Filtros, Convoluciones y Transformaciones Básicas

Antonio Falcó y Juan Pardo

Introducción a la Inteligencia Artificial

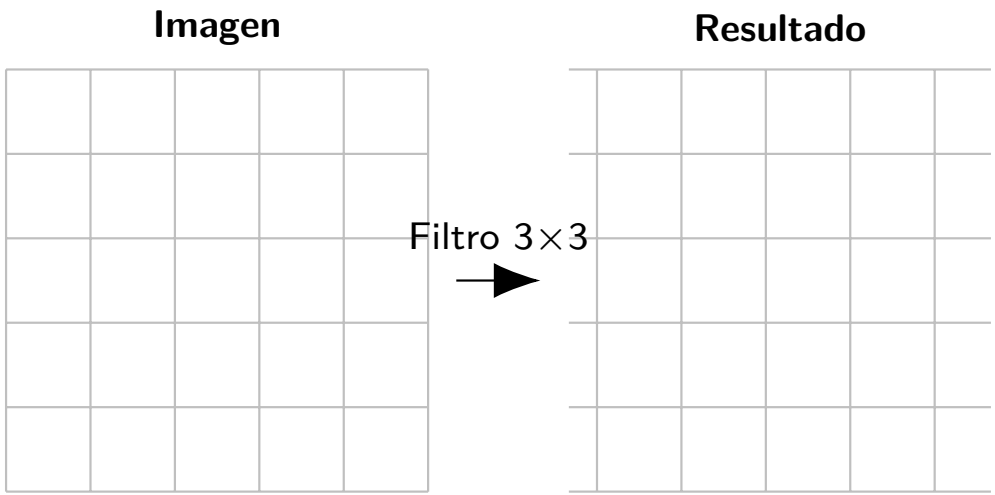
Lecture 2

Objetivos de la sesión

- Comprender qué es un filtro y cómo actúa sobre una imagen.
- Introducir la operación de convolución como base del procesado digital.
- Explorar ejemplos de filtros: media, gaussiano y de bordes (Sobel).
- Aplicar transformaciones geométricas: traslación, rotación, escalado.
- Conectar estas técnicas con el preprocesado para modelos de IA.

Que es un filtro?

- Un filtro transforma una imagen modificando los valores de sus píxeles.
- Se define mediante una pequeña matriz de pesos (**kernel**) que se aplica localmente.
- Los filtros permiten:
  - Suavizar (reducir ruido)
  - Resaltar bordes o texturas
  - Detectar formas



La operación de convolución

Definición discreta

Para una imagen f y un kernel k de tamaño (m, n):

$$(g * f)(x, y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} f(x - i, y - j)\, k(i, j)$$

- Cada píxel se reemplaza por una combinación ponderada de sus vecinos.
- El resultado depende de la forma del kernel.

Ejemplo intuitivo

Un filtro de media 3×3 reemplaza cada píxel por la media de su vecindario inmediato.

Filtro de media

$$K = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

- Suaviza la imagen y reduce el ruido.
- Pierde información de bordes.

En Python

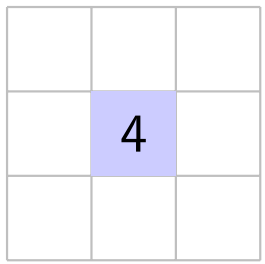
```
K = np.ones((3,3))/9
out = np.zeros_like(img[:,:,0], dtype=float)
for i in range(1, H-1):
    for j in range(1, W-1):
        out[i,j] = np.sum(img[i-1:i+2, j-1:j+2,0] * K)
plt.imshow(out, cmap='gray')
```

Filtro gaussiano

- Usa pesos mayores en el centro que en los bordes.
- Produce un suavizado más natural.

$$K = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Idea visual



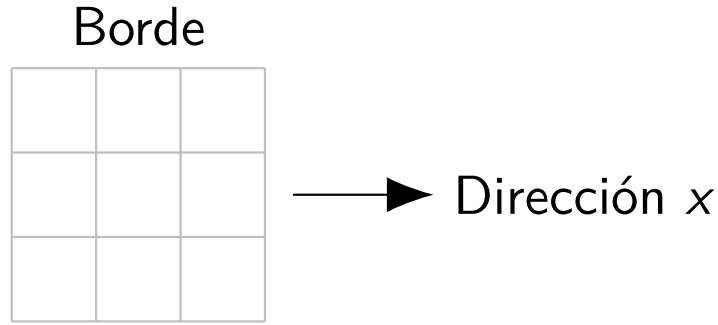
Centro con mayor peso

Detección de bordes (Sobel)

- Resalta los cambios abruptos de intensidad.
- Filtro Sobel horizontal y vertical:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$



Magnitud del gradiente

$$G = \sqrt{(f * G_x)^2 + (f * G_y)^2}$$

Filtro de Sobel: concepto y fórmulas

- **Idea**: aproximar derivadas en x e y con kernels 3×3 para medir cambios de intensidad.
- Kernels de Sobel (horizontal/vertical):

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \qquad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

- **Gradiente**: $g_x = f * G_x,\ g_y = f * G_y$; **magnitud**: $\|\nabla f\| \approx \sqrt{g_x^2 + g_y^2}$.
- **Orientación**: $\theta = \operatorname{atan2}(g_y, g_x)$ (dirección del borde).
- **Normalización**: reescala a $[0, 255]$ para visualizar (evitar saturación).
- **Bordes**: padding (ceros, replicar o reflejar) o ignorar el marco.
- **Ruido**: suele precederse de un suavizado (p. ej. gaussiano 3×3).

Filtro de Sobel: implementación en Python

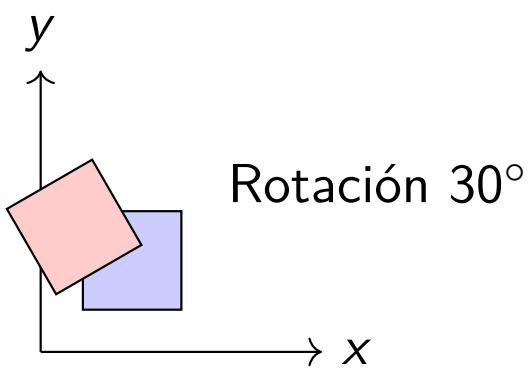
```
import numpy as np
import matplotlib.pyplot as plt
img = plt.imread('imagen.jpg')
# Convertir a escala de grises si es RGB (sin tildes en comentarios)
if img.ndim == 3:
    img_gray = (0.299*img[...,0] + 0.587*img[...,1] + 0.114*img[...,2]).astype(np.float32)
else:
    img_gray = img.astype(np.float32)

H, W = img_gray.shape
Gx = np.array([[-1,0,1],[-2,0,2],[-1,0,1]], dtype=np.float32)
Gy = np.array([[-1,-2,-1],[0,0,0],[1,2,1]], dtype=np.float32)
gx = np.zeros_like(img_gray, dtype=np.float32)
gy = np.zeros_like(img_gray, dtype=np.float32)

for i in range(1, H-1):
    for j in range(1, W-1):
        region = img_gray[i-1:i+2, j-1:j+2]
        gx[i,j] = np.sum(region * Gx)
        gy[i,j] = np.sum(region * Gy)

mag = np.sqrt(gx**2 + gy**2)
# Normalización más segura con pequeño epsilon para evitar división por cero
mag = (255 * (mag / (mag.max() + 1e-8))).astype(np.uint8)

plt.figure(figsize=(10,4))
plt.subplot(1,2,1); plt.imshow(img_gray, cmap='gray'); plt.title('Imagen original'); plt.ax
plt.subplot(1,2,2); plt.imshow(mag, cmap='gray'); plt.title('Magnitud del gradiente (Sobel)
plt.tight_layout()
plt.show()
```

Transformaciones geométricas

- Permiten cambiar la posición o forma de la imagen.
- Se basan en una función de mapeo $T(x, y) = (x', y')$.

Ejemplos

- **Traslación:** $x' = x + a,\ \ y' = y + b$
- **Escalado:** $x' = s_x x,\ \ y' = s_y y$
- **Rotación:** $\begin{bmatrix} x' \\ y' \end{bmatrix} = R(\theta) \begin{bmatrix} x \\ y \end{bmatrix}$



Ejercicio guiado: aplicar un filtro

```
import numpy as np, matplotlib.pyplot as plt

img = plt.imread('imagen.jpg')
img_gray = img[:,:,0].astype(np.float32)  # canal 0 en float
K = np.array([[-1,0,1],[-2,0,2],[-1,0,1]], dtype=np.float32)  # Sobel vertical (Gx)
H,W = img_gray.shape
out = np.zeros_like(img_gray)

for i in range(1,H-1):
    for j in range(1,W-1):
        out[i,j] = np.sum(img_gray[i-1:i+2, j-1:j+2] * K)

# normalización suave para visualizar
m = out.max() if out.max()!=0 else 1.0
plt.imshow((out/m), cmap='gray'); plt.axis('off'); plt.show()
```

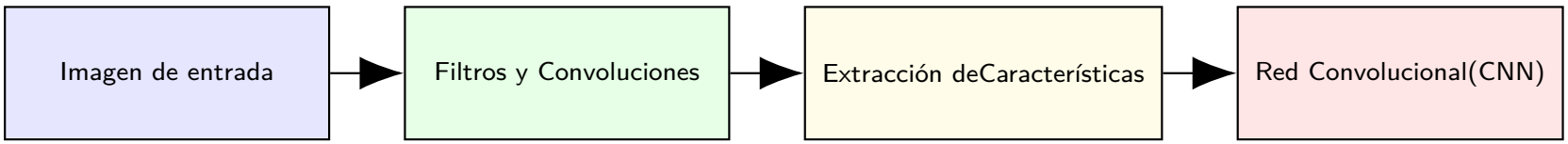
Para pensar

- ¿Qué cambia si usas el Sobel horizontal (Gy)?
- ¿Cómo afecta aplicar antes un gaussiano 3×3?

Conclusiones

- Las convoluciones son la base de la visión por computador.
- Los filtros permiten extraer información relevante de las imágenes.
- Las transformaciones geométricas amplían el repertorio de operaciones visuales.
- Estos conceptos se reutilizan en redes convolucionales (CNNs).

De los filtros a las redes convolucionales



Idea clave

Los mismos principios de los filtros se amplían en las CNNs, donde los pesos del kernel se aprenden automáticamente para extraer características relevantes.