ĐẠI HỌC TÔN ĐỨC THẮNG
TON DUC THANG UNIVERSITY

COMPUTER VISION

BÁO CÁO GIỮA KỲ

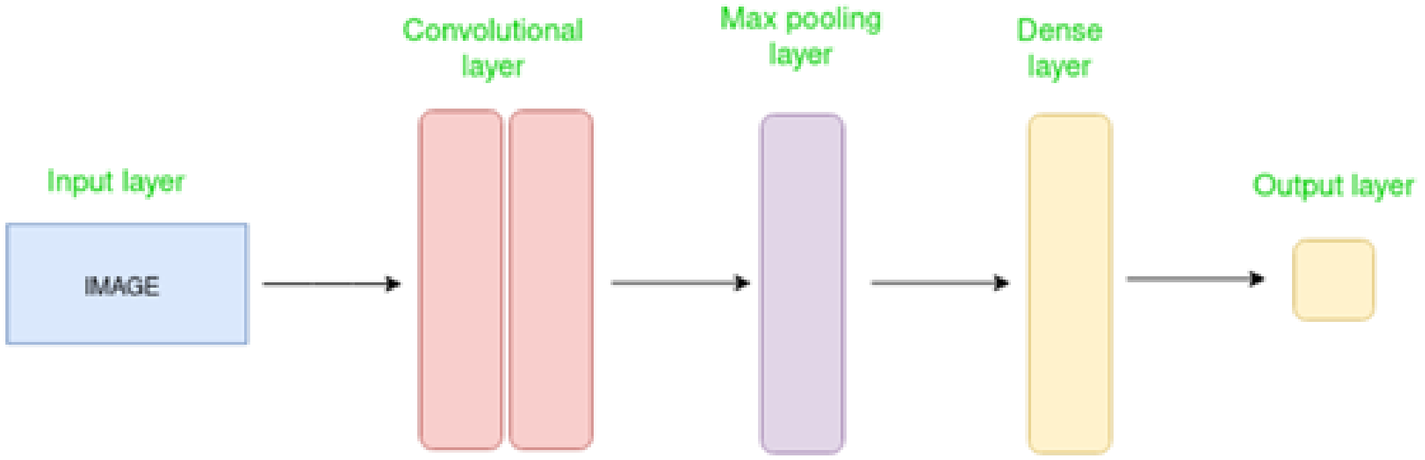
GIẢNG VIÊN HƯỚNG DẪN: TS.PHẠM VĂN HUY

Convolutional Neural Network

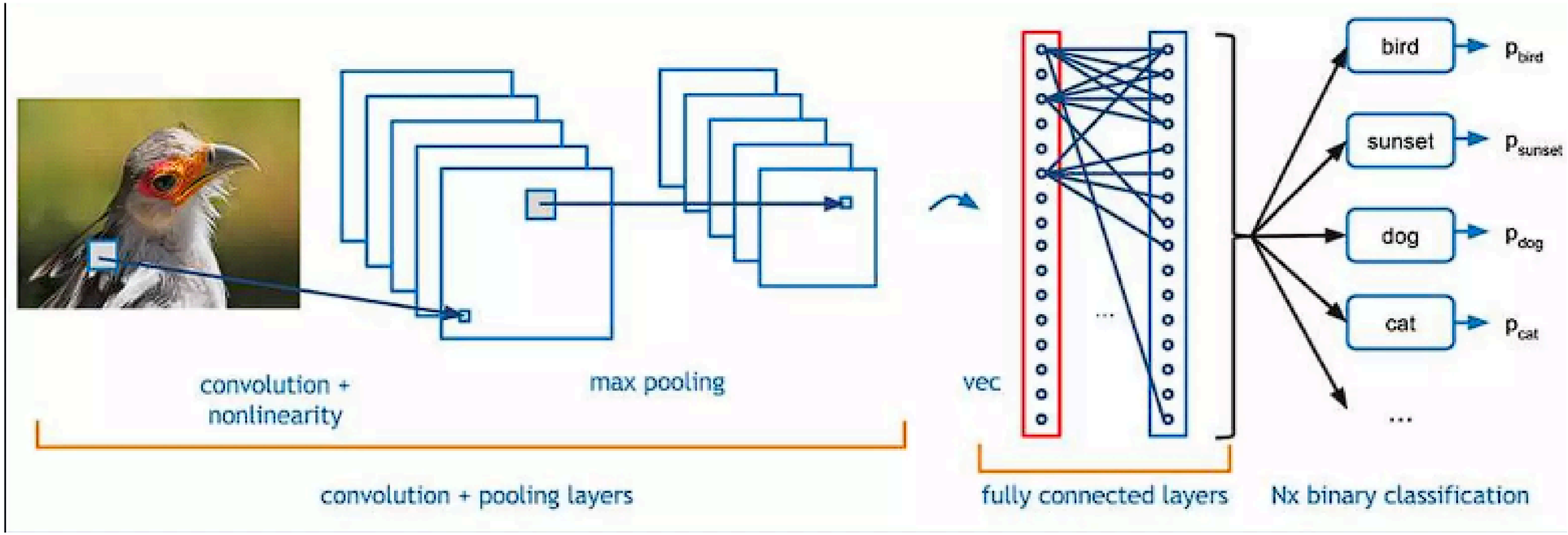
CNN là một nhánh của Deep Learning (học sâu).

Một trong những mô hình chính được sử dụng trong các bài toán như :

- nhận dạng hình ảnh
- phân loại hình ảnh
- phát hiện đối tượng
- nhận diện khuôn mặt, v.v.



Convolutional Neural Network



Convolution Layer

INPUT IMAGE

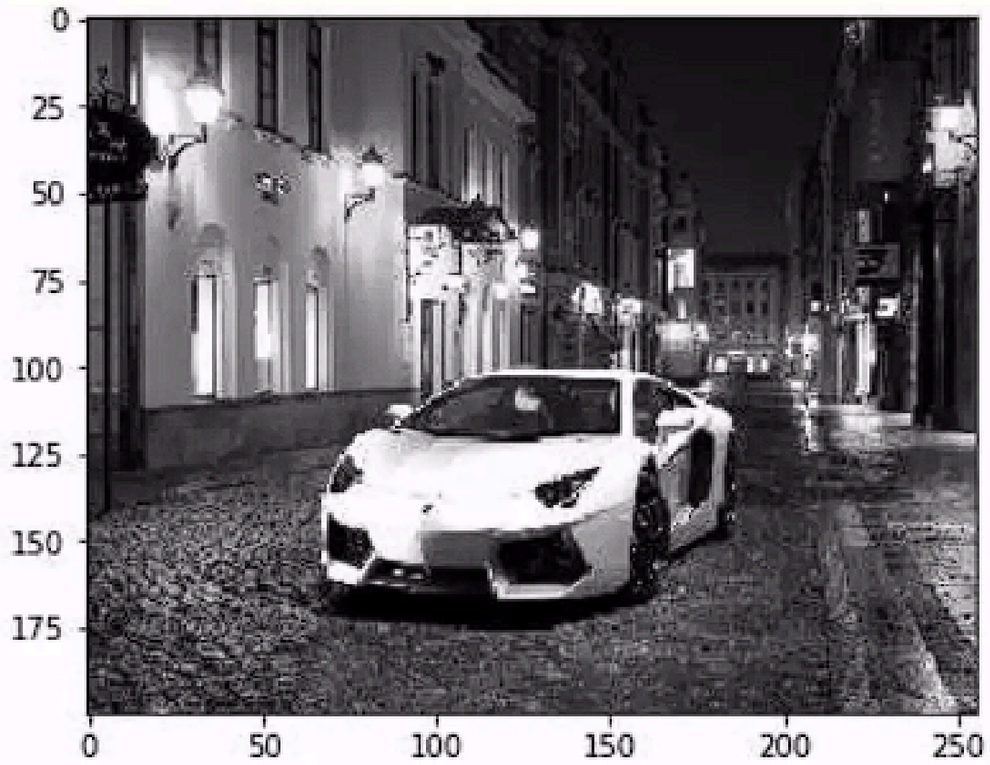
18	54	51	239	244	188
55	121	75	78	95	88
35	24	204	113	109	221
3	154	104	235	25	130
15	253	225	159	78	233
68	85	180	214	245	0

WEIGHT

1	0	1
0	1	0
1	0	1

429

Convolution Layer



Original Image

Convoluted image

Convolution Layer

Stride là số pixel mà bộ lọc di chuyển mỗi lần trượt trên ảnh.

Padding (đệm) là việc thêm viền 0 xung quanh ảnh đầu vào để:

- Giữ nguyên kích thước đầu ra
- Giúp bộ lọc bao phủ đầy đủ các cạnh của ảnh

0	0	0	0	0	0	0	0
0	18	54	51	239	244	188	0
0	55	121	75	78	95	88	0
0	35	24	204	113	109	221	0
0	3	154	104	235	25	130	0
0	15	253	225	159	78	233	0
0	68	85	180	214	245	0	0
0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0
0	18	54	51	239	244	188	0
0	55	121	75	78	95	88	0
0	35	24	204	113	109	221	0
0	3	154	104	235	25	130	0
0	15	253	225	159	78	233	0
0	68	85	180	214	245	0	0
0	0	0	0	0	0	0	0

WEIGHT

1	0	1
0	1	0
1	0	1

139

The Pooling Layer

Pooling Layer giúp giảm kích thước dữ liệu sau các lớp tích chập.

Nó lấy đặc trưng quan trọng nhất, thường bằng cách chọn giá trị lớn nhất trong vùng – gọi là Max Pooling.

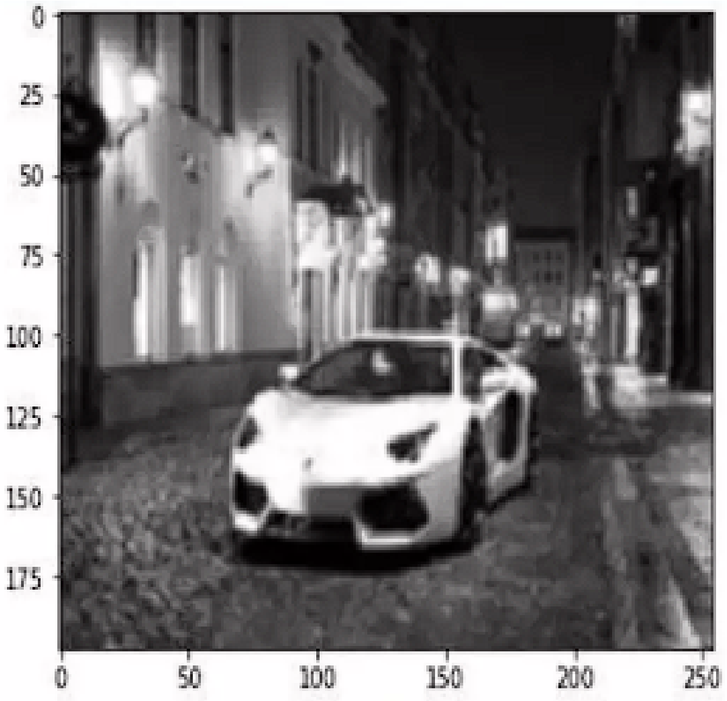
Nhờ vậy, mô hình học nhanh hơn và ít bị overfitting hơn.

429	505	686	856
261	792	412	640
633	653	851	751
608	913	713	657

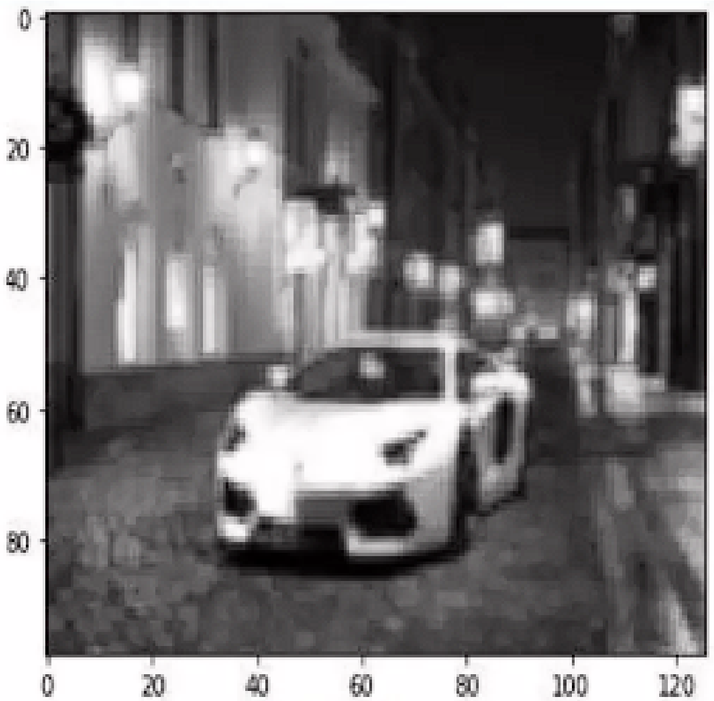
792	856
913	851

The Pooling Layer

Ảnh sau pooling nhỏ hơn nhưng vẫn giữ được hình dạng và đặc trưng quan trọng của xe. Điều này giúp mô hình giảm kích thước dữ liệu nhưng vẫn hiểu được nội dung hình ảnh.



Convoluted image

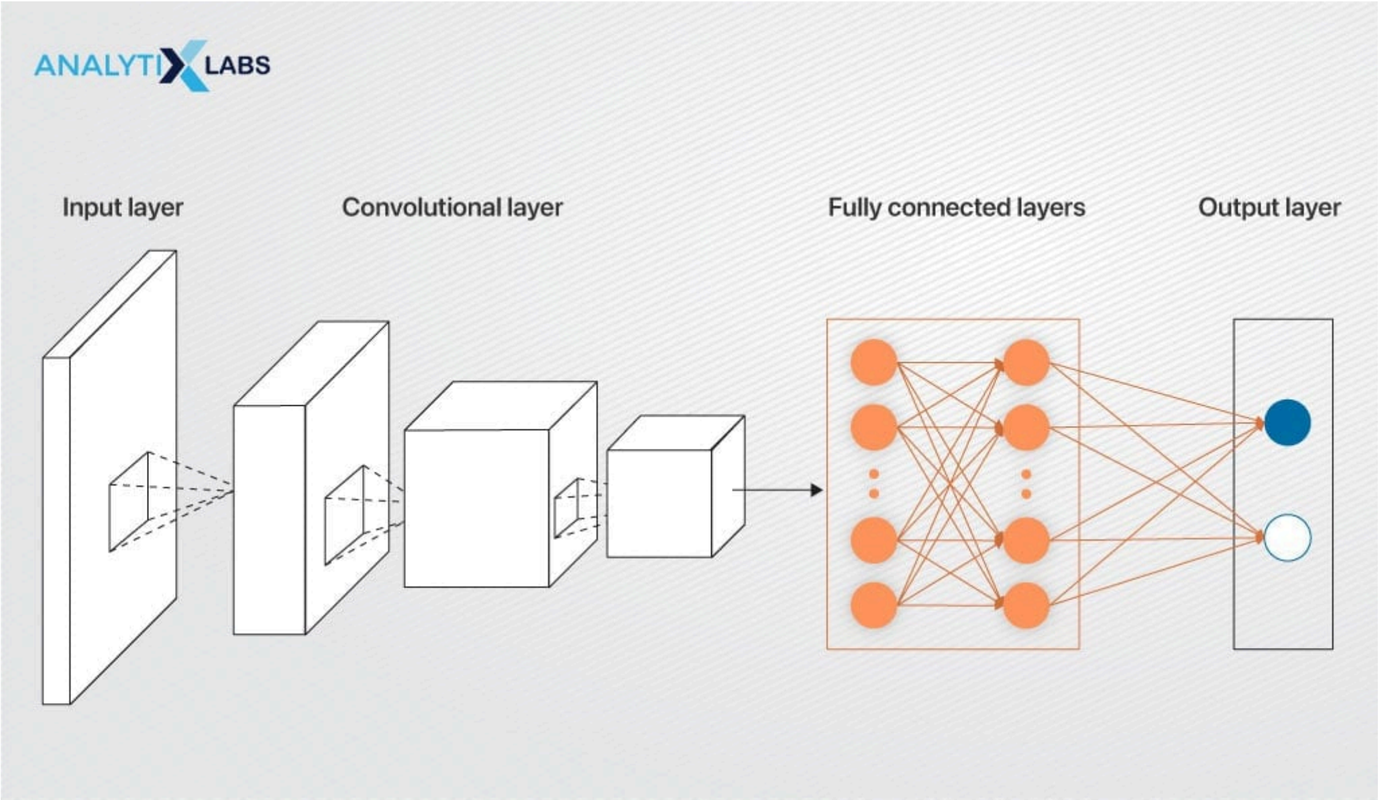


MAX Pooling

The Output Layer

Lớp Fully Connected sẽ chuyển đặc trưng thành xác suất, thường qua hàm Softmax.

