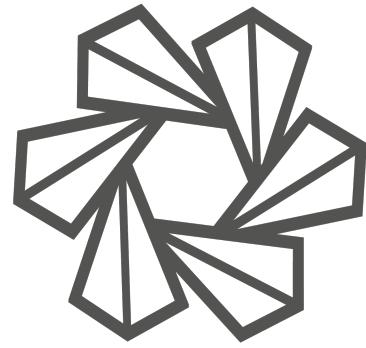
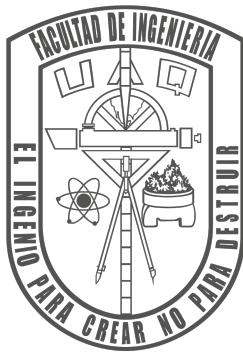
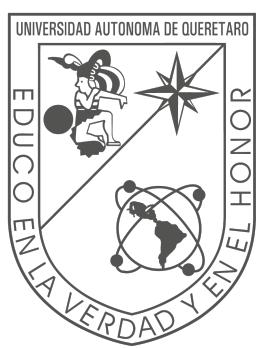


Universidad Autónoma de Querétaro

Facultad de Ingeniería
División de Investigación y Posgrado



Reporte 6
Convoluciones

Maestría en Ciencias en Inteligencia Artificial
Optativa de especialidad II - Deep Learning

Aldo Cervantes Marquez

Expediente: 262775

Profesor: Dr. Sebastián Salazar Colores

Santiago de Querétaro, Querétaro, México

Semestre 2022-2

17 de Septiembre de 2022

Índice

1. Introducción	1
2. Marco Teórico	1
2.1. RGB	1
2.2. Manejo de Imagenes con Python	1
2.3. Convoluciones	2
2.4. Uso de Convoluciones en la Práctica	2
3. Justificación	3
4. Resultados	3
4.1. Convolución 1	3
4.2. Convolución 2	4
4.3. Convolución 3	4
4.4. Convolución 4	5
4.5. Convolución 5	5
4.6. Convolución 6	6
5. Conclusiones	6
Referencias	7
6. Anexo: Programa completo en Google Colab	8

1. Introducción

La presente práctica consiste en el uso de convoluciones o máscaras para el acondicionamiento y modificación de las imágenes píxel a píxel por cada gama de color (RGB) [1]. Con el fin de poder formalizar el entendimiento del uso de las convoluciones en imágenes, pudiendo extraer mejor información y tener una mejor apreciación de manejo de la información que nos puede interesar de una imagen, hablando desde el punto de vista del filtrado y categorización de datos para entrenar una red, o para observar espectros de colores, etc.

2. Marco Teórico

2.1. RGB

Consistente en un código de valores dividido en 3 partes según su intensidad (Rojo, Verde y Azul). Teniendo valores entre 0 y 255. Por lo que se puede tener 255^3 posibilidades de colores (16777216). En donde la combinación (0,0,0) es el color negro y la combinación (255,255,255) es el color blanco veasé Figura 1. [2]

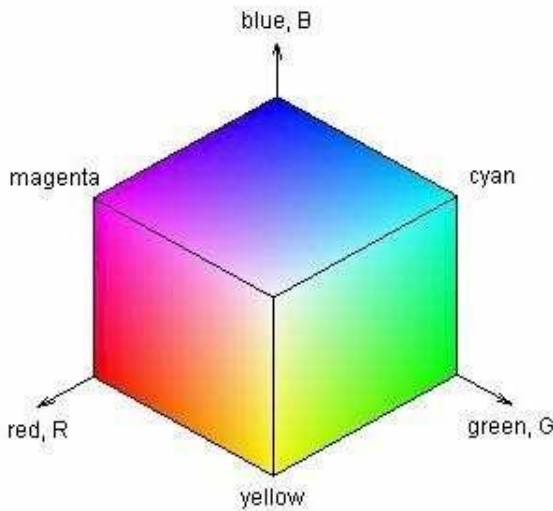


Figura 1: Paleta cúbica de colores RGB.

