Vài Mạng Học sâu và Ứng dụng (CNN & RNN)

Vương Thùy Dương

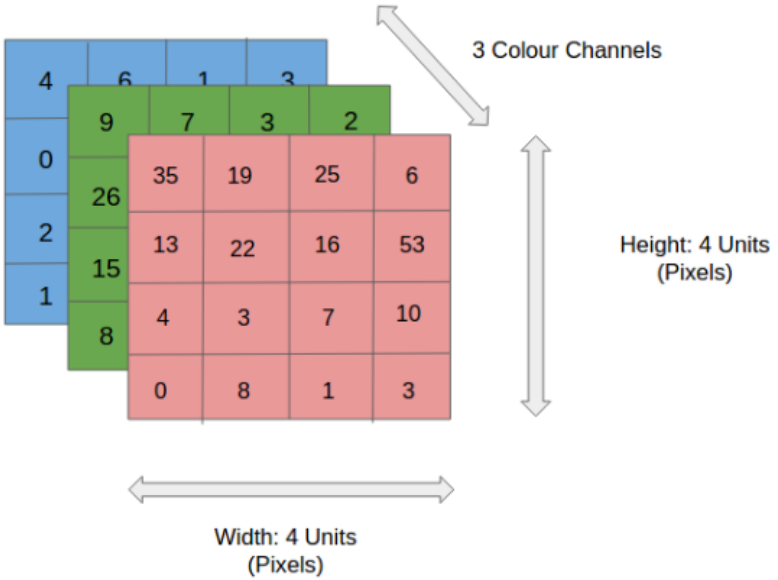
MIT

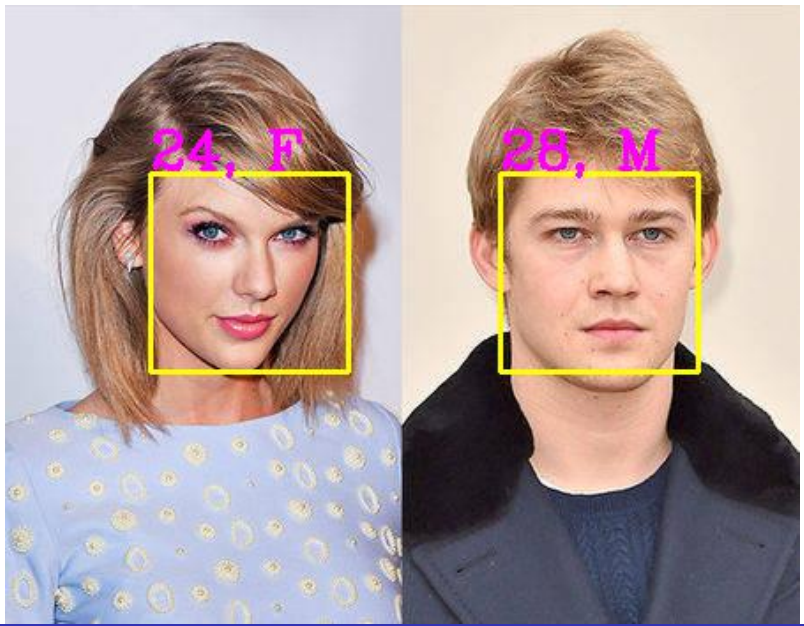
Sơ lược

Nhận diện hình ảnh

Bài toán

Dầu vào: hình ảnh, biểu diễn dưới dạng ma trận (2 chiều), mỗi pixel ảnh biểu thị tương ứng với 1 phần tử của ma trận.
Đầu ra: Miêu tả (ngắn gọn) của hình ảnh
Ví dụ: khuôn mặt người → giới tính, tuổi





Giải pháp bằng mạng NN

Ý tưởng

Flatten ma trận biểu diễn bức ảnh thành vector 1 chiều, rồi xử lý bằng mạng NN

Vấn đề

Spatial relationship

Flatten phá hủy cấu trúc bức ảnh
Mỗi pixel trong ảnh phụ thuộc vào các pixel "hàng xóm". Flatten làm mất đi quan hệ "hàng xóm" (spatial relationship) này

Độ phức tạp của model

- Ảnh 256x256 $\rightarrow 256 \times 256 = 65536 = 2^{16}$ pixels

Vấn đề

Spatial relationship

Flatten phá hủy cấu trúc bức ảnh
Mỗi pixel trong ảnh phụ thuộc vào các pixel "hàng xóm". Flatten làm mất đi quan hệ "hàng xóm" (spatial relationship) này

Độ phức tạp của model

- Ảnh 256x256 $\rightarrow 256 \times 256 = 65536 = 2^{16}$ pixels
- Linear model cần 2^{16} trọng số (weight)

Vấn đề

Spatial relationship

Flatten phá hủy cấu trúc bức ảnh
Mỗi pixel trong ảnh phụ thuộc vào các pixel "hàng xóm". Flatten làm mất đi quan hệ "hàng xóm" (spatial relationship) này

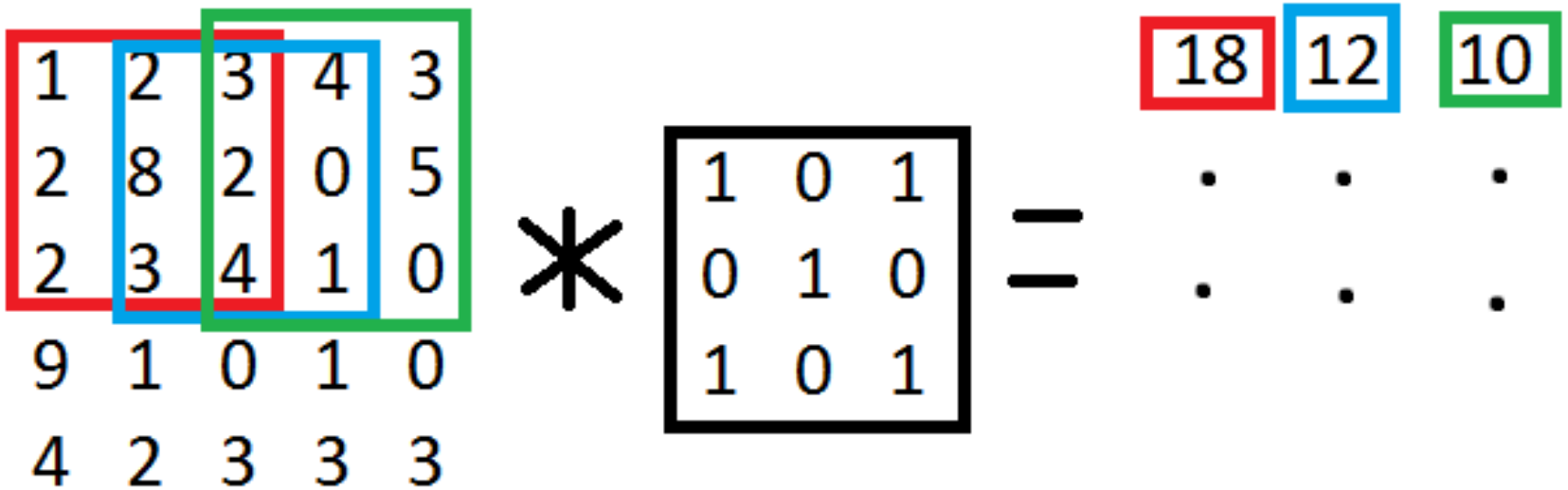
Độ phức tạp của model

- Ảnh 256x256 $\rightarrow 256 \times 256 = 65536 = 2^{16}$ pixels
- Linear model cần 2^{16} trọng số (weight)
- model phức tạp $\rightarrow$ nhiều trọng số $\rightarrow$ tính toán phức tạp

