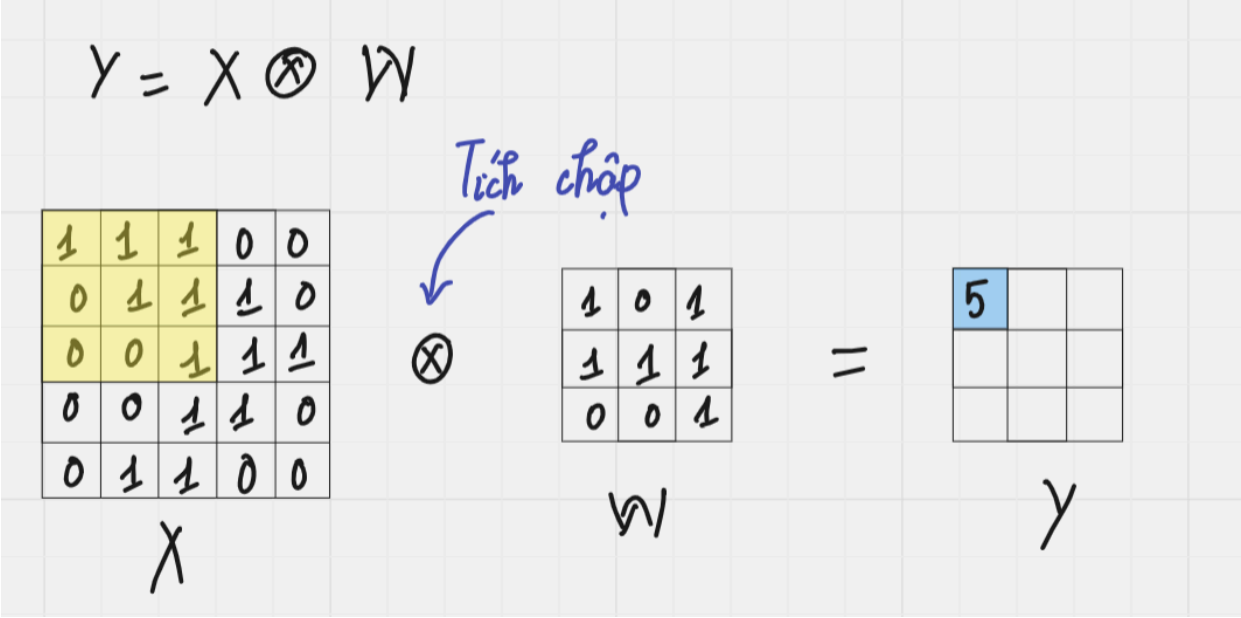
Convolutional Neural Network

Trong Python thì một ảnh có lớn hơn 3 chiều thì nó sẽ được gọi là **Tensor**

Trong mạng CNN, một bộ lọc (filter matrix) thì sẽ được gọi là Kernel. Đóng vai trò như một hệ số.

Convolution (Tích chập)

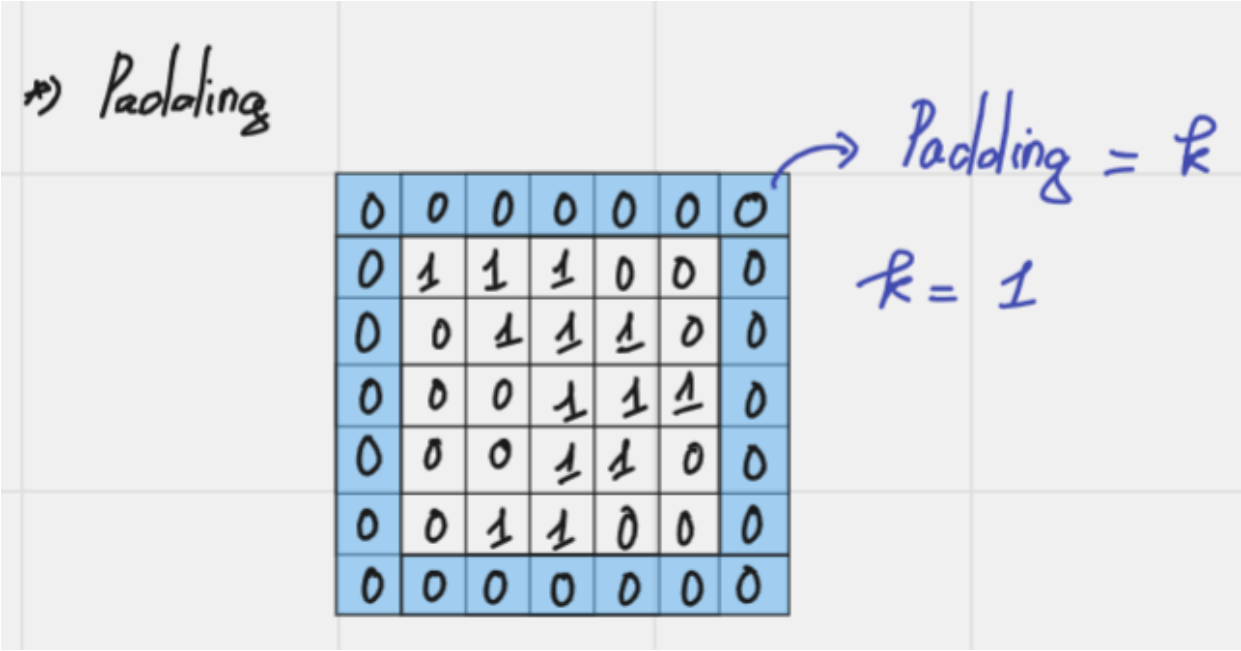
Trong học máy, các bài toán hồi quy, mục tiêu là đi tìm những tham số w1, w2,.. để xây dựng mô hình, thì tương tự với bài toán CNN. Mục tiêu trong bài toán CNN là đi tìm các hệ số Kernel



