KHOA KỸ THUẬT VÀ CÔNG NGHỆ

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**

****

**ĐỒ ÁN CƠ SỞ NGÀNH**

**HỌC KỲ 2, NĂM HỌC 2020 - 2021**

**TÌM HIỂU MẠNG NEURAL**

**NETWORK ANN VÀ ỨNG DỤNG**

*Giáo viên hướng dẫn:*

Họ tên: Ngô Thanh Huy

*Sinh viên thực hiện:*

Họ tên: Phan Minh Nhựt

MSSV: 110119038

Lớp: DA19TTA

*Trà Vinh, tháng 06 năm 2021*

KHOA KỸ THUẬT VÀ CÔNG NGHỆ

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**

****

**ĐỒ ÁN CƠ SỞ NGÀNH**

**HỌC KỲ 2, NĂM HỌC 2020 - 2021**

**TÌM HIỂU MẠNG NEURAL**

**NETWORK ANN VÀ ỨNG DỤNG**

*Giáo viên hướng dẫn:*

Họ tên: Ngô Thanh Huy

*Sinh viên thực hiện:*

Họ tên: Phan Minh Nhựt

MSSV: 110119038

Lớp: DA19TTA

*Trà Vinh, tháng 06 năm 2021*

**NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

*Trà Vinh, ngày ….. tháng …… năm ……*

**Giáo viên hướng dẫn**

*(Ký tên và ghi rõ họ tên)*

**NHẬN XÉT CỦA THÀNH VIÊN HỘI ĐỒNG**

*Trà Vinh, ngày ….. tháng …… năm ……*

**Thành viên hội đồng**

*(Ký tên và ghi rõ họ tên)*

**LỜI CẢM ƠN**

Em xin chân thành cảm ơn thầy, cô khoa Kĩ Thuật và Công Nghệ, bộ môn Công nghệ Thông Tin của Trường Đại học Trà Vinh đã tận tình dạy dỗ, truyền đạt cho em nhiều kiến thức, kinh nghiệm quý báu trong suốt quá trình học trong trường. Đặc biệt, em xin tỏ lòng biết ơn sâu sắc đến giảng viên, thầy Ngô Thanh Huy đã trực tiếp dìu dắt, giúp đỡ em tận tình, chu đáo trong suốt thời gian em hoàn thiện đồ án cơ sở ngành.

Xin chân thành cảm ơn các bạn trong lớp Công Nghệ Thông Tin A khoá 2019 của trường Đại Học Trà Vinh đã giúp đỡ, động viên tôi rất nhiều trong quá trình thực hiện đề tài.

Em xin chân thành cảm ơn!

Trà Vinh, tháng 06 năm 2021

# MỤC LỤC

[MỤC LỤC 6](#_Toc85051578)

[DANH MỤC HÌNH ẢNH – BẢNG BIỂU 8](#_Toc85051579)

[TÓM TẮT ĐỒ ÁN CƠ SỞ NGÀNH 9](#_Toc85051580)

[MỞ ĐẦU 10](#_Toc85051581)

[CHƯƠNG 1: TỔNG QUAN 11](#_Toc85051582)

[1.1 Trước tiên cần tìm hiểu Neural Network là gì? 11](#_Toc85051583)

[1.2 Artificial Neural Network ( ANN ) là gì? 11](#_Toc85051584)

[1.3 Đặc điểm của Artificial Neural Network ( ANN ) là gì? 11](#_Toc85051585)

[1.4 Tránh nhầm lẫn giữa các định nghĩa Neural Network 12](#_Toc85051586)

[CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT 14](#_Toc85051587)

[2.1 Cở sở lý thuyết, lý luận 14](#_Toc85051588)

[2.1.1 Ý tưởng tạo ra Artificial Neural Network ( ANN ). 14](#_Toc85051589)

[2.1.2 Cấu tạo của Artificial Neural Network ( ANN ). 15](#_Toc85051590)

[2.1.3 Hàm kích hoạt 17](#_Toc85051591)

[2.1.4 Cách Artificial Neural Network ( ANN ) hoạt động. 20](#_Toc85051592)

[2.1.5 Mạng nơ-ron nhân tạo so với các kỹ thuật học máy khác 22](#_Toc85051593)

[2.1.6 Mạng thần kinh nhân tạo so với AI cổ điển: 22](#_Toc85051594)

[2.2 Giả thuyết khoa học 23](#_Toc85051595)

[2.2.1 Huấn lyện mạng Artificial Neural Network. 23](#_Toc85051596)

[2.2.2 Phương pháp huấn luyện mạng Artificial Neural Network. 25](#_Toc85051597)

[2.2.3 Quá trình huấn luyện ANN 26](#_Toc85051598)

[2.3 Phương pháp nghiên cứu 28](#_Toc85051599)

[CHƯƠNG 3: ĐÁNH GIÁ KẾT QUẢ 30](#_Toc85051600)

[3.1 Mô tả công việc nghiên cứu 30](#_Toc85051601)

[3.2 Kết quả nghiên cứu 30](#_Toc85051602)

[3.3 Ưu điểm, nhược điểm của Artificial Neural Network. 39](#_Toc85051603)

[CHƯƠNG 4: KẾT LUẬN 40](#_Toc85051604)

[CHƯƠNG 5: HƯỚNG PHÁT TRIỂN 41](#_Toc85051605)

[DANH MỤC TÀI LIỆU THAM KHẢO 42](#_Toc85051606)

[PHỤ LỤC 43](#_Toc85051607)

# DANH MỤC HÌNH ẢNH – BẢNG BIỂU

[Hình 1 Thuật toán CNN 13](#_Toc85050897)

[Hình 2 Mạng nơ-ron của con người 14](#_Toc85050898)

[Hình 3 Hình ảnh minh hoạ của hệ số trên 15](#_Toc85050899)

[Hình 4 Kiến trúc cở bản của mạng ANN 16](#_Toc85050900)

[Hình 5 Mô tả sự thay đổi của vùng quyết định khi tăng số lượng lớp ẩn (hidden layer). 17](#_Toc85050901)

[Hình 6 Hàm kích hoạt 18](#_Toc85050902)

[Hình 7 Các giá trị hàm kích hoạt 19](#_Toc85050903)

[Hình 8 Cấu trúc hoạt động của mạng nơ-ron 20](#_Toc85050904)

[Hình 9 Ví dụ mạng nơ-ron nhiều lớp 21](#_Toc85050905)

[Hình 10 Mỗi lớp mạng nơ-ron sẽ trích xuất các tính năng từ hình ảnh đầu vào. 24](#_Toc85050906)

[Hình 11 Quá trình huấn luyện ANN 26](#_Toc85050907)

[Hình 12 Ảnh minh hoạ 27](#_Toc85050908)

[Hình 13 Một số ảnh đã được gắn nhãn tương ứng. 28](#_Toc85050909)

[Hình 14 Huấn luyện lần 1 31](#_Toc85050910)

[Hình 15 Huấn luyện lần 10 32](#_Toc85050911)

[Hình 16 Ma trận hỗn loạn 33](#_Toc85050912)

[Hình 17 Tỷ lệ nhận dạng với tỷ lệ học LR = 0,03 (a) và tỷ lệ học LR = 0,1 (b) 34](#_Toc85050913)

[Hình 18 Kết quả mạng ANN khi kết hợp 2 giá trị LR khác nhau. 35](#_Toc85050914)

[Hình 19 Tỷ lệ nhận dạng đúng với các số lượng lớp ẩn khác nhau 36](#_Toc85050915)

[Hình 20 Khảo sát mạng ANN với số lượng neuron khác nhau. 37](#_Toc85050916)

[Hình 21 Kết quả thực nghiệm 38](#_Toc85050917)

# TÓM TẮT ĐỒ ÁN CƠ SỞ NGÀNH

Mạng neuron nhân tạo, Artificial Neural Network (ANN), là một mô hình xử lý thông tin dựa theo cách thức xử lý thông tin của các hệ neuron sinh học. ANN được ứng dụng rộng rãi trong lĩnh vực khoa học máy tính, cụ thể là các dạng bài toán phân lớp: dự đoán, nhận dạng,… ANN được tạo nên từ một lượng các phần tử (neuron) kết nối với nhau thông qua các trọng số liên kết. Vấn đề nghiên cứu ở đây là tìm hiểu về mạng Neural Network ANN và Ứng dụng của nó. Mô hình ứng dụng thực nghiệm được sử dụng là “nhận diện chữ số viết tay bằng mạng Artificial Neural Network (ANN)”[1]. Kết quả nghiên cứu cho thấy mạng ANN nhận dạng chữ số với độ chính xác lên đến 95%.

# MỞ ĐẦU

1. **Lý do chọn đề tài**

Hiện nay, với hệ thống dữ liệu hình ảnh khổng lồ trên toàn thế giới, việc phân tích, xử lý để khai thác sử dụng thông tin trong ảnh là một nhu cầu thiết yếu. Tuy nhiên, với số lượng dữ liệu khổng lồ, việc phân tích, xử lý thủ công sẽ mất rất nhiều thời gian và nguồn nhân lực. Thực tiễn đã chứng minh, việc ứng dụng trí tuệ nhân tạo vào các nhiệm vụ trên đã giúp tiết kiệm được nhiều thời gian và công sức. Từ giữa năm 2011, Google đã giới thiệu dự án Deep Learning sử dụng mạng neuron nhân tạo dùng cho nhận dạng giọng nói và sau đó mở rộng lên các lĩnh vực khác như Gmail, Google dịch, Google ảnh. Do vậy để tiếp cận với trí tuệ nhân tạo, ta phải nghiên cứu từ cái nhỏ nhất, nên em chọn tìm hiểu mạng Neural Network ANN, vì nó dể dàng tiếp cận và nghiên cứu, từ đó có thể nâng cao sự hiểu biết của mình về trí tuệ nhận tạo.

