ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN TỬ - VIỄN THÔNG

HỌC PHẦN TRÍ TUỆ NHÂN TẠO

TIỂU LUẬN CUỐI KỲ: XÂY DỰNG HỆ THỐNG AI TỰ ĐỘNG PHÂN LOẠI LÁ CÂY

Sinh viên thực hiện	:	Đinh Văn Quang	20DT1
Nhóm 10		Hà Phước Phúc	20DT2
		Nguyễn Văn Quý	21DT2
Lớp học phần	:	20.38	
Giảng viên hướng dẫn	:	TS. Hoàng Lê Uyên Thục	

Đà Nẵng, Ngày 6 tháng 11 năm 2024

MỤC LỤC

Bảng 1. *Phân công chi tiết công việc từng thành viên*

Thành viên	Công việc	Mức độ đóng góp
Đinh Văn Quang	- Sưu tầm 50 ảnh lá mỏ quạ, chuyển ảnh gốc thành ảnh xám và nhị phân. - Gán nhãn dữ liệu cho 250 lá cây. - Code trích xuất đặc trưng HOG, phân loại lá cây bằng thuật toán ANN với đặc trưng HOG và Hu, đánh giá mô hình sử dụng phương pháp K-fold cross validation. - Làm báo cáo phần đặc trưng HOG, thuật toán ANN, phương pháp đánh giá K-fold cross-validation và dùng KNN để phân loại lá cây dựa vào đặc trưng HOG và Hu.	30%
Hà Phước Phúc	- Tìm hiểu và sưu tầm dữ liệu 50 ảnh lá cây sầu đâu - Xử lý dữ liệu ảnh RGB sang ảnh xám và nhị phân -Viết Code trích đặc trưng Hu và HOG cho dữ liệu 250 ảnh lá cây - Viết Code cho mô hình phân loại KNN, ANN với đặc trưng Hu và HOG (4 mô hình) - Viết Code đánh giá chi tiết phần phân loại ANN với Hu và HOG (3 mô hình) - So sánh các đặc trưng. So sánh các thuật toán ML - Soạn và viết báo cáo tổng thể	35%
Nguyễn Văn Quý	-Sưu tầm 50 ảnh lá chanh. -Đánh số thứ tự từ 001-250 cho dữ liệu, chuyển ảnh gốc thành ảnh xám và nhị phân. -Code trích xuất đặc trưng mô men Hu, đặc trưng HOG của bộ 250 dữ liệu. -Code phân loại lá cây bằng thuật toán KNN+Hu, KNN+HOG, ANN+Hu, ANN+HOG. -Code đánh giá k-fold của các thuật toán. -Làm báo cáo.	35%

Mức độ đóng góp của cả nhóm (100%) là tổng của các thành viên

1. Giới thiệu đề tài

1.1 Tính cấp thiết

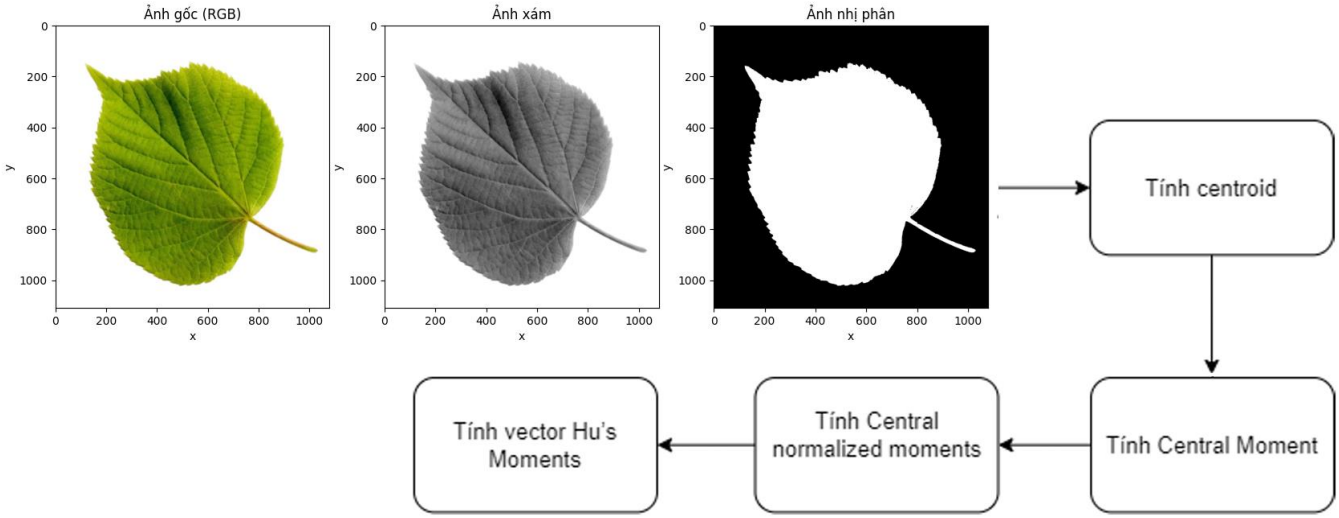
- Gần đây, lĩnh vực nhận dạng hình ảnh đã thu hút sự chú ý mạnh mẽ trong cộng đồng nghiên cứu, trở thành một trong những nhánh quan trọng nhất của trí tuệ nhân tạo (Artificial Intelligence - AI) và học sâu (Deep Learning - DL). Nó không chỉ đơn thuần là việc sử dụng các thuật toán máy học (Machine learning - ML) để tự động nhận biết và phân loại các đối tượng, vật thể hay bối cảnh trong hình ảnh, mà còn là một hành trình sâu sắc để khám phá và phân tích nội dung tiềm ẩn trong hình ảnh và video.
- Sự hấp dẫn của nhận dạng hình ảnh nằm ở khả năng ứng dụng đa dạng trong nhiều lĩnh vực khác nhau. Từ việc nhận dạng ký tự như chữ ký và chữ viết tay, đến nhận diện khuôn mặt và dấu vân tay, thậm chí là phân loại các loại lá cây, lĩnh vực này không ngừng mở rộng và phát triển. Các phương pháp và thuật toán ngày càng tinh vi được đưa ra nhằm đáp ứng nhu cầu ngày càng cao từ thực tiễn, làm cho nhận dạng hình ảnh trở thành một công cụ hữu ích và thiết yếu.
- Trong nghiên cứu đa dạng sinh học, việc nhận diện lá cây không chỉ đơn thuần là xác định các loài quý hiếm mà còn giúp phát hiện các loài mới, phân loại và lập bản đồ phân bố sinh trưởng của chúng. Phương pháp truyền thống thường yêu cầu sự tham gia của các nhà nghiên cứu và chuyên gia, tốn kém thời gian và nguồn lực. Tuy nhiên, với sự phát triển vượt bậc của trí tuệ nhân tạo, các công cụ phần mềm nhận diện lá cây tự động đã ra đời, mang đến một bước tiến mới.
- Nhờ vào các phương pháp và thuật toán tiên tiến, nhận dạng lá cây giờ đây có thể thực hiện một cách tự động và khách quan hơn bao giờ hết, không chỉ tiết kiệm thời gian mà còn giảm bớt gánh nặng cho các chuyên gia và nhà nghiên cứu, đặc biệt trong bối cảnh thiếu hụt nhân lực trong ngành sinh vật học.
- Trong báo cáo dưới đây, nhóm sẽ trình bày hai thuật toán K-Nearest Neighbors (KNN) và Artificial Neural Network (ANN) cùng với đặc trưng Hu’s Moments và đặc trưng là Histogram of Oriented Gradients (HOG) được áp dụng để phân loại lá cây từ bộ dữ liệu sưu tầm. Bộ dữ liệu này bao gồm 5 loại lá cây khác nhau, và chúng tôi sẽ thảo luận về cách trích xuất các đặc trưng của lá cây và lựa chọn thuật toán phân loại nhằm đạt được độ chính xác cao nhất. Cuối cùng sẽ dùng phương pháp K-fold Cross Validation dùng để đánh giá độ chính xác của 2 mô hình phân loại và so sánh mức độ hiệu quả của chúng.

1.2 Giới thiệu phương pháp sử dụng

1.2.1 Đặc trưng Hu’s Moment

Hu’s Moments là một tập hợp của 7 đặc trưng không thay đổi (invariant features) được phát triển bởi Hu Ming Kei vào năm 1962. Chúng được sử dụng rộng rãi trong việc nhận dạng hình dạng (shape recognition) và phân tích hình ảnh (image analysis). Hu’s Moments có đặc

tính bất biến dưới các phép biến đổi affine như dịch chuyển (translation), xoay (rotation), và thay đổi tỷ lệ (scale). Điều này khiến Hu's Moments trở thành công cụ hữu ích trong xử lý ảnh, đặc biệt khi các đối tượng có thể xuất hiện với các hình dạng khác nhau, nhưng vẫn cần được nhận diện một cách chính xác.



Hình 1. *Sơ đồ quá trình trích xuất đặc trưng Hu's Moments*

Để tính được Hu's Moment ảnh cần được chuyển sang dạng ảnh nhị phân với giá trị 0 cho nền và giá trị 1 cho đối tượng. Quá trình tính Hu's Moment được mô tả như hình 1.

Quá trình trích xuất đặc trưng Hu's moment

Bước 1: Tiền xử lý dữ liệu

Dữ liệu ảnh RGB ban đầu được thực hiện chuyển về ảnh nhị phân (Binary) với giá 0 cho nền và giá trị 1 cho đối tượng (vật thể màu trắng và nền màu đen)

Bước 2: Tính Centroid

$$\overline{x} = \frac{\sum_x \sum_y x \,.\, s(x,y)}{\sum_x \sum_y s(x,y)}, \quad \overline{y} = \frac{\sum_x \sum_y y \,.\, s(x,y)}{\sum_x \sum_y s(x,y)} \qquad (1)$$

Bước 3: Tính Central Moment

$$m_{pq} = \sum_x \sum_y (x-\overline{x})^p (y-\overline{y})^q s(x,y); \;\; p,q = 0,1,2,3 \qquad (2)$$

Bước 4: Tính Central Normalized Moments

$$M_{pq} = \frac{m_{pq}}{m_{00}^{\frac{p+q}{2}+1}} \qquad (3)$$

Bước 5: Tính vector HU's Moments

$S1 = M_{20} + M_{02}$ (4)

$S2 = (M_{20} - M_{02})(M_{20} - M_{02}) + 4M_{11}M_{11}$ (5)

$S3 = (M_{30} - 3M_{12})(M_{30} - 3M_{12}) + (M_{30} - 3M_{21})(M_{30} - 3M_{21})$ (6)

$S4 = (M_{30} + 3M_{12})^2 + (M_{03} + 3M_{21})^2$ (7)

$S5 = (M_{30} - 3M_{12})^2(M_{30} + M_{12})[(M_{30} + 3M_{12})^2 - 3(M_{03} + 3M_{21})^2]$ (8)
$+ (3M_{21} + M_{03})(M_{03} + M_{12})[3(M_{30} + 3M_{12})^2 - (M_{03} + M_{21})^2]$

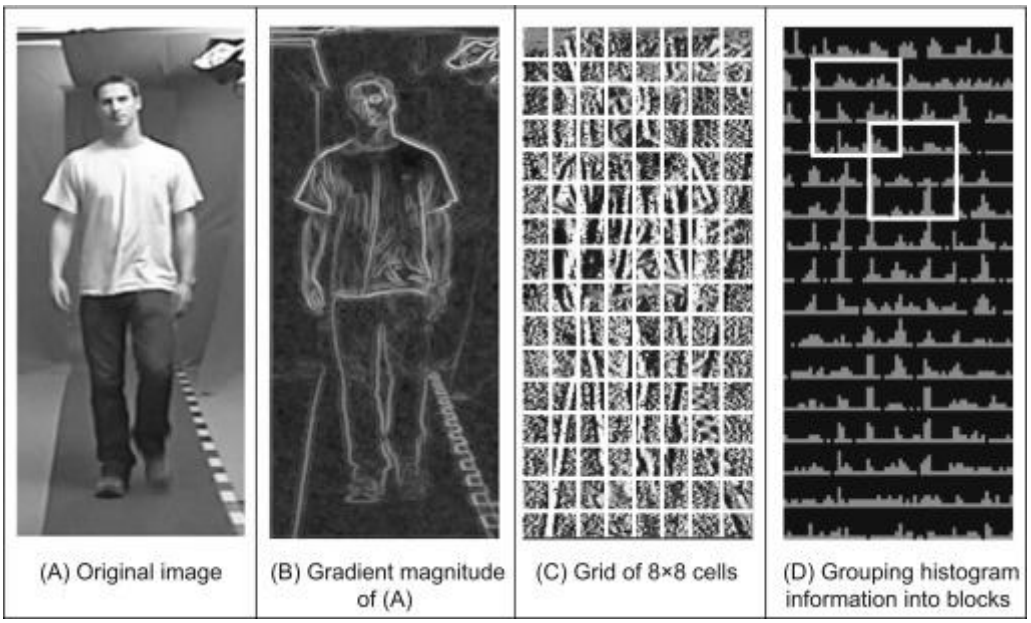
$$S6 = (M_{20} + M_{02}) [(M_{30} + M_{12})^2 - (M_{03} + M_{21})^2] + 4M_{11}(M_{30} + M_{12}) (M_{03} + M_{21}) \quad (9)$$

$$S7 = (3M_{21} - M_{03}) (M_{30} + M_{12}) [(M_{30} + M_{12})^2 - 3(M_{21} + M_{03})^2 - (M_{30} - 3M_{12}) (M_{21} + M_{02}) [3(M_{30} + M_{12})^2 - (M_{21} + M_{03})^2]] \quad (10)$$

Bước 6: Xuất ra vector đặc trưng của ảnh: mỗi vector đặc trưng gồm 7 chiều (7D)

1.2.2 Đặc trưng HOG

HOG là viết tắt của Histogram of Oriented Gradient, là một phương pháp mô tả đặc trưng được sử dụng chủ yếu để trích xuất thông tin về hình dạng và sự xuất hiện của đối tượng trong ảnh. Thuật toán này sẽ tạo ra các bộ mô tả đặc trưng nhằm mục đích phát hiện vật thể. Từ một bức ảnh, ta sẽ lấy ra 2 ma trận quan trọng giúp lưu thông tin ảnh đó là độ lớn gradient (gradient magnitute) và phương của gradient (gradient orientation). Bằng cách kết hợp 2 thông tin này vào một biểu đồ phân phối histogram, trong đó độ lớn gradient được đếm theo các nhóm bins của phương gradient. Cuối cùng ta sẽ thu được vector đặc trưng HOG đại diện cho histogram.