Sau đó, mô hình tính sai số bằng Cross-Entropy và dùng backpropagation để cập nhật trọng số.



Các loại CNN

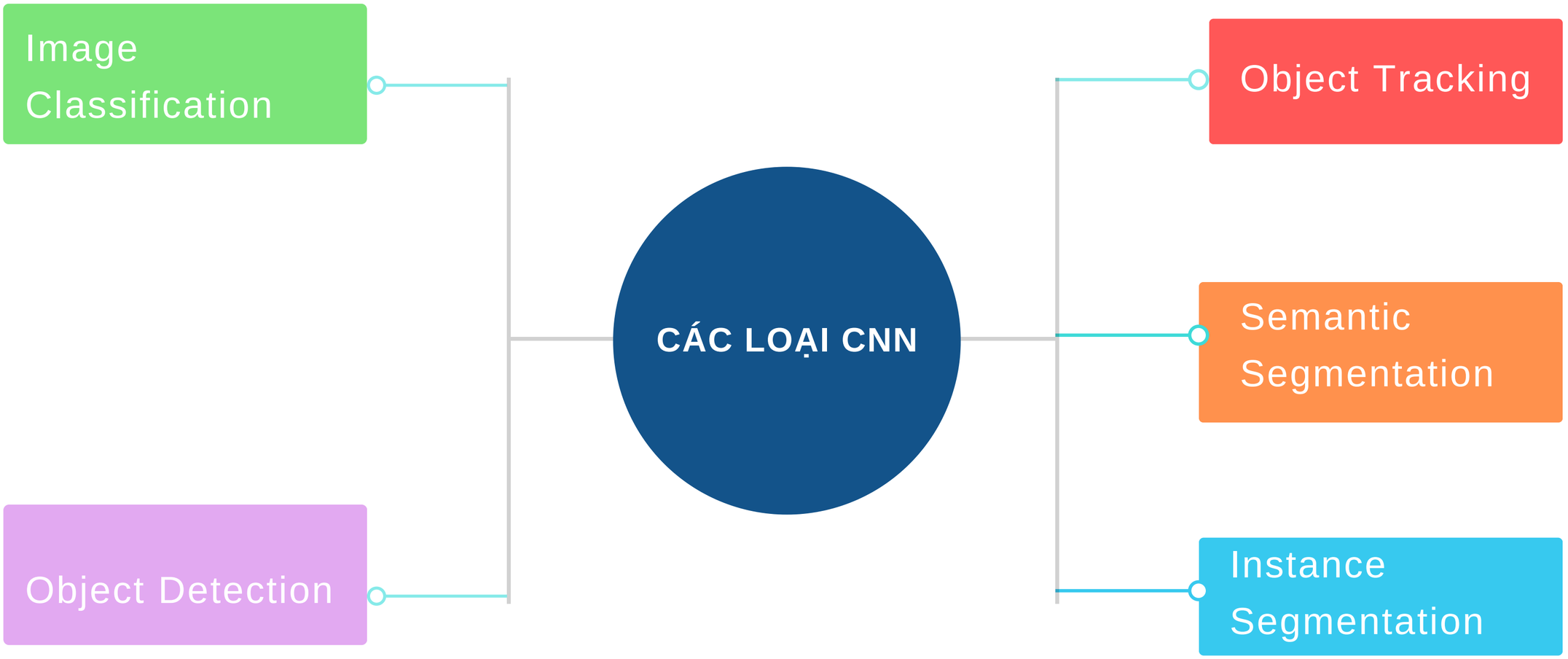


Image Classification
Object Detection
CÁC LOẠI CNN
Object Tracking
Semantic Segmentation
Instance Segmentation

Image Classification

LeNet (1998)

dùng cho nhận dạng chữ số viết tay

AlexNet (2012)

Mở đầu cho kỷ nguyên Deep Learning hiện đại, thắng giải ImageNet

ZFNet (2013)

Cải tiến AlexNet bằng cách điều chỉnh filter và visualization

GoogLeNet (2014)

Dùng kiến trúc Inception để tối ưu hiệu năng và tốc độ.

VGGNet (2014)

Dùng nhiều lớp tích chập nhỏ (3×3) giúp mô hình sâu và dễ huấn luyện.

ResNet (2015)

Giới thiệu “skip connection” để huấn luyện mạng cực sâu mà không bị mất thông tin.

Object Detection

R-CNN (2014)

Xác định vùng có khả năng chứa vật thể rồi dùng CNN để phân loại.

Fast R-CNN (2015)

Tăng tốc bằng cách chia sẻ feature map cho toàn ảnh.

Faster R - CNN(2016)

Thêm Region Proposal Network để tự động tìm vùng vật thể nhanh hơn.

YOLO_v1 (2016)

phát hiện vật thể trong một lần chạy duy nhất — nhanh hơn nhiều, phù hợp thời gian thực.

Mask R-CNN(2017)

Thêm branch mới để dự đoán mặt nạ (mask) cho từng đối tượng → dùng cho Instance Segmentation

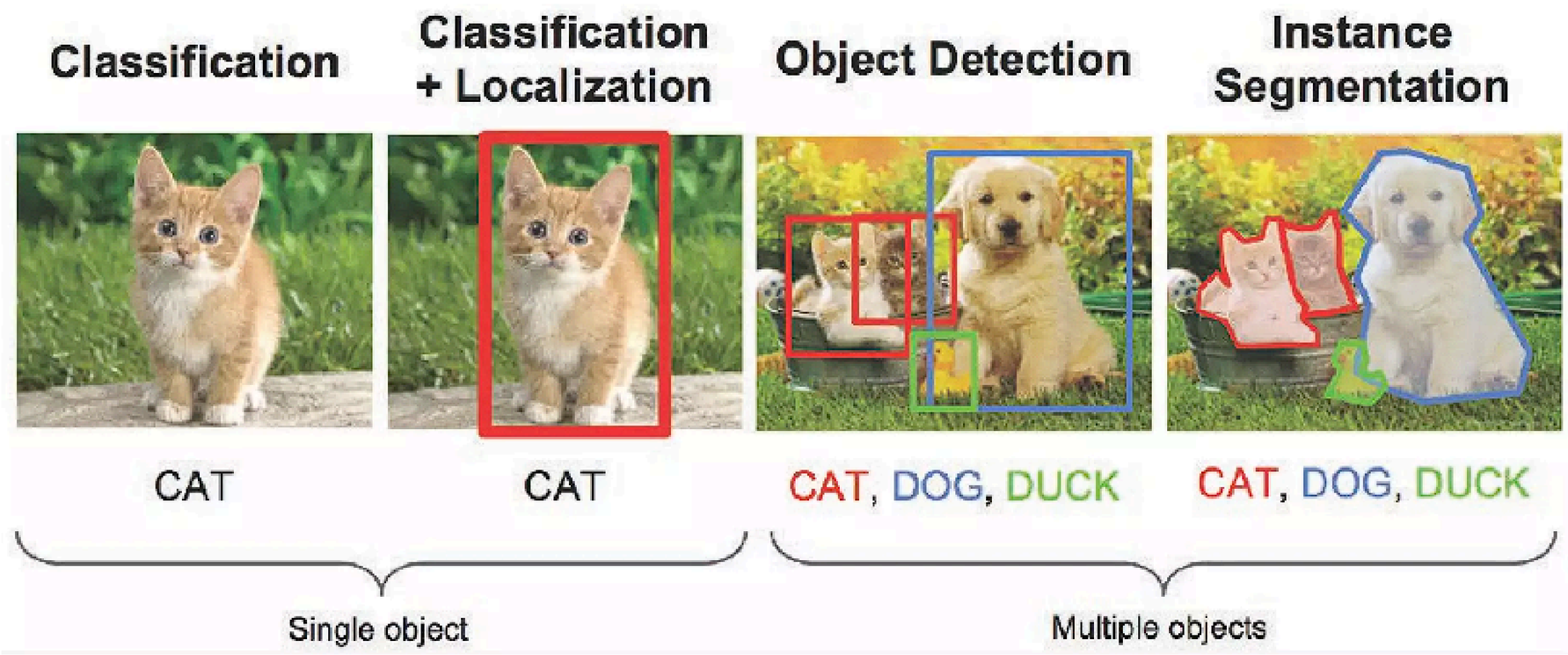
YOLO_v11 (2024)

Cải tiến backbone + attention, hỗ trợ đa nhiệm (detect, segment, pose) → tối ưu tốc độ và độ chính xác.

Regions with CNN features

CNN truyền thống chỉ làm tốt việc phân loại toàn bộ hình ảnh, chứ không thể xác định vị trí từng vật thể trong ảnh.

- R-CNN : Selective Search->Cropped Imaged ->CNN



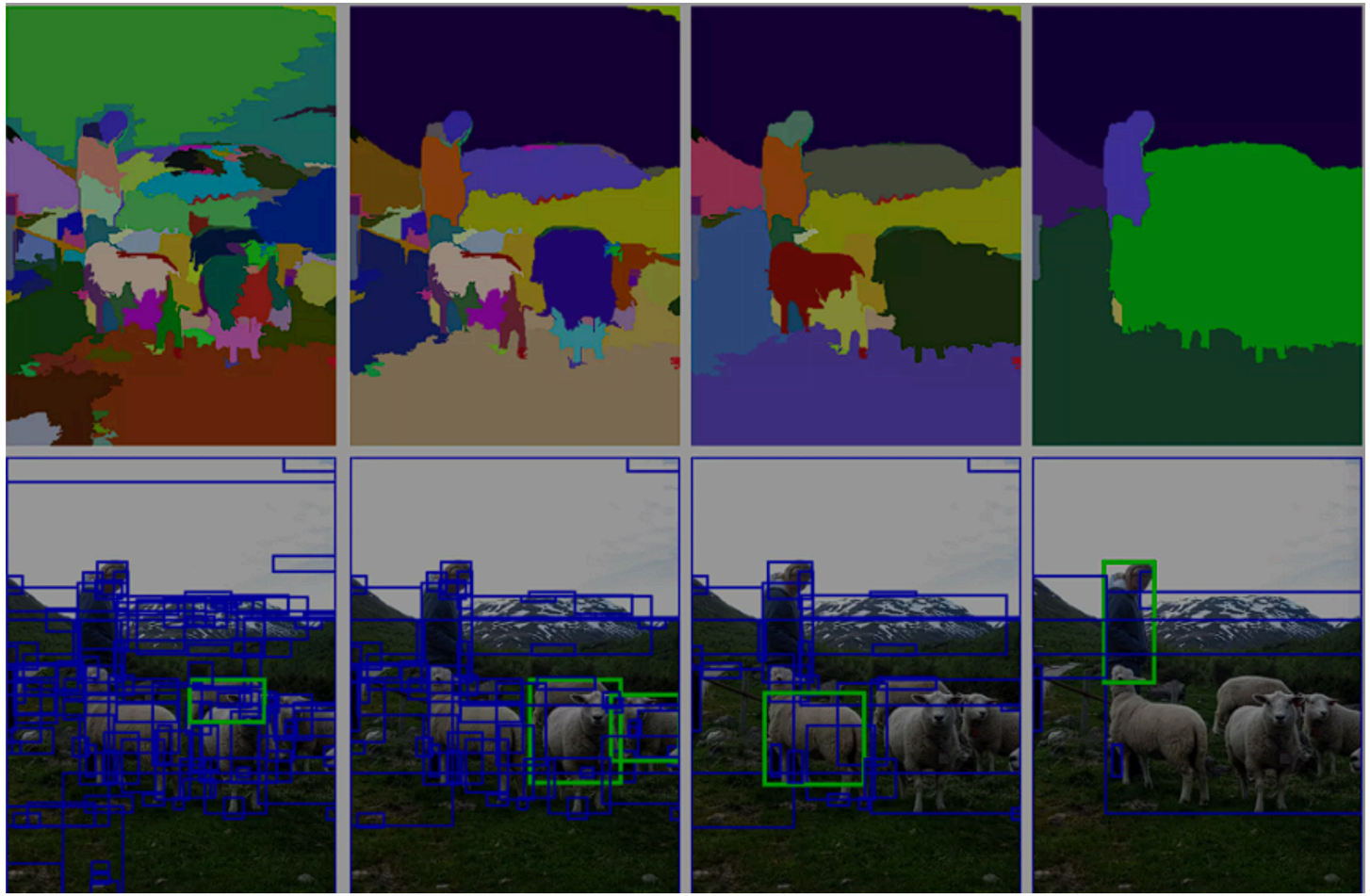
Selective Search

Segmentation + Region Merging

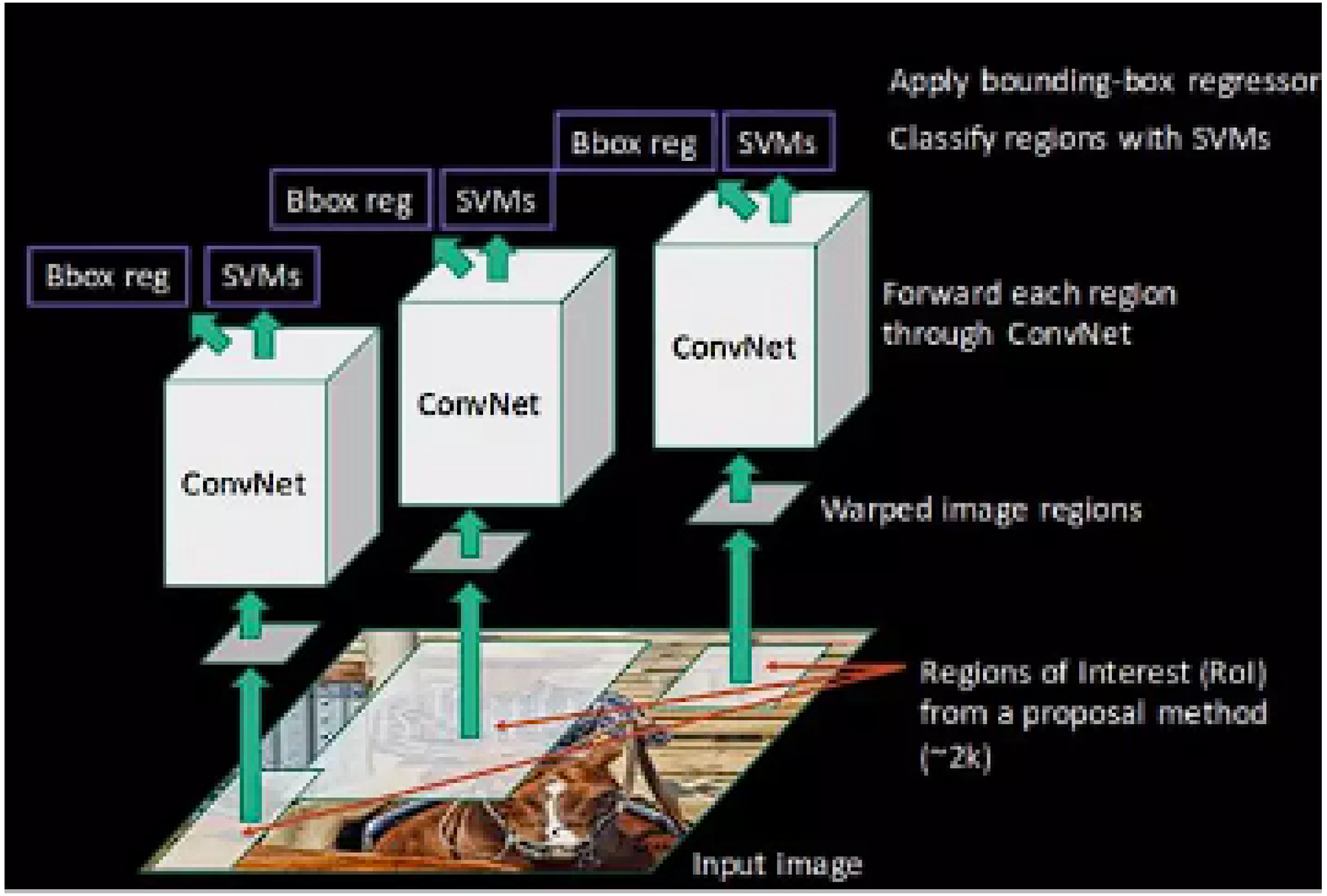
Chia ảnh thành nhiều vùng nhỏ dựa trên màu, kết cấu, độ sáng.

