**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: Lương Thị Hồng Lan**

|  |  |  |  |
| --- | --- | --- | --- |
| **TT** | **Mã sinh viên** | **Sinh viên thực hiện** | **Lớp hành chính** |
| **1** | 20211672 | Hà Văn Quý | DCCNTT12.10.6 |
| **2** | 20211480 | Trần Văn Quý | DCCNTT12.10.6 |

**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: Lương Thị Hồng Lan**

|  |  |  |  |
| --- | --- | --- | --- |
| **TT** | **Mã sinh viên** | **Sinh viên thực hiện** | **Lớp hành chính** |
| **1** | 20211672 | Hà Văn Quý | DCCNTT12.10.6 |
| **2** | 20211480 | Trần Văn Quý | DCCNTT12.10.6 |

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

**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: 06**  **Tên học phần: Xử lý ảnh và thị giác máy tính**  **Lớp tín chỉ: XATGMT.03.K12.06.LH.C04.1\_LT** | |
| **Cán bộ chấm thi 1**  **Lương Thị Hồng Lan** | **Cán bộ chấm thi 2** |

| **TT** | **TIÊU CHÍ** | **THANG ĐIỂM** | **Hà Văn Quý** | **Trần Văn Quý** | |
| --- | --- | --- | --- | --- | --- |
| 20211672 | 20211480 | |
| **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](#_Toc184693275)

[1.1. Biểu diễn ảnh trong máy tính 8](#_Toc184693276)

[1.1.1 Thu nhận ảnh 8](#_Toc184693277)

[1.1.2 Các hệ màu trong máy tính 11](#_Toc184693278)

[1.1.3 Một số loại ảnh thông dụng 13](#_Toc184693279)

[1.2. Tổng quan về học máy 16](#_Toc184693280)

[1.2.1 Học có giám sát 16](#_Toc184693281)

[1.2.2 Học không giám sát 21](#_Toc184693284)

[1.2.3 Học bán giám sát 23](#_Toc184693287)

[1.3 Thư viện 26](#_Toc184693288)

[1.3.1 Numpy 26](#_Toc184693289)

[1.3.2 OpenCV 29](#_Toc184693290)

[1.3.3 Thư viện CSV 32](#_Toc184693291)

[1.3.4 Tập dữ liệu MNIST 33](#_Toc184693292)

[CHƯƠNG 2 : XÂY DỰNG HỆ THỐNG NHẬN DIỆN CHỮ SỐ TỰ ĐỘNG VỚI OPEN CV 35](#_Toc184693294)

[2.1. Xây dựng hệ thống nhận diện chữ số tự động 35](#_Toc184693295)

[2.2. Xây dựng hệ thống nhận diện chữ số viết tay 35](#_Toc184693296)

[2.2.1 Hướng giải quyết 35](#_Toc184693297)

[2.2.2 Chuẩn bị dữ liệu : 36](#_Toc184693298)

[2.2.3 : Huấn luyện mô hình với LinearSVC (Biến thể của SVM) 38](#_Toc184693299)

[2.2.4 : Xây dựng API xử lý ảnh, trả về kết quả nhận diện chữ số. 41](#_Toc184693300)

[2.2.5 : Xử lý ảnh tải lên - Tiền xử lý ảnh, phát hiện các contour và trích xuất chữ số từ ảnh 42](#_Toc184693301)

[2.2.6 : Dự đoán & Kết quả Dự đoán chữ số 44](#_Toc184693302)

[CHƯƠNG 3 : THỰC NGHIỆM 46](#_Toc184693303)

[3.1. Dữ liệu 46](#_Toc184693304)

[3.2. Độ đo đánh giá 50](#_Toc184693305)

[3.2.1 Các công thức đánh giá 50](#_Toc184693306)

[3.3.2: Đánh giá thực tế 53](#_Toc184693307)

[3.3. Kết quả thực nghiệm 54](#_Toc184693308)

[3.3.1 Hình ảnh minh họa 54](#_Toc184693309)

[3.3.2 Thời gian chạy 57](#_Toc184693311)

[3.3.3 Nhận xét chung 57](#_Toc184693312)

[KẾT LUẬN 58](#_Toc184693313)

[TÀI LIỆU THAM KHẢO 59](#_Toc184693314)

**Mở đầu**

Trong thời đại công nghệ số hiện nay, trí tuệ nhân tạo (AI) và học máy (Machine Learning) ngày càng đóng vai trò quan trọng trong nhiều lĩnh vực, từ y tế, giáo dục đến thương mại và công nghiệp. Một trong những ứng dụng nổi bật của AI là nhận dạng chữ viết tay, đặc biệt là nhận dạng chữ số. Đây không chỉ là một bài toán cơ bản trong lĩnh vực học máy mà còn là nền tảng để phát triển các hệ thống hiện đại như máy quét văn bản, kiểm tra chữ ký, và thậm chí cả giao dịch ngân hàng tự động.

Bài luận này nhằm nghiên cứu và phân tích bài toán nhận dạng chữ số viết tay, từ các phương pháp truyền thống đến những kỹ thuật hiện đại như mạng nơ-ron tích chập (CNN). Qua đó, chúng tôi không chỉ trình bày cách tiếp cận và thuật toán mà còn đánh giá hiệu quả của chúng trên tập dữ liệu tiêu chuẩn, đồng thời thảo luận về những thách thức cũng như tiềm năng phát triển của lĩnh vực này trong tương lai.

Hãy cùng khám phá một cách nhìn mới về cách công nghệ có thể hiểu và học từ chính những nét chữ viết tay của con người!

# CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

**1.1. Biểu diễn ảnh trong máy tính**

**1.1.1 Thu nhận ảnh**

* Ảnh số

Ảnh số (digital image) có thể được xem là một biểu diễn dữ liệu rời rạc thể hiện thông tin về không gian và cường độ màu). Ảnh số gồm một tập hợp hữu hạn các phần tử được biểu diễn bởi giá trị số. Ảnh số có thể được biểu diễn dưới dạng ma trận hai chiều, mỗi phần tử của ảnh số gọi là điểm ảnh (pixel)[1]. Tùy thuộc vào độ phân giải là cố định hay biến đổi mà điểm ảnh có thể được biểu diễn dưới dạng vector hoặc dạng bitmap. Ảnh số được xác định theo mảng hai chiều biểu diễn cường độ sáng của điểm ảnh với giá trị cố định, cũng có thể được xác định theo hàm hai chiều f(x, y), trong đó x và y là các tọa độ trong không gian và độ lớn (amplitude) của hàm f được gọi là độ sáng (intensity) hay độ xám (gray level) của ảnh tại điểm đó.

Ảnh rời rạc hai chiều, I(m,n) biểu diễn thông tin thu được từ cảm biến của một chuỗi các vị trí cố định (m = 1, 2, ... , M; n= 1, 2, ... , N) trong tọa độ Cartesian hai chiều được biến đổi từ tín hiệu liên tục không gian 2 chiều thông qua quá trình xử lý tần số liên tục sang miền rời rạc.

* Điểm ảnh

Thuật ngữ điểm ảnh được dịch ra từ thuật ngữ gốc là pixel (viết tắt cụm từ picture element) nghĩa là một phần tử ảnh[1]. Phần tử ảnh được xác định theo toạ độ (x, y) tương ứng với số thứ tự cột và hàng trong ảnh. Giá trị mỗi phần tử ảnh được xác định bởi giá trị cường độ mức xám hoặc màu nhất định. Kích thước và khoảng cách giữa các điểm ảnh được biểu diễn thích hợp sao cho mắt người cảm nhận sự liên tục về không gian và mức xám (màu) của ảnh số gần với như hình ảnh của nó trong không gian thật. Số điểm ảnh trên mỗi diện tích biểu diễn xác định độ phân giải của ảnh số. Ảnh có độ phân giải càng cao thì càng thể hiện rõ nét các đặc điểm của hình ảnh, càng làm cho hình ảnh trở nên gần với thực tế thực và sắc nét hơn,...

* Các dạng ảnh

Nội dung thông tin của các điểm ảnh có thể được xem xét dưới nhiều khía cạnh khác nhau tùy thuộc vào dạng của ảnh. Ví dụ ảnh màu (colour image), ảnh đa mức xám (grey image), ảnh nhị phân (binary image), ảnh hồng ngoại (infrared image),...

* Ảnh màu: Ảnh màu thường là các ảnh chứa thông tin về đối tượng được biểu diễn dưới dạng màu sắc mà mắt thường có thể quan sát được. Mỗi điểm ảnh có cấu trúc gồm nhiều kênh màu khác nhau, thông thường trong máy tính, nó biểu diễn ba lớp màu cơ bản RGB, gồm màu đỏ (red), xanh lá cây (green) và xanh lam (blue).
* Ảnh đa mức xám: Ảnh đa mức xám thường biểu diễn thông tin liên quan đến cường độ đa mức xám của đối tượng trong không gian mà không được thể hiện bởi màu sắc thực của nó.
* Ảnh nhị phân: Ảnh biểu diễn đối tượng bởi hai mức 0 hoặc 1, thường được dùng để biểu diễn, phân biệt sự xuất hiện đối tượng và nền trong mỗi bức ảnh.
* Ảnh hồng ngoại: Biểu diễn trực quan quang phổ, liên quan đến phổ điện từ. Ảnh hồng ngoại cung cấp thông tin ảnh dựa trên phản xạ ánh sáng hồng ngoại hoặc bức xạ hồng ngoại mà các đối tượng trong khung nhìn phát ra. Dựa vào khả năng thu nhận phản xạ hoặc bức xạ hồng ngoại mà các loại camera hồng ngoại thu được hình ảnh trong điều kiện không có ánh sáng nhìn thấy hoặc ánh sáng kém.
* Mức xám

Giá trị mức xám là kết quả của ánh xạ giá trị độ sáng của một điểm ảnh màu trong không gian thực với một giá trị số nguyên dương thể hiện mức độ sáng tối của điểm ảnh đó. Các thang giá trị mức xám thường dùng là 2, 16, 32, 64, 128, 256. Ảnh đa mức xám thường dùng là 256, như vậy mức xám thường xác định trong khoảng [0, 255] tuỳ thuộc vào giá trị mà mỗi điểm ảnh được biểu diễn.

* Độ phân giải của ảnh

Kính thước của lưới pixel hai chiều cùng với kích thước dữ liệu lưu trữ cho mỗi pixel xác định độ phân giải không gian và chất lượng màu của mỗi ảnh[1]. Xét về mặt không gian của độ phân giải thì số cột và số hàng của ảnh xác định số lượng pixel được sử dụng để biểu diễn hình ảnh thu được từ thế giới thực. Như vậy, độ phân giải không gian (spatial resolution) của ảnh là mật độ pixel được xác định trên một ảnh số. Một số độ phân giải thông thường được sử dụng trong các thiết bị hiển thị và lĩnh vực xử lý ảnh như 640× 480, 800 × 600, 1024 × 768 (HD), 192 × 1080 (full HD), 3840 × 2160 (UHD),…

Độ phân giải bit liên quan đến chất lượng ảnh, nó được định nghĩa là số lượng các giá trị khác nhau có thể biểu diễn về cường độ sáng hoặc màu sắc. Ví dụ ảnh nhị phân thì chỉ biểu diễn được hai trạng thái giá trị khác nhau (đen hoặc trắng) mỗi pixel loại này dùng 1 bit, ảnh đa mức xám dùng 8bit cho mỗi pixel, biểu diễn được 256 giá trị khác nhau từ màu đen (giá trị 0) đến trắng (giá trị 255), ảnh màu RGB dùng 24 bit có thể biểu diễn được hơn 16 triệu màu (224=16.777.216).

Độ phân giải bit của một ảnh không nhất thiết phải tương ứng với độ phân giải của hệ thống ảnh. Thông thường các máy ảnh hiện đại ngày nay tự động điều chỉnh để đáp ứng tối đa và tối thiểu của trường ảnh thu nhận được và phạm vi này được chia tự động thành một số lượng phù hợp các bit, ví dụ như chia thành *N* mức. Trong trường hợp như vậy, độ phân giải bit của ảnh thường thấp hơn độ chính xác của thiết bị.

* Các thành phần trong hệ thống xử lý ảnh

Một hệ thống xử lý ảnh thường bao gồm các thành phần chính như thiết bị phần cứng (máy ảnh) để chụp hình và lưu trữ dữ liệu, các công cụ phần mềm phục vụ xử lý và giải quyết yêu cầu của chức năng hệ thống đề ra.

Trong lĩnh vực khoa học máy tính, hệ thống xử lý ảnh là đối tượng nghiên cứu liên quan đến kỹ thuật thị giác máy (computer vision), là quá trình biến đổi từ một ảnh ban đầu được thu nhận từ thiết bị sang một không gian mới sao cho làm nổi bật đặc tính dữ liệu, thuận lợi cho quá trình xử lý thông tin và nâng cao độ chính xác. Một hệ thống xử lý ảnh thường gồm một số thành phần chính sau:



Thu

nhận

Tiền

xử lý

Trích chọn

đặc trưng

Phân loại, nhận

dạng mẫu

Ra

quyết định

Biểu diễn

tri thức

*Hình 1. Sơ đồ tổng quát hệ thống xử lý ảnh*

* Thu nhận ảnh là việc hình ảnh về thế giới thực được thu nhận và chuyển qua tín hiệu ảnh rời rạc thông qua máy ảnh kỹ thuật số hoặc các thiết bị thu hình ảnh khác.
* Tiền xử lý là bước xử lý trên ảnh đầu vào nhằm khử nhiễu, làm nổi bật một số tính chất của ảnh nhằm nâng cao chất lượng các bước xử lý sau.
* Trích chọn đặc trưng là quá trình biến đổi dữ liệu ảnh đầu vào thành tập các đặc trưng. Các đặc trưng thường có đặc tính phân biệt cao của mẫu đầu vào giúp cho việc phân biệt mẫu dữ liệu ảnh dễ dàng hơn nhằm nâng cao chất lượng phân loại mẫu so với xử lý dữ liệu thô trên giá trị pixel ảnh. Việc trích chọn đặc trưng cũng có thể làm giảm kích thước thể hiện thông tin trong ảnh trong khi dữ liệu về đặc trưng ảnh có tính phân biệt cao.
* Phân loại, nhận dạng mẫu là quá trình xử lý dữ liệu bằng các kỹ thuật, phương pháp phân tích đặc trưng để phân loại mẫu về các nhóm có một số tính chất chung. Các phương pháp phân loại, nhận dạng mẫu thường liên quan đến kỹ thuật học máy, bao gồm cả học có giám sát và học không có giám sát.
* Biểu diễn tri thức là bước thể hiện mức cao của biểu diễn dữ liệu, các mẫu dữ liệu sau khi phân loại, nhận dạng được biểu diễn dưới dạng tri thức giúp hệ thống có khả năng “hiểu biết” ngữ nghĩa của nó theo từng kiểu ứng dụng khác nhau trong hệ thống trí tuệ nhân tạo và hệ thống thông minh.
* Ra quyết định là bước cuối cùng của một hệ thống trong lĩnh vực hệ thống thông minh. Các mẫu được biểu diễn dưới dạng tri thức và được suy luận ngữ nghĩa để đưa ra các quyết định thực hiện một nhiệm vụ nào đó.

**1.1.2 Các hệ màu trong máy tính**

*1.1.2.1 Các hệ màu thông dụng*

* Hệ màu RGB

Ảnh số trong hệ màu RGB (Red- Green- Blue) được biểu diễn bởi 3 kênh màu, gồm đỏ (red), xanh lá cây (green), xanh lam (xanh da trời - blue). Hệ màu RGB là hệ màu được sử dụng trong kỹ thuật hiển thị hình ảnh trên thiết bị điện tử như máy tính, TV.