Hình 2. *Trích đặc trưng HOG*

(a) Ảnh gốc đã chuyển sang ảnh xám,
(b) Độ lớn gradient của ảnh gốc,
(c) Ảnh của gradient được chia thành các ô nhỏ 8x8 cells,
(d) Nhóm thông tin histogram vào các khối.

Quá trình trích xuất đặc trưng HOG

Bước 1: Tiền xử lý dữ liệu

Chuyển ảnh RGB sang ảnh xám (Gray), Thay đổi kích thước (Resize) ảnh xám để giảm kích thước dữ liệu giúp đơn giản hóa quá trình xử lý và tập trung vào cấu trúc hình dạng của vật thể mà không bị ảnh hưởng bởi màu sắc.

Bước 2: Tính gradients (hướng x và y)

Tính gradient của ảnh được thực hiện bằng cách sử dụng các mặt nạ lọc. Mặt nạ lọc thông dụng là mặt nạ Sobel 3x3 tính gradient theo hai hướng (x và y)

Bước 3: Tính độ lớn và phương của gradient

Độ lớn của gradient: $|G| = \sqrt{f_x^2 + f_y^2}$

Phương của gradient: $\theta = \tan^{-1}\frac{f_y}{f_x}$

- Lặp lại tính toán độ lớn và phương của gradient cho tất cả pixels, phương sẽ giúp chia các gradient vào các hướng nhất định trong histogram.

Bước 4: Tạo histogram của gradients trong các ô (cell)

- Chia hình ảnh thành các ô (cell)
- Tính histogram của mỗi ô (cell): Điều này giúp biểu diễn hướng chính của gradient trong từng ô

Bước 5: Chuẩn hóa gradients trong các ô (cell)

- Nhóm các ô thành các khối (blocks):
- Mỗi khối, ta có vector V = [v_1, v_2, …, v_{36}]
- Tính k: k = $\sqrt{{v^2}_1 + {v^2}_2 + \cdots + {v^2}_{36}}$
- Chuẩn hóa vector = $\left(\frac{a_1}{k}; \frac{a_2}{k}; \ldots; \frac{a_{36}}{k}\right)$
- Chuẩn hóa: Áp dụng chuẩn hóa L2-Hys cho các histogram trong từng khối, làm cho đặc trưng không bị ảnh hưởng bởi sự thay đổi về độ sáng và độ tương phản giữa các vùng trong ảnh.

Bước 6: Trích xuất đặc trưng cho toàn bộ ảnh

Kết hợp tất cả các khối đã chuẩn hóa: Ghép các vector đặc trưng từ các khối đã chuẩn hóa để tạo thành một vector đặc trưng HOG tổng thể cho toàn bộ ảnh.

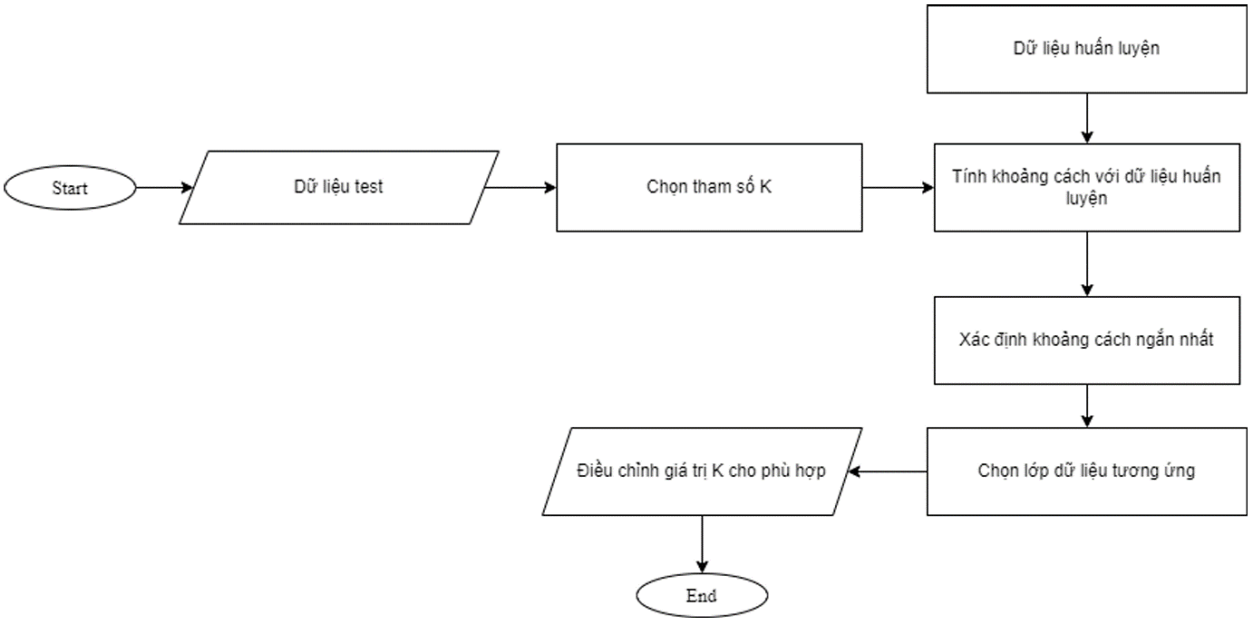
1.2.3 Thuật toán KNN

Thuật toán KNN học có giám sát hoạt động theo nguyên tắc mọi điểm dữ liệu nằm gần nhau đều thuộc cùng một lớp. Giả định cơ bản ở đây là những thứ ở gần nhau thì giống nhau, từ đó công việc của chúng ta là sẽ tìm k điểm gần với dữ liệu cần kiểm tra nhất. Nó được sử dụng rộng rãi cho cả bài toán phân loại (classification) và hồi quy (regression). Đặc điểm nổi bật của KNN là việc không cần một mô hình cụ thể cho việc huấn luyện, mà dựa trên lưu trữ và so sánh trực tiếp với dữ liệu mẫu (dữ liệu huấn luyện).

Việc tìm khoảng cách giữa 2 điểm cũng có nhiều công thức có thể sử dụng, tùy trường hợp mà chúng ta lựa chọn cho phù hợp. Đây là 2 cách cơ bản thường được dùng để tính khoảng cách 2 điểm dữ liệu x, y có k thuộc tính.

Công thức Euclidean: $d = \sqrt{\sum_{i=1}^{k} |x_i - y_i|^2}$ (11)

Công thức Manhattan: $d = \sum_{i=1}^{k} |x_i - y_i|$



Hình 3. *Sơ đồ thuật toán KNN*

Bước 1. Chuẩn bị dữ liệu để huấn luyện và đánh giá

Bước 2. Chọn số lượng (K) các điểm dữ liệu gần nhất mà thuật toán sẽ xem xét khi dự đoán nhãn cho một điểm dữ liệu mới.

Bước 3. Tính toán khoảng cách giữa điểm dữ liệu mới và các điểm dữ liệu trong tập huấn luyện bằng cách sử dụng phép đo khoảng cách *"d"* ở trên [11]

Trong đó: x_i là dữ liệu huấn luyện

y_i là dữ liệu test

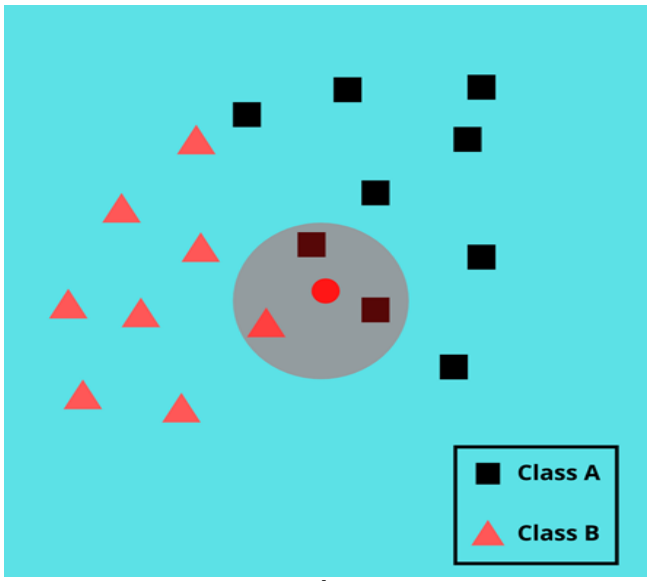
Bước 4. Chọn ra K điểm dữ liệu gần nhất với điểm dữ liệu mới. Dựa trên nhãn của các điểm dữ liệu, dự đoán nhãn cho điểm dữ liệu mới bằng cách sử dụng trọng số của các nhãn. Chọn nhãn xuất hiện nhiều nhất trong các điểm dữ liệu gần nhất dựa trên khoảng cách (nhãn chiếm ưu thế).

Bước 5. Đánh giá hiệu suất của mô hình bằng cách sử dụng các phương pháp đánh giá. Tùy chỉnh tham số K để cải thiện hiệu suất của mô hình.

Hãy xem xét một ví dụ:

Trong Hình 4 đã cho, chúng ta có hai lớp dữ liệu. Lớp A đại diện cho hình vuông và lớp B đại diện cho hình tam giác. Bài toán gán một điểm dữ liệu đầu vào mới cho một trong hai lớp với việc sử dụng thuật toán KNN.

Xem xét K = 3 có nghĩa là thuật toán sẽ xem xét ba hàng xóm gần điểm dữ liệu nhất, trong ô tròn màu xám có thể thấy một hình tam giác và hai hình vuông có thể được coi là những hàng xóm gần nhất. Vì vậy, điểm dữ liệu mới dựa trên K = 3, sẽ được gán cho Lớp A.



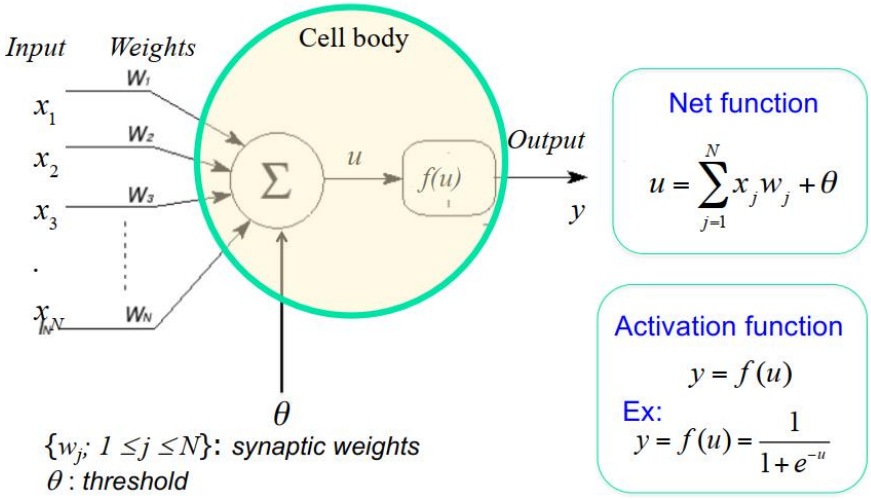
Hình 4. *Ví dụ về thuật toán KNN*

1.2.4 Thuật toán ANN

Mạng neuron nhân tạo (ANN) là một trong những mô hình phổ biến trong học máy, đặc biệt hiệu quả cho các bài toán phân loại phức tạp như nhận diện đặc trưng hình ảnh.

Mô hình ANN có khả năng học và nhận diện đặc trưng phức tạp từ dữ liệu hình ảnh, giúp tối ưu hóa hiệu quả nhận diện các loại lá cây khác nhau. Với đặc trưng Hu’s moment và HOG, thuật toán ANN có thể học các thông tin quan trọng về hình dạng và đường biên của lá, từ đó cải thiện khả năng phân loại chính xác giữa các loài lá cây.

Mô hình ANN áp dụng phân loại lá được thiết kế với một lớp ẩn, nhằm tối ưu hoá hiệu năng và tốc độ xử lý. Mô hình sử dụng hàm kích hoạt là hàm sigmoid và hàm kết nối là hàm tuyến tính phù hợp để học các đặc trưng phức tạp của dữ liệu.



Hình 5. *Mô hình neuron cơ bản*

1.2.5 Phương pháp đánh giá K-Fold Cross-Validation

K-Fold Cross-Validation là một phương pháp phổ biến trong đánh giá hiệu suất của các mô hình AI, đặc biệt là với các tập dữ liệu nhỏ. Phương pháp này chia tập dữ liệu thành K phần và thực hiện quá trình huấn luyện và kiểm tra K lần để đảm bảo rằng mô hình không bị overfitting hoặc underfitting.

