

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN THỊ GIÁC MÁY TÍNH**

**Nguyễn Huỳnh Bảo Tín
Lâm Văn Tới**

**NHẬN DẠNG ĐỊA CHỈ BIỂN HIỆU ÁP DỤNG XỬ
LÝ ẢNH VÀ HỌC SÂU**

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN CNTT

TP. HCM, <NĂM 2020>

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN THỊ GIÁC MÁY TÍNH**

**Nguyễn Huỳnh Bảo Tín – 1653089
Lâm Văn Tới - 1653093**

**NHẬN DẠNG ĐỊA CHỈ BIỂN HIỆU ÁP DỤNG XỬ LÝ
ẢNH VÀ HỌC SÂU**

KHÓA LUẬN TỐT NGHIỆP CỬ NHÂN CNTT

GIÁO VIÊN HƯỚNG DẪN

GS.TS Lê Hoài Bắc

KHÓA 2016 - 2020

LỜI CẢM ƠN

Đầu tiên chúng em xin được gửi lời cảm ơn chân thành đến **GS.TS Lê Hoài Bắc** – Người thầy trực tiếp hướng dẫn và chỉnh sửa cho chúng em trong suốt quá trình thực hiện khóa luận. Nhờ vào kinh nghiệm và khả năng giải quyết các vấn đề của thầy, chúng em đã đi đúng hướng và hoàn thành được khóa luận này đúng hạn và tương đối thành công. Bên cạnh đó, thầy còn truyền đạt thêm cho chúng em nhiều kiến thức thực tiễn, niềm đam mê với bộ môn Thị giác máy tính, giải quyết cho chúng em được rất nhiều khó khăn mắc phải trong khi lập trình hay nghiên cứu.

Chúng em cũng xin gửi lời cảm ơn trân quý đến chị Hồ Trần Nhật Thủy – supervisor của chúng em tại công ty giải pháp Vietbando Việt Nam và là một người chị đã từng bước giúp đỡ chúng em tiếp cận, học hỏi và rèn luyện được các kỹ năng cần thiết trong môi trường làm việc thực tế. Không những thế, chị Thủy còn trực tiếp giám sát từng bước thực hiện khóa luận của chúng em và đưa ra các lời khuyên bổ ích, cũng như hỗ trợ các điều kiện cần có trong suốt thời gian nghiên cứu và làm khóa luận này ở công ty.

Em xin chân thành cảm ơn!

TP.HCM, Ngày 17 Tháng 08 Năm 2020

Sinh viên thực hiện

Nguyễn Huỳnh Bảo Tín

Lâm Văn Tới

MỤC LỤC

LỜI CẢM ƠN	i
MỤC LỤC	ii
DANH SÁCH HÌNH VẼ	vi
DANH SÁCH BẢNG BIỂU	vii
DANH MỤC THUẬT NGỮ.....	viii
TÓM TẮT KHÓA LUẬN	ix
CHƯƠNG 1 MỞ ĐẦU.....	1
1.1 Lý do chọn đề tài	1
1.2 Mục đích.....	1
1.3 Đối tượng.....	1
1.4 Phạm vi nghiên cứu	2
CHƯƠNG 2 TỔNG QUAN.....	3
2.1 Các hướng nghiên cứu đã có.....	3
2.1.1 Trong nước	3
2.1.2 Ngoài nước	3
2.2 Những vấn đề còn tồn tại	4
2.3 Hướng giải quyết	4

2.4 Cấu trúc khóa luận.....	4
CHƯƠNG 3 CÁC KIẾN THỨC NỀN TẢNG.....	6
3.1 Nhận dạng kí tự quang học (OCR).....	6
3.1.1 Định nghĩa.....	6
3.1.2 Phân loại.....	6
3.1.3 Quy trình nhận dạng	6
3.1.3a Nhị phân hóa và cải tiến chất lượng hình ảnh	7
3.1.3b Phát hiện dòng văn bản.....	9
3.1.3c Phân vùng ảnh.....	11
3.1.3d Rút trích đặc trưng	12
3.2 Nhận dạng thực thể định danh (Named-Entity Recognition hay NER).....	16
3.3.1 Khái niệm.....	16
3.3.2 Vai trò	17
3.3.3 Các hướng tiếp cận.....	18
3.3 Chỉnh hướng văn bản (Text Skew Correction)	18
3.3.1 Khái niệm.....	18
3.3.2 Ứng dụng trong Scene Text.....	19
3.3.3 Các bước thực hiện.....	19
3.4 Khoảng cách Levenshtein	20
3.4.1 Khái niệm.....	20

3.4.2 Thuật toán.....	20
3.4.2a Quy hoạch động.....	21
3.4.2b Mã giải.....	21
CHƯƠNG 4 PHƯƠNG PHÁP ĐỀ XUẤT	23
4.1 Tìm kiếm các hướng tiếp cận và những vấn đề liên quan	23
4.2 Quá trình xây dựng mô hình và giải pháp	24
4.3 Giai đoạn 1: Thu thập và tiền xử lý dữ liệu	24
4.3.1 Thu thập dữ liệu	25
4.3.2 Tạo dữ liệu cho mô hình OCR	25
4.3.3 Tiền xử lý dữ liệu	28
4.3.3a Khi huấn luyện	28
4.3.3b Khi đánh giá	29
4.4 Giai đoạn 2: Xác định mô hình phù hợp.....	37
4.4.1 Lenet.....	37
4.4.2 SmallerVGGNet.....	38
4.4.2a Một vài khái niệm	38
4.4.2b SmallerVGGNet.....	40
4.5 Giai đoạn 3: Tùy chỉnh các tham số huấn luyện	42
4.6 Giai đoạn 4: Đánh giá chất lượng mô hình	43
CHƯƠNG 5 CÀI ĐẶT	44

5.1 Ngôn ngữ lập trình sử dụng	44
5.2 Bộ xử lý hình ảnh (Graphic Processing Unit hay GPU)	45
CHƯƠNG 6 THỰC NGHIỆM VÀ KẾT QUẢ.....	47
6.1 Các vấn đề gặp phải khi thực nghiệm	47
6.2 Kết quả	49
6.2.1 Huấn luyện mô hình OCR.....	49
6.2.2 Tesseract OCR + Mô hình OCR tự xây dựng.....	53
6.2.3 Mô hình phát hiện vùng địa chỉ	55
CHƯƠNG 7 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	57
7.1 Kết luận.....	57
7.2 Hướng phát triển	57
TÀI LIỆU THAM KHẢO	59

DANH SÁCH HÌNH VẼ

Hình 3.1	Quá trình thực hiện nhận dạng ký tự quang học	6
Hình 3.2	Sử dụng bộ lọc Gaussian để tạo hiệu ứng mờ	9
Hình 3.3	Ví dụ về phát hiện văn bản	10
Hình 3.4	Một ví dụ về phân vùng ảnh	11
Hình 3.5	Một ví dụ khi áp dụng phân vùng ảnh lên từng ký tự	11
Hình 3.6	Khoanh vùng theo mật độ	12
Hình 3.7	Khoanh vùng theo hướng	13
Hình 3.8	Biểu đồ khi chiếu ngang và dọc	13
Hình 3.9	Trắc diện tích khoảng cách của các cạnh với khung hình chữ nhật	14
Hình 3.10	Giao điểm và khoảng cách	14
Hình 3.11	Ví dụ về phương pháp cấu trúc	15
Hình 3.12	Các đặc tính của phương pháp cấu trúc	15
Hình 3.13	Ví dụ về nhận dạng thực thể định danh	16
Hình 3.14	Đoạn text trước khi xoay	18
Hình 3.15	Đoạn text sau khi xoay	18
Hình 3.16	Các Scene Text đã được cắt	19
Hình 3.17	Sau khi chỉnh góc nhìn	19
Hình 3.18	Mã giải thuật toán Levenshtein	22
Hình 3.19	Ví dụ áp dụng thuật toán trên ngôn ngữ Python	22
Hình 4.1	Chỉnh sửa các tham số trong phát hiện văn bản	24
Hình 4.2	Một số font chữ được sử dụng để tạo dữ liệu	25
Hình 4.3	Màu nền được sử dụng để tạo dữ liệu	26
Hình 4.4	Dữ liệu sau khi được tạo từ TextRecognitionDataGenerator	26
Hình 4.5	Dữ liệu huấn luyện	27
Hình 4.6	Các nhãn ký tự được huấn luyện	27
Hình 4.7	Một số ký tự nhận dạng sai từ mô hình huấn luyện được	28
Hình 4.8	Tiền xử lý dữ liệu cho tập huấn luyện	28
Hình 4.9	Thay đổi kích thước hình ảnh	30
Hình 4.10	Nới rộng vùng chứa văn bản	30
Hình 4.11	Khung chứa địa chỉ trước khi nới rộng	31
Hình 4.12	Khung chứa địa chỉ sau khi nới rộng	31
Hình 4.13	Ảnh ROI trước khi chuyển đổi	32
Hình 4.14	Ảnh ROI sau khi chuyển đổi	32
Hình 4.15	Ảnh ROI sau khi dùng bộ lọc song phương	32
Hình 4.16	Chuyển đổi gam màu và sử dụng bộ lọc song phương	33
Hình 4.17	Chuyển đổi ảnh nhị phân dạng 1	34
Hình 4.18	Ảnh nhị phân sau khi chuyển đổi dạng 1	34
Hình 4.19	Chuyển đổi ảnh nhị phân dạng 2	35
Hình 4.20	Ảnh nhị phân sau khi chuyển đổi dạng 2	35
Hình 4.21	Kết quả sau khi áp dụng CRAFT cho ảnh ROI	36
Hình 4.22	Ảnh một ký tự sau khi cắt ra từ ảnh ROI của CRAFT	36
Hình 4.23	Ảnh các ký tự chưa sắp xếp đúng vị trí ban đầu	36
Hình 4.24	Ảnh các ký tự sau khi sắp xếp	37
Hình 4.25	Cấu trúc mạng Lenet	38

Hình 4.26 Một phần cấu trúc mạng SmallerVGGNet	41
Hình 4.27 Dữ liệu sau khi được thay đổi tỉ lệ về 64x64 và 32x32	43
Hình 5.1 Ví dụ về các ưu điểm của Python	45
Hình 6.1 Ví dụ về địa chỉ viết liền với số điện thoại.....	47
Hình 6.2 Ví dụ về phần địa chỉ không nằm trên cùng một dòng	48
Hình 6.3 Cắt bỏ những phần không cần thiết cho địa chỉ	49
Hình 6.4 Kết quả huấn luyện mô hình OCR với SmallerVggNet.....	50
Hình 6.5 Kết quả huấn luyện mô hình OCR ở dạng biểu đồ	50
Hình 6.6 Ví dụ về việc tách ký tự sai của CRAFT	53
Hình 6.7 Quá trình thực hiện nhận dạng của hệ thống	54

DANH SÁCH BẢNG BIỂU

Bảng 1. Kết quả dự đoán trên 127 kí tự được cắt từ CRAFT với dữ liệu ảnh trên đường Nguyễn Thị Nhỏ.....	42
Bảng 2. Một số kí tự được dự đoán sai bởi LeNet nhưng đúng với SmallVggNet	42
Bảng 3. Kết quả và độ chính xác của một vài tấm ảnh áp dụng các mô hình OCR....	52
Bảng 4. Kết quả thực nghiệm các mô hình nhận dạng trên 600 tấm ảnh	55
Bảng 5. Kết quả thực nghiệm mô hình phát hiện vùng địa chỉ trên 600 tấm ảnh.....	56

DANH MỤC THUẬT NGỮ

Scene Text	Các kí tự có background, font chữ, góc nghiêng, ngữ cảnh khác nhau.
Population	Tổng thể.
one hot vector	Một dãy bit mang giá trị 0 hoặc 1. Giá trị đúng chính là index có giá trị 1, còn lại tất cả đều có giá trị 0.
ROI	Vùng quan tâm trong một hình ảnh

TÓM TẮT KHÓA LUẬN

Trong thời đại công nghệ 4.0. Việc cập nhật dữ liệu bằng thủ công gần như đã được thay thế bởi các phần mềm, ứng dụng hữu ích có sử dụng AI, giúp tiết kiệm được rất nhiều công sức và chi phí phải bỏ ra để thuê người làm. Nhìn nhận được sự tiện ích này, nhóm chúng em muốn tạo ra một ứng dụng giúp các doanh nghiệp rút trích được dữ liệu (cụ thể là địa chỉ trên biển hiệu) thông qua các hình ảnh, nhờ đó việc cập nhật dữ liệu cho hệ thống sẽ trở nên tối ưu hơn về thời gian cũng như chi phí. Cũng giống như những mô hình nhận dạng khác, các phương pháp chính để mô hình trở nên thông minh cần có bộ dữ liệu thích hợp dùng cho huấn luyện, đa dạng các loại font chữ, tiền xử lý ảnh, rút trích đặc trưng, tìm kiếm mô hình phù hợp và cuối cùng là tinh chỉnh các tham số sao cho việc huấn luyện trở nên chính xác nhất. Ngoài ra, mô hình còn sử dụng thêm API để lọc các khung hình chữ nhật (vùng chứa các kí tự) trả về xem có phải là khung chứa địa chỉ hay không, nhằm rút ngắn thời gian chạy chương trình cũng như không nhận dạng nhầm các khung hình chữ nhật không chứa địa chỉ thành địa chỉ.

CHƯƠNG 1

MỞ ĐẦU

1.1 Lý do chọn đề tài

Được biết đến sức mạnh của ứng dụng Google map là bộ dữ liệu phong phú một phần do người dùng tự cung cấp, và phần lớn nhờ vào các chuyên gia, nhà nghiên cứu làm việc cho Google đã đề ra được các giải pháp, ứng dụng về AI giúp công ty đạt được thành công như hôm nay. Một trong số đó là việc tự động cập nhật thông tin vị trí cho từng tọa độ cụ thể trên bản đồ. Nhưng ở Việt Nam, giải pháp này chưa được phát triển do nhu cầu không cao, nhiều công ty còn e dè sẽ không có được độ chính xác tốt. Nắm được hạn chế của Google Map là không cập nhật tốt ở từng địa phương cụ thể, chúng em chọn đề tài này nhằm khai thác mặt hạn chế đó nhờ vào Xử lý ảnh – Thị giác máy tính.

