HỌC VIỆN NGÂN HÀNG

KHOA CÔNG NGHỆ THÔNG TIN VÀ KINH TẾ SỐ

TIỂU LUẬN NHÓM

Học Phần: Trí Tuệ Nhân Tạo

***ĐỀ TÀI*: ỨNG DỤNG CỦA TRÍ TUỆ NHÂN TẠO TRONG BÀI TOÁN NHẬN DIỆN BIỂN SỐ XE TẠI HỌC VIỆN**

Giáo viên hướng dẫn: TS. Vũ Trọng Sinh

Lớp học phần: 232IS54A01 – Nhóm 16

Danh sách thành viên:

Họ và tên	**Mã sinh viên**
1. Phạm Bảo Anh	24A4042424
2. Hà Gia Bảo	24A4042425
3. Nguyễn Đức Công	24A4042427
1. Ngô Văn Minh	24A4042598
2. Nguyễn Cảnh Phong	24A4042605

Hà Nội, 5/2024

HỌC VIỆN NGÂN HÀNG

KHOA CÔNG NGHỆ THÔNG TIN VÀ KINH TẾ SỐ

TIỂU LUẬN NHÓM

Học Phần: Trí Tuệ Nhân Tạo

ĐỀ TÀI: ỨNG DỤNG CỦA TRÍ TUỆ NHÂN TẠO TRONG BÀI TOÁN NHẬN DIỆN BIỂN SỐ XE TẠI HỌC VIỆN

Giáo viên hướng dẫn: TS. Vũ Trọng Sinh

Lớp học phần: 232IS54A01 – Nhóm 16

Danh sách thành viên:

Họ và tên	**Mã sinh viên**
1. Phạm Bảo Anh	24A4042424
2. Hà Gia Bảo	24A4042425
3. Nguyễn Đức Công (Leader)	24A4042427
3. Ngô Văn Minh	24A4042598
4. Nguyễn Cảnh Phong	24A4042605

Hà Nội, 5/2024

BẢNG PHÂN CÔNG CÔNG VIỆC

STT	Mã SV	Họ và tên	Phân công	Đánh giá
1	24A4042424	Phạm Bảo Anh	• Tìm ảnh • Định hướng xây dựng bài toán	18.5%
2	24A4042425	Hà Gia Bảo	• Phân tách biển số • Phân mảnh ký tự	21%
3	24A4042427	Nguyễn Đức Công (Leader)	• Phân tách biển số • Phân mảnh ký tự • Làm báo cáo	21%
4	24A4042598	Ngô Văn Minh	• Huấn luyện mô hình CNN • Phân mảnh ký tự	21%
5	24A4042605	Nguyễn Cảnh Phong	• Tìm ảnh • Test mô hình	18.5%

LỜI CẢM ƠN

Trong suốt thời gian từ khi bắt đầu thực hiện bài thực tập đến nay, chúng em đã nhận được rất nhiều sự quan tâm, giúp đỡ của Khoa và quý thầy cô. Với lòng biết ơn sâu sắc, chúng em xin gửi lời cảm ơn đầu tiên đến quý thầy cô của Khoa Công Nghệ Thông Tin Và Kinh Tế Số – Học viện Ngân Hàng đã tạo điều kiện cho chúng em được học tập bộ môn "Trí tuệ nhân tạo" để chúng em có thể củng cố và áp dụng kiến thức đã được học của môn vào việc tìm hiểu và phân tích các bài toán nghiệp vụ trong thực tế của các cơ quan, doanh nghiệp.

Chúng em cũng xin đặc biệt gửi lời cảm ơn đến thầy Vũ Trọng Sinh đã tận tâm hướng dẫn, giúp đỡ chúng em trong suốt quá trình làm bài, từ những ngày đầu tiên cho đến khi hoàn thành bài báo cáo. Nếu không có sự chỉ bảo của thầy thì bài thu hoạch này của chúng em rất khó có thể hoàn thiện được. Trong quá trình phân tích bài toán, do kiến thức, lý luận cũng như kinh nghiệm thực tiễn của chúng em còn hạn chế nên bài báo cáo của nhóm sẽ không thể tránh khỏi những thiếu sót. Chúng em rất mong nhận được góp ý quý báu của quý thầy, cô để bài báo cáo của em được hoàn thiện. Từ đó chúng em có thể rút ra được nhiều kinh nghiệm để hoàn thành tốt hơn các bài báo cáo sau này.

Sau cùng, chúng em xin kính chúc quý thầy, cô trong Khoa Công Nghệ Thông Tin Và Kinh Tế Số thật dồi dào sức khỏe, niềm tin để tiếp tục thực hiện sứ mệnh trồng người.

Chúng em xin chân thành cảm ơn!

LỜI CAM ĐOAN

Tôi xin cam đoan kết quả đạt được trong báo cáo là sản phẩm nghiên cứu, tìm hiểu của riêng nhóm chúng tôi. Trong toàn bộ nội dung của báo cáo, những điều được trình bày hoặc là của nhóm chúng tôi hoặc là được tổng hợp từ nhiều nguồn tài liệu. Tất cả các tài liệu tham khảo đều có xuất xứ rõ ràng và được trích dẫn hợp pháp.

Tôi xin hoàn chịu trách nhiệm và chịu mọi hình thức kỷ luật theo quy định cho lời cam đoan của mình.

NHÓM SINH VIÊN THỰC HIỆN

Phạm Bảo Anh
Hà Gia Bảo
Nguyễn Đức Công
Ngô Văn Minh
Nguyễn Cảnh Phong

MỤC LỤC

BẢNG PHÂN CÔNG CÔNG VIỆC i
LỜI CẢM ƠN ii
LỜI CAM ĐOAN iii
MỤC LỤC iv
DANH MỤC HÌNH ẢNH 1
LỜI MỞ ĐẦU 3
CHƯƠNG 1: GIỚI THIỆU BÀI TOÁN 4
1.1. GIỚI THIỆU BÀI TOÁN 4
1.2. HƯỚNG GIẢI QUYẾT BÀI TOÁN 4
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 6
2.1. CƠ BẢN VỀ THỊ GIÁC MÁY TÍNH 6
2.2. ÁP DỤNG VÀO BÀI TOÁN THỰC TẾ 7
2.2.1. Thư viện sử dụng 7
2.2.2. Các kỹ thuật cần sử dụng 8
CHƯƠNG 3: TRIỂN KHAI BÀI TOÁN 14
3.1. Giai đoạn 1 – nhận diện hình ảnh biển số xe từ ảnh tạp 14
3.1.1. Đọc ảnh và xử lý ảnh 14
3.1.2. Khoanh vùng biển số - phát hiện và phân tích contour 20
3.1.3. Phân tách ký tự từ biển số xe 30
3.2. Giai đoạn 2 – triển khai mô hình nhận diện ký tự 33
3.2.1. Unzip data, import thư viện và đọc data từ folder thư mục 33
3.2.2. Khởi tạo mô hình mạng neural tích chập 34
3.2.3. Huấn luyện mô hình 36
3.2.4. Vẽ biểu đồ đánh giá độ chính xác và hàm mất mát 38
3.3. Kết hợp 2 giai đoạn 40
KẾT LUẬN 43
TÀI LIỆU THAM KHẢO 44

DANH MỤC HÌNH ẢNH

Hình 1: Ảnh màu và ảnh xám 9
Hình 2: Ảnh sau quá trình tiền xử lý 9
Hình 3: Ảnh bị nhiễu Gaus 10
Hình 4: Khoanh một vùng duy nhất trong ảnh 11
Hình 5: Ảnh gốc bị lệch cần phải xử lý 12
Hình 6: Ảnh sau xử lý xoay và nhị phân hóa 12
Hình 7: Danh sách cách hình ảnh sau khi phân mảnh 12
Hình 8: Import thư viện 14
Hình 9: Đọc hình ảnh và xám hóa hình ảnh 14
Hình 10: Ảnh gốc chụp cận, độ sáng không cao 15
Hình 11: Ảnh đặt yêu cầu ngay sau khi nhị phân hóa 15
Hình 12: Hình ảnh ở xa và có nhiều góc cạnh 16
Hình 13: Hàm giảm độ sáng thông qua hàm clip của thư viện numpy 16
Hình 14: Xám hóa và giảm độ sáng 16
Hình 15: Hàm giảm nhiễu, tăng độ tương phản và loại bỏ nhiễu 17
Hình 16: Vị trí biển số xe đã rõ hơn 17
Hình 17: Hàm nhị phân hóa ảnh 17
Hình 18: Hình ảnh sau nhị phân hóa 18
Hình 19: Không khoang vùng được do nhiều cạnh rác 18
Hình 20: Hàm phát hiện cạnh Canny 19
Hình 21: Hàm phát hiện biên bằng Dilation kết hợp phân tích hình thái học 19
Hình 22: Ảnh sau phân tích cạnh và phát hiện biên 19
Hình 23: Hàm Sobel hóa hình ảnh 20
Hình 24: Hình ảnh sau Sobel hóa 20

Hình 25: Gọi hàm tìm Contours, xắp xếp các contour theo thứ tự 24
Hình 26: Vẽ Approx lên hai hình ảnh đã tạo.....25
Hình 27: Contours vẽ vào đúng các đường viền 25
Hình 28: Code phân tích để lọc ra contour biển số 26
Hình 29: Sau xử lý thì chỉ còn duy nhất một contours tại đúng vị trí của biển số xe ...27
Hình 30: Hàm tìm góc nghiêng của biển số xe 28
Hình 31: Code cắt biển số xe và lưu ra file 28
Hình 32: Ảnh gốc 29
Hình 33: Ảnh sau khi bóc tách biển số.....30
Hình 34: Hình ảnh sau quá trình tiền xử lý 31
Hình 35: Tạo Kernel hình chữ nhật và tìm contours 31
Hình 36: Phân tích contours 31
Hình 37: Vẽ hình chữ nhật bao quanh ký tự và cắt ký tự.....32
Hình 38: Khoanh vùng các ký tự.....32
Hình 39: Import các thư viện cần dùng 33
Hình 40: Đoạn code dọc dữ liệu từ folder train và test 34
Hình 41: Khởi tạo mô hình mạng neural tích chập CNN.....34
Hình 42: Mô hình tổng quan 36
Hình 43: Epoch cao nhất đạt được 38
Hình 44: Hàm vẽ biểu đồ đánh giá mô hình huấn luyện.....38
Hình 45: Kết quả huấn luyện mô hình.....39
Hình 46: Import thư viện và load model, lấy mảng ký tự đã tách từ giai đoạn trên 40
Hình 47: Hàm sử dụng để chuyển đổi ảnh và hiển thị kết quả dự đoán.....41
Hình 48: Kết quả của mô hình.....42

LỜI MỞ ĐẦU

Trí tuệ nhân tạo hay còn được biết đến với cái tên AI là một cụm từ nóng như lò than trong những năm gần đây. Sau đợt dịch Covid năm 2019 – 2020 - 2021, thì lượng thông tin được đưa lên mạng tăng cực kỳ mạnh trong thời điểm này. Theo dữ liệu từ ITU vào năm 2019, có khoảng 4.1 tỷ người đã sử dụng Internet (54 %) dân số thế giới. Lượng thông tin này chưa hề có dấu hiệu chững lại mà tiếp tục tăng chóng mặt. Vào năm 2021 đã tăng lên thành 4.9 tỷ người (63%) dân số thế giới. Chính vì những lượng thông tin khổng lồ như thế làm cho môi trường phát triển của lĩnh vực khai phá và phân tích dữ liệu trở nên giàu có và tất nhiên cũng kéo theo trí tuệ nhân tạo và học máy phát triển. Nguồn nhân lực về lĩnh vực AI hiện vẫn còn chưa nhiều nên có thể đây sẽ là một môi trường tuyệt với để các bạn hướng đến sau này.

Bài toán của nhóm chúng mình đề cập đến bài toán gửi xe tại học viện. Cụ thể là sử dụng thị giác máy tính vào bài toán nhận diện biển số xe khi xe gửi và lấy xe theo từng lần gửi bằng mã sinh viên. Hi vọng quý độc giả có thể theo dõi bài báo cáo của nhóm mình!

