

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**KHOA ĐIỆN – ĐIỆN TỬ**

**------**🙟🙟🕮🙝🙝**-----**

**BÁO CÁO BÀI TẬP LỚN MÔN**

**XỬ LÝ ẢNH SỐ**

**NHẬN DẠNG SỐ VIẾT TAY**

**Giảng viên hướng dẫn: TS. Nguyễn Thúy Bình**

**Họ và tên sinh viên : Lê Quang Huy**

**Nguyễn Tiến Hải**

**Lớp : Điện tử tin học công nghiệp 2-K61**

*Hà Nội, 05/2024*

**MỤC LỤC**

[CHƯƠNG 1: TỔNG QUAN VỀ NHẬN DẠNG CHỮ SỐ VIẾT TAY 3](#_Toc166063073)

[1.1. Tổng quan về bài toán nhận dạng chữ số viết tay 3](#_Toc166063074)

[1.2. Ứng dụng bài toán nhận dạng số viết tay 3](#_Toc166063075)

[CHƯƠNG 2: MÔ HÌNH MẠNG NƠ-RON TÍCH CHẬP SÂU 0](#_Toc166063076)

[2.1. Mạng nơ-ron tích chập 0](#_Toc166063077)

[**2.1.1. Tích chập** 0](#_Toc166063078)

[**2.1.2. Mô hình mạng nơ-ron tích chập** 0](#_Toc166063079)

[**2.1.3. Hoạt động của mô hình CNN** 4](#_Toc166063080)

[2.2. Mô hình mạng nơ-ron tích chập sâu cho bài toán 4](#_Toc166063081)

[3.1. Thực nghiệm 6](#_Toc166063082)

[**3.1.1. Tập dữ liệu** 6](#_Toc166063083)

[**3.1.2. Chuẩn bị dữ liệu cho huấn luyện mô hình** 6](#_Toc166063084)

[**3.1.3. Chuẩn hóa dữ liệu** 7](#_Toc166063085)

[**3.1.4. Xây dựng và huấn luyện mô hình** 8](#_Toc166063086)

[3.2. Kết quả mô phỏng 12](#_Toc166063087)

[3.3. Kết luận và hướng phát triển 13](#_Toc166063088)

DANH MỤC HÌNH ẢNH

Chương 2:

[Hình 2. 1.Minh họa tích chập 0](#_Toc166063123)

[Hình 2. 2.Minh họa kiến trúc CNN dùng trong phân loại ảnh 1](#_Toc166063124)

[Hình 2. 3.Tích chập với các bộ lọc 2](#_Toc166063125)

[Hình 2. 4.Mô hình kích hoạt ReLU 2](#_Toc166063126)

[Hình 2. 5.Tính toán với phương pháp MaxPooling 3](#_Toc166063127)

[Hình 2. 6.Mô hình CNN 4](#_Toc166063128)

Chương 3:

[Hình 3. 1.Minh học tập dữ liệu Mnist 6](#_Toc166063110)

[Hình 3. 2.Hình ảnh ví dụ trong tập dữ liệu kiểm tra 7](#_Toc166063111)

[Hình 3. 3.Kết quả huấn luyện sau 20 epochs 10](#_Toc166063112)

[Hình 3. 4.Kết quả quá trình huyến luyện tập kiểm tra 10](#_Toc166063113)

[Hình 3. 5.Biểu đồ model loss 11](#_Toc166063114)

[Hình 3. 6.Biểu đồ Epoch 11](#_Toc166063115)

[Hình 3. 7.Hình ảnh kiểm tra 12](#_Toc166063116)

# **CHƯƠNG 1: TỔNG QUAN VỀ NHẬN DẠNG CHỮ SỐ VIẾT TAY**

## **1.1. Tổng quan về bài toán nhận dạng chữ số viết tay**

Bài toán Nhận dạng chữ số viết tay (Handwritten Digit Recognition) đã thu hút sự quan tâm rộng rãi trong ngành công nghệ thông tin và trí tuệ nhân tạo. Đây là một thách thức thú vị, đòi hỏi các kỹ thuật xử lý ảnh và học máy tiên tiến để có thể phân loại chính xác các chữ số viết tay từ 0 đến 9.

Trong những năm gần đây, các phương pháp dựa trên mạng nơ-ron tích chập (Convolutional Neural Networks - CNNs) đã chứng tỏ hiệu quả vượt trội so với các phương pháp truyền thống. CNNs có khả năng học các đặc trưng phức tạp từ dữ liệu ảnh, giúp cải thiện đáng kể độ chính xác của việc phân loại chữ số viết tay.

