

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN 1
--- & ---



BÁO CÁO

MÔN: HỆ CSDL ĐA PHƯƠNG TIỆN

ĐỀ TÀI: XÂY DỰNG HỆ CSDL LƯU TRỮ VÀ NHẬN DẠNG ẢNH HOA

Giảng viên: Nguyễn Đình Hóa
Lớp học phần: Nhóm 03
Nhóm thực hiện: Nhóm 12

MSV	Họ và tên
B19DCCN314	Nguyễn Quang Huy
B19DCCN142	Tạ Đình Duy
B19DCCN229	Nguyễn Công Hậu

Hà Nội, 06/2023

MỤC LỤC

Lời nói đầu.....	4
Yêu Cầu.....	5
Phần 1. Xây dựng bộ dữ liệu ảnh hoa	5
1. Các nguồn dữ liệu:	5
2. Hình ảnh	7
3. Cách lưu trữ ảnh :	9
4.Miêu tả:	9
Phần 2. Một số kỹ thuật xử lý và tìm kiếm ảnh hiện hành	10
I, Các kỹ thuật xử lý ảnh (trích rút đặc trưng ảnh)	10
Giới thiệu :	10
1. Color Histogram	10
2. HOG	11
3. SIFT	13
4. SURF	14
5. LBP (Local Binary Patterns)	15
6. GLCM:.....	16
II, Các kỹ thuật phân loại hình ảnh	17
Giới thiệu	17
1. K-nearest neighbors (KNN)	18
2. Support vector machine (SVM).....	19
3. Naïve bayes classification (NBC)	20
4. Deep learning.....	21
Phần 3. Hệ thống nhận dạng và tìm kiếm ảnh	23
I, Sơ đồ khối và quy trình thực hiện	23
1. Sơ đồ khối	23

2. Quy trình thực hiện	23
II. Thuộc tính và các kỹ thuật trích rút	24
1.Thuộc tính sử dụng về màu sắc	24
2.Thuộc tính sử dụng về hình dạng	30
III. Lưu trữ, quản lý và tìm kiếm trong CSDL.....	40
1. Lưu trữ và quản lý:	40
2. Tìm kiếm:.....	41
3.Kết quả đạt được :	42
Phần 4. Cài đặt chương trình +DEMO.....	42
I. Module trích xuất đặc trưng hình ảnh	43
II. Module tìm kiếm	46
III. Kết quả kiểm nghiệm	47
Phần 5: TÀI LIỆU THAM KHẢO.....	49

LỜI NÓI ĐẦU

Lời đầu tiên, chúng em xin gửi lời cảm ơn đến Học viện Công nghệ Bưu chính Viễn Thông đã tạo điều kiện cho chúng em được học môn Hệ cơ sở dữ liệu đa phương tiện. Đặc biệt, chúng em xin gửi lời cảm ơn chân thành và sâu sắc nhất tới thầy Nguyễn Đình Hóa, giảng viên bộ môn đã hướng dẫn và truyền đạt những kiến thức hết sức bổ ích và quý báu trong suốt thời gian học tập vừa qua. Với vốn hiểu biết sâu rộng và kinh nghiệm nhiều năm giảng dạy cũng như làm việc trong môi trường công nghệ thông tin, thầy khiến chúng em thật sự ấn tượng, khâm phục và “ngỡ ngàng” trước những hiểu biết của thầy.

Hệ cơ sở dữ liệu đa phương tiện là một môn học thật sự rất hay và bổ ích tuy nhiên đây là môn có khối lượng kiến thức tương đối nhiều và khó có thể hiểu rõ, hiểu sâu nhanh chóng khi thời lượng học trên lớp có hạn. Mặc dù thầy đã truyền đạt nhiệt tình và tận tâm nhưng do năng lực có hạn, khả năng tư duy và khả năng tiếp thu không nhanh nên trong báo cáo bài tập lớn này chúng em khó có thể tránh khỏi được những sai sót. Do đó, chúng em kính mong thầy xem xét và bổ sung giúp đỡ chúng hoàn thiện bài báo cáo này một cách đầy đủ và đúng đắn hơn. Một lần nữa, chúng em xin chân thành cảm ơn thầy !

Chúng em xin chân thành cảm ơn thầy!

Nhóm 12

YÊU CẦU

Xây dựng hệ CSDL nhận dạng ảnh hoa.

1. Hãy sưu tầm ít nhất 100 bức ảnh về ít nhất 10 loài hoa khác nhau, mỗi ảnh chỉ gồm 1 bông hoa, các bức ảnh đều có cùng kích thước và bố cục. Hãy trình bày đặc điểm của loại dữ liệu này.

2. Hãy tìm hiểu các kỹ thuật xử lý và phân loại ảnh hoa đang hiện hành.

3. Xây dựng hệ thống nhận dạng ảnh hoa với đầu vào là một ảnh mới của một bông hoa thuộc loài hoa đã có và chưa có trong phần 1, đầu ra là kết quả nhận dạng nhãn của loài hoa trong ảnh đầu vào.

a. Trình bày sơ đồ khối của hệ thống và quy trình thực hiện yêu cầu của đề bài.

b. Trình bày các thuộc tính được sử dụng để nhận dạng nhãn của ảnh hoa trong hệ thống, cùng các kỹ thuật để trích rút các thuộc tính đó.

c. Trình bày cách lưu trữ các thuộc tính ảnh hoa và cách nhận dạng ảnh hoa dựa trên các thuộc tính đó.

4. Demo hệ thống và đánh giá kết quả đã đạt được

PHẦN 1. XÂY DỰNG BỘ DỮ LIỆU ẢNH HOA

1. Các nguồn dữ liệu:

- Chủ đề: bộ dữ liệu ảnh hoa

- Miêu tả bộ dữ liệu: Nhóm thu thập bộ dữ liệu gồm tổng cộng 243 ảnh hoa, bao gồm 13 loài hoa khác nhau và được lựa chọn kỹ lưỡng trong các tập dataset sau:

 | [Flowers | Kaggle](#)

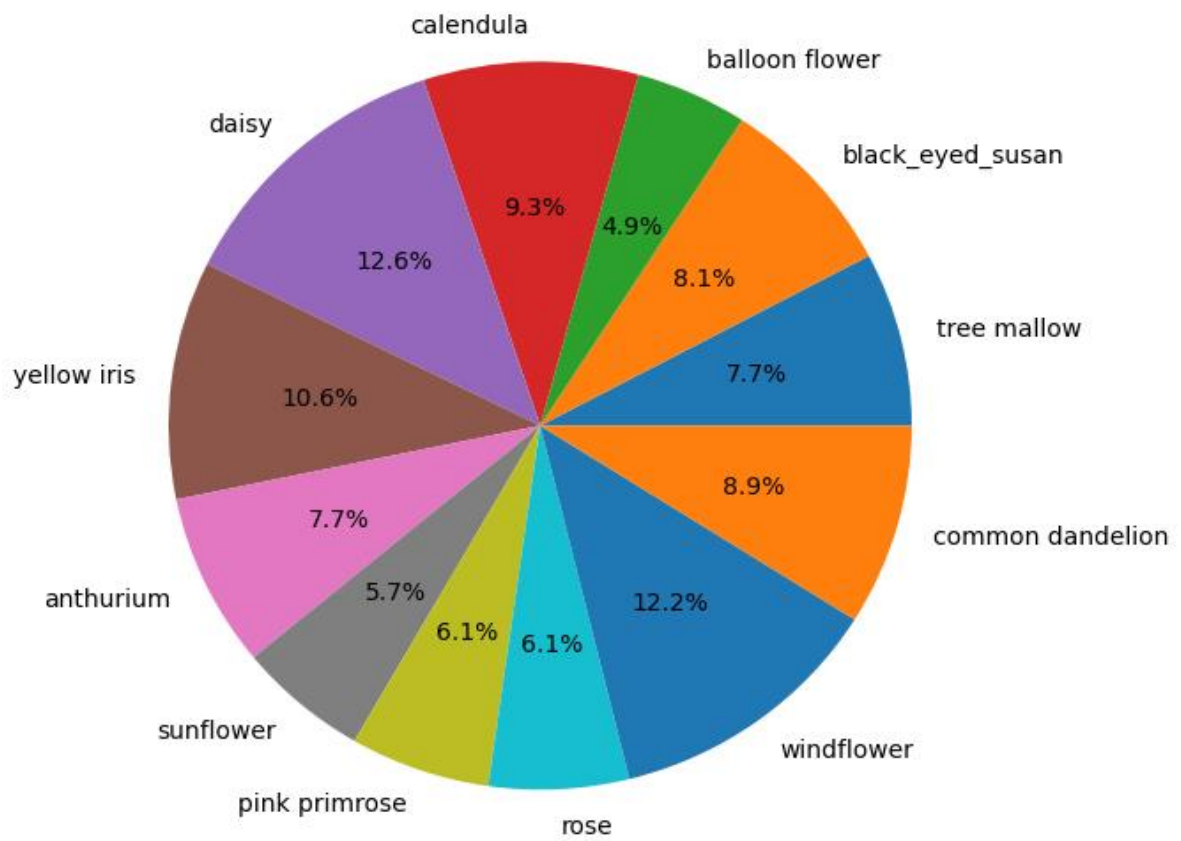
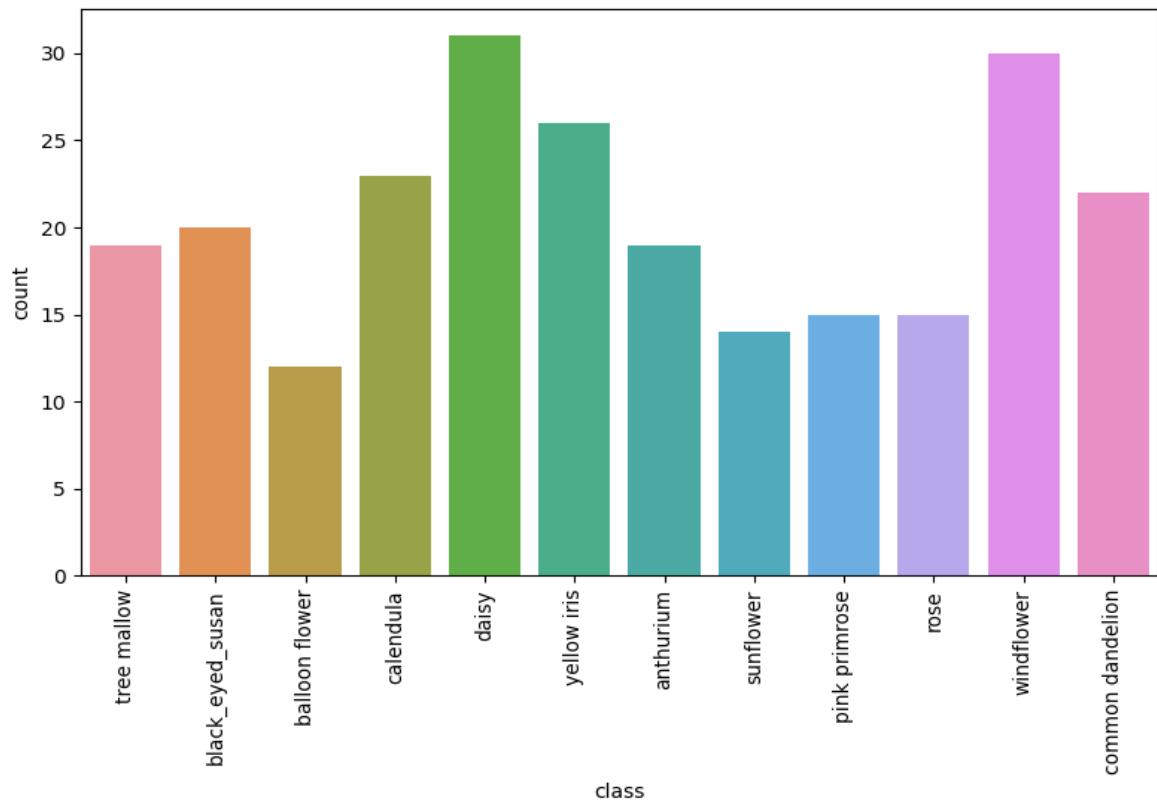
[Flower Dataset | Kaggle](#)

[Oxford 102 Flower Dataset | Kaggle](#)

[Hackathon Blossom \(Flower Classification\) | Kaggle](#)

[flower_data_13classes | Kaggle](#)

- Hình ảnh thể hiện mô tả về dataset :

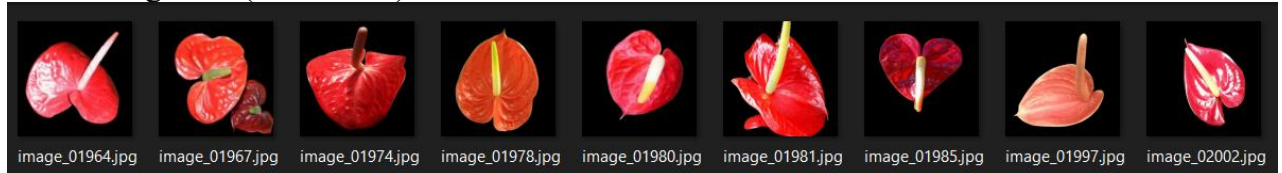


2. Hình ảnh

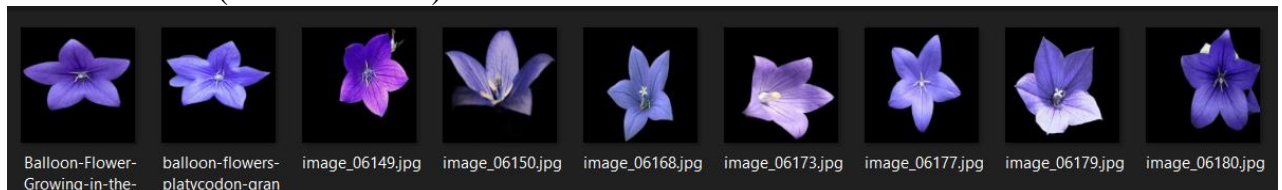
Bao gồm 12 loài hoa, cụ thể:

['black_eyed_susan', 'calendula', 'sunflower', 'rose', 'daisy', 'pink primrose', 'yellow iris', 'balloon flower', 'common dandelion', 'windflower', 'anthurium', 'tree mallow']

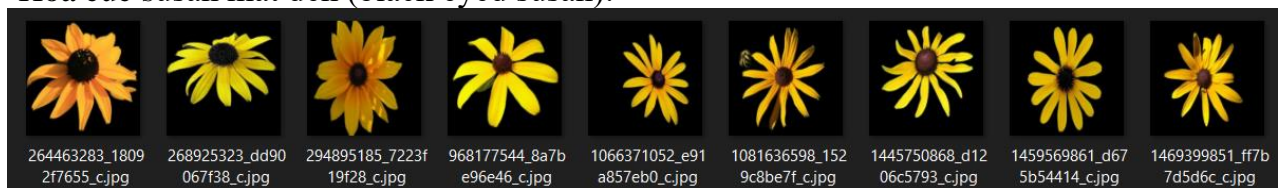
- Hoa hồng môn (anthurium):



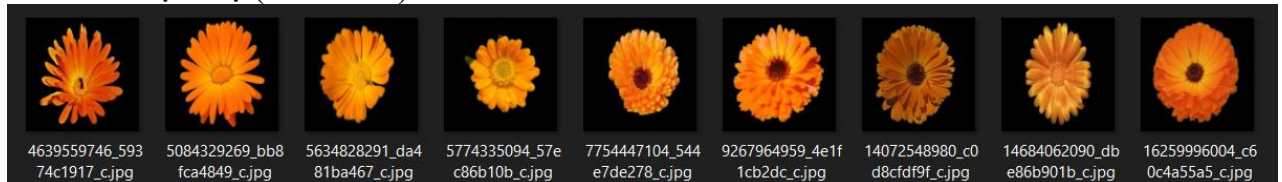
- Hoa cát cánh (balloon flower):



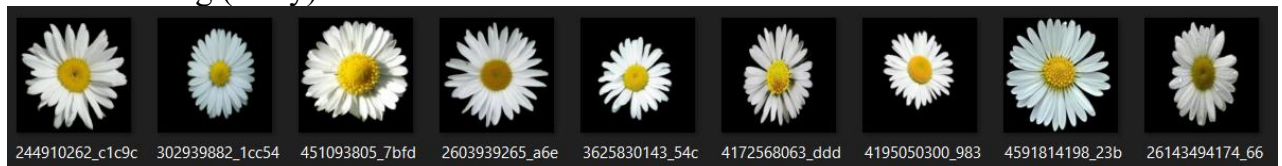
-Hoa cúc susan mắt đen (black eyed susan):



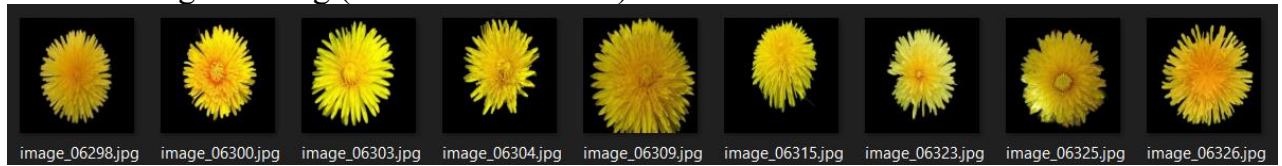
-Hoa cúc vạn thọ (calendula):



-Hoa cúc trắng (daisy):



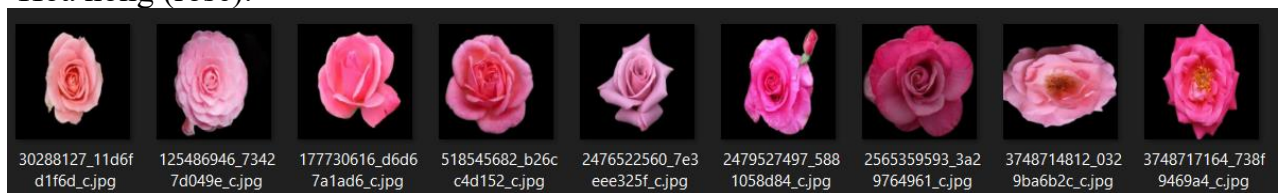
-Hoa bồ công anh vàng (common dandelion):



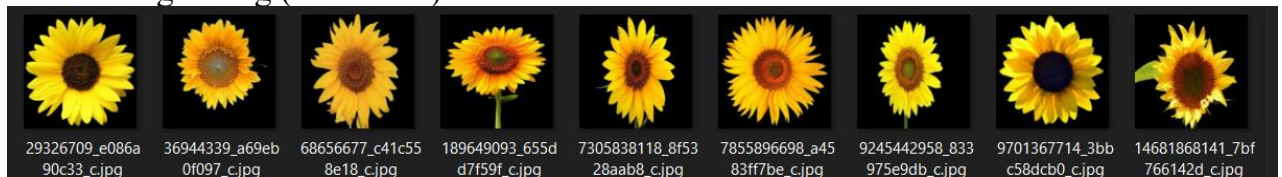
-Hoa anh thảo (pink primrose):



