



**ĐẠI HỌC
BÁCH KHOA HÀ NỘI**
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

ĐỀ TÀI: NHẬN DIỆN CHỮ SỐ VIẾT TAY BẰNG MACHINE LEARNING VỚI THUẬT TOÁN K-NN

Nhóm bảo vệ: Nhóm 6

Học phần: Nhập môn trí tuệ nhân tạo (AC2060)

Giảng viên hướng dẫn: PGS.TS. Vũ Hải

ONE LOVE. ONE FUTURE.

Đề tài:

Nhận diện chữ số viết tay Machine Learning bằng thuật toán K-NN

Phần 1

Tóm tắt đề tài

Phần 2

Giới thiệu đề tài

Phần 3

Tổng quan quá trình nhận diện chữ viết tay bằng học máy

Phần 4

Cơ sở lý thuyết mô hình kNN trong nhận diện chữ viết tay

Phần 5

Chương trình nhận diện chữ số viết tay

Phần 6

Kết quả đề tài

Danh sách thành viên

Mã lớp: 158785

Nhóm thuyết trình: 6

STT	Họ và tên sinh viên	Mã số sinh viên	Nhiệm vụ
1	Đinh Mạnh Tiến (Nhóm trưởng)	20231638	Code chương trình nhận diện số viết tay (gán nhãn vector & huấn luyện kiểm thử mô hình & tính khoảng cách và phân loại)
2	Phạm Thu Uyên	20231648	Code chương trình nhận diện số viết tay (tiền xử lý ảnh & chuyển đổi ảnh vector)
3	Trần Phương Thảo	20231634	Tìm hiểu bài báo, xây dựng lý thuyết về phân tích mô hình kNN của đề tài
4	Nguyễn Huyền Trang	20231640	Tìm hiểu bài báo, xây dựng lý thuyết về tổng quan chữ số viết tay bằng học máy



HUST

PHẦN 1

TÓM TẮT ĐỀ TÀI



I. Tóm tắt đề tài

Đề tài tập trung xây dựng hệ thống nhận diện chữ số viết tay bằng thuật toán k-Nearest Neighbors (kNN) trên bộ dữ liệu MNIST. Quy trình bao gồm tiền xử lý ảnh và chuyển đổi chúng thành vector đặc trưng, sau đó áp dụng kNN dựa trên các độ đo khoảng cách (Euclidean, Manhattan) để phân loại. Kết quả cho thấy kNN đạt độ chính xác trên tập kiểm thử, với hiệu suất bị ảnh hưởng bởi giá trị k, độ đo khoảng cách, và đặc biệt là hạn chế về tốc độ xử lý trên dữ liệu lớn. Hướng phát triển trong tương lai bao gồm so sánh với các thuật toán phức tạp hơn và cải thiện bước tiền xử lý dữ liệu.



HUST

PHẦN 2

GIỚI THIỆU ĐỀ TÀI



II. Giới thiệu đề tài

1. Lý do chọn đề tài:

- Trong bối cảnh Trí tuệ nhân tạo đang ngày càng phát triển và ứng dụng rộng rãi vào mọi mặt của đời sống, việc nắm vững các kiến thức nền tảng về học máy là vô cùng cần thiết. Với mục tiêu môn học Nhập môn Trí tuệ nhân tạo, đề tài "**Nhận diện chữ số viết tay bằng Machine Learning – Sử dụng thuật toán kNN**" đã được chúng em lựa chọn dựa trên các lý do cụ thể sau:
 - **Tính nền tảng và phù hợp với môn học:** Môn học trang bị cho sinh viên những kiến thức cơ bản về các thuật toán tìm kiếm, các khái niệm về học máy. Trong đó, thuật toán k-Nearest Neighbors (kNN) là một trong những thuật toán phân loại kinh điển trong số các bài toán ở lĩnh vực này.
 - **Tính ứng dụng thực tiễn:** Bài toán có ý nghĩa thực tiễn lớn, ứng dụng rộng rãi trong việc đọc mã bưu chính, séc ngân hàng, hoặc nhập liệu tự động. Việc triển khai kNN giúp sinh viên nắm vững nền tảng AI cơ bản, hiểu cách các thuật toán đơn giản có thể giải quyết các vấn đề hữu ích trong đời sống.

II. Giới thiệu đề tài

1. Lý do chọn đề tài:

- **Tính khả thi:** Đề tài này phù hợp với sinh viên năm 2 nhờ sự sẵn có của các tài nguyên quan trọng. Sinh viên có thể dễ dàng tiếp cận bộ dữ liệu MNIST – bộ dữ liệu chuẩn và phổ biến cho nhận diện chữ số viết tay (Yevhen Chyckarov et al., "Handwritten Digits Recognition Using SVM, KNN, RF and Deep Learning Neural Networks"). Đồng thời, các thư viện lập trình như Scikit-learn (cho kNN) và OpenCV (cho tiền xử lý ảnh) trong Python phù hợp với kiến thức sinh viên.
- **Tính mở rộng:** Đề tài này còn mở ra nhiều hướng phát triển tiềm năng trong tương lai. Sau khi hoàn thành mô hình cơ bản, sinh viên có thể tiếp tục khám phá các thuật toán học máy khác (như SVM, Random Forest), hoặc thậm chí là các mô hình học sâu (Deep Learning) để so sánh và cải thiện hiệu suất.

2. Mục tiêu và ý nghĩa đề tài:

a. Mục tiêu

- **Xây dựng hệ thống nhận diện chữ số viết tay đơn giản bằng cách sử dụng thuật toán học máy cổ điển k-Nearest Neighbors (kNN):** Mục tiêu cốt lõi của đề tài là triển khai một mô hình nhận diện từ đầu đến cuối. Chúng em sẽ tập trung vào việc triển khai các bước cơ bản của quy trình học máy, bao gồm tiền xử lý dữ liệu, trích xuất đặc trưng và áp dụng thuật toán kNN.
- **Ứng dụng viết chương trình nhận diện chữ số viết tay, chứng minh hiệu quả của kNN trong bài toán phân loại đơn giản:** Việc xây dựng một chương trình có khả năng nhận diện chữ số viết tay không chỉ minh chứng cho khả năng ứng dụng lý thuyết vào thực tiễn, mà còn giúp sinh viên nhận thấy hiệu quả của thuật toán k-NN.
- **Phân tích ảnh hưởng của các yếu tố độ chính xác của mô hình:** Không chỉ áp dụng thuật toán, đề tài còn khảo sát mức độ ảnh hưởng của các nhân tố đến hiệu quả mô hình.

2. Mục tiêu và ý nghĩa đề tài:

b. Ý nghĩa

- **Xác định các cơ sở kiến thức về học máy cổ điển (Classical Machine Learning):** Nghiên cứu và ứng dụng **thuật toán k-NN** - phương pháp đơn giản nhưng hiệu quả trong phân loại dữ liệu, giúp củng cố kiến thức nền tảng và hiểu sâu nguyên lý hoạt động của học máy cổ điển.
- **Ứng dụng viết chương trình nhận diện chữ số viết tay, chứng minh hiệu quả của kNN trong bài toán phân loại đơn giản:** Xây dựng chương trình nhận diện chữ số viết tay bằng Python (Scikit-learn, OpenCV), qua đó chứng minh hiệu quả của kNN trong bài toán phân loại hình ảnh dù thuật toán đơn giản.
- **Đánh giá kết quả của chương trình về hiệu suất nhận diện chữ số viết tay và ưu – nhược điểm của kNN trong thực tế:** Phân tích hiệu suất nhận diện chữ số viết tay, đo độ chính xác và nhận diện lỗi để hiểu rõ ưu điểm (đơn giản, dễ triển khai) và nhược điểm (hiệu suất với dữ liệu lớn) của kNN. Qua đó, rèn luyện tư duy phản biện và khả năng ra quyết định dựa trên dữ liệu.

