

---

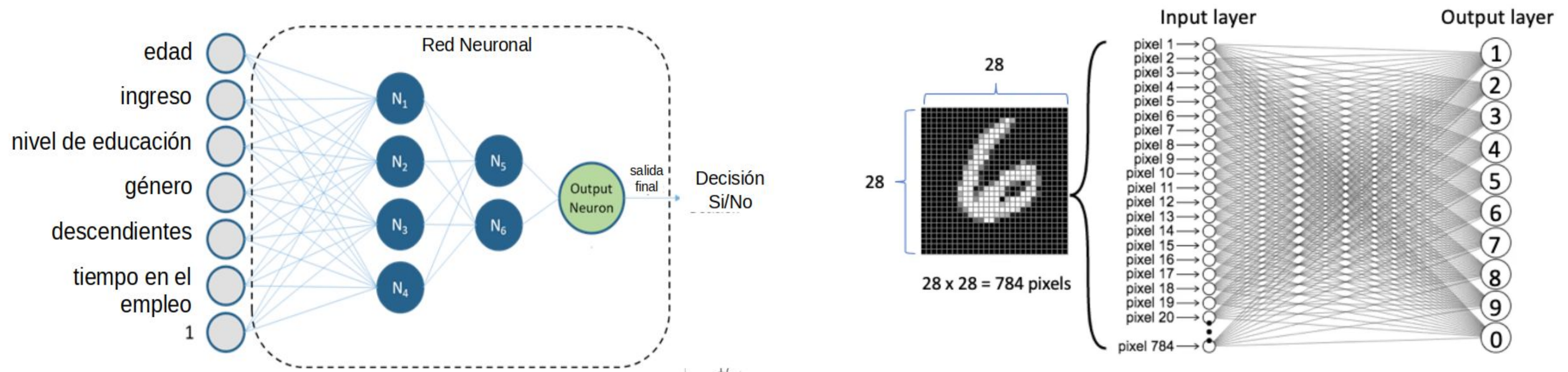
# Nociones de Aprendizaje Automático para Aplicaciones Nucleares

Redes neuronales Convolucionales

---

# Introducción

- Ya vimos como funcionaban las redes totalmente conectadas (un poco).
- Mencionamos que podían usarse como aproximadores universales y las usamos principalmente con datos representados como vectores donde las características no tienen relación espacial o un orden específico.
- En el notebook se hizo una red para clasificar MNIST, pero también señalamos que para imágenes no eran lo ideal, ¿por qué?



[Fuente](#)



# Introducción

---

- Cuando se utilizan redes totalmente conectadas con imágenes, cada píxel se considera de forma independiente, lo que implica perder la información espacial crucial entre píxeles adyacentes.
- Aunque es posible usarlas para imágenes, no son la opción más eficiente.
- Además, el número de parámetros crece mucho con el tamaño de la imagen. Por ejemplo, para procesar las imágenes MNIST de 28x28 píxeles en escala de grises, la pequeña red que mostramos tenía alrededor de 23,000 parámetros.
- Las redes convolucionales surgieron para abordar estas limitaciones, inspirándose en el funcionamiento del sistema visual biológico, donde se procesan patrones locales y relaciones espaciales.

# Un poco de historia

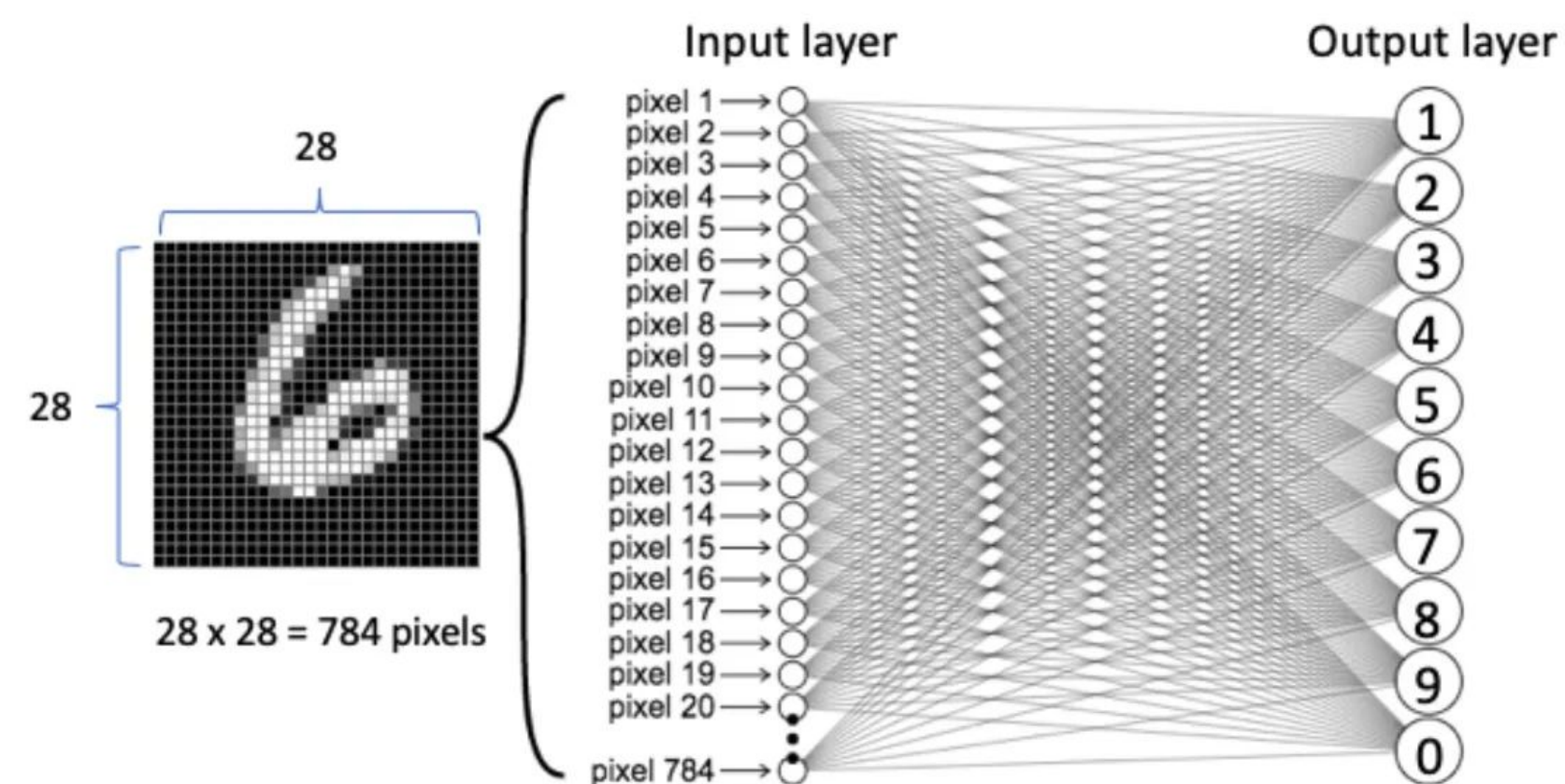
---

- **1958 - 1968.** Hubel & Wiesel (Nobel 1981). Estudio de la corteza visual en gatos y monos.
  - Las neuronas parecían estar organizadas en capas.
  - Muchas neuronas de la corteza visual tienen un **pequeño campo receptivo local**.
  - Algunas neuronas reaccionan sólo a imágenes de líneas horizontales, mientras que otras reaccionan sólo a líneas con orientaciones diferentes (dos neuronas pueden tener el mismo campo receptivo pero reaccionan a orientaciones de línea diferentes).
  - Algunas neuronas tienen **campos receptivos más amplios** y reaccionan a **patrones más complejos**, que son combinaciones de los patrones de nivel inferior.
- **1980.** Primeros intentos de RNA basadas en este tipo de arquitectura.
- **1998,** LeCun. LeNet-5. Capas convolucionales y capas de agrupamiento.
- Años **2000** y posteriores.
  - Rendimiento sobrehumano en algunas tareas visuales complejas.
  - Utilizado en otras tareas: reconocimiento de voz y procesamiento del lenguaje natural

# Idea general

## Arquitectura *Fully connected*:

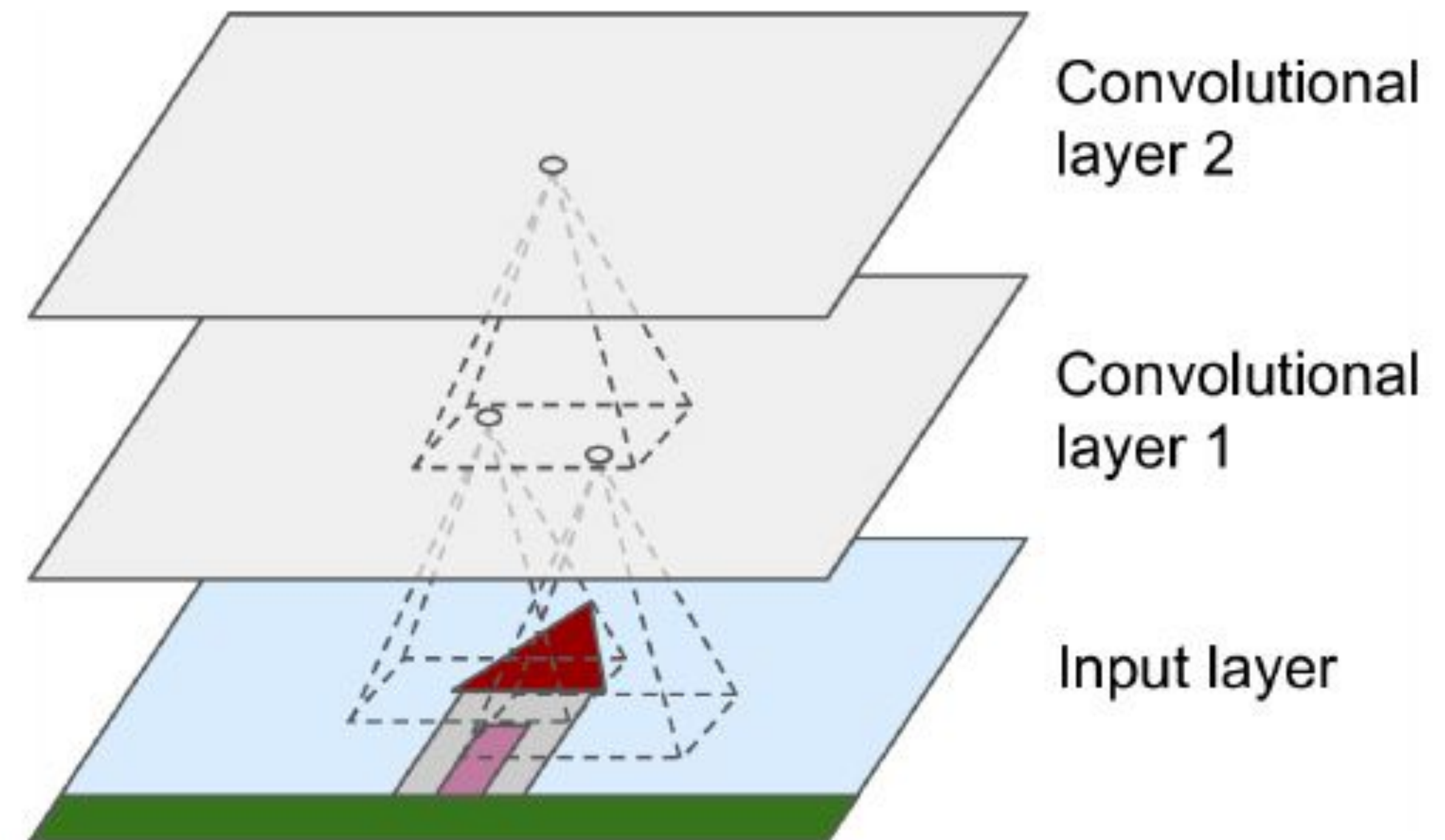
Cada unidad está conectada con todas las anteriores y las próximas