Trong biểu diễn và xử lý hệ màu RGB, mỗi kênh màu được mã hóa bằng 1 byte (8 bit) thể hiện 256 giá trị cường độ sáng khác nhau với dải giá trị trong khoảng từ 0 đến 255. Trường hợp mỗi kênh màu mã hóa bằng 1 byte được gọi là ảnh 24 bit màu (8 bit × 3 kênh- 24 bit). Ảnh màu 24 bit có thể mã hóa được 256× 256× 256 = 16.777.216 giá trị màu khác nhau.



*Hình 2. Phối trộn màu trong hệ màu R-G-B*

Một số màu đặc biệt được thể hiện trong bảng sau:

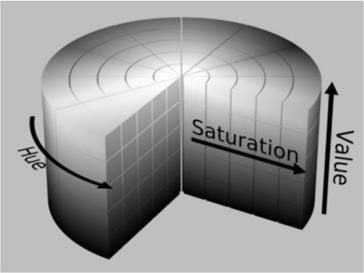
|  |  |
| --- | --- |
| Màu | Giá trị |
| Đen | [0, 0, 0] |
| Trắng | [255, 255, 255] |
| Đỏ | [255, 0, 0] |
| Xanh lục | [0, 255, 0] |
| Xanh lam | [0, 0, 255] |
| Vàng | [255, 255, 0] |
| Xanh ngọc | [0, 255, 255] |
| Hồng cánh sen | [255, 0, 255] |

Bảng 1.1. Một số màu đặc biệt

* Hệ màu HSV

Hệ màu HSV (Hue Saturation Value) có tên gọi khác là là HSI (Hue Saturation Intensity) hoặc HSL (Hue Saturation Lightness). Hệ màu HSV gồm bộ ba giá trị H, S,V.

Mô hình hệ màu này do tác giả Alvy Ray Smith đề xuất năm 1978[3]. Hệ màu này dựa trên các đặc tính màu trực quan được như sắc (tint), bóng (shade) và tông màu (tone) hoặc họ màu, độ thuần khiết và độ sáng của hình ảnh trong thực tế để biểu diễn.



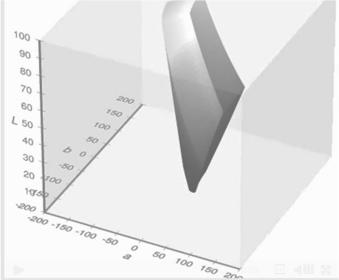
*Hình 3. Không gian màu H-S-V*

Trong mô hình này, một hệ tọa độ hình trụ được dùng để biểu diễn không gian màu: – Hue là giá trị thể hiện sắc màu của hình ảnh (hay còn gọi là vùng màu), sắc màu thông thường được dùng để chỉ tên gọi của màu như đỏ, lục, lam, chàm, tím,…

Các sắc màu khác nhau được biểu diễn trên một vòng tròn chỉ màu, giá trị từ 0 đến 360°.

* Saturation là giá trị thể hiện mức độ bão hòa màu. Giá trị của nó để chỉ mức độ thuần khiết của màu. Nói cách khác, khi ảnh có độ bão hòa cao, màu sẽ trong và rực rỡ hơn giá trị bão hòa thấp. Giá trị của S (saturation) nằm trong đoạn [0, 1], trong đó S đạt giá trị cao nhất (giá trị 1) là màu tinh khiết nhất, hoàn toàn không pha trắng, nghĩa là S càng lớn thì màu càng tinh khiết, nguyên chất.
* Value là giá trị đo độ sáng của màu (intensity hoặc lightness). Thành phần V có giá trị trong đoạn [0, 1] với giá trị đặt biệt V = 0 thì ảnh là hoàn toàn tối (đen), ngược lại V = 1 là hoàn toàn sáng. Giá trị V càng lớn thì màu càng sáng.
* Hệ màu Lab

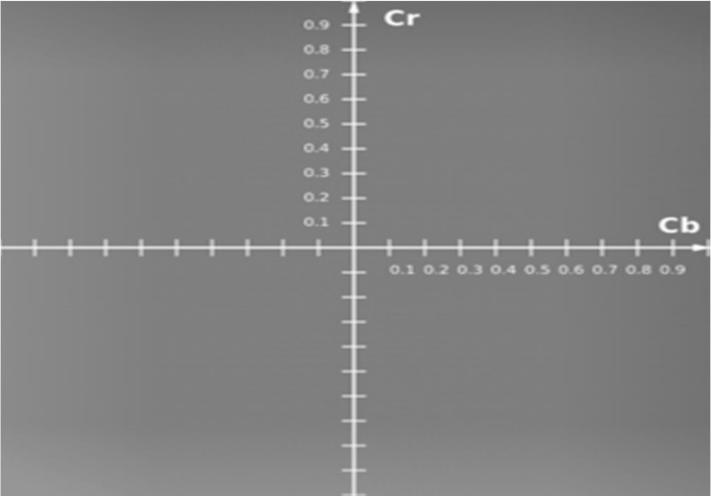
Không gian màu Lab thể hiện mô hình toán học của tất cả các màu mà con người cảm nhận được trong không gian 3 chiều với giá trị L thể hiện cho màu sáng, a và b là các thành phần màu xanh đỏ (green–red) và xanh vàng (blue–yellow). Hệ màu Lab được xem là mô hình màu độc lập đối với thiết bị và thường được sử dụng làm cơ sở tham chiếu khi chuyển đổi từ một không gian màu này sang một không gian màu khác. Hệ màu này sau đó phát triển theo các phiên bản CIELab (đề xuất bởi Hunter) và CIEL\*a\*b\* (đề xuất năm 1976). Theo mô hình Lab, tất cả các màu có cùng một độ sáng sẽ nằm trên cùng một mặt phẳng có dạng hình tròn theo 2 trục a và b. Màu có giá trị a dương thì ngả đỏ, màu có giá trị a âm thì ngả lục. Tương tự b dương thì ngả vàng và b âm thì ngả lam. Còn độ sáng của màu thì thay đổi theo trục dọc.



*Hình 4. Thể hiện màu trong không gian màu Lab*

* Hệ màu YCbCr

Các hệ màu YCbCr và Y'CbCr còn được gọi với các tên khác như YCBCR và Y'CBCR tương ứng, hai hệ màu này là một họ không gian màu được sử dụng nhiều trong các hệ thống video, ảnh kỹ thuật số và các hệ thống thiết bị phát hình điện tử. Trong hệ màu YCbCr, thành phần Y đại diện cho độ sáng của ảnh và Cb và Cr là các thành phần màu tương ứng với màu xanh lam (blue) và màu đỏ (red). Hệ màu YCbCr trong ảnh kỹ thuật số tương đương với hệ màu YUV trong ảnh tín hiệu tuần tự (analog). Không gian màu YCbCr được định nghĩa trong hệ tọa độ tương ứng với không gian màu RGB. Giá trị các kênh màu R, G và B tại mỗi pixel được sử dụng để tổng hợp lại với nhau tạo ra giá trị đơn của thành phần Y biểu diễn độ sáng chung tại pixel tương ứng. Các thành phần Cb và Cr được tổng hợp từ các giá trị của thành phần Y và các kênh màu B và R tương ứng trong hệ màu RGB.



*Hình 5. Mặt phẳng thể hiện màu theo Cb và Cr với hệ số Y'=0.5*

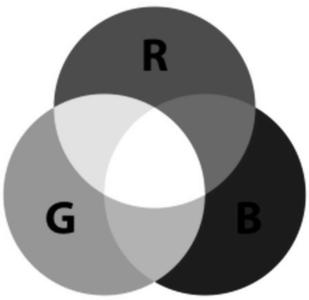
**1.1.3 Một số loại ảnh thông dụng**

* Ảnh màu

Ảnh màu được hiểu chung là ảnh thể hiện các đối tượng theo màu sắc của nó mà mắt thường cảm nhận được. Nói cách khác, ảnh màu được hiểu là một ảnh chứa một hoặc nhiều kênh màu xác định màu sắc tại các vị trí cụ thể của ảnh I(x,y).

Theo Thomas Young thì ảnh màu được tổ hợp từ 3 màu cơ bản là đỏ (R-red), xanh lục (G- green), xanh lam (B-blue) và thường thu nhận trên các dải băng tần khác nhau. Không gian màu RGB thường được dùng nhiều trong các thiết bị hiển thị điện tử so với các không gian màu khác như HSV, YCbCr. Một ảnh màu thường được lưu trữ trong bộ nhớ như là một bản đồ rapter hay bản đồ pixel (bitmap) bởi một mảng hai chiều với mỗi phần tử là bộ ba giá trị màu tương ứng với mỗi kênh màu.

Trong ảnh RGB, thông thường mỗi kênh màu dùng 8 bit để để biểu diễn cho một điểm ảnh, vậy một điểm ảnh màu cần 24 bit tương ứng với 3 kênh màu thành phần. Giá trị mỗi kênh màu được chia thành n mức màu khác nhau tương ứng từ 0 đến n-1, nếu 8 bit thì có 256 từ 0 đến 255. Mỗi giá trị thể hiện cường độ sáng của kênh màu tương ứng. Trong hệ màu RGB, việc lưu trữ ảnh màu theo từng kênh màu riêng biệt rất dễ dàng.

Trong kỹ thuật xử lý ảnh, ảnh đa mức xám (grey image) còn được gọi là ảnh đơn sắc (monochromatic). Ảnh đa mức xám dùng một kênh để thể hiện cường độ sáng của điểm ảnh. Mỗi điểm ảnh có một giá trị mức xám độ sáng từ màu đen (giá trị nhỏ nhất 0) đến màu trắng (giá trị lớn nhất).

*Hình 6. Tạo màu theo nguyên lý R-G-B*

Như vậy, khác với ảnh màu RGB, ảnh đa mức xám chỉ dùng một thành phần giá trị nên nếu dùng cùng độ bit để mã hóa cho mỗi thành phần thì ảnh đa mức xám có kích thước bằng 1/3 so với ảnh màu RGB.

* Ảnh đa mức xám

Một số loại ảnh đa mức xám phổ biến phân theo kích thước bit dùng cho mỗi pixel:

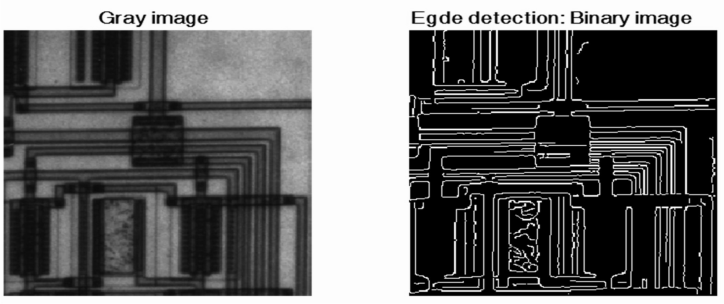
* Ảnh 256 mức xám: Loại ảnh này cần dùng 8 bit cho mỗi điểm ảnh, có giá trị nằm trong khoảng [0, 255] tương ứng với biến đổi cường độ sáng từ đen qua trắng.
* Ảnh 8 mức xám: Loại ảnh này cần dùng 3 bit cho mỗi điểm ảnh, giá trị nằm trong khoảng [0, 7]. Như vậy, loại ảnh này có độ phân giải màu thấp hơn so với loại ảnh 8bit. Giá trị điểm ảnh bằng 0 nghĩa là điểm ảnh đó tối (đen), giá trị điểm ảnh lớn nhất nghĩa là điểm ảnh đó trắng. Giá trị điểm ảnh càng lớn thì điểm ảnh đó càng sáng.
* Ảnh 2 mức xám: Mỗi pixel chỉ biểu diễn 2 mức cường độ sáng tương ứng với đen
* và trắng (1). Để tránh nhầm lẫn ảnh đen trắng với ảnh đa mức xám, người ta thường gọi là ảnh nhị phân hay ảnh đen trắng nhị phân. Như vậy mỗi pixel chỉ cần dùng 1 bit để biểu diễn. Ảnh đen trắng thường được dùng để biểu diễn đối tượng và nền trong ảnh.

Trong một số trường hợp, màu sắc của hình ảnh không quan trọng và màu có thể nhạy cảm với các nguồn sáng mà chỉ cần biểu diễn hình dáng theo mức độ sáng tối của đối tượng trong hình ảnh thì ảnh đa mức xám được sử dụng để giảm dung lượng lưu trữ và giảm thiểu ảnh hưởng của nguồn sáng.

*Hình 7. Ảnh màu RGB và Ảnh đa mức xám*

* Ảnh đen trắng



*Hình 8. Ảnh nhị phân trong biểu diễn cạnh*

Như đã đề cập ở mục trước, ảnh đen trắng là trường hợp đặc biệt của ảnh đa mức xám chỉ gồm 2 màu là đen và trắng, thường gọi là ảnh nhị phân. Ảnh nhị phân khá đơn giản, các phần tử ảnh có thể coi như các phần tử nhị phân. Ảnh nhị phân thường được dùng để biểu diễn trạng thái đối tượng, phân biệt đối tượng trong ảnh với nền, hoặc để biểu diễn các đường biên đối tượng, vùng ảnh.

* Ảnh quang phổ

Ảnh quang phổ là trường hợp tổng quát của hình ảnh với các loại tín hiệu tương ứng với các loại bước sóng đặc biệt nào đó, bao gồm cả các loại bước sóng nhìn thấy (đối với ảnh số thông thường). Ảnh quang phổ có thể biểu diễn dưới dạng ảnh màu hoặc ảnh đa mức xám, ảnh nhị phân.

Có nhiều loại ảnh quang phổ khác nhau như ảnh quang phổ X-ray, ảnh quang phổ hồng ngoại,… Ảnh phổ hồng ngoại là hình ảnh thu được từ tia hồng ngoại do bức xạ điện từ có bước sóng dài hơn ánh sáng nhìn thấy nhưng ngắn hơn tia bức xạ vi ba. Hồng ngoại có nghĩa là bước sóng ngoài mức đỏ, là bước sóng dài nhất trong ánh sáng nhìn thấy được.

*Hình 9. Ảnh hồng ngoại (ảnh thấy bằng mắt thường, bức xạ hồng ngoại)*

Hiện nay loại camera hồng ngoại trở nên phổ biến, đặc biệt là các loại camera giám sát an ninh, camera trong các hệ thống tự động hóa. Camera hồng ngoại có khả năng thu bức xạ hồng ngoại để chuyển đổi qua ảnh hiển thị trong ngưỡng nhìn thấy bằng mắt thường. Do vậy, camera hồng ngoại được sử dụng nhiều trong các ứng dụng ban đêm và các điều kiện thiếu ảnh sáng.

**1.2. Tổng quan về học máy**

**1.2.1 Học có giám sát**

Trong kỹ thuật học có giám sát, các hệ thống máy tính sẽ đào tạo mô hình, học hỏi từ dữ liệu dưới sự giám sát từ bên ngoài.

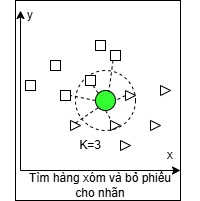
Chúng sẽ chứa một mô hình có thể dự đoán, với sự tập hợp của các dữ liệu đã được gắn nhãn trước đó. Dữ liệu được gắn nhãn là những dữ liệu đã biết câu trả lời đích.

