BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM

ĐỒ ÁN CƠ SỞ

HỆ THỐNG NHẬN DIỆN VÀ TỰ ĐỘNG HÓA LƯU TRỮ THÔNG TIN HÓA ĐƠN

Ngành: **KHOA HỌC DỮ LIỆU**

Chuyên ngành: **KHOA HỌC DỮ LIỆU**

Giảng viên hướng dẫn : ThS. Nguyễn Quang Phúc

Sinh viên thực hiện :

2286400029-Hồ Gia Thành

2286400015-Huỳnh Thái Linh

2286400011-Trương Minh Khoa

Lớp: 22DKHA1

TP. Hồ Chí Minh, 2025

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ TP. HCM

ĐỒ ÁN CƠ SỞ

HỆ THỐNG NHẬN DIỆN VÀ TỰ ĐỘNG HÓA LƯU TRỮ THÔNG TIN HÓA ĐƠN

Ngành: **KHOA HỌC DỮ LIỆU**

Chuyên ngành: **KHOA HỌC DỮ LIỆU**

Giảng viên hướng dẫn : ThS. Nguyễn Quang Phúc

Sinh viên thực hiện :

2286400029-Hồ Gia Thành

2286400015-Huỳnh Thái Linh

2286400011-Trương Minh Khoa

Lớp: 22DKHA1

TP. Hồ Chí Minh, 2025

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .

TP. HCM, Ngày.....tháng.....năm 2025

Giảng viên hướng dẫn

(Ký tên, đóng dấu)

LỜI CAM ĐOAN

Chúng tôi, Hồ Gia Thành, Trương Minh Khoa và Huỳnh Thái Linh, xin cam đoan rằng:

Tất cả nội dung của bài báo cáo này là kết quả từ quá trình nghiên cứu và làm việc chung của cả ba chúng tôi. Các thông tin được trình bày trong báo cáo đều được thu thập từ những nguồn đáng tin cậy và đã được xử lý cẩn thận.

Chúng tôi đảm bảo rằng không có bất kỳ hành vi sao chép hay sử dụng thông tin không chính xác nào từ các nguồn khác. Mọi tài liệu tham khảo đã được ghi nguồn rõ ràng và tuân thủ đúng các quy định về trích dẫn học thuật.

Bài báo cáo này là sản phẩm nghiên cứu chung của chúng tôi và chưa từng được nộp hoặc công bố ở bất kỳ đâu trước đây. Chúng tôi hoàn toàn chịu trách nhiệm về tính trung thực và chính xác của nội dung báo cáo này.

Chúng tôi hy vọng rằng báo cáo này sẽ cung cấp một cái nhìn toàn diện và chi tiết về hệ thống nhận diện và tự động hóa lưu trữ thông tin hóa đơn, đồng thời góp phần nghiên cứu và phát triển các giải pháp công nghệ trong việc tối ưu hóa quy trình xử lý dữ liệu hóa đơn trong thời đại số.

TP. HCM, Ngày.....tháng.....năm 2025

Sinh viên

Hồ Gia Thành
Huỳnh Thái Linh
Trương Minh Khoa

DANH MỤC CÁC KÝ HIỆU, TỪ VIẾT TẮT VÀ TỪ KHÓA

OCR	Optical Character Recognition (Nhận diện ký tự quang học).
Tesseract	Tesseract OCR (Công cụ nhận diện ký tự quang học mã nguồn mở).
CSV	Comma-Separated Values (Giá trị phân tách bằng dấu phẩy).
OpenCV	Open Source Computer Vision Library (Thư viện thị giác máy tính mã nguồn mở).
PSM	Page Segmentation Mode (Chế độ phân đoạn trang trong Tesseract OCR).
OEM	OCR Engine Mode (Chế độ động cơ OCR trong Tesseract).
Store Name	Tên cửa hàng (Trường thông tin hóa đơn, ví dụ: Bách Hóa Xanh).
Address	Địa chỉ (Trường thông tin hóa đơn).
Date	Ngày giao dịch (Trường thông tin hóa đơn, định dạng DD/MM/YYYY).
Employee	Nhân viên (Trường thông tin hóa đơn, tên nhân viên bán hàng).
Bill no	Số hóa đơn (Trường thông tin hóa đơn, mã hóa đơn).
Products	Danh sách sản phẩm (Trường thông tin hóa đơn, bao gồm tên, số lượng, giá).
Total_products	Tổng tiền sản phẩm (Trường thông tin hóa đơn, tổng giá trị của các sản phẩm).
Paid	Số tiền thanh toán (Trường thông tin hóa đơn, số tiền khách hàng trả).
Change	Tiền thối (Trường thông tin hóa đơn, số tiền thối lại).
Discount	Giảm giá (Trường thông tin hóa đơn, giá trị giảm giá nếu có).

Mục lục

1 TỔNG QUAN **11**

1.1 Giới thiệu đề tài . . . 11
1.2 Nhiệm vụ của đồ án . . . 11
1.2.1 Tính cấp thiết của đề tài . . . 12
1.2.2 Ý nghĩa khoa học và thực tiễn của đề tài . . . 13
1.3 Mục tiêu . . . 13
1.3.1 Mục tiêu tổng quan . . . 13
1.3.2 Mục tiêu cụ thể . . . 13
1.4 Đối tượng và phạm vi . . . 14
1.4.1 Đối tượng . . . 14
1.4.2 Phạm vi . . . 14
1.5 Phương pháp nghiên cứu . . . 14
1.5.1 Phương pháp nghiên cứu sơ bộ . . . 14
1.5.2 Phương pháp nghiên cứu tài liệu . . . 14
1.5.3 Phương pháp nghiên cứu thống kê . . . 14
1.5.4 Phương pháp thực nghiệm . . . 15
1.5.5 Phương pháp đánh giá . . . 15
1.6 Những đóng góp nghiên cứu của đề tài . . . 15
1.6.1 Trong lĩnh vực học thuật . . . 15
1.6.2 Trong thực tiễn kinh doanh . . . 15

2 CƠ SỞ LÝ THUYẾT **16**

2.1 Nhận diện ký tự quang học (OCR - Optical Character Recognition) 16
2.1.1 Khái niệm và định nghĩa OCR . . . 16
2.1.2 Lịch sử phát triển . . . 17
2.1.3 Nguyên lý hoạt động của OCR . . . 18
2.1.4 Các loại OCR . . . 19
2.1.5 Các phương pháp và kỹ thuật trong OCR . . . 19
2.1.6 Ứng dụng của OCR . . . 20
2.1.7 Thách thức trong OCR . . . 20
2.2 Tesseract . . . 20
2.2.1 Khái niệm cơ bản về Tesseract . . . 20

2.2.2 Lịch Sử Phát Triển . . . 21
2.2.3 Các thành phần chính của Tesseract . . . 21
2.2.4 Nguyên lý hoạt động của Tesseract . . . 22
2.2.5 Ưu điểm và hạn chế của Tesseract . . . 22
2.2.6 Ứng dụng của Tesseract . . . 23
2.3 Xử lý ảnh (Image Processing) . . . 23
2.3.1 Vai trò của xử lý ảnh trong OCR . . . 23
2.3.2 Các bước tiền xử lý ảnh . . . 24
2.3.3 Công cụ sử dụng . . . 26
2.4 Biểu thức chính quy (Regex) . . . 26
2.4.1 Vai trò của Regex trong trích xuất thông tin . . . 26
2.4.2 Cách thức hoạt động . . . 27
2.4.3 Các ứng dụng cụ thể trong OCR . . . 27
2.4.4 Ưu điểm và hạn chế của Regex trong OCR . . . 28
2.5 Framework Flask Trong Hệ Thống OCR . . . 28
2.5.1 Tổng quan về Flask . . . 28
2.5.2 Các tính năng chính của Flask . . . 28
2.5.3 Kiến trúc của Flask trong hệ thóng OCR . . . 29
2.5.4 Ứng dụng của Flask trong hệ thống OCR . . . 30

3 PHƯƠNG PHÁP THỰC NGHIỆM 31
3.1 Giới thiệu về dữ liệu . . . 31
3.1.1 Nguồn dữ liệu . . . 31
3.1.2 Đặc Điểm Dữ Liệu . . . 31
3.1.3 Mục đích sử dụng . . . 32
3.1.4 Quy trình thu thập và xử lý sơ bộ . . . 32
3.1.5 Ý nghĩa của Dataset . . . 32
3.2 Tiền xử lý dữ liệu hình ảnh . . . 33
3.2.1 Mục đích của tiền xử lý . . . 33
3.2.2 Quy trình tiền xử lý . . . 33
3.2.3 Ý nghĩa của quy trình tiền xử lý . . . 35
3.2.4 Tổng hợp pipeline tiền xử lý . . . 35
3.2.5 Cắt ảnh thành các đoạn nhỏ để tối ưu nhận diện OCR . . . 36
3.3 Trích xuất thông tin từ văn bản OCR . . . 38
3.3.1 Quy trình trích xuất thông tin . . . 38
3.4 Mô hình OCR . . . 40
3.4.1 Cấu hình OCR ban đầu . . . 41
3.4.2 Nhận diện văn bản từ toàn bộ hình ảnh . . . 41
3.4.3 Đánh giá chất lượng kết quả . . . 41
3.4.4 Chuyển sang xử lý các đoạn ảnh nếu cần . . . 42

3.4.5 Trả về kết quả 42

4 KẾT QUẢ THỰC NGHIỆM **43**
4.1 Tổng quan về quá trình thực nghiệm 43
4.2 Kết quả nhận diện và trích xuất thông tin 43
4.2.1 Đánh giá hiệu suất OCR 43
4.2.2 Phân tích lỗi 47
4.3 Giao diện web 47
4.3.1 Trang upload hình ảnh hóa đơn 48
4.3.2 Trang hiển thị kết quả 48
4.4 Kết quả lưu trữ vào file CSV 49
4.5 Đánh giá tổng thể 50

5 KẾT LUẬN VÀ KIẾN NGHỊ **52**
5.1 Kết luận 52
5.2 Kiến nghị 52

Tài liệu tham khảo **53**

Danh sách hình vẽ

2.1 Optical Character Recognition - OCR. Nguồn [2] 16
2.2 Tesseract. Nguồn [8] 21
2.3 Grayscale Conversion. Nguồn [14] 24
2.4 Deskewing Image. Nguồn [15] 24
2.5 Noise Reduction. Nguồn [16] 25
2.6 Contrast Enhancement. Nguồn [17] 25
2.7 Scaling image. Nguồn [18] 25

4.1 Hóa đơn hoàn toàn không nhận diện được thông tin. 44
4.2 Hóa đơn nhận diện sai các trường và sai kí tự sản phẩm. 44
4.3 Hóa đơn nhận diện thiếu và không nhận diện được sản phẩm được. 45
4.4 Hóa đơn nhận diện mức trung bình vẫn còn sai một số chỗ. 45
4.5 Hóa đơn nhận diện khá tốt nhưng vẫn còn đa số chỗ sai. 46
4.6 Hóa đơn nhận diện gần tốt nhưng vẫn còn các trường không thể nhận diện. 46
4.7 Hóa đơn với lỗi nhận diện do nhiễu ký tự 47
4.8 Giao diện upload hình ảnh hóa đơn 48
4.9 Trang hiển thị kết quả trích xuất từ hóa đơn 49
4.10Nội dung file CSV chứa thông tin hóa đơn được trích xuất. 50

Danh sách đoạn mã

2.1 Định tuyến trang chủ trong Flask 29
2.2 Hiển thị trang người dùng trong Flask 29

3.1 Chuyển ảnh sang màu xám 33
3.2 Xoay ảnh với góc nghiêng 33
3.3 Loại bỏ nhiễu nền 34
3.4 Áp dụng ngưỡng Otsu cho ảnh 34
3.5 Áp dụng phương pháp hình thái học 35
3.6 Thay đổi kích thước ảnh với tỷ lệ 35
3.7 Dòng mã tiền xử lý hình ảnh 35
3.8 Dòng mã quy trình cắt ảnh 37
3.9 Biểu thức chính quy trích xuất ngày tháng 39
3.10 Biểu thức chính quy cho số lượng và giá 40
3.11 Biểu thức chính quy cho tiền tệ 40
3.12 Cấu hình tùy chỉnh cho OCR 41
3.13 Đoạn mã để trích xuất văn bản thô từ toàn bộ hình ảnh hóa đơn. 41
3.14 Kiểm tra các trường trong kết quả trích xuất từ hóa đơn 41
3.15 Kiểm tra số lượng trường None và trích xuất thông tin 42
3.16 Trả về kết quả và số lượng trường None 42

Chương 1

TỔNG QUAN

1.1 Giới thiệu đề tài

Trong thời đại công nghệ số bứt phá, khi mọi thứ đều chuyển động nhanh như một cơn gió, việc quản lý và lưu trữ thông tin hóa đơn không còn chỉ là một công việc hành chính khô khan. Nó đã trở thành một nghệ thuật, một chìa khóa để doanh nghiệp mở lối đi riêng trong thị trường đầy cạnh tranh. Hệ thống nhận diện và tự động hóa lưu trữ thông tin hóa đơn ra đời như một người bạn đồng hành thông minh, giúp doanh nghiệp "đọc hiểu" từng dòng hóa đơn, rút ra những thông tin quan trọng và sắp xếp chúng gọn gàng, khoa học. Với sự hỗ trợ của trí tuệ nhân tạo, hệ thống này không chỉ tiết kiệm thời gian mà còn mang lại sự chính xác, minh bạch, tạo nền tảng vững chắc cho các quyết định tài chính và chiến lược. Đề tài "Hệ Thống Nhận Diện và Tự Động Hóa Lưu Trữ Thông Tin Hóa Đơn" chính là một lời giải thiết thực, giúp doanh nghiệp vượt qua bài toán dữ liệu phức tạp, tối ưu hóa hoạt động và tự tin bước đi trong kỷ nguyên số.

1.2 Nhiệm vụ của đồ án

Nhiệm vụ của đồ án là chú trọng vào việc quản lý thông tin hóa đơn, mang đến một giải pháp thông minh cho doanh nghiệp trong thời đại số. Đồ án tập trung khám phá và ứng dụng các kỹ thuật học máy cùng công nghệ nhận diện hình ảnh để xây dựng một hệ thống tự động trích xuất và lưu trữ dữ liệu từ hóa đơn. Bắt đầu từ việc tìm hiểu các nền tảng lý thuyết như nhận diện ký tự quang học (OCR) và các mô hình học sâu, đồ án hướng đến một quy trình vừa chính xác vừa hiệu quả. Quan trọng hơn, đồ án còn đặt mục tiêu đánh giá hiệu suất hệ thống qua các thước đo như độ chính xác nhận diện, tốc độ xử lý và khả năng ứng dụng trong thực tiễn, từ đó đưa ra những nhận định để chọn giải pháp tốt nhất, giúp doanh nghiệp đưa ra quyết định và tối ưu hóa hoạt động kinh doanh.

1.2.1 Tính cấp thiết của đề tài

Trong thời đại công nghệ số, khi mọi giao dịch đều được số hóa với tốc độ chóng mặt, việc quản lý thông tin hóa đơn không chỉ là một nhiệm vụ hành chính mà đã trở thành một chiến lược sống còn đối với doanh nghiệp. Sự phát triển mạnh mẽ của thương mại điện tử và các quy trình kinh doanh hiện đại đã tạo ra một khối lượng dữ liệu hóa đơn khổng lồ, đòi hỏi các doanh nghiệp phải đổi mới không ngừng để theo kịp nhịp độ thị trường. Hệ thống nhận diện và tự động hóa lưu trữ thông tin hóa đơn ra đời như một giải pháp đột phá, giúp doanh nghiệp không chỉ quản lý mà còn khai thác giá trị tiềm ẩn từ những con số khô khan.

