HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN 1

BÁO CÁO BÀI TẬP LỚN

HỌC PHẦN: NHẬP MÔN TRÍ TUỆ NHÂN TẠO

ĐỀ TÀI: NHẬN DIỆN CHỮ VIẾT TIẾNG VIỆT QUA HÌNH ẢNH

Giảng viên hướng dẫn	**: TS. Nguyễn Thị Mai Trang**
Nhóm sinh viên thực hiện	**:**
B22DCCN407	**Đinh Quang Hưng**
B22DCCN395	**Phùng Bá Huy**
B22DCCN553	**Hoàng Văn Nam**
B22DCCN477	**Hoàng Sơn Lâm**
B22DCCN823	**Lê Đức Thiện**
Nhóm	**: 04**

Hà Nội – 2025

PHÂN CÔNG CÔNG VIỆC

TT	Công việc / Nhiệm vụ	SV thực hiện	Thời hạn hoàn thành
1	Làm tài liệu chương 1	Lê Đức Thiện	2/5/2025
2	Làm tài liệu chương Chương 2	Hoàng Sơn Lâm	2/5/2025
3	Huấn luyện, chụp ảnh huấn luyện	Phùng Bá Huy	5/5/2025
4	Phân công nhiệm vụ, sinh 50000 ảnh và nhãn, hướng dẫn huấn luyện, tổng hợp báo cáo	Đinh Quang Hưng	11/5/2025
5	Làm tài liệu chương 3	Hoàng Văn Nam	10/5/2025

MỤC LỤC

PHÂN CÔNG CÔNG VIỆC 2
MỤC LỤC 3
DANH MỤC CÁC HÌNH VẼ 5
DANH MỤC CÁC BẢNG BIỂU 6
DANH MỤC CÁC TỪ VIẾT TẮT 7
PHẦN MỞ ĐẦU 8
CHƯƠNG 1: GIỚI THIỆU BÀI TOÁN VÀ CÔNG NGHỆ ÁP DỤNG 9
1.1. Bài toán nhận dạng ký tự quang học – OCR 9
1.1.1. Định nghĩa OCR 9
1.1.2. Tầm quan trọng của OCR 9
1.1.3. Các thành phần chính của bài toán OCR 9
1.1.4. Ứng dụng của OCR 10
1.2. Công nghệ Tesseract OCR 11
1.2.1. Tesseract là gì? 11
1.2.2. Lịch sử phát triển 11
1.2.3. Các thành phần quan trọng 11
1.3. Ưu điểm và lý do chọn Tesseract 12
1.3.1. Ưu điểm của Tesseract 12
1.3.2. Lý do chọn Tesseract 12
1.3.3. So sánh Tesseract với các công nghệ OCR khác 13
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VỀ NHẬN DẠNG KÝ TỰ QUANG HỌC 15
2.1. Quy trình tổng quan của OCR 15
2.1.1. Tiền xử lý ảnh 17
2.1.2. Phân đoạn ký tự 19
2.1.3. Nhận dạng ký tự 21
2.2. Kiến trúc LSTM và ứng dụng trong OCR 21
2.2.1. Giới thiệu 21
2.2.2. Kiến trúc LSTM 21
2.2.3. LSTM trong Tesseract OCR 26
2.3. Cơ chế huấn luyện trong Tesseract 27

2.3.1. Tổng quan về quá trình huấn luyện....28
2.3.2. File cấu hình và dữ liệu phụ trợ....30
2.3.3. Các lệnh và công cụ huấn luyện....31
CHƯƠNG 3: THỰC NGHIỆM, ĐÁNH GIÁ VÀ KẾT QUẢ MÔ HÌNH....35
3.1. Chuẩn bị dữ liệu huấn luyện....35
3.2. Cấu hình huấn luyện....35
3.3. Sử dụng mô hình và đánh giá kết quả....37
KẾT LUẬN....**40**
TÀI LIỆU THAM KHẢO....**41**

DANH MỤC CÁC HÌNH VẼ

Hình 1: Quy trình tổng quan của OCR 14
Hình 2: Sơ đồ mạng nơ-ron hồi quy 20
Hình 3: Kiến trúc LSTM 22
Hình 4: Các phép tính tại cổng đầu vào, cổng quên và cổng đầu ra trong một đơn vị LSTM 23
Hình 5: Các phép tính toán trong ô nhớ tiềm năng của LSTM 24
Hình 6: Các phép tính toán trong ô nhớ của LSTM 24
Hình 7: Các phép tính của trạng thái ẩn 25
Hình 8: Cấu trúc thư mục 36
Hình 9: Vòng lặp huấn luyện 36
Hình 10: Ảnh trong tập huấn luyện 38
Hình 11: Kết quả với ảnh trong tập huấn luyện 38
Hình 13: Kết quả cho ảnh chưa có trong tập huấn luyện 39

DANH MỤC CÁC BẢNG BIỂU

Bảng 1: Bảng so sánh Tesseract với các công nghệ khác.....13
Bảng 2: Tham số và giải thích cho công cụ text2image.....32
Bảng 3: Tham số và giải thích cho sinh tệp .lstmf.....33
Bảng 4: Tham số và giải thích cho công cụ lstmtraining.....33
Bảng 5: Quy trình huấn luyện.....35
Bảng 6: Thành phần và ý nghĩa mỗi vòng lặp huấn luyện.....38

DANH MỤC CÁC TỪ VIẾT TẮT

Từ viết tắt	Giải thích
CER	Character Error Rate/Tỷ lệ lỗi ký tự
WER	Word Error Rate/Tỷ lệ lỗi từ
OCR	Optical Character Recognition/Nhận dạng ký tự quang học
LSTM	Long Short-Term Memory/Một loại mạng nơ-ron hồi tiếp
RNN	Recurrent Neural Networks/Mạng nơ-ron hồi tiếp
CTC	Connectionist Temporal Classification/Phương pháp huấn luyện mạng nơ-ron để xử lý dữ liệu tuần tự không có căn chỉnh giữa đầu vào và đầu ra.

PHẦN MỞ ĐẦU

Trong thời đại công nghệ số, khối lượng dữ liệu văn bản dưới dạng hình ảnh ngày càng gia tăng, từ sách, tài liệu in, hóa đơn, đến các biểu mẫu hành chính. Việc chuyển đổi dữ liệu từ ảnh sang dạng văn bản có thể chỉnh sửa được (text) là một nhu cầu thiết yếu, đặc biệt trong các lĩnh vực như lưu trữ, tìm kiếm thông tin, và phân tích dữ liệu. Công nghệ Nhận diện Chữ viết Quang học (Optical Character Recognition - OCR) đã trở thành một công cụ quan trọng để đáp ứng nhu cầu này, cho phép tự động hóa quá trình số hóa văn bản từ hình ảnh.

Trong bối cảnh đó, Tesseract, một phần mềm OCR mã nguồn mở được phát triển bởi Google, nổi bật với khả năng nhận diện văn bản mạnh mẽ và hỗ trợ huấn luyện mô hình tùy chỉnh. Điều này đặc biệt phù hợp với các ngôn ngữ hoặc font chữ đặc thù, chẳng hạn như tiếng Việt với các ký tự có dấu phức tạp. Tuy nhiên, để đạt hiệu quả caoKể từ đó, việc xây dựng một mô hình OCR tùy chỉnh, đặc biệt cho tiếng Việt, vẫn còn là một thách thức cần được nghiên cứu và giải quyết. Đề tài này tập trung vào việc xây dựng và huấn luyện một mô hình OCR sử dụng Tesseract, nhằm nâng cao độ chính xác trong nhận diện văn bản tiếng Việt từ ảnh văn bản in, đồng thời đánh giá hiệu quả của mô hình thông qua các chỉ số chuẩn.

Sự bùng nổ của dữ liệu số trong thời đại 4.0 đã làm gia tăng nhu cầu số hóa các tài liệu vật lý sang dạng kỹ thuật số. Các tài liệu như sách, báo, hóa đơn, hoặc hợp đồng thường tồn tại dưới dạng hình ảnh hoặc bản in, khiến việc trích xuất thông tin trở nên khó khăn và tốn thời gian nếu thực hiện thủ công. Công nghệ OCR ra đời như một giải pháp hiệu quả, cho phép chuyển đổi hình ảnh văn bản thành dạng text có thể chỉnh sửa và tìm kiếm.

Trong số các công cụ OCR hiện có, Tesseract là một phần mềm mã nguồn mở nổi bật nhờ tính linh hoạt và khả năng huấn luyện mô hình tùy chỉnh. Đặc biệt, với các ngôn ngữ có đặc điểm riêng biệt như tiếng Việt, việc sử dụng mô hình OCR tiêu chuẩn có thể không đạt độ chính xác cao do sự khác biệt về font chữ, dấu thanh, và cách trình bày văn bản. Do đó, cần thiết phải nghiên cứu và phát triển một giải pháp OCR tùy chỉnh, tối ưu hóa cho các đặc thù của ngôn ngữ và font chữ tiếng Việt, nhằm nâng cao hiệu quả nhận diện và đáp ứng nhu cầu thực tiễn.

CHƯƠNG 1: GIỚI THIỆU BÀI TOÁN VÀ CÔNG NGHỆ ÁP DỤNG

1.1. Bài toán nhận dạng ký tự quang học – OCR

1.1.1. Định nghĩa OCR

Nhận dạng ký tự quang học là một công nghệ cho phép chuyển đổi hình ảnh chứa văn bản, chẳng hạn như tài liệu in, ảnh chụp, hoặc bản quét, thành dữ liệu văn bản số có thể chỉnh sửa, tìm kiếm và xử lý trên máy tính. Quá trình này bao gồm việc phân tích hình ảnh để nhận diện các ký tự, từ, hoặc cụm từ, sau đó chuyển đổi chúng thành định dạng văn bản kỹ thuật số, chẳng hạn như chuỗi ký tự ASCII hoặc Unicode. OCR không chỉ đơn thuần là việc "đọc" văn bản mà còn liên quan đến việc hiểu cấu trúc và ngữ cảnh của văn bản để đảm bảo độ chính xác cao.

Công nghệ OCR hoạt động dựa trên sự kết hợp của các kỹ thuật xử lý hình ảnh, học máy, và nhận dạng mẫu. Nó đóng vai trò quan trọng trong việc số hóa thông tin, giúp tự động hóa các quy trình xử lý dữ liệu vốn đòi hỏi nhiều công sức nếu thực hiện thủ công. Đặc biệt, với các ngôn ngữ có ký tự phức tạp như tiếng Việt, OCR cần được tối ưu hóa để nhận diện chính xác các dấu thanh và font chữ đa dạng.

1.1.2. Tầm quan trọng của OCR

Trong thời đại công nghệ số, khối lượng dữ liệu văn bản dưới dạng hình ảnh ngày càng gia tăng, từ sách cổ, tài liệu hành chính, hóa đơn, đến các biển quảng cáo hoặc biển số xe. Việc chuyển đổi các dữ liệu này sang dạng văn bản số không chỉ giúp tiết kiệm thời gian và chi phí mà còn mở ra khả năng ứng dụng trong nhiều lĩnh vực. OCR là cầu nối giữa thế giới vật lý (tài liệu in) và thế giới số, cho phép khai thác thông tin một cách hiệu quả hơn.

1.1.3. Các thành phần chính của bài toán OCR

Bài toán OCR thường bao gồm các phần sau:

Thu nhận hình ảnh: Hình ảnh chứa văn bản được thu thập từ các nguồn như máy quét, máy ảnh hoặc tệp PDF. Chất lượng hình ảnh đóng vai trò quan trọng, ảnh hưởng trực tiếp đến độ chính xác của quá trình nhận dạng ký tự quang học (OCR). Hình ảnh chất lượng thấp có thể gây khó khăn cho việc nhận diện chính xác các ký tự hoặc từ.

Tiền xử lý hình ảnh: Trước khi tiến hành nhận diện, hình ảnh được tiền xử lý để cải thiện chất lượng. Các bước bao gồm chuyển đổi hình ảnh sang thang độ xám hoặc nhị phân, khử nhiễu, điều chỉnh độ sáng và độ tương phản, cũng như căn chỉnh hình ảnh. Những thao tác này giúp tăng cường độ rõ nét, tạo điều kiện thuận lợi cho các giai đoạn phân đoạn và nhận diện tiếp theo.

Phân đoạn văn bản: Ở giai đoạn này, các vùng chứa văn bản trong hình ảnh được xác định và phân chia thành các dòng, từ hoặc ký tự. Các phương pháp phổ biến bao gồm sử dụng histogram, phép biến đổi Hough hoặc phân tích thành phần kết nối. Quá trình phân đoạn đảm bảo rằng văn bản được tổ chức đúng cách để thuận tiện cho việc nhận diện.

Nhận diện ký tự: Việc nhận diện ký tự dựa trên các thuật toán học máy hoặc học sâu, chẳng hạn như sự kết hợp giữa mạng nơ-ron tích chập (CNN) và mạng nơ-ron hồi tiếp (LSTM). Các mô hình này dự đoán ký tự hoặc từ dựa trên các đặc trưng được trích xuất từ hình ảnh. Đặc biệt, với các ngôn ngữ phức tạp như tiếng Việt, các mô hình cần được huấn luyện để xử lý tốt các dấu thanh và ký tự đặc biệt.