Dưới đây là các bước cơ bản của phương pháp:

Bước 1. Chia tập dữ liệu

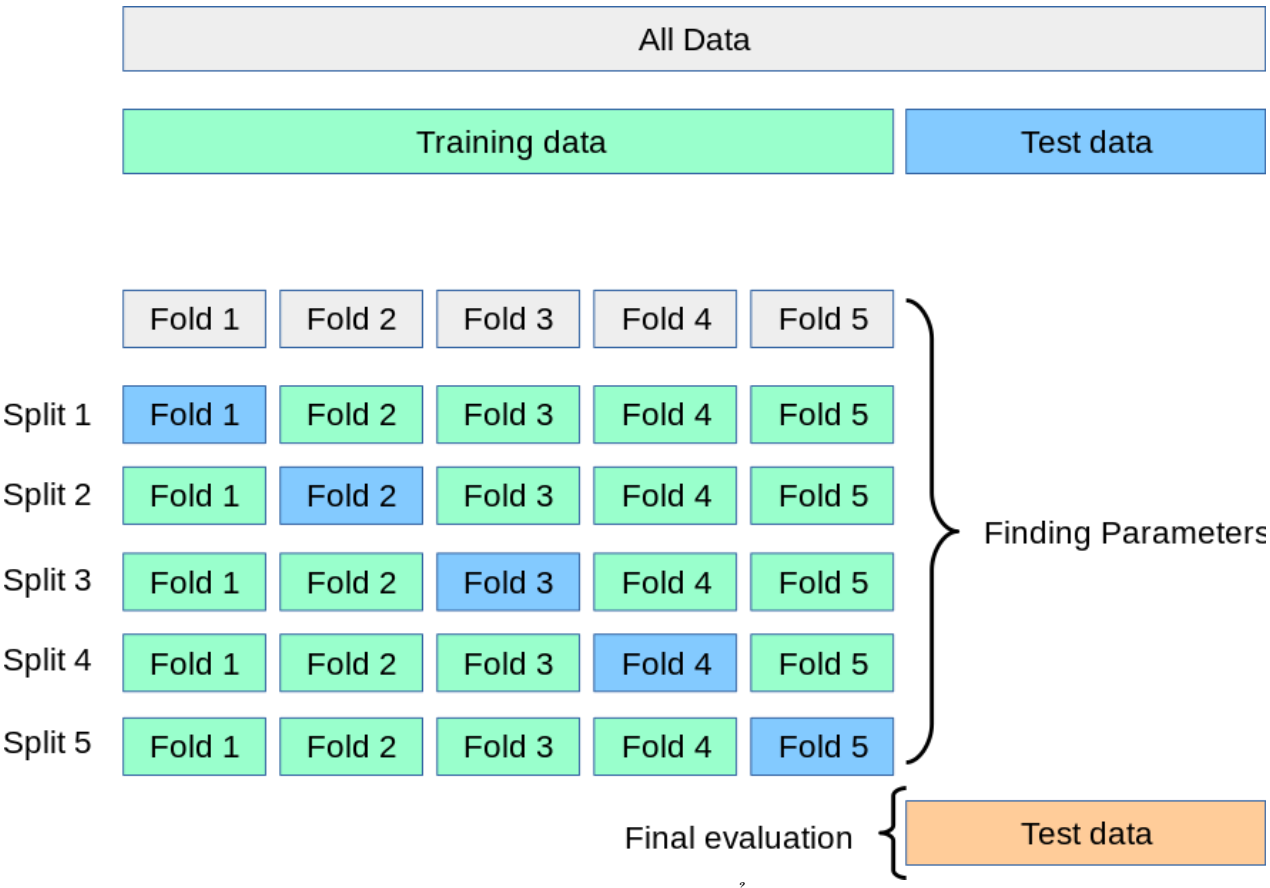
Tập dữ liệu được chia thành K phần có kích thước bằng nhau (thường là từ 5 đến 10 folds). Mỗi fold sẽ lần lượt đóng vai trò là tập kiểm tra (test set), trong khi các fold còn lại sẽ là tập huấn luyện (train set).

Bước 2. Huấn luyện và kiểm tra

Mô hình sẽ được huấn luyện K lần. Ở mỗi lần, mô hình được huấn luyện trên K−1 folds và kiểm tra trên fold còn lại.

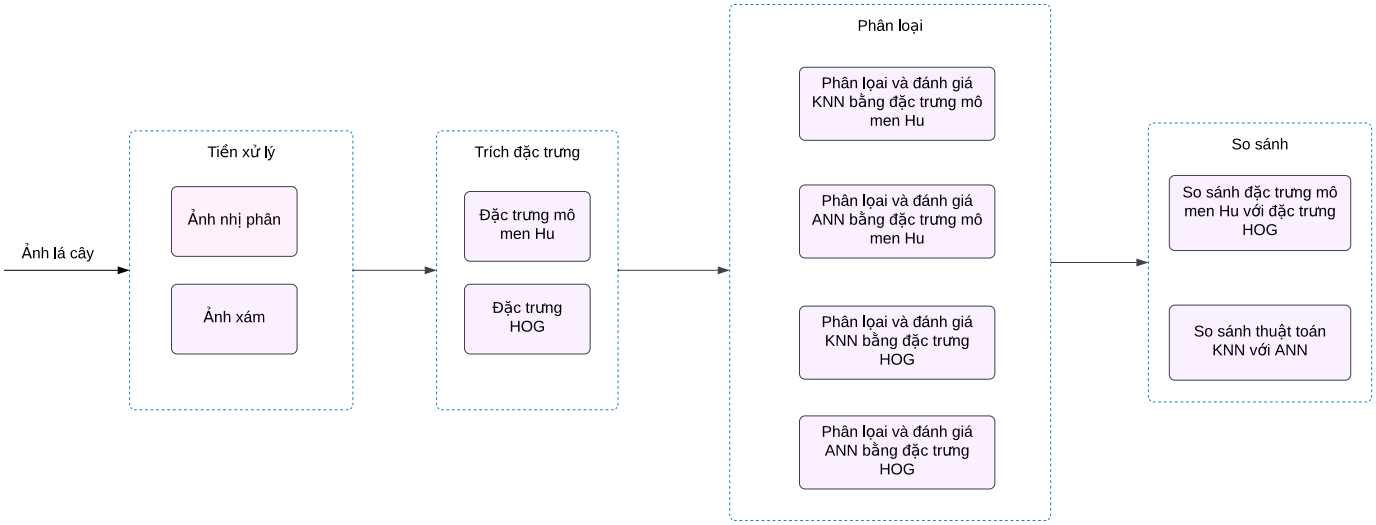
Bước 3. Tính toán kết quả trung bình

Sau khi hoàn tất K lần huấn luyện và kiểm tra, ta lấy trung bình các chỉ số hiệu suất như độ chính xác của mô hình, Recall, Precision, F1-score. Điều này giúp đánh giá hiệu suất và độ chính của mô hình.



Hình 6. *Phương pháp K-fold để đánh giá mô hình*

2. Phương pháp thực hiện

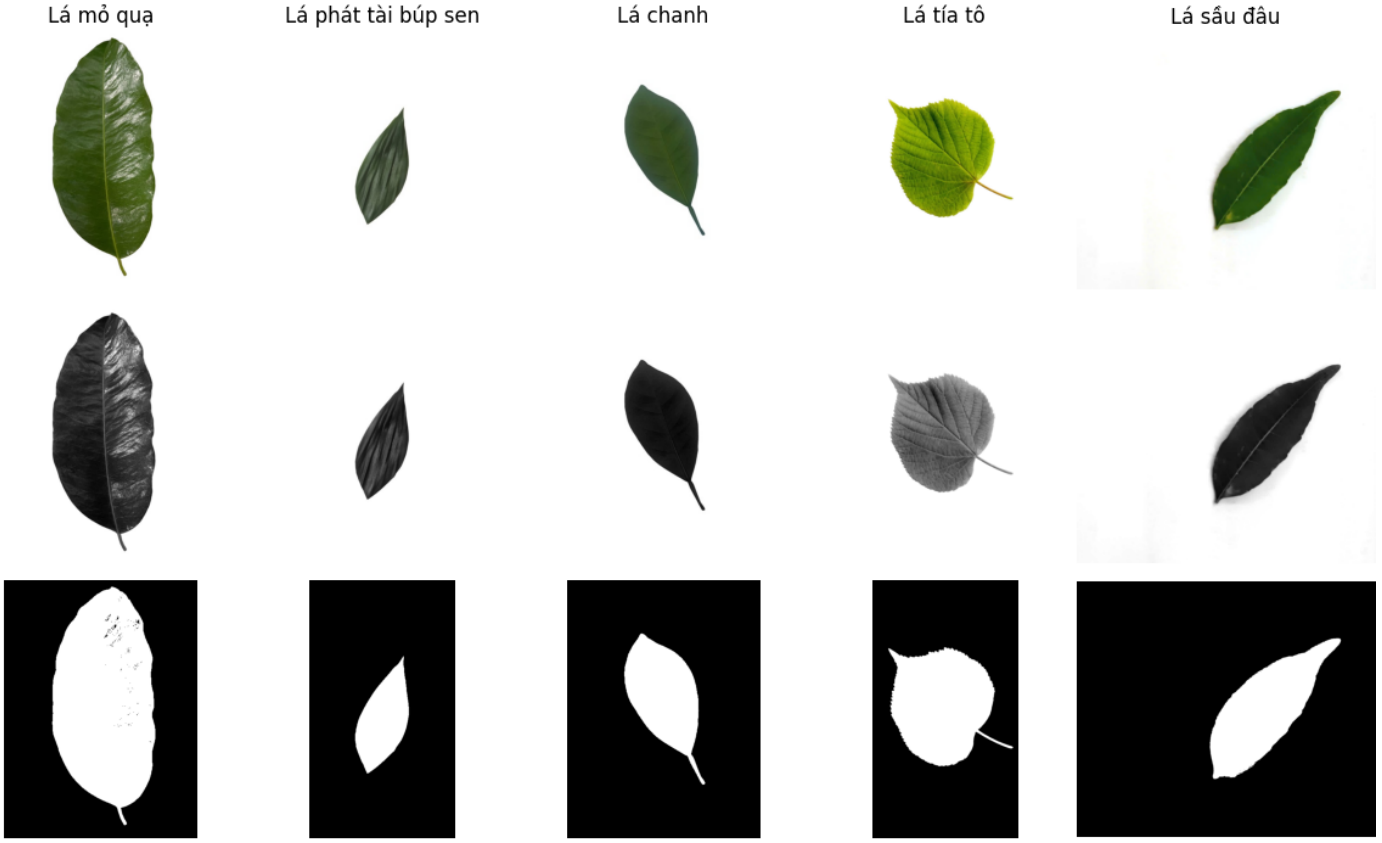


Hình 7. *Sơ đồ khối tổng quát hệ thống phân loại và đánh giá*

2.1 Chuẩn bị dữ liệu lá cây

Nghiên cứu này thực hiện với bộ dữ liệu lá cây đã được chuẩn bị cho quá trình phân loại và đánh giá. Bộ dữ liệu gồm 5 loài lá cây sưu tầm khác nhau, trong đó 3 loài do nhóm tự chụp và 2 loài tham khảo trong cơ sở dữ liệu của lớp học phần *(20N38)*.

Mỗi loài gồm 50 bức ảnh lá cây, tổng cộng có tất cả 250 mẫu, cùng với đó là tập dữ liệu lá cây đã được chuyển sang ảnh xám và ảnh nhị phân để thuận lợi cho việc trích xuất đặc trưng Hu's moment và HOG.



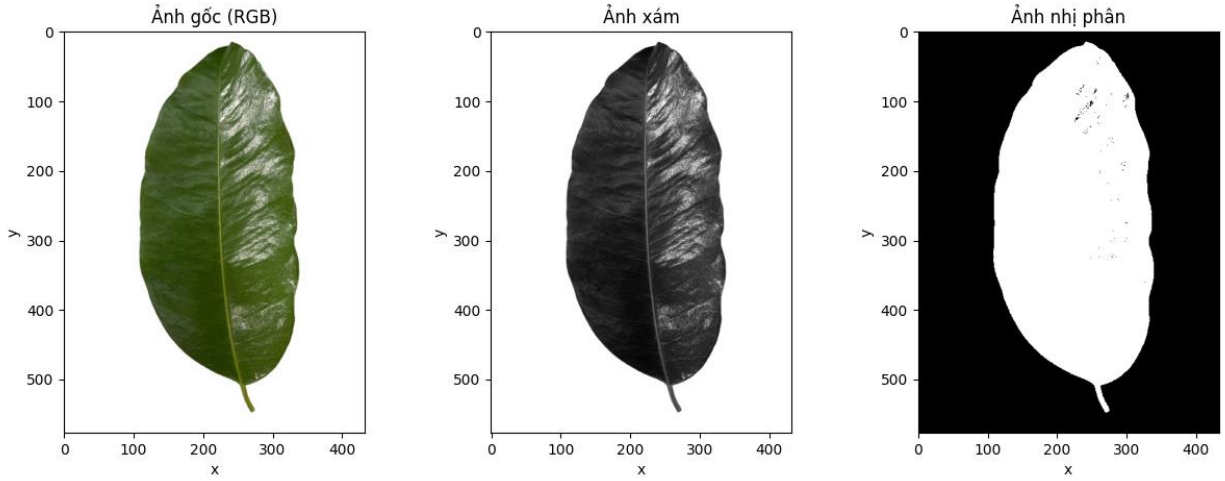
Hình 8. *Bộ dữ liệu về 5 loài lá cây được sử dụng*

2.2 Trích đặc trưng của lá cây

2.2.1 Sử dụng Hu's moment để trích xuất các đặc trưng của lá cây

- Đối tượng lá cây sẽ được chuyển sang dạng nhị phân như **Hình 8** trước khi được mô tả bằng đặc trưng hình dạng dùng Hu's moment.
- Quá trình tính toán các giá trị đặc trưng Hu's gồm các bước đã được nêu ở **mục 1.2.1**
- Sau khi tính toán các giá trị Hu's moment, thu được một bảng giá trị gồm 7 đặc trưng. Nhận thấy giá trị của các moment này rất nhỏ và gần như bằng 0. Do đó nhóm đã thực hiện chuẩn hoá giá trị của 7 đặc trưng này bằng *"Phương pháp logarit hóa"*

Cụ thể phương pháp này lấy logarit cơ số 10 của giá trị Hu's moment ban đầu.



Hình 9. *Ảnh đối tượng lá mỏ quạ*

- **Bảng 2** là giá trị của 7 đặc trưng Hu's moment trước và sau khi chuẩn hoá của loại *"Lá mỏ quạ"* trong lớp thứ nhất của bộ dữ liệu lá cây đã được đề cập trước đó.

Bảng 2. *Tập đặc trưng Hu's moment của một mẫu dữ liệu*

Hu's moment	Trước khi chuẩn hóa	Sau khi chuẩn hóa
S1	0.000792690154319307	3.100892797
S2	2.33206980665358E-07	6.632249563
S3	6.69458006610927E-12	11.17416281
S4	4.44825752374373E-13	12.35161308
S5	-2.44209006427737E-25	-24.61244033
S6	-9.93135842281759E-17	-16.00306125
S7	-7.27738388515019E-25	-24.13760995

-Toàn bộ tập dữ liệu 250 lá cây cần cho mô hình được lưu vào *file hu.csv*

2.2.2 Sử dụng HOG để trích xuất các đặc trưng của lá cây

Quá trình thực hiện đặc trưng HOG gồm 6 bước đã được nêu ở **mục 1.2.2**

Đối tượng lá cây sẽ được chuyển từ ảnh RGB sang ảnh xám như **Hình 8** trước rút trích đặc trưng HOG.