Ngoài ra, các kỹ thuật học sâu (Deep Learning) khác như mạng nơ-ron hồi quy (Recurrent Neural Networks - RNNs) và mạng nơ-ron tăng cường (Reinforcement Learning Networks) cũng đang được áp dụng để giải quyết bài toán này. Các phương pháp này có khả năng học các mô hình phức tạp hơn, mang lại hiệu quả cao hơn so với các phương pháp truyền thống dựa trên hình học hoặc mô hình thống kê.

Việc nghiên cứu và phát triển các thuật toán nhận dạng chữ số viết tay không chỉ có ứng dụng trong các sản phẩm công nghệ, mà còn có tiềm năng rất lớn trong các lĩnh vực như xử lý ngôn ngữ tự nhiên, ghi chú tự động, và các hệ thống trợ giúp người dùng.

## **1.2. Ứng dụng bài toán nhận dạng số viết tay**

Công nghệ nhận dạng số viết tay đã trở thành một phần không thể thiếu trong cuộc sống hàng ngày, từ các ứng dụng quản lý hàng hóa đến việc bảo mật thông tin cá nhân. Trên thị trường bán lẻ, việc sử dụng hệ thống nhận dạng số viết tay giúp cải thiện quản lý hàng tồn kho và tăng cường trải nghiệm mua sắm cho khách hàng thông qua việc đọc mã vạch sản phẩm một cách chính xác và nhanh chóng.

Ngoài ra, tích hợp công nghệ nhận dạng số viết tay vào các hệ thống bảo mật cũng đem lại lợi ích lớn. Việc sử dụng chữ ký số hoặc mã PIN viết tay không chỉ giúp xác minh danh tính một cách hiệu quả mà còn nâng cao độ bảo mật cho dữ liệu cá nhân và tài khoản ngân hàng.

Công nghệ này cũng được áp dụng trong việc quét và chuyển đổi hình ảnh văn bản có chứa số viết tay thành dữ liệu số, giúp tăng cường hiệu suất làm việc và giảm thiểu sai sót trong quá trình nhập liệu.

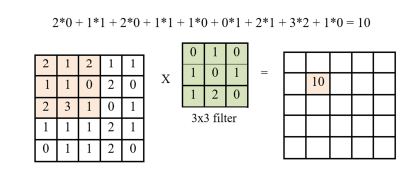
Cuối cùng, trong lĩnh vực giáo dục và nghiên cứu, công nghệ nhận dạng số viết tay đóng vai trò quan trọng trong việc chuyển đổi biểu đồ và sơ đồ viết tay thành dữ liệu số, giúp sinh viên và nhà nghiên cứu dễ dàng truy cập và phân tích thông tin một cách hiệu quả.

# **CHƯƠNG 2: MÔ HÌNH MẠNG NƠ-RON TÍCH CHẬP SÂU**

## **2.1. Mạng nơ-ron tích chập**

### **2.1.1. Tích chập**

Tích chập (Convolution) được sử dụng đầu tiên trong xử lý tín hiệu số (Signal Processing). Nhờ vào nguyên lý biến đổi thông tin, các nhà khoa học đã áp dụng kỹ thuật này vào xử lý ảnh và video số. Để dễ hình dung, chúng ta có thể xem tích chập như một cửa sổ số trượt (sliding window) áp đặt lên một ma trận như minh họa dưới đây.



Hình 2. 1.Minh họa tích chập

Ma trận bên trái là một ảnh xám, mỗi giá trị của ma trận tương ứng với một điểm ảnh (pixel) có giá trị biến thiên từ 0-255. Cửa sổ trượt còn có tên gọi khác là Kernel. Ở đây, ta dùng một ma trận filter kích thước 3x3 nhân từng thành phần tương ứng với ma trận ảnh bên trái. Giá trị đầu ra là tổng của các tích. Kết quả của tích chập là một ma trận sinh ra từ việc trượt ma trận filter và thực hiện tích chập cùng lúc lên toàn bộ ma trận ảnh bên trái

### **2.1.2. Mô hình mạng nơ-ron tích chập**

Mô hình CNN chỉ đơn giản gồm một vài layer của convolution kết hợp với các hàm kích hoạt phi tuyến như *Relu* hay *tanh* để tạo ra thông tin ở mức trừu tượng hơn cho các layer tiếp theo.

Trong mô hình mạng nơ-ron truyền thẳng (feedforward nơ-ron network), các layer kết nối trực tiếp với nhau thông qua một trọng số w (weighted vector). Các layer này còn được gọi là có kết nối đầy đủ (fully connected layer) hay affine layer.

Trong mô hình CNN thì ngược lại. Các layer liên kết được với nhau thông qua cơ chế convolution. Layer tiếp theo là kết quả convolution từ layer trước đó, nhờ vậy mà ta có được các kết nối cục bộ. Nghĩa là mỗi nơ-ron ở layer tiếp theo sinh ra từ filter áp đặt lên một vùng ảnh cục bộ của nơ-ron layer trước đó. Mỗi layer như vậy được áp đặt các filter khác nhau, thông thường có vài trăm đến vài nghìn filter như vậy. Một số layer khác như pooling/subsampling layer dùng để chắt lọc lại các thông tin hữu ích hơn (loại bỏ các thông tin nhiễu).