Hậu xử lý: Sau khi nhận diện, văn bản có thể chứa lỗi do hạn chế của quá trình OCR. Giai đoạn hậu xử lý sử dụng từ điển, quy tắc ngữ pháp hoặc các mô hình ngôn ngữ để sửa lỗi và cải thiện độ chính xác. Kết quả cuối cùng là văn bản đầu ra có ý nghĩa, sát với nội dung gốc của hình ảnh.

1.1.4. Ứng dụng của OCR

Nhận dạng ký tự quang học có phạm vi ứng dụng rộng rãi, mang lại giá trị lớn trong nhiều lĩnh vực nhờ khả năng chuyển đổi văn bản từ hình ảnh thành dữ liệu số. Trong quản lý tài liệu, OCR được sử dụng để số hóa sách, báo, và tài liệu lịch sử, cho phép lưu trữ, tìm kiếm và phân tích dễ dàng. Ví dụ, các thư viện áp dụng OCR để chuyển sách cổ thành dạng kỹ thuật số, góp phần bảo tồn và phổ biến tri thức. Tương tự, trong trích xuất dữ liệu tự động, OCR hỗ trợ rút trích thông tin từ hóa đơn, biên lai, biểu mẫu, hoặc giấy tờ tùy thân như chứng minh nhân dân, giúp các ngân hàng và tổ chức hành chính tự động hóa quy trình nhập liệu, giảm thiểu sai sót và tiết kiệm thời gian.

Ứng dụng của OCR còn mở rộng sang lĩnh vực giao thông và an ninh, đặc biệt trong nhận diện biển số xe, khi camera sử dụng OCR để tự động ghi lại biển số trong thời gian thực, hỗ trợ quản lý giao thông và tăng cường an ninh. Trong hỗ trợ người khiếm thị, OCR chuyển đổi văn bản in thành giọng nói hoặc chữ nổi thông qua các thiết bị hỗ trợ, giúp người khiếm thị tiếp cận thông tin một cách độc lập và hiệu quả hơn. Ngoài ra, OCR kết hợp với công nghệ dịch máy trong dịch thuật, cho phép trích xuất và dịch văn bản từ hình ảnh, hỗ trợ giao tiếp đa ngôn ngữ trong các tình huống thực tế.

Trong công nghiệp, OCR được ứng dụng để đọc mã số, nhãn sản phẩm hoặc thông tin in trên bao bì trong dây chuyền sản xuất, đảm bảo quy trình tự động hóa chính xác và hiệu quả. Trong giáo dục, OCR giúp chuyển đổi ghi chú viết tay hoặc tài liệu in thành văn bản số, hỗ trợ lưu trữ, chia sẻ và quản lý tài liệu học tập. Nhờ

tính linh hoạt và khả năng tích hợp, OCR không chỉ nâng cao hiệu suất trong nhiều lĩnh vực mà còn mở ra tiềm năng cho các ứng dụng sáng tạo trong tương lai.

1.2. Công nghệ Tesseract OCR

1.2.1. Tesseract là gì?

Tesseract là một phần mềm nhận dạng ký tự quang học (OCR) mã nguồn mở, được thiết kế để chuyển đổi hình ảnh chứa văn bản thành dữ liệu văn bản số. Được phát triển ban đầu bởi Hewlett-Packard vào những năm 1980, Tesseract đã trở thành một trong những công cụ OCR phổ biến nhất nhờ tính miễn phí, khả năng tùy chỉnh cao, và hỗ trợ nhiều ngôn ngữ, bao gồm cả tiếng Việt. Tesseract có thể xử lý các loại hình ảnh văn bản từ tài liệu in, ảnh chụp, đến bản quét, và được sử dụng rộng rãi trong các dự án học thuật, thương mại, và nghiên cứu.

Tesseract không chỉ là một công cụ OCR độc lập mà còn là một thư viện có thể tích hợp vào các ứng dụng thông qua các API như Pytesseract hoặc giao diện dòng lệnh. Với khả năng huấn luyện mô hình tùy chỉnh, Tesseract đặc biệt phù hợp cho các ngôn ngữ hoặc font chữ đặc thù, chẳng hạn như tiếng Việt với các ký tự có dấu thanh phức tạp.

1.2.2. Lịch sử phát triển

Tesseract đã trải qua hành trình phát triển đáng kể, từ một dự án nội bộ của HP (1985-1995) tập trung vào nhận dạng mẫu truyền thống, đến một công cụ OCR mã nguồn mở được sử dụng rộng rãi toàn cầu. Năm 1995, HP mở mã nguồn Tesseract nhưng không thu hút nhiều chú ý do hạn chế về hiệu suất. Đến năm 2005, Tesseract được phát hành công khai dưới giấy phép Apache 2.0, và từ năm 2006, Google tiếp quản, cải tiến mạnh mẽ qua các phiên bản 2.x, 3.x, mở rộng hỗ trợ ngôn ngữ và nâng cao độ chính xác. Năm 2018, Tesseract 4.0 tích hợp mạng nơ-ron tái phát với đơn vị LSTM, cải thiện khả năng nhận diện văn bản phức tạp.

Tính đến năm 2025, Tesseract 5.x, được duy trì bởi Google và cộng đồng mã nguồn mở, tiếp tục tối ưu hóa tốc độ, độ chính xác và khả năng huấn luyện mô hình tùy chỉnh, đặc biệt cho các ngôn ngữ có ký tự đặc biệt như tiếng Việt.

1.2.3. Các thành phần quan trọng

Tesseract bao gồm nhiều thành phần cốt lõi hỗ trợ quá trình nhận diện và huấn luyện:

lstmtraining: Công cụ dòng lệnh dùng để huấn luyện mô hình LSTM trên dữ liệu tùy chỉnh. Công cụ này cho phép người dùng tạo mô hình nhận diện ký tự cho các font chữ hoặc ngôn ngữ không có trong tệp traineddata mặc định.

traineddata: Tệp dữ liệu chứa thông tin huấn luyện cho từng ngôn ngữ, bao gồm mô hình nhận diện ký tự, thông tin font chữ, và từ điển ngôn ngữ. Mỗi ngôn ngữ có một tệp traineddata riêng.

lstmf: Định dạng tệp trung gian được tạo trong quá trình huấn luyện, chứa thông tin về hình ảnh và văn bản chuẩn (ground truth). Tệp lstmf được sử dụng để cung cấp dữ liệu đầu vào cho quá trình huấn luyện LSTM.

tessdata: Thư mục chứa các tệp traineddata và các tài nguyên khác cần thiết để chạy Tesseract. Người dùng có thể tải hoặc tạo tệp traineddata tùy chỉnh cho ngôn ngữ hoặc font chữ cụ thể.

1.3. Ưu điểm và lý do chọn Tesseract

1.3.1. Ưu điểm của Tesseract

Tesseract là một công cụ nhận dạng ký tự quang học nổi bật với nhiều ưu điểm vượt trội, khiến nó trở thành lựa chọn hàng đầu cho các dự án học thuật và ứng dụng tùy chỉnh. Là phần mềm mã nguồn mở được cung cấp miễn phí dưới giấy phép Apache 2.0, Tesseract cho phép người dùng sử dụng, chỉnh sửa và tích hợp mà không tốn chi phí bản quyền, đặc biệt phù hợp với các dự án có ngân sách hạn chế. Công cụ này dễ dàng tích hợp với nhiều ngôn ngữ lập trình như Python, C++, và Java, đồng thời hỗ trợ giao diện dòng lệnh cho phép sử dụng trực tiếp mà không cần cấu hình phức tạp.

Tesseract cho phép huấn luyện mô hình tùy chỉnh trên dữ liệu riêng, rất hữu ích cho các ngôn ngữ đặc thù như tiếng Việt hoặc phông chữ không phổ biến, với khả năng đạt hiệu quả cao chỉ với vài nghìn mẫu dữ liệu, tiết kiệm thời gian và tài nguyên so với các mô hình học sâu khác. Với hỗ trợ hơn 100 ngôn ngữ, bao gồm tiếng Việt, Tesseract có thể xử lý văn bản đa ngôn ngữ trong cùng một hình ảnh, đáp ứng nhu cầu đa dạng.

Cộng đồng phát triển mạnh mẽ cung cấp tài liệu phong phú, hướng dẫn chi tiết, mẫu dữ liệu huấn luyện và diễn đàn thảo luận trên GitHub, hỗ trợ người dùng hiệu quả. Đặc biệt, Tesseract tỏ ra vượt trội khi xử lý văn bản in chất lượng cao như sách, báo, hoặc biểu mẫu hành chính, phù hợp với trọng tâm của các dự án OCR tập trung vào tài liệu in, mang lại độ chính xác cao và tính ứng dụng rộng rãi.

1.3.2. Lý do chọn Tesseract

Tesseract được chọn làm công nghệ chính cho đề tài này nhờ vào các đặc điểm nổi bật, đáp ứng tốt các yêu cầu cụ thể của dự án nhận dạng ký tự quang học (OCR). Trước hết, Tesseract hỗ trợ tiếng Việt một cách hiệu quả, cho phép huấn luyện mô hình tùy chỉnh để cải thiện độ chính xác khi xử lý các ký tự đặc thù như â, ê, ơ và các

dấu thanh (huyền, sắc, hỏi, ngã, nặng), cũng như các phông chữ đặc trưng trong tài liệu tiếng Việt như sách giáo khoa, báo chí hay giấy tờ hành chính.

Là một giải pháp mã nguồn mở và miễn phí, Tesseract phù hợp với các dự án học thuật hoặc nghiên cứu có ngân sách hạn chế, nhưng vẫn mang lại hiệu suất cạnh tranh so với các công cụ thương mại. Tính linh hoạt của Tesseract thể hiện qua khả năng tích hợp dễ dàng với Python và các thư viện xử lý hình ảnh như OpenCV, cho phép xây dựng một pipeline OCR hoàn chỉnh từ tiền xử lý đến hậu xử lý.

Ngoài ra, Tesseract có khả năng mở rộng nhờ hỗ trợ huấn luyện thêm dữ liệu để nâng cao độ chính xác và khả năng kết hợp với các công nghệ học sâu như mạng nơ-ron tích chập, giúp xử lý hiệu quả các trường hợp phức tạp. Cuối cùng, với tài liệu hướng dẫn phong phú và sự hỗ trợ từ cộng đồng mã nguồn mở tích cực trên các nền tảng như GitHub, Tesseract dễ dàng được triển khai, tối ưu hóa và tùy chỉnh theo nhu cầu cụ thể, khiến nó trở thành lựa chọn lý tưởng cho đề tài này.

1.3.3. So sánh Tesseract với các công nghệ OCR khác

Để đánh giá Tesseract trong bối cảnh các công nghệ OCR khác, bảng dưới đây so sánh Tesseract với Google Cloud Vision API và EasyOCR dựa trên các tiêu chí quan trọng:

Bảng 1: Bảng so sánh Tesseract với các công nghệ khác

Tiêu chí	**Tesseract**	**Google Cloud Vision**	**EasyOCR**
Loại	Mã nguồn mở, miễn phí	Dịch vụ API thương mại	Mã nguồn mở, miễn phí
Hỗ trợ ngôn ngữ	Hơn 100 ngôn ngữ, có thể tùy chỉnh	Hỗ trợ hơn 100 ngôn ngữ	Nhiều ngôn ngữ, tập trung vào học sâu
Độ chính xác	Tốt sau huấn luyện, phụ thuộc dữ liệu	Rất cao, đặc biệt với văn bản phức tạp	Tốt với văn bản in, hạn chế với viết tay
Khả năng tùy chỉnh	Cao, hỗ trợ huấn luyện mô hình riêng	Thấp, không hỗ trợ huấn luyện tùy chỉnh	Trung bình, hỗ trợ huấn luyện hạn chế
Tích hợp	Dễ dàng với	API REST, cần	Dễ dàng với

	Python, C++, CLI	kết nối internet	Python
Chi phí	Miễn phí	Tính phí theo số lượng yêu cầu	Miễn phí
Yêu cầu phần cứng	Chạy trên máy tính cá nhân	Yêu cầu kết nối đám mây	Yêu cầu GPU để huấn luyện
Phù hợp với	Dự án học thuật, tùy chỉnh ngôn ngữ	Ứng dụng thương mại quy mô lớn	Dự án nghiên cứu học sâu

Nhận xét:

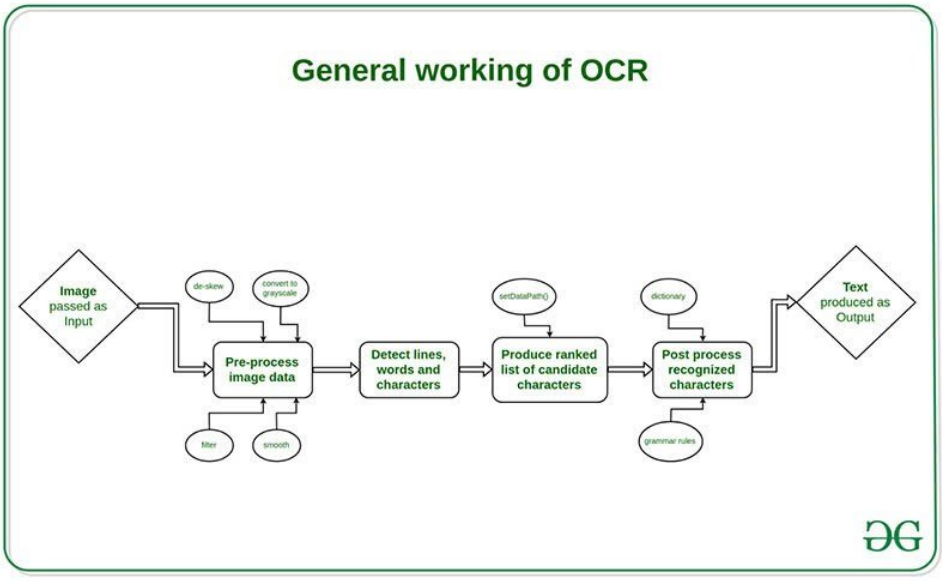
Tesseract: Nổi bật về chi phí (miễn phí), khả năng tùy chỉnh, và phù hợp với các dự án cần huấn luyện mô hình cho ngôn ngữ hoặc font chữ đặc thù như tiếng Việt.

