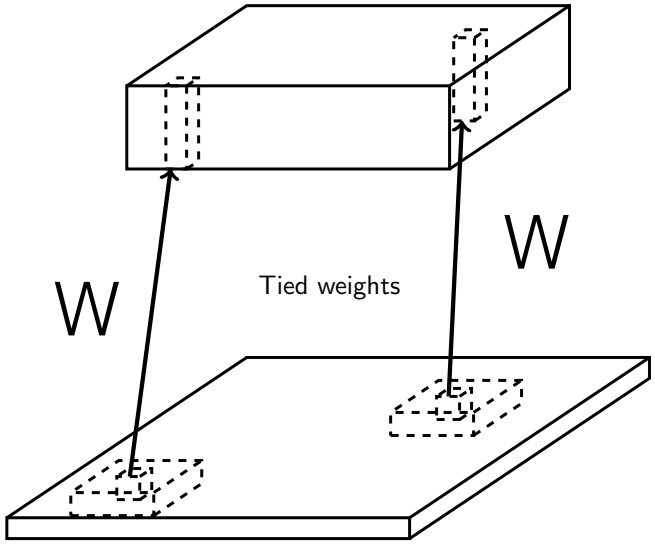
Redes Convolucionales

Señales y Convoluciones

Convoluciones

- El objetivo de este capítulo es definir la operación de convolución para poder formalizar matemáticamente la operación que efectúan las capas convencionales de una CNN.



$$y = \sigma(x * k_w)$$

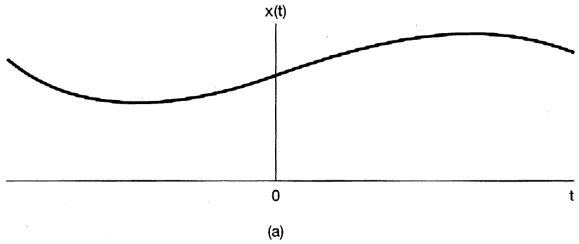
Señales

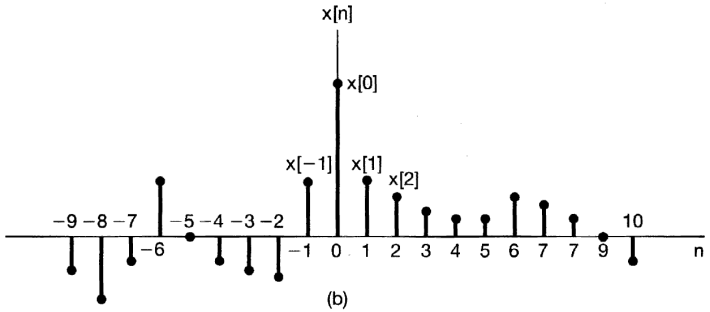
- Una señal se puede definir como una función (de 1 o más variables independientes) que se utiliza para representar el patrón de variación de cierta cantidad (variable dependiente) de interés (*).
- Por ejemplo, una señal de voz podría representar cómo cambia **en el tiempo** el sonido (presión acústica) emitido por una persona.
- Una imagen podría describir como cambia **en el espacio** la radiación (luz) recibida desde un conjunto de objetos.

(*) Oppenheim et al. (1996) Signals and Systems, MIT Press.

Señales

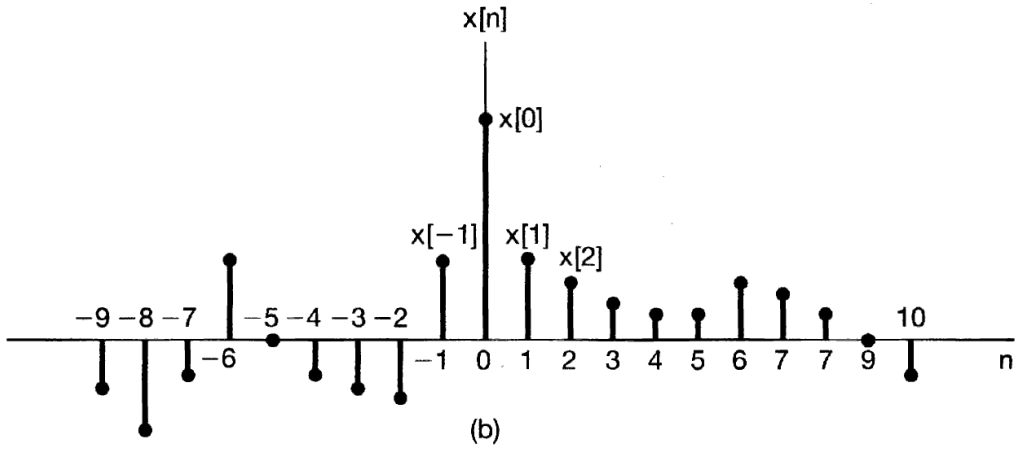
- Una señal se dice continua cuando la variable independiente es continua y **discreta cuando la variable independiente es discreta**. Muchas veces, una señal discreta se obtiene después de muestrear una señal continua para procesarla en un computador/dispositivo digital.





Señales

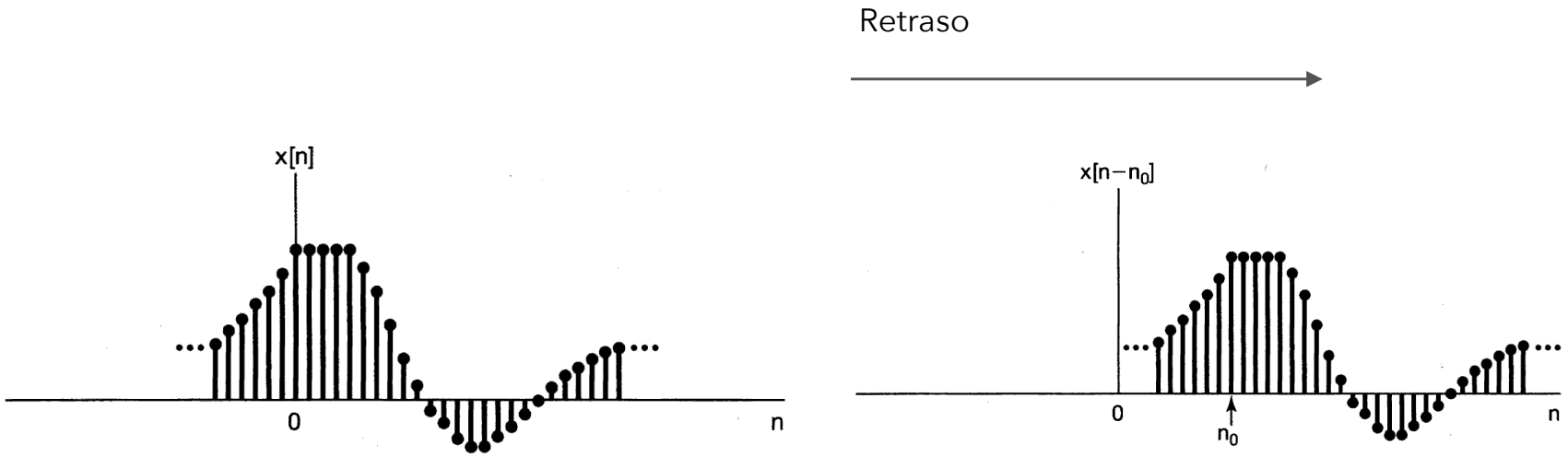
- En adelante nos referimos a la variable independiente como el tiempo, pero claramente puede tratarse de otra cantidad física dependiendo del contexto.
- Preferiremos x[n] para representar una señal discreta donde n es la variable independiente y x la variable dependiente. Para enfatizar el caso continuo escribiremos x(t). A no ser que especifiquemos lo contrario, el en el caso discreto $n \in \mathbb{Z}$ y $t \in \mathbb{R}$.



Retardos y Avances

- Parte importante del estudio de una señal consiste en describir como ésta se transforma para generar otras señales. Dos transformaciones básicas consisten en **retrasar y avanzar** una señal en el tiempo.

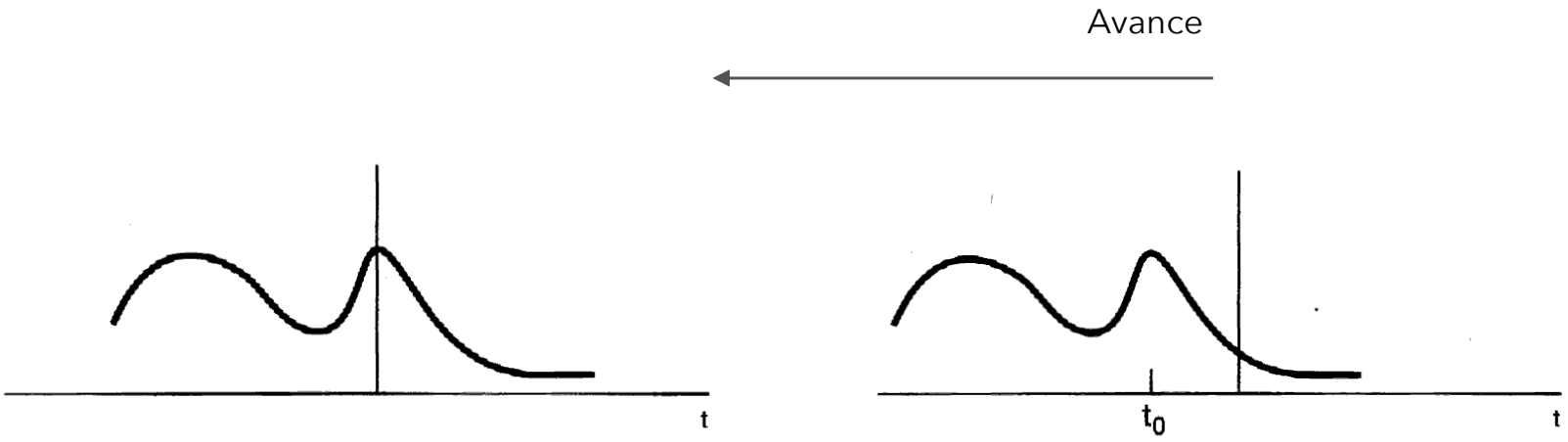
$$y[n] = x[n - n_0] \quad \forall n \ (n_0 > 0)$$



Retardos y Avances

- Parte importante del estudio de una seña consiste en describir como ésta se transforma para generar otras señales. Dos transformaciones básicas consisten en **retrasar y avanzar** una señal en el tiempo.

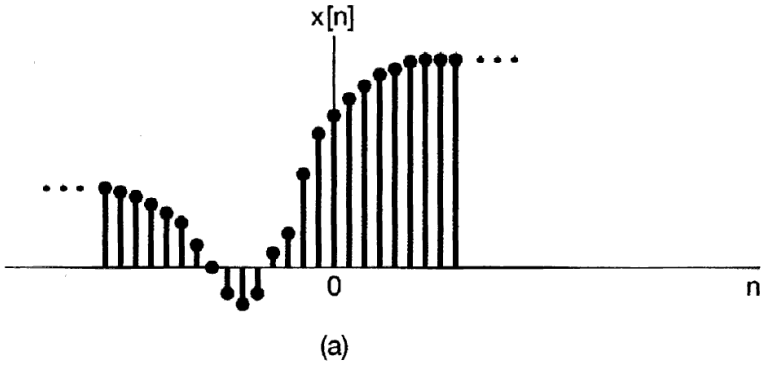
$$y[n] = x[n + n_0] \quad \forall n \ (n_0 > 0)$$
$$= x[n - n_0] \quad \forall n \ (n_0 < 0)$$

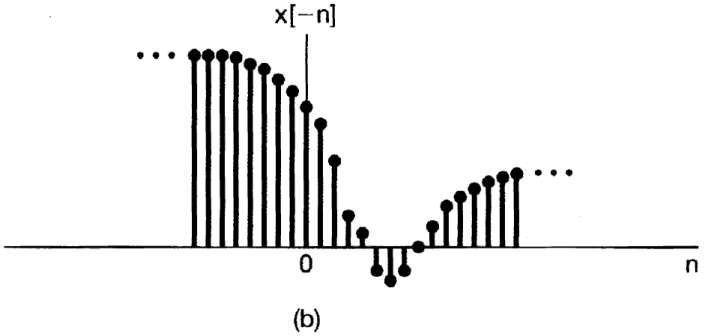


Reflexiones y Escalamientos

- Otras operaciones importantes son las reflexiones y los escalamientos. Una reflexión se define como sigue:

$$y[n] = x[-n] \quad \forall n$$

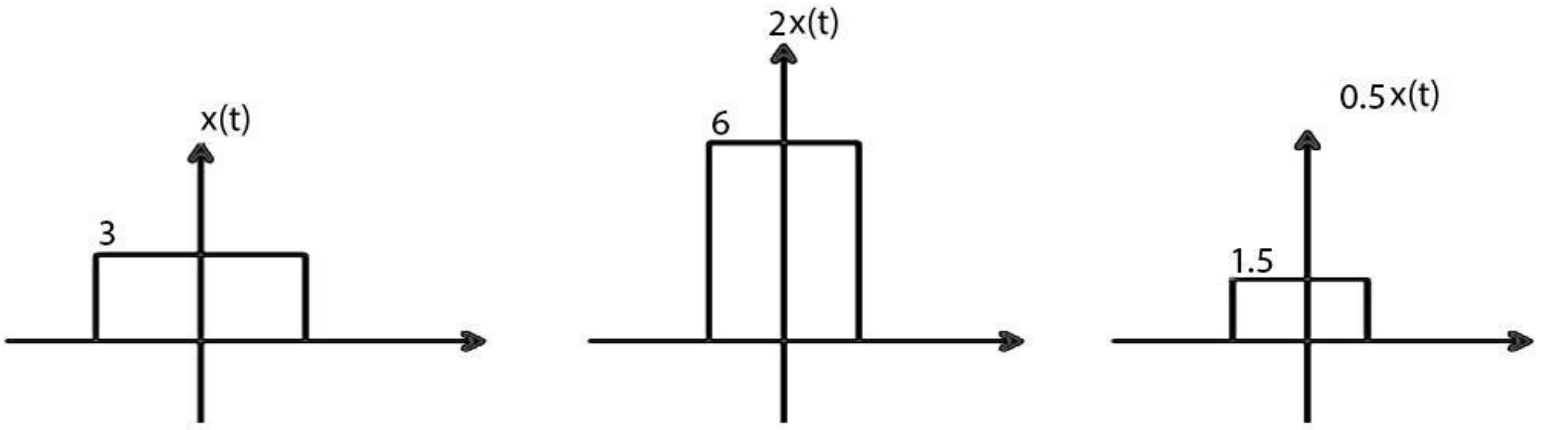




Reflexiones y Escalamientos

- Un escalamiento puede ocurrir en la variable dependiente (escalamiento de amplitud o simplemente escalamiento) o en a variable independiente (escalamiento o dilatación temporal). La primera se define como sigue

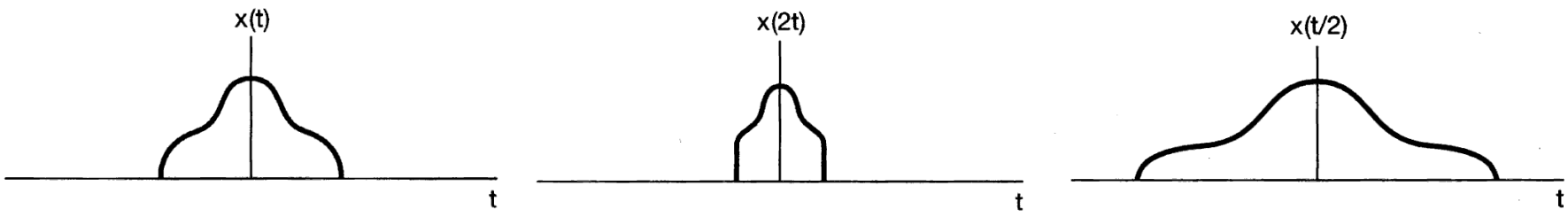
$$y[n] = \alpha \cdot x[n] \quad \forall n \ (\alpha \in \mathbb{R})$$



Reflexiones y Escalamientos

- Un escalamiento temporal en cambio consiste en la siguiente operación:

$$y[n] = x[k\,n] \quad \forall n(k \in \mathbb{R}^{+})$$

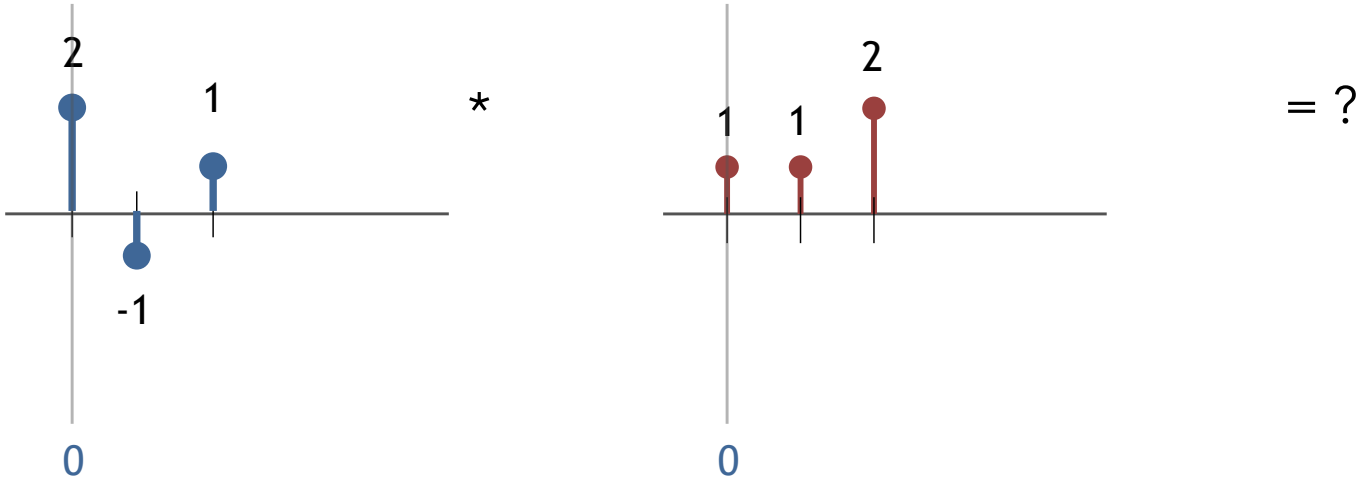


Convolución 1D

- Una operación clave para describir transformaciones que puede experimentar una señal es la **convolución.** En el caso 1D, la convolución de dos señales $x[n]$ e $y[n]$ será otra señal 1D $z[n]$ denotada $x * y$ definida como

