HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

**KHOA CÔNG NGHỆ THÔNG TIN**

******

**BÁO CÁO**

***XỬ LÝ ẢNH***

**NHÓM MÔN HỌC: Nhóm 04**

**Giảng viên: Đinh Thị Hương**

**Sinh viên: Nguyễn Xuân Lụu**

**Mã số sinh viên: B18DCCN386**

**Hà Nội, Năm 2020**

Mục Lục

[1. Bài toán 1](#_Toc73538104)

[2. Cách tiếp cận 1](#_Toc73538105)

[2.1 Chuẩn bị dữ liệu 1](#_Toc73538106)

[2.2. Xây dựng model 1](#_Toc73538107)

[2.3. Xây dựng loss function: 3](#_Toc73538108)

[3. Cài đặt model sử dụng thư viện keras: 4](#_Toc73538109)

[1. Thêm các thư viện cần thiết: 4](#_Toc73538110)

[2. Load dữ liệu MNIST 5](#_Toc73538111)

[3. Reshape lại dữ liệu 5](#_Toc73538112)

[4. One hot encoding label (Chuyển các nhãn thành vector) 5](#_Toc73538113)

[5. Định nghĩa model 6](#_Toc73538114)

[6. Xác định loss function 7](#_Toc73538115)

[7. Thực hiện trên model với dữ liệu 8](#_Toc73538116)

[8. Vẽ đồ thị 8](#_Toc73538117)

[9. Đánh giá với test set 9](#_Toc73538118)

[10. Lưu model và xây dựng demo nhận diện chữ số trên web 10](#_Toc73538119)

[4. Tài liệu tham khảo 11](#_Toc73538120)

[5. Phụ lục 11](#_Toc73538121)

# 1. Bài toán

Bài toán phân loại ảnh giúp nhận diện chữ số viết tay. Đầu vào là bức ảnh xám có kích thước 28\*28 của các chữ số từ 0 đến 9.

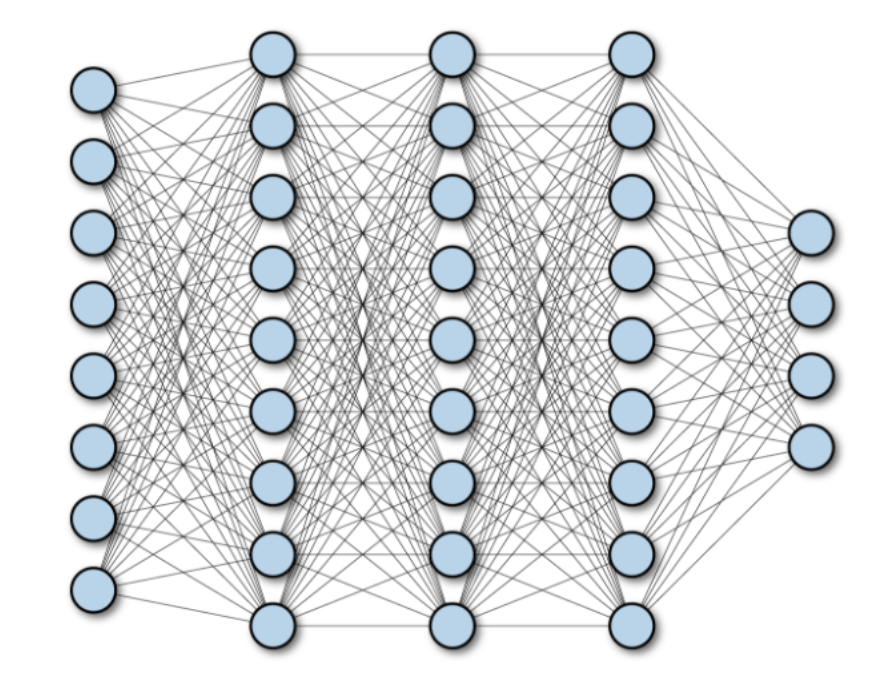
# 2. Cách tiếp cận

## 2.1 Chuẩn bị dữ liệu

Sử dụng tập dữ liệu có sẵn MNIST là tập dữ liệu từ 0 đến 9. Trong đó mỗi hình là một ảnh đen trắng chứa một chữ số được viết tay có kích thước 28x28. Bộ dataset có 60000 ảnh training và 10000 ảnh test

