

Clasificación de imágenes

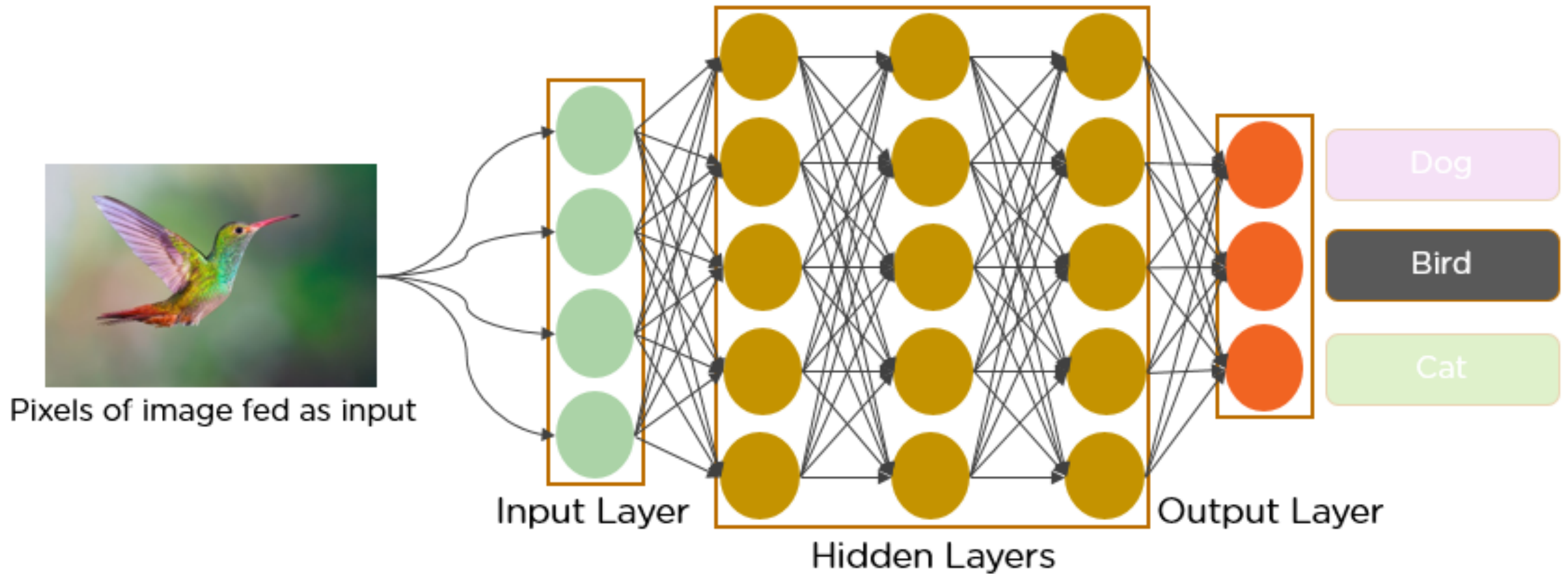


Imagen digital

- Está representada por una matriz de $M \times N$ pixels.
- Cada pixel tiene un valor numérico que indica su color y posición en la imagen.
- Su valor dependerá del tipo de imagen

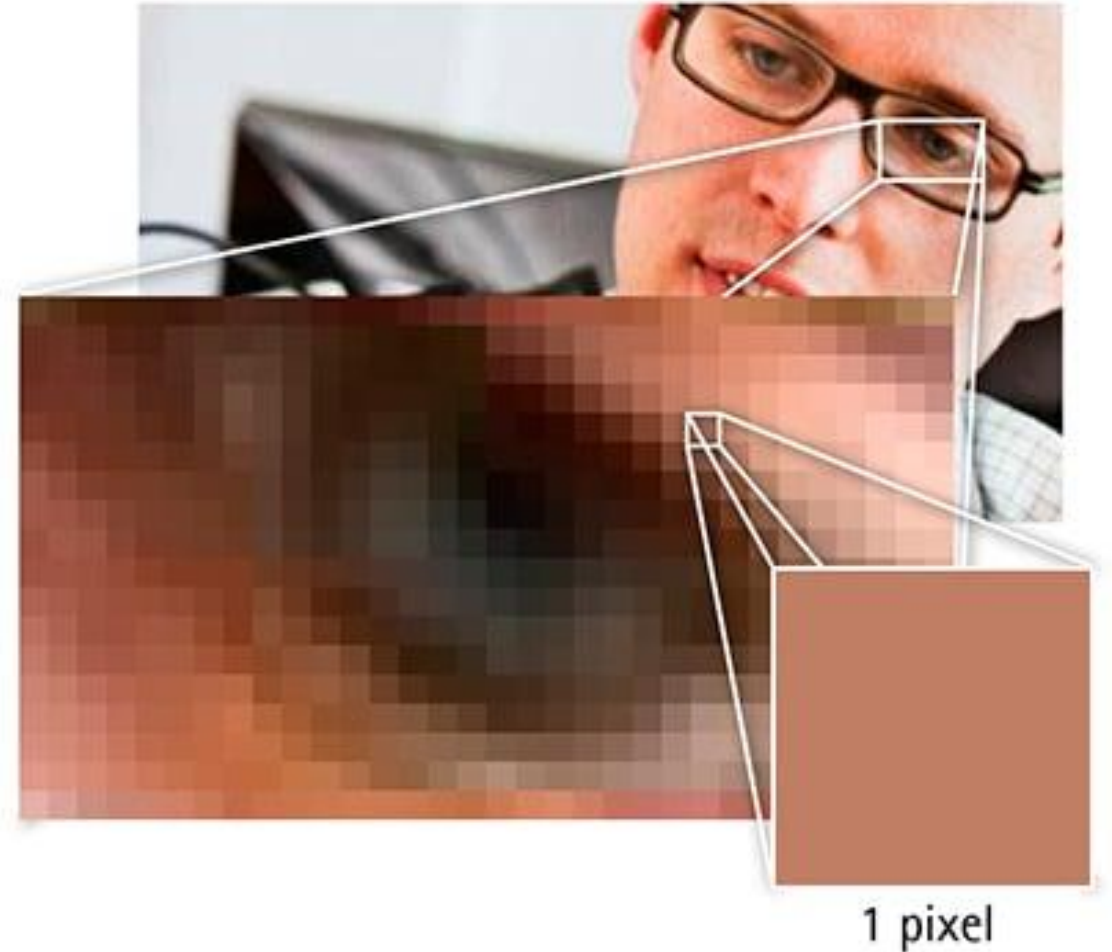


Imagen en tonos de grises



Greyscale image

		112		
	112	105	106	
105	105	105	105	112
105	112	112	105	112
112	106	106	112	105
105	112	112	112	106
106	112	116	117	105
120	123	105	105	10
	127	127	106	
	127	127	106	

Pixel/intensity value

Imagen en tonos de grises



0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0				
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29				
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0				
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1				
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49				
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36				
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62				
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0				
0	87	252	250	248	215	60	0	1	121	252	255	248	144	6	0				
0	13	113	255	255	245	255	182	181	248	252	242	208	36	0	19				
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0				
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0				
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4				
0	22	206	252	246	251	241	100	24	113	255	245	255	194	9	0				
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0				
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3				
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0				
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4				
0	18	146	250	255	247	255	255	255	249	255	240	255	129	0	5				
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0				
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1				
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0				

0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0				
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29				
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0				
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1				
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49				
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36				
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62				
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0				
0	87	252	250	248	215	60	0	1	121	252	255	248	144	6	0				
0	13	113	255	255	245	255	182	181	248	252	242	208	36	0	19				
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0				
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0				
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4				
0	22	206	252	246	251	241	100	24	113	255	245	255	194	9	0				
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0				
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3				
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0				
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4				
0	18	146	250	255	247	255	255	255	249	255	240	255	129	0	5				
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0				
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1				
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0				

¿Cómo ven humanos vs computadoras?