- Chuẩn hóa kích thước hình ảnh: Đảm bảo tất cả các ảnh lá có cùng kích thước 128x128 pixels để tính toán HOG đồng nhất cho 250 lá.
- Gán nhãn: 5 nhãn gồm lá mỏ quạ, lá phát tài búp sen, lá chanh, lá tía tô, lá sầu đâu.
- Kích thước 1 cell: 8x8 pixels = 64 block/image
- Số cell 1 block: 1x1 cell
- 8 feature/cell
- Số đặc trưng trích xuất được từ một ảnh: 64 x 1 x 8 = 512 features/image

- Các giá trị đặc trưng được lưu lại dưới dạng bảng Excel
-Toàn bộ tập dữ liệu 250 lá cây cần cho mô hình được lưu vào *file hog.csv*

2.3. Thực hiện phân loại lá cây dùng phương pháp KNN và đánh giá hệ thống

- Thay đổi giá trị K để thấy ảnh hưởng của K đến hiệu quả của hệ thống.
- Đánh giá hệ thống dùng phương pháp 5-fold cross validation

Bước 1: Chuẩn bị bộ dữ liệu đã được lưu vào file csv trong quá trình trích xuất đặc trưng ở **mục 2.1** để phục vụ cho quá trình huấn luyện và đánh giá.

- Đối với đặc trưng HOG, là một ma trận có kích thước là 250x512, tức là có 250 vector đặc trưng của 250 hình ảnh, với mỗi vector sẽ có 512 chiều đặc trưng.
- Tương tự với đặc trưng Hu's moment, thu được ma trận có kích thước 250x7, cụ thể là có 250 vector đặc trưng của 250 hình, với mỗi vector có 7 chiều đặc trưng Hu's moment.
- Đầu ra 5 class được gán nhãn lần lượt là: (1) Lá mỏ quạ, (2) Lá phát tài búp sen, (3) Lá chanh, (4) Lá tía tô, (5) Lá sầu đâu.

Bước 2: Xây dựng mô hình phân loại KNN, dựa vào việc so sánh khoảng cách gần nhất của đối tượng cần phân loại so với dữ liệu huấn luyện.

- Từ đó chọn ra K điểm gần nhất và lấy số lượng lớp nào chiếm ưu thế nhát sẽ gán cho đối tượng cần phân loại thuộc lớp đó.
- Áp dụng công thức Euclidean hoặc Mahattan để tính khoảng cách giữa hai đối tượng.

Bước 3: Đánh giá mô hình bằng phương pháp K-Fold Cross-Validation với k=5

- Tập test được chia ngẫu nhiên về các lớp, sao cho mỗi fold có tổng 50 mẫu test
- Từ 250 hình ảnh trong tập dữ liệu sẽ được chia làm 5 phần. Một phần sẽ được làm tập test và các phần còn lại sẽ là tập train và lặp lại như vậy qua lần lượt từng fold.
- Độ chính xác của mô hình là trung bình độ chính xác của từng fold

2.4 Thực hiện phân loại lá cây dùng phương pháp ANN và đánh giá hệ thống

- Hàm kết nối (net function) là hàm tuyến tính, hàm kích hoạt (activation function) là hàm sigmoid, 1 lớp ẩn, tốc độ học là 0.05.
- Thay đổi số neuron lớp ẩn lần lượt là 10 và 15, để thấy ảnh hưởng của số neuron lớp ẩn đến hiệu quả của hệ thống.
- Đánh giá hệ thống dùng phương pháp 5-fold cross validation.

Bước 1: Chuẩn bị bộ dữ liệu gồm 5 loại lá cây rồi từ bộ dữ liệu đó chuyển thành một bộ dữ liệu gồm các ảnh xám và một bộ dữ liệu gồm ảnh nhị phân **(Hình 8)**.

Bước 2: Đối với dữ liệu ảnh xám, trích xuất đặc trưng HOG như đã trình bày ở **mục 1.2.2**. Đối với bộ dữ liệu ảnh nhị phân, trích xuất 7 đặc trưng Hu với công thức tính toán đã đề cập ở **mục 1.2.1**

Bước 3: Chuẩn bị bộ dữ liệu đã được lưu vào file csv trong quá trình trích xuất đặc trưng ở bước 2 để phục vụ cho quá trình huấn luyện và đánh giá. Đối với đặc trưng HOG, là một ma trận có kích thước là 250x512, tức là có 250 vector đặc trưng của 250 hình ảnh, với mỗi

vector sẽ có 512 chiều đặc trưng. Tương tự với đặc trưng Hu, ta thu được vector đặc trưng có kích thước 250x7, cụ thể là có 250 vector đặc trưng, với 7 chiều Hu's moment cho mỗi vector đặc trưng. Và đầu ra 5 class được mã hóa nhãn thành dạng one-hot-encoding với Lá mỏ quạ (10000), Lá phát tài búp sen (01000), Lá chanh (00100), Lá tía tô (00010), Lá sầu đâu (00001).

Bước 4: Xây dựng mô hình phân loại ANN bằng cách thiết lập và huấn luyện mô hình Neural Network cụ thể là thiết lập hàm kích hoạt (activation function) là hàm sigmoid, 1 lớp ẩn, tốc độ học là 0.05 khởi tạo trọng số và bias ngẫu nhiên cho các lớp. Xây dựng hàm truyền tiến (Forward Propagation) để thực hiện tính toán đầu ra của lớp ẩn và lớp đầu ra của mạng neural. Xây dựng hàm truyền ngược (Backward Propagation) để tính toán các gradient để cập nhật trọng số, giúp giảm sai số của mô hình qua các vòng lặp.

Bước 5: Đánh giá mô hình bằng phương pháp K-Fold Cross-Validation với k=5

- Từ 250 hình ảnh trong tập dữ liệu sẽ được chia làm 5 phần. Một phần sẽ được làm tập test và các phần còn lại sẽ là tập train và lặp lại như vậy qua lần lượt từng fold
- Độ chính xác của mô hình là trung bình độ chính xác của từng fold

2.5 So sánh và nhận xét các đặc trưng, các thuật toán ML đã dùng

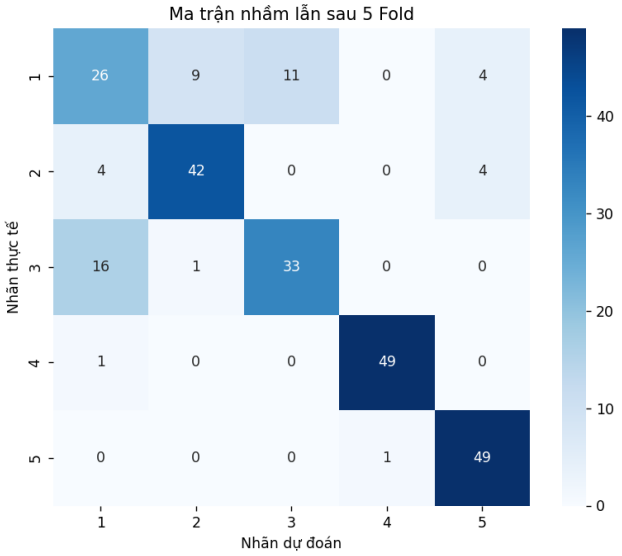
Dữ liệu 50 mẫu trong tập test được chọn ngẫu nhiên từ 5 lớp

2.5.1 Dùng KNN để phân loại dựa vào đặc trưng Hu và HOG

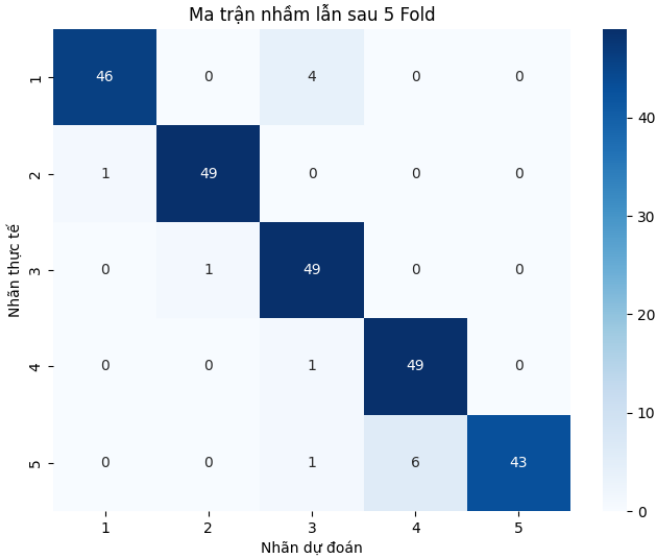
Trường hợp 1: giá trị KNN với K = 3

Bảng 3. *Độ chính xác trung bình của KNN với đặc trưng Hu và HOG*

K		KNN với Hu	KNN với HOG
3	*Accuracy*	0.7960	0.9440
	Precision	0.8019	0.9499
	Recall	0.7960	0.9440
	F1-score	0.7931	0.9425



Hình 10. *KNN với đặc trưng Hu*



Hình 11. *KNN với đặc trưng HOG*

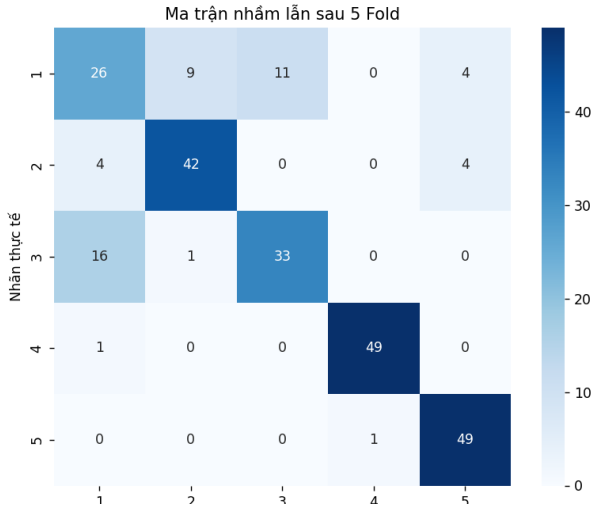
Bảng 4. *Độ chính xác của phương pháp KNN với đặc trưng Hu và HOG*

K		KNN với Hu	KNN với HOG
3	*Fold 1*	0.7800	0.9400
	Fold 2	0.7600	0.9400
	Fold 3	0.7600	0.9400
	Fold 4	0.8400	1.0000
	Fold 5	0.8400	0.9000
	Mean_Accuracy	0.7960	0.9440

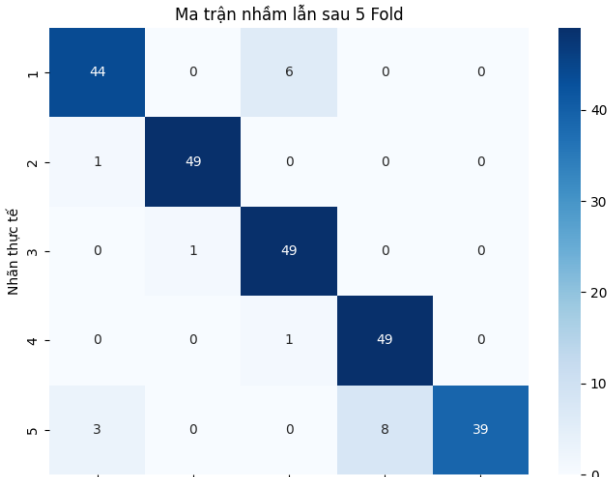
Trường hợp 2: giá trị KNN với K = 5

Bảng 5. *Độ chính xác trung bình của KNN với đặc trưng Hu và HOG*

K		KNN với Hu	KNN với HOG
5	*Accuracy*	0.7960	0.9200
	Precision	0.8019	0.9300
	Recall	0.7960	0.9200
	F1-score	0.7931	0.9190



Hình 12*. KNN với đặc trưng Hu*



Hình 13. *KNN với đặc trưng HOG*

Bảng 6. *Độ chính xác của KNN với đặc trưng Hu và HOG*

K		KNN với Hu	KNN với HOG
5	*Fold 1*	0.7800	0.9200
	Fold 2	0.7600	0.9000
	Fold 3	0.7600	0.9000
	Fold 4	0.8400	0.9800
	Fold 5	0.8400	0.9000
	Mean_Accuracy	0.7960	0.9200

Từ **Bảng 5** cho thấy các thông số đánh giá của KNN với đặc trưng HOG có độ chính xác cao hơn KNN với đặc trưng Hu, chi tiết có thể đánh giá qua ma trận nhầm lẫn (Confusion matrix),

Hình 12 có độ chính xác gần tuyệt đối với các giá trị dự đoán trên hai nhãn 4 và 5, trong đó:

Lớp 1: 26 mẫu đúng, 9 mẫu nhận nhầm sang lớp 2, 11 mẫu nhận nhầm sang lớp 3 và 4 mẫu nhận nhầm sang lớp 4

Lớp 2: 42 mẫu đúng, 4 mẫu nhận nhầm sang lớp 1 và 4 mẫu nhận nhầm sang lớp 5

Lớp 3: 33 mẫu đúng, 16 mẫu nhận nhầm sang lớp 1, 1 mẫu nhận nhầm sang lớp 2

Lớp 4: 49 mẫu đúng, 1 mẫu nhận nhầm sang lớp 1

Lớp 5: 49 mẫu đúng, 1 mẫu nhận nhầm sang lớp 4

Hình 13 có độ chính xác cao với các giá trị dự đoán trên cả ba nhãn 2, 3 và 4 trong đó:

Lớp 1: 44 mẫu đúng, 6 mẫu nhận nhầm sang lớp 3

Lớp 2: 49 mẫu đúng, 1 mẫu nhận nhầm lớp 1

Lớp 3: 49 mẫu đúng, 1 mẫu nhận nhầm lớp 2

Lớp 4: 49 mẫu đúng, 1 mẫu nhận nhầm lớp 3

Lớp 5: 39 mẫu đúng, 3 mẫu nhận nhầm sang lớp 1, 8 mẫu nhận nhầm lớp 4

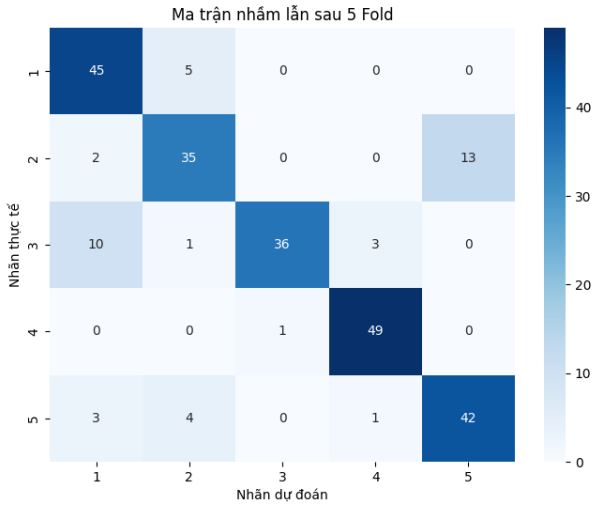
Từ **Bảng 6** giá trị accuracy cho mỗi fold, nhìn chung thì độ chính xác mà hệ thống đạt được lá tương đối cao. Tuy nhiên, qua quá trình đánh giá, ta có thể thấy rõ đặc trưng HOG vượt trội hơn trong việc phân loại các lớp trong tập dữ liệu này khi sử dụng KNN (đạt độ chính xác là 0.9200 của đặc trưng HOG so với 0.7960 của đặc trưng Hu).

2.5.2 Dùng ANN để phân loại dựa vào đặc trưng Hu và HOG

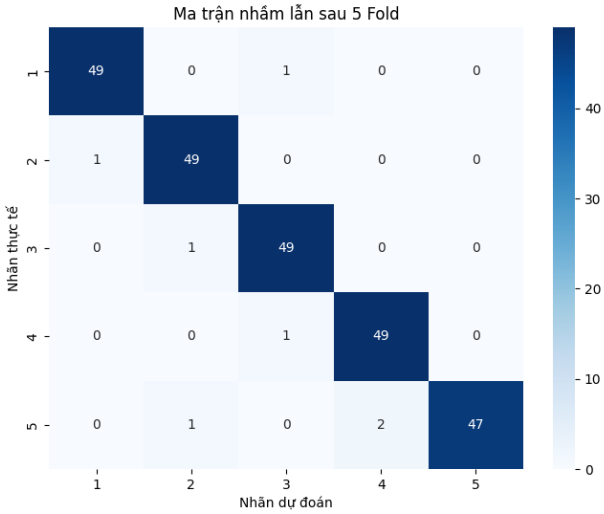
Trường hợp 1: tham số lớp ẩn là 10 neuron

Bảng 7. *Độ chính xác trung bình của ANN với đặc trưng Hu và HOG*

	ANN với Hu	ANN với HOG
Accuracy	0.8320	0.9720
Precision	0.8495	0.9734
Recall	0.8320	0.9720
F1-score	0.8291	0.9719



Hình 14. *ANN với đặc trưng Hu*



Hình 15. *ANN với đặc trưng HOG*

Bảng 8. *Độ chính xác của ANN với đặc trưng Hu và HOG*

	ANN với Hu	**ANN với HOG**
Fold 1	0.8000	0.9800
Fold 2	0.8400	1.0000
Fold 3	0.7800	0.9400
Fold 4	0.8800	1.0000
Fold 5	0.8600	0.9600
Mean_Accuracy	0.8320	0.9720

Từ **Bảng 7** cho thấy các thông số đánh giá của ANN với đặc trưng HOG có độ chính xác cao hơn ANN với đặc trưng Hu, chi tiết có thể đánh giá qua ma trận nhầm lẫn

Hình 14 có độ chính xác khá cao với các giá trị dự đoán trên hai nhãn 1 và 4, trong đó:

Lớp 1: 45 mẫu đúng, 5 mẫu nhận nhầm sang lớp 2

Lớp 2: 35 mẫu đúng, 2 mẫu nhận nhầm lớp 1 và 13 mẫu nhận nhầm lớp 5

Lớp 3: 36 mẫu đúng, 10 mẫu nhận nhầm lớp 1, 1 mẫu nhận nhầm lớp 2 và 3 mẫu nhận nhầm lớp 3

Lớp 4: 49 mẫu đúng, 1 mẫu nhận nhầm lớp 3

Lớp 5: 42 mẫu đúng, 3 mẫu nhận nhầm lớp 1, 4 mẫu nhận nhầm lớp 2 và 1 mẫu nhầm lớp 4

Hình 15 có độ chính xác cao với các giá trị dự đoán trên cả năm nhãn, trong đó:

Lớp 1: 49 mẫu đúng, 1 mẫu nhận nhầm sang lớp 3

Lớp 2: 49 mẫu đúng, 1 mẫu nhận nhầm lớp 1

Lớp 3: 49 mẫu đúng, 1 mẫu nhận nhầm lớp 2

Lớp 4: 49 mẫu đúng, 1 mẫu nhận nhầm lớp 3

Lớp 5: 49 mẫu đúng, 1 mẫu nhận nhầm lớp 2, 2 mẫu nhận nhầm lớp 4

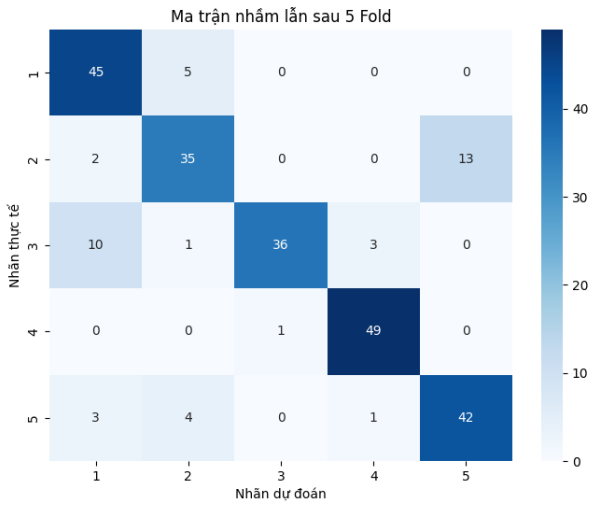
Từ **Bảng 8** giá trị accuracy cho mỗi fold, nhìn chung thì độ chính xác mà hệ thống đạt được lá tương đối cao. Tuy nhiên, qua quá trình đánh giá ta có thể thấy rõ đặc trưng HOG có vẻ

vượt trội hơn trong việc phân loại các lớp trong tập dữ liệu này khi sử dụng ANN (đạt độ chính xác là 0.9720 của đặc trưng HOG so với 0.8320 của đặc trưng Hu)

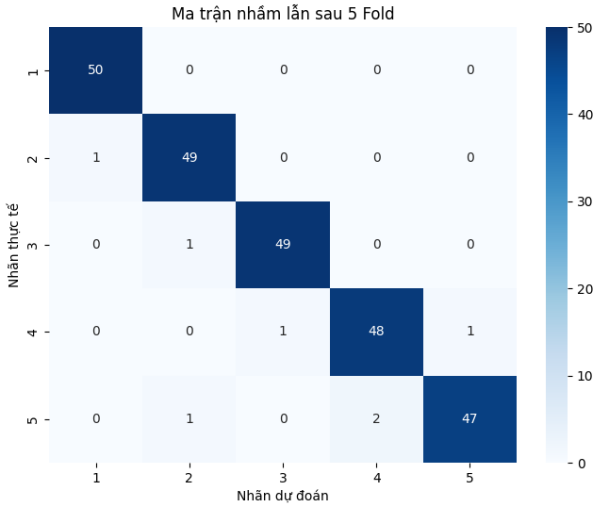
Trường hợp 2: tham số lớp ẩn là 15 neuron

Bảng 9. *Độ chính xác trung bình của ANN với đặc trưng Hu và HOG*

	ANN với Hu	**ANN với HOG**
Accuracy	0.8440	0.9720
Precision	0.8680	0.9733
Recall	0.8440	0.9720
F1-score	0.8435	0.9729



Hình 16. *ANN với đặc trưng Hu*



Hình 17. *ANN với đặc trưng HOG*

Bảng 10. *Độ chính xác của ANN với đặc trưng Hu và HOG*

	ANN với Hu	**ANN với HOG**
Fold 1	0.8400	0.9600
Fold 2	0.8400	1.0000
Fold 3	0.7800	0.9400
Fold 4	0.8800	0.9800
Fold 5	0.8800	0.9800
Mean_Accuracy	0.8440	0.9720

Tương tự như trường hợp 1, các thông số đánh giá của ANN với đặc trưng HOG có độ chính xác cao hơn ANN với đặc trưng Hu

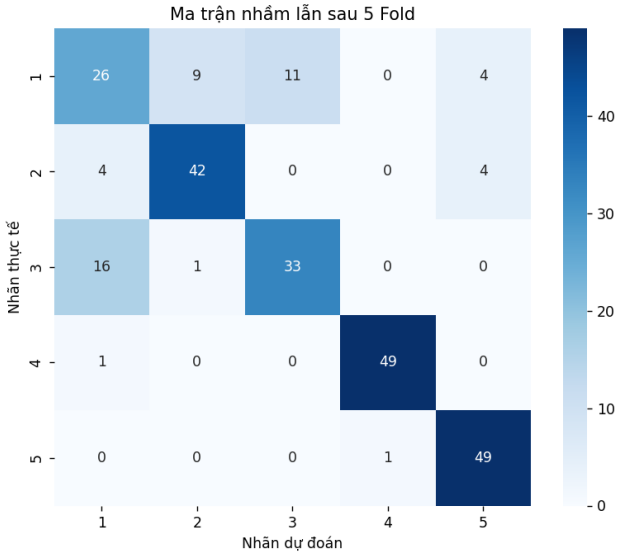
Kết quả ở trường hợp 2 không thay đổi đáng kể khi thay đổi số neuron lớp ẩn từ 10 lên 15 neuron. Trong đó Accuracy của ANN với đặc trưng Hu tăng đáng kể so với trường hợp 1 (từ 83.2% tăng lên 84.4%), còn ANN với đặc trưng HOG thì giữ nguyên.

2.5.3 Dùng KNN và ANN để phân loại dựa vào đặc trưng Hu
Chọn tham số với K = 3 và số neuron lớp ẩn = 15

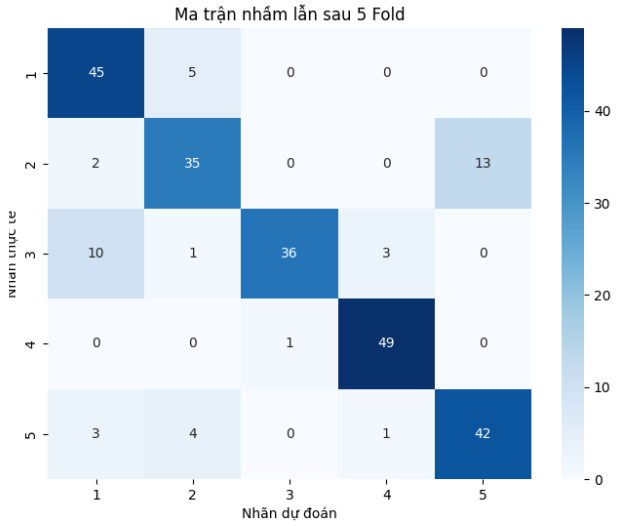
Bảng 11. *Độ chính xác trung bình của KNN và ANN với Hu*

	KNN với Hu	**ANN với Hu**
Accuracy	0.7960	0.8440
Precision	0.8019	0.8680
Recall	0.7960	0.8440
F1-score	0.7931	0.8435

Bảng 11 đánh giá độ chính xác đối với đặc trưng Hu moment thì phương pháp phân loại ANN sẽ cho ra các thông số đánh giá cao hơn so với KNN (0.7960 so với 0.8440)



Hình 18*. KNN với đặc trưng Hu*



Hình 19*. ANN với đặc trưng Hu*

Bảng 12. *Độ chính xác của KNN và ANN với đặc trưng Hu*

	KNN với Hu	**ANN với Hu**
Fold 1	0.7800	0.8400
Fold 2	0.7600	0.8400
Fold 3	0.7600	0.7800
Fold 4	0.8400	0.8800
Fold 5	0.8400	0.8800
Mean_Accuracy	0.7960	0.8440

Nhìn chung với đặc trưng Hu thì cả 2 phương pháp đều cho ra các thông số đánh giá thấp nhất, số lượng các nhãn bị nhận nhầm sang các lớp khác tương đối cao **Hình 18**.
Hình 19 cho thấy độ chính xác trên nhãn 1 và 4 cao hơn các nhãn còn lại, trong đó nội dung đã được nhận xét trong ***"trường hợp 1 của mục 2.5.2".*** Tuy nhiên, model ANN cho thấy sự nhầm lẫn ít hơn so với KNN.

