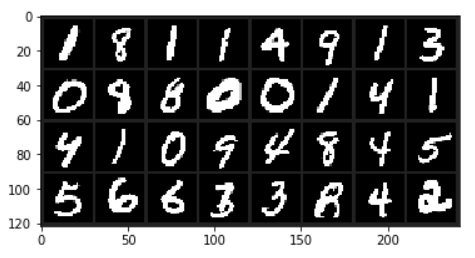
# **1.1.** **Mô tả tập dữ liệu**

MNIST được giới thiệu năm 1998 bởi Yann Lecun và cộng sự nhằm đánh giá các mô hình phân lớp.

[MNIST](http://yann.lecun.com/exdb/mnist/) là tập dữ liệu về chữ số viết tay từ 0 đến 9, bao gồm 2 tập con: training set gồm 60.000 ảnh các chữ số viết tay và test set gồm 10.000 ảnh các chữ số. Trong đó, mỗi hình là một ảnh đen trắng chứa một số được viết tay có kích thước là 28x28.

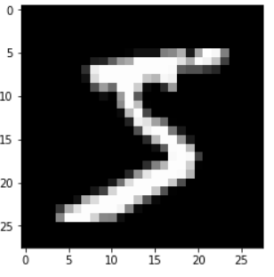
Website chính thức của tập dữ liệu: <http://yann.lecun.com/exdb/mnist/>



Tập dữ liệu MNIST chữ số viết tay từ 0 đến 9

# **1.2. Mô tả bài toán**

Ban đầu bài toán có ảnh xám kích thước 28\*28 của chữ số từ 0 đến 9 và muốn dự đoán số đấy là số mấy. Ví dụ:



Dữ liệu đầu tiên trong MNIST dataset

Bài toán phân lớp được áp dụng để dự đoán chữ số viết tay qua ảnh.

Kỹ thuật học máy được áp dụng: Neural Network (C[onvolutional Neural Network](https://nttuan8.com/bai-6-convolutional-neural-network/))

# **1.3. Áp dụng bài toán học máy**

#### *# 1. Thêm các thư viện cần thiết*

**import numpy as np**

**import matplotlib.pyplot as plt**

**from keras.models import Sequential**

**from keras.layers import Dense, Dropout, Activation, Flatten**

**from keras.layers import Conv2D, MaxPooling2D**

**from keras.utils import np\_utils**

**from keras.datasets import mnist**

## **1.3.1. Mô tả về xử lý dữ liệu (Training set, Validation set, Test set)**

Trong bài toán trên:

* 50.000 dữ liệu trong tổng số 60.000 dữ liệu được gọi là **training set**, có thể hiểu là dữ liệu dùng để dạy cho model học.
* 10.000 dữ liệu còn lại gọi là **validation set**, là để đánh giá xem model hiện tại có tốt không, thường được dùng để chỉnh các tham số của model.
* 10.000 dữ liệu **test set** có sẵn là để đánh giá xem model hoạt động với dữ liệu thực tế có tốt không.

Như vậy MNIST dataset có 60.000 dữ liệu ở training set ở trong MNIST, ta sẽ chia ra 50.000 dữ liệu cho training set và 10.000 dữ liệu cho validation set. Vẫn giữ nguyên 10.000 dữ liệu của test set.

# 2. Load dữ liệu MNIST

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

X\_val, y\_val = X\_train[50000:60000,:], y\_train[50000:60000]

X\_train, y\_train = X\_train[:50000,:], y\_train[:50000]

print(X\_train.shape)

#show 25 dữ liệu

plt.figure(figsize=(10, 10))

for i in range(25):

plt.subplot(5, 5, i+1)

