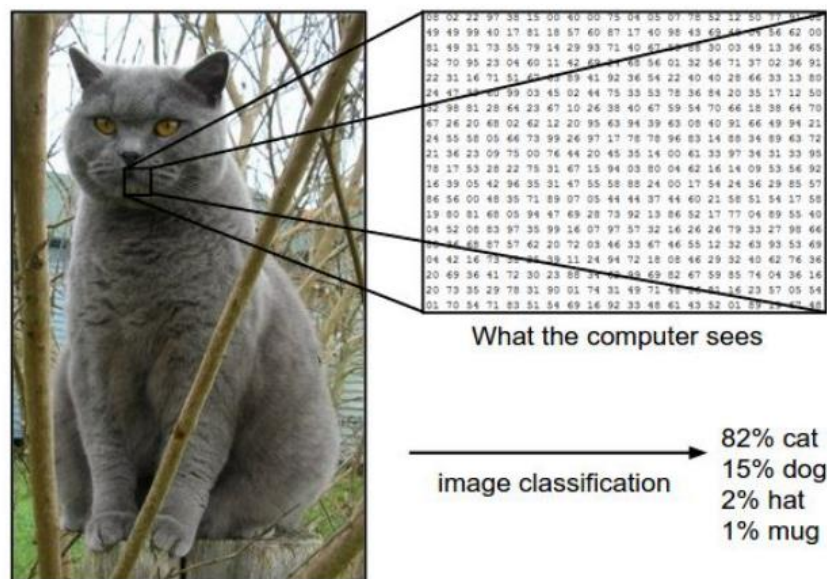


Báo cáo tuần 5

1. Image Classification with CNNs

1. Image Classification

Định nghĩa: Là một bài toán trong Computer Vision dùng để phân loại hoặc gắn nhãn một hình ảnh dựa trên nội dung quan sát được



Cách hoạt động:

1. **Input:** ảnh (RGB).
2. **Feature Extraction:** Học các đặc trưng
3. **Classification:** Phân loại thành các class.

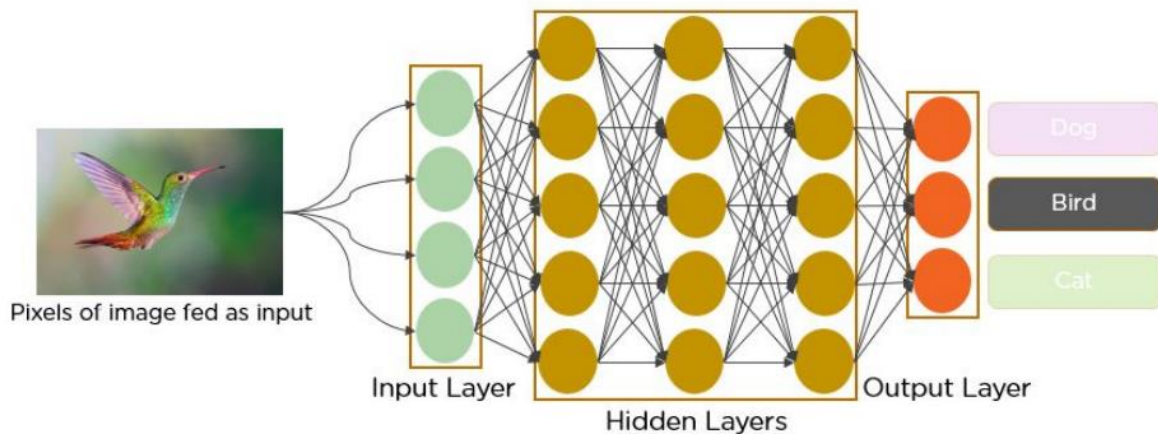


Các loại Metrics đánh giá:

- Accuracy
- Precision
- Recall
- F1-Score
- Confusion Matrix

2. Image Classification with Neuron Network

Nếu ta xem ảnh là vector dài $N = H \cdot W \cdot CN = H$, thì có thể đưa vào một **fully-connected neural network**.



Vấn đề Overfitting

- **Overfitting** xảy ra khi mạng nơ-ron học thuộc lòng dữ liệu thay vì học đặc trưng tổng quát.

- Nguyên nhân chính: số lượng tham số (weights) quá lớn so với số lượng dữ liệu → mạng có thể ghi nhớ chi tiết thay vì học trừu tượng.
- Regularization giúp giảm overfitting, nhưng không phải cách duy nhất hay tối ưu nhất.

