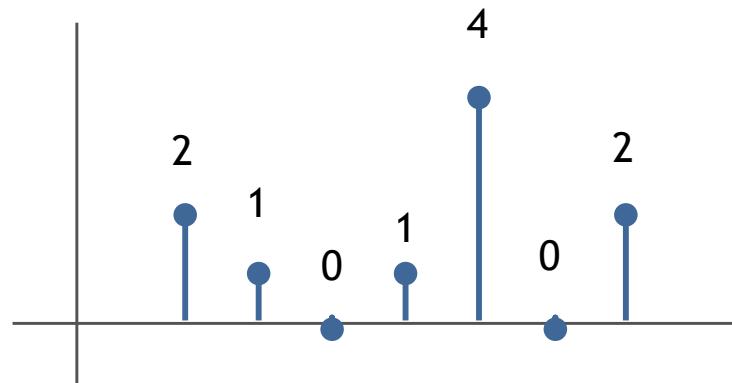
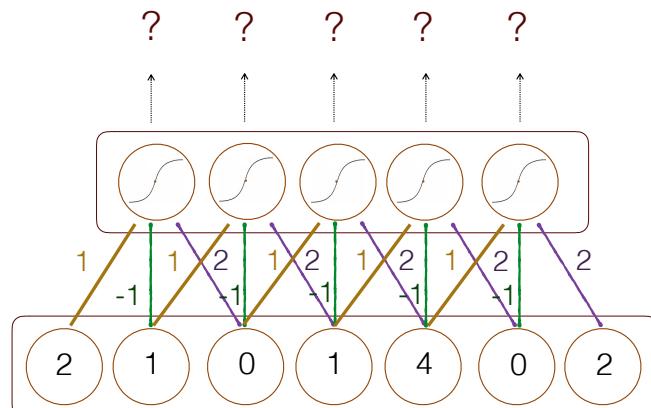
- Recordemos este ejercicio de la sección anterior:



(\*) Asumimos umbrales de activación 0 y funciones de activación lineales.

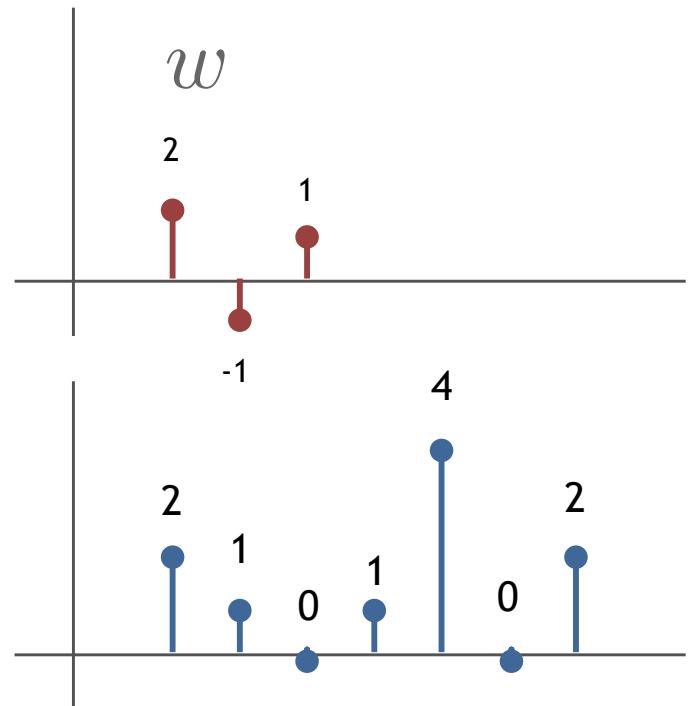
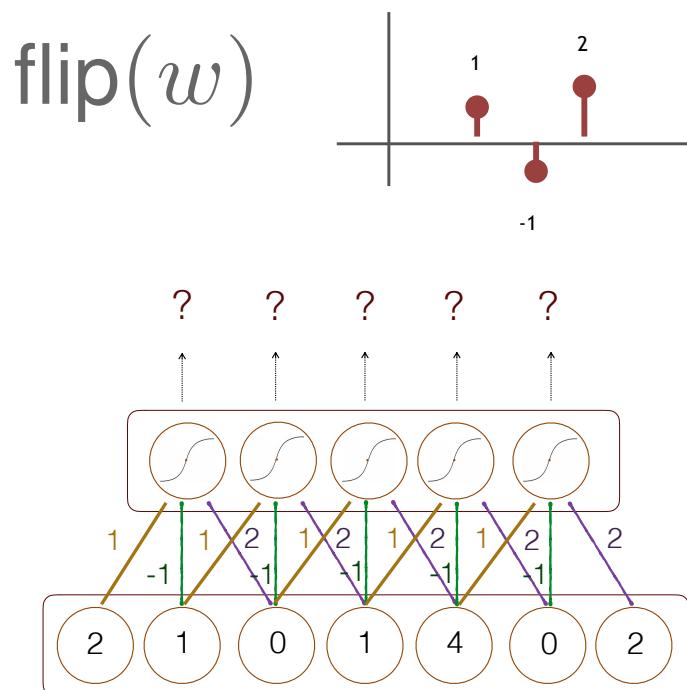
# FC vía Convolución (Caso 1D)

- Ahora estamos en condiciones de ver el patrón de entrada como una señal unidimensional.