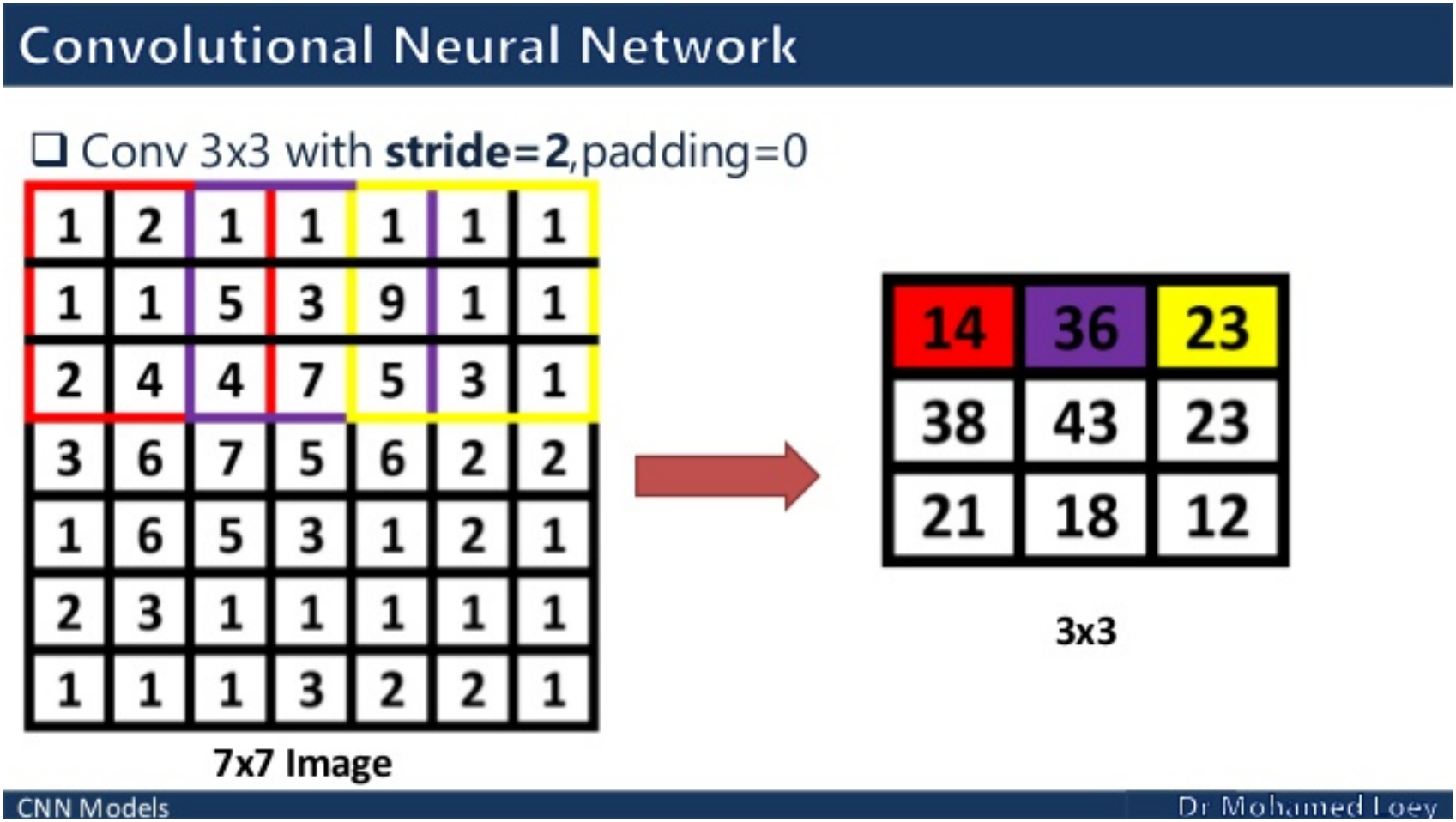
Giải pháp

Tích chập (Convolution)

"Nhân" các ma trận con (trái) vs ma trận tham số (phải) để được một ma trận mới chứa thông tin về mối quan hệ giữa các pixel "hàng xóm"

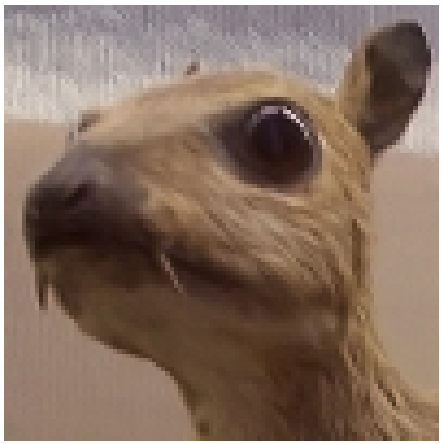


Giải pháp-Tích chập



Tích chập-hình dung

Input image



Convolution Kernel

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Feature map

Tích chập-Công dụng

- Giữ quan hệ "hàng xóm" (spatial relationship). Ví dụ, ma trận tích chập phù hợp có thể giúp tìm cạnh, nét của ảnh.

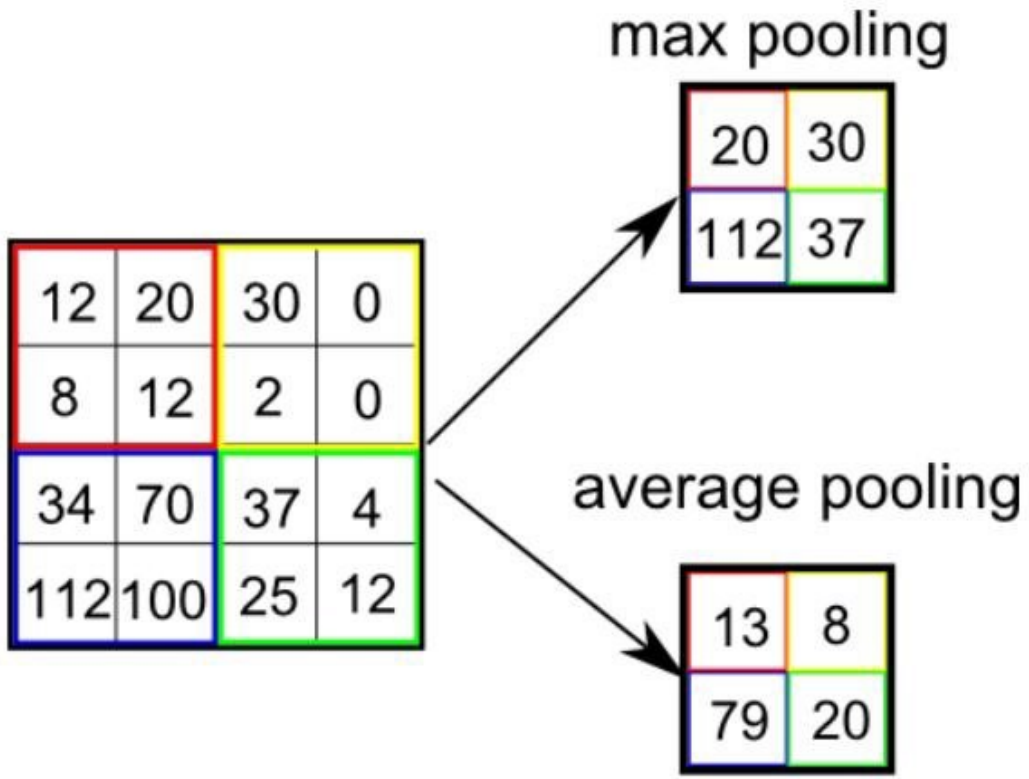
Tích chập-Công dụng

- Giữ quan hệ "hàng xóm" (spatial relationship). Ví dụ, ma trận tích chập phù hợp có thể giúp tìm cạnh, nét của ảnh.
- Giảm độ lớn của input, nhất là khi dùng stride lớn.