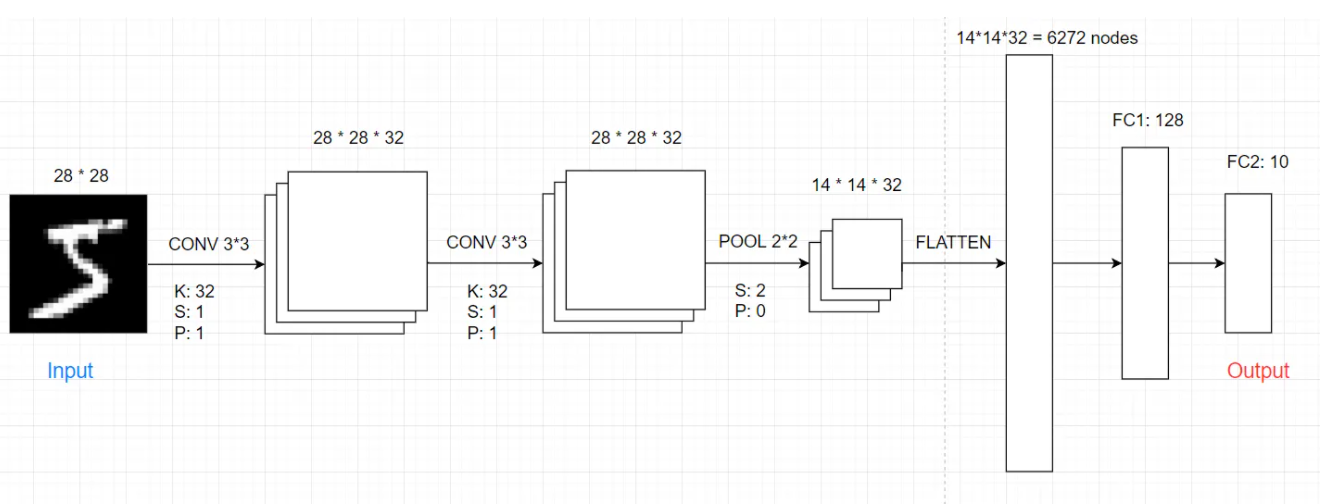
## 2.2. Xây dựng model

Sử dụng CNN (Mạng nơ-ron tích chập):

*Tại sao lại sử dụng CNN:*

Với các mạng **fully connected neural network** mỗi node trong 1 hidden layer được kết nối với tất cả các node của layer trước. Do đó với bài toán xử lý ảnh, ví dụ 1 bức ảnh màu kích thước 64 \* 64 -> Input layer cần 64 \* 64 \* 3 = 12288 node. Do đó số lượng weight hay số lượng kết nối giữa các layer rất lớn.

Chính vì vậy ta sử dụng CNN. Mỗi một node ở input layer sẽ chỉ kết đến một vài node ở hidden layer. Thường là convolution layer, sử dụng các mặt nạ để trích rút ra các đặc trưng của ảnh. Hay các phép tính nhân tích chập ta sẽ xử lý được vấn đề số lượng lớn các tham số.