Google Cloud Vision: Có độ chính xác cao, dễ sử dụng, và phù hợp với các ứng dụng thương mại quy mô lớn. Tuy nhiên, chi phí sử dụng và yêu cầu kết nối internet là những hạn chế lớn.

EasyOCR: Sử dụng công nghệ học sâu hiện đại, dễ tích hợp với Python, nhưng yêu cầu phần cứng mạnh (GPU) để huấn luyện và khả năng tùy chỉnh không linh hoạt bằng Tesseract.

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT VỀ NHẬN DẠNG KÝ TỰ QUANG HỌC

2.1. Quy trình tổng quan của OCR



Hình 1: Quy trình tổng quan của OCR

Quy trình xử lý ảnh văn bản từ đầu vào đến đầu ra text trong hệ thống OCR sử dụng Tesseract như sau sau:

Đầu vào (Input Image): Quá trình nhận dạng ký tự quang học (OCR) bắt đầu với đầu vào là hình ảnh chứa văn bản, thường ở các định dạng phổ biến như JPEG, PNG, hoặc PDF. Những hình ảnh này có thể là tài liệu quét, ảnh chụp từ máy ảnh, hoặc các tệp số hóa từ sách, hóa đơn, hoặc biển quảng cáo. Chất lượng của hình ảnh đầu vào đóng vai trò quan trọng, vì độ phân giải thấp, nhiễu, hoặc văn bản bị méo mó có thể ảnh hưởng đến độ chính xác của quá trình nhận diện. Các hệ thống OCR, như Tesseract, hỗ trợ nhiều định dạng hình ảnh và cần được cấu hình phù hợp để xử lý các đặc điểm riêng biệt của từng loại đầu vào, đảm bảo rằng văn bản trong hình ảnh có thể được nhận diện hiệu quả.

*Tiền xử lý hình ảnh (Pre-process image data)***:** Tiền xử lý hình ảnh là bước quan trọng nhằm cải thiện chất lượng hình ảnh trước khi tiến hành nhận diện văn bản. Quá trình này bao gồm chuyển đổi hình ảnh sang thang độ xám hoặc nhị phân hóa để giảm độ phức tạp tính toán và làm nổi bật các ký tự. Các bộ lọc như Gaussian blur hoặc median filter được sử dụng để loại bỏ nhiễu, đặc biệt trong các hình ảnh chất lượng thấp. Ngoài ra, việc điều chỉnh độ sáng và độ tương phản giúp văn bản trở nên rõ ràng hơn, trong khi kỹ thuật xoay hoặc căn chỉnh ảnh (deskewing) sửa các lỗi lệch hướng do góc chụp hoặc quét không chính xác. Các bước tiền xử lý này đảm bảo

rằng hình ảnh được tối ưu hóa, tạo điều kiện thuận lợi cho các giai đoạn tiếp theo trong quy trình OCR.

*Phân đoạn văn bản (Text Segmentation)***:** Phân đoạn văn bản là quá trình xác định và tách các vùng chứa văn bản trong hình ảnh để chuẩn bị cho việc nhận diện. Đầu tiên, hệ thống xác định các vùng văn bản (text regions) để loại bỏ các phần không liên quan như hình ảnh hoặc đồ họa. Sau đó, văn bản được chia nhỏ thành các dòng (lines), từ (words), và ký tự (characters) thông qua các thuật toán phân tích bố cục. Với các hình ảnh có bố cục phức tạp, như bảng biểu hoặc văn bản đa cột, việc phân đoạn đòi hỏi các kỹ thuật nâng cao để đảm bảo tách biệt chính xác. Quá trình này rất quan trọng vì nó giúp hệ thống OCR tập trung vào từng đơn vị văn bản, cải thiện độ chính xác của bước nhận diện ký tự tiếp theo.

Nhận diện ký tự : Nhận diện ký tự là giai đoạn cốt lõi của OCR, nơi các ký tự hoặc từ được dự đoán dựa trên đặc trưng của hình ảnh. Trong các hệ thống hiện đại như Tesseract, mô hình LSTM được sử dụng để xử lý chuỗi đặc trưng trích xuất từ hình ảnh, thường kết hợp với mạng nơ-ron tích chập . LSTM học các mối quan hệ ngữ cảnh giữa các ký tự, giúp dự đoán chính xác hơn, đặc biệt với các ngôn ngữ phức tạp như tiếng Việt .Việc áp dụng thông tin ngữ cảnh, chẳng hạn như thứ tự ký tự hoặc mẫu ngôn ngữ, giúp cải thiện độ chính xác, đặc biệt trong các trường hợp văn bản bị biến dạng hoặc viết tay.

Hậu xử lý: Hậu xử lý là bước cuối cùng nhằm tinh chỉnh kết quả nhận diện để đảm bảo văn bản đầu ra có ý nghĩa và chính xác. Quá trình này bao gồm việc sử dụng từ điển để kiểm tra và sửa lỗi nhận diện, chẳng hạn như các ký tự bị nhầm lẫn do hình ảnh chất lượng thấp. Các quy tắc ngữ pháp và mô hình ngôn ngữ được áp dụng để đảm bảo rằng văn bản tuân theo cấu trúc ngôn ngữ tự nhiên, ví dụ, sửa lỗi chính tả hoặc định dạng câu. Kết quả cuối cùng có thể được định dạng thành các định dạng như văn bản thuần (text), HTML (hOCR), hoặc PDF, tùy thuộc vào yêu cầu ứng dụng. Hậu xử lý đóng vai trò quan trọng trong việc nâng cao chất lượng và tính ứng dụng của văn bản được trích xuất

Đầu ra: Đầu ra của quá trình OCR là chuỗi văn bản số có thể chỉnh sửa, tìm kiếm, hoặc tích hợp vào các ứng dụng khác. Văn bản này là kết quả của việc chuyển đổi hình ảnh chứa văn bản thành định dạng kỹ thuật số, chẳng hạn như chuỗi ký tự trong tệp văn bản, cơ sở dữ liệu, hoặc hệ thống quản lý nội dung. Đầu ra có thể được sử dụng trong các ứng dụng như số hóa tài liệu, tự động hóa nhập liệu, hoặc hỗ trợ người khiếm thị. Chất lượng của đầu ra phụ thuộc vào hiệu quả của các bước tiền xử lý, phân đoạn, nhận diện, và hậu xử lý, đảm bảo rằng văn bản được trích xuất chính xác và phù hợp với mục đích sử dụng.

Mỗi bước trong quy trình này đều có những thách thức và kỹ thuật riêng, đòi hỏi sự kết hợp của nhiều lĩnh vực từ xử lý ảnh, học máy đến ngôn ngữ học.

2.1.1. Tiền xử lý ảnh

Tiền xử lý ảnh là một trong những bước quan trọng nhất trong quy trình OCR, quyết định đến chất lượng của các bước tiếp theo. Mục đích chính của tiền xử lý là loại bỏ các yếu tố nhiễu, tăng cường chất lượng ảnh, và chuẩn hóa dữ liệu đầu vào.

Chuyển đổi grayscale

Chuyển đổi ảnh màu thành ảnh xám là bước đầu tiên trong hầu hết các hệ thống OCR. Công thức phổ biến để chuyển đổi ảnh RGB sang grayscale là:

$$Grayscale = 0.299 * R + 0.587 * G + 0.114 * B$$

Trong đó R, G, B lần lượt là giá trị của các kênh màu đỏ, xanh lá và xanh dương. Quá trình này giảm độ phức tạp tính toán bằng cách loại bỏ thông tin màu sắc không cần thiết cho nhận dạng ký tự, đồng thời chuẩn bị hình ảnh cho các bước phân ngưỡng sau đó. Chuyển đổi grayscale giúp tập trung vào cường độ sáng của pixel, làm nổi bật các đặc trưng của văn bản so với nền, tạo điều kiện thuận lợi cho các giai đoạn tiếp theo trong quy trình OCR.

Khử nhiễu

Khử nhiễu là một bước quan trọng trong tiền xử lý ảnh OCR, nhằm loại bỏ các yếu tố nhiễu có thể xuất hiện từ quá trình quét, chụp ảnh hoặc do chất lượng kém của tài liệu. Các phương pháp khử nhiễu phổ biến bao gồm lọc trung vị, hiệu quả trong việc loại bỏ nhiễu muối tiêu bằng cách thay thế giá trị pixel bằng trung vị của các pixel lân cận; lọc Gaussian, sử dụng hàm làm mờ để giảm nhiễu Gaussian; lọc song phương, giữ lại các cạnh của văn bản trong khi loại bỏ nhiễu; và biến đổi sóng con, phân tích ảnh ở nhiều tỷ lệ để phát hiện và loại bỏ nhiễu một cách tinh vi. Những kỹ thuật này giúp làm sạch hình ảnh, đảm bảo các ký tự trở nên rõ ràng hơn và giảm thiểu sai sót trong các bước phân đoạn và nhận diện sau này.

Điều chỉnh độ tương phản

Điều chỉnh độ tương phản được thực hiện để tăng cường sự phân biệt giữa văn bản và nền, giúp văn bản nổi bật hơn trong hình ảnh. Các kỹ thuật phổ biến bao gồm cân bằng histogram, phân phối lại các mức xám để tối ưu hóa phạm vi giá trị pixel; kéo dãn tương phản, mở rộng khoảng giá trị pixel để tăng độ tương phản; và điều

chỉnh gamma, thay đổi độ sáng theo cách phi tuyến tính để cải thiện khả năng nhận diện. Bằng cách làm nổi bật các ký tự so với nền, đặc biệt trong các hình ảnh có độ tương phản thấp hoặc ánh sáng không đồng đều, bước này nâng cao chất lượng hình ảnh, tạo điều kiện thuận lợi cho việc phân ngưỡng và phân đoạn văn bản trong quy trình OCR

Phân ngưỡng (Thresholding)

Phân ngưỡng là quá trình chuyển đổi ảnh xám thành ảnh nhị phân, chỉ chứa các giá trị đen và trắng, nhằm tách biệt văn bản khỏi nền một cách rõ ràng. Các phương pháp phân ngưỡng bao gồm phân ngưỡng toàn cục, sử dụng một ngưỡng cố định cho toàn bộ ảnh; phương pháp Otsu, tự động tìm ngưỡng tối ưu dựa trên phân phối histogram để tối đa hóa sự khác biệt giữa văn bản và nền; ngưỡng cố định, áp dụng một giá trị ngưỡng được chọn trước; và các phương pháp cục bộ như Niblack hoặc Sauvola, tính toán ngưỡng dựa trên trung bình và phương sai cục bộ, thích ứng tốt với các vùng có độ tương phản thay đổi. Bước phân ngưỡng giúp đơn giản hóa hình ảnh, giảm nhiễu và chuẩn bị dữ liệu cho quá trình phân đoạn và nhận diện ký tự.

Hiệu chỉnh độ nghiêng

Hiệu chỉnh độ nghiêng được thực hiện để sửa các lỗi lệch hướng trong hình ảnh, thường xảy ra khi tài liệu được quét hoặc chụp không thẳng, gây khó khăn cho việc phân đoạn và nhận diện văn bản. Các kỹ thuật phổ biến bao gồm biến đổi Hough, phát hiện các đường thẳng trong hình ảnh để xác định góc nghiêng của văn bản; phân tích phân phối chiều cao hàng, dựa trên khoảng cách giữa các dòng văn bản để ước lượng góc lệch; và tính moment quán tính, xác định hướng chính của văn bản thông qua phân tích hình học. Bằng cách căn chỉnh lại hình ảnh theo hướng chuẩn, bước này đảm bảo văn bản được sắp xếp đúng, giúp cải thiện độ chính xác của các bước phân đoạn dòng và nhận diện ký tự sau đó.

Resize và chuẩn hóa kích thước

Resize và chuẩn hóa kích thước ảnh là bước cuối cùng trong tiền xử lý, nhằm tối ưu hóa thời gian xử lý và chuẩn hóa dữ liệu đầu vào cho các mô hình nhận diện. Quá trình này điều chỉnh kích thước hình ảnh để phù hợp với yêu cầu của hệ thống OCR, đồng thời giữ tỷ lệ phù hợp để tránh làm biến dạng ký tự. Các phương pháp nội suy như song tuyến ,láng giềng gần nhất (nearest neighbor), hoặc bicubic được sử dụng, tùy thuộc vào yêu cầu về tốc độ và chất lượng. Resize giúp giảm tải tính toán cho các thuật toán OCR, trong khi chuẩn hóa kích thước đảm bảo rằng các đặc trưng của

văn bản được giữ nguyên và đồng nhất, tạo điều kiện thuận lợi cho việc phân đoạn và nhận diện ký tự một cách chính xác.