1.2 Mục đích

Như đã đề cập trước đó, mục đích chính của luận văn là tạo ra được một mô hình có thể cập nhật vị trí chính xác nhất ở từng địa phương cụ thể, những nơi khó nhận dạng địa chỉ nhất cũng như những nơi dễ nhận dạng, với việc làm đơn thuần chỉ là nhập vào một tấm ảnh có chứa địa chỉ biển hiệu hoặc số nhà. Chúng em sẽ ưu tiên xây dựng nhận dạng trên mô hình chứ không dấu trước.

1.3 Đối tượng

Đối tượng chính ứng dụng nhắm đến là các doanh nghiệp hoặc cơ quan nhà nước có làm việc với dữ liệu về tình hình đất đai hay bản đồ ở Việt Nam.

1.4 Phạm vi nghiên cứu

Hiện tại các nghiên cứu về nhận dạng **Scene Text**[\[1\]](#) ở Việt Nam là không nhiều, nhất là **Scene Text** trên các biển hiệu. Phạm vi nghiên cứu sẽ là các con đường ở thành phố Hồ Chí Minh, nơi mỗi hộ dân cư có các biển hiệu hoặc số nhà được treo/dán trước nhà.

CHƯƠNG 2

TỔNG QUAN

2.1 Các hướng nghiên cứu đã có

2.1.1 Ngoài nước

Các công trình nghiên cứu về nhận dạng **Scene Text** đã được phát triển từ những năm cuối thế kỉ 20, khi các cuộc thi nổi tiếng như ICDAR (chủ đề chính là Nhận dạng kí tự được tổ chức ở nhiều quốc gia khác nhau mỗi hai năm) đã tạo một làn sóng lớn thúc đẩy các nhà nghiên cứu tạo ra các mô hình hiện đại như hiện nay. Vấn đề về ngôn ngữ, hình dạng của các kí tự khác Latin đã được giải quyết khi các nhà nghiên cứu tham gia cuộc thi đến từ nhiều quốc gia, châu lục khác nhau và cùng chung mong muốn là tạo ra được một mô hình có thể dự đoán được tất cả các ngôn ngữ trên thế giới.

2.1.2 Trong nước

Các nhà nghiên cứu về đề tài này trong nước cũng có chung một mục đích, nhưng không có nhiều sự thay đổi về hướng tiếp cận cũng như các thuật toán sử dụng, chủ yếu là thay đổi các tham số cũng như dữ liệu có liên quan đến bảng chữ cái Việt Nam. Các cuộc thi với quy mô nhỏ hơn thông qua những doanh nghiệp lớn với các nền tảng như Vin AI hay Zalo AI cũng đã được tổ chức ở Việt Nam như cuộc thi về nhận dạng chữ viết tay,... và đạt được thành công ngoài mong đợi. Điều đó nói lên được các nhà nghiên cứu trong nước đang ngày một phát triển lĩnh vực này không kém gì các quốc gia tiên tiến khác.

2.2 Những vấn đề còn tồn tại

Do sự đa dạng về font chữ, màu sắc, góc nghiêng, vị trí,... của **Scene Text** nên việc tạo ra một mô hình có độ chính xác tuyệt đối là cực kì khó, gần như là không thể.

Ngoài ra, vấn đề về sự tương phản giữa màu nền và màu chữ cũng còn đang nan giải khi không phải biển hiệu nào cũng phân biệt sáng tối (nền gam màu sáng và chữ gam màu tối hoặc ngược lại).

Cách thiết kế vị trí của địa chỉ trên biển hiệu cũng rất đa dạng. Có những địa chỉ không có tên phường, quận, thành phố, những địa chỉ số nhà và tên đường không nằm trên cùng một hàng, và những địa chỉ viết tắt tên đường, phường, quận,...

Nhiều từ yếu tố môi trường như trời tối, quá sáng hay cây che khuất vùng có địa chỉ, hoặc thậm chí nhiều từ chính biển hiệu (những đường gạch dọc, ngang, chéo) là rất nhiều.

2.3 Hướng giải quyết

Sau khi xem xét những vấn đề nêu trên, chúng em nhận thấy được **population** cần có là những biển hiệu ít nhiều, vùng địa chỉ phải nằm trên cùng một hàng, nền và chữ cần được phân biệt sáng tối và nên đưa về một kênh màu (greyscale) để giải quyết vấn đề đa dạng về màu sắc.

2.4 Cấu trúc khóa luận

Chương 1: Phần mở đầu. Lý do chọn đề tài, mục đích, đối tượng hướng đến và phạm vi nghiên cứu.

Chương 2: Giới thiệu tổng quan về các hướng nghiên cứu đã có trong nước và ngoài nước, những vấn đề còn tồn tại và hướng giải quyết đi kèm.

Chương 3: Các kiến thức nền tảng. Trình bày các kiến thức nền tảng cần được nắm rõ trước khi thực hiện khóa luận này.

Chương 4: Phương pháp đề xuất. Quá trình xây dựng mô hình và giải pháp.

Chương 5: Cài đặt. Cài đặt GPU và ngôn ngữ lập trình chính sử dụng.

Chương 6: Thực nghiệm và kết quả. Trình bày các vấn đề mắc phải khi thực nghiệm và kết quả đạt được.

Chương 7: Kết luận và hướng phát triển.

CHƯƠNG 3

CÁC KIẾN THỨC NỀN TẢNG

3.1 Nhận dạng kí tự quang học (OCR)

3.1.1 Định nghĩa

Nhận dạng kí tự quang học là quá trình rút trích các đặc trưng của từng điểm ảnh trong một khung hình chữ nhật chứa các kí tự cần nhận dạng. Sau đó thông qua các lớp trong mô hình học sâu được huấn luyện và trả về dự đoán của từng kí tự ứng với từng vị trí trong đoạn văn bản.

Trong nhận dạng kí tự, các văn bản được coi là đã được phát hiện, định vị và các khung hình chữ nhật chứa văn bản có sẵn.



Hình 3.1 Quá trình thực hiện nhận dạng kí tự quang học

3.1.2 Phân loại

Các phương pháp có sẵn để thực hiện nhận dạng kí tự có thể được phân loại thành các cách tiếp cận từ trên xuống và từ dưới lên.

Trong các cách tiếp cận từ trên xuống, một tập hợp các từ trong từ điển được sử dụng để xác định từ nào phù hợp với hình ảnh đã cho. Hình ảnh hầu như không được phân đoạn trong các phương pháp này. Do đó, cách tiếp cận từ trên xuống đôi khi được gọi là nhận dạng phân đoạn tự do.

Trong các cách tiếp cận từ dưới lên, hình ảnh được phân thành nhiều thành phần và hình ảnh được phân đoạn được truyền qua một công cụ nhận dạng. Hoặc là một bộ nhận dạng ký tự quang học (OCR) hoặc một loại được huấn luyện tùy chỉnh được sử dụng để nhận dạng văn bản.

3.1.3 Quy trình nhận dạng

3.1.3a Nhị phân hóa và cải tiến chất lượng hình ảnh

Nhị phân hóa hình ảnh là một bước cần thiết đầu tiên của hầu hết các hệ thống phân tích hình ảnh tài liệu hay các hình ảnh thực tế chứa ký tự.

Trong bộ môn Xử lý ảnh hay Thị giác máy tính, phương pháp Otsu là một thuật toán được áp dụng rộng rãi cho việc nhị phân hóa. Cụ thể, thuật toán này thực hiện lấy ngưỡng hình ảnh tự động ở dạng đơn giản nhất, trả về một ngưỡng cường độ duy nhất phân tách các điểm ảnh thành hai lớp, nền trước là nền sau.

Thuật toán này thực hiện tìm kiếm một ngưỡng nhất định với mục đích làm giảm thiểu phương sai cục bộ, được định nghĩa là tổng phương sai có trọng số của hai lớp:

$$\sigma_w^2(t) = \omega_0(t)\sigma_0^2(t) + \omega_1(t)\sigma_1^2(t)$$

Trong đó trọng số ω_0 và ω_1 là xác suất của 2 lớp cách nhau một ngưỡng t và σ_0^2 và σ_1^2 là phương sai của hai lớp này.

Xác suất các lớp $\omega_{0,1}(t)$ được tính toán từ biểu đồ:

$$\omega_0(t) = \sum_{i=0}^{t-1} p(i)$$

$$\omega_1(t) = \sum_{i=t}^{L-1} p(i)$$

Đối với hai lớp, việc giảm thiểu phương sai cục bộ tương đương với việc tối đa hóa phương sai giữa các lớp:

$$\begin{aligned}\sigma_b^2(t) &= \sigma^2 - \sigma_w^2(t) = \omega_0(\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2 \\ &= \omega_0(t)\omega_1(t)[\mu_0(t) - \mu_1(t)]^2\end{aligned}$$

Biểu thức được thể hiện dưới dạng lớp xác suất ω và lớp trung bình μ , với lớp trung bình $\mu_0(t)$, $\mu_1(t)$ và μ_T là:

$$\mu_0(t) = \frac{\sum_{i=0}^{t-1} ip(i)}{\omega_0(t)}$$

$$\mu_1(t) = \frac{\sum_{i=t}^{L-1} ip(i)}{\omega_1(t)}$$

$$\mu_T = \sum_{i=0}^{L-1} ip(i)$$

Qua đó có thể dễ dàng có các mối liên hệ:

$$\begin{aligned}\omega_0\mu_0 + \omega_1\mu_1 &= \mu_T \\ \omega_0 + \omega_1 &= 1\end{aligned}$$

Lớp xác suất và lớp trung bình có thể được tính toán theo một chu trình. Từ đó có thể thấy được ý tưởng này mang lại một thuật toán hiệu quả.

***Thuật toán**

1. Tính toán biểu đồ và xác suất từng mức cường độ ảnh
2. Thiết lập trọng số $\omega_i(0)$ và $\mu_i(0)$
3. Duyệt qua tất cả các ngưỡng $t = 1, \dots$ cường độ tối đa

- a. Cập nhật ω_i và μ_i

b. Tính $\sigma_b^2(t)$

4. Ngưỡng mong muốn ứng với giá trị $\sigma_b^2(t)$ lớn nhất

Đối với những tài liệu lịch sử, hình ảnh thực tế thông thường sẽ có chất lượng thấp, việc cải tiến chất lượng hình ảnh cũng rất cần thiết trong phân xử lý ảnh này. Cụ thể, ta có thể áp dụng những bộ lọc giảm nhiễu, làm mờ hay tinh chỉnh độ sáng tối.



Hình 3.2 Sử dụng bộ lọc Gaussian để tạo hiệu ứng mờ

3.1.3b Phát hiện dòng văn bản (Text Line Detection)



Hình 3.3 Ví dụ về phát hiện văn bản

***Định nghĩa**

Phát hiện dòng văn bản là quá trình phát hiện các đoạn văn bản có trong hình ảnh, trả về kết quả bao quanh nó với một khung hình chữ nhật.

***Phân loại**

Phát hiện văn bản có thể được thực hiện bằng cách sử dụng các kỹ thuật dựa trên hình ảnh hoặc kỹ thuật dựa trên tần số.

Trong kỹ thuật dựa trên hình ảnh, một hình ảnh được phân thành nhiều phân đoạn. Mỗi phân đoạn là một thành phần được kết nối của các điểm ảnh có đặc điểm tương tự nhau. Các tính năng thống kê của các thành phần kết nối được sử dụng để nhóm chúng và tạo thành văn bản. Các phương pháp học máy như SVM (Support Vector Machine) và mạng nơ-ron tích chập được sử dụng để phân loại các thành phần thành văn bản và phi văn bản.

Trong kỹ thuật dựa trên tần số, biến đổi Fourier rời rạc (DFT) hoặc biến đổi sóng rời rạc (DWT) được sử dụng để trích xuất các hệ số tần số cao. Giả định rằng văn bản hiện diện trong một hình ảnh có các thành phần tần số cao và chỉ chọn các hệ số tần số cao sau đó lọc văn bản từ các vùng không có văn bản trong một hình ảnh.

3.1.3c Phân vùng ảnh

Trong thị giác máy tính, phân vùng ảnh là một quá trình chia một bức ảnh số thành nhiều tập hợp điểm ảnh khác nhau. Mục tiêu của phân vùng ảnh là để làm đơn giản hóa và thay đổi biểu diễn của một tấm ảnh theo ý muốn để dễ dàng phân tích.

Phân vùng ảnh thường được sử dụng để xác định vị trí các đối tượng, đường biên,... Chính vì thế, các điểm ảnh trong cùng một nhãn đối tượng sẽ có những đặc tính giống nhau về màu sắc, cường độ hoặc kết cấu của ảnh.



Hình 3.4 Một ví dụ về phân vùng ảnh

Ở bước này, sau khi có được những đoạn văn bản đã được phát hiện trước đó, việc phân vùng ảnh sẽ được áp dụng. Những phần được phân vùng sẽ tương trưng cho những vùng văn bản cần rút trích trên ảnh, từ đó các các đoạn văn bản sẽ được phân mảnh thành từng kí tự riêng lẻ.



Hình 3.5 Một ví dụ khi áp dụng phân vùng ảnh lên từng kí tự

3.1.3d Rút trích đặc trưng

Sau khi có được đối tượng mong muốn để chuẩn bị cho quá trình OCR, ta thực hiện việc tiền xử lý trước khi đưa vào bộ nhận dạng kí tự.

Rút trích đặc trưng là một bước tiền xử lý quan trọng trong việc Xử lý ảnh.