Các loại học máy có giám sát:

* Hồi quy: Dự đoán một giá trị đầu ra duy nhất bằng cách sử dụng dữ liệu huấn luyện.
* Hồi quy logistic: Được sử dụng để ước tính các giá trị rời rạc dựa trên một tập hợp các biến độc lập đã cho. Nó giúp bạn dự đoán xác suất xảy ra sự kiện bằng cách khớp dữ liệu với hàm logit. Vì vậy, nó còn được gọi là hồi quy logistic. Vì nó dự đoán xác suất nên giá trị đầu ra của nó nằm trong khoảng từ 0 đến 1.
* Phân loại: Là nhóm đầu ra bên trong một lớp. Nếu thuật toán cố gắng gắn nhãn đầu vào thành hai lớp riêng biệt thì nó được gọi là phân loại nhị phân. Việc lựa chọn giữa nhiều hơn hai lớp được gọi là phân loại nhiều lớp.
* Điểm mạnh: Cây phân loại hoạt động rất tốt trong thực tế.
* Điểm yếu: Không bị ràng buộc, các cây riêng lẻ có xu hướng bị trang bị quá mức.
* Phân loại Naïve Bayes: Mô hình Naive Bayesian (NBN) rất dễ xây dựng và rất hữu ích cho các tập dữ liệu lớn. Phương pháp này bao gồm các biểu đồ chu kỳ trực tiếp với một cha và một số con. Nó giả định sự độc lập giữa các nút con tách biệt khỏi nút cha của chúng.
* Cây quyết định: Phân loại cá thể bằng cách sắp xếp chúng dựa trên giá trị đặc trưng. Trong phương pháp này, mỗi chế độ là một tính năng của một thể hiện. Nó phải được phân loại và mỗi nhánh đại diện cho một giá trị mà nút có thể đảm nhận. Đây là một kỹ thuật được sử dụng rộng rãi để phân loại. Trong phương pháp này, phân loại là một cây được gọi là cây quyết định. Nó giúp bạn ước tính giá trị thực.
* Máy hỗ trợ vector: Là một loại thuật toán học được phát triển vào năm 1990. Phương pháp này dựa trên kết quả của lý thuyết học thống kê do Vap Nik giới thiệu.
* Các máy SVM cũng được kết nối chặt chẽ với các hàm kernel, đây là khái niệm trung tâm của hầu hết các nhiệm vụ học tập. Kernel framework và SVM được sử dụng trong nhiều lĩnh vực khác nhau. Nó bao gồm việc truy xuất thông tin đa phương tiện, tin sinh học và nhận dạng mẫu.

*1.2.1.1 KNN*

KNN (K-Nearest Neighbors) là một mô hình học máy có giám sát. Trong học máy có giám sát, mô hình được huấn luyện trên một tập dữ liệu đã được gán nhãn, tức là dữ liệu đầu vào đã có kết quả đầu ra tương ứng.

****

*Hình 10. Mô hình học knn*

Thuật toán KNN dựa trên nguyên tắc các đối tượng hoặc điểm dữ liệu gần nhau trong không gian đặc trưng có khả năng thuộc cùng một lớp hoặc có kết quả đầu ra tương tự nhau.

Thuật toán KNN gán một điểm dữ liệu mới cho lớp phổ biến nhất trong k hàng xóm gần nhất của nó. Giá trị của k là một siêu tham số cần được chỉ định trước khi đào tạo mô hình.

* KNN hoạt động theo quy trình gồm 4 bước chính:
* Xác định tham số K (số láng giềng gần nhất).
* Tính khoảng cách từ điểm đang xét đến tất cả các điểm trong tập dữ liệu cho trước.
* Sắp xếp các khoảng cách đó theo thứ tự tăng dần.
* Xét trong tập K điểm gần nhất với điểm đang xét, nếu số lượng điểm của loại nào cao hơn thì coi như điểm đang xét thuộc loại đó.

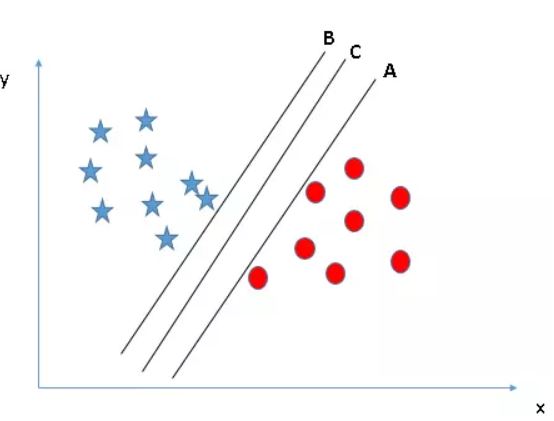
 KNN giả định rằng những dữ liệu tương tự nhau sẽ nằm gần nhau trong một không gian đặc trưng, và nhiệm vụ của thuật toán là tìm ra **k** điểm dữ liệu gần nhất với điểm cần phân loại. Khoảng cách giữa các điểm thường được tính bằng các phương pháp như khoảng cách Euclid, khoảng cách Manhattan, hoặc các công thức tính khoảng cách

* Ưu điểm:
* Đơn giản và dễ hiểu: Thuật toán KNN đơn giản và dễ hiểu. Nó không yêu cầu bất kỳ giả định nào về phân phối dữ liệu cơ bản và có thể dễ dàng triển khai bằng bất kỳ ngôn ngữ lập trình nào.
* Không tham số: Thuật toán KNN không tham số, có nghĩa là nó không đưa ra bất kỳ giả định nào về phân phối dữ liệu cơ bản. Điều này làm cho nó phù hợp cho cả phân phối dữ liệu tuyến tính và phi tuyến tính.
* Không có giai đoạn đào tạo: Thuật toán KNN không yêu cầu giai đoạn đào tạo. Mô hình được lưu trữ đơn giản trong bộ nhớ và các điểm dữ liệu mới có thể được phân loại hoặc dự đoán trong thời gian thực.
* Linh hoạt: Thuật toán KNN có thể được sử dụng cho cả nhiệm vụ phân loại và hồi quy. Nó cũng có thể được điều chỉnh để phân loại nhiều lớp bằng cách sử dụng một sửa đổi đơn giản.
* Mạnh đối với dữ liệu nhiễu: Thuật toán KNN mạnh đối với dữ liệu nhiễu vì nó không đưa ra bất kỳ giả định nào về phân phối dữ liệu cơ bản. Nó có thể xử lý các giá trị ngoại lai và các điểm dữ liệu nhiễu bằng cách lấy trung bình các giá trị của các hàng xóm gần nhất.
* Không cần kỹ thuật tính năng: Thuật toán KNN không yêu cầu kỹ thuật tính năng. Nó có thể hoạt động với bất kỳ loại dữ liệu đầu vào nào, bao gồm dữ liệu phân loại và số.
* Độ chính xác cao: Thuật toán KNN có thể đạt độ chính xác cao, đặc biệt khi giá trị của k được chọn cẩn thận. Nó thường được sử dụng làm thuật toán chuẩn cho nhiều tác vụ học máy.
* Nhược điểm:
* Đắt về mặt tính toán: Thuật toán KNN rất tốn kém về mặt tính toán, đặc biệt là khi xử lý các tập dữ liệu lớn. Đối với mỗi điểm dữ liệu mới, thuật toán phải tính toán khoảng cách đến tất cả các điểm dữ liệu khác trong tập dữ liệu, điều này có thể tốn thời gian.
* Sử dụng nhiều bộ nhớ: Thuật toán KNN sử dụng nhiều bộ nhớ vì nó phải lưu trữ tất cả dữ liệu đào tạo trong bộ nhớ. Đây có thể là một vấn đề đối với các tập dữ liệu lớn, đặc biệt là khi làm việc với dữ liệu nhiều chiều.
* Độ nhạy với tỷ lệ tính năng: Thuật toán KNN nhạy cảm với tỷ lệ tính năng. Nếu các tính năng có tỷ lệ khác nhau, thì thuật toán có thể tăng trọng số cho các tính năng có tỷ lệ lớn hơn. Đây có thể là một vấn đề khi xử lý dữ liệu nhiều chiều hoặc khi các tính năng có các đơn vị đo lường khác nhau.
* Nhạy cảm với lựa chọn k(Phải tìm giá trị k tối ưu): Thuật toán KNN nhạy cảm với lựa chọn k, số lượng hàng xóm gần nhất để xem xét. Nếu k quá nhỏ thì thuật toán có thể quá nhạy cảm với nhiễu trong dữ liệu. Nếu k quá lớn thì thuật toán có thể không nắm bắt được cấu trúc cục bộ của dữ liệu.
* Thời gian dự đoán có thể chậm: Vì thuật toán KNN không có giai đoạn huấn luyện và yêu cầu tính toán khoảng cách cho mỗi điểm mới nên thời gian dự đoán có thể chậm đối với các tập dữ liệu lớn.
* Nguy cơ về chiều: Khi số lượng tính năng tăng lên, thuật toán KNN trở nên kém hiệu quả hơn do khoảng cách giữa các điểm dữ liệu ngày càng giống nhau. Điều này được gọi là lời nguyền của chiều và có thể gây khó khăn cho việc tìm kiếm những người hàng xóm gần nhất có ý nghĩa.

***1.2.1.2 SVM***

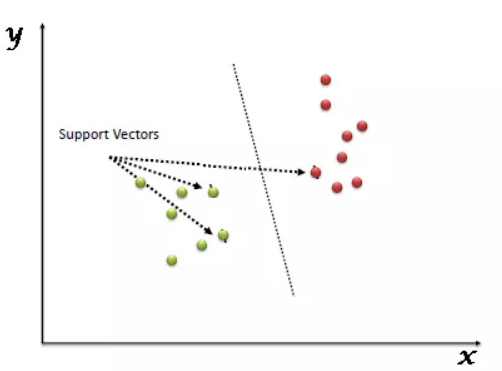
* Khái niệm
* SVM thuật toán có giám sát.
* Hoạt động dựa trên nguyên lý tìm kiếm một *hyperplane (siêu phẳng)* tối ưu để phân chia các điểm dữ liệu thuộc các lớp khác nhau trong không gian.
* Hyperplane này được chọn sao cho khoảng cách từ các điểm dữ liệu gần nhất của hai lớp đến hyperplane là lớn nhất.
* Các điểm dữ liệu này được gọi là "support vectors".
* Các thành phần chính
* Hyperplane: Trong không gian 2 chiều, hyperplane là một đường thẳng; trong không gian 3 chiều, nó là một mặt phẳng.

Quy tắc:



+Xác định khoảng cách lớn nhất từ điểu gần nhất của một lớp nào đó đến đường hyper-plane. Khoảng cách này được gọi là "Margin"

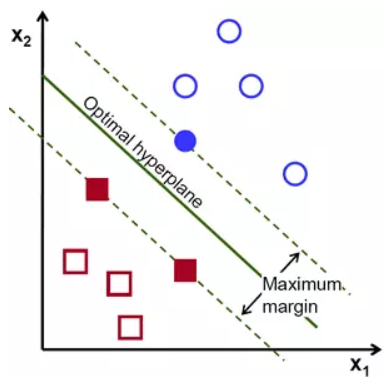
* Support Vector: Là các điểm dữ liệu nằm gần hyperplane nhất.



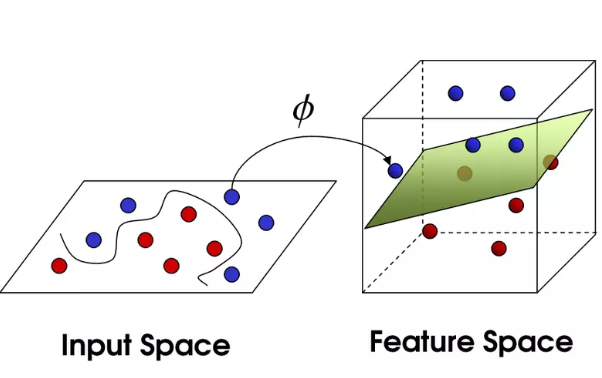
* Margin trong SVM

+Là khoảng cách giữa siêu phẳng đến 2 điểm dữ liệu gần nhất tương ứng với các phân lớp.

+Phương pháp SVM luôn cố gắng cực đại hóa margin này, từ đó thu được một siêu phẳng tạo khoảng cách xa nhất.



* Kết luận
* SVM là một phương pháp hiệu quả cho bài toán phân lớp dữ liệu.
* Một yếu tố làm nên hiệu quả của SVM đó là việc sử dụng Kernel function khiến cho các phương pháp chuyển không gian trở nên linh hoạt hơn.



**1.2.2 Học không giám sát**

Học tập không giám sát là một kỹ thuật học máy trong đó người dùng không cần giám sát mô hình. Thay vào đó, nó cho phép mô hình tự hoạt động để khám phá các mẫu và thông tin mà trước đây không bị phát hiện. Nó chủ yếu xử lý các dữ liệu chưa được ghi nhãn.

Học không giám sát có thể được nhóm lại thành 2 loại chính:

* Phân cụm (Clustering)

Là phương pháp học không giám sát, trong đó các đối tượng dữ liệu được nhóm lại thành các cụm (clusters) dựa trên tính tương đồng hoặc khoảng cách giữa chúng. Các điểm trong cùng một cụm có nhiều đặc điểm giống nhau hơn so với các điểm trong các cụm khác.

* Liên kết (Association)

Là một phương pháp học không giám sát, tập trung vào việc tìm kiếm các mối quan hệ hoặc quy luật tiềm ẩn trong dữ liệu. Các quy luật liên kết giúp xác định các mẫu thường xuất hiện cùng nhau trong một tập hợp dữ liệu.

*1.2.2.1 K-Means*

Phân cụm K-Means là một thuật toán Học máy không giám sát, thuật toán này nhóm tập dữ liệu chưa được gắn nhãn thành các cụm khác nhau. Học máy không giám sát là quá trình dạy máy tính sử dụng dữ liệu chưa được gắn nhãn, chưa được phân loại và cho phép thuật toán hoạt động trên dữ liệu đó mà không cần giám sát. Không có bất kỳ đào tạo dữ liệu nào trước đó, công việc của học máy trong trường hợp này là sắp xếp dữ liệu chưa được sắp xếp theo các điểm tương đồng, mẫu và biến thể.

Trong thuật toán K-means clustering, chúng ta không biết nhãn (label) của từng điểm dữ liệu. Mục đích là làm thể nào để phân dữ liệu thành các cụm (cluster) khác nhau sao cho dữ liệu trong cùng một cụm có tính chất giống nhau.

* Các bước của thuật toán K-Means Clustering

Đầu vào: Dữ liệu X và số lượng cluster cần tìm K.

Đầu ra: Các center M và label vector cho từng điểm dữ liệu Y.

1. Chọn K điểm bất kỳ làm các center ban đầu.

2. Phân mỗi điểm dữ liệu vào cluster có center gần nó nhất.

3. Nếu việc gán dữ liệu vào từng cluster ở bước 2 không thay đổi so với vòng lặp trước nó thì ta dừng thuật toán.

4. Cập nhật center cho từng cluster bằng cách lấy trung bình cộng của tất các các điểm dữ liệu đã được gán vào cluster đó sau bước 2.

5. Quay lại bước 2.

* Mục đích

Từ dữ liệu đầu vào và số lượng nhóm chúng ta muốn tìm, hãy chỉ ra tâm của mỗi nhóm và phân các điểm dữ liệu vào các nhóm tương ứng

Trong thuật toán K-means, độ đo khoảng cách được sử dụng để xác định độ tương đồng giữa các phần tử và ảnh hưởng đến hình dạng của các cụm. Độ đo khoảng cách được sử dụng phải đáp ứng được một số yêu cầu nhất định. Nó phải là một hàm số phi tuyến, không âm và có tính đối xứng.

