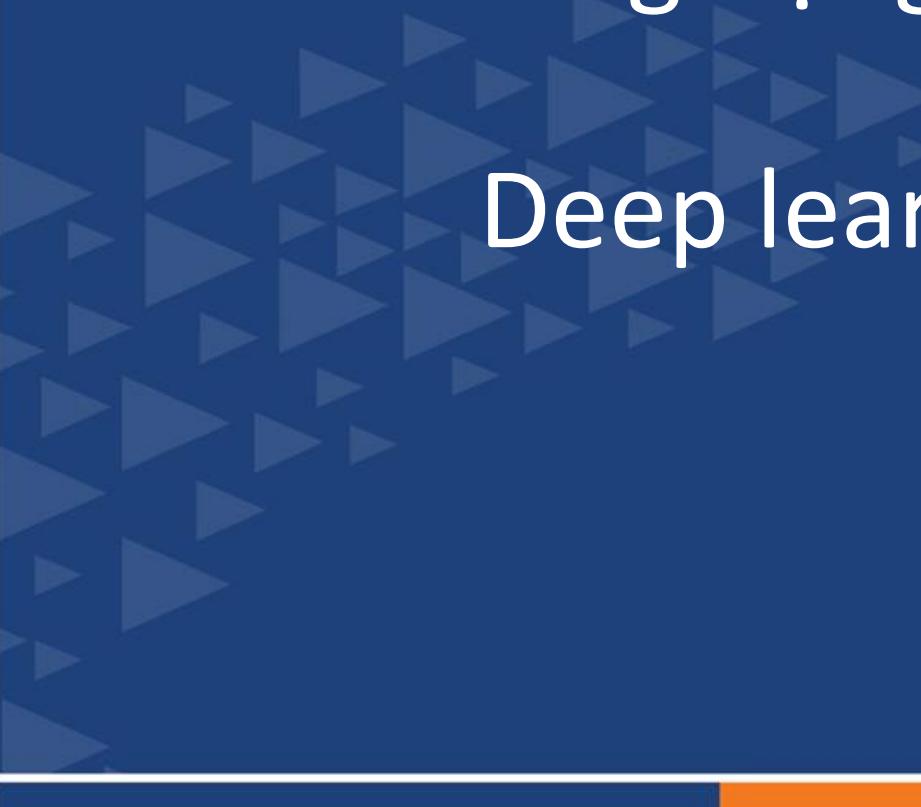


Bài 12: Mạng nơ-ron và Học sâu ứng dụng trong Thị giác máy tính



A decorative pattern of blue triangles of varying sizes and shades, arranged in a staggered, radiating pattern from the bottom left corner towards the center.

Deep learning in Computer Vision

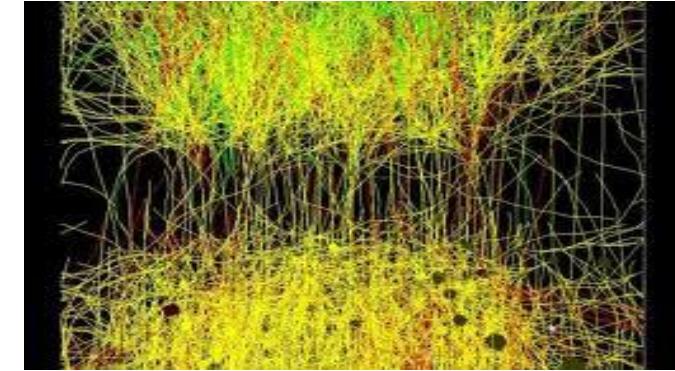
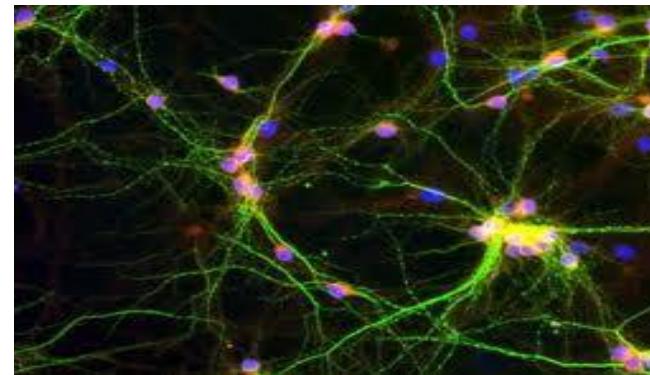
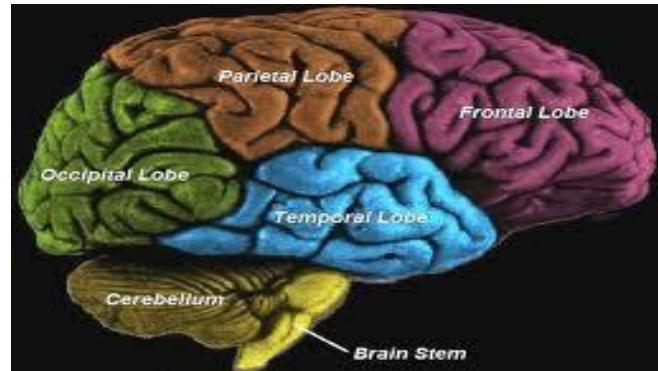
AI Academy Vietnam

Nội dung buổi học

1. Giới thiệu về mạng nơ-ron nhân tạo (ANN)
2. Các mô hình mạng nơ-ron
3. Giải thuật lan truyền ngược (Backpropagation)
4. Mạng nơ-ron học sâu (Deep Neural Network) và mạng tích chập CNN
5. Ứng dụng trong phát hiện/nhận dạng ảnh

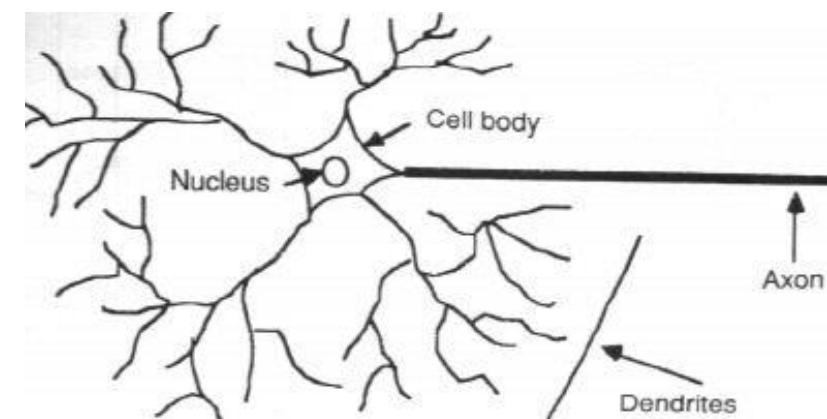
Giới thiệu về mạng nơ-ron nhân tạo Artificial Neural Network (ANN)

Mạng nơ-ron sinh học



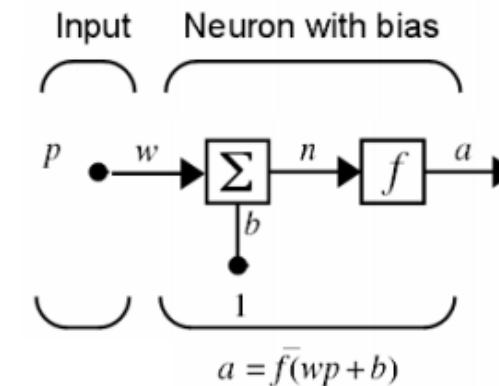
- Hàng trăm tỉ nơ-ron
- Mỗi nơ-ron có hàng ngàn kết nối
- Tổ chức thành lớp (layers)

Q: Quá trình học của con người?



Artificial Neural Networks (ANN)

- ANN là một mô hình mạng chứa nhiều đơn vị xử lý, mô phỏng quá trình học tập và tính toán của bộ não sinh học.
- Mỗi đơn vị xử lý gọi là một nơ-ron nhân tạo. Mỗi nơ-ron nhân tạo giả lập một nơ-ron sinh học, gồm:
 - 1 đơn vị tính toán (tổng)
 - 1 ngưỡng kích hoạt (bias b) và
 - 1 hàm kích hoạt (hay hàm truyền, transfer function) đặc trưng cho tính chất của nơ-ron.
- Các nơ-ron nhân tạo được liên kết với nhau bằng các kết nối. Mỗi kết nối có một trọng số kết nối (weights), đặc trưng cho khả năng nhớ của ANN.



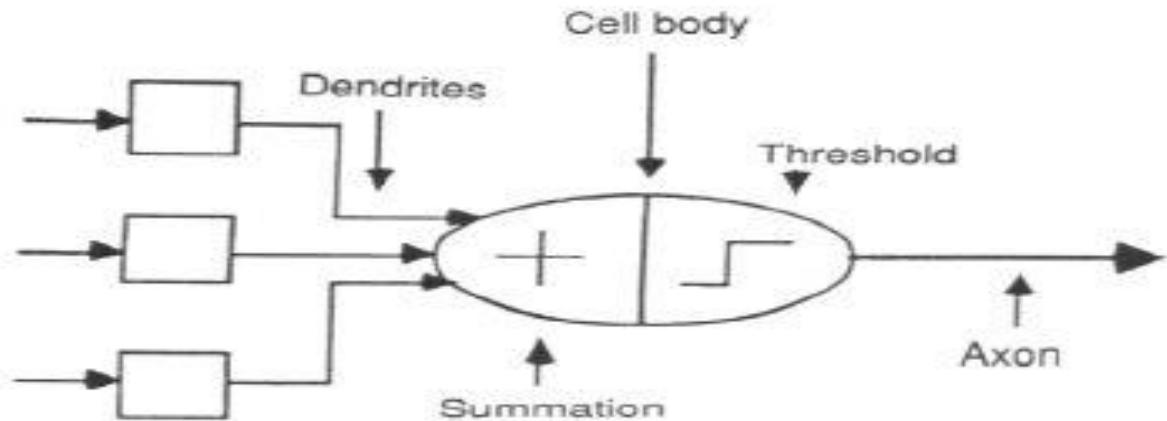
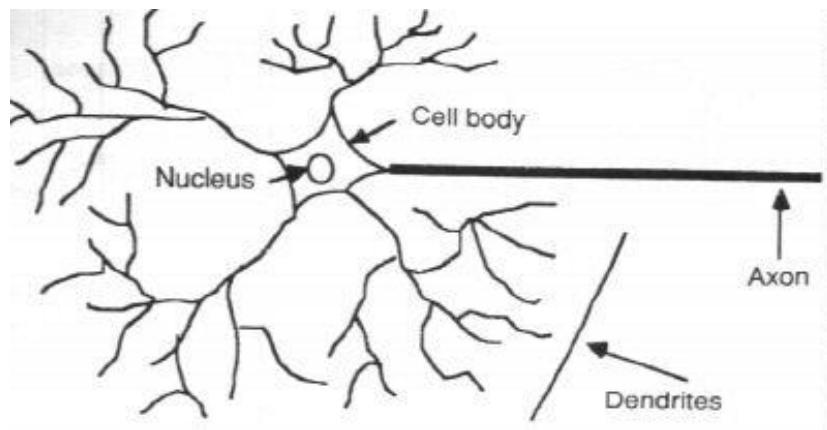
Artificial Neural Networks (ANN)

- ANN học từ dữ liệu thông qua hiệu chỉnh các trọng số kết nối giữa các nơ-ron
- Quá trình huấn luyện ANN là 1 quá trình điều chỉnh các ngưỡng kích hoạt và các trọng số kết nối, dựa trên dữ liệu học.

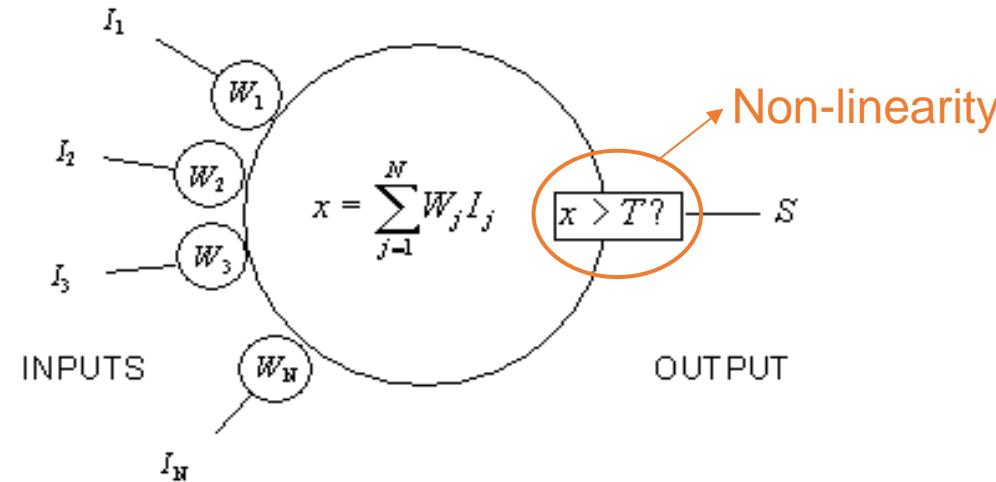
=>Nhiệm vụ: Xây dựng một cấu trúc ANN và huấn luyện nó dựa trên dữ liệu thu thập được (gọi là tập dữ liệu học).

Neuron Model

- Mô hình của 1 nơ-ron:



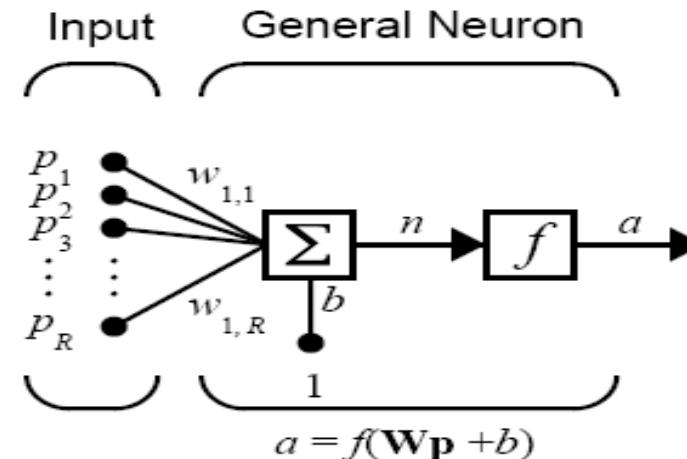
Biểu diễn toán học của 1 nơ-ron



- Tính toán tổng trọng số của các tín hiệu vào và so sánh với 1 ngưỡng
- Nếu tổng lớn hơn ngưỡng thì trả ra đầu ra là 1, trái lại là -1.

Hàm kích hoạt và hàm truyền

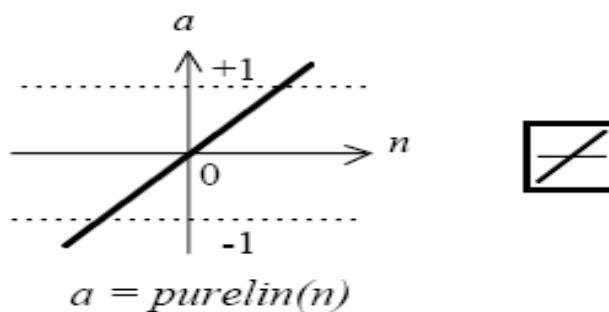
- Activation function



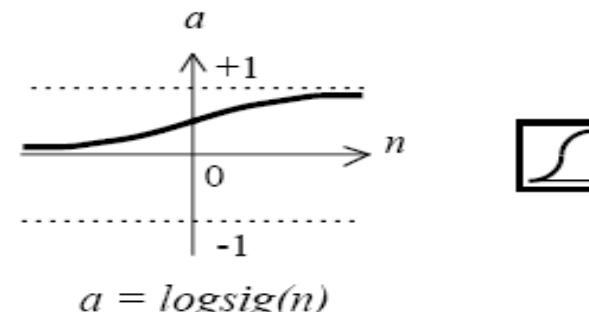
Where

R = number of elements in input vector

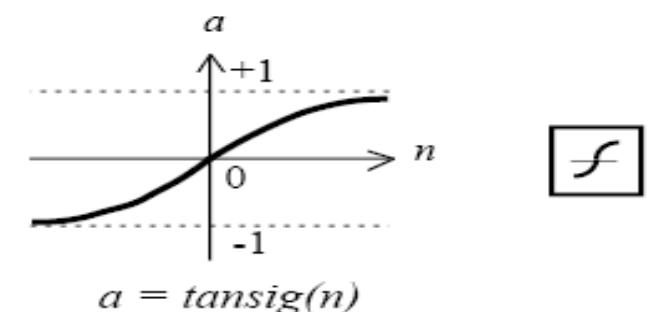
- Hàm truyền (Transfer functions):



Linear Transfer Function

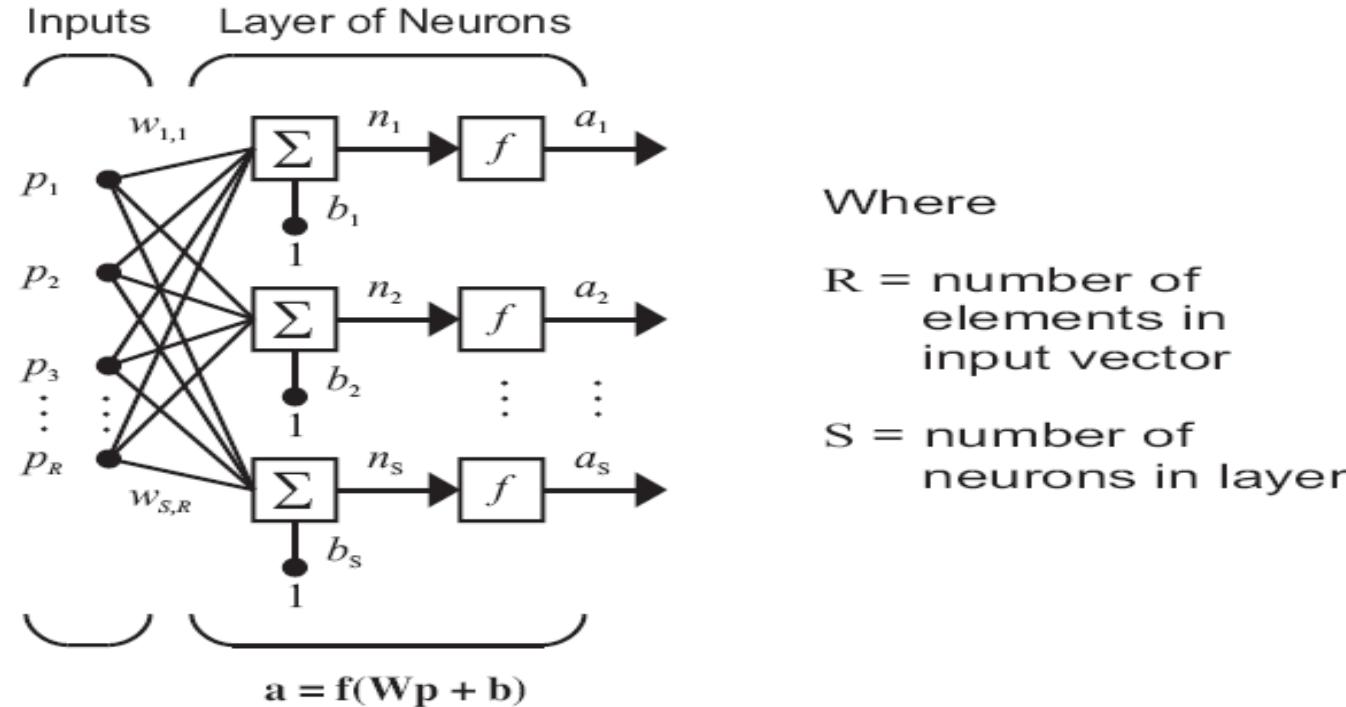


Log-Sigmoid Transfer Function



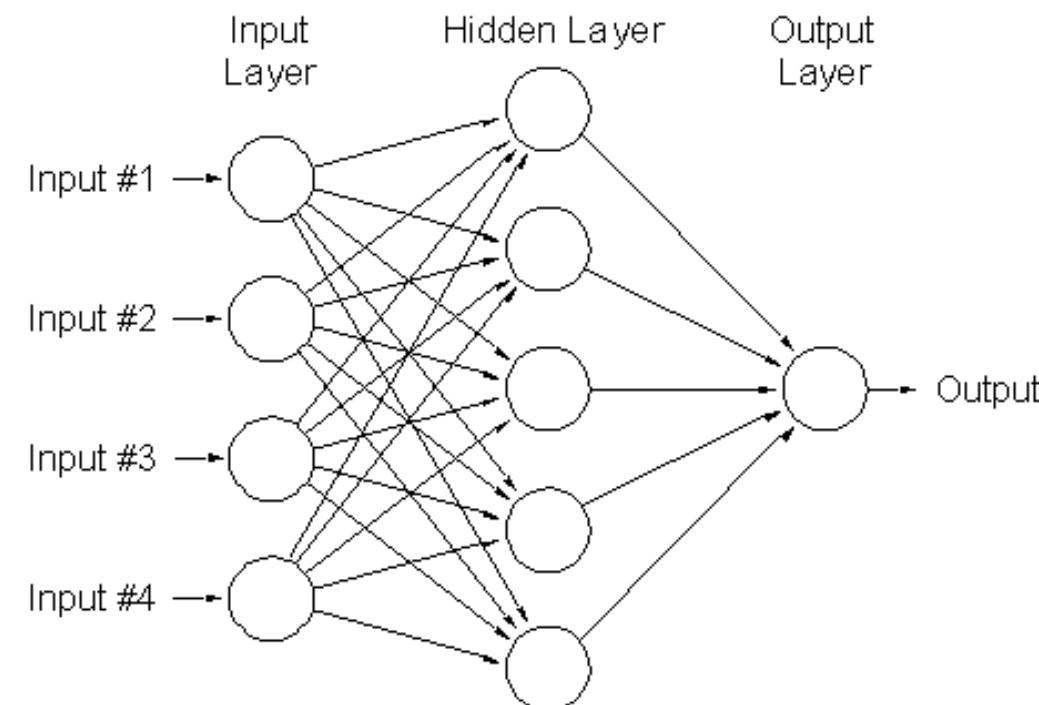
Tan-Sigmoid Transfer Function

Kiến trúc mạng (Network architecture)



- Mạng nơ-ron là một hệ thống gồm nhiều nơ-ron đơn lẻ kết nối với nhau, để thực hiện một chức năng tính toán nào đó.
- 2 hoặc nhiều neurons có thể kết hợp tạo thành 1 lớp (layer)
- 1 ANN có thể có 1 hoặc nhiều hơn 1 lớp (VD: mạng ANN 1 lớp)

- Tính năng của mạng phục thuộc vào:
 - Cấu trúc mạng
 - các trọng số kết nối và quá trình tính toán tại các nơ-ron đơn lẻ (hàm truyền).
- Mạng nơ-ron có thể học từ dữ liệu mẫu và tổng quát hóa dựa trên các mẫu đã học (mạng có khả năng nhớ)



Ví dụ hoạt động của ANN

- Xét bài toán: Cho 2 vector x và y. Xác định $y=f(x)=?$

x	1	2	3	4	5
y	2	4	6	8	10

$$y = f(x) = 2x$$

Cho $x = 7 \rightarrow y = 14$

- Thay đổi dữ liệu:

x	1	2	3	4	5
y	2.5	4.1	6	8.33	10.1

$$y = f(x) = ?$$

Cho $x = 7 \rightarrow y = ? \rightarrow$ khó xác định

- Ứng dụng ANN xác định $y=f(x)$:

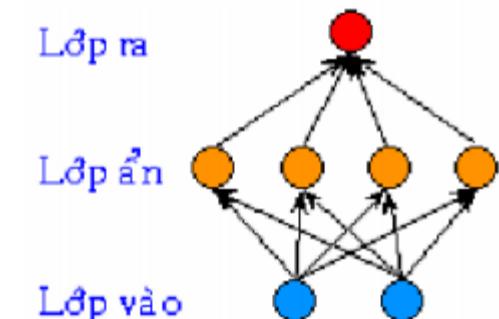
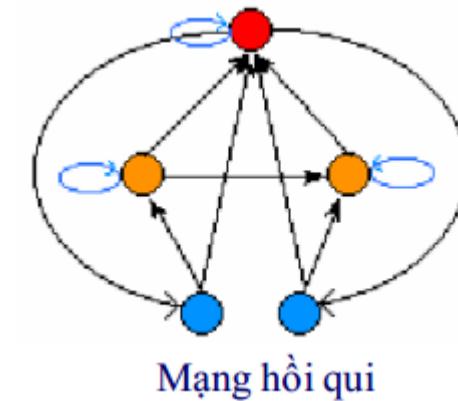
- Thu thập dữ liệu 2 vectors x và y (dữ liệu nhiều, độ chính xác lớn)
- Xây dựng mạng nơ-ron và huấn luyện mạng dựa trên dữ liệu thu thập
- Khi đó ANN hoạt động như 1 hàm: $y=f_{ANN}(x)$
- Với 1 giá trị x_n ta có $y_n = f_{ANN}(x_n) \sim f(x_n)$

☞ ANN có thể được huấn luyện để xấp xỉ một quan hệ phi tuyến bất kỳ

Các mô hình mạng nơ-ron

Phân loại mạng ANN

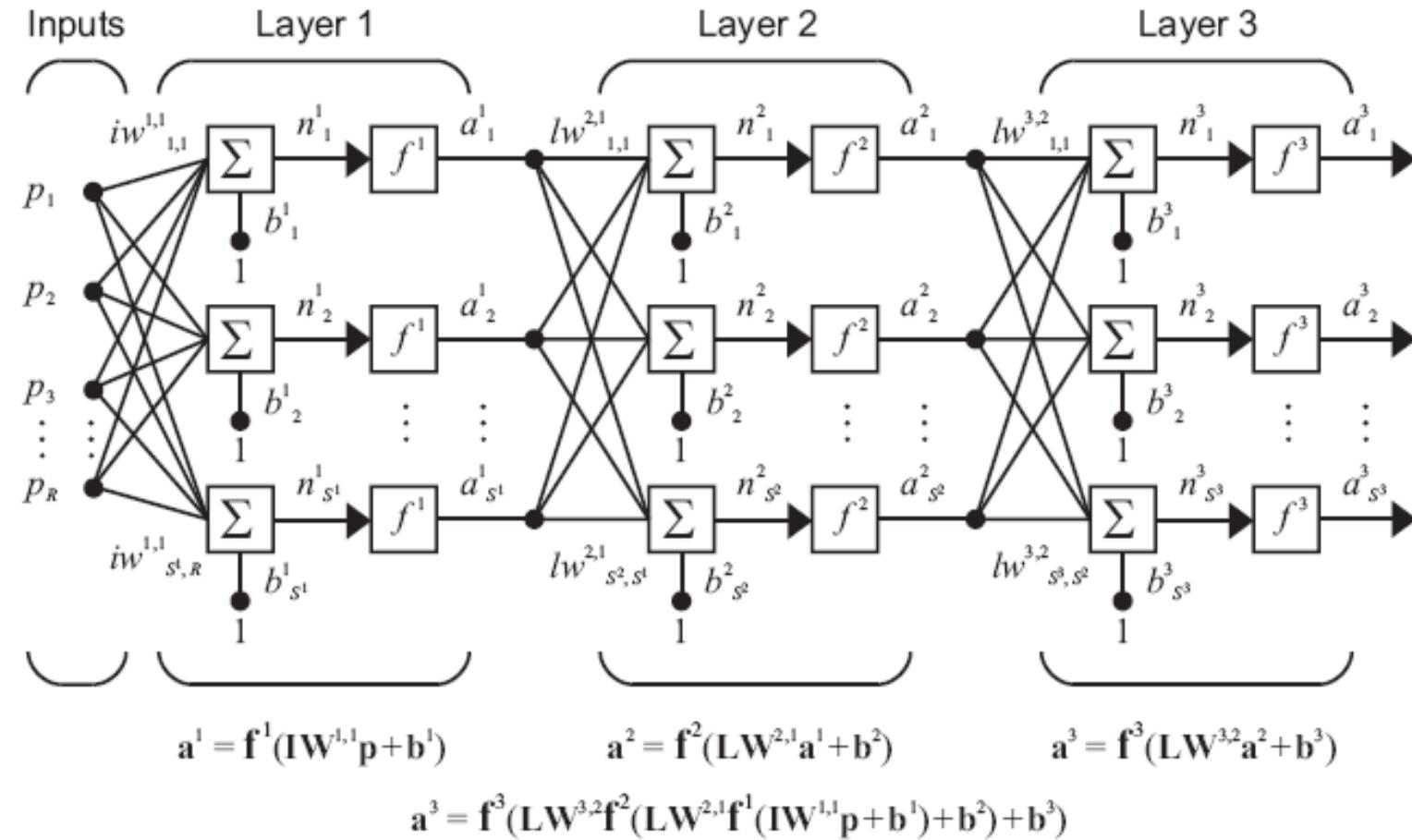
- Dựa theo kiểu kết nối các nơ-ron: có 2 loại
 - Mạng truyền thẳng (feedforward NN, **FNN**): Kết nối không tạo thành chu trình
 - Mạng hồi qui (recurrent NN, **RNN**): Kết nối tạo thành các chu trình
- Dựa vào số lớp: có 2 loại
 - Mạng 1 lớp
 - Mạng nhiều lớp: Gồm lớp vào, các lớp ẩn và lớp ra.
 - Lớp vào không tính số lớp.
 - Không kết nối trên cùng lớp; không nhảy lớp; không kết nối từ lớp sau ngược lại lớp trước.



Mạng truyền thẳng

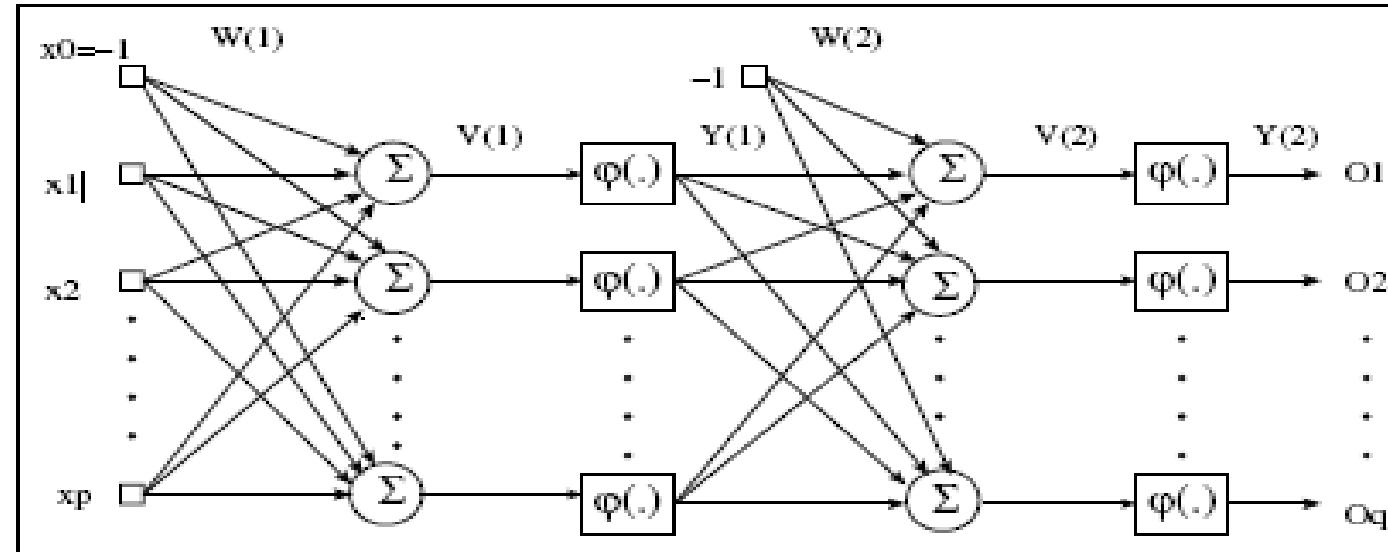
Mạng nhiều lớp truyền thẳng FNN

- Multilayer Perceptron (MLP)



- Dạng thức mô hình tính toán chung là nhau cho các layers

ANN lan truyền thuận



- Mạng nơ-ron có thể được tổ chức thành nhiều lớp, với ngõ ra của lớp trước là ngõ vào của lớp sau.
- Quá trình tính toán trên mạng được thực hiện lần lượt trên từng lớp.

ANN lan truyền thuận

- Bias Nodes
 - 1 node được thêm vào mỗi lớp với đầu ra là 1 giá trị hằng
- Lan truyền thuận:
 - Tính toán các giá trị cho mỗi nơ-ron trên mỗi lớp từ đầu vào đến đầu ra
 - Với mỗi nơ-ron:
 - Tính trung bình trọng số các đầu vào
 - Tính toán hàm kích hoạt (activation function)

Ví dụ : Cho mạng 3 lớp như hình vẽ, tính giá trị ngõ ra, với:

- Lớp vào: $\mathbf{p} = [2, 1.5, 3]^T$

- Lớp ẩn 1:

$$\mathbf{b}^1 = [0.5, 0.6]^T$$

$$\mathbf{W}^1 = [0.1, 0.2, 0.3; 0.5, 0.4, 0.6]$$

f^1 : hàm purelin

- Lớp ẩn 2:

$$\mathbf{b}^2 = [0.5, 0.6]^T$$

$$\mathbf{W}^2 = [0.5, 0.2; 0.1, 0.6]; f^2: \text{hàm Logsig}$$

- Lớp ra: $b^3 = 0.2$; $\mathbf{W}^3 = [0.4, 0.2]$; $f^3: \text{hàm Tangsig}$

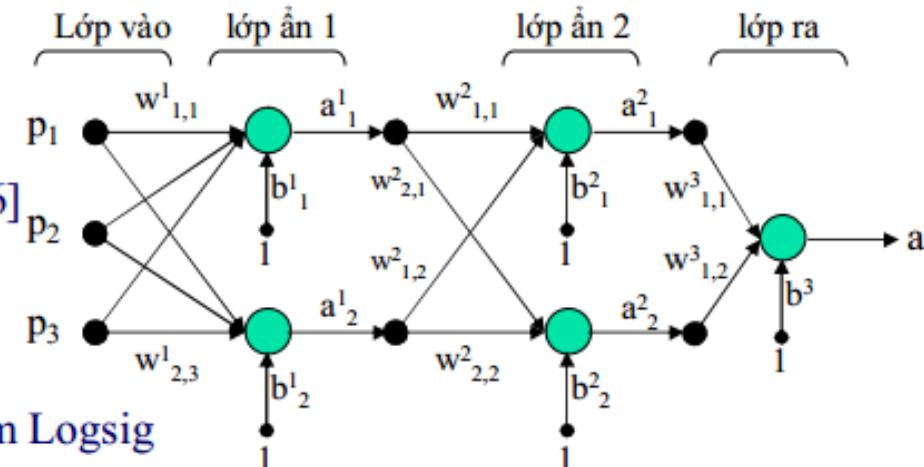
- Ngõ ra lớp ẩn 1: Giống Ví dụ ở phần mạng nơ-ron 1 lớp. Ta có $a^1_1=1.9$ và $a^1_2=4$

- Ngõ ra lớp ẩn 2: ngõ vào lớp 2 chính là ngõ ra lớp 1

$$\mathbf{W}^2 \mathbf{p}^2 + \mathbf{b}^2 = [0.5, 0.2; 0.1, 0.6] * [1.9, 4]^T + [0.5, 0.6]^T = [2.25, 3.19]^T$$

$$\mathbf{a}^2 = f^2(\mathbf{W}^2 \mathbf{p}^2 + \mathbf{b}^2) = [f^2(2.25), f^2(3.19)]^T = \left[\frac{1}{1 + e^{-2.25}} \quad \frac{1}{1 + e^{-3.19}} \right]^T = [.9047, .9605]^T$$

hay: $a^2_1 = 0.9047$ và $a^2_2 = 0.9605$

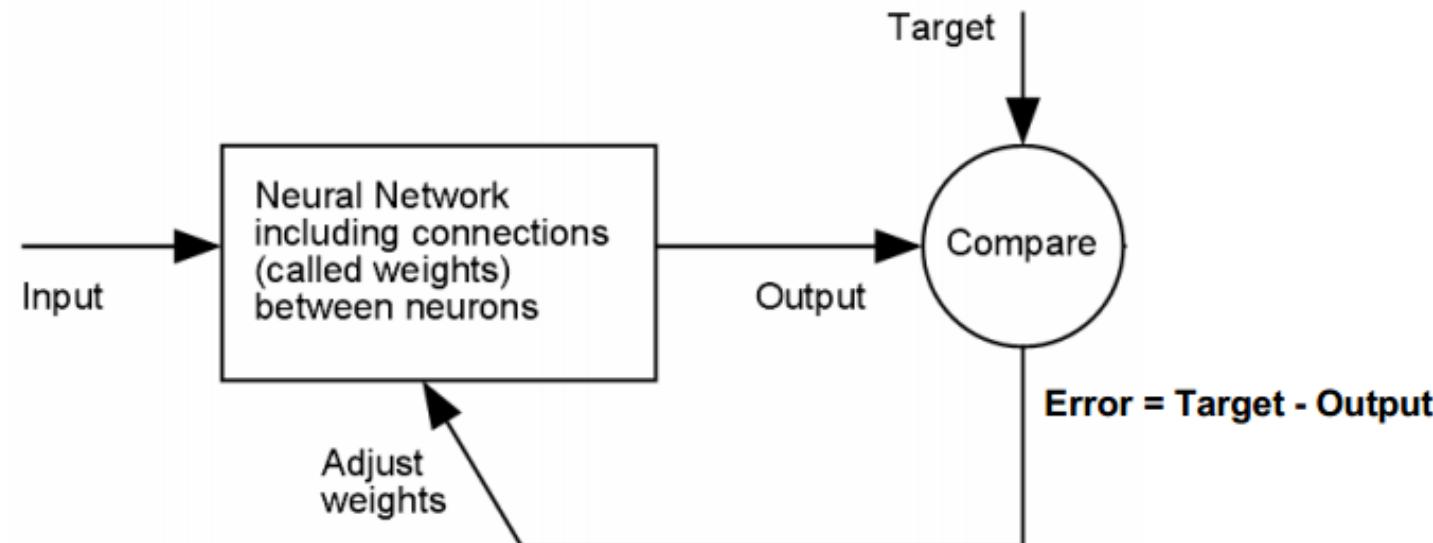


Giải thuật lan truyền ngược

Quá trình học của ANN

- Quá trình huấn luyện ANN có thể mô tả như sau:

Dựa trên sự sai biệt (*Error*) giữa ngõ ra mong muốn (*Target*) và ngõ ra thực tế của ANN (*Output*), giải thuật huấn luyện sẽ điều chỉnh các trọng số kết nối của ANN, sao cho: $\min\{\text{Error}\}$



Giải thuật huấn luyện ANN

- Giải thuật truyền ngược cập nhật các trọng số theo nguyên tắc:

$$w_{ij}(k+1) = w_{ij}(k) + \eta g(k)$$

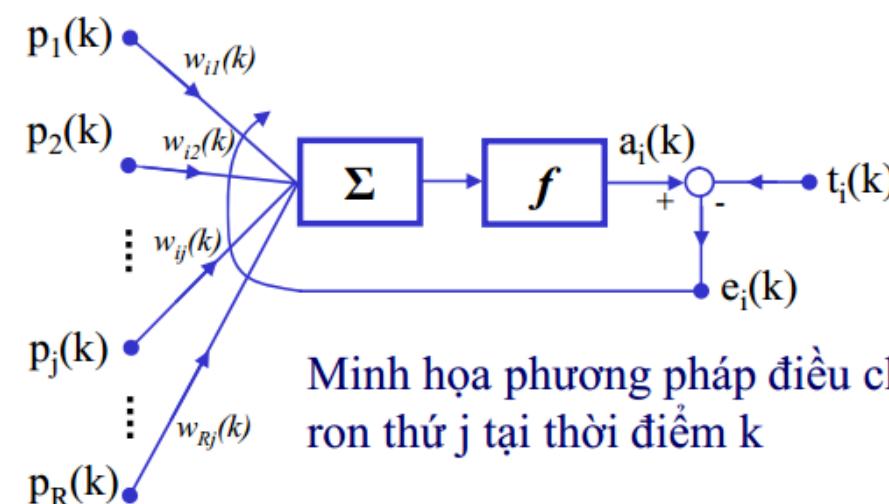
trong đó:

- $w_{ij}(k)$ là trọng số của kết nối từ nơ-ron j đến nơ-ron i , ở thời điểm hiện tại (vòng lặp k)
- η là tốc độ học (learning rate, $0 < \eta \leq 1$)
- $g(k)$ là gradient hiện tại

- Có nhiều phương pháp xác định gradient $g(k)$, dẫn tới có nhiều giải thuật truyền ngược cải tiến.

Giải thuật huấn luyện ANN

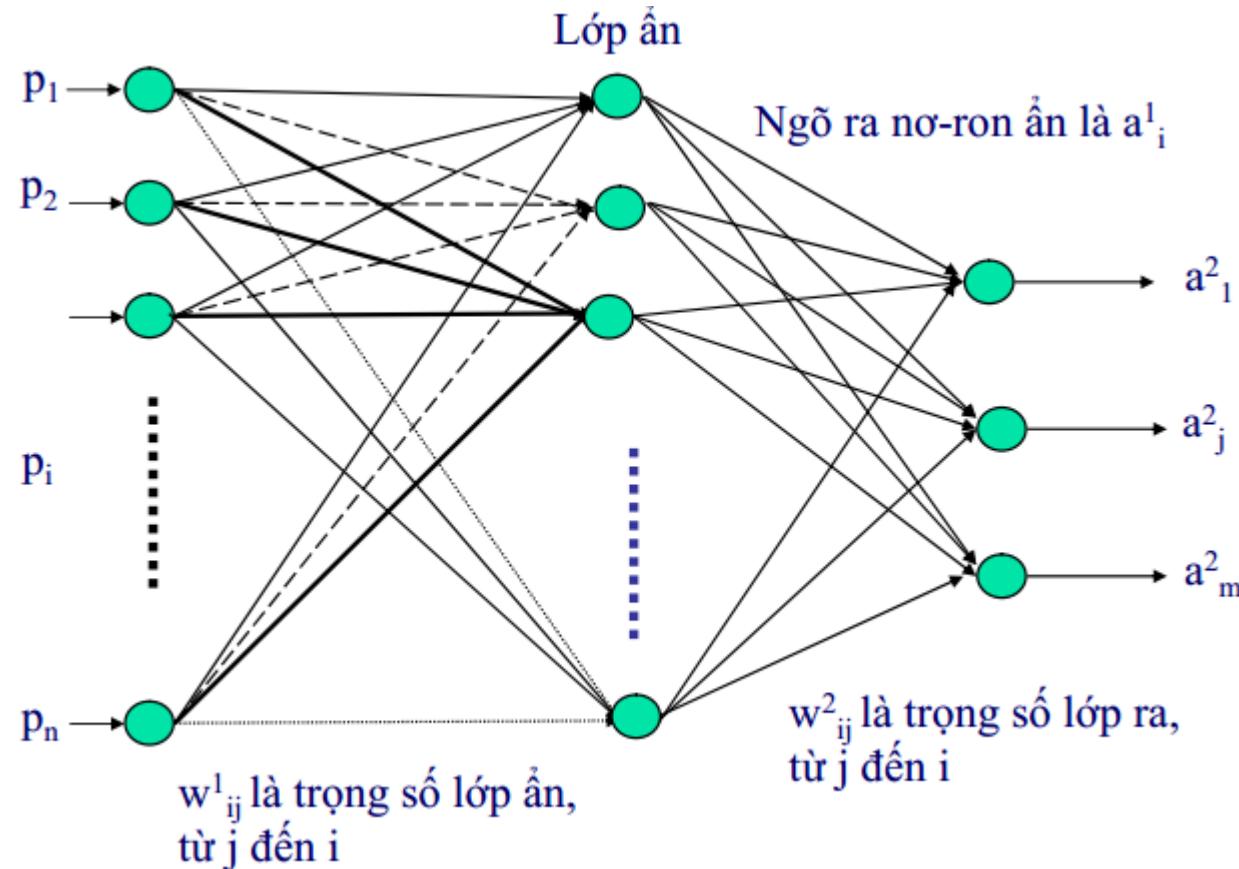
- Để cập nhật các trọng số cho mỗi chu kỳ huấn luyện, giải thuật truyền ngược cần 2 thao tác:
 - Thao tác truyền thuận (forward pass phase):** Áp vectơ dữ liệu vào trong tập dữ liệu học cho ANN và tính toán các ngõ ra của nó.
 - Thao tác truyền ngược (backward pass phase):** Xác định sai khác (lỗi) giữa ngõ ra thực tế của ANN và giá trị ngõ ra mong muốn trong tập dữ liệu học. Sau đó, truyền ngược lỗi này từ ngõ ra về ngõ vào của ANN và tính toán các giá trị mới của các trọng số, dựa trên giá trị lỗi này.



Minh họa phương pháp điều chỉnh trọng số neuron thứ j tại thời điểm k

Giải thuật BP – Gradient descent

- Xét một MLP 2 lớp:



Giải thuật BP – Gradient descent

- Pha truyền thuận

Tính ngõ ra lớp ẩn (hidden layer):

$$n^l_i(k) = \sum_j w^l_{ij}(k) p_j(k) \text{ tại thời điểm k}$$

$$a^l_i(k) = f^l(n^l_i(k))$$

với f^l là hàm kích truyền của các nơ-ron trên lớp ẩn.

Ngõ ra của lớp ẩn là ngõ vào của các nơ-ron trên lớp ra.

Tính ngõ ra ANN (output layer):

$$n^2_i(k) = \sum_j w^2_{ij}(k) a^l_j(k) \text{ tại thời điểm k}$$

$$a^2_i(k) = f^2(n^2_i(k))$$

với f^2 là hàm truyền của các nơ-ron trên lớp ra

Giải thuật BP – Gradient descent

- Pha truyền ngược

Tính tổng bình phương của lỗi: $E(k) = \frac{1}{2} \sum_i [t_i(k) - a^2_i(k)]^2$
 với $t(k)$ là ngõ ra mong muốn tại k

Tính sai số các nơ-ron ngõ ra:

$$\Delta_i(k) = -\frac{\partial E(k)}{\partial n^2_i(k)} = [t_i(k) - a_i(k)] f^2' [n^2_i(k)]$$

Tính sai số các nơ-ron ẩn:

$$\delta_i(k) = -\frac{\partial E(k)}{\partial n^1_i(k)} = f^1' [n^1_i(k)] \sum_j \Delta_j(k) w_{ji}$$

Cập nhật trọng số
 lớp ẩn: $w^1_{ij}(k+1) = w_{ij}(k) + \eta \delta_i(k) p_j(k)$

lớp ra: $w^2_{ij}(k+1) = w^2_{ij}(k) + \eta \Delta_i(k) a^1_j(k)$

Huấn luyện đến khi nào?

- Đủ số thời kỳ (epochs) ấn định trước
- Hàm mục tiêu đạt giá trị mong muốn
- Hàm mục tiêu phân kỳ.

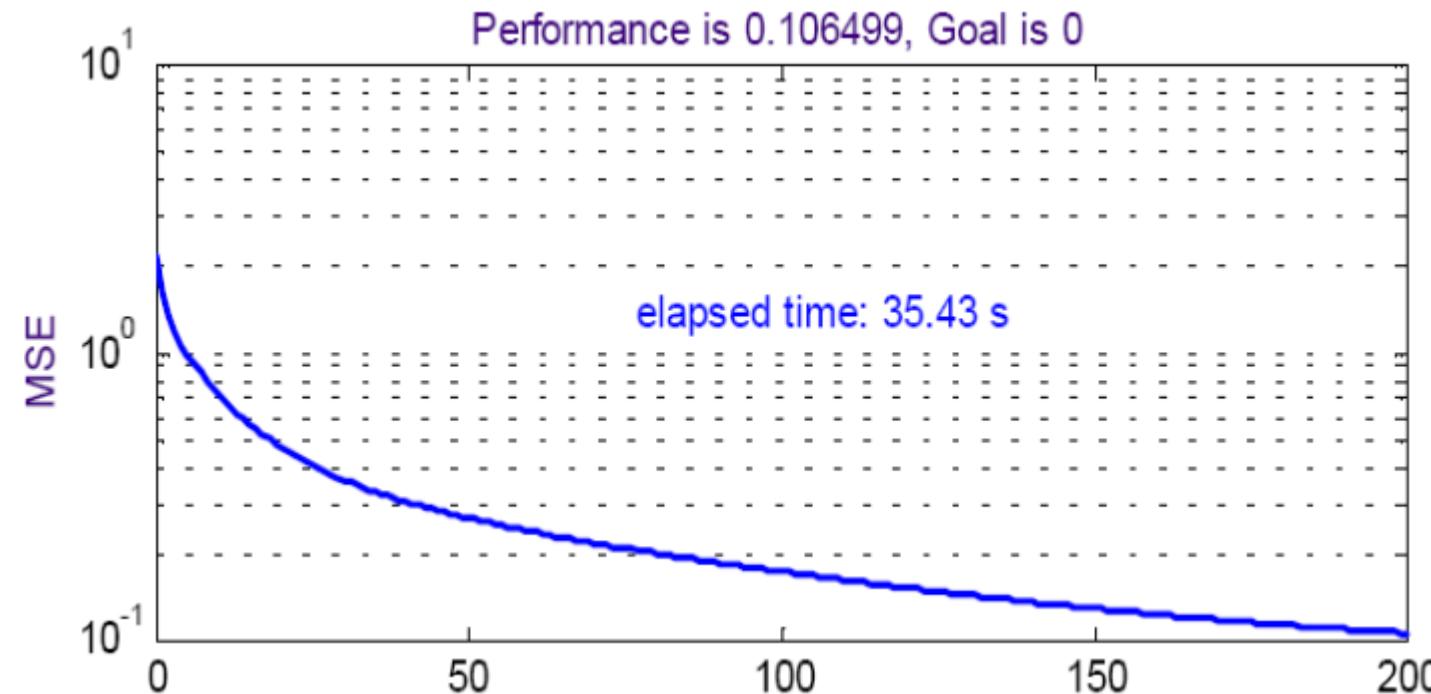
Hàm mục tiêu MSE

- Định nghĩa MSE: Giả sử ta có tập mẫu học: $\{p_1, t_1\}, \{p_2, t_2\}, \dots, \{p_N, t_N\}$, với $p=[p_1, p_2, \dots, p_N]$ là vectơ dữ liệu ngõ vào, $t=[t_1, t_2, \dots, t_N]$ là vectơ dữ liệu ngõ ra mong muốn. Gọi $a=[a_1, a_2, \dots, a_N]$ là vectơ dữ liệu ra thực tế thu được khi đưa vectơ dữ liệu vào p qua mạng. MSE:

$$MSE = \frac{1}{N} \sum_{i=1}^N (t_i - a_i)^2 \quad \text{được gọi là hàm mục tiêu}$$

- Mean Square Error (MSE) là lỗi bình phương trung bình, được xác định trong quá trình huấn luyện mạng. MSE được xem như là một trong những tiêu chuẩn đánh giá sự thành công của quá trình huấn luyện.
- MSE càng nhỏ, độ chính xác của ANN càng cao. ,
- Có thể sử dụng một số hàm sai số khác.

- MSE được xác định sau mỗi chu kỳ huấn luyện mạng (epoch) và được xem như 1 mục tiêu cần đạt đến. Quá trình huấn luyện kết thúc (đạt kết quả tốt) khi MSE đủ nhỏ.



- Cải tiến tốc độ hội tụ của giải thuật gradient descent: 1 nguyên tắc cập nhật trọng số của ANN:

$$w_{ij}(k+1) = w_{ij}(k) + \nabla w_{ij}(k)$$

$$\nabla w_{ij}(k) = \eta g(k) + \mu \nabla w_{ij}(k-1)$$

với

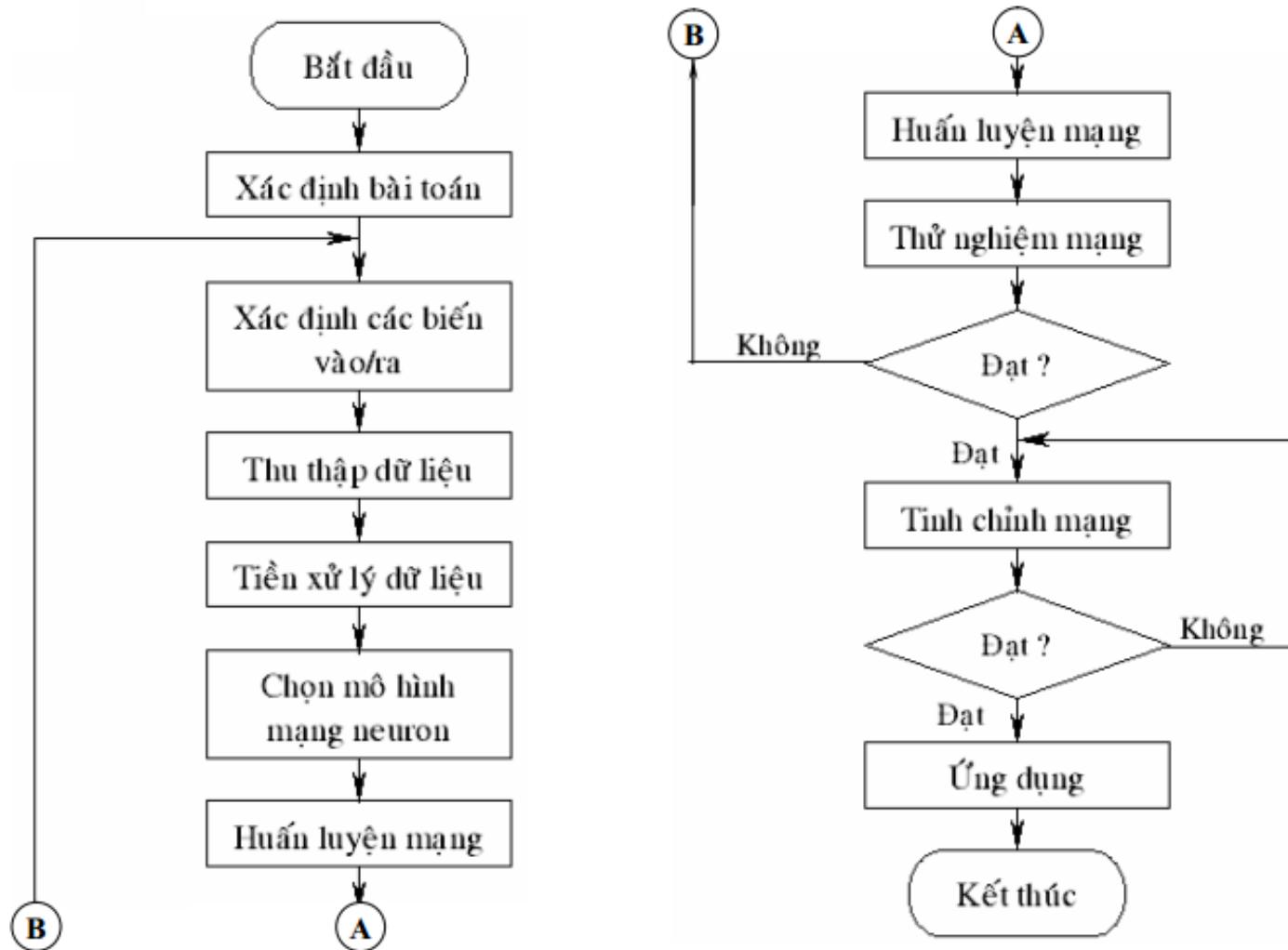
$g(k)$: gradient; η : tốc độ học

$\nabla_{ij}(k-1)$ là giá trị trước đó của $\nabla_{ij}(k)$

μ : momentum

- Thông thường, tổng giá trị của moment và tốc độ học nên gần bằng 1: $\mu \in [0.8, 1]$; $\eta \in [0, 0.2]$

Quy trình thiết kế ANN

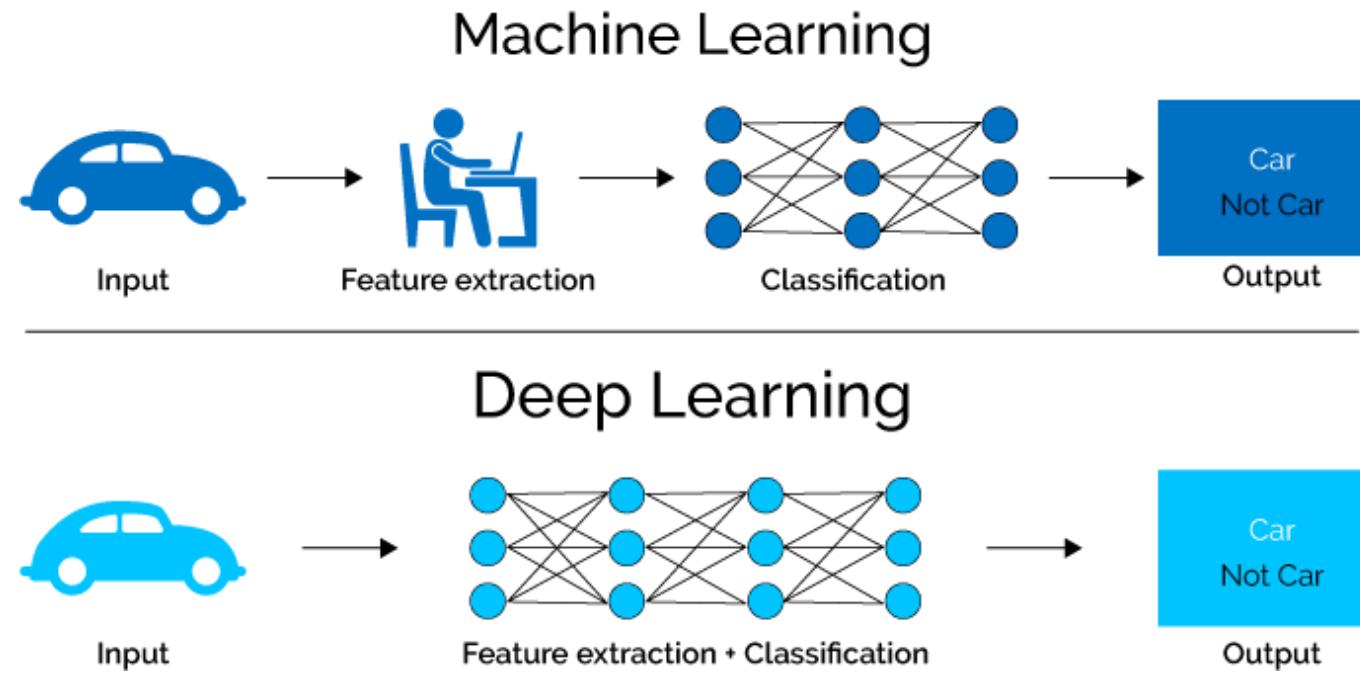


Thiết kế ANN

- Số lớp của mạng
- Số nơ-ron mỗi lớp ẩn
- Hàm kích hoạt tại mỗi nơ-ron
- Giải thuật huấn luyện mạng
- Hiểu đầu ra của mạng.

Mạng nơ-ron học sâu Deep Neural Network

Học máy và học sâu



Nguồn: [CS224 NLP with DL course at Stanford](#)

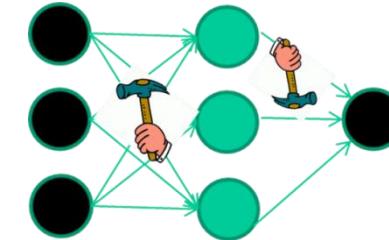
- Học máy cơ bản: Trích chọn đặc trưng (feature) từ ảnh → Học bộ phân loại từ các feature thu được → Hiệu chỉnh tham số.
- Học sâu: Feature được trích chọn tự động; Phân loại đối tượng, hiệu chỉnh tham số cũng có thể được tìm tự động

Deep learning Yann LeCun, Yoshua Bengio & Geoffrey Hinton, Nature, vol. 521, pages 436–444 (2015):

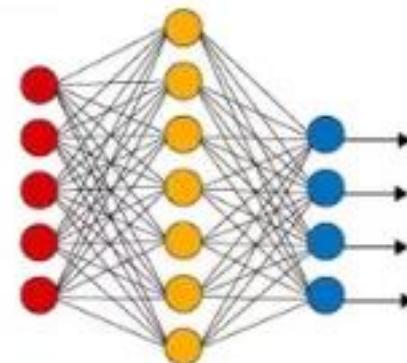
“Deep learning allows **computational models** that are composed of **multiple processing layers** to **learn representations** of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics. Deep learning discovers intricate structure in large data sets by using the backpropagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. Deep convolutional nets have brought about breakthroughs in processing images, video, speech and audio, whereas recurrent nets have shone light on sequential data such as text and speech.”

Mạng nơ-ron học sâu

- Nhiều lớp ẩn
 - Số lượng lớn các trọng số phải học
 - Độ phức tạp tính toán lớn!!!
- ➔ Học trọng số của từng lớp ẩn một

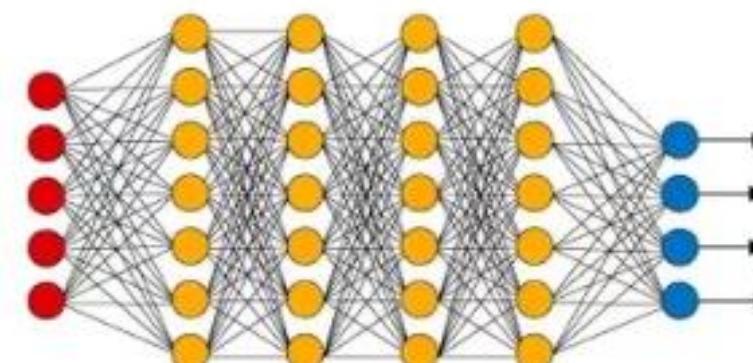


Simple Neural Network



● Input Layer

Deep Learning Neural Network

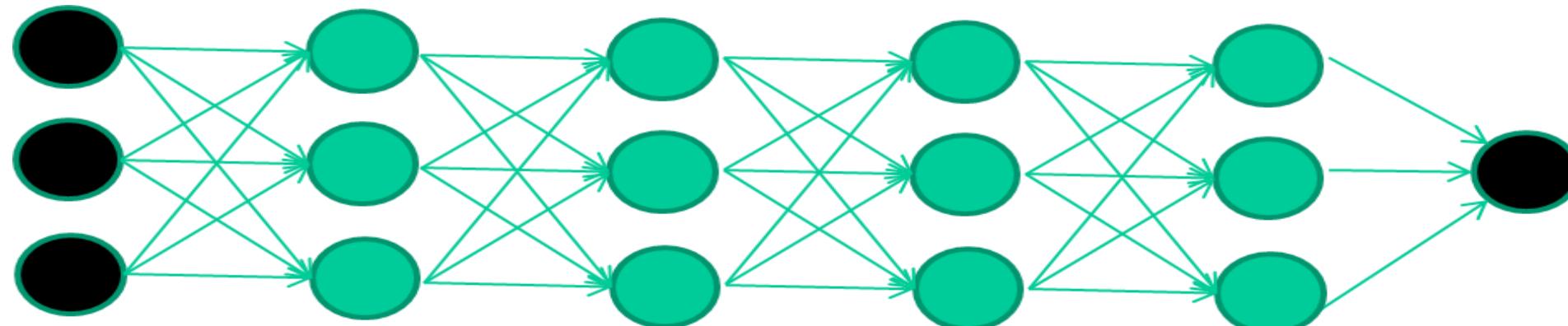


● Hidden Layer

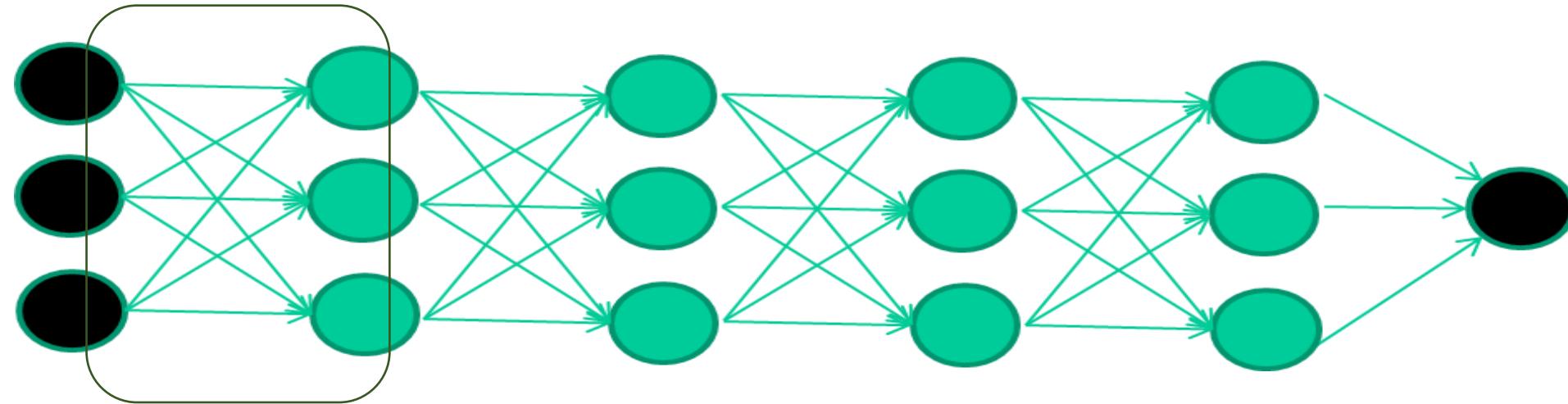
● Output Layer

Nguồn: Internet

Một phương pháp mới để huấn luyện ANN nhiều lớp

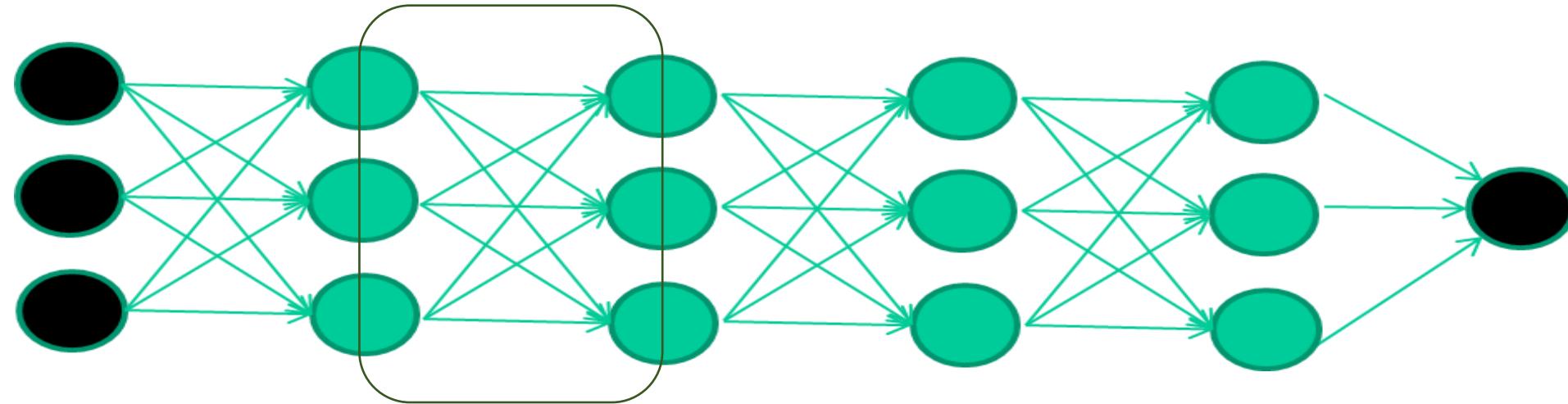


The new way to train multi-layer NNs...



Train this layer first

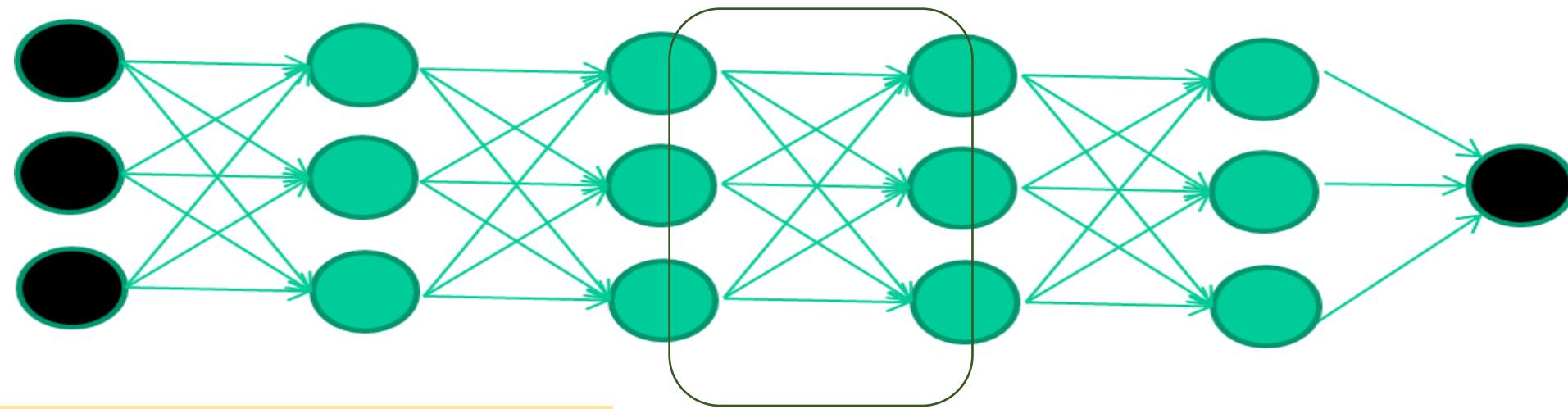
The new way to train multi-layer NNs...



Train this layer first

then this layer

The new way to train multi-layer NNs...

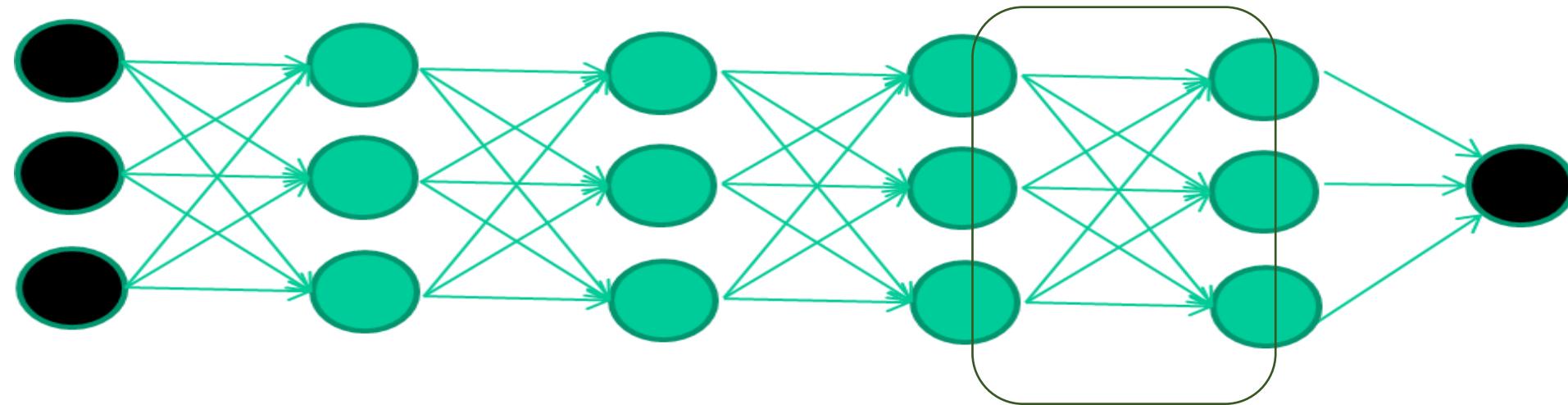


Train this layer first

then this layer

then this layer

The new way to train multi-layer NNs...



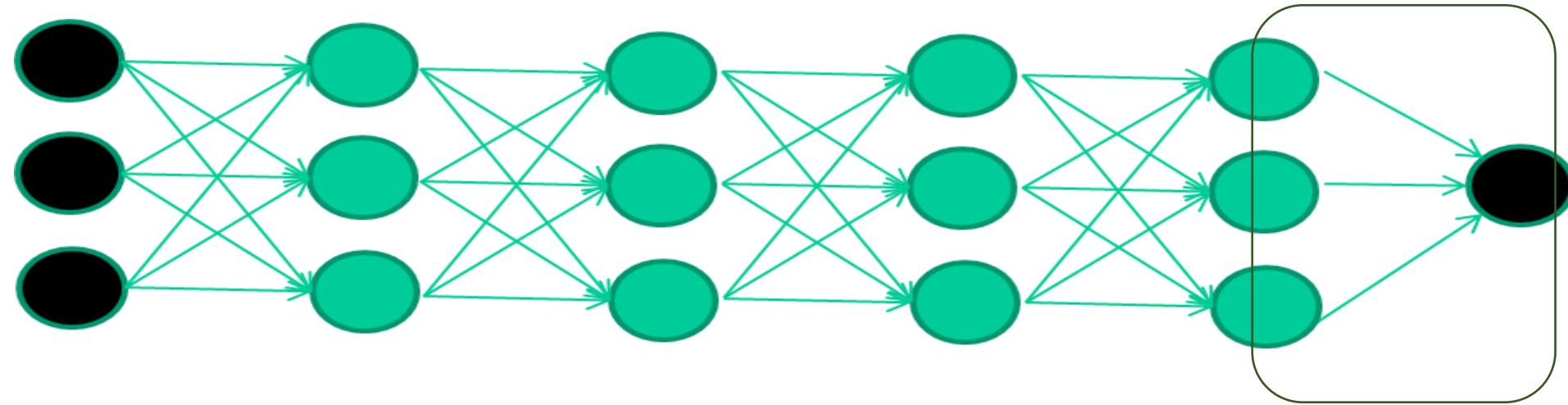
Train this layer first

then this layer

then this layer

then this layer

The new way to train multi-layer NNs...



Train this layer first

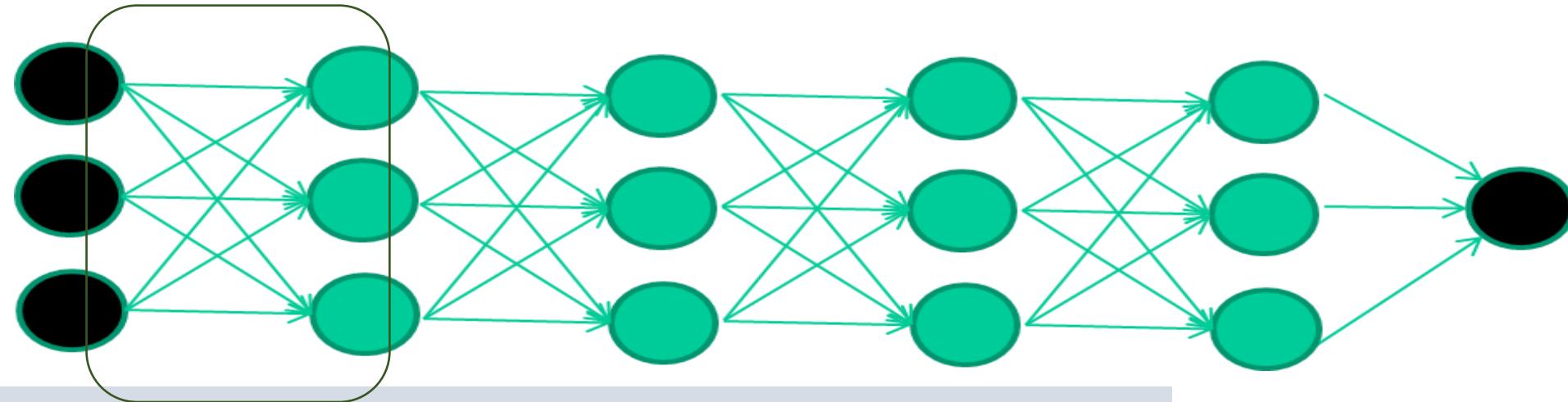
then this layer

then this layer

then this layer

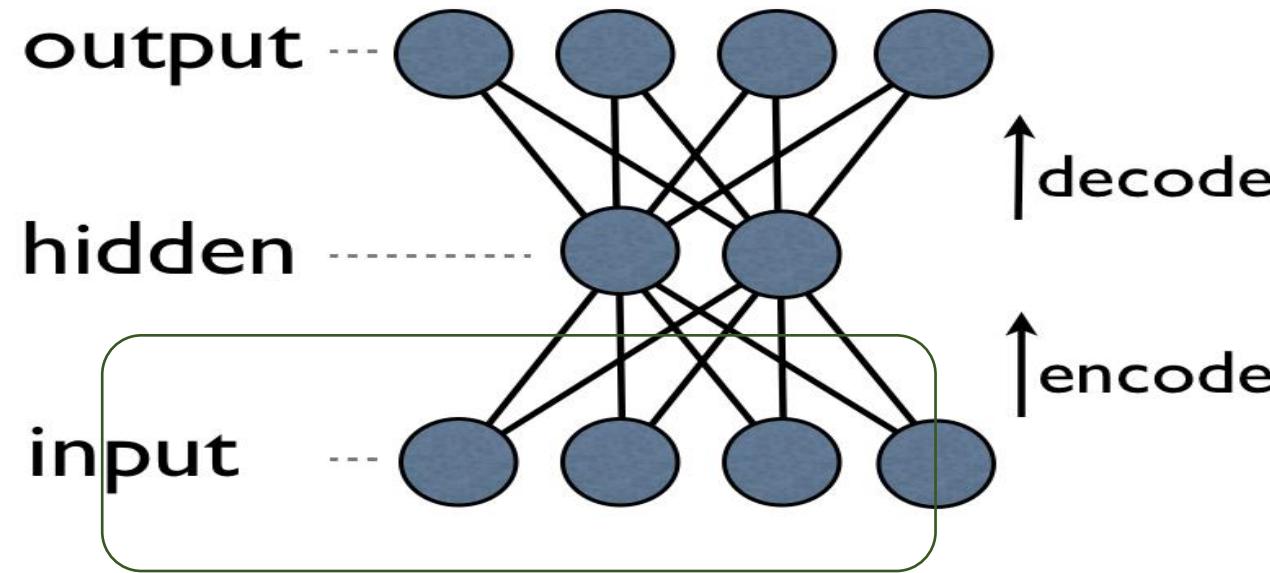
finally this layer

The new way to train multi-layer NNs...

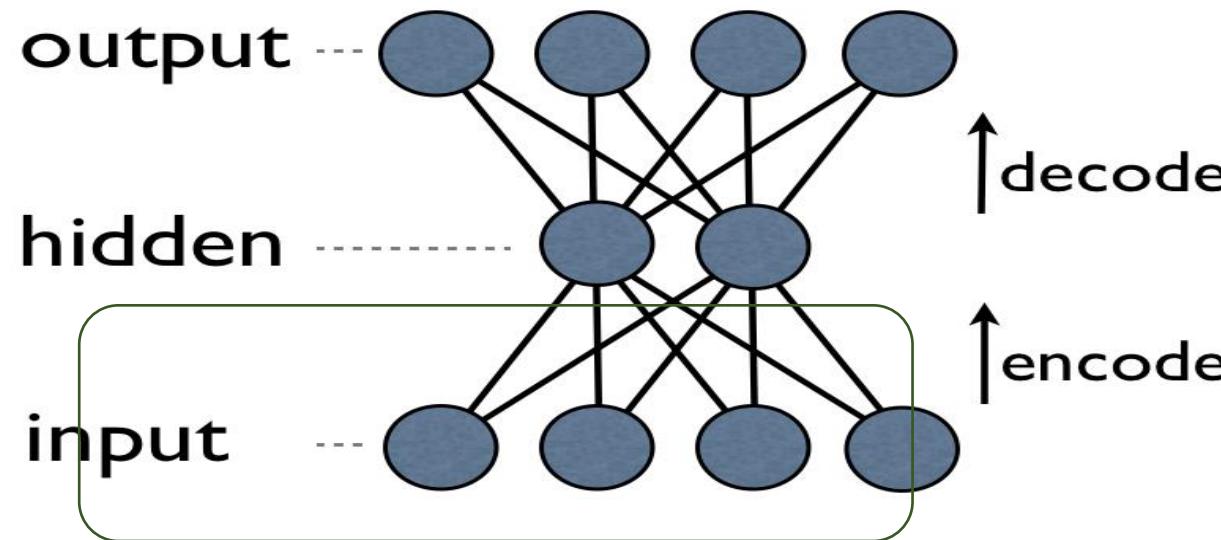


*EACH of the (non-output) layers is trained to be an
*auto-encoder**

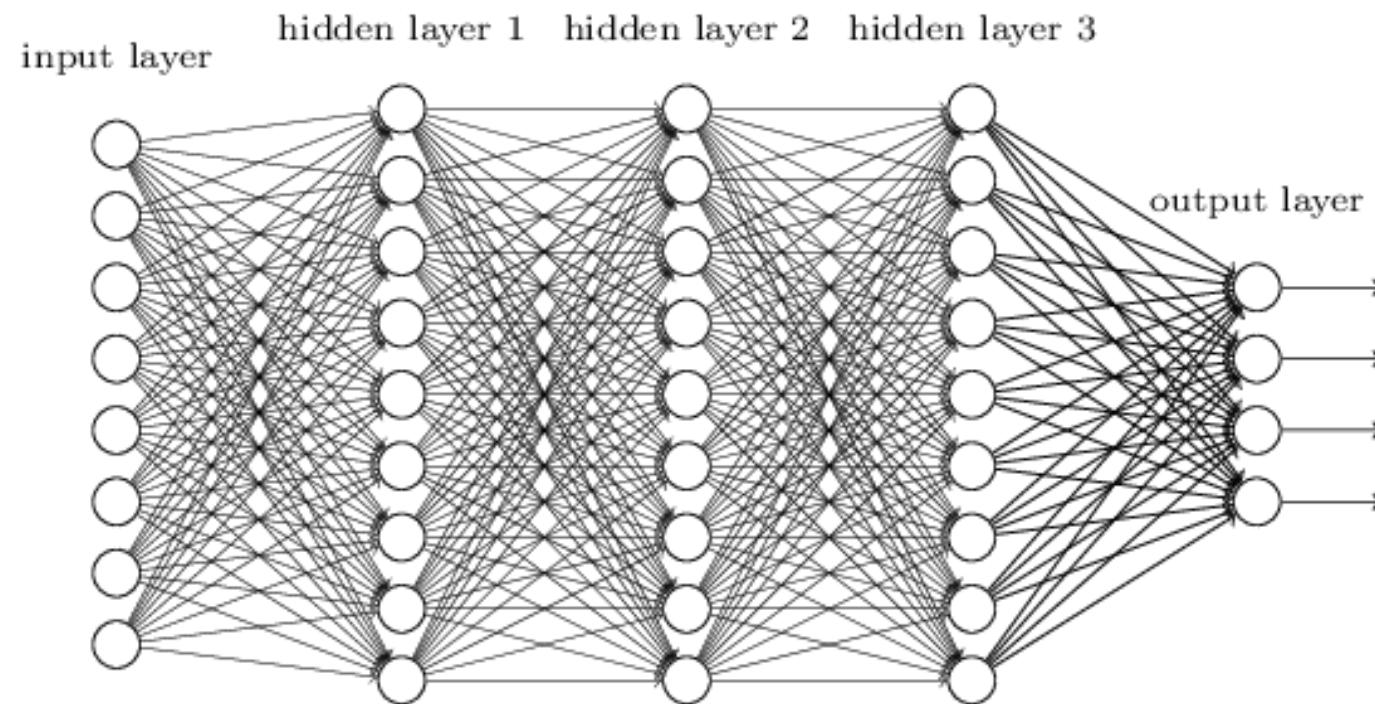
*Basically, it is forced to learn good features that describe
what comes from the previous layer*



an auto-encoder is trained, with an absolutely standard weight-adjustment algorithm to reproduce the input

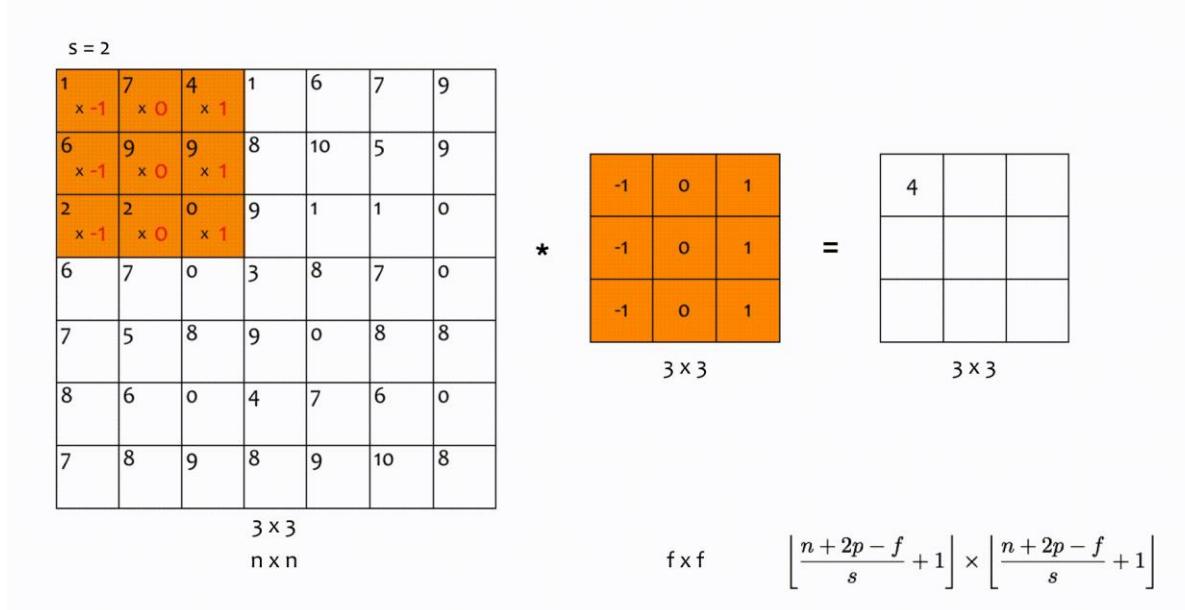


By making this happen with (many) fewer units than the inputs, this forces the 'hidden layer' units to become good feature detectors



Mô hình mạng neural tích chập

- Convolutional Neural Network (CNN)
- Convolution (tích chập):



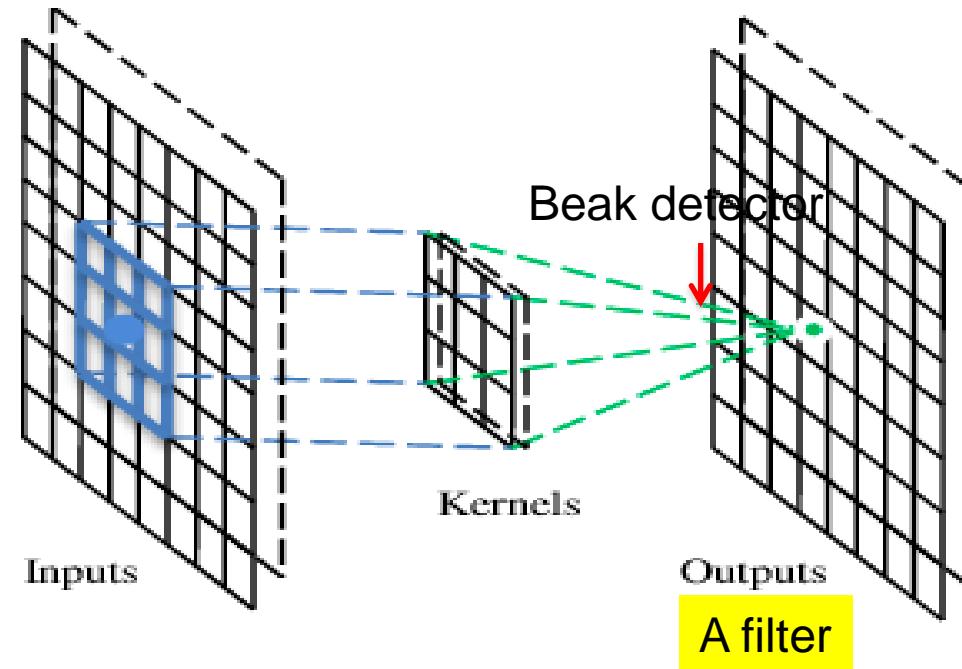
- Conv Layer: Các lớp tích chập nhằm trích chọn các đặc trưng của ảnh đầu vào

Một số khái niệm cơ bản

- **Filter** (còn gọi là Kernel hay Feature Detector): ma trận lọc. Thông thường, ở các lớp đầu tiên của Conv Layer sẽ có kích thước là [5x5x3]
- **Convolved Feature** (Activation Map hay Feature Map): đầu ra của ảnh khi cho bộ lọc quét qua hết các vị trí trong ảnh và thực hiện phép nhân vô hướng.
- **Receptive field**: vùng ảnh được chọn để tính tích chập, bằng đúng kích thước của bộ lọc.
- **Depth**: số lượng bộ lọc.
- **Stride**: khoảng cách dịch chuyển của bộ lọc sau mỗi lần tính. Ví dụ khi stride=2. Tức sau khi tính xong tại 1 vùng ảnh, nó sẽ dịch sang phải 2 pixel. Tương tự cho việc dịch xuống dưới.
- **Zero-Padding**: thêm các giá trị 0 ở xung quanh biên ảnh, để đảm bảo phép tích chập được thực hiện đủ trên toàn ảnh.

A convolutional layer

CNN là một neural network với một số lớp nhân chập (convolutional layers). Một lớp nhân chập thực hiện phép lọc với một số các bộ lọc trên ảnh.



Convolution

These are the network parameters to be learned.

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

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

: :

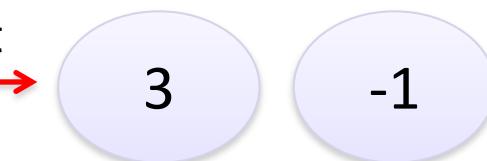
Each filter detects a small pattern (3 x 3).

Convolution

stride=1

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

Dot
product



6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

Convolution

If stride=2

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

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

$$\begin{array}{c} 3 \\ -3 \end{array}$$

Convolution

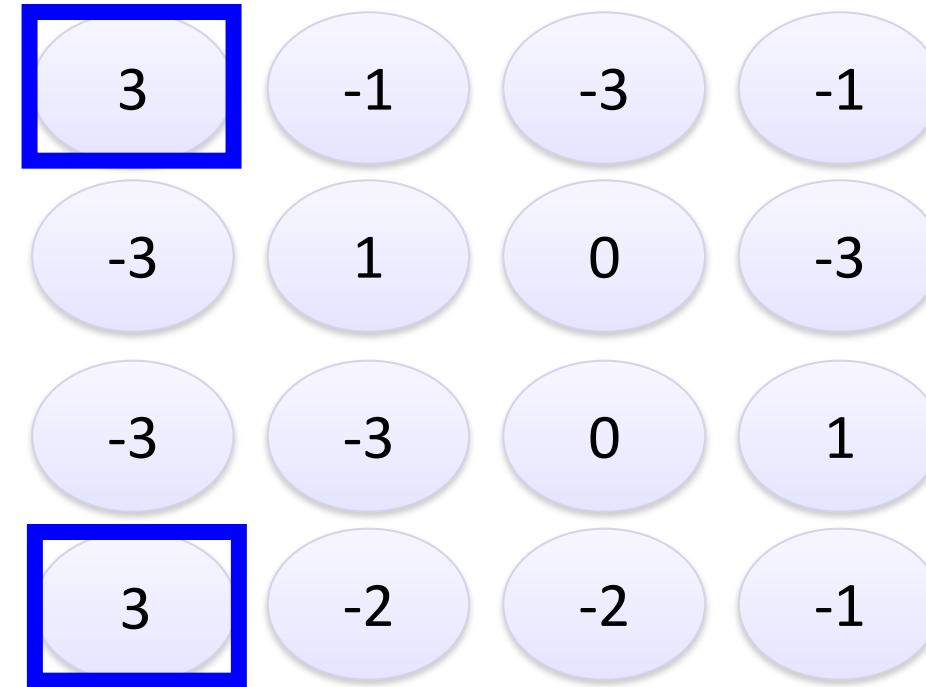
stride=1

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

6 x 6 image

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1



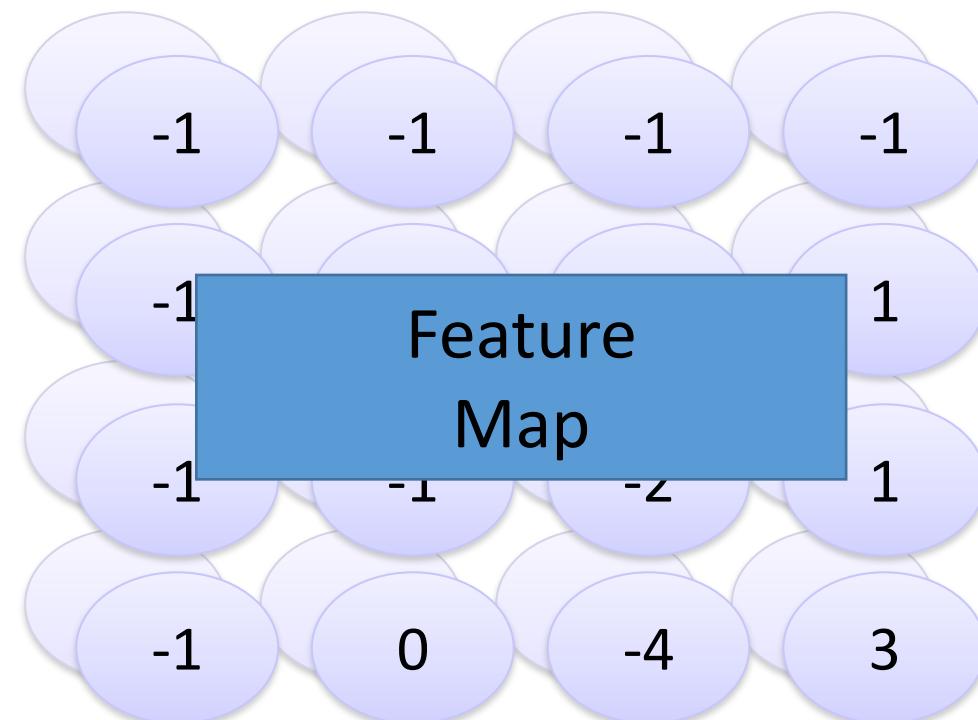
Convolution

stride=1

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

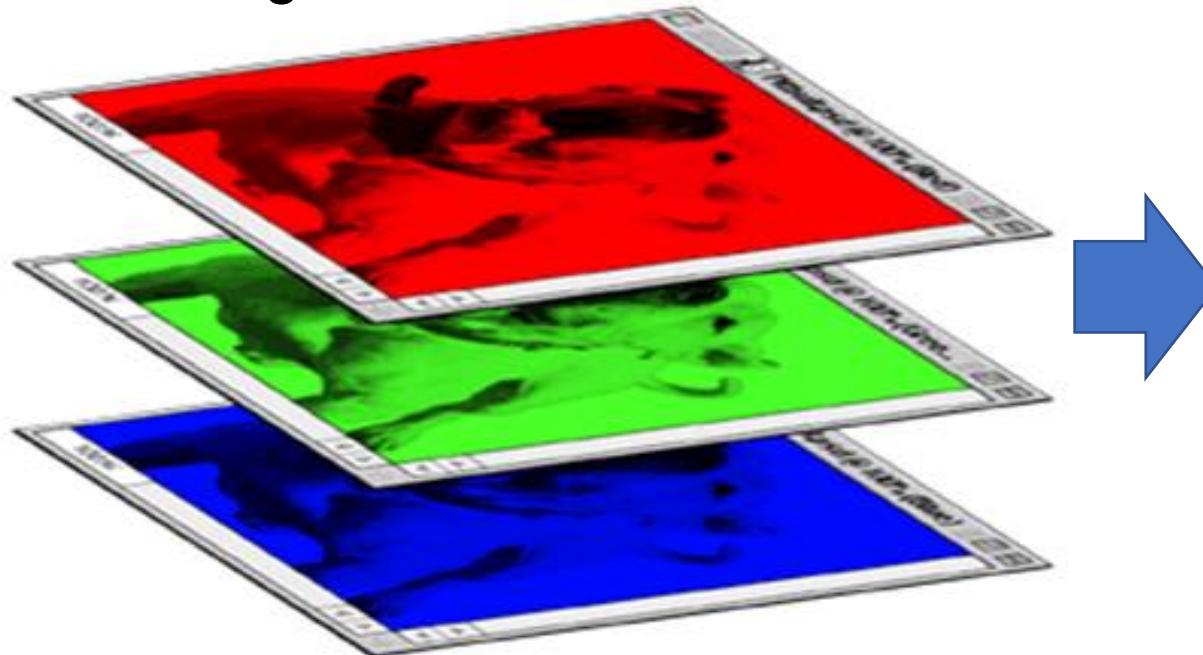
6 x 6 image

Repeat this for each filter



Color image: RGB 3 channels

Color image



1	-1	-1
-1	1	-1
-1	-1	1

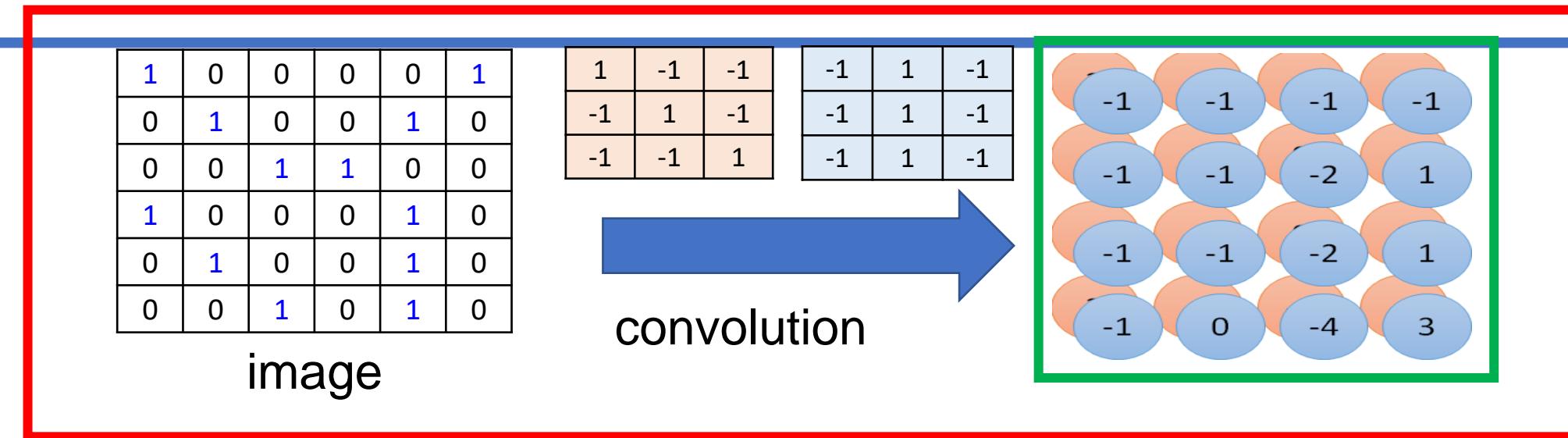
Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2

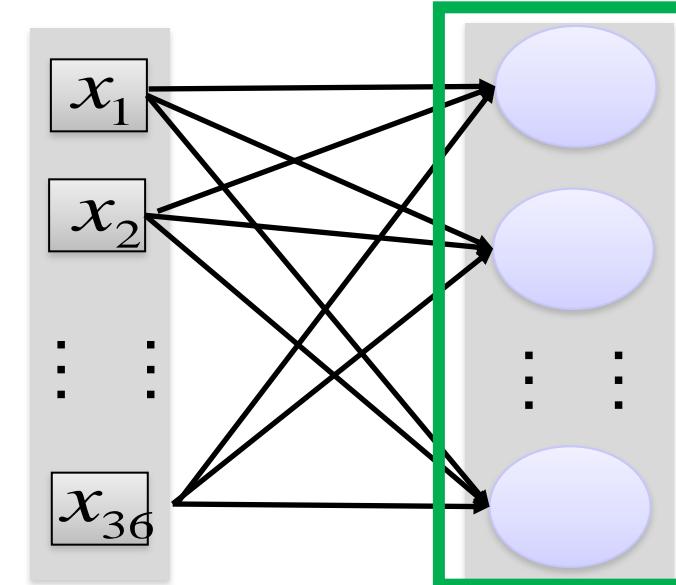
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	1	0	0
1	0	0	0	1	0
0	1	0	0	1	0
0	0	1	0	1	0

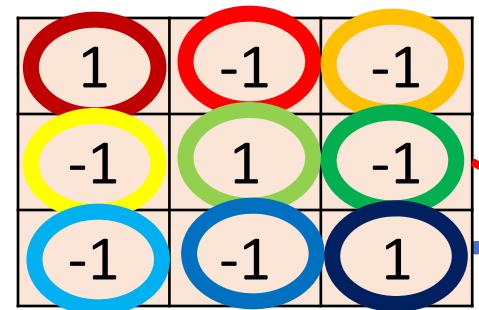
Convolution v.s. Fully Connected



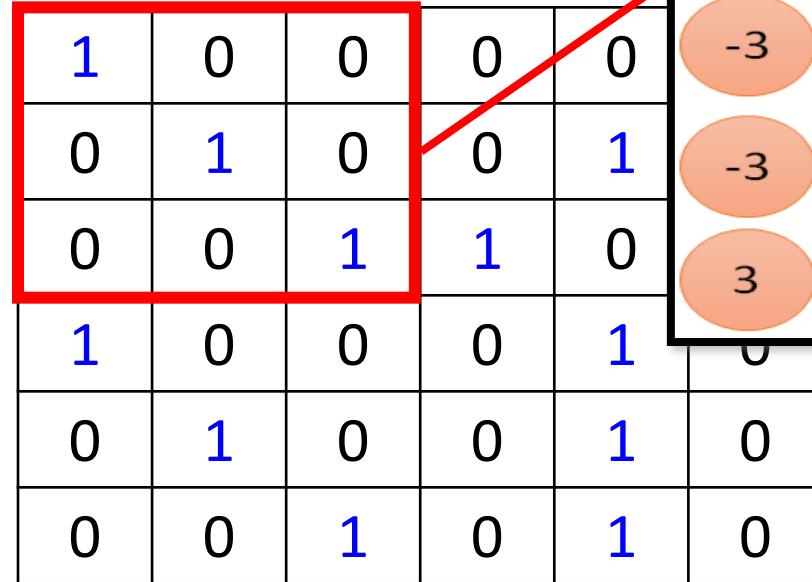
Fully-
connected

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

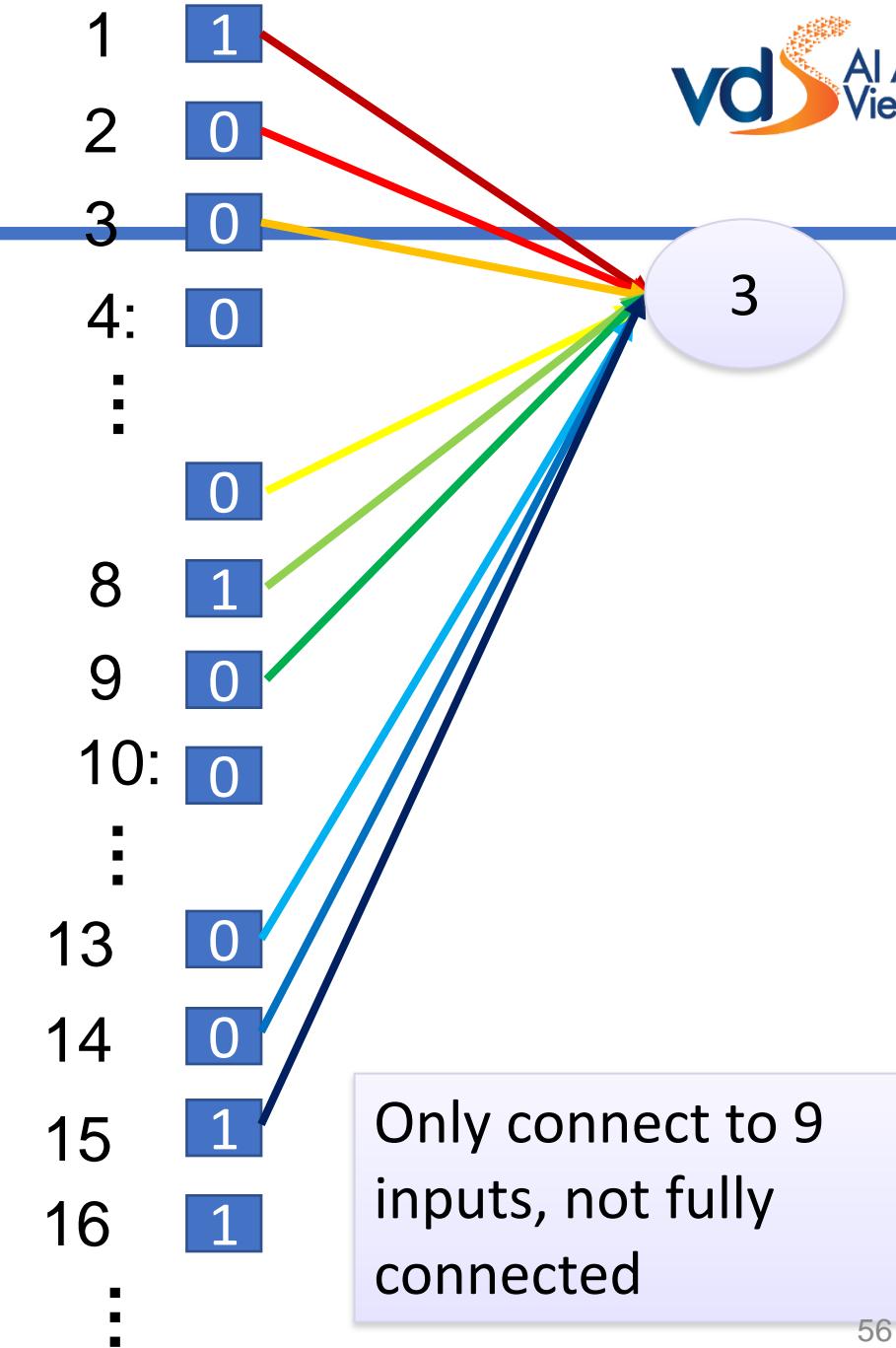
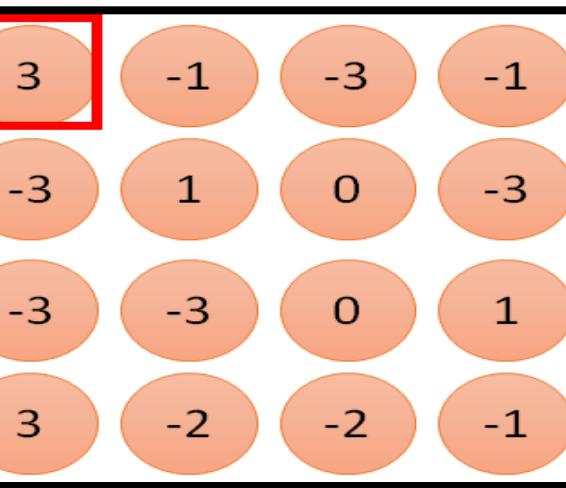


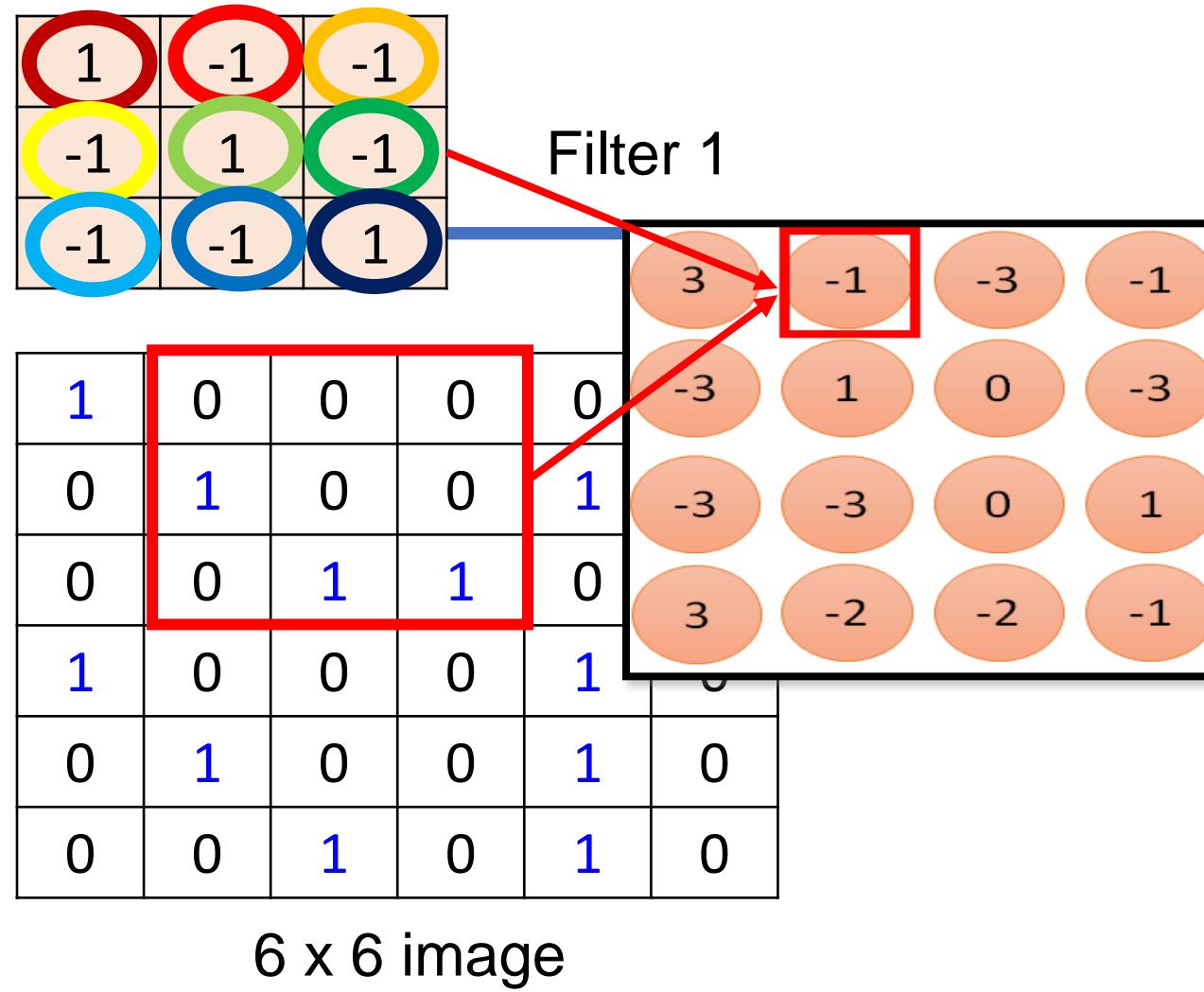


Filter 1



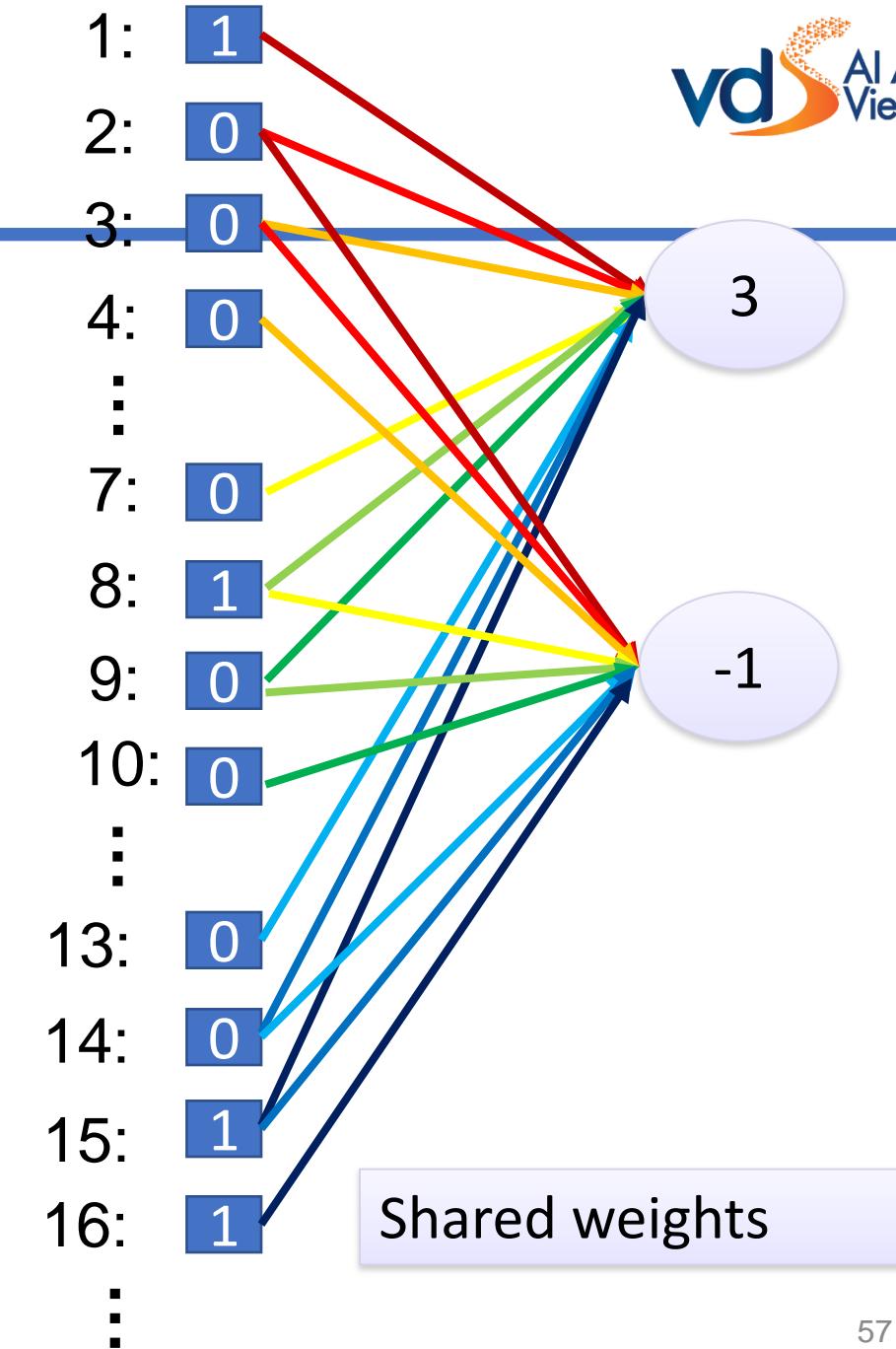
fewer parameters!





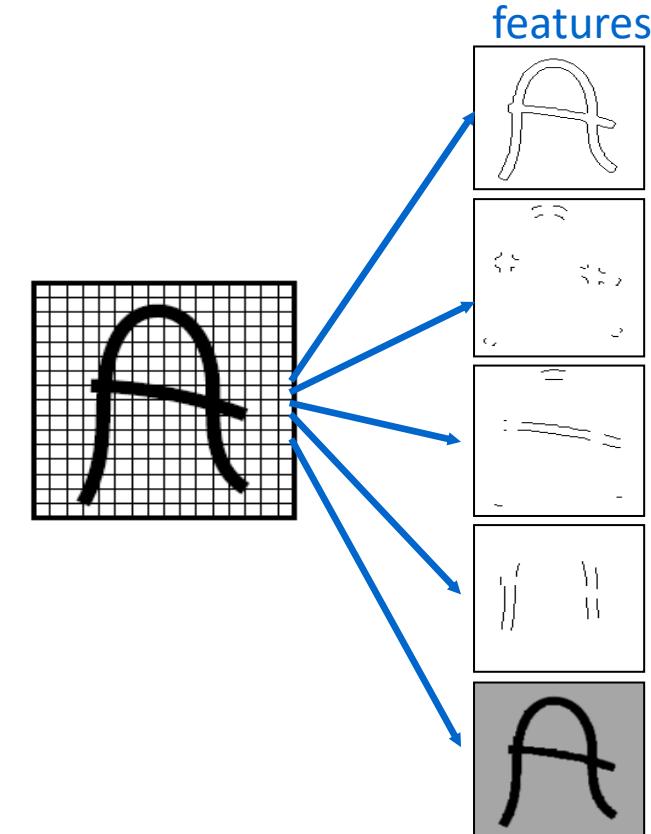
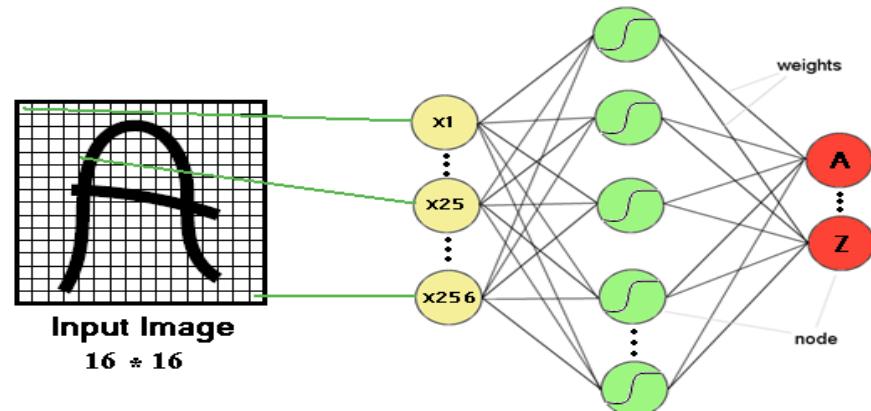
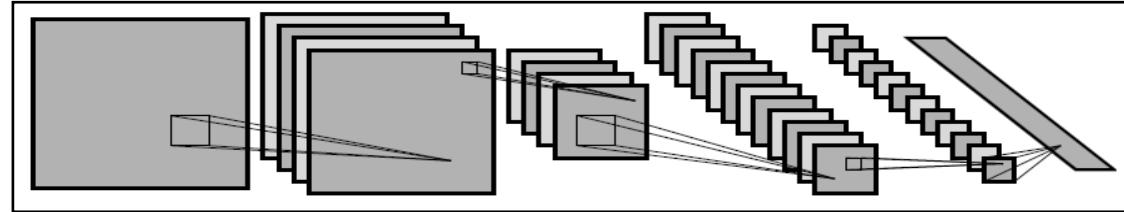
Fewer parameters

Even fewer parameters

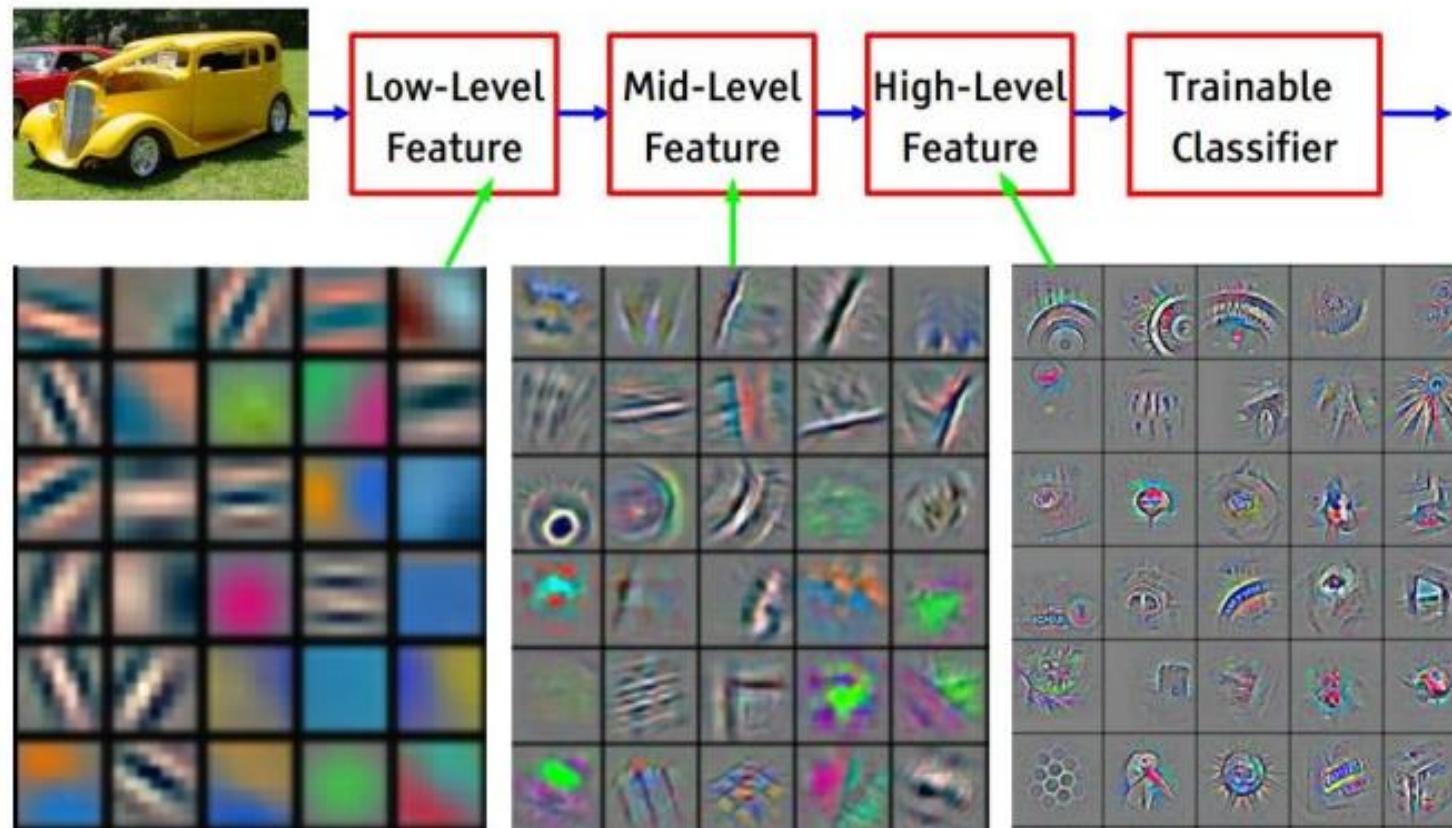


Feature extraction layer or Convolution layer

- Detect the same feature at different positions in the input image.

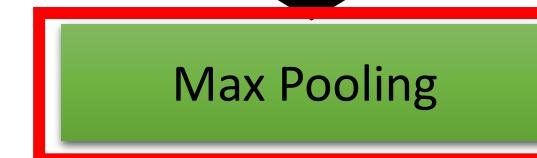
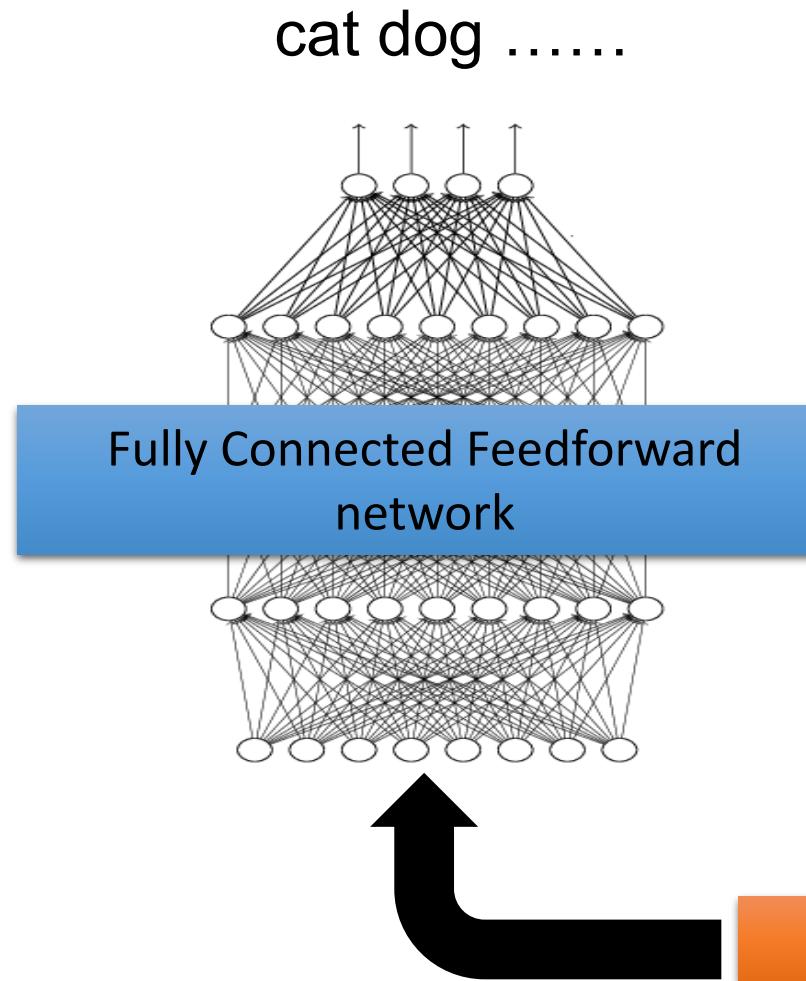


[From recent Yann
LeCun slides]



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

The whole CNN



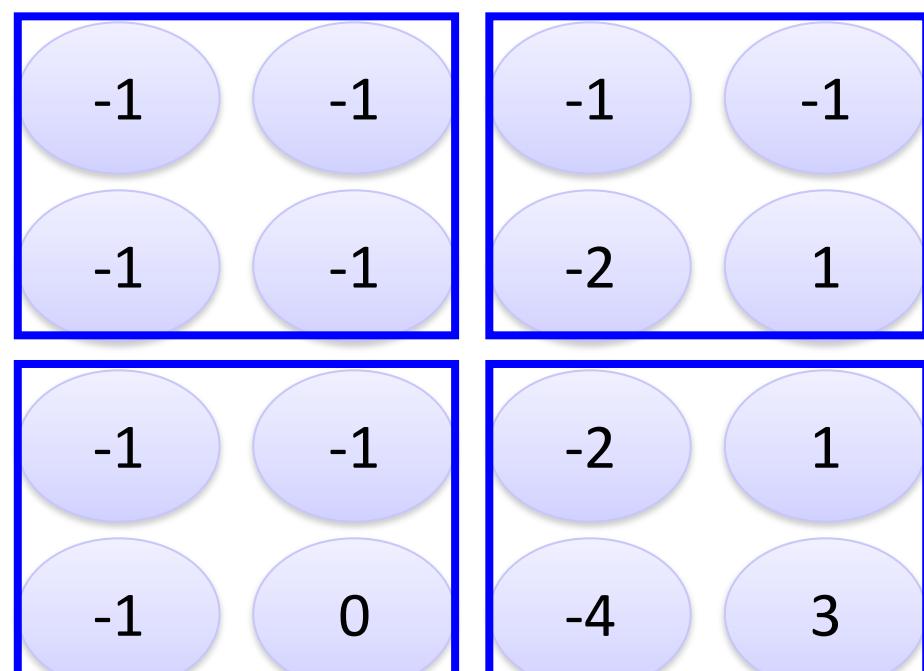
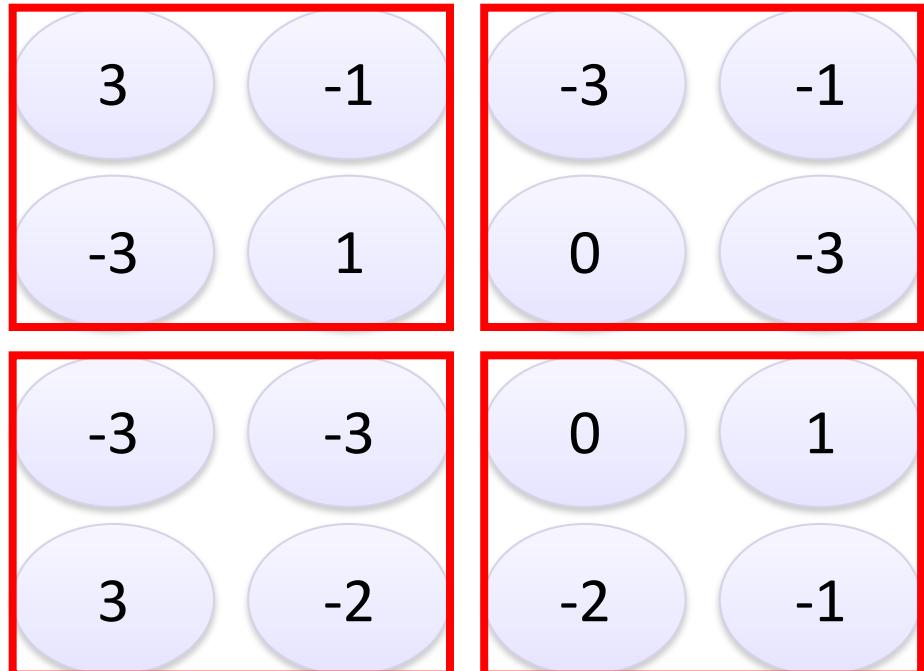
Max Pooling

1	-1	-1
-1	1	-1
-1	-1	1

Filter 1

-1	1	-1
-1	1	-1
-1	1	-1

Filter 2



Why Pooling

- Subsampling pixels will not change the object

bird



Subsampling

bird



We can subsample the pixels to make image smaller



fewer parameters to characterize the image

A CNN compresses a fully connected network

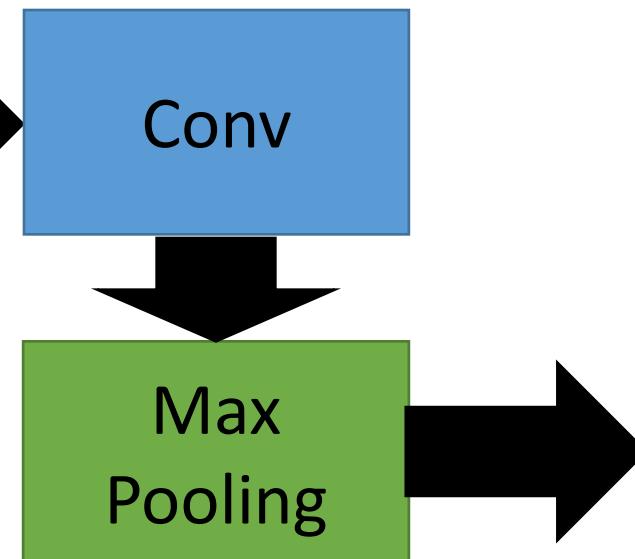
2 ways

- Reducing number of connections
- Shared weights on the edges
- Max pooling further reduces the complexity

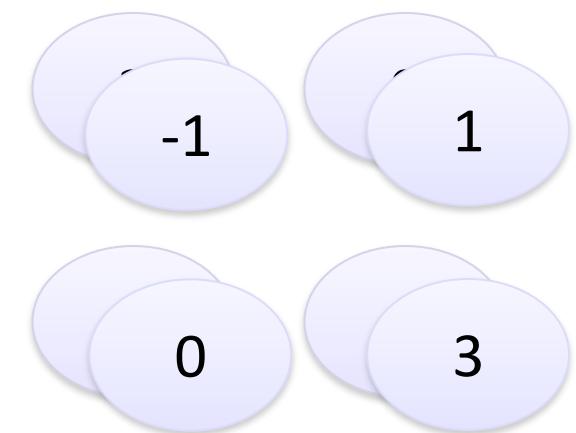
Max Pooling

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

6 x 6 image



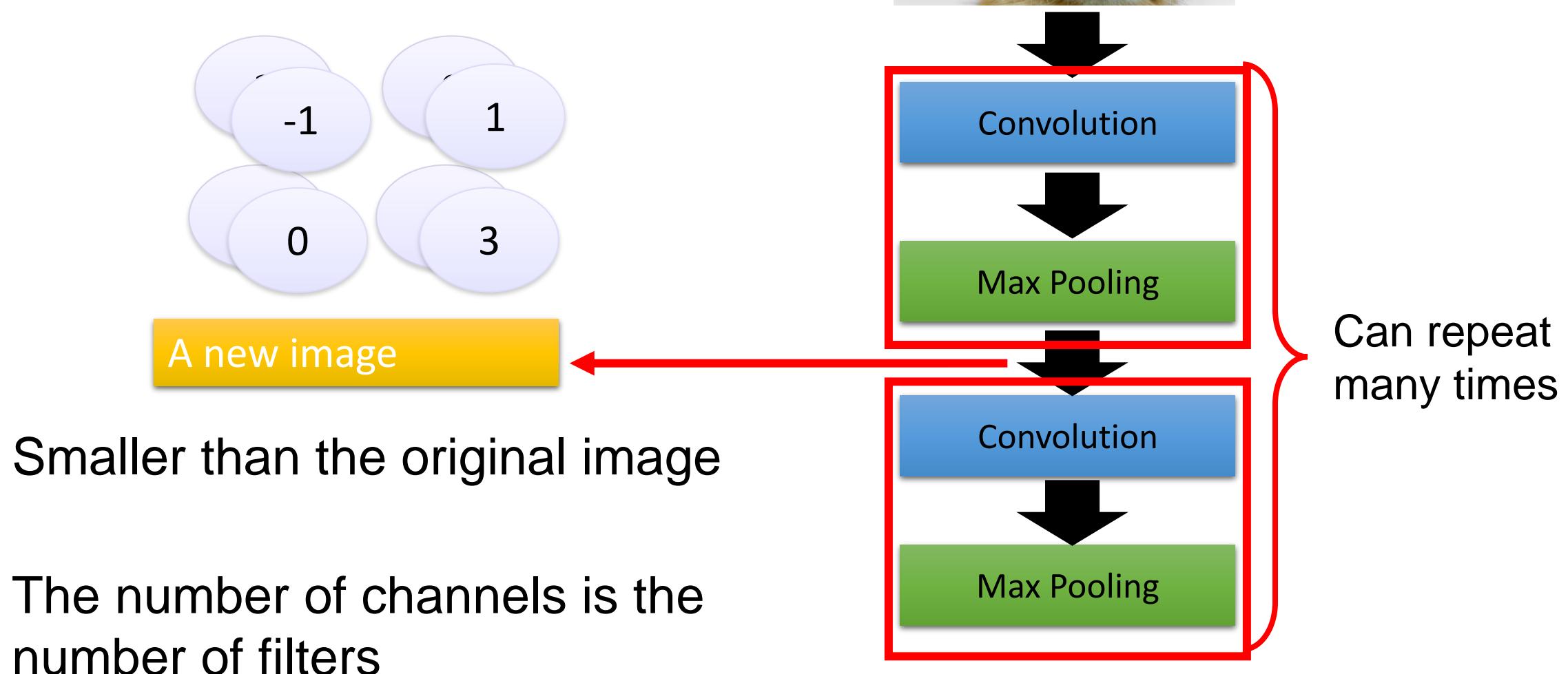
New image
but smaller



2 x 2 image

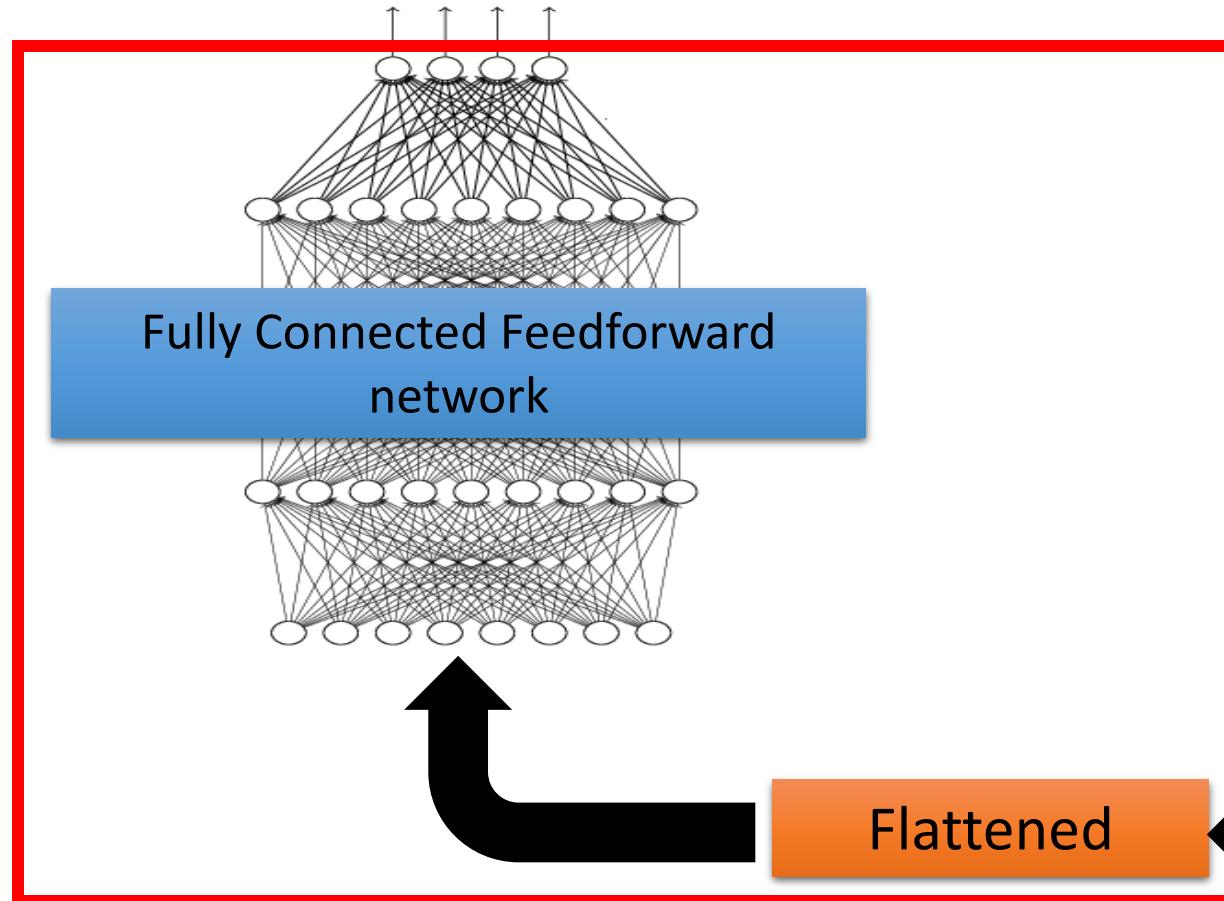
Each filter
is a channel

The whole CNN



The whole CNN

cat dog



Convolution

Max Pooling

Convolution

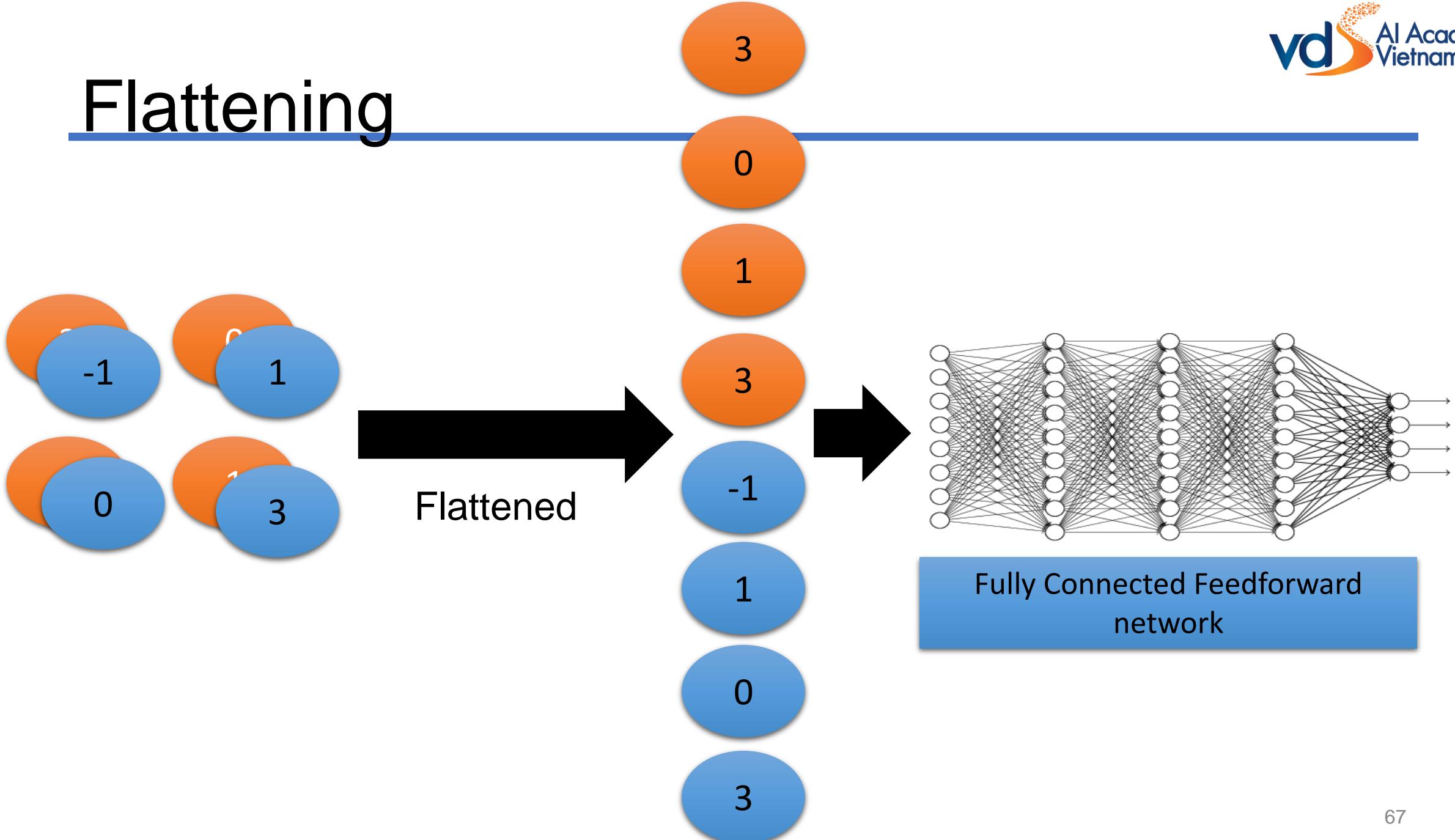
Max Pooling

A new image

A new image

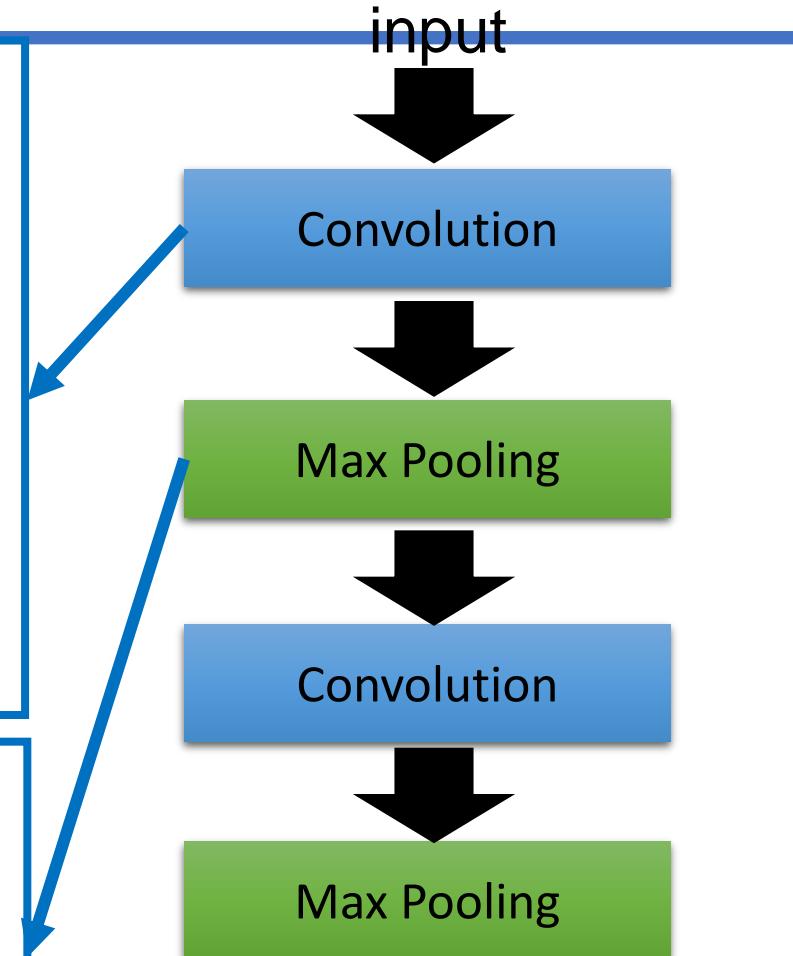
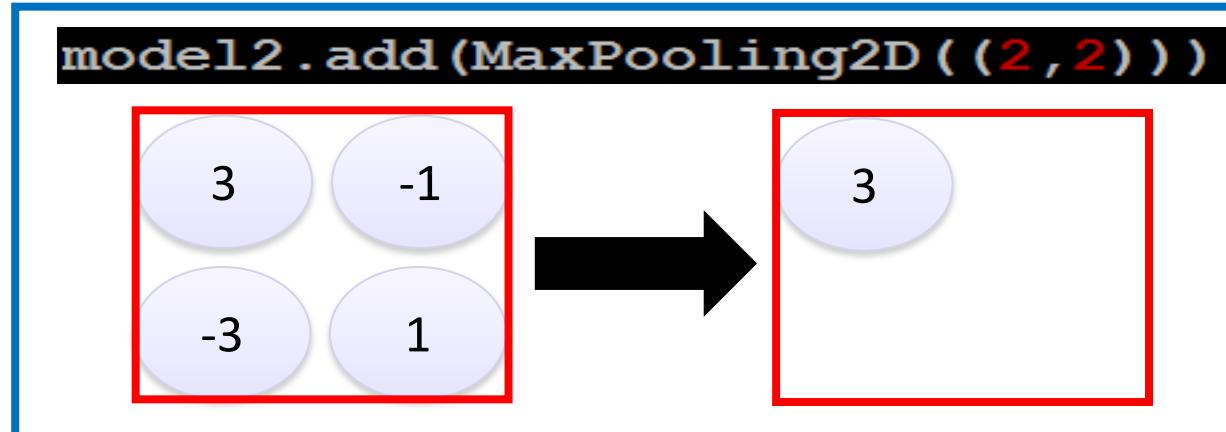
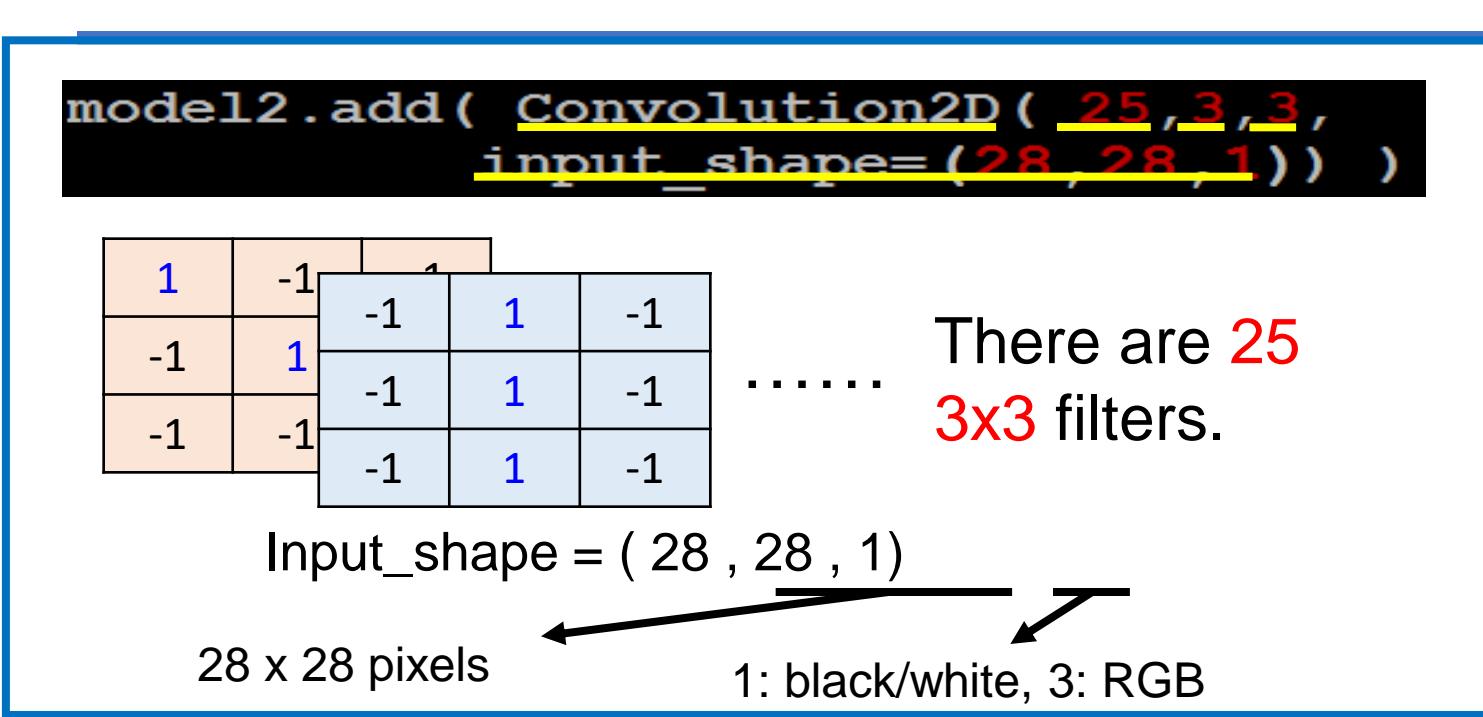
Flattened

Flattening



CNN in Keras

Only modified the *network structure* and *input format (vector -> 3-D tensor)*



CNN in Keras

Only modified the *network structure* and *input format (vector -> 3-D array)*

How many parameters for each filter?

```
model2.add( Convolution2D( 25, 3, 3,  
    input_shape=(28, 28, 1) ) )
```

9

25 x 26 x 26

```
model2.add(MaxPooling2D( (2, 2) ))
```

25 x 13 x 13

```
model2.add(Convolution2D(50, 3, 3))
```

225=
25x9

50 x 11 x 11

```
model2.add(MaxPooling2D( (2, 2) ))
```

50 x 5 x 5

Input

Convolution

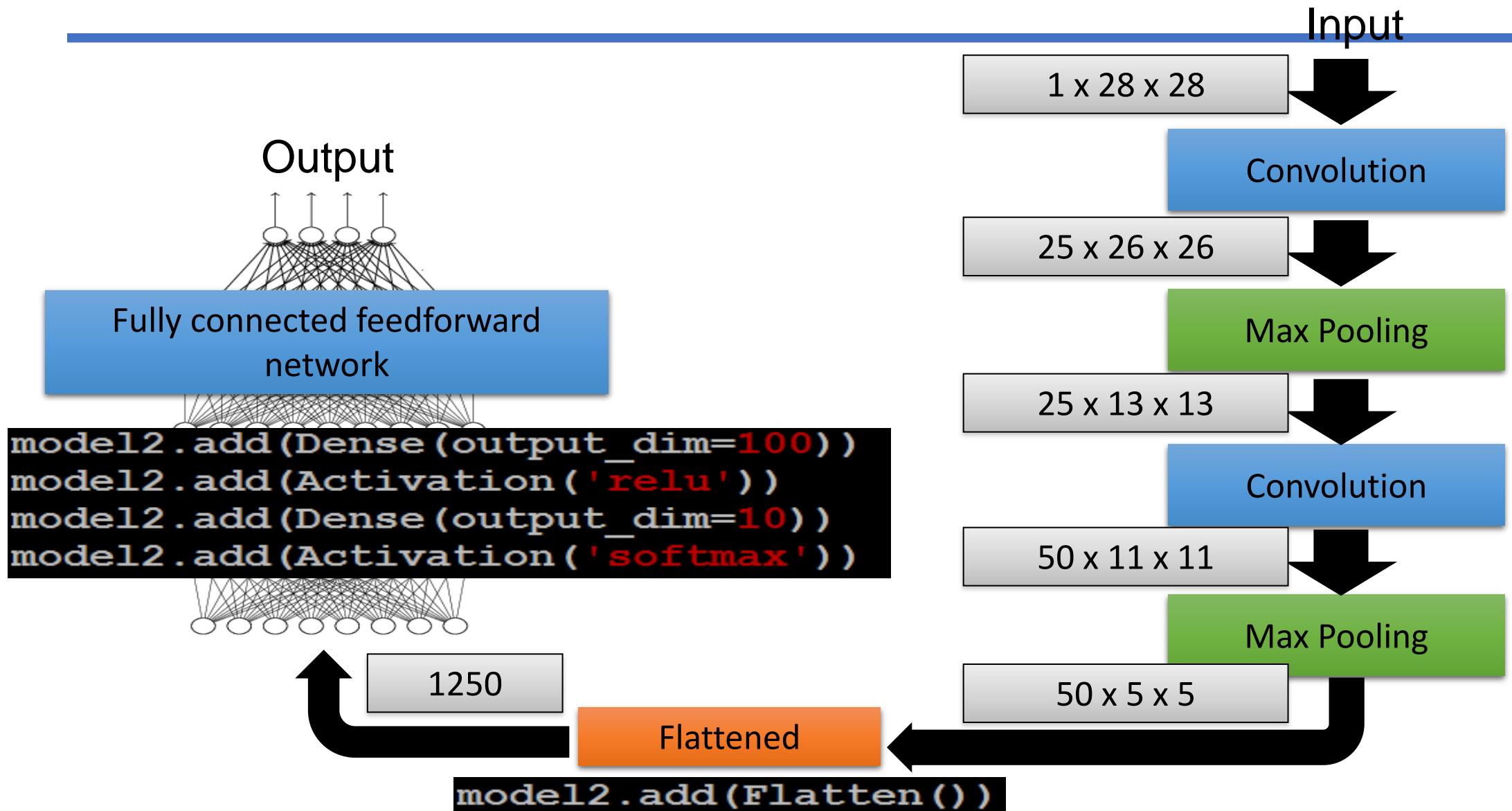
Max Pooling

Convolution

Max Pooling

CNN in Keras

Only modified the *network structure* and *input format (vector -> 3-D array)*



Thực hành ứng dụng ANN, CNN trong nhận dạng ảnh

Thực hành

- Chuẩn bị dữ liệu
- Huấn luyện mô hình
- Sử dụng mô hình đã huấn luyện để nhận dạng ảnh chữ viết, mặt người (trạng thái/cười)
- Đánh giá hiệu năng của mô hình đã huấn luyện

Tài liệu tham khảo

1. David Forsyth and Jean Ponce, Computer Vision: A Modern Approach, Book,
2. Adrian Rosebrock, Deep Learning for Computer vision with Python
3. CS231n: Convolutional Neural Networks for Visual Recognition, Stanford
4. <https://keras.io/layers/convolutional/>
5. Other materials from Computer vision courses (available online).

THỊ GIÁC MÁY TÍNH

AI Academy Vietnam

CẢM ƠN!