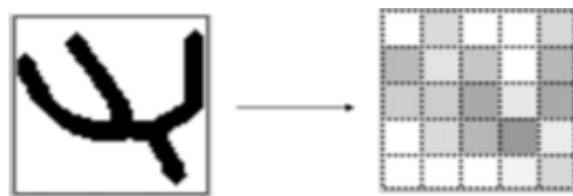
Ở bước này, tất cả những hình ảnh kí tự ở dạng nhị phân sẽ được chuẩn hóa về ma trận không gian vector $N \times N$ với việc giữ nguyên tỷ lệ khung hình gốc, mỗi một vector sẽ đại diện cho một kí tự, việc làm này gọi là Rút trích đặc trưng.

Mục đích chung của Rút trích đặc trưng là sử dụng những đặc trưng có được, thông qua các thuật toán, tối ưu hóa tỷ lệ nhận dạng. Có hai phương pháp chính trong việc rút trích đặc trưng:

1. Phương pháp thống kê

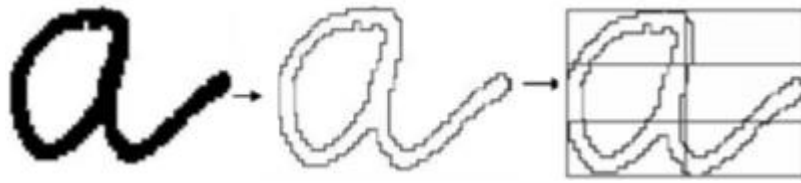
***Khoanh vùng:** Mỗi hình ảnh kí tự sẽ được phân chia thành $N \times M$ vùng. Các đặc trưng sẽ được trích xuất từ mỗi vùng ở mỗi tính năng cục bộ cụ thể như:

- Tính năng khoanh vùng theo mật độ: Số lượng điểm ảnh ở nền trước trong mỗi ô vuông (grid) được xem là một đặc điểm. Những ô vuông có mật độ các điểm ảnh cao hơn sẽ có màu đậm hơn như hình bên dưới



Hình 3.6 Khoanh vùng theo mật độ

- Tính năng khoanh vùng theo hướng: Dựa trên các viền cạnh của hình ảnh kí tự. Đối với mỗi vùng, bao quanh nó là những đường viền và ta có thể tạo ra được một biểu đồ định hướng bằng cách phân tích các điểm ảnh với ma trận liên kề 3×3 .



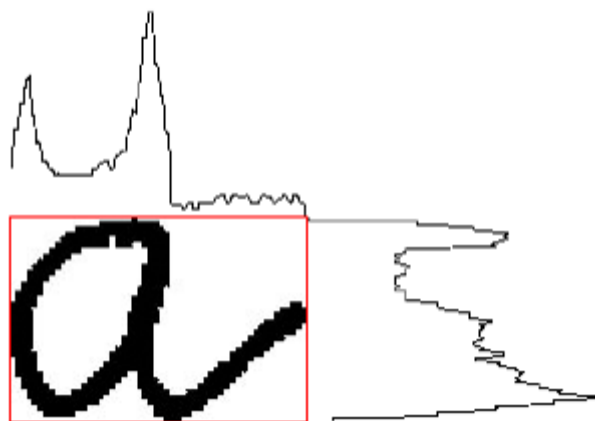
Hình 3.7 Khoanh vùng theo hướng

Dựa trên cấu trúc cơ bản của hình ảnh ký tự, ta có thể phân biệt các đường thẳng riêng lẻ.

*Biểu đồ chiếu và trắc diện

- Biểu đồ chiếu: Ý tưởng cơ bản sử dụng biểu đồ chiếu là có thể đơn giản hóa hình ảnh ký tự hai chiều về một chiều. Các đặc trưng sau khi chiếu có thể độc lập với nhiễu và sự biến dạng, nhưng còn phụ thuộc vào phép xoay ảnh.

Biểu đồ chiếu đếm số lượng điểm ảnh trong mỗi cột và hàng của hình ảnh ký tự. Biểu đồ chiếu có thể phân tách các ký tự ví dụ như “m” và “n”.



Hình 3.8 Biểu đồ khi chiếu ngang và dọc

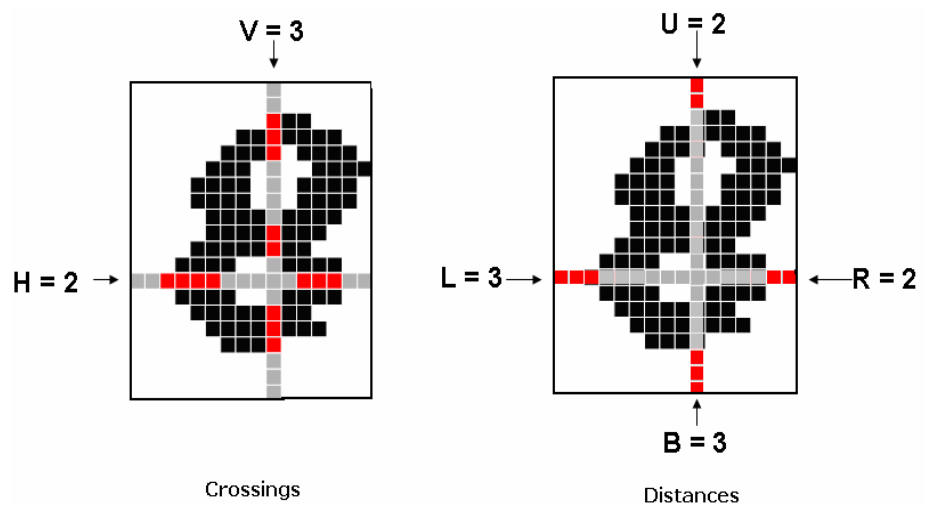
- Trắc diện: Trắc diện đếm khoảng cách giữa khung hình chữ nhật chứa ký tự đến biên cạnh gần nhất của ký tự. Phương pháp này biểu diễn khá tốt hình dạng bên ngoài của các ký tự và phân biệt được nhiều chữ cái khác nhau như “p” và “q”.



Hình 3.9 Trắc diện tính khoảng cách của các cạnh với khung hình chữ nhật

***Giao điểm và khoảng cách**

- Giao điểm là phương pháp đếm số vùng chứa các điểm ảnh của kí tự so với nền ảnh theo cả chiều dọc và chiều ngang tại điểm đang xét.
- Khoảng cách là phương pháp tính toán khoảng cách giữa điểm ảnh ngoài cùng của kí tự đến khung hình chữ nhật theo chiều dọc và chiều ngang tại điểm đang xét.



Hình 3.10 Giao điểm và khoảng cách

==

2. Phương pháp cấu trúc

Các kí tự có thể được thể hiện bằng các đặc điểm cấu trúc với khả năng thích

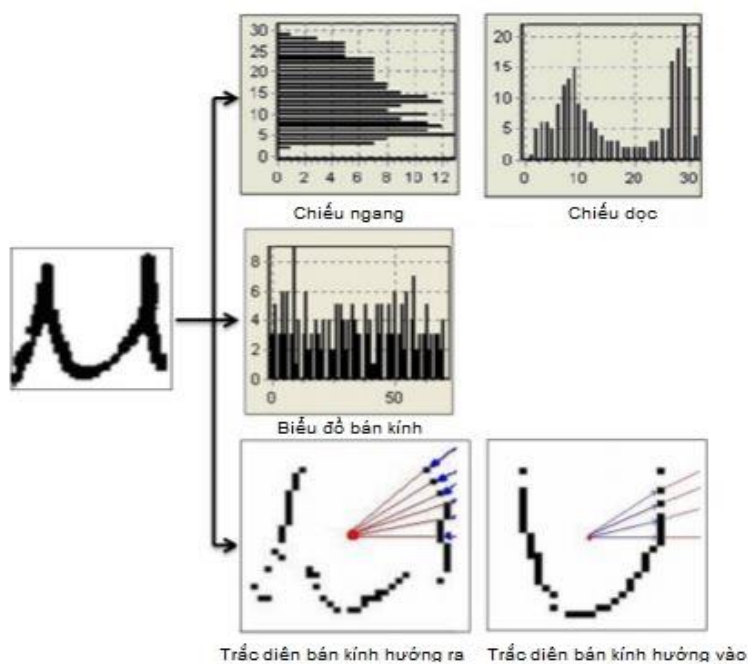
ứng cao đối với các biến dạng và biến thể kiểu dáng. Nó dựa trên các thuộc tính về cấu trúc liên kết và hình học của kí tự, ví dụ như: tỷ lệ cạnh, các điểm giao, vòng lặp, điểm nhánh, đường nét và hướng, điểm uốn giữa hai điểm, đường cong theo chiều ngang ở trên hoặc dưới.



Hình 3.11 Ví dụ về phương pháp cấu trúc

Có ba loại đặc tính chính ở phương pháp này:

- Biểu đồ chiều theo chiều ngang và dọc
- Biểu đồ bán kính
- Trắc diện bán kính hướng ra ngoài và hướng vào trong



Hình 3.312 Các đặc tính của phương pháp cấu trúc

*Sau khi rút trích các đặc trưng, đưa vào bộ nhận dạng.

3.2 Nhận dạng thực thể định danh (Named-Entity Recognition hay NER)

3.2.1 Khái niệm

Nhận dạng thực thể định danh (NER) [2] là một nhiệm vụ con của việc trích xuất thông tin mà nó thực hiện tìm kiếm và phân loại các thành phần đã được định danh trong văn bản vào những loại đã xác định trước như tên người, địa điểm, thời gian, số lượng,...



Hình 3.13 Ví dụ về nhận dạng thực thể định danh

(Nguồn: <https://sharecode.vn/source-code/ner-name-entity-recognition-nhan-dien-thuc-the-co-ten-trong-van-bang-tieng-viet-14723.html>)

3.2.2 Vai trò

NER có hai vai trò chính trong việc xử lý văn bản là xác định thực thể và trích xuất thực thể.

* Trong xác định thực thể, NER thực hiện nhiệm vụ phân loại thực thể đó theo các dạng thực thể đã được định sẵn. Để thực hiện nhiệm vụ này, cơ sở dữ liệu cần có sẽ phụ thuộc vào phạm vi của bài toán ta đang giải quyết, thường là một quần thể lớn như tập hợp địa chỉ bao gồm các tên đường, quận, số nhà,... hay có thể là tập hợp nhỏ tên của các nhân viên trong một công ty.

* Trong phần trích xuất thực thể, thường chú trọng phần kiểm tra lỗi chính tả và độ tin cậy của một loại thực thể. Với kiểm tra lỗi chính tả, đây là phần không thể thiếu cho một mô hình NER chuẩn cần có, nó giúp cho việc trích xuất đạt độ chính xác cao nhất và giảm thiểu sự nhận dạng sai các thực thể khác. Nếu như ta chấp nhận có thể sai một hoặc một vài kí tự, ta cần xác định được độ tin cậy của một thực thể để dựa vào đó, dự đoán được chính xác thực thể với độ chính xác cao nhất trong cơ sở dữ liệu.

Ví dụ ta cần trích xuất dữ liệu có tên “Nguyễn Tri Phương” trong cơ sở dữ liệu về tên đường, thì mô hình NER sẽ trả về kết quả “Nguyễn Tri Phương” vì đây là một thực thể có độ tin cậy cao nhất trong cơ sở dữ liệu.

3.2.3 Các hướng tiếp cận

Có hai hướng tiếp cận chính trong việc xây dựng một mô hình NER đó là kỹ thuật dựa trên ngữ pháp và kỹ thuật dựa trên thống kê.

Với kỹ thuật dựa trên ngữ pháp, kỹ thuật này sẽ mang lại độ chính xác cao hơn do được xây dựng bằng thủ công, đổi lại sẽ tốn nhiều thời gian để xây dựng cũng như phân tích với bộ dữ liệu ngôn ngữ.

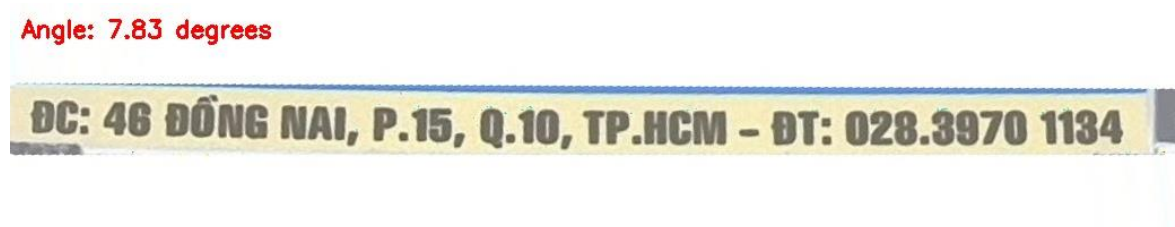
Với kỹ thuật dựa trên thống kê sẽ mang lại tốc độ nhanh hơn về mặt thời gian truy xuất và rút trích, nhưng lại mất thời gian để gắn nhãn chú thích bằng tay và đòi hỏi phải có một số lượng lớn dữ liệu có liên quan.

3.3 Chỉnh hướng văn bản (Text Skew Correction)

3.3.1 Khái niệm



Hình 3.14 Đoạn text trước khi xoay

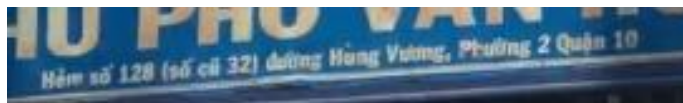


Hình 3.15 Đoạn text sau khi xoay

Chỉnh hướng văn bản (Text Skew Correction) [3] là một công cụ hữu ích giúp chúng ta tự động điều chỉnh lại đoạn văn bản về đúng góc nhìn – hàng ngang (theo cách sắp xếp của chữ Latin) hoặc hàng dọc (theo Hán tự Trung Quốc, Kanji Nhật Bản,...) mỗi khi chúng bị lệch, nghiêng hoặc xéo.

Chỉnh hướng văn bản được thực hiện trên rất nhiều giải thuật khác nhau. Điển hình nhất là các giải thuật như phát hiện dòng – Một kỹ thuật dựa trên tìm kiếm các dòng trong một bức ảnh, hay giải thuật tính độ nghiêng vùng chứa văn bản để đưa về góc nhìn phù hợp.