Hệ thống này, với khả năng tự động nhận diện, trích xuất và phân loại dữ liệu hóa đơn, đóng vai trò như một người trợ lý thông minh, giúp doanh nghiệp tiết kiệm thời gian, giảm thiểu sai sót và tối ưu hóa nguồn lực. Bằng cách ứng dụng các công nghệ như nhận diện ký tự quang học (OCR), học máy và học sâu, hệ thống không chỉ đơn thuần lưu trữ mà còn phân tích dữ liệu, mang lại những góc nhìn sâu sắc về tình hình tài chính và xu hướng kinh doanh. Cụ thể, hệ thống mang lại những lợi ích thiết thực như:

- Tối ưu hóa quy trình tài chính: Tự động hóa việc xử lý hóa đơn giúp doanh nghiệp giảm thiểu chi phí vận hành, từ việc nhập liệu thủ công đến kiểm tra sai sót, từ đó nâng cao hiệu quả quản lý dòng tiền.
- Cải thiện ra quyết định chiến lược: Dữ liệu hóa đơn được phân tích và lưu trữ khoa học cung cấp thông tin giá trị, giúp doanh nghiệp đưa ra các quyết định dựa trên số liệu thay vì phán đoán, từ quản lý chi phí đến lập kế hoạch kinh doanh.
- Nâng cao trải nghiệm khách hàng: Hệ thống cho phép xử lý nhanh chóng các yêu cầu liên quan đến hóa đơn, như tra cứu hay đối chiếu, mang lại sự tiện lợi và chuyên nghiệp trong mắt khách hàng.
- Tăng cường khả năng cạnh tranh: Trong một thị trường mà thời gian và độ chính xác là yếu tố quyết định, một hệ thống quản lý hóa đơn hiệu quả giúp doanh nghiệp phản ứng nhanh hơn trước các biến động, củng cố vị thế trên thị trường.

Với những giá trị vượt trội ấy, hệ thống nhận diện và tự động hóa lưu trữ thông tin hóa đơn không chỉ là một công cụ hỗ trợ mà còn là một động lực thúc đẩy doanh nghiệp vươn xa, góp phần định hình một thị trường số hóa bền vững, hiệu quả và tràn đầy tiềm năng.

1.2.2 Ý nghĩa khoa học và thực tiễn của đề tài

Ý nghĩa khoa học: Đề tài góp phần làm giàu lĩnh vực khoa học dữ liệu bằng cách ứng dụng các thuật toán tiên tiến như OCR và học sâu để nhận diện, phân tích thông tin hóa đơn. Nó không chỉ củng cố lý thuyết xử lý dữ liệu mà còn phát triển các mô hình mới, hỗ trợ nghiên cứu sâu hơn về tài chính và hành vi giao dịch, mở ra hướng đi mới cho công nghệ số hóa.

Ý nghĩa thực tiễn: Hệ thống giúp doanh nghiệp quản lý hóa đơn hiệu quả, từ trích xuất dữ liệu đến phân tích tài chính, hỗ trợ ra quyết định chính xác. Nó giảm sai sót, tăng minh bạch, tối ưu chi phí và nâng cao cạnh tranh. Giải pháp có thể áp dụng rộng rãi, từ thương mại điện tử đến các ngành khác, đồng thời cung cấp dữ liệu giá trị cho chính sách, thúc đẩy thị trường số bền vững.

1.3 Mục tiêu

1.3.1 Mục tiêu tổng quan

Đề tài hướng tới xây dựng hệ thống nhận diện và tự động hóa lưu trữ thông tin hóa đơn, giúp doanh nghiệp trích xuất, phân tích và quản lý dữ liệu hiệu quả. Sử dụng công nghệ OCR và học máy, hệ thống tối ưu hóa quy trình tài chính, hỗ trợ ra quyết định chiến lược và nâng cao năng lực cạnh tranh trong thị trường số.

1.3.2 Mục tiêu cụ thể

Với đề tài này, chúng tôi đặt mục tiêu xây dựng một hệ thống thông minh, sử dụng dữ liệu hóa đơn từ các doanh nghiệp, bao gồm thông tin giao dịch, chi tiết thanh toán và thời gian. Quá trình thực hiện bao gồm các bước: xác định yêu cầu, làm sạch dữ liệu, lựa chọn công nghệ, triển khai mô hình nhận diện, đánh giá hiệu suất và tối ưu hóa hệ thống. Cụ thể, tôi sẽ áp dụng công nghệ OCR kết hợp học sâu để trích xuất dữ liệu hóa đơn chính xác. Hiệu quả của hệ thống sẽ được đánh giá qua các chỉ số như độ chính xác nhận diện và tốc độ xử lý. Kết quả kỳ vọng là một hệ thống mượt mà, giúp doanh nghiệp quản lý hóa đơn hiệu quả và đề xuất chiến lược tài chính tối ưu.

1.4 Đối tượng và phạm vi

1.4.1 Đối tượng

Đối tượng nghiên cứu của đề tài là những hóa đơn từ các doanh nghiệp, từ nhỏ đến lớn, hoạt động trong nhiều lĩnh vực khác nhau, đặc biệt trong thương mại điện tử. Những hóa đơn này chứa đựng thông tin giao dịch, thanh toán và chi tiết tài chính, như những mảnh ghép kể câu chuyện về dòng chảy kinh doanh. Bằng cách thu thập và phân tích dữ liệu từ chúng, đề tài tìm cách khám phá các yếu tố ảnh hưởng đến hiệu quả tài chính, từ đó đưa ra những giải pháp tối ưu hóa quy trình quản lý và vận hành.

1.4.2 Phạm vi

Đề tài tập trung vào việc xây dựng và triển khai hệ thống nhận diện, trích xuất và tự động hóa lưu trữ thông tin hóa đơn. Chúng tôi sẽ áp dụng các công nghệ tiên tiến như nhận diện ký tự quang học (OCR) và học máy để xử lý dữ liệu hóa đơn, tạo ra một quy trình quản lý thông minh, chính xác. Dựa trên kết quả phân tích, đề tài sẽ đề xuất các chiến lược tài chính và vận hành phù hợp, đồng thời đánh giá hiệu quả của hệ thống. Giải pháp này không chỉ mang lại giá trị thực tiễn cho doanh nghiệp mà còn góp phần định hình một tương lai số hóa bền vững cho thị trường.

1.5 Phương pháp nghiên cứu

1.5.1 Phương pháp nghiên cứu sơ bộ

Chúng tôi tìm hiểu về quy trình quản lý hóa đơn, các yếu tố ảnh hưởng đến hiệu quả tài chính, và các công nghệ như OCR, học máy. Qua đó, xác định thách thức cần giải quyết và vạch ra lộ trình nghiên cứu phù hợp.

1.5.2 Phương pháp nghiên cứu tài liệu

Chúng tôi nghiên cứu tài liệu từ các nghiên cứu khoa học và bài báo công nghệ để thu thập tri thức về các phương pháp nhận diện và phân tích dữ liệu. Dựa vào đó, lựa chọn các kỹ thuật học máy và OCR tối ưu cho hệ thống nhận diện hóa đơn.

1.5.3 Phương pháp nghiên cứu thống kê

Chúng tôi sử dụng các công cụ thống kê để phân tích và đánh giá hiệu quả hệ thống nhận diện. Phương pháp bao gồm thống kê mô tả và phân tích tỷ lệ lỗi nhận diện để đánh giá độ chính xác của OCR và mô hình học máy.

1.5.4 Phương pháp thực nghiệm

Dữ liệu hóa đơn,được thu thập từ các cá nhân thành viên trong nhóm và kho dữ liệu Roboflow[1], trở thành "phòng thí nghiệm" của chúng tôi. Chúng tôi tiền xử lý dữ liệu, áp dụng các mô hình học máy để nhận diện và phân loại thông tin, rồi kiểm chứng hiệu quả của hệ thống. Thực nghiệm này đảm bảo hệ thống không chỉ lý thuyết mà còn khả thi trong thực tiễn.

1.5.5 Phương pháp đánh giá

Cuối cùng, chúng tôi đo lường hiệu quả hệ thống qua các chỉ số như độ chính xác nhận diện, tốc độ xử lý và khả năng tích hợp. Bằng cách so sánh các kết quả, chúng tôi đánh giá tính hiệu quả của hệ thống, đảm bảo nó mang lại giá trị thực tiễn cho doanh nghiệp trong việc quản lý hóa đơn.

1.6 Những đóng góp nghiên cứu của đề tài

1.6.1 Trong lĩnh vực học thuật

Đề tài này như một ngọn lửa soi sáng góc nhỏ của khoa học dữ liệu, mang đến cách tiếp cận mới mẻ trong việc xử lý thông tin hóa đơn. Bằng cách ứng dụng công nghệ nhận diện ký tự quang học (OCR) và các mô hình học sâu, đề tài không chỉ làm giàu lý thuyết về phân tích dữ liệu mà còn xây dựng những mô hình nhận diện tối ưu, mở đường cho các nghiên cứu tương lai. Hơn nữa, việc tổng hợp và xử lý tập dữ liệu hóa đơn đa dạng, từ nguồn cá nhân và Roboflow, tạo nên một kho tàng quý giá, phục vụ các nhà khoa học khám phá sâu hơn về quản lý tài chính và trí tuệ nhân tạo.

1.6.2 Trong thực tiễn kinh doanh

Như một người dẫn đường tận tụy, đề tài mang đến cho doanh nghiệp một công cụ sắc bén để quản lý hóa đơn hiệu quả. Hệ thống nhận diện và tự động hóa giúp doanh nghiệp "giải mã" dữ liệu hóa đơn, từ đó hiểu rõ hơn về dòng tiền và xu hướng tài chính. Các giải pháp đề xuất, từ tối ưu hóa quy trình đến cải thiện chiến lược tài chính, không chỉ nâng cao hiệu suất vận hành mà còn giúp doanh nghiệp vững vàng hơn trong cuộc đua thị trường số. Với tính ứng dụng cao, hệ thống này hứa hẹn thúc đẩy sự tăng trưởng bền vững, mang lại lợi thế cạnh tranh trong kỷ nguyên thương mại điện tử bùng nổ.

[1]Roboflow là nền tảng hỗ trợ phát triển mô hình học máy, đặc biệt trong nhận diện hình ảnh, cung cấp các công cụ để thu thập dữ liệu, gán nhãn, và huấn luyện mô hình.

Chương 2

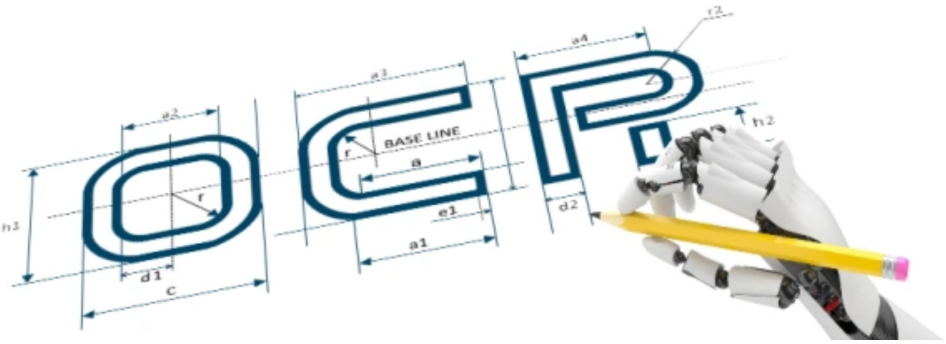
CƠ SỞ LÝ THUYẾT

2.1 Nhận diện ký tự quang học (OCR - Optical Character Recognition)

2.1.1 Khái niệm và định nghĩa OCR

Nhận diện Ký tự quang học (Optical Character Recognition - OCR) [1] là một công nghệ sử dụng các thuật toán máy tính để nhận diện và chuyển đổi văn bản có sẵn trong hình ảnh hoặc tài liệu quét thành văn bản có thể chỉnh sửa, tìm kiếm hoặc phân tích được. Công nghệ này cho phép các hệ thống máy tính đọc và hiểu văn bản từ các hình ảnh quét hoặc tài liệu văn bản viết tay hoặc in ấn, sau đó chuyển đổi chúng thành dạng kỹ thuật số. OCR là một công cụ quan trọng trong các ứng dụng như quét tài liệu, số hóa văn bản, tìm kiếm thông tin và tự động hóa các quy trình văn phòng.

OCR không chỉ giới hạn ở việc nhận diện chữ viết in mà còn có thể xử lý chữ viết tay trong một số trường hợp, tùy thuộc vào độ phức tạp của văn bản và chất lượng hình ảnh đầu vào. Với sự phát triển của trí tuệ nhân tạo, OCR ngày nay đã trở nên chính xác hơn, có khả năng xử lý nhiều ngôn ngữ và định dạng văn bản đa dạng.



Hình 2.1: Optical Character Recognition - OCR. Nguồn [2]

2.1.2 Lịch sử phát triển

Công nghệ OCR là kết quả của hơn 100 năm phát triển, từ máy móc cơ khí đơn giản đến hệ thống AI hiện đại, trải qua 4 giai đoạn chính với các bước tiến công nghệ quan trọng.[3]

1. Tiền OCR (Trước 1914): Những Ý Tưởng Đầu Tiên

Các thiết bị cơ học sơ khai xuất hiện từ thế kỷ 18–19, nhưng chỉ đọc được font chữ khắc sẵn. **Năm 1914, Emanuel Goldberg**[4] phát minh thiết bị quét văn bản đầu tiên, đặt nền móng cho OCR.

2. Thời Kỳ Sơ Khai (1920–1970): Từ Bằng Sáng Chế Đến Ứng Dụng Thực Tế

- **1929: Gustav Tauschek** (Áo) nhận bằng sáng chế máy OCR cơ học.
- **1950s: IBM và RCA** phát triển hệ thống đọc số trên séc, ứng dụng trong ngân hàng.
- **1965: David H.Shepard** (Mỹ) giới thiệu **"Gismo"** – phần mềm OCR đầu tiên có thể nhận dạng nhiều font chữ.

3. Bùng Nổ Kỷ Nguyên Số (1970–2000): OCR Trở Thành Công Cụ Toàn Cầu

- **1974: Ray Kurzweil** Chế tạo máy đọc sách cho người khiếm thị, kết hợp OCR và giọng nói.
- **1980s–1990s:** Phần mềm như **Caere OmniPage** và **Abbyy FineReader** xuất hiện, cho phép scan sách, tài liệu với độ chính xác cao. Các thuật toán **mạng nơ-ron** bắt đầu được thử nghiệm để nhận dạng chữ viết tay.
- **1996:** HP phát hành **Tesseract** – nền tảng OCR mã nguồn mở tiền thân của Google Tesseract.

4. Kỷ Nguyên AI (2000–Nay): OCR Thông Minh Và Đa Năng

- **2006:** Google tích hợp OCR vào **Google Books**, số hóa hàng triệu đầu sách.
- **2010s: Tesseract 4.0** áp dụng **LSTM** (Long Short-Term Memory), tăng độ chính xác lên hơn 95%. Các nền tảng đám mây như **Google Vision API, Azure OCR** ra đời, xử lý ảnh trực tuyến chỉ trong vài giây.
- **2020s:** OCR kết hợp AI đa phương thức (GPT-4 Vision), xử lý được chữ viết tay, video.[5]

2.1.3 Nguyên lý hoạt động của OCR

Quá trình nhận dạng ký tự trong OCR có thể được chia thành một số bước cơ bản:[6]

1. Thu nhận hình ảnh

Hình ảnh chứa văn bản được thu thập từ nhiều nguồn khác nhau như:

- Máy quét (scanner) tài liệu giấy.
- Ảnh chụp từ điện thoại, máy ảnh.
- File PDF hoặc hình ảnh kỹ thuật số (JPG, PNG, v.v.).

