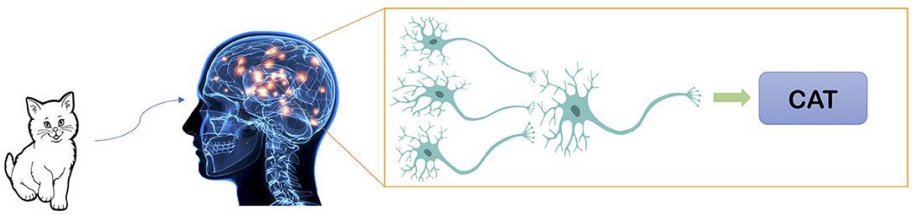
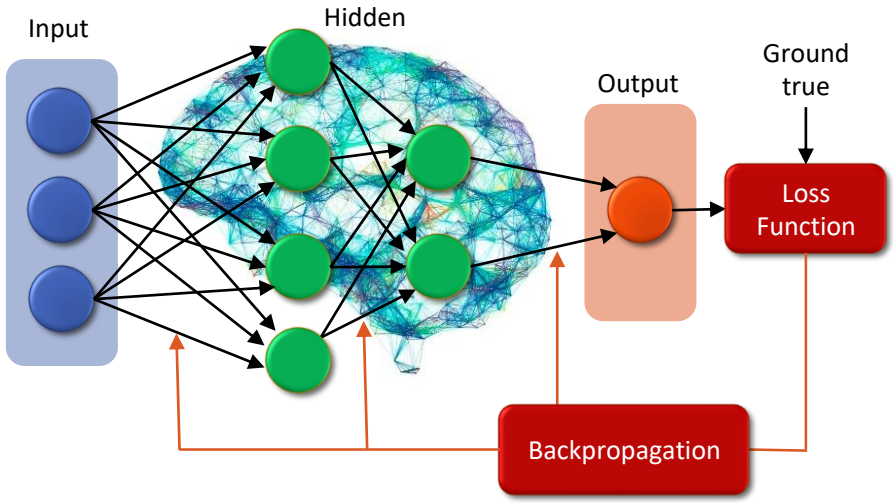
DEEP LEARNING

CONVOLUTIONAL NEURAL NETWORK





Tôn Quang Toại
Khoa Công nghệ thông tin
Trường đại học Ngoại ngữ - Tin học TP.HCM (HUFLIT)

Nội dung

- Một số vấn đề của Multi Layer Perceptron
- Cấu trúc của Convolutional Neural Network
  - Tầng Convolution (và hàm Activation)
  - Tầng Pooling
  - Tầng Fully connected và Tầng output
- Huấn luyện Convolutional Neural Network
- Đánh giá hiệu năng

MỘT SỐ VẤN ĐỀ CỦA MULTI LAYER PERCEPTRON

Một bài toán thị giác máy tính

- Image classification

Cat/Dog (0/1)

- Object detection

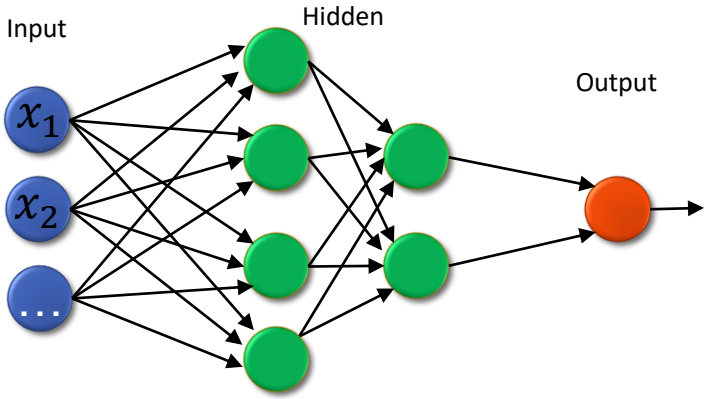
Cat/Dog (0/1)

x_1, y_1, x_2, y_2

- Image segmentation

Ba vấn đề của Multi Layer Perceptron

- Vấn đề 1: MLP có *kết nối* đầy đủ

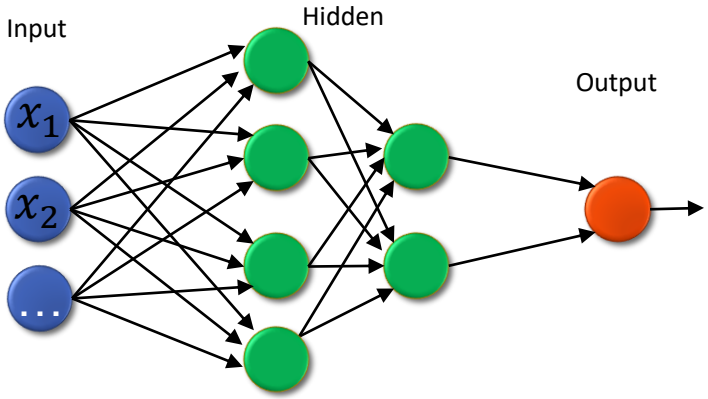


- Ví dụ data: $32 \times 32 \times 3$ $\rightarrow 1 \times 3072$

$x \in \mathbb{R}^{1\times 3072}$ $\quad W \in \mathbb{R}^{3072\times 10}$ $\quad \hat{y} \in \mathbb{R}^{1\times 2}$

Ba vấn đề của Multi Layer Perceptron

- Vấn đề 1: MLP có *kết nối* đầy đủ



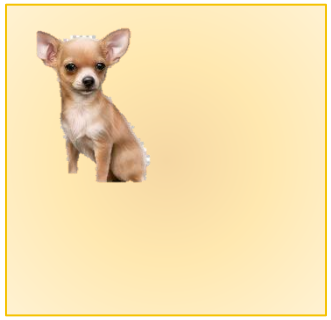
  - Ví dụ: Nếu tầng ẩn thứ nhất có 1000 neuron
    - Ảnh 255x255x3(RGB) = 195.075 thì sẽ có 195.075.000 tham số
    - Ảnh 1000x1000x3(RGB) số tham số là ????

- Nhận xét: Hiệu suất "tổng quát hóa" của MLP bị ảnh hưởng nếu số lượng các weights (tham số tự do – free parameters) quá lớn

Ba vấn đề của Multi Layer Perceptron

- Vấn đề 2: MLP bỏ qua *mối tương quan* cục bộ
  - Các pixel lân cận có quan hệ với nhau
  - Các pixel ở gần có mối tương quan mạnh hơn các pixels ở xa

- Vấn đề 3: MLP không đáp ứng tốt khi dữ liệu bị biến đổi nhiều
  - Vị trí của đối tượng,
  - Xoay, biến dạng của đối tượng



Chiến lược thiết kế mạng neuron

- Hai thước đo thiết kế
  - Generalization
  - Learning speed

- Chiến lược thiết kế

> "good generalization performance can be obtained if some prior knowledge about the task is built into the network"
>
> ***Yann LeCun***
>
> ***"Generalization and Network Design Strategies", 1989***

- *Nhận xét:* Cần trợ giúp, tạo điều kiện thuận lợi nhất cho mạng neuron học
  - Data: số lượng, chất lượng, dễ học
  - Neural Network Architectures: Tìm kiếm mẫu phù hợp bài toán, Đủ không gian lưu trữ mẫu
  - Loss function: Chọn mục tiêu học phù hợp
  - Optimizer: Phương pháp học nhanh

Chiến lược thiết kế mạng neuron

- Chú ý khi điều chỉnh kiến trúc mạng
  - Điều chỉnh kiến trúc mạng → ảnh hưởng độ phức tạp của mạng (network complexity, được phản ánh qua số lượng tham số của mạng)

- Mục tiêu
  - Giảm thiểu số lượng tham số để tăng tính tổng quát của mạng
  - Không làm suy giảm khả năng của mạng (network's capability)