3.3.2 Ứng dụng trong Scene Text



Hình 3.16 Các Scene Text đã được cắt

Lý do rõ rệt nhất khi dùng chỉnh hướng văn bản trong **Scene Text** là với những dữ liệu **Scene Text** thông thường sẽ không tuân theo một định dạng hay góc nhìn nhất định (máy quay chỉ mang tính tương đối), các mẫu và form của chúng cũng rất đa dạng. Do đó, đây là một bước thiết yếu cần phải có trong OCR để nhận dạng một cách tối ưu nhất.



Hình 3.17 Sau khi chỉnh góc nhìn

3.3.3 Các bước thực hiện

* Phát hiện vùng văn bản trong một hình ảnh

Để phát hiện được vùng có chứa văn bản ta cần tìm được đường viền bao bọc tất cả phần văn bản và nối nó lại thành một hình chữ nhật. Rất may mắn, chúng ta có thể sử dụng Opencv - một thư viện nổi tiếng có rất nhiều hàm hỗ trợ cho phần Phát hiện này, với các kĩ thuật như chuyển đổi sang ảnh xám (`cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)`), đảo ngược màu nền và chữ (`cv2.bitwise_not`), lấy ngưỡng (`cv2.threshold`) và hàm tìm đường viền (`cv2.findContours`).

* Tính độ nghiêng của vùng chứa văn bản

Chúng ta cũng dùng các hàm tiện dụng của Opencv để tự động tính toán độ nghiêng, qua đó thực hiện việc xoay ngược lại với độ nghiêng đó. Bằng cách sử dụng hàm

`cv2.minAreaRect`, hàm này nhận input đầu vào là tọa độ các điểm của khung hình chữ nhật tìm được ở bước trước đó, và trả về độ nghiêng trong khoảng $[0;90]$.

* Xoay hình ảnh để đưa văn bản về đúng góc nhìn

Cuối cùng việc xoay hình cũng qua hai hàm có sẵn là `cv2.getRotationMatrix2D` giúp lấy ma trận xoay 2 chiều và `cv2.warpAffine` – đây là hàm sử dụng ma trận có được trước đó để chuyển đổi tuyến tính (Linear Transformation) về hình ảnh kết quả sau khi xoay.

3.4 Khoảng cách Levenshtein

3.4.1 Khái niệm

Trong bộ môn khoa học máy tính, khoảng cách Levenshtein [\[4\]](#) nổi tiếng là một thuật toán thể hiện sự khác biệt giữa hai chuỗi ký tự. Trong đó, sự sai khác thể hiện qua ba phép biến đổi:

- Xóa một ký tự
- Thêm một ký tự
- Thay ký tự này bằng ký tự khác

Một ví dụ điển hình nhất là khoảng cách giữa hai chuỗi “trần” và “tăng” là 3, vì đã sử dụng ba lần biến đổi như sau:

- trần -> tần (xóa ký tự “r”)
- tần -> tăng (thêm ký tự “g”)
- tăng -> tăng (thay ký tự “à” bằng ký tự “ă”)

3.4.2 Thuật toán

Để tính toán khoảng cách Levenshtein, ta cần sử dụng thuật toán quy hoạch động, một thuật toán tính toán trên mảng 2 chiều $(a+1)*(b+1)$ với a và b là độ dài của hai chuỗi cần tính.

3.4.2a Quy hoạch động

Đây là một thuật toán sử dụng các kỹ thuật dựa trên chuỗi ký tự để làm giảm thời gian chạy của các thuật toán. Tính chất của nó được thể hiện qua ba phương pháp là các bài toán con chồng lên nhau (overlapping subproblem), lưu trữ (memoization), và cấu trúc con tối ưu (optimal substructure).

- Với phương pháp overlapping subproblem, mỗi bài toán con được sử dụng để giải quyết nhiều bài toán lớn khác nhau. Điển hình nhất là dãy Fibonacci, ta cần cập nhật các giá trị đã được tính để tính cho giá trị lớn hơn tiếp theo. Ví dụ ta biết $F3 = F1 + F2$ và $F4 = F2 + F3$ sẽ sử dụng hai lần tính $F2$, và việc tính $F5 = F3 + F4$ sẽ phải tốn rất nhiều thời gian để tính lại giá trị $F2$ nhiều lần hơn thế nếu như ta không cập nhật $F2$ ngay tại lúc tính toán.
- Để cập nhật các giá trị của bài toán con đã giải, ta sử dụng hướng tiếp cận memoization để lưu trữ các kết quả dự phòng cho việc sử dụng các kết quả đó sau này.
- Với phương pháp cấu trúc con tối ưu, mỗi lời giải tối ưu cho các bài toán con có thể được sử dụng để tìm các lời giải tối ưu cho bài toán toàn cục. Ví dụ như bài toán tìm đường đi ngắn nhất, mỗi bước đi từ điểm bắt đầu cần được tìm một cách ngắn nhất để sau khi đệ quy về, ta sẽ tìm được một đường đi tối ưu nhất từ điểm bắt đầu cho tới điểm kết thúc.

3.4.2b Mã giải

Bản chất của thuật toán Levenshtein sẽ sử dụng một mảng 2 chiều để tính toán sự biến đổi trong một chuỗi. Gọi S,T là hai chuỗi cần tính khoảng cách và m,n là độ dài của hai chuỗi:

```
int LevenshteinDistance(char s[1..m], char t[1..n])
    // d is a table with m+1 rows and n+1 columns
    declare int d[0..m, 0..n]

    for i from 0 to m
        d[i, 0] := i
    for j from 0 to n
        d[0, j] := j

    for i from 1 to m
        for j from 1 to n
        {
            if s[i] = t[j] then cost := 0
            else cost := 1
            d[i, j] := minimum(
                d[i-1, j] + 1,      // trường hợp xoá
                d[i, j-1] + 1,      // trường hợp thêm
                d[i-1, j-1] + cost  // trường hợp thay thế
            )
        }
    return d[m, n]
```

Hình 3.18 Mã giải thuật toán Levenshtein

Sau khi duyệt hết các sự biến đổi, giá trị $d[m, n]$ chính là kết quả cuối cùng trả về giá trị sai khác giữa hai chuỗi kí tự.

```
>>> import pylev
>>> pylev.levenshtein('kitten','sitting')
3
>>> pylev.levenshtein('garden', 'regarding')
4
>>>
```

Hình 3.19 Ví dụ áp dụng thuật toán trên ngôn ngữ Python

CHƯƠNG 4

PHƯƠNG PHÁP ĐỀ XUẤT

4.1 Tìm kiếm các hướng tiếp cận và những vấn đề liên quan

Đây là bước đầu tiên chúng em cần phải làm để tìm ra các giải pháp dùng để tiếp cận đề bài và nhận biết được những hạn chế, khó khăn trong bài toán này.

Với phần Phát hiện văn bản, như đã đề cập ở phần khái niệm, một trong những kỹ thuật đầu tiên được sử dụng là dùng SVM (Support Vector Machine) hay mạng nơ-ron tích chập để phát hiện được các vùng văn bản. Tuy rằng độ chính xác còn rất thấp nhưng bước đầu hiểu được ứng dụng của những giải thuật cơ bản này trên bài toán đang thực hiện.

Sau khi đã thực hiện chạy thử thử một vài trường hợp thì đúc kết được việc thu thập và xử lý dữ liệu trở thành một phần thiết yếu của bài toán cần được thực hiện đầu tiên.

4.2 Quá trình xây dựng mô hình và giải pháp

Được biết trong xử lý ảnh, cách hoạt động của một mô hình nhận dạng đối tượng hay văn bản thường được đưa về hai phần chính đó là: Phát hiện và nhận dạng. Với mỗi phần cần được chọn lựa một mô hình tốt để quá trình nhận dạng có được độ chính xác cao nhất.

Ở **phần 1** - phát hiện văn bản, sau khi nghiên cứu tìm tòi thì đã tìm thấy mã nguồn text detection ctpn (connectionist text proposal network) của Mr.Eragonruan [\[6\]](#) trên github để tham khảo và chúng em sử dụng trên dữ liệu ảnh chụp biển hiệu với kết quả như mong đợi. Các dòng chữ trên tấm ảnh được phát hiện gần như đầy đủ và

chính xác nên ở phần này không cần phải huấn luyện lại. Thay vào đó, chỉnh sửa các tham số trong thư mục config để phát hiện được đầy đủ dòng địa chỉ và biển số nhà.

```
class Config:
    MAX_HORIZONTAL_GAP = 50
    TEXT_PROPOSALS_MIN_SCORE = 0.7
    TEXT_PROPOSALS_NMS_THRESH = 0.2
    MIN_V_OVERLAPS = 0.7
    MIN_SIZE_SIM = 0.7
    MIN_RATIO = 0.5
    LINE_MIN_SCORE = 0.9
    TEXT_PROPOSALS_WIDTH = 10
    MIN_NUM_PROPOSALS = 2
```

Hình 4.1 Chỉnh sửa các tham số trong phát hiện văn bản

Phần 2 – nhận dạng kí tự, Chúng em đã huấn luyện được một mô hình OCR cho **Scene Text** với độ chính xác hơn 99% trên cả tập huấn luyện và tập đánh giá. Nhưng để khớp với mô hình có đầu vào là một kí tự, chúng em còn sử dụng CRAFT – một mã nguồn khác của tổ chức Clova AI Research [7] trên github để phân tách từng kí tự trong đoạn văn bản được trả về ở phần 1, sau đó đưa vào mô hình. Song song đó, chúng em còn sử dụng thư viện Tesseract OCR của Google [8] (có tích hợp mô hình nhận dạng) để cải thiện độ chính xác chung của cả luận văn.

4.3 Giai đoạn 1: Thu thập và tiền xử lý dữ liệu

Đây là giai đoạn quan trọng nhất, nó quyết định tất cả các giai đoạn sau sẽ phụ thuộc vào việc dữ liệu có cấu trúc như thế nào, kích thước, ngữ cảnh ra sao,...

4.3.1 Thu thập dữ liệu

Đây là một ứng dụng về xử lý ảnh nên việc thu thập, chọn lọc dữ liệu hình ảnh cũng tuân theo những quy định chung như sau:

- * Phạm vi lấy dữ liệu nằm trong vùng nội thành Thành phố Hồ Chí Minh (Như đã có đề cập ở phần phạm vi nghiên cứu ở Chương 1).
- * Quy chuẩn lấy hình ảnh theo hướng giải quyết ở Chương 2 (về màu nền, độ sáng, kích thước, độ phân giải,...).
- * Góc máy quay phải thẳng, không được nghiêng quá 30 độ trong quá trình thu thập dữ liệu.
- * Khi đi lấy dữ liệu cần có bộ định vị trên xe (nếu có). Chúng em sử dụng những bộ dữ liệu có tích hợp định vị kèm với từng tấm ảnh, cho biết được vị trí đang trên đường nào, phường gì, quận mấy.

4.3.2 Tạo dữ liệu cho mô hình OCR

Vì dữ liệu lấy được ở phần thu thập dữ liệu thực tế không đủ nhiều để huấn luyện một mô hình OCR, nên cần có một mô hình tạo dữ liệu đúng với dữ liệu cần huấn luyện.

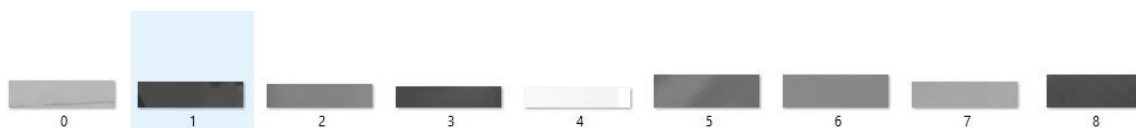
Chúng em đã sử dụng mã nguồn mở TextRecognitionDataGenerator do Mr.Belval công bố trên Github [\[9\]](#) để tạo bộ dữ liệu với màu nền được tổng hợp giống với màu nền địa chỉ và kiểu văn bản **Scene Text**.

The quick brown fox jumps over the lazy dog.

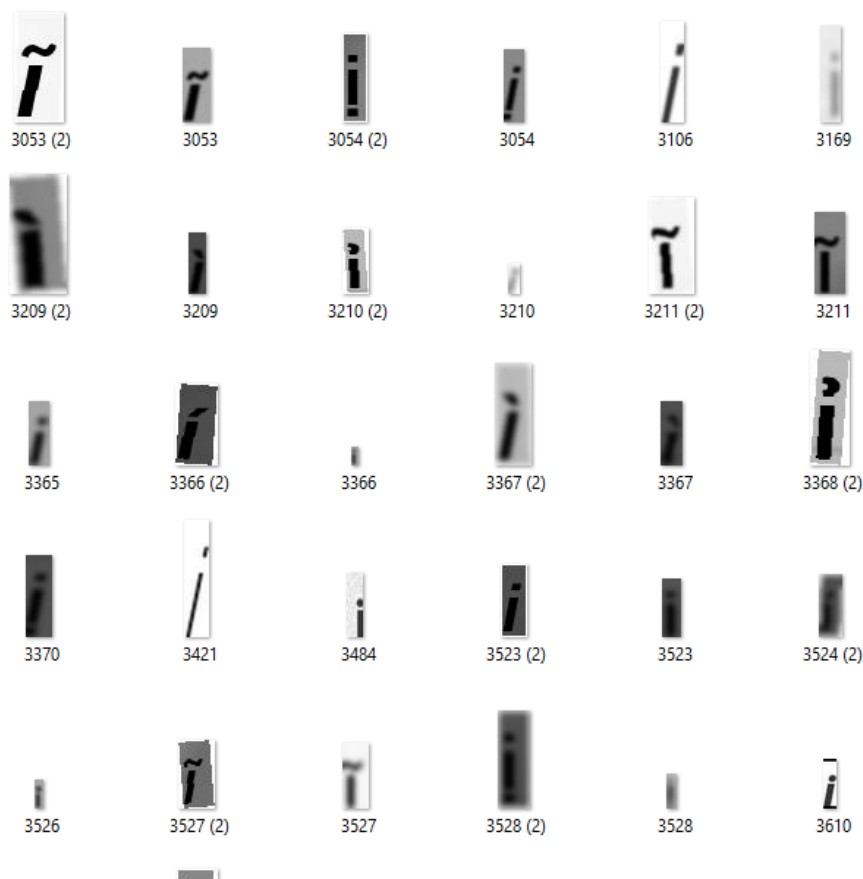
The quick brown fox jumps over the lazy dog.

