

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

---o0o---

Báo cáo Đồ án cuối học kì
Đề tài: Phát hiện và trích xuất thông tin
biển số xe

Thành viên nhóm:

Nguyễn Đức Hoan - 17520501

Trịnh Việt Hoàng – 17520522

TP. Hồ Chí Minh, tháng 8/2020

Mục lục

Mục lục.....	2
1. Giới thiệu bài toán:	3
2. Cách giải quyết:	3
a) Grayscale Conversion:	5
b) Threshold Binary, Invetered:.....	5
c) Biến đổi hình thái:	6
d) Tìm và sắp xếp các contour:.....	8
3. Mô tả dữ liệu:	10
a) Pre-trained model của mạng WPOD-Net :.....	10
b) Dataset của bộ kí tự chữ và số trên biển báo:.....	14
4. Kết quả đạt được:	16
5. Khó khăn:	16
6. Hướng phát triển:	17
7. Tư liệu tham khảo:	17
8. File Google Colab thực thi chương trình của nhóm:	17

1. Giới thiệu bài toán:

Hiện nay chúng ta đang sống trong một thế giới rộng lớn, nơi mà có rất nhiều thông tin hiện hữu xung quanh chúng ta. Do đó dẫn đến một nhu cầu tất yếu, đó là làm sao để có thể lưu trữ và quản lý lượng thông tin khổng lồ này, và tận dụng nó để khiến cho cuộc sống này trở nên tốt đẹp hơn. Có thể kể đến trong lĩnh vực giao thông công cộng, khi mà mỗi người trong chúng ta hằng ngày khi bước chân ra khỏi nhà để đi làm, đi gặp bạn bè hay vui chơi đều không thể không bắt gặp hình ảnh những đoàn xe máy, xe hơi nối đuôi nhau đi trên đường. Mỗi phương tiện tham gia giao thông đều có một mã định danh, hay chúng ta thường gọi là biển số xe. Thu thập và quản lý tự động thông tin về số xe của từng phương tiện đi trên 1 con đường sẽ giúp cho các nhà quản lý quy hoạch giao thông có thể đếm được số lượng phương tiện, xuất xứ của xe đó, quản lý gửi đỗ xe, hướng di chuyển của xe cũng như phục vụ cho các công tác an ninh như truy bắt tội phạm, chống trộm,...

Đó là lí do sau ngày đêm suy nghĩ tìm tòi, nhóm đưa đến ý tưởng thiết kế 1 hệ thống tự động nhận diện vùng có biển số xe trong 1 tấm ảnh, và sau đó trích xuất thông tin biển số xe đó dưới dạng các kí tự. Ví dụ 1 tấm ảnh chụp mặt trước của xe hơi sẽ được tự động phát hiện và cắt (crop) ra vùng biển số xe, sau đó từng kí tự của số xe sẽ được trích xuất để lưu trong cơ sở dữ liệu dưới dạng 1 dãy số.

2. Cách giải quyết:

Hệ thống có thể chia ra thành 2 bài toán nhỏ:

1. Nhận diện vùng biển số xe: Đây là bài toán object detection trong lĩnh vực máy học. Nhóm sử dụng mạng WPOD-Net để phát hiện vùng biển số xe. Đây là 1 pre-trained model đã được huấn luyện sẵn cho việc phát hiện vùng biển số.

2. Đọc nội dung in trên biển số: Nhóm sử dụng giải thuật SVM kết hợp cùng 1 số bước tiền xử lí dữ liệu bằng thư viện OpenCV để thực hiện

bài toán Image Classification: từ 1 biển số xe sẽ phân tích từng khu vực chứa kí tự trong biển số, sau đó phân tích tìm ra kí tự ở khu vực đó là kí tự gì, số gì.

Input: Ảnh chụp mặt trước hoặc mặt sau của phương tiện giao thông.

Output: Tô sáng quanh vùng chứa biển số trong ảnh và in ra số xe của phương tiện đó.

Model pre-trained của mạng WPOD-net được huấn luyện với ảnh màu ở các góc chụp và điều kiện ánh sáng được nhóm mô tả ở phần III, do đó bước này nhóm cho input là 1 ảnh màu bất kì chụp mặt trước hoặc mặt sau của xe hơi, output thì mạng WPOD-Net sẽ crop (cắt) ra được vùng có biển số xe và tự động biến đổi hình thái học để vùng cắt ra có hình chữ nhật nằm ngang.

