

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



BÁO CÁO BÀI TẬP LỚN

HỆ CƠ SỞ DỮ LIỆU ĐA PHƯƠNG TIỆN

THÀNH VIÊN NHÓM 08

HỌ VÀ TÊN : Nguyễn Như Tuấn MSV : B17DCCN659

HỌ VÀ TÊN : Nguyễn Thị Dung MSV : B17DCCN155

HỌ VÀ TÊN : Phạm Quốc Đạt MSV : B17DCCN118

GIẢNG VIÊN HƯỚNG DẪN: TS. Nguyễn Đình Hóa

Mục Lục

Câu 1: Đặc điểm của kho ảnh:	4
Câu 2: Kỹ thuật xử lý và phân loại ảnh hoa đang hiện hành	5
I. Kỹ thuật xử lý ảnh	5
1.Quá trình xử lý ảnh	5
Lọc trung bình - Median Filter	5
2.Đặc trưng của ảnh	6
3.Ảnh và biểu diễn ảnh:	6
II. Các phương pháp xử lý ảnh hiện nay	7
III. Kỹ thuật phân loại ảnh	8
1.Tìm hiểu về SVM	8
2.Tìm Hiểu về KNN	16
IV. Phương pháp rút trích đặc trưng hình ảnh HOG	18
Câu 3. Xây dựng hệ thống	23
V. Xây dựng hệ thống	23
1. Sơ đồ khối	23
2. Quá trình thực hiện	23
1. Bước 1: Tiền xử lý	23
2. Bước 2: Trích rút đặc trưng	24
3.Bước 3: Huấn luyện mô hình (Phân loại ảnh)	29
4. Bước 4: Nhận dạng ảnh	32
4.1 Kết quả train và test:	32
4.2 Kết quả tìm kiếm hình ảnh từ 1 hình ảnh:	32
4.3 Code	32

Yêu Cầu

1. Hãy sưu tầm ít nhất 100 bức ảnh về ít nhất 15 loại hoa khác nhau mỗi ảnh chỉ gồm 1 bông hoa, các bức ảnh đều có cùng kích thước.
2. Tìm hiểu các kỹ thuật xử lý và phân loại ảnh hoa đang hiện hành
3. Xây dựng hệ thống nhận dạng ảnh hoa
 - Đầu vào: 1 bông hoa thuộc loại hoa đã có và chưa có.
 - Đầu ra: nhãn của hoa
- a. Sơ đồ khối
Quy trình thực hiện
- b. Trình bày thuộc tính được sử dụng để nhận dạng nhãn và
Kỹ thuật để trích rút các thuộc tính đó
- c. Cách lưu trữ các thuộc tính ảnh hoa
Các nhận dạng ảnh hoa dựa trên các thuộc tính đó
4. Demo và đánh giá kq đạt dc

Câu 1: Đặc điểm của kho ảnh:

- Có khoảng 200 ảnh
- 15 loại hoa, chia vào 15 folder khác nhau.
- Mỗi ảnh chỉ có 1 bông hoa.
- Kích thước 64*64
- Mỗi loại hoa thì có nhiều hình dạng và màu sắc khác nhau, có ảnh chụp gần và ảnh chụp xa, mỗi loại có nhiều màu sắc.

Câu 2: Kỹ thuật xử lý và phân loại ảnh hoa đang hiện hành

I. Kỹ thuật xử lý ảnh

1. Quá trình xử lý ảnh

- Thu nhận ảnh:
Đặc điểm của kho ảnh:
 - Có bao nhiêu ảnh? 100
 - 15 loại hoa, chia vào 15 folder khác nhau.
 - Mỗi ảnh chỉ có 1 bông hoa.
 - Kích thước bao nhiêu?
 - Tại sao lấy kích thước này?
 - Mỗi loại hoa thì có nhiều hình dạng và màu sắc khác nhau, có ảnh chụp gần và ảnh chụp xa, mỗi loại có nhiều màu sắc.
- Tiền xử lý
 - resize ảnh (64x64) , đổi đuôi về jpg
 - ❖ Lọc trung bình - Median Filter
 - Ý tưởng
 - Với lọc trung bình, mỗi điểm ảnh (Pixel) được thay thế bằng trung bình trọng số của các điểm trong vùng lân cận.