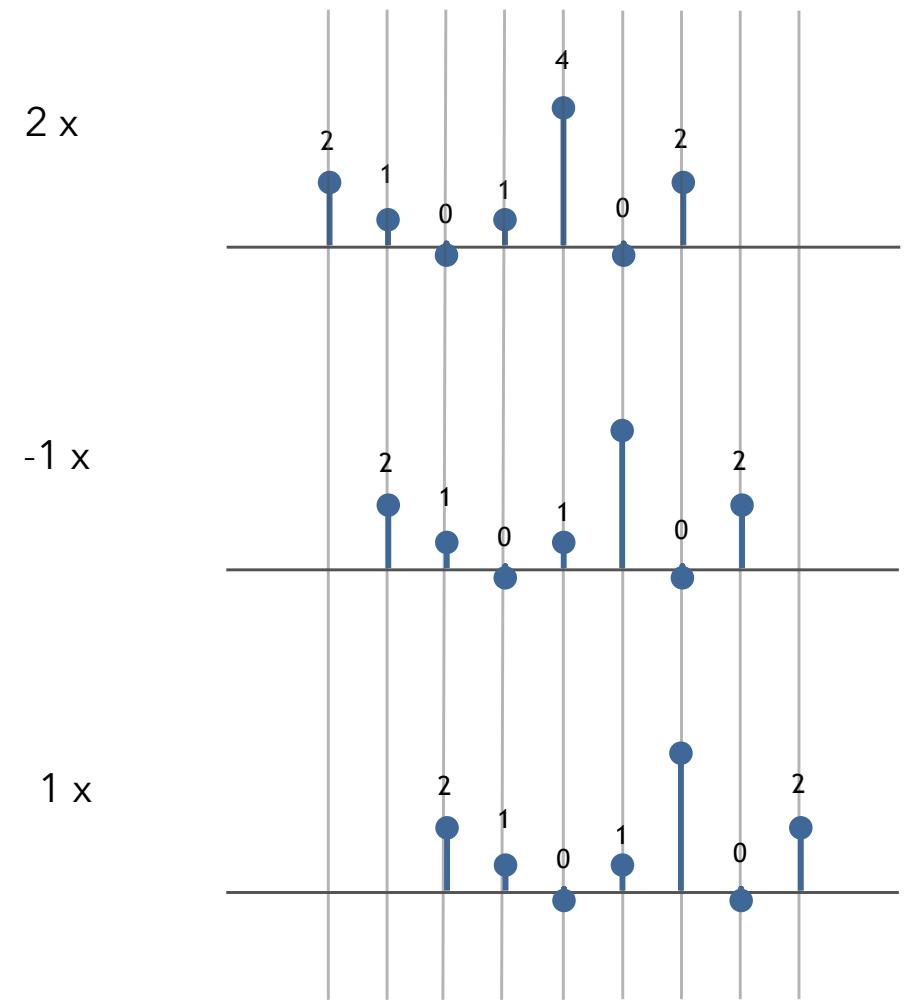
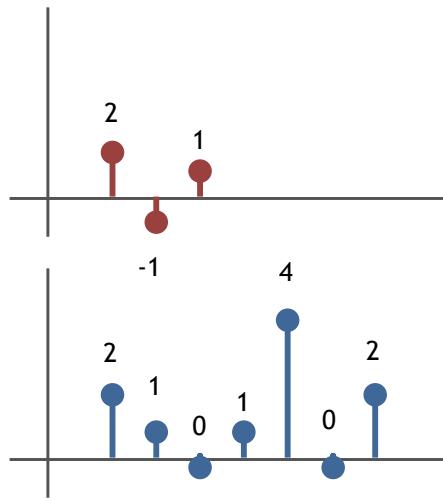


# FC vía Convolución (Caso 1D)

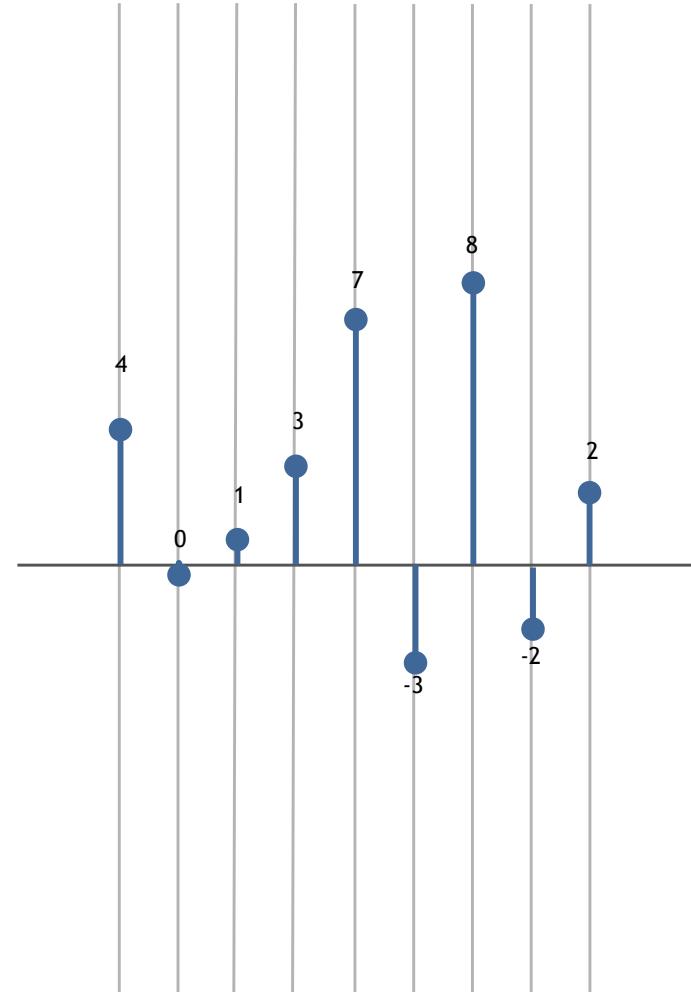
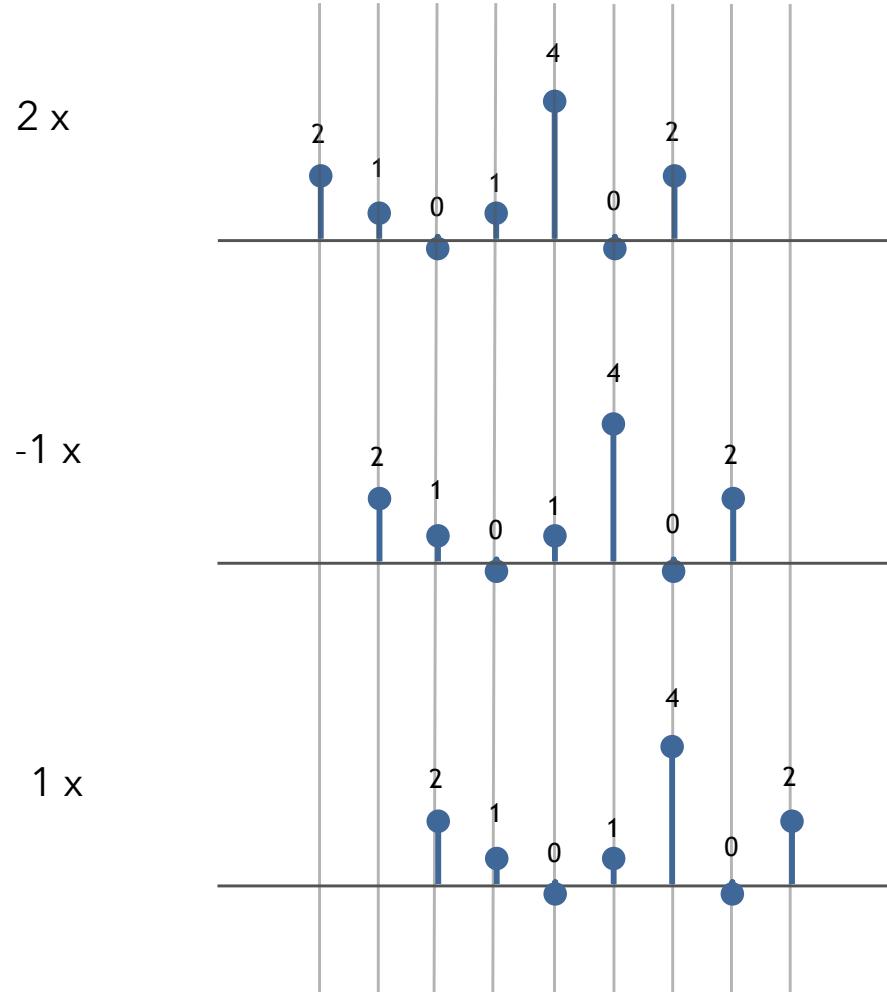
- Si nuestro modelo es correcto, los pesos de compartidos deben corresponder al kernel de la convolución. Es así?
- Recordemos que para calcular  $x * w$  vía correlaciones (como lo hacíamos antes) primero debemos voltear  $w$ .