The quick brown fox jumps over the lazy dog.

Hình 4.2 Một số font chữ được sử dụng để tạo dữ liệu



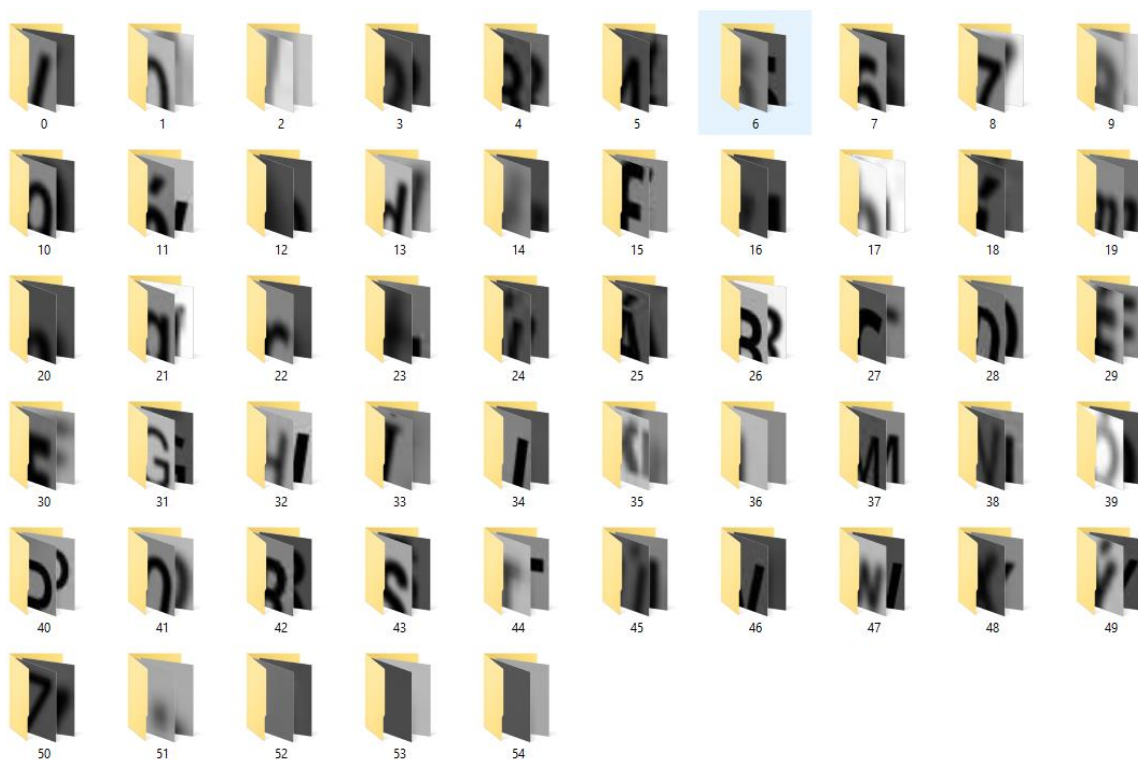
Hình 4.3 Màu nền được sử dụng để tạo dữ liệu



Hình 4.4 Dữ liệu sau khi được tạo từ TextRecognitionDataGenerator

Do mục đích là muốn xây dựng bộ OCR tiếng việt không dấu, nhưng dữ liệu thực tế thu thập là những địa chỉ tiếng việt có dấu nên chúng em gắn nhãn những ký tự có dấu về một ký tự đại diện sau đó sử dụng phần kiểm tra lỗi chính tả để có được kết quả chính xác, ví dụ như: â, à, á, ả, ã, â, ă, ằ, ắ, ẳ, ẵ, ặ sẽ gom về ký tự a. Có những ký tự viết hoa, viết thường gần giống nhau nên sẽ gom về 1 ký tự là chữ viết hoa, ví dụ như: O và o sẽ gắn nhãn là O, p và P sẽ gắn nhãn là P,... Tóm lại, dữ liệu huấn

luyện sẽ gồm 55 thư mục tương ứng với 55 nhãn ký tự, mỗi thư mục có khoảng 1814 ký tự.



Hình 4.5 Dữ liệu huấn luyện

/ 0 1 2 3 4 5 6 7 8 9 a b c d e f g h i m n q r t y A B C D E F G H I J K L M N O P Q R S T U V W X Y Z : - . ,

Hình 4.6 Các nhãn ký tự được huấn luyện

Sau quá trình kiểm thử mô hình đã huấn luyện được, tiến hành lọc lại những điểm dữ liệu dự đoán sai và gắn nhãn đúng cho nó bỏ vào tập dữ liệu ban đầu để huấn luyện lại.



Hình 20 Một số ký tự nhận dạng sai từ mô hình huấn luyện được

4.3.3 Tiền xử lý dữ liệu

4.3.3a Khi huấn luyện

```
random.seed(42)
random.shuffle(imagePaths)

# loop over the input images
for imagePath in imagePaths:
    # load the image, pre-process it, and store it in the data list
    image = cv2.imread(imagePath)
    image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    image = cv2.resize(image, (64, 64))
    image = img_to_array(image)
    data.append(image)
    # extract the class label from the image path and update the
    # labels list
    label = imagePath.split(os.path.sep)[-2]
    #print(label)
    #label = 1 if label == "mask" else 0
    labels.append(label)

# scale the raw pixel intensities to the range [0, 1]
data = np.array(data, dtype="float") / 255.0
labels = np.array(labels)
# partition the data into training and testing splits using 75% of
# the data for training and the remaining 25% for testing
(trainX, testX, trainY, testY) = train_test_split(data,
    labels, test_size=0.25, random_state=42)
# convert the labels from integers to vectors

trainY = to_categorical(trainY, num_classes=52)
testY = to_categorical(testY, num_classes=52)
```

Hình 4.8 Tiền xử lý dữ liệu cho tập huấn luyện

Việc đầu tiên trong các bước xử lý là trộn lại thứ tự các hình ảnh được đưa vào một cách ngẫu nhiên, điều này giúp mô hình tránh việc học theo thứ tự và giúp mô hình hội tụ nhanh hơn. Sau đó thực hiện chuyển kênh màu BGR- 3 kênh về GRAY- 1 kênh (ảnh xám) để bỏ qua vấn đề về đa dạng màu sắc như đã nêu ở chương 2.

Sau khi thực hiện huấn luyện với nhiều kích cỡ hình ảnh khác nhau, 64x64 cho ra kết quả tối ưu hơn các kích thước còn lại, đặc biệt trên mô hình SmallerVGGNet.

Tiếp đến là đoạn chuẩn hóa dữ liệu (đưa giá trị điểm ảnh [0-255] về [0-1]) để làm tăng tốc độ huấn luyện của mô hình và chia tập dữ liệu thành tập huấn luyện và tập kiểm thử.

Cuối cùng là đưa kiểu dữ liệu của các nhãn về các **one hot vector**.

4.3.3b Khi đánh giá

Mô hình sẽ được chia các bước tiền xử lý cho các khung hình chữ nhật có được thành 3 bước: Trước khi phát hiện, sau khi phát hiện và trước khi nhận dạng.

- Trước khi phát hiện

Sử dụng hàm *resize* của thư viện *OpenCV* để đưa ảnh input đầu vào về một kích thước nhất định thuận lợi cho việc phát hiện văn bản. Đây là một bước rất quan trọng làm tăng hiệu quả phát hiện được các dòng chữ.

```
def resize_image(img):
    img_size = img.shape
    im_size_min = np.min(img_size[0:2])
    im_size_max = np.max(img_size[0:2])

    im_scale = float(600) / float(im_size_min)
    if np.round(im_scale * im_size_max) > 1200:
        im_scale = float(1200) / float(im_size_max)
    new_h = int(img_size[0] * im_scale)
    new_w = int(img_size[1] * im_scale)

    new_h = new_h if new_h // 16 == 0 else (new_h // 16 + 1) * 16
    new_w = new_w if new_w // 16 == 0 else (new_w // 16 + 1) * 16

    re_im = cv2.resize(img, (new_w, new_h), interpolation=cv2.INTER_LINEAR)
    return re_im, (new_h / img_size[0], new_w / img_size[1])
```

Hình 4.9 Thay đổi kích thước hình ảnh

- Sau khi phát hiện

Sau khi đã có được các khung hình chữ nhật, nhận thấy độ chính xác nhận dạng thấp do mô hình đặt các khung hình chữ nhật quá sát vùng chứa văn bản, dẫn đến việc có một vài nét sẽ bị mất nên cần phải nới rộng khung hình chữ nhật ra một con số cố định theo chiều ngang và chiều dọc.

```
#padding for polygon
pW = 0.025
pH = 0.25

dX = int(max(P2[0]-P1[0],P3[0]-P4[0]) * pW)
dY = int(max(P4[1]-P1[1],P3[1]-P2[1]) * pH)
P1[0] = max(0, P1[0]-dX)
P1[1] = max(0, P1[1]-dY)
P2[0] = min(w, P2[0]+dX)
P2[1] = max(0, P2[1]-dY)
P3[0] = min(w, P3[0]+dX)
P3[1] = min(h, P3[1]+dY)
P4[0] = max(0, P4[0]-dX)
P4[1] = min(h, P4[1]+dY)
```

Hình 4.10 Nới rộng vùng chứa văn bản



Hình 4.11 Khung chứa địa chỉ trước khi nói rộng



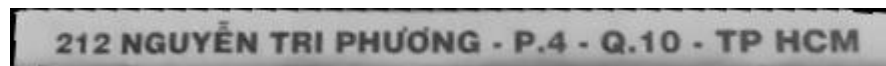
Hình 4.12 Khung chứa địa chỉ sau khi nói rộng

- Trước khi nhận dạng

Ở bước này, ta đã có được các khung hình chữ nhật được tiền xử lý trước đó và đã được chỉnh hướng, thực hiện chuyển đổi các bit trong ảnh **ROI** (Region Of Interest) với hàm **bitwise_not**. Vì mô hình OCR chúng em xây dựng với những dữ liệu input là chữ đen và nền khác đen, nên việc chuyển đổi này giúp cho mô hình dự đoán được chính xác nhất khi đưa về cùng một gam màu chữ nhất định.

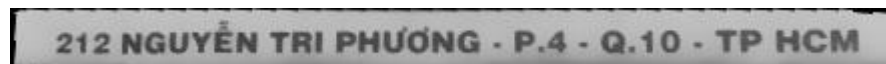


Hình 4.13 Ảnh ROI trước khi chuyển đổi



Hình 4.14 Ảnh ROI sau khi chuyển đổi

Ngoài ra, bộ lọc song phương (bilateralFilter) cũng được sử dụng để làm mịn, bảo toàn cạnh và giảm nhiễu cho ảnh ROI trước khi thực hiện dự đoán.



Hình 4.15 Ảnh ROI sau khi dùng bộ lọc song phương

```

rows, cols = roi2.shape
whites=0
blacks=0
tmp=[]

for k in range(rows):
    for m in range(cols):
        tmp.append(int(roi2[k,m]))
pix_min=min(tmp)
pix_max=max(tmp)
thresh_pix=(pix_max + pix_min)/2

for k in range(rows):
    for m in range(cols):
        if(int(roi2[k, m]>thresh_pix)):
            whites=whites+1
        else:
            blacks=blacks+1

if(whites < blacks):
    roi2=cv2.bitwise_not(roi2)

roi2 = cv2.bilateralFilter(roi2, 11, 17, 17)

```

Hình 4.16 Chuyển đổi gam màu và sử dụng bộ lọc song phương

Để mô hình OCR của thư viện Tesseract trả về kết quả tốt nhất thì ảnh đầu vào cần được xử lý, dù đã áp dụng chỉnh hướng và áp dụng bộ lọc song phương cho ảnh ROI như trên nhưng nhiều kết quả trả về từ Tesseract vẫn còn chưa tốt. Để cải thiện nhóm em đưa ra ý tưởng chuyển đổi ảnh ROI về dạng ảnh nhị phân cho những ảnh ROI không tốt trước đó.