Dataset để nhận diện kí tự chữ và số trên biển báo đều ở dạng ảnh binary (chỉ gồm các pixel có giá trị là 0 và 255) khi được huấn luyện bằng SVM để tạo ra model, do đó từ sau bước sử dụng pre-trained model WPOD-Net thì nhóm phải xử lí ảnh chụp vùng cắt ra được và chuyển thành ảnh binary, sau đó cắt từng kí tự trên biển số ra và chạy bài toán Image Classification đối với từng kí tự được cắt ra bằng model đã huấn luyện của bộ kí tự.

Sau đây là các bước xử lí ảnh chụp vùng biển số trước khi đưa vào model SVM:

Trước khi bắt đầu, ta cần áp dụng một số kỹ thuật xử lý để giảm nhiễu (noise) và làm nổi bật các đặc trưng chính (key features) của các kí tự cần nhận dạng.

- Grayscale
- Threshold ra ảnh binary
- Biến đổi hình thái: dilation - giãn nở ảnh
- Tìm contour

a) Grayscale Conversion:

Trong xử lý ảnh, việc chuyển đổi ảnh màu sang ảnh xám là công việc vô cùng phổ biến. Ảnh màu thực chất là tập hợp của nhiều ma trận có cùng kích thước.

Công việc xử lý thông tin trên ảnh sẽ trở nên dễ dàng hơn nếu ta chỉ cần xử lý dữ liệu trên một ma trận số thay vì nhiều ma trận số. Do đó, việc biến đổi ảnh màu về ảnh số (Grayscale Conversion) chính là biến đổi thông tin ảnh về một ma trận số hai chiều duy nhất

Hiện thực: sử dụng câu lệnh:

```
# Áp dụng kỹ thuật Grayscale Conversion để chuyển đổi ảnh biến số về ảnh xám  
gray = cv2.cvtColor(lpImg[0], cv2.COLOR_BGR2GRAY)
```

source : ảnh màu gốc

dest : ảnh xám đã chuyển đổi

mode : độ chuyển đổi màu

Ở đây, ta dùng chế độ CV_BGR2GRAY để chuyển từ ảnh màu từ format RGB (Red - Green - Blue) sang ảnh xám (Gray)

Nhìn vào hình trên, ta có thể thấy rõ số lượng phần cứng ở NISC đã được giảm đi đáng kể, góp phần giảm tiêu hao năng lượng. Ở kiến trúc RISC, lệnh được lưu trong bộ nhớ lệnh (Program Memory – PM) sẽ được giải mã qua bộ decoder và tạo ra các tín hiệu điều khiển truyền trực tiếp vào vào datapath, thay vì phải lưu trung gian giữa các bộ nhớ microcode như ở CISC. Trong VLM thì tối ưu hơn ở chỗ là lệnh gồm các tập lệnh chạy song song trước khi được giải mã và nạp vào datapath. Trong NISC thì lệnh lưu trong bộ nhớ chương trình (PM) sẽ được nạp trực tiếp vào các ngõ vào điều khiển của datapath.

b) Threshold Binary, Invetered:

Ảnh đen trắng thường được ứng dụng trong bài toán phân vùng ảnh (Image Segmentation). Ảnh đen trắng chỉ có hai màu là màu đen với giá trị là 0 và màu trắng với giá trị là 255.

Do đó ta phải tìm cách biến đổi ảnh xám đã được chuyển đổi ở bước trên (hay là một ma trận 2 chiều với mỗi ô (pixel) có giá trị trong khoảng 0-255 thành một ma trận 2 chiều với mỗi ô có giá trị bây giờ chỉ là 0 hoặc 255.

Vì vậy, ở bước này, ta cần đưa ra một ngưỡng (threshold) nhằm mục đích xác định trên ảnh đâu là điểm ảnh màu đen và đâu là điểm ảnh màu trắng.

