Explicación de una Operación de Convolución

Concepto Básico

Una convolución es una operación matemática que combina dos matrices para producir una tercera matriz. En el contexto de procesamiento de imágenes, la convolución se usa para aplicar filtros (kernels) a las imágenes para detectar características como bordes, texturas y otros patrones.

Proceso de Convolución

Imagen de Entrada (5x5):

```
1 2 3 0 1
4 5 6 1 2
7 8 9 2 3
1 2 3 0 1
4 5 6 1 2
```

Kernel (3x3):

```
1 0 -1
1 0 -1
1 0 -1
```

Pasos de la Convolución:

1. **Multiplicación y Suma:**
   - El kernel se desliza sobre la imagen de entrada.
   - En cada posición, se multiplica cada valor del kernel por el valor correspondiente de la imagen de entrada.
   - Los productos se suman para obtener un valor de la imagen de salida.
2. **Deslizamiento del Kernel:**
   - El kernel se mueve a la siguiente posición, repitiendo la multiplicación y suma hasta cubrir toda la imagen.

Cálculo de la Imagen de Salida:

Para ilustrar el proceso de cálculo para cada posición:

- **Primera posición (superior izquierda):**

```
Kernel:
1 0 -1
1 0 -1
1 0 -1

Imagen (posición 1):
1 2 3
4 5 6
7 8 9

Multiplicación y Suma:
(1*1 + 0*2 + -1*3) +
(1*4 + 0*5 + -1*6) +
(1*7 + 0*8 + -1*9) = 0
```

- **Segunda posición (desplazamiento a la derecha):**

```
Kernel:
1 0 -1
1 0 -1
1 0 -1

Imagen (posición 2):
2 3 0
5 6 1
8 9 2

Multiplicación y Suma:
(1*2 + 0*3 + -1*0) +
(1*5 + 0*6 + -1*1) +
(1*8 + 0*9 + -1*2) = 2
```

- **Repetición del proceso para toda la imagen:**

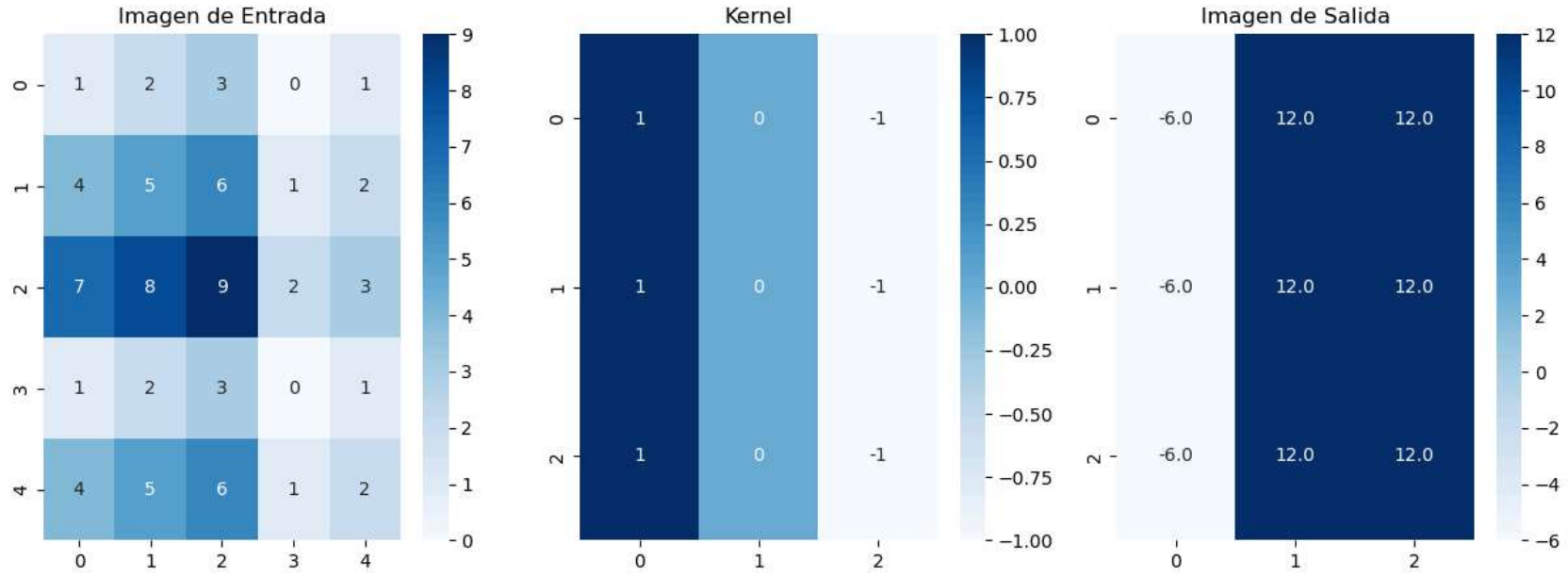
Imagen de Salida (3x3):

```
 0  2  1
-3  6 -1
-3  8  1
```

In [2]:

```python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Definir la imagen de entrada y el kernel
input_image = np.array([
    [1, 2, 3, 0, 1],
    [4, 5, 6, 1, 2],
    [7, 8, 9, 2, 3],
    [1, 2, 3, 0, 1],
    [4, 5, 6, 1, 2]
])

kernel = np.array([
    [1, 0, -1],
    [1, 0, -1],
    [1, 0, -1]
])

# Realizar la convolución manualmente
def convolve2d(image, kernel):
    kernel_height, kernel_width = kernel.shape
    image_height, image_width = image.shape
    output_height = image_height - kernel_height + 1
    output_width = image_width - kernel_width + 1
    output = np.zeros((output_height, output_width))

    for i in range(output_height):
        for j in range(output_width):
            output[i, j] = np.sum(image[i:i+kernel_height, j:j+kernel_width] * kernel)

    return output

output_image = convolve2d(input_image, kernel)

# Visualizar las imágenes
fig, axs = plt.subplots(1, 3, figsize=(15, 5))

sns.heatmap(input_image, annot=True, fmt="d", cmap="Blues", ax=axs[0])
axs[0].set_title('Imagen de Entrada')

sns.heatmap(kernel, annot=True, fmt="d", cmap="Blues", ax=axs[1])
axs[1].set_title('Kernel')
```

```
sns.heatmap(output_image, annot=True, fmt=".1f", cmap="Blues", ax=axs[2])
axs[2].set_title('Imagen de Salida')

plt.show()
```



In [ ]: