

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

Giáo viên hướng dẫn: TS. Vũ Trọng Sinh

Lớp học phần: 232IS54A01 – Nhóm 16

Danh sách thành viên:

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

Hà Nội, 6/2024

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

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
4. Ngô Văn Minh	24A4042598
5. Nguyễn Cảnh Phong	24A4042605

Hà Nội, 6/2024

BẢNG PHÂN CÔNG CÔNG VIỆC

STT	Mã SV	Họ và tên	Phân công	Đánh giá
1	24A4042424	Phạm Bảo Anh	<ul style="list-style-type: none">• Tìm ảnh• Định hướng xây dựng bài toán	18.5%
2	24A4042425	Hà Gia Bảo	<ul style="list-style-type: none">• Phân tách biểu số• Phân mảnh ký tự	21%
3	24A4042427	Nguyễn Đức Công (Leader)	<ul style="list-style-type: none">• Phân tách biểu số• Phân mảnh ký tự• Làm báo cáo	21%
4	24A4042598	Ngô Văn Minh	<ul style="list-style-type: none">• Huấn luyện mô hình CNN• Phân mảnh ký tự	21%
5	24A4042605	Nguyễn Cảnh Phong	<ul style="list-style-type: none">• Tìm ảnh• Test mô hình	18.5%

Link GitHubRepository [tại đây!](#)

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

Em xin cam đoan kết quả đạt được trong báo cáo là sản phẩm nghiên cứu, tìm hiểu của riêng nhóm. Trong toàn bộ nội dung của báo cáo, những điều được trình bày hoặc là của nhóm 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	ii
LỜI CẢM ƠN	iii
LỜI CAM ĐOAN	iv
MỤC LỤC	v
DANH MỤC HÌNH ẢNH	1
LỜI MỞ ĐẦU	4
CHƯƠNG 1: GIỚI THIỆU BÀI TOÁN	5
1.1. GIỚI THIỆU BÀI TOÁN	5
1.2. ĐỀ XUẤT GIẢI QUYẾT BÀI TOÁN	5
1.2.1. Sơ đồ tổng quát bài toán	5
1.2.2. Tổng quan các bước thực hiện	6
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT	8
2.1. CƠ BẢN VỀ THỊ GIÁC MÁY TÍNH	8
2.2. ỨNG DỤNG VÀO BÀI TOÁN THỰC TẾ	9
2.2.1. Thư viện sử dụng	9
2.2.2. Thuật toán Convolutional Neural Network	10
2.2.3. Các kỹ thuật cần sử dụng	14
CHƯƠNG 3: TRIỂN KHAI BÀI TOÁN	20
3.1. GIAI ĐOẠN 1 – NHẬN DIỆN HÌNH ẢNH BIỂN SỐ XE TỪ ẢNH CHỤP	20
3.1.1. Đọc ảnh và xử lý ảnh	20
3.1.2. Khoanh vùng biển số - phát hiện và phân tích contour	27
3.1.3. Phân tách ký tự từ biển số xe	37
3.2. GIAI ĐOẠN 2 – TRIỂN KHAI MÔ HÌNH NHẬN DIỆN KÝ TỰ	40
3.2.1. Unzip data, import thư viện và đọc data từ folder thư mục	40
3.2.2. Khởi tạo mô hình mạng neural tích chập	41
3.2.3. Huấn luyện mô hình	43
3.2.4. Vẽ biểu đồ đánh giá độ chính xác và hàm mất mát	45
3.3. KẾT HỢP 2 GIAI ĐOẠN	47
KẾT LUẬN	50
TÀI LIỆU THAM KHẢO	51

DANH MỤC HÌNH ẢNH

Hình 1. Sơ đồ tổng quát bài toán	6
Hình 2. Bước tách biên số xe từ ảnh chụp.....	6
Hình 3. Bước phân đoạn ký tự từ biên số đã cắt	7
Hình 4. Bước dự đoán các ký tự biên số	7
Hình 5. Mô hình mạng CNN	11
Hình 6. Convolutional Layer	12
Hình 7. Pooling Layer	13
Hình 8: Ảnh màu và ảnh xám.....	15
Hình 9: Ảnh sau quá trình tiền xử lý	15
Hình 10: Ảnh bị nhiễu Gaus	16
Hình 11: Khoanh một vùng duy nhất trong ảnh	17
Hình 12: Ảnh góc bị lệch cần phải xử lý.....	17
Hình 13: Ảnh sau xử lý xoay và nhị phân hóa	18
Hình 14: Danh sách cách hình ảnh sau khi phân mảnh.....	18
Hình 15: Import thư viện.....	20
Hình 16: Đọc hình ảnh và xám hóa hình ảnh	20
Hình 17: Ảnh gốc chụp cận, độ sáng không cao	21
Hình 18: Ảnh đặt yêu cầu ngay sau khi nhị phân hóa	22
Hình 19: Hình ảnh ở xa và có nhiều góc cạnh.....	22
Hình 20: Hàm giảm độ sáng thông qua hàm clip của thư viện numpy	23
Hình 21: Xám hóa và giảm độ sáng	23
Hình 22: Hàm giảm nhiễu, tăng độ tương phản và loại bỏ nhiễu	23
Hình 23: Vị trí biên số xe đã rõ hơn	24
Hình 24: Hàm nhị phân hóa ảnh.....	24

Hình 25: Hình ảnh sau nhị phân hóa	25
Hình 26: Không khoảng vùng được do nhiều cạnh rác	25
Hình 27: Hàm phát hiện cạnh Canny	26
Hình 28: Hàm phát hiện biên bằng Dilation kết hợp phân tích hình thái học	26
Hình 29: Ảnh sau phân tích cạnh và phát hiện biên	26
Hình 30: Hàm Sobel hóa hình ảnh	27
Hình 31: Hình ảnh sau Sobel hóa	27
Hình 32: Gọi hàm tìm Contours, sắp xếp các contour theo thứ tự	31
Hình 33: Vẽ Approx lên hai hình ảnh đã tạo	32
Hình 34: Contours vẽ vào đúng các đường viền	32
Hình 35: Code phân tích để lọc ra contour biên số	33
Hình 36: Sau xử lý thì chỉ còn duy nhất một contours tại đúng vị trí của biển số xe ...	34
Hình 37: Hàm tìm góc nghiêng của biển số xe	35
Hình 38: Code cắt biển số xe và lưu ra file	35
Hình 39: Ảnh gốc	36
Hình 40: Ảnh sau khi bóc tách biển số	37
Hình 41: Hình ảnh sau quá trình tiền xử lý	38
Hình 42: Tạo Kernel hình chữ nhật và tìm contours	38
Hình 43: Phân tích contours	38
Hình 44: Vẽ hình chữ nhật bao quanh ký tự và cắt ký tự	39
Hình 45: Khoanh vùng các ký tự	39
Hình 46: Import các thư viện cần dùng	40
Hình 47: Đoạn code đọc dữ liệu từ folder train và test	41
Hình 48: Khởi tạo mô hình mạng neural tích chập CNN	41
Hình 49: Mô hình tổng quan	43

