**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.

1. **Lý Do Chọn Đề Tài**

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.

1. **Mục Tiêu Nghiên Cứu**

Đề tài hướng đến việc đạt được các mục tiêu cụ thể sau:

* **Xây dựng mô hình OCR tùy chỉnh**: Phát triển một mô hình OCR dựa trên Tesseract, được tối ưu hóa cho nhận diện văn bản tiếng Việt từ ảnh văn bản in.
* **Huấn luyện mô hình**: Sử dụng tập dữ liệu gồm 5000 mẫu ảnh văn bản tiếng Việt để huấn luyện mô hình, đảm bảo khả năng nhận diện chính xác các ký tự và từ trong ngữ cảnh thực tế.
* **Đánh giá độ chính xác**: Đo lường hiệu suất của mô hình sau huấn luyện thông qua các chỉ số chuẩn như Tỷ lệ Sai sót Ký tự (Character Error Rate - CER) và Tỷ lệ Sai sót Từ (Word Error Rate - WER), từ đó xác định mức độ cải thiện so với mô hình Tesseract tiêu chuẩn.

1. **Phạm Vi Và Phương Pháp**

**Phạm Vi**

Nghiên cứu tập trung vào nhận diện văn bản tiếng Việt hoặc các ngôn ngữ đặc thù từ ảnh văn bản in (ví dụ: sách, báo, tài liệu hành chính). Đề tài không bao gồm xử lý văn bản viết tay do tính phức tạp và yêu cầu kỹ thuật cao hơn. Các ảnh văn bản được sử dụng trong nghiên cứu đảm bảo chất lượng rõ nét, không bị nhiễu hoặc biến dạng nghiêm trọng.

**Công Cụ**

* **Tesseract**: Phần mềm OCR mã nguồn mở, được sử dụng làm nền tảng chính để xây dựng và huấn luyện mô hình.
* **Python**: Hỗ trợ trong việc tiền xử lý dữ liệu, tạo tập huấn luyện, và tích hợp Tesseract vào quy trình xử lý.
* **Command Line (CMD)**: Sử dụng các lệnh dòng lệnh của Tesseract để huấn luyện và kiểm tra mô hình.

**Phương Pháp**

Nghiên cứu được thực hiện theo phương pháp thực nghiệm, bao gồm các bước chính:

* **Thu thập và tiền xử lý dữ liệu**: Tạo tập dữ liệu gồm 5000 mẫu ảnh văn bản tiếng Việt, kèm theo văn bản chuẩn (ground truth) tương ứng. Các ảnh được tiền xử lý để cải thiện chất lượng (ví dụ: điều chỉnh độ sáng, loại bỏ nhiễu).
* **Huấn luyện mô hình**: Sử dụng Tesseract để huấn luyện mô hình OCR tùy chỉnh, dựa trên tập dữ liệu đã chuẩn bị. Quá trình huấn luyện bao gồm việc tạo tệp ngôn ngữ, font chữ, và các tham số tối ưu.
* **Đánh giá mô hình**: Áp dụng mô hình đã huấn luyện trên tập dữ liệu kiểm tra riêng biệt. Tính toán các chỉ số CER và WER để đánh giá độ chính xác và so sánh với mô hình Tesseract mặc định.
* **Phân tích và cải tiến**: Dựa trên kết quả đánh giá, phân tích các lỗi nhận diện phổ biến và đề xuất các phương án cải tiến mô hình (nếu cần).

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

**1. Bài Toán Nhận Dạng Ký Tự Quang Học (OCR)**

**1.1 Định nghĩa OCR**

Nhận dạng ký tự quang học (Optical Character Recognition - OCR) 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 (pattern recognition). 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.