Region Proposal

Sinh ra ~2000 hộp giới hạn đại diện cho các vùng có khả năng chứa vật thể.



Quy trình hoạt động của R-CNN



Các vấn đề của R-CNN

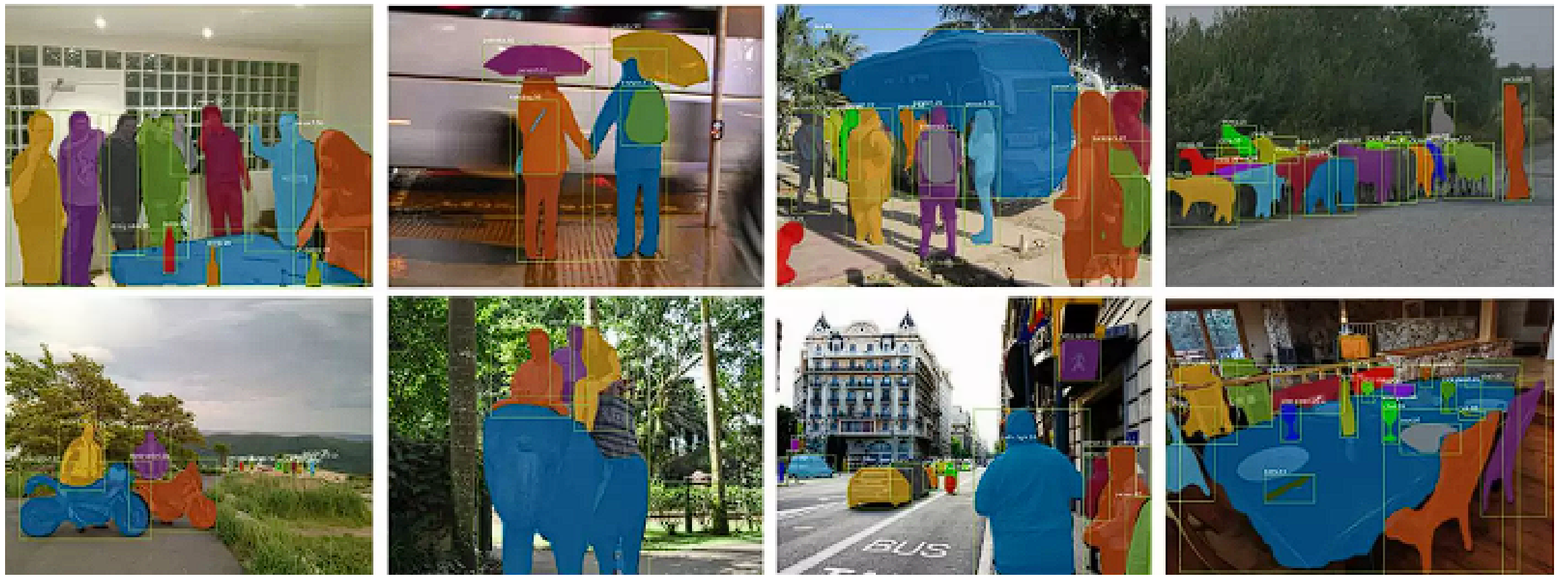
1. **Quá nhiều vùng cần xử lý**
   - Mỗi ảnh trích xuất 2000 vùng (Selective Search)
   - Tốn nhiều thời gian và không thể huấn luyện end-to-end
2. **CNN chạy lặp lại quá nhiều lần**
   - Với N ảnh, R-CNN phải chạy CNN tới N × 2000 lần,
   - → Mô hình rất chậm khi huấn luyện và dự đoán.
3. **Ba mô hình riêng biệt**
   - Quá trình phát hiện đối tượng phải chia thành 3 phần:
     a. **CNN** để trích đặc trưng
     b. **SVM** để phân loại
     c. **Hồi quy tuyến tính** để hiệu chỉnh hộp bao

→ Làm cho pipeline phức tạp, khó tối ưu và chiếm nhiều tài nguyên.

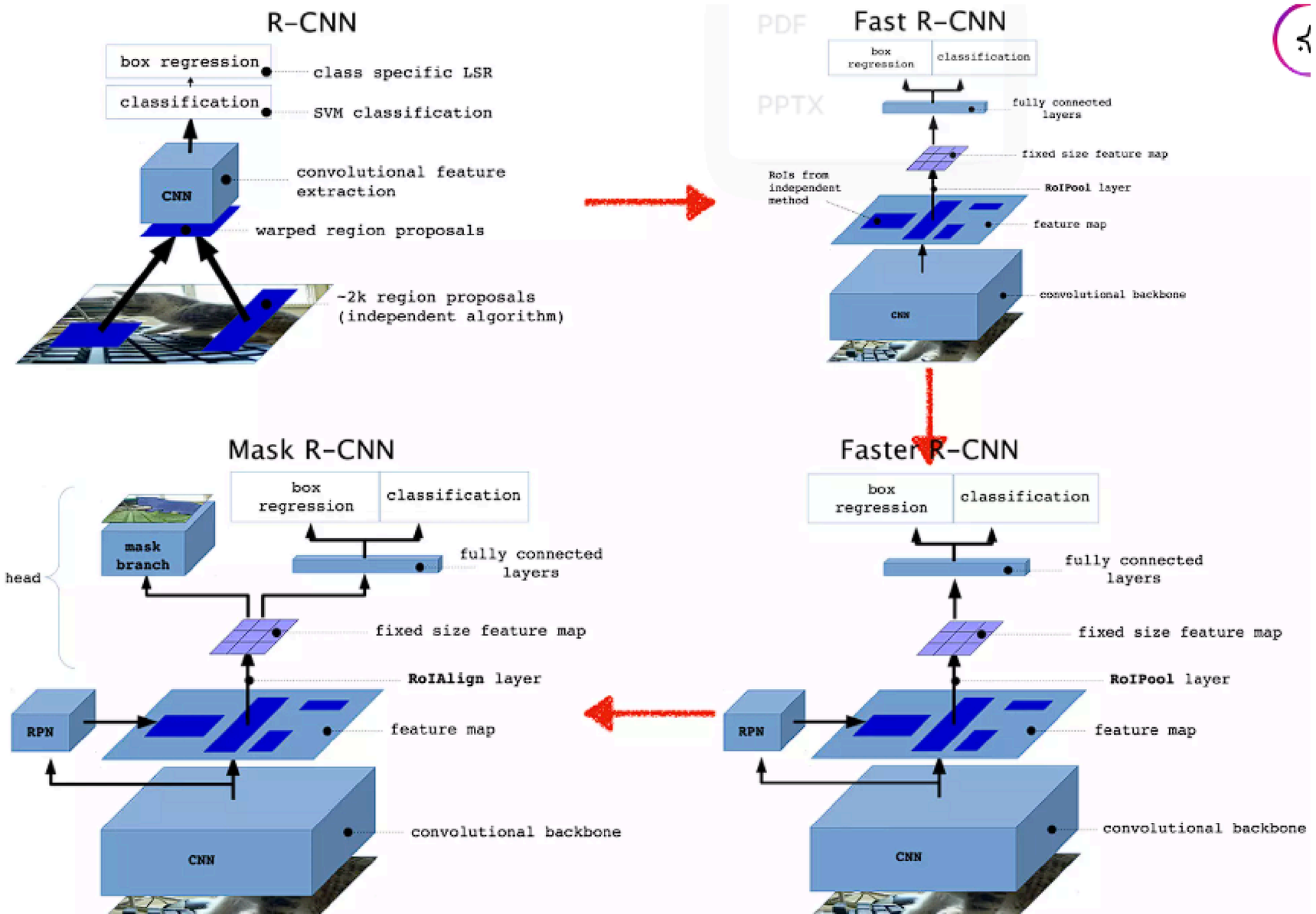
Mask R-CNN

Mask R-CNN ra đời để khắc phục giới hạn của Faster R-CNN — chỉ khoanh vùng vật thể mà chưa tách rời từng đối tượng.

Nó thêm một nhánh mới để dự đoán mặt nạ (mask) song song với bounding box, giúp mô hình vừa phát hiện, vừa phân vùng chính xác từng đối tượng riêng biệt.



Mask R-CNN


R-CNN
box regression
class specific LSR
classification
SVM classification
CNN
convolutional feature extraction
warped region proposals
~2k region proposals (independent algorithm)
Fast R-CNN
box regression
classification
fully connected layers
fixed size feature map
RoIs from independent method
RoIPool layer
feature map
CNN
convolutional backbone
Faster R-CNN
box regression
classification
fully connected layers
fixed size feature map
RoIPool layer
RPN
feature map
CNN
convolutional backbone
Mask R-CNN
box regression
classification
mask branch
head
fully connected layers
fixed size feature map
RoIAlign layer
RPN
feature map
CNN
convolutional backbone

Mask R-CNN

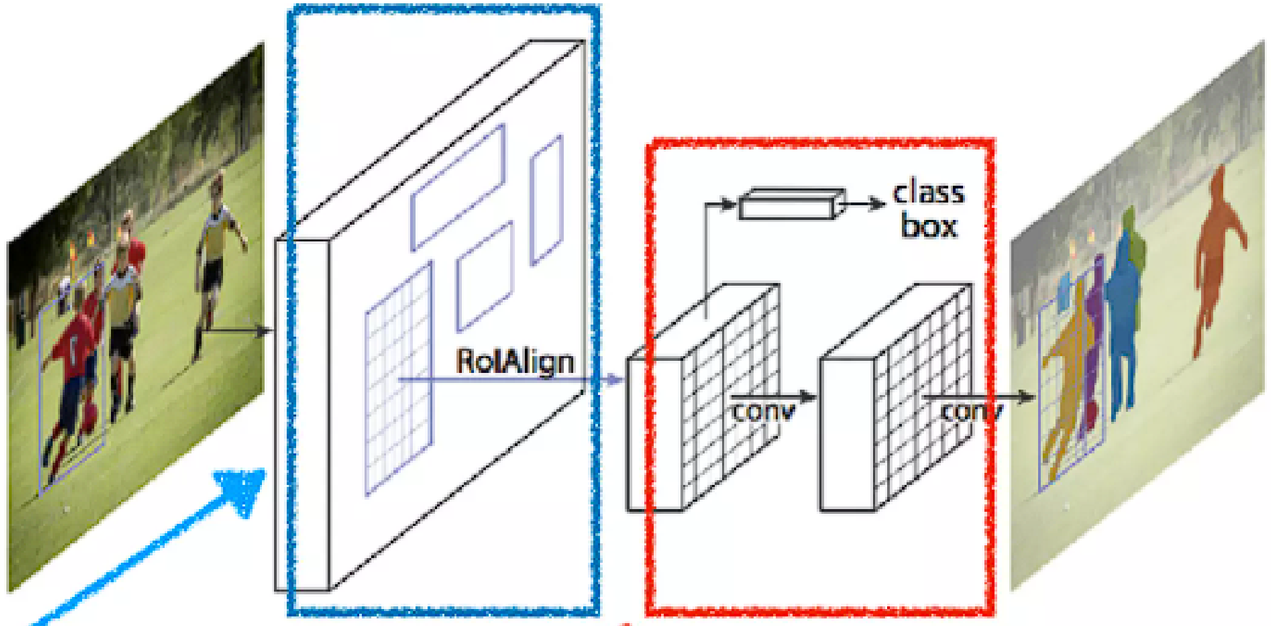


Figure 1. The **Mask R-CNN** framework for instance segmentation.

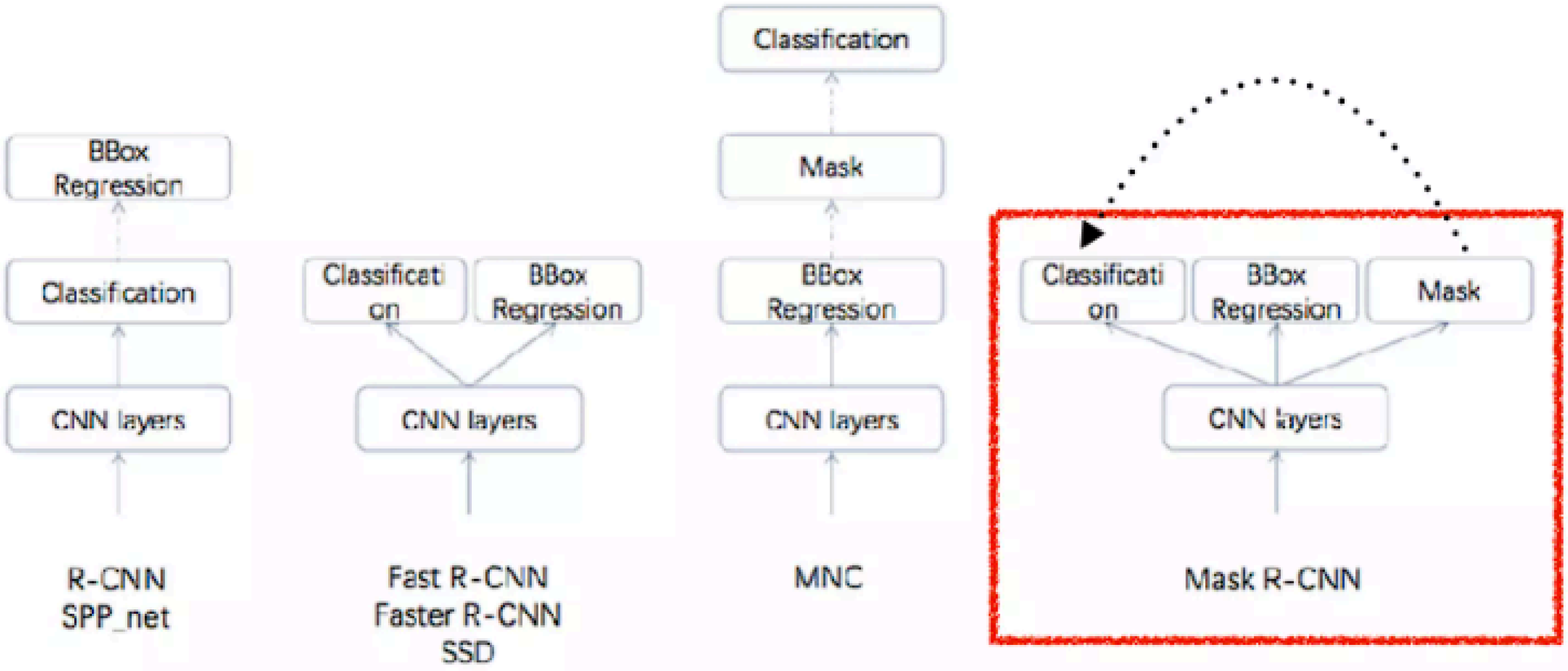
1. RoIAlign

Giúp căn chỉnh chính xác giữa vùng đề xuất và đặc trưng - khắc phục lỗi làm tròn tọa độ pixel của Faster R-CNN.