Hình 50: Epoch cao nhất đạt được	45
Hình 51: Hàm vẽ biểu đồ đánh giá mô hình huấn luyện	45
Hình 52: Kết quả huấn luyện mô hình.....	46
Hình 53: Import thư viện và load model, lấy mảng ký tự đã tách từ giai đoạn trên	47
Hình 54: Hàm sử dụng để chuyển đổi ảnh và hiển thị kết quả dự đoán.....	48
Hình 55: Kết quả của mô hình.....	49

LỜI MỞ ĐẦU

Trí tuệ nhân tạo (AI) là một chuyên ngành của khoa học máy tính: ứng dụng máy móc, thiết bị vào các công việc đòi hỏi trí tuệ của con người. Máy móc, thiết bị được phát triển để có năng lực như con người và năng lực đó có thể được kiểm chứng. Trí tuệ nhân tạo được bắt đầu nghiên cứu từ những năm 1950 của thế kỷ XX nhưng mới được ứng dụng thực tế rộng rãi trong vòng 10 năm trở lại đây. Trong cách mạng công nghiệp 4.0, AI đóng vai trò quan trọng làm nền tảng để tăng hiệu suất và giảm thiểu chi phí nhân công trong quá trình hoạt động của nhiều tổ chức doanh nghiệp trên thế giới. Trong đó có thể kể đến những lĩnh vực như y tế, kinh doanh, giao thông, giáo dục,... là những lĩnh vực đã được hưởng lợi từ cuộc cách mạng 4.0 với AI là yếu tố mang ý nghĩa chủ chốt. Đặc biệt, trong lĩnh vực giao thông, việc áp dụng AI (thị giác máy tính , máy học) đã góp phần thay đổi cũng như định hình tương lai của ngành giao thông.

Thực tế hiện nay, không chỉ trên thế giới, mà tại Việt Nam, số lượng xe cộ tham gia giao thông trên đường là rất lớn dẫn đến tiêu tốn rất nhiều nhân lực và vật lực cho việc quản lý phương tiện cá nhân trong bãi gửi xe. Nếu không có một công cụ thuận tiện thì việc quản lý phương tiện cá nhân rất mất thời gian, dễ gây nhầm lẫn, thiệt hại cho người sử dụng dịch vụ tại các bãi đỗ xe.

Để giảm tải cho các công việc, nhân lực trong 1 số quy trình nhưng xác định và quản lý trong bãi đỗ xe, trên thế giới đã phát triển công nghệ giám sát tự động đối với các phương tiện giao thông, chính nhờ tính cá nhân của biển số xe mà nó đã trở thành đối tượng chính được sử dụng để nghiên cứu, phát triển trong công nghệ này.

Bởi vậy nhóm 16 muốn chọn đề tài “ Ứng dụng của trí tuệ nhân tạo trong bài toán nhận diện biển số xe tại học viện ngân hàng ” như bước căn bản trong việc tìm hiểu các công cụ giám sát mạnh hơn như kiểm soát xe lưu thông trên đường hay nhận dạng khuôn mặt ... đang được thế giới rất chú trọng lúc này.

CHƯƠNG 1: GIỚI THIỆU BÀI TOÁN

1.1. GIỚI THIỆU BÀI TOÁN

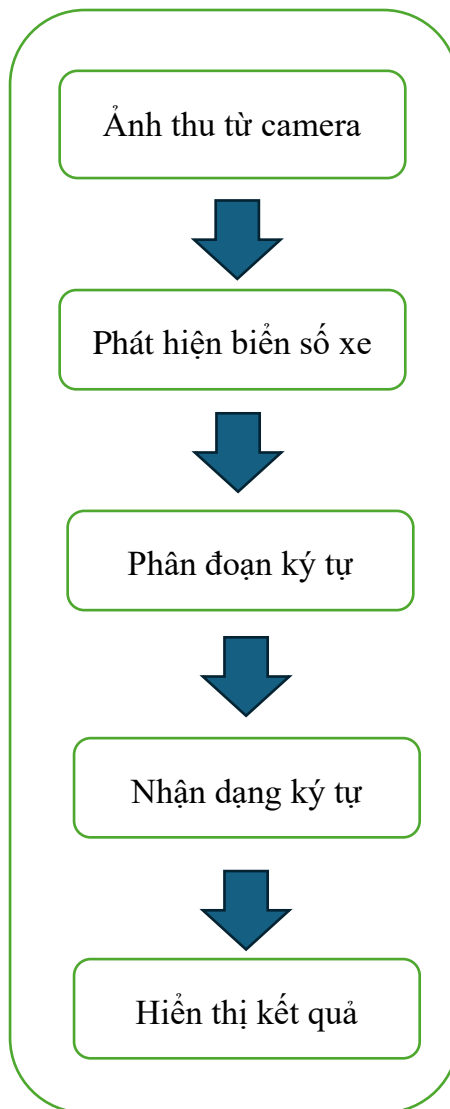
Hiện nay tại 1 số địa điểm gửi xe lớn tại 1 số cơ sở kinh doanh hay trường đại học trên địa bàn Hà Nội, đã và đang ứng dụng AI vào việc quản lý và quản lý xe vào và ra trong phạm vi quản lý của tổ chức, doanh nghiệp. Cụ thể, đó là việc ứng dụng nhận diện biển số xe thông qua hình ảnh được cắt từ các thiết bị camera có sẵn từ đó lưu trữ thông tin về thời điểm ra, vào của xe cộ. Dựa trên cơ sở đó, ứng dụng nhằm phục vụ trong 1 số trường hợp cụ thể như: xác định chính xác số lượng và thông tin của từng xe trong bãi đỗ trong thời điểm hiện tại nhằm đưa ra quyết định có tiếp tục nhận xe hay không. Ngoài ra, đây cũng là biện pháp ứng phó cho 1 số tình huống đặc biệt ví dụ như mất xe,... việc xác định được chính xác xe bị lấy ra tại thời điểm nào sẽ giúp ích cho việc giảm thời gian tìm kiếm thông tin đối tượng khả nghi, thay vì việc phải ngồi xem xét, tìm kiếm trong từng đoạn video dài.

