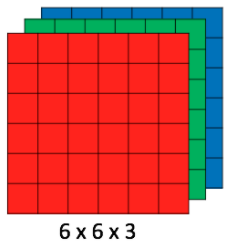
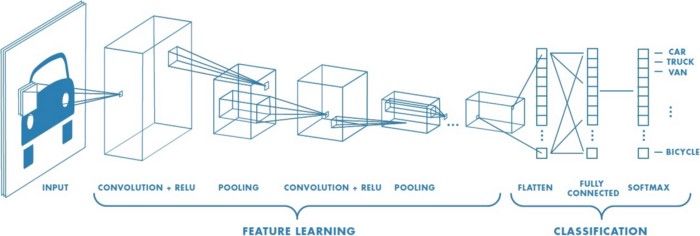
# A- LÝ THUYẾT

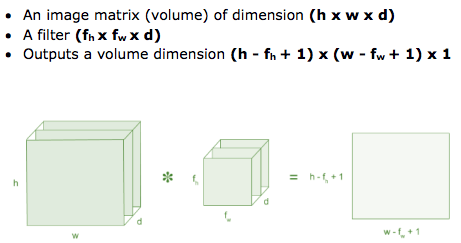
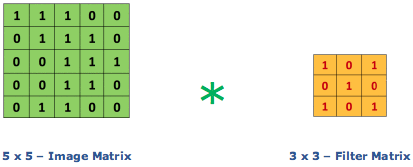
# 1. Giới thiệu

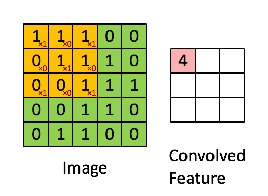
* Trong mạng neural, mô hình mạng neural tích chập (CNN) là 1 trong những mô hình để nhận dạng và phân loại hình ảnh. Trong đó, xác định đối tượng và nhận dạng khuôn mặt là 1 trong số những lĩnh vực mà CNN được sử dụng rộng rãi.
* CNN phân loại hình ảnh bằng cách lấy 1 hình ảnh đầu vào, xử lý và phân loại nó theo các hạng mục nhất định (Ví dụ: Chó, Mèo, Hổ, ...). Máy tính coi hình ảnh đầu vào là 1 mảng pixel và nó phụ thuộc vào độ phân giải của hình ảnh. Dựa trên độ phân giải hình ảnh, máy tính sẽ thấy H x W x D (H: Chiều cao, W: Chiều rộng, D: Độ dày). Ví dụ: Hình ảnh là mảng ma trận RGB 6x6x3 (3 ở đây là giá trị RGB).

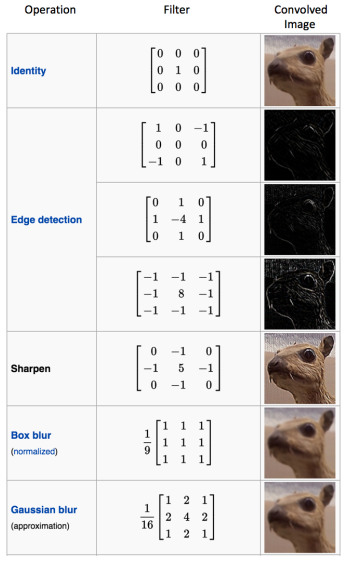


* Về kỹ thuật, mô hình CNN để training và kiểm tra, mỗi hình ảnh đầu vào sẽ chuyển nó qua 1 loạt các lớp tích chập với các bộ lọc (Kernals), tổng hợp lại các lớp được kết nối đầy đủ (Full Connected) và áp dụng hàm Softmax để phân loại đối tượng có giá trị xác suất giữa 0 và 1. Hình dưới đây là toàn bộ luồng CNN để xử lý hình ảnh đầu vào và phân loại các đối tượng dựa trên giá trị.