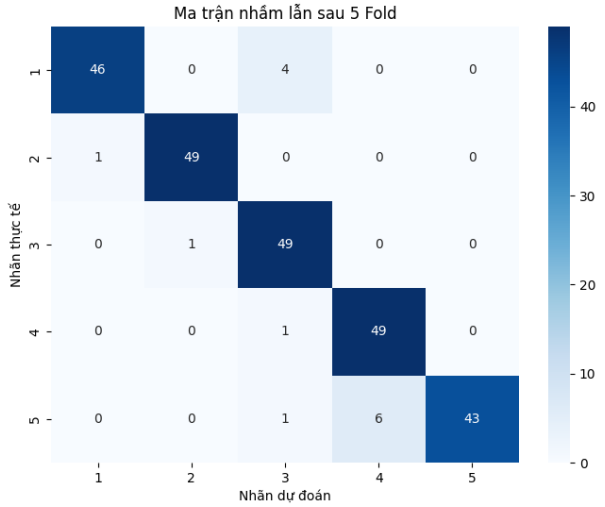
2.5.4 Dùng KNN và ANN để phân loại dựa vào đặc trưng HOG

Chọn Tham số với K = 3 và số neuron lớp ẩn = 15

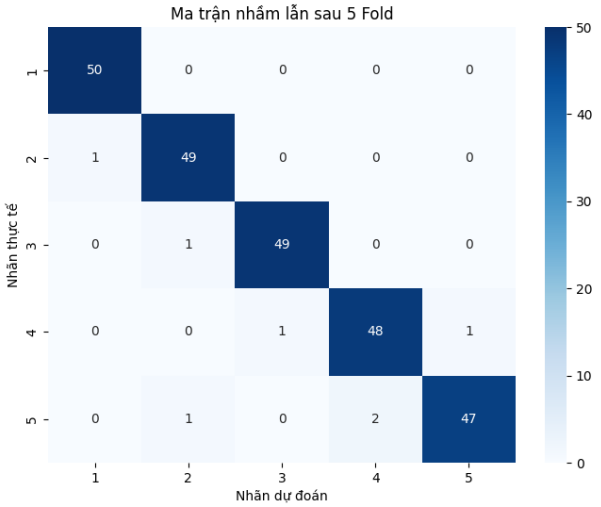
Bảng 13. *Độ chính xác trung bình KNN và ANN với đặc trưng HOG*

	KNN với HOG	**ANN với HOG**
Accuracy	0.9440	0.9720
Precision	0.9499	0.9733
Recall	0.9440	0.9720
F1-score	0.9425	0.9729

Bảng 13 đánh giá độ chính xác đối với đặc trưng HOG thì phương pháp phân loại ANN sẽ cho ra các thông số đánh giá cao hơn nhiều so với phương pháp KNN (0.972 và 0.944), trong các trường hợp 50 mẫu trong tập test được chọn ngẫu nhiên thì độ chính xác của ANN đạt gần 100%



Hình 20. *KNN với đặc trưng HOG*



Hình 10. *ANN với đặc trưng HOG*

Bảng 14*. Độ chính xác của ANN với đặc trưng HOG*

	KNN với HOG	**ANN với HOG**
Fold 1	0.9400	0.9600
Fold 2	0.9400	1.0000
Fold 3	0.9400	0.9400
Fold 4	1.0000	0.9800
Fold 5	0.9000	0.9800
Mean_Accuracy	0.9440	0.9720

Từ **Bảng 14**, nhìn chung khi dựa vào đặc trưng HOG thì cả 2 mô hình KNN và ANN đều có sự nhầm lẫn trong việc phân loại nhãn của các lớp nhưng không nhiều, trong đó:

Hình 20 cho thấy độ chính xác cao với các giá trị dự đoán trên nhãn 2, 3, 4

Lớp 1: 46 mẫu đúng, 4 mẫu nhận nhầm lớp 3

Lớp 2: 49 mẫu đúng, 1 mẫu nhận nhầm lớp 1

Lớp 3: 49 mẫu đúng, 1 mẫu nhận nhầm lớp 2

Lớp 4: 49 mẫu đúng, 1 mẫu nhận nhầm lớp 3

Lớp 5: 43 mẫu đúng, 1 mẫu nhận nhầm lớp 3 và 6 mẫu nhận nhầm sang lớp 4

Hình 21 cho thấy độ chính xác gần như tuyệt đối với các giá trị dự đoán trên cả 5 nhãn

Lớp 1: 50 mẫu đúng

Lớp 2: 49 mẫu đúng, 1 mẫu nhận nhầm lớp 1

Lớp 3: 49 mẫu đúng, 1 mẫu nhận nhầm lớp 2

Lớp 4: 48 mẫu đúng, 1 mẫu nhận nhầm lớp 3, 1 mẫu nhận nhầm lớp 5

Lớp 5: 47 mẫu đúng, 1 mẫu nhận nhầm lớp 2, 2 mẫu nhận nhầm lớp 4

Tuy nhiên, model ANN cho thấy sự nhầm lẫn ít hơn so với KNN.

3. Kết luận và hướng phát triển

3.1 Kết luận

- Trong nghiên cứu này, nhóm đã thực hiện việc xây dựng hệ thống phân loại tự động cho 5 loại lá cây khác nhau, sử dụng hai phương pháp học máy: K-Nearest Neighbors (KNN) và Artificial Neural Network (ANN). Để đặc trưng hóa cấu trúc và hình dạng của các mẫu lá, nhóm đã áp dụng hai phương pháp trích xuất đặc trưng: Hu Moments và Histogram of Oriented Gradients (HOG).
- Kết quả thực nghiệm cho thấy hiệu suất đáng kể của cả hai phương pháp phân loại:
    + Đối với phương pháp KNN: đặc trưng Hu Moments đạt độ chính xác 79.6% và HOG đạt độ chính xác 94.40%
    + Đối với phương pháp ANN: đặc trưng Hu Moments đạt độ chính xác 84.00% và HOG đạt độ chính xác cao nhất với 97.20%
- Sự thành công của hệ thống phân loại này có được nhờ quá trình xử lý dữ liệu kỹ lưỡng và việc trích xuất các đặc trưng phù hợp cho từng đối tượng. Đáng chú ý, đặc trưng HOG đã thể hiện hiệu quả vượt trội so với Hu Moments trong việc mô tả chi tiết đặc trưng hình dạng của đối tượng.
- Tuy nhiên, Hu Moments vẫn thể hiện được những ưu điểm riêng, đặc biệt trong việc mô tả đối tượng trong ảnh nhị phân với các tính chất bất biến quan trọng như:
    + Khả năng thích ứng với sự co giãn của ảnh
    + Độc lập với vị trí của đối tượng trong ảnh
    + Không phụ thuộc vào góc xoay của ảnh
- Về phần thuật toán khi xét KNN và ANN với cùng một loại đặc trưng kết quả cho thấy ANN cho hiệu xuất phân loại cao hơn thuật KNN, dẫn chứng được thể hiện rõ qua các chỉ số Accuracy, Precision, Recall, và F1-Score.

3.2 Hướng phát triển

- Dựa trên kết quả nghiên cứu hiện tại, nhóm đề xuất một số hướng phát triển quan trọng cho hệ thống phân loại lá cây.

- Đầu tiên, cần tối ưu hóa các mô hình KNN và ANN hiện có thông qua việc điều chỉnh các tham số như: thử nghiệm các giá trị k khác nhau cho KNN và điều chỉnh cấu trúc mạng neural (số lớp ẩn, số node) cho ANN.
- Bên cạnh đó, việc tăng cường chất lượng đặc trưng bằng cách kết hợp Hu Moments và HOG với các đặc trưng khác như SIFT, SURF, cùng với việc nghiên cứu phương pháp giảm chiều dữ liệu hiệu quả như PCA và LDA sẽ góp phần cải thiện hiệu suất của hệ thống.
- Ứng dụng Deep Learning, đặc biệt là các mô hình CNN đã chứng minh hiệu quả vượt trội trong các bài toán xử lý ảnh. Việc thiết kế kiến trúc CNN phù hợp với đặc điểm của lá cây và áp dụng các kỹ thuật augmentation để tăng độ đa dạng dữ liệu là rất cần thiết. Đồng thời, việc tích hợp các mô hình pre-trained thông qua transfer learning với các mô hình như ResNet, VGG, hay EfficientNet, kết hợp với fine-tuning phù hợp với bài toán cụ thể sẽ giúp cải thiện đáng kể hiệu suất phân loại.
- Những đề xuất này không chỉ nhằm nâng cao độ chính xác của hệ thống mà còn mở rộng phạm vi ứng dụng trong thực tiễn, đặc biệt trong lĩnh vực nông nghiệp và nghiên cứu sinh học. Với sự phát triển không ngừng của công nghệ AI, việc tích hợp các phương pháp mới sẽ tạo ra những bước tiến quan trọng trong lĩnh vực phân loại và nhận dạng thực vật.

TÀI LIỆU THAM KHẢO

[1]. Zhihu Huang and Jinsong Leng, “Analysis of Hu’s Moment Invariants on Image Scaling and Rotation”, 2nd Int. Conf. on Computer Engineering and Technology (ICCET), 2010, pp. 476-480

[2]. Lubis, Zulkarnain, Poltak Sihombing, and Herman Mawengkang.“Optimization of K Value at the K-NN algorithm in clustering using the expectation maxi mization algorithm." IOP Conference Series: Materials Science and Engineer ing. Vol. 725. No. 1. IOP Publishing, 2020.

[3]. H. Lê, U. Thục, V. Tuấn, and P. Tuấn, “HU’S MOMENTS FOR VISUAL PATTERN RECOGNITION.” Accessed: Nov. 06, 2024. [Online]. Available: https://elib.vku.udn.vn/bitstream/123456789/98/1/20181207124544.pdf

[4]. Phạm Đình Khánh, *Khanh’s blog*, 2019. https://phamdinhkhanh.github.io/2019/11/22/HOG.html (accessed Nov. 06, 2024)

[5]. “C34 | HOG Feature Vector Calculation | Computer Vision | Object Detection | EvODN,” www.youtube.com. https://www.youtube.com/watch?v=28xk5i1_7Zc

[6]. Geeks for geeks, “Artificial Neural Networks and its Applications,” GeeksforGeeks, Jun. 24, 2020. https://www.geeksforgeeks.org/artificial-neural-networks-and-its-applications/

[7] Dataset: https://bit.ly/du-lieu-la-cay_20N38

PHỤ LỤC

1. Bảng biểu

2. Hình ảnh

3. Code

3.1 Trích đặc trưng Hu's moment, lưu file hu.csv

```
import numpy as np
from PIL import Image
import os
import csv

# Function to calculate central moments
def central_moment(data, p, q, x_centroid, y_centroid):
    y, x = np.ogrid[:data.shape[0], :data.shape[1]]
    return np.sum(((x - x_centroid) ** p) * ((y - y_centroid) ** q) * data)

# Function to calculate central normalized moments
def central_normalized_moment(moments, p, q, m00):
    return moments[f'm_{p}{q}'] / (m00 ** (((p + q) / 2) + 1))

# Function to calculate Hu moments
def calculate_hu_moments(M):
    S = {}
    S[1] = M['M_20'] + M['M_02']
    S[2] = (M['M_20'] - M['M_02']) ** 2 + 4 * (M['M_11'] ** 2)
    S[3] = (M['M_30'] - 3 * M['M_12']) ** 2 + (3 * M['M_21'] - M['M_03']) ** 2
    S[4] = (M['M_30'] + M['M_12']) ** 2 + (M['M_03'] + M['M_21']) ** 2
    S[5] = (M['M_30'] - 3 * M['M_12']) * (M['M_30'] + M['M_12']) * (
            (M['M_30'] + M['M_12']) ** 2 - 3 * (M['M_03'] + M['M_21']) ** 2) + \
           (3 * M['M_21'] - M['M_03']) * (M['M_03'] + M['M_21']) * (
                   3 * (M['M_30'] + M['M_12']) ** 2 - (M['M_03'] + M['M_21']) ** 2)
    S[6] = (M['M_20'] - M['M_02']) * ((M['M_30'] + M['M_12']) ** 2 - (M['M_03'] + M['M_21']) ** 2) + \
           4 * M['M_11'] * (M['M_30'] + M['M_12']) * (M['M_03'] + M['M_21'])
    S[7] = (3 * M['M_21'] - M['M_03']) * (M['M_30'] + M['M_12']) * (
            (M['M_30'] + M['M_12']) ** 2 - 3 * (M['M_03'] + M['M_21']) ** 2) - \
           (M['M_30'] - 3 * M['M_12']) * (M['M_03'] + M['M_21']) * (
                   3 * (M['M_30'] + M['M_12']) ** 2 - (M['M_03'] + M['M_21']) ** 2)
    return S
def process_image(image_path):
    im = Image.open(image_path).convert("L")
    threshold = 128
    im_bin = im.point(lambda p: 255 if p > threshold else 0)
    data = np.array(im_bin)

    m00 = np.sum(data == 255).astype(np.float64) * 255

    # Tính toán tọa độ trọng tâm
    height, width = data.shape
    x_centroid = np.sum(np.arange(width) * np.sum(data, axis=0)) / m00
    y_centroid = np.sum(np.arange(height) * np.sum(data, axis=1)) / m00

    # Tính toán các mômen trung tâm đến bậc 3
    moments = {}
    for p in range(4):
        for q in range(4):
            if p + q <= 3:
                moments[f'm_{p}{q}'] = central_moment(data, p, q, x_centroid, y_centroid)

    # Tính toán các mômen chuẩn hóa trung tâm
    normalized_moments = {}
    for p in range(4):
        for q in range(4):
            if 0 < p + q <= 3:
                normalized_moments[f'M_{p}{q}'] = central_normalized_moment(moments, p, q, m00)

    # Tính toán các mômen Hu
    hu_moments = calculate_hu_moments(normalized_moments)

    return hu_moments
```

```python
def process_images_in_folder(input_folder, output_csv):
    classes = ['1', '2', '3', '4', '5']  # Define your classes
    class_counter = [0] * len(classes)  # To keep track of the number of images per class
    train_limit = 40  # Number of training images per class
    test_limit = 10   # Number of test images per class

    with open(output_csv, mode='w', newline='') as file:
        writer = csv.writer(file)
        header = ['Image'] + [f"S{i}" for i in range(1, 8)] + ['Labels']  # Add Set to header
        writer.writerow(header)  # Write header to CSV

        # Iterate through the folder and process each image
        for filename in os.listdir(input_folder):
            if filename.endswith(".png") or filename.endswith(".jpg"):  # Filter for image files
                # Determine the class based on the count
                for idx, count in enumerate(class_counter):
                    if count < (train_limit + test_limit):  # If less than total images in the class
                        image_path = os.path.join(input_folder, filename)  # Full path to image
                        hu_moments = process_image(image_path)  # Calculate Hu moments

                        row = [filename] + [hu_moments[i] for i in range(1, 8)] + [classes[idx]]
                        writer.writerow(row)  # Write row to CSV
                        class_counter[idx] += 1  # Increment the count for that class
                        break  # Exit loop after assigning the class

# Example usage
input_folder = "images/NhiPhan"  # Input folder with images
output_csv = "Output/hu2.csv"  # Output CSV file

process_images_in_folder(input_folder, output_csv)  # Process the folder and save results
```