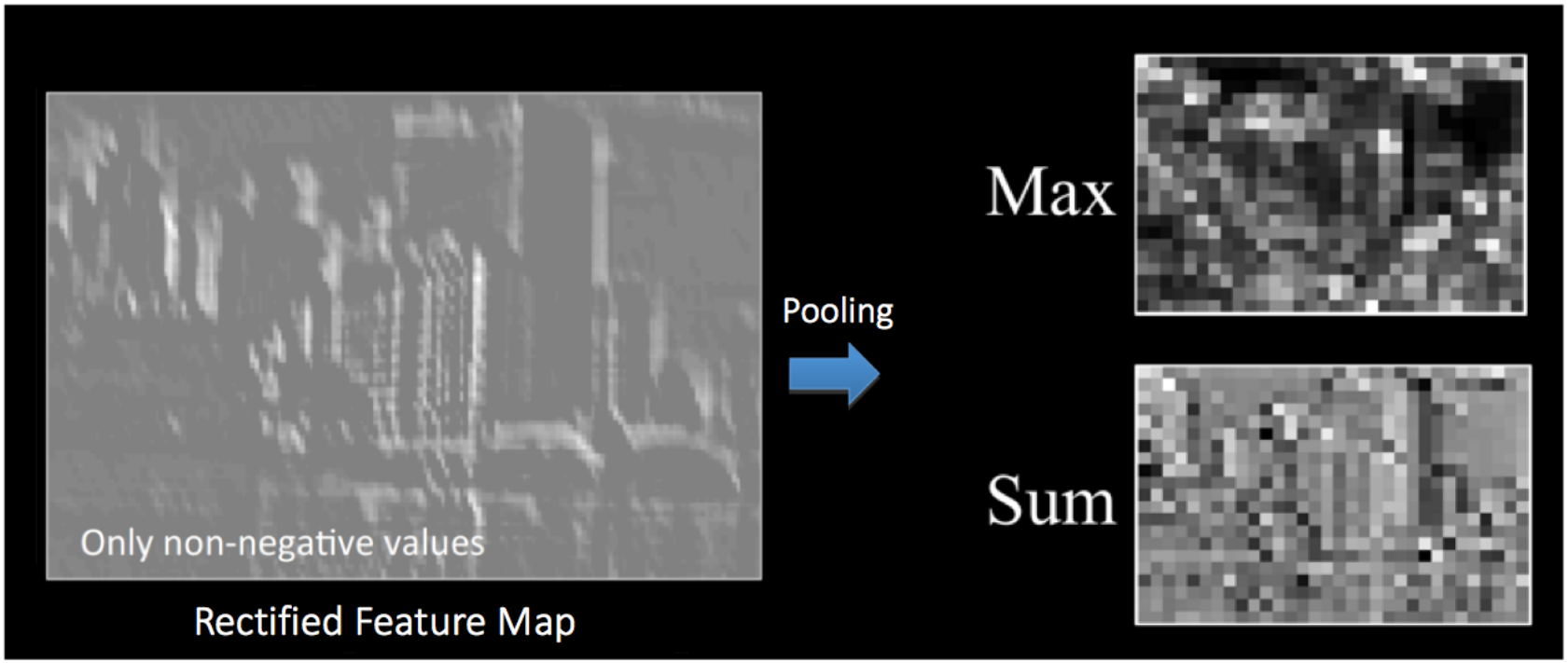
Giải pháp-Pooling

Pooling

Tạo ma trận mới từ max/average của các ma trận con



Pooling-hình dung



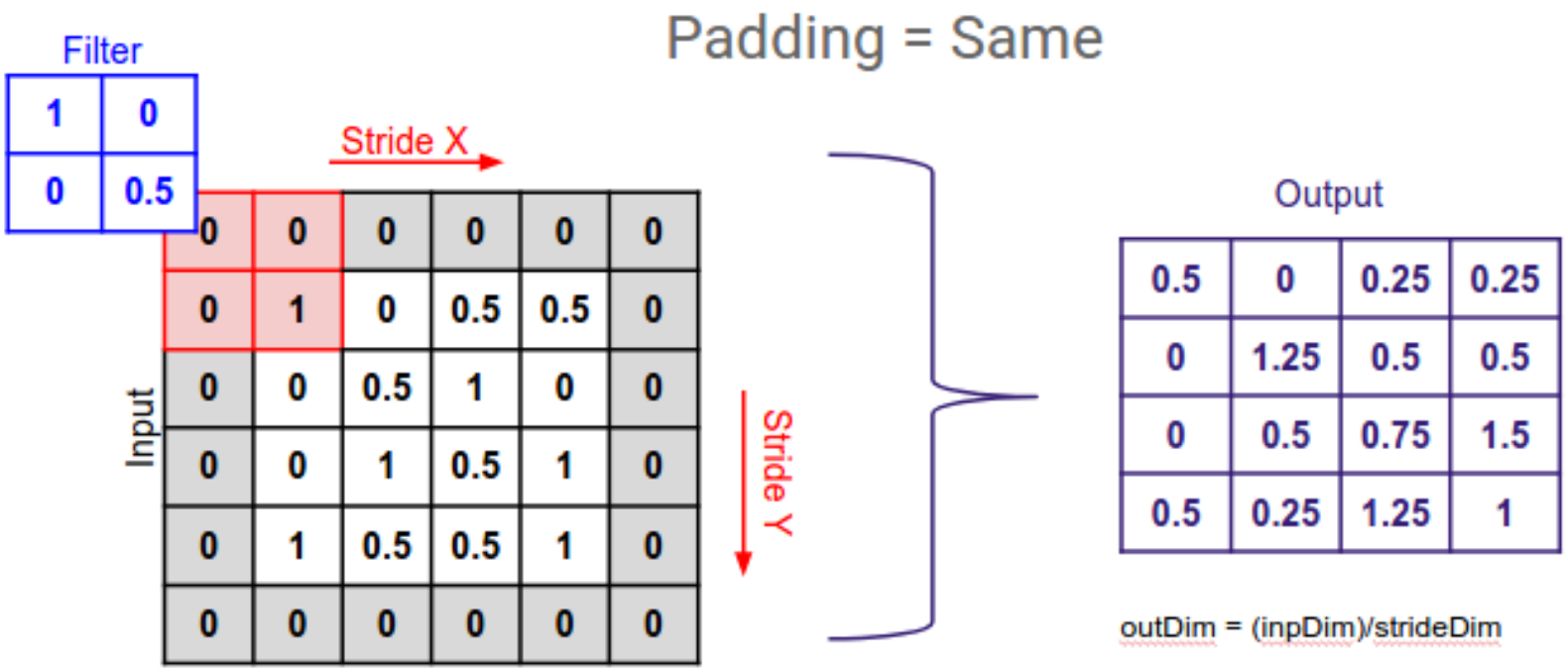
Pooling-Công dụng

- Giảm độ lớn của input $\rightarrow$ model đơn giản hơn, tính toán nhanh hơn.
  Có thể coi như một cách giảm độ phân giải của ảnh. Ảnh ít "nét" hơn, tuy nhiên vẫn đủ để nhận dạng.

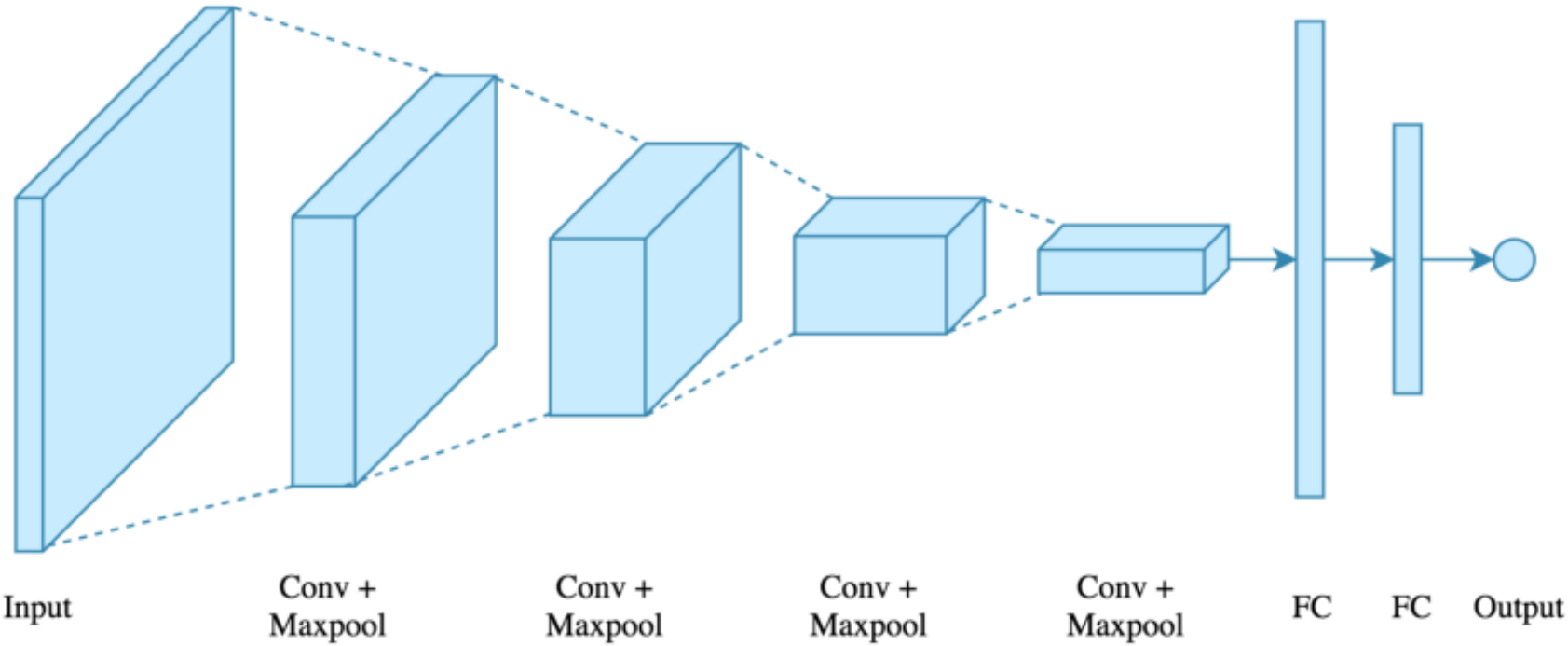
Pooling-Công dụng

- Giảm độ lớn của input $\rightarrow$ model đơn giản hơn, tính toán nhanh hơn.
  Có thể coi như một cách giảm độ phân giải của ảnh. Ảnh ít "nét" hơn, tuy nhiên vẫn đủ để nhận dạng.
- Giảm ảnh hưởng của lỗi, vì chỉ chọn max từ một nhóm pixels.

Padding


Padding = Same
Filter
1
0
0
0.5
Stride X
Stride Y
Input
0 0 0 0 0 0
0 1 0 0.5 0.5 0
0 0 0.5 1 0 0
0 0 1 0.5 1 0
0 1 0.5 0.5 1 0
0 0 0 0 0 0
Output
0.5 0 0.25 0.25
0 1.25 0.5 0.5
0 0.5 0.75 1.5
0.5 0.25 1.25 1
outDim = (inpDim)/strideDim

Cấu trúc mạng CNN



Cấu trúc mạng CNN

- 4 layer đầu (bên trái) kết hợp Convolution và Pooling để trích xuất thông tin tổng quát (high-level feature) từ bức ảnh đầu vào

Cấu trúc mạng CNN

- 4 layer đầu (bên trái) kết hợp Convolution và Pooling để trích xuất thông tin tổng quát (high-level feature) từ bức ảnh đầu vào
- Tiếp theo, tập hợp những thông tin này vào vector 1 chiều.

