

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**



**BÁO CÁO BÀI TẬP LỚN**

**MÔN HỌC: HỆ CƠ SỞ DỮ LIỆU ĐA PHƯƠNG TIỆN**

**Đề tài: Xây dựng hệ cơ sở dữ liệu lưu trữ và tìm kiếm ảnh  
hoa quả**

**Nhóm:** 03

**Nhóm bài tập:** 04

**Giảng viên:** Nguyễn Đình Hóa

**Nhóm sinh viên:** Nguyễn Công Hiệp – B20DCCN239  
Phạm Duy Hùng – B20DCCN299  
Bùi Quốc Huy – B20DCCN305

**Hà Nội, tháng 6 năm 2024**

## MỤC LỤC

<b>LỜI CẢM ƠN</b>	<b>3</b>
<b>I. GIỚI THIỆU CHUNG</b>	
1. Yêu cầu	4
2. Hình ảnh và cách lưu trữ ảnh hoa quả	
3. Định dạng, tính chất dữ liệu ảnh	4
<b>II. CÁC KỸ THUẬT XỬ LÝ ẢNH HOA QUẢ HIỆN HÀNH</b>	<b>8</b>
1. Các kỹ thuật xử lý và trích rút đặc trưng ảnh hiện hành	8
1.1. Color Histogram	10
1.2. HOG	12
1.3. SIFT	
1.4. SURF	
1.5. LBP (Local Binary Patterns)	
1.6. GLCM	16
2. Các phương pháp đánh giá độ tương đồng	16
2.1 Độ tương đồng tuyệt đối	16
2.2 Khoảng cách Ô-clit	16
2.3 Khoảng cách Cosine	16
<b>III. XÂY DỰNG HỆ THỐNG NHẬN DẠNG HOA QUẢ</b>	<b>17</b>
1. Sơ đồ khối của hệ thống và quy trình thực hiện yêu cầu của đề bài	17
2. Trình bày các thuộc tính được sử dụng để nhận dạng nhãn của ảnh hoa quả trong hệ thống, cùng các kỹ thuật để trích rút các thuộc tính đó.	17
3. Trình bày cách lưu trữ các thuộc tính ảnh quả và cách nhận dạng ảnh hoa quả dựa trên các thuộc tính đó.	22
<b>IV. DEMO HỆ THỐNG VÀ ĐÁNH GIÁ KẾT QUẢ</b>	<b>22</b>
1. Kết quả demo phân loại hoa quả	
2. Đánh giá kết quả đào tạo và dự đoán	
<b>TÀI LIỆU THAM KHẢO</b>	

## LỜI CẢM ƠN

Trước tiên với tình cảm sâu sắc và chân thành nhất, cho phép chúng em xin gửi lòng biết ơn đến quý thầy cô tại Học viện Công nghệ Bưu chính Viễn thông đã tạo điều kiện hỗ trợ, giúp đỡ chúng em suốt quá trình học tập và nghiên cứu vừa qua.

Đặc biệt, trong học kỳ này, học viện đã tổ chức cho chúng em được tiếp cận với các môn học rất hữu ích đối với sinh viên. Chúng em xin chân thành cảm ơn thầy Nguyễn Đình Hóa đã tận tâm hướng dẫn chúng em trong môn học Hệ cơ sở dữ liệu Đa phương tiện. Thầy đã luôn bên cạnh, tạo điều kiện trong suốt quá trình nghiên cứu, động viên và giúp đỡ để chúng em hoàn thành tốt báo cáo này.

Do kiến thức còn nhiều hạn chế và khả năng tiếp thu thực tế còn nhiều bỡ ngỡ chưa hoàn hảo nên bài báo cáo sẽ còn nhiều thiếu sót, kính mong sự góp ý và giúp đỡ từ thầy. Cuối cùng chúng em xin kính chúc quý thầy cô dồi dào sức khỏe, niềm tin để tiếp tục thực hiện sứ mệnh cao đẹp của mình là truyền đạt kiến thức cho thế hệ mai sau. Chúng em xin chân thành cảm ơn.

## I. GIỚI THIỆU CHUNG

### 1. Yêu cầu bài toán

Xây dựng hệ CSDL lưu trữ và tìm kiếm ảnh hoa quả.

1. Hãy xây dựng/sưu tầm một bộ dữ liệu ảnh gồm ít nhất 100 files ảnh về các loại quả khác nhau, các ảnh có cùng kích thước, vật trong ảnh có cùng tỉ lệ khung hình (SV tùy chọn định dạng ảnh).

2. Hãy xây dựng một bộ thuộc tính để nhận diện các ảnh quả khác nhau từ bộ dữ liệu đã thu thập.

3. Xây dựng hệ thống tìm kiếm ảnh hoa quả với đầu vào là một ảnh mới về một quả nào đó (quả đã có và không có trong dữ liệu), đầu ra là 3 ảnh giống nhất, xếp thứ tự giảm dần về độ tương đồng nội dung với 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 quá trình trích rút, lưu trữ và sử dụng các thuộc tính để tìm kiếm ảnh trong hệ thống.

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

### 2. Hình ảnh và cách lưu trữ ảnh

- Nguồn dữ liệu ảnh : [Fruit classification\(10 Class\) \(kaggle.com\)](https://www.kaggle.com/datasets/fra91/fruits-101-classification)

- Bộ dữ liệu gồm 8 loại quả, cụ thể :

[Apple, Bell pepper, Lemon, Mango, Orange, Pear, Plums, Strawberries]

