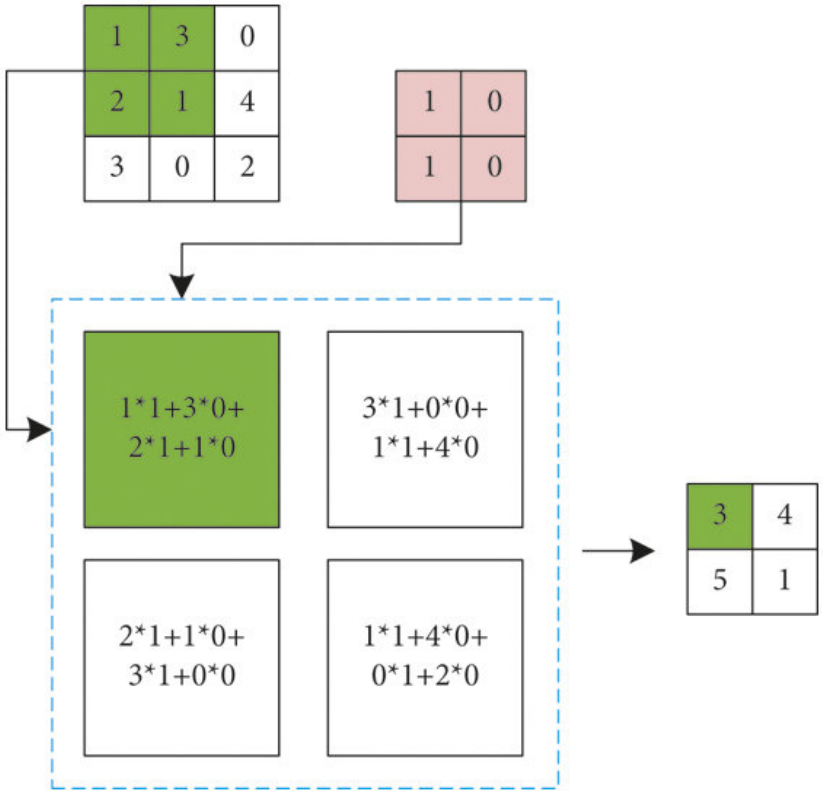
2D List - Convolutional calculation

Hoàng-Nguyên Vũ

1. **Mô tả:**

- **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.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.



2. **Bài tập:** Cho 2 ma trận sau: A = $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$ và Kernal B = $\begin{bmatrix} 2 & 4 \\ 1 & 3 \end{bmatrix}$. Sử dụng Python, không dùng thư viện numpy

Câu 1. Hãy tính Convolutional khi áp Kernal B vào A

Câu 2. Hãy tính Convolutional khi áp Kernal C = $\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$ vào A

```
mat_a = [''' Your Code Here ''']
kernal = [''' Your Code Here ''']
# Your code here
```

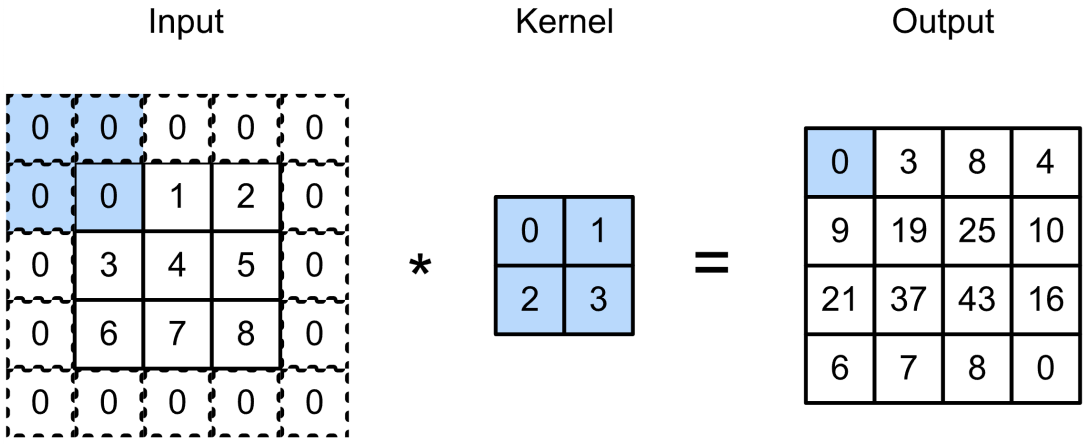
Output:
- Câu 1: [[29, 39], [59, 69]]
- Câu 2: [30]

2D List - Convolutional calculation

Hoàng-Nguyên Vũ

1. **Mô tả:**

- **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.