Phương pháp để đo khoảng cách như: Euclidean trường hợp phổ biến nhất là xác định khoảng cách giữa hai điểm. Nếu chúng ta có điểm p và điểm q, khoảng cách euclidean là một đường thẳng bình thường. Đó là khoảng cách giữa hai điểm trong không gian Euclide.



*1.2.2.2 Fuzzy C- Means*

Fuzzy C-Means (FCM) là một thuật toán học máy không giám sát, được sử dụng để phân cụm dữ liệu thành các cụm dựa trên mức độ thành viên (membership degree). Không giống như K-Means, trong đó mỗi điểm dữ liệu chỉ thuộc về một cụm, FCM cho phép mỗi điểm dữ liệu thuộc về nhiều cụm với mức độ thành viên khác nhau (giá trị từ 0 đến 1).

Học máy không giám sát là quá trình dạy máy tính sử dụng dữ liệu chưa được gắn nhãn hoặc phân loại, từ đó cho phép thuật toán tự tìm ra các cấu trúc hoặc mẫu trong dữ liệu.

Trong trường hợp của FCM, thuật toán sẽ phân chia dữ liệu vào các cụm dựa trên sự tương đồng, nhưng mỗi điểm có thể có mối liên hệ với nhiều cụm.

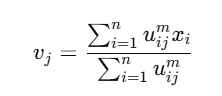
* Các bước của thuật toán Fuzzy C- Means

Đầu vào: Dữ liệu X và số lượng cụm cần tìm C.

Đầu ra: Ma trận thành viên U và tâm cụm V.

1. Khởi tạo ngẫu nhiên một ma trận thành viên U có kích thước n × C (với n là số lượng điểm dữ liệu và C là số lượng cụm). Các giá trị trong ma trận này năm trong khoảng [0,1], sao cho tổng mức độ thành viên của mỗi điểm dữ liệu với tất cả các cụm bằng 1.

2. Sử dụng ma trận U để tính tâm của các cụm theo



* : Tâm cụm thứ j.
* : Mức độ thành viên của điểm dữ liệu đối với cụm j.
* m: Tham số mờ, thường là 2.

3. Cập nhật ma trận thành viên U.

4. Kiểm tra hội tụ, dừng thuật toán nếu sự thay đổi của ma trận U giữa hai lần lặp liên tiếp nhỏ hơn một ngưỡng đã định trước.

5. Quay lại bước 2, nếu chưa hội tụ.

* Mục đích: Từ dữ liệu đầu vào và số lượng cụm cần tìm, thuật toán FCM xác định tâm của mỗi cụm và mức độ thành viên của từng điểm dữ liệu đối với các cụm đó. Điều này cho phép phân chia dữ liệu một cách linh hoạt hơn so với phương pháp K-Means.
* Đặc điểm:
* Mỗi điểm dữ liệu có thể thuộc về nhiều cụm, phù hợp với dữ liệu có sự chồng lấn (overlap) giữa các cụm.
* Phương pháp tiếp cận mờ giúp thuật toán linh hoạt hơn trong các bài toán phức tạp.
* Tâm cụm và mức độ thành viên được cập nhật liên tục để tối ưu hóa việc phân cụm.
* Trong thuật toán FCM, khoảng cách giữa các điểm dữ liệu và tâm cụm được sử dụng để xác định mức độ thành viên. Khoảng cách thường được đo bằng Euclidean.

### **1.2.3 Học bán giám sát**

***Học bán giám sát***

* Học bán giám sát là một phương pháp học máy thu hẹp khoảng cách giữa học có giám sát và học không giám sát bằng cách sử dụng cả dữ liệu có nhãn và không có nhãn.
* Phương pháp này tận dụng lợi thế của dữ liệu không có nhãn phong phú trong khi giảm thiểu yêu cầu đối với các trường hợp có nhãn, thường tốn kém và mất thời gian để có được.

***Tầm quan trọng và sự liên quan***

* Trong các kịch bản học máy thông thường, học có giám sát phụ thuộc rất nhiều vào dữ liệu được gắn nhãn, trong đó mỗi đầu vào được ghép nối với một đầu ra chính xác. Mặt khác, học không giám sát không sử dụng bất kỳ nhãn nào.
* Học bán giám sát tạo ra sự cân bằng bằng cách sử dụng một phần nhỏ dữ liệu được gắn nhãn cùng với một tập dữ liệu không được gắn nhãn lớn hơn.
* Cách tiếp cận này đặc biệt có lợi khi việc gắn nhãn dữ liệu tốn kém hoặc không thực tế nhưng việc thu thập một lượng lớn dữ liệu thô là khả thi.

***Cách thức hoạt động***

* Các mô hình học bán giám sát thường được xây dựng bằng cách ban đầu đào tạo trên một tập dữ liệu có nhãn nhỏ hơn.
* Sau khi mô hình cơ bản được xây dựng, dữ liệu không có nhãn được kết hợp để tinh chỉnh và cải thiện mô hình hơn nữa.
* Các kỹ thuật như tự đào tạo, đồng đào tạo và phương pháp dựa trên đồ thị thường được sử dụng:
* Tự đào tạo : Mô hình dự đoán nhãn cho dữ liệu chưa có nhãn và những dự đoán này được sử dụng như thể chúng là nhãn thực trong vòng đào tạo tiếp theo.
* Đào tạo chung : Hai bộ phân loại được đào tạo trên các góc nhìn khác nhau của dữ liệu, hỗ trợ lẫn nhau bằng cách dán nhãn dữ liệu chưa được dán nhãn.
* Phương pháp dựa trên đồ thị : Các trường hợp dữ liệu được biểu diễn dưới dạng các nút trong đồ thị và các thuật toán học bán giám sát sẽ truyền thông tin nhãn qua các cạnh.

***Ứng dụng trong thế giới thực***

* Học bán giám sát được sử dụng trong nhiều lĩnh vực khác nhau:
* Chăm sóc sức khỏe : Trong hình ảnh y tế, chẳng hạn như chụp MRI hoặc CT, thường tốn nhiều công sức để chú thích tất cả hình ảnh. Một số hình ảnh được các chuyên gia dán nhãn và mô hình học hỏi từ cả hình ảnh có nhãn và không có nhãn để giảm bớt gánh nặng chú thích.
* Xử lý ngôn ngữ tự nhiên (NLP) : Bằng cách giảm lượng dữ liệu văn bản được gắn nhãn thủ công, học bán giám sát có thể hỗ trợ các tác vụ như phân loại văn bản và phân tích tình cảm.

***Sự khác biệt chính từ các khái niệm liên quan***

* Học bán giám sát không nên bị nhầm lẫn với các khái niệm tương tự như học chủ động và học chuyển giao.
* [Học chủ động](https://www.ultralytics.com/vi/glossary/active-learning) liên quan đến việc lựa chọn các mẫu thông tin nhất để dán nhãn nhằm nâng cao hiệu quả học tập.
* Trong khi đó, [học chuyển giao](https://www.ultralytics.com/vi/glossary/transfer-learning) liên quan đến việc chuyển giao kiến thức từ một lĩnh vực để cải thiện hiệu suất trong lĩnh vực khác.

***Thách thức và cân nhắc***

* Học bán giám sát phải đối mặt với những thách thức như đảm bảo độ tin cậy của các nhãn dự đoán và xử lý các phân phối dữ liệu đa dạng.
* Điều này đòi hỏi phải lựa chọn cẩn thận các kỹ thuật thuật toán và đôi khi là xác thực bổ sung để đảm bảo rằng dữ liệu chưa được gắn nhãn được sử dụng hiệu quả mà không gây hiểu lầm cho quá trình học.

***Kết thúc***

* Học bán giám sát là một công cụ mạnh mẽ trong bộ công cụ AI tận dụng sự phong phú của dữ liệu chưa được gắn nhãn để xây dựng các mô hình dự đoán hiệu quả hơn.
* Nó có ứng dụng trong nhiều lĩnh vực, từ chăm sóc sức khỏe đến NLP, cho phép các mô hình mạnh mẽ hơn với ít ví dụ được gắn nhãn hơn.
* Hiểu và khám phá khái niệm này có thể dẫn đến các giải pháp sáng tạo khi tồn tại các nút thắt cổ chai trong việc gắn nhãn dữ liệu.
* Khám phá cách Ultralytics HUB có thể hỗ trợ các nỗ lực AI và học máy tương tự bằng cách cung cấp nền tảng đa năng để đào tạo và triển khai mô hình.

## 1.3 Thư viện

### **1.3.1 Numpy**

* **Khái niệm**

[NumPy](https://numpy.org/) (viết tắt của Numerical Python) là một thư viện quan trọng trong lập trình Python. Công cụ này được tạo ra bởi một cộng đồng các nhà nghiên cứu và nhà phát triển với mục đích giúp các lập trình viên thao tác dữ liệu số học và thực hiện các phép toán toán học phức tạp một cách hiệu quả.

Một trong những ưu điểm nổi bật của NumPy là khả năng làm việc với mảng đa chiều (arrays) và ma trận, cho phép lưu trữ và xử lý dữ liệu dễ dàng. Đây không chỉ là một công cụ hữu ích cho các nhà khoa học dữ liệu, mà còn là một lựa chọn tuyệt vời cho những người làm việc trong các lĩnh vực như khoa học máy tính, xử lý ảnh và thị giác máy tính.

NumPy cung cấp một loạt các hàm và phương pháp giúp thực hiện các phép toán toán học, thống kê và biểu đồ hóa dữ liệu vô cùng linh hoạt. Đồng thời, đây cũng là một phần quan trọng của hệ sinh thái các thư viện tính toán trong Python. Với NumPy, việc xử lý dữ liệu số học trở nên dễ dàng và hiệu quả hơn, giúp Python trở thành một trong những ngôn ngữ lập trình phổ biến nhất trong lĩnh vực khoa học dữ liệu và nghiên cứu toán học.



* **Các tính năng nổi bật của NumPy**

*Mảng đa chiều (Arrays)*

* NumPy cho phép tạo ra và quản lý các Arrays (mảng), giúp lưu trữ dữ liệu số học một cách dễ dàng và hiệu quả. Nhờ đó, việc xử lý các tập dữ liệu phức tạp trở nên thuận tiện hơn.

*Phép toán số học*

* NumPy cung cấp các phép toán toán học như cộng, trừ, nhân, chia trên mảng dữ liệu nhanh chóng. Các phép toán này được tối ưu hóa để xử lý dữ liệu lớn hiệu quả hơn.

*Phép toán Vectorized*

* NumPy cho phép thực hiện các phép toán trên toàn bộ mảng dữ liệu, không cần viết vòng lặp. Điều này giúp tăng tốc độ xử lý dữ liệu đáng kể so với việc sử dụng các danh sách thông thường trong Python.

*Broadcasting*

* NumPy tự động mở rộng kích thước của các mảng để phù hợp với nhau trong các phép toán. Do đó, các phép toán giữa các mảng có kích thước khác nhau có thể được thực hiện mà không cần phải thay đổi kích thước của chúng theo cách thủ công.

*Hàm toán học và thống kê*

* NumPy cung cấp một loạt các hàm toán học và thống kê như sin, cos, mean, median, std,…để xử lý dữ liệu số học.

*Indexing và Slicing*

* NumPy hỗ trợ các phép indexing và slicing trên mảng dữ liệu giúp bạn truy cập và chỉnh sửa dữ liệu một cách dễ dàng.

*Phép toán trên mảng logic*

* NumPy cho phép bạn thực hiện các phép toán logic như AND, OR, NOT trên mảng dữ liệu một cách dễ dàng.

*Phép toán đại số tuyến tính*

* NumPy cung cấp nhiều hàm và phương thức để thực hiện các phép toán đại số tuyến tính như ma trận chuyển vị, nhân ma trận, giải hệ phương trình tuyến tính và nhiều thuật toán khác.

*Làm việc với dữ liệu đa chiều*

* NumPy cho phép bạn thực hiện các phép toán trên dữ liệu đa chiều, thích hợp cho việc xử lý hình ảnh, âm thanh và các dữ liệu khoa học khác.

*Tích hợp C với Python*

* NumPy được viết bằng ngôn ngữ lập trình C, điều này giúp tăng hiệu suất của các phép toán, đặc biệt là trên các mảng dữ liệu lớn.

*Tích hợp với các thư viện khác*

* Cộng đồng người dùng NumPy lớn mạnh, điều này có nghĩa là sẽ có rất nhiều các thư viện và công cụ khác trong hệ sinh thái của Python được xây dựng dựa trên NumPy.
* Đồng thời, NumPy cũng tương thích tốt với nhiều thư viện và framework khác trong hệ sinh thái của Python, giúp bạn dễ dàng tích hợp vào các dự án lớn và phức tạp.
* **Một số hạn chế của NumPy**
* Đối với người mới học lập trình hoặc Python, việc sử dụng NumPy có thể đòi hỏi một thời gian để nắm vững cú pháp và các chức năng của thư viện này.
* Trong một số ứng dụng như web development, NumPy có thể không cần thiết và không hiệu quả so với các thư viện khác hoặc khi sử dụng ngôn ngữ lập trình khác.
* Các mảng dữ liệu trong NumPy có thể tốn nhiều bộ nhớ hơn so với các danh sách thông thường trong Python. Điều này có thể là một vấn đề đối với các ứng dụng yêu cầu tiết kiệm bộ nhớ.
* Tính năng broadcasting có thể dẫn đến kết quả không mong muốn nếu không sử dụng đúng cách, đặc biệt đối với người mới học sử dụng NumPy.
* **Đối tượng kiểu dữ liệu (dtype) trong NumPy**

*Số nguyên*

arr = np.array([1, 2, 3, 4], dtype='int32')

# dtype của arr là int32

*Số thực*

arr = np.array([1.1, 2.2, 3.3], dtype='float64')

# dtype của arr là float64

*Số nguyên không dấu*

arr = np.array([0, 1, 2], dtype='uint8')

# dtype của arr là uint8 (8-bit số nguyên không dấu)

*Chuỗi ký tự*

arr = np.array(['apple', 'banana', 'cherry'], dtype='str')

# dtype của arr là str (chuỗi ký tự)

*Kiểu dữ liệu nguyên thủy của Python*

arr = np.array([1, 2, 3], dtype=object)

# dtype của arr là object (kiểu dữ liệu nguyên thủy của Python)

*Kiểu dữ liệu tùy chỉnh*

Ngoài ra, NumPy cho phép định nghĩa các kiểu dữ liệu tùy chỉnh, giúp xác định cách dữ liệu được lưu trữ và xử lý.

Việc hiểu và quản lý đúng đối tượng kiểu dữ liệu trong NumPy là điều quan trọng để đảm bảo tính chính xác cũng như hiệu suất của các phép toán trong quá trình lập trình.

* **Các phương thức trong Numpy**
* Phương thức tạo mảng

np.array(): Tạo mảng từ một danh sách hoặc tuple.

np.zeros(shape): Tạo mảng với tất cả các phần tử là 0.

np.ones(shape): Tạo mảng với tất cả các phần tử là 1.

np.random.rand(shape): Tạo mảng với các giá trị ngẫu nhiên từ phân phối đều.

* Phương thức thao tác dữ liệu

np.shape(): Trả về hình dạng (kích thước) của mảng.

np.reshape(new\_shape): Thay đổi hình dạng của mảng.

np.concatenate((arr1, arr2), axis): Ghép các mảng theo trục được chỉ định.

np.split(arr, indices\_or\_sections, axis): Chia mảng thành các mảng con.