Chất lượng ảnh đầu vào ảnh hưởng lớn đến độ chính xác của OCR, nên ưu tiên ảnh rõ nét, độ phân giải cao và ít nhiễu.

2. Xử lý hình ảnh (Preprocessing)

Giai đoạn này nhằm cải thiện chất lượng hình ảnh để tăng độ chính xác của việc nhận diện. Các kỹ thuật thường được sử dụng bao gồm:

- **Chuyển đổi màu sắc:** Chuyển hình ảnh sang thang độ xám hoặc nhị phân hóa (black-and-white) để làm nổi bật văn bản.
- **Loại bỏ nhiễu:** Áp dụng các bộ lọc để xóa bỏ các điểm nhiễu, vết bẩn hoặc các chi tiết không liên quan trên hình ảnh.
- **Cân chỉnh hình ảnh:** Điều chỉnh góc nghiêng hoặc xoay hình ảnh để văn bản được căn chỉnh đúng hướng.
- **Tăng cường độ tương phản:** Làm rõ sự khác biệt giữa văn bản và nền để dễ dàng nhận diện ký tự.

3. Nhận Dạng Ký Tự (Character Recognition)

Đây là bước cốt lõi của OCR, nơi các thuật toán được sử dụng để xác định từng ký tự. Có hai phương pháp chính:

- **Nhận diện dựa trên mẫu (Pattern Matching):** So sánh các ký tự trong hình ảnh với cơ sở dữ liệu mẫu ký tự đã được định nghĩa trước. Phương pháp này hiệu quả với các phông chữ cố định nhưng kém linh hoạt với chữ viết tay hoặc phông chữ không chuẩn.
- **Nhận diện dựa trên đặc trưng (Feature Extraction):** Trích xuất các đặc điểm như đường nét, góc cạnh, hoặc hình dạng của ký tự, sau đó sử dụng các mô hình học máy (như mạng nơ-ron) để phân loại. Phương pháp này hiện đại hơn và phù hợp với các tình huống phức tạp.

4. Xử lý hậu kỳ (Post-processing)

Sau khi ký tự được nhận diện, các kết quả có thể bị lỗi do nhiễu hoặc ký tự khó đọc. Hệ thống sẽ áp dụng các thuật toán kiểm tra chính tả và ngữ pháp để đảm bảo kết quả nhận dạng chính xác nhất có thể.

5. Xuất kết quả

Sau khi nhận dạng, văn bản sẽ được chuyển đổi thành một định dạng có thể chỉnh sửa được, chẳng hạn như .txt, .csv hoặc .json.

2.1.4 Các loại OCR

OCR có thể được phân loại thành các loại sau:[6]

- **OCR (Optical Character Recognition):** Nhận dạng ký tự in hoặc viết tay từ hình ảnh.
- **ICR (Intelligent Character Recognition):** Nhận dạng chữ viết tay (thường phức tạp hơn OCR).
- **OMR (Optical Mark Recognition):** Nhận dạng dấu tích, ô đánh dấu (ví dụ: phiếu trắc nghiệm).
- **MICR (Magnetic Ink Character Recognition):** Nhận dạng ký tự in bằng mực từ tính (thường dùng trong ngân hàng, như số trên séc).

2.1.5 Các phương pháp và kỹ thuật trong OCR

Phương pháp truyền thống

- **So khớp mẫu:** So sánh ký tự trong hình ảnh với các mẫu ký tự đã lưu.
- **Trích xuất đặc trưng:** Phân tích các đặc điểm hình học của ký tự (đường nét, góc, vòng cung).

Phương pháp hiện đại

- **Học máy (Machine Learning):** Sử dụng các thuật toán như SVM, Random Forest để phân loại ký tự.
- **Học sâu (Deep Learning):** Áp dụng mạng nơ-ron tích chập (CNN) và mạng nơ-ron tái phát (RNN) như LSTM để nhận diện văn bản trong các ngữ cảnh phức tạp.
- **Mô hình Transformer:** Các mô hình như Tesseract 4.0 hoặc OCR dựa trên Vision Transformer (ViT) cải thiện khả năng nhận diện văn bản đa dạng.

2.1.6 Ứng dụng của OCR

OCR được ứng dụng rộng rãi trong nhiều lĩnh vực, bao gồm:[6]

- **Số hóa tài liệu**: Chuyển đổi sách, báo hoặc tài liệu giấy thành định dạng kỹ thuật số.
- **Tự động hóa nhập liệu:** Nhập dữ liệu từ hóa đơn, biểu mẫu hoặc thẻ ID mà không cần nhập tay.
- **Xử lý dữ liệu lớn:** Phân tích và trích xuất thông tin từ các tài liệu không có cấu trúc.
- **Xử Lý Hóa Đơn và Biên Lai:** OCR được sử dụng để tự động hóa quá trình xử lý các hóa đơn và biên lai, giảm thiểu lỗi và tăng tốc độ công việc.

2.1.7 Thách thức trong OCR

Dù OCR là một công nghệ mạnh mẽ, nhưng nó vẫn đối mặt với một số thách thức lớn:

- **Chất Lượng Hình Ảnh:** Nếu hình ảnh quét bị mờ, bị nhiễu, hoặc không đồng nhất, hệ thống OCR sẽ gặp khó khăn trong việc nhận diện ký tự.
- **Văn Bản Viết Tay:** Văn bản viết tay có thể rất khó nhận diện, đặc biệt khi người viết có chữ viết không rõ ràng.
- **Ký Tự Đặc Biệt và Phông Chữ Phức Tạp:** Những phông chữ phức tạp hoặc ký tự đặc biệt (như trong các ngôn ngữ có dấu) cũng có thể gây khó khăn cho các hệ thống OCR.

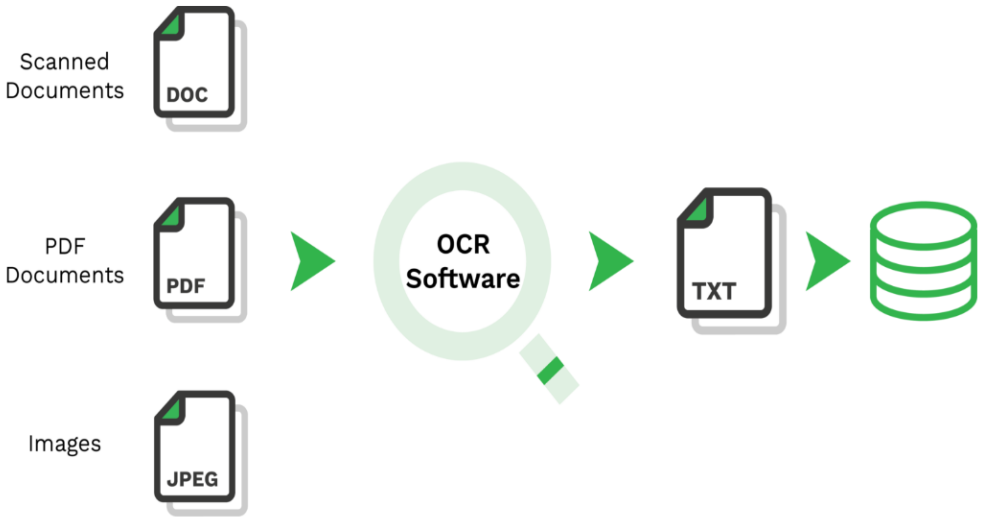
2.2 Tesseract

2.2.1 Khái niệm cơ bản về Tesseract

Tesseract là gì?: Tesseract là một công cụ OCR mã nguồn mở, được phát triển ban đầu bởi HP[1] (1985-1994) và hiện được duy trì bởi Google. Nó nhận dạng văn bản từ hình ảnh số hóa, hỗ trợ nhiều ngôn ngữ và định dạng hình ảnh.[7]

Mục đích: Chuyển đổi hình ảnh chứa văn bản (PDF, ảnh chụp, tài liệu quét) thành văn bản có thể chỉnh sửa và tìm kiếm được.

[1]Hewlett-Packard (HP) là công ty công nghệ đa quốc gia của Mỹ, phát triển Tesseract như một dự án nội bộ trước khi chuyển giao mã nguồn cho Google vào năm 2005. Xem thêm: Smith, R. (2007). "An Overview of the Tesseract OCR Engine.



Hình 2.2: Tesseract. Nguồn [8]

2.2.2 Lịch Sử Phát Triển

Ban đầu, Tesseract được phát triển bởi HP Labs vào năm 1985 và đã trở thành công cụ OCR đầu tiên có khả năng nhận diện chữ viết tay và in máy với độ chính xác cao. Sau khi mã nguồn của nó được Google phát hành dưới giấy phép mã nguồn mở vào năm 2006 [9], Tesseract đã tiếp tục phát triển mạnh mẽ và hiện nay là một trong những công cụ OCR phổ biến nhất. Với sự hỗ trợ từ cộng đồng mã nguồn mở, Tesseract không ngừng cải tiến và nâng cao khả năng nhận diện văn bản từ hình ảnh có chất lượng khác nhau, kể cả các tài liệu in bị mờ hoặc bị nhiễu.[10]

2.2.3 Các thành phần chính của Tesseract

Tesseract bao gồm các thành phần cốt lõi sau:

- **Thư viện lõi (Tesseract Engine):** Động cơ chính thực hiện các bước nhận dạng và xử lý văn bản.
- **Dữ liệu huấn luyện (Trained Data):** Tập hợp các tệp dữ liệu ngôn ngữ được sử dụng để nhận diện ký tự và ngữ cảnh. Người dùng có thể tùy chỉnh hoặc huấn luyện thêm dữ liệu để hỗ trợ các ngôn ngữ hoặc phông chữ đặc thù.
- **Giao diện dòng lệnh và API:** Tesseract cung cấp giao diện dòng lệnh để sử dụng trực tiếp và API (như Tesseract OCR API) để tích hợp vào các ứng dụng phần mềm.
- **Công cụ hỗ trợ:** Các công cụ như jTessBoxEditor hỗ trợ người dùng trong việc huấn luyện và tinh chỉnh dữ liệu OCR. [7]

2.2.4 Nguyên lý hoạt động của Tesseract

Tesseract thực hiện OCR thông qua các bước chính sau:

1. Xử lý hình ảnh (Preprocessing):

- **Nhị phân hóa:** Chuyển đổi hình ảnh thành đen trắng để tăng độ tương phản, giảm nhiễu. Phương pháp thường dùng là ngưỡng hóa thích nghi (adaptive thresholding).
- **Loại bỏ nhiễu:** Làm mịn hình ảnh, loại bỏ các điểm nhiễu hoặc đường kẻ không cần thiết.
- **Chuẩn hóa:** Điều chỉnh kích thước, độ nghiêng, hoặc xoay hình ảnh để chuẩn bị cho bước nhận dạng.

2. Phân tích bố cục (Layout Analysis)

- **Phân đoạn trang:** Xác định các vùng chứa văn bản, hình ảnh, bảng biểu.
- **Phân đoạn dòng và từ:** Chia hình ảnh thành các dòng văn bản, sau đó chia thành các từ riêng lẻ.

3. Nhận Dạng Ký Tự (Character Recognition)

- Sử dụng các mô hình học máy (trước đây là các thuật toán truyền thống, hiện nay tích hợp mạng nơ-ron) để nhận diện từng ký tự.
- Tesseract 4.0 trở lên sử dụng mạng nơ-ron LSTM (Long Short-Term Memory) để cải thiện độ chính xác, đặc biệt với văn bản phức tạp hoặc nhiều ngôn ngữ.

4. Xử lý hậu kỳ (Post-processing)

- Sử dụng từ điển và ngữ cảnh ngôn ngữ để sửa lỗi chính tả.
- Tinh chỉnh kết quả dựa trên các quy tắc ngữ pháp hoặc mẫu ngôn ngữ cụ thể.[11]

2.2.5 Ưu điểm và hạn chế của Tesseract

Ưu điểm:

- **Mã nguồn mở & miễn phí:** Dễ dàng tích hợp vào các dự án mà không lo về chi phí bản quyền.
- **Hỗ trợ đa ngôn ngữ:** Bao gồm tiếng Việt, tiếng Anh, Trung và hơn 100 ngôn ngữ khác.

- **Khả năng tùy chỉnh cao:** Cho phép huấn luyện lại mô hình (fine-tuning) để tối ưu cho các trường hợp đặc thù.
- **Khả năng tích hợp:** Tesseract có thể được sử dụng trong nhiều nền tảng từ ứng dụng web, di động đến hệ thống nhúng.

Hạn chế:

- **Phụ thuộc vào chất lượng hình ảnh:** Tesseract hoạt động kém hiệu quả với hình ảnh có độ phân giải thấp, nhiễu hoặc ánh sáng không đồng đều.
- **Khả năng nhận dạng chữ viết tay hạn chế:** Tesseract được tối ưu cho việc nhận dạng chữ in, do đó khả năng nhận diện chữ viết tay còn hạn chế.
- **Tài nguyên tính toán:** Phiên bản sử dụng mô hình LSTM (Long Short-Term Memory) tuy mạnh nhưng tốn nhiều CPU/GPU.

2.2.6 Ứng dụng của Tesseract

Tesseract được sử dụng rộng rãi trong nhiều lĩnh vực, bao gồm:

- **Số hóa tài liệu:** Chuyển đổi sách, báo, hóa đơn hoặc tài liệu giấy thành định dạng văn bản số.
- **Tự động hóa quy trình:** Ứng dụng trong các hệ thống quản lý tài liệu, như trích xuất thông tin từ hóa đơn hoặc biểu mẫu.
- **Phân tích hình ảnh:** Hỗ trợ trích xuất văn bản từ ảnh chụp, biển báo hoặc giao diện người dùng trong các ứng dụng AI.
- **Hỗ trợ người khuyết tật:** Chuyển đổi văn bản trong hình ảnh thành âm thanh hoặc định dạng dễ tiếp cận.

2.3 Xử lý ảnh (Image Processing)

2.3.1 Vai trò của xử lý ảnh trong OCR

Xử lý ảnh là bước nền tảng trong công nghệ Nhận diện ký tự quang học (OCR), đảm bảo chất lượng đầu vào cho quá trình nhận diện văn bản. Tiền xử lý ảnh giúp cải thiện độ chính xác của OCR bằng cách tối ưu hóa hình ảnh đầu vào, làm nổi bật văn bản và giảm thiểu các yếu tố gây nhiễu. Các tác vụ chính bao gồm chuyển đổi ảnh sang định dạng phù hợp, loại bỏ nhiễu, điều chỉnh góc nghiêng (deskew), và cải thiện độ tương phản, giúp văn bản dễ đọc hơn cho các thuật toán OCR.[12]

2.3.2 Các bước tiền xử lý ảnh

Tiền xử lý ảnh trong OCR bao gồm một chuỗi các bước nhằm chuẩn hóa và tối ưu hóa hình ảnh. Các bước chính bao gồm:[13]

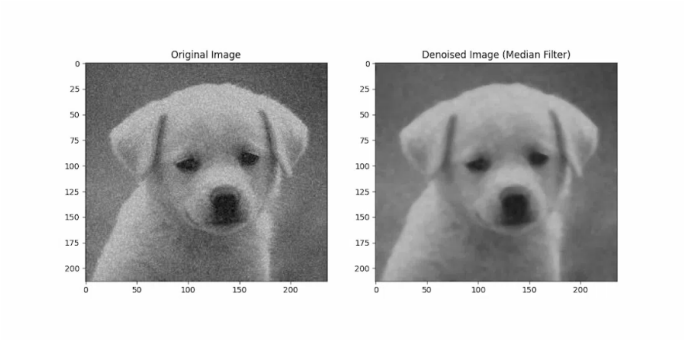
1. Chuyển đổi sang ảnh xám (Grayscale Conversion): Chuyển đổi ảnh màu sang ảnh xám giúp giảm độ phức tạp của dữ liệu, loại bỏ thông tin màu không cần thiết và tăng tốc độ xử lý. Phương pháp phổ biến sử dụng công thức tính giá trị độ sáng (luminance) từ các kênh màu RGB.

