C:\Users\tdqua_000\Dropbox\SS-Slides\DeCuong-CDIO\Template CDIO v4.2\Templates\Hinh anh\LogoTruong.png

Bộ môn Phương Pháp Nghiên Cứu Khoa Học

Khoa Công Nghệ Thông Tin

Đại học Khoa học Tự nhiên TP HCM

**PHƯƠNG PHÁP NGHIÊN CỨU KHOA HỌC**

**BÁO CÁO ĐỀ TÀI**

**NHẬN DẠNG CHỮ SỐ VIẾT TAY SỬ DỤNG MẠNG NƠRON**

**-**

**HANDWRITTEN DIGIT RECONIGTION USING CONVOLUTIONAL NEURAL NETWORK**

MỤC LỤC

[Thông tin nhóm 3](#_Toc502915732)

[Abstract 4](#_Toc502915733)

[1 Introduction – Giới thiệu 4](#_Toc502915734)

[2 Related Works 5](#_Toc502915735)

[3 Dataset MNIST– Tập dữ liệu MNIST 5](#_Toc502915736)

[4 Method – Phương pháp 6](#_Toc502915737)

[4.1 Mạng Nơ Ron 6](#_Toc502915738)

[4.2 Mô Hình 1-layer neural network 7](#_Toc502915739)

[4.3 5 – Layer fully connected neural network 13](#_Toc502915740)

[4.4 Convolutionel Neural Network. 16](#_Toc502915741)

[5 Kết quả thực nghiệm. 23](#_Toc502915742)

[6 Kết luận và phát triển 26](#_Toc502915743)

[7 Nguồn tham khảo. 26](#_Toc502915744)

# Thông tin nhóm

|  |  |  |
| --- | --- | --- |
| **MSSV** | **Họ Tên** | **Email** |
| 1512641 | Võ Anh Tuấn | voanhtuanvn12@gmail.com |
| 1512639 | Phạm Anh Tuấn |  |
| 1512030 | Trần Quốc Bảo |  |

# Abstract

Trong những năm gần đây, cùng với cuộc cách mạng công nghệ 4.0, dữ liệu ảo đang có sự phát triển mạnh. Tuy nhiên, phần lớn dữ liệu vẫn còn được lưu trữ dưới dạng ảnh. Chúng tôi xin được trình bày một phương pháp nhận dạng và phân loại chữ số viết tay bằng mạng nơ ron tích chập – convolutional neural network sử dụng tập dữ liệu MNIST dataset. Kết quả đạt được trên tập MNIST đạt 98%.

# Introduction – Giới thiệu

Trong những năm gần đây, cùng với cuộc cách mạng công nghệ 4.0, dữ liệu ảo đang có sự phát triển mạnh. Tuy nhiên, phần lớn dữ liệu vẫn còn được lưu trữ dưới dạng ảnh. Trong phạm vi bài viết, chúng tôi giới hạn các dữ liệu dạng ảnh là các bức ảnh chứa chữ số viết tay. Nhiệm vụ của chúng ta là số hóa các bức ảnh. Số hóa ở đây không phải chuyển các bức ảnh chụp thành các tập các pixels để lưu vào máy tính, mà là xác định tổng thể ảnh chứa số viết tay nào và phân loại chúng.

Nhận dạng đối tượng là một lĩnh vực trong ngành thị giác máy tính, với mục tiêu tìm và xác định các đối tượng trong một hình ảnh hoặc video. Chúng có thể nhận ra nhiều vật thể trong hình ảnh, cho dùng hình ảnh của vật thể có thể khác nhau ở các điểm hình, ở nhiều kích cỡ và quy mô khác nhau hoặc thậm chí chúng có thể được dịch hoặc xoay. Các đối tượng thậm chí có thể được nhận ra khi chúng bị che khuất một phần khỏi tầm nhìn. Nhiệm vụ này vẫn còn là một thách thức đối với hệ thống quan sát của máy tính. Nhiều phương pháp nhận dạng đã được thực hiện trong nhiều thập kỷ.

Nếu bạn là một lập trình viên đam mê công nghệ và AI thì chúng tôi tin chắc rằng đây là một lĩnh vực nghiên cứu khá thú vị và là mảnh đất công nghệ màu mỡ dành cho bạn. Và nếu như bạn không phải là một lập trình viên thì có bao giờ bạn tự đặt ra câu hỏi - làm thế nào và bằng cách nào mà chiếc smart phone của bạn có thể nhận dạng các ký tự trong ảnh để chuyển thành một số dưới dạng text mà bạn thấy ở trên bức ảnh mà mình đang scan. Trên thực tế - nhận diện ảnh có rất nhiều ứng dụng trong đời sống của bạn chẳng hạn như giúp bạn quet mã thẻ ngân hang khi thanh toán – tra giá trên các sản phẩm – nhận diện khuôn mặt và hay gần nhận chính là ID face - một chức năng giúp nâng cao tính bảo mật của điện thoại. Vậy tại sao trong đề tài này chúng tôi lại muốn giới thiệu cho bạn về nhận diện chữ số trong nhận diện ảnh. Vì sở dĩ chúng tôi nhận ra rằng đây là một có ứng dụng cực lớn – và là nền tảng cho các ứng dụng có kích thước dữ liệu , tầm ảnh hưởng lớn hơn trong lĩnh vực xử lí ảnh trong đời sống của con người.

Trong bài viết, chúng tôi xin được giới thiệu một phương pháp nhận dạng chữ số viết tay sử dụng mạng convolution neural network, tạm dịch là mạng nơron tích chập.

# Related Works

Có nhiều cách để phân loại ảnh. Đầu tiên là Phân loại tuyến tính (Linear Classifiers). Theo [khảo sát](http://cs231n.stanford.edu/reports/2015/pdfs/vishnu_final.pdf) training với tập dữ liệu 30000 bức ảnh số trong tập MNIST đã được gán nhãn. Testing với tập 5000 bức ảnh. Kết quả cho thấy tỉ lệ chính xác trên tập train là 88.03% và tỉ lệ chính xác trên tập testing là 88.69%. Nhưng trong tập dữ liệu Chars74K, với tập training gồm 6400 ảnh và tập testing gồm 1600 ảnh. Kết quả chỉ đạt 30.15%, thấp hơn nhiều so với tập MNIST.

Tiếp theo là thuật toán KNN (K-Nearest Neighbor). Qua [khảo sát](http://cs231n.stanford.edu/reports/2015/pdfs/vishnu_final.pdf) với việc tìm 100 neighbors của mỗi bức ảnh. Tập training gôm 30000 bức ảnh lấy từ tập MNIST, tập testing gồm 5000 ảnh. Kết quả đạt được là 93,73% với tập training và 93.28% trên tập testing.

Hai thuật toán trên tỏ ra hiệu quả trên tập MNIST với độ chính xác khá cao, nhưng chúng tôi đã tìm hiểu về mạng nơ ron và nhận thấy với việc sử dụng mạng nơ ron, ta có thể tăng độ chính xác cho việc phân loại ảnh.

# Dataset MNIST– Tập dữ liệu MNIST

Là một tập dữ liệu được phát triển bới Yann LeCun, Corinna Cortes và Christopher Burges để đánh giá mô hình học máy trên vấn đề phân loại chữ viết tay. Bộ dữ liệu được xây dựng từ một số liệu tài liệu được quét từ Viện Tiêu chuẩn và Công nghệ Quốc gia (NIST). Đây cũng chính là tên của bộ dữ liệu : Bộ Số liệu NIST hoặc MNIST.

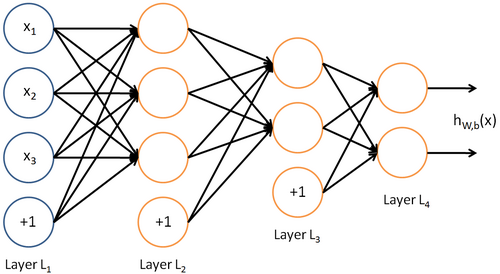
Hình ảnh của chữ số được lấy từ nhiều tài liệu đã được quét, được chuẩn hóa về kích thước và ở giữa. Điều này làm cho nó là một tập dữ liệu tuyệt vời để đánh giá các mô hình, cho phép nhà phát triển tập trung vào việc học máy với rất ít dọn dẹp dữ liệu hoặc chuẩn bị cần thiết.

Mỗi hình ảnh là một hình vuông 28 x 28 pixel (tổng số 784 pixel). Một điểm tiêu chuẩn của bộ dữ liệu được sử dụng để đánh giá và so sánh các mô hình, nơi có 60.000 hình ảnh được sử dụng để đào tạo mô hình và một bộ 10.000 hình ảnh riêng được sử dụng để kiểm tra nó.

# Method – Phương pháp

## 4.1 Mạng Nơ Ron

Theo nghĩa sinh học, mạng Neural là một tập hợp các dây thần kinh kết nối với nhau. Trong [Deep learning](https://blog.duyetdev.com/2016/02/deep-learning-la-gi.html#.VvajANx96b8),  Neural networks để chỉ mạng Neural nhân tạo, cấu thành từ các lớp Neural.



[Deep Learning](https://blog.duyetdev.com/2016/02/deep-learning-la-gi.html#.VvajANx96b8) được xây dựng trên cơ sở 1 mạng lưới các Neural nhân tạo, tổ chức thành từng lớp (Layer). Kết quả của mỗi lớp lại biểu diễn các thuộc tính (features) của lớp cao hơn (lớp phía sau).

Theo hình trên, các giá trị nhận được ở Layer 1, lần lượt thông qua các lớp đến lớp cuối cùng. Layer 1 gọi là Input Layer, Layer 4 là Output layer, các layer ở giữa còn lại gọi là Hidden Layer.

Theo mô hình này, các lớp cao hơn sẽ *"hiểu"*được các giá trị đưa vừa ở lớp sau nó.  
Ví dụ, mình muốn sử dụng Deep Learning để xử lý ảnh. Layer 1 có input là các pixel của ảnh. Layer đầu tiên này được dùng để huấn luyện (training) để nhận diện các hình dáng (shapes) cơ bản của vật thể. Các layers cao hơn sử dụng các shapes của ảnh này, để "hiểu" là các shapes này dùng để biểu diễn vật gì.

Đối với tập dữ liệu MNIST, mạng nơron có nhiệm vụ phân loại các tập dữ liệu là các chữ số viết tay đã được gán nhãn, nhận ra chúng thuộc vào tập nào trong tập các số từ 0 đến 9. Nó dựa trên một số biến số như (trọng sô “weight” và độ lệch “bias”) và cần có một giải pháp để phân loại với độ chính xác chấp nhận được. Giá trị chính xác “correct values” sẽ được học thông qua một quá trình đào tạo được thể hiện một cách trừu tượng thông qua vòng lặp sau :

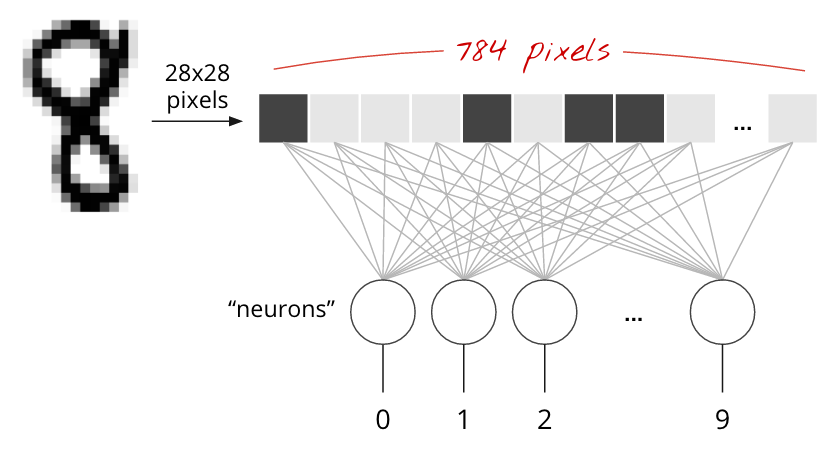
**Training digits 🡪 updates to weights and biases 🡪 better recognition (loop)**

Thông qua vòng lặp này, giá trị chính xác sẽ được tối ưu hóa dần bằng cách tự điều chỉnh trọng số cũng như độ lệch.

Sau đây ta sẽ tìm hiểu một số mô hình mạng neural phổ biến.

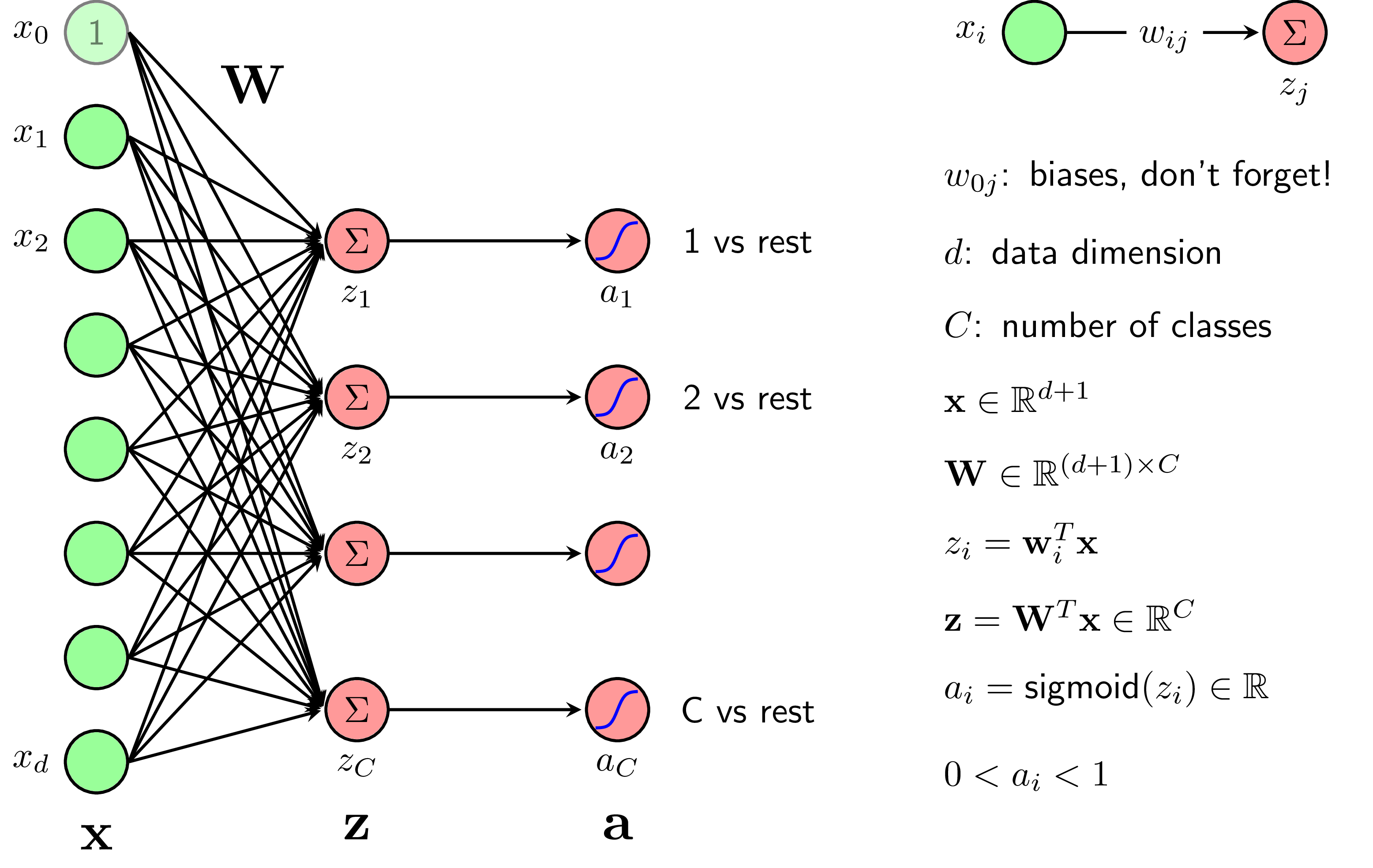
## 4.2 Mô Hình 1-layer neural network

Tập chữ số viết tay MNIST là một tập dữ liệu có những bức ảnh trắng đen có kích cỡ 28x28 pixels. Giải pháp đơn giản nhất để phân loại chúng là dùng tất cả 28x28 = 784 pixels làm tập input cho mô hình 1-Layer neural network.



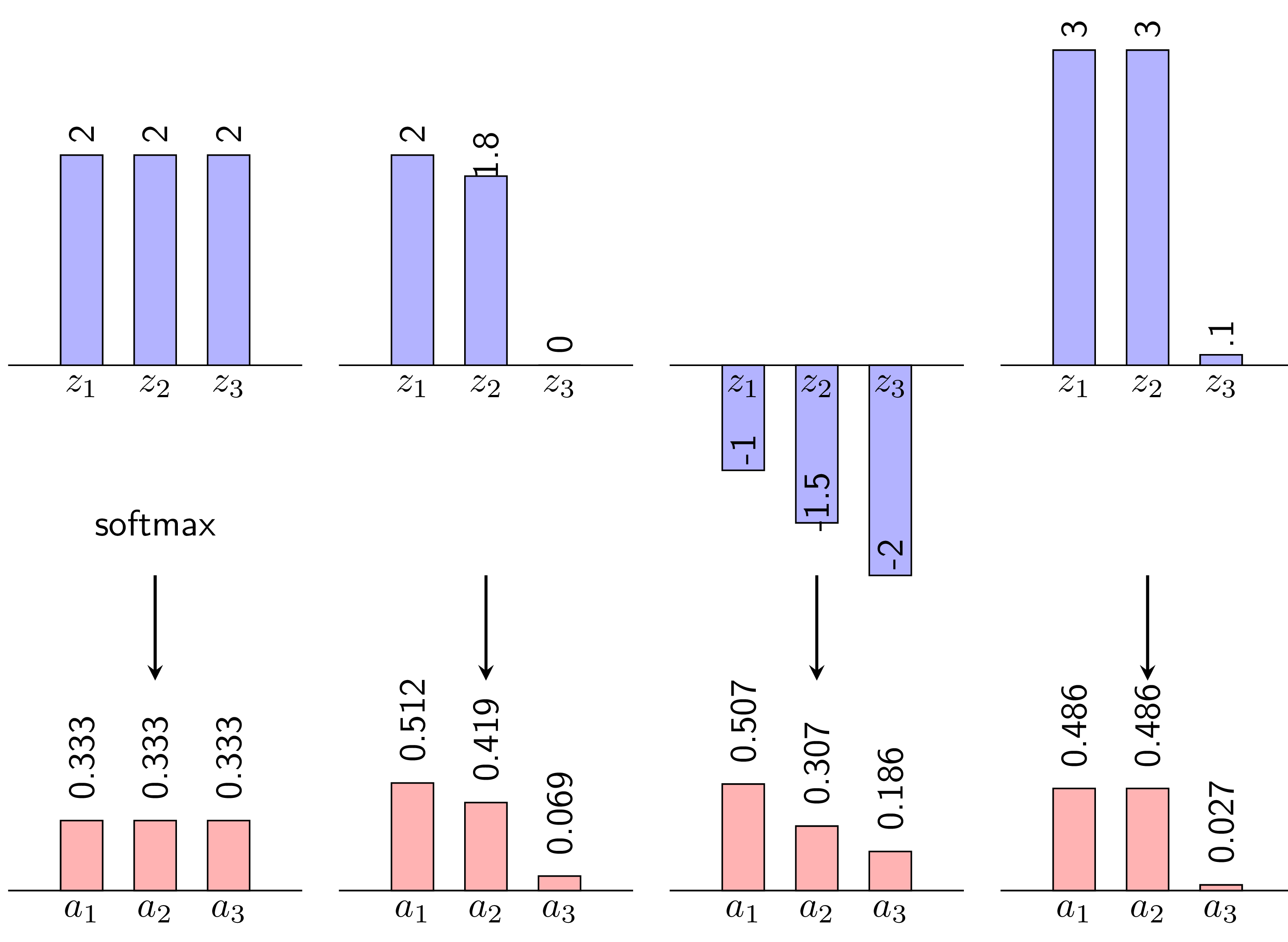
Mỗi một neural trong mạng sẽ tính toán tổng các trọng số của tất cả các đầu vào của nó, cộng thêm một độ lệch “bias” để thông qua một số chức năng kích hoạt phi tuyến.

Ở đây chúng ta sẽ thiết kế một mạng nơ ron gồm 1 lớp có 10 neural ở output vì chúng ta muốn phân loại các chữ số thành 10 lớp ( từ 0 đến 9). Output layer thường được biểu thị dưới dạng one-vs-rest, tức là tập output chỉ có 1 neural mang giá trị 1, tất cá các neural còn lại mang giá trị 0.



Chúng ta cần một mô hình xác suất sao cho với mỗi input  (trong trường hợp này, số lượng của x là 28x28 + 1 = 785 do có thêm một hệ số bias) ,  thể hiện xác suất để input đó rơi vào class . Vậy điều kiện cần là các   phải dương và tổng của chúng bằng 1. Để có thể thỏa mãn điều kiện này, chúng ta cần nhìn vào mọi giá trị  và dựa trên quan hệ giữa các này để tính toán giá trị của . Ngoài các điều kiện  lớn hơn 0 và có tổng bằng 1, chúng ta sẽ thêm một điều kiện cũng rất tự nhiên nữa, đó là: giá trị  càng lớn thì xác suất dữ liệu rơi vào class  càng cao. Điều kiện cuối này chỉ ra rằng chúng ta cần một hàm đồng biến ở đây. Nhận thấy rằng có thể nhận các giá trị cả âm và dương. Một hàm số mượt đơn giản có thể chắc chắn rằng sẽ trở thành một giá trị dương, và đồng biến là hàm . Điệu kiện mượt để dễ tính đạo hàm hơn sau này. Điều kiện cuối cùng là tổng của các bằng 1 có thể được đảm bảo nếu :

Ta gọi đây là hàm softmax.



**Tính toán hàm mất mát**

Như vậy, ứng với tập các bức ảnh và một mạng W cho trước, mạng neural sẽ cho ra một bộ 10 số ứng với xác xuất phân loại bức ảnh đó thuộc nhãn gì. Việc còn lại là so sánh xác suất đó với nhãn xem có trùng khơp hay không, nêu có sai lệch, ta sẽ phải điều chỉnh mạng W sao cho độ sai lệch với kết quả thực là thấp nhất. Gọi giá trị dự đoán tìm được là và giá trị thực tế là . Một sự lựa chọn đơn giản cho hàm mất mác có thể nghĩ đến là :

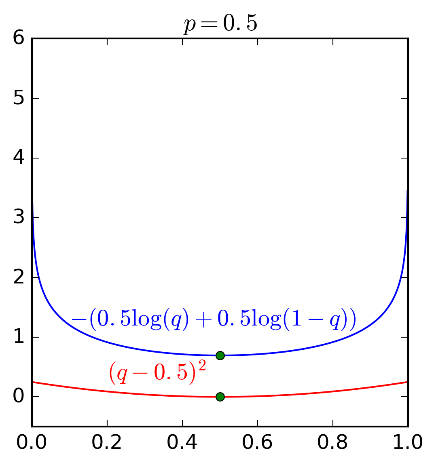
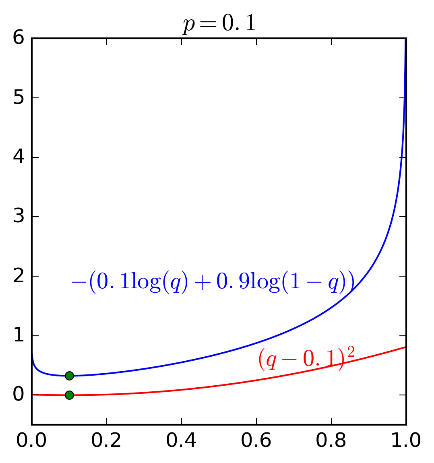
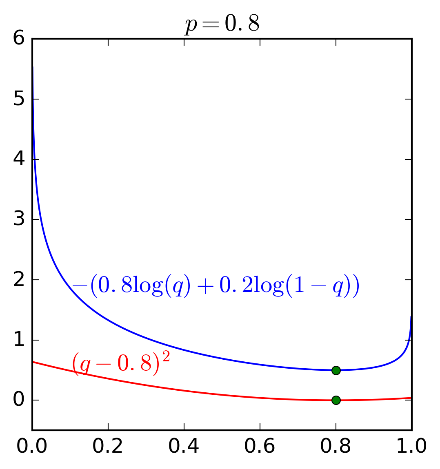
.

Nhưng đây không phải là một sự lựa chọn tốt. Khi đánh giá sự khác nhau (hay khoảng cách) giữa hai phân bố xác suất (probability distributions), chúng ta có một đại lượng đo đếm khác hiệu quả hơn. Đại lượng đó có tên là cross entropy.

Cross Entropy giữa hai phân phối **p** và **q** được định nghĩa là :

với

Để hiểu rõ hơn ưu điểm của hàm cross entropy với hàm bình phương khoảng cách thông thường, chúng ta cũng xem hình ảnh dưới đây. Ví dụ C = 2 và

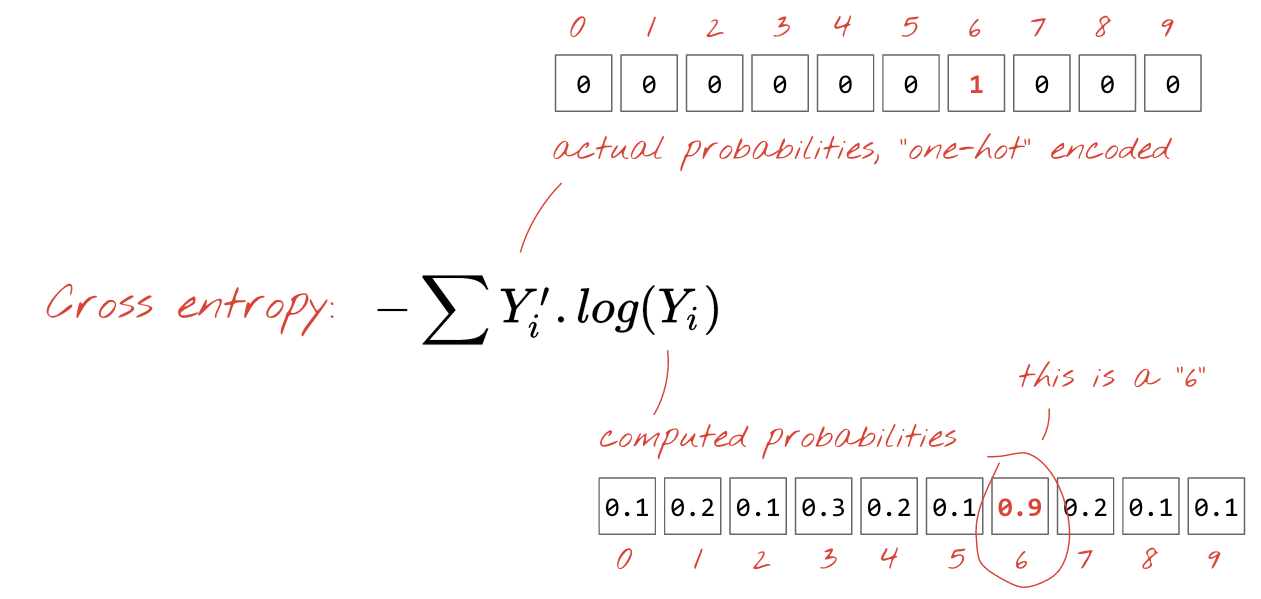
  

Ta có hai nhận xét sau :

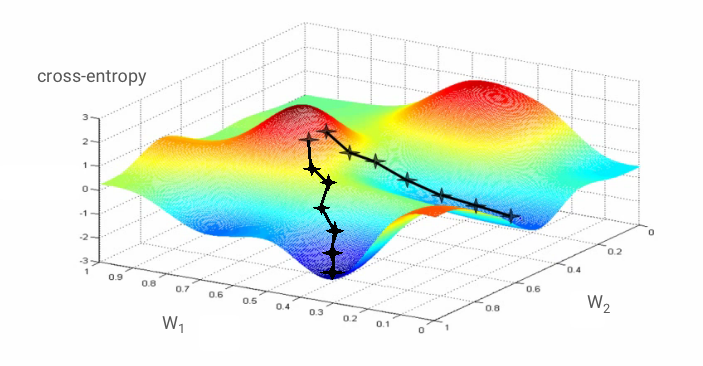
* Giá trị nhỏ nhất của cả hai hàm số đạt được khi q = p tại hoành độ của các điểm màu xanh lục.
* Quan trọng hơn, hàm cross entropy nhận giá trị rất cao (tức loss rất cao) khi q ở xa p. Trong khi đó, sự chênh lệch giữa các loss ở gần hay xa nghiệm của hàm bình phương khoảng cách là không đáng kể. Về mặt tối ưu, hàm cross entropy sẽ cho nghiệm gần với p hơn vì những nghiệm ở xa bị trừng phạt rất nặng.

Hai tính chất trên đây khiến cho cross entropy được sử dụng rộng rãi khi tính khoảng cách giữa hai phân phối xác suất.

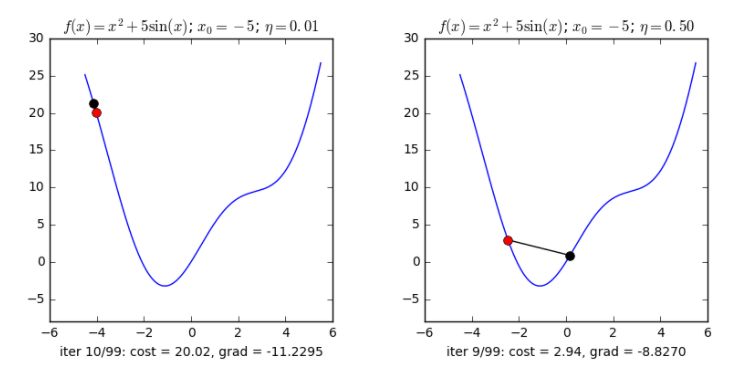
Trong bài toán supervised learning, p thường là *đầu ra thực sự* vì đầu ra thực sự chỉ có 1 thành phần bằng 1, còn lại bằng 0 (one-hot), q thường là *đầu ra dự đoán*, khi mà không có xác suất nào tuyệt đối bằng 1 hoặc tuyệt đối bằng 0 cả.

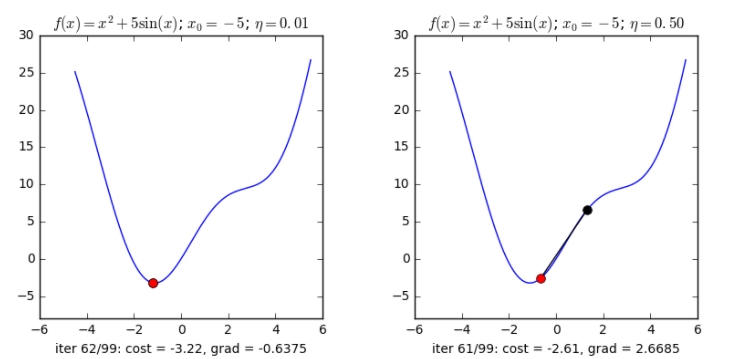


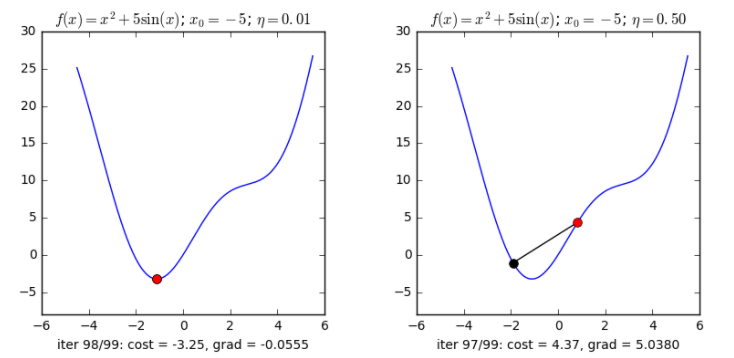
Như vậy có thể nói hàm cross entropy đại diện cho độ mật mát( độ lêch) của kết quả dự đoán so với kết quả thực tế. Ta phải tối ưu giá trị entropy sao cho độ mất mát là thấp nhất.



Việc tối ưu hàm mất mát phụ thuộc vào ma trận trọng số W và sộ lệch (bias) và phụ thuộc nhiều vào learing rate (tốc độ học). Việc xác định leaning rate là không đơn giản. vì nếu chọn learning rate quá thấp thì sẽ mất khá nhiều thời gian để các điểm hội tụ về vị trị min, còn nếu quá cao thì điểm sẽ có khả năng không hội tụ về min mà sẽ chạy xung quanh min.





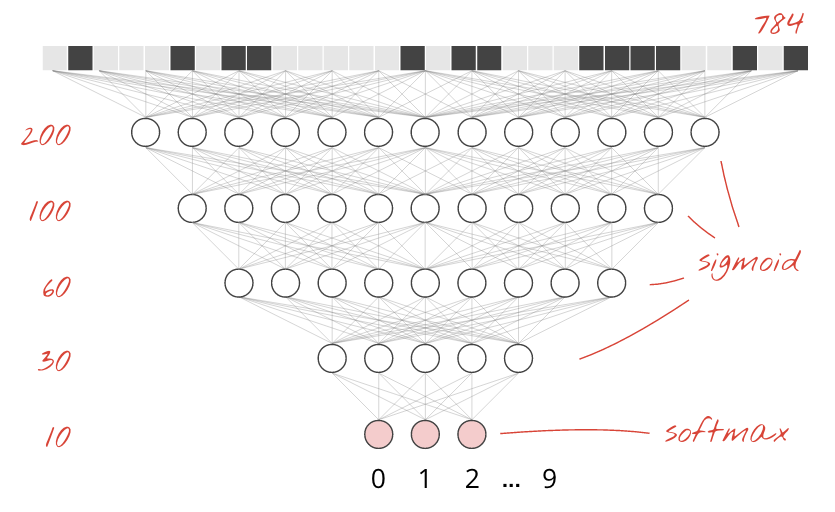


Tổng kết lại, quá trình training sẽ có vòng lặp như sau :

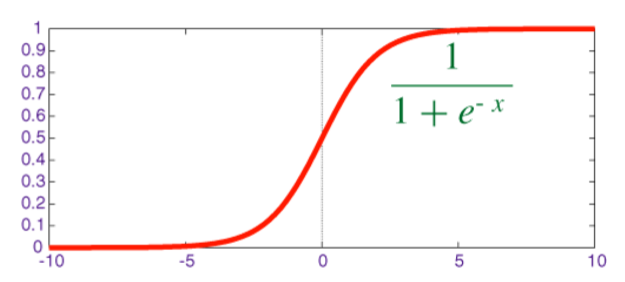
**Huấn luyện tập ảnh => xác định hàm mất mác => tối ưu hàm mất mác => cập nhật trọng số và bias => lập lại với các tập hình ảnh huấn luyện kế tiếp.**

## 4.3 5 – Layer fully connected neural network

Để nâng cao độ chính xác trong việc nhận dạng, chúng ta sẽ thêm vào nhiều lớp vào mạng neural để có thể rút trích được nhiều đặc điểm (shape) để nhận dạng. Sau đây là mô hình ví dụ :



Chúng ta sẽ giữ lại hàm kích hoạt softmax ở lớp cuối cùng, những lớp trên chúng ta sẽ sử dụng hàm kích hoạt đơn giản hơn: sigmoid:

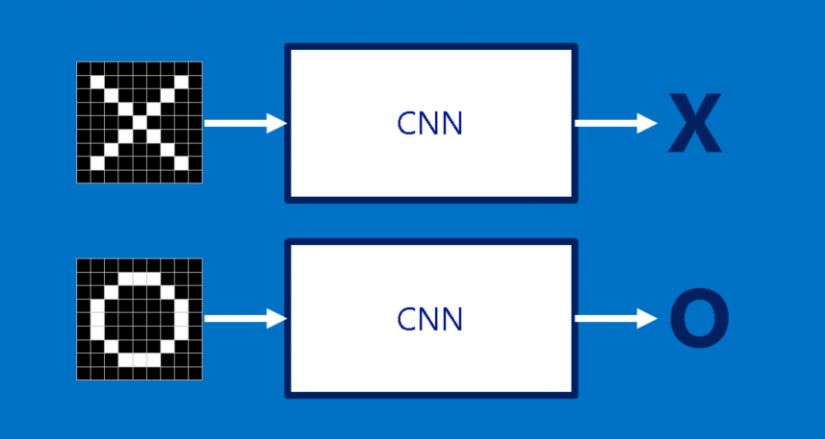


## 4.4 Convolutionel Neural Network.

Convolutional Neural Networks (còn được gọi là ConvNets hay CNNs) là một dạng Neural Networks tỏ ra rất hiệu quả trong các lĩnh vực nhận diện đối tượng và phân loại. Một trong những mạng ConvNets đầu tiên ra đời là mạng LeNet, giúp đẩy mạnh về lĩnh vực DeepLearning. Là một sự phát minh của nhà khoa học máy tính Yann LeCun(1960) được sử dụng chu yếu trong các nhiêm vụ nhận diện ký tự như đọc mã Zip, chữ số, v.v..  Chúng có thể học được cách phân loại các hình ảnh thậm chí còn tốt hơn con người trong một số trường hợp. Nếu có một phương pháp mà không nói quá lên tí nào, thì đó chính là CNNs.

Cái hay ở đây là nó rất dễ hiểu khi ta chia nó ra thành các thành phần cơ bản.

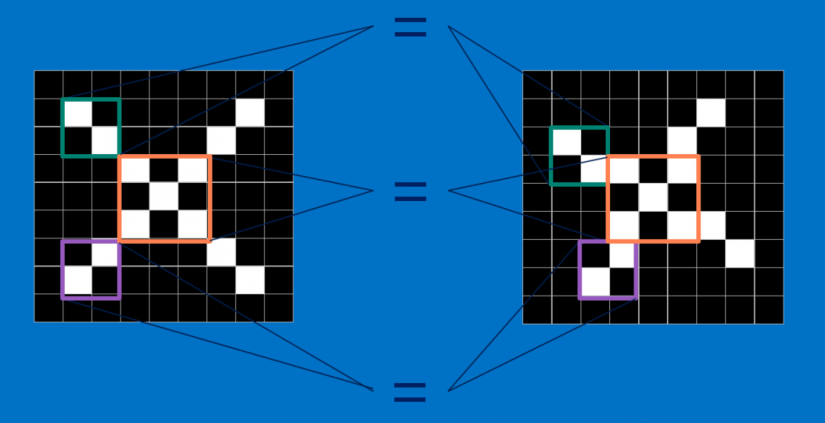
Chúng ta sẽ dùng một ví dụ rất đơn giản: xác định xem hình ảnh là một X hay một O. Ví dụ này là đủ để minh họa các nguyên lý đằng sau CNNs, nhưng vẫn đủ đơn giản để tránh bị sa lầy vào các chi tiết không cần thiết. CNN của chúng ta có một công việc. Mỗi lần chúng ta đưa nó một bức hình, nó phải quyết định xem nó có một X hay một O. Giả sử rằng luôn có một hoặc là cái này hoặc là cái kia.



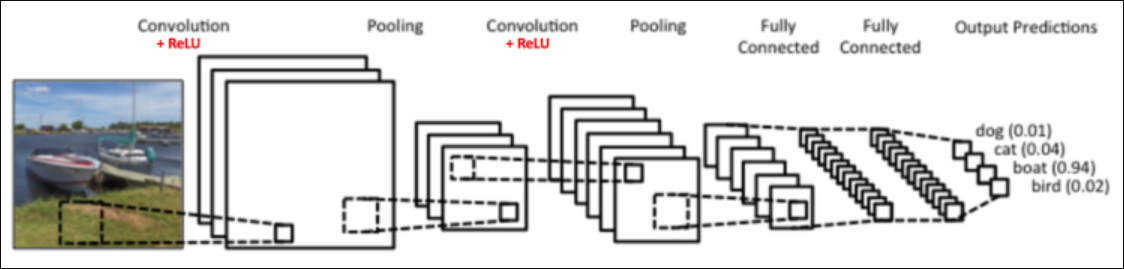
Một cách tiếp cận đơn giản để giải quyết vấn đề này là lưu một hình ảnh của một X và của một O, và so sánh mỗi hình ảnh mới với hai hình mẫu đó để xem nó khớp với bên nào hơn. Điều khiến cho công việc này khó khăn là do máy tính chỉ hiểu các con số. Với máy tính, một hình ảnh trông giống như một mảng hai chiều các điểm ảnh (tưởng tượng nó như một bàn cờ khổng lồ) với một số trong mỗi ô. Trong ví dụ của chúng ta một điểm ảnh có giá trị 1 là trắng, và -1 là đen. Khi so sánh hai hình ảnh, nếu có giá trị điểm ảnh không trùng, thì những ảnh đó đối với máy tính là không khớp. Lý tưởng nhất là chúng ta muốn xem liệu chúng thuộc X hay O ngay cả khi chúng bị tịnh tiến, bị co, xoay hoặc biến dạng. Đây là khi ta cần CNNs.

**Feature**

Thay vì so sánh tổng thể bức ảnh, CNNs so sánh hình ảnh theo từng mảnh. Các mảnh mà nó tìm được gọi là các feature. Bằng cách tìm ở mức thô các feature khớp nhau ở cùng vị trí trong hai hình ảnh, CNNs nhìn ra sự tương đồng tốt hơn nhiều so với việc khớp toàn bộ bức ảnh.



Mỗi feature giống như một hình ảnh mini - một mảng hai chiều nhỏ. Các feature khớp với các khía cạnh chung của các bức ảnh. Trong trường hợp các hình ảnh X, các feature bao gồm các đường chéo và hình chữ thập, sẽ nắm bắt tất cả những đặc điểm quan trọng của hầu hết các hình ảnh X. Những feature này có lẽ sẽ khớp với phần cánh và phần trung tâm của bất kỳ hình ảnh một X nào.

Dưới đây là hình ảnh trực quan về kiến trúc LeNet học cách nhận diện hình ảnh. 

Mạng CNNs ở hình trên tương tự với kiến trúc LeNet và phân loại một hình ảnh đầu vào thành bốn nhóm với các tỉ lệ khác nhau : chó(dog), mèo(cat), thuyền(boat), chim(bird). Như trong hình mình họa, khi nhận tham số đầu vào là hình ảnh, mạng sẽ gán đúng xác suất cho thuyền(boat) là 0.94 – cao nhất trong 4 loại. Tổng của tất cả các xác suất trong lớp output phải bằng 1.

*Có 4 thành phần chính trong hoạt động của ConvNet đó là :*

*Covolution.*

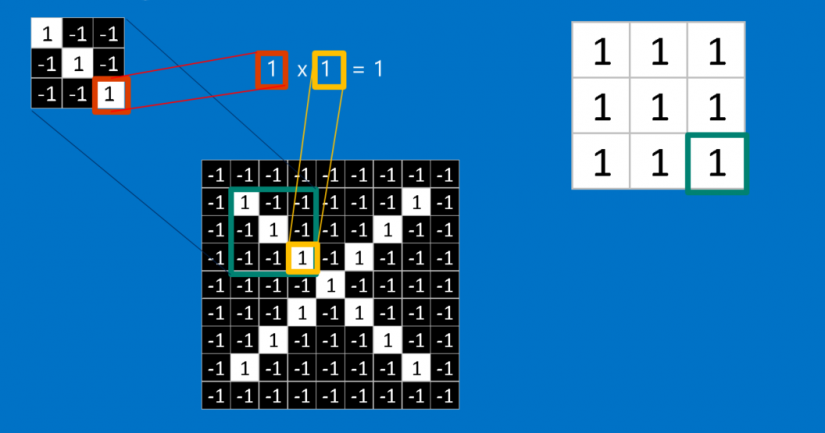
*Pooling or Sub Sampling.  
Non Linearity (ReLu).  
Classification ( Fully Connected Layer).*

Về cơ bản, mỗi hình ảnh có thể được biểu diện dưới dạng ma trận các giá trị pixels. Một ảnh cơ bản thì có 3 kênh R-G-B tương ứng với các màu cơ bản : Đỏ, Xanh lá, Xanh dương. Bạn có thể hình dung đó là 3 ma trận 2 chiều xếp chồng lên nhau, mỗi pixels có giá trị từ 0 – 255. Ở trong phạm vi bài viết, chúng ta sẽ giả sử ảnh chỉ có 2 màu trắng và đen.

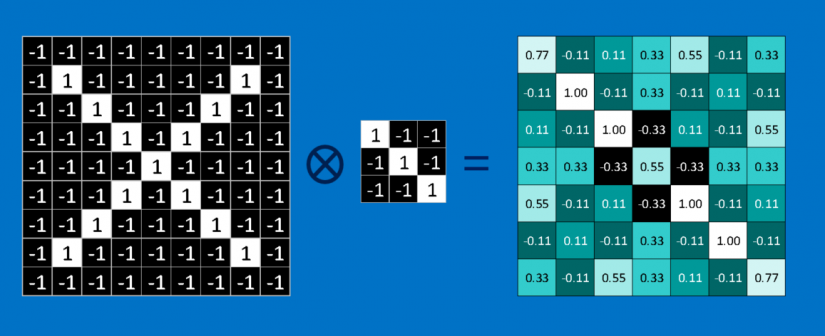
**Covolution (Tích chập) .**

Khi xem một hình ảnh mới, CNN không biết chính xác nơi các feature này sẽ khớp nên nó sẽ thử chúng khắp mọi nơi, ở mọi vị trí có thể. Khi tính toán sự khớp của một feature trên toàn bộ ảnh, chúng ta làm thành một filter (bộ lọc). Phần toán ta sử dụng để làm điều này được gọi là tích chập, từ đó mà Mạng Nơ-ron Tích chập(Convolutional Neural Networks) có tên là như vậy.

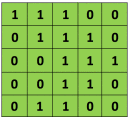
Để tính toán sự khớp của một feature đối với một mảnh của hình ảnh, ta chỉ cần nhân mỗi điểm ảnh trong feature với giá trị của điểm ảnh tương ứng trong mảnh hình ảnh. Sau đó cộng tổng lại và chia cho số lượng điểm ảnh trong feature. Nếu cả hai điểm ảnh màu trắng (giá trị 1) thì 1 x 1 = 1. Nếu cả hai đều là màu đen, thì ( -1 ) x ( -1 ) = 1. Dù bằng cách nào, mỗi điểm ảnh mà khớp thì đều cho ra kết quả 1. Tương tự như vậy, bất kỳ cái nào không khớp đều ra -1. Nếu tất cả các điểm ảnh trong feature đều khớp, thì cộng tổng lại rồi chia cho số điểm ảnh thì sẽ ra là 1. Tương tự, nếu không có điểm ảnh nào trong feature khớp với mảnh hình ảnh, thì kết quả là -1.



Để hoàn tất tích chập, chúng ta lặp lại quá trình này, xếp các feature với mọi mảnh hình ảnh có thể. Ta có thể lấy kết quả từ mỗi tích chập và tạo một mảng hai chiều mới từ đó, dựa vào vị trí của mỗi mảnh nằm trong hình ảnh. Bản đồ các phần khớp nhau này cũng là một phiên bản đã được lọc từ hình ảnh gốc. Nó là một bản đồ thể hiện nơi tìm thấy feature trong hình ảnh. Các giá trị gần 1 cho thấy sự khớp mạnh, các giá trị gần -1 cho thấy sự khớp mạnh với âm bản của feature, và các giá trị gần bằng 0 cho thấy không khớp với bất kỳ loại nào.



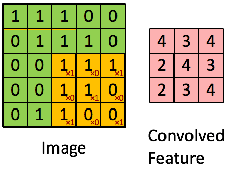
Để hiểu rõ hơn về cách hoạt động của CNNs. ConvNets được lấy tên từ các phép toán “phức tạp” “convolution”. Với mục đích trích suất các chức năng(features) từ hình ảnh đầu vào. Như đã đề cấp ở trên, mọi hình ảnh có thể được xem như là một ma trận của các pixels. Giả sử ta có một ma trận 5x5 đại diện cho một ảnh có giá trị pixels là 0 hoặc 1:



Ta cũng giả sử có một ma trận 3x3 như sau :



Ma trận này được gọi là convlution, ta sẽ tính ra một mảng mới tạo thành từ cách trượt (sliding) mảng 3x3 trên mảng 5x5 và lấy tổng của tích từng cặp số tương ứng để cho ra một ma trận như sau :



Trong thuật ngữ của CNNs, ma trận 3x3 còn được được xem như là một bộ lọc(filter) hoặc hạt nhân(kernel) hoăc là máy dò đặc tính(feature detector) và ma trận được hình thành bằng cách trượt bộ lọc và tính toán để cho ra một ma trận mới. Mục đích của bước convolution để lọc các dữ liệu, chỉ còn lại những đặt tính cơ bản của hình ảnh, ví dụ với một hình ảnh cho trước, bằng nhiều bộ lọc khác nhau sẽ cho ra một hình ảnh mới có nhưng đặc điểm giống với ảnh cũ với những tính năng khác nhau.

Ta có thể làm mờ bức ảnh ban đầu bằng cách lấy giá trị trung bình của các điểm ảnh xung quanh cho vị trí điểm ảnh trung tâm.

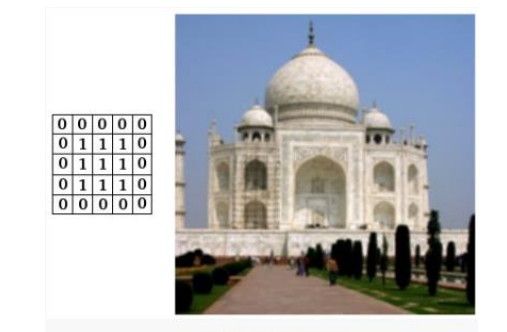


Figure 1. Ảnh mờ sau khi chập

Ngoài ra, ta có thể phát hiện biên cạnh bằng cách tính vi phân (độ dị biệt) giữa các điểm ảnh lân cận.

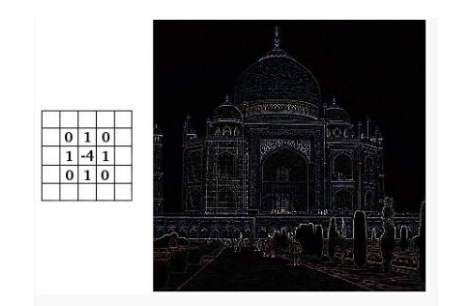
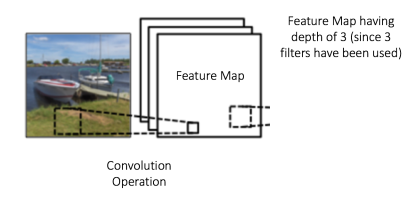


Figure 2. Ảnh được phát hiện biên(cạnh) sau khi chập

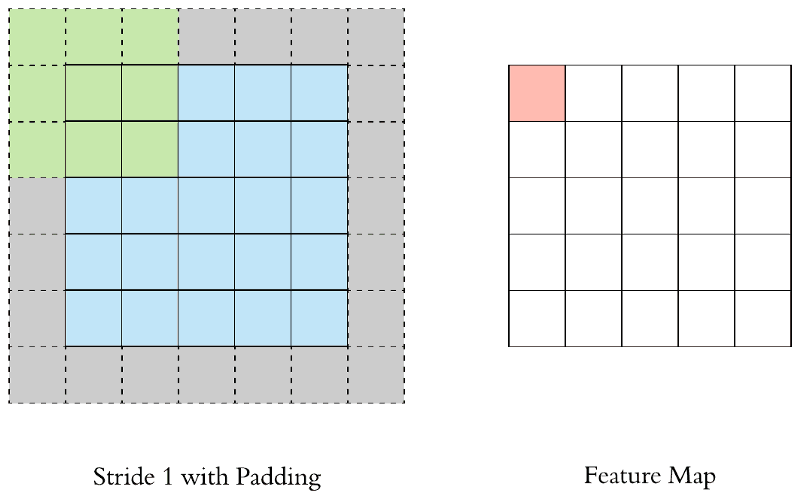
Về kích cỡ của bảng Convoled Feature, được điều khiển bởi 3 tham số và ta cần phải xác định:

Thứ nhất là Depth : tương ứng với số lượng bộ lọc sử dụng cho hoặc động convolution. Trong ví dụ minh họa, ta sử dụng 3 bộ lọc khác nhau để trích dữ liệu từ ảnh ban đầu.



Thứ hai là Stride :Stride là khoảng cách giữa 2 kernel khi quét. Với stride = 1, kernel sẽ quét 2 ô ngay cạnh nhau, nhưng với stride = 2, kernel sẽ quét ô số 1 và ô số 3. Bỏ qua ô ở giữa.Điều này nhằm tránh việc lặp lại giá trị ở các ô bị quét.

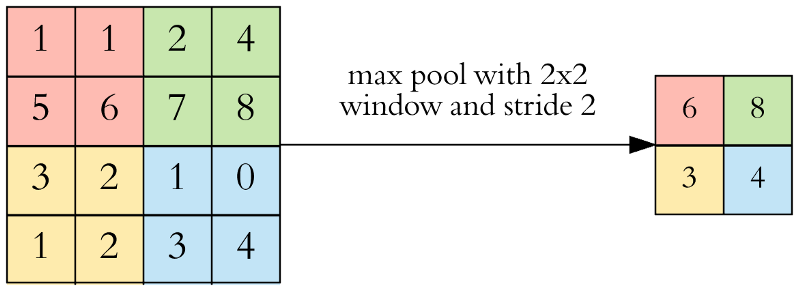
Thứ 3 là Zero-padding: Chúng ta chọn stride và size của kernel càng lớn thì size của feature map càng nhỏ, một phần lý do đó là bởi kernel phải nằm hoàn toàn trong input. Có một cách để giữ nguyên kích cỡ của feature map so với ban đầu. Đấy là **Padding**. Khi ta điều chỉnh padding = 1, tức là ta đã thêm 1 ô bọc xung quanh các cạnh của input, muốn phần *bọc*này càng dày thì ta cần phải tăng padding lên. Hãy nhìn vào ví dụ sau, ta xét padding = 1:



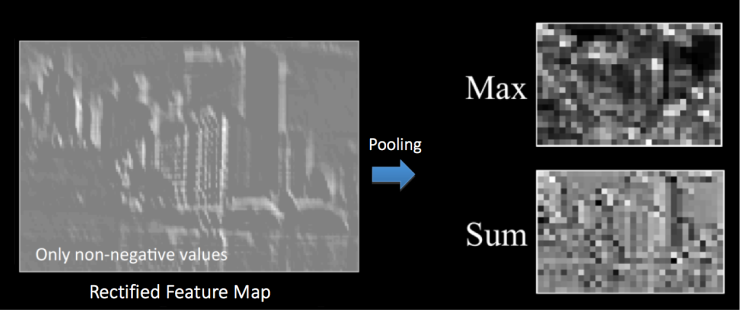
**Pooling (Giảm số chiều)**

Còn được gọi là Subsampling hay Downsampling, có chức năng làm giảm kích thước của bản đồ tính năng nhưng vẫn giữ lại được các thông tin quan trọng nhất. Có thể gồm nhiều loại khác nhau như Max, Average, Sum.

Mục đích của **pooling** rất đơn giản, nó làm giảm số hyperparameter mà ta cần phải tính toán, từ đó giảm thời gian tính toán, tránh overfitting. Loại pooling ta thường gặp nhất là *max pooling*, lấy giá trị lớn nhất trong một *pooling window*. Pooling hoạt động gần giống với convolution, nó cũng có 1 cửa sổ trượt gọi là *pooling window*, cửa sổ này trượt qua từng giá trị của ma trận dữ liệu đầu vào (thường là các feature map trong convolutional layer), chọn ra **một** giá trị từ các gía trị nằm trong cửa sổ trượt (với max pooling ta sẽ lấy giá trị lớn nhất).  
Hãy cùng nhìn vào ví dụ sau, tôi chọn pooling window có kích thước là 2 \* 2, stride = 2 để đảm bảo không trùng nhau, và áp dụng max pooling:



Cụ thể , với hình minh họa dưới đây, khi áp dụng Max Pooling và SumPooling sẽ cho ra 2 ma trận mới :

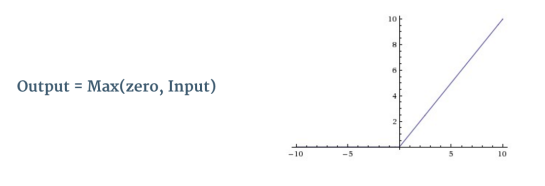


Mục đích của việc giảm kích thước ma trận là làm giảm số lượng phần tử và các thông số tính toán trong mạng, do đó có thể kiểm soát được tình trạng overfitting. Làm cho mạng không thay đổi với các biến đổi nhỏ.

**Non Linearity (ReLu).**

Đóng vai trò nhỏ nhưng quan trọng trong quá trình này là  Rectified Linear Unit hoặc ReLU. Toán của cái này cũng rất đơn giản -- bất cứ nơi nào có số âm, hoán đổi nó với 0. Điều này giúp CNN giữ vững sự tin cậy toán học bằng cách giữ các giá trị đã được học khỏi bị mắc kẹt gần 0 hoặc bị thổi bay về vô tận. Đó là thứ dầu mỡ bôi trơn CNNs- không hấp dẫn mấy, nhưng nếu không có nó, chúng sẽ không đi xa hơn được.

Đây là một hàm phi tuyến.



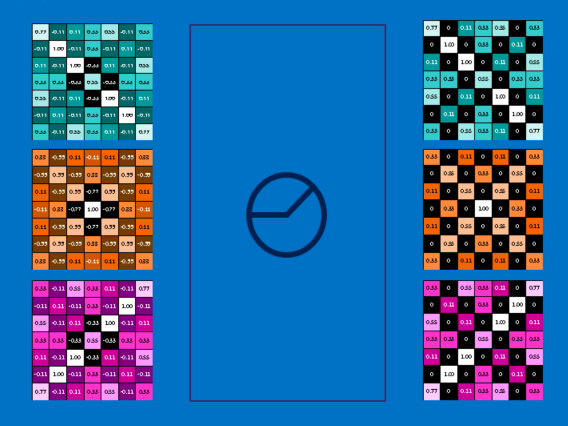
Hàm ReLU (Rectified Linear Unit) được sử dụng rộng rãi gần đây vì tính đơn giản của nó. Đồ thị của hàm ReLU được minh họa trên. Nó có công thức toán học  rất đơn giản. Ưu điểm chính của nó là:

+ ReLU được chứng minh giúp cho việc training các *Deep Networks* nhanh hơn rất nhiều (theo [Krizhevsky et al.](http://www.cs.toronto.edu/~fritz/absps/imagenet.pdf)). Sự tăng tốc này được cho là vì ReLU được tính toán gần như tức thời và gradient của nó cũng được tính cực nhanh với gradient bằng 1 nếu đầu vào lớn hơn 0, bằng 0 nếu đầu vào nhỏ hơn 0.

+ Mặc dù hàm ReLU không có đạo hàm tại , trong thực nghiệm, người ta vẫn thường định nghĩa  và khẳng định thêm rằng, xác suất để input của một unit bằng 0 là rất nhỏ.

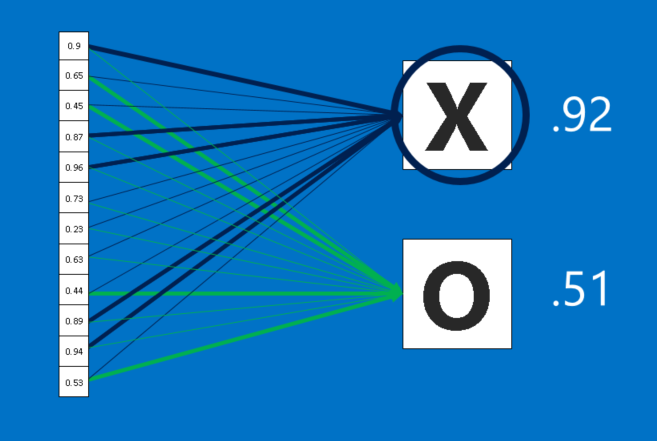
Một ví dụ : Hàm trên áp dụng cho mỗi pixels, thay thế tất cả các giá trị pixels âm trong bản đồ tính năng bới số 0. Mục đích đề cập đến tính không tuyến tính trong ConvNet, vì hầu hết các dữ liệu trong thế giới thực có dạng phi tuyến ( Convolution là một quá trình tuyến tính – phép nhân và phép cộng ma trận). ReLU được chứng minh giúp cho việc training các *Deep Networks* nhanh hơn rất nhiều.

Đầu ra của một layer ReLU có kích thước giống với đầu vào, chỉ là tất cả các giá trị âm được loại bỏ.



**Fully Connected Layer**

Là một lớp Perceptron Multi Layer sử dụng hàm kích hoạt SoftMax ở lớp output. Thuật ngữ "Kết nối hoàn toàn" ngụ ý rằng mỗi neuron trong lớp trước đó được nối với mỗi nơ-ron trên lớp tiếp theo. Output của các lớp Convolutional và Pooling đại diện cho những đặt điểm quang trọng của hình ảnh input. Mục đích của lớp Fully Connected là sử dụng các tính năng này để phân loại hình ảnh đầu vào thành các lớp khác nhau dựa trên bộ dữ liệu huấn luyện.



Đề xuất cải tiến

*Overfitting, dropout.*

Như ta đã biết overfitting là một vấn nạn đối với Machine Learning, đặc biệt là trong mạng NN. Khi muốn cho mô hình trở nên phức tạp hơn, ta thường tăng số lượng layer và số lượng unit của mỗi layer thay vì tăng bậc của feature như trước đây đã đề cập. Nhưng khi mô hình phức tạp hơn, overfitting bắt đầu xuất hiện, vì số lượng tham số trong mạng NN nhiều và có “độ sâu” khác nhau, vì vậy khi áp dụng chung 𝝺 cho tất cả các tham số sẽ yêu cầu số lần training lớn để tìm được 𝝺 phù hợp. Có một kĩ thuật dành riêng cho mạng NN và khá đơn giản, đó là Drop-out (một kĩ thuật khác được biết đến tương tự là Drop-Connect).   
Drop-out là một kĩ thuật Regularization để chống lại vấn đề overfitting. Cách Drop-out thực hiện là xoá bỏ một số unit trong các step training ứng với một giá trị xác suất **p** cho trước.

Quan sát lược đồ dưới đây, ta thấy rằng độ mất mát giữa tập training và tập testing càng ngày càng “xa nhau” sau vài ngàn lần lặp. Thuật toán chỉ chạy trên tập training và tối ưu hàm mất mát ở tập training. Nó chưa “nhìn thấy” tập testing trước đó nên không có gì ngạc nhiên rằng sao một lúc thì nó không còn ảnh hưởng với tập testing. Trong khi tập training, ta đã huấn luyện để càng lặp nhiều thì giá trị cross entropy càng giảm. Nhưng tập testing thì không như vậy, đơn giản vì ta không huấn luyện trên tập testing. Rõ rang thuật toán chạy rất tốt trên tập training ngưng lại sai khi ta sử dụng tập testing, đó là hiện tượng overfitting. Một kỹ thuật để khắc phục lỗi overfitting là “drop out”.



Khi áp dụng “dropout”, ở mỗi lần lặp lại quá trình tập huấn, bạn thả(drop) một cách ngẫu nhiên các neural. Bạn chọn một xác suất pkeep để một neural được giữ lại, thường là từ 50% đến 75% và sau mỗi lần lặp lại, bạn sẽ loại bỏ ngẫu nhiên các neural với trọng số và độ lệch của chúng. Số lượng các neural khác nhau sẽ được giảm xuống ở mỗi lần lặp và dĩ nhiên bạn cần phải tăng số lượng của các neural còn lại theo một tỉ lệ đảm bảo hoạt động ở lớp tiếp theo không thay đổi. Khi kiểm tra hiệu suất của mạng, dĩ nhiên bạn phải đặt tất cả các neural trở lại(pkeep =1).



* Drop-Out được áp dụng trên một layer của mạng NN với một xác suất **p** cho trước (ta có thể sử dụng nhiều Drop-Out khác nhau cho những layer khác nhau, nhưng trên 1 layer sẽ chỉ có 1 Drop-Out).
* Tại mỗi step trong quá trình training, khi thực hiện Forward Propagation (Lan truyền xuôi) đến layer sử dụng Drop-Out, thay vì tính toán tất cả unit có trên layer, tại mỗi unit ta “gieo xúc xắc” xem unit đó có được tính hay không dựa trên xác suất **p**. Với những unit được tính, ta tính toán bình thường còn với những unit không được tính giá trị tại unit đó = 0.
* Khi thực hiện tính toán trên mạng NN trong quá trình test (sử dụng mạng NN để dự đoán) thay vì làm như trên, ta thực hiện tính toán trên tất cả các unit nhưng trọng số trên mỗi connect đến các unit của layer được áp dụng Drop-Out được thay thế bằng giá trị của trọng số đó với xác suất **p** hay θ := θ \* **p**

Tại sao Drop-Out hiệu quả:

Đối với mạng NN, không giống như những kĩ thuật khác như Linear Regression hay Logistic Regression là đẩy bậc của feature lên mà ta tăng số lượng hidden layer và số lượng unit trong các layer lên. Nhưng đối khi do việc thêm vào quá nhiều hidden layer cũng như unit khiến cho mô hình phức tạp hơn mức cần thiết và khiến chúng bị overfitting. Chính vì vậy để tránh bị overfitting trong trường hợp này, ta cần phải giản lược mạng NN hiện có. Khi áp dụng Drop-Out vào 1 layer nào đó, thì thực tế tại mỗi training step mạng NN của ta chỉ còn 1 phần.

**Để tạo ra mô hình CNNs, cần phải xác định những siêu tham số(Hyperparameters) sau** :

* Đối với mỗi layer tích chập, bao nhiêu feature? bao nhiêu điểm ảnh trong mỗi feature?
* Đối với mỗi layer pooling, kích cỡ ô vuông cửa sổ như thế nào? duyệt mỗi bước bao nhiêu?
* Đối với mỗi các layer được kết nối đầy đủ thêm vào, bao nhiêu nơron ẩn?

**Đề xuất mô hình mạng CNNs dựa theo LeCun**

Image 28x28x1

🡳

[5x5x32]

Max pool 2x2

🡳

[5x5x64]

Max pool 2x2

🡳

Fully connected layer [1024]

🡳

Dropout 0.75

🡳

Fully connected layer [10]

🡳

Softmax

Mô tả chi tiết:

Với ảnh đầu vào, ta test reshape về kích cỡ 28x28x1. Đối với tập MNIST thì không cần, vì ảnh đã có kích cỡ ban đầu là 28x28x1.

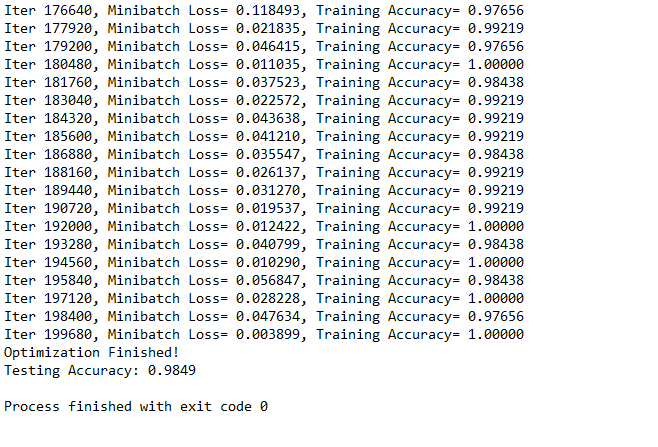
Dùng max pool với cửa sổ trượt là 2x2 để tạo lớp convolution thứ nhất (5x5 conv, 1 input, 32 outputs)

Tiếp tục tạo với lớp convolution thứ hai (5x5 conv, 32 inputs, 64 outputs)

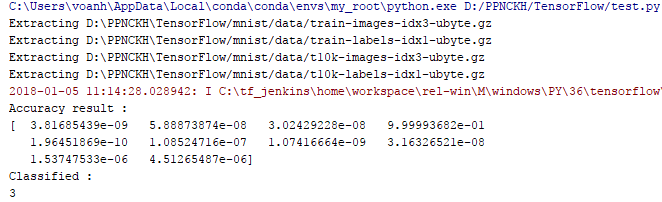
Sau đó tạo 2 lớp Fully Connected layer với đầu ra lần lượt là 1024 và 10. Với hệ số dropot là 0.75.

Cuối cùng là softmax để tính xác suất của từng class, ở đây là 10 class.

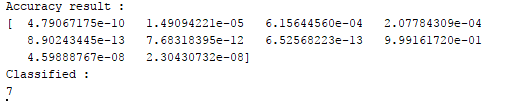
# Kết quả thực nghiệm.



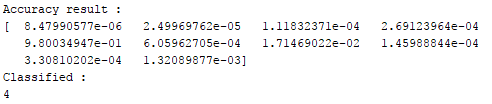






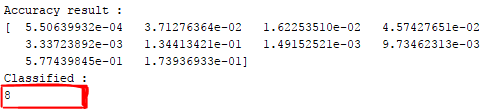






Tuy nhiên cũng có một số trường hợp nhận diện sai





# Kết luận và phát triển

Mạng nơ ron là một công cụ hiệu quả trong lĩnh vực phân loại. Ta có thể áp dụng mạng convolutional ở trên cho nhiều tập dữ liệu khác nhau. Tuy nhiên cần phải điều chỉnh lại các thông số cần thiết như dropout. Vẫn chưa có một sự tường minh nào cho việc hoạt động của các tầng hidden layer trong mô hình CNN, nên chúng tôi nghĩ rằng việc chọn thông số phải dựa trên thực nghiệm nhiều lần mới rút ra được con số tối ưu.

# Nguồn tham khảo.

[[Recognizing Handwritten Digits and Characters Vishnu Sundaresan Stanford University vishnu@stanford.edu Jasper Lin Stanford University jasperlin@stanford.edu]](http://cs231n.stanford.edu/reports/2015/pdfs/vishnu_final.pdf)

[[CHAPTER 1. Using neural nets to recognize handwritten digits By Michael Nielsen]](http://neuralnetworksanddeeplearning.com/chap1.html)

[[ImageNet Classification with Deep Convolutional Neural Networks]](http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf)

[[Convolutional Neural Network Committees for Handwritten Character Classification]](http://ieeexplore.ieee.org/abstract/document/6065487/?part=1)

[[Convolutional Networks for Images, Speech, and Time-Series]](https://www.researchgate.net/profile/Yann_Lecun/publication/2453996_Convolutional_Networks_for_Images_Speech_and_Time-Series/links/0deec519dfa2325502000000.pdf)

[[CONVOLUTIONAL NEURAL NETWORK]](http://socs.binus.ac.id/2017/02/27/convolutional-neural-network/)

[[A Quick Introduction to Neural Networks]](https://ujjwalkarn.me/2016/08/09/quick-intro-neural-networks/)

[[An Intuitive Explanation of Convolutional Neural Networks]](https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/)

[[Neural networks là gì?]](http://blog.duyet.net/2016/03/neural-networks.html#.WkhrJN-WbIV)

[[Olavi Stenroos Object detection from images using convolutional neural networks]](https://aaltodoc.aalto.fi/bitstream/handle/123456789/27960/master_Stenroos_Olavi_2017.pdf?sequence=1)

[[Deep Learning for Computer Vision: A comparison between Convolutional Neural Networks and Hierarchical Temporal Memories on object recognition tasks]](http://amslaurea.unibo.it/9095/1/Vincenzo_Lomonaco_tesi.pdf)

[[Conv Nets: A Modular Perspective]](http://colah.github.io/posts/2014-07-Conv-Nets-Modular/)

[[HANDWRITTEN DIGITS RECOGNITION USING GOOGLE TENSORFLOW WITH PYTHON]](http://dataaspirant.com/2017/05/03/handwritten-digits-recognition-tensorflow-python/)