3. Yêu cầu bài toán - Ý tưởng thực hiện

a. Yêu cầu bài toán

- **Đầu vào & Đầu ra:** xây dựng hệ thống tiếp nhận hình ảnh chữ số viết tay đơn lẻ (từ 0 đến 9) và trả về kết quả là chữ số được nhận diện.
- **Mô hình chính:** Áp dụng thuật toán học máy cổ điển k-Nearest Neighbors (kNN) để thực hiện nhiệm vụ phân loại này.
- **Hiệu suất & Phân tích:** Đánh giá độ chính xác của mô hình trên dữ liệu thử nghiệm và phân tích các yếu tố ảnh hưởng đến kết quả đạt được.

3. Yêu cầu bài toán - Ý tưởng thực hiện

b. Ý tưởng thực hiện

- Sử dụng bộ dữ liệu MNIST và tiền xử lý ảnh bằng OpenCV (chuyển xám, nhị phân hóa, chuẩn hóa kích thước).
- Xây dựng và huấn luyện mô hình kNN bằng thư viện Scikit-learn.
- Trích xuất đặc trưng bằng cách chuyển đổi ma trận pixel thành vector.
- Đánh giá hiệu suất mô hình, đồng thời khảo sát ảnh hưởng của k, độ đo khoảng cách và các kỹ thuật tiền xử lý.



HUST

PHẦN 3

TỔNG QUAN QUÁ TRÌNH NHẬN DIỆN CHỮ SỐ VIẾT TAY BẰNG HỌC MÁY



III. Tổng quan quá trình nhận diện chữ số viết tay bằng học máy

1. Khái niệm nhận diện chữ viết tay:

- Nhận diện chữ viết tay là bài toán trong lĩnh vực thị giác máy tính (**Computer Vision**), nhằm chuyển đổi hình ảnh các ký tự viết tay thành dạng số liệu có thể xử lý bằng máy tính. Theo Yevhen Chychkarov và cộng sự (2020), bài toán nhận diện chữ số viết tay là một ví dụ điển hình của học có giám sát (Supervised Learning), trong đó mô hình được huấn luyện để phân biệt giữa 10 lớp (từ 0 đến 9) dựa trên tập dữ liệu **MNIST [1]**. Quá trình xử lý thường bao gồm tiền xử lý ảnh (**chuyển ảnh về kích thước 28×28 pixel**), trích xuất đặc trưng (**biến ảnh thành vector 784 chiều**) và phân loại bằng các thuật toán như **KNN, SVM, Random Forest hoặc mạng nơ-ron sâu**. Đây là một bài toán kinh điển, được dùng rộng rãi trong đào tạo học máy vì tính trực quan, dễ hiểu và có bộ dữ liệu chuẩn để đánh giá mô hình.

[1] Yevhen Chychkarov, Anastasiia Serhiienko, Iryna Syrmamiikh, Anatolii Kargin. "Handwritten Digits Recognition Using SVM, KNN, RF and Deep Learning Neural Networks." *International Journal of Computer Science and Network Security (IJCSNS)*, Vol.20, No.10, pp. 13-19, October 2020.

2. Tổng quan học máy cổ điển (ML) và kNN:

a. Tổng quan về Học máy cổ điển (Classical Machine Learning - ML):

- Học máy là một lĩnh vực của Trí tuệ Nhân tạo (AI), cho phép máy tính học từ dữ liệu để đưa ra dự đoán hoặc quyết định mà không cần lập trình cụ thể từng bước. Thay vì viết các quy tắc cứng nhắc, máy học phát hiện mẫu trong dữ liệu (patterns) để giải quyết bài toán. (Mitchell, 1997).

“A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E .”

(Mitchell, T.M., 1997. Machine Learning, McGraw-Hill)

2. Tổng quan học máy cổ điển (ML) và kNN:

b. Thuật toán k-Nearest Neighbors (kNN):

- Thuật toán k-Nearest Neighbors (kNN) là một phương pháp phân loại phi tham số đơn giản nhưng hiệu quả, trong đó một điểm dữ liệu mới được phân loại dựa trên đa số nhãn của k điểm dữ liệu gần nhất trong tập huấn luyện (Hastie et al., 2009).
- Nguyên lý hoạt động:
- **Tính toán khoảng cách:** Khi một hình ảnh chữ số viết tay mới cần được nhận diện, kNN sẽ tính toán khoảng cách (ví dụ: khoảng cách Euclidean, Manhattan) từ hình ảnh đó đến tất cả các hình ảnh chữ số đã có trong tập dữ liệu huấn luyện.

3. Các bước chính trong bài toán ML:

Trong đề tài này, quy trình giải quyết bài toán nhận diện chữ số viết tay bằng học máy được thực hiện qua các bước chính như sau:

- **Dữ liệu đầu vào:** Hệ thống nhận diện sẽ làm việc với các hình ảnh chứa chữ số viết tay. Bài báo sử dụng bộ dữ liệu MNIST – một tập hợp các ảnh chữ số viết tay từ 0 đến 9, được đánh giá là "nổi tiếng và khá đầy đủ" để huấn luyện và kiểm thử mô hình.
- **Tiền xử lý dữ liệu:** Các hình ảnh thô cần được chuẩn bị trước khi đưa vào mô hình. Theo bài báo, các bước này bao gồm chia tỷ lệ (scaling) ảnh về kích thước 28×28 pixel và thực hiện các hoạt động lọc (filtering). Các thao tác này thường được thực hiện thông qua thư viện OpenCV. Mục đích là để làm sạch và chuẩn hóa dữ liệu, giúp thuật toán làm việc hiệu quả hơn.

3. Các bước chính trong bài toán ML:

- **Trích xuất đặc trưng và Biểu diễn:** Sau khi tiền xử lý, mỗi hình ảnh chữ số sẽ được biến đổi thành một dạng biểu diễn số học mà thuật toán có thể hiểu được. Phổ biến nhất là "duỗi thẳng" ma trận pixel của ảnh thành một vector một chiều (ví dụ: một ảnh 28×28 pixel sẽ trở thành vector 784 chiều). Các giá trị trong vector này chính là các đặc trưng của hình ảnh.
- **Phân loại:** Đây là bước cốt lõi nơi mô hình đưa ra dự đoán. Tại bước này, một thuật toán phân loại học máy sẽ được sử dụng để nhận diện các chữ số. Cụ thể, bài báo đã nghiên cứu và so sánh hiệu quả của kNN cùng các thuật toán khác như SVM, Random Forest và mạng nơ-ron trong việc phân loại chính xác hình ảnh chữ số vào 10 lớp tương ứng [2].

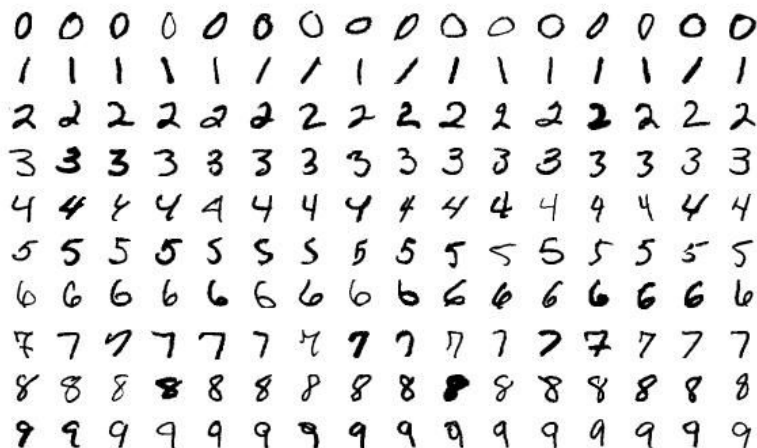
[2] Yevhen Chychkarov, Anastasiia Serhiienko, Iryna Syrmamiikh, Anatolii Kargin. "Handwritten Digits Recognition Using SVM, KNN, RF and Deep Learning Neural Networks." *International Journal of Computer Science and Network Security (IJCSNS)*, Vol.20, No.10, pp. 13-19, October 2020.