Hình 2.3: Grayscale Conversion. Nguồn [14]

2. Xoay ảnh (Deskewing): Ảnh văn bản có thể bị nghiêng do quá trình quét hoặc chụp không chính xác. Deskewing sử dụng các thuật toán như Hough Transform để xác định góc nghiêng của văn bản và xoay ảnh về vị trí chuẩn, đảm bảo văn bản nằm ngang, hỗ trợ OCR nhận diện chính xác hơn.

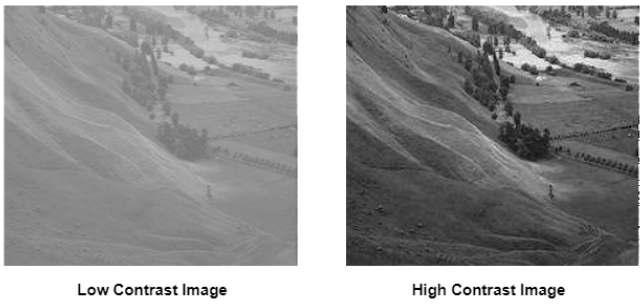
Hình 2.4: Deskewing Image. Nguồn [15]

3. Lọc bỏ nhiễu (Noise Reduction): Nhiễu trong ảnh (do chất lượng thấp, bụi bẩn, hoặc ánh sáng không đồng đều) có thể làm giảm hiệu suất OCR. Các kỹ thuật lọc như Gaussian Blur, Median Filter, hoặc các phép toán hình thái học (morphological operations) như mở (erosion) và đóng (dilation) được sử dụng để loại bỏ nhiễu và làm mịn ảnh.



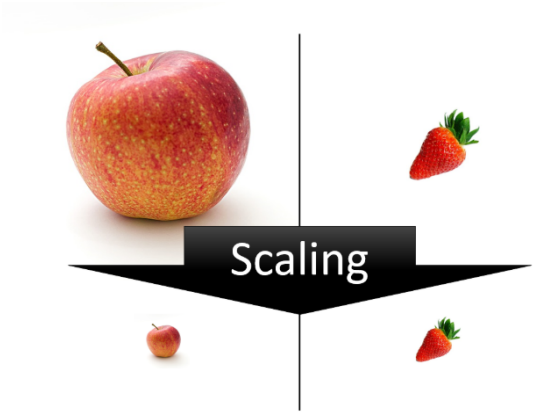
Hình 2.5: Noise Reduction. Nguồn [16]

4. Cải thiện độ tương phản (Contrast Enhancement): Độ tương phản thấp có thể khiến văn bản khó phân biệt với nền. Các kỹ thuật như Histogram Equalization hoặc Adaptive Thresholding được sử dụng để tăng cường độ tương phản, làm nổi bật văn bản.



Hình 2.6: Contrast Enhancement. Nguồn [17]

5. Thay đổi tỷ lệ ảnh (Scaling): Điều chỉnh kích thước ảnh để chuẩn hóa độ phân giải, phù hợp với thuật toán OCR. Thu nhỏ (downscaling) hoặc phóng to (upscaling) sử dụng nội suy (Bilinear, Bicubic,...) để giữ chất lượng văn bản, giảm thời gian xử lý và tăng hiệu suất OCR.



Hình 2.7: Scaling image. Nguồn [18]

2.3.3 Công cụ sử dụng

Để thực hiện các bước tiền xử lý ảnh trong OCR, một số thư viện và công cụ phổ biến được sử dụng, bao gồm:

- **OpenCV:** OpenCV (Open Source Computer Vision Library) [19] là một thư viện mã nguồn mở mạnh mẽ, được sử dụng rộng rãi trong xử lý ảnh và thị giác máy tính. Hỗ trợ nhiều ngôn ngữ lập trình (Python, C++, Java). OpenCV cung cấp các hàm hỗ trợ hầu hết các bước tiền xử lý ảnh trong OCR, bao gồm:
  - Chuyển đổi màu sắc (RGB sang grayscale, nhị phân hóa).
  - Lọc nhiễu (Gaussian, median, bilateral filtering).
  - Xoay và căn chỉnh ảnh (sử dụng Hough Transform hoặc các ma trận biến đổi hình học).
  - Thao tác hình thái học (erosion, dilation, opening, closing).
  - Phân đoạn văn bản thông qua các thuật toán như tìm contour hoặc phân tích thành phần liên kết.
- **Tesseract OCR:** Mặc dù Tesseract chủ yếu là một công cụ OCR, nó cũng tích hợp một số chức năng tiền xử lý ảnh cơ bản như nhị phân hóa và lọc nhiễu. Tuy nhiên, để đạt hiệu quả cao, Tesseract thường được kết hợp với OpenCV để xử lý ảnh phức tạp hơn trước khi nhận diện văn bản.[7]
- **Deskew:** Thư viện deskew giúp xác định góc nghiêng của văn bản và tự động điều chỉnh lại ảnh. Các thuật toán trong thư viện này giúp nhận diện các góc nghiêng và tiến hành quay ảnh để văn bản trở nên thẳng đứng, giúp quá trình nhận diện văn bản bằng OCR trở nên hiệu quả hơn.
- **Pillow (PIL):** Pillow là một thư viện Python hỗ trợ xử lý ảnh, thường được sử dụng để thực hiện các tác vụ như thay đổi kích thước, cắt ảnh, hoặc chuyển đổi định dạng, bổ trợ cho quá trình tiền xử lý trước khi áp dụng OCR.[20]

2.4 Biểu thức chính quy (Regex)

2.4.1 Vai trò của Regex trong trích xuất thông tin

Biểu thức chính quy (Regex) là công cụ mạnh mẽ để tìm kiếm, phân tích và trích xuất thông tin từ văn bản. Trong hệ thống OCR, Regex được sử dụng để xác định và trích xuất các trường thông tin cụ thể từ dữ liệu văn bản không cấu trúc, chẳng hạn như tên cửa hàng, địa chỉ, ngày tháng,

số điện thoại, mã sản phẩm hoặc giá trị hóa đơn. Việc ứng dụng Regex giúp tăng độ chính xác và tự động hóa quá trình xử lý dữ liệu từ hình ảnh quét.[21]

2.4.2 Cách thức hoạt động

Regex hoạt động bằng cách định nghĩa các mẫu (pattern) văn bản dựa trên cú pháp đặc biệt. Các mẫu này được sử dụng để khớp (match) với các chuỗi ký tự trong văn bản OCR, từ đó trích xuất hoặc xác thực thông tin. Các thành phần chính của Regex bao gồm:[21]

- **Ký tự đặc biệt:** Các ký tự như `\d` (số), `\w` (ký tự chữ), `\s` (khoảng trắng) được dùng để xác định loại ký tự cần tìm.
- **Bộ lặp (Quantifiers):** Các ký hiệu như `*`, `+`, `{n,m}` chỉ định số lần lặp lại của một ký tự hoặc nhóm ký tự.
- **Nhóm (Groups) và lớp ký tự (Character Classes):** Sử dụng `( )` để nhóm các mẫu và `[ ]` để định nghĩa tập hợp ký tự (ví dụ: `[0-9]` cho số từ 0-9).
- **Mỏ neo (Anchors):** Các ký hiệu như `^` (bắt đầu chuỗi) và `$` (kết thúc chuỗi) xác định vị trí khớp trong văn bản.

Ví Dụ: Để trích xuất ngày tháng dạng DD/MM/YYYY, mẫu Regex có thể là (`\d{2}/\d{2}/\d{4}`). Trong OCR, Regex giúp xử lý văn bản thô, nhận diện các mẫu như số điện thoại `\d{10}`, số tiền (`\d+\.\d{2}`), hoặc email (`[\w\.-]+\@[\w\.-]+\.\w+`).

2.4.3 Các ứng dụng cụ thể trong OCR

- **Xác thực dữ liệu:** Regex kiểm tra tính hợp lệ của thông tin trích xuất, ví dụ: đảm bảo ngày tháng đúng định dạng hoặc mã sản phẩm tuân theo quy tắc cụ thể.
- **Trích xuất dữ liệu từ các biểu mẫu giấy:** Regex có thể được sử dụng để nhận diện và trích xuất các trường dữ liệu từ các biểu mẫu giấy quét, chẳng hạn như hóa đơn, đơn đặt hàng, hoặc thẻ tín dụng.
- **Tìm kiếm và lọc thông tin từ văn bản quét:** Regex giúp nhận diện các thông tin quan trọng như tên người, địa chỉ, và các số liệu tài chính từ các văn bản quét.
- **Chuẩn hóa dữ liệu:** Loại bỏ hoặc thay thế các ký tự không mong muốn (như dấu cách thừa) để chuẩn hóa văn bản trước khi lưu trữ hoặc xử lý tiếp.[22]

2.4.4 Ưu điểm và hạn chế của Regex trong OCR

Ưu điểm:

- Regex cho phép xác định các mẫu cụ thể và trích xuất thông tin nhanh chóng, chính xác.
- Cải thiện hiệu quả tự động hóa trong xử lý văn bản không có cấu trúc, giảm thiểu công sức và sai sót của con người.
- Được sử dụng rộng rãi trong các hệ thống OCR để nâng cao độ chính xác của dữ liệu trích xuất.

Hạn chế:

- Cần có kỹ năng và hiểu biết vững về các biểu thức chính quy để xây dựng mẫu phù hợp.
- Các mẫu Regex có thể trở nên phức tạp và khó bảo trì khi xử lý với dữ liệu có định dạng không đồng nhất.
- Không thể xử lý được tất cả các tình huống trong văn bản phức tạp, vì vậy cần kết hợp với các kỹ thuật khác như học máy hoặc xử lý ngôn ngữ tự nhiên (NLP).

2.5 Framework Flask Trong Hệ Thống OCR

2.5.1 Tổng quan về Flask

Flask là một framework phát triển ứng dụng web viết bằng ngôn ngữ Python, được thiết kế để đơn giản và linh hoạt. Flask được phát triển bởi Armin Ronacher vào năm 2010 và hiện nay là một trong những framework phổ biến trong cộng đồng lập trình Python. Flask được biết đến với tính dễ sử dụng và khả năng mở rộng mạnh mẽ, làm cho nó trở thành lựa chọn lý tưởng cho các dự án web nhỏ đến vừa.

Điểm đặc biệt của Flask là nó thuộc loại **microframework**, tức là không bao gồm các thành phần phức tạp và không cần thiết cho mọi dự án. Flask chỉ cung cấp những tính năng cơ bản nhất để xây dựng một ứng dụng web, đồng thời cho phép lập trình viên tự do lựa chọn và tích hợp thêm các thư viện hoặc công cụ bên ngoài nếu cần thiết.[23]

2.5.2 Các tính năng chính của Flask

Flask cung cấp một số tính năng quan trọng giúp các lập trình viên phát triển ứng dụng web một cách hiệu quả, bao gồm:

1. Routing (Định tuyến): Flask cho phép định nghĩa các URL route dễ dàng, giúp các yêu cầu HTTP được chuyển đến các hàm xử lý tương ứng. Các route có thể được định nghĩa với các phương thức HTTP khác nhau như GET, POST, PUT, DELETE,...

Ví dụ:

```
@app.route('/home', methods=['GET'])
def home():
    return 'Welcome to the homepage!'
```

Đoạn mã 2.1: Định tuyến trang chủ trong Flask

2. Template Engine (Công cụ mẫu): Flask sử dụng Jinja2 làm công cụ template, giúp tách biệt mã nguồn và giao diện. Jinja2 cho phép bạn dễ dàng tạo ra các trang HTML động bằng cách kết hợp các biến và logic vào trong các mẫu.

Ví dụ:

```
from flask import render_template

@app.route('/user/<name>')
def user(name):
    return render_template('user.html', name=name)
```

Đoạn mã 2.2: Hiển thị trang người dùng trong Flask

3. Request và Response (Yêu cầu và Phản hồi): Flask hỗ trợ việc xử lý các yêu cầu HTTP từ client và trả về các phản hồi tương ứng. Các đối tượng **request** và **response** trong Flask giúp dễ dàng làm việc với dữ liệu từ client như tham số URL, dữ liệu form, hoặc cookies.

4. Flask Extensions (Mở rộng Flask): Flask có một hệ sinh thái các thư viện mở rộng phong phú, hỗ trợ các tính năng như cơ sở dữ liệu (SQLAlchemy, MongoDB), xác thực người dùng (Flask-Login, Flask-Security), và nhiều tính năng khác. Điều này giúp tăng khả năng mở rộng và tùy chỉnh của ứng dụng.

5. Debugging và Error Handling (Gỡ lỗi và xử lý lỗi): Flask cung cấp một chế độ gỡ lỗi mạnh mẽ, giúp lập trình viên dễ dàng xác định và khắc phục lỗi trong ứng dụng. Nó cũng hỗ trợ các mã lỗi HTTP tiêu chuẩn (như 404, 500), và cho phép bạn tùy chỉnh các trang lỗi nếu cần.

2.5.3 Kiến trúc của Flask trong hệ thống OCR

Kiến trúc ứng dụng Flask trong hệ thống nhận diện thông tin hóa đơn thường bao gồm các thành phần chính:[24]

1. Tầng giao diện người dùng

- Flask cung cấp các tuyến đường (routes) để người dùng tải hình ảnh hóa đơn lên thông qua giao diện web.
- Sử dụng Jinja2 để hiển thị kết quả OCR (ví dụ: thông tin hóa đơn được trích xuất) trên trình duyệt.

2. Tầng xử lý logic:

- Flask gọi các hàm xử lý hình ảnh (ví dụ: tiền xử lý, OCR) được viết bằng Python, sử dụng các thư viện như OpenCV và Pytesseract.
- Kết quả OCR được cấu trúc hóa thành các trường thông tin (như số hóa đơn, ngày, tổng tiền) và lưu vào tệp CSV.

3. Tầng lưu trữ dữ liệu:

- Flask tích hợp với Pandas để ghi dữ liệu trích xuất vào tệp CSV.
- Hỗ trợ lưu trữ tệp hình ảnh hoặc kết quả trên hệ thống tệp hoặc dịch vụ đám mây.

4. Tầng giao tiếp:

- Flask xử lý các yêu cầu HTTP (POST để nhận hình ảnh, GET để trả về kết quả).
- Đảm bảo bảo mật thông qua kiểm tra định dạng tệp và xác thực người dùng (nếu cần).

2.5.4 Ứng dụng của Flask trong hệ thống OCR

Trong bối cảnh hệ thống nhận diện thông tin hóa đơn, Flask được ứng dụng như sau:

- **Xây dựng API xử lý hình ảnh:** Flask cung cấp các endpoint API để nhận hình ảnh hóa đơn từ người dùng, thực hiện OCR, và trả về dữ liệu trích xuất dưới dạng JSON hoặc hiển thị trên giao diện web.

- **Tích hợp OCR:** Flask gọi các thư viện OCR như Tesseract để nhận diện văn bản từ hình ảnh hóa đơn, sau đó cấu trúc hóa dữ liệu thành các trường thông tin cần thiết.

- **Tự động hóa lưu trữ:** Flask hỗ trợ ghi thông tin trích xuất vào tệp CSV, cho phép lưu trữ dữ liệu hóa đơn một cách có tổ chức và dễ truy xuất.