CHƯƠNG 1: GIỚI THIỆU BÀI TOÁN

1.1. GIỚI THIỆU BÀI TOÁN

Tại học viện hiện nay phương thức gửi xe đã khá là ổn, thế nhưng vẫn có một số điểm chưa tối ưu đó là những điều dưới đây:

- Thứ nhất:Cần nhân lực tại các điểm chốt gửi xe: Điều này làm tiêu tốn một lượng nhân lực không nhỏ, làm tiêu tốn tài nguyên của học viện. Chưa kể, khi nhân lực là con người nhiều khi nhân viên của học viện cáu gắt thì cũng làm cho người gửi xe bực mình. Các bạn thử nghĩ mà xem, sinh viên là người sử dụng dịch vụ của học viện vậy mà nhân viên của dịch vụ đó lại cáu gắt với bạn thì liệu bạn có hài lòng? Nếu không phải vì gửi xe khi đi học thì liệu các bạn có muốn gửi xe ở trường?
- Thứ hai: Camera ở khu vực ra vào chốt chưa tận dụng được nguồn lực của chúng: Dường như Camera tại những khu vực này chỉ có tác dụng ghi lại, vậy tại sao chúng ta không tận dụng chúng để làm nhiều nhiệm vụ cùng lúc? Ví dụ như dùng nó để ghi lại hình ảnh biển số cung cấp cho các quá trình ở phía dưới.
- Thứ ba: Chức năng của thẻ sinh viên và thẻ gửi xe chưa được tối ưu: Cụ thể là trong khi thẻ sinh viên được tích hợp tính năng quẹt thẻ thì tại sao chúng ta không tận dụng nó để thanh toán phí gửi xe mỗi lượt? Đối với thẻ gửi xe, dường như nó chả có tác dụng gì ngoài việc "Đăng ký gửi xe" vậy mà lại cung cấp cho nó chức năng quét để xác nhận xe ra, vào thật là lãng phí công nghệ vào một công việc như thế. Nếu chúng ta có thể kết hợp 2 loại thẻ này vào làm một thì mới đúng là tận dụng công nghệ.

1.2. HƯỚNG GIẢI QUYẾT BÀI TOÁN

Để giải quyết các điểm tiêu cực mà phần trên mình đã nêu ra thì nhóm mình có giải pháp như sau:

BOT gửi xe sẽ sử dụng cơ chế tương tự như các trạm BOT thu phí tự động. Nhưng thay vì, dùng mã QR code để quét tài khoản và trừ phí thì chúng ta sẽ sử dụng thẻ sinh viên để quét mỗi khi vào gửi xe.

- Khi đưa xe vào gửi thì sinh viên sẽ tự quét thẻ của mình, sau đó barrier sẽ mở ra và để sinh viên vào. Trong khi sinh viên quét thẻ của mình vào bộ quét thì bộ quét sẽ đọc thẻ và Cam sẽ quét biển số xe rồi đưa vào AI để nhận

diện biển số xe (AI thị giác máy tính mà mình sẽ làm trong bài này).

- Sau khi có được đầu ra là biển số xe thì hệ thống gửi xe của nhà trường sẽ tiến hành đăng ký (match) hai dòng thông tin là mã sinh viên và biển số xe lại với nhau, điều này sẽ đảm bảo lúc đi bạn đi xe nào thì lúc về sẽ lấy xe đó chứ không có chuyện lấy xe khác và dùng thẻ sinh viên khác cũng không được.
- Khi lấy xe ra thì sinh viên sẽ quét thẻ vào bộ quét. Bộ quét sẽ lấy mã sinh viên và Cam sẽ quét biển số xe 1 lần nữa rồi trả về kết quả, nếu hợp lệ thì sẽ trừ phí trong thẻ của sinh viên và xóa thông tin đăng ký trong hệ thống gửi xe và mở Barrier.

Đó là cách giải quyết của nhóm chúng mình, tuy là chưa được chỉn chu nhưng về cơ bản đã tận dụng được nguồn lực của các thiết bị, hệ thống hiện có trong nhà trường. Đồng thời cắt giảm được lực lượng nhân lực để điều hướng cho các nghiệp vụ khác ví dụ sắp xếp chỗ để xe cho sinh viên ,…

Trong bài báo cáo này, nhóm mình sẽ thực hiện cái cốt lõi của bài toán và phương hướng giải quyết của nó, đó chính là “Mô hình AI – thị giác máy tính – đọc biển số xe”. Đầu vào của mô hình chính là hình ảnh chụp có chứa biển số xe và đầu ra là chuỗi ký tự của biển số xe đã được sắp xếp.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

2.1. CƠ BẢN VỀ THỊ GIÁC MÁY TÍNH

Các bước thực hiện phổ biến trong các bài toán thị giác máy tính thì gồm có các bước sau:

- Thu thập dữ liệu
  - Mô tả: Xác định nguồn dữ liệu (có thể là từ internet, cơ sở dữ liệu hiện có, camera, etc.)
  - Đảm bảo dữ liệu đủ đa dạng để mô hình có thể học tốt.
  - Ví dụ: Ảnh chụp các biển báo giao thông, ảnh khuôn mặt người, ảnh chụp động vật.
- Tiền xử lý dữ liệu
  - Mô tả: Xử lý và làm sạch dữ liệu để chuẩn bị cho việc huấn luyện mô hình.
  - Bao gồm các công việc như cắt ảnh, thay đổi kích thước, chuyển đổi ảnh màu sang xám, chuẩn hóa, v.v.
  - Ví dụ: Chuyển đổi ảnh màu sang ảnh đen trắng.
  - Chuẩn hóa kích thước ảnh về cùng một kích thước cố định (ví dụ: 224x224 pixels).
- Gán nhãn dữ liệu
  - Mô tả: Gán nhãn cho dữ liệu để mô hình có thể học từ đó. Nếu sử dụng các thuật toán học có giám sát, dữ liệu phải có nhãn.
  - Ví dụ: Ảnh con mèo được gán nhãn là "mèo", ảnh con chó được gán nhãn là "chó".
- Chia tách dữ liệu
  - Mô tả: Chia dữ liệu thành các tập huấn luyện, kiểm tra và kiểm định để đánh giá hiệu suất của mô hình.
  - Ví dụ: 70% dữ liệu cho huấn luyện, 20% cho kiểm tra và 10% cho kiểm định.
- Xây dựng mô hình
  - Mô tả: Lựa chọn và xây dựng mô hình thị giác máy tính phù hợp. Các mô hình có thể là các thuật toán máy học truyền thống (như SVM, K-NN) hoặc các mạng neural sâu (như CNN, RNN).

- Ví dụ: Sử dụng mạng convolutional neural network (CNN) để nhận diện khuôn mặt.

❖ Huấn luyện mô hình

- Mô tả: Huấn luyện mô hình bằng cách sử dụng dữ liệu huấn luyện và tối ưu hóa các tham số của mô hình.
- Ví dụ: Sử dụng thuật toán gradient descent để tối ưu hóa các trọng số của mạng CNN.

❖ Đánh giá mô hình

- Mô tả: Đánh giá hiệu suất của mô hình trên dữ liệu kiểm tra và kiểm định. Sử dụng các chỉ số như độ chính xác (accuracy), độ nhạy (recall), độ đặc hiệu (precision), F1-score, v v.
- Ví dụ: Mô hình đạt độ chính xác 95% trên tập kiểm tra.

2.2. ÁP DỤNG VÀO BÀI TOÁN THỰC TẾ

Trong bài toán này, nhóm chúng em cũng tuân theo các bước cơ bản trên nhưng có một số thay đổi để phù hợp với bài toán. Cụ thể nhóm sẽ chia thành hai giai đoạn lớn. Mỗi giai đoạn sẽ có các quy trình nhỏ thực hiện các nhiệm vụ như phía trên đã trình bày (có thể cả quá trình hoặc một số bước). Giai đoạn 1 là chiết xuất hình ảnh biển số xe từ hình ảnh toàn cục (hình ảnh tạp, có các cảnh quan xung quanh). Giai đoạn 2 là xác định ký tự trong biển số xe. Chi tiết các giai đoạn này mình sẽ đề cập đến ở phần 3.

2.2.1. Thư viện sử dụng

2.2.1.1. Thư viện xử lý ảnh

Có rất nhiều thư viện xử lý ảnh như là OpenCV – chuyên dụng cho thị giác máy tính Scikit-image, TensorFlow và Keras. Nhưng nhóm sẽ sử dụng OpenCV để có thể giới thiệu cụ thể nhất về cách hoạt động của từng bước trong từng giai đoạn.

Ngoài ra còn có mô hình YOLO chuyên sử dụng cho nhận dạng và phát hiện đối tượng (cực kỳ phù hợp cho bài toán này, nhưng mình không muốn sử dụng nó vì nó quá mạnh mẽ, không làm nổi bật được tính lý thuyết của bài!)

2.2.1.2. Thư viện vẽ biểu đồ

Về thư viện vẽ biểu đồ thì ngay lập tức mình nghĩ đến đó chính là matplotlib vì

đáp ứng được những yêu cầu của nhóm mình như là vẽ nhiều ảnh trên 1 khung, hiển thị ngay trên cửa sổ code,…

2.2.1.3. Thư viện toán học

Các thư viện toán học thì có Numpy, Sympy, Scipy,… mỗi cái có một mục đích và thế mạnh riêng nhưng nhóm mình sẽ sử dụng Numpy và Math để tính toán toán học thuần túy và xử lý mảng,… nên sẽ không cần các thư viện chuyên sâu kia.

2.2.1.4. Thư viện học máy chuyên sâu

Để có thể huấn luyện cho máy học được các ký tự thì cần phải có một thư viện chuyên dụng. Trong hằng hà sa số các thư viện như caffe, scikit-learn, PyTorch,.. thì nhóm mình chọn sử dụng TensorFlow.Keras vì các điểm sau:

- Hiệu năng cao: Nó được tối ưu hóa trên nhiều loại phần cứng qua đó đẩy nhanh tiến độ huấn luyện.
- Cộng đồng lớn: Khi bị lỗi phần nào đó thì sẽ dễ dàng tìm được cách sửa.

2.2.1.5. Thư viện đánh giá kết quả

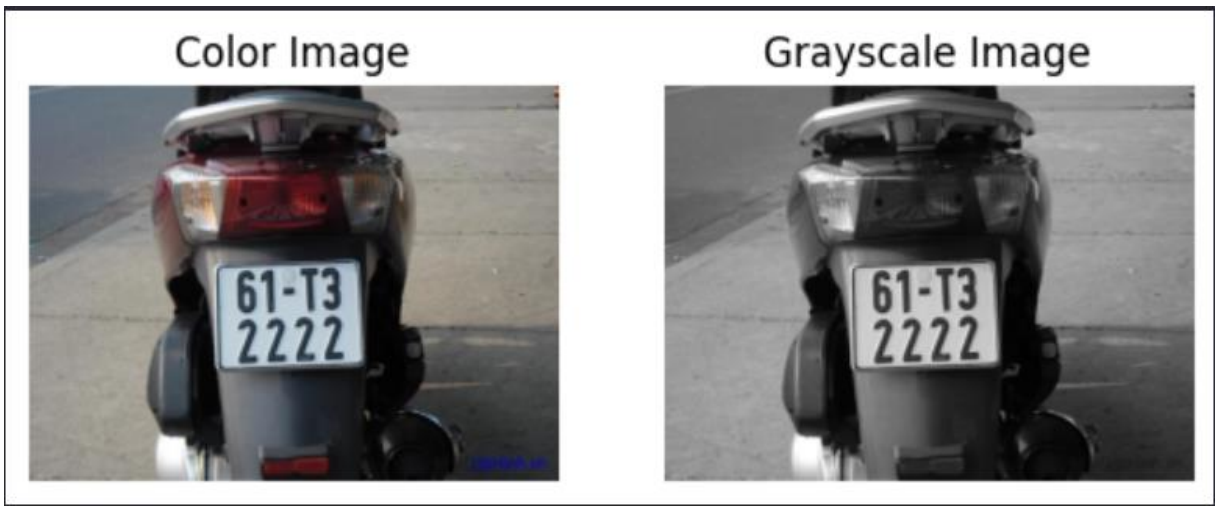
Vì đã cài sẵn TensorFlow.Keras nên nhóm sẽ dùng luôn thư viện này để đánh giá kết quả bằng lossFunction

