HỌC SÂU

BÀI 4. MẠNG NƠ-RON TÍCH CHẬP (CNN)

Mạng nơ-ron tích chập

Mạng nơ-ron tích chập (Convolutional Neural Network - CNN) là một loại mạng nơ-ron đặc biệt được thiết kế để xử lý dữ liệu có cấu trúc lưới, đặc biệt là hình ảnh. CNN đã trở thành công nghệ nền tảng trong lĩnh vực thị giác máy tính (Computer Vision) và nhiều ứng dụng xử lý hình ảnh khác.

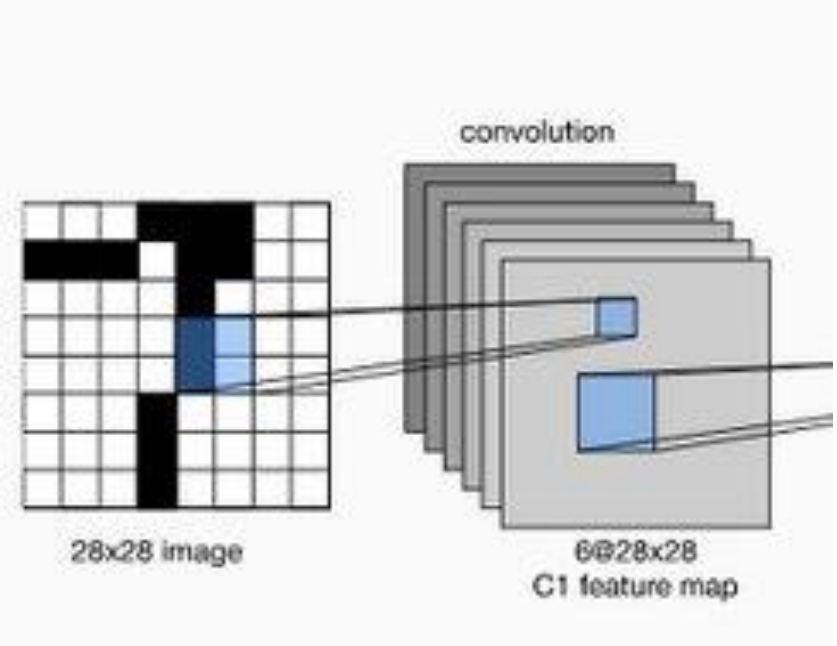
1.1. Tại sao cần CNN?

Mạng nơ-ron thông thường (Fully Connected Neural Network) gặp phải một số hạn chế khi xử lý hình ảnh:

- **Số lượng tham số lớn**: Một hình ảnh nhỏ 28x28 pixel đã cần 784 nơ-ron đầu vào
- **Không bảo toàn thông tin không gian**: Mất đi mối quan hệ giữa các pixel lân cận
- **Không bất biến với dịch chuyển**: Không nhận dạng được đối tượng khi vị trí thay đổi

CNN giải quyết những vấn đề này bằng cách:

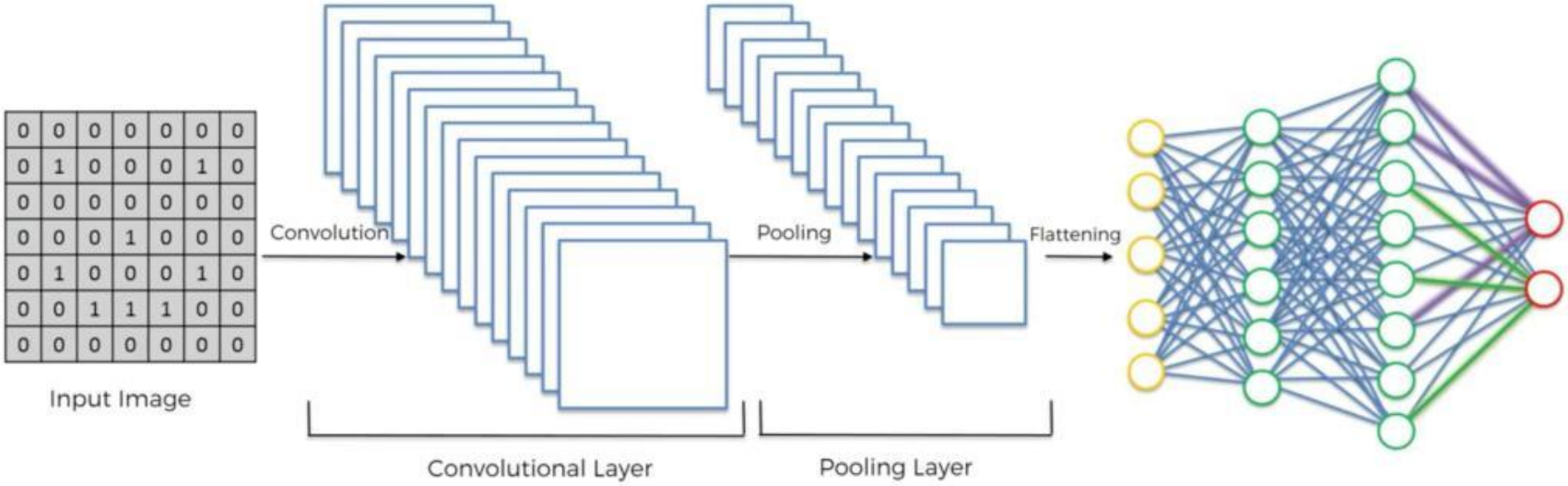
- Sử dụng các bộ lọc (filter) nhỏ thay vì kết nối đầy đủ
- Bảo toàn thông tin không gian thông qua cấu trúc lưới
- Tạo ra tính bất biến dịch chuyển (translation invariance)



2. Cấu trúc của mạng CNN

Một mạng CNN điển hình bao gồm các lớp sau:

- Lớp tích chập (Convolutional Layer)
- Hàm kích hoạt (Activation Function)
- Lớp gộp (Pooling Layer)
- Lớp Dropout
- Lớp kết nối đầy đủ (Fully Connected Layer)



2.1. Phép tích chập

Phép tích chập

- Sử dụng các bộ lọc (kernel/filter) nhỏ (thường là 3x3 hoặc 5x5)
- Trượt (slide) bộ lọc qua toàn bộ hình ảnh đầu vào
- Tại mỗi vị trí, thực hiện phép tích chập: nhân từng phần tử của bộ lọc với vùng tương ứng trên hình ảnh và cộng lại