- Giả sử rằng có 1 ma trận lọc (Kernel) (3x3) quét qua từng điểm ảnh của ảnh đầu vào I_{src} . Tại vị trí mỗi điểm ảnh lấy giá trị của các điểm ảnh tương ứng trong vùng (3x3) của ảnh gốc đặt vào ma trận lọc (Kernel). Giá trị điểm ảnh của ảnh đầu ra I_{dst} là giá trị trung bình của tất cả các điểm trong ảnh trong ma trận lọc (Kernel).

- **Thuật toán**

- 1 ảnh đầu vào với $I(x,y)$ là giá trị điểm ảnh tại 1 điểm (x,y) và 1 ngưỡng θ .

- Bước 1: Tính tổng các thành phần trong ma trận lọc (Kernel).
- Bước 2: Chia lấy trung bình của tổng các thành phần trong ma trận được tính ở trên với số lượng các phần tử của cửa sổ lọc ra 1 giá trị $I_{tb}(x, y)$.
- Bước 3: Hiệu chỉnh:
 - o Nếu $I(x,y) - I_{tb}(x,y) > \theta$ thì $I(x,y) = I_{tb}(x,y)$.
 - o Nếu $I(x,y) - I_{tb}(x,y) \leq \theta$ thì $I(x,y) = I(x,y)$.

- **Chú ý**

- θ là 1 giá trị cho trước và có thể có hoặc không tùy thuộc vào mục đích.

- **Tác dụng**

- Trong lọc trung bình, thường ưu tiên cho các hướng để bảo vệ biên của ảnh khỏi bị mờ khi làm trơn ảnh. Các kiểu ma trận lọc (Kernel) được sử dụng tùy theo các trường hợp khác nhau. Các bộ lọc trên là bộ lọc tuyến tính theo nghĩa là điểm ảnh ở tâm cửa sổ sẽ được thay bởi tổ hợp các điểm lân cận chập với ma trận lọc (Kernel).

2. Đặc trưng của ảnh

- Đặc trưng màu sắc
- Đặc trưng kết cấu
- Đặc trưng hình dạng
- Đặc trưng cục bộ bất biến

3. Ảnh và biểu diễn ảnh:

- Ảnh trong thực tế là một ảnh liên tục cả về không gian và giá trị độ sáng. Để có thể xử lý ảnh bằng máy tính thì cần thiết phải tiến hành số hóa ảnh
- Quá trình số hóa biến đổi các tín hiệu liên tục sang tín hiệu rời rạc thông qua quá trình lấy mẫu (rời rạc hóa về không gian) và lượng tử hóa các thành phần giá trị mà về nguyên tắc bằng mắt thường không thể phân biệt được hai điểm liên kề nhau
- màn hình khụng liên tục mà gồm các điểm nhỏ, gọi là pixel. Mỗi pixel gồm một tập tọa độ (x, y) và màu.
- Ảnh có thể được biểu diễn theo một trong hai mô hình: mô hình Vector hoặc mô hình Raster
 - o Mô hình Vector: Ngoài mục đích tiết kiệm không gian lưu trữ, dễ dàng hiển thị và in ấn, các ảnh biểu diễn theo mô hình vector cũng có điểm dễ dàng lựa chọn, sao cho, di chuyển, tìm kiếm
 - o Mô hình Raster: Ảnh được biểu diễn dưới dạng ma trận các điểm ảnh
 - mỗi điểm ảnh có thể được biểu diễn bởi một hay nhiều bit
 - Mô hình Raster thuận lợi cho việc thu nhận, hiển thị và in ấn

II. Các phương pháp xử lý ảnh hiện nay

- **Nhiều xạ không đẳng hướng:** thường được gọi là khuếch tán Perona-Malik. Phương pháp này giúp giảm nhiễu hình ảnh mà không cần loại bỏ các phần quan trọng của hình ảnh.
- **Lọc tuyến tính:** là một kỹ thuật xử lý hình ảnh kỹ thuật số khác. Nó dùng để chỉ các tín hiệu đầu vào biến đổi theo thời gian. Việc này tạo tạo ra các tín hiệu đầu ra phụ thuộc vào ràng buộc của tuyến tính.