III. Tổng quan quá trình nhận diện chữ số viết tay bằng học máy

4. Tập dữ liệu MNIST:

a. Thông tin chung:

- Tập dữ liệu MNIST (Modified National Institute of Standards and Technology) là bộ dữ liệu hình ảnh kinh điển về chữ số viết tay, được phát triển bởi Yann LeCun và cộng sự từ nguồn dữ liệu của Viện Tiêu chuẩn Hoa Kỳ (NIST). Đây là tập dữ liệu chuẩn thường dùng để đánh giá các thuật toán nhận dạng chữ số trong học máy.



III. Tổng quan quá trình nhận diện chữ số viết tay bằng học máy

4. Tập dữ liệu MNIST:

b. Cấu trúc:

Tập bao gồm các hình ảnh chữ số từ 0 đến 9:

- Mỗi hình ảnh trong MNIST đều là ảnh xám (grayscale) và có kích thước chuẩn hóa là 28×28 pixel. Điều này có nghĩa là mỗi ảnh được biểu diễn bởi 784 điểm ảnh, mỗi điểm có một giá trị cường độ sáng.
- Để tiện cho việc xử lý bởi các thuật toán học máy, các hình ảnh này thường được "duỗi thẳng" (flatten) thành một vector một chiều với 784 giá trị pixel.
- Bộ dữ liệu được chia thành hai phần riêng biệt: một tập huấn luyện (training set) chứa khoảng 60,000 ảnh để mô hình học, và một tập kiểm thử (testing set) chứa khoảng 10,000 ảnh để đánh giá hiệu suất của mô hình trên dữ liệu mới).



HUST

PHẦN 4

CƠ SỞ LÝ THUYẾT MÔ HÌNH KNN TRONG NHẬN DIỆN CHỮ VIẾT TAY

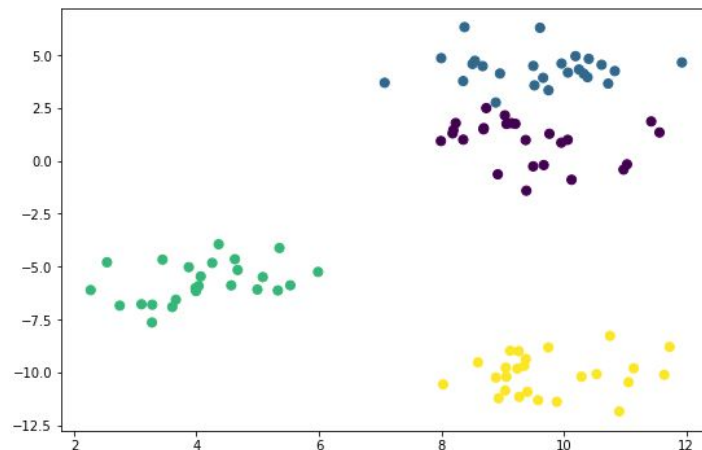


IV. Phân tích mô hình k-NN trong nhận diện chữ viết tay

1. Bài toán nhận dạng và vai trò của kNN:

a, Bài toán nhận dạng

- Nhận dạng chữ số viết tay là một bài toán trong lĩnh vực máy tính, nhằm phân loại hình ảnh các chữ số viết tay thành các nhãn tương ứng.
- Dữ liệu đầu vào là hình ảnh, đầu ra xác định đúng chữ số mà hình ảnh đó biểu diễn.
- Là một dạng bài toán phân loại thuộc nhóm học giám sát, có mục tiêu là ánh xạ đầu vào sang một nhãn đầu ra thuộc tập $\{0,1,...,9\}$.
- Dữ liệu đầu vào là ảnh số kích thước 28x28 pixel [3].



[3] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). *Gradient-based learning applied to document recognition*. *Proceedings of the IEEE*, 86(11), 2278–2324.

1. Bài toán nhận dạng và vai trò của kNN:

b, Vai trò kNN

kNN là thuật toán phân loại phi tham số, hoạt động theo nguyên lý “bỏ phiếu đa số” từ các điểm lân cận. Dữ liệu đầu vào được gán nhãn dựa trên đa số nhãn của k điểm dữ liệu gần nhất trong tập huấn luyện

[1].

Được lựa chọn vì:

- Mô hình đơn giản, dễ cài đặt và trực quan hóa.
- Phù hợp với sinh viên mới học về trí tuệ nhân tạo.
- Cho kết quả khá tốt với bài toán MNIST.
- Là thuật toán cơ sở để so sánh với các mô hình phức tạp hơn.

2. Thuật toán kNN - Nguyên lý hoạt động

a, Quy trình hoạt động

Bước 1: Tính khoảng cách

- Khi cần phân loại một dữ liệu mới, kNN sẽ đi tính khoảng cách từ điểm đó tới tất cả các điểm dữ liệu trong tập.
- Các loại khoảng cách thường được sử dụng: Euclid, Manhattan

Bước 2: Chọn k điểm lân cận gần nhất

- Sắp xếp các điểm dữ liệu đó theo thứ tự tăng dần khoảng cách đến điểm mới.
- Lấy ra k điểm gần nhất, giá trị k thường là một siêu tham số cần chọn trước.

Bước 3: Phân loại theo nguyên tắc đa số

- Nhìn vào nhãn (label) của k điểm lân cận.
- Gán nhãn cho điểm mới dựa trên nhãn chiếm đa số trong các điểm lân cận.

IV. Phân tích mô hình k-NN trong nhận diện chữ viết tay

2. Thuật toán kNN - Nguyên lý hoạt động

a, Quy trình hoạt động

Ví dụ: Giả sử ta có tập dữ liệu D có gắn nhãn gồm 15 điểm như trên ảnh

E2		f_x	$=\text{SQRT}((A18-A2)^2 + (B18-B2)^2)$			
	A	B	C	D	E	F
1	x	y	label		Distance	
2	2	17	+		8.06	
3	3	11	-		11.40	
4	4	23	+		23.35	
5	1	12	+		12.04	
6	2	6	-		6.32	
7	6	2	+		6.32	
8	11	4	-		11.70	
9	3	24	-		24.19	
10	14	2	-		14.14	
11	9	4	+		9.85	
12	24	23	+		33.24	
13	7	6	+		9.22	
14	23	4	-		23.35	
15	14	16	-		21.26	
16	12	3	+		12.37	
17						
18	3	9				

E2	$\sqrt{f_x}$	=SQRT((A18-A2)^2 + (B18-B2)^2)					
	A	B	C	D	E	F	G
1	x	y	label		Distance	Rank	
2	2	17	+		8.06	3	
3	3	11	-		11.40		
4	4	23	+		23.35		
5	1	12	+		12.04		
6	2	6	-		6.32	1	
7	6	2	+		6.32	2	
8	11	4	-		11.70		
9	3	24	-		24.19		
10	14	2	-		14.14		
11	9	4	+		9.85	5	
12	24	23	+		33.24		
13	7	6	+		9.22	4	
14	23	4	-		23.35		
15	14	16	-		21.26		
16	12	3	+		12.37		
17							
18	3	9					

2. Thuật toán kNN - Nguyên lý hoạt động

a, Quy trình hoạt động

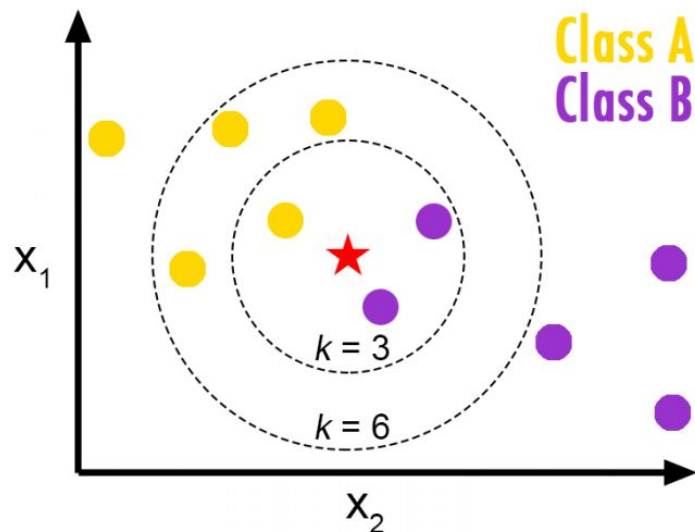
Ví dụ: Giả sử ta có tập dữ liệu D có gắn nhãn gồm 15 điểm như trên ảnh

1. Điểm cần dự đoán nhãn A(3,9)
2. Ta tính khoảng cách từ điểm A đến các điểm dữ liệu trong D bằng công thức Euclidian.
3. Ta chọn $K=5$, và tìm ra 5 điểm có khoảng cách gần với điểm A nhất.
4. Trong 5 điểm ta thấy có 4 điểm mang nhãn (+) và 1 điểm mang nhãn (-).
5. Vậy ta có thể đưa ra kết luận là điểm A cần dự đoán mang nhãn (+).