2.2.2. Các kỹ thuật cần sử dụng

2.2.2.1. Giai đoạn 1 – tìm biển số xe và cắt ảnh biển số xe

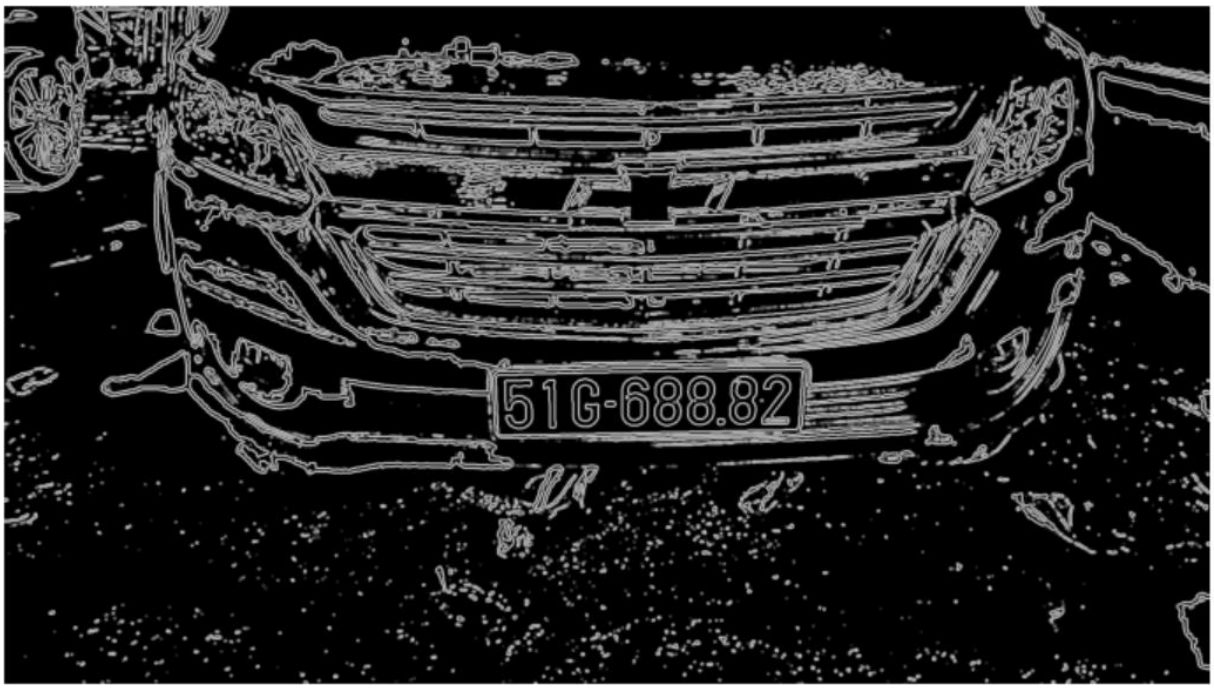
Trong giai đoạn này, các kỹ thuật quan trọng gồm có các kỹ thuật sau:

- ❖ Kỹ thuật chuyển đổi kích thước ảnh: Hiểu đơn giản là ảnh đầu vào của bạn có thể không phải lúc nào cũng bằng nhau nên chúng ta phải chuyển hết về 1 kích cỡ để các quá trình sau được đồng nhất.
- ❖ Kỹ thuật Grayscale: Hay còn gọi là kỹ thuật xám hóa hình ảnh. Đầu vào sẽ là một hình ảnh ở không gian màu RGB và sau khi xám hóa thì nó sẽ trở thành một hình ảnh ở không gian màu xám (không gian màu xám là không gian mà tại đó một pixel ảnh sẽ được biểu diễn bằng một giá trị duy nhất từ 0 – 255, 0 là đen tuyệt đối và 255 là trắng tuyệt đối).



Hình 1: Ảnh màu và ảnh xám

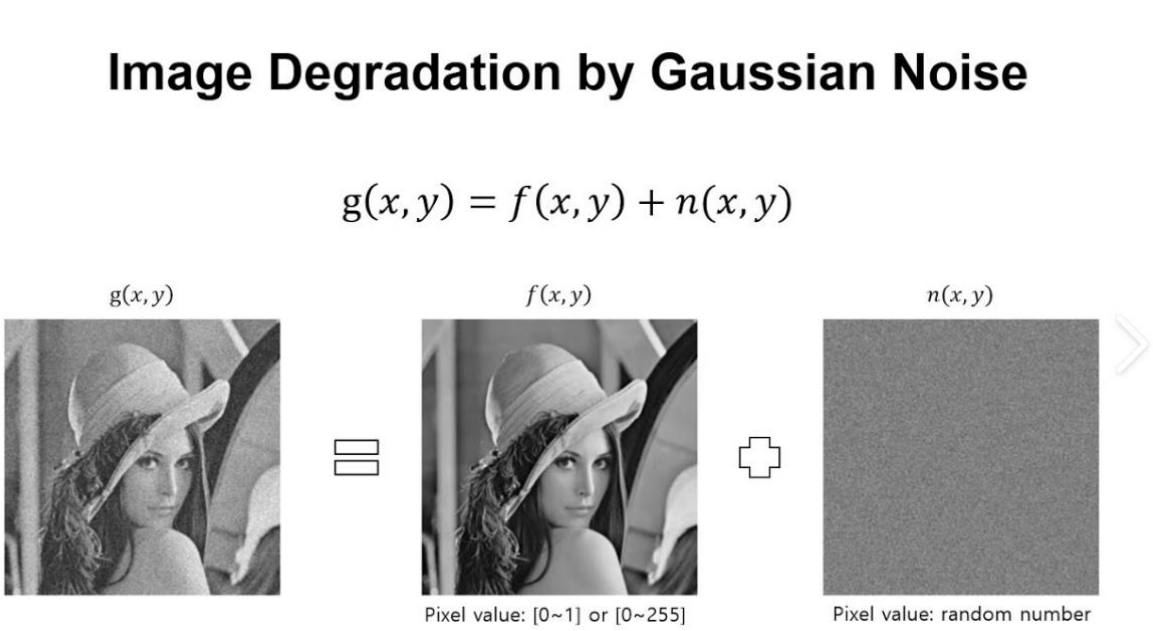
❖ Kỹ thuật tiền xử lý ảnh: Trong giai đoạn này mục tiêu của bạn là phải tạo ra hình ảnh góc cạnh nhất, nét liền nhất và đơn giản nhất, loại bỏ được các tạp cảnh xung quanh. Trong kỹ thuật này chứa rất nhiều kỹ thuật nhỏ cụ thể như sau:



Hình 2: Ảnh sau quá trình tiền xử lý

➢ Kỹ thuật làm mờ ảnh và làm mịn ảnh: Kỹ thuật này có các ứng dụng sau:

- Giảm nhiễu ảnh: khi ảnh chụp bị nhiễu. Sau khi trải qua bước này thì ảnh sẽ trở nên mịn màng hơn.



Hình 3: Ảnh bị nhiễu Gaus

- Giảm độ nét: Việc làm giảm độ nét của ảnh đóng vai trò rất quan trọng, nó sẽ loại bỏ các điểm mà ta không muốn tập trung vào. Từ đó làm cho các bước phía dưới trở nên dễ dàng hơn.
- Giảm độ lớn của ảnh: Ảnh quá nét dẫn tới dung lượng quá cao làm tiêu tốn tài nguyên của hệ thống cho nên ta phải làm mờ để giảm dung lượng xuống để tăng hiệu suất làm việc của máy.

➢ Kỹ thuật cân bằng Histogram: Là một kỹ thuật tăng độ tương phản của ảnh, hiểu nôm na là cái nào sáng sẽ sáng hơn, cái nào tối sẽ tối hơn.

➢ Kỹ thuật lọc trung vị: Là một kỹ thuật cho phép thay thế các pixel bằng giá trị trung vị của các pixel lân cận, vừa giúp loại bỏ nhiễu trong khi vẫn giữ được chi tiết của đường biên.

➢ Kỹ thuật nhị phân hóa ảnh: Là một kỹ thuật để tạo ra một ảnh nhị phân từ một ảnh xám hoặc ảnh RGB dùng để phân tách nền của ảnh khỏi đối tượng. Đây là một kỹ thuật rất quan trọng để trích xuất "vùng ảnh hữu ích" từ ảnh đã cho.

➢ Ngoài ra còn một số kỹ thuật nâng cao khác như tăng giảm độ sáng, tăng màu đen, tìm biên tại điểm có mức độ tương phản cao,…

❖ Phân tích hình thái học: Là một kỹ thuật dùng để làm sạch các chi tiết không cần thiết trong ảnh.

❖ Edge Dectection: Là một kỹ thuật dùng để phát hiện các cạnh rõ ràng trong ảnh.

- Kỹ thuật phát hiện và phân tích contour(đường viền): Đây là một trong những quá trình quan trọng nhất trong bài toán này. Vì nó sẽ quyết định xem bạn có khoanh vùng được biển số từ các cạnh xuất hiện không. Trong bước này, bạn sẽ phải vừa quan sát, vừa tìm ra những đặc điểm, tính chất của đối tượng mà muốn khoanh vùng. Sau đó tổng kết lại để khoanh được một vùng chính xác duy nhất trong bức ảnh.

Hình 4: Khoanh một vùng duy nhất trong ảnh

- Kỹ thuật cắt ảnh: Sau khi đã chọn được vùng ảnh chúng ta sẽ cần cắt ảnh này ra trên ảnh gốc và thực hiện các quá trình xử lý ảnh tương tự như như các bước phía trên nhưng đơn giản hơn. Để ra được một bức ảnh biển số xe rõ ràng nhất có thể. Nhưng không phải ảnh nào cũng thẳng hướng chính diện mà có một số ảnh sẽ ở hướng chéo. Lúc này các bạn sẽ cần thực hiện xoay lại ảnh cho cân.

Hình 5: Ảnh gốc bị lệch cần phải xử lý

Hình 6: Ảnh sau xử lý xoay và nhị phân hóa

- Kỹ thuật phân mảnh ảnh biển số xe: Kỹ thuật này sẽ khoanh vùng các ký tự có trong ảnh biển số xe sau xử lý ở kỹ thuật phía trên rồi tiến hành cắt ra thành bộ các ảnh làm nguồn cho quá trình phía dưới.

Hình 7: Danh sách cách hình ảnh sau khi phân mảnh

2.2.2.2. Giai đoạn 2 – nhận diện ký tự và đọc ký tự

a. Nhận diện ký tự

Về cơ bản phần nhận diện ký tự cũng sử dụng các kỹ thuật tương tự như khi tách biển số xe từ ảnh tạp đó là các kỹ thuật: xám hóa, resize ảnh, nhị phân hóa ảnh, phát hiện và phân tích contours,.... Ngoài ra còn có một số kỹ thuật khác như kỹ thuật cắt ảnh theo dòng, kỹ thuật đảo pixel,…

b. Đọc ký tự

Đọc ký tự cũng là một phần cực kỳ quan trọng trong giai đoạn này. Để thực hiện được phần này đầu tiên chúng ta phải tìm nguồn dữ liệu về các ký tự. Tất nhiên các bạn hoàn toàn có thể sử dụng các ảnh mà mình đã cắt ra để train nhưng mà điều này cực kỳ mất công vì các lý do sau:

- Thứ nhất: Khi cắt ảnh ra không phải chúng sẽ không có nhãn mà ta sẽ phải gán nhãn cho chúng.
- Thứ hai: Chưa chắc ảnh đầu vào của chúng ta phù hợp với tất cả các ảnh đầu vào, sẽ có cách ảnh dễ cắt, ảnh khó cắt.
- Thứ ba: Các bạn sẽ phải phân loại, copy chúng vào từng thư mục của chúng ví dụ. Các ảnh 1 sẽ phải xếp vào thư mục 1, 0 sẽ phải vào thư mục 0.

❖ Chính vì các lý do trên mà nhóm sẽ sử dụng tập data này trên kaggle tại đây.

Phần đọc ký tự sẽ nặng về kỹ thuật huấn luyện mô hình cụ thể là: huấn luyện một mạng neural tích chập CNN, kỹ thuật xây dựng lớp CNN có kiến trúc phù hợp với việc nhận diện ký tự trong ảnh, kỹ thuật biên dịch mô hình, kỹ thuật đánh giá mô hình ,… Chi tiết mình sẽ nói thêm ở phần 3 tại các đoạn code mà nhóm mình sử dụng.

CHƯƠNG 3: TRIỂN KHAI BÀI TOÁN

3.1. Giai đoạn 1 – nhận diện hình ảnh biển số xe từ ảnh tạp

Về nguồn dữ liệu đầu vào của giai đoạn này là hàng loạt các hình ảnh được nhóm chụp quanh khu vực sinh sống, trong khuôn viên trường và ở trên đường,…

3.1.1. Đọc ảnh và xử lý ảnh

3.1.1.1. Đọc ảnh và xám hóa hình ảnh

Đây là bước đầu tiên nhưng không kém phần quan trọng vì hình ảnh sau khi đọc và xám hóa thì sẽ được sử dụng liên tục cho các bước phía sau.