- **Mạng nơron:** là các mô hình tính toán được sử dụng rộng rãi trong học máy. Chúng được dùng để giải quyết các nhiệm vụ khác nhau.
- **Pixelation:** là việc chuyển hình ảnh đã in thành những hình ảnh được số hóa (chẳng hạn như GIF).
- **Phân tích thành phần chính:** một kỹ thuật xử lý hình ảnh kỹ thuật số. Nó được sử dụng để trích xuất tính năng.
- Một phần phương trình vi phân, giúp khử nhiễu hình ảnh.
- **Các mô hình Markov ẩn:** một kỹ thuật được sử dụng để phân tích hình ảnh theo hai chiều (2D).
- **Wavelets:** viết tắt của một hàm toán học được sử dụng trong nén hình ảnh.
- **Bản đồ tự tổ chức:** kỹ thuật xử lý hình ảnh kỹ thuật số để phân loại hình ảnh thành một số layer (lớp).
- **Phân tích thành phần độc lập:** phân tách tín hiệu đa biến, tính toán thành các thành phần phụ cộng.

III. Kỹ thuật phân loại ảnh

1. Tìm hiểu về SVM

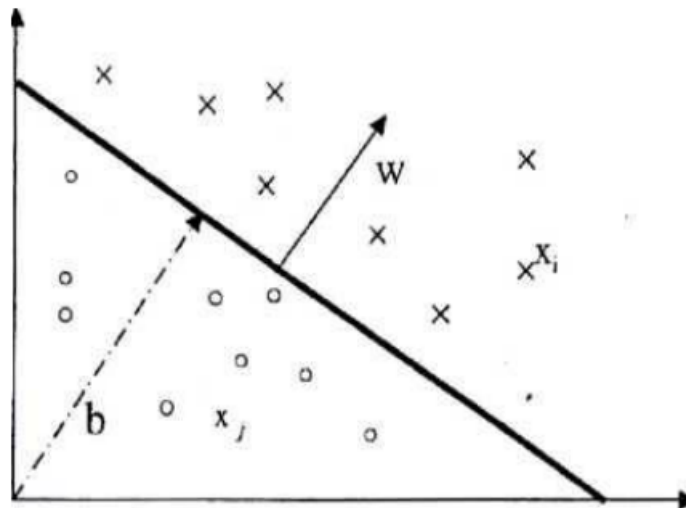
1.1 Giới thiệu

Support Vector Machines (SVM) là kỹ thuật mới đối với việc phân lớp dữ liệu, là phương pháp học sử dụng không gian giả thuyết các hàm tuyến tính trên không gian đặc trưng nhiều chiều, dựa trên lý thuyết tối ưu và lý thuyết thống kê. Trong kỹ thuật SVM không gian dữ liệu nhập ban đầu sẽ được ánh xạ vào không gian đặc trưng và trong không gian đặc trưng này mặt siêu phẳng phân chia tối ưu sẽ được xác định. Ta có tập S gồm e các mẫu học $S = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), \dots, (x_e, y_e)\}$ ($X \times Y$) với một vector đầu vào n chiều $x_i \in \mathbb{R}^n$ thuộc lớp I hoặc lớp II (tương ứng nhãn $y_i = 1$ đối với lớp I và $y_i = -1$ đối với lớp II). Một tập mẫu học được gọi là tầm thường nếu tất cả các nhãn là bằng nhau.

Đối với các dữ liệu phân chia tuyến tính, chúng ta có thể xác định được siêu phẳng $f(x)$ mà nó có thể chia tập dữ liệu. Khi đó, với mỗi siêu phẳng nhận được ta có: $f(x) \geq 0$ nếu đầu vào x thuộc lớp dương, và $f(x) < 0$ nếu x thuộc lớp âm $f(x) = w \cdot x + b$, trong đó w là vector pháp tuyến n chiều và b là giá trị ngưỡng. Vector pháp tuyến w xác định chiều của siêu phẳng $f(x)$, còn giá trị