Sau khi nghiên cứu nhóm em đưa ra 2 dạng thuật toán đưa ảnh ROI về nhị phân cho nhiều trường hợp ảnh khác nhau. Chúng em áp dụng song song 2 thuật toán để có được ảnh nhị phân tốt nhất cho mọi trường hợp ảnh ROI khác nhau.

```

pixels=0
whites=0
blacks=0

rows, cols = roi.shape

tmp=[]

for k in range(rows):
    for m in range(cols):
        tmp.append(int(roi[k,m]))

pix_min=min(tmp)
pix_max=max(tmp)
thresh_pix=(pix_max + pix_min)/2

for k in range(rows):
    for m in range(cols):
        if(int(roi[k, m]>thresh_pix)):
            whites=whites+1
        else:
            blacks=blacks+1
        pixels+=int(roi[k,m])

mean = int(pixels/(rows*cols))

if(whites > blacks):
    for i in range(rows):
        for j in range(cols):
            if(roi[i,j]>mean):
                roi[i, j] = 255
            else:
                roi[i, j] = 0
else:
    for i in range(rows):
        for j in range(cols):
            if(roi[i,j]>mean):
                roi[i, j] = 0
            else:
                roi[i, j] = 255

roi = cv2.bilateralFilter(roi, 11, 17, 17)

```

Hình 4.17 Chuyển đổi ảnh nhị phân dạng 1

54 Thành Thái, P. 12, Q.10 - ĐT: 3864.1670

Hình 21 Ảnh nhị phân sau khi chuyển đổi dạng 1

```

whites=0
blacks=0

rows, cols = roi.shape

tmp=[]

for k in range(rows):
    for m in range(cols):
        tmp.append(int(roi[k,m]))

pix_min=min(tmp)
pix_max=max(tmp)
thresh_pix=(pix_max + pix_min)/2

tmp1=[]
tmp2=[]

for k in range(rows):
    for m in range(cols):
        if(int(roi[k, m]>thresh_pix)):
            tmp1.append(int(roi[k,m]))
            whites=whites+1
        else:
            tmp2.append(int(roi[k,m]))
            blacks=blacks+1

if(whites > blacks):
    thresh = min(tmp1)
    for i in range(rows):
        for j in range(cols):
            if(roi[i,j]>thresh):
                roi[i, j] = 255
            else:
                roi[i, j] = 0
else:
    thresh=max(tmp2)
    for i in range(rows):
        for j in range(cols):
            if(roi[i,j]>thresh):
                roi[i, j] = 0
            else:
                roi[i, j] = 255

roi = cv2.bilateralFilter(roi, 11, 17, 17)

```

Hình 4.19 Chuyển đổi ảnh nhị phân dạng 2

86 Hùng Vương - Phường 1 - Quận 10 - TP.HCM

Hình 4.20 Ảnh nhị phân sau khi chuyển đổi dạng 2

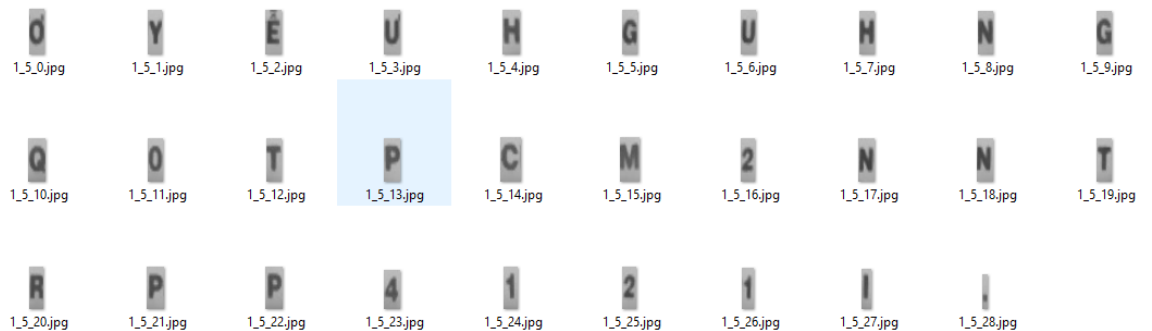
Do mô hình OCR tự xây dựng đầu vào là một ảnh ký tự nên để sử dụng mô hình chúng em sử dụng CRAFT để tách từng ký tự từ ảnh ROI. Để xác định đúng vị trí của ký tự trong chuỗi địa chỉ ảnh ROI ban đầu, chúng em đưa tọa độ mỗi ký tự vào danh sách sau đó sắp xếp danh sách đó tăng dần theo giá trị tọa độ x của mỗi ký tự.



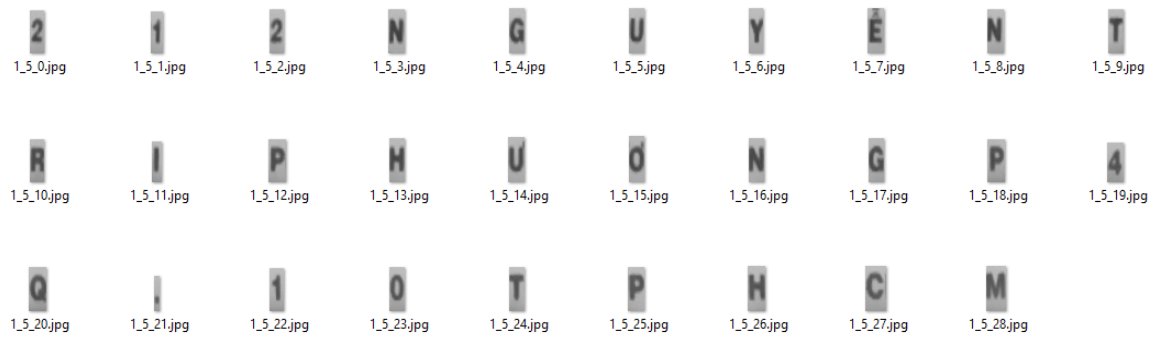
Hình 22 Kết quả sau khi áp dụng CRAFT cho ảnh ROI



Hình 23 Ảnh một ký tự sau khi cắt ra từ ảnh ROI của CRAFT



Hình 24 Ảnh các ký tự chưa sắp xếp đúng vị trí ban đầu



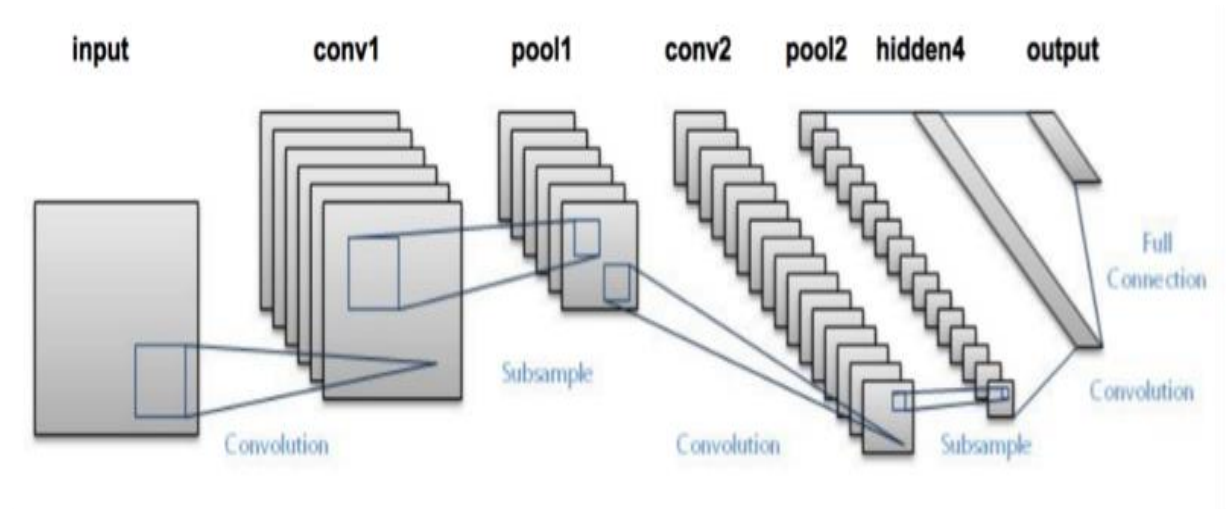
Hình 25 Ảnh các ký tự sau khi sắp xếp

4.4 Giai đoạn 2: Xác định mô hình phù hợp

Ở giai đoạn này, việc tìm kiếm mô hình phù hợp với bài toán cũng là một bước quan trọng không kém. Mô hình tốt giúp chúng ta tối ưu được độ chính xác và tối thiểu sự mất mát. Chính vì thế, nhận thấy được bài toán nhận diện văn bản này có tính giản đơn hơn về mặt rút trích và lưu trữ các đặc trưng so với nhận diện đối tượng, chúng em đã áp dụng những mô hình phân lớp từ đơn giản đến nâng cấp hơn.

4.4.1 Lenet

Lenet [\[10\]](#) là một trong những mạng lồi nổi tiếng nhất và đơn giản nhất của CNN (Convolutional Neural Network). Cấu trúc chỉ bao gồm 2 lớp Convolution và Maxpooling, 2 lớp fully connected và cuối cùng là lớp output (softmax hoặc sigmoid).



Hình 4.25 Cấu trúc mạng Lenet

(Nguồn: <https://www.pyimagesearch.com/2016/08/01/lenet-convolutional-neural-network-in-python/>)

Chính nhờ sự giản đơn của Lenet nên thời gian huấn luyện được rút ngắn đáng kể và tốc độ hội tụ cũng nhanh hơn qua từng epoch, trả về độ chính xác cực kì cao (99,6% trên tập kiểm thử).

Tuy nhiên, vì là một mô hình mạng đơn giản nên việc rút trích và sử dụng các đặc trưng của hình ảnh cũng đơn giản theo, dẫn đến đánh giá trên tập kiểm thử có **population** khác trở nên không tốt => Không tốt cho bài toán về đa phân loại.

4.4.2 SmallerVGGNet

4.4.2a Một vài khái niệm

* Convolution 2D

Convolution 2D là một lớp có đầu vào của phép toán tích chập là ba chiều. Tuy nhiên, nó được gọi là “tích chập hai chiều” vì chuyển động của bộ lọc trên hình ảnh xảy ra theo hai chiều. Bộ lọc này thường chạy trên hình ảnh ba lần trên ba lớp Conv.

*** Activation**

Activation được biết đến là một hàm kích hoạt hay một “công” toán học với đầu vào là một Neuron và đầu ra của nó ở lớp tiếp theo, phụ thuộc vào một quy tắc hay một ngưỡng nào đó. Hoặc nó còn được xem là một phép biến đổi ánh xạ các tín hiệu đầu vào thành các tín hiệu đầu ra cần thiết cho mạng Neuron hoạt động.

*** MaxPooling 2D**

MaxPooling2D được tạo ra nhờ vào ý tưởng “tích lũy” các đặc điểm từ ma trận được tạo ra bằng cách xoay bộ lọc trên một hình ảnh. Chức năng chính của lớp này là làm giảm kích thước không gian và số lượng các tham số, qua đó số lượng và thời gian tính toán trong mạng cũng giảm theo.

*** Flatten**

Lớp Flatten có chức năng rất đơn giản trong mạng là thu gọn các kích thước không gian đầu vào thành kích thước kênh.

*** Dense**

Dense là một lớp phổ biến có trong hầu hết các mạng Neuron. Đây đơn giản là một lớp mà mỗi đơn vị hoặc tế bào thần kinh được kết nối với các tế bào thần kinh trong lớp tiếp theo.

*** BatchNormalization**

BatchNormalization là một phương pháp được sử dụng để làm cho mạng nơ-ron nhân tạo nhanh hơn và ổn định hơn thông qua việc chuẩn hóa lớp đầu vào bằng cách tập trung lại và thay đổi tỉ lệ.

* Dropout

Dropout là một kỹ thuật chính quy quá, với mục đích làm giảm độ phức tạp của mô hình để tránh tình trạng “quá khớp”.

4.4.2b *SmallerVGGNet*

Với mục đích tìm kiếm các mạng CNN có chiều sâu hơn Lenet để rút trích được nhiều đặc trưng hơn, hướng tiếp cận tiếp theo của chúng em là mạng VGG.

VGGNet [\[11\]](#) được phát triển và công bố bởi Simonyan and Zisserman vào năm 2014 với nhiều cải tiến so với các mạng CNN trước đó. Cụ thể, VGGNet có nhiều biến thể khác nhau (11 lớp, 13 lớp, 16 lớp và 19 lớp) và có đến 3 giai đoạn sử dụng hàm kích hoạt ReLU, phân thành 3 giai đoạn đánh giá hỗ trợ cho việc dự đoán thay vì chỉ một lần duy nhất ở đầu ra mô hình Lenet.

Về nguyên tắc thiết kế mạng VGG rất đơn giản: 2 hoặc 3 lớp Convolution và sau đó là 1 lớp Max Pooling 2D, cuối cùng là lớp Flatten để chuyển ma trận 4 chiều của lớp Conv về ma trận 2 chiều. Tiếp nối sau đó là các lớp FC (Fully Connected layers) và lớp đầu ra Softmax hoặc Sigmoid.

Với tập dữ liệu lớn và tính chất bài toán không cần một mạng lưới quá “sâu” (Nếu như quá nhiều lớp sẽ rất lâu để hội tụ và làm cho mô hình “quá khớp”). Nên chúng em chọn SmallerVGGNet [\[12\]](#) – phiên bản ít lớp hơn mạng VGG được tạo bởi Mr.Adrian trên PyImageSearch. Mạng này còn sử dụng các lớp nổi tiếng như BatchNormalization – một kỹ thuật dùng để cải thiện tốc độ, hiệu suất và tính ổn định của mạng Neuron, và Dropout – một kỹ thuật cắt giảm các nút trong mạng một cách ngẫu nhiên, làm giảm sự phụ thuộc lẫn nhau giữa các Neuron để tránh tình trạng “quá khớp”.

```

# CONV => RELU => POOL
model.add(Conv2D(32, (3, 3), padding="same",
    input_shape=inputShape))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(3, 3)))
model.add(Dropout(0.25))
# (CONV => RELU) * 2 => POOL
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
# (CONV => RELU) * 2 => POOL
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
# first (and only) set of FC => RELU layers
model.add(Flatten())
model.add(Dense(1024))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dropout(0.5))
# softmax classifier
model.add(Dense(classes))
model.add(Activation("softmax"))

```

Hình 26 Một phần cấu trúc mạng SmallerVGGNet




Cũng giống như các mạng VGG, SmallerVGGNet là một phiên bản được cắt giảm với số lớp Conv là 5. Các lớp này xếp chồng lên nhau theo từng tầng với độ sâu của các lớp tăng dần từ 32 cho đến 128. Tiếp theo đó là giảm kích thước các lớp MaxPooling và hàm kích hoạt Softmax với đầu ra là 55 lớp. Mỗi tầng có sử dụng một lớp BatchNormalization và Dropout với tỉ lệ 25% trong tổng số Neuron trong mạng.

Sau khi thực hiện huấn luyện trên mô hình này, kết quả thu được gần như giống với kết quả huấn luyện Lenet về mặt tốc độ hội tụ cũng như độ chính xác trên tập kiểm

thử. Nhưng khác với Lenet, lần đánh giá trên mô hình SmallerVGGNet với cùng một tập test lại tốt hơn rất nhiều. Điều đó cho thấy được, các **Scene Text** cần được học một cách “sâu” hơn cũng như kĩ hơn thông qua nhiều lớp để giữ lại được nhiều đặc điểm hơn, và tránh học vẹt khi sử dụng kĩ thuật Dropout, nhờ đó mới có thể nhận dạng đúng những dữ liệu thuộc một **population** khác với tập dữ liệu huấn luyện cũng như kiểm thử.

