

COLE.VN
connecting knowledge

Chủ đề:
Giới thiệu về học sâu - Deep neural network

Mục đích buổi học

- Học viên tiếp cận được các khái niệm cơ bản trong học sâu
- Các vấn đề của học sâu

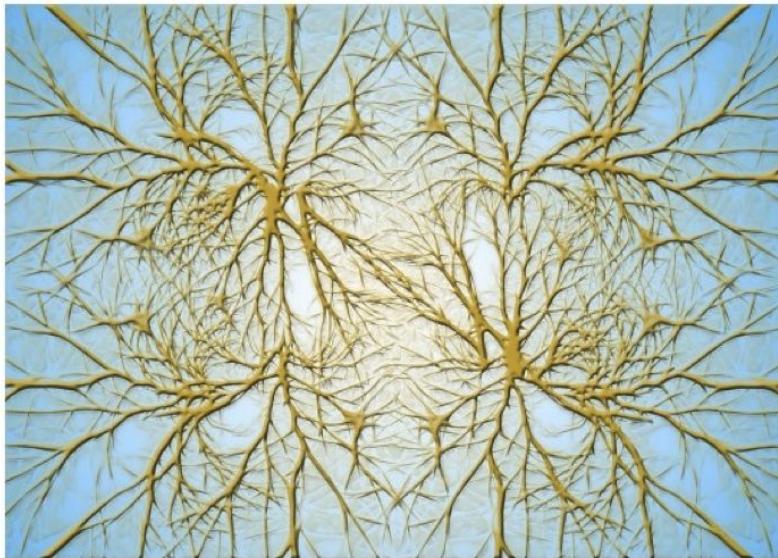
Nội dung chính

CNN

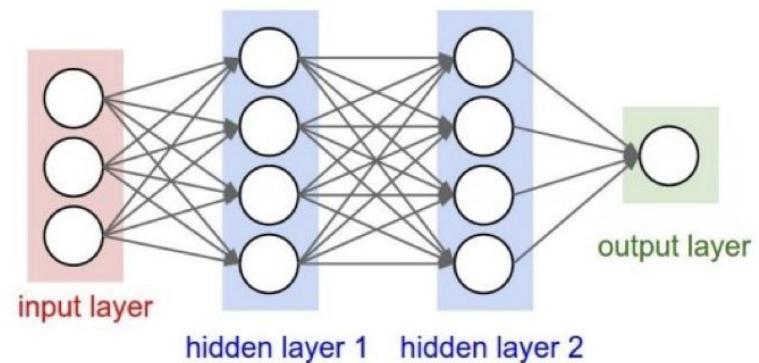
Huấn luyện mô hình mạng

Mạng nơ-ron - nhân tạo

Nơ-ron sinh học:
Kết nối phức tạp

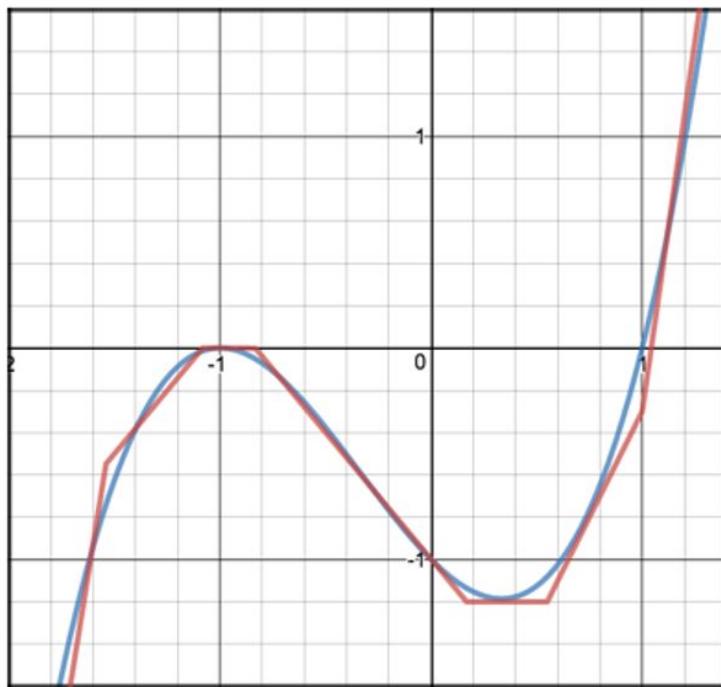


Mạng nơ-ron nhân tạo: Các nơ-ron tổ chức thành các lớp (layers) để tăng hiệu quả tính toán nhờ song song hóa



Định lý xấp xỉ tổng quát

Theorem (Universal Function Approximators). Một mạng nơ-ron từ hai lớp trở lên với số lượng nơron đủ lớn có thể xấp xỉ bất kỳ hàm liên tục nào với độ chính xác tùy ý



$$n_1(x) = \text{Relu}(-5x - 7.7)$$

$$n_2(x) = \text{Relu}(-1.2x - 1.3)$$

$$n_3(x) = \text{Relu}(1.2x + 1)$$

$$n_4(x) = \text{Relu}(1.2x - .2)$$

$$n_5(x) = \text{Relu}(2x - 1.1)$$

$$n_6(x) = \text{Relu}(5x - 5)$$

$$\begin{aligned} Z(x) = & -n_1(x) - n_2(x) - n_3(x) \\ & + n_4(x) + n_5(x) + n_6(x) \end{aligned}$$

Universal Function Approximation Theorem*

Hornik theorem 1: Whenever the activation function is *bounded and nonconstant*, then, for any finite measure μ , standard multilayer feedforward networks can approximate any function in $L^p(\mu)$ (the space of all functions on R^k such that $\int_{R^k} |f(x)|^p d\mu(x) < \infty$) arbitrarily well, provided that sufficiently many hidden units are available.

Hornik theorem 2: Whenever the activation function is *continuous, bounded and non-constant*, then, for arbitrary compact subsets $X \subseteq R^k$, standard multilayer feedforward networks can approximate any continuous function on X arbitrarily well with respect to uniform distance, provided that sufficiently many hidden units are available.

- In words: Given any continuous function $f(x)$, if a 2-layer neural network has enough hidden units, then there is a choice of weights that allow it to closely approximate $f(x)$.

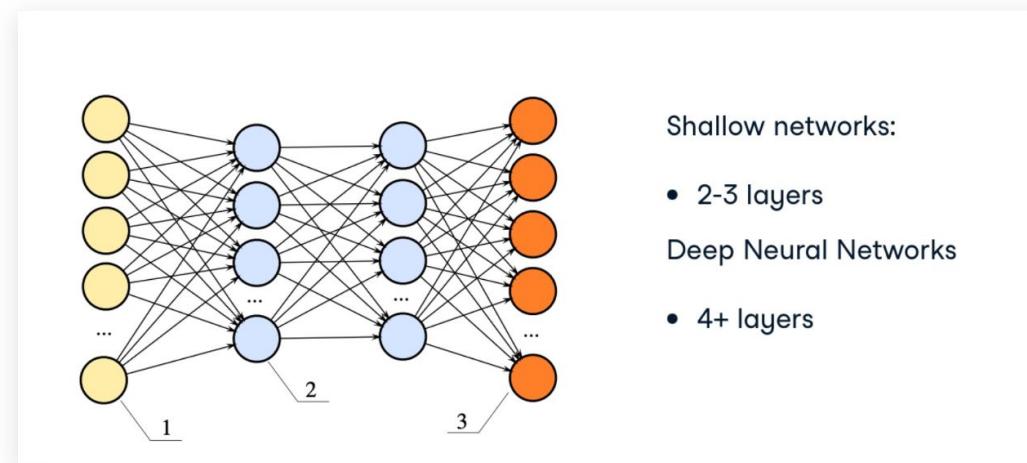
Cybenko, G. (1989). Approximations by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2, 183-192.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2), 251-257. Leshno, M., Lin, V. Y., Pinkus, A., & Schocken, S. (1993). Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks*, 6(6), 861-867.

Deep Neural Network

Deep Neural Network là gì?

- Ta xây dựng NN bằng cách xếp các lớp neural lên nhau
- Mạng nhỏ nhất: 1 lớp đầu vào, 1 lớp ẩn, 1 lớp đầu ra
 - Mạng nông (shallow networks): cho dù bao nhiêu node trong từng layer
- Nếu nhiều layer hơn → coi như ta đã đang xây dựng deep NN



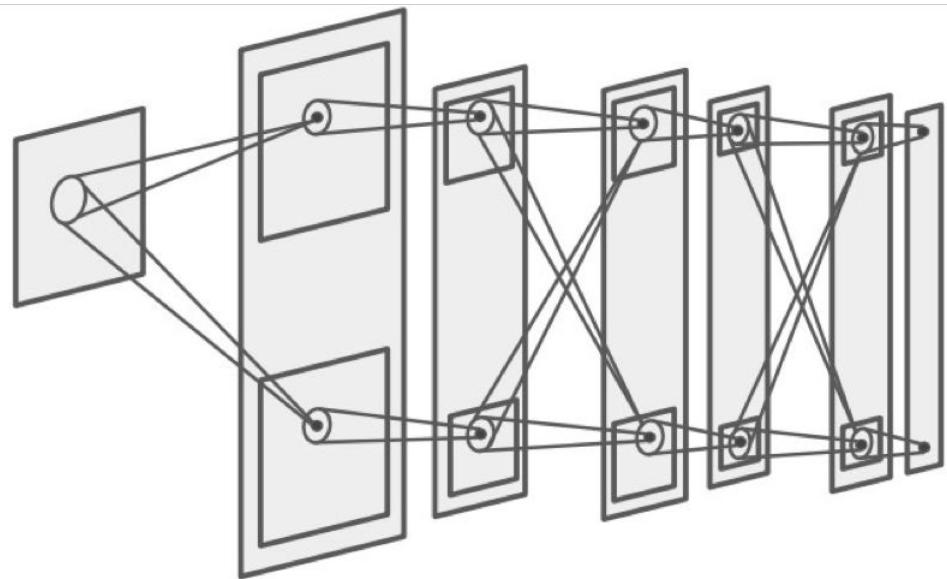
Tại sao cần mạng nhiều lớp?

- Mạng nơ-ron nhiều lớp (thậm chí chỉ cần duy nhất một lớp ẩn!) là hàm xấp xỉ tổng quát
- Mạng nơ-ron có thể biểu diễn hàm bất kỳ nếu nó đủ rộng (số nơ-ron trong một lớp đủ nhiều), đủ sâu (số lớp đủ lớn).
- Nếu muốn giảm độ sâu của mạng trong nhiều trường hợp sẽ phải bù lại bằng cách tăng chiều rộng lên lũy thừa lần!
- Mạng nơ-ron một lớp ẩn có thể cần tới số lượng nơ-ron cao gấp lũy thừa lần so với một mạng nhiều tầng
- Mạng nhiều lớp cần số lượng nơ-ron ít hơn rất nhiều so với các mạng nông (shallow networks) để cùng biểu diễn một hàm số giống nhau
→ Mạng nhiều lớp giá trị hơn

Convolution Neural Network - Lịch sử

Neocognitron [Fukushima 1980]

“sandwich” architecture (SCSCSC...)
simple cells: modifiable parameters
complex cells: perform pooling

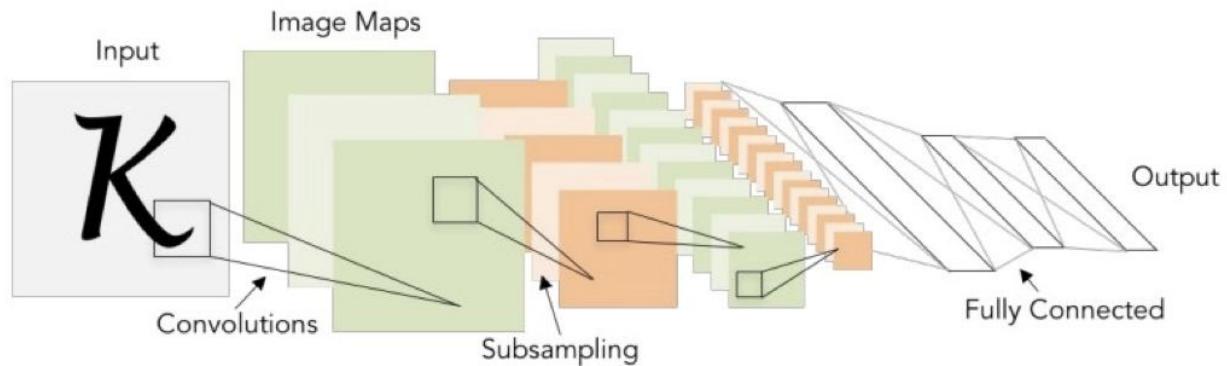


Ý tưởng CNNs xuất phát đầu tiên từ công trình của Fukushima năm 1980

Convolution Neural Network - Lịch sử

Gradient-based learning applied to document recognition

[LeCun, Bottou, Bengio, Haffner 1998]



Năm 1998, LeCun áp dụng BackProp huấn luyện mạng CNNs cho bài toán nhận dạng văn bản

Convolution Neural Network - Lịch sử

ImageNet Classification with Deep Convolutional Neural Networks [Krizhevsky, Sutskever, Hinton, 2012]

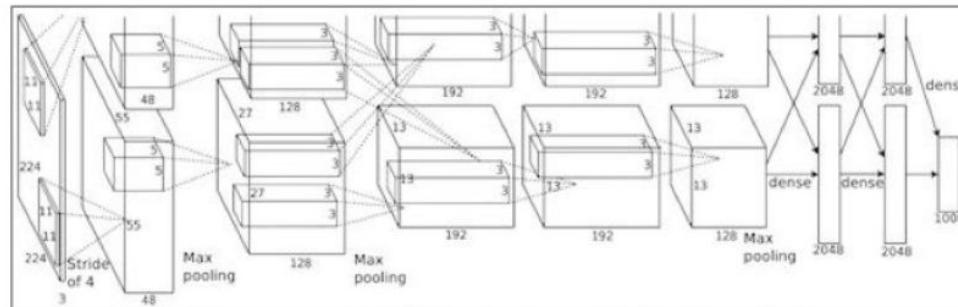
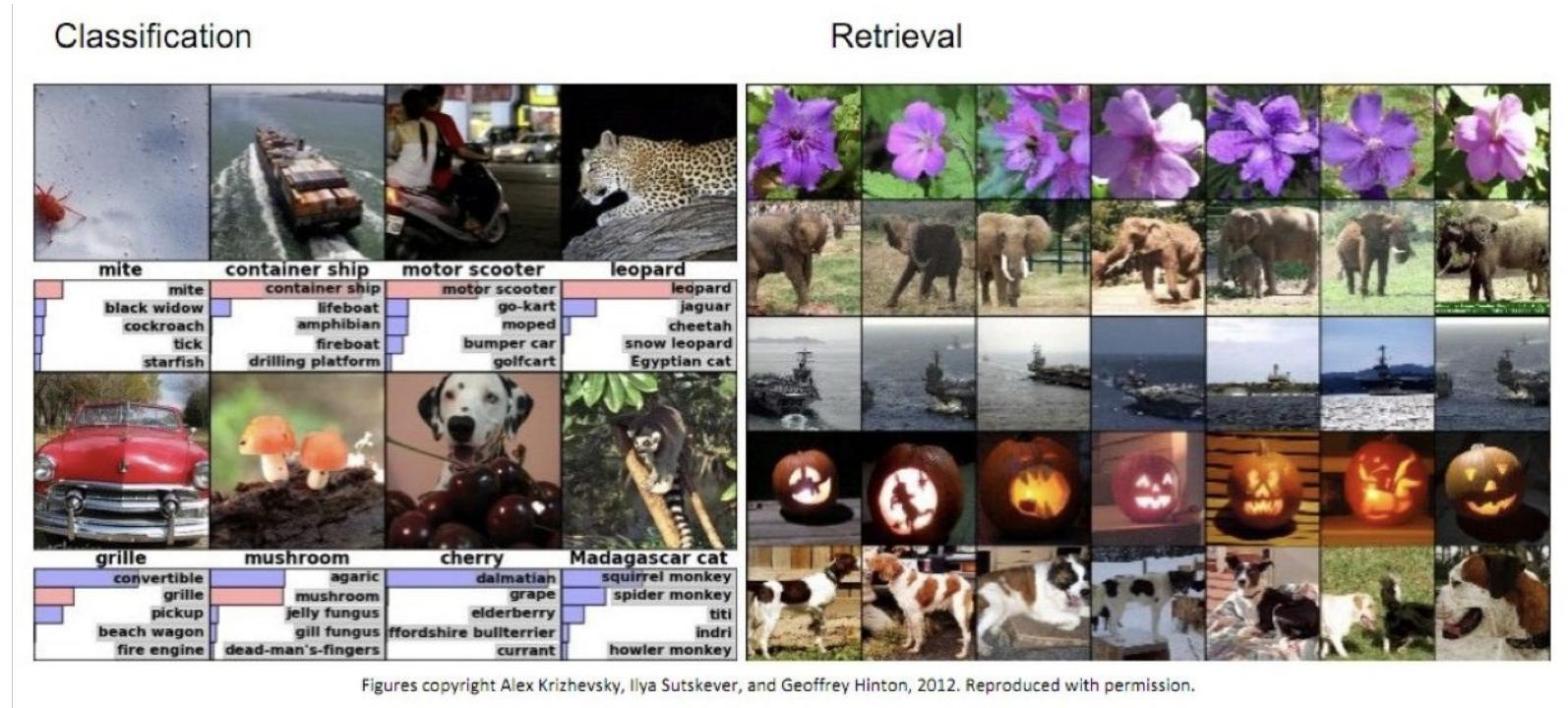


Figure copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

“AlexNet”

Năm 2012, CNNs gây tiếng vang lớn khi vô địch cuộc thi ILSRC 2012, vượt xa phương pháp đứng thứ 2 theo cách tiếp cận thị giác máy tính truyền thống.

Convolution Neural Network - Lịch sử

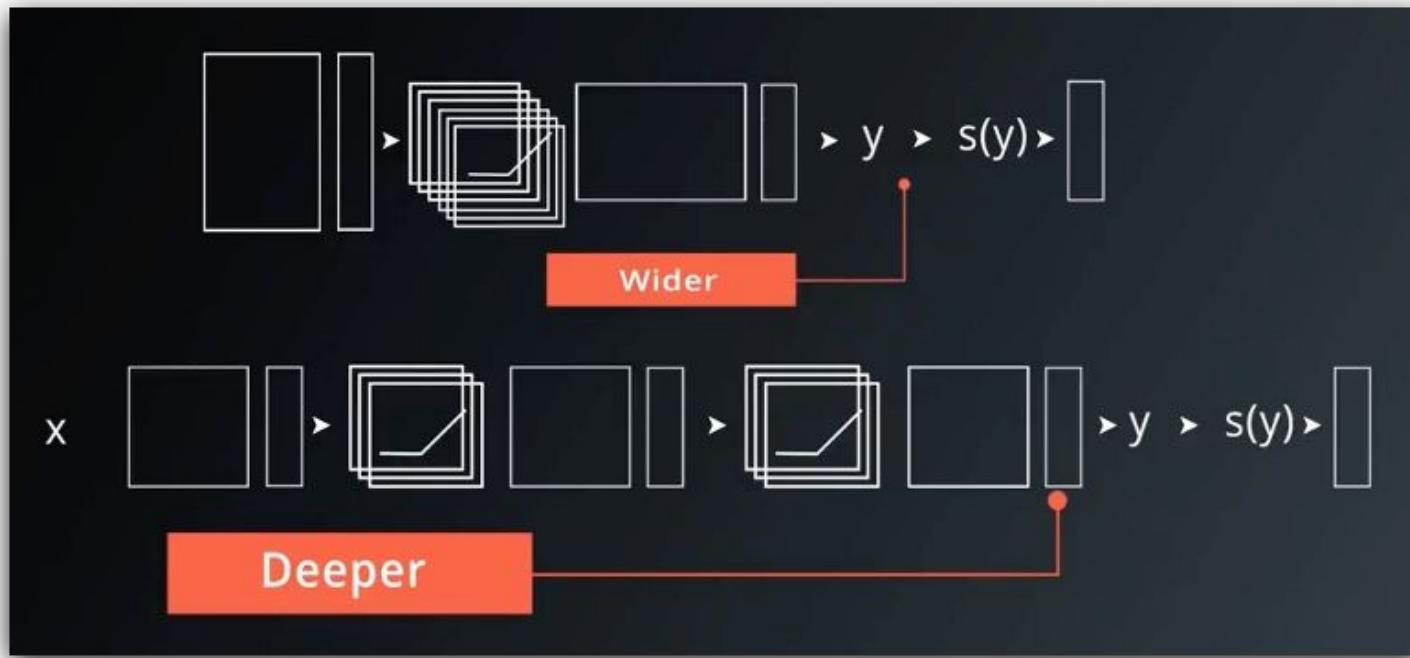


Figures copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

Hiện nay CNNs ứng dụng khắp nơi, ví dụ trong bài toán phân loại ảnh, truy vấn ảnh

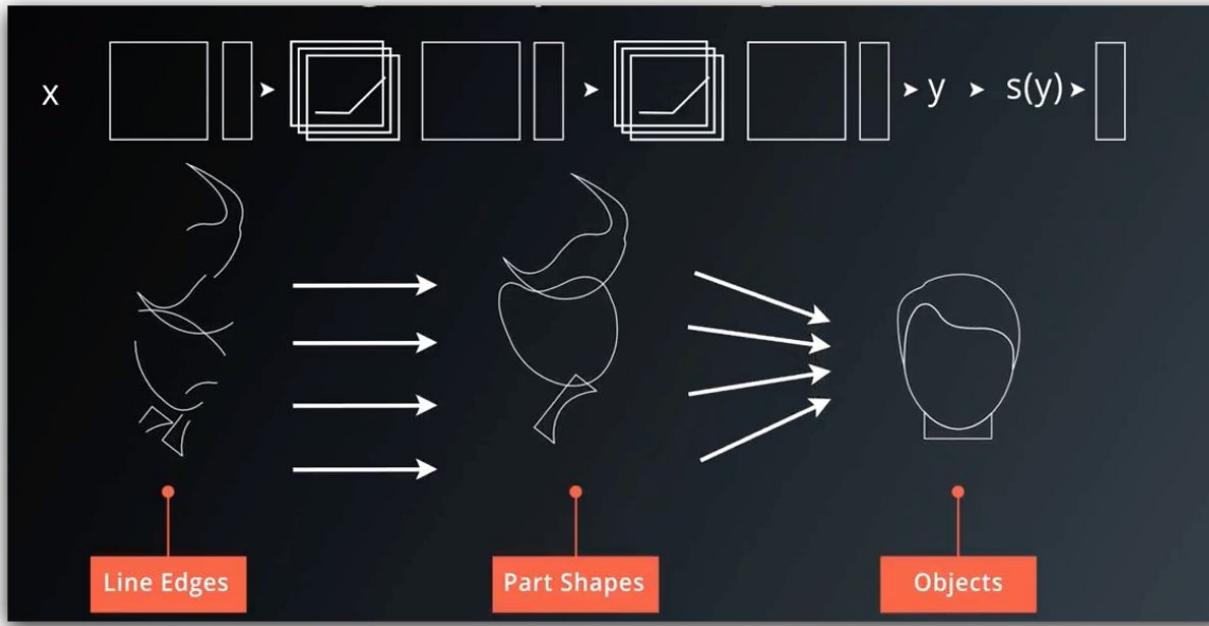
Convolution Neural Network - Ý tưởng

- Cải tiến NN: deeper tốt hơn wider
- Super wider NN có thể memorize data chứ không thể generalize data



Convolution Neural Network - Ý tưởng

Model: học dễ dàng hơn từ abstract đến cụ thể → tổng quát hoá data tốt hơn (học cả các đặc trưng trung gian)



Convolution Neural Network - Ý tưởng

Vấn đề MLP

- Sử dụng quá nhiều parameters: chỉ gồm fully connected layers
- Bỏ hết các thông tin về không gian: chỉ nhận đầu vào là vector
 - Phải chuyển ma trận sang vector
 - Khó khăn trong việc nhận dạng, không biết được vị trí tương quan của các pixel