**1.2 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.

**1.3 Các thành phần chính của bài toán OCR**

Bài toán OCR thường bao gồm các bước sau:

1. **Thu nhận hình ảnh**: Hình ảnh chứa văn bản được thu thập từ các nguồn như máy quét, máy ảnh, hoặc tệp PDF.
2. **Tiền xử lý hình ảnh**: Hình ảnh được xử lý để cải thiện chất lượng, bao gồm:
   * Chuyển đổi sang thang độ xám hoặc nhị phân hóa.
   * Loại bỏ nhiễu (noise removal).
   * Điều chỉnh độ sáng, tương phản.
   * Xoay hoặc căn chỉnh ảnh nếu văn bản bị lệch (deskewing).
3. **Phân đoạn văn bản**: Xác định các vùng chứa văn bản, chia thành các dòng, từ, hoặc ký tự riêng lẻ.
4. **Nhận diện ký tự**: Sử dụng các thuật toán học máy hoặc học sâu để dự đoán các ký tự hoặc từ dựa trên đặc trưng hình ảnh.
5. **Hậu xử lý**: Sửa lỗi nhận diện bằng cách áp dụng từ điển, quy tắc ngữ pháp, hoặc các mô hình ngôn ngữ để đảm bảo văn bản đầu ra có ý nghĩa.

**1.4 Ứng dụng của OCR**

OCR có phạm vi ứng dụng rộng rãi trong nhiều lĩnh vực, bao gồm:

* **Quản lý tài liệu**: Số hóa sách, báo, tài liệu lịch sử để lưu trữ, tìm kiếm, và phân tích. Ví dụ, các thư viện sử dụng OCR để chuyển đổi sách cổ thành dạng kỹ thuật số, giúp bảo tồn và phổ biến kiến thức.
* **Trích xuất dữ liệu tự động**: Rút trích thông tin từ hóa đơn, biên lai, biểu mẫu, hoặc giấy tờ tùy thân. Ví dụ, các ngân hàng sử dụng OCR để tự động đọc thông tin từ chứng minh nhân dân hoặc hộ chiếu.
* **Nhận diện biển số xe**: Ứng dụng trong giao thông và an ninh, cho phép camera tự động nhận diện và ghi lại biển số xe trong thời gian thực.
* **Hỗ trợ người khiếm thị**: 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 dễ dàng hơn.
* **Xử lý tài liệu hành chính**: Tự động hóa việc nhập liệu từ các biểu mẫu, đơn từ, hoặc hợp đồng, giảm thiểu sai sót và tiết kiệm thời gian.
* **Ứng dụng trong dịch thuật**: Kết hợp OCR với công nghệ dịch máy để trích xuất và dịch văn bản từ hình ảnh, hỗ trợ giao tiếp đa ngôn ngữ.
* **Nhận diện chữ viết trong công nghiệp**: Đọc các mã số, nhãn sản phẩm, hoặc thông tin in trên bao bì trong dây chuyền sản xuất.
* **Ứng dụng trong giáo dục**: Chuyển đổi ghi chú viết tay hoặc tài liệu in thành văn bản số để lưu trữ hoặc chia sẻ.

**1.5 Ví dụ minh họa**

Để minh họa cách OCR hoạt động, hãy xem xét một ví dụ cụ thể:

* **Đầu vào**: Một ảnh chụp của một trang sách chứa đoạn văn:  
  *"Công nghệ OCR giúp chuyển đổi hình ảnh thành văn bản số, mở ra nhiều ứng dụng thực tiễn."*
* **Quá trình xử lý**:
  1. Hình ảnh được chuyển sang thang độ xám và loại bỏ nhiễu.
  2. Hệ thống OCR phân đoạn văn bản thành các dòng và từ.
  3. Mô hình nhận diện ký tự dự đoán từng ký tự, bao gồm các dấu thanh của tiếng Việt (ví dụ: "đổi", "thành").
  4. Hậu xử lý kiểm tra ngữ pháp và từ điển để đảm bảo văn bản có ý nghĩa.
* **Đầu ra**: Chuỗi văn bản số:  
  + Công nghệ OCR giúp chuyển đổi hình ảnh thành văn bản số, mở ra nhiều ứng dụng thực tiễn.  
  + Văn bản này có thể được lưu dưới dạng tệp .txt, .docx, hoặc tích hợp vào các ứng dụng khác như tìm kiếm hoặc phân tích dữ liệu.

**2. Công Nghệ Áp Dụng: Tesseract OCR**

**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 (HP) 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 (cho Python) hoặc giao diện dòng lệnh (CLI). 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.

**2.2 Lịch sử phát triển**

Tesseract đã trải qua nhiều giai đoạn phát triển, từ một dự án nội bộ đến một công cụ mã nguồn mở được cộng đồng toàn cầu sử dụng:

* **1985-1995**: Tesseract được khởi tạo bởi HP như một phần của nghiên cứu về nhận dạng ký tự. Trong giai đoạn này, Tesseract chủ yếu được sử dụng nội bộ và tập trung vào các thuật toán nhận dạng mẫu truyền thống.
* **1995**: HP quyết định mở mã nguồn Tesseract, nhưng dự án không nhận được nhiều sự chú ý do hạn chế về hiệu suất và khả năng hỗ trợ ngôn ngữ.
* **2005**: Tesseract được phát hành công khai dưới giấy phép Apache License 2.0, đánh dấu bước ngoặt trong việc thu hút cộng đồng mã nguồn mở.
* **2006**: Google tiếp quản dự án, đầu tư mạnh mẽ vào việc cải tiến Tesseract. Các phiên bản 2.x và 3.x được phát hành, tập trung vào việc mở rộng danh sách ngôn ngữ hỗ trợ và cải thiện độ chính xác nhận diện.
* **2018**: Tesseract 4.0 ra mắt, tích hợp mạng nơ-ron tái phát (Recurrent Neural Network - RNN) với đơn vị LSTM (Long Short-Term Memory). Phiên bản này cải thiện đáng kể khả năng nhận diện văn bản phức tạp và font chữ đa dạng.
* **Hiện tại (2025)**: Tesseract tiếp tục được duy trì bởi Google và cộng đồng mã nguồn mở. Các bản cập nhật gần đây tập trung vào tối ưu hóa hiệu suất, hỗ trợ thêm ngôn ngữ, và cải thiện khả năng huấn luyện mô hình tùy chỉnh. Phiên bản mới nhất (tính đến 2025) là Tesseract 5.x, với các cải tiến về tốc độ xử lý và độ chính xác cho các ngôn ngữ có ký tự đặc biệt.

**2.3 Kiến trúc Tesseract 4**

Tesseract 4 sử dụng một kiến trúc OCR hiện đại, kết hợp các kỹ thuật xử lý hình ảnh truyền thống với mạng nơ-ron học sâu, cụ thể là mô hình LSTM. Kiến trúc này cho phép Tesseract nhận diện văn bản theo ngữ cảnh, cải thiện khả năng xử lý các ký tự phức tạp, font chữ không chuẩn, và các ngôn ngữ có cấu trúc đặc biệt. Quy trình xử lý của Tesseract bao gồm các bước chính sau:

1. **Tiền xử lý hình ảnh**:
   * Chuyển đổi hình ảnh sang thang độ xám hoặc nhị phân hóa để giảm độ phức tạp.
   * Loại bỏ nhiễu bằng các bộ lọc hình ảnh (ví dụ: Gaussian blur).
   * Điều chỉnh độ sáng, tương phản, hoặc căn chỉnh ảnh để đảm bảo văn bản dễ đọc.
   * Phát hiện và sửa các lỗi xoay lệch (deskewing) hoặc biến dạng hình ảnh.
2. **Phân đoạn văn bản**:
   * Xác định các vùng chứa văn bản (text regions) trong hình ảnh.
   * Chia nhỏ văn bản thành các dòng, từ, và ký tự riêng lẻ.
   * Phát hiện bố cục phức tạp như bảng biểu hoặc cột văn bản.
3. **Nhận diện ký tự**:
   * Sử dụng mô hình LSTM để dự đoán các ký tự hoặc từ dựa trên đặc trưng hình ảnh.
   * Kết hợp thông tin ngữ cảnh (context) để cải thiện độ chính xác, đặc biệt với các ngôn ngữ có dấu như tiếng Việt.
   * Áp dụng tệp traineddata chứa dữ liệu huấn luyện cho ngôn ngữ cụ thể.
4. **Hậu xử lý**:
   * Kiểm tra từ điển để sửa lỗi nhận diện (ví dụ: thay thế ký tự sai thành từ đúng).
   * Áp dụng quy tắc ngữ pháp và mô hình ngôn ngữ để đảm bảo văn bản đầu ra có ý nghĩa.
   * Định dạng văn bản đầu ra thành các định dạng như text, HTML, hoặc PDF.

**2.4 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 (ví dụ: "vie" cho tiếng Việt) 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.

**2.5 Hỗ trợ tiếng Việt**

Tesseract hỗ trợ tiếng Việt thông qua tệp traineddata mặc định cho ngôn ngữ "vie". Tuy nhiên, để đạt độ chính xác cao với các font chữ đặc thù hoặc văn bản tiếng Việt có bố cục phức tạp, người dùng cần huấn luyện lại mô hình với dữ liệu riêng. Quá trình huấn luyện này bao gồm việc tạo tập dữ liệu chứa ảnh văn bản tiếng Việt và văn bản chuẩn tương ứng, sau đó sử dụng lstmtraining để tối ưu hóa mô hình.

**3. Ưu Điểm và Lý Do Chọn Tesseract**

**3.1 Ưu điểm của Tesseract**

Tesseract sở hữu nhiều ưu điểm vượt trội, khiến nó trở thành lựa chọn hàng đầu cho các dự án OCR, đặc biệt trong nghiên cứu học thuật và ứng dụng tùy chỉnh:

* **Miễn phí và mã nguồn mở**: Tesseract được cung cấp miễn phí dưới giấy phép Apache 2.0, cho phép người dùng sử dụng, chỉnh sửa, và tích hợp vào các dự án mà không phải trả phí bản quyền. Điều này đặc biệt hữu ích cho các dự án có ngân sách hạn chế.
* **Dễ tích hợp**: Tesseract hỗ trợ nhiều ngôn ngữ lập trình như Python (thông qua Pytesseract), C++, và Java. Ngoài ra, giao diện dòng lệnh (CLI) của Tesseract cho phép sử dụng trực tiếp mà không cần tích hợp phức tạp.
* **Hỗ trợ tùy biến mô hình**: Tesseract cho phép huấn luyện mô hình tùy chỉnh trên dữ liệu riêng, rất phù hợp với các ngôn ngữ đặc thù như tiếng Việt hoặc font chữ không phổ biến. Người dùng có thể tạo tệp traineddata mới để tối ưu hóa hiệu suất nhận diện.
* **Khả năng huấn luyện với dữ liệu nhỏ**: Không giống như các mô hình học sâu khác yêu cầu hàng chục nghìn mẫu dữ liệu, Tesseract có thể đạt kết quả tốt với tập dữ liệu nhỏ hơn (vài nghìn mẫu), giúp tiết kiệm thời gian và tài nguyên.
* **Hỗ trợ đa ngôn ngữ**: Tesseract hỗ trợ hơn 100 ngôn ngữ, bao gồm tiếng Việt, và có thể xử lý văn bản đa ngôn ngữ trong cùng một hình ảnh.
* **Cộng đồng hỗ trợ mạnh mẽ**: Là một dự án mã nguồn mở, Tesseract có một cộng đồng phát triển tích cực, cung cấp tài liệu, hướng dẫn, và công cụ hỗ trợ phong phú. Người dùng có thể tìm thấy các tài nguyên như mẫu dữ liệu huấn luyện, diễn đàn thảo luận, và mã nguồn mở trên GitHub.
* **Hiệu suất tốt với văn bản in**: Tesseract đặc biệt hiệu quả với các tài liệu in chất lượng cao, chẳng hạn như sách, báo, hoặc biểu mẫu hành chính, vốn là trọng tâm của đề tài này.

