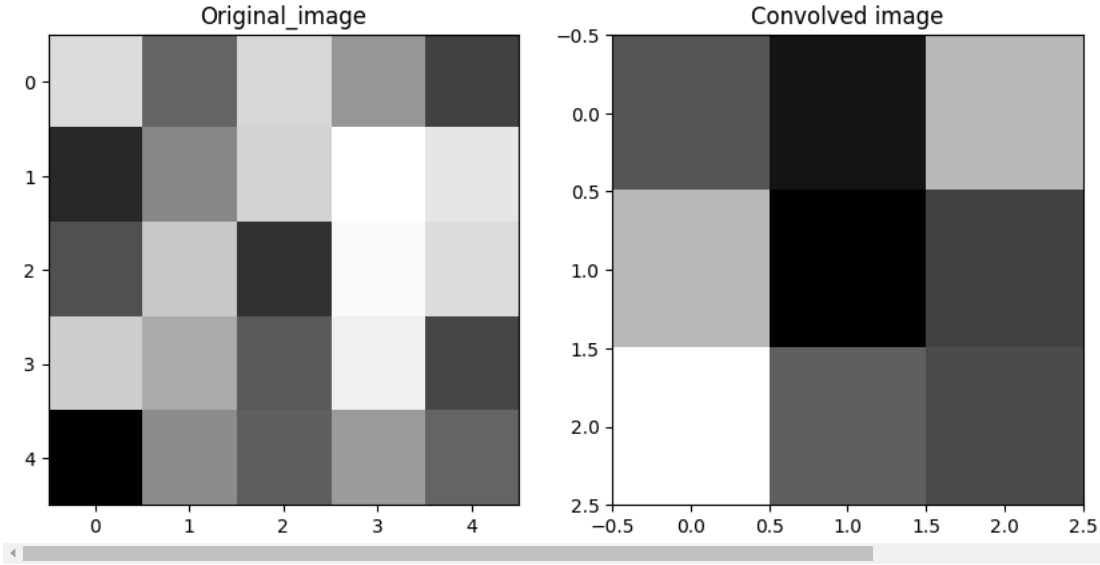
```
from google.colab import drive

# Mount google drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

# Functions to perform Convolution

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):
      patch = image[i:i+kernel_height, j:j+kernel_width]
      output[i,j] = np.sum(patch * kernel)

  return output


image = np.random.rand(5,5)

# Define a kernel (eg Edge detection)
kernel = np.array([[1,0,-1],
                   [1,0,-1],
                  [1,0,-1]])


# Apply Convolution
convolved_image = convolve2d(image, kernel)

# Plot the results
fig, axes = plt.subplots(1,2, figsize=(10,15))
axes[0].imshow(image,cmap='gray')
axes[0].set_title('Original_image')
axes[1].imshow(convolved_image,cmap='gray')
axes[1].set_title('Convolved image')
```

```
Text(0.5, 1.0, 'Convolved image')
```

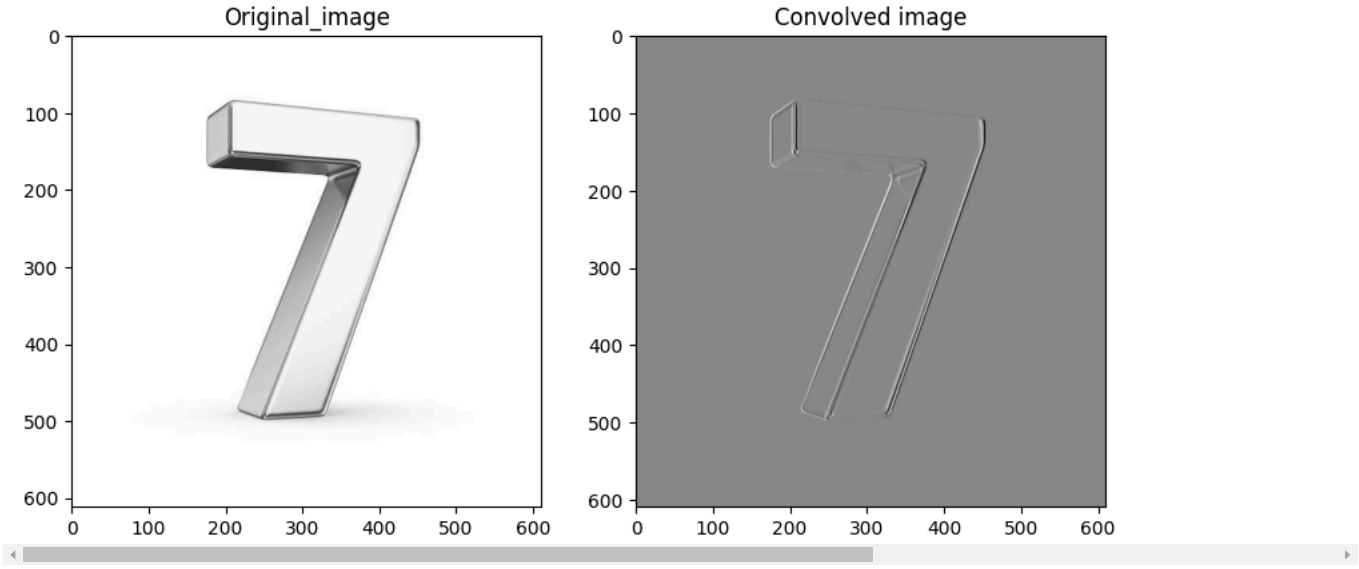



```
image
```

```
array([[0.86040782, 0.40778372, 0.85473932, 0.59227334, 0.2634535 ],
       [0.17435622, 0.53995533, 0.83625164, 0.99475628, 0.90760638],
       [0.32658351, 0.79237151, 0.219768  , 0.97544105, 0.86130839],
       [0.81297469, 0.67715782, 0.37490683, 0.94553724, 0.28748487],
       [0.01101059, 0.56265918, 0.39233962, 0.60739827, 0.41181093]])
```