¿Qué tan difícil es para una computadora reconocer una imagen?

- Llevando el problema a una escala humana:
 - ▣ La imagen tiene el tamaño de una ciudad.
 - ▣ Los píxeles son del tamaño de una cancha de fútbol.
 - ▣ La persona tiene visión acotada a píxeles cercanos.
 - ▣ Para reconocer un objeto hay que recorrer todos los píxeles.



Representación de una imagen en RGB

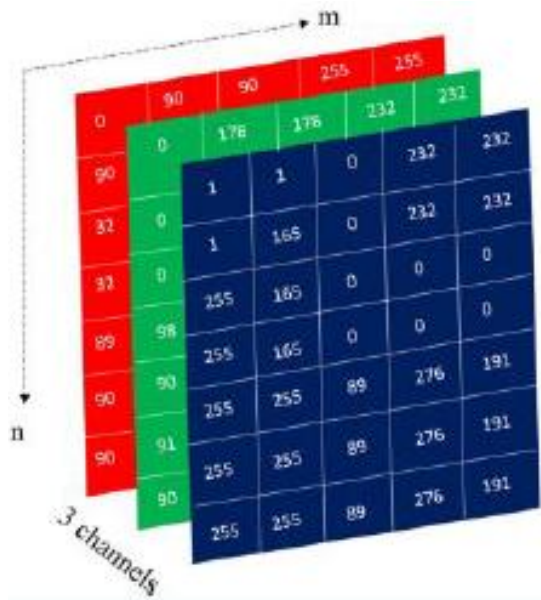


Imagen color - RGB

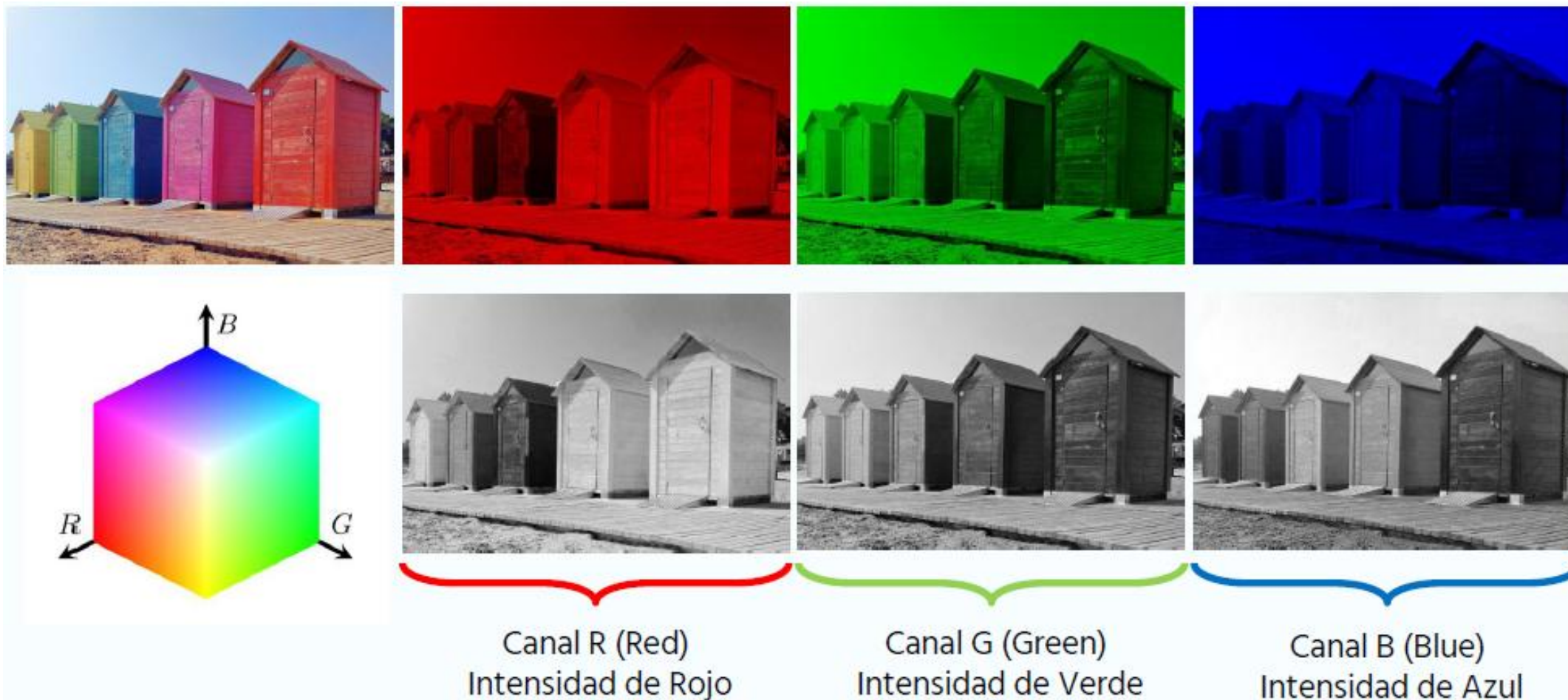


Imagen color - RGB



- Al igual que las imágenes en tonos de grises, cada canal RGB tiene valores que van de 0 a 255.
- Normalmente las imágenes se convierten a punto flotante y se escalan para que queden con valores entre 0 y 1.

Imagen color - RGB

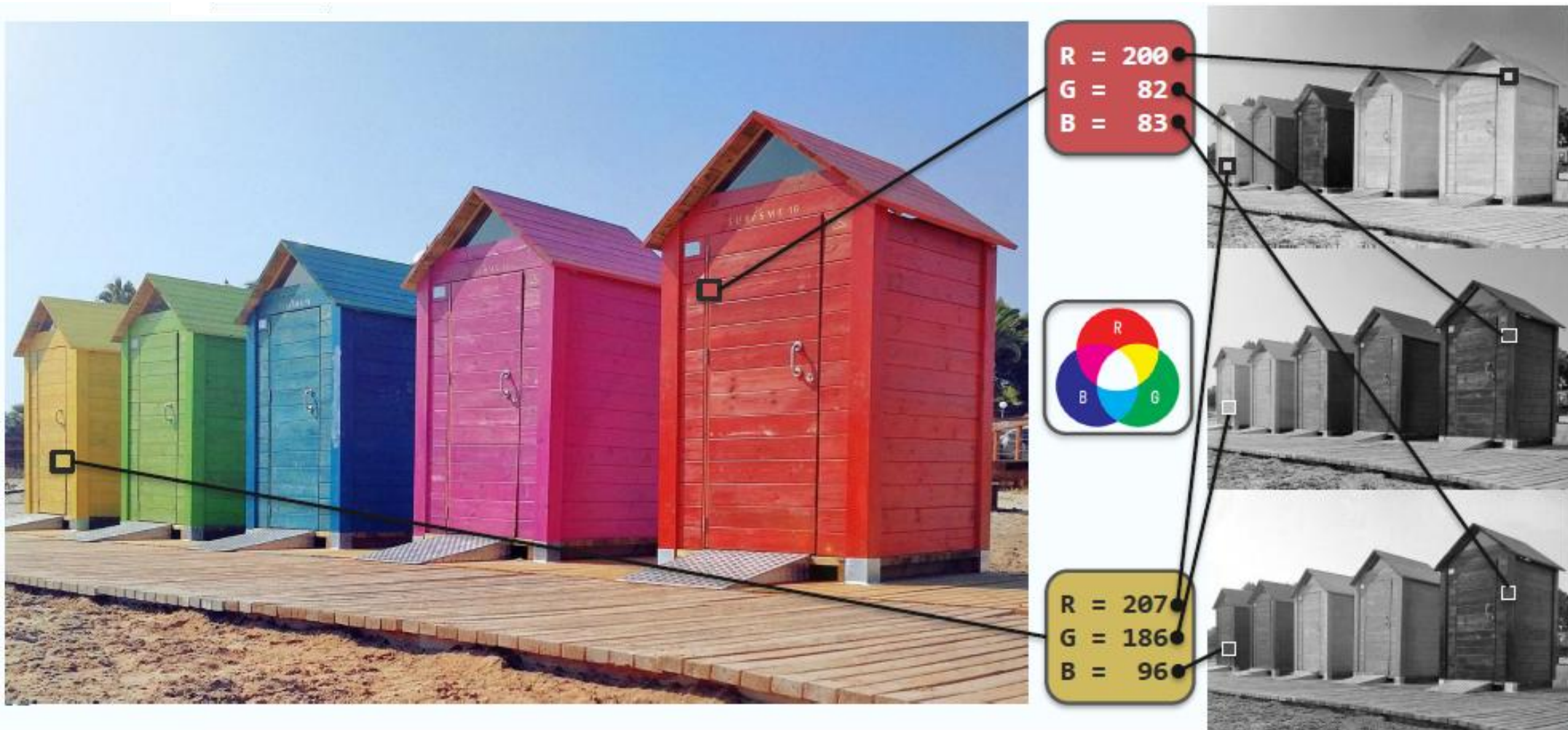


Imagen color - RGB

```
from skimage import io
import matplotlib.pyplot as plt

imgColor = io.imread('Casitas.jpg')

fig, axs = plt.subplots(2, 2, figsize=(12, 8))
axs = axs.flatten()

# Mostrar la imagen original
axs[0].imshow(imgColor)
axs[0].set_title('Imagen Original')
axs[0].axis('off') # Ocultar ejes

# Mostrar canales RGB
C = ['Reds_r', 'Greens_r', 'Blues_r']
for p in range(3):
    axs[p + 1].imshow(imgColor[:, :, p], cmap=C[p])
    axs[p + 1].set_title(C[p][: -2]) # Titulos
    axs[p + 1].axis('off')          # Ocultar ejes
plt.show() # Muestra figura
```



[Ver Imagenes_con_skimage.ipynb](#)

Conversión RGB a Tonos de Grises

- La conversión a escala de grises consiste en combinar la información de los tres canales de color (rojo, verde y azul) en un solo valor de intensidad luminosa.
- La fórmula más común es:
$$\textit{Gris} = 0.299 * \textit{Rojo} + 0.587 * \textit{Verde} + 0.114 * \textit{Azul}$$
- Los coeficientes 0.299, 0.587 y 0.114 son ponderaciones que representan la contribución relativa de cada canal al brillo percibido por el ojo humano.



Conversión RGB a Tonos de Grises

```
from skimage import io, color
import matplotlib.pyplot as plt

DATOS_DIR = 'Datos/' # ruta a las imagenes
imgColor = io.imread(DATOS_DIR + 'Casitas.jpg')

imgGray = color.rgb2gray(imgColor) # grises

fig, axs = plt.subplots(1, 2, figsize=(12, 8))

axs[0].imshow(imgColor)
axs[0].set_title('Imagen Original')
axs[0].axis('off') # Ocultar ejes

axs[1].imshow(imgGray, cmap='gray')
axs[1].set_title('Escala de Grises')
axs[1].axis('off') # Ocultar ejes

plt.show()
```

Imagen Original



Escala de Grises



Ver Imagenes_con_skimage.ipynb

Formatos para almacenamiento de imágenes

- Formatos más utilizados: PNG, TIFF y JPEG
- Comprimen la información
- PNG y TIFF
 - ▣ Compresión sin pérdida
 - ▣ Representación en 16 bits
- JPEG
 - ▣ Compresión con pérdida (variable)



Compresión 80%



Compresión 10%

Transformaciones de Imágenes

- Transformaciones comúnmente usadas:
 - ▣ Brillo y Contraste
 - ▣ Flip (espejo) horizontal y vertical
 - ▣ Rotación
 - ▣ Traslación vertical y horizontal
 - ▣ Skew
 - ▣ Zoom in/out

Imagen Original



Transformación de Imágenes – Brillo/Contraste



Menos Brillo



Mas Brillo



Menos Contraste



Más Contraste

Transformaciones – Espejo (flip)



Espejo
Horizontal



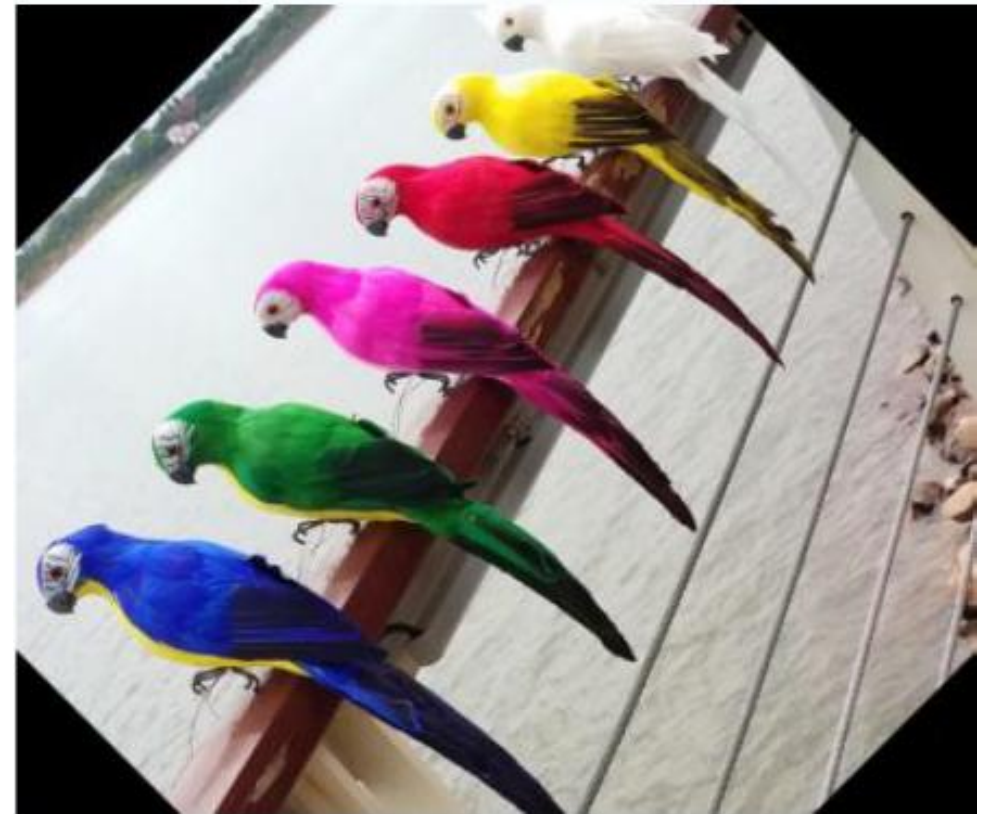
Espejo
Vertical



Espejo
Horizontal
Y Vertical

Transformaciones de Imágenes - Rotación

- Algunas transformaciones pueden provocar problemas:
 - ▣ Pérdida de información por píxeles fuera de la imagen.
 - ▣ Nuevas áreas donde no hay información (no deseados, perjudican el reconocimiento).