2. Thuật toán kNN - Nguyên lý hoạt động

b, Đặc điểm của kNN

- KNN sẽ không phải xây dựng mô hình trước khi nhận dữ liệu mới, nó sẽ chỉ “lưu trữ” toàn bộ dữ liệu huấn luyện.
- Khi cần dự đoán nhãn cho một điểm mới, kNN sẽ thực hiện phép tính khoảng cách và tìm kiếm k lân cận ngay tại thời điểm đó luôn.



3. Cách tính khoảng cách

a, Khoảng cách Euclid (Euclidean Distance)

Khái niệm: Khoảng cách Euclidean là độ đo khoảng cách thông thường trong không gian hình học, biểu diễn độ dài đường thẳng nối hai điểm. Đây là công thức khoảng cách được sử dụng rộng rãi nhất trong kNN vì tính trực quan của nó [5].

Công thức: Đối với hai điểm $p=(p_1,p_2,...,p_n)$ và $q=(q_1,q_2,...,q_n)$ trong không gian n chiều, khoảng cách Euclidean được tính bằng căn bậc hai của tổng bình phương các hiệu giữa các tọa độ tương ứng:

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_n - q_n)^2}$$

4. Các yếu tố ảnh hưởng đến hiệu suất

Giá trị của k:

- k quá nhỏ → dễ bị nhiễu, gây quá khớp (overfitting).
- k quá lớn → dễ bỏ qua đặc trưng quan trọng, gây dưới khớp (underfitting).
- Thường chọn k lẻ (để tránh hòa phiếu) và tối ưu thông qua kiểm thử chéo (cross-validation) [5], [6].

Hàm khoảng cách:

- Ảnh hưởng trực tiếp đến việc chọn hàng xóm.
- Khoảng cách phù hợp giúp tăng độ chính xác phân loại [5], [6].

[5] Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2nd ed.). Springer.

[6]: Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Nguồn tham khảo:

- [3] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). *Gradient-based learning applied to document recognition. Proceedings of the IEEE*, 86(11), 2278–2324.
- [4] Yevhen Chychkarov, Anastasiia Serhiienko, Iryna Syrmamiikh, Anatolii Kargin. "Handwritten Digits Recognition Using SVM, KNN, RF and Deep Learning Neural Networks." *International Journal of Computer Science and Network Security (IJCSNS)*, Vol.20, No.10, pp. 13-19, October 2020.
- [5] Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2nd ed.). Springer.
- [6]: Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.



HUST

PHẦN 5

CHƯƠNG TRÌNH NHẬN DIỆN CHỮ SỐ VIẾT TAY



1. Tổng quan bài toán. Cấu trúc chương trình:

a) Tổng quan bài toán và sơ đồ khối:

Phần 1: Huấn luyện mô hình KNN

- Đọc ảnh huấn luyện digits.png. Sau đó chia ảnh thành các ô nhỏ (ảnh chữ số 20x20)
- Gắn nhãn cho từng chữ số và Huấn luyện mô hình KNN với dữ liệu train.

Phần 2: Nhập ảnh và tiền xử lý

- Đọc ảnh cần nhận dạng và kiểm tra
- Xử lý ảnh: cân bằng histogram, làm mờ Gaussian, nhị phân hóa (thresholding), tìm contour, cắt ảnh và căn giữa, resize về 20x20

Phần 3: Nhận dạng và hiển thị kết quả

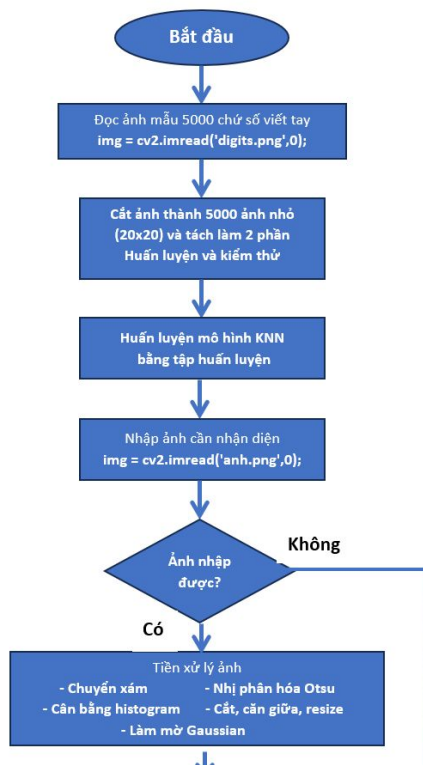
- Dùng mô hình KNN dự đoán chữ số trong ảnh sau tiền xử lý.
- Hiển thị ảnh sau xử lý và kết quả nhận dạng lên màn hình. Cuối cùng nhấn Enter để kết thúc chương trình

1. Tổng quan bài toán. Cấu trúc chương trình

b) Phân tích chức năng

- **Huấn luyện KNN:** Xây dựng mô hình KNN từ ảnh digits.png, chuẩn hóa dữ liệu thành vector 400 chiều (20x20 pixel).
- **Tiền xử lý ảnh đầu vào:** Chuẩn bị ảnh bằng cách cân bằng histogram, làm mờ Gaussian, nhị phân hóa và trích xuất chữ số.
- **Chuẩn hóa kích thước:** Tạo ảnh vuông nền trắng và resize về 20x20 để giữ tỷ lệ.
- **Nhận dạng:** Dùng mô hình KNN đã huấn luyện để dự đoán chữ số từ ảnh đã xử lý.
- **Hiển thị kết quả:** Trình bày ảnh sau xử lý và kết quả nhận dạng, chờ người dùng kết thúc.

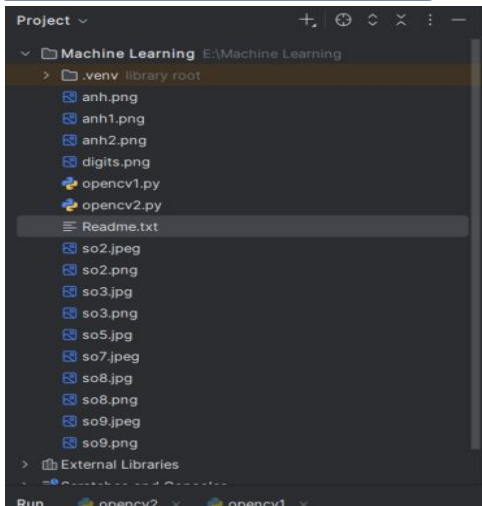
Sơ đồ khối chương trình



V. Chương trình nhận diện chữ viết tay

1. Tổng quan bài toán. Cấu trúc chương trình

c) Cấu trúc chương trình.



