BỘ CÔNG THƯƠNG

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI

BÁO CÁO THÍ NGHIỆM/ THỰC NGHIỆM
HỌC PHẦN: TRÍ TUỆ NHÂN TẠO

ĐỀ TÀI: Ứng dụng mạng CNN vào việc nhận diện chữ số viết tay

Giảng viên	**: Nguyễn Lan Anh**
Lớp	**: 20221IT6043008**
Nhóm	**: 10**
Sinh viên thực hiện	**:** Nguyễn Thị Bích_2020608404
	Khổng Thị Thùy Linh_2020604488
	Lê Đắc Đạt_2020605449

HÀ NỘI_2022

MỤC LỤC

LỜI NÓI ĐẦU

Lời đầu tiên, chúng em xin gửi lời cảm ơn sâu sắc đến cô Nguyễn Lan Anh. Trong quá trình tìm hiểu và học tập bộ môn..., chúng em đã nhận được sự giảng dạy và hướng dẫn rất tận tình, tâm huyết của cô. Cô đã giúp chúng em tích lũy thêm nhiều kiến thức hay và bổ ích. Từ những kiến thức mà cô truyền đạt, chúng em xin trình bày lại những gì mình đã tìm hiểu về đề tài:

"Ứng dụng mạng CNN vào việc nhận diện chữ số viết tay".

Tuy nhiên, kiến thức về bộ môn Trí tuệ nhân tạo của chúng em vẫn còn những hạn chế nhất định. Do đó, không tránh khỏi những thiếu sót trong quá trình hoàn thành bài tập lớn này. Mong cô xem và góp ý để bài tiểu luận của em được hoàn thiện hơn.

Kính chúc cô thành công hơn nữa trong sự nghiệp "trồng người" và cô luôn dồi dào sức khỏe để tiếp tục dìu dắt nhiều thế hệ học trò đến những bến bờ tri thức.

Chúng em xin chân thành cảm ơn!

MỞ ĐẦU

1. Tên đề tài

"Ứng dụng mạng CNN vào việc nhận diện chữ số viết tay"

2. Lý do chọn đề tài

- Ngày nay các kĩ thuật về mạng noron đã được nghiên cứu và phát triển rộng rãi.Ứng dụng của CNN vào các vấn đề như nhận diện đối tượng , nhận dạng tín hiệu ,nhận dạng tiếng nói hay nhận dạng chữ viết tay nói chung và nhận dạng chữ số viết tay nói riêng là một vấn đề thách thức đối với những nhà nghiên cứu .Chữ số viết tay xuất hiện ở hầu hết các công việc trong cơ quan , doanh nghiệp,trường học,…Trong các trường phổ thông hiện nay , có một bộ phận quản lí điểm 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 .Để nhận dạng chữ số viết tay có nhiều phương pháp và kĩ thuật khác nhau như :logic mờ, giải thuật di truyền , mô hình xác suất thống kê, mô hình mạng noron.Vì vậy xuất phát từ nhu cầu thực tiễn ,chúng em chọn đề tài : "Ứng dụng mạng CNN vào nhận diện chữ số viết tay".

3. Mục đích đề tài

Đề tài *"Ứng dụng mạng CNN vào nhận diện chữ số viết tay".*

- Nhận diện được chữ số viết tay hay không.
- Từ đó có thể thông kế và đưa ra đâu là chữ số viết tay, đâu là chữ số viết thường.

Chương I: Tổng Quan Về Trí Tuệ Nhân Tạo

1.1. TRÍ TUỆ NHÂN TẠO LÀ GÌ?

- *AI - Artificial Intelligence* hay còn gọi là *Trí tuệ nhân tạo* là một ngành khoa học, kỹ thuật chế tạo máy móc thông minh, đặc biệt là các chương trình máy tính thông minh.
- AI được thực hiện bằng cách nghiên cứu cách suy nghĩ của con người, cách con người học hỏi, quyết định và làm việc trong khi giải quyết một vấn đề nào đó, và sử dụng những kết quả nghiên cứu này như một nền tảng để phát triển các phần mềm và hệ thống thông minh, từ đó áp dụng vào các mục đích khác nhau trong cuộc sống. Nói một cách dễ hiểu thì AI là việc sử dụng, phân tích các dữ liệu đầu vào nhằm đưa ra sự dự đoán rồi đi đến quyết định cuối cùng.

1.2. MỤC ĐÍCH CỦA TRÍ TUỆ NHÂN TẠO

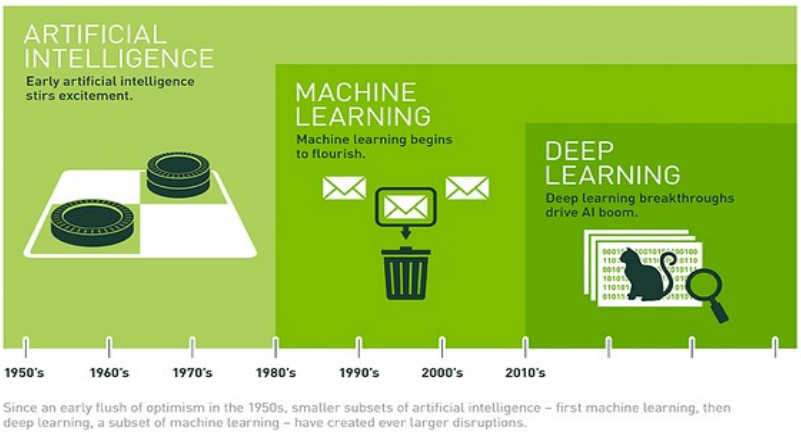
- Tạo ra các hệ thống chuyên gia - là các ứng dụng máy tính được phát triển để giải quyết các vấn đề phức tạp trong một lĩnh vực cụ thể, ở mức độ thông minh và chuyên môn của con người.
- Thực hiện trí thông minh của con người trong máy móc - Tạo ra các hệ thống có thể hiểu, suy nghĩ, học hỏi và hành xử như con người.

1.3. MỘT SỐ ỨNG DỤNG TRÍ TUỆ NHÂN TẠO

- Quản trị: Các hệ thống AI trợ giúp các công việc hành chính hàng ngày, để giảm thiểu lỗi của con người và tối đa hóa hiệu quả.
- Điều trị từ xa: Đối với các tình huống không khẩn cấp, bệnh nhân có thể liên hệ với hệ thống AI của bệnh viện để phân tích các triệu chứng của họ, nhập các dấu hiệu quan trọng của họ và đánh giá xem có cần phải chăm sóc y tế hay không. Điều này làm giảm khối lượng công việc của các chuyên gia y tế bằng cách chỉ đưa các trường hợp quan trọng đến họ.
- Hỗ trợ chuẩn đoán: Thông qua thị giác máy tính và mạng lưới thần kinh tích chập, AI hiện có khả năng đọc quét hình ảnh cộng hưởng từ để kiểm tra khối u và sự phát triển ác tính khác của nó, với tốc độ nhanh hơn so với các bác sĩ x-quang và sai số thấp hơn đáng kể.
- Phẫu thuật có sự trợ giúp của robot: Robot phẫu thuật có sai số rất nhỏ và có thể thực hiện phẫu thuật suốt ngày đêm mà không bị kiệt sức.

- Giám sát các chỉ số quan trọng. Ngoài ra còn rất nhiều những ứng dụng trong các lĩnh vực khác trong đời sống như nhận diện khuôn mặt, nhận diện giọng nói, ô tô tự lái…

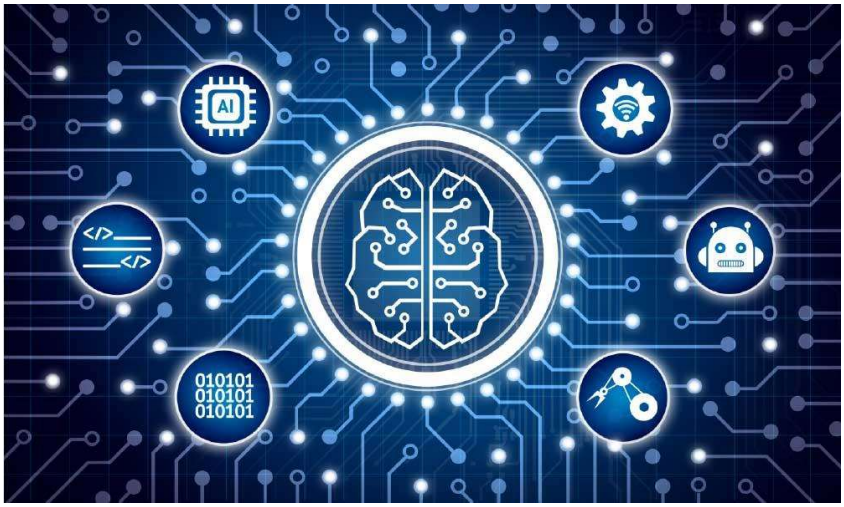
1.4. PHÂN BIỆT AI, MACHINE LEARNING, DEEP LEARNING



Hình 1.1: Phân biệt AI, Machine Learning và Deep Learning

1.4.1. AI (TRÍ TUỆ NHÂN TẠO)

- Trí tuệ nhân tạo là trí tuệ máy móc được tạo ra bởi con người. Trí tuệ này có thể tư duy, suy nghĩ, học hỏi, … như con người. Xử lý dữ liệu ở mức độ rộng hơn, quy mô hơn, hệ thống, khoa học và nhanh hơn so với con người.

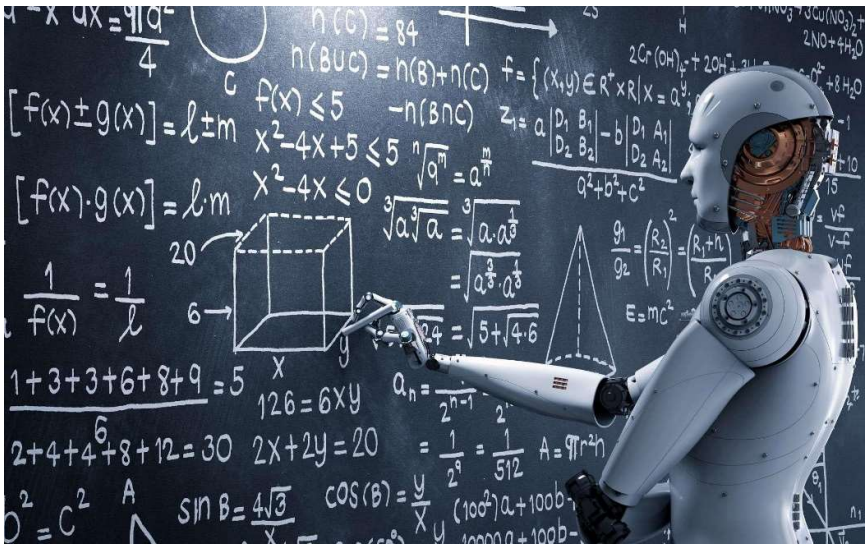


Hình 1.2: AI

- AI có ba mức độ khác nhau:

- Narrow AI: Trí tuệ nhân tạo được cho là hẹp khi máy có thể thực hiện một nhiệm vụ cụ thể tốt hơn so với con người. Nghiên cứu hiện tại về AI hiện đang ở cấp độ này.
- General AI: Trí tuệ nhân tạo đạt đến trạng thái chung khi nó có thể thực hiện bất kỳ nhiệm vụ sử dụng trí tuệ nào có cùng độ chính xác như con người.
- Strong AI: AI rất mạnh khi nó có thể đánh bại con người trong nhiều nhiệm vụ cụ thể.