Se aplanla la entrada

## Arquitectura convolucional:

Las unidades en una capa son sensibles a un pequeño campo receptivo en la capa anterior

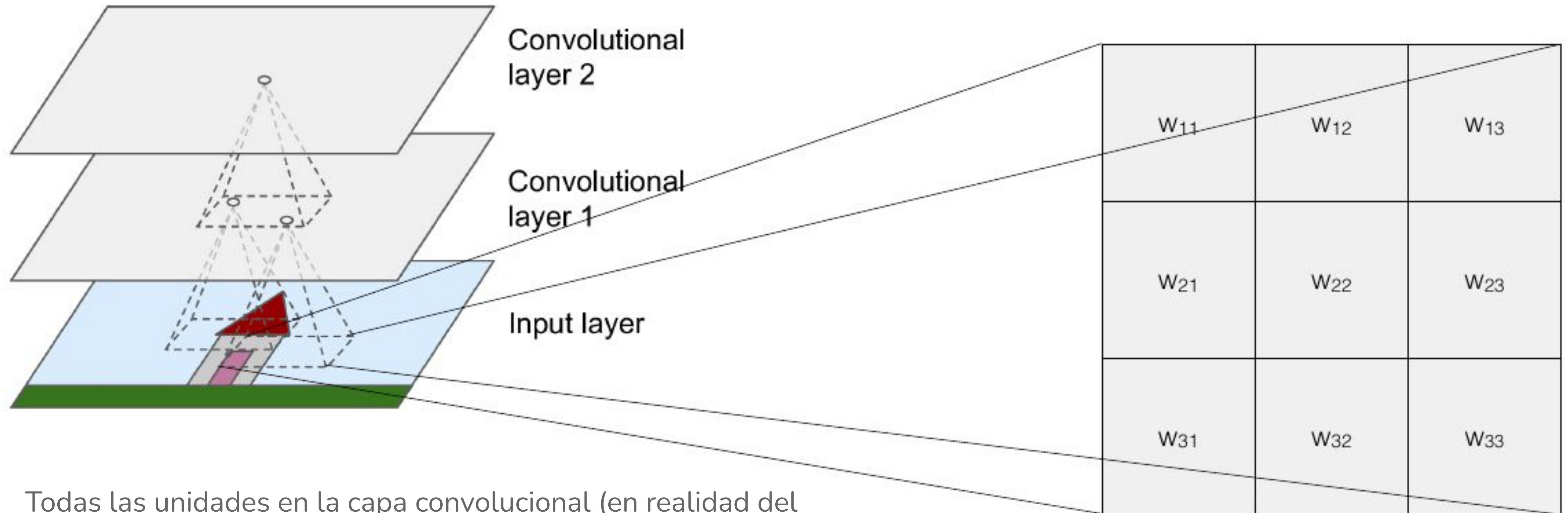


Se mantiene la estructura 2D

Imagen del Gerón



# Idea general



Todas las unidades en la capa convolucional (en realidad del filtro) comparten pesos (i.e., hay relativamente un número pequeño de pesos por cada capa).

**Pesos ajustables**

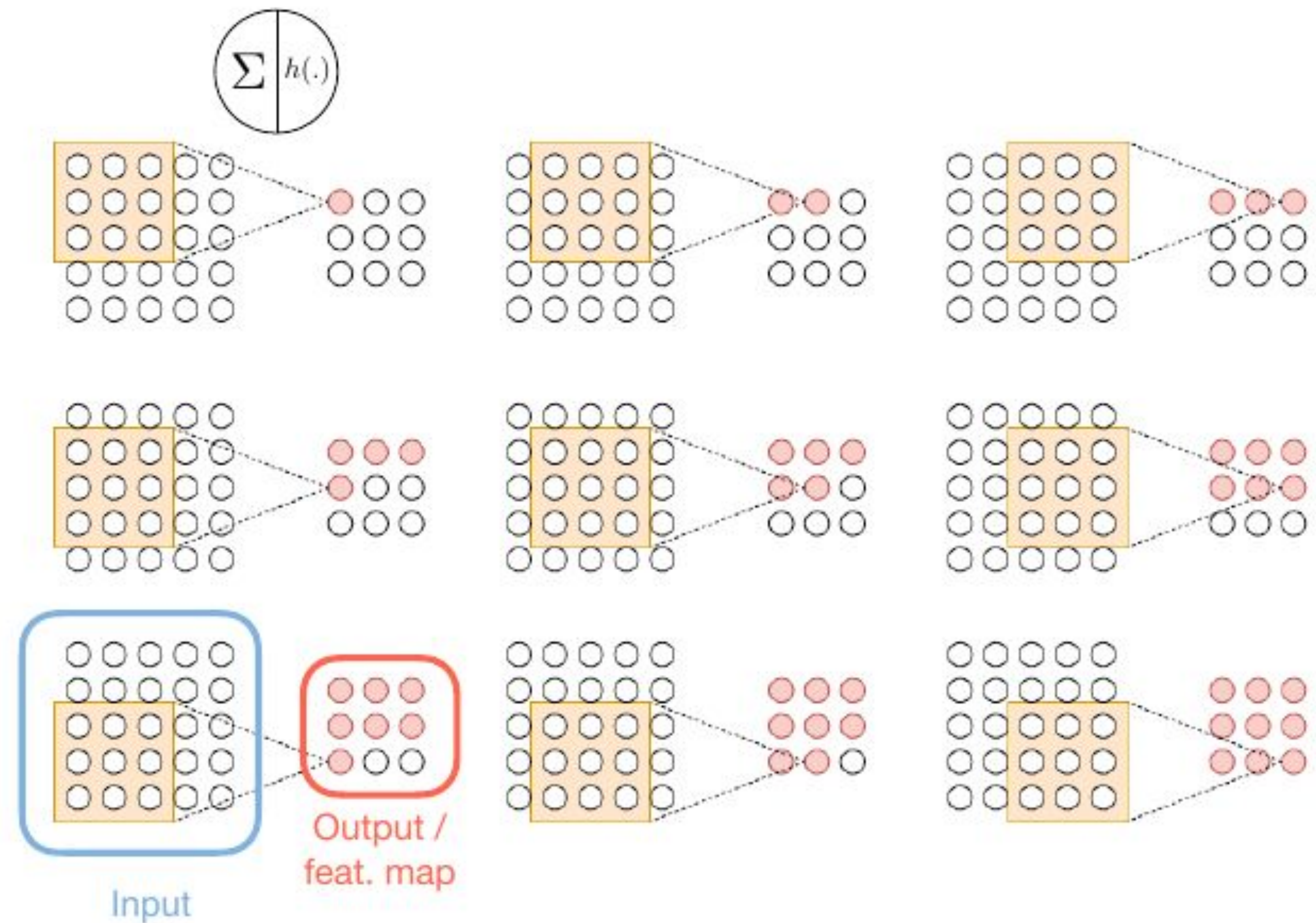
Imagen del Gerón

# Idea general

La salida es calculada pasando el *kernel* a través de la imagen (o de la capa anterior) para producir un mapa de características o *Feature Map*

$W_{11}$	$W_{12}$	$W_{13}$
$W_{21}$	$W_{22}$	$W_{23}$
$W_{31}$	$W_{32}$	$W_{33}$

Pesos ajustables





# Convolución

1	0	1	1	0	1	1	0	1	0
0	1	0	0	1	1	0	0	1	0
1	0	1	1	0	0	0	1	1	0
1	1	0	1	1	1	1	0	1	0
1	0	1	1	0	1	0	0	0	1
1	1	0	1	0	1	0	1	0	0
0	0	0	0	1	1	0	0	1	0
0	0	0	1	1	1	1	0	1	1
0	0	1	1	0	1	0	1	1	1
0	1	1	1	0	1	1	0	1	1

tomemos una imagen

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

vamos a convoluir un pedazo



# Convolución

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

\*



1	-1	1
-1	1	-1
1	-1	1

elegimos un filtro  
(núcleo/Kernel) que  
extrae formas de  
cruz

=

?	?	?
?	?	?
?	?	?

Convolución

(no es un producto de matrices)

# Convolución

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

\*



1	-1	1
-1	1	-1
1	-1	1

elegimos un filtro  
(núcleo/Kernel) que  
extrae formas de  
cruz

=

?		

$$\begin{aligned} &1 \times 1 + 0 \times (-1) + 1 \times 1 + \\ &0 \times (-1) + 1 \times 1 + 0 \times (-1) + \\ &1 \times 1 + 0 \times (-1) + 1 \times 1 = 5 \end{aligned}$$

Convolución

(no es un producto de matrices)



# Convolución

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

\*

1	-1	1
-1	1	-1
1	-1	1

=

5		

elegimos un filtro  
(núcleo/Kernel) que  
extrae formas de  
cruz

Convolución

(no es un producto de matrices)

# Convolución

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

\*



1	-1	1
-1	1	-1
1	-1	1

elegimos un filtro  
(núcleo/Kernel) que  
extrae formas de  
cruz

=

5	?	