Đầu tiên các bạn sẽ phải Import thư viện cần sử dụng cho bài toán của chúng ta đó là các thư viện sau: matplotlib.pyplot, numpy, pandas và một thư viện do nhóm tự code là PreProcess,…

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import PreProcess as pre
```

Hình 8: Import thư viện

```
# Đọc và thay đổi kích thước hình ảnh
img = cv2.imread("image/82.jpg")
if img is None:
    raise FileNotFoundError("The image file was not found or could not be opened. Please check the file path and try again.")
img = cv2.resize(img, dsize=(1920, 1080))
# Chuyển ảnh sang grayscale để giảm độ sáng
imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Hình 9: Đọc hình ảnh và xám hóa hình ảnh

3.1.1.2. Xử lý hình ảnh

Nhiệm vụ tối thượng của giai đoạn này là tạo ra được một hình ảnh mà trong hình ảnh đó:

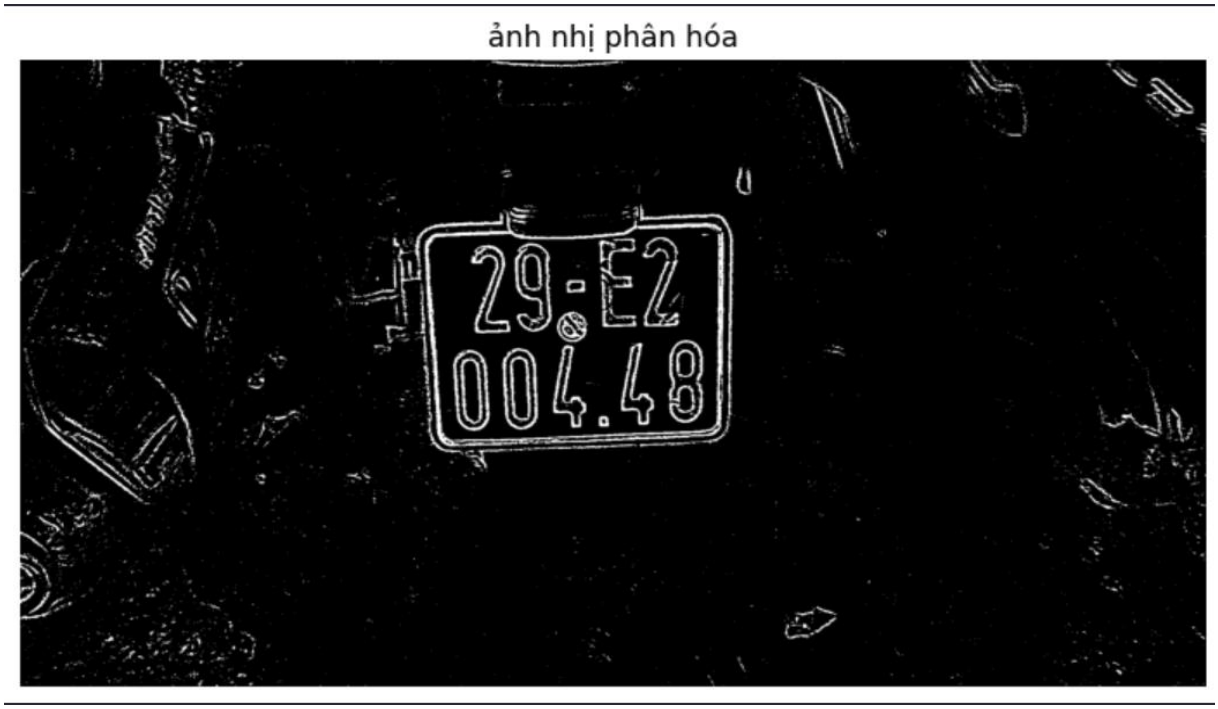
- Tập trung được tối đa vào phần biển số xe.
- Các tạp cảnh phải bị mờ đi

- Đường viền của biển số xe phải rõ, không đứt đoạn.

Sau quá trình làm việc thì nhóm rút ra được một số kinh nghiệm như sau:

❖ Ảnh chụp cận, độ sáng không cao thì dùng nhị phân hóa là sẽ được ảnh đạt yêu cầu.

Hình 10: Ảnh gốc chụp cận, độ sáng không cao



Hình 11: Ảnh đặt yêu cầu ngay sau khi nhị phân hóa

❖ Đối với những bức ảnh ở xa hoặc có nhiều góc cạnh thì cần xử lý khá nhiều:

➢ Đầu tiên ta sẽ phải giảm độ sáng của bức ảnh xuống và tăng tính tương phản của hình ảnh.

▪ Ở đây mình sẽ ví dụ với hình ảnh này

Hình 12: Hình ảnh ở xa và có nhiều góc cạnh

```
# Giảm độ sáng của ảnh
def decrease_brightness(image, value):
    return np.clip(image - value, 0, 255).astype(np.uint8)
```

Hình 13: Hàm giảm độ sáng thông qua hàm clip của thư viện numpy

Hình 14: Xám hóa và giảm độ sáng

➢ Tiếp theo là giảm nhiễu, làm mờ và tăng tính tương phản bằng cách sử dụng

GaussianBlur và MedianBlur kèm theo Equalized.

```python
#  Làm mịn và giảm nhiễu bằng GaussianBlur
def ReduceNoiceAndSmooth(imgGray):
    blurred_image = cv2.GaussianBlur(imgGray, (7, 7), 1.4)
    return blurred_image
# Tăng độ tương phản và lọc trung vị để loại bỏ nhiễu
def IncreaseContrastAndMedianFilter(imgBlurred):
    # cân bằng Histogram - tăng cường độ tương phản bằng equalize
    equalized = cv2.equalizeHist(imgBlurred)
    # lọc trung vị để loại bỏ nhiễu sử dụng MedianBlur
    denoised = cv2.medianBlur(equalized, 7)
    return denoised
```

Hình 15: Hàm giảm nhiễu, tăng độ tương phản và loại bỏ nhiễu

Hình 16: Vị trí biển số xe đã rõ hơn

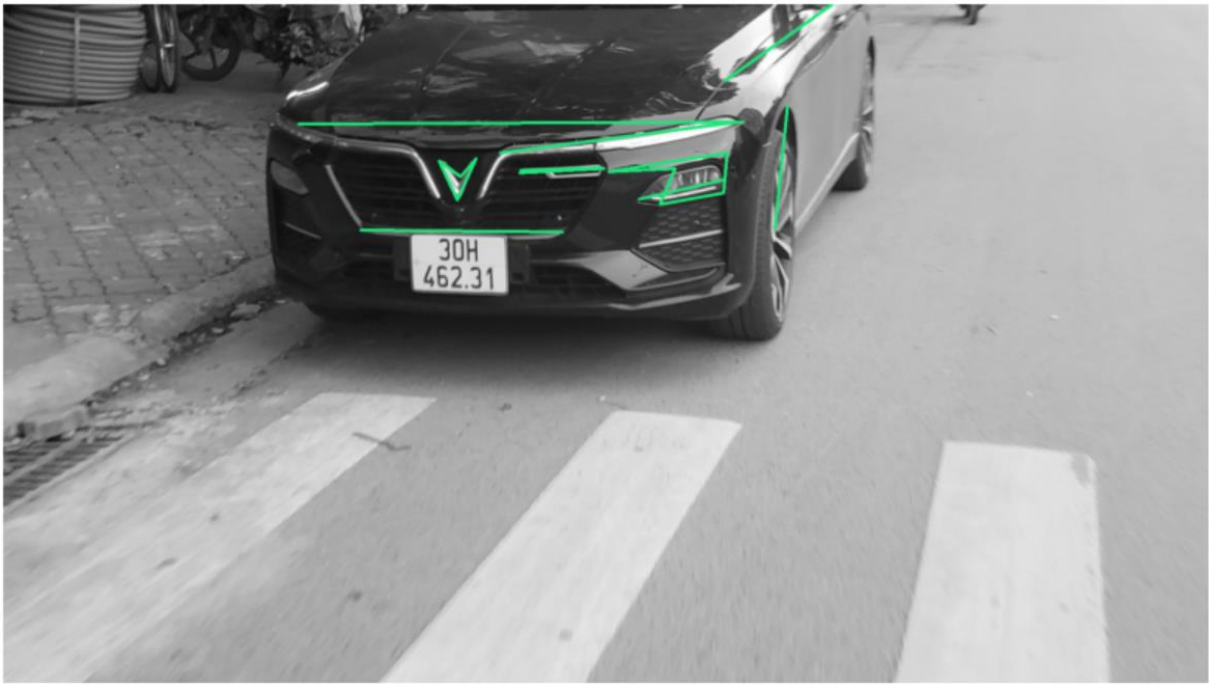
- Bước tiếp theo chúng ta sẽ nhị phân hóa bức ảnh

```python
# nhị phân hóa hình ảnh
def BinarizeImg(imgGray):
    ADAPTIVE_THRESH_BLOCK_SIZE = 19
    ADAPTIVE_THRESH_WEIGHT = 9
    imgThresh = cv2.adaptiveThreshold(imgGray, 255.0, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY_INV, ADAPTIVE
    return imgThresh
```

Hình 17: Hàm nhị phân hóa ảnh

Hình 18: Hình ảnh sau nhị phân hóa

- Ta có thể thấy sau nhị phân hóa hình ảnh đã sắc nét và có thể thấy được góc cạnh, thế nhưng nếu mang hình ảnh này đi phân tích contour thì sẽ bị lỗi ngay vì đơn giản ảnh này có quá nhiều cạnh rác. Ta cần phải loại bỏ nó và làm dày các đường biên lên bằng cách sử dụng các phương pháp sau:



Hình 19: Không khoang vùng được do nhiều cạnh rác

- Kỹ thuật phân tích cạnh: Kỹ thuật này làm cho hình ảnh trở nên rõ ràng về các chi tiết cạnh hơn. Thường thì kỹ thuật này sẽ được kết hợp với một kỹ thuật khác gọi là Kỹ thuật phân tích hình thái học Dilation để tạo ra hình ảnh có thể đem ra để tìm contour.

```
def EdgeDetectionWithCanny(GradientImg):
    canny_image = cv2.Canny(GradientImg, 250, 255)
    return canny_image
```

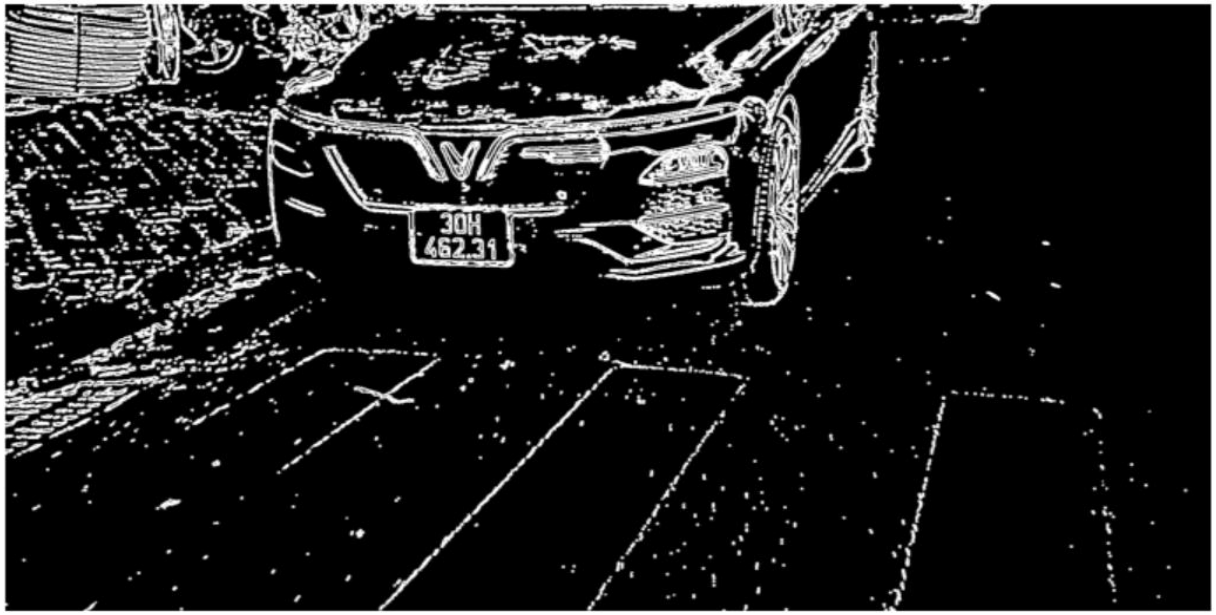
Hình 20: Hàm phát hiện cạnh Canny

```
# phát hiện biên bằng Dilation
def GradientMO(BinaryImg):
    # Tạo kernel cho các phép toán hình thái học
    kernel = np.ones((3,3), np.uint8)

    # Giãn nở và xói mòn
    dilation = cv2.dilate(BinaryImg, kernel, iterations=1)
    erosion = cv2.erode(BinaryImg, kernel, iterations=1)

    # Tính gradient hình thái học
    morph_gradient = cv2.morphologyEx(BinaryImg, cv2.MORPH_GRADIENT, kernel)
    return morph_gradient
```