2.1.2. Phân đoạn ký tự

Phân đoạn là quá trình chia nhỏ ảnh đầu vào thành các thành phần có ý nghĩa như vùng văn bản, đoạn văn, dòng, từ và ký tự. Đây là bước quan trọng và cũng là bước khó khăn trong OCR, đặc biệt đối với tài liệu có bố cục phức tạp.

Phân đoạn vùng văn bản

Phân đoạn vùng văn bản là một bước quan trọng trong nhận dạng ký tự quang học, giúp tách biệt các vùng văn bản khỏi hình ảnh, biểu đồ và các thành phần phi văn bản khác. Các phương pháp phổ biến bao gồm RLSA, kết nối các pixel gần nhau trong hình ảnh nhị phân để tạo thành các khối văn bản dựa trên quét ngang và dọc, hiệu quả cho tài liệu đơn giản nhưng nhạy với nhiễu. Phân tích thành phần kết nối nhóm các pixel liền kề thành các đối tượng như ký tự hoặc từ, sau đó phân loại để loại bỏ thành phần phi văn bản, phù hợp với bố cục phức tạp nhưng yêu cầu hình ảnh chất lượng cao.

Phương pháp dựa trên histogram chiều ngang và dọc phát hiện vùng văn bản bằng cách phân tích sự tập trung pixel qua histogram, xác định các đỉnh tương ứng với dòng hoặc cột văn bản, hữu ích cho tài liệu có tổ chức nhưng kém hiệu quả với văn bản lệch hoặc xen kẽ đồ họa. Các phương pháp này, tùy thuộc vào đặc điểm hình ảnh, có thể được sử dụng riêng lẻ hoặc kết hợp để tối ưu hóa việc tách văn bản, đảm bảo độ chính xác cho các bước tiếp theo trong quy trình OCR.

Phân đoạn dòng

Sau khi xác định vùng văn bản, việc tách văn bản thành các dòng riêng biệt là một bước quan trọng trong quy trình nhận dạng ký tự quang học để chuẩn bị cho việc nhận diện ký tự. Các phương pháp phổ biến bao gồm phương pháp dựa trên histogram chiều ngang, phân tích histogram của pixel đen theo hàng để tìm các "thung lũng" – những vùng có ít pixel, biểu thị khoảng trắng giữa các dòng, phù hợp với tài liệu có bố cục đều nhưng nhạy với văn bản lệch hoặc khoảng cách dòng không nhất quán. Phương pháp Hough Transform phát hiện các đường thẳng đại diện cho dòng văn bản bằng cách chuyển đổi hình ảnh sang không gian Hough, xác định các đường có mật độ pixel cao, hiệu quả cho văn bản thẳng nhưng phức tạp hơn với văn bản cong hoặc nghiêng.

Phương pháp dựa trên clustering nhóm các thành phần kết nối (như ký tự hoặc từ) thành các dòng dựa trên sự gần gũi về không gian hoặc đặc trưng hình học, linh hoạt với bố cục phức tạp nhưng yêu cầu tiền xử lý tốt để tránh nhầm lẫn với các thành phần phi văn bản. Tùy thuộc vào đặc điểm hình ảnh và độ phức tạp của bố cục, các

phương pháp này có thể được sử dụng riêng lẻ hoặc kết hợp để đảm bảo tách dòng chính xác, hỗ trợ hiệu quả cho các bước tiếp theo trong quy trình OCR

Phân đoạn từ

Sau khi tách dòng văn bản, bước phân đoạn từ được thực hiện để phân tích mỗi dòng và tách thành các từ riêng biệt, tạo điều kiện cho việc nhận diện văn bản chính xác. Phương pháp dựa trên khoảng cách xác định các khoảng trắng giữa các từ bằng cách đo lường khoảng cách giữa các thành phần pixel, tận dụng đặc điểm rằng khoảng cách giữa các từ thường lớn hơn khoảng cách giữa các ký tự trong cùng một từ. Phương pháp histogram chiều dọc phân tích mật độ pixel theo cột trong mỗi dòng, xác định các "thung lũng" hoặc vùng có ít pixel, tương ứng với khoảng trống giữa các từ, đặc biệt hiệu quả với các dòng văn bản có bố cục rõ ràng

Ngoài ra, phân tích khoảng cách giữa các thành phần kết nối phân loại các khoảng cách thành hai loại: khoảng cách trong từ giữa các ký tự và khoảng cách giữa các từ, dựa trên ngưỡng khoảng cách hoặc đặc trưng hình học, giúp tách từ chính xác ngay cả trong các bố cục phức tạp. Các phương pháp này, khi kết hợp, đảm bảo tách từ hiệu quả, đặt nền tảng cho bước phân đoạn ký tự tiếp theo trong quy trình OCR.

Phân đoạn ký tự

Phân đoạn ký tự là bước cuối cùng trong quá trình phân đoạn, nhằm tách mỗi từ thành các ký tự riêng lẻ để chuẩn bị cho nhận diện ký tự. Phương pháp phân tích thành phần kết nối xác định các đối tượng riêng biệt trong từ bằng cách nhóm các pixel liền kề, mỗi nhóm tương ứng với một ký tự, phù hợp với các từ có ký tự tách biệt rõ ràng. Phương pháp dựa trên đường viền tìm các đường bao quanh mỗi ký tự, sử dụng các thuật toán phân tích hình học để xác định ranh giới ký tự, hiệu quả khi ký tự có hình dạng đặc trưng. Phương pháp cắt dọc sử dụng histogram chiều dọc để xác định các vị trí có ít pixel giữa các ký tự, cho phép cắt từ thành các ký tự riêng lẻ dựa trên các "thung lũng" trong histogram.

Tuy nhiên, phân đoạn ký tự đối mặt với nhiều thách thức: ký tự liền nhau khiến việc tách biệt trở nên khó khăn do các ký tự chạm hoặc chồng lấn; ký tự bị đứt đoạn có thể bị chia thành nhiều phần do nhiễu hoặc chất lượng hình ảnh kém; nhiễu và méo mó làm thay đổi hình dạng ký tự, gây sai lệch trong phân đoạn. Đặc biệt, với các ngôn ngữ phức tạp như tiếng Việt, có dấu thanh và dấu phụ, việc phân đoạn ký tự càng trở nên thách thức, đòi hỏi các thuật toán chuyên biệt kết hợp thông tin ngữ cảnh và đặc trưng ngôn ngữ để đảm bảo độ chính xác, hỗ trợ hiệu quả cho quá trình nhận diện trong OCR.

2.1.3. Nhận dạng ký tự

Sau khi phân đoạn, mỗi ký tự được nhận dạng để chuyển từ hình ảnh sang mã số hoặc văn bản. Có nhiều phương pháp nhận dạng ký tự, từ các phương pháp truyền thống đến các kỹ thuật học sâu hiện đại.

2.2. Kiến trúc LSTM và ứng dụng trong OCR

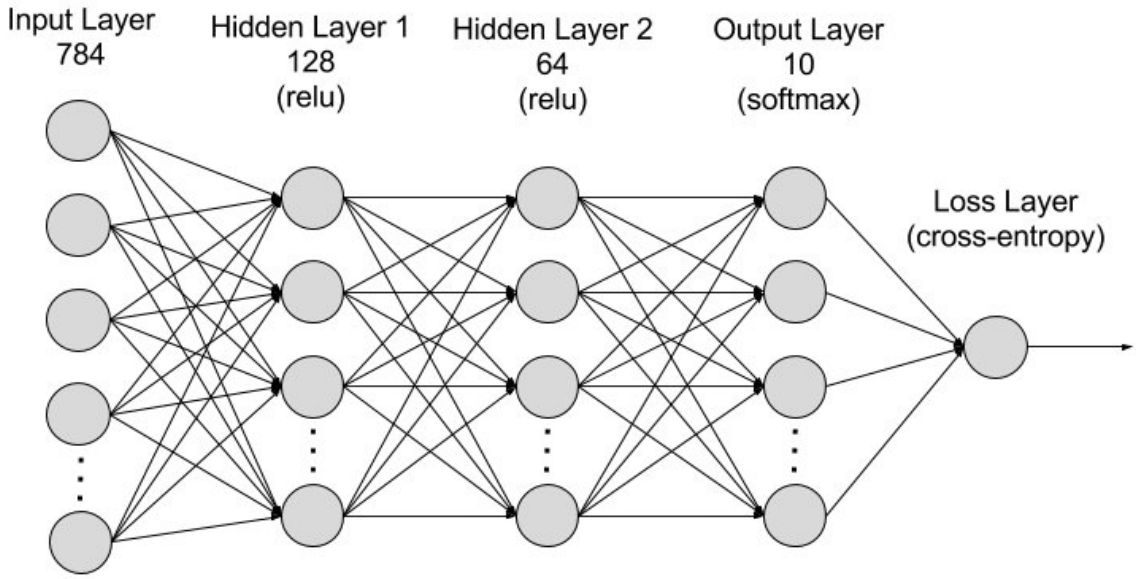
2.2.1. Giới thiệu

Nhận dạng ký tự quang học là quá trình chuyển đổi hình ảnh văn bản thành văn bản kỹ thuật số có thể chỉnh sửa. Công nghệ này đã phát triển mạnh mẽ trong những năm gần đây nhờ vào sự phát triển của các mô hình học sâu, đặc biệt là mạng nơ-ron hồi quy dài-ngắn hạn.

Báo cáo này tập trung vào mạng LSTM và ứng dụng của nó trong lĩnh vực OCR, đặc biệt là trong công cụ Tesseract - một trong những hệ thống OCR mã nguồn mở phổ biến nhất hiện nay. Chúng tôi sẽ phân tích kiến trúc LSTM, ưu điểm của nó so với các mạng nơ-ron hồi quy truyền thống (RNN), và cách LSTM được tích hợp vào quy trình nhận dạng văn bản từ hình ảnh.

2.2.2. Kiến trúc LSTM

Vấn đề với RNN truyền thống



Hình 2: Sơ đồ mạng nơ-ron hồi quy

Mạng nơ-ron hồi quy là một loại mạng nơ-ron được thiết kế đặc biệt để xử lý dữ liệu tuần tự như văn bản, âm thanh, hoặc chuỗi thời gian. Ý tưởng cơ bản của RNN

là sử dụng thông tin từ các bước trước đó để tác động đến quyết định tại bước hiện tại.

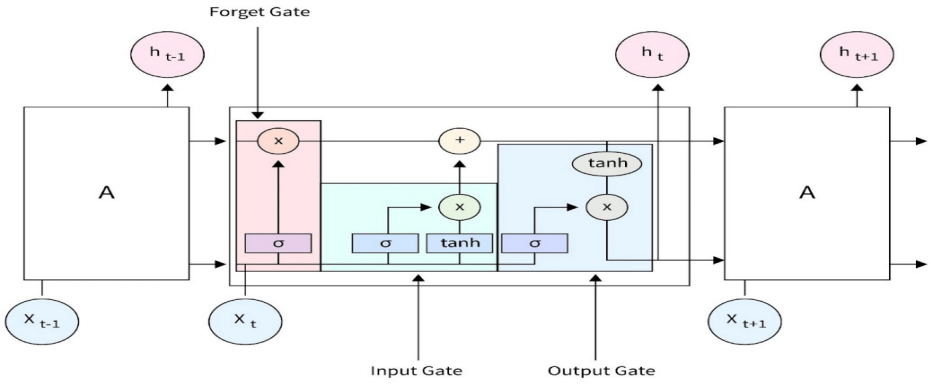
Tuy nhiên, RNN truyền thống gặp phải vấn đề nghiêm trọng khi xử lý chuỗi dài: vấn đề mất mát gradient. Khi thông tin phải truyền qua nhiều bước thời gian, gradient có xu hướng trở nên rất nhỏ, khiến mạng không thể học được các phụ thuộc dài hạn. Trong OCR, điều này có nghĩa là RNN khó có thể liên kết các ký tự ở vị trí xa nhau trong một từ dài hoặc câu.

Kiến trúc LSTM

Bộ nhớ dài-ngắn hạn (LSTM) là một loại mạng nơ-ron hồi tiếp (RNN) đặc biệt, được thiết kế để học và ghi nhớ hiệu quả các phụ thuộc dài hạn trong dữ liệu tuần tự. Các RNN truyền thống gặp khó khăn với vấn đề mất hoặc bùng nổ gradient, điều này cản trở khả năng của chúng trong việc nắm bắt các mối quan hệ kéo dài qua nhiều bước thời gian. LSTM khắc phục hạn chế này bằng cách giới thiệu một cấu trúc ô nhớ độc đáo, có khả năng chọn lọc lưu giữ hoặc quên thông tin khi cần thiết.

Điểm đột phá cốt lõi của LSTM nằm ở các cơ chế cổng: cổng quên, cổng đầu vào và cổng đầu ra. Các cổng này điều tiết luồng thông tin, cho phép mạng tập trung vào dữ liệu quan trọng trong khi loại bỏ những chi tiết không cần thiết. Kiến trúc này giúp LSTM đặc biệt hiệu quả trong các tác vụ đòi hỏi ngữ cảnh qua các chuỗi dài, chẳng hạn như mô hình hóa ngôn ngữ, dự đoán chuỗi thời gian và phân tích video.