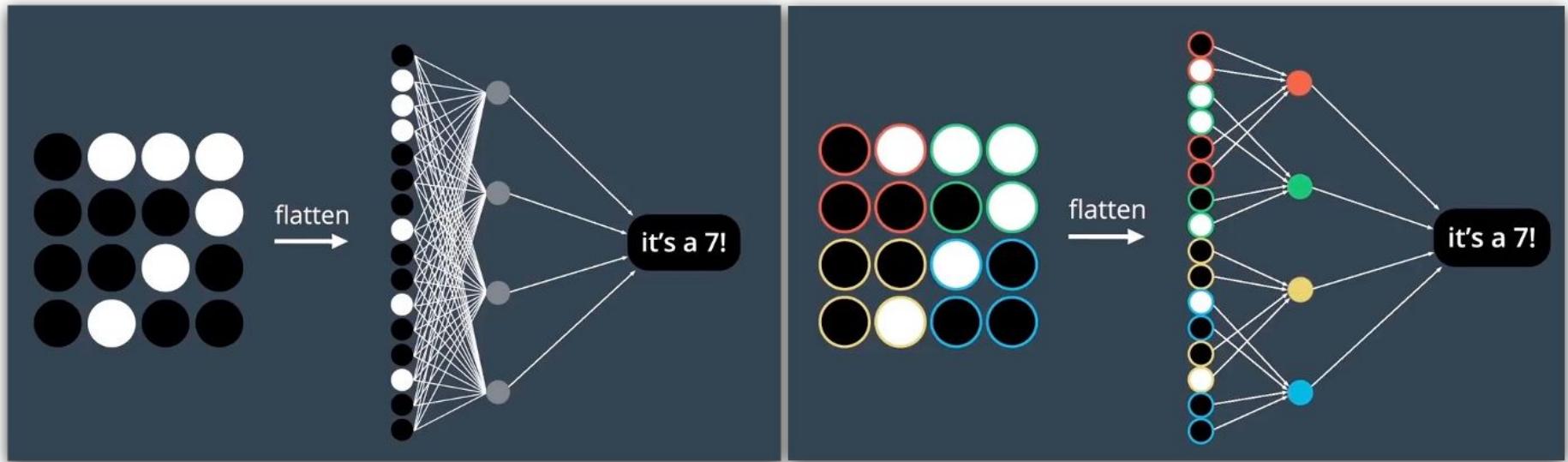
CNN

- Sparsely connected layers
- Nhận đầu vào là ma trận

Convolution Neural Network - Ý tưởng

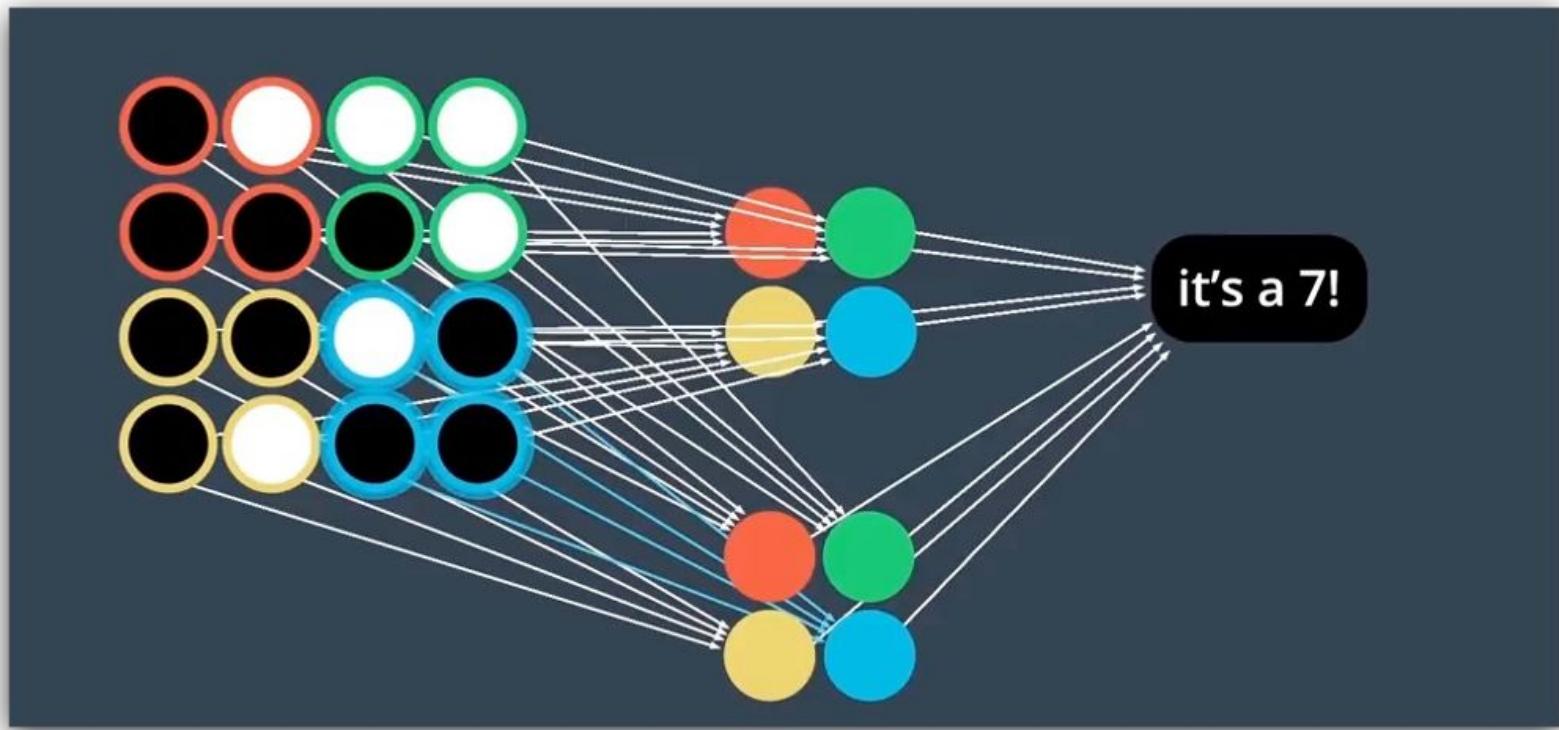
Nhận dạng số 7:

- Không dùng fully connected layer mà chia ra từng vùng, mỗi hidden node chỉ quan tâm đến một vùng ảnh mà thôi
- Giá trị các hidden node sẽ được tổng hợp sau



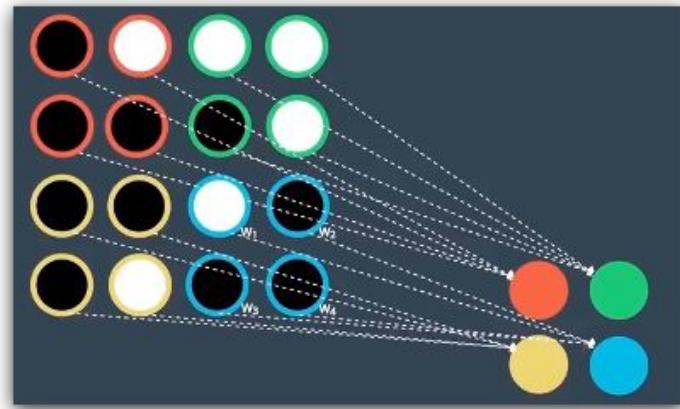
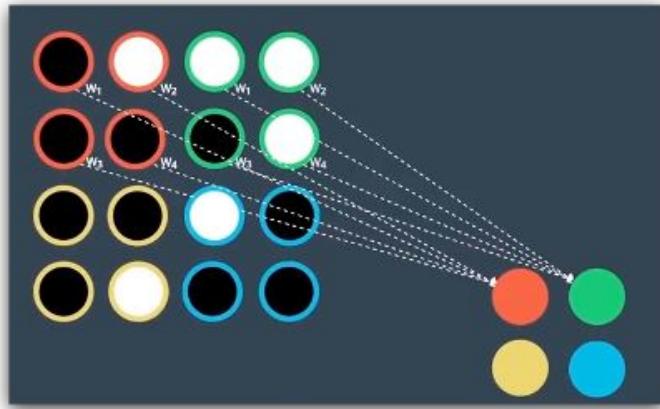
Convolution Neural Network - Ý tưởng

- Giữ được thông tin không gian
- Có thể tăng số lượng hidden nodes

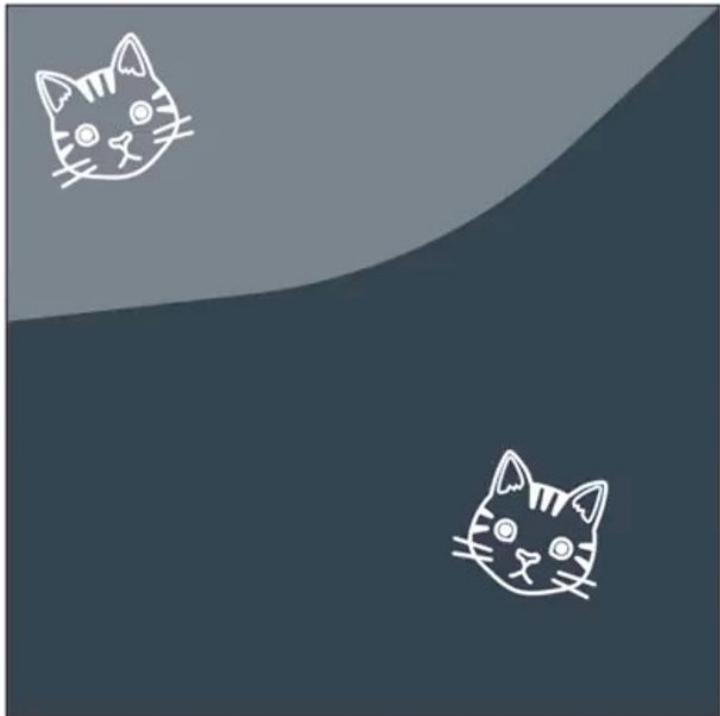


Convolution Neural Network - Ý tưởng

- Shared weight: các nút share chung weight
 - Các vùng khác nhau trong ảnh chia sẻ thông tin chung
 - Nói cách khác: các mẫu cần tìm trong ảnh có thể xuất hiện ở bất kỳ vị trí nào



Convolution Neural Network - Ý tưởng



Cat

Dùng weight sharing để nhận dạng mèo ở các vị trí khác nhau

Lớp tích chập

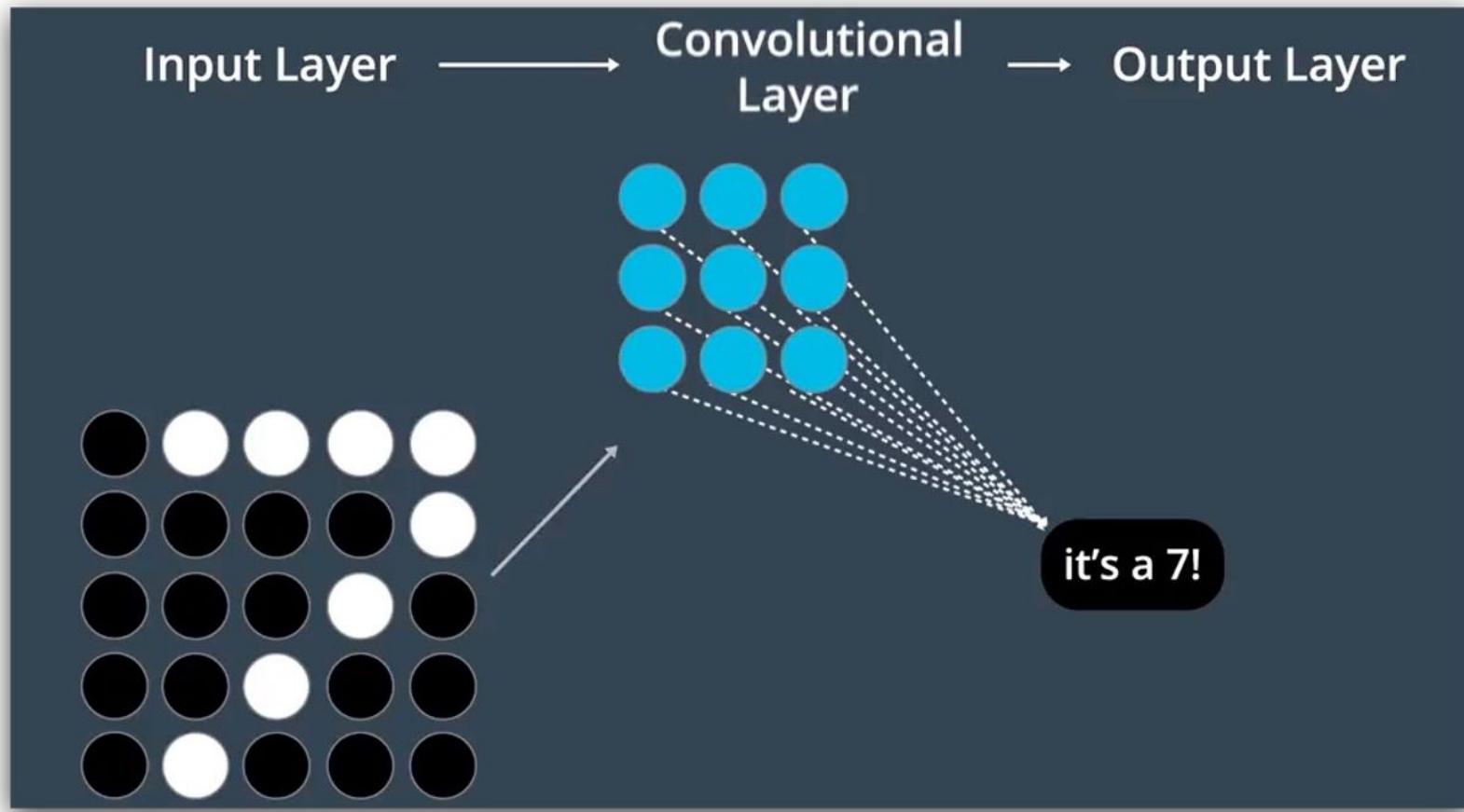
Traditional computer vision:

- Công việc vất vả vì đòi hỏi rất nhiều thời gian để tinh chỉnh thuật toán trích xuất đặc trưng (feature extraction) và tiền xử lý (preprocessing) với các tiếp cận khác nhau

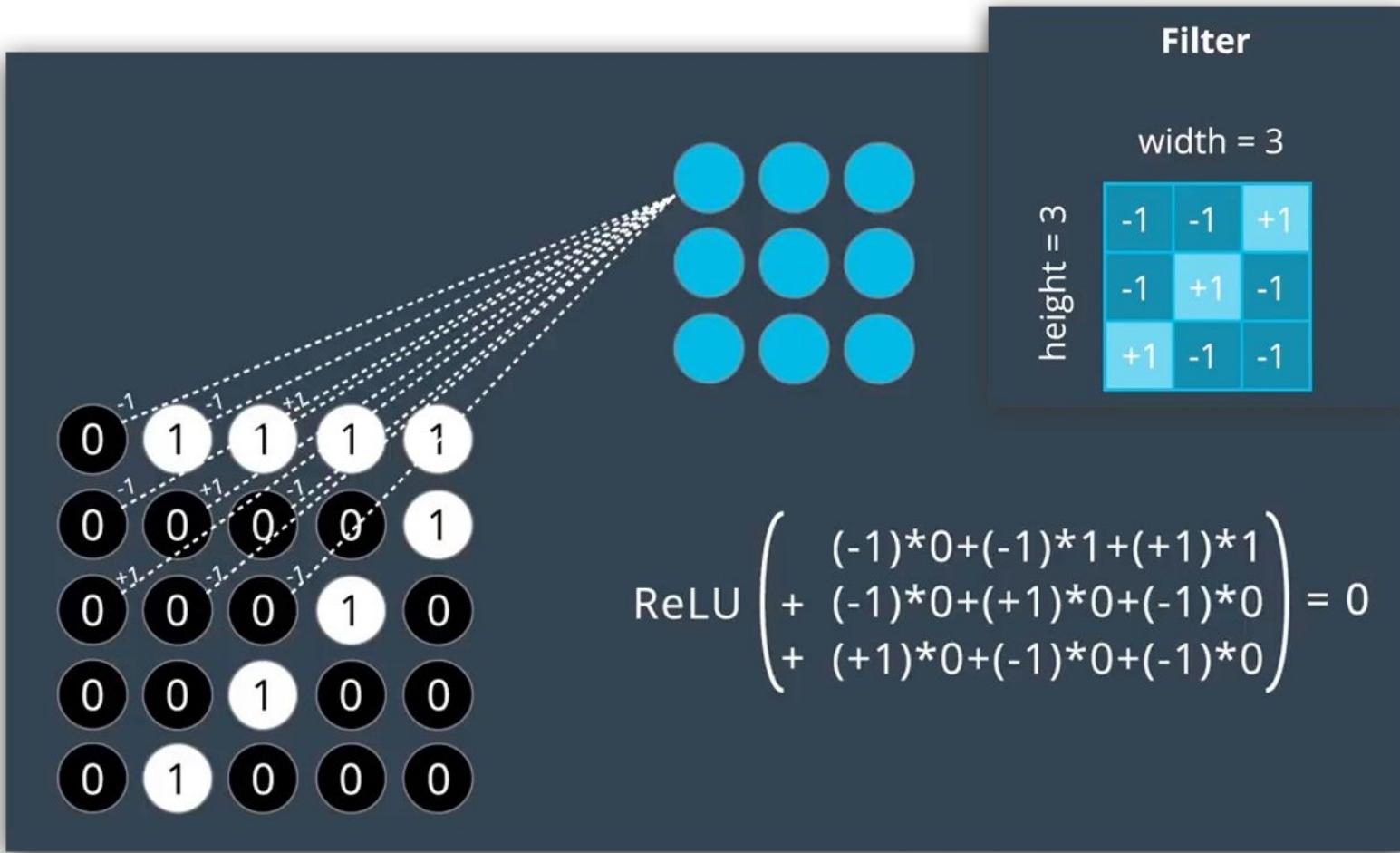
CV với Deep learning

- Không cần làm như truyền thống
- Thu thập (rất nhiều) bức ảnh đã được gán nhãn và cho thuật toán từ tìm các kernel phù hợp thông qua quá trình học
- Kernel ~ Trích chọn đặc trưng
- Nhược điểm: cần rất nhiều dữ liệu

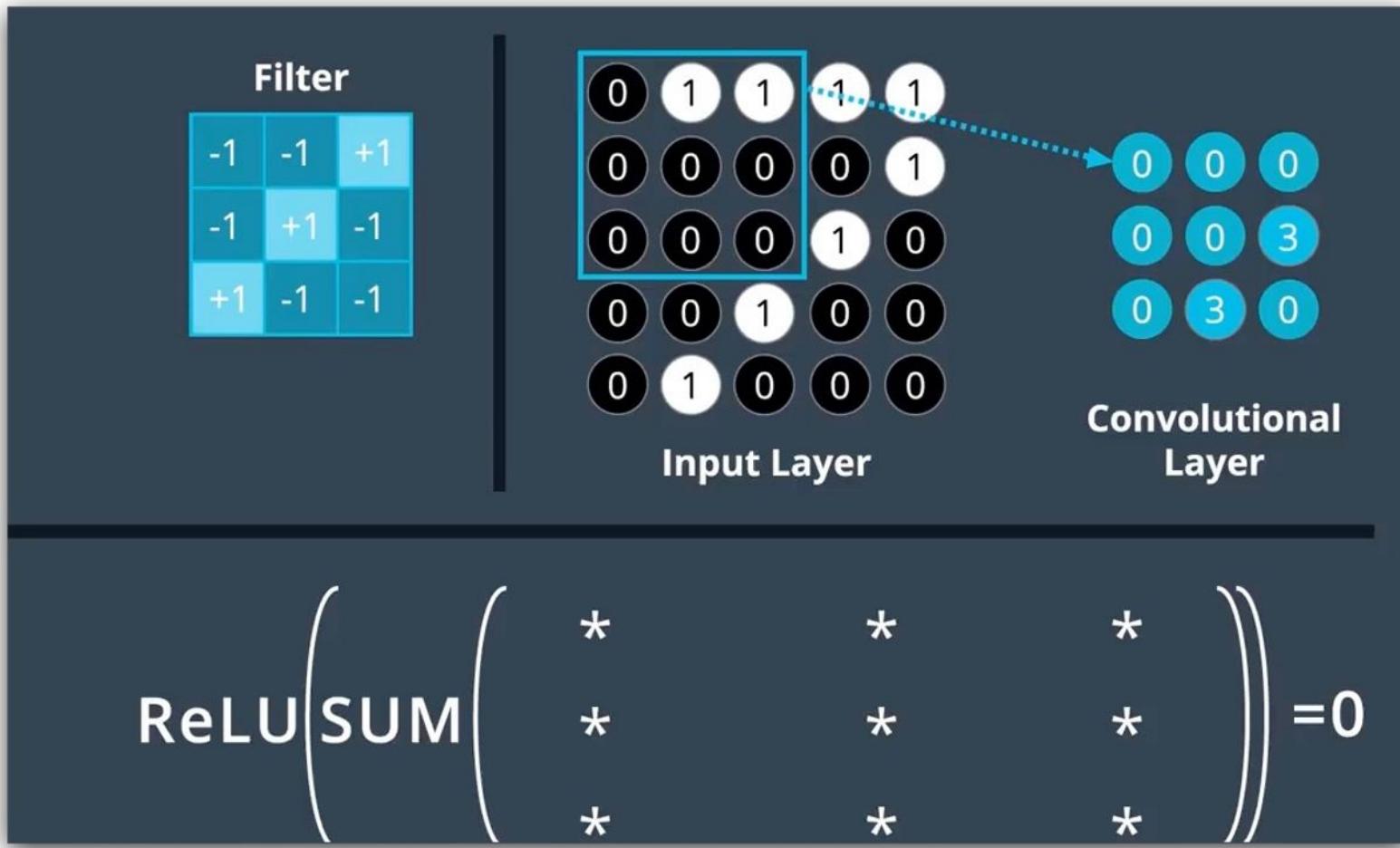
Lớp tích chập



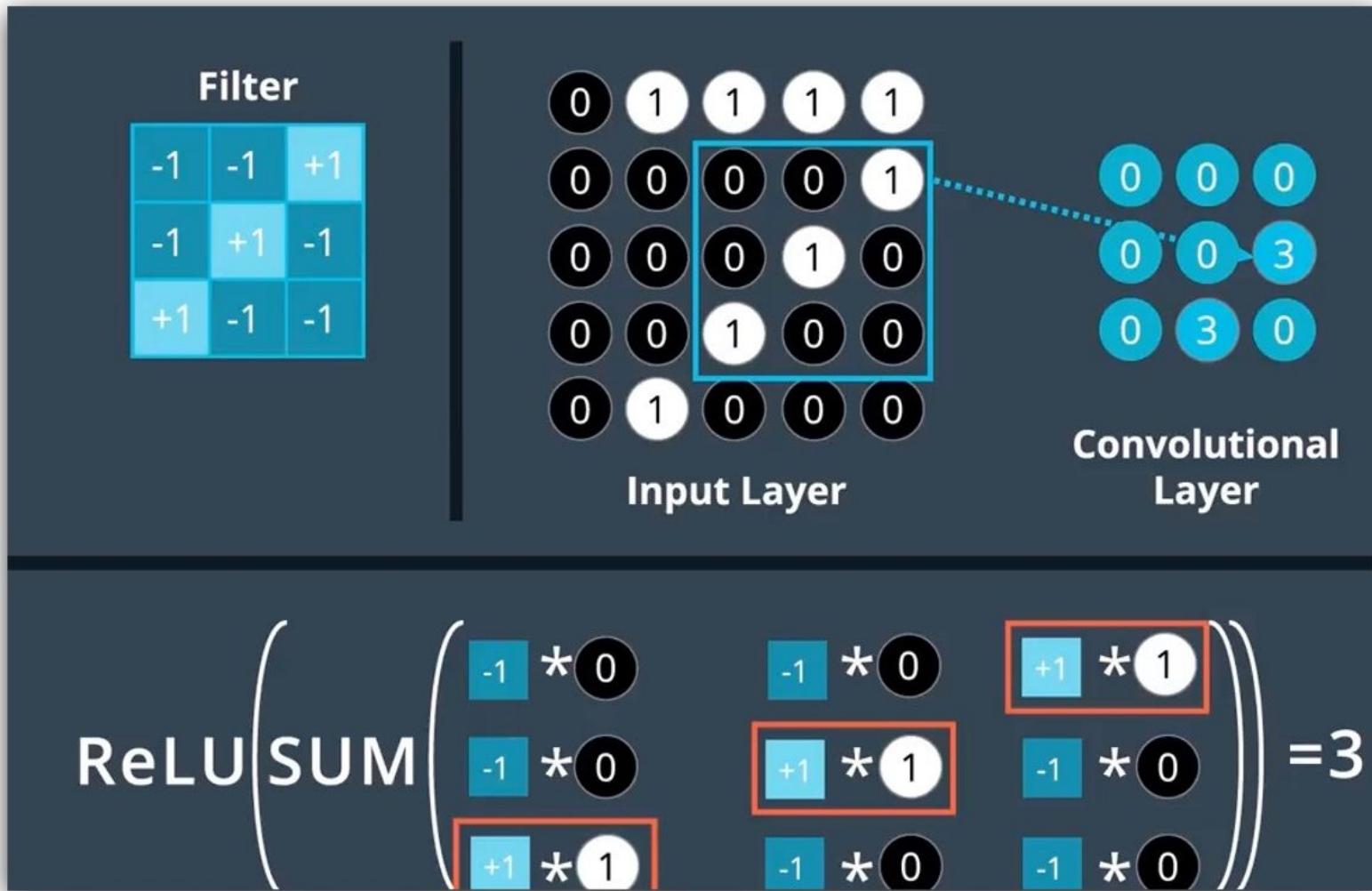
Lớp tích chập



Lớp tích chập

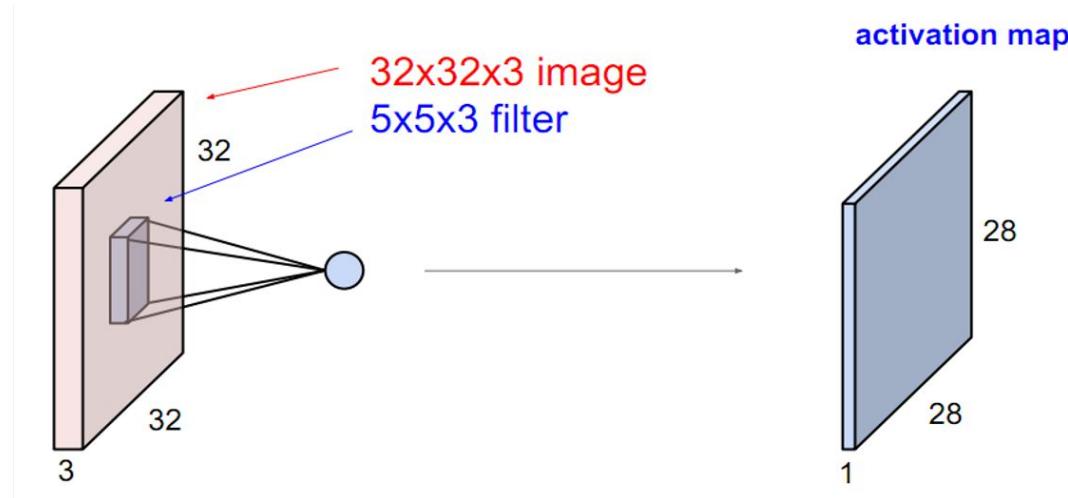


Lớp tích chập



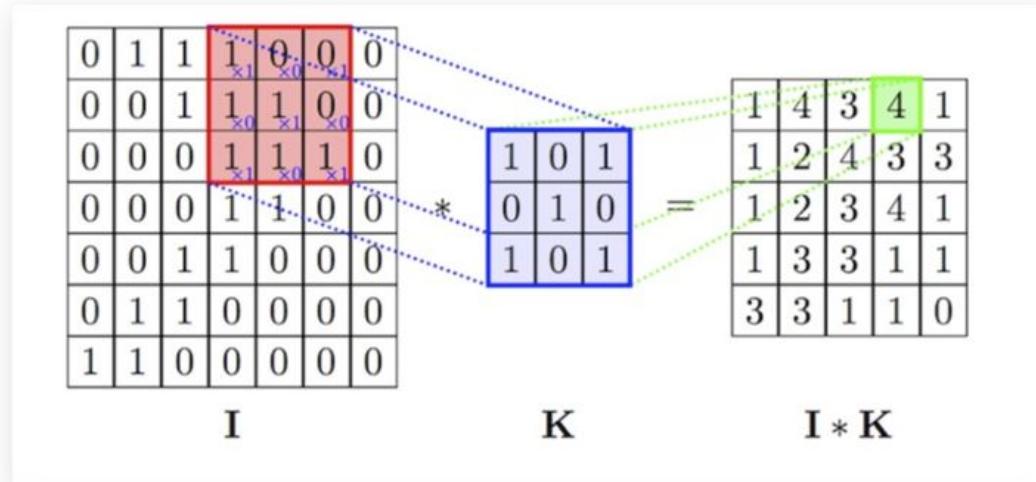
Lớp tích chập

- Khác với nơ-ron kết nối đầy đủ, mỗi nơ-ron tích chập (filter) chỉ kết nối cục bộ với dữ liệu đầu vào
- Nơ-ron tích chập trượt từ trái sang phải và từ trên xuống dưới khói dữ liệu đầu vào và tính toán để sinh ra một bản đồ kích hoạt (activation map)
- Chiều sâu của nơ-ron tích chập bằng chiều sâu của khối dữ liệu đầu vào



