

TRƯỜNG ĐẠI HỌC VINH
VIỆN KỸ THUẬT VÀ CÔNG NGHỆ



ĐỒ ÁN KỸ THUẬT LẬP TRÌNH
NGÀNH KỸ THUẬT ĐIỆN TỬ TRUYỀN THÔNG

Đề tài:

NHẬN DẠNG MỘT KÍ TỰ VIẾT TAY IN HOA BẰNG PYTHON

Giao viên hướng dẫn: TS. Mai Thế Anh

Sinh viên thực hiện : Đình Trọng Chiến

Mssv : 155D5202070008

Lớp : 56K - ĐTTT

Khóa : 2015 - 2020

NGHỆ AN – 2021

LỜI CAM ĐOAN	3
CHƯƠNG 1:TỔNG QUAN	2
I. GIỚI THIỆU ĐỒ ÁN	2
1. Lời nói đầu:	2
2. Cơ sở lý thuyết:	2
2.1. AI (Artificial Intelligence)	2
2.2. MC (Machine Learning)	2
2.3. Convolutional Neural Network (CNNs – Mạng nơ-ron tích chập)	2
3. Đồ án nhận dạng chữ viết tay in hoa	2
3. 1. Phân loại:	2
3. 2. Nhiệm vụ đồ án	3
II. CẤU TRÚC	3
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VÀ CƠ CẤU	3
I. Phần mềm làm đồ án : Pycharm	3
II. Tìm hiểu các thư viện cần thiết và quan trọng nhất cho đồ án	3
III. CƠ SỞ HOẠT ĐỘNG	4
1. Cấu trúc của Keras	4
2. Mô hình CNN	5
3. CNN	6
IV. XÂY DỰNG ĐỒ ÁN NHẬN DẠNG CHỮ VIẾT TAY	7
1. Tìm hiểu và xử lý dữ liệu	7
2. Chuẩn bị dữ liệu	14
3. Xây dựng mô hình	17
4. TÓM LƯỢC MODEL	20
5. ĐÁNH GIÁ MÔ HÌNH	21
6. ÁP DỤNG	24

CHƯƠNG 3: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	32
TÀI LIỆU THAM KHẢO	33

CHƯƠNG 1: TỔNG QUAN

I. GIỚI THIỆU ĐỒ ÁN

1. Lời nói đầu:

Nhận dạng là bài toán xuất hiện cách đây khá lâu và vẫn luôn thu hút được nhiều sự quan tâm, nghiên cứu. Đặc biệt là trong vài thập niên gần đây, do sự thúc đẩy của quá trình tin học hoá trong mọi lĩnh vực, bài toán nhận dạng không còn dừng lại ở mức độ nghiên cứu nữa mà nó trở thành một lĩnh vực để áp dụng vào thực tế. Các bài toán nhận dạng đang được ứng dụng trong thực tế hiện nay tập trung vào nhận dạng mẫu, nhận dạng tiếng nói và nhận dạng chữ. Trong số này, nhận dạng chữ là bài toán được quan tâm rất nhiều và cũng đã đạt được nhiều thành tựu rực rỡ. Các ứng dụng có ý nghĩa thực tế lớn có thể kể đến như: nhận dạng chữ in dùng trong quá trình sao lưu sách báo trong thư viện, nhận dạng chữ viết tay dùng trong việc phân loại thư ở bưu điện, thanh toán tiền trong nhà băng và lập thư viện sách cho người mù (ứng dụng này có nghĩa: scan sách bình thường, sau đó cho máy tính nhận dạng và trả về dạng tài liệu mà người mù có thể đọc được).

2. Cơ sở lý thuyết:

2.1. AI (Artificial Intelligence)

Gọi là trí tuệ nhân tạo, giúp mô phỏng được suy nghĩ, khả năng học tập, cư xử, thích ứng ... của con người áp dụng cho máy móc và những hệ thống máy vi tính

2.2. MC (Machine Learning)

Là lĩnh vực của trí tuệ nhân tạo, liên quan đến việc nghiên cứu, và xây dựng các kĩ thuật cho phép các hệ thống “học” tự động từ dữ liệu để giải quyết những vấn đề cụ thể

2.3. Convolutional Neural Network (CNNs – Mạng nơ-ron tích chập)

Là một trong những mô hình Deep Learning tiên tiến. Hầu hết các bài toán liên quan đến nhận dạng và phân loại ảnh hiện nay đều sử dụng phương pháp này trong việc xây dựng model từ cơ bản cho đến phức tạp. Nó giúp cho chúng ta xây dựng được những hệ thống thông minh với độ chính xác cao như hiện nay. Như hệ thống xử lý ảnh lớn như Facebook, Google hay Amazon đã đưa vào sản phẩm của mình những chức năng thông minh như nhận diện khuôn mặt người dùng, phát triển xe hơi tự lái hay drone giao hàng tự động.

3. Đồ án nhận dạng chữ viết tay in hoa

3.1. Phân loại:

Nhận dạng chữ in: phục vụ cho công việc tự động hóa đọc tài liệu, tăng tốc độ và hiệu quả nhập thông tin vào máy tính trực tiếp từ các nguồn tài liệu

Nhận dạng chữ viết tay : với những mức độ ràng buộc khác nhau về cách viết, kiểu chữ,... phục vụ cho các ứng dụng đọc và xử lý chứng từ, hóa đơn, phiếu ghi,

bản thảo biết tay. Nhận dạng chữ viết được tách thành 2 hướng phát triển : online và offline

Tuy nhiên, trên thế giới cũng như Việt Nam, bài toán nhận diện chữ viết tay vẫn là thách thức lớn đối với các nhà nghiên cứu. Bài toán vẫn chưa giải quyết trọn vẹn vì phụ thuộc quá nhiều vào người viết và sự biến đổi quá đa dạng trong cách viết cũng như trạng thái tinh thần của từng người viết.

Đặc biệt đối với nghiên cứu nhận dạng chữ viết tiếng Việt càng gặp nhiều khó khăn hơn khi có thêm dấu, rất dễ nhầm lẫn với nhiều. Nhìn chung, các sản phẩm phần mềm nhận dạng chữ văn bản Tiếng Việt chữ in của nước ta đã đạt những kết quả khả quan, đặc biệt phần mềm VNDOCR (phần mềm nhận dạng chữ của Viện Công Nghệ Thông Tin Hà Nội) được sử dụng rộng rãi trong cơ quan nhà nước. Riêng nhận dạng chữ viết tay vẫn đang được nghiên cứu và phát triển.

3. 2. **Nhiệm vụ đồ án**

Nghiên cứu cách thức hoạt động và quá trình máy học, nghiên cứu về các thư viện quan trọng trong việc nhận biết, đặc biệt về TensorFlow và API Keras và hoàn thành demo nhận dạng 1 chữ cái in hoa.

II. CẤU TRÚC

Đồ án gồm 3 chương:

Chương 1: Giới thiệu tổng quan về đề tài đồ án, trình bày ngắn gọn lí do chọn đề tài, sự cần thiết của đề tài và tính ứng dụng thực tiễn

Chương 2: Trình bày cơ sở lý thuyết để thực hiện đồ án, từng thành phần cấu tạo nên đồ án

Chương 3: Kết luận, hoàn thành demo nhận dạng, tổng hợp lại đồ án, nêu ra thiếu sót và hướng phát triển đồ án trong tương lai

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VÀ CƠ CẤU

I. Phần mềm làm đồ án : Pycharm

Sử dụng môi trường : Python 3.7.6

Kiểm tra môi trường python : gõ R + Windows để mở hộp thoại R -> gõ cmd ->

go python : nếu xuất hiện

```
Python 3.7.6 (tags/v3.7.6:43364a7ae0, Dec 19 2019, 00:42:30) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> -
```

II. Tìm hiểu các thư viện cần thiết và quan trọng nhất cho đồ án (Còn 1 số thư viện khác)

1. Numpy

Là thư viện lõi phục vụ khoa học máy tính của Python, hỗ trợ việc tính toán các mảng nhiều chiều.

- Để cài đặt, gõ : pip install numpy vào phần Terminal và đợi cài đặt

- Để dùng thư viện : import numpy as np

2. TensorFlow

Là thư viện mã nguồn mở dùng cho tính toán số học sử dụng đồ thị luồng dữ liệu.

Nó tích hợp sẵn rất nhiều các thư viện Machine Learning và có khả năng tương

thích, mở rộng tốt. Được Google phát triển cho machine learning phục vụ cả nghiên cứu lẫn xây dựng ứng dụng thực tế. Với TensorFlow, Machine Learning trở nên gần gũi và dễ tiếp cận hơn với lập trình viên.

- Để cài đặt : gõ pip install tensorflow vào phần Terminal và đợi cài đặt
- Để sử dụng : import tensorflow as tf
- Lưu ý : phải tải bản tương thích với môi trường python đang sử dụng

3. **Pandas**

Là 1 open source, được cộng đồng đánh giá là high-performance, việc xử lý dữ liệu, tính toán sẽ dễ dàng hơn rất nhiều cách truyền thống

- Sử dụng đọc file csv

CSV (*Comma Separated Values*) : là loại định dạng văn bản đơn giản mà trong đó, các giá trị được ngăn cách bởi dấu phẩy.

- Để sử dụng : import csv, import pandas as pd

4. **Matplotlib**

Là thư viện vẽ đồ thị mạnh mẽ và hữu ích trên Python và thư viện Numy. Module được sử dụng nhiều nhất là Pyplot. Module được sử dụng nhiều nhất của Matplotlib là Pyplot cung cấp giao diện như MATLAB nhưng thay vào đó, nó sử dụng Python và nó là nguồn mở

- Để cài đặt, gõ : pip install matplotlib vào phần Terminal và đợi cài đặt
- Để sử dụng : import matplotlib as plt

5. **Cv2 (openCV : open source computer vision)**

Là một trong những mã nguồn hàng đầu trong việc xử lý hình ảnh theo thời gian thực. Đó là thư viện miễn phí và hỗ trợ đa nền tảng cung cấp theo giấy phép BSD mã nguồn mở

- Để cài đặt, gõ : pip install opencv-python
- Để sử dụng : import cv2

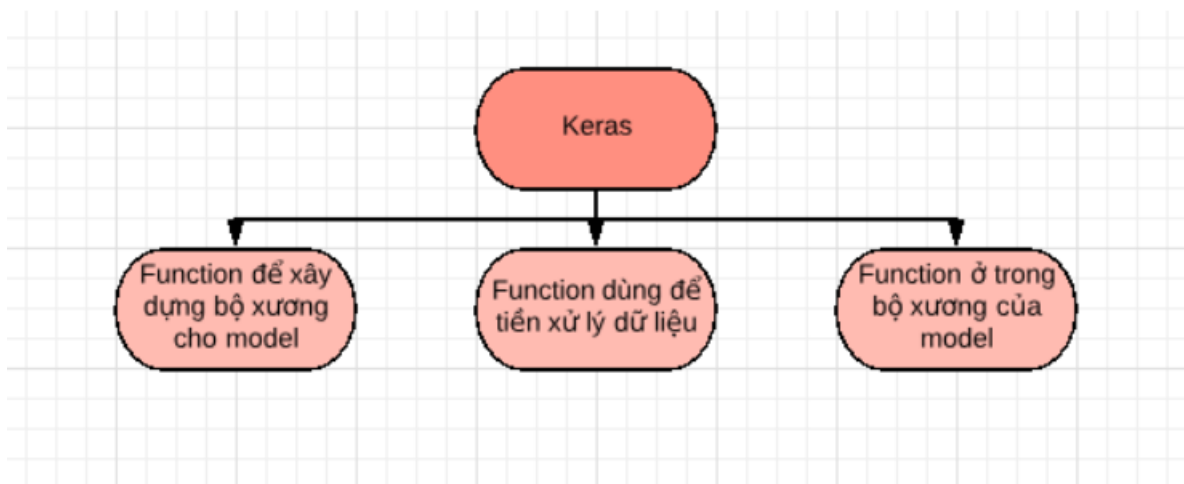
6. **Keras**

Là một thư viện được phát triển vào năm 2015 bởi François Chollet, là một kỹ sư nghiên cứu deep learning tại google. Nó là một open source cho neural network được viết bởi ngôn ngữ python. keras là một API bậc cao có thể sử dụng chung với các thư viện deep learning nổi tiếng như tensorflow(được phát triển bởi gg), CNTK(được phát triển bởi microsoft), theano(người phát triển chính Yoshua Bengio)

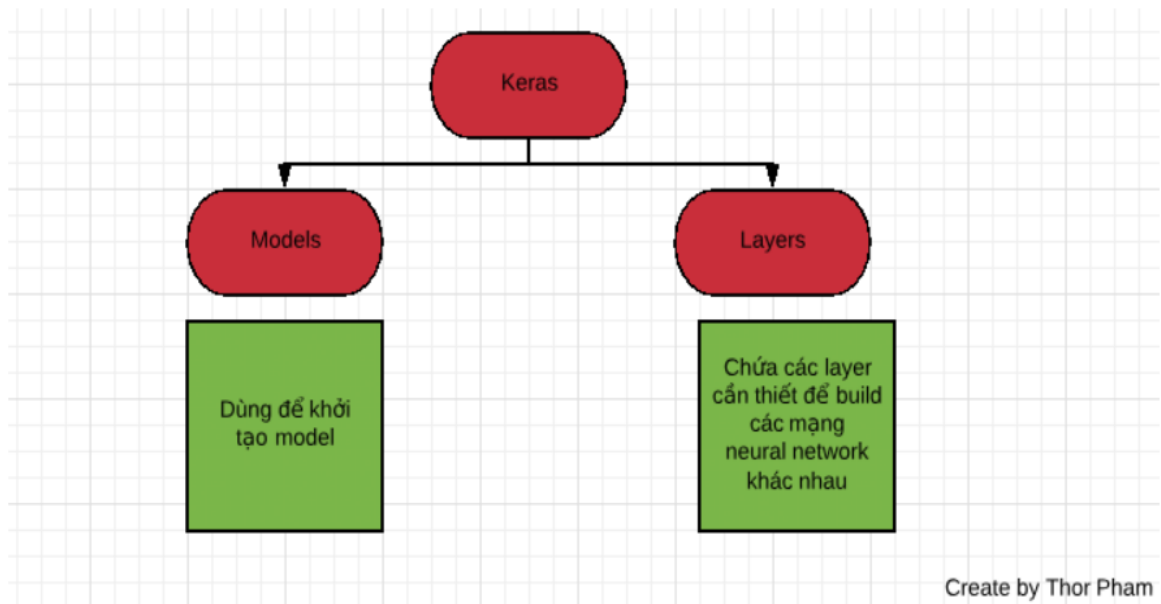
III. CƠ SỞ HOẠT ĐỘNG

1. *Cấu trúc của Keras*

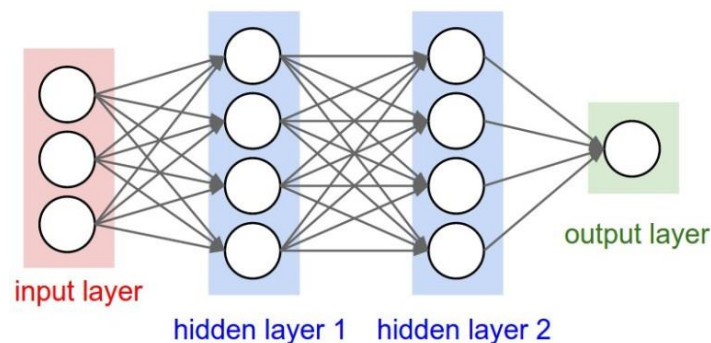
Chúng ta có thể chia thành 3 phần chính:



Các module dùng để xây dựng bộ xương cho model:



2. Mô hình CNN

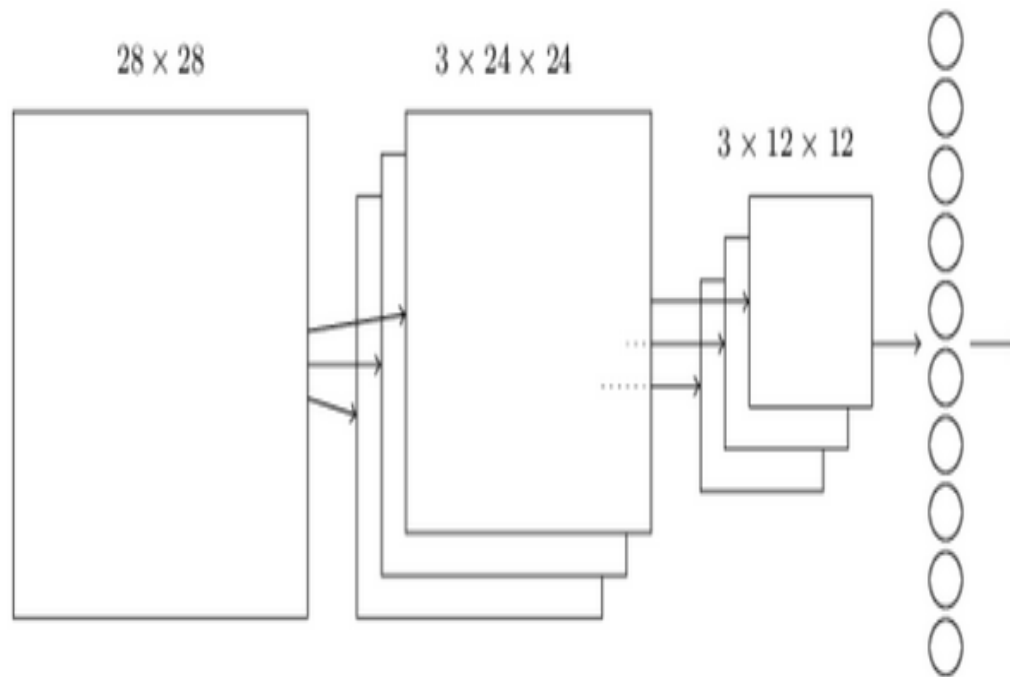


-Mô hình mạng neural ở trên gồm 3 lớp: lớp nhập (input), lớp ẩn(hidden) và lớp xuất (output). Mỗi nút trong lớp nhập nhận giá trị của một biến độc lập và chuyển vào mạng. Dữ liệu từ tất cả các nút trong lớp nhập được tích hợp – ta gọi là tổng trọng số – và chuyển kết quả cho các nút trong lớp ẩn. Gọi là “ẩn” vì các nút trong lớp này chỉ liên lạc với các nút trong lớp nhập và lớp xuất, và chỉ có người thiết kế

mạng mới biết lớp này (người sử dụng không biết lớp này). Các nút trong lớp xuất nhận các tín hiệu tổng trọng hóa từ các nút trong lớp ẩn. Mỗi nút trong lớp xuất tương ứng với một biến phụ thuộc

3. CNN

Ta đã có nhắc trên cơ sở lý thuyết là *Convolutional Neural Network (CNNs – Mạng nơ-ron tích chập)* và giờ tìm hiểu sâu hơn cách hoạt động



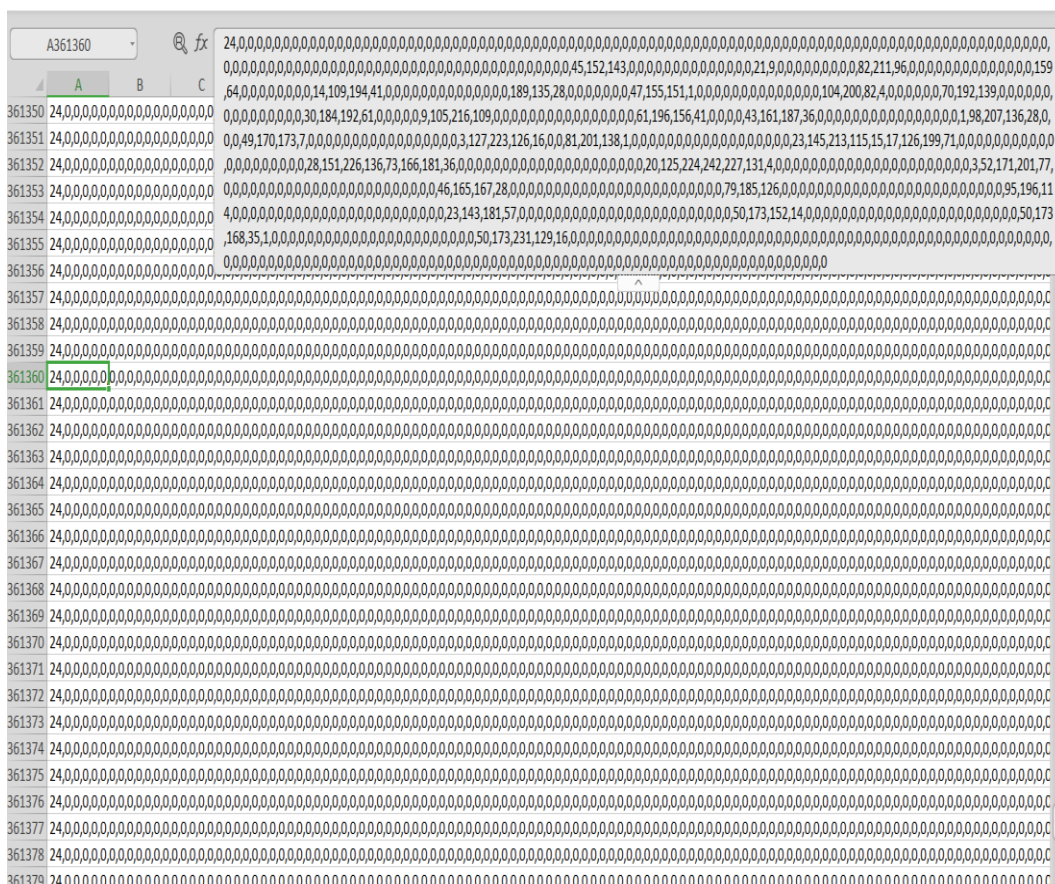
Trong bài ta áp dụng bằng cách các filters để tìm ra các features cho ảnh đầu vào, sau đó các sử dụng các features đó cho việc phân loại ảnh. Trong kiến trúc này có nhiều thành phần, nhưng core của nó là một thành phần được gọi là lớp Convolution (CONV). Lớp CONV này sẽ chứa các filters cho kiến trúc CNN. Tuy nhiên, thay vì phải tự đưa ra các filters thủ công như CV, các filters sẽ được tìm ra một cách tự động thông qua quá trình training. Có nghĩa là các giá trị trong mỗi một filter lúc này (chính là các weights) ban đầu sẽ được khởi tạo ngẫu nhiên, sau quá trình học mà chúng sẽ được cập nhật lại, và kết quả cố định cuối cùng được ổn định khi kết thúc training. Khi xây dựng CNN, chúng ta chỉ cần chỉ ra là ở mỗi lớp CONV cần số lượng bao nhiêu filter, size mỗi filter, giá trị stride, giá trị padding...

- File dữ liệu định dạng csv

- Bộ dữ liệu hình ảnh mà chúng ta sử dụng trong bài toán này bao gồm hình ảnh của 26 chữ cái trong tiếng Anh từ A đến Z tải bộ dữ liệu (82MB) ở link sau: <https://www.kaggle.com/sachinpatel21/az-handwritten-alphabets-in-csv->

format. Sau khi tải về và giải nén, sẽ nhận được tệp dữ liệu định dạng CSV: **A_Z Handwritten Data.csv**

- Trong tệp vừa nhận được sẽ là 1 file Excel định danh csv như hình sau ta có định dạng của các chữ thành các ma trận hơn 300000 dòng
- Mở file CSV :



IV. XÂY DỰNG ĐỒ ÁN NHẬN DẠNG CHỮ VIẾT TAY

1. Tìm hiểu và xử lý dữ liệu

Đầu tiên ta import các thư viện cần thiết :

```

import numpy as np
import coremltools
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os

#Ép chạy trên CPU
os.environ['CUDA_VISIBLE_DEVICE']=-1

from sklearn.preprocessing import MinMaxScaler
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 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 keras import optimizers

```

Đọc bộ dữ liệu **A_Z HandwrittenData.csv** :

```

dataset = pd.read_csv('A_Z HandwrittenData.csv').astype('float32')

dataset.rename(columns={'0':'label'}, inplace=True)
#Chia x làm datatrain , theo cột và y nhãn chữ
X = dataset.drop('label', axis = 1)
y = dataset['label']
print("Shape:" ,X.shape)
print("Culoms count:" ,len(X.iloc[1]))

```

```

Shape: (372037, 784)
Culoms count: 784

```

-Sử dụng thư viện pandas trong việc đọc file csv sẽ nhanh hơn hiệu quả và thao tác tốt hơn so với cách đọc file

```
import csv

with open ('A_Z HandwrittenData.csv','rt') as f:

    data = csv.reader(f)

    rows =[]

    for row in data:

        rows.append(row)
```

Vì việc đọc file theo thư viện csv làm cho việc đọc file lâu do lượng dữ liệu lớn

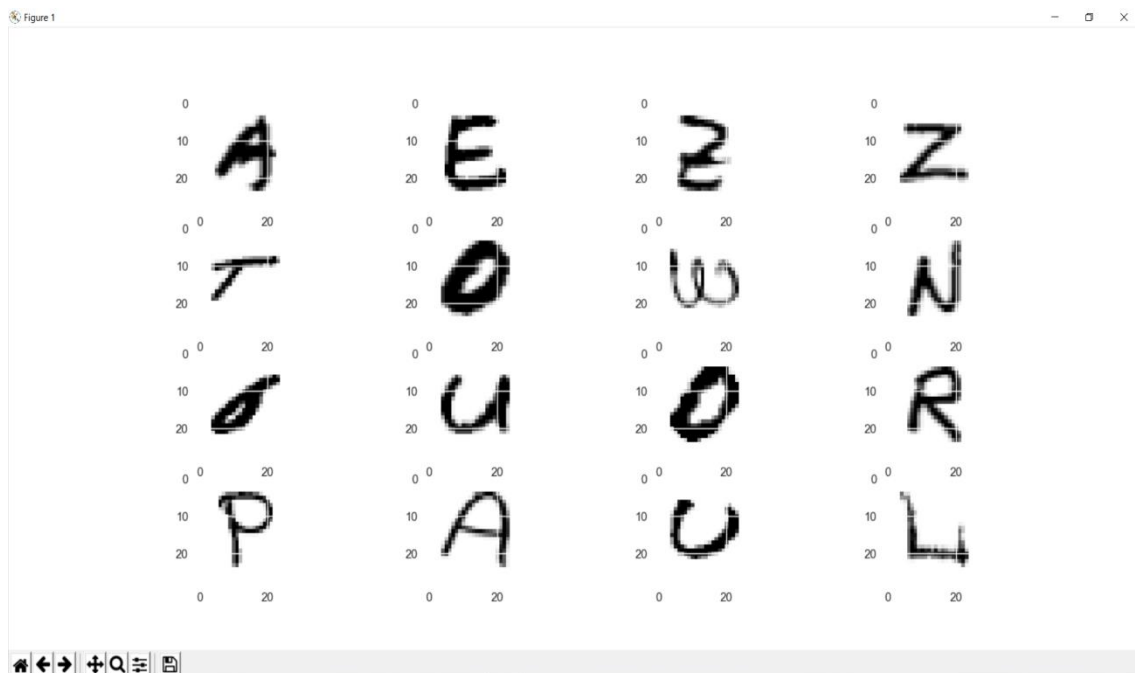
Chạy dòng đầu của data:

```
    0.1  0.2  0.3  0.4  0.5  0.6  ...  0.566  0.567  0.568  0.569  0.570  0.571
0  0.0  0.0  0.0  0.0  0.0  0.0  ...   0.0   0.0   0.0   0.0   0.0   0.0
1  0.0  0.0  0.0  0.0  0.0  0.0  ...   0.0   0.0   0.0   0.0   0.0   0.0
2  0.0  0.0  0.0  0.0  0.0  0.0  ...   0.0   0.0   0.0   0.0   0.0   0.0
3  0.0  0.0  0.0  0.0  0.0  0.0  ...   0.0   0.0   0.0   0.0   0.0   0.0
4  0.0  0.0  0.0  0.0  0.0  0.0  ...   0.0   0.0   0.0   0.0   0.0   0.0

[5 rows x 784 columns]
```

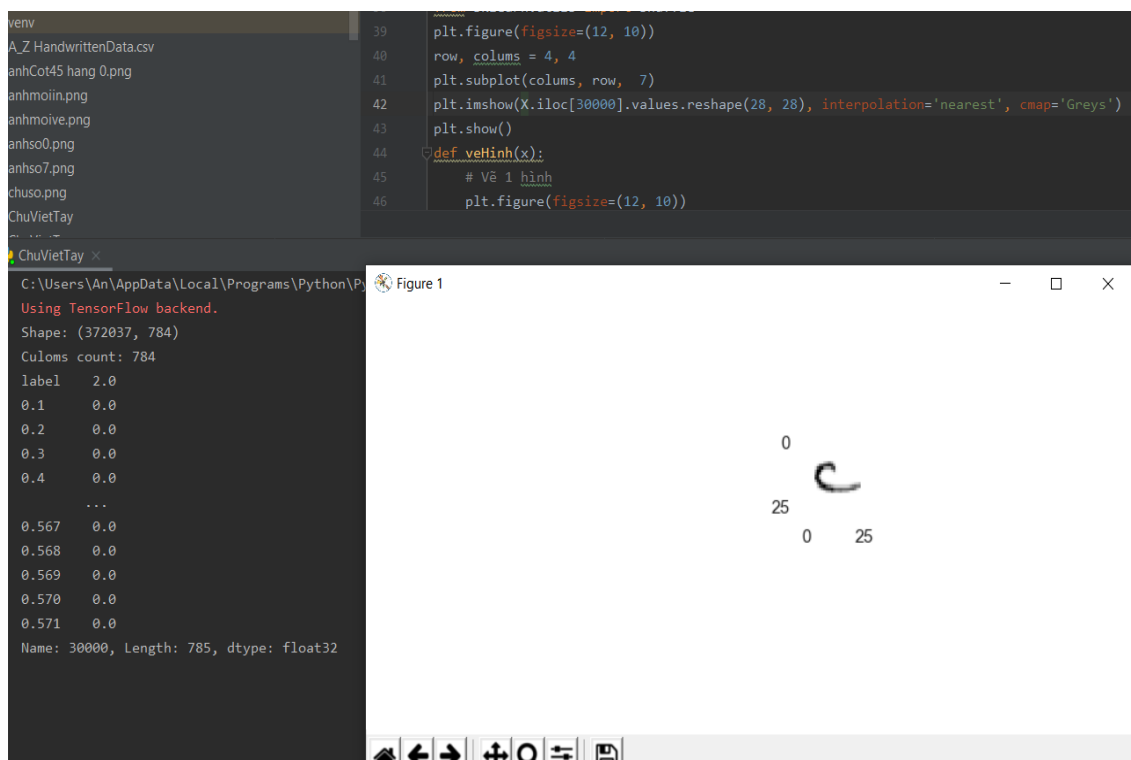
Bộ dữ liệu chứa khoảng 372037 chữ cái viết tay in hoa phân thành 26 chữ cái. Mỗi một chữ cái có độ phân giải thấp (28*28 pixel).

Chúng ta sẽ cắt ngẫu nhiên một vài chữ cái làm ví dụ:

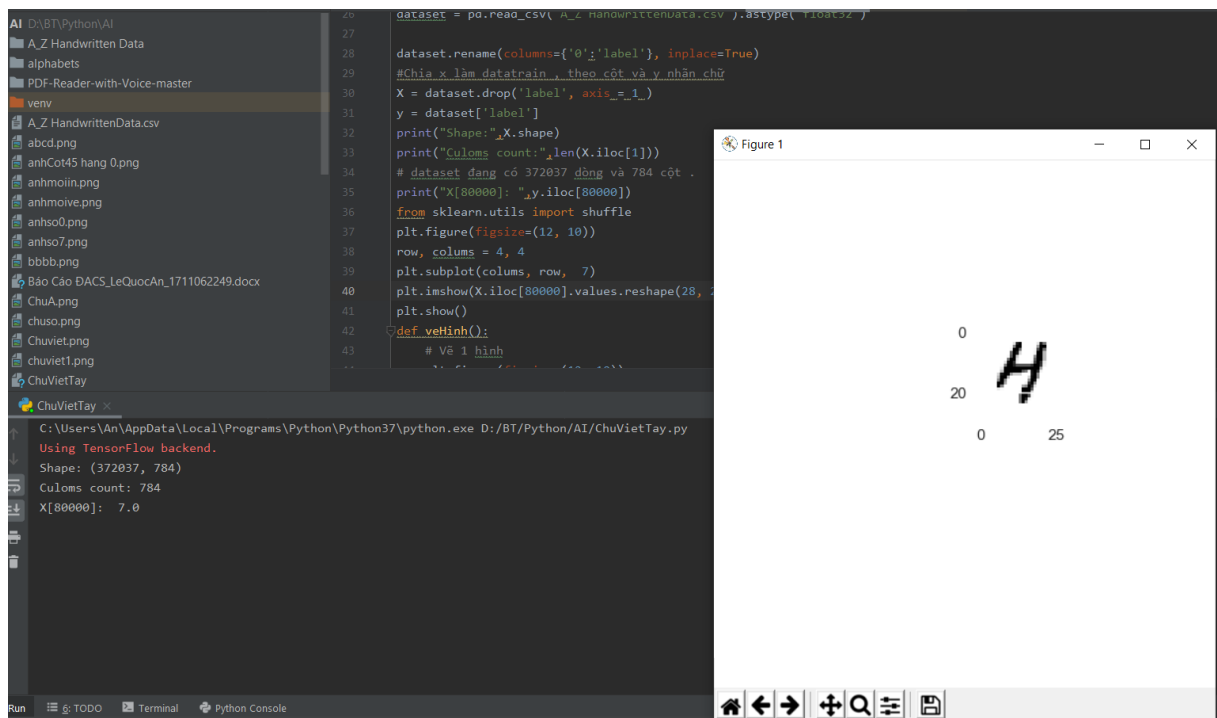


Để hiểu rõ hơn về file dữ liệu:

Ký tự đầu tiên của list là 2, vậy ký tự mà ta mong chờ thu được là ký tự C. Đúng như vậy, ảnh mà ta nhận được từ đoạn code trên là:



Tương tự với chữ H:



- Mỗi ảnh là một mảng NumPy 2 chiều, 28x28, với mỗi pixel có giá trị từ 0 đến 255.
- Nhãn là một mảng của các số nguyên từ 0 đến 25, tương ứng với mỗi chữ cái:

Nhãn	Lớp
0	A
1	B
2	C
3	D
4	E
5	F
6	G

7	H
8	I
9	J
10	K
11	L
12	M
13	N
14	O
15	P
16	Q
17	R
18	S
19	T
20	U
21	V
22	W
23	X
24	Y
25	Z

Tiếp theo mỗi ảnh sẽ được gán với một nhãn duy nhất. Vì tên của mỗi lớp không có trong tập dữ liệu, nên chúng ta có thể định nghĩa ở đây để dùng về sau:

```
alphabets_mapper = {0:'A', 1:'B', 2:'C', 3:'D', 4:'E', 5:'F', 6:'G', 7:'H', 8:'I', 9:'J', 10:'K',
                    11:'L', 12:'M', 13:'N', 14:'O', 15:'P', 16:'Q', 17:'R', 18:'S', 19:'T', 20:'U',
                    21:'V', 22:'W', 23:'X', 24:'Y', 25:'Z'}
```

Số lượng nhãn:



➤ Chúng ta có thể thấy chữ F và V có độ quan sát tương đối thấp (Dữ liệu của chữ ít)

```
F : 1164
V : 340
```

2. Chuẩn bị dữ liệu

Trước tiên do tập dữ liệu lớn ta dùng `from sklearn.model_selection import train_test_split`

để chia dữ liệu (75% dữ liệu sẽ làm huấn luyện và 25% còn lại sẽ được làm tập thử nghiệm)

➤ Với tập dữ liệu này, 279027 chữ sẽ được dùng để huấn luyện và 93010 chữ sẽ được dùng để đánh giá khả năng phân loại nhận diện ảnh của mạng neuron.

```
X_train, X_test, Y_train, Y_test = train_test_split(X, y)
```

Tập dữ liệu sau khi được tải sẽ trả về 4 mảng NumPy:

- 2 mảng X_train và Y_train (nhãn của X_train tương ứng) là tập huấn luyện. Mô hình sẽ học từ dữ liệu của 2 mảng này.
- 2 mảng X_test và Y_test (nhãn của X_test tương ứng) là tập kiểm thử. Sau khi mô hình được huấn luyện xong, chúng ta sẽ chạy thử mô hình với dữ liệu đầu vào từ X_test để lấy kết quả, và so sánh kết quả đó với dữ liệu đối ứng từ Y_test để đánh giá chất lượng của mạng neuron.

Tiếp theo, xử lý các dữ liệu bằng cách điều chỉnh độ giãn cách của thuộc tính . Chúng ta sẽ sử dụng **Min-max scaling** , lúc này các giá trị được điều chỉnh sao cho dải giá trị của nó nằm gọn trong đoạn $[0, 1]$.

Và chuyển dữ liệu thành dạng float :

```
X_train = X_train.reshape(X_train.shape[0], 28, 28, 1).astype('float32')
X_test = X_test.reshape(X_test.shape[0], 28, 28, 1).astype('float32')
```

Và sau khi chuyển dữ liệu:

[illegible]

- Chúng ta có thể khám phá dữ liệu một chút trước khi huấn luyện mô hình. Câu lệnh sau sẽ cho ta thấy có 279.027 ảnh trong tập huấn luyện, với mỗi ảnh được biểu diễn theo dạng 28x28 pixel:

```
X_train.shape: (279027, 28, 28, 1)
```

- Tương tự, tập huấn luyện cũng có 279.027 nhãn đối ứng, và theo 26 chữ cái(được hiểu theo số 0->25):

```
Y_train.shape: (279027, 26)
```

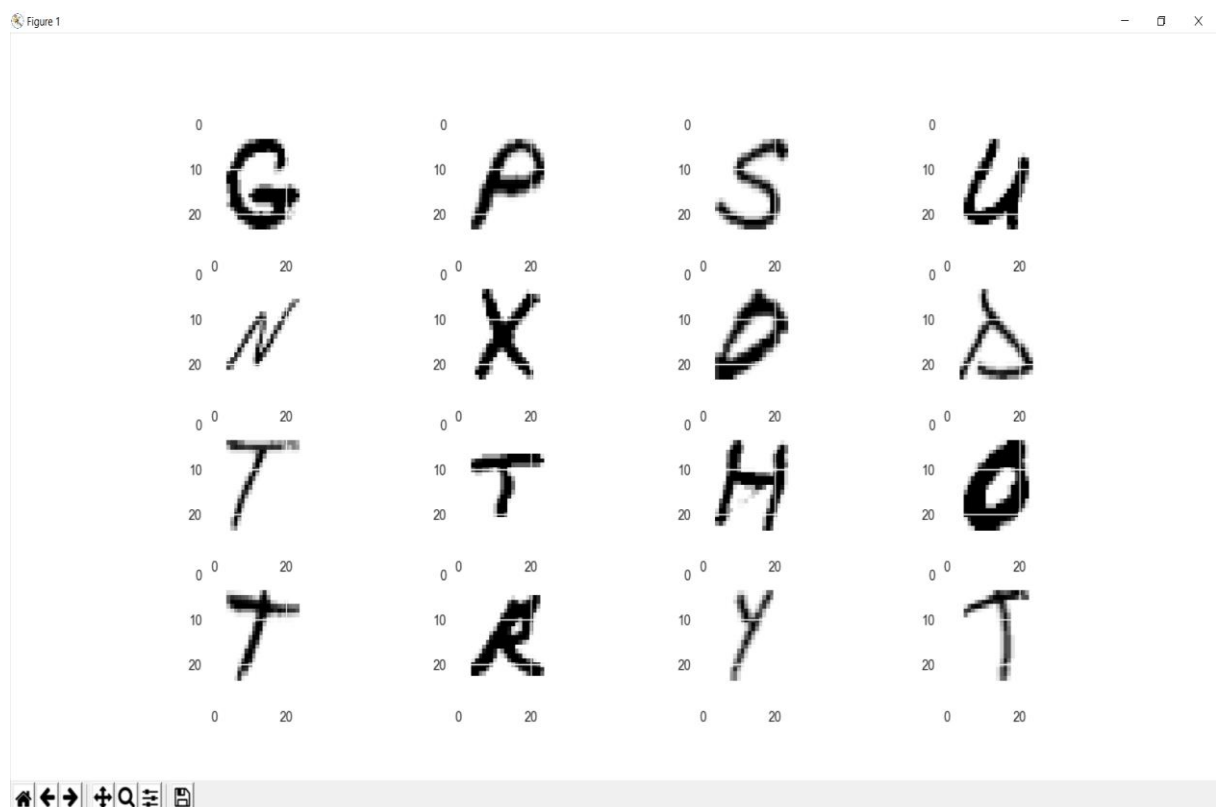
- Có 93.010 ảnh trong tập kiểm thử, mỗi ảnh cũng được biểu diễn ở dạng 28 x 28 pixel:

```
X_test.shape : (93010, 28, 28, 1)
```

- Và tập kiểm thử cũng chứa 93,010 nhãn:

```
Y_test.shape : (93010, 26)
```

Dữ liệu sau khi được xử lý qua giãn tỷ lệ :



3. Xây dựng mô hình

Thành phần cơ bản của một mạng neuron là các layer. Các layer trích xuất các điểm đặc biệt từ dữ liệu mà chúng đón nhận. Khi thực hiện tốt, những điểm đặc biệt này

mang nhiều ý nghĩa và phục vụ cho toán của chúng ta. Đa số các mô hình deep learning đều chứa các layer đơn gian được xâu chuỗi lại với nhau.

Ta sử dụng Keras để tạo mới một mạng nơ-ron như đã nói từ trước. Để tạo mạng nơ-ron sử dụng Keras, chúng ta sẽ sử dụng model Keras Sequential và tạo layer bằng cách thêm layer Dense vào model sequential

Đa số các layer, ví dụ tf.keras.layers.Dense, đều có các trọng số sẽ được học trong quá trình huấn luyện.

```
model = Sequential()  
model.add(Conv2D(32, (5, 5), input_shape=(28, 28, 1), activation='relu'))  
model.add(MaxPooling2D(pool_size=(2, 2)))  
model.add(Dropout(0.3))  
model.add(Flatten())  
model.add(Dense(128, activation='relu'))  
model.add(Dense(len(y.unique()), activation='softmax'))  
sgd = optimizers.SGD(lr=_0.001)  
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])  
model.summary()  
model.fit(X_train, Y_train, validation_data=(X_test, Y_test), epochs=3, batch_size=100, verbose=2)
```

Trong layer Dense đầu tiên bao gồm 1 hidden layer, truyền vào các tham số:

Convolutional Layers: chứa các layer trong mạng nơ ron tích chập

- (5,5): Kết nối trong một vùng nhỏ của các neuron đầu vào như một filter có kích thước 5×5
- input_shape: Đây là kích thước của feature mà ta đã tạo ra
- relu: Là hàm kích hoạt (activation function)

Pooling Layers : Chứa các layer dùng trong mạng CNN

- dùng để lấy feature nổi bật (dùng max) và giúp giảm parameter khi training ,đoạn code sau thực hiện max-pooling với size là 2×2

Dropout : Dùng để ngăn chặn overfitting

- Ngăn chặn bằng cách bỏ qua những nơ-ron ngẫu nhiên trong mỗi vòng lặp, weights sẽ không được cập nhật và sẽ bị loại bỏ trong quá trình training
- 0.3: để biểu diễn xác suất của một nơ-ron sẽ bị loại bỏ trong quá trình training (xác suất dao động trong khoảng 0.2-0.5 để có kết quả tốt nhất.)

Do Total params : 594,138 (Quá cao) xảy ra overfitting

Lớp đầu tiên là một CONV. Input là ảnh đầu vào có size là (None, 28, 28, 1). Tham số thứ nhất là None thể hiện số lượng ảnh training. Thông số Conv2D(32, 5, 5) có nghĩa là ta muốn tạo 32 filter với kích thước mỗi filter là 5 x 5.

Trong mạng neuron trên, lớp đầu tiên, [tf.keras.layers.Flatten](#), chuyển đổi định dạng của hình ảnh từ mảng hai chiều (28,28,1) thành mảng một chiều ($12 \times 12 \times 32 = 4608$)

Sau layer làm phẳng ảnh (từ 2 chiều thành 1 chiều), phần mạng neuron còn lại gồm một chuỗi hai layer [tf.keras.layers.Dense](#). Đây là các layer neuron được kết nối hoàn toàn (mỗi một neuron của layer này kết nối đến tất cả các neuron của lớp trước và sau nó).

Layer Dense đầu tiên có 128 nút (hoặc neuron). Layer thứ hai (và cuối cùng) là lớp softmax có 26 nút, với mỗi nút tương đương với điểm xác suất, và tổng các giá trị của 26 nút này là 1 (tương đương 100%)

```
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 24, 24, 32)	832
max_pooling2d_1 (MaxPooling2D)	(None, 12, 12, 32)	0
dropout_1 (Dropout)	(None, 12, 12, 32)	0
flatten_1 (Flatten)	(None, 4608)	0
dense_1 (Dense)	(None, 128)	589952
dense_2 (Dense)	(None, 26)	3354

```

Total params: 594,138
Trainable params: 594,138
Non-trainable params: 0

```

4. TÓM LƯỢC MODEL

Model nhận đầu vào là các ảnh (28 x 28 x 1), đầu ra là lớp fc softmax với nodes = 10. Ở giữa là 1 kiến trúc CNN đơn giản với thứ tự các lớp như sau: CONV (relu) => MAXPOOL => FLATTEN => FC(relu) => FC(softmax). Vì mỗi filter mang 5 x 5 weights (w1 ... w8) nên tổng số weights sẽ là 32 x 5 x 5 = 800 , mỗi filters cộng w0 (bias weight) cho nó. Vì vậy, kết quả chính xác phải là 32 x 5 x 5 + 32 x 1 = 320.

Thêm nữa, xét mỗi filter (5 x 5) trượt trên hình gốc (28 x 28 x 1) ko có padding = "SAME" thì output là một hình (26 x 26 x 1). Vì vậy 32 filter sẽ cho ra output có size là (24 x 24 x 32)

Xét lớp thứ hai là Max Pooling. Lớp này chỉ đơn giản là giảm kích thước ma trận input theo một cửa sổ kích thước (2 x 2). Vì vậy lớp này ko có params (weights) nào cả. Và từ input = (24 x 24 x 32), output của lớp này sẽ là (12 x 12 x 32).

Lớp Flatten, chuyển đổi định dạng của hình ảnh từ mảng hai chiều (28,28,1) thành mảng một chiều ($12 \times 12 \times 32 = 4608$)

Kết luận, tổng số lượng weights sẽ là 594138. Số lượng weights càng nhiều thì độ phức tạp của model càng lớn và dễ dẫn đến overfit, hoặc quá ít thì không đủ mạnh với những dữ liệu phức tạp.

5. ĐÁNH GIÁ MÔ HÌNH

Chúng ta đánh giá các chất lượng của mô hình bằng tập kiểm thử:

```
#chúng đánh giá các chất lượng của mô hình bằng tập kiểm thử
test_loss, test_acc = model.evaluate(X_test, Y_test, verbose=0)
```

```
Epoch 1/2
- 99s - loss: 0.2127 - accuracy: 0.9420
Epoch 2/2
- 112s - loss: 0.0944 - accuracy: 0.9743
Tỷ lệ: 97.90990352630615%
```

Ta thấy rằng độ accuracy của mô hình, khi đánh giá bằng tập kiểm thử. Khoảng cách giữa hai độ accuracy khi huấn luyện và khi kiểm thử thể hiện sự overfitting. Overfitting xảy ra khi một mô hình ML hoạt động kém hơn khi được cung cấp các đầu vào mới, mà mô hình chưa từng thấy trước đây trong quá trình đào tạo.

Huấn luyện càng nhiều thì độ chuẩn xác càng cao, ta kiểm thử qua 2 vòng kiểm thử độ chính xác đã: ~0.98:

Đưa ra dự đoán:

Từ đầu ta đã chia file dữ liệu thành 25% để làm dữ liệu test.

Với một mô hình đã được đào tạo, chúng ta có thể dùng nó để đưa ra dự đoán với một số ảnh.

```
predictions = model.predict(X_test)
```

Ở đây, mô hình sẽ dự đoán nhãn cho từng hình ảnh trong bộ thử nghiệm. Dự đoán đầu tiên:

```
print(predictions[0])
```

```
[5.6288453e-11 2.1388123e-08 4.3435125e-07 3.8401326e-05 1.9647519e-08
1.2210364e-08 5.0070270e-08 3.3436684e-11 1.2760907e-05 9.9973005e-01
3.5592059e-12 5.6558824e-10 1.0007132e-10 3.0308279e-08 7.8299860e-07
9.9229098e-08 5.7336771e-09 3.5619105e-11 1.4900252e-04 5.9345461e-05
1.2590397e-06 2.9073051e-09 2.5283940e-08 1.6557332e-09 7.2836260e-06
4.0949752e-07]
```

Trong ví dụ này, dự đoán là một mảng 10 số thực, mỗi số tương ứng với "độ tự tin" của mô hình rằng ảnh đó thuộc về nhãn đó. Chúng ta có thể thấy nhãn nào có độ tự tin cao nhất:

```
print(np.argmax(predictions[0]))
```

```
9
```

Với mô hình tự tin nhất ảnh này là chữ ở nhãn 9 và đây là chữ:

```
9
```

Tiếp tục ta cho dự đoán 1 loạt chữ:

- Ta lấy ngẫu nhiên trong 25% dữ liệu đã cắt ra sẵn để training để đánh giá mô hình


```

#NHẬN LOẠT CHỮ
plt.figure(figsize=(12,10))
row, columns = 4, 4
for i in range(16):
    plt.subplot(row, columns, i+1)
    plt.imshow(X_test[i+5].reshape(28,28), interpolation='nearest', cmap='Greys')
    plt.xticks([])
    plt.yticks([])
    prob = model.predict(X_test[i+5].reshape(1,28,28,1))
    pred = int(np.argmax(prob, axis=1))
    response = alphabets_mapper[pred] + "(" + str(round(prob[0][pred] * 100,2)) + " %"
    plt.title(response)

plt.show()
#####

```

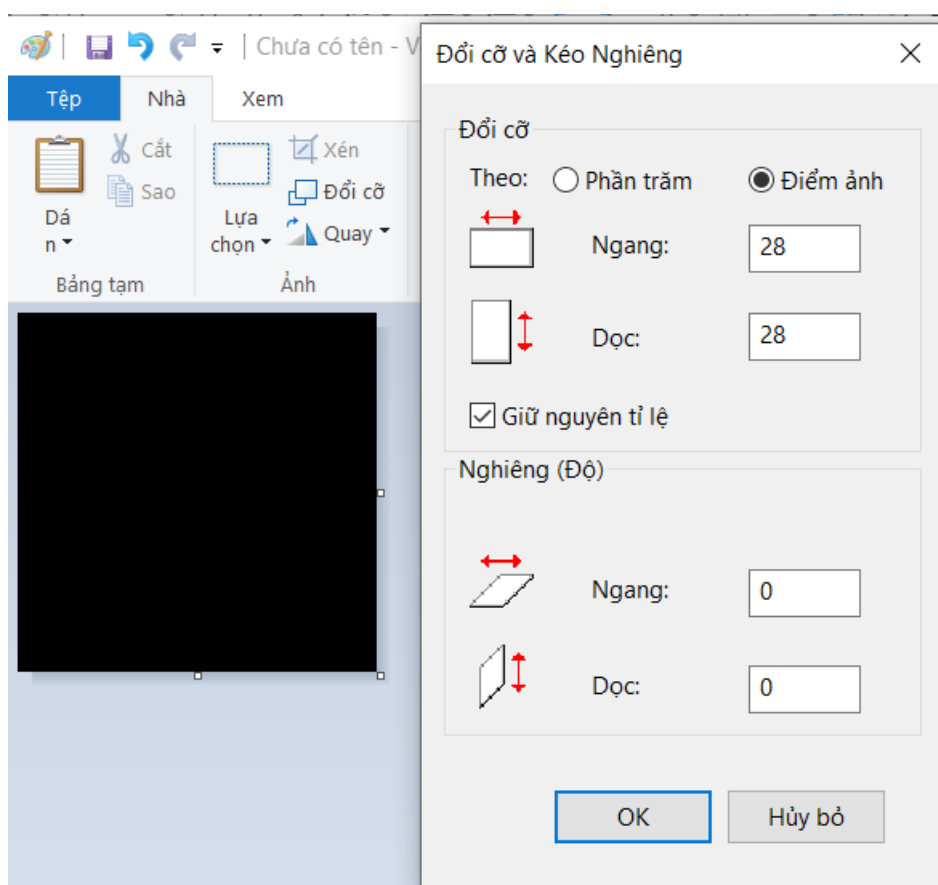
Kết quả nhận được

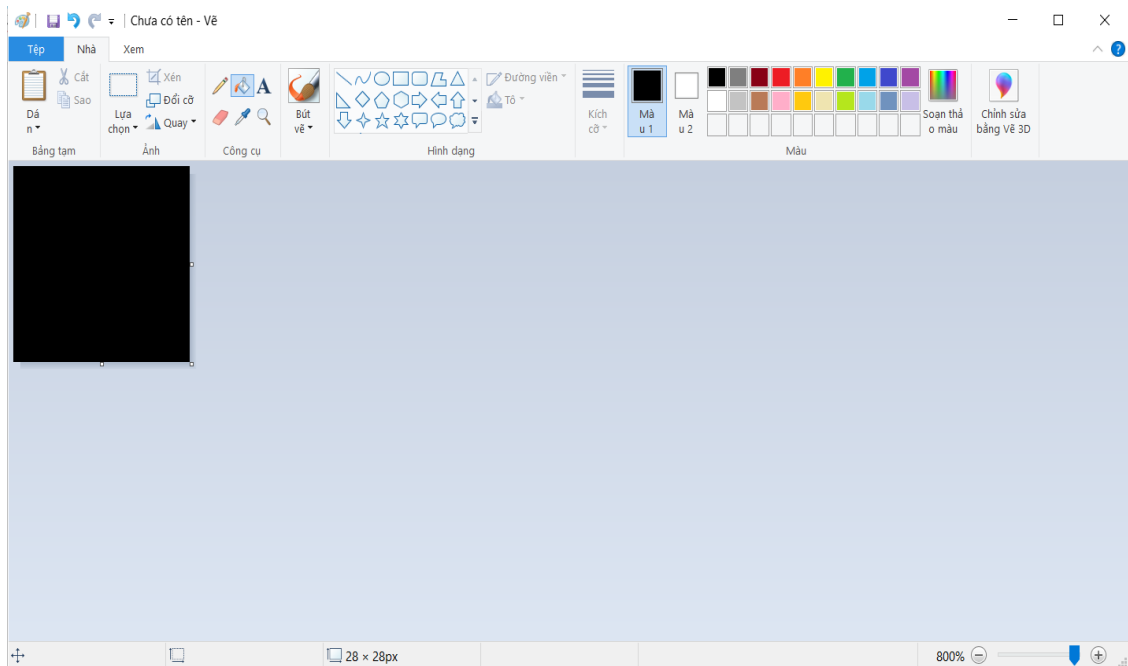


6. ÁP DỤNG

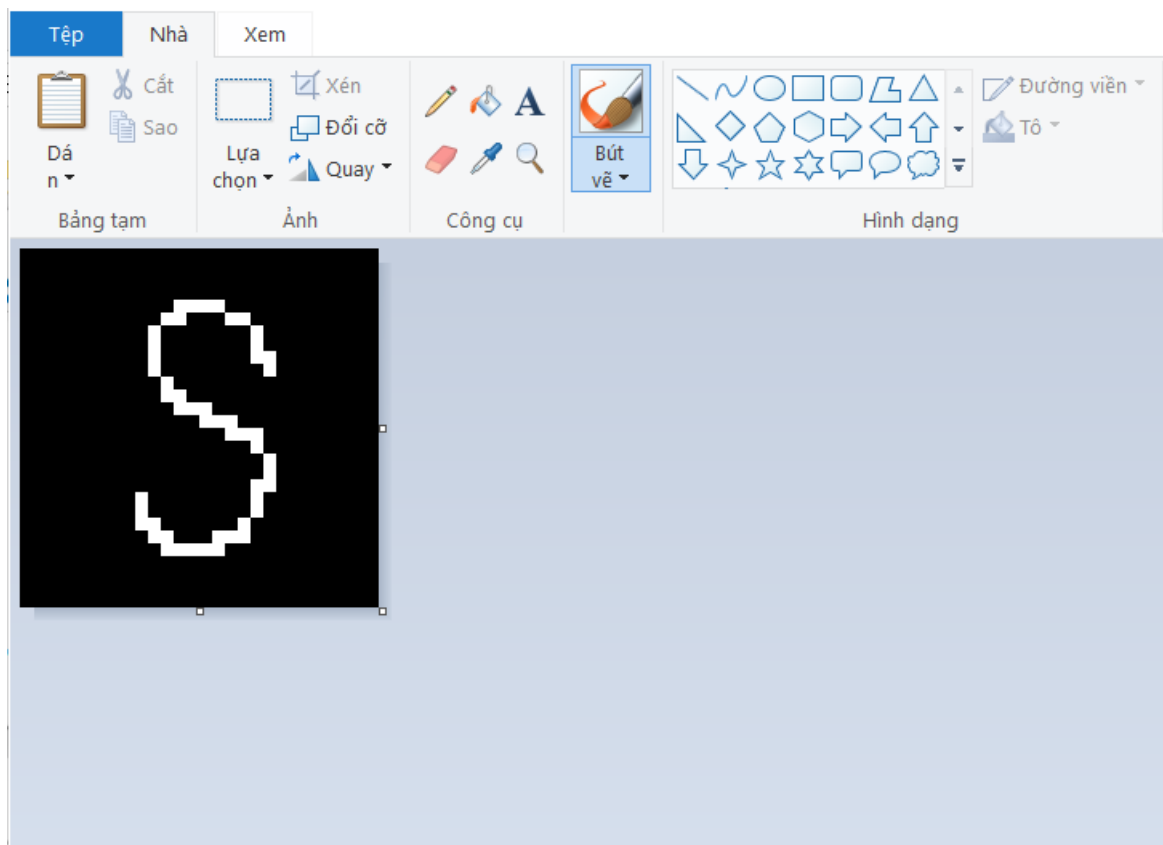
Mục tiêu là đưa 1 chữ viết tay bằng ứng dụng vẽ (Paint)

Chữ viết có size 28*28.





Vẽ 1 chữ bất kỳ



Bước tiếp theo ta đưa vào phần mềm để thực hiện nhận biết nhưng trước tiên ta phải xử lý dữ liệu đầu vào của chữ:

- Chữ vào nền đen chữ trắng (nhận vào là 1 ảnh xám)

```
#NHÂN 1 CHỮ
```

```
import cv2
```

```
img = cv2.imread('ChuA.png', 0)
```

```
img = np.array(img).astype('float32')
```

```
img = img.reshape(-1, 784)
```

```
print(img.shape[0])
```

```
print(img.shape)
```

```
img = standard_scaler.transform(img)
```

```
img = img.reshape(1, 28, 28, 1).astype('float32')
```

```
prediction = model.predict(img)
```

```
plt.figure(figsize=(12, 10))
```

```
row, columns = 4, 4
```

```
plt.subplot(columns, row, 7)
```

```
plt.imshow(img.reshape(28, 28), interpolation='nearest', cmap='Greys')
```

```
plt.xticks([])
```

```
plt.yticks([])
```

```
plt.xlabel('Chữ đang test là chữ :{}'.format(alphabets_mapper[np.argmax(prediction)]))
```

```
plt.show()
```

Và kết quả:

S

Chữ đang test là chữ :S

Thiết kế giao diện:

```

from PIL import Image, ImageTk
from tkinter import filedialog
from tkinter import messagebox
root = Tk()
root.geometry("550x300+300+150")
root.resizable(width=True, height=True)

def openfn():
    filename = filedialog.askopenfilename(title='open')
    return filename

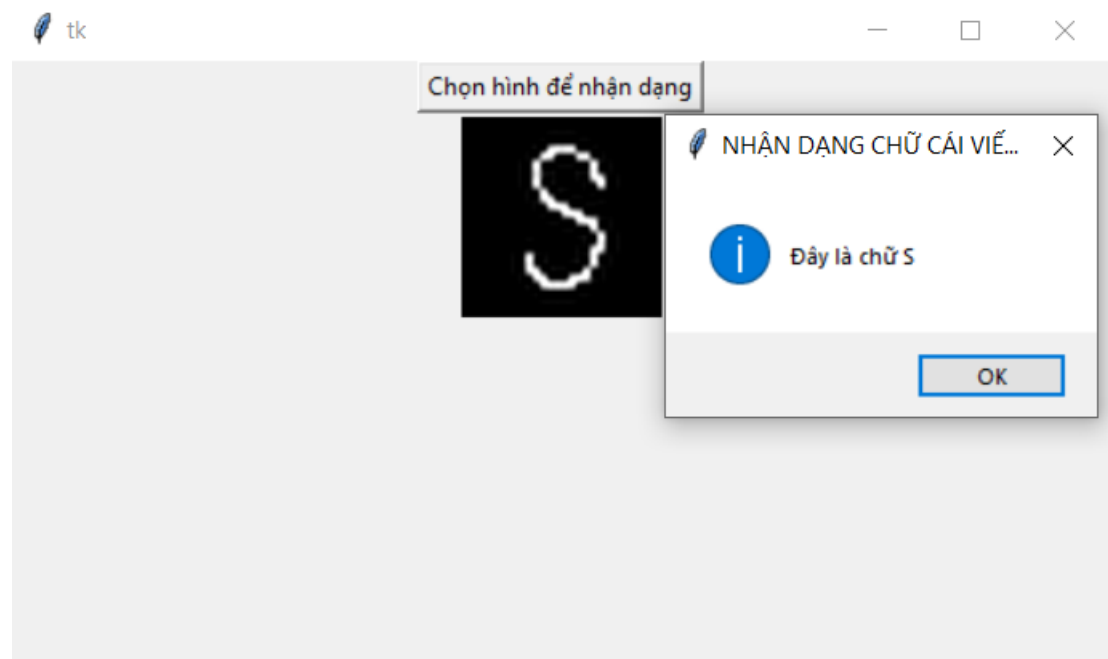
#def data config

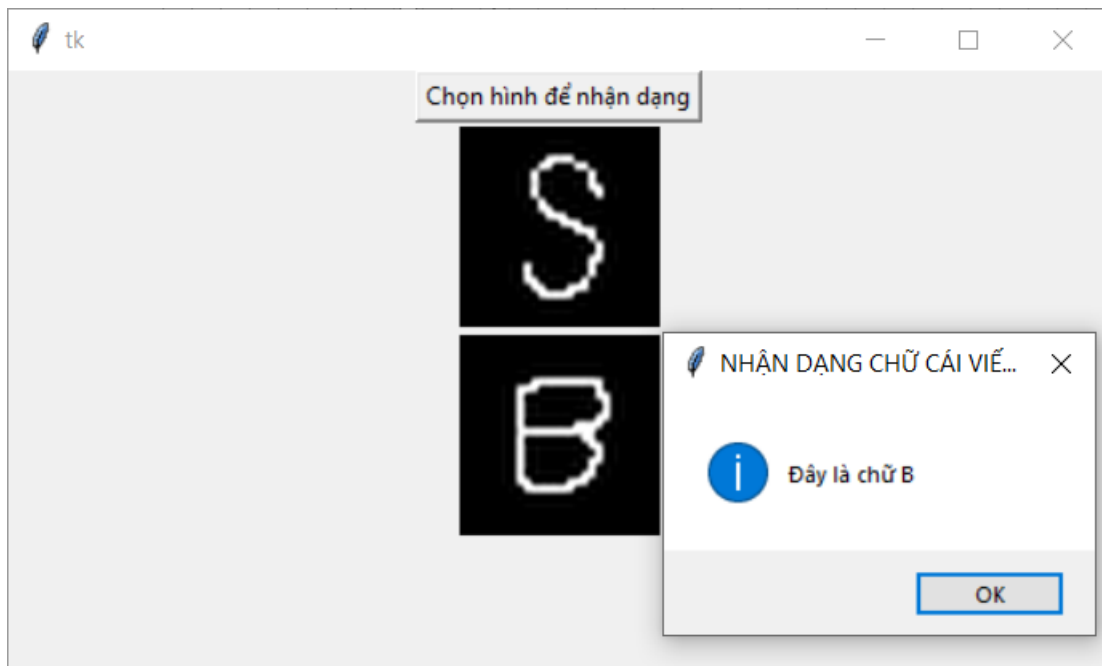
def HienThi(x):
    Prediction = model.predict(x)
    return alphabets_mapper[np.argmax(Prediction)]

def open_img():
    x = openfn()
    imgage = Image.open(x)
    imgage = imgage.resize((100, 100), Image.ANTIALIAS)
    imgage = ImageTk.PhotoImage(imgage)
    panel = Label(root, image=imgage)
    panel.image = imgage
    panel.pack()
    im = cv2.imread(x,0)
    im = np.array(im).astype('float32')
    im = im.reshape(-1, 784)
    im = standard_scaler.transform(im)
    im = im.reshape(im.shape[0], 28, 28, 1).astype('float32')
    messagebox.showinfo("NHÂN DẠNG CHỮ CÁI VIẾT TAY", "Đây là chữ {}".format(HienThi(im)))
    plt.figure(figsize=(12, 10))
    row, columns = 4, 4
    plt.subplot(columns, row, 7)
    plt.imshow(im.reshape(28, 28), interpolation='nearest', cmap='Greys')
    plt.show()

btn = Button(root, text='Chọn hình để nhân dạng', command=open_img).pack()

```





✓ Hoàn thành được demo nhận diện được 1 chữ cái in hoa

THÊM :

Bắt đầu xử lý một loạt chữ theo hướng nhận diện nhiều chữ 1 lượt:

A B C D E F Y

```

from skimage.feature import hog
image = cv2.imread("abcd.png")
im_gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
im_blue = cv2.GaussianBlur(im_gray, (5, 5), 0)
im_thre = cv2.threshold(im_blue, 90, 255, cv2.THRESH_BINARY_INV)
contours, hierarchy = cv2.findContours(thre, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
rects = [cv2.boundingRect(cnt) for cnt in contours]

```

```

for i in contours:
    (x, y, w, h) = cv2.boundingRect(i)
    cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 3)
    roi = thre[y:y + h, x:x + w]
    roi = np.pad(roi, (20, 20), 'constant', constant_values=(0, 0))
    roi = roi.reshape(roi.shape[0], 28, 28, 1).astype('float32')
    roi = cv2.dilate(roi, (3, 3))

    roi_hog_fd = hog(roi, orientations=9, pixels_per_cell=(14, 14), cells_per_block=(1, 1), block_norm="L2")
    nbr = model.predict(np.array([roi_hog_fd], np.float32))
    cv2.putText(image, str(int(nbr[0])), (x, y), cv2.FONT_HERSHEY_DUPLEX, 2, (0, 255, 255), 3)
    cv2.imshow("image", image)
cv2.imwrite("image_pand.jpg", image)

```

! Kết quả chỉ đạt tới cắt được từng khung nhưng chưa thể nhận dạng cả dòng được (Hướng phát triển)

CHƯƠNG 3: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

- Đồ án tìm hiểu về Tensorflow và hầu hết về API Keras, để tìm hiểu về nhận dạng chữ viết tay in hoa Là một bài 'Hello World' cho người bắt đầu AI
- Hoàn thành demo nhận dạng chữ 1 chữ cái in hoa

Khuyết điểm:

- Quá trình học máy lâu (từ 90~120 giây) cho 1 vòng huấn luyện
- Đầu vào nhận dạng vẽ ở Pain có trường hợp bị lỗi đầu vào

Hướng phát triển

- Phát triển chương trình thành nhận dạng đoạn văn bản
- Áp dụng vào thực tiễn : phát hiện biển số xe, chấm đề kiểm tra,...

Cải thiện:

- Tập huấn luyện nhanh và tăng độ chính xác
- Xử lí file đầu vào chuẩn xác hơn

TÀI LIỆU THAM KHẢO :

- 1) <https://www.kaggle.com/sachinpatel21/az-handwritten-alphabets-in-csv-format>
- 2) <https://thorphan.github.io/blog/>
- 3) <https://forum.machinelearningcoban.com/t/xay-dung-mang-cnn-voi-tensorflow-va-keras/3974?fbclid=IwAR36q7WyF41qCqhi81CpzHTwWw2iKCmSX2h8BQt7MvB9yUkaHugTgA3vjmQ>
- 4) <https://techblog.vn/mini-series-chuan-bi-du-lieu-cho-thuat-toan-machine-learning?fbclid=IwAR2QnCV8-xXxBtzQkbgbk3VY0eJe9yfIiHYkDY4j1xduzwmOMPq7vFsoGOE>

- 5) <https://topdev.vn/blog/thuat-toan-cnn-convolutional-neural-network/>
- 6) <https://www.tensorflow.org/tutorials?hl=vi>