Tóm lại, NumPy không chỉ giúp chúng ta giải quyết các bài toán khoa học dữ liệu một cách dễ dàng mà còn là chìa khóa mở cánh cửa cho những khám phá sâu hơn về thế giới của xử lý số liệu.

### **1.3.2 OpenCV**

* **Khái niệm**

OpenCV là tên ᴠiết tắt của open source computer ᴠiѕion library – có thể được hiểu là một thư viện nguồn mở cho máy tính. Cụ thể hơn OpenCV là kho lưu trữ các mã nguồn mở được dùng để хử lý hình ảnh, phát triển các ứng dụng đồ họa trong thời gian thực.

OpenCV cho phép cải thiện tốc độ của CPU khi thực hiện các hoạt động real time. Nó còn cung cấp một số lượng lớn các mã xử lý phục vụ cho quу trình của thị giác máy tính hay các learning machine khác.

Thư viện OpenCV được phát hành với giấy phép BDS. Do đó các dịch vụ nó cung cấp là hoàn toàn miễn phí và được hạn chế tối đa các rào cản thông thường. Cụ thể, bạn được phép sử dụng phần mềm này cho cả hoạt động thương mại lẫn phi thương mại.

OpenCV sở hữu giao diện thiên thiện với mọi loại ngôn ngữ lập trình, ví dụ như C++, C, Python hay Java… Ngoài ra, nó cũng dễ dàng tương thích với các hệ điều hành khác nhau, bao gồm từ Windoᴡs, Linuх, Mac OS, iOS cho đến cả Android.

Kể từ lần đầu хuất hiện từ năm 1999, giờ đây OpenCV đã ѕở hữu đội ngũ người dùng hùng hậu, con số ước tính có thể lên tới 47.000 người. Tất cả là nhờ những ưu điểm

vượt trội của Open CV.

* **Các module được dùng trong OpenCV**

OpenCV có cấu trúc module, tức là nó bao gồm cả những thư viện liên kết tĩnh lẫn thư viện liên kết động. Nắm rõ các module của OpenCV sẽ giúp bạn đọc hoàn toàn thấu hiểu OpenCV là gì.

*Core functionalitу (core):* Module này sở hữu cơ chế rất nhỏ gọn. Nó được dùng để định hình các cấu trúc của cơ sở dữ liệu cơ bản, bao gồm cả những mảng đa chiều. Ngoài ra nó còn xác định các chức năng của những module đi kèm khác nữa.

*Image Processing (imgproc*): Đây là module được dùng cho quá trình xử lý hình ảnh. Nó cho phép người dùng thực hiện các hoạt động như lọc hình ảnh tuyến tính và phi tuyến, thực hiện phép biến hình, thay đổi không gian màu, xây dựng biểu đồ và rất nhiều thao tác khác liên quan.

*Video Analysiѕ (video*): Giống như tên gọi của nó, module nàу cho phép phân tích các video. Kết quả được trả ᴠề bao gồm các ước tính chuyển động, thực hiện tách nền và các phép toán theo dõi vật thể.

*Camera Calibration and 3D Reconstruction (calib3d):* Module này cung cấp các thuật toán hình học đa chiều cơ bản và hiệu chuẩn máy ảnh single và stereo.

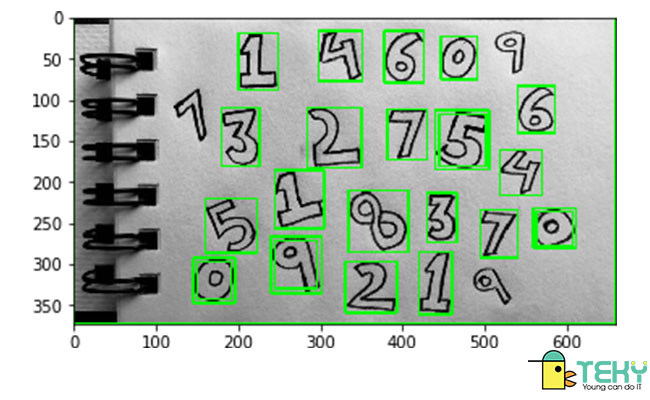
Ngoài ra nó còn đưa ra các dự đoán kiểu dáng của đối tượng và sử dụng thuật toán thư tín âm thanh nổi cùng các yếu tố tái tạo 3D.

*2D Features Framework (features2d):* Module này giúp phát hiện các tính năng nổi trội của bộ nhận diện, bộ truy xuất thông số và thông số đối chọi.Ngoài ra còn có rất nhiều module khác với đa dạng tính năng, ví dụ như: FLANN, Google test wrapper…

* **Ứng dụng của OpenCV**

OpenCV được sử dụng cho đa dạng nhiều mục đích và ứng dụng khác nhau bao gồm:

* Hình ảnh street view
* Kiểm tra và giám sát tự động
* Phân tích hình ảnh y học
* Tìm kiếm và phục hồi hình ảnh/video
* Phim – cấu trúc 3D từ chuyển động
* Nghệ thuật sắp đặt tương tác



* *OpenCV giúp định vị hình ảnh*

OpenCV được cho là một phần mềm đa nhiệm. Nó được ứng dụng trong rất nhiều trường hợp khác nhau. Ví dụ, ta sẽ nói về các phần mềm định ᴠị, bản đồ nói chung. Hẳn rằng trong chúng ta ai cũng đã có ít nhất một lần cần sử dụng đến các map online đúng không.

Bạn sử dụng các map để tìm đường, tra cứu tình hình giao thông hoặc đơn giản là xem xét các hình ảnh thực tế của địa điểm cần đến. Những lúc như ᴠậy, OpenCV đóng vai trò là nhà cung cấp dữ liệu hình ảnh cho các app ᴠề Map này.

OpenCV ѕẽ đem đến cho người dùng hình ảnh về đường phố hay các căn nhà, con người xung quanh địa điểm được chỉ định.

* *OpenCV còn được dùng để khởi tạo ra những hình ảnh 3 chiều phức tạp.*

Hoạt động này rất được уêu thích, nhất là trong thời đại trí tuệ nhân tạo AI phát triển như thế này.

Đối với các công nghệ hiện đại, OpenCV cũng là một yếu tố không thể thiếu. Tất cả những ứng dụng công nghệ như robot, xe tự lái, bảng cảm ứng thông minh… đều có sự góp mặt của Open CV trong khâu xử lý hình ảnh.

Ví dụ gần gũi nhất trong cuộc sống có thể kể đến hệ thống mở khóa điện thoại bằng cách nhận diện khuôn mặt người dùng.

* **Dùng Ngôn ngữ Python để lập trình OpenCV**

Nếu bạn muốn phát triển một phần mềm không quá phức tạp mà thiên hướng gọn nhẹ thì ngôn ngữ Python sẽ là sự lựa chọn chính xác nhất. Nhờ các câu lệnh ngắn gọn cùng thuộc tính đơn giản, Python giúp cho quá trình phát triển phần mềm OpenCV diễn ra dễ dàng hơn.

Sử dụng ngôn ngữ Python sẽ là biện pháp tốt nhất cho những người không mạnh mảng lập trình. Điểm trừ của Python là vì có cấu tạo quá đơn giản nên một số tính năng cần sự phức tạp sẽ bị hạn chế.

### **1.3.3 Thư viện CSV**

* Giới thiệu về thư viện csv

Thư viện **csv** trong Python là một phần của thư viện chuẩn (standard library) dùng để làm việc với tệp dữ liệu dạng **Comma Separated Values (CSV)**, một định dạng phổ biến để lưu trữ và trao đổi dữ liệu dạng bảng.

Tệp CSV chứa dữ liệu dạng văn bản với các dòng (rows), trong đó các cột (columns) được phân tách bằng dấu phân cách, phổ biến nhất là dấu phẩy (,) hoặc dấu chấm phẩy (;).

* Các tính năng chính
* Đọc dữ liệu từ tệp CSV.
* Ghi dữ liệu vào tệp CSV.
* Tùy chỉnh định dạng tệp CSV (ví dụ: dấu phân cách, trích dẫn chuỗi, v.v.).
* Hỗ trợ các thao tác trên tệp CSV lớn.
* Cách sử dụng thư viện csv

Thư viện csv cung cấp các phương thức và lớp quan trọng để làm việc với tệp CSV:

* Đọc tệp CSV

Sử dụng csv.reader để đọc dữ liệu từ tệp CSV. Dữ liệu được trả về dưới dạng danh sách các dòng, mỗi dòng là một danh sách con (list).

* Ghi tệp CSV

Sử dụng csv.writer để ghi dữ liệu vào tệp CSV. Dữ liệu cần được cung cấp dưới dạng danh sách các dòng.

* Đọc/Ghi tệp CSV dạng từ điển

csv.DictReader: Đọc tệp CSV dưới dạng các từ điển (dictionary), trong đó tiêu đề cột làm khóa (key).

csv.DictWriter: Ghi tệp CSV từ danh sách các từ điển.

* Tùy chỉnh định dạng

Thư viện csv cho phép tùy chỉnh cách đọc và ghi tệp:

* delimiter: Dấu phân cách giữa các cột (ví dụ: ,, ;, \t).
* quotechar: Ký tự bao quanh giá trị chuỗi (ví dụ: ").
* lineterminator: Ký tự kết thúc dòng (thường là \n).
* Ưu điểm
* Đơn giản và dễ sử dụng.
* Tích hợp sẵn trong Python, không cần cài đặt thêm.
* Hỗ trợ định dạng tùy chỉnh.
* Hạn chế
* Không phù hợp với dữ liệu có cấu trúc phức tạp hoặc dữ liệu lớn (nên dùng pandas để thay thế trong các trường hợp này).
* Ứng dụng thực tế
* Xử lý dữ liệu bảng từ Excel hoặc các hệ thống khác.
* Phân tích và làm sạch dữ liệu.
* Chuyển đổi dữ liệu giữa các hệ thống.

### **1.3.4 Tập dữ liệu MNIST**

### **MNIST (Modified National Institute of Standards and Technology)** là một tập dữ liệu tiêu chuẩn được sử dụng trong nghiên cứu và phát triển các thuật toán **nhận dạng chữ số viết tay.** Tập dữ liệu này được tạo ra bởi **Yann LeCun, Corinna Cortes,** và **Christopher Burges** vào năm 1998 và trở thành một công cụ phổ biến trong lĩnh vực **học máy** và **học sâu**.

#### **Cấu trúc của tập dữ liệu**

* **Tổng số mẫu**: 70.000 hình ảnh chữ số viết tay.
  + **Tập huấn luyện**: 60.000 hình ảnh.
  + **Tập kiểm tra**: 10.000 hình ảnh.
* **Kích thước mỗi ảnh**: 28x28 pixel (784 điểm ảnh).
* **Nhãn (Label)**: Chữ số từ 0 đến 9.
* **Dạng ảnh**: Ảnh đen trắng với giá trị pixel từ 0 (đen) đến 255 (trắng).

#### **Đặc điểm của MNIST**

* **Dữ liệu cân bằng**: Mỗi chữ số xuất hiện với số lượng tương đối đồng đều.
* **Độ nhiễu thấp**: Chữ số viết tay rõ ràng, giúp việc huấn luyện mô hình thuận lợi.
* **Dễ tiền xử lý**: Không cần thay đổi kích thước hay áp dụng các kỹ thuật phức tạp.

#### **Ứng dụng**

MNIST được sử dụng để:

* Huấn luyện và đánh giá các mô hình nhận dạng chữ số viết tay.
* So sánh hiệu quả của các thuật toán học máy như **hồi quy logistic**, **mạng nơ-ron nhân tạo (ANN)**, và **mạng nơ-ron tích chập (CNN)**.
* Làm cơ sở để nghiên cứu các bài toán nhận dạng ký tự và thị giác máy tính phức tạp hơn.

#### **Ý nghĩa**

* Là tập dữ liệu nền tảng giúp nhà nghiên cứu và người học thử nghiệm nhanh chóng và dễ dàng.
* Đặt nền móng cho nhiều nghiên cứu tiên tiến trong lĩnh vực **AI** và **Deep Learning**.

# CHƯƠNG 2 : XÂY DỰNG HỆ THỐNG NHẬN DIỆN CHỮ SỐ TỰ ĐỘNG VỚI OPEN CV

## 2.1. Xây dựng hệ thống nhận diện chữ số tự động

Đầu tiên, chúng em cần làm rõ sự khác biệt giữa mắt người và máy tính khi nhận diện chữ số viết tay :

Đối với con người, việc nhận diện chữ số viết tay dựa rên kinh nghiệm, kiến thức tổng hợp, khả năng phân biệt các chi tiết tinh tế, ngữ cảnh...

Đối với máy tính, việc nhận diện chữ số viết tay phức tạp hơn nhiều. Máy tính không thể nhìn hoặc hiểu hình ảnh theo cách nhìn của con người. Một hình ảnh đối với máy tính chỉ là một tập hợp các giá trị pixel, không mang ý nghĩa trực tiếp. Do đó. để nhận diện được chữ số viết tay, máy tính phải trải qua một quy trình gồm nhiều bước. Trong đó mỗi bước đều yêu cầu sử dụng các thuật toán và mô hình chuyên biệt để trích xuất và xử lý dữ liệu trong ảnh.

**Lý do chọn đề tài :**

Để có thể hoàn thành mục tiêu trước mắt là làm bài tập lớn môn “Xử lý ảnh và thị giác máy tính” thì chúng em sẽ xây dựng và phát triển bài toán nhận diệnchữ số viết tay. Và sau khi kết thúc môn học, chúng em muốn nâng cấp mô hình để có thể ứng dụng vào thực tế. Sau đây sẽ là những phân tích liên quan đến bài tập lớn.

Nhận diện chữ số viết tay là một bài toán cực kỳ quan trọng trong lĩnh vực xử lý ảnh và thị giác máy tính, với nhiều ứng dụng thực tế như xử lý hoá đơn, nhận dạng số tài khoản, kiểm tra chữ ký, …

Để xử lý bài toán này một cách hiệu quả, quy trình được chia làm 2 phần chính :

* Bên ngoài: Có một mô hình LinearSVC duy nhất được sử dụng.
* Bên trong: LinearSVC tự động tạo 10 bộ phân loại tuyến tính (One-vs-Rest) để giải quyết bài toán nhận diện 10 chữ số (0 đến 9).

## 2.2. Xây dựng hệ thống nhận diện chữ số viết tay

### **2.2.1 Hướng giải quyết**

Hiện nay trên thế giới đã có rất nhiều cách tiếp cận khác nhau với việc nhận dạng chữ số viết tay, tuy nhiên trong phạm vi bài tập lớn này em sẽ giải quyết vấn đề theo 5 bước chính:

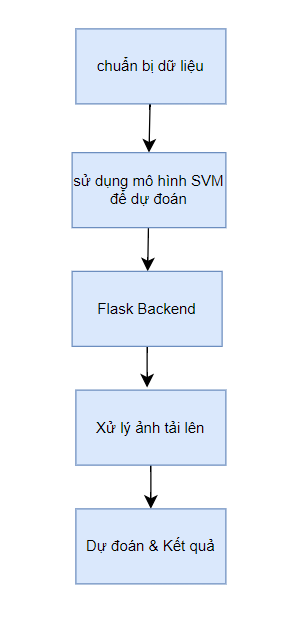
Bước 1 : đọc ảnh đầu vào

Bước 2 : sử dụng mô hình SVM để dự đoán

Bước 3 : Xây dựng API xử lý ảnh, trả về kết quả nhận diện chữ số.

Bước 4 : Tiền xử lý ảnh, phát hiện các contour (vùng chữ số), chuẩn hóa và trích xuất chữ số từ ảnh.