2. **Bài tập:** Cho 2 ma trận sau: A = $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ và Kernal B = $\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$. Sử dụng Python, không dùng thư viện numpy

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

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

```
mat_a = [''' Your Code Here ''']
kernal = [''' Your Code Here ''']
# Your code here
```

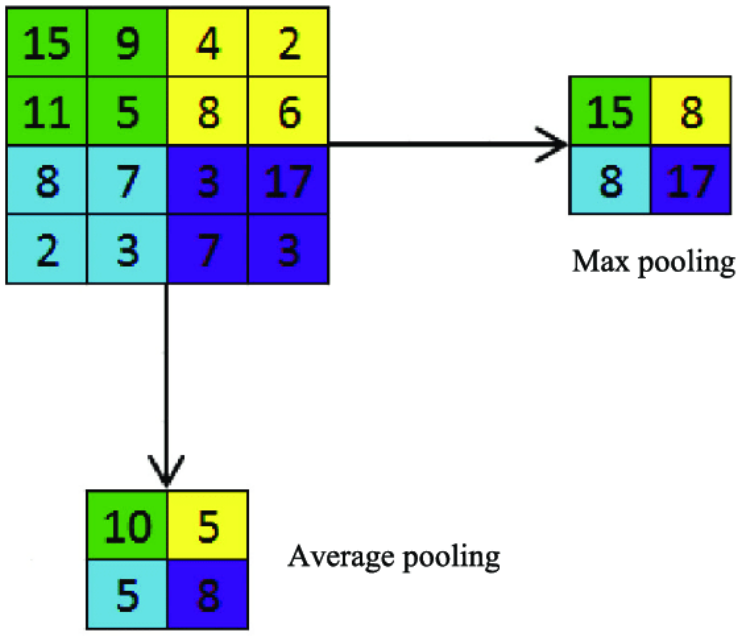
Output:
- **Câu 1:** [[0, 0, 0, 0], [0, 4, 4, 0], [0, 5, 5, 0], [0, 1, 1, 0]]
- **Câu 2:** [[0, 4, 0], [0, 5, 0], [0, 5, 0]]

2D List - Convolutional calculation

Hoàng-Nguyên Vũ

1. **Mô tả:**

- **Max Pooling** và **Average Pooling** là hai kỹ thuật thu gọn phổ biến được sử dụng trong mạng nơ-ron tích chập (CNN) để giảm kích thước dữ liệu và cải thiện hiệu suất tính toán.
  + **Max Pooling:**
    - Lấy giá trị tối đa trong một vùng chọn (window) được xác định trước.
    - Giữ lại các đặc điểm chủ đạo nhất trong vùng chọn.
    - Giảm nhiễu và biến động dữ liệu.
    - Thường được sử dụng trong các nhiệm vụ nhận dạng đối tượng, nơi các đặc điểm chi tiết có thể không quan trọng.
  + **Average Pooling:**
    - Tính trung bình của tất cả các giá trị trong vùng chọn.
    - Giữ lại nhiều thông tin hơn so với Max Pooling.
    - Có thể làm mịn dữ liệu, giúp giảm nhiễu.
    - Thường được sử dụng trong các nhiệm vụ phân đoạn ảnh và phát hiện đối tượng, nơi cần có thông tin chi tiết hơn về cấu trúc hình ảnh.



2. **Bài tập:** Cho 2 ma trận sau: A = $\begin{bmatrix} 0 & 0 & 0 & 4 \\ 0 & 4 & 0 & 2 \\ 0 & 1 & 0 & 2 \\ 0 & 3 & 0 & 2 \end{bmatrix}$. Sử dụng Python, không dùng thư viện numpy

Câu 1. Hãy output của A với bước nhảy là 2 cho cả chiều ngang và dọc và max pooling

Câu 2. Hãy output của A với bước nhảy là 2 cả chiều ngang và dọc và average pooling

```
mat_a = [''' Your Code Here ''']
kernal = [''' Your Code Here ''']
# Your code here
```

Output:
- Câu 1: [[4, 4], [3, 2]]
- Câu 2: [[1, 1.5], [1, 1]]