2.2. Manejo de Imágenes con Python

Existen principalmente 2 comandos para poder visualizar una imagen en Python, entre las que destacan las librerías de *Opencv* y de *matplotlib.image*, en donde ambos tenían comandos parecidos únicamente cambiando el objeto creado de la librería [3].

```
img = mpimg.imread()
```

$$img = cv2.imread()$$

Siendo la única función exclusiva de la librería que se utilizó. La imagen que se utilizó, fue una imagen de un paisaje pintado con una dimensión de (775, 1600, 3) (véase Figura 2).



Figura 2: Imagen a aplicar convolución.

2.3. Convoluciones

Las convoluciones constan de kernels que permiten modificar la imagen píxel a píxel, generalmente dichos kernels representan una matriz de valores, los cuales interactúan con la cantidad de datos en igual forma para realizar la convolución (véase Figura 3) [4, 5].

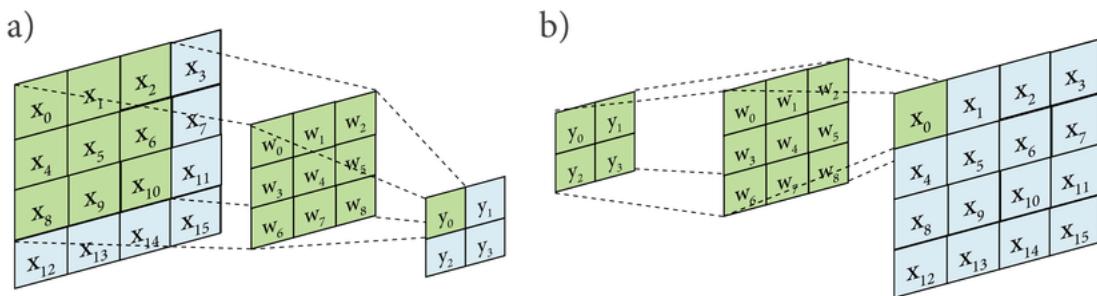


Figura 3: Paleta cúbica de colores RGB.

2.4. Uso de Convoluciones en la Práctica

Sea el kernel definido como k y los píxeles de la imagen img como $p_{x,y}$ suponiendo que el color RGB no es de importancia para este caso:

$$k = \begin{bmatrix} c_{11} & \cdots & c_{1n} \\ \vdots & \ddots & \vdots \\ c_{n1} & \cdots & c_{nn} \end{bmatrix} \quad (1)$$

$$img = \begin{bmatrix} p_{11} & \cdots & p_{1n} \\ \vdots & \ddots & \vdots \\ p_{n1} & \cdots & p_{nn} \end{bmatrix} \quad (2)$$

Entonces la convolución para obtener un nuevo píxel np se realiza de la siguiente manera:

$$np_{x,y} = \begin{bmatrix} p_{11} & \cdots & p_{1n} \\ \vdots & \ddots & \vdots \\ p_{n1} & \cdots & p_{nn} \end{bmatrix}_{ren \times col} \cdot \begin{bmatrix} c_{11} & \cdots & c_{1n} \\ \vdots & \ddots & \vdots \\ c_{n1} & \cdots & c_{nn} \end{bmatrix}_{ren \times col} = \sum_{i=1,j=1}^{ren,col} p_{i,j} \times c_{i,j} \quad (3)$$

Siendo la matriz de píxeles de la misma dimensión $[a \times b]$

3. Justificación

El uso de convoluciones tiene utilidad para poder reducir dimensiones de las imágenes, resaltar contornos y mejorar calidad o enfocar áreas de interés según sea el caso. Por lo que son de gran importancia para utilizarse en el análisis de la información que se pueda obtener a partir de ellas [6].