1.4.2. MACHINE LEARNING (HỌC MÁY)



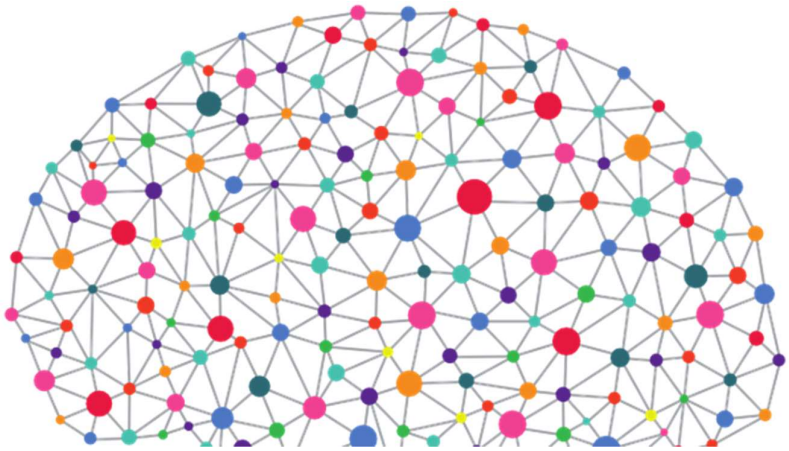
Hình 1.3: Machine Learning

- Machine Learning còn được gọi là học máy. Bạn có thể viết ứng dụng có AI mà không sử dụng học máy, nhưng bạn phải viết cả triệu triệu dòng code để xây dựng các trường hợp xảy ra.

- Học máy là cách để có được AI, máy tự học mà không cần phải code nhiều như không có học máy. Nói cách khác, nếu AI là mục tiêu thì học máy là phương tiện để đạt được mục tiêu đó.

- Máy sẽ được "học" bằng cách train nó với một lượng dữ liệu khổng lồ với một thuật toán, thuật toán có khả năng điều chỉnh và xây dựng model. Tuy nhiên, nếu như trong training dữ liệu có ngôn ngữ khác trong thực tế (tiếng Việt thay vì tiếng Anh…) thì rất có thể máy sẽ dự báo không chính xác nữa.

1.4.3. DEEP LEARNING (HỌC SÂU)



Hình 1.4: Deep Learning

- Deep Learning được bắt nguồn từ thuật toán Neural network của AI, là một ngành nhỏ của Machine Learning.
- Deep learning tập trung giải quyết các vấn đề liên quan đến mạng thần kinh nhân tạo nhằm nâng cấp các công nghệ như nhận diện giọng nói, tầm nhìn máy tính và xử lý ngôn ngữ tự nhiên.
- Trí tuệ nhân tạo có thể được hiểu đơn giản là được cấu thành từ các lớp xếp chồng lên nhau, trong đó mạng thần kinh nhân tạo nằm ở dưới đáy, Machine learning nằm ở tầng tiếp theo và Deep learning nằm ở tầng trên cùng.

Chương II: Cơ Sở Lý Thuyết

2.1. GIỚI THIỆU CƠ BẢN VỀ NEURAL NETWORKS

2.1.1. NEURAL NETWORKS LÀ GÌ?

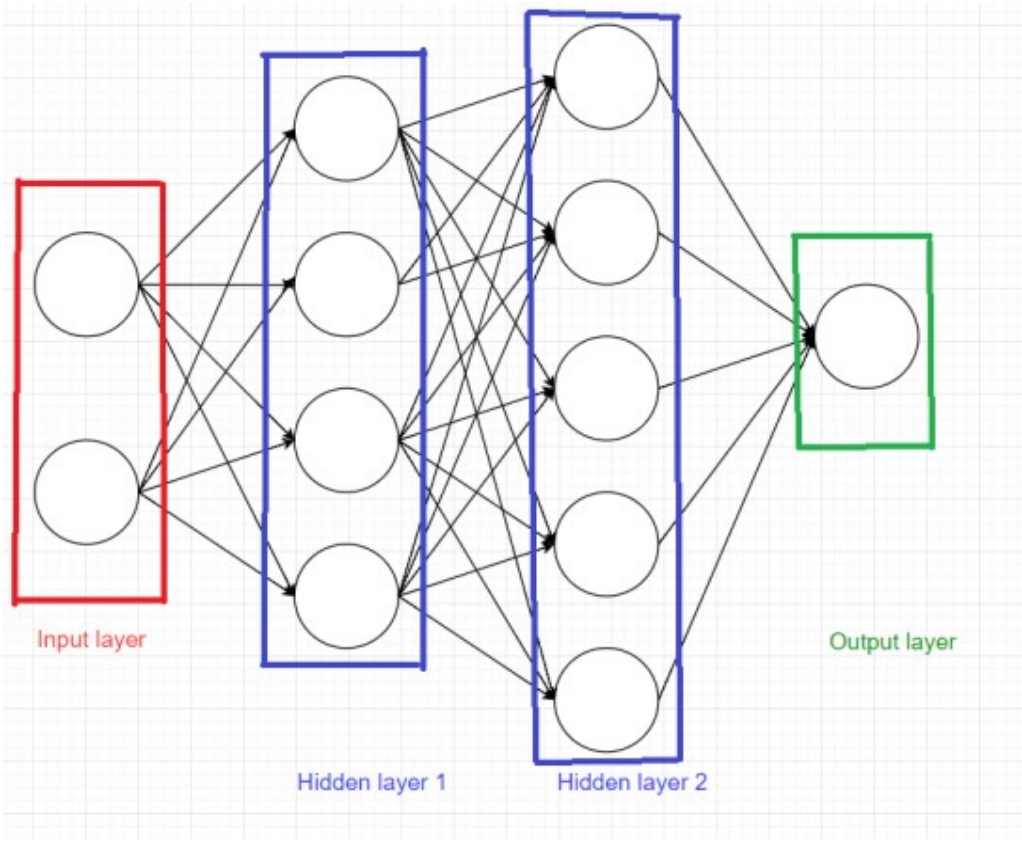
- Neural Network đọc tiếng việt là Mạng nơ-ron nhân tạo, đây là một chuỗi những thuật toán được đưa ra để tìm kiếm các mối quan hệ cơ bản trong tập hợp các dữ liệu. Thông qua việc bắt bước cách thức hoạt động từ não bộ con người. Nói cách khác, mạng nơ ron nhân tạo được xem là hệ thống của các tế bào 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, đang nhanh chóng trở nên phổ biến hơn trong sự phát triển của những hệ thống giao dịch điện tử.
- 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.
- Neural Network có sự tương đồng chuẩn mạnh vối những phương pháp thống kê như đồ thị đường cong và phân tích hồi quy. Neural Network 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.

2.1.2. MÔ HÌNH CỦA MỘT NEURON

- Layer đầu tiên là input layer, các layer ở giữa được gọi là hidden layer, layer cuối cùng được gọi là output layer. Các hình tròn được gọi là node.
- Mỗi mô hình luôn có 1 input layer, 1 output layer, có thể có hoặc không các hidden layer. Tổng số layer trong mô hình được quy ước là số layer – 1 (Không tính input layer).

- Ví dụ như ở hình trên có 1 input layer, 2 hidden layer và 1 output layer. Số lượng layer của mô hình là 3 layer.

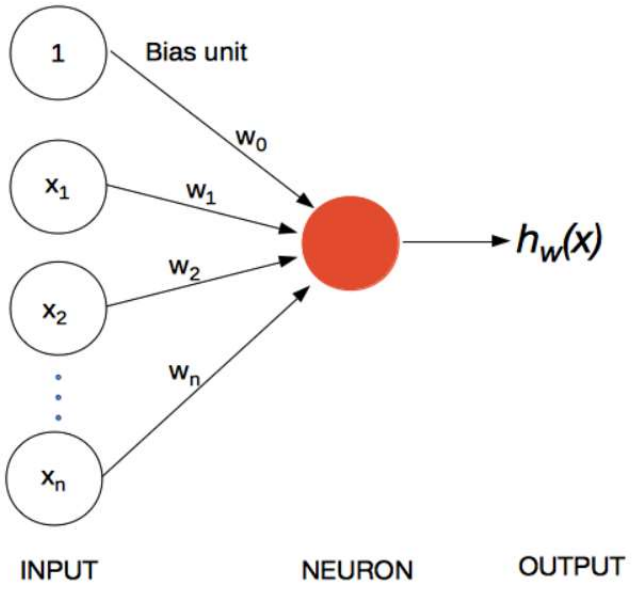
- Liên kết với tất cả các node ở layer trước đó với các hệ số w riêng.
- Mỗi node có 1 hệ số bias b riêng.
- Diễn ra 2 bước: tính tổng linear và áp dụng activation function.



Hình 2.1: Mô hình tổng quát

2.1.3. MẠNG PERCEPTRON

- Đây là mạng thần kinh lâu đời nhất, do Frank Rosenblatt tạo ra vào năm 1958. Nó có một lớp nơ-ron duy nhất và là dạng mạng nơ-ron đơn giản nhất.



