KESHAV MEMORIAL INSTITUTE OF TECHNOLOGY
AN AUTONOMOUS INSTITUTION - ACCREDITED BY NAAC WITH 'A' GRADE
Narayanaguda, Hyderabad.

Deep Learning

CNN

EXERCISE-1

18-11-2024

BY
ASHA

EXERCISE 2: 3* 3 convolution on 5* 5 image with and without padding

1.Import necessary libraries

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

2.Load the cat image and perform image Preprocessing

```
# Load the cat image and convert it to grayscale
image_path = 'img.jpeg'  # replace with the path to your cat image
new_image = Image.open(image_path).convert('L')  # Convert to grayscale
# Convert the image to a numpy array
image_array = np.array(new_image)
```

3.Display the original image

```
# Display the original image
plt.imshow(image_array, cmap='gray')
plt.title("Original Image")
plt.axis('off')
plt.show()
```

4.Define the 3x3 filter (kernel) matrix

```
]: # Define the 3x3 filter (kernel) matrix
filter_kernel = np.array([
    [1, 0, -1],
    [1, 0, -1],
    [1, 0, -1]
])
```

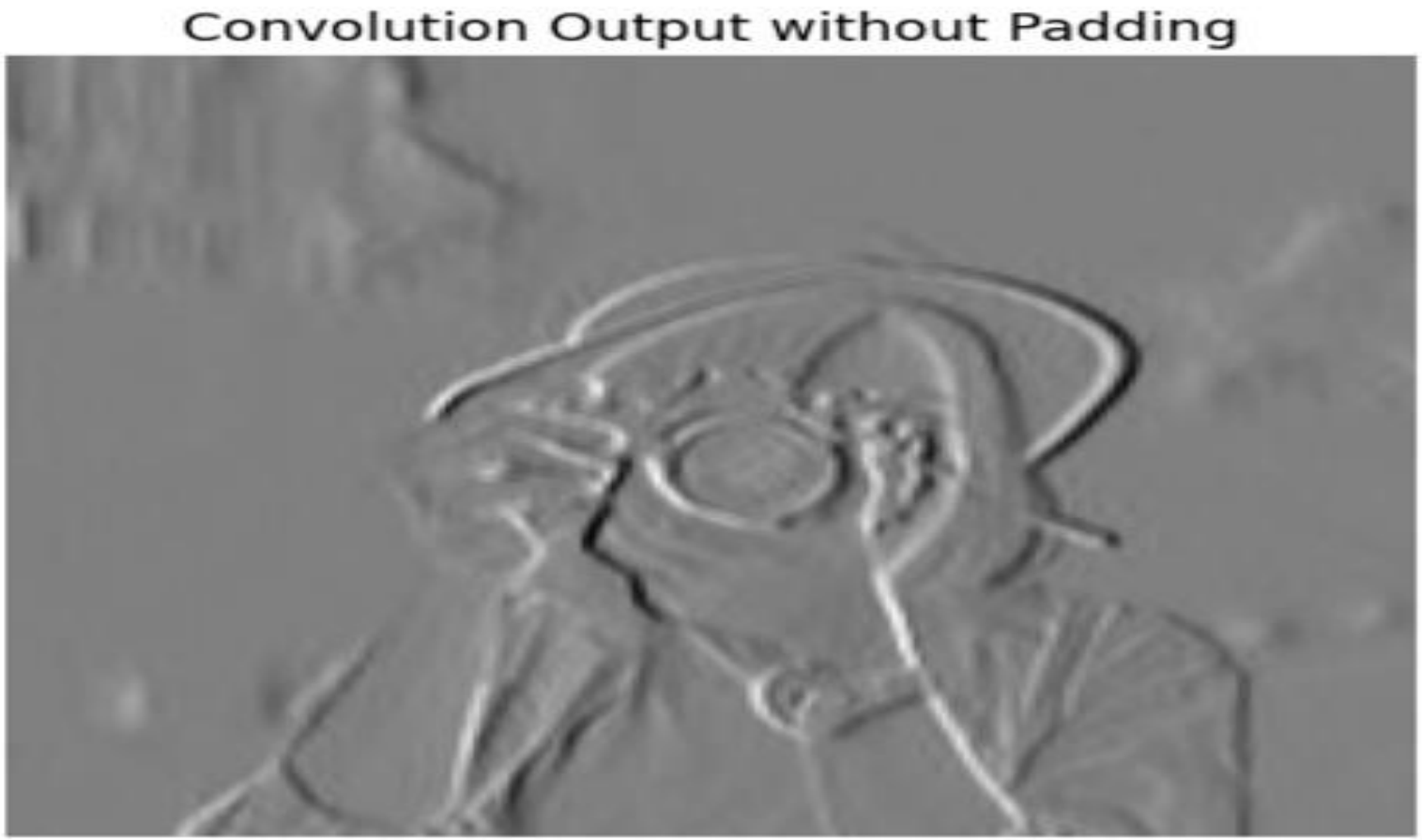
