

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

**Khoa Công Nghệ Thông Tin**

**BÁO CÁO**

**Môn: Hệ cơ sở dữ liệu đa phương tiện**

**Đề** **tài: Nhận diện quần áo**

**GVHD: TS. Nguyễn Đình Hóa**

**Nhóm môn học: 1**

**Nhóm BTL: 17**

**Thành viên:**

|  |  |  |
| --- | --- | --- |
| 1. | Hoàng Văn Đức | B15DCCN131 |
| 2. | Phạm Thị Thảo | B15DCCN527 |
| 3. | Đỗ Hoàng Nam | B15DCCN373 |
|  |  |  |
|  |  |  |

***LỜI NÓI ĐẦU***

Trong sự phát triển của khoa học công nghệ, thiết bị số và internet, việc truyền tải dữ liệu và thông tin ngày càng nhanh chóng. Với lượng dữ liệu khổng lồ và phong phú, việc tìm hiểu, sàng lọc những dữ liệu thông tin có ích lại càng trở nên quan trọng hơn bao giờ hết.

Trên cơ sở những kiến thức đã học trong môn Cơ sở dữ liệu đa phương tiện, chúng em tìm hiểu thực hiện hóa một chương trình phân loại quần áo từ dữ liệu đã thu thập được dựa trên xử lý ảnh và thuật toán trích rút đặc trưng.

Các nội dung chính bao gồm:

1. Tổng quan về xử lý ảnh.
2. Kỹ thuật trích rút đặc trưng HOG (*Histogram of Oriented Gradient*)
3. Kỹ thuật phân loại KNN ([K-nearest neighbors](https://machinelearningcoban.com/2017/01/08/knn/))
4. Demo chương trình

Do lượng kiến thức còn hạn hẹp nên trong quá trình thực hiện có thể tồn tại những thiếu sót. Chúng em rất mong được thầy/cô và các bạn góp ý để chúng em có thể hoàn thiện hơn.

Chúng em xin chân thành cảm ơn!

# Tổng quan về xử lý ảnh

## Ảnh số là gì?

Ảnh số (digital image) là một thành phần biểu diễn hình ảnh trong hầu hết các thiết bị điện tử hiện nay như máy ảnh, điện thoại, máy tính, các công cụ hỗ trợ có sử dụng ảnh. Để bắt đầu với xử lý ảnh, chúng ta phải nắm được các kiến thức cơ bản về ảnh số và cách biểu diễn ảnh số trong máy tính.

**Định nghĩa***:* ***Ảnh số****thực tế là biểu diễn số học của hình ảnh trong máy tính, thường là biểu diễn nhị phân. Chúng là một tập hợp hữu hạn các giá trị số, gọi là điểm ảnh (pixel - picture element). Thông thường một hình ảnh được chia thành các hàng và cột chứa điểm ảnh. Điểm ảnh là thành phần bé nhất biểu diễn ảnh, có giá trị số biểu diễn màu sắc, độ sáng… của một thành phần trong bức ảnh.*



Hình 1.1: Ví dụ về ảnh số

## Một số không gian màu trong ảnh số

Thông thường các ảnh số sẽ có một mức giá trị nhất định cho các pixel ảnh. Giá trị này sẽ ảnh hưởng đến chất lượng của ảnh, và các đặc tính của ảnh. Việc biểu diễn ảnh ở các không gian màu khác nhau sẽ cho ra các ảnh đầu ra khác nhau. Phục vụ cho quá trình xử lý ảnh hay trải nghiệm ảnh.

*Dưới đây chúng em sẽ phân tích không gian màu dựa trên ảnh 8bit cho một giá trị. Như vậy một giá trị sẽ nằm trong ngưỡng là 0 -> 255 ( 20 -> 27).*

### Không gian màu RGB

RGB là không gian màu phổ biến dùng trong máy tính, máy ảnh, điện thoại và nhiều thiết bị kĩ thuật số khác. 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.



