**BÁO CÁO BÀI TẬP 8**

**Nhóm 18**

1. Lương Hữu Phú Lộc – 1511844
2. Phạm Ngọc Khôi Nguyên – 1512221
3. Nguyễn Trọng Phúc – 1512534
4. Mai Thiện Quang – 1512640
5. **Yêu cầu**

Dùng các thuật toán phù hợp SVM, KNN … nhận dạng số xe máy **cho các hình dưới đây. Phân tích độ chính xác và thời gian xử lý**

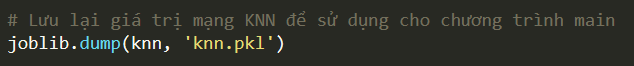


1. **Thực hiện**
2. **Huấn luyện mạng**

* Phương pháp: sử dụng mạng KNN của thư viện Scikit-learn của Python
* Data nhóm lấy từ hình font chữ trong file bài tập (như hình dưới), với mỗi kí tự nhóm lấy 20 mẫu với mỗi mẫu có góc quay từ -5 đến 5 độ so với hình gốc
* Data này chia làm 2 phần, 10 mẫu dùng để huấn luyện mạng, 10 mẫu dùng để test



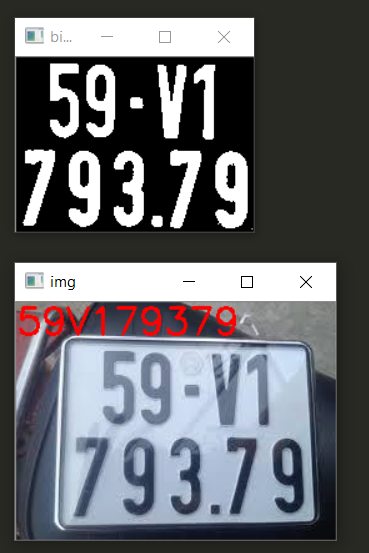
* Các trích đặc trưng từ ảnh: với mỗi ảnh, resize kích thước thành 20x50, sau đó chuyển sang ảnh nhị phân và biến đổi thành vector hàng 1x1000, mỗi cột của vector tương ứng với 1 pixel của ảnh ban đầu.
* Sau khi training, nhóm sử dụng công cụ **joblib** của thư viện Scikit-learn để lưu lại giá trị mạng để có thể sử dụng sau này mà không phải mất thời gian training lại.



1. **Nhận diện biển số**

* Để tách biển số nhóm sử dụng phương pháp threshold do trong hình biển số có màu trắng tách biệt với nền
* Đối với những hình biển số bị nghiêng như hình 4, hình 5, nhóm sử dụng phương pháp biến đổi hình học **warpPerspective** để đưa hình biển số về lại thẳng hướng đối diện người nhìn
* Sau khi nhận diện được biển số, tách từng chữ số trên biển sau đó trích đặc trưng giống như đặc trưng khi huấn luyện mạng ban đầu và dùng mạng KNN để xác định kí tự

1. **Kết quả**

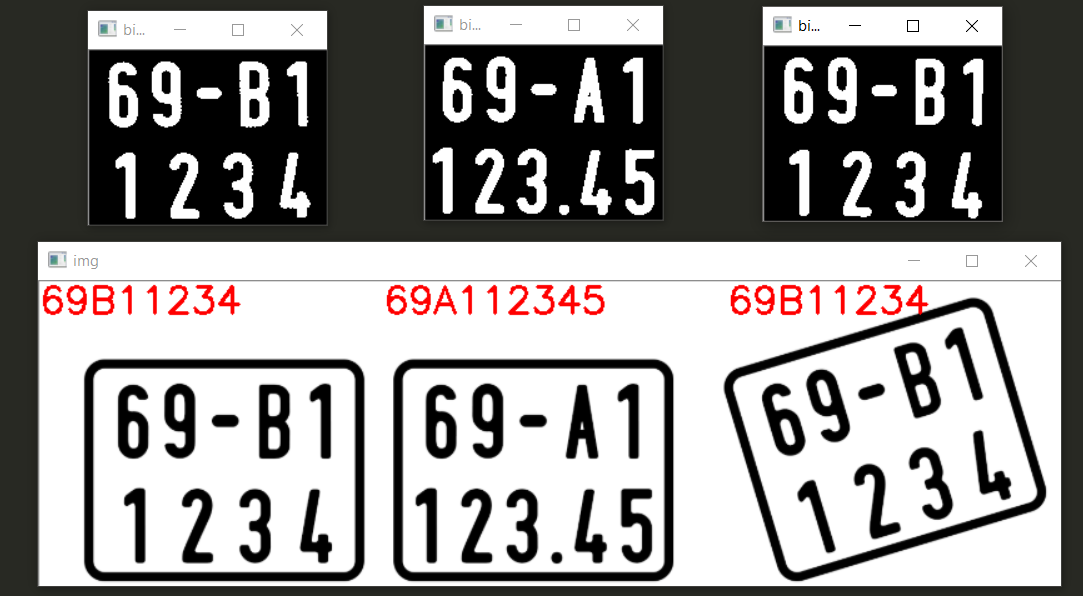
Bên trái là ảnh chứa biển số, bên phải là ảnh nhị phân sau khi biến đổi hình học và kết quả nhận diện

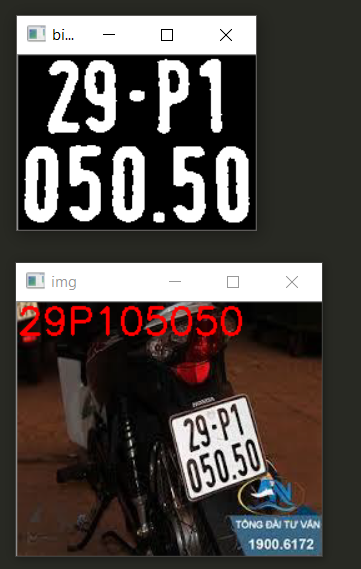
****



****

****





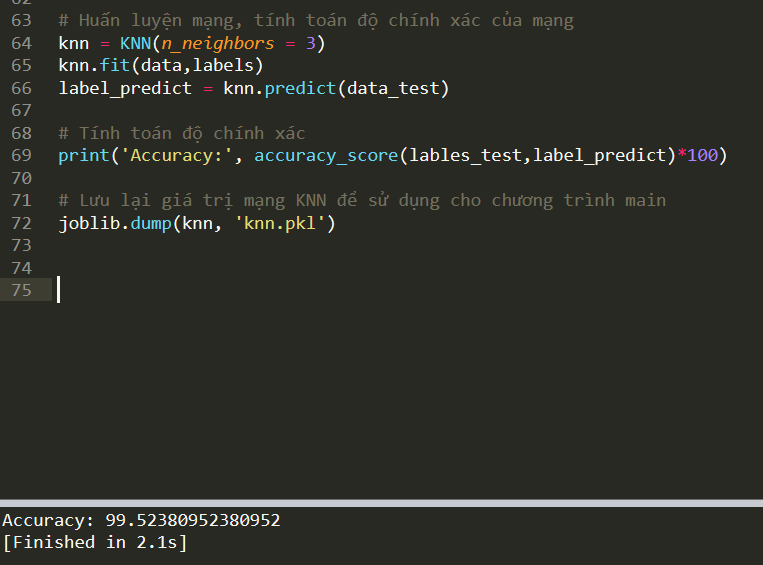
****



****

1. **Nhận xét:**

* Phương pháp sử dụng tương đối đơn giản và không hoàn toàn đúng trong mọi trường hợp vì mức ngưỡng threshold để nhận diện biển số nhóm phải điều chỉnh để phù hợp với hình
* Về phương pháp nhận diện, độ chính xác tính toán được sau khi training mạng là:



* **Độ chính xác** là 99.52% rất cao, nhưng tuy nhiên đây là do data của nhóm tự tạo tương đối sạch, không bị nhiễu, mất nét, số lượng data cũng ít (chỉ 10 mẫu train và 10 mẫu test), khi áp dụng vào thực tế kết quả sẽ có thể không chính xác 🡪 giải pháp là thay đổi data là các số lấy từ những biển số trong thực tế
* **Về thời gian xử lý:** bản chất mạng KNN không phải là training hay tính toán gì phức tạp mà chỉ là dựa vào những điểm xung quanh để phân loại 1 điểm mới nên thời gian xử lý tương đối nhanh (như trên hình chương trình xây dựng mạng KNN hoàn thành chỉ sau 2.1s), ở chương trình chính do load lại mạng KNN đã có sẵn nhờ hàm **joblib** nên tốc độ xử lý nhanh.

1. **Chương trình**

* Trong file nén báo cáo gồm các 2 file python là knn\_train.py và main.py trong đó knn\_train.py để tạo mạng KNN, còn file main.py là chương trình nhận dạng
* Folder Data là hình của các kí tự dùng để train và test.
* File knn.pkl là mạng knn được lưu lại sau khi training, ta có thể load lại file này trong chương trình khác để không tốn thời gian training lại.
* Chương trình main có load lại file này, do đó khi chạy file main.py cần chú ý đường dẫn đến file knn.pkl

Chương trình tạo mạng KNN, trong chương trình có sử dụng **thư viện Scikit-learn**

**import** **numpy** **as** **np**

**import** **cv2**

**import** **os**

**from** **sklearn.externals** **import** joblib

**from** **sklearn.neighbors** **import** KNeighborsClassifier **as** KNN

**from** **sklearn.metrics** **import** accuracy\_score

*# Sử dụng KNN của thư viện Scikit-learn*

*# Data gồm các kí tự số từ 0-9, chữ từ A-Z*

*# mỗi kí tự gồm 20 hình với độ nghiêng khác nhau*

*# Data này do nhóm tự tạo dựa trên hình chứa font*

*# trong file BTVN08*

path = 'Data'

data = data\_test = np.empty((0, 20\*50))

labels = []

lables\_test = []

*# Lần lượt load từng image trong mỗi folder*

*# để trích đặc trưng cho mạng KNN*

*# Data chia làm 2 phần: 10 kí tự để train*

*# và 10 kí tự dùng để test*

*# Nhãn của mỗi đặc trưng tương ứng là tên folder chứa ảnh*

**for** label **in** os.listdir(path):

imgPath = path +'/' + label

i = 0

**for** image **in** os.listdir(imgPath):

img = cv2.imread(path +'/' + label + '/' + image,1)

img = cv2.resize(img,(20,50),cv2.INTER\_LINEAR)

gray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY);

gray = cv2.medianBlur(gray,5)

ret, binary = cv2.threshold(gray,100,255,cv2.THRESH\_BINARY\_INV)

*# Chuyển hình thành vector 1x1000 làm đặc trưng*

H = binary.reshape((1,20\*50))

**if** i <10:

data = np.append(data,H,0)

labels.append(ord(label))

**else**:

data\_test = np.append(data\_test,H,0)

lables\_test.append(ord(label))

i += 1

*# Reshape lại label theo cột để tương ứng với đặc trưng*

data = np.asarray(data,np.float32)

labels = np.asarray(labels,np.float32)

data\_test = np.asarray(data\_test,np.float32)

lables\_test = np.asarray(lables\_test,np.float32)

*# Huấn luyện mạng, tính toán độ chính xác của mạng*

knn = KNN(n\_neighbors = 3)

knn.fit(data,labels)

label\_predict = knn.predict(data\_test)

*# Tính toán độ chính xác*

**print**('Accuracy:', accuracy\_score(lables\_test,label\_predict)\*100)

*# Lưu lại giá trị mạng KNN để sử dụng cho chương trình main*

joblib.dump(knn, 'knn.pkl')

**Chương trình chính**

**import** **numpy** **as** **np**

**import** **cv2**

**from** **sklearn.externals** **import** joblib

*# Sử dụng công cụ joblib trong thư viện scikit-learn*

*# để load bộ phân loại knn đã training từ trước*

*# Chú ý đường dẫn đến file knn.pkl*

*# Trong project này, file knn.pkl nằm trong cùng 1 thư mục với file main*

*# nếu không, phải thay đổi đường dẫn đến folder chứa file này*

knn = joblib.load('knn.pkl')

*# Tên file hoặc đường dẫn đến file muốn nhận dạng*

image\_name = 'img5.png'

*# Hàm con sắp xếp 4 điểm theo thứ tự:*

*# top-left, top-right, bottom-right, bottom-left*

**def** order\_points(pts):

rect = np.zeros((4, 2), dtype = "float32")

*# top-left có tổng 2 toạ độ (x,y) nhỏ nhất*

*# ngược lại bottom-right có tổng lớn nhất*

s = pts.sum(axis = 1)

rect[0] = pts[np.argmin(s)]

rect[2] = pts[np.argmax(s)]

*# top-right có hiệu giữa 2 toạ độ (x,y) nhỏ nhất*

*# ngược lại, bottom-left có hiệu lớn nhất*

diff = np.diff(pts, axis = 1)

rect[1] = pts[np.argmin(diff)]

rect[3] = pts[np.argmax(diff)]

**return** rect

*# Hàm con biến đổi hình học*

*# dùng để đưa biển số xe bị nghiêng về lại đối diện người nhìn*

**def** four\_point\_transform(image, pts):

rect = order\_points(pts)

(tl, tr, br, bl) = rect

*# Tính chiều rộng của ảnh sau khi biến đổi*

*# Chiều rộng bằng maximum của khoảng cách giữa (bottom-left, bottom-right) và (top-left, top-right)*

widthA = np.sqrt(((br[0] - bl[0]) \*\* 2) + ((br[1] - bl[1]) \*\* 2))

widthB = np.sqrt(((tr[0] - tl[0]) \*\* 2) + ((tr[1] - tl[1]) \*\* 2))

maxWidth = max(int(widthA), int(widthB))

*# Tính chiều cao của ảnh sau khi biến đổi*

*# Chiều cao bằng maximum của khoảng cách giữa (top-right, bottom-right) và (top-left, bottom-left)*

heightA = np.sqrt(((tr[0] - br[0]) \*\* 2) + ((tr[1] - br[1]) \*\* 2))

heightB = np.sqrt(((tl[0] - bl[0]) \*\* 2) + ((tl[1] - bl[1]) \*\* 2))

maxHeight = max(int(heightA), int(heightB))

dst = np.array([

[0, 0],

[maxWidth - 1, 0],

[maxWidth - 1, maxHeight - 1],

[0, maxHeight - 1]], dtype = "float32")

*# Tính ma trận biến đổi hình học*

*# sử dụng ma trận này để đưa hình bị nghiêng về lại thẳng,*

*# đối diện người nhìn*

M = cv2.getPerspectiveTransform(rect, dst)

warped = cv2.warpPerspective(image, M, (maxWidth, maxHeight))

**return** warped

*# Hàm con dùng để xác định vị trí của biển số,*

*# sau đó biến đổi hình học nếu biển số bị nghiêng,...*

*# Kết quả của hàm trả về là ma trận ảnh chứa biển số*

*# và số lượng biển số trong hình*

**def** plate\_detect(img):

gray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

gray = cv2.bilateralFilter(gray,9,25,10)

ret, binary = cv2.threshold(gray,157,255,cv2.THRESH\_BINARY)

i = 0

thresh = np.zeros((10,140,190),np.uint8)

\_, contours, hierachy = cv2.findContours(binary,cv2.RETR\_TREE,cv2.CHAIN\_APPROX\_SIMPLE)

*# Sắp xếp contour theo thứ tự trái sang phải*

sorted\_ctrs = sorted(contours, key=**lambda** contours: cv2.boundingRect(contours)[0])

**for** contour **in** sorted\_ctrs:

**if** cv2.contourArea(contour) > 500 **and** cv2.contourArea(contour) < 35000:

(x,y,w,h) = cv2.boundingRect(contour)

epsilon = 0.1\*cv2.arcLength(contour,True)

approx = cv2.approxPolyDP(contour,epsilon,True)

**if** len(approx) == 4:

approx = approx.reshape((4,2))

warped = four\_point\_transform(img,approx)

warped = cv2.resize(warped,(190,140),cv2.INTER\_CUBIC)

warped = cv2.cvtColor(warped,cv2.COLOR\_BGR2GRAY)

ret, binary = cv2.threshold(warped,155,255,cv2.THRESH\_BINARY\_INV)

thresh[i] = binary

i += 1

**return** thresh, i

*# Chương trình chính:*

img = cv2.imread(image\_name,1)

*# Khởi tạo ma trận chứa hình biển số và số lượng*

plates = np.zeros((10,140,190),np.uint8)

*# Xác định biển số và số lượng biển số:*

plates, n = plate\_detect(img)

*# Với mỗi biển số, sắp xếp các kí tự chữ,số*

*# theo thứ tự từ trái qua phải,*

*# hàng trên trước rồi đến hàng dưới*

**for** i **in** range(0,n):

binary = plates[i]

cv2.imshow('binary'+np.str(i),binary)

\_, contours, hierachy = cv2.findContours(binary,cv2.RETR\_EXTERNAL,cv2.CHAIN\_APPROX\_SIMPLE)

cnt1 = []

cnt2 = []

**for** contour **in** contours:

**if** cv2.contourArea(contour) > 350:

**if** cv2.boundingRect(contour)[1] < 50:

cnt1.append(contour)

**else**:

cnt2.append(contour)

*# Sắp xếp các kí tự chữ, số theo thứ tự*

sorted\_ctrs = sorted(cnt1, key=**lambda** cnt1: cv2.boundingRect(cnt1)[0]) + sorted(cnt2, key=**lambda** cnt2: cv2.boundingRect(cnt2)[0])

*# Sau khi sắp xếp, lần lượt xác định từng kí tự*

*# dựa trên mạng knn trước đó*

string = ""

**for** contour **in** sorted\_ctrs:

(x,y,w,h) = cv2.boundingRect(contour)

digit = binary[y:y+h,x:x+w]

digit\_resize = cv2.resize(digit,(20,50),cv2.INTER\_LINEAR)

*# Trích đặc trưng của mỗi hình*

feature = digit\_resize.reshape((1,20\*50))

feature = np.float32(feature)

*# Sử dụng mạng knn để tìm kí tự,*

*# kết quả lưu lại vào biến string*

result = knn.predict(feature)

strCurrentChar = str(chr(int(result)))

string += strCurrentChar

**print**('Result:',string)

*# Ghi kết quả lên hình tương ứng*

cv2.putText(img, string, (0+275\*i,25), cv2.FONT\_HERSHEY\_SIMPLEX, 1,(0,0,255),2)

cv2.imshow('img',img)

cv2.waitKey(0)