Trong suốt quá trình huấn luyện, CNN sẽ tự động học được các thông số cho các filter. Ví dụ, trong nhiệm vụ phân lớp ảnh như được minh họa trong hình 2, CNN sẽ cố gắng tìm ra thông số tối ưu cho các filter tương ứng theo thứ tự raw pixel edges shapes facial high-level features. Layer cuối cùng được dùng để phân lớp ảnh.

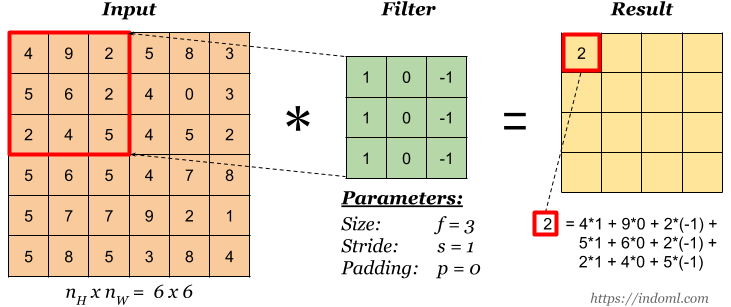


Hình 2. 2.Minh họa kiến trúc CNN dùng trong phân loại ảnh

CNN có tính bất biến và tính kết hợp cục bộ (Location Invariance and Compositionality). Với cùng một đối tượng, nếu đối tượng này được chiếu theo các góc độ khác nhau (translation, rotation, scaling) thì độ chính xác của thuật toán sẽ bị ảnh hưởng đáng kể. Pooling layer sẽ cho tính bất biến đối với phép dịch chuyển (translation), phép quay (rotation) và phép co giãn (scaling). Tính kết hợp cục bộ cho ta các cấp độ biểu diễn thông tin từ mức độ thấp đến mức độ cao và trừu tượng hơn thông qua convolution từ các filter. Đó là lý do tại sao CNN cho ra mô hình với độ chính xác rất cao.

#### **2.1.2.1. Convolutional layer**

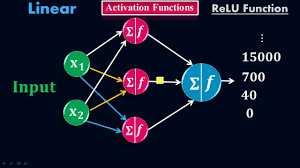
Layer này chính là nơi thể hiện tư tưởng ban đầu của CNN. Thay vì kết nối toàn bộ điểm ảnh, layer này sẽ sử dụng một tập các bộ lọc (filters) có kích thước nhỏ so với ảnh (thường là 5×5 hoặc 3×3) áp vào một vùng trong ảnh và tiến hành tính tích chập giữa bộ lọc và giá trị điểm ảnh trong vùng cục bộ đó. Bộ lọc sẽ lần lượt được dịch chuyển theo một giá trị bước trượt (stride) chạy dọc theo ảnh và quét toàn bộ ảnh.



Hình 2. 3.Tích chập với các bộ lọc

Như vậy, với một bức ảnh 32×32 và một filter 3×3, ta sẽ có kết quả là một tấm ảnh mới có kích thước 32×32 (với điều kiện đã thêm padding vào ảnh gốc để tính tích chập cho các trường hợp filter quét ra các biên cạnh) là kết quả tích chập của filter và ảnh. Với bao nhiêu filter trong lớp này thì ta sẽ có bấy nhiêu ảnh tương ứng mà lớp này trả ra và được truyền vào lớp tiếp theo. Các trọng số của filter ban đầu sẽ được khởi tạo ngẫu nhiên và sẽ được học dần trong quá trình huấn luyện mô hình. Hình 3 minh họa của một phép tính convolution với bộ lọc có kích thước 3×3.

#### **2.1.2.2. Rectified Linear Unit (ReLU) Layer**



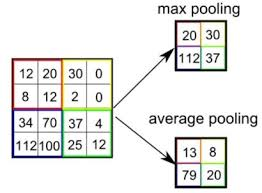
Hình 2. 4.Mô hình kích hoạt ReLU

Layer này thường được cài đặt ngay sau layer Convolution. Layer này sử dụng hàm kích hoạt f(x) = max (0, x). Nói một cách đơn giản, layer này có nhiệm vụ chuyển toàn bộ giá trị âm trong kết quả lấy từ lớp Convolution thành giá trị 0. Ý nghĩa của cách cài đặt này chính là tạo nên tính phi tuyến cho mô hình. Tương tự như trong mạng truyền thẳng, việc xây dựng dựa trên các phép biến đổi tuyến tính sẽ khiến việc xây dựng đa tầng đa lớp trở nên vô nghĩa. Có rất nhiều cách để khiến mô hình trở nên phi tuyến như sử dụng các hàm kích hoạt sigmoid, tanh, nhưng hàm f(x) = max (0, x) dễ cài đặt, tính toán nhanh mà vẫn hiệu quả.