Hình 21: Hàm phát hiện biên bằng Dilation kết hợp phân tích hình thái học



Hình 22: Ảnh sau phân tích cạnh và phát hiện biên

- Ngoài ra mình còn phát hiện ra một hàm có vẽ hữu dụng nhờ đặc điểm của biển số xe đó là kỹ thuật Sobel hóa hình ảnh. Đặc điểm này có được là do tại vùng biên của biển số luôn có vùng đen bao quanh mà bên trong lại mà màu trắng dẫn đến chúng tương phản rất mạnh. Chúng ta sẽ lọc biển số xe dựa trên tính chất này.

```python
# phát hiện biên bằng Sobel operator bằng cách tìm vị trí có độ tương phản diễn ra mạnh nhất
def SobelOperator(DilatedImg):
    # sử dụng Sobel operator để tìm biên mà tại đó điểm tương phản là mạnh nhất

    # Sử dụng Sobel Operator để tìm biên
    sobelx = cv2.Sobel(DilatedImg, cv2.CV_64F, 1, 0, ksize=5)
    sobely = cv2.Sobel(DilatedImg, cv2.CV_64F, 0, 1, ksize=5)

    # Tính độ lớn của gradient
    gradient_magnitude = np.sqrt(sobelx ** 2 + sobely ** 2)

    # Chuẩn hóa về phạm vi từ 0 đến 255 và chuyển đổi sang kiểu uint8
    gradient_magnitude = (gradient_magnitude / np.max(gradient_magnitude) * 255).astype(np.uint8)
    return gradient_magnitude
```

Hình 23: Hàm Sobel hóa hình ảnh

Hình 24: Hình ảnh sau Sobel hóa

3.1.2. Khoanh vùng biển số - phát hiện và phân tích contour

Vì phần này là một phần rất quan trọng nên mình sẽ nói kỹ một chút. Cụ thể là các đề mục dưới đây, kính mời các bạn độc giả đón xem.

3.1.2.1. Nguyên lý hoạt động của hàm tìm contour

a. Khái niệm

Contours là những đường cong liên tục kết nối các điểm liền kề (có cùng màu hoặc cường độ) trên biên của đối tượng trong hình ảnh. Chính vì vậy trong quá trình xử lý ảnh thì ta phải chuyển đổi sang ảnh nhị phân để máy có thể nhận diện được các đường viền này tốt hơn. Điều này đã trở thành điều tất yếu trong quá trình xác định Contours

b. Hàm findContours

Hàm findContours này gồm các tham số sau:

$$contours, hierarchy = cv2.findContours(edges, phương\ pháp\ lấy, phương\ pháp\ xấp\ xỉ)$$

Trong đó:

- Tham số đầu vào
  - Edges: ảnh nhị phân đầu vào.
  - Phương pháp lấy contour: các phương thức này sẽ xác định các mà contour được lấy và tổ chức. Mảng hierarchy sẽ hứng lấy mức độ phân cấp từ phương thức này. Có các phương thức lấy contours sau:
    - cv2.RETR_EXTERNAL: Chỉ lấy các contours ngoài cùng (outer contours). Các contours bên trong sẽ bị bỏ qua.
    - cv2.RETR_LIST: Lấy tất cả các contours mà không cần phân cấp (no hierarchical relationships). Các contours được lưu trữ dưới dạng danh sách đơn giản.
    - cv2.RETR_CCOMP: Lấy tất cả các contours và tổ chức chúng thành hai cấp độ: contours ngoài cùng (external contours) và contours bên trong (holes). Mỗi contour ngoài cùng sẽ có một contour con tương ứng.
    - cv2.RETR_TREE: Lấy tất cả các contours và tổ chức chúng thành cây phân cấp (hierarchical tree), như đã giải thích trước đó.
  - Phương thức xấp xỉ contours
    - cv2.CHAIN_APPROX_NONE: Lưu trữ tất cả các điểm trên contours mà không xấp xỉ. Điều này có thể tốn nhiều bộ nhớ hơn.
    - cv2.CHAIN_APPROX_SIMPLE: Xấp xỉ các contours bằng cách loại bỏ

các điểm thẳng hàng không cần thiết, như đã giải thích trước đó.

- Ngoài ra còn số phương pháp xấp xỉ đặc biệt khác không liên quan như L1 và Kcos.

➢ Kết quả trả về là các contours tìm được và biến phân cấp.

3.1.2.2. Nguyên lý hoạt động của các hàm khác có liên quan

a. Hàm vẽ contours – cv2.drawContours

$$cv2.drawContours(image, contours, contourIdx, color, thickness)$$

Trong đó:

➢ Tham số đầu vào:

- Image: Hình ảnh đầu vào(các contours sẽ được vẽ lên hình ảnh này)
- Contours: Danh sách các contours được tìm thấy bởi cv2.findContours
- ContoursIdx: Chỉ số của các contours cần được vẽ. Nếu chỉ số này là -1 thì tất cả các contours sẽ được vẽ.
- Color: Màu sắc của contours vẽ lên ảnh.
- Thickness: Độ dày của contours được vẽ

➢ Kết quả trả về: Hàm này không trả về kết quả mà kết quả sau khi thực hiện hàm này chính là hình ảnh đầu vào đã được vẽ contours.

b. Hàm tính chu vi, diện tích của contours

$$perimeter = cv2.arcLength(contour, isClosed)$$

Trong đó:

➢ Tham số đầu vào:

- Contour: Contour cần tính chu vi
- isClosed: biến này cho biết contour có đóng kín hay không. Nếu là true thì tức là đóng kín.

➢ Kết quả trả về: chu vi của contour truyền vào

➢ Hàm tính diện tích có tham số đầu vào và kết quả như trên

c. Hàm tìm hình chữ nhật bao quanh contours

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

Trong đó:

➢ Tham số đầu vào: Contour cần tìm hình chữ nhật bao quanh

- Kết quả trả về:
  - Tọa độ X của góc trên bên trái hình chữ nhật bao quanh contour
  - Tọa độ Y của góc trên bên trái hình chữ nhật bao quanh contour
  - W hay Width là chiều rộng của hình chữ nhật (chiều ngang)
  - H hay Height là chiều cao của hình chữ nhật (chiều dọc)

d. Hàm xấp xỉ contour với một đa giác

Hàm này là một hàm có ứng dụng rất nhiều không chỉ trong đề tài này mà còn trong các bài toán về thị giác máy tính khác. Đây là một hàm rất hữu ích để đơn giản hóa các contours phức tạp và giảm số lượng điểm cần xử lý. Hàm này như sau:

$$approx = cv2.approxPolyDP(curve, epsilon, closed)$$

Trong đó:

- Tham số đầu vào:
  - Curve: Contour cần xấp xỉ đây là một mảng numpy gồm tọa độ(x,y) của contour.
  - Epsilon: Độ chính xác gần đúng, khoảng cách tối đa từ contour ban đầu đến contour xấp xỉ. Giá trị này tỷ lệ thuận với độ chi tiết của contour xấp xỉ; giá trị càng lớn thì contour xấp xỉ càng đơn giản.
  - Closed: Biến boolean cho biết contour có kín hay không. Nếu là True, contour xấp xỉ sẽ được đóng kín.
- Kết quả trả về: Thường được gọi là approx là Contour xấp xỉ, đây là một mảng numpy chứa các điểm (tọa độ x, y) của contour xấp xỉ.

❖ Ngoài các hàm trên đây nhóm mình đã đề cập thì còn rất rất nhiều hàm khác hữu ích như hàm vẽ hình chữ nhật bao quanh, vẽ hình elipse quanh, hình chữ nhật bé nhất bao quanh contours,… nhưng vì ít dùng trong bài nên nhóm mình sẽ không đi sâu quá vào nó.

3.1.2.3. Triển khai quá trình phân tích và phát hiện contours

Đầu vào yêu cầu cho giai đoạn này là hình ảnh sau khi xử lý. Tối thiểu phải là hình ảnh nhị phân hóa, cao cấp hơn thì là các hình ảnh nhị phân hóa đã thông qua nhiều bước xử lý.

a. Gọi hàm tìm contours và xắp xếp các contours

```
contours, hierarchy = cv2.findContours(SobelImage, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)  # Tìm các đường viền trong ảnh
contours = sorted(contours, key=cv2.contourArea, reverse=True)[:10]  # Sắp xếp và lấy 10 đường viền có diện tích lớn nhất
screenCnt = []  # Danh sách các đường viền của biển số xe
# Tạo một ảnh trắng để vẽ các contour
imgColor = cv2.cvtColor(imgGray, cv2.COLOR_GRAY2BGR)
Sobel=cv2.cvtColor(SobelImage, cv2.COLOR_GRAY2BGR)
```

Hình 25: Gọi hàm tìm Contours, xắp xếp các contour theo thứ tự

Ở đoạn code trên đây, nhóm đã gọi hàm tìm contours trong chế độ lấy tất cả contours và hàm xấp xỉ contours là loại bỏ các điểm thẳng hàng không cần thiết. Vì đơn giản hình ảnh này sẽ có rất nhiều góc cạnh nên vẽ hết để dễ có cái nhìn tổng quát phục vụ cho việc phân tích sau này.

Gọi hàm sorted để xắp xếp mảng contours theo diện tích lớn nhất (lấy 10 contours)

Khởi tạo hàm ScreenCnt để lưu danh sách đường viền (sau này cần sử dụng)

Tạo 2 hình ảnh từ ảnh nhị phân hóa sau xử lý và ảnh xám để thấy được cách vẽ contours.

b. Vẽ tất cả các hình ảnh contours tìm được

Tiếp theo ta sẽ duyệt qua từng phần tử trong mảng contours, tính chu vi của các contours có nét liền.

Gọi hàm xấp xỉ đa giác với các contours có chu vi với độ chính xác gần đúng là 0.06* chu vi. Để tạo ra các đa giác bao quanh các contours. Và đóng kín đa giác. Gọi hàm drawContours để vẽ các contours lên hai hình ảnh đã tạo ở phía trên.

Cuối cùng là sử dụng thư viện plt để hiển thị hai hình ảnh sau khi được vẽ contours lên màn hình.