Convolución

(no es un producto de matrices)



# Convolución

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

\*



1	-1	1
-1	1	-1
1	-1	1

elegimos un filtro  
(núcleo/Kernel) que  
extrae formas de  
cruz

=

5	?	

$$0 \times 1 + 1 \times (-1) + 1 \times 1 + \\ 1 \times (-1) + 0 \times 1 + 0 \times (-1) + \\ 0 \times 1 + 1 \times (-1) + 1 \times 1 = -1$$

Convolución

(no es un producto de matrices)

# Convolución

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

\*

1	-1	1
-1	1	-1
1	-1	1

=

5	-1	

elegimos un filtro  
(núcleo/Kernel) que  
extrae formas de  
cruz

Convolución

(no es un producto de matrices)



# Convolución

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

\*

1	-1	1
-1	1	-1
1	-1	1

=

5	-1	-1

elegimos un filtro  
(núcleo/Kernel) que  
extrae formas de  
cruz

Convolución

(no es un producto de matrices)

# Convolución

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

\*



1	-1	1
-1	1	-1
1	-1	1

elegimos un filtro  
(núcleo/Kernel) que  
extrae formas de  
cruz

=

5	-1	-1
-3		

Convolución

(no es un producto de matrices)



# Convolución

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

\*

1	-1	1
-1	1	-1
1	-1	1

=

5	-1	-1
-3	3	

elegimos un filtro  
(núcleo/Kernel) que  
extrae formas de  
cruz

Convolución

(no es un producto de matrices)

# Convolución

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

\*

1	-1	1
-1	1	-1
1	-1	1

=

5	-1	-1
-3	3	1

elegimos un filtro  
(núcleo/Kernel) que  
extrae formas de  
cruz

Convolución

(no es un producto de matrices)



# Convolución

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

\*



1	-1	1
-1	1	-1
1	-1	1

elegimos un filtro  
(núcleo/Kernel) que  
extrae formas de  
cruz

=

5	-1	-1
-3	3	1
4		

Convolución

(no es un producto de matrices)

# Convolución

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

\*

1	-1	1
-1	1	-1
1	-1	1

=

5	-1	-1
-3	3	1
4	-2	

elegimos un filtro  
(núcleo/Kernel) que  
extrae formas de  
cruz

Convolución

(no es un producto de matrices)



# Convolución

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

\*

1	-1	1
-1	1	-1
1	-1	1

=

5	-1	-1
-3	3	1
4	-2	0

elegimos un filtro  
(núcleo/Kernel) que  
extrae formas de  
cruz

Convolución

(no es un producto de matrices)

# Convolución

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

\*

1	-1	1
-1	1	-1
1	-1	1

elegimos un filtro  
(núcleo/Kernel) que  
extrae formas de  
cruz

=

5	-1	-1
-3	3	1
4	-2	0

Reconoce una  
cruz

Convolución

(no es un producto de matrices)



# Convolución

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

\*

1	-1	1
-1	1	-1
1	-1	1

elegimos un filtro  
(núcleo/Kernel) que  
extrae formas de  
cruz

=

5	-1	-1
-3	3	1
4	-2	0







También si es  
similar a una cruz

Convolución

(no es un producto de matrices)

# Convolución

- Un filtro busca un patrón particular en la imagen y también puede detectar cosas similares
- Tiene invarianza traslacional, lo puede encontrar en cualquier parte
- Ese fue un filtro específico para buscar cruces, hay muchos para hacer diferentes cosas.
- Muy lindo pero ¿dónde entra lo *automático*?

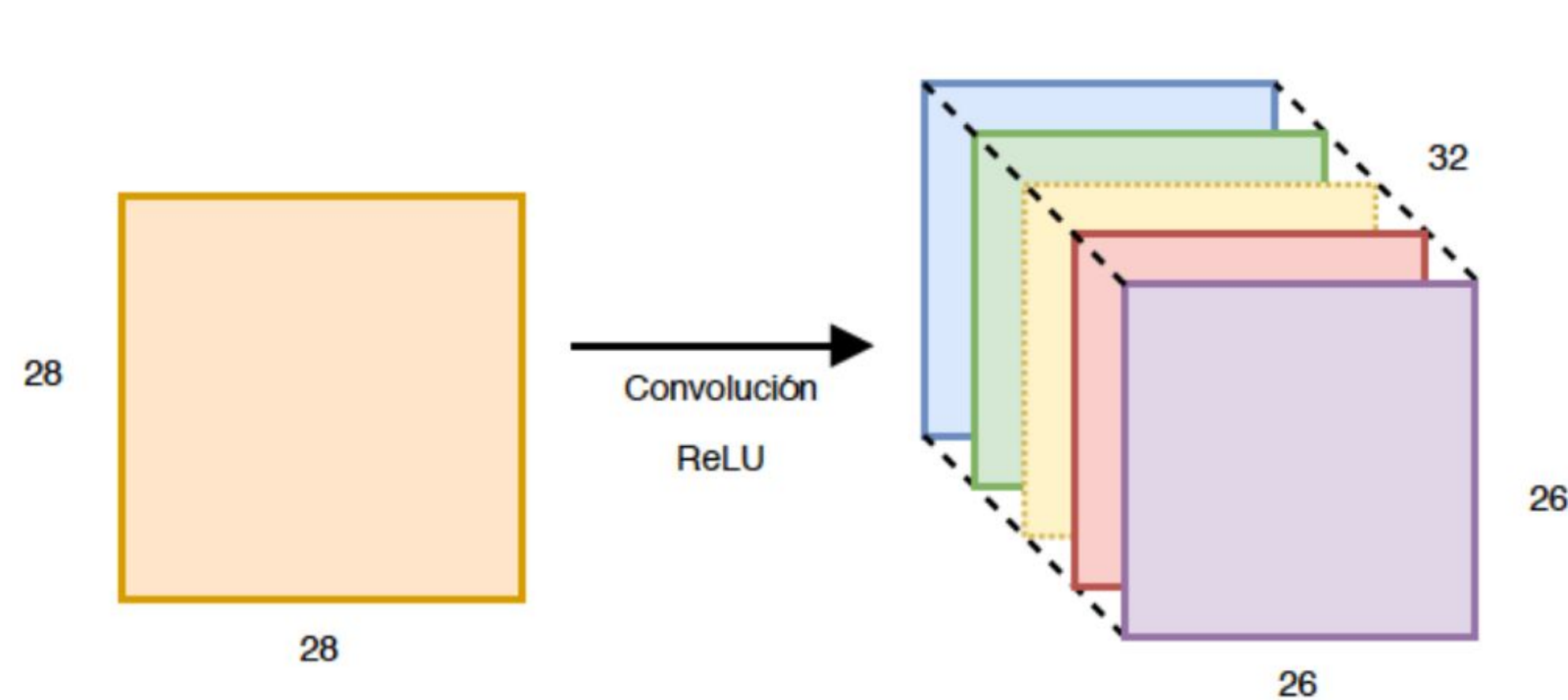
Operación	Núcleo	Imagen resultante
Identidad	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Detección de bordes	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Enfocar	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Desenfoque de cuadro (normalizado)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

[https://es.wikipedia.org/wiki/Núcleo\\_\(procesamiento\\_digital\\_de\\_imágenes\)](https://es.wikipedia.org/wiki/Núcleo_(procesamiento_digital_de_imágenes))

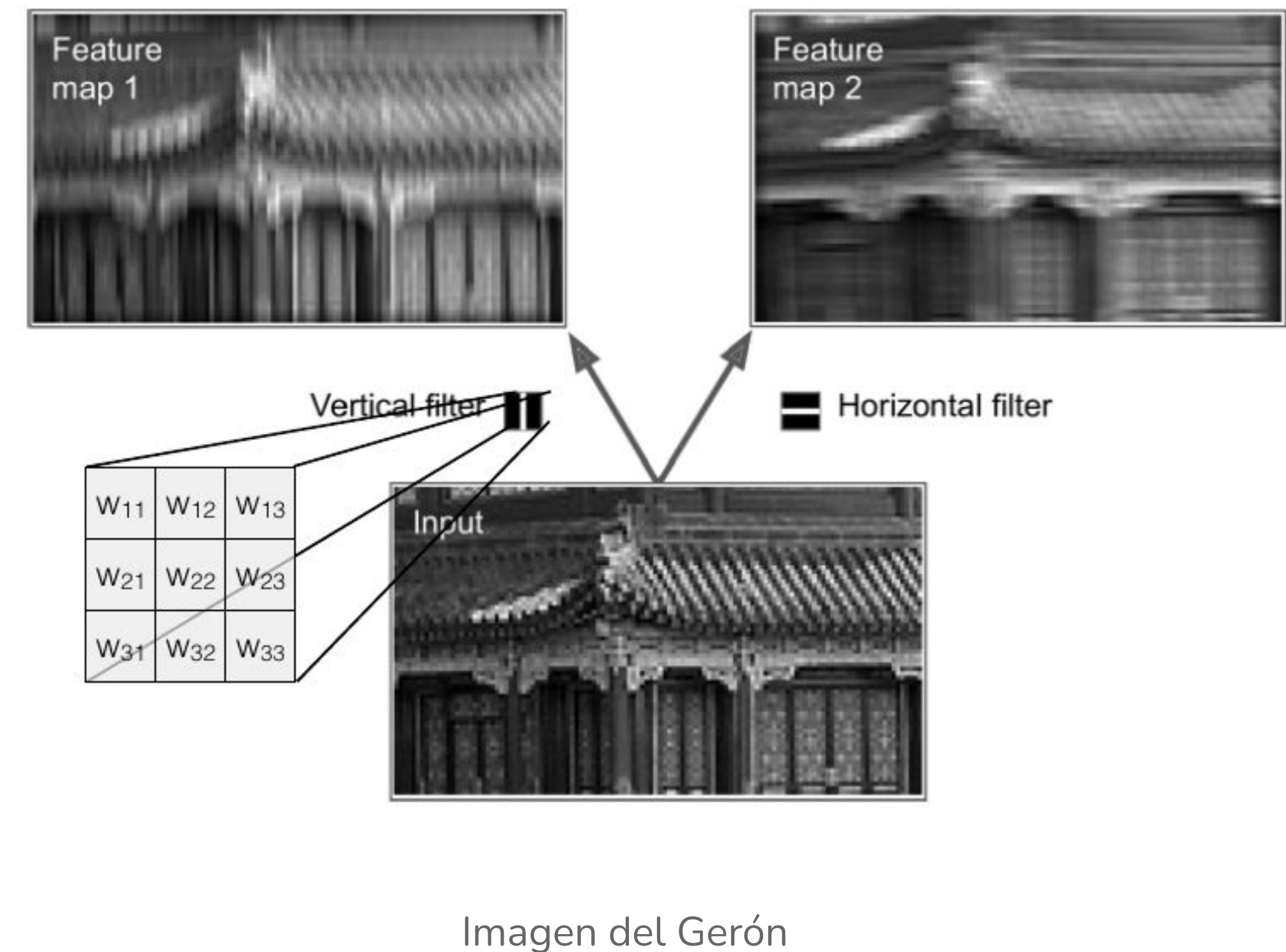


# Arquitectura convolucional

- El chiste es que no le vamos a dar los valores que van en los filtros.
- Le vamos a definir cosas como el tamaño o la cantidad, pero los valores que van dentro de cada uno los va a aprender la red.