Các vấn đề khác:

1. Mất thông tin không gian.
2. Coi pixel tương đương nhau.
3. Gradient khó học, dễ vanishing.
4. Inductive bias kém.
5. Không mở rộng được cho ảnh lớn/video.
6. Generalization kém với dataset nhỏ.

Ý tưởng “Structure” – Tái sử dụng trọng số:

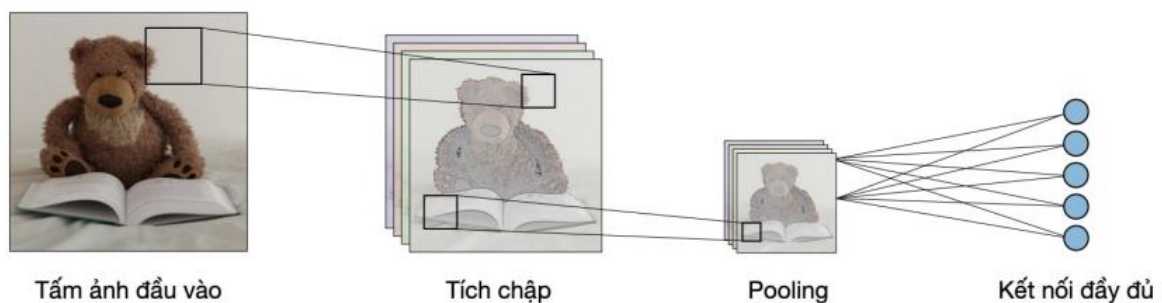
- Thay vì để mỗi vùng ảnh học **riêng một tập trọng số**, ta có thể **tái sử dụng cùng một trọng số ở nhiều vị trí khác nhau**.
- Điều này giúp:
 - Giảm số lượng tham số → giảm tỉ lệ weight/data → ít overfitting hơn.

- Vẫn giữ được khả năng biểu diễn mạnh mẽ của mô hình.
- Làm mô hình nhỏ gọn, dễ lưu trữ.
- Đây chính là **trọng tâm của Convolutional Neural Network (CNN)**.

2.Convolution Neural Network

1 Tổng quan

Kiến trúc truyền thống của một mạng CNN – Mạng neural tích chập (Convolutional neural networks), còn được biết đến với tên CNNs, là một dạng mạng neural được cấu thành bởi các tầng sau:



2. Convolution layer

2.1 Phép tính Convolution

- Convolution (ký hiệu \otimes) là một phép toán trên **ma trận dữ liệu (ảnh X)** và **kernel/filter W** (ma trận nhỏ).

- Kết quả là một **ma trận đặc trưng (feature map)**:
- Trong xử lý ảnh, **X** là ảnh gốc (ma trận pixel), **W** là kernel (ví dụ: 3×3 , 5×5).
- Kernel được quét (slide) qua ảnh, tại mỗi vị trí tính **tích từng phần tử và cộng tổng**.

1 _{x1}	1 _{x0}	1 _{x1}	0	0
0 _{x0}	1 _{x1}	1 _{x0}	1	0
0 _{x1}	0 _{x0}	1 _{x1}	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

Convolved
Feature

2.2. Padding

- Khi kernel quét, các pixel biên thường bị bỏ qua \rightarrow Y nhỏ hơn X.
- Giải pháp: thêm viền 0 quanh ảnh (zero padding).

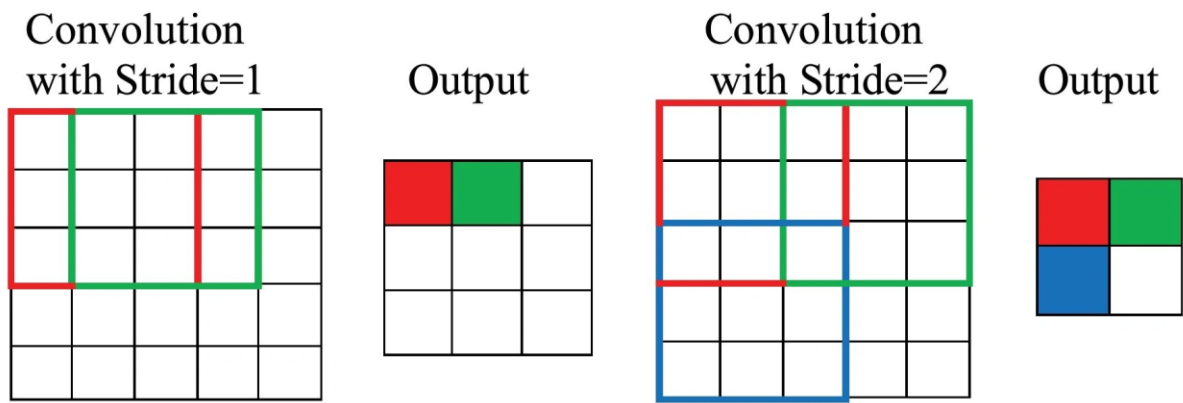
- Mục đích:
 - Giữ nguyên kích thước output.
 - Cho kernel xử lý được cả biên ảnh.

