**Pattern Recognition**

**FA21-BCS-066**

**HUSSAIN ALI**

**BCS-7A**

**Code**

import numpy as np

# Define the convolution operation (without flipping the kernel)

def convolve2d(image, kernel, stride=1, padding=0, verbose=False):

# Add padding to the input image if needed

if padding > 0:

image = np.pad(image, [(padding, padding), (padding, padding)], mode='constant', constant\_values=0)

kernel\_height, kernel\_width = kernel.shape

image\_height, image\_width = image.shape

# Calculate output dimensions

output\_height = (image\_height - kernel\_height) // stride + 1

output\_width = (image\_width - kernel\_width) // stride + 1

# Create an empty output matrix for results

output = np.zeros((output height, output\_width))

# Apply convolution

for i in range(0, output\_height):

for j in range(0, output\_width):

# Extract the region of interest from the image

region = image[i\*stride:i\*stride + kernel\_height, j\*stride:j\*stride + kernel\_width]

# Element-wise multiplication and sum

output[i, j] = np.sum(region \* kernel)

# Print the step-by-step process if verbose mode is on

if verbose:

print(f"Region (i={i}, j={j}):")

print(region)

print(f"Kernel \* Region:")

print(region \* kernel)

print(f"Sum: {output[i, j]}\n")

return output

# Your specified input image

input\_image = np.array([

[2, 2, 2, 3],

[2, 1, 3, 3],

[2, 1, 3, 2],

[1, 3, 2, 2]

])

# Your specified convolution kernel (non-flipped)

kernel = np.array([

[1, -1, -1],

[1, 2, -1],

[1, 1, 1]

])

# Display input image and kernel

print("Input Image:")

print(input\_image)

print("\nConvolution Kernel (non-flipped):")

print(kernel)

# Perform the convolution with stride=1, padding=0, and verbose=True for step-by-step output

convolution\_result = convolve2d(input\_image, kernel, stride=1, padding=0, verbose=True)

# Display the convolution result

print("\nFinal Convolution Result:")

print(convolution\_result)