Por ejemplo, acá se usó un kernel de 3x3 y se le dijo que genere 32 filtros.



# Arquitectura convolucional

- Se busca que los diferentes filtros en las diferentes capas se combinen para alcanzar características de más alto nivel.
- En las primeras capas se pueden detectar patrones simples como líneas rectas o inclinadas, pero a medida que se profundiza se combinan y detecten cosas más complejas.

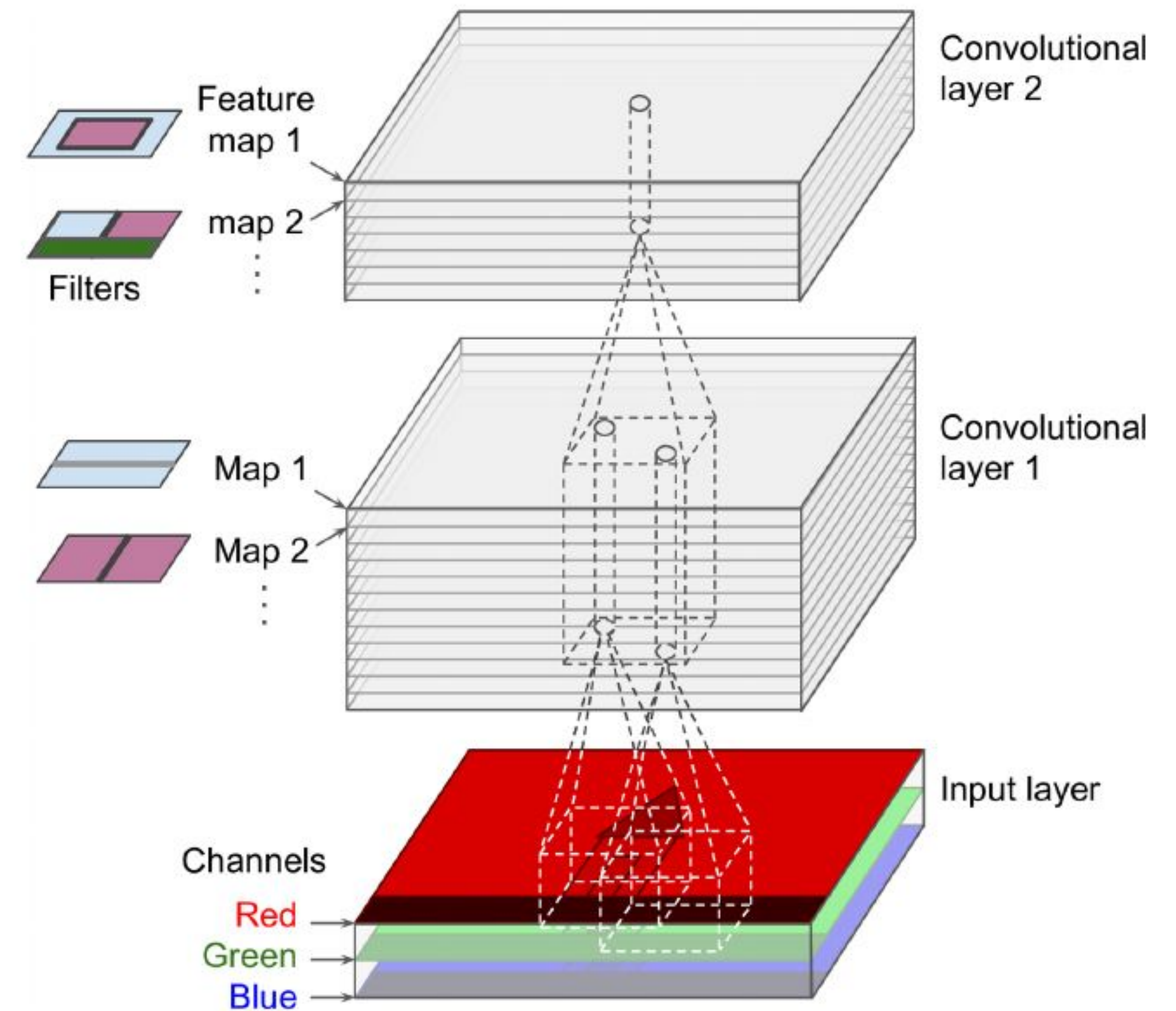
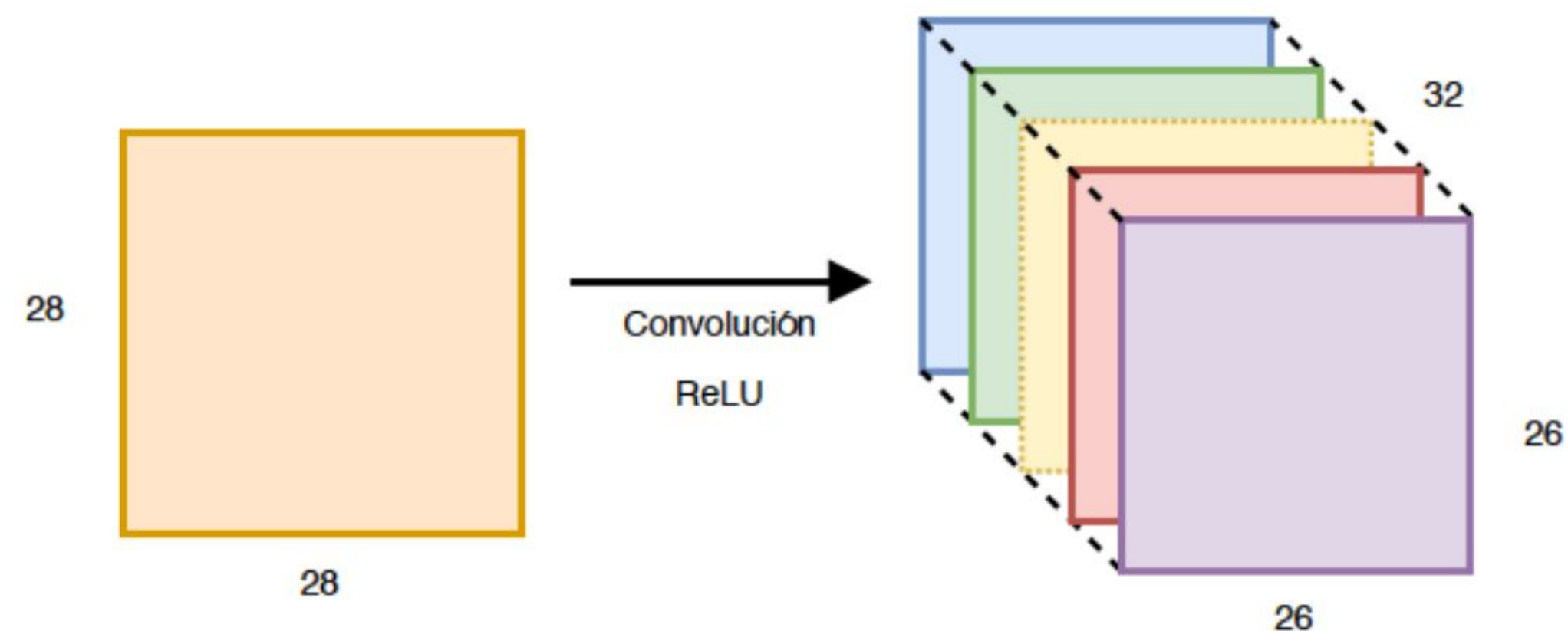


Imagen del Gerón

[¡Redes Neuronales CONVOLUCIONALES! ¿Cómo funcionan? | DotCSV](#)

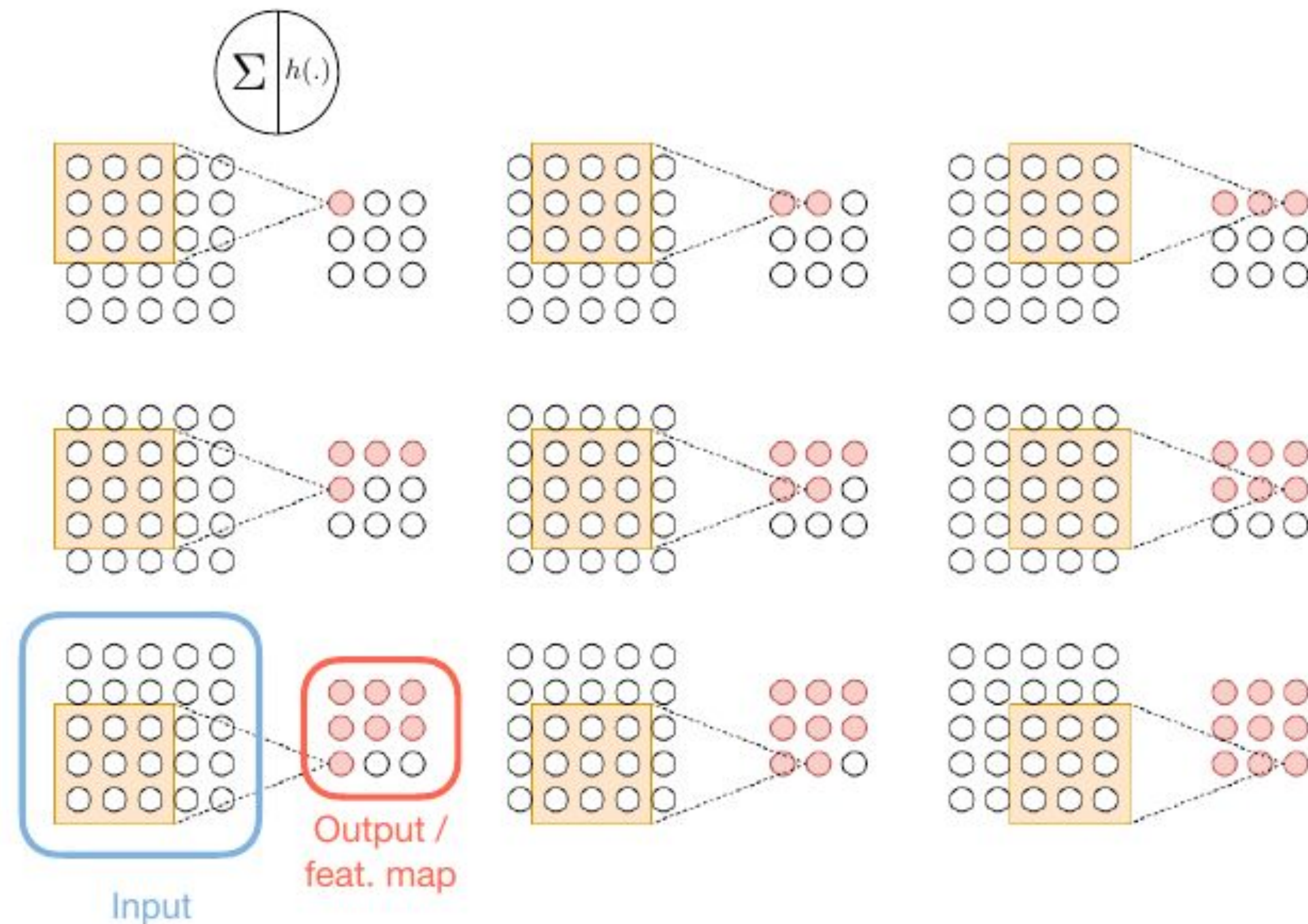


# Arquitectura convolucional

---

Resumiendo...