3.2 Trích đặc trưng HOG, lưu file hog.csv

```
import os
import numpy as np
import pandas as pd
from PIL import Image
from skimage.feature import hog

input_folder = 'images/Goc'
resized_folder = 'resize'
os.makedirs(resized_folder, exist_ok=True)

# Đường dẫn để lưu tệp CSV đầu ra
output_csv = 'Output/hog.csv'
# Danh sách để lưu các HOG features
hog_features_list = []
# Nhãn cho mỗi nhóm 50 ảnh
labels = ['1', '2', '3', '4', '5']

def resize_image(image, target_size):
    # Lấy kích thước gốc
    original_size = image.size
    target_width, target_height = target_size

    # Tính toán tỷ lệ
    ratio = min(target_width / original_size[0], target_height / original_size[1])
    new_size = (int(original_size[0] * ratio), int(original_size[1] * ratio))

    # Resize ảnh
    resized_image = image.resize(new_size)  # Loại bỏ Image.ANTIALIAS
    # Tạo một ảnh trắng (hoặc đen) với kích thước mong muốn
    new_image = Image.new("L", (target_width, target_height))

    # Tính toán vị trí để đặt ảnh đã resize vào giữa ảnh mới
    paste_x = (target_width - new_size[0]) // 2
    paste_y = (target_height - new_size[1]) // 2

    # Dán ảnh đã resize vào ảnh mới
    new_image.paste(resized_image, (paste_x, paste_y))

    return new_image
```

```python
# Duyệt qua tất cả các tệp trong thư mục
for idx, filename in enumerate(os.listdir(input_folder), start=1):
    if filename.endswith(('.png', '.jpg', '.jpeg', '.bmp', '.tiff')):
        try:
            # Đọc ảnh nhị phân bằng PIL
            img_path = os.path.join(input_folder, filename)
            image = Image.open(img_path).convert('L')  # Chuyển ảnh thành ảnh xám (grayscale)

            # Resize ảnh về kích thước cố định để tính HOG
            resized_image = resize_image(image, (128, 128))  # Resize về 128x128
            resized_image_np = np.array(resized_image)

            # Lưu ảnh đã resize vào thư mục
            resized_image.save(os.path.join(resized_folder, filename))

            # Tính toán các đặc trưng HOG
            hog_features = hog(resized_image_np, orientations=8, pixels_per_cell=(16, 16),
                               cells_per_block=(1, 1), visualize=False)

            # Chuyển thành mảng 1D để lưu trữ
            hog_features = hog_features.flatten()

            # Xác định nhãn dựa trên chỉ số của ảnh
            label = labels[(idx - 1) // 50]  # Chia nhóm mỗi 50 ảnh

            # Lưu kết quả vào danh sách
            hog_features_list.append({
                'HOG': list(hog_features),  # Lưu toàn bộ đặc trưng HOG dưới dạng danh sách
                'Tên Ảnh': filename,  # Thêm tên ảnh vào
                'Labels': label  # Thêm nhãn vào
            })

        except Exception as e:
            print(f"Lỗi xảy ra khi xử lý ảnh {img_path}: {str(e)}")

if hog_features_list:
    df = pd.DataFrame(hog_features_list)

    # Tách từng giá trị của HOG thành các cột riêng biệt
    hog_df = pd.DataFrame(df['HOG'].to_list(), columns=[f'HOG_{i + 1}' for i in range(len(df['HOG'][0]))])

    # Kết hợp các cột tên ảnh và HOG trước
    temp_df = pd.concat([df[['Tên Ảnh']], hog_df], axis=1)

    # Thêm cột nhãn ở cuối
    final_df = pd.concat([temp_df, df[['Labels']]], axis=1)

    # Lưu DataFrame vào tệp CSV
    final_df.to_csv(output_csv, index=False, encoding='utf-8-sig')

    print(f"Kết quả đã được lưu vào tệp CSV: {output_csv}")
else:
    print("Không có đặc trưng HOG nào được tính toán.")
```

3.3 Mô hình phân loại lá cây KNN với đặc trưng Hu's moment và đánh giá

```
import numpy as np
import pandas as pd
from sklearn.model_selection import KFold
from sklearn.metrics import precision_score, recall_score, f1_score, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Hàm tính khoảng cách Euclidean
def euclidean_distance(x1, x2):
    return np.sqrt(np.sum((x1 - x2) ** 2))


# Hàm KNN tự định nghĩa
def knn(X_train, y_train, X_test, k=5):
    y_pred = []
    for test_point in X_test:
        distances = [euclidean_distance(test_point, train_point) for train_point in X_train]
        k_indices = np.argsort(distances)[:k]
        k_nearest_labels = [y_train[i] for i in k_indices]
        most_common = np.bincount(k_nearest_labels).argmax()
        y_pred.append(most_common)
    return np.array(y_pred)


# Bước 1: Đọc file CSV chứa các đặc trưng Hu và nhãn
input_csv = 'hu_.csv'  # Đường dẫn tới tệp CSV
df = pd.read_csv(input_csv)

# Bước 2: Chuẩn bị dữ liệu
features = df.iloc[:, 1:8]  # Các cột đặc trưng Hu
labels = df.iloc[:, 8]  # Cột nhãn Hu

# Chuyển các đặc trưng và nhãn thành mảng numpy
X = features.values
y = labels.values

# Bước 3: Chia dữ liệu theo phương pháp K-Fold với k=5
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Đánh giá mô hình KNN với k=5
y_true_k5 = []
y_pred_k5 = []

# Lưu các chỉ số cho từng fold
accuracies = []
precisions = []
recalls = []
f1_scores = []
confusion_matrices = []
```

```
print("Đánh giá mô hình KNN với k=5:")
for fold, (train_index, test_index) in enumerate(kf.split(X)):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    # Huấn luyện và đánh giá mô hình KNN
    y_pred = knn(X_train, y_train, X_test, k=3)     #thay đổi k tại đây

    y_true_k5.extend(y_test)
    y_pred_k5.extend(y_pred)

    # Tính toán các chỉ số đánh giá
    accuracy = np.mean(y_pred == y_test)
    precision = precision_score(y_test, y_pred, average='weighted', zero_division=1)
    recall = recall_score(y_test, y_pred, average='weighted', zero_division=1)
    f1 = f1_score(y_test, y_pred, average='weighted', zero_division=1)
    conf_matrix = confusion_matrix(y_test, y_pred)

    # Thêm các chỉ số vào danh sách
    accuracies.append(accuracy)
    precisions.append(precision)
    recalls.append(recall)
    f1_scores.append(f1)
    confusion_matrices.append(conf_matrix)

    # In ra nhãn thực tế và nhãn dự đoán cho mỗi ảnh trong fold
    print("\nSo sánh chi tiết dự đoán:")
    print("Ground Truth:  ", '   '.join(map(str, y_test)))
    print("Predicted:     ", '   '.join(map(str, y_pred)))
    print("\n" + "-" * 50 + "\n")

    # In ra các chỉ số và ma trận nhầm lẫn
    print(f"Fold {fold + 1}:")
    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1 Score: {f1:.4f}")
    print("Ma trận nhầm lẫn:")
    print(conf_matrix)
# Tính toán kết quả trung bình
mean_accuracy = np.mean(accuracies)
mean_precision = np.mean(precisions)
mean_recall = np.mean(recalls)
mean_f1_score = np.mean(f1_scores)

# Tính toán ma trận nhầm lẫn trung bình
total_confusion_matrix = np.sum(confusion_matrices, axis=0)

# In kết quả trung bình
print("\nKết quả trung bình sau 5 fold:\n")
print("Ma trận nhầm lẫn trung bình:")
print(total_confusion_matrix)
# Vẽ biểu đồ ma trận nhầm lẫn
plt.figure(figsize=(8, 6))
sns.heatmap(total_confusion_matrix, annot=True, fmt='d', cmap='Blues', cbar=True, square=True)

# Cài đặt nhãn cho biểu đồ
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix - Tổng hợp sau 5 fold')
plt.xticks(ticks=np.arange(5) + 0.5, labels=['1', '2', '3', '4', '5'])
plt.yticks(ticks=np.arange(5) + 0.5, labels=['1', '2', '3', '4', '5'])
plt.show()
print(f"Mean Accuracy: {mean_accuracy:.4f}")
print(f"Mean Precision: {mean_precision:.4f}")
print(f"Mean Recall: {mean_recall:.4f}")
print(f"Mean F1 Score: {mean_f1_score:.4f}")
```

3.4 Mô hình phân loại lá cây KNN với đặc trưng HOG và đánh giá
Để t add luôn cho nó chung format

```
import numpy as np
import pandas as pd
from sklearn.model_selection import KFold
from sklearn.metrics import precision_score, recall_score, f1_score, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

# Hàm tính khoảng cách Euclidean
def euclidean_distance(x1, x2):
    return np.sqrt(np.sum((x1 - x2) ** 2))

# Hàm KNN tự định nghĩa
def knn(X_train, y_train, X_test, k=5):
    y_pred = []
    for test_point in X_test:
        distances = [euclidean_distance(test_point, train_point) for train_point in X_train]
        k_indices = np.argsort(distances)[:k]
        k_nearest_labels = [y_train[i] for i in k_indices]
        most_common = np.bincount(k_nearest_labels).argmax()
        y_pred.append(most_common)
    return np.array(y_pred)
```

```
# Bước 1: Đọc file CSV chứa các đặc trưng HOG và nhãn
input_csv = 'hog_.csv'  # Đường dẫn tới tệp CSV
df = pd.read_csv(input_csv)

# Bước 2: Chuẩn bị dữ liệu
features = df.iloc[:, 1:513]    # Các cột đặc trưng HOG
labels = df.iloc[:, 513]         # Cột nhãn HOG

# Chuyển các đặc trưng và nhãn thành mảng numpy
X = features.values
y = labels.values

# Bước 3: Chia dữ liệu theo phương pháp K-Fold với k=5
kf = KFold(n_splits=5, shuffle=True, random_state=42)

# Đánh giá mô hình KNN với k=5
y_true_k5 = []
y_pred_k5 = []

# Lưu các chỉ số cho từng fold
accuracies = []
precisions = []
recalls = []
f1_scores = []
confusion_matrices = []
```

```
print("Đánh giá mô hình KNN với k=5:")
for fold, (train_index, test_index) in enumerate(kf.split(X)):
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y[train_index], y[test_index]

    # Huấn luyện và đánh giá mô hình KNN
    y_pred = knn(X_train, y_train, X_test, k=5)    #thay đổi k tại đây

    y_true_k5.extend(y_test)
    y_pred_k5.extend(y_pred)

    # Tính toán các chỉ số đánh giá
    accuracy = np.mean(y_pred == y_test)
    precision = precision_score(y_test, y_pred, average='weighted', zero_division=1)
    recall = recall_score(y_test, y_pred, average='weighted', zero_division=1)
    f1 = f1_score(y_test, y_pred, average='weighted', zero_division=1)
    conf_matrix = confusion_matrix(y_test, y_pred)

    # Thêm các chỉ số vào danh sách
    accuracies.append(accuracy)
    precisions.append(precision)
    recalls.append(recall)
    f1_scores.append(f1)
    confusion_matrices.append(conf_matrix)

    # In ra nhãn thực tế và nhãn dự đoán cho mỗi ảnh trong fold
    print("\nSo sánh chi tiết dự đoán:")
    print("Ground Truth:  ", '   '.join(map(str, y_test)))
    print("Predicted:     ", '   '.join(map(str, y_pred)))
    print("\n" + "-" * 50 + "\n")

    # In ra các chỉ số và ma trận nhầm lẫn
    print(f"Fold {fold + 1}:")
    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1 Score: {f1:.4f}")
    print("Ma trận nhầm lẫn:")
    print(conf_matrix)
# Tính toán kết quả trung bình
mean_accuracy = np.mean(accuracies)
mean_precision = np.mean(precisions)
mean_recall = np.mean(recalls)
mean_f1_score = np.mean(f1_scores)

# Tính toán ma trận nhầm lẫn trung bình
total_confusion_matrix = np.sum(confusion_matrices, axis=0)

# In kết quả trung bình
print("\nKết quả trung bình sau 5 fold:\n")
print("Ma trận nhầm lẫn trung bình:")
print(total_confusion_matrix)
# Vẽ biểu đồ ma trận nhầm lẫn
plt.figure(figsize=(8, 6))
sns.heatmap(total_confusion_matrix, annot=True, fmt='d', cmap='Blues', cbar=True, square=True)

# Cài đặt nhãn cho biểu đồ
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix - Tổng hợp sau 5 fold')
plt.xticks(ticks=np.arange(5) + 0.5, labels=['1', '2', '3', '4', '5'])
plt.yticks(ticks=np.arange(5) + 0.5, labels=['1', '2', '3', '4', '5'])
plt.show()
print(f"Mean Accuracy: {mean_accuracy:.4f}")
print(f"Mean Precision: {mean_precision:.4f}")
print(f"Mean Recall: {mean_recall:.4f}")
print(f"Mean F1 Score: {mean_f1_score:.4f}")
```