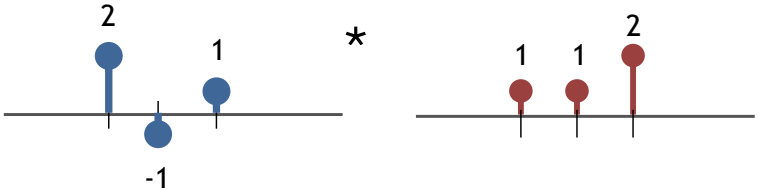
$$z[n] = x * y[n] = \sum_{k=-\infty}^{\infty} x[k]y[n-k] \quad \forall n$$

- Por ejemplo:



Convolución vía Corrimientos

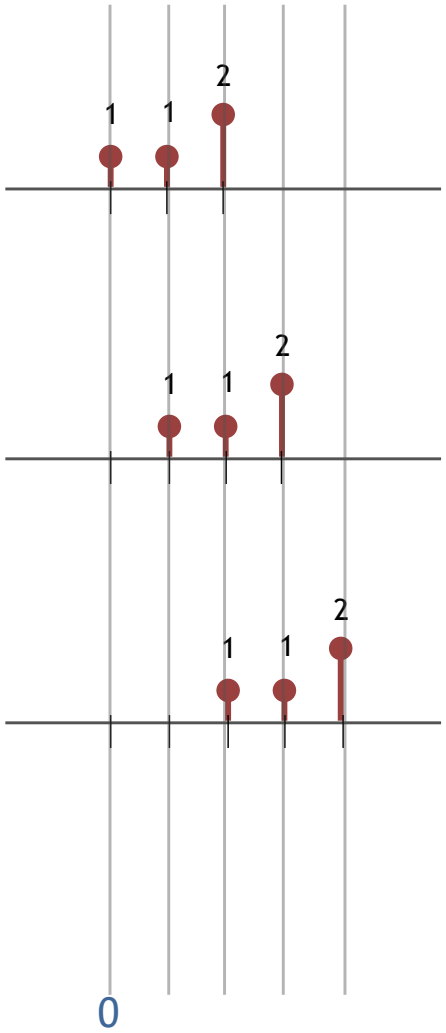
$$z[n] = x * y[n] = \sum_{k=-\infty}^{\infty} x[k]y[n-k] \quad \forall n$$



2 x

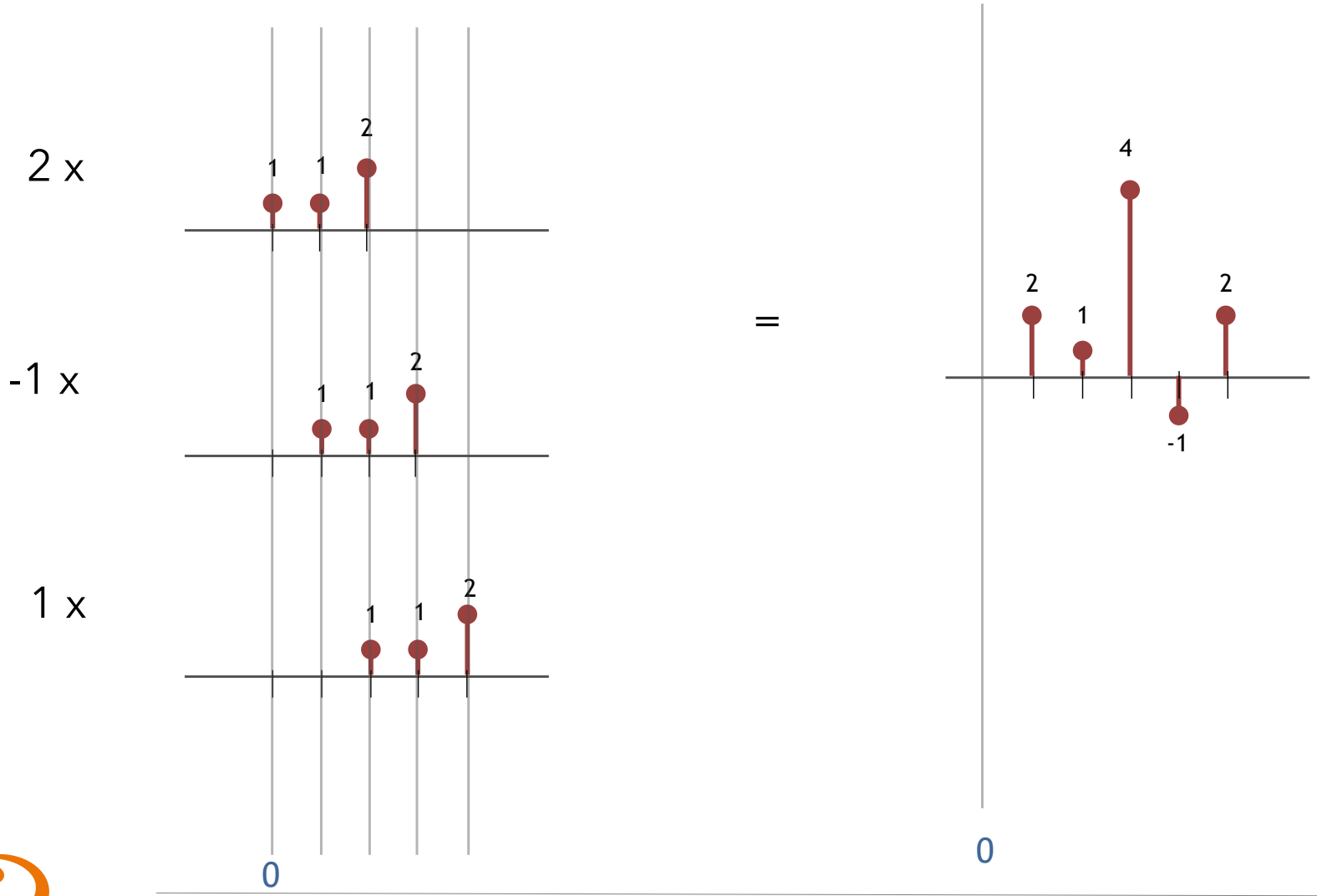
-1 x

1 x



Convolución vía Corrimientos

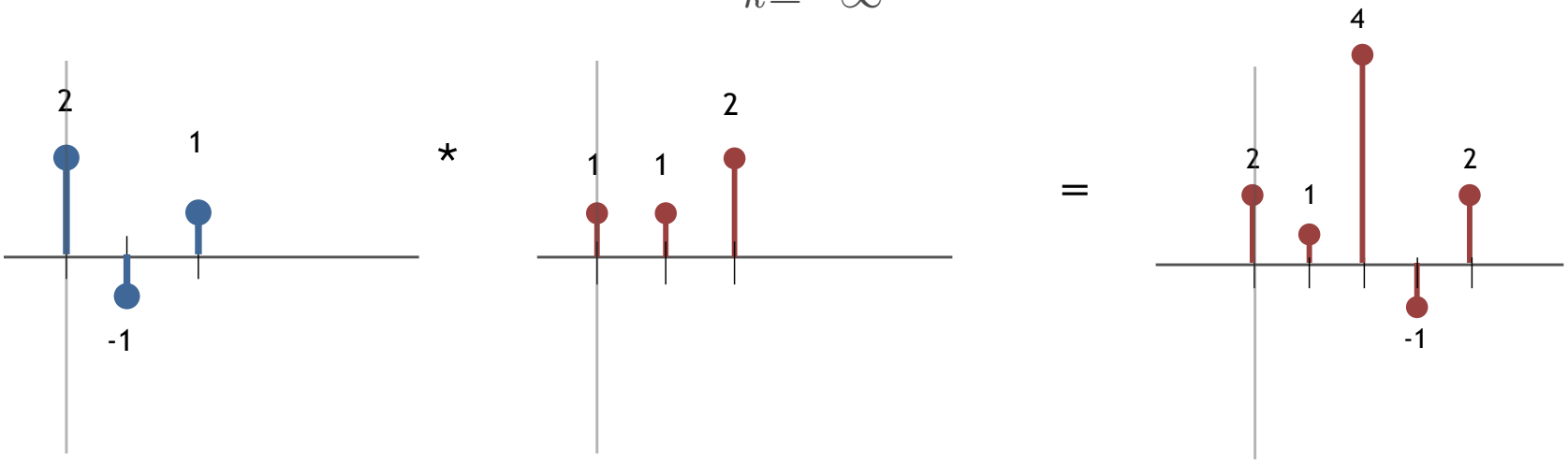
$$z[n] = x * y[n] = \sum_{k=-\infty}^{\infty} x[k]y[n-k] \quad \forall n$$



Convolución vía Corrimientos

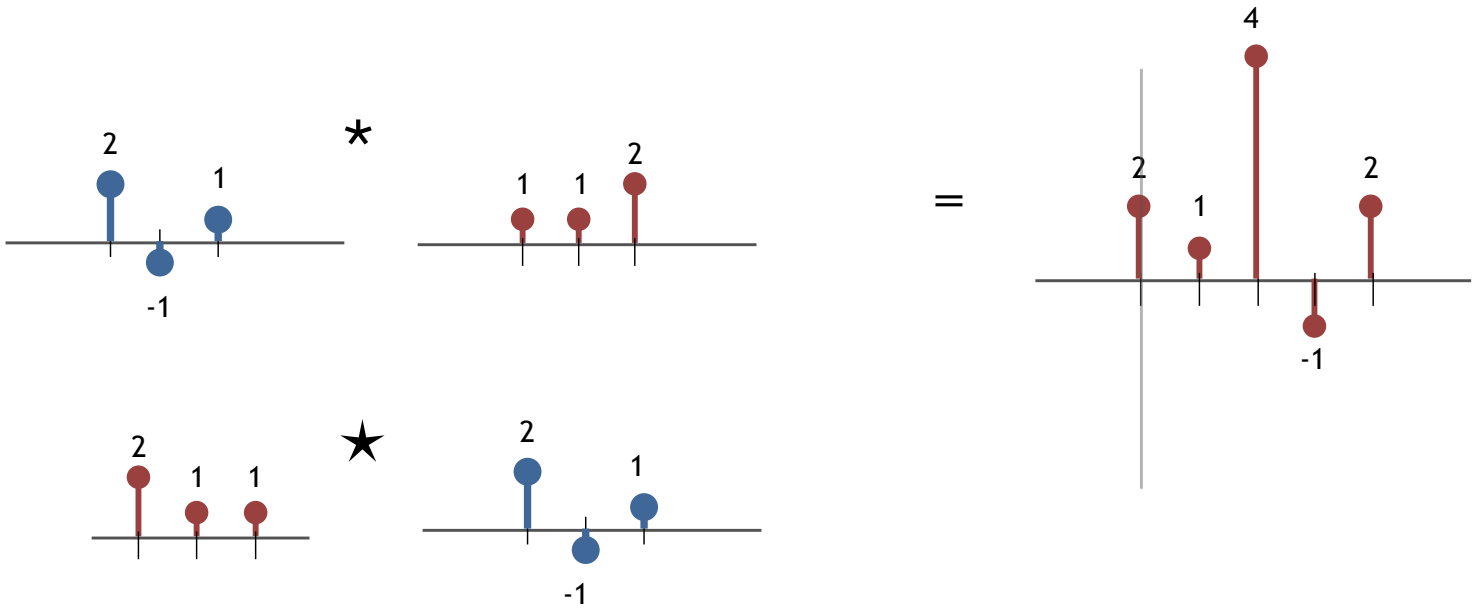
- Entonces …

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



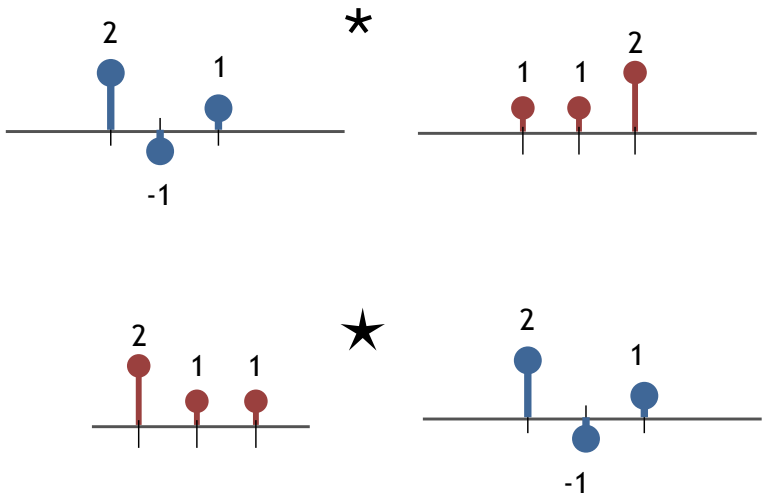
Convolución vía Correlación

- Es posible también calcular la convolución haciendo un flip de la segunda señal y correlacionándola con la primera:

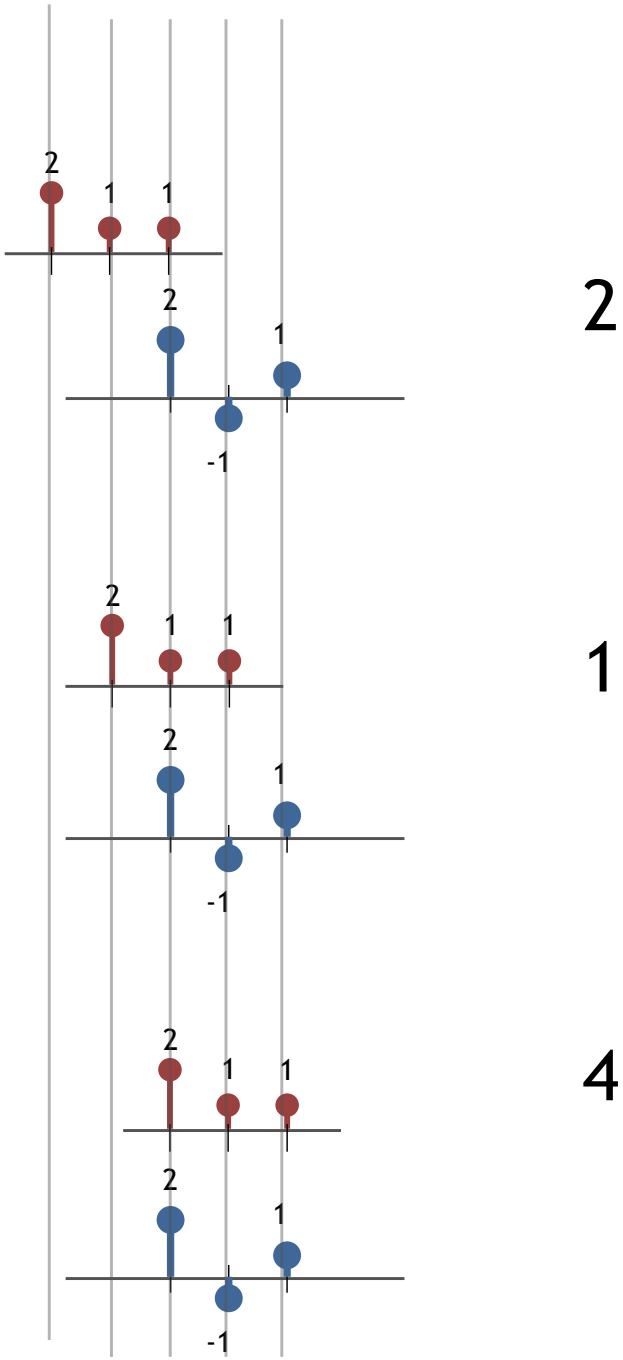


Convolución vía Correlación

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



=



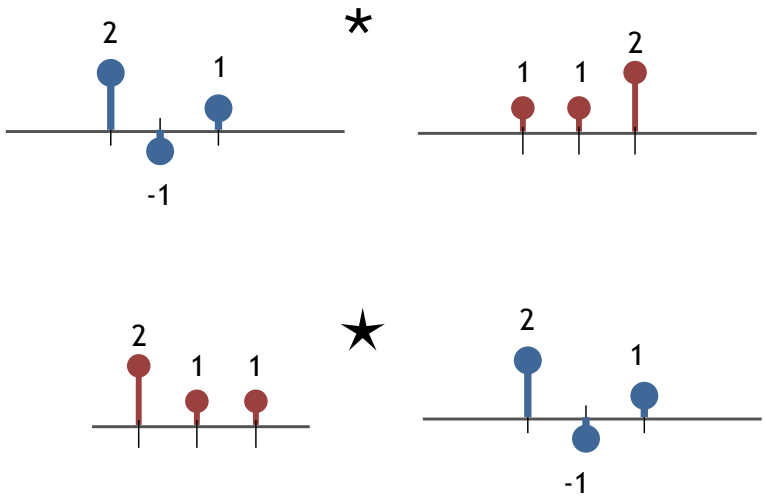
2

1

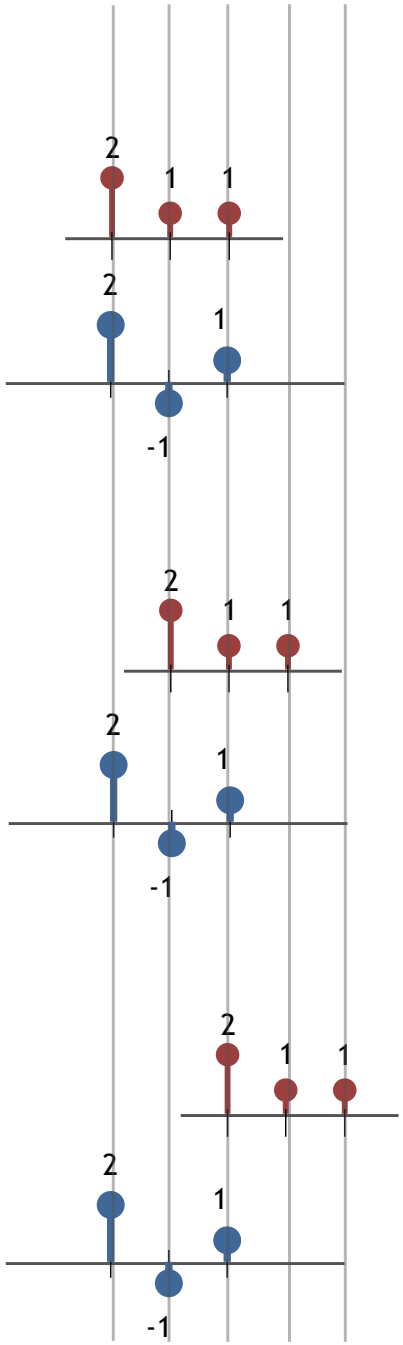
4

Convolución vía Correlación

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

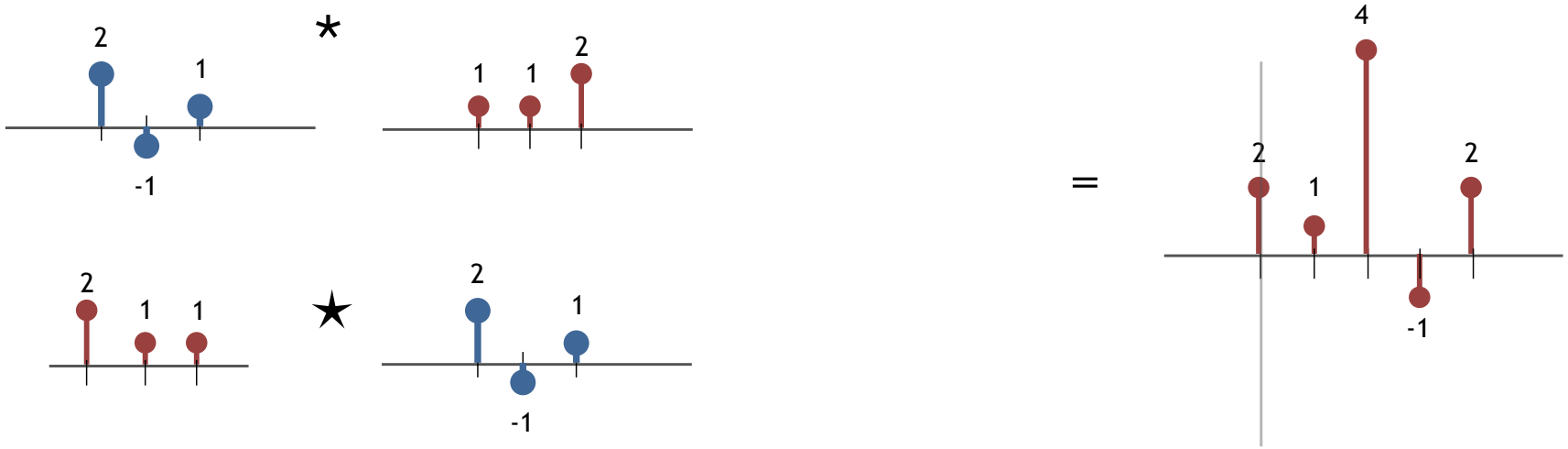


=



Convolución vía Correlación

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

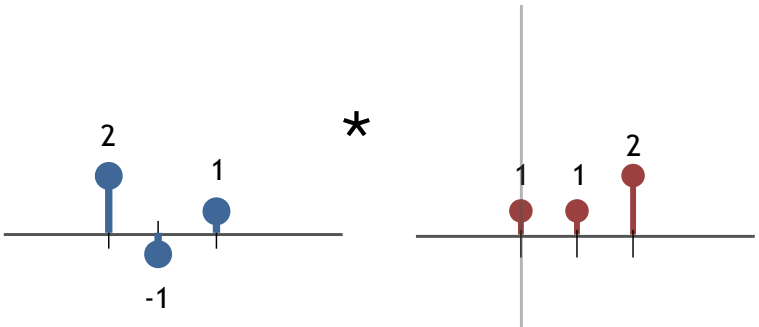


Convolución vía Correlación

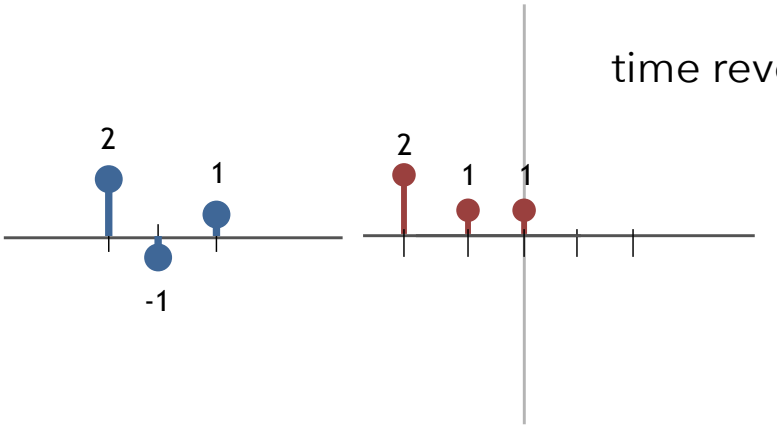
$$z[n] = x * y[n] = \sum_{k=-\infty}^{\infty} x[k]y[n-k] \quad \forall n$$

$$\bar{y}[t] = y[-t]$$
$$\bar{y}[k-n] = y[n-k]$$
$$\bar{y}[k+(-n)] = y[n-k]$$

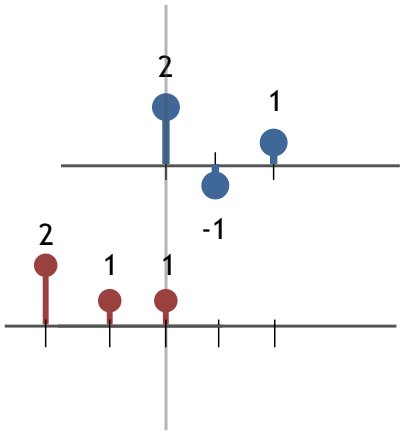
time reversal +
time shift



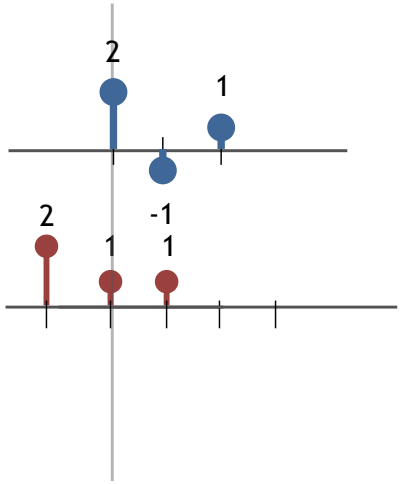
time reversal



time reversal +
time shift
n=0

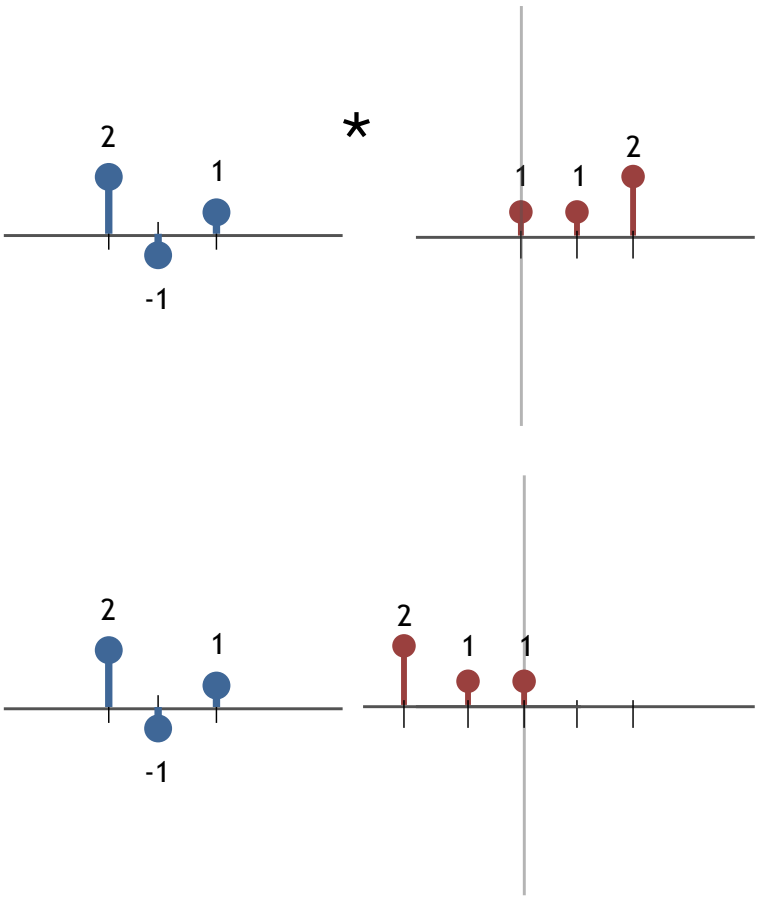


time reversal +
time shift
n=1

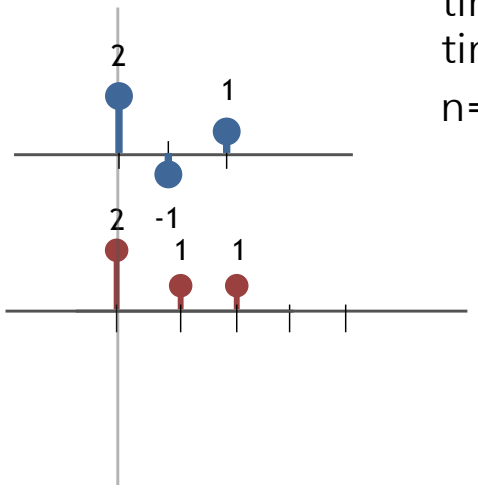


Convolución vía Correlación

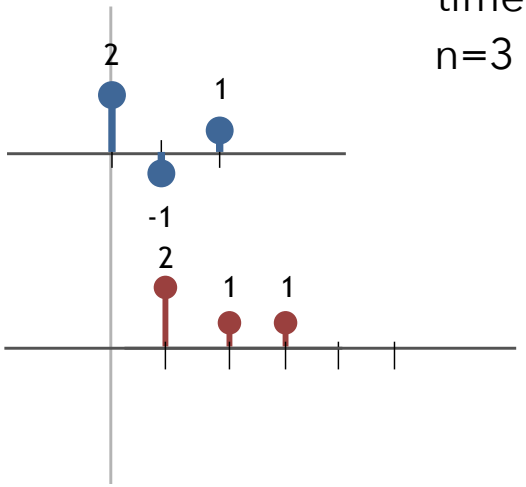
$$z[n] = x * y[n] = \sum_{k=-\infty}^{\infty} x[k]y[n-k] \quad \forall n$$



time reversal +
time shift
n=2



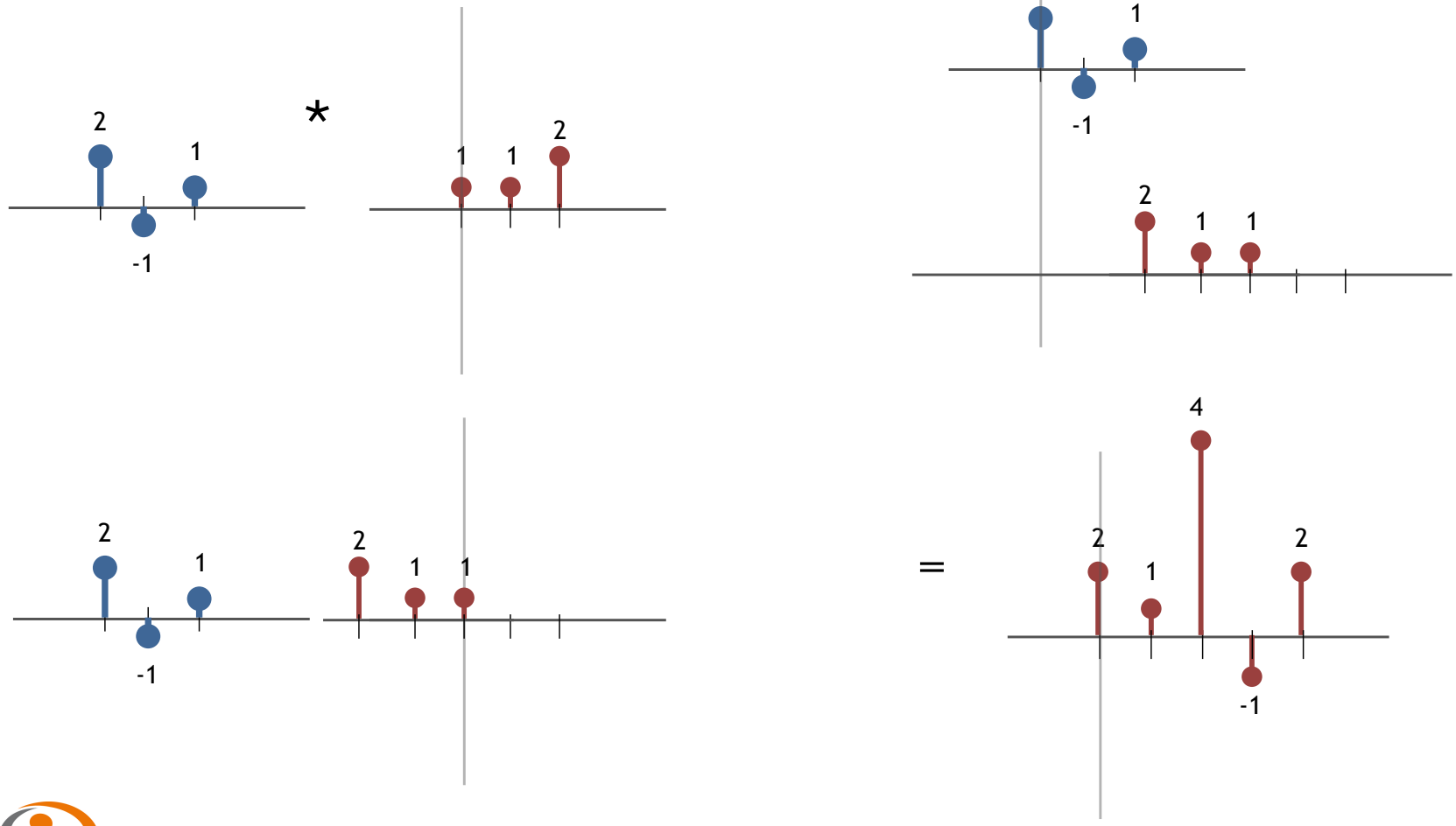
time reversal +
time shift
n=3



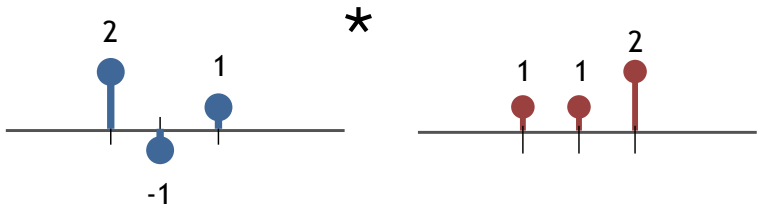
Convolución vía Correlación

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

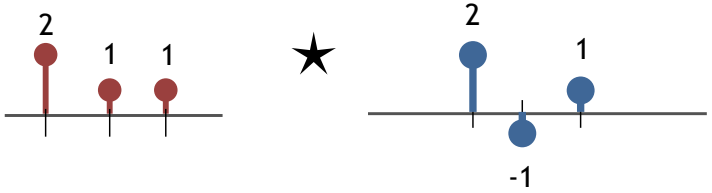
time reversal +
time shift
n=4



Convolución vía Correlación



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



$$z[n] = \bar{y} \star x[n] = \sum_{k=-\infty}^{\infty} x[k]\bar{y}[k-n]$$

Convolución

- Propiedades
  - Conmutatividad: $x * y = y * x$
  - Asociatividad: $x * y * z = (x * y) * z = x * (y * z)$
  - Distributividad: $x * (y + z) = (x * y) + (x * z)$
  - Linealidad:

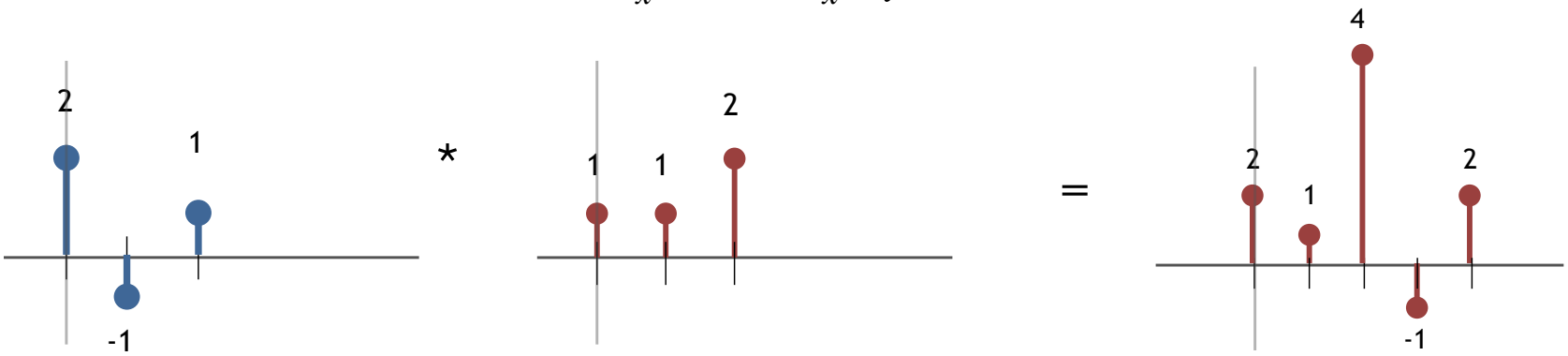
$$(\alpha x + \beta y) * z = \alpha(x * z) + \beta(y * z)$$

$$x * (\alpha y + \beta z) = \alpha(x * y) + \beta(x * z)$$

Convolución via Producto Matricial

- La linealidad implica que podremos siempre escribir la convolución como una operación matricial sobre las representaciones vectoriales de las señales. Es decir, si $z = x * y$

$$\exists A_x \ : \ \vec{z} = A_x \cdot \vec{y}$$



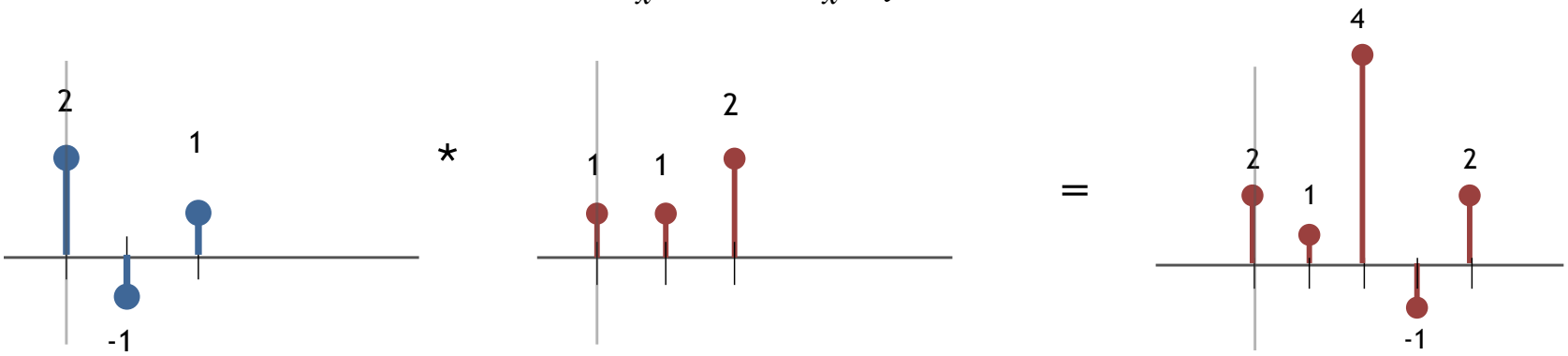
Toeplitz Matrix

$$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ -1 & 2 & 0 & 0 & 0 \\ 1 & -1 & 2 & 0 & 0 \\ 0 & 1 & -1 & 2 & 0 \\ 0 & 0 & 1 & -1 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 2 \\ 0 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 2 \\ 1 \\ 4 \\ -1 \\ 2 \end{pmatrix}$$

Convolución via Producto Matricial

- La linealidad implica que podremos siempre escribir la convolución como una operación matricial sobre las representaciones vectoriales de las señales. Es decir, si $z = x * y$

$$\exists A_x \ : \ \vec{z} = A_x \cdot \vec{y}$$

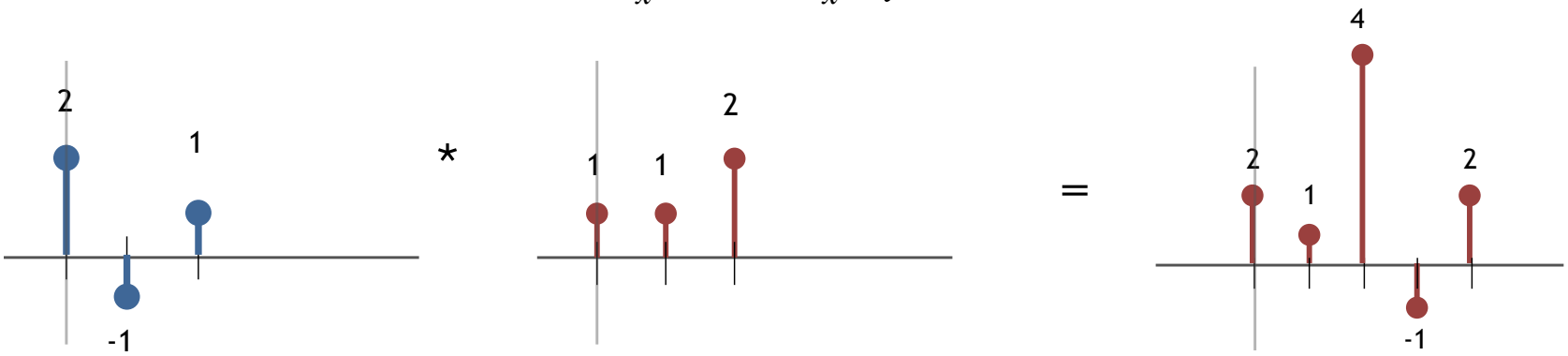


$$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ -1 & 2 & 0 & 0 & 0 \\ 1 & -1 & 2 & 0 & 0 \\ 0 & 1 & -1 & 2 & 0 \\ 0 & 0 & 1 & -1 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 2 \\ 0 \\ 0 \end{pmatrix} \qquad \begin{pmatrix} 2 \\ 1 \\ 4 \\ -1 \\ 2 \end{pmatrix}$$

Convolución via Producto Matricial

- La linealidad implica que podremos siempre escribir la convolución como una operación matricial sobre las representaciones vectoriales de las señales. Es decir, si $z = x * y$

$$\exists A_x \ : \ \vec{z} = A_x \cdot \vec{y}$$



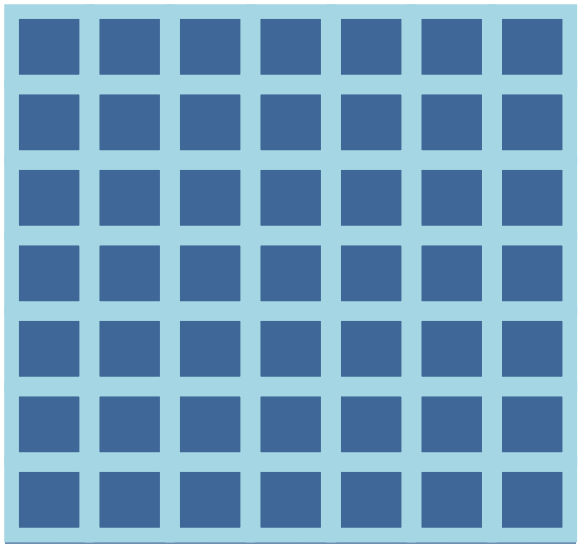
Toeplitz Matrix

$$\begin{pmatrix} 1 & -1 & 2 & 0 & 0 \\ 0 & 1 & -1 & 2 & 0 \\ 0 & 0 & 1 & -1 & 2 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \\ 4 \\ -1 \\ 2 \end{pmatrix}$$

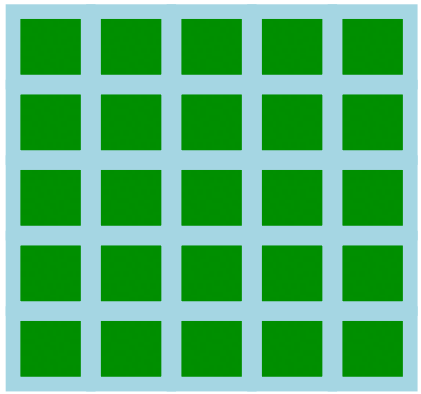
Convolución 2D

- La extensión al caso 2D (o kD) es muy sencilla. Dadas dos señales $x[n,m]$ e $y[n,m]$ a convención de x e y será otra señal 2D $z[n,m]$ denotada $x * y$ definida como

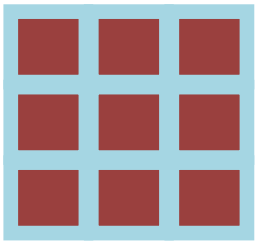
$$z[n,m] = x * y[n,m] = \sum_{k,l=-\infty}^{\infty} x[k,l]y[n-k,m-l] \quad \forall n,m$$



Z



X

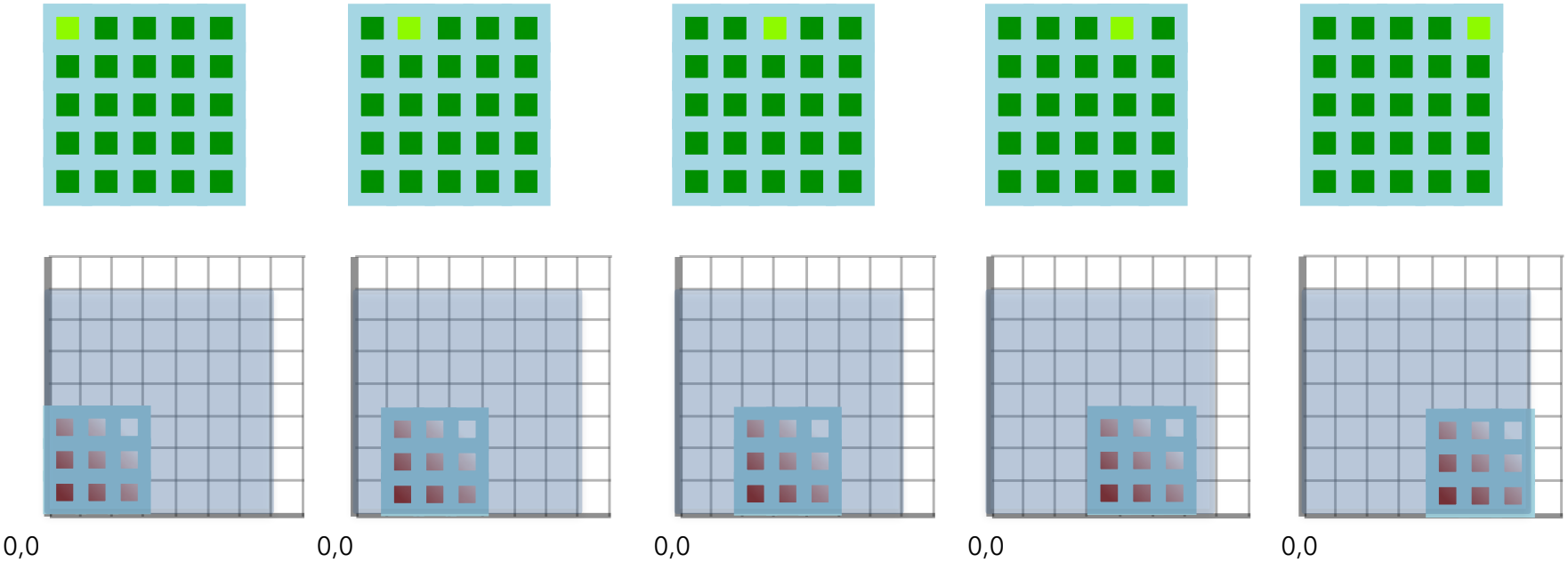


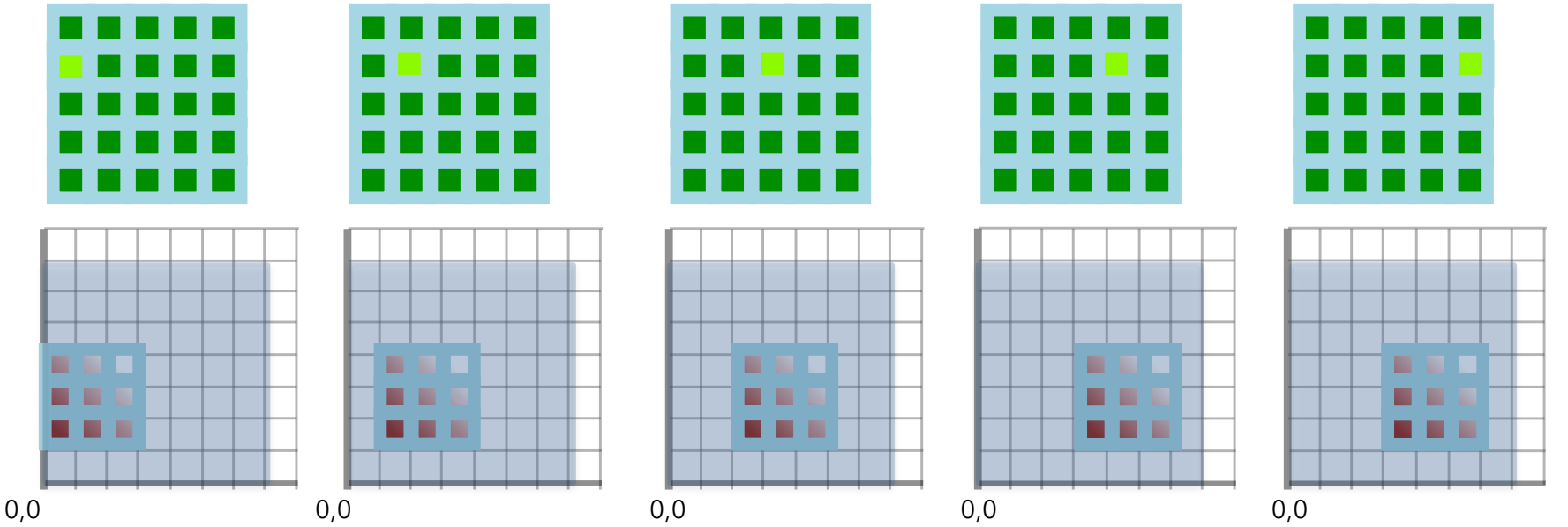
V

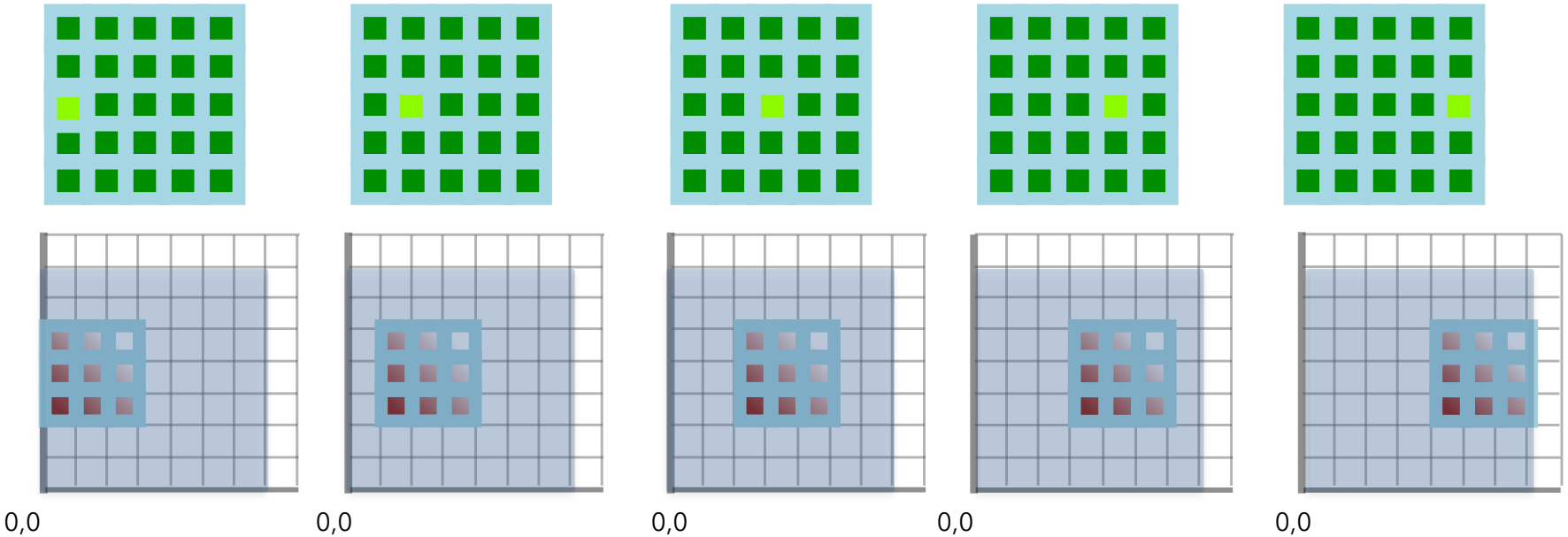
Convolución 2D vía Corrimientos

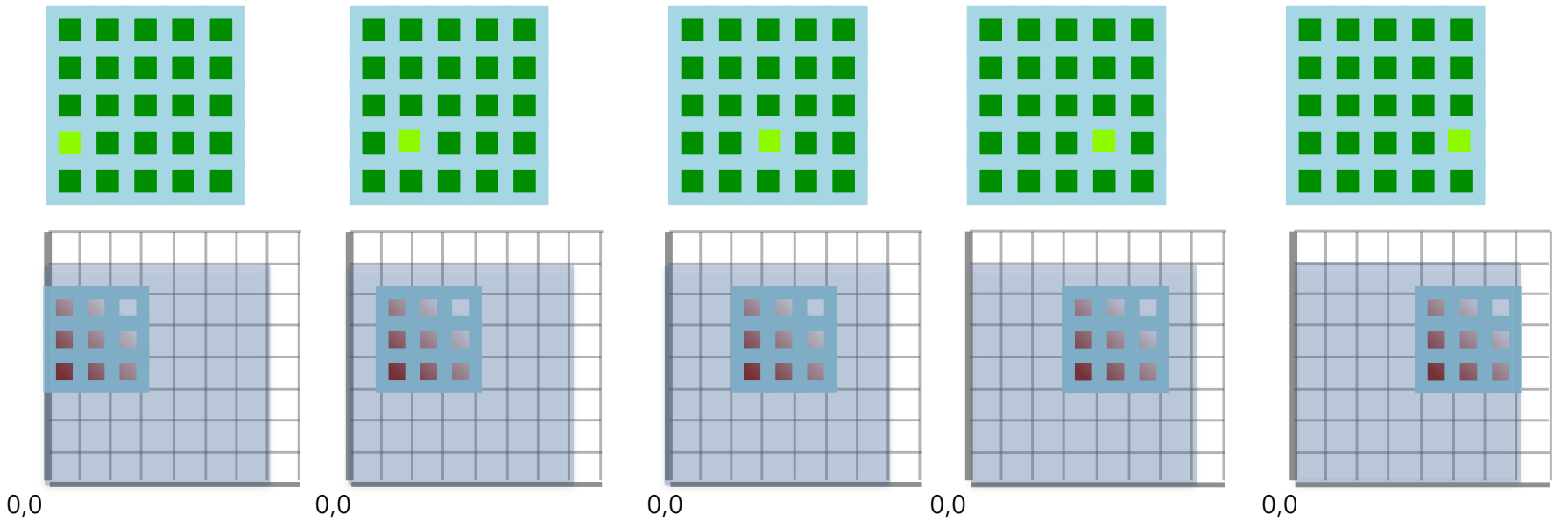
- En este caso, el cálculo vía corrimientos se vuelve engorroso e impráctico