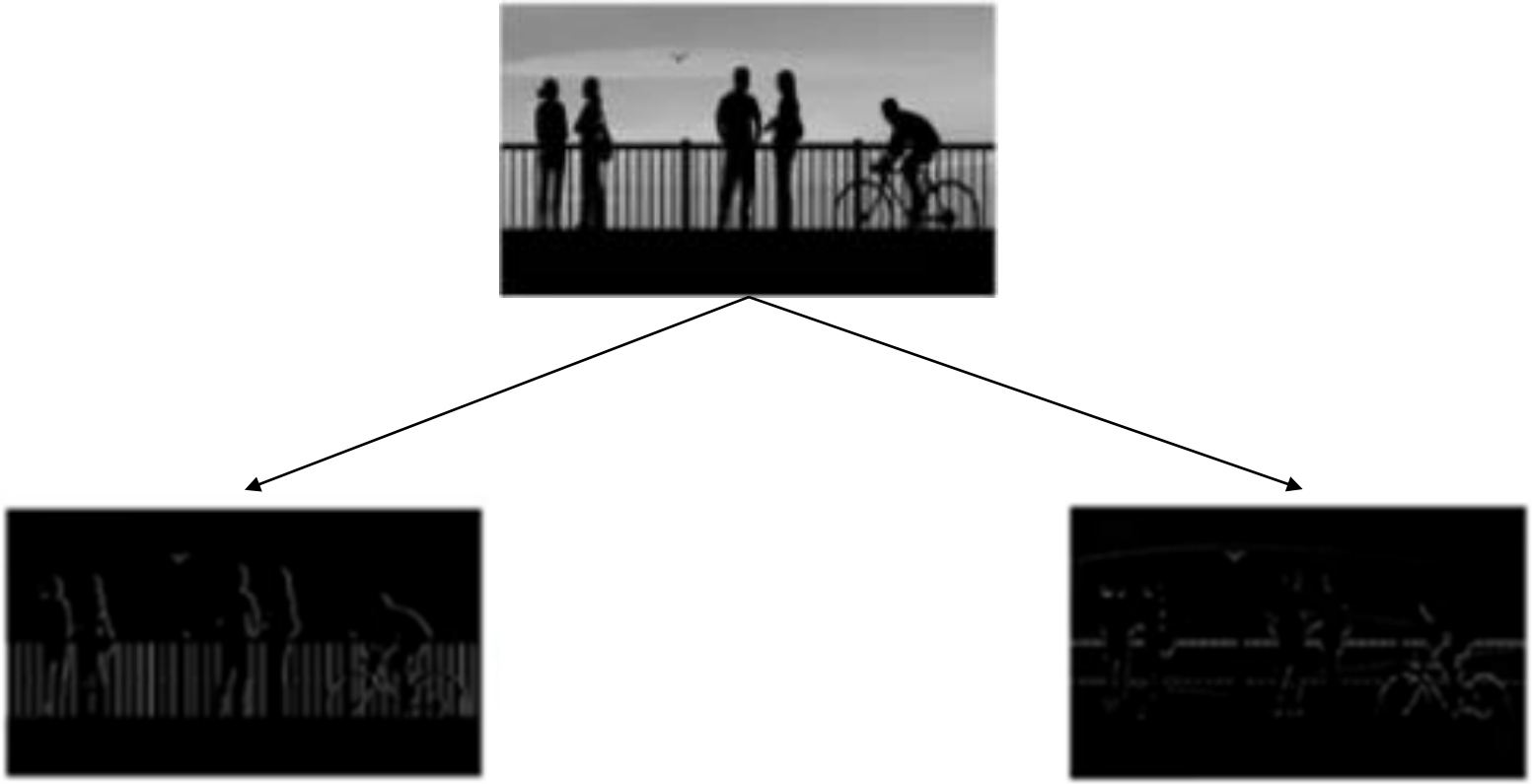
CẤU TRÚC CỦA
CONVOLUTIONAL NEURAL NETWORKS

Tầng Convolution

- Phép toán Convolution
- Vấn đề của phép toán convolution
- Padding
- Strided Convolution

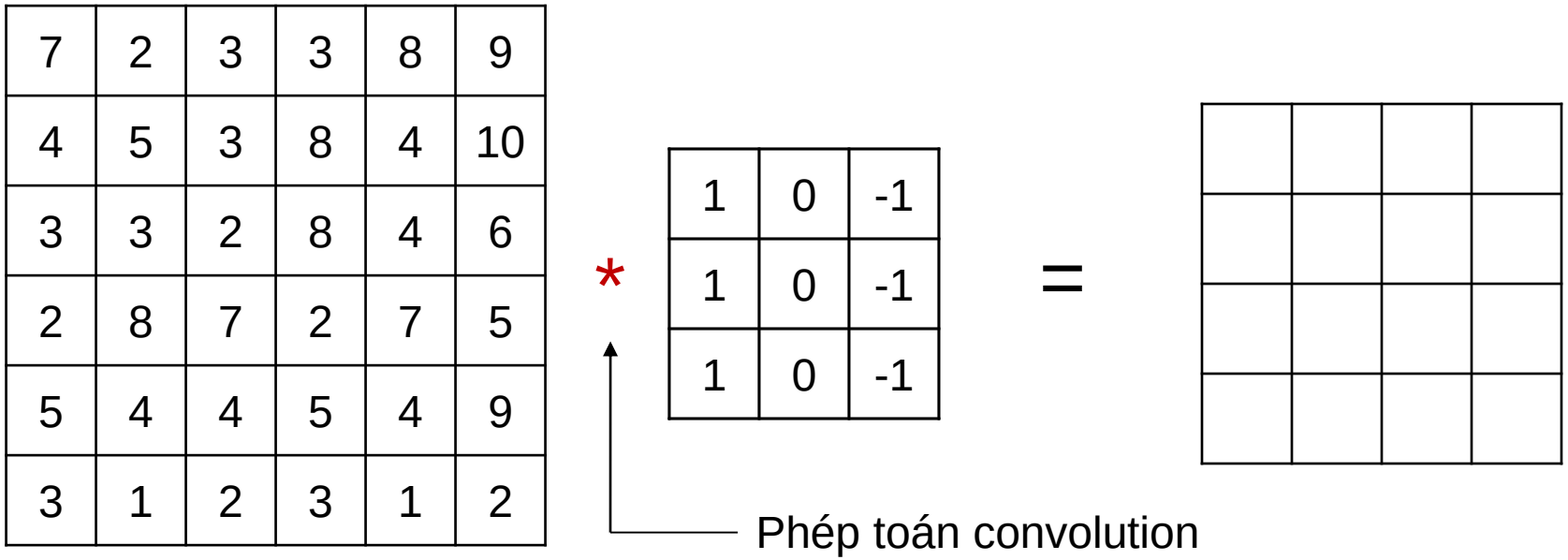
Xác định các cạnh trong ảnh

- Làm sao để xác định các cạnh (edges) trong ảnh



Phép toán Convolution

- Xác định cạnh đứng trong ảnh



Phép toán Convolution

- Xác định cạnh đứng trong ảnh

7^{1}	2^{0}	3^{-1}	3	8	9
4^{1}	5^{0}	3^{-1}	8	4	10
3^{1}	3^{0}	2^{-1}	8	4	6
2	8	7	2	7	5
5	4	4	5	4	9
3	1	2	3	1	2

*

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

=

Phép toán Convolution

- Xác định cạnh đứng trong ảnh

7	2^{1}	3^{0}	3^{-1}	8	9
4	5^{1}	3^{0}	8^{-1}	4	10
3	3^{1}	2^{0}	8^{-1}	4	6
2	8	7	2	7	5
5	4	4	5	4	9
3	1	2	3	1	2

*

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

=

Xác định các cạnh trong ảnh

- Xác định cạnh đứng trong ảnh

7	2	3^{1}	3^{0}	8^{-1}	9
4	5	3^{1}	8^{0}	4^{-1}	10
3	3	2^{1}	8^{0}	4^{-1}	6
2	8	7	2	7	5
5	4	4	5	4	9
3	1	2	3	1	2

*

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

=

Phép toán Convolution

- Xác định cạnh đứng trong ảnh

7	2	3	3^{1}	8^{0}	9^{-1}
4	5	3	8^{1}	4^{0}	10^{-1}
3	3	2	8^{1}	4^{0}	6^{-1}
2	8	7	2	7	5
5	4	4	5	4	9
3	1	2	3	1	2

*

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

=

Phép toán Convolution

- Xác định cạnh đứng trong ảnh

7	2	3	3	8	9
4^{1}	5^{0}	3^{-1}	8	4	10
3^{1}	3^{0}	2^{-1}	8	4	6
2^{1}	8^{0}	7^{-1}	2	7	5
5	4	4	5	4	9
3	1	2	3	1	2