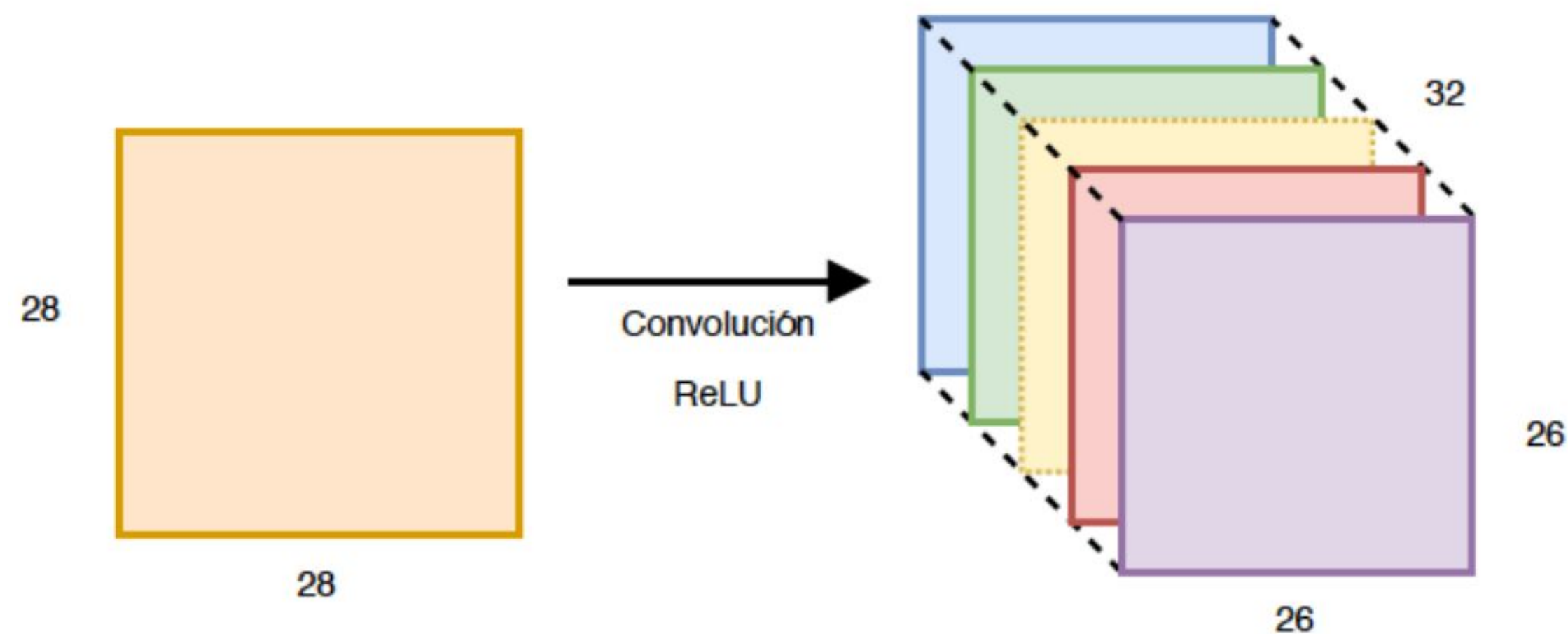
# Arquitectura convolucional



- Se pasa un **kernel** por la imagen (o la capa anterior).
- Los pesos que tenga esa matriz definen un **filtro** (estos pesos son los parámetros entrenables de siempre).
- El resultado de aplicar esa convolución sobre toda la capa anterior generan un **mapa de características**.
- El tamaño del kernel lo definimos nosotros y es el mismo para toda la capa.



# Arquitectura convolucional



- Cada uno de esos mapas de características son el resultado de aplicar un filtro.
- Todos esos mapas de características juntos son una **capa**.
- La cantidad de filtros de la capa (ergo, de mapas de características) es otro de los hiperparámetros definidos por nosotros.

# Arquitectura convolucional

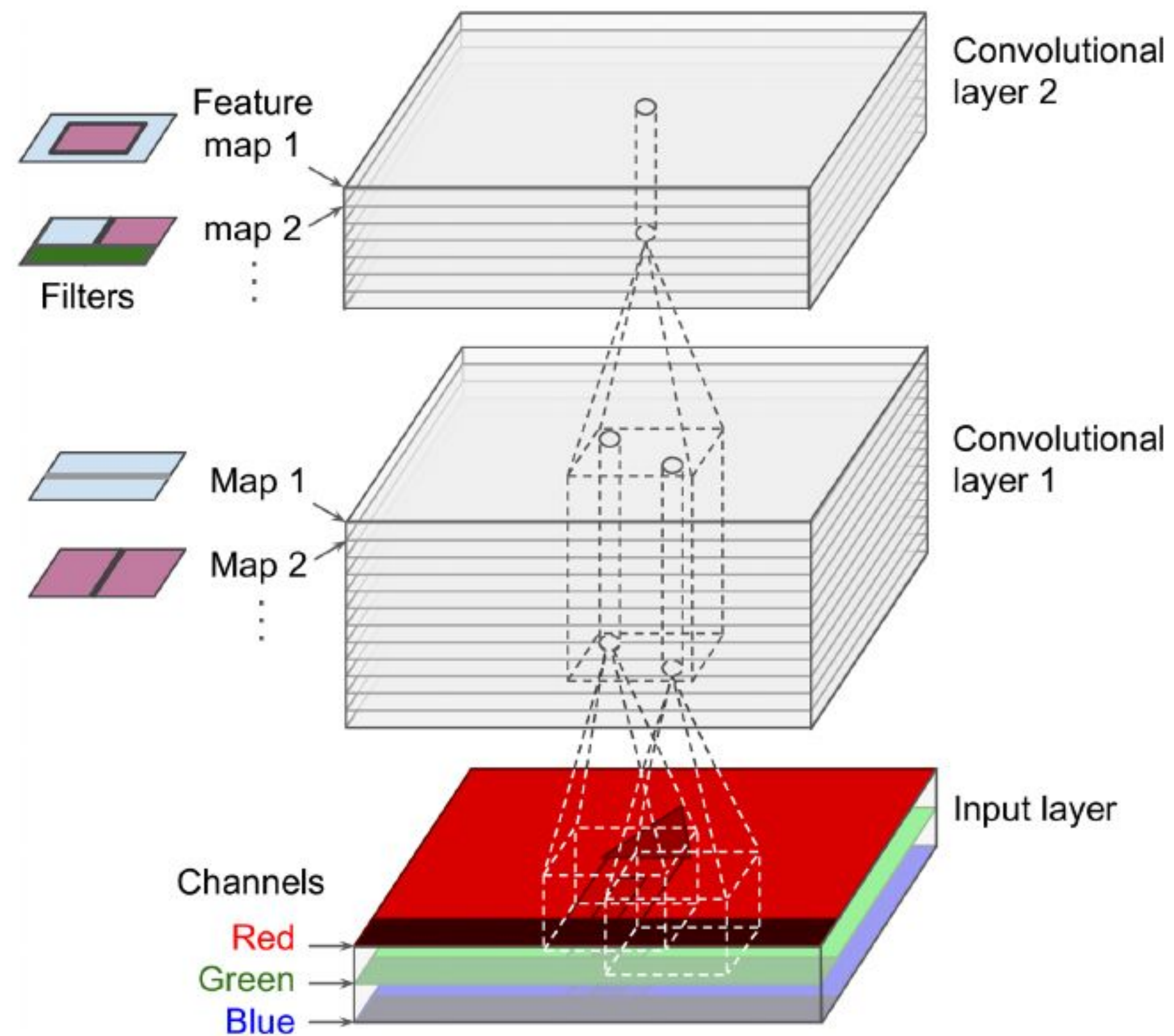


Imagen del Gerón

- La convolución que se realiza sobre la capa anterior se hace sobre toda la profundidad
  - Si es sobre la entrada se hace sobre todos los canales (se aplica la convolución sobre cada canal y se suman).
  - Si es sobre una capa convolucional sobre todos los mapas de características

Piensen en todo lo que “ve” una neurona en la capa 2 vs. una en la capa 1



# Padding

- Si se quiere mantener la dimensión de la entrada se puede rellenar (padding).
- Puede ser útil para no perder detalles en los bordes.
- *Valid* y *same*.