Kmit

5.Define Convolution Function

```python
# Convolution function
def convolve(image, kernel, padding=0):
    # Add padding if needed
    if padding > 0:
        image = np.pad(image, ((padding, padding), (padding, padding)), mode='constant')

    # Dimensions of the image and kernel
    image_height, image_width = image.shape
    kernel_height, kernel_width = kernel.shape

    # Calculate output dimensions
    output_height = image_height - kernel_height + 1
    output_width = image_width - kernel_width + 1

    # Initialize the output matrix
    output = np.zeros((output_height, output_width))

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

    return output
```

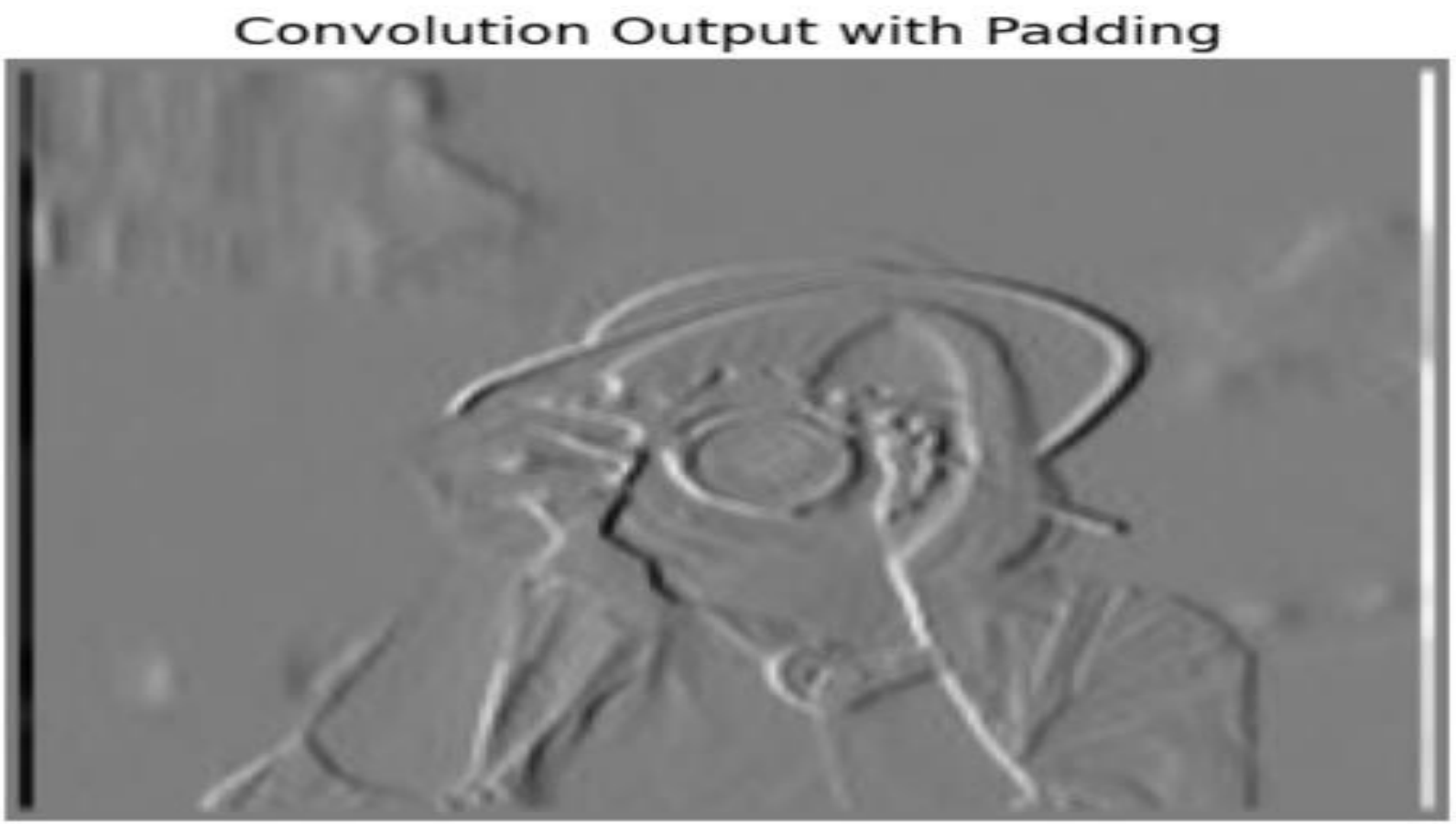
6.Convolution Operation Without padding

```
# Convolve without padding
output_without_padding = convolve(image_array, filter_kernel, padding=0)
plt.imshow(output_without_padding, cmap='gray')
plt.title("Convolution Output without Padding")
plt.axis('off')
plt.show()
```