#### **2.1.2.3. Pooling Layer**

Layer này sử dụng một cửa sổ trượt quét qua toàn bộ ảnh dữ liệu, mỗi lần trượt theo một bước trượt (stride) cho trước. Khác với layer Convolution, layer Pooling không tính tích chập mà tiến hành lấy mẫu (subsampling). Khi cửa sổ trượt trên ảnh, chỉ có một giá trị được xem là giá trị đại diện cho thông tin ảnh tại vùng đó (giá trị mẫu) được giữ lại. Các phương thức lấy phổ biến trong layer Pooling là MaxPooling (lấy giá trị lớn nhất), MinPooling (lấy giá trị nhỏ nhất) và AveragePooling (lấy giá trị trung bình).

Xét một ảnh có kích thước 32×32 và layer Pooling sử dụng bộ lọc có kích thước 2×2 với bước trượt stride là 2, phương pháp sử dụng là MaxPooling. Bộ lọc sẽ lần lượt trượt qua ảnh, với mỗi lần trượt chỉ có giá trị lớn nhất trong 4 giá trị nằm trong vùng cửa sổ 2×2 của bộ lọc được giữ lại và đưa vào ma trạn đầu ra. Như vậy, sau khi qua layer Pooling, ảnh sẽ giảm kích thước xuống còn 16×16 (kích thước mỗi chiều giảm 2 lần).



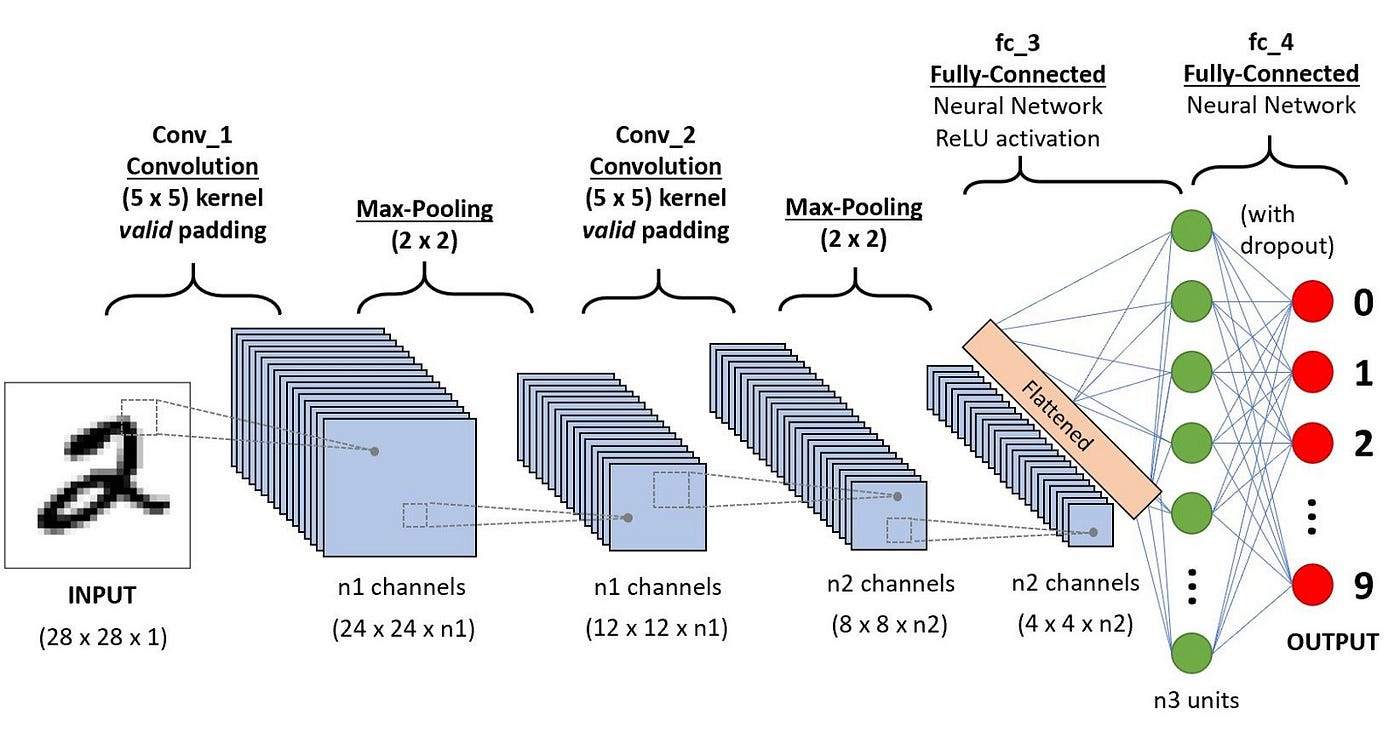
Hình 2. 5.Tính toán với phương pháp MaxPooling

Pooling Layer có vai trò giảm kích thước dữ liệu. Với một bức ảnh kích thước lớn qua nhiều Pooling Layer sẽ được thu nhỏ lại tuy nhiên vẫn giữ được những đặc trưng cần cho việc nhận dạng (thông qua cách lấy mẫu). Việc giảm kích thước dữ liệu sẽ làm giảm lượng tham số, tăng hiệu quả tính toán và góp phần kiểm soát hiện tượng quá khớp (overfitting).