1. **Mục đích nghiên cứu**

Tìm hiểu về mạng Artificial Neural Network (ANN), tìm hiểu những thuật ngữ, cấu tạo của ANN cũng nhưng cách hoạt động của đó. Cách nó “học” các thông số để đánh giá cho bài toán nhận dạng chữ số viết tay.

1. **Đối tượng nghiên cứu**

* Mạng Artificial Neural Network.
* Ứng dụng nhận dạng chữ số viết tay dùng mạng Artificial Neural Network.

1. **Phạm vi nghiên cứu**

* Tìm hiểu về lý thuyết của ANN trên các trang web.
* Xây dựng hệ thống nhận dạng chữ số viết tay trên ngôn ngữ python với thư viện MNIST [2] (Modified National Institute of Standards and Technology).

# CHƯƠNG 1: TỔNG QUAN

## 1.1 Trước tiên cần tìm hiểu Neural Network là gì?

Neural Network đọc tiếng việt là mạng nơ-ron nhân tạo, đây là một chuỗi những thuật toán được đưa ra để tìm kiếm các mối quan hệ cơ bản trong tập hợp các dữ liệu. Thông qua việc bắt bước cách thức hoạt động từ não bộ con người.  Nói cách khác, mạng nơ-ron nhân tạo được xem là hệ thống của các tế bào thần kinh nhân tạo. Đây thường có thể là hữu cơ hoặc nhân tạo về bản chất.

Neural Network có khả năng thích ứng được với mọi thay đổi từ đầu vào. Do vậy, nó có thể đưa ra được mọi kết quả một cách tốt nhất có thể mà bạn không cần phải thiết kế lại những tiêu chí đầu ra.

Khái niệm này có nguồn gốc từ [trí tuệ nhân tạo](https://blog.itnavi.com.vn/lap-trinh-ai/?amp), đang nhanh chóng trở nên phổ biến hơn trong sự phát triển của những hệ thống giao dịch điện tử.

## 1.2 Artificial Neural Network ( ANN ) là gì?

Artificial Neural Network là mạng nơ-ron nhân tạo và là mô hình toán học hoặc mô hình toán được xây dựng thông qua các nơ-ron sinh học. Nó bao gồm các nhóm việc, nơ-ron nhân tạo có thể nối hút với nhau và xử lý các thông tin bằng biện pháp truyền theo các kết nối rồi tính toán giá trị mới tại các nút.

Ở nhiều trường hợp, mạng Artificial Neural Network là hệ thống thích ứng có thể tự thay đổi cấu trúc của mình dựa vào thông tin bên ngoài hoặc bên trong chảy qua mạng ở quá trình “học”.

Nhiều mạng Artificial Neural Network còn là công cụ giúp mô hình hóa dữ liệu thống kê phi tuyến.

Chúng còn được sử dụng để mô hình hóa cho các mối quan hệ có tính phức tạp giữa các dữ liệu vào hoặc giữa kết quả để tìm kiếm mẫu trong dữ liệu.

## 1.3 Đặc điểm của Artificial Neural Network ( ANN ) là gì?

Trong lĩnh vực tài chính, mạng nơ-ron nhân tạo hỗ trợ cho quá trình phát triển các quy trình như: giao dịch thuật toán, dự báo chuỗi thời gian, phân loại chứng khoán, mô hình rủi ro tín dụng và xây dựng chỉ báo độc quyền và công cụ phát sinh giá cả.

Mạng nơ-ron nhân tạo có thể hoạt động như mạng nơ-ron của con người. Mỗi một nơ-ron thần kinh trong nơ ron nhân tạo là hàm toán học với chức năng thu thập và phân loại các thông tin dựa theo cấu trúc cụ thể.

Mạng ANN có sự tương đồng chuẩn mạnh vối những phương pháp thống kê như đồ thị đường cong và phân tích hồi quy.

Mạng ANN có chứa những lớp bao hàm các nút được liên kết lại với nhau. Mỗi nút lại là một tri giác có cấu tạo tương tự với hàm hồi quy đa tuyến tính.

Bên trong một lớp tri giác đa lớp, chúng sẽ được sắp xếp dựa theo các lớp liên kết với nhau. Lớp đầu vào sẽ thu thập các mẫu đầu vào và lớp đầu ra sẽ thu nhận các phân loại hoặc tín hiệu đầu ra mà các mẫu đầu vào có thể phản ánh lại.

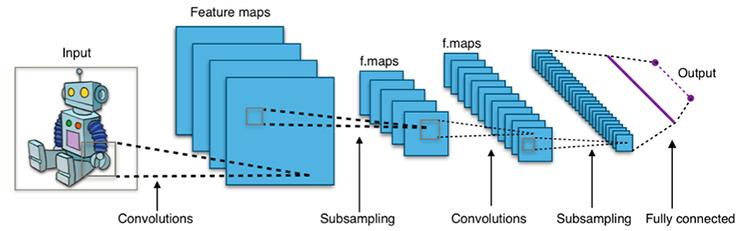
## 1.4 Tránh nhầm lẫn giữa các định nghĩa Neural Network

Không khó để bạn bắt gặp những định nghĩa hoặc những cụm từ có chứa từ neural network trong đó. Tuy nhiên, ý nghĩa của các cụm từ này lại hoàn toàn khác nhau. Vì vậy, cần phân biệt được chúng để tránh bị nhầm lẫn cho việc sử dụng về sau.

Convolutional Neural Network là gì?

Convolutional Neural Network (CNN) là một cụm từ dùng để chỉ mạng nơ -ron tích chập. Đây là mô hình Deep Learning tiên tiến cho phép chúng ta sử dụng được các hệ thống thông tin với độ chính xác vô cùng cao.

CNN được ứng dụng phổ biến trong những bài toán nhận dạng object trong ảnh. Còn ANN là mạng nơ-ron nhân tạo cần phân biệt kĩ.



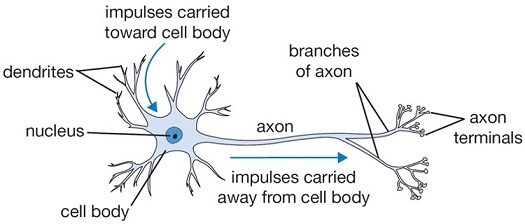
Hình 1 Thuật toán CNN

# CHƯƠNG 2: NGHIÊN CỨU LÝ THUYẾT

## 2.1 Cở sở lý thuyết, lý luận

### 2.1.1 Ý tưởng tạo ra Artificial Neural Network ( ANN ).

Artificial Neural Network (ANN) được hình thành bằng cách học theo cách mà não người xử lý thông tin. Các nơ-ron sinh học của chúng ta tiếp nhận thông tin từ các sợi nhánh (Dendrite) và đưa vào nhân (Nucleus). Nhân nơ-ron sẽ có trách nhiệm xử lý các thông tin đến và quyết định xem sẽ gửi tín hiệu gì tiếp theo đến các nơ-ron khác thông qua sợi trục (Axon).



Hình 2 Mạng nơ-ron của con người

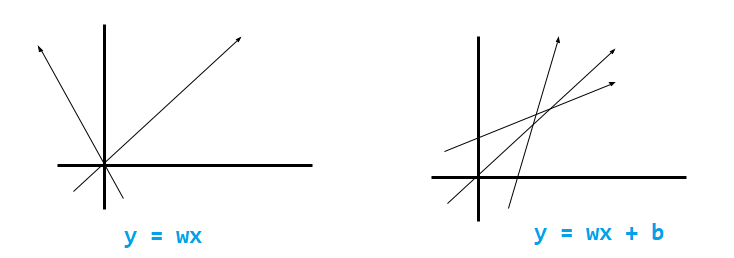
Vậy nơ-ron nhân tạo thì sẽ có cấu tạo như nào ?

Nơ-ron nhân tạo được thiết kế tương đồng như nơ-ron sinh học. Nó sẽ tiếp nhận các tín hiệu đầu vào, ứng với mỗi tín hiệu này sẽ là một trọng số (weight). Các trọng số (weight) này đại diện cho mức độ quan trọng của tín hiệu đầu vào. Các nơ-ron nhân tạo sẽ tổng hợp thông tin từ các đầu vào tương ứng theo trọng số (weight) của chúng và đưa vào hàm kích hoạt (activation function). Mạng thần kinh nhân tạo ANN được cấu tạo từ nhiều các nơron liên kết lại với nhau. Trong quá trình huấn luyện (training), các trọng số (weight) đầu vào của các nơron sẽ được tinh chỉnh sao cho phù hợp với tập huấn luyện (training set).

Phần tổng hợp trên nơ-ron nhân tạo chính là tổng trọng số của các đầu vào:

u=∑i=1mwixi

Phần tổng hợp u sẽ được cộng thêm với hệ số bias b. Bias đóng vai trò như hệ số chặn (intercept) trong hàm số tuyến tính (linear function), nó giúp cho các đường tuyến tính (linear) tự do hơn, do đó nó còn được biết đến với một cái tên là hệ số tự do. Hình 3 sẽ minh họa rõ hơn về sự hữu ích của hệ số này. Nếu không có hệ số bias b, đường tuyến tính (linear) của hàm số y=wx chỉ có thể đi qua gốc tọa độ (origin), như hình vẽ ở bên trái. Bằng việc thêm hệ số bias b, hàm số y=wx+b trở nên tự do hơn như hình bên phải, giúp cho model của chúng ta có thể dễ dàng thích ứng (fit) với tập huấn luyện (training set).