Hình 1.2: Ảnh RGB

Với mối ảnh được biểu diễn dựa trên không gian màu RGB thì 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 biểu diễn 24bit, số lượng màu tối đa có thể biểu diễn sẽ là:

**255×255×255 = 16581375**

### Không gian màu Gray

Tương tự như không gian màu RGB. Mỗi kênh màu của ảnh Gray cũng sử dụng 8 bit để biểu diễn giá trị. Tuy nhiên đối với ảnh Gray thì mỗi pixel chỉ sử dụng một giá trị 0-255 để biểu thị thay vì 3 giá trị cho 3 kênh như ảnh RGB, Do đó. ảnh Gray sẽ không hiển thị được màu thay vào đó sẽ là ảnh mức xám.(ảnh đen trắng)



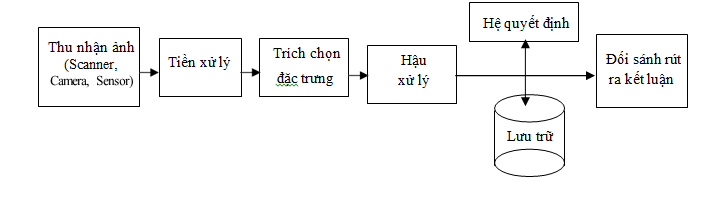
Hình 1.3: Ảnh Gray

## Xử lý ảnh là gì?

Xử lý ảnh là một phân ngành trong xử lý số tín hiệu với tín hiệu xử lý là ảnh. Đây là một phân ngành khoa học mới rất phát triển trong những năm gần đây. Xử lý ảnh gồm 4 lĩnh vực chính: xử lý nâng cao chất lượng ảnh, nhận dạng ảnh, nén ảnh và truy vấn ảnh. Sự phát triển của xử lý ảnh đem lại rất nhiều lợi ích cho cuộc sống của con người.

Quá trình xử lý ảnh được xem như là quá trình thao tác ảnh đầu vào nhằm cho ra kết quả mong muốn. Kết quả đầu ra của một quá trình xử lý ảnh có thể là một ảnh “tốt hơn” hoặc một kết luận.

Sơ đồ tổng quát của một hệ thống xử lý ảnh:



* *Thu nhận ảnh:*

Ảnh được thu từ nhiều nguồn khác nhau:máy ảnh, máy quay phim, máy quét, ảnh vệ tinh…

Mục đích: biến đổi thông tin hình ảnh về các cấu trúc được lưu trữ trong máy tính, có thể hiển thị ra các thiết bị ngoại vi như là máy in, màn hình…

Gồm hai tiến trình:   
+ Biến đổi năng lượng quang học thành năng lượng điện.   
+ Tổng hợp năng lượng điện thành ảnh hoặc ma trận số.

* *Tiền xử lý:*

Là quá trình sử dụng các kỹ thuật xử lý ảnh để làm ảnh tốt lên theo mục đích sử dụng.

Mục đích:  
+ Điều chỉnh độ chiếu sáng để khắc phục hậu quả của việc chiếu sáng không đều.   
+ Giảm nhỏ thành phần nhiễu của ảnh tức là các đối tượng xuất hiện ngoài ý muốn.   
+ Hiệu chỉnh giá trị độ sáng giữa nền và đối tượng.   
+ Chuẩn hoá độ lớn, màu, dạng của ảnh.   
+ Điều chỉnh bộ lọc để khuyếch đại và nén các tần số.

* *Trích chọn đặc trưng:*

Việc chọn các tính chất để thể hiện ảnh gọi là trích chọn đặc trưng (Feature Selection) gắn với việc tách các đặc tính của ảnh dưới dạng các thông tin định lượng hoặc làm cơ sở để phân biệt lớp đối tượng này với đối tượng khác trong phạm vi ảnh nhận được.

*Ví dụ: trong nhận dạng ký tự trên phong bì thư, chúng ta miêu tả các đặc trưng của từng ký tự giúp phân biệt ký tự này với ký tự khác.*

* *Hậu xử lý:*

Tùy từng bài toán mà phần hậu xử lý giúp khôi phục lại một số thông tin cần thiết của ảnh đã bị mất sau quá trình xử lý trước đó.

* *Hệ quyết định, lưu trữ:*

Quá trình này thường thu được bằng cách so sánh với mẫu chuẩn đã được học (hoặc lưu) từ trước. Nội suy là phán đoán theo ý nghĩa trên cơ sở nhận dạng.  
  