# FC vía Convolución (Caso 1D)

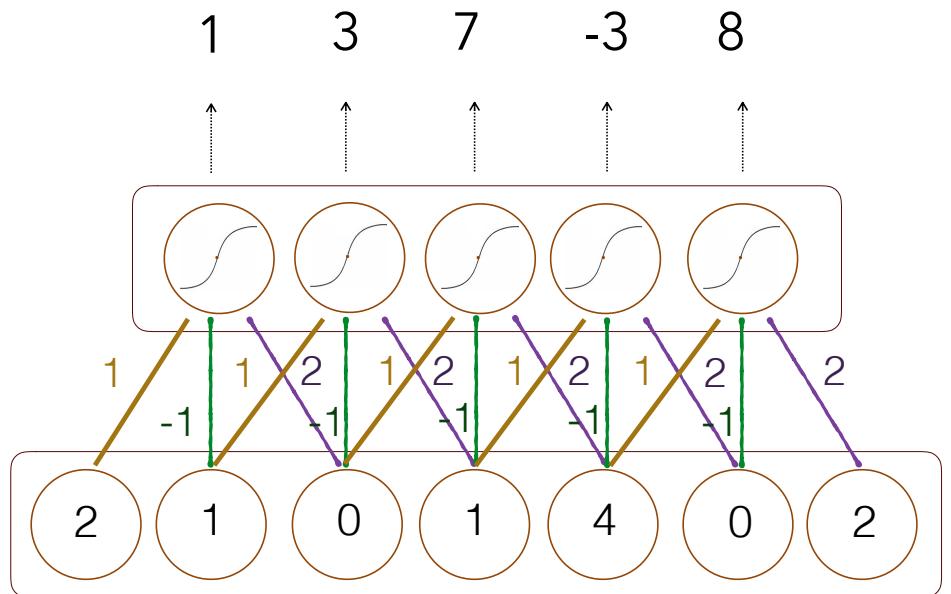
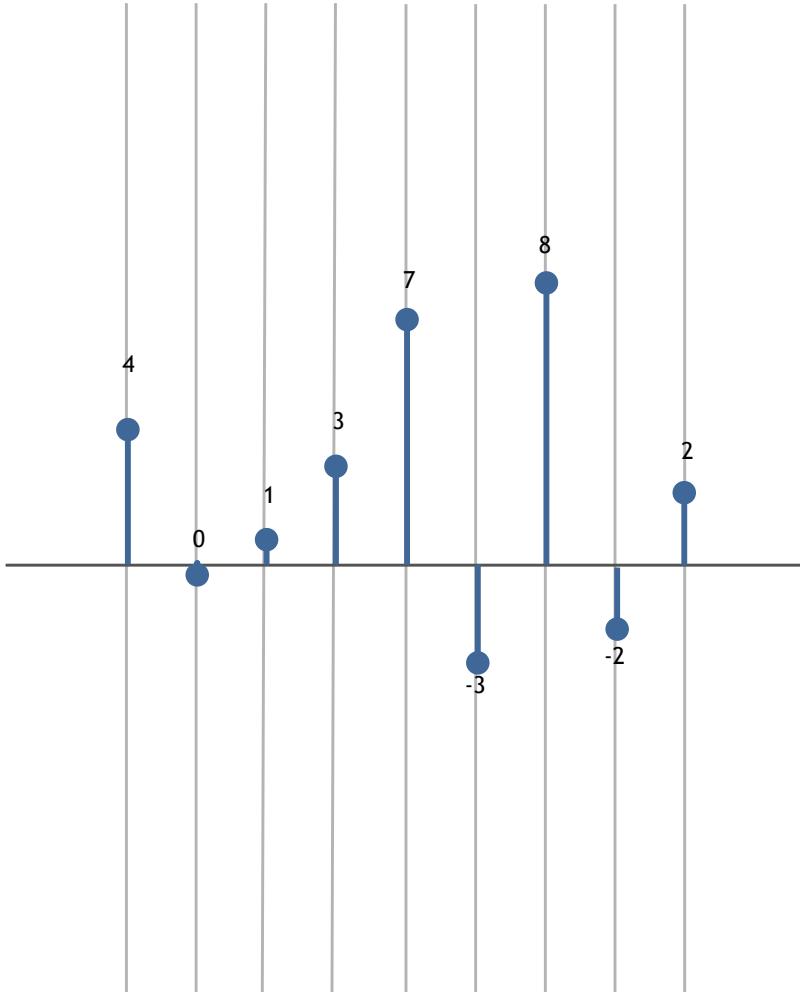