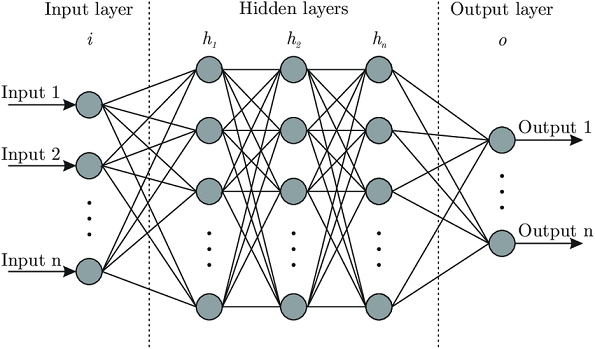
Hình 3 Hình ảnh minh hoạ của hệ số trên

Trước khi được đưa đến các nơ-ron tiếp theo hoặc đầu ra, các tín hiệu được tổng hợp sẽ được đưa qua hàm kích hoạt (activation). Được ký hiệu toán học nhu sau:

y=φ(u+b).

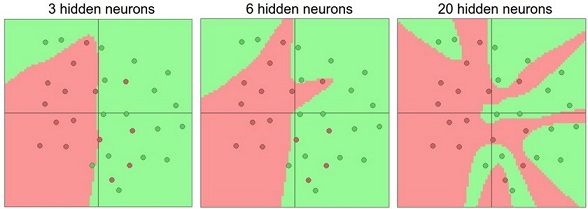
### 2.1.2 Cấu tạo của Artificial Neural Network ( ANN ).

Cấu tạo của một ANN bao gồm 3 lớp (layer) đó là lớp đầu vào (input layer), lớp ẩn (hidden layer) và lớp đầu ra (output layer). Thông tin sẽ được đi theo dạng truyền thẳng (feed forward) từ đầu vào cho đến đầu ra. Hình dưới đây mô tả cấu tạo cơ bản của một ANN:



Hình 4 Kiến trúc cở bản của mạng ANN

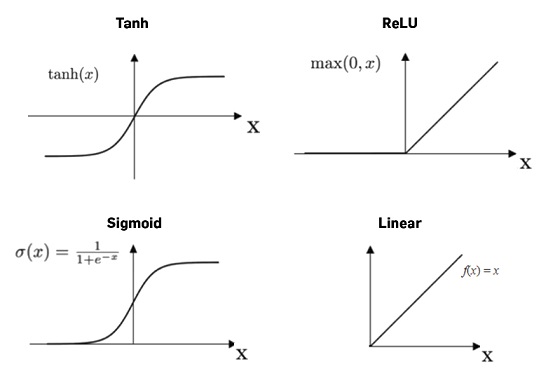
Lớp đầu vào (input layer) sẽ có số nơron tương ứng với số biến không phụ thuộc (independent variable). Lớp đầu ra thì sẽ có số nơron tương ứng với số nhóm (class) trong bài toán phân loại (classification) hoặc chỉ là duy nhất một nơron trong trường hợp bài toán hồi quy (regression).Còn lớp ẩn (hidden layer) có thể lựa chọn tùy ý cả về số lượng lớp (layer) và số lượng nơron trong mỗi lớp (layer) mà không có một quy định nào cụ thể. Thường thì số lượng nơ-ron hay được chọn sẽ là lũy thừa của 2, như 16, 32, 64, 128,… mà không có một lý do nào cụ thể. Về số lượng lớp ẩn (hidden layer) thì cần phải cân nhắc, càng nhiều lớp ẩn (hidden layer) sẽ càng khiến cho model quá vừa vặn (overfit) theo tập huấn luyện (training set).



Hình 5 Mô tả sự thay đổi của vùng quyết định khi tăng số lượng lớp ẩn (hidden layer).

### 2.1.3 Hàm kích hoạt

Hàm kích hoạt (activation function) luôn xuất hiện ở đầu ra của mỗi nơ-ron, nó sẽ đưa ra quyết định xem có đưa thông tin đã được tổng hợp sang nơ-ron khác hay không. Nhìn một cách tổng quát nó giống như một chiếc công tắc giúp bật tắt nơ-ron. Có rất nhiều loại hàm kích hoạt (activation function) và những nhà nghiên cứu vẫn đang tìm những hàm kích hoạt (activation function) mới hiệu quả hơn, 4 hàm kích hoạt (activation function) dưới đây là thông dụng hơn cả:



Hình 6 Hàm kích hoạt

Trong đó Sigmoid-Softmax và Linear được sử dụng ở các nơron đầu ra, tương ứng cho bài toán phân loại (classification) hay hồi quy (regression).

Các hàm kích hoạt (activation function) Sigmoid, Tanh và ReLU còn được gọi là các hàm kích hoạt phi tuyến tính (non-linear activation function) và được dùng trong kết nối giữa các nơron với nhau.

Nếu như hàm kích hoạt (activation function) Linear được sử dụng trong kết nối giữa các neuron, chúng ta sẽ vô hình chúng quy tất cả các lớp nơron về một vì tổng hợp các hàm tuyến tính thì vẫn sẽ là một hàm tuyến tính.

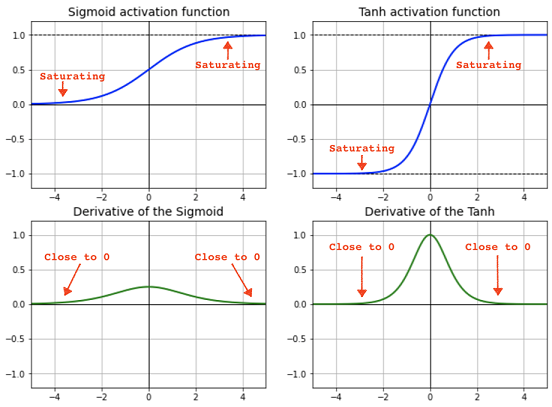
Đó là lý do mà các hàm kích hoạt phi tuyến tính (non-linear activation function) được sử dụng, từ đây chúng ta có thể kết hợp các lớp nơron một cách đúng nghĩa, mang đến khả năng xử lý những dữ liệu phức tạp với độ chính xác cao.

Trong số 3 hàm kích hoạt phi tuyến tính (non-linear activation function) trên, ReLU là hàm kích hoạt thường được sử dụng hơn cả, do hiện tượng tiêu biến độ dốc (vanishing gradient) ở hàm kích hoạt (activation function) Sigmoid và Tanh.

Hiện tượng tiêu biến độ dốc (vanishing gradient) xảy ra đối với các giá trị ở những điểm bão hòa của hàm số Tanh và Sigmoid, lúc này đạo hàm (derivative) trở nên rất nhỏ, hậu quả làm cho việc thay đổi các trọng số (weight) trong quá trình huấn luyện (training) gần như bằng 0, việc huấn luyện (training) của model gần như không có tác dụng.

Có một ví dụ rất hay về hiện tượng này, đó là như khi bạn nhấn ga một chiếc xe nhưng nó gần như không nhúc nhích về phía trước.

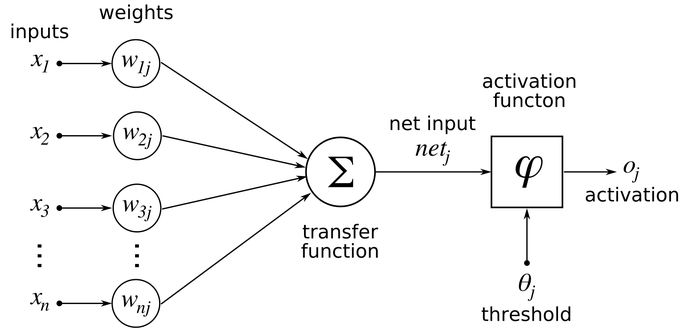
Với hàm kích hoạt (activation function) ReLU, các giá trị dương giờ đây được giữ nguyên giá trị, không còn những điểm bão hòa nữa, đạo hàm (derivative) luôn có độ lớn đủ tốt cho việc huấn luyện (training).



Hình 7 Các giá trị hàm kích hoạt

### 2.1.4 Cách Artificial Neural Network ( ANN ) hoạt động.

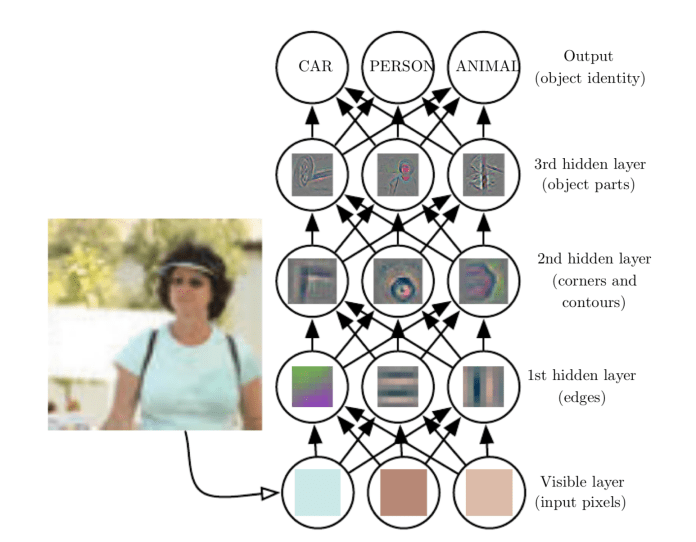
Thành phần cốt lõi của ANN là các tế bào thần kinh nhân tạo. Mỗi nơ-ron nhận đầu vào từ một số nơ-ron khác, nhân chúng với trọng số được chỉ định, cộng chúng và chuyển tổng cho một hoặc nhiều nơ-ron. Một số tế bào thần kinh nhân tạo có thể áp dụng một hàm kích hoạt cho đầu ra trước khi chuyển nó cho biến tiếp theo.