3.5 Mô hình phân loại lá cây ANN với đặc trưng Hu's moment và đánh giá

```
import numpy as np
import pandas as pd
from sklearn.model_selection import StratifiedKFold, KFold
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import confusion_matrix, accuracy_score, precision_recall_fscore_support
import seaborn as sns
import matplotlib.pyplot as plt
from collections import Counter

def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def sigmoid_derivative(x):
    return x * (1 - x)

def initialize_parameters(input_size, hidden_size, output_size):
    # Khởi tạo trọng số ngẫu nhiên
    W1 = np.random.randn(input_size, hidden_size) * 0.01
    b1 = np.zeros((1, hidden_size))
    W2 = np.random.randn(hidden_size, output_size) * 0.01
    b2 = np.zeros((1, output_size))
    return W1, b1, W2, b2

def forward_propagation(X, W1, b1, W2, b2):
    # Net function là tuyến tính, và activation là sigmoid cho hidden layer
    z1 = np.dot(X, W1) + b1  # net function tuyến tính
    a1 = sigmoid(z1)  # hàm kích hoạt sigmoid

    # Net function và activation cho output layer
    z2 = np.dot(a1, W2) + b2  # net function tuyến tính
    a2 = sigmoid(z2)  # hàm kích hoạt sigmoid

    return z1, a1, z2, a2


def backward_propagation(X, y, a1, a2, W2):
    m = X.shape[0]
    # Tính đạo hàm
    dz2 = a2 - y
    dW2 = np.dot(a1.T, dz2) / m
    db2 = np.sum(dz2, axis=0, keepdims=True) / m

    da1 = np.dot(dz2, W2.T)
    dz1 = da1 * sigmoid_derivative(a1)
    dW1 = np.dot(X.T, dz1) / m
    db1 = np.sum(dz1, axis=0, keepdims=True) / m

    return dW1, db1, dW2, db2
```

```python
def train_model(X, y, hidden_size, learning_rate, epochs):
    input_size = X.shape[1]
    output_size = y.shape[1]
    # Khởi tạo tham số
    W1, b1, W2, b2 = initialize_parameters(input_size, hidden_size, output_size)
    # Training
    for epoch in range(epochs):
        # Forward propagation
        z1, a1, z2, a2 = forward_propagation(X, W1, b1, W2, b2)
        # Backward propagation
        dW1, db1, dW2, db2 = backward_propagation(X, y, a1, a2, W2)
        # Cập nhật tham số
        W1 -= learning_rate * dW1
        b1 -= learning_rate * db1
        W2 -= learning_rate * dW2
        b2 -= learning_rate * db2

        if (epoch + 1) % 100 == 0:
            loss = np.mean(np.square(y - a2))
            print(f'Epoch {epoch + 1}/{epochs}, Loss: {loss:.4f}')

    return W1, b1, W2, b2

def plot_confusion_matrix(conf_matrix, fold_num):
    plt.figure(figsize=(8, 6))
    # Tạo labels cho trục bắt đầu từ 1
    num_classes = conf_matrix.shape[0]
    labels = list(range(1, num_classes + 1))

    # Vẽ heatmap với labels mới
    sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
                xticklabels=labels,
                yticklabels=labels)

    plt.title(f'Ma trận nhầm lẫn - Fold {fold_num}')
    plt.ylabel('Nhãn thực tế')
    plt.xlabel('Nhãn dự đoán')
    plt.show()

def predict(X, W1, b1, W2, b2):
    # Forward propagation
    _, _, _, a2 = forward_propagation(X, W1, b1, W2, b2)
    return np.argmax(a2, axis=1)
```

```
def predict_and_evaluate(X_test, y_test, W1, b1, W2, b2, le):
    y_pred = predict(X_test, W1, b1, W2, b2)

    # Convert one-hot encoded test labels back to original labels
    y_test_orig = np.argmax(y_test, axis=1) if len(y_test.shape) > 1 else y_test

    # Convert numeric labels back to original class labels
    y_test_labels = le.inverse_transform(y_test_orig)
    y_pred_labels = le.inverse_transform(y_pred)

    print("\nSo sánh chi tiết dự đoán:")
    print("Ground Truth: ", end="")
    for true in y_test_labels:
        print(f"{true:3}", end=" ")
    print("\nPredicted:    ", end="")
    for pred in y_pred_labels:
        print(f"{pred:3}", end=" ")
    print("\n")

    # Print summary for each class
    counts = Counter(zip(y_test_labels, y_pred_labels))

    # Calculate metrics
    conf_matrix = confusion_matrix(y_test_orig, y_pred)
    accuracy = accuracy_score(y_test_orig, y_pred)
    precision, recall, f1, _ = precision_recall_fscore_support(y_test_orig, y_pred,
                                                               average='weighted')

    return conf_matrix, accuracy, precision, recall, f1
# Đọc và chuẩn bị dữ liệu
data = pd.read_csv("hu.csv")
X = data.iloc[:, 1:8].values  # 7 Hu moments
y = data['Labels'].values

# Chuẩn hóa dữ liệu
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Chuyển đổi labels thành one-hot encoding
le = LabelEncoder()
y = le.fit_transform(y)
y_onehot = np.eye(len(np.unique(y)))[y]

# Thiết lập k-fold cross validation
# skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
skf = KFold(n_splits=5, shuffle=True, random_state=42)
all_metrics = []
total_conf_matrix = None
fold_num = 1
```

```
print("Bắt đầu huấn luyện và đánh giá mô hình...\n")
for fold, (train_index, test_index) in enumerate(skf.split(X, y), 1):
    print(f"{'=' * 50}")
    print(f"Fold {fold}")
    print(f"{'=' * 50}")
    # Split data for this fold
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y_onehot[train_index], y_onehot[test_index]
    # Train model
    W1, b1, W2, b2 = train_model(X_train, y_train,
                                 hidden_size=15,
                                 learning_rate=0.05,
                                 epochs=2000)
    # Evaluate and show detailed predictions
    conf_matrix, accuracy, precision, recall, f1 = predict_and_evaluate(
                                                    X_test, y_test, W1, b1, W2, b2, le)

    if total_conf_matrix is None:
        total_conf_matrix = conf_matrix
    else:
        total_conf_matrix += conf_matrix

    # Save metrics
    all_metrics.append({
        'Fold': fold,
        'Accuracy': accuracy,
        'Precision': precision,
        'Recall': recall,
        'F1-score': f1
    })
    # Print fold results
    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1-score: {f1:.4f}")
    print("\nConfusion Matrix:")
    print(conf_matrix)
    plot_confusion_matrix(conf_matrix, fold_num)
    fold_num += 1
print(f"{'=' * 50}")
# Hiển thị confusion matrix gộp
print("Confusion Matrix after 5 fold:")
print(total_conf_matrix)

plt.figure(figsize=(8, 6))
# Tạo labels bắt đầu từ 1
num_classes = total_conf_matrix.shape[0]
labels = list(range(1, num_classes + 1))

# Thêm labels vào heatmap
sns.heatmap(total_conf_matrix,
            annot=True,
            fmt='d',
            cmap='Blues',
            xticklabels=labels,  # Thêm labels cho trục x
            yticklabels=labels)  # Thêm labels cho trục y

plt.title("Ma trận nhầm lẫn sau 5 Fold")
plt.ylabel('Nhãn thực tế')
plt.xlabel('Nhãn dự đoán')
plt.show()

# Tính toán và in metrics trung bình
metrics_df = pd.DataFrame(all_metrics)
print(f"\n{metrics_df.mean()[1:].round(4)}")
```

3.6 Mô hình phân loại lá cây ANN với đặc trưng HOG và đánh giá

```
import numpy as np
import pandas as pd
from sklearn.model_selection import StratifiedKFold, KFold
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import confusion_matrix, accuracy_score, precision_recall_fscore_support
import seaborn as sns
import matplotlib.pyplot as plt
from collections import Counter

def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def sigmoid_derivative(x):
    return x * (1 - x)

def initialize_parameters(input_size, hidden_size, output_size):
    # Khởi tạo trọng số ngẫu nhiên
    W1 = np.random.randn(input_size, hidden_size) * 0.01
    b1 = np.zeros((1, hidden_size))
    W2 = np.random.randn(hidden_size, output_size) * 0.01
    b2 = np.zeros((1, output_size))
    return W1, b1, W2, b2

def forward_propagation(X, W1, b1, W2, b2):
    # Net function là tuyến tính, và activation là sigmoid cho hidden layer
    z1 = np.dot(X, W1) + b1  # net function tuyến tính
    a1 = sigmoid(z1)  # hàm kích hoạt sigmoid

    # Net function và activation cho output layer
    z2 = np.dot(a1, W2) + b2  # net function tuyến tính
    a2 = sigmoid(z2)  # hàm kích hoạt sigmoid

    return z1, a1, z2, a2
def backward_propagation(X, y, a1, a2, W2):
    m = X.shape[0]
    # Tính đạo hàm
    dz2 = a2 - y
    dW2 = np.dot(a1.T, dz2) / m
    db2 = np.sum(dz2, axis=0, keepdims=True) / m

    da1 = np.dot(dz2, W2.T)
    dz1 = da1 * sigmoid_derivative(a1)
    dW1 = np.dot(X.T, dz1) / m
    db1 = np.sum(dz1, axis=0, keepdims=True) / m

    return dW1, db1, dW2, db2
```

```python
def train_model(X, y, hidden_size, learning_rate, epochs):
    input_size = X.shape[1]
    output_size = y.shape[1]
    # Khởi tạo tham số
    W1, b1, W2, b2 = initialize_parameters(input_size, hidden_size, output_size)
    # Training
    for epoch in range(epochs):
        # Forward propagation
        z1, a1, z2, a2 = forward_propagation(X, W1, b1, W2, b2)
        # Backward propagation
        dW1, db1, dW2, db2 = backward_propagation(X, y, a1, a2, W2)
        # Cập nhật tham số
        W1 -= learning_rate * dW1
        b1 -= learning_rate * db1
        W2 -= learning_rate * dW2
        b2 -= learning_rate * db2

        if (epoch + 1) % 100 == 0:
            loss = np.mean(np.square(y - a2))
            print(f'Epoch {epoch + 1}/{epochs}, Loss: {loss:.4f}')

    return W1, b1, W2, b2

def plot_confusion_matrix(conf_matrix, fold_num):
    plt.figure(figsize=(8, 6))
    # Tạo labels cho trục bắt đầu từ 1
    num_classes = conf_matrix.shape[0]
    labels = list(range(1, num_classes + 1))

    # Vẽ heatmap với labels mới
    sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
                xticklabels=labels,
                yticklabels=labels)

    plt.title(f'Ma trận nhầm lẫn - Fold {fold_num}')
    plt.ylabel('Nhãn thực tế')
    plt.xlabel('Nhãn dự đoán')
    plt.show()

def predict(X, W1, b1, W2, b2):
    # Forward propagation
    _, _, _, a2 = forward_propagation(X, W1, b1, W2, b2)
    return np.argmax(a2, axis=1)
```

```python
def predict_and_evaluate(X_test, y_test, W1, b1, W2, b2, le):
    # Get predictions
    y_pred = predict(X_test, W1, b1, W2, b2)

    # Convert one-hot encoded test labels back to original labels
    y_test_orig = np.argmax(y_test, axis=1) if len(y_test.shape) > 1 else y_test

    # Convert numeric labels back to original class labels
    y_test_labels = le.inverse_transform(y_test_orig)
    y_pred_labels = le.inverse_transform(y_pred)

    # Print detailed comparison horizontally
    print("\nSo sánh chi tiết dự đoán:")
    print("Ground Truth: ", end="")
    for true in y_test_labels:
        print(f"{true:3}", end=" ")
    print("\nPredicted:    ", end="")
    for pred in y_pred_labels:
        print(f"{pred:3}", end=" ")
    print("\n")

    # Print summary for each class
    counts = Counter(zip(y_test_labels, y_pred_labels))

    # Calculate metrics
    conf_matrix = confusion_matrix(y_test_orig, y_pred)
    accuracy = accuracy_score(y_test_orig, y_pred)
    precision, recall, f1, _ = precision_recall_fscore_support(y_test_orig, y_pred,
                                                               average='weighted')

    return conf_matrix, accuracy, precision, recall, f1

# Đọc và chuẩn bị dữ liệu
data = pd.read_csv("output_csv/hog 1.csv")
X = data.iloc[:, 1:513].values  # 512 HOG
y = data['Labels'].values

# Chuẩn hóa dữ liệu
scaler = StandardScaler()
X = scaler.fit_transform(X)

# Chuyển đổi labels thành one-hot encoding
le = LabelEncoder()
y = le.fit_transform(y)
y_onehot = np.eye(len(np.unique(y)))[y]

# Thiết lập k-fold cross validation
# skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
skf = KFold(n_splits=5, shuffle=True, random_state=42)
all_metrics = []
total_conf_matrix = None
fold_num = 1
```

```
print("Bắt đầu huấn luyện và đánh giá mô hình...\n")
for fold, (train_index, test_index) in enumerate(skf.split(X, y), 1):
    print(f"{'=' * 50}")
    print(f"Fold {fold}")
    print(f"{'=' * 50}")
    # Split data for this fold
    X_train, X_test = X[train_index], X[test_index]
    y_train, y_test = y_onehot[train_index], y_onehot[test_index]
    # Train model
    W1, b1, W2, b2 = train_model(X_train, y_train,
                                 hidden_size=15,
                                 learning_rate=0.05,
                                 epochs=2000)
    # Evaluate and show detailed predictions
    conf_matrix, accuracy, precision, recall, f1 = predict_and_evaluate(
        X_test, y_test, W1, b1, W2, b2, le)

    if total_conf_matrix is None:
        total_conf_matrix = conf_matrix
    else:
        total_conf_matrix += conf_matrix

    # Save metrics
    all_metrics.append({
        'Fold': fold,
        'Accuracy': accuracy,
        'Precision': precision,
        'Recall': recall,
        'F1-score': f1
    })
    # Print fold results
    print(f"Accuracy: {accuracy:.4f}")
    print(f"Precision: {precision:.4f}")
    print(f"Recall: {recall:.4f}")
    print(f"F1-score: {f1:.4f}")
    print("\nConfusion Matrix:")
    print(conf_matrix)
    plot_confusion_matrix(conf_matrix, fold_num)
    fold_num += 1


print(f"{'=' * 50}")
# Hiển thị confusion matrix gộp
print("Confusion Matrix after 5 fold:")
print(total_conf_matrix)

plt.figure(figsize=(8, 6))
# Tạo labels bắt đầu từ 1
num_classes = total_conf_matrix.shape[0]
labels = list(range(1, num_classes + 1))

# Thêm labels vào heatmap
sns.heatmap(total_conf_matrix,
            annot=True,
            fmt='d',
            cmap='Blues',
            xticklabels=labels,  # Thêm labels cho trục x
            yticklabels=labels)  # Thêm labels cho trục y

plt.title("Ma trận nhầm lẫn sau 5 Fold")
plt.ylabel('Nhãn thực tế')
plt.xlabel('Nhãn dự đoán')
plt.show()

# Tính toán và in metrics trung bình
metrics_df = pd.DataFrame(all_metrics)
print(f"\n{metrics_df.mean()[1:].round(4)}")
```