# FC vía Convolución (Caso 1D)



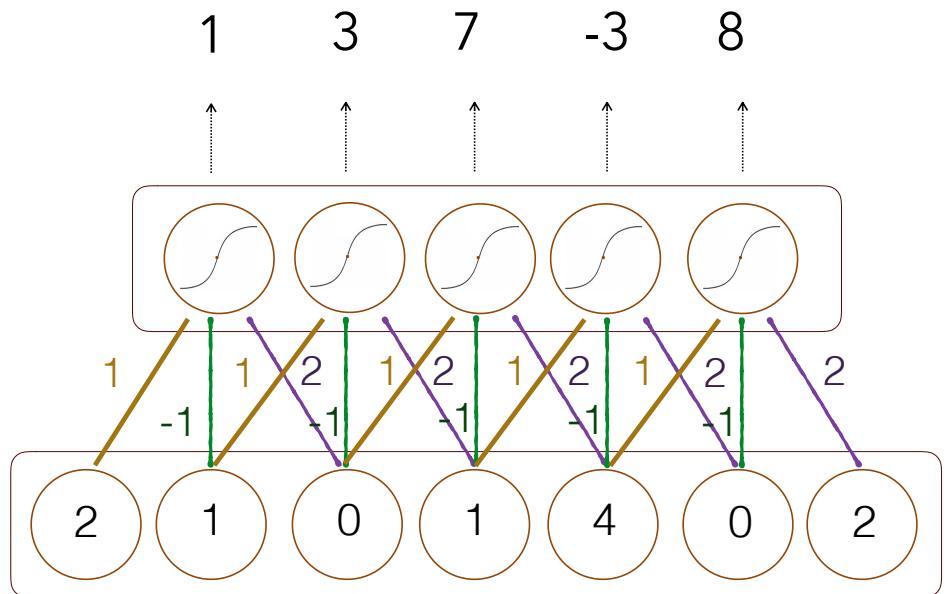
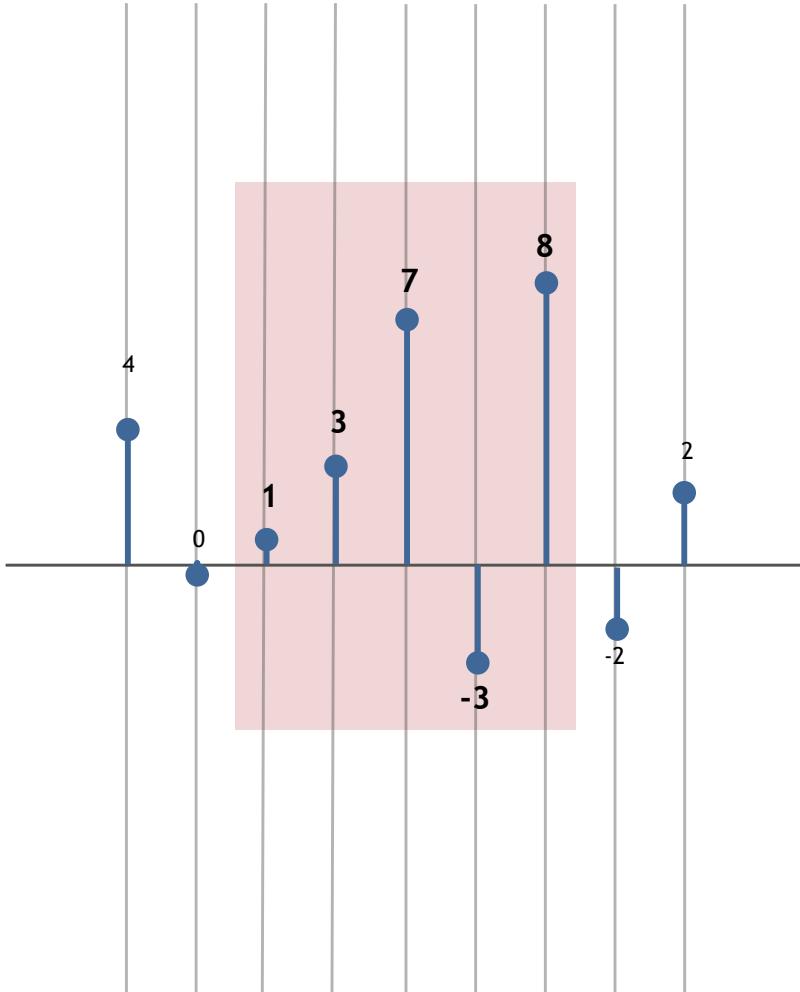
# FC vía Convolución (Caso 1D)

- El resultado es “casi” el que habíamos hecho



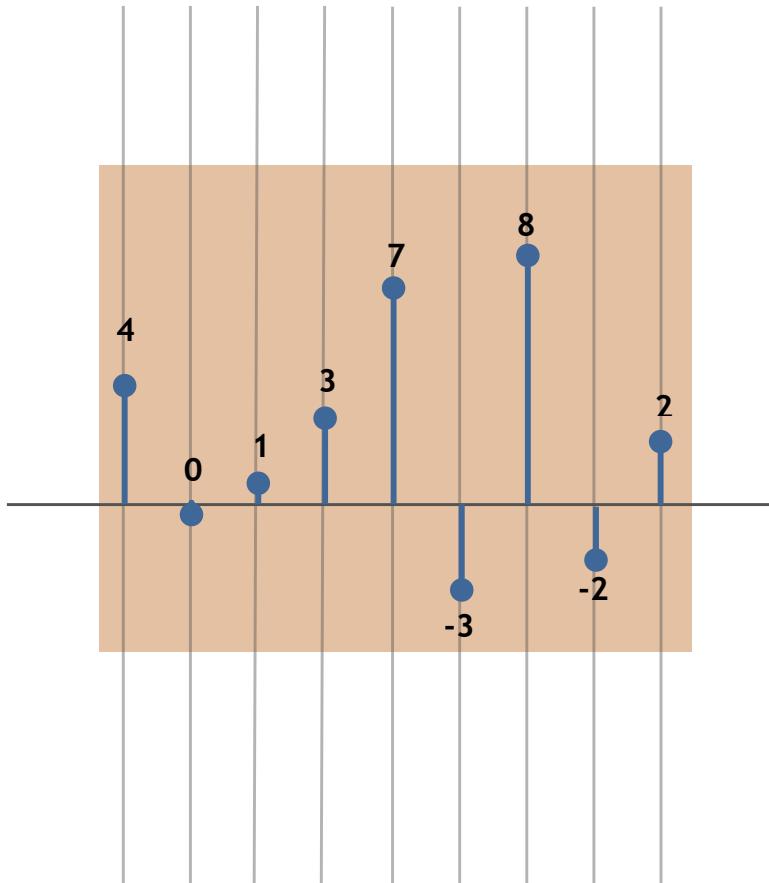
# FC vía Convolución (Caso 1D)

