

BÀI 7. MẠNG NORON TÍCH CHẬP

(CNN – Convolutional Neural Network)

4.1. Giới thiệu

Mạng noron tích chập là một loại mạng noron đã đạt được nhiều thành tựu trong các bài toán có liên quan đến hình ảnh như nhận dạng hình ảnh(image recognition) và phân lớp hình ảnh (image classification) hiện nay. Ngoài ra, mạng CNN còn được ứng dụng trong các bài toán xử lý ngôn ngữ tự nhiên (natural language processing) như phát hiện thư rác, phân loại văn bản...

Các mạng noron truyền thẳng nhiều lớp(multilayer perceptron) chỉ được xây dựng để nhận dữ liệu đầu vào dưới dạng vector. Đối với một số loại dữ liệu, đặc biệt là dữ liệu ở dạng hình ảnh, mạng noron truyền thẳng tỏ ra không hiệu quả. Để áp dụng mạng noron truyền thẳng nhiều lớp cho việc xử lý các dữ liệu ở dạng hình ảnh, chúng ta cần phải chuyển đổi được hình ảnh về dưới dạng vector. Điều này thường gây ra sự mất mát nhiều thông tin trong dữ liệu gốc ban đầu. Mạng CNN được giới thiệu bởi LeCun đã lược bỏ công việc trích xuất các đặc trưng một cách thủ công.

Mạng CNN có kiến trúc được cấu tạo bởi một số loại layer bao gồm:

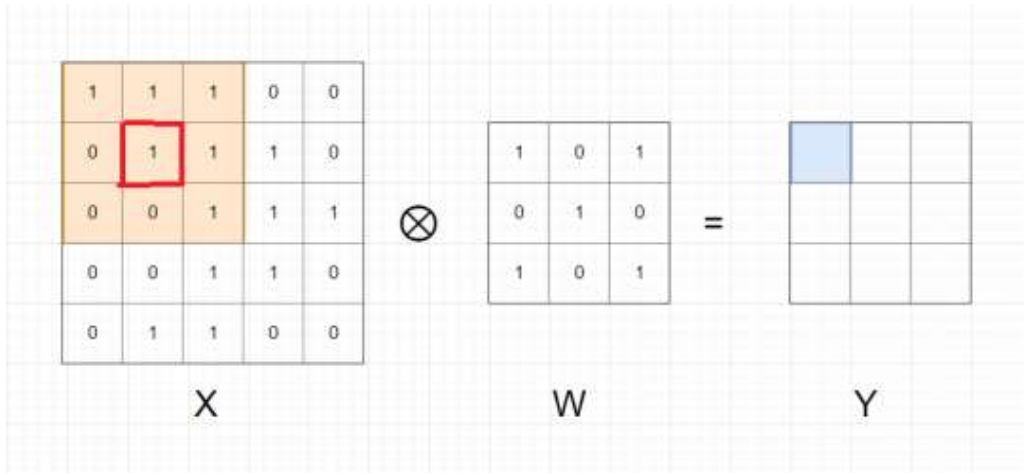
- Convolutional layer
- Pooling layer
- Fully connected layer

4.2. Tích chập

Để cho dễ hình dung mình sẽ lấy ví dụ trên ảnh xám, tức là ảnh được biểu diễn dưới dạng ma trận A kích thước $m \times n$. Ta định nghĩa **kernel** là một ma trận vuông kích thước $k \times k$ trong đó k là số lẻ. k có thể bằng 1, 3, 5, 7, 9,... Ví dụ kernel kích thước 3×3

$$W = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

Kí hiệu phép tính convolution (\otimes), kí hiệu $Y = X \otimes W$. Với mỗi phần tử x_{ij} trong ma trận X lấy ra một ma trận có kích thước bằng kích thước của kernel W có phần tử x_{ij} làm trung tâm (đây là vì sao kích thước của kernel thường lẻ) gọi là ma trận A . Sau đó tính tổng các phần tử của phép tính của ma trận A và ma trận W , rồi viết vào ma trận kết quả Y .

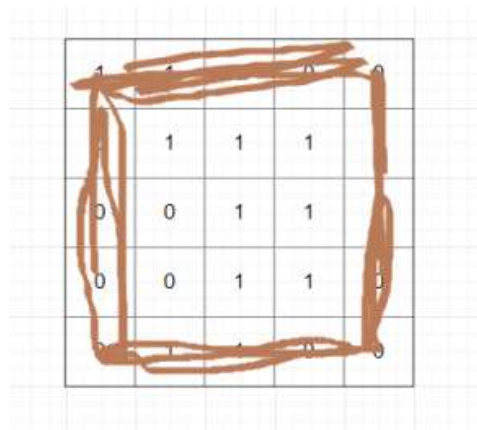


Ví dụ khi tính tại x_{22} (ô khoanh đỏ trong hình), ma trận A cùng kích thước với W , có x_{22} làm trung tâm có màu nền da cam như trong hình. Sau đó tính:

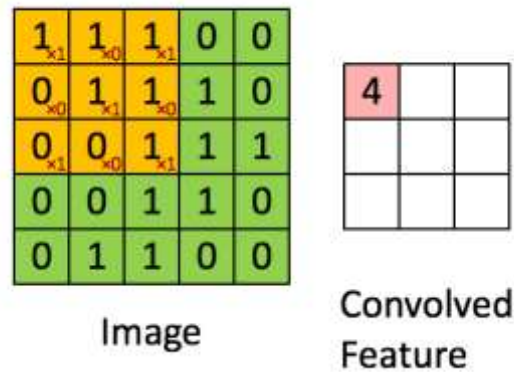
$$y_{11} = \text{sum}(A \otimes W) = x_{11} * w_{11} + x_{12} * w_{12} + x_{13} * w_{13} + x_{21} * w_{21} + x_{22} * w_{22} + x_{23} * w_{23} + x_{31} * w_{31} + x_{32} * w_{32} + x_{33} * w_{33} = 4$$

. Và làm tương tự với các phần tử còn lại trong ma trận.

Thế thì sẽ xử lý thế nào với phần tử ở viền ngoài như x_{11} ? Bình thường khi tính thì sẽ bỏ qua các phần tử ở viền ngoài, vì không tìm được ma trận A ở trong X .



Nên ta để ý thấy ma trận Y có kích thước nhỏ hơn ma trận X. Kích thước của ma trận Y là $(m-k+1) * (n-k+1)$.



Như ở trên thì mỗi lần thực hiện phép tính convolution xong thì kích thước ma trận Y đều nhỏ hơn X. Tuy nhiên, để ma trận Y thu được có kích thước bằng ma trận X, ta thêm giá trị 0 ở viền ngoài ma trận X.

0	0	0	0	0	0	0
0	1	1	1	0	0	0
0	0	1	1	1	0	0
0	0	0	1	1	1	0
0	0	0	1	1	0	0
0	0	1	1	0	0	0
0	0	0	0	0	0	0

Ma trận X khi thêm viền 0 bên ngoài

Rõ ràng là giờ đã giải quyết được vấn đề tìm $A \otimes W$ cho phần tử x_{11} , và ma trận Y thu được sẽ bằng kích thước ma trận X ban đầu. Phép tính này gọi là convolution với **padding=1**. Padding=k nghĩa là thêm k vector 0 vào mỗi phía của ma trận.

Như ở trên ta thực hiện tuần tự các phần tử trong ma trận X, thu được ma trận Y cùng kích thước ma trận X, ta gọi là **stride=1**.

0	0	0	0	0	0	0
0	1	1	1	0	0	0
0	0	1	1	1	0	0
0	0	0	1	1	1	0
0	0	0	1	1	0	0
0	0	1	1	0	0	0
0	0	0	0	0	0	0

Stride=1, Padding=1

Tuy nhiên nếu **stride=k** ($k > 1$) thì ta chỉ thực hiện phép tính convolution trên các phần tử $x_{1+i*k, 1+j*k}$. Ví dụ $k = 2$.

0	0	0	0	0	0	0
0	1	1	1	0	0	0
0	0	1	1	1	0	0
0	0	0	1	1	1	0
0	0	0	1	1	0	0
0	0	1	1	0	0	0
0	0	0	0	0	0	0

Padding=1, Stride=2

Hiểu đơn giản là bắt đầu từ vị trí x_{11} sau đó nhảy k bước theo chiều dọc và ngang cho đến hết ma trận X . Kích thước của ma trận Y là $3*3$ đã giảm đi đáng kể so với ma trận X .







Công thức tổng quát cho phép tính convolution của ma trận X kích thước $m*n$ với kernel kích thước $k*k$, $\text{stride} = s$, $\text{padding} = p$ ra ma trận Y sẽ có kích thước là:

$$\left(\frac{m - k + 2p}{s} + 1\right) * \left(\frac{n - k + 2p}{s} + 1\right).$$

Stride thường dùng để giảm kích thước của ma trận sau phép tính convolution.