*

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

=

Phép toán Convolution

- Xác định cạnh đứng trong ảnh

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

*

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

=

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

Phép toán Convolution

- Một số filters

Vertical

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

Horizontal

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

Sobel

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

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

Scharr

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

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

Phép toán Convolution

- Tham số hóa filter

7	2	3	3	8	9
4	5	3	8	4	10
3	3	2	8	4	6
2	8	7	2	7	5
5	4	4	5	4	9
3	1	2	3	1	2

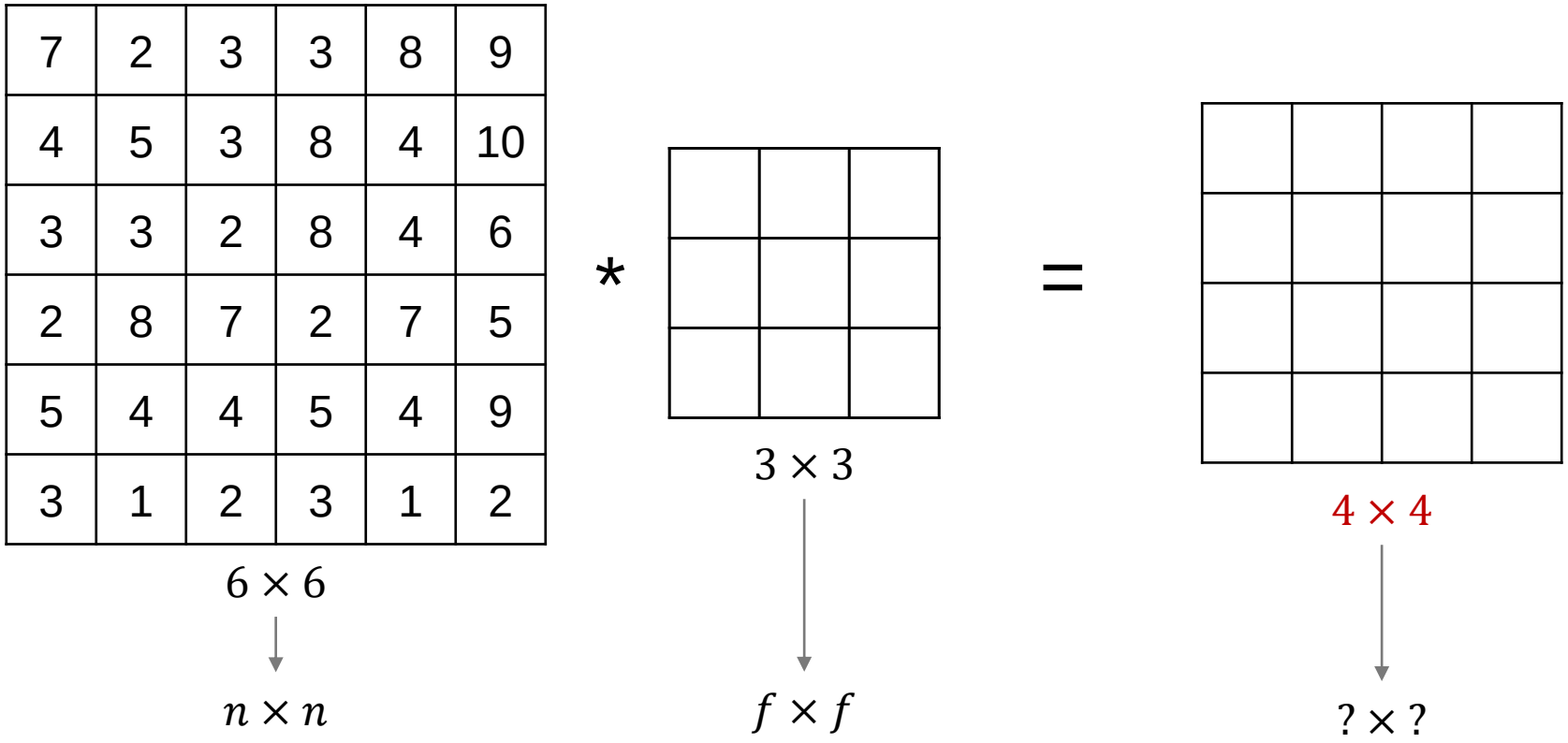
$*$

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_8

$=$

- *Nhận xét:* learn các bộ lọc để phát hiện các loại cạnh khác nhau (đứng, ngang, nghiêng k độ)

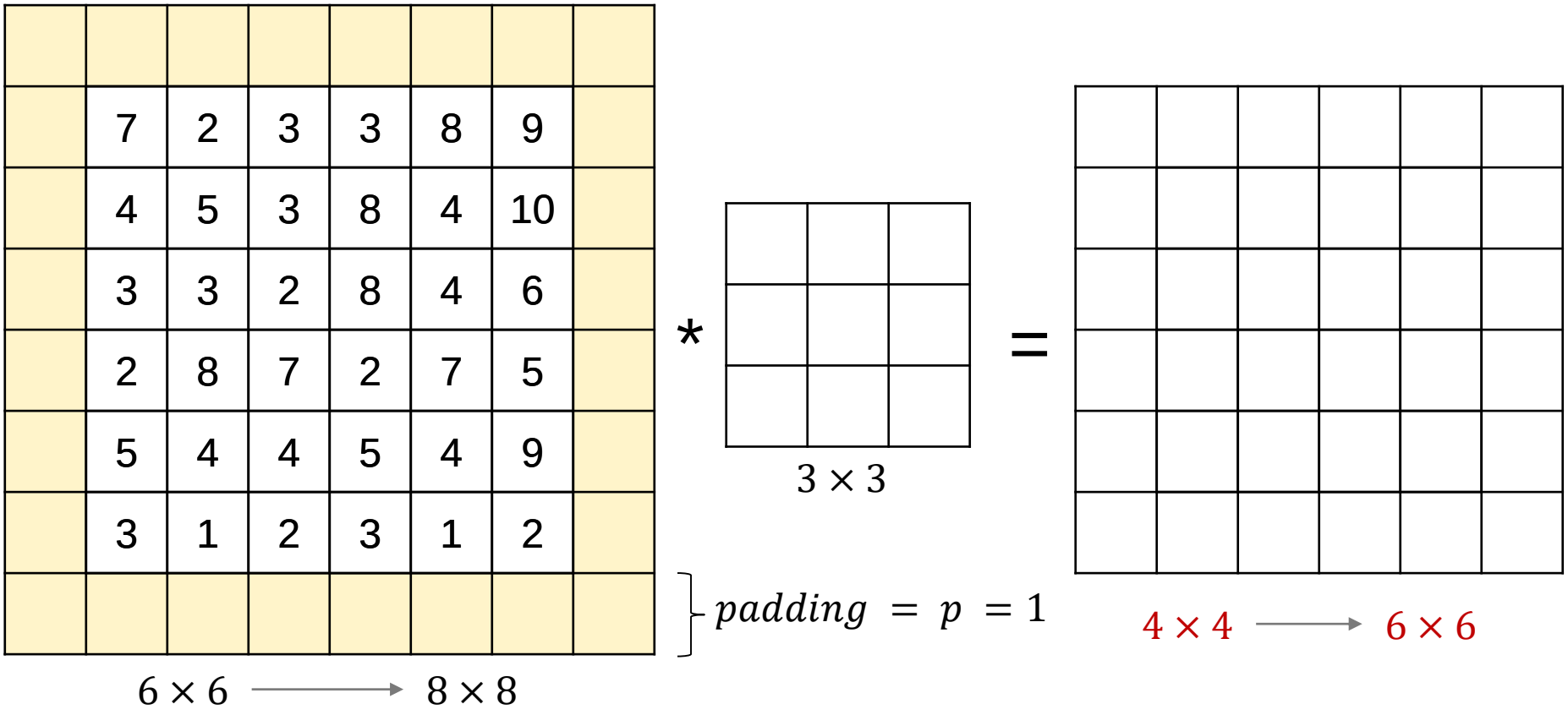
Vấn đề với phép toán Convolution



- *Nhận xét*
  - Ảnh kết quả nhỏ lại
  - Biên của ảnh bị mất

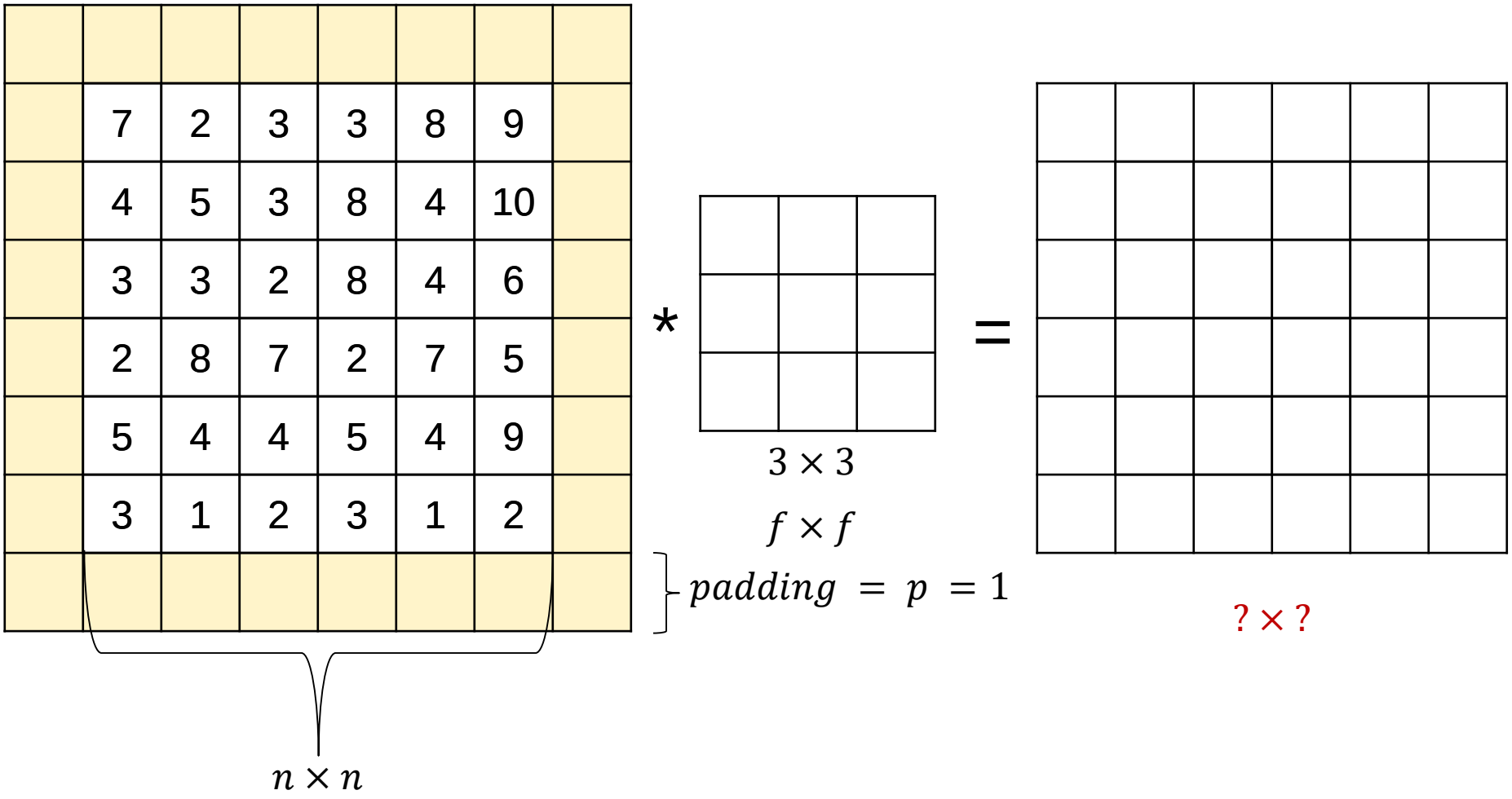
Padding: Cải tiến phép convolution

- Đệm vào ảnh một đường biên xung quanh



Padding: Cải tiến phép convolution

- Đệm vào ảnh một đường biên xung quanh



Padding: Cải tiến phép convolution

- Valid và Same Convolution
  - Valid: no padding
  - Same: Thêm padding để output size = input size

$$p = \frac{f-1}{2}$$

Stride

- Ví dụ: stride = 2

1	1	2	1	0	1	2
3	2	0	2	0	0	1
3	1	1	2	0	2	3
1	2	1	2	2	0	3
1	3	4	3	2	2	1
0	3	4	3	1	3	2
0	3	4	1	1	1	1