Hình 2.2: Perceptron – mô hình đơn giản của một artificial neuron (neuron nhân tạo) trong DL

2.1.4. ỨNG DỤNG CỦA NEURAL NETWORKS

- Với số lượng các ứng dụng được triển khai ngày càng lớn, Neural Network, máy học và trí tuệ nhân tạo được ứng dụng trong nhiều công việc khác nhau như:

- Nhận dạng chữ viết tay: Mạng nơron nhân tạo Neural Network được sử dụng với mục đích chuyển đổi các ký tự viết tay thành các ký tự kỹ thuật số để máy tính dễ dàng nhận ra được các ký tự đó.
- Dự đoán các giao dịch chứng khoán: Việc theo dõi sàn giao dịch chứng khoán thường khó khăn và khó hiểu do có nhiều yếu tố ảnh hưởng đến thị trường này. Neural Network ra đời có thể kiểm tra được các yếu tố nói trên và dự đoán giá chứng khoán hàng ngày. Điều này mang đến nhiều lợi ích cho các nhà môi giới chứng khoán.
- Vấn đề đi lại của các nhân viên bán hàng: Neural Network giúp giải quyết và tìm ra con đường tối ưu cho các nhân viên bán hàng trong việc đi lại giữa các thành phố thuộc cùng một khu vực cụ thể đồng thời mang đến doanh thu cao và giảm thiểu chi phí.
- Nén hình ảnh: Ý tưởng đằng sau của Neural Network nén dữ liệu là lưu trữ, mã hoá và tái tạo các hình ảnh mang tính thực tế. Bằng cách sử dụng Neural

Network nén hình ảnh, bạn có thể dễ dàng tối ưu hoá kích thước dữ liệu của mình. Điều này sẽ giúp bộ nhớ của bạn tiết kiệm được dung lượng đáng để.

- Ngoài ra, Neural Network cũng được sử dụng nhiều trong nhiều ứng dụng và công nghệ khác nhau như thị giác máy tính, trò chơi điện tử, dịch tự động, lọc mạng xã hội, nhận dạng giọng nói và chẩn đoán y tế. Điều đáng ngạc nhiên hơn là Neural Network cũng được sử dụng nhiều trong các hoạt động truyền thống và sáng tạo như nghệ thuật hay hội hoạ.

2.2. CONVOLUTIONAL NEURAL NETWORKS (CNN)

2.2.1. ĐỊNH NGHĨA CNN LÀ GÌ?

a) Convolutinal

- Đây là một loại cửa sổ dạng trượt nằm trên một ma trận. Những convolutional layer sẽ có các parameter được học để điều chỉnh và lấy ra những thông tin chính xác nhất mà không cần phải chọn feature. Convolution hay tích chập chính là nhân các phần tử trong ma trận. Sliding Window còn được gọi là kernel, filter hoặc feature detect và là loại ma trận có kích thước nhỏ.

Hình 2.3: Mạng CNN

b) feature

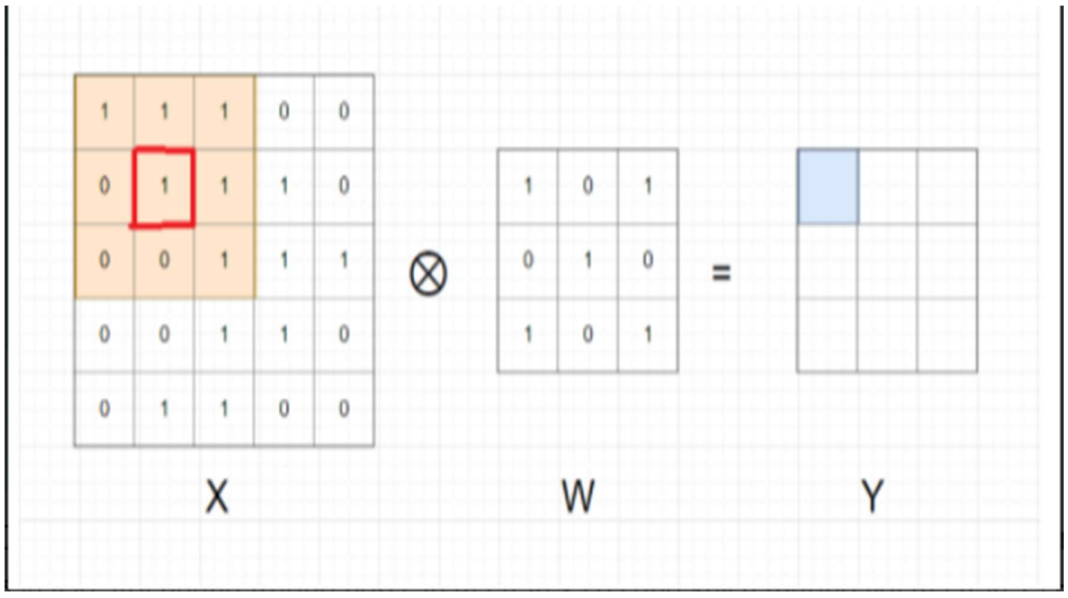
- Feature là đặc điểm, các CNN sẽ so sánh hình ảnh dựa theo từng mảnh và những mảnh này được gọi là Feature. Thay vì phải khớp các bức ảnh lại với nhau thì CNN sẽ nhìn ra sự tương động khi tìm kiếm thô các Feature khớp với nhau bằng 2 hình ảnh tốt hơn. Mỗi Feature được xem là một hình ảnh mini có nghĩa chúng là những mảng 2 chiều nhỏ. Các Feature này đều tương ứng với các khía cạnh nào đó của hình ảnh và chúng có thể khớp lại với nhau.

2.2.2. CÁC LỚP CƠ BẢN CỦA MỘT CONVOLUTION NEURAL NETWORKS

a) convolution Layer

a.1) Phép tính convolution

- Ta giả sử ma trận cần tình convolution là ma trận X có kích thước n*m. Và 1 ma trận k có kích thước là x*x. Kí hiệu phép tính convolution (⊗), kí hiệu Y = X ⊗W Với mỗi phần tử xi j trong ma trận X lấy ra một ma trận có kích thước bằng kích thước của kernel W có phần tử xij làm trung tâm (đây là vì sao kích thước của kernel thường lẻ) gọi là ma trận A. Sau đó tính tổng các phần tử của phép tính element-wise của ma trận A và ma trận W, rồi viết vào ma trận kết quả Y (như hình 2.4).



Hình 2.4:

- Dễ thấy với phép tính như trên thì shape của ma trân Y sẽ nhỏ hơn shape của ma trận X đầu vào. Với những trường hợp cần ma trận Y có cùng khích thương với ma trận X chúng ta thêm 1 hệ số được gọi là padding vào ma trận X rồi thực hiện phép tính convolution như bình thường.

a.2) Lớp tích chập (Convolution)

- Tích chập là lớp đầu tiên để trích xuất các tính năng từ hình ảnh đầu vào. Tích chập duy trì mối quan hệ giữa các pixel bằng cách tìm hiểu các tính năng hình ảnh bằng cách sử dụng các ô vương nhỏ của dữ liệu đầu vào. Nó là 1 phép toán có 2 đầu vào như ma trận hình ảnh và 1 bộ lọc hoặc hạt nhân.
- Giả sử input của 1 convolutional layer tổng quát là tensor kích thước H*W*D. Kernel có kích thước F*F*D (kernel luôn có depth bằng depth của input và F thường là số lẻ vì ô vuông lưới chẵn * chẵn thì sẽ không có 1 ô vuông ở tâm đối xứng-> giảm độ chính xác), stride: S, padding: P. Convolutional layer áp dụng K kernel.
- Output có kích thước

$$(\frac{H+2*P-F}{S}+1)*(\frac{W+2*p-F}{S}+1)*K$$

Lưu ý: Ta có thể trồng nhiều lớp convolution lên nhau để lấy được đặc trưng của ảnh. Trước khi output của lớp convolution trước làm input của lớp sau thì ta đưa qua 1 hàm phi tuyến tính.

a.3) Ý nghĩa của lớp Convolution

- Convolution sẽ giúp làm mờ, làm nét ảnh. Lấy được các đặc trưng của ảnh. Mỗi kernel khác nhau sẽ đều có những tác dụng khác nhau (Hình 2.5).

Ví dụ: tôi apply 3 kernel để thu được 3 feature riêng biệt từ input ban đầu:

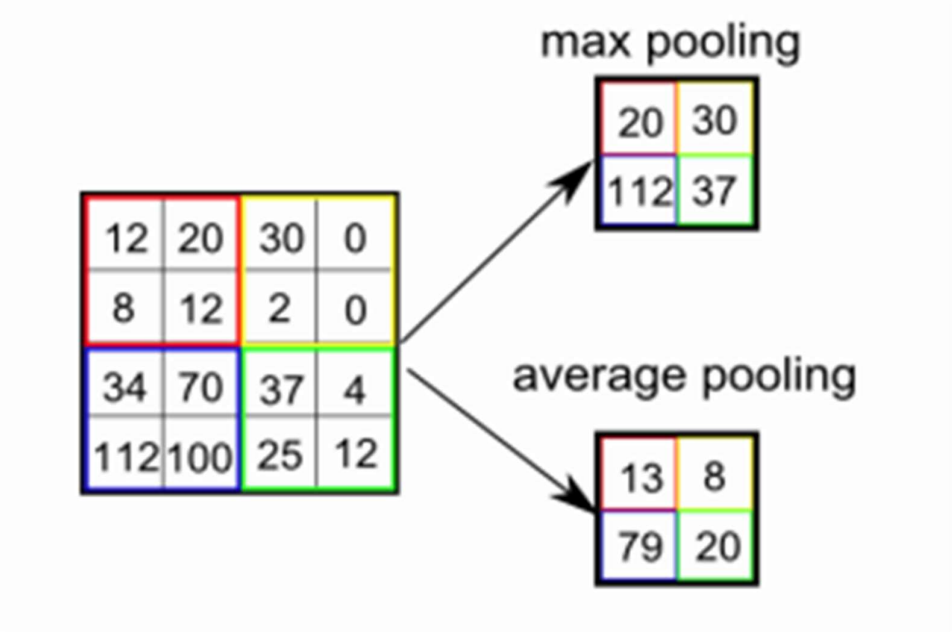
Hình 2.5: Ví dụ về các lớp convolution

- Và cũng đừng lo việc phải tìm bao nhiêu kernel hay lưu các kernel về để dùng dần. Đó là việc của CNN, nó sẽ tự động tìm các kernel, tự dò ra các feature.

b) Lớp Pooling

- Giống như convolution layer, dữ liệu đầu vào được convoluted với một bộ lọc để tạo thành một ma trận convolution.
- 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.
- Ví dụ trong một Pooling layer một cụm 3x3 được lấy từ dữ liệu đầu và 5x5. Điều này sẽ được miêu tả ở Hình 2.3, có hai hàm pooling, average hoặc max còn tùy thuộc vào

hàm. Giá trị trung bình của tất cả các phần tử sau đó được đặt vào C_{11} hoặc giá trị tối đa của cụm được đặt vào C_{11}. Quá trình này được lặp đi lặp lại tùy thuộc vào chức năng, từ phần tử C_{11} cho đến phần tử C_{33} cho pooling 3x3.