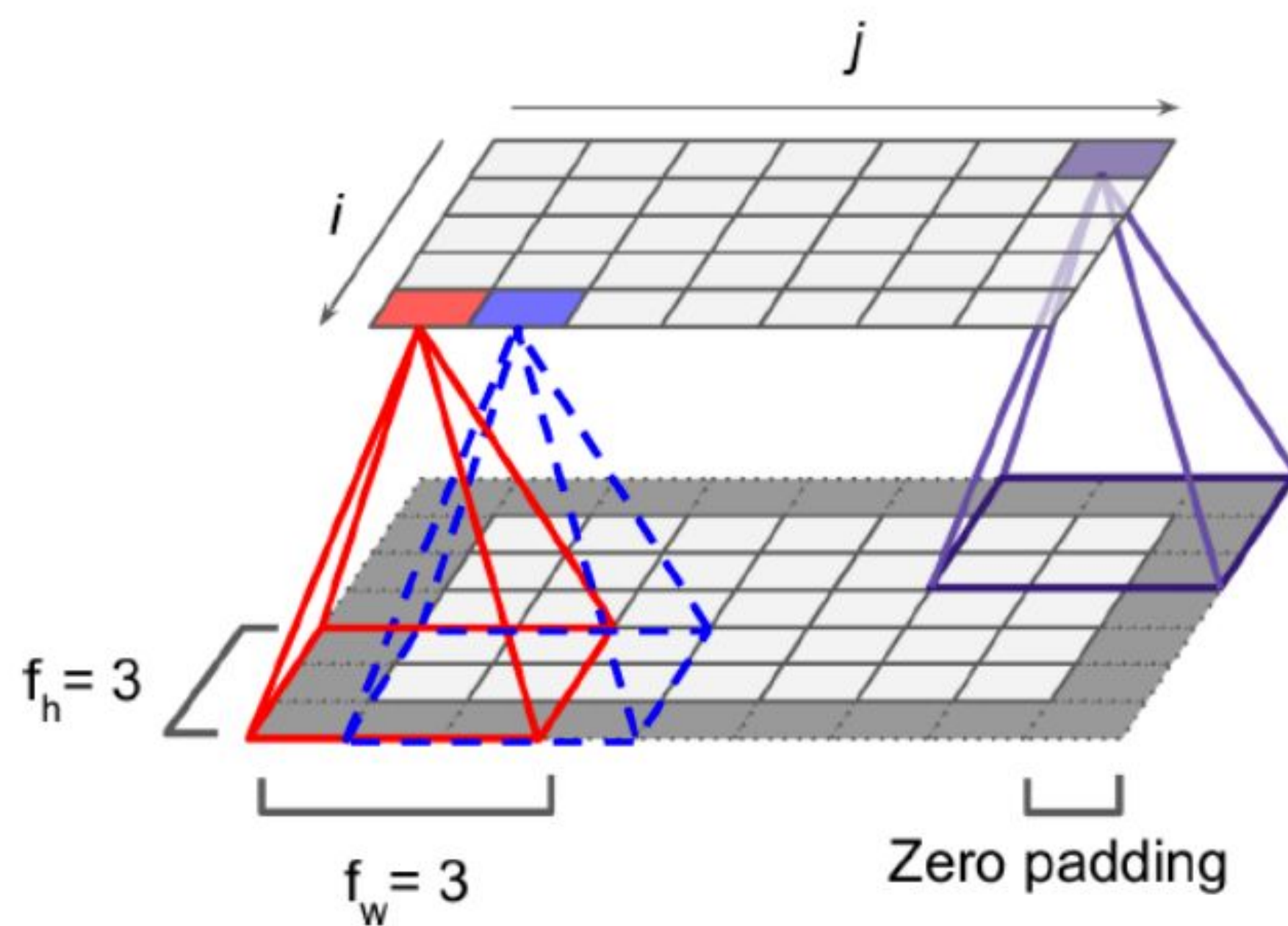
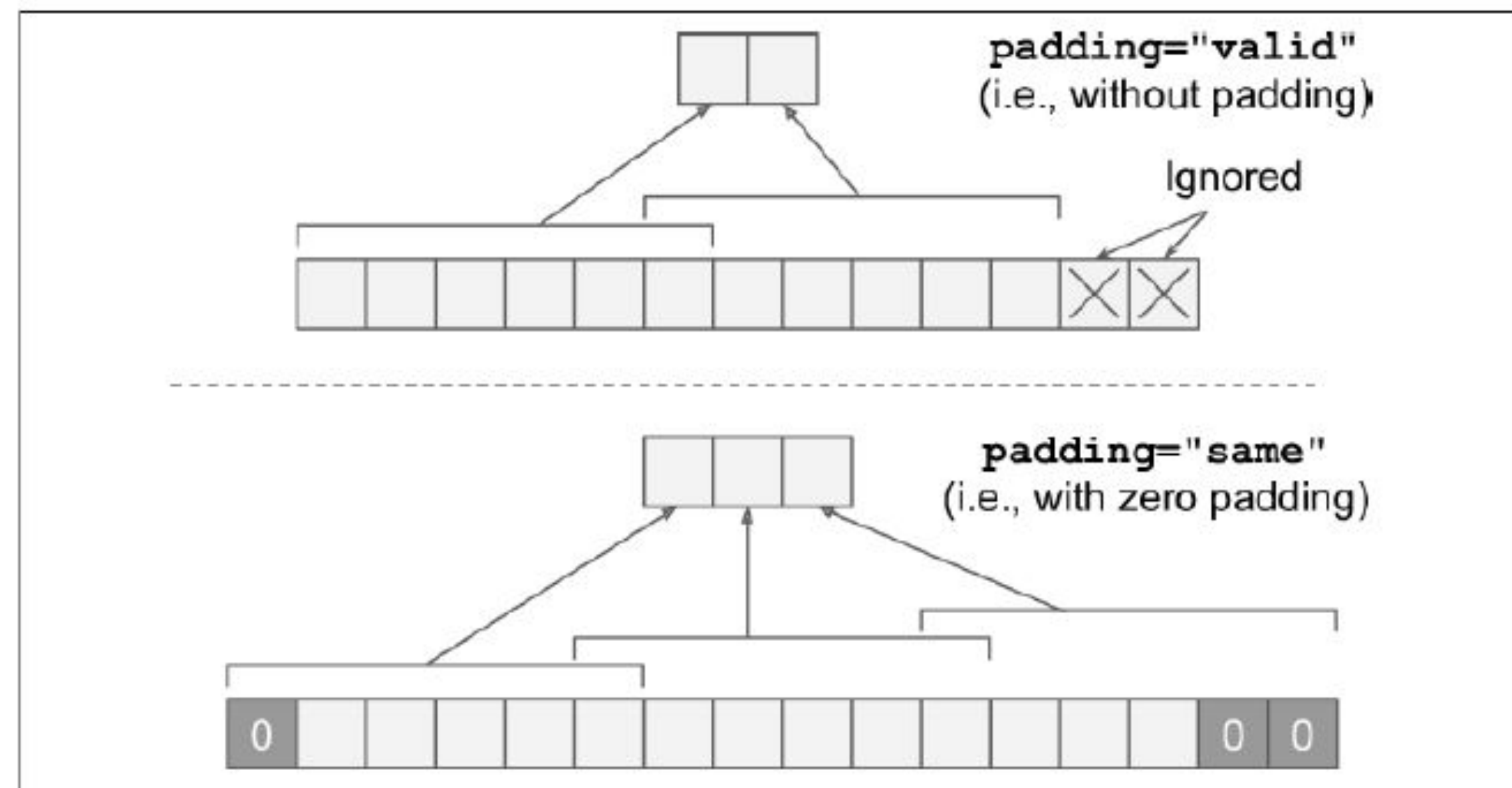


Imagen del Gerón



# Striding

- Otra forma de controlar la dimensión es controlar cuanto queremos que se mueva el kernel por la imagen con el parámetro *stride*.

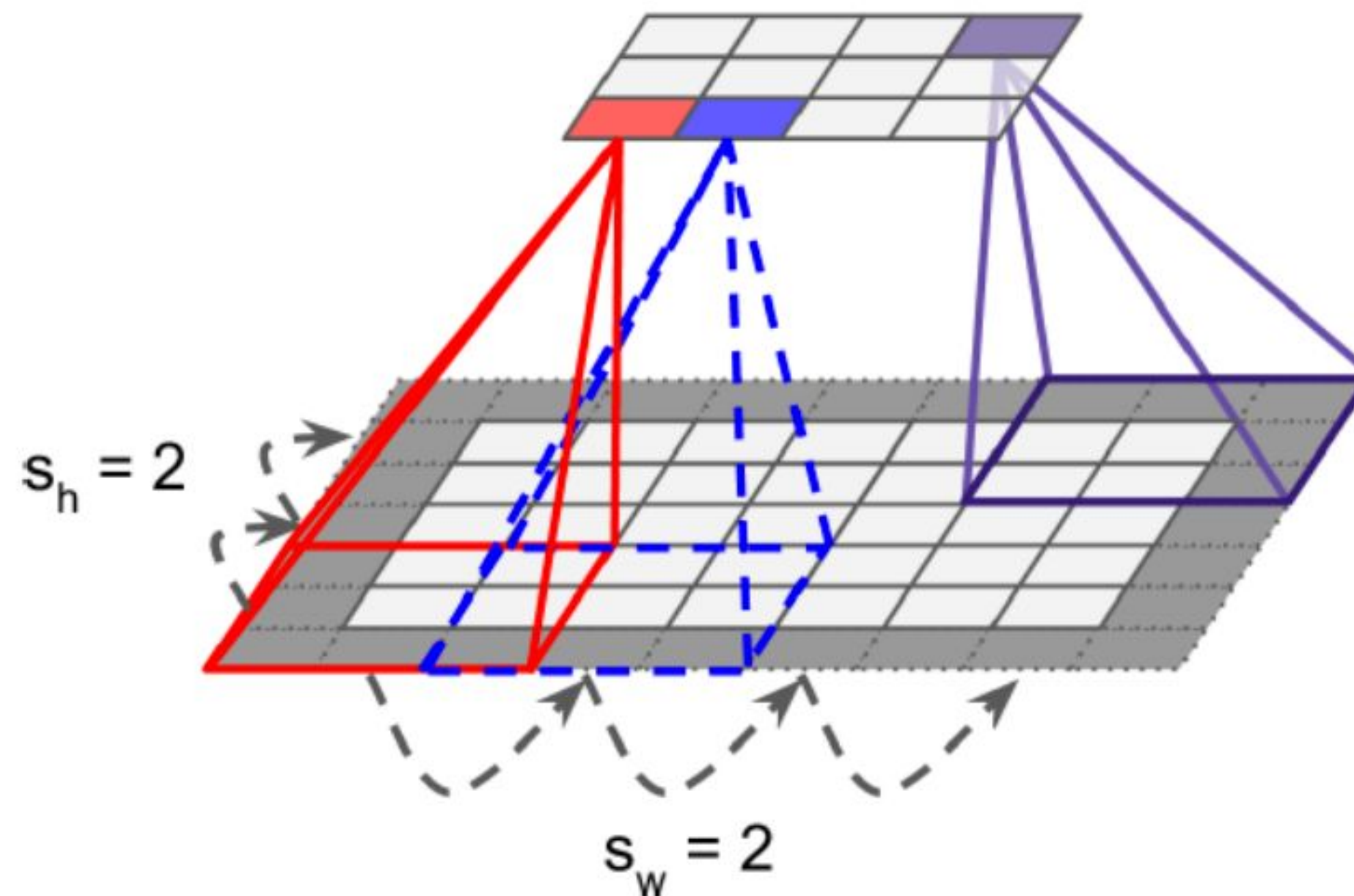
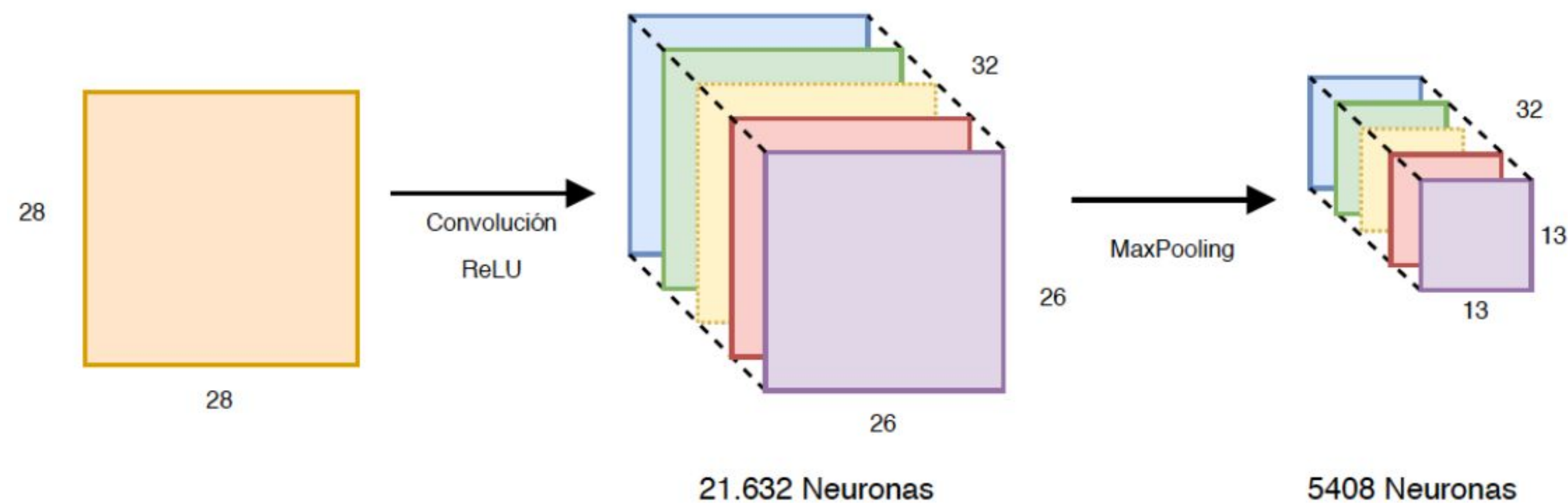


