

# Convolutional Neural Networks (CNNs)

---

Convolutional Neural Networks (CNNs)

---

# Introducción

## Motivación

¿Es posible hacer que las redes neuronales sean capaces **ver**?

Propiedades necesarias:

- 1 **Localidad**
- 2 **Invarianza al movimiento**
- 3 **Composición jerárquica**

# Introducción

Propiedades necesarias:

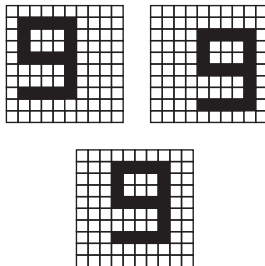
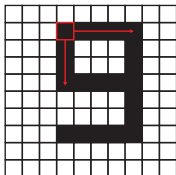
- 1 **Localidad:** Fijarse en zonas concretas. Elementos cercanos están relacionados.
- 2 **Invarianza al movimiento**
- 3 **Composición jerárquica**



# Introducción

Propiedades necesarias:

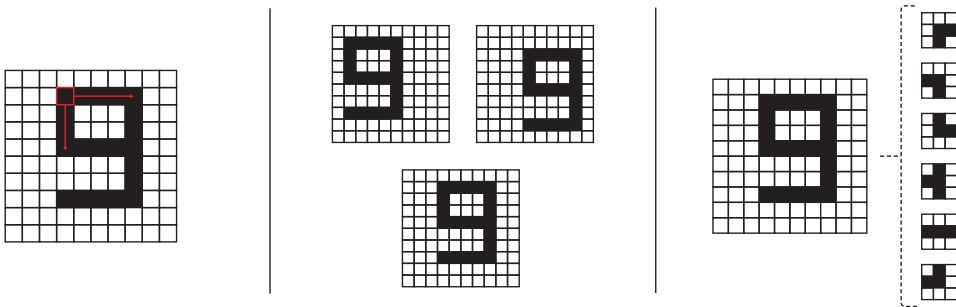
- 1 **Localidad:** Fijarse en zonas concretas. Elementos cercanos están relacionados.
- 2 **Invarianza al movimiento:** Misma salida si la posición del objeto de entrada cambia.
- 3 **Composición jerárquica**



# Introducción

Propiedades necesarias:

- 1 **Localidad:** Fijarse en zonas concretas. Elementos cercanos están relacionados.
- 2 **Invarianza al movimiento:** Misma salida si la posición del objeto de entrada cambia.
- 3 **Composición jerárquica:** Detectar patrones que definen un objeto.



## ¿Cómo funciona nuestra visión?

En 1981, David Hubel y Torsten Wiesel reciben el Nobel de medicina por sus contribuciones en el campo de la neurociencia y su investigación sobre el procesamiento visual en el cerebro.







## Convolutional networks

En 1990, LeCun entrena una red convolucional utilizando backpropagation. Aboga por el aprendizaje de características end-to-end en la clasificación de imágenes.



Convolutional Neural Networks (CNNs)

---

# Convoluciones

# Convoluciones

## Codificación de imágenes

Las imágenes son matrices de **pixels**. Cada uno posee un valor entre  $[0, 255]$  que representan la intensidad.

- Si la imagen es en escala de grises, solo tendremos **una matriz**.
- Si es en color, tendremos **tres matrices**, una por cada **canal** (Rojo, Verde y Azul).

Imagen de 6x6 (Grayscale)



255	255	255	255	255	255
255	0	0	0	255	255
255	0	255	0	255	255
255	0	0	0	255	255
255	255	255	0	255	255
255	255	255	255	255	255

Imagen de 6x6 (RGB)




# Convoluciones

## Procesamiento dentro de una red

No pueden ser tratados como vectores “no estructurados” normales, han de ser invariantes al movimiento.



Además, los modelos resultantes tendrían un tamaño enorme:

- Una pequeña imagen en escala de grises de  $100 \times 100$  generaría un vector de 10000.

# Convoluciones

## Procesamiento dentro de una red

No pueden ser tratados como vectores “no estructurados” normales, han de ser invariantes al movimiento.