- El resultado es “casi” el que habíamos hecho

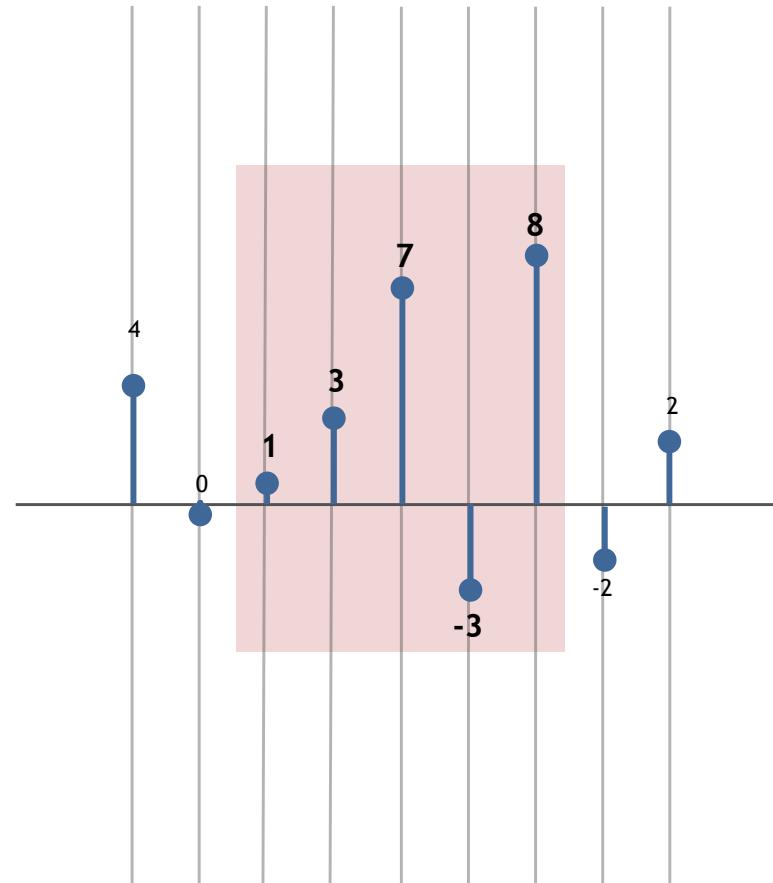


# Convolución Completa versus Válida

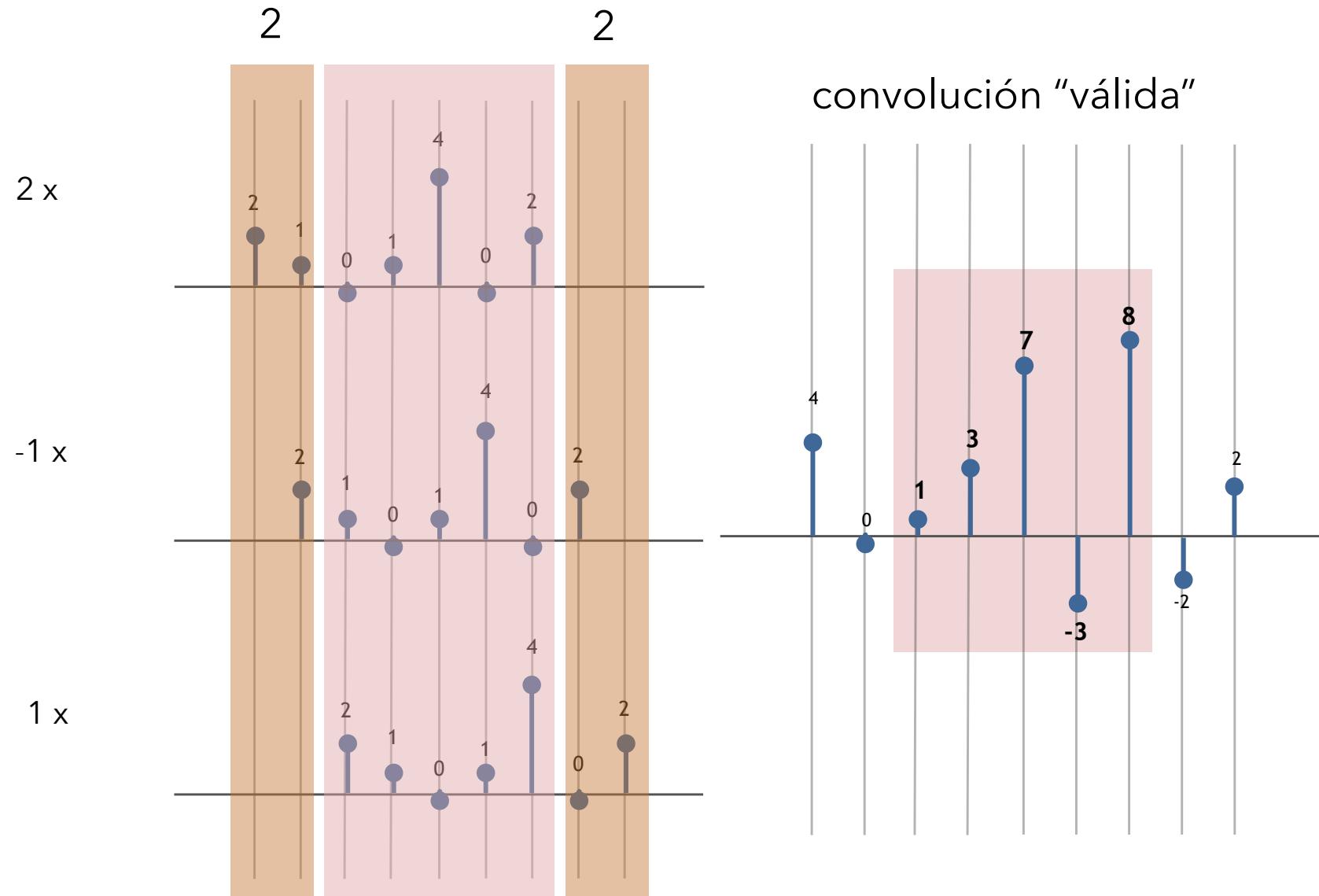
convolución completa



convolución "válida"