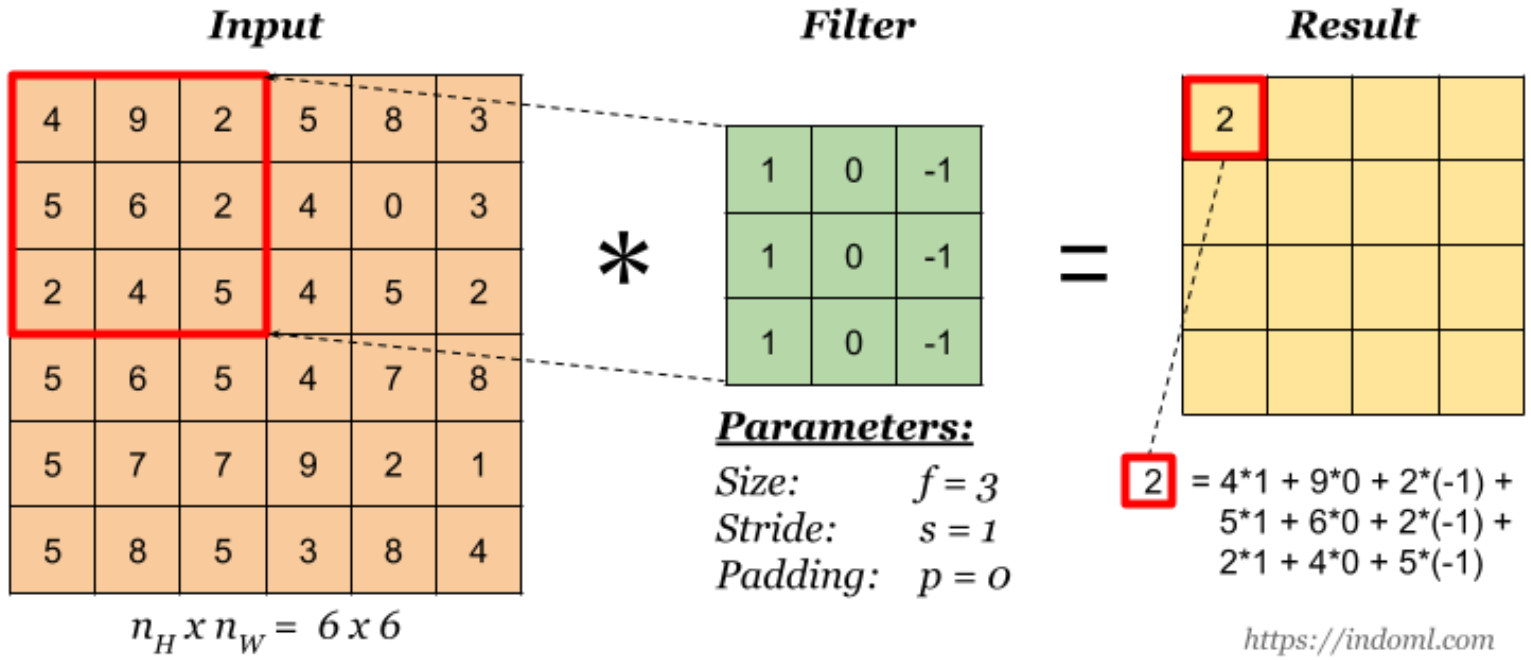
Bộ lọc

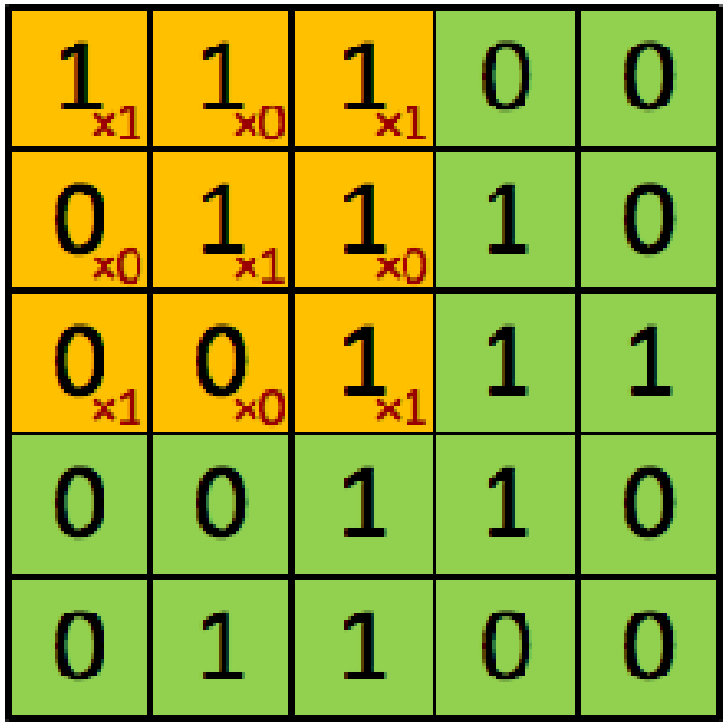
Bộ lọc (hay còn gọi là kernel hoặc filter) trong mạng nơ-ron tích chập (CNN) là một ma trận nhỏ với các trọng số (weight) mà mạng nơ-ron học được trong quá trình huấn luyện. Đây là một khái niệm cốt lõi trong CNN, thực hiện chức năng trích xuất đặc trưng từ dữ liệu đầu vào.

Bộ lọc là một ma trận nhỏ (thường có kích thước 3x3, 5x5, hoặc 7x7) chứa các giá trị trọng số.

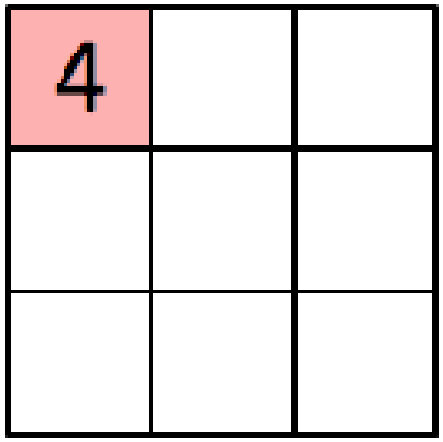
Filter

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


Input
Filter
Result
4 9 2 5 8 3
5 6 2 4 0 3
2 4 5 4 5 2
5 6 5 4 7 8
5 7 7 9 2 1
5 8 5 3 8 4
*
1 0 -1
1 0 -1
1 0 -1
=
2
Parameters:
Size: f = 3
Stride: s = 1
Padding: p = 0
2 = 4*1 + 9*0 + 2*(-1) +
5*1 + 6*0 + 2*(-1) +
2*1 + 4*0 + 5*(-1)
n_H x n_W = 6 x 6
https://indoml.com



Image



Convolved Feature

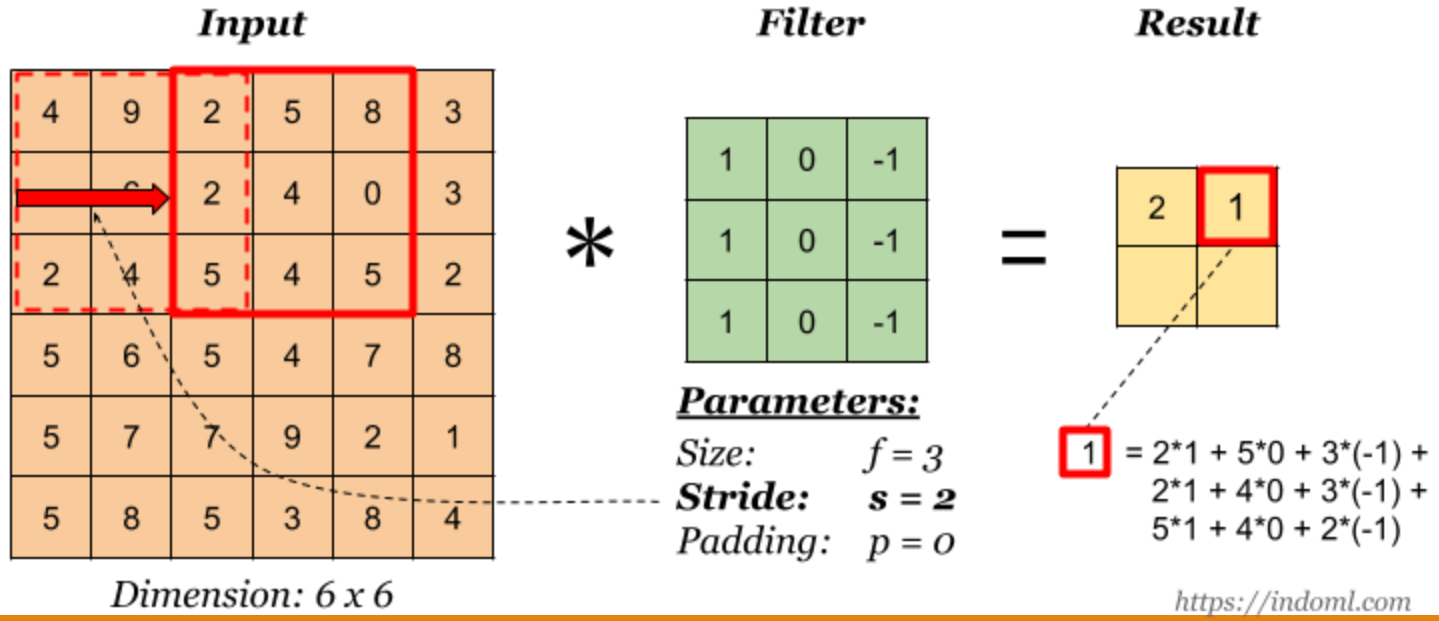
Bộ lọc

- Mỗi bộ lọc được thiết kế để phát hiện một đặc trưng cụ thể trong hình ảnh
- Các bộ lọc ở tầng đầu thường phát hiện đặc trưng đơn giản như cạnh, góc, đường thẳng
- Các bộ lọc ở tầng sau phát hiện đặc trưng phức tạp hơn như mắt, mũi, bánh xe, v.v.

Stride (Bước nhảy)

Stride là khoảng cách (số pixel) mà bộ lọc di chuyển sau mỗi lần tính toán trong quá trình tích chập.

- **Stride = 1**: Bộ lọc di chuyển 1 pixel mỗi lần (tiêu chuẩn)
- **Stride = 2**: Bộ lọc di chuyển 2 pixel mỗi lần (giảm kích thước đầu ra)



Stride (Bước nhảy)

Ảnh hưởng của stride:

- Stride lớn hơn giúp giảm kích thước đầu ra (downsampling)
- Giảm thời gian tính toán
- Có thể làm mất thông tin nếu stride quá lớn

Padding (Đệm)

Padding là quá trình thêm các pixel (thường là 0) xung quanh biên của hình ảnh đầu vào.

Mục đích của padding:

1. Giữ nguyên kích thước đầu ra sau phép tích chập
2. Giữ lại thông tin ở biên của ảnh đầu vào

Các loại padding:

- **Valid padding** (không padding): Kích thước đầu ra nhỏ hơn đầu vào
- **Same padding**: Thêm đủ padding để kích thước đầu ra bằng đầu vào

0	0	0	0	0	0	0
0	60	113	56	139	85	0
0	73	121	54	84	128	0
0	131	99	70	129	127	0
0	80	57	115	69	134	0
0	104	126	123	95	130	0
0	0	0	0	0	0	0

Kernel

0	-1	0
-1	5	-1
0	-1	0

114				

2.2. Phép tính chập với dữ liệu có nhiều kênh chiều sâu

Trong thực tế, hình ảnh thường có nhiều kênh màu (ví dụ: RGB có 3 kênh).

CNN xử lý dữ liệu đa kênh thông qua bộ lọc có cùng số kênh với đầu vào.