#### **2.1.2.4. Fully Connected (FC) Layer**

Layer này tương tự với layer trong mạng nơ-ron truyền thẳng, các giá trị ảnh được liên kết đầy đủ vào các nơ-ron trong layer tiếp theo. Sau khi ảnh được xử lý và rút trích đặc trưng từ các layer trước đó, dữ liệu ảnh sẽ không còn quá lớn so với mô hình truyền thẳng nên ta có thể sử dụng mô hình truyền thẳng để tiến hành nhận dạng

### **2.1.3. Hoạt động của mô hình CNN**



Hình 2. 6.Mô hình CNN

Mô hình CNN được hình thành bằng cách kết nối các layer nêu trên lại với nhau. Mô hình bắt đầu với Convolutional Layer. ReLU Layer thường được cài đặt ngay sau Convolutional Layer hoặc thậm chí kết hợp cả hai layer này thành một layer. Các layer tiếp theo có thể là Convolutional hay Pooling tùy theo kiến trúc mà ta muốn xây dựng. Cuối cùng sẽ là Fully Connected Layer để tiến hành phân lớp**.**

## **2.2. Mô hình mạng nơ-ron tích chập sâu cho bài toán**

Layer đầu tiên của mạng là một Input Layer, layer này chứa các ảnh cần phân loại. Mỗi ảnh là một ma trận xám có kích thước n x n. Ví dụ, kích thước của ảnh là 28×28, khi đó mỗi ảnh có 784 phần tử, mỗi phần tử là một giá trị mức xám.

Layer tiếp theo của mạng là một Convolutional Layer được gọi là Conv2D. Layer này nhận đầu vào là các ảnh từ lớp Input. Trong Convolutional Layer, sử dụng nhiều filter với kích thước bằng nhau để quét trên ảnh đầu vào (từ Input layer) và tạo ra các ánh xạ đặc trưng cho ảnh. Sau Convolutional Layer, sử dụng hàm kích hoạt ReLU.

Tiếp theo, định nghĩa một Pooling Layer có giá trị tối đa được gọi là MaxPooling2D. Layer này nhận đầu vào là kết quả của Convolutional Layer ở trên và nó thực hiện chắt lọc lại thông tin, loại bỏ thông tin nhiễu trước khi truyền cho layer tiếp theo của mạng.

Sau Pooling Layer, sử dụng một Dropout Layer với giá trị Dropout được thiết lập là 0,25. Nó được cấu hình để loại trừ ngẫu nhiên 25% tổng số các nơ-ron trong layer để giảm vấn đề overfitting.

Để chắt lọc được nhiều thông tin hữu ích từ ảnh, xây dựng mạng CNN sâu hơn bằng cách bổ sung thêm một số layer của mạng, chúng bao gồm các layer sau: Convolutional, Pooling, Dropout. Trong đó, Convolutional Layer sẽ sử dụng nhiều filter với kích thước bằng nhau, Pooling Layer sử dụng cửa sổ với kích thước pool size là 2×2, Dropout Layer với giá trị Dropout là 0,25.

Tiếp theo, sử dụng một layer chuyển đổi dữ liệu ma trận 2D thành một véc-tơ gọi là Flatten. Kết quả thu được một véc-tơ các giá trị đặc trưng của ảnh, véc-tơ này phù hợp với định dạng đầu vào của một MLP.

Sau khi thu được véc-tơ đặc trưng của ảnh qua các layer của CNN, sử dụng MLP làm bộ phân loại để phân loại ảnh. MLP của chúng tôi sử dụng nhiều layer ẩn với số nơ-ron được cấu hình trong quá trình thực nghiệm. Do đây là nhiệm vụ phân loại đa lớp (10 lớp) nên đầu ra của MLP sử dụng 10 nơ-ron và một hàm kích hoạt softmax để đưa ra các dự đoán là các giá trị xác suất cho mỗi lớp.

