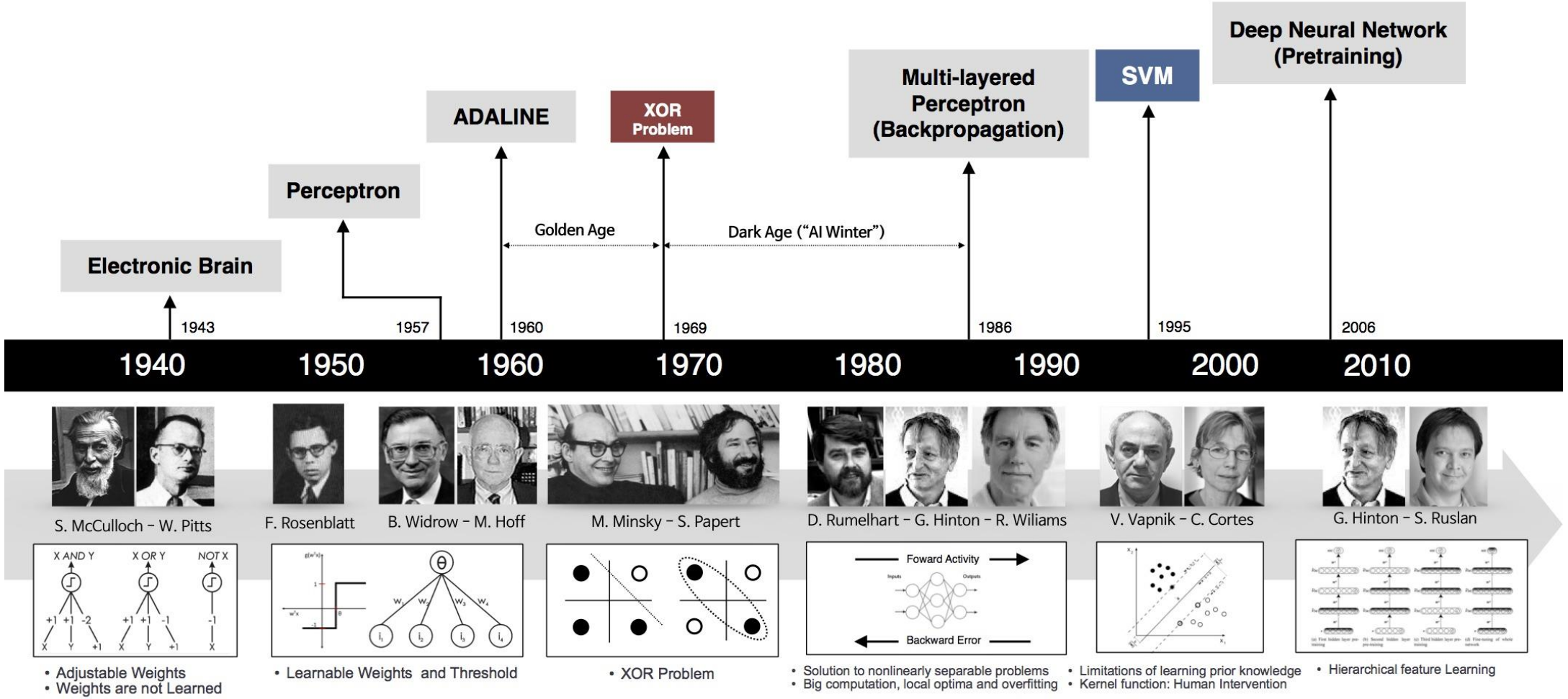
Deep Learning

Contents

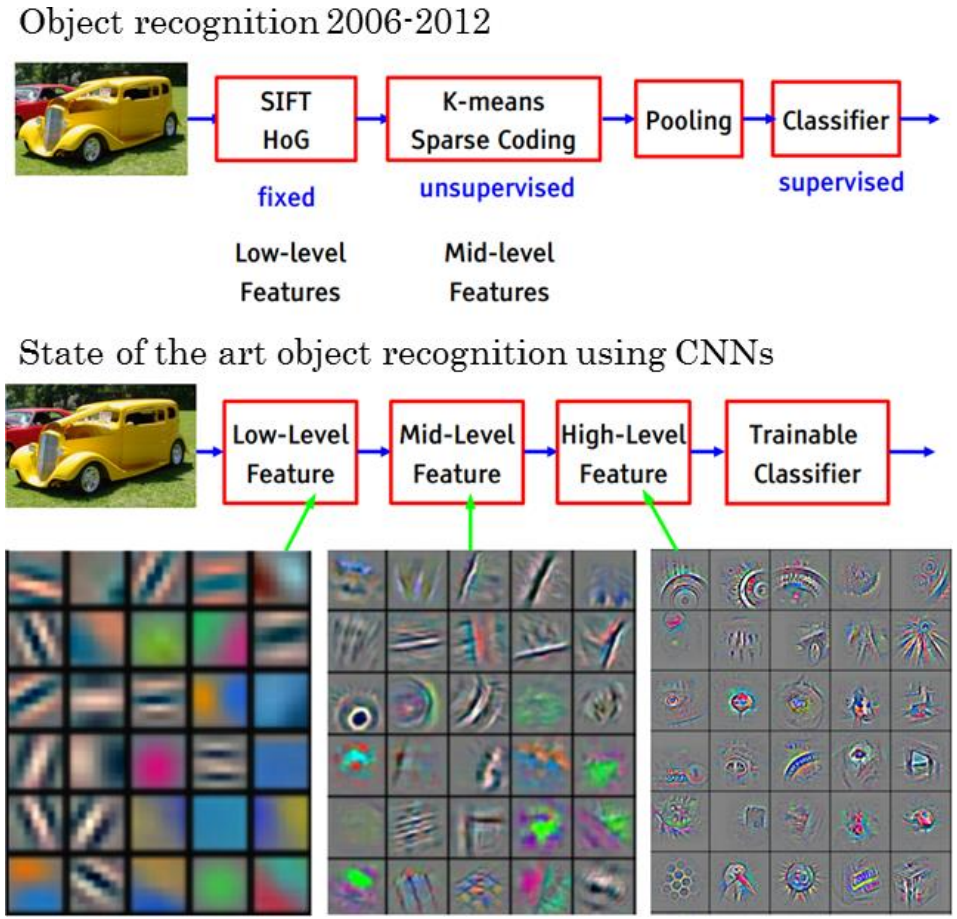
- Deep Learning
  - ➢Convolutional Neural Network
  - ➢Một số cấu trúc deep neural networks
  - ➢Giới thiệu nâng cao: RNN, Auto encoder.

History of Deep Learning



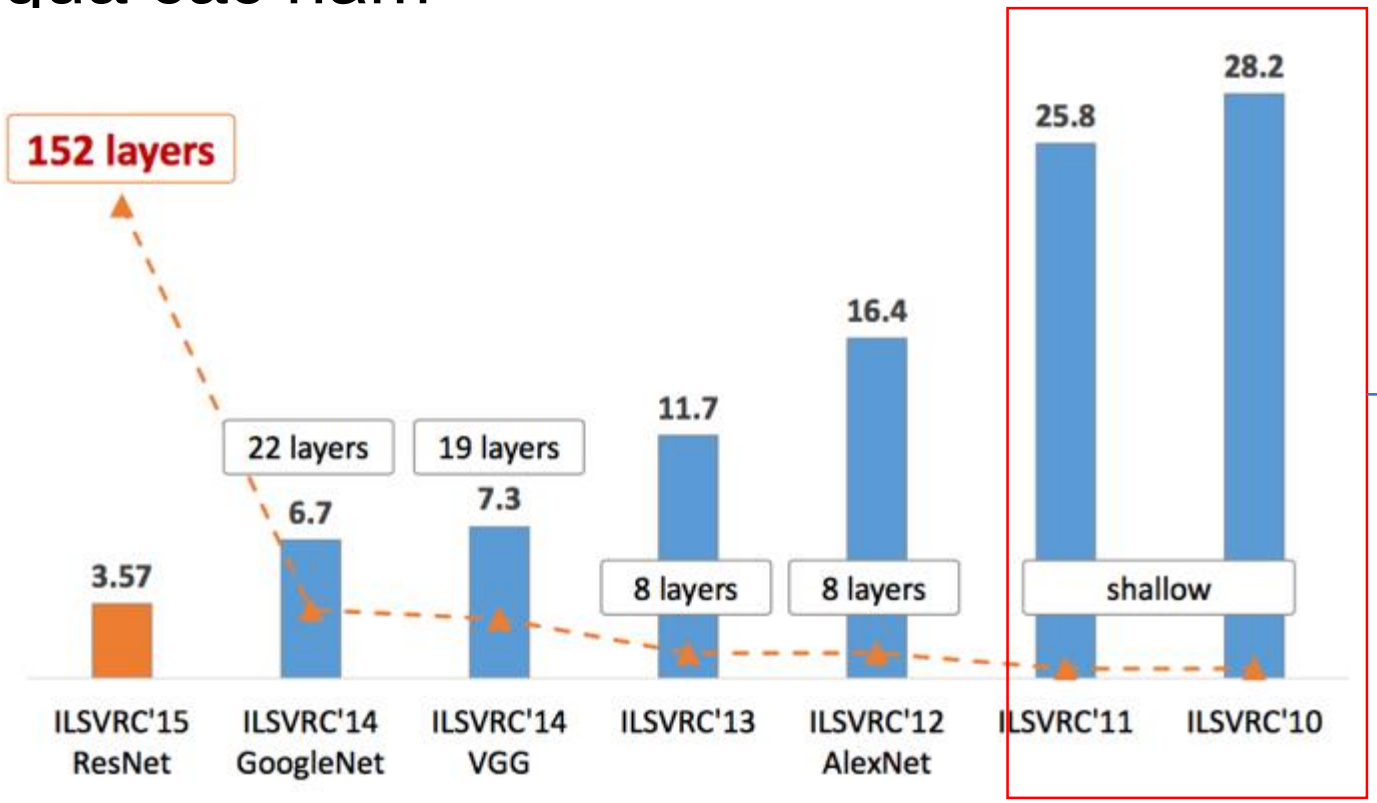
Why Deep Learning

- 2010-11: sử dụng hand-craft feature và các bộ phân lớp khác
- 2012-2016: ConvNets, được sử dụng như là feature learning
  - 2012: AlexNet: thành công mở ra kỷ nguyên mới
  - 2013: ZFNet: vài cải tiến trên AlexNet
  - 2014:
    - VGGNet: sâu hơn, đơn giản hơn
    - InceptionNet: mạng trong mạng, sâu hơn
  - 2015: ResNet: sâu hơn nữa, có cấu trúc giữ được thông tin ban đầu
  - 2016: Ensembled networks
  - 2017: Squeeze Network …



Cuộc thi nhận dạng ảnh

- Cuộc thi nhận dạng ảnh trên bộ dữ liệu lớn ImageNet, các đội thắng cuộc qua các năm



Handcraft features với các bộ phân lớp cổ điển.

Deep Convolutional Neural Network

Network structures and learning methods

Computer Vision Problems

Image Classification

64x64

Cat? (0/1)

Neural Style Transfer

Object detection

Andrew Ng

Deep Learning on large images

64x64 × 3

Cat? (0/1)

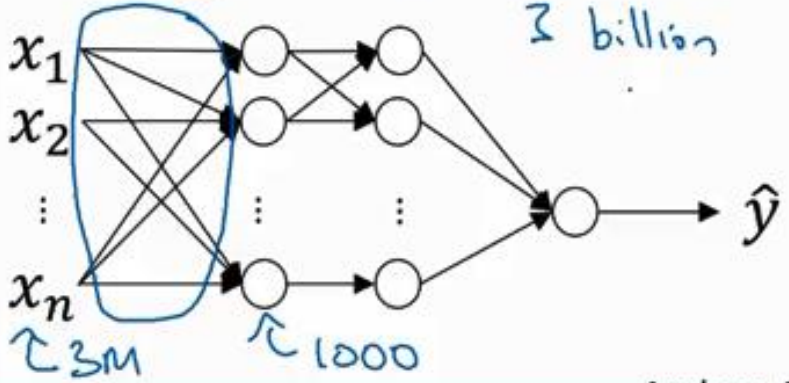
12288

1000 × 1000 × 3
= 3 million

$x \in \mathbb{R}^{3M}$

$W^{[1]}$ (1000, 3M)

3 billion



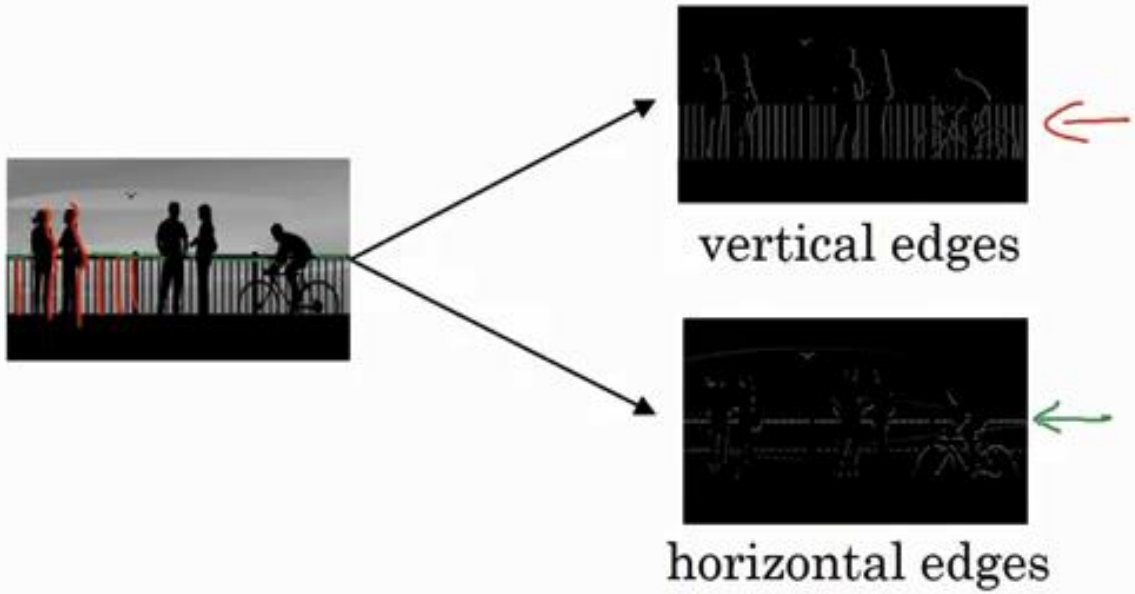
Andrew Ng

deeplearning.ai

Convolutional Neural Networks

Edge detection example

Computer Vision Problem



Andrew Ng

Vertical edge detection

$3\times1 + 1\times1 + 2\times1 + 0\times0 + 5\times0 + 7\times0 + 1\times-1 + 8\times-1 + 2\times-1 = -5$

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

6×6

"convolution"

*

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

3×3 filter

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

4×4

python: conv-forward
tensorflow: tf.nn.conv2d
keras: Conv2D

Andrew Ng

Vertical edge detection

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

6x6

*

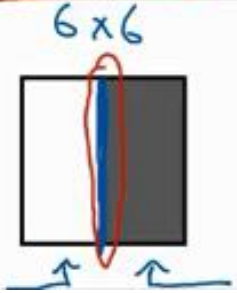
1	0	-1
1	0	-1
1	0	-1

3x3

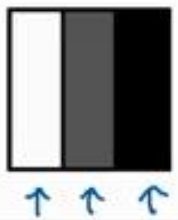
=

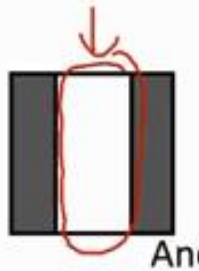
0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

4x4



*





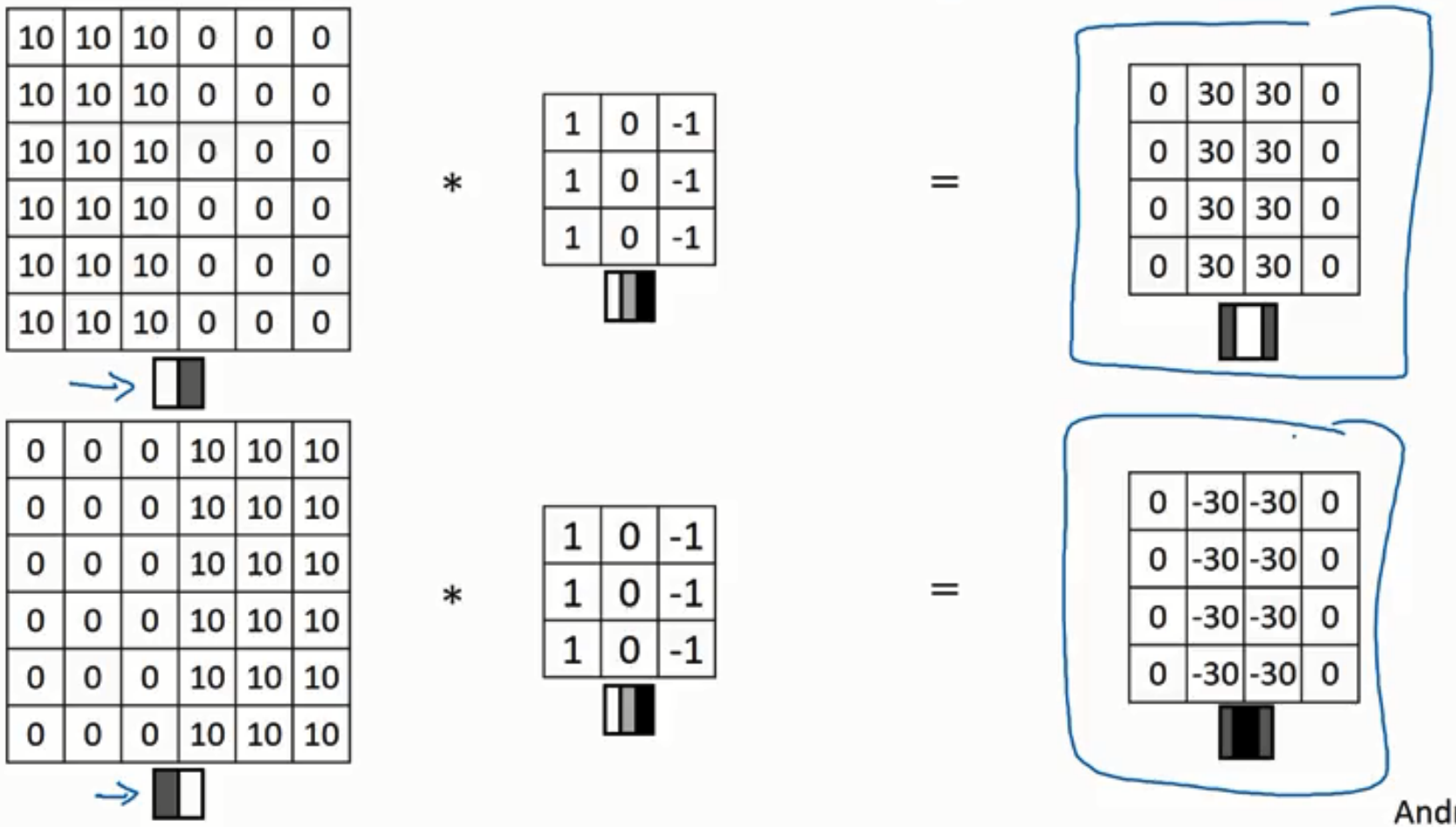
Andrew Ng

deeplearning.ai

Convolutional Neural Networks

More edge detection

Vertical edge detection examples



Andrew Ng

Vertical and Horizontal Edge Detection

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

Vertical

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

Horizontal

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

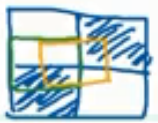
6 x 6

*

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

=

0	0	0	0
30	10	-10	-30
30	10	-10	-30
0	0	0	0



Andrew Ng

Learning to detect edges

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

1	0	-1
2	0	-2
1	0	-1

Sobel filter

3	0	-3
10	0	-10
3	0	-3

Scharr filter

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

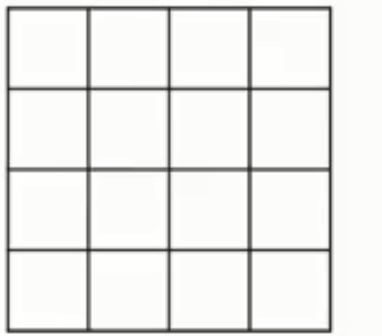
convolution

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

3×3

=

45°
70°
73°



3×3

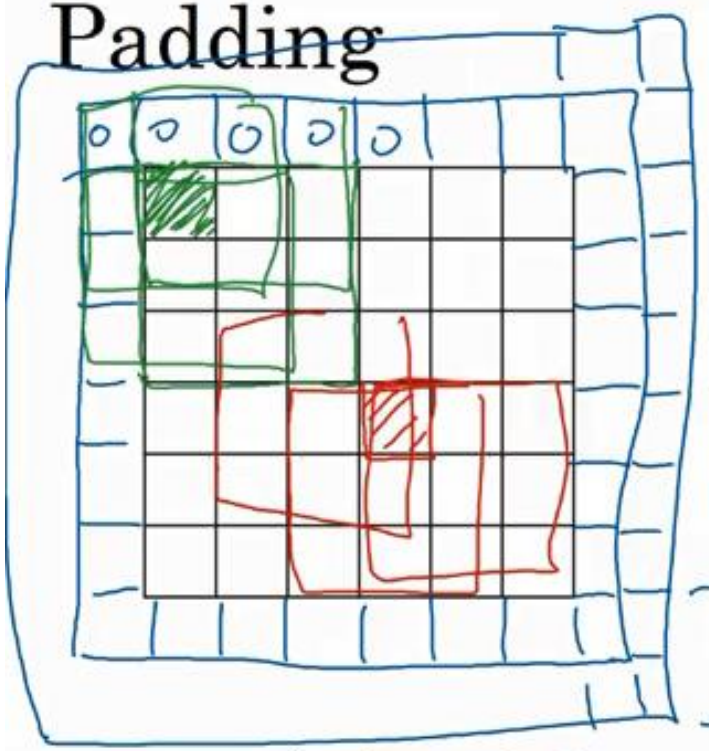
Andrew Ng

deeplearning.ai

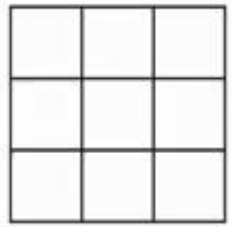
Convolutional Neural Networks

Padding

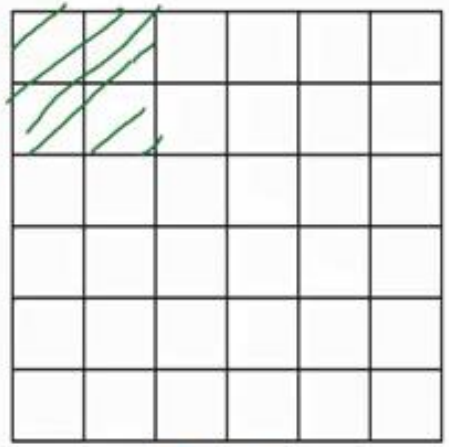
Padding



- shrinky output
- throw away info from edge



3×3
f×f



6×6

p=2

6×6 → 8×8
n×n

n-f+1 × n-f+1
6-3+1=4

~~4×4~~

p = padding = 1

n+2p-f+1 × n+2p-f+1
6+2-3+1 × ___ = 6×6

Andrew Ng

Valid and Same convolutions

↳ no padding

"Valid": $n \times n \quad * \quad f \times f \quad \rightarrow \quad n-f+1 \times n-f+1$

$6 \times 6 \quad * \quad 3 \times 3 \quad \rightarrow \quad 4 \times 4$

"Same": Pad so that output size is the same as the input size.

$n+2p-f+1 \times n+2p-f+1$

$n+2p-f+1 = n \Rightarrow p = \frac{f-1}{2}$

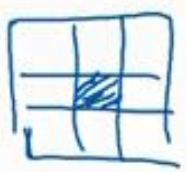
$3 \times 3 \quad p = \frac{3-1}{2} = 1$

$5 \times 5 \quad p = 2$

$f = 5$

f is usually odd

1×1
3×3
5×5
7×7



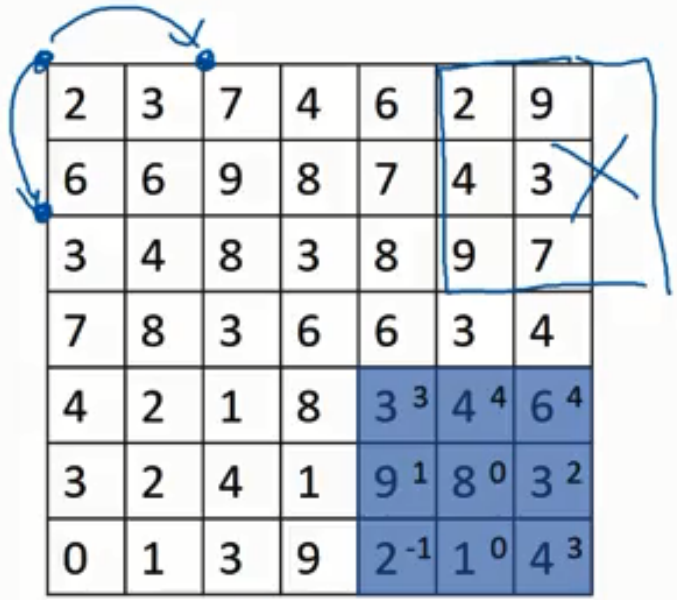
Andrew Ng

deeplearning.ai

Convolutional Neural Networks

Strided convolutions

Strided convolution



2	3	7	4	6	2	9
6	6	9	8	7	4	3
3	4	8	3	8	9	7
7	8	3	6	6	3	4
4	2	1	8	3^{3}	4^{4}	6^{4}
3	2	4	1	9^{1}	8^{0}	3^{2}
0	1	3	9	2^{-1}	1^{0}	4^{3}

7×7

*

3	4	4
1	0	2
-1	0	3

3×3

=

91	100	83
69	91	127
44	72	74

3×3

stride = 2

$\lfloor z \rfloor = \text{floor}(z)$

$n \times n$ * $f \times f$

padding p stride s

$s = 2$

$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$

$$\frac{7+0-3}{2} + 1 = \frac{4}{2} + 1 = 3$$

Andrew Ng

Summary of convolutions

$n \times n$ image $\quad f \times f$ filter

padding p $\quad$ stride s

Output size:

$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$

Andrew Ng

Technical note on cross-correlation vs. convolution

Convolution in math textbook:

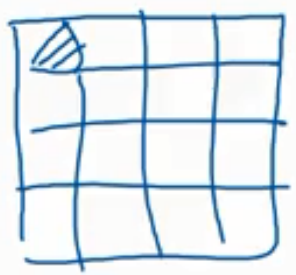
2	3	7	4	6	2
6	6	9	8	7	4
3	4	8	3	8	9
7	8	3	6	6	3
4	2	1	8	3	4
3	2	4	1	9	8

*

3	4	5
1	0	2
-1	9	7

7	2	5
9	0	4
-1	1	3

=



$(A * B) * C = A * (B * C)$

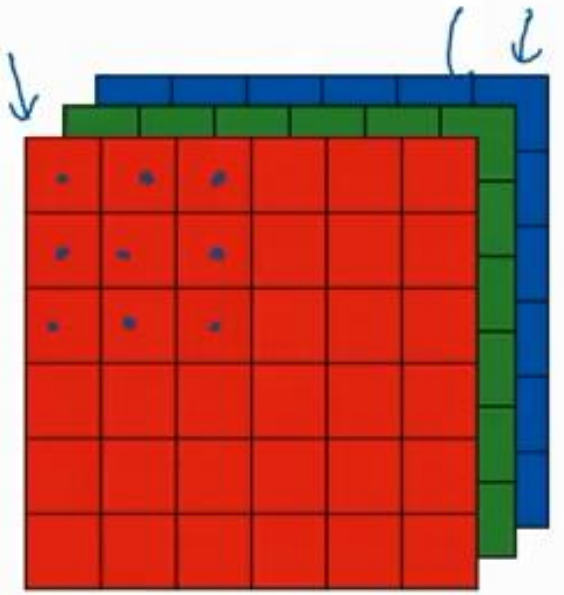
Andrew Ng

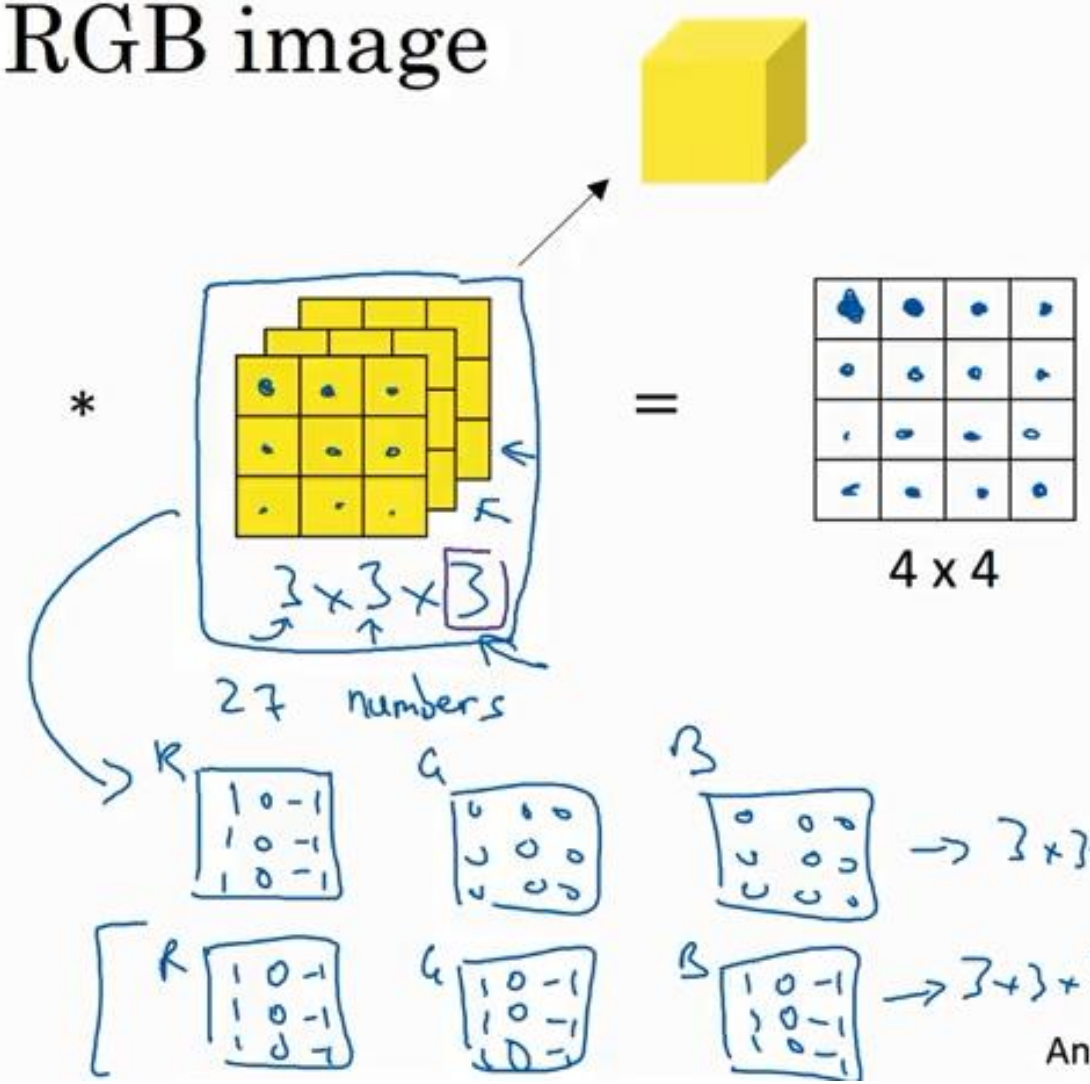
deeplearning.ai

Convolutional Neural Networks

Convolutions over volumes

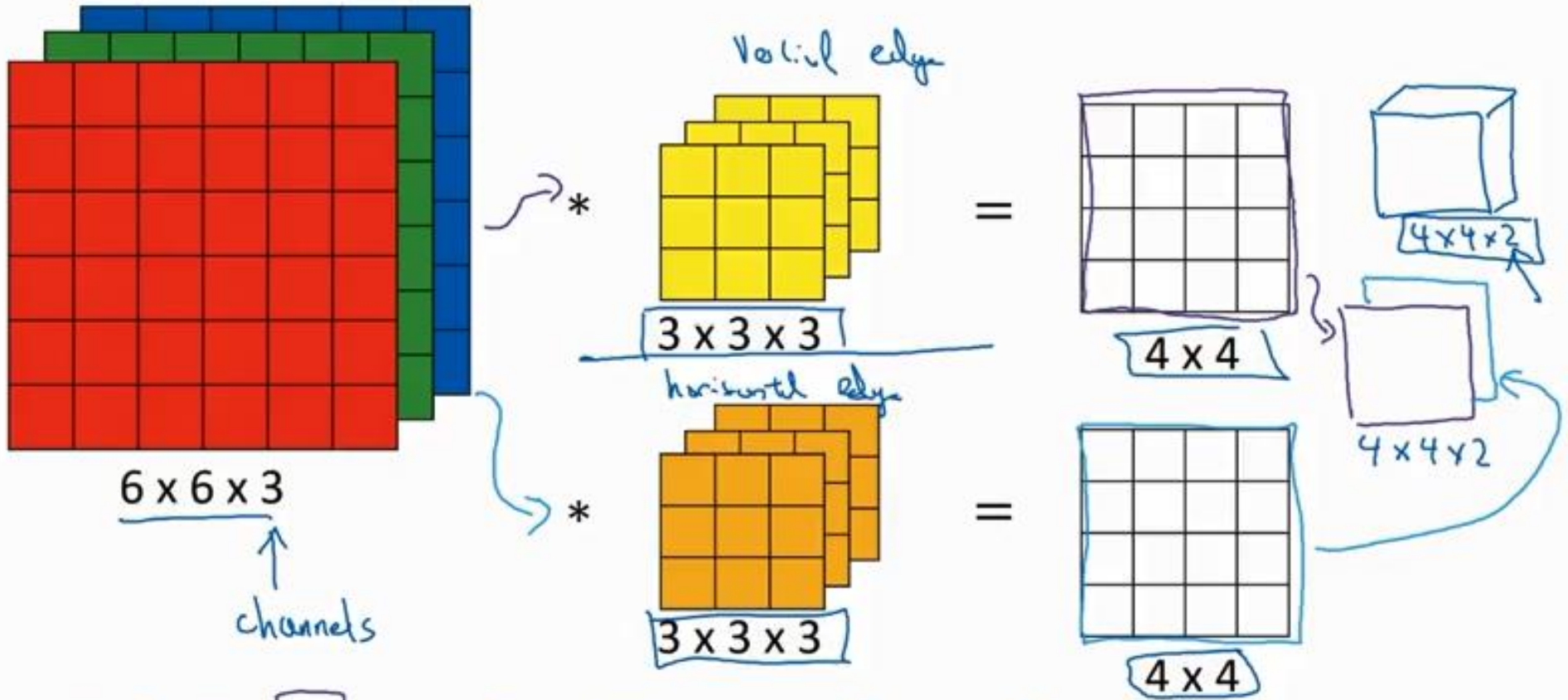
Convolutions on RGB image





Andrew Ng

Multiple filters



Summary: $n \times n \times n_c$ * $f \times f \times n_c$ → $(n-f+1) \times (n-f+1) \times n_c'$

$6 \times 6 \times 3$ * $3 \times 3 \times 3$ → $4 \times 4 \times 2$ (# filters)

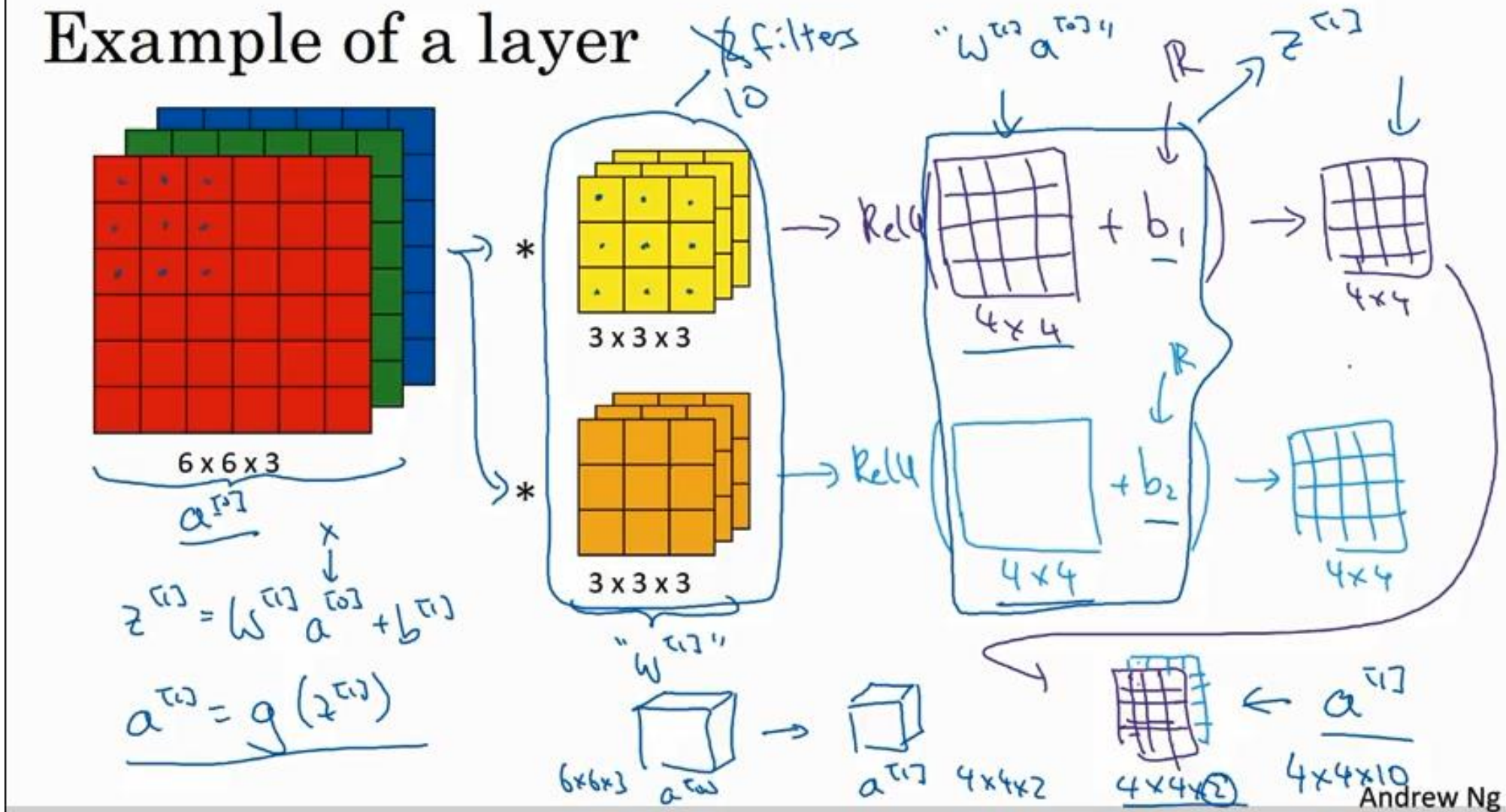
Andrew Ng

deeplearning.ai

Convolutional Neural Networks

One layer of a convolutional network

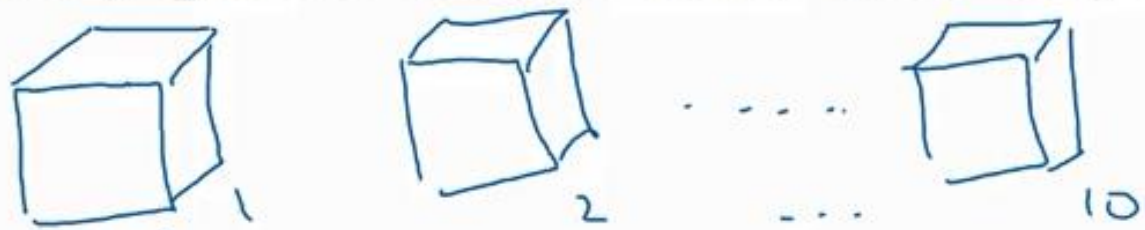
Example of a layer



Andrew Ng

Number of parameters in one layer

If you have 10 filters that are 3 x 3 x 3 in one layer of a neural network, how many parameters does that layer have?



Andrew Ng

Summary of notation

If layer l is a convolution layer:

$f^{[l]}$ = filter size

$p^{[l]}$ = padding

$s^{[l]}$ = stride

$n_c^{[l]}$ = number of filters

→ Each filter is: $f^{[l]} \times f^{[l]} \times n_c^{[l-1]}$

Activations: $a^{[l]} \rightarrow n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

Weights: $f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$

bias: $n_c^{[l]} - (1,1,1,n_c^{[l]})$ ← #filters in layer l.

Input: $n_H^{[l-1]} \times n_W^{[l-1]} \times n_c^{[l-1]}$ ←

Output: $n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$ ←

$$n_{H/W}^{[l]} = \left\lfloor \frac{n_{H/W}^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$$

$A^{[l]} \rightarrow m \times n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

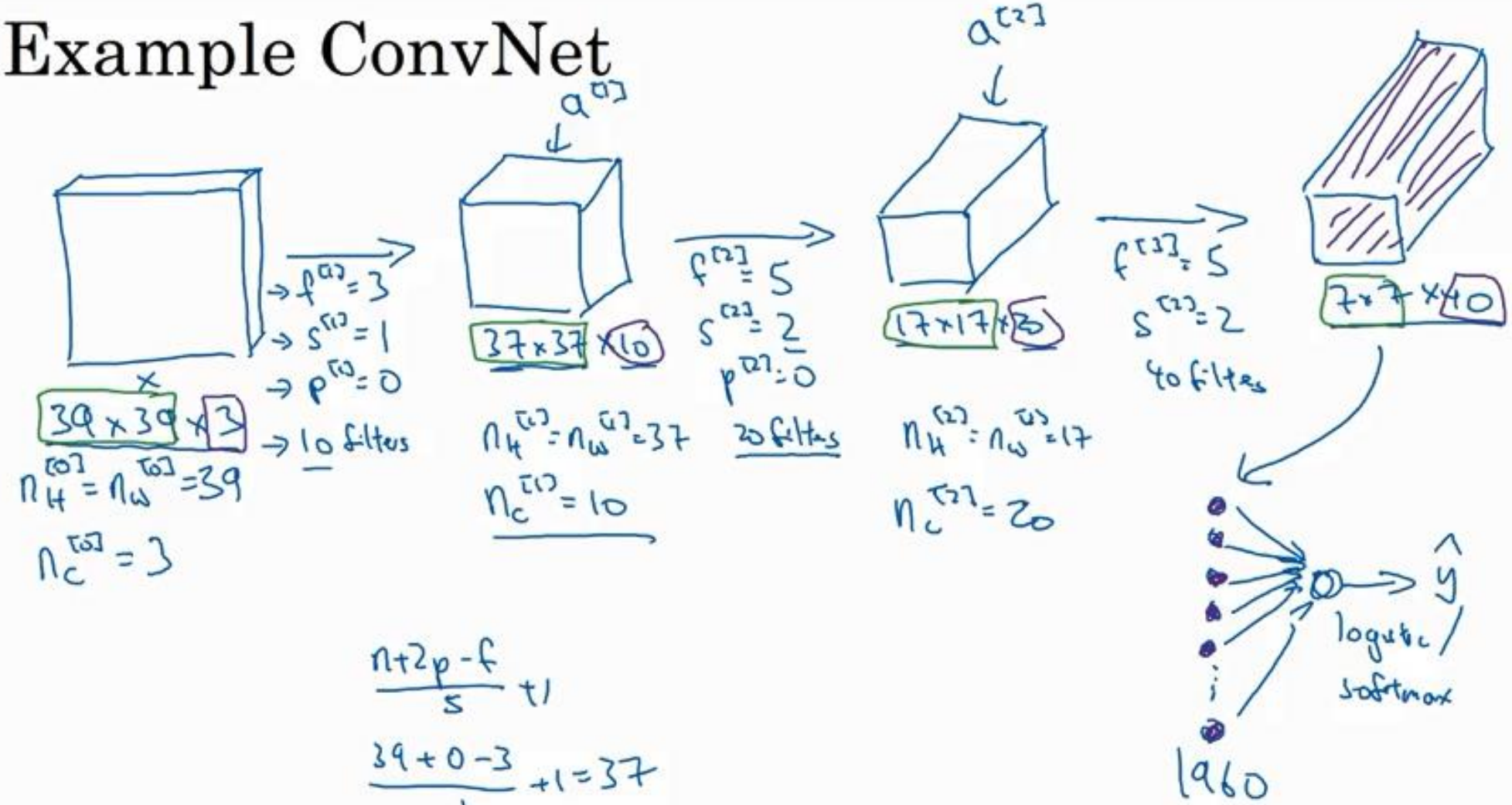
$n_c \times n_H \times n_W$

deeplearning.ai

Convolutional Neural Networks

A simple convolution network example

Example ConvNet



$$\frac{n+2p-f}{s}+1$$

$$\frac{39+0-3}{1}+1=37$$

Andrew Ng

Types of layer in a convolutional network:

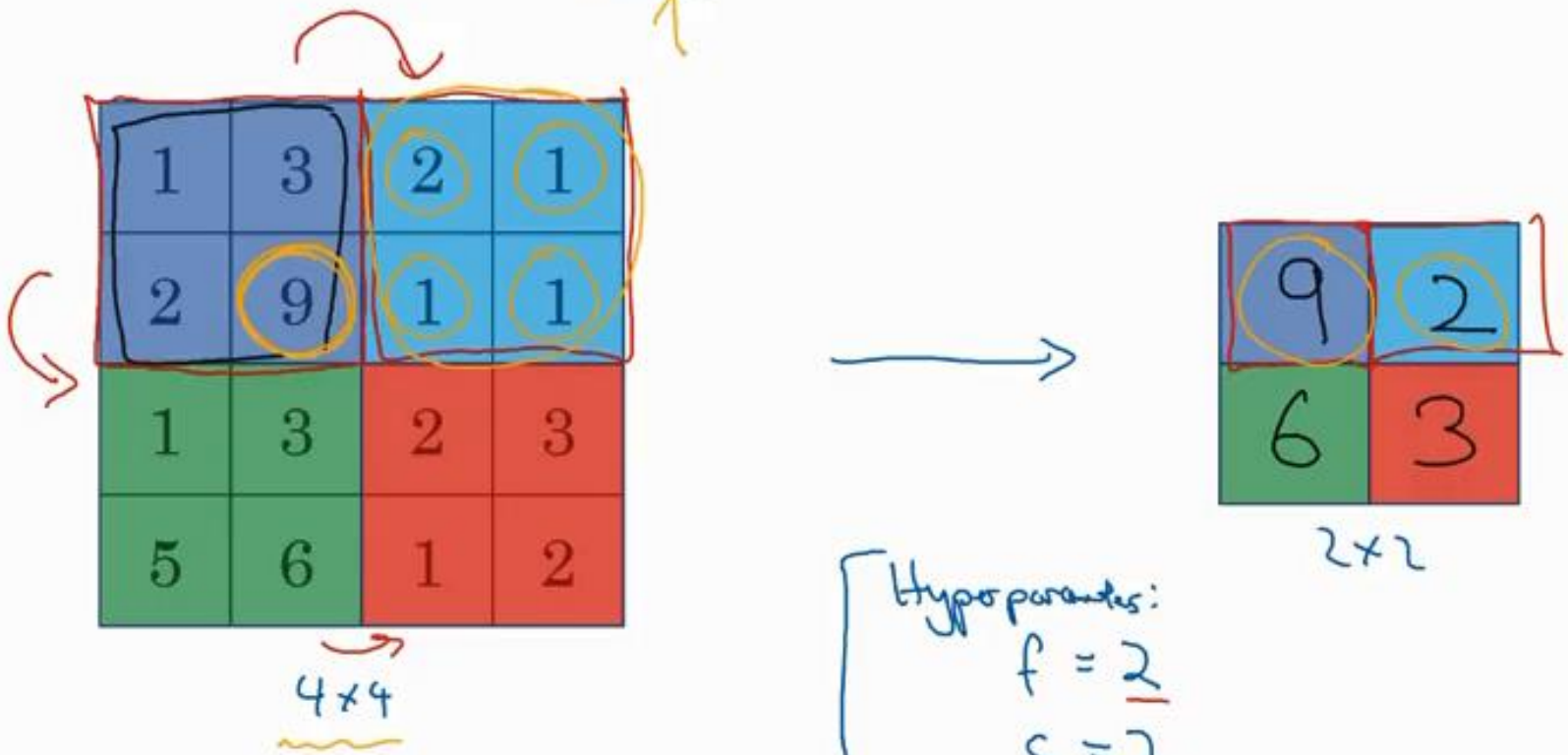
- Convolution (CONV) ←
- Pooling (POOL) ←
- Fully connected (FC) ←

deeplearning.ai

Convolutional Neural Networks

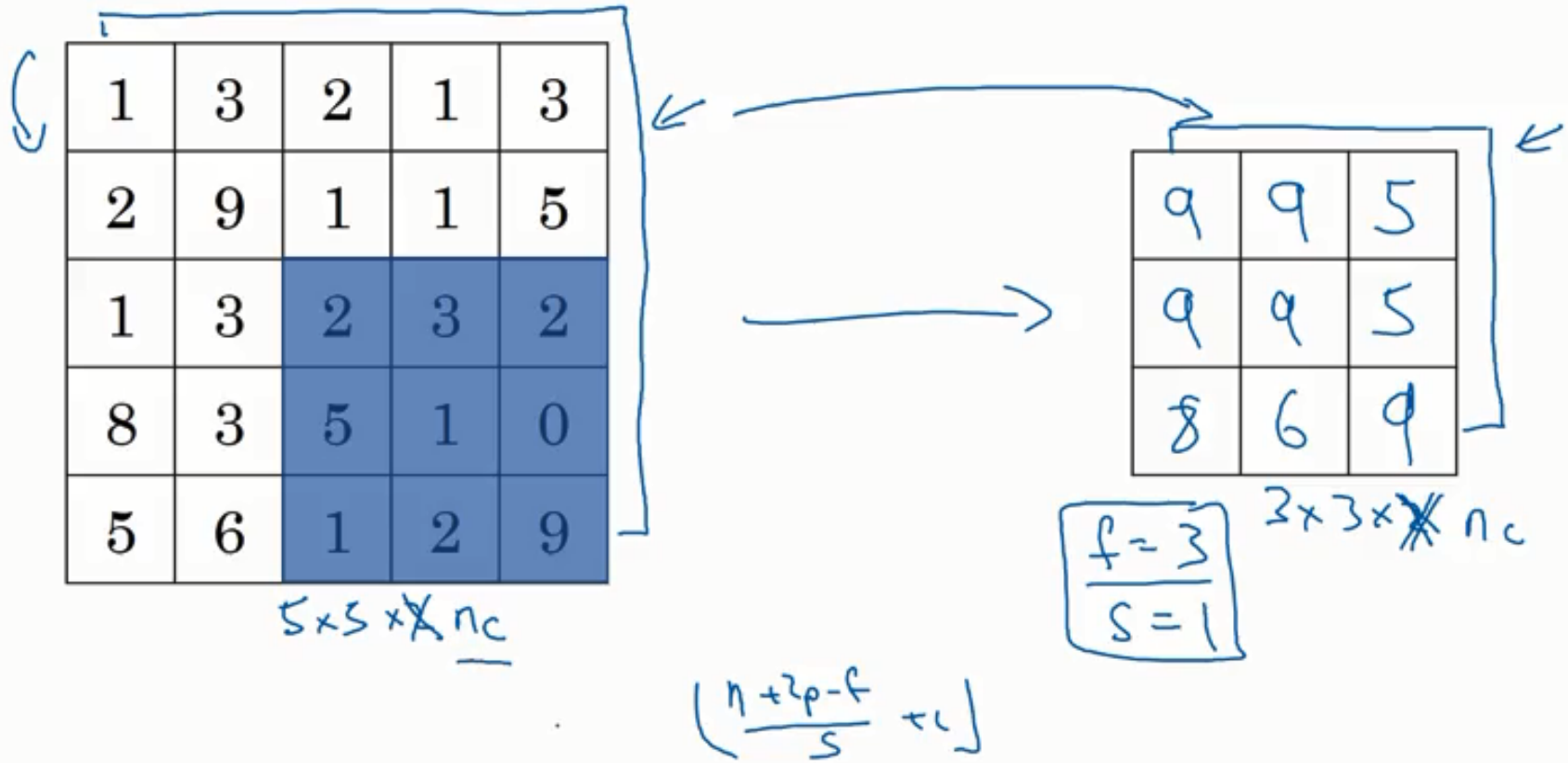
Pooling layers

Pooling layer: Max pooling



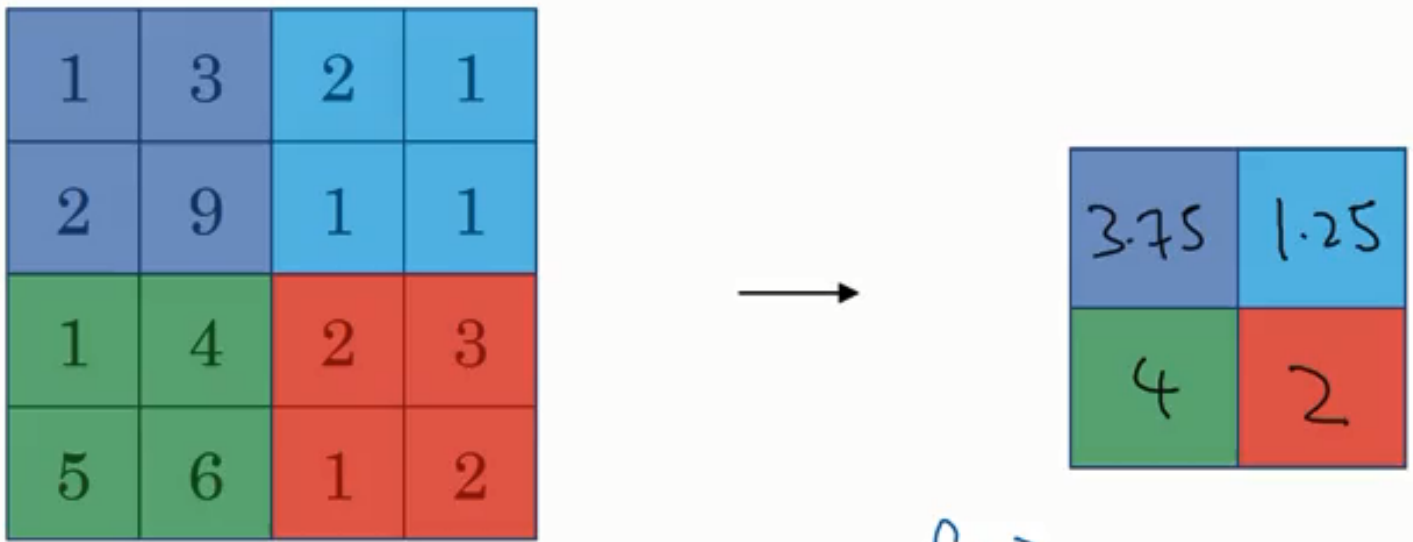
Andrew Ng

Pooling layer: Max pooling



Andrew Ng

Pooling layer: Average pooling



$f = 2$

$s = 2$

$7 \times 7 \times 1000 \rightarrow 1 \times 1 \times 1000$

Andrew Ng

Summary of pooling

Hyperparameters:

f : filter size

s : stride

Max or average pooling

~~p : padding~~

No parameters to learn!

$f=2, s=2$

$f=3, s=2$

$n_H \times n_W \times n_C$

$\downarrow$

$\left\lfloor \frac{n_H - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n_W - f}{s} + 1 \right\rfloor$

$\times n_C$

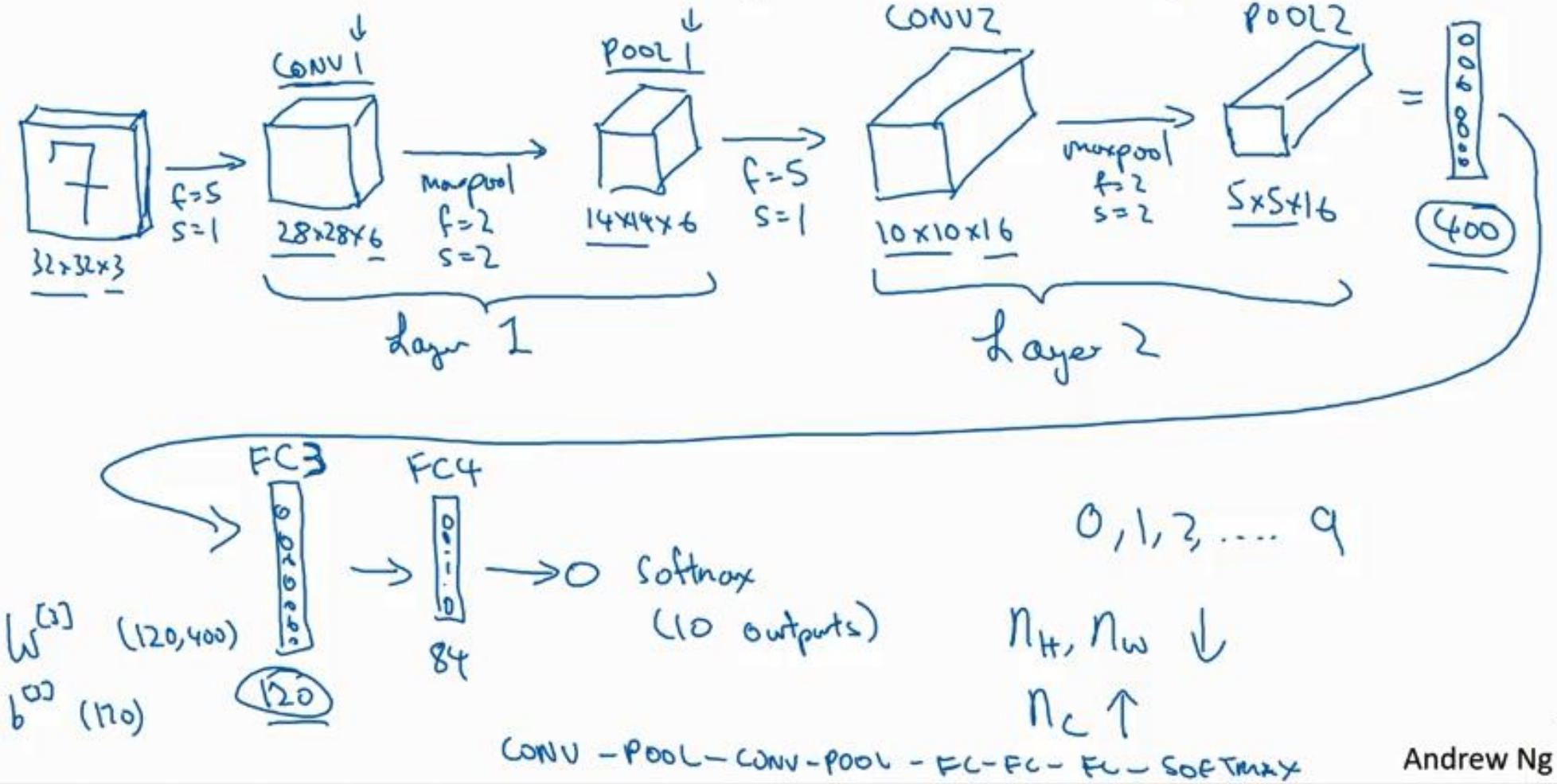
Andrew Ng

deeplearning.ai

Convolutional
Neural Networks

Convolutional neural network example

Neural network example (LeNet-5)



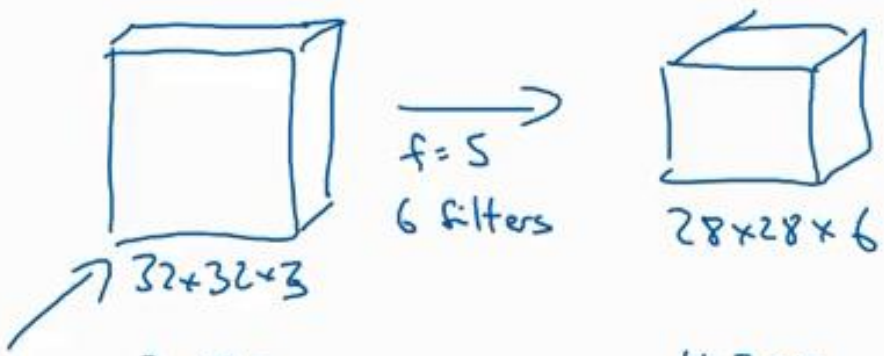
Andrew Ng

deeplearning.ai

Convolutional Neural Networks

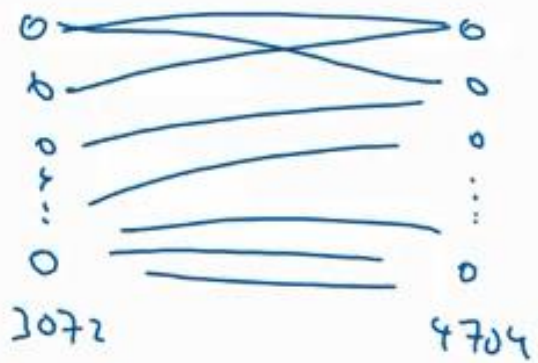
Why convolutions?

Why convolutions



3,072

4,704



5×5 − 25

26

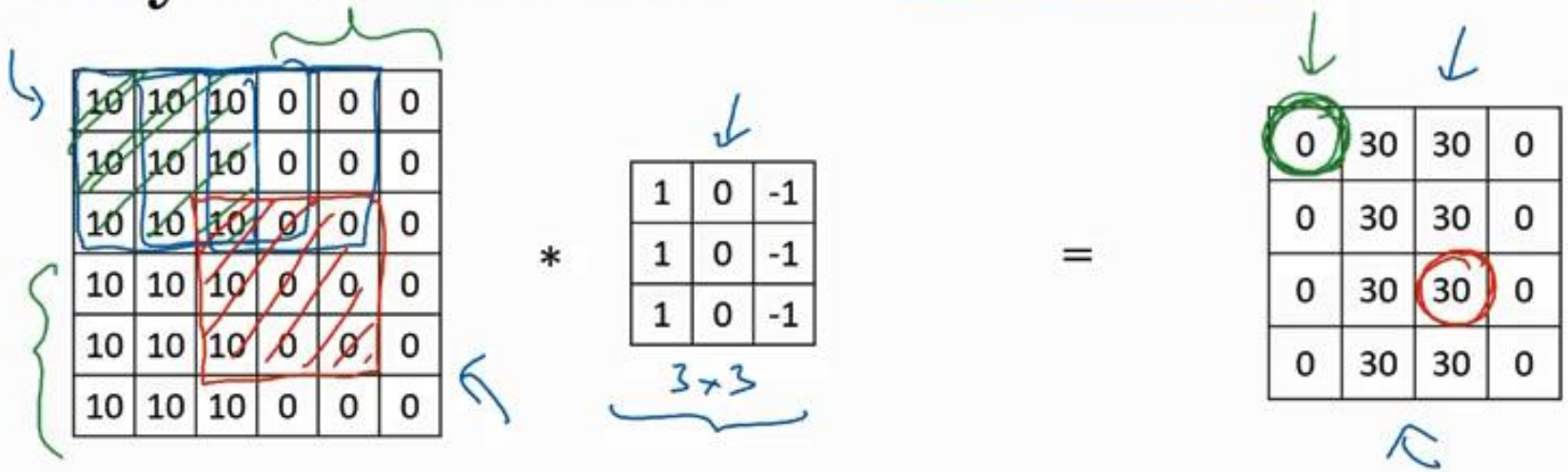
6 × 26 = 156 parameters

3,072 × 4,704 ≈ 14M

Andrew Ng

Why convolutions

translation invariance



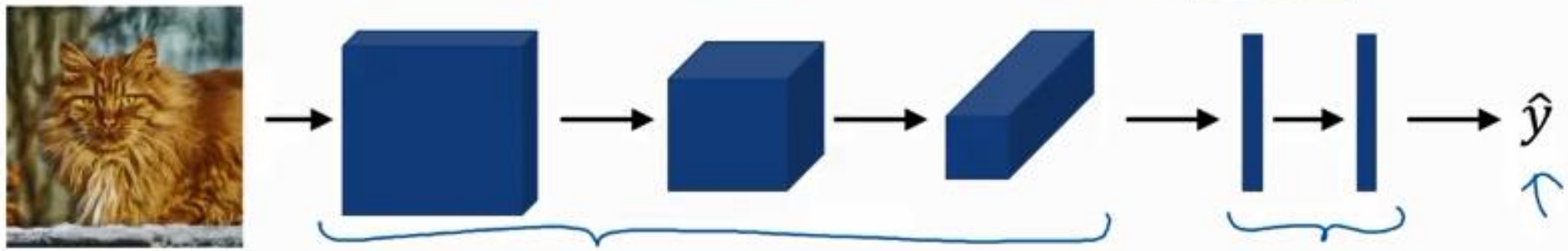
Parameter sharing: A feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image.

→ **Sparsity of connections:** In each layer, each output value depends only on a small number of inputs.

Andrew Ng

Putting it together

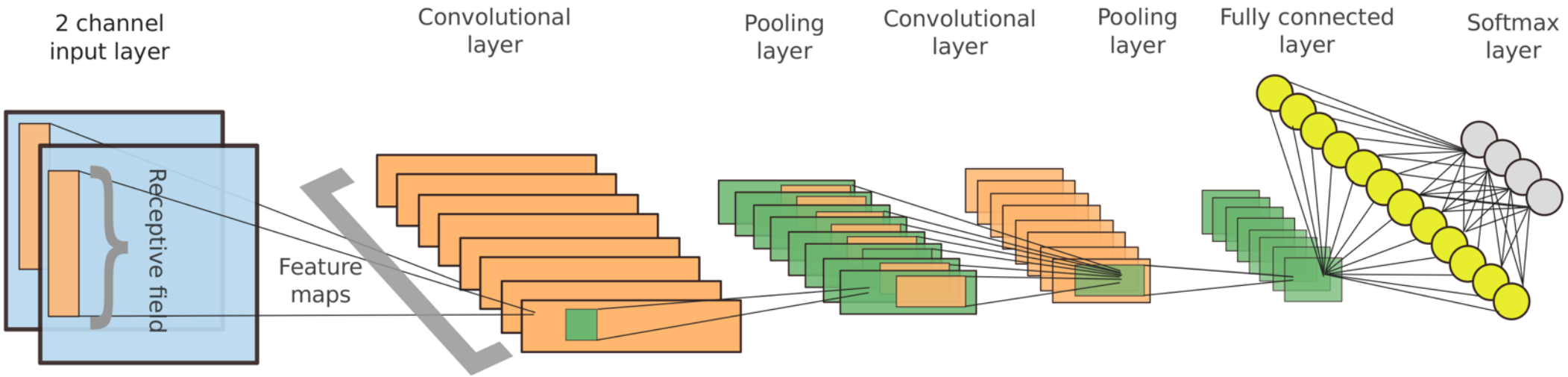
Training set $(x^{(1)}, y^{(1)}) \ldots (x^{(m)}, y^{(m)})$.

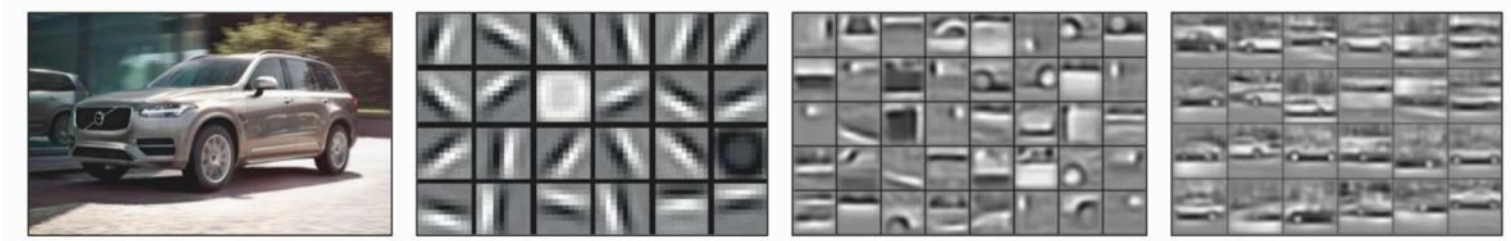


Cost $J = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$

Use gradient descent to optimize parameters to reduce J

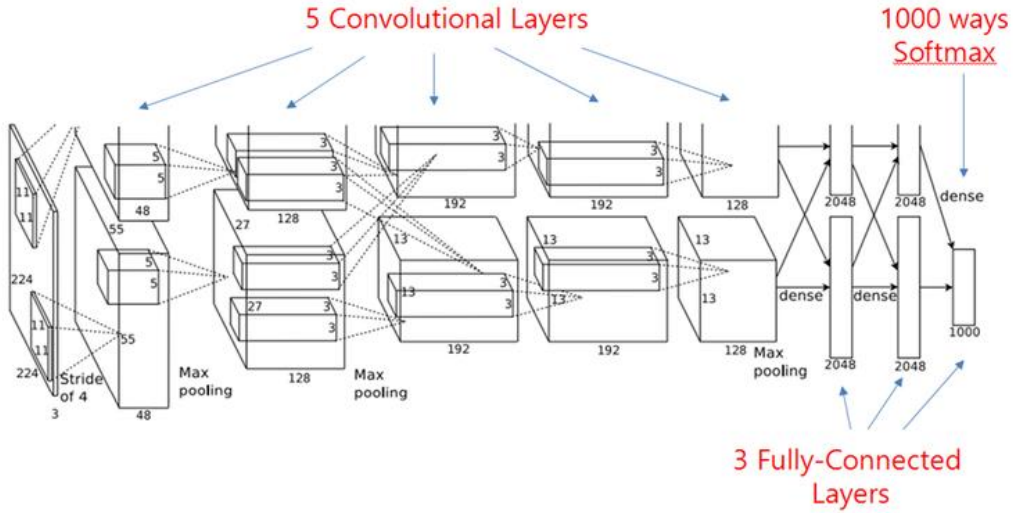
Cấu trúc cơ bản





Case Study: AlexNet

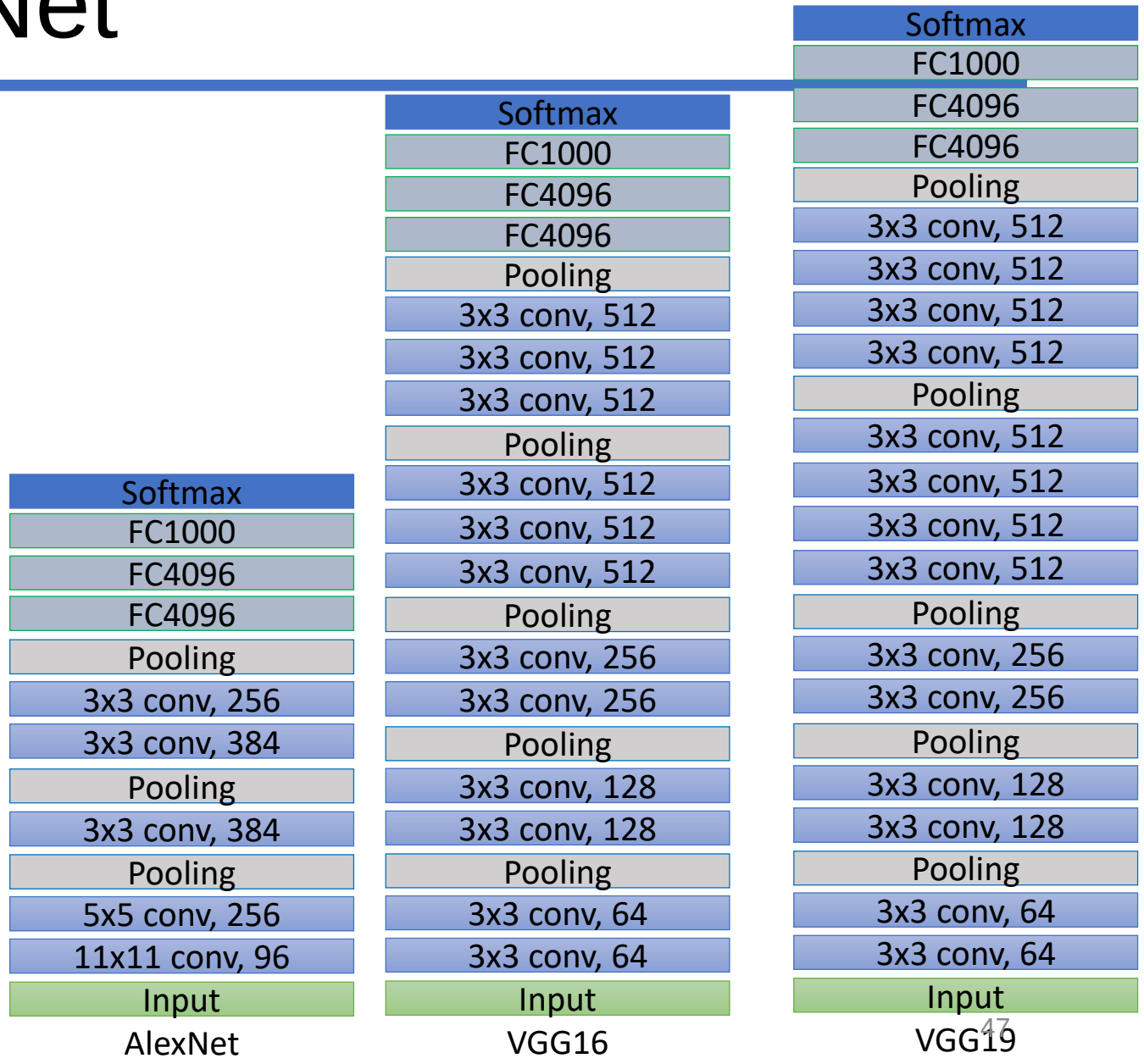
- Full (simplified) AlexNet architecture:
- [227x227x3] INPUT
- [55x55x96] CONV1: 96 11x11 filters at stride 4, pa
- [27x27x96] MAX POOL1: 3x3 filters at stride 2
- [27x27x96] NORM1: Normalization layer
- [27x27x256] CONV2: 256 5x5 filters at stride 1, pa
- [13x13x256] MAX POOL2: 3x3 filters at stride 2
- [13x13x256] NORM2: Normalization layer
- [13x13x384] CONV3: 384 3x3 filters at stride 1, pa
- [13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
- [13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
- [6x6x256] MAX POOL3: 3x3 filters at stride 2
- [4096] FC6: 4096 neurons
- [4096] FC7: 4096 neurons
- [1000] FC8: 1000 neurons (class scores)



[Krizhevsky et al. 2012]

Case Study: VGGNet

- Bộ lọc nhỏ hơn, mạng sâu hơn *[Simonyan and Zisserman, 2014]*
- 8 layers (AlexNet) -> 16 - 19 layers (VGG16Net)
- Chồng 3 lớp 3x3 conv (stride 1) layers có receptive field tương đương 1 lớp 7x7 conv
- Càng sâu
  - Càng có khả năng phi tuyến
  - Ít tham số hơn



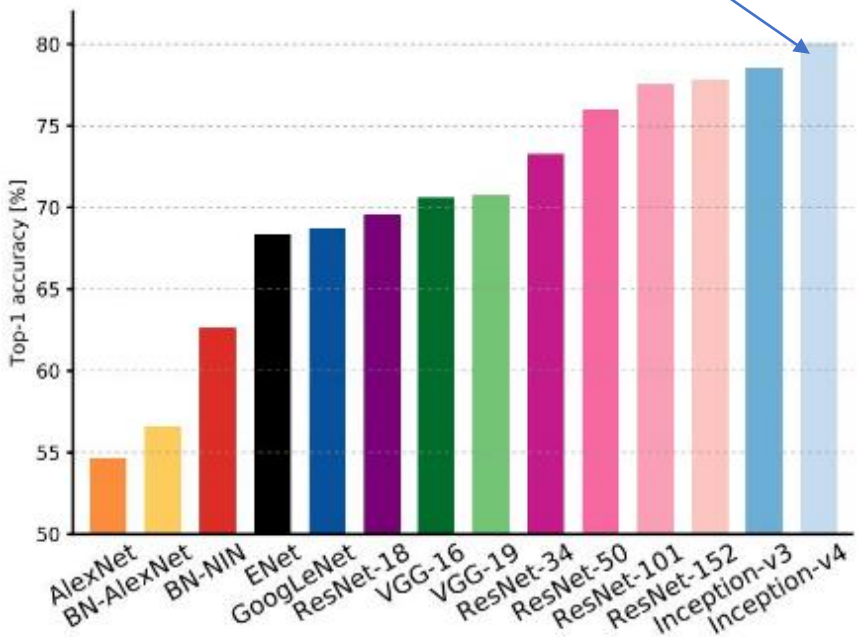
Phân tích so sánh các mô hình

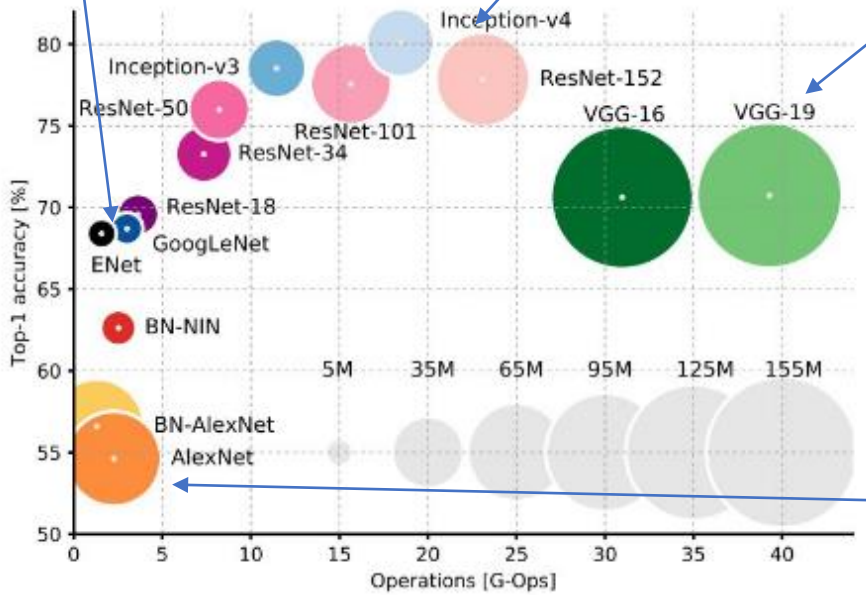
IncepV4:ResNet+Inception

GoogleNet: hiệu quả

ResNet: Mức độ tính toán trung bình, độ chính xác cao

VGG: Chiếm nhiều bộ nhớ, nhiều tác vụ tính toán





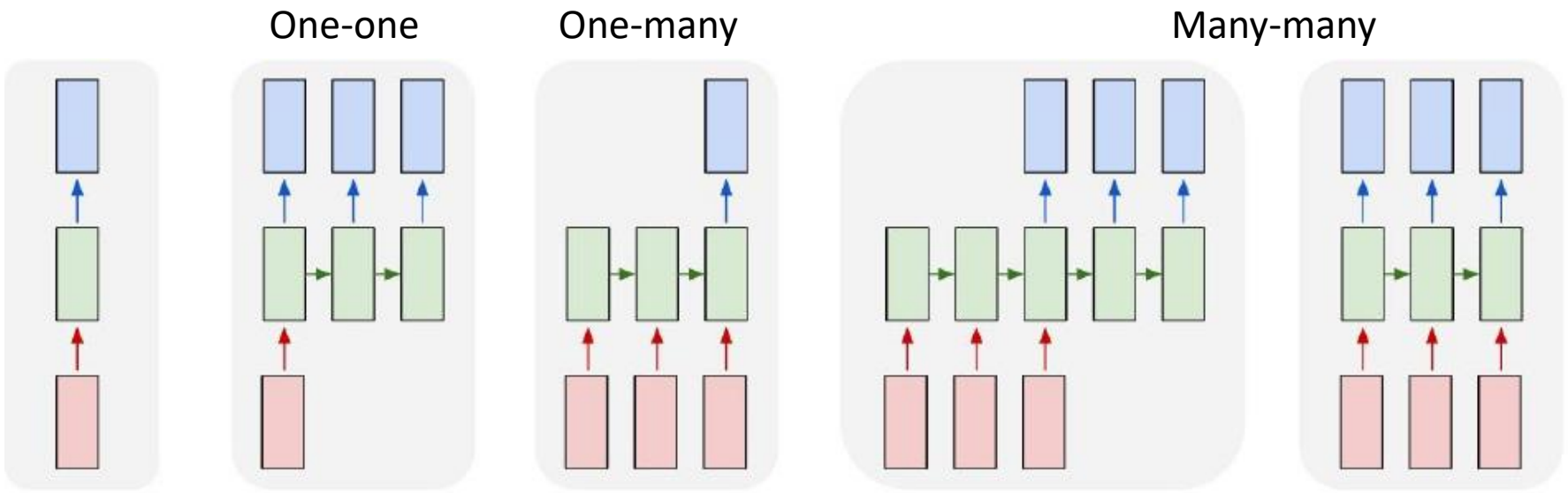
AlexNet: nhỏ, nhiều bộ nhớ, độ chính xác thấp

An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Deep Recurent Neural Network

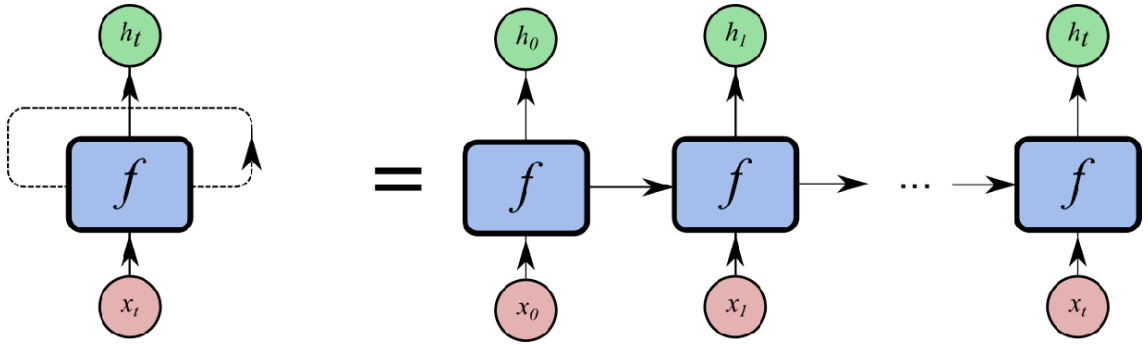
Network structures and learning methods

Sequence Learning



Recurrent Neural Network

- Sử dụng để dự đoán các thông tin theo chuỗi, theo thời gian.
- Xử lý các vector đầu vào x bằng các weight, neuron tương tự nhau trên tất cả các time steps.



$$h_t = f_W(h_{t-1}, x_t)$$

new state

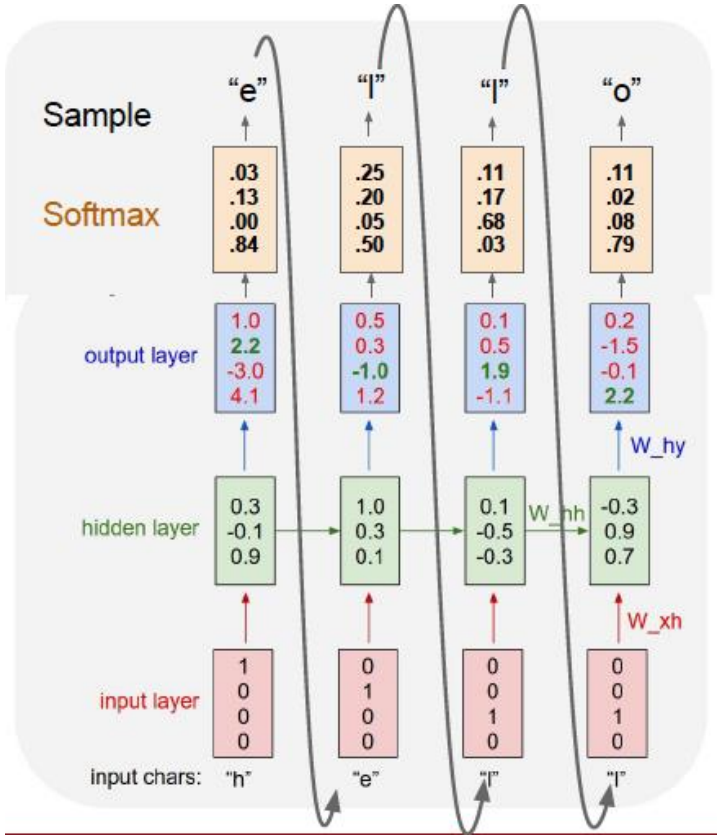
some function with parameters W

old state

input vector at some time step

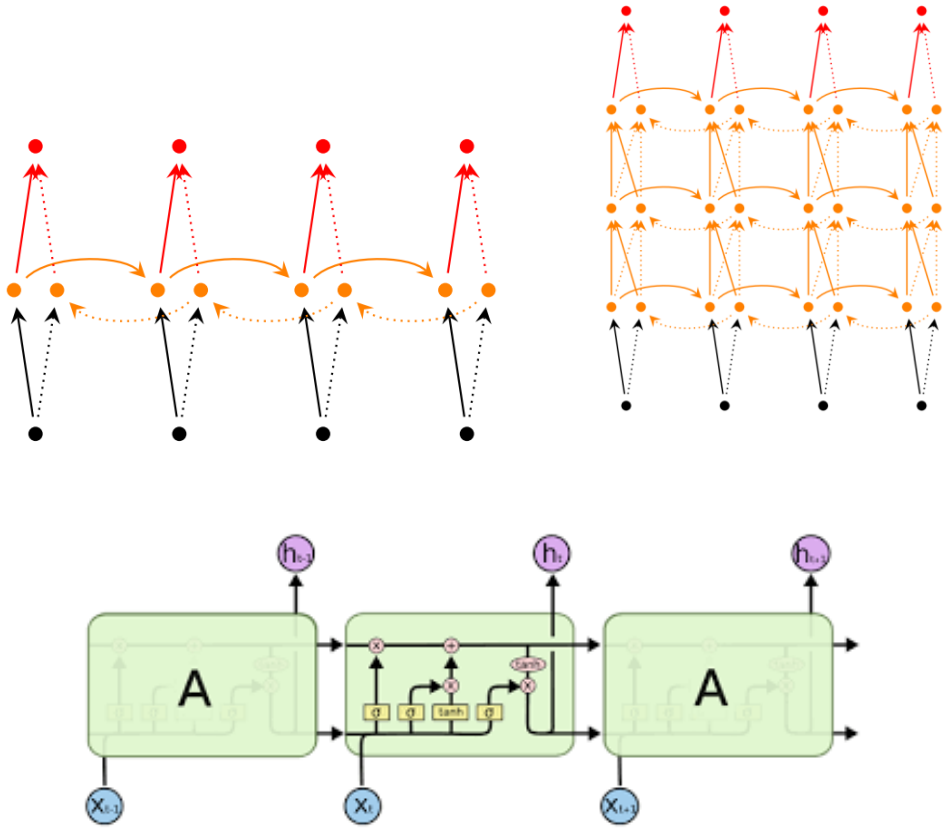
Bài toán với chuỗi ký tự

- Sampling các chuỗi ký tự
- Từ vựng: [h,e,l,o]
- Sampling các ký tự tiếp theo kết hợp với đầu ra của time step trước



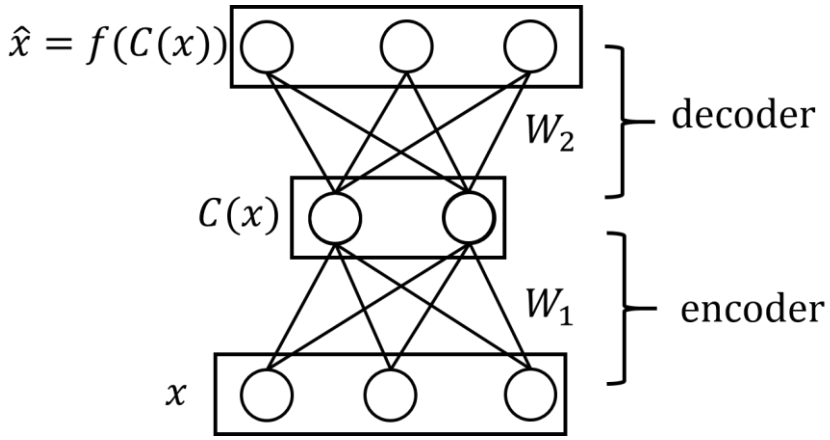
Các cấu trúc RNN khác

- Kết nối hai chiều (Bidirectional):
  - Đầu vào tại thời điểm t phụ thuộc vào
    - Thành phần trước trong chuỗi.
    - Thành phần sau thời điểm t trong chuỗi.
- Deep (Bidirectional) RNNs:
  - Tương tự như kết nối hai chiều.
  - Có nhiều lớp tại mỗi timestep.
- Long short-term memory

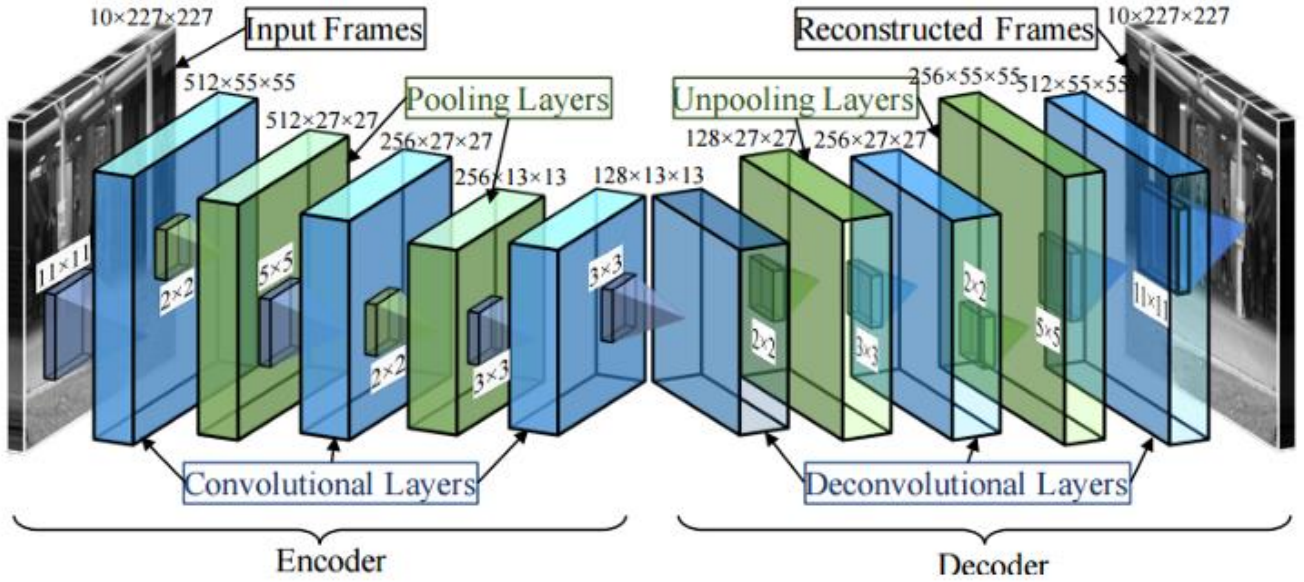


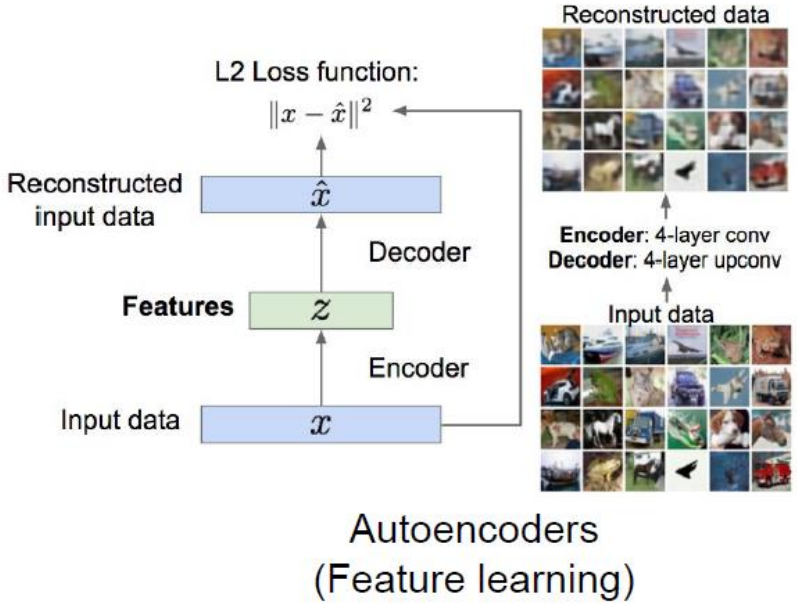
Auto Encoder

- Unsupervised learning: không có nhãn phục vụ huấn luyện mạng như các cấu trúc trước
- Unsupervised Encode Input Vector
  - Tái tạo input:
  - Số lượng node/neuron đầu vào và đầu ra như nhau.
  - Huấn luyện sao cho đầu ra sấp xỉ đầu vào.
  - Các lớp giữa:
    - Số neuron ít hơn: undercomplete autoencoder
    - Số neuron nhiều hơn: overcomplete autoencoder
  - Input vector được encode bằng neuron lớp giữa



Deep Convolutional Autoencoder





Autoencoders
(Feature learning)

Tổng kết Deep Learning

- Dễ dàng xây dựng được mô hình biểu diễn, tính toán
- Mô hình đa dạng, có thể sử dụng trong nhiều bài toán khác nhau (phân lớp, phân cụm, phân tích chuỗi)
- Sử dụng trong nhiều ứng dụng khác nhau: ảnh, text, voice …

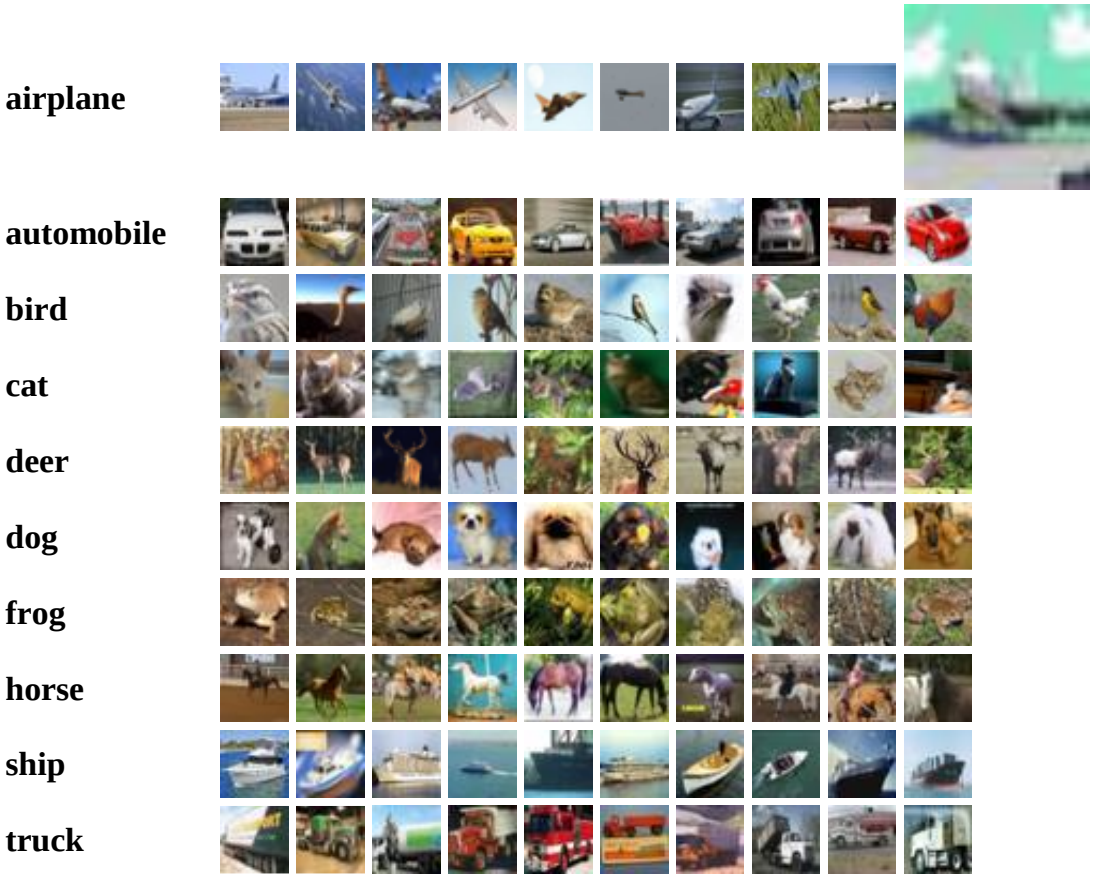
- Vấn đề hộp đen (blackbox): không giải thích được, không suy diễn được
- Cần nhiều dữ liệu tính toán
- Huấn luyện mô hình phức tạp

CIFAR10

CIFAR

Demo nhận dạng đối tượng

- Bài toán nhận dạng đối tượng
- Dữ liệu: CIFAR-10
  - 10 lớp ảnh màu 32x32
  - Nhiều góc độ, vùng của đối tượng
  - Train 40000, validate 10000, test 10000
- Các bước thiết kế, huân luyện và nhận dạng theo hướng dẫn trên
  - https://github.com/ai-academy-vn/BasicML



https://www.cs.toronto.edu/~kriz/cifar.html

Bài tập

- Quan sát và làm lại mẫu bài tập trên github
- Tìm và thay thế các phần thiếu hoặc thay thế các tham số, các phương pháp huấn luyện
- Thử với các bài toán trong các buổi học trước
  - Handcraft feature + bộ phân lớp cơ bản
  - Sử dụng deep learning
  - So sánh trên cùng bộ dữ liệu

Q&A

Thank you!