#### 1. Apple



Ảnh 1.1

Ảnh 1.2

Ảnh 1.3

## 2. Bell Pepper



Ảnh 2.1



Ảnh 2.2



Ảnh 2.3

## 3. Lemon



Ảnh 3.1



Ảnh 3.2

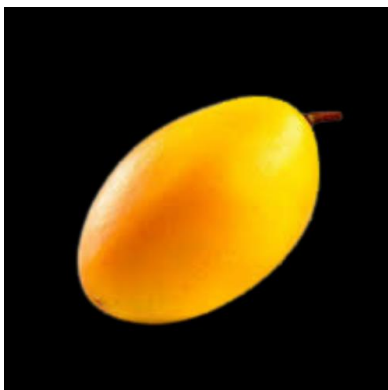


Ảnh 3.3

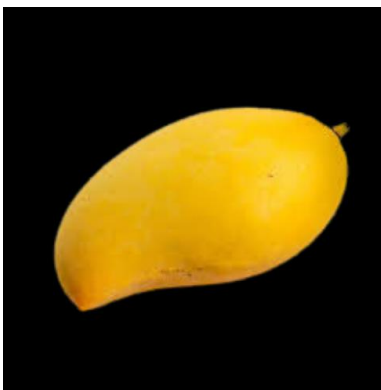
## 4. Mango



Ảnh 4.1

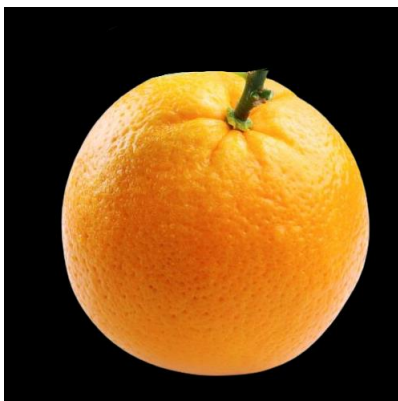


Ảnh 4.2

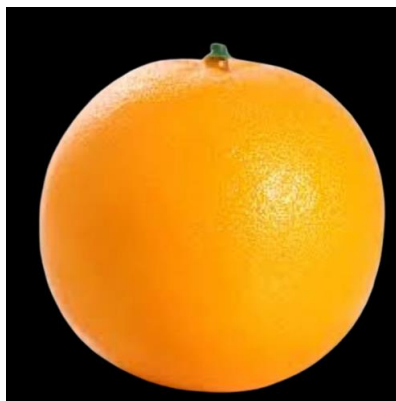


Ảnh 4.3

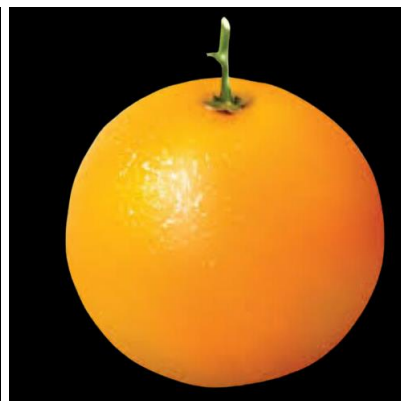
## 5. Orange



**Ảnh 5.1**

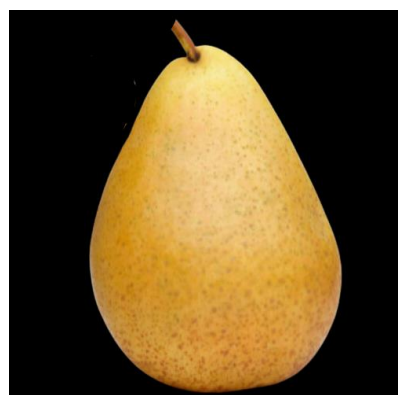


**Ảnh 5.2**



**Ảnh 5.3**

## 6. Pear



**Ảnh 6.1**



**Ảnh 6.2**



**Ảnh 6.3**

## 7. Plums



**Ảnh 7.1**



**Ảnh 7.2**



**Ảnh 7.3**

## 8. Strawberries



Ảnh 8.1



Ảnh 8.2



Ảnh 8.3

- Cách lưu trữ ảnh :

Các hình ảnh của 8 loại quả được lưu vào 8 thư mục khác nhau :



### 3. Định dạng, tính chất của ảnh

- Mỗi bức ảnh có kích thước 500 x 500 px, định dạng tệp hình ảnh là .png , các bức ảnh đều đã được xóa nền
- Bức ảnh có 1 hoặc 1 vài quả, đa số các bức ảnh có hình quả chiếm khoảng 60 – 70% diện tích bức ảnh
- Các hình ảnh của cùng 1 loại quả 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 quả khác nhau thì có thể màu sắc và hình dáng có thể khá tương đồng(ví dụ quả nho và quả táo)
- Sự kết hợp giữa yếu tố màu sắc và hình dáng sẽ tạo nên đặc trưng của các loại quả và có thể phân biệt các loại quả với nhau.

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



## II. CÁC KỸ THUẬT XỬ LÝ VÀ PHÂN LOẠI ẢNH HOA QUẢ HIỆN HÀNH

### 1. Các kỹ thuật xử lý và trích rút đặc trưng ảnh hiện hành

#### - *Khái niệm đặc trưng ảnh:*

Trong phạm vi xử lý ảnh, đặc trưng ảnh là một phần thông tin của ảnh thích hợp cho các nhiệm vụ tính toán liên quan đến một ứng dụng nhất định. Những đặc trưng đó có thể là các kết cấu đặc biệt trong ảnh như các điểm, các cạnh của một đối tượng hoặc một đối tượng nào đó có trong ảnh. Mặt khác, các đặc trưng của ảnh cũng có thể là kết quả của một phép biến đổi toàn diện hoặc là các phương pháp phát hiện điểm đặc trưng được áp dụng trên toàn bộ ảnh đó.

Trích rút đặc trưng ảnh là quá trình xử lý làm cho ảnh ban đầu được biến đổi thành các dạng mà máy tính có thể dễ dàng nhận dạng hơn.

#### - *Tại sao phải trích rút đặc trưng ảnh ?*

Không giống như thị giác con người, thị giác máy có khả năng nhìn nhận rất hạn chế bởi đối với máy tính, mỗi hình ảnh chỉ là một ma trận các điểm ảnh. Vì vậy, đối với các lĩnh vực có liên quan đến xử lý các lượng lớn hình ảnh, bài toán trích rút đặc trưng ảnh là rất quan trọng đối với công tác nhận dạng, phân loại dữ liệu hình ảnh hoặc gán gũ hơn là tìm kiếm dữ liệu trong một lượng lớn dữ liệu hình ảnh có sẵn.

Mặt khác, việc trích rút đặc trưng ảnh cũng là một công cụ rất có ích trong việc xác định các phần tương đồng hoặc giống nhau của các ảnh, từ đó có thể phát triển các ứng dụng ghép nối hình ảnh, dựng ảnh 3D hay so sánh một cách dễ dàng và chính xác hơn.