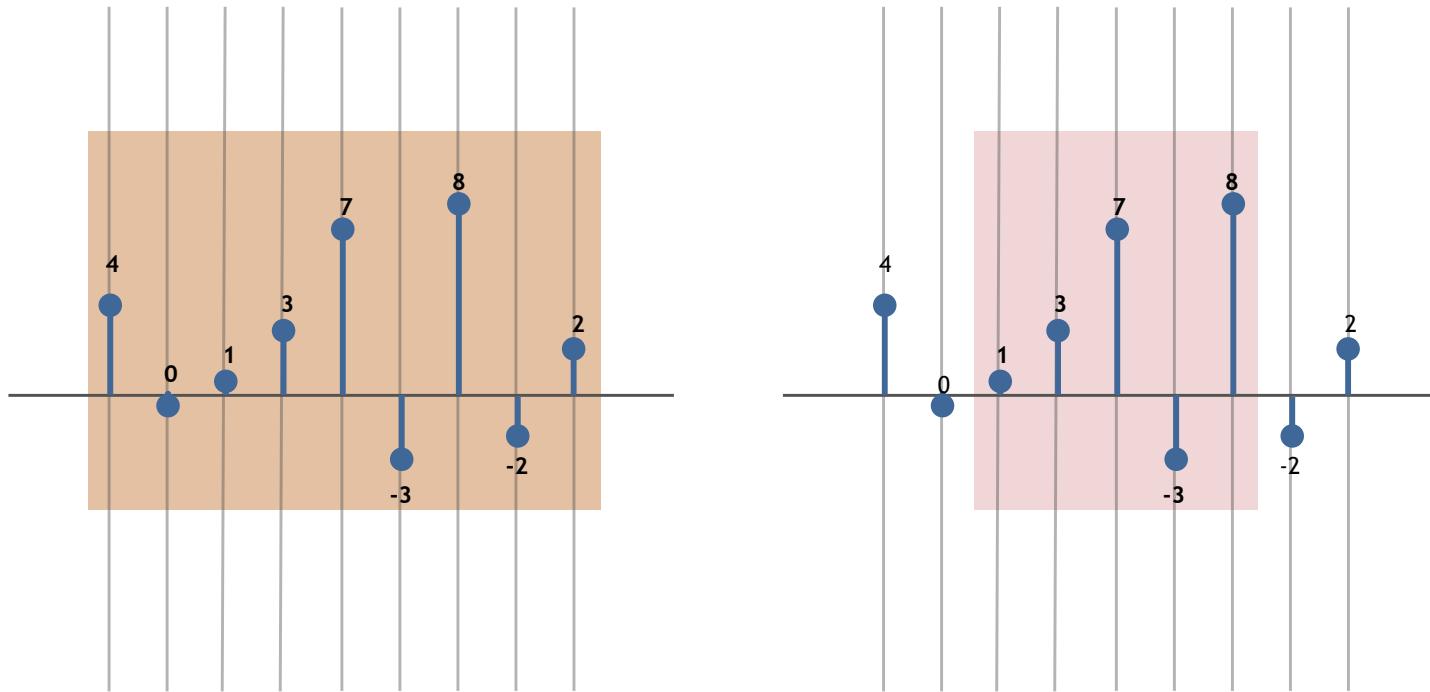


# Convolución Completa versus Válida



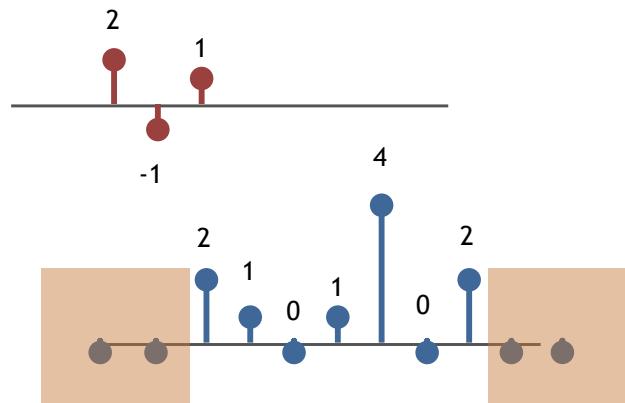
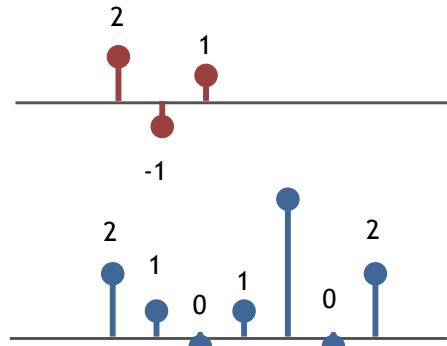
# Convolución Completa versus Válida

Si el kernel (filtro) de la convolución es de tamaño  $S$ , **una convolución “válida” se obtiene removiendo de la convolución completa  $S-1$  valores a inicio y  $S-1$  al final** (y luego adelantando la señal  $S-1$  unidades de tiempo).



# Convolución Completa versus Válida

- **Es posible también computar una convolución completa a partir de una convolución válida** usando padding, una operación que "aumenta" nuestro arreglo de datos rellenando con P zeros.



- Concretamente, necesitamos padding de  $P=S-1$  zeros al incio y al final de la señal que se convoluciona con el kernel (formalmente se requiere también un retraso temporal de  $P=S-1$  unidades para obtener el resultado en tiempos consistentes con la definición).

# Efecto del Padding

- Si el tamaño del kernel es  $S$  y el tamaño del input es  $I$ , el resultado de la convolución válida es siempre de tamaño  $I-S+1$ . El resultado de la convolución completa en cambio es de tamaño  $I+S-1$ .



- Es común en redes neuronales, elegir el tamaño del resultado de manera de preservar el tamaño de entrada (es decir, no se usa ni una convolución completa ni una válida).

# Efecto del Padding

- El resultado de la convolución completa en cambio es de tamaño  **$I+S-1$ .**

$$\begin{matrix} \text{Red Box} & S & * & \text{Green Box} & I \\ & & & & \\ & = & & \text{Blue Box} & I+S-1 \end{matrix}$$

- Las cuentas calzan si pensamos la convolución completa como una válida aplicada al input con padding.

$$\begin{matrix} \text{Red Box} & S & * & \text{Light Green Box} & \text{Dark Green Box} & \text{Light Green Box} & I+2P \\ & & & | & & & \\ & = & & \text{Blue Box} & & & (I+2P)-S+1 \end{matrix}$$

# Efecto del Padding

- Es común en redes neuronales, elegir el tamaño del resultado de manera de preservar el tamaño de entrada (es decir, no se usa ni una convolución completa ni una válida).

$$\begin{matrix} \text{S} & * & \text{I} \end{matrix}$$

$$= \begin{matrix} \text{I} \end{matrix}$$

- El tamaño de salida se puede siempre controlar via padding.

$$\begin{matrix} \text{S} & * & \text{I} + 2\text{P} \end{matrix}$$

# Convolución Válida

- La convolución válida se puede escribir como la siguiente operación

$$\tilde{w} \star x[m] = \text{flip}(w) \star x[m] = \sum_r x[m+r] \tilde{w}[r]$$

- En el caso 2D:

$$\tilde{w} \star x[m, n] = \text{flip}(w) \star x[m, n] = \sum_{r,s} \tilde{w}[r, s] x[m+r, n+s]$$

# Convolución Completa

- Recordemos en cambio que (caso 1D)

$$\begin{aligned} w * x[i] &= x * w[i] = \sum_{m=-\infty}^{\infty} x[m]w[i-m] \\ &= \sum_{m+r=i}^{\infty} x[m]w[r] \end{aligned}$$

- Y en el caso 2D:

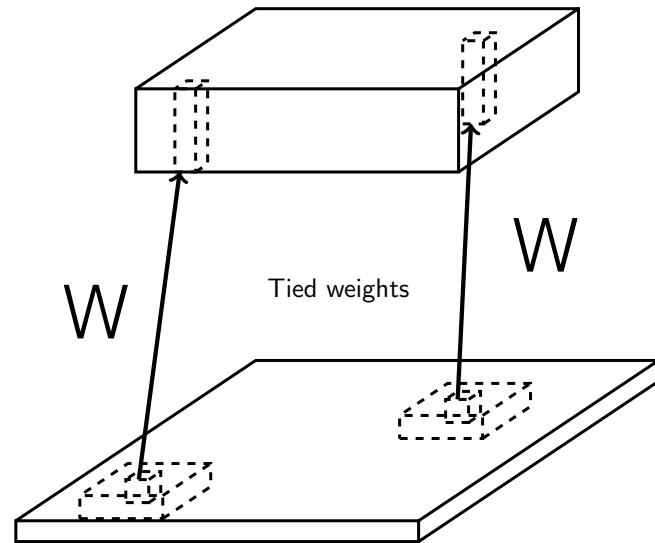
$$\begin{aligned} w * x[i, j] &= x * w[i, j] = \sum_{m,n=-\infty}^{\infty} x[m, n]w[i - m, j - n] \\ &= \sum_{m+r=i, n+s=j}^{\infty} x[m, n]w[r, s] \end{aligned}$$



# Capas Convolucionales

- Dada una señal  $x$  una capa convolucional es una capa compuesta de **O** canales, capa uno de los cuales computa como salida:

$$y = g(w * x - b)$$



- La señal  $w$  se denomina el filtro o el kernel correspondiente al canal y es aprendible desde datos.
- Notemos que la convolución matemática usa stride 1 mientras que nuestra capa puede modificar ese parámetro.