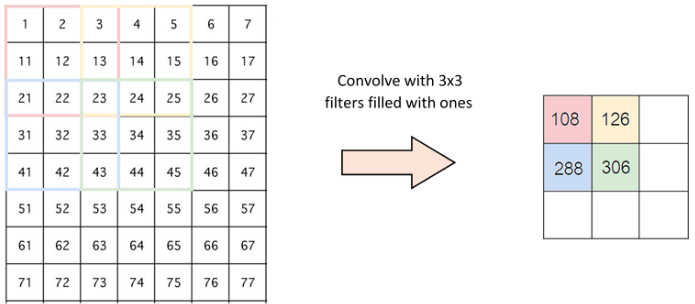
# 2. Lớp tích chập - Convolution Layer

* 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.
* Xem xét 1 ma trận 5 x 5 có giá trị pixel là 0 và 1. Ma trận bộ lọc 3 x 3 như hình bên dưới.
* Sau đó, lớp tích chập của ma trận hình ảnh 5 x 5 nhân với ma trận bộ lọc 3 x 3 gọi là 'Feature Map' như hình bên dưới.



* Sự kết hợp của 1 hình ảnh với các bộ lọc khác nhau có thể thực hiện các hoạt động như phát hiện cạnh, làm mờ và làm sắc nét bằng cách áp dụng các bộ lọc. Ví dụ dưới đây cho thấy hình ảnh tích chập khác nhau sau khi áp dụng các Kernel khác nhau.

# 3. Bước nhảy - Stride

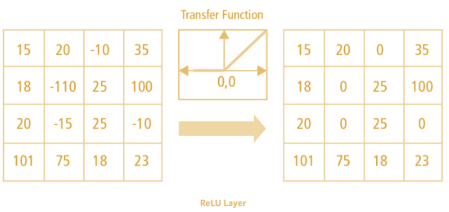
* Stride là số pixel thay đổi trên ma trận đầu vào. Khi stride là 1 thì ta di chuyển các kernel 1 pixel. Khi stride là 2 thì ta di chuyển các kernel đi 2 pixel và tiếp tục như vậy. Hình dưới là lớp tích chập hoạt động với stride là 2. 

# 4. Đường viền - Padding

* Đôi khi kernel không phù hợp với hình ảnh đầu vào. Ta có 2 lựa chọn:
* Chèn thêm các số 0 vào 4 đường biên của hình ảnh (padding).
* Cắt bớt hình ảnh tại những điểm không phù hợp với kernel.

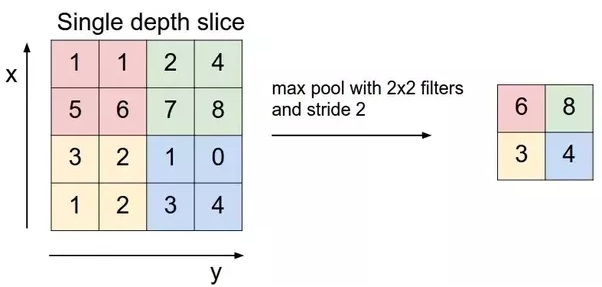
# 5. Hàm phi tuyến - ReLU

* ReLU viết tắt của Rectified Linear Unit, là 1 hàm phi tuyến. Với đầu ra là: ƒ (x) = max (0, x).
* Tại sao ReLU lại quan trọng: ReLU giới thiệu tính phi tuyến trong ConvNet. Vì dữ liệu trong thế giới mà chúng ta tìm hiểu là các giá trị tuyến tính không âm.



* Có 1 số hà phi tuyến khác như tanh, sigmoid cũng có thể được sử dụng thay cho ReLU. Hầu hết người ta thường dùng ReLU vì nó có hiệu suất tốt.

# 6. Lớp gộp - Pooling Layer

* Lớp pooling sẽ giảm bớt số lượng tham số khi hình ảnh quá lớn. Không gian pooling còn được gọi là lấy mẫu con hoặc lấy mẫu xuống làm giảm kích thước của mỗi map nhưng vẫn giữ lại thông tin quan trọng. Các pooling có thể có nhiều loại khác nhau:
* Max Pooling
* Average Pooling
* Sum Pooling
* Max pooling lấy phần tử lớn nhất từ ma trận đối tượng, hoặc lấy tổng trung bình. Tổng tất cả các phần tử trong map gọi là sum pooling 

# 7. Tóm tắt

* Đầu vào của lớp tích chập là hình ảnh
* Chọn đối số, áp dụng các bộ lọc với các bước nhảy, padding nếu cần. Thực hiện tích chập cho hình ảnh và áp dụng hàm kích hoạt ReLU cho ma trận hình ảnh.
* Thực hiện Pooling để giảm kích thước cho hình ảnh.
* Thêm nhiều lớp tích chập sao cho phù hợp
* Xây dựng đầu ra và dữ liệu đầu vào thành 1 lớp được kết nối đầy đủ (Full Connected)
* Sử dụng hàm kích hoạt để tìm đối số phù hợp và phân loại hình ảnh.