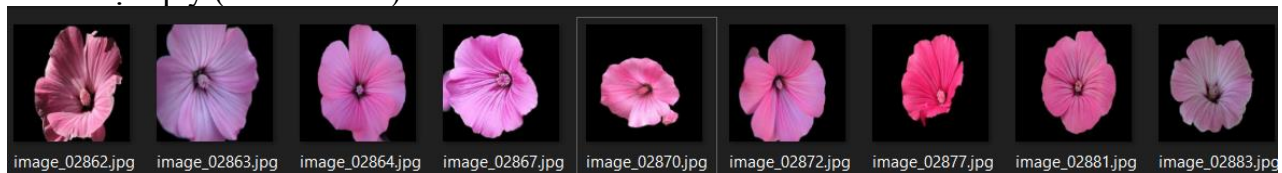
-Hoa hồng (rose):



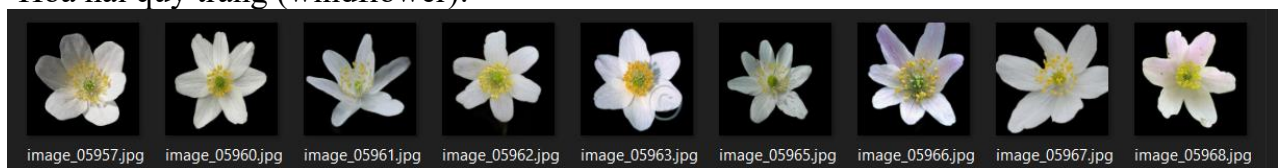
-Hoa hướng dương (sunflower):



-Hoa thực quỳ (tree mallow):



-Hoa hải quỳ trắng (windflower):

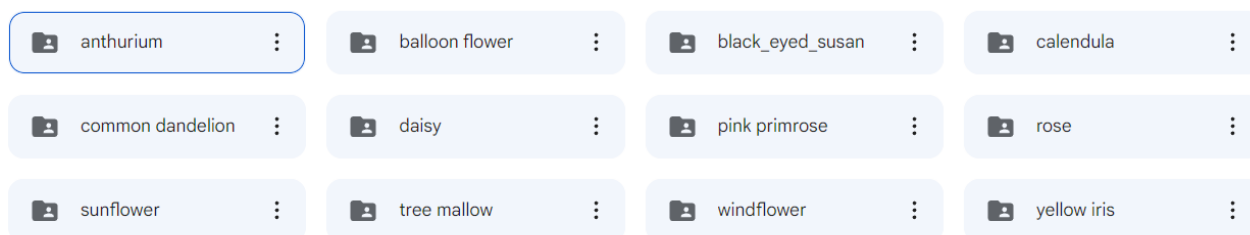


-Hoa diên vĩ vàng (yellow iris):



3. Cách lưu trữ ảnh :

Các hình ảnh của 13 loài hoa, được lưu vào 12 thư mục khác nhau.



4. Miêu tả:

- Bộ dữ liệu gồm tổng cộng 243 bức ảnh.

+ Kích thước ảnh của các bông hoa khác nhau đã được xóa background.

+ Định dạng tệp hình ảnh là “*.jpg”.

+ Bao gồm 12 loại hoa.

- Nhận xét về bộ dữ liệu:

+ Mỗi bức ảnh chỉ có một bông hoa, đa số các bức ảnh có bông hoa chiếm khoảng 60-70% diện tích của ảnh. Các bức ảnh đều đã được xóa nền chỉ còn lại mặt của bông hoa.

+ Các ảnh hoa cùng loại được chụp ở 1 vài góc độ, nhưng đa phần 80% là chụp hướng từ trên xuống, chính diện nghĩa là có thể nhìn thấy nhụy hoa, dễ dàng nhất trong việc đếm số lượng cánh hoa

+ Các hình ảnh của cùng 1 loại hoa có màu sắc giống nhau tuy nhiên có thể về độ bão hòa ảnh và độ sáng khác nhau do cách chụp khác nhau. Các loài hoa khác nhau thì có thể có màu sắc trùng với nhau (ví dụ như Hoa cúc susan mắt đen và Hoa bồ công anh vàng, hoa hướng dương đều có màu vàng)

+ Các hình ảnh của cùng 1 loại hoa cũng sẽ có hình dáng giống nhau. Tuy nhiên hình dáng của các loài hoa khác nhau lại không khác hoàn toàn (Ví dụ như Hoa cúc trắng và Hoa cúc vạn thọ có hình dáng tròn khá giống nhau).

+ Sự kết hợp giữa yếu tố màu sắc và hình dáng sẽ tạo nên đặc trưng của các loài hoa và có thể phân biệt các loài hoa với nhau. (Ví dụ hoa màu đỏ và hình dáng giống với hình trái tim thì sẽ là Hoa hồng môn)

=> Do đó hướng trích rút tập trung áp dụng đặc trưng về màu sắc và hình dạng.

PHẦN 2. MỘT SỐ KỸ THUẬT XỬ LÝ VÀ TÌM KIẾM ẢNH HIỆN HÀNH

I, Các kỹ thuật xử lý ảnh (trích rút đặc trưng ảnh)

Giới thiệu :

Một số đặc trưng nội dung của ảnh:

- Đặc trưng màu sắc: Màu sắc là một đặc trưng nổi bật và được sử dụng phổ biến nhất trong tìm kiếm ảnh theo nội dung. Mỗi một điểm ảnh (thông tin màu sắc) có thể được biểu diễn như một điểm trong không gian màu sắc ba chiều. Các không gian màu sắc thường dùng là: RGB, Munsell, CIE, HSV. Tìm kiếm ảnh theo màu sắc tiến hành tính toán biểu đồ màu cho mỗi ảnh để xác định tỉ trọng các điểm ảnh của ảnh mà chứa các giá trị đặc biệt (màu sắc). Các nghiên cứu gần đây đang cố gắng phân vùng ảnh theo các màu sắc khác nhau và tìm mối quan hệ giữa các vùng này.

- Đặc trưng kết cấu: Trích xuất nội dung ảnh theo kết cấu nhằm tìm ra mô hình trực quan của ảnh và cách thức chúng được xác định trong không gian. Kết cấu được biểu diễn bởi các texel mà sau đó được đặt vào một số các tập phụ thuộc vào số kết cấu được phát hiện trong ảnh. Các tập này không chỉ xác định các kết cấu mà còn chỉ rõ vị trí các kết cấu trong ảnh. Việc xác định các kết cấu đặc biệt trong ảnh đạt được chủ yếu bằng cách mô hình các kết cấu như những biến thể cấp độ xám 2 chiều.

- Đặc trưng hình dạng: Màu sắc và kết cấu là những thuộc tính có khái niệm toàn cục trong một ảnh. Trong khi đó, hình dạng không phải là một thuộc tính của ảnh. Nói tới hình dạng không phải là nhắc đến hình dạng của một ảnh. Thay vì vậy, hình dạng có khuynh hướng chỉ đến một khu vực đặc biệt trong ảnh, hay hình dạng chỉ là biên của một đối tượng nào đó trong ảnh. Mục tiêu chính của biểu diễn hình dạng trong nhận dạng mẫu là đo thuộc tính hình học của một đối tượng được dùng trong phân lớp, so sánh và nhận dạng đối tượng.

- Đặc trưng cục bộ bất biến (local invariant feature) là một thuật toán xử lý ảnh để tìm ra các điểm đặc biệt trên hình ảnh và tạo ra các đặc trưng (feature) độc lập với vị trí và tỷ lệ. Các đặc trưng này được tạo ra bằng cách tính toán các giá trị đặc biệt (như gradient, độ cong, ...) tại các điểm cụ thể trên hình ảnh và biểu diễn chúng dưới dạng một vector đặc trưng. Đặc trưng cục bộ bất biến thường được sử dụng trong các ứng dụng như nhận dạng đối tượng, phát hiện khuôn mặt, phân loại ảnh và định vị vật thể. Với các đặc trưng này, các hình ảnh có thể được so sánh và phân loại dễ dàng hơn, mà không phụ thuộc vào các biến đổi vị trí, góc quay và tỷ lệ của đối tượng.

Có rất nhiều kỹ thuật để có thể trích xuất đặc trưng của ảnh. Nhóm đã tìm hiểu một vài kỹ thuật phổ biến dưới đây:

1. Color Histogram

- Histogram của một ảnh số biểu diễn sự phân bố số lượng điểm ảnh tương ứng với một giá trị màu có trong ảnh (giá trị đó có thể là một bộ giá trị RGB đối ảnh RGB,

HSV đối với ảnh HSV hay giá trị mức xám đối ảnh xám).

- Color histogram là một dạng đặc trưng toàn cục biểu diễn phân phối của các màu trên ảnh. Color histogram thống kê số lượng các pixel có giá trị nằm trong một khoảng màu nhất định (bins) cho trước. Color histogram có thể tính trên các dạng ảnh RGB hoặc HSV, thông dụng là HSV (Hue – vùng màu, Saturation – độ bão hòa màu, Value – độ sáng).

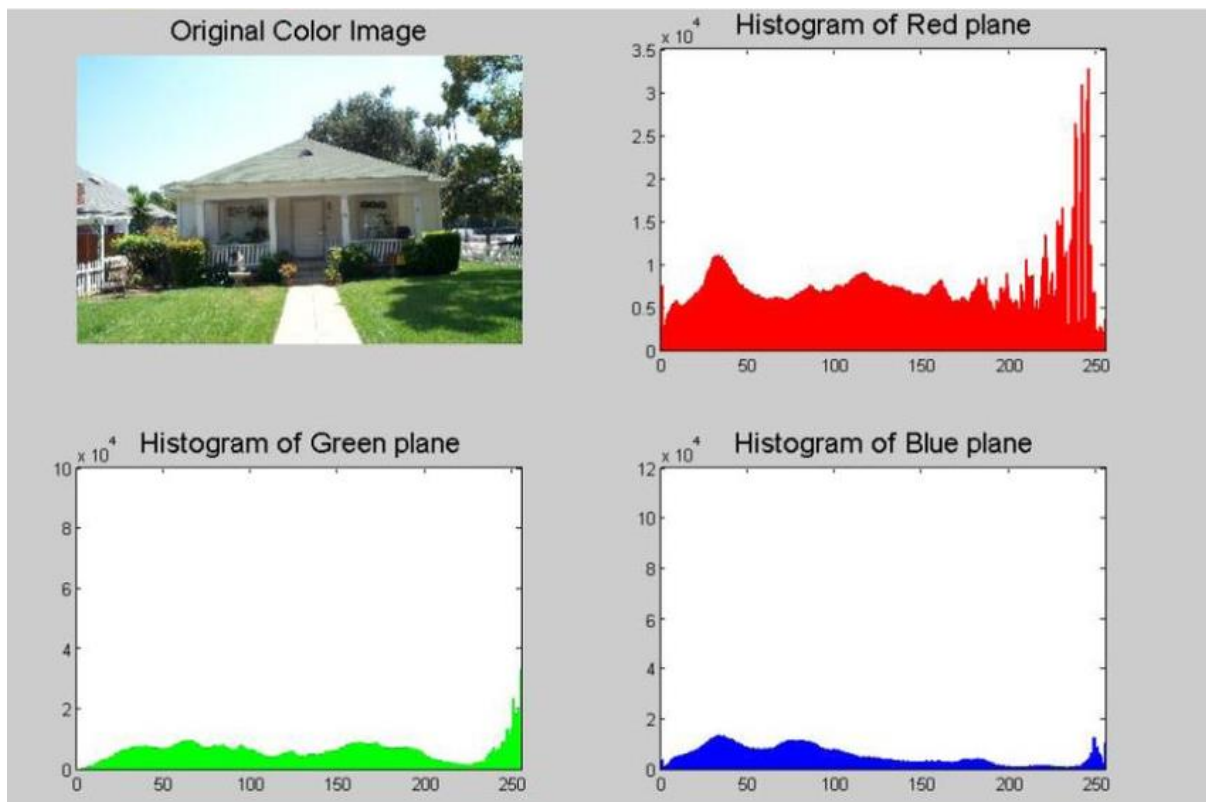
- Color histogram thống kê số lượng các pixel có giá trị nằm trong một khoảng màu nhất định cho trước (RGB hoặc HSV) với mỗi kênh màu cố định, chia kênh màu này thành n bin. Sau đó, thống kê số lượng pixel trên ảnh có giá trị màu thuộc về “ n bin” này. Cuối cùng, nối các bin của các kênh màu lại với nhau để tạo thành đặc trưng.

- Ưu điểm

- + Tính toán nhanh, đơn giản, thích hợp trong các ứng dụng thời gian thực
- + Khi ảnh bị xoay đi thì phân phối màu hầu như là không đổi

- Nhược điểm:

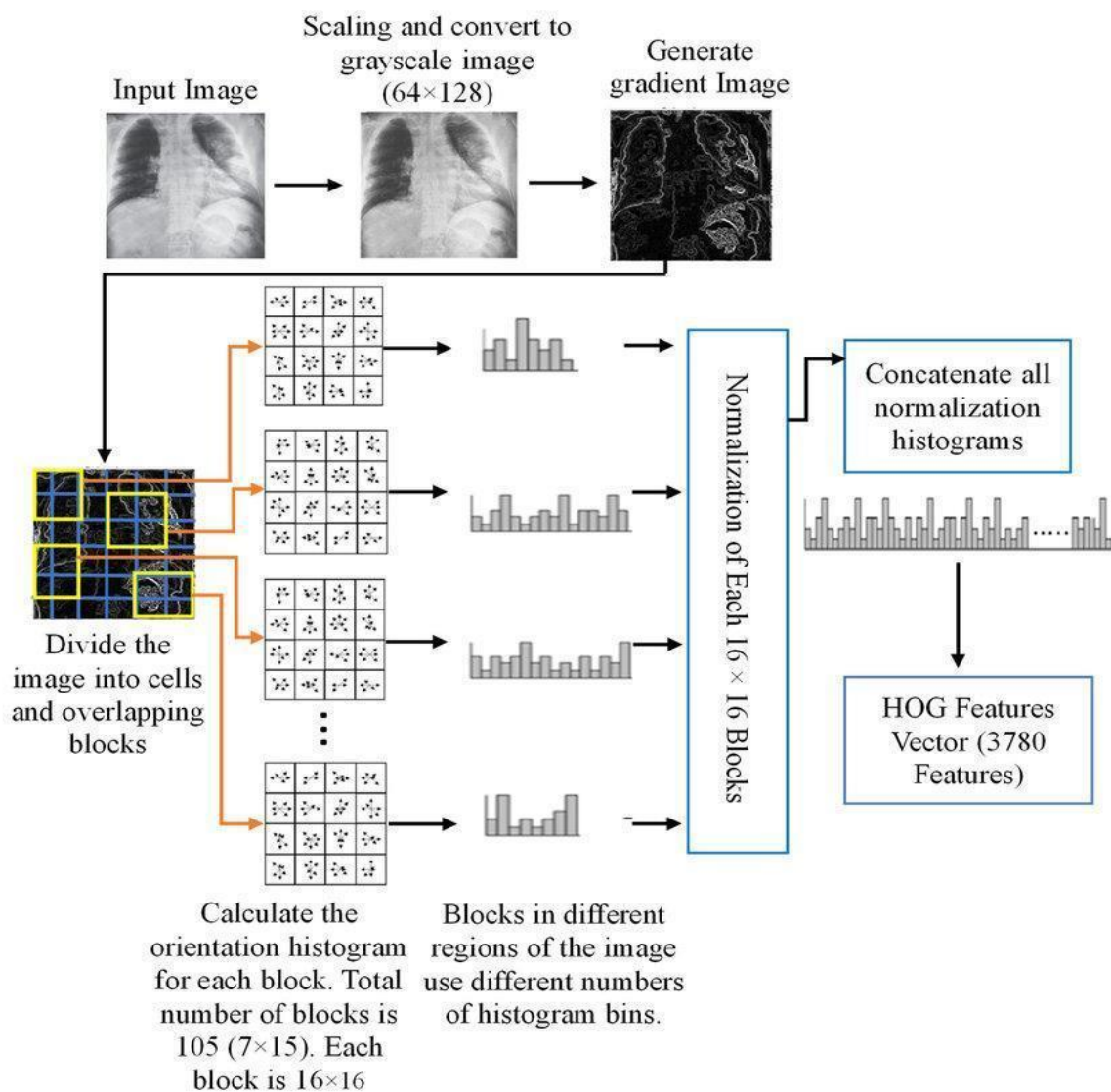
- + Không nói lên được sự tương đồng về hình dáng, cấu trúc ảnh
- + Dễ bị nhiễu với thanh đổi về cường độ sáng



2. HOG

- HOG (histogram of oriented gradients) là một feature descriptor được sử dụng trong computer vision và xử lý hình ảnh tập trung vào cấu trúc và hình dạng của một đối tượng. HOG tương tự như các biểu đồ edge orientation, scale-invariant feature transform descriptors, shape contexts nhưng HOG được tính toán trên một lưới dày đặc các cell và chuẩn hóa sự tương phản giữa các block để nâng cao độ chính xác. HOG được sử dụng chủ yếu để mô tả hình dạng và sự xuất hiện của một object trong ảnh.

- Bài toán tính toán Hog thường gồm 5 bước:
 - + Chuẩn hóa hình ảnh trước khi xử lý
 - + Tính toán gradient theo cả hướng x và y.
 - + Lấy phiếu bầu cùng trọng số trong các cell
 - + Chuẩn hóa các block
 - + Thu thập tất cả các biểu đồ cường độ gradient định hướng để tạo ra feature vector cuối cùng.
- Ưu điểm:
 - + Khả năng trích xuất đặc trưng tốt
 - + Độ chính xác cao
 - + Dễ dàng tích hợp với các phương pháp khác
- Nhược điểm:
 - + Không hiệu quả với các đối tượng nhỏ
 - + Có thể bị đánh lừa bởi các đối tượng giống nhau
 - + Thời gian tính toán lâu



3. SIFT

- SIFT (Scale-invariant feature transform) là một feature descriptor được sử dụng trong computer vision và xử lý hình ảnh được dùng để nhận dạng đối tượng, matching image, hay áp dụng cho các bài toán phân loại... SIFT cho phép tìm ra các đặc trưng cục bộ của một hình ảnh một cách tự động và khả năng chống lại các biến đổi về tỷ lệ và góc độ

- SIFT bao gồm các bước:

+ Scale-space extrema detection: SIFT sử dụng một khối lọc LoG để tìm kiếm các điểm đặc trưng trên nhiều tỷ lệ khác nhau. Các khối lọc LoG được áp dụng trên hình ảnh gốc với các tỷ lệ khác nhau để tìm kiếm các điểm đặc trưng ở các tỷ lệ khác nhau.

+ Key point localization: Sau khi tìm được các điểm cực đại trên các đáp ứng của khối lọc LoG, SIFT sử dụng các phương pháp tiêu chuẩn để xác định vị trí chính xác của các điểm đặc trưng.

+ Orientation assignment: Với mỗi điểm đặc trưng, SIFT tính toán hướng của gradient tại vị trí điểm đó để gán một hướng cho điểm đặc trưng.

+ Descriptor computation: SIFT tính toán một vector mô tả 128 chiều (hay còn gọi là descriptor) cho mỗi điểm đặc trưng bằng cách tính toán độ lớn và hướng của gradient cho các pixel trong một vùng xung quanh của điểm đặc trưng. Sau đó, vector mô tả được chuẩn hóa để giảm tác động của ánh sáng. Các descriptor này sẽ được dùng để nhận dạng đối tượng trong ảnh, hay dùng cho các bài toán classification

- Hình ảnh sau khi áp dụng biến đổi SIFT, ứng với mỗi keypoint ta sẽ thu được: tọa độ keypoint, scale và orientation của keypoint, descriptor.

- Ưu điểm:

+ Các keypoint sẽ ít bị phụ thuộc bởi cường độ sáng, nhiễu, góc xoay của ảnh do các descriptor được tạo ra từ gradients do đó nó đã bất biến với các thay đổi về độ sáng (ví dụ: thêm 10 vào tất cả các pixel hình ảnh sẽ mang lại cùng một mô tả chính xác).

+ Có thể xử lý khi xoay ảnh

+ Tìm được các đặc trưng của ảnh mà không bị ảnh hưởng bởi sự biến đổi tỷ lệ. Linh hoạt trong việc áp dụng cho mọi kích thước ảnh

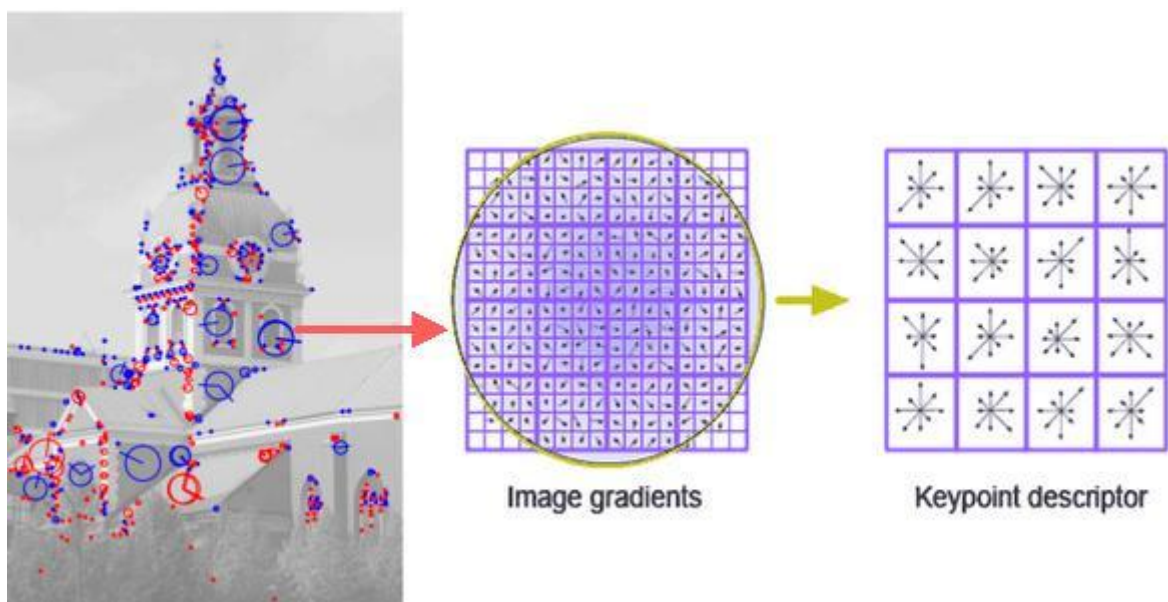
+ Tính ổn định và độ tin cậy cao

- Nhược điểm:

+ Tốc độ xử lý chậm hơn

+ Độ phức tạp tính toán cao

+ Số lượng điểm đặc trưng lớn



4. SURF

- SURF (Speeded-Up Robust Features) cũng gồm các bước như ở SIFT:

- + Scale-space extrema detection.
- + Keypoint localization.
- + Orientation assignment.
- + Keypoint descriptor.

Nhưng, ở từng bước SURF sẽ có những sự cải thiện để cải thiện tốc độ xử lý mà vẫn đảm bảo độ chính xác trong việc detection.

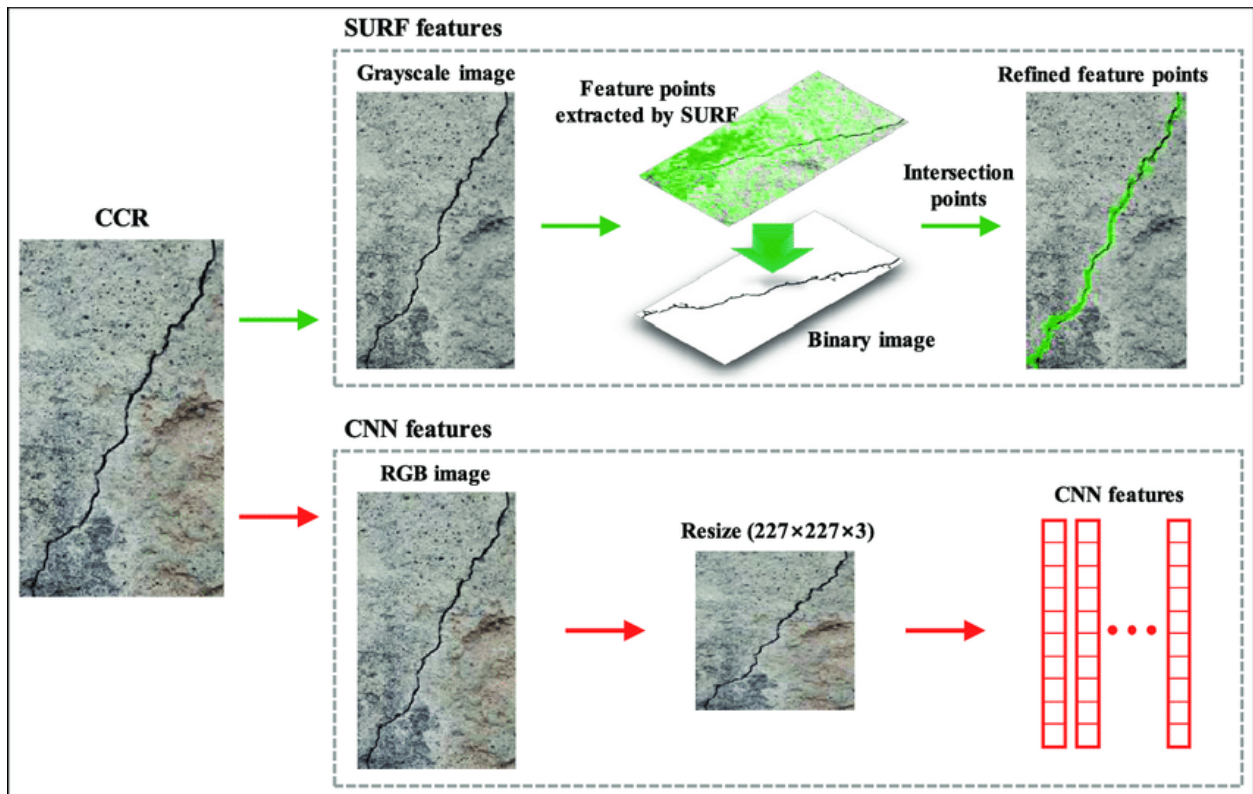
- Ở SIFT, việc tìm Scale-space dựa trên việc tính gần đúng LoG (Laplace of Gaussian) dùng DoG (Difference of Gaussian), trong khi đó SURF sử dụng Box Filter, tốc độ xử lý sẽ được cải thiện đáng kể với việc dùng ảnh tích phân (integral image)

- Ở bước Orientation Assignment, SURF sử dụng wavelet response theo 2 chiều dọc và ngang, sau đó tính hướng chính bằng cách tính tổng các response đó, có một điều đáng chú ý là wavelet response cũng dễ dàng tính được với ảnh tích phân (integral image)

- Ưu điểm :

+ Khả năng tính toán nhanh các toán tử sử dụng bộ lọc hộp, do đó cho phép các ứng dụng thời gian thực như theo dõi và nhận dạng đối tượng.(tốt hơn so với SIFT)

- Nhược điểm :



5. LBP (Local Binary Patterns)

- Local Binary Patterns (LBP) là một toán tử đơn giản nhưng rất hiệu quả, nó gắn nhãn các pixel của hình ảnh bằng cách phân ngưỡng các pixel xung quanh và đưa ra ảnh kết quả dưới dạng nhị phân. Do khả năng phân tách tốt và phi phí tính toán thấp, toán tử LBP đã trở thành một phương pháp phổ biến trong các ứng dụng khác nhau. Có lẽ tính chất quan trọng nhất của toán tử LBP trong các ứng dụng thực tế là sự chống chịu của nó đối với các thay đổi mức xám, ví dụ như biến đổi về ánh sáng. Một tính chất quan trọng khác là tính đơn giản tính toán của nó, phục vụ cho việc xử lý hình ảnh trong thời gian thực.

- Quy tắc tìm LBP của một hình ảnh như sau:

- + Đặt giá trị pixel làm pixel trung tâm.
- + Thu thập các pixel lân cận của nó (thông thường lấy ma trận 3×3 ; tổng số pixel lân cận là 8)
- + Đặt ngưỡng giá trị pixel lân cận của nó thành 1 nếu giá trị của nó lớn hơn hoặc bằng giá trị pixel trung tâm, nếu không, đặt ngưỡng đó thành 0.
- + Sau khi tạo ngưỡng, thu thập tất cả các giá trị ngưỡng từ vùng lân cận theo chiều kim đồng hồ hoặc ngược chiều kim đồng hồ. Bộ sưu tập sẽ cung cấp cho bạn mã nhị phân gồm 8 chữ số. Chuyển đổi mã nhị phân thành số thập phân.
- + Thay thế giá trị pixel trung tâm bằng số thập phân kết quả và thực hiện quy trình tương tự cho tất cả các giá trị pixel có trong hình ảnh.

- Ưu điểm:

- + Khả năng phân loại cao: Điều này đã được kiểm chứng ở nhiều nghiên cứu khác nhau, trên các tập dữ liệu và bài toán khác nhau.

+ Đơn giản và chi phí tính toán thấp: So với các đặc trưng khác như SIFT, SURF hay HOG, LBP yêu cầu ít bước tính toán hơn và các bước tính toán cũng đơn giản hơn

+ Chống chịu với thay đổi độ sáng: Do LBP được cấu trúc từ mức xám tương đối giữa các pixel (hiệu của pixel láng giềng và pixel trung tâm) nên nó không bị ảnh hưởng khi mức xám biến đổi do chiếu sáng.

- Nhược điểm:

+ Không chống chịu với xoay (rotation variant): Khi hình ảnh xoay, các giá trị mức xám sẽ di chuyển dọc theo chu vi của vòng tròn xung quanh pixel trung tâm. Điều này khiến chuỗi nhị phân cũng xoay theo và cho ra giá trị khác.

+ Độ dài vector đặc trưng tăng theo hàm số mũ khi số lượng pixel láng giềng tăng: Khi xét n pixel láng giềng, độ dài của vector đặc trưng (histogram) là 2^n sẽ tăng nhanh khi n tăng.

+ Làm mất thông tin về cấu trúc không gian: Biểu diễn histogram khiến cấu trúc không gian của hình ảnh bị mất đi 12 - Nhạy cảm với nhiễu và không trích xuất được đặc trưng kích thước lớn: Toán tử LBP có một nhược điểm trong nhận dạng khuôn mặt là nó xét vùng không gian nhỏ, do đó phép so sánh bit được thực hiện giữa hai giá trị pixel đơn lẻ bị ảnh hưởng nhiều bởi nhiễu. Hơn nữa, các đặc trưng được tính toán trong vùng lân cận 3×3 cục bộ không thể nắm bắt cấu trúc tỷ lệ lớn hơn (cấu trúc vĩ mô) có thể là đặc điểm nổi trội của khuôn mặt.



6. GLCM:

- Là một phương pháp thống kê để kiểm tra kết cấu xem xét mối quan hệ không gian của các pixel

- Các hàm GLCM mô tả kết cấu của hình ảnh bằng cách tính toán tần suất xuất hiện của các cặp pixel với các giá trị cụ thể và trong một mối quan hệ không gian cụ thể

trong một hình ảnh, tạo GLCM, sau đó trích xuất các biện pháp thống kê từ ma trận này.

- Cách thức tính:

- + Chọn vùng quan tâm trong ảnh để tính toán ma trận GLCM.
- + Xác định các thông số như kích thước cửa sổ, bước nhảy và số cấp độ để tính toán ma trận GLCM.
- + Tính toán ma trận GLCM cho mỗi hướng bằng cách đếm số lần xuất hiện của các cặp giá trị cường độ xung quanh nhau.
- + Chuẩn hóa ma trận GLCM để đảm bảo tổng các giá trị trong ma trận bằng 1.
- + Tính toán các đặc trưng hình ảnh từ ma trận GLCM, bao gồm độ đồng nhất, độ đa dạng, độ lệch chuẩn, độ phân tán và độ liên kết.
- + Lựa chọn các đặc trưng hình ảnh để sử dụng trong các ứng dụng phân loại và nhận dạng hình ảnh.

- Ưu điểm:

- + Cung cấp thông tin về mức độ tương quan giữa các pixel trong ảnh
 - + Đơn giản, dễ sử dụng
- không yêu cầu định dạng ảnh cụ thể

- Nhược điểm:

- + Nhạy cảm với nhiễu
- + Phụ thuộc vào kích thước cửa sổ
- + Chỉ áp dụng được cho ảnh xám
- + Không thể xử lý các vùng không đồng nhất

II, Các kỹ thuật phân loại hình ảnh

Giới thiệu

- Có nhiều thuật toán khác nhau được ứng dụng trong việc phân loại hình ảnh. Các thuật toán này được chia thành hai nhóm chính là Học có giám sát (supervised learning) và Học không giám sát (unsupervised learning).

2.1 Phân loại có giám sát

- Trong học máy có giám sát, thuật toán được huấn luyện trên một tập hình ảnh đã được dán nhãn. Từ dữ liệu mẫu này, thuật toán có thể trích xuất thông tin, phục vụ phân loại ngay cả những hình ảnh chưa từng nhìn thấy trước đó.

- Nó được gọi là việc học có giám sát bởi vì quá trình của thuật toán học từ tập dữ liệu đầu vào có thể được coi là một “giáo viên” giám sát quá trình học tập. Chúng ta biết câu trả lời đúng, thuật toán sẽ lặp đi lặp lại làm cho việc dự đoán về dữ liệu đầu vào liên tục được “giáo viên” hoàn thiện. Việc học dừng lại khi thuật toán đạt được mức hiệu suất ở mức chấp nhận được.

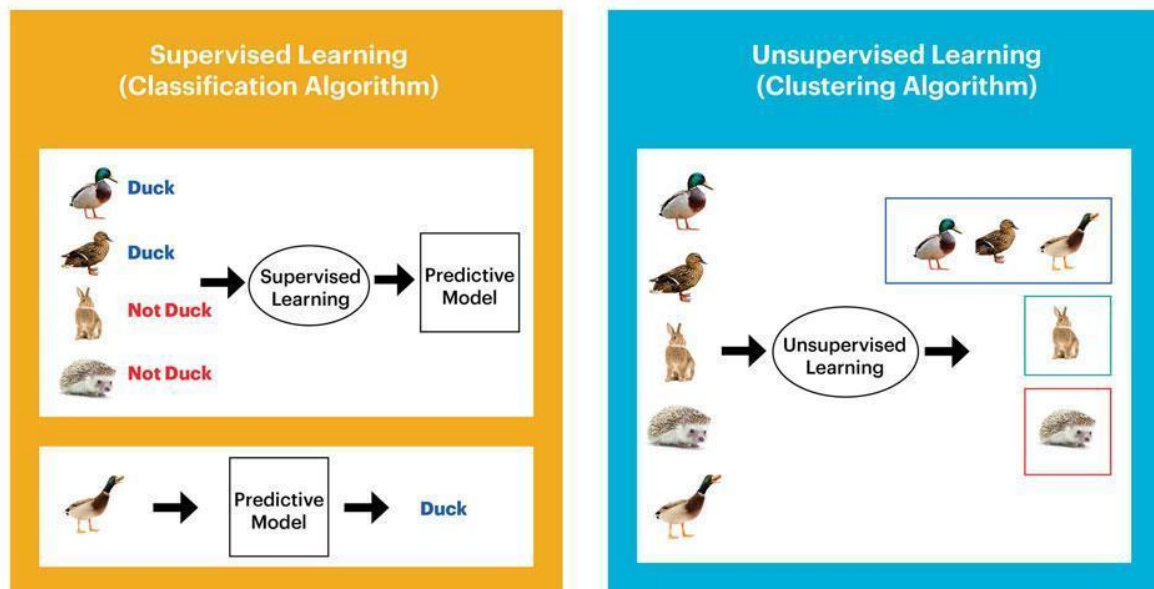
- Việc học tập có giám sát có thể được nhóm lại thành các vấn đề về phân loại và hồi quy.

+ Phân loại (Classification): Việc phân loại diễn ra khi biến đầu ra là một thể loại nào đó, chẳng hạn như “đỏ” hoặc “xanh” hoặc “bệnh” và “không có bệnh”.

+ Hồi quy (Regression): Việc hồi quy xảy ra là khi biến đầu ra là một giá trị thực, chẳng hạn như “đô la” hay “trọng lượng”.

2.2 Phân loại không giám sát

- Trong học máy không giám sát, thuật toán chỉ sử dụng dữ liệu thô để đào tạo. Các nhãn phân loại thường không xuất hiện trong kiểu học này và mô hình học bằng cách nhận dạng các mẫu trong tập dữ liệu huấn luyện.
- Mục tiêu của việc học không giám sát là để mô hình hóa cấu trúc nền tảng hoặc sự phân bố trong dữ liệu để hiểu rõ hơn về nó.
- Đây được gọi là học tập không giám sát vì không giống như việc học có giám sát ở trên, không có câu trả lời đúng và không có vị “giáo viên” nào cả. Các thuật toán được tạo ra chỉ để khám phá và thể hiện các cấu trúc hữu ích bên trong dữ liệu.
- Các vấn đề học tập không giám sát có thể được phân ra thành hai việc chia nhóm và kết hợp.
 - + Chia nhóm: Vấn đề về chia nhóm là nơi bạn muốn khám phá các nhóm vốn có bên trong dữ liệu, chẳng hạn như phân nhóm khách hàng theo hành vi mua hàng.
 - + Kết hợp: Vấn đề về học tập quy tắc kết hợp là nơi bạn muốn khám phá các quy tắc mô tả dữ liệu của bạn, chẳng hạn như những người mua X cũng có khuynh hướng mua Y.



