DEEP NERAL NETWOK FOR IMAGE

GVHD: ThS Nguyễn Ngọc Đức

MEMBER OF GROUP

MEMBER

1. Nguyễn Tấn Phát - 20127588

2. Lê Ngọc Tường - 20127383

3. Đặng Minh Đức - 20127136

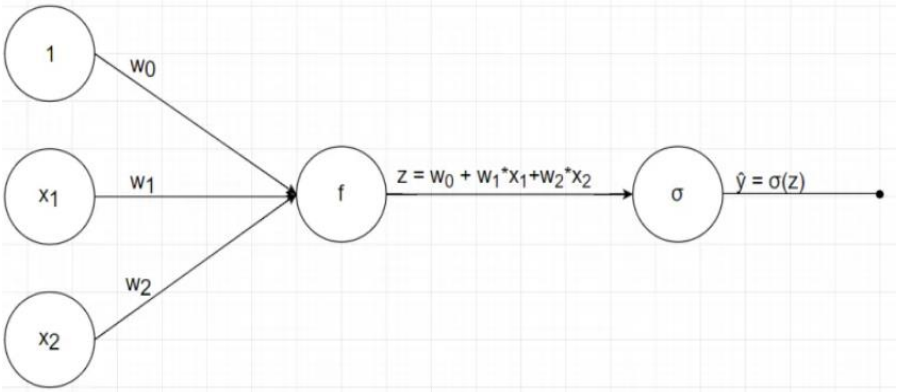
TABLE OF CONTENTS

01

MACHINE LEARNING

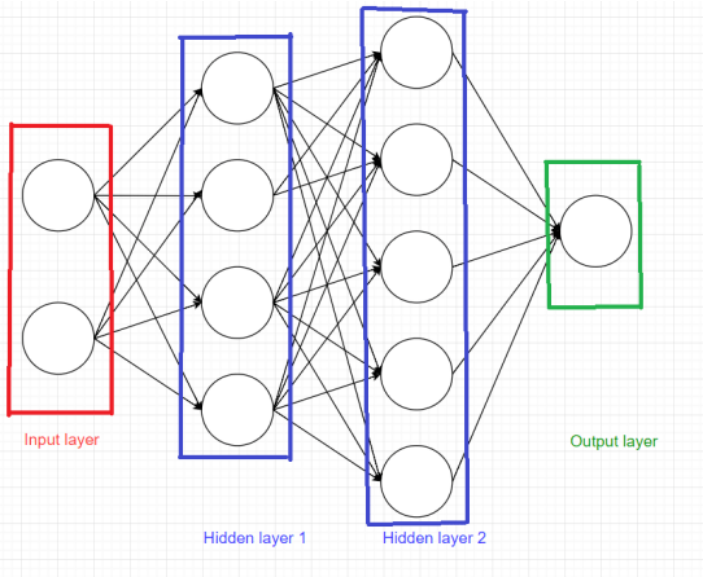
NEURAL NETWORK

NEURAL NETWORK



Logistics Regression

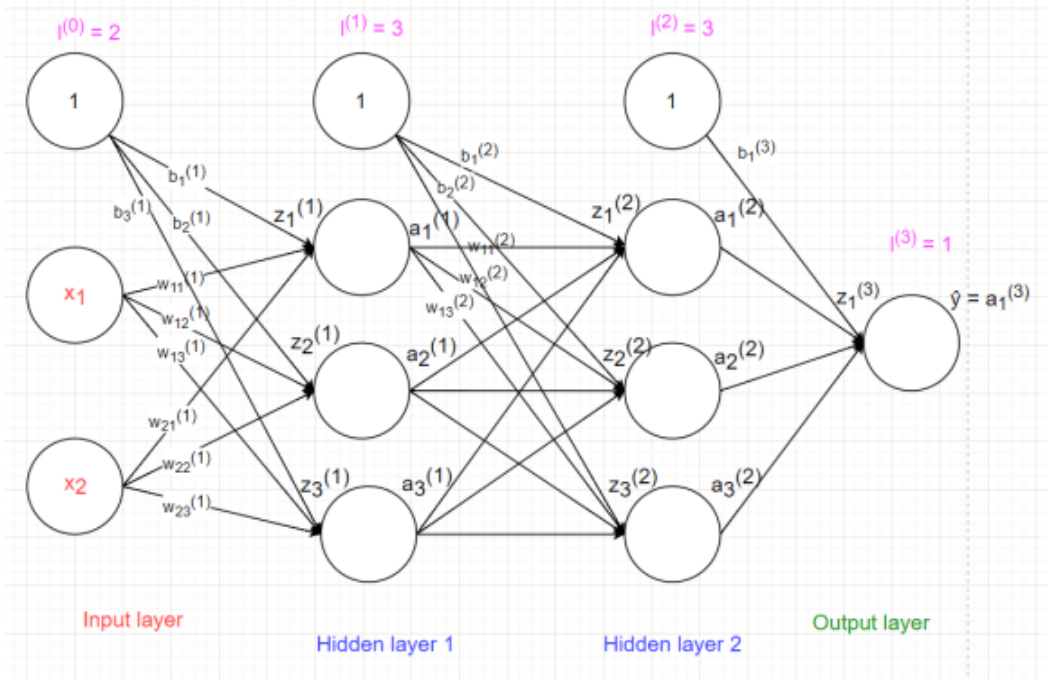
Là mô hình neural network đơn giản nhất chỉ với input layer và output layer

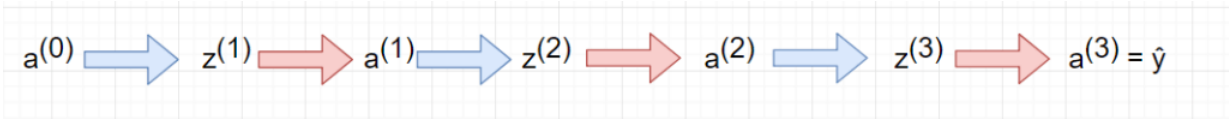


Neural Network

Tổng số layer mô hình = số layer – 1
Không tính input layer

FEED FOR WARD

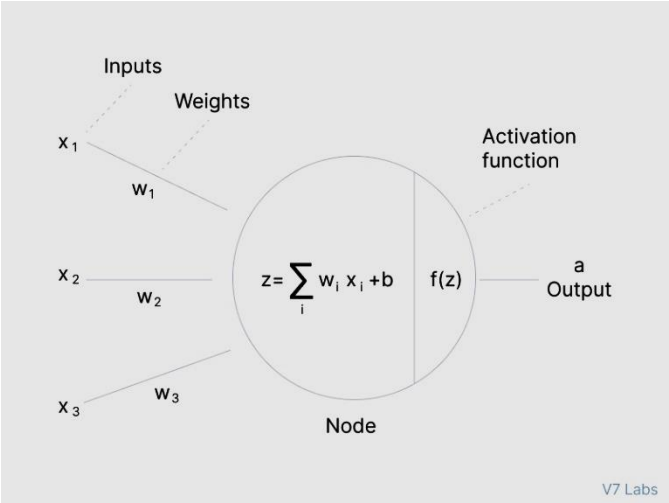




02

ACTIVATION FUNCTION

ACTIVATION FUNCTION



Áp dụng trên đầu ra các neuron tầng ẩn và làm input cho tầng tiếp theo

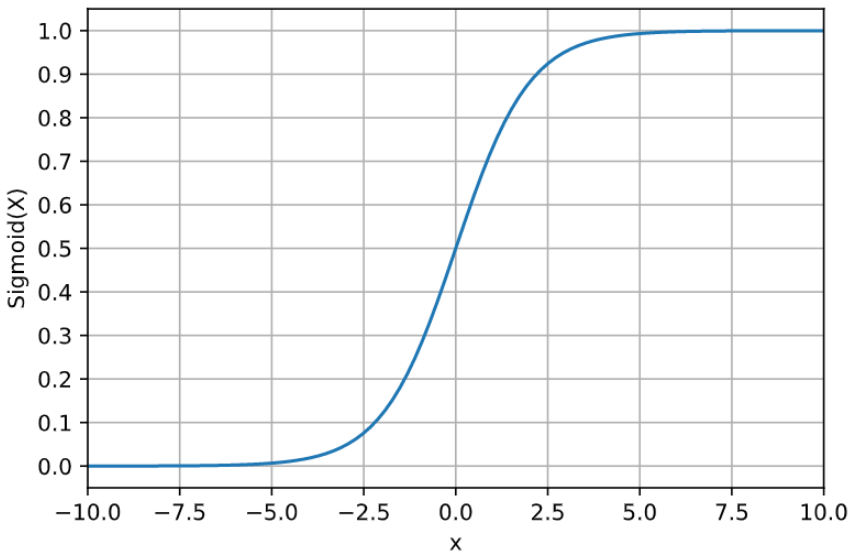
BENEFIT

Nếu không có hàm phi tuyến, mạng neuron dù có nhiều lớp cũng chỉ có ý nghĩa như 1 lớp tuyến tính

Hàm phi tuyến giúp mô hình học được các quan hệ phức tạp và tiềm ẩn

HÀM SIGMOID

$$\sigma(x) = \frac{1}{1 + e^{-x}} \rightarrow \sigma'(x) = \sigma(x) * (1 - \sigma(x))$$

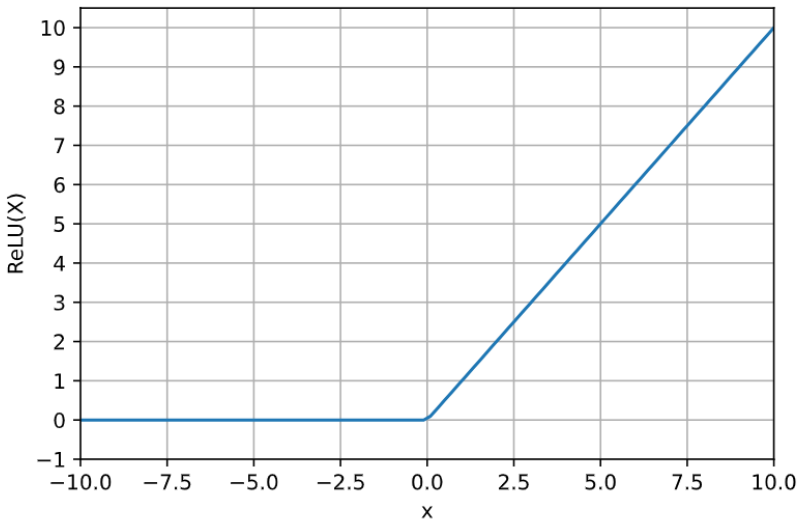


Tiệm cận ở hai đầu và nằm trong khoảng (0, 1):

- Dễ tính đạo hàm.
- Bão hòa và triệt tiêu gradient ở 2 đầu.
- Không có trung tâm là 0, gây khó khăn cho hội tụ.

HÀM RELU

$$f(x) = max(0, x) \rightarrow f'(x) = \begin{cases} 0 \; if \; x < 0 \\ 1 \; if \; x > 0 \end{cases}$$

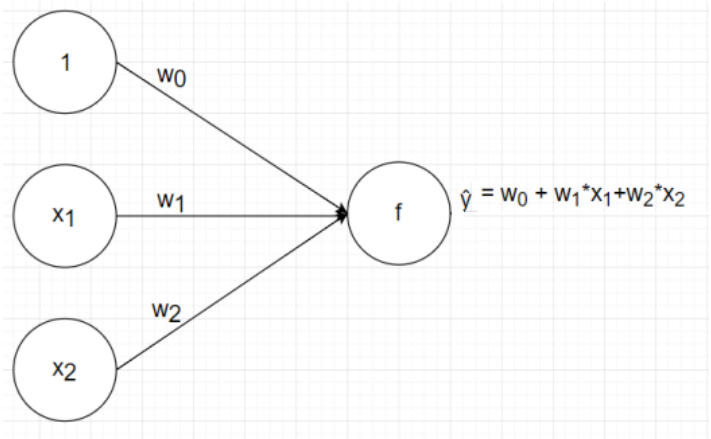


- Tính toán nhanh.
- Tốc độ hội tụ nhanh, không bị bão hòa ở 2 đầu.

GRADIENT DESCENT - LINEAR REGRESSION

Nhằm mục đích tính xấp xỉ W để loss function đạt giá trị thấp nhất,
từ đó ta thực hiện Gradient Descent

Ví dụ mô hình linear regression:



GRADIENT DESCENT - LINEAR REGRESSION

Hàm lỗi là hàm bậc 2, dạng parabol:

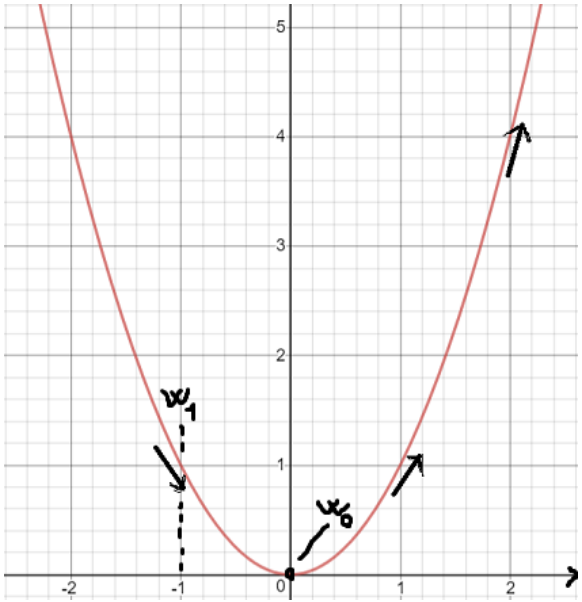
$$J = \frac{1}{2N} * (\sum_{i=1}^{N} (\hat{y}_i - y_i)^2)$$

Và ta cần tìm w_0, w_1, w_2

Từ hình bên ta có thể thấy:

$$w_1 = -1: \ J(w_1)' < 0$$

Tại đây hàm loss giảm và ta cần tăng $w_1, w_2 = w_1 + \varepsilon$ còn nếu loss > 0 thì ta giảm w_1 đến khi đến gần cực trị



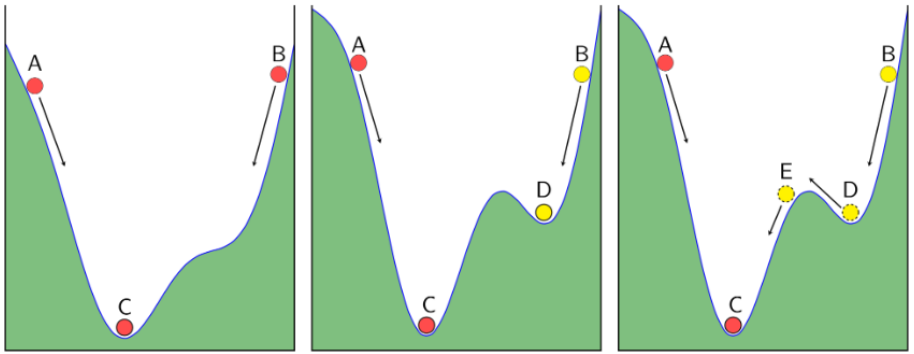
GRADIENT DESCENT - MOMENTUM

Tuy vậy điều này chưa tối ưu bởi điểm tìm được chỉ là cực tiểu, không phải giá trị nhỏ nhất.

Một trong các giải pháp là sử dụng momentum.

Thay vì thay đổi w chỉ phụ thuộc vào gradient, trọng số mới còn được tính đựa trên vận tốc (đà) trước đó:

$$v_t = \gamma v_{t-1} + \epsilon \nabla_w J(w) \ \ (v_0 = 0, \gamma \sim 0.9)$$
$$\rightarrow \quad w_{t+1} = w_t - v_t$$

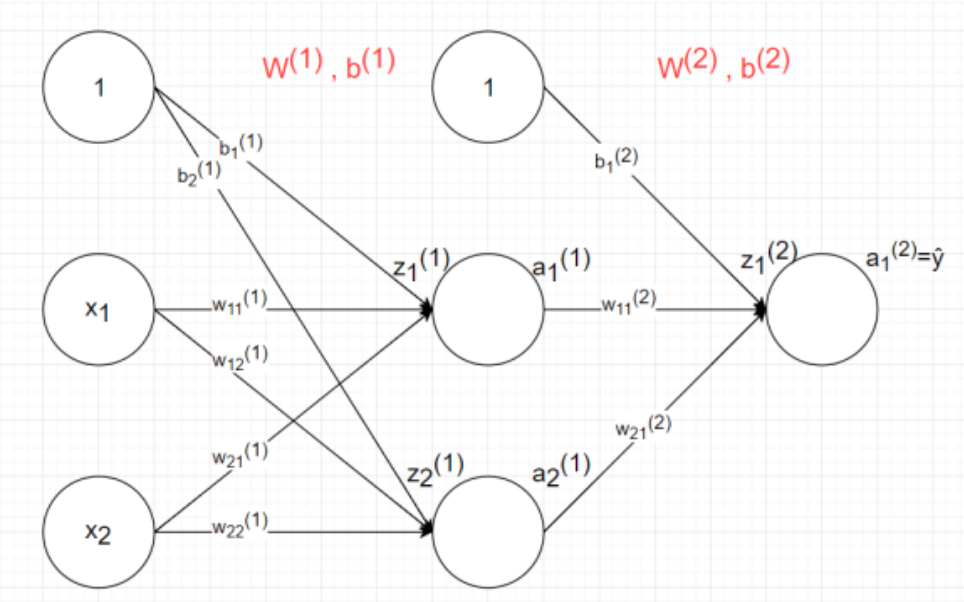


Từ hình c, ta có thể thấy thay vì dừng ở D, vận tốc trước đã tạo đà để vật thể chuyển động tiếp đến cực trị thấp hơn

BACKPROPAGATION

Để tính đạo hàm của hàm lỗi – loss function theo W của các lớp trong neural network, ta thực hiện backpropagation

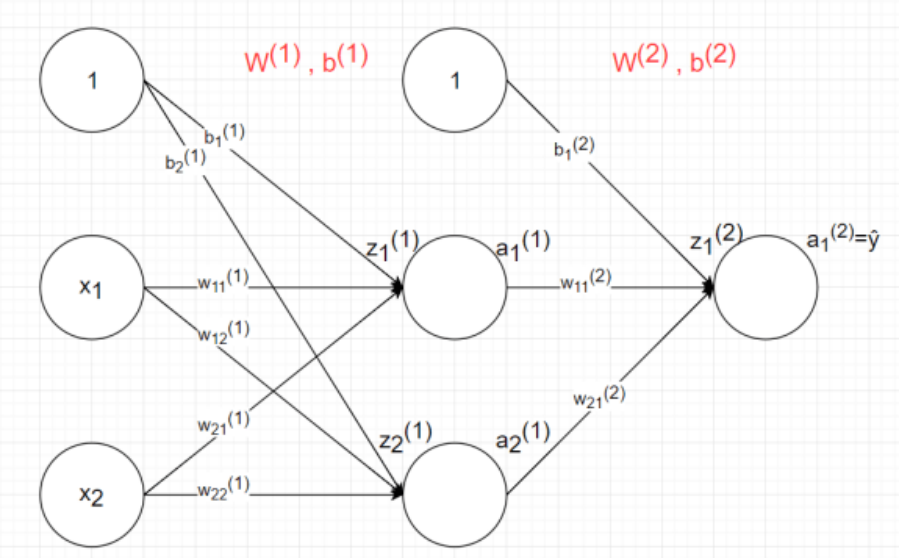
Ví dụ với mô hình đơn giản:



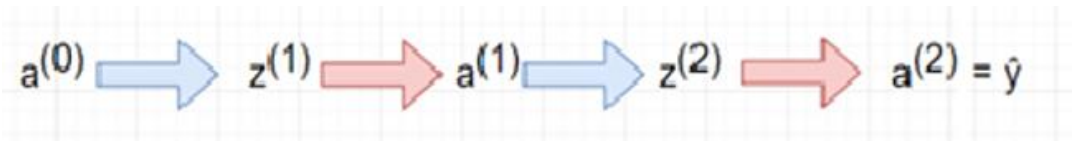
BACKPROPAGATION

Trọng số random:

- $\begin{bmatrix} 1 & w_{11}^{(1)} & w_{12}^{(1)} \\ 1 & w_{21}^{(1)} & w_{22}^{(1)} \end{bmatrix} = W^{(1)}$
- $\begin{bmatrix} 1 & w_{11}^{(2)} & w_{21}^{(2)} \end{bmatrix} = W^{(2)}$



Đầu tiên, dùng feedforward để tính các giá trị z và a cho từng lớp.



BACKPROPAGATION

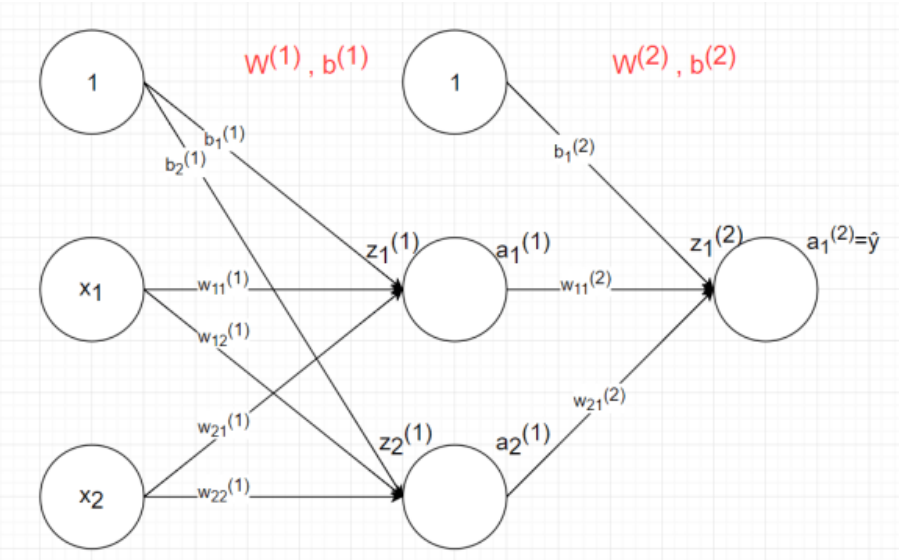
Gradient cần tính là gradient của các hệ số trên đối với hàm loss -> sử dụng chain rule.
Ta đi từ lớp output về input:

- *Ở lớp thứ 2:*

$$\frac{\partial L}{\partial w_{11}^{(2)}} = \frac{\partial L}{\partial \hat{y}_i} * \frac{\partial \hat{y}_i}{\partial z_1^{(2)}} * \frac{\partial z_1^{(2)}}{\partial w_{11}^{(2)}} = (\hat{y}_i - y_i) * a_1^{(1)}$$

Tương tự, ta tính cho:

$$\frac{\partial L}{\partial w_{21}^{(2)}} = (\hat{y}_i - y_i) * a_2^{(1)}, \frac{\partial L}{\partial b_1^{(2)}} = (\hat{y}_i - y_i)$$



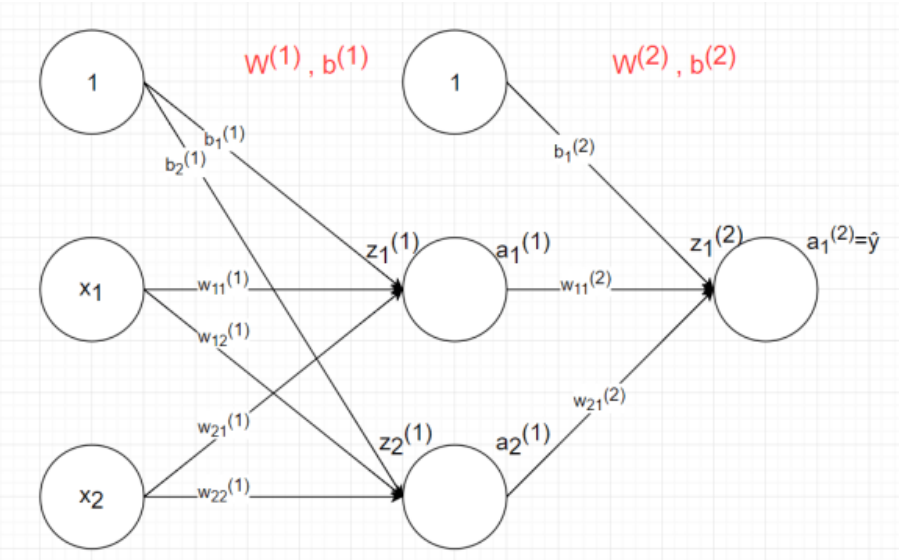
$$L = Loss(\hat{y})$$

BACKPROPAGATION

- *Ở lớp thứ 1:* $$\frac{\partial L}{\partial w_{11}^{(1)}} = \frac{\partial L}{\partial \hat{y}_i} * \frac{\partial \hat{y}_i}{\partial z_1^{(2)}} * \frac{\partial z_1^{(2)}}{\partial a_1^{(1)}} * \frac{\partial a_1^{(1)}}{\partial z_1^{(1)}} * \frac{\partial z_1^{(1)}}{\partial w_{11}^{(1)}} = (\hat{y}_i - y_i) * w_{11}^{(2)} * a_1^{(1)}.\left(1 - a_1^{(1)}\right) * x_1$$

Tương tự, ta tính cho:

$$\frac{\partial L}{\partial w_{12}^{(1)}}, \frac{\partial L}{\partial w_{12}^{(1)}}, \frac{\partial L}{\partial w_{21}^{(1)}}, \frac{\partial L}{\partial w_{22}^{(1)}}, \frac{\partial L}{\partial b_2^{(1)}}, \frac{\partial L}{\partial b_1^{(1)}}$$



$$L = Loss(\hat{y})$$

BACKPROPAGATION

Ở dạng biểu diễn ma trận ta có:

$$Z^{(1)} = X * W^{(1)} + b^{(1)} \quad \rightarrow A^{(1)} = \sigma(Z^{(1)})$$
$$Z^{(2)} = A^{(1)} * W^{(2)} + b^{(2)} \quad \rightarrow \hat{Y} = A^{(2)} = \sigma(Z^{(2)})$$
$$L = Loss(\hat{Y}) \quad \rightarrow J = \sum Loss(\hat{Y})$$

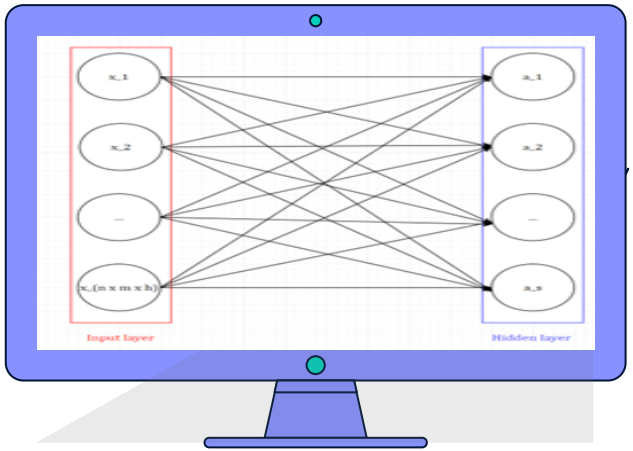
$$\frac{\partial L}{\partial W^{(2)}} = \frac{\partial L}{\partial \hat{Y}} * \frac{\partial \hat{Y}}{\partial Z^{(2)}} * \frac{\partial Z^{(2)}}{\partial W^{(2)}}$$
$$= (\hat{\boldsymbol{Y}} - \boldsymbol{Y}) * (\boldsymbol{A^{(1)}})^{\boldsymbol{T}}$$
$$\frac{\partial L}{\partial W^{(1)}} = \frac{\partial L}{\partial \hat{Y}} * \frac{\partial \hat{Y}}{\partial Z^{(2)}} * \frac{\partial Z^{(2)}}{\partial A^{(1)}} * \frac{\partial A^{(1)}}{\partial Z^{(1)}} * \frac{\partial Z^{(1)}}{W^{(1)}}$$
$$= \boldsymbol{X^T} * \left(\hat{\boldsymbol{Y}} - \boldsymbol{Y}\right) * \left(\boldsymbol{W^{(2)}}\right)^{\boldsymbol{T}} \otimes \boldsymbol{A^{(1)}} \otimes (\boldsymbol{1} - \boldsymbol{A^{(1)}})$$

Có thể thấy, càng về gần lớp đầu, tính đạo hàm càng phức tạp, vì vậy ta nên chọn hàm phi tuyến dễ đạo hàm.

WHY IS CNN?

Giả sử ta có một bức ảnh kích thước 600 * 800 pixel, khi đó tổng số pixel của bức hình là 480,000 pixel. Nếu ta coi một pixel là một thuộc tính thì input layer sẽ có 480,000 nodes. Và lớp input sẽ được biểu diễn dưới dạng ma trận 480,000 * 1

Khi đó, nếu ở lớp ẩn đầu tiên (Hidden layer) có 1000 thuộc tính thì ma trận trọng số W sẽ có kích thước 480,000 * 1000 và sẽ có 480,000,000 phần tử.



Số lượng node của các lớp rất lớn, dẫn đến số lượng phần tử của trọng số W tăng theo cấp số nhân. Khó để kiểm soát.

Khi một bức ảnh $\boldsymbol{m} * \boldsymbol{n}$ được đưa về dạng vector $\boldsymbol{mn} * \boldsymbol{1}$ thì ta đã làm mất đi các đặc trưng về không gian của bức ảnh.

03

MACHINE LEARNING
CONVOLUTION NEURAL NETWORK

TYPES OF PHOTOS

pixel

$$\begin{bmatrix} w_{1,1} & \cdots & w_{1,m} \\ \vdots & \ddots & \vdots \\ w_{n,1} & \cdots & w_{n,m} \end{bmatrix}$$

Ảnh màu

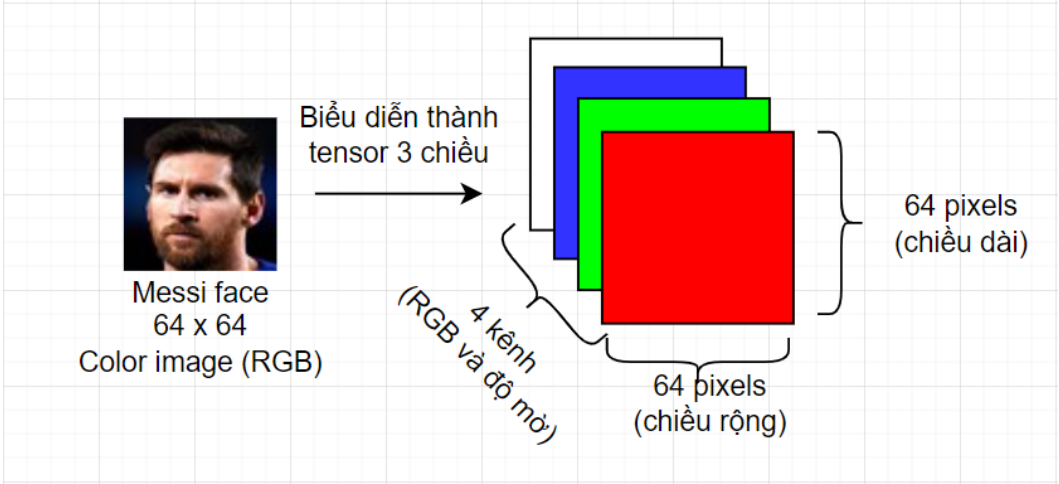
$$\begin{bmatrix} (r_{1,1}, g_{1,1}, b_{1,1}) & \cdots & (r_{1,m}, g_{1,m}, b_{1,m}) \\ \vdots & \ddots & \vdots \\ (r_{n,1}, g_{n,1}, b_{n,1}) & \cdots & (r_{n,m}, g_{n,m}, b_{n,m}) \end{bmatrix}$$

$$\begin{bmatrix} r_{1,1} & \cdots & r_{1,m} \\ \vdots & \ddots & \vdots \\ r_{n,1} & \cdots & r_{n,m} \end{bmatrix}, \begin{bmatrix} g_{1,1} & \cdots & g_{1,m} \\ \vdots & \ddots & \vdots \\ g_{n,1} & \cdots & g_{n,m} \end{bmatrix}, \begin{bmatrix} b_{1,1} & \cdots & b_{1,m} \\ \vdots & \ddots & \vdots \\ b_{n,1} & \cdots & b_{n,m} \end{bmatrix}$$

Ảnh xám

$w_{ij} \in [0, 255]$

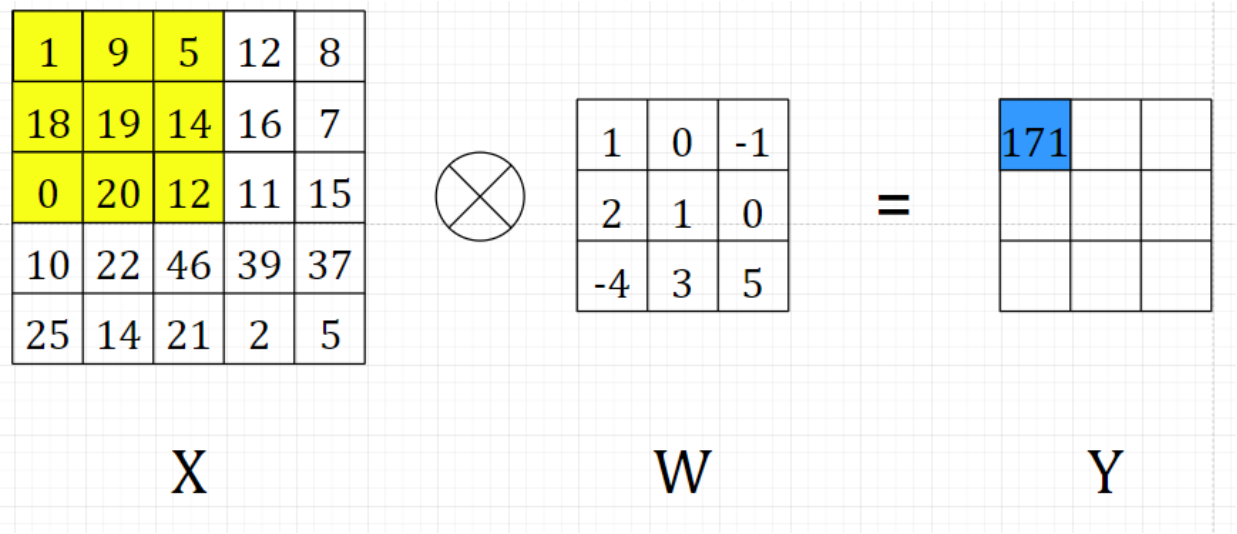
TENSOR



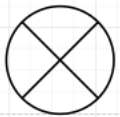
Tensor

Dữ liệu có dạng nhiều hơn 2 chiều bao gồm các ma trận xếp chồng lên nhau

PHÉP TÍNH CONVOLUTION



1	9	5
18	19	14
0	20	12



1	0	-1
2	1	0
-4	3	5

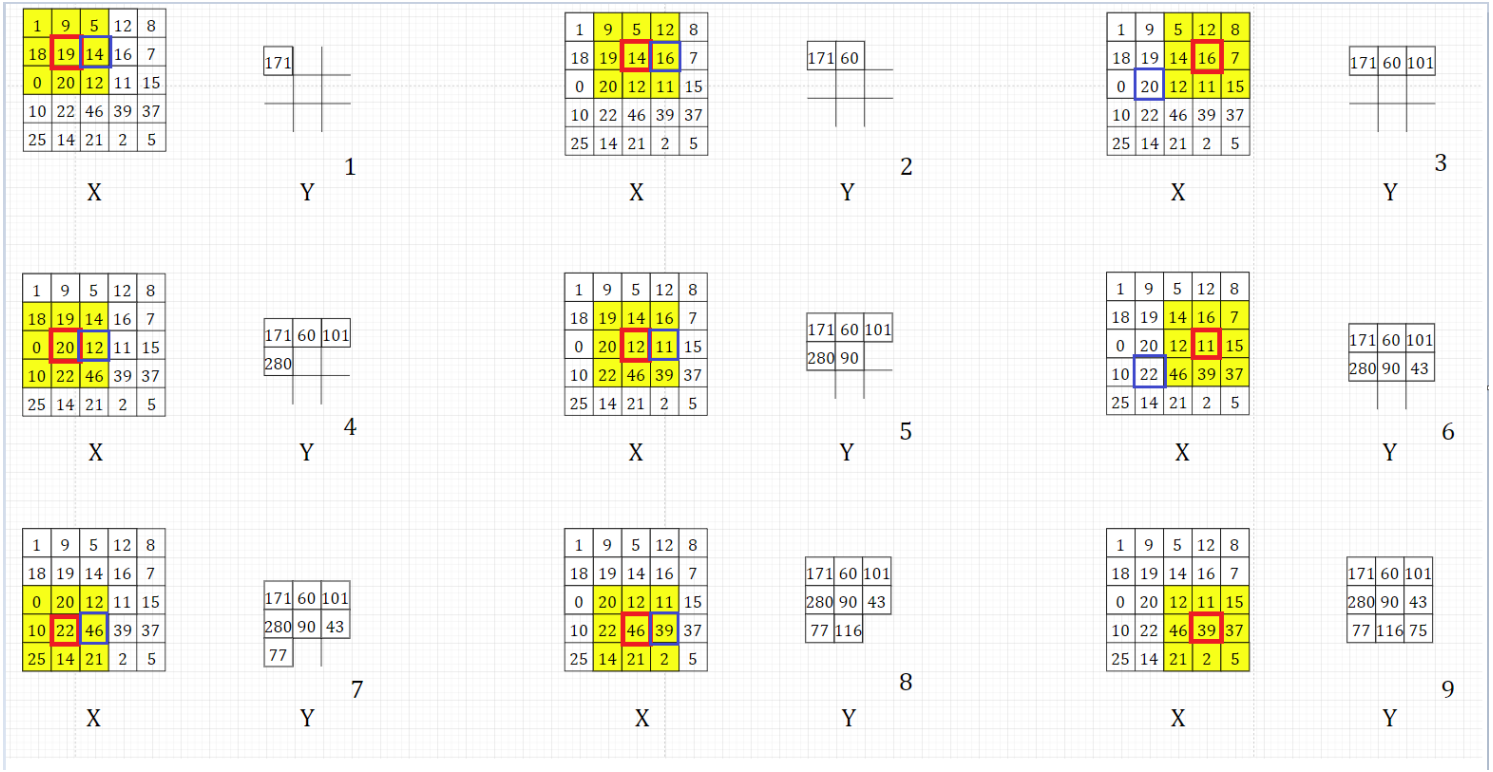
=

1	0	-5
36	19	0
0	60	60

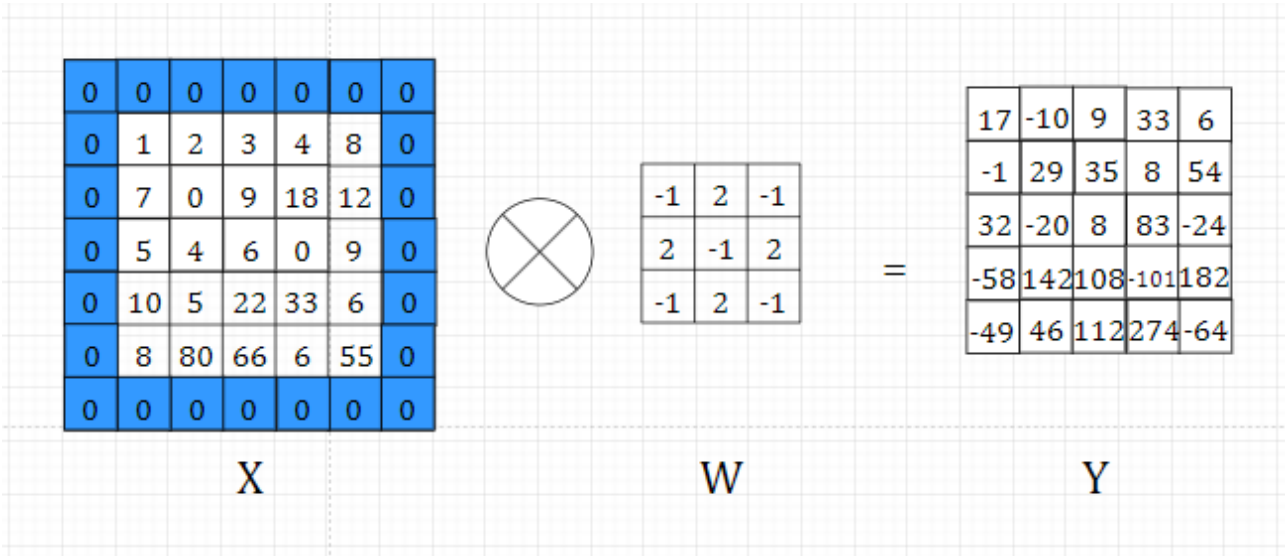
y_11 = sum (

1	0	-5
36	19	0
0	60	60

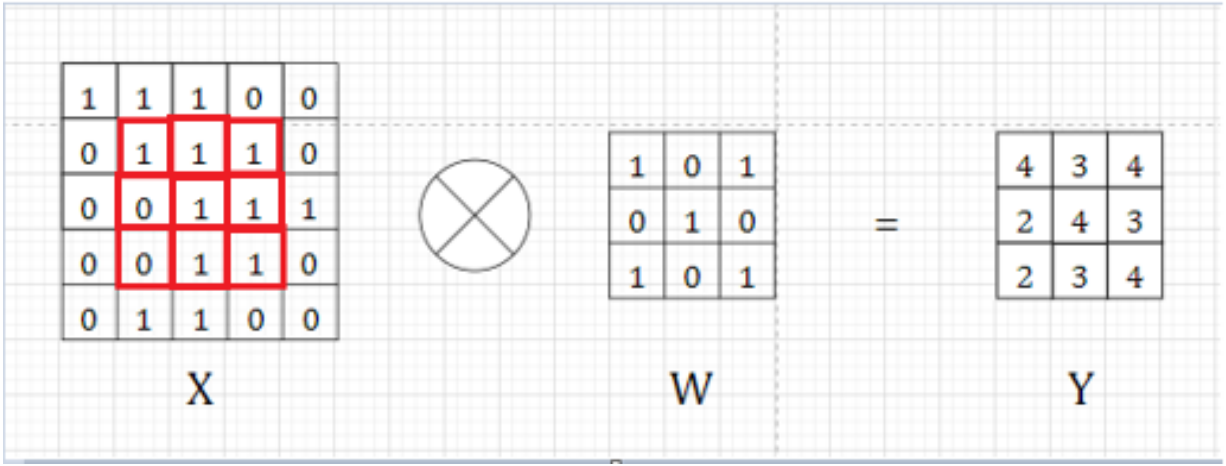
) = 1 + 0 + (-5) + 36+ 19 + 0 + 60 + 60 = 171


1 9 5 12 8
18 19 14 16 7
0 20 12 11 15
10 22 46 39 37
25 14 21 2 5
X
171
Y
1
1 9 5 12 8
18 19 14 16 7
0 20 12 11 15
10 22 46 39 37
25 14 21 2 5
X
171 60
Y
2
1 9 5 12 8
18 19 14 16 7
0 20 12 11 15
10 22 46 39 37
25 14 21 2 5
X
171 60 101
Y
3
1 9 5 12 8
18 19 14 16 7
0 20 12 11 15
10 22 46 39 37
25 14 21 2 5
X
171 60 101
280
Y
4
1 9 5 12 8
18 19 14 16 7
0 20 12 11 15
10 22 46 39 37
25 14 21 2 5
X
171 60 101
280 90
Y
5
1 9 5 12 8
18 19 14 16 7
0 20 12 11 15
10 22 46 39 37
25 14 21 2 5
X
171 60 101
280 90 43
Y
6
1 9 5 12 8
18 19 14 16 7
0 20 12 11 15
10 22 46 39 37
25 14 21 2 5
X
171 60 101
280 90 43
77
Y
7
1 9 5 12 8
18 19 14 16 7
0 20 12 11 15
10 22 46 39 37
25 14 21 2 5
X
171 60 101
280 90 43
77 116
Y
8
1 9 5 12 8
18 19 14 16 7
0 20 12 11 15
10 22 46 39 37
25 14 21 2 5
X
171 60 101
280 90 43
77 116 75
Y
9

PADDING


0 0 0 0 0 0 0
0 1 2 3 4 8 0
0 7 0 9 18 12 0
0 5 4 6 0 9 0
0 10 5 22 33 6 0
0 8 80 66 6 55 0
0 0 0 0 0 0 0
X
-1 2 -1
2 -1 2
-1 2 -1
W
=
17 -10 9 33 6
-1 29 35 8 54
32 -20 8 83 -24
-58 142 108 -101 182
-49 46 112 274 -64
Y

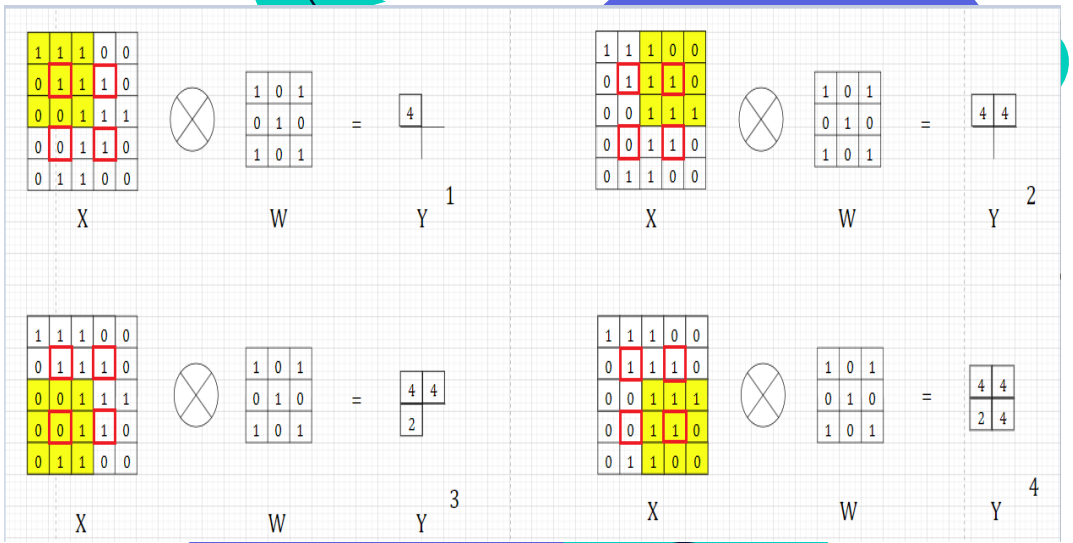
STRIDE



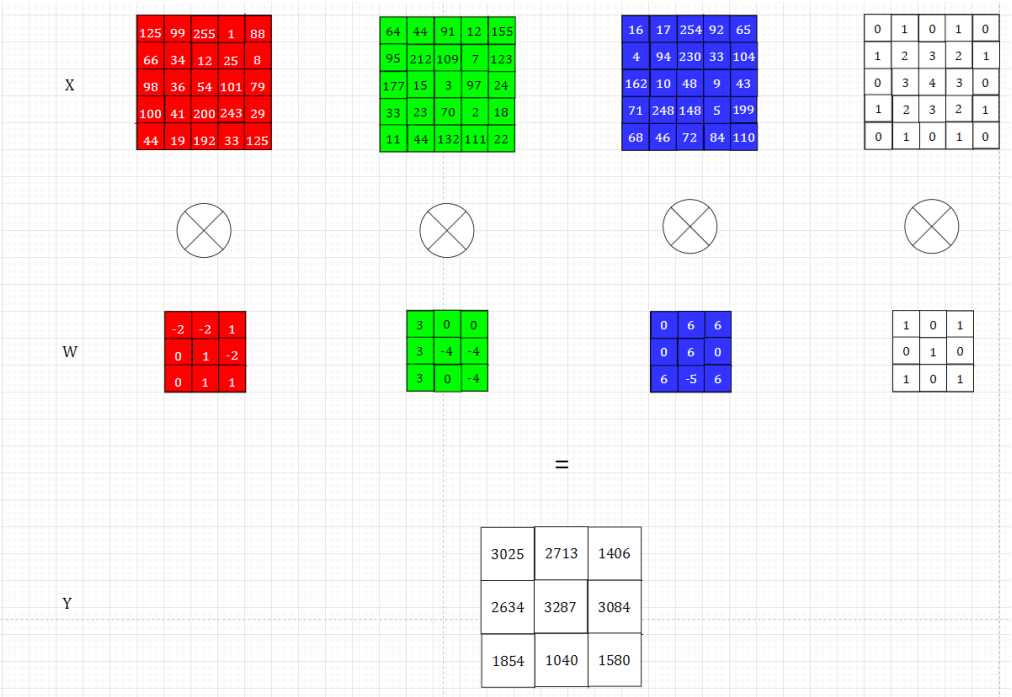
Stride = 1

Nếu stride > 1 thì ta chỉ cần thực hiện phép tính convolution trên các phần tử $x_{1+i*stride,1+j*stride}$

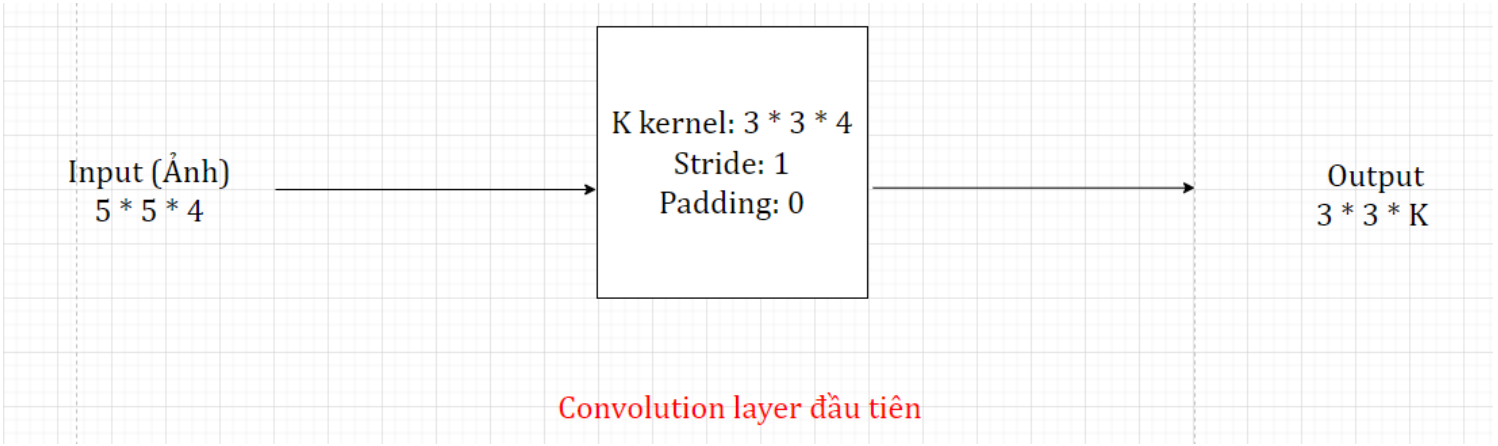
VD: Với stride = 2



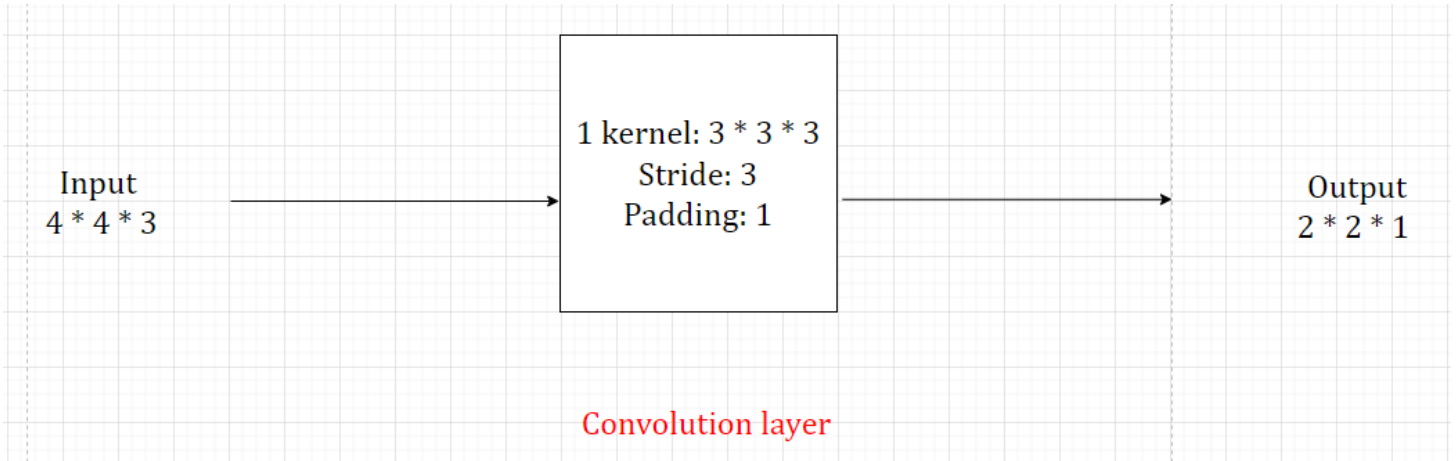
CONVOLUTIONAL LAYER - FIRST



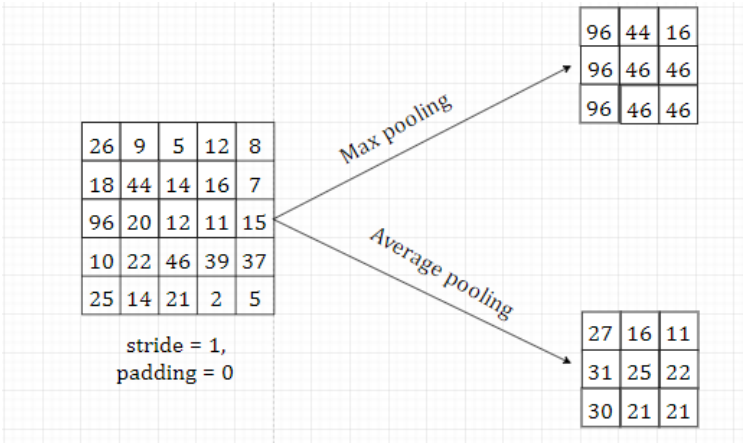
CONVOLUTIONAL LAYER - FIRST



CONVOLUTIONAL LAYER - GENERALITY



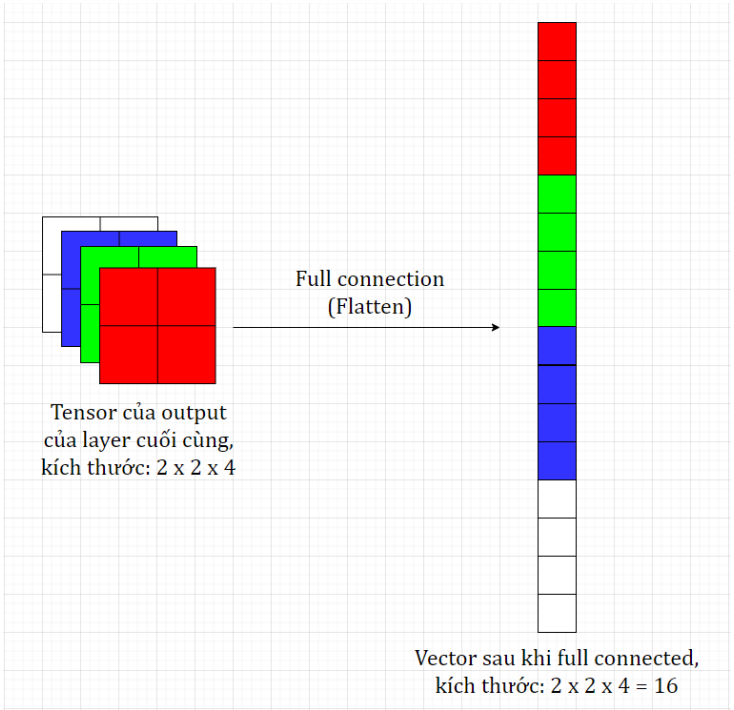
POOLING LAYER



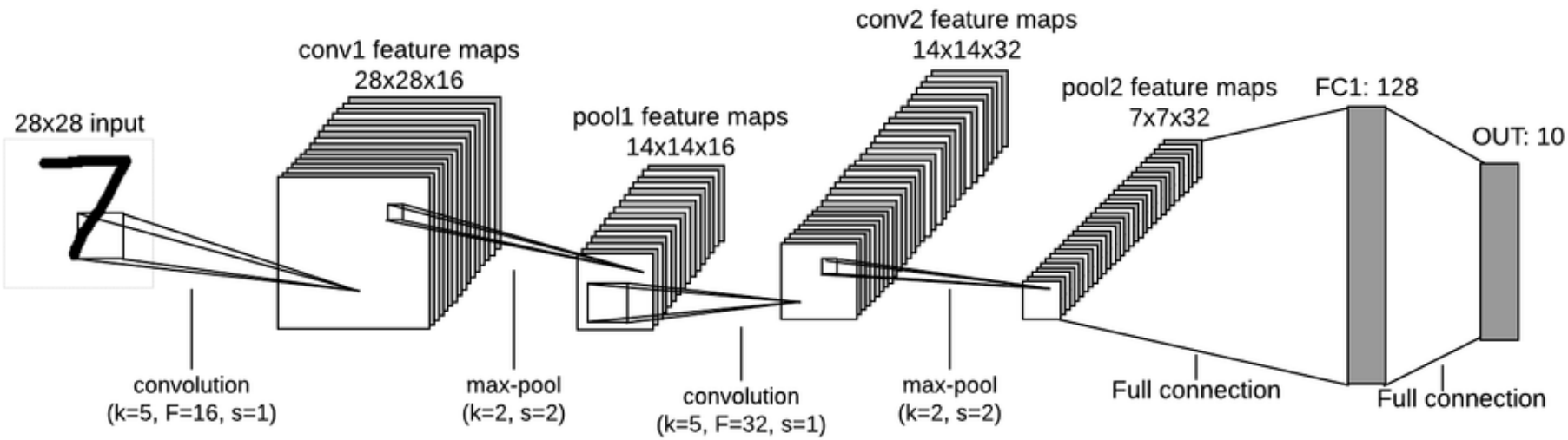
Pooling layer

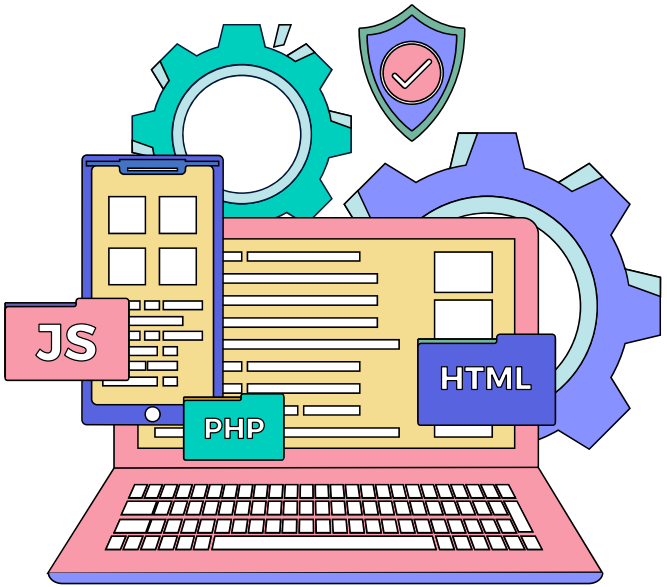
Thường được dùng giữa các convolutional layer để giảm kích thước dữ liệu nhưng vẫn giữ được các thuộc tính quan trọng.

FULLY CONNECTED LAYER



VISUALISE CONVOLUTION NEURAL NETWORK





ALEXNET

04

CNN

INTRODUCTION - ALEXNET

Mạng AlexNet đã thắng hạng nhất trong cuộc thi ILSVRC năm 2012.

Mô hình giải quyết bài toán phân lớp một bức ảnh vào lớp có 1000 node khác nhau (vd gà, chó, mèo ...).

Đầu ra của mô hình là một vector có 1000 phần tử. Phần tử thứ i của vector đại diện cho xác suất bức ảnh thuộc về lớp thứ i.

Đây là tiền đề cho sự phát triển của CNN nói riêng và Deep Neural Network nói chung

INPUT - ALEXNET

Đầu vào của mạng AlexNet là một bức ảnh RGB có kích thước 256x256 pixel. Toàn bộ các bức ảnh của tập train và tập test đều có cùng kích thước là 256x256.

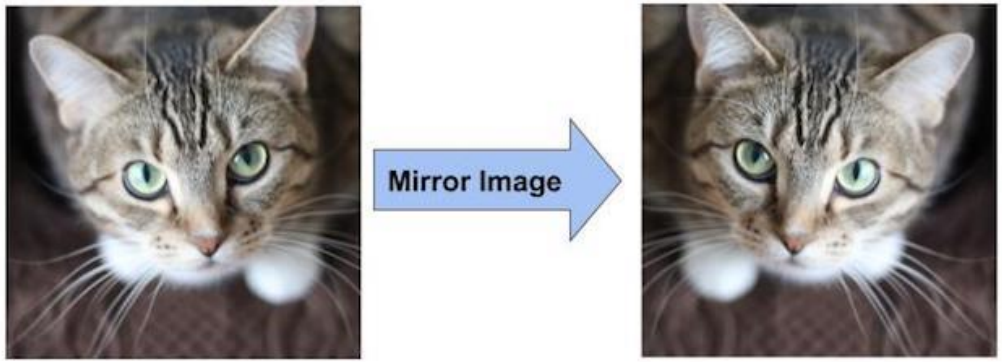
Những ảnh có kích thước nhỏ hơn 256 thì sẽ được phóng bự lên đến kích thước 256, những bức hình nào có kích thước lớn hơn thì sẽ được cắt loại phần thừa.

Nếu ảnh đầu vào là ảnh xám (grayscale), bức ảnh trên sẽ được chuyển đổi thành định dạng RGB bằng cách tạo ra 3 layer kênh màu giống nhau từ ảnh xám.

DATA AUGMENTATION- REDUCING OVERFITTING

Mirroring

. Ý tưởng của việc này là lấy ảnh trong gương của một bức hình (ảnh ảo).

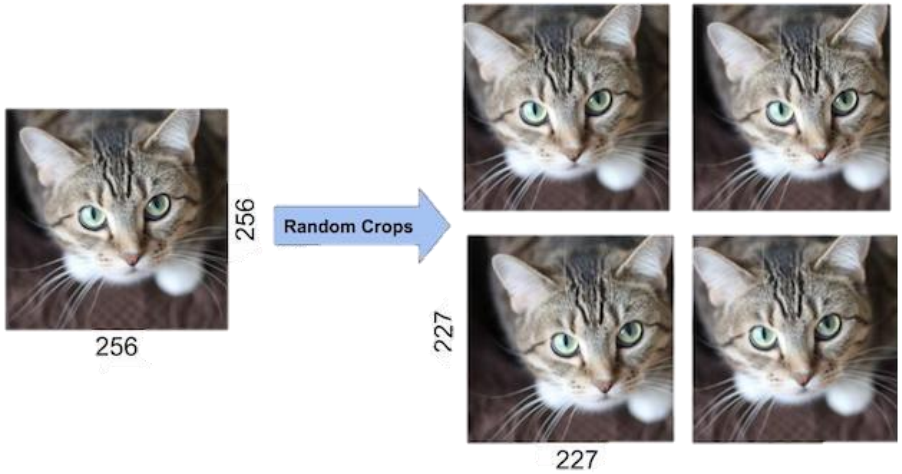


Ngoài ra cũng có thể quay một góc bất kì (180, 90, hay 12 độ); zoom ảnh; tăng độ sáng, giảm độ sáng ảnh,...

DATA AUGMENTATION- REDUCING OVERFITTING

Random Crops

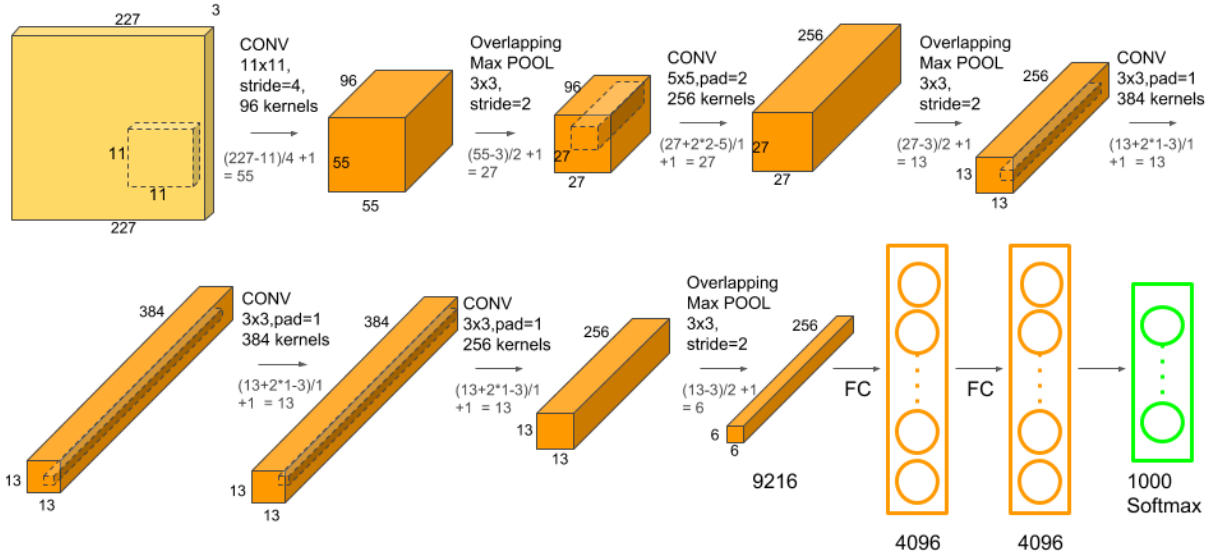
Việc lựa chọn vị trí ảnh gốc một cách ngẫu nhiên cũng giúp chúng ta có thêm một ảnh khác so với ảnh gốc ban đầu.



Nhóm tác giả của AlexNet rút trích ngẫu nhiên bức ảnh có kích thước 227x227 từ bức ảnh 256x256 ban đầu làm input dầu vào cho mô hình. Bằng cách này, chúng ta có thể tăng số lượng dữ liệu lên gấp nhiều lần.

MODEL - ALEXNET

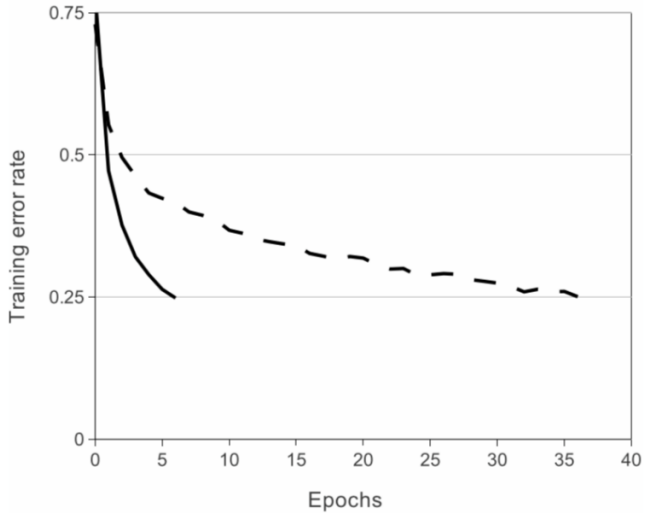
AlexNet bao gồm 5 convolution Layer và 3 Fully connected Layers.

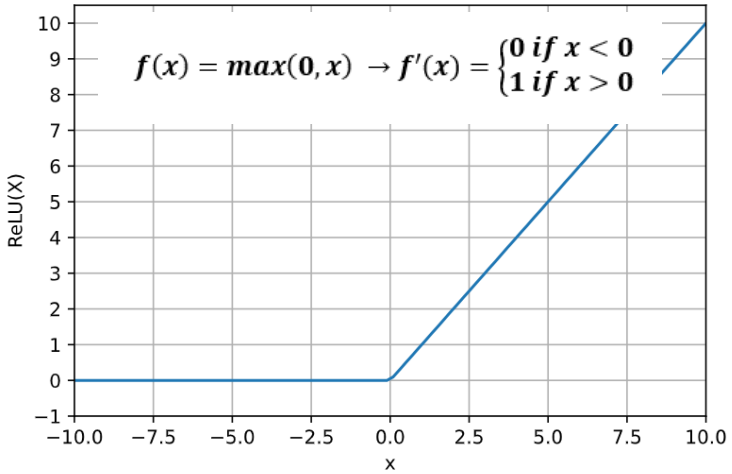


- Convolution layer rút trích các thông tin hữu ích trong các bức ảnh.
- Conv 1 và conv 2 kết nối với nhau qua một Overlapping Max Pooling ở giữa.
- Và cuối cùng là một bộ phân lớp softmax với 1000 lớp nhãn

ACTIVATION FUNCTION - ALEXNET

Một cải tiến quan trọng khác của AlexNet là việc sử dụng hàm phi tuyến ReLU (đầu tiên). Trước đây, các nhóm nghiên cứu khác thường sử dụng hàm kích hoạt là hàm Tanh hoặc hàm Sigmoid để huấn luyện mô hình neural network.



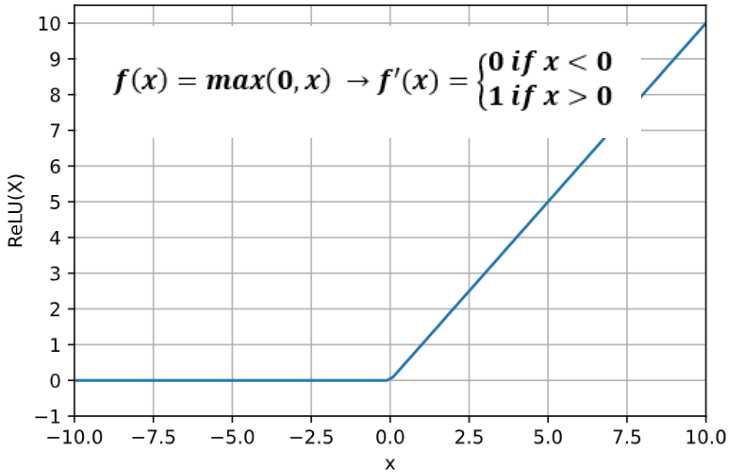


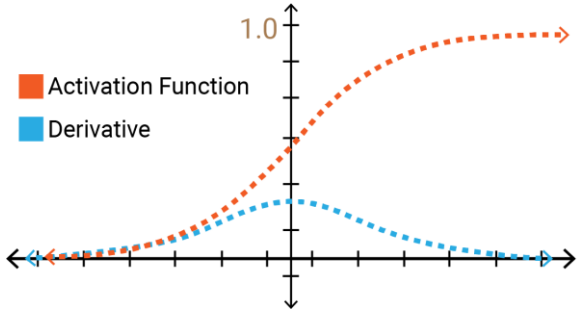
Việc sử dụng ReLU (đường nét liền trong hình), AlexNet đạt độ lỗi 25% trên tập huấn luyện và nhanh hơn gấp 6 lần so với mô hình tương tự nhưng sử dụng Tanh (đường nét đứt trong hình).