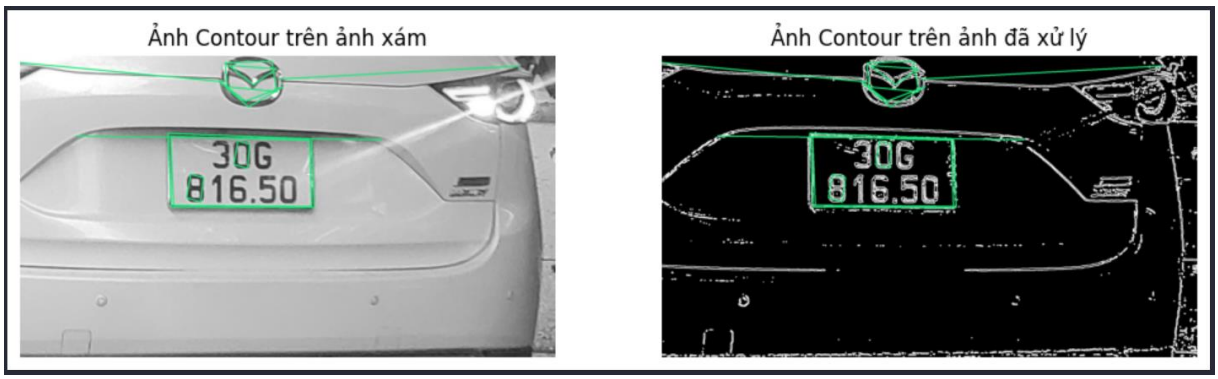
```
# Vẽ từng contour
for i, contour in enumerate(contours):
    # Tính chu vi của contour
    peri = cv2.arcLength(contour, True)

    # Xấp xỉ contour thành đa giác
    approx = cv2.approxPolyDP(contour, 0.06 * peri, True)

    # Vẽ đa giác lên ảnh
    cv2.drawContours(imgColor, [approx], -1, (31,229,124), 3)  # Khoanh vùng biển số xe
    cv2.drawContours(Sobel, [approx], -1, (31,229,124), 3)  # Khoanh vùng biển số xe
# Hiển thị ảnh
# Sử dụng Matplotlib để hiển thị ảnh gốc và ảnh kết quả
plt.figure(figsize=(12, 6))

# Vẽ ảnh gốc
plt.subplot(1, 2, 1)
plt.imshow(imgColor, cmap="gray")
plt.title("Ảnh Contour trên ảnh xám")
plt.axis("off")
# Vẽ ảnh kết quả với bounding box và ellipse
plt.subplot(1, 2, 2)
plt.imshow(Sobel)
plt.title("Ảnh Contour trên ảnh đã xử lý")
plt.axis("off")
plt.show()
```

Hình 26: Vẽ Approx lên hai hình ảnh đã tạo



Hình 27: Contours vẽ vào đúng các đường viền

c. Phân tích các contours đã khoanh vùng

```
for c in contours:
    epsilon = 0.06 * cv2.arcLength(c, True)
    approx = cv2.approxPolyDP(c, epsilon, True)
    if len(approx) == 4:  # Kiểm tra nếu đường viền xấp xỉ có 4 điểm (4 cạnh)

        # Kiểm tra xem các đường chéo gần với 90 độ
        angles = []
        for i in range(4):
            pt1 = tuple(approx[i][0])
            pt2 = tuple(approx[(i + 1) % 4][0])
            pt3 = tuple(approx[(i + 2) % 4][0])
            angle = math.degrees(math.atan2(pt3[1]-pt2[1], pt3[0]-pt2[0]) - math.atan2(pt1[1]-pt2[1], pt1[0]-pt2[0]))
            angle = abs(angle) if angle < 180 else 360 - abs(angle)
            angles.append(angle)

        if all(angle > 70 and angle < 100 for angle in angles):
            cv2.drawContours(img, [approx], -1, (0, 255, 0), 2)  # Vẽ đường viền xấp xỉ bằng màu xanh lá
```

Hình 28: Code phân tích để lọc ra contour biển số

Trong hàm này ta sẽ duyệt qua các phần tử trong mảng contours một lần nữa, tính cho chu vi và sử dụng xấp xỉ đa giác tương tự như bên trên. Sau đó, ta sẽ bắt đầu phân tích. Để phân tích thì ta thường sẽ áp dụng các kiến thức về hình học vào. Ở đây ta có thể nhận thấy những điều sau:

- Biển số xe bắt buộc phải có 4 cạnh.
- Biển số xe phải có các góc là góc vuông (hoặc gần vuông)

Đoạn code trên được giải thích như sau :

- Đầu tiên ta sẽ kiểm tra xem approx có phải là đa giác có 4 cạnh không
  - Tiếp theo là tạo một vòng for : Vòng lặp này lặp qua bốn điểm trong approx.
  - pt1, pt2, và pt3 là ba điểm liên tiếp từ danh sách. Bằng cách sử dụng phép chia dư 4 (% 4), nó đảm bảo các điểm được liên kết một cách vòng tròn, duy trì hình dạng đóng kín.
  - Hàm math.atan2 tính toán góc của vector giữa hai điểm. Sự khác biệt giữa các góc này cho góc nội tại tại pt2.
  - math.degrees chuyển đổi góc từ radian sang độ.
  - Dòng angle = abs(angle) if angle < 180 else 360 - abs(angle) đảm bảo góc nằm trong khoảng [0, 180] độ, đại diện cho góc nhỏ nhất giữa các cạnh.
  - Góc được tính toán sẽ được thêm vào danh sách angles.
  - Tiếp theo chúng ta sẽ tiến hành kiểm tra hai góc của đa giác. Nếu chúng nằm trong khoảng từ 70 – 100 độ thì đó chính là hình ảnh mà chúng ta cần tìm.

❖ Bây giờ chúng ta sẽ sử dụng matplotlib hiển thị kết quả để xem kết quả thế nào.

Hình 29: Sau xử lý thì chỉ còn duy nhất một contours tại đúng vị trí của biển số xe

d. Cắt hình ảnh sau khi lấy được contour cần tìm

Để cắt được hình ảnh từ contour đã xác định thì ta phải làm theo từng bước sau:

❖ Đầu tiên ta sẽ phải lưu contour tìm được vào một biến.

❖ Tiếp theo chúng ta sẽ xác định độ nghiêng của biển số. Để xoay lại hình ảnh nếu bị nghiêng.

- Để tìm góc nghiêng thì ta phải tính ra các góc của biển số
- Trích xuất tọa độ của bốn điểm góc từ một mảng screenCnt.
- Tạo một danh sách chứa các tọa độ này và sắp xếp chúng theo tọa độ y (từ cao đến thấp).
- Trích xuất lại tọa độ của hai điểm có tọa độ y lớn nhất.
- Tính toán độ chênh lệch giữa các giá trị x và y của hai điểm này.
- Tính toán góc nghiêng của hình dạng (có thể là biển số xe) dựa trên độ chênh lệch này.

```
############### Tìm góc của biển số xe ######################
(x1, y1) = screenCnt[0, 0]
(x2, y2) = screenCnt[1, 0]
(x3, y3) = screenCnt[2, 0]
(x4, y4) = screenCnt[3, 0]
array = [[x1, y1], [x2, y2], [x3, y3], [x4, y4]]
sorted_array = sorted(array, reverse=True, key=lambda x: x[1])
(x1, y1) = sorted_array[0]
(x2, y2) = sorted_array[1]
doi = abs(y1 - y2)  # Tính độ chênh lệch y
ke = abs(x1 - x2)  # Tính độ chênh lệch x
angle = math.atan(doi / ke) * (180.0 / math.pi)  # Tính góc nghiêng của biển số

#####################################
```

Hình 30: Hàm tìm góc nghiêng của biển số xe

❖ Cắt biển số xe.

```
########## Cắt biển số và căn chỉnh đúng góc ################
mask = np.zeros(anh_xam.shape, np.uint8)  # Tạo mặt nạ với kích thước bằng ảnh gốc
new_image = cv2.drawContours(mask, [screenCnt], 0, 255, -1)  # Vẽ đường viền biển số lên mặt nạ
cv2.imwrite("output/mask.jpg", new_image)
# Cắt ảnh
(x, y) = np.where(mask == 255)  # Lấy tọa độ của các pixel nằm trong biển số
(topx, topy) = (np.min(x), np.min(y))  # Tọa độ trên cùng của biển số
(bottomx, bottomy) = (np.max(x), np.max(y))  # Tọa độ dưới cùng của biển số

roi = img[topx:bottomx, topy:bottomy]  # Cắt vùng ảnh chứa biển số

imgThresh = anh_nhi_phan[topx:bottomx, topy:bottomy]  # Cắt vùng ảnh ngưỡng chứa biển số
ptPlateCenter = (bottomx - topx) / 2, (bottomy - topy) / 2  # Tính trung điểm của biển số
if x1 < x2:
    rotationMatrix = cv2.getRotationMatrix2D(ptPlateCenter, -angle, 1.0)  # Ma trận xoay với góc âm
else:
    rotationMatrix = cv2.getRotationMatrix2D(ptPlateCenter, angle, 1.0)  # Ma trận xoay với góc dương

roi = cv2.warpAffine(roi, rotationMatrix, (bottomy - topy, bottomx - topx))  # Xoay ảnh biển số
imgThresh = cv2.warpAffine(imgThresh, rotationMatrix, (bottomy - topy, bottomx - topx))  # Xoay ảnh ngưỡng biển s
roi = cv2.resize(roi, (0, 0), fx=3, fy=3)  # Phóng to ảnh biển số
imgThresh = cv2.resize(imgThresh, (0, 0), fx=3, fy=3)  # Phóng to ảnh ngưỡng biển số
# Lưu lại ảnh
cv2.imwrite("output/contours_detected.jpg", img)
cv2.imwrite("output/cropped_plate.jpg", roi)
cv2.imwrite("output/cropped_threshold_plate.jpg", imgThresh)
```

Hình 31: Code cắt biển số xe và lưu ra file

➢ Tạo mặt nạ với kích thước bằng ảnh gốc:

- Loại bỏ nhiễu: Chỉ giữ lại phần biển số xe, loại bỏ các phần không liên quan của ảnh gốc.
- Tách biệt vùng quan tâm: Dễ dàng xử lý và phân tích phần biển số xe mà không bị ảnh hưởng bởi các phần khác của ảnh.
- Chuẩn bị cho các bước xử lý tiếp theo: Cắt và xoay ảnh biển số một cách

chính xác, đặc biệt khi ảnh có góc nghiêng.

- Vẽ đường viền biển số lên mặt nạ.
- Lấy tọa độ của các pixel nằm trong biển số.
- Cắt vùng ảnh chứa biển số và ảnh ngưỡng[1] chứa biển số.
- Tính trung điểm của biển số.
- Tạo ma trận xoay và xoay ảnh biển số.
- Phóng to ảnh biển số và ảnh ngưỡng biển số.
- • Lưu lại ảnh:
- cv2.imwrite("output/contours_detected.jpg", img): Lưu ảnh với các đường viền đã khoanh vùng.
- cv2.imwrite("output/cropped_plate.jpg", roi): Lưu ảnh biển số đã cắt.
- cv2.imwrite("output/cropped_threshold_plate.jpg", imgThresh): Lưu ảnh ngưỡng biển số đã cắt.

❖ Sau khi có được hình ảnh biển số xe từ ảnh tạp ta sẽ chuyển sang bước tiếp theo.

Hình 32: Ảnh gốc

[1] Còn được gọi là ảnh nhị phân hóa

Hình 33: Ảnh sau khi bóc tách biển số

3.1.3. Phân tách ký tự từ biển số xe

Đầu vào của bước này là hình ảnh biển số xe, đầu ra của giai đoạn này là danh sách hình ảnh các ký tự được bóc tách từ biển số đó.

3.1.3.1. Tiền xử lý ảnh biển số xe

Về cơ bản các bước tiền xử lý cũng giống như ở bước khoanh vùng biển số nhưng đơn giản hơn rất nhiều. Nguyên nhân là do ảnh biển số xe chủ yếu hiển thị rất rõ các góc cạnh. Có lẽ công việc khó khăn nhất trong giai đoạn này là làm thế nào có thể tách chiết được cách ký tự một cách chính xác nhất và không lặp lại.

Hình 34: Hình ảnh sau quá trình tiền xử lý

3.1.3.2. Phân tách ký tự từ biển số xe

Sau khi tiến hành tiền xử lý biển số xe, chúng ta sẽ dựa vào các đặc điểm của biển số xe để tìm ra các contours phù hợp và vẽ hình chữ nhật đúng khung chữ cần lấy. Dưới đây là các bước cụ thể:

```
    kerel3 = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))  # Tạo kernel hình chữ nhật
    thre_mor = cv2.morphologyEx(imgThresh, cv2.MORPH_DILATE, kerel3)  # Áp dụng phép dãn ảnh
    cont, hier = cv2.findContours(thre_mor, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)  # Tìm các đường viền trong ảnh đã dãn
```

Hình 35: Tạo Kernel hình chữ nhật và tìm contours

```
    ###################### Lọc các ký tự #################
    char_x_ind = {}  # Tạo dictionary lưu chỉ số của các ký tự
    char_x = []  # Danh sách lưu tọa độ x của các ký tự
    kichthuoc = cv2.imread("output/cropped_threshold_plate.jpg")
    height, width, _ = kichthuoc.shape  # Lấy kích thước ảnh biển số
    roiarea = height * width  # Tính diện tích ảnh biển số
    for ind, cnt in enumerate(cont):
        (x, y, w, h) = cv2.boundingRect(cont[ind])  # Tạo hình chữ nhật bao quanh ký tự
        ratiochar = w / h  # Tính tỷ lệ chiều rộng / chiều cao của ký tự
        char_area = w * h  # Tính diện tích của ký tự
        if (Min_char * roiarea < char_area < Max_char * roiarea) and (0.1 < ratiochar < 0.7):  # Lọc các ký tự hợp lệ
            if x in char_x:  # Nếu tọa độ x đã tồn tại, tăng x lên 1 để tránh trùng lặp
                x = x + 1
            char_x.append(x)  # Thêm tọa độ x vào danh sách
            char_x_ind[x] = ind  # Lưu chỉ số của ký tự vào dictionary
    # Sắp xếp các tọa độ x theo thứ tự tăng dần
    char_x = sorted(char_x)
    # Sao chép hình ảnh gốc để vẽ tất cả các hình chữ nhật
    roi_with_rects = roi.copy()
    line1=[]
    line2=[]
```

Hình 36: Phân tích contours

Ở đoạn code này chúng ta sẽ phải tạo một mảng để lưu ký tự và một dictionary để lưu chỉ số của các ký tự. Tại sao lại phải lưu như vậy vì đơn giản các ký tự khi được cắt ra có thể có thứ tự không đúng theo thứ tự khi cắt, ta phải lưu chỉ số lại để sau này sắp xếp theo đúng thứ tự thực của biển số.