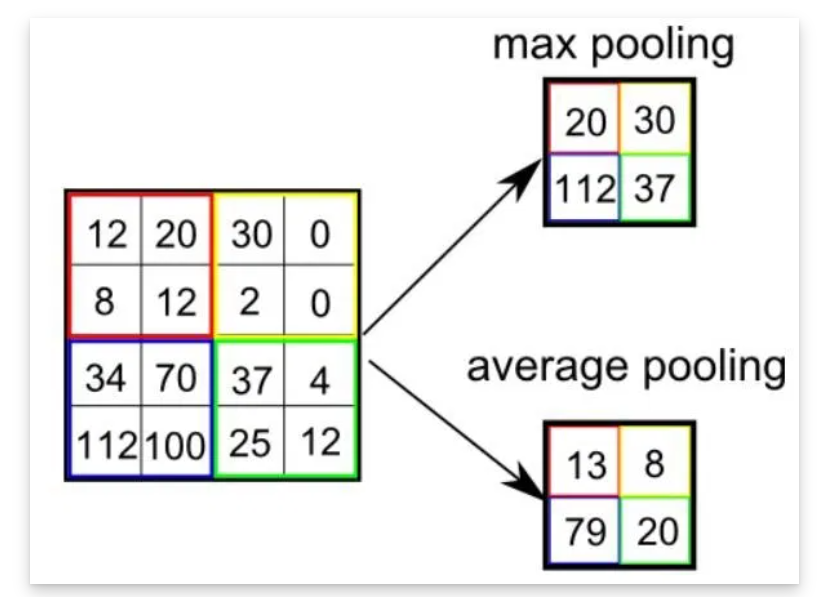
**Model**:

*Convolutional layer:*

Các convolutional layer có các parameter(kernel, cửa sổ, mặt nạ) dùng để lấy ra các đặc trưng của ảnh.

*Pooling layer:*

Giúp giảm kích thước dữ liệu nhưng vẫn giữ được các thuộc tính quan trọng. Có 2 loại pooling layer phổ biến đó là max pooling và average pooling. Ví dụ với pooling layer với kích thước K\*K.

* Max pooling: Trên mỗi vùng có kích thước K\*K của ma trận đầu vào ta tìm giá trị lớn nhất rồi điền vào ma trận kết quả
* Average pooling: Trên mỗi vùng có kích thước K\*K của ma trận đầu vào ta tìm giá trị trung bình rồi điền vào ma trận kết quả

Với pooling layer có kích thước 2 \* 2. Khi đó output width và height của dữ liệu sẽ giảm đi một nửa.

*Fully connected layer:*

Kết hợp các đặc điểm của ảnh để ra được output của model

## 2.3. Xây dựng loss function:

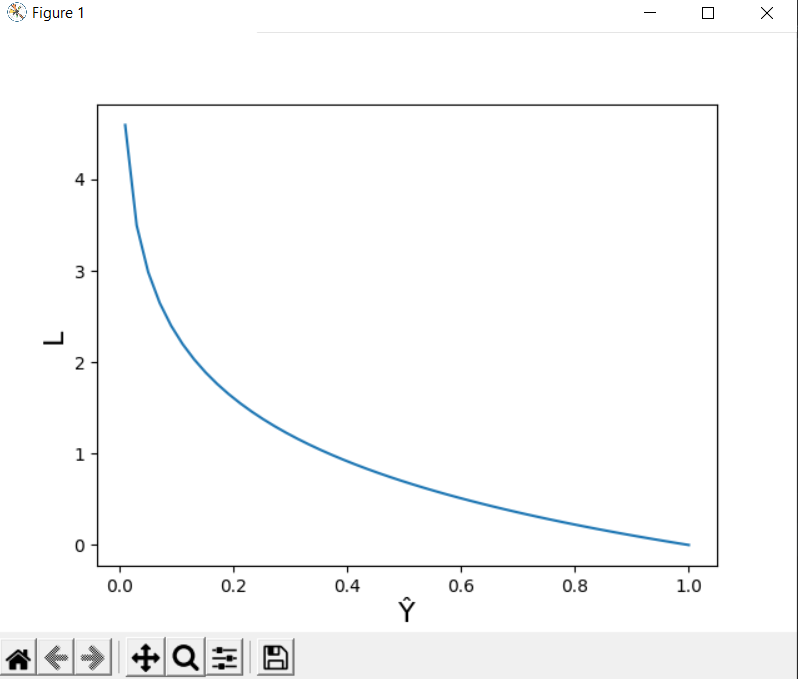
Là hàm xác định độ chênh lệch giữa giá trị thật và giá trị dự đoán, dùng để đánh giá độ tốt của model.