Mô hình	LeNet	SmallVggNet
Độ chính xác	95,276%	97,638%

Bảng 1. Kết quả dự đoán trên 127 kí tự được cắt từ CRAFT với dữ liệu ảnh trên đường Nguyễn Thị Nhỏ

Mô hình	LeNet	SmallVggNet
	N	K
	U	I
	U	I

Bảng 2. Một số kí tự được dự đoán sai bởi LeNet nhưng đúng với SmallerVggNet

4.5 Giai đoạn 3: Tùy chỉnh các tham số và huấn luyện

Để mô hình sử dụng có thể phù hợp với dữ liệu đang có thì việc tinh chỉnh các tham số không thể thiếu trong bất kì mô hình Học sâu nào.

Như đã đề cập ở phần tiền xử lý dữ liệu, trước khi được tải lên bộ nhớ cho quá trình huấn luyện, kích thước dữ liệu được thay đổi tỉ lệ là 64x64. Trước đó chúng em đã thử các kích thước nhỏ hơn như 16x16 hay 32x32, giống với cách thay đổi tỉ lệ các bộ dữ liệu nổi tiếng như MNIST. Kết quả thu được sau khi huấn luyện cho thấy tập dữ liệu **Scene Text** cần được đưa về kích cỡ lớn hơn thế vì có nhiều nhãn (55 nhãn so với chỉ 10 ở MNIST).



Hình 27 Dữ liệu sau khi được thay đổi tỉ lệ về 64x64 và 32x32

4.6 Giai đoạn 4: Đánh giá chất lượng mô hình

Thuật toán chính dùng để đánh giá mô hình OCR là Levenshtein như đã nêu ở phần kiến thức nền tảng.

Với bộ chữ cái tiếng việt đa dạng (89 kí tự bao gồm các kí tự có dấu so với bộ chữ cái tiếng anh chỉ 26) nên việc dự đoán sẽ trở nên khó khăn hơn khi cần tới 89 nhãn để phân loại từng kí tự. Chính vì thế, chúng em đã sử dụng NER để làm giảm số nhãn cần dự đoán với cách thức như sau:

- Phân tích các chuỗi kí tự đầu vào không dấu kết hợp với độ tin cậy của nó, NER sẽ trả về kết quả đúng nhất cho chuỗi đó ở dạng có dấu. Ví dụ đầu vào là “Dong Nai” thì NER sẽ phân tích và cho đầu ra là “Đồng Nai”.
- Đổi bộ chữ cái tiếng việt thành tiếng anh để tiến hành huấn luyện (lúc này số nhãn chỉ còn 26). Với việc dự đoán chuỗi kí tự không dấu chính xác, thì NER sẽ cho ra được kết quả cuối cùng chính xác ở dạng tiếng việt có dấu.

CHƯƠNG 5

CÀI ĐẶT

Ở phần này chúng em sẽ trình bày chi tiết về cách cài đặt các thành phần được dùng trong hệ thống. Bên cạnh đó, cũng trình bày thêm về cách tinh chỉnh các tham số để làm tăng độ chính xác của bài toán.

5.1 Ngôn ngữ lập trình sử dụng

Nói về ngôn ngữ sử dụng cho xử lý ảnh – thị giác máy tính thì hầu hết mọi người sẽ nghĩ đến C++ và Python. C++ là ngôn ngữ cơ bản ở mức độ tầm trung chú trọng đến các kiến thức nền và cấu trúc của dữ liệu. Tuy nhiên, ngôn ngữ chính chúng em chọn để xây dựng hệ thống là Python vì một số lý do như sau:

- Python là ngôn ngữ lập trình đa nền tảng bậc cao. Lập trình Python hoạt động nhanh và mạnh mẽ hơn C++ qua tốc độ xử lý và cách tối ưu hóa bộ nhớ. Không những vậy, vì là ngôn ngữ bậc cao nên Python cũng có các công cụ hỗ trợ cho việc lập trình hướng đối tượng, xây dựng cấu trúc dữ liệu,... giống như C++ hay Java.
- Python có cấu trúc dễ đọc, dễ sử dụng, không có nhiều câu lệnh rườm rà trong việc khai báo cấu trúc dữ liệu. Do đó, ngôn ngữ này không đòi hỏi nhiều về khả năng lập trình, mà người dùng cần phải hiểu rõ những khái niệm, bản chất các hàm, các biến được sử dụng trong ngôn ngữ lập trình này.
- Python còn có các hàm tiện lợi cho việc quản lý bộ nhớ, hay dọn dẹp những dữ liệu không cần thiết,...
- Đặc biệt hơn, đa số người dùng chọn Python nhờ vào khả năng mở rộng và khả năng nhúng của nó. Python là một framework có thể dễ dàng kết hợp với

các ngôn ngữ khác như C hay C++, thuận tiện cho việc thực hiện những ứng dụng lớn cần có sự chi tiết của ngôn ngữ C++ và hiệu năng chạy chương trình của Python.



Hình 5.1 Ví dụ về các ưu điểm của Python

(Nguồn: <https://www.pinterest.com/pin/345932815130283288/>)

5.2 Bộ xử lý hình ảnh (Graphic Processing Unit hay GPU)

Bộ xử lý hình ảnh hay còn được gọi tắt là GPU là bộ vi xử lý chuyên dụng nhận nhiệm vụ tăng tốc, xử lý hình ảnh cho bộ xử lý trung tâm CPU.

Với sự ra đời của những chiếc card đồ họa GPU tân tiến như hiện nay, thật dễ hiểu khi một GPU có thể tiếp nhận hàng ngàn cho đến hàng triệu luồng dữ liệu cùng một lúc và tăng tốc thời gian chạy của các phần mềm tới hơn 100 lần so với một chiếc CPU thông thường.

- **Cài đặt GPU cho hệ thống**

Với việc tận dụng GPU để huấn luyện mô hình cũng như làm tăng tốc độ xử lý cho các chương trình, góp phần đẩy nhanh quá trình phát triển các thuật toán, mô hình liên quan đến AI, học máy hay xử lý ảnh. Qua đó, để cài đặt cho hệ thống nhận dạng này, chúng em đã sử dụng GPU và CUDA (một động cơ tính toán trong GPU được phát triển bởi NVIDIA) trên hầu hết mã nguồn sử dụng.

Card đồ họa chúng em sử dụng là NVIDIA GeForce RTX 2060 với 6GB VRAM và 4GB Share Memory, vừa đủ để xử lý đa luồng cho CRAFT cũng như OCR tự huấn luyện.

CHƯƠNG 6

THỰC NGHIỆM VÀ KẾT QUẢ

6.1 Các vấn đề gặp phải khi thực nghiệm

Đầu tiên có thể đề cập đến sự đa dạng trong việc thiết kế biển hiệu của mỗi người. Có những biển hiệu thể hiện vùng địa chỉ không trên cùng một dòng, hay phần địa chỉ viết liền với phần số điện thoại hoặc email. Vì chúng em đã thu hẹp kiểu phát hiện vùng địa chỉ về một form nhất định (phải nằm trên cùng một dòng), nên điều đó đã trực tiếp ảnh hưởng đến kết quả phát hiện cũng như nhận dạng văn bản.



Hình 6.1 Ví dụ về địa chỉ viết liền với số điện thoại



Hình 28 Ví dụ về phần địa chỉ không nằm trên cùng một dòng

Do đó, hướng để khắc phục vấn đề này là thu hẹp phạm vi nghiên cứu (chỉ lấy những phần địa chỉ nằm trên cùng một dòng), tạo ra những bộ lọc để cắt bỏ những phần bị phát hiện chung như số điện thoại, email, website,... và xóa bỏ những kí tự đặc biệt không được dùng trong phần địa chỉ biển hiệu.

```

text = re.split(r'ĐT',text)[0]
text = re.split(r'Tel',text)[0]
text = re.split(r'TEL',text)[0]
text = re.split(r'TEI',text)[0]
text = re.split(r'TEU',text)[0]
text = re.split(r'DT',text)[0]
text = re.split(r'BT',text)[0]
text = re.split(r'Email',text)[0]
text = re.split(r'Hotline',text)[0]
text = re.split(r'DI',text)[0]
text = re.split(r'BE',text)[0]
text = re.split(r'DĐ',text)[0]
text = re.split(r'MST',text)[0]
text = re.split(r'ĐI',text)[0]
text = re.split(r'website',text)[0]
text = re.split(r'DB',text)[0]
text = re.split(r'www',text)[0]
text = re.split(r'WWW',text)[0]

text = text.replace(":", "")
text = text.replace("@", "")
text = text.replace("$", "")
text = text.replace("&", "")
text = text.replace("%", "")
text = text.replace("#", "")
text = text.replace("^", "")
text = text.replace("?", "")
text = text.replace("|", "")
text = text.replace("'", "")

```

Hình 6.3 Cắt bỏ những phần không cần thiết cho địa chỉ

6.2 Kết quả

6.2.1 huấn luyện mô hình OCR

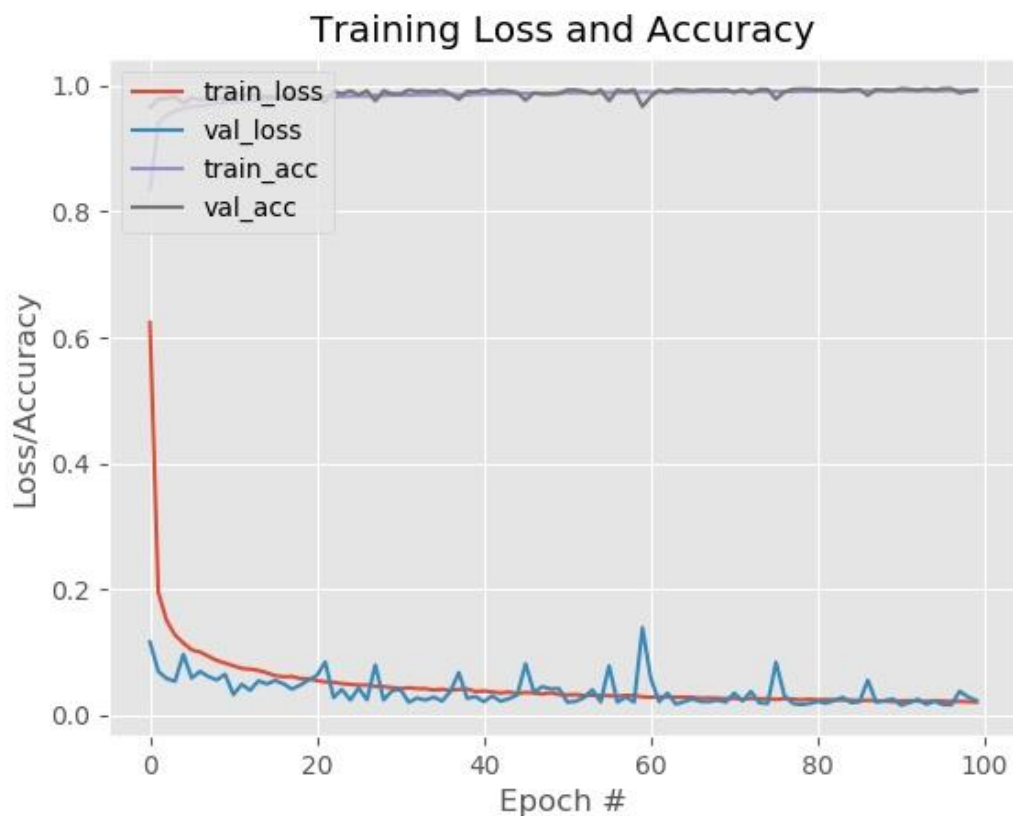
Sau khi sử dụng mô hình SmallerVggNet để huấn luyện tập dữ liệu hình ảnh với hơn 200.000 bức hình, sau 100 Epochs, kết quả thu được vượt ngoài mong đợi với độ chính xác trên tập đánh giá là 99,31%.

```

698/698 [=====] - 58s 83ms/step - loss: 0.0234 - acc: 0.9920 - val_loss: 0.0553 - val_acc: 0.9850
Epoch 88/100
698/698 [=====] - 58s 83ms/step - loss: 0.0227 - acc: 0.9921 - val_loss: 0.0202 - val_acc: 0.9942
Epoch 89/100
698/698 [=====] - 58s 83ms/step - loss: 0.0222 - acc: 0.9924 - val_loss: 0.0229 - val_acc: 0.9932
Epoch 90/100
698/698 [=====] - 58s 83ms/step - loss: 0.0213 - acc: 0.9925 - val_loss: 0.0256 - val_acc: 0.9924
Epoch 91/100
698/698 [=====] - 58s 84ms/step - loss: 0.0222 - acc: 0.9924 - val_loss: 0.0155 - val_acc: 0.9957
Epoch 92/100
698/698 [=====] - 58s 83ms/step - loss: 0.0226 - acc: 0.9920 - val_loss: 0.0196 - val_acc: 0.9947
Epoch 93/100
698/698 [=====] - 58s 83ms/step - loss: 0.0221 - acc: 0.9924 - val_loss: 0.0251 - val_acc: 0.9927
Epoch 94/100
698/698 [=====] - 58s 83ms/step - loss: 0.0214 - acc: 0.9924 - val_loss: 0.0168 - val_acc: 0.9953
Epoch 95/100
698/698 [=====] - 58s 83ms/step - loss: 0.0226 - acc: 0.9921 - val_loss: 0.0221 - val_acc: 0.9931
Epoch 96/100
698/698 [=====] - 58s 83ms/step - loss: 0.0220 - acc: 0.9923 - val_loss: 0.0164 - val_acc: 0.9955
Epoch 97/100
698/698 [=====] - 58s 84ms/step - loss: 0.0214 - acc: 0.9927 - val_loss: 0.0161 - val_acc: 0.9955
Epoch 98/100
698/698 [=====] - 58s 83ms/step - loss: 0.0216 - acc: 0.9926 - val_loss: 0.0376 - val_acc: 0.9887
Epoch 99/100
698/698 [=====] - 58s 83ms/step - loss: 0.0211 - acc: 0.9925 - val_loss: 0.0287 - val_acc: 0.9917
Epoch 100/100
698/698 [=====] - 58s 83ms/step - loss: 0.0204 - acc: 0.9928 - val_loss: 0.0230 - val_acc: 0.9931



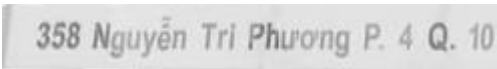
```