Hình 8 Cấu trúc hoạt động của mạng nơ-ron

Về cốt lõi, điều này có vẻ giống như một phép toán rất đơn giản. Nhưng khi đặt hàng trăm, hàng nghìn và hàng triệu tế bào thần kinh trong nhiều lớp và xếp chồng chúng lên nhau, sẽ có được một mạng thần kinh nhân tạo có thể thực hiện các nhiệm vụ rất phức tạp, chẳng hạn như [phân loại hình ảnh](https://bdtechtalks.com/2019/01/14/what-is-computer-vision/) hoặc nhận dạng giọng nói.

Mạng nơron nhân tạo bao gồm một lớp đầu vào, lớp này nhận dữ liệu từ các nguồn bên ngoài (tệp dữ liệu, hình ảnh, cảm biến phần cứng, micrô ...), một hoặc nhiều lớp ẩn xử lý dữ liệu và lớp đầu ra cung cấp một hoặc nhiều điểm dữ liệu dựa trên chức năng của mạng. Ví dụ, một mạng nơ-ron phát hiện người, ô tô và động vật sẽ có một lớp đầu ra với ba nút. Một mạng phân loại các giao dịch ngân hàng giữa an toàn và gian lận sẽ có một đầu ra duy nhất.



Hình 9 Ví dụ mạng nơ-ron nhiều lớp

### 2.1.5 Mạng nơ-ron nhân tạo so với các kỹ thuật học máy khác

Mạng nơ-ron nhân tạo chỉ là một trong số các thuật toán để thực hiện  [học máy](https://bdtechtalks.com/2017/08/28/artificial-intelligence-machine-learning-deep-learning/), một nhánh của trí tuệ nhân tạo phát triển hành vi dựa trên kinh nghiệm. Có nhiều kỹ thuật học máy khác có thể tìm thấy các mẫu trong dữ liệu và thực hiện các tác vụ như phân loại và dự đoán. Một số kỹ thuật này bao gồm mô hình hồi quy, máy vectơ hỗ trợ, phương pháp k-gần nhất và cây quyết định.

Tuy nhiên, khi nói đến việc xử lý dữ liệu lộn xộn và không có cấu trúc như hình ảnh, âm thanh và văn bản, mạng nơ-ron vượt trội hơn các kỹ thuật học máy khác.

Ví dụ: nếu bạn muốn thực hiện các nhiệm vụ phân loại hình ảnh bằng các thuật toán học máy cổ điển, bạn sẽ phải thực hiện nhiều “kỹ thuật tính năng” phức tạp, một quá trình phức tạp và gian khổ đòi hỏi nỗ lực của một số kỹ sư và chuyên gia miền. Mạng nơ-ron và thuật toán học sâu không yêu cầu kỹ thuật tính năng và tự động trích xuất các tính năng từ hình ảnh nếu được đào tạo tốt.

Tuy nhiên, điều này không có nghĩa là mạng nơ-ron là sự thay thế cho các kỹ thuật học máy khác. Các loại thuật toán khác yêu cầu ít tài nguyên tính toán hơn và ít phức tạp hơn, điều này khiến chúng thích hợp hơn khi bạn đang cố gắng giải quyết một vấn đề không yêu cầu mạng nơ-ron.

Các kỹ thuật học máy khác cũng có thể diễn giải được, có nghĩa là việc điều tra và sửa chữa các quyết định mà chúng đưa ra sẽ dễ dàng hơn. Điều này có thể làm cho chúng thích hợp hơn trong các trường hợp sử dụng mà khả năng diễn giải quan trọng hơn độ chính xác.

### 2.1.6 Mạng thần kinh nhân tạo so với AI cổ điển:

Các chương trình AI truyền thống, dựa trên quy tắc dựa trên các nguyên tắc của phần mềm cổ điển [3]. Các chương trình máy tính được thiết kế để chạy các hoạt động trên dữ liệu được lưu trữ trong các vị trí bộ nhớ và lưu kết quả trên một vị trí bộ nhớ khác. Logic của chương trình là tuần tự, xác định và dựa trên các quy tắc được xác định rõ ràng. Các hoạt động được điều hành bởi một hoặc nhiều bộ xử lý trung tâm.

Tuy nhiên, mạng nơ-ron không tuần tự, cũng không xác định. Ngoài ra, bất kể phần cứng bên dưới là gì, không có bộ xử lý trung tâm nào kiểm soát logic. Thay vào đó, logic được phân tán trên hàng nghìn tế bào thần kinh nhân tạo nhỏ hơn. ANN không chạy hướng dẫn; thay vào đó họ thực hiện các phép toán trên đầu vào của họ. Các hoạt động tập thể của họ phát triển hành vi của mô hình.

## 2.2 Giả thuyết khoa học

Để mạng nơ-ron nhân tạo có thể hoạt động cần phải huấn luyện cho nó “học”.

### 2.2.1 Huấn lyện mạng Artificial Neural Network.

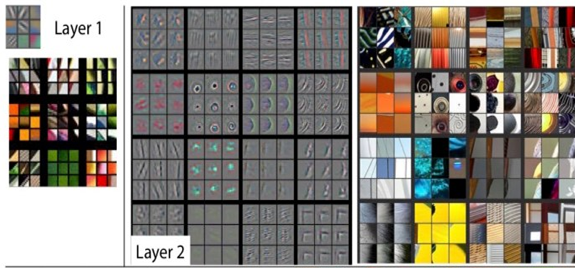
Mạng nơron nhân tạo bắt đầu bằng cách gán các giá trị ngẫu nhiên cho trọng số của các kết nối giữa các nơron. Chìa khóa để ANN thực hiện đúng và chính xác nhiệm vụ của mình là điều chỉnh các trọng số này về đúng số lượng. Nhưng việc tìm kiếm trọng số phù hợp không phải là điều dễ dàng, đặc biệt là khi đang xử lý nhiều lớp và hàng nghìn tế bào thần kinh.

Việc hiệu chuẩn này được thực hiện bằng cách “huấn luyện” mạng với các ví dụ được chú thích.

Ví dụ: nếu muốn đào tạo bộ phân loại hình ảnh được đề cập ở trên, cần cung cấp cho nó nhiều ảnh, mỗi ảnh được gắn nhãn với lớp tương ứng (người, ô tô hoặc động vật). Khi cung cấp cho nó ngày càng nhiều ví dụ đào tạo, mạng nơ-ron dần dần điều chỉnh trọng số của nó để ánh xạ từng đầu vào thành các đầu ra chính xác.

Về cơ bản, những gì xảy ra trong quá trình đào tạo là mạng tự điều chỉnh để thu thập các mẫu cụ thể từ dữ liệu. Một lần nữa, trong trường hợp của mạng phân loại hình ảnh, khi đào tạo mô hình AI với các ví dụ chất lượng, mỗi lớp sẽ phát hiện một lớp tính năng cụ thể.

Ví dụ: lớp đầu tiên có thể phát hiện các cạnh ngang và dọc, các lớp tiếp theo có thể phát hiện các góc và hình tròn. Xa hơn nữa trong mạng, các lớp sâu hơn sẽ bắt đầu chọn ra các tính năng nâng cao hơn như khuôn mặt và vật thể.



Hình 10 Mỗi lớp mạng nơ-ron sẽ trích xuất các tính năng từ hình ảnh đầu vào.

Khi chạy một hình ảnh mới thông qua một mạng nơ-ron được đào tạo tốt, trọng lượng được điều chỉnh của các nơ-ron sẽ có thể trích xuất các đặc điểm phù hợp và xác định chính xác hình ảnh thuộc lớp đầu ra nào.

Một trong những thách thức của việc đào tạo mạng nơ-ron là tìm ra số lượng và chất lượng phù hợp của các ví dụ đào tạo. Ngoài ra, việc đào tạo các mô hình AI lớn đòi hỏi lượng lớn tài nguyên máy tính. Để vượt qua thách thức này, nhiều kỹ sư sử dụng " học chuyển tiếp "[4], một kỹ thuật đào tạo trong đó bạn lấy một mô hình được đào tạo trước và tinh chỉnh mô hình đó bằng các ví dụ mới dành cho miền cụ thể. Học chuyển giao đặc biệt hiệu quả khi đã có một mô hình AI gần với trường hợp sử dụng của bạn.

### 2.2.2 Phương pháp huấn luyện mạng Artificial Neural Network.

- *Phương pháp học*: Mạng neural nhân tạo phỏng theo việc xử lý thông tin của bộ não người. Do vậy, đặc trưng cơ bản của mạng là có khả năng học, khả năng tái tạo các hình ảnh và dữ liệu khi đã học. Trong trạng thái học, thông tin được lan truyền theo hai chiều nhiều lần để học các trọng số. Có 3 kiểu học chính, mỗi kiểu học tương ứng với một nhiệm vụ học trừu tượng. Đó là học có giám sát (có mẫu), học không giám sát và học tăng cường. Thông thường, loại kiến trúc mạng nào cũng có thể dùng được cho các nhiệm vụ.

*+ Học có giám sát****:*** Một thành phần không thể thiếu của phươngpháp này là sự có mặt của một người thầy (ở bênngoài hệ thống). Người thầy này có kiến thức vềmôi trường thể hiện qua một tập hợp các cặp đầuvào - đầu ra đã được biết trước. Hệ thống học (ởđây là mạng neural) sẽ phải tìm cách thay đổi cáctham số bên trong của mình (các trọng số và các ngưỡng) để tạo nên một ánh xạ có khả năng ánh xạ các đầu vào thành các đầu ra mong muốn. Sự thay đổi này được tiến hành nhờ việc so sánh giữa đầu ra thực sự và đầu ra mong muốn.