1. K-nearest neighbors (KNN)

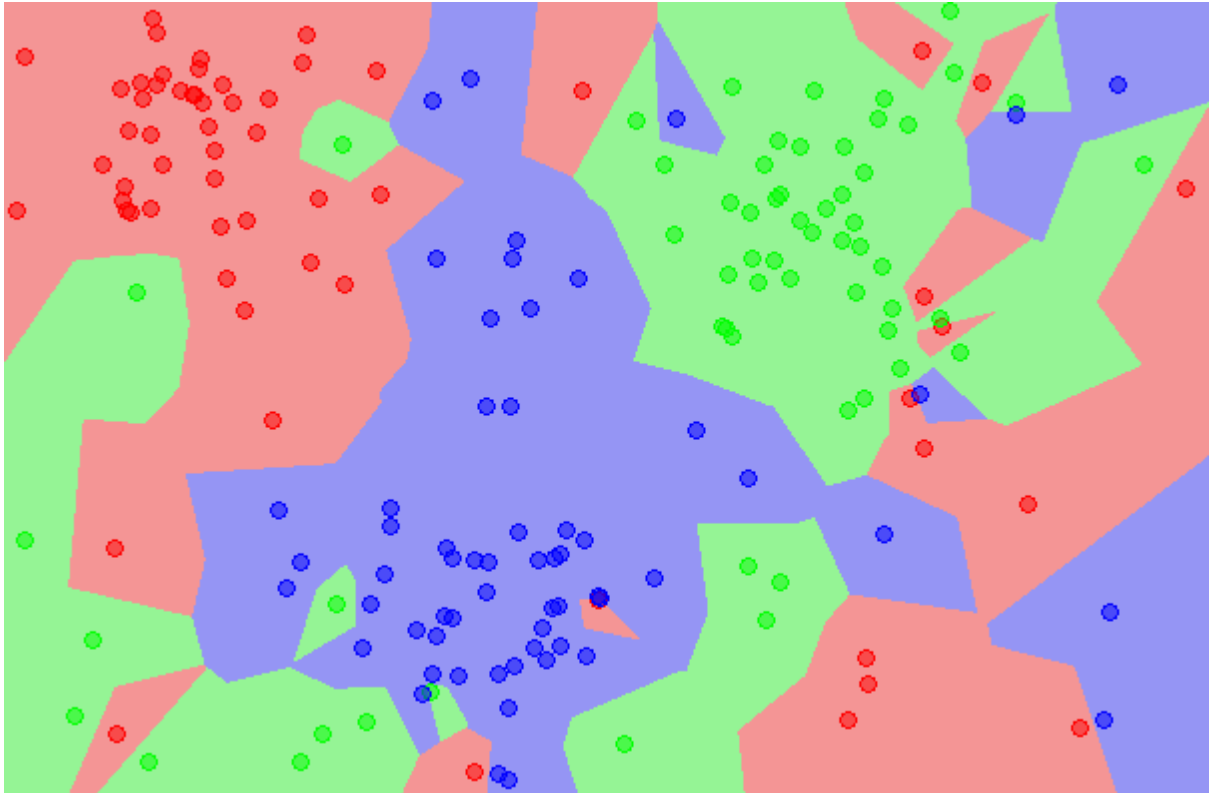
- KNN (K-Nearest Neighbors) là một trong những thuật toán học có giám sát đơn giản nhất được sử dụng nhiều trong khai phá dữ liệu và học máy.
 - Khi training, thuật toán này không học một điều gì từ dữ liệu training (lazy learning), mọi tính toán được thực hiện khi nó cần dự đoán kết quả của dữ liệu mới.
- K-nearest neighbor có thể áp dụng được vào cả hai loại của bài toán Supervised learning là Classification và Regression.

- Ưu điểm:

- + Độ phức tạp tính toán của quá trình training là bằng 0
- + Việc dự đoán kết quả của dữ liệu mới rất đơn giản
- + Không cần giả sử về phân phối của các lớp

- Nhược điểm:

- + KNN rất nhạy cảm với nhiễu khi K nhỏ
- + KNN là một thuật toán mà mọi tính toán đều nằm ở khâu test, trong đó việc tính khoảng cách tới từng điểm dữ liệu trong training set sẽ tốn nhiều thời gian. Khi K càng lớn thì độ phức tạp tăng lên
- + Lưu toàn bộ dữ liệu trong bộ nhớ cũng ảnh hưởng tới hiệu năng của KNN



2. Support vector machine (SVM)

- “Support Vector Machine” (SVM) là một thuật toán học máy có giám sát, có thể được sử dụng cho cả thử thách classification hoặc regression.
- Support Vector Machine là bài toán đi tìm mặt phân cách sao cho margin tìm được là lớn nhất, đồng nghĩa với việc các điểm dữ liệu an toàn nhất so với mặt phân cách.
- Bài toán tối ưu trong SVM là một bài toán lồi với hàm mục tiêu là strictly convex, nghiệm của bài toán này là duy nhất. Hơn nữa, bài toán tối ưu đó là một Quadratic Programming (QP).

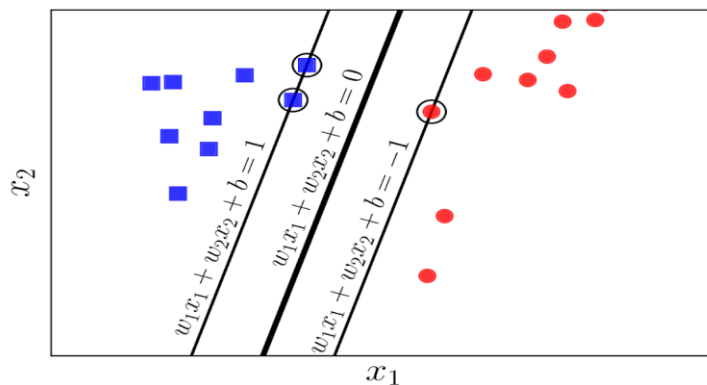
$$(\mathbf{w}, b) = \arg \min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|_2^2$$

$$\text{subject to: } 1 - y_n(\mathbf{w}^T \mathbf{x}_n + b) \leq 0, \forall n = 1, 2, \dots, N$$

- SVM xác định class cho điểm dữ liệu mới:

$$\text{class}(\mathbf{x}) = \text{sgn}(\mathbf{w}^T \mathbf{x} + b)$$

(Trong đó hàm sgn là hàm xác định dấu, nhận giá trị 1 nếu đối số là không âm và -1 nếu ngược lại.)



- Ưu điểm:

- + Xử lý trên không gian số chiều cao
- + Tiết kiệm bộ nhớ: Do chỉ có một tập hợp con của các điểm được sử dụng trong quá trình huấn luyện và ra quyết định thực tế cho các điểm dữ liệu mới nên chỉ có những điểm cần thiết mới được lưu trữ trong bộ nhớ khi ra quyết định
- + Tính linh hoạt: phân lớp thường là phi tuyến tính. Khả năng áp dụng Kernel mới cho phép linh động giữa các phương pháp tuyến tính và phi tuyến tính từ đó khiến cho hiệu suất phân loại lớn hơn.

- Nhược điểm:

- + Bài toán số chiều cao: Trong trường hợp số lượng thuộc tính (p) của tập dữ liệu lớn hơn rất nhiều so với số lượng dữ liệu (n) thì SVM cho kết quả khá tồi
- + Chưa thể hiện rõ tính xác suất: Việc phân lớp của SVM chỉ là việc cố gắng tách các đối tượng vào hai lớp được phân tách bởi siêu phẳng SVM. Điều này chưa giải thích được xác suất xuất hiện của một thành viên trong một nhóm là như thế nào.

3. Naïve bayes classification (NBC)

- Định lý bayes

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

- Naive Bayes là một trong nhóm các thuật toán áp dụng định lý Bayes với một giả định khá ngây thơ - đúng nghĩa đen của từ Naive, rằng mọi features đầu vào đều độc lập với nhau

- Naive Bayes là bộ phân loại theo xác suất (probability classifier) nên ta có thể tính toán xác suất bằng cách sử dụng định lý Bayes.

- Gồm các bước:

- + Xác định các đặc trưng (features) của dữ liệu và các nhãn (labels) cần phân loại.
- + Tính toán xác suất tiên nghiệm cho các nhãn, tức là xác suất của mỗi nhãn trước khi xem xét các đặc trưng.
- + Tính toán xác suất đồng thời cho từng đặc trưng và nhãn, tức là xác suất của mỗi đặc trưng trong mỗi nhãn.
- + Sử dụng luật Bayes để tính toán xác suất hậu nghiệm cho mỗi nhãn dựa trên các đặc trưng của dữ liệu.

+ Chọn nhãn có xác suất hậu nghiệm cao nhất là nhãn được phân loại cho dữ liệu đó.

- Ưu điểm:

- + Tính toán nhanh và hiệu quả
- + Dễ triển khai
- + Hiệu quả với số lượng đặc trưng lớn
- + Dự báo tốt với dữ liệu thiếu

- Nhược điểm:

- + Giả định về tính độc lập giữa các đặc trưng thường không chính xác trong thực tế
- + Làm giảm độ chính xác của mô hình
- + Khả năng dự đoán thấp khi các đặc trưng có mối tương quan cao

TRAINING										TEST
class = B	hanoi	pho	chaolong	buncha	onai	banhgio	saigon	hutu	banhbo	
d1: \mathbf{x}_1	2	1	1	0	0	0	0	0	0	$d = V = 9$ $\Rightarrow N_B = 11$ $(20 = N_B + V)$
d2: \mathbf{x}_2	1	1	0	1	1	0	0	0	0	
d3: \mathbf{x}_3	0	1	0	0	1	1	0	0	0	
Total	3	3	1	1	2	1	0	0	0	
$\Rightarrow \hat{\lambda}_B$	4/20	4/20	2/20	2/20	3/20	2/20	1/20	1/20	1/20	
class = N										
d4: \mathbf{x}_4	0	1	0	0	0	0	1	1	1	$\Rightarrow N_N = 4$ $(13 = N_N + V)$
$\Rightarrow \hat{\lambda}_N$	1/13	2/13	1/13	1/13	1/13	1/13	2/13	2/13	2/13	

d5: $\mathbf{x}_5 = [2, 0, 0, 1, 0, 0, 0, 1, 0]$

$p(B|d5) \propto p(B) \prod_{i=1}^d p(x_i|B)$
 $= \frac{3}{4} \left(\frac{4}{20}\right)^2 \frac{2}{20} \frac{1}{20} \approx 1.5 \times 10^{-4}$

$p(N|d5) \propto p(N) \prod_{i=1}^d p(x_i|N)$
 $= \frac{1}{4} \left(\frac{1}{13}\right)^2 \frac{1}{13} \frac{2}{13} \approx 1.75 \times 10^{-5}$

$\Rightarrow p(\mathbf{x}_5|B) > p(\mathbf{x}_5|N) \Rightarrow d5 \in \text{class}(B)$

4. Deep learning

- Hiểu theo một cách khác thì DL là một tập hợp các thuật toán mô phỏng lại cách thức hoạt động của bộ não của con người trong việc phân tích dữ liệu, nhằm tạo ra các models cái mà được sử dụng cho việc đưa ra quyết định dựa trên dữ liệu đầu vào. Về bản chất, DL bao gồm nhiều layers, mỗi layer có thể coi là một mạng Neural Network (NN).

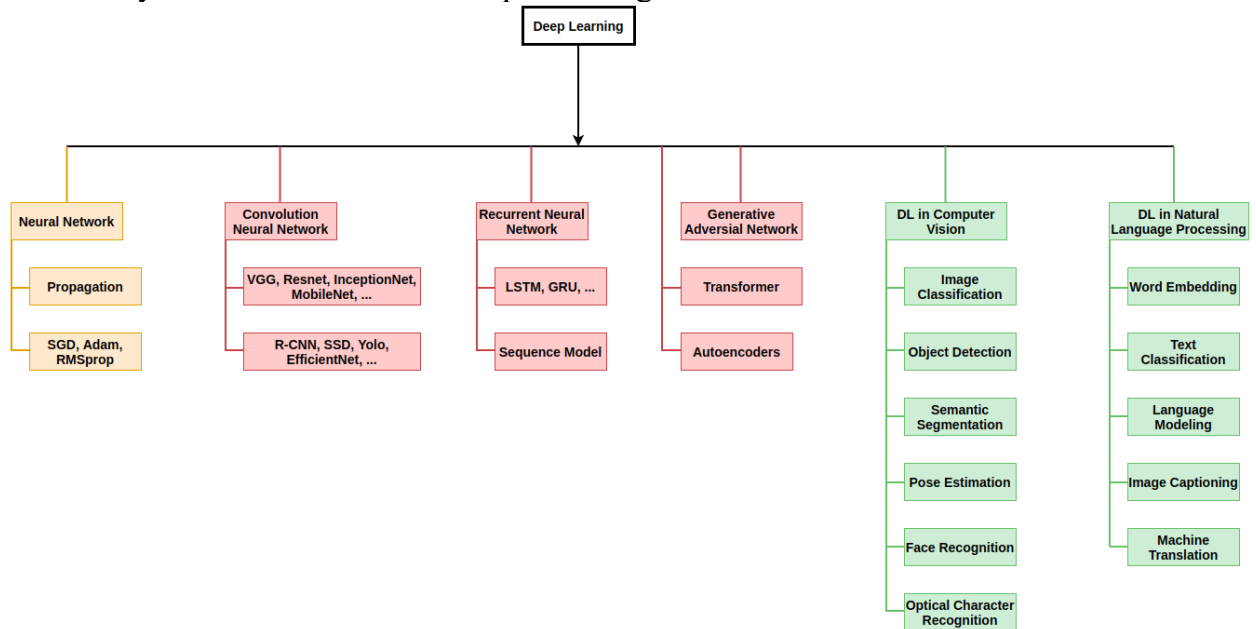
- Giống như não bộ của con người, NN có chứa các neurons. Mỗi neurons nhận các tín hiệu ở đầu vào, nhân chúng với các trọng số (weights), tổng hợp chúng lại, và sau cùng là áp dụng các hàm kích hoạt (thường là non-linear) lên chúng. Các neurons được sắp xếp thành các layers liên tiếp nhau (stack).

- Các kiến trúc mạng DL đều sử dụng thuật toán [Backpropagation](#) để tính toán và cập nhật các trọng số của nó thông qua quá trình huấn luyện. Về mặt ý tưởng, dữ liệu

được đưa vào mạng DL, sinh ra output, so sánh output với giá trị thực tế (sử dụng loss function) rồi điều chỉnh trọng số theo kết quả so sánh đó.

- Việc điều chỉnh trọng số là một quá trình tối ưu, thường sử dụng thuật toán [SGD](#). Ngoài SGD, một số thuật toán khác cũng hay được sử dụng. Đó là Adam, Radam, SRMprop.

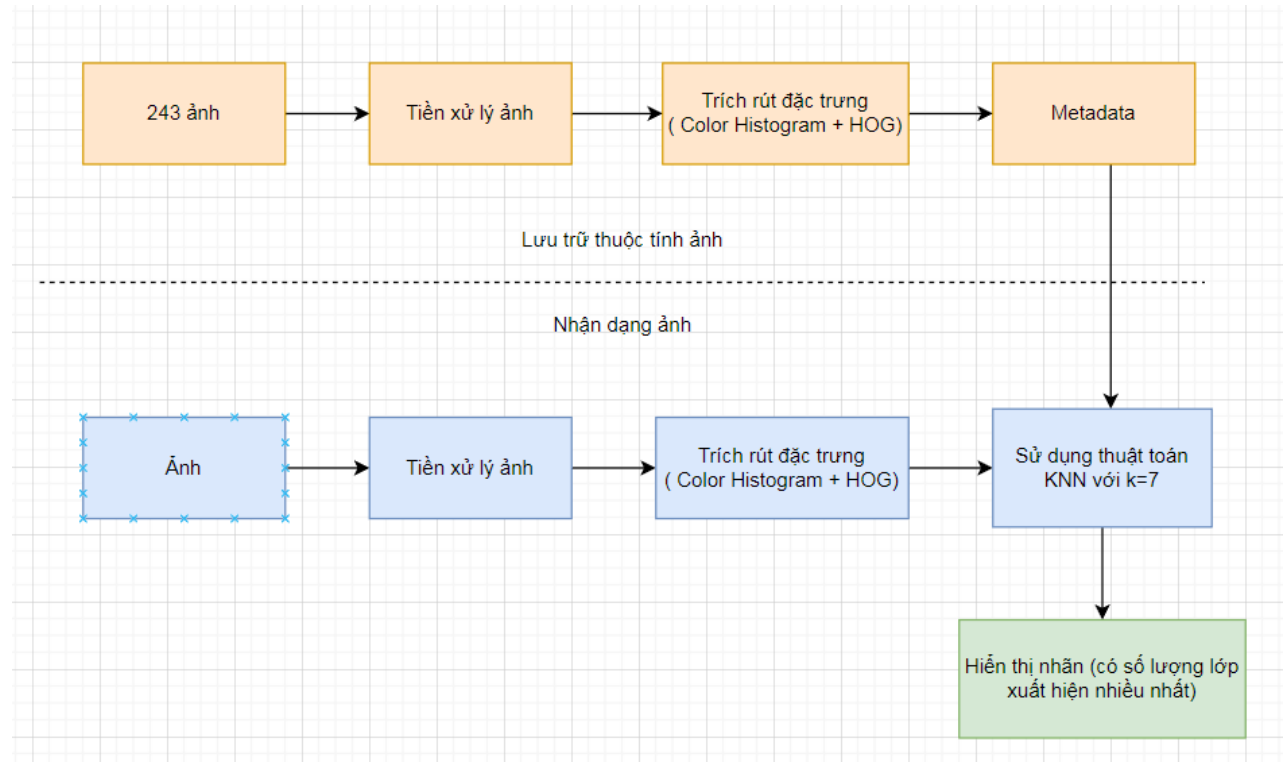
- Dưới đây là một số thuật toán Deep Learning:



PHẦN 3. HỆ THỐNG NHẬN DẠNG VÀ TÌM KIẾM ẢNH

I, Sơ đồ khối và quy trình thực hiện

1. Sơ đồ khối



2. Quy trình thực hiện

Sơ đồ khối bao gồm 9 bước chính chia thành 2 giai đoạn:

- Giai đoạn 1: Lưu trữ thuộc tính ảnh

- + Tiền xử lý bộ dữ liệu ảnh hoa 243 ảnh (chuyển từ ảnh đầu vào mô hình màu RGB sang mô hình màu HSV / ảnh xám / resize ảnh về kích cỡ 496x496 nếu cần thiết)

- + Phân tích và trích rút đặc trưng ảnh bằng Color Histogram (dùng HSV) và Histogram of Oriented Gradients (HOG)

- + Sau khi trích rút đặc trưng sẽ lưu lại đặc trưng vào file .npy

- Giai đoạn 2: Nhận dạng ảnh

- + Ảnh truy vấn là ảnh mới không có trong bộ dữ liệu đào tạo, bước đầu cũng được tiền xử lý (chuyển từ ảnh đầu vào mô hình màu RGB sang mô hình màu HSV / ảnh xám / resize ảnh về kích cỡ 496x496 nếu cần thiết)