*

0	0	1
0	1	0
1	0	0

=

Strided convolution: Có chức năng là giảm kích thước của mẫu (downsampling)

Stride

- Ví dụ: stride = 2

1^{0}	1^{0}	2^{1}	1	0	1	2
3^{0}	2^{1}	0^{0}	2	0	0	1
3^{1}	1^{0}	1^{0}	2	0	2	3
1	2	1	2	2	0	3
1	3	4	3	2	2	1
0	3	4	3	1	3	2
0	3	4	1	1	1	1

*

0	0	1
0	1	0
1	0	0

=

Stride

- Ví dụ: stride = 2

1	1	2^0	1^0	0^1	1	2
3	2	0^0	2^1	0^0	0	1
3	1	1^1	2^0	0^0	2	3
1	2	1	2	2	0	3
1	3	4	3	2	2	1
0	3	4	3	1	3	2
0	3	4	1	1	1	1

*

0	0	1
0	1	0
1	0	0

=

Stride

- Ví dụ: stride = 2

1	1	2	1	0^{0}	1^{0}	2^{1}
3	2	0	2	0^{0}	0^{1}	1^{0}
3	1	1	2	0^{1}	2^{0}	3^{0}
1	2	1	2	2	0	3
1	3	4	3	2	2	1
0	3	4	3	1	3	2
0	3	4	1	1	1	1

*

0	0	1
0	1	0
1	0	0

=

Stride

- Ví dụ: stride = 2

1	1	2	1	0	1	2
3	2	0	2	0	0	1
3^{0}	1^{0}	1^{1}	2	0	2	3
1^{0}	2^{1}	1^{0}	2	2	0	3
1^{1}	3^{0}	4^{0}	3	2	2	1
0	3	4	3	1	3	2
0	3	4	1	1	1	1

*

0	0	1
0	1	0
1	0	0

=

Stride

- Ví dụ: stride = 2

1	1	2	1	0	1	2
3	2	0	2	0	0	1
3	1	1^{0}	2^{0}	0^{1}	2	3
1	2	1^{0}	2^{1}	2^{0}	0	3
1	3	4^{1}	3^{0}	2^{0}	2	1
0	3	4	3	1	3	2
0	3	4	1	1	1	1

*

0	0	1
0	1	0
1	0	0

=

Stride

- Ví dụ: stride = 2

1	1	2	1	0	1	2
3	2	0	2	0	0	1
3	1	1	2	0^0	2^0	3^1
1	2	1	2	2^0	0^1	3^0
1	3	4	3	2^1	2^0	1^0
0	3	4	3	1	3	2
0	3	4	1	1	1	1

*

0	0	1
0	1	0
1	0	0

=

Stride

- Ví dụ: stride = 2

1	1	2	1	0	1	2
3	2	0	2	0	0	1
3	1	1	2	0	2	3
1	2	1	2	2	0	3
1^{0}	3^{0}	4^{1}	3	2	2	1
0^{0}	3^{1}	4^{0}	3	1	3	2
0^{1}	3^{0}	4^{0}	1	1	1	1

*

0	0	1
0	1	0
1	0	0

=

Stride

- Ví dụ: stride = 2

1	1	2	1	0	1	2
3	2	0	2	0	0	1
3	1	1	2	0	2	3
1	2	1	2	2	0	3
1	3	4^0	3^0	2^1	2	1
0	3	4^0	3^1	1^0	3	2
0	3	4^1	1^0	1^0	1	1

*

0	0	1
0	1	0
1	0	0

=

Stride

- Ví dụ: stride = 2

1	1	2	1	0	1	2
3	2	0	2	0	0	1
3	1	1	2	0	2	3
1	2	1	2	2	0	3
1	3	4	3	2^0	2^0	1^1
0	3	4	3	1^0	3^1	2^0
0	3	4	1	1^1	1^0	1^0

$Input$: $n \times n$

$*$

0	0	1
0	1	0
1	0	0

$filter$: $f \times f$

$=$

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

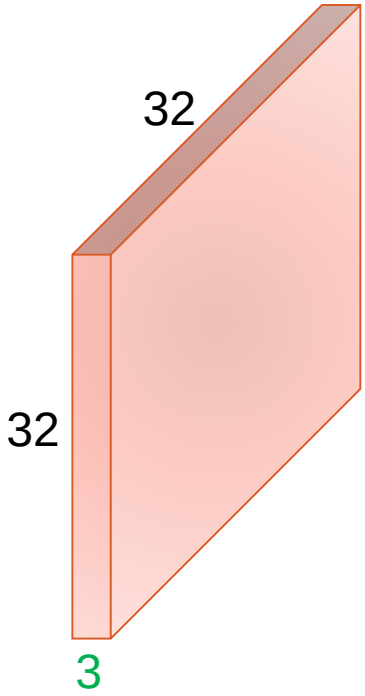
$padding: p$

$stride: s$

CONVOLUTION TRÊN ẢNH RGB

Data

- Data: $32 \times 32 \times 3$ image
  - Mỗi ảnh gồm 3 channels R, G, B độc lập cấu tạo nên hình