Padding (Đệm)

Mỗi lần thực hiện nhân tích chập thì ảnh thu được sẽ có kích thước nhỏ hơn so với ảnh ban đầu. Nếu muốn ma trận Y có kích thước bằng ma trận X thì cần phải bổ sung các giá trị 0 ở viền ngoài ma trận X

Padding = k → Thêm k vector 0 và mỗi phía của ma trận X



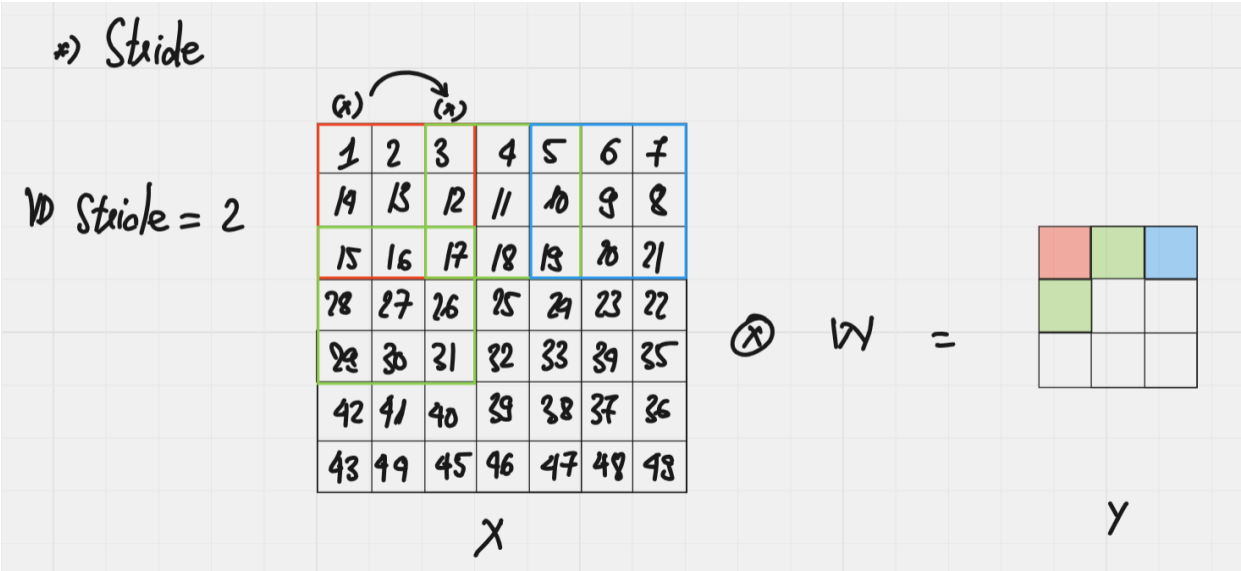
Stride (Bước nhảy)

Stride được sử dụng khi muốn giảm kích thước của ảnh đầu ra

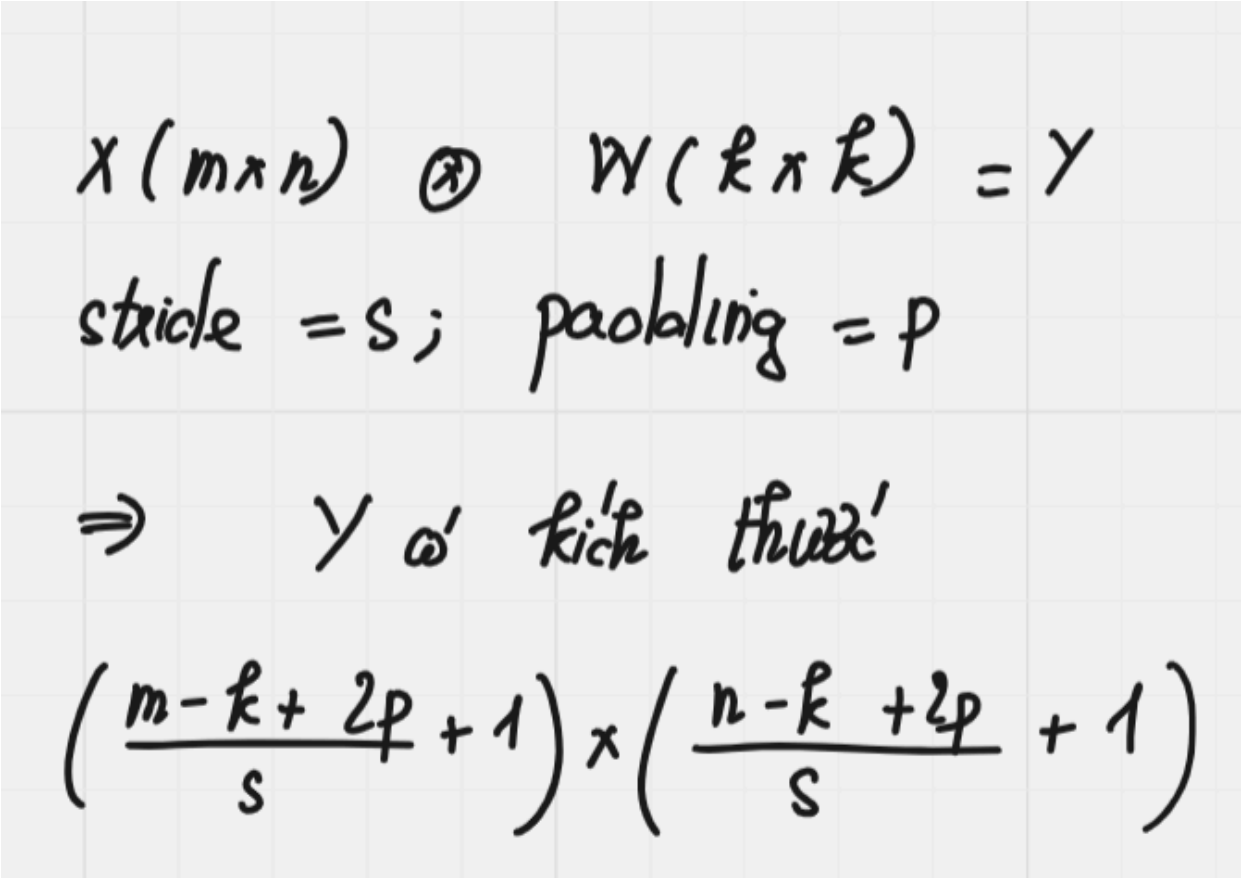
Khi stride = 1 → Di chuyển các kernel 1 pixel

