**Lời nói đầu   
Tôi là (tên bạn)**

123 TÊN ĐƯỜNG PHỐ

QUẬN/HUYỆN, TỈNH/THÀNH PHỐ

**(+84) 000-0000**

**NO\_REPLY@EXAMPLE.COM**

**Kỹ năng**

Chèn văn bản vào đây Chèn văn bản vào đây Chèn văn bản vào đây Chèn văn bản vào đây Chèn văn bản vào đây.

**Kinh nghiệm**

THÁNG NĂM 20XX - HIỆN TẠI

**Tên công ty, Địa điểm** *- Chức danh*

* Chèn văn bản vào đây Chèn văn bản vào đây.
* Chèn văn bản vào đây .
* Chèn văn bản vào đây Chèn văn bản vào đây Chèn văn bản vào đây.

THÁNG NĂM 20XX - THÁNG NĂM 20XX

**Tên công ty, Địa điểm** *- Chức danh*

* Chèn văn bản vào đây Chèn văn bản vào đây.
* Chèn văn bản vào đây.

THÁNG NĂM 20XX - THÁNG NĂM 20XX

**Tên công ty, Địa điểm** *- Chức danh*

* Chèn văn bản vào đây Chèn văn bản vào đây.
* Chèn văn bản vào đây Chèn văn bản vào đây.
* Chèn văn bản vào đây.

**Học vấn**

THÁNG NĂM 20XX - THÁNG NĂM 20XX

**Tên trường, Địa điểm** *- Trình độ*

Chèn văn bản vào đây Chèn văn bản vào đây Chèn văn bản vào đây Chèn văn bản vào đây Chèn văn bản vào đây.

**Thành tích**

Chèn văn bản vào đây Chèn văn bản vào đây.

Chèn văn bản vào đây.

**PHỤ LỤC**

**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.

**1.2 Giải pháp và lý thuyết.**

**PHẦN II - THỰC HIỆN.**

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

**2.1.1 Chụp ảnh.**

**2.1.2 Cắt ảnh và đánh nhãn.**

Sau quá trình thu thập ảnh chụp thực tế rất nhiều biển số, ta cần thực hiện cắt từng chữ cái và chữ số từ những ảnh đã chụp được. Đầu tiên, sử dụng một số hỗ trợ từ opencv - thư viện thị giác máy tính nổi tiếng – để thực hiện sàng lọc và cắt các đối tượng nghi ngờ là ảnh của chữ số và tạo thành từng tập tin ảnh riêng lẻ. Sau đó lưu chúng vào một thư mục chung.

import cv2

import os

import numpy as np

import csv

# liệt kê các tập tin ảnh chụp từ thư mục 'raw\_images'

path = 'raw\_images'

list\_file = os.listdir(path)

# file index dùng để tạo tên file khi lưu từng ảnh ký tự

file\_index = 0

# bắt đầu duyệt từng tập tin

for file in list\_file:

# đọc ảnh xám

image\_raw = cv2.imread(path + '/' + file, cv2.IMREAD\_GRAYSCALE)

# làm mờ ảnh để lọc nhiễu

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

# lấy giá trị độ sáng trung bình của ảnh.

# mục đích để nhị phân hóa ở bước tiếp theo

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

# nhị phân hóa

\_, image\_bin = cv2.threshold(image\_raw, int(mean\_raw)+50, 255, cv2.THRESH\_BINARY)

# phát hiện các đường biên trong ảnh.

# mục đích phát hiện các đường biên để dự đoán khu vực có khả năng chứa một đối tượng nào đó

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

# bắt đầu duyệt toàn bộ các đường biên tìm được

for contour in contours:

# xác định một vùng không gian hình chữ nhật chứa đường biên.

box = cv2.boundingRect(contour)

# sau đó trích xuất ra từng thông số của khu vực đó bao gồm x,y, width, height

x = box[0]

y = box[1]

width = box[2]

height = box[3]

# ở bước tiếp theo này, ta kiểm tra tính hợp lệ của đường biên dựa vào các thông số.

# mục đích ta sẽ lọc đi các vùng có khả năng lớn không phải là biển số

if width < height :

   continue

if width < 500 or width > 1000:

   continue

scale = float(width) / float(height)

if scale < 1.1 or scale > 2:

   continue

# Sau bước lọc, ta cắt ảnh trong khu vực nhận diện từ ảnh xám

image\_plate = image\_raw[y:y+height, x:x+width]

# thực hiện quá trình nhị phân hóa.

image\_plate\_mean = cv2.mean(image\_plate)[0]

\_, image\_plate\_bin = cv2.threshold(image\_plate, image\_plate\_mean,255, cv2.THRESH\_BINARY\_INV)

# tiếp tục thực hiện phát hiện đường biên trong ảnh biển số

plate\_contours, \_ = cv2.findContours(image\_plate\_bin, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

for plate\_contour in plate\_contours:

character\_box = cv2.boundingRect(plate\_contour)

character\_x = character\_box[0]

character\_y = character\_box[1]

character\_width = character\_box[2]

character\_height = character\_box[3]

character\_size\_raito = character\_height / character\_width

# lọc sơ bộ vùng chứa đường biên

if character\_size\_raito < 1.5 or character\_size\_raito > 5:

   continue

if character\_width < 50 or character\_width > 200:

   continue

# thực hiện cắt ảnh nghi ngờ là chữ số từ ảnh biển số.

# và thực hiện nhị phân hóa

image\_character = image\_plate[character\_y:character\_y+character\_height, character\_x:character\_x+character\_width]

image\_character\_mean = cv2.mean(image\_character)[0]

\_, image\_character\_bin = cv2.threshold(image\_character, image\_character\_mean, 255, cv2.THRESH\_BINARY\_INV)

# quy chuẩn lại kích thước ảnh thành 64x64

image\_character\_bin = cv2.resize(image\_character\_bin, (64,64))

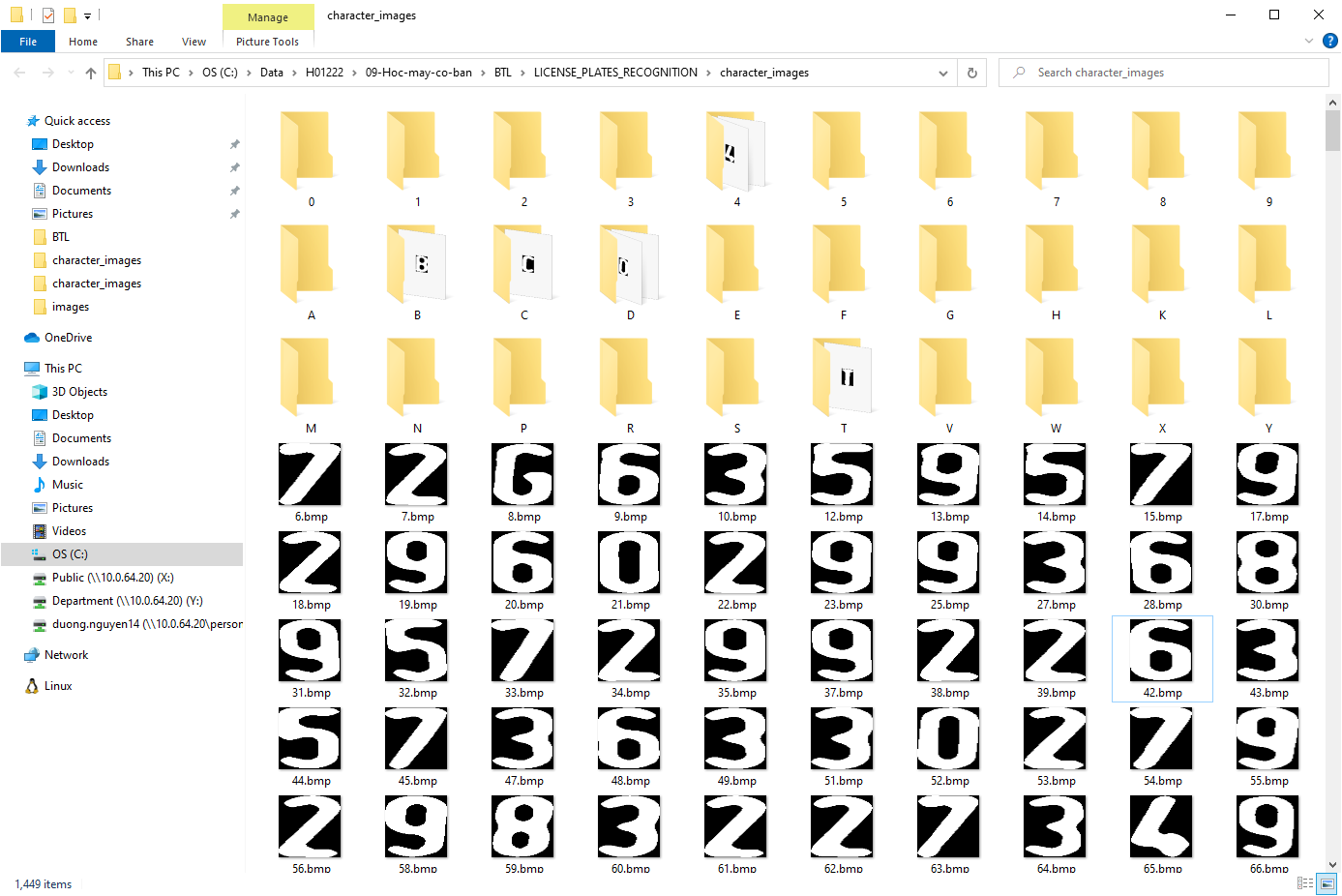
# và lưu nó vào tập tin

file\_index += 1

cv2.imwrite('character\_images/' + str(file\_index) + '.bmp', image\_character\_bin)

print('Finished !')

Sau khi chạy đoạn code phía bên trên, trong thư mục character\_images có nhiều ảnh được tạo ra. Tuy nhiên, trong những ảnh này, có những ảnh không phải chứa ký tự chữ và số. Ta phải xóa nó đi bằng cách thủ công. Đồng thời sau đó ta cũng phải phân nó vào các thư mục có tên tương ứng.



**2.1.3 Tạo dữ liệu training.**

Sau khi có được cách ảnh và được phân lớp vào trong các thư mục có tên tương ứng, bằng cách chạy một đoạn script bên dưới để phân tích và tạo ra tập tin csv chứa dữ liệu các ảnh được mô tả dưới dạng một array. Mỗi dòng trong file tương ứng một ảnh.

import cv2

import os

import numpy as np

import csv

path = 'train\_images'

list\_dir = os.listdir(path)

file\_bin = open('train\_datas/number\_data\_bin.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

mat = cv2.imread(file\_path)

width = mat.shape[0]

height = mat.shape[1]

raito = width / height

mat\_resize = cv2.resize(mat, (24,48))

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

median = mat\_gray.mean() + 30

\_, mat\_bin = cv2.threshold(mat\_gray, median, 255, cv2.THRESH\_BINARY\_INV)

# generate from bin

byte\_array\_bin = np.asarray(mat\_bin).reshape(-1)

data\_row\_bin = ''

for val in byte\_array\_bin:

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

data\_row\_bin += character + '\n'

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

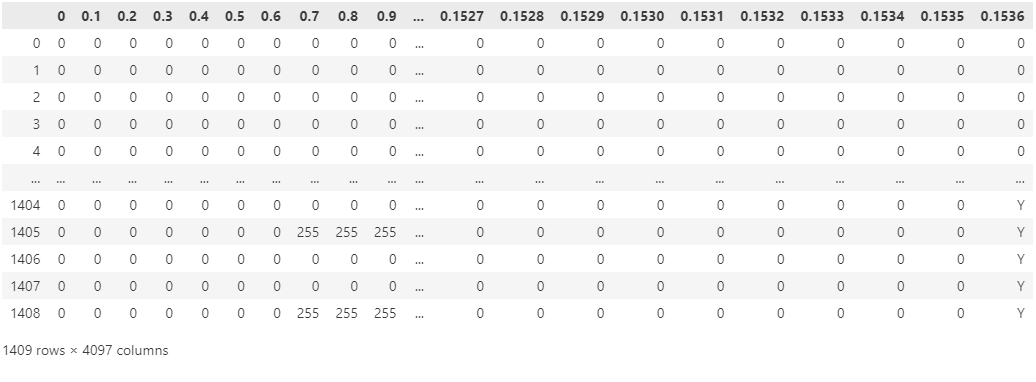
# cv2.imshow('t', mat\_bin)

# cv2.waitKey(1000)

file\_bin.close()

print('Finished !')

Kết thúc, một tập tin có tên “number\_data\_bin.csv” được tạo ra. Nếu dùng pandas tải nội dung file này ta có dữ liệu như bên dưới:



Kích thước file csv là 1409 x 4097, tức là đã có 1049 bức ảnh đã được liệt kê. Bởi vì mỗi bức ảnh được cố định kích thước là 64x64 nên sẽ sinh ra 4096 cột features, kèm thêm một cột values cuối cùng là 4097 cột.

**2.2 Training**

**2.4.1 Training cho Linear Regression.**

import pandas as pd

import os

from sklearn.preprocessing import LabelEncoder

from sklearn.linear\_model import LinearRegression

import pickle

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

duLieu = pd.read\_csv(duongDan)

duLieu

maTran = duLieu.values

rows, cols = maTran.shape

x\_train = maTran[:,0:cols-1]

labels = maTran[:, cols-1:cols]

labels = labels.astype('str')

le = LabelEncoder()

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

lnr = LinearRegression()

lnr.fit(x\_train, y\_train)

b = lnr.intercept\_

w = lnr.coef\_

b , w , w.shape

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()

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/lnr.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()

* + 1. **Training cho KNN.**

import pandas as pd

import os

from sklearn.preprocessing import LabelEncoder

from sklearn.neighbors import KNeighborsClassifier

import pickle

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

duLieu = pd.read\_csv(duongDan)

duLieu

maTran = duLieu.values

rows, cols = maTran.shape

x\_train = maTran[:,0:cols-1]

labels = maTran[:, cols-1:cols]

labels = labels.astype('str')

le = LabelEncoder()

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

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()

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/knn.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()

* + 1. **Training cho Logistic Regression.**

import pandas as pd

import os

from sklearn.preprocessing import LabelEncoder

from sklearn.linear\_model import LogisticRegression

import pickle

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

duLieu = pd.read\_csv(duongDan)

duLieu

maTran = duLieu.values

rows, cols = maTran.shape

x\_train = maTran[:,0:cols-1]

labels = maTran[:, cols-1:cols]

labels = labels.astype('str')

le = LabelEncoder()

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

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()

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/lgr.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()

* + 1. **Training cho Support Vector Machine.**

import pandas as pd

import os

from sklearn.preprocessing import LabelEncoder

from sklearn.svm import SVC

import pickle

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

duLieu = pd.read\_csv(duongDan)

duLieu

maTran = duLieu.values

rows, cols = maTran.shape

x\_train = maTran[:,0:cols-1]

labels = maTran[:, cols-1:cols]

labels = labels.astype('str')

le = LabelEncoder()

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

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()

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/svm.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()

**2.3 Xây dựng các bộ dự đoán.**

**2.5.1 Bộ dự đoán dựa trên Linear Regression.**

import pandas as pd

import os

import numpy as np

from matplotlib import pyplot

import scipy.optimize as opt

from sklearn.preprocessing import LabelEncoder

from sklearn.linear\_model import LinearRegression

import pickle

class LNRDetector:

    model = None

    objects = {}

    def \_\_init\_\_(self):

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

        file\_model\_reader = open(model\_file\_path, 'rb')

        self.model = pickle.load(file\_model\_reader)

        self.model.intercept\_

        self.model.coef\_

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

        cnt = csv\_objects.shape[0]

        for i in range(cnt):

            id = csv\_objects[i][0]

            name = csv\_objects[i][1]

            self.objects[id] = name

        pass

    def detect(self, image, size):

        # index = 250

        # x\_test = duLieu[index:index+1,0:cols-1]

        # result = linear\_regression.predict(x\_test)

        # round(result[0])

        pass

**2.5.2 Bộ dự đoán dựa trên KNN.**

**2.5.3 Bộ dự đoán dựa trên Logistic Regression.**

**2.5.4 Bộ dự đoán dựa trên Support Vector Machine.**

**2.6 Dự đoán và nhận dạng.**

**PHẦN III - KẾT LUẬN.**