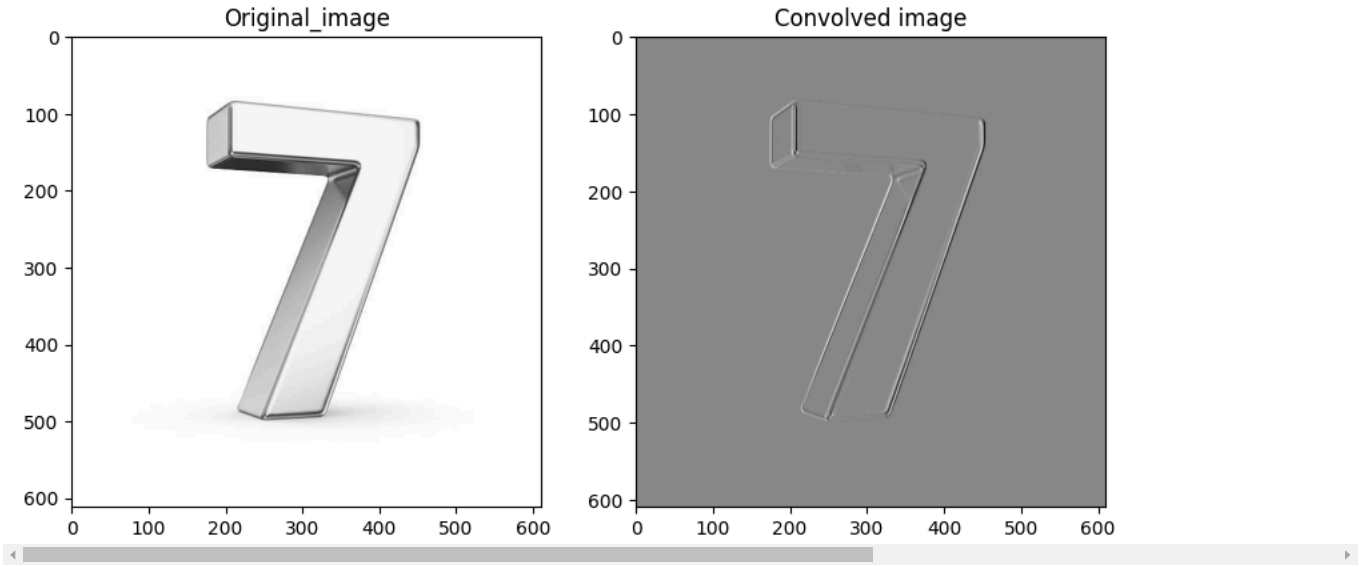
```
image_path = '/content/drive/MyDrive/40_datascience_project/Day2 Dog Breed Prediction//.jpg'
image = Image.open(image_path).convert('L') #Convert to grayscale
image = np.array(image)

# image = np.random.rand(5,5)

# Define a kernel (eg Edge detection)
kernel = np.array([[1,0,-1],
                   [1,0,-1],
                  [1,0,-1]])


# Apply Convolution
convolved_image = convolve2d(image, kernel)

# Plot the results
fig, axes = plt.subplots(1,2, figsize=(10,15))
axes[0].imshow(image,cmap='gray')
axes[0].set_title('Original_image')
axes[1].imshow(convolved_image,cmap='gray')
axes[1].set_title('Convolved image')
```

```
Text(0.5, 1.0, 'Convolved image')
```




```
# image_path = '/content/drive/MyDrive/40_datascience_project/Day2 Dog Breed Prediction/biswash pp.jpg'
# image_path = '/content/drive/MyDrive/40_datascience_project/Day2 Dog Breed Prediction/20230121_190547.jpg'
image = Image.open(image_path).convert('L') #Convert to grayscale
image = np.array(image)

# image = np.random.rand(5,5)

# Define a kernel (eg Edge detection)
kernel = np.array([[1,0,-1],
                   [1,0,-1],
                  [1,0,-1]])


# Apply Convolution
convolved_image = convolve2d(image, kernel)

# Plot the results
fig, axes = plt.subplots(1,2, figsize=(10,15))
axes[0].imshow(image,cmap='gray')
axes[0].set_title('Original_image')
axes[1].imshow(convolved_image,cmap='gray')
axes[1].set_title('Convolved image')
```

```
Text(0.5, 1.0, 'Convolved image')
```



Pooling

```
# Functions to perform Pooling

def max_pooling(image,pool_size):
  pool_height, pool_width = pool_size
  image_height, image_width = image.shape

  output_height = image_height // pool_height
  output_width = image_width // pool_width

  output = np.zeros((output_height, output_width))

  for i in range(output_height):
    for j in range(output_width):
      patch = image[i*pool_height:(i+1)*pool_height, j*pool_width:(j+1)* pool_width]
      output[i,j] = np.max(patch)

  return output


# Apply Pooling

pool_size = (2,2)

pooled_image = max_pooling(image, pool_size)

# Plot the results
fig, axes = plt.subplots(1,2, figsize=(10,15))
axes[0].imshow(image,cmap='gray')
axes[0].set_title('Original_image')
axes[1].imshow(pooled_image,cmap='gray')
axes[1].set_title('Pooled Image')
```

```
Text(0.5, 1.0, 'Pooled Image')
```