Universidad Autónoma de Aguascalientes

Centro de Ciencias Básicas

Departamento de Ciencias de la Computación

Optativa Profesionalizante II: Machine Learning y Deep Learning

10° "A"

Actividad 6: Convolución

Docente: Dr. Francisco Javier Luna Rosas

Alumno: Joel Alejandro Espinoza Sánchez (211800)

Fecha de Entrega: Aguascalientes, Ags., 3 de abril del 2023.

Actividad 6: Convolución

Convolución

El alumno deberá elaborar un documento `*.pdf` donde implemente una convolución.

El documento debe contener lo siguiente:

- Análisis de la convolución.
- Implementación de la convolución en Python o en el lenguaje de programación de preferencia.
- Evaluación de la convolución en Python o en el lenguaje de programación de preferencia.

El alumno deberá subir a la plataforma el archivo (`*.pdf`) y un documento auto-reproducible (`*.html`) que deberá contener:

- Portada.
- Evidencias de la actividad.
- Conclusiones.
- Referencias (formato APA).

La convolución es una técnica muy utilizada en el procesamiento de imágenes para aplicar filtros. En este contexto, el filtro se llama kernel o máscara de convolución, y su tamaño y forma determinan cómo se aplicará la convolución a la imagen. La convolución es una operación

matemática que se utiliza comúnmente en procesamiento de señales y procesamiento de imágenes para aplicar filtros y realizar operaciones de detección de bordes, suavizado, entre otras. Básicamente, la convolución se realiza al desplazar un pequeño kernel o matriz de pesos sobre toda la imagen de entrada, multiplicando los valores de los píxeles de la imagen por los valores del kernel y sumando los productos. Este proceso se repite para cada píxel de la imagen, generando una nueva imagen de salida convolucionada.

La convolución es un proceso fundamental en la visión por computadora y el procesamiento de imágenes, ya que permite aplicar diferentes filtros y técnicas de detección de bordes y características. Además, se puede utilizar para reducir el ruido y mejorar la calidad de las imágenes, así como para extraer información útil de las mismas. La convolución es ampliamente utilizada en lenguajes de programación como Python, y existen numerosas librerías y herramientas disponibles para realizar convoluciones de manera eficiente y sencilla.

Los filtros convolucionales se utilizan para aplicar efectos visuales a las imágenes, como el desenfoque, la detección de bordes, el realce de características, la eliminación de ruido y otros efectos similares. Los filtros convolucionales también se utilizan en la extracción de características, como en el caso de los filtros de Gabor, que se utilizan para detectar patrones específicos en las imágenes.

En Python, la biblioteca OpenCV ofrece una amplia gama de funciones de convolución que permiten aplicar diferentes tipos de filtros a las imágenes. Además, la biblioteca Tensorflow, junto con Keras, también ofrece herramientas para construir redes convolucionales profundas que pueden aprender a aplicar filtros a las imágenes automáticamente.

Su implementación en código se realiza a continuación, primeramente importando las librerías necesarias:

In [1]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

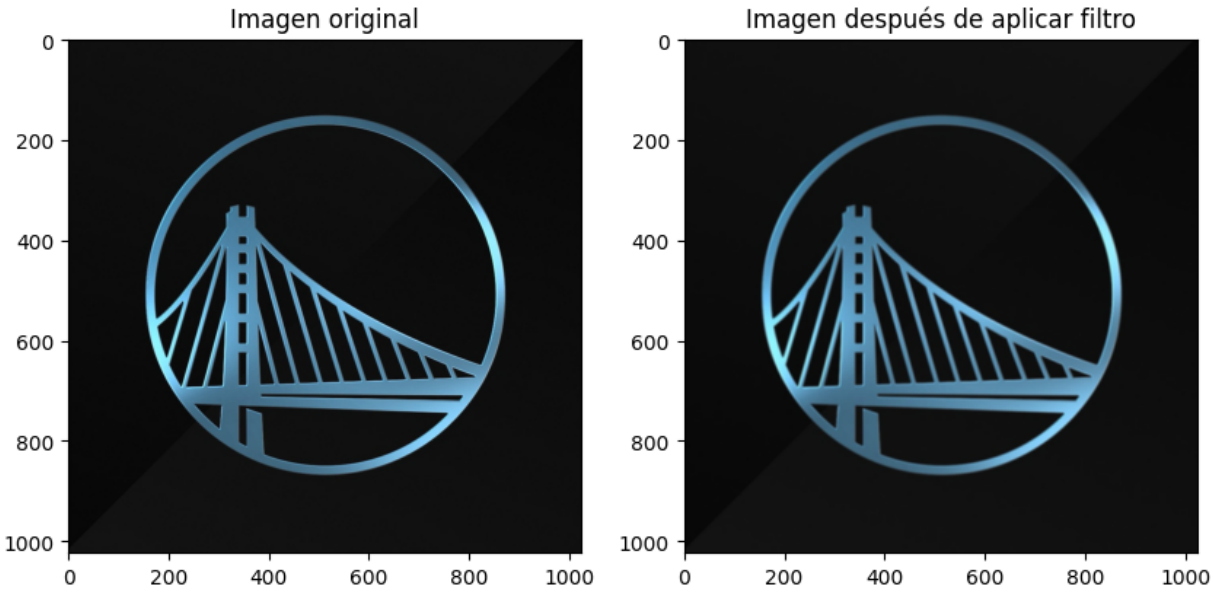
A continuación se muestra la primera convolución antes y después de realizarse:

In [2]:
```
# Cargar imagen
img = cv2.imread("imagen.jpg")

# Crear kernel de desenfoque
kernel = np.ones((5,5),np.float32)/25

# Aplicar filtro de desenfoque
blurred = cv2.filter2D(img,-1,kernel)

# Mostrar imagenes
fig, axs = plt.subplots(1, 2, figsize=(10, 10))
axs[0].imshow(img, cmap="gray")
axs[0].set_title("Imagen original")
axs[1].imshow(blurred, cmap="gray")
axs[1].set_title("Imagen después de aplicar filtro")
plt.show()
```



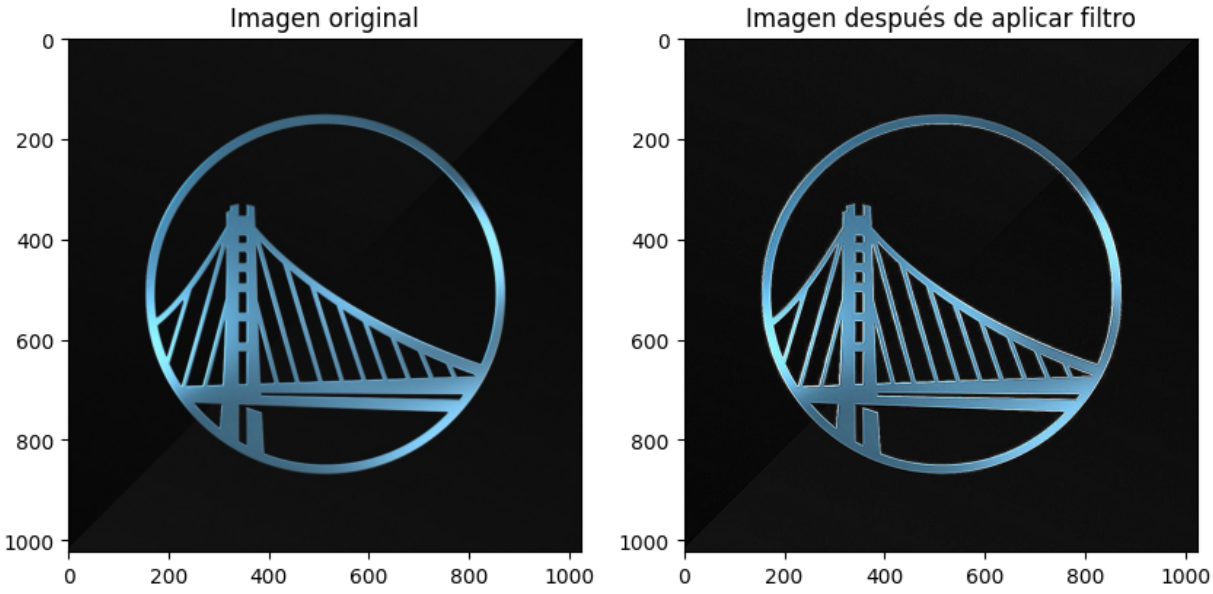
A continuación la segunda convolución:

In [3]:
```
# Cargar imagen
img = cv2.imread("imagen.jpg")

# Crear kernel de realce de bordes
kernel = np.array([[-1,-1,-1], [-1,9,-1], [-1,-1,-1]])

# Aplicar filtro de realce de bordes
edges = cv2.filter2D(img,-1,kernel)

# Mostrar imagenes
fig, axs = plt.subplots(1, 2, figsize=(10, 10))
axs[0].imshow(img, cmap="gray")
axs[0].set_title("Imagen original")
axs[1].imshow(edges, cmap="gray")
axs[1].set_title("Imagen después de aplicar filtro")
plt.show()
```