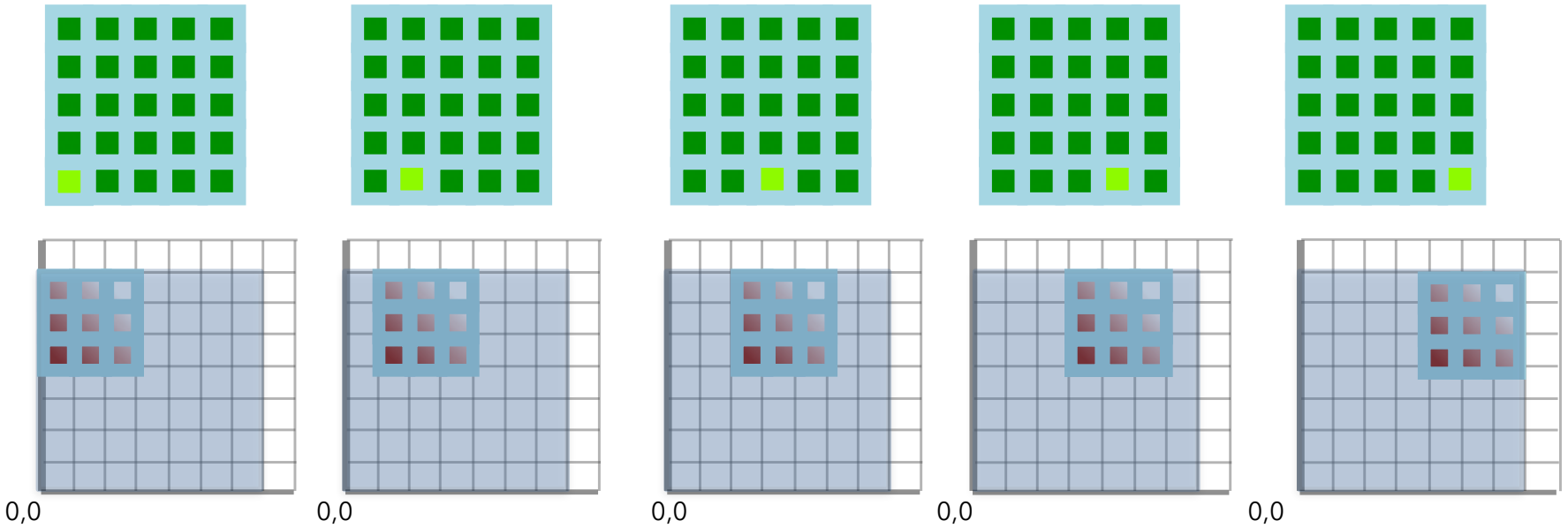
$$z[n,m] = x * y[n,m] = \sum_{k,l=-\infty}^{\infty} x[k,l]y[n-k,m-l] \quad \forall n,m$$




0,0
0,0
0,0
0,0
0,0


0,0
0,0
0,0
0,0
0,0

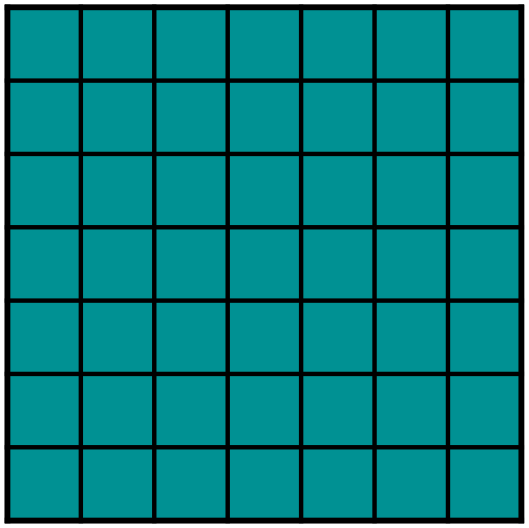

0,0
0,0
0,0
0,0
0,0


0,0
0,0
0,0
0,0
0,0

Convolución 2D

- Hacer el cálculo vía correlaciones resulta en cambio mucho más simple y es más consistente con la operación que habíamos hecho anteriormente.

$$z[n,m] = x * y[n,m] = \sum_{k,l=-\infty}^{\infty} x[k,l]y[n-k,m-l] \quad \forall n,m$$



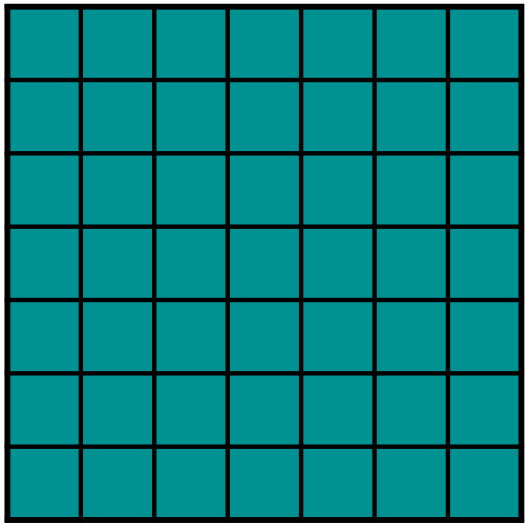
z

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

x

1	0	-1
0	2	0
0	1	1

y



z

=

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

x

*

1	0	-1
0	2	0
0	1	1

y

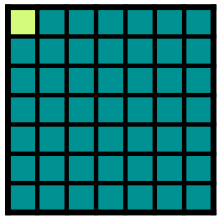
1	1	0
0	2	0
-1	0	1

$\bar{y}$

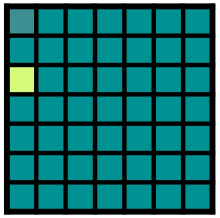
★

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

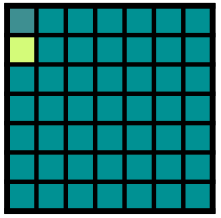
x



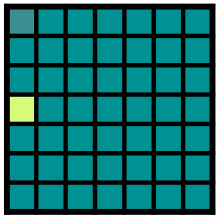
1	1	0				
0	2	0				
-1	0	3	3	2	1	0
		0	0	1	3	1
		3	1	2	2	3
		2	0	0	2	2
		2	0	0	0	1



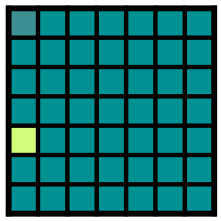
1	1	3	3	2	1	0
0	2	0	0	1	3	1
-1	0	3	1	2	2	3
		2	0	0	2	2
		2	0	0	0	1

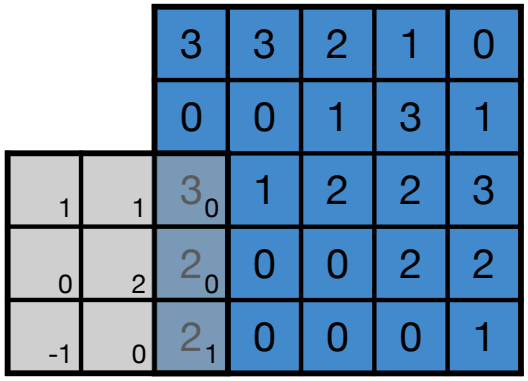


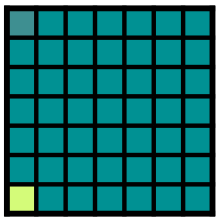
1	1	0				
0	2	3	3	2	1	0
-1	0	0	0	1	3	1
		3	1	2	2	3
		2	0	0	2	2
		2	0	0	0	1

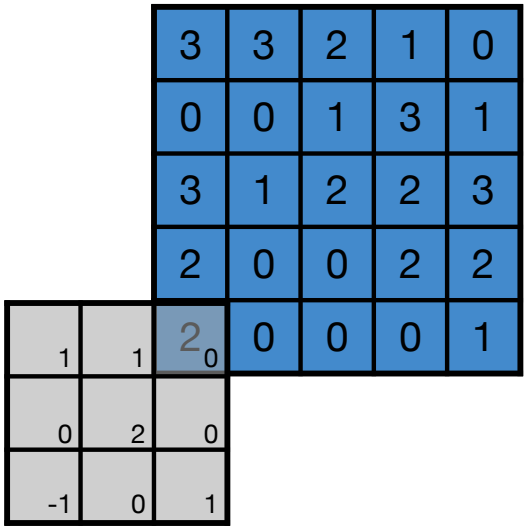


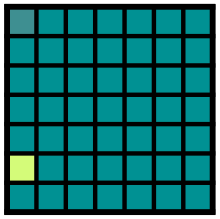
		3	3	2	1	0
1	1	0	0	1	3	1
0	2	3	1	2	2	3
-1	0	2	0	0	2	2
		2	0	0	0	1

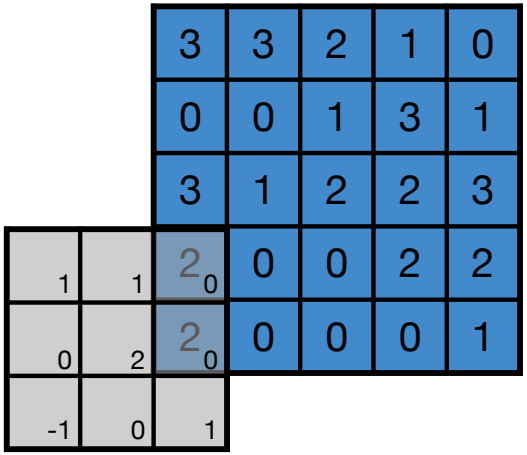


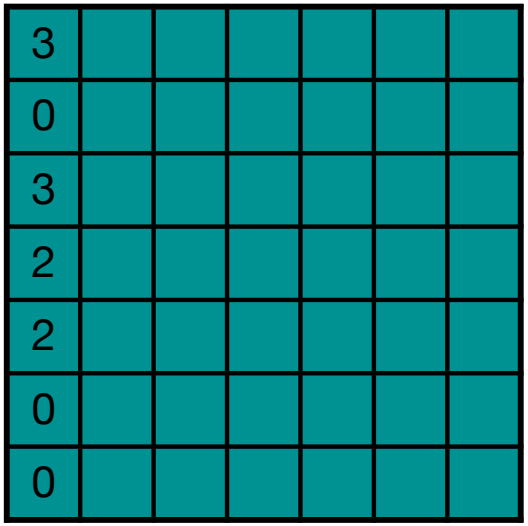

3 3 2 1 0
0 0 1 3 1
1 1 3 1 2 2 3
0 2 2 0 0 2 2
-1 0 2 1 0 0 0 1

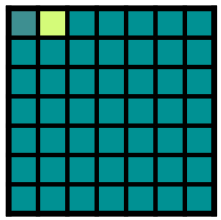


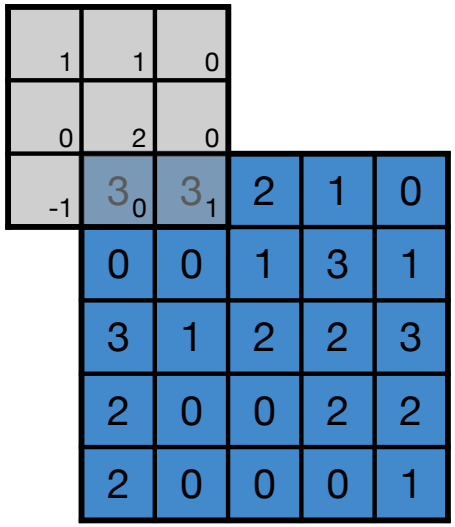

3 3 2 1 0
0 0 1 3 1
3 1 2 2 3
2 0 0 2 2
1 1 2 0 0 0 1
0 2 0
-1 0 1

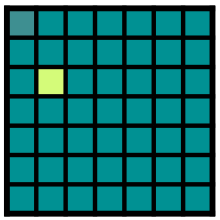


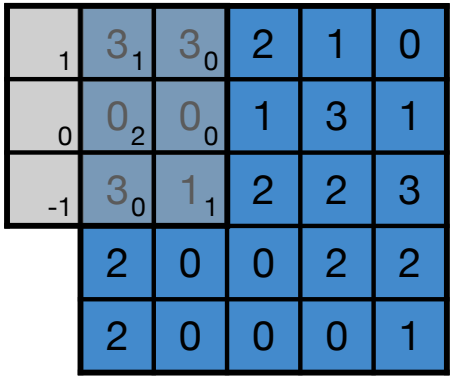

3 3 2 1 0
0 0 1 3 1
3 1 2 2 3
1 1 2 0 0 2 2
0 2 2 0 0 0 1
-1 0 1

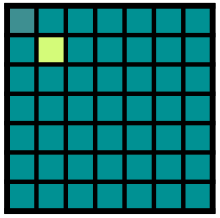

3
0
3
2
2
0
0

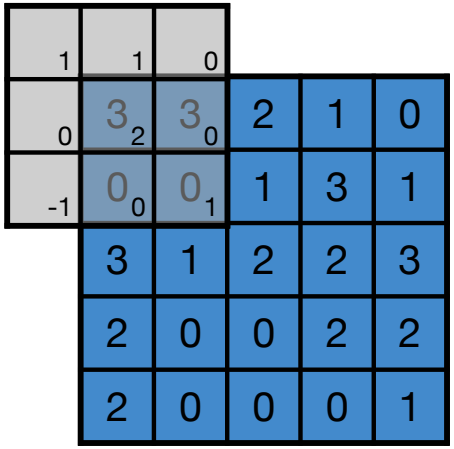


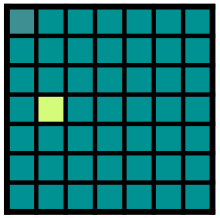

1 1 0
0 2 0
-1 3 0 3 1 2 1 0
0 0 1 3 1
3 1 2 2 3
2 0 0 2 2
2 0 0 0 1

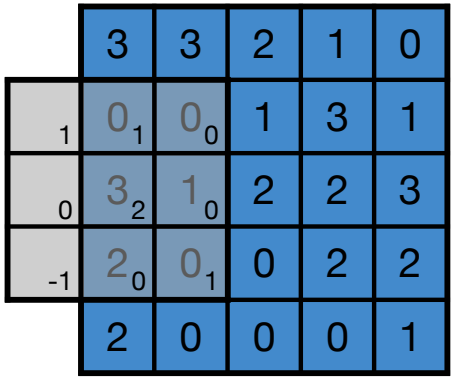


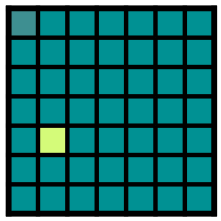

1 3 1 3 0 2 1 0
0 0 2 0 0 1 3 1
-1 3 0 1 1 2 2 3
2 0 0 2 2
2 0 0 0 1



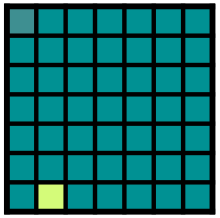

1 1 0
0 3 0 2 1 0
-1 0 0 0 1 1 3 1
3 1 2 2 3
2 0 0 2 2
2 0 0 0 1

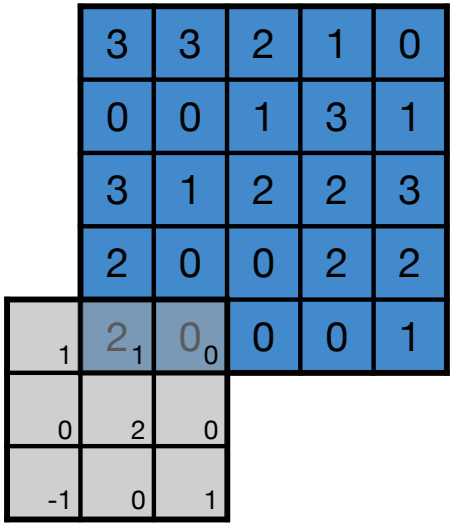


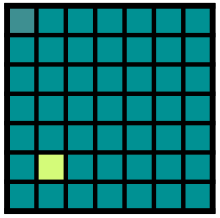

3 3 2 1 0
1 0 1 0 1 3 1
0 3 2 1 0 2 2 3
-1 2 0 0 1 0 2 2
2 0 0 0 1

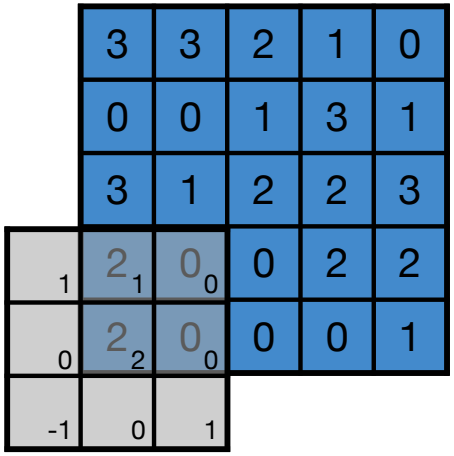


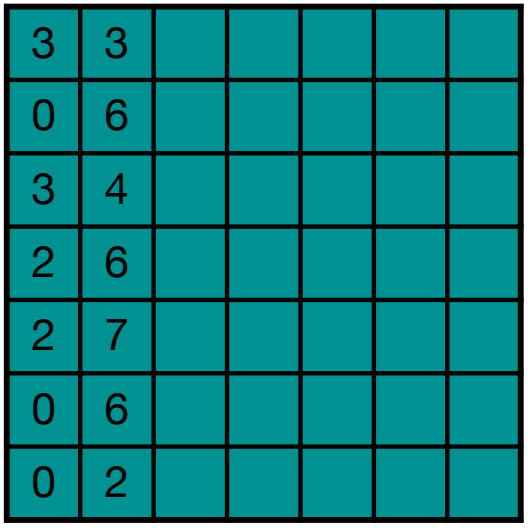

3 3 2 1 0
0 0 1 3 1
1 3 1 2 2 3
0 2 0 0 2 2
-1 2 0 0 1 0 0 1

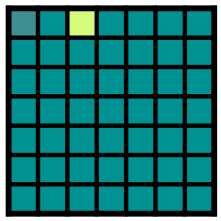



3 3 2 1 0
0 0 1 3 1
3 1 2 2 3
2 0 0 2 2
1 2 1 0 0 0 1
0 2 0
-1 0 1

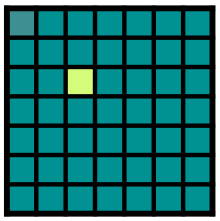



3 3 2 1 0
0 0 1 3 1
3 1 2 2 3
1 2 1 0 0 2 2
0 2 0 0 0 1
-1 0 1

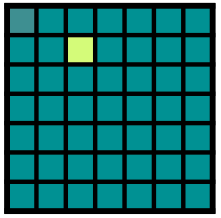

3 3
0 6
3 4
2 6
2 7
0 6
0 2



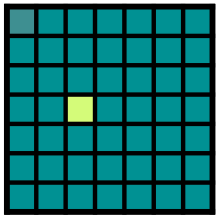
	1	1	0	
	0	2	0	
3_{-1}	3_{0}	2_{1}	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1