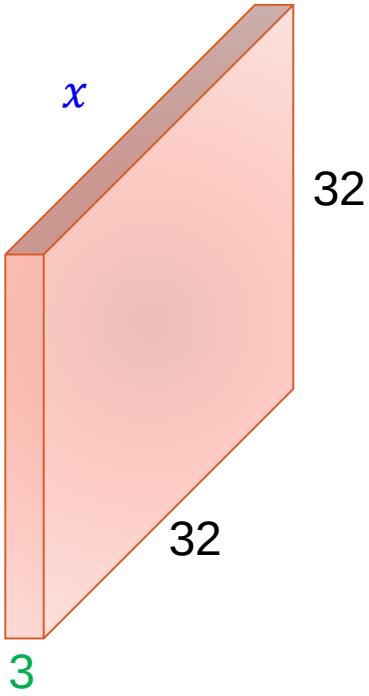


$32 \times 32 \times 3$

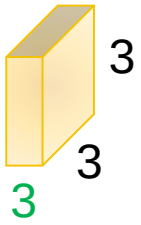
$height \times width \times channel$

Bộ lọc 3D

- Filter 3D



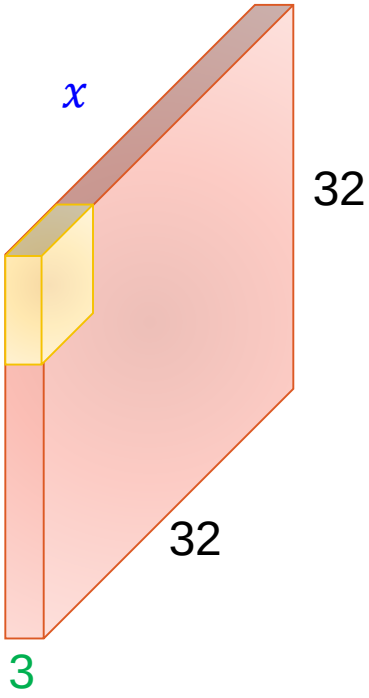
$32 \times 32 \times 3$



$3 \times 3 \times 3$

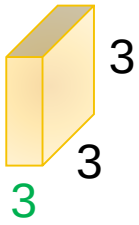
Bộ lọc 3D

- Filter 3D



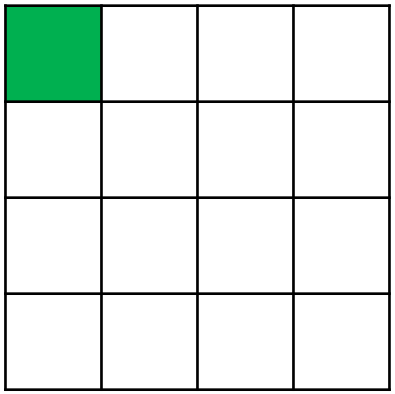
32 × 32 × 3

*



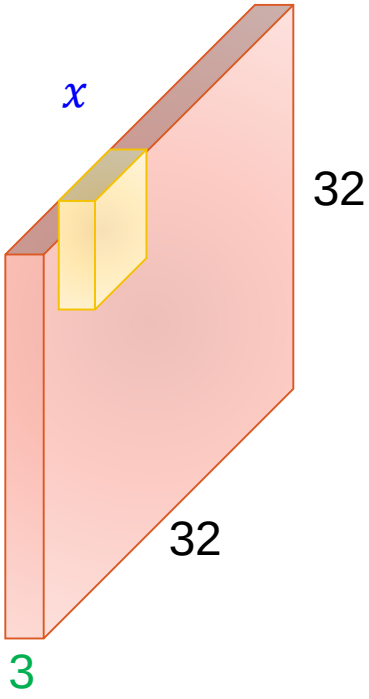
3 × 3 × 3

=



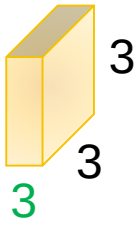
Bộ lọc 3D

- Filter 3D



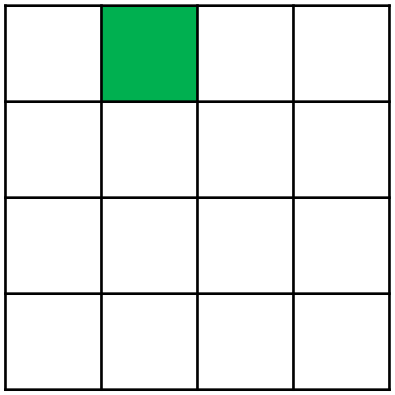
32 × 32 × 3

*



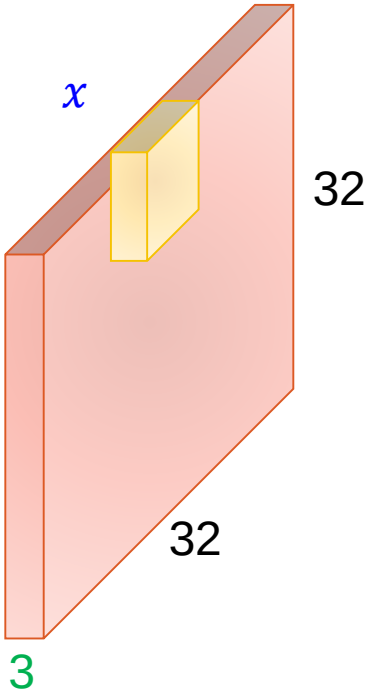
3 × 3 × 3

=



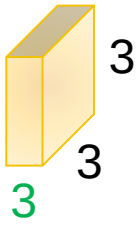
Bộ lọc 3D

- Filter 3D



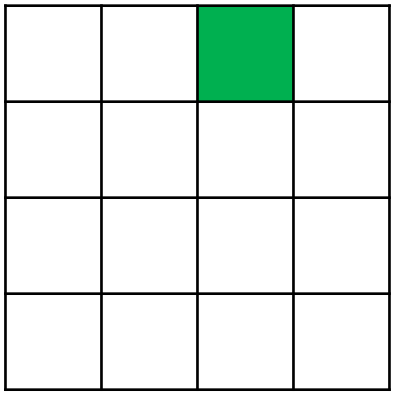
32 × 32 × 3

*



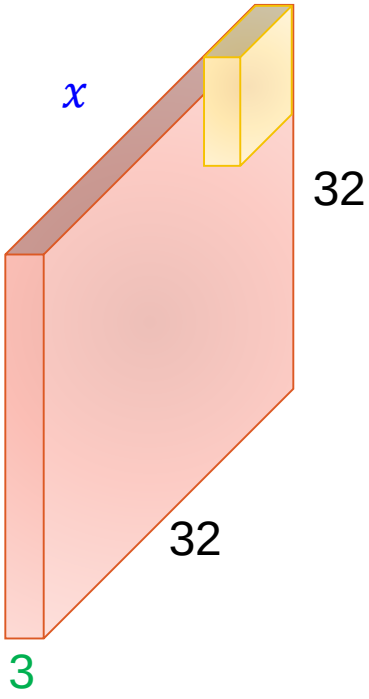
3 × 3 × 3

=



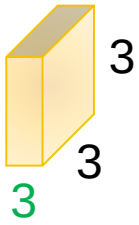
Bộ lọc 3D

- Filter 3D



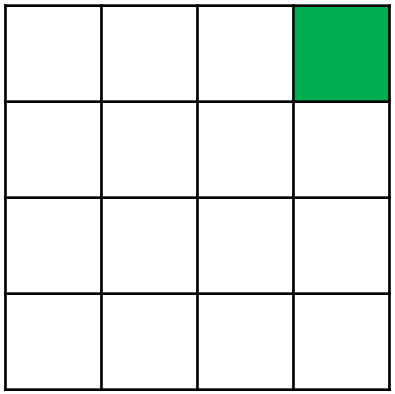
32 × 32 × 3

*

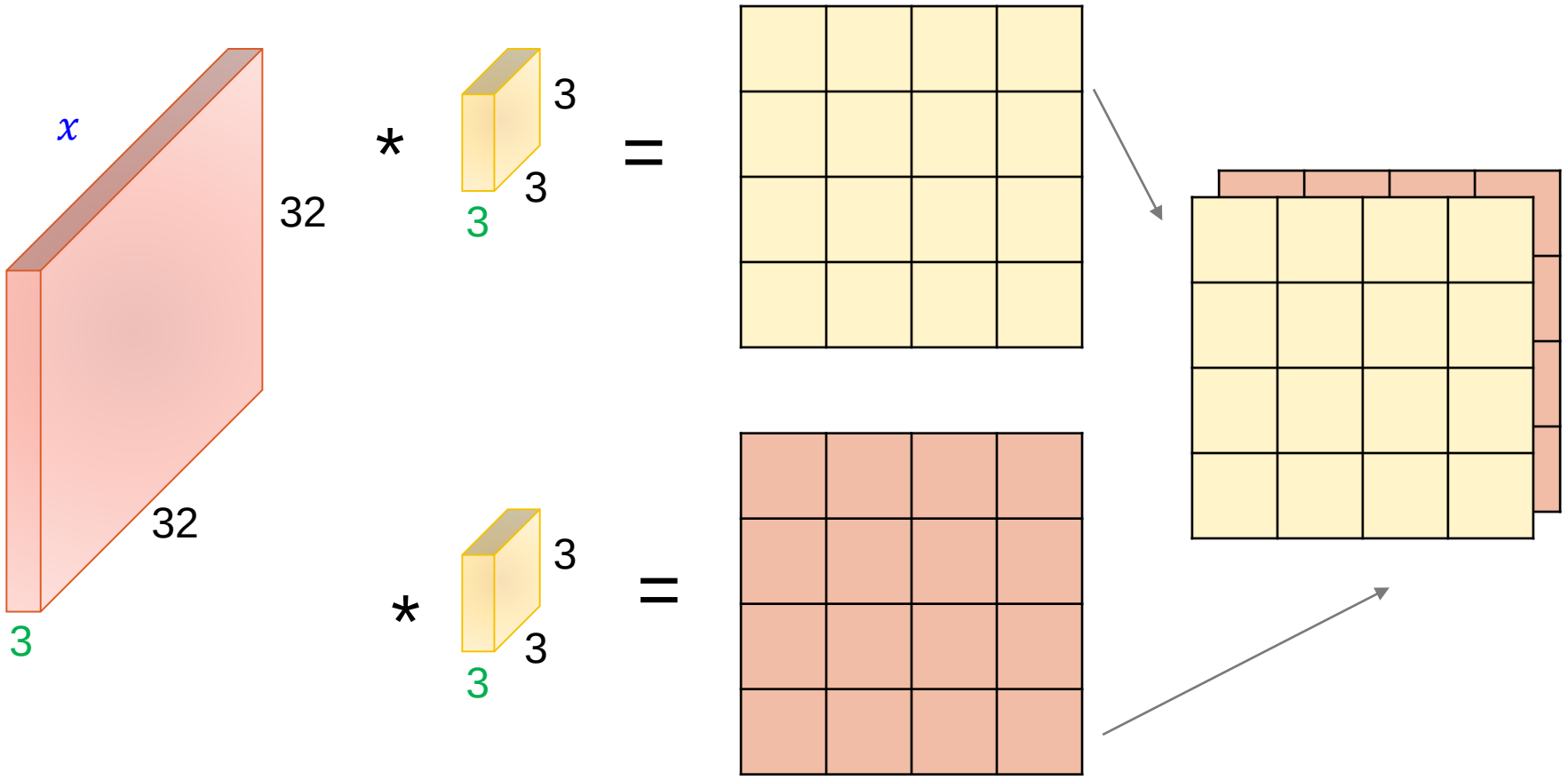


3 × 3 × 3

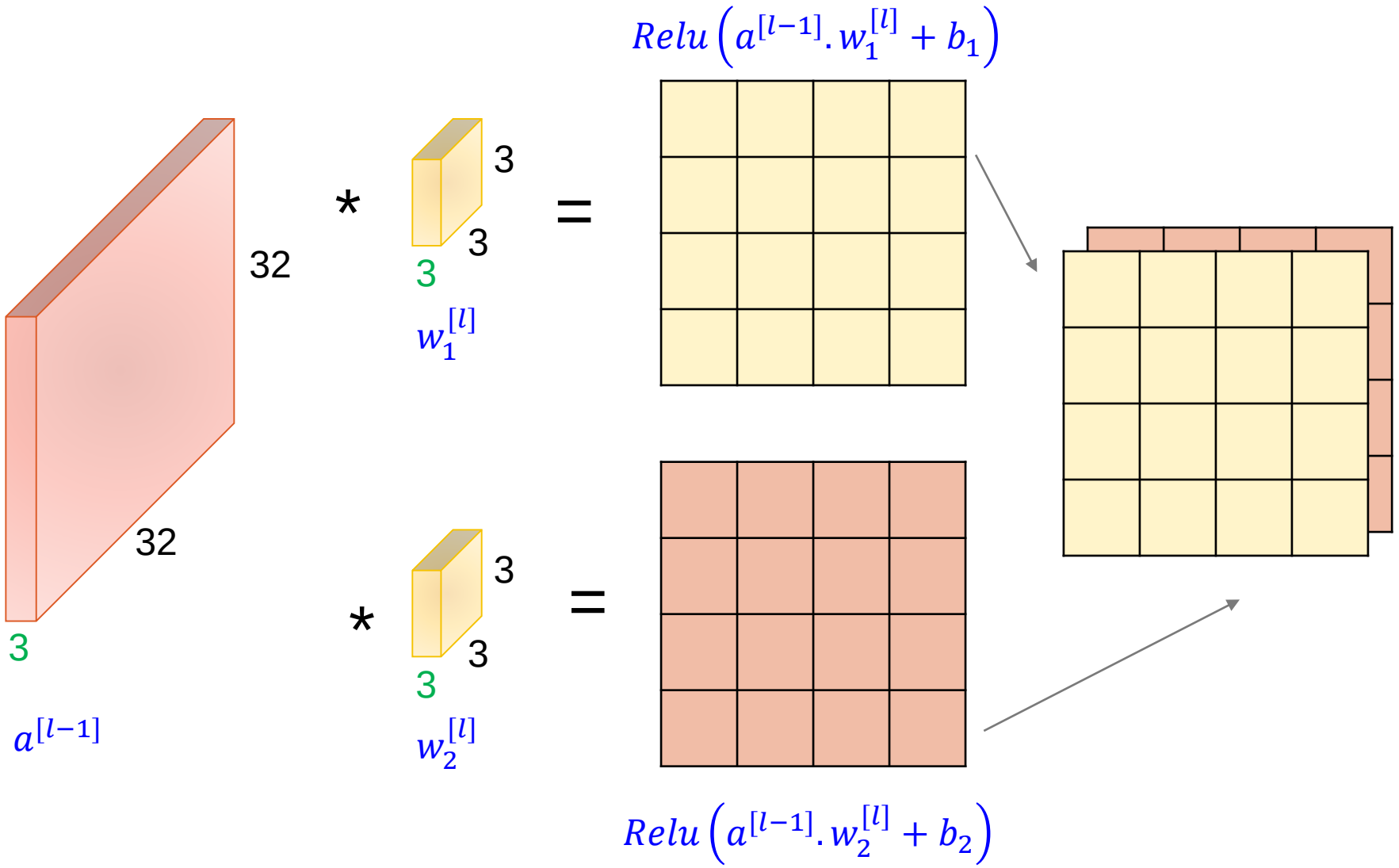
=



Nhiệu bộ lọc 3D



Toán học cho Convolution



Toán học cho Convolution

- Số tham số của một tầng

- Ví dụ: Trong 1 tầng có
  - Số filter: 10 filter
  - Kích thước của filter: $3 \times 3 \times 4$

  - Tính số tham số
    - Số tham số 1 filter:$3 \times 3 \times 4 + 1(bias) = 36 + 1 = 37$