**CHƯƠNG 3: THỰC NGHIỆM VÀ KẾT QUẢ MÔ PHỎNG**

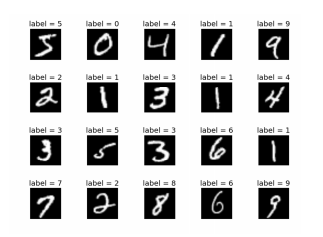
## **3.1. Thực nghiệm**

### **3.1.1. Tập dữ liệu**

Trong phần thực nghiệm, chúng em sử dụng tập dữ liệu MNIST. Đây là tập dữ liệu thường dùng để đánh giá hiệu quả của các mô hình nhận dạng ký tự số viết tay. Tập dữ liệu MNIST có nguồn gốc từ tập NIST do tổ chức National Institute of Standards and Technology (NIST) cung cấp, sau đó được LeCun cập nhật và chia thành 2 tập riêng biệt:

Tập dữ liệu huấn luyện gồm có 60.000 ảnh kích thước 28×28 của chữ số viết tay được dùng cho việc huấn luyện mô hình học máy. Tất cả các ảnh trong tập dữ liệu đều được căn chỉnh và biến đổi thành dữ liệu dạng điểm gồm 60.000 phần tử (ký tự số) có 784 chiều là giá trị mức xám của các điểm ảnh, 10 lớp (giá trị từ 0 đến 9).

Tập dữ liệu kiểm tra gồm có 10.000 ảnh của chữ số viết tay được dùng cho việc kiểm thử. Các ảnh trong tập dữ liệu kiểm tra cũng được biến đổi và căn chỉnh thành dữ liệu điểm gồm 10.000 phần tử trong 784 chiều, 10 lớp (giá trị từ 0 đến 9). Hình 6 là ví dụ về một số mẫu của tập dữ liệu.



Hình 3. 1.Minh học tập dữ liệu Mnist

### **3.1.2. Chuẩn bị dữ liệu cho huấn luyện mô hình**

Để tải tập dữ liệu MNIST, sử dụng thư viện Tensorflow. Cơ sở dữ liệu MNIST là một tâp dữ liệu lớn về các chữ số viết tay thường được sử dụng để đào tạo các hệ thống xử lý hình ảnh khác nhau. Sử dụng module ***tf.keras.datasets.mnist***để tải và sử dụng tập dữ liệu này. Sau khi tải bộ dữ liệu MNIST, chia thành hai tập dữ liệu để huấn luyện và kiểm tra.

import tensorflow as tf

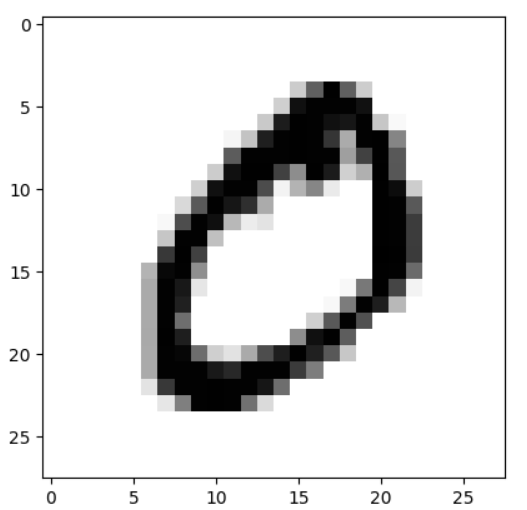
mnist = tf.keras.datasets.mnist

(x\_train, y\_train), (x\_test, y\_test) = mnist.load\_data()

import matplotlib.pyplot as plt

plt.imshow(x\_train[1], cmap = plt.cm.binary)

plt.show()



Hình 3. 2.Hình ảnh ví dụ trong tập dữ liệu kiểm tra

Sau khi thực thi đoạn chương trình trên, tập dữ liệu MNIST sẽ được tải và lưu trên máy tính. Đồng thời chương trình cũng đưa ra hình ảnh đầu tiên của tập dữ liệu kiểm tra.

### **3.1.3. Chuẩn hóa dữ liệu**

Trước tiên, cần kiểm tra giá trị của từng pixel trước khi chuẩn hóa. Chuẩn hóa là một bước tiền xử lý phổ biến trong học máy giúp thu nhỏ dữ liệu để có trung bình bằng không và phương sai đơn vị. Điều này có thể giúp cải thiện hiệu suất của mô hình bằng cách đảm bảo rằng tất cả các đặc trưng đầu vào đều ở cùng một tỷ lệ.

Trong ảnh mức xám, các giá trị nằm trong khoảng từ 0 – 255, vì vậy cần chuẩn hóa bằng cách sử dụng *tf.keras.utils.normalize.*

x\_train = tf.keras.utils.normalize(x\_train, axis =1 )

x\_test = tf.keras.utils.normalize(x\_test, axis =1 )

plt.imshow(x\_train[0], cmap = plt.cm.binary)

print(y\_train[0])

Sau khi chuẩn hóa, ta nhận được tập giá trị nằm trong khoảng từ 0 -1 vì chúng đã được chia cho 255.