Tại Học viện Ngân hàng, theo như quan sát và tìm hiểu của nhóm, hệ thống camera tại khu vực gửi xe mới chỉ dừng ở việc ghi lại video theo thời gian thực chứ chưa ứng dụng công nghệ vào khâu phân tích biển số để lưu trữ thông tin. Điều này, sẽ gây khó khăn trong việc kiểm soát lưu lượng xe trong bãi gửi gây việc ùn tắc trong các bãi đỗ do vượt quá sức chứa của bãi. Ngoài ra, việc truy xuất thời điểm xe ra vào phục vụ nghiệp vụ tìm kiếm xe mất sẽ mất nhiều thời gian.

1.2. ĐỀ XUẤT GIẢI QUYẾT BÀI TOÁN

Để giải quyết các vấn đề đã nêu trong mục 1.1, nhóm 16 đề xuất giải pháp ứng dụng, tích hợp AI vào trong hệ thống kiểm soát xe cộ ở trường, cụ thể là công cụ phân tích nhận diện biển số dựa trên các hình ảnh được cắt ra từ camera được gắn ở mỗi cổng. Dựa trên việc nhận diện biển số, hệ thống có thể lưu trữ thông tin như thời điểm vào và thời điểm ra của thiết bị sau đó tính toán được lưu lượng xe hiện tại trong trường và đưa ra sự điều chỉnh dòng xe vào ra hợp lý.

1.2.1. Sơ đồ tổng quát bài toán



Hình 1. Sơ đồ tổng quát bài toán

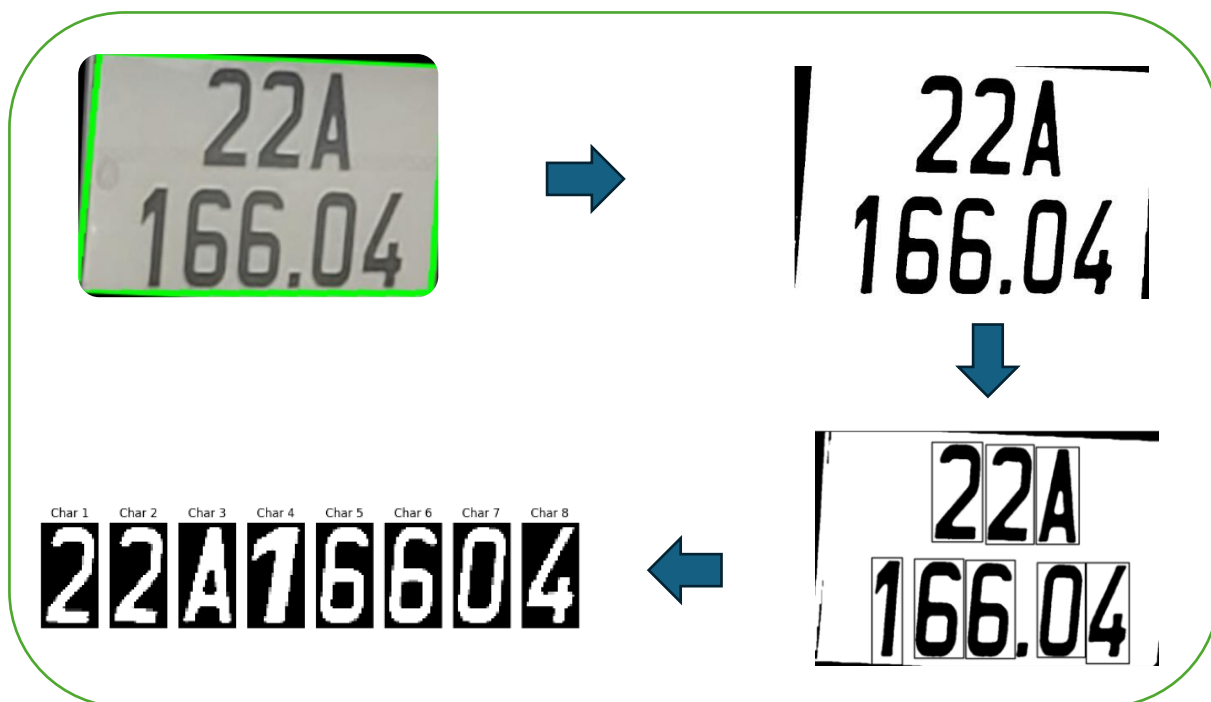
1.2.2. Tổng quan các bước thực hiện

- ❖ **Bước 1: Phát hiện biển số:** ta sẽ tách biển số ra khỏi hình ảnh ban đầu bằng cách tìm khung chữ nhật gồm (x, y, width, height) bao lấy biển số trong ảnh. Nếu tìm thấy thì cắt biển số ra để xử lý bước tiếp theo, kết quả bước này là biển số đã cắt.



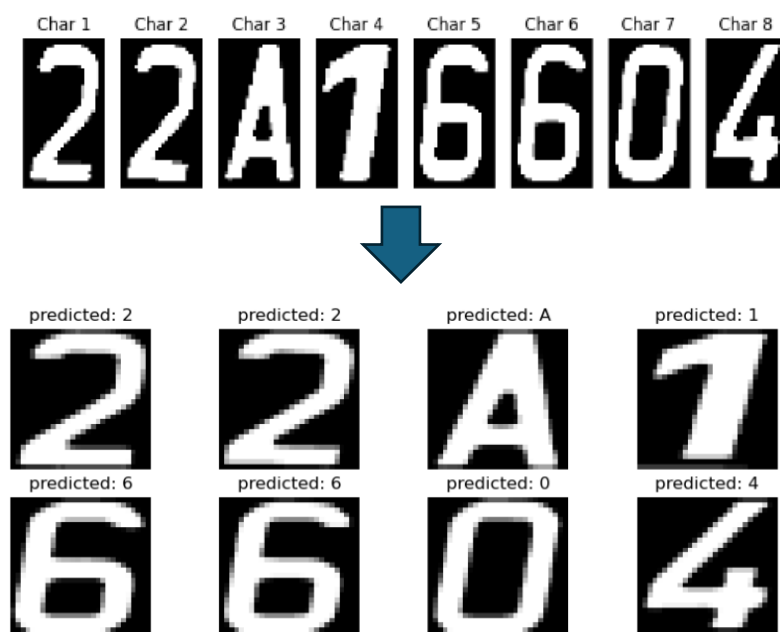
Hình 2. Bước tách biển số xe từ ảnh chụp

- ❖ **Bước 2: Phân đoạn ký tự:** Ta sẽ tách từng ký tự riêng lẻ từ biển số xe đã được phát hiện bằng cách tiền xử lý ảnh biển số xe đã cắt sau đó tìm tọa độ của các ký rồi vẽ các box hình chữ nhật bao quanh ký tự, chuyển các ký tự về dạng ảnh mà mô hình có thể nhận biết.