The image shows a 7x7 grid of pixels. A 3x3 kernel is highlighted with an orange border, covering the top-left corner. A single pixel at the center of the kernel (row 2, column 2) is highlighted with a red border. The grid contains the following values:

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

2.3. Stride

- Bình thường stride = 1: kernel di chuyển từng pixel.
- Nếu stride = 2: kernel nhảy 2 pixel/lần → output nhỏ hơn nhiều.



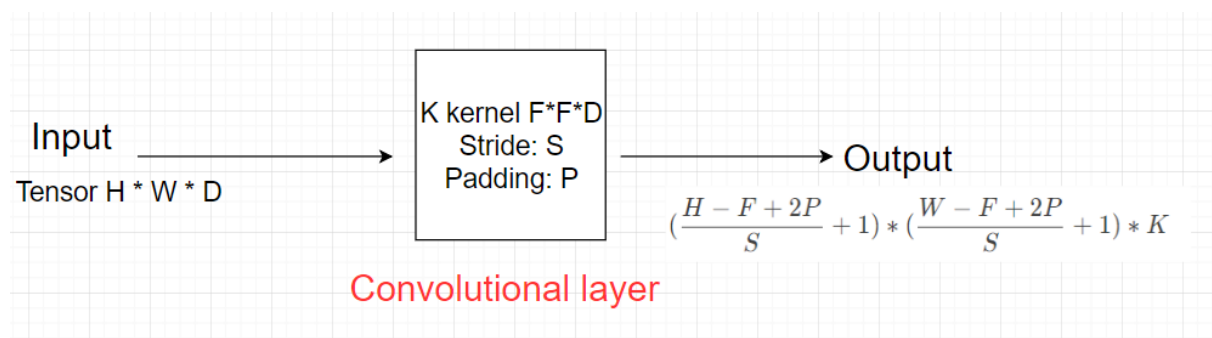
- Công thức tính kích thước output:

$$H_{out} = \frac{H_{in} - K + 2P}{S} + 1, \quad W_{out} = \frac{W_{in} - K + 2P}{S} + 1$$

Trong đó:

- H_{in} , W_{in} : chiều cao, chiều rộng ảnh đầu vào.
- K : kích thước kernel.
- P : padding.
- S : stride.

2.4. Tổng quát

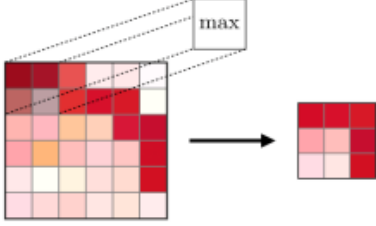
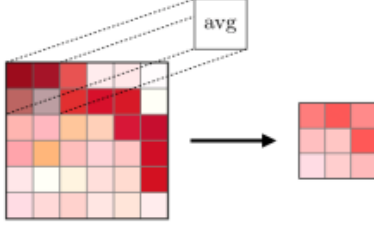


3.Pooling layer

Dùng để **giảm kích thước dữ liệu**, giữ lại đặc trưng quan trọng.

Hai loại chính:

- **Max Pooling**: chọn giá trị lớn nhất trong vùng $K \times K$.
- **Average Pooling**: lấy trung bình.