Bằng cách giảm thiểu các vấn đề liên quan đến gradient, LSTM đảm bảo việc học ổn định ngay cả trong các mạng sâu. Khả năng này đã khiến LSTM trở thành nền tảng trong các ứng dụng như nhận dạng giọng nói, dịch máy và phát hiện bất thường, nơi việc hiểu các mối liên hệ theo thời gian là rất quan trọng.



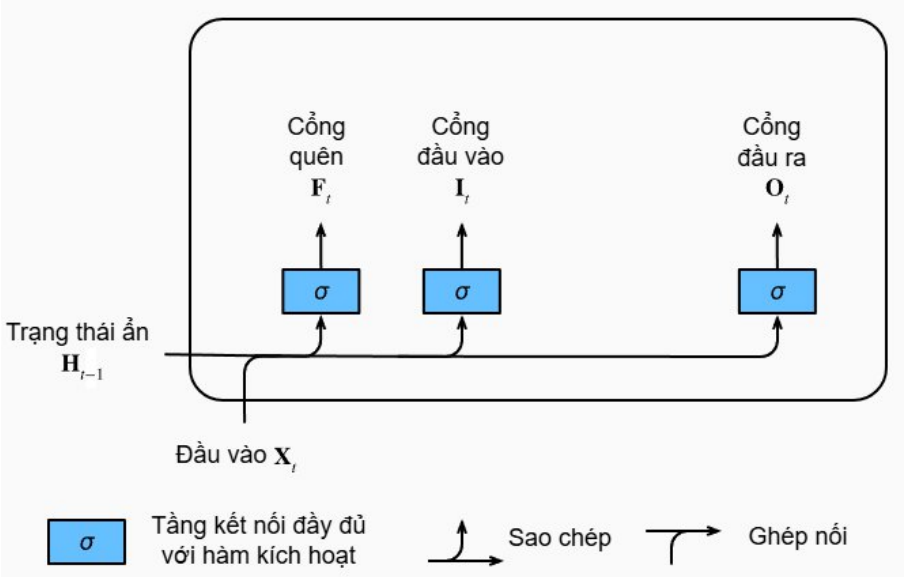
Hình 3: Kiến trúc LSTM

Có thể cho rằng thiết kế này được lấy cảm hứng từ các cổng logic trong máy tính. Để kiểm soát một ô nhớ chúng ta cần một số các cổng. Một cổng để đọc các thông tin từ ô nhớ đó (trái với việc đọc từ các ô khác). Chúng ta sẽ gọi cổng này là cổng đầu ra (output gate). Một cổng thứ hai để quyết định khi nào cần ghi dữ liệu vào ô nhớ. Chúng ta gọi cổng này là cổng đầu vào (input gate). Cuối cùng, chúng ta cần một cơ chế để thiết lập lại nội dung chứa trong ô nhớ, được chi phối bởi một cổng quên (forget gate). Động lực của thiết kế trên cũng tương tự như trước đây, đó là để đưa ra quyết định khi nào cần nhớ và khi nào nên bỏ qua đầu vào trong trạng thái tiềm ẩn thông qua một cơ chế chuyên dụng.

Ba cổng được giới thiệu trong LSTM đó là: cổng đầu vào, cổng quên và cổng đầu ra. Bên cạnh đó chúng ta sẽ giới thiệu một ô nhớ có kích thước giống với trạng thái ẩn. Nói đúng hơn đây chỉ là phiên bản đặc biệt của trạng thái ẩn, được thiết kế để ghi lại các thông tin bổ sung.

Cổng đầu vào, cổng quên và cổng đầu ra

Dữ liệu được đưa vào các cổng LSTM là đầu vào ở bước thời gian hiện tại X_t và trạng thái ẩn ở bước thời gian trước đó H_{t-1} Những đầu vào này được xử lý bởi một tầng kết nối đầy đủ và một hàm kích hoạt sigmoid để tính toán các giá trị của các cổng đầu vào, cổng quên và cổng đầu ra. Kết quả là, tất cả các giá trị đầu ra tại ba cổng đều nằm trong khoảng [0,1].



Hình 4: Các phép tính tại cổng đầu vào, cổng quên và cổng đầu ra trong một đơn vị LSTM.

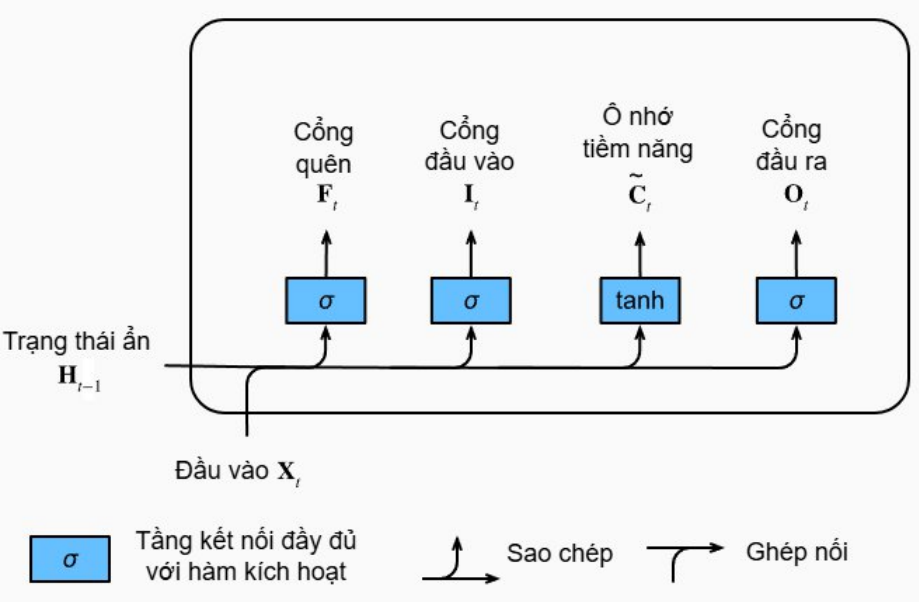
Chúng ta giả sử rằng có h nút ẩn, mỗi minibatch có kích thước *n* và kích thước đầu vào là *d*. Như vậy, đầu vào là $X_t \in R^{n \times d}$ và trạng thái ẩn của bước thời gian trước

đó là $H_{t-1} \in R^{n \times h}$.Tương tự, các cổng được định nghĩa như sau: cổng đầu vào là $I_t \in R^{n \times h}$, cổng quên là $F_t \in R^{n \times h}$, và cổng đầu ra là $O_t \in R^{n \times h}$.Chúng được tính như sau:

- Forget gate: $F_t = \sigma(W_{xf} * X_t + W_{hi} * H_{t-1} + b_F)$
- Input gate: $I_t = \sigma(W_{xi} * X_t + W_{hf} * H_{t-1} + b_I)$
- Output gate: $O_t = \sigma(W_{xo} * X_t + W_{ho} * H_{t-1} + b_O)$

trong đó $W_{xi}, W_{xf}, W_{xo} \in R^{d \times h}$ và $W_{hi}, W_{hf}, W_{ho} \in R^{h \times h}$ là các trọng số và $b_i, b_f, b_o \in R^{1 \times h}$ là các hệ số điều chỉnh.

Ô nhớ tiềm năng



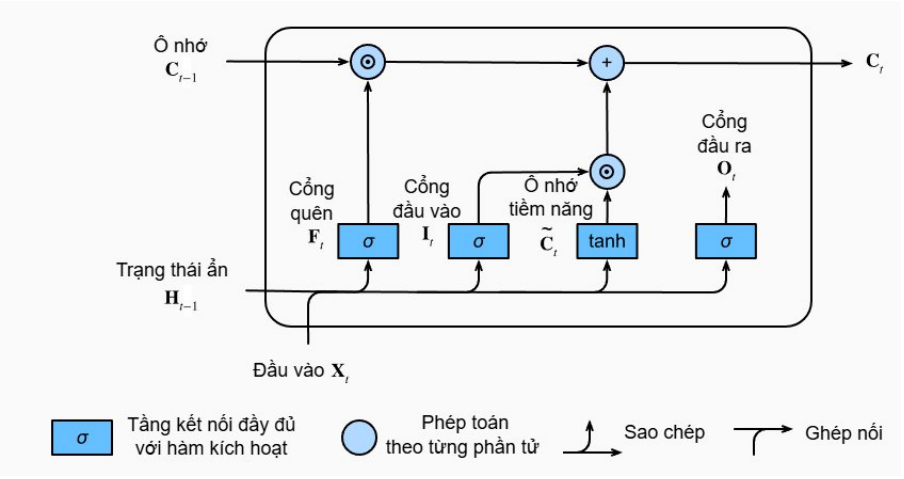
Hình 5: Các phép tính toán trong ô nhớ tiềm năng của LSTM

Tiếp theo, chúng ta sẽ thiết kế một ô nhớ. Vì ta vẫn chưa chỉ định tác động của các cổng khác nhau, nên đầu tiên ta sẽ giới thiệu ô nhớ tiềm năng $\tilde{C}_t \in R^{n \times h}$. Các phép tính toán cũng tương tự như ba cổng mô tả ở trên, ngoài trừ việc ở đây ta sử dụng hàm kích hoạt *tanh* với miền giá trị nằm trong khoảng $[-1,1]$. Điều này dẫn đến phương trình sau tại bước thời gian t.

$$\tilde{C}_t = \tanh\left(X_t W_{xc} + H_{t-1} W_{hc} + b_c\right)$$

Ở đây $W_{xc} \in R^{d \times h}$ và $W_{hc} \in R^{h \times h}$ là các tham số trọng số và $b_c \in R^{1 \times h}$ là một hệ số điều chỉnh.

Ô nhớ



Hình 6: Các phép tính toán trong ô nhớ của LSTM

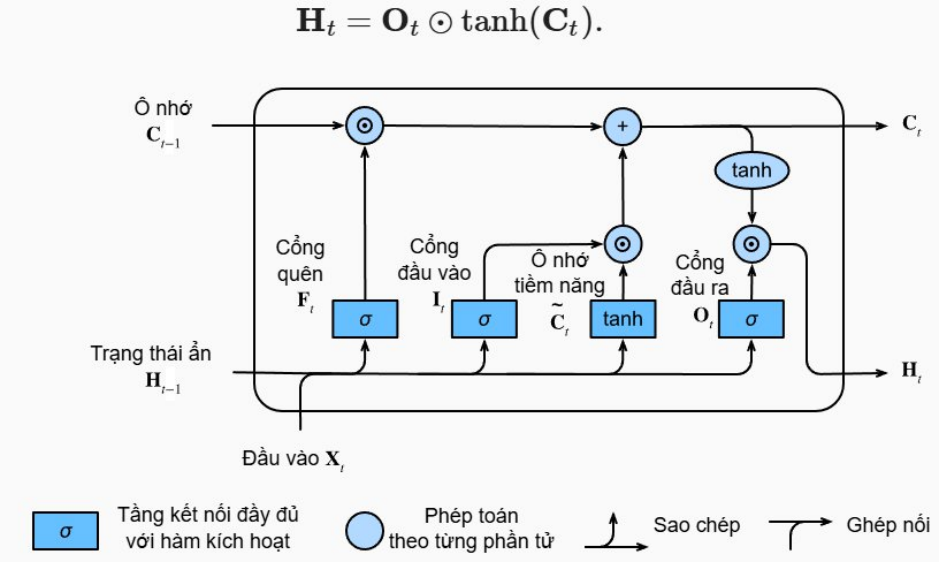
Trong LSTM, chúng ta có hai tham số, I_t điều chỉnh lượng dữ liệu mới được lấy vào thông qua $\tilde{C}_t$ và tham số quên F_t chỉ định lượng thông tin cũ cần giữ lại trong ô nhớ $C_{t-1} \in R^{n \times h}$. Sử dụng cùng một phép nhân theo từng điểm đi đến phương trình cập nhật như sau:

$$C_t = F_t \odot C_{t-1} + I_t \odot \tilde{C}_t$$

Nếu giá trị ở cổng quên luôn xấp xỉ bằng 1 và cổng đầu vào luôn xấp xỉ bằng 0, thì giá trị ô nhớ trong quá khứ C_{t-1} sẽ được lưu lại qua thời gian và truyền tới bước thời gian hiện tại. Thiết kế này được giới thiệu nhằm giảm bớt vấn đề tiêu biến gradient cũng như nắm bắt các phụ thuộc dài hạn trong chuỗi thời gian tốt hơn

Các Trạng thái Ẩn

$$\mathbf{H}_t = \mathbf{O}_t \odot \tanh(\mathbf{C}_t).$$



Hình 7: Các phép tính của trạng thái ẩn

Cuối cùng, chúng ta cần phải xác định cách tính trạng thái ẩn $H_t \in R^{n \times h}$. Đây là nơi cổng đầu ra được sử dụng. Trong LSTM, đây chỉ đơn giản là một phiên bản có

kiểm soát của hàm kích hoạt *tanh* trong ô nhớ. Điều này đảm bảo rằng các giá trị của H_t luôn nằm trong khoảng (-1, 1). Bất cứ khi nào giá trị của cổng đầu ra là 1, thực chất chúng ta đang đưa toàn bộ thông tin trong ô nhớ tới bộ dự đoán. Ngược lại, khi giá trị của cổng đầu ra là 0, chúng ta giữ lại tất cả các thông tin trong ô nhớ và không xử lý gì thêm.