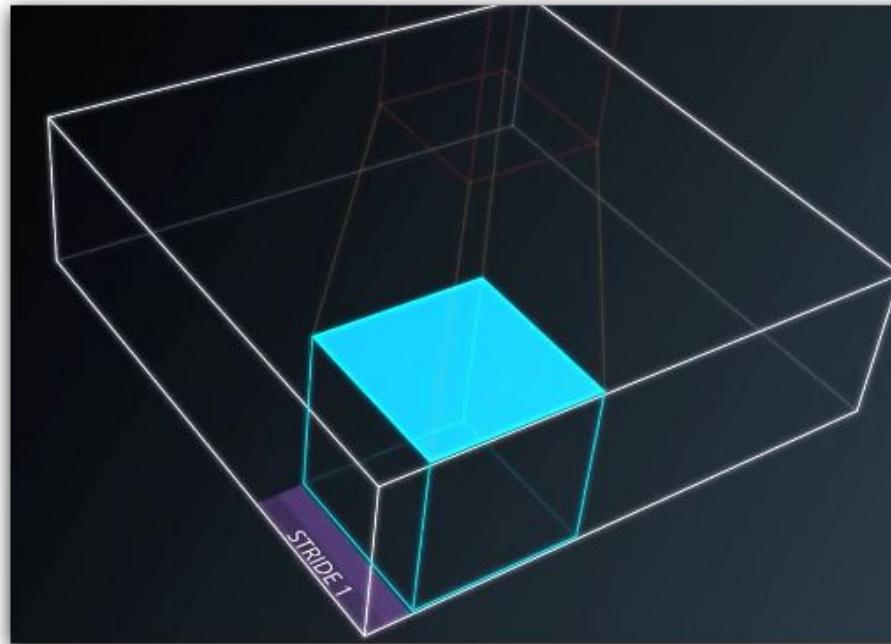
Lớp tích chập

- Tích chập: áp dụng một số bộ lọc nhất định cho tín hiệu vào, nhằm trích xuất các tính năng cụ thể từ nó
- Những “bộ lọc” này còn được gọi là “hạt nhân-Kernel”
 - có thể đơn giản như một ma trận 3×3 gồm các số không và số một
- Kernel hoạt động như một loại thấu kính mà qua đó hình ảnh gốc được truyền qua



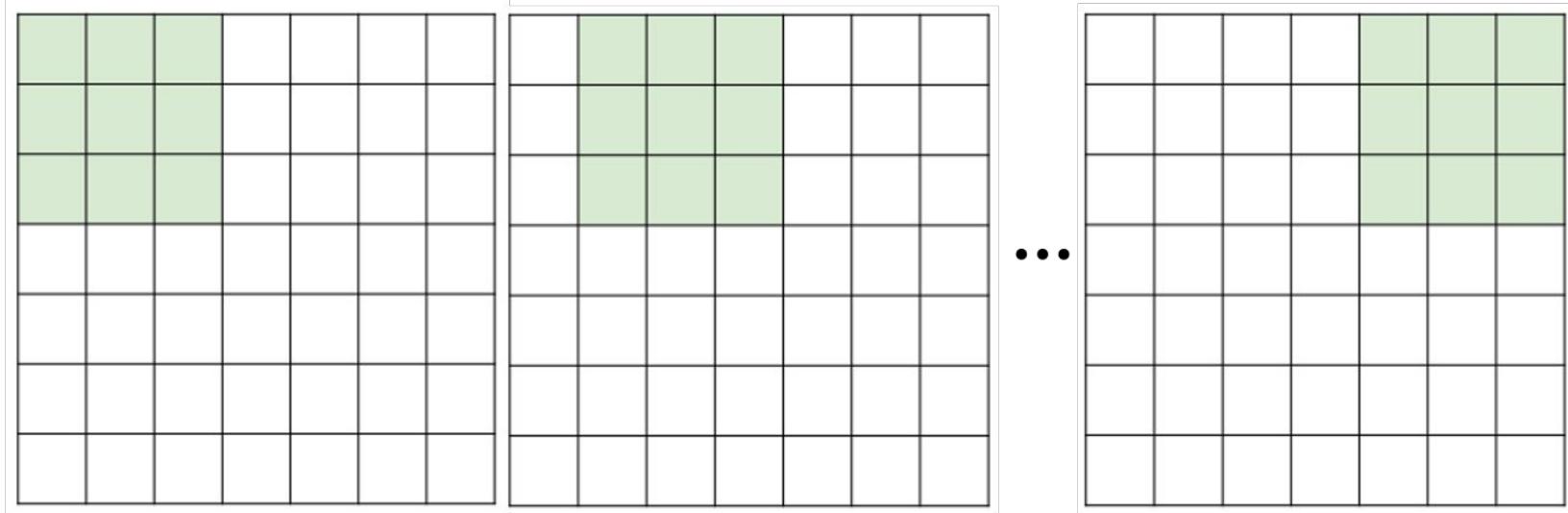
Lớp tích chập - Stride

- Stride: số lượng pixel dịch chuyển mỗi lần của kernel
 - Stride 1: output cùng kích cỡ với input
 - Stride 2: output có kích cỡ giảm khoảng một nửa (phụ thuộc vào xử lý biên ảnh)



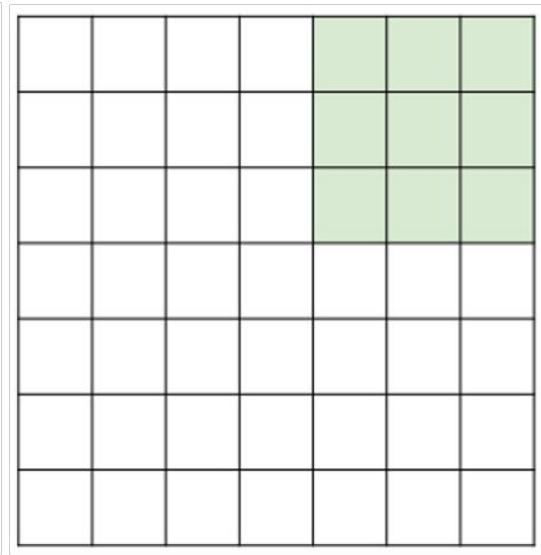
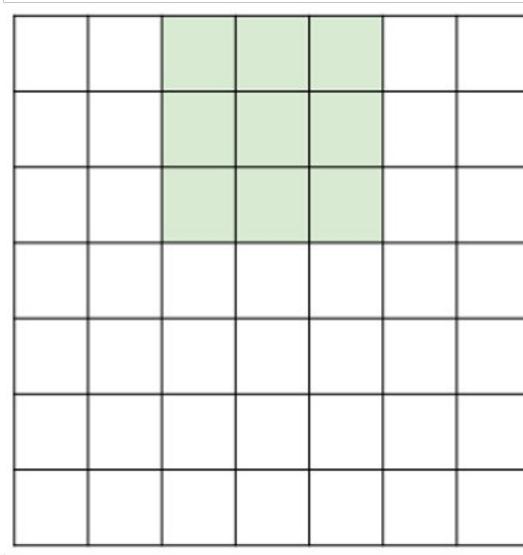
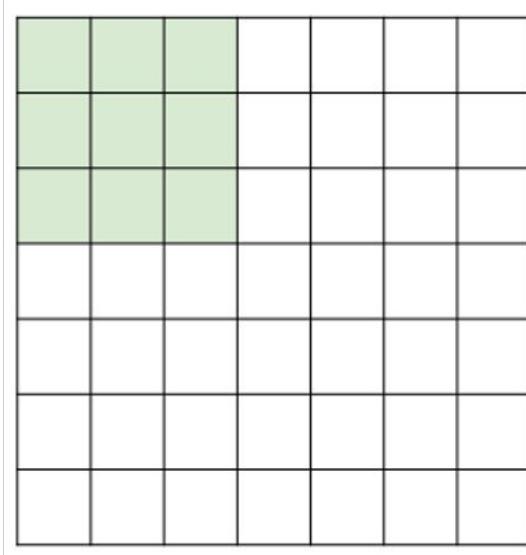
Lớp tích chập

- Bước nhảy stride = 1
- Đầu vào kích thước 7×7 , nơ-ron kích thước 3×3
- Đầu ra kích thước 5×5



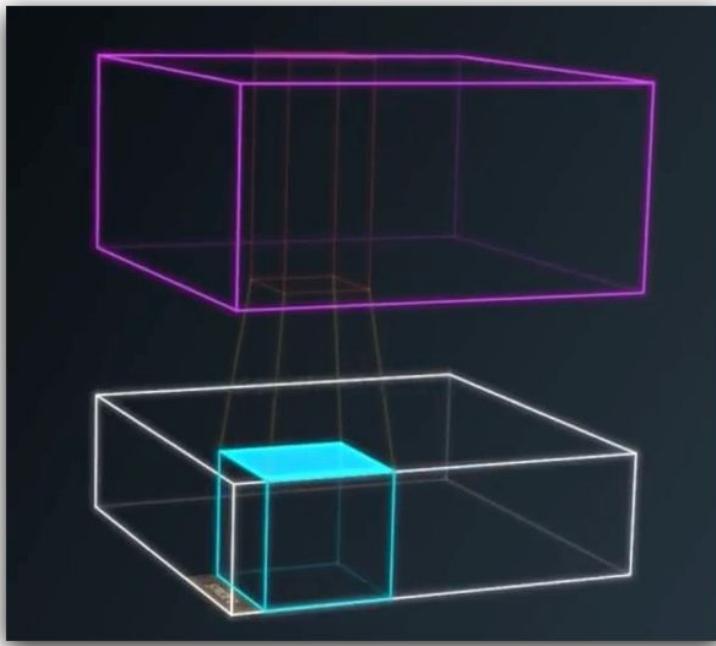
Lớp tích chập

- Bước nhảy stride = 2
- Đầu vào kích thước 7×7 , nơ-ron kích thước 3×3
- Đầu ra kích thước 3×3

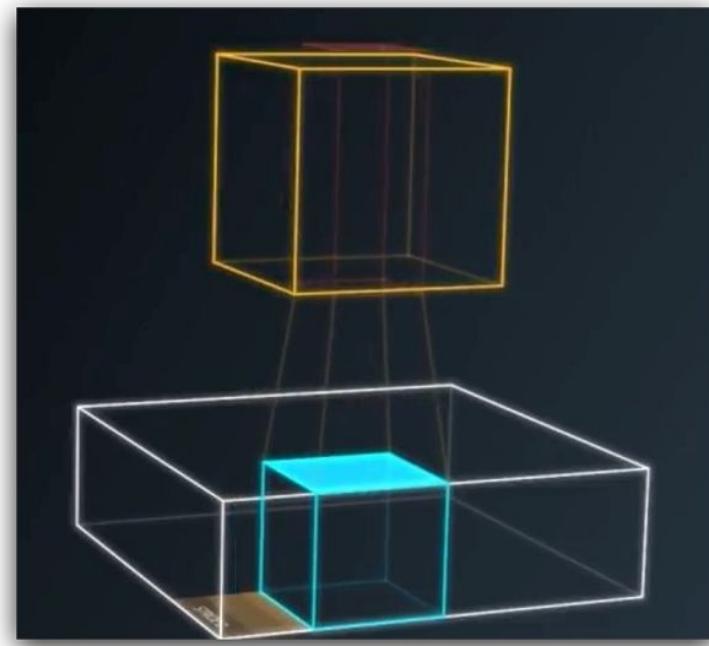


Lớp tích chập

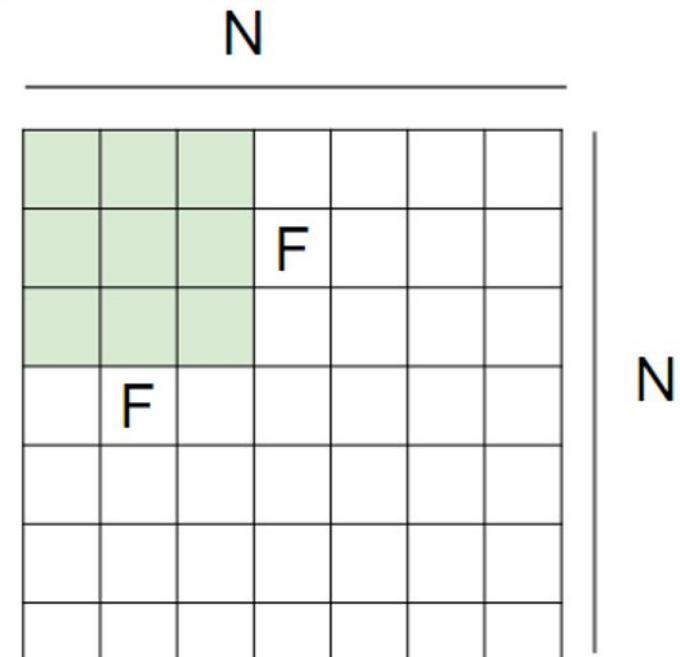
Stride 1



Stride 2



Lớp tích chập

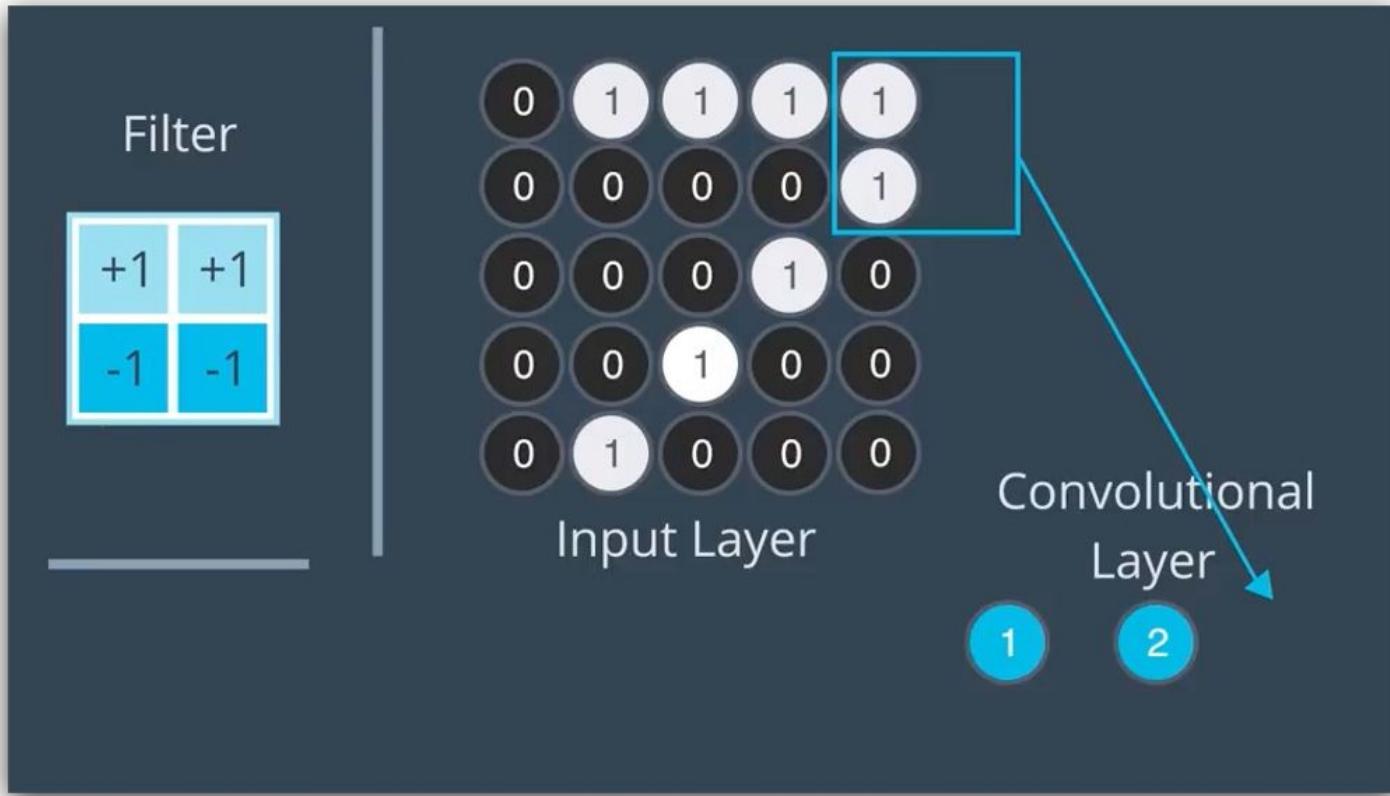


Output size:
 $(N - F) / \text{stride} + 1$

e.g. $N = 7$, $F = 3$:
stride 1 $\Rightarrow (7 - 3)/1 + 1 = 5$
stride 2 $\Rightarrow (7 - 3)/2 + 1 = 3$
stride 3 $\Rightarrow (7 - 3)/3 + 1 = 2.33 : \backslash$

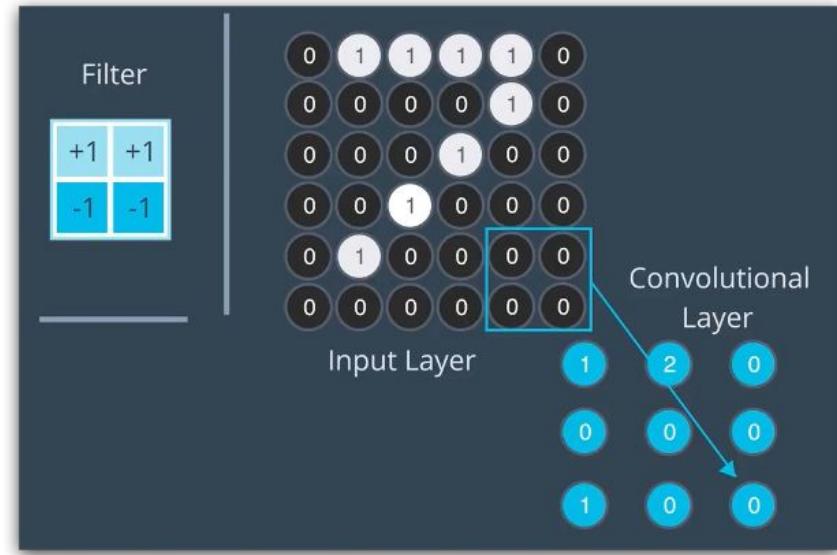
Lớp tích chập

- Giả sử stride 2 làm thế nào nếu không đủ node?



Lớp tích chập

- Same padding: chèn số 0 Cùng kích thước



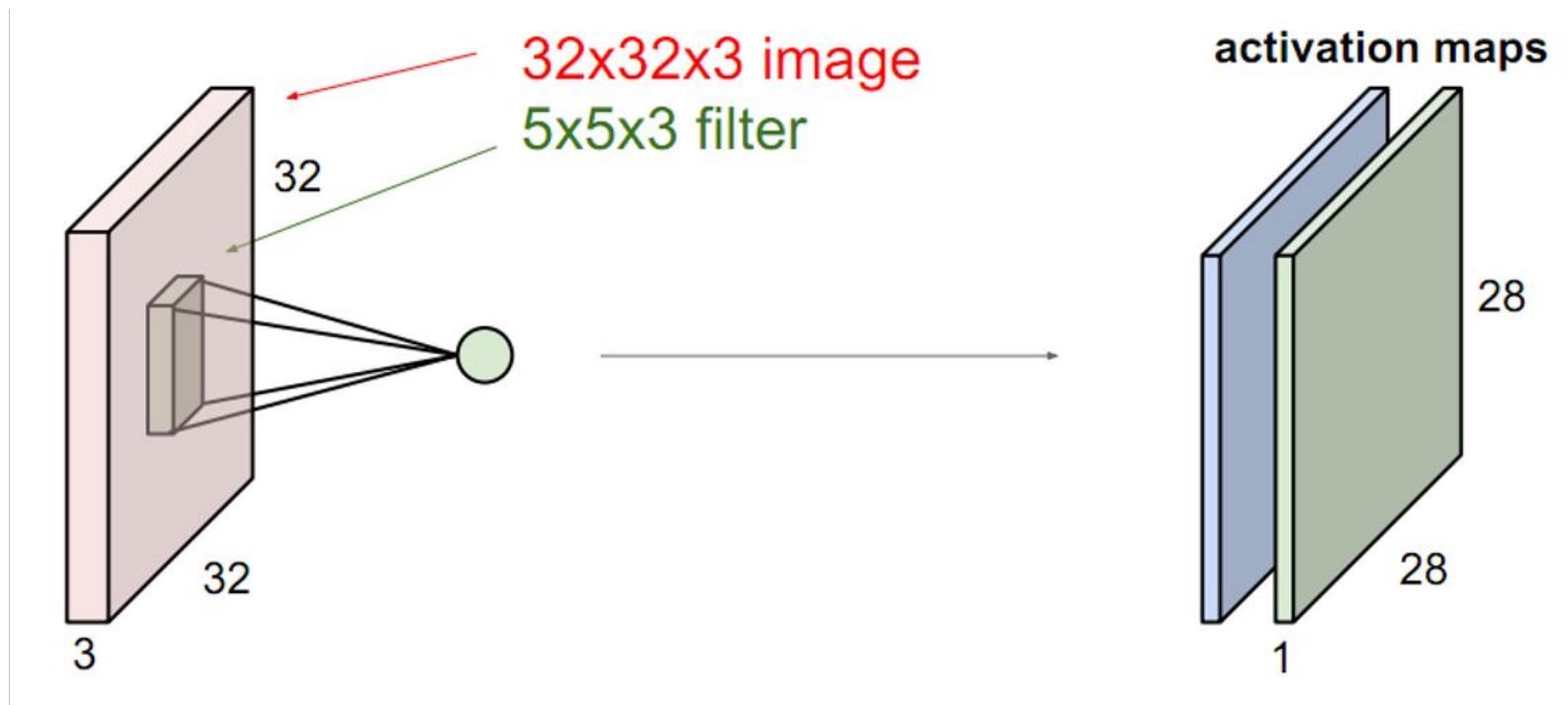
Lớp tích chập

- Để bảo toàn kích thước thường thêm viền bởi các số 0 (zero padding).
- Ví dụ: đầu vào kích thước 7×7 , nơron kích thước 3×3 , bước nhảy stride 1, padding viền độ rộng 1.
- Khi đó kích thước đầu ra là 7×7

| | | | | | | | | | |
|---|---|---|---|---|---|---|--|--|--|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| 0 | | | | | | | | | |
| 0 | | | | | | | | | |
| 0 | | | | | | | | | |
| 0 | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |
| | | | | | | | | | |

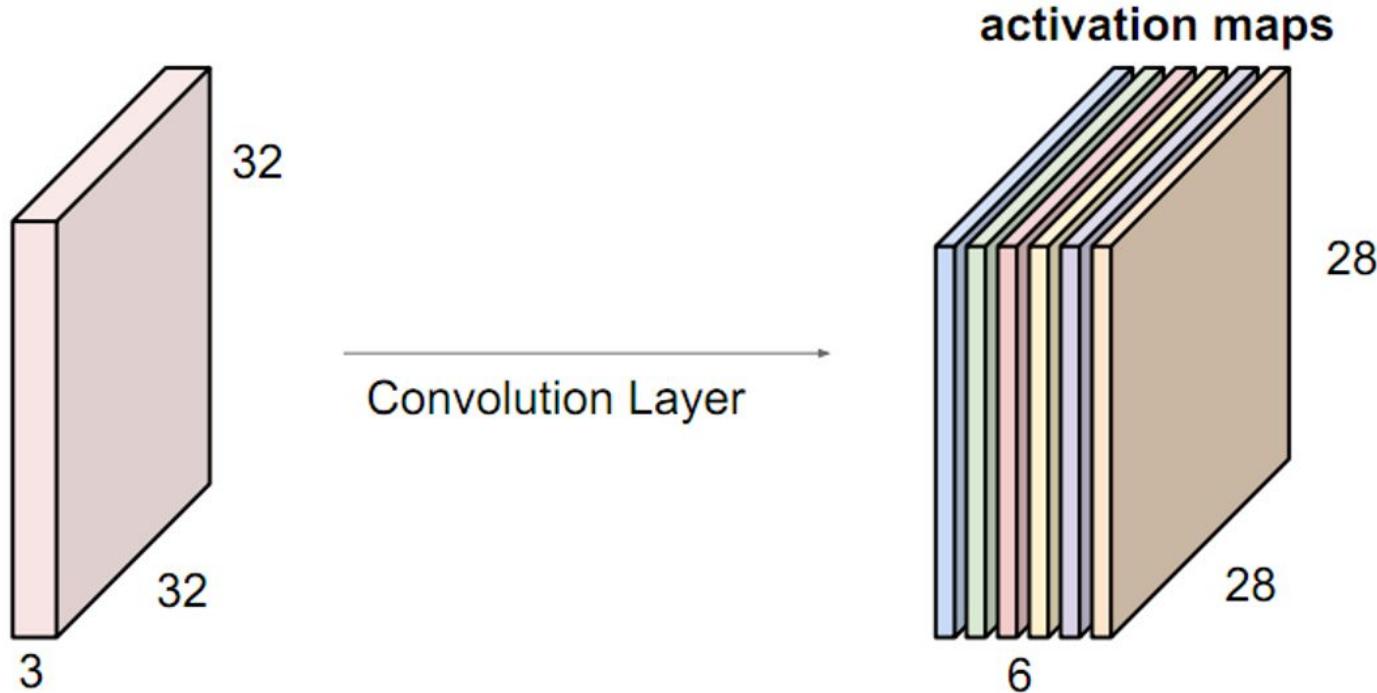
Lớp tích chập

- Giả sử có thêm nơ-ron tích chập khác thì nó cũng hoạt động tương tự và sinh ra bản đồ kích hoạt thứ hai
- Lưu ý trọng số của các nơ-ron tích chập là khác nhau



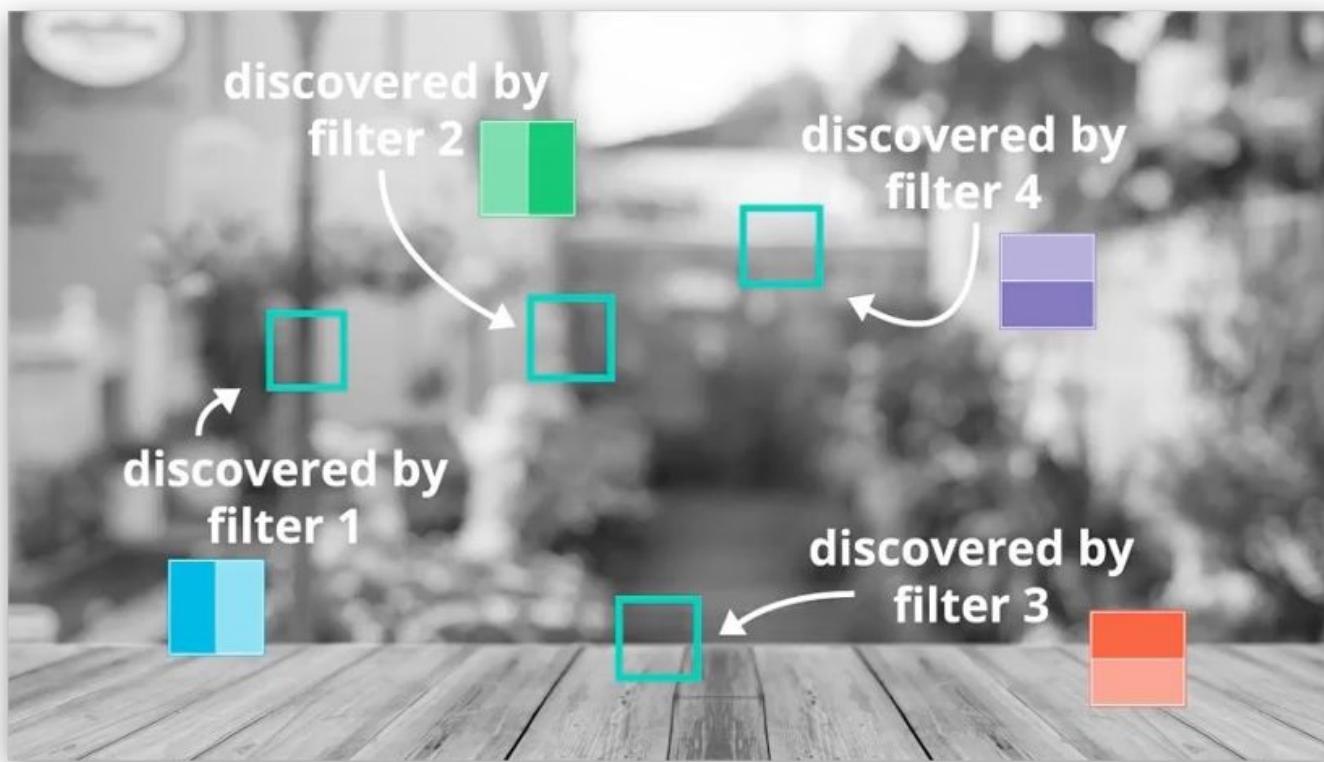
Lớp tích chập

- Giả sử có 6 nơ-ron tích chập sẽ sinh ra 6 bản đồ kích hoạt
- Các bản đồ kích hoạt ghép với nhau thành một “ảnh mới”

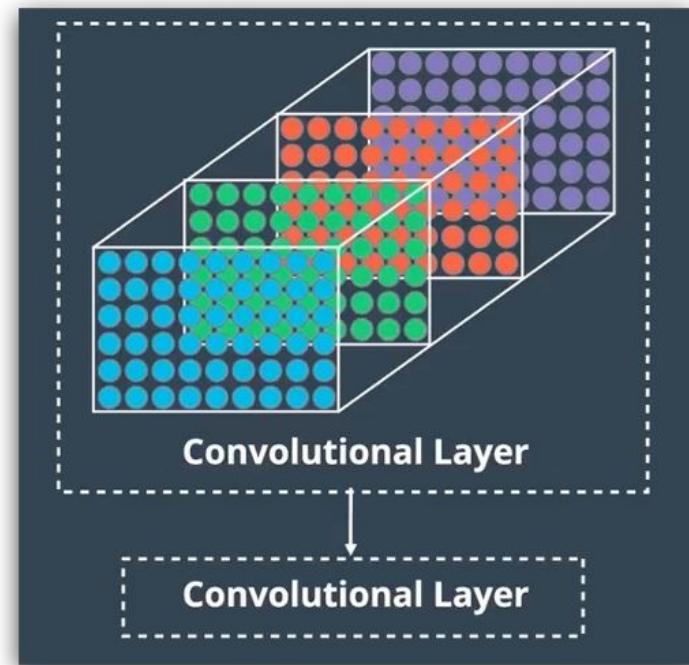
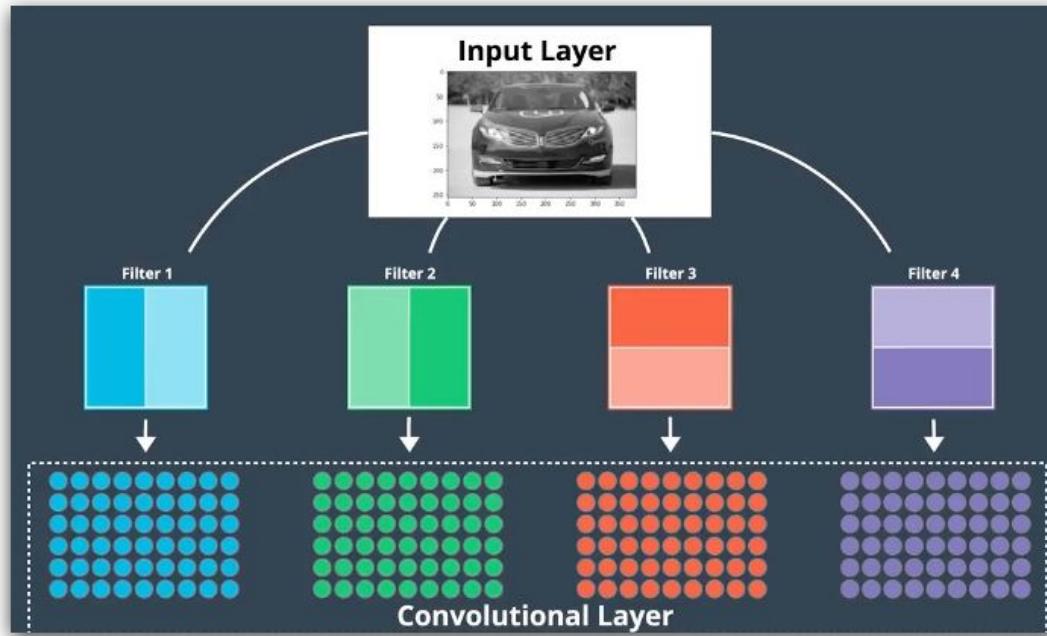


Lớp tích chập

- Cần nhiều filter: để nhận biết các đặc trưng khác nhau của đối tượng

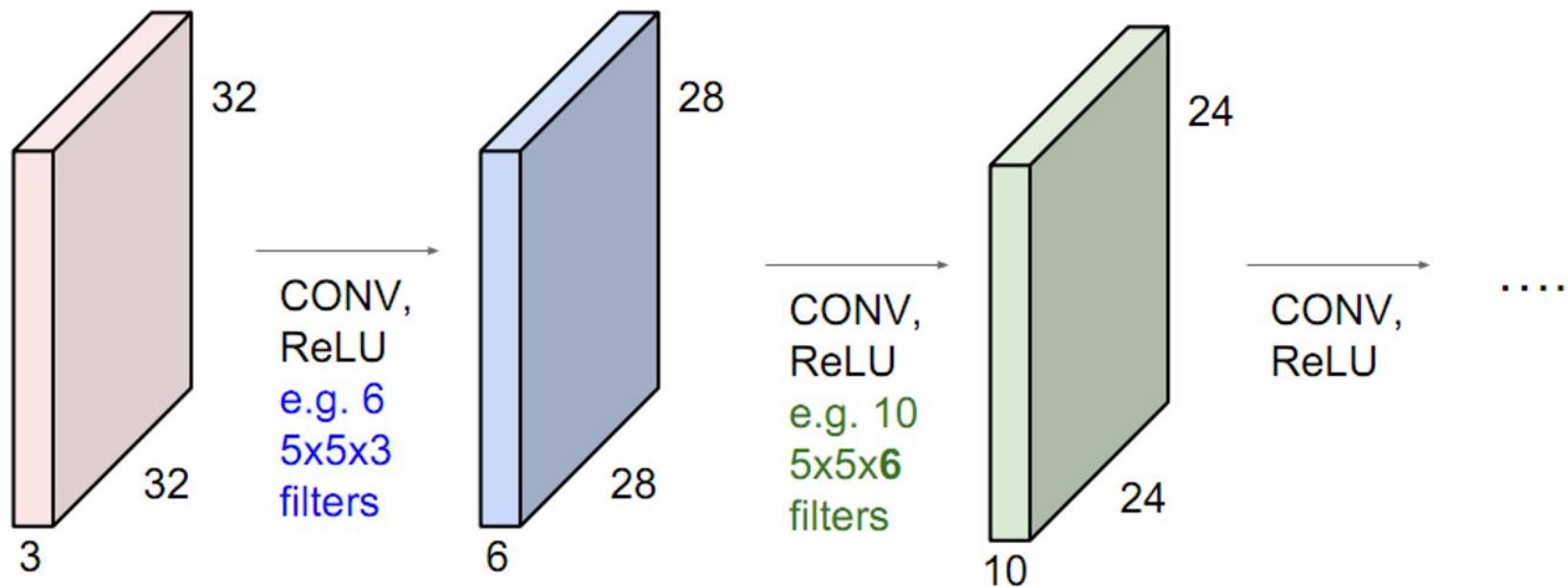


Lớp tích chập



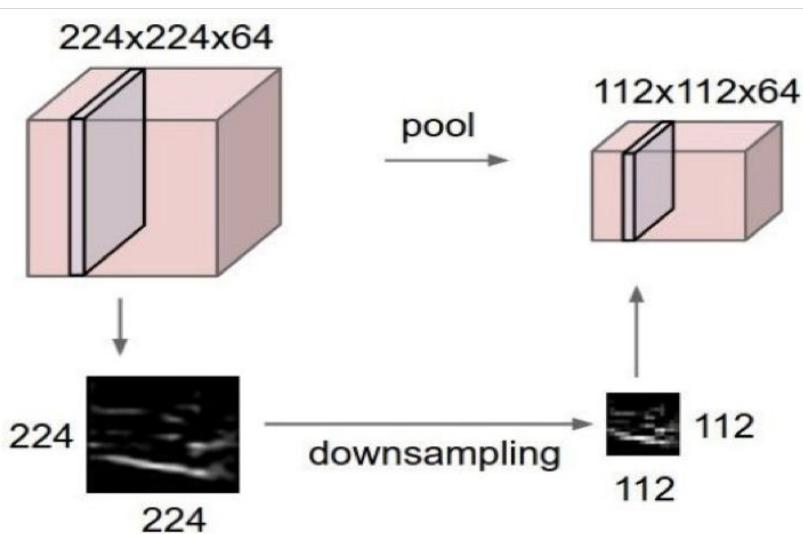
CNNs

- Mạng nơ-ron tích chập là một dãy các lớp tích chập nối liên tiếp nhau xen kẽ bởi các hàm kích hoạt (ví dụ ReLU)

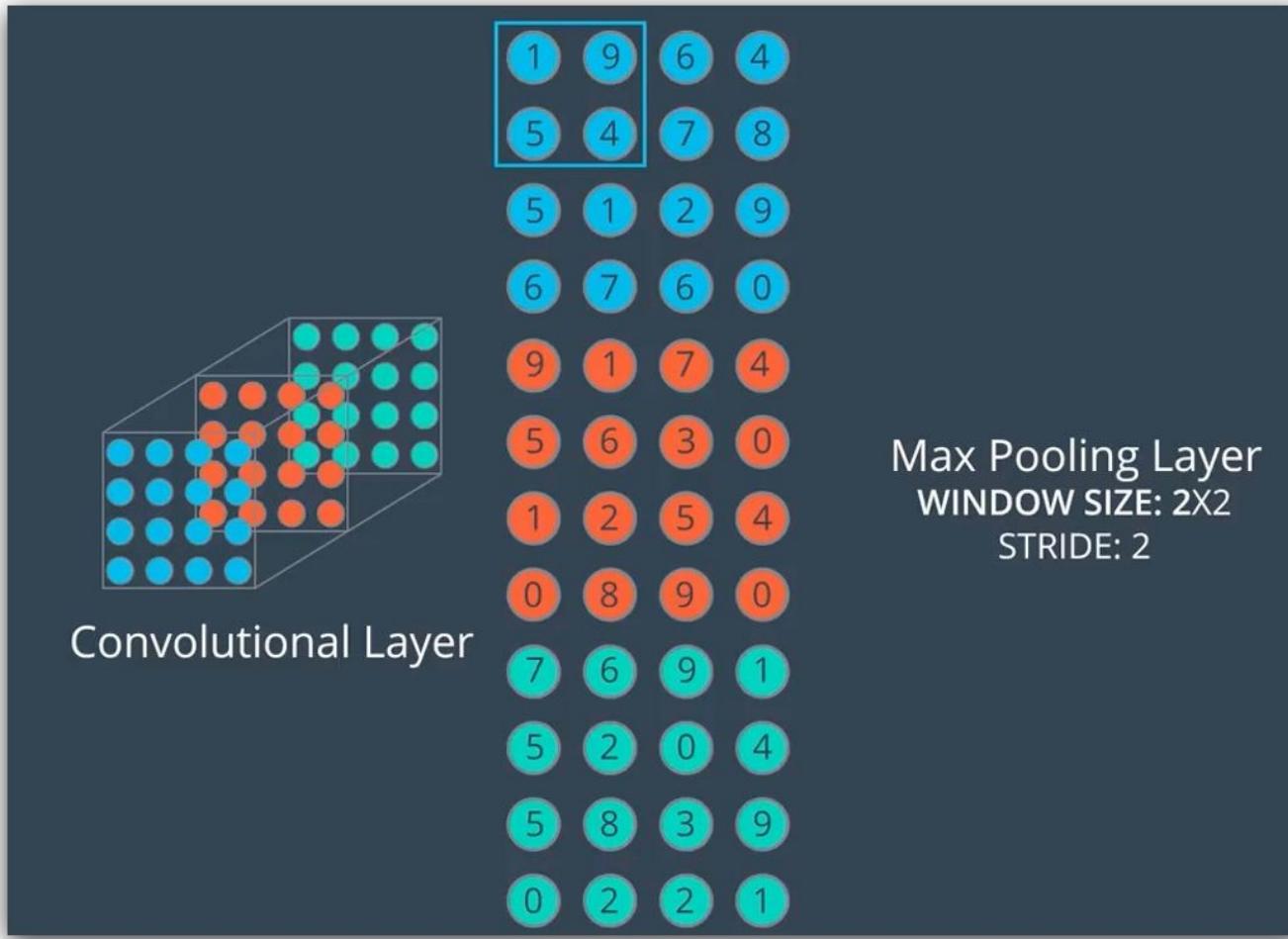


Lớp gộp (pooling layer)

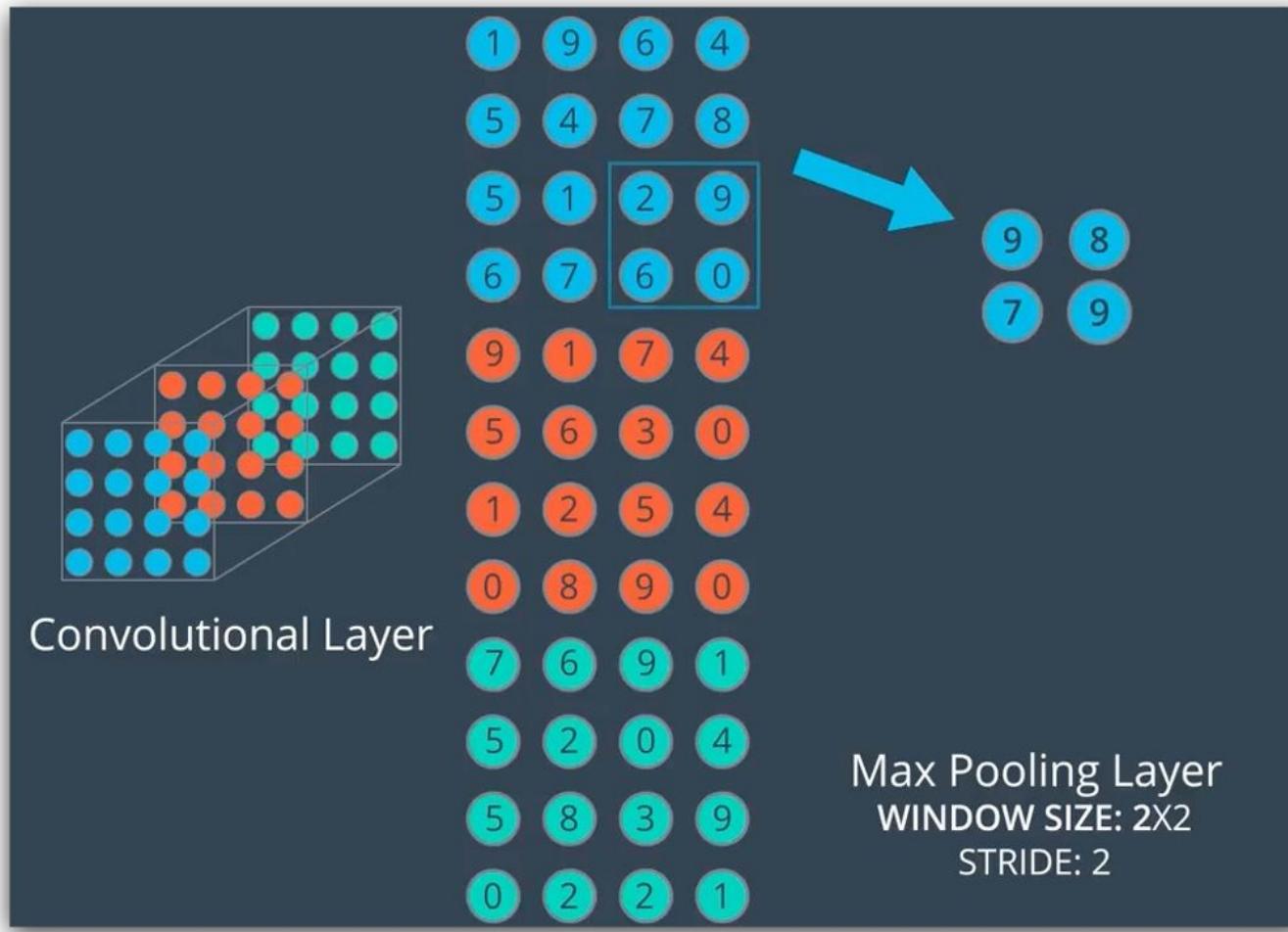
- Giúp giảm độ phân giải của khối dữ liệu để giảm bộ nhớ và khối lượng tính toán
- Hoạt động độc lập trên từng bản đồ kích hoạt
- Lớp gộp max pooling giúp mạng biểu diễn bất biến đối với các thay đổi tịnh tiến (translation invariance) hoặc biến dạng (deformation invariance) của dữ liệu đầu vào



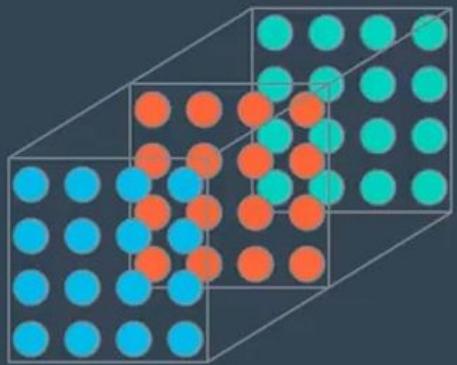
Lớp gộp max pooling



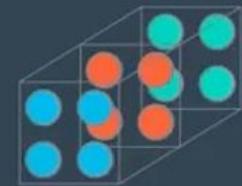
Lớp gộp max pooling



Lớp gộp max pooling

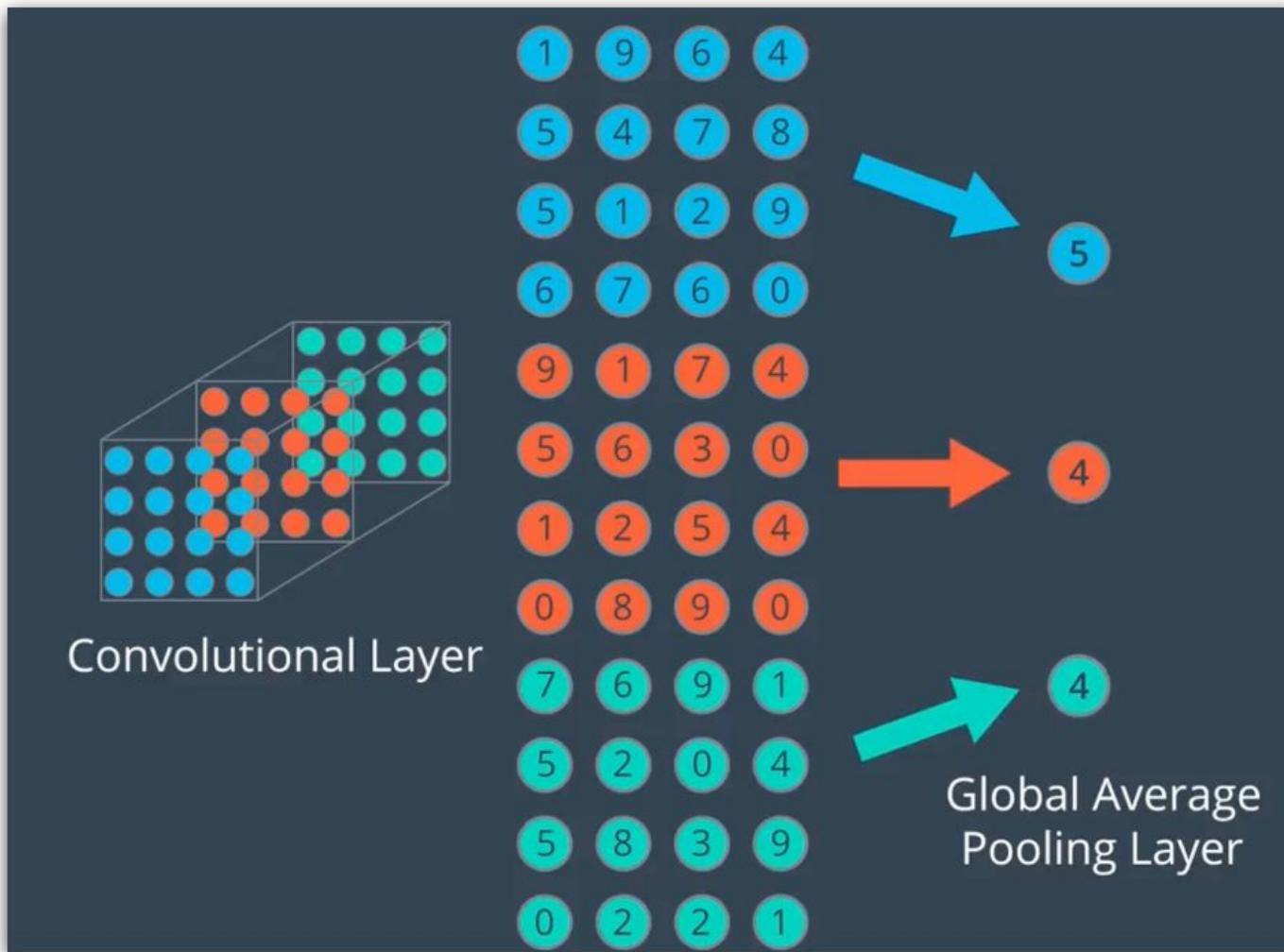


Convolutional Layer



Max Pooling Layer
WINDOW SIZE: 2X2
STRIDE: 2

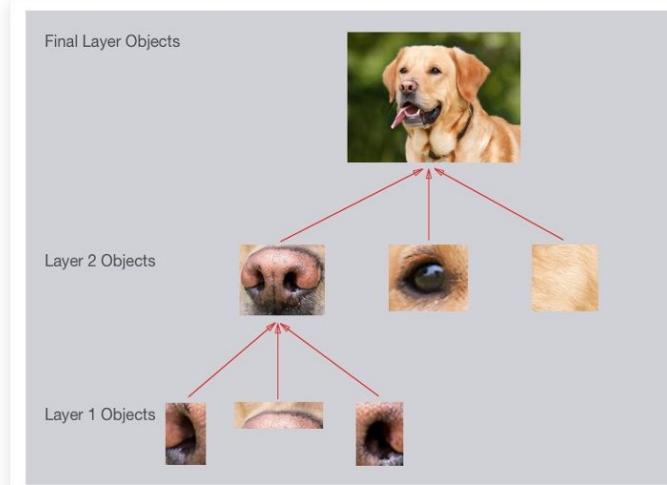
Lớp gộp Average pooling



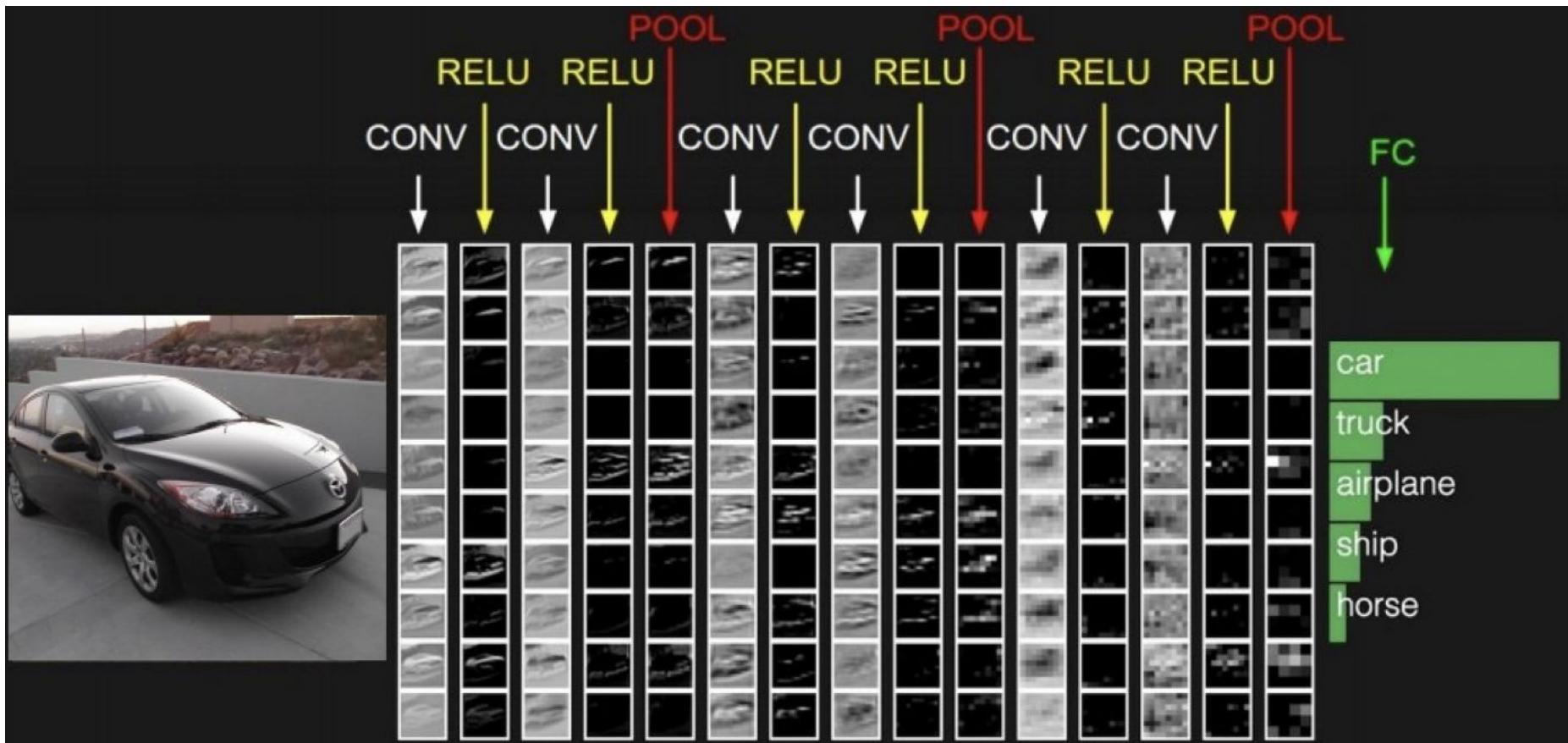
Hoạt động CNN

- Con người: nhận biết các hình đơn giản rồi đến thành phần phức tạp hơn của đối tượng và cuối cùng là cả đối tượng
- Con người: nhận biết con chó qua từng thành phần: mũi , mắt, lông...
- CNN: dùng nhiều kernel để nhận ra các đặc trưng khác nhau
 - Số lượng kernel (filter): filter depth

Chú ý: CNN tự học được các đặc trưng, không cần trích chọn đặc trưng

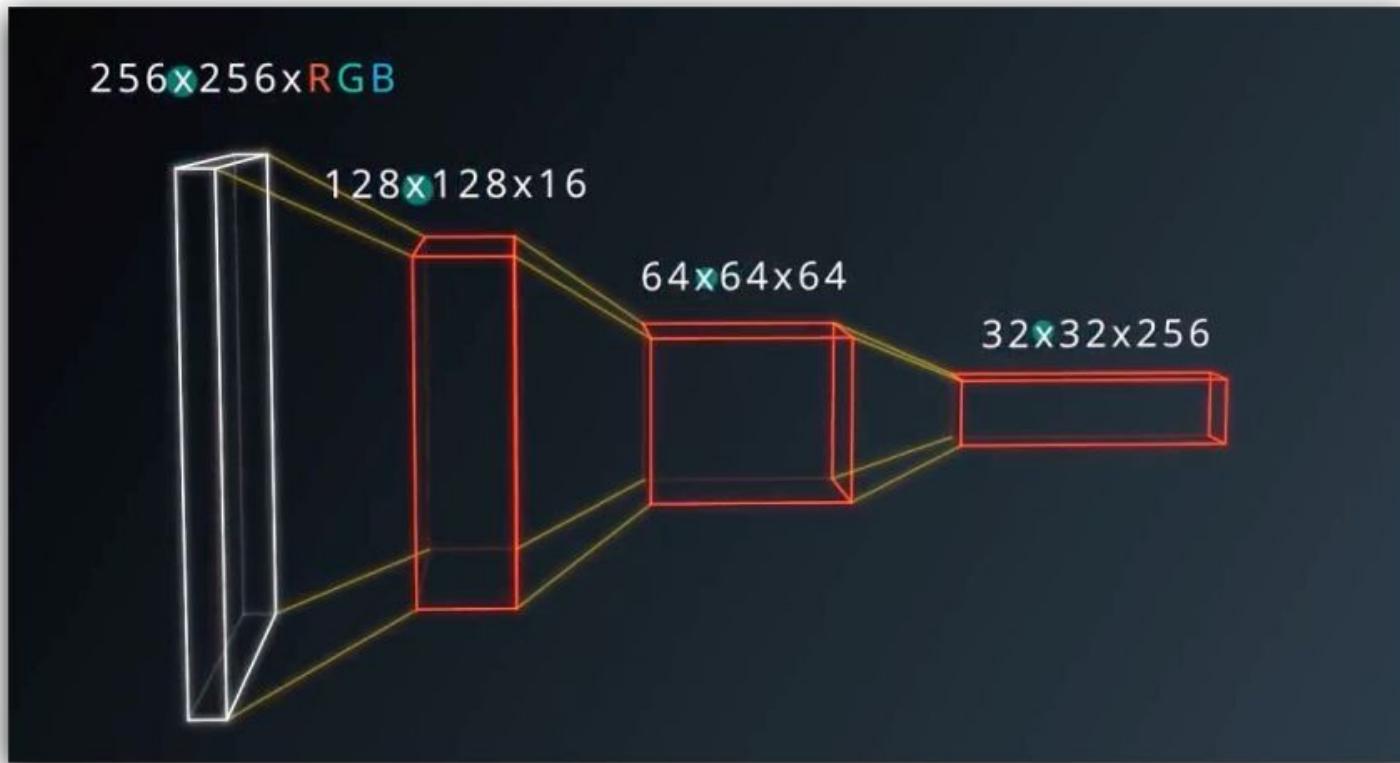


Hoạt động CNN



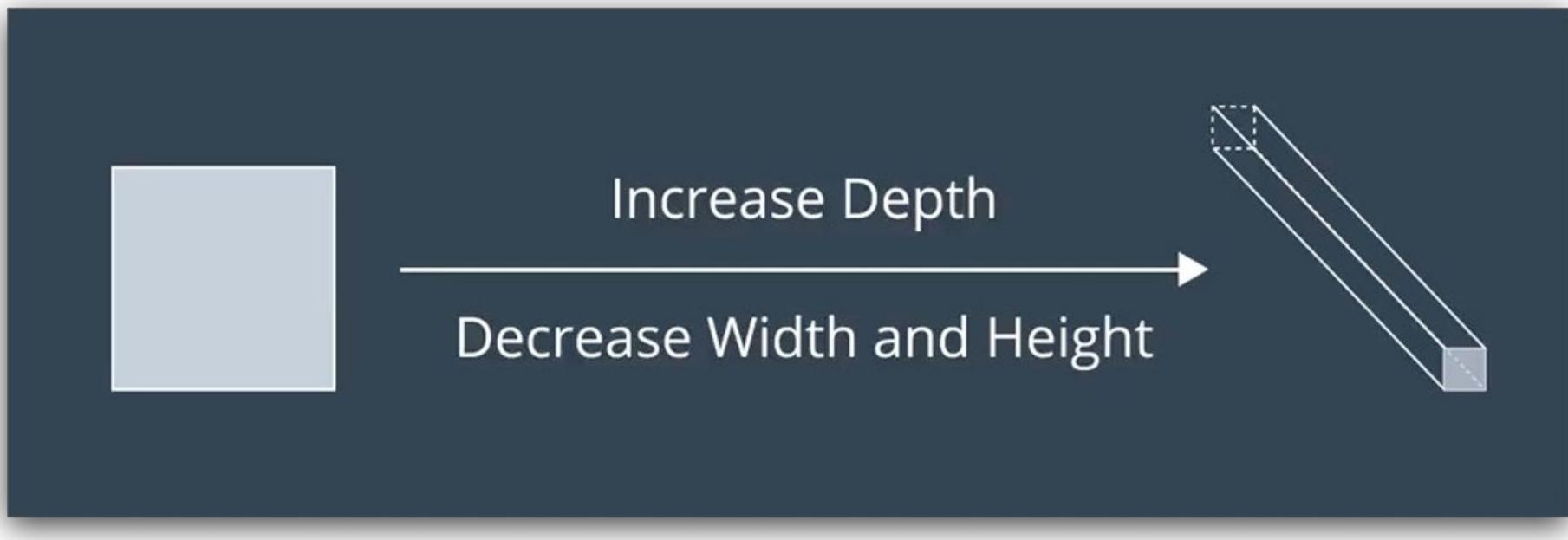
Kiến trúc CNN

Từ ảnh gốc, các layer sau ảnh giảm kích thước nhưng tăng chiều sâu



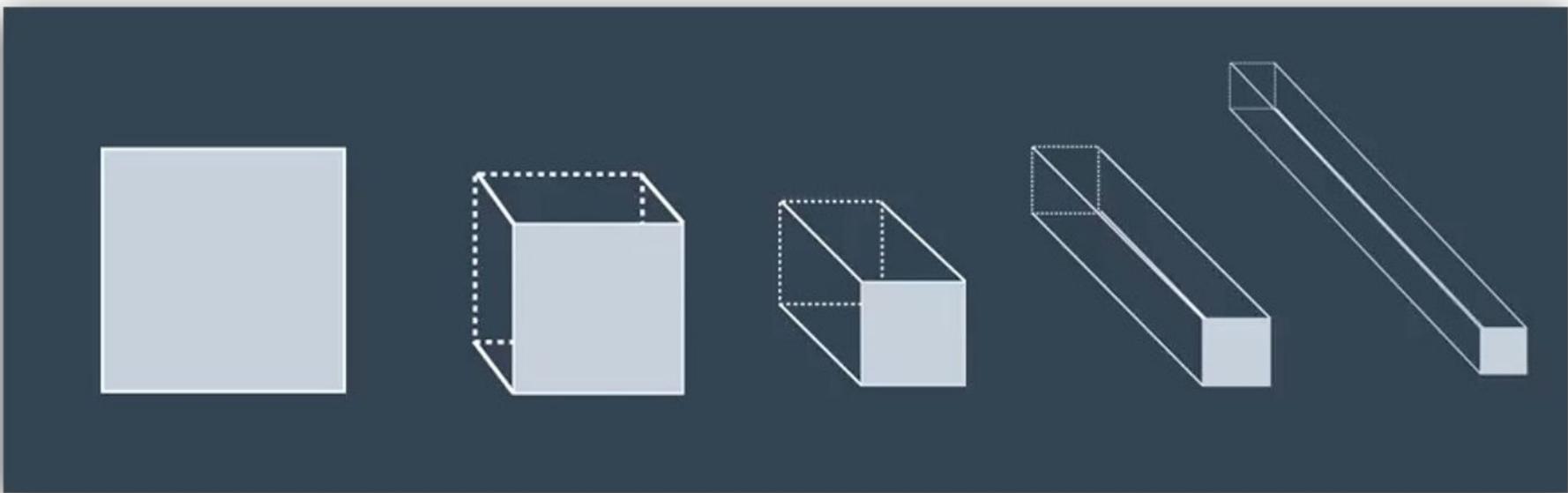
Kiến trúc CNN

Increase depth bởi Convolution layer Max Pooling giảm kích cỡ



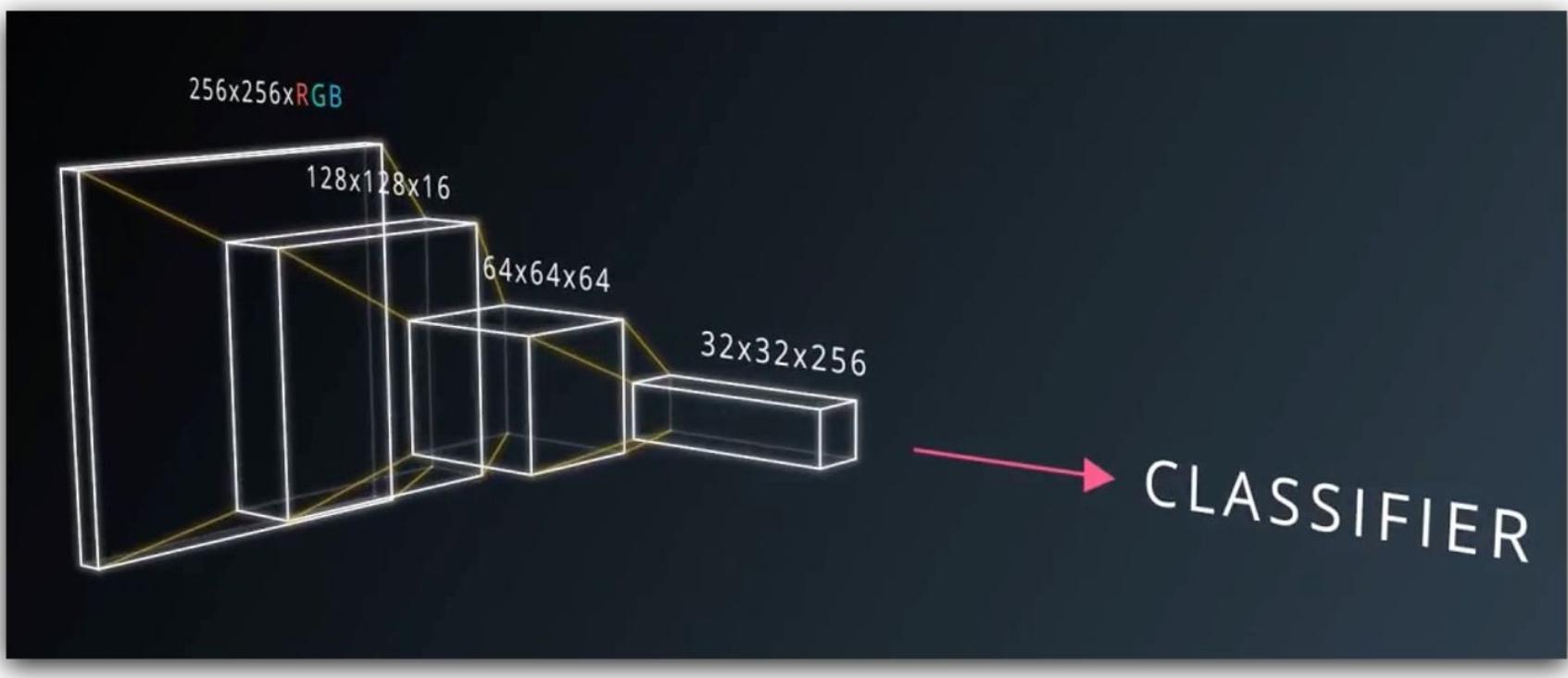
Kiến trúc CNN

Increase depth bởi Convolution layer Max Pooling giảm kích cỡ



Kiến trúc CNN

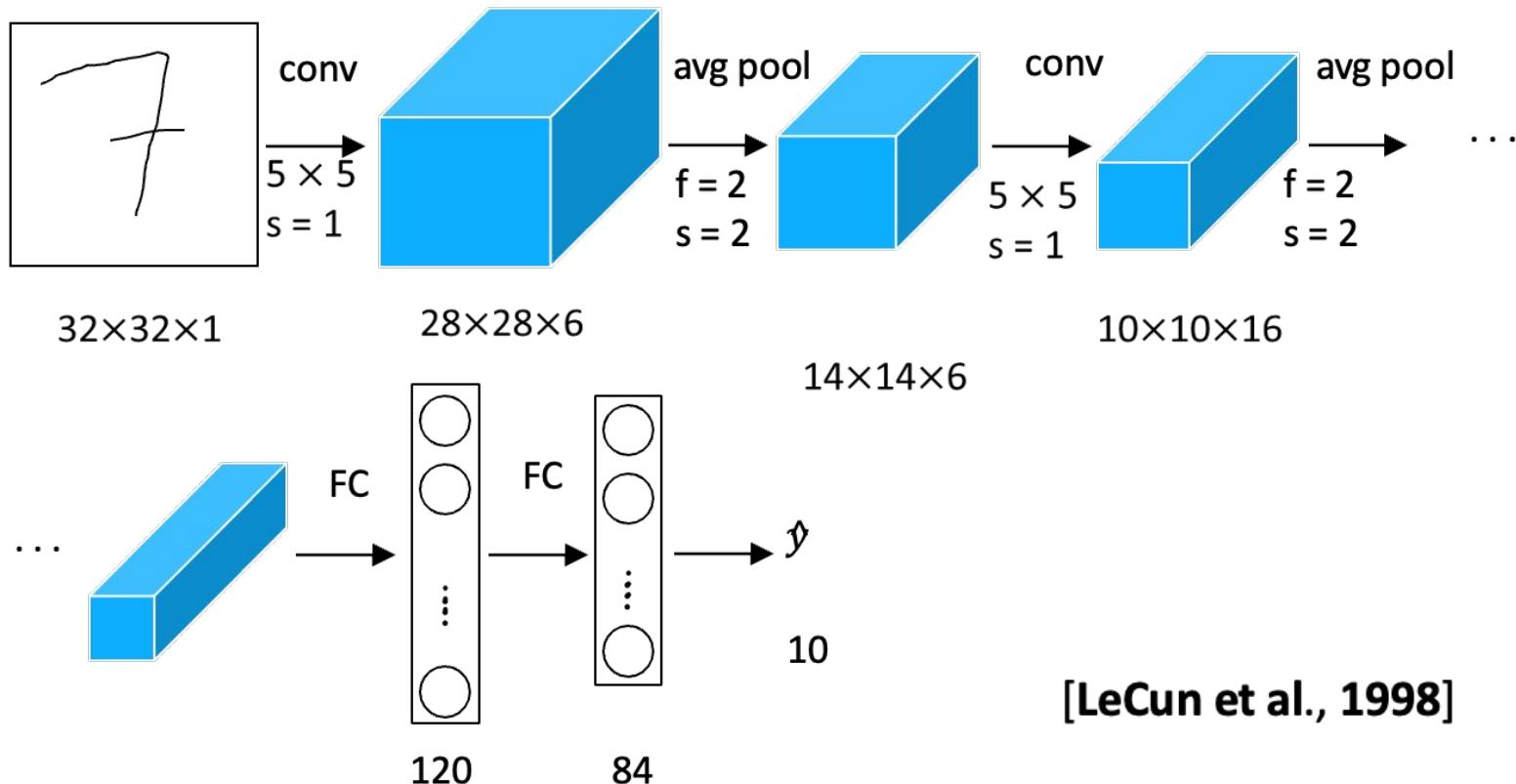
Bộ phân loại ở trên cùng



Một số mạng CNNs cơ bản

- LeNet-5
- AlexNet
- VGG
- GoogleNet
- ResNet
- ...

LeNet

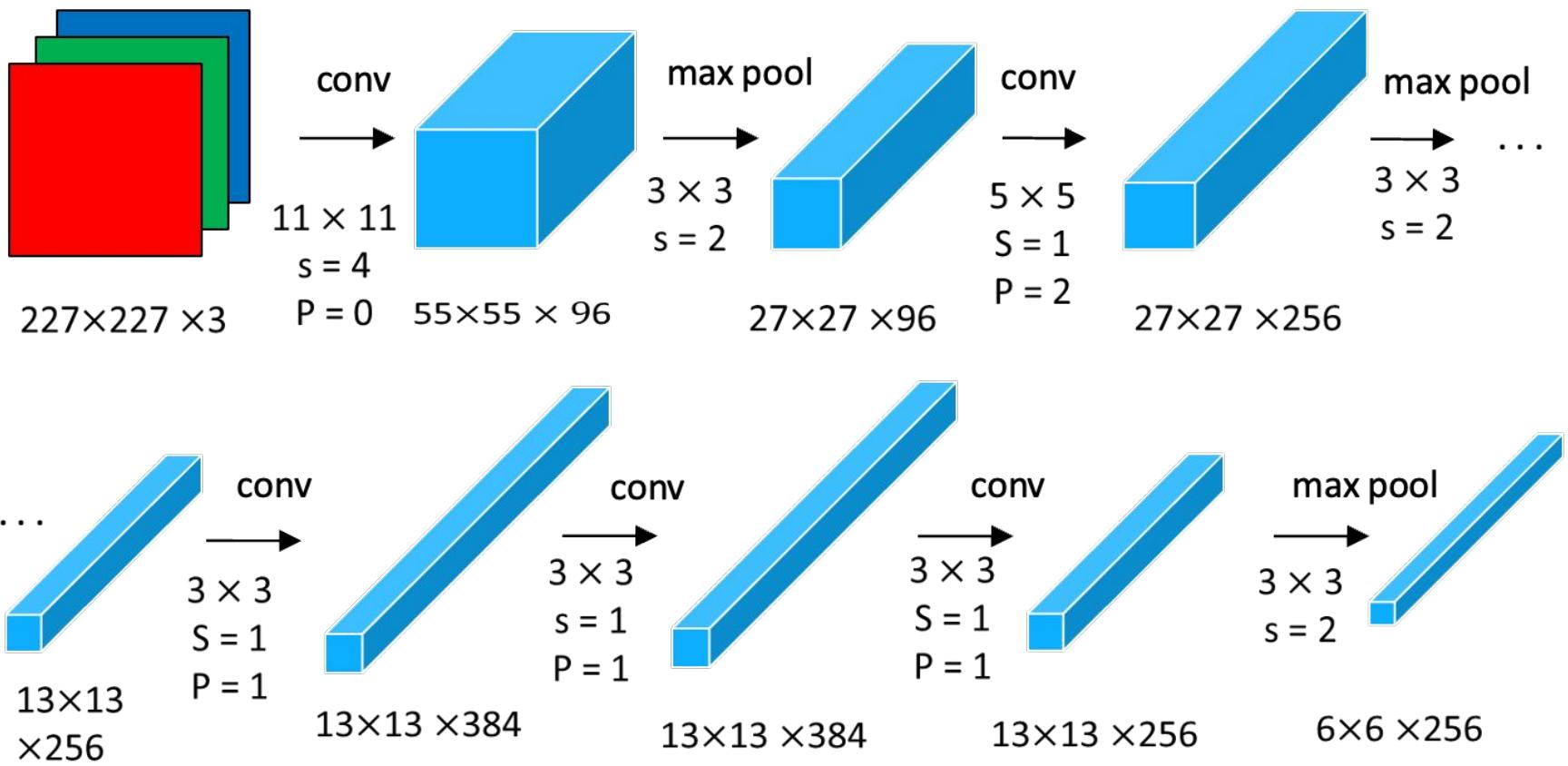


- Lưu ý: Output size = $(N+2P-F)/\text{stride} + 1$

- Những ảnh hưởng đến mạng CNN hiện tại:
- Fixed size input images
- Lặp lại Block:
 - convolutional layer
 - pooling layer
- Số lượng filter tăng dần theo chiều sâu của mạng
- Có 2 thành phần riêng biệt cho việc trích chọn đặc trưng và phân loại

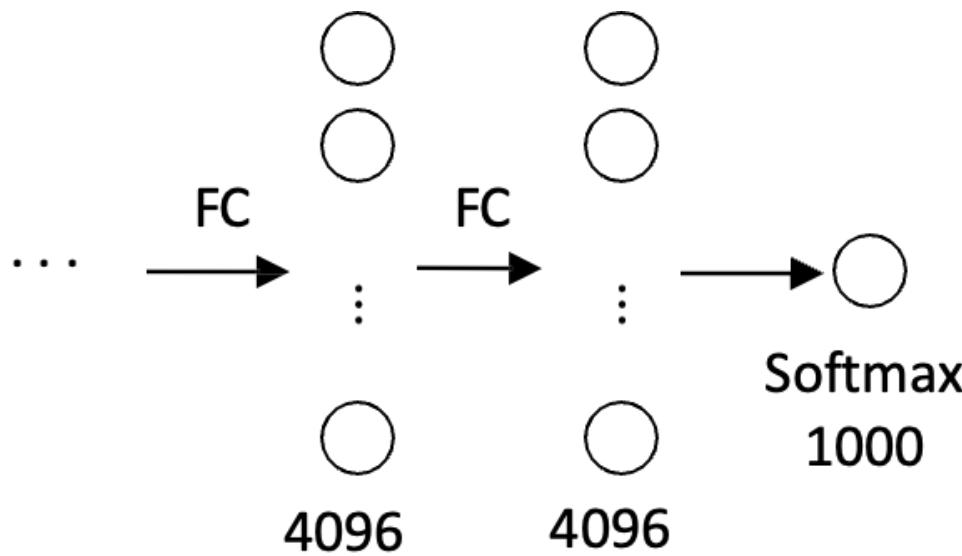
- ImageNet Classification with Deep Convolutional Neural Networks - Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton; 2012
- Một trong những mạng CNNs lớn nhất tại thời điểm đó
- Đánh dấu sự phát triển của deep learning trong computer vision
- Có 60M tham số so với 60k tham số LeNet
- Mất khoảng 1 tuần để huấn luyện bằng 02 GPU tốt nhất tại thời điểm đó

AlexNet



[Krizhevsky et al., 2012]

