**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ ĐÔNG Á**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP LỚN**

**HỌC PHẦN: XỬ LÝ ẢNH VÀ THỊ GIÁC MÁY TÍNH**

**Đề tài số 6: Xây dựng hệ thống nhận diện chữ số viết tay**

**Giảng viên hướng dẫn: Lương Thị Hồng Lan**

|  |  |  |  |
| --- | --- | --- | --- |
| **TT** | **Mã sinh viên** | **Sinh viên thực hiện** | **Lớp hành chính** |
| **1** | 20210102 | Nguyễn Tấn Dũng | DCCNTT 12.10.1 |
| **2** | 20210297 | Tạ Đức Thư | DCCNTT 12.10.1 |

**Bắc Ninh, năm 2024**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ ĐÔNG Á**

**KHOA CÔNG NGHỆ THÔNG TIN**

**BÀI TẬP LỚN**

**HỌC PHẦN: XỬ LÝ ẢNH VÀ THỊ GIÁC MÁY TÍNH**

**Đề tài số 6: Xây dựng hệ thống nhận diện chữ số viết tay**

**Giảng viên hướng dẫn: Lương Thị Hồng Lan**

|  |  |  |  |
| --- | --- | --- | --- |
| **TT** | **Mã sinh viên** | **Sinh viên thực hiện** | **Lớp hành chính** |
| **1** | 20210102 | Nguyễn Tấn Dũng | DCCNTT 12.10.1 |
| **2** | 20210297 | Tạ Đức Thư | DCCNTT 12.10.1 |

|  |  |
| --- | --- |
|  |  |

**Bắc Ninh, năm 2024**

|  |  |
| --- | --- |
| TRƯỜNG ĐẠI HỌC CÔNG NGHỆ ĐÔNG Á  **KHOA CÔNG NGHỆ THÔNG TIN** | **KỲ THI KẾT THÚC HỌC PHẦN**  **HỌC KỲ 1, NĂM HỌC 2024** – **2025** |

|  |  |
| --- | --- |
| **PHIẾU CHẤM THI BÀI TẬP LỚN KẾT THÚC HỌC PHẦN**  **Mã đề thi: 6**  **Tên học phần: XỬ LÝ ẢNH VÀ THỊ GIÁC MÁY TÍNH**  **Lớp Tín chỉ:** XATGMT.03.K12.01.LH.C04.1\_LT | |
| **Cán bộ chấm thi 1**  *(Ký và ghi rõ họ tên)*  Lê Thị Hồng Lan | **Cán bộ chấm thi 2**  *(Ký và ghi rõ họ tên)* |

| **TT** | **TIÊU CHÍ** | **THANG ĐIỂM** | **TÊN SV 1** | **TÊN SV 2** | **TÊN SV 3** | **TÊN SV 4** | **TÊN SV 5** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| MÃ SV 1 | MÃ SV 2 | MÃ SV 3 | MÃ SV 4 | MÃ SV 5 |
| **1** | **Nội dung báo cáo trên Word đầy đủ** | **3.5** |  |  |  |  |  |
| 1.1 | Có bố cục rõ ràng (mục lục, phần mở đầu, nội dung chính, kết luận). | 0,5 |  |  |  |  |  |
| 1.2 | Nội dung phân tích rõ ràng, logic. | 0,5 |  |  |  |  |  |
| 1.3 | Có dẫn chứng, số liệu minh họa đầy đủ. | 0,5 |  |  |  |  |  |
| 1.4 | Ngôn ngữ và trình bày chuẩn, không lỗi chính tả. | 0,5 |  |  |  |  |  |
| 1.5 | Có trích dẫn tài liệu tham khảo đúng quy cách. | 0,5 |  |  |  |  |  |
| 1.6 | Được trình bày chuyên nghiệp (canh lề, font chữ, khoảng cách dòng hợp lý). | 0,5 |  |  |  |  |  |
| 1.7 | Tài liệu đầy đủ, bám sát yêu cầu của đề bài. | 0,5 |  |  |  |  |  |
| **2** | **Nội dung thuyết trình đầy đủ** | **1.0** |  |  |  |  |  |
| 2.1 | Trình bày tự tin, phát âm rõ ràng, mạch lạc. | 0,5 |  |  |  |  |  |
| 2.2 | Nội dung thuyết trình đúng trọng tâm, không lan man. | 0,5 |  |  |  |  |  |
| **3** | **Slides báo cáo đầy đủ nội dung + Hỏi đáp** | **3.0** |  |  |  |  |  |
| 3.1 | Slides có bố cục rõ ràng (mở đầu, nội dung, kết luận). | 0,5 |  |  |  |  |  |
| 3.2 | Thiết kế slides đẹp, chuyên nghiệp (màu sắc, hình ảnh minh họa). | 0,5 |  |  |  |  |  |
| 3.3 | Nội dung trên slides ngắn gọn, dễ hiểu, súc tích. | 0,5 |  |  |  |  |  |
| 3.4 | Nội dung slides phù hợp với nội dung báo cáo. | 0,5 |  |  |  |  |  |
| 3.5 | Trả lời câu hỏi đầy đủ, chính xác. | 0,5 |  |  |  |  |  |
| 3.6 | Trả lời câu hỏi tự tin, thuyết phục. | 0,5 |  |  |  |  |  |
| **4** | **Code đầy đủ** | **2.5** |  |  |  |  |  |
| 1.1 | Code được trình bày rõ ràng, có chú thích đầy đủ. | 0,5 |  |  |  |  |  |
| 1.2 | Code chạy đúng, không lỗi. | 0,5 |  |  |  |  |  |
| 1.3 | Code tối ưu, không dư thừa. | 0,5 |  |  |  |  |  |
| 1.4 | Đáp ứng đầy đủ các yêu cầu chức năng theo đề bài. | 0,5 |  |  |  |  |  |
| 1.5 | Có tính sáng tạo hoặc cải thiện so với yêu cầu. | 0,5 |  |  |  |  |  |
| **TỔNG ĐIỂM BẰNG SỐ:** | | **10** |  |  |  |  |  |
| **TỔNG ĐIỂM BẰNG CHỮ:** | | *Mười tròn* |  |  |  |  |  |

**Mục Lục**

[**CHƯƠNG 1. CƠ SỞ LÝ THUYẾT 8**](#_Toc184687924)

[**1. Xử lý ảnh và thị giác máy tính 8**](#_Toc184687925)

[**1.1. Xử lý ảnh 8**](#_Toc184687926)

[**1.2. Thị giác máy tính 8**](#_Toc184687927)

[**1.3. Cách thị giác máy tính hoạt động 9**](#_Toc184687928)

[**1.4. Lịch sử phát triển của thị giác máy tính 10**](#_Toc184687929)

[**1.5. Nhận dạng đối tượng 11**](#_Toc184687930)

[**2. Nhận dạng đối tượng và bài toán đặt ra 11**](#_Toc184687931)

[**2.1 Phát biểu bài toán 11**](#_Toc184687932)

[**2.2 Các kỹ thuật sử dụng trong bài toán nhận dạng 12**](#_Toc184687933)

[**2.3. Ngôn ngữ lập trình và ứng dụng sử dụng 17**](#_Toc184687934)

[**2.3.1.Visual code 17**](#_Toc184687935)

[**2.3.2. Python 19**](#_Toc184687936)

[**CHƯƠNG 2: XÂY DỰNG HỆ THỐNG 22**](#_Toc184687937)

[**2.1 Bài toán 22**](#_Toc184687938)

[**2.1.1 Đặt vấn đề 22**](#_Toc184687939)

[**2.1.2 Mô tả bài toán 22**](#_Toc184687940)

[**2.2. Xây dựng hệ thống 24**](#_Toc184687941)

[**2.2.1 Các thư viện 24**](#_Toc184687942)

[**2.3. Một số mã nguồn quan trọng 26**](#_Toc184687943)

[**CHƯƠNG 3: KẾT QUẢ THỰC NGHIỆM 30**](#_Toc184687944)

[**3.1. Dữ liệu 30**](#_Toc184687945)

[**3.2. Kết quả thực nghiệm 30**](#_Toc184687946)

[**3.2.1. Chạy tự động ảnh 30**](#_Toc184687947)

[**3.2.2. Nhận diện chữ viết tay thủ công 31**](#_Toc184687948)

[**KẾT LUẬN 33**](#_Toc184687949)

**Lời nói đầu**

Nhận diện chữ viết tay (Handwritten Text Recognition - HTR) là một lĩnh vực quan trọng trong nghiên cứu trí tuệ nhân tạo (AI) và học máy (machine learning), đặc biệt trong bối cảnh số hóa thông tin và tự động hóa các quy trình xử lý dữ liệu. Chữ viết tay là một dạng dữ liệu đặc biệt do sự biến thiên lớn trong cách viết của mỗi người, gây khó khăn cho việc nhận diện chính xác. Tuy nhiên, với sự phát triển mạnh mẽ của các công nghệ như học sâu (deep learning) và mạng nơ-ron nhân tạo (neural networks), việc xây dựng các hệ thống nhận diện chữ viết tay ngày càng trở nên hiệu quả và chính xác hơn.

Mục tiêu của hệ thống nhận diện chữ viết tay là chuyển đổi các ký tự và từ ngữ viết tay thành dạng văn bản số hóa có thể xử lý và lưu trữ trên máy tính. Điều này không chỉ giúp tự động hóa việc nhập liệu từ các tài liệu giấy mà còn hỗ trợ trong các ứng dụng nhận diện chữ viết tay như quét chữ ký, biên dịch các tài liệu khảo sát, hoặc nhận diện văn bản từ ảnh và tài liệu số.

Trong quá trình xây dựng hệ thống nhận diện chữ viết tay, có nhiều thử thách cần phải vượt qua, bao gồm việc xử lý sự đa dạng trong phong cách viết, độ nghiêng của văn bản, chất lượng của hình ảnh đầu vào, và đặc biệt là việc phân biệt giữa các ký tự giống nhau nhưng có hình dạng khác biệt. Những tiến bộ trong các kỹ thuật học máy như mạng nơ-ron hồi tiếp (RNN), mạng nơ-ron tích chập (CNN), và các phương pháp học không giám sát đã giúp cải thiện đáng kể hiệu suất nhận diện.

Hệ thống nhận diện chữ viết tay hiện nay không chỉ đơn thuần là nhận diện các ký tự riêng biệt mà còn có khả năng nhận diện các từ, câu và thậm chí là cấu trúc ngữ pháp trong văn bản viết tay. Các ứng dụng của hệ thống này ngày càng trở nên phổ biến trong nhiều lĩnh vực, từ giáo dục, y tế đến ngân hàng và công nghiệp dịch vụ.

Thông qua việc nghiên cứu và phát triển các mô hình nhận diện chữ viết tay, chúng ta không chỉ cải thiện hiệu suất công việc mà còn mở rộng khả năng tương tác với máy tính, góp phần thúc đẩy sự tiến bộ trong các công nghệ tự động hóa và trí tuệ nhân tạo.

**CHƯƠNG 1. CƠ SỞ LÝ THUYẾT**

**1. Xử lý ảnh và thị giác máy tính**

**1.1. Xử lý ảnh**

Xử lý ảnh (Tiếng Anh: digital image processing) hay xử lý ảnh kỹ thuật số là sự sử dụng máy tính số để xử lý các ảnh kỹ thuật số thông qua một thuật toán. Là một phân ngành tổng thể của xử lý tín hiệu số, xử lý ảnh kỹ thuật số có nhiều lợi thế hơn so với xử lý ảnh analog. Nó cho phép áp dụng nhiều thuật toán hơn cho dữ liệu đầu vào và có thể tránh được các vấn đề như sự tích tụ nhiễu và méo hình trong quá trình xử lý. Vì ảnh được định nghĩa trên hai chiều (hoặc nhiều hơn), xử lý ảnh kỹ thuật số có thể được mô hình hóa dưới dạng hệ thống đa chiều. Sự phát triển của xử lý ảnh kỹ thuật số chủ yếu bị ảnh hưởng bởi ba yếu tố: đầu tiên là sự phát triển của máy tính; thứ hai là sự phát triển của toán học (đặc biệt là sáng tạo và cải tiến lý thuyết toán rời rạc); thứ ba là nhu cầu ứng dụng rộng rãi trong môi trường, nông nghiệp, quân sự, công nghiệp và khoa học y tế ngày càng tăng.

**1.2. Thị giác máy tính**

 Thị giác máy tính là một lĩnh vực của trí tuệ nhân tạo (AI) sử dụng machine learning và mạng nơ-ron để dạy máy tính và hệ thống cách trích xuất thông tin có ý nghĩa từ hình ảnh kỹ thuật số, video và các đầu vào trực quan khác, cũng như đưa ra khuyến nghị hoặc thực hiện hành động khi chúng nhìn thấy lỗi hoặc sự cố.

Thị giác máy tính hoạt động giống như thị giác của con người, ngoại trừ việc con người có lợi thế hơn. Thị giác của con người có lợi thế là có nhiều bối cảnh trong suốt cuộc đời để đào tạo cách phân biệt các vật thể, chúng cách xa bao nhiêu, chúng có đang di chuyển hay có gì đó không ổn với hình ảnh không.

Thị giác máy tính đào tạo máy móc thực hiện các chức năng này, nhưng phải thực hiện trong thời gian ngắn hơn nhiều bằng máy ảnh, dữ liệu và thuật toán thay vì võng mạc, dây thần kinh thị giác và vỏ não thị giác. Bởi vì một hệ thống được đào tạo để kiểm tra sản phẩm hoặc theo dõi tài sản sản xuất có thể phân tích hàng nghìn sản phẩm hoặc quy trình mỗi phút, nhận thấy những khiếm khuyết hoặc vấn đề không thể nhận thấy, nên nó có thể nhanh chóng vượt qua khả năng của con người.

## 1.3. Cách thị giác máy tính hoạt động

Thị giác máy tính cần rất nhiều dữ liệu. Nó chạy phân tích dữ liệu liên tục cho đến khi phân biệt được sự khác biệt và cuối cùng là nhận dạng được hình ảnh. Ví dụ, để đào tạo máy tính nhận dạng lốp ô tô, cần phải cung cấp cho máy tính một lượng lớn hình ảnh lốp xe và các mục liên quan đến lốp xe để tìm hiểu sự khác biệt và nhận dạng lốp xe, đặc biệt là lốp xe không có khuyết tật.

Hai công nghệ thiết yếu được sử dụng để thực hiện điều này: một loại machine learning được gọi là deep learning và mạng nơ-ron tích chập (CNN).

Máy học sử dụng các mô hình thuật toán cho phép máy tính tự học về bối cảnh của dữ liệu trực quan. Nếu đủ dữ liệu được cung cấp thông qua mô hình, máy tính sẽ "xem" dữ liệu và tự học cách phân biệt một hình ảnh với hình ảnh khác. Các thuật toán cho phép máy tự học, thay vì ai đó lập trình để nhận dạng hình ảnh.

CNN giúp mô hình machine learning hoặc deep learning "xem" bằng cách chia nhỏ hình ảnh thành các pixel được gắn thẻ hoặc nhãn. Nó sử dụng các nhãn để thực hiện phép tích chập (một phép toán trên hai hàm để tạo ra hàm thứ ba) và đưa ra dự đoán về những gì nó "nhìn thấy". Mạng nơ-ron chạy các phép tích chập và kiểm tra độ chính xác của các dự đoán của nó trong một loạt các lần lặp cho đến khi các dự đoán bắt đầu trở thành sự thật. Sau đó, nó nhận dạng hoặc nhìn thấy hình ảnh theo cách tương tự như con người.

Giống như con người tạo ra một hình ảnh ở xa, CNN đầu tiên phân biệt các cạnh cứng và hình dạng đơn giản, sau đó điền thông tin khi chạy các lần lặp của các dự đoán của nó. CNN được sử dụng để hiểu các hình ảnh đơn lẻ. Mạng nơ-ron hồi quy (RNN) được sử dụng theo cách tương tự cho các ứng dụng video để giúp máy tính hiểu cách các hình ảnh trong một loạt các khung hình liên quan đến nhau.

## 1.4. Lịch sử phát triển của thị giác máy tính

Các nhà khoa học và kỹ sư đã cố gắng phát triển các phương pháp để máy móc có thể nhìn và hiểu dữ liệu hình ảnh trong khoảng 60 năm. Thí nghiệm bắt đầu vào năm 1959 khi các nhà sinh lý học thần kinh cho một con mèo xem một loạt hình ảnh, cố gắng liên hệ phản ứng trong não của nó. Họ phát hiện ra rằng nó phản ứng đầu tiên với các cạnh cứng hoặc đường thẳng và về mặt khoa học, điều này có nghĩa là quá trình xử lý hình ảnh bắt đầu bằng các hình dạng đơn giản như các cạnh thẳng.

Vào khoảng thời gian đó, công nghệ quét hình ảnh máy tính đầu tiên đã được phát triển, cho phép máy tính số hóa và thu thập hình ảnh. Một cột mốc khác đã đạt được vào năm 1963 khi máy tính có thể chuyển đổi hình ảnh hai chiều thành dạng ba chiều. Vào những năm 1960, AI nổi lên như một lĩnh vực nghiên cứu hàn lâm và nó cũng đánh dấu sự khởi đầu của nhiệm vụ AI nhằm giải quyết vấn đề thị giác của con người.

Năm 1974 chứng kiến sự ra đời của công nghệ nhận dạng ký tự quang học (OCR), có thể nhận dạng văn bản được in bằng bất kỳ phông chữ hoặc kiểu chữ nào. Tương tự như vậy, nhận dạng ký tự thông minh (ICR) có thể giải mã văn bản viết tay bằng mạng nơ-ron. Kể từ đó, OCR và ICR đã tìm được đường vào xử lý tài liệu và hóa đơn, nhận dạng biển số xe, thanh toán di động, chuyển đổi máy và các ứng dụng phổ biến khác.

Năm 1982, nhà khoa học thần kinh David Marr đã xác định rằng thị giác hoạt động theo thứ bậc và giới thiệu các thuật toán để máy móc phát hiện các cạnh, góc, đường cong và các hình dạng cơ bản tương tự. Đồng thời, nhà khoa học máy tính Kunihiko Fukushima đã phát triển một mạng lưới các tế bào có thể nhận dạng các mẫu. Mạng lưới này, được gọi là Neocognitron, bao gồm các lớp tích chập trong mạng nơ-ron.

Đến năm 2000, trọng tâm nghiên cứu là nhận dạng đối tượng; và đến năm 2001, các ứng dụng nhận dạng khuôn mặt thời gian thực đầu tiên đã xuất hiện. Việc chuẩn hóa cách gắn thẻ và chú thích các tập dữ liệu trực quan đã xuất hiện trong những năm 2000. Năm 2010, tập dữ liệu ImageNet đã có sẵn. Nó chứa hàng triệu hình ảnh được gắn thẻ trên một nghìn lớp đối tượng và cung cấp nền tảng cho CNN và các mô hình học sâu được sử dụng ngày nay. Năm 2012, một nhóm từ Đại học Toronto đã đưa CNN vào cuộc thi nhận dạng hình ảnh. Mô hình có tên AlexNet đã giảm đáng kể tỷ lệ lỗi khi nhận dạng hình ảnh. Sau bước đột phá này, tỷ lệ lỗi đã giảm xuống chỉ còn vài phần trăm.

## 1.5. Nhận dạng đối tượng

**1.5.1 Nhận dạng đối tượng là gì ?**

Nhận dạng đối tượng là một lĩnh vực của trí tuệ nhân tạo và thị giác máy tính, nhằm mục tiêu cho phép máy tính có khả năng xác định và phân loại các đối tượng có trong một hình ảnh hoặc video. Nói cách khác, đó là quá trình máy tính "hiểu" những gì nó đang "nhìn thấy".

**1.5.2. Sự khác biệt giữa nhận dạng đối tượng và phát hiện đối tượng**

Mặc dù hai thuật ngữ này thường được sử dụng thay thế cho nhau, nhưng chúng có những ý nghĩa hơi khác nhau:

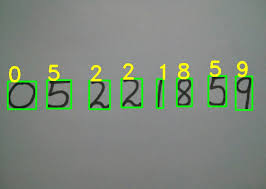
**- Phát hiện đối tượng (Object Detection):** Tập trung vào việc xác định vị trí của các đối tượng trong một hình ảnh, thường bằng cách vẽ các khung bao quanh các đối tượng đó.

**- Nhận dạng đối tượng (Object Recognition):** Tập trung vào việc xác định loại của các đối tượng đó.

## 2. Nhận dạng đối tượng và bài toán đặt ra

## 2.1 Phát biểu bài toán

Xây dựng một hệ thống có khả năng tự động phát hiện và xác định các đối tượng cụ thể trong một hình ảnh, đồng thời trích xuất các thông tin liên quan đến đối tượng đó.



Hình: Minh họa nhận diện đối tượng

**- Hình ảnh đầu vào:** Một bức ảnh chứa nhiều đối tượng khác nhau (ví dụ: người, xe, vật dụng).

**- Kết quả đầu ra:** Hệ thống sẽ vẽ các khung bao quanh các đối tượng được phát hiện và đưa ra nhãn cho từng đối tượng (ví dụ: "người", "xe ô tô"). Ngoài ra, hệ thống có thể cung cấp các thông tin bổ sung như vị trí, kích thước, màu sắc của đối tượng.

Các vấn đề cần đặt ra:

**- Các loại đối tượng:** Hệ thống sẽ phát hiện và nhận dạng những loại đối tượng nào? (ví dụ: người, mặt, xe, biển báo giao thông)

**- Độ chính xác:** Hệ thống cần đạt được độ chính xác bao nhiêu khi phát hiện và nhận dạng đối tượng?

**- Tốc độ:** Hệ thống cần xử lý hình ảnh với tốc độ như thế nào để đáp ứng các ứng dụng thực tế?

**- Môi trường làm việc:** Hệ thống có thể hoạt động tốt trong các điều kiện ánh sáng khác nhau, các góc chụp khác nhau hay không?

**- Dữ liệu huấn luyện:** Cần bao nhiêu dữ liệu huấn luyện để hệ thống đạt hiệu quả cao?

## 2.2 Các kỹ thuật sử dụng trong bài toán nhận dạng

**2.2.1.. Phương pháp truyền thống:**

**-** DBSCAN (Density-Based Spatial Clustering).

**Ưu điểm của DBSCAN:**

**+ Xác định được các cụm không có hình dạng cố định**: DBSCAN có thể phát hiện các cụm dạng phức tạp (không nhất thiết là hình cầu như K-means).

**Loại bỏ nhiễu**: Các điểm nhiễu được nhận diện và loại bỏ khỏi cụm.

**Không cần biết số cụm trước**: Số cụm được tự động xác định dựa trên dữ liệu.

**Nhược điểm của DBSCAN:**

**+ Phụ thuộc vào tham số**:

Kết quả phụ thuộc nhiều vào việc lựa chọn **ε** và **MinPts**.

Cần tinh chỉnh các tham số phù hợp với từng tập dữ liệu.

**+ Không hiệu quả với dữ liệu mật độ thay đổi**:

Nếu dữ liệu có cụm với mật độ thay đổi lớn, DBSCAN có thể hoạt động kém.

**+ Tốn kém khi dữ liệu lớn**:

Với các tập dữ liệu lớn, DBSCAN có thể chậm vì cần tính toán khoảng cách giữa các điểm.

**2.2.2. Học máy hoạt động như thế nào?**

- Quá trình học máy thường bao gồm các bước sau:

**+ Thu thập và chuẩn bị dữ liệu:** Dữ liệu đầu vào được thu thập và xử lý để loại bỏ nhiễu, xử lý dữ liệu thiếu, và chuyển đổi dữ liệu thành định dạng phù hợp cho thuật toán học máy.

**+ Chọn mô hình và thuật toán:** Lựa chọn mô hình và thuật toán học máy phù hợp với bài toán cụ thể.

**+ Huấn luyện mô hình:** Sử dụng dữ liệu huấn luyện để điều chỉnh các tham số của mô hình sao cho mô hình có thể đưa ra dự đoán chính xác.

**+ Đánh giá mô hình:** Đánh giá hiệu suất của mô hình trên tập dữ liệu kiểm tra (khác với tập huấn luyện) để đảm bảo mô hình không bị quá khớp (overfitting) và có khả năng tổng quát hóa tốt trên dữ liệu mới.

**+ Triển khai và dự đoán:** Sử dụng mô hình đã huấn luyện để đưa ra dự đoán trên dữ liệu mới.

**2.2.3. Phương pháp học máy**

**- Học có giám sát (Supervised Learning)**

**Học có giám sát** được sử dụng ở phần nhận dạng chữ viết tay, cụ thể là khi huấn luyện mô hình học sâu để dự đoán các ký tự từ ảnh viết tay. Dấu hiệu cho thấy học có giám sát được áp dụng trong dự án:

**+ Sử dụng**:

**CTC Loss**: Hàm mất mát (CTC loss) trong ctc.py cho thấy việc sử dụng một mô hình học sâu, và các mô hình này thường yêu cầu dữ liệu đã được gán nhãn (ảnh viết tay được gán nhãn với văn bản tương ứng) để huấn luyện.

**Nhận dạng văn bản**: Trong bước nhận dạng văn bản, mô hình học sâu dự đoán các ký tự từ các vùng ảnh (bounding box) và so sánh kết quả với nhãn thực tế (ground truth) để tối ưu hóa mô hình.

**+ Cách hoạt động**:

Một tập dữ liệu gán nhãn bao gồm các ảnh chữ viết tay và văn bản tương ứng được sử dụng để huấn luyện mô hình.

Mô hình được huấn luyện để tối thiểu hóa lỗi dự đoán (CTC loss) giữa kết quả đầu ra và nhãn thực tế.

**- Học không giám sát (Unsupervised Learning)**

**Học không giám sát** được áp dụng ở phần phát hiện từ và phân nhóm các bounding box, cụ thể là thuật toán DBSCAN trong aabb\_clustering.py.

**+ Sử dụng:**

**DBSCAN**: Đây là một thuật toán học không giám sát dùng để phân cụm các bounding box (các vùng chứa văn bản) dựa trên khoảng cách Jaccard. DBSCAN không yêu cầu dữ liệu gán nhãn mà tự động nhóm các vùng dựa trên mật độ và khoảng cách.

**+ Cách hoạt động**:

Sau khi các bounding box được phát hiện trong ảnh, DBSCAN sẽ nhóm các bounding box gần nhau thành các cụm, tương ứng với các từ hoặc cụm từ trong ảnh.

Không cần nhãn cụ thể cho từng bounding box hoặc cụm, vì việc nhóm chỉ dựa trên khoảng cách và mối quan hệ giữa các bounding box.

**2.2.4. Các thuật toán học máy phổ biến**

Một số thuật toán học máy phổ biến bao gồm:

**- Hồi quy tuyến tính (Linear Regression):** Thuật toán này được sử dụng để dự đoán các giá trị số, dựa trên mối quan hệ tuyến tính giữa các giá trị khác nhau. Ví dụ, kỹ thuật này có thể được sử dụng để dự đoán giá nhà dựa trên dữ liệu lịch sử của khu vực.

**- Hồi quy logistic (Logistic Regression):** Thuật toán học có giám sát này đưa ra dự đoán cho các biến phản hồi theo danh mục, chẳng hạn như câu trả lời "có/không" cho các câu hỏi. Thuật toán này có thể được sử dụng cho các ứng dụng như phân loại thư rác và kiểm soát chất lượng trên dây chuyền sản xuất.

**- Cây quyết định (Decision Tree):** Cây quyết định có thể được sử dụng để dự đoán các giá trị số (hồi quy) và phân loại dữ liệu thành các danh mục. Cây quyết định sử dụng chuỗi phân nhánh các quyết định được liên kết có thể được biểu diễn bằng sơ đồ cây. Một trong những ưu điểm của cây quyết định là chúng dễ xác thực và kiểm tra, không giống như hộp đen của mạng nơ-ron.

**- Máy vectơ hỗ trợ (Support Vector Machine - SVM):** Tìm siêu phẳng tối ưu để phân loại dữ liệu.

**- K-means Clustering:** Phân cụm dữ liệu thành k nhóm dựa trên khoảng cách giữa các điểm dữ liệu.

**2.2.5. Các trường hợp sử dụng máy học trong thế giới thực**

Học máy (Machine Learning) đã vượt ra khỏi phạm vi nghiên cứu lý thuyết và trở thành một công nghệ cốt lõi trong nhiều ứng dụng thực tế. Từ những tiện ích hàng ngày đến những đột phá khoa học, học máy đang thay đổi cách chúng ta sống và làm việc. Dưới đây là một số ví dụ cụ thể:

+ Thị giác máy tính:

Nhận dạng hình ảnh: Học máy cho phép máy tính "nhìn" và hiểu hình ảnh, tương tự như con người. Công nghệ này được ứng dụng rộng rãi trong tìm kiếm hình ảnh theo nội dung, nhận dạng khuôn mặt để xác thực danh tính, phân loại hình ảnh tự động, và thậm chí tạo ra hình ảnh mới. Ví dụ, Google Photos sử dụng học máy để tự động gắn thẻ và phân loại ảnh theo người, địa điểm và sự vật.

Xe tự lái: Thị giác máy tính là nền tảng cho xe tự lái, giúp xe "nhìn thấy" môi trường xung quanh, nhận diện người đi bộ, phương tiện khác, biển báo giao thông và đưa ra quyết định lái xe an toàn. Các công ty như Tesla, Waymo đang tiên phong trong việc ứng dụng học máy cho xe tự lái.

Chẩn đoán y tế: Học máy có thể hỗ trợ bác sĩ phân tích hình ảnh y tế như X-quang, MRI, CT scan để phát hiện các dấu hiệu bệnh lý khó nhận thấy bằng mắt thường, từ đó nâng cao độ chính xác và tốc độ chẩn đoán. Một số ứng dụng bao gồm phát hiện khối u, bệnh tim mạch và các bệnh về mắt.

Kiểm soát chất lượng: Trong sản xuất, học máy được sử dụng để tự động kiểm tra chất lượng sản phẩm, phát hiện các khuyết tật nhỏ và đảm bảo tính đồng nhất của sản phẩm, giúp tiết kiệm thời gian và chi phí.

+ Xử lý ngôn ngữ tự nhiên (NLP):

Trợ lý ảo: Các trợ lý ảo như Siri, Alexa, Google Assistant sử dụng NLP để hiểu và phản hồi lại lời nói của người dùng. Chúng có thể thực hiện các tác vụ như đặt lịch hẹn, gửi tin nhắn, phát nhạc, tìm kiếm thông tin và điều khiển các thiết bị thông minh trong nhà.

Chatbot: Chatbot, được hỗ trợ bởi NLP, được sử dụng rộng rãi trong dịch vụ khách hàng, hỗ trợ kỹ thuật và bán hàng trực tuyến. Chúng có thể trả lời các câu hỏi thường gặp, hướng dẫn khách hàng và giải quyết các vấn đề đơn giản, giúp giảm tải cho nhân viên hỗ trợ.

Dịch máy: Các công cụ dịch máy như Google Translate và DeepLY sử dụng học máy để cải thiện chất lượng dịch thuật, giúp người dùng dễ dàng hiểu nội dung bằng các ngôn ngữ khác nhau.

Phân tích cảm xúc: Học máy có thể phân tích văn bản để xác định cảm xúc của người viết (tích cực, tiêu cực, trung lập). Điều này hữu ích cho việc theo dõi dư luận xã hội, đánh giá phản hồi của khách hàng và nghiên cứu thị trường.

+ Dự đoán:

Dự báo tài chính: Học máy được sử dụng trong lĩnh vực tài chính để dự đoán giá cổ phiếu, biến động thị trường, đánh giá rủi ro đầu tư và đưa ra quyết định giao dịch.

Dự đoán nhu cầu: Các doanh nghiệp sử dụng học máy để dự đoán nhu cầu sản phẩm, tối ưu hóa quản lý kho hàng và chuỗi cung ứng, giảm thiểu lãng phí và tăng hiệu quả hoạt động.

Dự báo thời tiết: Học máy kết hợp với dữ liệu khí tượng để xây dựng các mô hình dự báo thời tiết chính xác hơn, giúp người dân chuẩn bị tốt hơn cho các hiện tượng thời tiết khắc nghiệt.

+ Khuyến nghị:

Khuyến nghị sản phẩm: Các nền tảng thương mại điện tử như Amazon, Tiki sử dụng học máy để đề xuất sản phẩm phù hợp với sở thích và hành vi mua sắm của từng khách hàng, từ đó tăng doanh số bán hàng và cải thiện trải nghiệm người dùng.

Khuyến nghị nội dung: Các dịch vụ giải trí như Netflix, Spotify, YouTube sử dụng học máy để đề xuất phim, nhạc, video mà người dùng có thể quan tâm, dựa trên lịch sử xem và nghe của họ.

+ Các lĩnh vực khác:

Phát hiện gian lận: Ngân hàng và các tổ chức tài chính sử dụng học máy để phát hiện các giao dịch gian lận, bảo vệ tài khoản của khách hàng và ngăn chặn các hoạt động rửa tiền.

Chăm sóc sức khỏe cá nhân hóa: Học máy có thể phân tích dữ liệu sức khỏe cá nhân để đưa ra khuyến nghị về chế độ dinh dưỡng, luyện tập và thậm chí dự đoán nguy cơ mắc bệnh, giúp người dùng chủ động chăm sóc sức khỏe tốt hơn.

Nông nghiệp thông minh: Học máy được ứng dụng trong nông nghiệp để tối ưu hóa việc sử dụng nước, phân bón, thuốc trừ sâu, theo dõi sức khỏe cây trồng và dự đoán năng suất, góp phần nâng cao hiệu quả sản xuất nông nghiệp.

Học máy đang ngày càng phát triển và được ứng dụng rộng rãi hơn trong cuộc sống. Với sự tiến bộ của công nghệ và lượng dữ liệu ngày càng tăng, học máy hứa hẹn sẽ mang lại nhiều đột phá hơn nữa trong tương lai.

## 2.3. Ngôn ngữ lập trình và ứng dụng sử dụng

## 2.3.1.Visual code

Visual Code (tên đầy đủ là Visual Studio Code, thường được viết tắt là VS Code) là một trình soạn thảo mã nguồn mở (source code editor) miễn phí, do Microsoft phát triển. Đây là công cụ phổ biến dành cho lập trình viên, hỗ trợ viết và phát triển mã nguồn trong nhiều ngôn ngữ lập trình khác nhau như JavaScript, Python, Java, C++, PHP, HTML, CSS, và nhiều ngôn ngữ khác.



- Các tính năng nổi bật của Visual Studio Code:

+ Hỗ trợ đa ngôn ngữ:  
VS Code hỗ trợ rất nhiều ngôn ngữ lập trình ngay từ lúc cài đặt. Ngoài ra, bạn có thể cài đặt các extension (tiện ích mở rộng) để hỗ trợ thêm ngôn ngữ khác.

+ Giao diện trực quan và dễ tùy chỉnh:  
Người dùng có thể thay đổi giao diện (themes), bố cục, hoặc phím tắt theo nhu cầu.

+ Tích hợp Git:  
Visual Studio Code cho phép bạn làm việc với Git ngay trong ứng dụng, bao gồm commit, pull, push và quản lý nhánh (branch).

+ Hệ thống tiện ích mở rộng (Extensions):  
Người dùng có thể cài đặt rất nhiều extension từ Marketplace để bổ sung tính năng như:

Trình gỡ lỗi (debugging).

Hỗ trợ cú pháp ngôn ngữ.

Trình định dạng code (code formatter).

Công cụ kiểm tra lỗi (linters).

+ Tích hợp Terminal:  
VS Code có một terminal tích hợp giúp bạn chạy lệnh mà không cần chuyển đổi giữa các cửa sổ.

+ Tính năng IntelliSense:  
Hỗ trợ gợi ý mã (code suggestion), tự động hoàn thành cú pháp (auto-completion), và kiểm tra lỗi khi lập trình.

+ Debugging mạnh mẽ:  
Hỗ trợ gỡ lỗi ngay trong môi trường lập trình mà không cần các công cụ bên ngoài.

- Ưu điểm của Visual Studio Code:

Miễn phí và mã nguồn mở.

Nhẹ, khởi động nhanh, phù hợp với nhiều hệ điều hành (Windows, macOS, Linux).

Cộng đồng hỗ trợ rộng lớn với nhiều tài liệu và hướng dẫn.

Nhược điểm:

Một số tính năng nâng cao yêu cầu cấu hình hoặc cài đặt thêm extension.

Hiệu suất có thể bị ảnh hưởng khi mở các dự án lớn hoặc cài quá nhiều tiện ích mở rộng.

Tại sao Visual Studio Code được yêu thích?

Với khả năng tùy biến cao, giao diện thân thiện, và cộng đồng hỗ trợ đông đảo, VS Code là công cụ lý tưởng cho cả người mới bắt đầu lẫn lập trình viên chuyên nghiệp.

## 2.3.2. Python

Python đã trở thành ngôn ngữ lập trình thống trị trong lĩnh vực học máy, nhờ vào sự đơn giản, dễ học, tính linh hoạt và hệ sinh thái phong phú các thư viện hỗ trợ. Việc lựa chọn Python giúp đơn giản hóa quá trình phát triển và triển khai các mô hình học máy.



**- Ưu điểm của Python trong Học Máy:**

Cú pháp đơn giản, dễ đọc: Python nổi tiếng với cú pháp rõ ràng, gần gũi với ngôn ngữ tự nhiên, giúp giảm thiểu thời gian học và dễ dàng đọc, hiểu, bảo trì mã nguồn. Điều này đặc biệt hữu ích cho những người mới bắt đầu học lập trình.

Hệ sinh thái thư viện mạnh mẽ: Python sở hữu một bộ sưu tập khổng lồ các thư viện chuyên dụng cho học máy, cung cấp các công cụ mạnh mẽ cho mọi giai đoạn của quy trình phát triển mô hình:

NumPy: Cung cấp các cấu trúc dữ liệu mảng hiệu suất cao và các hàm toán học cần thiết cho tính toán khoa học.

Pandas: Cho phép thao tác và phân tích dữ liệu một cách hiệu quả với DataFrame, giúp dễ dàng xử lý và chuẩn bị dữ liệu cho học máy.

Scikit-learn: Thư viện toàn diện cho các thuật toán học máy cổ điển, bao gồm phân loại, hồi quy, phân cụm, lựa chọn đặc trưng và nhiều hơn nữa. Scikit-learn cung cấp một API nhất quán và dễ sử dụng.

TensorFlow vs PyTorch: Hai framework hàng đầu cho học sâu, hỗ trợ xây dựng và huấn luyện các mạng nơ-ron phức tạp. Cả hai đều cung cấp khả năng tính toán trên GPU để tăng tốc độ huấn luyện.

Keras: API cấp cao, thân thiện với người dùng, chạy trên TensorFlow, giúp đơn giản hóa việc phát triển mô hình học sâu.

Matplotlib và Seaborn: Cho phép trực quan hóa dữ liệu với các biểu đồ và đồ thị chất lượng cao, giúp dễ dàng phân tích và diễn giải kết quả.

Tính linh hoạt và khả năng mở rộng: Python có thể dễ dàng tích hợp với các hệ thống và công nghệ khác, cho phép triển khai mô hình học máy trong nhiều môi trường khác nhau, từ máy tính cá nhân đến các hệ thống đám mây.

Cộng đồng lớn mạnh: Python có một cộng đồng người dùng đông đảo và hoạt động tích cực, sẵn sàng hỗ trợ và chia sẻ kiến thức. Điều này giúp người dùng dễ dàng tìm kiếm tài liệu, hướng dẫn và giải đáp thắc mắc.

**- Môi trường phát triển Python cho Học Máy:**

Anaconda: Bộ phân phối Python phổ biến cho khoa học dữ liệu và học máy, bao gồm nhiều thư viện cần thiết (NumPy, Pandas, Scikit-learn,…) và các công cụ quản lý môi trường (conda). Anaconda giúp đơn giản hóa việc cài đặt và quản lý các thư viện Python.

Google Colab: Môi trường đám mây miễn phí cho phép chạy mã Python trực tiếp trên trình duyệt, cung cấp tài nguyên tính toán mạnh mẽ (GPU, TPU) và dễ dàng chia sẻ. Google Colab là một lựa chọn tuyệt vời cho việc học tập và thử nghiệm các mô hình học máy.

Jupyter Notebook/Lab: Cho phép tạo và chia sẻ các tài liệu tương tác, kết hợp mã Python, văn bản, hình ảnh và phương trình toán học. Jupyter Notebook/Lab rất hữu ích cho việc ghi chép, trình bày và chia sẻ kết quả nghiên cứu.

# **CHƯƠNG 2: XÂY DỰNG HỆ THỐNG**

## 2.1 Bài toán

## 2.1.1 Đặt vấn đề

Trong các cơ quan, tổ chức, và doanh nghiệp, rất nhiều tài liệu giấy được viết tay hoặc ghi chú bởi nhân viên. Thay vì phải nhập lại toàn bộ nội dung từ giấy vào hệ thống máy tính, hệ thống nhận diện chữ viết tay có thể tự động quét và nhận dạng chữ viết để chuyển nó thành văn bản. Bài toán đặt ra là tự động trích xuất thông tin từ tài liệu viết tay thông qua hình ảnh sau đó hiển thị lên giao diện.

## 2.1.2 Mô tả bài toán

Xây dựng một hệ thống nhận diện chữ viết tay từ ảnh đầu vào, bao gồm các bước từ xử lý ảnh, phát hiện vùng chữ viết, nhận diện ký tự và hiển thị kết quả dưới dạng văn bản

**Đầu vào**

Một tệp ảnh chứa chữ viết tay (định dạng: .png, .jpg, hoặc .bmp).

Ảnh có thể chứa một hoặc nhiều dòng chữ viết tay với kích thước không cố định.

**Đầu ra**

Văn bản nhận diện được từ ảnh (chuỗi ký tự).

Hiển thị bounding boxes (nếu cần) đánh dấu các vùng chữ viết được phát hiện trên ảnh.

**Các bước xử lý trong hệ thống**

**1. Tiền xử lý ảnh**

Mục tiêu: Chuẩn bị ảnh cho các bước tiếp theo.

Thao tác:

Đọc ảnh từ tệp đầu vào.

Chuyển đổi ảnh về dạng grayscale.

Resize ảnh về kích thước chuẩn (ví dụ: 128x32).

Chuẩn hóa giá trị pixel về khoảng [0, 1].

**2. Phát hiện và phân cụm vùng chữ viết**

Mục tiêu: Xác định các vùng trong ảnh chứa ký tự hoặc từ.

Thao tác:

Áp dụng thuật toán phát hiện bounding boxes để xác định vị trí các ký tự hoặc cụm từ.

Phân cụm các bounding boxes gần nhau thành các nhóm chữ liên tiếp (sử dụng DBSCAN hoặc phương pháp tương tự).

**3. Nhận diện ký tự**

Mục tiêu: Chuyển đổi vùng bounding boxes thành chuỗi ký tự.

Thao tác:

Trích xuất đặc trưng từ các vùng bounding boxes.

Áp dụng mô hình nhận diện ký tự (dựa trên phương pháp giải mã chuỗi, như CTC Loss).

**4. Kết quả**

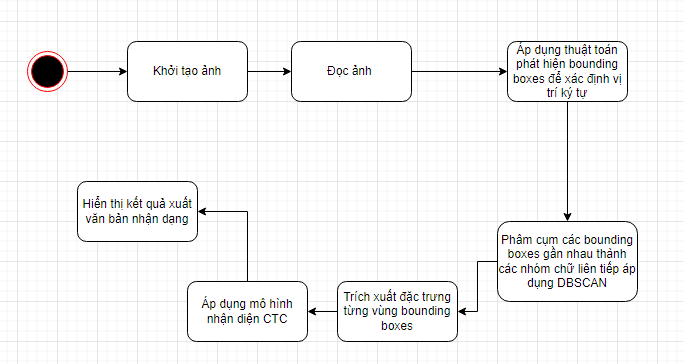
Mục tiêu: Tạo văn bản hoàn chỉnh từ các ký tự đã nhận diện và hiển thị kết quả.

Thao tác:

Giải mã chuỗi ký tự từ bounding boxes thành từ và câu.

Xuất kết quả nhận diện ra màn hình.

Hiển thị ảnh với bounding boxes đánh dấu vùng chữ viết (tùy chọn).



Hình 1: Mô hình hoạt động của hệ thống

## 2.2. Xây dựng hệ thống

## 2.2.1 Các thư viện

**Thư viện json:**

**Mục đích**: Làm việc với dữ liệu ở định dạng JSON (JavaScript Object Notation)

**Thư viện cv2:** OpenCV (Open Source Computer Vision Library).

**Mục đích**: Xử lý và phân tích hình ảnh, video.

**Thư viện: Matplotlib.**

**Mục đích**: Hiển thị và trực quan hóa dữ liệu dưới dạng biểu đồ, đồ thị, hoặc hình ảnh.

**Thư viện: Gradio.**

**Mục đích**: Xây dựng giao diện người dùng (UI) trực quan để tương tác với các mô hình học máy hoặc ứng dụng Python.

**Thư viện:** **ONNXRuntime.**

**Mục đích**: Chạy các mô hình học máy đã được chuyển đổi sang định dạng ONNX (Open Neural Network Exchange) trên nhiều nền tảng.

**Thư viện: NumPy.**

**Mục đích**: Xử lý số học hiệu năng cao với các mảng (arrays) và ma trận (matrices).

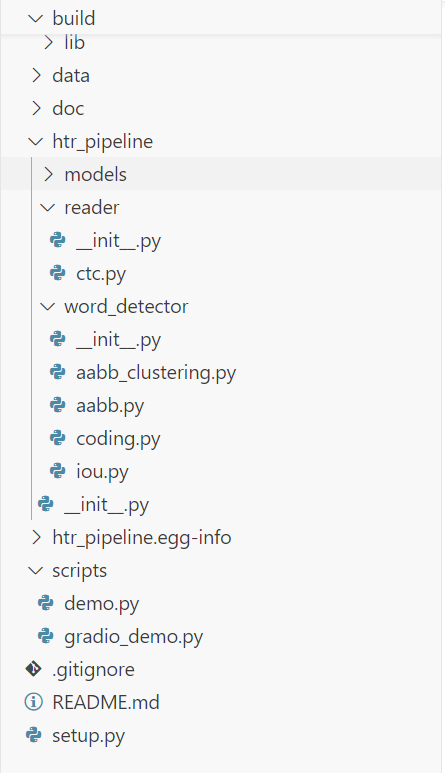
- Xử lý dữ liệu : **numpy**, **math**

Tính toán ma trận, xử lý bounding box

- Phân cụm: **sklearn.cluster**

Phân nhóm bounding box thành các cụm

**Các file module**

****

Hình 2: file module đang sử dụng

**Module** htr\_pipeline:

\_\_init\_\_.py: File khởi tạo module chính.

**Module** htr\_pipeline.models:

\_\_init\_\_.py: File khởi tạo module.

detector.onnx: Mô hình học máy.

reader.onnx: Mô hình học máy.

reader.json: File cấu hình mô hình.

**Module** htr\_pipeline.reader:

\_\_init\_\_.py: File khởi tạo module.

ctc.py: Xử lý mã hóa CTC (Connectionist Temporal Classification).

**Module** htr\_pipeline.word\_detector:

\_\_init\_\_.py: File khởi tạo module.

aabb.py: Xử lý các hộp bao quanh (Axis-Aligned Bounding Box).

aabb\_clustering.py: Phân cụm AABB.

coding.py: Xử lý mã hóa.

iou.py: Tính toán chỉ số giao nhau trên hợp (Intersection over Union - IoU).

## 2.3. Một số mã nguồn quan trọng

**Mã nguồn CTC:**

- Beam search mở rộng các chuỗi dự đoán bằng cách tính xác suất tổng hợp giữa nhãn trống và các ký tự đầu ra.

|  |
| --- |
| def ctc\_single\_word\_beam\_search(predictions: np.ndarray,  chars: List[str],  beam\_width: int,  prefix\_tree: PrefixTree):  res = []  for batch\_idx in range(predictions.shape[1]):  num\_timesteps = predictions.shape[0]  prev = [Beam('', 1, 0)]  # go over all time-steps  for time\_idx in range(num\_timesteps):  curr = []  # list of beams at current time-step  # go over best beams  best\_beams = sorted(prev, key=lambda x: x.prob\_total, reverse=True)[:beam\_width]  # get best beams  for beam in best\_beams:  # calc probability that beam ends with non-blank  pr\_non\_blank = 0  if beam.text != '':  # char at time-step t must also occur at t-1  label\_idx = chars.index(beam.text[-1]) + 1  pr\_non\_blank = beam.prod\_non\_blank \* predictions[time\_idx, batch\_idx, label\_idx]  # calc probability that beam ends with blank  pr\_blank = beam.prob\_total \* predictions[time\_idx, batch\_idx, 0]  # save result  curr.append(Beam(beam.text, pr\_blank, pr\_non\_blank))  # extend current beam with characters according to language model  # nextChars = beam.getNextChars()  next\_chars = prefix\_tree.get\_next\_chars(beam.text)  for c in next\_chars:  # extend current beam with new character  label\_idx = chars.index(c) + 1  if beam.text != '' and beam.text[-1] == c:  # same chars must be separated by blank  pr\_non\_blank = predictions[time\_idx, batch\_idx, label\_idx] \* beam.prob\_blank  else:  # different chars can be neighbours  pr\_non\_blank = predictions[time\_idx, batch\_idx, label\_idx] \* beam.prob\_total  # save result  curr.append(Beam(beam.text + c, 0, pr\_non\_blank))  # move current beams to next time-step  prev = curr  # return most probable beam  prev = [beam for beam in prev if prefix\_tree.is\_word(beam.text)]  # only keep words  best\_beams = sorted(prev, key=lambda x: x.prob\_total, reverse=True)  res.append(best\_beams[0].text if best\_beams else '')  return res |

- Phương pháp greedy decoding giải mã trực tiếp dựa trên đường dẫn tốt nhất (best path).

|  |
| --- |
| def ctc\_best\_path(predictions: np.ndarray, chars: List[str]) -> List[str]:      # shape of predictions: WxBxC      res = []      for b in range(predictions.shape[1]):          # get char indices along best path          best\_path = np.argmax(predictions[:, b], axis=1)          # collapse best path (using itertools.groupby), map to chars, join char list to string          best\_path\_decoded = [chars[c - 1] for c, \_ in groupby(best\_path) if c != 0]          text = ''.join(best\_path\_decoded)          res.append(text)      return res |

- Nhận diện văn bản trong hình ảnh

|  |
| --- |
| def read(img: np.ndarray, decoder: str, prefix\_tree: Optional[PrefixTree] = None) -> str:      """Nhận diện văn bản trong hình ảnh."""      img = transform(img)      img = img[None, None].astype(np.float32)      outputs = \_ORT\_SESSION.run(None, {'input': img})      if decoder == 'best\_path':          text = ctc\_best\_path(outputs[0], \_CHARS)[0]      elif decoder == 'word\_beam\_search':          text = ctc\_single\_word\_beam\_search(outputs[0], \_CHARS, 25, prefix\_tree)[0]      else:          raise Exception('Unknown decoder. Available: "best\_path" and "word\_beam\_search".')      return text |

# **CHƯƠNG 3: KẾT QUẢ THỰC NGHIỆM**

## ****3.1. Dữ liệu****

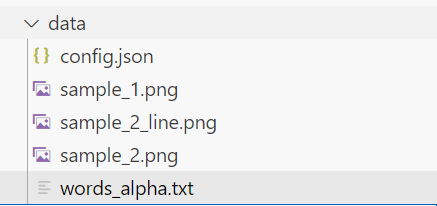
**Dữ liệu đầu vào** sẽ được đọc từ camera theo thời gian thực, sau khi văn bản viết tay được đưa vào khay và camera hiển thị ngay trên giao diện. Sau đó, hệ thống sẽ cắt dữ liệu theo từng khung cố định, dựa trên khu vực chứa văn bản viết tay được đặt sẵn từ đầu.

**Xử lý ảnh trước nhận diện:** Sau khi cắt được thông tin, hệ thống sẽ xử lý ảnh bằng OpenCV để chuyển sang ảnh xám, áp dụng phân ngưỡng nhằm làm nổi bật nét chữ viết tay. Các bước xử lý có thể bao gồm loại bỏ nhiễu, làm mịn hoặc tăng cường tương phản để đảm bảo ký tự viết tay rõ ràng hơn.

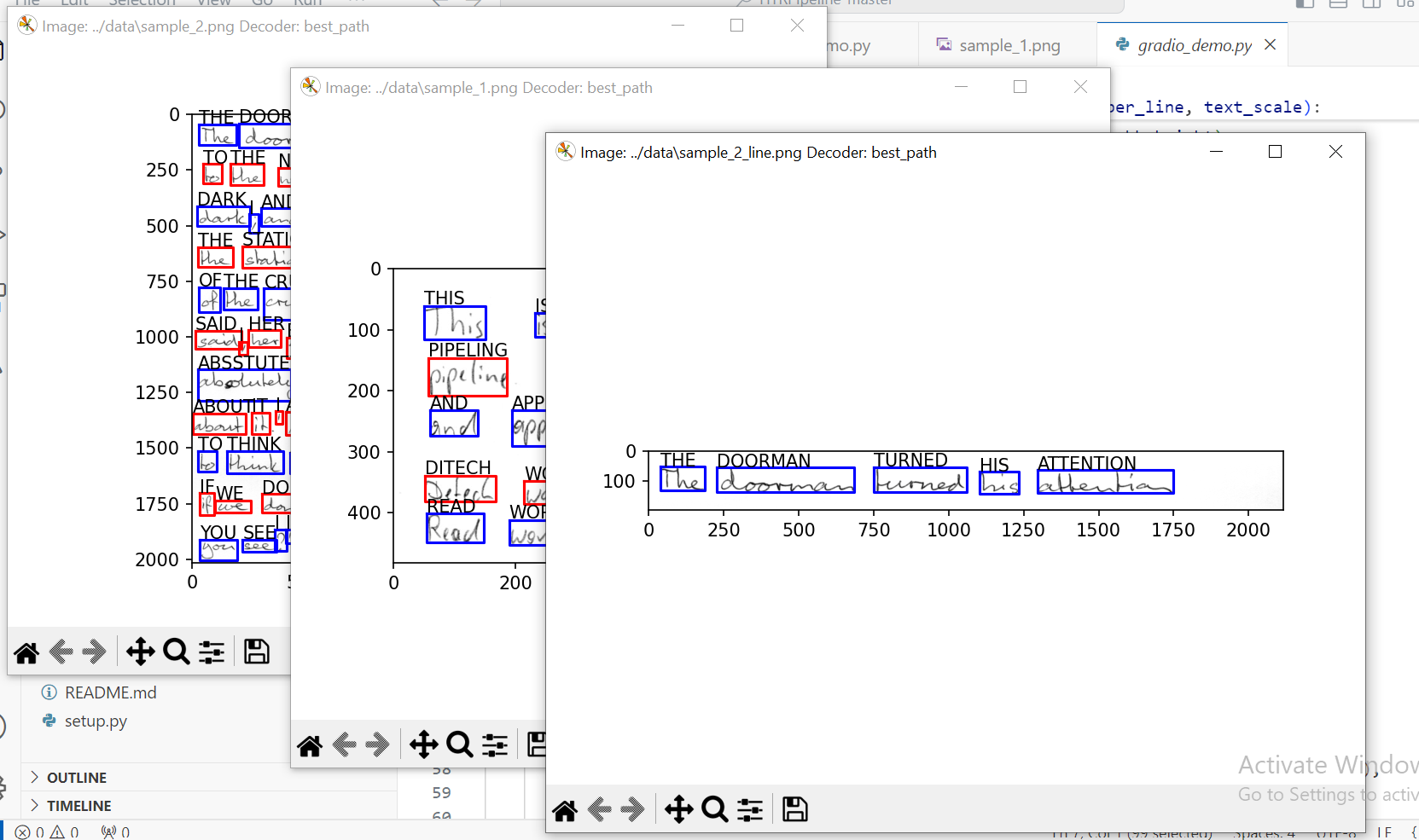
**Nhận diện chữ viết tay:** Ảnh sau khi được xử lý sẽ được đưa vào công cụ OCR như Tesseract hoặc một mô hình nhận diện chữ viết tay hiện đại (ví dụ: các mô hình dựa trên deep learning như CRNN hoặc mô hình Transformer). Hệ thống sau đó chuyển dữ liệu nhận diện thành văn bản số và hiển thị lên màn hình.

## ****3.2. Kết quả thực nghiệm****

## ****3.2.1. Chạy tự động ảnh****

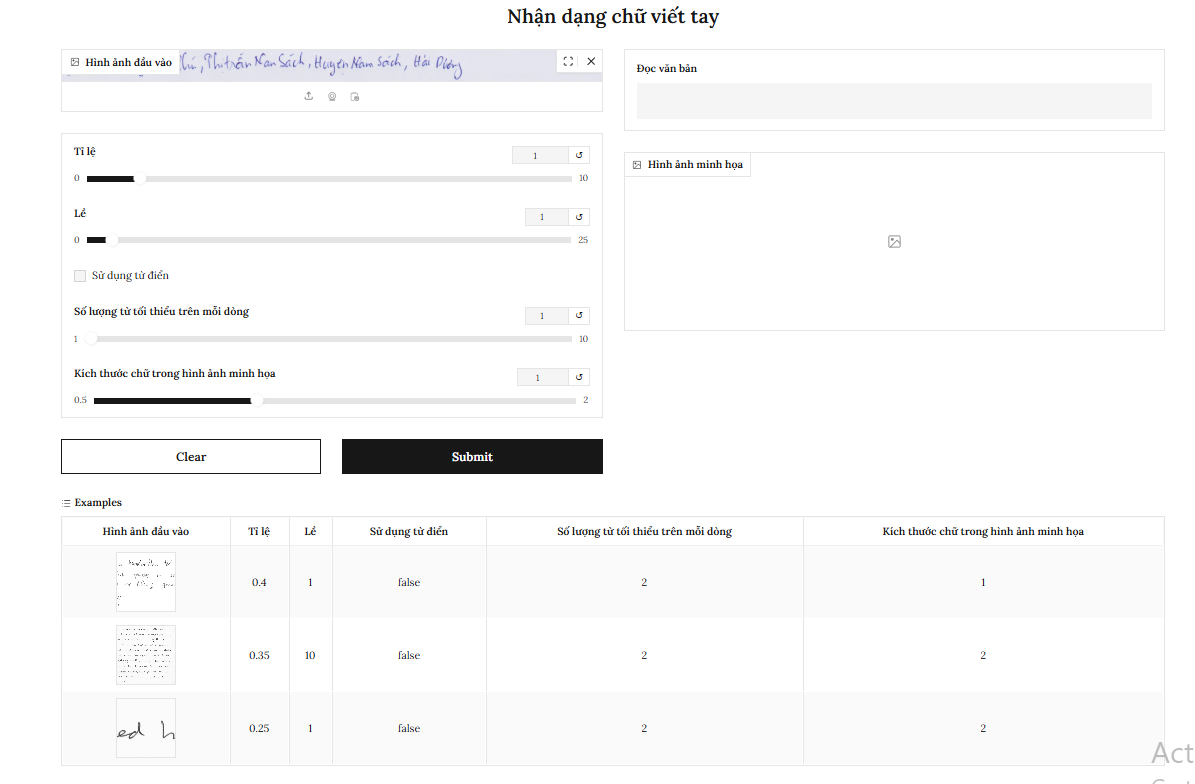
****

**Hình 3: File ảnh chạy tự động**

****

**Hình 4: Kết quả thực nghiệm chạy chương trình ảnh có sẵn**

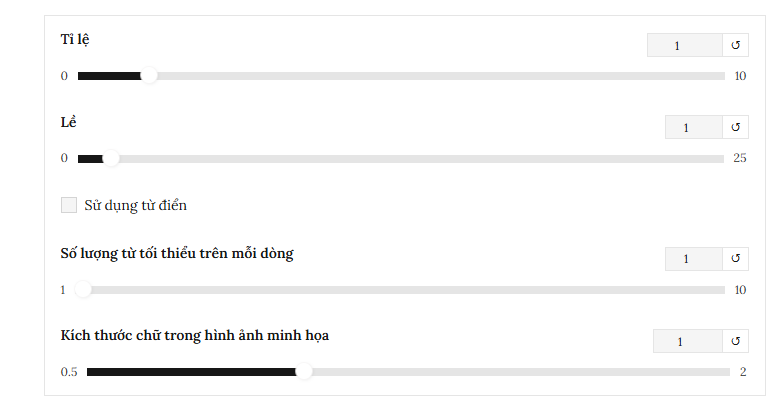
## ****3.2.2. Nhận diện chữ viết tay thủ công****

****

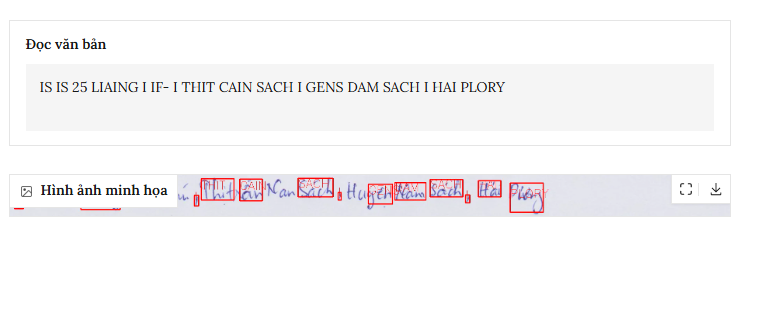
**Hình 5: Tổng thể nhận diện chữ viết**

****

**Hình 6: Hình ảnh đầu vào**

****

**Hình 7: Chỉnh tỷ lệ ảnh lấy ra**

****

**Hình 8: Hiện hình ảnh và đọc văn bản**

****

**Hình 9: Hiển thị cách file đã lưu chạy trong data**

# **KẾT LUẬN**

Hệ thống nhận diện chữ số viết tay không chỉ là một bài toán quan trọng trong lĩnh vực trí tuệ nhân tạo mà còn mang lại ý nghĩa to lớn trong việc giải quyết các vấn đề thực tiễn trong nhiều lĩnh vực như ngân hàng, giáo dục, và quản lý dữ liệu. Với sự kết hợp giữa các phương pháp xử lý ảnh hiện đại và các mô hình học sâu tiên tiến như mạng nơ-ron tích chập (CNN), hệ thống có khả năng trích xuất đặc trưng, phân loại chính xác và đưa ra kết quả dự đoán nhanh chóng. Tuy nhiên, để đạt hiệu suất cao, hệ thống cần được xây dựng trên nền tảng dữ liệu chất lượng, được xử lý và tăng cường cẩn thận để giải quyết những thách thức như sự đa dạng trong cách viết tay, nhiễu từ môi trường, hoặc sự không đồng nhất trong dữ liệu thực tế.

Dù vậy, sự phát triển không ngừng của công nghệ trí tuệ nhân tạo và các phương pháp học sâu đã giúp khắc phục nhiều hạn chế, cho phép các hệ thống nhận diện ngày càng đạt độ chính xác cao và khả năng ứng dụng rộng rãi. Từ việc tự động hóa quy trình xử lý séc trong ngân hàng, chấm điểm bài thi viết tay trong giáo dục, đến tối ưu hóa nhập liệu trong các hệ thống số hóa, các ứng dụng của hệ thống này đã và đang làm thay đổi cách chúng ta tương tác với dữ liệu viết tay.

Tóm lại, việc nghiên cứu và triển khai hệ thống nhận diện chữ số viết tay không chỉ góp phần thúc đẩy quá trình tự động hóa và số hóa, mà còn mở ra những tiềm năng to lớn trong việc ứng dụng trí tuệ nhân tạo vào các bài toán thực tế phức tạp, hứa hẹn sẽ tiếp tục phát triển và đóng góp nhiều hơn cho cuộc sống hiện đại trong tương lai.