Khi đánh giá sự khác nhau (hay khoảng cách) giữa hai phân bố xác suất (probability distributions), chúng ta có một đại lượng đo có tên là **cross entropy**.

Với yi là giá phân bố thực chữ số (i-1) còn là phân bố dự đoán được

Đánh giá hàm L:

Giả sử ảnh là số 5 thì



* Khi modul dự đoán gần 1, tức là gần với giá trị thật y6 thì L nhỏ, xấp xỉ 0
* Ngược lại khi modul dự đoán gần 0, tức là khác xa so với giá trị thật thì L rất lớn

# 3. Cài đặt model sử dụng thư viện keras:

## 1. Thêm các thư viện cần thiết:

Numpy:

Matplotlib: vẽ đồ thị

Keras: Thư viện neural network, Chỉ cần định nghĩa model và loss function,  framework sẽ lo phần backpropagation(Lan truyền ngược)

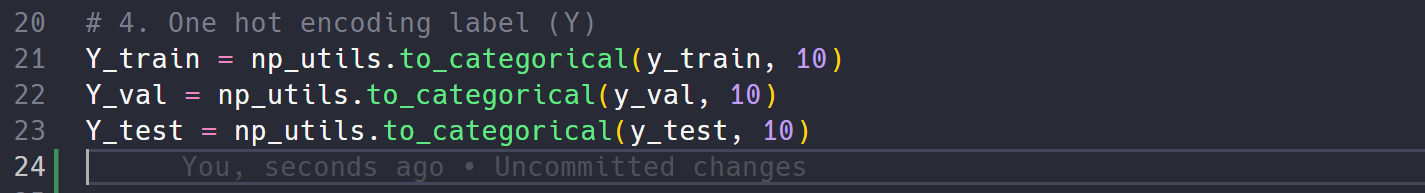
## 2. Load dữ liệu MNIST

Load dữ liệu từ MNIST dataset, bao gồm 60.000 training set và 10.000 test set. Sau đó chia bộ training set thành 2: 50.000 cho training set và 10.000 dữ liệu cho validation set. Validation cũng có dữ liệu giống với training set, nhưng mô hình không nhìn thấy nhãn. Mô hình chỉ đơn thuần dùng dữ liệu đầu vào của validation set để tính toán ra output. Sau đó nó so sánh với nhãn để tính loss function. Parameter hoàn toàn không được điều chỉnh ở bước này.

## 3. Reshape lại dữ liệu

Do dữ liệu input cho mô hình convolutional neural network là 1 tensor 4 chiều (N, W, H, D). Trong đó N là số lượng ảnh mỗi lần training , W,H là kích thước W=H=28, Do đầu vào là ảnh xám nên D=1. Ban đầu khi load dữ liệu ta chỉ có (N, W, H) nên phải reshape lại (N, W, H, D) để giống với dữ liệu keras yêu cầu

## 4. One hot encoding label (Chuyển các nhãn thành vector)

******

Ví dụ: label Y của ảnh ví dụ số 5 thành vector [0, 0, 0, 0, 0, 1, 0, 0, 0, 0]

## 5. Định nghĩa model

1. Model = Sequential() xếp các layer lên nhau để tạo model.
2. Ở layer đầu tiên cần chỉ rõ input\_shape của ảnh, input\_shape = (W, H, D), ta dùng ảnh xám kích thước (28,28) nên input\_shape = (28, 28, 1)
3. Khi thêm Convolutional Layer ta cần chỉ rõ các tham số: K (số lượng layer), kernel size (W, H), hàm activation sử dụng. Convolutional layer 2 để trích rút các đặc trưng trừ tượng hơn của bức ảnh
4. Khi thêm Maxpooling Layer cần chỉ rõ size của kernel.. Lớp pooling gom 1 tập các điểm ảnh vào và cho kích thước nhỏ hơn
5. Bước Flatten chuyển từ tensor sang vector chỉ cần thêm flatten layer.
6. Thêm Fully Connected Layer (FC) cần chỉ rõ số lượng node trong layer và hàm activation sử dụng trong layer,

Hàm softmax thay vì muốn kết quả hiển thị là số mấy, ta muốn dự đoán phần trăm hay xác suất dự đoán từng chữ số.