AlexNet



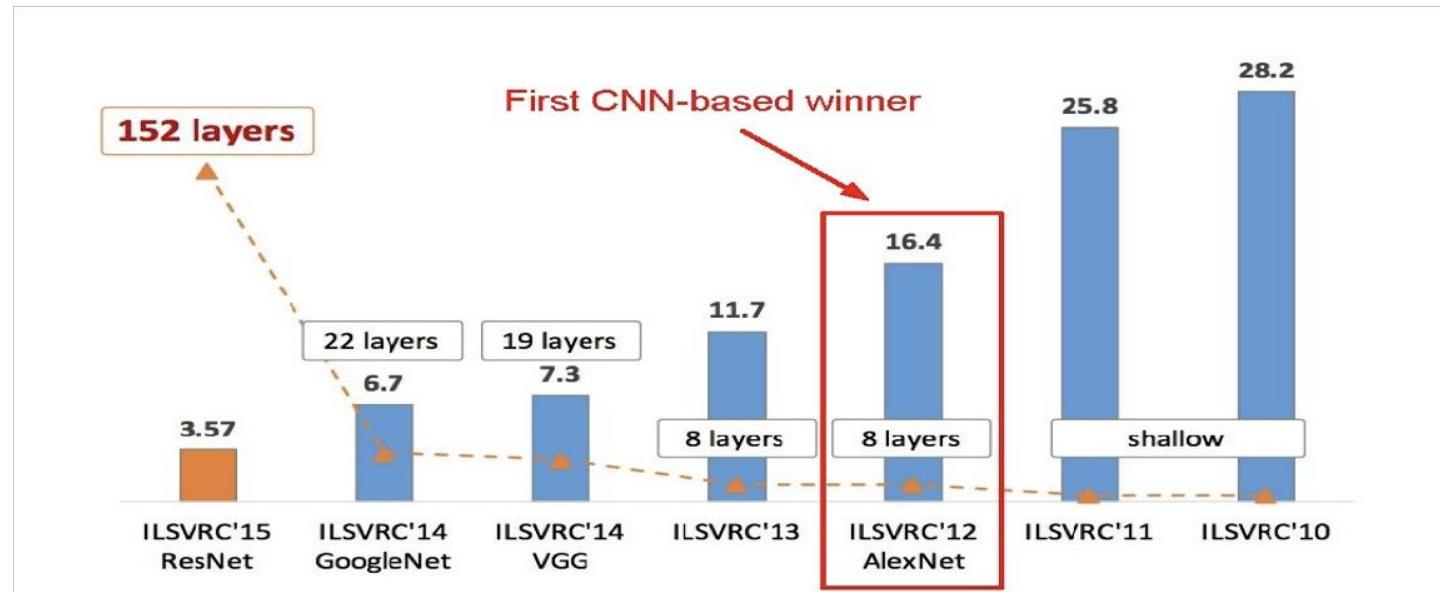
[Krizhevsky et al., 2012]

Ảnh hưởng đến mạng CNN hiện nay:

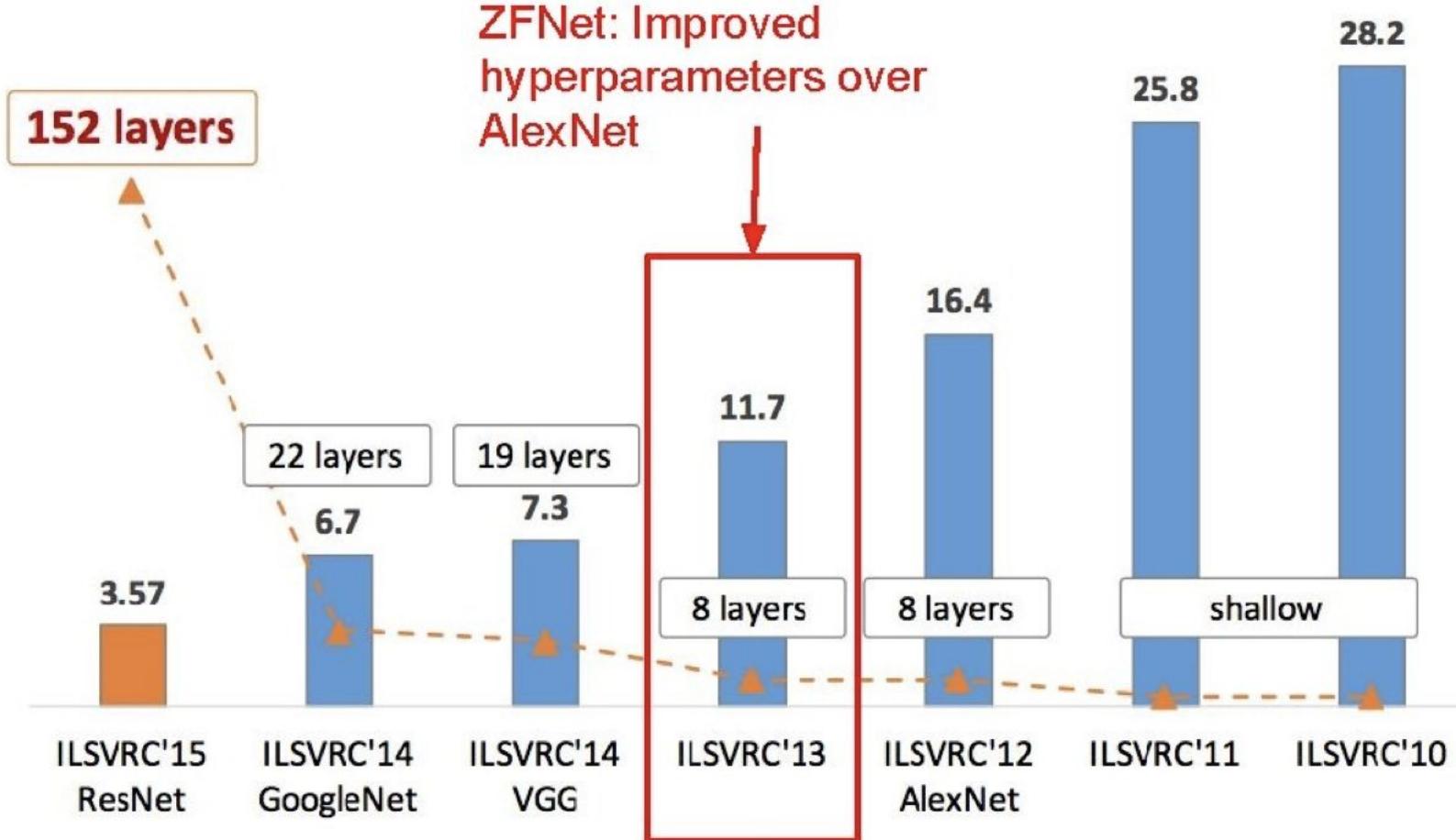
- Sử dụng hàm kích hoạt ReLU thay vì Sigmoid hay Tanh (S-shape function)
- Sử dụng hàm softmax cho mục tiêu phân loại
- Dropout giữa các lớp fully connected tránh việc bị overfitting
- Block: Conv-Conv
- Sử dụng data augmentation

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

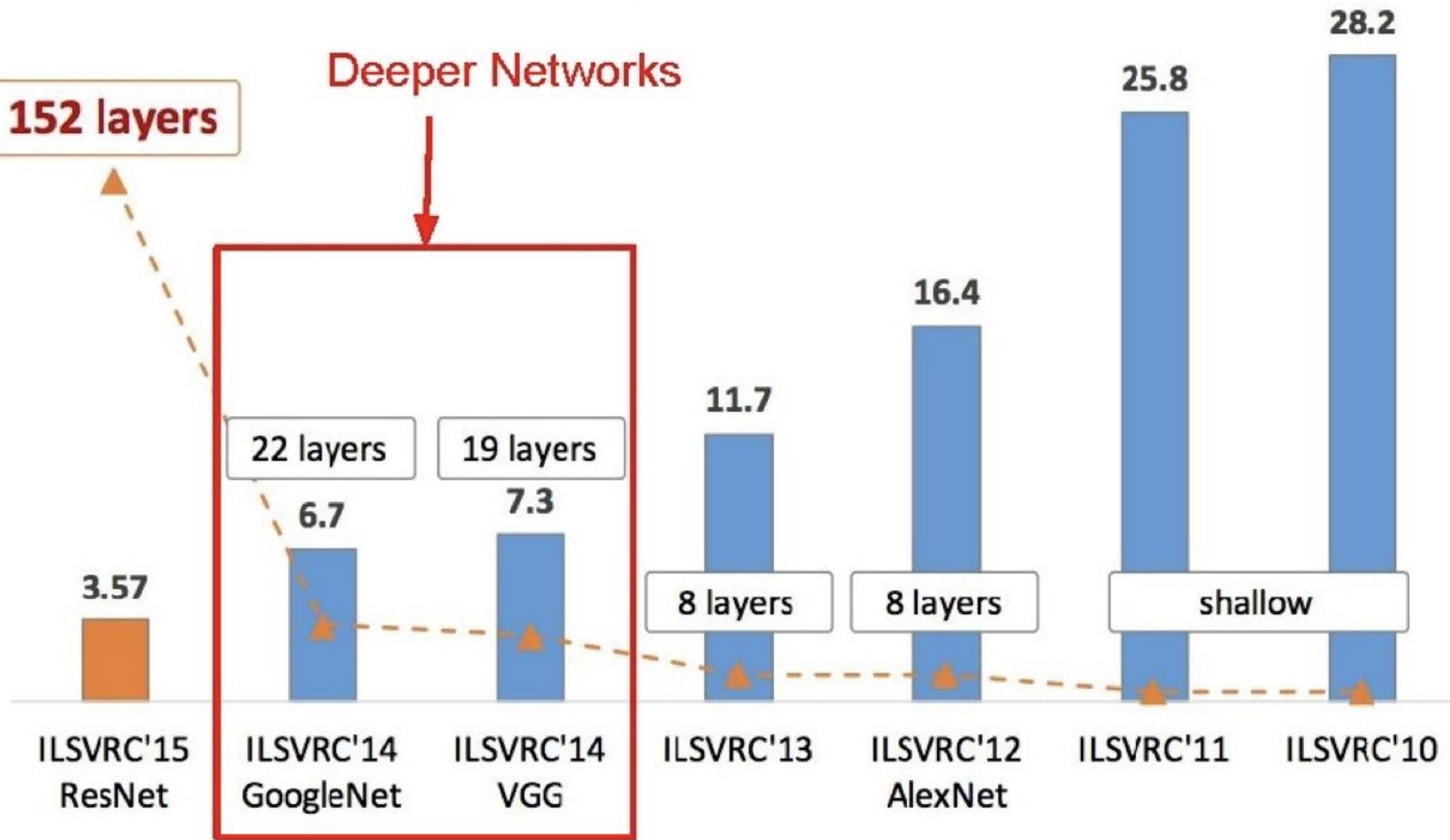
- “Olympics” thường niên về lĩnh vực thị giác máy tính.
- Các teams khắp thế giới thi đấu với nhau để xem ai là người có mô hình CV tốt nhất cho các bài toán như phân loại ảnh, định vị và phát hiện đối tượng trong ảnh



ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



VGGNet

- Very Deep Convolutional Networks For Large Scale Image Recognition - Karen Simonyan and Andrew Zisserman; 2015
- Á quân tại cuộc thi ILSVRC 2014
- Sâu hơn rất nhiều so với AlexNet
- 140 triệu tham số

VGGNet

Input

3x3 conv, 64

3x3 conv, 64

Pool 1/2

3x3 conv, 128

3x3 conv, 128

Pool 1/2

3x3 conv, 256

3x3 conv, 256

Pool 1/2

3x3 conv, 512

3x3 conv, 512

3x3 conv, 512

Pool 1/2

3x3 conv, 512

3x3 conv, 512

3x3 conv, 512

Pool 1/2

FC 4096

FC 4096

FC 1000

Softmax

Nơron kích thước bé

- Chỉ dùng conv 3x3, stride 1, pad 1 và 2x2 MAX POOL , stride 2

Mạng sâu hơn AlexNet:

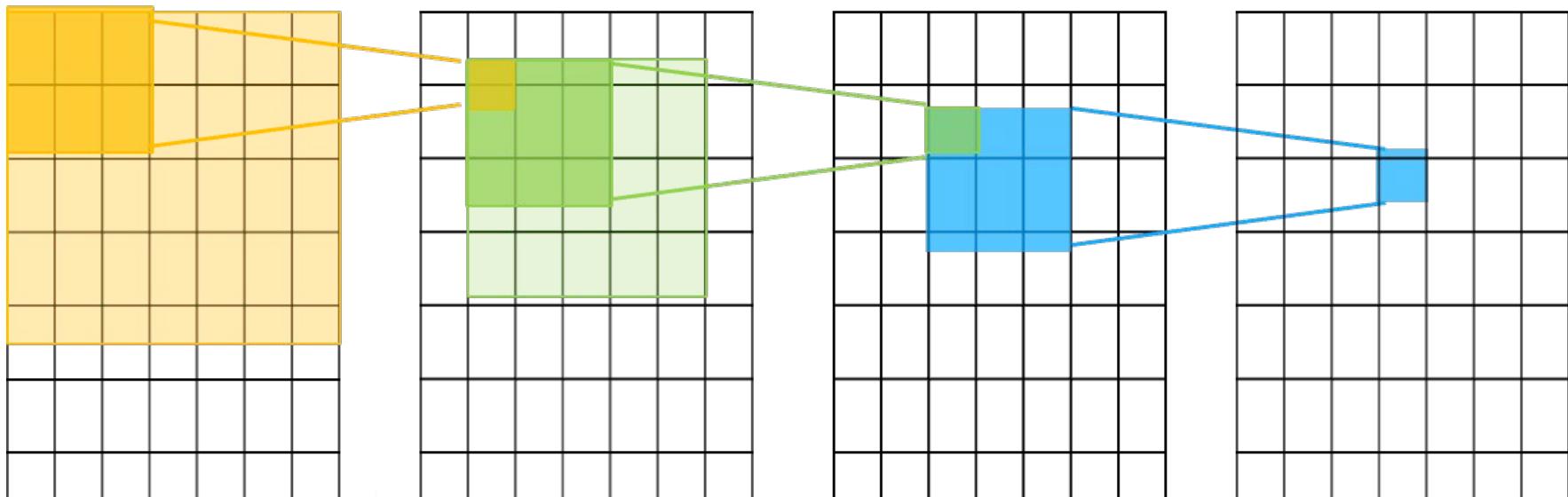
- 8 lớp VGGNet: 16 - 19 lớp

ZFNet: 11.7% top 5 error in ILSVRC'13

VGGNet: 7.3% top 5 error in ILSVRC'14

Tại sao dùng filter bé? (3x3 conv)

- Chồng 3 lớp 3x3 conv (stride 1) có cùng hiệu quả thu nhận thông tin như một lớp 7x7 conv.
- Nhưng sâu hơn, nhiều lớp phi tuyến hơn
- Và ít tham số hơn



VGGNet

| | | |
|---------------|--------------------------|--------------------------------------|
| Input | memory: 224*224*3=150K | params: 0 |
| 3x3 conv, 64 | memory: 224*224*64=3.2M | params: $(3*3*3)*64 = 1,728$ |
| 3x3 conv, 64 | memory: 224*224*64=3.2M | params: $(3*3*64)*64 = 36,864$ |
| Pool | memory: 112*112*64=800K | params: 0 |
| 3x3 conv, 128 | memory: 112*112*128=1.6M | params: $(3*3*64)*128 = 73,728$ |
| 3x3 conv, 128 | memory: 112*112*128=1.6M | params: $(3*3*128)*128 = 147,456$ |
| Pool | memory: 56*56*128=400K | params: 0 |
| 3x3 conv, 256 | memory: 56*56*256=800K | params: $(3*3*128)*256 = 294,912$ |
| 3x3 conv, 256 | memory: 56*56*256=800K | params: $(3*3*256)*256 = 589,824$ |
| 3x3 conv, 256 | memory: 56*56*256=800K | params: $(3*3*256)*256 = 589,824$ |
| Pool | memory: 28*28*256=200K | params: 0 |
| 3x3 conv, 512 | memory: 28*28*512=400K | params: $(3*3*256)*512 = 1,179,648$ |
| 3x3 conv, 512 | memory: 28*28*512=400K | params: $(3*3*512)*512 = 2,359,296$ |
| 3x3 conv, 512 | memory: 28*28*512=400K | params: $(3*3*512)*512 = 2,359,296$ |
| Pool | memory: 14*14*512=100K | params: 0 |
| 3x3 conv, 512 | memory: 14*14*512=100K | params: $(3*3*512)*512 = 2,359,296$ |
| 3x3 conv, 512 | memory: 14*14*512=100K | params: $(3*3*512)*512 = 2,359,296$ |
| 3x3 conv, 512 | memory: 14*14*512=100K | params: $(3*3*512)*512 = 2,359,296$ |
| Pool | memory: 7*7*512=25K | params: 0 |
| FC 4096 | memory: 4096 | params: $7*7*512*4096 = 102,760,448$ |
| FC 4096 | memory: 4096 | params: $4096*4096 = 16,777,216$ |
| FC 1000 | memory: 1000 | params: $4096*1000 = 4,096,000$ |

[Simonyan and Zisserman, 2014]

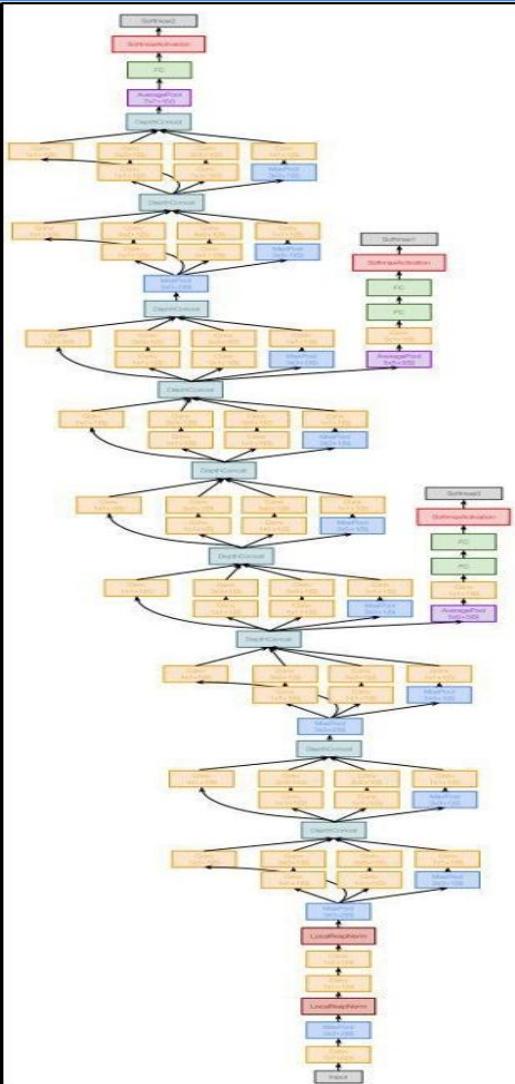
Ảnh hưởng VGGNet

- Sử dụng filter có kích thước nhỏ
- Sử dụng max pooling 2×2 và “same”
- Lặp lại block: Stack các lớp Conv trước khi đưa vào max pooling
- Model sâu với 16 hay 19 lớp
- Cung cấp pre-trained weight

GoogleNet

- Going Deeper with Convolutions - Christian Szegedy et al.;
2015
- Vô địch ILSVRC 2014
- Sâu hơn nhiều so với AlexNet
- Số tham số ít hơn 12 lần so với AlexNet
- Tập trung vào giảm độ phức tạp tính toán

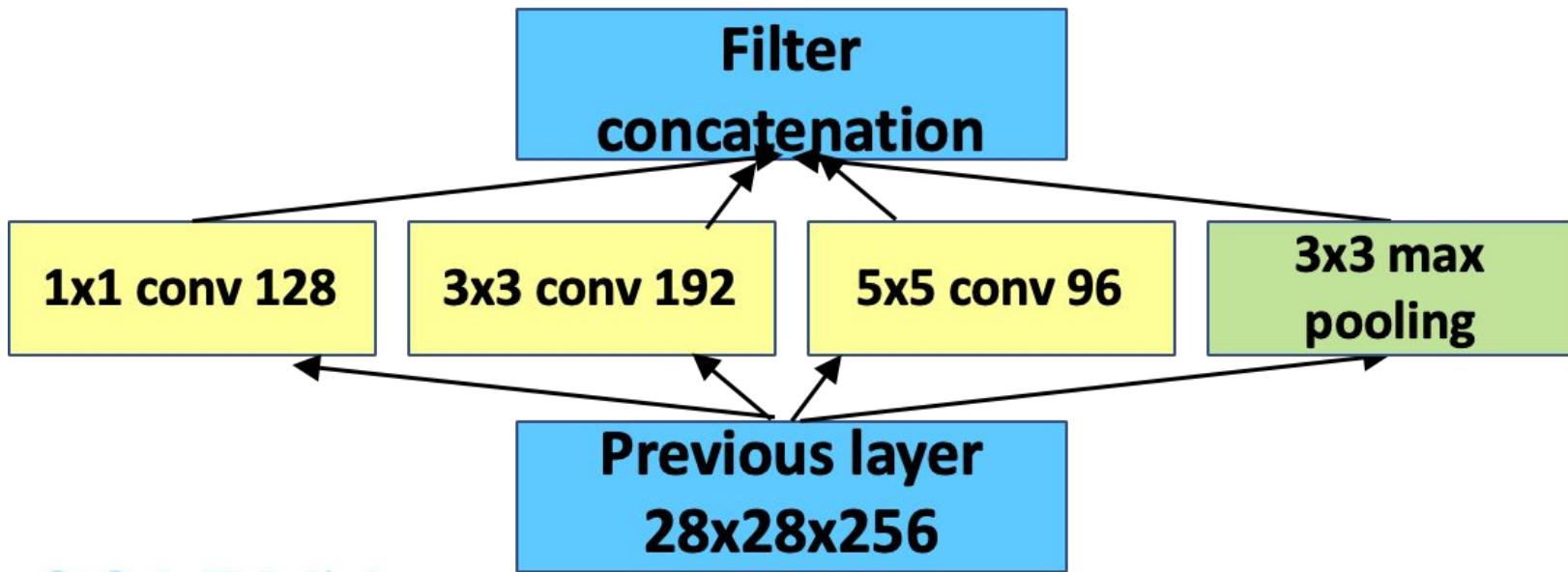
GoogleNet -



- 22 lớp
- Khối “Inception”
- Không có lớp kết nối đầy đủ (FC layers)
- Chỉ 5 triệu tham số!
- Vô địch tác vụ phân loại ảnh ILSVRC’14 (6.7% top 5 error)

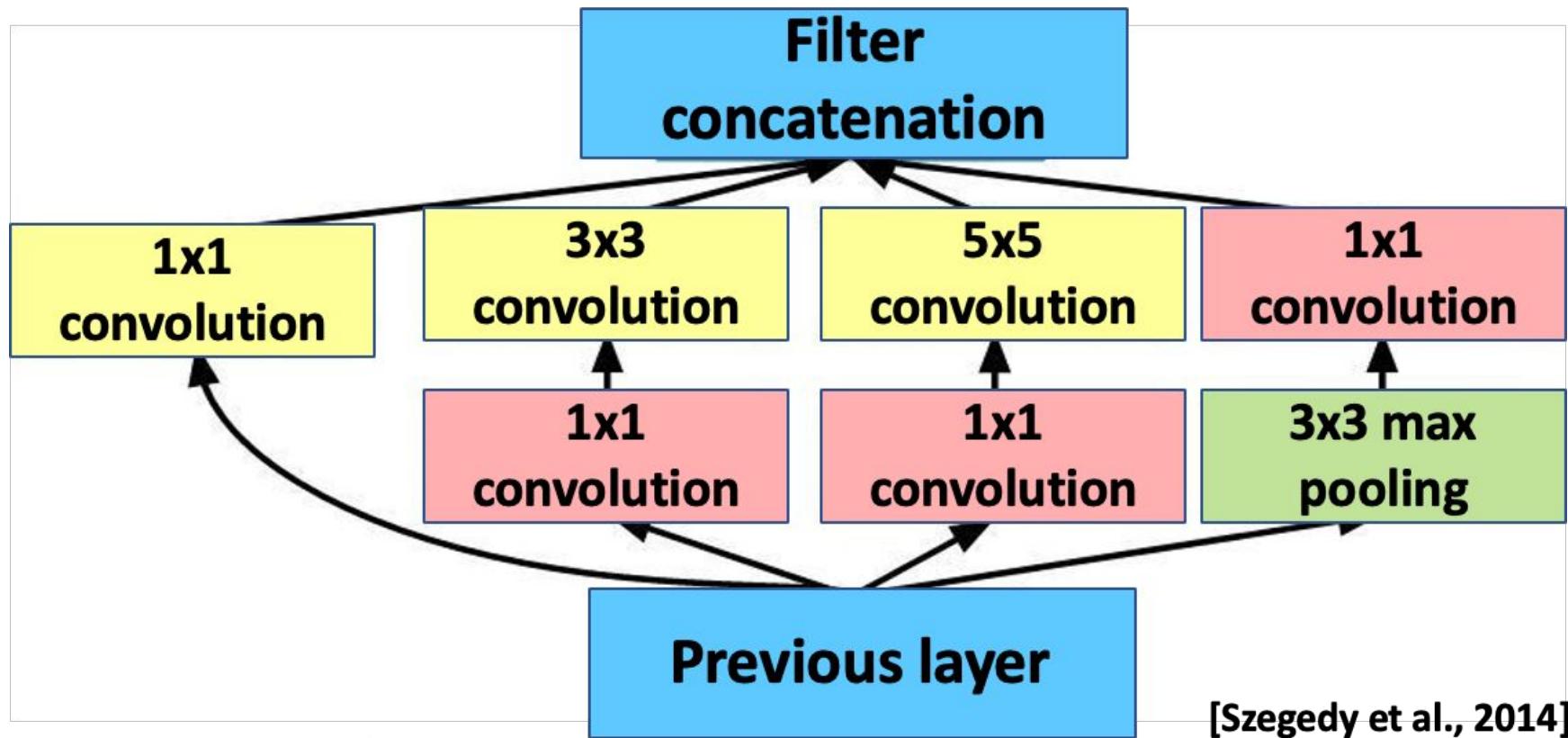
GoogleNet - Naïve Inception Model

- Số lượng phép tích chập:
- 1x1 conv, 128: $28 \times 28 \times 128 \times 1 \times 1 \times 256$
- 3x3 conv, 192: $28 \times 28 \times 192 \times 3 \times 3 \times 256$
- 5x5 conv, 96: $28 \times 28 \times 96 \times 5 \times 5 \times 256$
- Tổng cộng: 854M ops ==> Tính toán rất nặng!



GoogleNet

- Giải pháp: lớp nút cổ chai “bottleneck” sử dụng conv 1x1 để giảm chiều sâu khối dữ liệu.