Ở đây ta dùng kỹ thuật Threshold Binary, Inverted:

$\text{pixel}(x,y) = 0$ nếu $\text{gray}(x,y) > \text{threshold}$

$\text{pixel}(x,y) = 255$ nếu $\text{gray}(x,y) \leq \text{threshold}$

Hiện thực:

```
# Áp dụng Threshold Binary, Inverted để phân tách số và nền
#Simple Thresholding
#binary = cv2.threshold(gray, 172, 255, cv2.THRESH_BINARY_INV)[1]
binary = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY_INV)[1]
```

src: ảnh xám gốc

dst: ảnh đen trắng đã thay đổi

thres: ngưỡng

max_val: ngưỡng trắng

type: loại biến đổi. Ở đây type = 1, tức biến đổi đen trắng ngược với định nghĩa ở trên

Nhóm có thử nghiệm thay đổi tham số như ở phần code được comment đi, và cho ra thông số tối ưu được nhóm sử dụng là mức threshold = 127.

c) Biến đổi hình thái:

Nhóm sử dụng phép toán dilation – giãn nở ảnh để làm tăng kích thước vùng pixel 255, giúp dễ nhận diện contour hơn ở bước sau.

Hiện thực:

```
kernel3 = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))
thre_mor = cv2.morphologyEx(binary, cv2.MORPH_DILATE, kernel3)
```

Mô hình ứng dụng của nhóm khi detect ảnh chụp biển số xe ở góc chính diện hoặc hơi nghiêng thì sẽ sử dụng phép toán này, **nhưng ở trường hợp ảnh biển báo ở ảnh input đầu vào ở góc quá nghiêng hoặc được chụp ở khoảng cách quá xa**, model WPOD-Net sẽ cắt ra được biển số với các kí tự dính sát vào nhau hoặc không rõ ràng, dẫn đến việc nhận diện số contour bị sai:





Lúc này thì nhóm có thử nghiệm không sử dụng phép toán dilation mà sử dụng phép toán erosion để tách các vùng kí tự sao cho giảm sự dính vào nhau, tăng khả năng nhận diện được contour. Tuy nhiên khi làm như vậy thì sẽ dẫn đến 1 số trường hợp ảnh chụp ở góc chính diện của xe nhưng lại không nhận diện được đúng số lượng contour do phép toán erosion là trái ngược với phép toán dilation nên sẽ làm mất đi khả năng giảm nhiễu ở ảnh chụp góc chính diện.

d) Tìm và sắp xếp các contour:

Sử dụng hàm `findContours` của OpenCV để xác định những tọa độ điểm của các ký tự. Hàm này được xây dựng dựa trên lý thuyết đơn giản: contour là tập các điểm liên tục tạo thành đường cong (curve) - boundary mà có đặc điểm chung là các điểm có cùng/gần xấp xỉ một giá trị màu sắc, hoặc cùng mật độ (intensity).

```
cont, _ = cv2.findContours(thre_mor, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
```

`bin_img`: ảnh nhị phân gốc

`type`: dạng contour. Ở bài này dùng `RETR_LIST`

`method`: phương thức. Ở bài này dùng `CHAIN_APPROX_SIMPLE`

Ở đây ta tạo một hàm `sort_contours` duyệt hết các contour tìm được từ trái sang phải. Điều này quan trọng vì ta không chỉ muốn nhận dạng ký tự mà còn mong muốn sắp xếp chúng theo đúng trật tự:


```
# Hàm sắp xếp các contour từ trái sang phải theo đúng trật tự
def sort_contours(cnts):
    reverse = False
    i = 0
    boundingBoxes = [cv2.boundingRect(c) for c in cnts]
    (cnts, boundingBoxes) = zip(*sorted(zip(cnts, boundingBoxes), key=lambda b: b[1][i], reverse=reverse))
    return cnts
```

Bởi vì ta biết rằng height của ký tự có giá trị lớn hơn width nên ta có thể lọc những contours không liên quan bằng cách chỉ chọn những contours có tỉ lệ height/width nằm trong khoảng 1.5-3.5.