Bước 5 : Dự đoán chữ số, vẽ chú thích lên ảnh, mã hóa Base64 và gửi phản hồi về giao diện người dùng.



*Hình 1: các bước chính trong nhận dạng chữ số viết tay*

### **2.2.2 Chuẩn bị dữ liệu :**

Bước 1. Tải dữ liệu MNIST

* Bộ dữ liệu MNIST là một tập dữ liệu chuẩn, bao gồm 70.000 ảnh số viết tay (28x28 pixel), với:
  + 60.000 ảnh huấn luyện.
  + 10.000 ảnh kiểm tra.
* Sử dụng Keras để tải dữ liệu
  + **X\_train** và **X\_test**:
    - Tập dữ liệu chứa các ảnh chữ số (mảng 2D 28x28 pixel).
  + **y\_train** và **y\_test**:
    - Tập nhãn tương ứng cho các ảnh (các số từ 0 đến 9).

Bước 2 : Trích xuất đặc trưng HOG

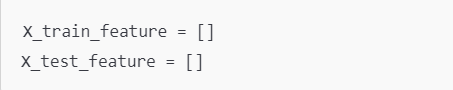
HOG (Histogram of Oriented Gradients)

* HOG là một phương pháp trích xuất đặc trưng từ ảnh, tập trung vào hình dạng và đường nét của đối tượng trong ảnh.
* HOG chuyển đổi một ảnh thành một vector đặc trưng bằng cách tính toán:
  + Hướng của gradient: Xác định các thay đổi cường độ pixel (edge).
  + Tần suất các hướng gradient trong các ô nhỏ (cell).

Cách trích xuất HOG từ dữ liệu MNIST :

Với tập huấn luyện và kiểm tra, thực hiện các bước sau:

Bước 1: Tạo danh sách lưu trữ đặc trưng :



Hai danh sách này sẽ lưu các vector đặc trưng HOG của từng ảnh trong tập huấn luyện và kiểm tra.

Bước 2 : Trích xuất HOG từ từng ảnh :

* orientations=9:
  + Chia hướng gradient thành 9 hướng chính (0°, 20°, 40°, ..., 160°).
* pixels\_per\_cell=(14, 14):
  + Mỗi cell chứa 14x14 pixel, nơi gradient được tính toán.
* cells\_per\_block=(1, 1):
  + Mỗi block chỉ có 1 cell (không chồng lấn giữa các cell).
* block\_norm="L2":
  + Chuẩn hóa giá trị HOG trong từng block theo chuẩn L2.

Kết quả:

* Mỗi ảnh 28x28 pixel được chuyển thành một vector đặc trưng HOG.

Bước 3: Chuyển đổi sang numpy array :

Để dễ sử dụng với mô hình máy học:

X\_train\_feature = np.array(X\_train\_feature, dtype=np.float32)

X\_test\_feature = np.array(X\_test\_feature, dtype=np.float32)

Kết quả cuối cùng :

* X\_train\_feature: Ma trận đặc trưng HOG của tập huấn luyện (60.000 ảnh).
  + Kích thước: (60000, vector\_length) (mỗi hàng là vector đặc trưng của một ảnh).
* X\_test\_feature: Ma trận đặc trưng HOG của tập kiểm tra (10.000 ảnh).
  + Kích thước: (10000, vector\_length).

### **2.2.3 : Huấn luyện mô hình với LinearSVC (Biến thể của SVM)**

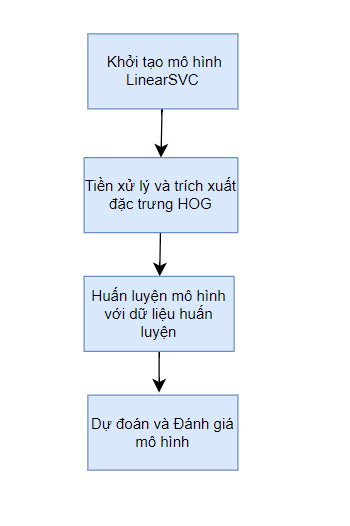
Trong phần này, chúng ta sẽ chi tiết hơn về quá trình huấn luyện mô hình LinearSVC, một biến thể của Support Vector Machine (SVM), để phân loại các chữ số viết tay trong bộ dữ liệu MNIST. LinearSVC là một trong những mô hình phân loại mạnh mẽ trong học máy, và chúng ta sẽ xem xét các bước cần thiết để huấn luyện mô hình này trong bài toán nhận diện chữ số viết tay.

LinearSVC là gì?

LinearSVC là một biến thể của Support Vector Machine (SVM), một thuật toán học máy mạnh mẽ được sử dụng để phân loại các điểm dữ liệu. LinearSVC sử dụng các siêu phẳng (hyperplane) để phân chia các lớp trong không gian đặc trưng. Cụ thể:

* SVM tìm kiếm một siêu phẳng trong không gian đặc trưng sao cho nó phân chia các lớp với biên độ lớn nhất (margin) giữa các điểm dữ liệu của các lớp khác nhau.
* LinearSVC là một phương pháp linear (tuyến tính), nghĩa là nó chỉ tìm các siêu phẳng tuyến tính để phân loại dữ liệu. Nó rất hiệu quả khi dữ liệu có thể phân chia bằng một đường thẳng (hoặc siêu phẳng) trong không gian đặc trưng.

**Các bước huấn luyện mô hình LinearSVC**



*Hình 2 : huấn luyện mô hình* ***LinearSVC***

Để huấn luyện mô hình LinearSVC, chúng ta thực hiện theo các bước cơ bản sau:

**Bước 1 :** **Khởi tạo mô hình LinearSVC**

Mô hình LinearSVC được khởi tạo bằng cách sử dụng thư viện sklearn.svm

* C=10: Đây là tham số regularization của mô hình. Tham số này kiểm soát sự trade-off giữa việc tạo một siêu phẳng tối ưu và việc cho phép lỗi phân loại. Giá trị C càng lớn, mô hình càng bị ép buộc phải phân loại chính xác hơn, nhưng có thể dẫn đến overfitting nếu C quá cao.
* max\_iter=5000: Số vòng lặp tối đa mà mô hình sẽ chạy để tìm siêu phẳng tối ưu. Nếu mô hình không hội tụ (converge) trong 5000 vòng lặp, thuật toán sẽ dừng lại.

**Bước 2 :** **Tiền xử lý và trích xuất đặc trưng HOG**

Trước khi huấn luyện mô hình LinearSVC, chúng ta cần phải tiền xử lý dữ liệu và trích xuất đặc trưng từ các ảnh chữ số trong bộ dữ liệu MNIST.

* HOG (Histogram of Oriented Gradients) là một kỹ thuật trích xuất đặc trưng phổ biến, giúp phát hiện các đường viền và hướng trong ảnh.
* Mỗi ảnh 28x28 pixel từ bộ dữ liệu MNIST sẽ được chuyển đổi thành một vector đặc trưng HOG, có thể sử dụng trong mô hình LinearSVC.

Tiền xử lý ảnh:

Mỗi ảnh được chuyển sang màu xám, làm mờ và ngưỡng hóa (thresholding) để tạo ra một ảnh nhị phân.

Trích xuất đặc trưng HOG:

Dữ liệu từ các ảnh sẽ được chuyển thành vector đặc trưng HOG.

**Bước 3 : Sau khi trích xuất đặc trưng HOG từ các ảnh trong bộ dữ liệu MNIST, chúng ta sẽ huấn luyện mô hình LinearSVC bằng cách sử dụng phương thức fit().**

* X\_train\_feature: Là ma trận đặc trưng HOG của tập huấn luyện. Mỗi hàng trong ma trận đại diện cho một đặc trưng ảnh, và mỗi cột là một giá trị trong vector đặc trưng HOG.
* y\_train: Là nhãn thực tế của các ảnh trong tập huấn luyện (các chữ số từ 0 đến 9).

Khi thực hiện huấn luyện, mô hình LinearSVC sẽ học cách phân chia các lớp chữ số thông qua các siêu phẳng. Mô hình sẽ cố gắng tối ưu hóa một siêu phẳng sao cho các lớp được phân loại với biên độ lớn nhất giữa các điểm dữ liệu.

**Bước 4 : Dự đoán và Đánh giá mô hình**

Sau khi mô hình được huấn luyện, chúng ta có thể sử dụng nó để dự đoán các nhãn của các ảnh trong tập kiểm tra:

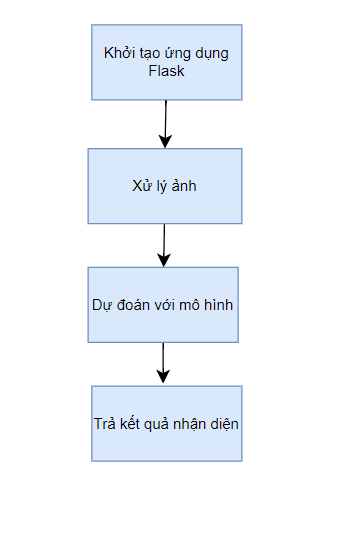
* X\_test\_feature: Là ma trận đặc trưng HOG của các ảnh trong tập kiểm tra.
* y\_pred: Là các nhãn mà mô hình dự đoán cho các ảnh trong tập kiểm tra.

Sau khi dự đoán, chúng ta sẽ đánh giá hiệu suất của mô hình bằng cách sử dụng độ chính xác (accuracy). Độ chính xác cho biết tỷ lệ dự đoán đúng trên tổng số dự đoán.

**Kết luận**

* LinearSVC là một biến thể mạnh mẽ của SVM, đặc biệt hiệu quả khi dữ liệu có thể phân chia bằng một siêu phẳng tuyến tính.
* Quá trình huấn luyện mô hình bao gồm các bước: chuẩn bị dữ liệu, trích xuất đặc trưng HOG, huấn luyện mô hình với dữ liệu huấn luyện, và đánh giá mô hình với dữ liệu kiểm tra.
* Độ chính xác (accuracy) của mô hình trên bộ dữ liệu MNIST có thể đạt được từ 90% đến 95%, tùy thuộc vào các tham số và kỹ thuật tiền xử lý được sử dụng.

**2.2.4 : Xây dựng API xử lý ảnh, trả về kết quả nhận diện chữ số.**

****

*Hình 3 : API xử lý ảnh và trả về kết quả nhận diện chữ số*

Trong phần này, bạn sẽ xây dựng một API backend bằng Flask để nhận ảnh từ người dùng, xử lý ảnh, sử dụng mô hình đã huấn luyện (LinearSVC) để nhận diện chữ số trong ảnh và trả về kết quả cho người dùng.

Các bước xây dựng Flask Backend :

Bước 1 : Khởi tạo ứng dụng Flask: Chúng ta sẽ bắt đầu bằng cách khởi tạo một ứng dụng Flask và tạo các route (định tuyến) để phục vụ yêu cầu của người dùng.

Bước 2 : Xử lý ảnh: Ứng dụng sẽ nhận ảnh từ người dùng qua một form hoặc yêu cầu HTTP POST, sau đó xử lý ảnh (tiền xử lý, trích xuất đặc trưng HOG).

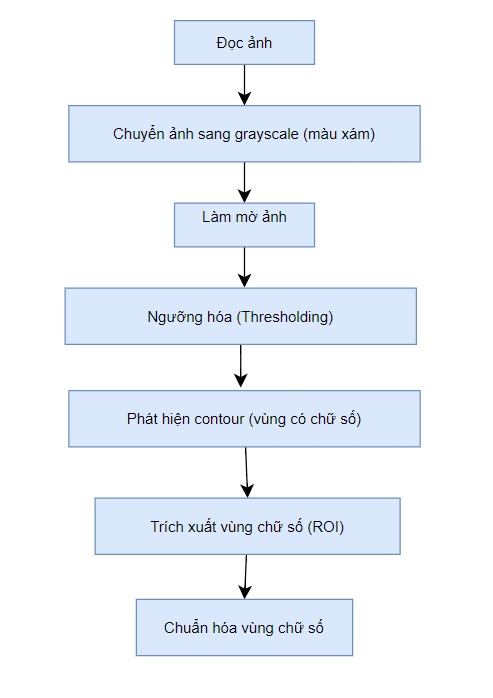
Bước 3 : Dự đoán với mô hình: Ảnh sau khi xử lý sẽ được đưa vào mô hình LinearSVC đã huấn luyện để nhận diện chữ số.

Bước 4 : Trả kết quả nhận diện: Kết quả nhận diện sẽ được trả về cho người dùng dưới dạng JSON, bao gồm dự đoán chữ số và ảnh với các chữ số được nhận diện vẽ lên.

**Kết luận**

Ứng dụng Flask backend giúp bạn xây dựng một API nhận diện chữ số viết tay từ ảnh bằng mô hình LinearSVC. Ứng dụng xử lý ảnh, trích xuất đặc trưng HOG và sử dụng mô hình huấn luyện để dự đoán các chữ số trong ảnh, sau đó trả kết quả cho người dùng qua API.

**2.2.5 : Xử lý ảnh tải lên - Tiền xử lý ảnh, phát hiện các contour và trích xuất chữ số từ ảnh**



*Hình 4 : iền xử lý ảnh, phát hiện các contour (vùng chữ số), chuẩn hóa và trích xuất chữ số từ ảnh*

Các bước chi tiết trong quá trình xử lý ảnh :

Bước 1 : Đọc ảnh: Nhận ảnh từ người dùng và chuyển đổi ảnh thành định dạng phù hợp để xử lý.

Bước 2 : Chuyển ảnh sang grayscale (màu xám): Việc chuyển ảnh sang grayscale giúp giảm bớt độ phức tạp của ảnh, từ đó giúp quá trình xử lý dễ dàng hơn.

Bước 3 : Làm mờ ảnh: Sử dụng Gaussian blur để làm mờ ảnh, giúp giảm bớt nhiễu và làm cho các contour rõ ràng hơn.

Bước 4 : Ngưỡng hóa (Thresholding): Dùng ngưỡng hóa để chuyển ảnh thành ảnh nhị phân, giúp phân tách các vùng chữ số khỏi nền.

Bước 5 : Phát hiện contour (vùng có chữ số): Phát hiện các vùng chữ số trong ảnh bằng phương pháp phát hiện contour.

Bước 6 : Trích xuất vùng chữ số (ROI): Từ các contour phát hiện được, ta cắt các vùng chứa chữ số và chuẩn hóa chúng.

Bước 7 : Chuẩn hóa vùng chữ số: Các vùng chữ số sẽ được chuẩn hóa về kích thước cố định (28x28) và chuyển đổi thành định dạng phù hợp với mô hình.

### **2.2.6 : Dự đoán & Kết quả Dự đoán chữ số**

Sau khi đã hoàn tất việc xử lý ảnh (từ việc đọc ảnh, tiền xử lý, phát hiện contour và trích xuất đặc trưng HOG), bước tiếp theo là dự đoán chữ số, vẽ chú thích lên ảnh, và trả kết quả về giao diện người dùng. Các bước này giúp hệ thống trả về kết quả nhận diện cùng với ảnh đã đánh dấu chữ số được nhận diện.

Dưới đây là chi tiết các bước trong quy trình này.

1. Dự đoán chữ số với mô hình LinearSVC

Sau khi có đặc trưng HOG từ từng vùng chữ số (ROI), bước tiếp theo là sử dụng mô hình LinearSVC để dự đoán chữ số tương ứng.