Hình 5.1. Cấu trúc chương trình

- Thư mục **Machine Learning**: thư mục chứa toàn bộ file dự án
- Thư mục **.venv**: chứa các file cài đặt module, cấu hình của OpenCV
- File **opencv1.py**: File Python chứa phần huấn luyện mô hình dựa trên tập dữ liệu MNIST
- File **opencv2.py**: File Python chứa phần huấn luyện mô hình dựa trên tập dữ liệu MNIST và phần nhập ảnh, tiền xử lý ảnh và nhận diện chữ số viết tay dựa trên ảnh nhập vào của người dùng
- File **digits.png**: Chứa 5000 chữ số viết tay
- File **readme.txt**: Chứa thông tin danh sách thành viên nhóm

2. Thư viện và nền tảng

Thư viện OpenCV (cv2):

Xử lý ảnh (đọc, biến đổi ảnh, tiền xử lý)

Thư viện cung cấp thuật toán KNN để huấn luyện và nhận dạng chữ số viết tay

Thư viện NumPy:

Xử lý và lưu trữ dữ liệu ảnh dưới dạng mảng số (arrays)

Thao tác chia tách, reshape dữ liệu phục vụ huấn luyện và nhận dạng

Nền tảng Python:

Ngôn ngữ lập trình chủ đạo để viết toàn bộ code

Hỗ trợ các thư viện OpenCV và NumPy để dàng tích hợp

V. Chương trình nhận diện chữ viết tay

3. Phân tích và thiết kế hệ thống

a) Tiền xử lý hình ảnh

```
def preprocess_image(image_path):  
    """usage  
    img_input = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)  
    if img_input is None:  
        raise ValueError("Không thể đọc ảnh!")  
  
    img_eq = cv2.equalizeHist(img_input)  
    blurred = cv2.GaussianBlur(img_eq, (3, 3), sigmaX=0)  
  
    _, binary = cv2.threshold(blurred, thresh=0, maxval=255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)  
  
    contours, _ = cv2.findContours(255 - binary, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)  
  
    largest = max(contours, key=cv2.contourArea)  
    x, y, w, h = cv2.boundingRect(largest)  
    roi = binary[y:y+h, x:x+w]  
  
    # làm ảnh vuông  
    size = max(w, h)  
    square = 255 * np.ones((size, size), dtype=np.uint8)  
    x_offset = (size - w) // 2  
    y_offset = (size - h) // 2  
    square[y_offset:y_offset+h, x_offset:x_offset+w] = roi
```

Hình 5.2. Một đoạn code tiền xử lý hình ảnh

V. Chương trình nhận diện chữ viết tay

3. Phân tích và thiết kế hệ thống

a) Tiền xử lý hình ảnh

Đọc ảnh từ đường dẫn file 'image_path' dưới dạng ảnh xám (grayscale)

```
def preprocess_image(image_path):  
    """usage  
    img_input = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)  
    if img_input is None:  
        raise ValueError("Không thể đọc ảnh!")
```

Mục đích: đảm bảo ảnh đầu vào đúng định dạng

Hình 5.3. đoạn code đọc ảnh từ đường dẫn file

- Đọc ảnh từ đường dẫn file 'image_path' dưới dạng ảnh xám (grayscale).
- `cv2.IMREAD_GRAYSCALE = 0`, ảnh chỉ có 1 kênh màu, giúp giảm độ phức tạp xử lý.
- Kiểm tra xem ảnh có đọc thành công không, nếu không thì báo lỗi và dừng.

3. Phân tích và thiết kế hệ thống

a) Tiền xử lý hình ảnh

Các câu lệnh cân bằng histogram, chỉnh độ sáng, giảm nhiễu và nhị phân hóa ảnh:

```
img_eq = cv2.equalizeHist(img_input)
blurred = cv2.GaussianBlur(img_eq, ksize: (3, 3), sigmaX: 0)

_, binary = cv2.threshold(blurred, thresh: 0, maxval: 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

contours, _ = cv2.findContours(255 - binary, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

largest = max(contours, key=cv2.contourArea)
x, y, w, h = cv2.boundingRect(largest)
roi = binary[y:y+h, x:x+w]
```

Hình 5.4. Đoạn code thực hiện cân bằng histogram, chỉnh sáng, giảm nhiễu và nhị phân hóa ảnh

3. Phân tích và thiết kế hệ thống

a) Tiền xử lý hình ảnh

Các câu lệnh cân bằng histogram, chỉnh độ sáng, giảm nhiễu và nhị phân hóa ảnh:

- `img_eq = cv2.equalizeHist(img_input)`

Cân bằng Histogram: Tăng độ tương phản, giúp chữ số rõ nét hơn

- `blurred = cv2.GaussianBlur(img_eq, (3, 3), 0)`

Làm mờ Gaussian: Giảm nhiễu, làm mịn ảnh để nhị phân hóa tốt hơn.

- `_ , binary = cv2.threshold(blurred, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)`

Nhị phân hóa Otsu: Chuyển ảnh sang đen trắng, tự động tách nền và chữ số.

3. Phân tích và thiết kế hệ thống

a) Tiền xử lý hình ảnh

Các câu lệnh cân bằng histogram, chỉnh độ sáng, giảm nhiễu và nhị phân hóa ảnh:

- `contours, _ = cv2.findContours(255 - binary, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)`

Tìm Contour: Phát hiện các đường viền của chữ số (trên ảnh đã đảo ngược để chữ số thành màu trắng).

- `largest = max(contours, key=cv2.contourArea)`

Chọn Contour lớn nhất: Xác định chữ số chính dựa trên diện tích lớn nhất.

- `x, y, w, h = cv2.boundingRect(largest)`

Tính Bounding Box: Lấy tọa độ và kích thước vùng chữ số.

- `roi = binary[y:y+h, x:x+w]`

Cắt vùng ROI: Trích xuất chính xác vùng chứa chữ số.

3. Phân tích và thiết kế hệ thống

a) Tiền xử lý hình ảnh

Làm vuông ảnh:

```
# Làm ảnh vuông
size = max(w, h)
square = 255 * np.ones(shape: (size, size), dtype=np.uint8)
x_offset = (size - w) // 2
y_offset = (size - h) // 2
square[y_offset:y_offset+h, x_offset:x_offset+w] = roi
```

Hình 5.5. Đoạn code làm ảnh vuông

3. Phân tích và thiết kế hệ thống

a) Tiền xử lý hình ảnh

Làm vuông ảnh:

- $x_offset = (size - w) // 2$

Tính dịch ngang: Xác định khoảng cách để căn giữa chữ số theo chiều ngang.

- $y_offset = (size - h) // 2$

Tính dịch dọc: Xác định khoảng cách để căn giữa chữ số theo chiều dọc.

- $square[y_offset:y_offset+h, x_offset:x_offset+w] = roi$

Đặt vào giữa: Gán vùng chữ số vào chính giữa ảnh vuông nền trắng.

3. Phân tích và thiết kế hệ thống

b) Chuyển đổi hình ảnh thành vector

```
resized = cv2.resize(square, dsize: (20, 20), interpolation=cv2.INTER_AREA)  
  
return resized.reshape(1, 400).astype(np.float32), resized
```

Hình 5.6. Đoạn code chuyển đổi hình ảnh thành vector

3. Phân tích và thiết kế hệ thống

b) Chuyển đổi hình ảnh thành vector

- `resized = cv2.resize(square, (20, 20), interpolation=cv2.INTER_AREA)`

Thay đổi kích thước: Thu nhỏ/phóng to ảnh về **20x20 pixel**; dùng **INTER_AREA** để giữ chất lượng tốt nhất.

- `return resized.reshape(1, 400).astype(np.float32), resized`

`resized.reshape(1, 400)`: Chuyển ảnh 20x20 pixel thành **vector 400 chiều** để đưa vào mô hình KNN.

`.astype(np.float32)`: Chuyển đổi dữ liệu sang **kiểu số thực 32-bit** để tương thích với KNN.

Hàm trả về: Cung cấp **vector 400 chiều** cho KNN và **ảnh 20x20** để hiển thị.