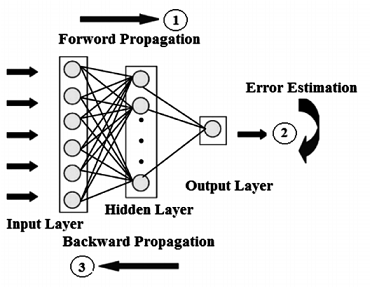
***+*** *Học không giám sát****:*** Trong học không có giám sát, ta được cho trướcmột số dữ liệu x và hàm chi phí cần được cực tiểuhóa có thể là một hàm bất kỳ của dữ liệu x và đầura của mạng, f – hàm chi phí được quyết định bởiphát biểu của bài toán. Phần lớn các ứng dụng nằmtrong vùng của các bài toán ước lượng như môhình hóa thống kê, nén, lọc, phân cụm.

*+ Học tăng cường:*Dữ liệu x thường không được tạo trước mà đượctạo ra trong quá trình một agent tương tác với môitrường. Tại mỗi thời điểm t, agent thực hiện hànhđộng yt và môi trường tạo một quan sát xt với mộtchi phí tức thời Ct, theo một quy trình động nào đó(thường là không được biết). Mục tiêu là một sáchlược lựa chọn hành động để cực tiểu hóa một chiphí dài hạn nào đó, nghĩa là chi phí tích lũy mongđợi. Quy trình hoạt động của môi trường và chiphí dài hạn cho mỗi sách lược thường không đượcbiết, nhưng có thể ước lượng được. Mạng neuralnhân tạo thường được dùng trong học tăng cườngnhư một phần của thuật toán toàn cục. Các bài toánthường được giải quyết bằng học tăng cường là cácbài toán điều khiển, trò chơi và các nhiệm vụ quyếtđịnh tuần tự (sequential decision making) khác.

### 2.2.3 Quá trình huấn luyện ANN

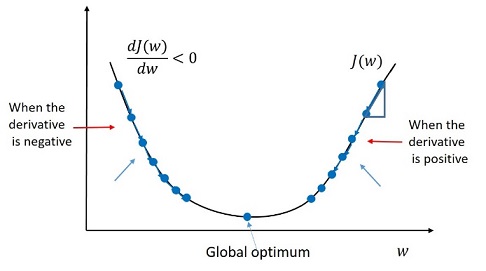
Quá trình huấn luyện (training) ANN được thực hiện thông qua cơ chế truyền ngược (backpropagation). Sai số giữa lớp đầu ra (output layer) và nhãn thực tế (groundtruth) sẽ được truyền ngược trở lại nhằm tinh chỉnh các trọng số (weight) giữa các kết nối.

Câu hỏi được đặt ra ở đây là điều chỉnh tăng giảm các trọng số (weight) như nào để giảm thiểu sai số ?



Hình 11 Quá trình huấn luyện ANN

Việc này sẽ được thực hiện nhờ phương pháp được gọi là suy giảm độ dốc (gradient descent). Sử dụng phương pháp đạo hàm và áp dụng nó vào các phương trình nhằm tìm ra điểm cực tiểu (minumum point) và điểm cực đại (maximum point), bây giờ sẽ sử dụng nó nhằm tìm hướng đi cho các trọng số (weight). Hãy tưởng tượng lúc này đang đi xuống một con dốc để xuống được đến chân dốc phần thấp nhất cũng tương ứng với điểm mà sai số thấp nhấp, suy giảm độ dốc (gradient descent) sử dụng đạo hàm (derivative) như một cây gậy chỉ đường, khi đạo hàm (derivative) âm, gia tăng trọng số (weight), khi đạo hàm (derivative) dương, chúng ta sẽ giảm trọng số (weight) lại, như mô tả hình dưới đây.



Hình 12 Ảnh minh hoạ

Trong hình trên có thể thấy các điểm màu xanh là các điểm dịch chuyển của sai số, khoảng cách giữa các điểm này được đặc trưng bởi tham số tốc độ học (learning rate). Tốc độ học (learning rate) càng lớn, các bước dịch chuyển càng dài hơn, dễ khiến cho sai số bật nhảy hỗn loạn, có thể bỏ qua mất điểm cực tiểu (minimum point). Ngược lại khi tốc độ học (learning rate) nhỏ, các bước dịch chuyển ngắn lại, lâu đạt đến được điểm cực tiểu, cũng như dễ bị kẹt lại ở các điểm cực tiểu cục bộ (local minimum), thay vì đạt đến điểm cực tiểu toàn cục (global minimum). Vì vậy việc lựa chọn một tốc độ học (learning rate) phù hợp là rất cần thiết, hoặc cần phải sử dụng phương pháp linh hoạt (adaptive) nhằm thay đổi tốc độ học (learning rate) theo từng giai đoạn.

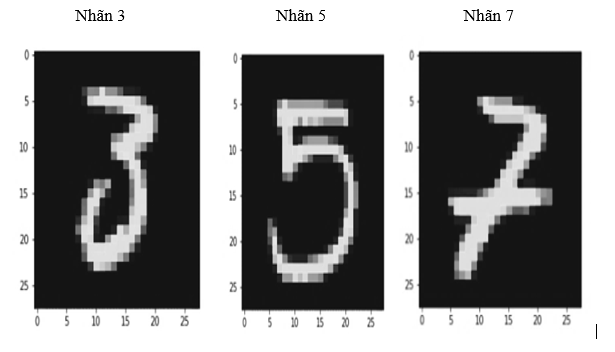
## 2.3 Phương pháp nghiên cứu

* **Nghiên cứu mô hình thực nghiệm nhận dạng chữ số viết tay với**

**Artificial Neural Network.**

Ở đây, em sử dụng mạng Artificial Neural Network và thuật toán lan truyền ngược sai số Back Propagation để xử lý bài toán. Mục đích là để kiểm chứng khả năng nhận dạng của mạng nơ-ron nhân tạo.

***Cơ sở dữ liệu***: Đồ án sử dụng tập dữ liệu MNIST [4] (Modified National Institute of Standards and Technology) để làm cơ sở dữ liệu đánh giá hệ thống. Bộ dữ liệu MNIST được chia thành hai phần: dữ liệu dành cho quá trình huấn luyện và dữ liệu dành cho quá trình kiểm tra. Dữ liệu huấn luyện gồm 50000 ảnh, dữ liệu kiểm tra gồm 10000 ảnh. Tất cả đều là hình ảnh đen trắng các chữ số viết tay từ 0 tới 9 có kích thước 28 pixel x 28 pixel đã được gắn nhãn đúng. Một số hình ảnh ngẫu nhiên được trích xuất từ bộ dữ liệu MNIST:



Hình 13 Một số ảnh đã được gắn nhãn tương ứng.

***Lựa chọn các thông số ban đầu:***Ngõ vào và ngõ ra của mạng sẽ là các thông số cố định của một mô hình mạng neuron nhân tạo. Ở bài toán nhận diện chữ số viết tay sử dụng bộ dữ liệu MNIST, mục đích của mạng sẽ nhận dạng được 10 chữ số khác nhau từ 0 đến 9, từ đó lớp ngõ ra cuối cùng cần 10 neuron thể hiện 10 chữ số khác nhau.

Với ảnh 2 chiều có kích thước 28 pixel x 28 pixel chuyển về dạng dữ liệu 1 chiều 784 pixel x 1 pixel cho mỗi hình ảnh. Ta xây dựng một mô hình mạng neuron với 784 neuron đầu vào tương ứng với 784 pixel. Với 10 giá trị ngõ ra (tương ứng với các nhãn 0, 1, 2, <, 9), sẽ có 10 neuron ở lớp ngõ ra. Nhận dạng chữ số viết tay dùng mạng neuron nhân tạo 128

Các thông số khác như: số neuron lớp ẩn, số lớp ẩn, tỷ lệ học, mini-batch size sẽ được làm rõ ở phần kết quả.

# CHƯƠNG 3: ĐÁNH GIÁ KẾT QUẢ

## 3.1 Mô tả công việc nghiên cứu

Mô hình thực nghiệm được lập trình với ngôn ngữ Python, framework được sử dụng là Numpy. Khởi tạo với mô hình mạng neuron ban đầu được tham khảo ở [5] gồm 3 lớp: lớp ngõ vào 784 neuron, 1 lớp ẩn có 30 neuron, lớp ngõ ra 10 neuron.

Thử nghiệm với 100 chu kỳ học, lúc này kết quả huấn luyện gần như đạt bão hòa trong 10 chu kỳ liên tiếp gần (mỗi chu kỳ học tương đương một lần quét qua hết  
tất cả ảnh huấn luyện), dữ liệu huấn luyện và dữ liệu kiểm tra được chia thành các tập con, mỗi tập con có 10 ảnh, tỷ lệ học = 0,03.

Kết quả thực nghiệm được tiến hành trên máy tính cá nhân có cấu hình: Intel(R) Core i3-4030U CPU@ 1.92GHz, RAM 8GB. Thời gian huấn luyện cho mô hình trên khoảng 10 phút, cho tỷ lệ nhận dạng đúng trên 95% ở trên dữ liệu kiểm tra.

Bên cạnh mô hình khởi tạo ban đầu, các mô hình với các thông số khác nhau sẽ được khảo sát để làm rõ chức năng của từng thông số trong mạng.