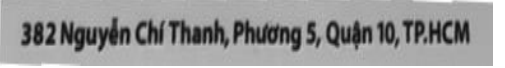


Hình 6.4 Kết quả huấn luyện mô hình OCR với SmallerVggNet



Hình 6.5 Kết quả huấn luyện mô hình OCR ở dạng biểu đồ

Thực hiện dự đoán trên tập test với mô hình vừa được huấn luyện và mô hình Tesseract OCR, ta được một số kết quả như sau:

Chuỗi văn bản	Tesseract OCR		OCR tự huấn luyện	
	Dự đoán	Độ chính xác	Dự đoán	Độ chính xác
	210A THAN BINH TRONG) PHUONG 4, QUAN 5, TP.HCM	95.65%	210A TRAN BINH TRONG PHUONG 4 QUAN 5 TP HCM	86.96%
	L1 LE HONG PHONG, P.2, Q. 10	86.67%	295 LE HONG PHONG, P 2 , Q 10	90.00%
	388 NGUYEN TRI PHUONG P. 4 Q. 10	93.94%	358 NGUYEN TRI PHUONG P 4 Q 10	87.88%

	73 TRAN PHU, P.4. Q.5	95.24%	73 TRAN PHL. P 4 Q.5	80.95%
	F121 TRAN PHU P.4 Q.5.	90.00%	I2I TRAN PHU P 4	65.00%
	382 NGUYEN CHI THANH, PHUONG 5, QUAN 10, TP.HCM.	97.87%	3 82 NGUYEN CHI THANH PHUONG 5 QUAN 10 TPHCM	89.36%
	4888 NGUYEN TRI PHUONG.P.9. 0.10	78.79%	458B NGUYEN TR1PHUON G P 9 Q 10	81.82%
	A4 DUONG HUONG GIONG -	71.43%	M8C DUONG HUONG GIANG	100.00%

Bảng 3. Kết quả và độ chính xác của một vài tấm ảnh áp dụng các mô hình OCR

Từ bảng 6.2, ta có thể nhận xét mô hình Tessract OCR nhận dạng rất tốt trên các khung địa chỉ có nhiều nhiễu, chất lượng kém và điều kiện màu sắc khác nhau. Nhưng

còn nhiều hạn chế chưa giải thích được đối với những tấm ảnh có điều kiện lý tưởng làm giảm độ chính xác đối với các vùng văn bản trên biển hiệu. Còn đối với mô hình OCR tự huấn luyện, bộ nhận dạng cũng rất tốt khi có điều kiện ít nhiễu hay ảnh chất lượng tốt. Tuy nhiên, bộ nhận dạng này phụ thuộc rất nhiều vào mô hình tách từng ký tự (CRAFT), nên ở một số ký tự như dấu phẩy hoặc khoảng trắng nếu không được phát hiện sẽ không nhận dạng được.

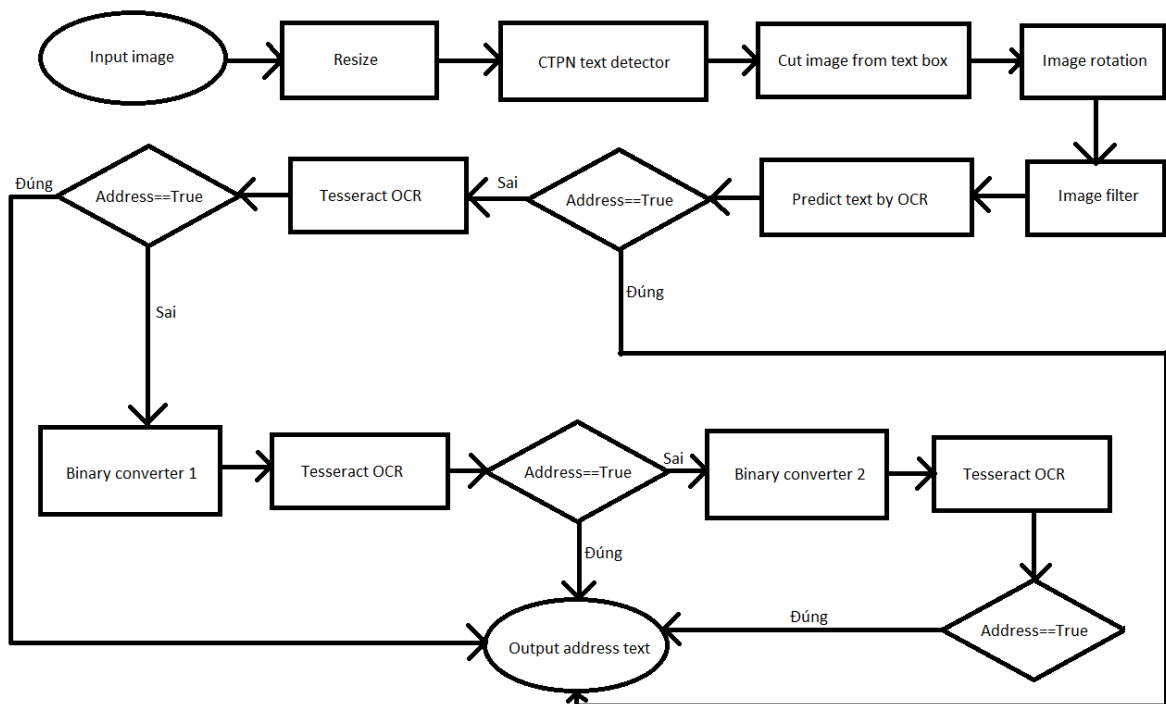


Hình 6.6 Ví dụ về việc tách ký tự sai của CRAFT

6.2.2 Tesseract OCR + mô hình OCR tự xây dựng

Nhận thấy được cả hai mô hình đều có sự khiếm khuyết riêng, chúng em đã đề xuất việc kết hợp cả hai mô hình để nhận dạng chung vì hai mô hình OCR này sử dụng hai mạng Neuron khác nhau và chúng đối lập nhau về điều kiện áp dụng, nên mô hình sẽ cải thiện được độ chính xác khi rơi vào một trong hai trường hợp trước đó đã đề cập.

Do mô hình OCR tự xây dựng có độ chính xác cao hơn Tesseract OCR nên mô hình OCR tự xây dựng sẽ được ưu tiên nhận dạng trước nếu nhận dạng còn chưa tốt sẽ cho chuyển tiếp qua mô hình Tesseract OCR để nhận dạng tiếp, quá trình tổng thể như sau:



Hình 6.7 Quá trình thực hiện nhận dạng của hệ thống

Nhận dạng chính xác phần số trong khung chứa địa chỉ cũng là một tiêu chí quan trọng vì khi kết hợp với phần định vị (nếu có như đã đề cập ở phần thu thập dữ liệu) sẽ trả về dữ liệu tên đường, phường, quận, cuối cùng sẽ có được đoạn văn bản địa chỉ hoàn chỉnh.

Ngoài khoảng cách Levenshtein để đánh giá kết quả nhận dạng được cho toàn hệ thống, thì các độ đo như Precision, Recall và F-Measure [13] cũng được áp dụng để đánh giá mô hình phát hiện vùng địa chỉ. Các độ đo sẽ được tính như sau:

- A: Số khung địa chỉ phát hiện (nhận dạng) đúng
- B: Số khung địa chỉ phát hiện (nhận dạng) sai
- C: Số khung địa chỉ không phát hiện (nhận dạng) được

$$\text{Precision} = \frac{A}{A+B}$$

$$\text{Recall} = \frac{A}{A+C}$$

$$\text{F-Measure} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

Trường hợp	Không gian mẫu	Chính xác	Nhận dạng sai	Không nhận dạng được	Precision (%)	Recall (%)	F-measure (%)
Nhận dạng vùng số của văn bản	600	258	207	135	55.5	65.6	60
Nhận dạng toàn bộ vùng văn bản	600	184	278	135	39.8	57.7	47.1

Bảng 4. Kết quả thực nghiệm các mô hình nhận dạng trên 600 tấm ảnh

6.2.3 Mô hình phát hiện vùng địa chỉ

Nhờ vào sự cải thiện độ chính xác của các mô hình nhận dạng, kết hợp với mô hình NER dùng để lọc lại các khung chữ nhật chứa địa chỉ, ta đã cải thiện được việc tìm

ra một khung duy nhất chứa vùng địa chỉ cần tìm, đúng với mục đích nghiên cứu của bài toán này.

Dữ liệu	Không gian mẫu	Phát hiện đúng	Phát hiện sai	Không phát hiện được	Precision (%)	Recall (%)	F-Measure (%)
Không sử dụng nhận dạng thực thể định danh	600	278	91	231	75.3	54.6	63.2
Sử dụng nhận dạng thực thể định danh	600	408	57	135	87.7	75.1	80.9

Bảng 5. Kết quả thực nghiệm mô hình phát hiện vùng địa chỉ trên 600 tấm ảnh

Kết quả thống kê trên 600 tấm ảnh kiểm thử khác nhau cho thấy tỉ lệ phát hiện đúng tốt hơn khi có sử dụng bộ nhận dạng thực thể định danh với F-Measure là 80,9%. Cho thấy được mô hình phát hiện tương đối tốt với bộ dữ liệu đa dạng. Tuy nhiên cần được huấn luyện với tập dữ liệu nhiều hơn nữa để tăng cường độ chính xác.

CHƯƠNG 7

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

7.1 Kết luận

Nhận dạng địa chỉ biển hiệu là một ứng dụng quan trọng cần được phát triển mạnh mẽ ở Việt Nam vì nó có nhiều ứng dụng trong thực tế, giúp các doanh nghiệp và nhà nước có được một API kết hợp với định vị, thu về được bộ dữ liệu dồi dào trên khắp đất nước với chi phí ít tốn kém và tốc độ xử lý tối ưu, hiệu quả.

Trong khóa luận này chúng em đã đề xuất được nhiều phương pháp nhằm cải thiện độ chính xác của mô hình nhận dạng bao gồm TextDetector, CRAFT, bộ nhận dạng thực thể định danh NER, Tesseract OCR và mô hình OCR tự đào tạo. Kết hợp sử dụng các phương pháp này, chúng em đã xây dựng được một hệ thống tương đối hiệu quả cho nhiều điều kiện về hình ảnh, văn bản khác nhau.

Điểm mạnh của phương pháp này là nhận dạng chính xác với những điều kiện tốt như font chữ dễ nhìn, màu chữ và nền phân biệt rõ rệt,... Tuy nhiên do cách thiết kế các biển hiệu của người dân còn rất đa dạng nên mô hình còn chưa thực sự hiệu quả ở các dạng biển hiệu hay tuyến đường khác. Ngoài ra, tập dữ liệu huấn luyện cũng còn rất hạn chế, chưa đủ nhiều.

7.2 Hướng phát triển

Thu thập thêm nhiều hình ảnh hơn nữa với các đoạn văn bản có font lạ, điều kiện sáng và chất lượng ảnh khác nhau. Hơn nữa, cần giải quyết và phát triển thuật toán để nhận dạng được các khung địa chỉ có form khác thường (không nằm trên cùng một hàng, chữ bị nghiêng, xéo,...). Cải thiện thêm bộ nhận dạng thực thể định danh cho

phép trả về kết quả gần đúng nhất, tránh trường hợp rút trích nhầm khi độ tin cậy không cao.

TÀI LIỆU THAM KHẢO

- [1]. **Wikipedia, the free encyclopedia.** “Scene Text” In: (). url: https://en.wikipedia.org/wiki/Scene_text.
- [2]. **Elaine Marsh, Dennis Perzanowski.** “Named Entity Task” In: *MUC-7 Evaluation of IE Technology: Overview of Results* (1998). url: https://www-nlpir.nist.gov/related_projects/muc/proceedings/muc_7_proceedings/marsh_slides.pdf.
- [3]. **Adrian Rosebrock.** “Text Skew Correction” In: *Text skew correction with Opencv and Python* (2017). url: <https://www.pyimagesearch.com/2017/02/20/text-skew-correction-opencv-python/>.
- [4]. **Vladimir I. Levenshtein,** *Binary codes capable of correcting deletions, insertions, and reversals*, Doklady Akademii Nauk SSSR, 163(4):845-848, 1965 (Russian). English translation in Soviet Physics Doklady, 10(8):707-710, 1966.
- [5]. **Toastdriven.** “Pylev” In: (). url: <https://github.com/toastdriven/pylev>.
- [6]. **Eragonruan.** “Text-detection-ctpn” In: (). url: <https://github.com/eragonruan/text-detection-ctpn>.
- [7]. **Clova AI Research.** “CRAFT-pytorch” In: (). url: <https://github.com/clovaai/CRAFT-pytorch>.
- [8]. **Tesseract-ocr.** “tesseract” In: (). url: <https://github.com/tesseract-ocr/tesseract>.

- [9]. **Belval**. “TextRecognitionDataGenerator” In: (). url: <https://github.com/Belval/TextRecognitionDataGenerator>.
- [10]. **Adrian Rosebrock**. “LeNet” In: Image classification with Keras and deep learning (2017). url: <https://www.pyimagesearch.com/2017/12/11/image-classification-with-keras-and-deep-learning/>.
- [11]. **Karen Simonyan & Andrew Zisserman**. VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION, ICLR 2015, 2015. url: <https://arxiv.org/pdf/1409.1556/>.
- [12]. **Adrian Rosebrock**. “SmallerVGGNet” In: *Keras and Convolutional Neural Networks* (CNNs) (2018). url: <https://www.pyimagesearch.com/2018/04/16/keras-and-convolutional-neural-networks-cnns/>.
- [13]. **David Martin Ward Powers**. Evaluation: From Precision, Recall and FFactor to ROC, Informedness, Markedness & Correlation. Jan 2008.