2.2.3. LSTM trong Tesseract OCR

Vai trò của LSTM trong Tesseract

Trong Tesseract 4.0 trở lên, LSTM đóng vai trò trung tâm trong quá trình nhận dạng ký tự. Thay vì sử dụng các đặc trưng được định nghĩa thủ công, Tesseract sử dụng mạng LSTM hai chiều để "đọc" hình ảnh văn bản theo cả hai hướng (từ trái sang phải và từ phải sang trái), sau đó kết hợp kết quả để đưa ra dự đoán về ký tự.

Quy trình xử lý của Tesseract với LSTM bao gồm:

Trích xuất đặc trưng: Hình ảnh đầu vào được chuyển đổi thành chuỗi các vector đặc trưng, thường sử dụng các cửa sổ trượt dọc theo dòng văn bản. Quá trình này trích xuất các đặc điểm quan trọng từ hình ảnh, tạo nền tảng cho việc nhận diện văn bản chính xác.

Mạng LSTM hai chiều: Các vector đặc trưng được đưa vào mạng LSTM hai chiều, cho phép mô hình học các phụ thuộc ngữ cảnh theo cả hai hướng của chuỗi văn bản. Điều này giúp cải thiện khả năng nhận diện các ký tự và từ trong bối cảnh phức tạp.

Lớp Connectionist Temporal Classification - CTC: Đầu ra của mạng LSTM được xử lý bởi lớp CTC, nhằm ánh xạ chuỗi dự đoán sang chuỗi ký tự thực tế. Lớp CTC giải quyết vấn đề căn chỉnh giữa đầu vào và đầu ra, đảm bảo rằng văn bản được nhận diện đúng thứ tự và không bị lệch.

Hậu xử lý: Kết quả nhận diện được tinh chỉnh thông qua các kỹ thuật hậu xử lý, chẳng hạn như kiểm tra từ điển hoặc áp dụng quy tắc ngữ pháp. Giai đoạn này giúp sửa lỗi và nâng cao độ chính xác của văn bản đầu ra, đảm bảo phù hợp với ngữ cảnh.

Xử lý chuỗi ký tự liên tục

Một trong những thách thức lớn trong OCR là nhận dạng văn bản có ký tự liên tục hoặc chữ viết tay, khi ranh giới giữa các ký tự không rõ ràng. LSTM trong Tesseract giải quyết vấn đề này một cách hiệu quả nhờ các đặc điểm sau:

Phân tích theo chuỗi thời gian: LSTM xử lý hình ảnh như một chuỗi thời gian, trong đó mỗi cột của hình ảnh được xem như một bước thời gian. Thay vì nhận diện từng ký tự riêng lẻ, LSTM tập trung vào việc nhận dạng các mẫu trong chuỗi, giúp nó xử lý tốt các văn bản liên tục hoặc chữ viết tay, nơi các ký tự thường bị dính liền.

CTC Decoding: Tesseract sử dụng giải thuật Connectionist Temporal Classification để giải quyết vấn đề căn chỉnh giữa chuỗi đầu vào (các vector đặc trưng) và chuỗi đầu ra (các ký tự). CTC cho phép mạng học cách ánh xạ từ một chuỗi dài (nhiều frame hình ảnh) sang chuỗi ký tự ngắn hơn mà không cần thông tin rõ ràng về ranh giới ký tự, rất hữu ích khi các ký tự không được phân tách rõ ràng.

Xử lý ngữ cảnh: Nhờ sử dụng LSTM hai chiều, Tesseract có khả năng xem xét ngữ cảnh từ cả phía trước và phía sau vị trí hiện tại trong chuỗi văn bản. Điều này giúp mô hình hiểu và nhận diện các mẫu phức tạp trong chữ viết liên tục hoặc chữ viết tay, cải thiện độ chính xác trong các tình huống khó.

Mô hình ngôn ngữ: Tesseract tích hợp mô hình ngôn ngữ kết hợp với LSTM để nâng cao độ chính xác. Mô hình này cân nhắc xác suất của các chuỗi từ trong ngôn ngữ đích, từ đó sửa lỗi và ưu tiên các chuỗi ký tự có ý nghĩa ngữ pháp, đặc biệt hữu ích khi xử lý văn bản có cấu trúc phức tạp hoặc không rõ ràng.

Nhờ các đặc điểm vượt trội của mạng LSTM, Tesseract có thể xử lý hiệu quả nhiều tình huống phức tạp trong nhận dạng văn bản. Với chữ viết tay nối liền, LSTM phân tích văn bản như một chuỗi thời gian, nhận diện mẫu thay vì ký tự riêng lẻ, giúp vượt qua ranh giới không rõ ràng giữa các ký tự. Đối với font chữ trang trí có ký tự chồng chéo, khả năng xử lý ngữ cảnh hai chiều của LSTM cho phép hiểu các mẫu phức tạp, đảm bảo nhận diện chính xác.

Trong trường hợp văn bản có khoảng cách không đều, giải thuật CTC hỗ trợ ánh xạ chuỗi đầu vào dài sang đầu ra ngắn hơn mà không cần phân tách rõ ràng. Ngoài ra, với văn bản bị méo hoặc biến dạng, sự kết hợp giữa mô hình ngôn ngữ và khả năng học ngữ cảnh của LSTM giúp sửa lỗi và duy trì độ chính xác, mang lại kết quả đáng tin cậy trong các điều kiện khó khăn.

2.3. Cơ chế huấn luyện trong Tesseract

Quá trình huấn luyện Tesseract là một quy trình gồm nhiều bước nhằm tạo ra một mô hình nhận diện ký tự quang học hiệu quả, đặc biệt với các ngôn ngữ hoặc phông chữ cụ thể. Bước đầu tiên là chuẩn bị dữ liệu huấn luyện, trong đó cần thu thập hoặc tạo ra các hình ảnh văn bản định dạng *.tif*, kèm theo các *file ground truth* (*.gt.txt*) chứa nội dung văn bản chính xác tương ứng. Dữ liệu này đóng vai trò nền tảng để mô hình học cách nhận diện ký tự một cách chính xác.

Tiếp theo, tiền xử lý dữ liệu được thực hiện để chuyển đổi dữ liệu thô thành các định dạng phù hợp cho quá trình huấn luyện. Các file hình ảnh và ground truth sẽ được xử lý để tạo ra các *file .box* (chứa thông tin vị trí ký tự) và *.lstmf* (định dạng đặc biệt cho mô hình LSTM). Bước này đảm bảo rằng dữ liệu đầu vào được chuẩn hóa và sẵn sàng cho các công cụ huấn luyện của Tesseract.

Chuẩn bị file cấu hình là bước quan trọng tiếp theo. Các file như font_properties (mô tả đặc điểm phông chữ), wordlist (danh sách từ vựng), và các file trong thư mục langdata (chứa thông tin ngôn ngữ) cần được tạo hoặc chỉnh sửa để phù hợp với ngôn ngữ hoặc yêu cầu cụ thể. Những file này giúp định hướng mô hình trong việc học các đặc trưng của văn bản mục tiêu.

Quá trình huấn luyện mô hình sử dụng các công cụ như text2image, tesseract, và lstmtraining để xây dựng mô hình dựa trên mạng nơ-ron LSTM. Công cụ text2image tạo dữ liệu huấn luyện từ văn bản và phông chữ, trong khi lstmtraining thực hiện việc huấn luyện mô hình LSTM để nhận diện ký tự và từ ngữ. Đây là bước cốt lõi, nơi mô hình học cách liên kết hình ảnh văn bản với nội dung chính xác.

Sau khi huấn luyện, đánh giá và tinh chỉnh được tiến hành để kiểm tra hiệu suất của mô hình. Các chỉ số như độ chính xác nhận diện ký tự (CER) hoặc từ (WER) được sử dụng để đánh giá. Nếu hiệu suất chưa đạt yêu cầu, mô hình có thể được tinh chỉnh bằng cách điều chỉnh tham số hoặc bổ sung dữ liệu huấn luyện.

Cuối cùng, đóng gói và triển khai là bước hoàn thiện, trong đó mô hình được tổng hợp thành file traineddata. File này có thể được tích hợp vào các ứng dụng OCR để sử dụng thực tế, cho phép nhận diện văn bản trong các hình ảnh với độ chính xác cao, phù hợp với ngôn ngữ hoặc phông chữ đã được huấn luyện.

2.3.1. Tổng quan về quá trình huấn luyện

Cấu trúc dữ liệu huấn luyện

Dữ liệu huấn luyện cơ bản cho Tesseract bao gồm hai thành phần chính:

File hình ảnh (.tif): Tesseract ưu tiên sử dụng định dạng TIFF (.tif) cho các file hình ảnh huấn luyện bởi định dạng này hỗ trợ lưu trữ hình ảnh không nén và có khả năng chứa nhiều trang, đảm bảo chất lượng dữ liệu đầu vào. Thông thường, các hình ảnh này là các dòng văn bản đơn lẻ được trình bày trên nền trắng, giúp mô hình dễ dàng nhận diện ký tự. Độ phân giải khuyến nghị cho hình ảnh là 300 DPI để đảm bảo độ rõ nét, tạo điều kiện thuận lợi cho việc nhận diện chính xác. Hình ảnh có thể ở dạng nhị phân (đen trắng) hoặc xám (grayscale), tùy thuộc vào yêu cầu cụ thể của quá trình huấn luyện, với mỗi loại mang lại lợi ích riêng trong việc xử lý và phân tích văn bản.

File ground truth (.gt.txt): File ground truth (*.gt.txt*) là file văn bản thuần túy (plain text) chứa nội dung chính xác của văn bản xuất hiện trong hình ảnh .tif tương ứng. Tên file ground truth thường được đặt theo quy tắc *[tên_file_hình_ảnh].gt.txt* để dễ dàng liên kết với hình ảnh. Nội dung trong file phải khớp hoàn toàn với văn bản trong hình ảnh, bao gồm cả khoảng trắng và các ký tự đặc biệt, nhằm đảm bảo tính chính xác khi huấn luyện mô hình Tesseract. Việc này giúp mô hình học cách nhận diện đúng các ký tự và cấu trúc văn bản dựa trên dữ liệu tham chiếu chính xác.

Ví dụ, nếu có file hình ảnh *"line_01.tif"* chứa văn bản *"Xin chào thế giới",* thì file ground truth tương ứng sẽ là *"line_01.gt.txt"* với nội dung *"Xin chào thế giới"*.

Các file .tif và .gt.txt cần được chuyển đổi thành các định dạng đặc biệt để huấn luyện LSTM trong Tesseract:

- File .box**:** Chứa thông tin về vị trí của từng ký tự trong hình ảnh. Mỗi dòng đại diện cho một ký tự với định dạng: *ký_tự x_left y_bottom x_right y_top page_number* . Được tạo ra bằng cách chạy lệnh *tesseract [image_file] [output_base] batch.nochop makebox*
- File .lstmf (LSTM Feature)**:** Đây là định dạng nhị phân chứa đặc trưng đã được trích xuất từ hình ảnh, cùng với thông tin ground truth. Mỗi file *.lstmf* tương ứng với một hình ảnh và ground truth của nó. Là định dạng đầu vào trực tiếp cho quá trình huấn luyện LSTM

Cấu trúc của file .lstmf

File .lstmf là định dạng nhị phân được sử dụng trong Tesseract để lưu trữ dữ liệu huấn luyện, đặc biệt khi huấn luyện các mô hình OCR như LSTM. Mặc dù không thể đọc trực tiếp, việc hiểu cấu trúc của file .lstmf giúp làm rõ cách Tesseract xử lý dữ liệu trong quá trình huấn luyện.

Header: Phần đầu của file .lstmf chứa các thông tin meta quan trọng, bao gồm kích thước hình ảnh, số lượng đặc trưng được trích xuất, và các tham số khác liên quan đến quá trình huấn luyện. Header cung cấp bối cảnh cần thiết để mô hình hiểu cách xử lý dữ liệu trong các phần tiếp theo.

Feature Vectors: Phần này lưu trữ chuỗi các vector đặc trưng được trích xuất từ hình ảnh. Các vector này thường là kết quả của việc quét hình ảnh theo từng cột, biểu diễn các đặc điểm hình ảnh dưới dạng số. Đây là dữ liệu đầu vào chính cho mạng LSTM, giúp mô hình học các mẫu văn bản.

Ground Truth Text: File .lstmf bao gồm văn bản chính xác (ground truth) tương ứng với hình ảnh. Văn bản này đóng vai trò như nhãn, cho phép mô hình so sánh đầu

ra dự đoán với kết quả đúng trong quá trình huấn luyện, từ đó điều chỉnh để cải thiện độ chính xác.

Bounding Box Information: Thông tin về hộp giới hạn (bounding box) xác định vị trí của các ký tự trong hình ảnh. Dữ liệu này hỗ trợ mô hình hiểu cấu trúc không gian của văn bản, đặc biệt hữu ích khi xử lý các ký tự hoặc từ có vị trí phức tạp, như trong văn bản bị méo hoặc chữ viết tay.

Tệp .lstmf được tạo ra bằng *lệnh tesseract [image_file] [output_base] lstm.train* khi đã có tệp *.gt.txt* tương ứng.