4. Resultados

Primeramente se observó que la librería de *opencv* como *cv2* modificaba la imagen visualmente en Jupyter Notebook de una manera desconocida, por eso fue que se prefirió utilizar la librería *matplotlib.image*.

4.1. Convolución 1

$$K_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (4)$$

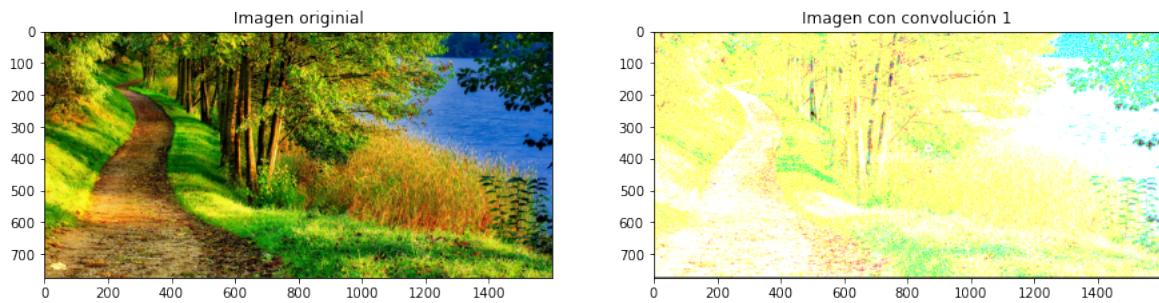


Figura 4: Resultado convolución 1.

4.2. Convolución 2

$$K_2 = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix} \quad (5)$$

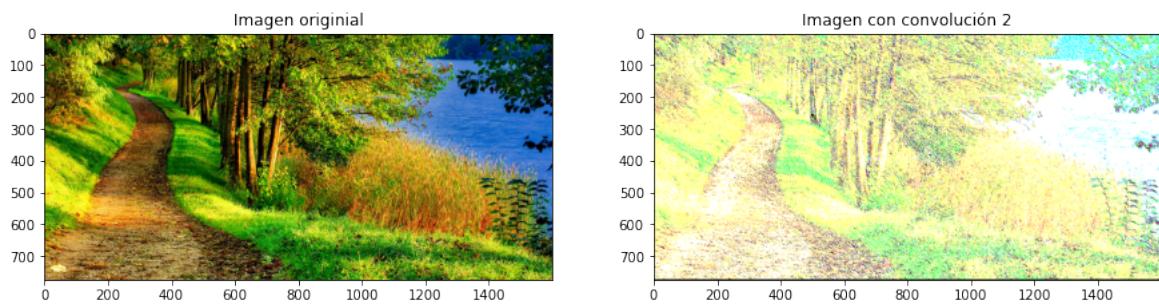


Figura 5: Resultado convolución 2.

4.3. Convolución 3

$$K_3 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad (6)$$

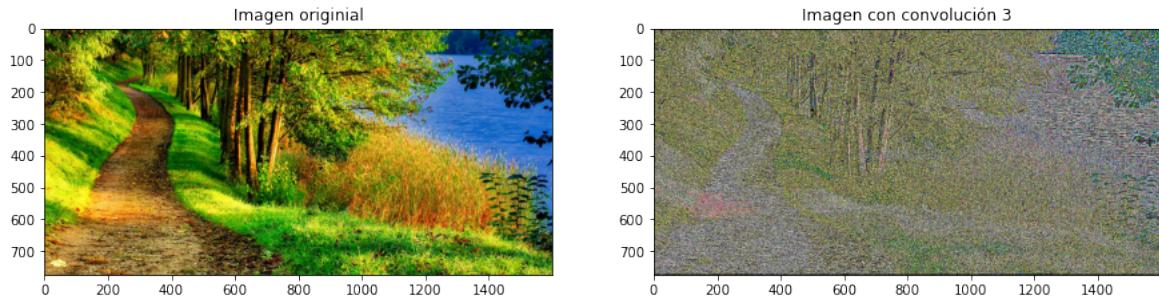


Figura 6: Resultado convolución 3.