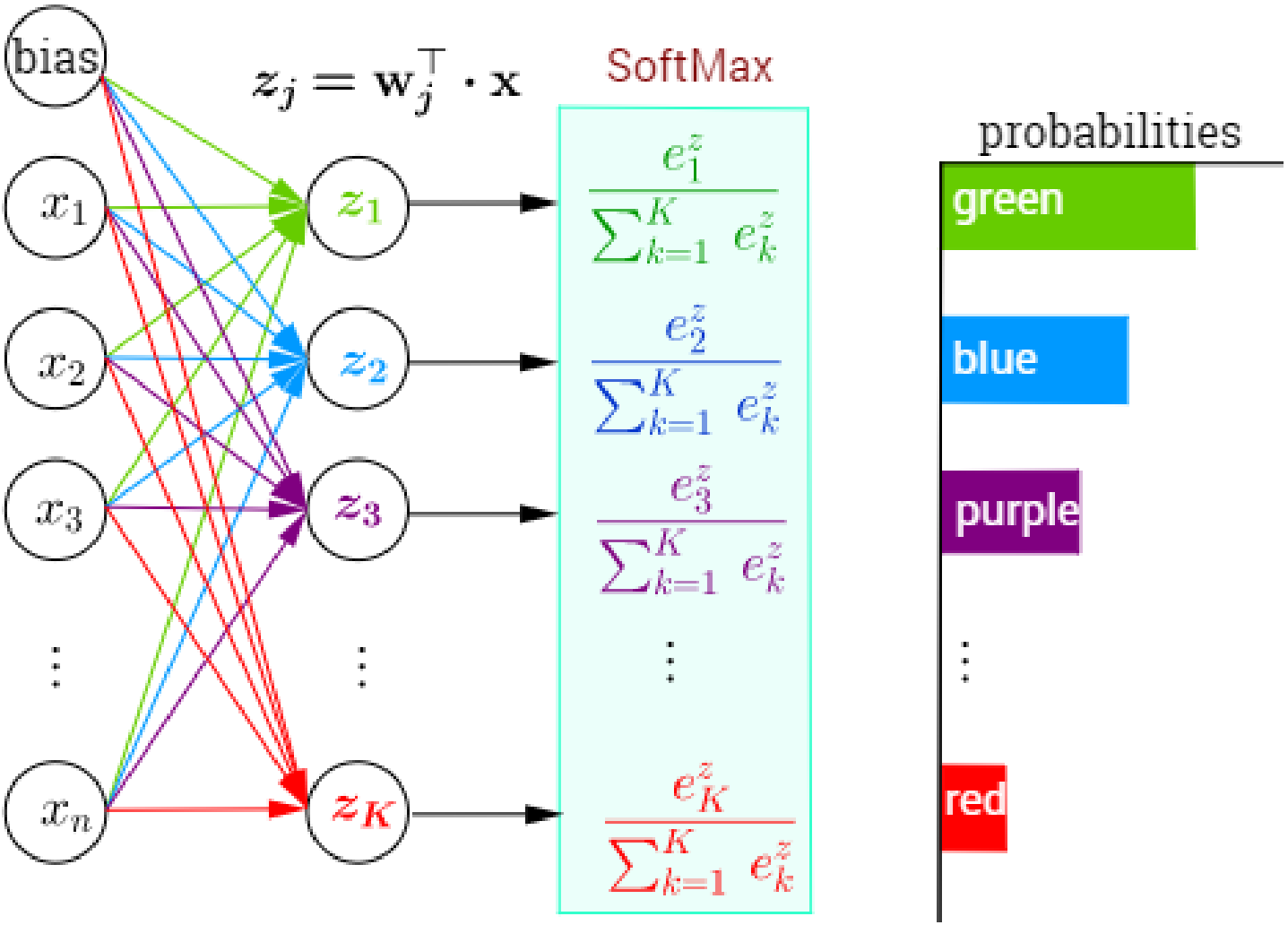
Cấu trúc mạng CNN

- 4 layer đầu (bên trái) kết hợp Convolution và Pooling để trích xuất thông tin tổng quát (high-level feature) từ bức ảnh đầu vào
- Tiếp theo, tập hợp những thông tin này vào vector 1 chiều.
- Vector này là đầu vào cho mạng NN bao gồm các lớp Fully-Connected (FC) (còn gọi là Dense).

Cấu trúc mạng CNN

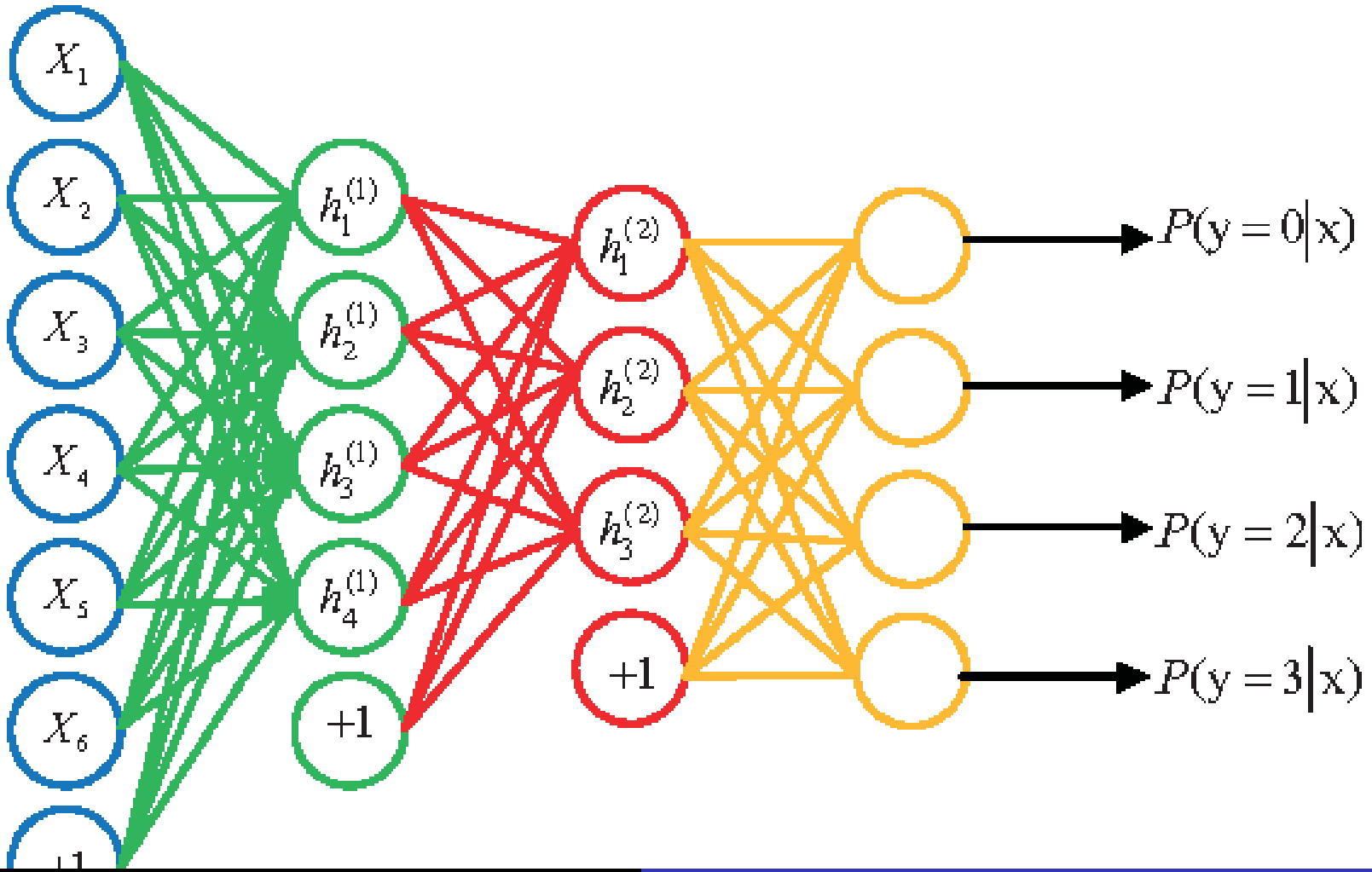
- 4 layer đầu (bên trái) kết hợp Convolution và Pooling để trích xuất thông tin tổng quát (high-level feature) từ bức ảnh đầu vào
- Tiếp theo, tập hợp những thông tin này vào vector 1 chiều.
- Vector này là đầu vào cho mạng NN bao gồm các lớp Fully-Connected (FC) (còn gọi là Dense).
- Ngoài ra, có thể có thêm 1 lớp softmax để phân loại hình ảnh.