V. Chương trình nhận diện chữ viết tay

3. Phân tích và thiết kế hệ thống

b) Chuyển đổi hình ảnh thành vector

```
img = cv2.imread('digits.png',0)
cells = [np.hsplit(row, indices_or_sections: 100) for row in np.vsplit(img, indices_or_sections: 50)]
x = np.array(cells)

train = x[:, :50].reshape(-1, 400).astype(np.float32)
test = x[:, 50:100].reshape(-1, 400).astype(np.float32)

k = np.arange(10)
train_labels = np.repeat(k, repeats: 250)[: , np.newaxis]
test_labels = np.repeat(k, repeats: 250)[: , np.newaxis]

knn = cv2.ml.KNearest_create()
knn.train(train, cv2.ml.ROW_SAMPLE, train_labels)
```

Hình 5.7. Kết quả của chuyển đổi hình ảnh sang vector

3. Phân tích và thiết kế hệ thống

c) Gắn nhãn vector

```
# PHẦN 1: HUẤN LUYỆN
img = cv2.imread('digits.png',0);
cells = [np.hsplit(row, indices_or_sections: 100) for row in np.vsplit(img, indices_or_sections: 50)]
x = np.array(cells)

train = x[:, :50].reshape(-1, 400).astype(np.float32)
test = x[:, 50:100].reshape(-1, 400).astype(np.float32)

k = np.arange(10)
```

Hình 5.7. Đoạn code gắn nhãn vector

3. Phân tích và thiết kế hệ thống

c) Gắn nhãn vector

- `img = cv2.imread('digits.png', 0)`
Đọc ảnh **digits.png** (chứa chữ số viết tay) ở chế độ đen trắng.
- `cells = [np.hsplit(row, 100) for row in np.vsplit(img, 50)]`
Chia ảnh thành **50x100** ô nhỏ (20x20 pixel), mỗi ô là một chữ số.
- `x = np.array(cells)`
Chuyển các ô thành mảng **numpy 3D** có kích thước (50, 100, 20, 20).
- `train = x[:, :50].reshape(-1, 400).astype(np.float32)`
Lấy **2500 ảnh đầu tiên** (50 hàng x 50 cột) để huấn luyện.
Mỗi ảnh 20x20 được chuyển thành **vector 400 chiều** (-1, 400).
Chuyển kiểu dữ liệu sang **float32** cho KNN.

3. Phân tích và thiết kế hệ thống

d) Huấn luyện và kiểm thử mô hình

Huấn luyện

```
k = np.arange(10)
train_labels = np.repeat(k, repeats: 250)[: , np.newaxis]
test_labels = np.repeat(k, repeats: 250)[: , np.newaxis]

knn = cv2.ml.KNearest_create()
knn.train(train, cv2.ml.ROW_SAMPLE, train_labels)
```

Hình 5.8. Đoạn code thực hiện huấn luyện

3. Phân tích và thiết kế hệ thống

d) Huấn luyện và kiểm thử mô hình

Huấn luyện

- `knn = cv2.ml.KNearest_create()`
Tạo đối tượng KNN để huấn luyện và phân loại.
- `knn.train(train, cv2.ml.ROW_SAMPLE, train_labels)`
Huấn luyện KNN với dữ liệu train (vector 400 chiều của chữ số) và `train_labels` (nhãn 0-9).
- `np.repeat(k, 250)[:, np.newaxis]`
Tạo `train_labels`: Lặp lại mỗi chữ số từ 0-9 250 lần, tạo mảng nhãn 2500 phần tử dạng cột ((2500,1)).
- `[:, np.newaxis]`:
Chuyển mảng nhãn **1 chiều thành 2 chiều (dạng cột)**, phù hợp với yêu cầu của mô hình.

=> Kết quả của bước này là mô hình KNN được huấn luyện, sẵn sàng để dự đoán nhãn cho các ảnh mới.

3. Phân tích và thiết kế hệ thống

d) Huấn luyện và kiểm thử mô hình

Kiểm thử

```
ret, result, neighbours, dist = knn.findNearest(test, k=5)
matches = result == test_labels.astype(np.float32)
correct = np.count_nonzero(matches)
accuracy = correct * 100.0 / result.size
print("Độ chính xác trên tập kiểm thử: {:.2f}%".format(accuracy))
```

Hình 5.9. Đoạn code thực hiện kiểm thử

3. Phân tích và thiết kế hệ thống

d) Huấn luyện và kiểm thử mô hình

Kiểm thử

- `knn.findNearest(...)`: dự đoán nhãn cho từng ảnh trong tập kiểm thử test bằng $k = 5$.
- `result`: chứa nhãn dự đoán, `test_labels` là nhãn thật.
- So sánh `result` với `test_labels` để tạo mảng `matches` (đúng/sai).
- `np.count_nonzero(matches)`: đếm số dự đoán đúng.
- Tính độ chính xác = $(\text{số đúng} / \text{tổng ảnh kiểm thử}) \times 100$.
- In ra độ chính xác dưới dạng phần trăm với 2 chữ số thập phân.



HUST

PHẦN 6

KẾT QUẢ ĐỀ TÀI



1. Kết quả thực nghiệm:

- Để đánh giá khả năng của mô hình k-NN trong việc nhận diện chữ số viết tay, chúng em sử dụng tập kiểm thử gồm 2.500 ảnh được tách từ tập **digits.png** (mỗi chữ số từ 0 đến 9 có 250 ảnh).
- Phương pháp đánh giá: **Độ chính xác (Accuracy)**: Tính bằng tỉ lệ số dự đoán đúng trên tổng số mẫu kiểm thử.
 - Mô hình đạt **82%** độ chính xác trên tập kiểm thử với tham số $k = 5$, sử dụng khoảng cách Euclidean.
- Chúng em cũng tiến hành thử nghiệm nhỏ với 50 ảnh viết tay của người dùng được in ra, viết tay trên Paint rồi lưu ảnh lại dạng anh_so.png và xử lý qua hàm preprocess_image().

Kết quả: Dự đoán đúng 41 ảnh, sai 9 ảnh.

1. Kết quả thực nghiệm:

- Một số lỗi cụ thể:
 - + Chữ số 4 bị nhầm thành 9 trong 3 trường hợp.
 - + Chữ 5 bị nhầm thành 3 trong 2 trường hợp
 - + Chữ 7 đôi khi bị nhầm với 1 do nét gạch ngang không rõ.
- Số ảnh dự đoán đúng: 41 / 50
 - + Số ảnh sai: 9 / 50
 - + Độ chính xác:

$$\frac{41}{50} \times 100\% = 82.00\%$$

- Nhận xét tổng quát:
 - + Mô hình k-NN hoạt động hiệu quả trên tập ảnh số học sinh viết tay nếu ảnh được tiền xử lý tốt.
 - + Tuy nhiên, độ chính xác giảm khi ảnh mờ, bị lệch sáng hoặc chữ viết quá phá cách.
 - + Độ chính xác 82,00% là chấp nhận được với mô hình đơn giản như k-NN. Để cải thiện, nhóm đề xuất có thể thử chuyển sang mô hình SVM hoặc CNN trong tương lai.

VI. Kết quả đề tài

Thật/Đoán	0	1	2	3	4	5	6	7	8	9
0	4	0	0	0	0	0	0	0	0	0
1	0	5	0	0	0	0	0	0	0	1
2	0	0	4	0	0	0	0	0	0	1
3	0	0	0	4	0	0	1	0	0	0
4	0	0	0	4	0	0	0	0	0	0
5	0	0	0	1	0	3	0	0	0	1
6	0	0	0	0	0	0	3	0	1	0
7	0	1	0	0	0	0	0	3	0	0
8	0	0	0	0	0	0	0	0	3	1
HUST 9	0	0	0	0	1	0	0	0	0	4

VI. Kết quả đề tài

1. Kết quả thực nghiệm:

- Số đoán đúng (đường chéo chính):

$$+ 4 + 4 + 4 + 4 + 4 + 3 + 3 + 3 + 3 + 4 = 41$$

$$+ \text{Số đoán sai: } 50 - 41 = 9$$

+ Độ chính xác:

$$\frac{41}{50} \times 100\% = 82.00\%$$

- Một số lỗi nhầm lẫn phổ biến gồm:

Số 1 đôi khi bị nhận thành 9

Số 5 nhầm với 3

Số 6 nhầm thành 8

- Những lỗi này thường xảy ra do nét viết tay không rõ ràng hoặc khác biệt với tập huấn luyện gốc (chữ số nền trắng – mực đen của digits.png).

