2D List - Convolutional calculation

Bài tập ngày 4 - Module 1

Ngày 14 tháng 6 năm 2024

Ngày thực hiện:	14/6/2024
Tác giả:	Đinh Thị Tâm
Nguồn:	AIO 2023
Nguồn dữ liệu (nếu có):	Link of Source code
Từ khóa:	Convolutional calculation
Người tóm tắt:	Đinh Thị Tâm

Nội dung:

- Tích chập (Convolution) là phép toán toán học thực hiện phép nhân từng phần tử giữa hai ma trận (hoặc mảng) và sau đó cộng tổng các kết quả nhân tại các vị trí tương ứng để tạo ra một ma trận mới. Ma trận mới này được gọi là ma trận tích chập.
- Để tính tích chập giữa hai ma trận này, ta thực hiện các bước sau:
  - Di chuyển ma trận bộ lọc: Di chuyển ma trận bộ lọc từng vị trí trên ma trận đầu vào. Tại mỗi vị trí, ta nhân từng phần tử của ma trận bộ lọc với phần tử tương ứng của ma trận đầu vào và cộng tổng các kết quả nhân.
  - Ghi kết quả: Kết quả tính toán tại mỗi vị trí di chuyển của ma trận bộ lọc được ghi vào một phần tử tương ứng trong ma trận tích chập.
  - Lặp lại: Lặp lại bước 1 và 2 cho đến khi ma trận bộ lọc đã di chuyển qua tất cả các vị trí trên ma trận đầu vào.

1. **Bài tập vận dụng:**

   - Cho 2 ma trận sau:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \quad \text{và Kernel} \quad B = \begin{pmatrix} 2 & 4 \\ 1 & 3 \end{pmatrix}$$

   Hãy dùng Python, không sử dụng Numpy thực hiện các yêu cầu sau:

   (a) Hãy tính Convolutional khi áp Kernel B vào A.

   (b) Hãy tính Convolutional khi áp Kernel

$$C = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix} \text{ vào A.}$$

2. Hiện thực thuật toán

```
mat_a = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
kernal = [[2, 4], [1, 3]]


def getConvolutional(a, kernel):
    # Get dimensions of input a and kernel
    rows_a = len(a)
    cols_a = len(a[0])
    rows_k = len(kernel)
    cols_k = len(kernel[0])

    # Define result dimensions
    result_height = rows_a - rows_k + 1
    result_width = cols_a - rows_k + 1

    # Initialize result a with zeros
    result = [[0 for _ in range(result_width)] for _ in range(result_height)]

    # Perform convolution
    for i in range(result_height):
        for j in range(result_width):
            sum = 0
            for ki in range(rows_k):
                for kj in range(cols_k):
                    sum += a[i + ki][j + kj] * kernel[ki][kj]
            result[i][j] = sum

    return result


# cau 1
x = getConvolutional(mat_a, kernal)
print(x)
# cau 2
kernal_c = [[1, 1, 1], [0, 0, 0], [1, 1, 1]]
x = getConvolutional(mat_a, kernal_c)
print(x)
```

3. output chương trình

```
[[29, 39], [59, 69]]
[[30]]
```