Transformaciones de Imágenes - Rotación

- Alternativas para completar partes de la imagen luego de la transformación



Replicar píxeles del
borde



Replicar el borde
simétricamente



Replicar el borde del
extremo opuesto

Transformación de Imágenes - Traslación



Traslación
Horizontal



Traslación
Vertical



Traslación
Horizontal
y Vertical

- Las áreas nuevas se completan igual que en la rotación.

Transformación de Imágenes - Zoom

Zoom in/out



Zoom in



Zoom out



- Las áreas nuevas se completan igual que en la rotación.

Transformaciones de una imagen

```
from skimage import transform, exposure

# Rotar una imagen una cantidad de grados
imgRotated = transform.rotate(imgColor, 45, resize=False)

# Escalar (aumentar o disminuir) una imagen usando un factor
imgScaled = transform.rescale(imgColor, 0.5, channel_axis=-1)

# Trasladar la imagen de forma horizontal o vertical
tform = transform.SimilarityTransform(translation=(10, 10)) # en x e y
imgTranslated = transform.warp(imgColor, tform, preserve_range=True)

# Ajusta contraste de la imagen con un factor
imgContrast = exposure.adjust_gamma(imgColor, gamma=1/factor)

# Guardar una imagen usando la extensión del archivo
io.imsave(DATOS DIR + 'Casitas.png', imgColor)
```

Ver Imagenes_con_skimage.ipynb

Dificultades en el procesamiento de imágenes

Ángulo de visión



Escala



Deformación



Oclusión



Condiciones de iluminación



Confusión de fondo



Variación intra-clase



Extracción de características

Extracción de características de bajo nivel

- Identificar y capturar detalles y patrones fundamentales de la información visual en la imagen.
- Estas características suelen ser básicas, como bordes, texturas y colores.
- Se utilizan como bloques de construcción para análisis más avanzados en tareas como reconocimiento de objetos o clasificación de imágenes.



Extracción de características

Extracción de características de alto nivel

- ❑ Identifica y captura características abstractas y semánticamente significativas.
- ❑ Estas características pueden incluir objetos, patrones complejos, o conceptos abstractos como la presencia de ciertos objetos o escenas.
- ❑ Es un proceso que implica la combinación y comprensión de múltiples características de bajo nivel para formar representaciones más avanzadas.
- ❑ Útiles para tareas como reconocimiento de objetos o clasificación de imágenes.



Binarización

- Binarización o umbralización es la conversión de una imagen a escala de grises o color a una monocromática (blanco y negro)
- Divide la imagen en blanco y negro usando un umbral explorando características locales o globales



Binarización

```
import matplotlib.pyplot as plt
from skimage import data

#Lee una imagen en tonos de grises
camera = data.camera()
binaria1 = camera > 150
binaria2 = camera > 50

plt.figure(figsize=(11, 4))
plt.subplot(131)
plt.imshow(camera, cmap='gray', interpolation='nearest')
plt.title("Original")

plt.subplot(132)
plt.imshow(binaria1, cmap='gray', interpolation='nearest')
plt.title("Umbral 150")

plt.subplot(133)
plt.imshow(binaria2, cmap='gray', interpolation='nearest')
plt.title("Umbral 50")

plt.tight_layout()
plt.show()
```

Original



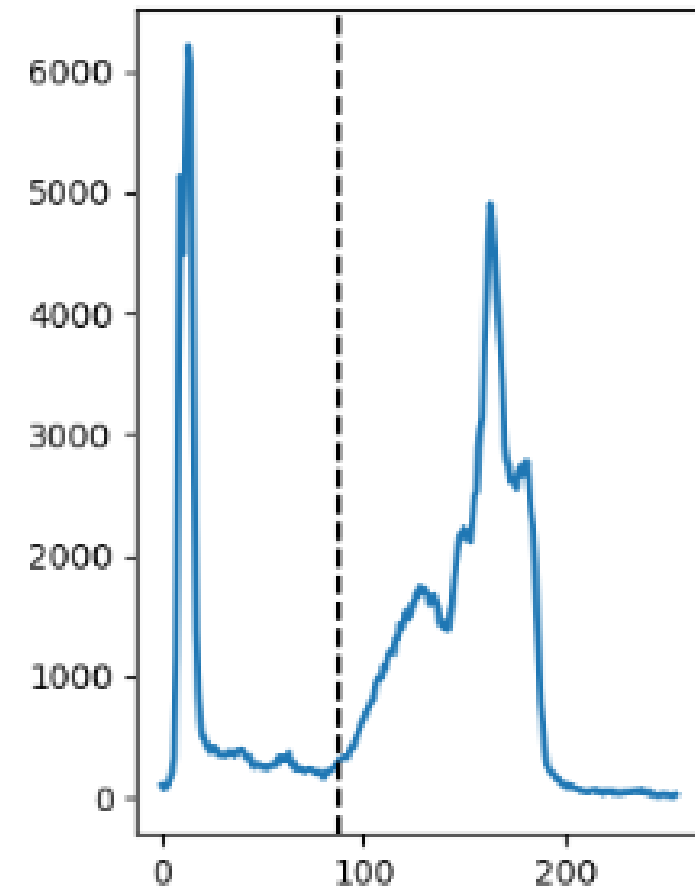
Umbral 150



Umbral 50



Binarización usando el umbral de Otsu



Binarización usando el umbral de Otsu

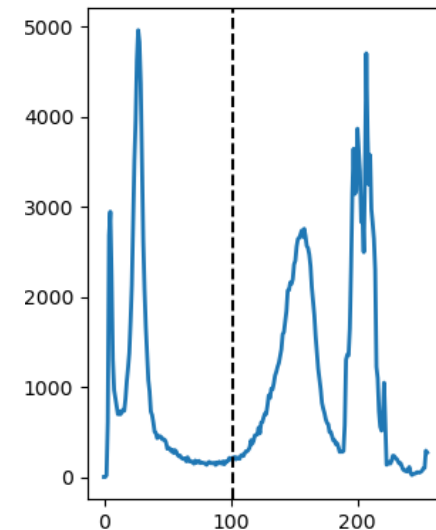
```
import matplotlib.pyplot as plt
from skimage import data, filters, exposure

camera = data.camera()
val = filters.threshold_otsu(camera)

hist, bins_center = exposure.histogram(camera)

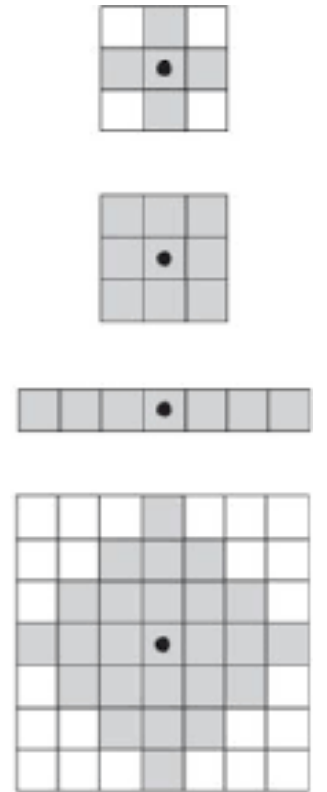
plt.figure(figsize=(9, 4))
plt.subplot(131)
plt.imshow(camera, cmap='gray', interpolation='nearest')
plt.axis('off')
plt.subplot(132)
plt.imshow(camera < val, cmap='gray', interpolation='nearest')
plt.axis('off')
plt.subplot(133)
plt.plot(bins_center, hist, lw=2)
plt.axvline(val, color='k', ls='--')

plt.tight_layout()
plt.show()
```



Operaciones morfológicas

- Las operaciones morfológicas son técnicas que se aplican a la forma o estructura de los objetos en una imagen.
- Se aplican a conjuntos de píxeles en una imagen con el objetivo de modificar su forma, tamaño o características.
- Utiliza una pequeña ventana (kernel) que se desplaza sobre la imagen y según el grado de coincidencia se agrega o elimina un pixel (según la operación).
- Aplicaciones típicas
 - ▣ Eliminar ruido
 - ▣ Segmentar objetos
 - ▣ Rellenar huecos



Operaciones morfológicas

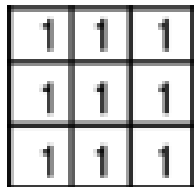
- Erosión: reducción del tamaño de los objetos.
- Dilatación: aumento del tamaño de los objetos.
- Apertura: Combinación de erosión seguida de dilatación, útil para eliminar ruido y separar objetos cercanos.
- Cierre: Combinación de dilatación seguida de erosión, útil para cerrar pequeños huecos en objetos.