**B- HIỆN THỰC**

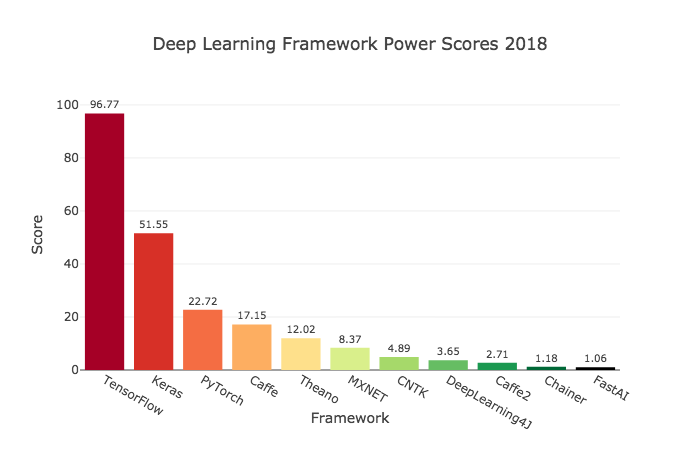
1. **Giới thiệu về keras**

Bản chất của bài toán Deep learning: Bạn có dữ liệu, bạn muốn máy tính học được các mô hình (model) từ dữ liệu, sau đó dùng mô hình đấy để dự đoán được các dữ liệu mới. Các bước cơ bản làm một bài toán deep learning :

* Xây dựng bài toán
* Chuẩn bị dữ liệu (dataset)
* Xây dựng model
* Định nghĩa loss function
* Thực hiện backpropagation và áp dụng gradient descent để tìm các parameter gồm weight và bias để tối ưu loss function.
* Dự đoán dữ liệu mới bằng model với các hệ số tìm được ở trên

Bước xây dựng model [neural network](https://nttuan8.com/bai-3-neural-network/) và convolutional neural network ta có thể xây dựng model hoàn chỉnh từ đầu bằng python. Tuy nhiên bước backpropagation trở nên phức tạp hơn rất rất nhiều. Khó để implement và tối ưu được tốc độ tính toán. Đấy là lý do các framework về deep learning ra đời với các đặc điểm:

* Người dùng chỉ cần định nghĩa model và loss function, framework sẽ lo phần backpropagation.
* Việc định nghĩa layer, activation function, loss function đơn giản hơn cho người dùng. Ví dụ để thêm layer trong neural network chỉ cần báo là layer có bao nhiêu node và dùng hàm activation gì.
* Tối ưu việc tính toán trên CPU và GPU.



Hình 1. Các deep learning framework phổ biến

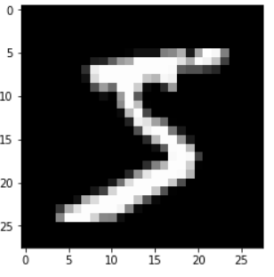
Có thể thấy tensorflow là framework phổ biến nhất tuy nhiên tensorflow khá khó sử dụng cho người mới bắt đầu.

Keras là một framework mã nguồn mở cho deep learning được viết bằng Python. Nó có thể chạy trên nền của các deep learning framework khác như: tensorflow, theano, CNTK. Với các API bậc cao, dễ sử dụng, dễ mở rộng, keras giúp người dùng xây dựng các deep learning model một cách đơn giản.

1. **MNIST Dataset**

### **Xây dựng bài toán**

Bạn có ảnh xám kích thước 28\*28 của chữ số từ 1 đến 9 và bạn muốn dự đoán số đấy là số mấy. Ví dụ:



Hình 2. Dữ liệu đầu tiên trong MNIST dataset.

### **Chuẩn bị dữ liệu**

[MNIST](http://yann.lecun.com/exdb/mnist/) là bộ cơ sở dữ liệu về chữ số viết tay, 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.

1. **Training set, test set là gì?**

Giả sử bạn đang luyện thi đại học và bạn có 10 bộ đề để luyện thi. Nếu bạn học và chữa cả 10 đề một cách chi tiết thì bạn sẽ không thể ước lượng được điểm thi của bạn khoảng bao nhiêu, dẫn đến bạn không chọn được trường phù hợp. Thế là bạn nghĩ ra một giải pháp tốt hơn, trong 10 đề đấy chỉ lấy 8 đề học và chữa chi tiết thôi còn để 2 đề lại coi như là đề thi thật. Như vậy bạn có thể ước lượng điểm thi của mình bằng cách đánh giá điểm ở 2 đề đấy.

Trong ví dụ trên thì:

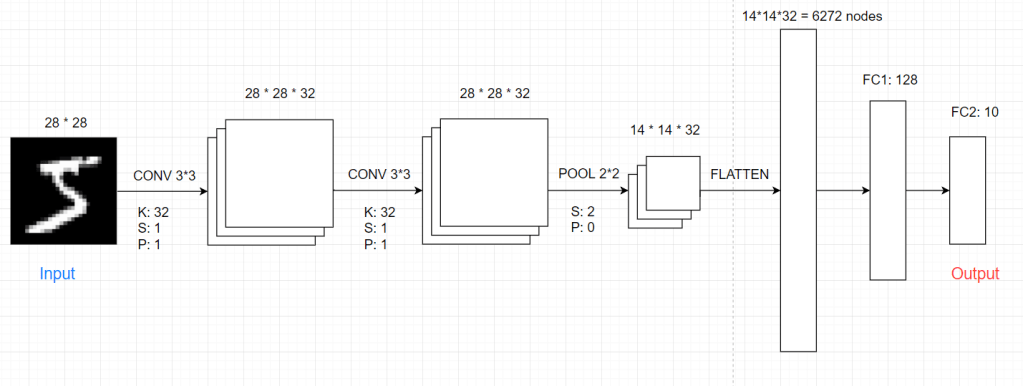
* 8 đề ôn luyện được gọi là **training set**, có thể hiểu là dữ liệu dùng để dạy cho model học.
* 2 đề để rành gọi là **validation set**, là để đánh giá xem model hiện tại có tốt không, thường được dùng để chỉnh các tham số của model.
* đề thi đại học thật là **test set**, là để đánh giá xem model hoạt 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.

### **Xây dựng model**

Vì input của model là ảnh nên nghĩ ngay đến convolutional neural network (CNN).

Mô hình chung bài toán CNN: Input image -> Convolutional layer (Conv) + Pooling layer (Pool) -> Fully connected layer (FC) -> Output.

Model cho bài toán

Input của model là ảnh xám kích thước 28\*28.

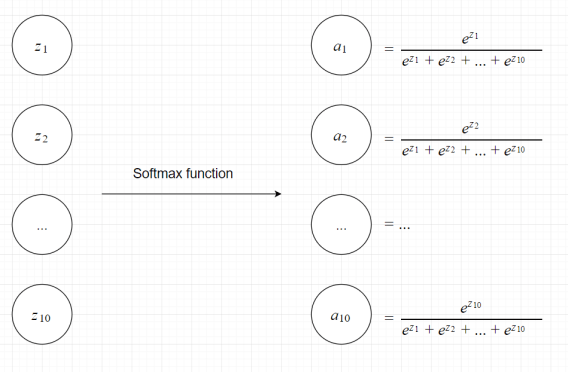
#### **Softmax function**

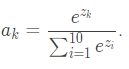
Giống như logistic regression, thay vì chỉ muốn kết quả là ảnh là số mấy, ta muốn dự đoán phần trăm của ảnh là số nào. Ví dụ: 90% ảnh là số 5, 1% ảnh là số 1,…

Ở mỗi layer sẽ thực hiện 2 bước: tính tổng linear các node ở layer trước và thực hiện activation function (ví dụ sigmoid function, softmax function). Do sau bước tính tổng linear cho ra các giá trị thực nên cần dùng **softmax function** dùng để chuyển đổi giá trị thực trong các node ở output layer sang giá trị phần trăm.

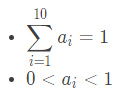
Vì mỗi ảnh sẽ thuộc 1 class từ 0 đến 9, nên tất cả sẽ có 10 class. Nên output layer sẽ có 10 node để tương ứng với phần trăm ảnh là số 0,1,..,9. Ví dụ: a\_6*a*6​ là xác xuất ảnh là số 5. (Sự khác biệt chỉ số do các số bắt đầu từ 0 trong khi chỉ số của node trong layer bắt đầu từ 1)

Hình 3. Softmax function

  
Tổng quát sau hàm activation:



Nhận xét:



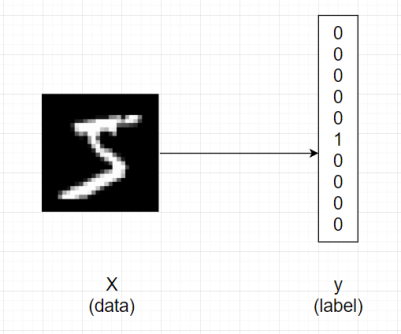
Do đó ta có thể coi *ai*​ là xác xuất ảnh là số (i-1).

**Với các bài toán classification (phân loại) thì nếu có 2 lớp thì hàm activation ở output layer là hàm sigmoid, còn nhiều hơn 2 lớp thì hàm activation ở ouput layer là hàm softmax**

=> Output layer có 10 nodes và activation là softmax function.

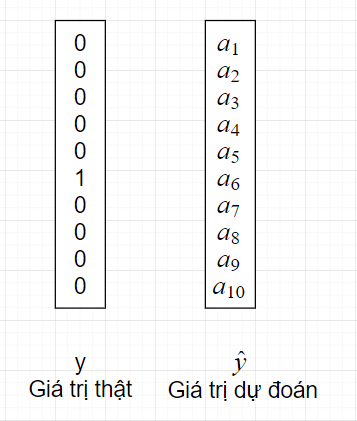
### **Loss function**

Để định nghĩa loss function, trước hết ta dùng one-hot encoding chuyển đổi label của ảnh từ giá trị số sang vector cùng kích thước với output của model. Ví dụ:

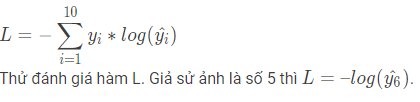


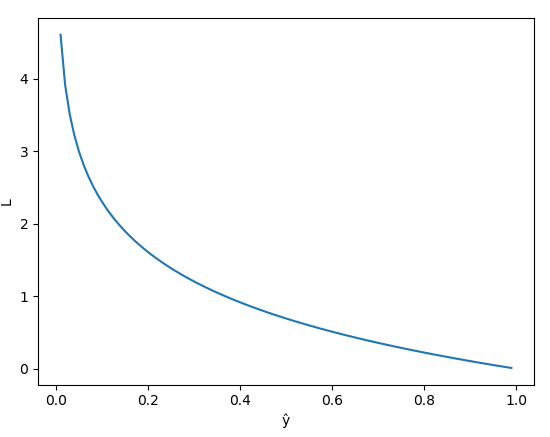
Để ý là label của data là số i là vector v kích thước 10\*1 với *vi*+1​=1 và các giá trị khác bằng 0. So với quy ước về phần trăm ở trên thì one-hot encoding có ý nghĩa là ta chắc chắn 100% ảnh này là số 5.

Giờ ta có giá trị thật (label) dạng one-hot encoding giá trị dự đoán ở output layer sau hàm softmax function cùng kích thước 10\*1. Ta cần định nghĩa hàm loss function để đánh giá độ tốt của model.



Mong muốn là *a*6​ gần 1 còn các giá trị a khác gần 0 vì như thế nghĩa là model dự đoán đúng được ảnh đầu vào là ảnh số 5. Ta định nghĩa loss function:





Nhận xét:

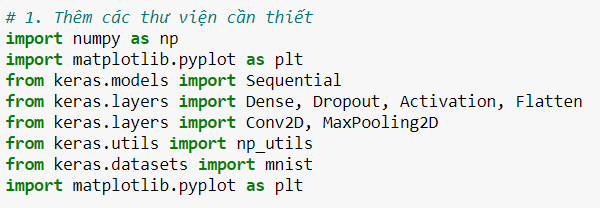
* Hàm L giảm dần từ 0 đến 1
* Khi model dự đoán  gần 1, tức giá trị dự đoán gần với giá trị thật *y*6​ thì L nhỏ, xấp xỉ 0
* Khi model dự đoán  gần 0, tức giá trị dự đoán ngược lại giá trị thật *y*6 thì L rất lớn

=> Hàm L nhỏ khi giá trị model dự đoán gần với giá trị thật và rất lớn khi model dự đoán sai, hay nói cách khác L càng nhỏ thì model dự đoán càng gần với giá trị thật. => Bài toán tìm model trở thành tìm giá trị nhỏ nhất của L.

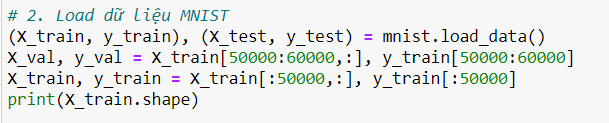
Hàm loss function định nghĩa như trên trong keras gọi là “**categorical\_crossentropy**“

1. **Ứng dụng keras MNIST Dataset – Nhận dạng số viết tay**

a. Sử dụng các thư viện

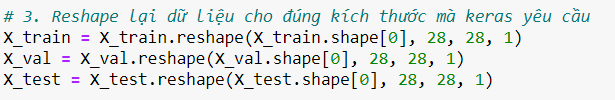


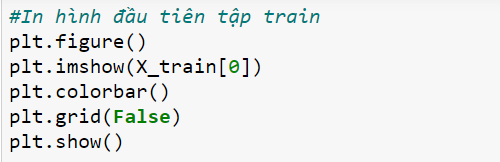
b. Load dữ liệu từ MNIST dataset, bao gồm 60.000 training set và 10.000 test set. Sau đó chia bộ traning set thành 2: 50.000 cho training set và 10.000 dữ liệu cho validation set.



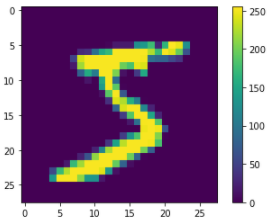
Kết quả: (50000, 28, 28)

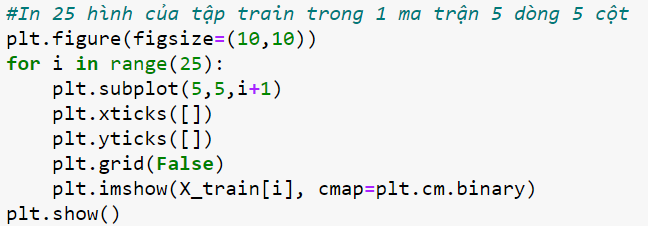
Dữ liệu input cho mô hình convolutional neural network là 1 tensor 4 chiều (N, W, H, D), trong bài này là ảnh xám nên W = H = 28, D = 1, N là số lượng ảnh cho mỗi lần training. Do dữ liệu ảnh ở trên có kích thước là (N, 28, 28) tức là (N, W, H) nên rần reshape lại thành kích thước N *28*28 \* 1 để giống kích thước mà keras yêu cầu.



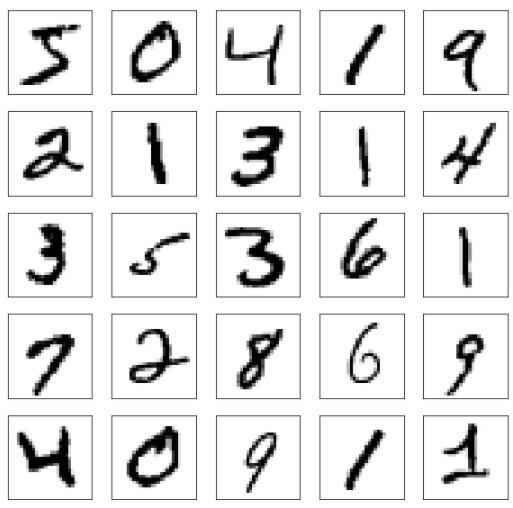


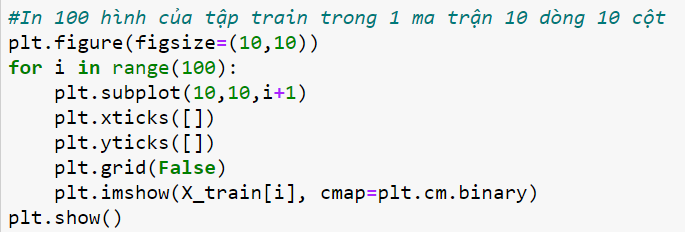
Kết quả:



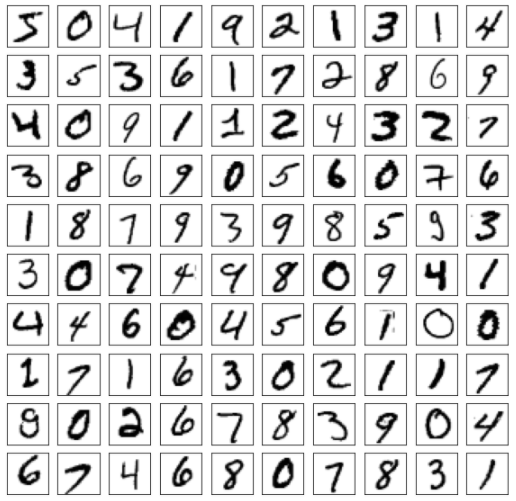


Kết quả:

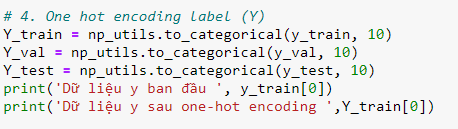




Kết quả:



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]

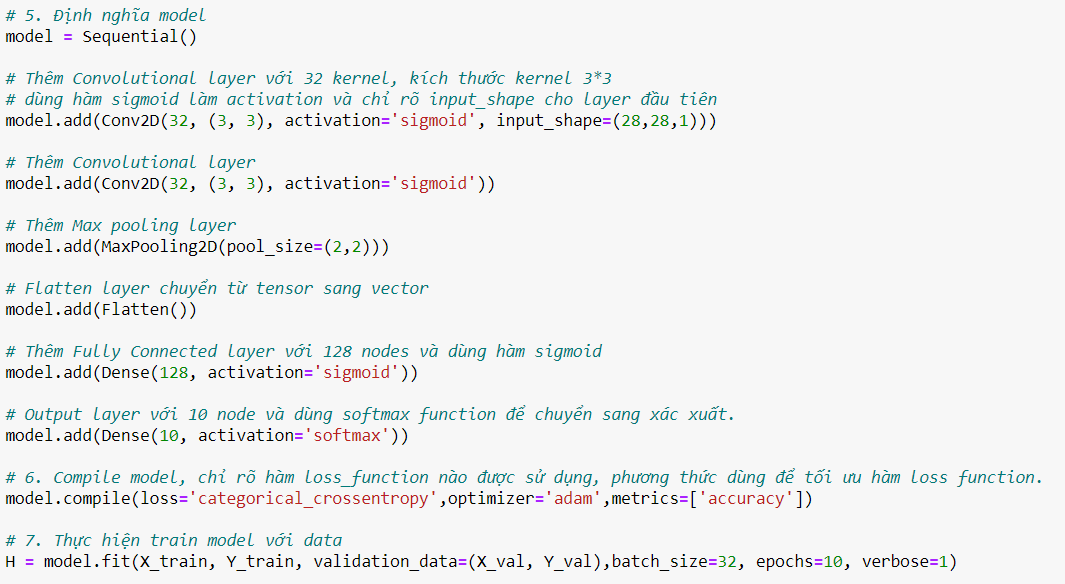


Dữ liệu y ban đầu 5

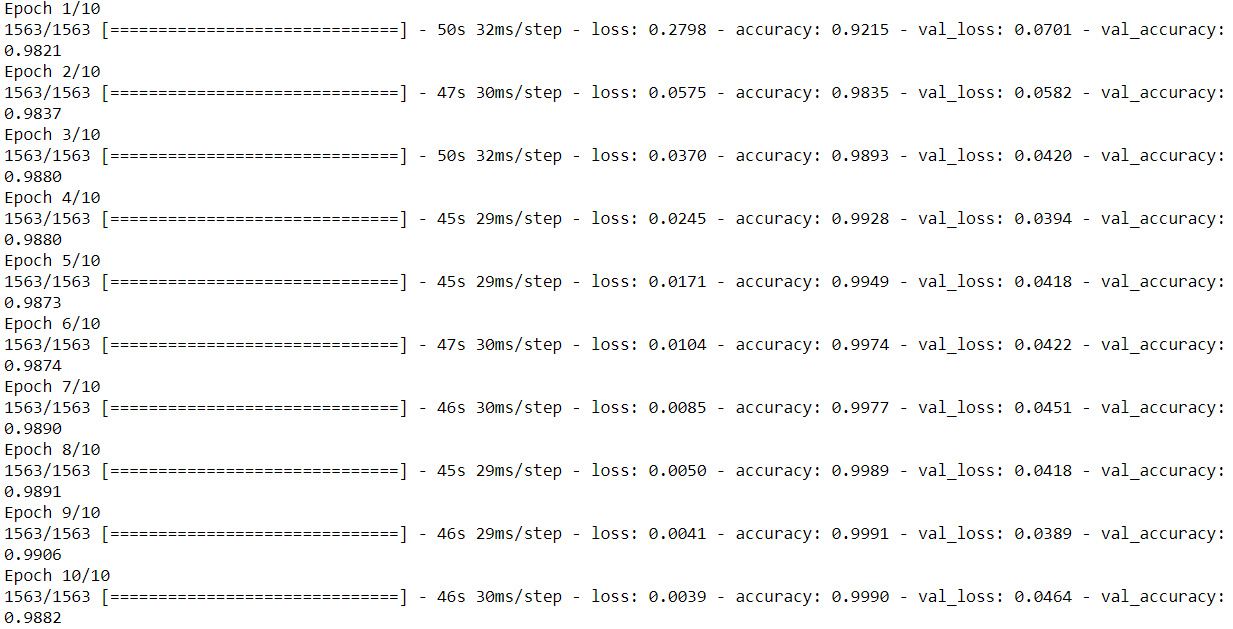
Dữ liệu y sau one-hot encoding [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]

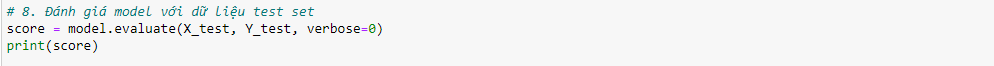
Bước tiếp theo định nghĩa model:

1. Model = Sequential() để nói cho keras là ta sẽ xếp các layer lên nhau để tạo model. Ví dụ input -> CONV -> POOL -> CONV -> POOL -> FLATTEN -> FC -> OUTPUT
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. cấu trúc: model.add(Conv2D(K, (W, H), activation='tên\_hàm\_activation'))
4. Khi thêm Maxpooling Layer cần chỉ rõ size của kernel, model.add(MaxPooling2D(pool\_size=(W, H)))
5. Bước Flatten chuyển từ tensor sang vector chỉ cần thêm flatten layer.
6. Để thêm Fully Connected Layer (FC) cần chỉ rõ số lượng node trong layer và hàm activation sử dụng trong layer, cấu trúc: model.add(Dense(số\_lượng\_node activation='tên\_hàm activation'))



Kết quả:





[0.03272830322384834, 0.9897000193595886]

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 dữ đoán chữ số có độ chính xác 98.97% với MNIST dataset. Nghĩa là dự đoán khoảng 100 ảnh thì sai 1 ảnh.