    - Số tham số 10 filter:$37 \times 10 = 370$

Toán học cho Convolution

- Xét tầng l
  - $f^{[l]}$: kích thước filter
  - $p^{[l]}$: kích thước padding
  - $s^{[l]}$: kích thước stride
  - $n_c^{[l]}$: Số lượng filter

$$n_H^{[l]} = \frac{n_H^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1$$

$$n_W^{[l]} = \frac{n_W^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1$$

- Input: $n_H^{[l-1]} \times n_W^{[l-1]} \times n_c^{[l-1]}$
- Output: $n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$
- Weights: $f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$
- Bias: $n_c^{[l]}$

Tầng Pooling

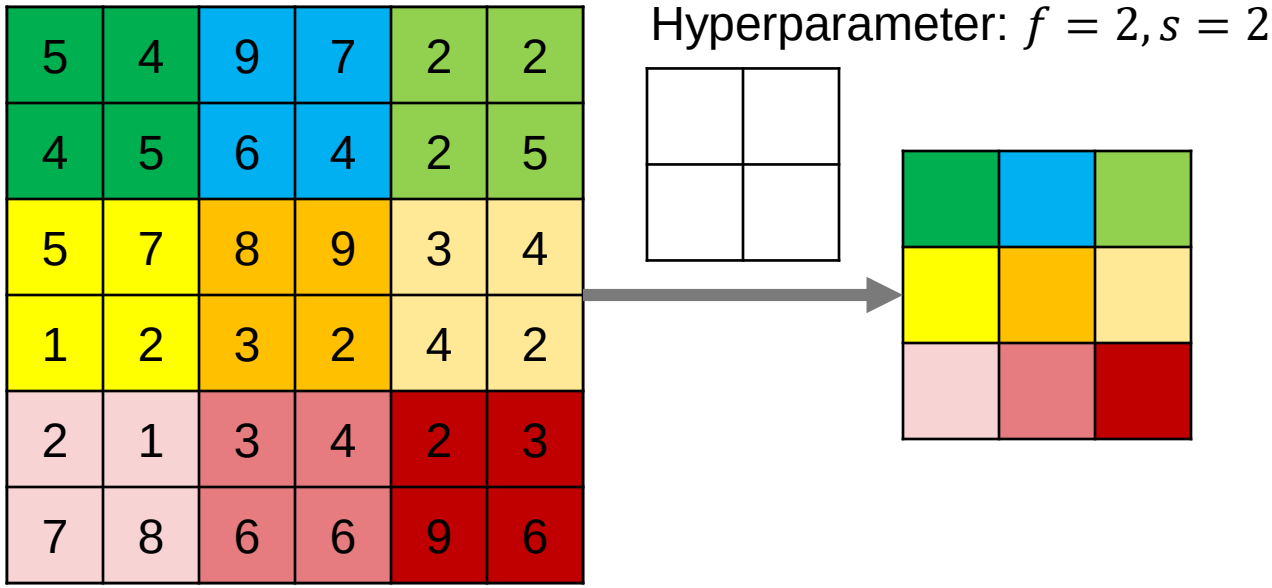
- Phép toán pooling: Trượt filter hai chiều qua mỗi kênh của feature map và tóm tắt các feature nằm trong khu vực được bộ lọc bao phủ.

- Có hai loại pooling thông dụng
  - Max pooling
  - Average pooling

- Nhận xét: Tầng pooling không có tham số để học

Tầng Pooling

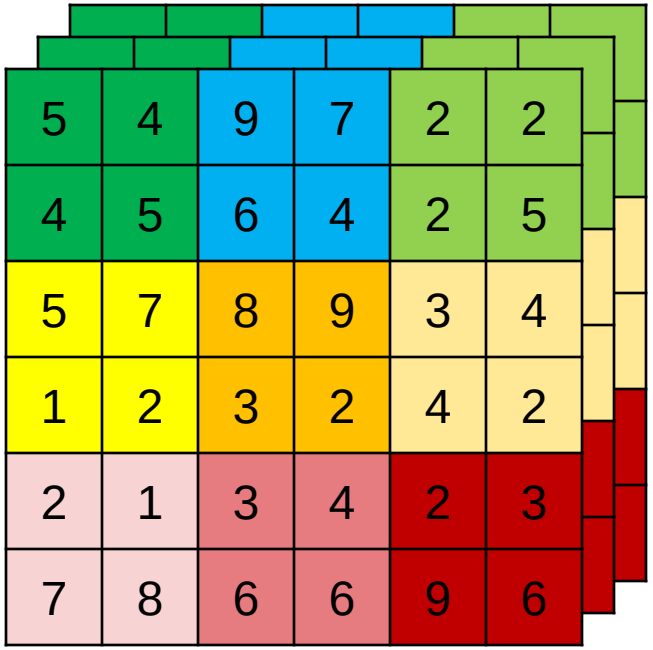
- Max pooling: chọn giá trị lớn nhất từ vùng của feature map được filter bao phủ.

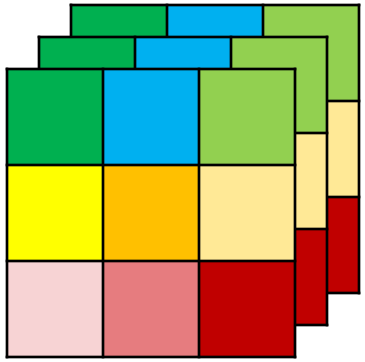


- Nhận xét: Kết quả là một feature map chứa các feature nổi bật nhất của feature map trước đó.

Tầng Pooling

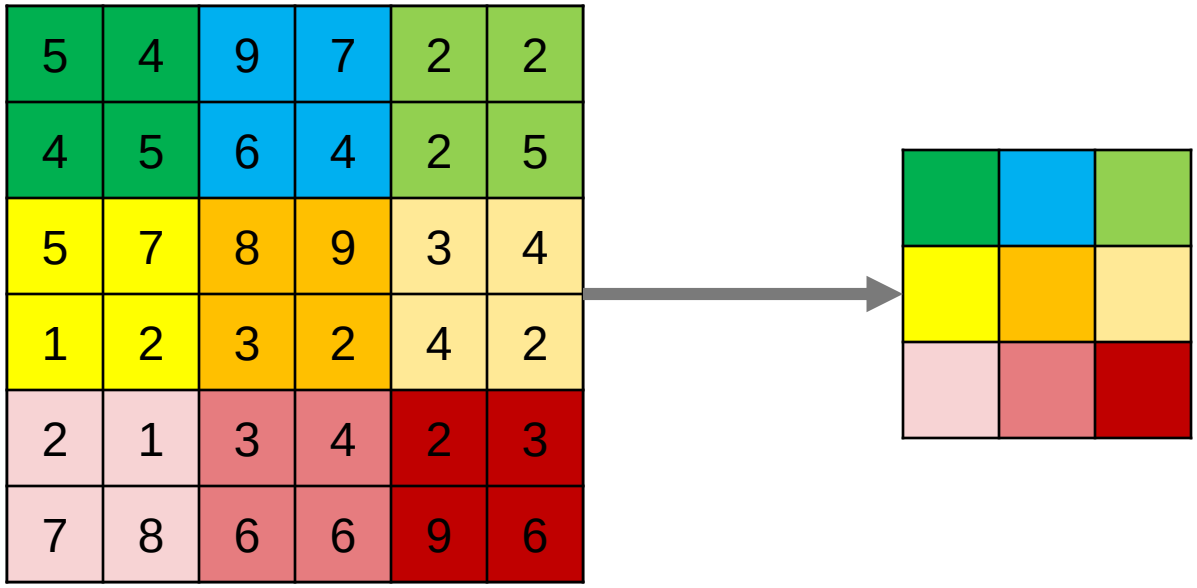
- Max pooling





Tầng Pooling

- Average pooling: tính giá trị trung bình từ vùng của feature map được filter bao phủ.