2.3.2. File cấu hình và dữ liệu phụ trợ

Thư mục langdata chứa các file cấu hình và dữ liệu phụ trợ cho từng ngôn ngữ:

+ *[lang].training_text*: Chứa văn bản mẫu để tạo dữ liệu huấn luyện tổng hợp bằng text2image. Nên bao gồm đa dạng các ký tự, từ, và cụm từ phổ biến trong ngôn ngữ đó
+ *[lang].punc*: Liệt kê các ký tự dấu câu được sử dụng trong ngôn ngữ đó
+ *[lang].numbers*: Liệt kê các ký tự số được sử dụng
+ *[lang].wordlist*: Danh sách các từ phổ biến trong ngôn ngữ đó, dùng để cải thiện khả năng nhận dạng
+ *[lang].unicharset*: Tập hợp tất cả các ký tự Unicode được sử dụng trong ngôn ngữ đó

File font_properties

File font_properties chứa thông tin về các font chữ được sử dụng trong dữ liệu huấn luyện: *[font_name] [italic] [bold] [fixed] [serif] [fraktur]*

Trong đó:

+ *[font_name]*: Tên của font chữ
+ *[italic], [bold], [fixed], [serif], [fraktur]*: Các giá trị 0 hoặc 1 cho biết font có thuộc tính đó hay không

File này rất quan trọng vì nó giúp Tesseract hiểu được các đặc điểm của font chữ, từ đó cải thiện khả năng nhận dạng.

File wordlist

File wordlist là một tệp chứa danh sách các từ hợp lệ trong một ngôn ngữ cụ thể, được sử dụng để hỗ trợ quá trình nhận dạng văn bản trong các hệ thống OCR như Tesseract. Mỗi dòng trong file wordlist chứa một từ, giúp hệ thống kiểm tra và cải thiện độ chính xác của kết quả nhận dạng thông qua việc đối chiếu với từ điển. Để đạt hiệu quả cao, file wordlist nên bao gồm các từ phổ biến trong ngôn ngữ đó cùng với các biến thể của chúng, chẳng hạn như dạng số nhiều, chia động từ hoặc các dạng từ khác, đảm bảo mô hình có thể nhận diện và sửa lỗi hiệu quả trong các ngữ cảnh đa dạng.

Các file cấu hình khác

Ngoài các file đã đề cập, còn có một số file cấu hình khác có thể cần thiết:

+ *[lang].config*: Chứa các thông số cấu hình cho quá trình huấn luyện. Có thể định nghĩa các tham số như learning rate, batch size, v.v.
+ *[lang].lstm-unicharset*: Phiên bản đặc biệt của unicharset cho LSTM. Được tạo ra trong quá trình huấn luyện
+ *[lang].traineddata*: File cuối cùng sau khi huấn luyện. Bao gồm mô hình LSTM và các thông tin cấu hình khác

2.3.3. Các lệnh và công cụ huấn luyện

Công cụ text2image được sử dụng để tạo dữ liệu huấn luyện tổng hợp từ text: *text2image --text=[text_file] --outputbase=[output_base] --font=[font_name] --fonts_dir=[fonts_directory] --fontconfig_tmpdir=[tmp_dir]*

Các tham số chính:

Tham số	Giải thích
--text	File văn bản đầu vào
--outputbase	Tiền tố cho các file đầu ra
--font	Tên font chữ sử dụng
--fonts_dir	Thư mục chứa font chữ
--fontconfig_tmpdir	Thư mục tạm thời cho fontconfig

Bảng 2: Tham số và giải thích cho công cụ text2image

Ví dụ: *text2image --text=vie.training_text --outputbase=vie.Arial --font="Arial" --fonts_dir=/usr/share/fonts --fontconfig_tmpdir=/tmp*

Lệnh này sẽ tạo ra các file .tif và .box tương ứng với văn bản trong file vie.training_text, sử dụng font Arial.

Sinh file .lstmf

Công cụ tesseract được sử dụng để tạo các file *.lstmf* từ các file *.tif* và *.gt.txt*: *tesseract [image_file] [output_base] lstm.train [config_file]*

Các tham số:

Tham số	Giải thích
[image_file]	File hình ảnh đầu vào (.tif)
[output_base]	Tiền tố cho file đầu ra
lstm.train	Chỉ định chế độ tạo file huấn luyện LSTM
[config_file]	File cấu hình tùy chọn

Bảng 3: Tham số và giải thích cho sinh tệp .lstmf

Ví dụ: *tesseract vie.Arial.exp0.tif vie.Arial.exp0 lstm.train*

Công cụ lstmtraining được sử dụng để huấn luyện mô hình LSTM:

lstmtraining --traineddata=[lang].traineddata --train_listfile=[list_file] --eval_listfile = [eval_list] --continue_from=[checkpoint] --model_output=[output_model] --max_iterations=[max_iter]

Các tham số chính:

Tham số	Giải thích
--traineddata	File traineddata làm cơ sở
--train_listfile	File chứa danh sách các file .lstmf dùng để huấn luyện
--eval_listfile	File chứa danh sách các file .lstmf dùng để đánh giá
--continue_from	Checkpoint để tiếp tục huấn luyện (nếu có)
--model_output	Tiền tố cho file mô hình đầu ra
--max_iterations	Số vòng lặp tối đa

Bảng 4: Tham số và giải thích cho công cụ lstmtraining

Ví dụ:

lstmtraining --traineddata=vie.traineddata --train_listfile=train_files.txt --eval_listfile=eval_files.txt --continue_from="" --model_output=vie/vie --max_iterations=10000

Quy trình huấn luyện đầy đủ

Dưới đây là quy trình đầy đủ để huấn luyện một mô hình từ đầu:

Quy trình	Đoạn mã
1. Chuẩn bị dữ liệu huấn luyện	*mkdir -p data/train data/eval* *# Tạo hoặc thu thập các file .tif và .gt.txt* *Tạo file .lstmf: for %f in (data\train*.tif) do (* *for %~nf in ("%f") do (* *tesseract "%f" "data\train\%~nf" lstm.train* *)* *)* *for %f in (data\eval*.tif) do (* *for %~nf in ("%f") do (* *tesseract "%f" "data\eval\%~nf" lstm.train* *)* *)*
2. Tạo file danh sách:	*(for %f in (data\train*.lstmf) do @echo %f) > train_files.txt* *(for %f in (data\eval*.lstmf) do @echo %f) > eval_files.txt*
3. Tạo các file cấu hình	*# Tạo file font_properties* *echo Arial 0 0 0 0 0 > font_properties*

	# Sao chép các file từ langdata *copy langdata\vie* .*
4. Bắt đầu huấn luyện	*# Khởi tạo từ một mô hình cơ sở (fine-tuning)* *combine_tessdata -e vie.traineddata vie.lstm* *# Huấn luyện mô hình* *lstmtraining --traineddata=vie.traineddata --train_listfile=train_files.txt --eval_listfile=eval_files.txt --continue_from=vie.lstm --model_output=vie/vie --learning_rate=0.001 --max_iterations=10000*
5. Đánh giá và tinh chỉnh	*# Đánh giá mô hình sau mỗi 1000 vòng lặp* *lstmeval --model vie\vie_checkpoint --traineddata vie.traineddata --eval_listfile eval_files.txt*
6. Đóng gói mô hình	*# Tạo file traineddata mới* *combine_lang_model --input_unicharset vie.unicharset --script_dir langdata* *--output_dir --lang vie --pass_through_recoder* *--lstm_trained_data vie_checkpoint*

Bảng 5: Quy trình huấn luyện

CHƯƠNG 3: THỰC NGHIỆM, ĐÁNH GIÁ VÀ KẾT QUẢ MÔ HÌNH

3.1.Chuẩn bị dữ liệu huấn luyện

TextRecognitionDataGenerator (TRDG) là một công cụ mã nguồn mở được phát triển bởi Belval, có sẵn trên GitHub, được thiết kế để tạo dữ liệu hình ảnh văn bản tổng hợp phục vụ cho việc huấn luyện các hệ thống nhận dạng ký tự quang học (OCR). Công cụ này cho phép người dùng tùy chỉnh việc tạo ra các hình ảnh văn bản với nhãn chính xác, đáp ứng nhu cầu huấn luyện và tinh chỉnh các mô hình OCR. Đặc biệt, TRDG tỏ ra hữu ích trong các trường hợp dữ liệu thực tế khan hiếm hoặc thiếu sự đa dạng, giúp cung cấp các tập dữ liệu phong phú để cải thiện hiệu suất của mô hình.

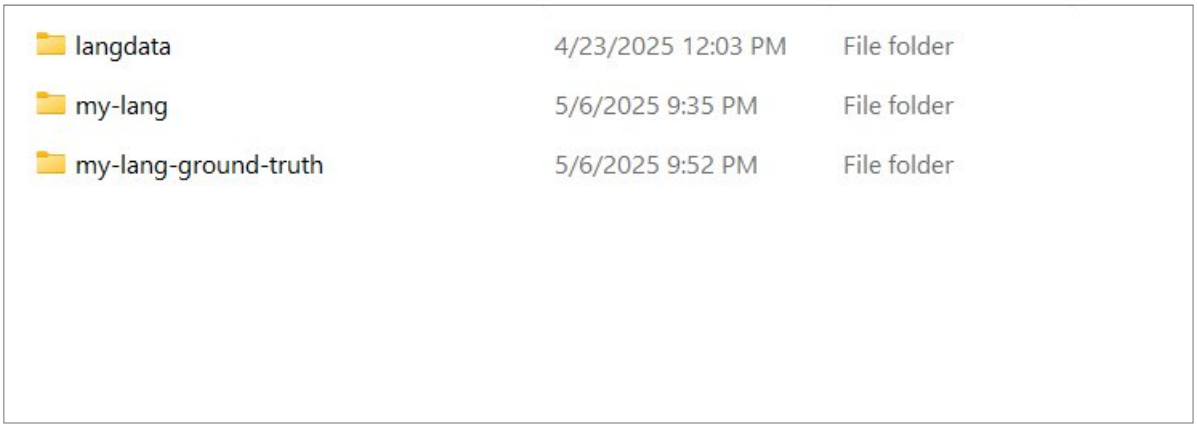
Mục tiêu của việc sử dụng TRDG trong trường hợp này là tạo ra 50.000 hình ảnh văn bản tiếng Việt, sử dụng font Times New Roman. Để đạt được mục tiêu này, TRDG sẽ được cấu hình với các tham số phù hợp, bao gồm việc chỉ định ngôn ngữ tiếng Việt, chọn font Times New Roman, và thiết lập số lượng ảnh cần sinh. Công cụ cho phép tùy chỉnh thêm các yếu tố như kích thước ảnh, độ nghiêng, màu chữ, và nền ảnh, đảm bảo dữ liệu đầu ra đa dạng và phù hợp với yêu cầu huấn luyện OCR.

Quá trình sử dụng TRDG bắt đầu bằng việc cài đặt công cụ thông qua lệnh pip install trdg hoặc chạy trong môi trường Docker. Sau đó, người dùng cần chuẩn bị một tệp từ điển tiếng Việt (dictionary file) chứa các từ hoặc câu sẽ được sử dụng để sinh văn bản. Font Times New Roman phải được thêm vào thư mục font của TRDG để đảm bảo công cụ sử dụng đúng kiểu chữ. Lệnh chạy TRDG sẽ bao gồm các tham số như -l vi để chỉ định ngôn ngữ tiếng Việt, -c 50000 để tạo 50.000 ảnh, và -ft để chỉ định đường dẫn đến tệp font Times New Roman.

Sau khi sinh ảnh, các hình ảnh văn bản tổng hợp sẽ được lưu trong thư mục đầu ra, kèm theo nhãn tương ứng. Các ảnh này có thể được sử dụng trực tiếp để huấn luyện các mô hình OCR như Tesseract hoặc các mô hình học sâu khác. Với khả năng tùy chỉnh cao và hỗ trợ ngôn ngữ không phải Latin như tiếng Việt, TRDG là một công cụ mạnh mẽ để tạo dữ liệu huấn luyện chất lượng, góp phần nâng cao độ chính xác của hệ thống nhận dạng văn bản

3.2.Cấu hình huấn luyện

Cấu trúc thư mục



Hình 8: Cấu trúc thư mục

Chạy câu lệnh trên cmd

make training MODEL_NAME=my-lang TESSDATA=../tessdata/ MAX_ITERATIONS=5000 LEARNING_RATE=0.002

Ý nghĩa từng tham số:

+ MAX_ITERATIONS: số vòng lặp
+ LEARNING_RATE: tốc độ học
+ MODEL_NAME: tên mô hình xuất ra