- + Phân tích trích rút đặc trưng ảnh truy vấn bằng Color Histogram và HOG

- + So sánh khoảng cách với tất cả các ảnh trong Database

- + Hiển thị nhãn, ảnh dự đoán

II. Thuộc tính và các kỹ thuật trích rút

Theo khảo sát của nhóm, mắt người thường phân biệt hoa dựa vào màu sắc và hình dạng => Do đó nhóm sử dụng hướng trích rút đặc trưng là màu sắc (chọn phương pháp Color Histogram) và trích rút đặc trưng hình dạng (chọn phương pháp HOG)

1. Thuộc tính sử dụng về màu sắc

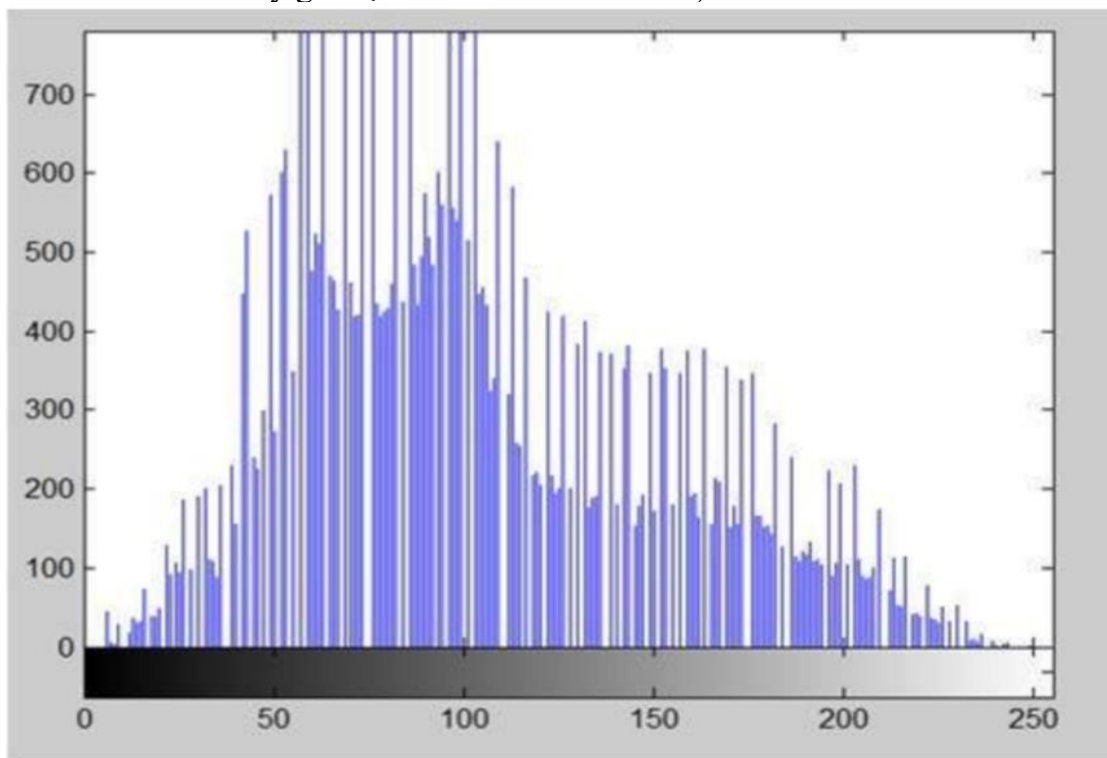
1.1. Kỹ thuật trích rút Color Histogram

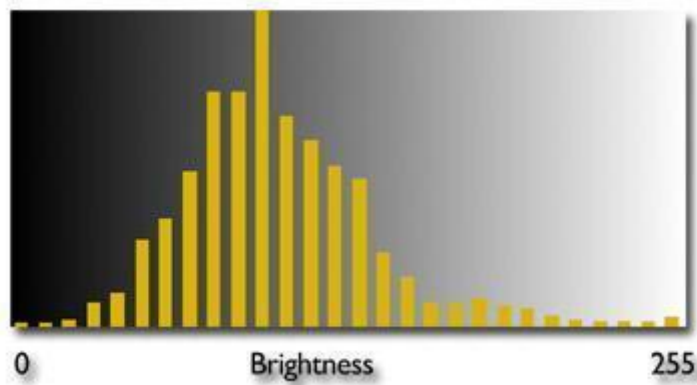
Histogram: là biểu đồ hình cột, mỗi cột biểu diễn tần suất xuất hiện của giá trị cường độ mức xám. Trong một biểu đồ màu, trục x biểu diễn cường độ mức xám và trục y hiển thị tần suất xuất hiện của cường độ.

+ Trục y: dọc biểu diễn số lượng điểm ảnh, các đỉnh càng cao thì càng có nhiều điểm ảnh ở khu vực đó và độ chi tiết càng nhiều.

+ Trục x: trục ngang x bắt đầu từ 0 và kết thúc tại 255. Tại $x = 0$ tối nhất và $x = 255$ sáng nhất. Như vậy càng nhiều điểm ảnh có giá trị x gần 0 thì ảnh tối ngược lại gần 255 thì ảnh sáng và nếu các điểm ảnh tập trung tại đường thẳng vuông góc với trục ngang x đi qua $x = 0$ hoặc 255 sẽ rất tối hoặc rất sáng dẫn đến khó quan sát.

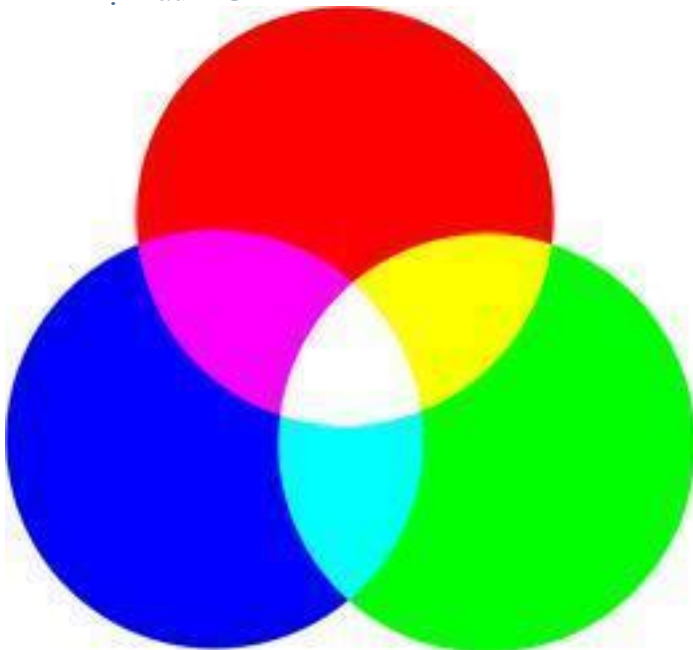
Histogram của một ảnh số biểu diễn sự phân bố số lượng điểm ảnh tương ứng với một giá trị màu có trong ảnh (giá trị đó có thể là một bộ giá trị RGB đối ảnh RGB, HSV đối với ảnh HSV hay giá trị mức xám đối ảnh xám).





1.2. Các hệ màu cơ bản

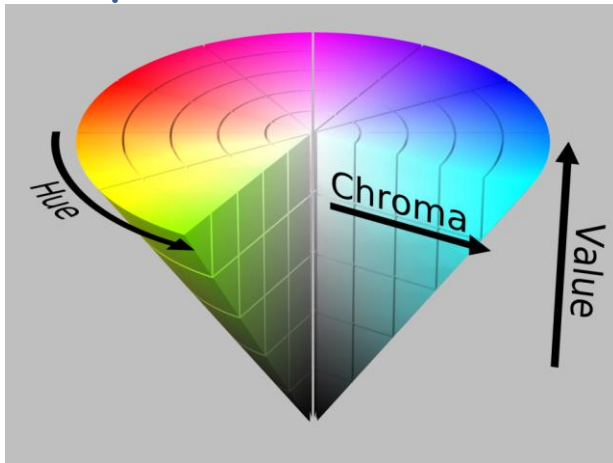
Hệ màu RGB



RGB là không gian màu phổ biến dùng trong máy tính, máy ảnh, điện thoại và nhiều thiết bị kỹ thuật số khác nhau. Không gian màu này khá gần với cách mắt người tổng hợp màu sắc. Nguyên lý cơ bản là sử dụng 3 màu sắc cơ bản R (red - đỏ), G (green - xanh lục) và B (blue - xanh lam) để biểu diễn tất cả các màu sắc.

Thông thường, trong mô hình 24 bit mỗi kênh màu sẽ sử dụng 8bit để biểu diễn, tức là giá trị R, G, B nằm trong khoảng 0 - 255. Bộ 3 số này biểu diễn cho từng điểm ảnh, mỗi số biểu diễn cho cường độ của một màu. Với mô hình màu 24bit thì số màu tối đa có thể tạo ra là $255 \times 255 \times 255 = 16581375$ màu.

Hệ màu HSV



Không gian màu HSV (còn gọi là HSB) là một cách tự nhiên hơn để mô tả màu sắc, dựa trên 3 số liệu:

+ Hue (Tông màu): Cho biết đây là màu gì? Là tổ hợp của 12 màu đậm nhạt khác nhau trên vòng tuần hoàn màu sắc, được biểu diễn dưới dạng một số từ 0 đến 360 độ.

Màu sắc	Góc(đơn vị: $^{\circ}$)
Đỏ	0-60
Vàng	60-120
Xanh lá	120-180
Cyan	180-240
Xanh da trời	240-300
Đỏ tươi	300-360

+ Saturation (Độ bão hòa): mô tả lượng màu xám trong một màu cụ thể, từ 0 đến 100 phần trăm. Giảm thành phần này về 0 sẽ tạo ra nhiều màu xám hơn và tạo hiệu ứng mờ dần.

+ Value (Độ sáng): Giá trị hoạt động cùng với độ bão hòa và mô tả độ sáng hoặc cường độ của màu, từ 0 đến 100 phần trăm, trong đó 0 là màu đen hoàn toàn và 100 là màu sáng nhất và hiển thị nhiều màu nhất.

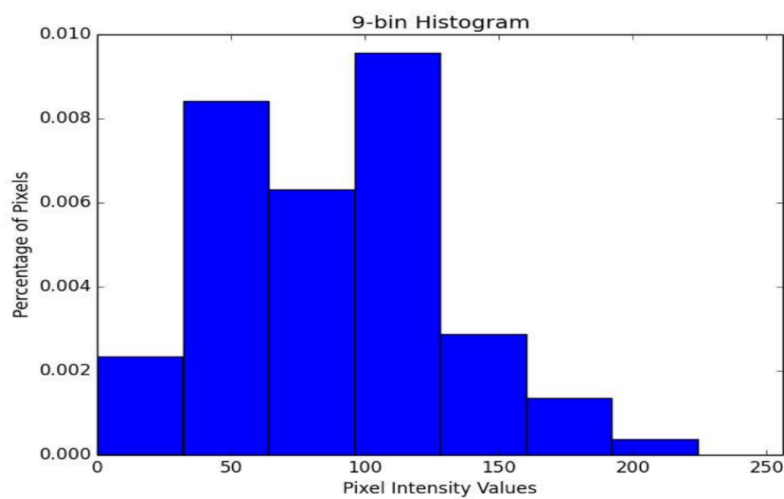
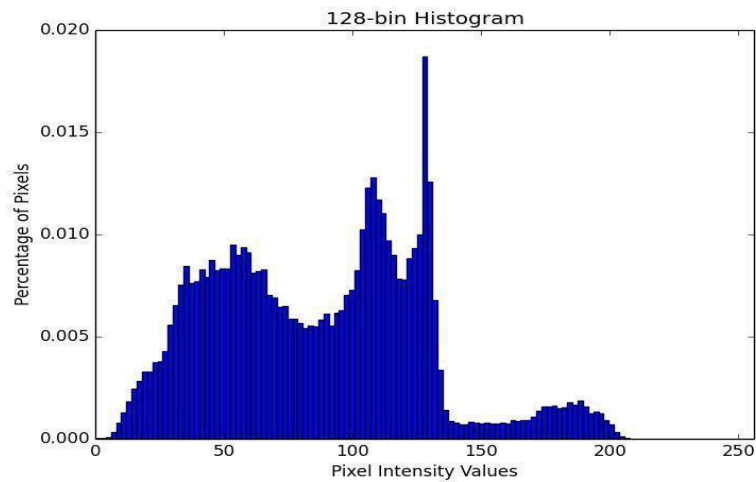
+ Thông thường ảnh được thể hiện dưới dạng RGB, mỗi điểm ảnh là sự tổ hợp từ 3 màu Red, Green, Blue tạo thành. Các giá trị như nhau của RGB có thể mô tả màu tương đối khác nhau trên các thiết bị khác nhau điều này sẽ làm ảnh hưởng đến kết quả bài toán nhận diện ảnh dựa trên màu sắc.

+ Do vậy thay vì sử dụng mô hình màu RGB, chúng em sử dụng mô hình màu HSV để tăng độ chính xác cho bài toán. Vậy nên, chuyển đổi ảnh RGB sang HSV.

1.3, Cách chia bin :

Điều quan trọng cần lưu ý là cách chọn số lượng bins. Nếu ta chọn quá ít bins, thì histogram sẽ có ít thành phần hơn và không thể phân biệt giữa các hình ảnh với các phân bố màu khác nhau đáng kể. Tương tự như vậy, nếu ta sử dụng quá nhiều bins, histogram sẽ có nhiều thành phần, dẫn đến tình trạng hình ảnh có nội dung rất giống

nhau nhưng lại có histogram khác biệt.



Ví dụ về cách chia bins: chia bins theo tỷ lệ (R, G, B) – (4, 4, 4)

=> Nếu chia dải thành 4 bins thì dải pixel thuộc về từng bin sẽ là :Bin 0 (0-63) ,Bin1 (64-127) , Bin2(128-191) , Bin3 (192-255)

Ảnh đầu vào: cỡ (256, 256, 3)



Ví dụ 1 pixel có giá trị là (20,20,20) thì nó sẽ thuộc 1. R-G-B Bin0-Bin0-Bin0 => giá trị biến đếm của 1. **R-G-B Bin0-Bin0-Bin0** tăng lên 1 đơn vị
 Kết quả sau khi chia bins theo tỷ lệ (R, G, B) – (4, 4, 4)

- | | |
|--------------------------------------|---------------------------------|
| 1. R-G-B Bin0-Bin0-Bin0 : 668 | 33. R-G-B Bin2-Bin0-Bin0: 0 |
| 2. R-G-B Bin0-Bin0-Bin1: 1304 | 34. R-G-B Bin2-Bin0-Bin1: 0 |
| 3. R-G-B Bin0-Bin0-Bin2: 3266 | 35. R-G-B Bin2-Bin0-Bin2: 0 |
| 4. R-G-B Bin0-Bin0-Bin3: 1915 | 36. R-G-B Bin2-Bin0-Bin3: 0 |
| 5. R-G-B Bin0-Bin1-Bin0: 990 | 37. R-G-B Bin2-Bin1-Bin0: 0 |
| 6. R-G-B Bin0-Bin1-Bin1: 1960 | 38. R-G-B Bin2-Bin1-Bin1: 365 |
| 7. R-G-B Bin0-Bin1-Bin2: 0 | 39. R-G-B Bin2-Bin1-Bin2: 1610 |
| 8. R-G-B Bin0-Bin1-Bin3: 1 | 40. R-G-B Bin2-Bin1-Bin3: 9491 |
| 9. R-G-B Bin0-Bin2-Bin0: 0 | 41. R-G-B Bin2-Bin2-Bin0: 0 |
| 10. R-G-B Bin0-Bin2-Bin1: 3 | 42. R-G-B Bin2-Bin2-Bin1: 879 |
| 11. R-G-B Bin0-Bin2-Bin2: 0 | 43. R-G-B Bin2-Bin2-Bin2: 12359 |
| 12. R-G-B Bin0-Bin2-Bin3: 0 | 44. R-G-B Bin2-Bin2-Bin3: 2028 |
| 13. R-G-B Bin0-Bin3-Bin0: 0 | 45. R-G-B Bin2-Bin3-Bin0: 0 |
| 14. R-G-B Bin0-Bin3-Bin1: 0 | 46. R-G-B Bin2-Bin3-Bin1: 0 |
| 15. R-G-B Bin0-Bin3-Bin2: 0 | 47. R-G-B Bin2-Bin3-Bin2: 5 |
| 16. R-G-B Bin0-Bin3-Bin3: 0 | 48. R-G-B Bin2-Bin3-Bin3: 2 |
| 17. R-G-B Bin1-Bin0-Bin0: 0 | 49. R-G-B Bin3-Bin0-Bin0: 0 |
| 18. R-G-B Bin1-Bin0-Bin1: 59 | 50. R-G-B Bin3-Bin0-Bin1: 0 |
| 19. R-G-B Bin1-Bin0-Bin2: 1619 | 51. R-G-B Bin3-Bin0-Bin2: 0 |
| 20. R-G-B Bin1-Bin0-Bin3: 3984 | 52. R-G-B Bin3-Bin0-Bin3: 0 |
| 21. R-G-B Bin1-Bin1-Bin0: 15 | 53. R-G-B Bin3-Bin1-Bin0: 0 |
| 22. R-G-B Bin1-Bin1-Bin1: 11407 | 54. R-G-B Bin3-Bin1-Bin1: 0 |
| 23. R-G-B Bin1-Bin1-Bin2: 1332 | 55. R-G-B Bin3-Bin1-Bin2: 0 |
| 24. R-G-B Bin1-Bin1-Bin3: 5011 | 56. R-G-B Bin3-Bin1-Bin3: 0 |
| 25. R-G-B Bin1-Bin2-Bin0: 0 | 57. R-G-B Bin3-Bin2-Bin0: 0 |
| 26. R-G-B Bin1-Bin2-Bin1: 2422 | 58. R-G-B Bin3-Bin2-Bin1: 0 |
| 27. R-G-B Bin1-Bin2-Bin2: 302 | 59. R-G-B Bin3-Bin2-Bin2: 150 |
| 28. R-G-B Bin1-Bin2-Bin3: 0 | 60. R-G-B Bin3-Bin2-Bin3: 1946 |
| 29. R-G-B Bin1-Bin3-Bin0: 0 | 61. R-G-B Bin3-Bin3-Bin0: 0 |
| 30. R-G-B Bin1-Bin3-Bin1: 0 | 62. R-G-B Bin3-Bin3-Bin1: 0 |
| 31. R-G-B Bin1-Bin3-Bin2: 0 | 63. R-G-B Bin3-Bin3-Bin2: 85 |
| 32. R-G-B Bin1-Bin3-Bin3: 0 | 64. R-G-B Bin3-Bin3-Bin3: 358 |

Thu được vector đặc trưng có độ dài là 64.