[Szegedy et al., 2014]

GoogleNet

Số lượng phép toán tích chập:

1x1 conv, 64: $28 \times 28 \times 64 \times 1 \times 1 \times 256$

1x1 conv, 64: $28 \times 28 \times 64 \times 1 \times 1 \times 256$

1x1 conv, 128: $28 \times 28 \times 128 \times 1 \times 1 \times 256$

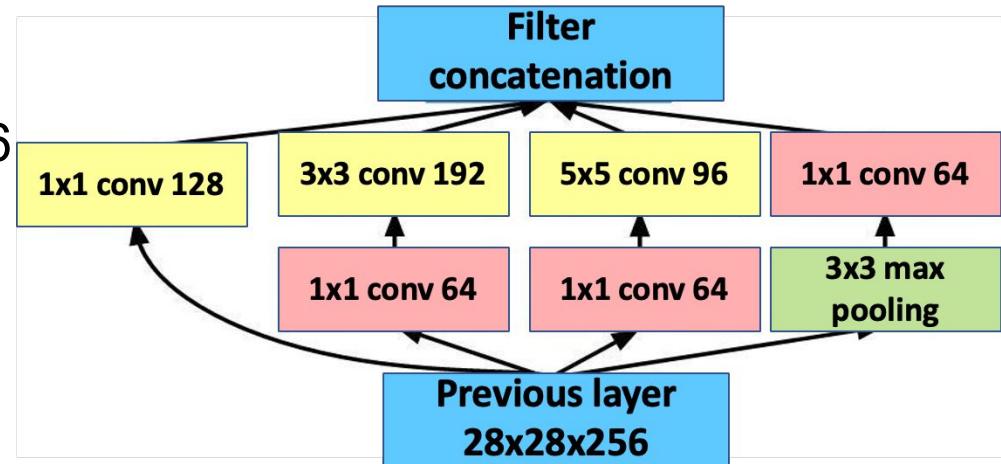
3x3 conv, 192: $28 \times 28 \times 192 \times 3 \times 3 \times 64$

5x5 conv, 96: $28 \times 28 \times 96 \times 5 \times 5 \times 264$

1x1 conv, 64: $28 \times 28 \times 64 \times 1 \times 1 \times 256$

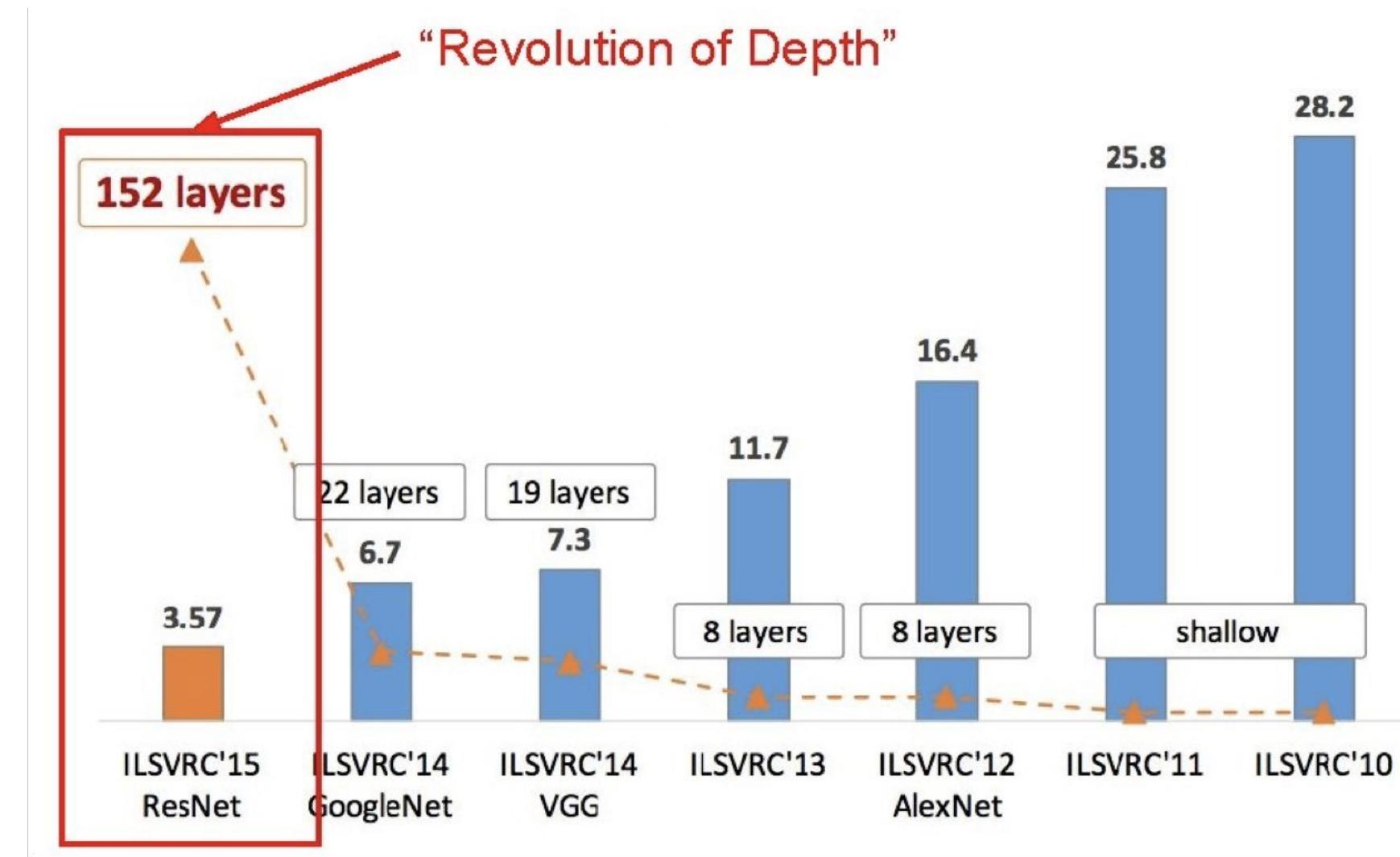
Tổng số: 353M ops

So với 854M ops với khối inception thường



[Szegedy et al., 2014]

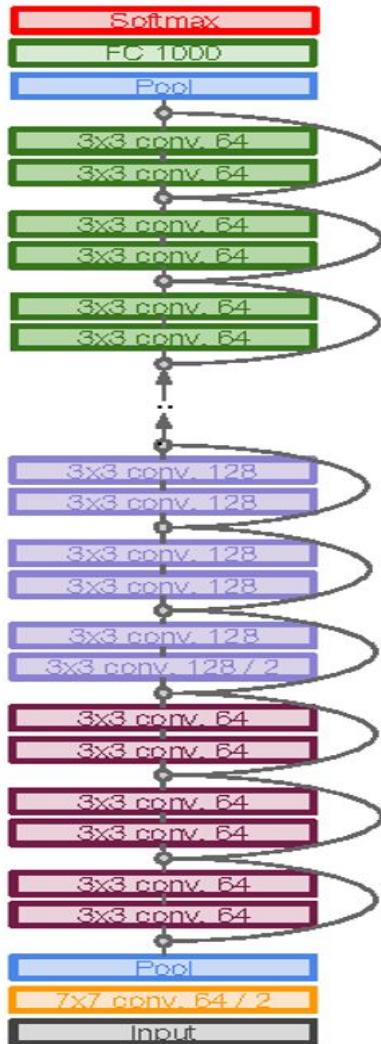
ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



ResNet

- Deep Residual Learning for Image Recognition - Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun; 2015
- Mạng rất sâu, tới 152 lớp
- Mạng càng sâu càng khó huấn luyện.
- Mạng càng sâu càng chịu nhiều ảnh hưởng của vấn đề triệt tiêu và bùng nổ gradient.
- ResNet đề xuất phương pháp học phần dư (residual learning) cho phép huấn luyện hiệu quả các mạng sâu hơn rất nhiều so với các mạng xuất hiện trước đó.

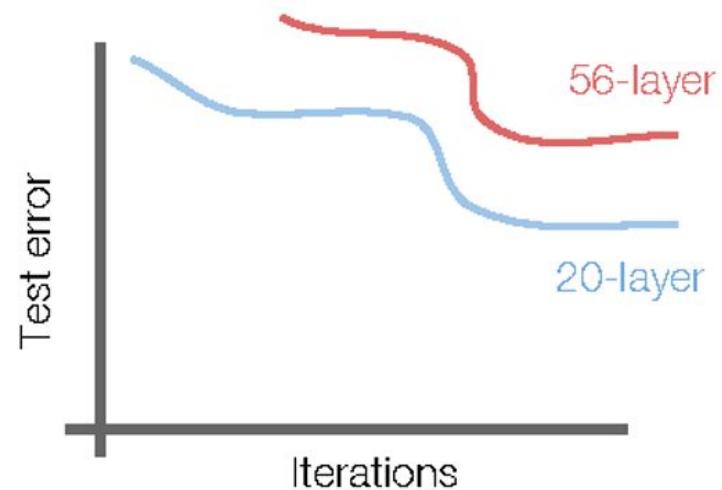
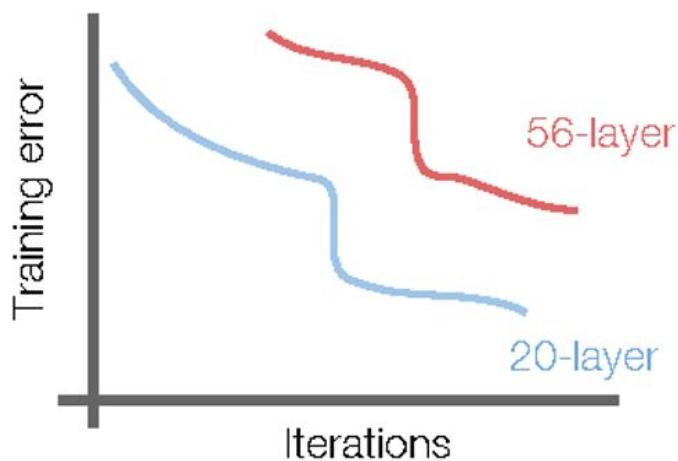
ResNet



- Vô địch tác vụ phân loại ILSVRC'15 (3.57% top 5 error, trong khi sai số của con người khoảng 5.1%)
- Càn quét tất cả các cuộc thi về phân loại ảnh tại ILSVRC'15 và COCO'15!

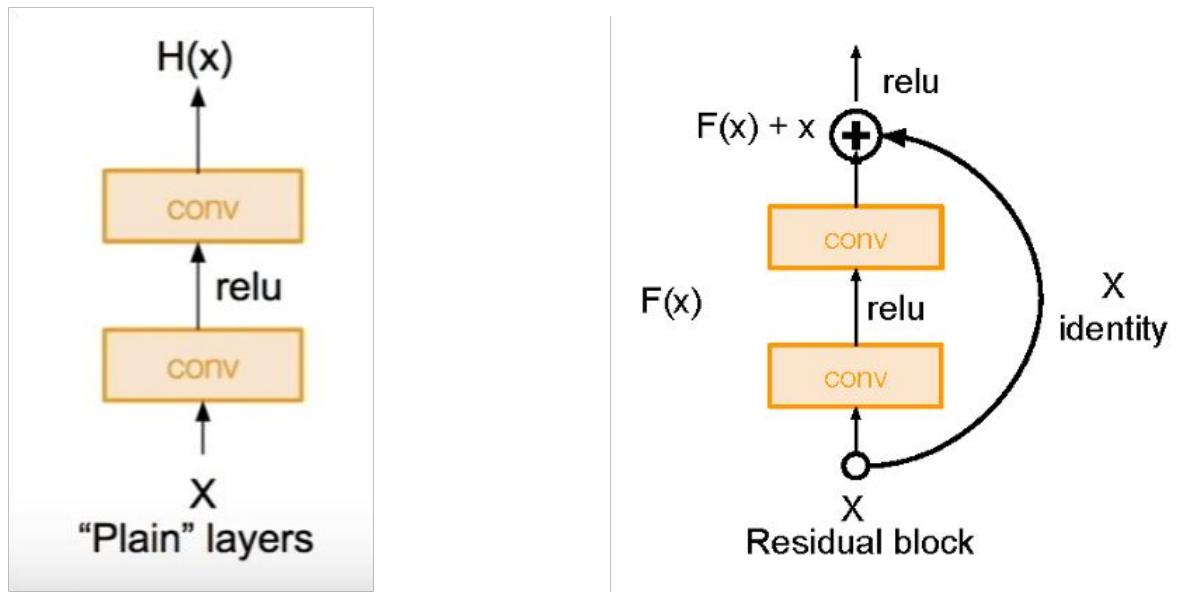
ResNet

- Điều gì xảy ra khi chúng ta tăng độ sâu mạng nơ-ron?
- Mạng 56 lớp làm việc kém hơn cả trên tập huấn luyện lẫn tập test (không phải do overfitting gây ra)

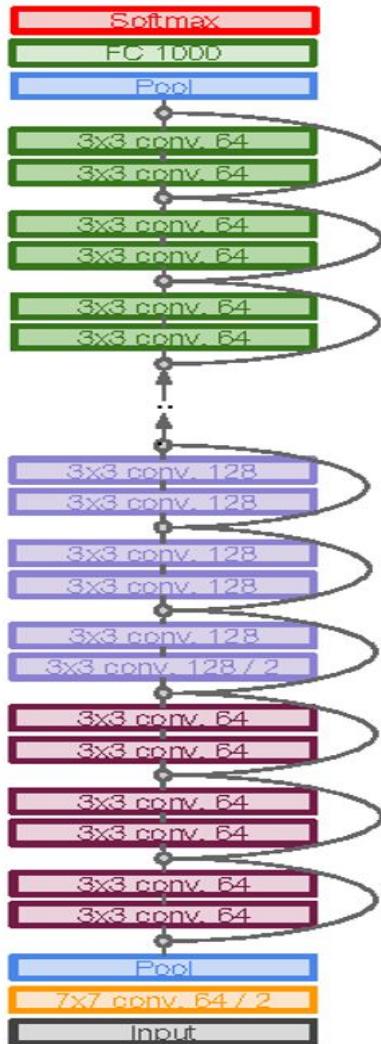


ResNet

- Giả thiết: Vấn đề ở chỗ bài toán tối ưu. Mạng rất sâu sẽ khó hơn để tối ưu.
- Giải pháp: Dùng các lớp mạng để học biểu diễn phần dư (sự sai khác giữa đầu ra và đầu vào) thay vì học trực tiếp đầu ra như trước.
- Học biểu diễn phần dư $F(x) = H(x) - x$ thay vì học trực tiếp $H(x)$

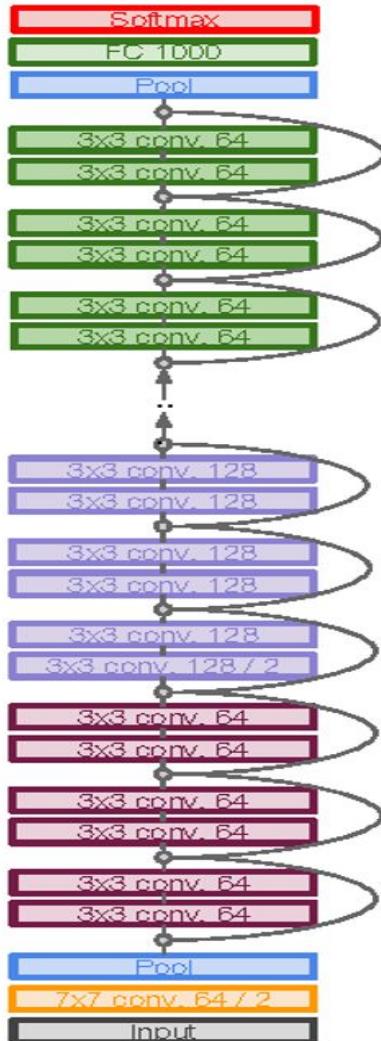


ResNet



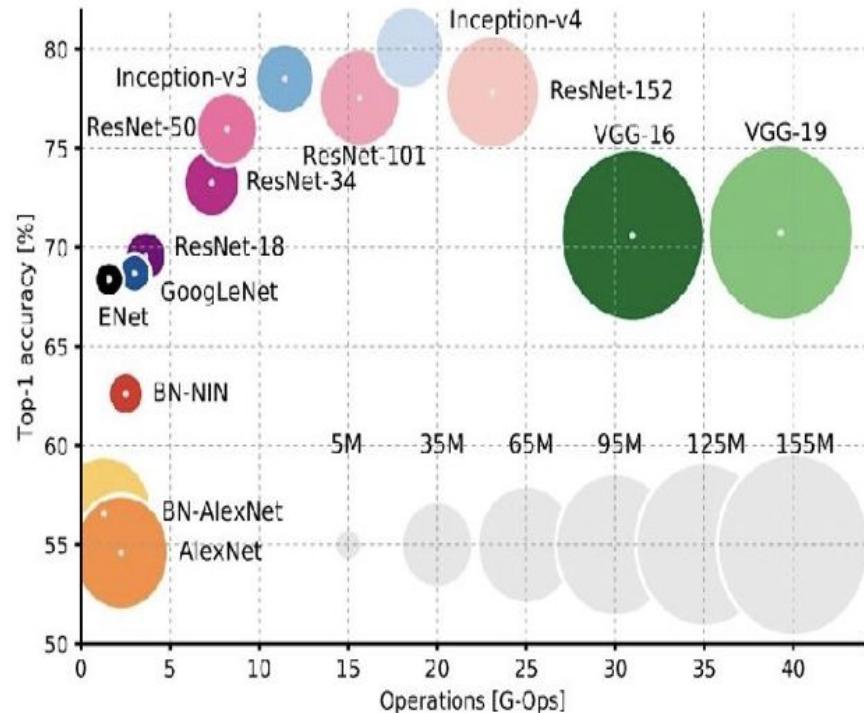
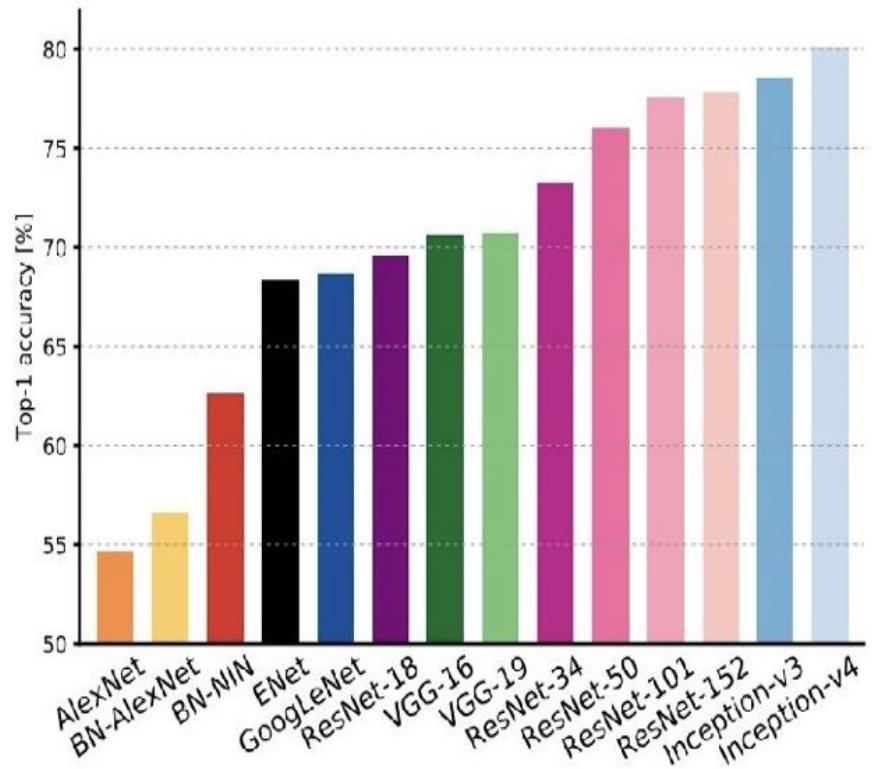
- Kiến trúc ResNet đầy đủ:
- Chồng các khối phần dư residual blocks
- Mỗi khối có hai lớp 3x3 conv
- Định kỳ tăng gấp đôi số lượng filter và giảm độ phân giải bằng conv bước nhảy stride 2
- Lớp conv phụ ở đầu mạng
- Không có lớp FC ở cuối (chỉ có lớp FC 1000 để xuất ra kết quả phân loại 1000 lớp)

ResNet



- Độ sâu của mạng khi tham gia cuộc thi ImageNet: 34, 50, 101, 152
- Với các mạng sâu (ResNet-50+), tác giả dùng lớp “bottleneck” để tăng hiệu quả (tương tự như GoogLeNet)

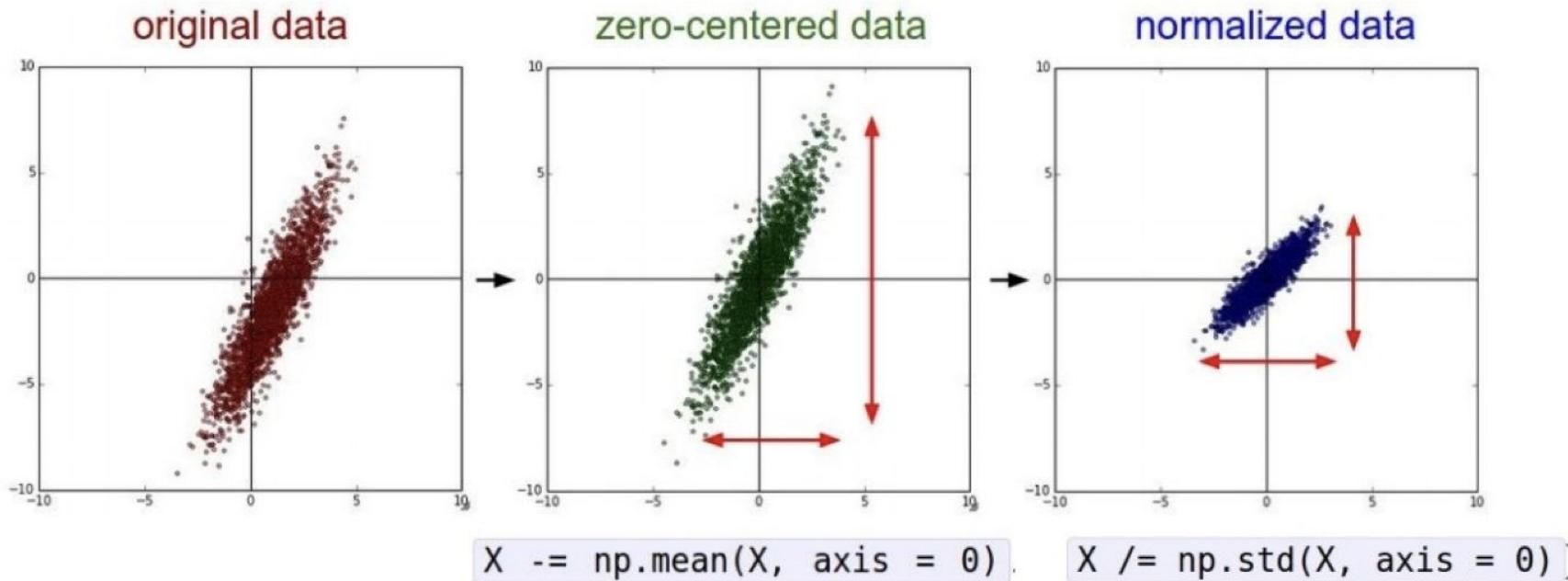
Accuracy Comparison



Tiền xử lý dữ liệu

Biến đổi phân phối dữ liệu về kỳ vọng bằng 0: trừ tất cả mẫu dữ liệu cho mẫu trung bình

Biến đổi phân phối dữ liệu về độ lệch chuẩn đơn vị



Tiền xử lý dữ liệu

Thường ít sử dụng PCA hoặc whitening

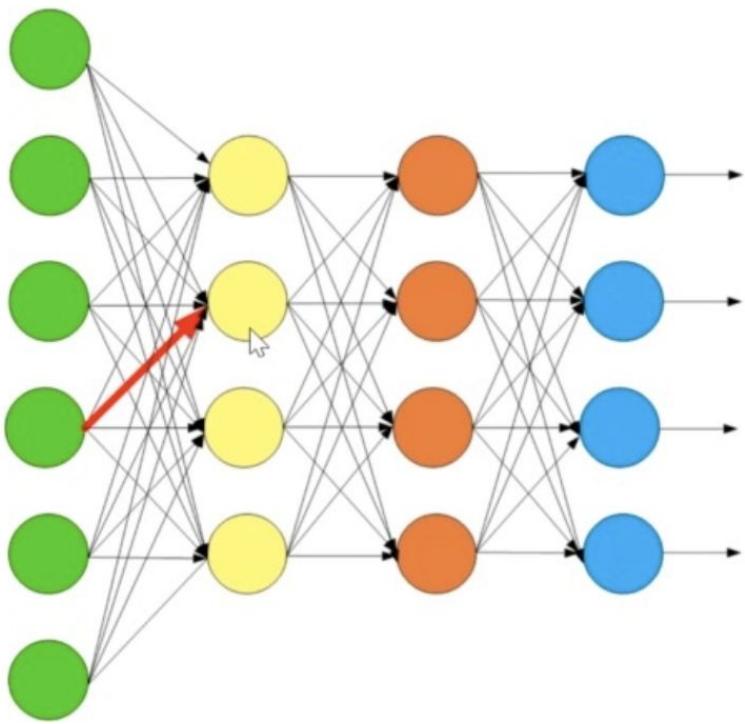
Ví dụ với bộ CIFAR10 với các ảnh kích thước 32x32x3

- Subtract the mean image (e.g. AlexNet)
(mean image = [32,32,3] array)
- Subtract per-channel mean (e.g. VGGNet)
(mean along each channel = 3 numbers)
- Subtract per-channel mean and
Divide by per-channel std (e.g. ResNet)
(mean along each channel = 3 numbers)

Batch normalization

- Đưa ra vào năm 2015 tại paper “Batch normalization: accelerating deep network by reducing internal covariate shift”
- Trước đó đã sử dụng một dạng batch norm là scale ảnh từ 0-255 về giá trị 0-1
- Nhằm giảm ảnh hưởng các điểm dữ liệu có giá trị lớn
- Được dùng để chuẩn hoá giá trị của hàm kích hoạt trước khi chuyển sang lớp sau
- Giảm số lượng epoch cần thiết để hội tụ khi training • Tăng độ ổn định learning rate có thể cao hơn
- Regularization (tránh overfitting)
- Thời gian huấn luyện chậm hơn