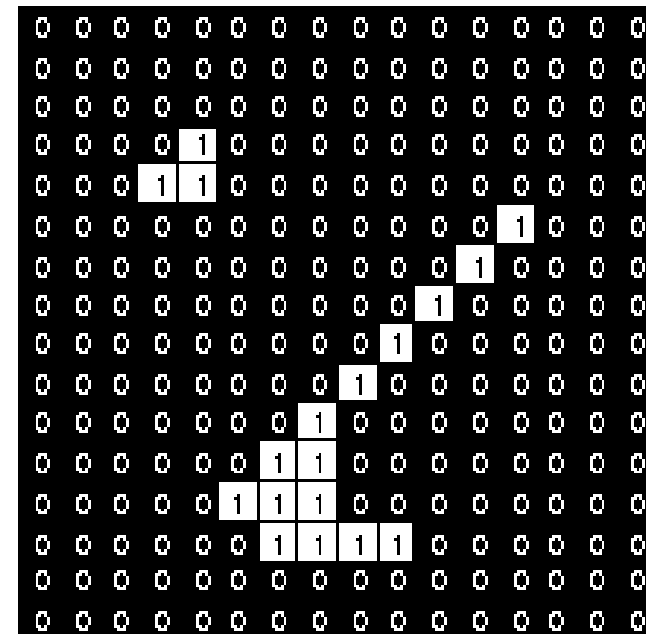
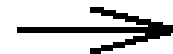
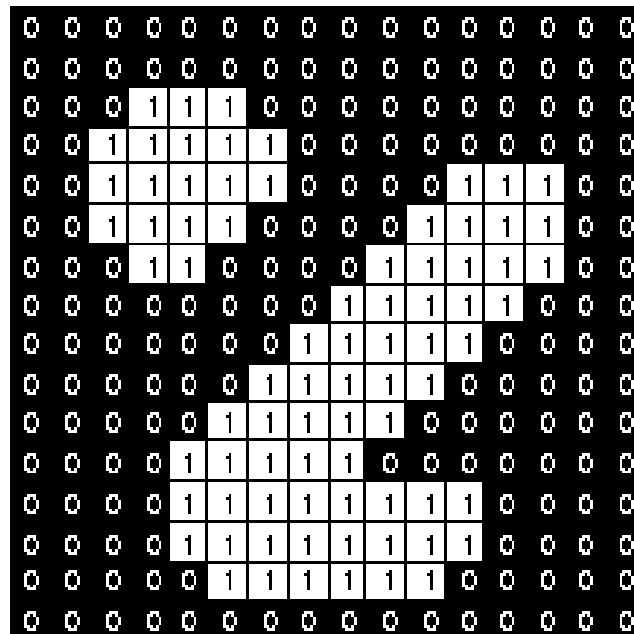
Operaciones morfológicas

□ Erosión con kernel de 3x3

Cuando no coinciden los píxeles de la imagen con los del kernel, elimina pixel central del kernel en la imagen.

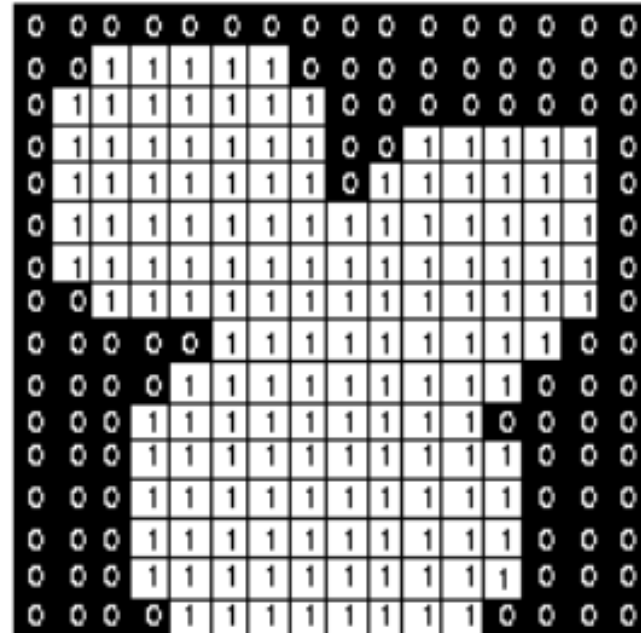
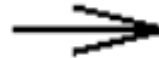
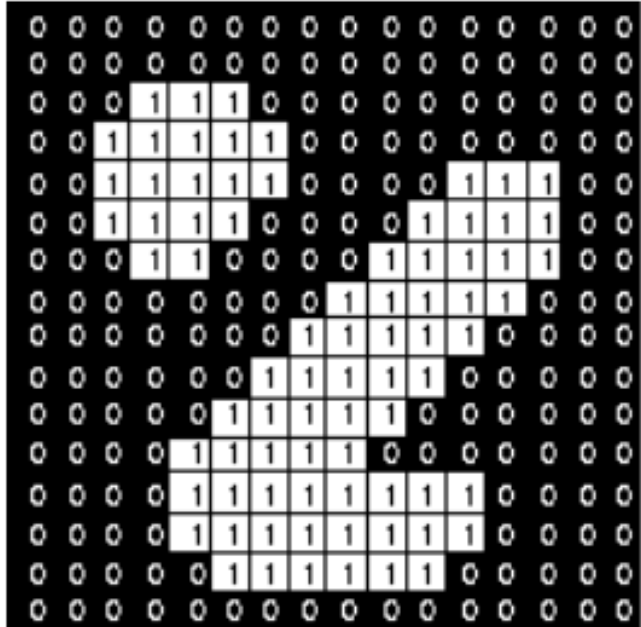


