Machine Learning
LABORATORY: CNN In Class

NAME: 張子中	**STUDENT ID#:** 313605013

Objectives:

- Students will implement 2D convolution and pooling operations from scratch (no PyTorch/TensorFlow high-level API like nn.Conv2d, F.max_pool2d, etc.) using only NumPy.
- Understand padding and stride behavior. Apply vertical and horizontal edge detection. Visualize results in black & white

Part 1. Instruction

- In this assignment, you will implement a basic Convolutional Neural Network operation pipeline using NumPy only — without using any deep learning libraries like PyTorch, TensorFlow, or OpenCV's built-in convolution functions.
- You will manually implement a general 2D convolution function that supports:
  - Padding (to preserve spatial dimensions)
  - Stride (to downsample the output)
- Then, you will apply vertical and horizontal edge detection filters to a grayscale input image and visualize the effects of:
  - Padding (padding=1)
  - Strided convolution (stride=2)
- Specifically, your tasks are to:
  - Load and normalize a grayscale image (e.g., checkerboard.png) for testing edge detection.
  - Implement the general convolution operation

$$C(j,k) = \sum_{l} \sum_{m} I(j+l,k+m)K(l,m)$$

$$\boldsymbol{I(j+l,k+m) = Image\ region}$$

$$\boldsymbol{K(l,m) = Kernel\ (Filter)}$$

$$\boldsymbol{C(j,k) = output\ at\ position\ (j,k)}$$

  - Apply the vertical edge detection kernel and horizontal edge detection kernel from Slide 6 to detect pattern structures in the image.
  - Apply convolution again using stride=2 to observe how spatial resolution changes (see Slide 8).
  - Visualize the output as black-and-white (binary) images using thresholding.
- At the end of the lab, please answer the two short questions to demonstrate your understanding of padding and stride.

Part 2. Code Template

Step | Procedure

1

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

# Step 1: Load a grayscale image and normalize
# ➤ Slide 5: Understanding image representation
image = cv2.imread('original.png', cv2.IMREAD_GRAYSCALE)
image = image.astype(np.float32) / 255.0  # Normalize to range [0, 1]
```

2

```
# Step 2: General Convolution Function
# ➤ Slide 8: C(j, k) = sum_l sum_m I(j + l, k + m) * K(l, m)
def convolve2d(image, kernel, padding=0, stride=1):
    # TODO 1: Flip kernel for convolution
    # TODO 2: Apply zero-padding if padding > 0
    # TODO 3: Calculate output height and width
    # TODO 4: Slide the kernel across the image with stride
    # TODO 5: At each position, compute the sum of element-wise
multiplication
    return np.zeros((1, 1))  # Placeholder, replace with real output
```

3

```
# Step 3: Define edge detection filters
# ➤ Slide 6: Vertical & Horizontal edge filters
vertical_filter = np.array([
    [x, x, x],
    [x, x, x],
    [x, x, x]
], dtype=np.float32)

horizontal_filter = np.array([
    [x, x, x],
    [ x,  x,  x],
    [ x,  x,  x]
], dtype=np.float32)
```

4

```
# Step 4: Convolve image with filters (padding=1, stride=1)
# ➤ Slide 7: Padding helps preserve image size
# TODO: vertical_edges = convolve2d(image, vertical_filter, padding=1)
#  TODO:  horizontal_edges  =  convolve2d(image,  horizontal_filter,
padding=1)

# Try strided convolutions (padding=1, stride=2)
# ➤ Slide 8: Stride reduces spatial resolution
#   TODO:   vertical_stride   =   convolve2d(image,   vertical_filter,
padding=1, stride=2)
#  TODO:  horizontal_stride  =  convolve2d(image,  horizontal_filter,
padding=1, stride=2)
```

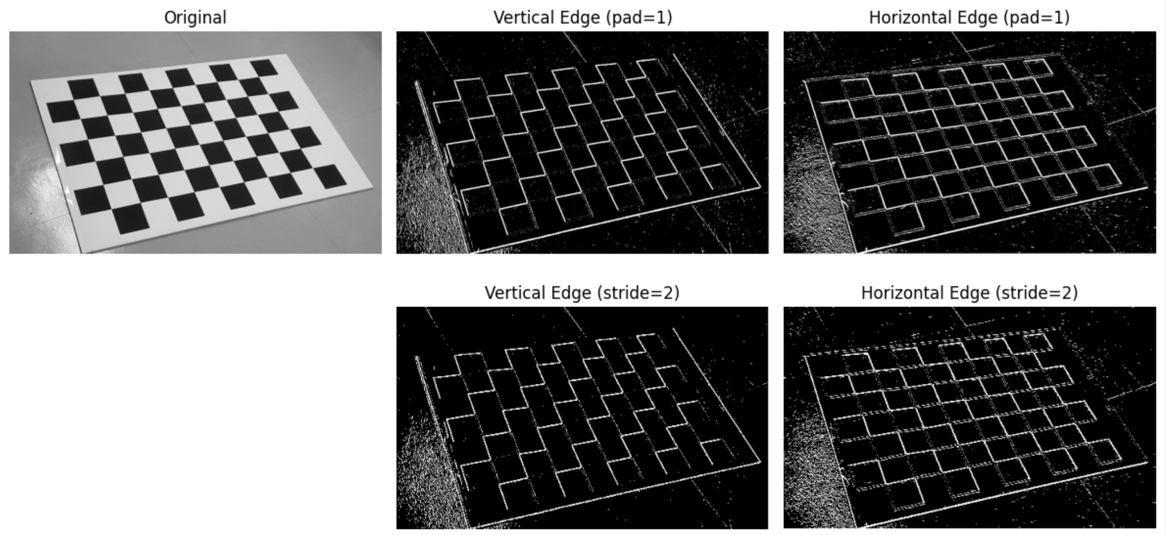
5

