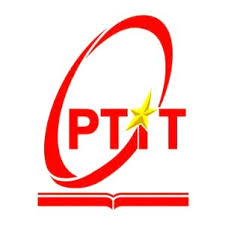
**BỘ THÔNG TIN VÀ TRUYỀN THÔNG**

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**----------o0o----------**

****

**ĐỒ ÁN**

**TỐT NGHIỆP ĐẠI HỌC**

**Đề tài:**

**NGHIÊN CỨU HỆ THỐNG NHẬN DẠNG KÍ TỰ**

**SỬ DỤNG MÔ HÌNH HỌC SÂU LENET-5**

**GIÁO VIÊN HƯỚNG DẪN : TS. NGUYỄN XUÂN SÂM**

**SINH VIÊN THỰC HIỆN : LÊ THỊ THU**

**MÃ SỐ SINH VIÊN : N16DCCN158**

**LỚP : D16CQCP01-N**

**KHÓA : 2016 – 2021**

**HỆ : ĐẠI HỌC CHÍNH QUY**

**TP.HCM, THÁNG 12, NĂM 2020**

**LỜI CẢM ƠN**

Trước tiên, em xin gửi lời tri ân sâu sắc đến các thầy cô trường Học Viện Công Nghệ Bưu Chính Viễn thông cơ sở tại TP.HCM đã tận tình dẫn dắt và truyền đạt cho em rất nhiều kiến thức quý báu trong các học kỳ vừa qua.

Đặc biệt, em xin gửi lời cảm ơn chân thành đến TS Nguyễn Xuân Sâm. Thầy đã hướng dẫn tận tình, truyền đạt kiến thức, chỉ bảo cho em trong suốt thời gian học tập và thực hiện đề tài thực tập, đề tài tốt nghiệp. Trong quá trình làm đề tài thực sự có rất nhiều khó khăn đối với em, thầy đã luôn nhiệt hướng dẫn, động viên em. Kính chúc thầy và gia đình nhiều sức khoẻ và thành công trong cuộc sống.

Con xin gửi lòng biết ơn và sự kính trọng đến cha mẹ, gia đình đã luôn động viên, tạo điều kiện tốt nhất cho con học tập. Và xin chân thành cảm ơn tất cả các bạn đã luôn sát cánh, giúp đỡ, động viên mình trong những thời điểm khó khăn, tiếp thêm động lực và ý chí giúp vượt qua khó khăn trong suốt quá trình học tập tại trường.

Tuy nhiên, vì thời gian và kiến thức còn hạn chế em đã cố gắng rất nhiều để hoàn thành đề tài này, nhưng trong quá trình làm bài em không tránh khỏi những thiếu sót, kính mong được lời nhận xét và góp ý của thầy cô.

Kính chúc tất cả Thầy cô, và các Anh/Chị lời chúc sức khỏe và thành công trong công việc cũng như trong cuộc sống.

Một lần nữa, em xin chân thành cảm ơn !

Sinh viên thực hiện

**MỤC LỤC**

[**MỞ ĐẦU 1**](#_Toc59110317)

[**CHƯƠNG 1. TỔNG QUAN MÔ HÌNH NHẬN DẠNG KÝ TỰ SỬ DỤNG MÔ HÌNH HỌC SÂU LENET-5 3**](#_Toc59110327)

[1.1 Giới thiệu chung 3](#_Toc59110328)

[1.2 Mô hình tổng quan 4](#_Toc59110329)

[1.2.1 Một số khái niệm 4](#_Toc59110330)

[Chuyển đổi ảnh màu sang ảnh xám bằng phương pháp GrayScale. 5](#_Toc59110331)

[1.2.2 Mạng CNNs (Lenet5) 6](#_Toc59110332)

[1.2.3 Kết quả 7](#_Toc59110333)

[1.3 Một số khó khăn trong nghiên cứu mô hình học sâu CNN 7](#_Toc59110334)

[1.4 Ứng dụng và đóng góp 7](#_Toc59110335)

[**CHƯƠNG 2. CƠ SỞ LÝ THUYẾT 8**](#_Toc59110336)

[2.1. AI ( Trí tuệ nhân tạo) 9](#_Toc59110337)

[2.2 Machine learning. 9](#_Toc59110338)

[2.2.1 Khái niệm 9](#_Toc59110339)

[2.2.2 Một số thuật toán Machine learning. 10](#_Toc59110340)

[2.3 Deep learning 11](#_Toc59110341)

[2.4 Mạng Nơ ron 12](#_Toc59110342)

[2.4.1 Giới thiệu về mạng Nơ ron 12](#_Toc59110343)

[2.4.2 Một số hàm truyền thông dụng 14](#_Toc59110344)

[2.4.3 Một số kiểu mạng Nơ ron 16](#_Toc59110345)

[2.4.4 Các phương pháp huấn luyện mạng Nơ ron 17](#_Toc59110346)

[2.5 Mạng Nơ ron tích chập 19](#_Toc59110347)

[2.5.1 Định nghĩa 19](#_Toc59110348)

[2.5.2 Tích chập 20](#_Toc59110349)

[2.5.3 Lan truyền tiến 23](#_Toc59110350)

[2.5.4 Thuật toán lan truyền ngược (Back-Propagation) 23](#_Toc59110351)

[2.5.5 Activation functions 25](#_Toc59110352)

[2.5.6 Kiến trúc mô hình học sâu lenet5 27](#_Toc59110353)

[**CHƯƠNG 3. XÂY DỰNG CHƯƠNG TRÌNH THỬ NGHIỆM 39**](#_Toc59110354)

[3.1 Mô hình 39](#_Toc59110355)

[3.2 Cài đặt chương trình 39](#_Toc59110356)

[3.2.1 Xây dựng Model 39](#_Toc59110357)

[3.2.2 Xây dựng App demo 43](#_Toc59110358)

[3.2.3 Kết quả một số trường hợp dự đoán 45](#_Toc59110359)

[**KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 48**](#_Toc59110360)

[**PHỤ LỤC 49**](#_Toc59110364)

[**TÀI LIỆU THAM KHẢO 50**](#_Toc59110365)

**DANH MỤC HÌNH**

[Hình 1.1 Mô hình tổng quan nhận dạng 4](#_Toc59105467)

[Hình 1.2 Ma trận điểm ảnh của ảnh xám 5](#_Toc59105468)

[Hình 2.1 Mối liên hệ giữa AI, machine learning và deep learning 8](#_Toc59105469)

[Hình 2.2 Các điểm gần mặt phân cách nhất của hai classes được khoanh tròn. 10](#_Toc59105470)

[Hình 2.3 Tập giá trị của dữ liệu 11](#_Toc59105471)

[Hình 2.4 Cấu tạo 1 nơ ron 13](#_Toc59105472)

[Hình 2.5 – Hàm đồng nhất (Identity function) 14](#_Toc59105473)

[Hình 2.6 – Hàm bước nhị phân (Binary step function) 14](#_Toc59105474)

[Hình 2.7 – Hàm Sigmoid 15](#_Toc59105475)

[Hình 2.8 – Hàm sigmoid lưỡng cực 15](#_Toc59105476)

[Hình 2.9 – Mạng nơron truyền thẳng nhiều lớp (Feed-forward neural network) 16](#_Toc59105477)

[Hình 2.10 – Mạng nơron hồi quy (Recurrent neural network) 17](#_Toc59105478)

[Hình 2.11 – Mô hình Học có thầy (Supervised learning model) 18](#_Toc59105479)

[Hình 2.12: Dòng dữ liệu trong LeNet 5. Đầu vào là một chữ số viết tay, đầu ra là một xác suất đối với 10 kết quả khả thi. 20](#_Toc59105480)

[Hình 2.13 Minh họa tích chập 20](#_Toc59105481)

[Hình 2.14 Ảnh mờ sau khi chập 21](#_Toc59105482)

[Hình 2.15 Ảnh được phát hiện biên sau khi chập 23](#_Toc59105483)

[Hình 2.16: Hàm sigmoid (trái) và tanh (phải). 25](#_Toc59105484)

[Hình 2.17: Hàm ReLU và tốc độ hội tụ khi so sánh với hàm tanh. 26](#_Toc59105485)

[Hình 2.18: Mô hình Softmax Regression dưới dạng Neural network. 27](#_Toc59105486)

[Hình 2.19 Mô hình mạng CNNs Lenet5 28](#_Toc59105487)

[Hình 2.20 Input Nơ ron 29](#_Toc59105488)

[Hình 2.21 Trượt kernel trên ma trân ảnh 30](#_Toc59105489)

[Hình 2.22 Các loại Pooling 33](#_Toc59105490)

[Hình 3.1 Mô hình nhận dạng kí tự sử dụng DeepLearning (Lenet5) 39](#_Toc59105491)

[Hình 3.2 Kiến trúc model 41](#_Toc59105492)

[Hình 3.3 Đánh giá model 41](#_Toc59105493)

[Hình 3.4 Biểu đồ thể hiện độ chính xác(trái), mất mát (phải) 42](#_Toc59105494)

# DANH MỤC TỪ VIẾT TẮT

|  |  |
| --- | --- |
| Từ viết tắt | Ý nghĩa |
| CNN | Convolutional neural network |
| kNN | k-nearest neighbors |
| SVM | Support vector machine |
| GPU | Máy chủ |
| RGB | Ảnh màu Red Green Blue |

# MỞ ĐẦU

1. **Tính cấp thiết của đề tài**

Nhận dạng là lĩnh vực được các nhà khoa học rất quan tâm để giải quyết các yêu cầu trong cuộc sống hiện nay, có nhiều lĩnh vực nhận dạng như nhận dạng tín hiệu, nhận dạng tiếng nói hay nhận dạng ảnh. Vấn đề nhận dạng ký tự nói chung và ký tự viết tay nói riêng là một vấn đề thách thức đối với những nhà nghiên cứu.

Ví dụ nhỏ như chữ số viết tay xuất hiện ở hầu hết trong các công việc của các cơ quan, nhà máy, xí nghiệp, trường học. Trong các trường phổ thông hiện nay, đều có một bộ phận quản lý điểm để thực hiện các khâu tiếp nhận và nhập vào máy tính bảng điểm viết tay của giáo viên bộ môn, công tác này luôn chiếm nhiều thời gian và đôi khi không đảm bảo tiến độ hoạt động của nhà trường.

Để nhận dạng kí tự, có nhiều phương pháp và kỹ thuật khác nhau như: logic mờ, giải thuật di truyền, mô hình xác xuất thống kê, mô hình mạng nơ ron. Đã có nhiều công trình nghiên cứu về nhận dạng kí tự, đặc biệt là chữ số viết tay đạt hiệu quả cao, tuy nhiên, các ứng dụng vẫn chưa đáp ứng hoàn toàn các yêu cầu của người dùng.

Hiện nay với sự phát triển không ngừng của máy tính, phương pháp Deep Learning ra đời đã đáp ứng cơ bản trong việc nhận dạng và xử lý ảnh. Deep Learning là một thuật toán dựa trên một số ý tưởng từ não bộ tới việc tiếp thu nhiều tầng biểu đạt, cả cụ thể lẫn trừu tượng, qua đó làm rõ nghĩa của các loại dữ liệu.

Deep Learning được ứng dụng trong nhận diện hình ảnh, nhận diện giọng nói, xử lý ngôn ngữ tự nhiên. Hiện nay rất nhiều các bài toán nhận dạng sử dụng Deep Learning để giải quyết do Deep Learning có thể giải 2 quyết các bài toán với số lượng lớn, kích thước đầu vào lớn với hiệu năng cũng như độ chính xác vượt trội so với các phương pháp phân lớp truyền thống.

Với những lý do trên, tôi chọn nghiên cứu đề tài : “Nghiên cứu hệ thống nhận dạng ký tự sử dụng mô hình học sâu lenet-5”

1. **Mục tiêu của nghiên cứu**

Mục tiêu chính của đề tài là nghiên cứu hệ thống nhận dạng ký tự sử dụng mô hình học sâu lenet-5

1. **Đối tượng và phạm vi nghiên cứu**
   1. Đối tượng

* Các kí tự chữ số viết tay
* Các phương pháp, giải thuật về nhận dạng
* Kĩ thuật deep learning
* Mạng Nơ ron CNN
  1. Phạm vi nghiên cứu
* Các chữ số tự nhiên từ 0 đến 9

1. Các phương pháp nghiên cứu
   1. Phương pháp lý thuyết

* Các tài liệu về cơ sở lý thuyết: xử lý ảnh số, lọc trích ảnh số, xác định biên, rút trích số, nhận dạng ký tự số.
* Nghiên cứu mạng Nơ ron nhận dạng chữ viết tay
* Phương pháp DeepLearming
* Một số tài liệu lập trình
  1. Phương pháp thực nghiệm
* Xây dựng chương trình thử nghiệm.
* Kiểm thử tính hiệu quả của chương trình với các chữ số khác nhau.

1. Ý nghĩa của đề tài

Về mặt khoa học: Đề tài sẽ cung cấp về mặt lý thuyết để làm rõ về các phương pháp và kĩ thuật nhận dạng kí tự nói chung và chữ số viết tay nói riêng

Về mặt thực tiễn: Góp phần hỗ trợ nhập văn bản, scan văn bản. Tiền xử lý trong các ứng dụng trên phần mềm như: game viết chữ của trẻ em, phần mềm hỗ trợ viết tay cho người khiếm thị, quét biển số xe, quét mã vạch… khi áp dụng với camera, detect được phản hồi khách hang qua bình luận, phần mềm sửa lỗi chính tả,…

# CHƯƠNG 1. TỔNG QUAN MÔ HÌNH NHẬN DẠNG KÝ TỰ SỬ DỤNG MÔ HÌNH HỌC SÂU LENET-5

* 1. **Giới thiệu chung**

Trong những năm gần đây, chúng ta đã được chứng kiến nhiều thành tựu vượt bậc trong lĩnh vực xử lý ảnh (image processing). Các hệ thống xử lý ảnh lớn như Facebook, Google hay Amazon đã đưa vào sản phẩm của mình những chức năng thông minh như nhận diện khuôn mặt người dùng, phát triển xe hơi tự lái hay drone giao hàng tự động. Mạng nơ-ron tích chập (Convolutional Neural Network - CNN) là một trong những mô hình học sâu (Deep Learning) tiên tiến giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay.

Các bài toán nhận dạng đang được ứng dụng trong thực tế hiện nay tập trung vào nhận dạng mẫu, nhận dạng tiếng nói và nhận dạng ký tự, chữ viết… Nhận dạng ký tự là bài toán được quan tâm rất nhiều vì nó là một trong các yêu cầu trong nhiều ứng dụng thực tế. Các ứng dụng của nhận dạng kí tự đã và đang được ứng dụng vào đời sống như phục vụ cho công việc tự động hóa đọc tài liệu, tăng tốc độ và hiệu quả nhập thông tin vào máy tính. Nhận dạng ký tự có thể phục vụ cho các ứng dụng đọc và xử lý các chứng từ, hóa đơn, phiếu ghi, scan văn bản...

Các hệ thống nhận dạng trước đây như (LeCun et al., 1998) [1], (Simard et al., 2003)[2], (Kégl & BusaFekete, 2009)[3] đều sử dụng các đặc trưng cơ bản từ ảnh ký tự như đường biên, cạnh, độ dày, giá trị mức xám, haar-like, với các xử lý đặc thù như lấy mẫu, dao động các điểm ảnh, biến đổi ảnh, thêm dữ liệu ảo. Sau đó hệ thống nhận dạng huấn luyện các mô hình học tự động như k láng giềng (kNN)[4], mạng Nơ ron, máy véc tơ hỗ trợ (SVM)[5], boosting. Các hệ thống này đã cơ bản đáp ứng yêu cầu của người sử dụng.