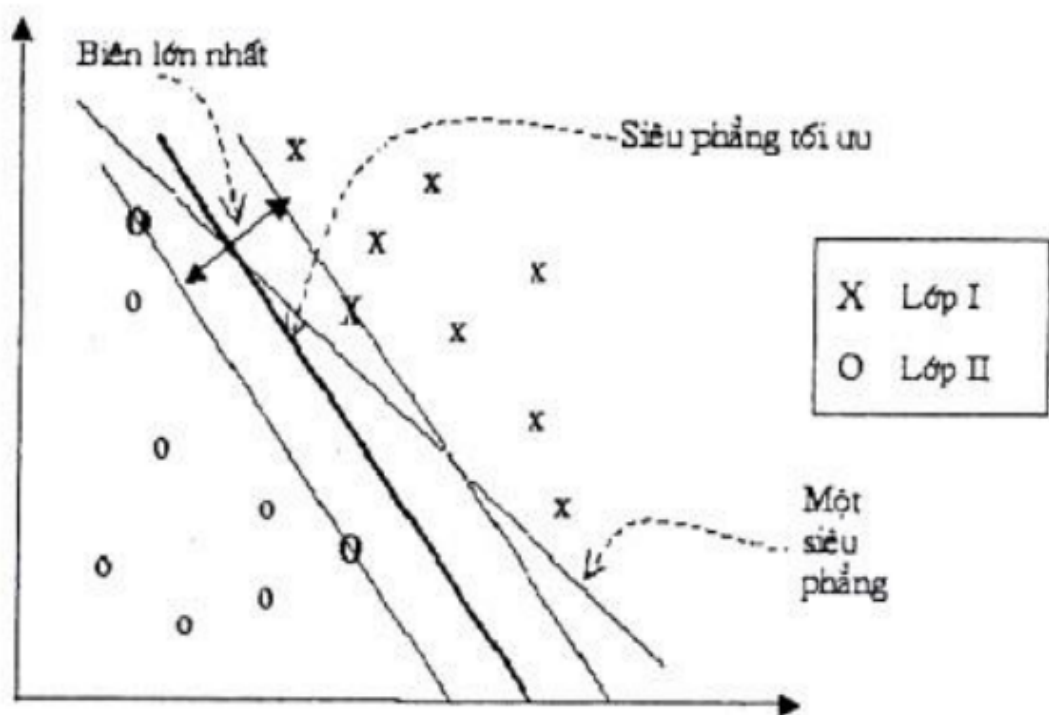
ngưỡng b xác định khoảng cách giữa siêu phẳng và gốc. Siêu phẳng có khoảng cách với dữ liệu gần nhất là lớn nhất (tức có biên lớn nhất) được gọi là siêu phẳng tối ưu.

Mục đích đặt ra ở đây là tìm được một ngưỡng (w, b) phân chia tập mẫu vào các lớp có nhãn 1 (lớp I) và -1 (lớp II) nêu ở trên với khoảng cách là lớn nhất.



Hình 1.1 Phân tách theo siêu phẳng (w, b) trong không gian 2 chiều của tập mẫu

Siêu phẳng có khoảng cách với dữ liệu gần nhất là lớn nhất (tức có biên lớn nhất) được gọi là siêu phẳng tối ưu.



Hình 1.2: Siêu phẳng tối ưu

1.2 Phân Lớp Dữ Liệu

- o **Phân lớp dữ liệu** là một kỹ thuật trong khai phá dữ liệu được sử dụng rộng rãi nhất và được nghiên cứu mở rộng hiện nay.
- o **Mục đích:** Để dự đoán những nhãn phân lớp cho các bộ dữ liệu hoặc mẫu mới.

Đầu vào: Một tập các mẫu dữ liệu huấn luyện, với một nhãn phân lớp cho mỗi mẫu dữ liệu.

Đầu ra: Bộ phân lớp dựa trên tập huấn luyện, hoặc những nhãn phân lớp.

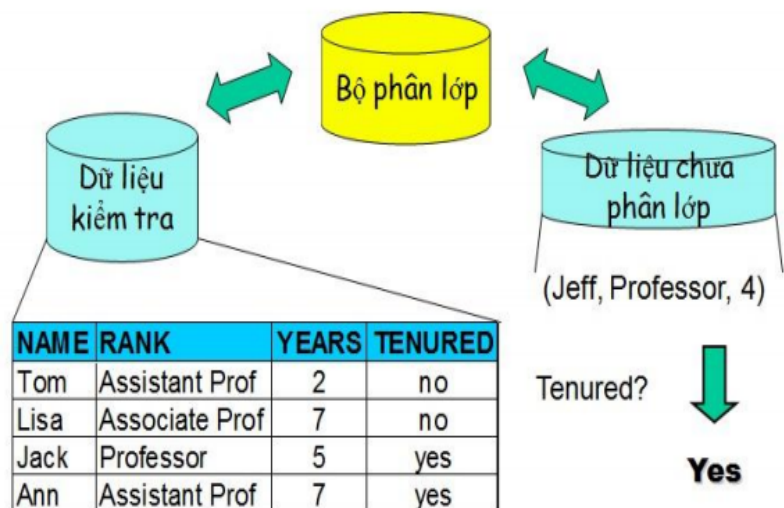
Phân lớp dữ liệu dựa trên tập huấn luyện và các giá trị trong một thuộc tính phân lớp và dùng nó để xác định lớp cho dữ liệu mới. Kỹ thuật phân lớp dữ liệu được tiến hành bao gồm 2 bước:

Bước 1. Xây dựng mô hình từ tập huấn luyện

- Mỗi bộ/mẫu dữ liệu được phân vào một lớp được xác định trước.
- Lớp của một bộ/mẫu dữ liệu được xác định bởi thuộc tính gắn nhãn lớp.
- Tập các bộ/mẫu dữ liệu huấn luyện - tập huấn luyện - được dùng để xây dựng mô hình.
- Mô hình được biểu diễn bởi các luật phân lớp, các cây quyết định hoặc các công thức toán học.