2. Mask Branch

Thêm một nhánh dự đoán mặt nạ, chạy song song với phân loại và định vị, giúp mô hình vừa phát hiện vừa phân vùng chi tiết từng đối tượng.

Mask R-CNN


BBox Regression
Classification
CNN layers
R-CNN
SPP_net
Classificati on
BBox Regression
CNN layers
Fast R-CNN
Faster R-CNN
SSD
Classification
Mask
BBox Regression
CNN layers
MNC
Classificati on
BBox Regression
Mask
CNN layers
Mask R-CNN

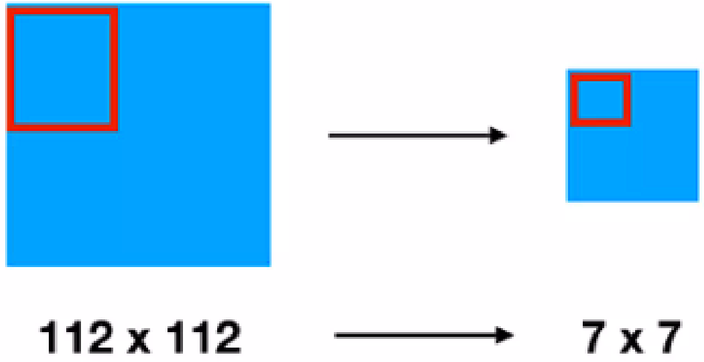
RoIAlign

Vấn đề của RoIPool (trước đây):

Trong Faster R-CNN, lớp RoIPool chia vùng đề xuất thành các ô cố định.

Nhưng nó làm tròn tọa độ pixel, gây sai lệch vị trí giữa vùng thật và đặc trưng

→ làm giảm độ chính xác khi tạo mask.



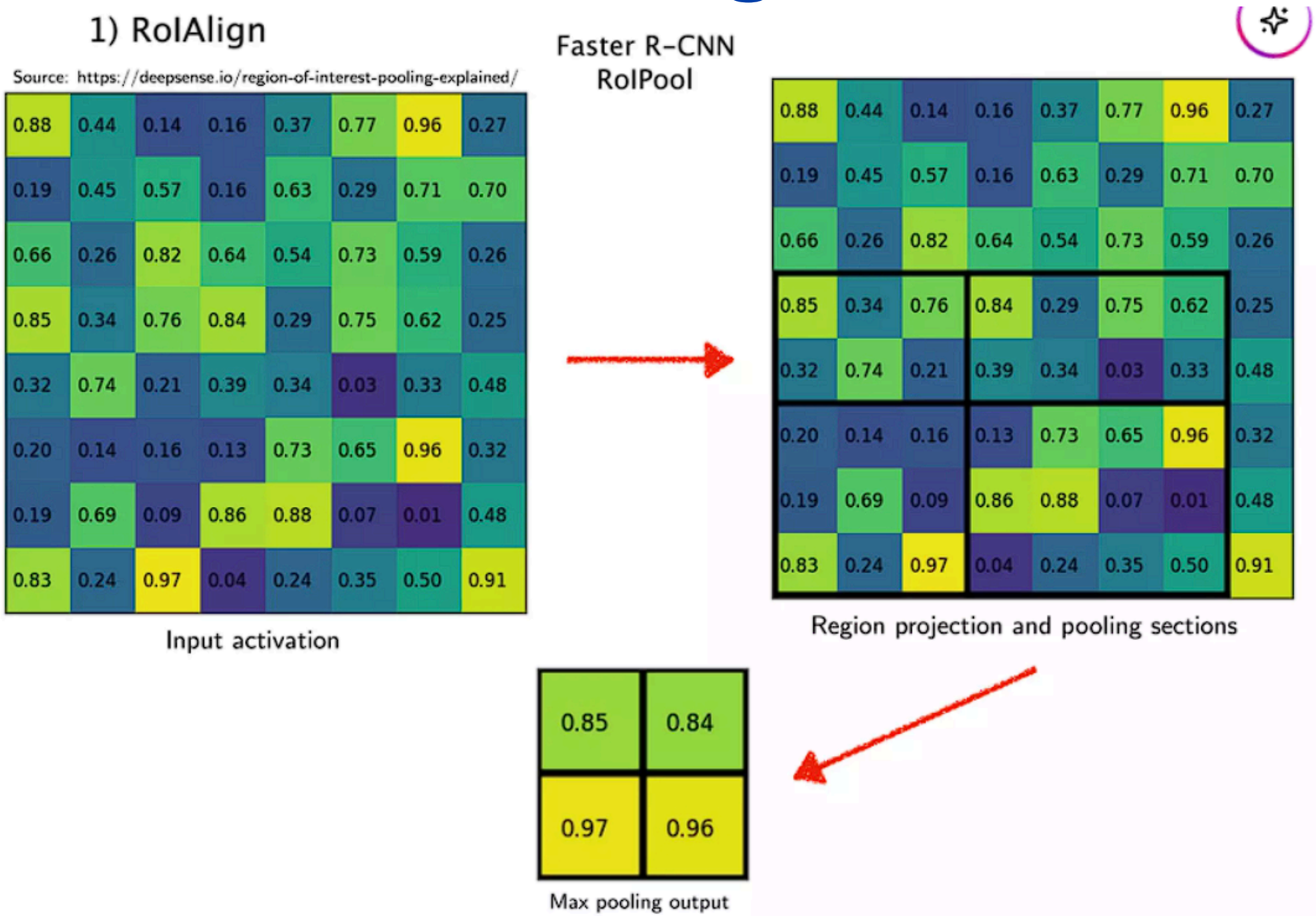
Ex)
[32/16] = 2, 32/16 = 2
[33/16] = 2, 33/16 = 2.06
[34/16] = 2, 34/16 = 2.12
[35/16] = 2, 35/16 = 2.18
[36/16] = 2, 36/16 = 2.25
[37/16] = 2, 37/16 = 2.31
[38/16] = 2, 38/16 = 2.37
[39/16] = 2, 39/16 = 2.43
[40/16] = 3, 40/16 = 2.5

Giải pháp – RoIAlign:

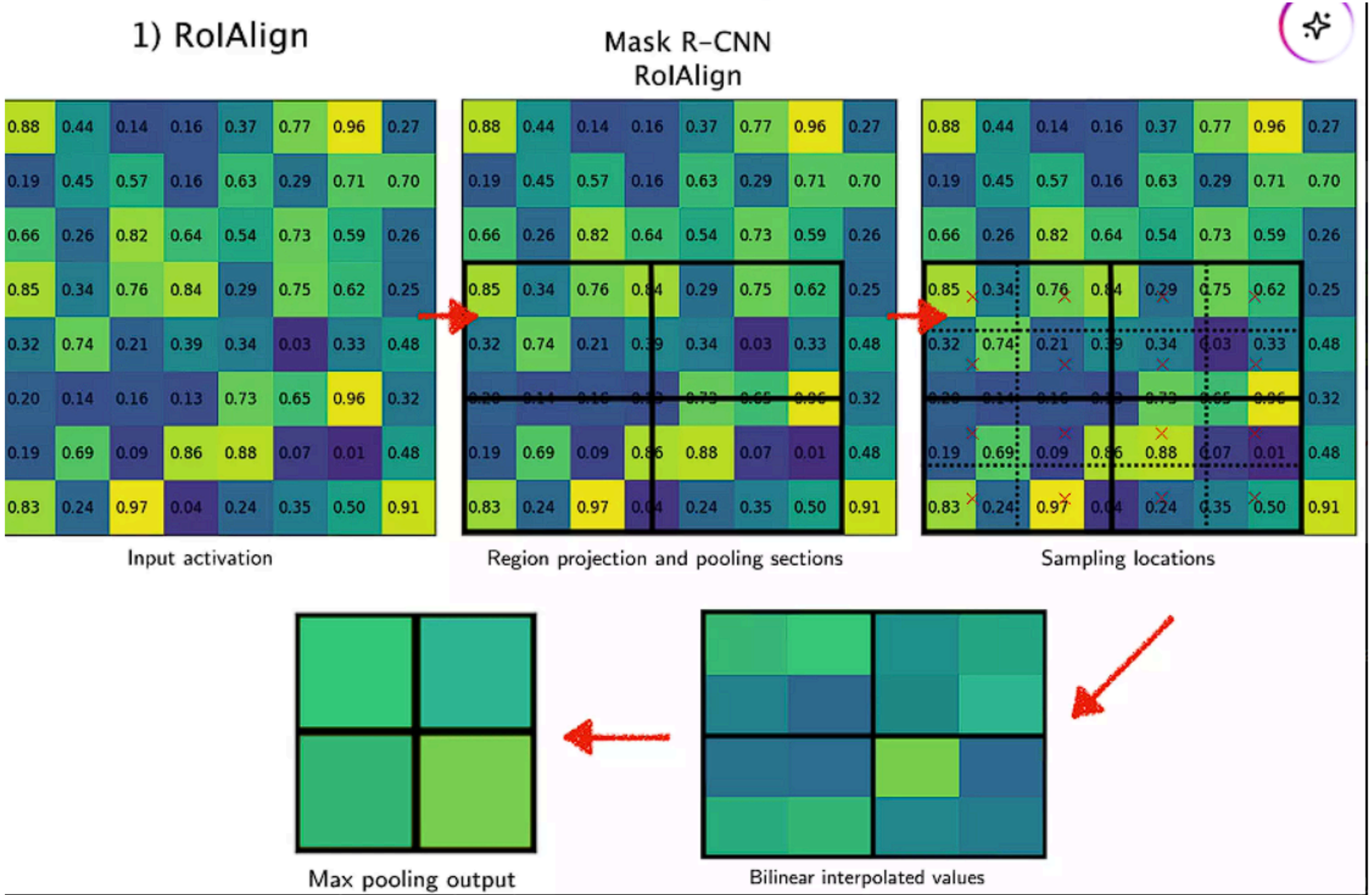
- Loại bỏ bước làm tròn, thay bằng nội suy tuyến tính để giữ nguyên tỷ lệ chính xác của vị trí pixel.

→ Mask Accuracy tăng 10% đến 50%

RoIAlign


1) RoIAlign
Source: https://deepsense.io/region-of-interest-pooling-explained/
0.88 0.44 0.14 0.16 0.37 0.77 0.96 0.27
0.19 0.45 0.57 0.16 0.63 0.29 0.71 0.70
0.66 0.26 0.82 0.64 0.54 0.73 0.59 0.26
0.85 0.34 0.76 0.84 0.29 0.75 0.62 0.25
0.32 0.74 0.21 0.39 0.34 0.03 0.33 0.48
0.20 0.14 0.16 0.13 0.73 0.65 0.96 0.32
0.19 0.69 0.09 0.86 0.88 0.07 0.01 0.48
0.83 0.24 0.97 0.04 0.24 0.35 0.50 0.91
Input activation
Faster R-CNN
RoIPool
0.88 0.44 0.14 0.16 0.37 0.77 0.96 0.27
0.19 0.45 0.57 0.16 0.63 0.29 0.71 0.70
0.66 0.26 0.82 0.64 0.54 0.73 0.59 0.26
0.85 0.34 0.76 0.84 0.29 0.75 0.62 0.25
0.32 0.74 0.21 0.39 0.34 0.03 0.33 0.48
0.20 0.14 0.16 0.13 0.73 0.65 0.96 0.32
0.19 0.69 0.09 0.86 0.88 0.07 0.01 0.48
0.83 0.24 0.97 0.04 0.24 0.35 0.50 0.91
Region projection and pooling sections
0.85 0.84
0.97 0.96
Max pooling output

RoIAlign


1) RoIAlign
Mask R-CNN
RoIAlign
Input activation
Region projection and pooling sections
Sampling locations
Max pooling output
Bilinear interpolated values

RoIAlign

Bilinear Interpolation

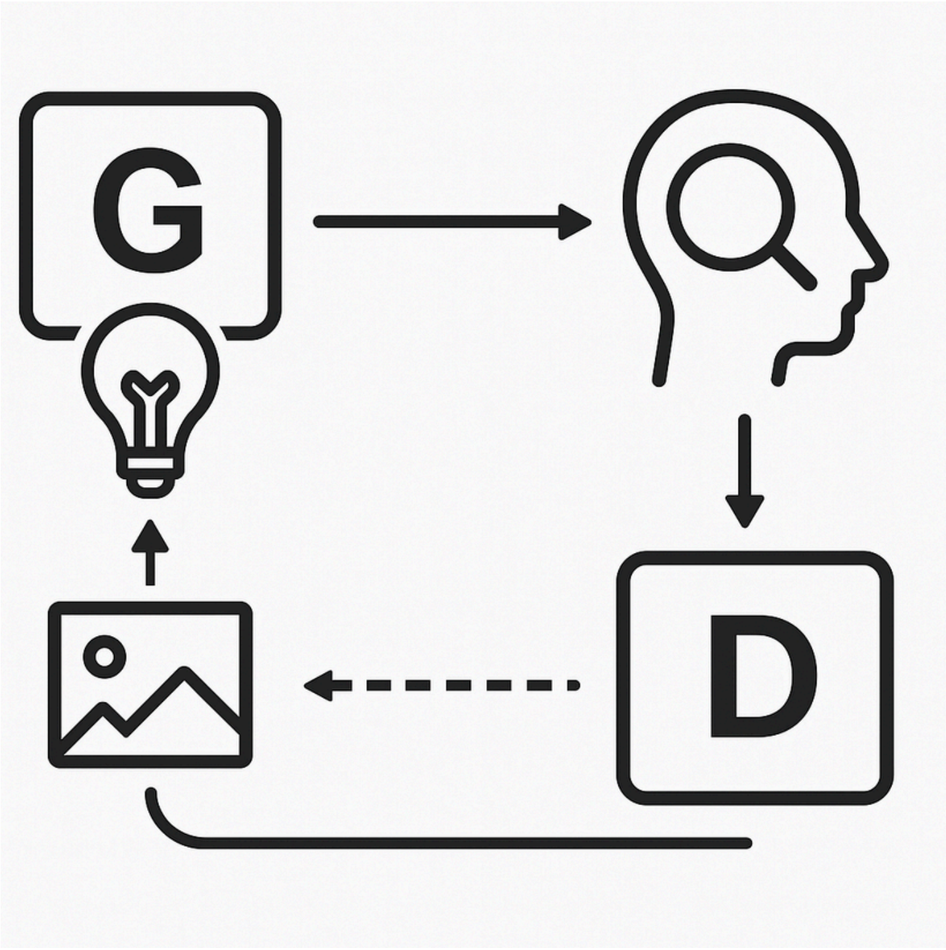
$$V(x, y) = \sum_{i=1}^{4} w_i \cdot V_i$$