## 3.2 Kết quả nghiên cứu

* Kết quá thực nghiệm trên mô hình nhận diện chữ số viết tay bằng mạng ANN.Epoch 1/10, Tỉ lệ huấn luyện thất bại: 0.2850, Kiểm thử tỉ lệ thất bại: 0.1904

Epoch 2/10, Tỉ lệ huấn luyện thất bại: 0.2001, Kiểm thử tỉ lệ thất bại: 0.2230

Epoch 3/10, Tỉ lệ huấn luyện thất bại: 0.1897, Kiểm thử tỉ lệ thất bại: 0.2267

Epoch 4/10, Tỉ lệ huấn luyện thất bại: 0.1882, Kiểm thử tỉ lệ thất bại: 0.2149

Epoch 5/10, Tỉ lệ huấn luyện thất bại: 0.1659, Kiểm thử tỉ lệ thất bại: 0.2511

Epoch 6/10, Tỉ lệ huấn luyện thất bại: 0.1655, Kiểm thử tỉ lệ thất bại: 0.2086

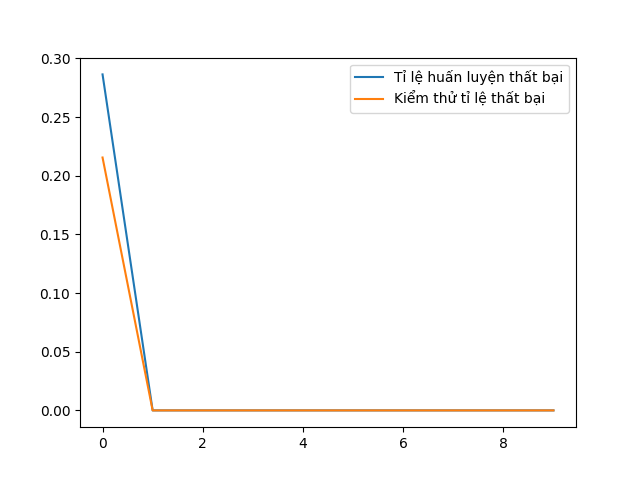
Epoch 7/10, Tỉ lệ huấn luyện thất bại: 0.1776, Kiểm thử tỉ lệ thất bại: 0.2929

Epoch 8/10, Tỉ lệ huấn luyện thất bại: 0.1661, Kiểm thử tỉ lệ thất bại: 0.2488

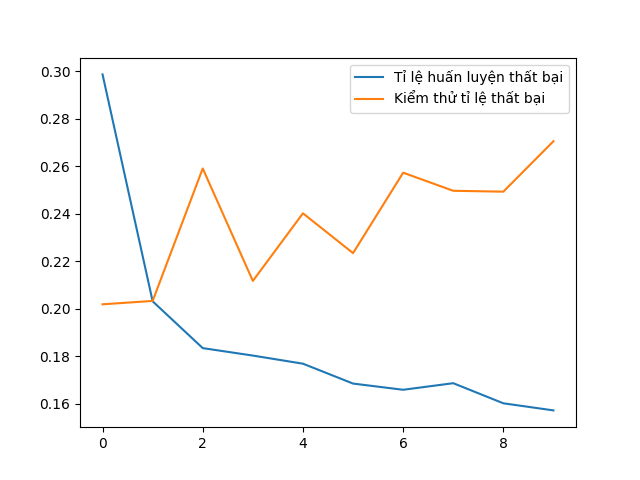
Epoch 9/10, Tỉ lệ huấn luyện thất bại: 0.1629, Kiểm thử tỉ lệ thất bại: 0.2771

Epoch 10/10, Tỉ lệ huấn luyện thất bại: 0.1456, Kiểm thử tỉ lệ thất bại: 0.2773

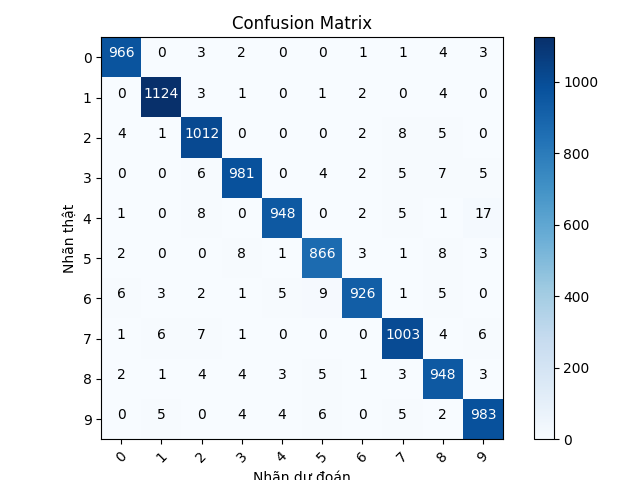
Độ chính xác của đào tạo: 0.9679, Kiểm tra độ chính xác: 0.9552



Hình 14 Huấn luyện lần 1



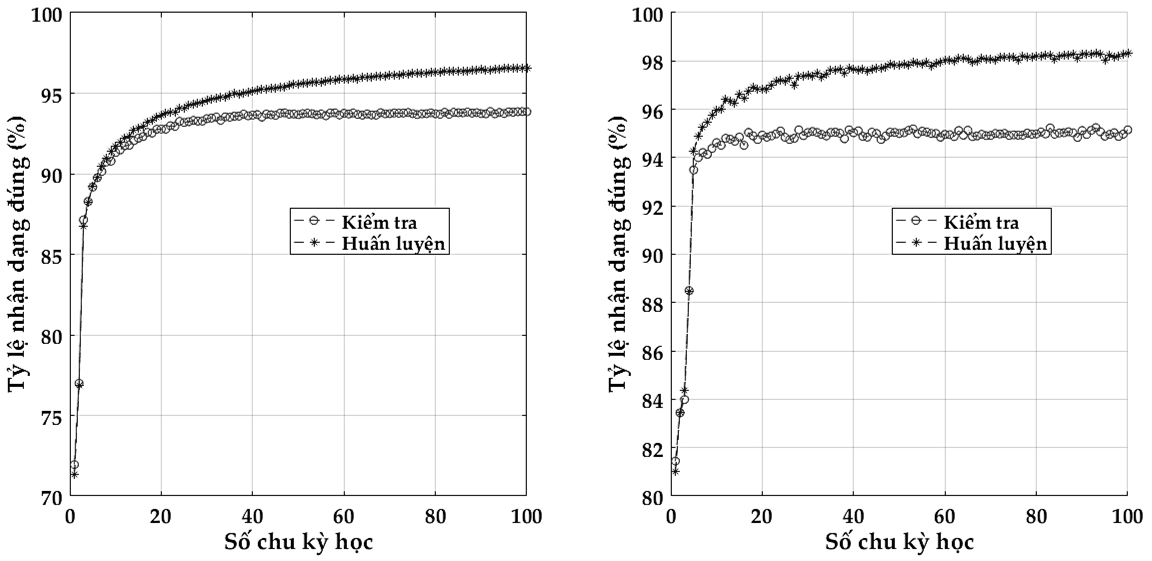
Hình 15 Huấn luyện lần 10



Hình 16 Ma trận hỗn loạn

* Tác động của tỷ lệ học – Learning Rate:

Việc lựa chọn tỷ lệ học (Learning Rate - LR) có vai trò quan trọng trong việcquyết định tốc độ học của mạng ANN. Tốc độ học càng nhanh sự thay đổitrọng số càng lớn, hàm mất mát càng nhanh hội tụ về giá trị nhỏ nhất.



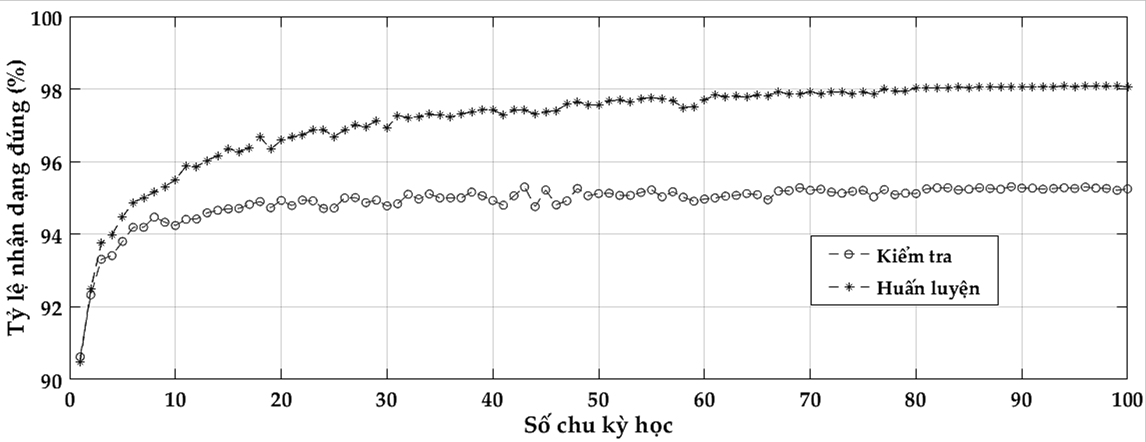
(a) (b)

Hình 17 Tỷ lệ nhận dạng với tỷ lệ học LR = 0,03 (a) và tỷ lệ học LR = 0,1 (b)

Hình 14 mô tả sự khác nhau khi chọn LR ở hai trường hợp cụ thể: mô hình (a) sử dụng LR = 0,03 nhỏ hơn so với mô hình (b) sử dụng LR = 0,1. Rõ ràng với một LR nhỏ tốc độ học của mạng khá chậm, sau 10 chu kỳ học vẫn chưa thể đạt ngưỡng bão hòa, tỷ lệ nhận dạng vẫn đang tiếp tục tăng nhẹ. Ở mô hình (b), LR lớn hơn mạng chỉ mất chưa đến 20 chu kỳ học đã đạt ngưỡng bão hòa. Tuy nhiên, LR lớn cho phép mạng học rất nhanh, chỉ với vài chu kỳ học đã đạt gần đến ngưỡng bão hòa nhưng nhược điểm là không ổn định, tỷ lệ nhận dạng dao động mạnh. Ngược lại ở LR thấp, tỷ lệ nhận dạng rất ổn định, tuy nhiên tốc độ học chậm.

Kết hợp ưu điểm riêng của từng loại, việc chọn LR lớn ở những chu kỳ học đầu nhằm tăng tốc độ học, chọn các LR thấp hơn ở những chu kỳ học sau sẽ giúp mạng học ổn định hơn.