Bước 2: Sử dụng mô hình – kiểm tra tính đúng đắn của mô hình và dùng nó để phân lớp dữ liệu mới.

- Phân lớp cho những đối tượng mới hoặc chưa được phân lớp.
- Đánh giá độ chính xác của mô hình Lớp biết trước của một mẫu/bộ dữ liệu đem kiểm tra được so sánh với kết quả thu được từ mô hình. Tỷ lệ chính xác bằng phần trăm các mẫu/bộ dữ liệu được phân lớp đúng bởi mô hình trong số các lần kiểm tra.



Hình 1.4: Sử dụng mô hình

❖ **Tại sao lại sử dụng thuật toán SVM trong phân lớp dữ liệu ???**

- SVM rất hiệu quả để giải quyết bài toán dữ liệu có số chiều lớn (ảnh của dữ liệu biểu diễn gene, protein, tế bào).

- SVM giải quyết vấn đề *overfitting* rất tốt (dữ liệu có nhiều và tách rời nhóm hoặc dữ liệu huấn luyện quá ít).
- Là phương pháp phân lớp nhanh.
- Có hiệu suất tổng hợp tốt và hiệu suất tính toán cao.

1.3 Thuật toán SVM

1.3.1 Giới thiệu

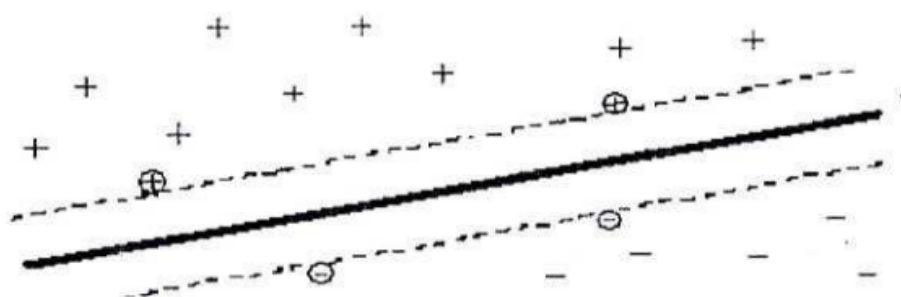
- Bài toán phân lớp (*Classification*) và dự đoán (*Prediction*) là hai bài toán cơ bản và có rất nhiều ứng dụng trong tất cả các lĩnh vực như: học máy, nhận dạng, trí tuệ nhân tạo, .v.v . Trong đồ án này, chúng em sẽ đi sâu nghiên cứu phương pháp Support Vector Machines (SVM), một phương pháp rất hiệu quả hiện nay.
- Phương pháp SVM được coi là công cụ mạnh cho những bài toán phân lớp phi tuyến tính được các tác giả Vapnik và Chervonenkis phát triển mạnh mẽ năm 1995.
- Phương pháp này thực hiện phân lớp dựa trên nguyên lý Cực tiểu hóa rủi ro có Cấu trúc SRM (*Structural Risk Minimization*), được xem là một trong các phương pháp phân lớp giám sát không tham số tính vi nhất cho đến nay. Các hàm công cụ đa dạng của SVM cho phép tạo không gian chuyên đôi để xây dựng mặt phẳng phân lớp.

1.3.2 Định nghĩa

- Là phương pháp dựa trên nền tảng của lý thuyết thống kê nên có một nền tảng toán học chặt chẽ để đảm bảo rằng kết quả tìm được là chính xác.
- Là thuật toán học giám sát (*supervised learning*) được sử dụng cho phân lớp dữ liệu.
- Là 1 phương pháp thử nghiệm, đưa ra 1 trong những phương pháp mạnh và chính xác nhất trong số các thuật toán nổi tiếng về phân lớp dữ liệu.
- SVM là một phương pháp có tính tổng quát cao nên có thể được áp dụng cho nhiều loại bài toán nhận dạng và phân loại.

1.3.3 Ý tưởng của phương pháp

- Cho trước một tập huấn luyện, được biểu diễn trong không gian vector, trong đó mỗi tài liệu là một điểm, phương pháp này tìm ra một siêu phẳng quyết định tốt nhất có thể chia các điểm trên không gian này thành hai lớp riêng biệt tương ứng là lớp + và lớp -.
- Chất lượng của siêu phẳng này được quyết định bởi khoảng cách (gọi là biên) của điểm dữ liệu gần nhất của mỗi lớp đến mặt phẳng này. Khi đó, khoảng cách biên càng lớn thì mặt phẳng quyết định càng tốt, đồng thời việc phân loại càng chính xác.
- Mục đích của phương pháp SVM là tìm được khoảng cách biên lớn nhất, điều này được minh họa như sau:



Hình 1 5: Siêu phẳng chia dữ liệu thành 2 lớp + và - với khoảng cách biên lớn nhất.

Các điểm gần nhất (điểm được khoanh tròn) là các Support Vector.

1.4 Nội dung phương pháp

1.4.1 Cơ sở lý thuyết

- SVM thực chất là một bài toán tối ưu, mục tiêu của thuật toán này là tìm được một không gian F và siêu phẳng quyết định f trên F sao cho sai số phân loại là thấp nhất.

- Cho tập mẫu $(x_1, y_1), (x_2, y_2), \dots (x_f, y_f)$ với $x_i \in \mathbb{R}^n$, thuộc vào hai lớp nhãn: $y_i \in \{-1, 1\}$ là nhãn lớp tương ứng của các x_i (-1 biểu thị lớp I, 1 biểu thị lớp II).
- Ta có, phương trình siêu phẳng chứa vector x_i trong không gian:

$$x_i \cdot w + b = 0, \quad X_i \cdot W + b > 0$$

- Đặt $f(X_i) = \text{sign}(X_i \cdot W + b) = -1, X_i \cdot W + b < 0$
- ⇒ Như vậy, $f(X_i)$ biểu diễn sự phân lớp của X_i vào hai lớp như đã nêu. Ta nói $y_i = +1$ nếu $X_i \in$ lớp I và $y_i = -1$ nếu $X_i \in$ lớp II. Khi đó, để có siêu phẳng f ta sẽ phải giải bài toán sau:
- Tìm min với W thỏa mãn điều kiện sau: $y_i(\sin(X_i \cdot W + b)) \geq 1$ với $i \in 1, n$.

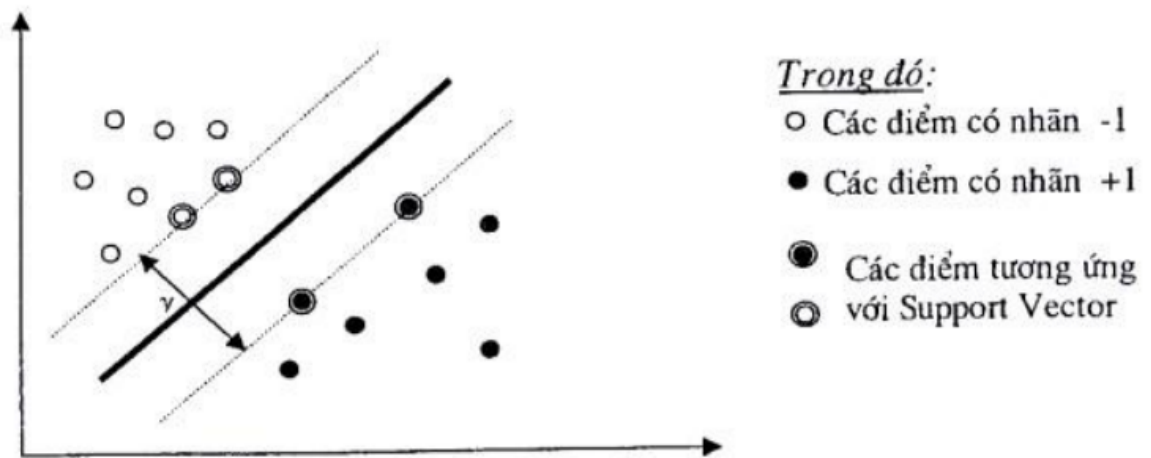
❖ Bài toán SVM có thể giải bằng kỹ thuật sử dụng toán tử Lagrange để biến đổi về thành dạng đẳng thức. Một đặc điểm thú vị của SVM là mặt phẳng quyết định chỉ phụ thuộc các Support Vector và nó có khoảng cách đến mặt phẳng quyết định là $1/\gamma$. Cho dù các điểm khác bị xóa đi thì thuật toán vẫn cho kết quả giống như ban đầu. Đây chính là điểm nổi bật của phương pháp SVM so với các phương pháp khác vì tất cả các dữ liệu trong tập huấn luyện đều được dùng để tối ưu hóa kết quả.