VI. Kết quả đề tài

Chỉ số đánh giá theo từng lớp:

Lớp 0:

$$\text{Precision} = 4 / (4 + 0) = 1.00$$

$$\text{Recall} = 4 / (4 + 0) = 1.00$$

$$\text{F1-score} = 2 \times 1.00 \times 1.00 / (1.00 + 1.00) = 1.00$$

Lớp 1:

$$\text{Precision} = 5 / (5 + 0) = 1.00$$

$$\text{Recall} = 5 / (5 + 0) = 1.00$$

$$\text{F1-score} = 2 \times 1.00 \times 1.00 / (1.00 + 1.00) = 1.00$$

Lớp 2:

$$\text{Precision} = 4 / (4 + 0) = 1.00$$

$$\text{Recall} = 4 / (4 + 0) = 1.00$$

$$\text{F1-score} = 2 \times 1.00 \times 1.00 / (1.00 + 1.00) = 1.00$$

Lớp 3:

$$\text{Precision} = 4 / (4 + 0) = 1.00$$

$$\text{Recall} = 4 / (4 + 0) = 1.00$$

$$\text{F1-score} = 2 \times 1.00 \times 1.00 / (1.00 + 1.00) = 1.00$$

Lớp 4:

$$\text{Precision} = 4 / (4 + 0) = 1.00$$

$$\text{Recall} = 4 / (4 + 0) = 1.00$$

$$\text{F1-score} = 2 \times 1.00 \times 1.00 / (1.00 + 1.00) = 1.00$$

Lớp 5:

$$\text{Precision} = 3 / (3 + 1) = 0.75$$

$$\text{Recall} = 3 / (3 + 1) = 0.75$$

$$\text{F1-score} = 2 \times 0.75 \times 0.75 / (0.75 + 0.75) = 0.75$$

VI. Kết quả đề tài

Chỉ số đánh giá theo từng lớp:

Lớp 6:

$$\text{Precision} = 3 / (3 + 0) = 1.00$$

$$\text{Recall} = 3 / (3 + 1) = 0.75$$

$$\text{F1-score} = 2 \times 1.00 \times 0.75 / (1.00 + 0.75) \approx 0.86$$

Lớp 7:

$$\text{Precision} = 3 / (3 + 0) = 1.00$$

$$\text{Recall} = 3 / (3 + 0) = 1.00$$

$$\text{F1-score} = 2 \times 1.00 \times 1.00 / (1.00 + 1.00) = 1.00$$

Lớp 8

$$\text{Precision} = 3 / (3 + 1) = 0.75$$

$$\text{Recall} = 3 / (3 + 0) = 1.00$$

$$\text{F1-score} = 2 \times 0.75 \times 1.00 / (0.75 + 1.00) \approx 0.86$$

LLớp 9:

$$\text{Precision} = 4 / (4 + 1) = 0.80$$

$$\text{Recall} = 4 / (4 + 1) = 0.80$$

$$\text{F1-score} = 2 \times 0.80 \times 0.80 / (0.80 + 0.80) = 0.80$$

$$\text{Độ chính xác tổng thể: } 41 / 50 = 82.00\%$$

VI. Kết quả đề tài

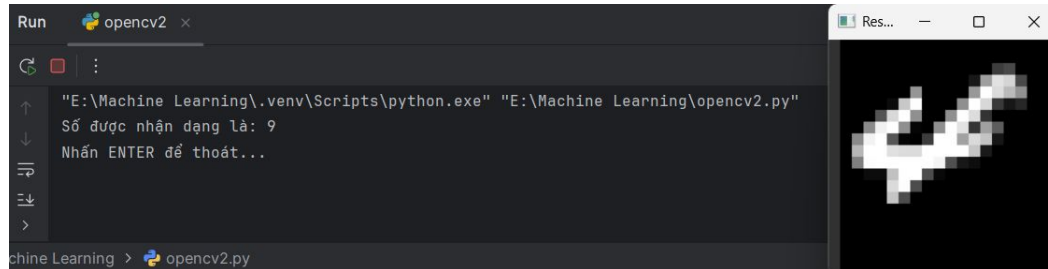
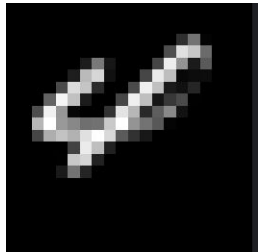
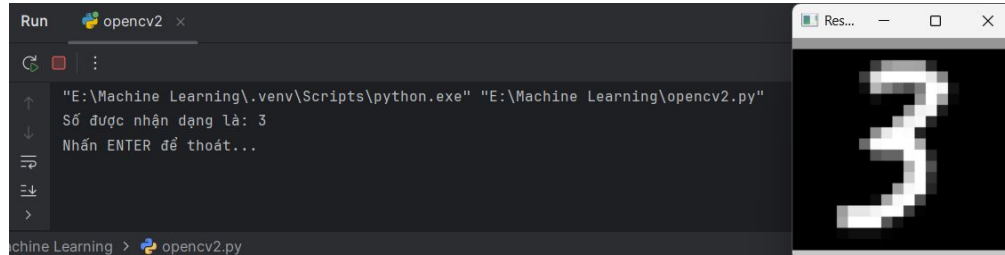
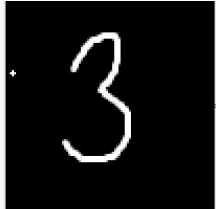
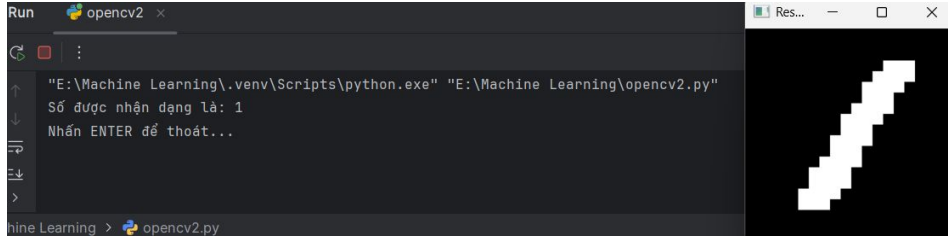
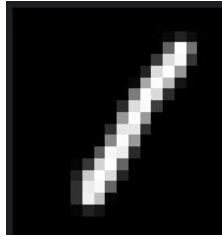
1. Kết quả thực nghiệm:

- Đề tài đã thành công trong việc xây dựng hệ thống nhận diện chữ số bằng kNN, đạt độ chính xác 82.00% trên tập kiểm thử MNIST, cho thấy kNN là một lựa chọn tốt cho bài toán phân loại hình ảnh cơ bản
- Kết quả thực nghiệm khẳng định giá trị k và các bước tiền xử lý dữ liệu ảnh hưởng lớn đến hiệu suất mô hình, giúp tối ưu hóa khả năng nhận diện của kNN.
- Hạn chế của kNN trên dữ liệu lớn: Dù hiệu quả, kNN bộc lộ nhược điểm về tốc độ và tài nguyên khi xử lý tập dữ liệu lớn với số chiều cao như MNIST, đồng thời nhạy cảm với dữ liệu nhiễu.