*Ví dụ: một loạt chữ số và nét gạch ngang trên phong bì thư có thể được nội suy thành mã điện thoại. Có nhiều cách phân loai ảnh khác nhau về ảnh. Theo lý thuyết về nhận dạng, các mô hình toán học về ảnh được phân theo hai loại nhận dạng ảnh cơ bản:   
– Nhận dạng theo tham số.   
– Nhận dạng theo cấu trúc.*

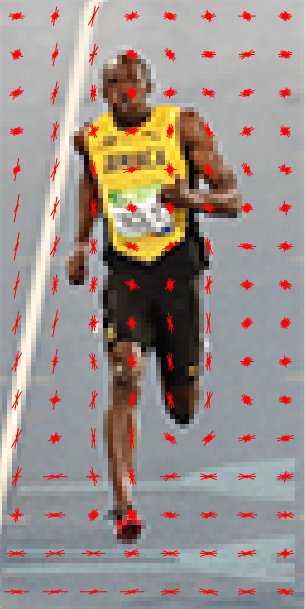
Một số đối tượng nhận dạng khá phổ biến hiện nay đang được áp dụng trong khoa học và công nghệ là: nhận dạng ký tự (chữ in, chữ viết tay, chữ ký điện tử), nhận dạng văn bản (Text), nhận dạng vân tay, nhận dạng mã vạch, nhận dạng mặt người, hay nhận dạng quần áo như trong bài trình bày này của chúng em…

* *Đối sánh và rút ra kết luận:*

Sau quá trình quyết định, ảnh được đưa ra kết quả của việc xử lý như phân loại, đo đạt, …

# Kỹ thuật trích rút đặc trưng HOG

HOG là viết tắt của Histogram of Oriented Gradient - một loại “feature descriptor”. Mục đích của “featura descriptor” là trừu tượng hóa đối tượng bằng cách trích xuất ra những đặc trưng của đối tượng đó và bỏ đi những thông tin không hữu ích. Vì vậy, HOG được sử dụng chủ yếu để mô tả hình dạng và sự xuất hiện của một đối tượng trong ảnh.



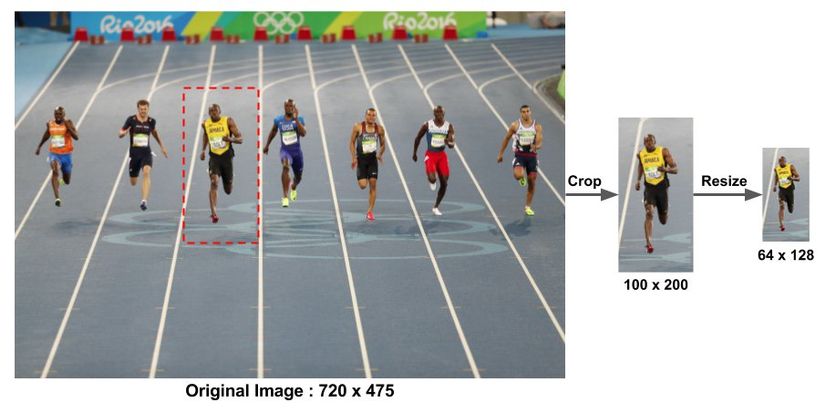
Bản chất của phương pháp HOG là sử dụng thông tin về sự phân bố của các cường độ gradient (intensity gradient) hoặc của hướng biên (edge directins) để mô tả các đối tượng cục bộ trong ảnh. Các toán tử HOG được cài đặt bằng cách chia nhỏ một bức ảnh thành các vùng con, được gọi là “tế bào” (cells) và với mỗi cell, ta sẽ tính toán một histogram về các hướng của gradients cho các điểm nằm trong cell. Ghép các histogram lại với nhau ta sẽ có một biểu diễn cho bức ảnh ban đầu. Để tăng cường hiệu năng nhận dạng, các histogram cục bộ có thể được chuẩn hóa về độ tương phản bằng cách tính một ngưỡng cường độ trong một vùng lớn hơn cell, gọi là các khối (blocks) và sử dụng giá trị ngưỡng đó để chuẩn hóa tất cả các cell trong khối. Kết quả sau bước chuẩn hóa sẽ là một vector đặc trưng có tính bất biến cao hơn đối với các thay đổi về điều kiện ánh sáng.