### **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

- 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:**

### **1.1 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

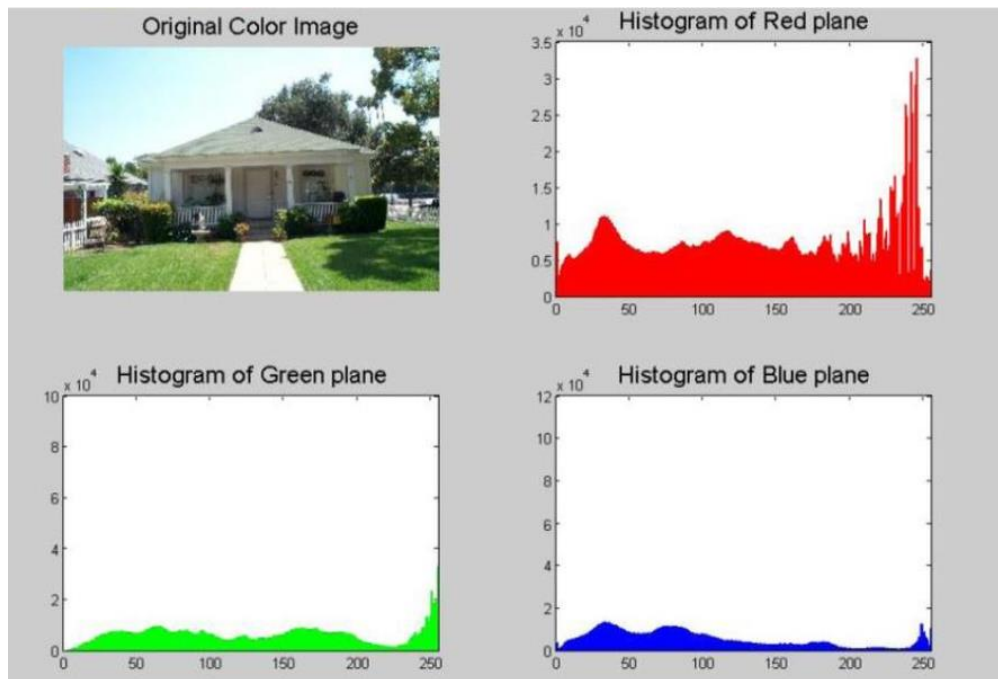
+ Tính toán nhanh, đơn giản, thích hợp trong các ứng dụng thời gian thực

+ 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



## 1.2 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 nâng cao độ chính xác. Hog được sử dụng để mô tả hình dạng và sự xuất hiện của một object trong ảnh.

- Bài toán tính toán Hog thường gồm 5 bước:

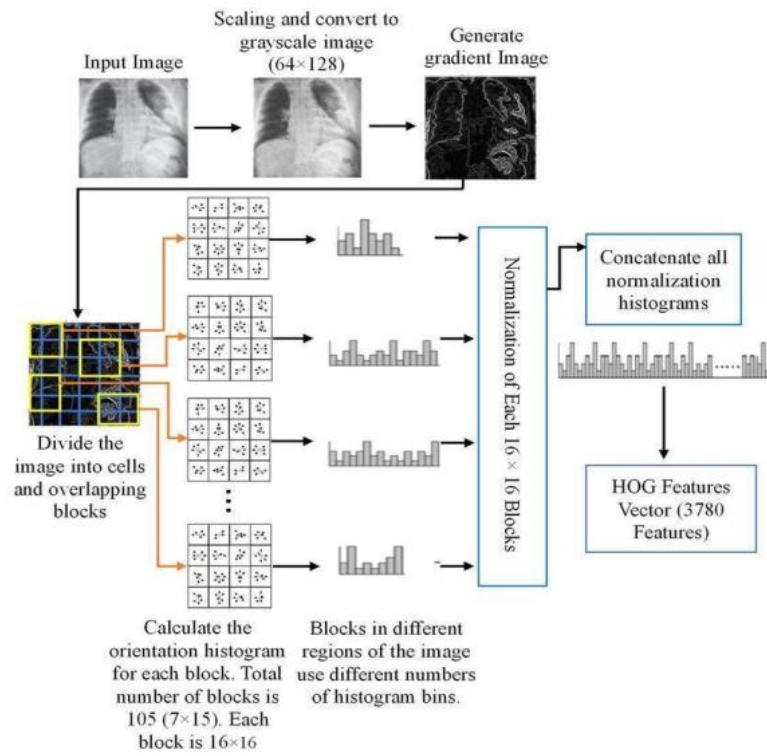
- + Chuẩn hóa hình ảnh trước khi xử lý
- + Tính toán gradient theo cả hướng x và y.
- + Lấy phiếu bầu cùng trọng số trong các cell
- + Chuẩn hóa các block
- + Thu thập các biểu đồ cường độ gradient định hướng để tạo ra feature vector cuối cùng.

- 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



### 1.3 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

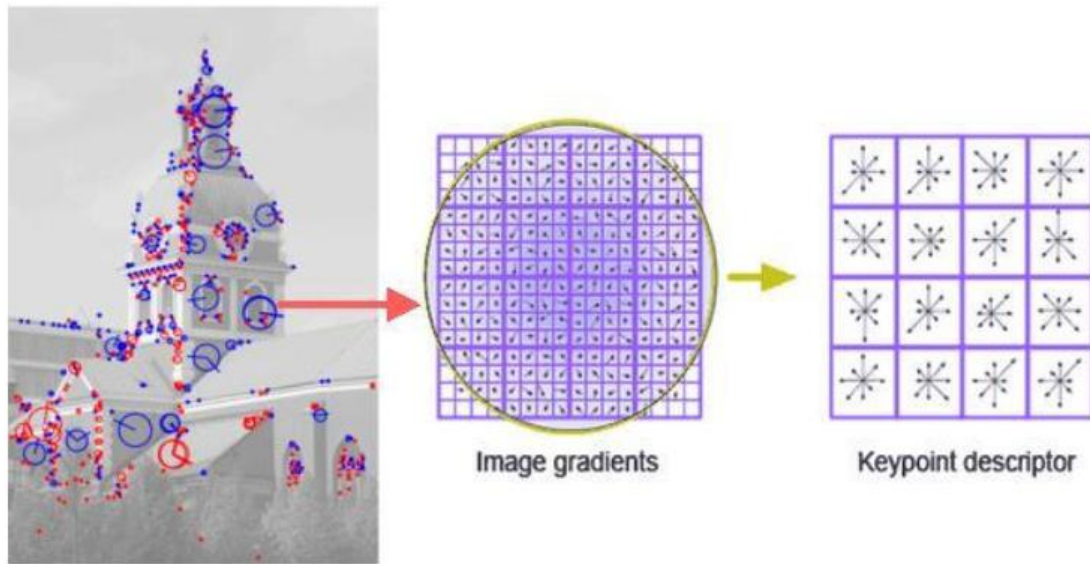
+ Tính ổn định và độ tin cậy cao

- Nhược điểm:

+ Tốc độ xử lý chậm hơn

+ Độ phức tạp tính toán cao

+ Số lượng điểm đặc trưng lớn



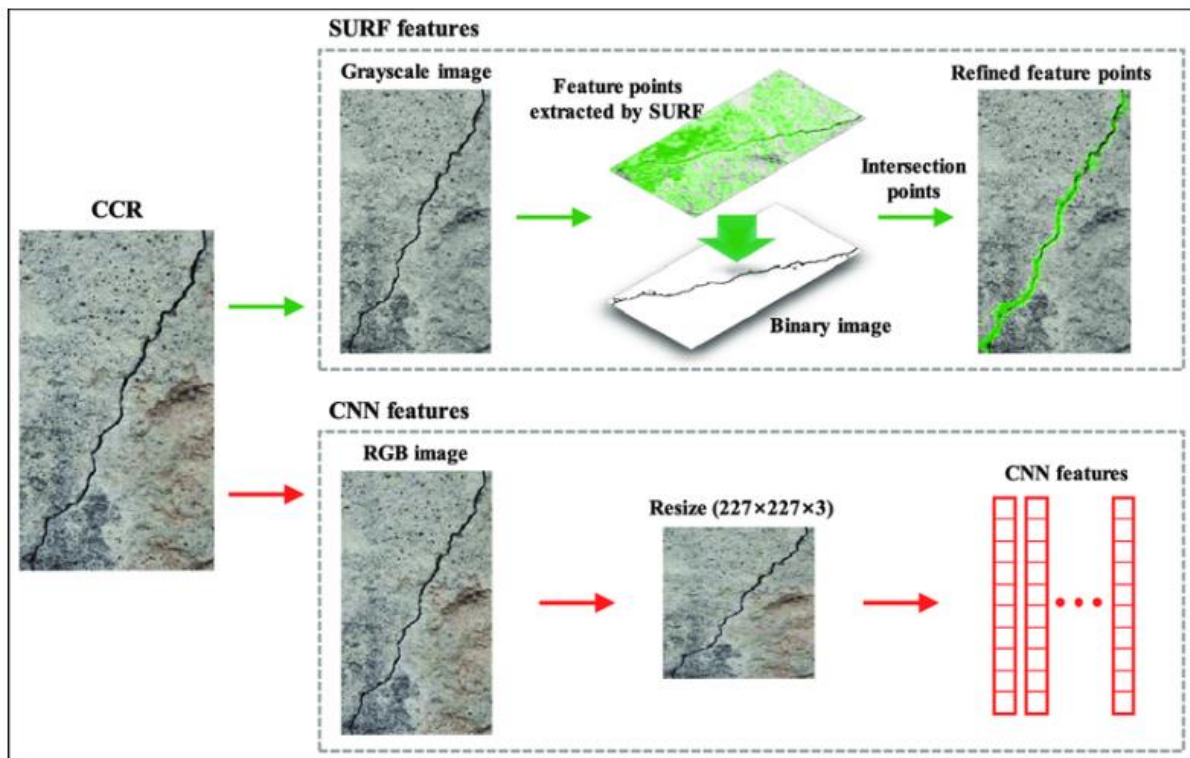
## 1.4 SURF

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 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)



## 1.5 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 x 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
  - 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  $3 \times 3$  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.

## 1.6. 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

## 2. Các phương pháp đánh giá độ tương đồng

### 2.1 Độ tương đồng tuyệt đối

Cách đơn giản nhất để tìm độ tương đồng (similarity) giữa 2 hình ảnh đó là xét xem chúng có giống hệt nhau hay không:

$$\begin{cases} \text{Similarity}(x, y) = 1 & \text{if } x = y \\ \text{Similarity}(x, y) = 0 & \text{if } x \neq y \end{cases}$$

trong đó, x và y là 2 ảnh quả.

### 2.2 Khoảng cách O-clit

Khoảng cách σ-clit (euclidean distance) có thể hiểu là độ dài đường thẳng nối 2 điểm trong không gian:

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

trong đó, d là khoảng cách σ-clit, n là số chiều trong không gian,  $x_i$ ;  $y_i$  là các giá trị trong vector biểu diễn của 2 hình ảnh hoa quả cần tính độ tương đồng.

### 2.3 Khoảng cách Cosine

Độ tương đồng của 2 hình ảnh vân tay khi sử dụng khoảng cách Cosine (cosine similarity) được tính như sau:

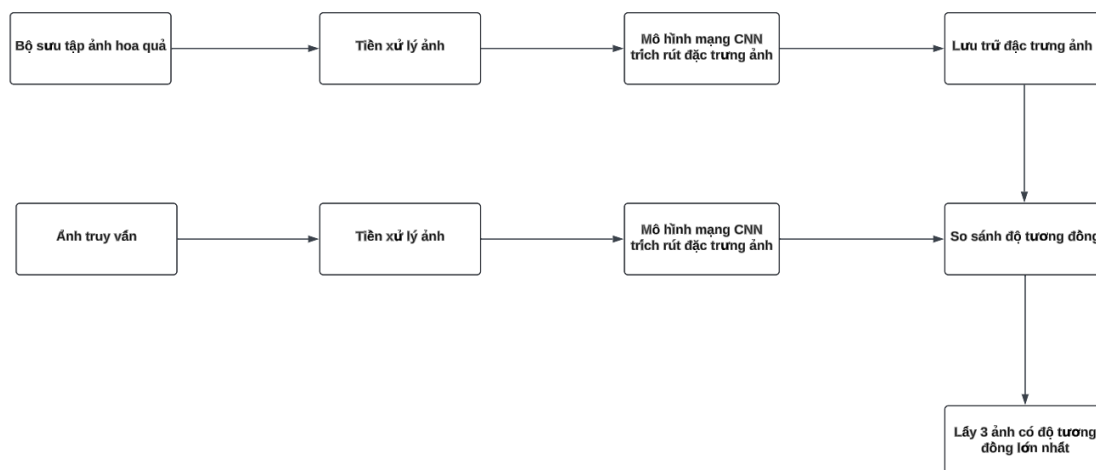
$$\text{Similarity}(x, y) = \cos(\theta) = \frac{\sum_{i=1}^n x_i y_i}{\sqrt{\sum_{i=1}^n x_i^2 \sum_{i=1}^n y_i^2}}$$

trong đó,  $\theta$  là góc giữa 2 vector biểu diễn cho 2 hình ảnh hoa quả,  $x_i$ ;  $y_i$  là các giá trị trong vector biểu diễn cho 2 hình ảnh hoa quả.

### III. XÂY DỰNG HỆ THỐNG NHẬN DẠNG HOA QUẢ

#### 1. Sơ đồ khối của hệ thống và quy trình thực hiện yêu cầu của đề bài

##### - Sơ đồ khối:



*Sơ đồ khối hệ thống nhận dạng ảnh hoa quả*

##### Quy trình thực hiện :

- Bước 1: Tiền xử lý bộ sưu tập ảnh quả , đưa hình ảnh về cùng kích thước 500 x 500, nền đen và tỉ lệ khung hình gần tương đương nhau
- Bước 2: Sử dụng mô hình CNN trích rút đặc trưng ảnh và lưu vào model
- Bước 3: Sử dụng model đã huấn luyện nhận diện đặc trưng ảnh truy vấn
- Bước 4: Tính toán độ tương đồng giữa ảnh truy vấn và các ảnh còn lại trong bộ sưu tập đã huấn luyện
- Bước 5: Tìm kiếm và hiển thị 3 ảnh có độ tương đồng gần nhất với ảnh truy vấn

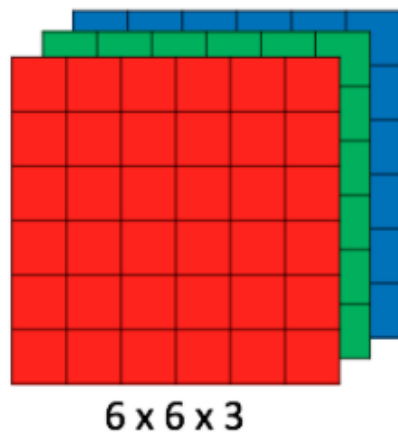
#### Q2. Trình bày các thuộc tính được sử dụng để nhận dạng nhãn của ảnh hoa quả trong hệ thống, cùng các kỹ thuật để trích rút các thuộc tính đó.