Theo dõi loss qua từng checkpoint

```
Total weights = 392755
Built network:[1,36,0,1[C3,3Ft16]Mp3,3TxyLfys48Lfx96RxLrx96Lfx192Fc147] from request [1,36,0,1 Ct3,3,16 Mp3,3 Lfys48 Lfx96 Lrx96 Lfx
192 O1c148]
Training parameters:
  Debug interval = 0, weights = 0.1, learning rate = 0.002, momentum=0.5
null char=146
At iteration 3600/3600/3600, mean rms=5.922%, delta=53.642%, BCER train=95.801%, BWER train=98.350%, skip ratio=0.000%, New worst BC
ER = 95.801 wrote checkpoint.
At iteration 3700/3700/3700, mean rms=5.891%, delta=54.402%, BCER train=95.935%, BWER train=98.467%, skip ratio=0.000%, New worst BC
ER = 95.935 wrote checkpoint.
At iteration 3800/3800/3800, mean rms=5.862%, delta=55.151%, BCER train=96.029%, BWER train=98.600%, skip ratio=0.000%, New worst BC
ER = 96.029 wrote checkpoint.
At iteration 3900/3900/3900, mean rms=5.830%, delta=55.805%, BCER train=96.136%, BWER train=98.637%, skip ratio=0.000%, New worst BC
ER = 96.136 wrote checkpoint.
At iteration 4000/4000/4000, mean rms=5.796%, delta=56.394%, BCER train=96.291%, BWER train=98.713%, skip ratio=0.000%, New worst BC
ER = 96.291 wrote checkpoint.
At iteration 4100/4100/4100, mean rms=5.767%, delta=57.077%, BCER train=96.521%, BWER train=98.867%, skip ratio=0.000%, New worst BC
ER = 96.521 wrote checkpoint.
At iteration 4200/4200/4200, mean rms=5.734%, delta=57.798%, BCER train=96.718%, BWER train=99.003%, skip ratio=0.000%, New worst BC
ER = 96.718 wrote checkpoint.
At iteration 4300/4300/4300, mean rms=5.699%, delta=58.453%, BCER train=96.946%, BWER train=99.140%, skip ratio=0.000%, New worst BC
ER = 96.946 wrote checkpoint.
At iteration 4400/4400/4400, mean rms=5.669%, delta=59.264%, BCER train=97.174%, BWER train=99.277%, skip ratio=0.000%, New worst BC
ER = 97.174 wrote checkpoint.
Warning: LSTMTrainer deserialized an LSTMRecognizer!
At iteration 4500/4500/4500, mean rms=5.642%, delta=60.083%, BCER train=97.445%, BWER train=99.490%, skip ratio=0.000%, New worst BC
ER = 97.445 wrote checkpoint.
At iteration 4600/4600/4600, mean rms=5.620%, delta=59.654%, BCER train=97.209%, BWER train=99.417%, skip ratio=0.000%, wrote checkp
oint.
At iteration 4700/4700/4700, mean rms=5.616%, delta=59.567%, BCER train=97.148%, BWER train=99.413%, skip ratio=0.000%, wrote checkp
oint.
At iteration 4800/4800/4800, mean rms=5.611%, delta=59.431%, BCER train=97.134%, BWER train=99.410%, skip ratio=0.000%, wrote checkp
oint.
At iteration 4900/4900/4900, mean rms=5.605%, delta=59.304%, BCER train=97.118%, BWER train=99.447%, skip ratio=0.000%, wrote checkp
oint.
At iteration 5000/5000/5000, mean rms=5.602%, delta=59.314%, BCER train=97.116%, BWER train=99.463%, skip ratio=0.000%, wrote checkp
oint.
Finished! Selected model with minimal training error rate (BCER) = 95.575
```

Hình 9: Vòng lặp huấn luyện

+ Total weights = 392755: Số lượng trọng số trong mạng LSTM.
+ Learning rate = 0.002: Tốc độ học (learning rate) khá nhỏ – giúp mô hình học ổn định.
+ Momentum = 0.5: Tăng cường cập nhật trọng số dựa vào lịch sử gradient.

Xét mỗi vòng lặp huấn luyện

At iteration 3700/3700/3700, mean rms=5.891%, delta=54.028%, BCER train=95.935%, BWER train=98.467%, skip ratio=0.000%, New worst BC

Thành phần	Ý nghĩa
Iteration 3700/3700/3700	Vòng lặp hiện tại / số vòng sau checkpoint / tổng số vòng lặp tối đa.
mean rms=5.891%	Root Mean Square Error trung bình – sai số trung bình trong dự đoán ký tự. Càng thấp càng tốt.
delta=54.028%	Mức thay đổi rms so với checkpoint trước. Giúp theo dõi sự tiến triển
BCER train=95.935%	Character Error Rate (CER) – phần trăm ký tự nhận đúng.
BWER train=98.467%	Word Error Rate (WER) – phần trăm từ nhận đúng.
skip ratio=0.000%	Tỷ lệ mẫu bị bỏ qua trong quá trình training. Thường do lỗi input.
New worst BC	Đánh dấu đây là mô hình tệ nhất mới (nếu BCER tăng lên). Nếu không có, thì mô hình đang cải thiện.
wrote checkpoint	Khi gặp dòng này, mô hình hiện tại được lưu lại thành checkpoint. Tesseract ghi lại trạng thái model hiện tại vì nó đạt thành tích tốt hơn về BCER.

Bảng 6: Thành phần và ý nghĩa mỗi vòng lặp huấn luyện

Sau khi chạy xong huấn luyện

Finished! Selected model with minimal training error rate (BCER) = 95.575

BCER = 95.575 → Đây là Best Character Error Rate, chính xác hơn là: 95.575% là tỷ lệ ký tự được nhận diện đúng trong tập huấn luyện. Tương đương với Character Accuracy = 95.575%, hay ngược lại CER ≈ 4.425%.

Kết luận từ kết quả training trên ta có: Character Accuracy (độ chính xác ký tự): 95.575% là khá tốt, chứng tỏ mô hình đã học được đáng kể từ dữ liệu huấn luyện. CER (Character Error Rate): ~4.425% là chấp nhận được

3.3.Sử dụng mô hình và đánh giá kết quả

Với mẫu trong tập huấn luyện

Ảnh:

Hình 10: Ảnh trong tập huấn luyện

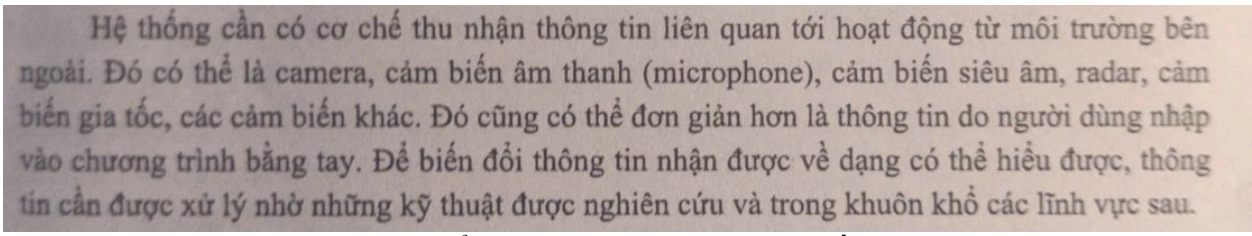
Kết quả:

```
D:\BTL-AI\pythonProject\.venv\Scripts\python.exe D:\BTL-AI\pythonProject\.venv\app.py
['5', '1', '1', '1', '1', '1', '6', '15', '287', '44', '82.577820', 'Aforhynchites']
['5', '1', '1', '1', '1', '2', '308', '6', '63', '53', '91.954880', 'cấp']
['5', '1', '1', '1', '1', '3', '385', '15', '177', '34', '93.128883', 'Elizaveta']
['5', '1', '1', '1', '1', '4', '575', '15', '181', '34', '89.635269', 'destrictus']
['5', '1', '1', '1', '1', '5', '770', '16', '84', '43', '93.021881', 'tùng']
```

Hình 11: Kết quả với ảnh trong tập huấn luyện

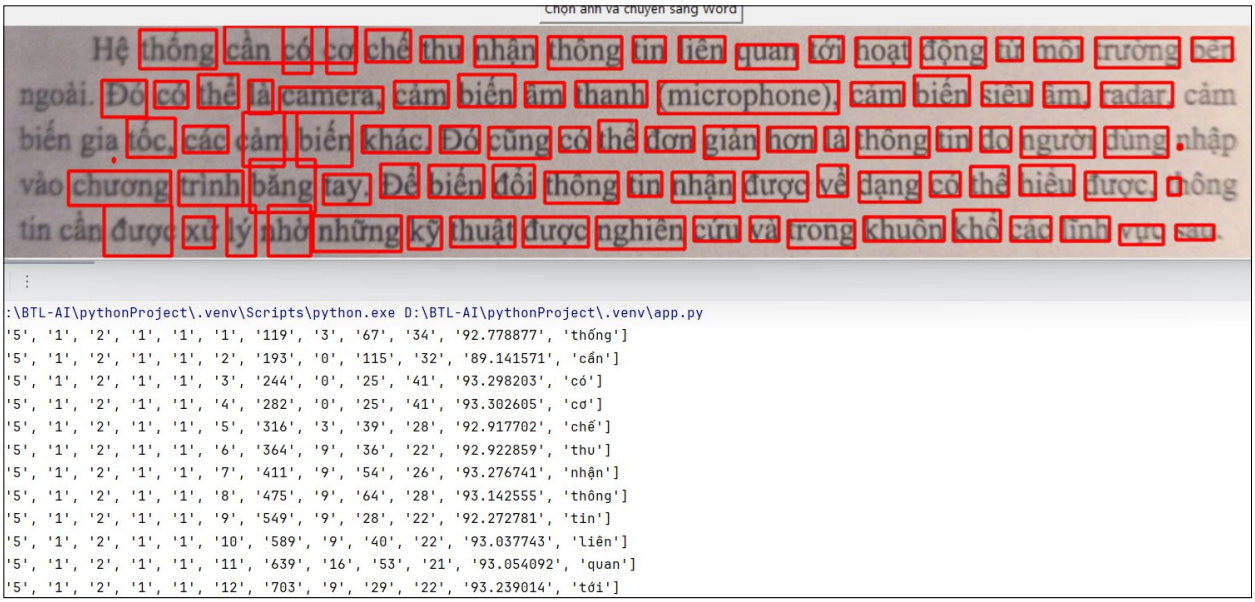
Nhận xét: Mô hình nhận diện tốt

Với ảnh chưa có trong tập huấn luyện



Hình 12: Ảnh chưa có trong tập huấn luyện

Kết quả:



Hình 13: Kết quả cho ảnh chưa có trong tập huấn luyện

Nhận xét:

Khả năng tổng thể tốt dù ảnh không nằm trong tập huấn luyện, hệ thống vẫn nhận diện được hầu hết các từ trong văn bản. Các từ được phát hiện và đóng khung khá chính xác. Hệ thống đã chia tách các từ riêng biệt thay vì nhận dạng từng ký tự, điều này giúp việc trích xuất ngữ nghĩa chính xác hơn. Các khung đỏ trong *Hình 11* cho thấy hệ thống xác định ranh giới từ hợp lý.

Nhận dạng từ tiếng Việt với dấu là một thử thách, tuy nhiên hệ thống đã thực hiện tốt nhiệm vụ này (ví dụ: các từ như "thông", "thu", "microphone", "chương", "biến", v.v. được nhận diện chính xác).

Dữ liệu đầu ra rõ ràng với Bảng kết quả phía dưới ảnh thể hiện rõ các thông tin gồm: vị trí từ (tọa độ khung), chỉ số độ tin cậy (confidence score), và từ được nhận diện. Điều này giúp dễ dàng đánh giá hiệu quả và mức độ chính xác của mô hình. Khả năng khái quát hóa tốt mặc dù ảnh chưa từng xuất hiện trong tập huấn luyện, mô hình vẫn xử lý hiệu quả, chứng tỏ khả năng khái quát hóa tốt của hệ thống OCR.

KẾT LUẬN

Quá trình thực nghiệm và đánh giá mô hình nhận dạng ký tự quang học tiếng Việt sử dụng công cụ TextRecognitionDataGenerator và Tesseract đã đạt được những kết quả khả quan. Dữ liệu huấn luyện được tạo ra với 50.000 ảnh văn bản tổng hợp, sử dụng font Times New Roman, đảm bảo tính đa dạng và phù hợp cho việc huấn luyện mô hình. Quá trình huấn luyện với các tham số cấu hình hợp lý (max_iterations=5000, learning_rate=0.002) đã giúp mô hình đạt được độ chính xác ký tự là 95.575%, tương ứng với tỷ lệ lỗi ký tự khoảng 4.425%. Đây là một kết quả tốt, cho thấy mô hình có khả năng học và nhận diện chính xác các ký tự tiếng Việt từ dữ liệu tổng hợp.

Đánh giá trên tập huấn luyện cho thấy mô hình nhận diện tốt các mẫu đã được huấn luyện. Với các ảnh chưa có trong tập huấn luyện, mô hình vẫn thể hiện khả năng nhận diện tương đối tốt, chứng minh tính tổng quát hóa của mô hình. Tuy nhiên, để cải thiện hiệu suất, đặc biệt với các trường hợp dữ liệu thực tế phức tạp hơn, cần xem xét bổ sung dữ liệu huấn luyện đa dạng hơn, tối ưu hóa thêm tham số hoặc thử nghiệm các kiến trúc mô hình nâng cao.

Tóm lại, mô hình OCR tiếng Việt được xây dựng và huấn luyện trong nghiên cứu này đã đạt được hiệu quả cao, đáp ứng tốt yêu cầu nhận dạng văn bản tiếng Việt trong các điều kiện kiểm tra. Kết quả này mở ra tiềm năng ứng dụng thực tế và là nền tảng cho các nghiên cứu cải tiến tiếp theo.

TÀI LIỆU THAM KHẢO

[1] Tesseract documentation: https://tesseract-ocr.github.io/

[2] Cài đặt Tesseract: https://github.com/tesseract-ocr/tesseract

[3] Mô hình LSTM: https://medium.com/@saurabh.rathor092/simple-rnn-vs-gru-vs-lstm-difference-lies-in-more-flexible-control-5f33e07b1e57