**3.2 Lý do chọn Tesseract**

Tesseract được chọn làm công nghệ chính cho đề tài này vì các lý do sau:

* **Phù hợp với tiếng Việt**: Tesseract hỗ trợ tiếng Việt và cho phép huấn luyện mô hình tùy chỉnh để cải thiện độ chính xác với các ký tự có dấu (â, ê, ơ, dấu huyền, sắc, hỏi, ngã, nặng) và font chữ đặc thù. Điều này rất quan trọng khi xử lý các tài liệu tiếng Việt như sách giáo khoa, báo chí, hoặc giấy tờ hành chính.
* **Chi phí thấp**: Là giải pháp miễn phí, Tesseract phù hợp với các dự án học thuật hoặc nghiên cứu không có ngân sách lớn, trong khi vẫn cung cấp hiệu suất cạnh tranh với các công cụ thương mại.
* **Tính linh hoạt**: Tesseract có thể được tích hợp 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.
* **Khả năng mở rộng**: Tesseract hỗ trợ huấn luyện thêm dữ liệu để cải thiện độ chính xác, đồng thời có thể kết hợp với các công nghệ học sâu khác (như mạng nơ-ron convolution) để nâng cao hiệu suất trong các trường hợp phức tạp.
* **Tài liệu và cộng đồng phong phú**: Với lượng lớn tài liệu hướng dẫn và sự hỗ trợ từ cộng đồng mã nguồn mở, Tesseract dễ dàng được triển khai và tối ưu hóa cho các nhu cầu cụ thể.

**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:

|  |  |  |  |
| --- | --- | --- | --- |
| **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 Python, C++, CLI | API REST, cần kết nối internet | Dễ dàng với 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. Tuy nhiên, độ chính xác ban đầu có thể thấp hơn nếu không huấn luyện đúng cách.
* **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.

**Lưu Đồ Pipeline Xử Lý Ảnh Văn Bản**

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

1. **Đầu vào (Input Image)**: Hình ảnh chứa văn bản, thường ở định dạng JPEG, PNG, hoặc PDF.
2. **Tiền xử lý hình ảnh (Preprocessing)**:
   * Chuyển đổi sang thang độ xám hoặc nhị phân hóa để giảm độ phức tạp.
   * Loại bỏ nhiễu bằng các bộ lọc như Gaussian blur hoặc median filter.
   * Điều chỉnh độ sáng, tương phản để làm nổi bật văn bản.
   * Xoay hoặc căn chỉnh ảnh để sửa lỗi lệch (deskewing).
3. **Phân đoạn văn bản (Text Segmentation)**:
   * Xác định các vùng chứa văn bản (text regions) trong hình ảnh.
   * Chia nhỏ văn bản thành các dòng (lines), từ (words), và ký tự (characters).
   * Xử lý các bố cục phức tạp như bảng biểu hoặc văn bản đa cột.
4. **Nhận diện ký tự (Character Recognition)**:
   * Sử dụng mô hình LSTM để dự đoán các ký tự hoặc từ dựa trên đặc trưng hình ảnh.
   * Áp dụng tệp traineddata của ngôn ngữ cụ thể (ví dụ: "vie" cho tiếng Việt).
   * Kết hợp thông tin ngữ cảnh để cải thiện độ chính xác.
5. **Hậu xử lý (Post-processing)**:
   * Kiểm tra từ điển để sửa lỗi nhận diện (ví dụ: thay thế "hinh anh" thành "hình ảnh").
   * Áp dụng quy tắc ngữ pháp và mô hình ngôn ngữ để đảm bảo văn bản có ý nghĩa.
   * Định dạng văn bản đầu ra thành các định dạng như text, HTML, hoặc PDF.
6. **Đầu ra (Output Text)**: Chuỗi văn bản số có thể chỉnh sửa, tìm kiếm, hoặc tích hợp vào các ứng dụng khác.