

Convolutional Neural Network

Mạng tích chập



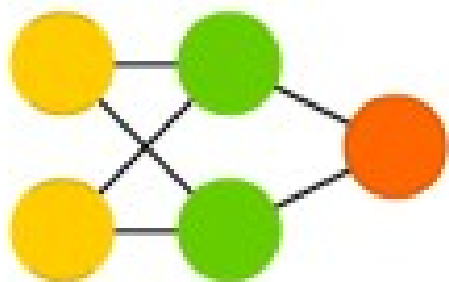
Outline



- Mạng nơ-ron tích chập CNN là gì?
- Thành phần cơ bản
 - Convolution layer
 - Fully connected layer
 - Activation
 - Pooling
- Hoạt động như thế nào?
 - Loss function, gradient

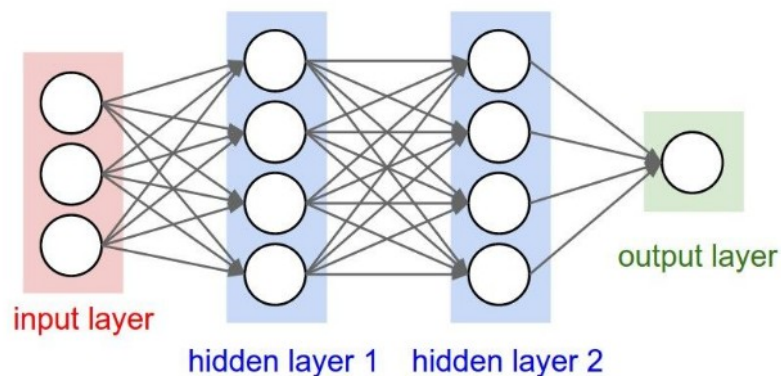


Mạng nơ-ron tích chập là gì?

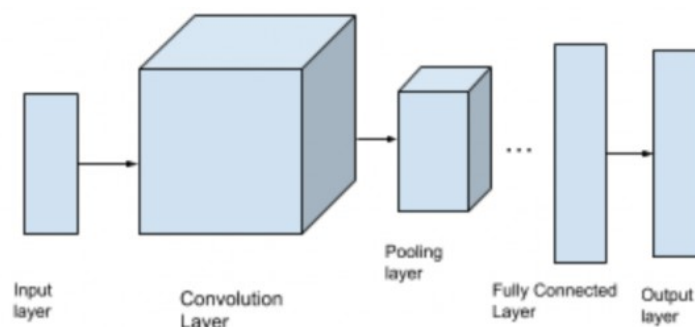


Neural Network

Mạng nơ-ron chứa các lớp tích chập được gọi là mạng nơ-ron tích chập.



Deep Neural Network

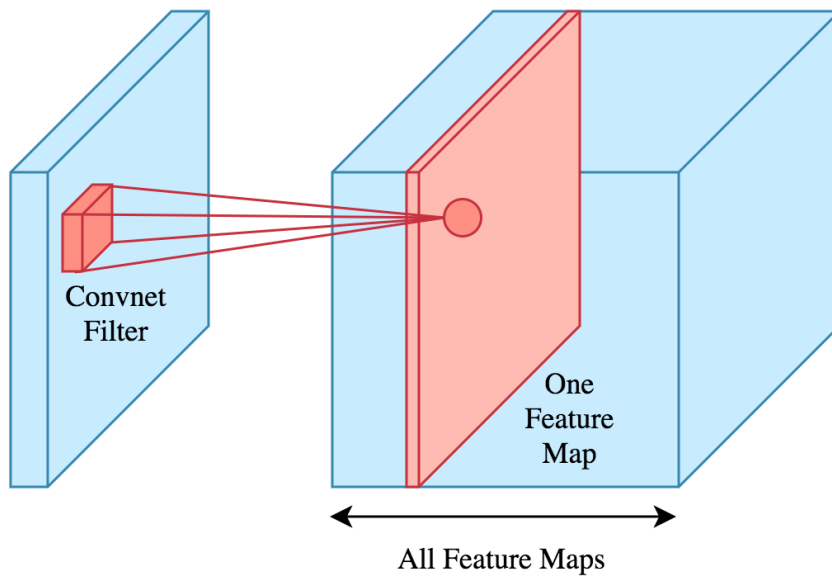


Convolutional Neural Network

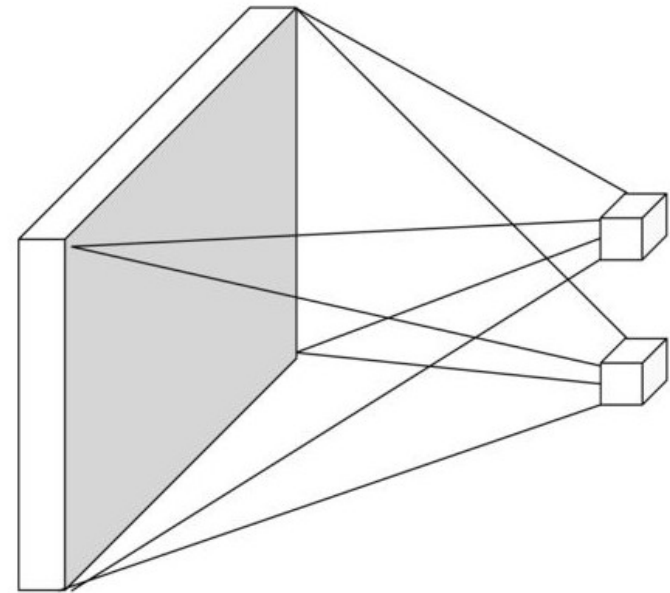
Thành phần cơ bản



- Convolution layer



- Fully connected layer

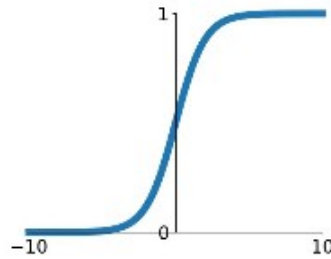




Activation functions

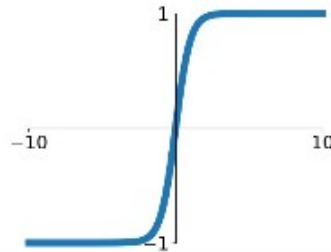
Sigmoid

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



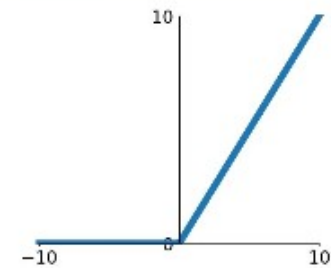
tanh

$$\tanh(x)$$



ReLU

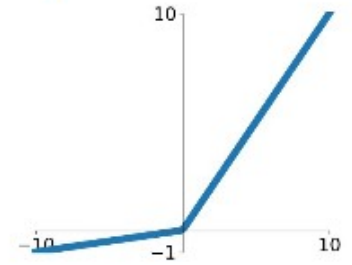
$$\max(0, x)$$



ReLU is a good default choice for most problems

Leaky ReLU

$$\max(0.1x, x)$$

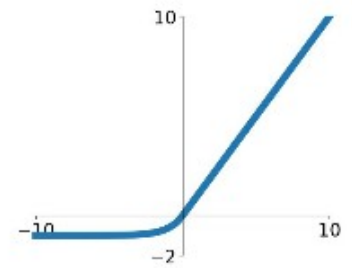


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

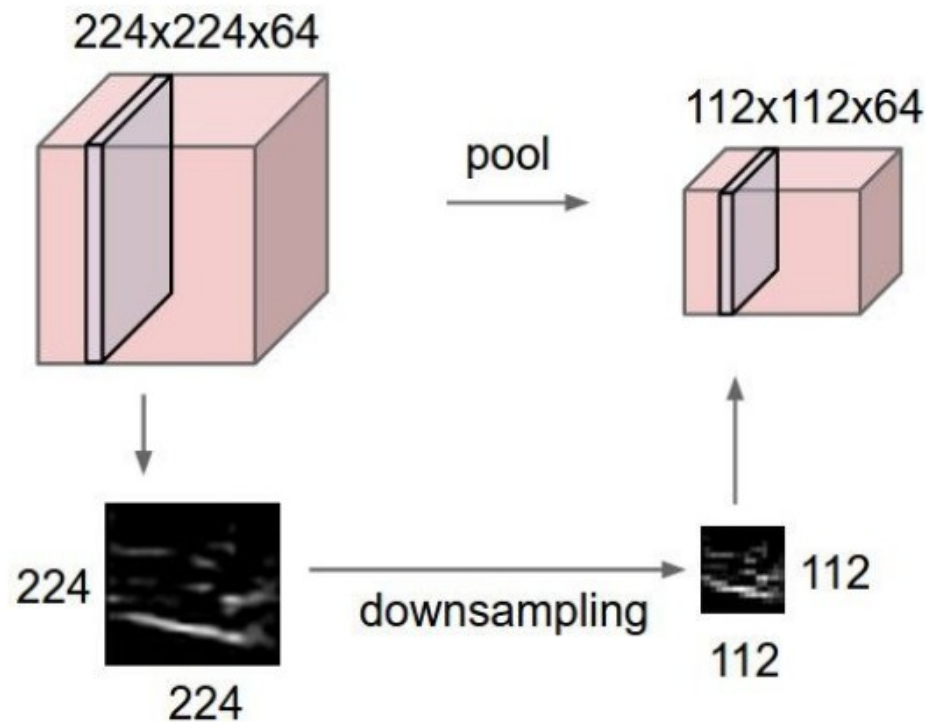
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$





Pooling layer

- makes the representations smaller and more manageable
- operates over each activation map independently:





- Batch Norm

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_1 \dots x_m\}$;

Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

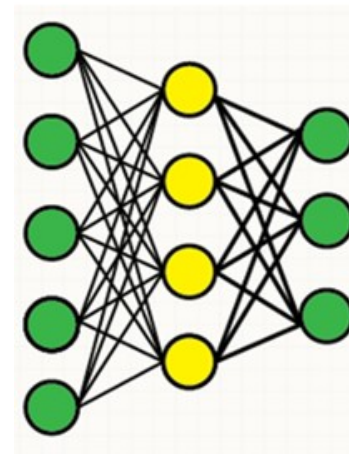
$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{ mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{ mini-batch variance}$$

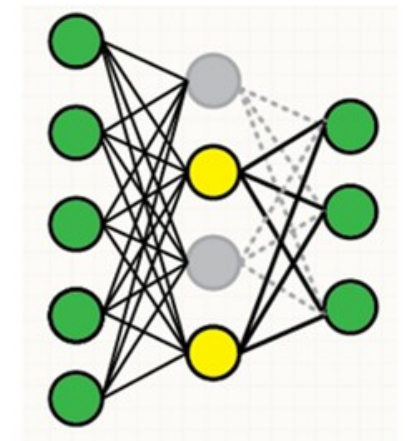
$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{ normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{ scale and shift}$$

- Dropout

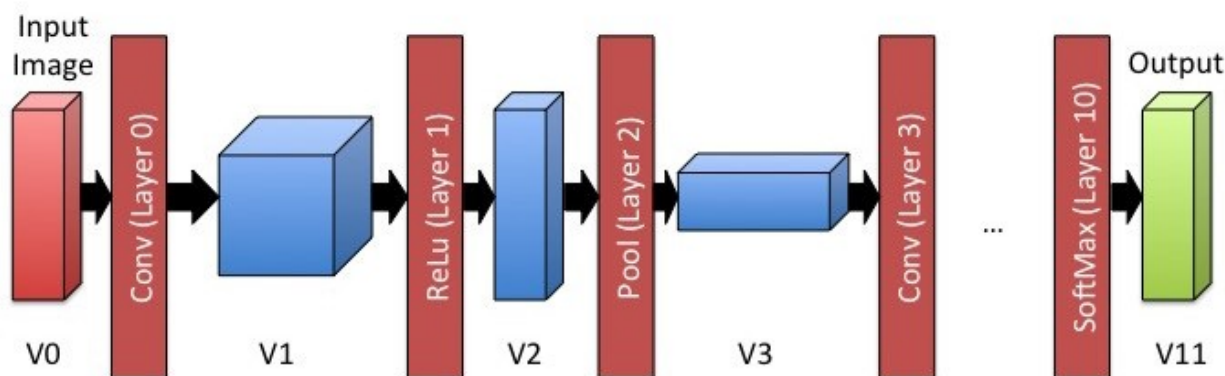


Original



Dropout

Hoạt động của mạng CNN



Mục tiêu: Tính các parameters trong Convnet filters, BatchNorm...

Cách làm: optimization, loss function, gradient

Loss function

- “Quantify our unhappiness with the scores across the training data “
- Loss = metric (Output, GroundTruth)
- Feed-forward & Back Prop
- Compute current output --> Improve weights

Linear Regression with Gradient Descent



$$h_{\theta}(x) = \sum_{j=0}^n \theta_j x_j$$

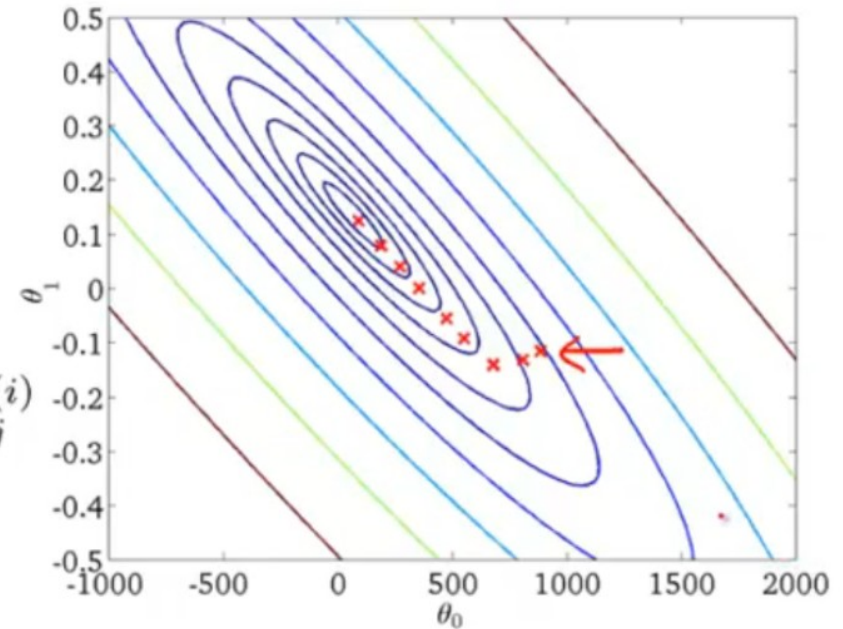
$$J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

Repeat {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(for every $j = 0, \dots, n$)

}





Batch gradient descent

$$\rightarrow J_{train}(\theta) = \frac{1}{2m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

Repeat {

$$\rightarrow \theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m (h_{\theta}(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

$$\frac{\partial}{\partial \theta_j} J_{train}(\theta)$$

(for every $j = 0, \dots, n$)

}

Stochastic gradient descent

$$\rightarrow \text{cost}(\theta, (x^{(i)}, y^{(i)})) = \frac{1}{2} (h_{\theta}(x^{(i)}) - y^{(i)})^2$$

$$J_{train}(\theta) = \frac{1}{m} \sum_{i=1}^m \text{cost}(\theta, (x^{(i)}, y^{(i)}))$$

1. Randomly shuffle dataset. \leftarrow

2. Repeat {

for $i = 1, \dots, m$ {

$$\theta_j := \theta_j - \alpha (h_{\theta}(x^{(i)}) - y^{(i)}) \cdot x_j^{(i)}$$

(for $j = 0, \dots, n$)

}

}

$$\frac{\partial}{\partial \theta_j} \text{cost}(\theta, (x^{(i)}, y^{(i)}))$$

$$\rightarrow (x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), (x^{(3)}, y^{(3)}), \dots$$

Optimizer



$$\begin{aligned}c(n) &= \rho c(n-1) + \eta \frac{\partial L}{\partial w}(n) \\w(n+1) &= w(n) - c(n)\end{aligned}$$

(a) Momentum

$$\begin{aligned}c(n) &= \rho c(n-1) + \eta \frac{\partial L}{\partial w}(w(n) + \rho c(n-1)) \\w(n+1) &= w(n) - c(n)\end{aligned}$$

(b) Nesterov Momentum

$$w(n+1) = w(n) - \eta \frac{\frac{\partial L(n)}{\partial w}}{\sqrt{\sum_{i=1}^n \left[\frac{\partial L(i)}{\partial w} \right]^2 + 10^{-7}}}$$

(c) Adagrad

$$\begin{aligned}\Delta(n) &= \alpha \Delta(n-1) + (1 - \alpha) \left[\frac{\partial L(n)}{\partial w} \right]^2 \\w(n+1) &= w(n) - \eta \frac{\frac{\partial L(n)}{\partial w}}{\sqrt{\Delta(n)}}\end{aligned}$$

(d) RMSProp

$$\begin{aligned}\Lambda(n) &= \beta \Lambda(n-1) + (1 - \beta) \frac{\partial L(n)}{\partial w} \\ \Delta(n) &= \alpha \Delta(n-1) + (1 - \alpha) \left[\frac{\partial L(n)}{\partial w} \right]^2 \\w(n+1) &= w(n) - \eta \frac{\Lambda(n)}{\sqrt{\Delta(n) + 10^{-7}}}\end{aligned}$$

(e) Adam