* ak là giá trị sau hàm activation
* zk là giá trị của hàm tổng

Nhận xét:

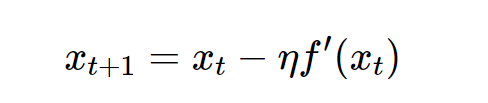
Do đó có thể coi ai là xác suất ảnh là số (i-1)

## 6. Xác định loss function

* Categorical\_crossentropy trong keras là hàm cross entropy
* Hàm tối ưu: Thuật toán để cải thiện weight và bias theo từng bước

Sử dụng thuật toán **adam**. Thường được dùng là hàm tối ưu trong các mô hình. Là sự kết hợp của 2 thuật toán **Momentum** và **RMSprop**. Về cơ bản vẫn dựa trên Gradient Descent.

Bản chất thì chúng ta sẽ đi tìm giá trị nhỏ nhất của hàm loss (gọi chung là f(x)).

 Gradient descent là thuật toán tìm giá trị nhỏ nhất của hàm số f(x) dựa trên đạo hàm

* được gọi là learning\_rate

Ta sẽ lặp lại tính x và thay vào f(x) cho đến khi f(x) đạt giá trị đủ nhỏ.

Với **Momentum** thì chúng ta sẽ cải thiện được hiện tượng Gradient Descent không tiến được tới điểm global minimum mà chỉ dừng lại ở local minimum.

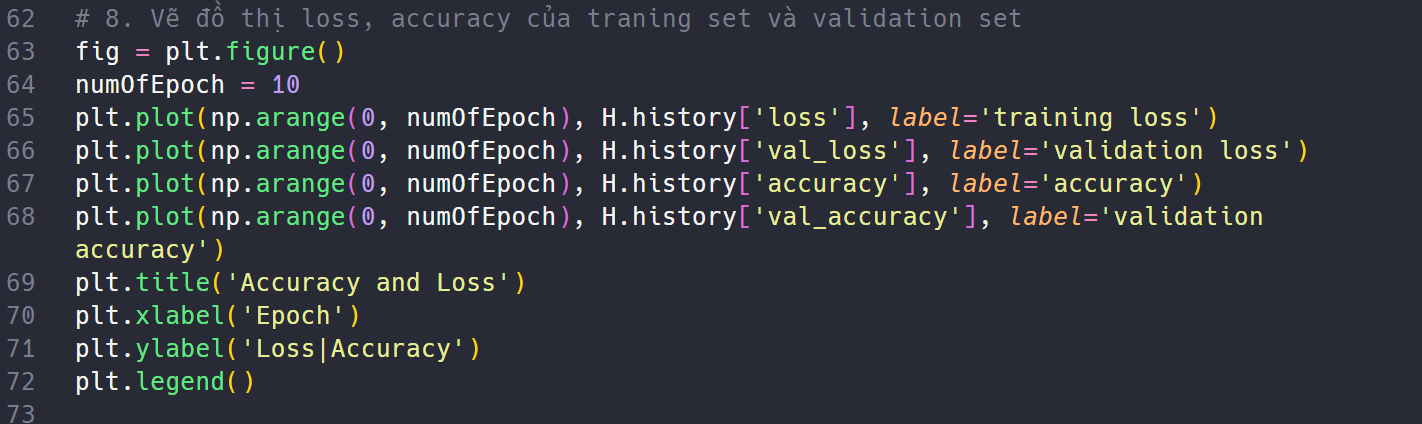
Còn với **RMSprop** learning rate cũng thay đổi giống như 1 tham số