- Nhận xét: Kết quả là một feature map chứa các feature là mức trung bình của các feature trong feature map trước đó.

Tầng Pooling

- Tại sao phải dung pooling?

Hành vi	Chức năng
Giảm kích thước feature map (representation) • Giảm số lượng tham số học • Giảm số lượng phép toán	• Học nhanh hơn • Tăng tốc độ tính toán
Tính giá trị max/averange	• Vẫn giữ thông tin quan trọng • Bất biến với phép xoay, tịnh tiến (trong chừng mực nào đó)

Tầng Pooling

- Tóm tắt
  - Hyperparameter
    - f: filter size
    - s: stride
  - Max pooling
  - Averange pooling
  - Kích thước đầu ra của tầng pooling
    - Input: n_H, n_W, n_C
    - Output: $\frac{n_H+2p-f}{s}+1, \frac{n_W+2p-f}{s}+1, n_C$

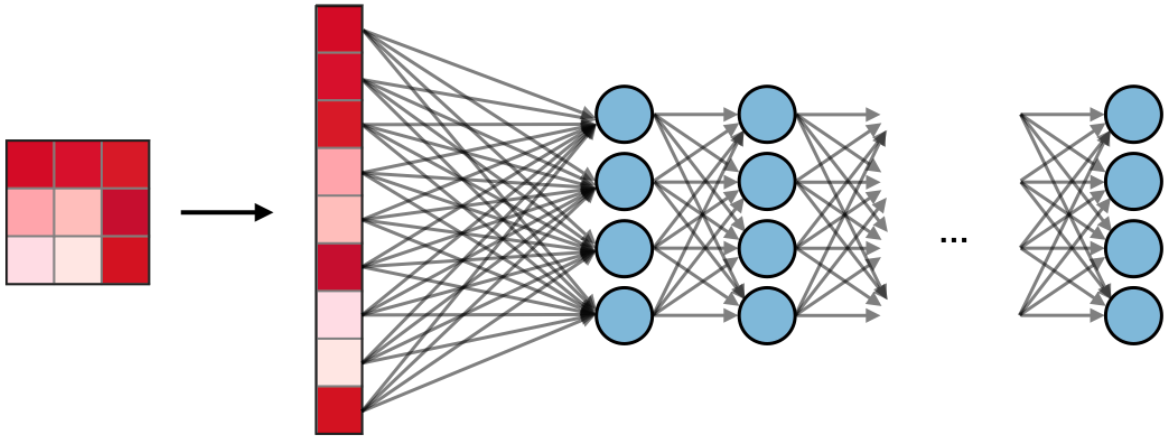
Tầng Fully connected

- Các tầng ở đầu của mạng dùng để trích chọn đặc trưng của data (Feature extractor)

- Sau khi có các features, chúng ta tiến hành tính toán: phân lớp dữ liệu (Classifier), Hồi quy (Regression), …
  - Fully connected layer (MLP, FC)
  - SVM
  - …

Tầng Fully connected

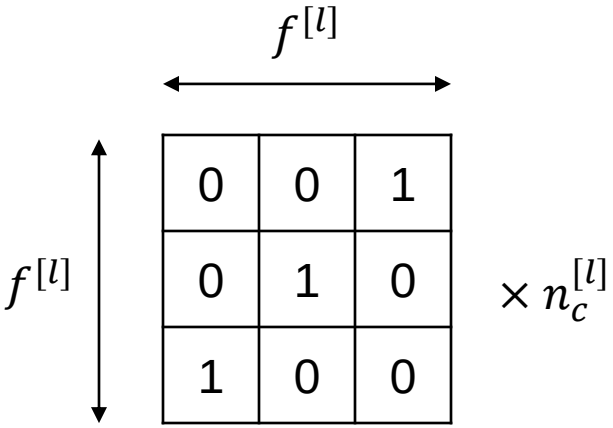
- Các bước kết nối với FC
  - Bước 1. Chuyển feature map tầng cuối thành 1 vector
  - Bước 2. Mỗi feature trong tầng cuối cùng được kết nối với mỗi neuron trong tầng đầu tiên của tầng fully connected layer

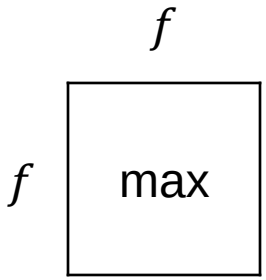


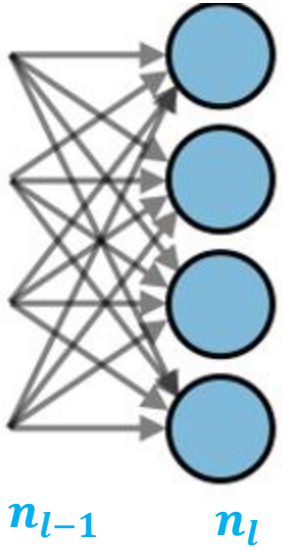
- Nhận xét: Có thể dùng nhiều hơn một tầng fully connected layer để tăng khả năng của mạng

Các loại tầng trong CNN

- Một mạng CNN có 3 loại tầng
  - Convolution (CONV)
  - Pooling (POOL)
  - Fully connected (FC)



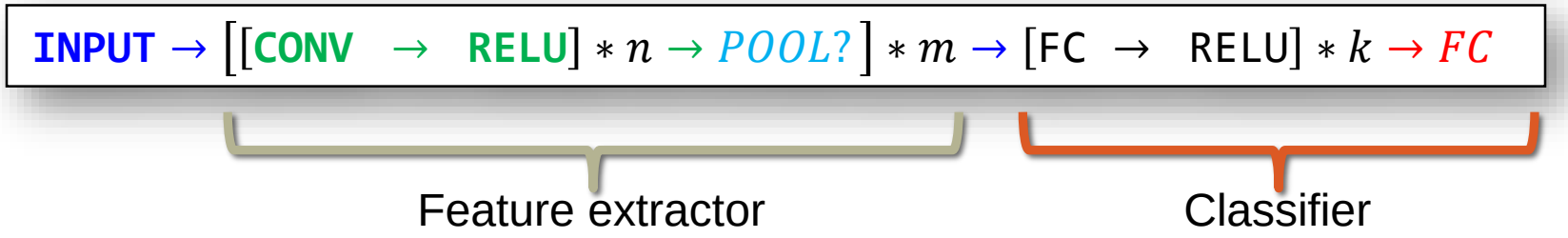




KIẾN TRÚC CHUNG CỦA CNN

Kiến trúc chung

- Kiến trúc CNN thông thường



  - Input: DATA + Preprocessing
  - Feature extractor: CONV + RELU + POOL
  - Classifier: FC + RELU

HÀM LOSS VÀ HÀM COST

Hàm Loss

- Hàm loss: Hàm loss là hàm dùng để đo lương sự khác biệt giữa giá trị dự đoán ($\hat{y}$) và giá trị thật sự (y)

- Nhận xét
  - Optimizer sử dụng hàm loss để tính lỗi của mô hình
  - Optimizer học các Weight thông qua đạo hàm của hàm loss

- Chọn hàm Loss
  - Phản ảnh mục tiêu của bài toán
  - Hàm có đạo hàm

Hàm Loss

- Một số hàm Loss thông dụng
  - Mean Absolute Error
  - Mean Squared Error
  - Binary Cross Entropy
  - Categorical Cross Entropy

Mean Absolute Error (MAE/L1 Loss)

- **Chức năng:** Dùng cho bài toán regression

- **Loss function** cho từng data point

$$L^{(i)} = \left| y^{(i)} - \hat{y}^{(i)} \right|$$

- **Cost function** cho m data points

$$Cost = \frac{1}{m} \sum_{i=1}^{m} L^{(i)} = \frac{1}{m} \sum_{i=1}^{m} \left| y^{(i)} - \hat{y}^{(i)} \right|$$

Mean Absolute Error (MAE/L1 Loss)

Sample ID	y	$\hat{y}$	$L = \lvert y - \hat{y} \rvert$
1	100	100	0
2	200	210	10
3	200	220	20
4	300	350	50
5	400	390	10
		$Cost =$	90

Mean Squared Error (MSE/L2 Loss)

- **Chức năng:** Dùng cho bài toán regression

- **Loss function** cho từng data point

$$L^{(i)} = \left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

- **Cost function** cho m data points

$$Cost = \frac{1}{m}\sum_{i=1}^{m} L^{(i)} = \frac{1}{m}\sum_{i=1}^{m}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$$

MSE vs MEA

Sample ID	y	$\hat{y}$	$L = \lvert y - \hat{y} \rvert$	$L = (y - \hat{y})^2$
1	100	100	0	0
2	200	210	10	100
	200	220	20	400
3	300	350	50	2500
4	400	390	10	100
		$Cost =$	90	3100

Binary Cross Entropy cost function (BCE)

- **Chức năng:** Dùng cho phân lớp nhị phân (binary classification)

- **Loss function** cho từng data point

$$L^{(i)} = -\left[y^{(i)} log\left(\hat{y}^{(i)}\right) + \left(1 - y^{(i)}\right) log\left(1 - \hat{y}^{(i)}\right)\right]$$

- **Cost function** cho m data points

$$Cost = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)} log\left(\hat{y}^{(i)}\right) + \left(1 - y^{(i)}\right) log\left(1 - \hat{y}^{(i)}\right)\right]$$

Binary Cross Entropy cost function (BCE)

- **Cost function** cho m data points

$$Cost = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}log\left(\hat{y}^{(i)}\right) + \left(1-y^{(i)}\right)\log\left(1-\hat{y}^{(i)}\right)\right]$$