2.2. Phép tính chập với dữ liệu có nhiều kênh chiều sâu

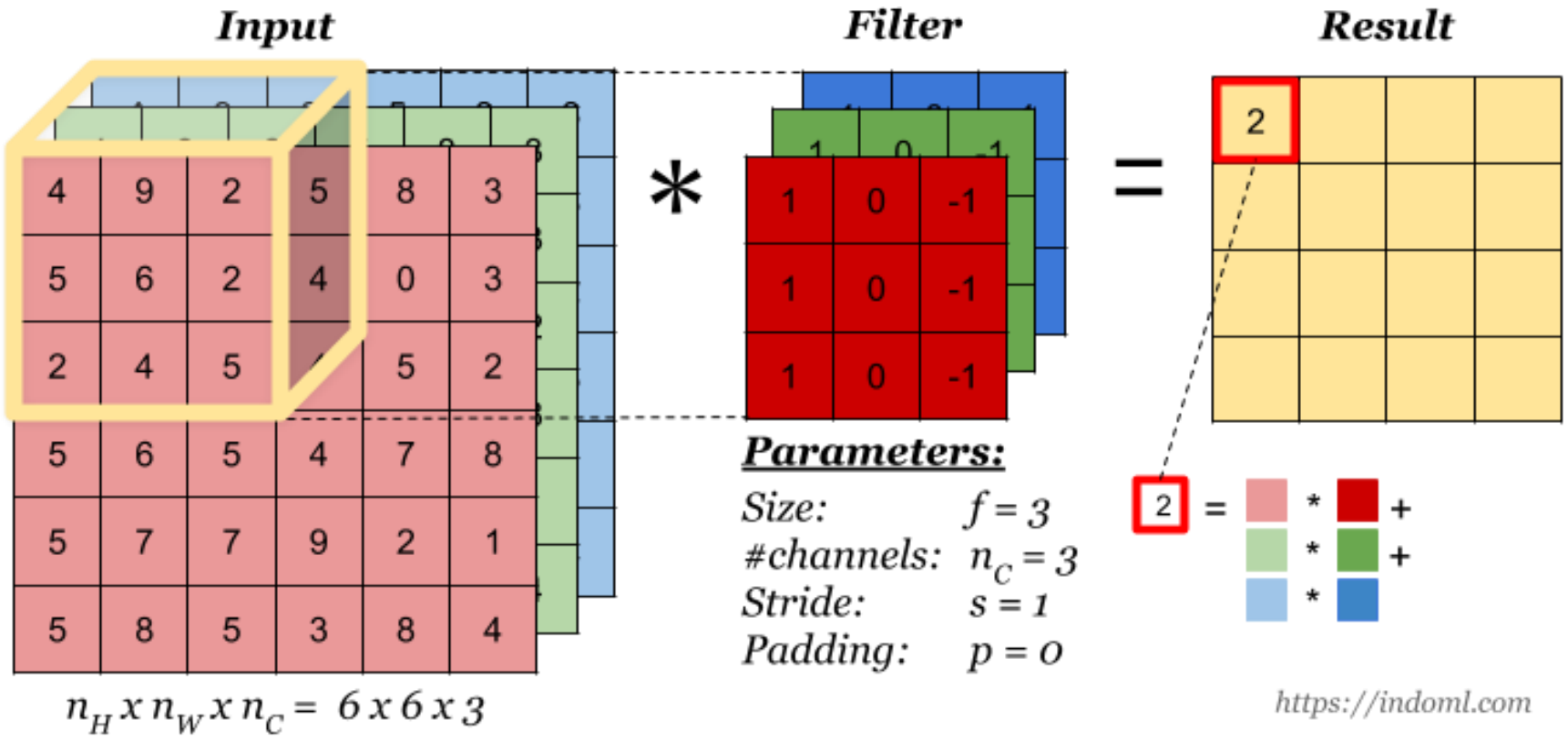
Nguyên lý hoạt động:
1.Đầu vào đa kênh: Ví dụ, ảnh RGB có 3 kênh (kênh đỏ, kênh xanh lá, kênh xanh dương), mỗi kênh là một ma trận 2D.

2.Bộ lọc đa kênh: Mỗi bộ lọc cũng có 3 kênh, tương ứng với 3 kênh của đầu vào.

3.Cách tính:

1. Thực hiện phép tích chập riêng biệt giữa mỗi kênh của đầu vào với kênh tương ứng của bộ lọc
2. Cộng tất cả kết quả lại để tạo ra một feature map đầu ra duy nhất

2.2. Phép tính chập với dữ liệu có nhiều kênh chiều sâu



```
Kênh R:          Kênh G:          Kênh B:
1  2  3  4  5    5  6  7  8  9    9  8  7  6  5
6  7  8  9  10   10 11 12 13 14   4  3  2  1  0
11 12 13 14 15   15 16 17 18 19   1  2  3  4  5
16 17 18 19 20   20 21 22 23 24   6  7  8  9  10
21 22 23 24 25   25 26 27 28 29   11 12 13 14 15
```

```
Kênh R:       Kênh G:       Kênh B:
1  0  1       0  1  0       1  0  1
0  1  0       1  0  1       0  1  0
1  0  1       0  1  0       1  0  1
```

Đầu ra tại vị trí (0,0) sẽ là tổng của 3 phép tích chập:

- Tích chập kênh R: (1×1) + (2×0) + ... = 35
- Tích chập kênh G: (5×0) + (6×1) + ... = 48
- Tích chập kênh B: (9×1) + (8×0) + ... = 31
- Tổng: 35 + 48 + 31 = 114

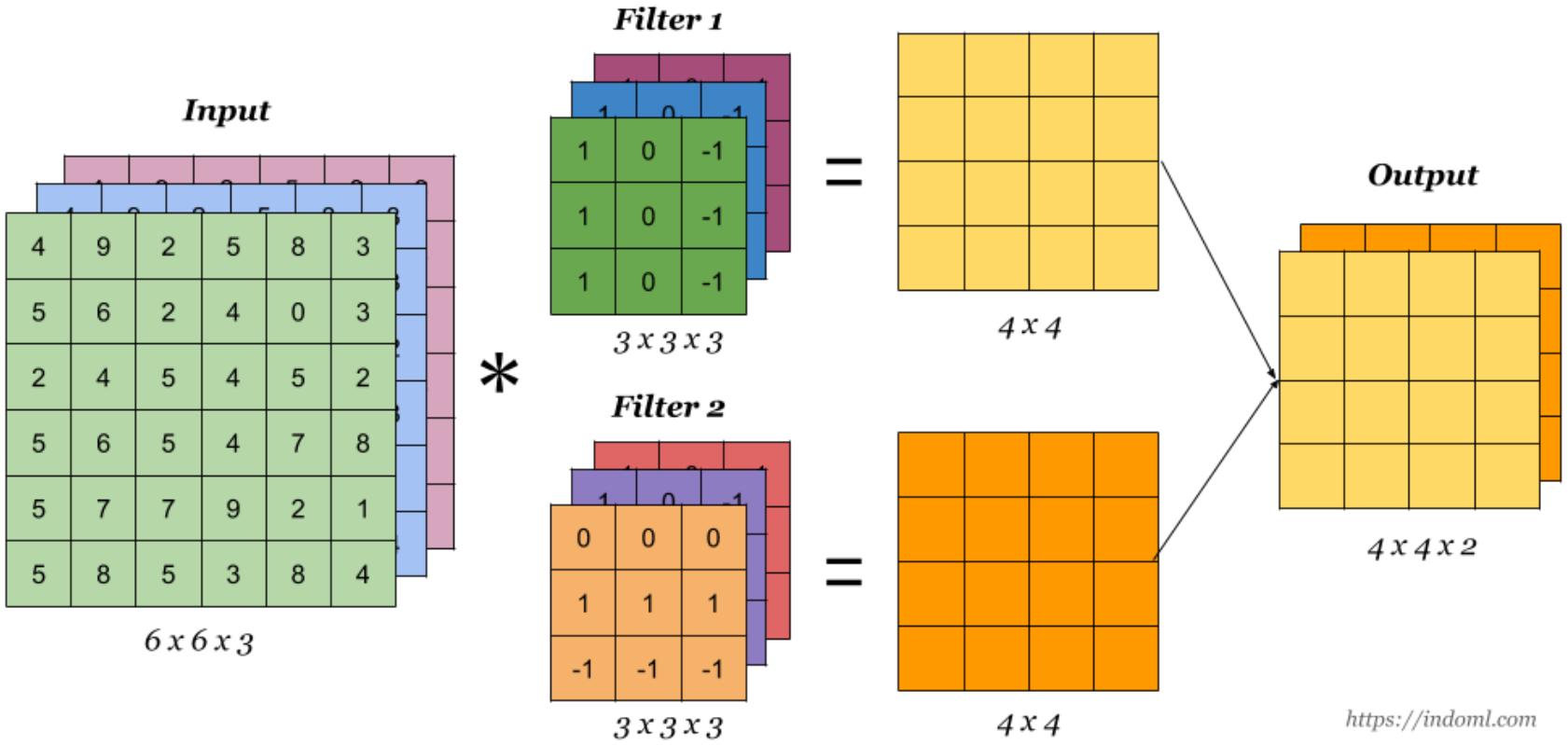
2.3. Phép tính chập với nhiều bộ lọc

Trong CNN thực tế, một lớp tích chập thường sử dụng nhiều bộ lọc khác nhau để trích xuất nhiều loại đặc trưng khác nhau từ dữ liệu đầu vào.

Nguyên lý hoạt động:

1. **Nhiều bộ lọc, mỗi bộ lọc có cùng kích thước**: Mỗi lớp tích chập thường có nhiều bộ lọc (từ vài chục đến hàng trăm).
2. **Mỗi bộ lọc tạo ra một feature map**: Khi áp dụng n bộ lọc, ta sẽ thu được n feature map đầu ra.
3. **Mỗi bộ lọc phát hiện một loại đặc trưng khác nhau**: Ví dụ, một bộ lọc có thể phát hiện cạnh ngang, bộ lọc khác phát hiện cạnh dọc, góc, hoa văn...