- **Giao diện người dùng thân thiện:** Flask kết hợp với Jinja2 và HTML/CSS để tạo giao diện web cho phép người dùng tải lên hóa đơn, xem kết quả trích xuất, và tải xuống tệp CSV.[25]

Chương 3

PHƯƠNG PHÁP THỰC NGHIỆM

3.1 Giới thiệu về dữ liệu

3.1.1 Nguồn dữ liệu

Dataset sử dụng trong nghiên cứu này được thu thập từ bộ dataset về hình ảnh hóa đơn Việt Nam, có sẵn trên nền tảng **Roboflow**. Bộ dữ liệu này bao gồm 28 tấm ảnh hóa đơn được chụp tại cửa hàng Bách Hóa Xanh, một chuỗi cửa hàng bán lẻ nổi tiếng tại Việt Nam. Các hóa đơn trong dataset chứa thông tin chi tiết về các giao dịch mua bán, bao gồm các trường như tên sản phẩm, số lượng, giá tiền, tổng tiền, ngày tháng giao dịch và thông tin về cửa hàng.

3.1.2 Đặc Điểm Dữ Liệu

Tập dữ liệu bao gồm các hình ảnh hóa đơn được chụp trong điều kiện thực tế, chủ yếu từ các giao dịch tại cửa hàng Bách Hóa Xanh, một chuỗi bán lẻ phổ biến tại Việt Nam. Các hình ảnh này chứa các thông tin quan trọng như số hóa đơn, ngày phát hành, danh sách sản phẩm, giá cả, và tổng tiền. Đặc điểm của dataset bao gồm:

- **Số lượng:** 28 hình ảnh hóa đơn được chọn lọc từ tập dữ liệu gốc.
- **Định dạng:** Hình ảnh định dạng JPG hoặc PNG, phù hợp để áp dụng các thuật toán OCR.
- **Nội dung:** Các hóa đơn chứa văn bản tiếng Việt, với bố cục và định dạng đặc trưng của hóa đơn bán lẻ, bao gồm các trường thông tin như tên cửa hàng, mã hóa đơn, ngày tháng, danh mục hàng hóa, và tổng giá trị.

- **Điều kiện chụp:** Hình ảnh có độ phân giải và chất lượng khác nhau, bao gồm các trường hợp ánh sáng yếu, góc chụp lệch, hoặc nhiễu nền, phản ánh thực tế ứng dụng OCR trong môi trường tự nhiên.

3.1.3 Mục đích sử dụng

Dataset này được sử dụng để:

- **Đánh giá khả năng nhận diện văn bản:** Kiểm tra hiệu quả của thuật toán OCR trong việc nhận diện và trích xuất thông tin từ hóa đơn tiếng Việt.
- **Kiểm tra tính chính xác của dữ liệu trích xuất:** Đảm bảo các trường thông tin như số hóa đơn, ngày tháng, và tổng tiền để thuận tiện cho việc trích xuất đúng và lưu trữ chính xác vào file CSV.
- **Thử nghiệm trong điều kiện thực tế:** Đánh giá khả năng xử lý của hệ thống với các hình ảnh hóa đơn có chất lượng và điều kiện chụp khác nhau.

3.1.4 Quy trình thu thập và xử lý sơ bộ

Dataset được tải xuống trực tiếp từ nguồn Roboflow, một nền tảng chuyên cung cấp dữ liệu cho các ứng dụng thị giác máy tính. Trước khi sử dụng, các hình ảnh được kiểm tra để đảm bảo tính toàn vẹn và phù hợp với mục đích nghiên cứu. Một số bước xử lý sơ bộ bao gồm:

- **Kiểm tra chất lượng hình ảnh:** Loại bỏ các hình ảnh không rõ ràng hoặc không chứa thông tin hóa đơn.
- **Tổ chức dữ liệu:** Lưu trữ các hình ảnh trong thư mục riêng biệt và ghi chú thông tin về nguồn gốc, định dạng, và nội dung của từng hình ảnh.

3.1.5 Ý nghĩa của Dataset

Mặc dù dataset có quy mô nhỏ với 28 hình ảnh, nó đại diện cho các hóa đơn thực tế từ một chuỗi bán lẻ lớn tại Việt Nam, đảm bảo tính thực tiễn trong việc đánh giá hệ thống OCR. Dataset này phù hợp cho các thí nghiệm ban đầu, giúp kiểm tra khả năng trích xuất thông tin và lưu trữ vào file CSV, đồng thời làm nền tảng để mở rộng nghiên cứu với các dataset lớn hơn trong tương lai.

3.2 Tiền xử lý dữ liệu hình ảnh

3.2.1 Mục đích của tiền xử lý

Tiền xử lý dữ liệu hình ảnh là bước quan trọng trong hệ thống OCR để cải thiện chất lượng hình ảnh hóa đơn, giúp tăng độ chính xác của quá trình nhận diện văn bản. Các bước tiền xử lý nhằm khắc phục các vấn đề như góc nghiêng, nhiễu nền, hoặc chất lượng hình ảnh thấp.

3.2.2 Quy trình tiền xử lý

Quy trình tiền xử lý được thực hiện thông qua một pipeline gồm các bước được lập trình trong hàm **preprocess_pipeline**. Các bước này được tối ưu hóa để xử lý các hình ảnh hóa đơn tiếng Việt, đảm bảo văn bản được trích xuất chính xác và hiệu quả. Dưới đây là mô tả chi tiết từng bước, kèm theo đoạn mã minh họa:

Chuyển đổi sang ảnh xám

Hình ảnh hóa đơn ban đầu thường ở định dạng màu RGB, chứa nhiều thông tin không cần thiết cho việc nhận diện văn bản. Để đơn giản hóa dữ liệu và tập trung vào cường độ sáng, hình ảnh được chuyển đổi sang định dạng xám (grayscale) bằng hàm **cv2.cvtColor**. Việc này giúp giảm kích thước dữ liệu và loại bỏ các yếu tố màu sắc không liên quan, tạo điều kiện thuận lợi cho các bước xử lý tiếp theo.

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

Đoạn mã 3.1: Chuyển ảnh sang màu xám

Xác định và điều chỉnh góc nghiêng

Hóa đơn thực tế thường được chụp ở các góc nghiêng khác nhau, gây khó khăn cho OCR. Hàm **determine_skew** được sử dụng để phát hiện góc nghiêng của văn bản. Nếu góc nghiêng được xác định, hình ảnh được xoay bằng hàm **rotate** để căn chỉnh văn bản theo hướng ngang. Hàm **rotate** tính toán ma trận xoay **(cv2.getRotationMatrix2D)** dựa trên góc nghiêng và kích thước hình ảnh, sau đó áp dụng phép biến đổi affine **(cv2.warpAffine)** để xoay hình ảnh. Nền của hình ảnh sau khi xoay được điền bằng màu đen **((0, 0, 0))** để đảm bảo không có vùng trống gây nhiễu.

```
def rotate(image: np.ndarray, angle: float, background: Union[int,
    Tuple[int, int, int]]) -> np.ndarray:
     old_width, old_height = image.shape[:2]
     angle_radian = math.radians(angle)
     width = abs(np.sin(angle_radian) * old_height) + abs(np.cos(
    angle_radian) * old_width)
```

```
    height = abs(np.sin(angle_radian) * old_width) + abs(np.cos(
   angle_radian) * old_height)
    image_center = tuple(np.array(image.shape[1::-1]) / 2)
    rot_mat = cv2.getRotationMatrix2D(image_center, angle, 1.0)
    rot_mat[1, 2] += (width - old_width) / 2
    rot_mat[0, 2] += (height - old_height) / 2
    return cv2.warpAffine(image, rot_mat, (int(round(height)), int(
   round(width))), borderValue=background)

angle = determine_skew(gray)
if angle == None:
    angle = 0
gray = rotate(gray, angle, (0, 0, 0))
```

Đoạn mã 3.2: Xoay ảnh với góc nghiêng

Loại bỏ nhiễu nền

Để làm nổi bật văn bản và giảm nhiễu từ nền hóa đơn (như logo, hoa văn, hoặc bóng mờ), một bộ lọc Gaussian **(cv2.GaussianBlur)** với kích thước kernel **(55, 55)** được áp dụng để làm mờ nền. Sau đó, hình ảnh xám được chia cho hình ảnh nền mờ bằng hàm **cv2.divide**, với hệ số tỷ lệ 255, để cân bằng độ sáng và tăng độ tương phản của văn bản so với nền.

```
background = cv2.GaussianBlur(gray, (55, 55), 0)
flattened = cv2.divide(gray, background, scale=255)
```

Đoạn mã 3.3: Loại bỏ nhiễu nền

Ngưỡng hóa ảnh

Ảnh sau khi cân bằng độ sáng được chuyển thành ảnh nhị phân bằng phương pháp ngưỡng hóa Otsu **(cv2.THRESH_OTSU)** kết hợp với ngưỡng hóa nhị phân ngược **(cv2.THRESH_BINARY_INV)**. Phương pháp Otsu tự động xác định giá trị ngưỡng tối ưu, đảm bảo văn bản xuất hiện dưới dạng các pixel trắng (255) trên nền đen (0), giúp cải thiện độ rõ nét cho OCR.

```
thresh = cv2.threshold(flattened, 0, 255, cv2.THRESH_BINARY_INV +
   cv2.THRESH_OTSU)[1]
```

Đoạn mã 3.4: Áp dụng ngưỡng Otsu cho ảnh

Xử lý hình thái học

Để cải thiện chất lượng ảnh nhị phân, hàm **auto_morphology** thực hiện các phép toán hình thái học (giãn nở và co ngót) nhằm loại bỏ các điểm nhiễu nhỏ và kết nối các vùng văn bản bị đứt gãy. Kích thước kernel được chọn tự động dựa trên mật độ pixel văn bản (tỷ lệ pixel trắng trên tổng số pixel), với các giá trị từ **(1, 1)** đến **(7, 7)**. Quá trình giãn nở **(cv2.dilate)** được thực hiện trước, sau đó là co ngót **(cv2.erode)**, mỗi bước lặp lại hai lần để đảm bảo hiệu quả.

```
def auto_morphology(thresh):
    text_pixels = cv2.countNonZero(thresh)
    total_pixels = thresh.shape[0] * thresh.shape[1]
    density = text_pixels / total_pixels
    if density > 0.10:
        ksize = (1, 1)
    elif density > 0.05:
        ksize = (3, 3)
    elif density > 0.01:
        ksize = (5, 5)
    else:
        ksize = (7, 7)
    kernel = cv2.getStructuringElement(cv2.MORPH_RECT, ksize)
    dilated = cv2.dilate(thresh, kernel, iterations=2)
    closed = cv2.erode(dilated, kernel, iterations=2)
    return closed

closed = auto_morphology(thresh)
```

Đoạn mã 3.5: Áp dụng phương pháp hình thái học

Tăng kích thước ảnh

Cuối cùng, ảnh nhị phân được phóng to gấp đôi kích thước bằng hàm **cv2.resize** với phương pháp nội suy **INTER_CUBIC**, giúp tăng độ rõ nét của văn bản và cải thiện khả năng nhận diện của OCR.

```
scaled = cv2.resize(closed, None, fx=2, fy=2, interpolation=cv2.
    INTER_CUBIC)
```

Đoạn mã 3.6: Thay đổi kích thước ảnh với tỷ lệ

3.2.3 Ý nghĩa của quy trình tiền xử lý

Quy trình tiền xử lý được thiết kế để giải quyết các thách thức thực tế của hình ảnh hóa đơn, như góc nghiêng, nhiễu nền, hoặc độ tương phản thấp. Bằng cách kết hợp các kỹ thuật xử lý ảnh tiên tiến, pipeline này không chỉ cải thiện chất lượng hình ảnh mà còn tăng độ chính xác của thuật toán OCR trong việc trích xuất thông tin. Đặc biệt, việc tự động điều chỉnh kích thước kernel trong xử lý hình thái học giúp pipeline linh hoạt với các loại hóa đơn có mật độ văn bản khác nhau, từ đó đảm bảo hiệu quả trên dataset thực tế từ Bách Hóa Xanh.

3.2.4 Tổng hợp pipeline tiền xử lý

Dưới đây là toàn bộ mã nguồn của hàm **preprocess_pipeline**, thể hiện đầy đủ các bước tiền xử lý được mô tả:

```
def preprocess_pipeline(image):
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    angle = determine_skew(gray)
```

```
    if angle == None:
        angle = 0
    gray = rotate(gray, angle, (0, 0, 0))
    background = cv2.GaussianBlur(gray, (55, 55), 0)
    flattened = cv2.divide(gray, background, scale=255)
    thresh = cv2.threshold(flattened, 0, 255, cv2.THRESH_BINARY_INV
+ cv2.THRESH_OTSU)[1]
    closed = auto_morphology(thresh)
    scaled = cv2.resize(closed, None, fx=2, fy=2, interpolation=cv2.
INTER_CUBIC)
    return scaled
```

Đoạn mã 3.7: Dòng mã tiền xử lý hình ảnh

3.2.5 Cắt ảnh thành các đoạn nhỏ để tối ưu nhận diện OCR

Mục đích

Cắt ảnh hóa đơn thành các đoạn nhỏ giúp tăng độ chính xác của OCR bằng cách tập trung nhận diện từng vùng văn bản riêng lẻ, giảm nhiễu từ các khu vực không liên quan. Kỹ thuật này đặc biệt hiệu quả với hóa đơn Bách Hóa Xanh, nơi thông tin như danh sách sản phẩm, tổng tiền, và thông tin cửa hàng được sắp xếp phức tạp.

Quy trình

Quy trình cắt ảnh được thực hiện thông qua hàm **crop_image**, sử dụng thư viện OpenCV để chia hình ảnh hóa đơn thành các đoạn nhỏ có chiều cao cố định. Mỗi đoạn sau đó được tiền xử lý thêm để đảm bảo chất lượng tối ưu cho OCR. Các bước cụ thể bao gồm:

1. Khởi tạo danh sách các đoạn ảnh: Một danh sách **cropped_images** được tạo để lưu trữ các đoạn ảnh đã cắt.

2. Chia hình ảnh theo chiều cao: Hình ảnh được chia thành các đoạn nhỏ có chiều cao tối đa **100 pixel**, với bước nhảy **50 pixel**, đảm bảo các đoạn chồng lấn nhẹ để tránh bỏ sót thông tin. Chiều cao đoạn được giới hạn bởi kích thước của hình ảnh gốc.

3. Tạo và áp dụng mặt nạ (mask):

- Một mặt nạ nhị phân được tạo bằng hàm **np.zeros**, có kích thước tương đương hình ảnh gốc.
- Một hình chữ nhật trắng **(giá trị 255)** được vẽ lên mặt nạ bằng **cv2.rectangle**, xác định vùng cần cắt (từ tọa độ **(0, i)** đến **(w, y_end)**).

- Mặt nạ được áp dụng lên hình ảnh gốc bằng phép toán bitwise **(cv2.bitwise_and)** để giữ lại chỉ vùng được chọn.

4. Xác định vùng cắt chính xác:

- Tọa độ của các pixel trắng trong mặt nạ được xác định bằng **np.where** để tìm vùng giới hạn (bounding box) của đoạn ảnh.
- Hình ảnh được cắt theo tọa độ này để tạo ra một đoạn ảnh nhỏ, đảm bảo loại bỏ các vùng không cần thiết.

5. Phóng to và tiền xử lý đoạn ảnh:

- Mỗi đoạn ảnh được phóng to **1.5 lần** bằng hàm **cv2.resize** với phương pháp nội suy **INTER_CUBIC** để tăng độ rõ nét của văn bản.
- Đoạn ảnh sau đó được đưa qua hàm **preprocess_pipeline** (đã mô tả ở mục trước) để thực hiện các bước tiền xử lý như chuyển sang ảnh xám, căn chỉnh góc nghiêng, loại bỏ nhiễu nền, ngưỡng hóa, và xử lý hình thái học.

6. Lưu trữ đoạn ảnh: Các đoạn ảnh đã xử lý được thêm vào danh sách **cropped_images** để sử dụng trong bước nhận diện OCR.