Khi stride = 2 → Di chuyển các kernel 2 pixel

VD: stride = 2



Công thức tổng quát sau khi thực hiện nhân chập của ma trận X (m*n) với kernel (k*k), stride = s, padding = p



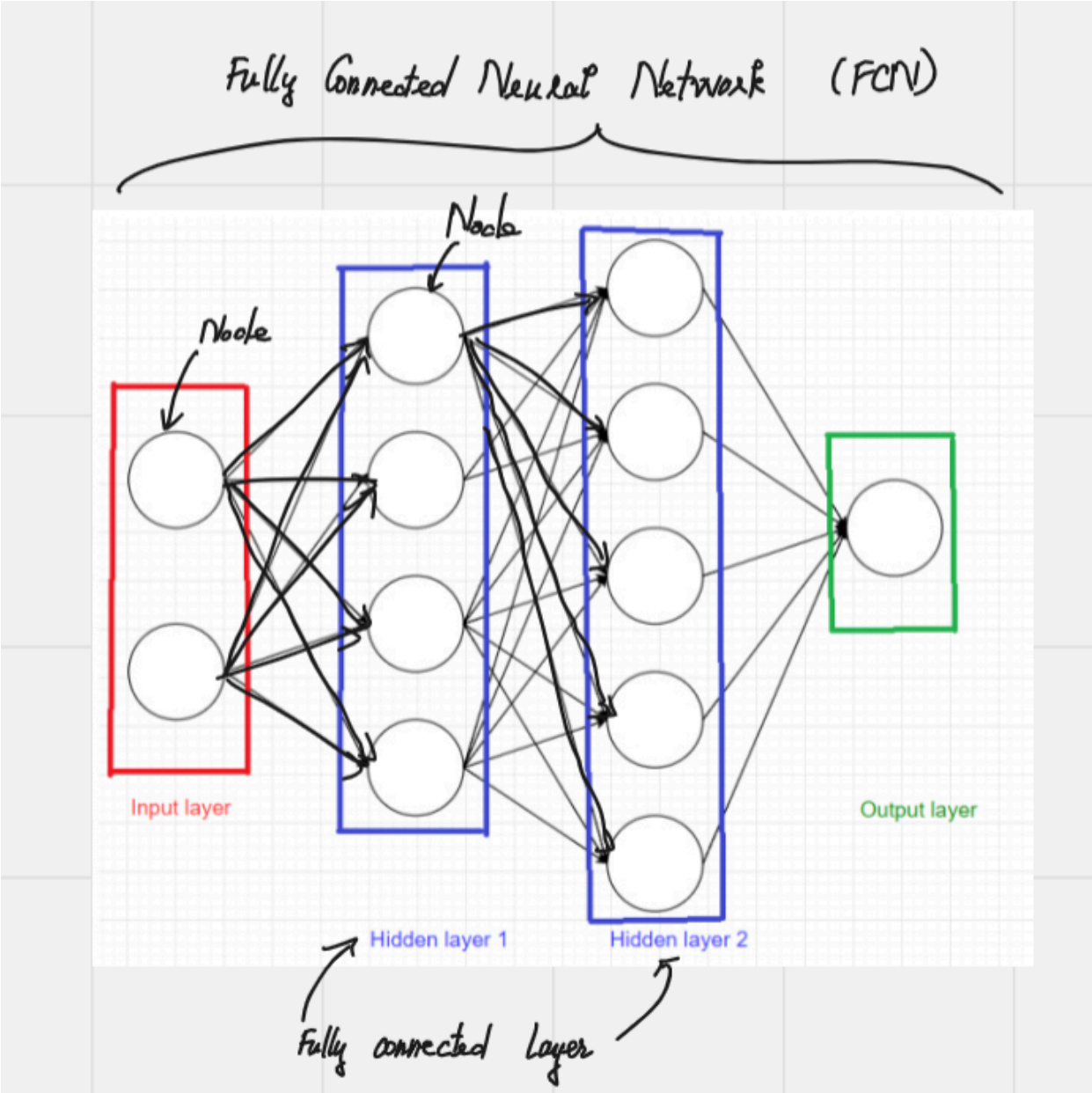
Ý nghĩa của Convolution

Mục đích của các phép tính convolution trên ảnh là làm mờ, làm nét, xác định các đường. Mỗi kernel khác nhau thì các phép tính convolution sẽ khác nhau.