Además, los modelos resultantes tendrían un tamaño enorme:

- Una pequeña imagen en escala de grises de  $100 \times 100$  generaría un vector de 10000.

## Definición

Operación matemática capaz de extraer características o patrones de unos datos de entrada, típicamente imágenes o señales.

Compuesta por:

- Datos de entrada **x**.
- Uno o varios **kernels** o filtros **u**.
- Salida **o**.

Convolutional Neural Networks (CNNs)

---

## Convoluciones 1D

# Convoluciones: 1D

Si nuestros datos de entrada son de una dimensión (una señal, por ejemplo), tendremos que aplicar la **convolución 1D**.

Tendremos por tanto:

- Vector 1D de entrada  $\mathbf{x} \in \mathbb{R}^W$
- Vector 1D kernel  $\mathbf{u} \in \mathbb{R}^w$
- Vector 1D de salida  $\mathbf{o} \in \mathbb{R}^{W-w+1}$

La operación de convolución se define como:

$$(\mathbf{x} \circledast \mathbf{u})[i] = \sum_{m=0}^{w-1} x_{m+i} \cdot u_m$$



Utilizaremos el operador  $\circledast$  para referirnos a la convolución.



# Convoluciones: 1D

Si nuestros datos de entrada son de una dimensión (una señal, por ejemplo), tendremos que aplicar la **convolución 1D**.

Tendremos por tanto:

- Vector 1D de entrada  $\mathbf{x} \in \mathbb{R}^W$
- Vector 1D kernel  $\mathbf{u} \in \mathbb{R}^w$
- Vector 1D de salida  $\mathbf{o} \in \mathbb{R}^{W-w+1}$

La operación de convolución se define como:

$$(\mathbf{x} \circledast \mathbf{u})[i] = \sum_{m=0}^{w-1} x_{m+i} \cdot u_m$$



Utilizaremos el operador  $\circledast$  para referirnos a la convolución.

# Convoluciones: 1D

Si nuestros datos de entrada son de una dimensión (una señal, por ejemplo), tendremos que aplicar la **convolución 1D**.

Tendremos por tanto:

- Vector 1D de entrada  $\mathbf{x} \in \mathbb{R}^W$
- Vector 1D kernel  $\mathbf{u} \in \mathbb{R}^w$
- Vector 1D de salida  $\mathbf{o} \in \mathbb{R}^{W-w+1}$

La operación de convolución se define como:

$$(\mathbf{x} \circledast \mathbf{u})[i] = \sum_{m=0}^{w-1} x_{m+i} \cdot u_m$$



Utilizaremos el operador  $\circledast$  para referirnos a la convolución.

# Convoluciones: 1D

Si nuestros datos de entrada son de una dimensión (una señal, por ejemplo), tendremos que aplicar la **convolución 1D**.

Tendremos por tanto:

- Vector 1D de entrada  $\mathbf{x} \in \mathbb{R}^W$
- Vector 1D kernel  $\mathbf{u} \in \mathbb{R}^w$
- Vector 1D de salida  $\mathbf{o} \in \mathbb{R}^{W-w+1}$

La operación de convolución se define como:

$$(\mathbf{x} \circledast \mathbf{u})[i] = \sum_{m=0}^{w-1} x_{m+i} \cdot u_m$$



Utilizaremos el operador  $\circledast$  para referirnos a la convolución.

# Convoluciones: 1D

Si nuestros datos de entrada son de una dimensión (una señal, por ejemplo), tendremos que aplicar la **convolución 1D**.

Tendremos por tanto:

- Vector 1D de entrada  $\mathbf{x} \in \mathbb{R}^W$
- Vector 1D kernel  $\mathbf{u} \in \mathbb{R}^w$
- Vector 1D de salida  $\mathbf{o} \in \mathbb{R}^{W-w+1}$

La operación de convolución se define como:

$$(\mathbf{x} \circledast \mathbf{u})[i] = \sum_{m=0}^{w-1} x_{m+i} \cdot u_m$$



Utilizaremos el operador  $\circledast$  para referirnos a la convolución.