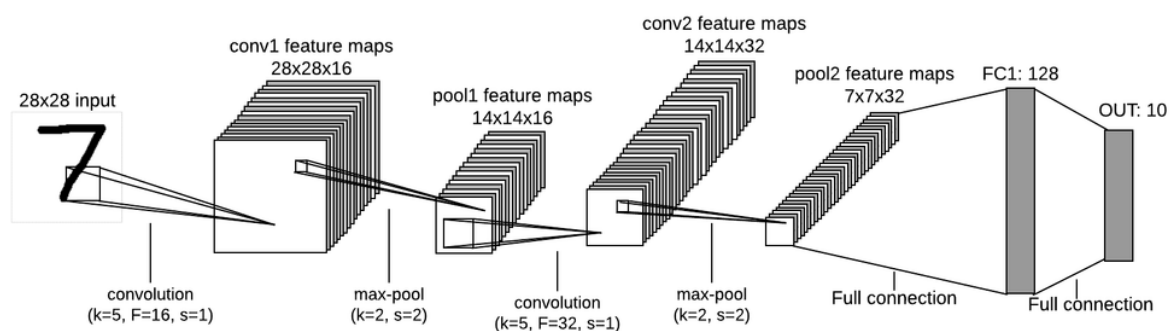
	Max pooling	Average pooling
Kiểu	Từng phép pooling chọn giá trị lớn nhất trong khu vực mà nó đang được áp dụng	Từng phép pooling tính trung bình các giá trị trong khu vực mà nó đang được áp dụng
Minh họa		
Nhận xét	<ul style="list-style-type: none">- Bảo toàn các đặc trưng đã phát hiện- Được sử dụng thường xuyên	<ul style="list-style-type: none">- Giảm kích thước feature map- Được sử dụng trong mạng LeNet

Ý nghĩa:

- Giảm số phép tính → mô hình nhanh hơn.
- Cho phép học các đặc trưng ở vùng lớn hơn.

4. Fully connected layer

Tầng kết nối đầy đủ (FC) nhận đầu vào là các dữ liệu đã được làm phẳng, mà mỗi đầu vào đó được kết nối đến tất cả neuron. Trong mô hình mạng CNNs, các tầng kết nối đầy đủ thường được tìm thấy ở cuối mạng và được dùng để tối ưu hóa mục tiêu của mạng ví dụ như độ chính xác của lớp.



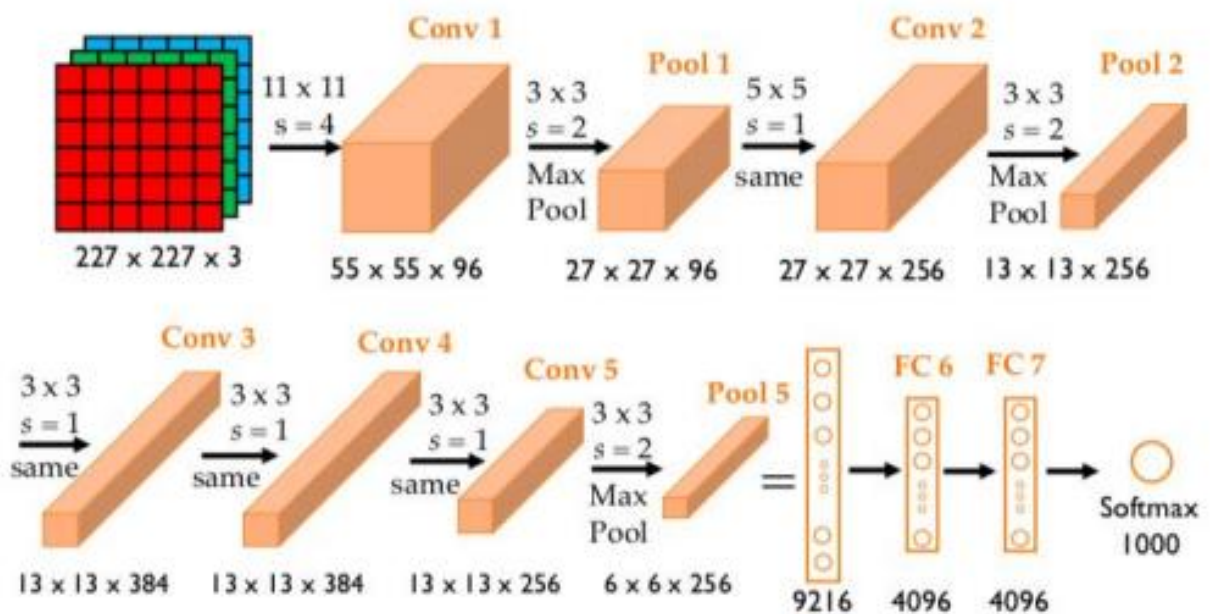
3. Các model CNN nổi tiếng

1. Alexnet(2012)

Kiến trúc :

- Input: RGB (227×227 trong bản gốc, thường preprocess thành 224×224).
- Conv1: 11×11, 96 filters, stride 4 → ReLU → LRN → MaxPool.
- Conv2: 5×5, 256 filters → ReLU → LRN → MaxPool.
- Conv3: 3×3, 384 filters → ReLU.