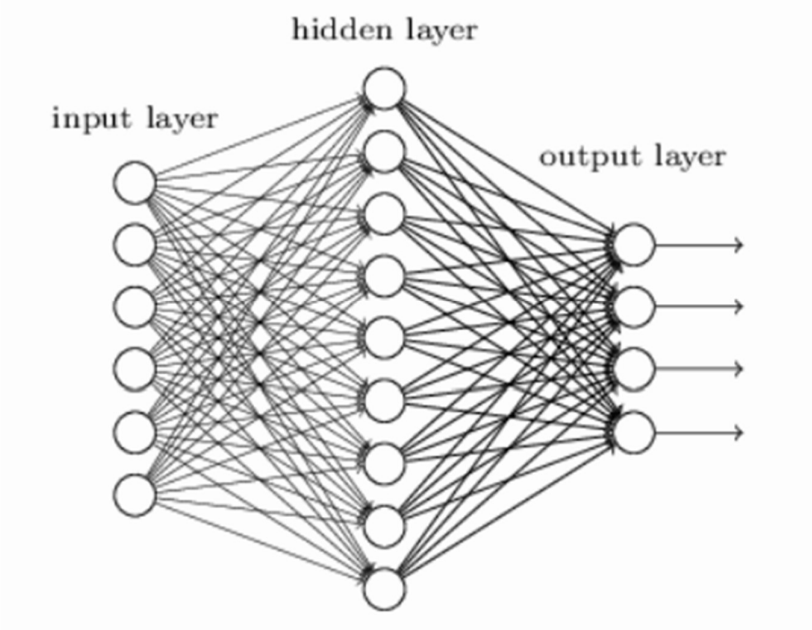
Hình 2.6: Pooling operation

- Pooling layer thường được dùng giữa các convolutional layer, để giảm kích thước dữ liệu nhưng vẫn giữ được các thuộc tính quan trọng. Việc giảm kích thước dữ liệu giúp giảm các phép tính toán trong model. Bên cạnh đó, với phép pooling kích thước ảnh giảm, do đó lớp convolution học được các vùng có kích thước lớn hơn.

c) Lớp Full Connected

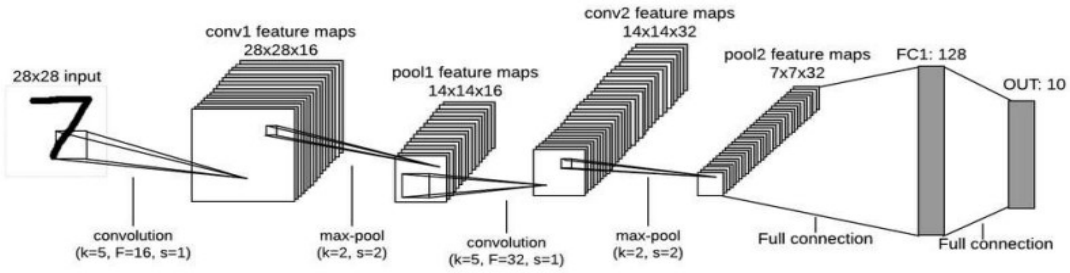
- Lớp này có nhiệm vụ đưa ra kết quả sau khi lớp convolutional layer và pooling layer đã nhận được ảnh truyền. Lúc này, ta thu được kết quả là model đã đọc được thông tin của ảnh và để liên kết chúng cũng như cho ra nhiều output hơn thì ta sử dụng fully connected layer.
- Ngoài ra, nếu như fully connected layer có được giữ liệu hình ảnh thì chúng sẽ chuyển nó thành mục chưa được phân chia chất lượng. Cái này khá giống với phiếu bầu rồi chúng sẽ đánh giá để bầu chọn ra hình ảnh có chất lượng cao nhất.
- Bằng nhiều convolution và pooling layer, đầu ra được nhận đến một fully connected layer dưới dạng đầu vào như trong Hình 2.4. Một fully connected layer thường được đặt ở phần cuối trong cấu trúc CNN, hoặc ở đâu đó trung gian. Loại layer này rất giống với đa lớp truyền tiếp perceptron (feedforward multilayer perceptron). Trong layer

này, tất cả thông tin được lọc, lấy mẫu được thu thập để CNN bắt đầu học. Hầu hết các trọng số cũng nằm trong phần này của mạng.



Hình 2.7: Full connected layer

- Sau đó ta dùng các fully connected layer để kết hợp các đặc điểm của ảnh để ra được output của model. Dưới đây là một mô hình CNN dùng để nhận dạng ký tự viết tay:



2.2.3. BỘ PHÂN LOẠI CONVNET.

* Nhận dạng chữ số viết tay dựa trên CNN (Face recognition)

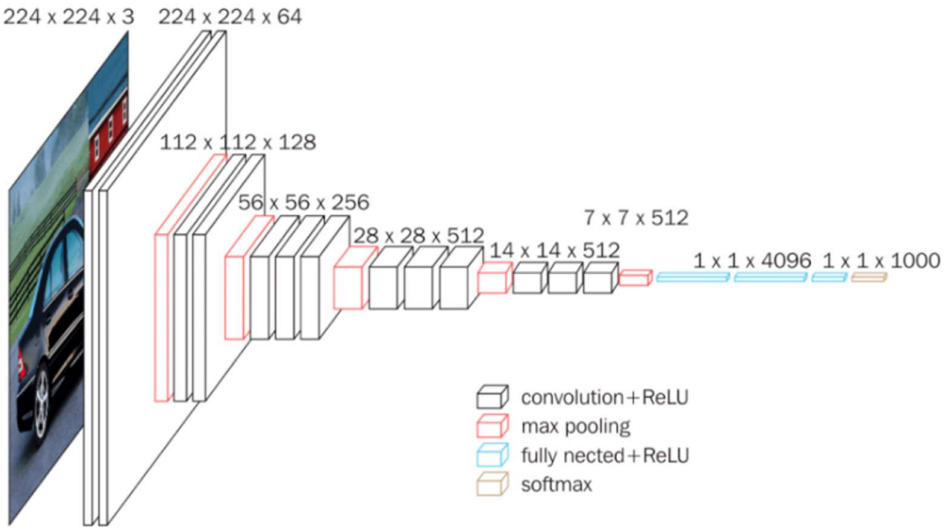
- CNN chủ yếu được sử dụng trong nhận dạng đối tượng bằng cách lấy hình ảnh làm đầu vào và sau đó phân loại chúng trong một danh mục nhất định. Nhận dạng chữ số viết tay là một trong những loại đó. Chúng ta sẽ có một bộ hình ảnh là các chữ số viết tay với các nhãn từ 0 đến 9. Đọc bài đăng khác của tôi để bắt đầu với CNN.

- Đối với điều này, chúng tôi sẽ sử dụng CƠ SỞ DỮ LIỆU MNIST của các chữ số viết tay. Tập dữ liệu này có một bộ đào tạo gồm 60.000 ví dụ và một bộ thử nghiệm gồm 10.000 ví dụ.
- Chúng tôi sẽ tạo ra một mô hình sẽ được đào tạo từ 60.000 đầu vào, và sau đó chúng tôi sẽ kiểm tra độ chính xác của mô hình của chúng tôi trên 10.000 ví dụ bộ thử nghiệm. Chúng ta sẽ sử dụng thư viện Keras với backend Tensorflow để xây dựng mô hình và sẽ tải xuống tập dữ liệu từ chính Keras bằng cách sử dụng từ keras.datasets import mnist.

2.2.4. MỘT SỐ MẠNG CNN NỔI TIẾNG

* nVGG 16

- VGG16 là mạng convolutional neural network được đề xuất bởi K. Simonyan and A. Zisserman, University of Oxford. Model sau khi train bởi mạng VGG16 đạt độ chính xác 92.7% top-5 test trong dữ liệu ImageNet gồm 14 triệu hình ảnh thuộc 1000 lớp khác nhau. Giờ áp dụng kiến thức ở trên để phân tích mạng VGG 16.



Hình 2.8: Minh họa kiến trúc VGG 16

- Phân tích về VGG 16:

- Convolutional layer: kích thước 3*3, padding=1, stride=1. Tại sao không ghi stride, padding mà vẫn biết? Vì mặc định sẽ là stride=1 và padding để cho output cùng width và height với input.
- Pool/2: max pooling layer với size 2*2

- 3*3 conv, 64: thì 64 là số kernel áp dụng trong layer đấy, hay depth của output của layer đấy.
- Càng các convolutional layer sau thì kích thước width, height càng giảm nhưng depth càng tăng.
- sau khá nhiều convolutional layer và pooling layer thì dữ liệu được flatten và cho vào fully connected layer.

2.2.5. SỬ DỤNG CNN VÀO BÀI TOÁN NHẬN ĐỐI TƯỢNG

- Một trong những lĩnh vực quan trọng của Trí tuệ nhân tạo (Artificial Intelligence) là thị giác máy (Computer Vision). Computer Vision là một lĩnh vực bao gồm các phương pháp thu nhận, xử lý ảnh kỹ thuật số, phân tích và nhận dạng các hình ảnh, phát hiện các đối tượng, tạo ảnh, siêu phân giải hình ảnh và nhiều hơn vậy. Object Detection có lẽ là khía cạnh sâu sắc nhất của thị giác máy do số lần sử dụng trong thực tế.
- Object Detection đề cập đến khả năng của hệ thống máy tính và phần mềm để định vị các đối tượng trong một hình ảnh và xác định từng đối tượng. Object Detection đã được sử dụng rộng rãi để phát hiện khuôn mặt, phát hiện xe, đếm số người đi bộ, hệ thống bảo mật và xe không người lái. Có nhiều cách để nhận diện đối tượng có thể được sử dụng cũng như trong nhiều lĩnh vực thực hành. Giống như mọi công nghệ khác, một loạt các ứng dụng sáng tạo và tuyệt vời của Object Detection sẽ đến từ các lập trình viên và các nhà phát triển phần mềm.
- Bắt đầu sử dụng các phương pháp nhận diện đối tượng hiện đại trong các ứng dụng và hệ thống, cũng như xây dựng các ứng dụng mới dựa trên các phương pháp này. Việc triển nhận diện đối tượng sớm liên quan đến việc sử dụng các thuật toán cổ điển, giống như các thuật toán được hỗ trợ trong OpenCV, thư viện computer vision phổ biến. Tuy nhiên, các thuật toán cổ điển này không thể đạt được hiệu suất đủ để làm việc trong các điều kiện khác nhau.
- Việc áp dụng đột phát và nhanh cóng của deep learning vào năm 2012 đã đưa vào sự tồn tại các thuật toán và phương pháp phát hiện đối tượng hiện đại và chính xác cao như R-CNN, Fast-RCNN, Faster-RCNN, RetinaNet và nhanh hơn nhưng rất chính xác như SSD và YOLO. Sử dụng các phương pháp và thuật toán này, dựa trên deep learning và cũng dựa trên việc học máy đòi hỏi rất nhiều kiến thức về toán học và việc học sâu. Có hàng triệu chuyên gia lập trình và các nhà phát triển phần mềm muốn tích

hợp và tạo ra các sản phẩm mới sử dụng object detection. Nhưng công nghệ này xa tầm tay của họ và phức tạp để hiểu và sử dụng thực tế của nó.