# Convoluciones: 1D

Si nuestros datos de entrada son de una dimensión (una señal, por ejemplo), tendremos que aplicar la **convolución 1D**.

Tendremos por tanto:

- Vector 1D de entrada  $\mathbf{x} \in \mathbb{R}^W$
- Vector 1D kernel  $\mathbf{u} \in \mathbb{R}^w$
- Vector 1D de salida  $\mathbf{o} \in \mathbb{R}^{W-w+1}$

La operación de convolución se define como:

$$(\mathbf{x} \circledast \mathbf{u})[i] = \sum_{m=0}^{w-1} x_{m+i} \cdot u_m$$



Utilizaremos el operador  $\circledast$  para referirnos a la convolución.

# Convoluciones: 1D

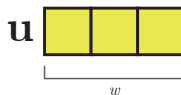
Si nuestros datos de entrada son de una dimensión (una señal, por ejemplo), tendremos que aplicar la **convolución 1D**.

Tendremos por tanto:

- Vector 1D de entrada  $\mathbf{x} \in \mathbb{R}^W$
- Vector 1D kernel  $\mathbf{u} \in \mathbb{R}^w$
- Vector 1D de salida  $\mathbf{o} \in \mathbb{R}^{W-w+1}$

La operación de convolución se define como:

$$(\mathbf{x} \circledast \mathbf{u})[i] = \sum_{m=0}^{w-1} x_{m+i} \cdot u_m$$



Utilizaremos el operador  $\circledast$  para referirnos a la convolución.

# Convoluciones: 1D

Las convoluciones pueden aplicar diferentes kernels o filtros.

Por ejemplo, en una señal eléctrica, podemos aplicar un filtro para buscar incrementos de voltaje (en 1 unidad):

$$(0, 0, 0, 0, 1, 2, 3, 3) \circledast (-1, 1) = (0, 0, 0, 1, 1, 1, 0)$$

Gráficamente:



Convolutional Neural Networks (CNNs)

---

## Convoluciones 2D



# Convoluciones: 2D

## Las dimensiones de la entrada definen el tipo de convolución

Cuando trabajamos con imágenes en escala de grises o cualquier matriz, necesitamos convoluciones 2D.

Tendremos por tanto:

- Matriz 2D de entrada  $\mathbf{x} \in \mathbb{R}^{W \times H}$
- Matriz 2D kernel  $\mathbf{u} \in \mathbb{R}^{w \times h}$
- Matriz 2D de salida  $\mathbf{o} \in \mathbb{R}^{W-w+1 \times H-h+1}$

La operación de convolución en 2D sería:

$$\mathbf{o}_{j,i} = (\mathbf{x} \circledast \mathbf{u})[j, i] = \sum_{n=0}^{h-1} \sum_{m=0}^{w-1} \mathbf{x}_{n+j, m+i} \cdot \mathbf{u}_{n,m}$$

# Convoluciones: 2D

## Las dimensiones de la entrada definen el tipo de convolución

Cuando trabajamos con imágenes en escala de grises o cualquier matriz, necesitamos convoluciones 2D.

Gráficamente:



# Convoluciones: 2D

## Las dimensiones de la entrada definen el tipo de convolución

Cuando trabajamos con imágenes en escala de grises o cualquier matriz, necesitamos convoluciones 2D.

Gráficamente:



# Convoluciones: 2D

## Las dimensiones de la entrada definen el tipo de convolución

Cuando trabajamos con imágenes en escala de grises o cualquier matriz, necesitamos convoluciones 2D.

Gráficamente:

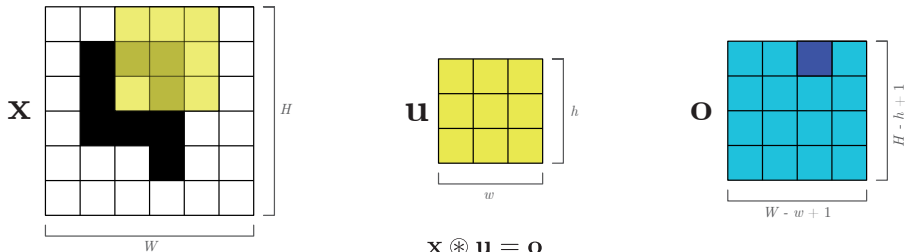


# Convoluciones: 2D

## Las dimensiones de la entrada definen el tipo de convolución

Cuando trabajamos con imágenes en escala de grises o cualquier matriz, necesitamos convoluciones 2D.

Gráficamente:



# Convoluciones: 2D

## Las dimensiones de la entrada definen el tipo de convolución

Cuando trabajamos con imágenes en escala de grises o cualquier matriz, necesitamos convoluciones 2D.

Gráficamente:



# Convoluciones: 2D

## Las dimensiones de la entrada definen el tipo de convolución

Cuando trabajamos con imágenes en escala de grises o cualquier matriz, necesitamos convoluciones 2D.

Gráficamente:



# Convoluciones: 2D

## Las dimensiones de la entrada definen el tipo de convolución

Cuando trabajamos con imágenes en escala de grises o cualquier matriz, necesitamos convoluciones 2D.

Gráficamente:





Convolutional Neural Networks (CNNs)

---

## Convoluciones 3D

# Convoluciones: 3D

## ¿Y si trabajamos con imágenes en color?

Como ya hemos mencionado, las imágenes RGB están formadas por 3 matrices de  $W \times H$ , es decir, un **volumen o tensor**. En este caso aplicaremos convoluciones 3D.

Tendremos por tanto:

- Tensor 3D de entrada  $\mathbf{x} \in \mathbb{R}^{C \times W \times H}$
- Tensor 3D kernel  $\mathbf{u} \in \mathbb{R}^{C \times w \times h}$
- Tensor 3D de salida  $\mathbf{o} \in \mathbb{R}^{W-w+1 \times H-h+1}$

La operación de convolución en 3D sería:

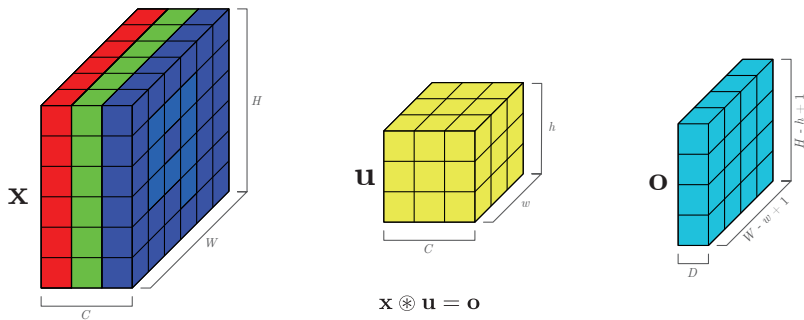
$$\mathbf{o}_{j,i} = \sum_{c=0}^{C-1} (\mathbf{x}_c \circledast \mathbf{u}_c)[j, i] = \sum_{c=0}^{C-1} \sum_{n=0}^{h-1} \sum_{m=0}^{w-1} \mathbf{x}_{c,n+j,m+i} \cdot \mathbf{u}_{c,n,m}$$

# Convoluciones: 3D

## ¿Y si trabajamos con imágenes en color?

Como ya hemos mencionado, las imágenes RGB están formadas por 3 matrices de  $W \times H$ , es decir, un **volumen o tensor**. En este caso aplicaremos convoluciones 3D.