Kết Luận:

- Trong trường hợp nhị phân phân tách tuyến tính, việc phân lớp được thực hiện qua hàm quyết định $f(x) = \text{sign}(\langle w, x \rangle + b)$, hàm này thu được bằng việc thay đổi vector chuẩn w , đây là vector để cực đại hóa biên chức năng.
- Việc mở rộng SVM để phân đa lớp hiện nay vẫn đang được đầu tư nghiên cứu. Có một phương pháp tiếp cận để giải quyết vấn đề này là xây dựng và kết hợp nhiều bộ phân lớp nhị phân SVM (Chẳng hạn: trong quá trình luyện với SVM, bài toán phân m lớp có thể được biến đổi thành bài toán phân $2 \cdot m$ lớp, khi đó trong mỗi hai lớp, hàm quyết định sẽ được xác định cho khả năng tổng quát hóa tối đa).
- Trong phương pháp này có thể đề cập tới hai cách là *một-đối-một*, *một-đối-tất cả*.

1.4.2 Bài toán phân 2 lớp với SVM

- Bài toán đặt ra là: Xác định hàm phân lớp để phân lớp các mẫu trong tương lai, nghĩa là với một mẫu dữ liệu mới xi thì cần phải xác định xi được phân vào lớp +1 hay lớp -1.
- Để xác định hàm phân lớp dựa trên phương pháp SVM, ta sẽ tiến hành tìm hai siêu phẳng song song sao cho khoảng cách γ giữa chúng là lớn nhất có thể để phân tách hai lớp này ra làm hai phía. Hàm phân tách tương ứng với phương trình siêu phẳng nằm giữa hai siêu phẳng tìm được.



Hình 1.6 Minh họa bài toán 2 phân lớp bằng phương pháp SVM

Các điểm mà nằm trên hai siêu phẳng phân tách được gọi là các Support Vector. Các điểm này sẽ quyết định đến hàm phân tách dữ liệu.

1.4.3 Bài toán nhiều phân lớp với SVM

- Để phân nhiều lớp thì kỹ thuật SVM nguyên thủy sẽ chia không gian dữ liệu thành 2 phần và quá trình này lặp lại nhiều lần. Khi đó hàm quyết định phân dữ liệu vào lớp thứ i của tập n , 2-lớp sẽ là: $f_i(x) = w_i \cdot x_i + b_i$
- Những phần tử x là support vector sẽ thỏa điều kiện +1 nếu thuộc lớp i $f_i(x) = -1$ nếu thuộc phần còn lại.
- Như vậy, bài toán phân nhiều lớp sử dụng phương pháp SVM hoàn toàn có thể thực hiện giống như bài toán hai lớp. Bằng cách sử dụng chiến lược "một-đối-một" (one - against - one). Giả sử bài toán cần phân loại có k lớp ($k > 2$), chiến lược "một-đối-một" sẽ tiến hành $k(k-1)/2$ lần phân lớp nhị phân sử dụng phương pháp SVM. Mỗi lớp sẽ tiến hành phân tách với $k-1$

lớp còn lại để xác định k-1 hàm phân tách dựa vào bài toán phân hai lớp bằng phương pháp SVM.

1.4.4 Các bước chính của phương pháp SVM

- Phương pháp SVM yêu cầu dữ liệu được diễn tả như các vector của các số thực. Như vậy nếu đầu vào chưa phải là số thì ta cần phải tìm cách chuyển chúng về dạng số của SVM. Tiền xử lý dữ liệu: Thực hiện biến đổi dữ liệu phù hợp cho quá trình tính toán, tránh các số quá lớn mô tả các thuộc tính. Thường nên co giãn (*scaling*) dữ liệu để chuyển về đoạn $[-1, 1]$ hoặc $[0, 1]$.
- Chọn hàm hạt nhân: Lựa chọn hàm hạt nhân phù hợp tương ứng cho từng bài toán cụ thể để đạt được độ chính xác cao trong quá trình phân lớp. Thực hiện việc kiểm tra chéo để xác định các tham số cho ứng dụng.
- Điều này cũng quyết định đến tính chính xác của quá trình phân lớp. Sử dụng các tham số cho việc huấn luyện với tập mẫu.
- Trong quá trình huấn luyện sẽ sử dụng thuật toán tối ưu hóa khoảng cách giữa các siêu phẳng trong quá trình phân lớp, xác định hàm phân lớp trong không gian đặc trưng nhờ việc ánh xạ dữ liệu vào không gian đặc trưng bằng cách mô tả hạt nhân, giải quyết cho bài toán tìm siêu phẳng phân lớp dữ liệu là phân tách và không phân tách tuyến tính trong không gian đặc trưng.
- Kiểm thử tập dữ liệu Test.