Hình 3. Bước phân đoạn ký tự từ biển số đã cắt

- ❖ **Bước 3: Nhận diện ký tự** bằng thuật toán Deep Learning, trong trường hợp này nhóm sử dụng mô hình mạng nơ-ron tích chập



Hình 4. Bước dự đoán các ký tự biển số

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 tập, 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 hàng 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

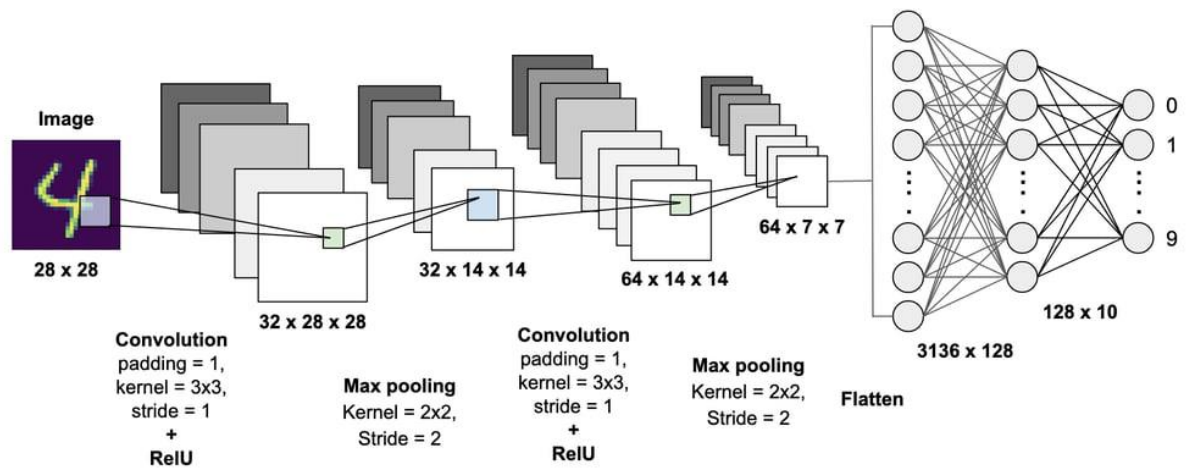
Vì đã cài sẵn TensorFlow.Keras nên nhóm sẽ dùng luôn thư viện này để đánh giá kết quả bằng lossFunction

2.2.2. Thuật toán *Convolutional Neural Network*

2.2.2.1. Định nghĩa CNN

CNN được viết tắt của Convolutional Neural Network là một trong những mô hình Deep Learning cực kỳ tiên tiến cho phép xây dựng những hệ thống có độ chính xác cao và thông minh. Nhờ khả năng đó, CNN có rất nhiều ứng dụng, đặc biệt là những bài toán cần nhận dạng vật thể (object) trong ảnh.

Trong đó, Convolutional là một loại cửa sổ dạng trượt nằm trên một ma trận. Các convolutional layer sẽ chứa các parameter có khả năng tự học, qua đó sẽ điều chỉnh và tìm ra cách lấy những thông tin chính xác nhất trong khi không cần chọn feature. Lúc này, convolution hay tích chập đóng vai trò là nhân các phần tử thuộc ma trận. Sliding Window, hay được gọi là kernel, filter hoặc feature detect, là loại ma trận có kích thước nhỏ.



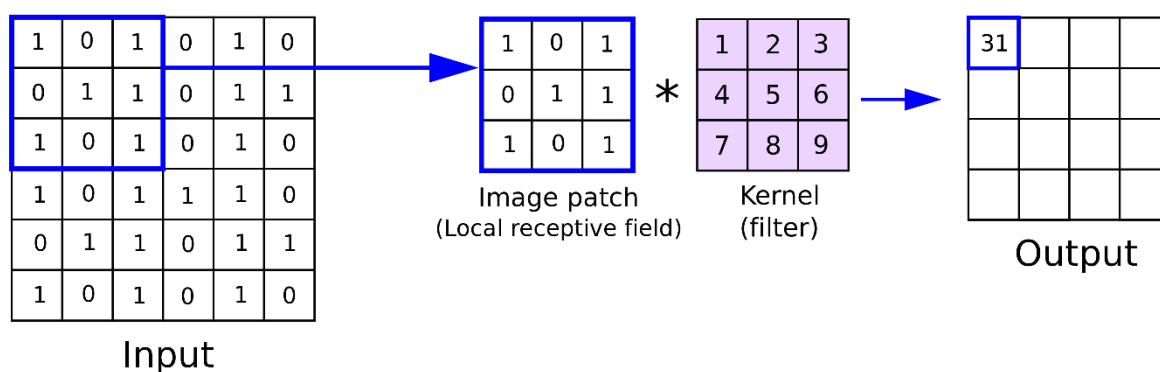
Hình 5. Mô hình mạng CNN

2.2.2.2. Các lớp cơ bản của mạng CNN

Convolutional layer

Lớp này là phần quan trọng nhất của toàn mạng CNN, nó có nhiệm vụ thực thi các tính toán. Các yếu tố quan trọng trong lớp Convolutional là: padding, stride, feature map và filter map.

- Sử dụng filter để áp dụng vào các vùng của ma trận hình ảnh. Các filter map là các ma trận 3 chiều, bên trong đó là những tham số và chúng được gọi là parameters.
- Stride tức là bạn dịch chuyển filter map theo từng pixel dựa vào các giá trị từ trái qua phải.
- Padding: Thường, giá trị viền xung quanh của ma trận hình ảnh sẽ được gán các giá trị 0 để có thể tiến hành nhân tích chập mà không làm giảm kích thước ma trận ảnh ban đầu.
- Feature map: Biểu diễn kết quả sau mỗi lần feature map quét qua ma trận ảnh đầu vào. Sau mỗi lần quét thì lớp Convolutional sẽ tiến hành tính toán.