2.3. TÌM HIỂU VỀ NGÔN NGỮ LẬP TRÌNH PYTHON

2.3.1. Python là gì?

- Python là một ngôn ngữ lập trình bậc cao cho các mục đích lập trình đa năng. Ngôn ngữ lập trình Python được tạo bởi Guido van Rossum và lần đầu ra mắt vào năm 1991. Python được thiết kế với ưu điểm mạnh là dễ đọc, dễ học và dễ nhớ. Python là ngôn ngữ có hình thức rất sáng sủa, cấu trúc rõ ràng, thuận tiện cho người mới học lập trình. Cấu trúc của Python còn cho phép người sử dụng viết mã lệnh với số lần gõ phím tối thiểu.

Hình 2.9:

2.3.2. Python có lịch sử như thế nào?

- Guido Van Rossum, một lập trình viên máy tính ở Hà Lan, đã tạo ra Python. Ông bắt đầu phát triển ngôn ngữ này vào năm 1989 tại Centrum Wiskunde & Informatica (CWI), ban đầu chỉ là một dự án tiêu khiển để giúp ông giết thời gian trong dịp Giáng sinh. Tên gọi của ngôn ngữ này được lấy cảm hứng từ chương trình truyền hình Monty

Python's Flying Circus của đài BBC vì Guido Van Rossum là một “fan cứng” của chương trình này.

-Lịch sử các phiên bản Python:

- Guido Van Rossum cho ra mắt phiên bản đầu tiên của ngôn ngữ Python (phiên bản 0.9.0) vào năm 1991. Ngôn ngữ này đã bao gồm các tính năng hữu ích như một số kiểu dữ liệu và hàm để xử lý lỗi.
- Python 1.0 đã được ra mắt vào năm 1994 với các hàm mới để dễ dàng xử lý danh sách dữ liệu, chẳng hạn như ánh xạ, lọc và lược bỏ.
- Python 2.0 đã được ra mắt vào ngày 16 tháng 10 năm 2000, với các tính năng hữu ích mới cho lập trình viên, chẳng hạn như hỗ trợ ký tự Unicode và cách xử lý chi tiết một danh sách nhanh chóng hơn.
- Python 3.0 đã được ra mắt vào ngày 3 tháng 12 năm 2008. Phiên bản này bao gồm các tính năng như hàm in và hỗ trợ nhiều hơn cho việc phân chia số và xử lý lỗi.

2.3.3. Tính năng chính của Python

- Ngôn ngữ lập trình đơn giản, dễ học: Python có cú pháp rất đơn giản, rõ ràng. Nó dễ đọc và viết hơn rất nhiều khi so sánh với những ngôn ngữ lập trình khác như C++, Java, C#. Python làm cho việc lập trình trở nên thú vị, cho phép bạn tập trung vào những giải pháp chứ không phải cú pháp.

- Miễn phí, mã nguồn mở: Bạn có thể tự do sử dụng và phân phối Python, thậm chí là dùng cho mục đích thương mại. Vì là mã nguồn mở, bạn không những có thể sử dụng các phần mềm, chương trình được viết trong Python mà còn có thể thay đổi mã nguồn của nó. Python có một cộng đồng rộng lớn, không ngừng cải thiện nó mỗi lần cập nhật.

- Khả năng di chuyển: Các chương trình Python có thể di chuyển từ nền tảng này sang nền tảng khác và chạy nó mà không có bất kỳ thay đổi nào. Nó chạy liền mạch trên hầu hết tất cả các nền tảng như Windows, macOS, Linux.

- Khả năng mở rộng và có thể nhúng: Giả sử một ứng dụng đòi hỏi sự phức tạp rất lớn, bạn có thể dễ dàng kết hợp các phần code bằng C, C++ và những ngôn ngữ khác (có thể gọi được từ C) vào code Python. Điều này sẽ cung cấp cho ứng dụng của bạn những tính năng tốt hơn cũng như khả năng scripting mà những ngôn ngữ lập trình khác khó có thể làm được.

- Ngôn ngữ thông dịch cấp cao: Không giống như C/C++, với Python, bạn không phải lo lắng những nhiệm vụ khó khăn như quản lý bộ nhớ, dọn dẹp những dữ liệu vô nghĩa, ...Khi chạy code Python, nó sẽ tự động chuyển đổi code sang ngôn ngữ máy tính có thể hiểu. Bạn không cần lo lắng về bất kỳ hoạt động ở cấp thấp nào.
- Thư viện tiêu chuẩn lớn để giải quyết những tác vụ phổ biến: Python có một số lượng lớn thư viện tiêu chuẩn giúp cho công việc lập trình của bạn trở nên dễ thở hơn rất nhiều, đơn giản vì không phải tự viết tất cả code. Ví dụ: Bạn cần kết nối cơ sở dữ liệu MySQL trên Web server? Bạn có thể nhập thư viện MySQLdb và sử dụng nó. Những thư viện này được kiểm tra kỹ lưỡng và được sử dụng bởi hàng trăm người. Vì vậy, bạn có thể chắc chắn rằng nó sẽ không làm hỏng code hay ứng dụng của mình.
- Hướng đối tượng: Mọi thứ trong Python đều là hướng đối tượng. Lập trình hướng đối tượng (OOP) giúp giải quyết những vấn đề phức tạp một cách trực quan. Với OOP, bạn có thể phân chia những vấn đề phức tạp thành những tập nhỏ hơn bằng cách tạo ra các đối tượng.

2.3.4. Python mang lại những lợi ích gì?

- Những lợi ích của Python bao gồm:

- Các nhà phát triển có thể dễ dàng đọc và hiểu một chương trình Python vì ngôn ngữ này có cú pháp cơ bản giống tiếng Anh.
- Python giúp cải thiện năng suất làm việc của các nhà phát triển vì so với những ngôn ngữ khác, họ có thể sử dụng ít dòng mã hơn để viết một chương trình Python.
- Python có một thư viện tiêu chuẩn lớn, chứa nhiều dòng mã có thể tái sử dụng cho hầu hết mọi tác vụ. Nhờ đó, các nhà phát triển sẽ không cần phải viết mã từ đầu.
- Các nhà phát triển có thể dễ dàng sử dụng Python với các ngôn ngữ lập trình phổ biến khác như Java, C và C++.
- Cộng đồng Python tích cực hoạt động bao gồm hàng triệu nhà phát triển nhiệt tình hỗ trợ trên toàn thế giới. Nếu gặp phải vấn đề, bạn sẽ có thể nhận được sự hỗ trợ nhanh chóng từ cộng đồng.
- Trên Internet có rất nhiều tài nguyên hữu ích nếu bạn muốn học Python. Ví dụ: bạn có thể dễ dàng tìm thấy video, chỉ dẫn, tài liệu và hướng dẫn dành cho nhà phát triển.

- Python có thể được sử dụng trên nhiều hệ điều hành máy tính khác nhau, chẳng hạn như Windows, macOS, Linux và Unix.

2.3.5. Python được sử dụng như thế nào?

- Ngôn ngữ Python được sử dụng nhiều trong lĩnh vực phát triển ứng dụng, bao gồm những ví dụ sau:

a) Phát trên web phía máy chủ

- Phát triển web phía máy chủ bao gồm những hàm backend phức tạp mà các trang web thực hiện để hiển thị thông tin cho người dùng. Ví dụ: các trang web phải tương tác với cơ sở dữ liệu, giao tiếp với các trang web khác và bảo vệ dữ liệu khi truyền qua mạng.
- Python hữu ích trong việc lập trình mã phía máy chủ bởi vì ngôn ngữ này cung cấp nhiều thư viện bao gồm mã viết sẵn cho các hàm backend phức tạp. Các nhà phát triển cũng sử dụng một loạt các khung Python cung cấp tất cả những công cụ cần thiết để xây dựng ứng dụng web một cách nhanh chóng và dễ dàng hơn. Ví dụ: các nhà phát triển có thể tạo ứng dụng web khung trong nháy mắt bởi vì họ không cần phải lập trình nó từ đầu. Sau đó, họ có thể kiểm tra ứng dụng web này bằng cách sử dụng các công cụ kiểm thử của khung, mà không cần phụ thuộc vào những công cụ kiểm thử bên ngoài.

b) Tự động hóa các tập lệnh Python

- Ngôn ngữ tập lệnh là một ngôn ngữ lập trình tự động hóa các tác vụ mà thường được con người thực hiện. Các lập trình viên thường xuyên sử dụng các tập lệnh Python để tự động hóa nhiều tác vụ hàng ngày như:

- Đổi tên một số lượng lớn tệp cùng lúc
- Chuyển đổi một tệp sang một loại tệp khác
- Loại bỏ các từ trùng lặp trong tệp văn bản
- Thực hiện các phép tính toán cơ bản
- Gửi email
- Tải xuống nội dung
- Thực hiện phân tích nhật ký cơ bản
- Tìm kiếm lỗi trong nhiều tệp

c) Khoa học dữ liệu và máy học

- Khoa học dữ liệu trích xuất thông tin quý giá từ dữ liệu và máy học (ML) dạy máy tính tự động học hỏi từ dữ liệu và đưa ra các dự đoán chính xác. Các nhà khoa học dữ liệu sử dụng Python cho các tác vụ khoa học dữ liệu sau:

- Sửa và loại bỏ dữ liệu không chính xác, hay còn được gọi là làm sạch dữ liệu
- Trích xuất và chọn lọc các đặc điểm đa dạng của dữ liệu
- Ghi nhãn dữ liệu gán tên có ý nghĩa cho dữ liệu
- Tìm các số liệu thống kê khác nhau từ dữ liệu
- Trực quan hóa dữ liệu bằng cách sử dụng các biểu đồ và đồ thị, chẳng hạn như biểu đồ đường, biểu đồ cột, biểu đồ tần suất và biểu đồ tròn

