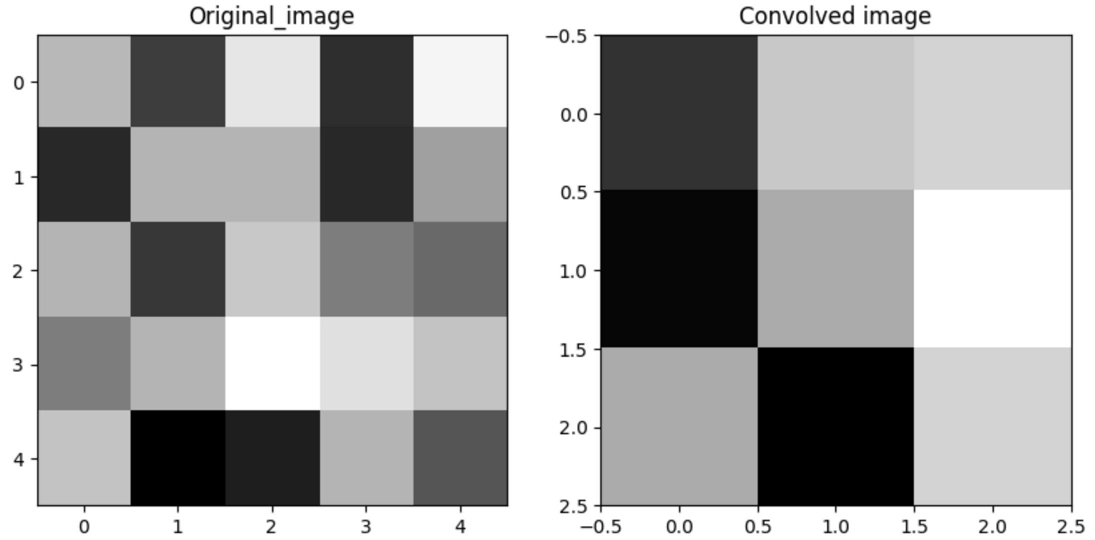
```
from google.colab import drive

# 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')
```

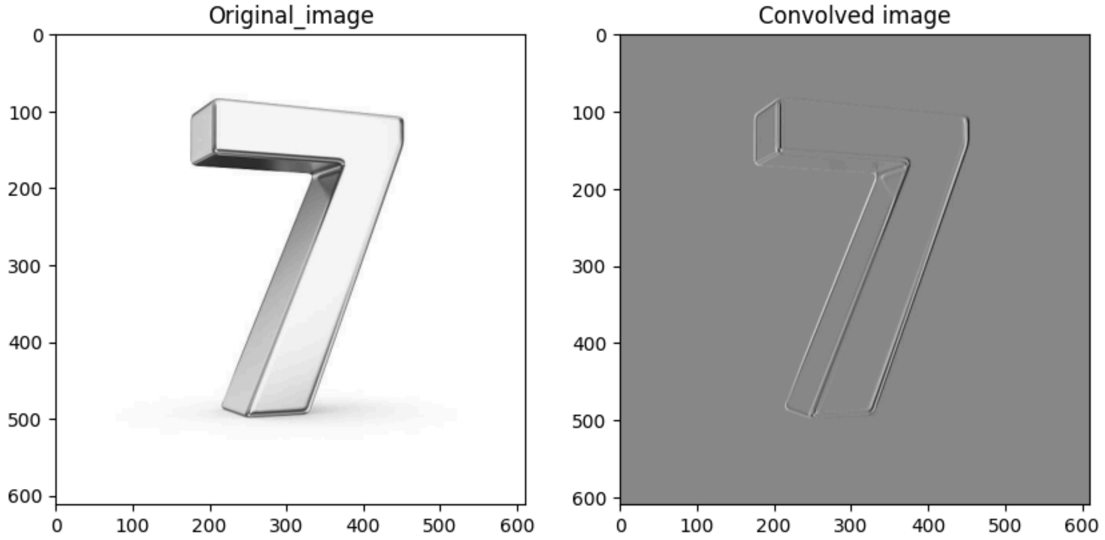


```
image
```

```
array([[0.70521784, 0.28787034, 0.85925434, 0.23325995, 0.91694421],
       [0.21172298, 0.69212285, 0.69748753, 0.2221177 , 0.62075643],
       [0.70088632, 0.27096899, 0.75730268, 0.51034842, 0.42908276],
       [0.50983768, 0.69347819, 0.94717776, 0.8475212 , 0.74122425],
       [0.74188209, 0.06645958, 0.17774638, 0.69234467, 0.36806151]])
```