Đánh giá lỗi của loại 1

$y^{(i)}$	$\hat{y}^{(i)}$	$y^{(i)}log\left(\hat{y}^{(i)}\right)$
0	∀	0
1	0.99	~0
1	0.01	$-\infty$

Đánh giá lỗi của loại 0

$y^{(i)}$	$\hat{y}^{(i)}$	$\left(1-y^{(i)}\right)\log\left(1-\hat{y}^{(i)}\right)$
1	∀	0
0	0.01	~0
0	0.99	$-\infty$

Categorical Cross Entroy cost function (CCE)

- **Chức năng:** Dùng cho phân lớp có nhiều lớp (Multi classification)

- **Loss function** cho từng data point

$$s^{(i)} = f(x^{(i)}; W)$$

$$\hat{y}^{(i)} = softmax\left(s^{(i)}\right) = \left(\frac{e^{s_k^{(i)}}}{\sum_j e^{s_j^{(i)}}}\right), \forall k$$

$$L^{(i)} = -y^{(i)}.log\left(\hat{y}^{(i)}\right)$$

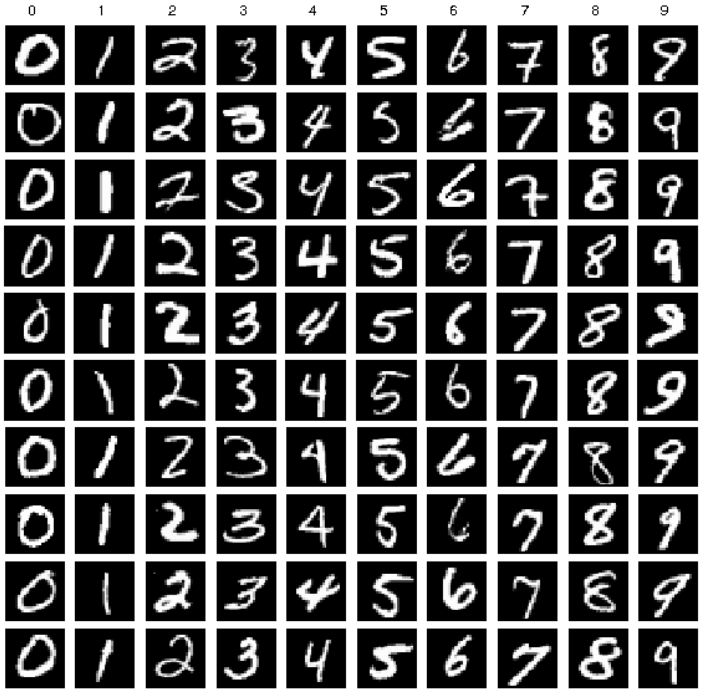
- **Cost function** cho m data points

$$Cost = \frac{1}{m}\sum_{i=1}^{m} L^{(i)} = -\frac{1}{m}\sum_{i=1}^{m} y^{(i)}.log\left(\hat{y}^{(i)}\right)$$

MỘT VÍ DỤ CNN

Bài toán phân lớp

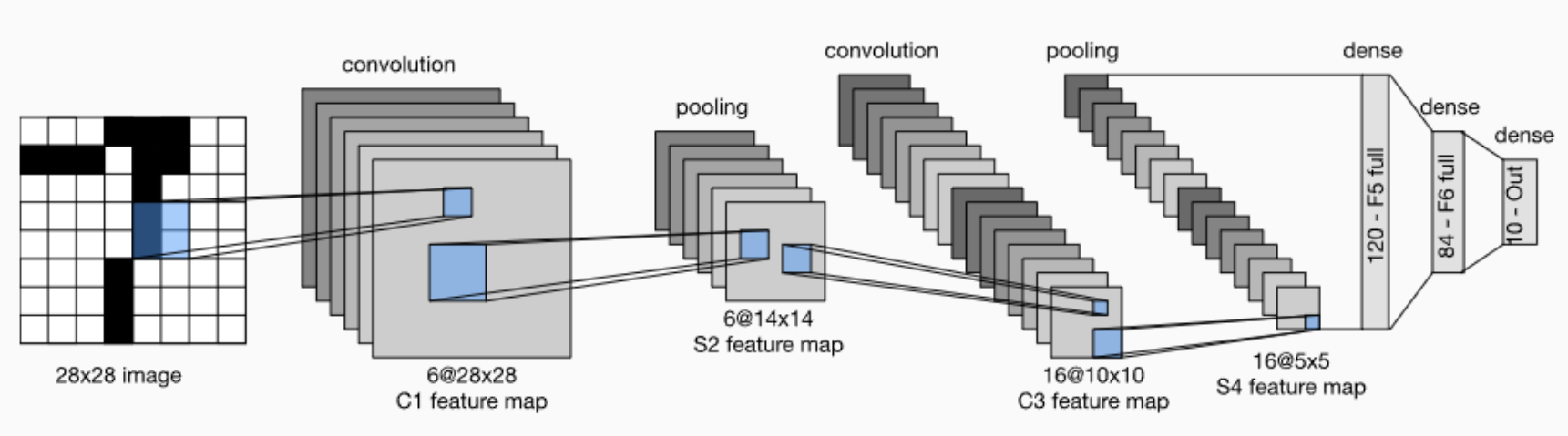
- Dataset: MNIST
  - Số lượng: 70.000 images
  - Kích thước ảnh: 28×28 ảnh mức xám



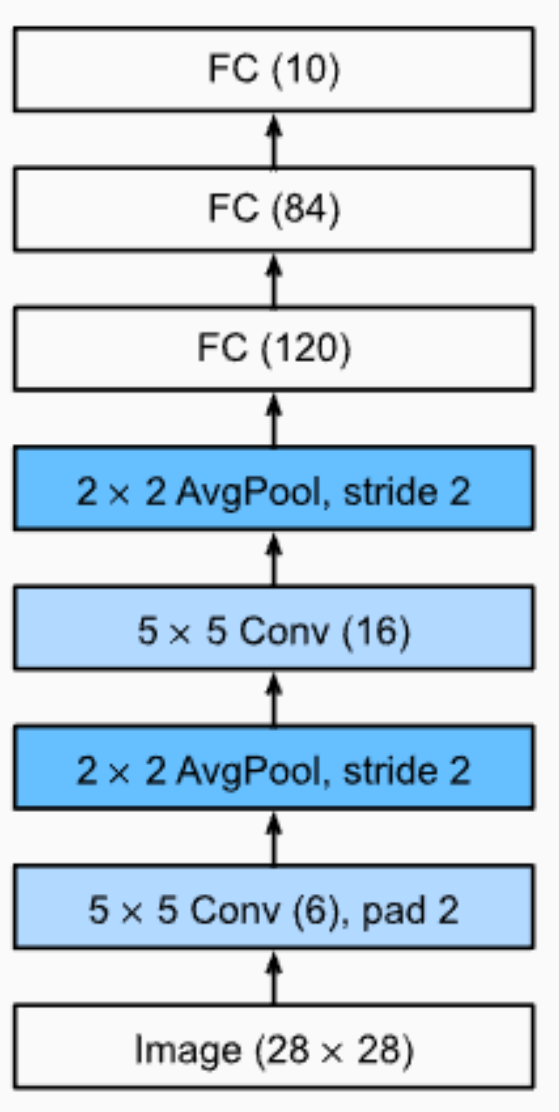
- Mong muốn: Xây dựng hệ thống có thể học từ các dữ liệu thu thập được.

Ví dụ CNN

- LeNet



Ví dụ CNN


FC (10)
FC (84)
FC (120)
2 × 2 AvgPool, stride 2
5 × 5 Conv (16)
2 × 2 AvgPool, stride 2
5 × 5 Conv (6), pad 2
Image (28 × 28)

Ví dụ CNN

- Import thư viện

```
import tensorflow as tf
from tensorflow import keras
import numpy as np
```

- Load data, tiền xử lý dữ liệu và chia dữ liệu

```
(train_x, train_y), (test_x, test_y) = keras.datasets.mnist.load_data()

train_x = train_x / 255.0
test_x = test_x / 255.0

train_x = tf.expand_dims(train_x, 3)
test_x = tf.expand_dims(test_x, 3)

val_x = train_x[:5000]
val_y = train_y[:5000]
```

Ví dụ CNN

- Xây dựng mô hình

```
lenet_5_model = keras.models.Sequential([
    keras.layers.Conv2D(6, kernel_size=5, strides=1,  activation='tanh',
                        input_shape=train_x[0].shape, padding='same'), #C1
    keras.layers.AveragePooling2D(), #S2
    keras.layers.Conv2D(16, kernel_size=5, strides=1, activation='tanh', padding='valid'), #C3
    keras.layers.AveragePooling2D(), #S4
    keras.layers.Conv2D(120, kernel_size=5, strides=1, activation='tanh', padding='valid'), #C5
    keras.layers.Flatten(), #Flatten
    keras.layers.Dense(84, activation='tanh'), #F6
    keras.layers.Dense(10, activation='softmax') #Output layer
])
```

Ví dụ CNN

- Huấn luyện mô hình

```
lenet_5_model.compile(optimizer='adam',
                      loss=keras.losses.categorical_crossentropy,
                      metrics=['accuracy'])

lenet_5_model.fit(
        train_x, train_y, epochs=5,
        validation_data=(val_x, val_y))
```

- Đánh giá mô hình

```
lenet_5_model.evaluate(test_x, test_y)
```

Tóm tắt

- Tầng Convolution (CONV)
- Tầng Pooling (Max pooling, Averange pooling) (POOL)
- Tầng Fully Connected (FC)
- Hàm Loss và Cost