Hơn nữa, ta cũng biết rằng mỗi ký tự trong biển số có height lớn hơn 1 nửa chiều cao thật của biển số, vì vậy ta có thể chọn thêm các contour có giá trị height từ 60% giá trị height của biển số.

```
for c in sort_contours(cont):
    (x, y, w, h) = cv2.boundingRect(c)

    # Tỉ lệ h/w dùng để lọc những ký tự không liên quan
    ratio = h/w

    # Chọn các contour thỏa mãn yêu cầu tỉ lệ h/w phải nằm trong
    # khoảng từ 1.5 đến 3.5
    if 1.5 <= ratio <= 3.5:

        # Áp dụng kết quả sau khi áp dụng kĩ thuật ROI
        # để chọn các contour có chiều cao từ 60% chiều cao biển số
        if h/roi.shape[0]>=0.6:

            # Vẽ khung chữ nhật quanh chữ số
            cv2.rectangle(roi, (x, y), (x + w, y + h), (0, 255,0), 2)

            # Tách số và resize
            curr_num = thre_mor[y:y+h,x:x+w]
            curr_num = cv2.resize(curr_num, dsize=(digit_w, digit_h))

            # Áp dụng ROI - Region of Interest
            _, curr_num = cv2.threshold(curr_num, 30, 255, cv2.THRESH_BINARY)
            curr_num = np.array(curr_num, dtype=np.float32)
            curr_num = curr_num.reshape(-1, digit_w * digit_h)
            crop_characters.append(curr_num)
```

Sau khi xử lý các contours ở trên, trích xuất các giá trị cần thiết và đưa vào model SVM đã thực hiện công hiện predict đó là chữ số nào. Cuối

cùng chính là ghép cả chuỗi số lại rồi hiển thị ra màn hình giá trị của biển số.

3. Mô tả dữ liệu:

a) Pre-trained model của mạng WPOD-Net :

Theo như bài báo nghiên cứu khoa học của tác giả Sérgio Montazzolli và Claudio Rosito Jung được chia sẻ tại¹ thì pre-trained model có được sau khi tác giả train từ 3 bộ dataset sau: **AOLP, SSIG và Cars**

- **AOLP**²: Đây là bộ dataset được cung cấp công khai tới cộng đồng bởi Phòng thí nghiệm Thị giác Nhân tạo NTUST. Trong AOLP được chia ra thành 3 nhóm như sau:

1. Ảnh được thu thập tại các vị trí kiểm soát xe ra vào (Access Control):

Kiểm soát truy cập đề cập đến các trường hợp xe đi qua một lối đi cố định ở tốc độ giảm hoặc dừng hoàn toàn, chẳng hạn như tại trạm thu phí hoặc lối vào / ra của một khu vực. Trong các kịch bản kiểm soát truy cập, máy ảnh thường được đặt cách biển số xe dưới 5 mét, trong phạm vi từ -30 đến +30 độ và nghiêng 0 đến 60 độ (vì độ nghiêng 0 độ song song với mặt đất). Trong ảnh, chiều rộng của biển số xe nằm trong khoảng từ 0,2 đến 0,25 chiều rộng của hình ảnh chụp (được hiển thị dưới dạng tỷ lệ chiều rộng trong Thông số ứng dụng) và hướng của nó nhỏ hơn 10 độ (Lưu ý rằng cả hai đều được đo trong ảnh bằng biển số được chiếu lên mặt phẳng hình ảnh). Các tham số trong Thông số ứng dụng được khái quát từ 681 hình ảnh được thu thập tại các cảnh kiểm soát truy cập khác nhau. Ánh sáng bao gồm trong nhà, ngoài trời, ban ngày, ban đêm và các điều kiện

¹

http://openaccess.thecvf.com/content_ECCV_2018/html/Sergio_Silva_License_Plate_Detection_ECCV_2018_paper.html

² <http://aolpr.ntust.edu.tw/lab/>

thời tiết khác nhau. Nếu được đo bằng cường độ trung bình trên một tấm, nó thay đổi từ 60 đến 130 độ theo thang màu xám 8 bit ³.



2. Ảnh được thu thập khi các camera trên đường phát hiện thấy xe vi phạm luật lệ giao thông như vượt quá tốc độ, vượt đèn đỏ (Traffic Law Enforcement). Có tất cả 757 ảnh được thu thập ở nhóm tình huống này.



3. Ảnh được chụp bởi cảnh sát khi tuần tra trên đường (Road Patrol). Họ chụp ảnh khi phát hiện đồ xe sai quy định, tìm kiếm xe bị mất, kiểm tra an ninh,... Bộ ảnh này được chụp ở góc nhìn và khoảng cách tùy ý. Có tất cả 611 ảnh được thu thập ở tình huống này.