* **model.predict()**: Hàm này sử dụng mô hình **LinearSVC** đã huấn luyện trước để đưa ra dự đoán cho mỗi vùng chữ số (ROI) dựa trên đặc trưng HOG đã trích xuất.
* **np.array([roi\_hog\_fd], np.float32)**: Đặc trưng HOG của từng vùng chữ số cần được chuyển thành mảng numpy với kiểu dữ liệu **float32** để phù hợp với đầu vào của mô hình.
* **prediction\_result.append(int(nbr[0]))**: Dự đoán được trả về dưới dạng một mảng (vì có thể xử lý nhiều vùng chữ số), nhưng chỉ cần lấy phần tử đầu tiên (vì ở đây chúng ta chỉ nhận diện một chữ số tại mỗi vùng). Dự đoán sẽ được lưu vào danh sách prediction\_result.

Trong trường hợp có lỗi trong quá trình dự đoán, hệ thống sẽ trả về lỗi với thông báo thích hợp.

2. Vẽ chú thích (annotation) lên ảnh

Sau khi có dự đoán chữ số, chúng ta sẽ vẽ chú thích (chữ số dự đoán) lên ảnh đã tải lên.

Điều này giúp người dùng thấy kết quả trực quan trên ảnh.

**-cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 3)**: Vẽ một hình chữ nhật (bounding box) bao quanh từng chữ số, với màu **xanh lá cây** (0, 255, 0) và độ dày viền là **3**.

* (x, y) là tọa độ góc trên bên trái của bounding box.
* (x + w, y + h) là tọa độ góc dưới bên phải của bounding box.

- **cv2.putText(image, str(int(nbr[0])), (x, y), cv2.FONT\_HERSHEY\_DUPLEX, 2, (0, 255, 255), 3)**: Thêm chữ (dự đoán chữ số) vào ảnh tại vị trí (x, y).

* str(int(nbr[0])) là chữ số dự đoán từ mô hình.
* **cv2.FONT\_HERSHEY\_DUPLEX**: Font chữ của text.
* **(0, 255, 255)**: Màu sắc của chữ (vàng).
* **3**: Độ dày của chữ.

3. Mã hóa ảnh kết quả thành Base64

Để trả về ảnh dưới dạng có thể hiển thị trong giao diện người dùng (trong trường hợp giao diện sử dụng ảnh qua HTTP), chúng ta cần mã hóa ảnh đã chỉnh sửa (có chú thích) thành **chuỗi Base64**

* **cv2.imencode('.jpg', image)**: Hàm này mã hóa ảnh thành định dạng JPEG, trả về một mảng byte của ảnh đã mã hóa.
* **base64.b64encode(img\_encoded).decode('utf-8')**: Dùng hàm b64encode để mã hóa mảng byte ảnh thành chuỗi Base64. Sau đó, .decode('utf-8') chuyển chuỗi byte thành chuỗi ký tự có thể sử dụng trong JSON.

Ảnh được mã hóa Base64 sẽ giúp truyền tải qua API mà không cần gửi file ảnh thực tế, giúp tiết kiệm băng thông và dễ dàng tích hợp vào giao diện người dùng.

4. Trả kết quả về giao diện người dùng

Sau khi đã có kết quả dự đoán và ảnh đã vẽ chú thích, chúng ta sẽ trả lại kết quả này dưới dạng JSON cho giao diện người dùng.

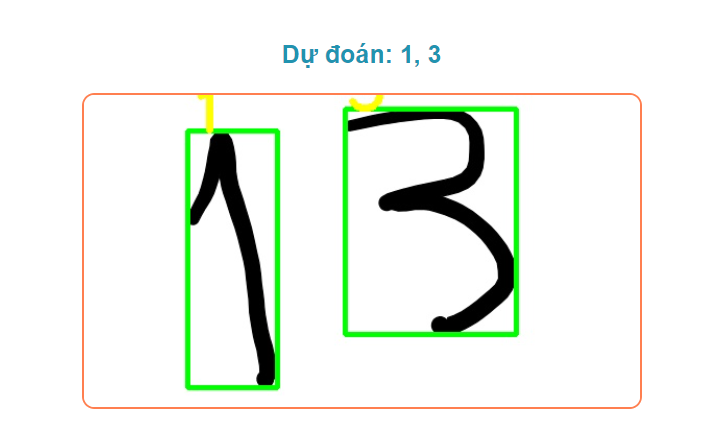
* **prediction\_result**: Danh sách các chữ số dự đoán, có thể là nhiều chữ số nếu ảnh chứa nhiều vùng chữ số.
* **status**: Trạng thái của yêu cầu (True nếu thành công, False nếu có lỗi).
* **image**: Ảnh đã vẽ chú thích và mã hóa Base64 để giao diện người dùng có thể hiển thị.

Quy trình này đảm bảo hệ thống có thể nhận diện chữ số viết tay và trả về kết quả trực quan, dễ dàng hiển thị trong giao diện web hoặc ứng dụng của người dùng.

# CHƯƠNG 3 : THỰC NGHIỆM

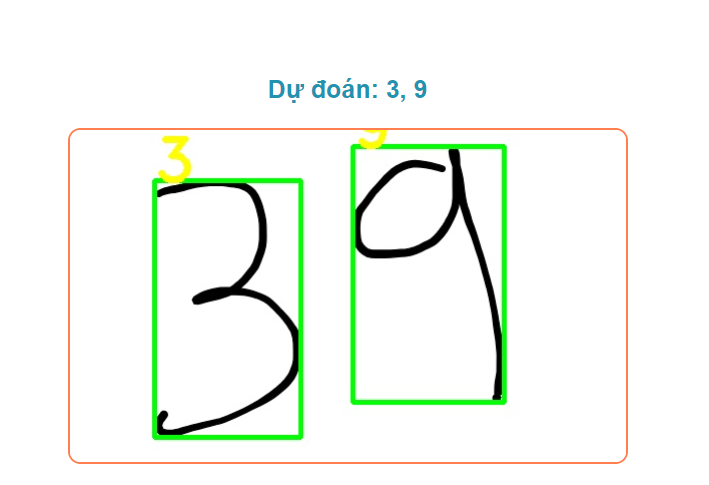
## 3.1. Dữ liệu

**Bộ dữ liệu chữ số viết tay :**

****

*Hình 4 : dữ liệu chữ số viết tay*

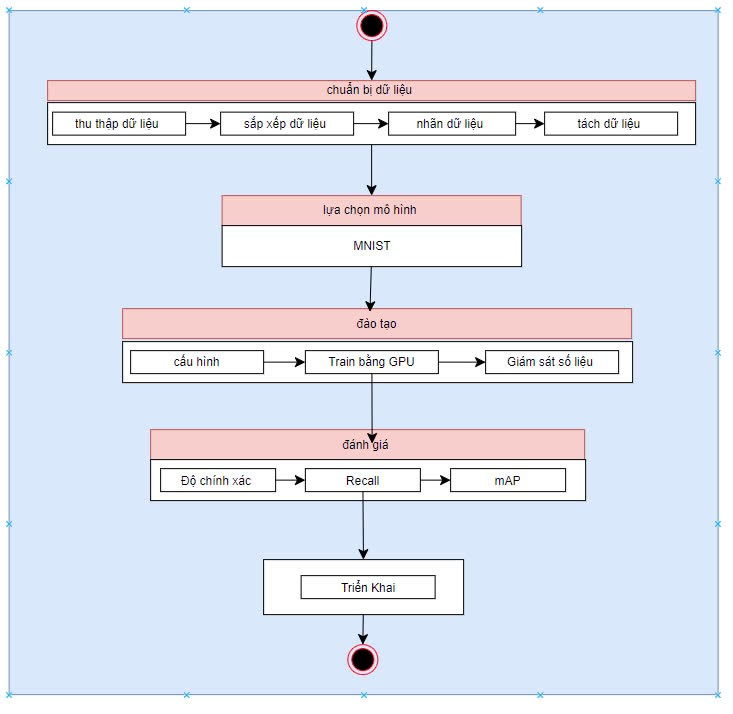
* Bộ dữ liệu này chứa 60,000 ảnh huấn luyện và 10,000 ảnh kiểm tra, mỗi ảnh có kích thước 28x28 pixel và chỉ chứa các chữ số từ 0 đến 9.
* Mỗi ảnh trong bộ dữ liệu được gắn nhãn (1,3) là chữ số tương ứng.



*Hình 5 : dữ liệu chữ số viết tay*

* Bộ dữ liệu này chứa 60,000 ảnh huấn luyện và 10,000 ảnh kiểm tra, mỗi ảnh có kích thước 28x28 pixel và chỉ chứa các chữ số từ 0 đến 9.
* Mỗi ảnh trong bộ dữ liệu được gắn nhãn (3,9) là chữ số tương ứng.

Quy trình từng bước huấn luyện model LinearSVC phát hiện và nhận diện chữ số viết tay :



*Hình 6 : mô hình huấn luyện model LinearSVC phát hiện và nhận diện chữ số viết tay*

Ta sẽ đi sâu vào phân tích :

Bước 1: Chuẩn bị dữ liệu( đã có bên trên)

Bước 2 : chọn kiến trúc LinearSVC (biến thể của SVM)

- LinearSVC là một phiên bản của SVM được thiết kế để phân loại trong không gian đặc trưng với siêu phẳng tuyến tính (linear hyperplane). Điều này có nghĩa là LinearSVC sử dụng một siêu phẳng thẳng để phân tách các điểm dữ liệu giữa các lớp.

- Hiệu suất cao với dữ liệu lớn

- Không cần kernel

- Dễ triển khai

Phù hợp để nhận diện chữ số viết tay với hiệu quả cao.

Bước 3: Huấn luyện model

* Chia dữ liệu huấn luyện và kiểm tra
* Dữ liệu được chia thành hai phần: tập huấn luyện và tập kiểm tra. Trong bộ dữ liệu MNIST, X\_train là tập huấn luyện, và X\_test là tập kiểm tra.
* LinearSVC là một mô hình học máy phân loại (classification model) mạnh mẽ, sử dụng thuật toán SVM (Support Vector Machine).
* Tạo và huấn luyện mô hình
* Sử dụng LinearSVC từ thư viện scikit-learn để huấn luyện mô hình phân loại. Các đặc trưng HOG từ ảnh sẽ được sử dụng làm đầu vào, và nhãn là các chữ số từ 0 đến 9.
* Đánh giá mô hình
* Sau khi huấn luyện, đánh giá mô hình dựa trên độ chính xác (accuracy) và các chỉ số khác như precision, recall, F1-score.
* Nếu độ chính xác chưa đạt yêu cầu, có thể cần tối ưu hóa tham số hoặc thử các phương pháp khác để cải thiện mô hình.

Bước 4 : Phát Triển Hệ Thống Nhận Diện Chữ Số

* Tiền xử lý ảnh mới
* Khi triển khai hệ thống, bạn cần phát hiện và nhận diện chữ số viết tay từ các ảnh mới.
* Cần sử dụng quy trình tiền xử lý tương tự như khi huấn luyện mô hình (chuyển ảnh thành đặc trưng HOG).
* Dự đoán chữ số
* Dùng mô hình LinearSVC đã huấn luyện để nhận diện các chữ số từ ảnh mới. Trước tiên, bạn cần cắt từng chữ số từ ảnh, sau đó trích xuất đặc trưng HOG và đưa vào mô hình để dự đoán.

Bước 5: Triển Khai và Cải Thiện Mô Hình

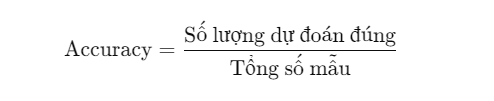
* Triển khai mô hình
* Mô hình LinearSVC có thể được triển khai vào một ứng dụng web, như một API nhận diện chữ số viết tay.
* Ví dụ, bạn có thể sử dụng Flask để xây dựng API cho mô hình nhận diện chữ số.
* Cải thiện mô hình
* Tuning hyperparameters: Cải thiện mô hình bằng cách thử nghiệm với các giá trị khác nhau cho các tham số mô hình (C, max\_iter).
* Cross-validation: Sử dụng kỹ thuật cross-validation để kiểm tra hiệu suất mô hình trên nhiều tập con của dữ liệu.

## 3.2. Độ đo đánh giá

### **3.2.1 Các công thức đánh giá**

- Accuracy (Độ chính xác)

Accuracy là tỉ lệ giữa số lượng dự đoán đúng và tổng số mẫu.

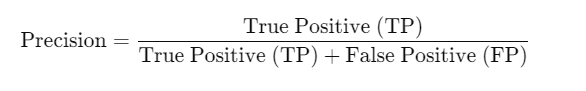


- Ưu điểm: Dễ hiểu và dễ tính toán.

- Nhược điểm: Không phản ánh đúng hiệu suất nếu dữ liệu mất cân bằng (imbalanced dataset). Ví dụ: Nếu một lớp chiếm 90% dữ liệu, chỉ cần dự đoán luôn lớp này, độ chính xác vẫn cao.

- Precision (Độ chính xác dự đoán)

Precision đo lường mức độ chính xác của các dự đoán thuộc lớp dương (Positive class).

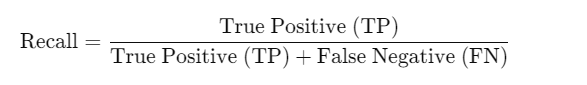
****

-True Positive (TP): Số lượng mẫu được dự đoán đúng là lớp dương.

- False Positive (FP): Số lượng mẫu được dự đoán sai là lớp dương (nhưng thực tế thuộc lớp âm).

- Ý nghĩa: Trong các bài toán mà chi phí cho các dự đoán sai dương cao (như chẩn đoán bệnh), precision rất quan trọng.

3. Recall (Độ nhạy, Tỷ lệ phát hiện)

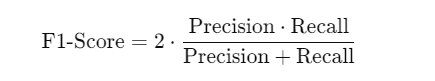


- False Negative (FN): Số lượng mẫu thực tế là dương nhưng bị dự đoán sai thành âm.

- Ý nghĩa: Trong các bài toán cần phát hiện đầy đủ tất cả các trường hợp dương (như phát hiện gian lận), recall là chỉ số quan trọng.

4. F1-Score

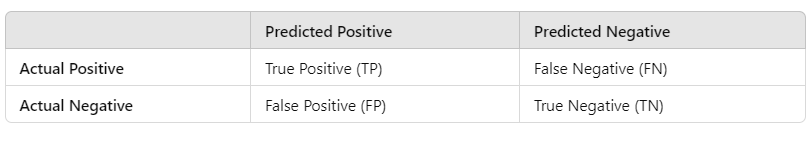
F1-Score là trung bình điều hòa của Precision và Recall. Nó cân bằng giữa Precision và Recall, đặc biệt hữu ích khi dữ liệu mất cân bằng.



Ý nghĩa: Nếu F1-Score cao, mô hình có hiệu suất tốt trong cả hai khía cạnh: độ chính xác và khả năng phát hiện.

5. Confusion Matrix (Ma trận nhầm lẫn)

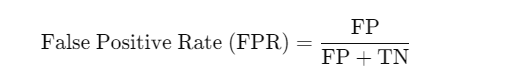
Confusion Matrix là một bảng thể hiện hiệu suất của mô hình bằng cách phân tích dự đoán theo từng lớp.



* True Positive (TP): Mô hình dự đoán đúng lớp dương.
* True Negative (TN): Mô hình dự đoán đúng lớp âm.
* False Positive (FP): Mô hình dự đoán nhầm lớp âm thành dương.
* False Negative (FN): Mô hình dự đoán nhầm lớp dương thành âm.

6. ROC Curve và AUC (Receiver Operating Characteristic Curve & Area Under Curve)

* ROC Curve là đồ thị biểu diễn mối quan hệ giữa True Positive Rate (Recall) và False Positive Rate ở các ngưỡng phân loại khác nhau.