- Conv4: 3×3 , 384 filters \rightarrow ReLU.
- Conv5: 3×3 , 256 filters \rightarrow ReLU \rightarrow MaxPool.
- FC1: 4096 neurons \rightarrow ReLU \rightarrow Dropout.
- FC2: 4096 neurons \rightarrow ReLU \rightarrow Dropout.
- FC3: 1000-way softmax.



Vì sao thành công

- Dữ liệu ImageNet lớn + GPU để huấn luyện mô hình sâu.
- AlexNet là minh chứng thực nghiệm rằng mạng sâu có thể vượt xa các phương pháp cổ điển.

2. VGG (2014)

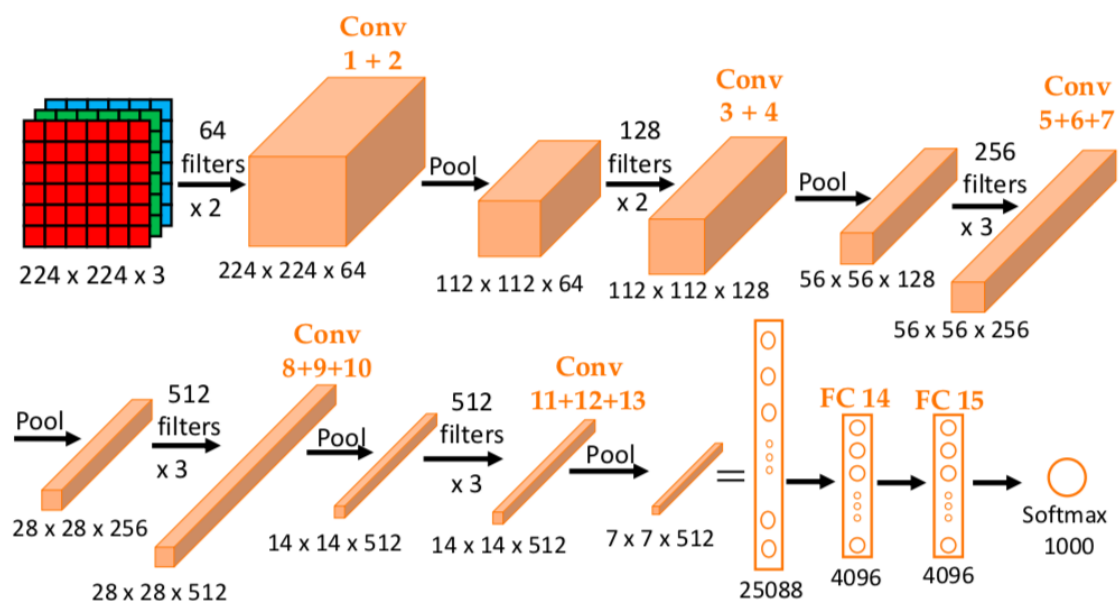
Thay vì dùng kernel lớn, VGG dùng **xếp chồng nhiều conv 3×3** — dễ phân tích và mở rộng, cho phép tăng chiều sâu (16 hoặc 19 layers) và cải thiện hiệu năng

Có nhiều phiên bản: VGG11, VGG13, VGG16, VGG19

Cấu hình điển hình

- **VGG16**: 13 conv (3×3) + 3 FC (tổng ~138 triệu tham số).
- **VGG19**: 16 conv (3×3) + 3 FC.
Mỗi block conv thường là (Conv3×3 × 2 or ×3) → ReLU → MaxPool.

□ Minh họa kiến trúc VGG16:



Lợi & Hạn

- **Lợi:** kiến trúc đơn giản, dễ triển khai, biểu diễn feature tốt → thường dùng làm feature extractor cho transfer learning.
- **Hạn:** tham số lớn (FC gây chi phí lớn) → tốn bộ nhớ và compute; khó tối ưu cho thiết bị giới hạn.