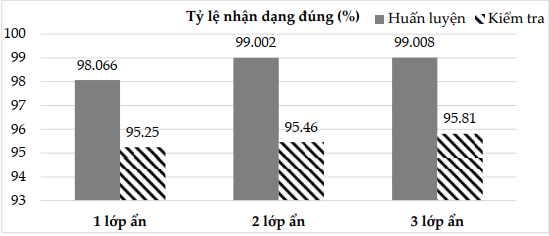
Cụ thể ở hình 15, ở chu kỳ học thứ 80, LR đã được giảm đi 10 lần so với các chu kỳ học trước, rõ ràng tỷ lệ nhận dạng của mạng đạt ngưỡng bão hòa và vẫn rất ổn định hơn so với các chu kỳ học đầu.



Hình 18 Kết quả mạng ANN khi kết hợp 2 giá trị LR khác nhau.

* Tác động của số lớp ẩn và số lượng neuron

Việc chọn lựa số lượng lớp ẩn trong mạng ANN hay số lượng nơ-ron trong các lớp ẩn vẫn chưa có một quy tắc cụ thể. Các thông số này quyết định độ phức tạp của mạng ANN hay độ khó của bài toán đặt ra. Thực nghiệm với mạng ANN được giữ nguyên các thông số, số lượng lớp ẩn lần lượt tăng lên 2 và 3 lớp ẩn cho kết quả nhận dạng như sau:

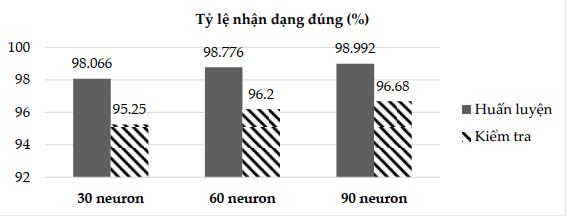


Hình 19 Tỷ lệ nhận dạng đúng với các số lượng lớp ẩn khác nhau

Khi tăng số lượng lớp ẩn, tỷ lệ nhận dạng đúng ở dữ liệu huấn luyện và kiểm  
tra đều tăng lên, tuy nhiên khi tăng số lớp ẩn độ chêch lệch giữa tỷ lệ nhận dạng ở tập kiểm tra so với tập huấn luyện được rút ngắn. Việc tăng số lớp ẩn làm cho mô hình mạng trở nên phức tạp kéo theo thời gian hoàn thành quá trình huấn luyện tăng lên.

Số lượng neuron trong lớp ẩn cũng quyết định đến tỷ lệ nhận dạng trong mạng ANN.

Số lượng neuron càng nhiều, độ phức tạp của mạng càng cao, số lượng trọng số càng nhiều. Hình 20 thể hiện kết quả khảo sát trên cùng mạng ANN có 1 lớp ẩn nhưng có số lượng neuron ở lớp ẩn tăng dần:



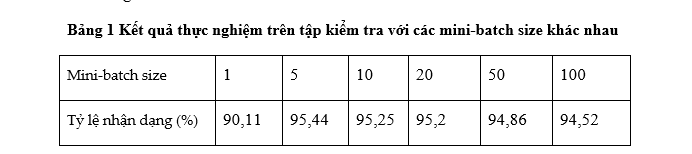
Hình 20 Khảo sát mạng ANN với số lượng neuron khác nhau.

Kết luận: Kết quả nhận được khi đánh giá trên tập kiểm tra (tương ứng với quá trình nhận dạng trên thực tế ). Nhận thấy rằng, khi tăng số lượng lớp ẩn hay số lượng nơ-ron trong mỗi lớp ẩn thì kết quả tỷ lệ nhận dạng cho mạng ANN đều tăng, tuy nhiên tăng số lượng neuron trong lớp ẩn sẽ có hiệu quả hơn so với tăng số lượng lớp ẩn trong mạng về tỷ lệ nhận dạng.

* Ảnh hưởng của mini-batch size

Mini-batch size được tạo ra bằng cách chia nhỏ tập dữ liệu huấn luyện và tập  
kiểm tra thành nhiều phần bằng nhau. Ví dụ tập huấn luyện MNIST có 50000 ảnh,  
chọn mini-batch size bằng 10 có nghĩa là chia tập huấn luyện thành 5000 tập con, mỗi tập con có 10 ảnh. Số lượng ảnh trong tập con quyết định có bao nhiêu ảnh dùng để huấn luyện cho một lần cập nhật weight, bias. Để hoàn thành một chu kỳ học, mạng ANN sẽ lần lượt chọn lần lượt các tập con để huấn luyện, sau mỗi tập con, các weight và bias sẽ được cập nhật cho đến khi hết số lượng các tập con.

Thực nghiệm với các mini-batch size khác nhau cho kết quả trên tập kiểm tra như hình 21:



Hình 21 Kết quả thực nghiệm

Mini-batch size lớn sẽ mở rộng cơ sở dữ liệu cho quá trình học, song song với đó, việc chọn các mini-batch size lớn cũng yêu cầu bộ nhớ lớn hơn cho việc tính toán.  
 Qua đánh giá thực nghiệm, ta thấy được rằng với các mô hình nhỏ vị đơn giản, cập nhật các thông số nhiều lần qua các mini-batch size nhỏ cho kết quả tốt hơn các minibatch size lớn.

Ngoài ra, ứng dụng với batch size nhỏ cũng phù hợp là bài toán nhận dạng thực tiễn, bộ não phải đưa ra quyết định sau khi học với một số lượng mẫu khá bé, do đó kết quả tối ưu chính là hướng đến chính là tăng tỷ lệ nhận dạng đúng với số lượng mẫu bé.

## Ưu điểm, nhược điểm của Artificial Neural Network.

* Ưu điểm:

- Dựa vào nghiên cứu trên ta thấy ANN nhận dạng chữ số với độ chính xác lên đến 95%.

**- Khả năng làm việc với kiến ​​thức chưa đầy đủ:** Sau khi đào tạo ANN, thông tin có thể tạo ra đầu ra ngay cả với dữ liệu không đủ. Việc mất hiệu suất ở đây phụ thuộc vào tầm quan trọng của việc thiếu dữ liệu.

**- Có phân phối bộ nhớ:** Đối với ANN là để có thể thích ứng, điều quan trọng là phải xác định các ví dụ và khuyến khích mạng theo đầu ra mong muốn bằng cách trình diễn các ví dụ này cho mạng. Sự kế thừa của mạng tỷ lệ thuận với các trường hợp đã chọn và nếu sự kiện không thể xuất hiện trên mạng theo tất cả các khía cạnh của nó, nó có thể tạo ra kết quả sai.

* Nhược điểm:

- Cần nhiều dữ liệu: Không giống như bộ não con người, có thể học cách làm mọi thứ với rất ít ví dụ, mạng nơ-ron cần hàng nghìn và hàng triệu ví dụ.

- Khả năng tổng quát hóa kém**:** Một mạng nơ-ron sẽ thực hiện chính xác một nhiệm vụ mà nó đã được huấn luyện, nhưng rất kém ở bất kỳ nhiệm vụ nào khác, ngay cả khi nó tương tự như vấn đề ban đầu.

- Mạng nơ-ron không rõ ràng: Vì mạng nơ-ron thể hiện hành vi của chúng về trọng lượng và kích hoạt nơ-ron nên rất khó xác định logic đằng sau các quyết định của chúng.

**- Khó khăn khi hiển thị sự cố với mạng:** ANN có thể hoạt động với dữ liệu số. Các vấn đề phải được chuyển đổi thành các giá trị số trước khi được đưa vào ANN. Cơ chế trình bày được giải quyết ở đây sẽ tác động trực tiếp đến hiệu suất của mạng. Nó phụ thuộc vào khả năng của người dùng.

# CHƯƠNG 4: KẾT LUẬN

Nghiên cứu trên đã trình bày những thuật ngữ cơ bản, cấu tạo mạng Artificial Neural Network cũng như cách mà mạng nơ-ron nhân tạo “học” các thông số để đánh giá cho bài toán nhận dạng chữ số viết tay. Tuy nhiên, quy mô bài toán còn nhỏ, việc lựa chọn các thông số vẫn còn là phương pháp thực nghiệm, một cách thủ công, mang tính chất tham khảo. Với các bài toán thực tiễn, xử lý ảnh dùng mạng neuron tích chập (Convolution Neural Network) sẽ có nhiều ưu điểm hơn và được sử dụng rộng rãi thay cho mạng Artificial Neural Network (ANN).

# CHƯƠNG 5: HƯỚNG PHÁT TRIỂN

Đưa Artificial Neural Network trở thành hệ thống nơ-ron nhân tạo cấp độ con người, giảm số lần huấn luyện Artificial Neural Network nhưng độ chính xác tăng lên.

Có một số nỗ lực để vượt qua giới hạn của mạng nơ-ron nhân tạo, chẳng hạn như một [sáng kiến ​​do DARPA[6] tài trợ nhằm tạo ra các mô hình AI có thể giải thích được](https://bdtechtalks.com/2019/01/10/darpa-xai-explainable-artificial-intelligence/) . Những phát triển thú vị khác bao gồm phát triển các mô hình lai kết hợp mạng nơ-ron nhân tạo và AI dựa trên quy tắc để tạo ra các hệ thống AI có thể diễn giải được và yêu cầu ít dữ liệu đào tạo hơn.

Mặc dù chúng ta vẫn còn một chặng đường dài trước khi đạt được mục tiêu về AI cấp độ con người ( nếu chúng ta sẽ đạt được nó ), thì mạng thần kinh nhân tạo ANN là lựa chọn đưa chúng ta đến gần hơn nhiều.