##### 1. Giới thiệu

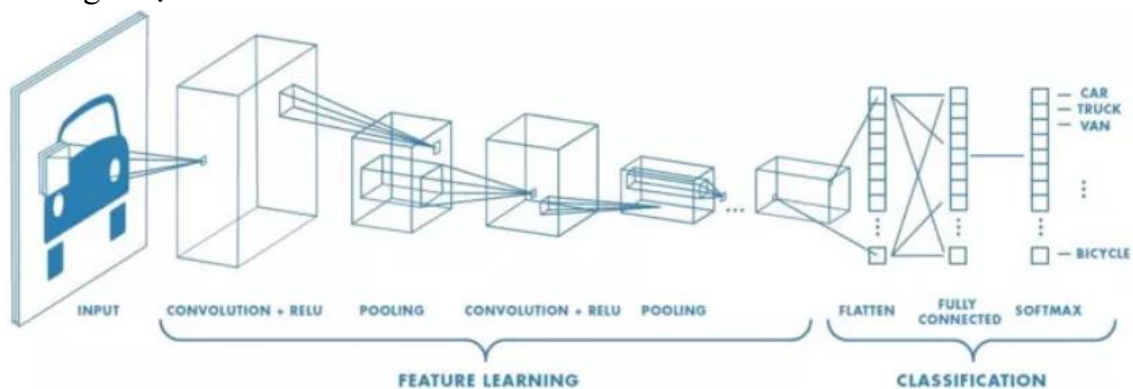
Trong mạng neural, mô hình mạng neural tích chập (CNN) là 1 trong những mô hình để nhận dạng và phân loại hình ảnh. Trong đó, xác định đối tượng và nhận dạng khuôn mặt là 1 trong số những lĩnh vực mà CNN được sử dụng rộng rãi.

CNN phân loại hình ảnh bằng cách lấy 1 hình ảnh đầu vào, xử lý và phân loại nó theo các hạng mục nhất định (Ví dụ: Chó, Mèo, Hổ, ...). Máy tính coi hình ảnh đầu vào là 1 mảng pixel và nó phụ thuộc vào độ phân giải của hình ảnh. Dựa trên độ phân giải hình ảnh, máy

tính sẽ thấy  $H \times W \times D$  (H: Chiều cao, W: Chiều rộng, D: Độ dày). Ví dụ: Hình ảnh là mảng ma trận RGB  $6 \times 6 \times 3$  (3 ở đây là giá trị RGB).



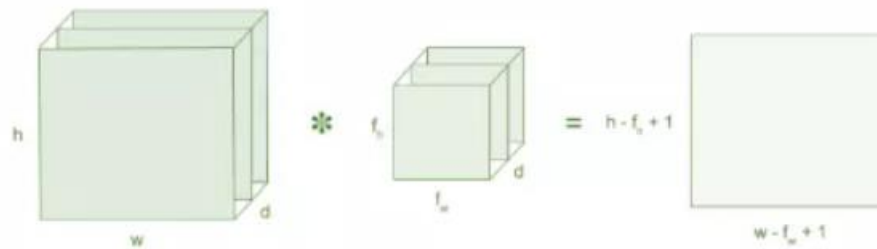
Về kỹ thuật, mô hình CNN để training và kiểm tra, mỗi hình ảnh đầu vào sẽ chuyển nó qua 1 loạt các lớp tích chập với các bộ lọc (Kernels), tổng hợp lại các lớp được kết nối đầy đủ (Full Connected) và áp dụng hàm Softmax để phân loại đối tượng có giá trị xác suất giữa 0 và 1. Hình dưới đây là toàn bộ luồng CNN để xử lý hình ảnh đầu vào và phân loại các đối tượng dựa trên giá trị.



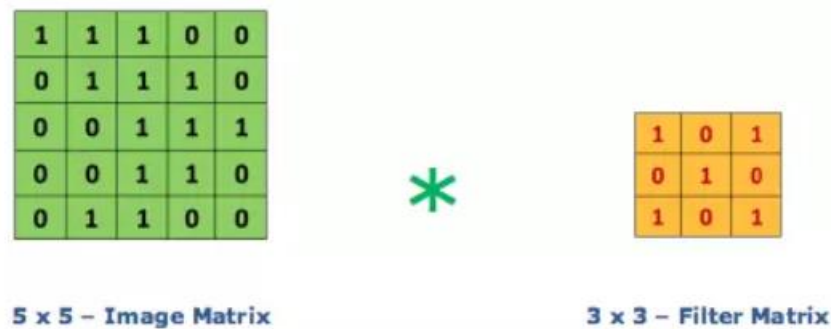
## 2. Lớp tích chập - Convulatrion Layer

Tích chập là lớp đầu tiên để trích xuất các tính năng từ hình ảnh đầu vào. Tích chập duy trì mối quan hệ giữa các pixel bằng cách tìm hiểu các tính năng hình ảnh bằng cách sử dụng các ô vuông nhỏ của dữ liệu đầu vào. Nó là 1 phép toán có 2 đầu vào như ma trận hình ảnh và 1 bộ lọc hoặc hạt nhân.

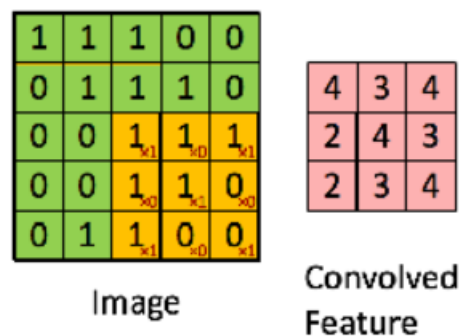
- An image matrix (volume) of dimension **(h x w x d)**
- A filter **(f<sub>h</sub> x f<sub>w</sub> x d)**
- Outputs a volume dimension **(h - f<sub>h</sub> + 1) x (w - f<sub>w</sub> + 1) x 1**










Xem xét 1 ma trận 5 x 5 có giá trị pixel là 0 và 1. Ma trận bộ lọc 3 x 3 như hình bên dưới.



Sau đó, lớp tích chập của ma trận hình ảnh 5 x 5 nhân với ma trận bộ lọc 3 x 3 gọi là 'Feature Map' như hình bên dưới.