Hình 6. Convolutional Layer

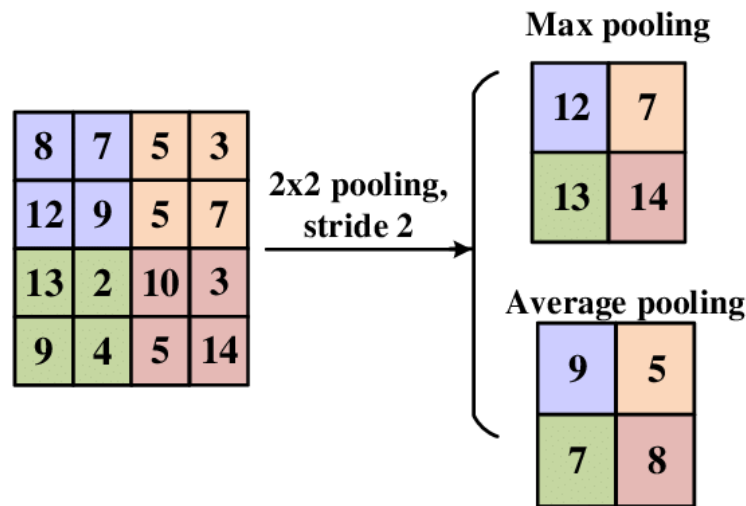
Relu Layer

Lớp ReLU này là hàm kích hoạt trong mạng CNN, được gọi là activation function. Nó có tác dụng mô phỏng những nơ ron có tỷ lệ truyền xung qua axon. Các hàm activation khác như Leaky, Sigmoid, Leaky, Maxout,.. tuy nhiên hiện nay, hàm ReLU được sử dụng phổ biến và thông dụng nhất.

Hàm này được sử dụng cho những yêu cầu huấn luyện mạng nơ ron với những ưu điểm nổi bật điển hình là hỗ trợ tính toán nhanh hơn. Trong quá trình dùng hàm ReLU, bạn cần chú ý đến việc tùy chỉnh những learning rate và dead unit. Những lớp ReLU được dùng sau khi filter map được tính và áp dụng ReLU lên các giá trị của filter map.

Pooling layer

Khi mà trận ảnh đầu vào có kích thước quá lớn, các lớp Pooling layer sẽ được đặt vào giữa những lớp Convolutional để làm giảm những parameters. Hiện, hai loại lớp Pooling được sử dụng phổ biến là Max pooling và Average.



Hình 7. Pooling Layer

Fully connected layer

Đây là lớp có nhiệm vụ đưa ra kết quả sau khi hai lớp Convolutional và Pooling đã nhận được ảnh truyền. Khi này, ta sẽ thu được một model đọc được thông tin của ảnh. Để có thể liên kế chúng cũng như cho nhiều đầu ra hơn ta sẽ sử dụng Fully connected layer. Ngoài ra, nếu lớp này có dữ liệu hình ảnh thì lớp sẽ chuyển chúng thành các mục chưa được phân chia chất lượng để tìm ra ảnh có chất lượng cao nhất.

2.2.2.3. Cấu trúc của mạng CNN

Mạng CNN là tập hợp những Convolutional layer xếp chồng lên nhau, đồng thời mạng sử dụng những hàm như ReLU và Tanh để kích hoạt các trọng số trong các node. Các lớp này sau khi qua các hàm activation sẽ có trọng số trong những node và có thể tạo ra những thông tin trừu tượng hơn đến với các lớp kế tiếp trong mạng.

Mạng này có tính kết hợp cả tính bất biến. Tức là, nếu cùng một đối tượng mà sử dụng chiếu theo các góc độ khác nhau thì sẽ có ảnh hưởng đến độ chính xác. Với dịch chuyển, co giãn hay quay ma trận ảnh thì lớp Pooling sẽ được dùng để hỗ trợ làm bất biến các tính chất này. Chính vì vậy mà mạng nơ ron này sẽ đưa ra những kết quả có độ chính xác tương ứng với từng mô hình.

Trong đó, lớp Pooling sẽ có khả năng tạo tính bất biến với phép dịch chuyển, co giãn và quay. Còn tính kết hợp cục bộ sẽ cho thấy những cấp độ biểu diễn, dữ liệu từ thấp đến cao với mức trừu tượng thông qua Convolution từ filter. Mạng CNN có những lớp liên kết nhau dựa vào cơ chế Convolution.

Các lớp tiếp theo sẽ là kết quả từ những lớp trước đó, vì vậy mà bạn sẽ có những liên kết cục bộ phù hợp nhất. Trong quá trình huấn luyện mạng, mạng nơ ron này sẽ tự học hỏi những giá trị thông qua filter layer dựa theo cách thức mà bạn thực hiện.

Cấu trúc cơ bản của một mô hình mạng CNN thường bao gồm 3 phần chính bao gồm:

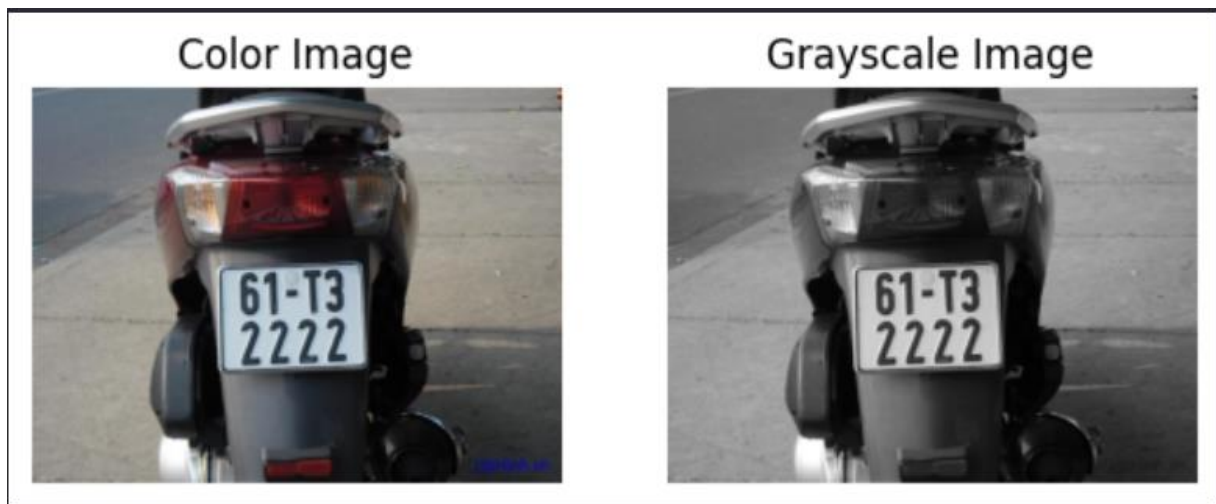
- Trường cục bộ/ Local receptive field: Lớp này sử dụng để tách lọc dữ liệu, thông tin hình ảnh để từ đó có thể lựa chọn các vùng có giá trị sử dụng hiệu quả cao nhất.
- Trọng số chia sẻ/ Shared weights and bias: Lớp này hỗ trợ làm giảm các tham số đến mức tối thiểu trong mạng CNN. Trong từng lớp convolution sẽ chứa các feature map riêng và từng feature thì sẽ có khả năng phát hiện một vài feature trong hình ảnh.
- Lớp tổng hợp/ Pooling layer: Đây là lớp cuối cùng và sử dụng để làm đơn giản các thông tin output. Tức là, sau khi tính toán xong và quét qua các layer trong mạng thì pooling layer sẽ được dùng để lược bỏ các thông tin không hữu ích. Từ đó cho ra kết quả theo kỳ vọng người dùng.