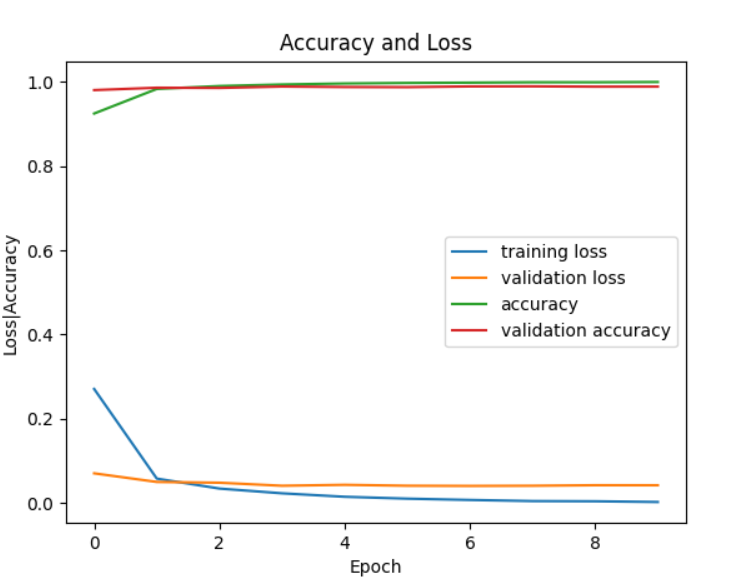
**Adam** là sự kết hợp của 2 thuật toán trên

## 7. Thực hiện trên model với dữ liệu

* Batch\_size: Là số mẫu trong một batch hay sử dụng cho một lần cập nhật trọng số.
* Epochs: Là số lần duyệt qua hết các mẫu trong tập huấn luyện. Với Gradient Descent đòi hỏi chúng ta phải đem toàn bộ dữ liệu qua mạng một vài lần để tìm được kết quả tối ưu.
* Verbose: Hiển thị ra màn hình

## 8. Vẽ đồ thị



Đánh giá model:

Khi epoch tăng thì training loss giảm và validation loss có tăng nên model vẫn xảy ra hiện tượng overfitting

## 9. Đánh giá với test set

Model dự đoán chữ số có độ chính xác 98,86% với MNIST dataset

## 10. Lưu model và xây dựng demo nhận diện chữ số trên web

# 4. Tài liệu tham khảo

|  |  |
| --- | --- |
| [1] | "Bài 7: Giới thiệu keras và bài toán phân loại ảnh.," [Online]. Available: https://nttuan8.com/bai-7-gioi-thieu-keras-va-bai-toan-phan-loai-anh/. |
| [2] | "Thuật toán CNN – Convolutional Neural Network," [Online]. Available: https://topdev.vn/blog/thuat-toan-cnn-convolutional-neural-network/. |
| [3] | "Underfitting and Overfitting in Machine Learning," [Online]. Available: https://datascience.foundation/sciencewhitepaper/underfitting-and-overfitting-in-machine-learning. |
| [4] | "An overview of gradient descent optimization algorithms," [Online]. Available: https://ruder.io/optimizing-gradient-descent/index.html#stochasticgradientdescent. |
| [5] | "Keras," [Online]. Available: https://keras.io/. |
| [6] | "machinelearningcoban," [Online]. Available: https://machinelearningcoban.com/. |

**5. Phụ lục**

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

# 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\_test.shape[0])

# 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)

print(X\_train.shape)

# 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])

# 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 xuất.

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

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

# đùng 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)

# 8. Vẽ đồ thị loss, accuracy của traning set và validation set

fig = plt.figure()

numOfEpoch = 10

plt.plot(np.arange(0, numOfEpoch), H.history['loss'], label='training loss')

plt.plot(np.arange(0, numOfEpoch), H.history['val\_loss'], label='validation loss')

plt.plot(np.arange(0, numOfEpoch), H.history['accuracy'], label='accuracy')

plt.plot(np.arange(0, numOfEpoch), H.history['val\_accuracy'], label='validation accuracy')

plt.title('Accuracy and Loss')

plt.xlabel('Epoch')

plt.ylabel('Loss|Accuracy')

plt.legend()

# 9. Đánh giá model với dữ liệu test set

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

print(score)

print(X\_test[0]);

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

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

plt.show()



**Link demo:** [**https://nxluu.github.io/mnist-dataset-demo/**](https://nxluu.github.io/mnist-dataset-demo/)