Kernel 3x3



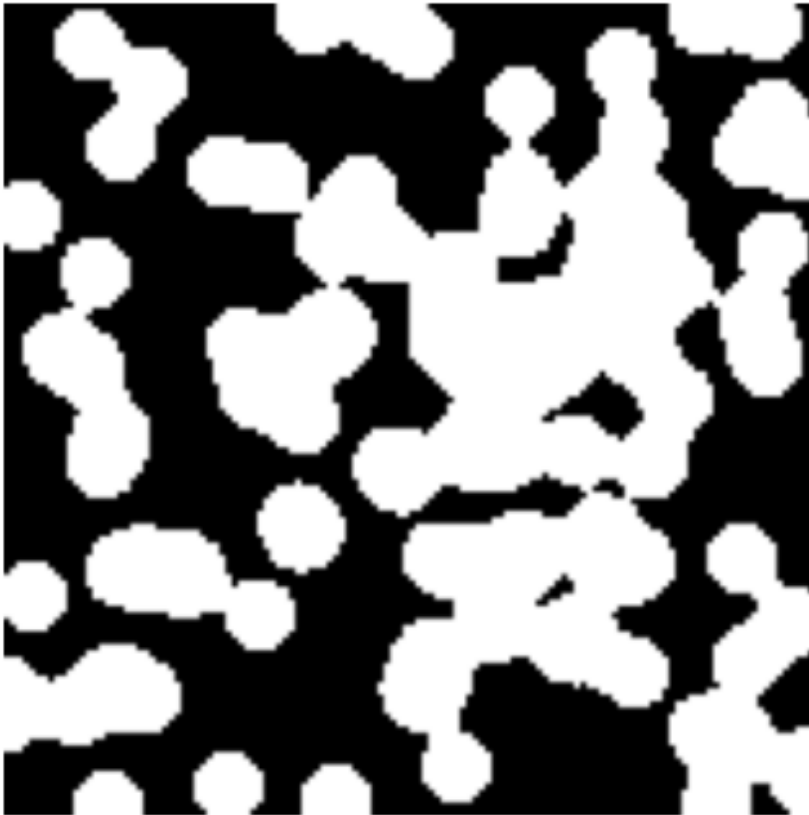
□ Dilatación con kernel de 3x3

Kernel 3x3

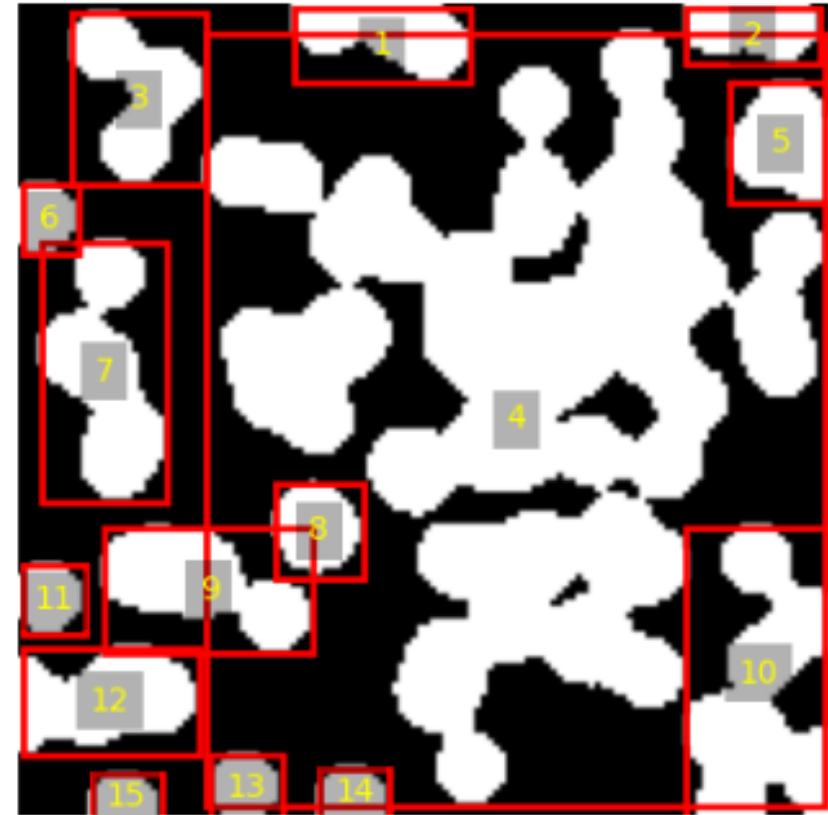


Identificación de regiones

Imagen original



Regiones identificadas



Segmentacion_con_regionprops.ipynb

Filtros en el dominio espacial

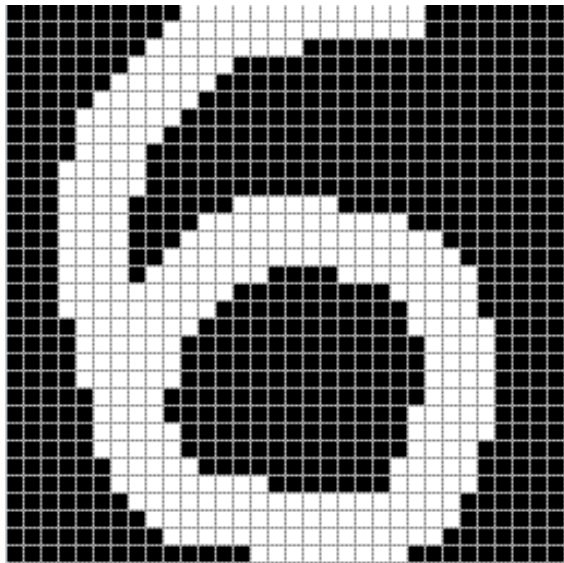
- El filtrado es una técnica para modificar o mejorar una imagen. Por ejemplo, se puede filtrar una imagen para realzar ciertas características o eliminar otras.



*Usaremos máscaras o kernels de **convolución***

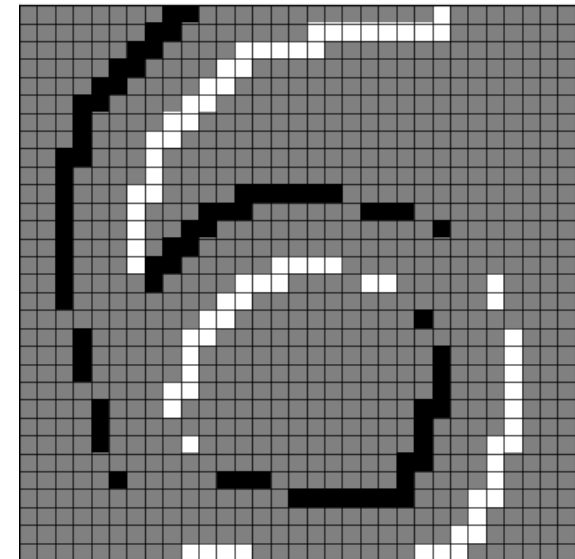
Convolución 2D

- La operación de convolución de una imagen con un filtro o kernel permite destacar ciertas características de dicha imagen.



1	0
0	-1

*Filtro de detección
de bordes*

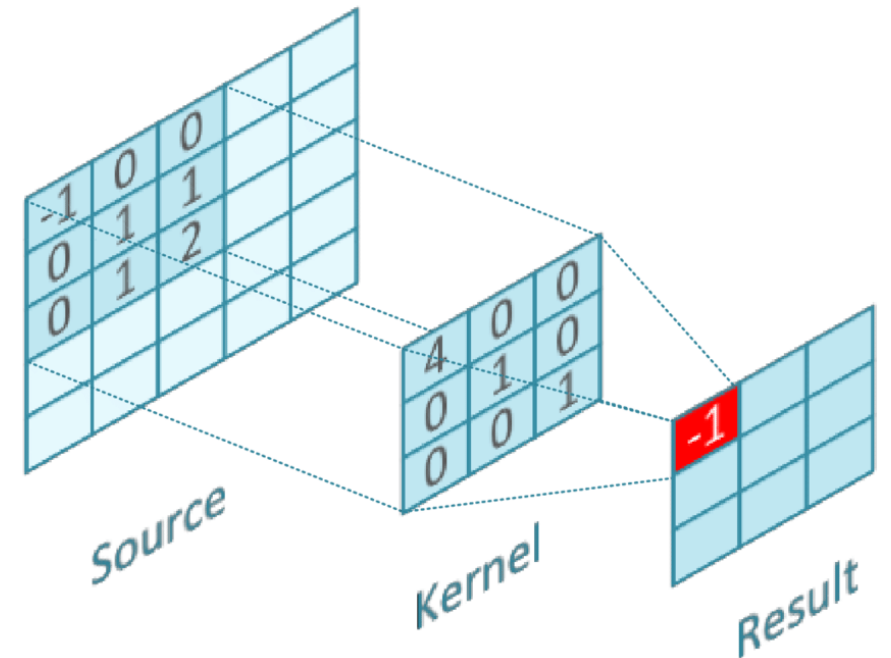


Convolución 2D

- La convolución discreta de dos funciones f y g se define como

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

- La función g se desplaza antes de multiplicar.



Convolución 2D

Entrada

1	2	8	2	1
1	4	9	3	1
1	2	8	2	1
1	3	7	1	2
1	2	8	2	2

Kernel

-1	0	1
-1	0	1
-1	0	1

Salida

	22			

$$\begin{aligned} &1 * (-1) + 2 * 0 + 8 * 1 + \\ &1 * (-1) + 4 * 0 + 9 * 1 + \\ &1 * (-1) + 2 * 0 + 8 * 1 = 22 \end{aligned}$$

Convolución 2D

Entrada

1	2	8	2	1
1	4	9	3	1
1	2	8	2	1
1	3	7	1	2
1	2	8	2	2

Kernel

-1	0	1
-1	0	1
-1	0	1

Salida

	22	-1		

$$\begin{aligned} &2 * (-1) + 8 * 0 + 2 * 1 + \\ &4 * (-1) + 9 * 0 + 3 * 1 + \\ &2 * (-1) + 8 * 0 + 2 * 1 = -1 \end{aligned}$$

Convolución 2D

Entrada

1	2	8	2	1
1	4	9	3	1
1	2	8	2	1
1	3	7	1	2
1	2	8	2	2

Kernel

-1	0	1
-1	0	1
-1	0	1

Salida

	22	-1	-22	

$$\begin{aligned} &8 * (-1) + 2 * 0 + 1 * 1 + \\ &9 * (-1) + 3 * 0 + 1 * 1 + \\ &8 * (-1) + 2 * 0 + 1 * 1 = -22 \end{aligned}$$

Convolución 2D

Entrada

1	2	8	2	1
1	4	9	3	1
1	2	8	2	1
1	3	7	1	2
1	2	8	2	2

Kernel

-1	0	1
-1	0	1
-1	0	1

Salida

	22	-1	-22	
	21			

$$\begin{aligned} &1 * (-1) + 4 * 0 + 9 * 1 + \\ &1 * (-1) + 2 * 0 + 8 * 1 + \\ &1 * (-1) + 3 * 0 + 7 * 1 = 21 \end{aligned}$$

Convolución 2D

Entrada

1	2	8	2	1
1	4	9	3	1
1	2	8	2	1
1	3	7	1	2
1	2	8	2	2

Kernel

-1	0	1
-1	0	1
-1	0	1

Salida

	22	-1	-22	
	21	-3		

$$\begin{aligned} &4 * (-1) + 9 * 0 + 3 * 1 + \\ &2 * (-1) + 8 * 0 + 2 * 1 + \\ &3 * (-1) + 7 * 0 + 1 * 1 = -3 \end{aligned}$$

Convolución 2D

Entrada

1	2	8	2	1
1	4	9	3	1
1	2	8	2	1
1	3	7	1	2
1	2	8	2	2

Kernel

-1	0	1
-1	0	1
-1	0	1

Salida

	22	-1	-22	
	21	-3	-20	

$$\begin{aligned} &9 * (-1) + 3 * 0 + 1 * 1 + \\ &8 * (-1) + 2 * 0 + 1 * 1 + \\ &7 * (-1) + 1 * 0 + 2 * 1 = -20 \end{aligned}$$

Convolución 2D

Entrada

1	2	8	2	1
1	4	9	3	1
1	2	8	2	1
1	3	7	1	2
1	2	8	2	2

Kernel

-1	0	1
-1	0	1
-1	0	1

Salida

	22	-1	-22	
	21	-3	-20	
	20			

$$\begin{aligned} &1 * (-1) + 2 * 0 + 8 * 1 + \\ &1 * (-1) + 3 * 0 + 7 * 1 + \\ &1 * (-1) + 2 * 0 + 8 * 1 = 20 \end{aligned}$$

Convolución 2D

Entrada

1	2	8	2	1
1	4	9	3	1
1	2	8	2	1
1	3	7	1	2
1	2	8	2	2

Kernel

-1	0	1
-1	0	1
-1	0	1

Salida

	22	-1	-22	
	21	-3	-20	
	20	-2		

$$\begin{aligned} &2 * (-1) + 8 * 0 + 2 * 1 + \\ &3 * (-1) + 7 * 0 + 1 * 1 + \\ &2 * (-1) + 8 * 0 + 2 * 1 = -2 \end{aligned}$$

Convolución 2D

Entrada

1	2	8	2	1
1	4	9	3	1
1	2	8	2	1
1	3	7	1	2
1	2	8	2	2

Kernel

-1	0	1
-1	0	1
-1	0	1

Salida

	22	-1	-22	
	21	-3	-20	
	20	-2	-18	

$$\begin{aligned} &8 * (-1) + 2 * 0 + 1 * 1 + \\ &7 * (-1) + 1 * 0 + 2 * 1 + \\ &8 * (-1) + 2 * 0 + 2 * 1 = -18 \end{aligned}$$

Convolución 2D

Entrada

1	2	8	2	1
1	4	9	3	1
1	2	8	2	1
1	3	7	1	2
1	2	8	2	2

Kernel

-1	0	1
-1	0	1
-1	0	1

Salida

	22	-1	-22	
	21	-3	-20	
	20	-2	-18	

□ Parámetros

▣ **Kernel_size:** tamaño del filtro o kernel. En este caso = 3

▣ **Stride:** desplazamiento del filtro cada vez que se aplica. En este caso = 1

Convolución 2D

Entrada

1	2	8	2	1
1	4	9	3	1
1	2	8	2	1
1	3	7	1	2
1	2	8	2	2

Kernel

-1	0	1
-1	0	1
-1	0	1

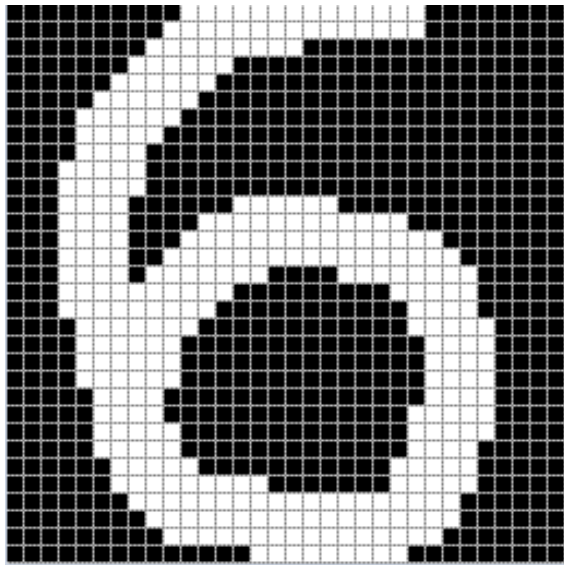
Salida

	22	-1	-22	
	21	-3	-20	
	20	-2	-18	

¿Por qué el resultado de la convolución tiene un tamaño menor al de la entrada?

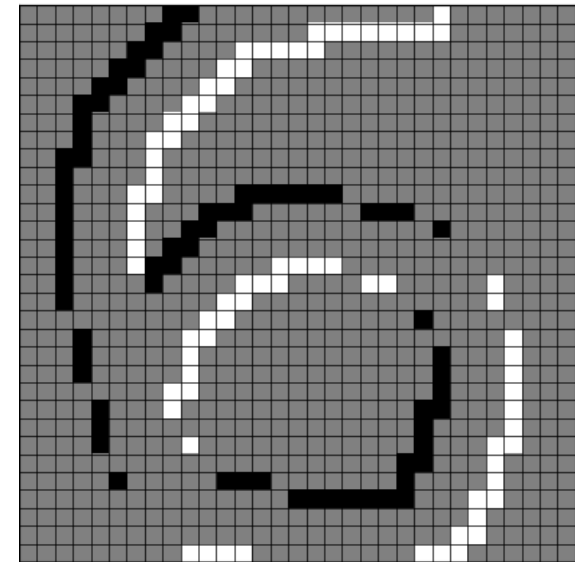
Convolución 2D

- La imagen original es de 32x32. ¿Qué tamaño tiene la imagen de la derecha?
- Analice el resultado de la convolución.



1	0
0	-1

*Filtro de detección
de bordes*



Convolución 2D

- La operación de convolución de una imagen con un filtro o kernel permite destacar ciertas características de dicha imagen.

1	0
0	-1

-1	0
0	1

0	1
-1	0

Imagen Original



Filtro 1



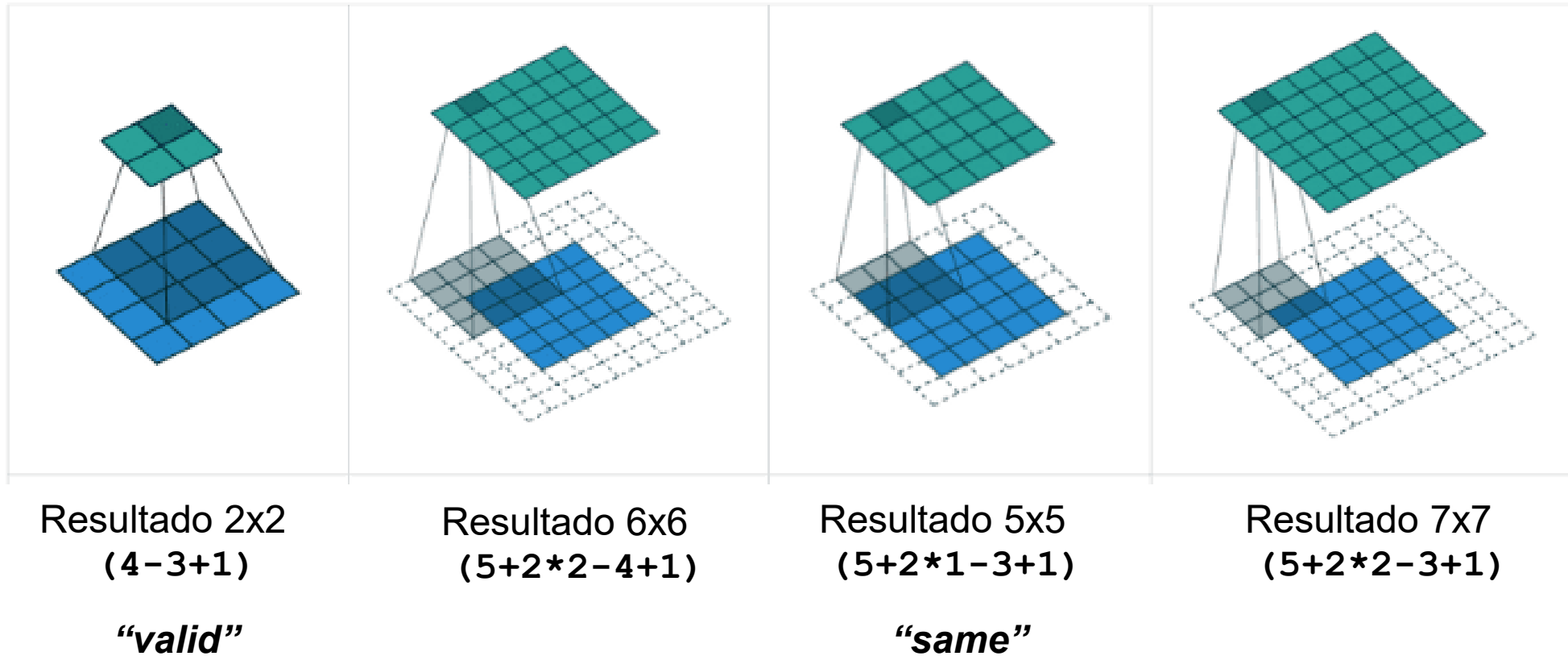
Filtro 2



Filtro 3



Convolución 2D - Padding



Filtros y Extracción de Características

Problema simple: definir filtros para “detectar” la X de ejemplo

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

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

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

Filtros y Extracción de Características

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



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

Filtros y Extracción de Características

$$\begin{aligned} &0 * 1 + 0 * (-1) + 0 * 1 + \\ &0 * (-1) + 1 * 1 + 0 * (-1) + \\ &0 * 1 + 0 * (-1) + 1 * 1 = 2 \end{aligned}$$

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

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

2				


Filtros y Extracción de Características

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



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

$$\begin{aligned} &0 * 1 + 0 * (-1) + 0 * 1 + \\ &1 * (-1) + 0 * 1 + 0 * (-1) + \\ &0 * 1 + 1 * (-1) + 0 * 1 = 2 \end{aligned}$$



2	-2			

Filtros y Extracción de Características

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



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

2	-2	2		

Filtros y Extracción de Características

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



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

2	-2	2	-2	

Filtros y Extracción de Características

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



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

2	-2	2	-2	2

Filtros y Extracción de Características

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



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

2	-2	2	-2	2
-2	3	-3	3	-2
2	-3	5	-3	2
-2	3	-3	3	-2
2	-2	2	-2	2

Filtros y Extracción de Características

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



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

2	-2	2	-2	2
-2	3	-3	3	-2
2	-3	5	-3	2
-2	3	-3	3	-2
2	-2	2	-2	2



RELU

2	0	2	0	2
0	3	0	3	0
2	0	5	0	2
0	3	0	3	0
2	0	2	0	2

Filtros y Extracción de Características

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



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

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



RELU

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

Filtros y Extracción de Características

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



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

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



RELU

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

Filtros y Extracción de Características

KERNELS

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

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

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

2	0	2	0	2
0	3	0	3	0
2	0	5	0	2
0	3	0	3	0
2	0	2	0	2

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

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

SALIDAS

Filtros y Extracción de Características

2	0	2	0	2
0	3	0	3	0
2	0	5	0	2
0	3	0	3	0
2	0	2	0	2

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

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

□ ¿Que representa cada nuevo valor?

Cada nuevo valor representa el grado de coincidencia entre el filtro y la sección correspondiente de la imagen original

Filtros y Extracción de Características

2	0	2	0	2
0	3	0	3	0
2	0	5	0	2
0	3	0	3	0
2	0	2	0	2

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

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

□ ¿El resultado del filtro es una imagen?

Si, pero el nuevo valor es una intensidad relacionada con la coincidencia del filtro

Filtros y Extracción de Características

2	0	2	0	2
0	3	0	3	0
2	0	5	0	2
0	3	0	3	0
2	0	2	0	2

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

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

- Si aplicamos un nuevo filtro al resultado, ¿qué sucede?

Un nuevo filtro relaciona las características de filtros anteriores, agregando un nivel de abstracción en la interpretación de la imagen