4.4. Convolución 4

$$K_4 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad (7)$$

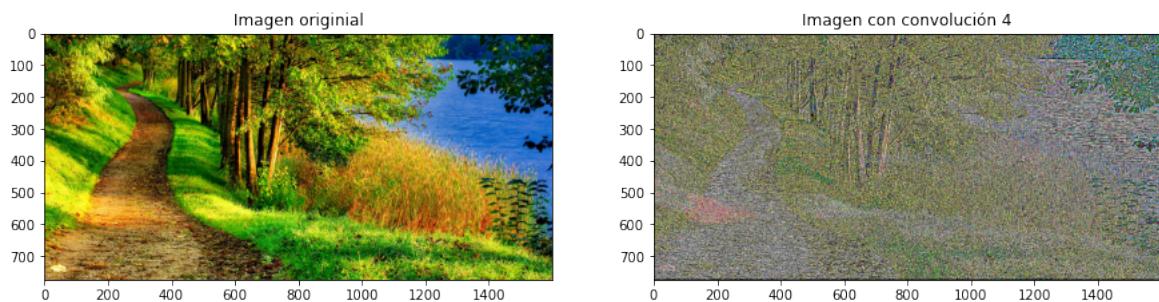


Figura 7: Resultado convolución 4.

4.5. Convolución 5

$$K_5 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad (8)$$

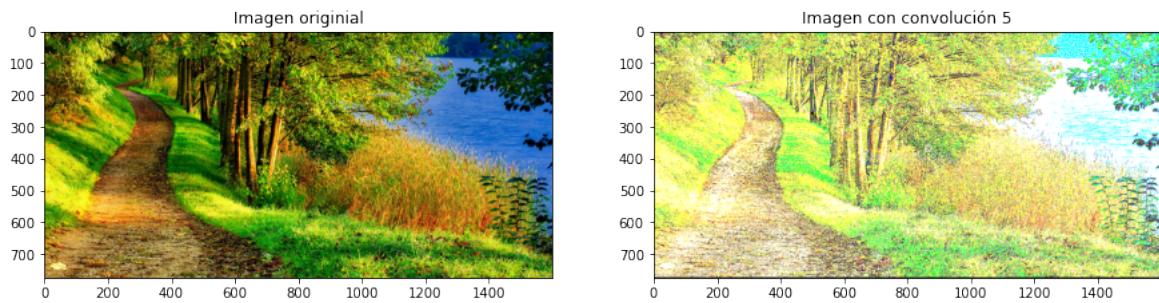


Figura 8: Resultado convolución 5.

4.6. Convolución 6

$$K_6 = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad (9)$$

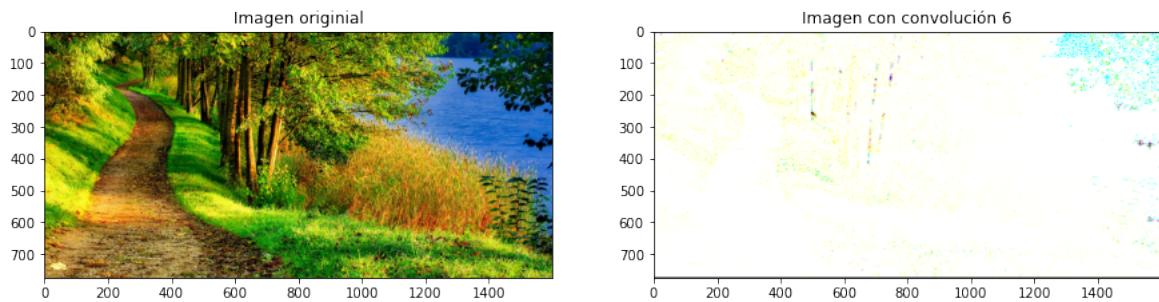


Figura 9: Resultado convolución 6.