# DANH MỤC TÀI LIỆU THAM KHẢO

[1] Thạc sĩ Trần Văn Nam (trường Đại học Trà Vinh), nhận dạng kí tự bằng mạng neural network ANN, tạp chí Khoa học và Công nghệ số 14, tháng 6/2014, T30-T34.

[2] Yann LeCun, Courant Institute (1989). *The MNIST Database of Handwritten Digits.*

[3] Dave Anderson, George McNeill (2006), Artificial Neural Networks Technology, Prepared for Rome Laboratory RL/C3C Griffiss AFB, NY 13441-5700, USA.

[4] Khái niệm học chuyển tiếp, Website: *https://bdtechtalks.com/2019/06/10/what-is-transfer-learning/*

[5] Michael A. Nielsen (2015), Neural Networks and Deep Learning, Online book:  
http://neuralnetworksanddeeplearning.com/

[6] Sáng kiến ​​của DARPA Website: https://bdtechtalks.com/2019/01/10/darpa-xai

explainable artificial-intelligence/

# PHỤ LỤC

* **Code để thực hiện quá trình training**

**# Import tập dữ liệu cần dùng**

import torch

import torch.nn as nn

import torchvision

import torchvision.transforms as transforms

import numpy as np

import matplotlib.pyplot as plt

**# Tải dữ liệu từ mnist**

*# root là nơi tải dữ liệu xuống*

*# train = true, trả về tập dữ liệu về datasets*

*# biến đổi thực hiện một số xử lý trước hữu ích*

train\_dataset = torchvision.datasets.MNIST(root='.', train=True, transform=

transforms.ToTensor(), download=True)

train\_dataset.data

train\_dataset.data.max() # tensor(255, dtype=torch.uint8)

train\_dataset.data.shape # torch.Size([60000, 28, 28])

train\_dataset.targets # between 0 and 9

learning\_rate = 0.03 # Tỉ lệ học

***# Tập dữ liệu kiểm tra***

test\_dataset = torchvision.datasets.MNIST(root='.', train=False,

transform=transforms.ToTensor(), download=True)

**# Xây dựng mô hình ANN**

model = nn.Sequential(nn.Linear(784, 128), nn.ReLU(), nn.Linear(128, 10))

*# không cần đến softmax cuối cùng!*

device = torch.device("cude:0" if torch.cuda.is\_available() else "cpu")

print(device)

model.to(device)

*# Thất bại và tối ưu hóa*

criterion = nn.CrossEntropyLoss()

optimizer = torch.optim.Adam(model.parameters(),lr=learning\_rate)

*# Tải dữ liệu*

*# Hữu ích vì nó tự động tạo các lô trong vòng huấn luyện*

*# và quan tâm đến việc xáo trộn*

batch\_size = 128

*# xáo trộn dữ liệu đào tạo, nhưng không trộn dữ liệu kiểm tra*

*# Em không muốn có sự tương quan giữa các dữ liệu*

*# không cần xáo trộn dữ liệu thử nghiệm*

train\_loader = torch.utils.data.DataLoader(dataset=train\_dataset,batch\_size

=batch\_size, shuffle=True)

test\_loader = torch.utils.data.DataLoader(dataset=test\_dataset, batch\_size=

batch\_size, shuffle=False)

*# Kiểm tra những gì trình tải dữ liệu làm*

*# ánh xạ các giá trị thành (0, 1)*

*# tạo dữ liệu về hình dạng (kích thước lô, màu sắc, chiều cao, chiều rộng)*

tmp\_loader = torch.utils.data.DataLoader(dataset=train\_dataset, batch\_size=1,

shuffle=True)

for x,y in tmp\_loader:

print(x)

print(x.shape)

print(y.shape)

break

train\_dataset.transform(train\_dataset.data.numpy()).max()

**# Số lần huấn luyện mô hình**

n\_epochs = 10

**# Nội dung cần lưu trữ**

train\_losses = np.zeros(n\_epochs)

test\_losses = np.zeros(n\_epochs)

for it in range(n\_epochs):

train\_loss = []

for inputs,targets in train\_loader:

*# Di chuyển dữ liệu sang GPU nếu thiết bị là GPU*

*# if device == 'cpu':*

inputs, targets = inputs.to(device), targets.to(device)

*# Định hình lại đầu vào -1 nghĩa là chỉ định bất kỳ giá trị nào phù hợp*

inputs = inputs.view(-1, 784)

*# Zero the parameter gradients*

optimizer.zero\_grad()

*# Forward pass*

outputs = model(inputs)

loss = criterion(outputs, targets)

*# Backward and Optimize*

loss.backward()

optimizer.step()

train\_loss.append(loss.item())

***# Nhận tỉ lệ thất bại huấn luyện và kiểm tra tỉ lệ thất bại***

train\_loss = np.mean(train\_loss)

test\_loss = []

for inputs, targets in test\_loader:

inputs, targets = inputs.to(device), targets.to(device)

inputs = inputs.view(-1, 784)

outputs = model(inputs)

loss = criterion(outputs, targets)

test\_loss.append(loss.item())

test\_loss = np.mean(test\_loss)

# Save losses

train\_losses[it] = train\_loss

test\_losses[it] = test\_loss

print(f'Epoch {it+1}/{n\_epochs}, Tỉ lệ huấn luyện thất bại: {train\_loss:.4f}, Kiểm thử tỉ lệ thất bại: {test\_loss:.4f}')

# Vẽ biểu đồ tỉ lệ thất bại

plt.plot(train\_losses, label='Tỉ lệ huấn luyện thất bại')

plt.plot(test\_losses, label='Kiểm thử tỉ lệ thất bại ')

plt.legend()

plt.show()

# Tính độ chính xác

# độ chính xác của huấn luyện

n\_correct = 0

n\_total = 0

for inputs, targets in train\_loader:

# di chuyển dữ liệu sang GPU

inputs, targets = inputs.to(device), targets.to(device)

# định hình lại đầu vào

inputs = inputs.view(-1, 784)

# chuyển tiếp qua

outputs = model(inputs)

# nhận dự đoán

# torch.max trả về cả max và argmax

\_, predictions = torch.max(outputs, 1)

# cập nhật số lượng

n\_correct += (predictions == targets).sum().item()

n\_total += targets.shape[0]

train\_acc = n\_correct / n\_total

# test accuracy

n\_correct = 0

n\_total = 0

for inputs, targets in test\_loader:

# duy chuyển dữ liệu vào GPU

inputs, targets = inputs.to(device), targets.to(device)

# định hình lại đầu vào

inputs = inputs.view(-1, 784)

# chuyển tiếp qua

outputs = model(inputs)

# nhận dự đoán

# torch.max trả về cả max và argmax

\_, predictions = torch.max(outputs, 1)

# cập nhật số lượng

n\_correct += (predictions == targets).sum().item()

n\_total += targets.shape[0]

test\_acc = n\_correct / n\_total

print(f'Độ chính xác của đào tạo: {train\_acc:.4f}, Kiểm tra độ chính xác: {test\_acc:.4f}')

#Ma trận hỗn loạn

from sklearn.metrics import confusion\_matrix

import numpy as np

import itertools

def plot\_confusion\_matrix(cm, classes, normalize=False, title='Confusion Matrix', cmap=plt.cm.Blues):

"""

The function prints and plots the confusion matrix.

Normalization can be applied by setting 'normalize=True'

"""

if normalize:

cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

print("Chuẩn hoá ma trận hỗn loạn")

else:

print('Ma trận hỗn loạn, không chuẩn hóa')

print(cm)

plt.imshow(cm, interpolation='nearest', cmap=cmap)

plt.title(title)

plt.colorbar()

tick\_marks = np.arange(len(classes))

plt.xticks(tick\_marks, classes, rotation=45)

plt.yticks(tick\_marks, classes)

fmt = '.2f' if normalize else 'd'

thresh = cm.max() / 2

for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):

plt.text(j, i, format(cm[i, j], fmt), horizontalalignment="center", color="white" if cm[i, j] > thresh else "black")

plt.tight\_layout()

plt.ylabel('Nhãn thật')

plt.xlabel('Nhãn dự đoán')

plt.show()

# get all predictions in an array and plot confusion matrix

x\_test = test\_dataset.data.numpy()

y\_test = test\_dataset.targets.numpy()

p\_test = np.array([])

for inputs, targets in test\_loader:

# move data to GPU

inputs, targets = inputs.to(device), targets.to(device)

# reshape the input

inputs = inputs.view(-1, 784)

# forward pass

outputs = model(inputs)

# get the prediction

#torch.max returns both max and argmax

\_, predictions = torch.max(outputs, 1)

# update p\_test

p\_test = np.concatenate((p\_test, predictions.cpu().numpy()))

cm = confusion\_matrix(y\_test, p\_test)

plot\_confusion\_matrix(cm, list(range(10)))

**# Thử hiển thi một số ví dụ đúng**

print("-------------------------------------------------------------")

countss = 1

while (countss < 7):

print ('Ảnh dự đoán đúng:', countss)

countss = countss + 1

misclassified\_idx = np.where(p\_test == y\_test)[0]

i = np.random.choice(misclassified\_idx)

plt.imshow(x\_test[i], cmap='gray')

plt.title("Giá trị thật: %s Dự đoán: %s" % (y\_test[i], int(p\_test[i])))

plt.show()

**# Thử hiển thị một số ví dụ phân loại sai**

print("-------------------------------------------------------------")

counts = 1

while (counts <= 3):

print ('Ảnh dự đoán sai:', counts)

counts = counts + 1

misclassified\_idx = np.where(p\_test != y\_test)[0]

i = np.random.choice(misclassified\_idx)

plt.imshow(x\_test[i], cmap='gray')

plt.title("Giá trị thật: %s Dự đoán: %s" % (y\_test[i], int(p\_test[i])))

plt.show()