# Stride >1?

- Los aspectos matemáticos de una convolución con stride > 1 se han estudiado sólo recientemente.

## Take it in your stride: Do we need striding in CNNs?

Chen Kong                    Simon Lucey  
Carnegie Mellon University  
`{chenk, slucey}@andrew.cmu.edu`

### Abstract

Since their inception, CNNs have utilized some type of striding operator to reduce the overlap of receptive fields and spatial dimensions. Although having clear heuristic motivations (i.e. lowering the number of parameters to learn) the mathematical role of striding within CNN learning remains unclear. This paper offers a novel and mathematical rigorous perspective on the role of the striding operator within modern CNNs. Specifically, we demonstrate theoretically that one can always represent a CNN that incorporates striding with an equivalent non-striding CNN which has more filters and smaller size. Through this equivalence we are then able to characterize striding as an additional mechanism for parameter sharing among channels, thus reducing training complexity. Finally, the framework presented in this paper offers a new mathematical perspective on the role of striding which we hope shall facilitate and simplify the future theoretical analysis of CNNs.

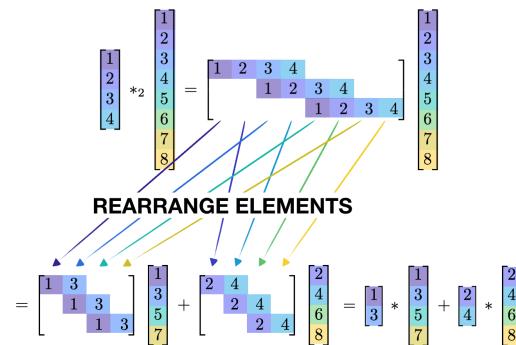
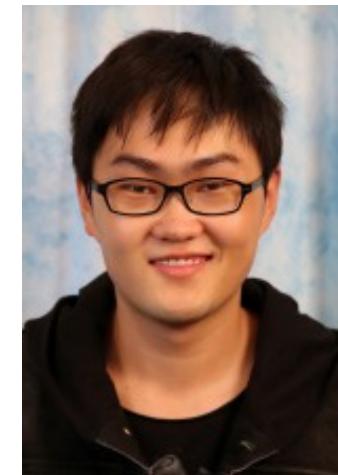


Figure 1: A toy example to show how 1-D convolution of stride two can be reduced to stride one by simply rearranging the elements in both filters and signal. The operator  $*_2$  denotes a convolution operator with stride of two. One can see by rearranging the columns in the banded strided Toeplitz matrix, the convolution of stride two is equivalent to the summation of the conventional convolutional opera-



# Capas Convolucionales 2D

- En el caso 2D vale exactamente lo mismo. Recordemos el ejemplo en que usamos un filtro 2D con campos receptivos de  $3 \times 3$  y strides de  $1 \times 1$ , para procesar un input de  $5 \times 5$

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

Input

0	1	2
2	2	0
0	1	2

pesos W

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

FM

# Capas Convolucionales 2D

- El resultado lo calculamos correlacionando el filtro con el volumen de entrada. Por ejemplo el resultado en la posición (0,0) lo calculamos como:

3 <sub>0</sub>	3 <sub>1</sub>	2 <sub>2</sub>	1	0
0 <sub>2</sub>	0 <sub>2</sub>	1 <sub>0</sub>	3	1
3 <sub>0</sub>	1 <sub>1</sub>	2 <sub>2</sub>	2	3
2	0	0	2	2
2	0	0	0	1

0	1	2
2	2	0
0	1	2

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

# Capas Convolucionales 2D

- El resultado en la posición (0,1) lo calculamos así:

3	3 <sub>0</sub>	2 <sub>1</sub>	1 <sub>2</sub>	0
0	0 <sub>2</sub>	1 <sub>2</sub>	3 <sub>0</sub>	1
3	1 <sub>0</sub>	2 <sub>1</sub>	2 <sub>2</sub>	3
2	0	0	2	2
2	0	0	0	1

0	1	2
2	2	0
0	1	2

12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

# Capas Convolucionales 2D

- Esto lo podemos escribir ahora como:

$$\begin{matrix} 12.0 & 12.0 & 17.0 \\ 10.0 & 17.0 & 19.0 \\ 9.0 & 6.0 & 14.0 \end{matrix} = \begin{matrix} 0 & 1 & 2 \\ 2 & 2 & 0 \\ 0 & 1 & 2 \end{matrix} \star \begin{matrix} 3 & 3 & 2 & 1 & 0 \\ 0 & 0 & 1 & 3 & 1 \\ 3 & 1 & 2 & 2 & 3 \\ 2 & 0 & 0 & 2 & 2 \\ 2 & 0 & 0 & 0 & 1 \end{matrix}$$

$$= \text{flip} \left( \begin{matrix} 2 & 1 & 0 \\ 0 & 2 & 2 \\ 2 & 1 & 0 \end{matrix} \right) * \begin{matrix} 3 & 3 & 2 & 1 & 0 \\ 0 & 0 & 1 & 3 & 1 \\ 3 & 1 & 2 & 2 & 3 \\ 2 & 0 & 0 & 2 & 2 \\ 2 & 0 & 0 & 0 & 1 \end{matrix}$$

# Capas Convolucionales 2D

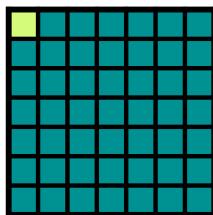
- El kernel de la capa es en realidad la matriz que usábamos para correlacionar, pero volteada.

$$\begin{matrix} 12.0 & 12.0 & 17.0 \\ 10.0 & 17.0 & 19.0 \\ 9.0 & 6.0 & 14.0 \end{matrix} = \text{flip} \left( \begin{matrix} 2 & 1 & 0 \\ 0 & 2 & 2 \\ 2 & 1 & 0 \end{matrix} \right) * \begin{matrix} 3 & 3 & 2 & 1 & 0 \\ 0 & 0 & 1 & 3 & 1 \\ 3 & 1 & 2 & 2 & 3 \\ 2 & 0 & 0 & 2 & 2 \\ 2 & 0 & 0 & 0 & 1 \end{matrix}$$

- Además, sabemos ahora que lo que computábamos era una convolución válida. En efecto, el input era de  $I \times I$ , el kernel era de  $S \times S$  y el resultado es de  $I - S + 1 \times I - S + 1$ .