### **3.1.4. Xây dựng và huấn luyện mô hình**

Để xây dựng và huấn luyện mô hình, cần thay đổi kích thước của hình ảnh để phù hợp cho việc tính tích chập.

import numpy as np

IMG\_SIZE = 28

x\_trainr = np.array(x\_train).reshape(-1, IMG\_SIZE, IMG\_SIZE, 1)

x\_testr = np.array(x\_test).reshape(-1, IMG\_SIZE, IMG\_SIZE, 1)

print('Training samples dimension', x\_trainr.shape)

print('Testing samples dimension', x\_testr.shape)

Sau đó, tạo ra một mạng nơ-ron để huấn luyện.

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, MaxPooling2D

model = Sequential()

#Fist convolution layer

model.add(Conv2D(64, (3,3), input\_shape = x\_trainr.shape[1:]))

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size = (2,2)))

#2nd convolution layer

model.add(Conv2D(64, (3,3)))

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2,2)))

#3nd convolution layer

model.add(Conv2D(64, (3,3)))

model.add(Activation('relu'))

model.add(MaxPooling2D(pool\_size=(2,2)))

#fully connected layer 1

model.add(Flatten())

model.add(Dense(64))

model.add(Activation('relu'))

#fully connected layer 2

model.add(Dense(32))

model.add(Activation('relu'))

#last fully connected layer

model.add(Dense(10))

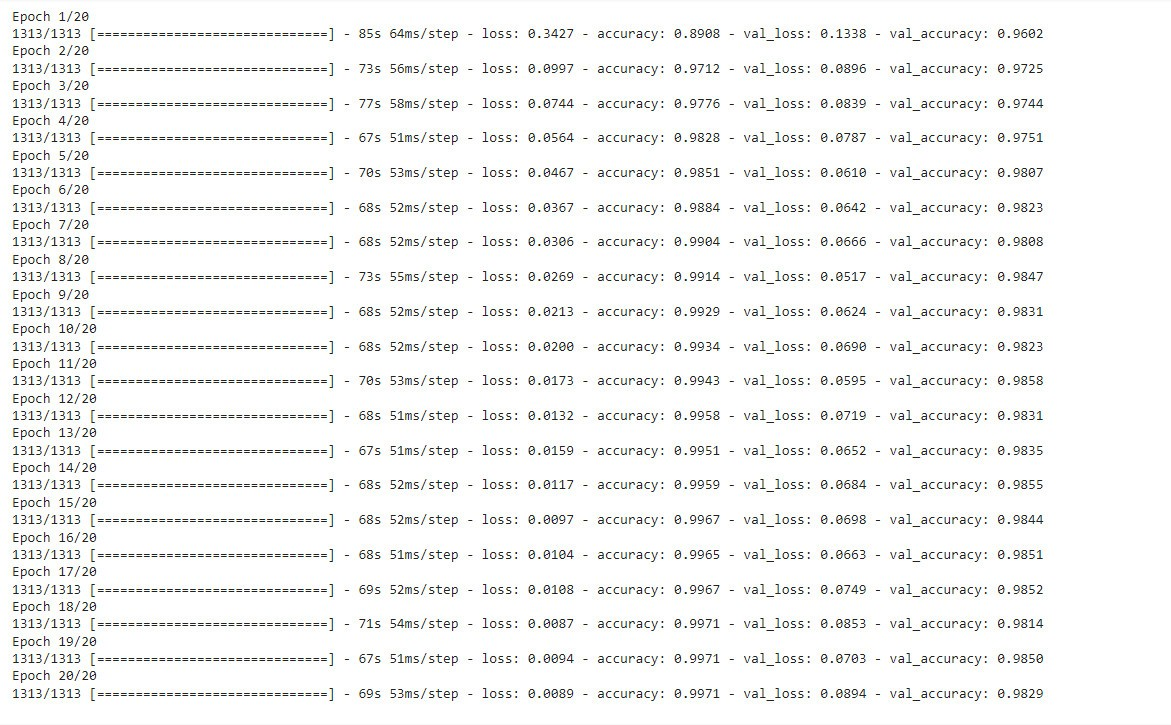
model.add(Activation('softmax'))

Định nghĩa các tham số cần thiết để huấn luyện mô hình, bao gồm hàm mất mát *sparse\_categorical\_crossentropy* , hàm tối ưu hóa *adam* và số liệu đánh giá ( độ chính xác ) *accuracy* để đánh giá tỷ lễ mẫu được phân loại chính xác

model.compile(loss = 'sparse\_categorical\_crossentropy', optimizer =

'adam', metrics = ['accuracy'])