Ý nghĩa của phép tính convolution:

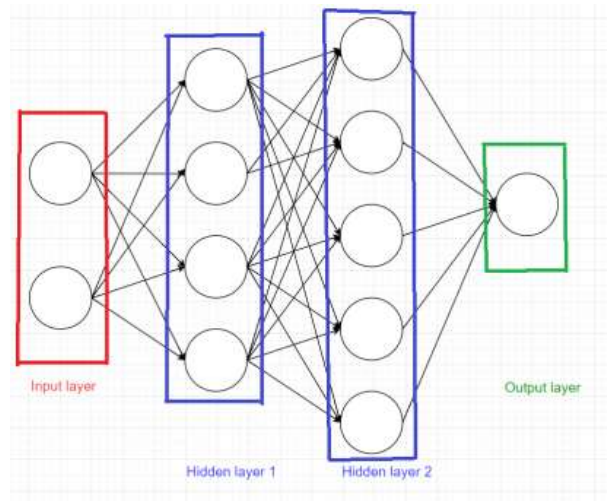
Mục đích của phép tính convolution trên ảnh là làm mờ, làm nét ảnh, xác định biên... Mỗi kernel khác nhau sẽ cho ra phép convolution có ý nghĩa khác nhau. Ví dụ:

Operation	Kernel ω	Image result $g(x,y)$
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

4.3. Mạng CNN

4.3.1. Tầng tích chập

Mô hình [neural network](#) từ những bài trước:

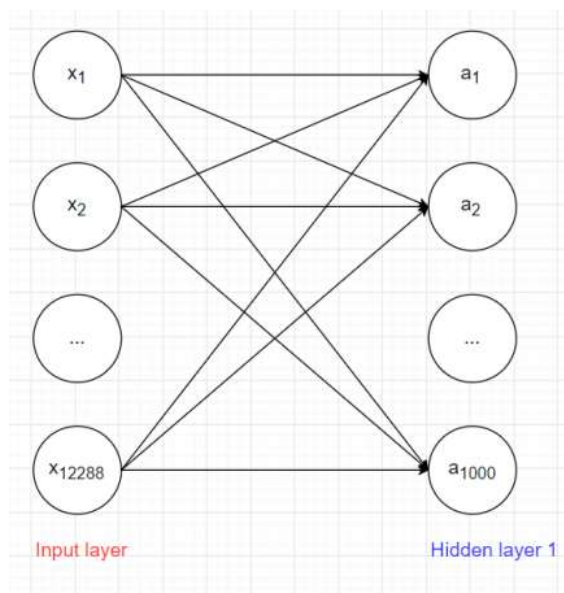


Mô hình neural network.

Mỗi hidden layer được gọi là **fully connected layer**, tên gọi theo đúng ý nghĩa, mỗi node trong hidden layer được kết nối với tất cả các node trong layer trước. Cả mô hình được gọi là **fully connected neural network (FCN)**.

Vấn đề của fully connected neural network với xử lý ảnh:

Ảnh màu 64×64 được biểu diễn dưới dạng 1 tensor $64 \times 64 \times 3$. Nên để biểu thị hết nội dung của bức ảnh thì cần truyền vào input layer tất cả các pixel ($64 \times 64 \times 3 = 12288$). Nghĩa là input layer giờ có 12288 nodes.



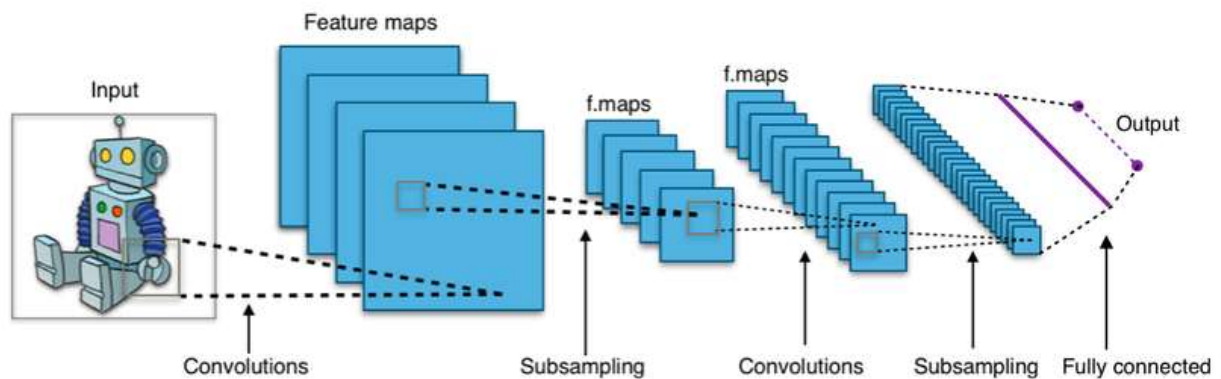
Input layer và hidden layer 1

Giả sử số lượng node trong hidden layer 1 là 1000. Số lượng weight W giữa input layer và hidden layer 1 là $12288 \times 1000 = 12288000$, số lượng bias là 1000 \Rightarrow tổng số parameter là: 12289000. Đây mới chỉ là số parameter giữa input layer và hidden layer 1, trong model còn nhiều layer nữa, và nếu kích thước ảnh tăng, ví dụ 512×512 thì số lượng parameter tăng cực kì nhanh. Do đó cần một giải pháp tốt hơn.