2.3. Phép tính chập với nhiều bộ lọc



2.3. Phép tính chập với nhiều bộ lọc

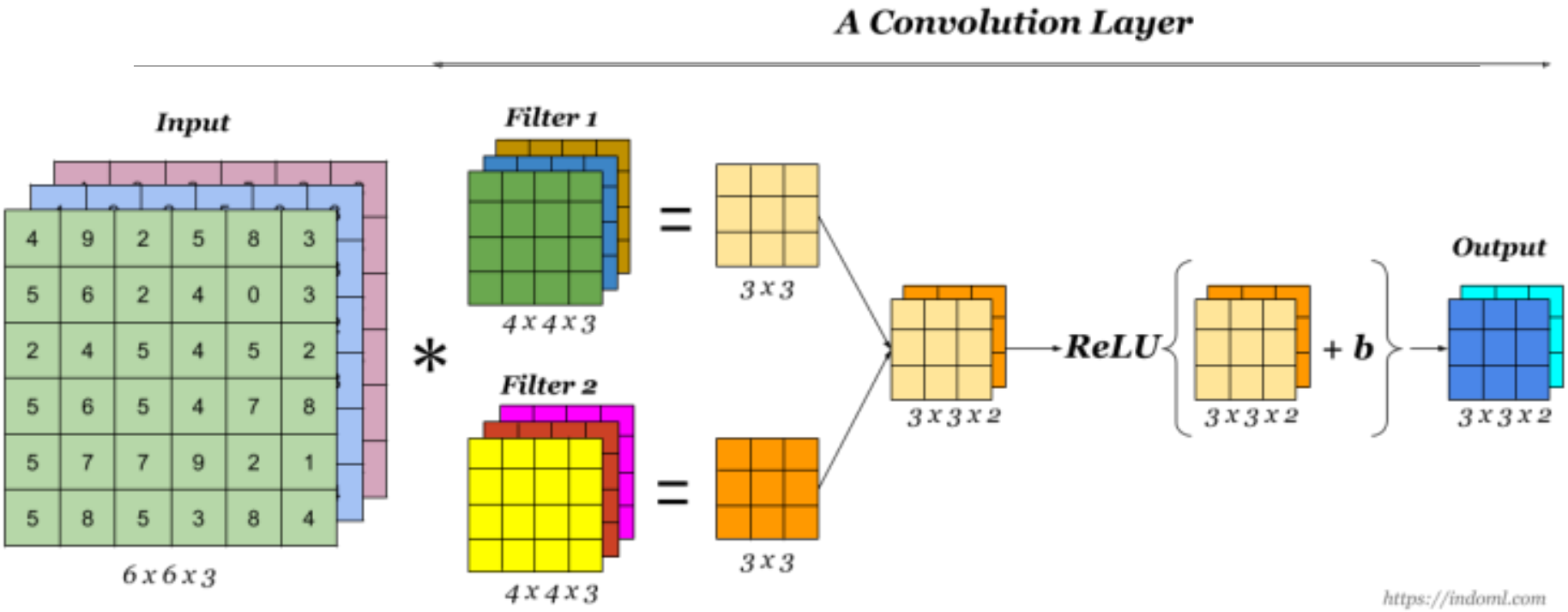
Ý nghĩa và lợi ích:

1. **Phát hiện nhiều đặc trưng khác nhau**: Mỗi bộ lọc chuyên biệt phát hiện một loại mẫu cụ thể trong dữ liệu.
2. **Tạo ra biểu diễn phong phú**: Tập hợp các feature map cung cấp biểu diễn đa chiều về dữ liệu đầu vào.
3. **Học phân cấp đặc trưng**: Các lớp tích chập sâu hơn có thể kết hợp các đặc trưng cơ bản từ các lớp trước để phát hiện các mẫu phức tạp hơn.
4. **Tạo cơ sở cho việc phân loại**: Các feature map cuối cùng chứa thông tin đủ để lớp fully connected có thể thực hiện phân loại chính xác.

2.4. Lớp tích chập

Lớp tích chập là một lớp trong CNN thực hiện các phép toán tích chập giữa dữ liệu đầu vào và các bộ lọc (filter/kernel) để tạo ra các bản đồ đặc trưng (feature map).

Đây là lớp chịu trách nhiệm chính trong việc học và trích xuất các đặc trưng từ dữ liệu.


A Convolution Layer
Input
4 9 2 5 8 3
5 6 2 4 0 3
2 4 5 4 5 2
5 6 5 4 7 8
5 7 7 9 2 1
5 8 5 3 8 4
6 x 6 x 3
*
Filter 1
4 x 4 x 3
=
3 x 3
Filter 2
4 x 4 x 3
=
3 x 3
3 x 3 x 2
ReLU
3 x 3 x 2
+ b
Output
3 x 3 x 2
https://indoml.com

2.5. Lớp pooling

Lớp Pooling là gì?

Lớp gộp (Pooling) là một lớp trong CNN có nhiệm vụ giảm kích thước (downsampling) các feature map, giúp:

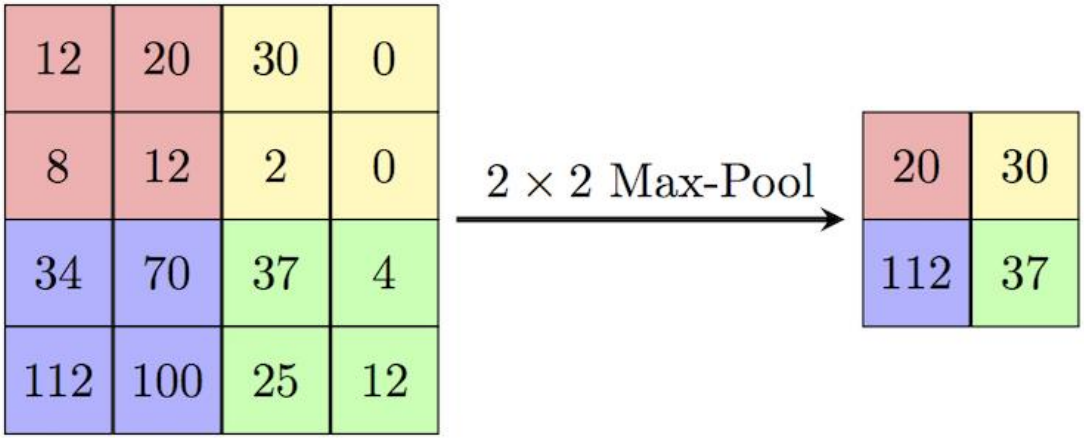
- Giảm số lượng tham số và tính toán trong mạng
- Kiểm soát overfitting
- Tăng tính bất biến với các biến đổi nhỏ trong dữ liệu đầu vào

Max Pooling

•**Nguyên lý**: Chọn giá trị lớn nhất trong một vùng

•**Cách hoạt động**: Chia feature map thành các vùng không chồng lấp, lấy giá trị max trong mỗi vùng

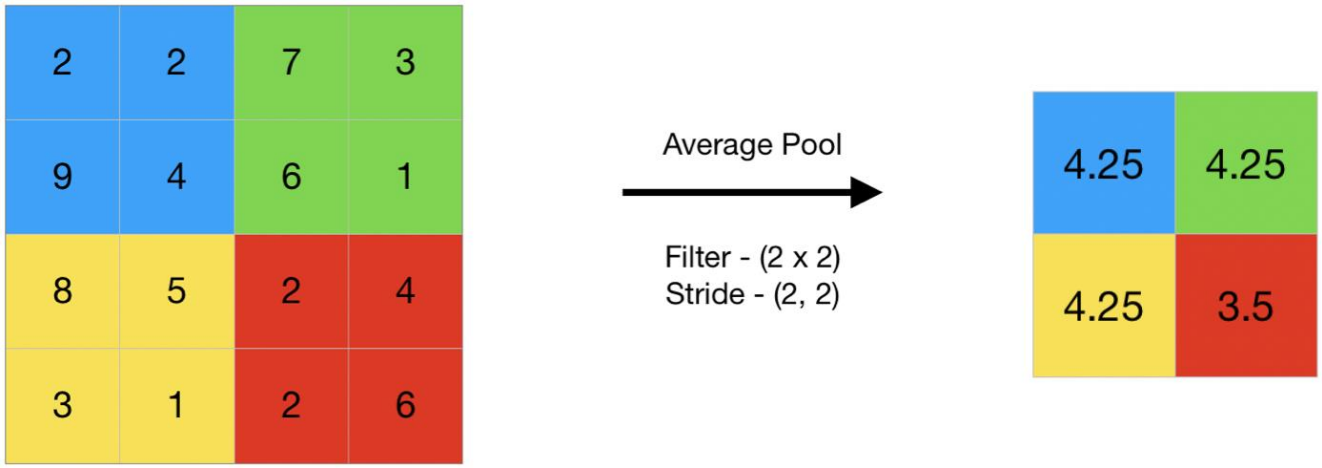
•**Ví dụ** với Max Pooling 2×2, stride=2



Ưu điểm: Giữ lại các đặc trưng nổi bật nhất, thường cho kết quả tốt trong nhận dạng hình ảnh

Average Pooling

- **Nguyên lý**: Tính giá trị trung bình trong một vùng
- **Cách hoạt động**: Tương tự Max Pooling nhưng lấy giá trị trung bình
- **Ví dụ** với Average Pooling 2×2, stride=2:

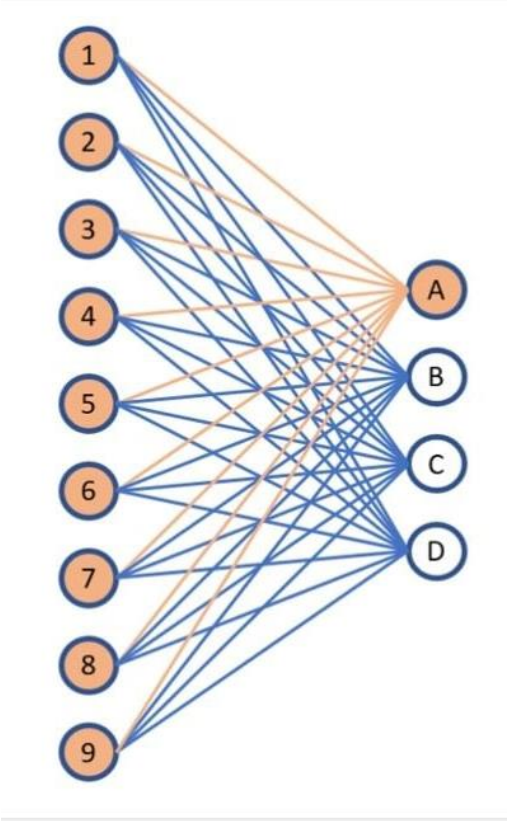


Ưu điểm: Giữ lại thông tin nền, có ích khi xử lý hình ảnh mịn, ảnh y tế

2.6. Lớp kết nối đầy đủ

Lớp kết nối đầy đủ (Fully Connected Layer) là một thành phần quan trọng trong kiến trúc mạng nơ-ron tích chập (CNN), thường được đặt ở các tầng cuối của mạng.