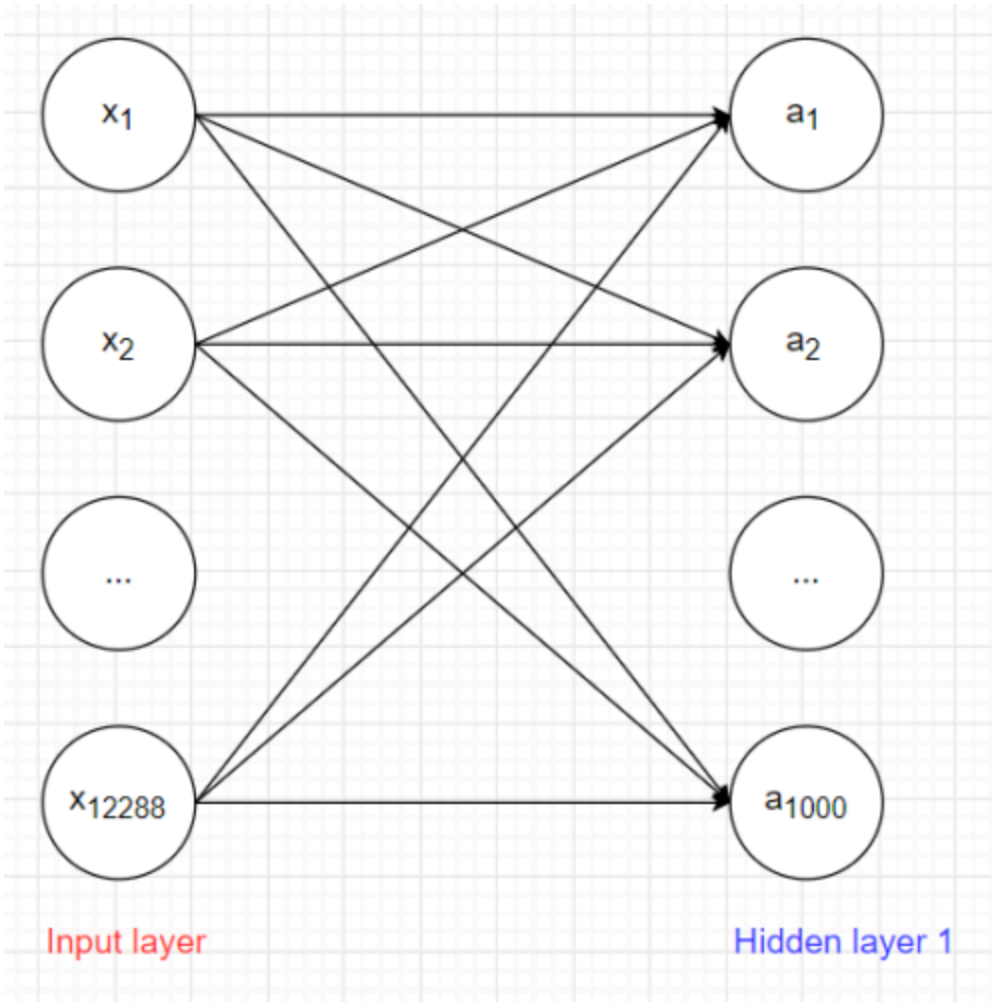
Môn Xử lý ảnh, với những bài toán nâng cao chất lượng ảnh: bộ lọc trung bình, bộ lọc trung bình có trọng số, các bài toán phát hiện biên đều cho thấy được ý nghĩa của tích chập.