Softmax

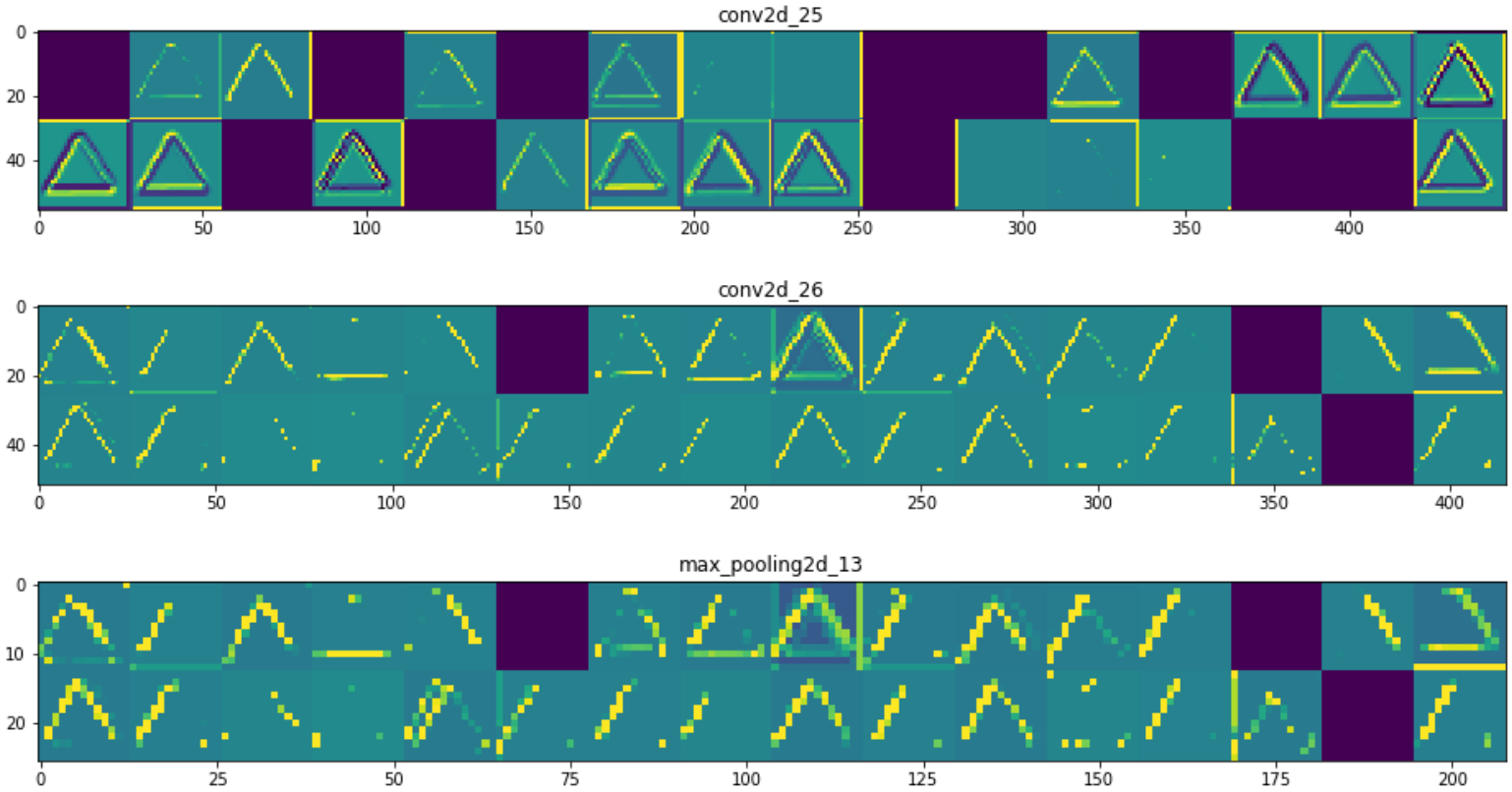

bias
x_1
x_2
x_3
x_n
$z_j = \mathbf{w}_j^\top \cdot \mathbf{x}$
z_1
z_2
z_3
z_K
SoftMax
$\frac{e_1^z}{\sum_{k=1}^K e_k^z}$
$\frac{e_2^z}{\sum_{k=1}^K e_k^z}$
$\frac{e_3^z}{\sum_{k=1}^K e_k^z}$
$\frac{e_K^z}{\sum_{k=1}^K e_k^z}$
probabilities
green
blue
purple
red

Softmax

op autoencoder [12].



Intermediate Ouput-Hình dung



Phân loại hình ảnh-Hình dung

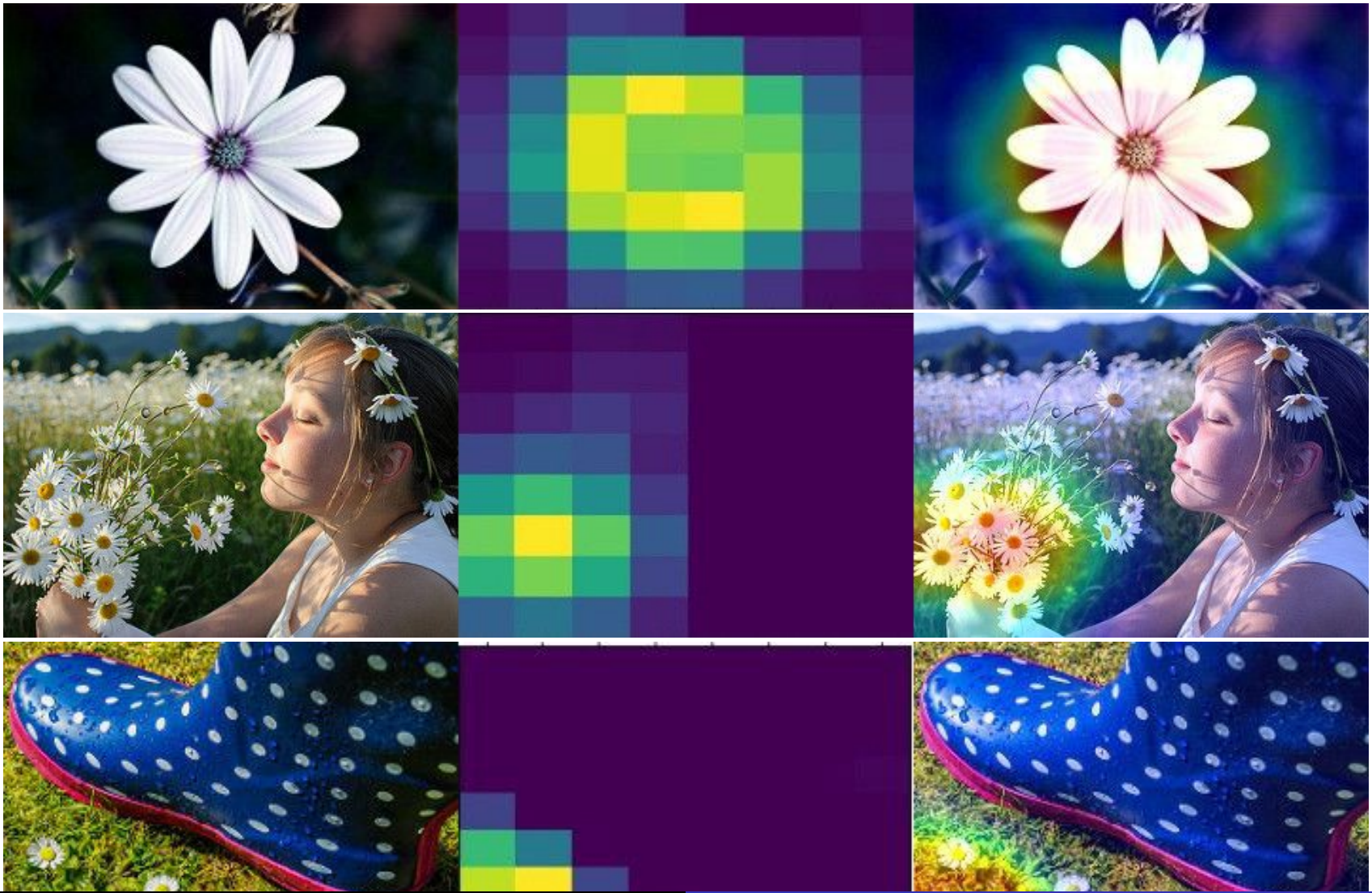


Image Segmentation (Phân chia hình ảnh)