5. Conclusiones

1. Se pudo observar desde la utilización de las librerías que podrían haber cambios en el aspecto de la imagen al momento de leerla.
2. En la primer convolución (Figura 4) se observa que la imagen se quemó totalmente puesto que al agregar muchos ceros, se blanquea la imagen (debido a lo mostrado anteriormente en el marco teórico).
3. En la segunda convolución (Figura 5) se observa que igual se blanqueo la imagen, pero en menor medida, debido a que los -1 y 1 permitieron disminuir el valor del pixel y hacer que se pudiera apreciar de mejor manera que en la convolución 1.

4. En la tercera convolución (Figura 6) se observa que se hicieron una especie de gránulos, pues al sumar los valores del kernel se obtiene un 0 y por lo tanto cuando los valores cercanos sean muy cercanos, se obtendrá un 0 (blanco).
5. En la cuarta convolución (Figura 7) se observa de igual modo que en la convolución 3, que la suma de los valores da igual a 0, por lo que al obtener píxeles vecinos parecidos (por ejemplo en el agua), se obtienen píxeles blancos.
6. En la quinta convolución (Figura 8), se observa que la suma de los valores son 1, por lo que se tendrá un resultado parecido al de la convolución 2, pero al darle más peso al valor central, se puede remarcar de mejor manera los colores más fuertes.
7. En la sexta convolución (Figura 9), se observa una imagen que no se puede distinguir, esto debido a que la multiplicación de la matriz por $\frac{1}{16}$ genera valores muy pequeños, obteniendo valores cercanos a 0 y por lo tanto únicamente los valores más oscuros se resaltarán.

Por lo que se observó, fue posible realizar las convoluciones con diferentes kernels y apreciar como modificaban la imagen y podían hacer que se resaltaran más cosas, como son los contornos, los colores más claros, etc. Sin embargo, considero que dichas convoluciones tendrán un mejor efecto en imágenes en escalas de grises.

Referencias

- [1] “Cómo usar máscaras en opencv - datasmarts.” <https://www.datasmarts.net/como-usar-mascaras-en-opencv/>. (Accessed on 09/17/2022).
- [2] “(pdf) human-centered content-based image retrieval.” https://www.researchgate.net/publication/228719004_Human-centered_content-based_image_retrieval/figures?lo=1. (Accessed on 09/18/2022).
- [3] “Manual image convolution-zero padding — kaggle.” <https://www.kaggle.com/code/thesherpafromalabama/manual-image-convolution-zero-padding/notebook>. (Accessed on 09/17/2022).
- [4] “2d convolution using python & numpy — by samrat sahoo — analytics vidhya — medium.” <https://medium.com/analytics-vidhya/2d-convolution-using-python-numpy-43442ff5f381>. (Accessed on 09/17/2022).
- [5] “5.2. imágenes rgb — introducción a la programación.” <https://cupi2-ip.github.io/IPBook/nivel4/seccion4-4.html>. (Accessed on 09/17/2022).
- [6] “Image processing in python: Algorithms, tools, and methods you should know - neptune.ai.” <https://neptune.ai/blog/image-processing-python>. (Accessed on 09/17/2022).