2.2.3. Các kỹ thuật cần sử dụng

2.2.3.1. Giai đoạn 1 – tìm biến số xe và cắt ảnh biến số xe

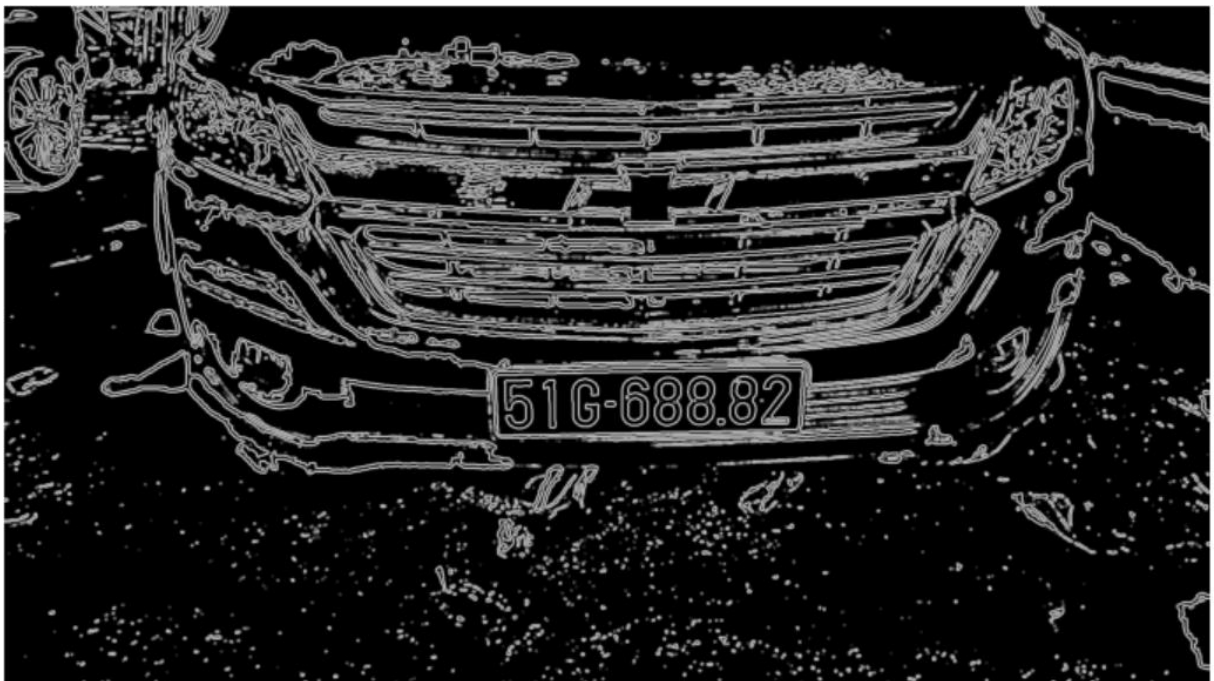
Trong giai đoạn này, các kỹ thuật quan trọng gồm có các kỹ thuật sau:

- ❖ 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 8: 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:

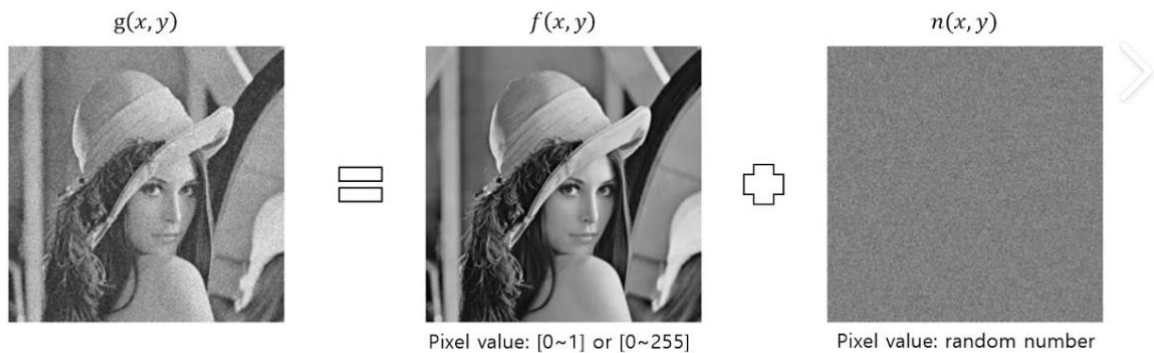


Hình 9: Ảnh sau quá trình tiền xử lý

- 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.

Image Degradation by Gaussian Noise

$$g(x, y) = f(x, y) + n(x, y)$$



Hình 10: 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, hiệu 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 Detection: 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 biên 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 11: 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 biên 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 12: Ảnh gốc bị lệch cần phải xử lý



Hình 13: 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 14: Danh sách cách hình ảnh sau khi phân mảnh

2.2.3.2. 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 tạc 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 CHỤP

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 15: 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 16: 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 17: Ảnh gốc chụp cận, độ sáng không cao

ảnh nhị phân hóa



Hình 18: 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 19: 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 20: Hàm giảm độ sáng thông qua hàm clip của thư viện numpy



Hình 21: 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.

```
# Làm mịn và giảm nhiễu bằng GaussianBlur
def ReduceNoiseAndSmooth(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 22: Hàm giảm nhiễu, tăng độ tương phản và loại bỏ nhiễu



Hình 23: Vị trí biển số xe đã rõ hơn

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

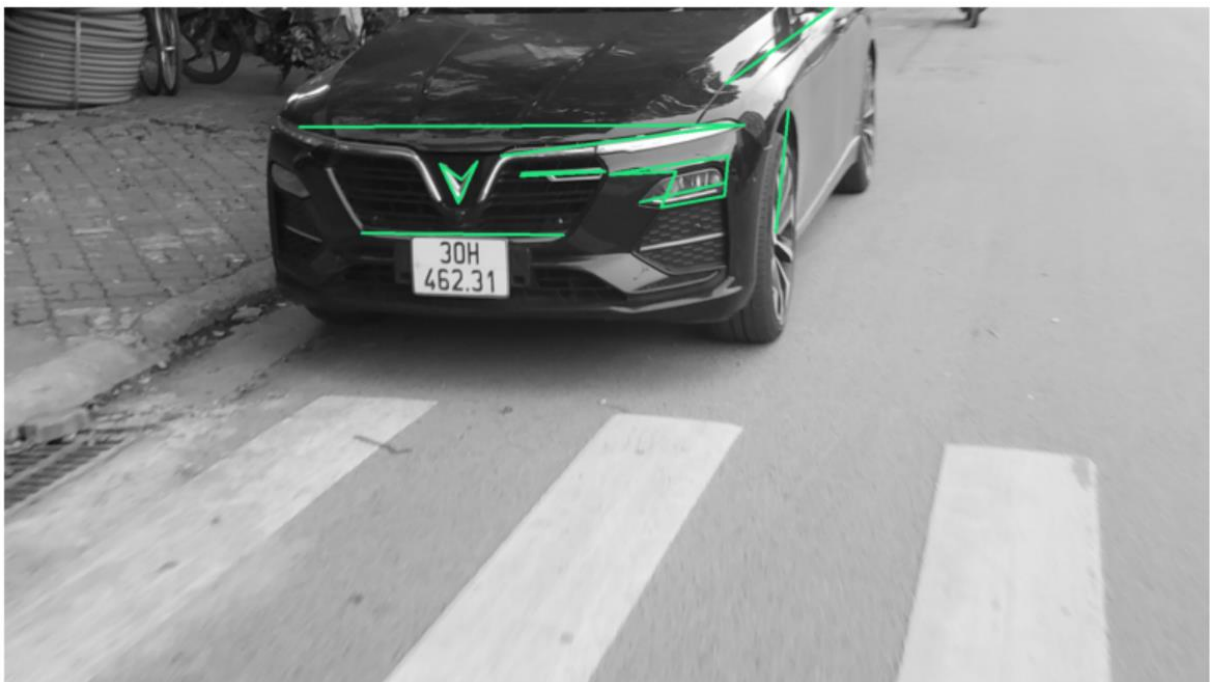
```
# nhị phân hóa hình ảnh
def BinarizeImg(imgGray):
    ADAPTIVE_THRESH_BLOCK_SIZE = 19
    ADAPTIVE_THRESH_WEIGHT = 9
    imgThresh = cv2.adaptiveThreshold(imgGray, 255.0, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY_INV, ADAPTIVE
    return imgThresh
```

Hình 24: Hàm nhị phân hóa ảnh



Hình 25: 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 26: Không khoảng 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 27: 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 28: Hàm phát hiện biên bằng Dilation kết hợp phân tích hình thái học



Hình 29: 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 màu trắng dẫn đến chúng tương phản rất mạnh. Chúng ta sẽ lọc biển số xe dựa trên tính chất này.

```
# 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 30: Hàm Sobel hóa hình ảnh



Hình 31: Hình ảnh sau Sobel hóa

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

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 liên 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
 - ContourIdx: 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 ellipse 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 sắp 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 32: Gọi hàm tìm Contours, sắp 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 sắp xếp contours là loại bỏ các điểm thừa 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 sắp 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 $0.06 \times$ 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)

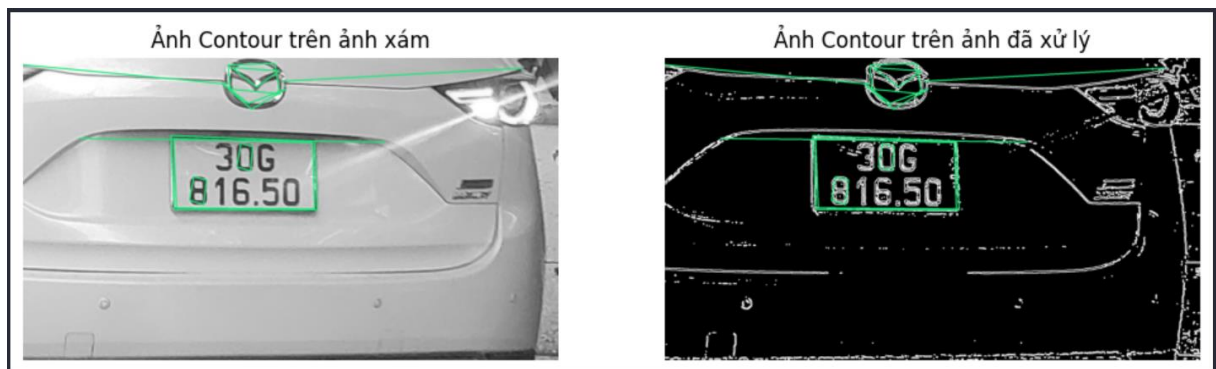
    # Xấp xỉ contour thành đa giác
    approx = cv2.approxPolyDP(contour, 0.06 * peri, True)

    # 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 33: Vẽ Approx lên hai hình ảnh đã tạo



Hình 34: 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 35: 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 36: 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 37: 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 số
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 38: 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 ngưỡng¹ 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 chụp ta sẽ chuyển sang bước tiếp theo.



Hình 39: Ảnh gốc

¹ Còn được gọi là ảnh nhị phân hóa



Hình 40: 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 41: 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:

```
kere13 = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3)) # Tạo kernel hình chữ nhật
thre_mor = cv2.morphologyEx(imgThresh, cv2.MORPH_DILATE, kere13) # Áp dụng phép giãn ảnh
cont, hier = cv2.findContours(thre_mor, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE) # Tìm các đường viền trong ảnh đã dãn
```

Hình 42: 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(cnt[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 43: 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 44: 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 cắt 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 45: 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 46: 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/train', # 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')
```

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

+ Code + Markdown

Python

Hình 47: Đoạn code đọc 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, (2,2), activation='relu', padding='same'))
model.add(Conv2D(32, (2,2), activation='relu', padding='same'))
model.add(Conv2D(64, (2,2), activation='relu', padding='same'))
model.add(Conv2D(64, (2,2), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
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 48: 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.summary()`

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 49: 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 50: 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

```
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 51: 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`.

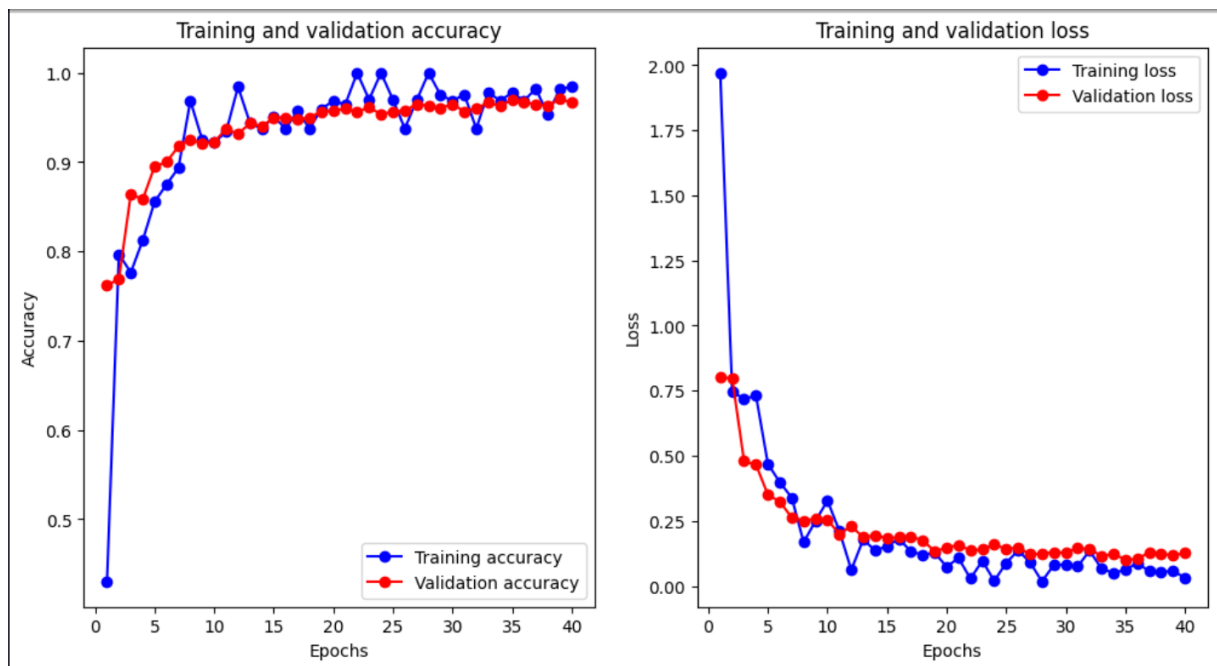
Lấy danh sách các giá trị độ chính xác (accuracy) trên tập validation qua các

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 52: 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 53: Import thư viện và load model, lấy mảng ký tự đã tách từ giai đoạn trên

```

# 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

```

Hình 54: Hàm sử dụng để chuyển đổi ảnh và hiển thị kết quả dự đoán

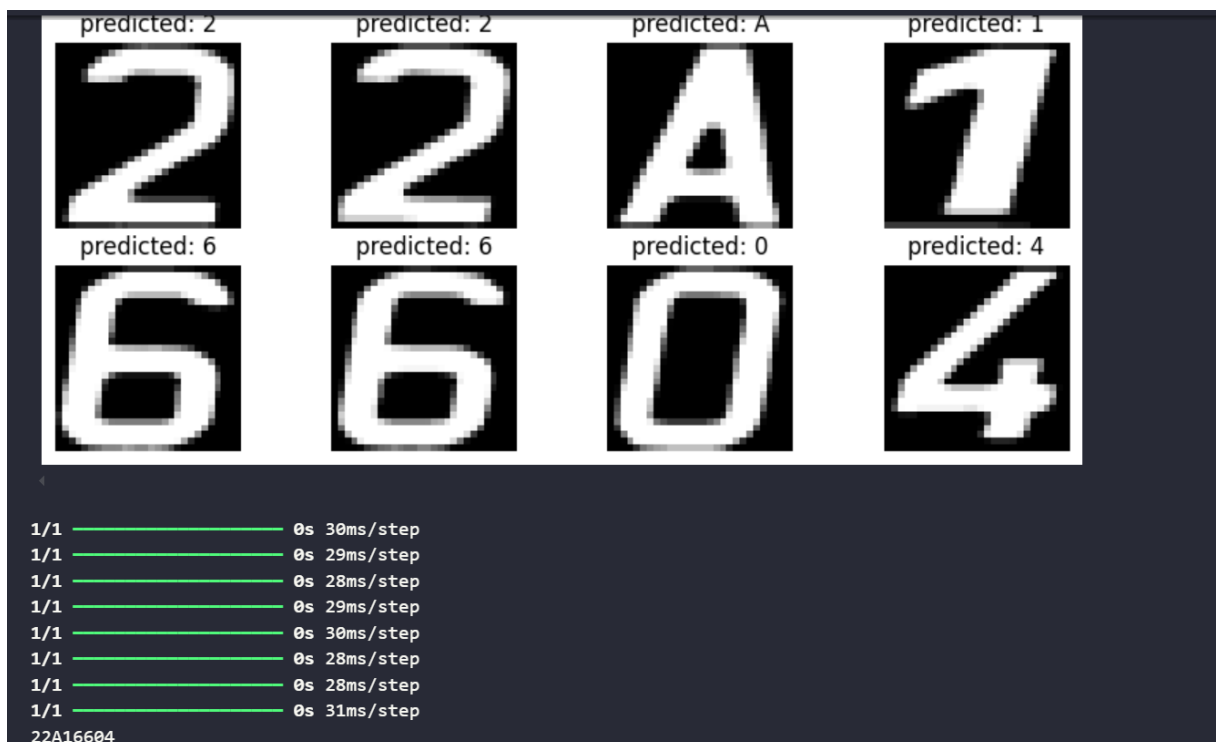
Vì các hình ảnh sau khi bóc tách từ mảng không phải là 28*28 như khi ta huấn luyện nên chúng ta sẽ cần có một hàm để chuyển đổi hình ảnh truyền vào thành hình ảnh 28*28 (hàm Fixdimension).

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 55: 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>

Myad S. E. O. (2022, December 9). Thuật toán CNN là gì? Thông tin cấu trúc của mạng CNN. *FPT Aptech*. <https://aptech.fpt.edu.vn/cnn-la-gi.html>