Tuy nhiên một vấn đề nữa xảy ra đó là với mỗi hình ảnh khác nhau thì sẽ có kích thước khác nhau thì sẽ có số lượng pixel khác nhau. Điều đó khiến sự bất bình đẳng khi so sánh hai biểu đồ tần suất. Một phương án giúp chuẩn hóa nó là chia biểu đồ tần suất cho số pixel

có trong ảnh. Ta được một histogram nhưng được tính toán dưới dạng % số lượng pixel trong ảnh. Điều đó thuận tiện hơn trong công việc so sánh hai biểu đồ histogram trong ảnh.

1.4, Hệ màu sử dụng trong bài toán và chọn tỷ lệ chia bins

Trong bài toán của nhóm thì nhóm sử dụng phương pháp tính toán histogram của hình ảnh HSV. Bởi vì ảnh hoa sẽ có duy nhất một bông hoa và các ảnh sẽ khác nhau về

góc chụp, độ sáng, màu sắc, môi trường vì thế thay vì sử dụng RGB ta sử dụng HSV thay vì chỉ có thông tin của màu sắc ta còn có thông tin về độ sáng, độ bão hòa. Chia bins theo tỷ lệ: (H,S,V) – (8,12,3) (những con số này sau khi chúng em thử nhiều trường hợp khác và chọn do nó cho ra kết quả tốt hơn các cách chia bins khác) Sử dụng bộ bins giảm thiểu được độ phức tạp tính toán, ngoài ra giảm thiểu độ nhiễu, giảm số chiều tính toán nhưng vẫn đem lại hiệu quả tốt. Ta chọn bộ bin (8,12,3) khi đó số chiều là: $8 \times 12 \times 3 = 288$

Tại sao lại sử dụng bộ bins này? (8,12,3) là trường hợp giảm được tối đa số giá trị của mỗi kênh nhưng vẫn đem lại độ hiệu quả mặt so sánh màu sắc của hai hình ảnh.

Vậy sau khi trích rút ta sẽ thu được vector đặc trưng có số phần tử là $8 \times 12 \times 3 = 288$.

1.5, Chuyển ảnh từ RGB sang HSV

Ban đầu ảnh thuộc hệ RGB. Nhóm thực hiện chuyển đổi tuyến tính từ RGB sang HSV theo công thức dưới đây:

$$r' = \frac{r}{255}, g' = \frac{g}{255}, b' = \frac{b}{255}$$

$$C_{max} = \max(r', g', b') \quad C_{min} = \min(r', g', b')$$

$$\Delta = C_{max} - C_{min}$$

$$H = \begin{cases} \frac{g' - b'}{\Delta} & \text{when } C_{max} = r' \\ 2 + \frac{(b' - r')}{\Delta} & \text{when } C_{max} = g' \\ 4 + \frac{(r' - g')}{\Delta} & \text{when } C_{max} = b' \end{cases}$$

$$H = \left(\frac{H}{6}\right) \% 1.0$$

$$s = \begin{cases} 0 & \text{when } \Delta = 0 \\ \frac{\Delta}{v} & \text{when } \Delta \neq 0 \end{cases}$$

$$v = C_{max}$$

$$\Rightarrow H = H * 180$$

$$\Rightarrow s = s * 255$$

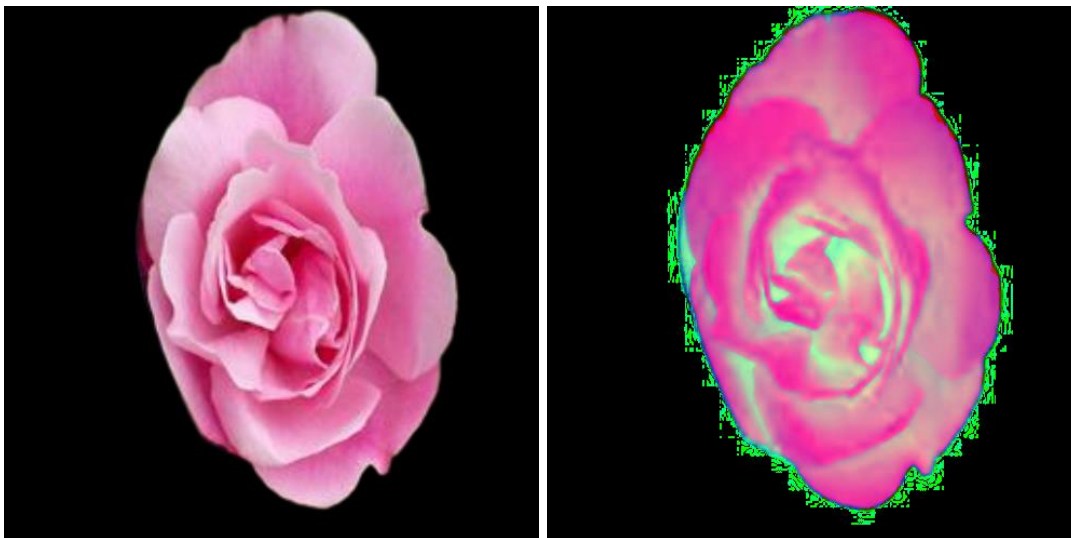
$$\Rightarrow v = v * 255$$

Áp dụng với từng pixel ta sẽ được kết quả ảnh HSV từ ảnh đầu vào RGB như bên dưới:

Ví dụ 1:



Ví dụ 2 :



2. Thuộc tính sử dụng về hình dạng

2.1, Kỹ thuật trích rút đặc trưng HOG

Nhóm sẽ lựa chọn trích rút bộ đặc trưng được gọi là HOG features, nó dựa trên sự biến đổi về màu sắc của hình ảnh để thu được hình dạng của vật thể.

- Cơ sở lý thuyết về kỹ thuật Histogram of Oriented Gradients (HOG): Lý thuyết của HOG dựa trên việc tính toán Gradient của sự thay đổi độ sáng màu sắc. Hình dạng của một vật thể cục bộ được mô tả thông qua hai ma trận gọi là ma trận độ lớn Gradient (Gradient magnitude) và ma trận phương Gradient (Gradient direction). Để tính toán được hai ma trận này, hình ảnh sẽ được chia thành một lưới ô vuông (cell) và trên đó sẽ xác định nhiều vùng ảnh cục bộ.

- Ưu điểm của thuật toán là:

+ Bộ mô tả HOG có một vài lợi thế chính so với các bộ mô tả khác. Vì nó hoạt

động trên các ô cục bộ, nó bất biến đối với các phép biến đổi hình học, thay đổi độ sáng.

+ Bộ mô tả tính năng cố gắng nắm bắt thông tin quan trọng trong hình ảnh và không mấy quan tâm thông tin không quá quan trọng của ảnh. Sau đó, có thể sử dụng thông tin hữu ích từ bộ mô tả tính năng để nhận dạng hình ảnh và phát hiện đối tượng.

2.2, Các bước xây dựng

Có 5 bước cơ bản để xây dựng một vector HOG cho hình ảnh, bao gồm:

- + Tiền xử lý
- + Tính gradient
- + Tính vector đặc trưng cho từng ô (cells)
- + Chuẩn hóa khối (blocks)
- + Tính toán vector HOG

2.2.1. Tiền xử lý

- Đổi về ảnh xám: với công thức $\text{pixel_gray} = 0.299 * R + 0.587 * G + 0.114 * B$
- Resize kích thước tất cả các hình ảnh trong tập dữ liệu về một kích thước chung (Nếu cần). (496*496)
- chọn các tham số
 - + Kích thước ô: 8x8 pixel
 - + Kích thước khối (vùng cục bộ): 2x2 ô (16*16 pixel)

Ví dụ : Ảnh gốc cỡ 496*496*3



Sau khi chuyển về ảnh xám



2.2.2. Tính Gradient

- Phương pháp sẽ lọc ma trận cường độ ảnh với các bộ lọc Sobel mask.
- Để tính bộ lọc Sobel, thực hiện tính tích chập của mặt nạ với hình ảnh ban đầu. Ký hiệu ảnh I là ma trận ảnh gốc và G_x , G_y là hai ma trận ảnh mà mỗi điểm trên nó lần lượt là đạo hàm theo trục x và trục y của ảnh. Ký hiệu $*$ là phép nhân tích chập giữa bộ lọc bên trái và đầu vào bên phải.
- Đây là bước đầu tiên, được thực hiện bằng hai phép nhân chập ảnh gốc với 2 chiều, tương ứng với các toán tử lấy đạo hàm theo hai hướng O_x và O_y . Trong đó, 2 hướng tương ứng đó là:
 $D_x = [-1 \ 0 \ 1]$ và $D_y = [-1 \ 0 \ 1]^T = [-1 \ 0 \ 1]$
- Nếu có một ảnh input là I , ta sẽ có 2 ảnh đạo hàm riêng theo 2 hướng đó, theo công thức:
 $G_x = I * D_x$ và $G_y = I * D_y$
- Giá trị độ lớn gradient và phương gradient có thể được tạo ra từ 2 đạo hàm G_x và G_y theo công thức bên dưới:
 - + Độ lớn gradient:

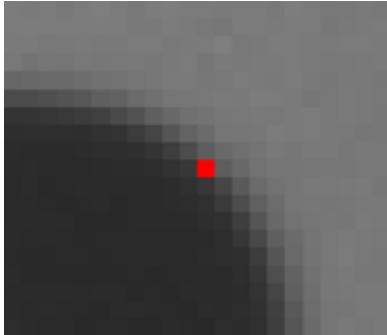
$$G = \sqrt{G_x^2 + G_y^2}$$

- + Phương gradient: xác định phương thay đổi cường độ màu sắc hay chính là phương đồ bóng của hình ảnh.

$$\theta = \arctan\left(\frac{G_y}{G_x}\right)$$

- Vậy mỗi bức ảnh giờ đây đã được lấy các đặc trưng biểu diễn thông qua 2 thông số liên quan đến mức độ thay đổi cường độ màu sắc (ma trận độ lớn Gradient) và hướng thay đổi cường độ màu sắc (ma trận phương Gradient).

- **Ví dụ :** Chúng ta sẽ áp dụng các công thức trên để tính được gradient của điểm ảnh này:



		93	
56			94
		55	

$$G_x = I * D_x = [56 \ x \ 94] * [-1 \ 0 \ 1] = [38]$$

$$G_y = I * D_y = [93 \ y \ 55] * [-1 \ 0 \ 1] = [-38]$$

$$G = \sqrt{G_x^2 + G_y^2} = \sqrt{38^2 + (-38)^2} \approx 53.74$$

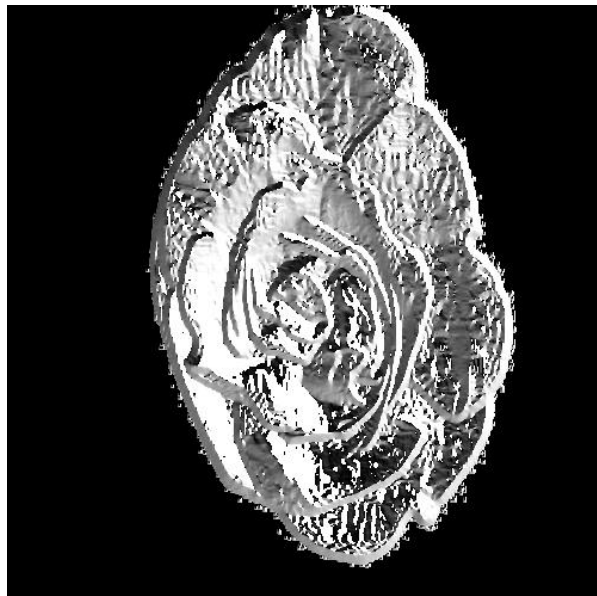
$$\theta = \operatorname{argtan} \frac{G_y}{G_x} = \operatorname{argtan} \frac{-38}{38} = 45 \text{ deg.}$$

Sau bước này, ta thu được 2 thông số cho mỗi điểm ảnh là hướng và cường độ. Như vậy ta sẽ có (ma trận độ lớn Gradient) và (ma trận phương Gradient).

G_x, g_y



G , theta



2.2.3 Tính vector đặc trưng cho từng ô (cells):

+ Đặc trưng của mỗi bức ảnh được biểu diễn thông qua 2 thông số đó là mức độ thay đổi cường độ màu sắc (ma trận gradient magnitude) và hướng thay đổi cường độ màu sắc (ma trận gradient direction). Do đó cần tạo ra được một bộ mô tả (feature descriptor) sao cho biến đổi bức ảnh thành một véc tơ mà thể hiện được cả 2 thông tin này. Và một biểu đồ Histogram thống kê độ lớn Gradient sẽ được tính toán trên mỗi ô cục bộ.

Véc tơ histogram sẽ được tạo ra như sau:

- **Bước 1:** Chia các bins tương ứng của phương gradient:

+ Chẳng hạn chia thành 9 bins. Độ lớn của phương gradient sẽ nằm trong khoảng $[0, 180]$ nên mỗi bins sẽ có độ dài là 20.

- **Bước 2:** Ánh xạ độ lớn gradient vào các bins tương ứng của phương gradient.

+ Mỗi một phương gradient sẽ ghép cặp với một độ lớn gradient ở cùng vị trí tọa độ. Khi biết được phương gradient thuộc bins nào trong véc tơ bins, ta sẽ điền vào giá trị giá trị của độ lớn gradient vào chính bin đó.

+ Đầu mút là các giá trị chia hết cho độ rộng của một bin (chẳng hạn 0, 20, 40,... là những đầu mút bin). Trong trường hợp độ lớn phương gradients không rơi vào các đầu mút, ta sẽ sử dụng linear interpolation (nội suy tuyến tính) để phân chia độ lớn gradient về 2 bins liền kề mà giá trị phương gradient rơi vào.

+ Cụ thể: giá trị phương gradient bằng x ghép cặp với độ lớn gradient bằng y , $x \in [x_0, x_1]$ tức là phương gradients rơi vào khoảng giữa bin thứ $(i-1)$ và bin thứ i . Khi đó tại 2 bins $(i-1)$ và i được điền vào giá trị cường độ theo công thức sau:

$$x_{i-1} = \frac{(x_1 - x)}{x_1 - x_0} * y$$

+ Giá trị tại bins $L-1$:

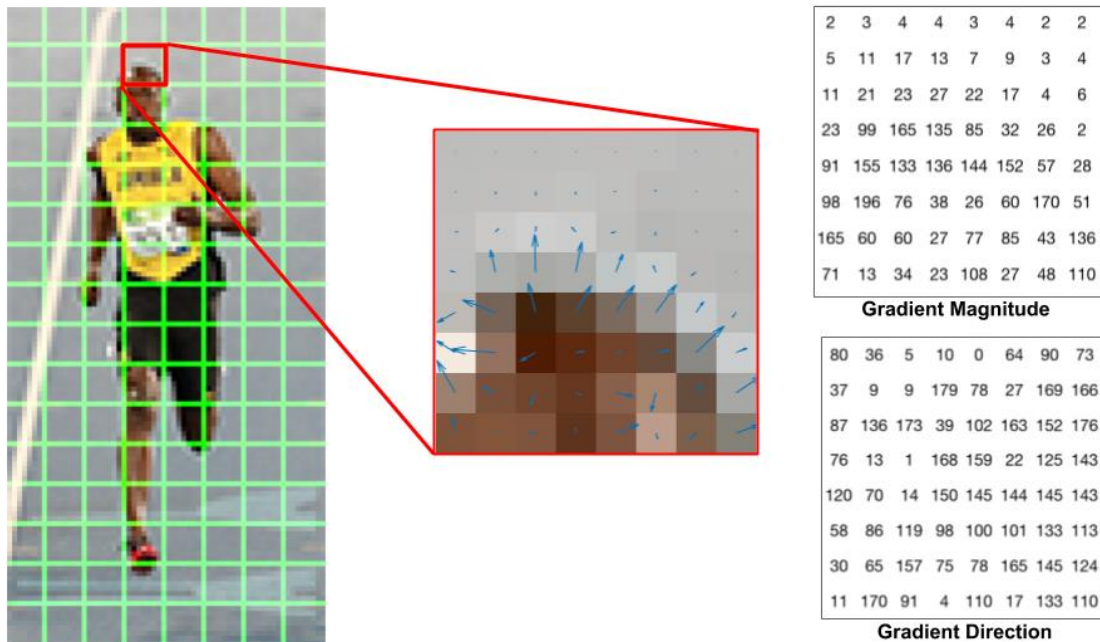
$$x_l = \frac{(x - x_0)}{x_1 - x_0} * y$$

+ Giá trị tại bins L:

- Bước 3: Tính tổng tất cả các độ lớn gradient thuộc cùng 1 bins của véc tơ bins ta thu được biểu đồ Histogram of Gradients.

VÍ DỤ :

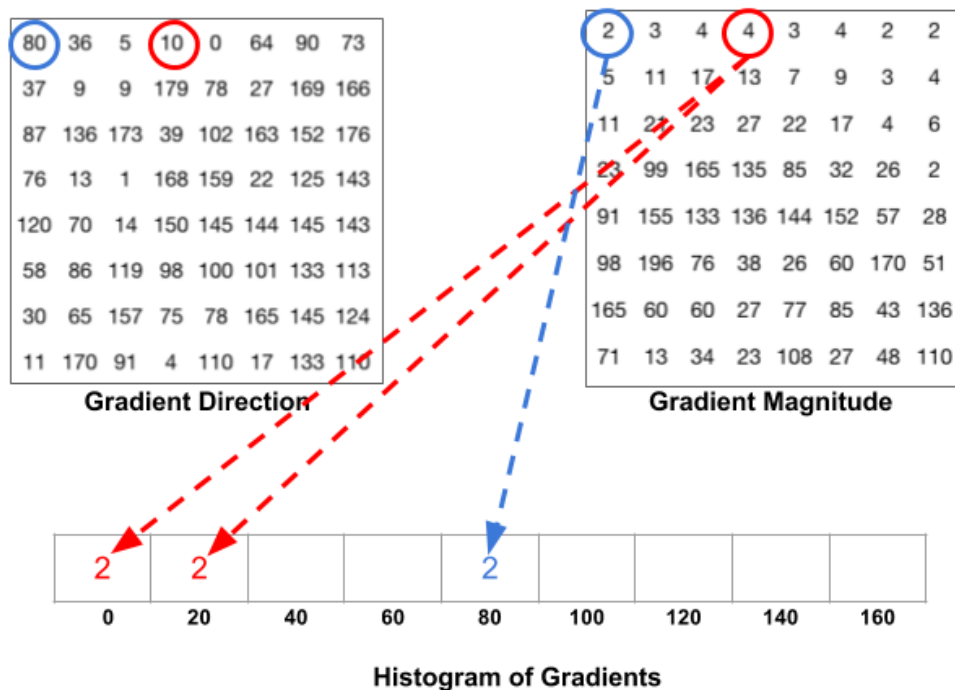
+ Xét trên một cell có kích thước thường là 8x8 pixels (giá trị này có thể thay đổi tùy vào cách chọn của người dùng). Trên mỗi một ô cục bộ 8x8 pixels cần tính ra 2 tham số đó là độ lớn gradient (gradient magnitude) và phương gradient (gradient direction). Như vậy tổng cộng $8 \times 8 \times 2 = 128$ giá trị cần tính bao gồm 64 giá trị độ lớn Gradient và 64 giá trị hướng Gradient (được 2 ma trận Ma trận Gradient Magnitude và Gradient Direction).