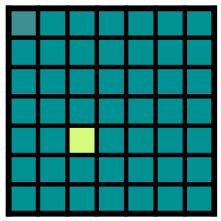
3_{1}	3_{1}	2_{0}	1	0
0_{0}	0_{2}	1_{0}	3	1
3_{-1}	1_{0}	2_{1}	2	3
2	0	0	2	2
2	0	0	0	1



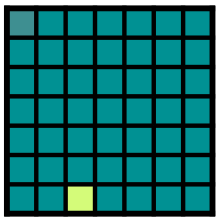
	1	1	0	
3_{0}	3_{2}	2_{0}	1	0
0_{-1}	0_{0}	1_{1}	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1



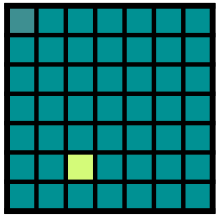
3	3	2	1	0
0_{1}	0_{1}	1_{0}	3	1
3_{0}	1_{2}	2_{0}	2	3
2_{-1}	0_{0}	0_{1}	2	2
2	0	0	0	1



3	3	2	1	0
0	0	1	3	1
3_1	1_1	2_0	2	3
2_0	0_2	0_0	2	2
2_{-1}	0_0	0_1	0	1

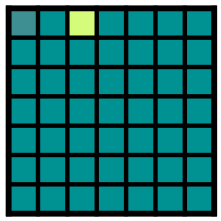


3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2_1	0_1	0_0	0	1
0	2	0		
-1	0	1		

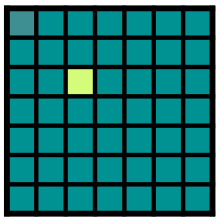


3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2_1	0_1	0_0	2	2
2_0	0_2	0_0	0	1
-1	0	1		

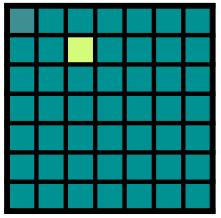
3	3	-1				
0	6	7				
3	4	5				
2	6	0				
2	7	2				
0	6	2				
0	2	2				



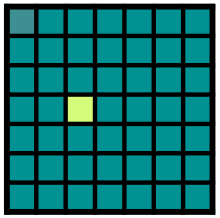
	1	1	0	
	0	2	0	
3_{-1}	3_{0}	2_{1}	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1



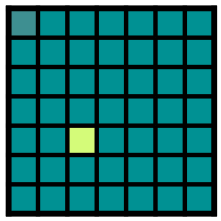
3_{1}	3_{1}	2_{0}	1	0
0_{0}	0_{2}	1_{0}	3	1
3_{-1}	1_{0}	2_{1}	2	3
2	0	0	2	2
2	0	0	0	1



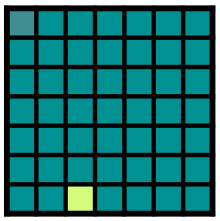
	1	1	0	
3_{0}	3_{2}	2_{0}	1	0
0_{-1}	0_{0}	1_{1}	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1



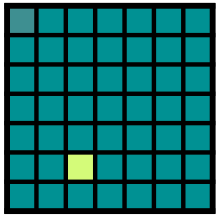
3	3	2	1	0
0_{1}	0_{1}	1_{0}	3	1
3_{0}	1_{2}	2_{0}	2	3
2_{-1}	0_{0}	0_{1}	2	2
2	0	0	0	1



3	3	2	1	0
0	0	1	3	1
3_1	1_1	2_0	2	3
2_0	0_2	0_0	2	2
2_{-1}	0_0	0_1	0	1

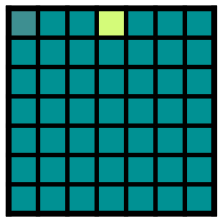


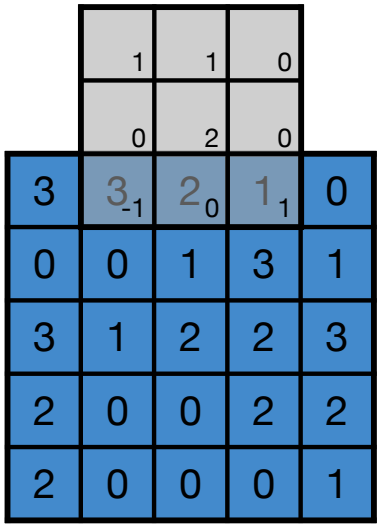
3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2_1	0_1	0_0	0	1
0	2	0		
-1	0	1		

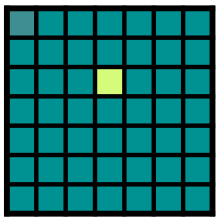


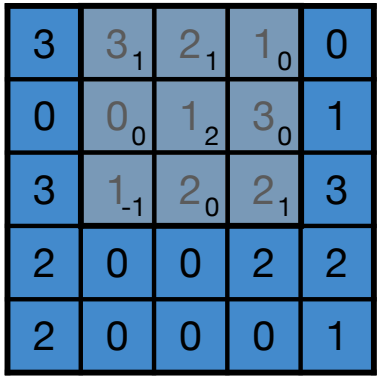
3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2_1	0_1	0_0	2	2
2_0	0_2	0_0	0	1
-1	0	1		

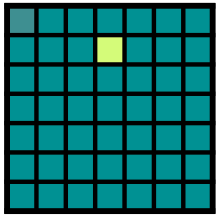
3	3	-1				
0	6	7				
3	4	5				
2	6	0				
2	7	2				
0	6	2				
0	2	2				

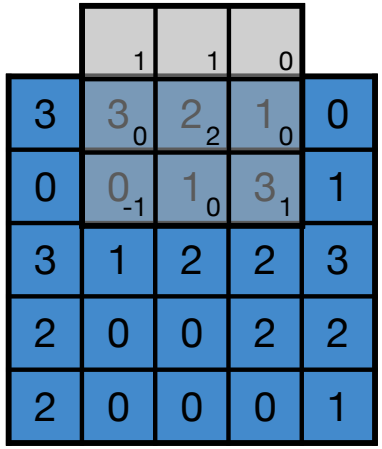


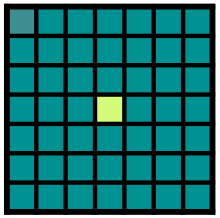

1 1 0
0 2 0
3 3 2 1 0
0 0 1 3 1
3 1 2 2 3
2 0 0 2 2
2 0 0 0 1

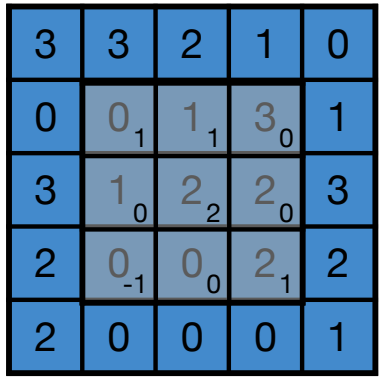


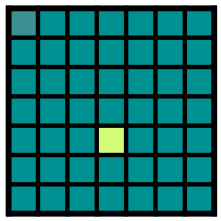

3 3 2 1 0
0 0 1 3 1
3 1 2 2 3
2 0 0 2 2
2 0 0 0 1



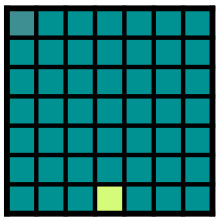

1 1 0
3 3 2 1 0
0 0 1 1 3 1
3 1 2 2 3
2 0 0 2 2
2 0 0 0 1



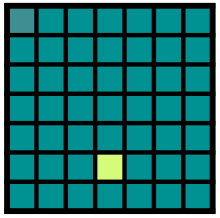

3 3 2 1 0
0 0 1 1 3 1
3 1 2 2 2 3
2 0 0 0 2 2
2 0 0 0 1



3	3	2	1	0
0	0	1	3	1
3	1_1	2_1	2_0	3
2	0_0	0_2	2_0	2
2	0_{-1}	0_0	0_1	1

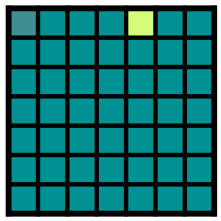


3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0_1	0_1	0_0	1
	0	2	0	
	-1	0	1	

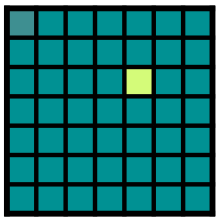


3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0_1	0_1	2_0	2
2	0_0	0_2	0_0	1
	-1	0	1	

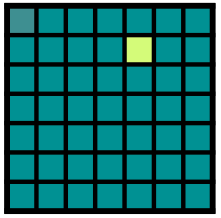
3	3	-1	-2			
0	6	7	7			
3	4	5	8			
2	6	0	7			
2	7	2	3			
0	6	2	0			
0	2	2	0			



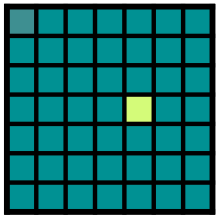
		1	1	0
		0	2	0
3	3	2_{-1}	1_{0}	0_{1}
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1



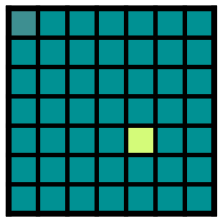
3	3	2_{1}	1_{1}	0_{0}
0	0	1_{0}	3_{2}	1_{0}
3	1	2_{-1}	2_{0}	3_{1}
2	0	0	2	2
2	0	0	0	1



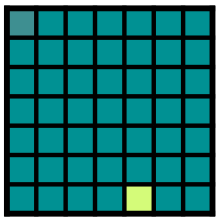
		1	1	0
3	3	2_{0}	1_{2}	0_{0}
0	0	1_{-1}	3_{0}	1_{1}
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1



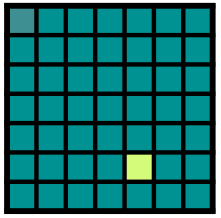
3	3	2	1	0
0	0	1_{1}	3_{1}	1_{0}
3	1	2_{0}	2_{2}	3_{0}
2	0	0_{-1}	2_{0}	2_{1}
2	0	0	0	1



3	3	2	1	0
0	0	1	3	1
3	1	2_1	2_1	3_0
2	0	0_0	2_2	2_0
2	0	0_{-1}	0_0	1_1

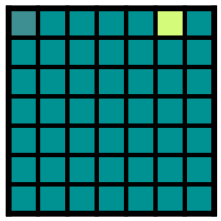


3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0_1	0_1	1_0
		0	2	0
		-1	0	1

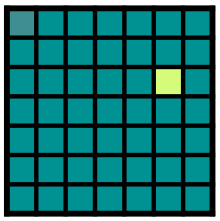


3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0_1	2_1	2_0
2	0	0_0	0_2	1_0
		-1	0	1

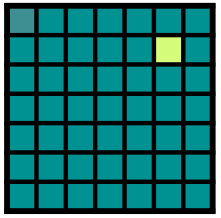
3	3	-1	-2	-2		
0	6	7	7	2		
3	4	5	8	10		
2	6	0	7	9		
2	7	2	3	9		
0	6	2	0	2		
0	2	2	0	0		



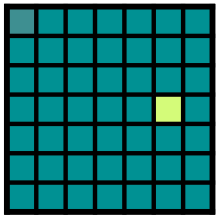
			1	1	0
			0	2	0
3	3	2	1_{-1}	0_{0}	1
0	0	1	3	1	
3	1	2	2	3	
2	0	0	2	2	
2	0	0	0	1	



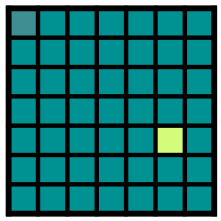
3	3	2	1_{1}	0_{1}	0
0	0	1	3_{0}	1_{2}	0
3	1	2	2_{-1}	3_{0}	1
2	0	0	2	2	
2	0	0	0	1	



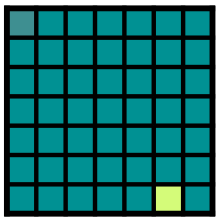
			1	1	0
3	3	2	1_{0}	0_{2}	0
0	0	1	3_{-1}	1_{0}	1
3	1	2	2	3	
2	0	0	2	2	
2	0	0	0	1	



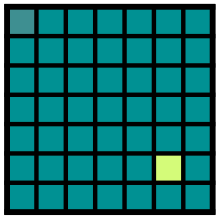
3	3	2	1	0	
0	0	1	3_{1}	1_{1}	0
3	1	2	2_{0}	3_{2}	0
2	0	0	2_{-1}	2_{0}	1
2	0	0	0	1	



3	3	2	1	0
0	0	1	3	1
3	1	2	2_1	3_1
2	0	0	2_0	2_2
2	0	0	0_{-1}	1_0

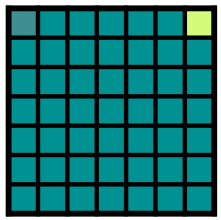


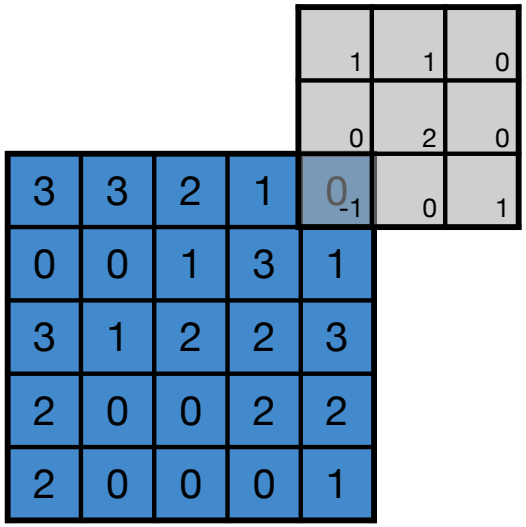
3	3	2	1	0	
0	0	1	3	1	
3	1	2	2	3	
2	0	0	2	2	
2	0	0	0_1	1_1	0
			0	2	0
			-1	0	1

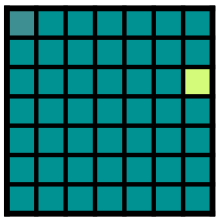


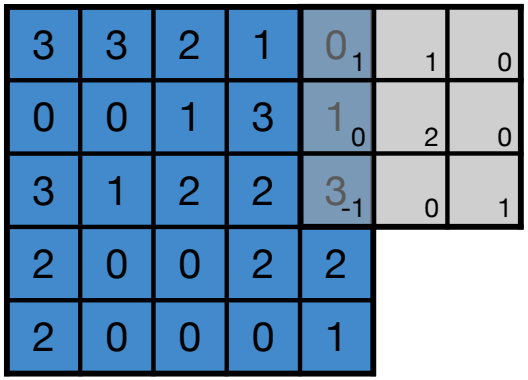
3	3	2	1	0	
0	0	1	3	1	
3	1	2	2	3	
2	0	0	2_1	2_1	0
2	0	0	0_0	1_2	0
			-1	0	1

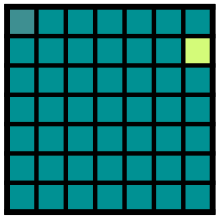
3	3	-1	-2	-2	-1	
0	6	7	7	2	-3	
3	4	5	8	10	1	
2	6	0	7	9	5	
2	7	2	3	9	7	
0	6	2	0	2	6	
0	2	2	0	0	1	

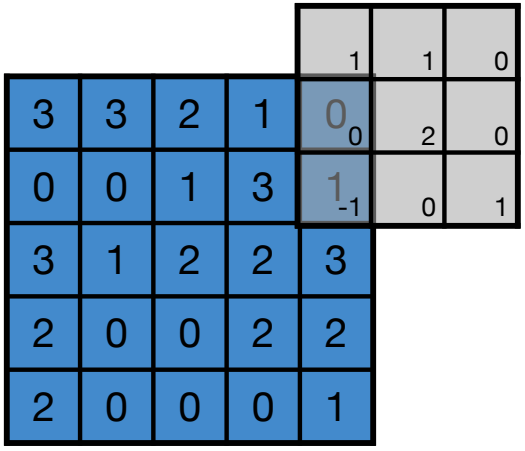


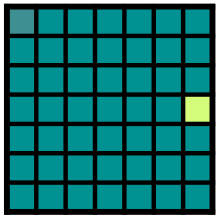

1 1 0
0 2 0
3 3 2 1 0 -1 0 1
0 0 1 3 1
3 1 2 2 3
2 0 0 2 2
2 0 0 0 1

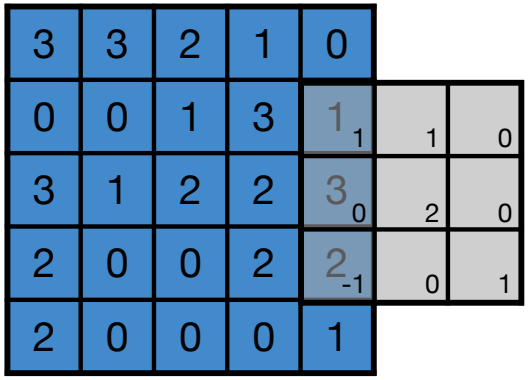


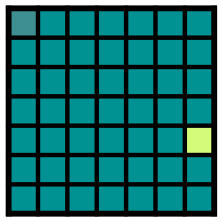

3 3 2 1 0 1 1 0
0 0 1 3 1 0 2 0
3 1 2 2 3 -1 0 1
2 0 0 2 2
2 0 0 0 1




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



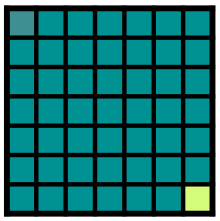

3 3 2 1 0
0 0 1 3 1 1 0
3 1 2 2 3 0 2 0
2 0 0 2 2 -1 0 1
2 0 0 0 1



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

1	0	
2	0	
0	1	

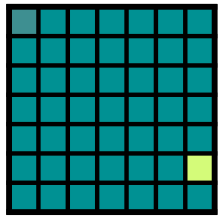
3		
2		
1		



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

1	0	
2	0	
0	1	

1		
0		
-1		



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

2	0	
1	0	
0	1	

2		
1		
0		

3	3	-1	-2	-2	-1	0
0	6	7	7	2	-3	-1
3	4	5	8	10	1	-3
2	6	0	7	9	5	-1
2	7	2	3	9	7	2
0	6	2	0	2	6	2
0	2	2	0	0	1	1

