AI VIET NAM – COURSE 2024

Lập trình Python căn bản - Day 6

Ngày 16 tháng 6 năm 2024

Ngày thực hiện:	16/06/2024
Người thực hiện:	Đinh Thị Tâm
Nguồn:	AIO2024 - Day 6
Nguồn dữ liệu (nếu có):	Link of Data Sources of Day 6
Từ khóa:	2D List - Convolutional calculation
Người tóm tắt:	Đinh Thị Tâm

Mô tả thuật toán:

- Padding là kỹ thuật thêm các đường viền xung quanh ảnh đầu vào trước khi thực hiện phép tích chập.
- Mục đích chính của padding là:
  - Giữ nguyên kích thước ảnh đầu ra: Sau khi áp dụng phép tích chập, ảnh đầu ra thường có kích thước nhỏ hơn ảnh đầu vào. Padding giúp bù đắp lại phần kích thước bị mất đi này, đảm bảo rằng ảnh đầu ra có kích thước tương đương hoặc gần bằng ảnh đầu vào.
  - Giảm thiểu mất thông tin: Khi thực hiện phép tích chập, các pixel ở rìa ảnh thường bị bỏ qua do kernel không thể bao phủ toàn bộ ảnh. Padding giúp bổ sung thêm các pixel xung quanh ảnh, đảm bảo rằng tất cả các pixel đều được tham gia vào quá trình tính toán và không có thông tin nào bị mất đi.
  - Kiểm soát kích thước của ảnh đầu ra: Padding có thể được sử dụng để điều chỉnh kích thước của ảnh đầu ra. Ví dụ, nếu muốn ảnh đầu ra có kích thước lớn hơn ảnh đầu vào, ta có thể sử dụng padding với kích thước lớn hơn kích thước kernel
- Một trong những phương pháp padding phổ biến nhất là Zero padding là một kỹ thuật được sử dụng phổ biến trong xử lý tín hiệu số (DSP) và mạng nơ-ron tích chập (CNN) để bổ sung thêm các giá trị 0 vào viền ngoài của ảnh trước khi thực hiện các phép toán tiếp theo. Việc này giúp kiểm soát kích thước đầu ra của tín hiệu sau khi xử lý và tránh tình trạng mất thông tin ở rìa dữ liệu. Sau khi thêm các giá trị 0 vào viền ngoài của ảnh chúng ta thực hiện cách tính convolution như đã được học ở bài trước đó để ra kết quả output cuối cùng.

1. **Bài tập:** Cho ma trận $A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ và Kernel $B = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$.

   Câu 1: Hãy tính Convolutional khi áp Kernel B vào A có sử dụng zero padding.

   Câu 2: Hãy tính Convolutional khi áp Kernel $C = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ vào A có sử dụng zero padding.

2. Cài đặt

```
   # get zero padding
def getZeroPadding(a):
    row_a = len(a)
    col_a = len(a[0])
    a_new = [[0]*(row_a+2) for x in range(col_a+2)]
    # input data of a in a_new
    for idr in range(1, row_a+1):
        for idc in range(1, col_a+1):
            a_new[idr][idc] = a[idr-1][idc-1]
    return a_new
# tinh tich chap


def getConvolutional(a, kernel):
    rows_a = len(a)
    cols_a = len(a[0])
    rows_k = len(kernel)
    cols_k = len(kernel[0])
    result_height = rows_a - rows_k + 1
    result_width = cols_a - rows_k + 1
    result = [[0 for _ in range(result_width)] for _ in range(result_height)]
    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


# main
# cau 1:
mat_a = [[0, 0, 0], [0, 4, 0], [0, 1, 0]]
mat_b = [[1, 1], [1, 1]]
mat_azero = getZeroPadding(mat_a)
# print(mat_azero)
mat_convolution = getConvolutional(mat_azero, mat_b)
print(f'Cau 1: {mat_convolution}')
# cau 2:
mat_c = [[0, 1, 0], [0, 1, 0], [0, 1, 0]]
mat_convolution = getConvolutional(mat_azero, mat_c)
print(f'Cau 2: {mat_convolution}')
```

3. Kết quả thực thi Cau 1: [[0, 0, 0, 0], [0, 4, 4, 0], [0, 5, 5, 0], [0, 1, 1, 0]]
   Cau 2: [[0, 4, 0], [0, 5, 0], [0, 5, 0]]