³ http://www.ssig.dcc.ufmg.br/wp-content/uploads/2017/03/SSIG-SegPlate_Database_License_Agreement.pdf



Tóm tắt đặc tính của 3 nhóm tình huống chụp ảnh trên:

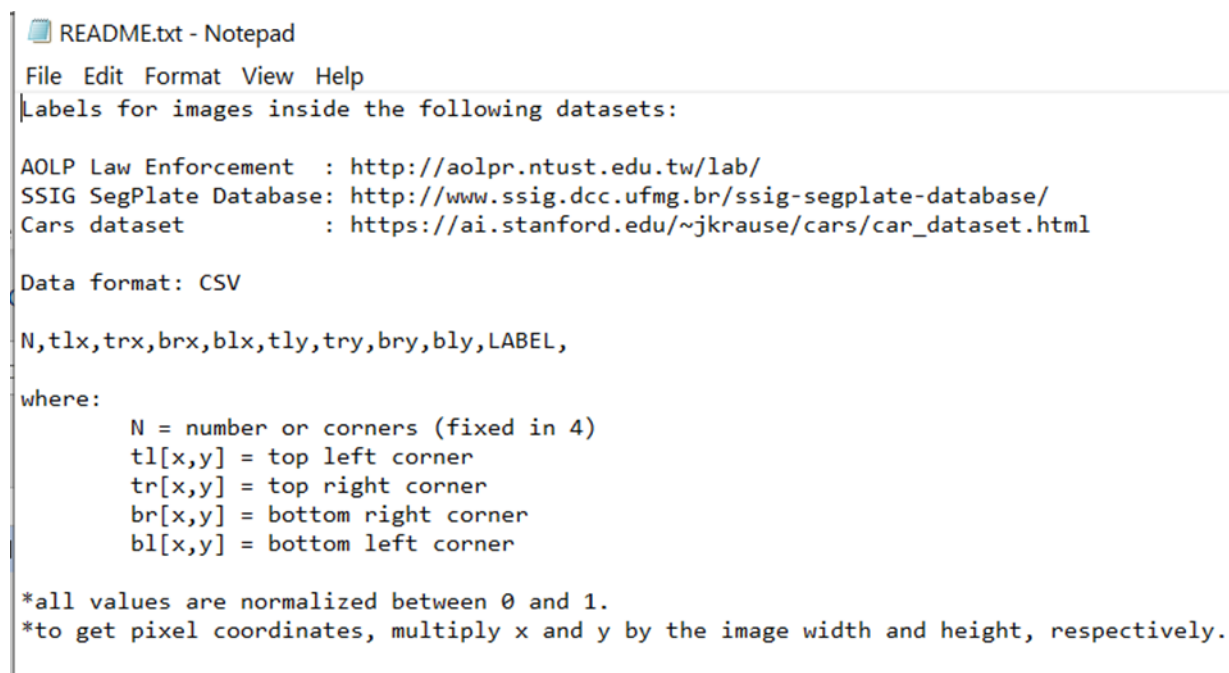
	Access Control	Traffic Law Enforcement	Road Patrol
Pan	-30°~30°	-40°~40°	-60°~60°
Tile	0°~60°	20°~70°	0°~50°
Width Ratio	0.20~0.25	0.10~0.20	0.10~0.40
Avg. Intensity	60~130	40~150	40~150
Distance	< 5m	< 15m	< 15m
Proj. Ori.	< 10°	< 15°	< 30°
No. Samples	681	757	611

- **SSIG⁴:** Đây là bộ dataset không được công khai tới cộng đồng mà ở dạng “under request”, nghĩa là cá nhân hay tổ chức để sử dụng bộ dataset này thì phải xin phép tác giả.
- **Cars⁵:** Đây là bộ dataset chụp các loại xe hơi ở tất cả góc nhìn, dù góc nhìn đó có biển số hay không có biển số xe. Bộ dataset này được cung cấp tới cộng đồng bởi đại học Stanford gồm tất cả 16185 ảnh và chia ra thành 196 loại xe hơi theo các tiêu chí như nơi sản xuất, năm sản xuất, hãng cung cấp như BMW, Tesla,... Bộ dataset này chủ yếu được tác giả sử dụng ở bước đánh giá mô hình chứ không ở bước training.