7.Convolution Operation With padding of 4

```
# Convolve with padding of 1
output_with_padding = convolve(image_array, filter_kernel, padding=4)
plt.imshow(output_with_padding, cmap='gray')
plt.title("Convolution Output with Padding")
plt.axis('off')
plt.show()
```



EXERCISE 2: MAX AND AVERAGE POOLING ON THE FEATURE MAP

Kmit

POOLING

1.Import Necessary Libraries

```
!pip install opencv-python-headless
```

```
Collecting opencv-python-headless
  Using cached opencv_python_headless-4.10.0.84-cp37-abi3-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (49.9 MB)
Requirement already satisfied: numpy>=1.17.0 in /opt/conda/lib/python3.10/site-packages (from opencv-python-headless) (1.26.3)
Installing collected packages: opencv-python-headless
Successfully installed opencv-python-headless-4.10.0.84
```

```
import numpy as np
import cv2
import matplotlib.pyplot as plt
```

2.Take image as input and Preprocess it

```
# Save or provide a path to a small test image for demonstration
test_image_path = "cat_image.png"
```

```
# Load the image and convert it to grayscale
def process_image(image_path):
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    if image is None:
        raise ValueError("Invalid image path or unable to load image.")
    return image
```

3.Implement Max Pooling

```
]: def max_pooling(matrix, pool_size=(2, 2), stride=2):
    pooled_matrix = []
    for i in range(0, matrix.shape[0], stride):
        row = []
        for j in range(0, matrix.shape[1], stride):
            row.append(np.max(matrix[i:i+pool_size[0], j:j+pool_size[1]]))
        pooled_matrix.append(row)
    return np.array(pooled_matrix)
```

4.Implement Average Pooling

```
def average_pooling(matrix, pool_size=(2, 2), stride=2):
    pooled_matrix = []
    for i in range(0, matrix.shape[0], stride):
        row = []
        for j in range(0, matrix.shape[1], stride):
            row.append(np.mean(matrix[i:i+pool_size[0], j:j+pool_size[1]]))
        pooled_matrix.append(row)
    return np.array(pooled_matrix)
```

5.Define Function to display Original Image and Pooled images

```
def plot_results(original, max_pooled, avg_pooled):

    # Display the original and pooled images
    plt.figure(figsize=(12, 4))

    plt.subplot(1, 3, 1)
    plt.title("Original Image")
    plt.imshow(original, cmap='gray')
    plt.axis('off')

    plt.subplot(1, 3, 2)
    plt.title("Max Pooled Image")
    plt.imshow(max_pooled, cmap='gray')
    plt.axis('off')

    plt.subplot(1, 3, 3)
    plt.title("Average Pooled Image")
    plt.imshow(avg_pooled, cmap='gray')
    plt.axis('off')

    plt.tight_layout()
    plt.show()
```

5. Main function to perform pooling on an image

```
# Main function to perform pooling on an image
def apply_pooling(image_path, pool_size=(2, 2), stride=4):
    image = process_image(image_path)
    max_pooled = max_pooling(image, pool_size, stride)
    avg_pooled = average_pooling(image, pool_size, stride)
    plot_results(image, max_pooled, avg_pooled)
```

```
test_image_path = "cat_image.png"
apply_pooling(test_image_path)
```