**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 06: Xây dựng hệ thống nhận diện chữ số viết tay**

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

|  |  |  |  |
| --- | --- | --- | --- |
| **TT** | **Mã sinh viên** | **Sinh viên thực hiện** | **Lớp hành chính** |
| **1** | **20210779** | **Vi Giang Huy** | **DCCNTT12.10.03** |
| **2** | **20210743** | **Trần Tăng Đức** | **DCCNTT12.10.03** |
| **3** | **20210730** | **Thái Bảo Tuấn** | **DCCNTT12.10.03** |
| **4** | **20210731** | **Nguyễn Anh Tuấn** | **DCCNTT12.10.03** |
| **5** | **20210877** | **Trịnh Minh Đức** | **DCCNTT12.10.03** |

**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 06: Xây dựng hệ thống nhận diện chữ số viết tay**

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

|  |  |  |  |
| --- | --- | --- | --- |
| **TT** | **Mã sinh viên** | **Sinh viên thực hiện** | **Lớp hành chính** |
| **1** | **20210779** | **Vi Giang Huy** | DCCNTT12.10.03 |
| **2** | **20210743** | **Trần Tăng Đức** | DCCNTT12.10.03 |
| **3** | **20210730** | **Thái Bảo Tuấn** | DCCNTT12.10.03 |
| **4** | **20210731** | **Nguyễn Anh Tuấn** | DCCNTT12.10.03 |
| **5** | **20210877** | **Trịnh Minh Đức** | DCCNTT12.10.03 |

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

**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.K12.03.LH.C04.1\_LT** | |
| **TS.Lương Thị Hồng Lan**  *(Ký và ghi rõ họ tên)* | **Cán bộ chấm thi 2**  *(Ký và ghi rõ họ tên)* |

| **TT** | **TIÊU CHÍ** | **THANG ĐIỂM** | **Vi Giang Huy** | **Trần Tăng Đức** | **Thái Bảo Tuấn** | **Nguyễn Anh Tuấn** | **Trịnh Minh Đức** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 202107  79 | 202107  43 | 202107  30 | 20210731 | 20210877 |
| **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* |  |  |  |  |  |

# **LỜI NÓI ĐẦU**

Trong thời đại công nghệ phát triển mạnh mẽ, nhận dạng chữ số viết tay là một bài toán quan trọng với nhiều ứng dụng thực tiễn như số hóa tài liệu, xử lý thông tin tự động và giáo dục. Đề tài "Xây dựng hệ thống nhận diện chữ số viết tay" được chúng em lựa chọn nhằm nghiên cứu và ứng dụng các kỹ thuật học máy và xử lý ảnh hiện đại, đặc biệt là mạng nơ-ron tích chập (CNN), để tạo ra một hệ thống nhận diện hiệu quả và chính xác.Sau một khoảng thời gian học tập cũng như làm việc chúng em cũng đã hoàn thành xong bài làm của mình.

Chúng em xin gửi lời cảm ơn chân thành đến các thầy cô khoa Công nghệ thông tin, trường Đại học Công nghệ Đông Á, đặc biệt là cô Lương Thị Hồng Lan, đã hỗ trợ chúng em trong suốt quá trình học và thực hiện đề tài. Mặc dù đã cố gắng hoàn thành tốt nhất, nhưng chắc chắn chúng em vẫn có thiếu sốt rất nhiều. Chúng em mong nhận được ý kiến đóng góp của cô để cải thiện và hoàn thiện hơn, chúng em xin cảm ơn.

MỤC LỤC

[LỜI NÓI ĐẦU i](#_Toc184676325)

[MỤC LỤC ii](#_Toc184676326)

[DANH MỤC HÌNH ẢNH iv](#_Toc184676327)

[DANH MỤC CÁC TỪ VIẾT TẮT v](#_Toc184676328)

[CHƯƠNG I: CƠ SỞ LÝ THUYẾT 6](#_Toc184676329)

[1.1. Bài toán nhận dạng 6](#_Toc184676330)

[1.2. Các kỹ thuật sử dụng trong bài toán nhận dạng 8](#_Toc184676331)

[1.2.1. Kỹ thuật CNN (Convolutional Neural Network) 8](#_Toc184676332)

[1.2.2 Kỹ thuật KNN (K-Nearest Neighbors) 8](#_Toc184676333)

[1.2.3 Kỹ thuật HOG (Histogram of Oriented Gradients) 9](#_Toc184676334)

[1.2.4 Kỹ thuật R-CNN (Region-based Convolutional Neural Network) 9](#_Toc184676335)

[1.3. Ngôn ngữ lập trình và các thư viện sử dụng 10](#_Toc184676336)

[1.3.1. Python 10](#_Toc184676337)

[1.3.2. Các thư viện sử dụng 10](#_Toc184676338)

[CHƯƠNG II: XÂY DỰNG HỆ THỐNG 13](#_Toc184676339)

[2.1 Mô tả bài toán 13](#_Toc184676340)

[2.2 Xây dựng hệ thống 13](#_Toc184676341)

[2.2.1 Giai đoạn tiền xử lý ảnh 15](#_Toc184676342)

[2.2.2 Chuyển đổi ảnh sang ảnh xám (Grayscale Conversion) 15](#_Toc184676343)

[2.1.3 Làm mịn 16](#_Toc184676344)

[2.1.4 Nhị phân hóa ảnh 17](#_Toc184676345)

[2.1.5 Chuẩn hóa kích thước hình ảnh 17](#_Toc184676346)

[2.1.6 Mô hình CNN. 18](#_Toc184676347)

[2.1.7 Giai đoạn Quản lý dữ liệu 21](#_Toc184676348)

[2.1.8 Giai đoạn Huấn luyện mô hình 22](#_Toc184676349)

[2.1.9 Giai đoạn Nhận dạng 24](#_Toc184676350)

[CHƯƠNG III: THỰC NGHIỆM 26](#_Toc184676351)

[3.1 Dữ liệu 26](#_Toc184676352)

[3.1.1 Mô tả dữ liệu đầu vào 26](#_Toc184676353)

[3.1.2 Kỹ thuật xử lý tiền dữ liệu 27](#_Toc184676354)

[3.1.3 Chia train-test 28](#_Toc184676355)

[3.2 Độ đo đánh giá 29](#_Toc184676356)

[3.3 Kết quả thực nghiệm 31](#_Toc184676357)

[KẾT LUẬN 39](#_Toc184676358)

[TÀI LIỆU THAM KHẢO 43](#_Toc184676359)

# **DANH MỤC HÌNH ẢNH**

[Hình 1.1 Hình ảnh tập dữ liệu MNIST chữ số viết tay 6](#_Toc184472655)

[Hình 2.1 Sơ đồ tổng quát nhận dang 13](#_Toc184472656)

[Dưới đây sẽ cụ thể hơn các giai đoạn đó và các kỹ thuật sử dụng trọng các giai đoạn 13](#_Toc184472657)

[Hình 3.1 Tệp file dữ liệu đầu vào 23](#_Toc184472658)

[Hình 3.3 Hàm tính độ đo 27](#_Toc184472659)

[Hình 3.5 Chức năng số 1 29](#_Toc184472660)

[Hình 3.6 Thu thập dữ liệu bằng paint 30](#_Toc184472661)

[Hình 3.7 Tệp file lưu dữ liệu 31](#_Toc184472662)

[Hình 3.9 Tệp file lưu dataset 32](#_Toc184472663)

[Hình 3.10 Tệp file model 32](#_Toc184472664)

[Hình 3.11 Đo độ chính xác 33](#_Toc184472665)

[Hình 3.13 Dùng thử chức năng dự đoán số 2 35](#_Toc184472666)

# **DANH MỤC CÁC TỪ VIẾT TẮT**

|  |  |  |
| --- | --- | --- |
| STT | TỪ VIẾT TẮT | GIẢI THÍCH |
| 1 | CNN | Convolutional Neural Network |
| 2 | SVM | Support Vector Machine |
| 3 | KNN | K-Nearest Neighbors |
| 4 | CNTT | Công nghệ thông tin |
| 5 | DR | Decryption Rounds |
| 6 | EAUT | East Asia University Technology |
| 7 | RSA | Rivest Shamir Adleman |

# **CHƯƠNG I: CƠ SỞ LÝ THUYẾT**

## **1.1. Bài toán nhận dạng**

Nhận dạng chữ số viết tay [1] là một nhánh quan trọng của bài toán nhận dạng mẫu trong lĩnh vực xử lý ảnh và học máy. Mục tiêu chính của bài toán là phân tích và xử lý một hình ảnh chứa các chữ số viết tay, sau đó xác định chính xác giá trị của từng chữ số. Đây không chỉ là bài toán về mặt kỹ thuật mà còn có ý nghĩa thực tiễn sâu sắc, vì các ứng dụng của nó đã và đang thay đổi nhiều lĩnh vực như xử lý tài liệu, số hóa dữ liệu, giáo dục, và tự động hóa.

Ví dụ điển hình cho bài toán này có thể kể đến hệ thống nhận dạng mã số bưu kiện trong lĩnh vực logistics, đọc và xử lý hóa đơn hoặc phiếu ghi tay trong kế toán, hay hỗ trợ người khiếm thị qua việc chuyển đổi chữ số viết tay sang dạng âm thanh. Một ứng dụng nổi bật khác là việc sử dụng nhận dạng chữ số viết tay trong các kỳ thi số hóa, nơi học sinh viết đáp án trên giấy và hệ thống tự động chấm điểm.

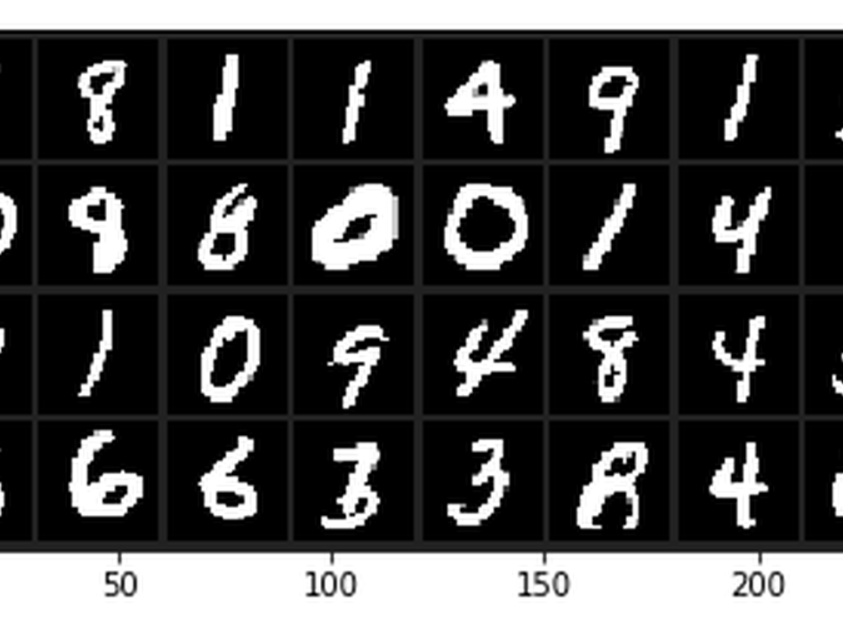
Các vấn đề cần đặt ra đối với bài toán nhận dạng chữ số viết tay:

**Tiền xử lý dữ liệu:** Một trong những thách thức lớn nhất của bài toán nhận dạng chữ số viết tay là hình ảnh đầu vào thường không đồng nhất. Chúng có thể bị ảnh hưởng bởi nhiễu, độ sáng không đều, hoặc kích thước không đồng nhất của chữ số. Vì vậy, tiền xử lý dữ liệu đóng vai trò rất quan trọng. Nó bao gồm các bước như chuyển đổi ảnh sang thang độ xám, làm sạch nhiễu, chuẩn hóa kích thước và tỷ lệ ảnh, cũng như cân bằng độ sáng và độ tương phản.

**Trích xuất đặc trưng từ dữ liệu:** Đặc trưng của chữ số viết tay rất đa dạng do phong cách cá nhân, kích thước và độ đậm nhạt của bút viết. Hệ thống cần có khả năng trích xuất các đặc trưng quan trọng và loại bỏ những yếu tố không cần thiết, ví dụ như nhiễu hoặc đường nét không liên quan.

**Lựa chọn mô hình học máy phù hợp:** Việc lựa chọn mô hình đóng vai trò quyết định đến hiệu suất của hệ thống. Các mô hình cần phù hợp với bản chất của dữ liệu, đồng thời đảm bảo cân bằng giữa độ chính xác và khả năng tổng quát hóa (generalization).

**Đánh giá và tối ưu hóa hệ thống:** Một hệ thống nhận dạng chữ số viết tay hiệu quả cần được đánh giá dựa trên các chỉ số như độ chính xác (accuracy), độ nhạy (recall), độ đặc hiệu (specificity), và F1-score. Dựa trên kết quả đánh giá, cần tiến hành các biện pháp tối ưu hóa để cải thiện hiệu năng, chẳng hạn như tinh chỉnh tham số (hyperparameter tuning) hoặc sử dụng các kỹ thuật tăng cường dữ liệu (data augmentation).



Hình 1.1 Hình ảnh tập dữ liệu MNIST chữ số viết tay

## **1.2. Các kỹ thuật sử dụng trong bài toán nhận dạng**

Trong bài toán nhận dạng chữ số viết tay, nhiều kỹ thuật đã được phát triển và ứng dụng thành công. Mỗi kỹ thuật đều có tư tưởng riêng, ưu nhược điểm rõ rệt, và phù hợp với các bài toán ở mức độ phức tạp khác nhau [2]

### **1.2.1. Kỹ thuật CNN (Convolutional Neural Network)**

Tư tưởng của CNN [4] : CNN là một mạng học sâu được thiết kế đặc biệt cho dữ liệu dạng ảnh. Thay vì sử dụng các đặc trưng được trích xuất thủ công, CNN tự động học các đặc trưng thông qua các lớp tích chập (convolutional layers), pooling layers, và fully connected layers.

Ưu điểm:

* Đạt độ chính xác cao trong nhận dạng chữ số viết tay.
* Tự động học đặc trưng mà không cần trích xuất thủ công.

Nhược điểm:

* Yêu cầu tài nguyên phần cứng mạnh để huấn luyện.p
* Cần lượng lớn dữ liệu huấn luyện để đạt hiệu quả tốt..

### **1.2.2 Kỹ thuật KNN (K-Nearest Neighbors)**

Tư tưởng của KNN: là một thuật toán học máy không tham số (non-parametric) hoạt động dựa trên nguyên tắc khoảng cách gần nhất. Khi một điểm dữ liệu mới xuất hiện, KNN sẽ xác định nhãn của nó bằng cách tìm k điểm lân cận gần nhất trong tập dữ liệu huấn luyện và áp dụng nguyên tắc đa số để xác định nhãn.Thường nó được dùng trong bài toán nhận dạng đơn giản mà ko cần huấn luyện mô hình

Ưu điểm:

* Dễ hiểu và dễ triển khai.
* Không yêu cầu huấn luyện mô hình trước, phù hợp với tập dữ liệu nhỏ.

Nhược điểm:

* Tốn thời gian khi làm việc với dữ liệu lớn, vì cần tính khoảng cách đến tất cả các điểm.
* Hiệu quả phụ thuộc mạnh vào giá trị của tham số k và phương pháp tính khoảng cách.

### **1.2.3 Kỹ thuật HOG (Histogram of Oriented Gradients)**

HOG là một kỹ thuật trích xuất đặc trưng trong nhận dạng hình ảnh, đặc biệt hữu ích cho phát hiện và nhận dạng đối tượng. Nguyên lý hoạt động chính của HOG dựa vào việc phân tích hướng gradient trong các vùng nhỏ của ảnh.

Ưu điểm:

- Hiệu quả cho đối tượng có cấu trúc cố định: Rất tốt trong phát hiện người đi bộ và vật thể có biên rõ ràng.

- Đơn giản và dễ triển khai: Không yêu cầu mạng nơ-ron phức tạp.

- Ổn định với thay đổi ánh sáng và nhiễu: Nhờ quá trình chuẩn hóa gradient.

Nhược điểm:

- Hạn chế với sự thay đổi hình học: Như xoay hoặc biến dạng.

- Hiệu suất không cao với dữ liệu phức tạp: Khó phát hiện vật thể trong bối cảnh nhiều nhiễu hoặc không đồng nhất.

- Tốn tài nguyên tính toán: Khi kích thước ảnh lớn.

### **1.2.4 Kỹ thuật R-CNN (Region-based Convolutional Neural Network)**

R-CNN là một mô hình dựa trên mạng nơ-ron tích chập (CNN) để phát hiện và nhận dạng vật thể trong hình ảnh. Đây là một trong những cải tiến quan trọng trong lĩnh vực nhận dạng đối tượng.

Ưu điểm:

- Độ chính xác cao: Nhờ mạng CNN, R-CNN có khả năng nhận dạng phức tạp và chính xác hơn so với các phương pháp truyền thống.

- Có thể áp dụng cho nhiều loại đối tượng: Được thiết kế để nhận dạng nhiều vật thể trong một ảnh.

Nhược điểm:

- Tốc độ chậm: Quá trình tạo vùng đề xuất và trích xuất đặc trưng từng vùng rất tốn thời gian.

- Tốn bộ nhớ: Mỗi vùng đề xuất cần phải lưu trữ và xử lý riêng biệt. Không khả thi cho thời gian thực: Không phù hợp với ứng dụng yêu cầu phản hồi nhanh, như camera giám sát.

## **1.3. Ngôn ngữ lập trình và các thư viện sử dụng**

### **1.3.1. Python**

Python là ngôn ngữ lập trình phổ biến nhất trong các bài toán xử lý ảnh và học máy nhờ vào tính dễ học, cú pháp đơn giản và hỗ trợ phong phú từ các thư viện mạnh mẽ.

- Tại sao chọn Python?

* Cộng đồng lớn, tài liệu phong phú.
* Tích hợp tốt với các thư viện học máy và học sâu.
* Tương thích với các công cụ như Jupyter Notebook, giúp trực quan hóa và kiểm thử mô hình nhanh chóng.

### **1.3.2. Các thư viện sử dụng**

**NumPy**:

Được sử dụng chủ yếu để làm việc với **mảng (array)** và tính toán khoa học. Nó cung cấp các công cụ hiệu quả để xử lý và thao tác trên các mảng lớn và phức tạp, đồng thời hỗ trợ nhiều phép toán số học và toán học cấp cao.

**Pandas:**

Được thiết kế để xử lý và phân tích dữ liệu có cấu trúc hoặc dạng bảng (tabular data). Với các công cụ dễ sử dụng và hiệu quả, Pandas được sử dụng phổ biến trong các lĩnh vực như khoa học dữ liệu, tài chính, và phân tích kinh doanh.

**Matplotlib và Seaborn:**

Để trực quan hóa dữ liệu. Trong khi **Matplotlib** cung cấp các công cụ cơ bản và linh hoạt để tạo đồ thị, thì **Seaborn** xây dựng trên nền tảng của Matplotlib và cung cấp các biểu đồ trực quan, thẩm mỹ, và dễ sử dụng hơn, đặc biệt là trong phân tích thống kê.

**Scikit-learn:**

Chủ yếu được sử dụng cho các bài toán học máy (machine learning). Nó cung cấp các công cụ dễ sử dụng để xây dựng và đánh giá mô hình học máy, bao gồm các thuật toán phân loại, hồi quy, clustering, giảm chiều dữ liệu, và nhiều kỹ thuật tiền xử lý dữ liệu.

**TensorFlow/Keras hoặc PyTorch:**

Là thư viện phổ biến và mạnh mẽ trong lĩnh vực học sâu (deep learning), dùng để xây dựng và triển khai các mô hình học máy phức tạp. Mỗi thư viện có những đặc điểm và ưu điểm riêng, nhưng cả hai đều hỗ trợ rất tốt việc xây dựng các mô hình mạng nơ-ron sâu (deep neural networks) và các bài toán học sâu khác.

**OpenCV:**

Là một thư viện mã nguồn mở mạnh mẽ dành cho xử lý và phân tích hình ảnh, video. OpenCV hỗ trợ nhiều tác vụ từ đơn giản đến phức tạp như nhận dạng chữ viết tay, nhận dạng đối tượng, xử lý hình ảnh, phân tích chuyển động, học sâu…

**Pyscreenshot**

Hỗ trợ chụp một vùng cụ thể của màn hình (bằng cách chỉ định tọa độ) và định dạng nó dưới dạng PNG,JPEG,...Thư viện này hữu ích trong tự động hóa, kiểm tra giao diện, và ghi lại hoạt động màn hình.