* 1. **Mô hình tổng quan**

CNN

(lenet-5)

Hình 1.1 Mô hình tổng quan nhận dạng

* + 1. **Một số khái niệm**
* Ảnh và điểm ảnh

Gốc của ảnh (ảnh tự nhiên) là ảnh liên tục về không gian và độ sáng. Để xử lý bằng máy tính (số), ảnh cần phải được số hóa. Số hóa ảnh là sự biến đổi gần đúng môt ảnh liên tục thành một tập điểm phù hợp với ảnh thật về vị trí (không gian) và độ sáng (mức xám). Khoảng cách giữa các điểm ảnh đó được thiết lập sao cho mắt người không phân biệt được ranh giới giữa chúng. Mỗi một điểm như vậy gọi là điểm ảnh (pixel).

* Độ phân giải của ảnh

Độ phân giải (Resolution) là mật độ điểm ảnh được ấn định trên một ảnh số được hiển thị. Theo định nghĩa, khoảng cách giữa các điểm ảnh phải được chọn sao cho mắt người vẫn thấy được sự lien tục của ảnh. Việc lựa chọn khoảng cách thích hợp tạo nên một mật độ phân bổ, đó chính là độ phân giải và được phân bố theo trục x và y trong không gian hai chiều.

* Mức xám của ảnh

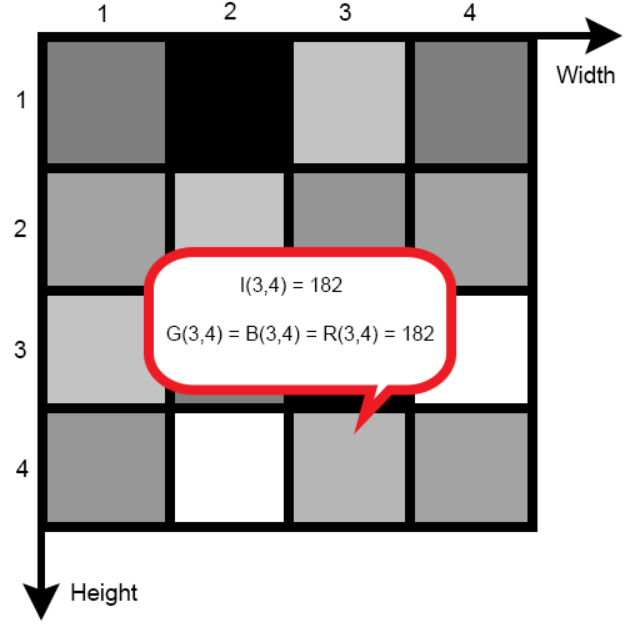
Là kết quả của sự biến đổi tương ứng một giá trị độ sáng của một điểm ảnh với một giá trị nguyên dương. Thông thường nó xác định trong khoảng 0…255. Tùy thuộc vào giá trị xám mà mỗi điểm ảnh được biểu diễn trên 1, 4, 8, 24 hay 32 bit.

**Chuyển đổi ảnh màu sang ảnh xám bằng phương pháp GrayScale.**

Ảnh trước khi đưa vào model để nhận dạng cần được tiền xử lí trước. Model được train cho các ảnh xám có kích thước 28x28 nên ta phải đưa ảnh màu thành ảnh xám bằng phương pháp Grayscale

GrayScale là một hệ thống màu có mô hình màu đơn giản nhất với 256 cấp độ xám biến thiên từ màu đen đến màu trắng.

Trong không gian màu RGB, để có 1 ảnh xám cần có phải có giá trị kênh màu Red(x, y) = Green(x, y) = Blue(x, y) (với x, y lần lượt là tọa độ của điểm ảnh).



Hình 1.2 Ma trận điểm ảnh của ảnh xám

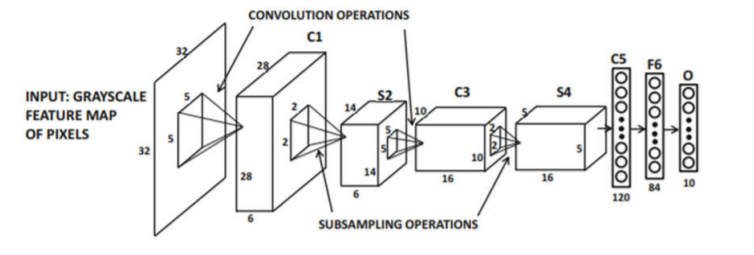
Để chuyển ảnh màu sang ảnh xám ta dùng công thức sau:

**I(x, y) = ( 2 \* Red(x, y) + 5 \* Green(x, y) + 1 \* Blue(x, y) ) / 8** (1)

Trong đó:

* I(x, y): cường độ sáng tại điểm ảnh (x, y) của ảnh xám
* Red(x, y): giá trị của kênh màu Red (Đỏ) tại điểm ảnh (x, y) của ảnh màu (RGB)
* Green(x, y): giá trị của kênh màu Green (Xanh lá cây) tại điểm ảnh (x, y) của ảnh màu (RGB).
* Blue(x, y): giá trị của kênh màu Blue (Xanh lơ) tại điểm ảnh (x, y) của ảnh màu (RGB).

Sau khi chuyển sang ảnh xám thì ta phải resize ảnh xám về ma trận (28,28,1)

* + 1. **Mạng CNNs (Lenet5)**

Kiến trúc mô hình lenet5: hai lớp Convolution, 2 lớp Pooling, các lớp deep (Flatten, Dense)

Về kỹ thuật, mô hình CNN để training và kiểm tra, mỗi hình ảnh đầu vào

sẽ chuyển nó qua 1 loạt các lớp tích chập với các bộ lọc (Kernals), tổng hợp lại các lớp được kết nối đầy đủ (Full Connected) và áp dụng hàm Softmax để phân loại đối tượng có giá trị xác suất giữa 0 và 1.

Cấu trúc đầy đủ của mô hình lenet5 như sau:

* Input shape 28x28x3
* Convolution layer 1 : Kernel 5x5x3 , stride = 1,no padding, number filter = 6 ,output = 28x28x6.
* Maxpooling layer : pooling size 2x2,stride = 2,padding = “same”,output = 14x14x6.
* Convolution layer 2 : kernel 5x5x6,stride = 1, no padding, number filter = 16,output = 10x10x16.
* Maxpooling layer : pooling size = 2x2, stride = 2, padding =”same”,output = 5x5x16.
* Flatten output = 5x5x16 = 400
* Fully connected 1 : output = 120
* Fully connected 2 : output = 84
* Softmax layer, output = 10 (10 digits). Nhược điểm của LeNet là mạng còn rất đơn giản và sử dụng sigmoid (or tanh) ở mỗi convolution layer mạng tính toán rất chậm.

Chúng ta sẽ tìm hiểu mô hình CNN lenet5 kĩ hơn trong nội dung của phần sau.

* + 1. **Kết quả**

Sauk hi đưa ảnh kí tự đi qua mạng CNNs thì sẽ thu được kết quả là dự đoán với sác xuất cao nhất ứng với lable có trong tập dữ liệu train

* 1. **Một số khó khăn trong nghiên cứu mô hình học sâu CNN**

Chưa thể hiểu hết về các layers của mô hình. Train model cần sử dụng GPU để tốc độ tính toán nhanh, train nhanh hơn, mà GPU lại phải đầu tư nhiều hơn về kinh tế (thuê phần mềm tích hợp GPU, hoặc mua PC cấu hình phù hợp để train model).

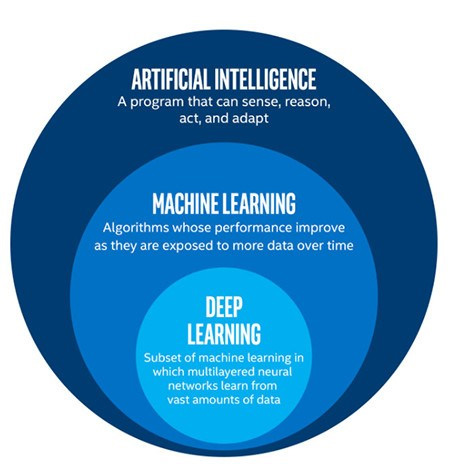
* 1. **Ứng dụng và đóng góp**

Hệ thống nhận dạng kí tự được ứng dụng trên rất nhiều lĩnh vực (y tế, giáo dục, ngân hàng,…). Hệ thống dùng để scan văn bản, hỗ trợ nhập điểm trong trường học, phục vụ game viết chữ cho trẻ em, hỗ trợ người bị bênh về mắt, hỗ trợ những người lười đánh máy,….

# CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

Trong những năm vừa qua, cùng với sự bùng nổ của cuộc [cách mạng công nghiệp 4.0](https://quantrimang.com/cach-mang-cong-nghiep-4-0-la-gi-147384), các thuật ngữ như [trí tuệ nhân tạo](https://quantrimang.com/ai) (AI), học máy (machine learning) và học sâu (deep learning) đang dần trở nên phổ biến và trở thành những khái niệm mà các công dân của kỷ nguyên 4.0 buộc phải nắm được.

Có thể giải thích mối liên hệ giữa 3 khái niệm này bằng cách tưởng tượng chúng như những vòng tròn, trong đó AI - ý tưởng xuất hiện sớm nhất - là vòng tròn lớn nhất, tiếp đến là machine learning - khái niệm xuất hiện sau, và cuối cùng là deep learning - thứ đang thúc đẩy sự bùng phát của AI hiện nay - là vòng tròn nhỏ nhất.



Hình 2.1 Mối liên hệ giữa AI, machine learning và deep learning

Việc xây dựng lên một hệ thống AI đương nhiên là cực kì phức tạp, tuy nhiên việc hiểu được nó lại không đến nỗi khó khăn như vậy. Hầu hết các trí thông minh nhân tạo hiện tại chỉ là những máy đoán thực sự tốt (guessing machines, tương tự như bộ não của chúng ta). Bạn bạn cung cấp cho hệ thống một nhóm dữ liệu (chẳng hạn như các chữ số từ 1 đến 10) và yêu cầu hệ thống tạo mô hình (x + 1, bắt đầu từ 0) và đưa ra các dự đoán. (Con số tiếp theo sẽ là mười một). Không có phép thuật nào cả, đây là công việc mà bộ não của con người làm mỗi ngày: sử dụng những gì chúng ta biết để đoán về những điều chúng ta chưa biết.

Điều khiến AI khác với các chương trình máy tính khác là thay vì phải lập ra những trình chương trình cụ thể cho mỗi trường hợp, chúng ta hoàn toàn có thể dạy học cho AI (machine learning), và nó cũng có khả năng tự học (deep learning).

## 2.1. AI ( Trí tuệ nhân tạo)

AI có thể được định nghĩa như một [ngành của khoa học máy tính](https://quantrimang.com/truong-dai-hoc-ve-khoa-hoc-may-tinh-tot-nhat-the-gioi-147491) liên quan đến việc tự động hóa các hành vi thông minh. AI là một bộ phận của khoa học máy tính và do đó nó phải được đặt trên những nguyên lý lý thuyết vững chắc, có khả năng ứng dụng được của lĩnh vực này. Nói nôm na cho dễ hiểu: đó là trí tuệ của máy móc được tạo ra bởi con người. Trí tuệ này có thể tư duy, suy nghĩ, học hỏi,... như trí tuệ con người. Xử lý dữ liệu ở mức rộng lớn hơn, quy mô hơn, hệ thống, khoa học và nhanh hơn so với con người.

Tuy nhiên hiện nay, công nghệ AI vẫn đang còn rất nhiều hạn chế. Đơn cử như [Alexa](https://quantrimang.com/cach-thiet-lap-tro-ly-ao-alexa-tren-android-150141)- một quản gia tuyệt vời, một trong những biểu tượng phổ biến nhất về ứng dụng của trí thông minh nhân tạo nhưng vẫn không thể vượt qua bài kiểm tra Turing.

Tóm lại, những gì chúng ta đang thực hiện với AI hiện nay nằm trong khái niệm “AI hẹp” (Narrow AI). Công nghệ này có khả năng thực hiện các nhiệm vụ cụ thể một cách tương tự, hoặc tốt hơn con người. Ví dụ về “AI hẹp” trong thực tế như công nghệ phân loại hình ảnh của Pinterest hay nhận diện khuôn mặt để tag bạn bè trên [Facebook](https://quantrimang.com/facebook).

Những công nghệ này thể hiện một số khía cạnh của trí thông minh con người, nhưng làm thế nào để được như vậy? Trí tuệ đó đến từ đâu? Hãy cùng đến với vòng tròn tiếp theo: học máy (machine learning).

## 2.2 Machine learning.

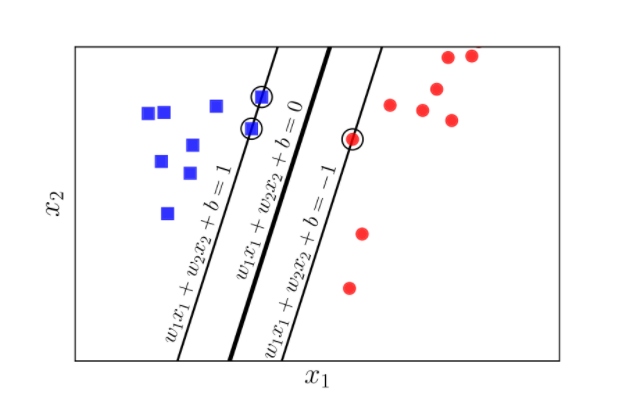
### 2.2.1 Khái niệm

Machine Learning là một thuật ngữ rộng để chỉ hành động bạn dạy máy tính cải thiện một nhiệm vụ mà nó đang thực hiện. Cụ thể hơn, machine learning đề cập tới bất kỳ hệ thống mà hiệu suất của máy tính khi thực hiện một nhiệm vụ sẽ trở nên tốt hơn sau khi hoàn thành nhiệm vụ đó nhiều lần. Hay nói cách khác, khả năng cơ bản nhất của machine learning là sử dụng thuật toán để phân tích những thông tin có sẵn, học hỏi từ nó rồi đưa ra quyết định hoặc dự đoán về một thứ gì đó có liên quan. Thay vì tạo ra một phần mềm với những hành động, hướng dẫn chi tiết để thực hiện một nhiệm vụ cụ thể, máy tính được “huấn luyện” bằng cách sử dụng lượng dữ liệu và các thuật toán để học cách thực hiện nhiệm vụ.

Machine learning cần một tập dữ liệu với đầy đủ các đặc trưng và lables. Dựa vào các thuật toán của machine mà mô hình có thể học và ghi nhớ để có thể dự đoán được kết quả mong muốn dựa vào sự so sánh đặc trưng của đối tượng.

### 2.2.2 Một số thuật toán Machine learning.

* **Support Vector Machines:** Một thuật toán cố gắng xây dựng một siêu mặt phẳng trong không gian nhiều chiều để phân biệt các đối tượng ở các lớp khác nhau; Làm sao cho khoảng cách giữa 2 đối tượng khác label gần nhau nhất có khoảng cách cực đại. Ý tưởng của thuật toán cực kỳ đơn giản, nhưng mô hình này lại rất phức tạp và có hiệu quả. Thực tế, ở một số bài toán, SVM là một mô hình machine learning cho hiệu quả tốt nhất.

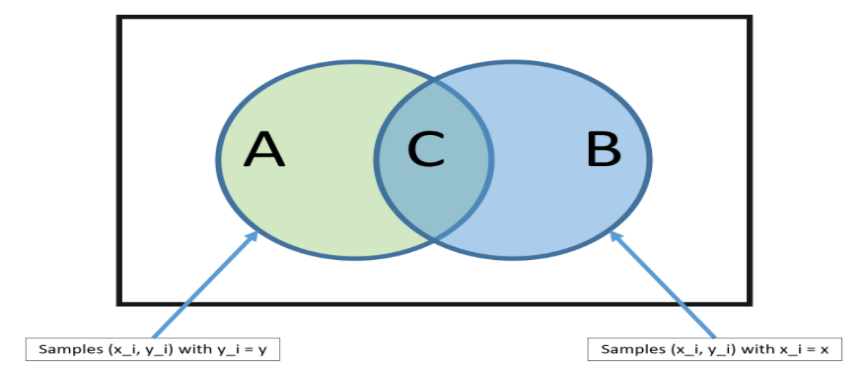


Hình 2.2 Các điểm gần mặt phân cách nhất của hai classes được khoanh tròn.

Giới thiệu thư viện: Chúng ta sẽ sử dụng hàm [sklearn.svm.SVC](http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html) của thư viên sklearn. Các bài toán thực tế thường sử dụng thư viện [libsvm](https://www.csie.ntu.edu.tw/~cjlin/libsvm/) được viết trên ngôn ngữ C, có API cho Python và Matlab.

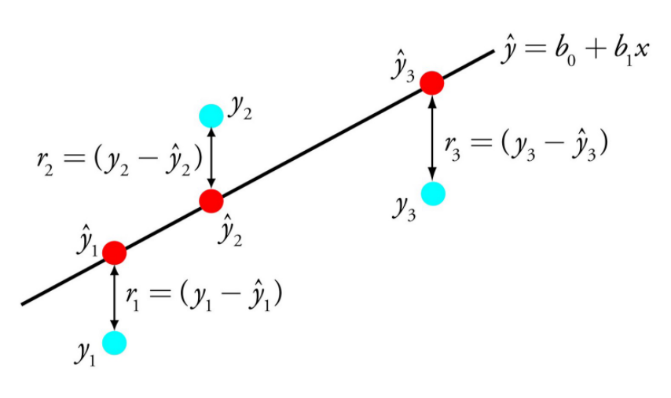
* **Mô hình xác suất(Probabilistic Models)**:

Các mô hình này cố gắng giải quyết bài toán bằng phân bố xác suất. Một thuật toán phổ biến nhất là phân loại Naive Bayes; Nó sử dụng lý thuyết Bayes và giả thiết các đặc trưng là độc lập. Điểm mạnh của mô hình xác suất là đơn giản nhưng hiệu quả. Đầu ra của nó không chỉ là label mà còn đi kèm xác suất thể hiện độ chính xác cho kết quả đó.



Hình 2.3 Tập giá trị của dữ liệu

* **Học sâu(Deep learning)**: Hiện đang là xu hướng trong machine learning dựa trên các mô hình mạng nơ ron nhân tạo(Artificial Neural Networks). Mạng nơ ron có cách tiếp cận kết nối và sử dụng ý tưởng theo cách bộ não con người làm việc. Chúng bao gồm số lượng lớn các nơ ron liên kết với nhau; được tổ chức thành các lớp(layers). Học sâu liên tục được phát triển với các cấu trúc mới sâu hơn; Nó không chỉ cố gắng học mà còn xây dựng các cấu trúc biểu diễn các đặc trưng quan trọng một cách tự động.
* **Linear Regression – Hồi quy tuyến tính**

 Hồi quy tuyến tính có lẽ là một trong những thuật toán nổi tiếng nhất và được hiểu rõ nhất trong thống kê và machine learning.

Mô hình tiên đoán chủ yếu quan tâm đến việc giảm thiểu sai sót của mô hình hoặc đưa ra các dự đoán chính xác nhất có thể, với một chi phí giải trình.

Biểu diễn hồi quy tuyến tính là một phương trình mô tả một đường thẳng mô tả phù hợp nhất mối quan hệ giữa các biến đầu vào (x) và các biến đầu ra (y), bằng cách tìm các trọng số cụ thể cho các biến đầu vào được gọi là các hệ số (B).

Ví dụ: y = B0 + B1 \* x

Chúng ta sẽ dự đoán y với biến x cho trước và mục tiêu của thuật toán hồi quy tuyến tính là tìm các giá trị cho các hệ số B0 và B

## 2.3 Deep learning

Có thể nói tính đến nay, AI đã gặt hái được khá nhiều bước tiến lớn. Hãy suy nghĩ về nó như là một loại machine learning với "[mạng thần kinh - neural networks](https://quantrimang.com/huong-dan-cho-nguoi-moi-hoc-ai-mang-tri-tue-nhan-tao-154435)" sâu có thể xử lý dữ liệu theo cách tương tự như một bộ não con người có thể thực hiện. Điểm khác biệt chính ở đây là ta chỉ cần đưa cho deep learning một tập dữ liệu lớn, nó sẽ tự học, tự trích xuất đặc trưng, và đưa ra kết quả mong muốn.

Tóm lại: Deep learning là loại machine learning mà trong đó máy tự đào tạo chính nó. Deep learning đòi hỏi rất nhiều dữ liệu đầu vào và sức mạnh tính toán hơn là machine learning, nhưng nó đã bắt đầu được triển khai bởi các công ty công nghệ lớn như Facebook, Amazon. Trong đó, một trong những cái tên nổi tiếng nhất về machine learning là [AlphaGo](https://quantrimang.com/tim-hieu-tri-tue-nhan-tao-alphago-khien-con-nguoi-than-phuc-121210), một máy tính có thể chơi cờ vây với chính bản thân nó cho đến khi nó có thể dự đoán những đường đi nước bước chính xác nhất đủ để đánh bại nhiều nhà vô địch trên thế giới.

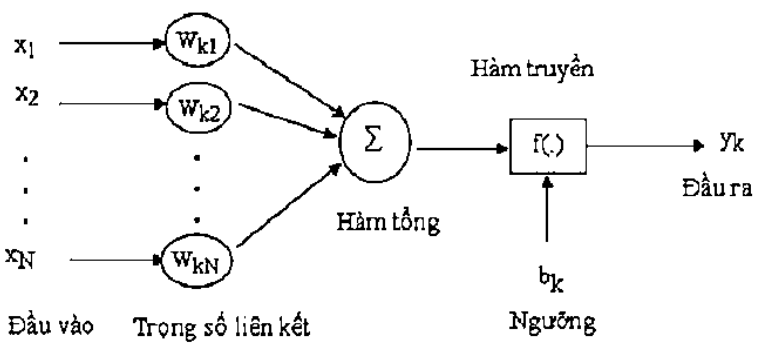
Deep learning đã cho phép ứng dụng nhiều vấn đề thực tế của máy đồng thời mở rộng lĩnh vực tổng thể của trí tuệ nhân tạo. Deep learning phá vỡ các cách thức con người làm việc bằng cách làm cho tất cả các loại máy móc trợ giúp có thể thực hiện được, gần hoặc giống hệt con người. Ô tô không người lái, chăm sóc sức khỏe con người tốt hơn… Tất cả đều được hiện thực hóa trong thời đại ngày nay. AI là hiện tại và tương lai của thế giới. Với sự trợ giúp của deep Learning, AI có thể hiện thực hóa ước mơ khoa học giả tưởng mà chúng ta đã tưởng tượng từ rất lâu

## 2.4 Mạng Nơ ron

### 2.4.1 Giới thiệu về mạng Nơ ron

***Định nghĩa:*** Mạng nơron nhân tạo, Artificial Neural Network (ANN) là một mô hình xử lý thông tin phỏng theo cách thức xử lý thông tin của các hệ nơron sinh học. Nó được tạo nên từ một số lượng lớn các phần tử (nơron) kết nối với nhau thông qua các liên kết (trọng số liên kết) làm việc như một thể thống nhất để giải quyết một vấn đề cụ thể nào đó. Một mạng nơron nhân tạo được cấu hình cho một ứng dụng cụ thể (nhận dạng mẫu, phân loại dữ liệu, ...) thông qua một quá trình học từ tập các mẫu huấn luyện. Về bản chất học chính là quá trình hiệu chỉnh trọng số liên kết giữa các nơron.

**Cấu trúc neural nhân tạo**



Hình 2.4 Cấu tạo 1 nơ ron

Các thành phần cơ bản của một nơron nhân tạo bao gồm:

* Tập các đầu vào: Là các tín hiệu vào (input signals) của nơron, các tín hiệu này thường được đưa vào dưới dạng một vector N chiều.
* Tập các liên kết: Mỗi liên kết được thể hiện bởi một trọng số liên kết – Synaptic weight. Trọng số liên kết giữa tín hiệu vào thứ j với nơron k thường được kí hiệu là wkj. Thông thường, các trọng số này được khởi tạo một cách ngẫu nhiên ở thời điểm khởi tạo mạng và được cập nhật liên tục trong quá trình học mạng.
* Bộ tổng (Summing function): Thường dùng để tính tổng của tích các đầu vào với trọng số liên kết của nó.
* Ngưỡng (còn gọi là một độ lệch - bias): Ngưỡng này thường được đưa vào như một thành phần của hàm truyền.
* Hàm truyền (Transfer function): Hàm này được dùng để giới hạn phạm vi đầu ra của mỗi nơron. Nó nhận đầu vào là kết quả của hàm tổng và ngưỡng.
* Đầu ra: Là tín hiệu đầu ra của một nơron, với mỗi nơron sẽ có tối đa là một đầu ra.

Xét về mặt toán học, cấu trúc của một nơron k, được mô tả bằng cặp biểu thức sau:

và

Trong đó: x1, x2, ..., xp: là các tín hiệu vào; (wk1, wk2,..., wkp) là các trọng số liên kết của nơron thứ k; uk là hàm tổng; bk là một ngưỡng; f là hàm truyền và yk là tín hiệu đầu ra của nơron.

Như vậy nơron nhân tạo nhận các tín hiệu đầu vào, xử lý (nhân các tín hiệu này với trọng số liên kết, tính tổng các tích thu được rồi gửi kết quả tới hàm truyền), và cho một tín hiệu đầu ra (là kết quả của hàm truyền).

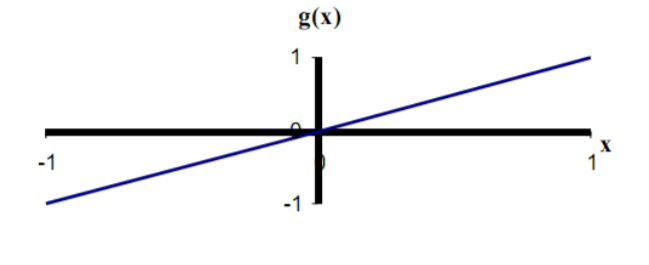
### 2.4.2 Một số hàm truyền thông dụng

Phần lớn các đơn vị trong mạng nơron chuyển net input bằng cách sử dụng một hàm vô hướng (scalar-to-scalar function) gọi là hàm kích hoạt, kết quả của hàm này là một giá trị gọi là mức độ kích hoạt của đơn vị (unit's activation). Loại trừ khả năng đơn vị đó thuộc lớp ra, giá trị kích hoạt được đưa vào một hay nhiều đơn vị khác. Các hàm kích hoạt thường bị ép vào một khoảng giá trị xác định, do đó thường được gọi là các hàm bẹp (squashing). Các hàm kích hoạt hay được sử dụng là:

* Hàm đồng nhất (Linear function, Identity function )

g(x) = x (2)

Nếu coi các đầu vào là một đơn vị thì chúng sẽ sử dụng hàm này. Đôi khi một hằng số được nhân với net-input để tạo ra một hàm đồng nhất.



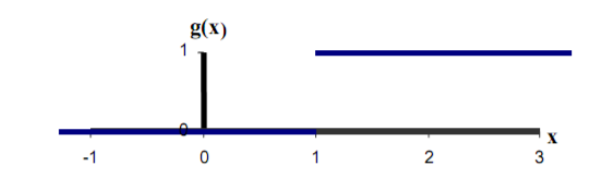
Hình 2.5 – Hàm đồng nhất (Identity function)

* Hàm bước nhị phân (Binary step function, Hard limit function)

Hàm này cũng được biết đến với tên "Hàm ngưỡng" (Threshold function hay Heaviside function). Đầu ra của hàm này được giới hạn vào một trong hai giá trị:

(3)

Dạng hàm này được sử dụng trong các mạng chỉ có một lớp. Trong hình vẽ sau, θ được chọn bằng 1.

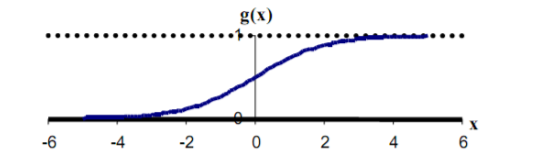


Hình 2.6 – Hàm bước nhị phân (Binary step function)

* Hàm sigmoid (Sigmoid function (logsig))

g(x) = (4)

Hàm này đặc biệt thuận lợi khi sử dụng cho các mạng được huấn luyện (trained) bởi thuật toán Lan truyền ngược (back-propagation), bởi vì nó dễ lấy đạo hàm, do đó có thể giảm đáng kể tính toán trong quá trình huấn luyện. Hàm này được ứng dụng cho các chương trình ứng dụng mà các đầu ra mong muốn rơi vào khoảng [0,1].

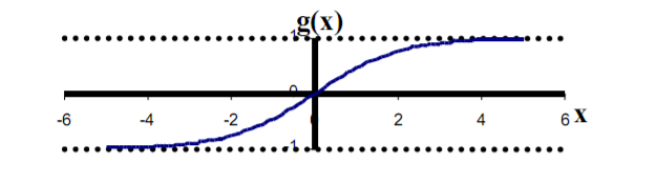


Hình 2.7 – Hàm Sigmoid

* Hàm sigmoid lưỡng cực (Bipolar sigmoid function (tansig))

g(x) = (5)

Hàm này có các thuộc tính tương tự hàm sigmoid. Nó làm việc tốt đối với các ứng dụng có đầu ra yêu cầu trong khoảng [-1,1].



Hình 2.8 – Hàm sigmoid lưỡng cực

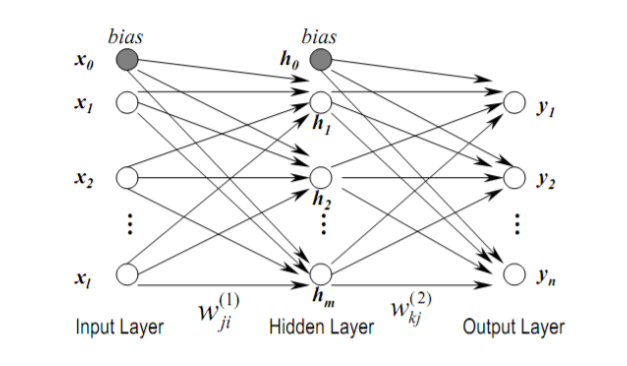
Các hàm chuyển của các đơn vị ẩn (hidden units) là cần thiết để biểu diễn sự phi tuyến vào trong mạng. Lý do là hợp thành của các hàm đồng nhất là một hàm đồng nhất. Mặc dù vậy nhưng nó mang tính chất phi tuyến (nghĩa là, khả năng biểu diễn các hàm phi tuyến) làm cho các mạng nhiều tầng có khả năng rất tốt trong biểu diễn các ánh xạ phi tuyến. Tuy nhiên, đối với luật học lan truyền ngược, hàm phải khả vi (differentiable) và sẽ có ích nếu như hàm được gắn trong một khoảng nào đó. Do vậy, hàm sigmoid là lựa chọn thông dụng nhất.

Đối với các đơn vị đầu ra (output units), các hàm chuyển cần được chọn sao cho phù hợp với sự phân phối của các giá trị đích mong muốn. Chúng ta đã thấy rằng đối với các giá trị ra trong khoảng [0,1], hàm sigmoid là có ích; đối với các giá trị đích mong muốn là liên tục trong khoảng đó thì hàm này cũng vẫn có ích, nó có thể cho ta các giá trị ra hay giá trị đích được căn trong một khoảng của hàm kích hoạt đầu ra. Nhưng nếu các giá trị đích không được biết trước khoảng xác định thì hàm hay được sử dụng nhất là hàm đồng nhất (identity function). Nếu giá trị mong muốn là dương nhưng không biết cận trên thì nên sử dụng một hàm kích hoạt dạng mũ (exponential output activation function).

### 2.4.3 Một số kiểu mạng Nơ ron

**Mạng truyền thẳng (Feed-forward neural network)**

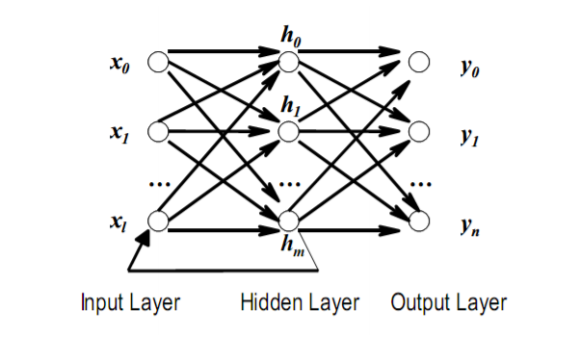
Dòng dữ liệu từ đơn vị đầu vào đến đơn vị đầu ra chỉ được truyền thẳng. Việc xử lý dữ liệu có thể mở rộng ra nhiều lớp, nhưng không có các liên kết phản hồi. Nghĩa là, các liên kết mở rộng từ các đơn vị đầu ra tới các đơn vị đầu vào trong cùng một lớp hay các lớp trước đó là không cho phép.



Hình 2.9 – Mạng nơron truyền thẳng nhiều lớp (Feed-forward neural network)

**Mạng hồi quy (Recurrent neural network)**

Có chứa các liên kết ngược. Khác với mạng truyền thẳng, các thuộc tính động của mạng mới quan trọng. Trong một số trường hợp, các giá trị kích hoạt của các đơn vị trải qua quá trình nới lỏng (tăng giảm số đơn vị và thay đổi các liên kết) cho đến khi mạng đạt đến một trạng thái ổn định và các giá trị kích hoạt không thay đổi nữa. Trong các ứng dụng khác mà cách chạy động tạo thành đầu ra của mạng thì những sự thay đổi các giá trị kích hoạt là đáng quan tâm.



Hình 2.10 – Mạng nơron hồi quy (Recurrent neural network)

* + 1. **Các phương pháp huấn luyện mạng Nơ ron**

#### 2.4.4.1 Khái niệm

Học là quá trình thay đổi hành vi của các vật theo một cách nào đó làm cho chúng có thể thực hiện tốt hơn trong tương lai.

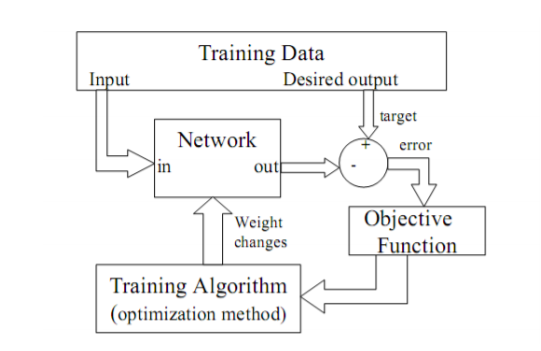
Một mạng nơron được huyấn luyện sao cho với một tập các vector đầu vào X, mạng có khả năng tạo ra tập các vector đầu ra mong muốn Y của nó. Tập X được sử dụng cho huấn luyện mạng được gọi là tập huấn luyện (*training set*). Các phần tử x thuộc X được gọi là các mẫu huấn luyện (*training example*). Quá trình huấn luyện bản chất là sự thay đổi các trọng số liên kết của mạng.

Trong quá trình này, các trọng số của mạng sẽ hội tụ dần tới các giá trị sao cho với mỗi vector đầu vào x từ tập huấn luyện, mạng sẽ cho ra vector đầu ra y như mong muốn.

Có ba phương pháp học phổ biến là học có giám sát (*supervised learning*), học không giám sát (*unsupervised learning*) và học củng cố (*Reinforcement learning*)

#### 2.4.4.2 Học có giám sát

Mạng được huấn luyện bằng cách cung cấp cho nó các cặp mẫu đầu vào và các đầu ra mong muốn (target values). Các cặp được cung cấp bởi "thầy giáo", hay bởi hệ thống trên đó mạng hoạt động. Sự khác biệt giữa các đầu ra thực tế so với các đầu ra mong muốn được thuật toán sử dụng để thích ứng các trọng số trong mạng. Điều này thường được đưa ra như một bài toán xấp xỉ hàm số - cho dữ liệu huấn luyện bao gồm các cặp mẫu đầu vào x, và một đích tương ứng t, mục đích là tìm ra hàm f(x) thoả mãn tất cả các mẫu học đầu vào.



Hình 2.11 – Mô hình Học có thầy (Supervised learning model)

#### 2.4.4.3 Học không giám sát

Là việc học không cần có bất kỳ một sự giám sát nào.

Trong bài toán học không giám sát, tập dữ liệu huấn luyện được cho dưới dạng: D = {(x1 , x2, ..., xN )}, với (x1 , x2 , ..., xN) là vector đặc trưng của mẫu huấn luyện. Nhiệm vụ của thuật toán là phải phân chia tập dữ liệu D thành các nhóm con, mỗi nhóm chứa các vector đầu vào có đặc trưng giống nhau. Như vậy với học không giám sát, số lớp phân loại chưa được biết trước, và tùy theo tiêu chuẩn đánh giá *độ tương tự* giữa các mẫu mà ta có thể có các lớp phân loại khác nhau.

#### 2.4.4.4 Học củng cố

Đôi khi còn được gọi là học thưởng-phạt (rewardpenalty learning), là sự tổ hợp của cả hai mô hình trên. Phương pháp này cụ thể như sau: với vector đầu vào, quan sát vector đầu ra do mạng tính được. Nếu kết quả được xem là “tốt” thì mạng sẽ được thưởng theo nghĩa tăng các trọng số kết nối lên; ngược lại mạng sẽ bị phạt, các trọng số kết nối không thích hợp sẽ được giảm xuống. Do đó học tăng cường là học theo nhà phê bình (critic), ngược với học có giám sát là học theo thầy giáo (teacher).

#### 2.4.4.5 Học có giám sát trong mạng Nơ ron

Học có giám sát có thể được xem như việc xấp xỉ một ánh xạ: X→ Y, trong đó X là tập các vấn đề và Y là tập các lời giải tương ứng cho vấn đề đó. Các mẫu (x, y) với x = (x1, x2, . . ., xn) ∈ X, y = (yl, y2, . . ., ym) ∈ Y được cho trước. Học có giám sát trong các mạng nơron thường được thực hiện theo các bước sau:

**B1**: Xây dựng cấu trúc thích hợp cho mạng nơron, chẳng hạn có (n + 1) nơron vào (n nơron cho biến vào và 1 nơron cho ngưỡng x0), m nơron đầu ra, và khởi tạo các trọng số liên kết của mạng.

**B2**: Đưa một vector x trong tập mẫu huấn luyện X vào mạng

**B3**: Tính vector đầu ra o của mạng

**B4.** So sánh vector đầu ra mong muốn y (là kết quả được cho trong tập  
huấn luyện) với vector đầu ra o do mạng tạo ra; nếu có thể thì đánh giá lỗi.

**B5**: Hiệu chỉnh các trọng số liên kết theo một cách nào đó sao cho ở lần tiếp theo khi đưa vector x vào mạng, vector đầu ra o sẽ giống với y hơn.

**B6**: Nếu cần, lặp lại các bước từ 2 đến 5 cho tới khi mạng đạt tới trạng  
thái hội tụ. Việc đánh giá lỗi có thể thực hiện theo nhiều cách, cách dùng nhiều nhất là sử dụng lỗi tức thời: Err = (**o** - **y**), hoặc Err = |**o** - **y**|; lỗi trung bình bình phương (MSE: mean-square error): Err = (**o**- **y**)2/2;Có hai loại lỗi trong đánh giá một mạng nơron. Thứ nhất, gọi là lỗi rõ ràng (*apparent error*), đánh giá khả năng xấp xỉ các mẫu huấn luyện của một mạng đã được huấn luyện. Thứ hai, gọi là lỗi kiểm tra (*test error*), đánh giá khả năng tổng quá hóa của một mạng đã được huấn luyện, tức khả năng phản ứng với các vector đầu vào mới. Để đánh giá lỗi kiểm tra chúng ta phải biết đầu ra mong muốn chocác mẫu kiểm tra.

Thuật toán tổng quát ở trên cho học có giám sát trong các mạng nơron có nhiều cài đặt khác nhau, sự khác nhau chủ yếu là cách các trọng số liên kết được thay đổi trong suốt thời gian học. Trong đó tiêu biểu nhất là thuật toán lan truyền ngược.

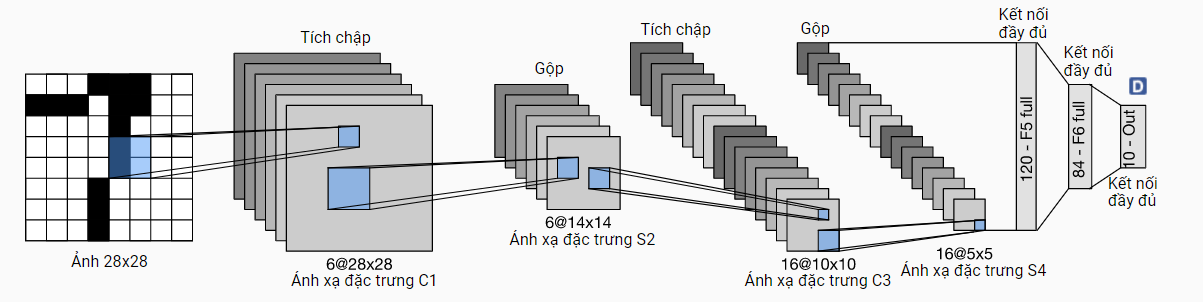
* 1. **Mạng Nơ ron tích chập**

### 2.5.1 Định nghĩa

Những năm gần đây, ta đã chứng kiến được nhiều thành tựu vượt bậc trong ngành Thị giác máy tính (Computer Vision). Các hệ thống xử lý ảnh lớn như Facebook, Google hay Amazon đã đưa vào sản phẩm của mình những chức năng thông minh như nhận diện khuôn mặt người dùng, phát triển xe hơi tự lái hay drone giao hàng tự động.

Convolutional Neural Network (CNNs – Mạng nơ-ron tích chập) là một trong những mô hình Deep Learning tiên tiến giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay. Yann LeCun, Leon Bottou, Yosuha Bengio và Patrick Haffner đã đề xuất kiến ​​trúc mạng thần kinh để nhận dạng ký tự viết tay và in bằng máy vào năm 1990 mà họ gọi là LeNet-5. Kiến trúc đơn giản và dễ hiểu, đó là lý do tại sao nó chủ yếu được sử dụng như một bước đầu tiên để giảng dạy.

Một cách đơn giản, ta có thể xem LeNet gồm hai phần: (i) một khối các tầng tích chập; và (ii) một khối các tầng kết nối đầy đủ. Trước khi đi vào các chi tiết cụ thể, hãy quan sát tổng thể mô hình bên dưới

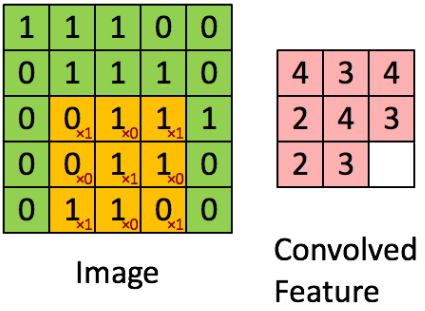


Hình 2.12: Dòng dữ liệu trong LeNet 5. Đầu vào là một chữ số viết tay, đầu ra là một xác suất đối với 10 kết quả khả thi.

### 2.5.2 Tích chập

Tích chập được sử dụng đầu tiên trong xử lý tín hiệu số (Signal  
processing). Nhờ vào nguyên lý biến đổi thông tin, các nhà khoa học đã áp dụng kĩ thuật này vào xử lý ảnh và video số.

Để dễ hình dung, ta có thể xem tích chập như một cửa sổ trượt (sliding window) áp đặt lên một ma trận. Bạn có thể theo dõi cơ chế của tích chập qua hình minh họa bên dưới.



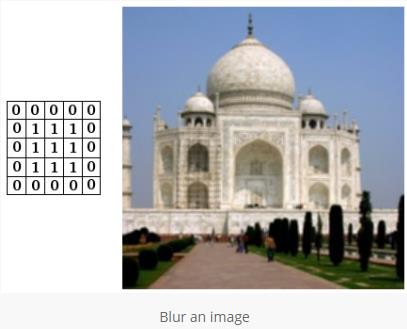
Hình 2.13 Minh họa tích chập

Ma trận bên trái là một bức ảnh đen trắng. Mỗi giá trị của ma trận tương đương với một điểm ảnh (pixel), 0 là màu đen, 1 là màu trắng (nếu là ảnh grayscale thì giá trị biến thiên từ 0 đến 255).

Sliding window còn có tên gọi là kernel, filter hay feature detector. Ở đây, ta dùng một ma trận filter 3×3 nhân từng thành phần tương ứng (element-wise) với ma trận ảnh bên trái. Gía trị đầu ra do tích của các thành phần này cộng lại.

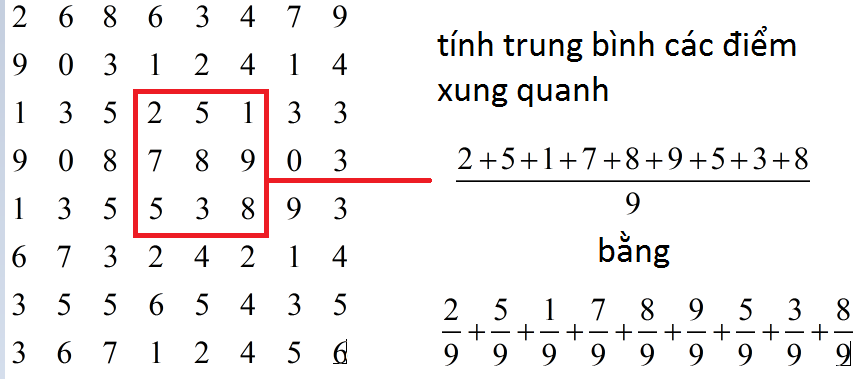
Kết quả của tích chập là một ma trận (convoled feature) sinh ra từ việc trượt ma trận filter và thực hiện tích chập cùng lúc lên toàn bộ ma trận ảnh bên trái. Dưới đây là một vài ví dụ của phép toán tích chập.

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.

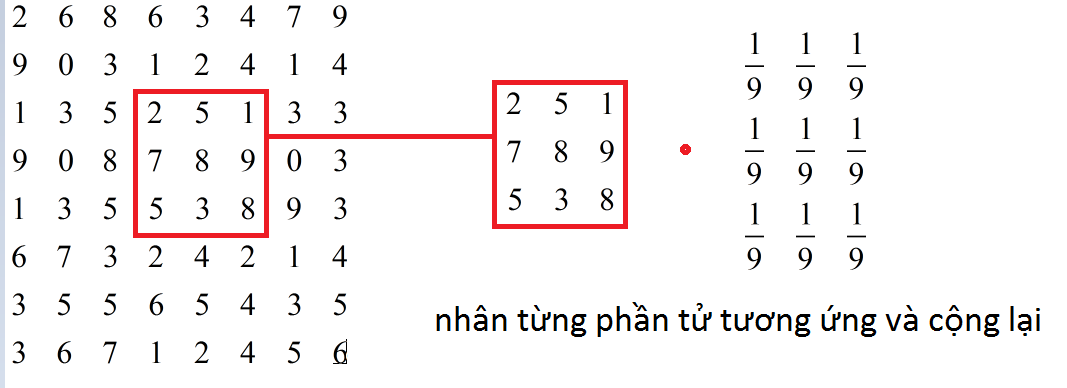


Hình 2.14 Ảnh mờ sau khi chập

Về bản chất thực hiện làm mờ ảnh chính là tạo ra ảnh mới sao cho giá trị mức xám của mỗi pixel ở ảnh mới đúng bằng giá trị trung bình của điểm tương ứng và 8 điểm lân cận trên ảnh ban đầu. Nói cách khác, với mỗi điểm trên hình ban đầu, bạn tính giá trị trung bình của nó (tại hàng i cột j) với 8 điểm xung quanh rồi viết lại giá trị mức xám ở vị trí tương ứng (cũng tại hàng i cột j) lên ảnh mới, sau đó tương tự với các điểm tiếp theo.

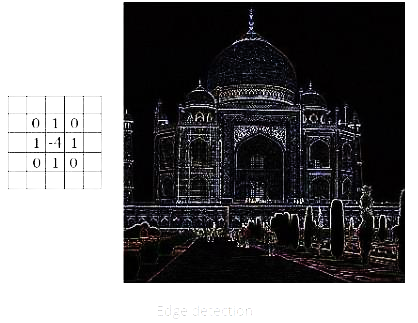


Vậy có nghĩa là ta đang tính trung bình cộng của 9 pixel (pixel tại điểm đó và 8 pixel lân cận), vậy phép tính đó cũng giống như nhân từng giá trị mức sáng của các pixel lân cận với 1/9  sau đó cộng lại với nhau. Vậy, nếu có một ma trận 3 x 3 với tất cả các con số trong ma trận đều là 1/9 , ta nhân từng phần tử của ma trận này với mức sáng của pixel tương ứng và cộng lại, ta sẽ có kết quả giống nhau (xem hình vẽ bên dưới).



Áp dụng tương tự cho mọi pixel trên ảnh ban đầu và lấy từng kết quả cho từng pixel của ảnh mới, ta sẽ được ảnh mới chính là ảnh mờ của ảnh ban đầu.

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



Hình 2.15 Ảnh được phát hiện biên sau khi chập

### 2.5.3 Lan truyền tiến

Tất cả các nốt mạng (nơ-ron) được kết hợp đôi một với nhau theo một chiều duy nhất từ tầng vào tới tầng ra. Tức là mỗi nốt ở một tầng nào đó sẽ nhận đầu vào là tất cả các nốt ở tầng trước đó mà không suy luận ngược lại. Hay nói cách khác, việc suy luận trong mạng NN là **suy luận tiến** (feedforward)

(6)

Trong đó,  số lượng nút ở tầng l tương ứng và ​ là nút mạng thứ *j* của tầng *l*. Còn  là tham số trọng lượng của đầu vào ​ đối với nút mạng thứ *i* của tầng *l*+1 và  là độ lệch (bias) của nút mạng thứ *i* của tầng  *l*+1. Đầu ra của nút mạng này được biểu diễn bằng ​ ứng với hàm kích hoạt *f*(*zi*​) tương ứng.

Để tiện tính toán, ta coi ​ là một đầu vào và ​= là tham số trọng lượng của đầu vào này. Lúc đó ta có thể viết lại công thức trên dưới dạng véc-tơ:

(7)

* + 1. **Thuật toán lan truyền ngược (Back-Propagation)**

Cần có một sự phân biệt giữa kiến trúc của một mạng và thuật toán học của nó, các mô tả trong các mục trên mục đích là nhằm làm rõ các yếu tố về kiến trúc của mạng và cách mà mạng tính toán các đầu ra từ tập các đầu vào. Sau đây là mô tả của thuật toán học sử dụng để điều chỉnh hiệu năng của mạng sao cho mạng có khả năng sinh ra được các kết quả mong muốn.

Về cơ bản có hai dạng thuật toán để luyện mạng: học có giám sát và học không giám sát. Các mạng nơron truyền thẳng nhiều lớp được luyện bằng phương pháp học có thầy. Phương pháp này căn bản dựa trên việc yêu cầu mạng thực hiện chức năng của nó và sau đó trả lại kết quả, kết hợp kết quả này với các đầu ra mong muốn để điều chỉnh các tham số của mạng, nghĩa là mạng sẽ học thông qua những sai sót của nó.

Thuật toán lan truyền ngược là dạng tổng quát của thuật toán trung bình bình phương tối thiểu (Least Means Square-LMS). Thuật toán này thuộc dạng thuật toán xấp xỉ để tìm các điểm mà tại đó hiệu năng của mạng là tối ưu. Chỉ số tối ưu (performance index) thường được xác định bởi một hàm số của ma trận trọng số và các đầu vào nào đó mà trong quá trình tìm hiểu bài toán đặt ra.

Bỏ qua sự phức tạp về mặt toán học, thuận toán có thể phát biểu đơn giản như sau:

Bước 1: Lan truyền xuôi các tính toán trong mạng truyền thẳng

* Khi đó, đầu ra của một lớp trở thành đầu vào của lớp kế tiếp
* Các nơron trong lớp thứ nhất nhận các tín hiệu từ bên ngoài
* Đầu ra của lớp cuối cùng được xem là đầu ra của mạng

Bước 2: Lan truyền lỗi (hay độ nhạy cảm) ngược lại qua mạng

* Thuật toán lan truyền ngược sử dụng chỉ số hiệu năng là trung bình bình phương lỗi của đầu ra so với giá trị đích. Đầu vào của thuật toán chính là tập các cặp mô tả hoạt động đúng của mạng

{(p1,t1), (p2, t2),…,(pQ,tQ)}

Trong đó pi là một đầu vào và ti là đầu ra mong muốn tương ứng, với i=1..Q.

* Mỗi đầu vào đưa vào mạng, đầu ra của mạng đối với nó được đem so sánh với đầu ra mong muốn.Thuật toán sẽ điều chỉnh các tham số của mạng để tối thiểu hóa trung bình bình phương lỗi.

Bước 3: Cập nhật lại các trọng số và độ lệch tương ứng

Lan truyền ngược (backpropagation) là giải thuật cốt lõi giúp cho các mô hình học sâu có thể dễ dàng thực thi tính toán được. Với các mạng NN hiện đại, nhờ giải thuật này mà thuật toán tối ưu với đạo hàm (gradient descent) có thể nhanh hơn hàng triệu lần so với cách thực hiện truyền thống. Cứ tưởng tượng 1 mô hình với lan truyền ngược chạy mất 1 tuần thì có thể mất tới 200,000 năm để huấn luyện với phương pháp truyền thống!

Mặc lan truyền ngược được sử dụng cho học sâu, nhưng nó còn là công cụ tính toán mạnh mẽ cho nhiều lĩnh vực khác từ dự báo thời tiết tới phân tích tính ổn định số học, chỉ có điều là nó được sử dụng với những tên khác nhau. Thực ra nó được khai phá lại để sử dụng cho rất nhiều lĩnh vực khác nhau. Nhưng một cách tổng quát không phụ thuộc vào ứng dụng thì tên của nó là “phép vi phân ngược” (reverse-mode differentiation).

Về cơ bản, nó là một kĩ thuật để nhanh chóng tính được đạo hàm. Và nó là một mẹo cần thiết mà bạn cần hành trang cho mình không chỉ trong lĩnh vực học sâu mà còn cho nhiều bài toán tính toán số học khác nữa.

* + 1. **Activation functions**

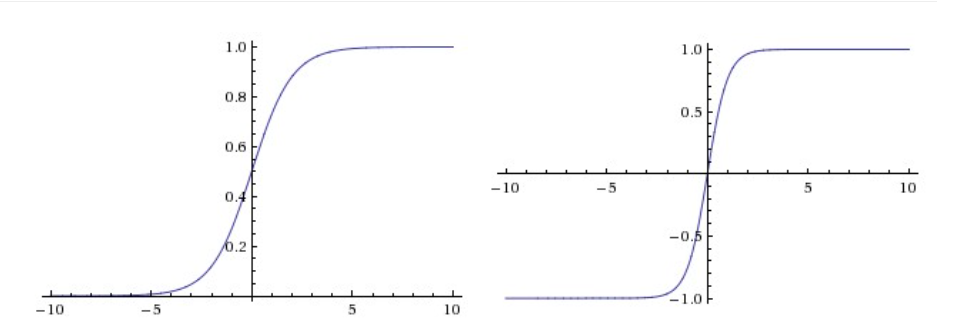
Mỗi output của một unit (trừ các input units) được tính dựa vào công thức:  
Ai(l) = f( wi(l)Ta(l-1) + bi(l))

Trong đó f(.)f(.) là một (nonlinear) activation function. Ở dạng vector, biểu thức bên trên được viết là:

A(l)  = f(w(l)Ta(l-1) + b(l)

Khi activation function f(.)f(.) được áp dụng cho một ma trận (hoặc vector), ta hiểu rằng nó được áp dụng cho từng thành phần của ma trận đó. Sau đó các thành phần này được sắp xếp lại đúng theo thứ tự để được một ma trận có kích thước bằng với ma trận input. Trong tiếng Anh, việc áp dụng lên từng phần tử như thế này được gọi là element-wise.

**Sigmoid và Tanh**



Hình 2.16: Hàm sigmoid (trái) và tanh (phải).

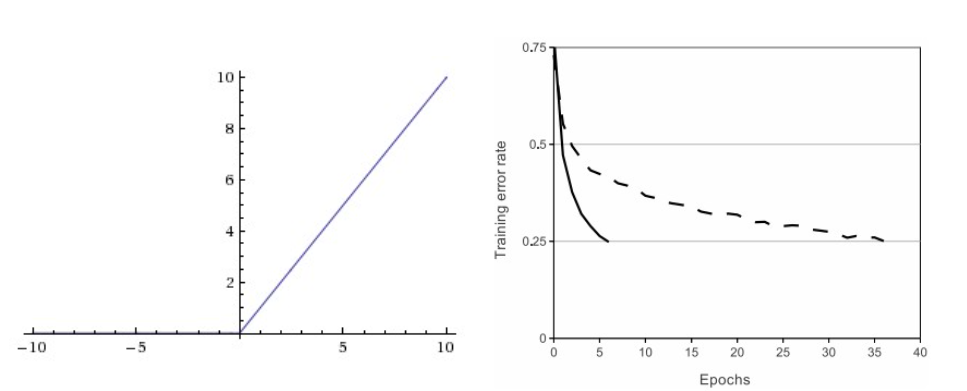
(Nguồn [CS231n Convolutional Neural Networks for Visual Recognition](https://cs231n.github.io/neural-networks-1/))

Hàm *sigmoid* có dạng f(s)=1/(1+) với đồ thị như trong Hình 2.16 (trái). Nếu đầu vào lớn, hàm số sẽ cho đầu ra gần với 1. Với đầu vào nhỏ (rất âm), hàm số sẽ cho đầu ra gần với 0. Hàm số này được sử dụng nhiều trong quá khứ ví có đạo hàm rất *đẹp*. Những năm gần đây, hàm số này ít khi được sử dụng. Nó có một nhược điểm cơ bản:

* *Sigmoid saturate and kill gradients (sigmoid bảo hòa nếu nó đạt giá trị lớn nhất hoặc nhỏ nhất)*: Một nhược điểm dễ nhận thấy là khi đầu vào có trị tuyệt đối lớn (rất âm hoặc rất dương), gradient của hàm số này sẽ rất gần với 0. Điều này đồng nghĩa với việc các hệ số tương ứng với unit đang xét sẽ gần như không được cập nhật. Bạn đọc sẽ hiểu rõ hơn phần này trong phần “Lan truyền ngược” ở trên.

Hàm *tanh* cũng có nhược điểm tương tự về việc gradient rất nhỏ với các đầu vào có trị tuyệt đối lớn.

**Relu**



Hình 2.17: Hàm ReLU và tốc độ hội tụ khi so sánh với hàm tanh.

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 Hình 5 (trái)). Nó có công thức toán học f(s)=max(0,s) - 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)). Hình 5 (phải) so sánh sự hội tụ của SGD khi sử dụng hai activation function khác nhau: ReLU và tanh. 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 s=0, trong thực nghiệm, người ta vẫn thường định nghĩa ReLU′(0)=0 và khẳng định thêm rằng, xác suất để input của một unit bằng 0 là rất nhỏ.

Hàm ReLU có nhiều biến thể khác như [Noisy ReLU, Leaky ReLu, ELUs](https://en.wikipedia.org/wiki/Rectifier_(neural_networks)). Tôi xin phép dừng phần này ở đây vì chưa có ý định đi sâu vào Deep Neural Networks.

**Softmax function**

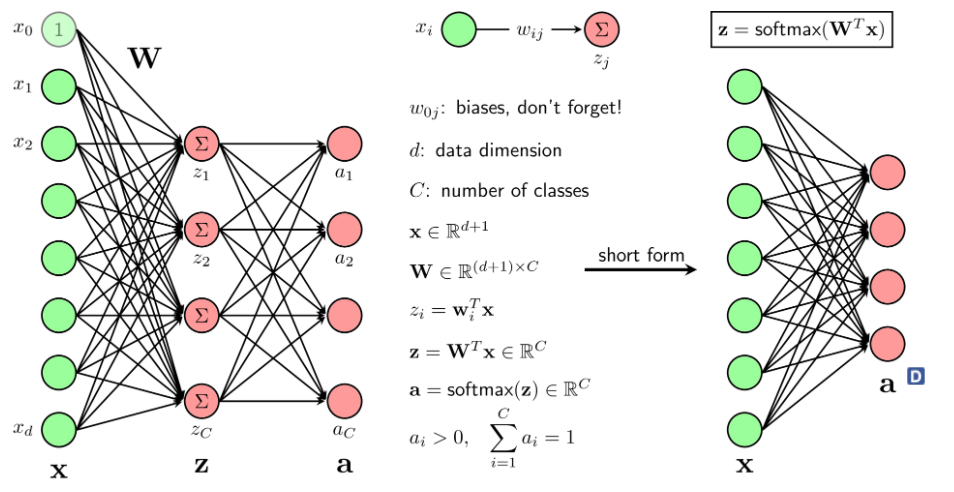
Xét lại bài toán phân loại, kết quả đầu ra cần dự đoán được dữ liệu đầu vào thuộc lớp nào, kết quả gồm 100% được chia đều cho tất cả các lớp, lớp nào có xác suất lớn nhất chính là kết quả đầu ra

Vậy yêu cầu là tìm một hàm nào đó trả về xác suất cho tín hiệu thuộc lớp nào và tổng các xác suất bằng 100%. Một vấn đề gặp phải nữa đó chính là tín hiệu từ các lớp của **Neural Network** có thể là các giá trị âm. Vậy ta cần một hàm số mượt luôn trả về giá trị dương để dễ dàng cho quá trình tính toán. Thêm vào đó, tín hiệu của neural càng mạnh thì xác suất thuộc lớp đó càng lớn, vì vậy ta cần một hàm đồng biến để có thể đánh giá chính xác giá trị của neural đó.

CTTQ: (8)

Hàm số này, tính tất cả các  dựa vào tất cả các , thõa mãn tất cả các điều kiện đã xét: dương, tổng bằng 1, giữ được thứ tự của . Hàm số này được gọi là softmax function.

Hình vẽ dưới đây thể hiện mạng Softmax Regression dưới dạng neural network:



Hình 2.18: Mô hình Softmax Regression dưới dạng Neural network.

* + 1. **Kiến trúc mô hình học sâu lenet5**

Phần này sẽ giới thiệu một trong những mạng sâu được sử dụng rộng rãi đó là mạng tích chập sâu (deep convolutional networks). Chúng ta sẽ làm việc cụ thể với mạng tích chập để giải quyết bài toán phân loại chữ số viết tay từ tập dữ liệu MNIST.

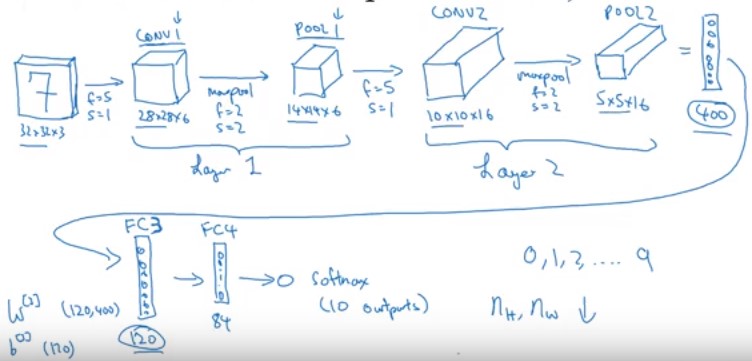
Chúng ta sẽ bắt đầu mạng tích chập với việc sử dụng mạng truyền thống để giải quyết bài toán này trong phần trước. Mặc dù nhiều phép toán lặp nhưng chúng ta sẽ xây dựng mạng hiệu quả hơn. Chúng ta sẽ khám phá ra rất nhiều kĩ thuật hiệu quả: Tích chập (convolution), giảm số chiều (pooling), sử dụng GPUs để huấn luyện được nhiều dữ liệu hơn chúng ta đã thực hiện trên mạng cũ, mở rộng giải thuật huấn luyện dữ liệu (để giảm quá khớp – overfitting), sử dụng kĩ thuật dropout để giảm overfitting, việc sử dụng tổng hợp các mạng và các kĩ thuật khác. Kết quả là hệ thống làm việc gần như con người. Trong số 10.000 bức ảnh huấn luyện, hệ thống của chúng ta sẽ phân loại đúng 9.967 bức ảnh.

Phần còn lại của chương sẽ thảo luận về học sâu dưới góc độ tổng quan và chi tiết. Chúng ta sẽ tìm hiểu làm thế nào để các mô hình mạng nơron tích chập có thể ứng dụng để giải quyết các bài toán nhận dạng tiếng nói, xử lí ngôn ngữ tự nhiên và các lĩnh vực khác. Và chúng ta sẽ nghiên cứu về mạng nơron trong tương lai và học sâu (deep learning), từ các ý tưởng như giao diện người sử dụng hướng đích đến vai trò của học sâu trong trí tuệ nhân tạo.

Phần này xây dựng dựa trên các phần trước sử dụng các ý tưởng như: lan truyền ngược (backpropagation), regularization, hàm softmax…



Chúng ta đã sử dụng mạng nơron mà trong đó các tầng liền kề liên kết đầy đủ với nhau. Tức là mỗi nơron trong mạng liên kết với tất cả các nơron trong tầng liền kề.



Hình 2.19 Mô hình mạng CNNs Lenet5

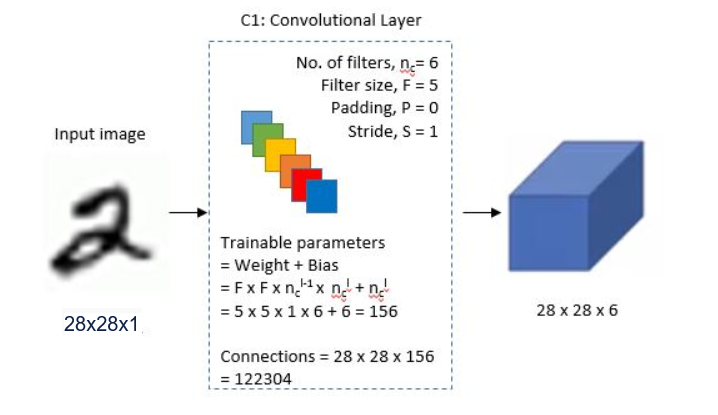
* 1. Khái niệm:
* Filter/ kernel: là ma trận lọc thường có kích thước là (5,5) hoặc (3,3)
* Stride(sải bước): là số bước mà ma trận lọc trượt trên ma trận ảnh

Hai khái niệm này sẽ dễ hiểu hơn qua các layers dưới đây

* 1. Chi tiết các lớp

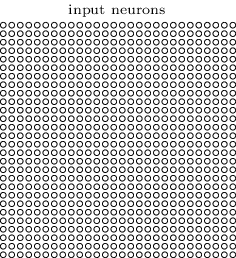
**Lớp Tích chập thứ nhất (Convolution)**

Đầu vào cho LeNet-5 là ảnh xám 28x28 đi qua lớp tích đầu tiên với 6 feature map hoặc bộ lọc có kích thước 5 × 5 và một bước sải. Kích thước hình ảnh thay đổi từ 28x28x1 thành 28x28x6



C1: Convolutional Layer

Ta có ma trận điểm ảnh cho ảnh 28x28 như sau:

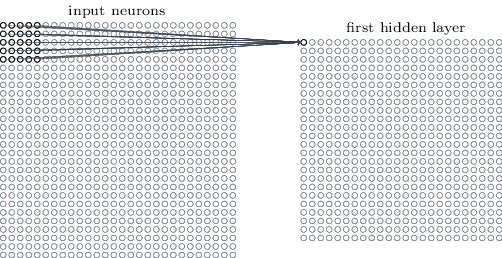


Hình 2.20 Input Nơ ron

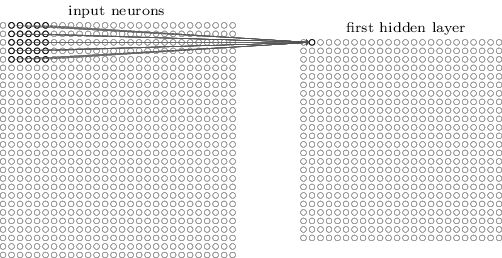
Như thường lệ chúng ta sẽ kết nối các điểm ảnh đầu vào cho các nơron ở tầng ẩn. Nhưng chúng ta sẽ không kết nối mỗi điểm ảnh đầu vào cho mỗi nơ ron ẩn. Thay vào đó, chúng ta chỉ kết nối trong phạm vi nhỏ, các vùng cục bộ của bức ảnh.

Để được chính xác hơn, mỗi nơron trong lớp ẩn đầu tiên sẽ được kết nối với một vùng nhỏ của các nơron đầu vào, ví dụ, một vùng 5 × 5, tương ứng với 25 điểm ảnh đầu vào.

Vùng đó trong bức ảnh đầu vào được gọi là vùng tiếp nhận cục bộ cho nơron ẩn. Đó là một kernel nhỏ trên các điểm ảnh đầu vào. Mỗi kết nối sẽ học một trọng số. Và nơron ẩn cũng sẽ học một độ lệch (overall bias). Ta có thể hiểu rằng nơron lớp ẩn cụ thể là học để phân tích trường tiếp nhận cục bộ cụ thể của nó. Sau đó chúng ta trượt trường tiếp nhận cục bộ trên toàn bộ bức ảnh. Đối với mỗi trường tiếp nhận cục bộ, có một nơron ẩn khác trong tầng ẩn đầu tiên . Để minh họa điều này một cách cụ thể, chúng ta hãy bắt đầu với một trường tiếp nhận cục bộ ở góc trên bên trái:



Sau đó, chúng ta trượt trường tiếp nhận cục bộ trên bởi một điểm ảnh bên phải (tức là bằng một nơron), để kết nối với một nơron ẩn thứ hai:

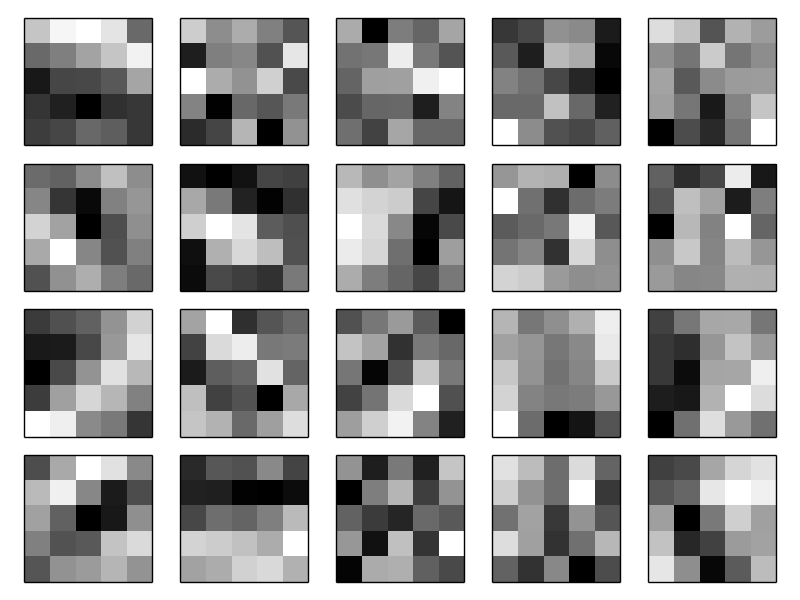


Hình 2.21 Trượt kernel trên ma trân ảnh

Và như vậy, việc xây dựng các lớp ẩn đầu tiên. Lưu ý rằng nếu chúng ta có một ảnh đầu vào 28 × 28 và 5 × 5 trường tiếp nhận cục bộ thì ta sẽ có 24 × 24 nơron trong lớp ẩn. Có được điều này là do chúng ta chỉ có thể di chuyển các trường tiếp nhận cục bộ ngang qua 23 nơron (hoặc xuống dưới 23 nơron), trước khi chạm với phía bên phải (hoặc dưới) của ảnh đầu vào. Trọng số và độ lệch (Shared weights and biases) : Mỗi một neuron ẩn có một độ lệch (bias) và 5 × 5 trọng số liên kết với trường tiếp nhận cục bộ. Những gì chúng ta vẫn chưa đề cập đến là chúng ta sẽ sử dụng các trọng số và độ lệch tương tự cho mỗi nơron ẩn 24 × 24. Nói cách khác, đối với những neuron ẩn thứ j, k, đầu ra là:

**(10)**

Ở đây, σ là hàm kích hoạt neuron - có lẽ là hàm sigmoid chúng ta sử dụng trong các chương trước. b là giá trị chung cho độ lệch. Mwl, m là một mảng 5 × 5 của trọng số chia sẻ. Và, cuối cùng, chúng ta sử dụng axy biểu thị giá trị kích hoạt đầu vào tại vị trí x, y.



Trên đây là 20 ảnh tương ứng với 20 feature map khác nhau (hay còn gọi là bộ lọc, hay là nhân). Mỗi feature map được thể hiện là một hình khối kích thước 5 × 5, tương ứng với 5 × 5 trọng số trong trường tiếp nhận cục bộ. Khối trắng có nghĩa là một trọng số nhỏ hơn, vì vậy các feature map đáp ứng ít hơn để tương ứng với điểm ảnh đầu vào. Khối sẫm màu hơn có nghĩa là trọng số lớn hơn, do đó, các feature map đáp ứng nhiều hơn với các điểm ảnh đầu vào tương ứng. Những hình ảnh trên cho thấy các kiểu đặc trưng mà lớp tích chập đáp ứng.

Một ưu điểm quan trọng của trọng số và độ lệch chia sẻ là nó làm giảm đáng kể số lượng các tham số liên quan đến một mạng tích chập. Đối với mỗi bản đồ đặc trưng chúng ta cần 25 = 5 × 5 trọng số chia sẻ và một độ lệch chia sẻ duy nhất. Vì vậy, mỗi bản đồ đực trưng cần 26 tham số.

Nếu chúng ta có 20 bản đồ đặc trưng thì cần 20 x 26 = 520 tham số để xác định lớp tích chập. Bây giờ chúng ta hãy làm phép so sánh, giả sử chúng ta có lớp đầu tiên kết nối đầy đủ, tức là có 784 = 28 × 28.784 = 28 × 28 nơron đầu vào, và số nơron lớp ẩn khiêm tốn là 30, như chúng ta sử dụng trong rất nhiều các ví dụ trước đó trong cuốn sách. Như vậy cần 784 × 30 trọng số, cộng thêm 30 sai lệch (bias), tổng số 23,550 tham số (parameter). Nói cách khác, lớp kết nối đầy đủ (fully – connected layer) sẽ cần số lượng tham số nhiều gấp 40 lần so với lớp tích chập (convolutional layer).

Tất nhiên, chúng ta không thể thực sự làm một so sánh trực tiếp giữa số lượng các tham số, bởi vì hai mô hình này khác nhau. Nhưng về trực giác dường như việc sử dụng bất biến dịch của các lớp tích chập sẽ giảm số lượng các tham số cần thiết mà vẫn đạt được hiệu quả giống như các mô hình kết nối đầy đủ. Mô hình mạng tích chập sẽ cho kết quả huấn luyện nhanh hơn giúp chúng ta xây dựng mạng sâu hơn sử dụng các lớp tích chập. Cái tên “convolutional” xuất phát là các hoạt động trong phương trình đôi  
khi được biết đến như convolution. Chính xác hơn một chút, người ta đôi khi viết phương trình như a1 = σ (b + w \* a0), trong đó a1 là tập kích hoạt đầu ra từ một bản đồ đặc trưng, a0 là tập hợp các kích hoạt đầu vào, và \* được gọi là phép toán chập.

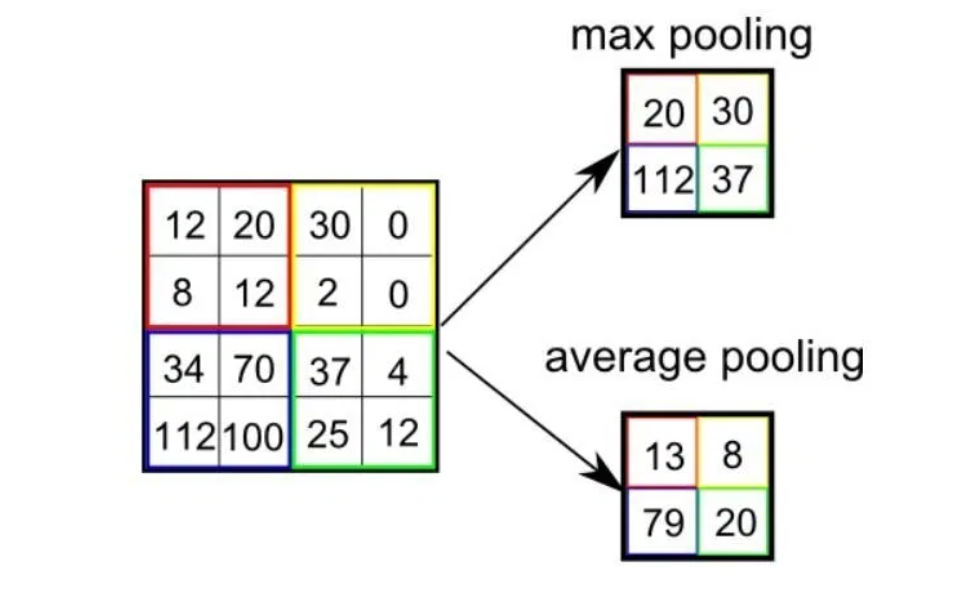
**Lớp thứ 2: Lớp gộp (Pooling)**

Ngoài các lớp tích chập vừa mô tả, mạng nơron tích chập cũng chứa các lớp pooling. Lớp pooling thường được sử dụng ngay sau lớp tích chập. Những gì các lớp pooling làm là đơn giản hóa các thông tin ở đầu ra từ các lớp tích chập nhưng vẫn giữ được các thuộc tính quan trọng. Kích thước dữ liệu giảm giúp giảm việc tính toán trong model.

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)

Gọi pooling size kích thước K\*K. Input của pooling layer có kích thước H\*W\*D, ta tách ra làm D ma trận kích thước H\*W. Với mỗi ma trận, trên vùng kích thước K\*K trên ma trận ta tìm maximum hoặc average của dữ liệu rồi viết vào ma trận kết quả. Quy tắc về stride và padding áp dụng như phép tính convolution trên ảnh.

Có 2 loại pooling layer phổ biến là: max pooling và average pooling.



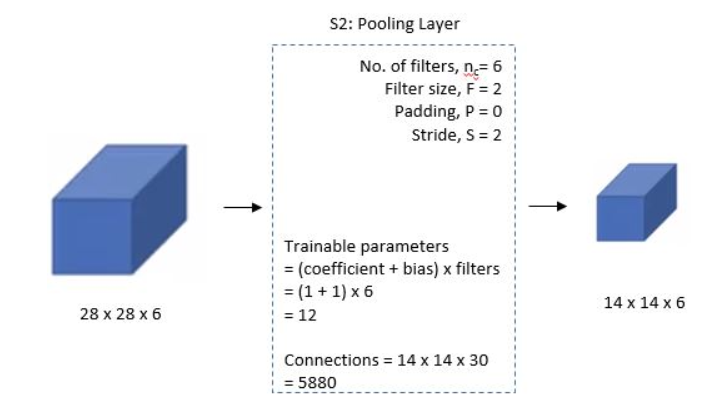
Hình 2.22 Các loại Pooling

* MaxPooling:

Maxpooling sẽ lấy ra giá trị lớn nhất mà kernel trượt trên ma trận ảnh. Ta dùng Maxpooling để trọng số ma trận tăng lên giúp cho khác biệt giữa các ma trận rõ hơn và độ chính xác sẽ cao hơn.