Trực quan ma trận Gradient Magnitude và Gradient Direction trên ô cục bộ 8x8 pixels

- Tiếp theo, tiến hành tính toán đặc trưng HOG tại mỗi cell sử dụng không gian hướng 9 bin. Hướng gradient sẽ chạy trong khoảng 0 độ đến 180 độ, trung bình 20 độ mỗi bin.
- Mỗi một phương gradient sẽ ghép cặp với một độ lớn gradient ở cùng một vị trí tọa độ, khi biết được phương gradient thuộc bin nào trong véc tơ bins ta sẽ điền vào giá trị của độ lớn gradient vào chính bin đó.
- Chẳng hạn: xét tại vị trí (0,0) phương gradient là 80, độ lớn gradient là 2: giá trị 80 thuộc vào bins thứ 5 nên tại vị trí thứ 5 của Histogram of Gradient ta điền giá trị 2.
- Như hình dưới, giá trị Gradient là 4 và phương là 10 nhưng nằm trong khoảng 0 và

20 nên giá trị 4 được tách ra tại hai bins liên kề.

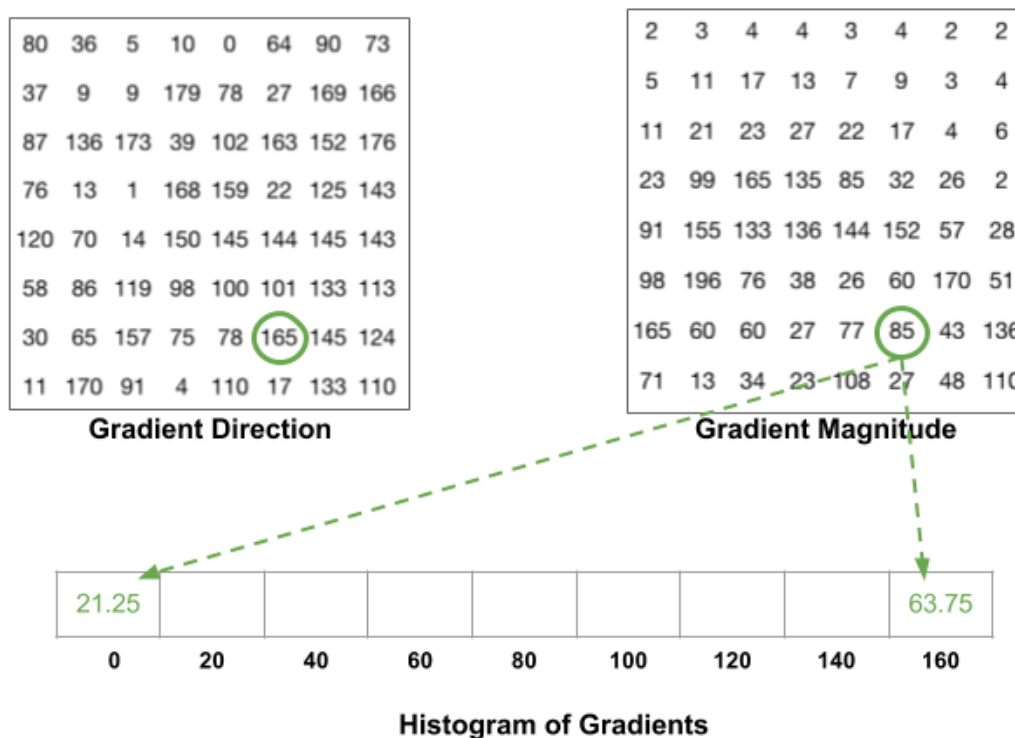


Trường hợp khác

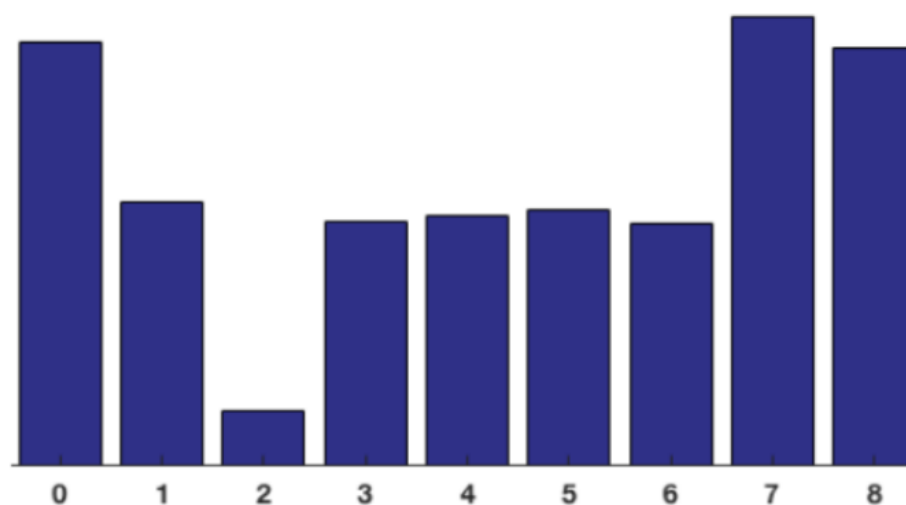
- Đầu mút là các giá trị chia hết cho độ rộng của bin (ví dụ 0, 20, 40,...) trong trường hợp độ lớn phương gradient không rơi vào các đầu mút,
- Với ví dụ trên xét điểm có phương gradient bằng 165 và độ lớn gradient là 85, có thể thấy $160 < 165 < 180$ ta phân chia giá trị về các bins 0 (thực chất là 180) và 160 theo các giá trị theo công thức interpolation.
- Cụ thể:

$$X_{160} = \frac{180-165}{180-160} * 85 = 63.75$$

$$X_0 = \frac{165-160}{180-160} * 85 = 21.25$$



- Tính tổng tất cả các độ lớn gradient thuộc cùng 1 bins của vector bins ta thu được biểu đồ Histogram of Gradients như sau .



2.2.4. Chuẩn hóa khối (blocks):

- Lý do chuẩn hóa là bởi vector histogram sẽ phụ thuộc vào cường độ các pixels nên có thể ảnh có cùng nội dung nhưng ảnh sáng ảnh tối khác nhau, giá trị histogram cũng khác nhau. Vì vậy cần chuẩn hóa vector histogram để cả hai bức ảnh có cùng một vector biểu diễn.

- Quá trình chuẩn hóa sẽ thực hiện trên một block kích thước 2x2 trên lưới ô vuông ban đầu. Ví dụ Ở đây, chúng ta sẽ kết hợp bốn ô 8x8 để tạo một khối 16x16. Như vậy chúng ta sẽ có 4 véc tơ histogram kích thước 1x9, tổng hợp các véc tơ sẽ thu được véc tơ histogram tổng hợp kích thước là 1x36 và sau đó chuẩn hóa theo norm chuẩn bậc 2

trên véc tơ này.

Để chuẩn hóa ma trận này, chúng ta sẽ chia từng giá trị này cho căn bậc hai của tổng bình phương các giá trị. Về mặt toán học, đối với một vector V đã cho:

$$V^- = \{a_1, a_2, a_3, \dots, a_{36}\}$$

Chúng ta tính toán căn bậc 2 của tổng bình phương:

$$|V^-| = \sqrt{a_1^2 + a_2^2 + a_3^2 + \dots + a_{36}^2}$$

Và chia tất cả các giá trị trong vector V với giá trị này:

$$\square \text{ Vector chuẩn hóa: } V^- = \left\{ \frac{a_1}{|V^-|}, \frac{a_2}{|V^-|}, \frac{a_3}{|V^-|}, \dots, \frac{a_{36}}{|V^-|} \right\}$$

Khi đó vector v đã được chuẩn hóa sao cho mỗi chiều có giá trị nằm trong [0;1]. Kết quả sẽ là một vector chuẩn hóa có kích thước 1x36.

-Có nhiều phương pháp có thể được dùng để chuẩn hóa khối. Gọi v là vector cần chuẩn hóa chứa tất cả các histogram của một khối. $\|v(k)\|$ là giá trị chuẩn hóa của v theo các chuẩn k=1, 3 và e là một hằng số nhỏ. Khi đó, các giá trị chuẩn hóa có thể tính bằng một trong những công thức sau:

$$\text{L2-norm: } f = \frac{v}{\sqrt{\|v_2\|^2 + e^2}}$$

$$\text{L1-norm: } f = \frac{v}{\|v_1\| + e}$$

$$\text{L1-sqrt: } f = \sqrt{\frac{v}{\|v_1\|^2 + e}}$$

2.2.5. Tính toán vector đặc trưng HOG:

Ghép các vector đặc trưng khối sẽ thu được vector đặc trưng HOG cho toàn bộ ảnh. Số chiều vector đặc trưng ảnh tính theo công thức :

$$size_{image} = n * size_{block}$$

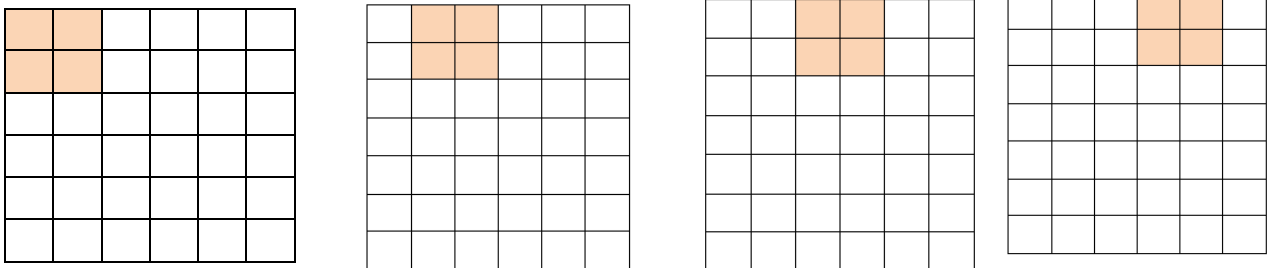
trong đó :

n là số khối của hình ảnh.

$size_{block}$ là số chiều của vector đặc trưng khối

Ví dụ :

- Xét 1 ảnh có kích thước 48×48 pixel tức là 6×6 ô (mỗi ô 8×8 pixel) quá trình tính toán HOG sẽ di chuyển 5 lần theo chiều ngang và 5 lần theo chiều dọc như vậy sẽ có 5×5 block được xét, mỗi block sau khi chuẩn hóa thu được vector 36 chiều. Vector HOG cuối cùng có kích thước $(5 \times 5) \times 36 = 900$ chiều.
- Ví dụ cho việc di chuyển các window biểu diễn việc lần lượt chuẩn hóa các block. Các block sau đề lên block trước 2 ô theo cả chiều ngang và chiều dọc. Mỗi ô vuông chính là một ô được định nghĩa ở trên có kích thước 8×8 pixel, mỗi block có kích thước $2 \times 2 = 4$ ô (có kích thước 16×16 pixel)



VD:

Ảnh 256×256 , kích thước cells = 8×8 , kích thước block = 16×16

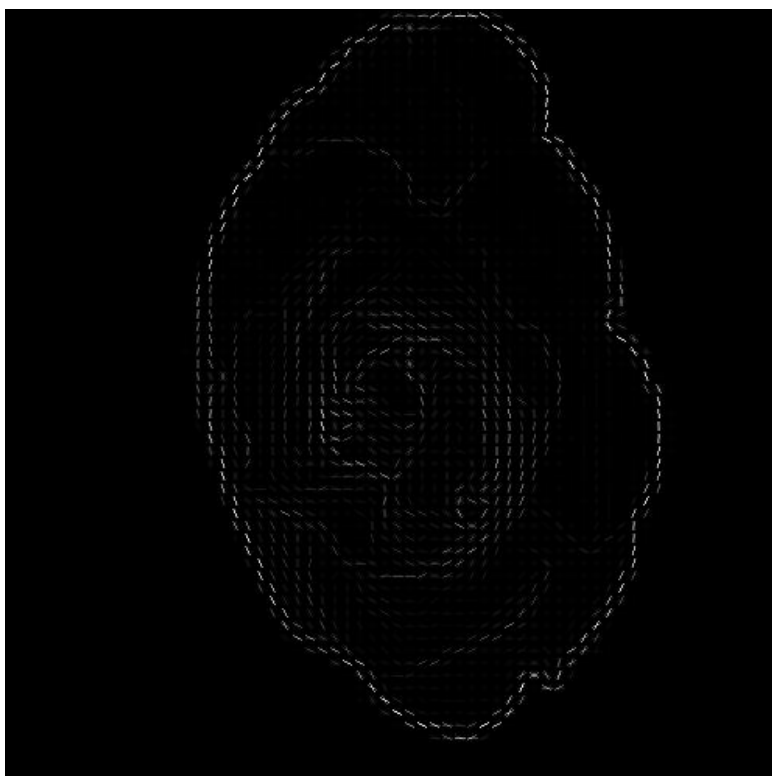
Quá trình tính toán HOG sẽ di chuyển $(256/8)-1 = 31$ lần theo chiều ngang và dọc Như vậy sẽ có tổng cộng $31 \times 31 = 961$ blocks mỗi patch tương ứng với 1 véc tơ histograms 36 chiều. Do đó cuối cùng véc tơ HOG sẽ có kích thước là $961 \times 36 = 34596$ chiều. Đây là một véc tơ kích thước tương đối lớn nên có thể mô phỏng được đặc trưng của ảnh khá tốt.

Ảnh 256×256 , kích thước cells = 16×16 , kích thước block = 64×64

Quá trình tính toán HOG sẽ di chuyển $(256/16)-1 = 13$ lần theo chiều ngang và dọc Như vậy sẽ có tổng cộng $13 \times 13 = 169$ blocks mỗi patch tương ứng với 1 véc tơ histograms 144 (16×9) chiều. Do đó cuối cùng véc tơ HOG sẽ có kích thước là $169 \times 144 = 24336$ chiều. Đây là một véc tơ kích thước tương đối lớn nên có thể mô phỏng được đặc trưng của ảnh khá tốt.

- Ví dụ 2 :Về bài toán

Hình ảnh của chúng em được chia thành block ô vuông kích thước 16×16 (mỗi ô 8×8 , ảnh 496×496). Quá trình tính toán HOG sẽ di chuyển $(496/8)-1 = 61$ lượt theo chiều ngang và 61 lượt theo chiều dọc. Như vậy sẽ có tổng cộng $61 \times 61 = 3721$ blocks, mỗi patch tương ứng với 1 véc tơ histograms 36 chiều. Do đó cuối cùng véc tơ HOG sẽ có kích thước là $3721 \times 36 = 133956$ chiều. Đây là một véc tơ kích thước tương đối lớn nên có thể mô phỏng được đặc trưng của ảnh khá tốt.



III. Lưu trữ, quản lý và tìm kiếm trong CSDL

1. Lưu trữ và quản lý:

- Dữ liệu bao gồm vector của 243 ảnh và nhãn của từng ảnh
- Tất cả các vector và nhãn được lưu trữ bên trong file .npy .
- Trong file .npy :Mỗi một hàng là 1 là mảng 2 chiều :

+ Chiều thứ nhất là 1 số từ 0-11 (đại diện cho nhãn). Nhãn ánh xạ: {'0': 'tree mallow', '1': 'black_eyed_susan', '2': 'balloon flower', '3': 'calendula', '4': 'daisy', '5': 'yellow iris', '6': 'anthurium', '7': 'sunflower', '8': 'pink primrose', '9': 'rose', '10': 'windflower', '11': 'common dandelion'}



















+Chiều thứ 2 chứa vector đặc trưng của ảnh(với X0-X287 là đặc trưng được trích xuất bằng Colour Histogram(dùng mô hình màu HSV) và từ X288-X13424 là đặc trưng được trích xuất bằng phương pháp HOG)

Hình dưới đây mô tả cách file .npy lưu trữ

	label	vector được trích xuất	
image 1	[9,	[X0,X1,X2,...,X286,X287 ,X288, ...,X 134243,X 134244]]	
image 2	[1,	[X0,X1,X2,...,X286,X287 ,X288, ...,X 134243,X 134244]]	
...	[2,	[...	...]]

image 243	[1,	[X0,X1,X2,...,X286,X287 ,X288, ,...,X 134243,X 134244]]
-----------	------	--

Ta thực hiện lưu trữ các đặc trưng theo tổ hợp các định dạng sau để dễ so sánh độ chính xác khi thực hiện trích rút đặc trưng riêng lẻ từng thuộc tính, và khi tổ hợp các thuộc tính lại:

 RGB.npy 	 nguyen cong ...	3 thg 6, 2023	998 KB	:
 HSV.npy 	 nguyen cong ...	3 thg 6, 2023	568 KB	:
 HOG.npy 	 nguyen cong ...	3 thg 6, 2023	251,4 MB	:
 flower_data.npy 	 nguyen cong ...	3 thg 6, 2023	173,2 MB	:
 concat_hog_rgb.npy 	 nguyen cong ...	3 thg 6, 2023	252,4 MB	:
 concat_hog_hsv.npy 	 nguyen cong ...	3 thg 6, 2023	252 MB	:

Giải thích nội dung của các file :

- RGB.npy: lưu trữ vector trích xuất đặc trưng ảnh theo phương pháp Color Histogram sử dụng mô hình màu RGB
- HSV.npy: lưu trữ vector trích xuất đặc trưng ảnh theo phương pháp Color Histogram sử dụng mô hình màu HSV
- HOG.npy: lưu trữ vector trích xuất đặc trưng ảnh theo phương pháp HOG
- Flower_data.npy : lưu trữ nội dung ảnh gốc sử dụng mô hình RGB
- Rgb_hog.npy : lưu trữ vector trích xuất đặc trưng ảnh theo tổ hợp 2 phương pháp color Histogram (sử dụng mô hình màu RGB chia tỷ lệ bin 8-8-8) và HOG
- Hsv_hog.npy : lưu trữ vector trích xuất đặc trưng ảnh theo tổ hợp 2 phương pháp color Histogram (sử dụng mô hình màu HSV chia tỷ lệ bin 8-12-3) và HOG

2. Tìm kiếm:

Sử dụng thuật toán KNN

-KNN (K-Nearest Neighbors) : là một trong những thuật toán học có giám sát đơn giản nhất được sử dụng nhiều trong khai phá dữ liệu và học máy. Ý tưởng của thuật toán này là nó không học một điều gì từ tập dữ liệu học (nên KNN được xếp vào loại lazy learning), mọi tính toán được thực hiện khi nó cần dự đoán nhãn mới của dữ liệu mới

-Lớp (nhãn) của một đối tượng dữ liệu mới có thể dự đoán các lớp (nhãn) của k hàng xóm gần nó nhất

-Các bước trong KNN

1. Ta có D là tập các dữ liệu đã được gán nhãn và A là dữ liệu chưa được phân loại
2. Đo khoảng cách (Euclidian/ Manhattan/ Minkowski hoặc trọng số) từ dữ liệu mới A đến tất cả các dữ liệu khác đã được phân loại trong D
3. Chọn K (K là tham số được chọn) khoảng cách nhỏ nhất (trong bài toán ta chọn k=3/5)
4. Kiểm tra danh sách các lớp có khoảng cách gần nhất và đếm số lượng của mỗi lớp xuất hiện
5. Lấy đúng lớp (lớp xuất hiện nhiều lần nhất)
6. Lớp của dữ liệu mới là lớp đã nhận được ở bước 5

-Công thức tính khoảng cách Euclidian để tính độ tương đồng giữa ảnh truy vấn và các ảnh trong CSDL bằng vector đặc trưng của chúng.

$$d(x, y) = \sqrt{\sum_{i=1}^n |x_i - y_i|^2}$$

3.Kết quả đạt được :

- +) Tập dữ liệu trên được chia 2 phần gồm 1 phần để train một phần dùng để test độ hiệu quả (theo tỷ lệ train: test là 2:1)
- +) Thí nghiệm được thực hiện với việc sử dụng riêng từng loại đặc trưng và kết hợp các đặc trưng, chi tiết kết quả thí nghiệm được triển khai bên dưới.