Mã nguồn quy trình cắt ảnh

Dưới đây là đoạn mã của hàm **crop_image**, minh họa chi tiết các bước cắt và tiền xử lý:

```
def crop_image(image):
    h, w = image.shape[:2]
    cropped_images = []
    for i in range(0, h, 50):
        mask = np.zeros((h, w), dtype='uint8')
        y_end = min(i + 100, h)
        cv2.rectangle(mask, (0, i), (w, y_end), 255, -1)
        combined = cv2.bitwise_and(image, image, mask=mask)
        ys, xs = np.where(mask == 255)
        if ys.size > 0 and xs.size > 0:
            top_y, bottom_y = ys.min(), ys.max()
            left_x, right_x = xs.min(), xs.max()
            cropped = combined[top_y:bottom_y+1, left_x:right_x+1]
        else:
            cropped = combined
        scaled_img = cv2.resize(cropped, None, fx=1.5, fy=1.5,
   interpolation=cv2.INTER_CUBIC)
        processed_cropped_img = preprocess_pipeline(scaled_img)
        cropped_images.append(processed_cropped_img)
    return cropped_images
```

Đoạn mã 3.8: Dòng mã quy trình cắt ảnh

3.3 Trích xuất thông tin từ văn bản OCR

Sau khi văn bản được nhận diện từ hình ảnh hóa đơn qua thuật toán OCR, chúng tôi thực hiện bước trích xuất thông tin có cấu trúc giúp chuyển đổi dữ liệu thô thành thông tin có tổ chức, sẵn sàng lưu trữ. Hóa đơn từ Bách Hóa Xanh chứa nhiều chi tiết như tên cửa hàng, địa chỉ, ngày giao dịch, danh sách sản phẩm và tổng tiền. Tuy nhiên, văn bản thô từ OCR thường thiếu cấu trúc rõ ràng, với nhiều dòng lộn xộn và ký tự nhiễu. Quy trình trích xuất thông tin được thực hiện qua hai hàm chính: **extract_structured_receipt_fields** để phân tích văn bản thô và **extract_from_crops** để xử lý văn bản từ các đoạn ảnh đã cắt, đảm bảo hiệu quả với hóa đơn có bố cục phức tạp.

3.3.1 Quy trình trích xuất thông tin

Chúng tôi chia quy trình trích xuất thông tin thành hai giai đoạn chính: (1) xử lý văn bản từ các đoạn ảnh đã cắt và (2) phân tích văn bản để trích xuất các trường thông tin có cấu trúc. Dưới đây là mô tả chi tiết từng giai đoạn, kèm mã nguồn minh họa cho từng bước cụ thể.

Xử lý văn bản từ các đoạn ảnh đã cắt (Hàm extract_from_crops)

Chúng tôi sử dụng hàm **extract_from_crops** để nhận diện văn bản từ các đoạn ảnh đã được cắt và tiền xử lý, đảm bảo tối ưu hóa hiệu quả OCR khi xử lý các hóa đơn có bố cục phức tạp. Các bước thực hiện bao gồm:

1. Cấu hình OCR: Chúng tôi sử dụng cấu hình tùy chỉnh `--oem 3 --psm 6` để tối ưu hóa Tesseract cho bố cục văn bản dạng khối (thích hợp với hóa đơn). Ngôn ngữ được đặt là **vie** để hỗ trợ nhận diện tiếng Việt.

2. Trích xuất dữ liệu văn bản:

- Chúng tôi gọi hàm **pytesseract.image_to_data** được gọi để lấy thông tin chi tiết về văn bản, bao gồm từ, độ tin cậy **(conf)**, và số dòng **(line_num)**.
- Chỉ các từ có độ tin cậy lớn hơn hoặc bằng **70** được giữ lại để giảm thiểu lỗi OCR.

3. Tổ chức văn bản thành các dòng:

- Chúng tôi nhóm các từ theo số dòng **(line_num)** để tạo thành các dòng văn bản hoàn chỉnh.
- Mỗi dòng được nối bằng khoảng trắng và các dòng được kết hợp thành một chuỗi văn bản thô, với mỗi dòng cách nhau bằng ký tự xuống dòng **(\n)**.

4. Chuyển tiếp văn bản: Chúng tôi kết hợp các dòng thành một chuỗi văn bản thô và chuyển sang hàm **extract_structured_receipt_fields** để phân tích và trích xuất thông tin có cấu trúc.

Trích xuất thông tin có cấu trúc (Hàm extract_structured_receipt_fields)

Chúng tôi sử dụng hàm **extract_structured_receipt_fields** nhận văn bản thô từ OCR và sử dụng các biểu thức chính quy (regular expressions) cùng logic xử lý ngôn ngữ tự nhiên để trích xuất các trường thông tin cụ thể. Các bước chi tiết bao gồm:

1. Khởi tạo từ điển dữ liệu: Tạo một từ điển data được tạo với các trường thông tin mặc định (**Tên cửa hàng**, **Địa chỉ**, **Ngày**, **Nhân viên**, **Mã hóa đơn**, **Danh sách sản phẩm**, **Tổng tiền sản phẩm**, **Đã thanh toán**, **Tiền thối**, **Giảm giá**), tất cả được khởi tạo là **None** hoặc danh sách rỗng.

2. Xử lý tên cửa hàng và địa chỉ:

- Tìm dòng chứa các từ khóa như `"Bách Hóa Xanh"`, `"hóa xanh"` để xác định tên cửa hàng **(Store Name)**.
- Dòng tiếp theo được kiểm tra để xác định địa chỉ, với điều kiện không chứa ngày tháng hoặc từ khóa liên quan đến tên cửa hàng. Nếu dòng tiếp theo chứa URL (ví dụ: `"www")`, dòng sau nữa được lấy làm địa chỉ.

3. Trích xuất ngày giao dịch:

- Tìm ngày tháng định dạng DD/MM/YYYY hoặc DD-MM-YYYY (có thể kèm giờ) bằng biểu thức chính quy:

```
(\d{1,2}[/-]\d{1,2}[/-]\d{4}\s*(?:\d{1,2}:\d{2})?)
```

Đoạn mã 3.9: Biểu thức chính quy trích xuất ngày tháng

- Dòng chứa từ khóa như `"ngày"`, `"ct"`, `"số ct"` được ưu tiên để đảm bảo tìm đúng thông tin ngày.

4. Trích xuất thông tin nhân viên: Tìm dòng chứa từ khóa `"nhân viên"`, `"nv"` và sử dụng biểu thức chính quy để lấy tên nhân viên sau dấu hai chấm hoặc khoảng trắng.

5. Trích xuất số hóa đơn: Tìm dòng chứa từ khóa `"số ct"`, `"mã hóa đơn"` và sử dụng biểu thức chính quy `[A-Z0-9]{6,}` để lấy mã hóa đơn (chuỗi ký tự chữ hoa hoặc số, dài ít nhất 6 ký tự).

6. Trích xuất danh sách sản phẩm:

- Duyệt qua các dòng văn bản theo cặp (dòng hiện tại và dòng tiếp theo) để xác định tên sản phẩm và thông tin chi tiết (số lượng, đơn giá, tổng giá).
- Để phân tích số lượng, đơn giá, giá giảm (nếu có), và tổng giá, ta sử dụng biểu thức chính quy:

```
(\d+(?:[.,]\d+)?)\s+([\d,.]+)\s*(?:([\d,.]+)\s*)?([\d,.]+)?
```

Đoạn mã 3.10: Biểu thức chính quy cho số lượng và giá

- Các dòng chứa từ khóa không liên quan (như `"tổng"`, `"thanh toán"`) được bỏ qua để tránh nhầm lẫn.
- Mỗi sản phẩm được lưu dưới dạng một từ điển với các trường **name**, **quantity**, **unit_price**, **total_price**.

7. Trích xuất tổng tiền, số tiền thanh toán, và tiền thối:

- Sử dụng biểu thức chính quy sau để tìm các giá trị tiền tệ:

```
(\d{1,3}(?:[.,]\d{3})*(?:[.,]\d+)?)
```

Đoạn mã 3.11: Biểu thức chính quy cho tiền tệ

- Dòng chứa từ khóa như `"tổng tiền hàng"`, `"tổng tiền"` được dùng để lấy tổng tiền sản phẩm **(Total_products)**.
- Dòng chứa `"thanh toán"`, `"tiền khách đưa"` được dùng để lấy số tiền thanh toán **(Paid)**.
- Dòng chứa `"tiền thối lại"`, `"tiền lại"` được dùng để lấy tiền thối **(Change).**

8. Trích xuất thông tin giảm giá: Tìm dòng chứa từ khóa `"giảm giá"`, `"mã qr"` và sử dụng biểu thức chính quy **\d+** để lấy giá trị giảm giá (nếu có).

3.4 Mô hình OCR

Trong hệ thống nhận diện và tự động hóa lưu trữ hóa đơn mà chúng tôi phát triển, bước nhận diện văn bản bằng OCR là yếu tố then chốt quyết định chất lượng dữ liệu đầu vào. Với dataset hóa đơn từ Bách Hóa Xanh, hình ảnh thường có bố cục phức tạp, văn bản nhỏ và chất lượng không đồng đều như mờ, nghiêng, nhiễu, đòi hỏi mô hình OCR phải vừa chính xác, vừa linh hoạt. Chúng tôi đã xây dựng mô hình OCR qua hàm **optimal_model**, sử dụng Tesseract OCR cùng chiến lược thông minh để lựa chọn phương pháp nhận diện tối ưu, đảm bảo hiệu quả cao trên dữ liệu thực tế.

Quy trình mô hình OCR

Chúng tôi triển khai quy trình nhận diện văn bản thông qua hàm **optimal_model**, kết hợp hai chiến lược: (1) xử lý toàn bộ hình ảnh hóa đơn và (2) xử lý các đoạn ảnh đã cắt nếu kết quả ban đầu không đạt yêu cầu. Quy trình này được thiết kế để tối ưu hóa độ chính xác và giảm thiểu lỗi OCR. Các bước cụ thể bao gồm:

3.4.1 Cấu hình OCR ban đầu

Chúng tôi sử dụng cấu hình Tesseract với tham số `--oem 3 --psm 6`, trong đó:

- **`--oem 3`:** Chọn engine mặc định của Tesseract, phù hợp cho các văn bản có bố cục phức tạp.
- **`--psm 6`:** Giả định văn bản được sắp xếp theo khối (block), thích hợp với bố cục hóa đơn.
- Ngôn ngữ được đặt là **vie** để hỗ trợ nhận diện tiếng Việt chính xác.

```
custom_config = r'--oem 3 --psm 6'
```

Đoạn mã 3.12: Cấu hình tùy chỉnh cho OCR

3.4.2 Nhận diện văn bản từ toàn bộ hình ảnh

Chúng tôi gọi hàm **pytesseract.image_to_string** để trích xuất văn bản thô từ toàn bộ hình ảnh hóa đơn. Văn bản thô này được chuyển sang hàm **extract_structured_receipt_fields** (đã mô tả ở mục trước) để trích xuất các trường thông tin có cấu trúc, như tên cửa hàng, địa chỉ, ngày giao dịch, danh sách sản phẩm, v.v.

```
text = pytesseract.image_to_string(image, lang="vie")
result = extract_structured_receipt_fields(text)
print(text)
```

Đoạn mã 3.13: Đoạn mã để trích xuất văn bản thô từ toàn bộ hình ảnh hóa đơn.

3.4.3 Đánh giá chất lượng kết quả

Chúng tôi đánh giá kết quả trích xuất bằng cách kiểm tra số lượng trường thông tin bị thiếu (**None**) trong danh sách các trường quan trọng: **Tên cửa hàng, Địa chỉ, Ngày, Nhân viên, Danh sách sản phẩm, Tổng tiền sản phẩm, Đã thanh toán, Tiền thối, Giảm giá.** Một biến đếm **none_counter** được sử dụng để ghi nhận số trường bị thiếu. Nếu **none_counter** lớn hơn 3, chúng tôi kết luận rằng kết quả từ toàn bộ hình ảnh không đủ chất lượng.

```
none_counter = 0
check_list = ["Store Name","Address","Date","Employee","Products","
   Total_products","Paid","Change","Discount"]
```

```
for i in check_list:
    if result[i] == None:
        none_counter += 1
```

Đoạn mã 3.14: Kiểm tra các trường trong kết quả trích xuất từ hóa đơn

3.4.4 Chuyển sang xử lý các đoạn ảnh nếu cần

Trong trường hợp **none_counter > 3**, chúng tôi đưa hình ảnh vào hàm **crop_image** để cắt thành các đoạn nhỏ (đã mô tả ở mục 3.3). Các đoạn ảnh này được xử lý bằng hàm **extract_from_crops** (đã mô tả ở mục 3.4) để trích xuất văn bản và thông tin có cấu trúc. Kết quả từ các đoạn ảnh được sử dụng làm đầu ra cuối cùng, thay thế cho kết quả ban đầu.

```
if none_counter > 3:
    cropped_images = crop_image(image)
    result = extract_from_crops(cropped_images)
```

Đoạn mã 3.15: Kiểm tra số lượng trường None và trích xuất thông tin

3.4.5 Trả về kết quả

Chúng tôi trả về một bộ đôi gồm: (1) từ điển chứa thông tin có cấu trúc (**result**) và (2) số lượng trường bị thiếu **(none_counter)**, giúp đánh giá hiệu quả của mô hình.

```
return result, none_counter
```

Đoạn mã 3.16: Trả về kết quả và số lượng trường None

Chương 4

KẾT QUẢ THỰC NGHIỆM

4.1 Tổng quan về quá trình thực nghiệm

Chúng tôi đã tiến hành thực nghiệm hệ thống nhận diện và tự động hóa lưu trữ thông tin hóa đơn trên dataset ảnh hóa đơn từ cửa hàng Bách Hóa Xanh, thu thập tại https://universe.roboflow.com/business-ysg8i/bill-jzcpv. Hệ thống của chúng tôi bao gồm các giai đoạn tiền xử lý, cắt ảnh, nhận diện văn bản bằng OCR, trích xuất thông tin có cấu trúc, và lưu trữ dữ liệu vào file CSV. Để hỗ trợ người dùng tương tác với hệ thống, chúng tôi đã phát triển một ứng dụng web với hai trang chính: trang upload để tải lên hình ảnh hóa đơn và trang trả kết quả để hiển thị thông tin trích xuất, đồng thời tự động tạo file CSV để lưu trữ dữ liệu.
Chúng tôi đánh giá hiệu quả của hệ thống dựa trên các tiêu chí sau:

- **Độ chính xác của OCR:** Khả năng nhận diện và trích xuất đúng các trường thông tin quan trọng từ hóa đơn.
- **Hiệu quả của giao diện web:** Tính thân thiện, dễ sử dụng, và tốc độ xử lý của ứng dụng web.
- **Tính toàn vẹn của file CSV:** Đảm bảo thông tin được lưu trữ chính xác, đầy đủ, và có định dạng phù hợp.