Imagen del Gerón



# Pooling

- Las *CNNs* son bastante intensivas en el uso de la memoria, una forma de mitigar esto es utilizar capas de reducción (*pooling layers*) que ayudan a reducir el tamaño de los mapas de características.
- También ayudan a reducir el *overfitting* y aumentan el nivel de invarianza traslacional de la red.
- *Max pooling* y *average pooling*.



# Pooling

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

\*



1	-1	1
-1	1	-1
1	-1	1

elegimos un filtro  
(núcleo/Kernel) que  
extrae formas de  
cruz

=

5	-1	-1
-3	3	1
4	-2	0

Convolución

(no es un producto de matrices)



# Pooling

1	0	1	1	0
0	1	0	0	1
1	0	1	1	0
1	1	0	1	1
1	0	1	1	0

Pedazo de la Imagen

\*

1	-1	1
-1	1	-1
1	-1	1

=

5	-1	-1
-3	3	1
4	-2	0

Convolución

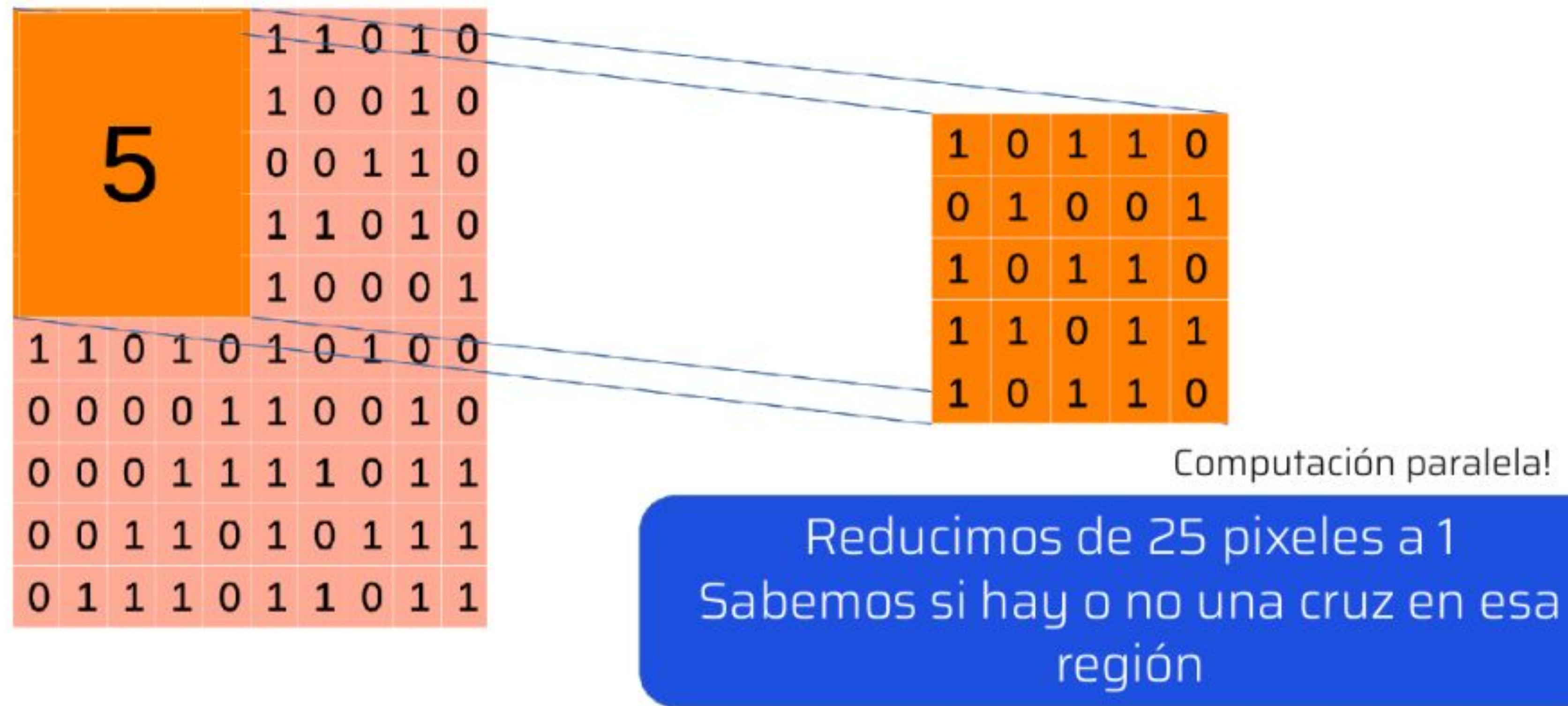
(no es un producto de matrices)