Image Segmentation-Xe tự lái

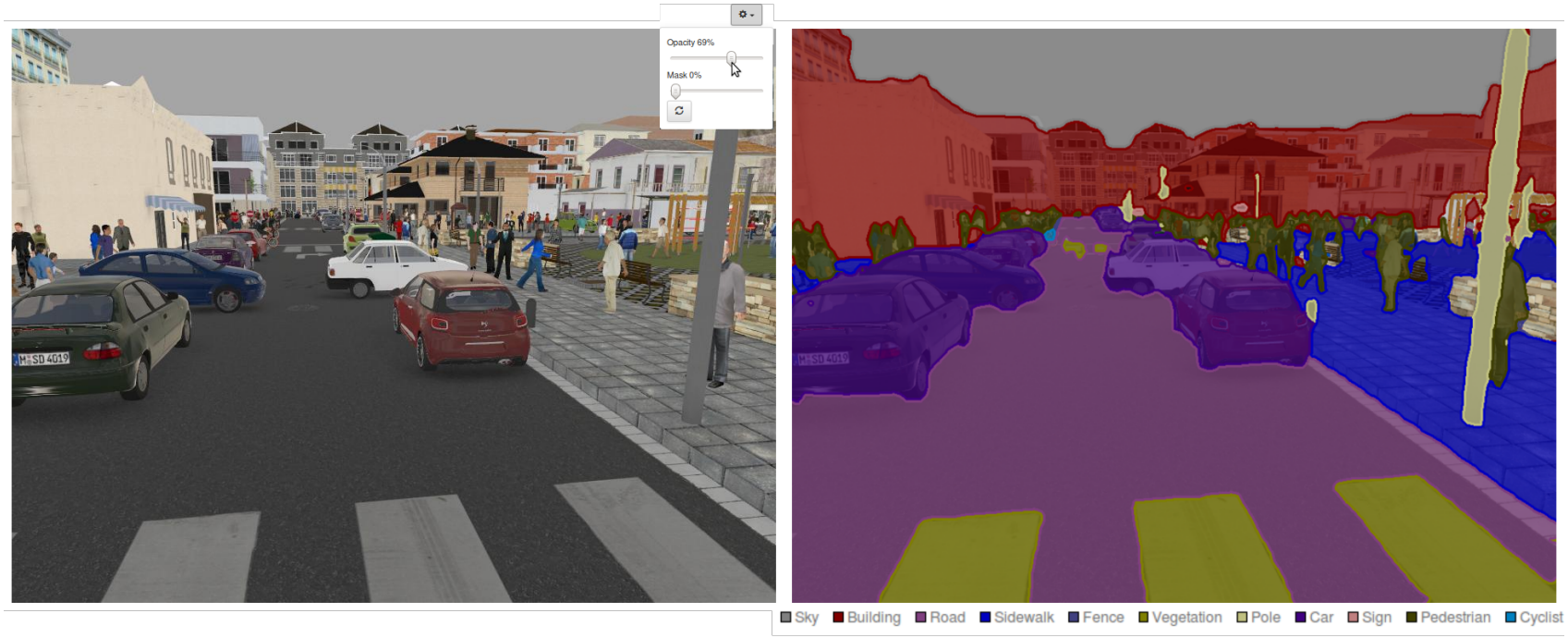


Image Classification (Phân loại hình ảnh)

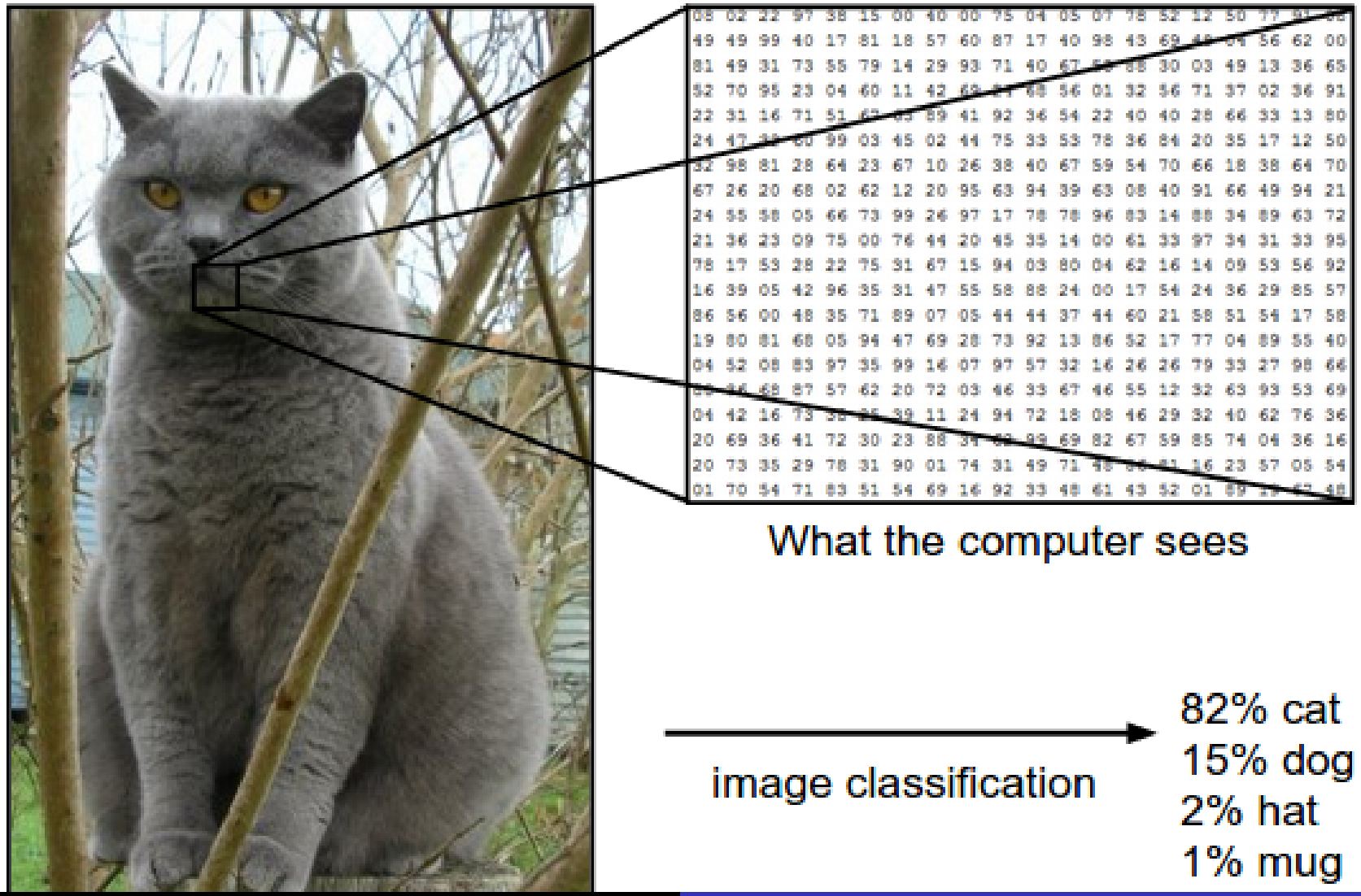
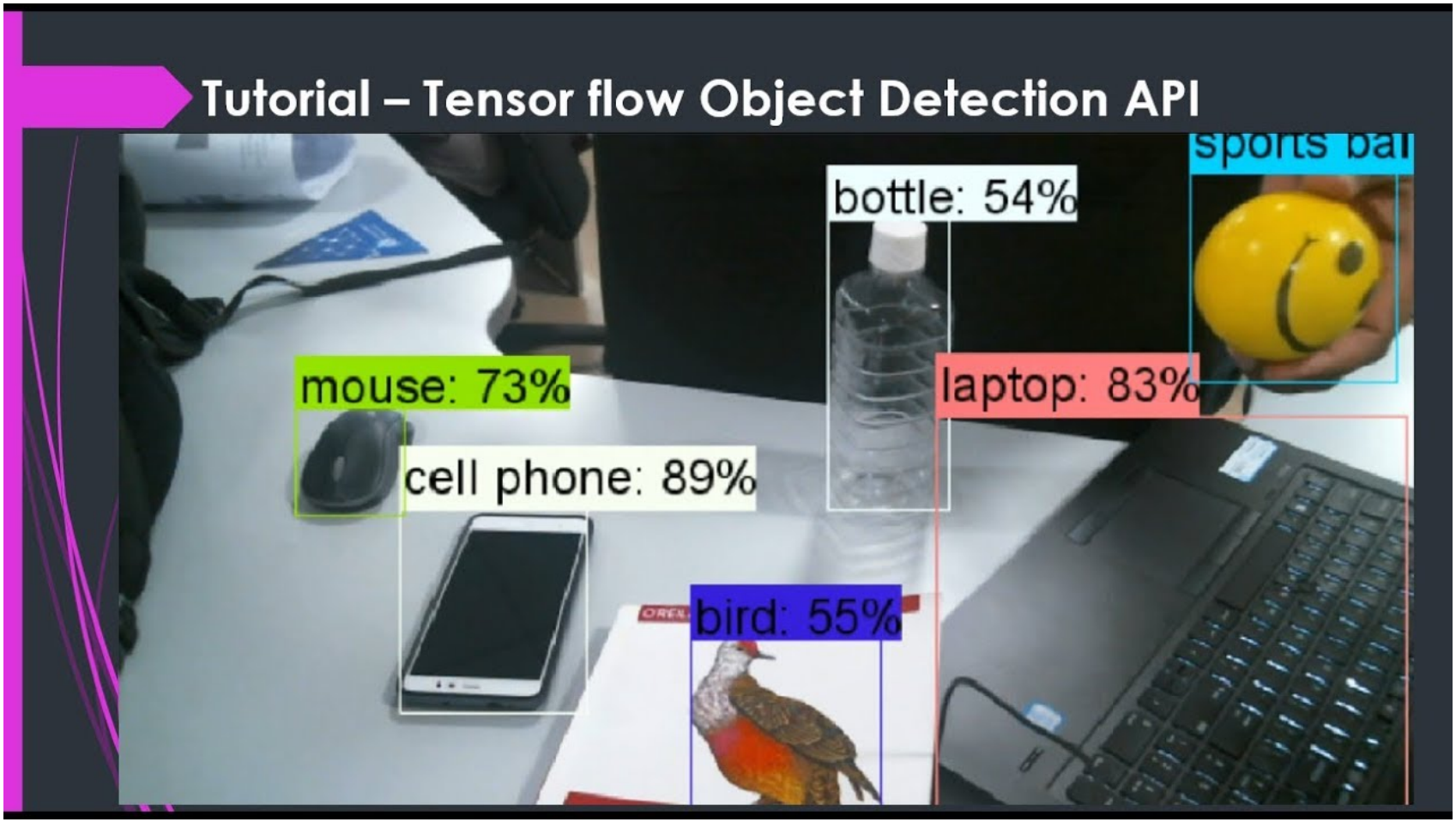


Image Detection (Phát hiện hình ảnh)



CNN-Ưu điểm

- rất thích hợp cho nhận dạng hình ảnh ($>$ con người).