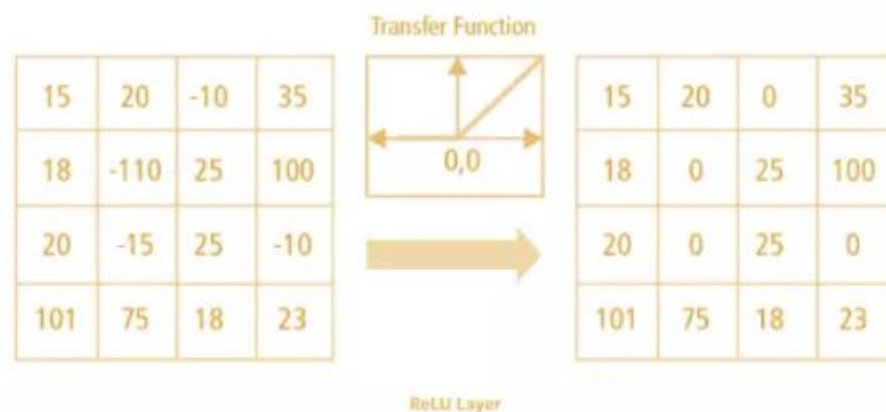


Sự kết hợp của 1 hình ảnh với các bộ lọc khác nhau có thể thực hiện các hoạt động như phát hiện cạnh, làm mờ và làm sắc nét bằng cách áp dụng các bộ lọc. Ví dụ dưới đây cho thấy hình ảnh tích chập khác nhau sau khi áp dụng các Kernel khác nhau.

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

### 3. Hàm phi tuyến – ReLU

ReLU viết tắt của Rectified Linear Unit, là 1 hàm phi tuyến. Với đầu ra là:  $f(x) = \max(0, x)$ . ReLU giới thiệu tính phi tuyến trong ConvNet. Vì dữ liệu trong thế giới mà chúng ta tìm hiểu là các giá trị tuyến tính không âm.



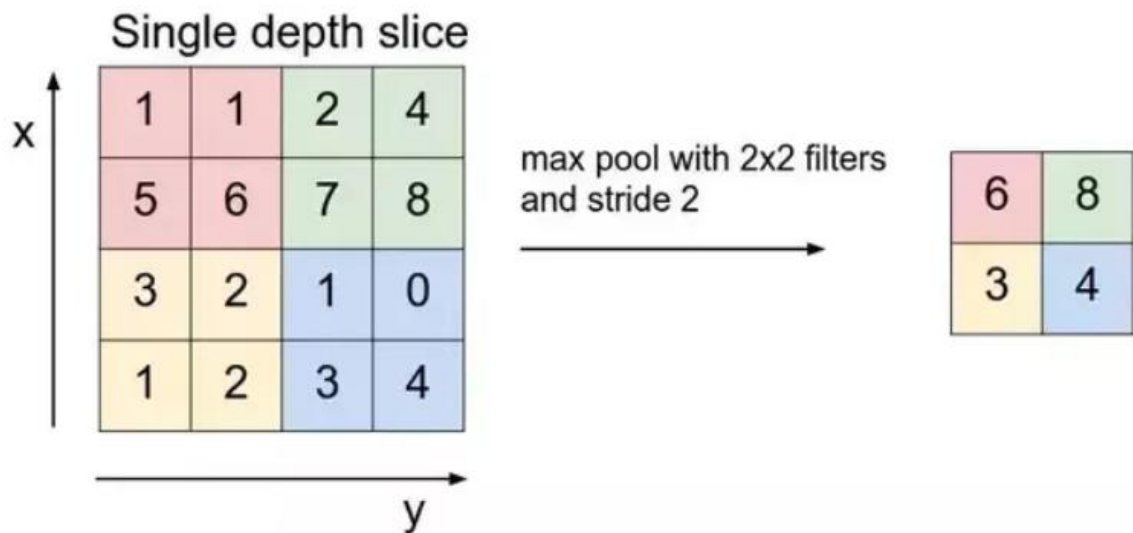
#### 4. Lớp gộp – Pooling Layer

Lớp pooling sẽ giảm bớt số lượng tham số khi hình ảnh quá lớn. Không gian pooling còn được gọi là lấy mẫu con hoặc lấy mẫu xuống làm giảm kích thước của mỗi map nhưng vẫn giữ lại thông tin quan trọng.

Các pooling có thể có nhiều loại khác nhau:

- Max Pooling
- Average Pooling
- Sum Pooling

Max pooling lấy phần tử lớn nhất từ ma trận đối tượng, hoặc lấy tổng trung bình. Tổng tất cả các phần tử trong map gọi là sum pooling



#### 5. Tóm tắt

- Đầu vào (Input): Hình ảnh quả với kích thước (width, height, channels).
- Lớp convolutional (Convolutional layer):
  - Bước 1: Sử dụng bộ lọc (filter) để quét qua toàn bộ hình ảnh.
  - Bước 2: Tính tổng trọng số của các pixel được nhân với trọng số của bộ lọc.
  - Bước 3: Áp dụng hàm kích hoạt để tạo ra các feature map.
- Lớp pooling (Pooling layer):
  - Bước 1: Chia feature map thành các vùng không chồng lấp.
  - Bước 2: Áp dụng một hàm giảm kích thước (max pooling, average pooling,...) để lấy giá trị tối đa hoặc trung bình của các vùng này.
- Lớp Flatten: Chuyển đổi feature map thành một vector 1D.
- Lớp Fully connected (Dense layer):
  - Bước 1: Kết nối mỗi nơ-ron với tất cả các nơ-ron ở lớp trước đó.
  - Bước 2: Áp dụng hàm kích hoạt để tạo ra đầu ra.
- Đầu ra (Output): Vector đặc trưng, thường được sử dụng để phân loại quả hoặc thực hiện các tác vụ khác.

## 6. Các đặc trưng quan trọng

Các đặc trưng quan trọng mà mạng CNN có thể học được bao gồm:

- Các đặc điểm hình dạng của quả: Các đặc điểm như hình dáng, đường viền, kích thước và tỷ lệ của quả có thể được mạng CNN học và sử dụng để phân loại và nhận dạng loại quả.
- Màu sắc và cấu trúc của quả: Mạng CNN có thể học cách nhận biết và phân biệt màu sắc và cấu trúc của quả. Ví dụ, màu sắc của vỏ, màu sắc của thịt và mối quan hệ giữa chúng có thể là các đặc trưng quan trọng.
- Các đặc điểm textural: Đối với một số loại quả, textural features như bề mặt, độ bóng, hoặc độ roughness cũng có thể là các đặc trưng quan trọng.
- Đặc điểm về mối liên kết giữa các bộ phận của quả: Mạng CNN có thể học được mối quan hệ giữa các bộ phận của quả, chẳng hạn như mối quan hệ giữa vỏ và thịt, hoặc giữa các hạt và môi trường xung quanh.
- Tùy thuộc vào loại quả cụ thể và dữ liệu huấn luyện, các đặc trưng quan trọng có thể thay đổi. Nhưng trong tất cả các trường hợp, mạng CNN sẽ cố gắng học và trích xuất những đặc điểm quan trọng nhất để giúp phân loại các loại quả một cách chính xác.