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

1. Tiền xử lý
2. Tính gradient
3. Tính vector đặc trưng cho từng ô (cells)
4. Chuẩn hóa khối (blocks)
5. Tính toán vector HOG

## Tiền xử lý

Ở bước tiền xử lý , thực hiện việc chia đều hình ảnh thành các khối, ô và tính toán đặc trưng ở các bước tiếp theo, chúng ta cần resize kích thước tất cả các hình ảnh trong tập dữ liệu về một kích thước chung giúp dễ dàng thực hiện các chuẩn hóa blocks.



*Ở phần trình bày này, chúng em sẽ để kích thước ảnh là 64 x 128.*

## Tính Gradient

Gradient 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 Ox và Oy. Trong đó, 2 hướng tương ứng đó là:

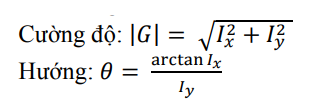
**Dx = [-1 0 1] và Dy = [1 0 -1]T**

*Với T là phép toán chuyển vị ma trận.*

Và nếu có một ảnh input là **I**, ta sẽ có 2 ảnh đạo hàm riêng theo 2 hướng đó, theo công thức:

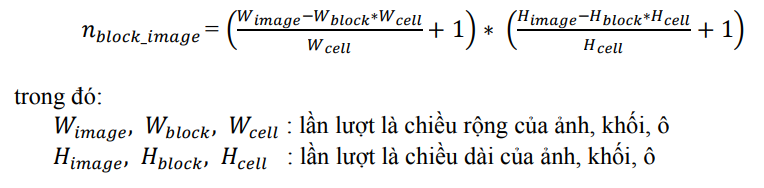
**Ix = I \* Dx và Iy = I \* Dy**

Khi đó, có thể tính được Gradient bao gồm hai thành phần cường độ(Gradient Magnitude) và hướng(Gradient Derection) theo công thức **(\*)**:



## Tính Vector đặc trưng cho từng ô (cells)

Để tính toán vector đặc trưng cho từng ô (cell), cần chia hình ảnh thành các block, mỗi block lại chia đều thành các cell. Để xác định được số block, sau đó sẽ sử dụng công thức:

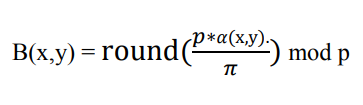


Sau khi xác định số block và kích thước mỗi block, cell, để tính toán vector đặc trưng cho từng cell, chúng ta cần:

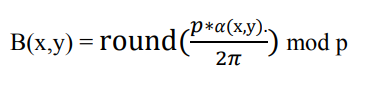
* Chia không gian hướng thành p bin(số chiều vector đặc trưng của ô).
* Rời rạc hóa góc hướng nghiêng tại mỗi điểm ảnh vào trong các bin.

Giả sử góc hướng nghiêng tại pixel ở vị trí (x,y) có độ lớn là alpha(x,y)

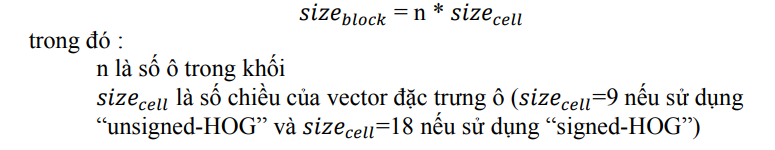
* Trường hợp rời rạc hóa unsigned-HOG với p=9:



* Trường hợp rời rạc hóa signed-HOG với p=18:



Giá trị bin được định lượng bởi tổng cường độ biến thiên của các pixels thuộc về bin đó. Sau khi tính toán đặc trưng ô, ta sẽ nối các vector đặc trưng ô để thu được vector đặc trưng khối. Số chiều vector đặc trưng khối tính theo công thức:



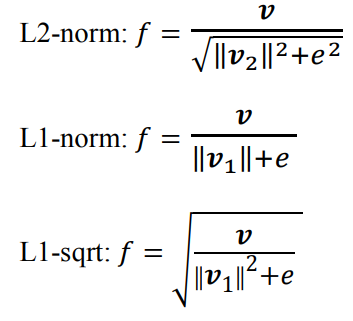
Đầu mút là các giá trị chia hết cho độ rộng của một bin (chẳng hạn 0, 20, 40,… là những đầu mút bin). Trong trường hợp độ lớn phương gradients không rơi vào các đầu mút, ta sẽ sử dụng linear interpolation để phân chia độ lớn gradient về 2 bins liền kề mà giá trị phương gradient rơi vào.