Gráficamente:

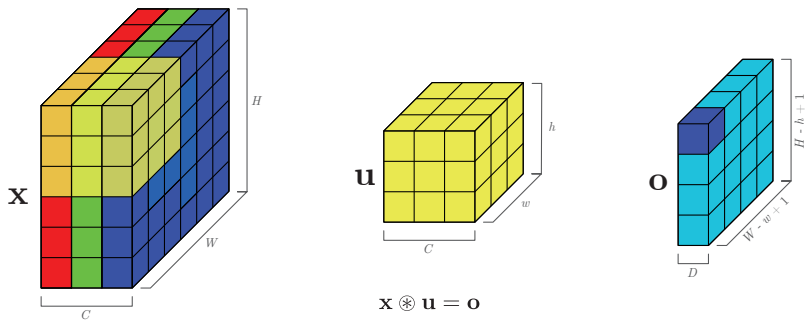


# Convoluciones: 3D

## ¿Y si trabajamos con imágenes en color?

Como ya hemos mencionado, las imágenes RGB están formadas por 3 matrices de  $W \times H$ , es decir, un **volumen o tensor**. En este caso aplicaremos convoluciones 3D.

Gráficamente:

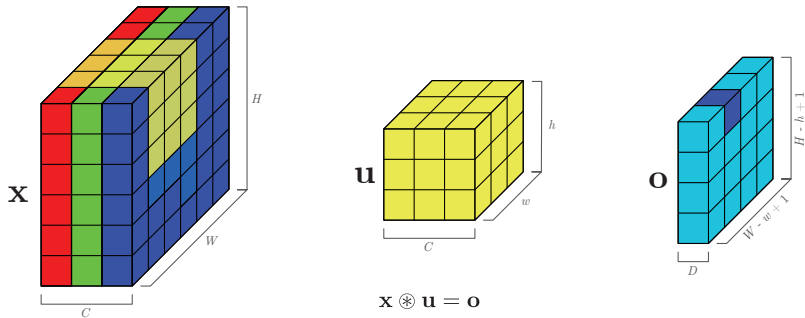


# Convoluciones: 3D

## ¿Y si trabajamos con imágenes en color?

Como ya hemos mencionado, las imágenes RGB están formadas por 3 matrices de  $W \times H$ , es decir, un **volumen o tensor**. En este caso aplicaremos convoluciones 3D.

Gráficamente:

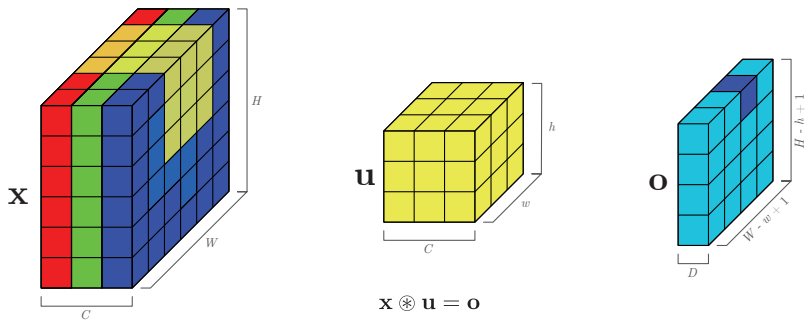


# Convoluciones: 3D

## ¿Y si trabajamos con imágenes en color?

Como ya hemos mencionado, las imágenes RGB están formadas por 3 matrices de  $W \times H$ , es decir, un **volumen o tensor**. En este caso aplicaremos convoluciones 3D.

Gráficamente:



# Convoluciones: 3D

## ¿Y si trabajamos con imágenes en color?

Como ya hemos mencionado, las imágenes RGB están formadas por 3 matrices de  $W \times H$ , es decir, un **volumen o tensor**. En este caso aplicaremos convoluciones 3D.

Gráficamente:

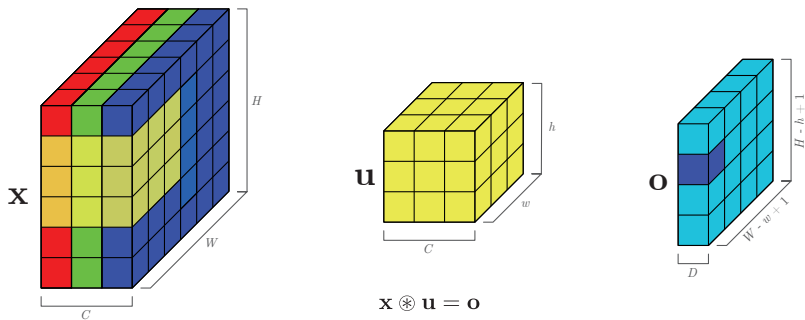


# Convoluciones: 3D

## ¿Y si trabajamos con imágenes en color?

Como ya hemos mencionado, las imágenes RGB están formadas por 3 matrices de  $W \times H$ , es decir, un **volumen o tensor**. En este caso aplicaremos convoluciones 3D.