Mạng CNN

1. **Convolutional Layer**



Vấn đề trong bài toán

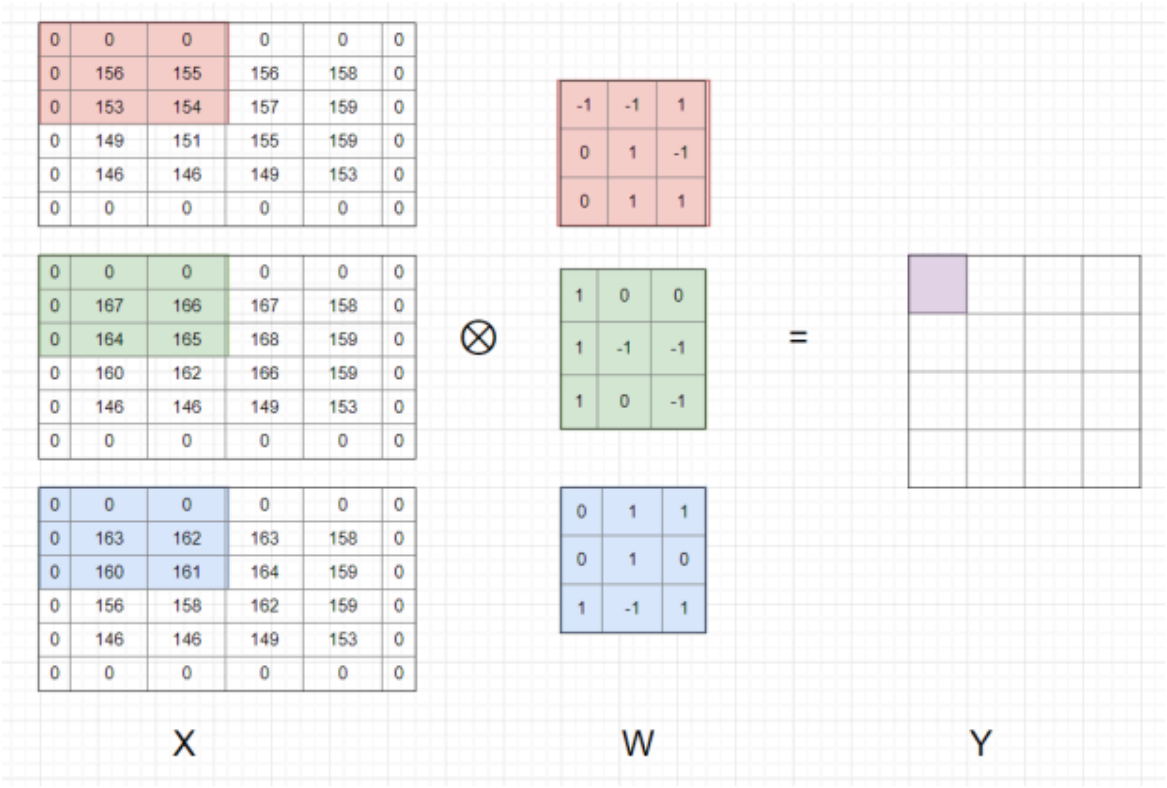


Số lượng parameter tăng cực kỳ nhanh, vậy nên cần có hướng giải quyết tốt khác.

Áp dụng phép tính convolution vào các layer sẽ giải quyết được vấn đề số lượng lớn parameter mà vẫn lấy được các đặc trưng của ảnh.

Convolution layer đầu tiên

Ảnh màu có 3 kênh màu: Red, Green, Blue



Khi biểu diễn ở dạng tensor 3 chiều thì có 3 chỉ số: i,j, và chỉ số độ sâu k.

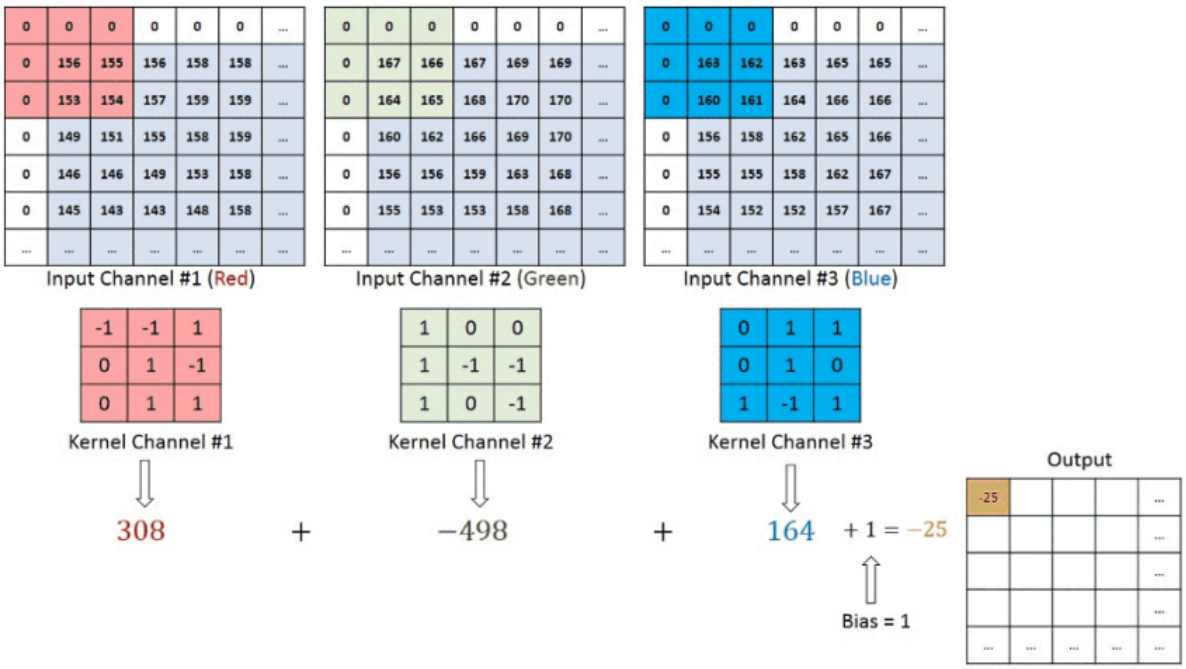
VD:

y11 = b + **(x111 * w111 + x121 * w121 + x131 * w131 + x211 * w211 + x221 * w221 + x231 * w231 + x311 * w311+x321 *w321+x331 *w331)** **+** **(x112 *w112+x122 *w122+x132 *w132+x212 *w212+x222 *w222+ x232 *w232 +x312 *w312 +x322 *w322 +x332 *w332)** **+** **(x113 *w113 +x123 *w123 +x133 *w133 +x213 *w213 +x223 * w223 +x233 * w233 +x313 * w313 +x323 * w323 +x333 * w333)** **= -25**

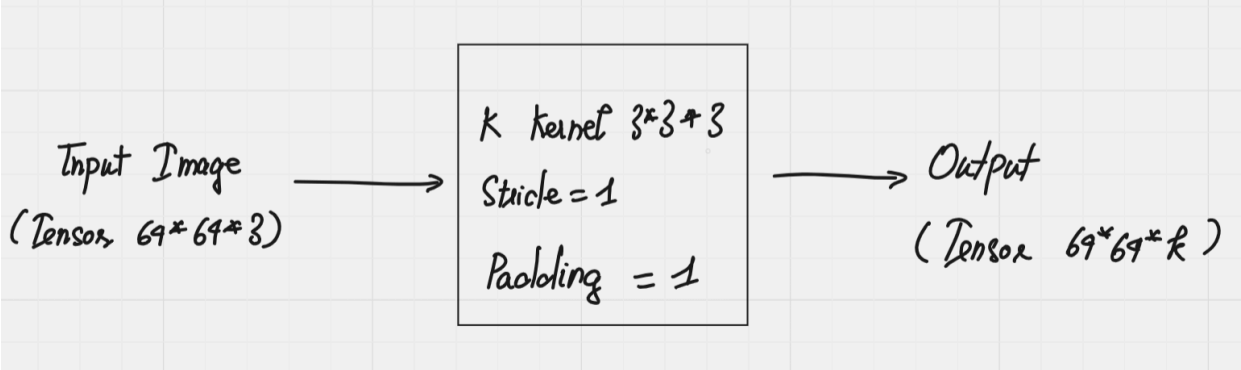
Nhận xét:

- Output Y của phép tính Convolution trên ảnh màu là 1 matrix
- Có 2 hệ số Bias được cộng vào sau bước tính tổng các phần tử của phép tính element-wise

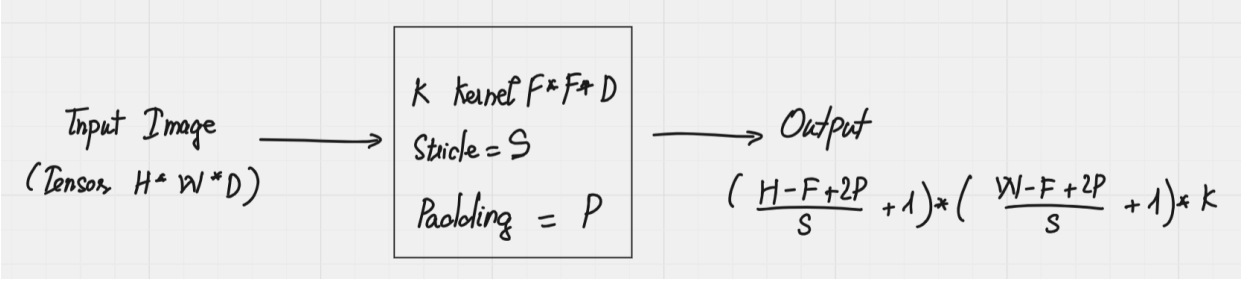
Các quy tắc đối với padding và stride



Với mỗi kernel khác nhau sẽ học được những đặc trưng khác nhau của ảnh, nên trong mỗi convolutional layer ta sẽ dùng nhiều kernel để học được nhiều thuộc tính của ảnh. Vì 1 kernel sẽ cho ra được 1 matrix, nên k kernel sẽ cho ra được k output matrix. Kết hợp k output matrix ta sẽ thành 1 tensor 3 chiều có chiều sâu k.



Công thức tổng quát



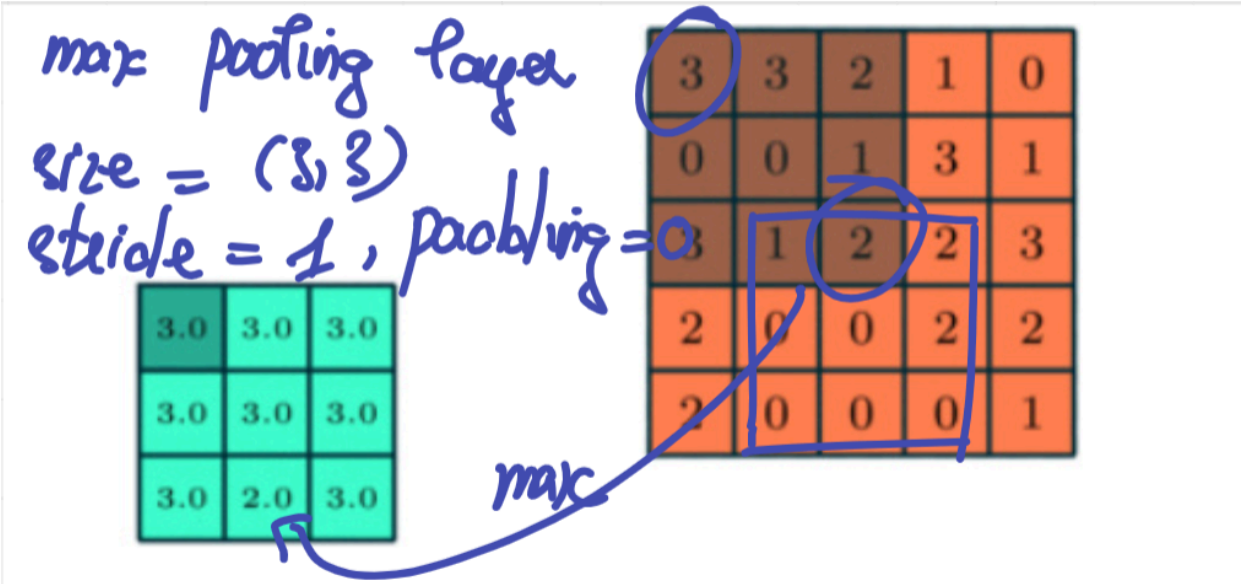
Lưu ý:

- Output của Convolutional Layer sẽ qua hàm Non-linear Activation Function trước khi trở thành Input của Convolutional Layer tiếp theo
- Tổng số Parameter của Layer: Mỗi kernel có kích thước F*F*D và có 1 hệ số bias, nên tổng parameter của 1 kernel là F*F*D + 1. Mà convolutional layer áp dụng K kernel => Tổng số parameter trong layer này là K * (F*F*D + 1)

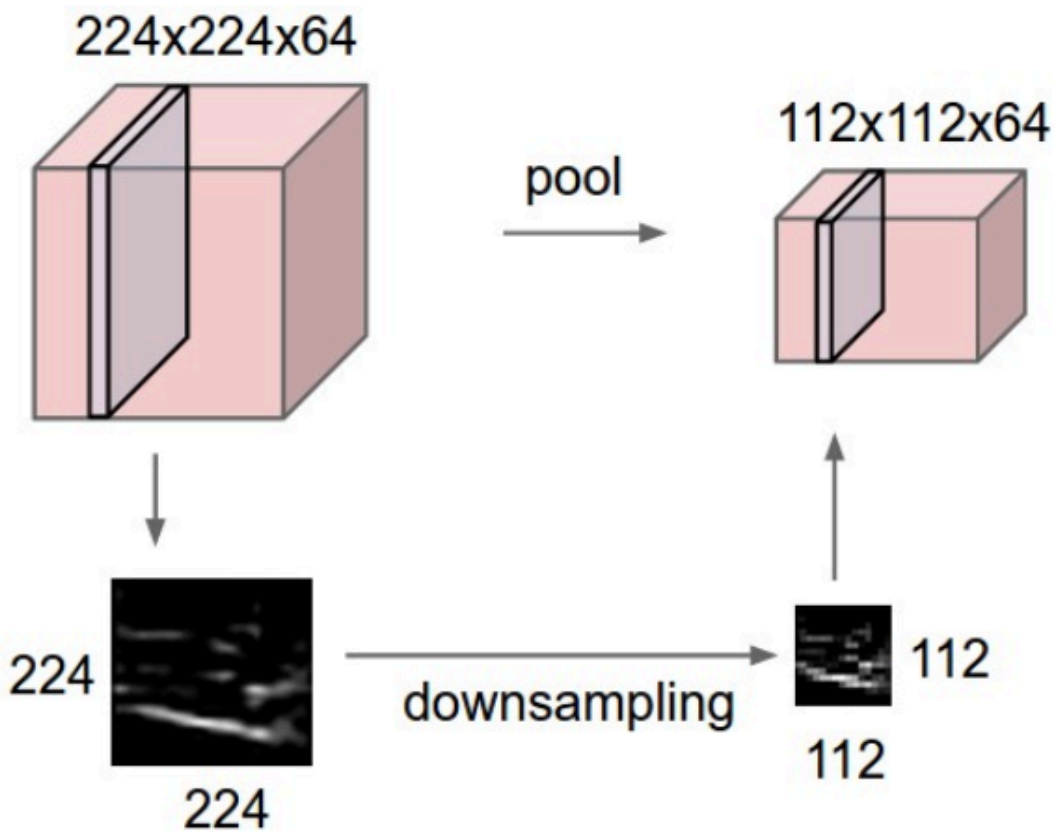
2. Pooling Layer

Pooling Layer thường được dùng giữa các Convolution layer để làm giảm kích thước dữ liệu nhưng vẫn giữ được các thuộc tính quan trọng. Việc giảm kích thước dữ liệu cho phép giúp giảm các phép tính trong model.

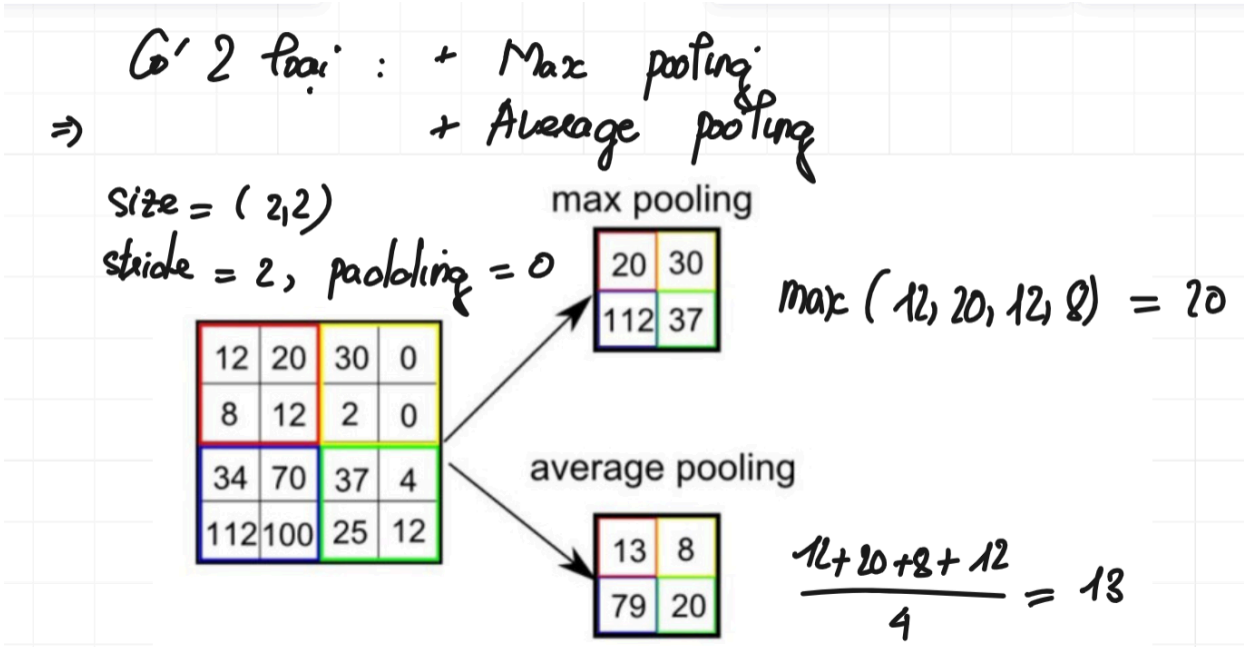
Gọi pooling size kích thước K*K. Input của pooling layer có kích thước H*W*D, ta tách ra làm D ma trận kích thước H*W. Với mỗi ma trận, trên vùng kích thước K*K trên ma trận ta tìm **maximum** hoặc **average** của dữ liệu rồi viết vào ma trận kết quả. Quy tắc về stride và padding áp dụng như phép tính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