- Các nhà khoa học dữ liệu sử dụng những thư viện ML của Python để đào tạo các mô hình ML và xây dựng các công cụ phân loại giúp phân loại dữ liệu một cách chính xác. Các chuyên gia từ nhiều lĩnh vực sử dụng những công cụ phân loại dựa trên Python để thực hiện các tác vụ phân loại, chẳng hạn như phân loại hình ảnh, văn bản cũng như lưu lượng truy cập mạng, nhận dạng giọng nói và nhận diện khuôn mặt. Các nhà khoa học dữ liệu cũng sử dụng Python cho deep learning, một kỹ thuật ML nâng cao.

d) Phát triển phần mềm

- Các nhà phát triển phần mềm thường sử dụng Python cho những tác vụ phát triển và ứng dụng phần mềm khác nhau, chẳng hạn như:

- Theo dõi lỗi trong mã của phần mềm
- Tự động xây dựng phần mềm
- Đảm nhận quản lý dự án phần mềm
- Phát triển nguyên mẫu phần mềm
- Phát triển các ứng dụng máy tính bằng cách sử dụng những thư viện Giao diện đồ họa người dùng (GUI)
- Phát triển từ các trò chơi văn bản đơn giản cho đến những trò chơi điện tử phức tạp

e) Tự động hóa kiểm thử phần mềm

- Kiểm thử phần mềm là quy trình kiểm tra xem kết quả thực tế từ phần mềm có khớp với kết quả mong đợi không để đảm bảo rằng phần mềm không có lỗi.

- Các nhà phát triển sử dụng khung kiểm thử đơn vị Python, chẳng hạn như Unittest, Robot và PyUnit, để kiểm thử các hàm do họ viết.
- Các kỹ sư kiểm thử phần mềm sử dụng Python để viết các trường hợp kiểm thử cho nhiều tình huống khác nhau. Ví dụ: họ sử dụng ngôn ngữ này để kiểm thử giao diện người dùng của một ứng dụng web, nhiều thành phần của phần mềm và những tính năng mới.

- Các nhà phát triển có thể sử dụng một số công cụ để tự động chạy tập lệnh kiểm thử. Những công cụ này có tên gọi là công cụ Tích hợp liên tục/Triển khai liên tục (CI/CD). Các kỹ sư kiểm thử phần mềm cũng như những nhà phát triển sử dụng các công cụ CI/CD như Travis CI và Jenkins để tự động hóa quy trình kiểm thử. Công cụ CI/CD tự động chạy các tập lệnh kiểm thử Python và báo cáo kết quả kiểm thử bất kỳ khi nào nhà phát triển thêm vào những dòng mã mới.

Chương III: Xây Dựng Chương Trình

3.1. TỔNG QUAN

- Các nhà phát triển đang dốc toàn lực để làm cho máy móc trở nên thông minh hơn, thông minh hơn con người. Học sâu là một trong những kỹ thuật góp phần giúp các nhà phát triển tăng cường máy móc. Để ghi nhớ cách thực hiện một nhiệm vụ, con người làm gì? Con người cứ thực hành và lặp đi lặp lại nhiệm vụ đó nhiều lần để họ thành thạo nhiệm vụ đó. Sau một thời gian, các tế bào thần kinh não bộ của chúng ta có thể tự động kích hoạt và thực hiện công việc một cách nhanh chóng và chính xác. Tương tự như vậy, học sâu tuân theo cách tiếp cận để giải quyết vấn đề. Các loại kiến trúc mạng thần kinh khác nhau được sử dụng bởi các thuật toán học sâu để giải quyết các loại vấn đề khác nhau.

3.1.1. GIỚI THIỆU

- Ở đây chúng tôi sẽ sử dụng bộ dữ liệu MNIST để triển khai ứng dụng nhận dạng chữ số viết tay. Để thực hiện điều này, chúng tôi sẽ sử dụng một loại mạng thần kinh sâu đặc biệt có tên là Mạng thần kinh chuyển đổi. Cuối cùng, chúng tôi cũng sẽ xây dựng Giao diện người dùng đồ họa (GUI) nơi bạn có thể vẽ trực tiếp chữ số và nhận ra nó ngay lập tức.

3.1.2. NHẬN DẠNG CHỮ SỐ VIẾT TAY LÀ GÌ?

- Nhận dạng chữ số viết tay là quá trình cung cấp khả năng cho máy nhận dạng chữ số viết tay của con người. Đó không phải là một nhiệm vụ dễ dàng đối với máy vì các chữ số viết tay không hoàn hảo, thay đổi tùy theo từng người và có thể được tạo ra với nhiều hương vị khác nhau.

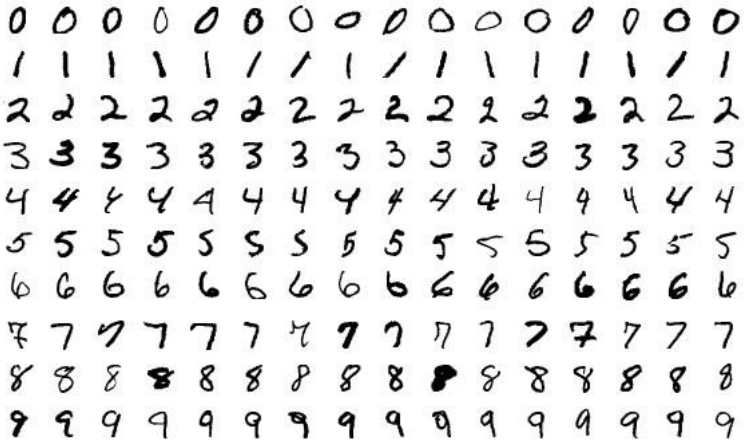
3.1.3. ĐIỀU KIỆN TIÊN QUYẾT?

- Cần có kiến thức cơ bản về học sâu với thư viện Keras, thư viện Tkinter để xây dựng GUI và lập trình Python để chạy dự án tuyệt vời này.
- Các lệnh cài đặt cần thiết cho thư viện này:

- pip install numpy
- pip install tensorflow
- pip install keras
- pip install pillow

3.1.4. TẬP DỮ LIỆU

- Cơ sở dữ liệu MNIST (Viện tiêu chuẩn và công nghệ quốc gia sửa đổi cơ sở dữ liệu) là một tập hợp lớn các chữ số viết tay. Nó có một tập huấn luyện gồm 60.000 ví dụ và một tập kiểm tra gồm 10.000 ví dụ. Nó là một tập hợp con của Cơ sở dữ liệu đặc biệt NIST 3 lớn hơn (các chữ số được viết bởi nhân viên của Cục điều tra dân số Hoa Kỳ) và Cơ sở dữ liệu đặc biệt 1 (các chữ số được viết bởi học sinh trung học) chứa hình ảnh đơn sắc của các chữ số viết tay. Các chữ số đã được chuẩn hóa kích thước và căn giữa trong một hình ảnh có kích thước cố định. Hình ảnh đen trắng (hai mặt) ban đầu từ NIST đã được chuẩn hóa kích thước để vừa với hộp 20x20 pixel trong khi vẫn giữ nguyên tỷ lệ khung hình của chúng. Các hình ảnh thu được chứa các mức xám do kỹ thuật khử răng cưa được sử dụng bởi thuật toán chuẩn hóa. các hình ảnh được căn giữa trong một hình ảnh 28x28 bằng cách tính khối lượng trung tâm của các pixel.



Hình 3.1: Bộ chữ số viết tay MNIST

- Nguồn tham khảo: http://yann.lecun.com/exdb/mnist/

3.2. CÁC BƯỚC THỰC HIỆN VÀ KẾT QUẢ

3.2.1. CÁC BƯỚC THỰC HIỆN

- Thêm các thư viện cần thiết

```
[ ]  import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
     import os
     from sklearn.preprocessing import MinMaxScaler
     from sklearn.model_selection import KFold
     import tensorflow as tf
     from keras.models import Sequential
     from keras.layers import Dense
     from keras.layers import Dropout
     from keras.layers import Flatten
     from keras.layers.convolutional import Conv2D
     from keras.layers.convolutional import MaxPooling2D
     # from keras.optimizers import SGD, Adam
     from keras.callbacks import ReduceLROnPlateau, EarlyStopping
     from keras import backend as K
     from keras.utils import np_utils
     from sklearn.model_selection import train_test_split
     from sklearn.metrics import confusion_matrix
     from scipy import optimize
     from keras.callbacks import ModelCheckpoint, EarlyStopping
     from keras.models import load_model
     from google.colab.patches import cv2_imshow
     import cv2
     from keras.datasets import mnist
```