Batch normalization



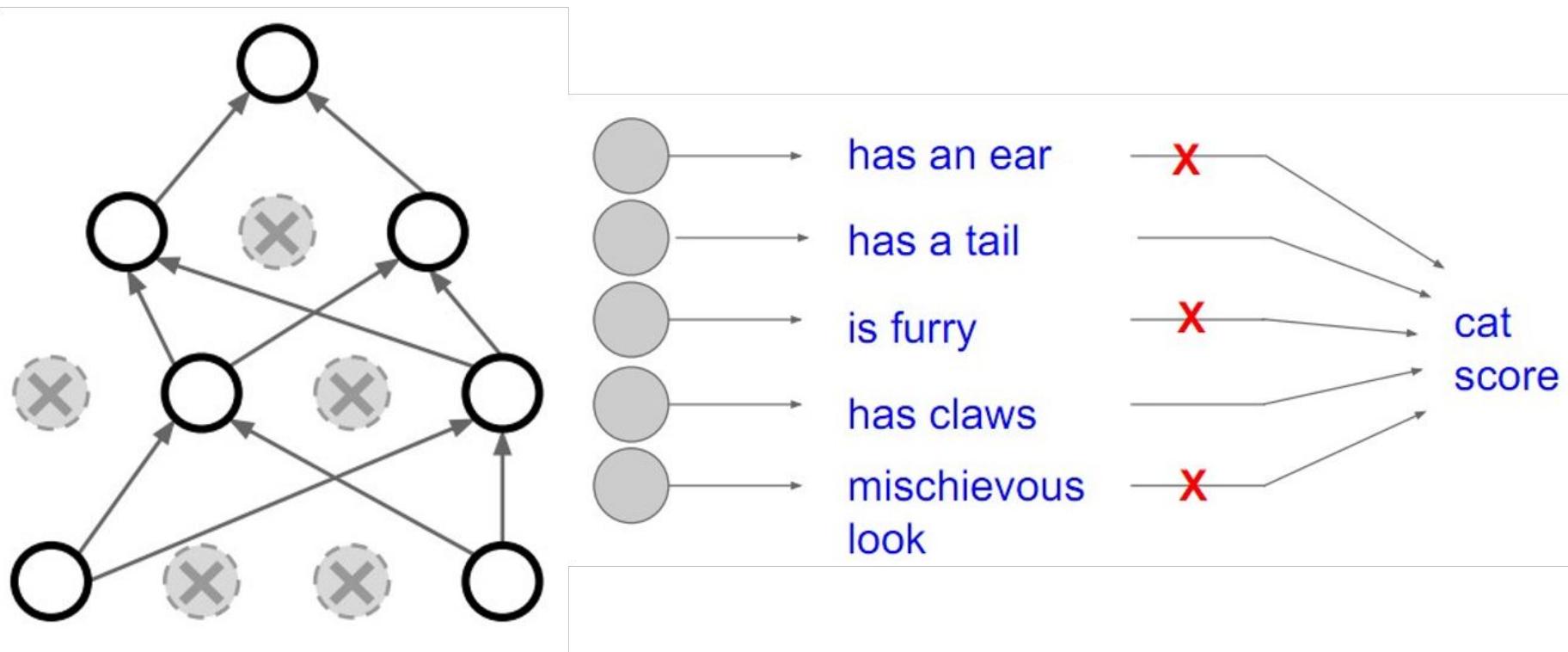
- Normalize output của activation function bằng cách nhân với 1 giá trị và cộng 1 giá trị khác

$$BN(\mathbf{x}) = \gamma \odot \frac{\mathbf{x} - \hat{\mu}}{\hat{\sigma}} + \beta$$

- Dựa trên các giá trị thống kê của minibatch hiện tại
- Giá trị đầu ra sau hàm batch norm sẽ có zero mean
- Weight không bị ảnh hưởng bởi các giá trị quá lớn

Dropout

- Ép mạng nơ-ron phải học biểu diễn dư thừa (redundant representation)



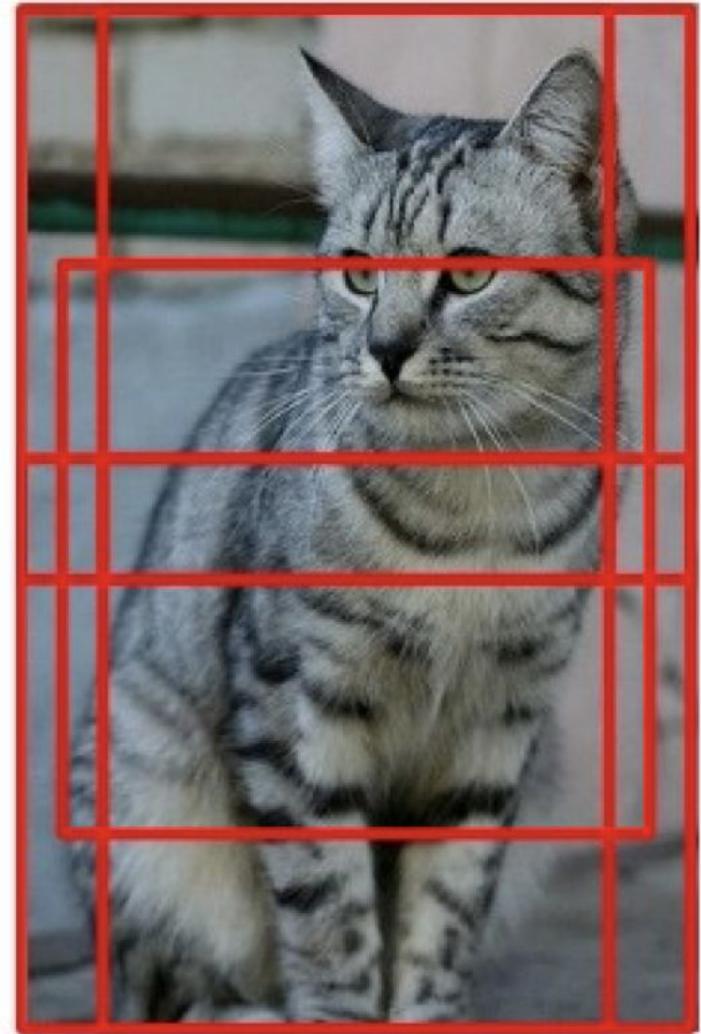
Làm giàu dữ liệu - Data Augmentation

- Flip ngang



Làm giàu dữ liệu - Data Augmentation

- Ví dụ ResNet:
- Chọn ngẫu nhiên L trong khoảng $[256, 480]$
- Resize ảnh để chiều nhỏ nhất bằng L
- Crop ngẫu nhiên vùng kích thước 224×224



Làm giàu dữ liệu - Data Augmentation

- Thay đổi màu sắc



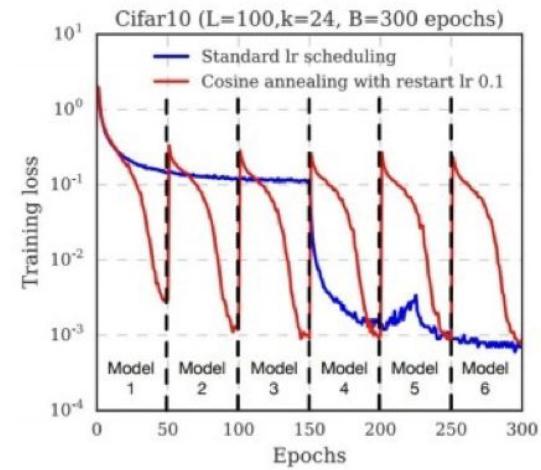
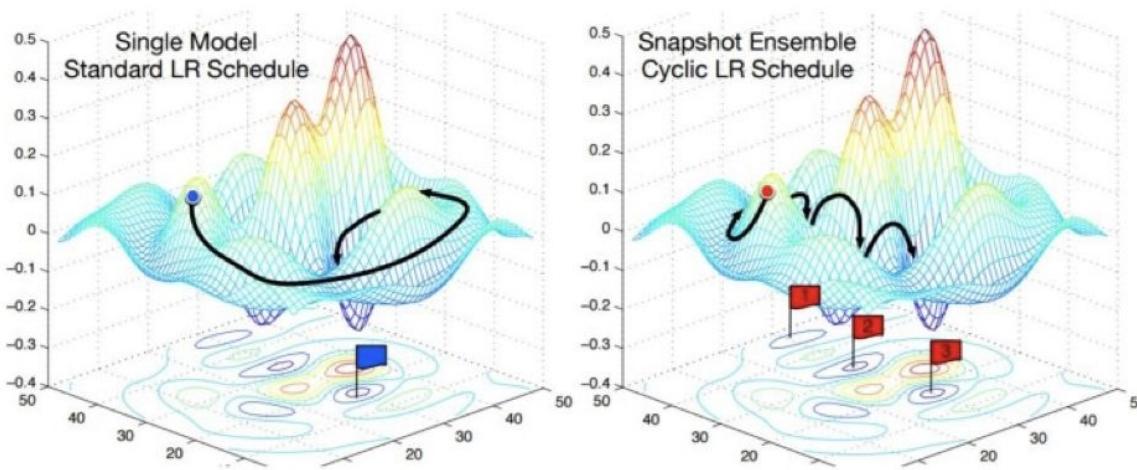
Làm giàu dữ liệu - Data Augmentation

Các phép biến đổi khác...

- Tịnh tiến
- Xoay ảnh
- stretching
- shearing
- lens distortions...

Model Ensembles

- Huấn luyện nhiều mô hình độc lập
- Khi test kết hợp kết quả nhiều mô hình
- Độ chính xác thường tăng 2%
- Thay vì huấn luyện nhiều mô hình độc lập, có thể dùng nhiều snapshot của cùng một mô hình trong quá trình huấn luyện



Transfer Learning - Ý tưởng

- Hiện tại các kiến trúc phổ biến đã được cài đặt sẵn trong các framework
 - Đã được thiết kế rất chi tiết, tỉ mỉ, dựa trên rất nhiều thí nghiệm với các siêu tham số khác nhau
 - Các model được đưa ra: nghiên cứu của các chuyên gia trong lĩnh vực này
 - Đã được huấn luyện trên bộ ImageNet
 - Để huấn luyện lại thường mất rất nhiều thời gian cho dù sử dụng state-of-the-art GPU
- Làm sao thay đổi các kiến trúc đã được đưa ra phục vụ cho bài toán của ta, tốt hơn so với việc xây dựng mạng CNN từ con số không (scratch)?

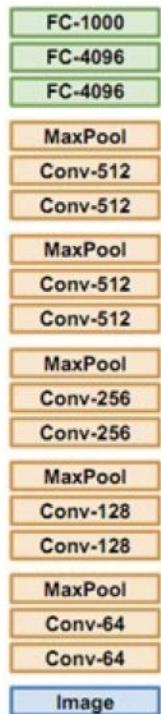
Transfer Learning - Ý tưởng

- Model càng lớn thì càng tốn thời gian huấn luyện
- Data càng nhiều thì NN lại càng tốn thời gian huấn luyện hơn nữa
 - Hiện nay data càng ngày càng nhiều
- Hiện nay có rất nhiều mạng pre-trained
- có sẵn
- Tận dụng tri thức của tác giả và việc lựa chọn các siêu tham số/ loss function/ optimizer như thế nào
- Các mạng CNN đã được huấn luyện ImageNet để nhận dạng đối tượng thường gặp hàng ngày
- Tuy nhiên ta lại muốn nhận dạng đối tượng khác
→ Nên tìm cách tận dụng các mạng có sẵn này

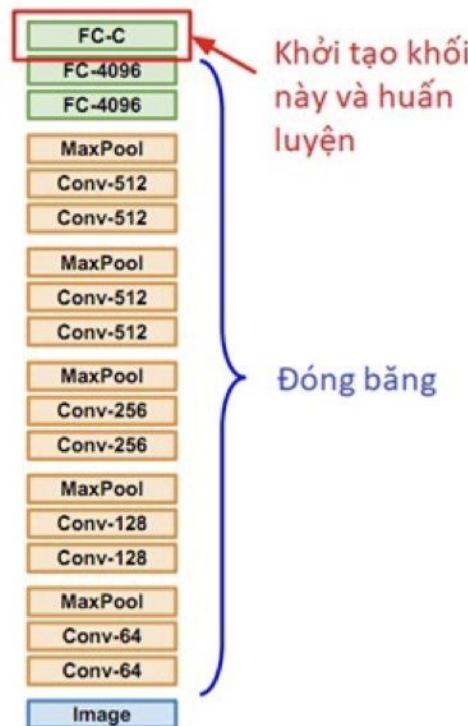
Transfer Learning

Huấn luyện mạng trên một tập dữ liệu lớn có sẵn, sau đó huấn luyện tiếp với tập dữ liệu của mình

1. Train on Imagenet



2. Small Dataset (C classes)



3. Bigger dataset

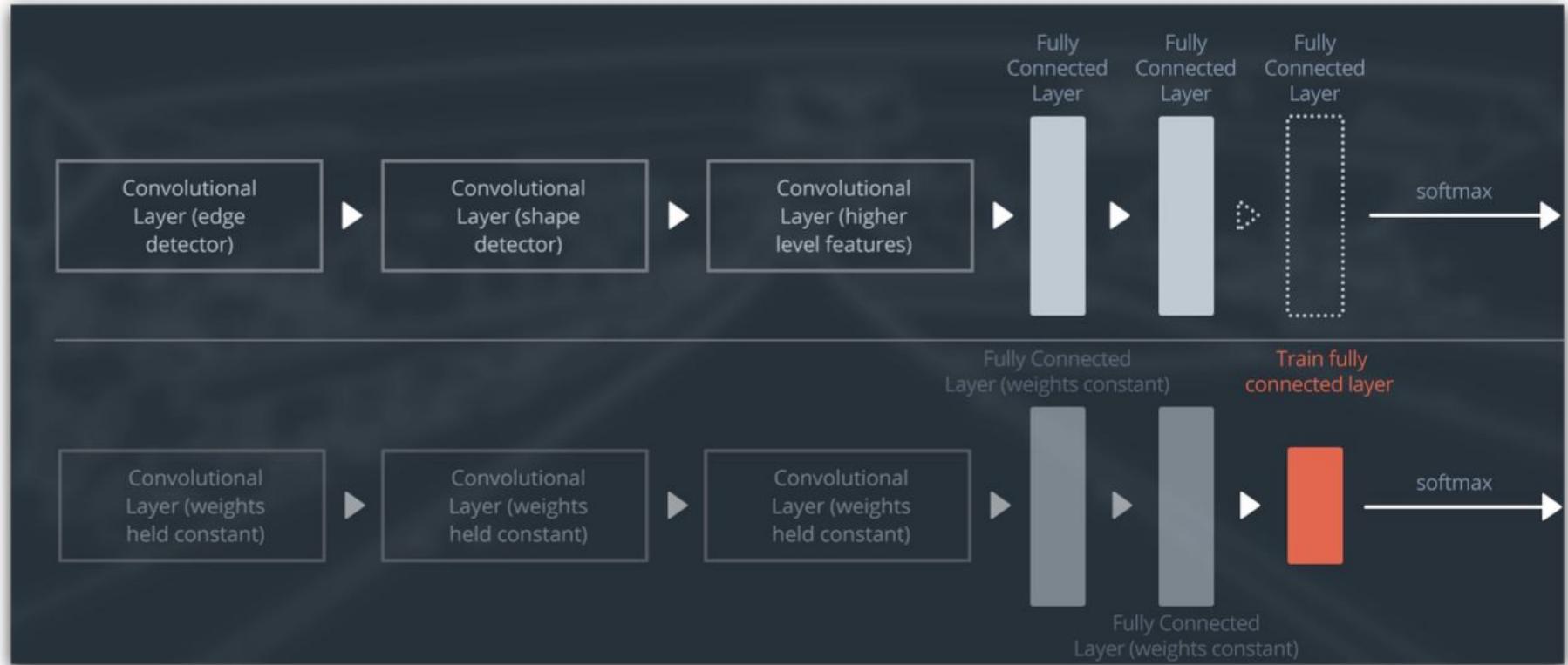


Transfer Learning - Các trường hợp

Small dataset, similar data

- Cắt lớp cuối của mạng
- Thêm fully connected layer tương đồng với số class của tập dữ liệu mới
- Khởi tạo ngẫu nhiên giá trị weight cho fully connected layer
- Đóng băng các weight của mạng pre-trained
- Đóng băng nhằm tránh overfitting khi huấn luyện các mạng có số lượng data thấp
- Vì data hai bộ dữ liệu là tương đồng, dữ liệu các layer trong mạng có thể tái sử dụng được
- Huấn luyện để cập nhật weight mới cho fully connected layer

Transfer Learning - Các trường hợp

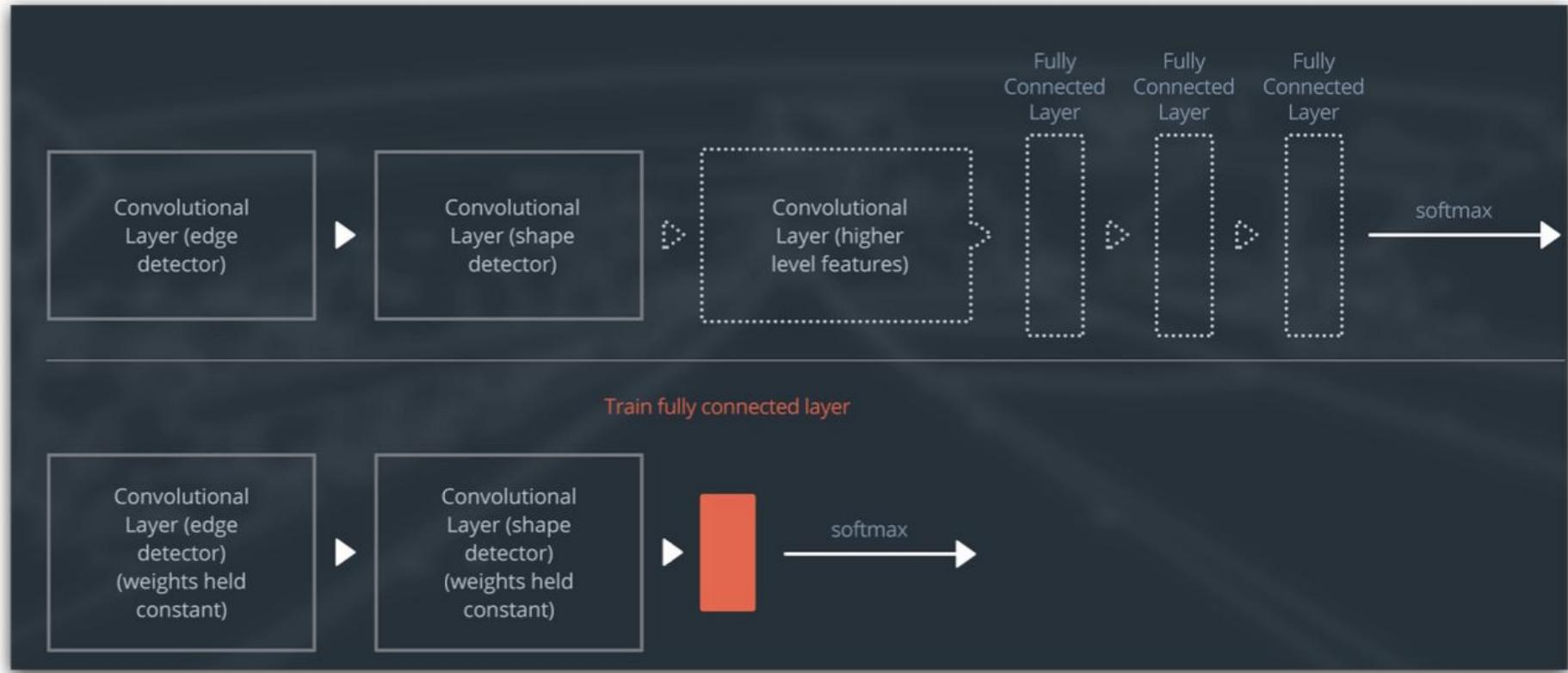


Transfer Learning - Các trường hợp

Small dataset, different data

- Cắt bỏ hầu hết các layer trong mạng pre-trained, giữ lại một số lớp đầu
- Do data mới và cũ có sự sai khác lớn, không chia sẻ cùng nhau các feature có độ phức tạp cao (higher level feature) nên mạng mới chỉ dùng các lower level features của pretrained
- Bổ sung thêm lớp fully connected layer tương ứng với số class của dữ liệu
- Khởi tạo ngẫu nhiên trọng số của fully connected layer
- Đóng băng các weights của mạng pre-trained Dataset small: đóng băng để tránh overfitting
- Huấn luyện mạng mới để update weight cho fully connected layer

Transfer Learning - Các trường hợp

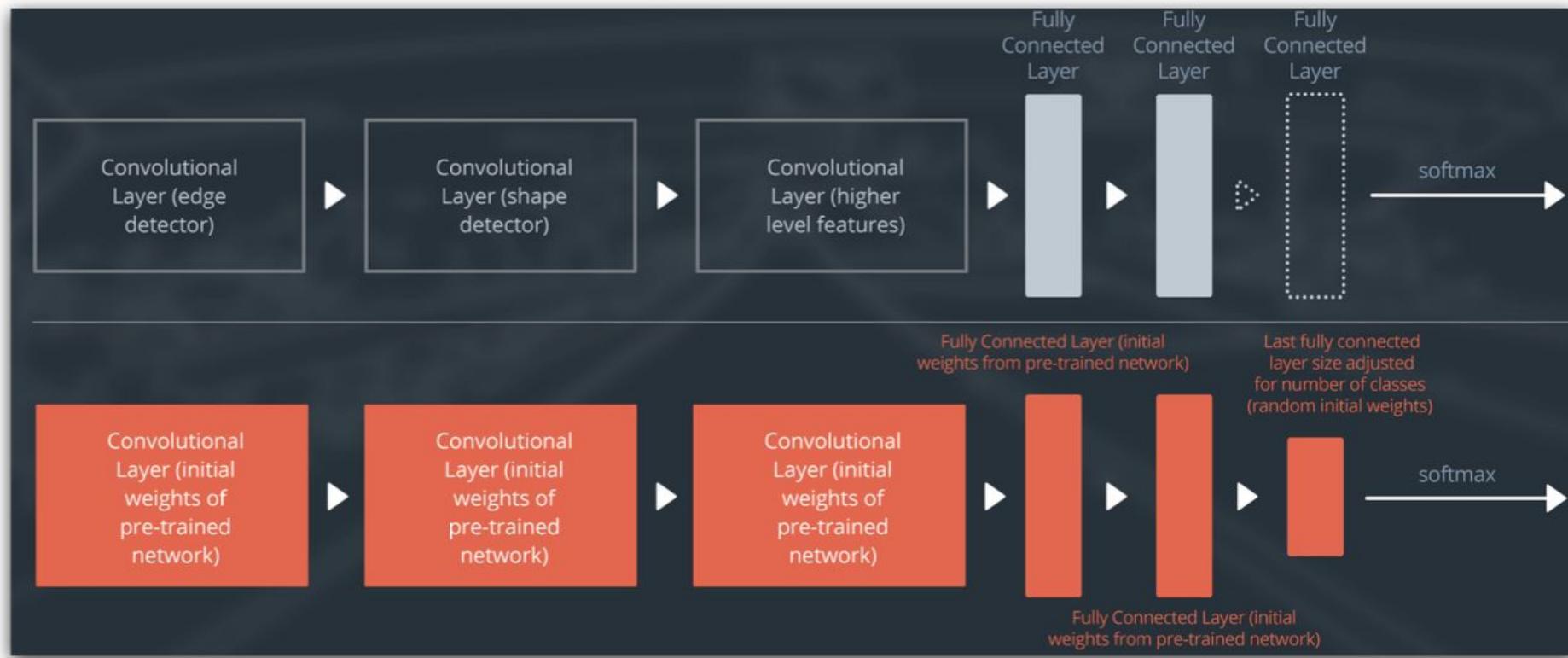


Transfer Learning - Các trường hợp

Large dataset, similar data

- Bỏ lớp fully connected layer cuối và thêm lớp fully connected layer mới phù hợp với số class của dataset
- Vì data cũ và mới có điểm chung nên có thể giữ lại hầu hết các layer của mạng pre-trained
- Khởi tạo ngẫu nhiên trọng số cho lớp fully connected layer
- Khởi tạo giá trị weight của các lớp còn lại dựa trên pre-trained
- Huấn luyện lại toàn bộ mạng
- Vì số lượng data lớn, khó có khả năng bị overfit nên có thể train lại toàn bộ mạng

Transfer Learning - Các trường hợp



Transfer Learning - Các trường hợp

- **Large dataset, different data**
- Bỏ lớp fully connected layer và thay thế bởi lớp fully connected layer phù hợp với số class của dữ liệu mới
- 2 cách xử lý
 - Giống large dataset, similar data
 - Có thể làm cho việc huấn luyện nhanh hơn (khởi tạo weight bằng giá trị pretrained)
 - Huấn luyện mạng từ đầu với các trọng số ngẫu nhiên
 - Nếu khởi tạo giá trị như pre-trained đem lại chất lượng thấp --> train lại tất cả từ đầu bằng các hệ số ngẫu nh

Các bài toán thị giác máy

Phân loại hình ảnh (image classification): Dự đoán nhãn của một đối tượng trong một hình ảnh.

- Input: Một hình ảnh với một đối tượng, chẳng hạn như một bức ảnh.
- Output: Nhãn lớp (ví dụ: một hoặc nhiều số nguyên được ánh xạ tới nhãn lớp).

Định vị đối tượng (object localization): Xác định vị trí hiện diện của các đối tượng trong ảnh và cho biết vị trí của chúng bằng bounding box.

- Input: Một hình ảnh có một hoặc nhiều đối tượng, chẳng hạn như một bức ảnh.
- Output: Một hoặc nhiều bounding box được xác định bởi tọa độ tâm, chiều rộng và chiều cao.

Phát hiện đối tượng (object detection): Xác định vị trí hiện diện của các đối tượng trong bounding box và nhãn của các đối tượng nằm trong một hình ảnh.

- Input: Một hình ảnh có một hoặc nhiều đối tượng, chẳng hạn như một bức ảnh.
- Output: Một hoặc nhiều bounding box và nhãn cho mỗi bounding box.

Phân đoạn ảnh (image segmentation): các đối tượng được nhận dạng bằng cách làm nổi bật các pixel cụ thể của đối tượng thay vì bounding box

Các bài toán thị giác máy

Semantic Segmentation



No objects, just pixels

Classification + Localization



Single Object

Object Detection



Multiple Object

Instance Segmentation



This image is CC0 public domain

Tổng kết buổi học

- Khái niệm về Trí tuệ nhân tạo và ứng dụng
- Phân biệt Artificial Intelligence, Machine Learning, Deep Learning
- Quy trình huấn luyện một mô hình và các khái niệm liên quan
- Kiến thức toán học nền tảng
- Kiến thức lập trình nền tảng

THANK YOU !

COLE.VN
Connecting knowledge



www.cole.vn