Ví dụ: giá trị phương gradient bằng  *x* ghép cặp với độ lớn gradient bằng y. *x*∈[*x0,x1*] tức là phương gradients rơi vào khoảng giữa bin thứ (l−1) và bin thứ l.

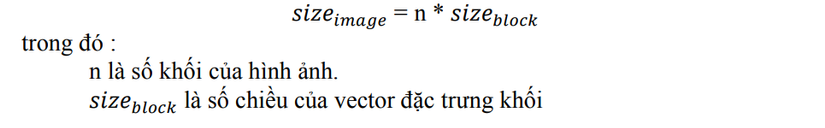
## Chuẩn hóa khối (Blocks)

Để tăng cường hiệu năng nhận dạng, các histogram cục bộ sẽ được chuẩn hóa về độ tương phản bằng cách tính một ngưỡng cường độ trong một khối và sử dụng giá trị đó để chuẩn hóa tất cả các ô trong khối. Kết quả sau bước chuẩn hóa sẽ là một vector đặc trưng có tính bất biến cao hơn đối với các thay đổi về điều kiện ánh sáng.

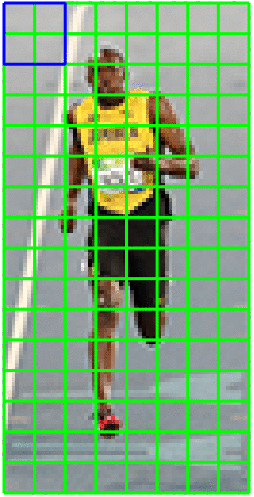
Có nhiều phương pháp có thể được dùng để chuẩn hóa khối. Gọi v là vector cần chuẩn hóa chứa tất cả các histogram của mội khối. ‖v(k)‖ là giá trị chuẩn hóa của v theo các chuẩn k=1, 3 và e là một hằng số nhỏ. Khi đó, các giá trị chuẩn hóa có thể tính bằng một trong những công thức sau:



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



## Tính vector đặc trưng HOG



* Với mỗi hình ảnh kích thước 64x128, chia thành các block 16x16 chồng nhau, sẽ có 7 block ngang và 15 block dọc, nên sẽ có 7x15 = 105 blocks.
* Mỗi block gồm 4 cell. Khi áp dụng biểu đồ 9-bin cho mỗi cell, mỗi block sẽ được đại diện bởi một vector có kích thước 36x1.
* Vì vậy, khi nối tất cả các vector trong một block lại với nhau, ta sẽ thu được vector đặc trưng HOG của ảnh có kích thước 105x36x1 = 3780x1.

# Kỹ thuật phân loại KNN

## Sơ lược về 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.

Với KNN, trong bài toán Classification, label của một điểm dữ liệu mới (hay kết quả của câu hỏi trong bài thi) được suy ra trực tiếp từ K điểm dữ liệu gần nhất trong training set. Label của một test data có thể được quyết định bằng major voting (bầu chọn theo số phiếu) giữa các điểm gần nhất, hoặc nó có thể được suy ra bằng cách đánh trọng số khác nhau cho mỗi trong các điểm gần nhất đó rồi suy ra label.

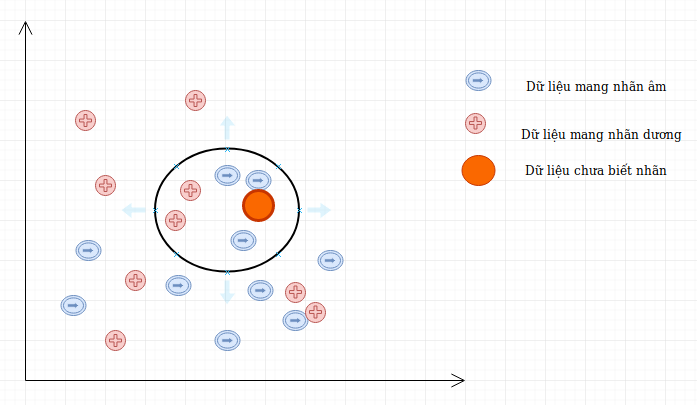
Kết quả đầu ra của một điểm dữ liệu sẽ bằng chính đầu ra của điểm dữ liệu đã biết gần nhất (trong trường hợp K=1), hoặc là trung bình có trọng số của đầu ra của những điểm gần nhất, hoặc bằng một mối quan hệ dựa trên khoảng cách tới các điểm gần nhất đó.

