BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT HƯNG YÊN**



**TIỂU LUẬN**

**HỌC MÁY CƠ BẢN**

......................Giảng viên HD: TS. Nguyễn Văn Hậu

Học viên thực hiện: Nguyễn Văn Dương

Phạm Thị Huê

*Hưng Yên, 11/2023*

# LỜI NÓI ĐẦU

Trong [khoa học máy tính](https://vi.wikipedia.org/wiki/Khoa_h%E1%BB%8Dc_m%C3%A1y_t%C3%ADnh), trí tuệ nhân tạo hay AI ([tiếng Anh](https://vi.wikipedia.org/wiki/Ti%E1%BA%BFng_Anh): *artificial intelligence*), đôi khi được gọi là trí thông minh nhân tạo, là [trí thông minh](https://vi.wikipedia.org/wiki/Tr%C3%AD_th%C3%B4ng_minh) được thể hiện bằng [máy móc](https://vi.wikipedia.org/wiki/M%C3%A1y_m%C3%B3c), trái ngược với trí thông minh tự nhiên của con người. Thông thường, thuật ngữ "trí tuệ nhân tạo" thường được sử dụng để mô tả các máy chủ móc (hoặc máy tính) có khả năng bắt chước các chức năng "nhận thức" mà con người thường phải liên kết với [tâm trí](https://vi.wikipedia.org/wiki/T%C3%A2m_tr%C3%AD), như "học tập" và "giải quyết vấn đề".

Trí tuệ nhân tạo có thể được phân thành ba loại hệ thống khác nhau: trí tuệ nhân tạo phân tích, lấy cảm hứng từ con người và nhân tạo. AI phân tích chỉ có các đặc điểm phù hợp với [trí tuệ nhận thức](https://vi.wikipedia.org/wiki/Nh%E1%BA%ADn_th%E1%BB%A9c); tạo ra một đại diện nhận thức về thế giới và sử dụng học tập dựa trên kinh nghiệm trong quá khứ để thông báo các quyết định trong tương lai. AI lấy cảm hứng từ con người có các yếu tố từ [trí tuệ](https://vi.wikipedia.org/wiki/Tr%C3%AD_tu%E1%BB%87_x%C3%BAc_c%E1%BA%A3m) nhận thức và [cảm xúc](https://vi.wikipedia.org/wiki/Tr%C3%AD_tu%E1%BB%87_x%C3%BAc_c%E1%BA%A3m); hiểu cảm xúc của con người, ngoài các yếu tố nhận thức và xem xét chúng trong việc [ra quyết định](https://vi.wikipedia.org/w/index.php?title=Ra_quy%E1%BA%BFt_%C4%91%E1%BB%8Bnh&action=edit&redlink=1). AI nhân cách hóa cho thấy các đặc điểm của tất cả các loại năng lực (nghĩa là trí tuệ nhận thức, cảm xúc và xã hội), có khả năng tự ý thức và tự nhận thức được trong các tương tác.

Trí tuệ nhân tạo được thành lập như một môn học thuật vào năm 1956, và trong những năm sau đó đã trải qua nhiều làn sóng lạc quan, sau đó là sự thất vọng và mất kinh phí (được gọi là " mùa đông AI "), tiếp theo là cách tiếp cận mới, thành công và tài trợ mới. Trong phần lớn lịch sử của mình, nghiên cứu AI đã được chia thành các trường con thường không liên lạc được với nhau. Các trường con này dựa trên các cân nhắc kỹ thuật, chẳng hạn như các mục tiêu cụ thể (ví dụ: " [robot học](https://vi.wikipedia.org/wiki/Robot_h%E1%BB%8Dc) " hoặc "học máy"),việc sử dụng các công cụ cụ thể ("logic" hoặc [mạng lưới thần kinh nhân tạo](https://vi.wikipedia.org/wiki/M%E1%BA%A1ng_n%C6%A1-ron_nh%C3%A2n_t%E1%BA%A1o)) hoặc sự khác biệt triết học sâu sắc. Các ngành con cũng được dựa trên các yếu tố xã hội (các tổ chức cụ thể hoặc công việc của các nhà nghiên cứu cụ thể).

Lĩnh vực này được thành lập dựa trên tuyên bố rằng [trí thông minh của con người](https://vi.wikipedia.org/wiki/Tr%C3%AD_th%C3%B4ng_minh_c%E1%BB%A7a_con_ng%C6%B0%E1%BB%9Di) "có thể được mô tả chính xác đến mức một cỗ máy có thể được chế tạo để mô phỏng nó".Điều này làm dấy lên những tranh luận triết học về bản chất của [tâm trí](https://vi.wikipedia.org/wiki/T%C3%A2m_tr%C3%AD) và đạo đức khi tạo ra những sinh vật nhân tạo có trí thông minh giống con người, đó là những vấn đề đã được [thần thoại](https://vi.wikipedia.org/wiki/L%E1%BB%8Bch_s%E1%BB%AD_ng%C3%A0nh_tr%C3%AD_tu%E1%BB%87_nh%C3%A2n_t%E1%BA%A1o), viễn tưởng và [triết học](https://vi.wikipedia.org/wiki/Tri%E1%BA%BFt_h%E1%BB%8Dc_v%E1%BB%81_tr%C3%AD_tu%E1%BB%87_nh%C3%A2n_t%E1%BA%A1o) từ [thời cổ đại](https://vi.wikipedia.org/wiki/Th%E1%BB%9Di_k%E1%BB%B3_c%E1%BB%95_%C4%91%E1%BA%A1i) đề cập tới. Một số người cũng coi AI là [mối nguy hiểm cho nhân loại](https://vi.wikipedia.org/wiki/%C4%90i%E1%BB%83m_k%E1%BB%B3_d%E1%BB%8B_c%C3%B4ng_ngh%E1%BB%87) nếu tiến triển của nó không suy giảm. Những người khác tin rằng AI, không giống như các cuộc cách mạng công nghệ trước đây, sẽ tạo ra [nguy cơ thất nghiệp hàng loạt](https://vi.wikipedia.org/w/index.php?title=Th%E1%BA%A5t_nghi%E1%BB%87p_c%C3%B4ng_ngh%E1%BB%87&action=edit&redlink=1).

Trong thế kỷ 21, các kỹ thuật AI đã trải qua sự hồi sinh sau những tiến bộ đồng thời về sức mạnh máy tính, [dữ liệu lớn](https://vi.wikipedia.org/wiki/D%E1%BB%AF_li%E1%BB%87u_l%E1%BB%9Bn) và hiểu biết lý thuyết; và kỹ thuật AI đã trở thành một phần thiết yếu của [ngành công nghệ](https://vi.wikipedia.org/w/index.php?title=C%C3%B4ng_nghi%E1%BB%87p_c%C3%B4ng_ngh%E1%BB%87&action=edit&redlink=1), giúp giải quyết nhiều vấn đề thách thức trong học máy, [công nghệ phần mềm](https://vi.wikipedia.org/wiki/C%C3%B4ng_ngh%E1%BB%87_ph%E1%BA%A7n_m%E1%BB%81m) và [nghiên cứu vận hành](https://vi.wikipedia.org/wiki/V%E1%BA%ADn_tr%C3%B9_h%E1%BB%8Dc).

Trong nội dung bài viết này, chúng ta cùng tìm hiệu một số ứng dụng cơ bản của lĩnh vực AI, ML thông qua một ví dụ đơn giản “Nhận dạng ký tự trên biển số xe sử dụng Hồi quy tuyến tính”.

**PHỤ LỤC**

[LỜI NÓI ĐẦU 2](#_Toc151137622)

[PHỤ LỤC 3](#_Toc151137623)

[PHẦN I - BÀI TOÁN VÀ GIẢI PHÁP. 4](#_Toc151137624)

[1.1 Bài toán. 4](#_Toc151137625)

[1.2 Giải pháp và lý thuyết. 4](#_Toc151137626)

[PHẦN II - THỰC HIỆN. 5](#_Toc151137627)

[2.1 Chuẩn bị và xử lý dữ liệu training. 5](#_Toc151137628)

[2.1.1 Chụp ảnh. 5](#_Toc151137629)

[2.1.2 Cắt ảnh biển số. 5](#_Toc151137630)

[2.1.3 Cắt ảnh ký tự từ ảnh biển số. 7](#_Toc151137631)

[2.1.4 Đánh nhãn / phân lớp. 10](#_Toc151137632)

[2.1.5 Tạo dữ liệu csv cho traning. 12](#_Toc151137633)

[2.1.6 Training. 15](#_Toc151137634)

[2.2 Dự đoán và nhận dạng. 19](#_Toc151137635)

[2.2.1 Tải lại ánh xạ nhãn và các model 19](#_Toc151137636)

[2.2.2 Thiết kế các method detect id từ ảnh ký tự và ảnh chữ số. 20](#_Toc151137637)

[2.2.3 Chuẩn bị ảnh test 21](#_Toc151137638)

[2.2.4 Test 22](#_Toc151137639)

[2.2.5 Một số kết quả thu được. 23](#_Toc151137640)

[PHẦN III - KẾT LUẬN. 25](#_Toc151137641)

# PHẦN I - BÀI TOÁN VÀ GIẢI PHÁP.

* 1. **Bài toán.**

Trong khi số lượng các phương tiện giao thông cơ giới đang tăng nhanh ở mọi nơi thì việc kiểm soát các hoạt động liên quan đến các phương tiện này bằng sức người ngày càng trở lên quá tải. Để đáp ứng thì có thể tăng số lượng nhân lực theo tỷ lệ thuận. Đó là tăng số lượng cảnh sát giao thông trên các tuyến đường; tăng số lượng nhân viên an ninh tại các khu vực đỗ, và giữ xe. Tất nhiên, giải pháp đó không phải là một giải pháp tốt bởi vì kèm theo đó là thời gian để đào tạo và quản lý nhân lực. Với sự phát triển của khoa học công nghệ, đặc biệt trong lĩnh vực thị giác máy tính, máy học, trí thông minh nhân tạo, chúng ta hoàn toàn có thể giải quyết bài toán đó một cách tốt hơn mà không những không cần tăng mà có thể giảm nhân lực quản lý.

* 1. **Giải pháp và lý thuyết.**

Để giải quyết bài toán này, ở đây, chúng ta thử với phương pháp sử dụng hồi quy tuyến tính đa biến.

Chúng ta sẽ chụp thật nhiều ảnh biển số, cắt lấy từng ảnh chữ cái và số. Trích rút thông tin từ các ảnh đó để thu được các bộ số đặc trưng X. Mỗi bộ số sẽ tương ứng cho một ký tự theo công thức hồi quy đa biến:

*Y = β0 + β1X1 + β2X2 + … + βnXn*

Y là đại diện cho một ký tự.

β0 là hằng số hồi quy

β1, β2,… βn: hệ số hồi quy, hay còn được gọi là hệ số góc.

X1, X2,… Xn: biến độc lập.

Các hệ số β0, β1, β2,… βn sẽ có được khi chúng mang tập hợp dữ liệu Y, X1, X2,… Xn vào training.

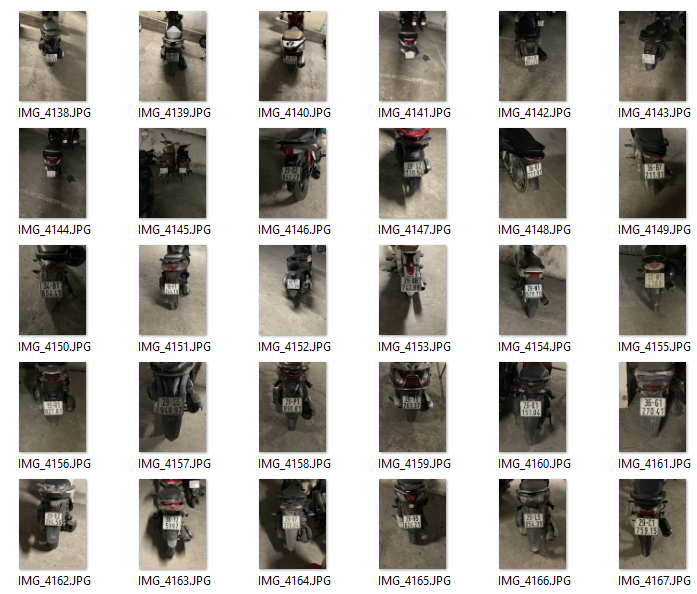
Sau khi đã có các tham số đặc trưng β0, β1, β2,… βn, chúng ta đưa một tập hợp X1, X2,… Xn khác lắp ráp vào công thức phía trên thì sẽ trả lại cho chúng ra giá trị Y. Và từ Y chúng ta sẽ đối chiếu và suy ngược ra tập dữ liệu X1, X2,… Xn là đặc trưng của ký tự nào.

# PHẦN II - THỰC HIỆN.

## 2.1 Chuẩn bị và xử lý dữ liệu training.

### 2.1.1 Chụp ảnh.

Để bắt đầu với giải pháp này, công việc đầu tiên là sẽ phải chuẩn bị một lượng lớn ảnh của các biển số trong thực tế. Trong trường hợp này có khoảng 500 ảnh được chụp từ một bãi đậu xe.



### 2.1.2 Cắt ảnh biển số.

Sau quá trình chụp ảnh, chúng ta vẫn chưa sử dụng được ngay các dữ liệu ảnh gốc này vì trong ảnh có quá nhiều các chi tiết khác nhau. Phần ảnh chứa các số nằm đâu đó trong không gian ảnh gốc. Bước tiền xử lý tiếp theo ta sẽ cắt khu vực nghi ngờ chứa hình ảnh biển số ra khỏi ảnh gốc. Để thực hiện điều đó ta cần tới sự hỗ trợ của thư viện xử lý ảnh opencv. Trong python, thư viện này có tên cv2. Chúng ta sử dụng đoạn script bên dưới để tiến hành xử lý cắt ảnh biển số từ ảnh gốc.

Ý tưởng của đoạn script là nhị phân hóa ảnh gốc theo ngưỡng trung bình động của từng bức ảnh. Khu vực ảnh biển số sẽ là một hình chữ nhật màu trắng do biển số thường sẽ có màu sáng hơn ngưỡng trung bình động của bức ảnh. Ta nhận dạng khu vực biển số dựa vào việc phát hiện đường biên và kiểm tra kích thước.

Nội dung script:

import cv2

import os

path = 'raw\_images'

list\_files = os.listdir(path)

file\_index = 0

for file\_name in list\_files:

    file\_path = path + '/' + file\_name

    mat = cv2.imread(file\_path)

    mat\_gray = cv2.cvtColor(mat, cv2.COLOR\_BGR2GRAY)

    mat\_blurred = cv2.GaussianBlur(mat\_gray, (7, 7), 0)

    thresh, mat\_bin = cv2.threshold(mat\_blurred, 128, 255, cv2.THRESH\_BINARY)

    contours, hierarchy = cv2.findContours(mat\_bin, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

    contour\_index = 0

    for contour in contours:

        box = cv2.boundingRect(contour)

        x = box[0]

        y = box[1]

        width = box[2]

        height = box[3]

        if width < height :

            continue

        if width < 500:

            continue

        scale = float(width) / float(height)

        if scale < 1.1 or scale > 2:

            continue

        mat\_crop\_gray = mat\_blurred[y:y+height, x:x+width]

        mat\_crop\_bin = mat\_bin[y:y+height, x:x+width]

        output\_path = 'plate\_images/' + str(file\_index) + '\_' + str(contour\_index) + '.jpg'

        cv2.imwrite(output\_path, output\_path)

        contour\_index += 1

        pass

    file\_index+=1

Sau khi chạy xong đoạn script trên, các ảnh nghi ngờ là biển số sẽ được cắt ra từ ảnh gốc và gom vào trong thư mục có tên “plate\_images”.



Trong thư mục ảnh biển số cũng có thể sẽ có các bức ảnh cắt bị lỗi, ta sẽ xóa thủ công để làm sạch dữ liệu để chuẩn bị cho bước tiếp theo.

### 2.1.3 Cắt ảnh ký tự từ ảnh biển số.

Ở bước trước, chúng ta có rất nhiều ảnh biển số. Ở bước này, chúng ta sẽ tiếp tục cắt từng ảnh ký tự chữ và số. Phương pháp tương đối giống với lần cắt biển số ra khỏi ảnh gốc.

Nội dung script :

import cv2

import os

import numpy as np

def save\_number\_blackwhite(index, image):

    output\_path = 'character\_images/blackwhites/' + str(file\_index) + '.bmp'

    cv2.imwrite(output\_path, image)

    pass

path = 'plate\_images'

list\_files = os.listdir(path)

file\_index = 0

for file\_name in list\_files:

    file\_path = path + '/' + file\_name

    mat = cv2.imread(file\_path)

    mat\_gray = cv2.cvtColor(mat, cv2.COLOR\_BGR2GRAY)

    mat\_blurred = cv2.GaussianBlur(mat\_gray, (17, 17), 0)

    mean = cv2.mean(mat\_blurred)[0]

    thresh, mat\_bin = cv2.threshold(mat\_blurred, int(mean), 255, cv2.THRESH\_BINARY\_INV)

    contours, hierarchy = cv2.findContours(mat\_bin, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

    for contour in contours:

        box = cv2.boundingRect(contour)

        x = box[0]

        y = box[1]

        width = box[2]

        height = box[3]

        if width > height :

            continue

        if width < 50 or width > 400:

            continue

        mat\_crop\_gray = mat\_blurred[y:y+height, x:x+width]

        cv2.rectangle(mat\_crop\_gray, (0,0, width, height), (0), 2)

        save\_number\_gray(file\_index, mat\_crop\_gray)

        mat\_crop\_bin = mat\_bin[y:y+height, x:x+width]

        cv2.rectangle(mat\_crop\_bin, (0,0, width, height), (0), 2)

        erode\_thresh = width / 10

        kernel = np.ones((int(erode\_thresh), int(erode\_thresh)), np.uint8)

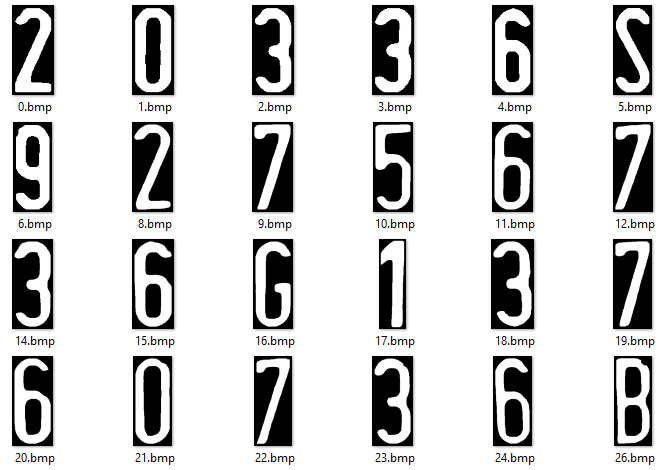
        mat\_crop\_bin = cv2.erode(mat\_crop\_bin, kernel)

        save\_number\_blackwhite(file\_index, mat\_crop\_bin)

        file\_index+=1

print("Finished !")

Kết thúc chạy script, trong thư mục “character\_images/blackwhites” chứa rất nhiều ảnh các chữ và số được cắt ra.



Các ảnh chữ số này được đánh tên theo tứ tự được cắt ra. Nội dung của chúng xuất hiện ngẫu nhiên dựa vào việc chụp ảnh và do đó nó rất lộn xộn. Trong đó sẽ có rất nhiều ảnh cũng được cắt ra nhưng lại không phải ảnh của chữ số do bị phát hiện nhầm. Chúng ta cần xóa thủ công nó để đảm bảo dữ liệu được sạch trước khi đưa đến bước training. Nếu không, các nội dung nhiễu và sai lệch này sẽ làm kết quả training cũng bị sai số.

### 2.1.4 Đánh nhãn / phân lớp.

Cũng trong thư mục ảnh này, sau khi đã xóa đi các dữ liệu xấu, ta tiến hành phân tách các ảnh dựa vào nội dung của chúng theo từng thư mục riêng : 0,1,2 …A,B.C … Z.

Việc này thực hiện một cách thủ công.



Trong thư mục này có tổng cộng 29 thư mục tương ứng cho các ký tự số và chữ, khuyết một số ký tự chữ như I,J,W,Q,U,Y. Có thể các cơ quan cấp phát đã loại bỏ các ký tự này do lý do đặc biệt hoặc do việc chụp ảnh đã không lấy được các biển số có chứa ký tự này.

Có thể chạy thêm một đoạn script như thế này để thấy được tần suất xuất hiện của các ký tự:

import os

import numpy as np

path = 'character\_images/blackwhites'

list\_dir = os.listdir(path)

for dir in list\_dir:

    list\_file = os.listdir(path + '/' + dir)

    print(dir, ',', len(list\_file))

Chúng ta sẽ thu được kết quả như sau:

0 , 144

1 , 425

2 , 424

3 , 287

4 , 185

5 , 192

6 , 212

7 , 193

8 , 191

9 , 328

A , 4

B , 67

C , 7

D , 16

E , 37

F , 14

G , 23

H , 18

K , 6

L , 37

M , 12

N , 6

P , 14

R , 4

S , 13

T , 8

V , 17

X , 18

Y , 10

Đồ thị hóa sẽ như thế này:

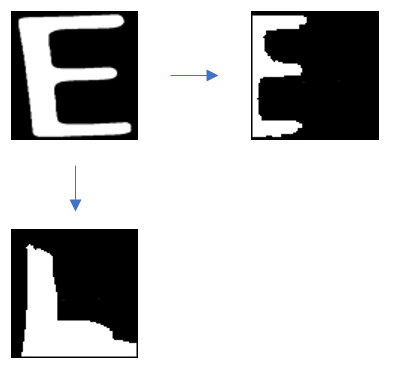
Phân bổ dữ liệu đang rất mất cân bằng, các ký tự số có tần suất cao hơn nhiều so với ký tự chữ. Ký tự số 1 xuất hiện 425 lần trong khi ký tự R chỉ xuất hiện 4 lần trên tổng số 2912 mẫu. Điều này do định dạng thông tin của biển số. Tuy nhiên ta sẽ xem xét thông tin này sau.

### 2.1.5 Tạo dữ liệu csv cho traning.

Ở bước này, từ dữ liệu ảnh đã được phân lớp vào các thư mục trong bước trước, chúng ta sẽ tạo ra một tập tin csv chứa dữ liệu training. Ý tưởng ở bước này là chúng ta sẽ đọc tất cả các ảnh ký tự chữ số ở bước trước và chuyển đổi nó thành một ma trận 2D có kích thước 32x32.



Sau đó sẽ tính tổng điểm sáng theo hai hướng ngang và dọc thu được 2 mảng 1 chiều kích thước 32 phần tử. Sau đó trừ offset nhỏ nhất của từng mảng để tăng mức độ ảnh hưởng của biến thiên giá trị.



Sau đó, 2 mảng 32 phần tử này sẽ được hợp lại với nhau thành mảng 64 phần tử. 64 phần tử này chính là 64 feature X sẽ được đưa vào traning.

Thực hiện trích xuất và tạo các thông tin trên từ ảnh, ta sử dụng lớp ImageExtractor. Đây là một lớp tùy chỉnh:

Nội dung script của ImageExtractor như sau :

import cv2

import numpy as np

import pandas as pd

def image\_extract\_info(image):

    info = []

    contours, hierarchy = cv2.findContours(image, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_NONE)

    contour\_count = len(contours)

    contour\_max\_area = 0

    contour\_max\_index = 0

    for index in range(0, contour\_count):

        contour\_area = cv2.contourArea(contours[index])

        if contour\_area > contour\_max\_area:

            contour\_max\_area = contour\_area

            contour\_max\_index = index

    bounding\_box = cv2.boundingRect(contours[contour\_max\_index])

    image\_crop = image[bounding\_box[1]:bounding\_box[1] + bounding\_box[3],bounding\_box[0]:bounding\_box[2] + bounding\_box[0]]

    image\_resized = cv2.resize(image\_crop, (32,32))

    data = np.array(image\_resized ,dtype=np.uint8)

    vdata = []

    hdata = []

    for i in range(0,32):

        vdata.append(0)

        hdata.append(0)

    for x in range(0,32):

        for y in range(0,32):

            value = data[y][x]

            vdata[x] += value

            hdata[y] += value

    minval = min(vdata)

    for i in range(0,32):

        vdata[i] = vdata[i] - minval

        info.append(vdata[i])

    minval = min(hdata)

    for i in range(0,32):

        hdata[i] = hdata[i] - minval

        info.append(hdata[i])

    return info

    pass

Sau khi nhận được 64 feature tương ứng một ký tự từ ảnh, ta thêm tên thư mục, đường dẫn ảnh vào cuối mảng là phần tử thứ 65 và 66. Tên thư mục chính là Y tương ứng của 64 features.

Cuối cùng, ghi mảng này thành 1 dòng trong file csv. Ta thu được hơn 2000 dòng dữ liệu.

Script :

import cv2

import os

import ImageExtractor

path = 'character\_images/blackwhites'

list\_dir = os.listdir(path)

file\_bin = open('train\_datas/train\_data.csv', 'w')

for dir in list\_dir:

    character = dir

    list\_file = os.listdir(path + '/' + dir)

    for file in list\_file:

        file\_path = path + '/' + character + '/' + file

        image\_raw = cv2.imread(file\_path, cv2.IMREAD\_GRAYSCALE)

        line\_data = ImageExtractor.image\_extract\_info(file\_path, image\_raw)

        line\_data.append(character)

        line\_data.append(file\_path)

        print(line\_data)

        data\_row\_bin = ''

        for val in line\_data:

            data\_row\_bin = data\_row\_bin + str(val) + ','

        data\_row\_bin += '\n'

        file\_bin.write(data\_row\_bin)

file\_bin.close()

print('Finished !')

### 2.1.6 Training.

Ý tưởng ở bước này ta sẽ đọc dữ liệu trong file csv ở bước trước và đồng thời training cho 4 loại mô hình:

Linear Regression,

Logistic Regression,

K Neighbors,

Support Vector Machine.

Chúng ta sẽ sử dụng các lớp tương ứng được hỗ trợ trong thư viện sklearn.

Các model sau khi được training và các nhãn sẽ được lưu xuống các tập bin nhị phân và csv. Mục đích là để sau đó chúng ta có thể tải tập tinh nhị phân, csv và khôi phục lại kết quả đã training và không cần phải training lại nữa.

Đọc dữ liệu csv cho tranning:

duongDan = os.getcwd() + '/train\_datas/train\_data.csv'

duLieu = pd.read\_csv(duongDan)

maTran = duLieu.values

rows, cols = maTran.shape

x\_train = maTran[:,0:cols-3]

labels = maTran[:, cols-3:cols-2]

labels = labels.astype('str')

le = LabelEncoder()

y\_train = le.fit\_transform(labels)

Do 2 cột cuối của dữ liệu csv là Y và đường dẫn ảnh nên khi lấy dữ liệu x\_train ta bỏ 2 cột cuối. Nhãn thì chúng ta sẽ lấy ở cột gần cuối (cột số 65 trong tổng số 66 cột).

Labels ban đầu lấy từ ma trận có 2 kiểu dữ liệu là int dành cho các số từ 0-9 và string cho các ký tự từ A-Z. Trong khi đó, dữ liệu Y đưa vào traning yêu cầu phải là dạng số. Do vậy ta phải thực hiện chuyển đổi 2 loại dữ liệu trong labels về đồng thời kiểu string. Sau đó sẽ encode, ánh xạ giá trị string tương ứng với một số nguyên và dùng số nguyên này đại diện cho Y để đưa vào tính toán.

Thực hiện điều đó dựa vào đoạn script này :

labels = labels.astype('str')

le = LabelEncoder()

y\_train = le.fit\_transform(labels)

Sau khi đã chuẩn bị xong dữ liệu, công việc thứ nhất chúng ta sẽ làm là lưu dữ liệu labels xuống một tập tin csv.

Đoạn script này sẽ thực hiện điều đó:

def save\_labels(labels, y\_train):

    cnt = y\_train.shape[0]

    object\_names = {}

    for i in range(cnt):

        name = labels[i]

        id = y\_train[i]

        key = tuple(name)

        object\_names[key] = id

    file\_object\_writer = open('model\_traineds/objects.csv', 'w')

    data\_row = ''

    for name in object\_names:

        id = object\_names[name]

        if data\_row != '':

            data\_row = data\_row + '\n'

        data\_row += str(id) + ',' + name[0] + ','

    file\_object\_writer.write(data\_row)

    file\_object\_writer.close()

    pass

Chúng ta cần viết thêm một số đoạn script để training cho từng loại model. Sau khi trainning, chúng ta lưu model xuống tập tin nhị phân thông qua sự hỗ trợ của thư viện pickle.

Linear Regression:

def train\_linear\_regression(x\_train, y\_train):

    lnr = LinearRegression()

    lnr.fit(x\_train, y\_train)

    model\_file\_path = os.getcwd() + '/model\_traineds/lnr.model'

    file\_model\_writer = open(model\_file\_path, 'wb')

    pickle.dump(lnr, file\_model\_writer)

    file\_model\_writer.close()

    pass

Logistic Regression:

def train\_logistic\_regression(x\_train, y\_train):

    lgr = LogisticRegression()

    lgr.fit(x\_train, y\_train)

    model\_file\_path = os.getcwd() + '/model\_traineds/lgr.model'

    file\_model\_writer = open(model\_file\_path, 'wb')

    pickle.dump(lgr, file\_model\_writer)

    file\_model\_writer.close()

    pass

K Neighbors:

def train\_knn(x\_train, y\_train):

    knn = KNeighborsClassifier()

    knn.fit(x\_train, y\_train)

    model\_file\_path = os.getcwd() + '/model\_traineds/knn.model'

    file\_model\_writer = open(model\_file\_path, 'wb')

    pickle.dump(knn, file\_model\_writer)

    file\_model\_writer.close()

    pass

Support Vector Machine:

def train\_svm(x\_train, y\_train):

    svm = SVC()

    svm.fit(x\_train, y\_train)

    model\_file\_path = os.getcwd() + '/model\_traineds/svm.model'

    file\_model\_writer = open(model\_file\_path, 'wb')

    pickle.dump(svm, file\_model\_writer)

    file\_model\_writer.close()

    pass

Cuối cùng là một hàm main để thực hiện tất cả các công việc trong 1 lần chạy:

if \_\_name\_\_ == '\_\_main\_\_':

    duongDan = os.getcwd() + '/train\_datas/train\_data.csv'

    duLieu = pd.read\_csv(duongDan)

    maTran = duLieu.values

    rows, cols = maTran.shape

    x\_train = maTran[:,0:cols-3]

    labels = maTran[:, cols-3:cols-2]

    labels = labels.astype('str')

    le = LabelEncoder()

    y\_train = le.fit\_transform(labels)

    save\_labels(labels, y\_train)

    train\_linear\_regression(x\_train, y\_train)

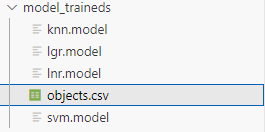
    train\_logistic\_regression(x\_train, y\_train)

    train\_knn(x\_train, y\_train)

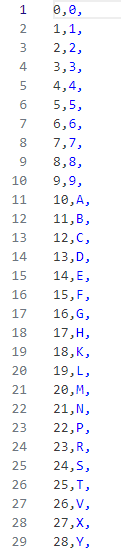
    train\_svm(x\_train, y\_train)

    pass

Sau khi chạy hàm main, 4 model tương ứng được lưu xuống 4 tập tin nhị phân.



kèm một tập tin csv lưu ánh xạ nhãn.



## 2.2 Dự đoán và nhận dạng.

### 2.2.1 Tải lại ánh xạ nhãn và các model

Sau khi có ánh xạ nhãn và các model đươc train tại phần **2.1**, bây giờ, ở giai đoạn test này, chúng ta không cần thực hiện training lại các dữ liệu nữa. Chúng ta sẽ tải trực tiếp dữ liệu đã train và nhãn tương ứng từ các tập tin nhị phân và csv.

Đoạn script này sẽ tải các ánh xạ nhãn từ tập tin csv:

def load\_labels():

    objects = {}

    csv\_objects = pd.read\_csv(os.getcwd() + '/model\_traineds/objects.csv').values

    cnt = csv\_objects.shape[0]

    for i in range(cnt):

        id = csv\_objects[i][0]

        name = csv\_objects[i][1]

        objects[id] = name

    return objects

Dữ liệu khi đọc tập tin csv sẽ được lưu dưới dạng dữ liệu map trên python để giúp dễ dàng thực hiện việc ánh xạ giữa tên nhãn (name) và id. Id là giá trị được ánh xạ với nhãn tương ứng mà đã được đưa vào trong quá trình training.

Đoạn script tiếp theo sẽ tải 4 kiểu modal từ tập tin nhị phân:

def load\_models():

    lnr\_model\_reader = open(os.getcwd() + '/model\_traineds/lnr.model', 'rb')

    lnr = pickle.load(lnr\_model\_reader)

    lnr\_model\_reader.close()

    knn\_model\_reader = open(os.getcwd() + '/model\_traineds/knn.model', 'rb')

    knn = pickle.load(knn\_model\_reader)

    knn\_model\_reader.close()

    lgr\_model\_reader = open(os.getcwd() + '/model\_traineds/lgr.model', 'rb')

    lgr = pickle.load(lgr\_model\_reader)

    lgr\_model\_reader.close()

    svm\_model\_reader = open(os.getcwd() + '/model\_traineds/lgr.model', 'rb')

    svm = pickle.load(svm\_model\_reader)

    svm\_model\_reader.close()

    return lnr, knn, lgr, svm

### 2.2.2 Thiết kế các method detect id từ ảnh ký tự và ảnh chữ số.

Ở phần này, chúng ta thực hiện thiết kế 4 method để thực hiện detect ra id của đối tượng cho từng loại model. Tương tự như bước tạo dữ liệu cho training, chúng ta lại sử dụng lại lớp ImageExtractor để lấy được 64 thông tin của một bức ảnh. Đó chính là dữ liệu test mà chúng ta sẽ đưa vào từng model và lấy ra được kết quả Y tương ứng. Lúc này Y chính là id của một đối tượng nào đó. Có được id, ta tra lại trong map của nhãn xem ký tự tương ứng của nó là gì và trả về. Nếu như không có id như vậy thì sẽ trả về ‘?’ – có nghĩa là đó là một đối tượng không giống với bất kỳ dữ liệu nào được train.

Các đoạn script:

Trong các đoạn script này, để biến info (là một array) thành test\_data (là một ma trận) thì nó sẽ chuyển thành ma trận 64x1. Ta sử dụng phép xoay ma trận T để chuyển lại thành 64x1 phù hợp với input của các model.

Linear Regression:

def detect\_with\_lnr(lnr, image):

    info = ImageExtractor.image\_extract\_info(image)

    test\_data = pd.DataFrame(info).T

    id = lnr.predict(test\_data)[0]

    name = '?'

    if id in objects:

        name = objects[id]

    return name

Logistic Regression:

def detect\_with\_knn(knn, image):

    info = ImageExtractor.image\_extract\_info(image)

    test\_data = pd.DataFrame(info).T

    id = knn.predict(test\_data)[0]

    name = '?'

    if id in objects:

        name = objects[id]

    return name

K Neighbors:

def detect\_with\_lgr(lgr, image):

    info = ImageExtractor.image\_extract\_info(image)

    test\_data = pd.DataFrame(info).T

    id = lgr.predict(test\_data)[0]

    name = '?'

    if id in objects:

        name = objects[id]

    return name

Support Vector Machine:

def detect\_with\_svm(svm, image):

    info = ImageExtractor.image\_extract\_info(image)

    test\_data = pd.DataFrame(info).T

    id = svm.predict(test\_data)[0]

    name = '?'

    if id in objects:

        name = objects[id]

    return name

### 2.2.3 Chuẩn bị ảnh test

Sử dụng ý tưởng và phương pháp giống mục 2.1.2 – Cắt ảnh biển số, ta sẽ cắt các ảnh test từ ảnh test gốc và lưu chung vào một thư mục khác tên là test\_images.



### 2.2.4 Test

Ý tưởng ở bước này là chúng ta sẽ lần lượt đọc ảnh từ thư mục ảnh test, phát hiện các khu vực nghi ngờ là ký tự, cắt ảnh khu vực đó ra và gọi các hàm detect. Kết quả thu được chúng ra sẽ vẽ lên ảnh test và hiển thị chúng lên.

Chúng ta có thể quan sát kết quả detect một cách trực quan.

Đoạn scrip của main sẽ như sau :

if \_\_name\_\_ == '\_\_main\_\_':

    objects = load\_labels()

    lnr, knn, lgr, svm = load\_models()

    path = 'test\_images'

    list\_files = os.listdir(path)

    for file\_name in list\_files:

        file\_path = path + '/' + file\_name

        image\_draw = None

        image\_raw = cv2.imread(file\_path, cv2.IMREAD\_GRAYSCALE)

        image\_raw = cv2.medianBlur(image\_raw, 19)

        image\_draw = cv2.cvtColor(image\_raw, cv2.COLOR\_GRAY2BGR)

        mean = cv2.mean(image\_raw)[0]

        thresh, image\_bin = cv2.threshold(image\_raw, int(mean), 255, cv2.THRESH\_BINARY\_INV)

        contours, hierarchy = cv2.findContours(image\_bin, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

        for contour in contours:

            box = cv2.boundingRect(contour)

            x = box[0]

            y = box[1]

            width = box[2]

            height = box[3]

            if width > height :

                continue

            if width < 40 or width > 400:

                continue

            if height < 200 or height > 500:

                continue

            mat\_bin\_crop = image\_bin[y:y+height, x:x+width]

            white\_point = cv2.countNonZero(mat\_bin\_crop)

            fill = white\_point \* 1.0 / (height \* width)

            if fill < 0.2:

                continue

            name = detect\_with\_knn(knn, mat\_bin\_crop)

            # draw

            cv2.rectangle(image\_draw, box, (255,0,0),2)

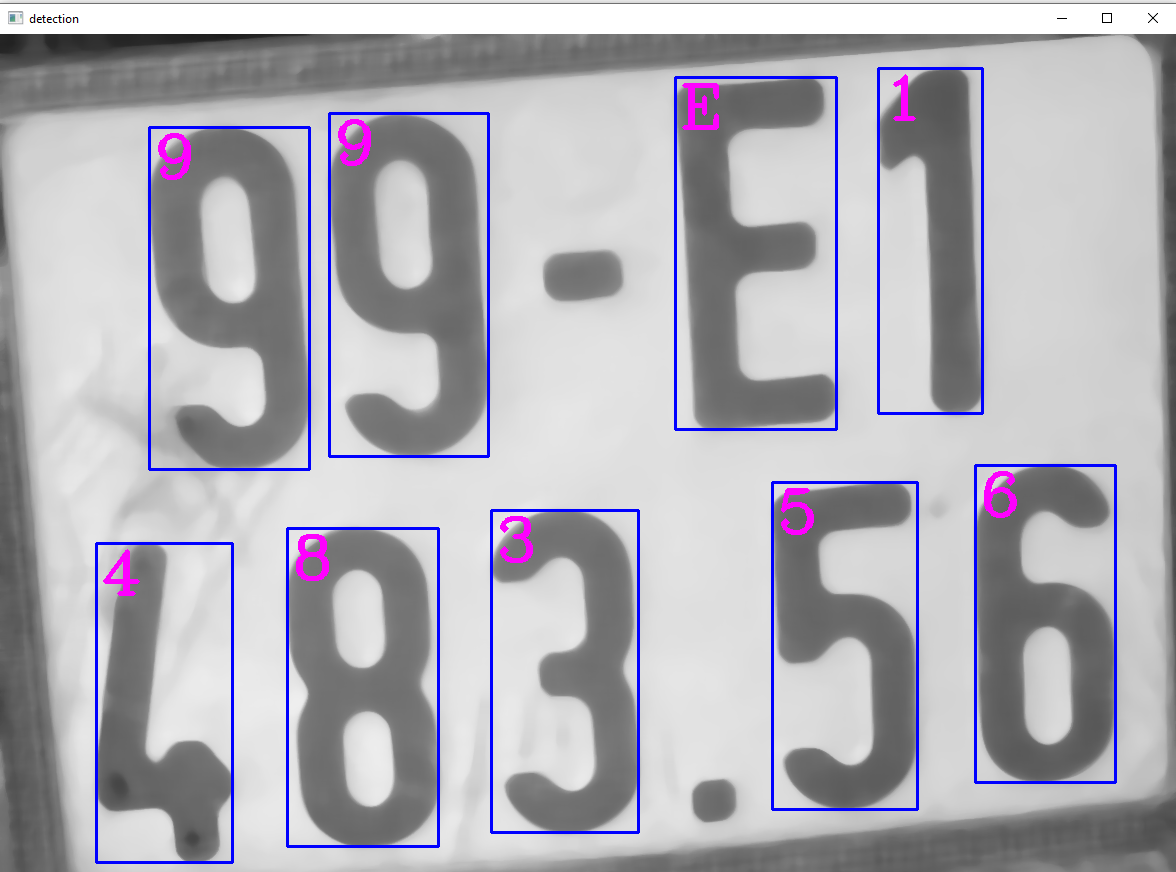
            cv2.putText(image\_draw, str(name), (box[0]+5, box[1]+50), cv2.FONT\_HERSHEY\_COMPLEX, 2, (255,0,255),3)

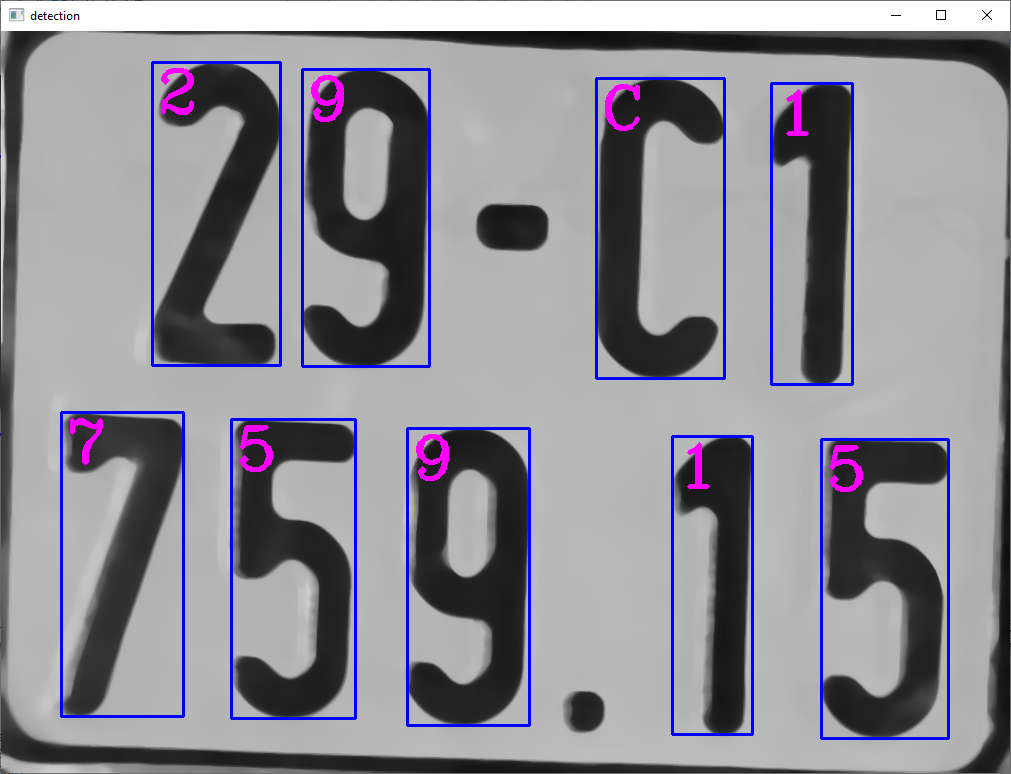
        cv2.imshow('detection', image\_draw)

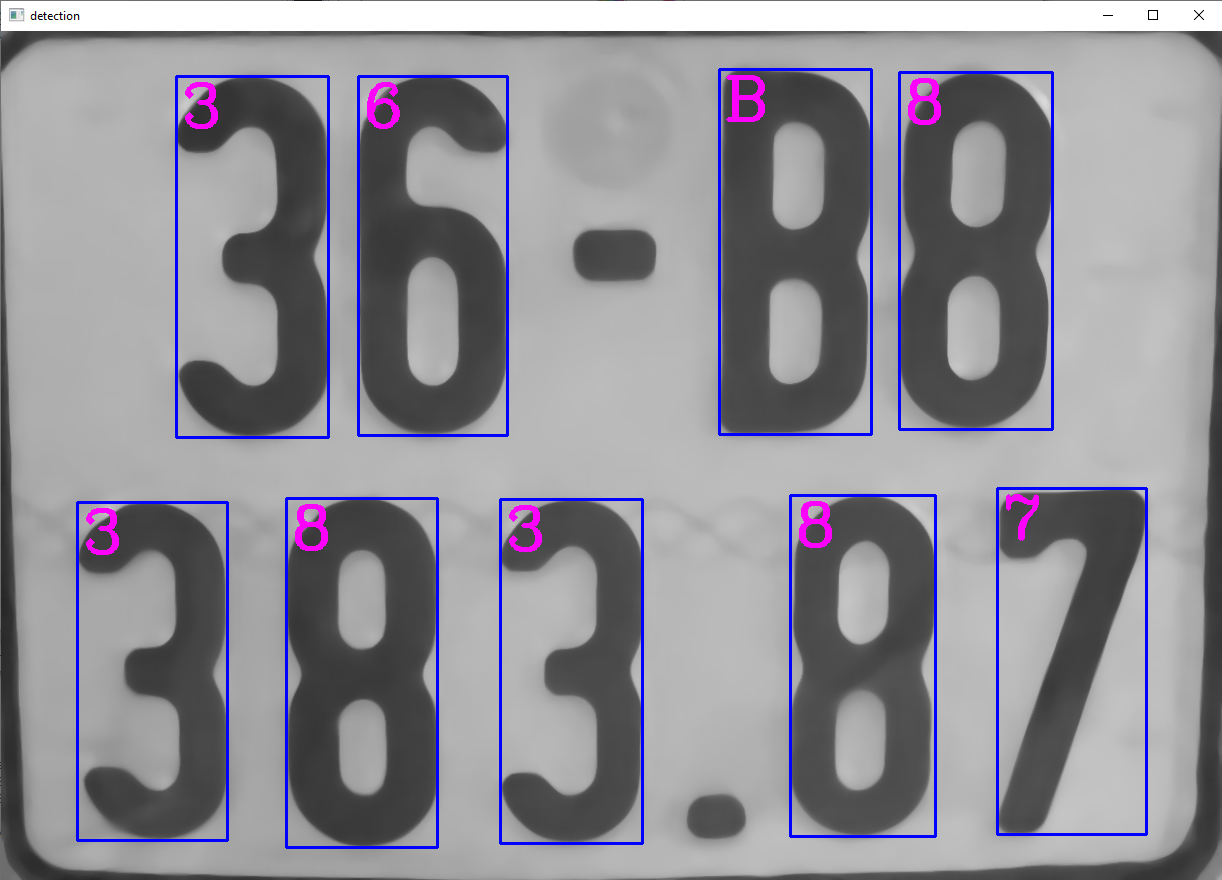
        cv2.waitKey(1000)

    pass

### 2.2.5 Một số kết quả thu được.









# PHẦN III - KẾT LUẬN.

Hồi quy tuyến tính là một phương pháp thống kê để hồi quy dữ liệu với biến phụ thuộc có giá trị liên tục trong khi các biến độc lập có thể có một trong hai giá trị liên tục hoặc là giá trị phân loại. Hồi quy tuyến tính cũng là một trong những thuật toán cơ bản nhất của Machine Learning. Hồi quy tuyến tính là một phương pháp rất đơn giản nhưng đã được chứng minh được tính hữu ích cho một số lượng lớn các tình huống. Cụ thể trong bài này giải quyết việc phát hiện các ký tự trên biển số một cách rất nhanh chóng và độ chính xác tương đối cao nhưng triển khai lại rất đơn giản. Trong ví dụ này vẫn còn tồn tại việc mất cân bằng dữ liệu và việc chuẩn hóa dữ liệu chưa được tối ưu. Nếu như việc chuẩn hóa và xử lý mất cân bằng tốt thì có thể cho kết quả tốt hơn.