⁴ http://www.ssig.dcc.ufmg.br/wp-content/uploads/2017/03/SSIG-SegPlate_Database_License_Agreement.pdf

⁵ https://ai.stanford.edu/~jkrause/cars/car_dataset.html

Toàn bộ quá trình label và training được tác giả Sérgio Montazzolli và Claudio Rosito Jung chia sẻ tại⁶. Dưới đây là mô tả của tác giả về cách 3 bộ dataset trên được label, được tiền xử lý dữ liệu (chuẩn hóa về 0 và 1) và sau đó được huấn luyện bằng framework Keras, mô tả này có thể xem ở⁶:



```
README.txt - Notepad
File Edit Format View Help
Labels for images inside the following datasets:

AOLP Law Enforcement : http://aolpr.ntust.edu.tw/lab/
SSIG SegPlate Database: http://www.ssig.dcc.ufmg.br/ssig-segplate-database/
Cars dataset          : https://ai.stanford.edu/~jkrause/cars/car_dataset.html

Data format: CSV

N,tlx,trx,brx,blx,tly,try,bry,blly,LABEL,

where:
    N = number of corners (fixed in 4)
    tl[x,y] = top left corner
    tr[x,y] = top right corner
    br[x,y] = bottom right corner
    bl[x,y] = bottom left corner

*all values are normalized between 0 and 1.
*to get pixel coordinates, multiply x and y by the image width and height, respectively.
```

Nhóm sử dụng file pre-trained model dạng file .json được chia sẻ từ 1 bài báo trên medium⁷. Tác giả của bài báo là bạn Quang Nguyen⁸ đã training tạo ra file .json sử dụng kết quả nghiên cứu của tác giả Sérgio Montazzolli và Claudio Rosito Jung.

Đánh giá độ chính xác của mô hình pre-trained do tác giả Sérgio Montazzolli và Claudio Rosito Jung đăng trong bài báo của họ:

⁶ <http://www.inf.ufrgs.br/~smsilva/alpr-unconstrained/>

⁷ <https://medium.com/@quangnhatnguyenle/detect-and-recognize-vehicles-license-plate-with-machine-learning-and-python-part-1-detection-795fda47e922>

⁸ <https://github.com/quangnhat185>

Table 2: Full ALPR results for all 5 datasets.

	OpenALPR		SSIG	AOLP	Proposed	Average
	EU	BR	Test	RP	CD-HARD	
Ours	93.52%	91.23%	88.56%	98.36%	75.00%	89.33%
Ours (no artf.)	92.59%	88.60%	84.58%	93.29%	73.08%	86.43%
Ours (unrect.)	94.44%	90.35%	87.81%	84.61%	57.69%	82.98%
<i>Commercial systems</i>						
OpenALPR	96.30%	85.96%	87.44%	69.72%*	67.31%	81.35%
Sighthound	83.33%	94.73%	81.46%	83.47%	45.19%	77.64%
Amazon Rekog.	69.44%	83.33%	31.21%	68.25%	30.77%	56.60%
<i>Literature</i>						
Laroca et al. [17]	-	-	85.45%	-	-	-
Li et al. [18]	-	-	-	88.38%	-	-
Li et al. [19]	-	-	-	83.63%	-	-
Hsu et al. [10]	-	-	-	85.70%**	-	-

*OpenALPR struggled to understand the “Q” letter in Taiwanese LPs.

**In [10] the authors provided an estimative, and not the real evaluation.

b) Dataset của bộ kí tự chữ và số trên biển báo:

Nhóm sử dụng tập dataset chứa các kí tự số từ 0-9 và chữ cái từ A-Z ở link được chia sẻ sau:

https://www.mediafire.com/file/3l3x7bd7rq9l15r/data_digit.zip/file

Nhóm có email hỏi xin phép người đăng bộ dataset này và email hỏi 1 số nguồn bộ dataset bộ kí tự khác dùng cho biển số xe được chia sẻ trên mạng, thì đều nhận được câu trả lời là họ cũng tìm kiếm trên Google và tải về mà không rõ ai là người thu thập các bộ dataset này. Do đó nhóm không thể biết được họ đã thu thập và tiền xử lí bộ dataset này như thế nào.

Tuy nhiên nhìn vào từng tấm ảnh trong bộ dataset, nhóm nhận thấy họ crop các vùng kí tự, resize về 1 kích thước và áp dụng 1 số thuật toán xử lí ảnh cơ bản để chuyển ảnh về dạng binary. Mọi ảnh trong bộ dataset này đều ở dạng ảnh binary.