Trong đó:

- V_i: Giá trị pixel lân cận
- w_i: Trọng số theo khoảng cách

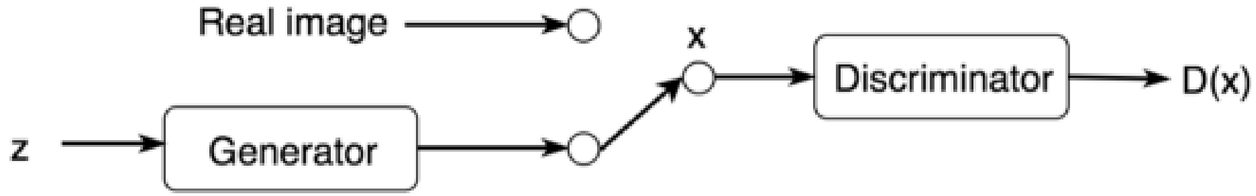
Kiến trúc của GAN

Generator

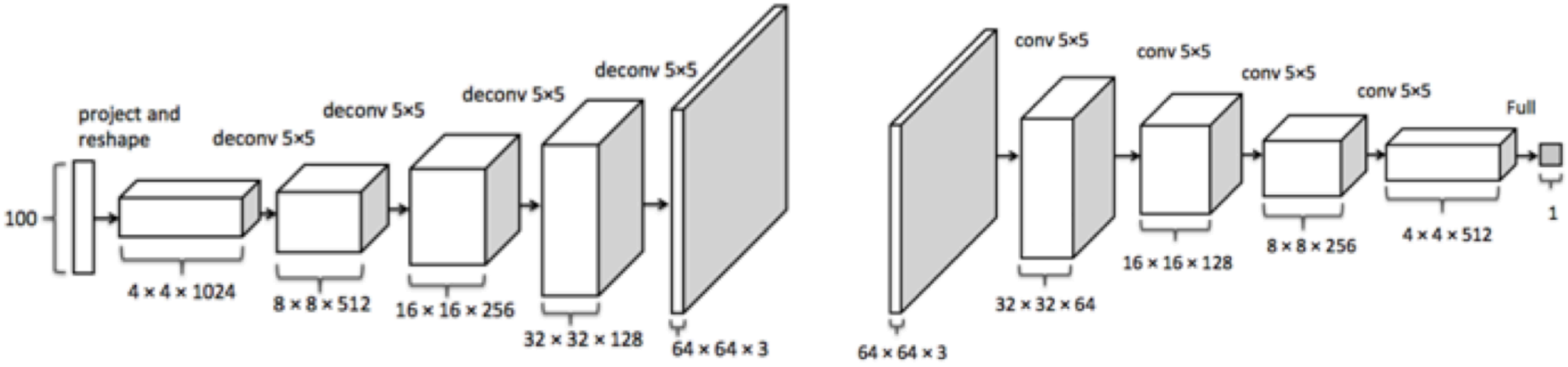


Discriminator

Kiến trúc của GAN



- z là một điểm nhiễu(sinh ra từ phân phối xác suất phổ biến nhất là Gaussian).
- G và D là các mạng Convolution nhiều lớp, sâu như hình dưới đây:



Kiến trúc của GAN

Để train được D, input gồm cả G(z) và x kèm nhãn. Như vậy, mục tiêu của D là maximinze:

$$\max_D V(D) = \underbrace{\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})]}_{\text{recognize real images better}} + \underbrace{\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]}_{\text{recognize generated images better}}$$

Để train được G, ta dựa vào D(G(z)). Update các weight của G sao cho G(z) có thể đánh lừa được D, khiến D đoán nhầm nhãn của G(z) là y = 1. G cố gắng minimize:

$$\min_G V(G) = \underbrace{\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]}_{\text{Optimize G that can fool the discriminator the most.}}$$

Kiến trúc của GAN

Tổng quát lại, D, G là kết quả của quá trình:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

PSEUDO CODE của GAN

pseudo code

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

    • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.

    • Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.

    • Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

  • Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.

  • Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

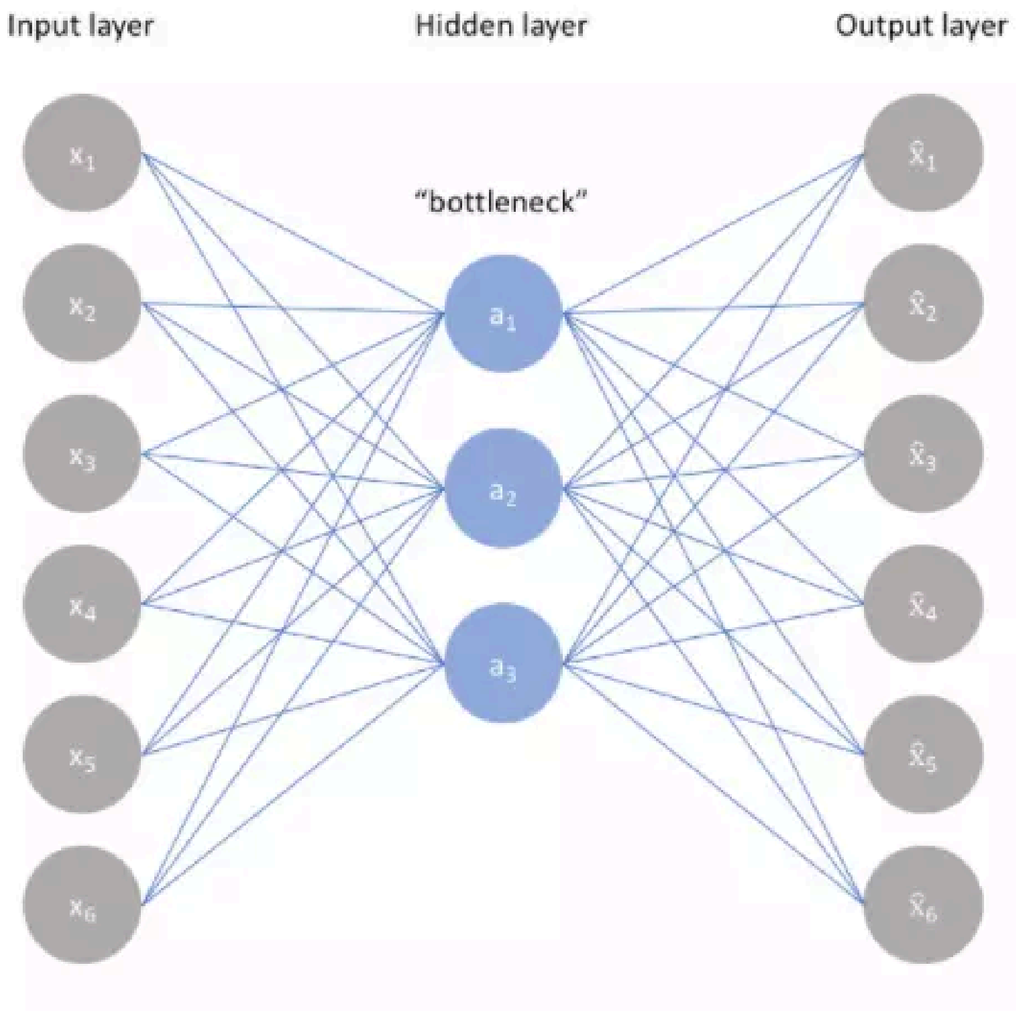
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Tại sao cần Autoencoder

- Học phi tuyến nhờ activation và nhiều tầng ẩn.
- Có thể dùng convolutional layer thay fully connected (phù hợp ảnh, video, chuỗi).
- Nhiều tầng nhỏ hiệu quả hơn PCA tuyến tính lớn.
- Autoencoder tạo biểu diễn nén ở từng tầng.
- Có thể dùng lại tầng đã huấn luyện để transfer learning.

AUTOENCODER

- Mạng nơ ron nhân tạo
- Mã hóa dữ liệu theo unsupervised
- Tìm ra cấu trúc ẩn trong dữ liệu, phân loại dữ liệu
- Tạo ra các nhóm dữ liệu mà không cần sự hướng dẫn từ dữ liệu huấn luyện



Ứng dụng của autoencoder

Image Coloring

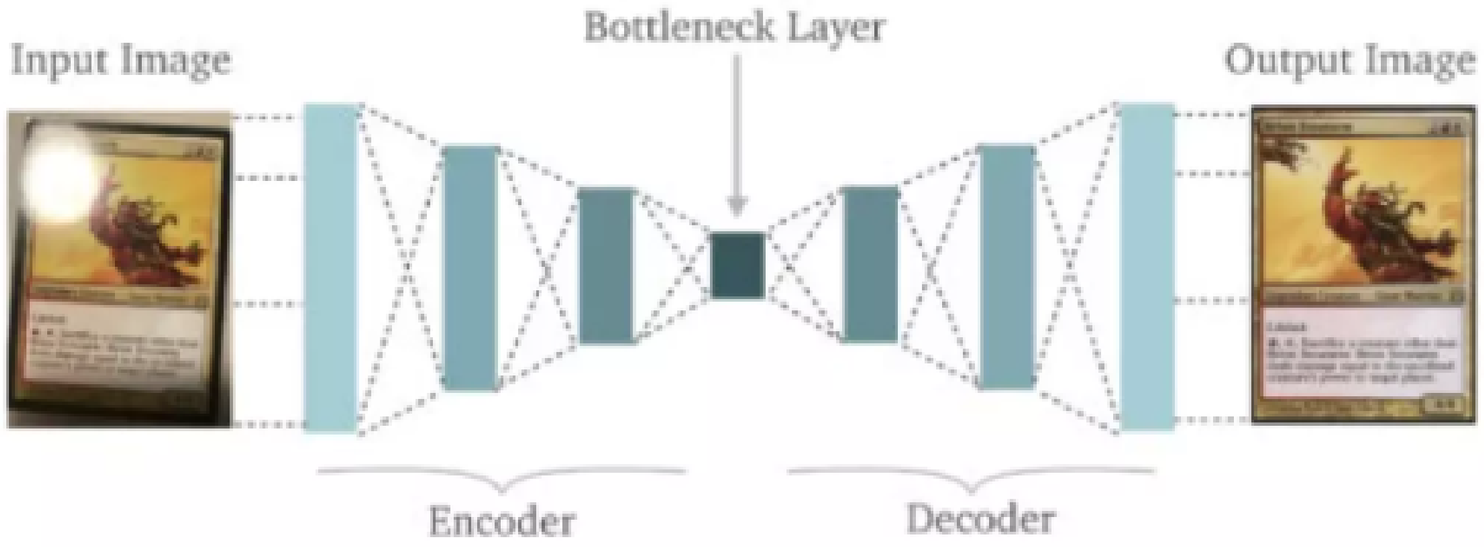
Cách hoạt động:

Encoder học đặc trưng (features) của ảnh đen trắng.

Decoder dự đoán và sinh lại ảnh ở dạng màu.