Nhưng hầu hết khi dùng pooling layer thì sẽ dùng size=(2,2), stride=2, padding=0. Khi đó output width và height của dữ liệu giảm đi một nửa, depth thì được giữ nguyên.



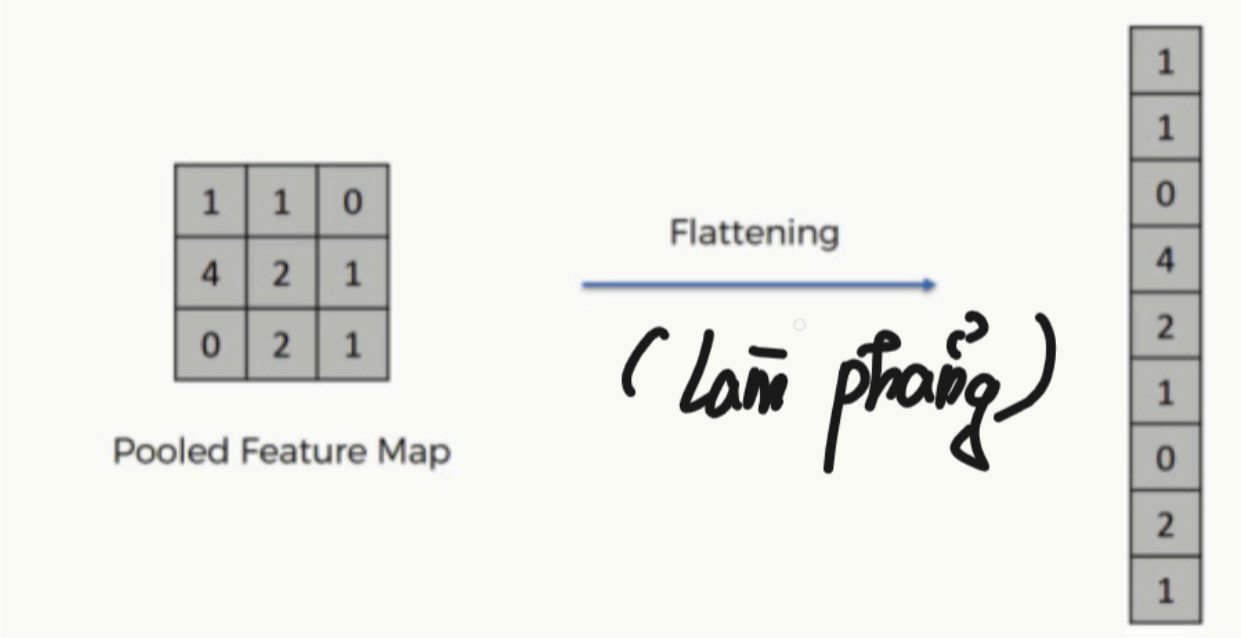
Có 2 loại pooling layer phổ biến:



3. Fully Connected Layer

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, khuôn mặt,...) thì tensor của output

của layer cuối cùng, kích thước H*W*D, sẽ được chuyển về 1 vector kích thước (H*W*D, 1)



Sau đó ta dùng các fully connected layer để kết hợp các đặc điểm của ảnh để ra được output của model.

Giới thiệu Keras và bài toán phân loại ảnh

Các bước cơ bản để xây dựng một bài toán Deep Learning

- Xây dựng bài toán
- Chuẩn bị tập dữ liệu (dataset)
- Xây dựng model
- Định nghĩa loss function
- Thực hiện Backpropagation (Lan truyền ngược) và áp dụng Gradient Descent để tìm ra các Parameter gồm weight (trọng số) và bias để tối ưu hàm loss function
- Dự đoán dữ liệu mới bằng model và các hệ số tìm được ở trên

Xây dựng bài toán Mnist Dataset

1. Xây dựng bài toán

Đầu vào là một ảnh chữ số viết tay từ 1 đến 9. Nhiệm vụ là dự đoán chữ số đấy là số mấy.

2. Chuẩn bị dữ liệu

Mục đích là dự đoán đúng những dữ liệu từ thực tế chứ không phải là dự đoán đúng các dữ liệu có sẵn. Bình thường thường chia bộ dữ liệu ra làm 3 phần: Training set, Validation set, và Test set.

Trong trường hợp có nhiều hơn một mô hình thì mô hình nào cho kết quả tốt hơn trên tập validation test sẽ được lựa chọn, và cuối cùng model tốt nhất sẽ được đánh giá trên tập test set

3. Xây dựng model