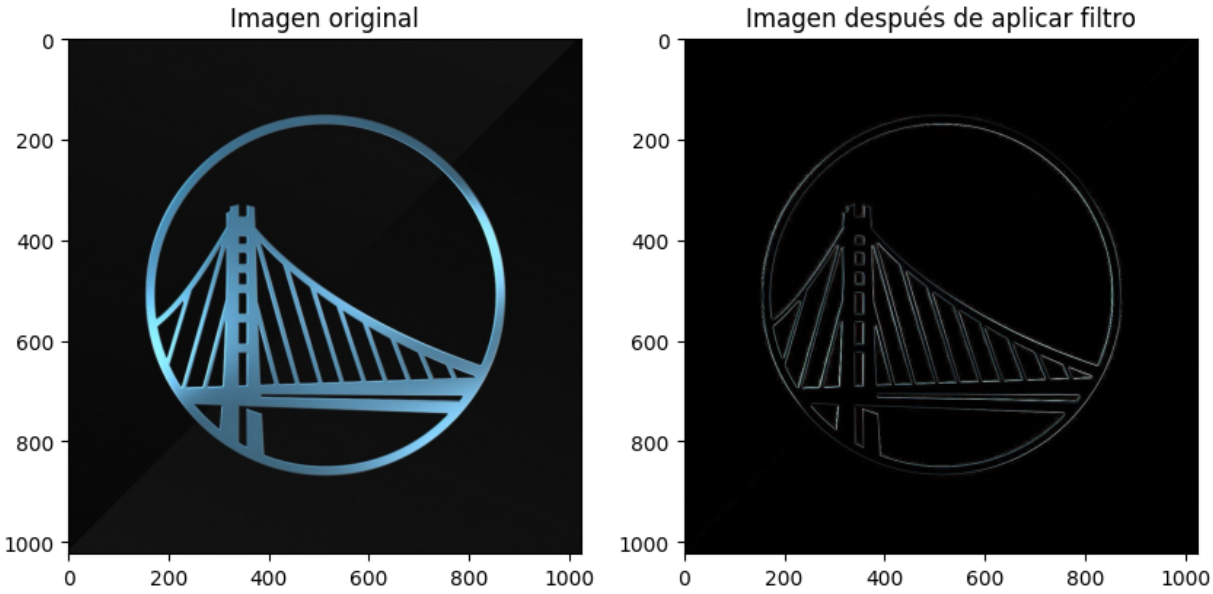
A continuación la tercera convolución:

In [4]:
```
# Cargar imagen
img = cv2.imread("imagen.jpg")

# Crear kernel de detección de bordes
kernel = np.array([[-1,-1,-1], [-1,8,-1], [-1,-1,-1]])

# Aplicar filtro de detección de bordes
edges = cv2.filter2D(img,-1,kernel)

# Mostrar imagenes
fig, axs = plt.subplots(1, 2, figsize=(10, 10))
axs[0].imshow(img, cmap="gray")
axs[0].set_title("Imagen original")
axs[1].imshow(edges, cmap="gray")
axs[1].set_title("Imagen después de aplicar filtro")
plt.show()
```



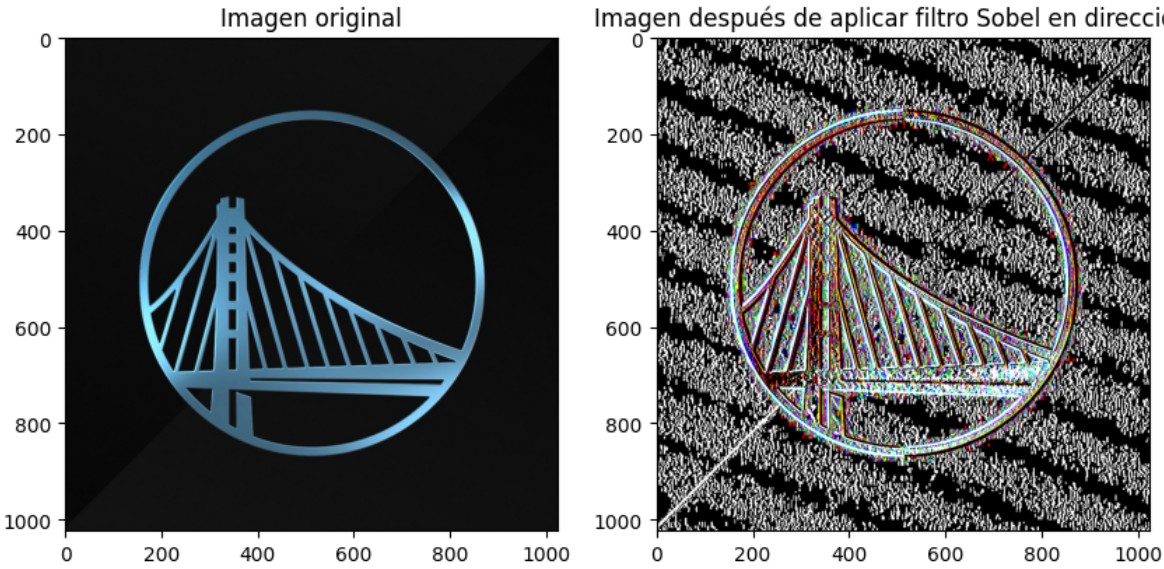
A continuación la cuarta convolución:

In [5]:
```
# Cargar imagen
img = cv2.imread("imagen.jpg")

# Aplicar filtro de sobel
sobelx = cv2.Sobel(img, cv2.CV_64F, 1, 0, ksize=5)

# Mostrar imagenes
fig, axs = plt.subplots(1, 2, figsize=(10, 10))
axs[0].imshow(img, cmap="gray")
axs[0].set_title("Imagen original")
axs[1].imshow(sobelx, cmap="gray")
axs[1].set_title("Imagen después de aplicar filtro Sobel en dirección x")
plt.show()
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
```



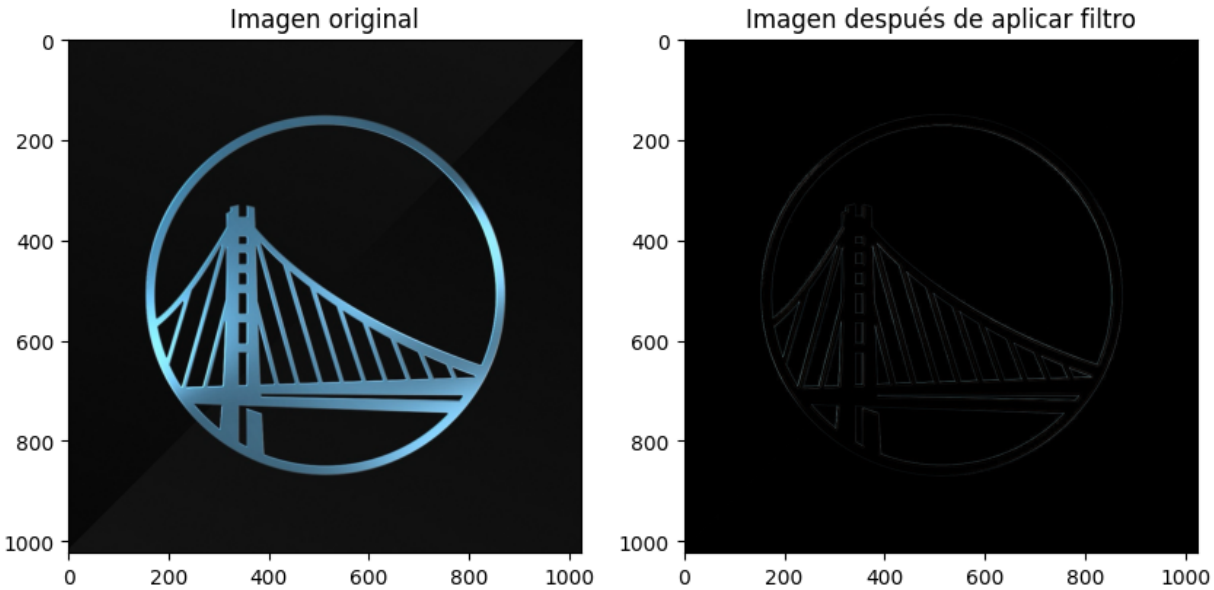
A continuación la quinta convolución:

In [6]:
```
# Cargar imagen
img = cv2.imread("imagen.jpg")

# Crear kernel laplaciano
kernel = np.array([[0,1,0], [1,-4,1], [0,1,0]])

# Aplicar filtro laplaciano
edges = cv2.filter2D(img,-1,kernel)

# Mostrar imagenes
fig, axs = plt.subplots(1, 2, figsize=(10, 10))
axs[0].imshow(img, cmap="gray")
axs[0].set_title("Imagen original")
axs[1].imshow(edges, cmap="gray")
axs[1].set_title("Imagen después de aplicar filtro")
plt.show()
```



Conclusiones

Es interesante e importante poder implementar las bases de estos temas para entenderlos en un futuro, pues, posteriormente no basta con sólo importar librerías que realicen el trabajo pesado, ya que, implementar manualmente las funciones matemáticas en las que recaen estos algoritmos nos enseña a qué hay detrás del algoritmo, cómo funciona y poder comprender realmente qué está ocurriendo como la base de una red neuronal y la forma en la que ésta aprende realizando el descenso del gradiente. Es muy útil la implementación de estos algoritmos en estas tareas para las futuras tareas de la materia y aplicaciones de Machine Learning en la vida personal.

Referencias

- Anónimo (s.f.) “Red neuronal artificial”. Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Red_neuronal_artificial.
- Data Scientest (2021) “Perceptrón. ¿Qué es y para qué sirve?”. Obtenido de Data Scientest: https://datascientest.com/es/perceptron-que-es-y-para-que-sirve.
- Luna, F. (2023) “El Modelo de McCulloch – Pitts”. Apuntes de ICI 10°.