Convoluciones

September 18, 2022

```
[10]: import cv2 as cv
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
import numpy as np

img=mpimg.imread('naturaleza.jpg')
print("Imagen original")
plt.imshow(img)
print(img.shape)
```

Imagen original
(775, 1600, 3)



1 Uso de convolución 1

$$K_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

```
[11]: img_or=mpimg.imread('naturaleza.jpg')
k_1=np.array([[0,0,0],[0,1,0],[0,0,0]])
print("convolución 1: ")
print(k_1)
plt.figure(figsize=(15,4))
plt.subplot(1,2,1)
plt.imshow(img_or)
plt.title("Imagen originial")
[x,y,z]=img_or.shape
conv1=np.zeros(img_or.shape)
for ren in range(x-2):
    for columna in range(y-2):
        conv1[ren,columna,0]=sum(sum(k_1*img_or[ren:ren+3,columna:columna+3,0]))
        conv1[ren,columna,1]=sum(sum(k_1*img_or[ren:ren+3,columna:columna+3,1]))
        conv1[ren,columna,2]=sum(sum(k_1*img_or[ren:ren+3,columna:columna+3,2]))

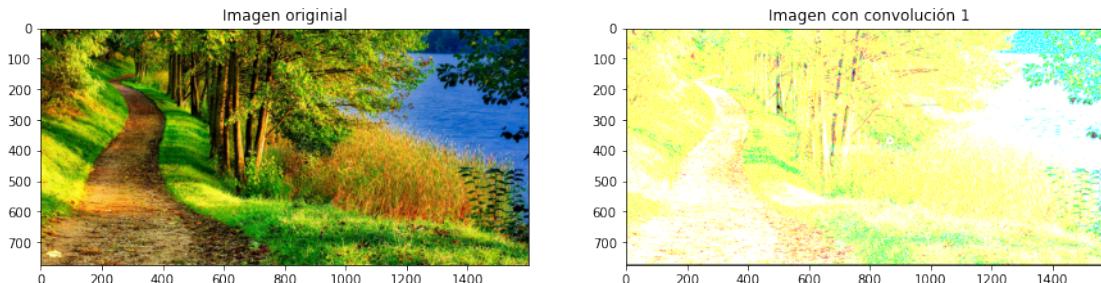
plt.subplot(1,2,2)
plt.imshow(conv1)
plt.title("Imagen con convolución 1")
```

convolución 1:

```
[[0 0 0]
 [0 1 0]
 [0 0 0]]
```

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

```
[11]: Text(0.5, 1.0, 'Imagen con convolución 1')
```



```
[12]: a=np.array([[1,2,3],[4,5,6],[7,8,9]])
b=np.ones((3,3))
print(a*b)
```

```
[[1.  2.  3.]
 [4.  5.  6.]
 [7.  8.  9.]]
```

2 Uso de convolución 2

$$K_2 = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}$$

```
[13]: img_or=mpimg.imread('naturaleza.jpg')
k_2=np.array([[1,0,-1],[0,1,0],[-1,0,1]])
print("convolución 2: ")
print(k_2)
plt.figure(figsize=(15,4))
plt.subplot(1,2,1)
plt.imshow(img_or)
plt.title("Imagen original")
[x,y,z]=img_or.shape
conv2=np.zeros(img_or.shape)
for ren in range(x-2):
    for columna in range(y-2):
        conv2[ren,columna,0]=sum(sum(k_2*img_or[ren:ren+3,columna:columna+3,0]))
        conv2[ren,columna,1]=sum(sum(k_2*img_or[ren:ren+3,columna:columna+3,1]))
        conv2[ren,columna,2]=sum(sum(k_2*img_or[ren:ren+3,columna:columna+3,2]))
```

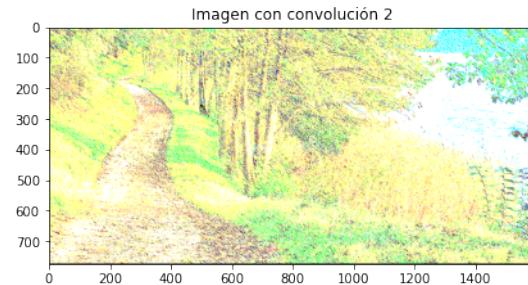
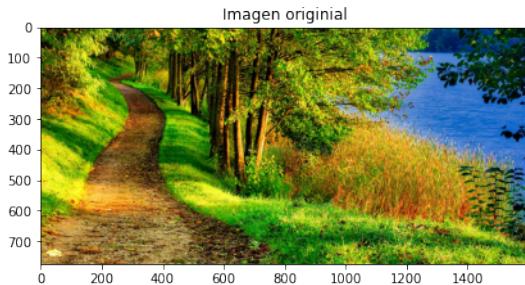
```
plt.subplot(1,2,2)
plt.imshow(conv2)
plt.title("Imagen con convolución 2")
```