Gráficamente:



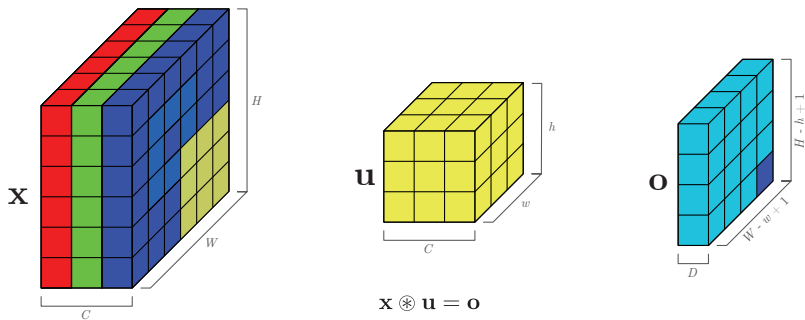


# Convoluciones: 3D

## ¿Y si trabajamos con imágenes en color?

Como ya hemos mencionado, las imágenes RGB están formadas por 3 matrices de  $W \times H$ , es decir, un **volumen o tensor**. En este caso aplicaremos convoluciones 3D.

Gráficamente:



## Múltiples dimensiones

La operación de convolución se puede aplicar en múltiples dimensiones, pero las más comunes son las anteriores, 1D, 2D y 3D.

## ¡Muy importante!

Los valores de  $\mathbf{o} \in \mathbb{R}$  son los únicos **parámetros** de la convolución, es decir, se aprenden durante el entrenamiento del modelo.

El propio modelo decide que filtros son más adecuados para la tarea que se pretende resolver.

Convolutional Neural Networks (CNNs)

---

## Convoluciones: Hiperparámetros

# Convoluciones: Hiperparámetros

Aunque la operación de convolución es siempre la misma, existen **múltiples hiperparámetros** que podemos ajustar.

Los más relevantes:

- Número  $D$  de filtros o kernels.
- Tamaño  $w \times h$  de cada kernel.
- Padding de la convolución.
- Stride de la convolución.

# Convoluciones: Hiperparámetros

## Número de kernels

En los ejemplos anteriores siempre aplicamos 1 filtro por simplificar. Normalmente este número de filtros  $D$  suele ser mayor.

- La idea es extraer más conocimiento de cada convolución.
- En 2 o 3 dimensiones, con  $D$  filtros, el tamaño de **o** será:  $D \times W - w + 1 \times H - h + 1$

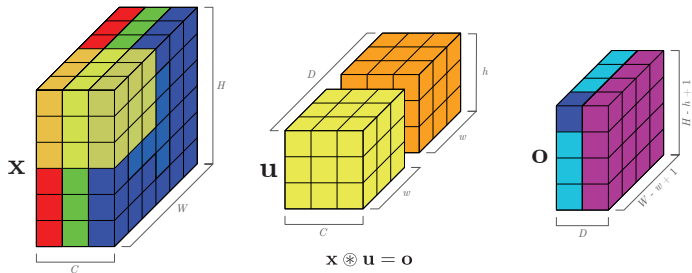


# Convoluciones: Hiperparámetros

## Número de kernels

En los ejemplos anteriores siempre aplicamos 1 filtro por simplificar. Normalmente este número de filtros  $D$  suele ser mayor.