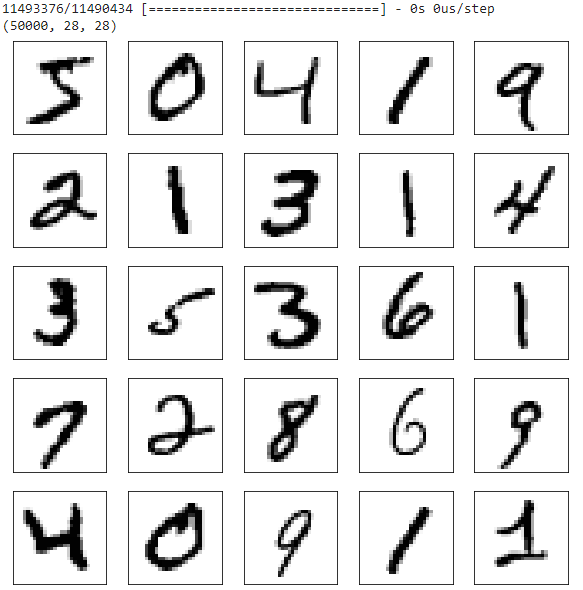
plt.xticks([])

plt.yticks([])

plt.imshow(X\_train[i], cmap='binary')

plt.show()

**OUT:**

****

Dữ liệu input cho mô hình Convolutional Neural Network là 1 tensor 4 chiều (N, W, H, D), trong bài này là ảnh xám nên W = H = 28, D = 1, N là số lượng ảnh cho mỗi lần training.

Do dữ liệu ảnh ở trên có kích thước là (N, 28, 28) tức là (N, W, H) nên cần reshape lại thành kích thước N *28* 28 \* 1 để giống kích thước mà Keras yêu cầu.

# 3. Reshape lại dữ liệu cho đúng kích thước mà keras yêu cầu

X\_train = X\_train.reshape(X\_train.shape[0], 28, 28, 1)

X\_val = X\_val.reshape(X\_val.shape[0], 28, 28, 1)

X\_test = X\_test.reshape(X\_test.shape[0], 28, 28, 1)

Bước này chuyển đổi one-hot encoding label Y của ảnh

Ví dụ số 5 thành vector [0, 0, 0, 0, 0, 1, 0, 0, 0, 0]

***# 4. One hot encoding label (Y)***

**Y\_train = np\_utils.to\_categorical(y\_train, 10)**

**Y\_val = np\_utils.to\_categorical(y\_val, 10)**

**Y\_test = np\_utils.to\_categorical(y\_test, 10)**

**print('Dữ liệu y ban đầu ', y\_train[0])**

**print('Dữ liệu y sau one-hot encoding ',Y\_train[0])**

**OUT:**

****

## **1.3.2. Mô tả về kỹ thuật học máy**

Xây dựng model:

Vì input của model là ảnh nên sử dụng convolutional neural network (CNN).

Mô hình chung bài toán CNN:

**Input image -> Convolutional layer (Conv) + Pooling layer (Pool) -> Fully connected layer (FC) -> Output.**

Thay vì chỉ muốn kết quả là ảnh là số mấy, ta muốn dự đoán phần trăm của ảnh là số nào. Ví dụ: 90% ảnh là số 5, 1% ảnh là số 1,…

*# 5. Định nghĩa model*

model = Sequential()

*# Thêm Convolutional layer với 32 kernel, kích thước kernel 3\*3*

*# dùng hàm sigmoid làm activation và chỉ rõ input\_shape cho layer đầu tiên*

model.add(Conv2D(32, (3, 3), activation='sigmoid', input\_shape=(28,28,1)))

*# Thêm Convolutional layer*

model.add(Conv2D(32, (3, 3), activation='sigmoid'))

*# Thêm Max pooling layer*

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

*# Flatten layer chuyển từ tensor sang vector*

model.add(Flatten())

*# Thêm Fully Connected layer với 128 nodes và dùng hàm sigmoid*

model.add(Dense(128, activation='sigmoid'))

*# Output layer với 10 node và dùng softmax function để chuyển sang xác suất.*

model.add(Dense(10, activation='softmax'))