convolución 2:

```
[[ 1  0 -1]
 [ 0  1  0]
 [-1  0  1]]
```

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

```
[13]: Text(0.5, 1.0, 'Imagen con convolución 2')
```



3 Uso de convolución 3

$$K_3 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

```
[14]: img_or=mpimg.imread('naturaleza.jpg')
k_3=np.array([[0,-1,0],[-1,4,-1],[0,-1,0]])
print("convolución 3: ")
print(k_3)
plt.figure(figsize=(15,4))
plt.subplot(1,2,1)
plt.imshow(img_or)
plt.title("Imagen original")
[x,y,z]=img_or.shape
conv3=np.zeros(img_or.shape)
for ren in range(x-2):
    for columna in range(y-2):
        conv3[ren,columna,0]=sum(sum(k_3*img_or[ren:ren+3,columna:columna+3,0]))
        conv3[ren,columna,1]=sum(sum(k_3*img_or[ren:ren+3,columna:columna+3,1]))
        conv3[ren,columna,2]=sum(sum(k_3*img_or[ren:ren+3,columna:columna+3,2]))

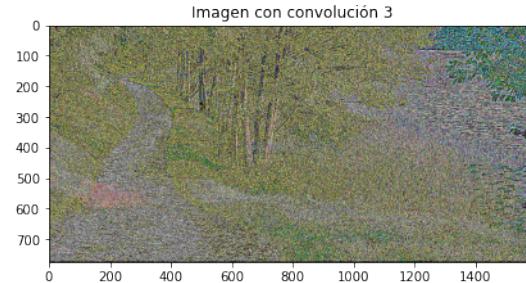
plt.subplot(1,2,2)
plt.imshow(conv3)
plt.title("Imagen con convolución 3")
```

convolución 3:

```
[[ 0 -1  0]
 [-1  4 -1]
 [ 0 -1  0]]
```

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

[14]: Text(0.5, 1.0, 'Imagen con convolución 3')



4 Uso de convolución 4

$$K_4 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

```
[15]: img_or=mpimg.imread('naturaleza.jpg')
k_4=np.array([[-1,-1,-1],[-1,8,-1],[-1,-1,-1]])
print("convolución 4: ")
print(k_4)
plt.figure(figsize=(15,4))
plt.subplot(1,2,1)
plt.imshow(img_or)
plt.title("Imagen original")
[x,y,z]=img_or.shape
conv4=np.zeros(img_or.shape)
for ren in range(x-2):
    for columna in range(y-2):
        conv4[ren,columna,0]=sum(sum(k_4*img_or[ren:ren+3,columna:columna+3,0]))
        conv4[ren,columna,1]=sum(sum(k_4*img_or[ren:ren+3,columna:columna+3,1]))
        conv4[ren,columna,2]=sum(sum(k_4*img_or[ren:ren+3,columna:columna+3,2]))
```

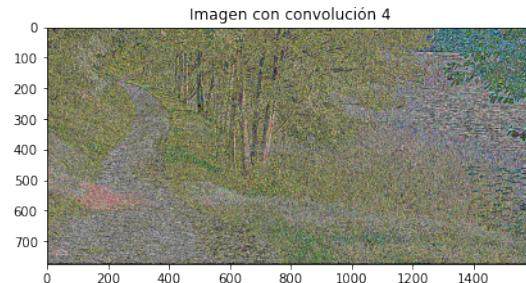
```
plt.subplot(1,2,2)
plt.imshow(conv4)
plt.title("Imagen con convolución 4")
```

convolución 4:

```
[[[-1 -1 -1]
 [-1  8 -1]
 [-1 -1 -1]]]
```

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

```
[15]: Text(0.5, 1.0, 'Imagen con convolución 4')
```