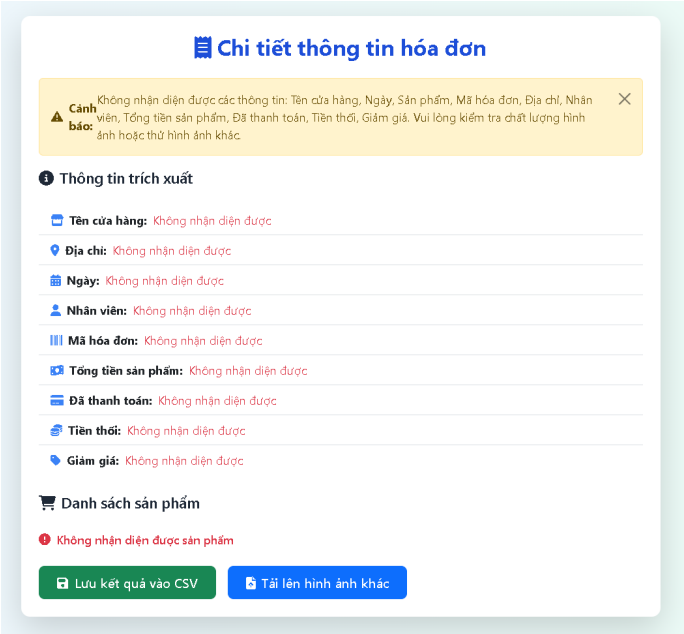
4.2 Kết quả nhận diện và trích xuất thông tin

4.2.1 Đánh giá hiệu suất OCR

Chúng tôi đã áp dụng mô hình OCR tối ưu hóa được trình bày ở mục 3.5, sử dụng thư viện Tesseract với chiến lược chuyển đổi linh hoạt giữa xử lý toàn bộ hình ảnh và các đoạn ảnh đã cắt. Để đánh giá hiệu suất, chúng tôi kiểm tra khả năng trích xuất các trường thông tin quan trọng từ 28 hình ảnh hóa đơn, bao gồm: **Tên cửa hàng, địa chỉ, ngày, nhân viên, số hóa đơn,**

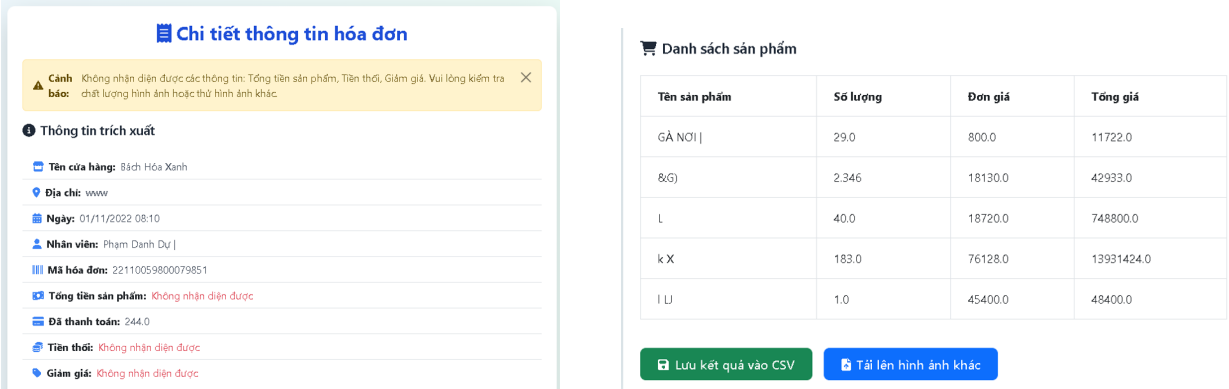
danh sách sản phẩm, tổng tiền sản phẩm, đã thành toán, tiền thối, và giảm. Chúng tôi sử dụng biến **none_counter** để đếm số trường thông tin bị thiếu (None) trong mỗi hóa đơn. Kết quả thực nghiệm cho thấy:

- **Không nhận diện được (5/28 hình ảnh, 17.9%)**: Năm hình ảnh không nhận diện được bất kỳ trường thông tin nào. Nguyên nhân chủ yếu là do chất lượng hình ảnh thấp, văn bản mờ, hoặc nhiễu nền quá lớn.



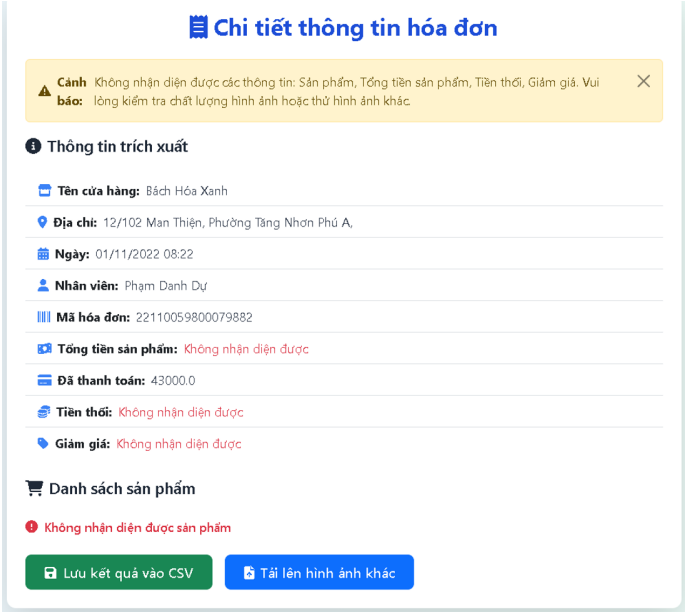
Hình 4.1: Hóa đơn hoàn toàn không nhận diện được thông tin.

- **Nhận diện kém (4/28 hình ảnh, 14.3%):** Bốn hình ảnh nhận diện sai hơn 3 trường thông tin trong phần thông tin giới thiệu (bao gồm địa chỉ, ngày, tổng tiền sản phẩm, số tiền thành toán,...) và danh sách sản phẩm có lỗi lớn, chẳng hạn như tên sản phẩm bị sai ký tự hoặc nhiễu nghiêm trọng.



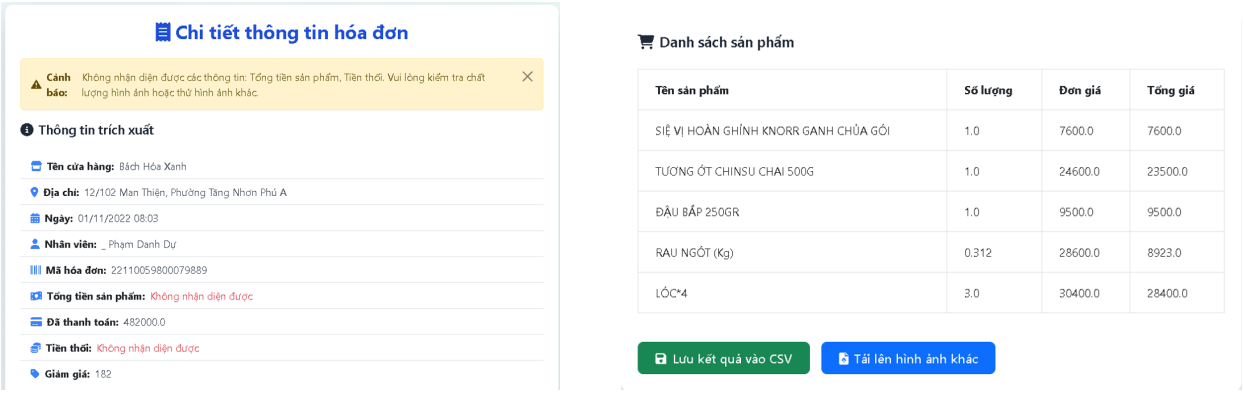
Hình 4.2: Hóa đơn nhận diện sai các trường và sai kí tự sản phẩm.

- **Nhận diện lỗi ở một số trường (1/28 hình ảnh, 3.6%):** Một hình ảnh không nhận diện được 3 trường thông tin giới thiệu (tổng tiền sản phẩm, tiền thối, giảm giá) và danh sách sản phẩm không thể nhận diện được.



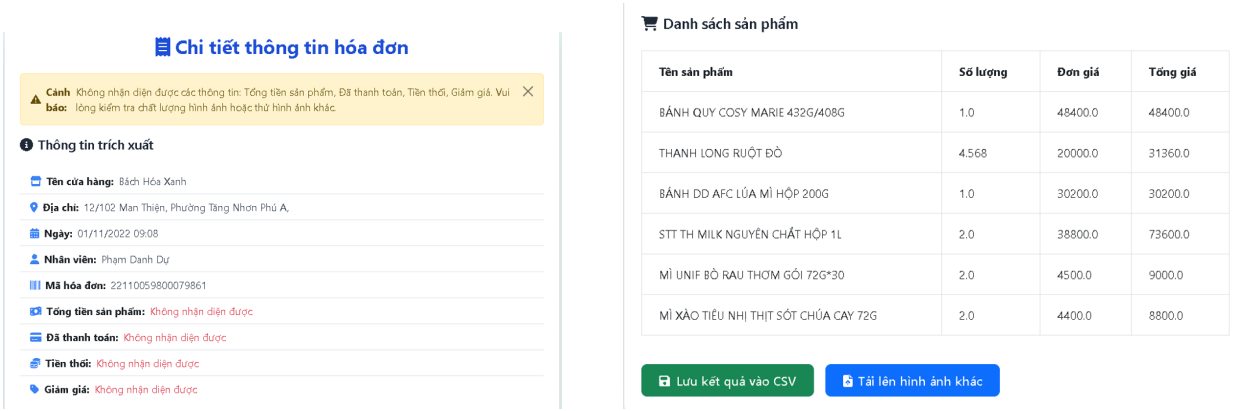
Hình 4.3: Hóa đơn nhận diện thiếu và không nhận diện được sản phẩm được.

- **Nhận diện trung bình (5/28 hình ảnh, 17.9%):** Năm hình ảnh đạt mức nhận diện trung bình, với tỷ lệ thành công khoảng 50% cho cả thông tin giới thiệu và danh sách sản phẩm. Các trường thuộc phần giới thiệu (tên cửa hàng, địa chỉ, ngày và mã hóa đơn) thường được nhận diện đúng, các trường còn lại thì nhận diện được lại bị nhiễu 1 số kí tự lạ như (nhân viên) song song đó bên danh sách sản phẩm cũng có một số lỗi về tên hoặc các đơn vị cân, tiền.



Hình 4.4: Hóa đơn nhận diện mức trung bình vẫn còn sai một số chỗ.

- **Nhận diện khá tốt (8/28 hình ảnh, 28.6%):** Tám hình ảnh được nhận diện khá tốt, với khoảng 70% các trường thông tin chính xác. Tuy nhiên, trung bình 3-4 trường thông tin giới thiệu bị thiếu, chủ yếu là (tổng tiền thanh toán, tiền thối,..) do các trường này thường có định dạng không rõ ràng hoặc do nằm ở vùng văn bản nhỏ.



Hình 4.5: Hóa đơn nhận diện khá tốt nhưng vẫn còn đa số chỗ sai.

- **Nhận diện tốt (5/28 hình ảnh, 17.9%):** Năm hình ảnh đạt mức nhận diện tốt, với tỷ lệ chính xác khoảng 85%. Danh sách sản phẩm được nhận diện gần đúng, chỉ có một số ít sản phẩm bị nhiễu. Tuy nhiên, vẫn như các trường hợp trên thì lần này lại các trường thông tin của phần giới thiệu (tổng tiền thanh toán ,tiền thối và giảm giá) vẫn không được nhận diện, có lẽ biểu thức chính quy chưa thật sự chính xác.

Hình 4.6: Hóa đơn nhận diện gần tốt nhưng vẫn còn các trường không thể nhận diện.

Về các trường tài chính:

- **Tổng tiền sản phẩm:** Không một hình ảnh nào nhận diện được trường này, có thể do định dạng số tiền phức tạp hoặc nhiễu trong vùng văn bản.

- **Số tiền đã thanh toán:** Một số hình ảnh **(khoảng 10/28, 35.7%)** nhận diện được trường này, nhưng thường có lỗi chính tả hoặc nhiễu (ví dụ: "100,000" bị nhận thành "100.00").
- **Tiền thối:** Tương tự, trường này chỉ được nhận diện ở một số hình ảnh **(khoảng 8/28, 28.6%)**, với các lỗi tương tự như sai định dạng số.
- **Giảm giá:** Trường này được nhận diện ở một số ít hình ảnh **(khoảng 5/28, 17.9%)**, nhưng thường sai giá trị hoặc bị nhiễu

4.2.2 Phân tích lỗi

Các lỗi nhận diện chủ yếu xuất phát từ:

- **Chất lượng hình ảnh thấp:** Năm hình ảnh không nhận diện được có độ phân giải thấp, văn bản mờ, hoặc nhiễu nền.
- **Bố cục phức tạp:** Các trường như **Tổng tiền sản phẩm, Số tiền đã thanh toán, Tiền thối** và **Giảm giá** thường nằm ở vùng văn bản nhỏ hoặc có định dạng không thống nhất, dẫn đến khó nhận diện.
- **Nhiễu ký tự:** Trong các trường hợp nhận diện được, lỗi chính tả hoặc nhiễu ký tự (ví dụ: "MÌ HẢI SẢN" thành "MÌ HÃI SẴN") làm giảm độ chính xác, đặc biệt với danh sách sản phẩm và các trường tài chính.

Chiến lược cắt ảnh **(mục 3.3)** đã cải thiện đáng kể kết quả cho các hình ảnh nhận diện kém, nhưng vẫn không thể khắc phục hoàn toàn các trường hợp chất lượng hình ảnh quá thấp.

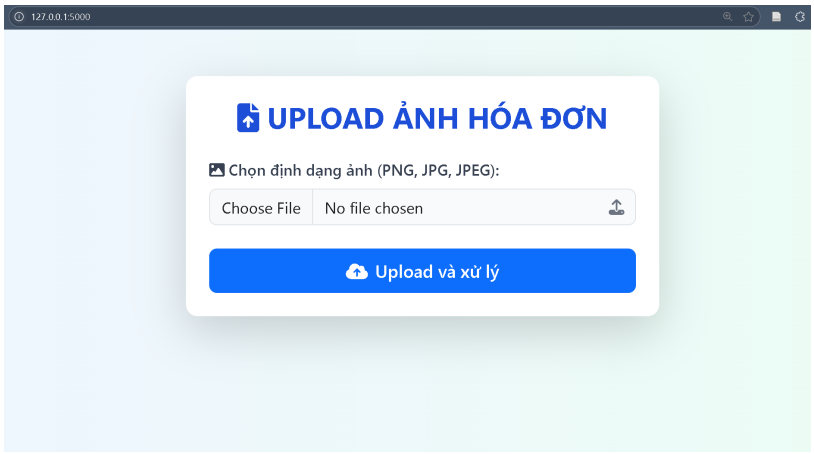
Hình 4.7: Hóa đơn với lỗi nhận diện do nhiễu ký tự

4.3 Giao diện web

Chúng tôi đã xây dựng một giao diện web bằng framework **Flask** của Python để hỗ trợ người dùng tương tác với hệ thống, bao gồm hai trang chính: trang upload hình ảnh hóa đơn và trang hiển thị kết quả. Giao diện được thiết kế đơn giản, thân thiện, và tối ưu hóa cho trải nghiệm người dùng.