elegimos un filtro  
(núcleo/Kernel) que  
extrae formas de  
cruz

Si tomamos el valor  
máximo con MaxPool  
estamos quedándonos  
con la cruz

5

# Pooling





# Pooling

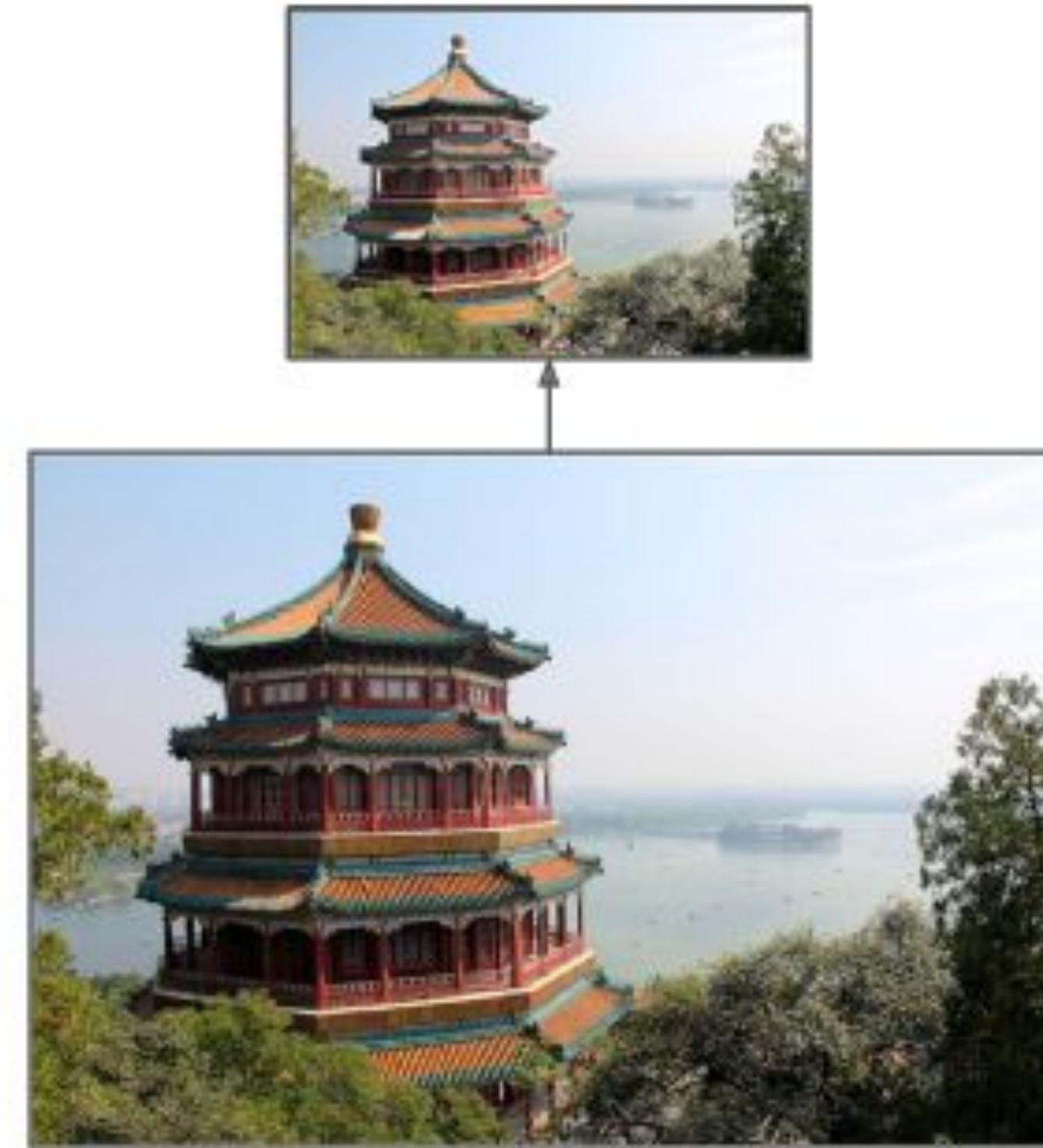
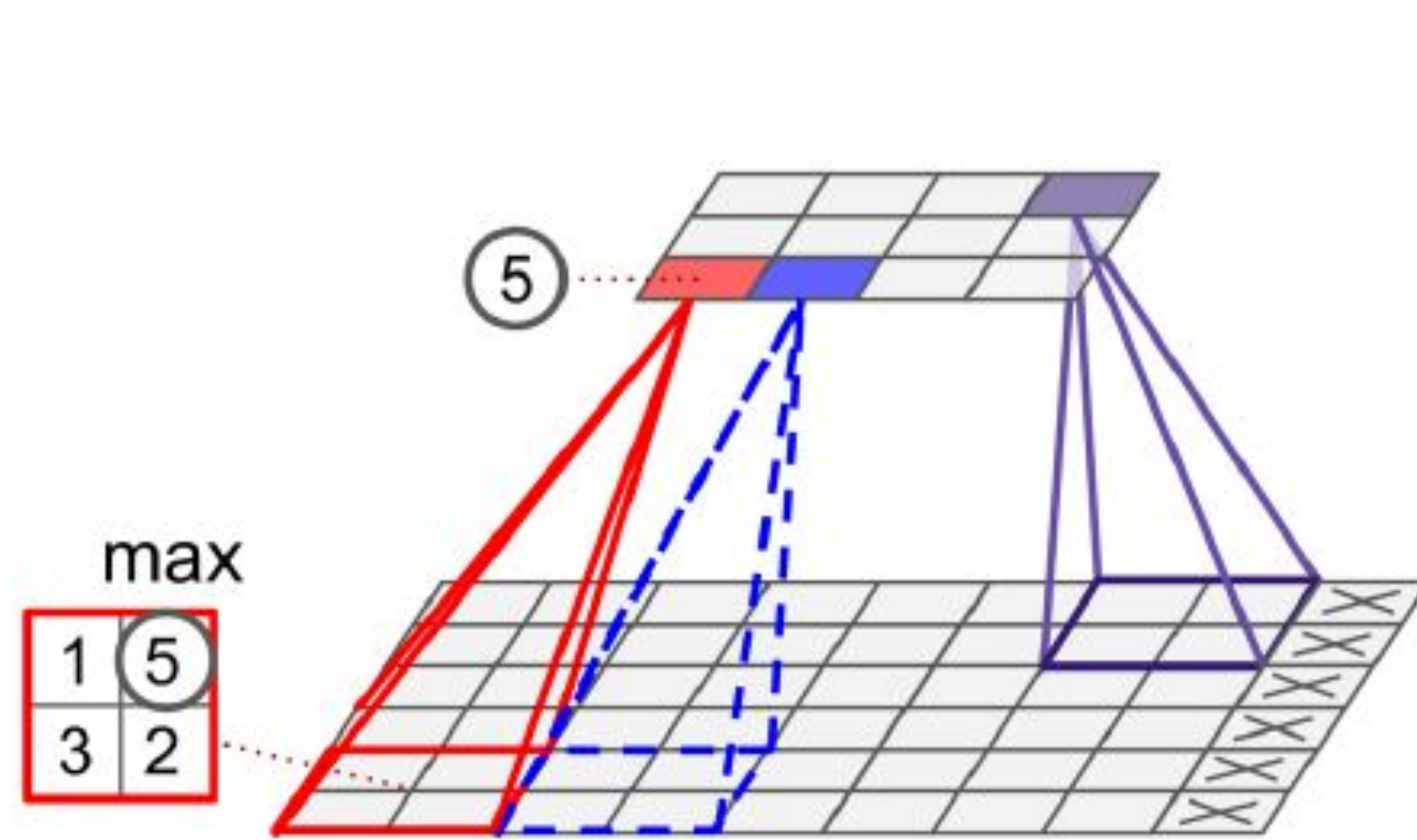


Imagen del Gerón

# Red convolucional completa

---

- La arquitectura va a ser, entonces, una combinación de capas convolucionales con capas de *pooling*.
- ¿Cuántas? ¿de qué características?, depende...(\*).
- El detalle que nos falta es el final, si nos creyeron que a medida que la red se hace más profunda extrae características más generales también nos pueden creer que podemos usar una red totalmente conectada para trabajar con estas características, ¿NOSIERTO?.
- Entonces, al final de todo, se agrega una red de las tradicionales para que trabaje con estas características y una salida acorde al problema con el que estamos trabajando (en el caso de MNIST sería una capa de clasificación).

(\*)si, como siempre, aunque hay algunas *rule of thumb*...



# Red convolucional completa

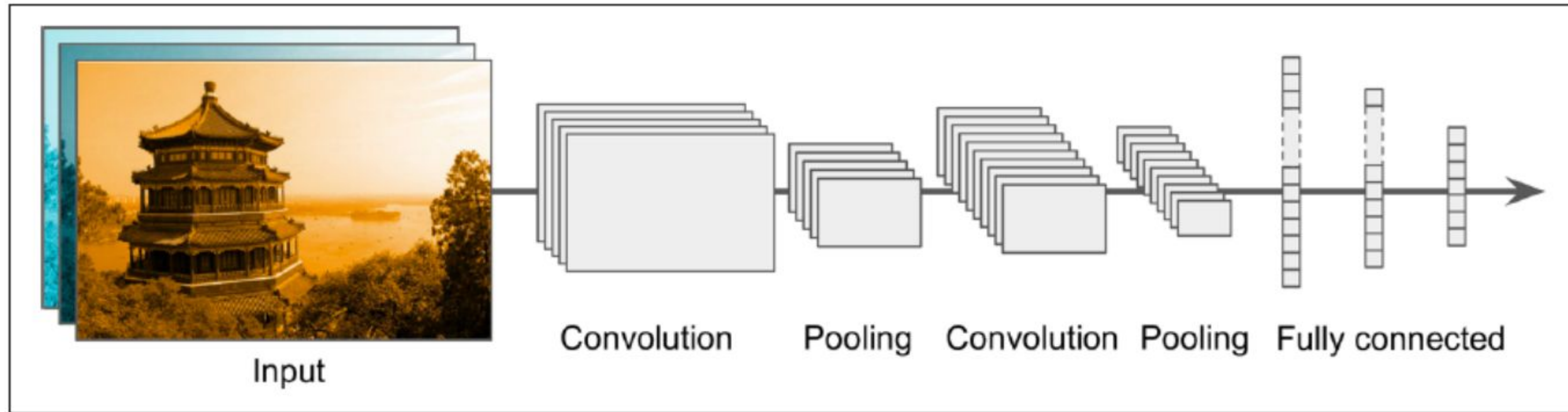


Imagen del Gerón

# Red convolucional completa

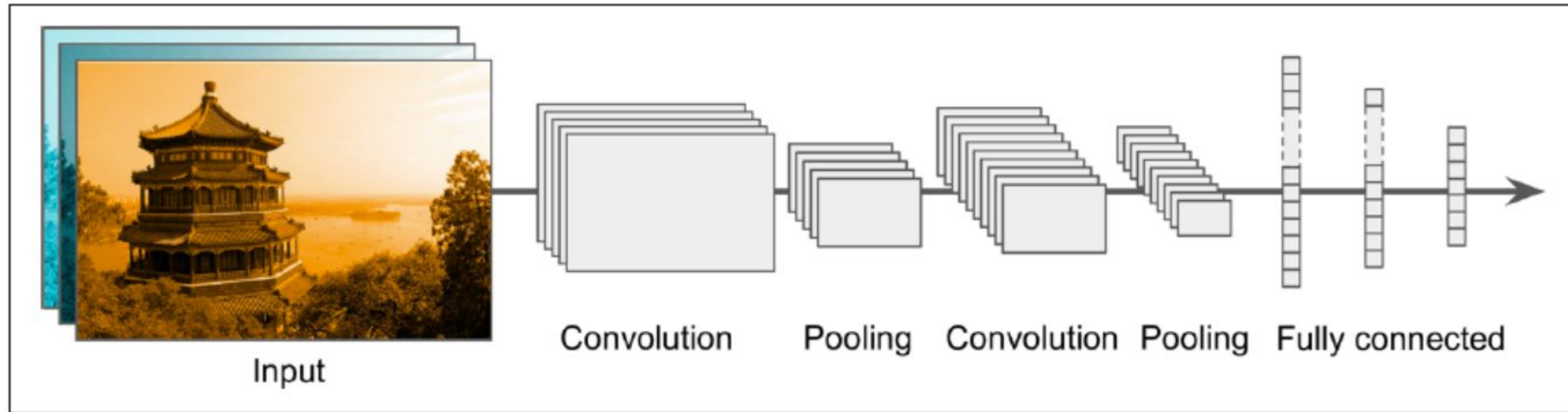


Imagen del Gerón

- A medida que se avanza aumenta el número de mapas de características.
- Disminuye las dimensiones de las imágenes (sobre todo por el pooling).  
la información espacial se va “transformando” en información sobre las características de las imágenes (se vuelve más *semántica*).
- Después de un cierto número de capas la información está lista para ser transmitida a una red neuronal totalmente conectada.



# Red convolucional completa

---

- Si bien su uso más extendido es en las imágenes, pueden usarse en cualquier tipo de datos que tenga dependencia espacial. Por ejemplo texto, audio, series temporales, etc.
- Si bien en la actualidad hay alternativas mucho más especializadas para estos casos, una red con convoluciones en una dimensión pueden dar una buena solución para problemas simples o incluso usarse en combinación con estas arquitecturas más complejas para el tratamiento de secuencias.
- Para una idea de esto pueden ver el capítulo 15 del Gerón.

# Parada técnica: Resumen

---

- **Redes convolucionales:** Redes creadas imitando el funcionamiento del córtex visual.
- **Capa Convolutional:** Es la capa que aplica los filtros a la imagen de entrada o a la capa anterior. Cada filtro recorre la imagen y genera un mapa de características (*feature map*) con la información extraída.
- **Filtros (o Kernels):** Pequeñas matrices de pesos que se usan para aplicar operaciones convolucionales sobre la imagen de entrada o la capa anterior.
- **Capa de Pooling:** Reduce la dimensionalidad de los mapas de características, disminuyendo la cantidad de información a procesar sin perder las características más importantes. Los tipos comunes son *Max Pooling* y *Average Pooling*.
- **Padding:** Se agregan ceros alrededor de la imagen para mantener las dimensiones después de la convolución.
- **Stride:** Es el tamaño del paso con el que los filtros se mueven sobre la imagen. Controla la resolución de la salida y la cantidad de información que se extrae en cada paso.