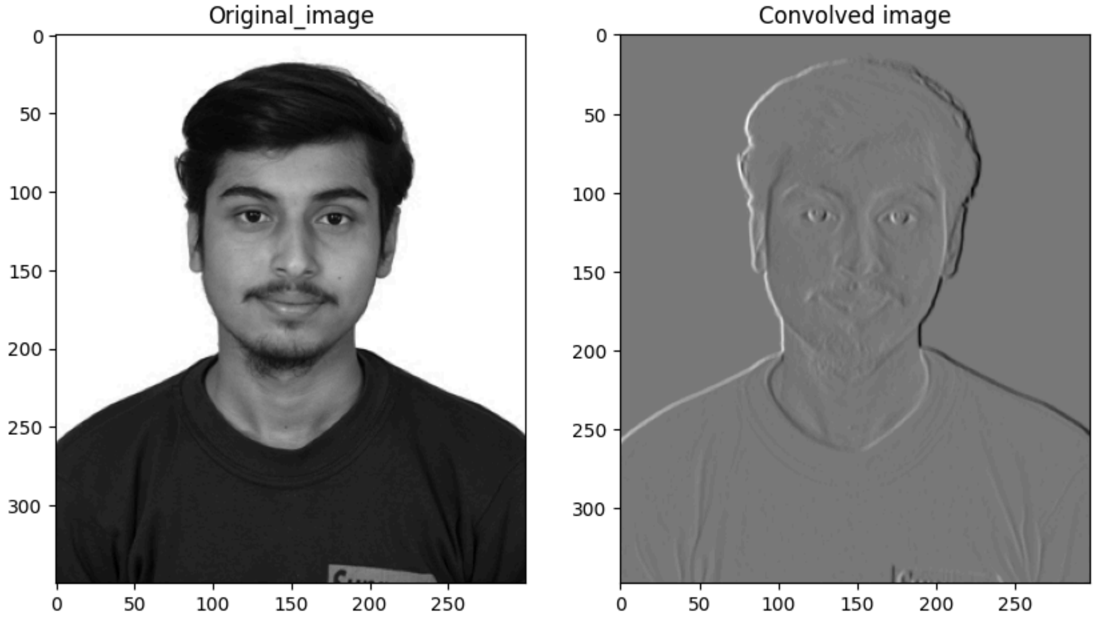
```
image_path = '/content/drive/MyDrive/40_datascience_project/Day2 Dog Breed Prediction/7.jpg'
image = Image.open(image_path).convert('L') #Convert to grayscale
image = np.array(image)

# 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 = 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')
```



Convolution using Pytorch

Generated code may be subject to a license | AIResearchHub/rlgallery

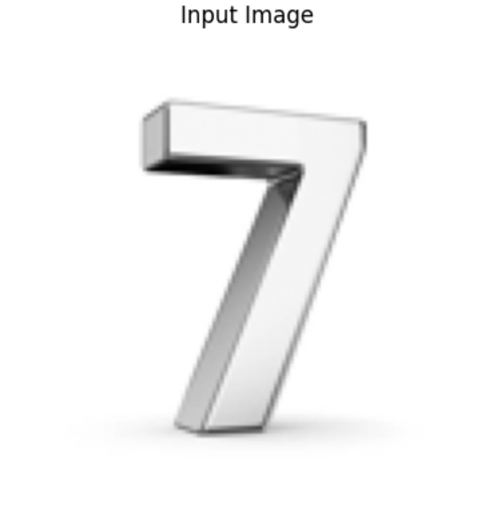
```
import torch
import torch.nn as nn
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt


# Load and preprocess the image
def load_image(image_path, image_size=(128, 128)):
    image = Image.open(image_path).convert('L')  # Convert to grayscale
    image = image.resize(image_size)             # Resize the image
    image = np.array(image)                      # Convert to numpy array
    image = torch.tensor(image, dtype=torch.float32)  # Convert to torch tensor
    image = image.unsqueeze(0).unsqueeze(0)      # Add channel and batch dimensions
    return image

# Path to the image
image_path = r"/content/drive/MyDrive/40_datascience_project/Day2 Dog Breed Prediction/7.jpg"

# Load the image
input_image = load_image(image_path)

# Print the shape of the image tensor
print("Input Image Shape:", input_image.shape)
```

```
Input Image Shape: torch.Size([1, 1, 128, 128])
```

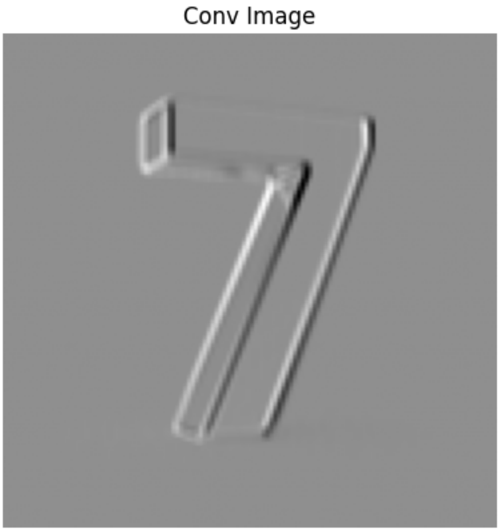
```
# Define a convolution layer

conv_layer = nn.Conv2d(in_channels=1,out_channels=1, kernel_size=3, stride=1, padding=0)

# Set a specific filter to convolutional layer
conv_layer.weight = nn.Parameter(torch.tensor([[[[1,0,-1],
                                                [1,0,-1],
                                                [1,0,-1]]]], dtype=torch.float32))

# Perform convolution
conv_output = conv_layer(input_image)

# Print the shape of the output tensor
print("Output Image Shape:", conv_output.shape)

# Display the input image
plt.imshow(input_image[0, 0].numpy(), cmap='gray')
plt.title("Input Image")
plt.axis('off')
plt.show()
print("Input Image Shape:", input_image.shape)
```

```
# Display the convolved image
plt.imshow(conv_output[0, 0].detach().numpy(), cmap='gray')
plt.title("Conv Image")
plt.axis('off')
plt.show()
print("Conv Image Shape:", conv_output.shape)
```

```
Output Image Shape: torch.Size([1, 1, 126, 126])
```



```
Input Image Shape: torch.Size([1, 1, 128, 128])
```



```
Conv Image Shape: torch.Size([1, 1, 126, 126])
```

Start coding or generate with AI.