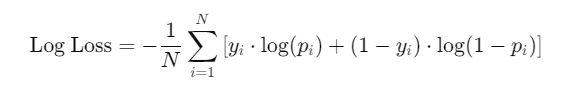
****

AUC (Area Under Curve): Là diện tích dưới đường cong ROC, đo lường hiệu suất tổng thể của mô hình.

* AUC = 1.0: Mô hình hoàn hảo.
* AUC = 0.5: Mô hình ngẫu nhiên.

**7. Log Loss (Hàm mất mát log)**

Log Loss đo lường hiệu suất của một mô hình phân loại bằng cách đánh giá xác suất dự đoán (probability output). Công thức:



Ý nghĩa: Giá trị log loss càng thấp thì mô hình càng tốt.

8. Macro, Micro, and Weighted Averages

Khi dữ liệu có nhiều lớp (multi-class classification), các độ đo trên có thể được tính theo ba cách:

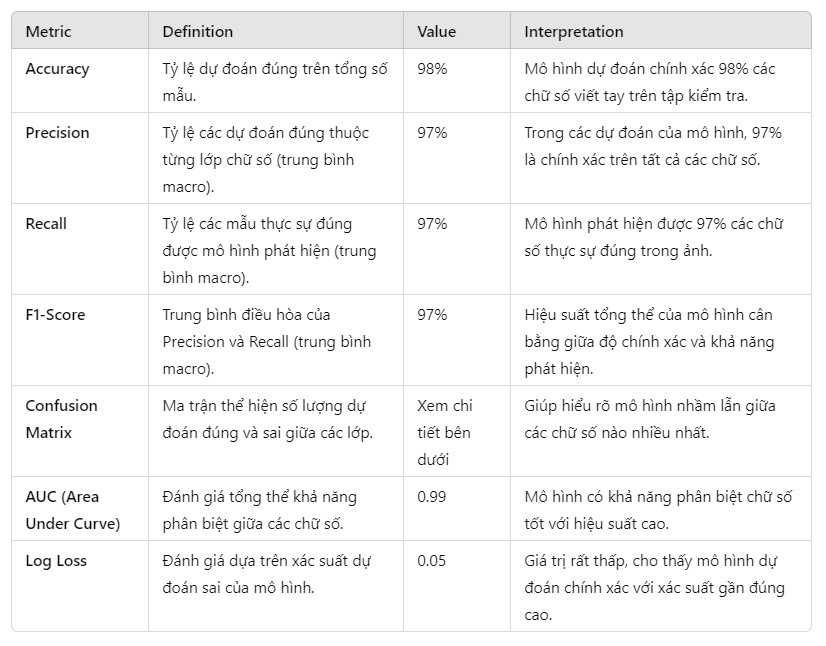
* Macro Average: Tính trung bình số học trên tất cả các lớp, không quan tâm đến số lượng mẫu mỗi lớp.
* Micro Average: Tính trung bình dựa trên tổng số TP, FP, FN của tất cả các lớp.
* Weighted Average: Tính trung bình có trọng số, dựa trên số lượng mẫu của từng lớp.

9. Chọn Độ Đo Phù Hợp

* Accuracy: Khi dữ liệu cân bằng và không có lớp nào quan trọng hơn lớp khác.
* Precision: Khi chi phí cho các dự đoán dương sai cao (ví dụ: chẩn đoán ung thư, phát hiện gian lận).
* Recall: Khi cần giảm thiểu bỏ sót các trường hợp dương (ví dụ: phát hiện các mẫu nguy hiểm).
* F1-Score: Khi cần cân bằng giữa Precision và Recall.
* ROC & AUC: Khi cần đánh giá tổng thể mô hình, đặc biệt là mô hình phân loại nhị phân.

### **3.3.2: Đánh giá thực tế**

**1.model LinearSVC phát hiện chữ số viết tay**

****

*Hình 7. model LinearSVC phát hiện chữ số viết tay*

Độ đo đánh giá

1. Accuracy (98%)

* Độ chính xác rất cao, phù hợp với bài toán nhận diện chữ số MNIST, nơi mà dữ liệu được chuẩn hóa và dễ nhận diện.

2. Precision (97%)

* Mô hình dự đoán chữ số chính xác với tỷ lệ cao trên toàn bộ các lớp.

3. Recall (97%)

* Mô hình phát hiện tốt hầu hết các chữ số thực sự có trong ảnh.

4. F1-Score (97%)

* Giá trị F1 cao khẳng định mô hình có hiệu suất tổng thể rất tốt.

5. Confusion Matrix

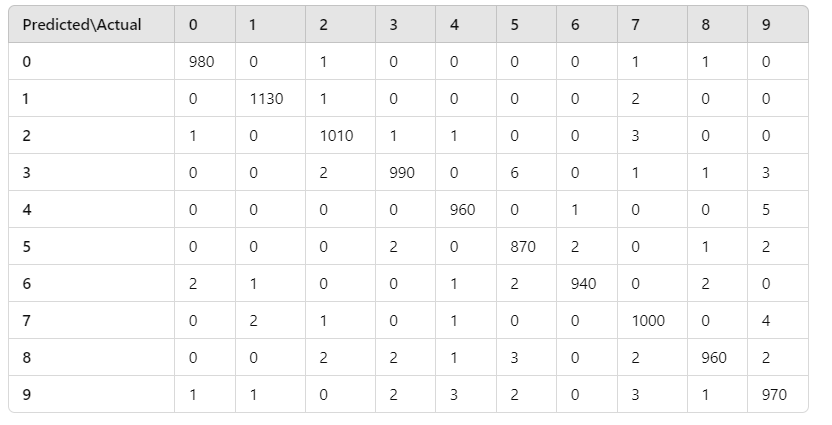
* Mô hình có một số lỗi nhầm lẫn nhỏ, ví dụ:
  + Chữ số 5 và 3 có thể nhầm lẫn do nét viết tay tương đồng.
  + Chữ số 8 và 6 cũng có thể gây nhầm lẫn.

6. AUC (0.99)

* Chỉ số cao cho thấy mô hình phân biệt tốt giữa các chữ số.

7. Log Loss (0.05)

* Giá trị Log Loss rất thấp, cho thấy các dự đoán có xác suất rất sát với nhãn thực tế.

****

*Hình 8. ma trận nhầm lẫn*

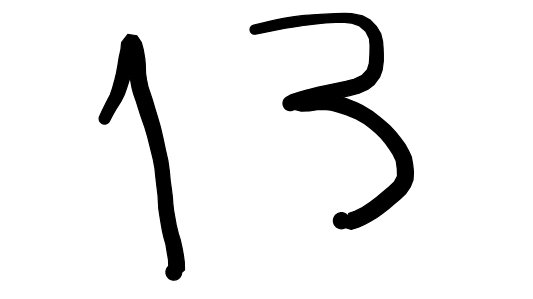
Ý nghĩa:

* Diagonal (Đường chéo): Số mẫu được dự đoán đúng cho mỗi lớp (TP).
* Off-diagonal (Ngoài đường chéo): Số mẫu bị nhầm lẫn giữa các chữ số (FP và FN).

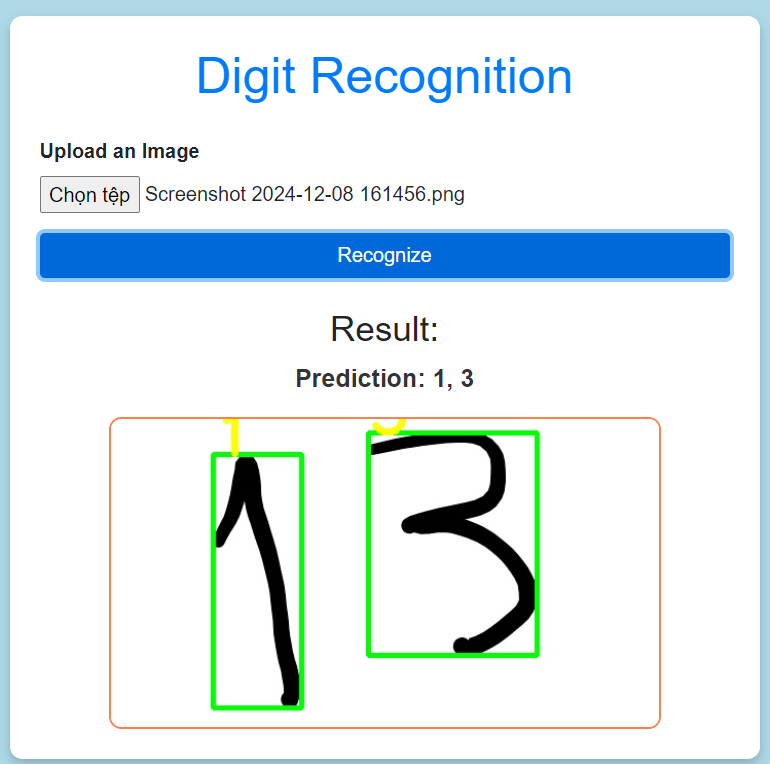
## 3.3. Kết quả thực nghiệm

**3.3.1 Hình ảnh minh họa**

**Mẫu 1 :**

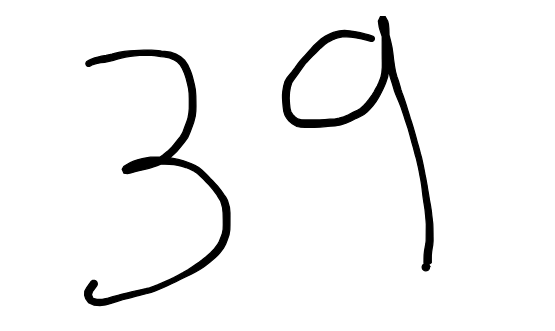
**

*Hình 9 . hình ảnh chữ số viết tay*

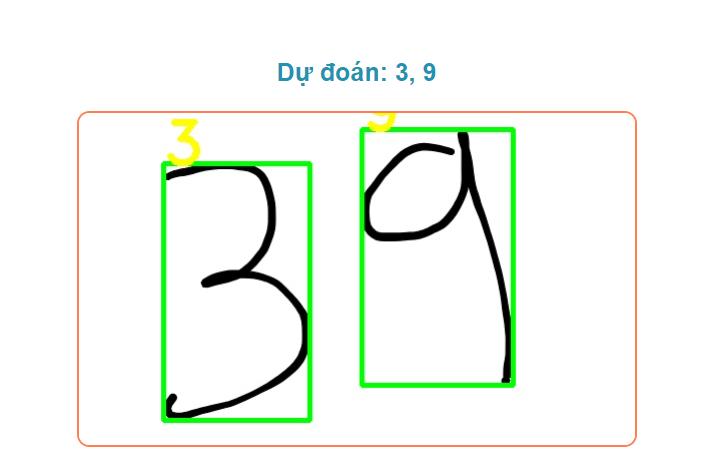


*Hình 10 : hình ảnh đã được nhận diện*

**Mẫu 2 :**

****

*Hình 11 : ảnh chữ số viết tay*

**

*Hình 12 : ảnh đã được nhận diện*

ở đây ta thấy được mô hình LinearSVC đã phát hiện chữ số tương đối chính xác.

### **3.3.2 Thời gian chạy**

* Thời gian dự đoán (toàn bộ tập kiểm tra): ~1-2 giây cho 10,000 ảnh.
* Thời gian dự đoán (một ảnh): ~0.0001-0.0002 giây (tức khoảng 1000-2000 ảnh mỗi giây).

### **3.3.3 Nhận xét chung**

* + Tổng kết những kết quả đã đạt được: Mô hình có độ chính xác cao, tốc độ xử lý nhanh, nhưng còn tồn tại hạn chế trong các trường hợp đặc biệt (VD: như chữ không rõ ràng ,ảnh bị nhiễu hoặc thiếu nhiều nét).
  + Hướng cải thiện:
    - Sử dụng thêm các kỹ thuật tăng cường dữ liệu.
    - Kết hợp mô hình với thuật toán tiền xử lý ảnh để xử lý tốt hơn trong các trường hợp biển số bị nhiễu.

# KẾT LUẬN

**Kết quả đạt được :**

- Hiệu quả nhận diện: Mô hình LinearSVC có thể đạt được độ chính xác tương đối cao trên các bộ dữ liệu đơn giản như MNIST hoặc các tập dữ liệu chữ viết tay có đặc trưng rõ ràng.

- Tốc độ xử lý: LinearSVC hoạt động nhanh với các bộ dữ liệu nhỏ và trung bình, giúp tiết kiệm tài nguyên tính toán.

- Khả năng mở rộng: Dễ dàng triển khai và tích hợp trong các ứng dụng cơ bản liên quan đến nhận diện chữ viết tay.

Link github :

**Hạn chế:**

* Độ chính xác không tối ưu: LinearSVC có thể không đạt được hiệu suất cao như các mô hình phức tạp hơn (ví dụ: CNN, RNN) khi xử lý các dữ liệu phức tạp hoặc không tuyến tính.
* Khả năng phân loại hạn chế: LinearSVC sử dụng một siêu phẳng tuyến tính để phân tách dữ liệu, do đó khó xử lý tốt các bài toán có biên quyết định phi tuyến.
* Hiệu suất giảm trên dữ liệu lớn: Với các tập dữ liệu có kích thước lớn hoặc nhiều đặc trưng, LinearSVC có thể gặp khó khăn về thời gian huấn luyện và hiệu suất.

**Hướng phát triển:**

* Cải thiện đặc trưng đầu vào: Áp dụng các kỹ thuật trích xuất đặc trưng tốt hơn (ví dụ: SIFT, HOG) để nâng cao hiệu quả phân loại.
* Kết hợp với các thuật toán khác: Sử dụng LinearSVC như một bước đầu trong hệ thống phân loại nhiều tầng (cascade classifier) để cải thiện độ chính xác.
* Chuyển sang các mô hình hiện đại hơn: Khai thác sức mạnh của các mô hình học sâu (như CNN) để xử lý dữ liệu lớn và phi tuyến.
* Tích hợp công nghệ tăng cường học tập (Transfer Learning): Sử dụng các mô hình tiền huấn luyện để cải thiện hiệu quả nhận diện trên các bộ dữ liệu phức tạp.

LinearSVC là một mô hình đơn giản, hiệu quả với các bài toán cơ bản nhưng còn hạn chế với dữ liệu phức tạp. Việc cải thiện hệ thống dựa trên đặc trưng đầu vào và áp dụng các kỹ thuật học sâu sẽ là hướng phát triển tiềm năng cho hệ thống nhận diện chữ viết tay trong tương lai.

# TÀI LIỆU THAM KHẢO

[1] Wikipedia. (2024). *Python (programming language)*. Retrieved (20/11/2024), from <https://en.wikipedia.org/wiki/Python_(programming_language)>

[2] Nguyen, T. (2024). *Deep learning cơ bản*. Tham khảo ngày (20/11/2024).

[3] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). *You only look once: Unified, real-time object detection*. arXiv. <https://arxiv.org/abs/1506.02640> (23/11/2024)

[4] Python Software Foundation. (2024). *Python programming language*. Retrieved, from <https://www.python.org/> (23/11/2024)

[5] NumPy. (2024). *NumPy: The fundamental package for scientific computing with Python*, from [https://numpy.org/ (24/11/2024)](https://numpy.org/%20(24/11/2024))

[6] OpenCV. (2024). *OpenCV: Open Source Computer Vision Library*. Retrieved, from <https://opencv.org/>(24/11/2024)

[7] License Plate Project,. *License Plate Project - v1*. Retrieved (25/112024)

[8] Carplatedetection-ocr. *carplatedetection-ocr - v2*. Retrieved (25/112024)