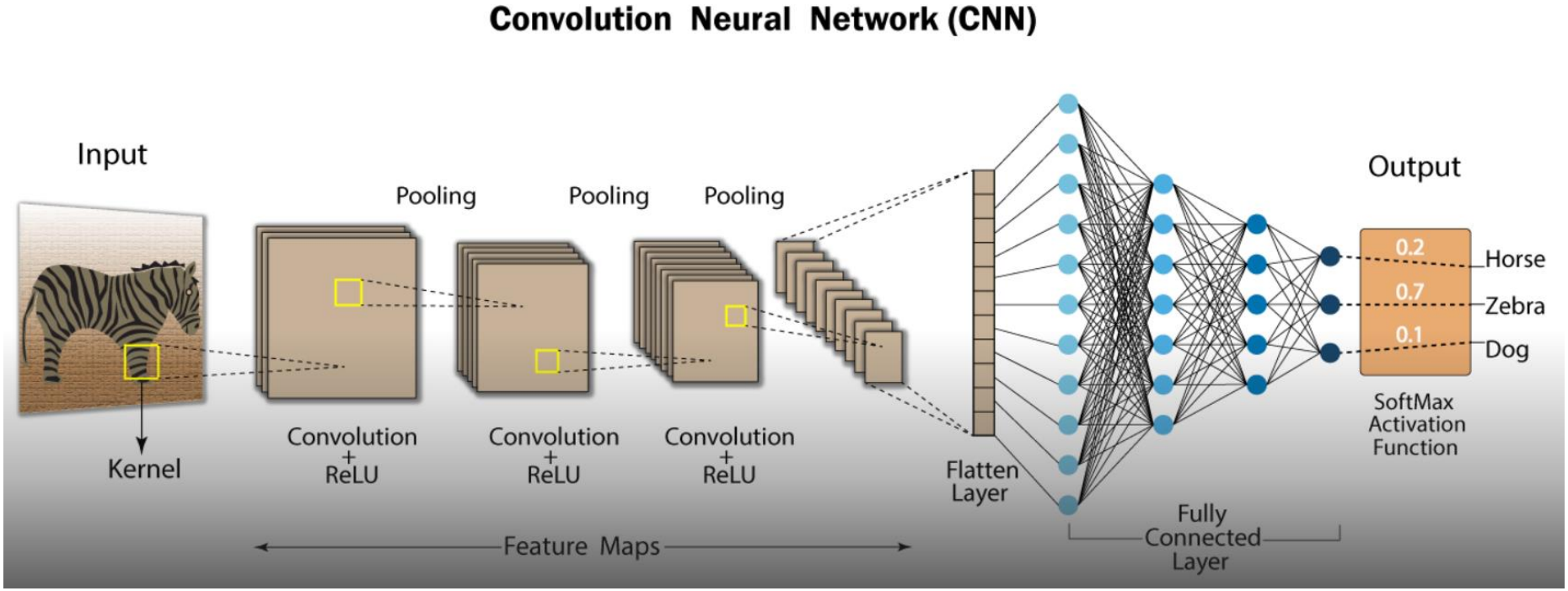
Lớp này kết nối mọi nơ-ron trong tầng hiện tại với mọi nơ-ron trong tầng trước đó, tương tự như trong mạng nơ-ron truyền thống (MLP).



2.6. Lớp kết nối đầy đủ

Trong CNN điển hình, cấu trúc thường là:

1. Nhiều lớp tích chập (Convolutional Layers) + lớp gộp (Pooling Layers) → trích xuất đặc trưng
2. Lớp làm phẳng (Flatten Layer) → chuyển đổi feature maps thành vector
3. **Lớp kết nối đầy đủ (Fully Connected Layers)** → phân loại/dự đoán


Convolution Neural Network (CNN)
Input
Pooling
Pooling
Pooling
Output
0.2
Horse
0.7
Zebra
0.1
Dog
Kernel
Convolution + ReLU
Convolution + ReLU
Convolution + ReLU
Flatten Layer
SoftMax Activation Function
Feature Maps
Fully Connected Layer

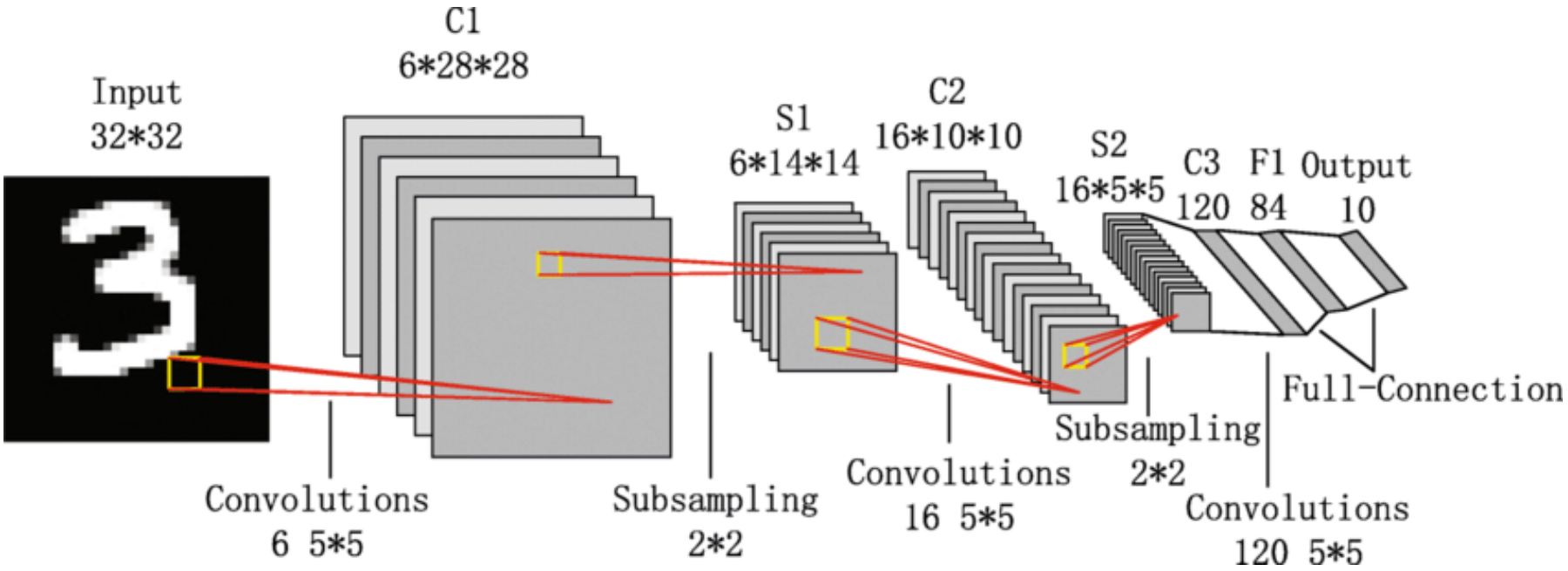
3. MỘT SỐ KIẾN TRÚC MẠNG CNN NỔI TIẾNG

LeNet

Người phát triển: Yann LeCun

Đặc điểm chính:

- Kiến trúc CNN đầu tiên, phát triển để nhận dạng chữ số viết tay
- 7 lớp (không tính đầu vào): 2 lớp tích chập, 2 lớp gộp (subsampling), 3 lớp kết nối đầy đủ
- Số tham số: khoảng 60,000
- Kích thước đầu vào: 32×32 pixel (ảnh xám)


Input
32*32
C1
6*28*28
S1
6*14*14
C2
16*10*10
S2
16*5*5
C3
120
F1
84
Output
10
Full-Connection
Convolutions
6 5*5
Subsampling
2*2
Convolutions
16 5*5
Subsampling
2*2
Convolutions
120 5*5

Cấu trúc:

1. **Đầu vào**: 32×32×1
2. **C1**: Lớp tích chập (6 bộ lọc 5×5, stride 1) → 28×28×6
3. **S2**: Lớp gộp trung bình (2×2, stride 2) → 14×14×6
4. **C3**: Lớp tích chập (16 bộ lọc 5×5, kết nối đặc biệt) → 10×10×16
5. **S4**: Lớp gộp trung bình (2×2, stride 2) → 5×5×16
6. **C5**: Lớp tích chập hoạt động như FC (120 bộ lọc 5×5) → 1×1×120
7. **F6**: Lớp kết nối đầy đủ (84 nơ-ron)
8. **Output**: Lớp kết nối đầy đủ (10 nơ-ron, cho 10 chữ số)