**Joblib**

Được thiết kế để giúp xử lý các tác vụ tính toán tốn thời gian, như lưu trữ dữ liệu lớn hoặc thực hiện tính toán song song (parallel computing). Joblib đặc biệt hữu ích trong các ứng dụng khoa học dữ liệu, học máy, hoặc bất kỳ nơi nào cần xử lý nhiều dữ liệu hoặc mô hình phức tạp.

**Tkinter**

Để xây dựng giao diện người dùng đồ họa (GUI). Nó cung cấp một cách dễ dàng và nhanh chóng để tạo ra các ứng dụng GUI với cửa sổ, nút bấm, hộp văn bản, menu và nhiều thành phần giao diện khác.

**PIL**

Thư viện xử lý hình ảnh phổ biến cho Python, cung cấp các hàm để đọc, lưu, thao tác và hiển thị hình ảnh. Xử lý hình ảnh chụp từ màn hình, chuyển đổi sang thang độ xám và chuẩn bị dữ liệu đầu vào cho mô hình.

# **CHƯƠNG II: XÂY DỰNG HỆ THỐNG**

## **2.1 Mô tả bài toán**

Bài toán tập trung vào việc xây dựng một ứng dụng nhận diện chữ viết tay số (digit recognition) sử dụng giao diện đồ họa với Tkinter. Người dùng sẽ viết các chữ số từ 0 đến 9 bằng công cụ Paint, và chương trình sẽ tự động chụp màn hình vùng vẽ, lưu ảnh vào các thư mục gắn nhãn phù hợp.   
Các ảnh này sau đó được xử lý thành ảnh grayscale, chuẩn hóa kích thước xuống 28x28 pixel, và hình ảnh được chuẩn hóa về [0, 1] bằng cách chia giá trị pixel cho 255. để lưu vào tệp CSV dưới dạng dữ liệu huấn luyện.

Dữ liệu được sử dụng để huấn luyện một mô hình học máy CNN nhằm phân loại chữ số viết tay. Sau khi huấn luyện, mô hình được lưu lại để tái sử dụng trong quá trình dự đoán. Người dùng có thể vẽ trực tiếp chữ số trên canvas của ứng dụng hoặc sử dụng chế độ "Live Paint" để vẽ trong Paint, sau đó ứng dụng sẽ tự động chụp ảnh, xử lý và dự đoán chữ số viết tay. Kết quả nhận diện được hiển thị trực tiếp trên giao diện, giúp kiểm tra tính chính xác của mô hình. Ứng dụng này không chỉ hỗ trợ việc tạo dữ liệu, huấn luyện mô hình mà còn cung cấp trải nghiệm tương tác trực quan trong quá trình dự đoán chữ số viết tay

## **2.2 Xây dựng hệ thống**

Để giải quyết bài toán nhận diện chữ viết tay, nhóm đã sử dụng các kỹ thuật chính sau đây:

Giao diện người dùng (GUI)

Tkinter: Dùng để xây dựng giao diện ứng dụng trực quan, bao gồm các thành phần như canvas để vẽ chữ số, nút bấm cho các thao tác như mở Paint, chụp màn hình, tạo dữ liệu, huấn luyện mô hình, và dự đoán. Điều này giúp người dùng tương tác dễ dàng với hệ thống mà không cần thao tác qua dòng lệnh.

Pillow (PIL): Sử dụng để xử lý và hiển thị hình ảnh trên giao diện. Các hình ảnh được resize và hiển thị trên canvas sau khi xử lý.

Thu thập và xử lý dữ liệu

Pyscreenshot: Được sử dụng để tự động chụp màn hình vùng người dùng vẽ trong Paint, tạo tập dữ liệu hình ảnh viết tay của chữ số từ 0 đến 9.

OpenCV: Sử dụng để xử lý hình ảnh, bao gồm:

Chuyển đổi ảnh sang grayscale (ảnh đen trắng).

Làm mờ ảnh bằng Gaussian Blur để giảm nhiễu.

Sử dụng kỹ thuật Threshold để chuyển đổi ảnh sang dạng nhị phân (black & white).

Resize ảnh xuống kích thước 28x28 pixel để chuẩn hóa cho mô hình.

Tạo dữ liệu dạng bảng (CSV): Các ảnh sau khi xử lý được chuyển đổi thành mảng 1D (784 pixel) và lưu dưới dạng tệp CSV, trong đó mỗi hàng đại diện cho một chữ số với nhãn tương ứng.

Huấn luyện mô hình nhận diện

Mô hình học máy CNN :

Sử dụng thuật toán CNN với kernel tuyến tính để phân loại chữ số.

Lý do chọn CNN: CNN hoạt động tốt trên các bài toán phân loại có số lượng đặc trưng lớn (784 pixel) và số lượng dữ liệu tương đối nhỏ.

Dữ liệu huấn luyện được chia thành tập huấn luyện và tập kiểm thử (80/20) để đánh giá độ chính xác của mô hình.

Tensor flow: Sử dụng thư viện này để xây dựng mô hình CNN và thực hiện các bước tiền xử lý dữ liệu.

Lưu mô hình: Mô hình sau khi huấn luyện được lưu lại bằng thư viện Joblib để tái sử dụng mà không cần huấn luyện lại.

Dự đoán và hiển thị kết quả

Dự đoán từ canvas: Người dùng vẽ trực tiếp chữ số lên canvas, ứng dụng sẽ xử lý ảnh tương tự như trong bước tạo dữ liệu, sau đó dự đoán kết quả sử dụng mô hình đã huấn luyện.

Live Paint: Ứng dụng cho phép người dùng vẽ chữ số trong Paint và tự động chụp màn hình, xử lý ảnh và dự đoán kết quả trong thời gian thực.

Hiển thị kết quả: Kết quả nhận diện chữ số được hiển thị trực tiếp trên giao diện, giúp người dùng dễ dàng kiểm tra và đánh giá mô hình.

Lưu trữ và quản lý dữ liệu

Dữ liệu hình ảnh và mô hình được lưu trữ trong các thư mục có cấu trúc rõ ràng. Tất cả dữ liệu huấn luyện, tập ảnh chụp, và mô hình học máy đều được lưu cục bộ để dễ dàng quản lý và sử dụng.

Đánh giá và tối ưu

Đánh giá độ chính xác: Độ chính xác của mô hình được tính toán dựa trên tập kiểm thử và hiển thị cho người dùng sau khi huấn luyện.

Tiền xử lý ảnh: Kỹ thuật Gaussian Blur và Threshold giúp làm sạch dữ liệu, loại bỏ nhiễu và tăng độ chính xác của dự đoán.

Về cơ bản chương trình sẽ có thể chia làm 3 giai đoạn cơ bản đó là giai đoạn tiền xử lý ảnh, giai đoạn chuyển thành dữ liệu, giai đoạn huấn luyện. Với việc bắt đầu việc nhận diện ảnh đầu vào đến kết thúc bằng nhận dạng.



Hình 2.1 Sơ đồ tổng quát nhận dang

Dưới đây sẽ cụ thể hơn các giai đoạn đó và các kỹ thuật sử dụng trọng các giai đoạn

### **2.2.1 Giai đoạn tiền xử lý ảnh**

Giai đoạn này góp phần làm tăng độ chính xác phân lớp của hệ thống nhận dạng, tuy nhiên nó cũng làm cho tốc độ nhận dạng của hệ thống chậm lại. Vì vậy, tùy thuộc vào chất lượng ảnh quét vào của từng văn bản cụ thể để chọn một hoặc một vài chức năng trong khối này. Nếu cần ưu tiên tốc độ xử lý và chất lượng của máy quét tốt thì có thể bỏ qua giai đoạn này. Khối tiền xử lý bao gồm một số chức năng: Chuyển đổi ảnh sang ảnh xám, làm mịn ảnh, Nhị phân hóa ảnh, Chuẩn hóa kích thước ảnh , chuyển đổi ảnh thành dữ liệu số,...

### **2.2.2 Chuyển đổi ảnh sang ảnh xám (Grayscale Conversion)**

Chuyển đổi ảnh sang màu xám có rất nhiều công dụng trong việc tiền xử lý ảnh ban đầu:

**Giảm kích thước dữ liệu:** Ảnh màu thường chứa nhiều thông tin hơn ảnh xám, do đó việc chuyển đổi thành ảnh xám giúp giảm kích thước dữ liệu.

**Tăng hiệu quả xử lý:** Khi xử lý ảnh, việc sử dụng ảnh xám thay vì ảnh màu giúp tăng tốc độ và hiệu quả của các thuật toán do ít dữ liệu hơn cần phải xử lý.

**Đơn giản hóa các thao tác xử lý ảnh:** Nhiều thuật toán xử lý ảnh hoạt động tốt hơn hoặc chỉ yêu cầu ảnh xám, chẳng hạn như nhận diện cạnh, phát hiện vật thể, và các kỹ thuật lọc ảnh.

**Tăng tính tương phản:** Ảnh xám giúp làm nổi bật các vùng sáng và tối, tăng tính tương phản của ảnh, giúp các chi tiết dễ dàng nhìn thấy hơn.

Với chức năng cơ bản của chu trình này giúp ảnh cần được xử lý chuyển ảnh màu sang ảnh grayscale để giảm thông tin không cần thiết (giảm từ 3 kênh màu RGB xuống còn 1 kênh độ sáng). Điều này giúp tối ưu xử lý ảnh trong các bước tiếp theo.

### **2.1.3 Làm mịn**

Làm mịn ảnh trong nhận diện chữ số viết tay có tác dụng quan trọng trong việc cải thiện chất lượng của hình ảnh đầu vào và tăng độ chính xác của các thuật toán nhận diện. Các tác dụng chính của làm mịn trong giai đoạn tiền xử lý ảnh:

**Giảm nhiễu:** Làm mịn ảnh giúp loại bỏ các điểm nhiễu và chi tiết không cần thiết, làm cho các chữ số viết tay trở nên rõ ràng hơn.

**Làm mềm các đường nét:** Giúp làm mềm các đường nét và biên của chữ số, tạo điều kiện cho các thuật toán nhận diện dễ dàng phát hiện và phân biệt các đường nét của chữ số.