Huấn luyện mô hình với số lần lặp lại toàn bộ dữ liệu huấn luyện trong quá trình là *epochs= 5*, mỗi epochs mô hình sẽ được cập nhật để giảm thiểu hàm mất mát trên tập dữ liệu. Và sử dụng 30% của tập dữ liệu huấn luyện để đánh giá hiệu suất của mô hình sau mỗi epoch

history = model.fit(x\_trainr, y\_train, epochs= 5,validation\_split= 0.3) 

Hình 3. 3.Kết quả huấn luyện sau 20 epochs

Sau quá trình huấn luyện, đánh giá hiệu suất của mô hình trên tập dữ liệu kiểm tra *x\_testr* và nhãn *y\_test* thu được kết quả với độ chính xác là 98,4 % và mất mát

5,63 %

test\_loss, test\_acc = model.evaluate(x\_testr, y\_test

print('test loss on 10000 test samples', test\_loss)

print('validation accuracy on 10000 test samples', test\_acc)

A number and a number

Description automatically generated with medium confidence

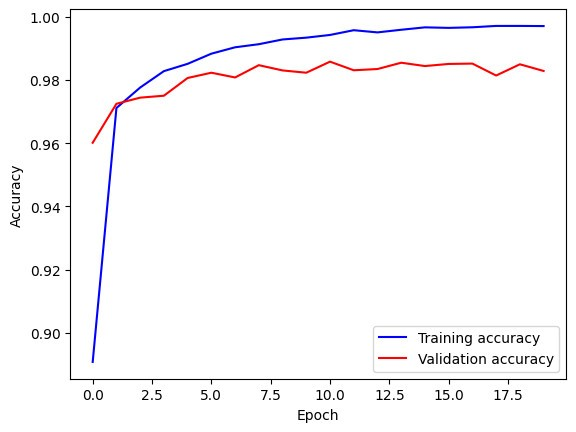
Hình 3. 4.Kết quả quá trình huyến luyện tập kiểm tra

Dưới đây là đồ thị biểu diễn giá trị của hàm mất mát và độ chính xác của mô hình trên tập dữ liệu huấn luyện và kiểm tra sau mỗi epoch



Hình 3. 5.Biểu đồ model loss

Khi số epoch tăng lên thì cả Train Loss và Validation Loss đều giảm



Hình 3. 6.Biểu đồ Epoch

Khi số epoch tăng lên thì cả Train Loss và Validation Loss đều tăng

## **3.2. Kết quả mô phỏng**

import cv2

img = cv2.imread('my\_drawing.png')

plt.imshow(img[:,:,::-1])

plt.axis('off')

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

resized = cv2.resize(gray, (28,28), interpolation = cv2.INTER\_AREA)

newing = tf.keras.utils.normalize (resized, axis = 1)

newing = np.array(newing).reshape(-1, IMG\_SIZE, IMG\_SIZE, 1)

predictions = model.predict(newing)

print(np.argmax(predictions))

Sau quá trình huấn luyện, kiểm tra, cho chạy chương trình trên ta thu được kết quả khá chính xác

Graphical user interface

Description automatically generated with medium confidence

Hình 3. 7.Hình ảnh kiểm tra

## **3.3. Kết luận và hướng phát triển**

Tập dữ liệu MNIST đã trở thành một điểm khởi đầu quen thuộc cho nhiều dự án nhận dạng chữ viết tay. Tuy nhiên, với kích thước ảnh nhỏ và độ phức tạp thấp, MNIST không thể hiện được đầy đủ khả năng của các mô hình nhận dạng chữ viết tay. Đối với các ứng dụng thực tế, đặc biệt là trong môi trường có nhiều nhiễu và biến dạng ảnh, cần có các tập dữ liệu phức tạp hơn để huấn luyện mô hình. Một ví dụ tiêu biểu là tập dữ liệu Oracle-MNIST, bao gồm các ký tự cổ từ nhiều danh mục khác nhau, với các thách thức đặc biệt về nhiễu và biến dạng ảnh. Sử dụng các tập dữ liệu mới và phức tạp hơn như Oracle-MNIST có thể giúp cải thiện hiệu suất của các mô hình nhận dạng chữ viết tay và làm cho chúng trở nên linh hoạt hơn trong các bối cảnh thực tế.

Ngoài ra, việc áp dụng các kỹ thuật tiên tiến hơn trong lĩnh vực học sâu và học máy, như mạng nơ-ron sâu, học chuyển giao và tăng cường học, cũng có thể đóng vai trò quan trọng trong việc cải thiện hiệu suất của các mô hình nhận dạng chữ viết tay. Các phương pháp này có thể giúp mô hình học được các đặc trưng phức tạp hơn và đồng thời giảm thiểu ảnh hưởng của nhiễu và biến dạng trong quá trình nhận dạng. Điều này làm tăng tính ứng dụng và độ tin cậy của các hệ thống nhận dạng chữ viết tay trong nhiều tình huống thực tế.