Kích thước dữ liệu:

- Số 0: 69 ảnh
- Số 1: 58 ảnh
- Số 2: 67 ảnh
- Số 3: 20 ảnh

- Số 4: 9 ảnh
 - Số 5: 69 ảnh
 - Số 6: 51 ảnh
 - Số 7: 22 ảnh
 - Số 8: 40 ảnh
 - Số 9: 41 ảnh
 - Chữ A: 18 ảnh
 - Chữ B: 54 ảnh
 - Chữ C: 36 ảnh
 - Chữ D: 52 ảnh
 - Chữ E: 84 ảnh
 - Chữ F: 12 ảnh
 - Chữ G: 21 ảnh
 - Chữ H: 60 ảnh
 - Chữ K: 14 ảnh
 - Chữ L: 18 ảnh
 - Chữ M: 15 ảnh
 - Chữ N: 25 ảnh
 - Chữ P: 60 ảnh
 - Chữ R: 30 ảnh
 - Chữ S: 14 ảnh
 - Chữ T: 5 ảnh
 - Chữ U: 34 ảnh
 - Chữ V: 11 ảnh
 - Chữ X: 32 ảnh
 - Chữ Y: 22 ảnh
 - Chữ Z: 24 ảnh
- Tổng cộng có: 1087 ảnh**

Nhóm sử dụng thư viện SVM của OpenCV nên không có hàm tính độ loss và accuracy. Do đó nhóm đếm thủ công số lượng ảnh trong tập test được dự đoán đúng bằng mô hình được huấn luyện ở tập train, từ đó suy ra tỉ lệ % độ chính xác. **Mô hình đạt độ chính xác 96,66 %** với 70% bộ dataset dùng để train, 30% dùng để test. Lưu ý là tỉ lệ 70% là tính trên từng tập ảnh, ví dụ như 70% của tập ảnh số 0, 70% của tập ảnh số 1, 70% của tập ảnh kí tự A,... Chi tiết về cách tính % độ chính xác được nhóm trình bày trong file Google Colab của nhóm.

4. Kết quả đạt được:

Hầu hết các trường hợp nằm trong phạm vi được huấn luyện của 2 mô hình máy học (WPOD-Net và bộ kí tự huấn luyện bằng SVM) cho kết quả nhận diện chính xác khi góc chụp không quá nghiêng, không quá mờ, điều kiện môi trường rõ ràng. Hai ảnh chụp dưới đây là khi nhóm in số xe trực tiếp lên ảnh, **hiện tại file Google Colab của nhóm đã sửa lại mã nguồn** và in ra số xe ở dạng console:



5. Khó khăn:

Một số ảnh nhận diện sai. Lí do có thể từ nhiều yếu tố như:

- Ánh sáng từ môi trường có thể ảnh hưởng đến sự đồng nhất của tất cả các ảnh, dẫn đến việc tiền xử lý bằng OpenCV có thể không áp dụng được ở 1 số ảnh và không tạo ra ảnh binary rõ nét để áp dụng ở bước Image Classification.

- Ảnh chụp mặt trước hoặc sau xe với góc quá nghiêng có thể dẫn đến kí tự trên biển báo không rõ nét, khó nhận diện ở cả bước tiền xử lý và máy học.

6. Hướng phát triển:

- Xây dựng thêm ảnh cho bộ dataset để tăng độ chính xác như là chụp thêm nhiều ảnh kí tự chữ, số ở góc nghiêng, cực nghiêng,...
- Thu thập thêm ảnh chụp mặt trước hoặc sau của phương tiện giao thông với các đặc tính đa dạng hơn về màu sắc, kích thước biển số,... để đánh giá được chính xác hơn độ chính xác của mô hình.
- Nghiên cứu cải tiến tối ưu hơn ở bước tiền xử lý dữ liệu.

7. Tư liệu tham khảo:

Nhóm tham khảo mã nguồn tại:

- WPOD-Net: <https://medium.com/@quangnhatnguyenle/detect-and-recognize-vehicles-license-plate-with-machine-learning-and-python-part-1-detection-795fda47e922>
- SVM: https://docs.opencv.org/4.4.0/d1/d73/tutorial_introduction_to_svm.html

8. File Google Colab thực thi chương trình của nhóm:

- https://colab.research.google.com/drive/1x9u_-XkoqkIS9ZA9u4_tsbBaCJFego2?usp=sharing