CNN-Ưu điểm

- rất thích hợp cho nhận dạng hình ảnh ($>$ con người).
- tiết kiệm thời gian (có thể dùng pretrained weight cho các layer đầu)

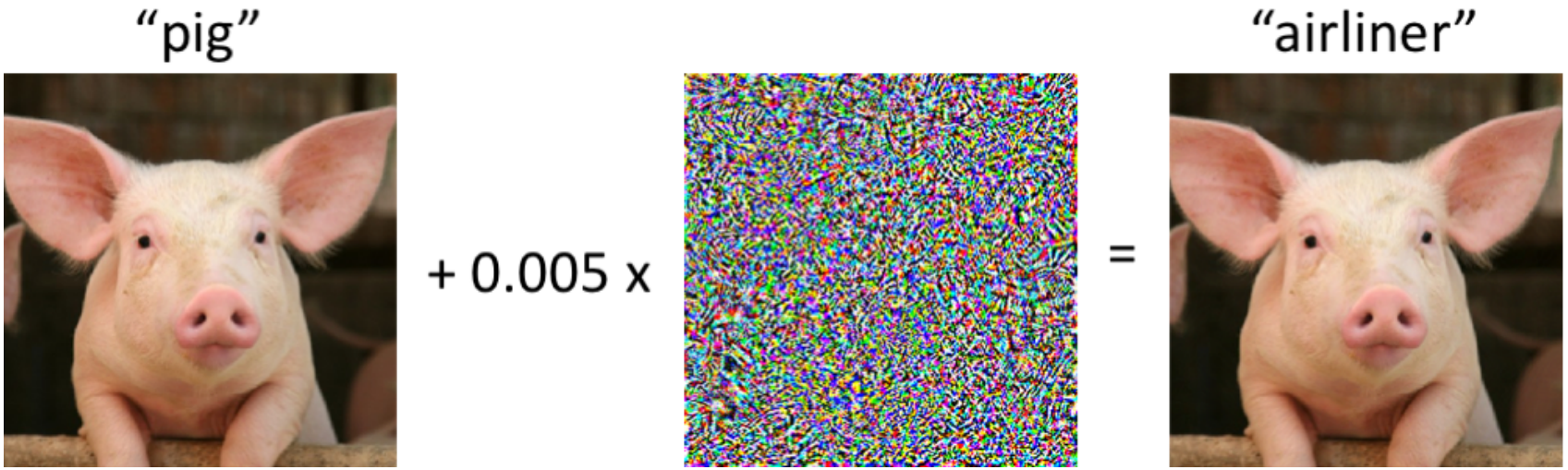
CNN-Nhược điểm

- Cần nhiều data

CNN-Nhược điểm

- Cần nhiều data
- Blackbox $\rightarrow$ chưa thể chứng minh về khả năng convergent, chưa giải nghĩa được model.

Heo hay máy bay



Sơ lược

Xử lý ngôn ngữ tự nhiên

Bài toán

Dầu vào: văn bản (câu văn, đoạn văn), biểu diễn dưới dạng chuỗi chữ/từ ngữ
Đầu ra: Sắc thái (sentiment), bản dịch (vd: Việt $\rightarrow$ Anh), dự đoán chữ tiếp theo (vd: autotype)

Giải pháp bằng mạng NN

Ý tưởng

Coi chuỗi chữ như vector 1 chiều, rồi xử lý bằng mạng NN

Vấn đề

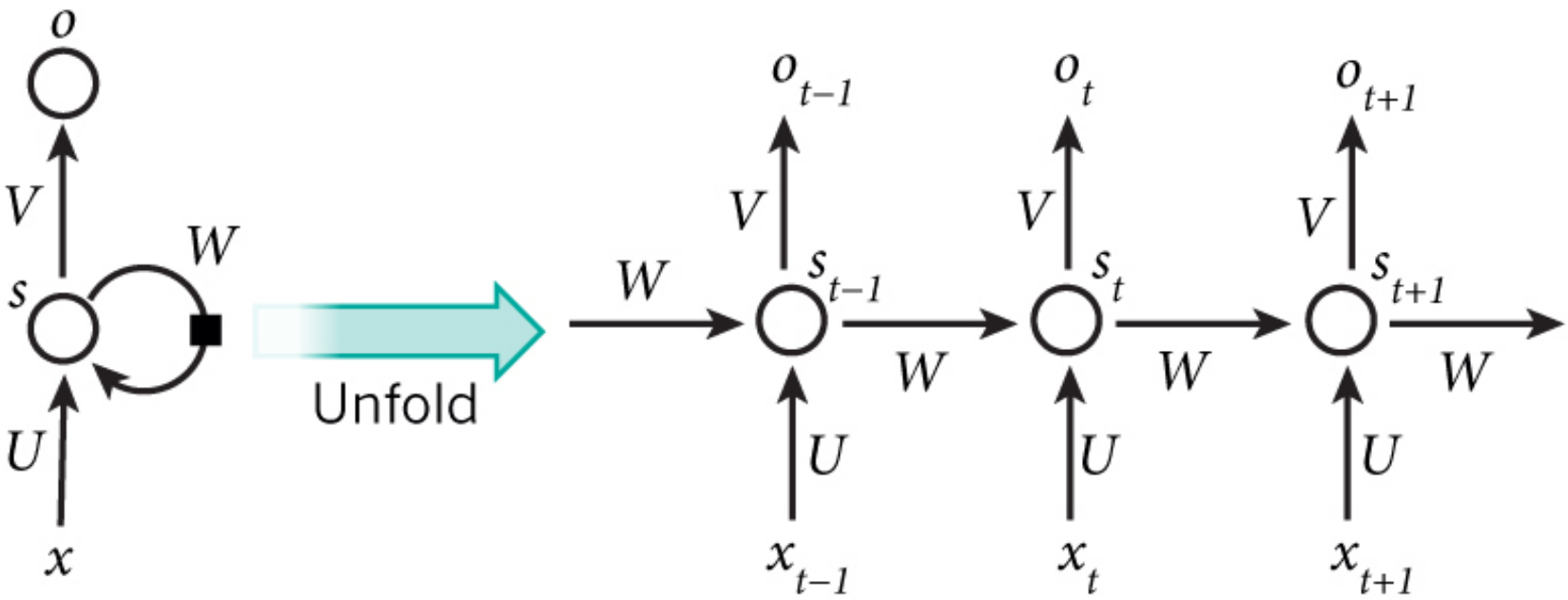
Các phần tử trong một vector là độc lập (có thể xáo trộn).
Các chữ trong chuỗi cần được sắp xếp theo thứ tự. Các chữ xếp trước ảnh hưởng đến các chữ sau.

Giải pháp

Trạng thái ẩn (Hidden state)

Cần một "bộ nhớ" để lưu trữ thông tin từ các chữ phía trước
$\rightarrow$ trạng thái ẩn

Cấu trúc Mạng RNN



Cấu trúc Mạng RNN

- x_t là đầu vào tại bước t.

Cấu trúc Mạng RNN

- x_t là đầu vào tại bước t.
- s_t là trạng thái ẩn tại bước t. Nó chính là bộ nhớ của mạng, được tính toán dựa trên cả các trạng thái ẩn phía trước và đầu vào tại bước đó: $s_t = f(Ux_t + Ws_{t-1})$

Cấu trúc Mạng RNN

- x_t là đầu vào tại bước t.
- s_t là trạng thái ẩn tại bước t. Nó chính là bộ nhớ của mạng, được tính toán dựa trên cả các trạng thái ẩn phía trước và đầu vào tại bước đó: $s_t = f(Ux_t + Ws_{t-1})$
- o_t là đầu ra tại bước t