Tiếp theo chúng ta sẽ lọc các contour dựa trên diện tích ký tự. Phần này cũng giống như cách thức thực hiện mà mình đã làm ở bước khoanh vùng biển số. Cuối

cùng chúng ta sẽ tạo 2 mảng để lưu ký tự của 2 dòng riêng biệt vì có biển có 2 dòng và có biển có 1 dòng.

Sau khi đã lọc được contour thì ta sẽ thực hiện câu lệnh sau:

```
for i in char_x:
    (x, y, w, h) = cv2.boundingRect(cont[char_x_ind[i]])  # Tạo hình chữ nhật bao quanh ký tự
    # Vẽ hình chữ nhật bao quanh ký tự với màu mới
    cv2.rectangle(roi_with_rects, (x, y), (x + w, y + h), (0,255,0), 2)
    char_image = roi[y:y+h, x:x+w]
    if 4.5 > h / w > 1 and h > height/2.6 and x>1  :
        if y < height/3 :
            i=1
            char_image = pre.dao_den_thanh_trang(char_image)
            # char_image = add_padding(char_image)
            line1.append(char_image)
        else :
            i=2
            char_image = pre.dao_den_thanh_trang(char_image)
            # char_image = add_padding(char_image)
            line2.append(char_image)
```

Hình 37: Vẽ hình chữ nhật bao quanh ký tự và cắt ký tự

Về cơ bản nguyên lý hoạt động của các dòng code cũng giống như bước khoanh vùng và cắt biển số phía trên. Cũng tạo hình chữ nhật bao quanh và lấy tọa độ của hình chữ nhật đó để ccắt và thêm ảnh đó và từng mảng dòng của chúng. Nếu tọa độ y của hình chữ nhật và nhỏ hơn 1/3 chiều cao ảnh thì thêm vào dòng 1 ngược lại thì thêm vào dòng 2.

Hình 38: Khoanh vùng các ký tự

Ở bước trên đây có một hàm đặc biệt gọi là "dao_den_thanh_trang". Đúng như cái tên của nó hàm này sẽ đảo ngược tất cả màu trong ảnh từ đen thành trắng. Trong trường hợp này khi cắt là nền trắng ký tự đen và sẽ chuyển thành nền đen ký tự trắng.

Tại sao lại phải sử dụng hàm này là bởi vì ở giai đoạn 2 tập dữ liệu huấn luyện là nền đen chữ trắng.

Sau khi có được 2 line là line1 và line2 thì ta sẽ nối 2 mảng này lại để được một mảng ký tự hoàn chỉnh để mang đi dự đoán.

3.2. Giai đoạn 2 – triển khai mô hình nhận diện ký tự

Trong giai đoạn này dữ liệu sẽ là tập train và test lấy ở trên kaggle gồm các hình ảnh chữ và số.

Chúng ta sẽ sử dụng mô hình này để huấn luyện một mô hình mạng neural cho phép nhận diện được ký tự đó là ký tự nào dựa theo dữ liệu đã học.

3.2.1. Unzip data, import thư viện và đọc data từ folder thư mục

Về cơ bản chúng ta sẽ unzip thư mục data nếu nó chưa được unzip, import các thư viện tensorflow phục vụ cho việc xử lý hình ảnh và huấn luyện mô hình sử dụng tensorflow cũng như import các thành phần cần thiết của TensorFlow và keras để tối ưu hóa và một số thành phần khác.

```
!unzip data
```

```
import matplotlib.pyplot as plt
import numpy as np
import cv2
import tensorflow as tf
from tensorflow.keras import optimizers
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import Dense, Flatten, MaxPooling2D, Dropout, Conv2D
import tensorflow.keras.backend as K
from keras.layers import InputLayer
```

Hình 39: Import các thư viện cần dùng

```
train_datagen = ImageDataGenerator(rescale=1./255, width_shift_range=0.1, height_shift_range=0.1)

train_generator = train_datagen.flow_from_directory(
        'data/val',  # this is the target directory
        target_size=(28,28),  # all images will be resized to 28x28
        batch_size=64,
        class_mode='sparse')

validation_generator = train_datagen.flow_from_directory(
        'data/test',  # this is the target directory
        target_size=(28,28),  # all images will be resized to 28x28 batch_size=1,
        class_mode='sparse')
```

Python

```
Found 14000 images belonging to 35 classes.
Found 3500 images belonging to 35 classes.
```

+ Code + Markdown

Hình 40: Đoạn code dọc dữ liệu từ folder train và test

- Tạo một đối tượng ImageDataGenerator để tạo ra các biến thể của dữ liệu huấn luyện. Trong trường hợp này, hình ảnh sẽ được chia tỷ lệ lại theo tỉ lệ 1/255 (để chuẩn hóa các giá trị pixel về khoảng [0, 1]) và áp dụng các phép biến đổi như dịch chuyển chiều rộng và cao.
- Sử dụng phương thức flow_from_directory của ImageDataGenerator để tạo ra các luồng dữ liệu từ các thư mục chứa hình ảnh. Cụ thể:
  - directory: Đường dẫn đến thư mục chứa các hình ảnh.
  - target_size: Kích thước mà tất cả các hình ảnh sẽ được resize về ở đây là 28*28
  - batch_size: Số lượng hình ảnh trong mỗi lô.
  - class_mode: Chế độ phân loại của dữ liệu, trong trường hợp này là 'sparse' cho dữ liệu phân loại.
- Dòng mã trên tạo ra một mạng nơ-ron tích chập (CNN) sử dụng thư viện TensorFlow và Keras. Dưới đây là giải thích chi tiết từng dòng code:

3.2.2. Khởi tạo mô hình mạng neural tích chập

```
K.clear_session()
model = Sequential()
model.add(InputLayer(input_shape=(28, 28, 3)))  # Chỉ rõ input_shape ở đây
model.add(Conv2D(16, (22,22),  activation='relu', padding='same'))
model.add(Conv2D(32, (16,16), activation='relu', padding='same'))
model.add(Conv2D(64, (8,8),  activation='relu', padding='same'))
model.add(Conv2D(64, (4,4), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(4, 4)))
model.add(Dropout(0.4))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(36, activation='softmax'))
model.compile(loss='sparse_categorical_crossentropy', optimizer=optimizers.Adam(learning_rate=0.0001), metrics=['accuracy'])
```

Hình 41: Khởi tạo mô hình mạng neural tích chập CNN

- Xóa tất cả các mô hình hoặc các phần tử trong mô hình trước đó từ bộ nhớ của TensorFlow. Điều này giúp làm sạch không gian bộ nhớ và đảm bảo rằng không có sự xung đột giữa các mô hình khi tạo ra và huấn luyện chúng.
- Tạo một đối tượng mô hình tuần tự (Sequential), một kiểu mô hình trong Keras cho các mô hình mạng nơ-ron tuần tự, trong đó các lớp được xếp chồng lên nhau một cách tuần tự.
- Thêm một lớp đầu vào vào mô hình với kích thước đầu vào là (28, 28, 3), tức là hình ảnh có kích thước 28x28 pixel và 3 kênh màu (RGB).
- model.add(Conv2D(16, (22,22), activation='relu', padding='same')):
- Thêm một lớp tích chập với 16 bộ lọc kích thước (22,22), hàm kích hoạt là ReLU và phương pháp đệm là 'same' để giữ nguyên kích thước của đầu ra như kích thước đầu vào.
- Thêm một lớp tích chập với 32 bộ lọc kích thước (16,16), hàm kích hoạt là ReLU và phương pháp đệm là 'same'.
- Thêm một lớp tích chập với 64 bộ lọc kích thước (8,8), hàm kích hoạt là ReLU và phương pháp đệm là 'same'.
- Thêm một lớp tích chập với 64 bộ lọc kích thước (4,4), hàm kích hoạt là ReLU và phương pháp đệm là 'same'.
- Thêm một lớp MaxPooling với kích thước cửa sổ (pool_size) là (4,4), giảm kích thước của đầu ra mỗi lần tích chập để giảm chiều dài của mạng và giảm việc tính toán.
- Thêm một lớp Dropout với tỷ lệ dropout là 0.4, là một kỹ thuật regularization để giảm overfitting bằng cách loại bỏ ngẫu nhiên một số lượng nơ-ron trong quá trình huấn luyện.
- Thêm một lớp Flatten để chuyển từ tensor 2D sang vector 1D, là bước cần thiết trước khi đưa vào các lớp kết nối đầy đủ (Dense) ở phần cuối của mạng.
- Thêm một lớp kết nối đầy đủ (fully connected) với 128 nơ-ron và hàm kích hoạt là ReLU.
- Thêm một lớp kết nối đầy đủ (fully connected) cuối cùng với 36 nơ-ron (tương ứng với số lượng lớp đầu ra, trong trường hợp này là 36 loại ký tự) và hàm kích hoạt là softmax để đưa ra xác suất dự đoán cho từng lớp.
- Biên soạn mô hình với hàm mất mát là sparse categorical crossentropy (dùng cho

bài toán phân loại có nhiều lớp), tối ưu hóa bằng trình tối ưu hóa Adam với tốc độ học là 0.0001 và đánh giá mô hình sử dụng độ chính xác (accuracy).

❖ Xem mô hình tổng quan bằng câu lệnh model.sumary()

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 28, 16)	23,248
conv2d_1 (Conv2D)	(None, 28, 28, 32)	131,104
conv2d_2 (Conv2D)	(None, 28, 28, 64)	131,136
conv2d_3 (Conv2D)	(None, 28, 28, 64)	65,600
max_pooling2d (MaxPooling2D)	(None, 7, 7, 64)	0
dropout (Dropout)	(None, 7, 7, 64)	0
flatten (Flatten)	(None, 3136)	0
dense (Dense)	(None, 128)	401,536
dense_1 (Dense)	(None, 36)	4,644

```
Total params: 757,268 (2.89 MB)

Trainable params: 757,268 (2.89 MB)
```

Hình 42: Mô hình tổng quan

3.2.3. Huấn luyện mô hình

```
class stop_training_callback(tf.keras.callbacks.Callback):
    def on_epoch_end(self, epoch, logs={}):
        # Kiểm tra độ chính xác trên tập validation
        if logs.get('val_accuracy', 0) >= 0.99:
            self.model.stop_training = True
```

```
batch_size = 64
callbacks = [stop_training_callback()]
history= model.fit(
      train_generator,
      steps_per_epoch = train_generator.samples // batch_size,
      validation_data = validation_generator,
      epochs = 40, verbose=1, callbacks=callbacks)
```

Đoạn code trên định nghĩa một callback tùy chỉnh trong TensorFlow/Keras và sử dụng nó để dừng quá trình huấn luyện mô hình khi đạt được một độ chính xác nhất định trên tập validation. Dưới đây là giải thích chi tiết từng dòng:

- Định nghĩa một lớp callback tùy chỉnh có tên là stop_training_callback, kế thừa từ tf.keras.callbacks.Callback. Callbacks trong TensorFlow/Keras là các đối tượng được sử dụng để thực hiện các hành động tùy chỉnh trong quá trình huấn luyện, chẳng hạn như dừng lại ở một số điều kiện nhất định.
- Định nghĩa phương thức on_epoch_end, được gọi khi mỗi epoch kết thúc.
- Kiểm tra nếu độ chính xác trên tập validation (val_accuracy) đạt hoặc vượt qua ngưỡng 0.99.
- Nếu đạt được điều kiện, thiết lập thuộc tính stop_training của mô hình thành True. Điều này sẽ dừng quá trình huấn luyện mô hình.
- Định nghĩa kích thước của mỗi batch được sử dụng trong quá trình huấn luyện.
- Tạo một danh sách callbacks, trong trường hợp này chỉ có một callback là stop_training_callback được định nghĩa ở trên.
- Bắt đầu quá trình huấn luyện mô hình bằng phương thức fit() của mô hình.
  - train_generator: Dữ liệu huấn luyện được tạo ra từ generator train_generator.
  - steps_per_epoch: Số bước (lượt batch) mà mỗi epoch sẽ chạy qua dữ liệu huấn luyện. Đối với generator, samples là số lượng mẫu có sẵn và batch_size đã được định nghĩa, vì vậy train_generator.samples // batch_size là số bước mỗi epoch.
  - validation_data: Dữ liệu validation được tạo ra từ generator validation_generator.
  - epochs: Số lượng epochs (vòng lặp qua toàn bộ tập dữ liệu) mà mô hình sẽ được huấn luyện.
  - verbose: Xác định cách hiển thị tiến trình huấn luyện, trong trường hợp này là 1 để hiển thị tiến trình.
  - callbacks: Danh sách các callbacks được sử dụng trong quá trình huấn luyện, ở đây là callback stop_training_callback() được sử dụng để dừng quá trình huấn luyện nếu đạt được độ chính xác mong muốn trên tập validation.
- Kết quả huấn luyện: Độ chính xác đạt 97 % tại epoch thứ 39

```
Epoch 39/40
218/218 ━━━━━━━━━━━━━━━━━━━━ 103s 469ms/step - accuracy: 0.9828 - loss: 0.0555 - val_accuracy: 0.9706 - val_loss: 0.1177
Epoch 40/40
```

Hình 43: Epoch cao nhất đạt được

❖ Lưu model: để sau này đỡ train lại và có thể gửi được cho người khác.