Ứng dụng của autoencoder

Feature Variation

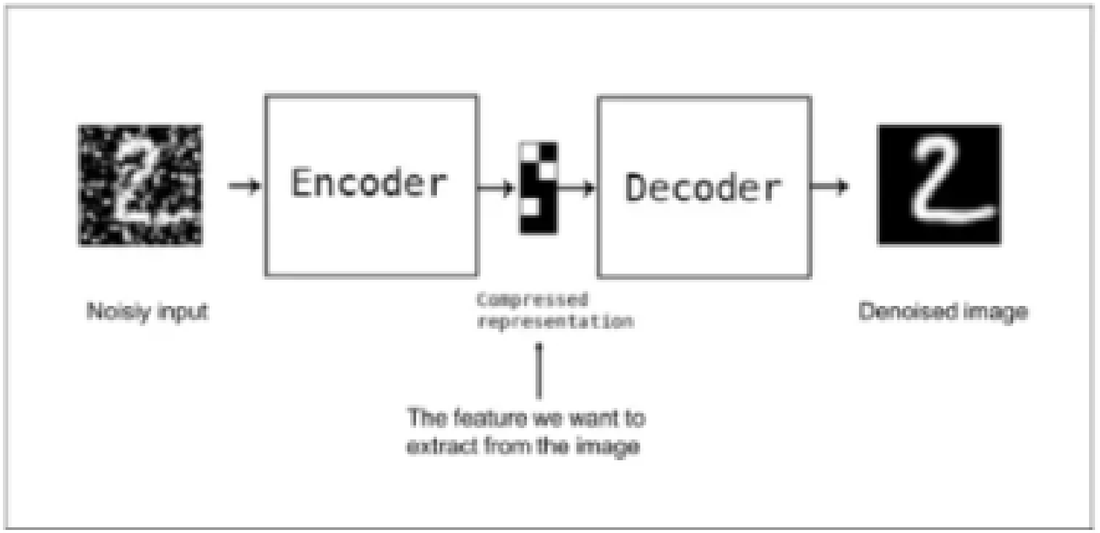


Ứng dụng của autoencoder

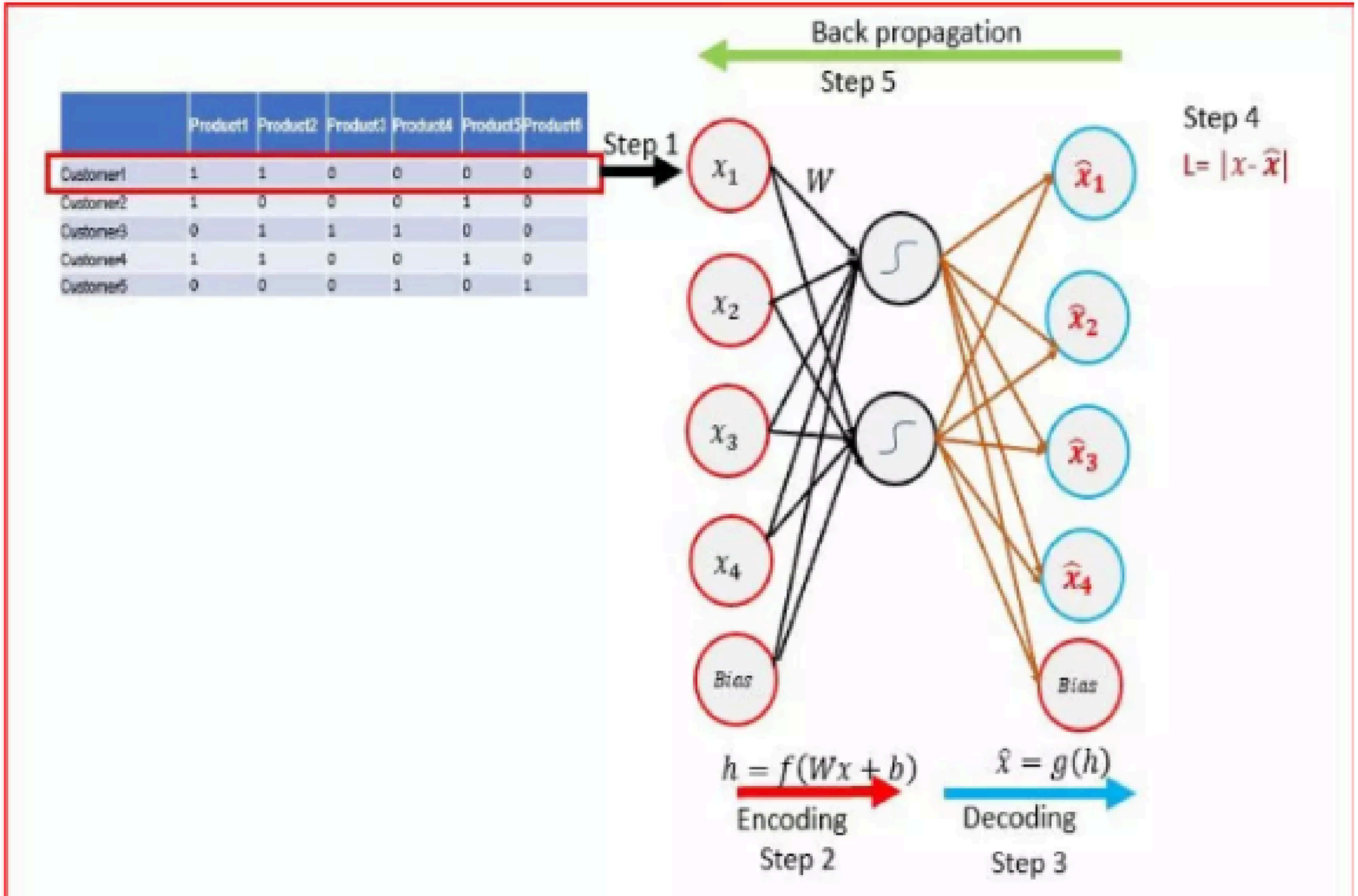
Dimensionality Reduction

Ứng dụng của autoencoder

Denoising Image

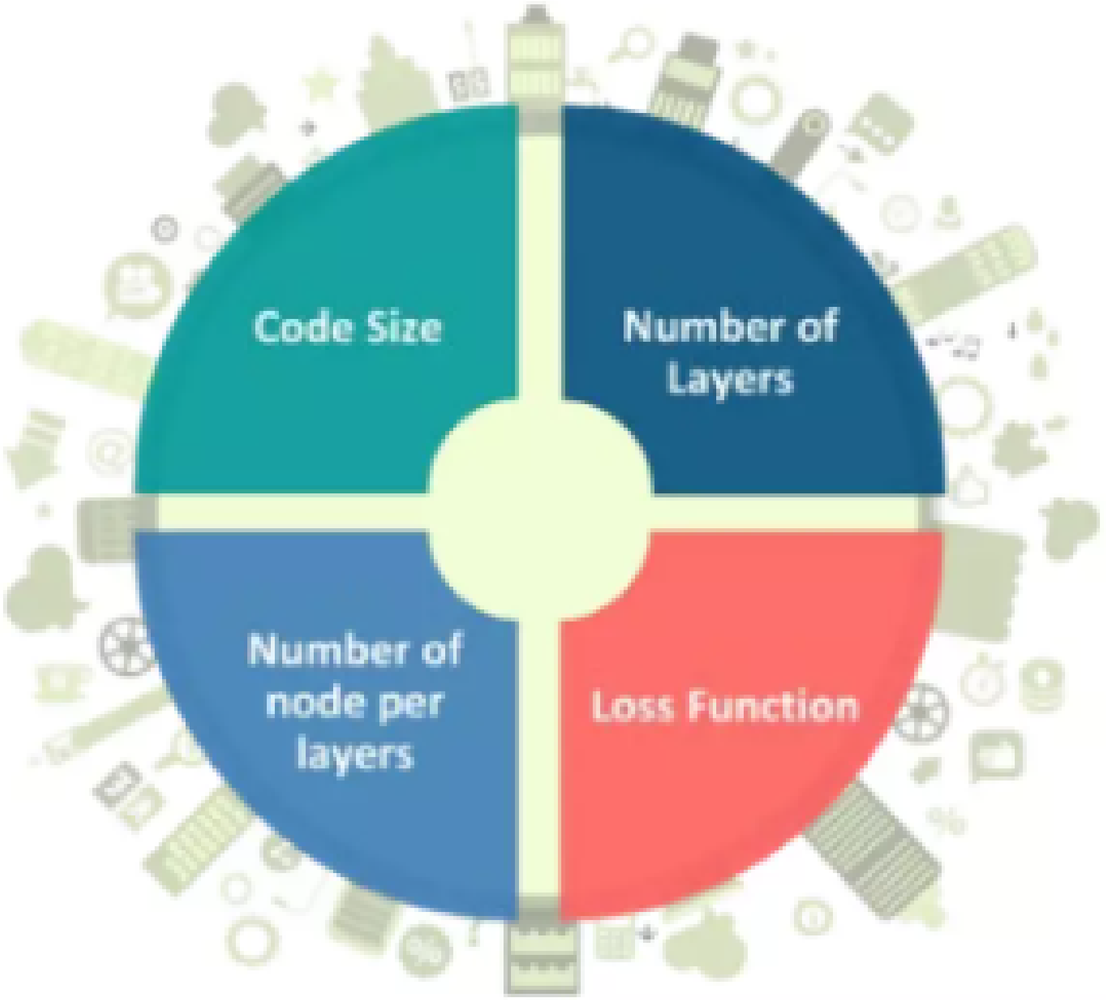


Cách thức hoạt động



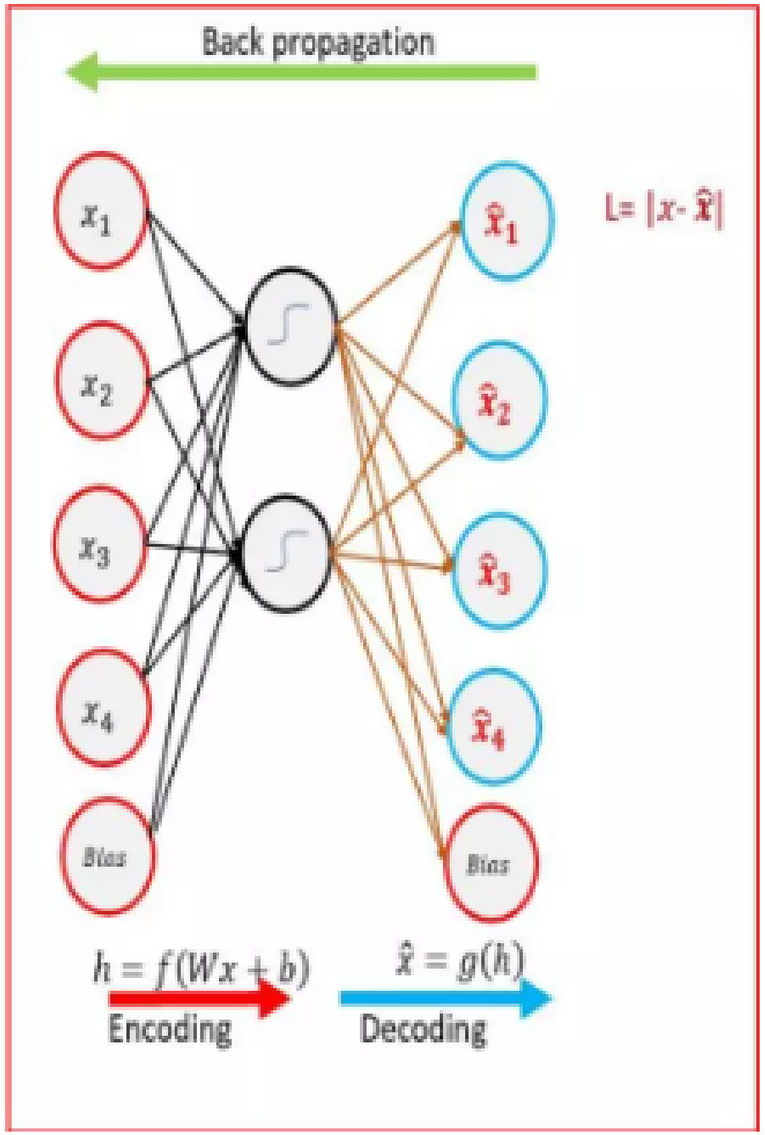
	Product1	Product2	Product3	Product4	Product5	Product6
Customer1	1	1	0	0	0	0
Customer2	1	0	0	0	1	0
Customer3	0	1	1	1	0	0
Customer4	1	1	0	0	1	0
Customer5	0	0	0	1	0	1

Hyperparamaters

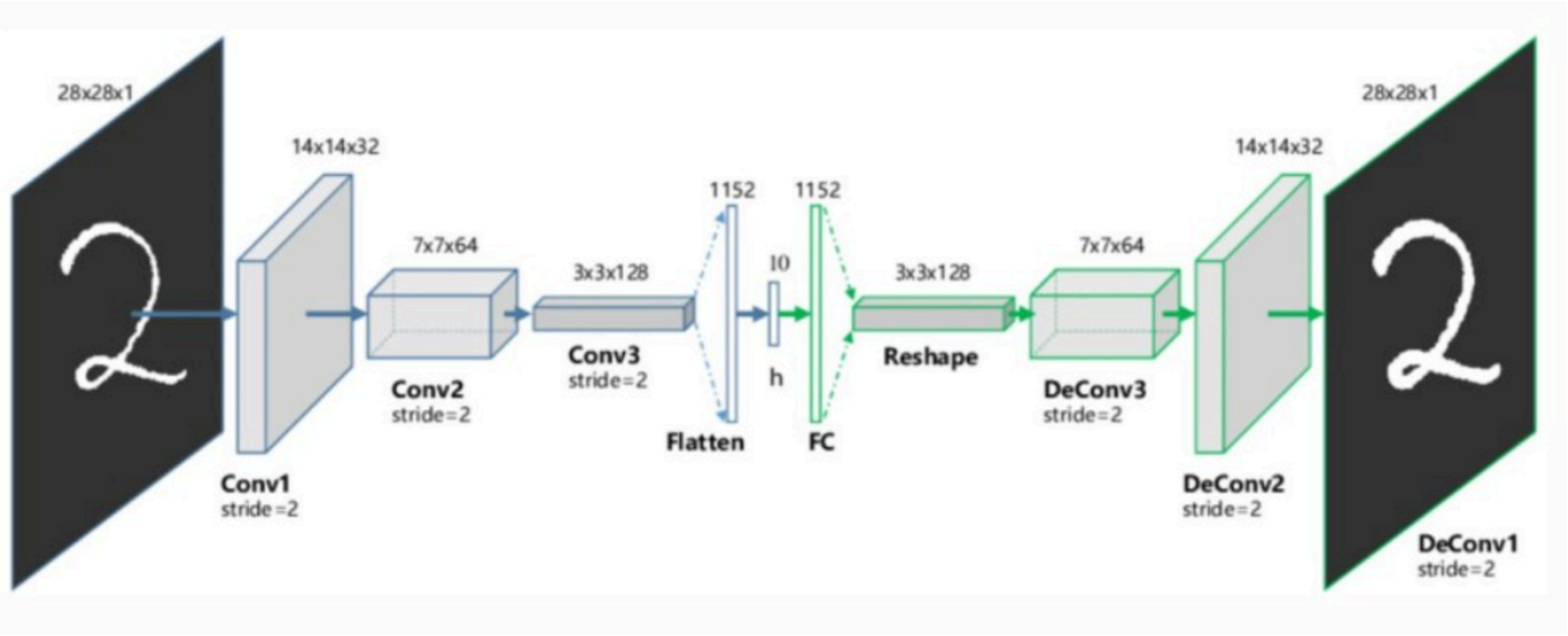


Undercomplete Autoencoder

- Hidden layer nhỏ hơn input.
- Mục tiêu: nén dữ liệu, giữ đặc trưng quan trọng.
- Gần giống PCA nhưng có thể mở rộng phi tuyến



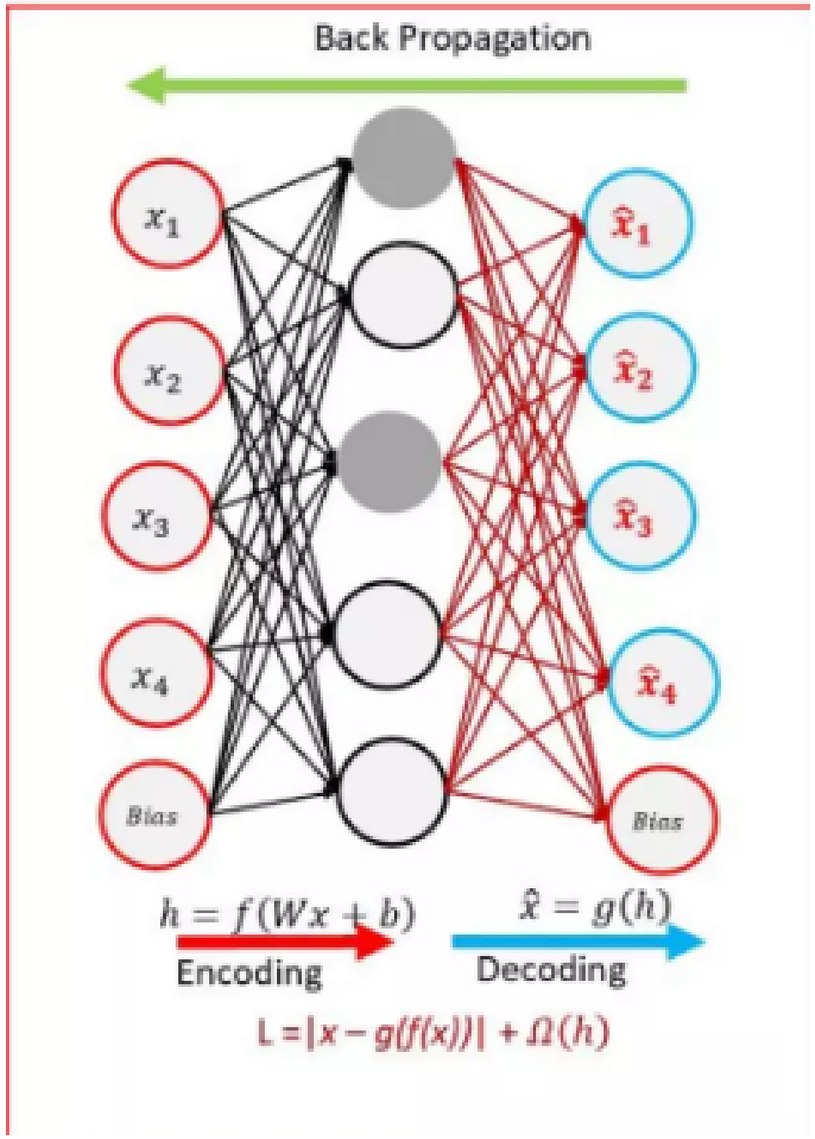
Convolutional Autoencoder (CAE)



- Sử dụng convolution thay vì fully connected.
- Hiệu quả cho dữ liệu ảnh, video.
- Ứng dụng: tái tạo ảnh, tô màu, tạo ảnh độ phân giải cao.