Convolución 2D

- Hacer el cálculo vía correlaciones resulta en cambio mucho más simple y es más consistente con la operación que habíamos hecho anteriormente.

$$z[n,m] = x * y[n,m] = \sum_{k,l=-\infty}^{\infty} x[k,l]y[n-k,m-l] \quad \forall n,m$$

3	3	-1	-2	-2	-1	0
0	6	7	7	2	-3	-1
3	4	5	8	10	1	-3
2	6	0	7	9	5	-1
2	7	2	3	9	7	2
0	6	2	0	2	6	2
0	2	2	0	0	1	1

z

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

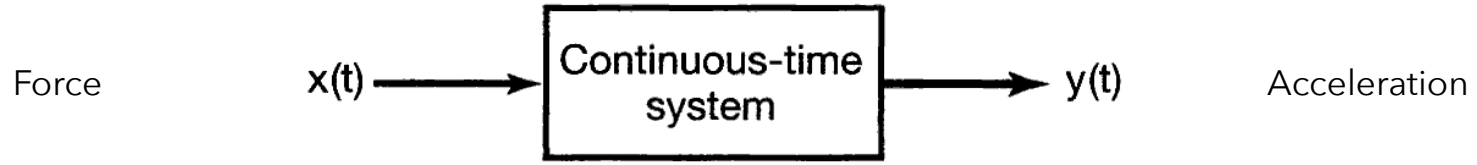
x

1	0	-1
0	2	0
0	1	1

y

Sistemas

- Transformaciones más generales se modelan como la acción de un cierto **sistema** sobre la señal. Por ejemplo (*)

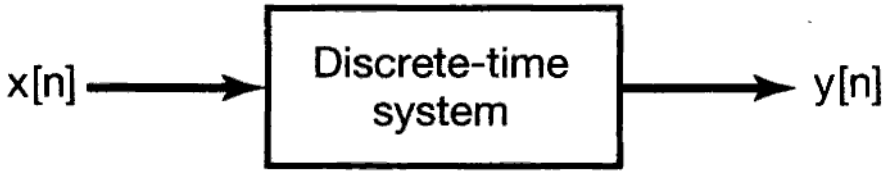


$$\frac{dv(t)}{dt} = \frac{1}{m}[f(t) - \rho v(t)]$$

(*) Oppenheim et al. (1996) Signals and Systems, MIT Press.

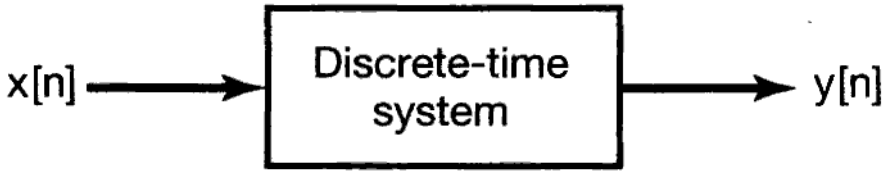
Sistemas

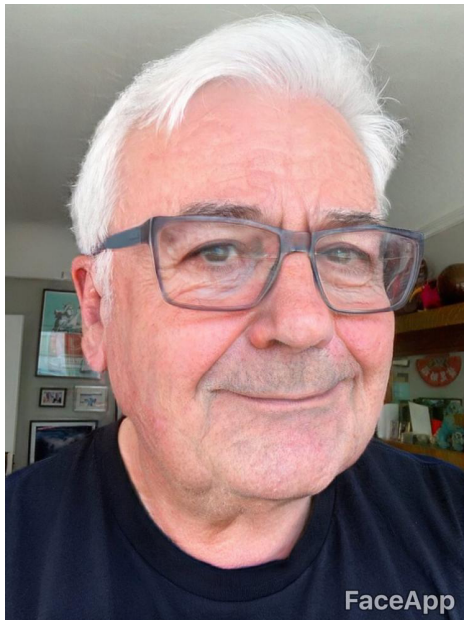
- Transformaciones más generales se modelan como la acción de un cierto **sistema** sobre la señal. Por ejemplo:



Sistemas

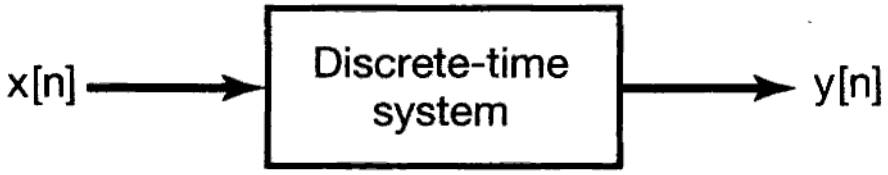
- Transformaciones más generales se modelan como la acción de un cierto **sistema** sobre la señal. Por ejemplo:





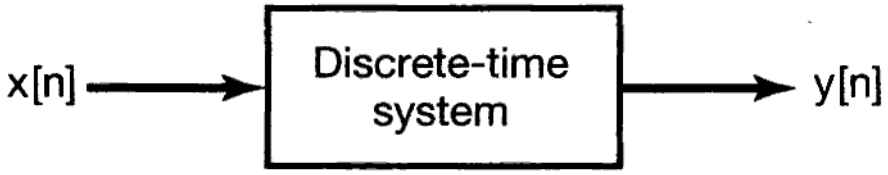
Sistemas

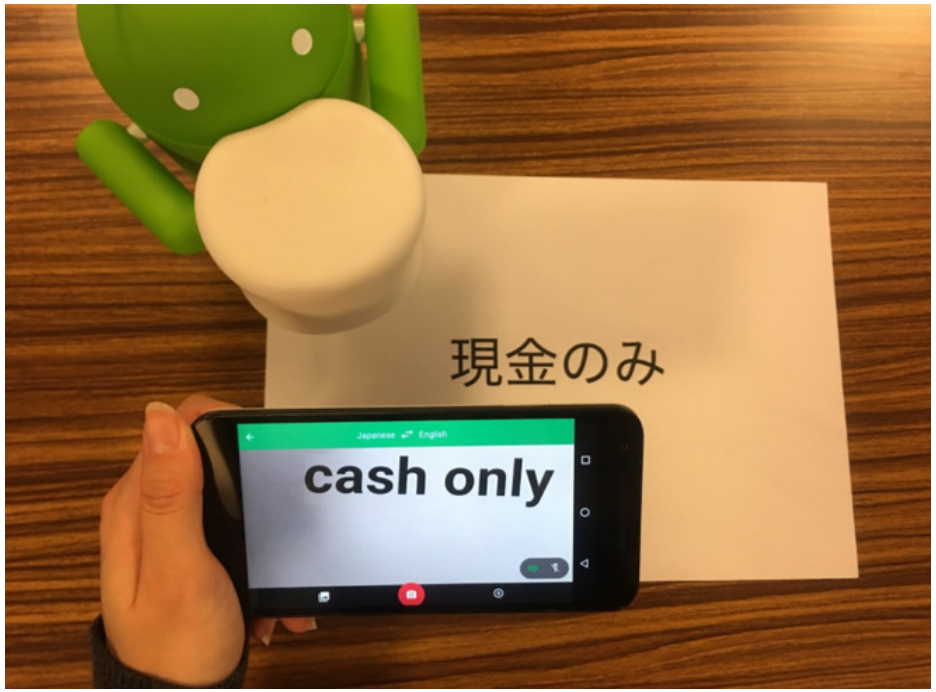
- Transformaciones más generales se modelan como la acción de un cierto **sistema** sobre la señal. Por ejemplo:



Sistemas

- Transformaciones más generales se modelan como la acción de un cierto **sistema** sobre la señal. Por ejemplo:





Sistemas Lineales Invariantes

- Una clase de sistemas que será relativamente fácil modelar o usar como aproximación de sistemas reales será la familia de los **sistemas lineales invariantes en el tiempo (LTI).**
- Si un sistema transforma $x_1[n]$ en $y_1[n]$ y $x_2[n]$ en $y_2[n]$, el sistema se denomina **lineal** cuando
    - Es **aditivo**, es decir la respuesta ante $x_1[n] + x_2[n]$ es $y_1[n] + y_2[n]$.
    - Es **homogéneo**, es decir la respuesta ante $\alpha \cdot x_1[n]$ es $\alpha \cdot y_1[n]$ para cualquier $\alpha \in \mathbb{R}$.
- En palabras: un sistema es lineal si la acción sobre una superposición de señales es a superposición de las respuestas.

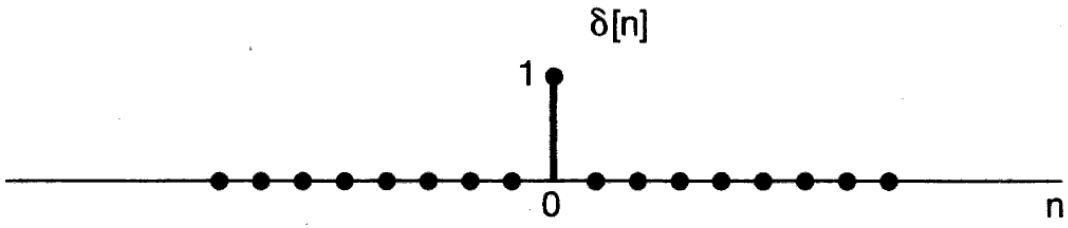
Sistemas Lineales Invariantes

- Si un sistema transforma $x[n]$ en $y[n]$, el sistema se denomina **invariante en el tiempo** cuando la respuesta frente a $x_1[n - n_0]$ es $y_1[n - n_0]$ y la respuesta frente a $x_1[n + n_0]$ es $y_1[n + n_0]$.
- En palabras: un retraso o adelanto en una señal que entra al sistema sólo causa el mismo retraso o adelanto en la señal que sale del sistema.
- Es importante notar que la linealidad de un sistema no implica su invarianza en el tiempo ni viceversa (e.g. $y(t) = t \cdot x(t)$).
- **Nuestro objetivo es modelar la acción de una neurona como la de un sistema LTI (seguida de una no-linealidad). Este tipo de neuronas permitirán manipular señales y aprender patrones de variación en 1 o más dimensiones.**

Impulso Unitario

- Una señal fundamental para describir la acción de un sistema LTI sobre una señal es el denominado **impulso unitario.**

$$\delta[n] = \begin{cases} 1 & n = 0 \\ 0 & n \neq 0 \end{cases}$$

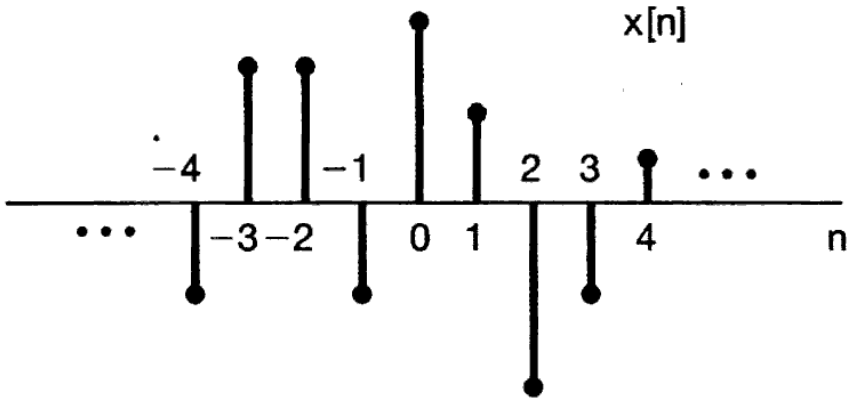


(*) En el caso continuo se trata de límite de un impulso de soporte infinitesimal centrado en 0 y con área 1.

Representación de una Señal

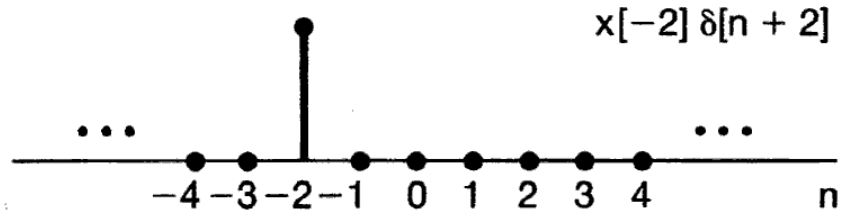
- La clave es que una señal se puede ver como una **superposición de impulsos** unitarios retardados o adelantados y apropiadamente escalados (en amplitud).

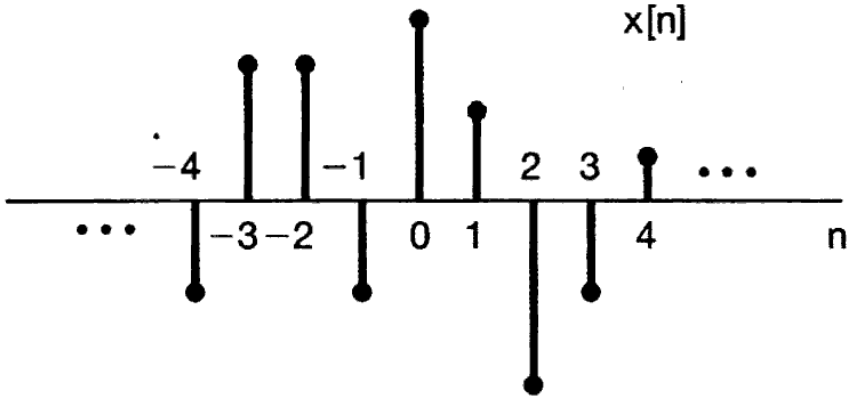
$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n-k]$$



Representación de una Señal

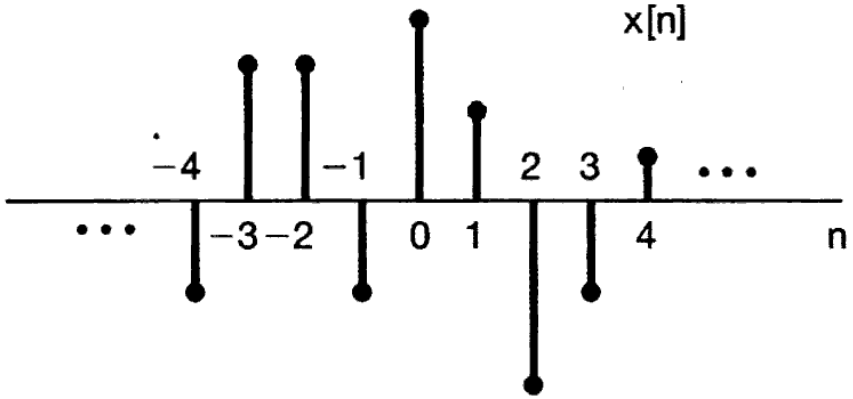
- Por ejemplo, adelantando el impulso unitario 2 tiempos y escalando por la magnitud de x[-2], obtenemos una señal que es equivalente a muestrear x en n=-2.





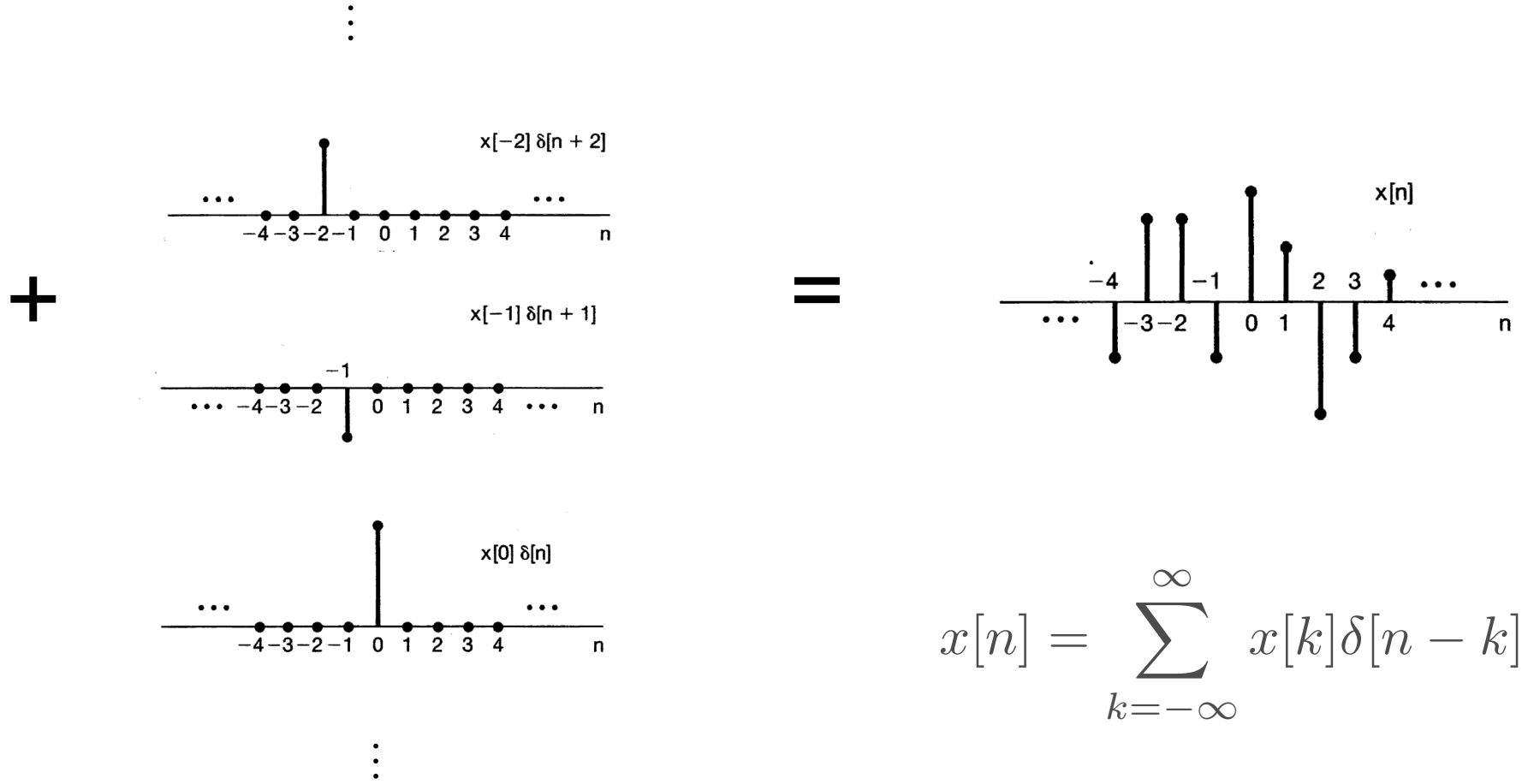
Representación de una Señal

- Por ejemplo, retardando el impulso unitario 1 tiempo y escalando por la magnitud de x[1], obtenemos una señal que es equivalente a muestrear x en n=1.