5 Uso de convolución 5

$$K_5 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

```
[16]: img_or=mpimg.imread('naturaleza.jpg')
k_5=np.array([[0,-1,0],[-1,5,-1],[0,-1,0]])
print("convolución 5: ")
print(k_5)
plt.figure(figsize=(15,4))
plt.subplot(1,2,1)
plt.imshow(img_or)
plt.title("Imagen original")
[x,y,z]=img_or.shape
conv5=np.zeros(img_or.shape)
for ren in range(x-2):
    for columna in range(y-2):
        conv5[ren,columna,0]=sum(sum(k_5*img_or[ren:ren+3,columna:columna+3,0]))
        conv5[ren,columna,1]=sum(sum(k_5*img_or[ren:ren+3,columna:columna+3,1]))
        conv5[ren,columna,2]=sum(sum(k_5*img_or[ren:ren+3,columna:columna+3,2]))

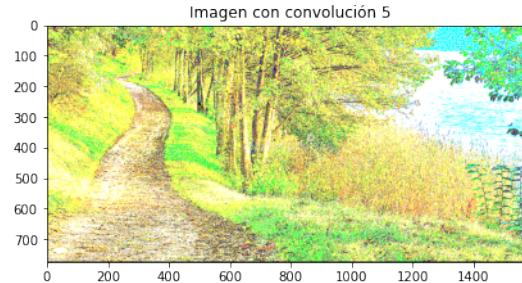
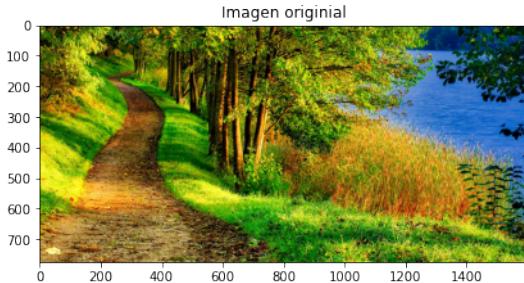
plt.subplot(1,2,2)
plt.imshow(conv5)
plt.title("Imagen con convolución 5")
```

convolución 5:

```
[[ 0 -1  0]
 [-1  5 -1]
 [ 0 -1  0]]
```

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

```
[16]: Text(0.5, 1.0, 'Imagen con convolución 5')
```



6 Uso de convolución 6

$$K_6 = \frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

```
[17]: img_or=mpimg.imread('naturaleza.jpg')
k_6=(1/16)*np.array([[1,2,1],[2,4,2],[1,2,1]])
print("convolución 6: ")
print(k_6)
plt.figure(figsize=(15,4))
plt.subplot(1,2,1)
plt.imshow(img_or)
plt.title("Imagen original")
[x,y,z]=img_or.shape
conv6=np.zeros(img_or.shape)
for ren in range(x-2):
    for columna in range(y-2):
        conv6[ren,columna,0]=sum(sum(k_6*img_or[ren:ren+3,columna:columna+3,0]))
        conv6[ren,columna,1]=sum(sum(k_6*img_or[ren:ren+3,columna:columna+3,1]))
        conv6[ren,columna,2]=sum(sum(k_6*img_or[ren:ren+3,columna:columna+3,2]))
```

```
plt.subplot(1,2,2)
plt.imshow(conv6)
plt.title("Imagen con convolución 6")
```

convolución 6:
[[0.0625 0.125 0.0625]
 [0.125 0.25 0.125]
 [0.0625 0.125 0.0625]]

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

[17]: Text(0.5, 1.0, 'Imagen con convolución 6')