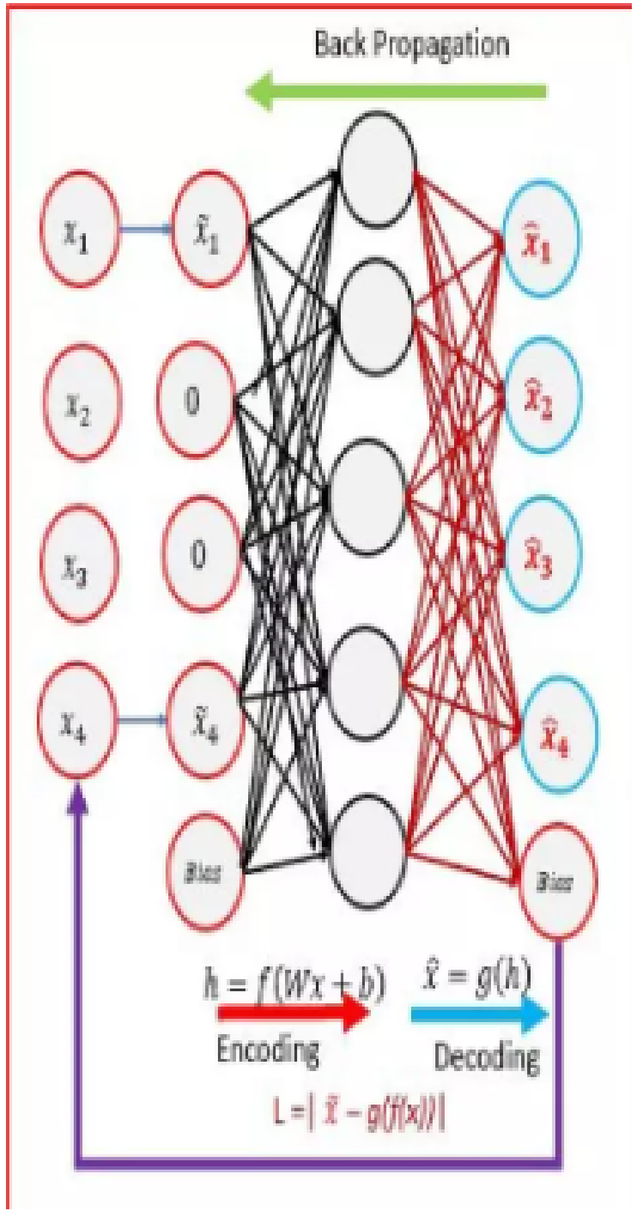
Sparse Autoencoder

- Hidden layer có nhiều node nhưng thêm ràng buộc “sparsity” → chỉ một số ít node kích hoạt.
- Giúp trích xuất đặc trưng tốt, tránh overfitting.



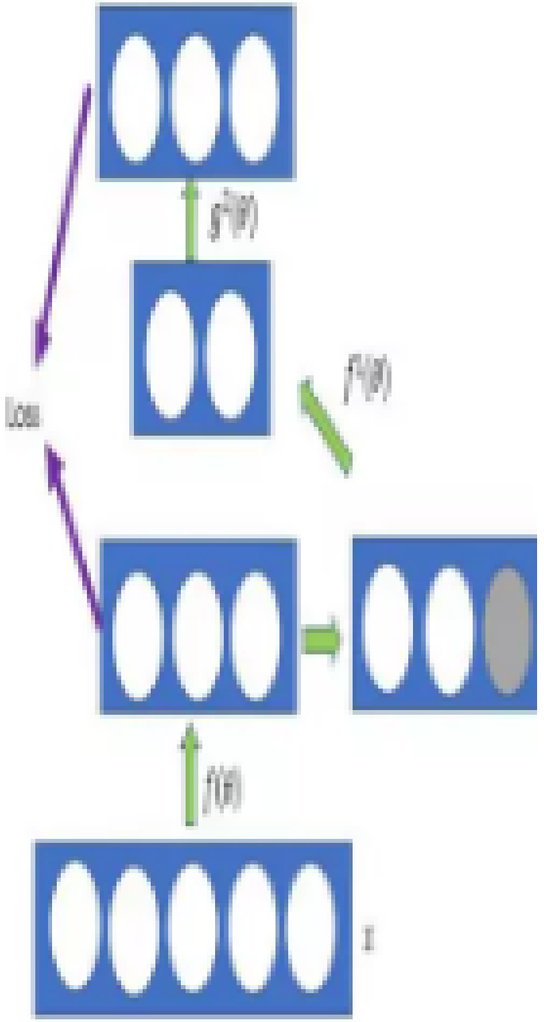
Denoising Autoencoder (DAE)

- Thêm nhiễu vào input, mô hình học cách tái tạo lại dữ liệu sạch.
- Giúp mô hình học đặc trưng mạnh mẽ, chống nhiễu.



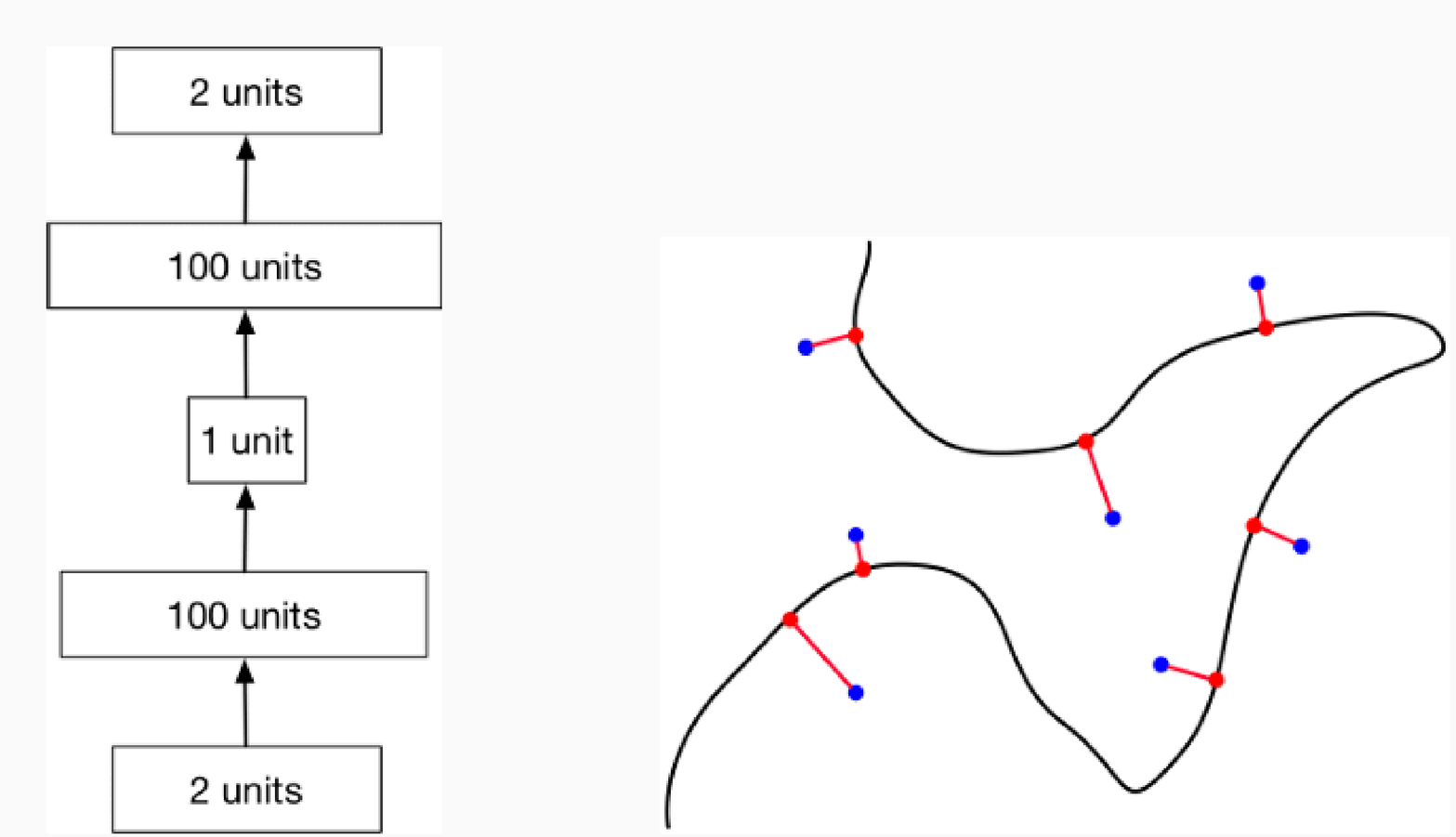
Stacked Denoising Autoencoder

- Nhiều tầng Denoising Autoencoder xếp chồng.
- Học đặc trưng phức tạp hơn, có thể dùng làm input cho các mô hình supervised khác.
- Huấn luyện thường theo kiểu layerwise pre-training rồi fine-tune bằng backprop.



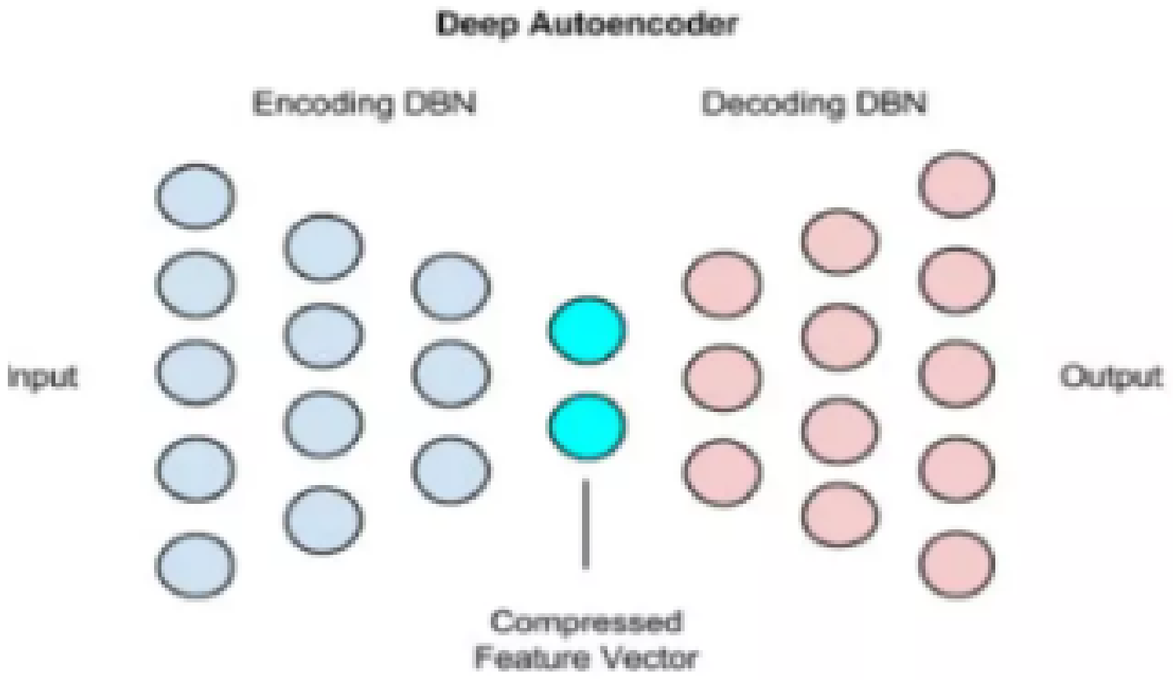
Deep autoender

Nonlinear autoencoder có thể học được mã hóa mạnh mẽ hơn so với **linear autoencoder** - tức PCA khi cùng một số chiều (dimensionality).



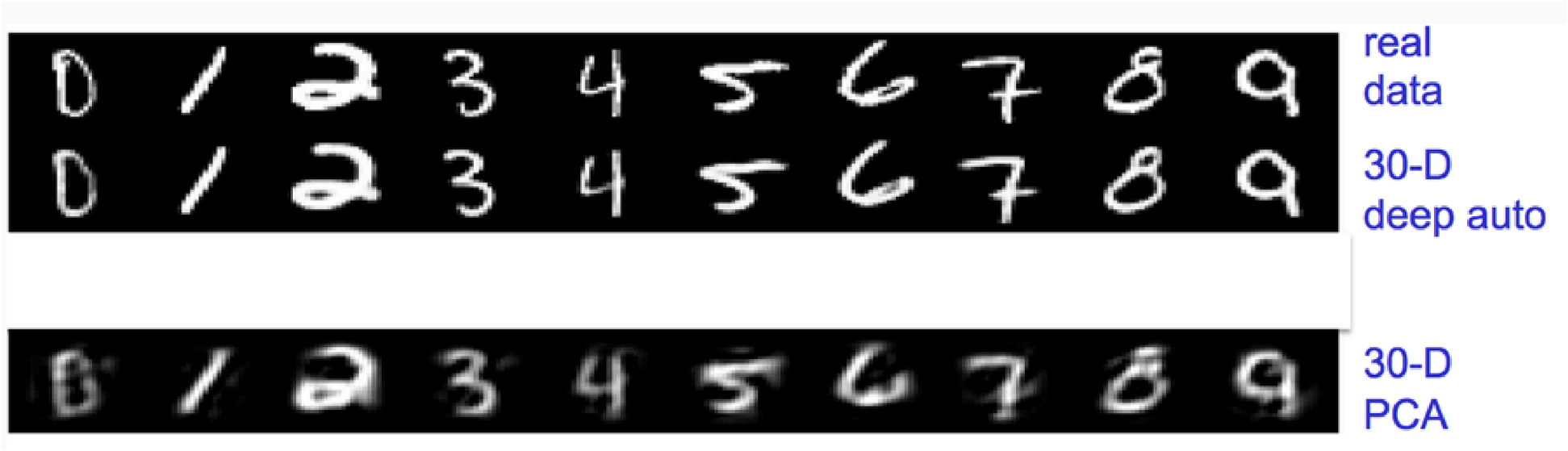
Deep Autoencoder

- Gồm nhiều tầng encoder–decoder.
- Học được đặc trưng bậc cao (high-order features).
- Cách huấn luyện: có thể pre-train từng tầng bằng RBM hoặc DAE, sau đó fine-tune end-to-end. Ngày nay thường train trực tiếp với Adam + regularization.
- Ứng dụng: tìm kiếm ảnh, nén dữ liệu, topic modeling.

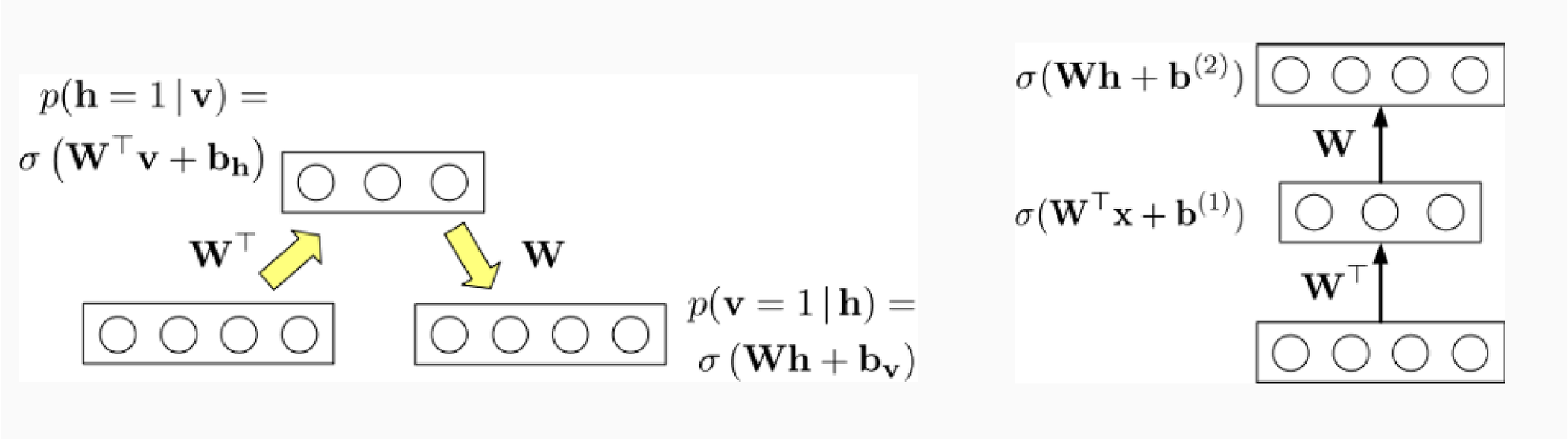


Deep autoender

Nonlinear autoencoder có thể học được mã hóa mạnh mẽ hơn so với **linear autoencoder** - tức PCA khi cùng một số chiều (dimensionality).