2. Hướng phát triển trong tương lai

- So sánh với các thuật toán phức tạp hơn: Mở rộng nghiên cứu bằng cách triển khai và so sánh kNN với các mô hình học máy và học sâu khác như SVM, Random Forest, hoặc mạng nơ-ron tích chập (CNN) trên cùng bài toán [1].
- Cải thiện tiền xử lý và trích xuất đặc trưng: Nghiên cứu và áp dụng các kỹ thuật tiền xử lý ảnh nâng cao

VI. Kết quả đề tài



VI. Kết quả đề tài

Run opencv2 x

↺

↻

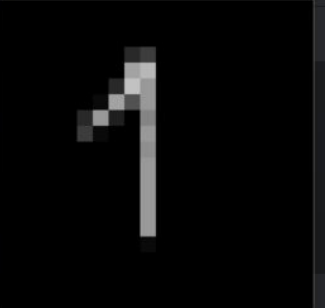
⏏

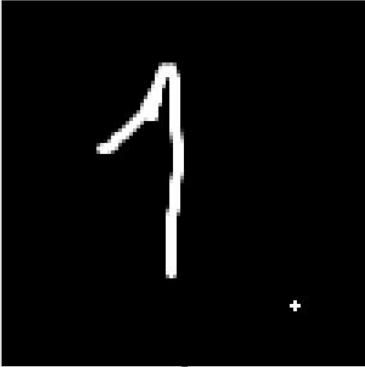
⋮

```
"E:\Machine Learning\.venv\Scripts\python.exe" "E:\Machine Learning\opencv2.py"  
Số được nhận dạng là: 1  
Nhấn ENTER để thoát...
```

Machine Learning > opencv2.py

Res...





Run opencv2 x

↺

↻


⏏

⋮

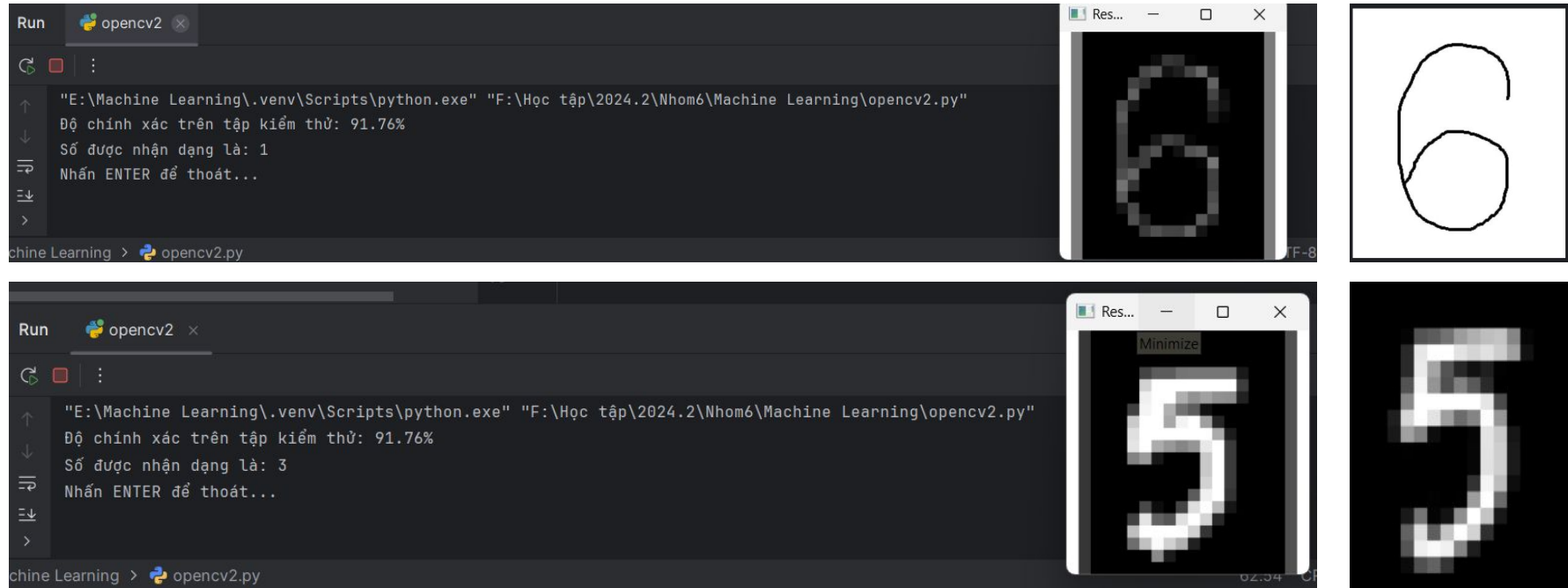
```
"E:\Machine Learning\.venv\Scripts\python.exe" "E:\Machine Learning\opencv2.py"  
Số được nhận dạng là: 3  
Nhấn ENTER để thoát...
```

Machine Learning > opencv2.py

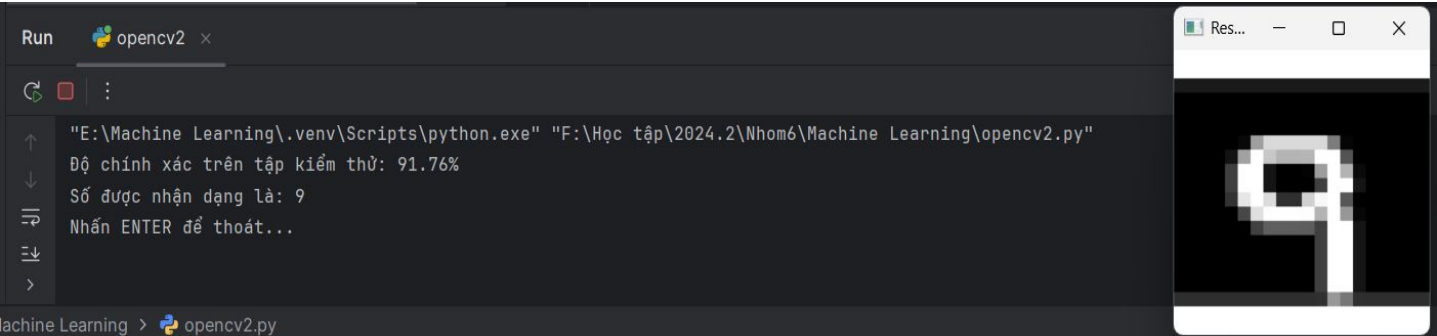
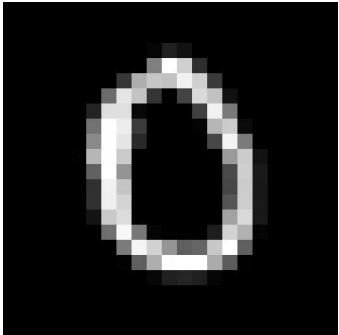
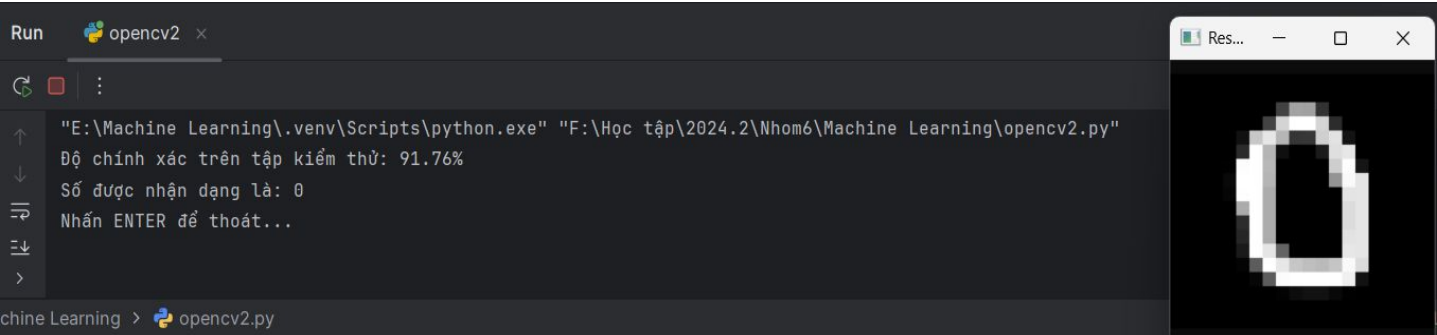
Res...



VI. Kết quả đề tài



VI. Kết quả đề tài



A decorative graphic on the left side of the slide. It features a dark blue background with a large, stylized circular pattern composed of many small red dots. The dots are arranged in concentric, slightly offset rings, creating a sense of depth and movement. In the center of this pattern, the word "HUST" is written in a bold, white, sans-serif font.

HUST

THANK YOU !