STT	Phương pháp trích xuất đặc trưng	Độ chính xác trên tập test
1	Colour Histogram (RGB)	59 %
2	Colour Histogram (HSV)	62 %
3	HOG	33 %
4	(1)+(3)	59 %
5	(2)+(3)	62 %

PHẦN 4. CÀI ĐẶT CHƯƠNG TRÌNH +DEMO

I. Module trích xuất đặc trưng hình ảnh

1. Hàm `rgb_to_hsv(pixel)`:

Đây là một hàm Python có chức năng chuyển đổi giá trị màu của một điểm ảnh từ hệ màu RGB sang hệ màu HSV. Quá trình chuyển đổi được thực hiện bằng cách tìm giá trị Hue (màu sắc), Saturation (độ bão hòa) và Value (độ sáng) của điểm ảnh. Hàm này nhận vào một tham số đầu vào là một đối tượng pixel (bao gồm các giá trị R, G, B) và trả về một danh sách chứa ba giá trị Hue, Saturation, Value tương ứng.

```
1 #This function converts an RGB pixel value to an HSV color model
2 def rgb_to_hsv(pixel):
3     # Extract red, green and blue values from input pixel. then Normalize the RGB values by dividing with 255.0 to get values between 0 and 1
4     r, g, b = pixel
5     r, g, b = r / 255.0, g / 255.0, b / 255.0
6
7     # Find the maximum and minimum value among R, G, and B values
8     v = max(r, g, b)
9     delta = v - min(r, g, b)
10
11     # If there is no difference between the maximum and minimum value, saturation and hue are set to zero
12     if delta == 0:
13         h = 0
14         s = 0
15     else:
16         # Calculate the saturation value
17         s = delta / v
18
19         # Calculate the hue value
20         if r == v:
21             h = (g - b) / delta
22         elif g == v:
23             h = 2 + (b - r) / delta
24         else:
25             h = 4 + (r - g) / delta
26         # Normalize the hue value to be between 0 and 1 and convert it to degrees
27         h = (h / 6) % 1.0
28
29     # Convert the hue, saturation, and value values from float to integer values within the ranges of 0-180, 0-255, and 0-255 respectively
30     return [int(h * 180), int(s * 255), int(v * 255)]
```

2. Hàm `covert_image_rgb_to_hsv(img)`:

Đây là một hàm Python có chức năng chuyển đổi một ảnh từ hệ màu RGB sang hệ màu HSV. Hàm này nhận vào một tham số đầu vào là một ảnh RGB và trả về một ảnh tương ứng được chuyển đổi sang hệ màu HSV. Quá trình chuyển đổi được thực hiện bằng cách duyệt qua từng điểm ảnh trong ảnh đầu vào, sử dụng hàm `rgb_to_hsv` để chuyển đổi giá trị màu của mỗi điểm ảnh từ RGB sang HSV. Cuối cùng, hàm trả về ảnh đã chuyển đổi sang hệ màu HSV dưới dạng một mảng numpy.

```

32 #This function takes an RGB image as input and converts it to an HSV color model
33 def covert_image_rgb_to_hsv(img):
34     # Create an empty array to hold the converted HSV image
35     hsv_image = []
36
37     # Loop through each pixel in the image
38     for i in img:
39         # Create an empty array to hold the converted HSV values of each pixel
40         hsv_image2 = []
41         for j in i:
42             # Convert each RGB pixel value to its HSV equivalent using the rgb_to_hsv() function
43             new_color = rgb_to_hsv(j)
44             hsv_image2.append(new_color)
45         # Add the converted HSV values of each pixel to the array that holds the converted HSV image
46         hsv_image.append(hsv_image2)
47
48     # Convert the list of lists to a numpy array and return the final HSV image
49     hsv_image = np.array(hsv_image)
50     return hsv_image

```

3. Hàm my_calcHist(image, channels, histSize, ranges):

Đây là một hàm Python tính toán histogram của một ảnh đầu vào dựa trên các kênh màu được chỉ định, kích thước histogram và khoảng giá trị. Hàm này nhận vào bốn tham số: image (ảnh đầu vào), channels (các kênh màu được sử dụng để tính histogram), histSize (kích thước của histogram) và ranges (khoảng giá trị cho từng kênh màu). Quá trình tính histogram được thực hiện bằng cách duyệt qua từng điểm ảnh trong ảnh đầu vào, trích xuất giá trị pixel cho các kênh màu đã chỉ định, sau đó tính toán chỉ mục bin tương ứng cho mỗi pixel dựa trên khoảng giá trị đã cho và kích thước histogram. Cuối cùng, hàm trả về histogram tính được dưới dạng một mảng numpy.

```

52 #This function calculates a histogram of an input image based on the specified channels, histogram size and ranges
53 def my_calcHist(image, channels, histSize, ranges):
54     # Initialize a numpy array to hold the histogram counts for each bin
55     hist = np.zeros(histSize, dtype=np.int64)
56
57     # Loop through every pixel in the image
58     for i in range(image.shape[0]):
59         for j in range(image.shape[1]):
60             # Extract the pixel values for the specified color channels
61             bin_vals = [image[i, j, c] for c in channels]
62
63             # Calculate the bin indices (bin_idxs) for each channel based on the pixel values and the range of possible pixel values
64             bin_idxs = [
65                 (bin_vals[c] - ranges[c][0]) * histSize[c]
66                 // (ranges[c][1] - ranges[c][0])
67                 for c in range(len(channels))
68             ]
69
70             # Increment the count of the corresponding bin by 1
71             hist[tuple(bin_idxs)] += 1
72
73     # Return the final histogram as a numpy array
74     return hist

```

4. Hàm convert_image_rgb_to_gray(img_rgb, resize="no"):

Đây là một hàm Python chuyển đổi ảnh từ hệ màu RGB sang hệ màu xám bằng cách sử dụng công thức $Y = 0,299R + 0,587G + 0,114B$. Hàm này nhận vào một tham số đầu vào là một ảnh RGB. Quá trình chuyển đổi được thực hiện bằng cách duyệt qua từng điểm ảnh trong ảnh đầu vào, tính giá trị màu xám tương ứng cho mỗi điểm ảnh bằng cách sử dụng công thức $Y = 0,299R + 0,587G + 0,114B$. Nếu có yêu cầu, hàm sẽ thay đổi kích thước của ảnh đầu ra thành (496, 496). Cuối cùng, hàm trả về ảnh xám đã

chuyển đổi dưới dạng một mảng numpy.

```
4 #This function converts an RGB image to a grayscale image using the formula  $Y = 0.299R + 0.587G + 0.114B$ 
5 def convert_image_rgb_to_gray(img_rgb, resize="no"):
6     # Get the height (h), width (w), and number of channels (_) of the input RGB image
7     h, w, _ = img_rgb.shape
8
9     # Create an empty numpy array of zeros with dimensions (h, w) to hold the converted grayscale values
10    img_gray = np.zeros((h, w), dtype=np.uint32)
11
12    # Convert each pixel from RGB to grayscale using the formula  $Y = 0.299R + 0.587G + 0.114B$ 
13    for i in range(h):
14        for j in range(w):
15            r, g, b = img_rgb[i, j]
16            gray_value = int(0.299 * r + 0.587 * g + 0.114 * b)
17            img_gray[i, j] = gray_value
18
19    # If the resize flag is specified, resize the grayscale image to dimensions of (496, 496)
20    if resize != "no":
21        img_gray = cv2.resize(src=img_gray, dsize=(496, 496))
22
23    # Return the final grayscale image as a numpy array
24    return np.array(img_gray)
```

5. Hàm hog_feature(gray_img):

Đây là một hàm Python tính toán đặc trưng HOG (Histogram of Oriented Gradients) của một ảnh xám bằng cách sử dụng thư viện scikit-image. Hàm này nhận vào một tham số đầu vào là một ảnh xám và trả về một vector đặc trưng HOG tương ứng với ảnh đó dưới dạng một mảng numpy. Quá trình tính toán đặc trưng HOG được thực hiện bằng cách sử dụng hàm hog() của module feature trong thư viện scikit-image và cung cấp các tham số để cấu hình quá trình tính toán, bao gồm: số lượng orientations, kích thước ô pixel, số lượng ô trong mỗi khối, tổng quát hóa căn bậc hai cho các giá trị gradient và chuẩn hóa khối bằng norm L2. Cuối cùng, hàm trả về vector đặc trưng HOG như một mảng numpy.

```
27 def hog_feature(gray_img): # default gray_image
28     # Compute the HOG features and the HOG visualization image using the scikit-image "feature" module's hog() function.
29     (hog_feats, hogImage) = feature.hog(
30         gray_img,
31         orientations=9,
32         pixels_per_cell=(8, 8),
33         cells_per_block=(2, 2),
34         transform_sqrt=True,
35         block_norm="L2",
36         visualize=True,
37     )
38
39     # Return the HOG feature descriptor as a numpy array
40     return hog_feats
41
```

6. Hàm feature_extraction(img):

Đây là một hàm Python trích xuất đặc trưng từ một ảnh RGB bằng cách sử dụng histogram của hướng gradient (HOG) và histogram màu ba chiều trong không gian màu HSV. Quá trình trích xuất đặc trưng được thực hiện bằng cách tính toán histogram màu ba chiều của ảnh đầu vào trong không gian màu HSV, sau đó chuyển đổi ảnh RGB thành ảnh xám và tính toán đặc trưng HOG của ảnh xám đó. Cuối cùng, hàm kết hợp

hai vector đặc trưng này lại và trả về một vector đặc trưng cuối cùng.

```
1 #This is a Python function that extracts features from an RGB image using
2 #Histogram of Oriented Gradients (HOG) and a 3D color histogram in HSV color space.
3 def feature_extraction(img): # RGB image
4     bins = [8, 12, 3]
5     ranges = [[0, 180], [0, 256], [0, 256]]
6     img_hsv = convert_image_rgb_to_hsv(img)
7     hist_my = my_calcHist(img_hsv, [0, 1, 2], bins, ranges)
8     embedding_hsv = list(hist_my.flatten())
9     embedding_hsv[0] = 0
10
11     gray_image = convert_image_rgb_to_gray(img)
12     embedding_hog = list(hog_feature(gray_image))
13
14     data = embedding_hsv + embedding_hog
15     return data
```

II. Module tìm kiếm

1.Hàm distance_euclidean(x, y):

Đây là một hàm Python tính toán khoảng cách Euclidean giữa hai vector x và y. Khoảng cách Euclidean được tính bằng căn bậc hai của tổng bình phương khoảng cách giữa từng cặp thành phần tương ứng của các vector

```
3 #This is a Python function that calculates the Euclidean distance between two vectors x and y
4 def distance_euclidean(x, y):
5     if len(x) != len(y):
6         return None # Invalid input
7     squared_distance = 0
8     for i in range(len(x)):
9         squared_distance += (x[i] - y[i]) ** 2
10    return squared_distance ** 0.5
```

2.Hàm knn(X_train, y_train, x_new, k):

Hàm thực hiện phân loại k-Nearest Neighbors (k-NN) trên một điểm dữ liệu mới x_new, cho một tập dữ liệu huấn luyện X_train và nhãn tương ứng y_train. Hàm tính toán khoảng cách Euclidean giữa mẫu mới và các mẫu huấn luyện, chọn ra k láng giềng gần nhất và đếm số lần xuất hiện của mỗi nhãn trong k láng giềng gần nhất. Cuối cùng, hàm trả về nhãn dự đoán cho mẫu mới, được xác định bằng cách chọn nhãn có số lần xuất hiện nhiều nhất trong k láng giềng gần nhất.

```

13 # This is a Python function that performs k-Nearest Neighbors (k-NN) classification on a new data point x_new,
14 # given a training dataset X_train and corresponding labels y_train
15 def knn(X_train, y_train, x_new, k):
16     distances = []
17     #each data point in X_train, the Euclidean distance between that point and x_new is calculated.
18     for i in range(len(X_train)):
19         d = distance_euclidean(X_train[i], x_new)
20         distances.append((d, y_train[i]))
21
22     #The list of distances is then sorted in ascending order based on the distance values, and the k nearest neighbors are selected
23     distances.sort()
24     k_nearest = distances[:k]
25
26     # a dictionary "labels" count the occurrences of each label among the k nearest neighbors
27     labels = {}
28     for distance in k_nearest:
29         label = distance[1]
30         if label in labels:
31             labels[label] += 1
32         else:
33             labels[label] = 1
34     #The predicted label for the new data point is the label with the highest count
35     sorted_labels = sorted(labels.items(), key=lambda x: (x[1],x[0]), reverse=True)
36     return sorted_labels[0][0]

```

III. Kết quả kiểm nghiệm

1. Hình ảnh hoa hồng (rose)

```

[86] 1 path_image_test1="/content/drive/MyDrive/Tài liệu năm 4/kỳ 2/HCSLDPT/BTL Của Huy/2 Code/Hauu/data_test/hoahong.jpg"
      2 img = cv2.imread(path_image_test1, cv2.COLOR_BGR2RGB)
      3 # print(img)
      4 img_embedding = feature_extraction(img)
      5 print(len(img_embedding))

134244

```

```

[87] 1 res= knn(X_HOG_HSV,y_HOG_HSV,img_embedding,7)
      2 print(dict[str(res)])
      3 cv2.imshow(img)

```




2. Hình ảnh hoa diên vĩ vàng (yellow iris)

```

1 path_image_test1="/content/drive/MyDrive/Tài liệu năm 4/kỳ 2/HCSOLDPT/BTL Của Huy/2 Code/Hauu/data_test/dienvivang.jpg"
2 img = cv2.imread(path_image_test1, cv2.COLOR_BGR2RGB)
3 img_embedding = feature_extraction(img)
4 print(len(img_embedding))
5 res= knn(X_HOG_HSV,y_HOG_HSV,img_embedding,7)
6 print(dict[str(res)])
7 cv2_imshow(img)
8

```

134244
yellow iris




3. Hoa hải quỳ trắng (windflower)

```

1 path_image_test1="/content/drive/MyDrive/Tài liệu năm 4/kỳ 2/HCSOLDPT/BTL Của Huy/2 Code/Hauu/data_test/haiquytrang.jpg"
2 img1 = cv2.imread(path_image_test1, cv2.COLOR_BGR2RGB)
3 img_embedding1 = feature_extraction(img1)
4 res1= knn(X_HOG_HSV,y_HOG_HSV,img_embedding1,7)
5 print(dict[str(res1)])
6 cv2_imshow(img1)
7

```

windflower



4. Hoa bồ công anh vàng (common dandelion)

```

1 path_image_test1="/content/drive/MyDrive/Tài liệu năm 4/kỳ 2/HCSOLDPT/BTL Của Huy/2 Code/Hauu/data_test/boconganhvang.jpg"
2 img1 = cv2.imread(path_image_test1, cv2.COLOR_BGR2RGB)
3 img_embedding1 = feature_extraction(img1)
4 res1= knn(X_HOG_HSV,y_HOG_HSV,img_embedding1,7)
5 print(dict[str(res1)])
6 cv2_imshow(img1)
7

```

common dandelion



PHẦN 5: TÀI LIỆU THAM KHẢO

1. [Color Histograms in Image Retrieval | Pinecone](#)
2. [Object Classification Using Histogram of Object Oriented Gradients | Engineering Education \(EngEd\) Program | Section](#)
3. [SIFT \(Scale-invariant feature transform\)](#)
4. [Introduction to SURF \(Speeded-Up Robust Features\)](#)
5. [Hiên thực trích đặc trưng Local Binary Patterns \(LBP\)](#)
6. [GLCMs — a Great Tool for Your ML Arsenal](#)
7. [Image Classification with K Nearest Neighbours](#)
8. [SVM \(Support Vector Machine\) for classification](#)
9. [Naïve Bayes vs. SVM for Image Classification](#)
10. <https://medium.com/@VigneshSaravanan/machine-learning-ml-deep-learning-dl-an-intro-86da03f8e8dc>
11. [HSV Color Model in Computer Graphics - GeeksforGeeks](#)
12. [Color models and color spaces - Programming Design Systems](#)
13. [Xử lý ảnh: thuật toán cân bằng histogram ảnh \(viblo.asia\)](#)
14. [Thị giác máy tính với OpenCV-Python Bài 4, Phần 1: Thay đổi không gian màu \(imc.org.vn\)](#)
15. <https://pyimagesearch.com/2021/04/28/opencv-image-histograms-cv2-calchist/>
16. [Thuật toán HOG \(Histogram of oriented gradient\)](#)
17. [Tìm hiểu về phương pháp mô tả đặc trưng HOG \(Histogram of Oriented Gradients\) \(viblo.asia\)](#)
18. [Trích đặc trưng HOG - Histograms of Oriented Gradients \(minhng.info\)](#)
19. [HOG \(Histogram of Oriented Gradients\): An Overview | by Mrinal Tyagi | Towards Data Science](#)
20. [Histogram of Oriented Gradients explained using OpenCV \(learnopencv.com\)](#)
21. [Histogram of Oriented Gradients \(HOG\) — Simplest Intuition | by Skillcate AI | Medium](#)
22. [Feature Descriptor | Hog Descriptor Tutorial \(analyticsvidhya.com\)](#)
23. [Histogram of Oriented Gradients \(HOG\) Boat Heading Classification | by Adam Van Etten | The DownLinQ | Medium](#)
24. [Using Histogram of Oriented Gradients \(HOG\) for Object Detection](#)

[\(\[opengenius.org\]\(http://opengenius.org\)\)](http://opengenius.org)

25. <https://machinelearningmastery.com/tutorial-to-implement-k-nearest-neighbors-in-python-from-scratch/>
26. <https://viblo.asia/p/knn-k-nearest-neighbors-1-djeZ14ejKWz>
27. https://www.python-engineer.com/courses/mlfromscratch/01_knn/
28. ...