- Load data bộ MNIST: https://paperswithcode.com/dataset/mnist

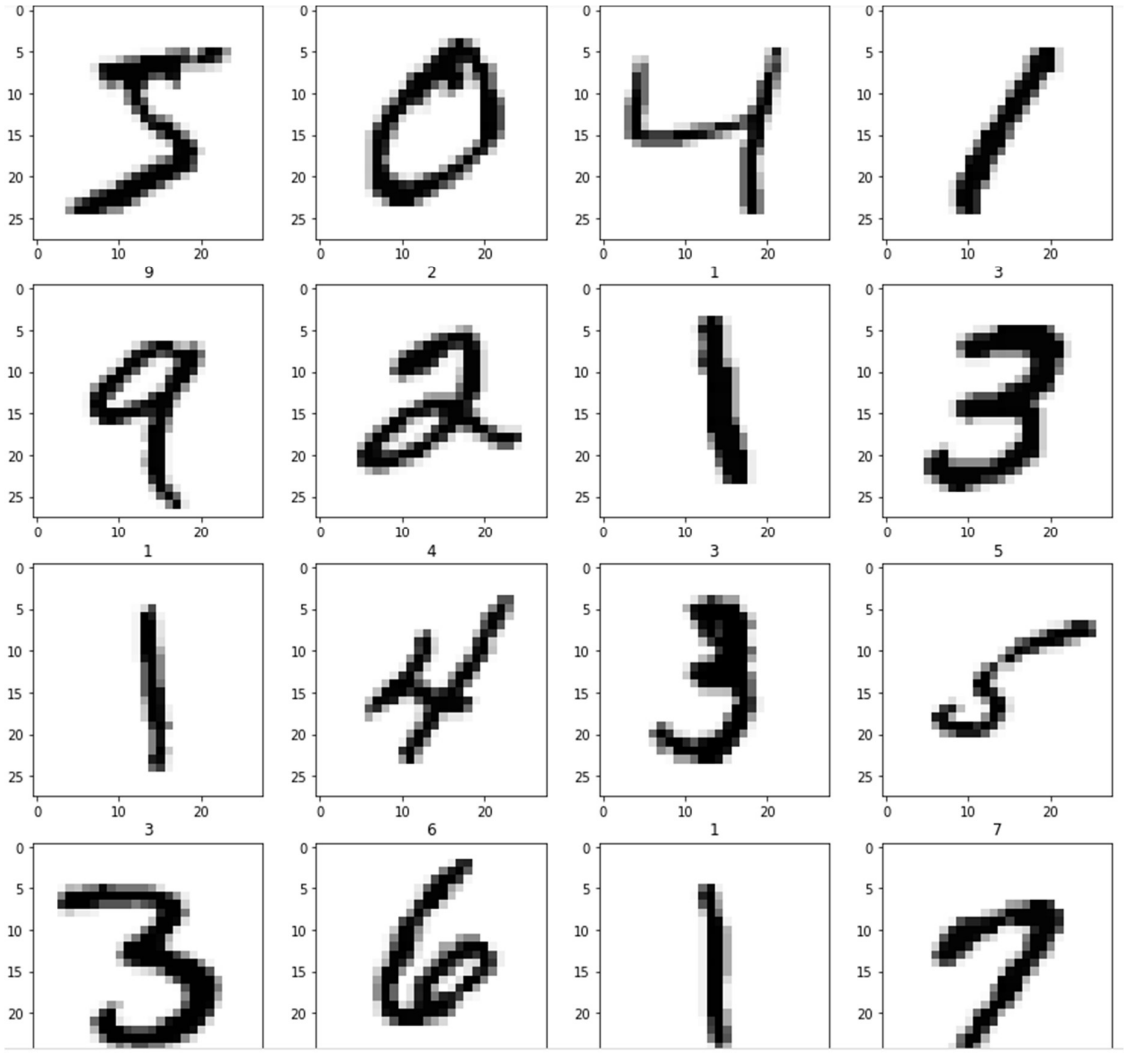
```
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 [==============================] - 2s 0us/step
```

- In random data trong bộ MNIST để test

```
plt.figure(figsize = (15,15))
row, colums = 4, 4
for i in range(16):
    plt.subplot(colums, row, i+1)
    plt.imshow(x_train[i].reshape(28,28),interpolation='nearest', cmap='Greys')
    plt.title((y_train[i]))
plt.show()
```

- Kết quả



- Chia x_train, y_train thành 2 tập x_train, y_train, x_val, y_val để tiến hành train Model

```
[ ] (x_train, x_val, y_train, y_val) = train_test_split(x_train, y_train, test_size=0.2, stratify=y_train, random_state=42)
```

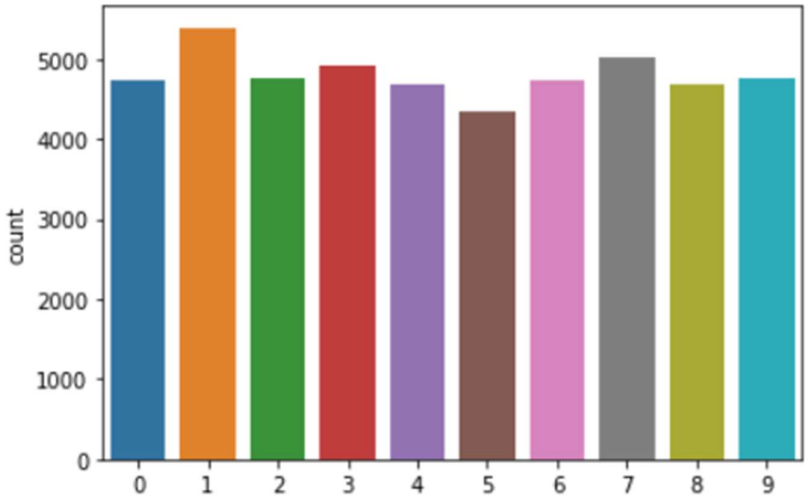
```
print("tap train co {} anh".format(x_train.shape[0]))
print("tap test co {} anh".format(x_test.shape[0]))
print("tap val co {} anh".format(x_val.shape[0]))
```

```
tap train co 48000 anh
tap test co 10000 anh
tap val co 12000 anh
```

- Đồ thị số lượng các ảnh từ số 0 → số 9 trong tập train

```
sns.countplot(y_train)
```

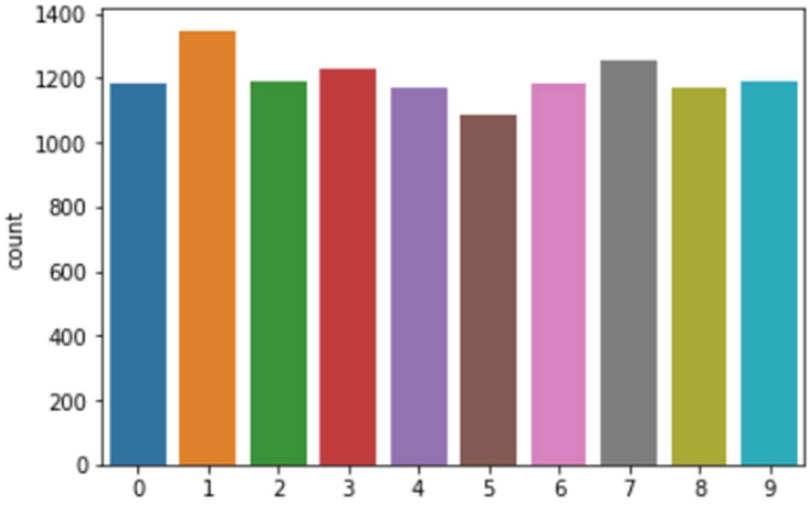
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f75d3670790>
```



- Đồ thị số lượng các ảnh từ số 0 → số 9 trong tập val

```
sns.countplot(y_val)
```

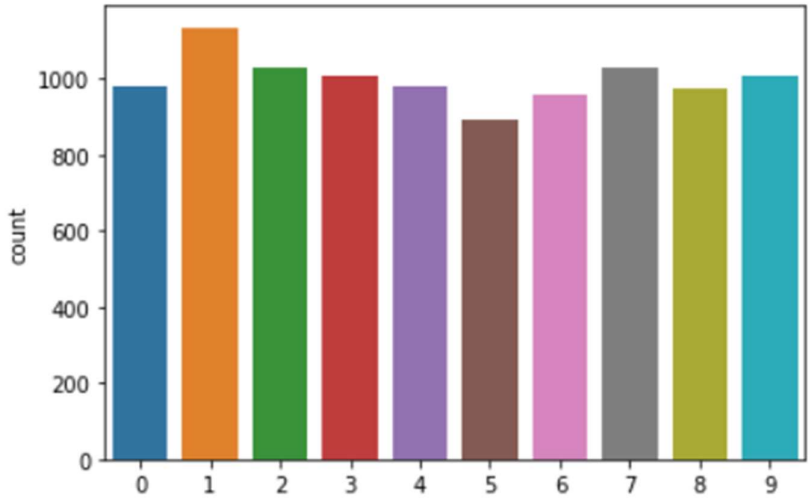
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f75d2ff0990>
```



- Đồ thị số lượng các ảnh từ số 0 → số 9 trong tập test

```
sns.countplot(y_test)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43
  FutureWarning
<matplotlib.axes._subplots.AxesSubplot at 0x7f75d2f7b910>
```



- Chuyển đầu ra thành dạng vector

```
[ ] y_train = tf.keras.utils.to_categorical(y_train, 10)
    y_test = tf.keras.utils.to_categorical(y_test, 10)
    y_val = tf.keras.utils.to_categorical(y_val, 10)
    print(y_train[0])
```

```
[0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
```

```
print(x_train.shape)
print(x_val.shape)
```

```
(48000, 28, 28)
(12000, 28, 28)
```

- Xây dựng Model
  - conv2d -> maxpoooling -> conv2d -> maxpoooling -> platten -> dense (64) -> 128 -> 26
  - Dùng hàm kích hoạt là softmax để đưa ra kết quả dưới dạng xác suất
  - Dùng hàm mất mát categorical_crossentropy

- Kết quả

```
model = Sequential()

model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(28,28,1)))
model.add(MaxPooling2D(pool_size=(2, 2), strides=2))

model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu', padding = 'same'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=2))

model.add(Conv2D(filters=128, kernel_size=(3, 3), activation='relu', padding = 'valid'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=2))

model.add(Flatten())

model.add(Dense(64,activation ="relu"))
model.add(Dense(128,activation ="relu"))

model.add(Dense(10,activation ="softmax"))
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 26, 26, 32)        320

 max_pooling2d (MaxPooling2D  (None, 13, 13, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 4, 4, 128)         73856

 max_pooling2d_2 (MaxPooling  (None, 2, 2, 128)        0
 2D)

 flatten (Flatten)           (None, 512)               0

 dense (Dense)               (None, 64)                32832

 dense_1 (Dense)             (None, 128)               8320

 dense_2 (Dense)             (None, 10)                1290

=================================================================
Total params: 135,114
Trainable params: 135,114
Non-trainable params: 0
```

- Train Model

```
[ ]  batch_size = 32
     epochs = 20
     h = model.fit(x_train,y_train,
                   batch_size = batch_size,
                   epochs = epochs,
                   verbose = 1,
                   validation_data = (x_val,y_val))
```