# Capas Convolucionales 2D

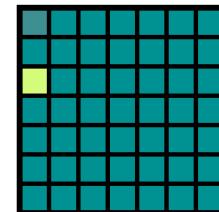
- Si preferimos, podemos implementar la capa usando una convolución completa o una convolución con un padding que nos permita obtener el mismo tamaño de salida.



0	1	2
2	2	0
0	1	$3_2$

0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1



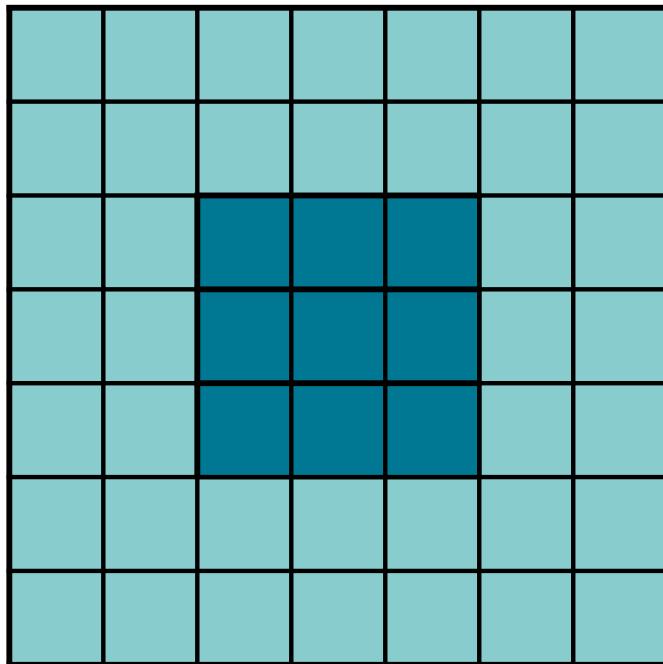
0	1	$3_2$	3	2	1	0
2	2	$0_0$	0	1	3	1
0	1	$3_2$	1	2	2	3

2	0	0	2	2
2	0	0	0	1

# Capas Convolucionales 2D

- Como en el caso 1D, podemos recuperar la convolución válida desde la convolución completa removiendo exactamente  $S-1$  elementos al comienzo y al final en cada dimensión:

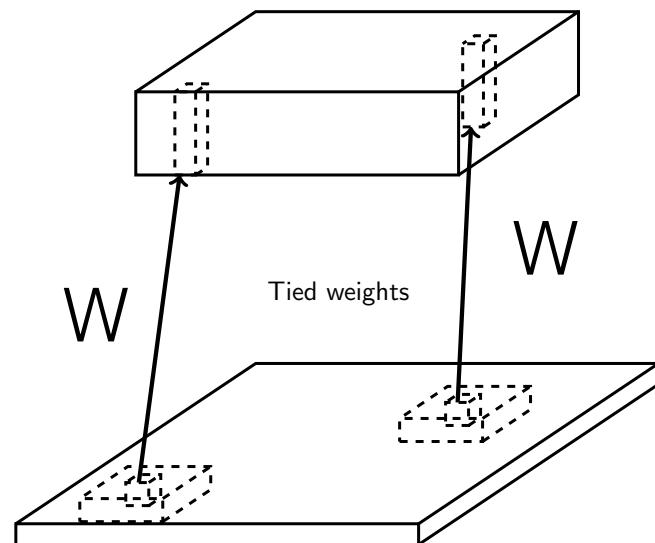


12.0	12.0	17.0
10.0	17.0	19.0
9.0	6.0	14.0

# Capas Convolucionales 2D

- Dada una señal bi-dimensional  $x$ , una capa convolucional bi-dimensional es una capa compuesta de  $O$  canales, cada uno de los cuales que implementa una transformación de la forma:

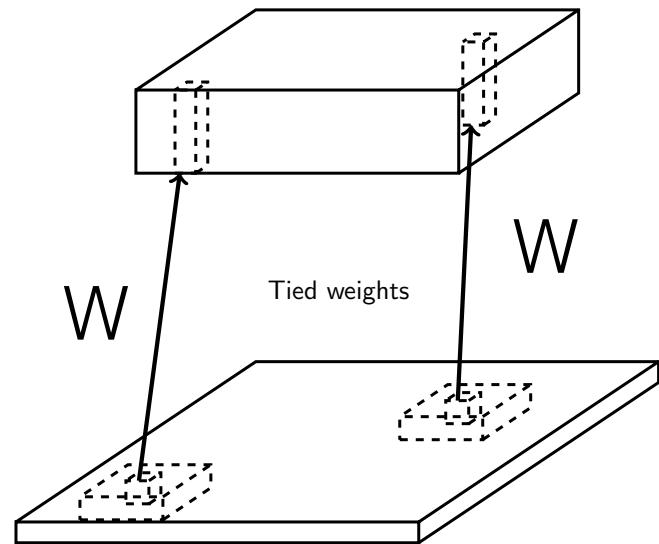
$$y = g(w * x - b)$$



- La señal  $w$  se denomina el filtro o el kernel correspondiente al canal y es aprendible desde datos.

# Capas Convolucionales

- En el caso de que a una capa bi-dimensional entre una señal 3D, podemos convenir lo mismo que habíamos acordado antes:



$$y_o = g \left( \sum_k w_{ok} * x_k - b_{ok} \right)$$

- En este caso, tenemos matrices diferentes que actúan sobre cada canal de entrada ( $k$  indexa los canales de entrada y  $o$  los canales de salida).

# Entonces ...

- La acción que efectuar nuestras capas convolucionales sobre el arreglo de entrada la podemos formalizar como una convolución del orden requerido.
- Hasta ahora habíamos computado esa operación correlacionando el arreglo de pesos con el volumen entrante. Ese arreglo de pesos corresponde en realidad al volteo del filtro que corresponde a la capa.
- En las secciones anteriores habíamos operado nuestras redes usando un tipo de convolución denominada convolución válida. Es posible obtener esa convolución desde la convolución clásica (completa) removiendo una cierta cantidad de elementos al comienzo y a final en cada dimensión.
- Es posible obtener una convolución completa desde la convolución usando padding, es decir, expandiendo el arreglo de entrada y rellenando el volumen de entrada con una cierta cantidad de ceros al comienzo y a final en cada dimensión. En la práctica, el padding nos permite obtener la forma que queramos de salida.