## 7. Tính độ tương đồng

Công thức được sử dụng là công thức cosine similarity, một phép đo độ tương tự giữa hai vector trong không gian đa chiều. Công thức này đo lường góc giữa hai vector và cho biết mức độ giống nhau của chúng.

Công thức cosine similarity được tính như sau:

$$\text{similarity} = \frac{\text{feature1} \cdot \text{feature2}^T}{\|\text{feature1}\| \cdot \|\text{feature2}\|}$$

Trong đó:

- $\text{feature1} \cdot \text{feature2}^T$  là tích vô hướng của hai vector.
- $\|\text{feature1}\|$  và  $\|\text{feature2}\|$  là độ dài của vector  $\text{feature1}$  và  $\text{feature2}$  tương ứng.

Công thức này cho ra một giá trị trong khoảng  $[-1, 1]$ , trong đó:

- 1 đại diện cho độ tương tự hoàn toàn (cùng hướng).
- 0 đại diện cho không có tương đồng.
- -1 đại diện cho độ tương tự hoàn toàn ngược nhau (trái hướng).

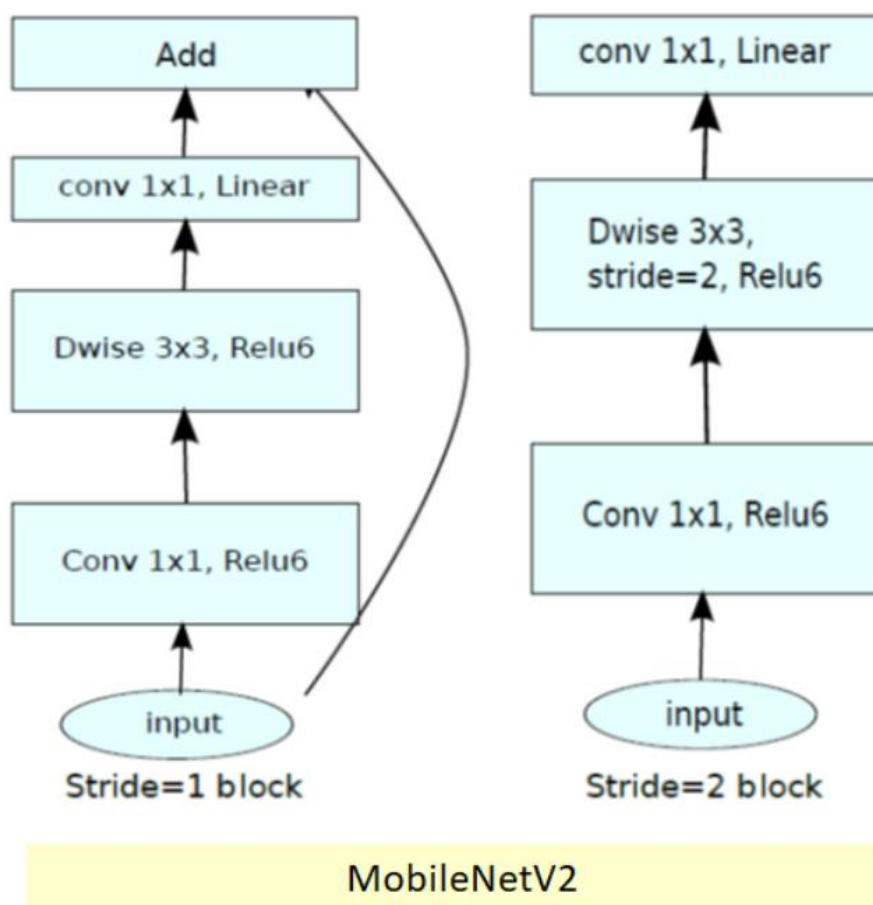


## IV. DEMO HỆ THỐNG VÀ ĐÁNH GIÁ KẾT QUẢ

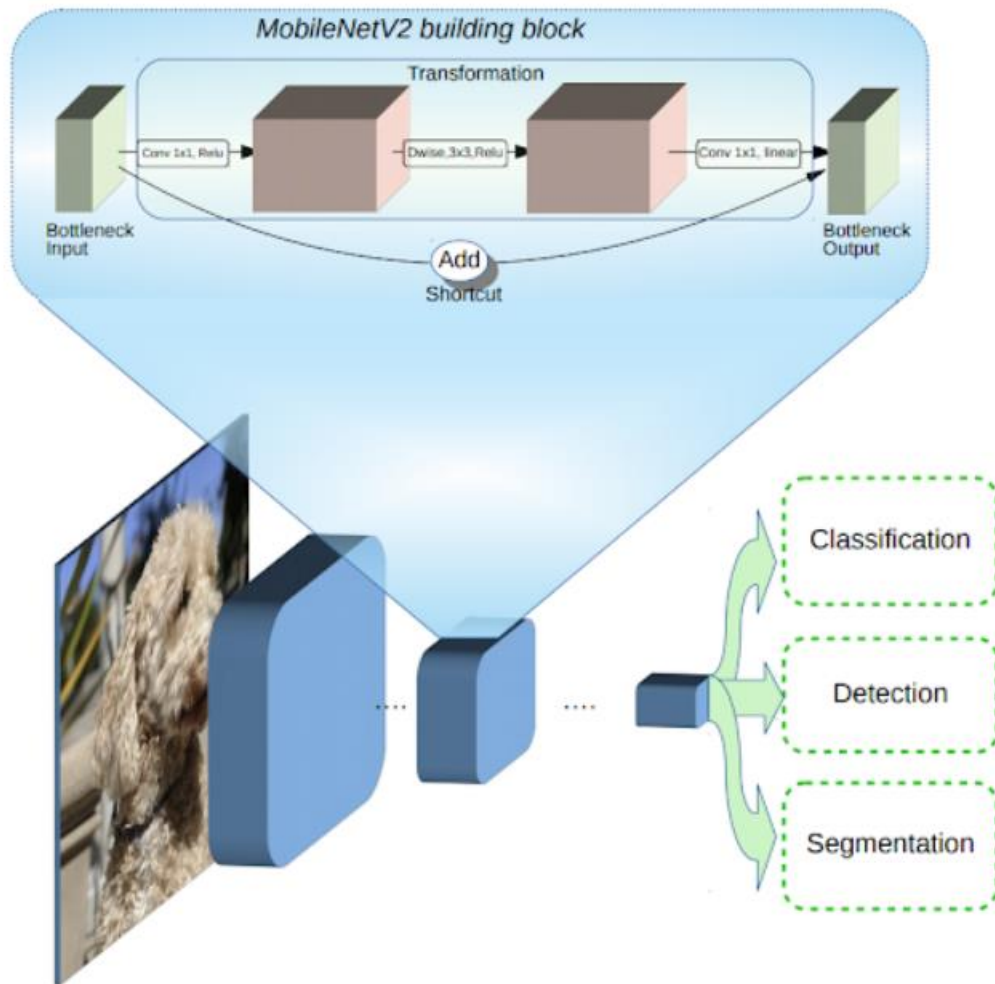
### 1. Giới thiệu mô hình MobileNetV2

MobileNetV2 là một mô hình mạng nơ-ron tích chập (Convolutional Neural Network - CNN) được phát triển bởi Google, được giới thiệu lần đầu trong bài báo "MobileNetV2: Inverted Residuals and Linear Bottlenecks" vào năm 2018. Mô hình này được thiết kế đặc biệt để chạy hiệu quả trên các thiết bị di động và nhúng, nơi tài nguyên tính toán và bộ nhớ bị giới hạn. Dưới đây là một số đặc điểm nổi bật của MobileNetV2:

**Kiến trúc Inverted Residual Block:** Một trong những cải tiến quan trọng của MobileNetV2 so với MobileNetV1 là việc sử dụng các khối dư nghịch đảo (Inverted Residual Blocks). Thay vì sử dụng các khối dư thông thường, MobileNetV2 sử dụng các khối có đầu vào kích thước thấp hơn (bottleneck) và mở rộng không gian đặc trưng trước khi áp dụng tích chập chiều sâu (depthwise convolution). Sau đó, không gian đặc trưng được giảm kích thước trở lại qua một lớp tích chập chuẩn (pointwise convolution). Cấu trúc này giúp giảm số lượng tính toán cần thiết mà vẫn duy trì được hiệu quả của mô hình.



MobileNetV2 sử dụng bottlenecks tuyến tính, thay vì sử dụng hàm kích hoạt phi tuyến (như ReLU) sau các lớp cuối cùng trong các khối nghịch đảo. Điều này giúp duy trì thông tin đặc trưng tốt hơn qua các lớp và cải thiện hiệu quả của mạng.



Cấu trúc bottleneck với  $t=6$  và  $\text{stride}=1$  có sử dụng skip connection

MobileNetV2 được thiết kế để cân bằng giữa độ chính xác và hiệu suất tính toán, giúp nó phù hợp để triển khai trên các thiết bị di động với tài nguyên hạn chế. Nó có thể đạt được kết quả tương đương hoặc tốt hơn so với các mô hình CNN khác nhưng với số lượng tham số và FLOPs (Floating Point Operations) ít hơn.

## 2. Mã nguồn và demo hệ thống

### 2.1 Khai báo thư viện

```
import os
import tkinter as tk
from tkinter import filedialog
from PIL import Image, ImageTk
import numpy as np
from keras.preprocessing import image
from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2, preprocess_input
```

## 2.2 Load mô hình MobileNetV2 được đào tạo trước để trích xuất đặc trưng ảnh

```
feature_extractor = MobileNetV2(weights='imagenet', include_top=False, input_shape=(500, 500, 3))
```

## 2.3 Load đường dẫn đến thư mục lưu trữ ảnh

```
train_image_paths = []
for root, _, files in os.walk("Black2/train"):
    for file in files:
        if file.endswith((".jpg", ".png")):
            train_image_paths.append(os.path.join(root, file))
```

## 2.4 Trích rút đặc trưng của tất cả ảnh trong thư mục ảnh

```
train_features = []
for image_path in train_image_paths:
    img = image.load_img(image_path, target_size=(500, 500))
    img = image.img_to_array(img)
    img = np.expand_dims(img, axis=0)
    img = preprocess_input(img) # Preprocess input according to MobileNetV2 requirements
    features = feature_extractor.predict(img)
    train_features.append(features.flatten())
```

## 2.5 Lưu đặc trưng ảnh đã trích rút trong file

```
np.save('train_features.npy', train_features)
np.save('train_image_paths.npy', train_image_paths)
```

## 2.6 Hàm hiển thị ảnh tương đồng trong của sổ mới

```
def display_similar_images(similar_images, window_title):

    similar_window = tk.Toplevel()
    similar_window.title(window_title)

    for image_path, similarity in similar_images:
        similarity_scalar = similarity.item()
        image_frame = tk.Frame(similar_window)
        image_frame.pack(pady=5)

        label = tk.Label(image_frame, text=f"Similarity: {similarity_scalar:.2f}")
        label.pack()

        img = Image.open(image_path)
        img = img.resize((100, 100))
        img_photo = ImageTk.PhotoImage(img)

        img_label = tk.Label(image_frame, image=img_photo)
        img_label.image = img_photo
        img_label.pack(side=tk.LEFT, padx=5)
```

## 2.7 Hàm tìm và hiển thị 3 ảnh tương đồng

```
def find_and_display_similar_images():

    input_image_path = selected_image_path.get()
    input_image = Image.open(input_image_path)
    input_image = input_image.resize((500, 500))
    input_img_photo = ImageTk.PhotoImage(input_image)

    input_img_label.config(image=input_img_photo)
    input_img_label.image = input_img_photo

    img = image.load_img(input_image_path, target_size=(500, 500))
    img = image.img_to_array(img)
    img = np.expand_dims(img, axis=0)
    img = preprocess_input(img)
    input_feature = feature_extractor.predict(img).flatten()

    similarities = []
    for train_feature, train_image_path in zip(train_features, train_image_paths):
        similarity = np.dot(input_feature, train_feature)
        similarities.append((train_image_path, similarity))

    similarities.sort(key=lambda x: x[1], reverse=True)
    top_similar_images = similarities[:3]

    display_similar_images(top_similar_images, "Top 3 Similar Images")
```

## 2.8 Giao diện app

```
title_label = tk.Label(app, text="Similar Images Finder", font=("Helvetica", 20, "bold"))
title_label.pack(side=tk.TOP, pady=15)
image_label = tk.Label(app)
image_label.pack()

selected_image_path = tk.StringVar()

input_img_label = tk.Label(app)
input_img_label.pack()

result_label = tk.Label(app, text="", font=("Helvetica", 12))
result_label.pack(pady=15)

button_frame = tk.Frame(app)
button_frame.pack(side=tk.BOTTOM, pady=15)

open_button = tk.Button(button_frame, text="Open Image", command=open_file_dialog)
open_button.pack(side=tk.LEFT, padx=5)

find_similar_button = tk.Button(button_frame, text="Find Similar Images", command=find_and_display_similar_images)
find_similar_button.pack(side=tk.LEFT, padx=5)

app.mainloop()
```

### 3. Kết quả demo hệ thống