```
# Step 5: Visualization and Binarization function for black-and-white
display
def binarize(img, threshold=0.5):
    img = img - np.min(img)
    if np.max(img) != 0:
        img = img / np.max(img)
    return (img > threshold).astype(np.float32)
```

```
# Visualization
# TODO: Use matplotlib to show:
# - Original image
# - Vertical edges (pad=1)
# - Horizontal edges (pad=1)
# - Vertical edges (stride=2)
# - Horizontal edges (stride=2)
```

5

```
#Example Output: (References Only)
```



Grading Assignment & Submission (30% Max)

Implementation:

1. (10%) Correctly implement the convolve2d() function, including kernel flipping, padding, and stride
2. (5%) Correctly apply vertical and horizontal edge detection filters.
3. (5%) Apply binary thresholding to convert outputs to black-and-white images, and clearly visualize them using matplotlib. Show all five views: original, vertical (pad=1), horizontal (pad=1), vertical (stride=2), and horizontal (stride=2).

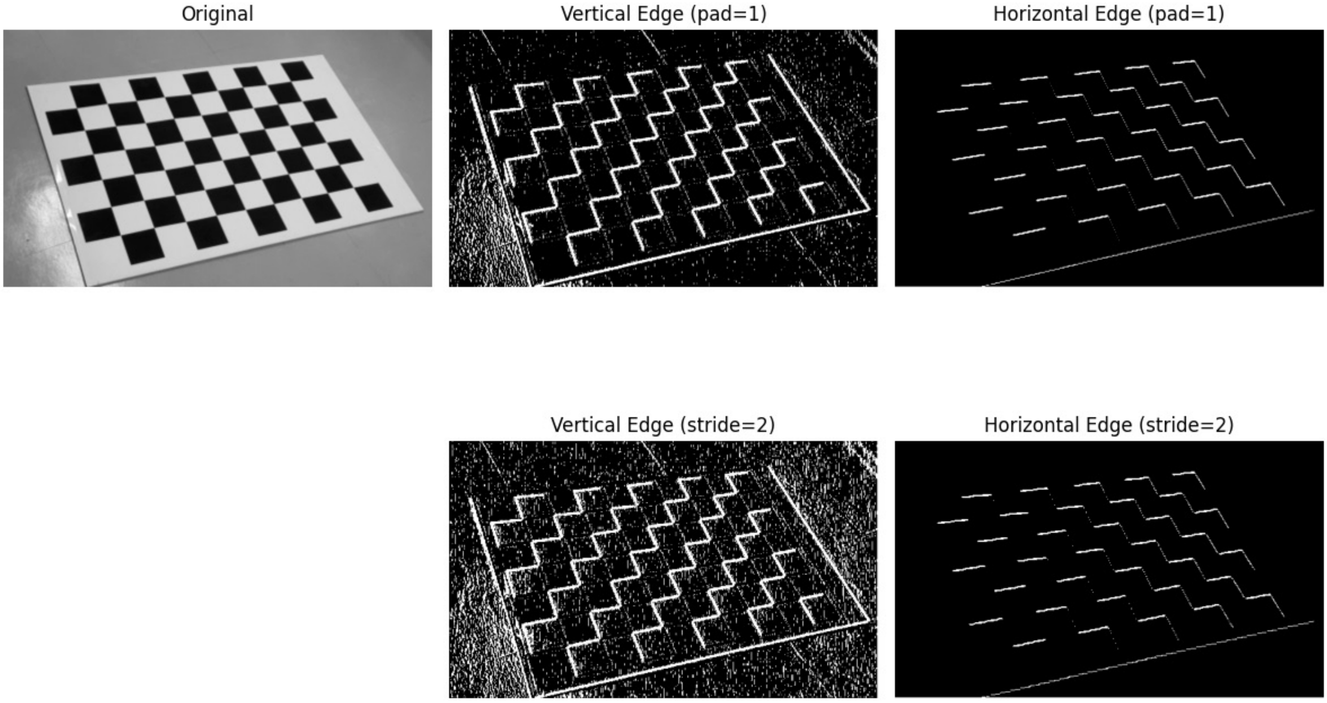
Question:

4. (5%) What types of patterns are detected by the vertical edge filter in an image? How is this different from the horizontal edge filter?
5. (5%) What is the effect of padding on the output image when applying convolution? Why is padding used?

Submission :

1. Report: Provide your screenshots of your results in the last pages of this PDF File.
2. Code: Submit your complete Python script in either .py or .ipynb format.
3. Upload both your report and code to the E3 system (**Labs6 In Class Assignment**). Name your files correctly:
   a. Report: StudentID_Lab6_InClass.pdf
   b. Code: StudentID_Lab6_InClass.py or StudentID_Lab6_InClass.ipynb
4. Deadline: 16:20 PM
5. Plagiarism is **strictly prohibited**. Submitting copied work from other students will result in penalties.

Results and Discussion:



A1:
Vertical filter detects vertical edges.
Horizontal filter detects horizontal edges.

A2:

Padding restores the spatial original image size. Without padding, the convolution operation reduces the dimensions of the original image, leading to smaller output.