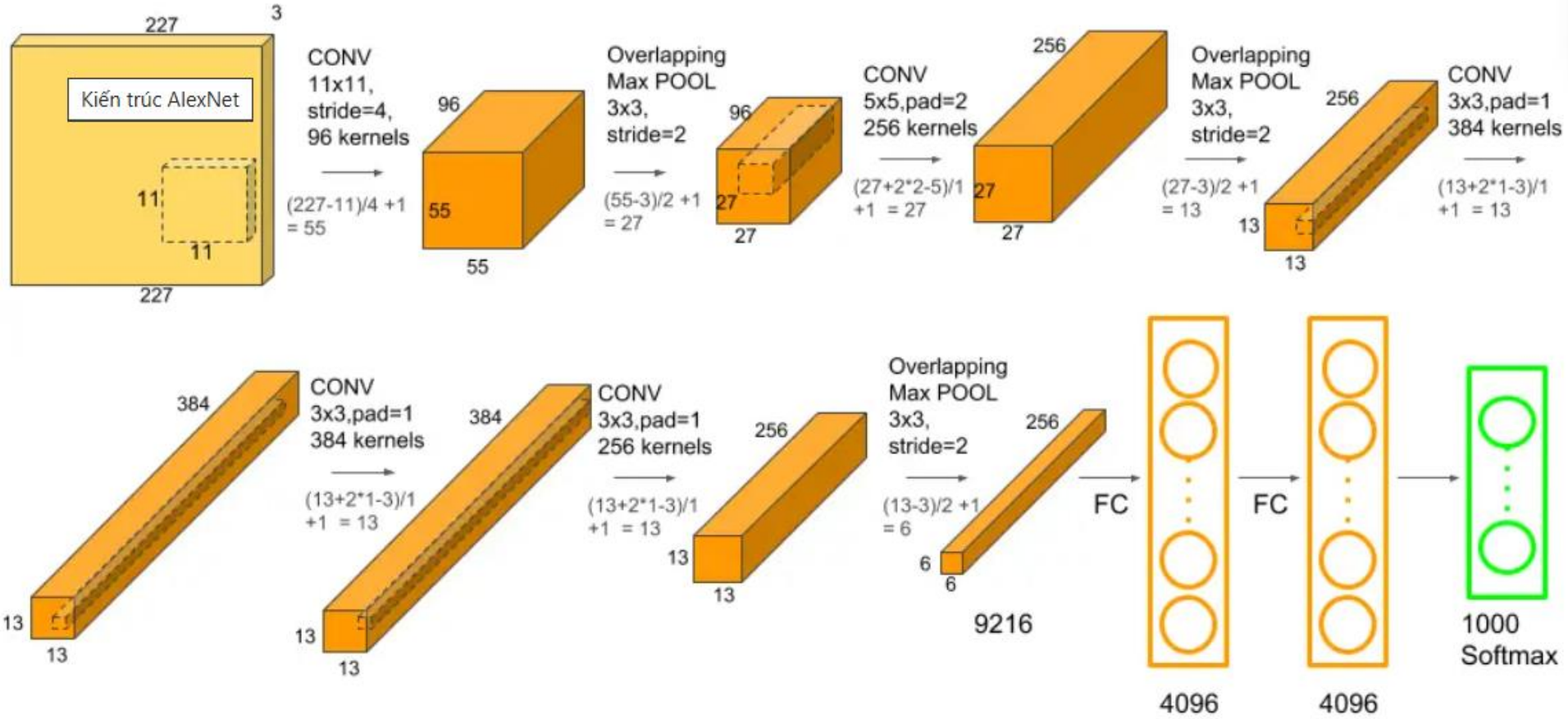
Tầm quan trọng: Đặt nền móng cho CNN hiện đại, chứng minh hiệu quả của tích chập trong xử lý hình ảnh.

3. MỘT SỐ KIẾN TRÚC MẠNG CNN NỔI TIẾNG

AlexNet (2012)

Người phát triển: Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton
Đặc điểm chính:

- Giành chiến thắng trong cuộc thi ImageNet 2012, giảm lỗi từ 26% xuống 15.3%
- Sâu hơn LeNet nhiều, sử dụng GPU để huấn luyện
- Giới thiệu ReLU, Local Response Normalization, Dropout
- Số tham số: khoảng 60 triệu
- Kích thước đầu vào: 227×227×3 (ảnh RGB)


Kiến trúc AlexNet
227
227
3
11
11
CONV
11x11,
stride=4,
96 kernels
(227-11)/4 +1
= 55
96
55
55
Overlapping
Max POOL
3x3,
stride=2
(55-3)/2 +1
= 27
96
27
27
CONV
5x5,pad=2
256 kernels
(27+2*2-5)/1
+1 = 27
256
27
27
Overlapping
Max POOL
3x3,
stride=2
(27-3)/2 +1
= 13
256
13
13
CONV
3x3,pad=1
384 kernels
(13+2*1-3)/1
+1 = 13
384
13
13
CONV
3x3,pad=1
384 kernels
(13+2*1-3)/1
+1 = 13
384
13
13
CONV
3x3,pad=1
256 kernels
(13+2*1-3)/1
+1 = 13
256
13
13
Overlapping
Max POOL
3x3,
stride=2
(13-3)/2 +1
= 6
256
6
6
9216
FC
4096
FC
4096
1000
Softmax

Cấu trúc:

1. **Đầu vào**: 227×227×3
2. **Conv1**: Lớp tích chập (96 bộ lọc 11×11, stride 4, ReLU) → 55×55×96
3. **Pool1**: Max Pooling (3×3, stride 2) → 27×27×96
4. **Conv2**: Lớp tích chập (256 bộ lọc 5×5, padding 2, ReLU) → 27×27×256
5. **Pool2**: Max Pooling (3×3, stride 2) → 13×13×256
6. **Conv3**: Lớp tích chập (384 bộ lọc 3×3, padding 1, ReLU) → 13×13×384
7. **Conv4**: Lớp tích chập (384 bộ lọc 3×3, padding 1, ReLU) → 13×13×384
8. **Conv5**: Lớp tích chập (256 bộ lọc 3×3, padding 1, ReLU) → 13×13×256
9. **Pool5**: Max Pooling (3×3, stride 2) → 6×6×256
10. **FC6**: Lớp kết nối đầy đủ (4096 nơ-ron, ReLU, Dropout)
11. **FC7**: Lớp kết nối đầy đủ (4096 nơ-ron, ReLU, Dropout)
12. **FC8**: Lớp kết nối đầy đủ (1000 nơ-ron, Softmax)

Tầm quan trọng: Tạo nên cuộc cách mạng trong thị giác máy tính, khởi đầu cho kỷ nguyên deep learning trong xử lý hình ảnh.