3.2.4. Vẽ biểu đồ đánh giá độ chính xác và hàm mất mát

```python
def plot_history(history):
    # Lấy các giá trị loss và accuracy từ history
    acc = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    loss = history.history['loss']
    val_loss = history.history['val_loss']

    epochs = range(1, len(acc) + 1)

    # Vẽ biểu đồ accuracy
    plt.figure(figsize=(12, 6))
    plt.subplot(1, 2, 1)
    plt.plot(epochs, acc, 'bo-', label='Training accuracy')
    plt.plot(epochs, val_acc, 'ro-', label='Validation accuracy')
    plt.title('Training and validation accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy')
    plt.legend()

    # Vẽ biểu đồ loss
    plt.subplot(1, 2, 2)
    plt.plot(epochs, loss, 'bo-', label='Training loss')
    plt.plot(epochs, val_loss, 'ro-', label='Validation loss')
    plt.title('Training and validation loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend()
```

Hình 44: Hàm vẽ biểu đồ đánh giá mô hình huấn luyện

Hàm plot_history(history) được sử dụng để vẽ biểu đồ hiển thị quá trình huấn luyện của mô hình, bao gồm độ chính xác (accuracy) và độ mất mát (loss) cho cả tập huấn luyện và tập validation theo từng epoch. Dưới đây là giải thích trong hàm này:

Định nghĩa hàm plot_history nhận vào tham số history, là đối tượng History trả về từ phương thức fit() của mô hình Keras.

Lấy danh sách các giá trị độ chính xác (accuracy) trên tập huấn luyện qua các epoch từ đối tượng history.

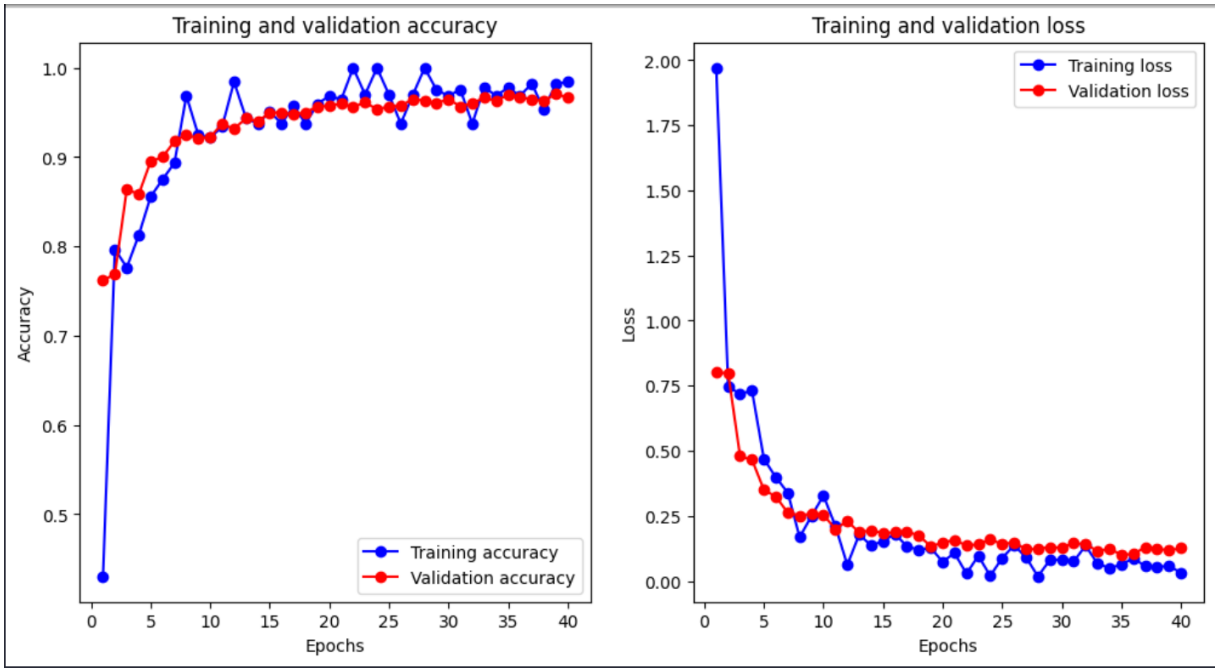
Lấy danh sách các giá trị độ chính xác (accuracy) trên tập validation qua các

epoch từ đối tượng history.

- ❖ Lấy danh sách các giá trị độ mất mát (loss) trên tập huấn luyện qua các epoch từ đối tượng history.
- ❖ Lấy danh sách các giá trị độ mất mát (loss) trên tập validation qua các epoch từ đối tượng history.
- ❖ Tạo danh sách các epoch từ 1 đến số lượng epoch đã huấn luyện.
- ❖ Vẽ biểu đồ đường cho độ chính xác trên tập huấn luyện, với 'bo-' chỉ ra rằng các điểm dữ liệu sẽ là các vòng tròn màu xanh (blue circles) và có đường nối.
- ❖ Vẽ biểu đồ đường cho độ chính xác trên tập validation, với 'ro-' chỉ ra rằng các điểm dữ liệu sẽ là các vòng tròn màu đỏ (red circles) và có đường nối.
- ❖ Gọi hàm plot_history với tham số là đối tượng history đã được tạo ra từ quá trình huấn luyện mô hình (model.fit()).
- ❖ Về subplot thứ hai thì có cơ chế tương tự

Mục đích của hàm này là trực quan hóa quá trình huấn luyện của mô hình, giúp bạn dễ dàng đánh giá mô hình có overfitting hay underfitting không, cũng như xem xét sự khác biệt giữa tập huấn luyện và tập validation.

- ❖ Kết quả đánh giá:



Hình 45: Kết quả huấn luyện mô hình

❖ Nhận xét biểu đồ:

- Độ chính xác (Accuracy):
  - Đường màu xanh biểu diễn độ chính xác trên tập huấn luyện.
  - Đường màu đỏ biểu diễn độ chính xác trên tập kiểm tra (validation).
  - Nếu khoảng cách giữa hai đường này không lớn, thì không có dấu hiệu của overfitting.
- Hàm mất mát (Loss):
- Đường màu xanh biểu diễn hàm mất mát trên tập huấn luyện.
- Đường màu đỏ biểu diễn hàm mất mát trên tập kiểm tra.
- Nếu cả hai đường này giảm theo thời gian mà không có khoảng cách lớn giữa chúng, thì cũng không có dấu hiệu của overfitting.

3.3. Kết hợp 2 giai đoạn

Đầu tiên chúng ta sẽ phải import thư viện để load model đã huấn luyện và lấy mảng ký tự đã tách từ giai đoạn phía trên

```
from tensorflow.keras.models import load_model

# Load the entire model back.
model = load_model('model.keras')
char_list = line1+line2
```

Hình 46: Import thư viện và load model, lấy mảng ký tự đã tách từ giai đoạn trên

```python
# Predicting the output
def fixdimension(img):
    new_img = np.zeros((28,28,3))
    for i in range(3):
        new_img[:,:,i] = img
    return new_img

def show_results(char):
    dic = {}
    characters = '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ'
    for i, c in enumerate(characters):
        dic[i] = c

    output = []
    for ch in char: # iterating over the characters
        img = cv2.resize(ch, (28, 28), interpolation=cv2.INTER_LINEAR)

        img = fixdimension(img)
        img = img.reshape(1, 28, 28, 3) # preparing image for the model
        y = model.predict(img)[0] # predicting the class
        class_index = np.argmax(y) # get the index of the highest probability class
        character = dic[class_index] # map index to character
        output.append(character) # storing the result in a list

    plate_number = ''.join(output)

    return plate_number
# Get predicted characters
```

(variable) img: Unknown

Hình 47: Hàm sử dụng để chuyển đổi ảnh và hiển thị kết quả dự đoán

Vì các hình ảnh sau khi bóc tách từ mảng không phải là 28*28 như khi ta huấn luyện nên chúng ta sẽ cần có một hàm để chuyển đổi hình ảnh truyền vào thành hình ảnh 28*28 (hàm Fixdimension).

Tiếp theo là hàm sử dụng model để dự đoán. Dưới đây là giải thích chi tiết của hàm này:

- dic = {}: Tạo một dictionary trống.
- characters = '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ': Một chuỗi chứa các ký tự có thể nhận diện.
- for i, c in enumerate(characters): dic[i] = c: Lặp qua các ký tự và lập một dictionary ánh xạ từ chỉ số (index) sang ký tự tương ứng.
- output = []: Tạo một danh sách trống để chứa các kết quả nhận diện.
- for ch in char: Lặp qua từng ảnh ký tự trong danh sách char.
  - img = cv2.resize(ch, (28, 28), interpolation=cv2.INTER_LINEAR): Thay đổi kích thước ảnh về 28x28 pixel.

- img = fixdimension(img): Gọi hàm fixdimension để chuyển ảnh grayscale thành ảnh ba kênh.
- img = img.reshape(1, 28, 28, 3): Định hình lại mảng numpy để phù hợp với đầu vào của mô hình dự đoán (một batch chứa một ảnh kích thước 28x28x3).
- y = model.predict(img)[0]: Dùng mô hình model để dự đoán lớp của ảnh, lấy kết quả đầu tiên (trường hợp mô hình trả về batch).
- class_index = np.argmax(y): Lấy chỉ số của lớp có xác suất cao nhất.
- character = dic[class_index]: Ánh xạ chỉ số lớp sang ký tự tương ứng.
- output.append(character): Thêm ký tự vào danh sách kết quả.

➢ plate_number = ''.join(output): Ghép các ký tự lại thành chuỗi duy nhất.

return plate_number: Trả về chuỗi kết quả cuối cùng.

Sau khi đã có được hai hàm này, công việc còn lại của chúng ta là gọi nó ra và sử dụng tham số truyền vào là ký tự mà chúng ta đã phân tách rồi gán kết quả vào một mảng mới. Cuối cùng là sử dụng matplotlib để đưa ra kết quả dạng hình ảnh để có cái nhìn trực quan nhất.

Hình 48: Kết quả của mô hình

KẾT LUẬN

Như vậy là trải qua ba chương nhóm mình đã cùng với các bạn giải quyết được phần cốt yếu của bài toán đã đề ra dựa trên mô hình AI thị giác máy tính. Hi vọng trải qua bài này các bạn có thể có cái nhìn tổng quan nhất về trí tuệ nhân tạo và hơn hết là các bạn sẽ có được những cái nhìn rõ nhất về ứng dụng của thị giác máy tính trong cuộc sống hiện đại. Đồng thời cũng hiểu được cơ chế làm việc của nó trong môi trường công nghệ số.

Bài làm của nhóm là phần tổng quát hóa để các bạn có thể dễ dàng đọc và tìm hiểu, không phải là một bài báo cáo quá chuyên sâu nên không thể tránh khỏi thiếu sót. Kính mong các thầy, cô và các bạn độc giả có thể nhận xét về những thiếu sót của bài báo cáo và góp ý với nhóm mình. Nhóm xin chân thành cảm ơn!

TÀI LIỆU THAM KHẢO

Sinh, T. V. (2024). *Slide bài giảng bộ môn trí tuệ nhân tạo.* Học Viện Ngân Hàng.

Thanh, P. (2021). *Youtube*. Retrieved from Youtube.com: https://www.youtube.com/@phamthanhscience