- La idea es extraer más conocimiento de cada convolución.
- En 2 o 3 dimensiones, con  $D$  filtros, el tamaño de  $\mathbf{o}$  será:  $D \times W - w + 1 \times H - h + 1$



# Convoluciones: Hiperparámetros

## Número de kernels

En los ejemplos anteriores siempre aplicamos 1 filtro por simplificar. Normalmente este número de filtros  $D$  suele ser mayor.

- La idea es extraer más conocimiento de cada convolución.
- En 2 o 3 dimensiones, con  $D$  filtros, el tamaño de  $\mathbf{o}$  será:  $D \times W - w + 1 \times H - h + 1$

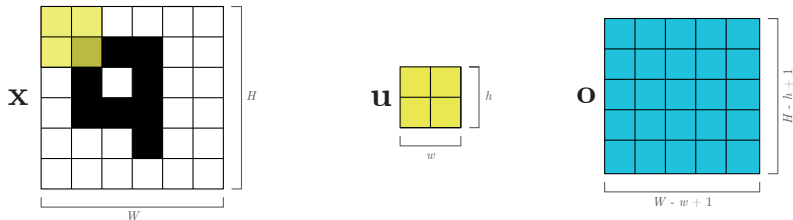


# Convoluciones: Hiperparámetros

## Tamaño de cada kernel

El tamaño  $w \times h$  de cada filtro de la convolución, es otro hiperparámetro que podemos alterar.

- Suelen ser cuadrados  $w = h$ .
- Si hay más de uno, los  $D$  han de tener el mismo tamaño.
- Influyen directamente en el tamaño de la salida. Para  $w = h = 2$ :





# Convoluciones: Hiperparámetros

## Tamaño de cada kernel

El tamaño  $w \times h$  de cada filtro de la convolución, es otro hiperparámetro que podemos alterar.

- Suelen ser cuadrados  $w = h$ .
- Si hay más de uno, los  $D$  han de tener el mismo tamaño.
- Influyen directamente en el tamaño de la salida. Para  $w = h = 3$ :



# Convoluciones: Hiperparámetros

## Tamaño de cada kernel

El tamaño  $w \times h$  de cada filtro de la convolución, es otro hiperparámetro que podemos alterar.

- Suelen ser cuadrados  $w = h$ .
- Si hay más de uno, los  $D$  han de tener el mismo tamaño.
- Influyen directamente en el tamaño de la salida. Para  $w = h = 4$ :

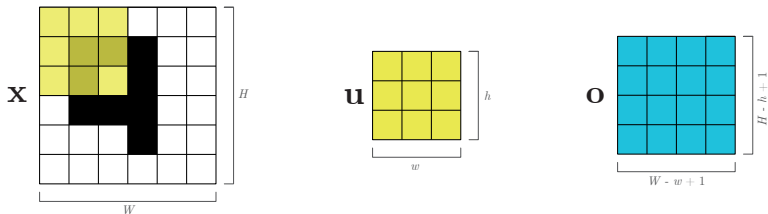


# Convoluciones: Hiperparámetros

## Padding

Al aplicar una convolución, vemos que la salida es de menor tamaño que la entrada. El **padding** nos permite evitarlo.

- El padding añade un “marco” de píxels a la entrada. El grosor es un hiperparámetro.
- El valor por defecto de estos nuevos píxels suele ser zero (zero padding).



# Convoluciones: Hiperparámetros

## Padding

Al aplicar una convolución, vemos que la salida es de menor tamaño que la entrada. El **padding** nos permite evitarlo.

- El padding añade un “marco” de píxels a la entrada. El grosor es un hiperparámetro.
- El valor por defecto de estos nuevos píxels suele ser zero (zero padding).



# Convoluciones: Hiperparámetros

## Stride

En los ejemplos anteriores, siempre movemos el filtro **u** de **una en una unidad** sobre la entrada **x** en cada dirección, pero esto no tiene por que ser siempre así.

- Ese valor es lo que se conoce como stride.
- Al igual que el padding, influye en el tamaño de salida.

Ya hemos visto lo que sucede con un **stride de tamaño uno**:



# Convoluciones: Hiperparámetros

## Stride

En los ejemplos anteriores, siempre movemos el filtro **u** de **una en una unidad** sobre la entrada **x** en cada dirección, pero esto no tiene por que ser siempre así.