ACTIVATION FUNCTION - ALEXNET

Ngoài ra, hàm này cũng không bị chặn trên nên cũng giải quyết được vấn đề vanishing gradient.

Vanishing gradient: Là vấn đề xảy ra khi huấn luyện các mạng neural nhiều lớp. Giá trị đạo hàm là thông tin phản hồi của quá trình lan truyền ngược. Khi sử dụng Sigmoid hay Tanh, giá trị này trở nên vô cùng nhỏ tại các lớp nơ-ron đầu tiên khiến cho việc cập nhật trọng số mạng không thể xảy ra.





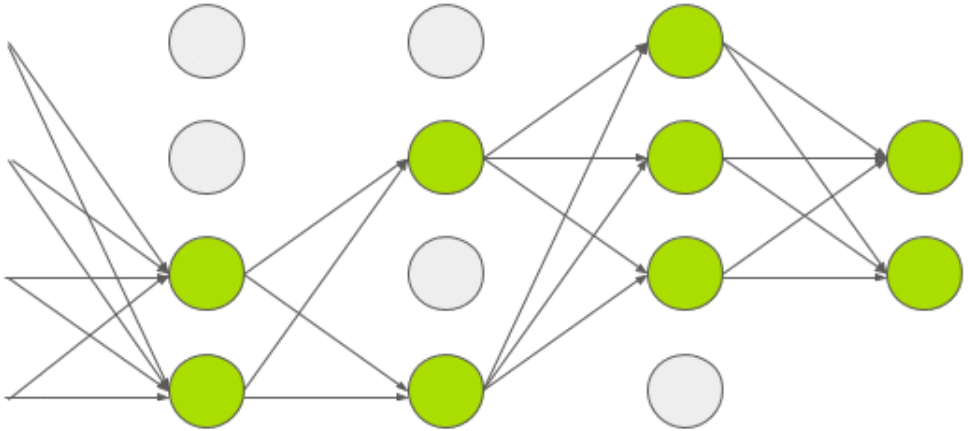
Hình trên là ví du về vanishing gradient trên Sigmoid, đạo hàm lớn nhất ở tâm đối xứng và tiệm cận 0 ở hai đầu.

DROPOUT - ALEXNET

Các tác giả của AlexNet đã thực nghiệm nhiều cách nữa để giảm overfitting. Họ sử dụng một kỹ thuật gọi là **dropout**.

Kỹ thuật này khá đơn giản, một neural sẽ có xác suất bị loại khỏi mô hình là 0.5. Khi một neural bị loại khỏi mô hình, nó sẽ không được tham gia vào quá trình lan truyền tiến hoặc lan truyền ngược.

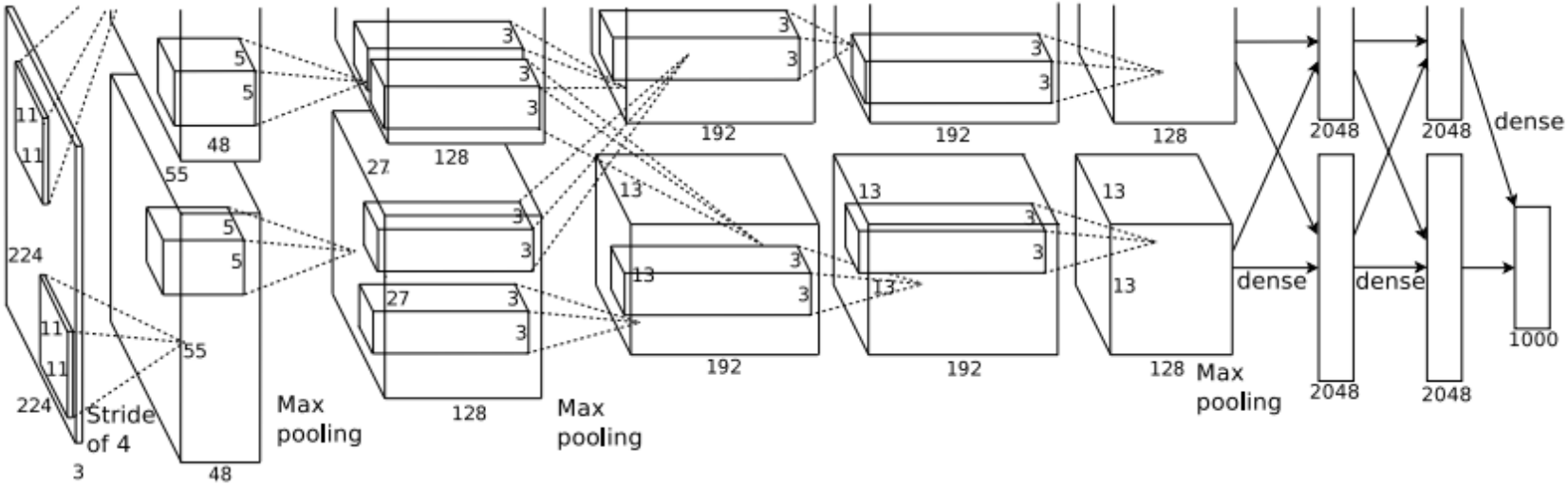
$$W' = W_{0,1} * W$$



$$W_{0,1} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Trong quá trình test, toàn bộ network được sử dụng, không có dropout, tuy nhiên, giá trị output sẽ scaled bởi tham số 0.5 tương ứng với những neural không sử dụng trong quá trình training.

LEARNING PROCESS - ALEXNET



Mô hình sử dụng 2 GPU cho training

LEARNING PROCESS - ALEXNET

Thuật toán gradient sử dụng để tối ưu W được tính như sau:

$$v_{i+1} = 0.9 \times v_i - 0.0005 \times \epsilon \times w_i - \epsilon \times \langle \frac{\partial L}{\partial w} | w_i \rangle_{D_i}$$

$$\rightarrow w_{i+1} = w_i + v_{i+1}$$

Với 0.9 là độ quán tính của momentum, 0.0005 là phân rã trọng lượng, ϵ là learning rate

- Để tìm được learning rate phù hợp, nhóm nghiên cứu phải điều chỉnh thủ công ϵ suốt quá trình training: khởi tao bằng 0.01 và chia cho 10 đến khi tỉ lệ hàm lỗi ngừng cải thiện.
- Train 1.2 triệu hình ảnh trong 90 chu kì, 5-6 ngày trên 2 GPU NIVIDA GTX 580 3 GB để hoàn thành.

CONCLUSION

RESULT

Năm 2012, Alenex đạt kết quả top-5 error với 16% làm giới nghiên cứu sững sờ:

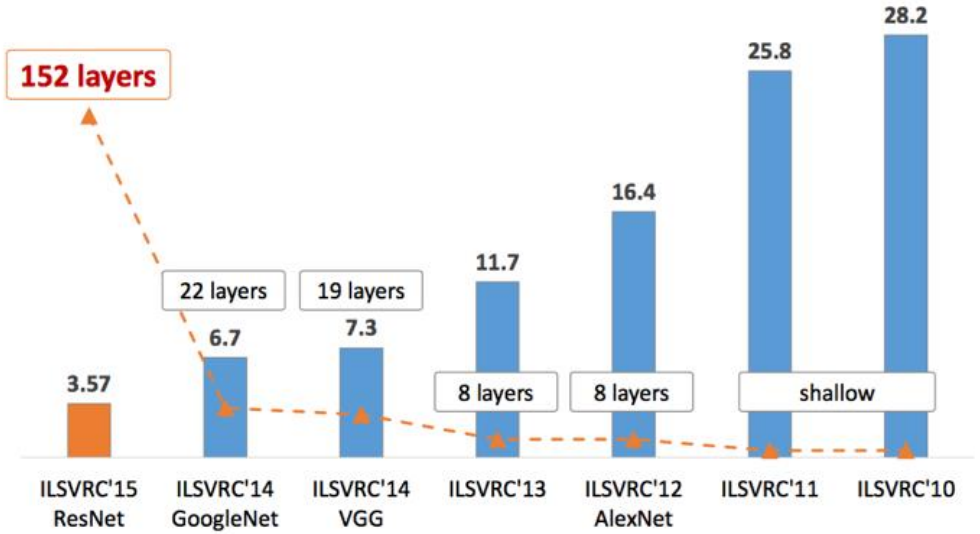
- Hai kỹ thuật đóng vai trò nổi bật: ReLU và dropout giúp giảm thời gian, giải quyết vanishing gradient và hiệu quả trong tránh overfitting.
- Sử dụng 2 GPU giúp tăng tốc thuật toán hiệu quả hơn so với CPU.

=> AlexNet được xem như là cuộc cách mạng của Deep CNN và là nền tảng của các mô hình: VGG, GoogleNet, ResNet,... đồng thời cũng tạo ra các ứng dụng đột phá.

IMPROVEMENT

- Tuy mạnh mẽ là vậy nhưng để tạo ra được AlexNet, thời điểm đó cũng gặp phải nhiều khó khăn như phần cứng, vì năm 2012 việc training mô hình với lượng tham số lớn là khó khăn, hiện nay với GPU người ta đã tạo ra được nhiều kiến trúc CNN phức tạp và hiệu quả.

- Một nhược điểm của AlexNet là độ lỗi, tuy đạt top1 thời đó nhưng đến thời điểm hiện tại nhiều mô hình đã vượt qua khả năng nhận biết của con người, tiêu biểu là ResNet với tỉ lệ lỗi chỉ 3.57%.



IMPROVEMENT

Từ đó ta có thể đưa ra một số giáp pháp để cải tiến AlexNet:

- Sử dụng các khối tích chập chứa các tầng lặp lại theo mẫu (như VGG).
- Sử dụng đa dạng các kích thước kernel và các nhánh song song (như mạng nối song song GoogleNet).
- Kỹ thuật batch normalization,...

THANKS!

Cảm ơn thầy và các bạn
đã lắng nghe

Do you have any questions?

Machine Learning: Deep Neural Network