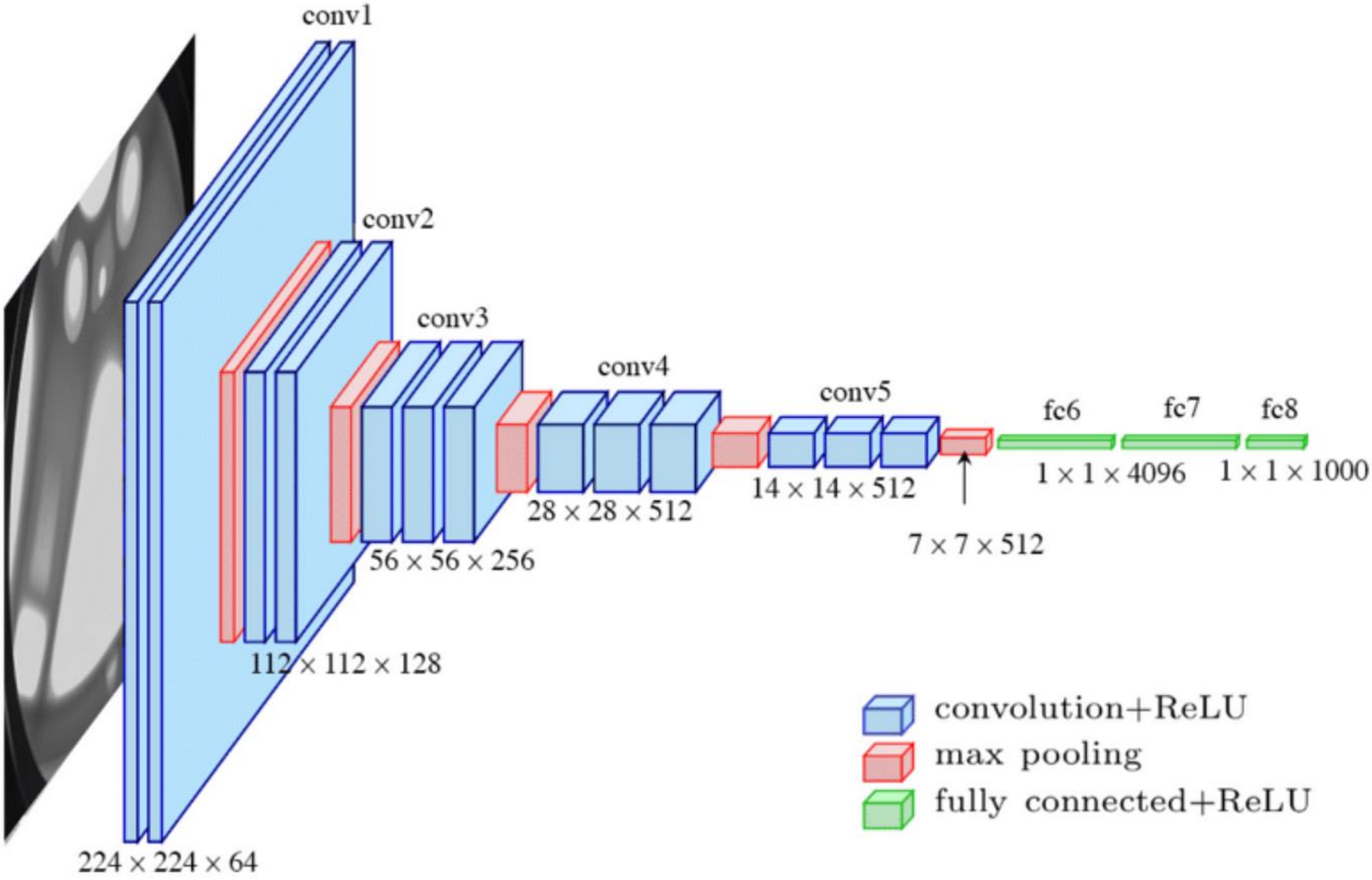
3. MỘT SỐ KIẾN TRÚC MẠNG CNN NỔI TIẾNG

VGG-16 (2014)

Người phát triển: Visual Geometry Group (Đại học Oxford)

Đặc điểm chính:

- Kiến trúc đơn giản, sâu, đồng nhất
- Chỉ sử dụng bộ lọc 3×3, stride 1, padding 1
- Max pooling 2×2, stride 2
- Số tham số: khoảng 138 triệu
- Kích thước đầu vào: 224×224×3


conv1
conv2
conv3
conv4
conv5
fc6
fc7
fc8
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
convolution+ReLU
max pooling
fully connected+ReLU

Cấu trúc:

1. **Đầu vào**: 224×224×3
2. **Block 1**: 2 lớp tích chập (64 bộ lọc 3×3, ReLU) → Max pool → 112×112×64
3. **Block 2**: 2 lớp tích chập (128 bộ lọc 3×3, ReLU) → Max pool → 56×56×128
4. **Block 3**: 3 lớp tích chập (256 bộ lọc 3×3, ReLU) → Max pool → 28×28×256
5. **Block 4**: 3 lớp tích chập (512 bộ lọc 3×3, ReLU) → Max pool → 14×14×512
6. **Block 5**: 3 lớp tích chập (512 bộ lọc 3×3, ReLU) → Max pool → 7×7×512
7. **FC1**: Lớp kết nối đầy đủ (4096 nơ-ron, ReLU, Dropout)
8. **FC2**: Lớp kết nối đầy đủ (4096 nơ-ron, ReLU, Dropout)
9. **FC3**: Lớp kết nối đầy đủ (1000 nơ-ron, Softmax)

Tầm quan trọng: Chứng minh tầm quan trọng của độ sâu mạng, tạo nên kiến trúc đơn giản nhưng hiệu quả cao.

4. Huấn luyện mạng CNN

Bước 1, Chuẩn bị dữ liệu

Thu thập dữ liệu: Để huấn luyện một CNN, chúng ta cần thu thập một tập ***dữ liệu lớn và đa dạng l***iên quan đến nhiệm vụ. Đối với phân loại hình ảnh, điều này có thể bao gồm hàng chục/trăm nghìn hình ảnh có nhãn thuộc nhiều loại khác nhau. Có lượng dữ liệu đủ lớn là điều kiện tiên quyết để xây dựng mô hình CNN sâu có hiệu quả.

4. Huấn luyện mạng CNN

- **Tiền xử lý dữ liệu**: Để huấn luyện CNN một cách hiệu quả, chúng ta cần thực hiện một số bước sau để tiền xử lý dữ liệu.
  - **Chuẩn hóa**: Chuyển đổi giá trị của các pixel về một phạm vi nhất định, hai khoảng chuẩn hóa phổ biến nhất là [0, 1] hoặc [-1, 1]. Việc chuẩn hóa này sẽ giúp cải thiện sự hội tụ trong quá trình huấn luyện.
  - **Tăng cường dữ liệu**: Áp dụng các phép biến đổi như xoay, lật, thay đổi kích thước và dịch chuyển để tăng kích thước tập dữ liệu một cách nhân tạo, từ đó giúp mô hình có thể học một cách tổng quát hóa.
  - **Chia dữ liệu**: Chia tập dữ liệu thành các tập huấn luyện, xác thực và kiểm tra. Tập huấn luyện được dùng để học, tập xác thực để điều chỉnh siêu tham số chẳng hạn như tốc độ học (learning rate), kích thước lô (batch size), và tập kiểm tra để đánh giá hiệu suất cuối cùng.

4. Huấn luyện mạng CNN

Bước 2, Thiết kế kiến trúc mô hình CNN:

Lựa chọn lớp:

- ***Chọn số lượng và loại lớp*** (tích chập, pooling, kết nối đầy đủ) dựa trên độ phức tạp của nhiệm vụ và tính chất của dữ liệu.
- Đối với lớp tích chập, cần xác định kích thước bộ lọc, stride, padding, và số lượng bộ lọc của mỗi lớp. Ngoài ra, cần lựa chọn hàm kích hoạt cho các lớp tích chập. Thông thường, hàm kích hoạt cho lớp tích chập là các hàm ReLU, các biến thể của ReLU, Swish, Mish.
- Đối với lớp pooling, cần xác định loại pooling mong muốn, có thể là max pooling hoặc average pooling hoặc một phương pháp mới khác.
- Đối với lớp kết nối đầy đủ, cần xác định số lượng lớp ẩn, số lượng neuron trong mỗi lớp, và hàm kích hoạt. Đối với các lớp ẩn, hàm kích hoạt phổ biến là các hàm ReLU, các biến thể của ReLU, Swish, Mish. Đối với lớp đầu ra, hàm kích hoạt thường dùng là Sigmoid, Tanh.

Điều chỉnh (regularization): Áp dụng các kỹ thuật như dropout và L2-regularization để ngăn chặn overfitting.

4. Huấn luyện mạng CNN
Bước 3, Xác thực và điều chuyển tham số

•**Xác thực**: Đánh giá mô hình trên tập xác thực (validation set) để theo dõi hiệu suất và phát hiện overfitting. Các chỉ số như độ chính xác, precision, recall, và F1-score thường được sử dụng.

•**Điều chỉnh siêu tham số**: Điều chỉnh các siêu tham số như tốc độ học, kích thước lô, và kiến trúc mạng dựa trên hiệu suất xác thực. Có thể sử dụng các kỹ thuật như tìm kiếm lưới và tìm kiếm ngẫu nhiên để hỗ trợ việc điều chỉnh các siêu tham số.

4. Huấn luyện mạng CNN
Bước 4, Kiểm tra và đánh giá

•Khi mô hình đã được huấn luyện và xác thực, đánh giá hiệu suất của nó trên tập kiểm tra để đánh giá khả năng của mô hình trên dữ liệu mới.

•Sử dụng các chỉ số đo lường phù hợp với bài toán.

4. Huấn luyện mạng CNN
Bước 5, Triển khai

•Lưu kiến trúc và các tham số của mô hình đã được huấn luyện để triển khai hoặc sử dụng trong tương lai.

•Lựa chọn môi trường và phương pháp để deploy mô hình đã huấn luyện trong thực tế.

5. Ứng dụng CNN vào bài toán phân loại ảnh

Phân loại hình ảnh là một nhiệm vụ cơ bản trong lĩnh vực thị giác máy tính, liên quan đến việc gán một nhãn cụ thể cho một hình ảnh từ một tập hợp các danh mục được xác định trước.

Quá trình này cho phép máy móc diễn giải và phân loại dữ liệu hình ảnh, giống như cách con người nhận biết và phân loại các đối tượng.

5. Ứng dụng CNN vào bài toán phân loại ảnh

Sự phát triển của các thuật toán phân loại hình ảnh được thúc đẩy bởi những tiến bộ đáng kể trong học máy và học sâu, đặc biệt là thông qua việc sử dụng mạng neuron tích chập (CNNs). Các mô hình CNN được thiết kế để tự động học các đặc trưng từ các hình ảnh đầu vào, do vậy các mô hình CNN đặc biệt hiệu quả cho bài toán phân loại hình ảnh.

5. Ứng dụng CNN vào bài toán phân loại ảnh

Phân loại hình ảnh có rất nhiều ứng dụng thực tiễn, ví dụ như xe tự lái, hệ thống nhận diện sinh trắc học, chẩn đoán y tế, ... Khi công nghệ tiếp tục phát triển, phân loại hình ảnh được dự đoán sẽ trở thành một phần không thể thiếu của nhiều ngành công nghiệp, nâng cao khả năng của máy móc trong việc hiểu và tương tác với thế giới xung quanh chúng.

5. Ứng dụng CNN vào bài toán phân loại ảnh

Một số vấn đề có thể ảnh hưởng đến độ chính xác và hiệu quả của các mô hình phân loại hình ảnh, bao gồm:

Sự biến đổi trong hình ảnh: Hình ảnh có thể thay đổi nhiều do sự khác biệt về ánh sáng, góc độ, tỷ lệ và nền. Sự biến đổi này có thể khiến các mô hình khó xác định đối tượng một cách nhất quán trong các điều kiện khác nhau.