- Ese valor es lo que se conoce como stride.
- Al igual que el padding, influye en el tamaño de salida.

Ya hemos visto lo que sucede con un **stride de tamaño uno**:



# Convoluciones: Hiperparámetros

## Stride

En los ejemplos anteriores, siempre movemos el filtro **u** de **una en una unidad** sobre la entrada **x** en cada dirección, pero esto no tiene por que ser siempre así.

- Ese valor es lo que se conoce como stride.
- Al igual que el padding, influye en el tamaño de salida.

Ya hemos visto lo que sucede con un **stride de tamaño uno**:



# Convoluciones: Hiperparámetros

## Stride

En los ejemplos anteriores, siempre movemos el filtro **u** de **una en una unidad** sobre la entrada **x** en cada dirección, pero esto no tiene por que ser siempre así.

- Ese valor es lo que se conoce como stride.
- Al igual que el padding, influye en el tamaño de salida.

Ya hemos visto lo que sucede con un **stride de tamaño uno**:





# Convoluciones: Hiperparámetros

## Stride

En los ejemplos anteriores, siempre movemos el filtro **u** de **una en una unidad** sobre la entrada **x** en cada dirección, pero esto no tiene por que ser siempre así.

- Ese valor es lo que se conoce como stride.
- Al igual que el padding, influye en el tamaño de salida.

¿Que sucede si cambiamos el tamaño del **stride a tres**?:



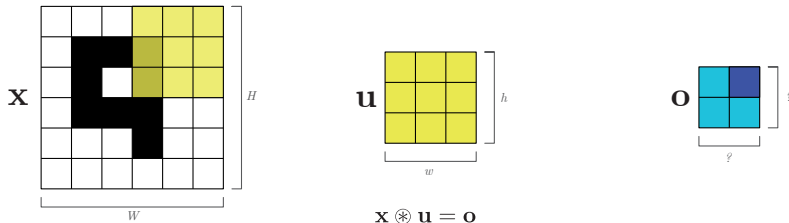
# Convoluciones: Hiperparámetros

## Stride

En los ejemplos anteriores, siempre movemos el filtro **u** de **una en una unidad** sobre la entrada **x** en cada dirección, pero esto no tiene por que ser siempre así.

- Ese valor es lo que se conoce como stride.
- Al igual que el padding, influye en el tamaño de salida.

¿Que sucede si cambiamos el tamaño del **stride a tres**?:



# Convoluciones: Hiperparámetros

## Stride

En los ejemplos anteriores, siempre movemos el filtro **u** de **una en una unidad** sobre la entrada **x** en cada dirección, pero esto no tiene por que ser siempre así.

- Ese valor es lo que se conoce como stride.
- Al igual que el padding, influye en el tamaño de salida.

¿Que sucede si cambiamos el tamaño del **stride a tres**?:



# Convoluciones: Hiperparámetros

## Stride

En los ejemplos anteriores, siempre movemos el filtro **u** de **una en una unidad** sobre la entrada **x** en cada dirección, pero esto no tiene por que ser siempre así.

- Ese valor es lo que se conoce como stride.
- Al igual que el padding, influye en el tamaño de salida.

¿Que sucede si cambiamos el tamaño del **stride a tres**?:



## Tamaño de la salida

Como acabas de ver, el padding y el stride también influyen en el tamaño de la salida  $\mathbf{o}$ .

- Si llamamos  $s$  al tamaño del stride.
- Y  $p$  al tamaño del padding.

El tamaño de la salida sería el siguiente:

$$\left( \frac{W + 2p - w}{s} + 1 \right) \times \left( \frac{H + 2p - h}{s} + 1 \right)$$

Convolutional Neural Networks (CNNs)

---

# Pooling

Convolutional Neural Networks (CNNs)

---

# Arquitecturas

Convolutional Neural Networks (CNNs)

---

## Referencias



## 1 Lecture 5: Convolutional networks