KNN là thuật toán đi tìm đầu ra của một điểm dữ liệu mới bằng cách chỉ dựa trên thông tin của K điểm dữ liệu trong training set gần nó nhất (K-lân cận), không quan tâm đến việc có một vài điểm dữ liệu trong những điểm gần nhất này là nhiễu.

## KNN hoạt động như thế nào?

*Đặt vấn đề:*

Giả sử ta có D là tập các dữ liệu đã được phân loại thành 2 nhãn (+) và (-) được biểu diễn trên trục tọa độ như hình vẽ và một điểm dữ liệu mới A chưa biết nhãn. Vậy làm cách nào để chúng ta có thể xác định được nhãn của A là (+) hay (-)?



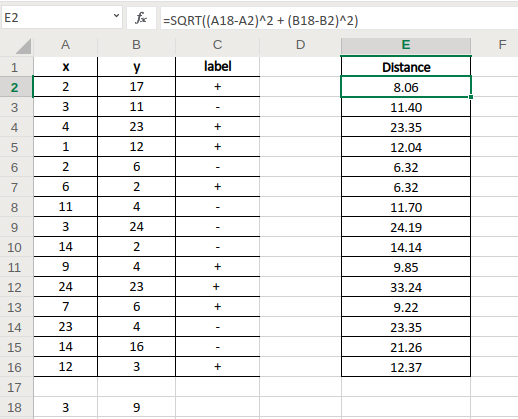
      KNN sẽ so sánh tất cả các đặc điểm của dữ liệu A với tất cả tập dữ liệu học đã được gắn nhãn và xem nó giống cái nào nhất, nếu dữ liệu (đặc điểm) của A giống với dữ liệu của điểm mang nhãn (+) thì điểm A mang nhãn (+), nếu dữ liệu A giống với dữ liệu nhãn (-) hơn thì nó mang nhãn (-).  
      Trong trường hợp của KNN, thực tế nó không so sánh dữ liệu mới (không được phân lớp) với tất cả các dữ liệu khác, thực tế nó thực hiện một phép tính toán học để đo khoảng cách giữa dữ liệu mới với tất cả các điểm trong tập dữ liệu học D để thực hiện phân lớp.

## Xây dựng KNN phân lớp

Để xây dựng một mô hình KNN ta cần 6 bước như sau:

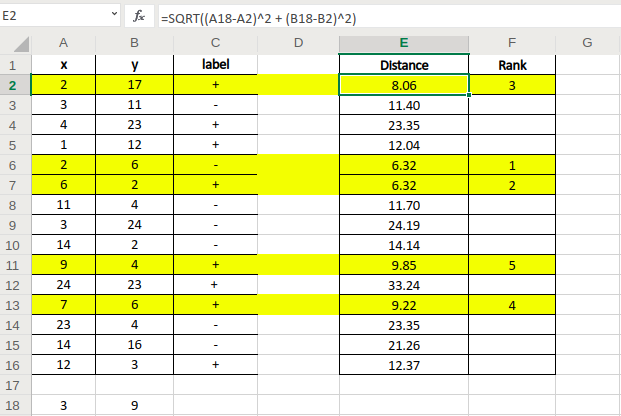
1. Khi có D là tập các điểm dữ liệu đã được gắn nhãn và A là dữ liệu chưa được phân loại.
2. Đo khoảng cách từ dữ liệu mới A đến tất cả các dữ liệu khác đã được phân loại trong D.
3. Chọn K (K là tham số mà bạn định nghĩa) khoảng cách nhỏ nhất.
4. Kiểm tra danh sách các lớp có khoảng cách ngắn nhất và 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 mà bạn đã nhận được ở bước 5.

*Một ví dụ:*



Giả sử chúng em có tập dữ liệu D có gắn nhãn gồm 15 điểm như trên ảnh.