Kí ức ngắn hạn-dài hạn

Kí ức ngắn hạn-dài hạn

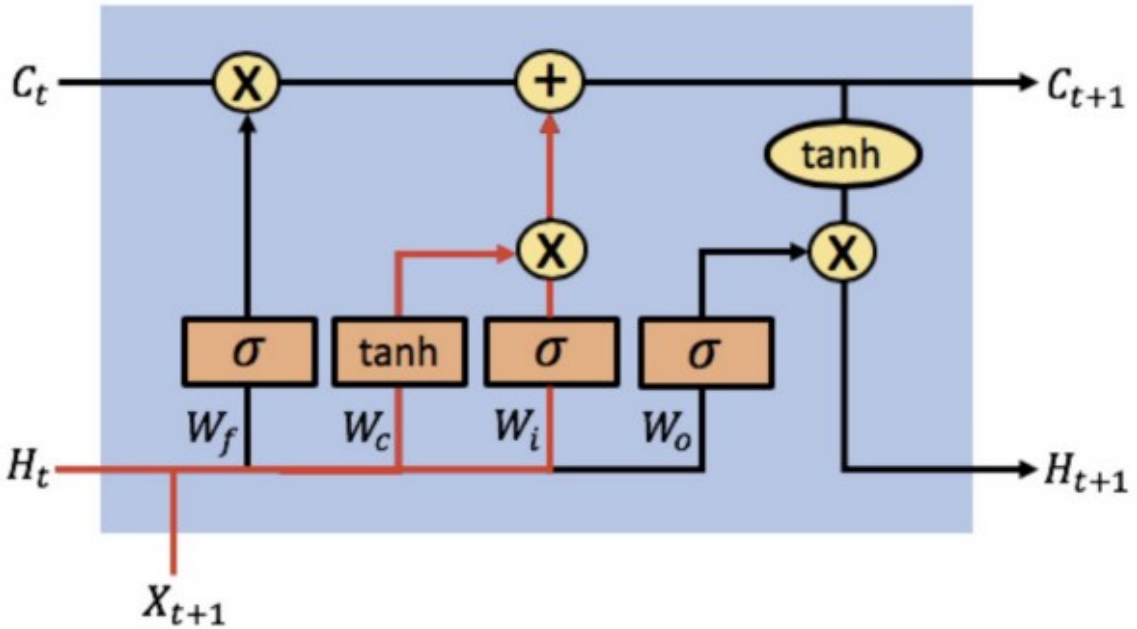
Ngắn hạn: vị trí đồ vật, chữ trước trong câu, $\cdots$
Dài hạn: địa chỉ nhà, định lý, $\cdots$
Cần quên đi để nhớ những thứ quan trọng

Dịch văn bản

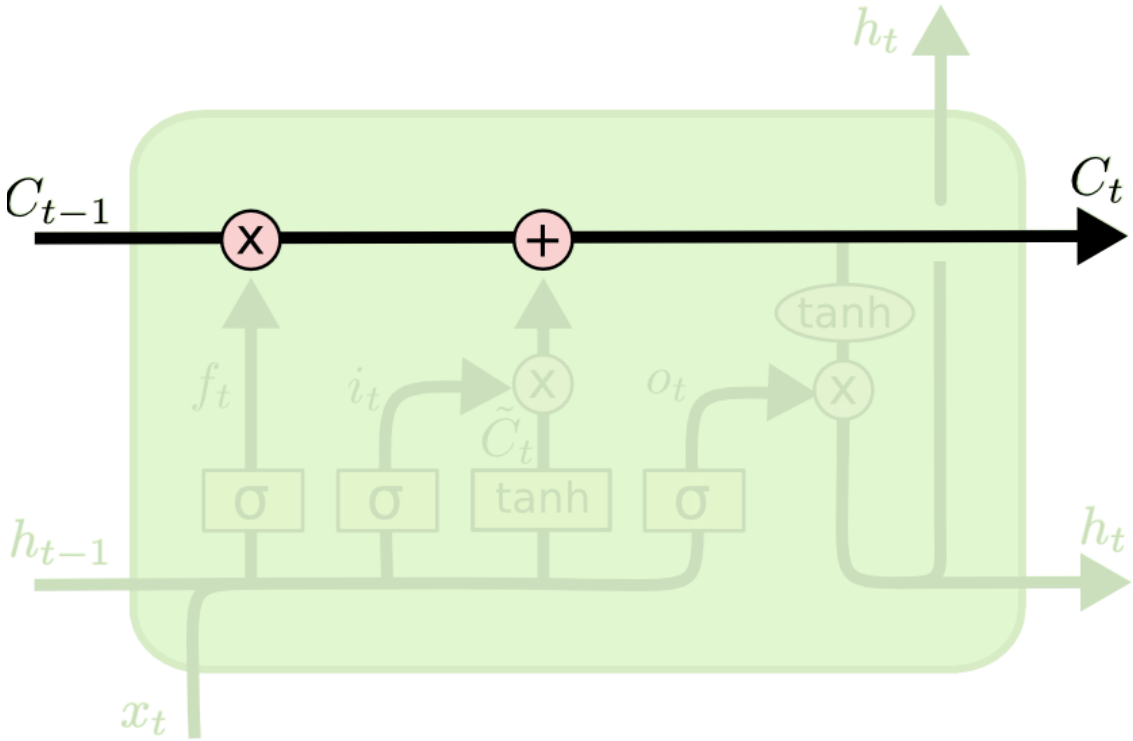
Tầm ảnh hưởng của từ đi trước và từ đi sau:
cùng câu $>$ cùng đoạn $>$ cùng văn bản
Ví dụ:

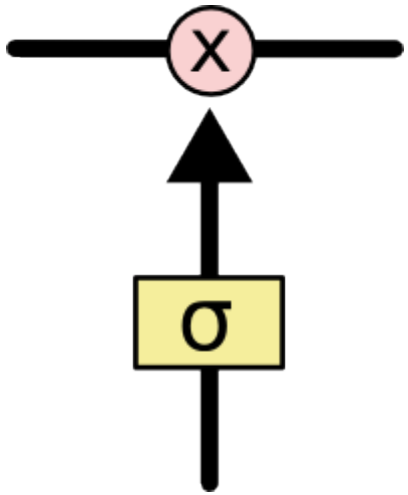
- Bạn thật **thông minh**. $\rightarrow$ You are so **intelligent**.
- Đường ống vừa **thông. Minh** rửa chén. $\rightarrow$ The pipe is **unclogged. Minh** washes the dishes.

Cấu trúc Khối LSTM

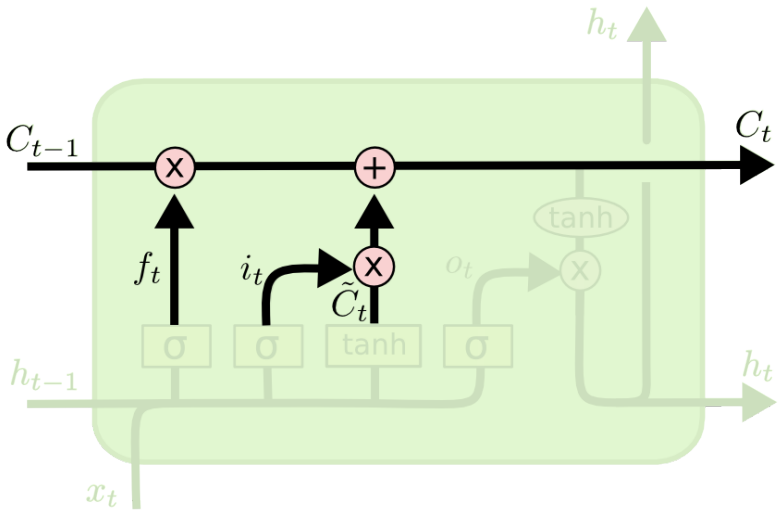


C-line & Gate



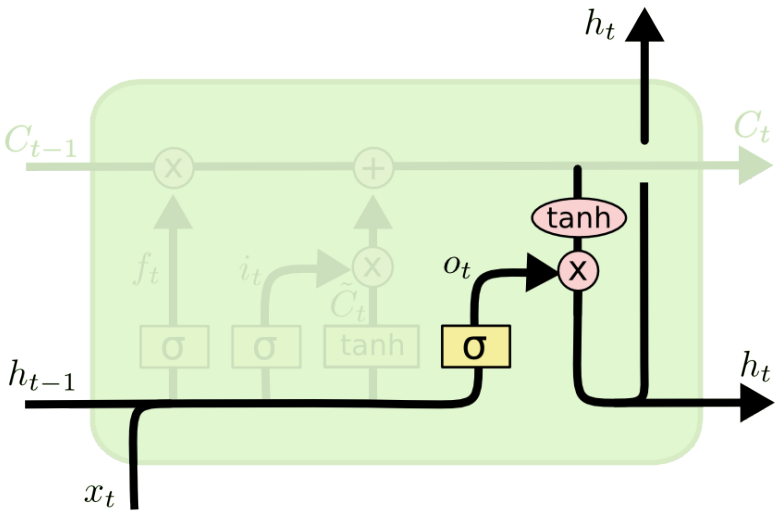


Công thức cho C



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Công thức cho output H



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Cấu trúc Khối LSTM

- X_t là đầu vào tại bước t
- C_t là "trạng thái" (cell state) tại bước t, được coi như "bộ nhớ dài hạn"
- H_t là đầu ra tại bước t, được coi như "bộ nhớ ngắn hạn"

Cấu trúc Khối LSTM

- X_t là đầu vào tại bước t
- C_t là "trạng thái" (cell state) tại bước t, được coi như "bộ nhớ dài hạn"
- H_t là đầu ra tại bước t, được coi như "bộ nhớ ngắn hạn"
- $C_t = f_t \cdot C_{t-1} + B_t$ với $f_t \in [0, 1], B_t \in [-1, 1]$ phụ thuộc vào H_{t-1} và X_t.
  $f_t = 0$ tương đương khởi động lại bộ nhớ hay "quên đi"