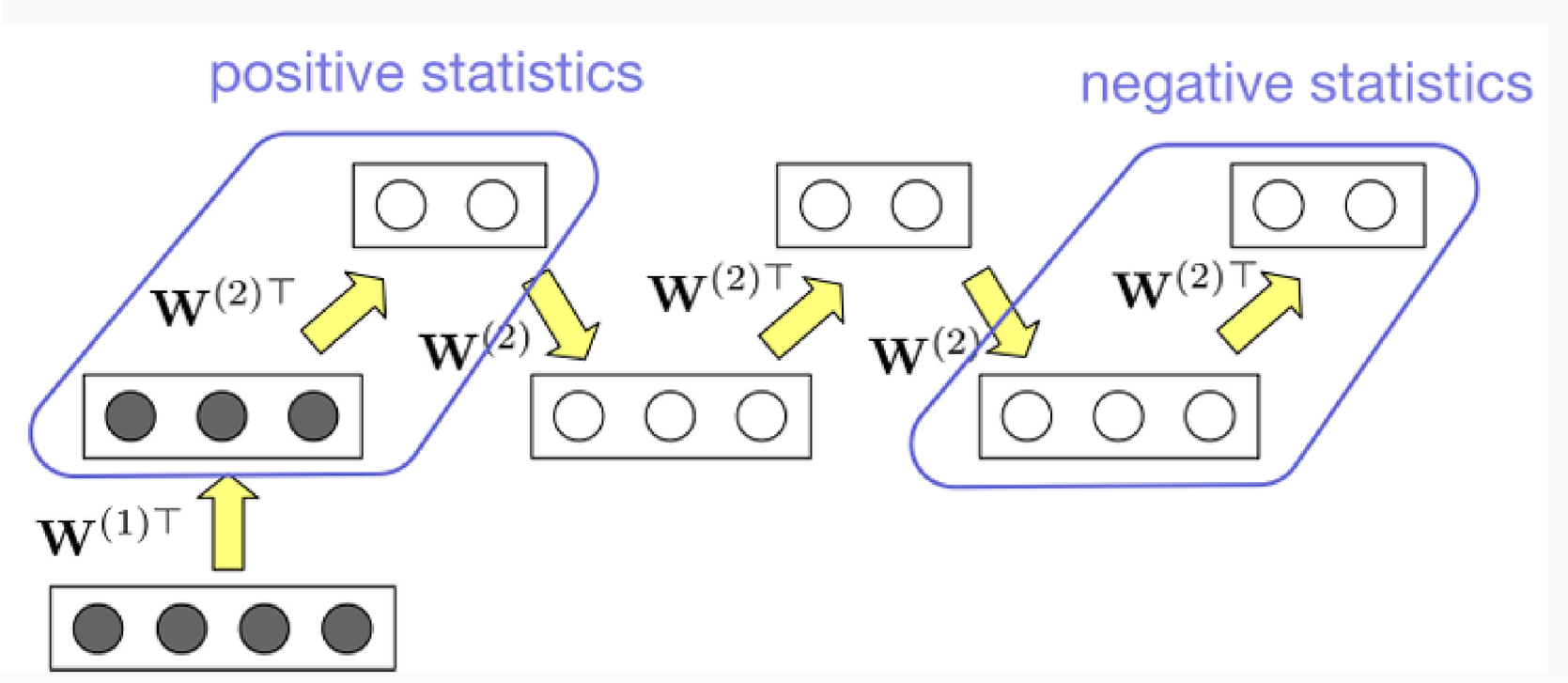
Layerwise training



RBMs(Restricted Boltzmann machine) autoencoder

- RBM là mô hình xác suất nên các nút được lấy mẫu ngẫu nhiên
- Autoencoder thì tính toán trực tiếp.
- RBM từng được dùng để huấn luyện từng layer của Deep Autoencoder theo kiểu layerwise training.”

Layerwise training

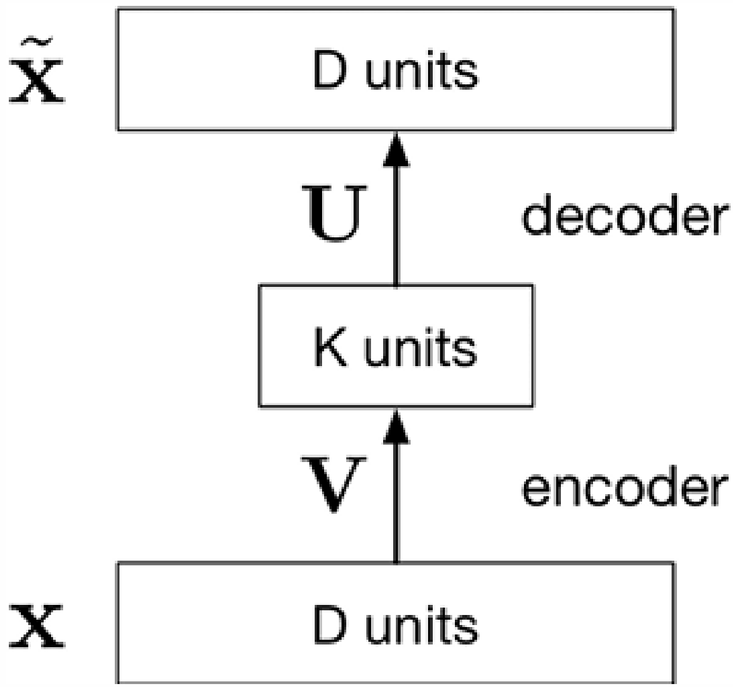


- Thay vì huấn luyện toàn bộ Deep Autoencoder một lúc (khó hội tụ), ta huấn luyện từng RBM riêng lẻ, từ dưới lên trên.
- Mỗi tầng học được biểu diễn ẩn, sau đó cung cấp dữ liệu cho tầng tiếp theo.

Principal Component Analysis (PCA)

Trong linear autoencoder

- 1 tầng ẩn
- hàm kích hoạt tuyến tính
- hàm loss $\mathcal{L}(\mathbf{x}, \tilde{\mathbf{x}}) = \|\mathbf{x} - \tilde{\mathbf{x}}\|^2$



Cách thức hoạt động :

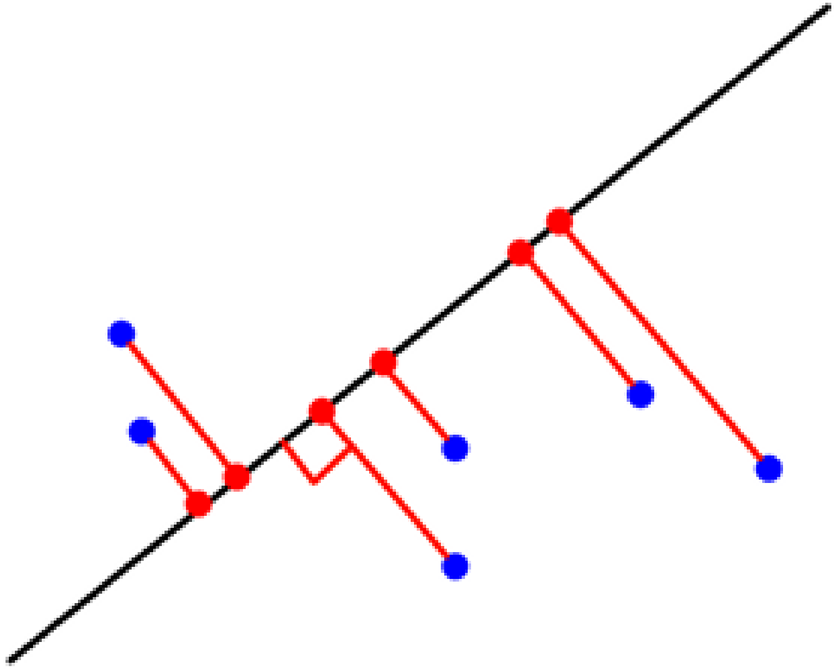
- Học tái tạo dữ liệu gốc $\tilde{\mathbf{x}} = \mathbf{UVx}$
- K >= D => Ma trận UV =1
- K<D => Ma trận V ánh xạ x từ không gian D chiều sang không gian K chiều => giảm chiều
- Đầu ra x~ nằm trong không gian con K-chiều(subspace) sinh ra bởi các cột của ma trận U.

Principal Component Analysis (PCA)

Mục tiêu : Giữ thông tin gốc nhiều nhất có thể, ta cần chọn điểm trong S gần với x nhất.

Mapping (cách biến đổi không gian) nào là tốt nhất mà nó có thể chọn?

P**rojection** (phép chiếu vuông góc) của x lên S chính là điểm trong S gần x nhất, tức là tối thiểu hóa khoảng cách đến x.



Trong linear autoencoder có thể biểu diễn phép chiếu S bằng cách

Chọn U=Q , V=Q^T , Q là cơ sở trực chuẩn cho subspace S

Principal Component Analysis (PCA)

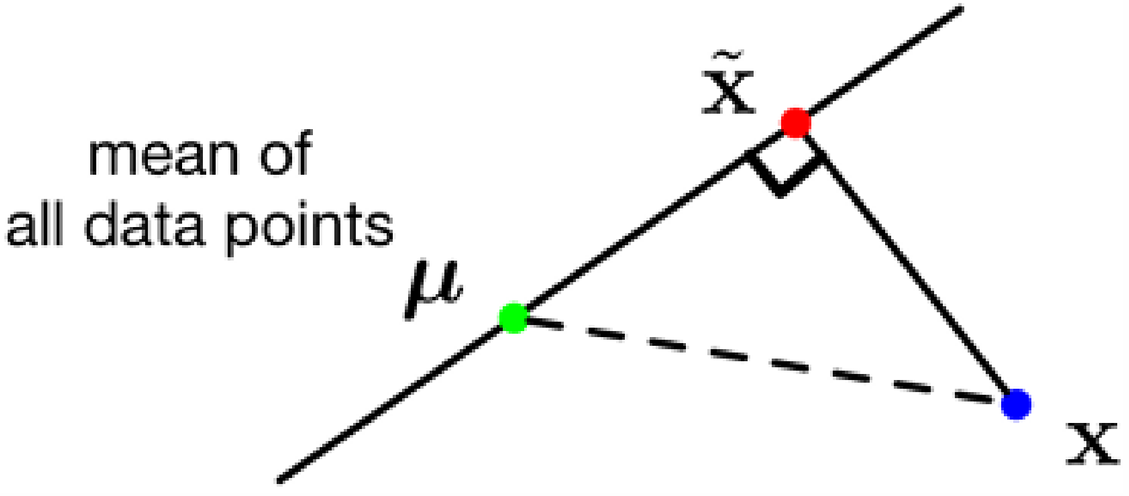
Mục tiêu autoencoder

- Học ra một subspace sao cho khoảng cách bình phương từ dữ liệu gốc x đến điểm chiếu x~ là nhỏ nhất.
- Tức là chọn subspace mà ở đó phương sai (variance) của các điểm chiếu là lớn nhất.

By the Pythagorean Theorem,

$$\underbrace{\frac{1}{N}\sum_{i=1}^{N}\|\tilde{\mathbf{x}}^{(i)}-\boldsymbol{\mu}\|^2}_{\text{projected variance}}+\underbrace{\frac{1}{N}\sum_{i=1}^{N}\|\mathbf{x}^{(i)}-\tilde{\mathbf{x}}^{(i)}\|^2}_{\text{reconstruction error}}$$

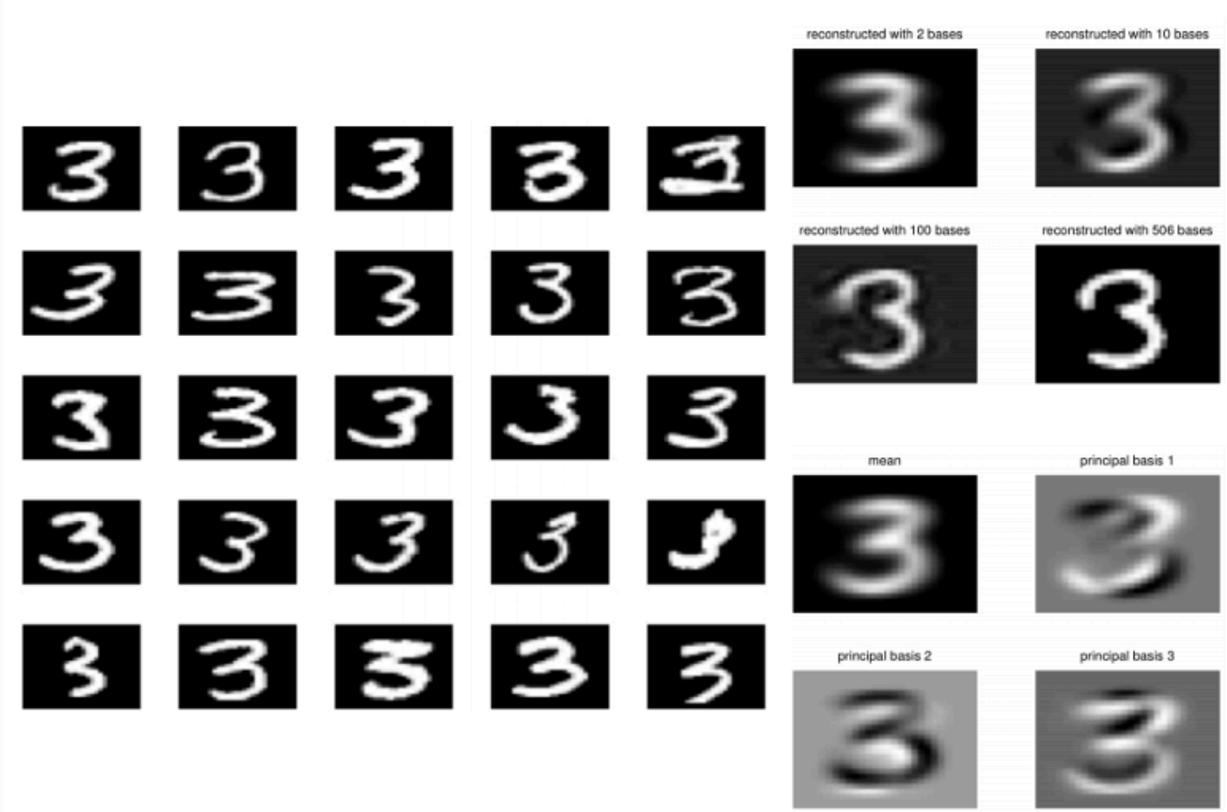
$$=\underbrace{\frac{1}{N}\sum_{i=1}^{N}\|\mathbf{x}^{(i)}-\boldsymbol{\mu}\|^2}_{\text{constant}}$$



Principal Component Analysis (PCA)

PCA for faces “Eigenfaces”

PCA for digits “Eigendigits”



MNIST

ĐẠI HỌC TÔN ĐỨC THẮNG
TON DUC THANG UNIVERSITY

Thank you