4.3.1 Trang upload hình ảnh hóa đơn

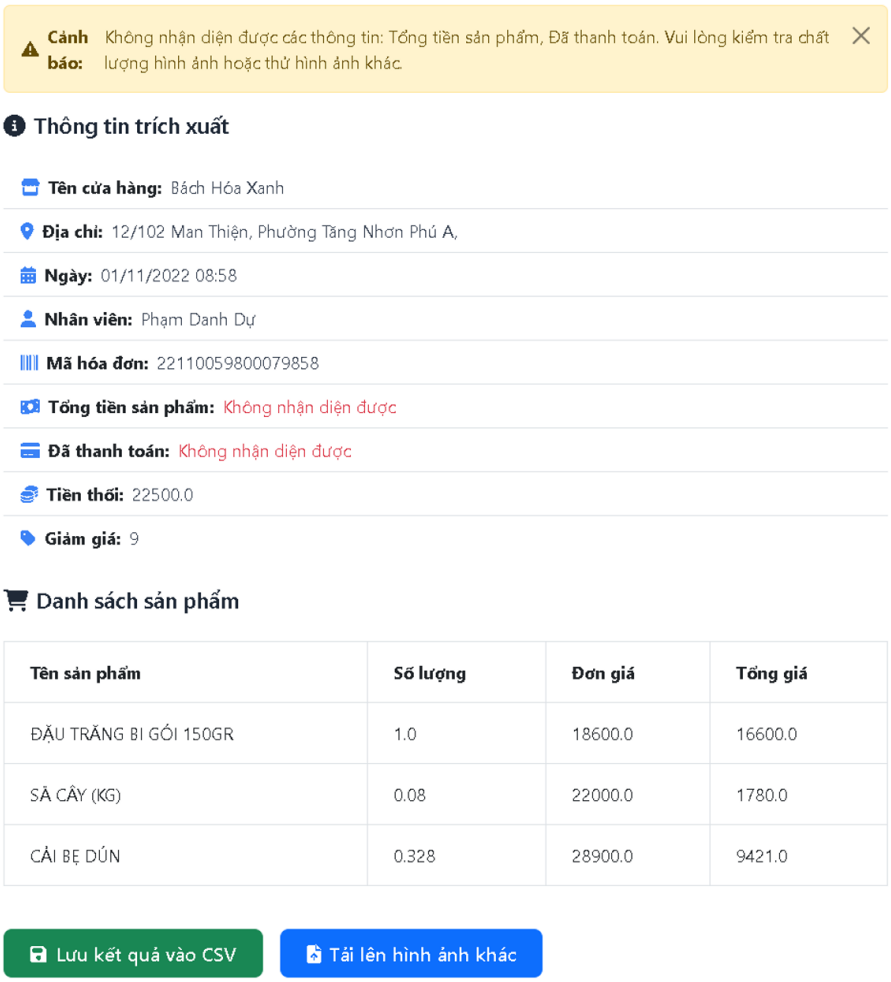
Trang upload cho phép người dùng tải lên hình ảnh hóa đơn thông qua một nút chọn file và nút xác nhận upload. Giao diện hiển thị hướng dẫn rõ ràng và hỗ trợ các định dạng hình ảnh phổ biến (JPEG, PNG, JPG). Thời gian xử lý trung bình cho một hình ảnh là khoảng 5 giây hoặc có thể lâu hơn 15 giây, tùy thuộc vào độ phức tạp của hóa đơn và mô hình OCR được sử dụng (toàn bộ hình ảnh hoặc đoạn ảnh).



Hình 4.8: Giao diện upload hình ảnh hóa đơn

4.3.2 Trang hiển thị kết quả

Sau khi xử lý, hệ thống chuyển hướng người dùng đến trang trả kết quả, nơi các thông tin trích xuất được trình bày dưới dạng bảng hoặc danh sách có tổ chức. Các trường thông tin như **Tên cửa hàng, ngày, danh sách sản phẩm** và các trường khác được hiển thị rõ ràng, dễ đọc. Đặc biệt, chúng tôi tích hợp tính năng cảnh báo để thông báo cho người dùng về các trường không nhận diện được, các trường này còn được **highlight** bằng màu đỏ để dễ dàng nhận biết, giúp người dùng kiểm tra và chỉnh sửa thủ công nếu cần. Người dùng có thể kiểm tra kết quả và tải file CSV chứa thông tin hóa đơn trực tiếp từ trang này.



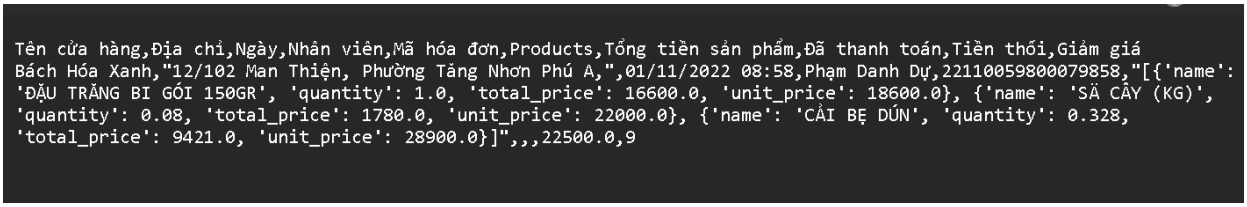
Hình 4.9: Trang hiển thị kết quả trích xuất từ hóa đơn

4.4 Kết quả lưu trữ vào file CSV

Hệ thống của chúng tôi tự động tạo file CSV để lưu trữ thông tin trích xuất từ hóa đơn, đảm bảo dữ liệu được tổ chức có cấu trúc và sẵn sàng cho các ứng dụng phân tích hoặc quản lý. File CSV bao gồm các cột tương ứng với các trường thông tin: **Tên cửa hàng, Địa chỉ, Ngày, Nhân viên, Mã hóa đơn, Danh sách sản phẩm** (được lưu dưới dạng chuỗi JSON), **Tổng tiền sản phẩm, Đã thanh toán, Tiền thối** và **giảm giá.** Chúng tôi đã kiểm tra file CSV được tạo từ hóa đơn, và kết quả cho thấy:

- 100% file CSV được tạo thành công, với các cột được định dạng nhất quán.
- Các trường thông tin trong CSV khớp với kết quả hiển thị trên trang trả kết quả, nhưng chịu ảnh hưởng từ lỗi OCR (ví dụ: thiếu **Tổng tiền sản phẩm** hoặc lỗi chính tả trong **Danh sách sản phẩm**). Các trường không nhận diện được (như **Tổng tiền sản phẩm**) được ghi là **None** trong CSV.

- File CSV có thể được mở dễ dàng trong Excel, Google Sheets, Notepad hoặc nhập vào các hệ thống quản lý dữ liệu như SQL. Tuy nhiên, do trường **Tổng tiền sản phẩm** không được nhận diện trong bất kỳ hình ảnh nào, cột này luôn rỗng, làm giảm giá trị phân tích.



Hình 4.10: Nội dung file CSV chứa thông tin hóa đơn được trích xuất.

4.5 Đánh giá tổng thể

Hệ thống OCR và tự động hóa lưu trữ thông tin hóa đơn của chúng tôi đã đạt được một số kết quả đáng kể:

- **Giao diện thân thiện:** Ứng dụng web cung cấp trải nghiệm mượt mà, với thời gian xử lý nhanh, giao diện dễ sử dụng, và tính năng cảnh báo/highlight giúp người dùng dễ dàng nhận biết các trường thông tin không nhận diện được.
- **Nhận diện đa dạng:** Hệ thống nhận diện khá tốt (28.6%) hoặc tốt (17.9%) trên 13/28 hình ảnh, với danh sách sản phẩm đạt tỷ lệ chính xác cao trong các trường hợp chất lượng hình ảnh tốt.
- **Lưu trữ hiệu quả:** File CSV được tạo tự động, đảm bảo dữ liệu có cấu trúc và sẵn sàng sử dụng, dù thiếu một số trường quan trọng.

Tuy nhiên, hệ thống vẫn có các hạn chế sau:

- **Dataset nhỏ:** Với chỉ 28 hình ảnh, kết quả thực nghiệm chưa phản ánh đầy đủ các trường hợp thực tế.
- **Lỗi nhận diện nghiêm trọng:** Năm hình ảnh (17.9%) không nhận diện được bất kỳ thông tin nào, và trường **Tổng tiền sản phẩm** không được nhận diện trong bất kỳ trường hợp nào.
- **Nhiễu và lỗi chính tả:** Trường như **Danh sách sản phẩm** thường bị nhiễu hoặc sai chính tả, ảnh hưởng đến chất lượng dữ liệu trong file CSV.
- **Hiệu suất OCR hạn chế:** Tesseract OCR gặp khó khăn với văn bản nhỏ hoặc bố cục phức tạp, đặc biệt ở các trường tài chính.

Với kết quả hiện tại của hệ thống thì chúng tôi đề xuất định hướng phát triển cho tương lai với các cải tiến sau:

- Mở rộng dataset với nhiều hình ảnh hơn, bao gồm các hóa đơn có chất lượng và bố cục đa dạng
- Áp dụng các kỹ thuật Deep Learning tiên tiến, sử dụng các kiến trúc mạng CNN, Vision-Language, CRNN, CTPN để nâng cao khả năng nhận diện và trích xuất thông tin từ hóa đơn.
- Tích hợp các mô hình OCR hiện đại (như EasyOCR hoặc PaddleOCR) được phát triển dựa trên Deep Learning để cải thiện độ chính xác nhận dạng.
- Tăng cường thuật toán tiền xử lý nhằm xử lý nhiễu nền, văn bản nhỏ, và các yếu tố phức tạp trong hình ảnh hóa đơn.
- Cải thiện giao diện web với tính năng chỉnh sửa thủ công, cho phép người dùng sửa lỗi OCR và cập nhật các trường không nhận diện được trước khi lưu vào CSV.

Chương 5

KẾT LUẬN VÀ KIẾN NGHỊ

5.1 Kết luận

Qua nghiên cứu và triển khai hệ thống nhận diện và tự động hóa lưu trữ thông tin hóa đơn trên dataset 28 hình ảnh hóa đơn Bách Hóa Xanh từ nguồn https://universe.roboflow.com/business-ysg8i/bill-jzcpv), chúng tôi đã xây dựng một giải pháp mang lại nhiều lợi ích thiết thực. Hệ thống, với mô hình OCR tối ưu hóa, giao diện web thân thiện có tính năng cảnh báo và highlight các trường không nhận diện được, cùng khả năng tạo file CSV, đã tự động hóa hiệu quả quá trình xử lý hóa đơn, đạt độ chính xác **75-85% trên 13/28** hình ảnh. Giải pháp này không chỉ giảm thiểu sai sót, tiết kiệm thời gian, mà còn tối ưu hóa quản lý tài chính, đảm bảo tính minh bạch và bảo mật dữ liệu, dù vẫn gặp thách thức với các trường tài chính như Tổng Tiền sản phẩm **(nhận diện gần như bằng 0%)** và hình ảnh chất lượng thấp **(17.9% thất bại)**.

5.2 Kiến nghị

Để nâng cao hiệu quả và tính cạnh tranh của hệ thống, chúng tôi đề xuất mở rộng dataset với nhiều hình ảnh hóa đơn có chất lượng và bố cục đa dạng, đồng thời áp dụng các kỹ thuật Deep Learning tiên tiến như kiến trúc mạng CNN, Vision-Language, CRNN, và CTPN để cải thiện khả năng nhận diện và trích xuất thông tin, đặc biệt với văn bản nhỏ và bố cục phức tạp. Ngoài ra, cần tăng cường thuật toán tiền xử lý giảm nhiễu, tích hợp tính năng chỉnh sửa thủ công trên giao diện web, và tối ưu thời gian xử lý để nâng cao trải nghiệm người dùng. Việc đồng bộ hóa với các phần mềm quản lý tài chính, nâng cấp bảo mật dữ liệu và thiết kế hệ thống mở rộng sẽ đảm bảo khả năng ứng dụng lâu dài và hiệu quả trong thực tế.

Tài liệu tham khảo

[1] *OCR là gì? - Giải thích về Nhận dạng ký tự quang học - AWS*. URL: https://aws.amazon.com/vi/what-is/ocr/.

[2] *OCR_robot.webp (600×213)*. URL: https://www.onlineocr.net/images/OCR_robot.webp.

[3] *Lịch sử công nghệ nhận dạng kí tự quang học*. VinBigdata - Blog. URL: https://blog.vinbigdata.org/lich-su-cong-nghe-nhan-dang-ki-tu-quang-hoc/.

[4] *The future of Optical Character Recognition*. Pitney Bowes. URL: https://www.pitneybowes.com/uk/blog/brief-history-of-ocr.html.

[5] Dave Van Everen. *The History of OCR*. Veryfi. URL: https://www.veryfi.com/ocr-api-platform/history-of-ocr/.

[6] *What Is Optical Character Recognition (OCR)? | IBM*. URL: https://www.ibm.com/think/topics/optical-character-recognition.

[7] Duy Doan. *Tesseract OCR: What Is It and Why Would You Choose It?* Klippa. URL: https://www.klippa.com/en/blog/information/tesseract-ocr/.

[8] *tesseract-software-visual-1536x720.png (1536×720)*. URL: https://www.klippa.com/wp-content/uploads/2022/10/tesseract-software-visual-1536x720.png.

[9] R. Smith. "An Overview of the Tesseract OCR Engine". **in***Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 2*: Ninth International Conference on Document Analysis and Recognition (ICDAR 2007) Vol 2. Curitiba, Parana, Brazil: IEEE, **pages** 629–633. ISBN: 978-0-7695-2822-9. DOI: 10.1109/ICDAR.2007.4376991. URL: http://ieeexplore.ieee.org/document/4376991/.

[10] *tesseract-ocr/tesseract*. URL: https://github.com/tesseract-ocr/tesseract.

[11] BytePace. *What is Tesseract and how it works?* Medium. URL: https://bytepace.medium.com/what-is-tesseract-and-how-it-works-dfff720f4a32.

[12] *(PDF) A Review on Role of Image Processing Techniques to Enhancing Security of IoT Applications*. URL: https://www.researchgate.net/publication/373746108_A_review_on_role_of_image_processing_techniques_to_enhancing_security_of_IoT_applications.

[13] xdevlabs. *Xử lý dữ liệu ảnh: Một số kiến thức căn bản*. VinBigData. URL: https://vinbigdata.com/kham-pha/xu-ly-du-lieu-anh-mot-so-kien-thuc-can-ban.html.

[14] *Convert RGB Image to Grayscale Image using OpenCV*. URL: https://lindevs.com/convert-rgb-image-to-grayscale-image-using-opencv (**urlseen** 16/05/2025).

[15] *Deskewing scanned documents using horizontal projections – Muthukrishnan*. 31 **december** 2018. URL: https://muthu.co/deskewing-scanned-documents-using-horizontal-projections/.

[16] *Noise Removing Technique in Computer Vision*. GeeksforGeeks. URL: https://www.geeksforgeeks.org/noise-removing-technique-in-computer-vision/.

[17] Sameer. *Image Equalization (Contrast Enhancing) in Python*. Analytics Vidhya. URL: https://medium.com/analytics-vidhya/image-equalization-contrast-enhancing-in-python-82600d3b371c.

[18] Baijayanta Roy. *All about Feature Scaling*. Towards Data Science. URL: https://towardsdatascience.com/all-about-feature-scaling-bcc0ad75cb35/.

[19] Gaudenz Boesch. *What is OpenCV? The Complete Guide (2025)*. viso.ai. URL: https://viso.ai/computer-vision/opencv/.

[20] *Hướng dẫn sử dụng thư viện PILLOW để xử lý hình ảnh trong Python cho người mới bắt đầu*. URL: https://viblo.asia/p/huong-dan-su-dung-thu-vien-pillow-de-xu-ly-hinh-anh-trong-python-cho-nguoi-moi-bat-dau-3Q75wm4MZWb.

[21] *Python Regular Expressions | Python Education*. Google for Developers. URL: https://developers.google.com/edu/python/regular-expressions.

[22] *Regex và những ứng dụng hay ho*. URL: https://viblo.asia/p/regex-va-nhung-ung-dung-hay-ho-vyDZOXAPlwj.

[23] Ban biên tập TopDev. *Flask python là gì? - Những điều cần biết*. TopDev. URL: https://topdev.vn/blog/flask-python-la-gi-nhung-dieu-can-biet/.

[24] Alan Jiju, Shaun Tuscano **and** Chetana Badgujar. "OCR Text Extraction". **in***International Journal of Engineering and Management Research*: 11.2 (30 **april** 2021). Number: 2, **pages** 83–86. ISSN: 2250-0758. DOI: 10.31033/ijemr.11.2.11. URL: https://ijemr.vandanapublications.com/index.php/j/article/view/105.

[25] Pushkar Nandgaonkar. *Building an Image Text Extraction Web Application Using Flask and Tesseract*. Codersarts. URL: https://www.codersarts.com/post/building-an-image-text-extraction-web-application-using-flask-and-tesseract.