* AveragePooling:

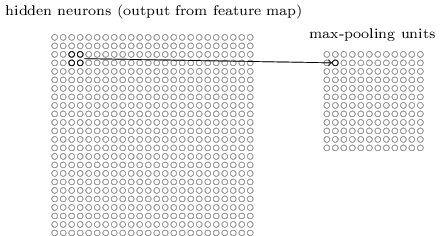
AveragePooling là lấy giá trị trung bình mà kernel trượt trên ma trận ảnh. Tuy Average có độ chính xác thấp hơn MaxPooling nhưng trong nhiều trường hợp ta phải dùng vì nó có sự sai số nhỏ hơn Maxpooling.



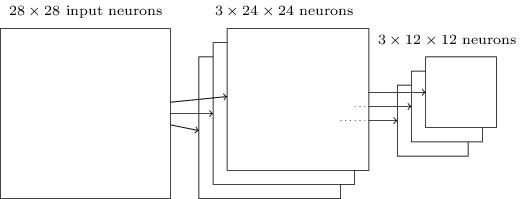
S2: Max Pooling Layer

LeNet-5 áp dụng lớp average pooling hoặc lớp lấy sub-sampling với kích thước bộ lọc 2 × 2 và sải bước sải là hai. Kích thước hình ảnh thu được sẽ giảm xuống còn 14x14x6.

Mỗi đơn vị trong lớp pooling có thể thu gọn một vùng 2 × 2 nơron trong lớp trước. Một thủ tục pooling phổ biến là max-pooling. Trong maxpooling, một đơn vị pooling chỉ đơn giản là kết quả đầu ra kích hoạt giá trị lớn nhất trong vùng đầu vào 2 × 2, như minh họa trong sơ đồ sau:



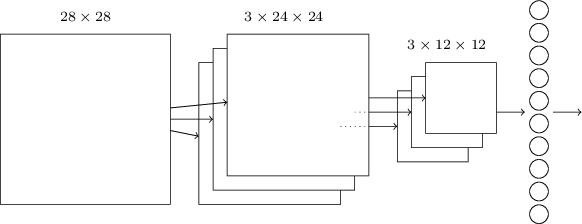
Lưu ý rằng bởi vì chúng ta có 24 × 24 nơron đầu ra từ các lớp tích chập, sau khi pooling chúng ta có 12 × 12 nơron. Như đã đề cập ở trên, lớp tích chập thường có nhiều hơn một feature map. Chúng ta áp dụng max-pooling cho mỗi feature map riêng biệt. Vì vậy, nếu có ba bản đồ đặc trưng, các lớp tích chập và max-pooling sẽ kết hợp như sau:



Chúng ta có thể hiểu max-pooling như là một cách cho mạng để hỏi xem một đặc trưng nhất được tìm thấy ở bất cứ đâu trong một khu vực của ảnh. Sau đó nó bỏ đi những thông tin định vị chính xác. Trực giác là một khi một đặc trưng đã được tìm thấy, vị trí chính xác của nó là không quan trọng như vị trí thô của nó so với các đặc trưng khác. Một lợi ích lớn là có rất nhiều tính năng gộp ít hơn (fewer pooled features), và vì vậy điều này sẽ giúp giảm số lượng các tham số cần thiết trong các lớp sau.

Max-pooling không phải là kỹ thuật duy nhất được sử dụng để pooling. Một phương pháp phổ biến khác được gọi là L2 pooling. Ở đây, thay vì lấy giá trị kích hoạt tối đa (maximum activation) của một vùng 2 × 2 nơron, chúng ta lấy căn bậc hai của tổng các bình phương của kích hoạt trong vùng 2 × 2. Trong khi các chi tiết thì khác nhau, nhưng về trực giác thì tương tự như max-pooling: L2 pooling là một cách để cô đọng thông tin từ các lớp tích chập. Trong thực tế, cả hai kỹ thuật đã được sử dụng rộng rãi. Và đôi khi người ta sử dụng các loại pooling khác.

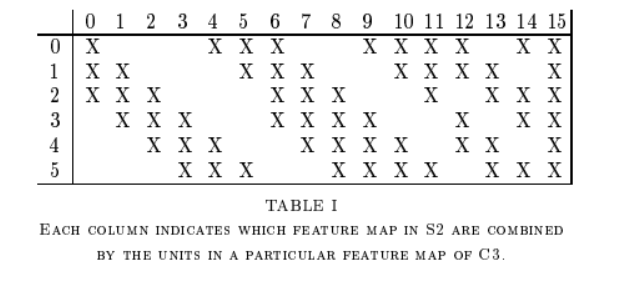
**Đặt tất cả chúng lại với nhau (Putting it all together):** Bây giờ chúng ta có thể đặt tất cả những ý tưởng lại với nhau để tạo thành một mạng tích chập hoàn chỉnh. Nó tương tự như kiến trúc chúng ta nhìn vào, nhưng có thêm một lớp 10 nơron đầu ra, tương ứng với 10 giá trị có thể cho các số MNIST ( '0', '1', '2', v.v…):



Mạng bắt đầu với 28 × 28 nơron đầu vào, được sử dụng để mã hóa các cường độ điểm ảnh cho ảnh MNIST. Sau đó là một lớp tích chập sử dụng 5 × 5 trường tiếp nhận cục bộ và 3 bản đồ đặc trưng. Kết quả là một lớp 3 × 24 × 24 nơron lớp ẩn. Bước tiếp theo là một lớp max-pooling, áp dụng cho 2 × 2 vùng qua 3 bản đồ đặc trưng (feauture maps). Kết quả là một lớp 3 × 12 × 12 nơron đặc trưng ở tầng ẩn.

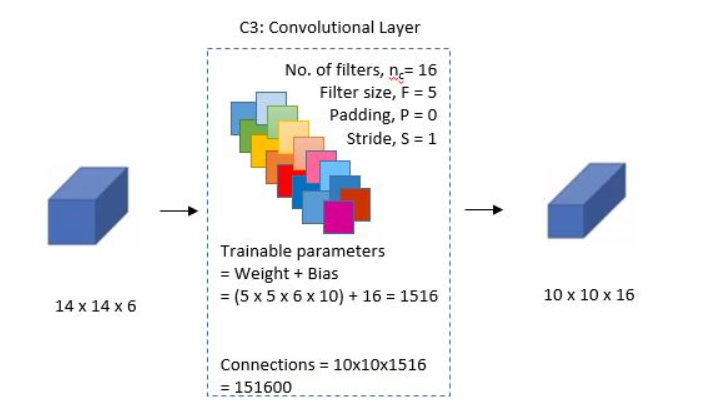
**Lớp thứ 3: Lớp tích chập thứ 2 (Conv layer)**

Tiếp theo, có một lớp tích chập thứ hai với 16 feature map có kích thước 5 × 5 và sải bước là 1. Trong lớp này, chỉ 10 trong số 16 feature map được kết nối với 6 feature map của lớp trước như hình dưới đây:



Bảng 2 from [LeCun et al., 1998]

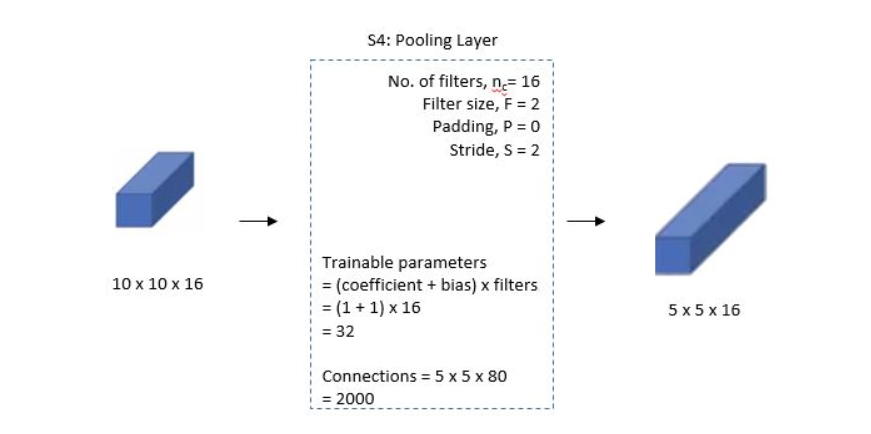
Nguyên nhân chính là để phá vỡ tính đối xứng trong mạng và giữ cho số lượng kết nối trong giới hạn hợp lý. Đó là lý do tại sao số lượng tham số đào tạo trong các lớp này là 1516 thay vì 2400 và tương tự, số lượng kết nối là 151600 thay vì 240000.



C3: Convolutional Layer

**Lớp thứ 4: Pooling layer**

Lớp thứ tư (S4) lại là lớp Maxpooling với kích thước bộ lọc 2 × 2 và sải bước là 2. Lớp này giống với lớp thứ hai (S2) ngoại trừ nó có 16 feature map nên đầu ra sẽ giảm xuống còn 5x5x16 .

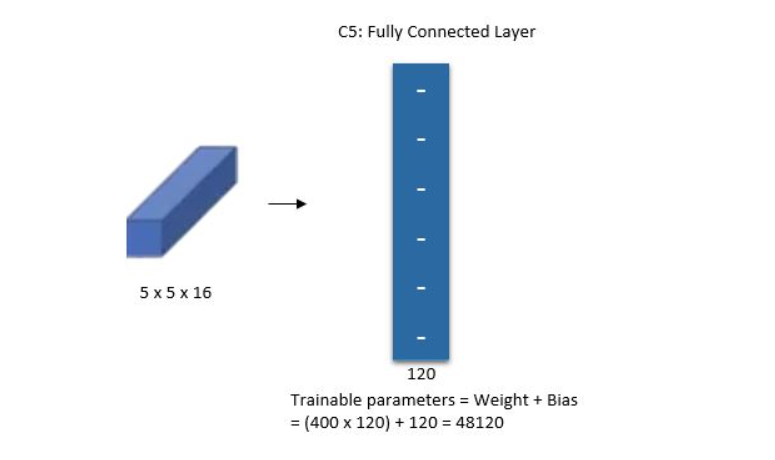


S4: Max Pooling Layer

**Lớp thứ 5: Lớp kết nối đầy đủ (fully connected convolutional)**

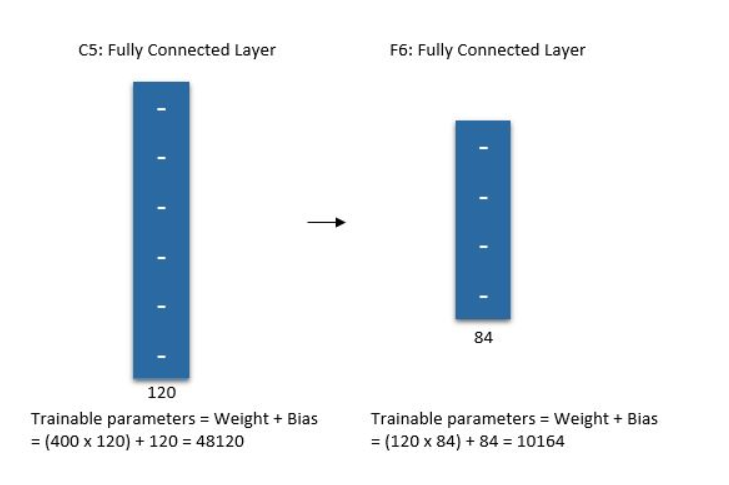
Sau khi ảnh được truyền qua nhiều convolutional layer và pooling layer thì model đã học được tương đối các đặc điểm của ảnh (ví dụ mắt, mũi, khung mặt,…) thì tensor của output của layer cuối cùng, kích thước H\*W\*D, sẽ được chuyển về 1 vector kích thước (H\*W\*D)

Lớp thứ năm (C5) là fully connected convolutional với 120 feature map với kích thước 1x1. Mỗi 120 đơn vị trong C5 được kết nối với tất cả 400 nút (5x5x16) trong lớp thứ tư S4.



C5: Fully Connected Convolutional Layer

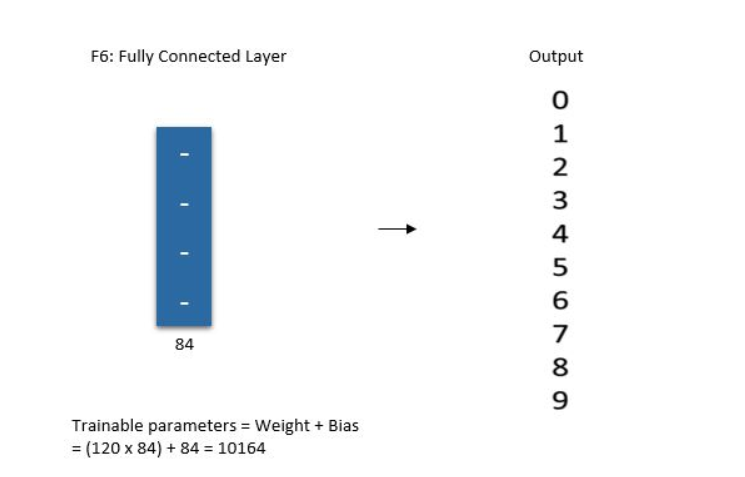
**Lớp thứ 6:** Là lớp kết nối đầy đủ với 84 đơn vị



F6: Fully Connected Layer

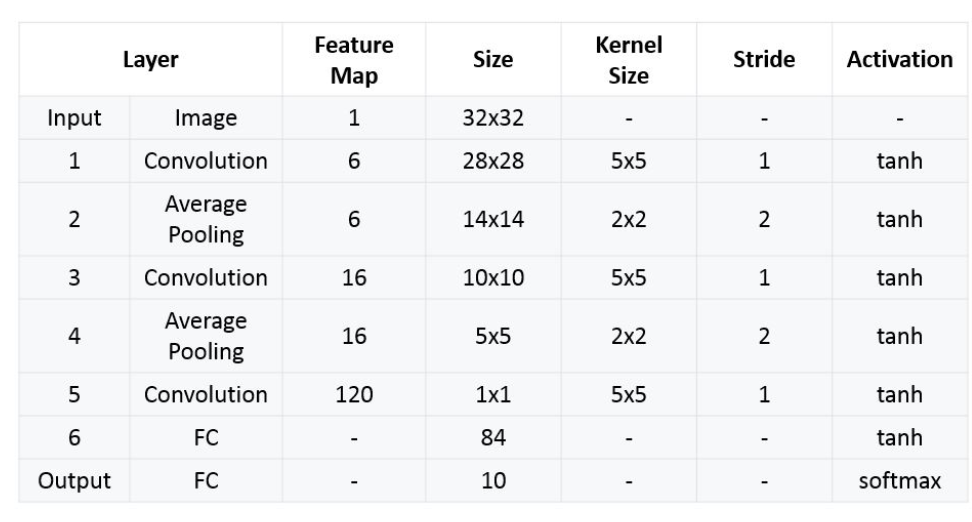
**Đầu ra:**

Cuối cùng, có một lớp đầu ra softmax được kết nối đầy đủ với 10 giá trị có thể có tương ứng với các chữ số từ 0 đến 9.