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



## Đánh giá thuật toán

Các thuật toán phân loại thương có những ưu điểm và nhược điểm riêng. Với kiến thức của bản thân, chúng em xin đưa ra một vài đánh giá về thuật toán KNN này về ưu điểm và nhược điểm như sau:

* ***Ưu điểm:***
* Thuật toán đơn giản, dễ dàng triển khai.
* Độ phức tạp tính toán nhỏ.
* Xử lý tốt với tập dữ liệu nhiễu
* ***Nhược điểm:***
* Với K nhỏ dễ gặp nhiễu dẫn tới kết quả đưa ra không chính xác
* Cần nhiều thời gian để thực hiện do phải tính toán khoảng cách với tất cả các đối tượng trong tập dữ liệu.
* Cần chuyển đổi kiểu dữ liệu thành các yếu tố định tính.

# Demo chương trình

## Triển khai chương trình

Ở phần demo này, để giúp tốc độ chương trình chạy nhanh, mà vẫn có độ chính xác tốt, chúng em sẽ sử dụng hình ảnh đầu vào với kích thước là 128 x 128.

Do kích thước ảnh khá nhỏ nên chúng e sẽ chia thành các cells là 8 x 8 để đảm bảo có cell đó có chứa thông tin, đặc trưng của vật.

Chi tiết chương trình demo:

Đầu tiên chúng em xây dựng hàm tính toán HOG feature gồm các hàm con như sau:

Hàm **caculate\_gradients** có chức năng tính toán gradient của hình ảnh đầu vào và trả lại 2 mảng giá trị tương ứng với cường độ và hướng của điểm ảnh trong ảnh.

def calculate\_gradients(*img*):

        # tinh kich thuoc hinh anh va khai bao mang chua cuong do va huong

        h, w = img.shape[0], img.shape[1]

        magnitude = np.zeros((h, w))

        direction = np.zeros((h, w))

        img = img.astype("float32")

        # lap tinh toan cuong do va huong cua tung pixel

        for y in range(1, h - 1):

            for x in range(1, w - 1):

                pCu = img[y, x]

                # compute x direction

                pBx = img[y, x - 1]

                pFx = img[y, x + 1]

                Gx = pFx - pBx

                # compute y direction

                pBy = img[y - 1, x]

                pFy = img[y + 1, x]

                Gy = pFy - pBy

                # Magnitude (Gradient) - cuong do

                G = np.sqrt(Gx \* Gx + Gy \* Gy)

                magnitude[y, x] = G

                # Direction - huong

                D = 0 if Gy == 0 else np.arctan(Gx / Gy)

                D = D \* 180.0 / np.pi

                D = D if D >= 0 else D + 180

                direction[y, x] = D

        return magnitude, direction

Tiếp theo chúng em xây dựng hàm Vote các giá trị cường độ và hướng cho các cells . sau đó chúng e sẽ chuẩn hóa theo 1, 1 block cho vector đặc trưng và tính trung bình chuẩn hóa về 8 bit.

def calculate\_hispercell(*magnitude*, *direction*, *bin\_num*):

        # chia chia bin cua cells

        if bin\_num == 9:

            bins = (0, 20, 40, 60, 80, 100, 120, 140, 160)

            vote = [0, 0, 0, 0, 0, 0, 0, 0, 0]

            bin\_range = 20

        elif bin\_num == 18:

            bins = (0, 10, 20, 30, 40, 50, 60, 70, 80, 90,

                    100, 110, 120, 130, 140, 150, 160, 170)

            vote = [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

            bin\_range = 10

        # lap va vote vao tung cells

        for x in range(magnitude.shape[1]):

            for y in range(magnitude.shape[0]):

                g = magnitude[x, y]

                d = direction[x, y]

                il = d // bin\_range

                ih = 0 if il == (bin\_num - 1) else il + 1

                divia = d - bins[*int*(il)]

                vote[*int*(il)] += (divia / bin\_range) \* g

                vote[*int*(ih)] += ((bin\_range - divia) / bin\_range) \* g

        # gian hoa tung cells roi tra ve vector cua hinh anh voi cac cells

        vote = np.asarray(vote) / 9

        vote = vote.astype("uint8")

        return vote

Cuối cùng để cho dễ thực hiện thao tác chúng e xây dựng một hàm **hog\_feature** để thực hiện từng bước của việc tính toán đặc trưng và trả về đặc trưng của ảnh:

def hog\_feature(*img*, *cells*=(8, 8), *blocks* =(1,1),  *bin\_num*=9):

        magnitude\_arr, direction\_arr = HOG.calculate\_gradients(img)

        max\_w = magnitude\_arr.shape[1] // cells[0]

        max\_h = magnitude\_arr.shape[0] // cells[1]

        cells\_array = []