**Cải thiện tính nhất quán:** Làm mịn ảnh có thể giúp tạo ra hình ảnh đầu vào đồng nhất hơn, giảm sự biến dạng và bất thường, từ đó cải thiện tính nhất quán của các đặc trưng được trích xuất.

**Tăng độ chính xác của phân đoạn:** Giúp cải thiện quá trình phân đoạn chữ số, đảm bảo rằng các chữ số riêng lẻ được tách biệt một cách chính xác, đặc biệt trong các trường hợp chữ số viết tay chạm nhau hoặc đè lên nhau.

**Giảm độ phức tạp tính toán:** Bằng cách làm mịn ảnh, số lượng điểm ảnh cần xử lý có thể được giảm bớt, từ đó giảm độ phức tạp tính toán và tăng hiệu quả của quá trình nhận diện.

Các kỹ thuật làm mịn phổ biến bao gồm Gaussian Blur, Median Filter và Bilateral Filter, mỗi loại có những ưu điểm riêng tùy thuộc vào đặc thù của nhiệm vụ nhận diện và chất lượng ảnh đầu vào.

### **2.1.4 Nhị phân hóa ảnh**

Nhị phân hóa ảnh là một kỹ thuật chuyển ảnh đa cấp xám sang ảnh nhị phân. Trong bất kỳ bài toán phân tích hoặc nâng cao chất lượng ảnh nào, nó cũng cần thiết để xác định các đối tượng quan trọng. Nhị phân hóa ảnh phân chia ảnh thành 2 phần: phần nền và phần chữ. Hầu hết các phương pháp nhị phân hóa ảnh hiện nay đều lựa chọn một ngưỡng thích hợp theo cường độ sáng của ảnh và sau đó chuyển tất cả các giá trị độ sáng lớn hơn ngưỡng đó thành một giá trị độ sáng và tất cả các giá trị bé hơn ngưỡng thành một giá trị độ sáng khác. Nói một cách đơn giản đó là giúp cho ảnh cần xử lý Chuyển ảnh grayscale thành ảnh nhị phân (black & white), giúp phân tách rõ ràng chữ số (vùng foreground) khỏi nền (background).

### **2.1.5 Chuẩn hóa kích thước hình ảnh**

Việc chuẩn hóa kích thước ảnh dựa trên việc xác định trọng tâm ảnh, sau đó xác định khoảng cách lớn nhất từ tâm ảnh đến các cạnh trên, dưới, trái, phải của hình chữ nhật bao quanh ảnh. Thông qua khoảng cách lớn nhất đó, có thể xác định được một tỷ lệ co, giãn của ảnh gốc so với kích thước đã xác định, từ đó hiệu chỉnh kích thước ảnh theo tỷ lệ co, giãn này. Như vậy, thuật toán chuẩn hóa kích thước ảnh luôn luôn đảm bảo được tính cân bằng khi co giãn ảnh, ảnh sẽ không bị biến dạng hoặc bị lệch.

Như trong chương trình nhận diện chữ số viết tay nó sẽ điều chỉnh kích thước thành 28x28 pixel để phù hợp với yêu cầu của mô hình nhận diện chữ số

### **2.1.6 Mô hình CNN.**

Trong chương trình chúng em , mô hình được sử dụng là Convolutional Neural Networks (CNN) .Đây là một trong những thuật toán học máy mạnh mẽ và phổ biến trong các bài toán nhận dạng. CNN là một loại mô hình học sâu (Deep Learning) đặc biệt, được thiết kế chủ yếu để xử lý dữ liệu có cấu trúc dạng lưới, chẳng hạn như hình ảnh và video. CNN được lấy cảm hứng từ cách hệ thống thị giác của con người hoạt động và rất hiệu quả trong các nhiệm vụ như nhận diện hình ảnh, phân loại đối tượng, và xử lý ngôn ngữ tự nhiên.  
 CNN bao gồm các cấu trúc cơ bản sau

**Lớp Convolution (Tích chập)**:

Là thành phần chính của CNN, dùng để trích xuất các đặc trưng từ hình ảnh đầu vào.

Tích chập được thực hiện bằng cách sử dụng các bộ lọc (filters/kernels), là các ma trận nhỏ trượt qua hình ảnh và thực hiện phép nhân chấm (dot product) tại mỗi vị trí.

Kết quả của quá trình này là một ma trận mới gọi là **feature map** (bản đồ đặc trưng), biểu diễn các đặc trưng trích xuất từ hình ảnh.

**Lớp Activation (Hàm kích hoạt)**:

Thường sử dụng hàm ReLU (Rectified Linear Unit), có nhiệm vụ biến đổi các giá trị âm thành 0, giúp mạng học các mối quan hệ phi tuyến tính.

**Lớp Pooling (Gộp)**:

Giảm kích thước của feature map, giúp giảm số lượng tham số và cải thiện hiệu quả tính toán.

Các kỹ thuật phổ biến: Max Pooling (lấy giá trị lớn nhất trong một vùng), Average Pooling (lấy giá trị trung bình).

Quá trình này cũng giúp mạng bền vững hơn trước các biến đổi nhỏ của dữ liệu, chẳng hạn như dịch chuyển hoặc phóng to/thu nhỏ.

**Lớp Fully Connected (Kết nối đầy đủ)**:

Sau khi các đặc trưng được trích xuất qua các lớp convolution và pooling, chúng được đưa vào lớp fully connected để thực hiện phân loại.Lớp này hoạt động giống như mạng neural truyền thống, nơi mỗi neuron được kết nối với tất cả các neuron của lớp trước đó.

**Lớp Output (Đầu ra)**:

Sử dụng hàm softmax để đưa ra xác suất của từng lớp, từ đó quyết định nhãn của hình ảnh

CNN có tác dụng trong bài toán nhận diện chữ số viết tay

**Trích xuất đặc trưng từ hình ảnh**

CNN tự động học và trích xuất các đặc trưng quan trọng từ hình ảnh đầu vào (các pixel chữ viết tay).

Đặc trưng này bao gồm:

Các đường nét, góc cạnh (edges).

Các chi tiết phức tạp hơn khi đi sâu vào các lớp.

Điều này giúp mô hình hiểu được các hình dạng và cấu trúc của chữ số viết tay mà không cần thiết kế thủ công các đặc trưng.

**Xử lý hình ảnh dưới dạng ma trận**

Mỗi hình ảnh chữ số được resize thành kích thước 28×2828 \times 2828×28, sau đó đưa vào CNN để xử lý.

CNN sử dụng các lớp **Convolution** và **Pooling** để giảm dần kích thước và giữ lại những thông tin quan trọng.

Điều này giúp mô hình tập trung vào các đặc trưng chính, đồng thời giảm chi phí tính toán.

**Phân loại chữ viết tay**

Sau khi trích xuất đặc trưng, CNN sử dụng các lớp kết nối đầy đủ (Fully Connected) để thực hiện phân loại:

Mỗi đầu ra của lớp Dense tương ứng với một chữ số từ 0 đến 9.

Lớp cuối cùng sử dụng hàm **softmax** để dự đoán xác suất thuộc về từng lớp.

Cuối cùng, lớp có xác suất cao nhất sẽ được chọn làm kết quả dự đoán.

**Tăng độ chính xác và giảm lỗi**

CNN giúp cải thiện độ chính xác so với các phương pháp truyền thống nhờ khả năng học tự động các đặc trưng quan trọng. Sử dụng các lớp như **ReLU** và **MaxPooling** giúp mô hình bền vững hơn với các biến đổi nhỏ trong dữ liệu, chẳng hạn như dịch chuyển, xoay hoặc thay đổi kích thước của chữ viết.

**Ứng dụng trong các bước của chương trình**

**Tạo và huấn luyện mô hình**:  
CNN được huấn luyện trên tập dữ liệu gồm các hình ảnh chữ số viết tay. Nó học cách nhận diện các chữ số dựa trên đặc trưng của chúng.

**Lợi ích cụ thể của CNN hệ thống**

**Tự động hóa trích xuất đặc trưng**: Không cần phải tự thiết kế các thuật toán để nhận diện các nét chữ.

**Hiệu suất cao**: Khả năng nhận diện chính xác ngay cả khi chữ viết không rõ ràng hoặc có biến dạng nhỏ.

**Tính tổng quát**: Sau khi huấn luyện, mô hình có thể nhận diện tốt trên các hình ảnh chữ số viết tay khác nhau, không chỉ giới hạn trong tập huấn luyện.

**Tính linh hoạt**: Có thể mở rộng để nhận diện các đối tượng hoặc ký tự khác bằng cách thay đổi dữ liệu và cấu trúc mô hình.

CNN là phần cốt lõi giúp chương trình của bạn thực hiện nhiệm vụ nhận diện chữ viết tay một cách hiệu quả và chính xác.

Sau khi mô hình đã được huấn luyện và lưu lại, chúng ta có thể sử dụng mô hình này để nhận diện các chữ số viết tay từ ảnh đầu vào mới. Sử dụng mô hình huấn luyện để dự đoán: Mô hình được nạp từ tệp và sử dụng để dự đoán chữ số từ dữ liệu mới.

### **2.1.7 Giai đoạn Quản lý dữ liệu**

Quản lý dữ liệu trong hệ thống nhận diện chữ viết tay là một bước quan trọng trong việc xây dựng mô hình học máy. Giai đoạn này liên quan đến việc thu thập, tổ chức và chuẩn bị dữ liệu để có thể sử dụng hiệu quả trong quá trình huấn luyện và nhận dạng.

**Thu thập dữ liệu:**

Dữ liệu đầu vào là các hình ảnh được tạo ra từ các thao tác vẽ trong ứng dụng Paint hoặc từ dữ liệu được ghi lại từ các nguồn khác. Những hình ảnh này đại diện cho các chữ số viết tay mà mô hình cần nhận diện.

Dữ liệu thu thập sẽ được tổ chức trong các thư mục riêng biệt, mỗi thư mục chứa các hình ảnh của một chữ số cụ thể (0-9). Mỗi hình ảnh được gắn nhãn với đúng chữ số mà nó đại diện.

**Tạo Tập Dữ liệu:**

Sau khi thu thập, dữ liệu hình ảnh cần được chuyển đổi thành dạng mà máy học có thể hiểu được. Điều này đụng đến việc trích xuất các đặc trưng từ hình ảnh.

Dữ liệu ảnh sẽ được tiền xử lý (như chuyển thành ảnh đen trắng, làm mờ, và chuyển sang dạng nhị phân) trước khi trích xuất các đặc trưng.

Tất cả các đặc trưng từ các hình ảnh (được mã hóa dưới dạng các pixel trong ảnh) sẽ được thu thập vào một bảng dữ liệu. Hình ảnh được chuẩn hóa về [0, 1] bằng cách chia giá trị pixel cho 255..

**Lưu trữ và Quản lý Dữ liệu:**

Dữ liệu sẽ được lưu trữ trong các tệp dạng CSV hoặc trong các cơ sở dữ liệu, với mỗi dòng tương ứng với một ảnh đã được tiền xử lý.

Thông tin về các nhãn sẽ được lưu trữ kèm theo mỗi hình ảnh, giúp việc huấn luyện và đánh giá mô hình trở nên chính xác hơn.

### **2.1.8 Giai đoạn Huấn luyện mô hình**

Huấn luyện mô hình là bước quan trọng tiếp theo trong việc phát triển một hệ thống nhận diện chữ viết tay. Trong giai đoạn này, mô hình học máy sẽ học từ các dữ liệu đã chuẩn bị để xây dựng khả năng phân loại chữ số viết tay.

**Chuẩn Bị Dữ liệu Huấn luyện:**

**Thu thập dữ liệu**: Dữ liệu được thu thập từ các nguồn khác nhau (ví dụ: hình ảnh chữ viết tay trong chương trình của bạn).

Tiền xử lý dữ liệu: Chuyển đổi hình ảnh sang định dạng phù hợp (như thang xám, kích thước cố định, v.v.), Chuẩn hóa giá trị đầu vào (ví dụ: chia giá trị pixel cho 255 để chuyển phạm vi từ [0, 255] về [0, 1]).

Gán nhãn: Xác định nhãn chính xác cho mỗi mẫu dữ liệu. Trong bài toán nhận diện chữ số viết tay, nhãn là các số từ 0 đến 9.

Tách dữ liệu:Dữ liệu huấn luyện (Training set): Dùng để huấn luyện mô hình (thường chiếm 70-80% dữ liệu),Dữ liệu kiểm tra (Validation set): Dùng để đánh giá hiệu suất mô hình trong quá trình huấn luyện,Dữ liệu kiểm định (Test set): Được giữ lại để đánh giá cuối cùng sau khi mô hình được tối ưu.

**Chọn Mô hình Học Máy:**

Trong bài toán này, mô hình học máy được lựa chọn là **Convolutional Neural Network (CNN)**. Mô hình này thích hợp vì nó có khả năng có khả năng tự động học các đặc trưng quan trọng từ hình ảnh qua các lớp tích chập (Convolutional Layers)

**Quá Trình Huấn Luyện:**

Khởi tạo mô hình: Bắt đầu với các trọng số ngẫu nhiên.

Forward Propagation:

* Dữ liệu được truyền qua các lớp của mô hình.
* Tại mỗi lớp, các phép toán như tích chập, kích hoạt (ReLU), và gộp (Pooling) được thực hiện.
* Kết quả cuối cùng là dự đoán của mô hình.

Tính toán hàm mất mát (Loss Function):

Đo lường sự khác biệt giữa dự đoán của mô hình và nhãn thực tế (ví dụ: hàm Cross-Entropy Loss trong phân loại đa lớp).

Backward Propagation (Lan truyền ngược):

Tính toán đạo hàm của hàm mất mát theo từng trọng số trong mô hình.

Cập nhật trọng số để giảm giá trị hàm mất mát, thường sử dụng các thuật toán tối ưu như Adam hoặc SGD (Stochastic Gradient Descent).

Lặp lại:

Tiến hành nhiều vòng lặp (epochs) với toàn bộ dữ liệu huấn luyện để cải thiện hiệu suất của mô hình.

**Đánh Giá Mô hình:**

Sau khi huấn luyện xong, mô hình sẽ được kiểm tra với tập dữ liệu kiểm tra. Mô hình sẽ đưa ra dự đoán cho dữ liệu kiểm tra và so sánh với nhãn thực tế để tính toán độ chính xác của mô hình.

Nếu độ chính xác đạt yêu cầu, mô hình có thể được lưu lại để sử dụng trong các bước nhận dạng sau này.

### **2.1.9 Giai đoạn Nhận dạng**

Nhận dạng là bước cuối cùng trong hệ thống, nơi mô hình đã huấn luyện sẽ được sử dụng để dự đoán các chữ số viết tay từ các hình ảnh đầu vào mới.

**Thu Thập Dữ Liệu Đầu Vào:**

Người dùng vẽ các chữ số trực tiếp trong phần mềm Paint hoặc trên một khu vực vẽ trong ứng dụng. Sau đó, phần mềm sẽ thu thập dữ liệu hình ảnh này.

**Tiền Xử Lý Dữ Liệu Nhận Dạng:**

Hình ảnh từ Paint hoặc khu vực vẽ sẽ được tiền xử lý trước khi đưa vào mô hình để nhận dạng. Các bước tiền xử lý bao gồm:

**Chuyển ảnh sang grayscale** để giảm độ phức tạp của dữ liệu.

**Làm mờ ảnh** để giảm nhiễu và giúp nhận dạng các đặc trưng rõ ràng hơn.

**Chuyển ảnh thành ảnh nhị phân** (black and white) để dễ dàng nhận diện các chi tiết.

**Resize ảnh** về kích thước chuẩn (28x28 pixel) để phù hợp với mô hình đã được huấn luyện.

**Làm phẳng ảnh** thành một vector 1 chiều với các giá trị pixel nhị phân.

**Chuẩn hóa giá trị pixel** : Giá trị pixel từ [0, 255] được chuẩn hóa về [0, 1] bằng cách chia mỗi giá trị cho 255.

**Dự Đoán Với Mô Hình:**

Sau khi dữ liệu được tiền xử lý, mô hình CNN đã được huấn luyện sẽ được sử dụng để dự đoán chữ số trong hình ảnh. Mô hình sẽ phân tích các giá trị pixel và đưa ra dự đoán về chữ số mà hình ảnh đại diện.

**Hiển Thị Kết Quả:**

Kết quả dự đoán sẽ được hiển thị cho người dùng dưới dạng một nhãn chữ số, ví dụ như "Dự đoán: 7".

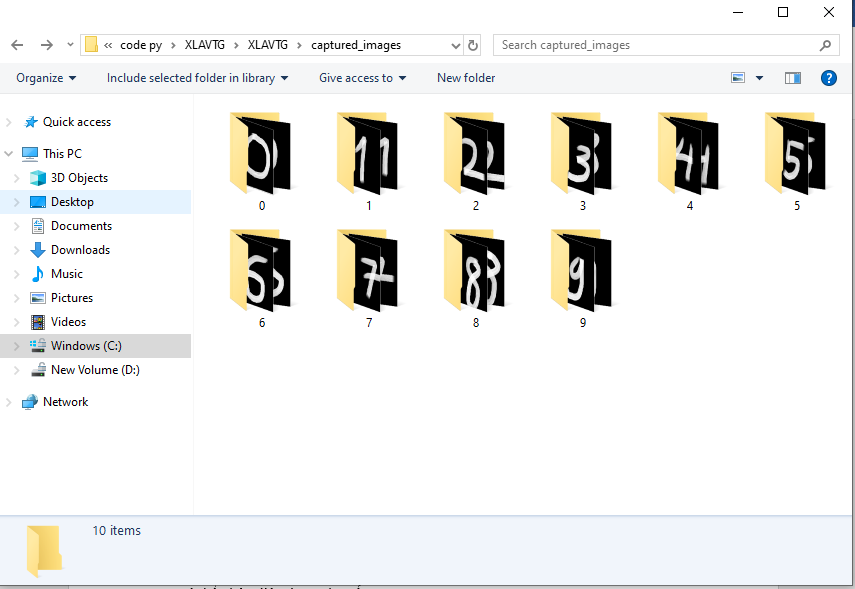
# **CHƯƠNG III: THỰC NGHIỆM**

## **3.1 Dữ liệu**

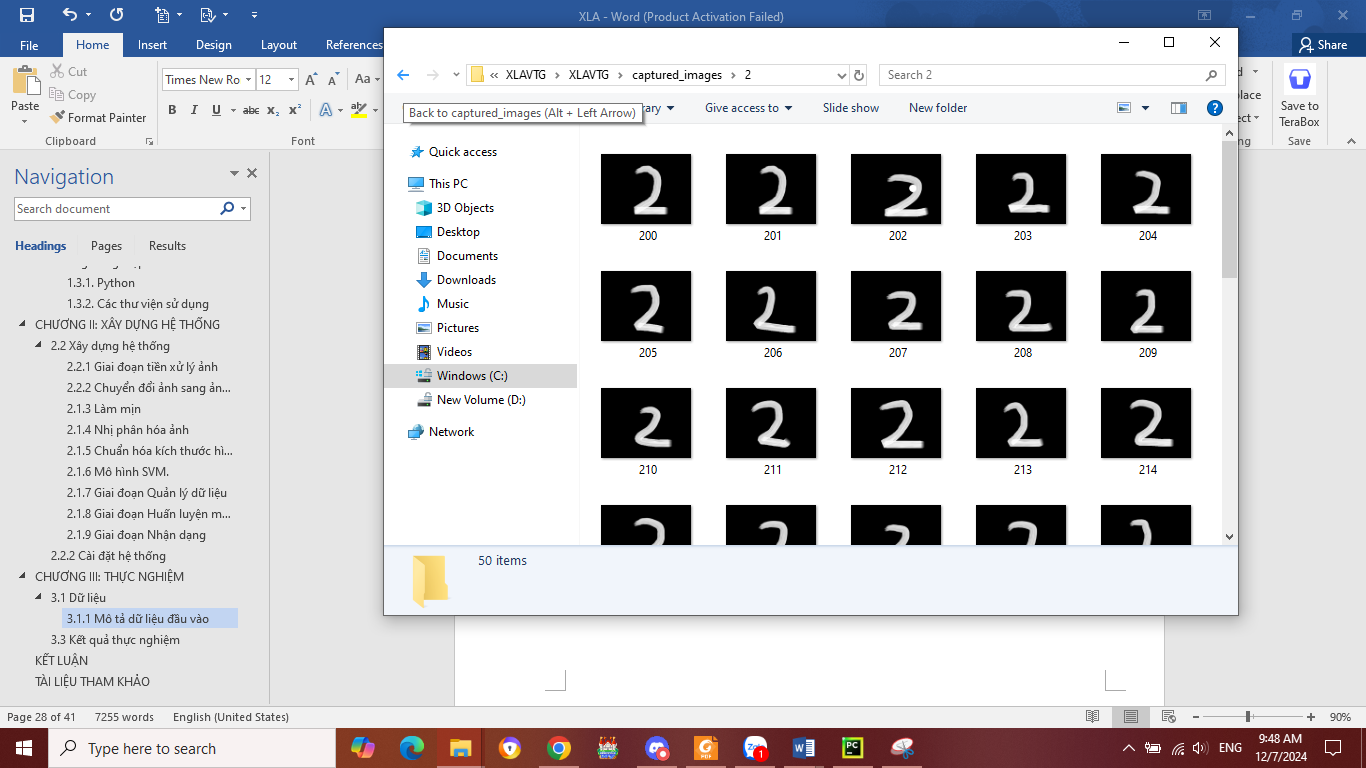
Dữ liệu đóng vai trò rất quan trọng trong việc xây dựng hệ thống nhận dạng chữ viết tay, đặc biệt với các mô hình sử dụng học sâu như mạng nơron. Chất lượng, độ tin cậy, tính sẵn có và sự phù hợp của dữ liệu là yếu tố quyết định hiệu quả của hệ thống. Ngay cả các mô hình đơn giản cũng có thể cho kết quả tốt nếu dữ liệu đầu vào được xử lý kỹ lưỡng, giữ được các thông tin đặc trưng quan trọng. Ngược lại, các mô hình phức tạp cũng khó đạt hiệu suất mong muốn nếu dữ liệu chứa nhiều nhiễu hoặc không phù hợp.

### **3.1.1 Mô tả dữ liệu đầu vào**

Tập dữ liệu được bọn em sử dụng là Tệp hình ảnh gồm các tệp từ 0-9 trong 10 tệp này mỗi tệp giống như tên file có các ảnh do chính bọn vẽ tay các số từ 0-9 bằng paint



Hình 3.1 Tệp file dữ liệu đầu vào



Tổng cộng có 500 bức ảnh có định dạng sẵn là (361,282,13)

### **3.1.2 Kỹ thuật xử lý tiền dữ liệu**

Quá trình tiền xử lý dữ liệu là 1 quá trình vô cùng quan trọng và bắt buộc phải có để trong quá trình xử lý dữ liệu để làm sạch dữ liệu trước khi đem đi huấn luyện

trong quá trình **tiền xử lý dữ liệu** để nhận diện chữ viết tay, chúng em đã áp dụng các kỹ thuật tiền xử lý như sau:

**Chuyển đổi ảnh sang thang độ xám (Grayscale Conversion)**

**Mô tả**:

Chuyển đổi ảnh từ không gian màu RGB (hoặc BGR) sang thang độ xám.

Giúp giảm số lượng kênh màu từ 3 xuống 1 (một ma trận duy nhất đại diện cho cường độ sáng).

Giảm kích thước dữ liệu, tăng tốc độ xử lý và tập trung vào các đặc trưng chính.

**Làm mờ ảnh (Gaussian Blur)**

**Mô tả**:

Áp dụng bộ lọc Gaussian để làm mờ ảnh.

Giảm nhiễu và chi tiết nhỏ không cần thiết, giúp làm nổi bật các đặc trưng chính của chữ viết tay.

**Ngưỡng hóa ảnh (Thresholding)**

**Mô tả**:

Chuyển đổi ảnh thành ảnh nhị phân (chỉ có 2 giá trị pixel: 0 và 255).

Những pixel có giá trị lớn hơn ngưỡng (100) được đặt thành 255 (trắng), còn lại là 0 (đen).

Giúp phân tách rõ nét giữa chữ viết tay và nền, làm đơn giản hóa quá trình trích xuất đặc trưng.

**Resize ảnh về kích thước chuẩn (Resizing)**

**Mô tả**:

Thu nhỏ hoặc phóng to ảnh về kích thước chuẩn 28x28.

Đây là định dạng đầu vào của nhiều mô hình nhận diện chữ viết tay (như MNIST) [5]

Giảm kích thước dữ liệu trong khi vẫn bảo toàn các đặc trưng cần thiết.

**Biến đổi ảnh thành mảng pixel 1D (Flattening)**

Chuyển ma trận 28x28 thành một mảng 1D gồm 784 phần tử.

Các pixel được ánh xạ sang giá trị nhị phân (0 hoặc 1) dựa trên ngưỡng.

Đây là bước tiền xử lý cuối cùng để chuẩn bị dữ liệu đầu vào cho mô hình.

**Gán nhãn (Labeling)**

**Mô tả**:

Mỗi ảnh được gán một nhãn tương ứng (từ 0 đến 9).

Đây là giá trị mục tiêu (label) dùng để huấn luyện mô hình.

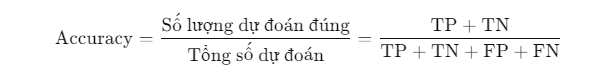
### **3.1.3 Chia train-test**

Chúng em đang sử dụng cách chia dữ liệu cơ bản với tỷ lệ 80% huấn luyện và 20% kiểm tra. Kỹ thuật này giúp đảm bảo dữ liệu kiểm tra được tách riêng để đánh giá mô hình mà không bị overfitting.

**Cách thực hiện trong code:**  
  
 Hình 3.2 Chia train và test  
Giải thích:  
Dữ liệu đầu vào (`X`):  
 - Đây là toàn bộ các đặc trưng (features) của dữ liệu. Trong trường hợp này, là mảng các pixel của mỗi ảnh (sau khi được xử lý thành mảng 1D).  
 - Được lấy từ tập dữ liệu CSV sau khi loại bỏ cột nhãn (`label`):  
  
Nhãn (`Y`):  
 - Đây là giá trị nhãn của từng ảnh, tương ứng với con số viết tay (0-9).  
 - Được lấy từ cột `label` trong tập dữ liệu:  
  
Tham số `test\_size=0.2`:  
 - Quy định tỷ lệ tập dữ liệu được sử dụng để kiểm tra (test set).   
 - Ở đây, 20% dữ liệu được dành cho kiểm tra (test set), 80% còn lại dành cho huấn luyện (training set).  
  
Trả về 4 tập con:  
 - `train\_x`: Các đặc trưng dùng để huấn luyện.  
 - `test\_x`: Các đặc trưng dùng để kiểm tra.  
 - `train\_y`: Nhãn tương ứng của `train\_x`.  
 - `test\_y`: Nhãn tương ứng của `test\_x`.

## **3.2 Độ đo đánh giá**

Trong hệ thống của chúng em có tính độ đo chính xác của chương trình dựa trên tệp dữ liệu đã huấn luyện Đây là độ đo chính để đánh giá hiệu quả của mô hình. Độ chính xác (Accuracy) được tính theo công thức :



Với:

**TP (True Positives):** Số lượng mẫu thuộc lớp dương (Positive) mà mô hình dự đoán đúng.

**TN (True Negatives):** Số lượng mẫu thuộc lớp âm (Negative) mà mô hình dự đoán đúng.

**FP (False Positives):** Số lượng mẫu thuộc lớp âm nhưng bị mô hình dự đoán nhầm là lớp dương.

**FN (False Negatives):** Số lượng mẫu thuộc lớp dương nhưng bị mô hình dự đoán nhầm là lớp âm.

**Trong bài toán phân loại chữ số viết tay:**

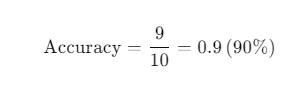
**TP/TN:** Số chữ số được mô hình dự đoán đúng (dự đoán khớp với nhãn thực tế).

**FP/FN:** Số chữ số mà mô hình dự đoán sai (không khớp với nhãn thực tế).



Hình 3.3 Hàm tính độ đo

Ví dụ:  
Giả sử có 10 mẫu trong tập kiểm tra (test\_x), nhãn thực tế (test\_y) là từ 0-9 và kết quả dự đoán (prediction của mô hình là ) 0,1,2,3,4,5,6,7,8,0 số lượng dự đoán đúng là 9 trừ chữ số cuối trên tổng số mẫu là 10 thì ta sẽ có công thức dưới



## **3.3 Kết quả thực nghiệm**

Sau một khoảng thời gian tìm hiểu và làm bài chúng em đã có thể tạo ra được một chương trình của hệ thống nhận diện chữ số viêt tay có đầy đủ cơ bản các chức năng để có thể nhận diện được chữ số

Giao diện chính của chương trình khi khởi chạy:

  
Hình 3.4 Giao diện chính khi được hoàn thành

Chương trình sẽ có các chức năng chính như sau:

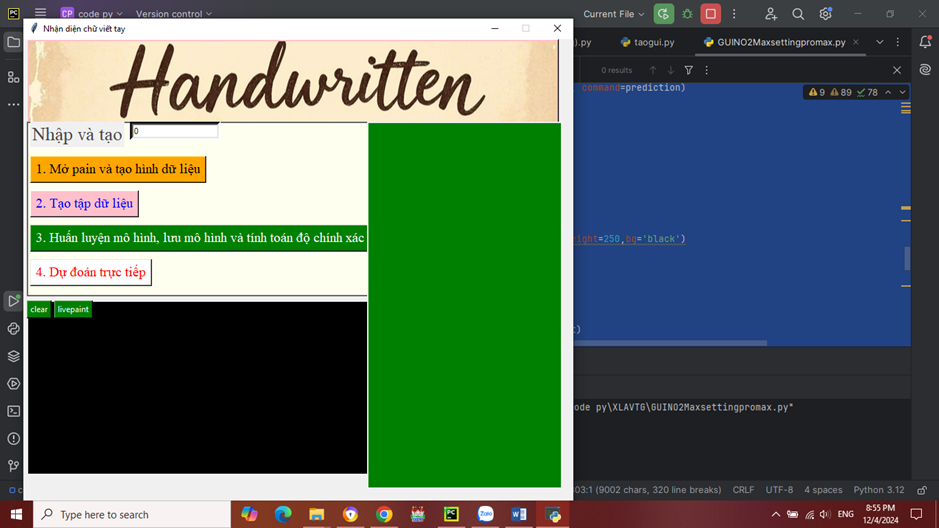
Nhập và tạo file hình dữ liệu

Tạo tập dữ liệu

Huấn luyện mô hình, lưu mô hình và tính toán độ chính xác

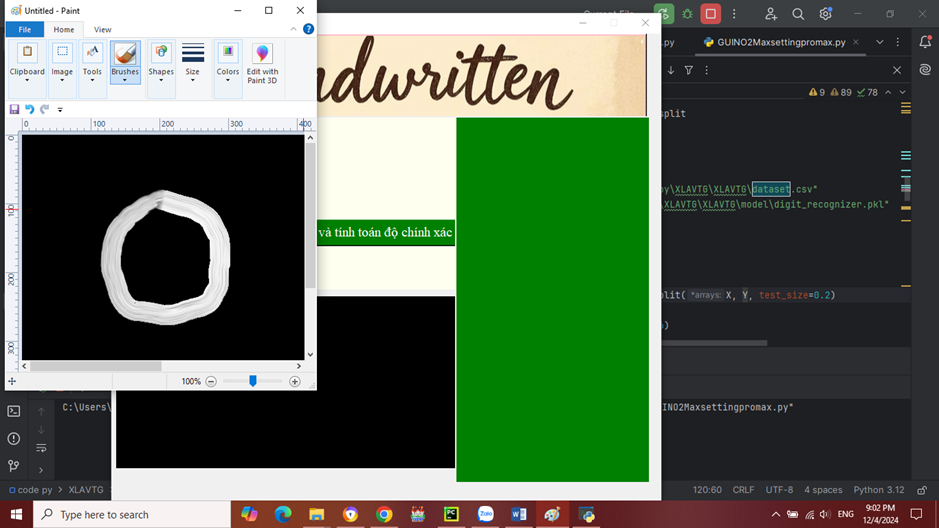
Dự đoán chữ số viết tay

Nhập một số bất kì vào phần **nhập và tạo** để tạo tên của folder hoặc là đưa những bức ảnh vào folder đã có sẵn sau đó nhấn vào **Mở pain và tạo hình dữ liệu** để bắt đầu tạo file trống để hình dữ liệu



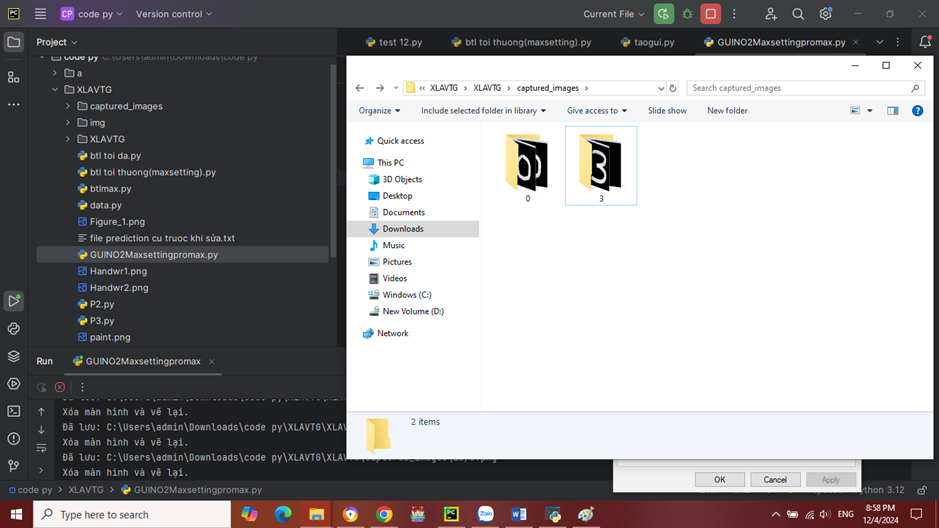
Hình 3.5 Chức năng số 1

Sau khi nhấn vào phần **Mở pain và tạo hình dữ liệu** sẽ đưa đến giao diện **Microsoft Paint** rồi khi đó chúng ta sẽ vẽ ra hình số mà chúng ta đã nhập ở phần nhập và tạo **,**  sau mỗi 8 giây chương trình sẽ tự chụp lại màn hình khoảng đen mà chúng ta đang vẽ để lưu ảnh vào trong file chúng ta có thể xóa đi và vẽ thêm nhiều hình dạng khác nhau để dễ hơn cho việc xử lý dữ liệu ảnh.



Hình 3.6 Thu thập dữ liệu bằng paint

Sau khi vẽ và tạo các hình ảnh dữ liệu thành công sẽ tạo ra được các file dữ liệu như hình ở dưới, lặp lại các quá trình ở trên để tạo thêm được nhiều dữ liệu hơn để hỗ trợ cho việc xử lý ảnh .



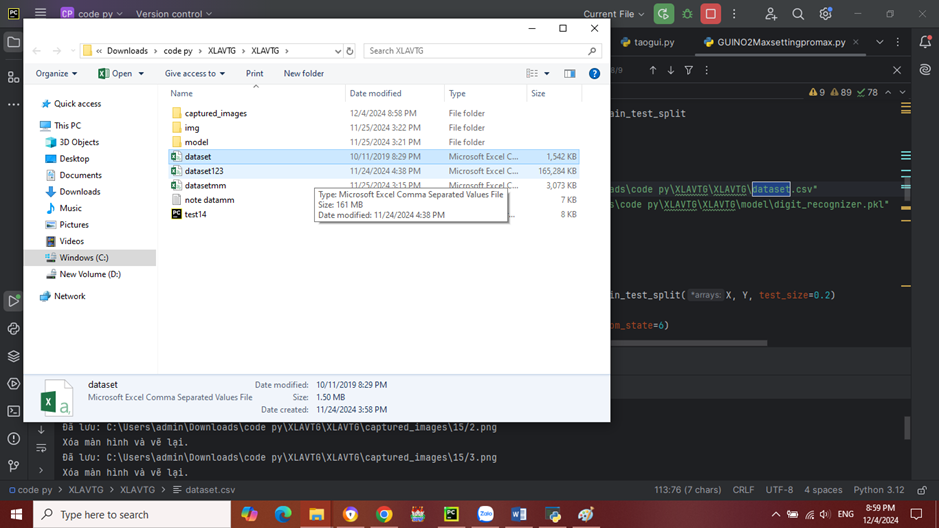
Hình 3.7 Tệp file lưu dữ liệu

Các bức ảnh dữ liệu sẽ được đưa vào trong 1 nơi folder lưu gồm các tệp mà ta đã tạo và vẽ từ bức trước

Dựa vào những bức ảnh chúng ta đã vẽ để sử dụng chức năng tạo tập dữ liệu

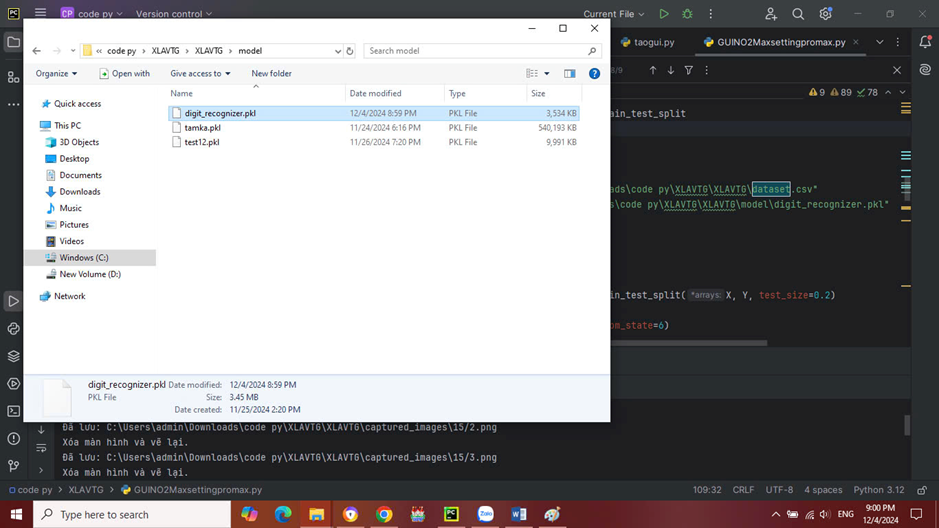
  
Hình 3.8 Sử dụng chức năng 2

Sau khi tạo tập dữ liệu thành công chúng ta sẽ nhận được một file **dataset**, chức năng của file **dataset** này là sử dụng để huấn luyện các mô hình AI/học máy và kiểm tra hiệu suất của mô hình sau khi huấn luyện



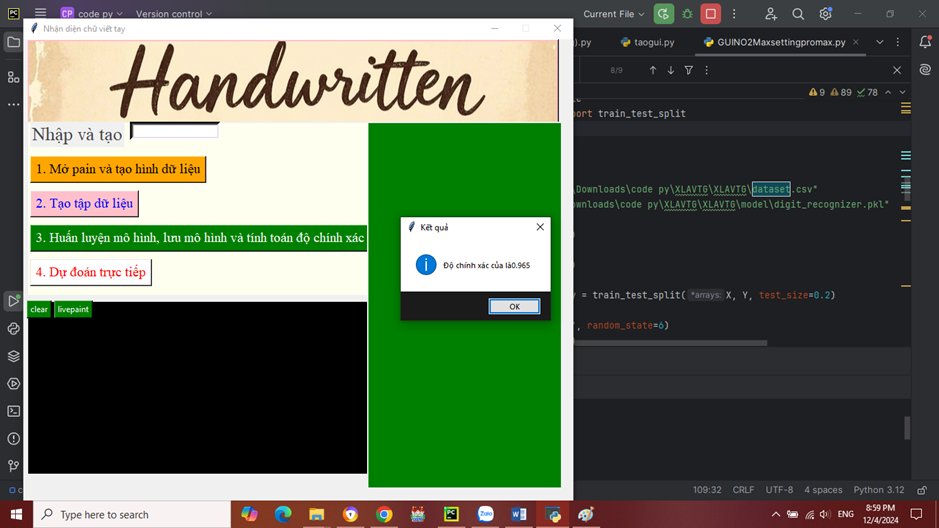
Hình 3.9 Tệp file lưu dataset

Sau khi tạo được file **dataset** như ở trên ta sẽ bắt đầu Huấn luyện mô hình dựa trên file **dataset** để tạo ra file **model** **digit\_recognizer.pkl** ở trong file này đã gồm có các mô hình nhận diện chữ số từ hình ảnh để dự đoán.



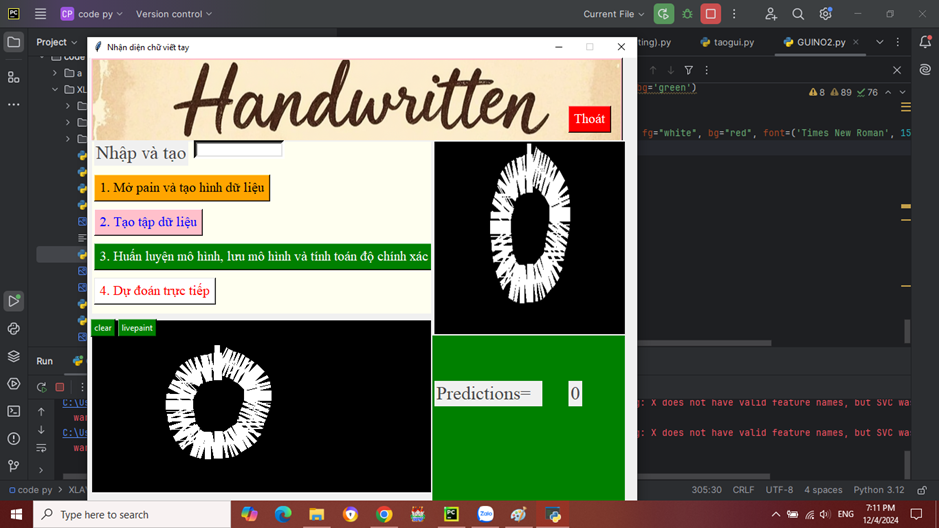
Hình 3.10 Tệp file model

Sau khi phân tích và xử lý file xong thì sẽ cho ra kết quả độ chính xác của file **model** **digit\_recognizer.pkl**

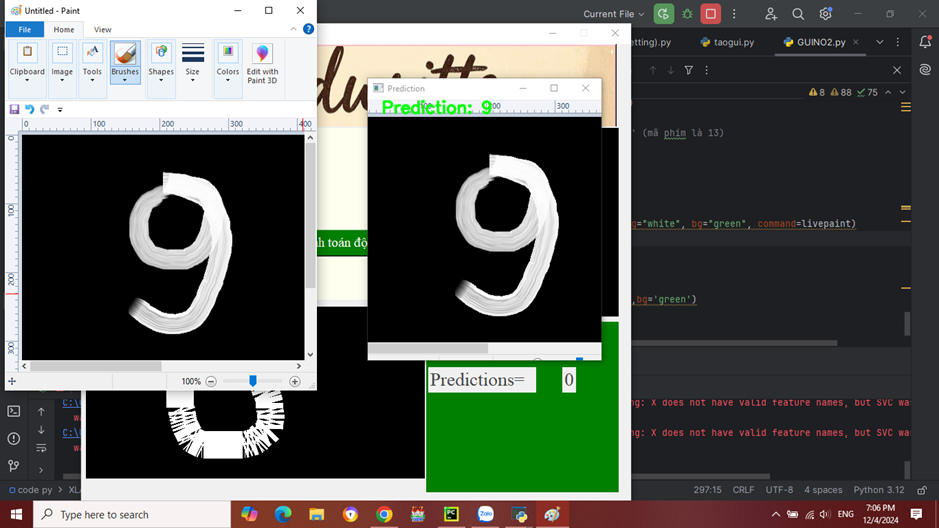


Hình 3.11 Đo độ chính xác

Đến phần chức năng dự đoán và xử lý hình ảnh chúng ta sẽ có hai cách dự đoán, cách đầu tiên chúng ta có thể dự đoán trực tiếp ở ngoài trương trình nhưng cách này sẽ có nhược điểm là độ chính xác của nó sẽ bị giảm đi một chút do đường nét vẽ không được liền mạch

  
Hình 3.12 Dùng thử chức năng dự đoán thứ nhất

Đến cách dự đoán và xử lý ảnh thứ 2 thì lần này chúng ta sẽ dự đoán ở trong **Microsoft Paint** cách này sẽ mang ưu điểm hơn cách trước nó sẽ cho ra độ chính xác cao hơn cách trước vì đường nét sẽ rõ ràng hơn.



Hình 3.13 Dùng thử chức năng dự đoán số 2

Cả 2 bức ảnh trên khi chúng ta vẽ lần lượt số 0 và 9 chương trình sẽ trả lại cho chúng ta nhận diện bức ảnh kết quả là số 0 và số 9

Tương tự chúng ta có thể làm với các chữ số khác.

# **KẾT LUẬN**

**Kết quả đạt được**

**Hoàn thiện quy trình cơ bản của nhận diện chữ viết tay:**

Người dùng có thể:

Tạo dữ liệu chữ số viết tay.

Huấn luyện mô hình nhận diện chữ số.

Lưu mô hình để sử dụng lại.

Thực hiện dự đoán trên dữ liệu vẽ tay.

**Ứng dụng tốt trong học tập và nghiên cứu:**

Phù hợp để thực hành các bước cơ bản của học máy, từ thu thập dữ liệu, tiền xử lý, huấn luyện, đến kiểm tra mô hình.

**Tính tương tác cao:**

Giao diện cung cấp một cách trực quan để hiểu rõ quy trình học máy.

**Ưu điểm của chương trình**

**Tính năng đầy đủ:**

+ Chương trình bao gồm nhiều chức năng, từ việc tạo tập dữ liệu, huấn luyện mô hình, dự đoán, đến sử dụng trực tiếp giao diện vẽ tay.

+ Người dùng có thể thực hiện toàn bộ quy trình xây dựng mô hình nhận diện chữ viết tay từ đầu.

**Giao diện đơn giản, trực quan:**

+ Giao diện sử dụng **Tkinter** giúp người dùng không cần am hiểu sâu về kỹ thuật vẫn có thể tương tác.

+ Có các nút rõ ràng cho từng bước: mở Paint, tạo dữ liệu, huấn luyện, và dự đoán.

**Tạo dữ liệu tự động:**

Chức năng tự động chụp màn hình từ Paint để tạo tập dữ liệu, giảm thiểu thao tác thủ công.

**Huấn luyện và lưu trữ mô hình:**

Chương trình tự động lưu mô hình đã huấn luyện dưới dạng file .pkl, giúp tái sử dụng mà không cần huấn luyện lại.

**Chức năng dự đoán trực tiếp:**

Hỗ trợ dự đoán số liệu trực tiếp từ hình vẽ tay, tiện lợi cho việc kiểm tra mô hình.

**Nhược điểm của chương trình**

**Yêu cầu hệ thống phụ thuộc:**

Chương trình yêu cầu phần mềm bên ngoài (Paint) và một số thư viện, khiến người dùng cần đảm bảo các yếu tố môi trường (như Paint đã cài sẵn trên Windows) để chương trình chạy đúng.

**Chưa tối ưu với dữ liệu ngoài dữ liệu đã thu thập:**

Chương trình chỉ huấn luyện với một tập dữ liệu rất hạn chế từ Paint và có thể gặp khó khăn với các mẫu chữ viết tay không giống với dữ liệu huấn luyện.

**Chạy phụ thuộc vào Paint và máy tính cá nhân:**

Phải mở Paint và vẽ chữ số, do đó chương trình không thực sự linh hoạt cho các trường hợp khác, như nhận diện chữ viết tay từ hình ảnh hoặc từ các nguồn khác.

**Không tối ưu hiệu năng:**

Chương trình đôi khi gây ra sự giật lag , không ổn định

**Quá trình thu thập dữ liệu còn thô sơ:**

Việc chụp ảnh màn hình của Paint chưa hoàn toàn chính xác và có thể gặp lỗi nếu màn hình không được thiết lập đúng như yêu cầu (ví dụ, vị trí Paint hay kích thước cửa sổ không phù hợp).

**HƯỚNG PHÁT TRIỂN**

**Cải thiện giao diện**:

+ Tích hợp vẽ trực tiếp trên Canvas thay vì mở Paint.

+ Sử dụng framework UI hiện đại như PyQt, Kivy để giao diện đẹp hơn.

**Cải tiến mô hình**:

+ Thay SVM bằng Deep Learning (CNN, ResNet).

+ Kết hợp dữ liệu MNIST và dữ liệu thực tế.

**Tích hợp thời gian thực**:

+ Dự đoán chữ số qua camera hoặc thiết bị cảm ứng.

+ Tạo API hỗ trợ nhận diện từ ứng dụng khác.

**Triển khai trên nền tảng khác**:

+ Chuyển thành ứng dụng web (Flask, Django).

+ Phát triển ứng dụng mobile với Flutter hoặc React Native.

**Tăng tính ứng dụng**:

+ Nhận diện thêm chữ cái hoặc từ viết tay.

+ Lưu, chia sẻ kết quả và thống kê lỗi nhận diện.

**Mở rộng cộng đồng**:

+ Đưa mã nguồn lên GitHub, nhận góp ý.

+ Viết tài liệu và tạo hướng dẫn sử dụng chi tiết.

# **TÀI LIỆU THAM KHẢO**

[1] "Nhận diện chữ số viết tay là gì?"

Nguồn: https://machinelearningcoban.com/2018/01/07/handwritingdigitrecognition/, ngày 09/11/2024

[2] "Cơ sở lý thuyết về nhận dạng ký tự quang học (OCR)"

Nguồn: https://viblo.asia/p/co-so-ly-thuyet-ve-nhan-dang-ky-tu-quang-hoc-Az45bDGv5xY, ngày 11/11/2024

[3] L. T. H. Lan, "Slide bài giảng của cô Lương Thị Hồng Lan", ngày 12/11/2024

[4] "Sử dụng mạng CNN trong nhận dạng chữ số viết tay"

Nguồn: https://towardsdatascience.com/handwritten-digit-recognition-using-convolutional-neural-networks-cnn-1d8f45f18e8d, ngày 12/11/2024

[5] "Tập dữ liệu MNIST và các bài toán ứng dụng"

Nguồn: https://yann.lecun.com/exdb/mnist/, ngày 12/11/2024