➢ Kết quả

```
Epoch 1/20
1500/1500 [==============================] - 16s 5ms/step - loss: 0.2324 - accuracy: 0.9445 - val_loss: 0.1023 - val_accuracy: 0.9678
Epoch 2/20
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0628 - accuracy: 0.9808 - val_loss: 0.0654 - val_accuracy: 0.9821
Epoch 3/20
1500/1500 [==============================] - 7s 4ms/step - loss: 0.0490 - accuracy: 0.9847 - val_loss: 0.0522 - val_accuracy: 0.9858
Epoch 4/20
1500/1500 [==============================] - 7s 4ms/step - loss: 0.0425 - accuracy: 0.9873 - val_loss: 0.0468 - val_accuracy: 0.9862
Epoch 5/20
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0342 - accuracy: 0.9899 - val_loss: 0.0610 - val_accuracy: 0.9836
Epoch 6/20
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0313 - accuracy: 0.9907 - val_loss: 0.0537 - val_accuracy: 0.9849
Epoch 7/20
1500/1500 [==============================] - 7s 4ms/step - loss: 0.0268 - accuracy: 0.9921 - val_loss: 0.0746 - val_accuracy: 0.9799
Epoch 8/20
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0269 - accuracy: 0.9921 - val_loss: 0.0539 - val_accuracy: 0.9877
Epoch 9/20
1500/1500 [==============================] - 7s 4ms/step - loss: 0.0216 - accuracy: 0.9941 - val_loss: 0.0567 - val_accuracy: 0.9865
Epoch 10/20
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0200 - accuracy: 0.9942 - val_loss: 0.0466 - val_accuracy: 0.9886
Epoch 11/20
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0185 - accuracy: 0.9949 - val_loss: 0.0461 - val_accuracy: 0.9902
Epoch 12/20
1500/1500 [==============================] - 7s 4ms/step - loss: 0.0195 - accuracy: 0.9945 - val_loss: 0.0737 - val_accuracy: 0.9879
Epoch 13/20
1500/1500 [==============================] - 7s 4ms/step - loss: 0.0180 - accuracy: 0.9958 - val_loss: 0.0614 - val_accuracy: 0.9846
Epoch 14/20
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0147 - accuracy: 0.9962 - val_loss: 0.0593 - val_accuracy: 0.9878
Epoch 15/20
1500/1500 [==============================] - 7s 4ms/step - loss: 0.0190 - accuracy: 0.9950 - val_loss: 0.0685 - val_accuracy: 0.9882
Epoch 16/20
1500/1500 [==============================] - 7s 4ms/step - loss: 0.0121 - accuracy: 0.9966 - val_loss: 0.0854 - val_accuracy: 0.9874
Epoch 17/20
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0169 - accuracy: 0.9958 - val_loss: 0.0598 - val_accuracy: 0.9893
Epoch 18/20
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0139 - accuracy: 0.9963 - val_loss: 0.0781 - val_accuracy: 0.9876
Epoch 19/20
1500/1500 [==============================] - 7s 4ms/step - loss: 0.0152 - accuracy: 0.9963 - val_loss: 0.0827 - val_accuracy: 0.9860
Epoch 20/20
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0148 - accuracy: 0.9962 - val loss: 0.0669 - val accuracy: 0.9887
```

- Kết quả sau khi train model

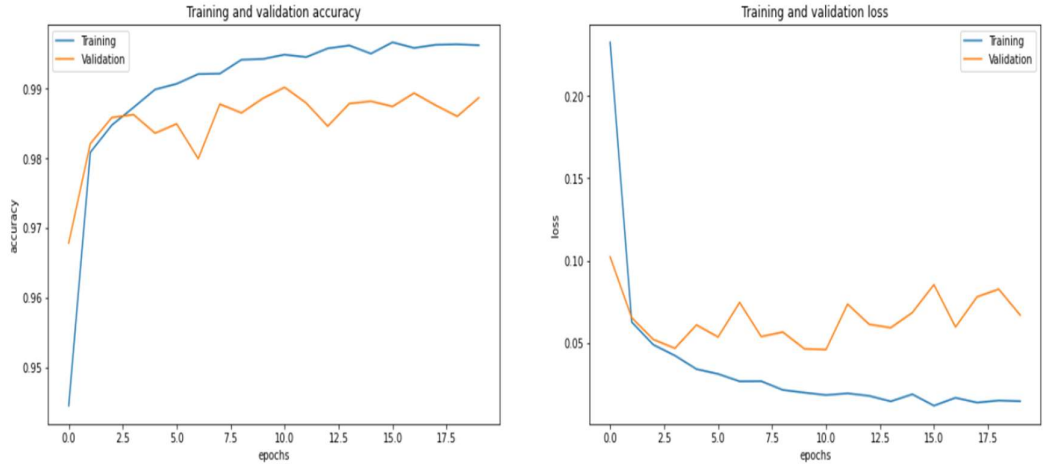
```
res = model.evaluate(x_test, y_test)
print("hiệu suất của mô hình là : {}".format(res[1] * 100))

plt.figure(figsize=(20, 6))
plt.subplot(121)
plt.plot(h.history['accuracy'], label='Training')
plt.plot(h.history['val_accuracy'], label='Validation')
plt.title('Training and validation accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend()

plt.subplot(122)
plt.plot(h.history['loss'], label='Training')
plt.plot(h.history['val_loss'], label='Validation')
plt.title('Training and validation loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```

- Ta thu được hiệu suất mô hình

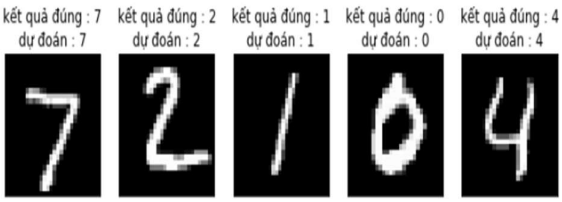
```
313/313 [==============================] - 2s 5ms/step - loss: 0.0551 - accuracy: 0.9907
hiệu suất của mô hình là : 99.07000064849854
```



3.2.2. KẾT QUẢ SAU KHI TEST

```
plt.figure(figsize=(10, 10))
for i in range(5):
    plt.subplot(1,5,i+1)
    plt.imshow(x_test[i], cmap='gray')
    s = "kết quả đúng : " + str(np.argmax(y_test[i])) +'\ndự đoán : ' + str(np.argmax(model.predict(x_test[i].reshape(1,28,28,1)))  )
    s = str(s)
    plt.title(s)
    plt.xticks([])
    plt.yticks([])
plt.show()
```

```
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 18ms/step
```



3.3. TỔNG KẾT VÀ KIẾN NGHỊ

3.3.1. TỔNG KẾT

- Ngày nay khi mà nền kinh tế, đời sống của con người ngày càng phát triển. Các công cụ, và giải phát ngày càng trở nên thông minh trong nền kinh tế 4.0 hiện nay. Việc chuyển đổi số và số hóa các hoạt động bình thường mà phải tốn rất nhiều công sức của nhiều người là điều tất nhiên. Nhu cầu của con người về những công nghệ thông minh cũng ngày càng tăng lên trong những năm gần đây. Nhất là về lĩnh vực trí tuệ nhân tạo. Mỗi ngày có vô số các bài báo được công bố về lĩnh vực này. Chính vì thế việc nhận diện ra chữ số viết tay của một cơ quan, công ty, nơi đông người là rất cần thiết. Từ những thực tiễn đó nhóm em nghĩ đề tài này có thể được nâng cấp và phát triển nhiều hơn. Từ đó có thể đưa vào thực tiễn để ứng dụng trong cuộc sống.

3.3.2. KIẾN NGHỊ

- Mặc dù đạt được kết quả tưởng đối tốt khi test nhưng nhóm em cũng đã nhận thấy được rất nhiều điểm yếu của ứng dụng đề tài này.

- Điểm yếu:

- Việc nhận diện còn khá kém. Không nhận được những chữ viết tay quá xấu không rõ ràng hoặc nghệ thuật.
- Model còn tương đối nặng.

- Hướng phát triển

- Chúng em vừa trình bày giải thuật máy học rừng ngẫu nhiên xiên phân (rODT) sử dụng các đặc trưng toàn cục (GIST), cho phép nhận dạng chính xác ký tự số viết tay. Bước tiền xử lý trích đặc trưng toàn cục từ ảnh ký tự số cho ra bảng dữ liệu có số chiều lớn. Kết quả thử nghiệm trên tập dữ liệu thực MNIST cho thấy rằng giải thuật rODT do chúng tôi đề xuất nhận dạng rất chính xác khi so sánh với các phương pháp nhận dạng hiện nay. Phương pháp đề xuất đạt hiệu quả nhận dạng chính xác cao nhưng không cần bất cứ xử lý đặc biệt nào. Các thử nghiệm cho nhận dạng ký tự viết tay gồm ký tự số và 26 ký tự alphabet cho thấy phương pháp của chúng em khá tốt. Trong tương lai gần, chúng em kết hợp hệ thống này với các phương pháp khác cho phép trích, đọc số xe. Hướng tiếp cận có thể áp dụng vào các vấn đề tương tự trong lĩnh vực nhận dạng, phân lớp, tìm kiếm ảnh.

TÀI LIỆU THAM KHẢO

1. L. Breiman, J.H. Friedman, R.A. Olshen and C. Stone. Classification and Regression Trees. Wadsworth International, 1984.

2. L. Breiman. Bagging predictors. Machine Learning 24(2):123–140, 1996.

3. L. Breiman. Random forests. Machine Learning 45(1):5–32, 2001.

4. C.C. Chang and C.J. Lin. Libsvm – a library for support vector machines. 2001. http://www.csie.ntu.edu.tw/cjlin/libsvm.

5. T.N. Do, S. Lallich, N.K. Pham and P. Lenca. Classifying very-high-dimensional data with random forests of oblique decision trees. in Advances in Knowledge Discovery and Management Vol. 292, Springer-Verlag, 2009, pp. 39-55.

6. M. Douze, M., H. Jégou, H. Sandhawalia, L. Amsaleg, and C. Schmid. Evaluation of GIST descriptors for web-scale image search. In Proceedings of the ACM International Conference on Image and Video Retrieval, 2009, pp. 1–8.

7. Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. Computational Learning Theory, 1995, pp. 23–37.

8. B. Kégl and R. Busa-Fekete. Boosting products of base classifiers. In Proceedings of the 26th Annual International Conference on Machine Learning, 2009, pp. 497–504.

9. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In Proceedings of the IEEE, 1998, pp. 2278–2324.

10. LeCun, Y. and C. Cortes. The MNIST database of handwritten digits. 1989.

11. D. Lowe. Distinctive image features from scale invariant keypoints. International Journal of Computer Vision, 2004, pp. 91–110. 12. S. Murthy, S. Kasif, S. Salzberg and R. Beigel. Oc1: Randomized induction of oblique decision trees. In Proceedings of the Eleventh National Conference on Artificial Intelligence, 1993, pp. 322–327. 13. A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. International Journal of Computer Vision 42, 145–175, 2001. 14. J.R. Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann, 1993. 15. Y. Simard, D. Steinkraus, J. Platt. Best Pratices for Convolutional Neural Network Applied to Visual Document Analysis. in Intl Conference on Document Analysis and Recogntion, 2003, pp. 958-962. 16. V. Vapnik. The Nature of Statistical Learning Theory. Springer-Verlag, 1995. 17. H. Witten and E. Frank. Data Mining: Practical

Machine Learning Tools and Techniques. Morgan Kaufmann, San Francisco, 2nd edition, 2005.