Representación de una Señal

- Sumando las señales correspondientes a todos los muestreos obtenemos la señal original:



$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n-k]$$

Representación de una Señal

- Tenemos entonces que una señal se puede ver como una **superposición de impulsos** unitarios:

$$x[n] = \sum_{k=-\infty}^{\infty} x[k]\delta[n-k]$$

- Equivalentemente, una señal se puede escribir siempre como la convolución del impulso unitario con ella misma!

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

Representación de Sistemas LTI

- La observación anterior nos permitirá representar de manera compacta la acción de un sistema LTI sobre una señal cualquiera.
- En efecto, si definimos $h_k[n]$ como la respuesta del sistema ante un impulso unitario adelantado k unidades en el tiempo $\delta[n-k]$, **obtenemos por linealidad** que la acción del sistema sobre una señal cualquiera $x[n] = \sum_k x[k]h_k[n]$ se puede escribir como

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

Es decir, como superposición de las respuestas del sistema a los impulsos unitarios retardados o adelantados.

Representación de Sistemas LTI

- La ventaja de todo esto aparece cuando explotamos la propiedad de invarianza temporal. Como $\delta[n-k]$ es sólo un adelanto o retardo del impulso unitario, tenemos que $h_k[n] = h_0[n-k]$, lo que nos permite escribir

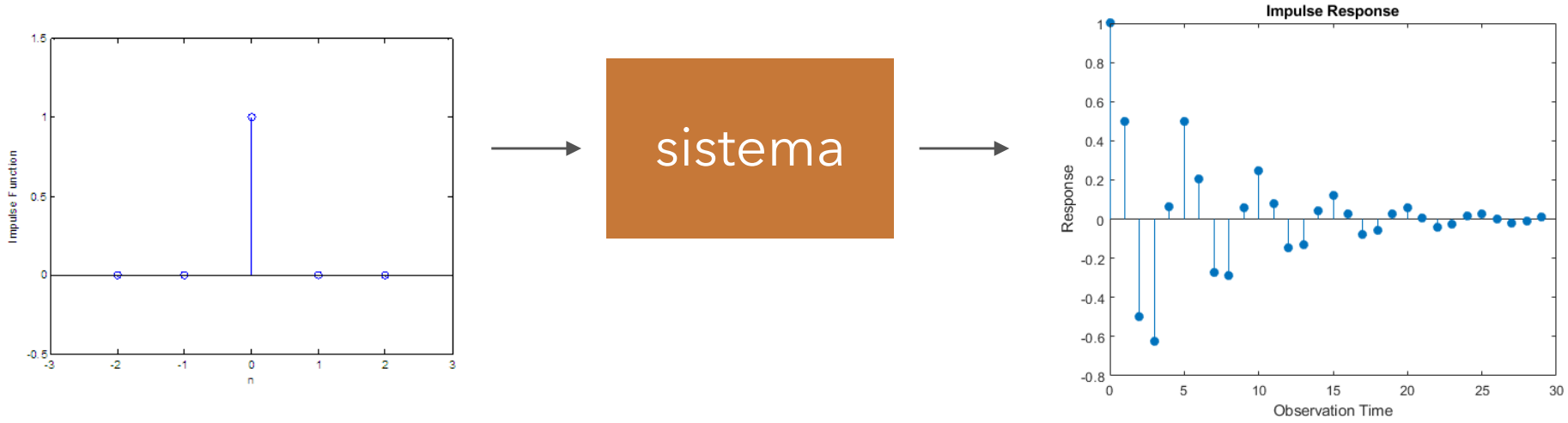
$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h_0[n-k] \quad \forall n$$

Es decir, para conocer la acción de un sistema LTI sobre cualquier señal, sólo necesitamos saber cómo responde a un impulso unitario!

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

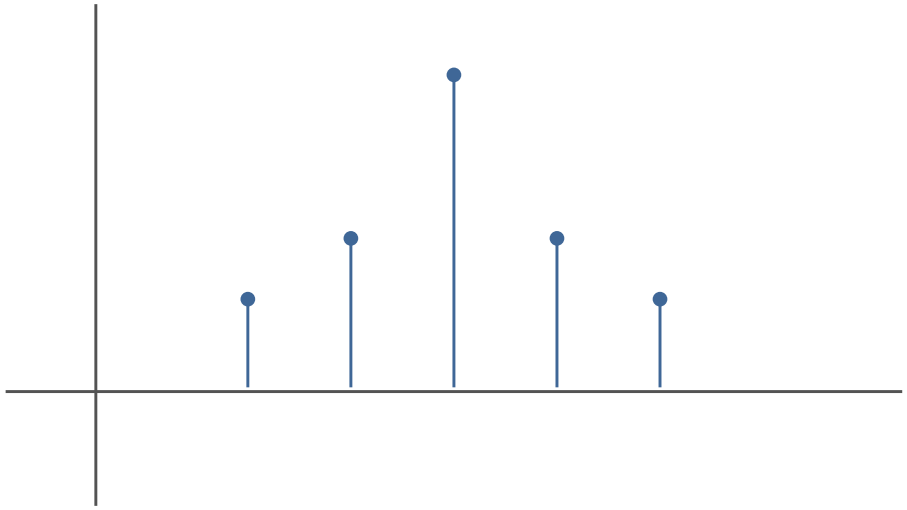
Representación de Sistemas LTI

Para conocer la acción de un sistema LTI sobre cualquier señal, sólo necesitamos saber cómo responde a un impulso unitario!



Representación de Sistemas LTI

- Cuando nuestro objetivo sea identificar/aproximar/aprender un sistema, lo anterior significa que **sólo necesitamos estimar el kernel.**



tarea mucho más sencilla que estimar a respuesta del sistema frente aa cualquier tipo de señal.

Representación de Sistemas LTI

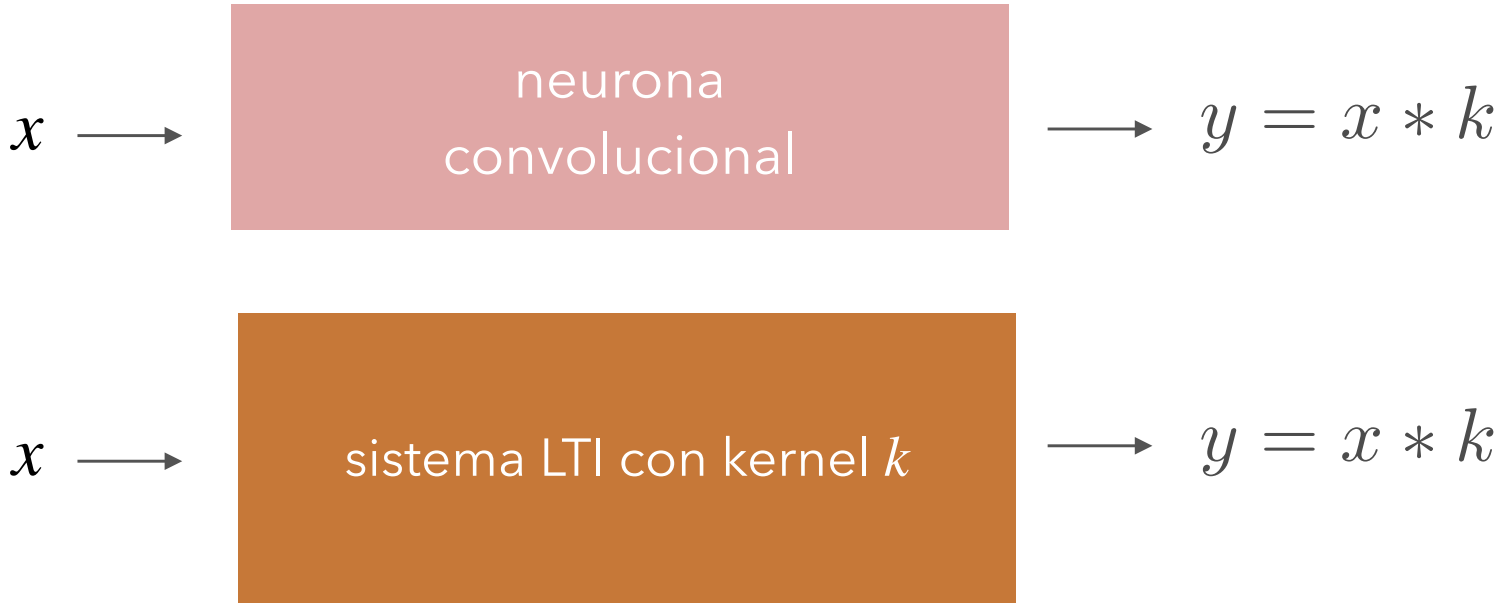
- Ahora, si examinamos la ecuación

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

notamos que se trata simplemente de una convolución $y = x * k$. Dado un kernel k, la acción del sistema sobre cualquier señal se obtiene **convolucionando el input con el kernel.**

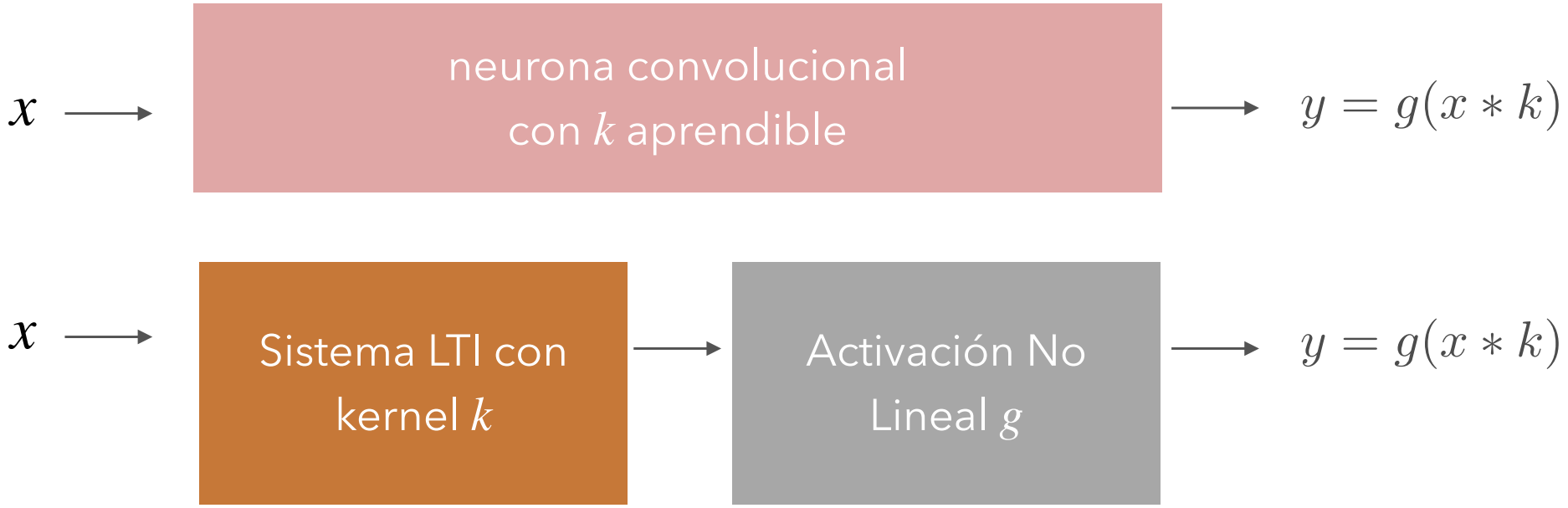
Neuronas Convolucionales?

- ¿Qué tal si modelamos la acción de una neurona como la de un sistema LTI $y = x * k$? Después de todo, sólo necesitamos sustituir la multiplicación matricial por otra operación lineal.



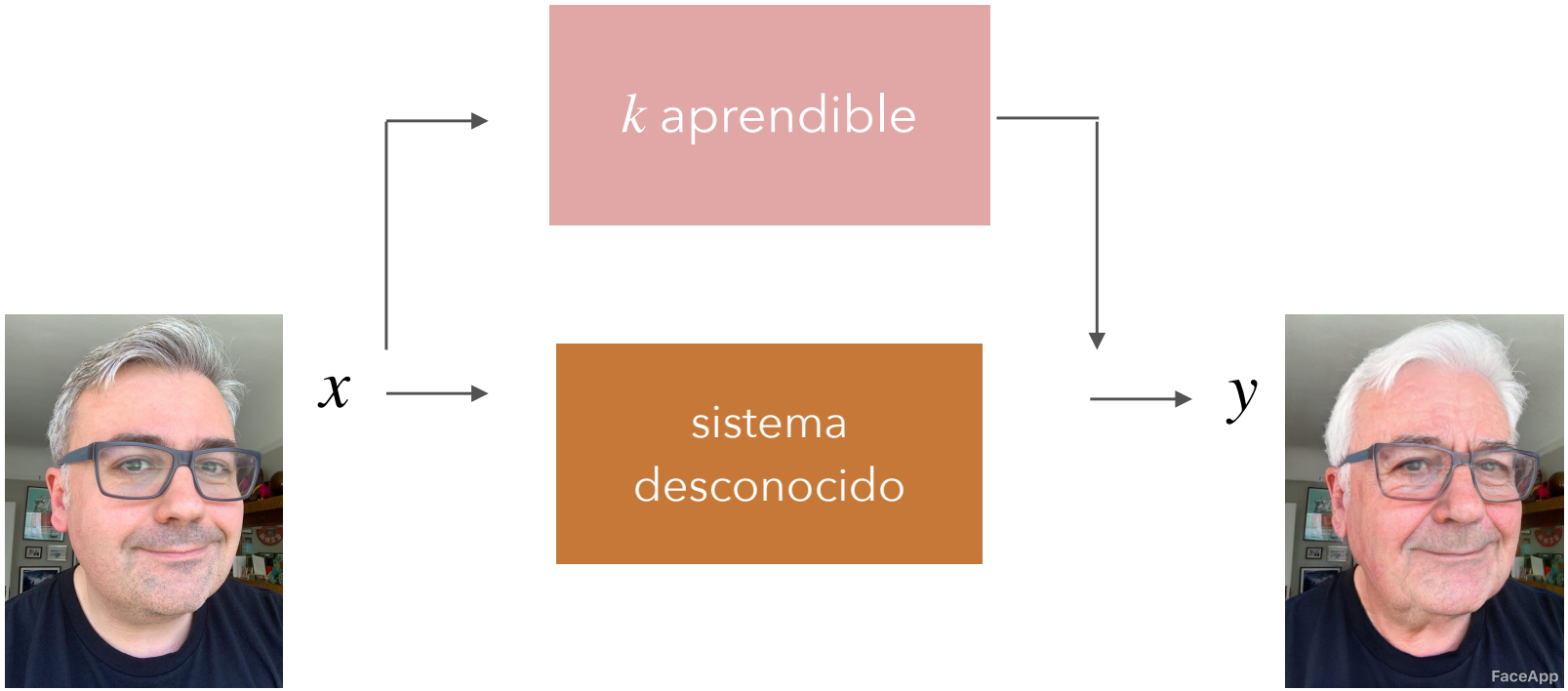
Neuronas Convolucionales?

- Para preservar la capacidad de una red neuronal de aproximar funciones no-lineales, sería importante mantener la estructura lineal + no-lineal de las neuronas tradicionales. Podríamos introducir además un "bias".



Neuronas Convolucionales?

- Para preservar la capacidad de una red neuronal de aproximar funciones no-lineales, sería importante mantener la estructura lineal + no-lineal de las neuronas tradicionales.



Entonces ...

- Los volúmenes de datos que hemos manipulado en capítulos anteriores se pueden abstraer matemáticamente como señales: funciones de una variable independiente que pensamos contienen un patrón de variación que nos gustaría modelar.
- La operación de convolución es una operación que permite transformar una señal a partir de otra señal. Una forma más general de modelar una señal es imaginarla como el resultado la acción de un cierto sistema LTI sobre una señal elemental como el impulso unitario.
- La importancia de la convolución radica en que si conocemos el kernel de un sistema LTI, es decir, su respuesta ante un impulso unitario, podemos conocer su acción sobre cualquier otra señal convolucionando esta señal con el kernel.
- La convolución de una señal con un cierto kernel o filtro se puede calcular vía corrimientos de la señal y escalamientos dictados por el kernel o vía correlaciones del kernel volteado con la señal.