3. ResNet (2015)

Khi CNN ngày càng **sâu hơn** (tăng số lớp), ta kỳ vọng mô hình học được đặc trưng tốt hơn → tăng độ chính xác.

Nhưng thực tế:

- **Vanishing Gradient:** Gradient quá nhỏ hoặc quá lớn khiến mô hình khó train.
- **Degradation Problem:** Khi tăng số lớp, training error lại **tăng lên** (ngược kỳ vọng), nghĩa là mô hình học kém đi.

=> **ResNet** ra đời để giải quyết vấn đề này.

Kiến trúc:

ResNet-18/34 (Plain Residual Block)

- Dùng **2 Conv 3×3** trong mỗi residual block.
- Thường gọi là **Basic Block**.

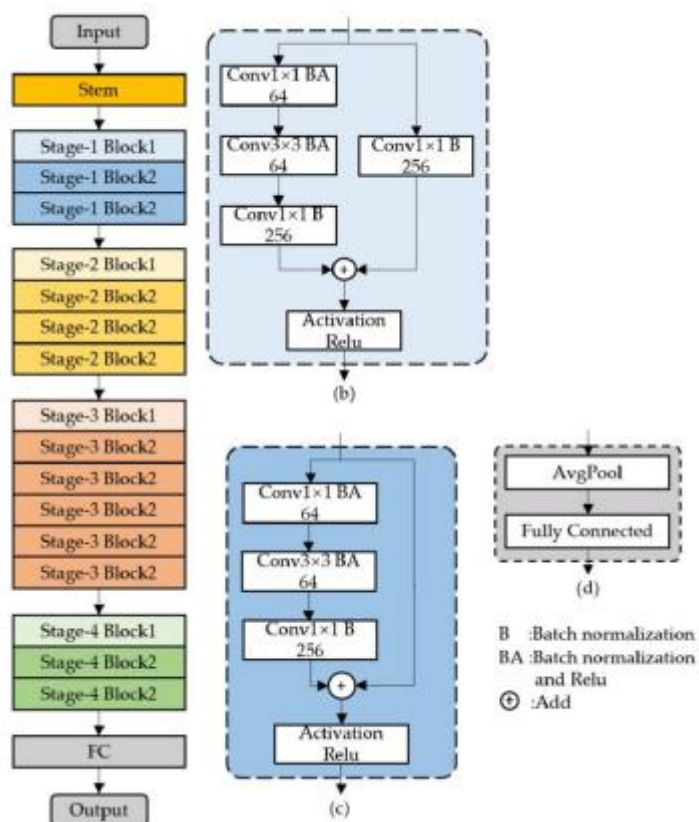
ResNet-50/101/152 (Bottleneck Block)

- Dùng **3 Conv** theo dạng bottleneck:

- 1×1 Conv (giảm chiều)
- 3×3 Conv (học đặc trưng)
- 1×1 Conv (tăng chiều lại)

Mục đích: giảm số phép tính khi mô hình rất sâu.

□ Minh họa kiến trúc ResNet-50:

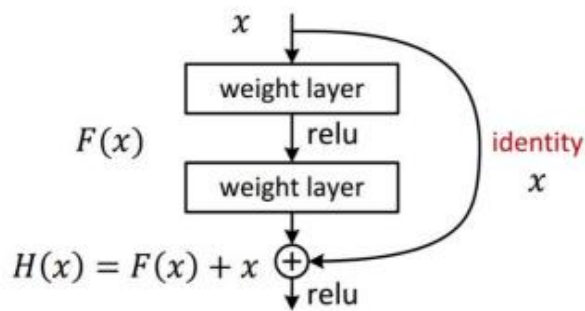


Shortcut Connection (Skip Connection)

Shortcut connection trong ResNet là đường kết nối tắt, cho phép input được cộng trực tiếp vào output. Giúp giải quyết

vanishing gradient

- Residual net



$H(x)$ is any desired mapping,
~~hope the 2 weight layers fit $H(x)$~~
hope the 2 weight layers fit $F(x)$
let $H(x) = F(x) + x$

Nhờ shortcut, ResNet có thể xây dựng mạng cực sâu (tới hàng trăm, thậm chí hàng nghìn layer) mà vẫn train hiệu quả.