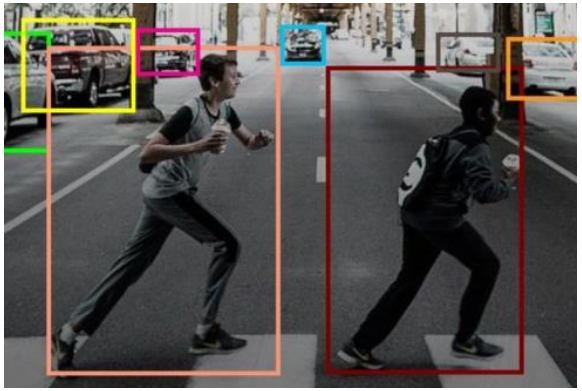
Che khuất và lộn xộn: Các đối tượng trong hình ảnh có thể bị che khuất một phần bởi các đối tượng khác (che khuất) hoặc bị bao quanh bởi một nền lộn xộn, điều này có thể gây nhầm lẫn cho mô hình và dẫn đến phân loại sai.

Biến đổi trong cùng một lớp và sự tương đồng giữa các lớp: Có thể có sự khác biệt đáng kể trong một lớp duy nhất (ví dụ, các giống chó khác nhau) và sự tương đồng...

5. Ứng dụng CNN vào bài toán phân loại ảnh

Một số vấn đề có thể ảnh hưởng đến độ chính xác và hiệu quả của các mô hình phân loại hình ảnh, bao gồm:

Sự biến đổi trong hình ảnh: Hình ảnh có thể thay đổi nhiều do sự khác biệt về ánh sáng, góc độ, tỷ lệ và nền. Sự biến đổi này có thể khiến các mô hình khó xác định đối tượng một cách nhất quán trong các điều kiện khác nhau.



5. Ứng dụng CNN vào bài toán phân loại ảnh

Che khuất và lộn xộn:

Các đối tượng trong hình ảnh có thể bị che khuất một phần bởi các đối tượng khác (che khuất) hoặc bị bao quanh bởi một nền lộn xộn, điều này có thể gây nhầm lẫn cho mô hình và dẫn đến phân loại sai.

5. Ứng dụng CNN vào bài toán phân loại ảnh

Biến đổi trong cùng một lớp và sự tương đồng giữa các lớp: Có thể có sự khác biệt đáng kể trong một lớp duy nhất (ví dụ, các giống chó khác nhau) và sự tương đồng...

5. Ứng dụng CNN vào bài toán phân loại ảnh

Biến đổi trong cùng một lớp và sự tương đồng giữa các lớp: Có thể có sự khác biệt đáng kể trong một lớp duy nhất (ví dụ, các giống chó khác nhau) và sự tương đồng tương đồng giữa các lớp khác nhau (ví dụ, mèo và chó), điều này gây khó khăn cho các mô hình trong việc phân loại hình ảnh.

5. Ứng dụng CNN vào bài toán phân loại ảnh

•**Chất lượng và số lượng dữ liệu**: Các bộ dữ liệu được chất lượng cao đóng vai trò quan trọng để huấn luyện các mô hình hiệu quả. Tuy nhiên, việc thu thập các bộ dữ liệu lớn, đa dạng và được gán nhãn chính xác rất tốn kém về tài nguyên và thời gian.

•**Sự mất cân bằng lớp**: Trong nhiều bộ dữ liệu, một số lớp có số lượng hình ảnh nhiều, một số lớp lại có quá ít hình ảnh. Sự mất cân bằng này có thể làm cho mô hình thiên vị về phía các lớp phổ biến hơn, dẫn đến hiệu suất kém trên các lớp thiểu số.

Thực hành

Thu thập và tiền xử lý dữ liệu:

- Giống như bài tập trước, trong bài tập này chúng ta vẫn sử dụng bộ dữ liệu MNIST. Bộ dữ liệu này có thể dễ dàng tải xuống bằng cách sử dụng TensorFlow.
- Dữ liệu cần được chuẩn hóa giống như các ví dụ trước.

Thu thập và tiền xử lý dữ liệu:

```
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.utils import to_categorical

# Load the MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# Preprocess the data
x_train = x_train.reshape((x_train.shape[0], 28, 28, 1)).astype('float32') / 255
x_test = x_test.reshape((x_test.shape[0], 28, 28, 1)).astype('float32') / 255
y_train = to_categorical(y_train, 10) # Convert labels to one-hot encoding
y_test = to_categorical(y_test, 10)
```

Thiết kế CNN:

- Trong ví dụ này, chúng ta sẽ sử dụng một mạng CNN có kiến trúc đơn giản, bao gồm:
- Hai lớp tích chập, trong đó: lớp tích chập thứ nhất sử dụng 32 bộ lọc kích thước 3x3, lớp tích chập thứ hai sử dụng 64 bộ lọc kích thước 3x3. Cả hai lớp tích chập đều sử dụng stride = 2, padding = 0, và hàm kích hoạt ReLU.
- Theo sau mỗi lớp tích chập là một lớp max pooling.
- Hai lớp kết nối đầy đủ, trong đó: lớp kết nối đầy đủ thứ nhất có 128 neuron, và sử dụng hàm kích hoạt ReLU; lớp kết nối đầy đủ thứ hai (lớp đầu ra) có 10 neuron, tương ứng với 10 lớp cần phân loại, và sử dụng hàm kích hoạt Softmax. Đầu ra của CNN chính là xác suất để một hình ảnh thuộc về một trong 10 lớp từ 0 đến 9.

Thiết kế CNN:

```
# Build the CNN model
model = Sequential([
    Conv2D(32, kernel_size = (3, 3), activation = 'relu', input_shape = (28, 28, 1)),
    MaxPooling2D(pool_size = (2, 2)),
    Conv2D(64, kernel_size = (3, 3), activation = 'relu'),
    MaxPooling2D(pool_size = (2, 2)),
    Flatten(),
    Dense(128, activation = 'relu'),
    Dense(10, activation = 'softmax')
])
```

Huấn luyện CNN:

Huấn luyện CNN:

- Sử dụng 10% dữ liệu huấn luyện để làm dữ liệu validation.
- Đánh giá mô hình sau huấn luyện.

```
# Compile the model
model.compile(optimizer = 'adam', loss = 'categorical_crossentropy',
              metrics = ['accuracy'])

# Train the model
model.fit(x_train, y_train, validation_split = 0.1, epochs = 10, batch_size = 32)

# Evaluate the model
test_loss, test_accuracy = model.evaluate(x_test, y_test)
print(f'Test accuracy: {test_accuracy:.4f}')

# Save the model (optional)
model.save('mnist_cnn_model.h5')
```

6. Ứng dụng CNN vào bài toán phát hiện đối tượng

Phát hiện đối tượng (object detection) là một bài toán quan trọng trong thị giác máy tính, liên quan đến việc xác định và định vị các đối tượng trong một hình ảnh.

Khác với phân loại hình ảnh, chỉ xác định sự hiện diện của một đối tượng, phát hiện đối tượng cung cấp cả nhãn lớp và vị trí của mỗi đối tượng bằng cách vẽ các hộp giới hạn (bounding box) xung quanh chúng.

Khả năng này làm cho phát hiện đối tượng đặc biệt có giá trị trong các ứng dụng như xe tự lái, giám sát và robot.

6. Ứng dụng CNN vào bài toán phát hiện đối tượng

Quá trình phát hiện đối tượng sử dụng CNN thường bao gồm một số bước chính như sau:

- **Trích xuất đặc trưng (feature extraction)**: CNN được sử dụng để trích xuất các feature map từ hình ảnh đầu vào. Các feature map này ghi nhận các đặc điểm thị giác cơ bản, như cạnh, kết cấu và hình dạng, ở các mức độ trừu tượng khác nhau.

- **Đề xuất vùng (region proposal)**: Các thuật toán region proposal để xuất các vùng trong hình ảnh có khả năng chứa đối tượng. Thuật toán region proposal có thể sử dụng các thuật toán truyền thống như tìm kiếm chọn lọc hoặc các phương pháp hiện đại hơn như mạng đề xuất vùng (Region Proposal Network RPN).

- **Phân loại và định vị**: Đối với mỗi vùng được đề xuất ở bước đề xuất vùng, CNN dự đoán lớp đối tượng và tinh chỉnh tọa độ của bounding box. Bước này bao gồm cả hai bài toán phân lớp (xác định loại đối tượng) và hồi quy (dự đoán hộp giới hạn).

- **Hậu xử lý (postprocess)**: Các kỹ thuật như Non Maximum Suppression (NMS) được áp dụng để loại bỏ các bounding box dư thừa và đảm bảo rằng mỗi đối tượng chỉ được phát hiện một lần.

6. Ứng dụng CNN vào bài toán phát hiện đối tượng

Một số thuật toán phát hiện đối tượng dựa trên CNN nổi tiếng là:

- **R-CNN**: là phương pháp tiên phong trong phát hiện đối tượng bằng CNN. R-CNN sử dụng tìm kiếm chọn lọc cho thuật toán đề xuất vùng và CNN để phân loại và định vị đối tượng.
- **Fast R-CNN và Faster R-CNN**: Các mô hình này là mô hình cải tiến của R-CNN bằng cách tích hợp các bước đề xuất vùng và phân loại. Sự cải tiến này giúp giảm đáng kể thời gian thực thi của thuật toán.
- **YOLO (You Only Look Once)**: là thuật toán phát hiện đối tượng thời gian thực bằng cách chia hình ảnh thành một lưới và dự đoán bounding box và phân lớp dựa trên lưới đã chia.