Fully Connected Output Layer

*Tóm lược kiến trúc mô hình lenet5*

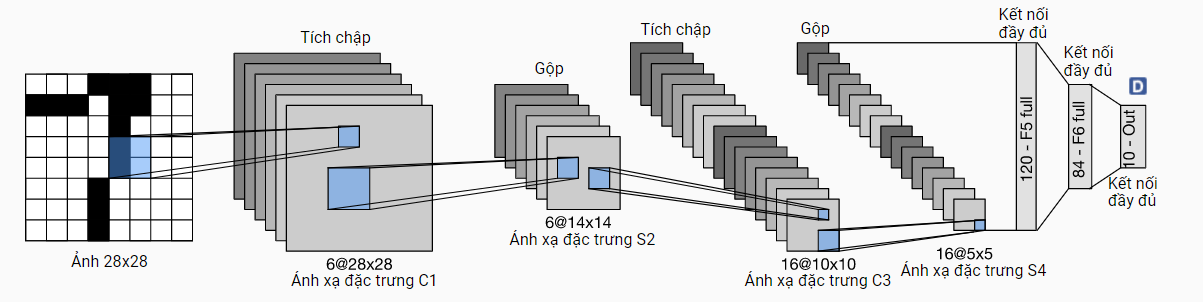


Bảng 3. Bảng tóm tắt kiến ​​trúc LeNet-5

# CHƯƠNG 3. XÂY DỰNG CHƯƠNG TRÌNH THỬ NGHIỆM

* 1. **Mô hình**

LeNet là một trong những mạng CNN lâu đời nổi tiếng nhất được Yann LeCUn phát triển vào những năm 1998s. Cấu trúc của LeNet gồm 2 layer (Convolution + maxpooling) và 2 layer fully connected layer và output là softmax layer . Chúng ta cùng tìm hiểu chi tiết architect của LeNet đối với dữ liệu mnist (accuracy lên đến 99%)



Hình 3.1 Mô hình nhận dạng kí tự sử dụng DeepLearning (Lenet5)

* 1. **Cài đặt chương trình**

Chương trình cài đặt sẽ gồm 2 phần là Model và App nhận dạng

* + 1. **Xây dựng Model**
* Để xây dựng được model yêu cầu cần có thư viện python, phần mềm jupyter notebook (có thể thay thế bằng colab hoặc pycharm)
* Import một số thư viện cần thiết: Tensorflow để xây dựng model, matplotlib để hiển thị hình ảnh, vẽ đồ thị…, thư viện numpy

import tensorflow as tf  
import matplotlib.pyplot as plt   
import tensorflow as tf  
from tensorflow import keras  
import numpy as np

Để import được các thư viện thì ta cần install trước theo cú pháp:

!pip install <lib>

* Tập dữ liệu MNIST được tải về bằng Tensorflow: tf.keras.datasets.mnist.load\_data()
* Chia tập dữ liệu thành 2 phần là train\_set và test\_set

(training\_images, training\_labels), (test\_images, test\_labels) = mnist.load\_data()

* Chuẩn hóa lại dữ liệu bằng cách chia các giá trị của pixel cho 255.0

training\_images = training\_images / 255.0  
training\_images = training\_images.reshape(60000, 28,28, 1) # lớp convolution (ảnh xám)

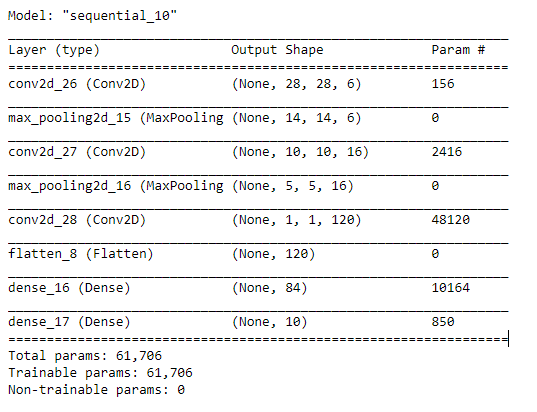
test\_images = test\_images / 255.0  
test\_images = test\_images.reshape(10000, 28,28,1)

* Import một số layer cần thiết để xây dựng cấu trúc model

from tensorflow.keras.layers import Conv2D  
from tensorflow.keras.layers import Activation  
from tensorflow.keras.layers import MaxPooling2D  
from tensorflow.keras.layers import AveragePooling2D  
from tensorflow.keras.layers import Flatten # duỗi ra  
from tensorflow.keras.layers import Dense #thu nhỏ lại  
from tensorflow.keras.models import Sequential #Khởi tạo model  
from tensorflow.keras.layers import Dropout # loại bỏ một số ngẫu nhiên các neuron, từ đó giúp cho quá trình huấn luyện không bị overfitting

* Xây dựng model

Tổng quan kiến trúc:



Hình 3.2 Kiến trúc model

* Train model

# Compile the model  
model.compile(optimizer='adam', loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])

# sparse\_categorical\_crossentropy làm việc với label là số  
# adam dựa vào độ lớn của đạo hàm sẽ tự động find tooling learning range ==> train nhanh hơn hiệu quả hơn

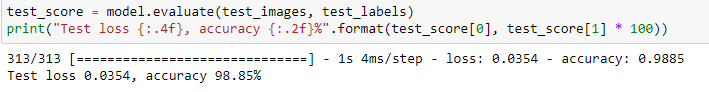
# Tiến hành training  
hist = model.fit(training\_images, training\_labels, epochs=10,batch\_size=128, validation\_data=(test\_images, test\_labels), verbose=1)

# model chỉ học thôi chứ không nhớ nên độ chính xá không bao giờ = 100%

# sẽ có train\_image/(tổng epoch) ảnh được train trên mỗi epoch

* Đánh giá model sau khi train

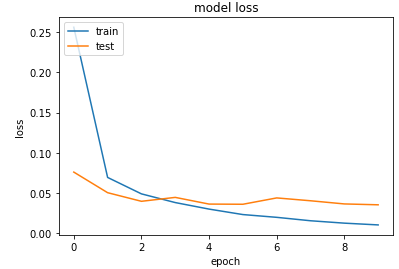
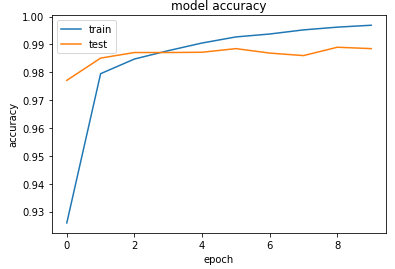
Độ chính xác của model là 98,85%, tổn thất là 3,54%



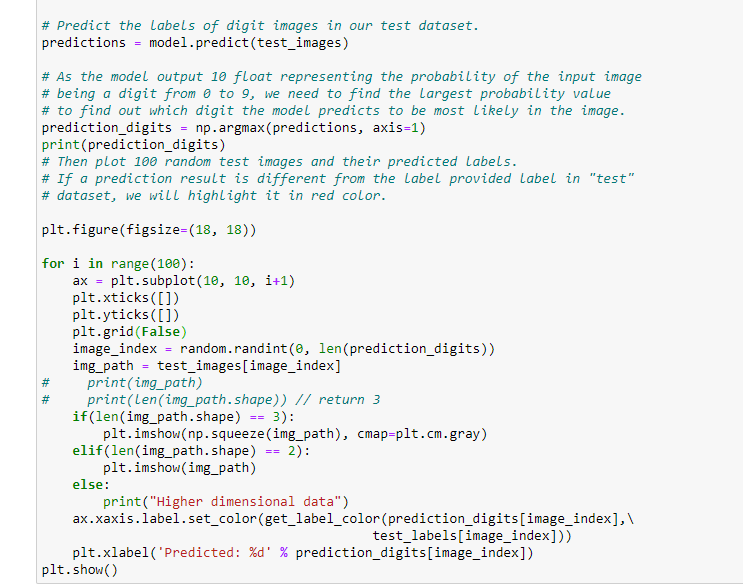
Hình 3.3 Đánh giá model

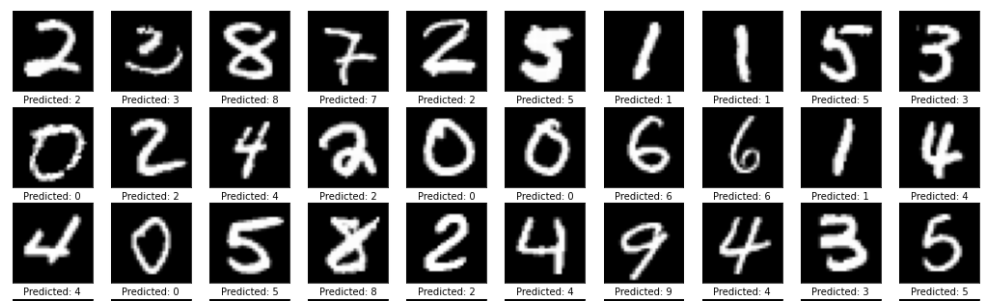
* Vẽ đồ thị thể hiện độ chính xác và tổn thất của model

Để vẽ được biểu đồ ta sử dụng thư viện matplotlib



Hình 3.4 Biểu đồ thể hiện độ chính xác(trái), mất mát (phải)

* Detect để thử một vài image trong tập test\_set

Kết quả:

* Tiến hành convert model sang .tflite để mobile sử dụng

# Convert Keras model to TF Lite format.  
converter = tf.lite.TFLiteConverter.from\_keras\_model(model)  
tflite\_model = converter.convert()

# Save the TF Lite model as file  
f = open('lenet5.tflite', "wb")  
f.write(tflite\_model)  
f.close()

* + 1. **Xây dựng App demo**

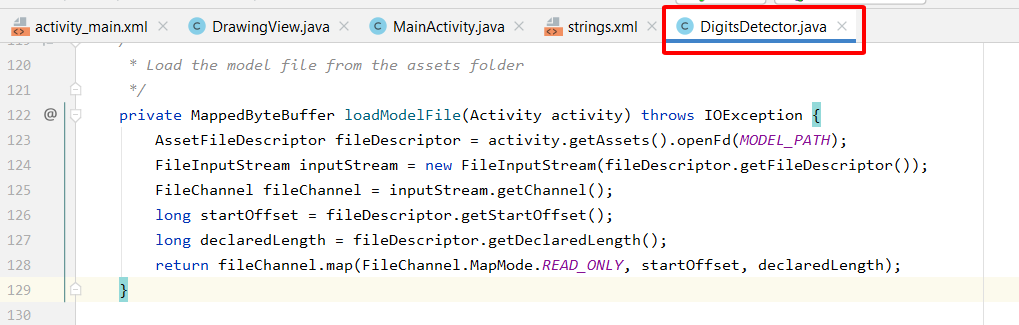
**Config**

Để dụng được Tensorflow lite trên mobile thì ta phải implement thư viện vào phần dependence của build.grade app

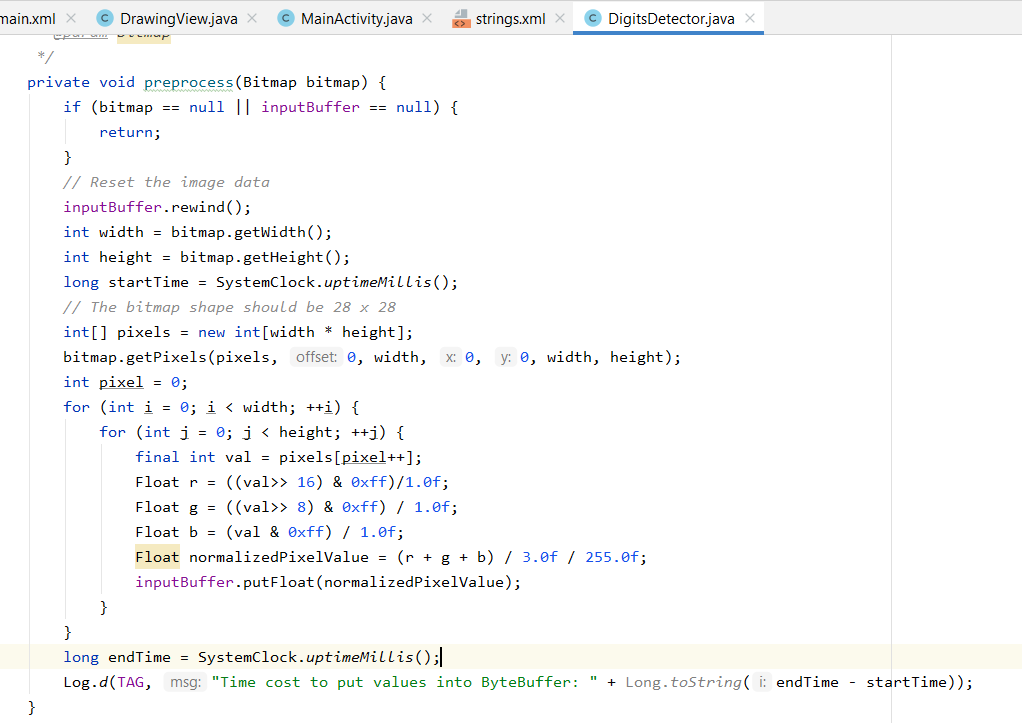
implementation 'org.tensorflow:tensorflow-lite:2.0.0'

Thêm

aaptOptions **{** noCompress "tflite"  
 noCompress "lite"  
 **}** vào config android

**Load model trên mobile**

Chuyển sang ảnh xám và lưu vào ByteBuffer để model nhận diện



* + 1. **Kết quả một số trường hợp dự đoán**

|  |  |
| --- | --- |
| Số 3 là mô hình đoán đúng nhiều nhất   * Số dự đoán 3 * Độ chính xác gần 100% | Số 4 thì độ chính xác thấp hơn nhưng vẫn đúng   * Số dự đoán 4 * Độ chính xác: 88,39% |
|  |  |

|  |  |
| --- | --- |
| Số 0 detect sai nhưng sác xuất lại khá lớn | Số 0 detect đúng nhưng xác suất lại chưa cao lắm |
|  |  |

|  |  |
| --- | --- |
| Số 7 detect sai nhưng sác xuất lại rất cao | Số 1 và số 7 dễ bị nhầm lẫn với nhau |
|  |  |

Link source code: <https://github.com/lethu123/Project-Graduate>

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

**Kết luận**

Qua luận văn này, em đã tìm hiểu và nắm khái quát được những kiến thức của mạng neural nhân tạo. Sử dụng mô hình CNN trong công việc xây dựng nhận dạng ký tự số viết tay. Luận văn đã được hoàn thành về mặt cơ bản nội dung đã đề ra. Đó là sự kết hợp giữa xử lí ảnh và lý thuyết nhận dạng nhằm giải quyết một phần bài toán nhận dạng ký tự số viết tay. Chương trình đang bước đầu thử nghiệm và đạt được kết quả thực nghiệm dựa trên 10.000 tập mẫu với độ chính xác trên 99, 85%.

**Hạn chế**

Tuy đã hoàn thành về nội dung và chương trình nhận dạng nhưng vẫn còn một số hạn chế cần được khắc phục như: Chương trình đang ở mức thử nghiệm, chưa áp dụng vào thực tế, nhận dạng ký tự còn sai nhiều.

**Huớng phát triển**

* Nâng cao hiệu quả chương trình, nhận diện nhiều kí tự cùng lúc.
* Phát triển chương trình thành module phần cứng. Có khả năng tương thích với các thiết bị quan sát như camera.
* Nghiên cứu theo hướng một ứng dụng cụ thể như : scan văn bản, nhận dạng CMND và các phần mềm sử dụng trong ngân hàng, sử dụng cho các nhà trường cho việc nhạp điểm tự động…

# PHỤ LỤC

**[1] LeCun:** (8/7/1960)là một nhà khoa học máy tính người Pháp làm việc chủ yếu trong các lĩnh vực máy học, thị giác máy tính, robot di động và khoa học thần kinh tính toán. Ông là Giáo sư Bạc của Viện Khoa học Toán học Courant tại Đại học New York, và Phó Chủ tịch, Trưởng nhóm Khoa học AI tại Facebook.

<https://scholar.google.com/citations?user=WLN3QrAAAAAJ&hl=en>

[2] <https://www.microsoft.com/en-us/research/wp-content/uploads/2003/08/icdar03.pdf>

[3] Research Group on Artiﬁcial Intelligence of the Hungarian Academy of Sciences and University of Szeged, Aradiv´ertan´uk tere 1., H-6720 Szeged, Hungary

[4] <https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning>

[5] <https://en.wikipedia.org/wiki/Support_vector_machine>

# TÀI LIỆU THAM KHẢO

1. <https://engmrk.com/lenet-5-a-classic-cnn-architecture/>

**2.** [**https://dlapplications.github.io/2018-07-06-CNN/**](https://dlapplications.github.io/2018-07-06-CNN/)

**3.** <https://d2l.aivivn.com/chapter_convolutional-neural-networks/index_vn.html>

**4.** <https://machinelearningcoban.com/2016/12/26/introduce/>

**5.** <https://developer.android.com/codelabs/digit-classifier-tflite?hl=Es#5>