#### *# 6. Compile model, chỉ rõ hàm loss\_function nào được sử dụng, phương thức*

*# dùng để tối ưu hàm loss function.*

model.compile(loss='categorical\_crossentropy',

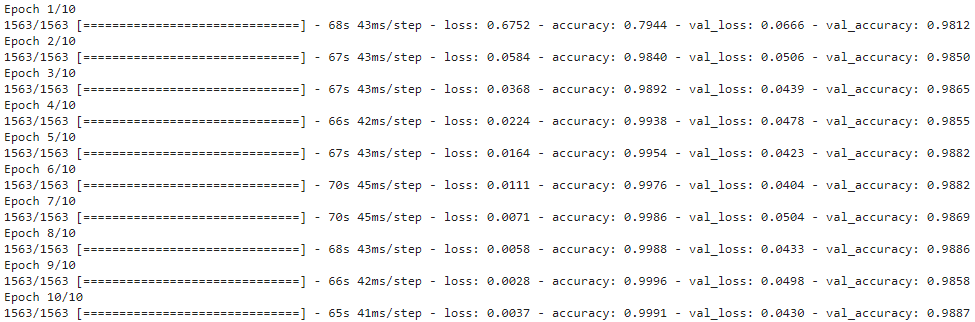
optimizer='adam',

metrics=['accuracy'])

#### *# 7. Thực hiện train model với data*

H = model.fit(X\_train, Y\_train, validation\_data=(X\_val, Y\_val), batch\_size=32, epochs=10, verbose=1)

**OUT:**



## **1.3.3. Mô tả phương pháp đánh giá**

*# 8. Đánh giá model với dữ liệu test set*

score = model.evaluate(X\_test, Y\_test, verbose=0)

print(score)

**OUT:**

**[0.03442202161136956, 0.9892]**

Ta sẽ dùng kết quả đánh giá của mode với **Test set** để làm kết quả cuối cùng của model. Tức model của chúng ta dự đoán chữ số có độ chính xác 98.92% với MNIST dataset cũng có nghĩa là dự đoán khoảng 100 ảnh thì sai 1 ảnh.

Thử dự đoán 1 ảnh từ tập dữ liệu MNIST

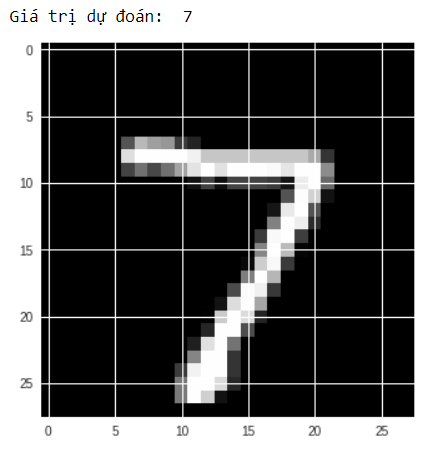
*# 9. Dự đoán ảnh*

plt.imshow(X\_test[0].reshape(28,28), cmap='gray')

y\_predict = model.predict(X\_test[0].reshape(1,28,28,1))

print('Giá trị dự đoán: ', np.argmax(y\_predict))

**OUT:**

****

Vậy dự đoán đúng giá trị X\_test[0] là chữ số 7.

# **1.4.** **Kết luận**

Chúng ta đã hoàn thành bài toán phân lớp từ load data, tạo model và tiến hành huấn luyện mô hình CNN cho tập dữ liệu MNIST với độ chính xác là 98.92 % rất cao . Chính vì thế mà ngày nay việc áp dụng kỹ thuật này đang trở nên vô cùng phổ biến cũng như tầm quan trọng của nó

* Chuẩn đoán ảnh X-ray của bệnh nhân có bị ung thư hay không
* Phân loại, nhận diện được các chữ, số viết tay => tự động đọc được biển số xe, văn bản.
* Phân loại được các biển báo giao thông => hỗ trợ cho ô tô tự lái
* …