Nhận xét:

- Trong ảnh các pixel ở cạnh nhau thường có **liên kết với nhau hơn** là những pixel ở xa. Ví dụ như phép tính [convolution](#). Để tìm biên trong ảnh, ta áp dụng sobel kernel trên mỗi vùng kích thước 3×3 . Hay làm nét ảnh ta áp dụng sharpen kernel cũng trên vùng có kích thước 3×3 .
- Trong phép tính convolution trong ảnh, chỉ 1 kernel được dùng trên toàn bộ bức ảnh. Hay nói cách khác là các pixel ảnh **chia sẻ** hệ số với nhau.

\Rightarrow Áp dụng phép tính convolution vào layer trong neural network ta có thể giải quyết được vấn đề lượng lớn parameter mà vẫn lấy ra được các đặc trưng của ảnh.

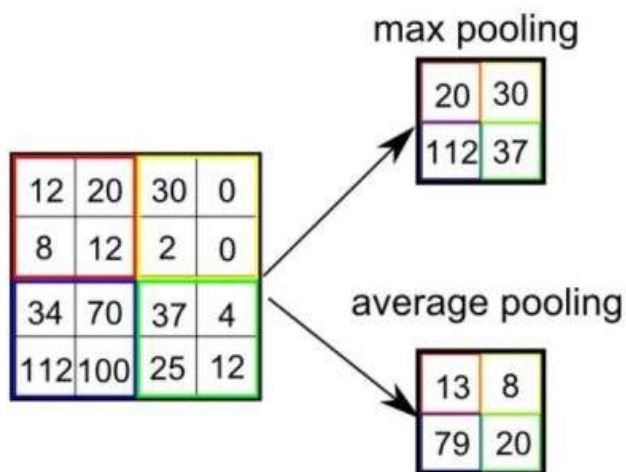


Cấu trúc chung của một mạng CNN

4.3.2. Tầng Pooling/Subsampling

Pooling layer thường được dùng giữa các convolutional layer, để giảm kích thước dữ liệu nhưng vẫn giữ được các thuộc tính quan trọng. Kích thước dữ liệu giảm giúp giảm việc tính toán trong model.

Gọi pooling size kích thước $K \times K$. Input của pooling layer có kích thước $H \times W$, trên vùng kích thước $K \times K$ trên ma trận ta tìm maximum hoặc average của dữ liệu rồi viết vào ma trận kết quả. Quy tắc về stride và padding áp dụng như phép tính convolution trên ảnh.

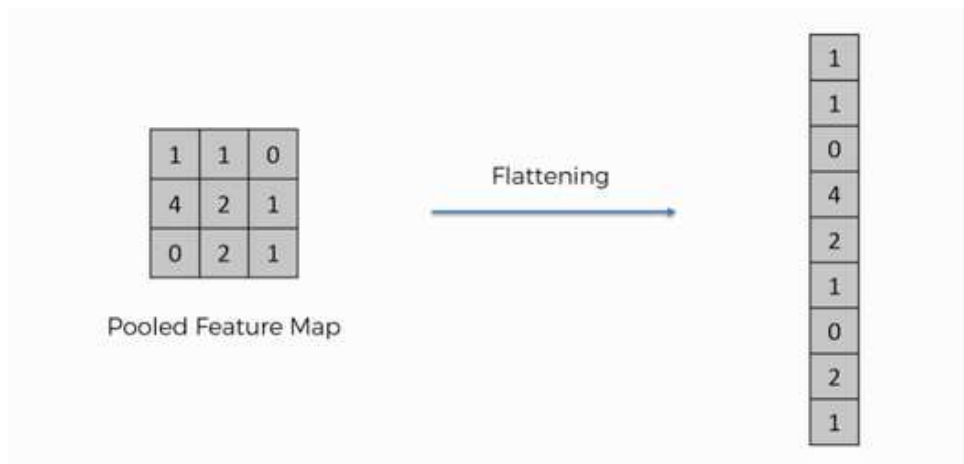


Ví dụ Max Pooling và Average Pooling(2x2)

Trong một số model người ta dùng convolutional layer với stride > 1 để giảm kích thước dữ liệu thay cho pooling layer.

4.3.3. Tầng kết nối đầy đủ FC

Sau khi ảnh được truyền qua nhiều convolutional layer và pooling layer thì model đã học được tương đối các đặc điểm của ảnh (ví dụ mắt, mũi, khung mặt,...) thì tensor của output của layer cuối cùng sẽ được chuyển về 1 vector.



Sau đó ta dùng các fully connected layer để kết hợp các đặc điểm của ảnh để ra được output của model. Dưới đây là một mô hình CNN dùng để nhận dạng ký tự viết tay:

