

# Convolutional Neural Network

Nhóm 3

# Nội dung

01. Giới thiệu CNN
02. CNN cơ bản
03. Các kiến trúc  
mạng CNN
04. Thực hiện huấn luyện



# **1. Giới thiệu CNN**

**1.1. Lịch sử của CNN**

**1.2. CNN và thị giác con người**

**1.3. Kiến trúc tổng quan của CNN**

# Giới thiệu CNN

- Mô hình học sâu **phổ biến nhất và có ảnh hưởng nhất** trong cộng đồng Thị giác máy tính.
- CNN là **xương sống (backbone)** của nhiều **hệ thống thị giác máy tính hiện đại** ngày nay.
- CNN được sử dụng trong nhiều trong:
  - Nhận dạng hình ảnh
  - Phân tích video
  - Hình ảnh MRI (y tế)
  - Xử lý ngôn ngữ tự nhiên.

# 1.1. Lịch sử của CNN neural network

Năm 1959, David Hubel và Torsten Wiesel đã mô tả “các tế bào đơn giản” và “các tế bào phức tạp” trong **vỏ não thị giác** của con người. Họ đề xuất rằng cả hai loại tế bào đều được sử dụng trong nhận dạng mâu.

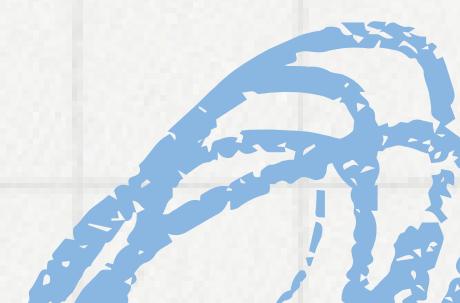
Năm 1980, Kunihiko Fukushima lấy cảm hứng từ công trình của Hubel và Wiesel về các tế bào đơn giản và phức tạp, nên đã đề xuất mô hình “neocognitron”. Đây là tiền thân sớm nhất của CNN. Các khái niệm về feature extraction, pooling layers, sử dụng convolution trong mạng nơ-ron và cuối cùng là nhận dạng/ phân loại đã được giới thiệu.



# 1.1. Lịch sử của CNN neural network

Cái tên mạng nơ-ron tích chập thực sự bắt nguồn từ thiết kế mạng “LeNet” của Yann LeCun và cộng sự. Nó được phát triển phần lớn từ năm 1989 đến năm 1998 cho nhiệm vụ nhận dạng chữ số viết tay.

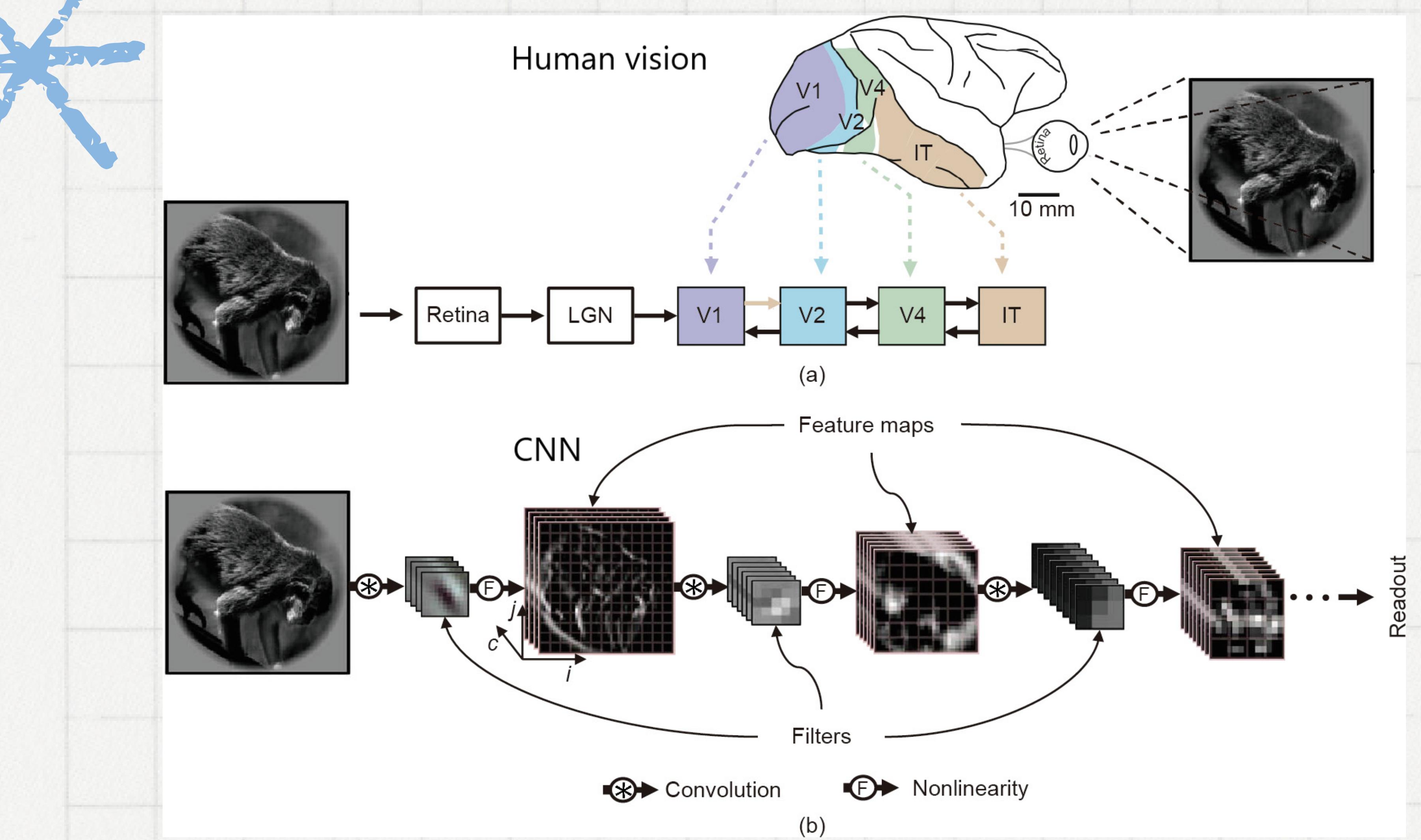
Năm 2012, CNN có mức độ phổ biến tăng vọt (điều này vẫn tiếp tục cho đến ngày nay), sau khi Alex Krizhevsky và Geoffrey Hinton đã xây dựng mô hình CNN (**AlexNet**) và sử dụng GPU để tăng tốc độ huấn luyện và đạt được top 1 trong cuộc thi Computer Vision thường niên của ImageNet, tạo nên làn sóng mạnh mẽ sử dụng deep CNN với sự trợ giúp của GPU để giải quyết các vấn đề về thị giác máy tính.



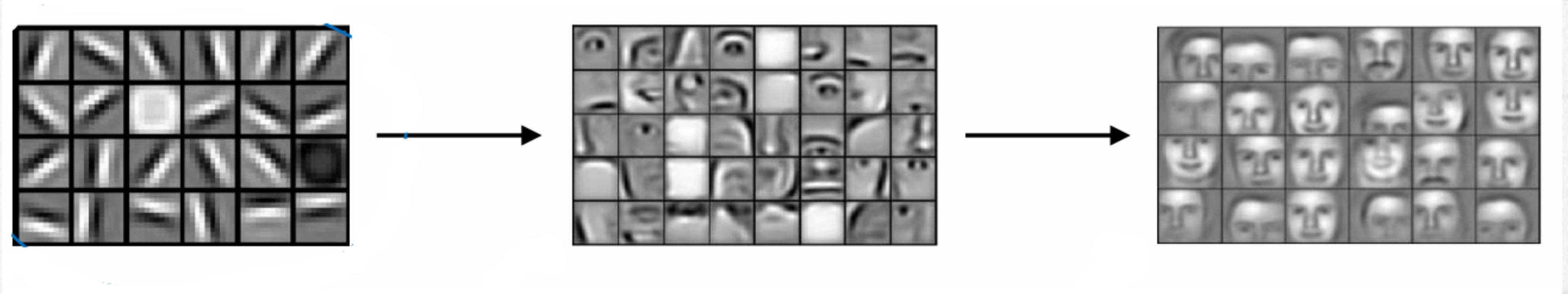
## 1.2. CNN và Thị giác con người

Các mô hình mạng neuron nhân tạo “được lấy cảm hứng trực tiếp từ bộ não con người”.

CNN và hệ thống thị giác của con người tuân theo **cấu trúc phân cấp “từ đơn giản đến phức tạp”**. Tuy nhiên, việc triển khai thực tế lại hoàn toàn khác; bộ não được xây dựng dựa trên các **tế bào thần kinh sinh học** và mạng CNN được xây dựng dựa trên **các phép toán**.

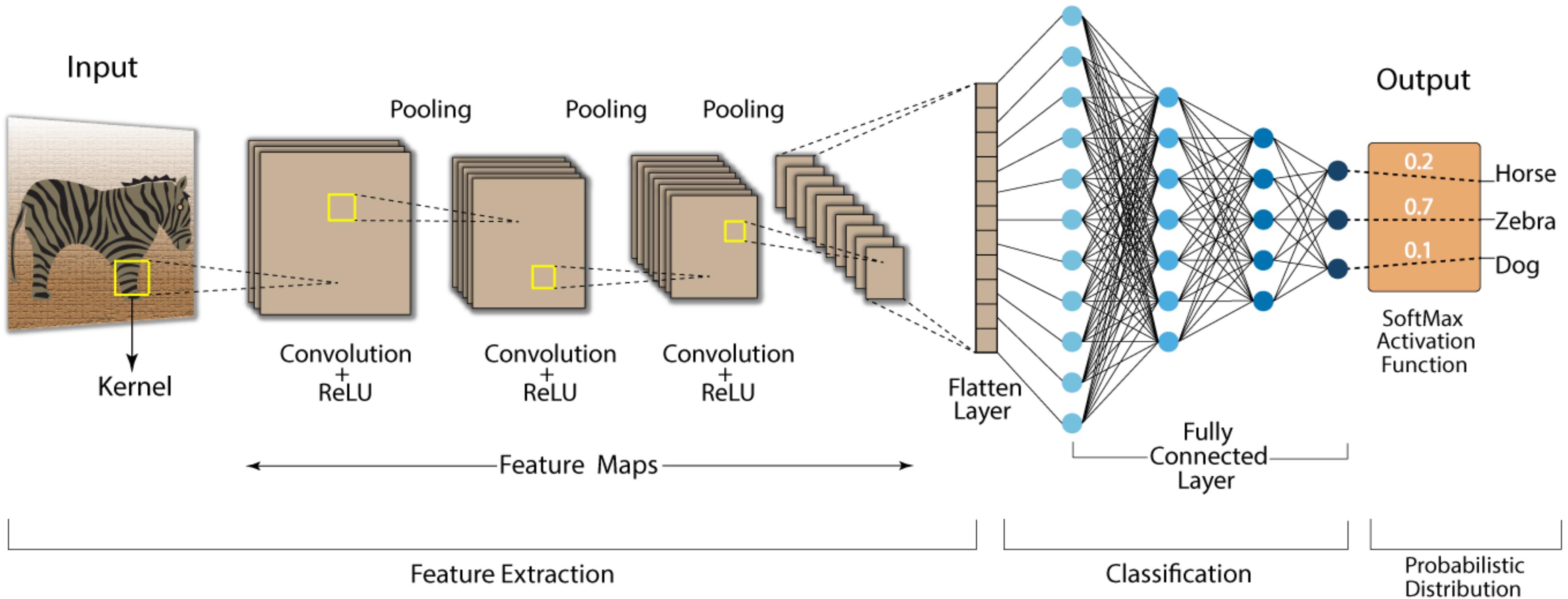


# Biểu diễn phân cấp của CNN



# Kiến trúc tổng quan của CNN

**Convolution Neural Network (CNN)**



## **2. CNN cơ bản**

2.1. Convolutional layer

2.2. Pooling layer

2.3 Fully connected  
layer

# CONVOLUTIONAL LAYER

Phép tính Convolution

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



1	0	1
0	1	0
1	0	1

=

1		

X  
(m\*n)

W  
(k\*k)

Y  
(m-k+1)\*(n-k+1)

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

# CONVOLUTIONAL LAYER

Phép tính Convolution

1 <small><math>\times 1</math></small>	1 <small><math>\times 0</math></small>	1 <small><math>\times 1</math></small>	0	0
0 <small><math>\times 0</math></small>	1 <small><math>\times 1</math></small>	1 <small><math>\times 0</math></small>	1	0
0 <small><math>\times 1</math></small>	0 <small><math>\times 0</math></small>	1 <small><math>\times 1</math></small>	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved  
Feature

# CONVOLUTIONAL LAYER

## Padding

Kích thước ma trận gốc:  
 $(m+p)*(n+p)$

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

# CONVOLUTIONAL LAYER

Stride

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

$x 1+i*k, 1+j*k$

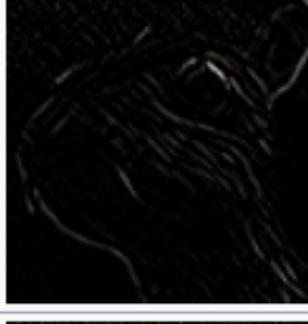
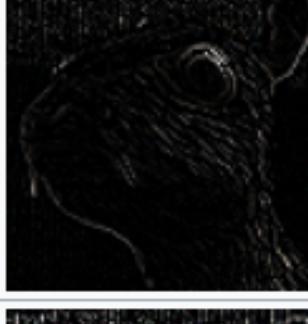
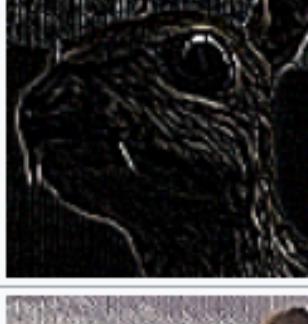
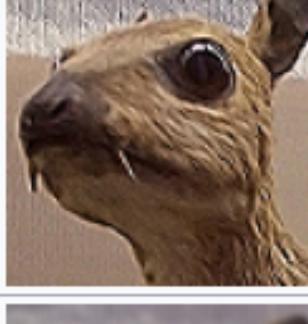
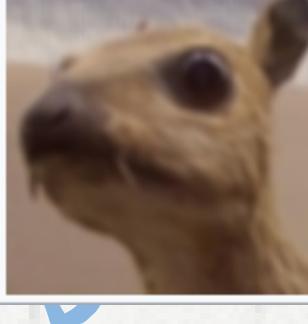
0	0	0	0	0	0	0
0	1	1	1	0	0	0
0	0	1	1	1	0	0
0	0	0	1	1	1	0
0	0	0	1	1	0	0
0	0	1	1	0	0	0
0	0	0	0	0	0	0

Stride=2, padding=1

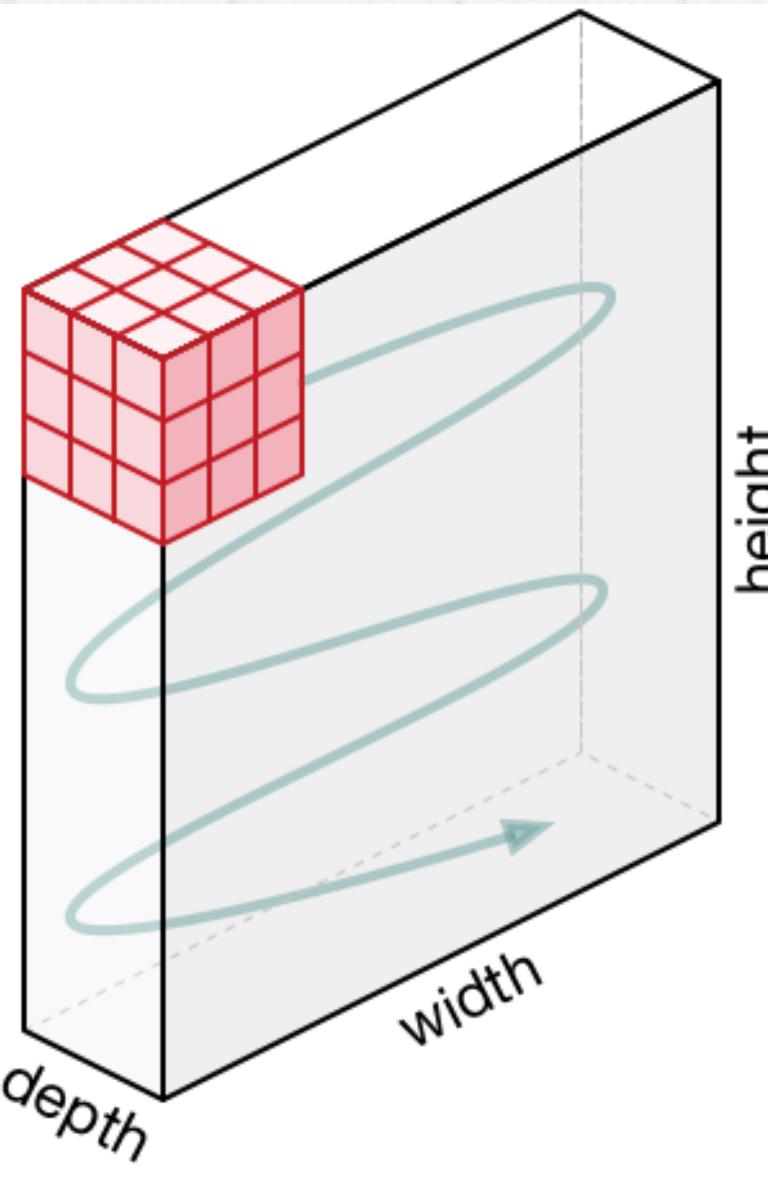
**Kích thước của ma trận sau khi thực hiện phép tính convolution với Stride=s và Padding=p**

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

# Ý nghĩa của phép tính **Convolution**

Operation	Kernel $\omega$	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)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

# Thực hiện trên ảnh màu



Phép tính convolution  
trên ảnh màu với  $k=3$ .

$$X \otimes W = Y$$

The diagram illustrates the computation of a 3x3x3 convolution kernel (W) applied to a 6x6x3 input tensor (X) to produce a 1x1x3 output tensor (Y). The input X is shown as three 6x6 matrices, each with a different color border: pink (top), green (middle), and blue (bottom). The kernel W is shown as three 3x3 matrices, also with colored borders: pink (top), green (middle), and blue (bottom). The result Y is a single 1x1x3 matrix with a purple border.

0	0	0	0	0	0
0	156	155	156	158	0
0	153	154	157	159	0
0	149	151	155	159	0
0	146	146	149	153	0
0	0	0	0	0	0

-1	-1	1
0	1	-1
0	1	1

0	0	0	0	0	0
0	167	166	167	158	0
0	164	165	168	159	0
0	160	162	166	159	0
0	146	146	149	153	0
0	0	0	0	0	0

1	0	0
1	-1	-1
1	0	-1

0	0	0	0	0	0
0	163	162	163	158	0
0	160	161	164	159	0
0	156	158	162	159	0
0	146	146	149	153	0
0	0	0	0	0	0

0	1	1
0	1	0
1	-1	1

Tensor X, W 3 chiều được viết  
dưới dạng 3 matrix.

# Thực hiện trên ảnh màu

0	0	0	0	0	0	0	...
0	156	155	156	158	158	158	...
0	153	154	157	159	159	159	...
0	149	151	155	158	159	159	...
0	146	146	149	153	158	158	...
0	145	143	143	148	158	158	...
...	...	...	...	...	...	...	...

Input Channel #1 (Red)

0	0	0	0	0	0	0	...
0	167	166	167	169	169	169	...
0	164	165	168	170	170	170	...
0	160	162	166	169	170	170	...
0	156	156	159	163	168	168	...
0	155	153	153	158	168	168	...
...	...	...	...	...	...	...	...

Input Channel #2 (Green)

0	0	0	0	0	0	0	...
0	163	162	163	165	165	165	...
0	160	161	164	166	166	166	...
0	156	158	162	165	166	166	...
0	155	155	158	162	165	167	...
0	154	152	152	157	157	167	...
...	...	...	...	...	...	...	...

Input Channel #3 (Blue)

-1	-1	1
0	1	-1
0	1	1

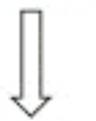
Kernel Channel #1



308

1	0	0
1	-1	-1
1	0	-1

Kernel Channel #2



-498

0	1	1
0	1	0
1	-1	1

Kernel Channel #3



164

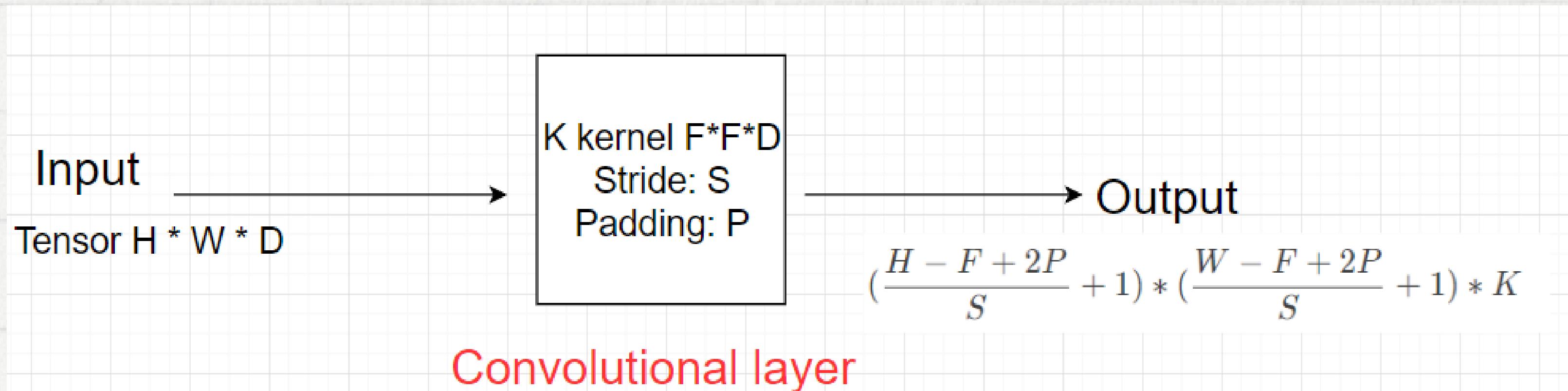
+ 1 = -25

Bias = 1

-25							...
							...
							...
							...
...	...	...	...	...	...	...	...

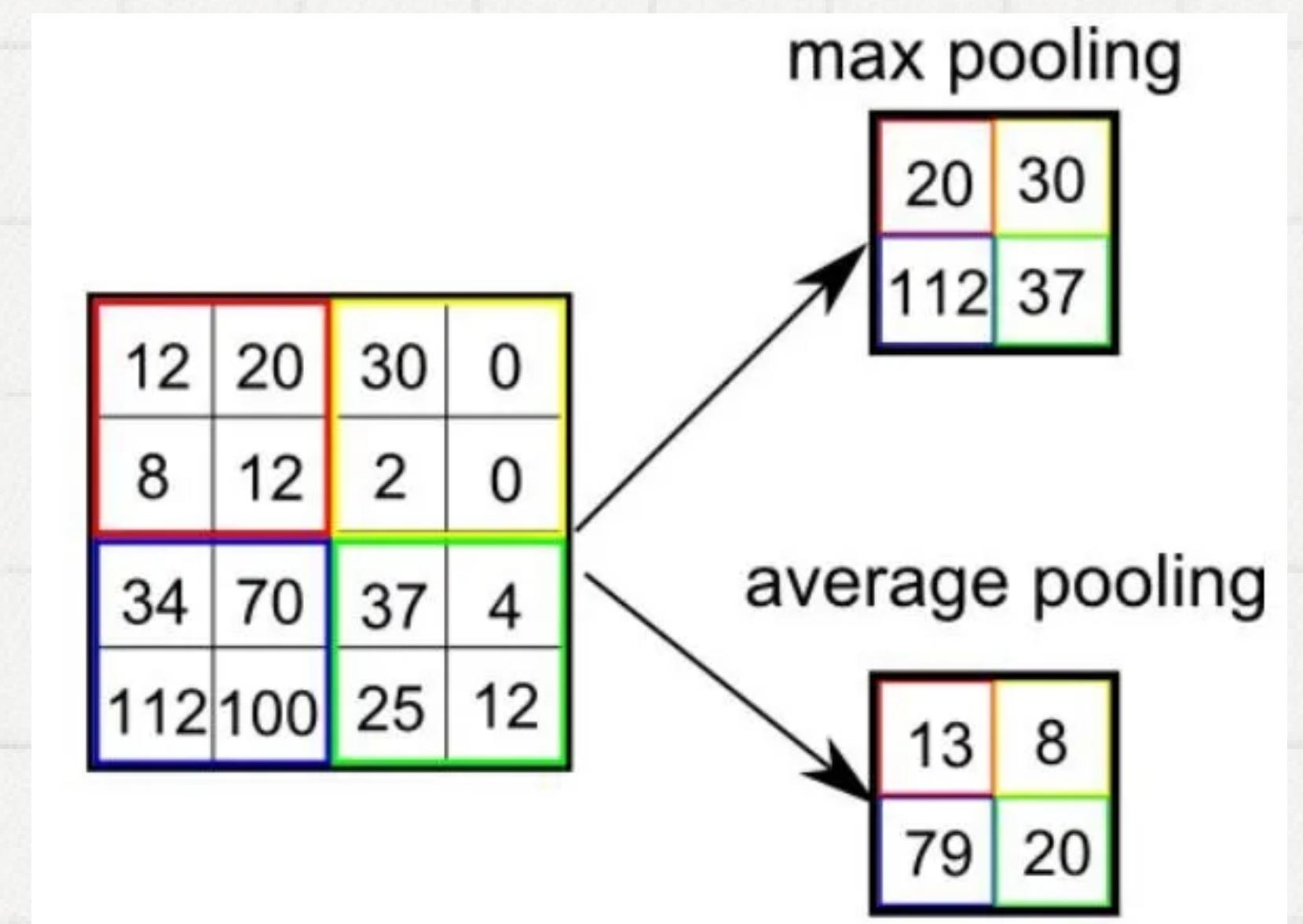
Output

- Output của convolutional layer sẽ qua hàm activation function trước khi trở thành input của convolutional layer tiếp theo.
- Tổng số parameter trong layer này là  $K * (F*F*D + 1)$ .



# POOLING LAYER

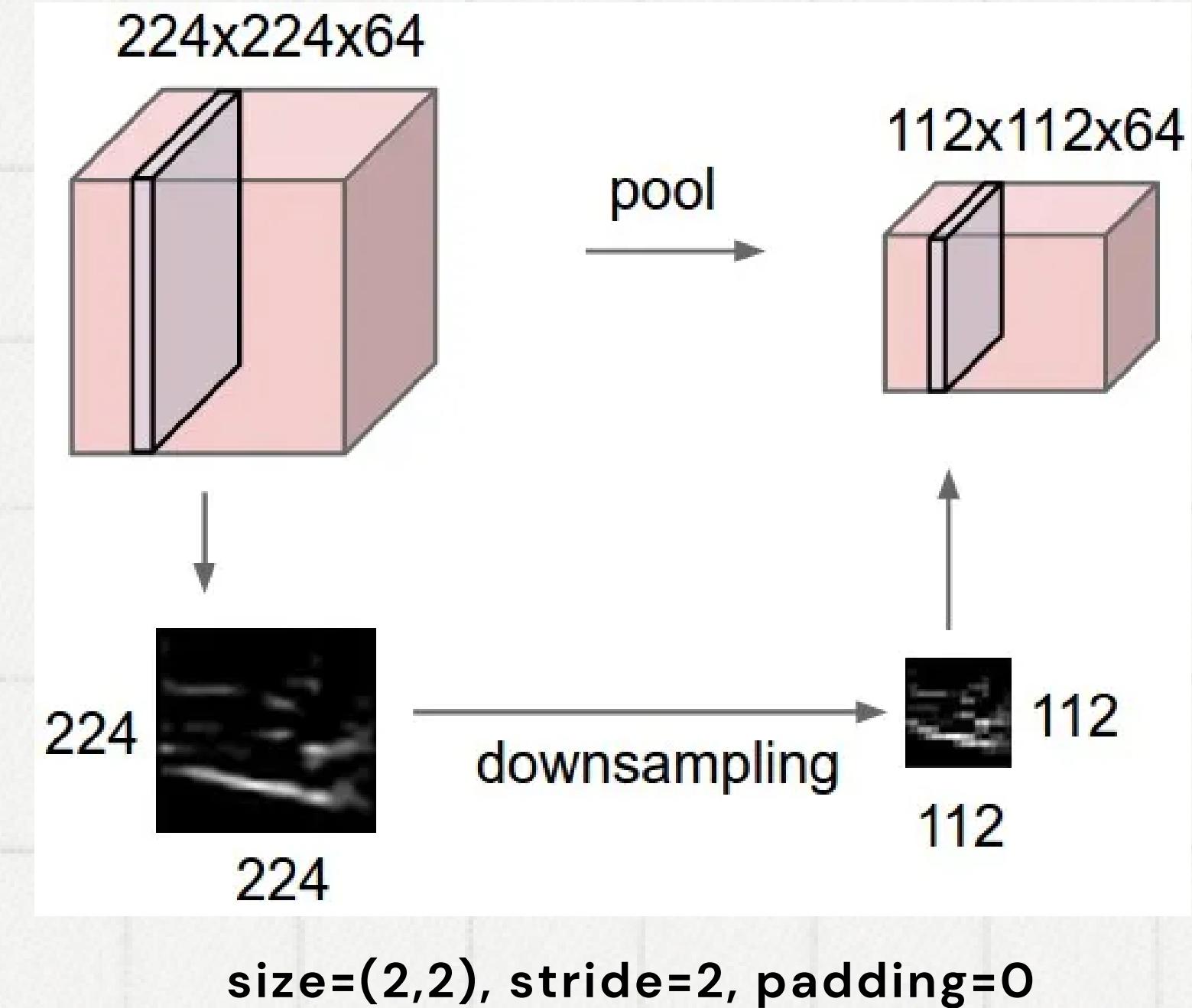
- Thực hiện quét tương tự như lớp convolution
- Giá trị tính toán là **max** (**max pooling**) hoặc **avg** (**avg pooling**)
- 2 loại pooling
  - Max pooling
  - Avg pooling



3.0	3.0	3.0
3.0	3.0	3.0
3.0	2.0	3.0

3	3	2	1	0
0	0	1	3	1
3	1	2	2	3
2	0	0	2	2
2	0	0	0	1

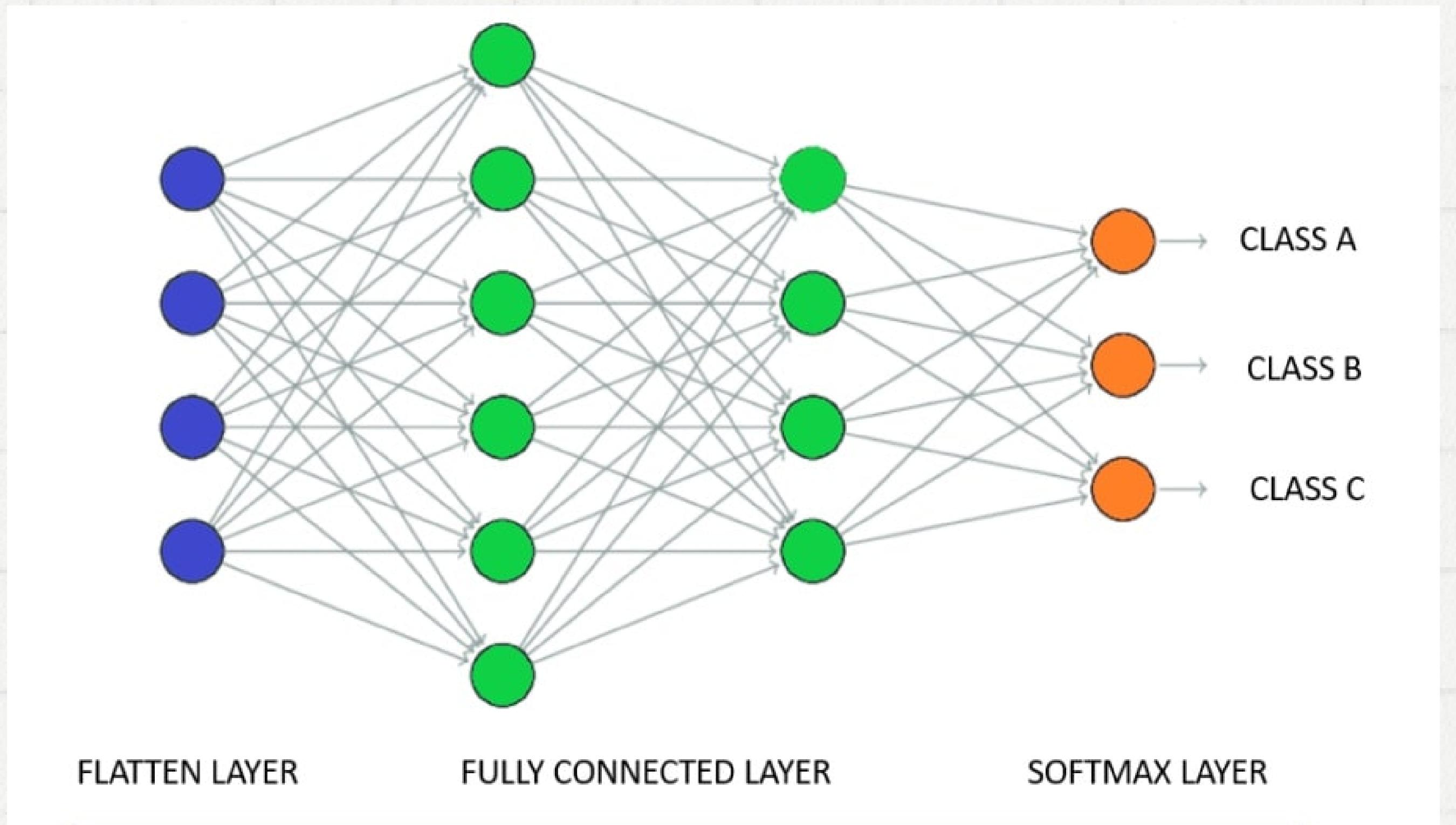
max pooling layer với  
size=(3,3), stride=1, padding=0

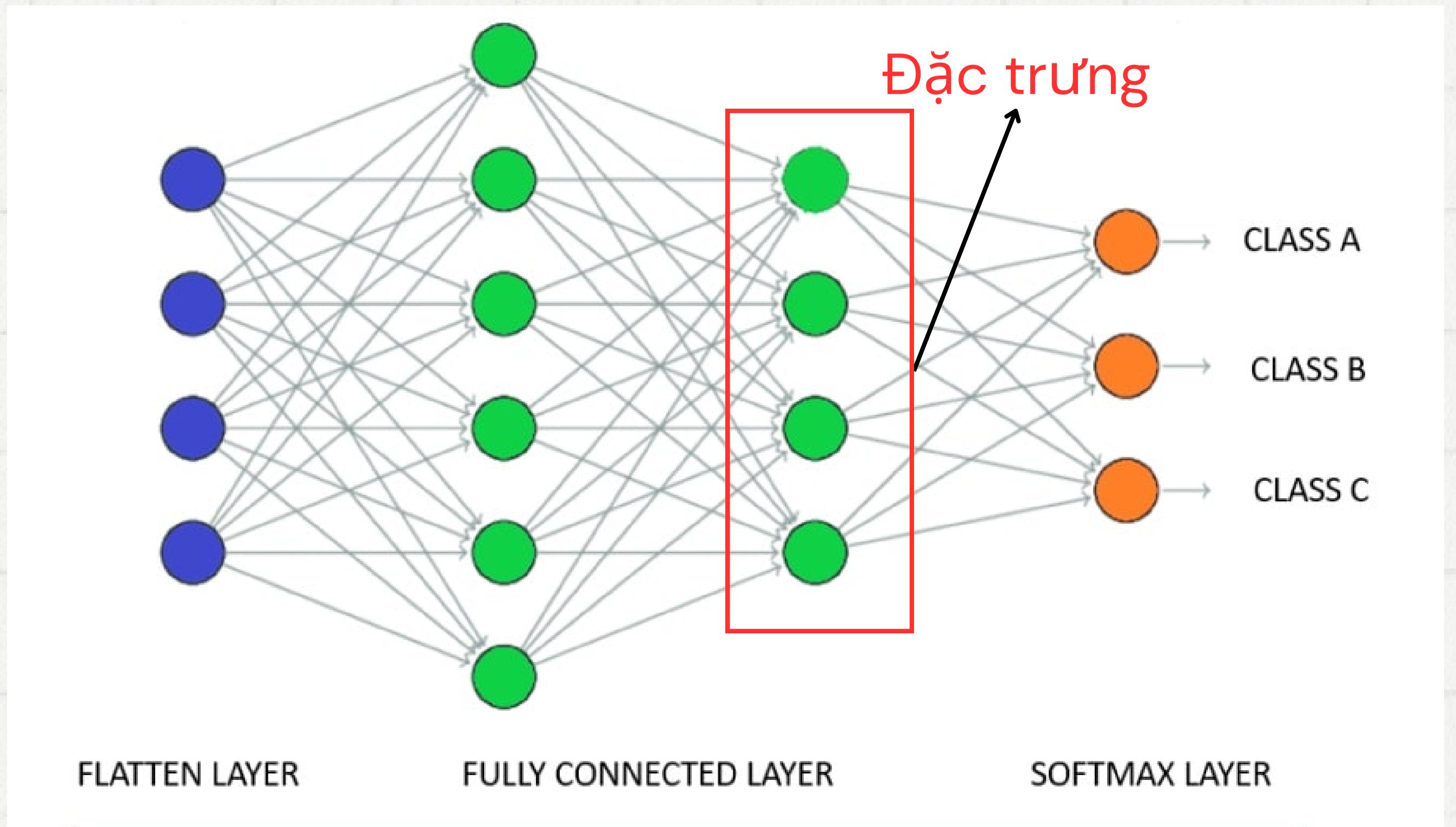


Trong một số model người ta dùng convolutional layer với stride > 1 để  
giảm kích thước dữ liệu thay cho pooling layer.

# Fully connected layer







# **4. Các kiến trúc mạng CNN**

# Các kiến trúc mạng CNN

01

ResNet

02

VGG

03

GoogleNet

04

DenseNet

05

MobileNet

06

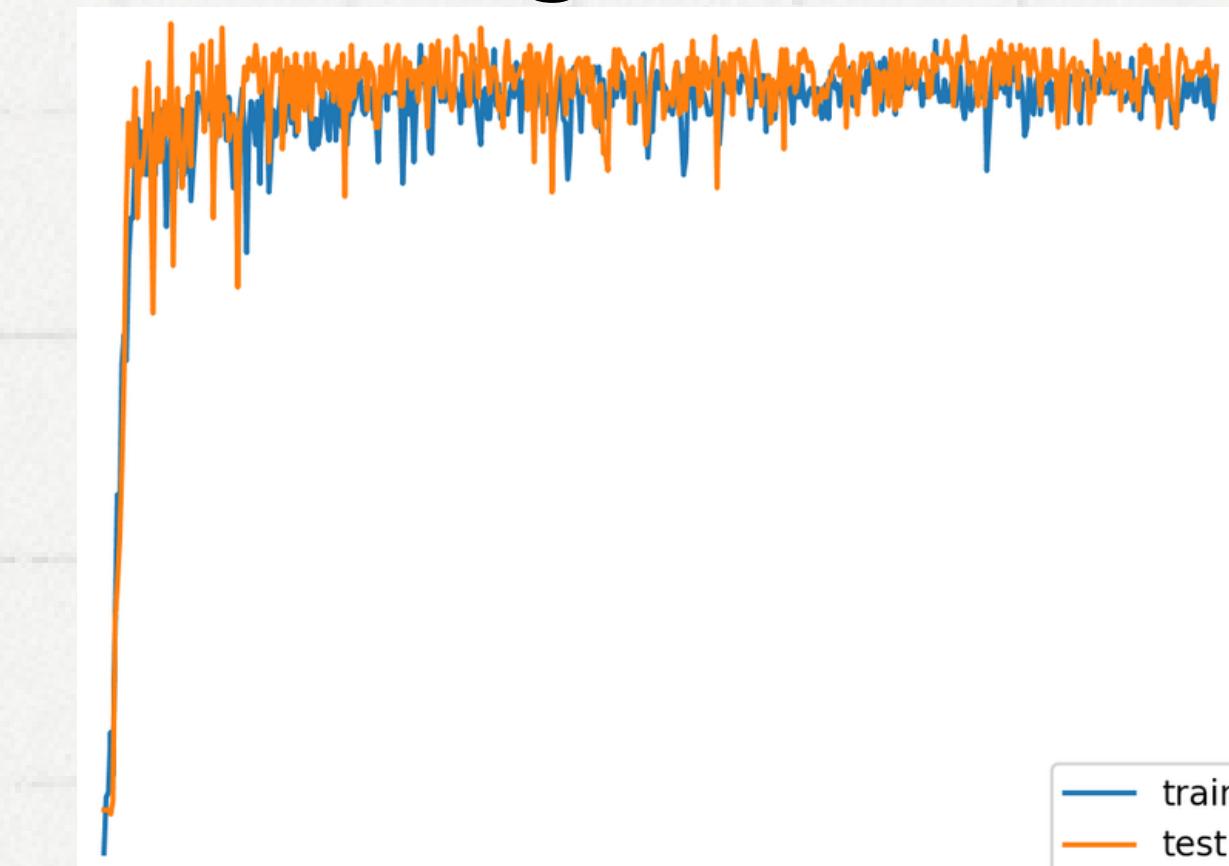
EffcienNet



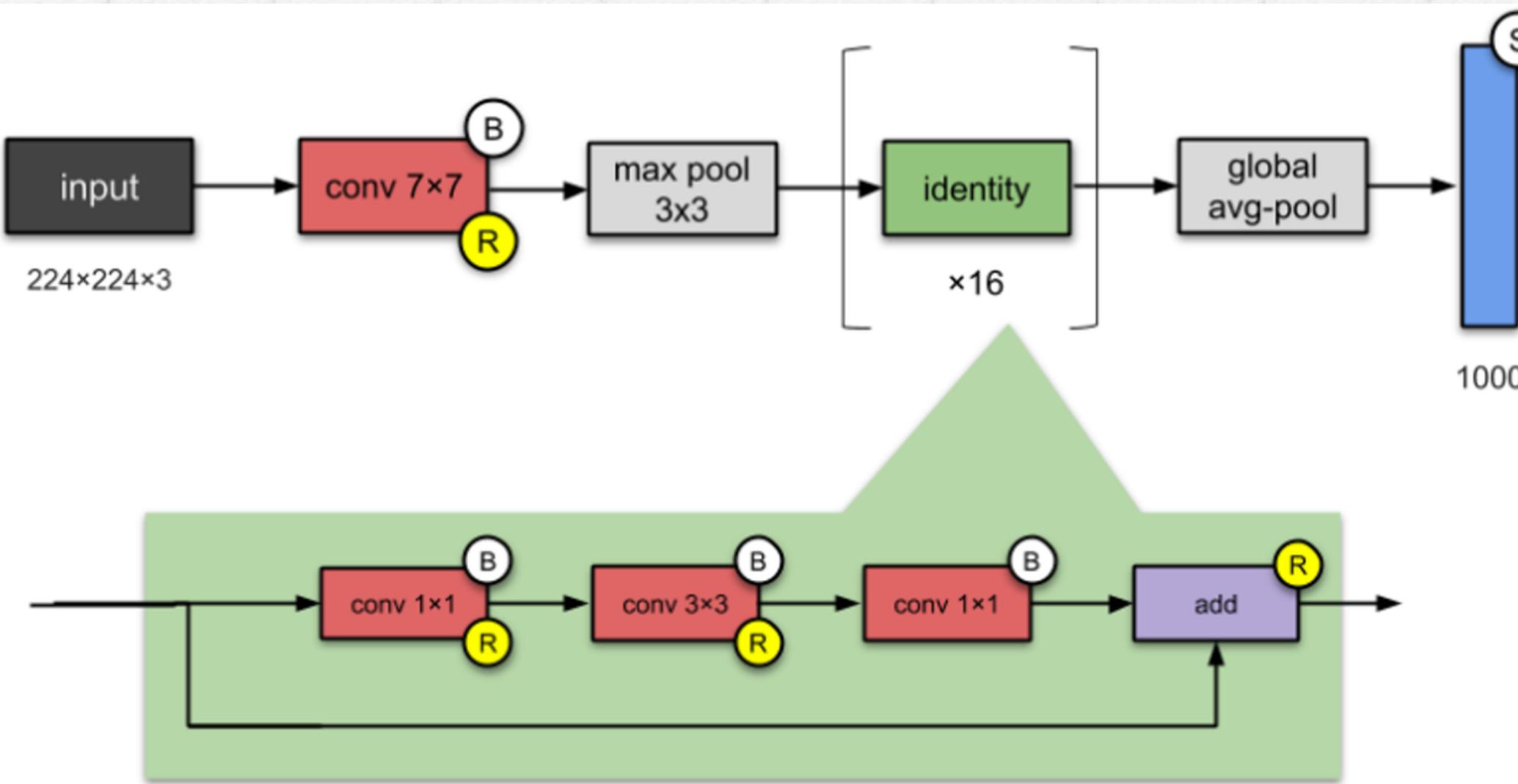
# ResNet

*Residual neural networks*

Giải quyết  
**Vanishing Gradient**

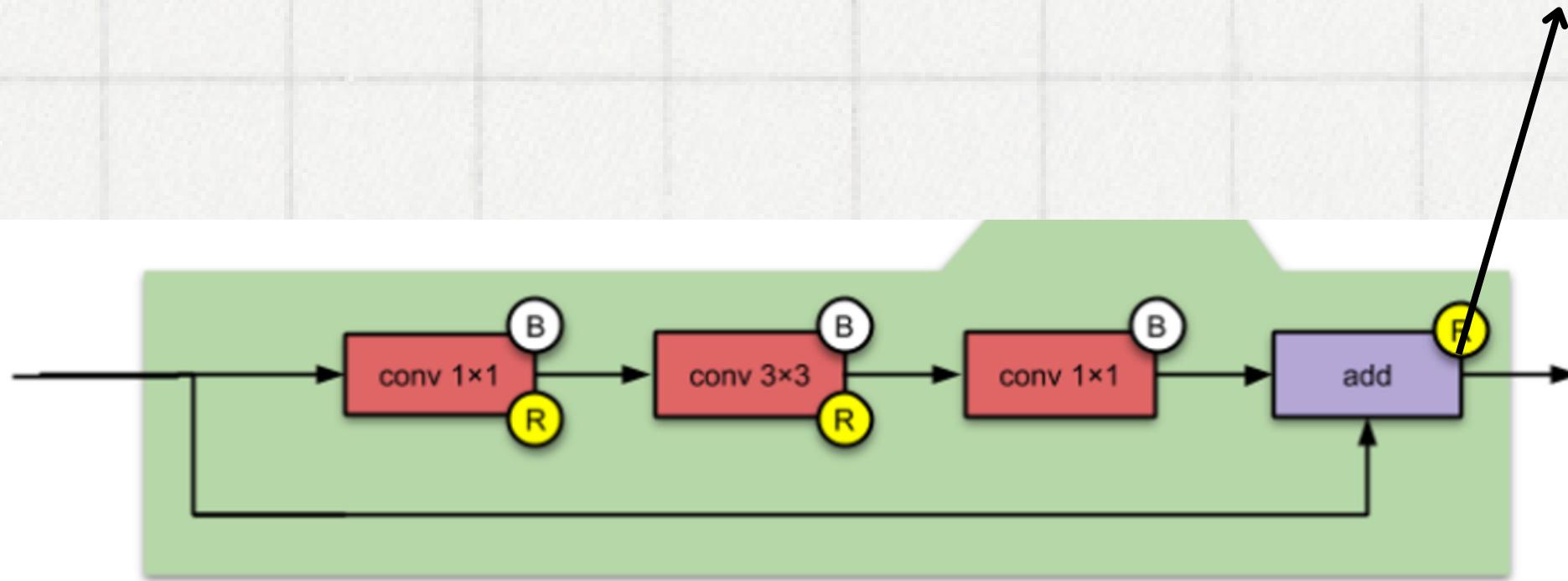


# Kiến trúc mạng ResNet



# Kiến trúc mạng ResNet

Skip connection



# **Thực hiện huấn luyện**

**Thank you  
very much!**

[www.reallygreatsite.com](http://www.reallygreatsite.com)