        # tinh toan feature theo tung cells

        for y in range(max\_h):

            for x in range(max\_w):

                mag = magnitude\_arr[y\*cells[1]:(y + 1) \* cells[1], x \* cells[0]:(x + 1) \* cells[0]]

                direct = direction\_arr[y\*cells[1]:(y + 1) \* cells[1], x \* cells[0]:(x + 1) \* cells[0]]

                val = HOG.calculate\_hispercell(mag, direct, bin\_num)

                cells\_array.append(val)

        cells\_array = np.asarray(cells\_array)

        # tinh toan feature theo norm trung binh

        cells\_array = np.reshape(cells\_array.shape[0] \* cells\_array.shape[1])

        return cells\_array

Tiếp đến chúng em xây dựng bộ dự đoán KNN bằng 2 hàm cơ bản:

Một hàm là **caculate\_distance()** để tính khoảng cách đến các điểm hàng xóm.

def caculate\_distance(*p1*, *p2*):

        p1 = p1.astype("float32")

        p2 = p2.astype("float32")

        sum\_dis = 0

        for i in range(len(p1)):

            sum\_dis += (p1[i] - p2[i]) \* (p1[i] - p2[i])

        return np.sqrt(sum\_dis)

Hàm predict để thực hiện tính toán trên tập dữ liệu và đưa ra kết quả dự đoán:

def predict(*x*, *y*, *k*, *f\_predict*):

        # x la feature anh, y label tuong ung cua feature anh, k la so hang xom gan nhat,

        # f\_predict la freature cua anh can du doan

        array\_distance = []

        # tinh toan khoang cach den cac diem

        for i in range(len(x)):

            d = KNN.caculate\_distance(x[i], f\_predict)

            array\_distance.append(d)

        array\_distance\_sorted = sorted(array\_distance, *key* = lambda *x* : x, *reverse*=False)

        label\_nn = []

        for i in range(k):

            for \_i in range(len(array\_distance)):

                if(array\_distance\_sorted[i] == array\_distance[\_i]):

                    label\_nn.append(y[\_i])

        print(label\_nn)

        label\_predict = 0

        count\_max = 0

        for lb in label\_nn:

            count = label\_nn.count(lb)

            if(count > count\_max):

                count\_max = count

                label\_predict = lb

        return label\_predict

Để thực hiện chương trình chúng e chia tập dataset ra làm 2:

* 1. Tập so sánh dùng để làm mẫu so sánh( tập train)
  2. Tập dự đoán không nằm trong tập so sánh



Đối với áo chúng e chia làm 3 tập là **Áo cộc**, **Áo sơ mi**, và **Váy**:

Trong đó có 21 Mẫu so sánh áo cộc, 20 mẫu so sánh áo sơ mi và 18 mẫu so sánh váy.

Đối với quẩn chúng em chia làm 2 tập là **Quần đùi** và **Quần dài**:

Trong đó có 20 Mẫu so sánh quần đùi, và 20 mẫu so sánh quần dài.

Để thử nghiệm tính hiệu quả của chương trình chúng e lấy 2 mẫu/ 1 tập để tiến hành thử nghiệm dự đoán.

Kết quả chạy của chương trình như sau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **K** | **Phân loại áo** | **Tỷ lệ phân loại áo** | **Phân loại quần** | **Tỷ lệ phân loại quần** |
| 3 | 6/6 | 100% | 4/4 | 100% |
| 9 | 5/6 | 83.3% | 4/4 | 100 % |
| 11 | 5/6 | 83.3% | 4/4 | 100 % |

## Kết luận

Thuật toán trích rút đặc trưng HOG và thuật toán phân loại KNN cho kết quả chấp nhận được. Tuy nhiên nó còn phụ thuộc vào việc chọn số láng giềng ( hệ số K), vậy để cái thiện độ chính xác và các thông số cần chọn, ta có thể tìm hiểu và triển khai thêm một số thuật toán trích rút khác như SIFT, SURF, …hay công cụ phân loại khác như SVM, cây quyết định, CNN (Deep learning), …

***Xin chân thành cảm ơn thầy và các bạn!***