2. Tìm Hiểu về KNN

1.1 Giới thiệu

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 của dữ liệu mới.

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

Ví dụ:

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 (-)?

Có thể thấy cách đơn giản nhất là so sánh tất cả các đặc điểm của dữ liệu A với tất cả tập dữ liệu học đã được gán 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 (-), trông có vẻ rất đơn giản nhưng đó là những gì mà KNN làm.

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. Phép tính khoảng cách giữa 2 điểm có thể là Euclidian, Manhattan, trọng số, Minkowski, ...

1.2 Ý tưởng của KNN

- Thuật toán KNN cho rằng những dữ liệu tương tự nhau sẽ tồn tại **gần nhau** trong một không gian, từ đó công việc của chúng ta là sẽ tìm k điểm gần với dữ liệu cần kiểm tra nhất. Việc tìm khoảng cách giữa 2 điểm cũng có nhiều công thức có thể sử dụng, tùy trường hợp mà chúng ta lựa chọn cho phù hợp. Đây là 3 cách cơ bản để tính khoảng cách 2 điểm dữ liệu x, y có k thuộc tính:

Distance functions

Euclidean	$\sqrt{\sum_{i=1}^k (x_i - y_i)^2}$
Manhattan	$\sum_{i=1}^k x_i - y_i $
Minkowski	$\left(\sum_{i=1}^k (x_i - y_i)^q \right)^{1/q}$

- Các bước trong KNN:

1. Ta có D là tập các điểm dữ liệu đã được gán nhãn và A là dữ liệu chưa được phân loại.
2. Đo khoảng cách (Euclidian, Manhattan, Minkowski, 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 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.

1.3 Ưu điểm

1. Thuật toán đơn giản, dễ dàng triển khai.
2. Độ phức tạp tính toán nhỏ.
3. Xử lý tốt với tập dữ liệu nhiễu

1.4 Nhược điểm

1. Với K nhỏ dễ gặp nhiễu dẫn tới kết quả đưa ra không chính xác

2. Cần nhiều thời gian để thực hiện do phải tính toán khoảng cách với tất cả các đối tượng trong tập dữ liệu.
3. Cần chuyển đổi kiểu dữ liệu thành các yếu tố định tính.

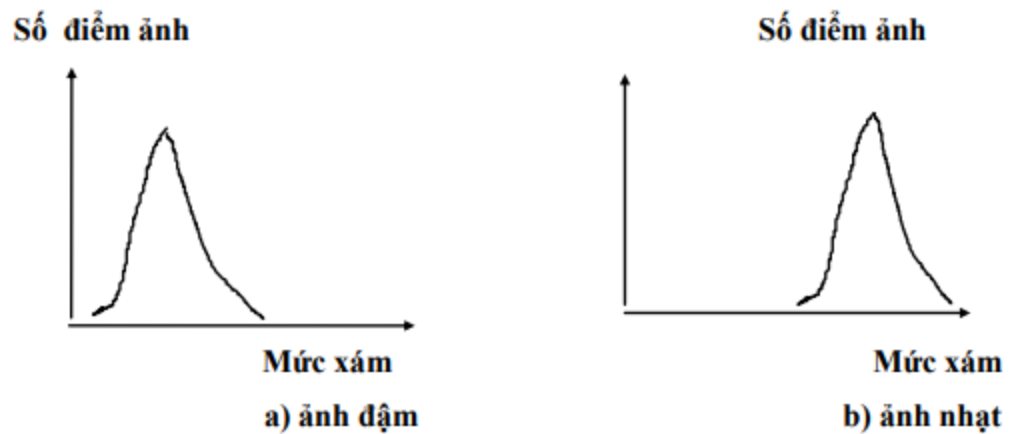
1.5 Ứng dụng

KNN là một mô hình đơn giản và trực quan nhưng vẫn có hiệu quả cao vì nó không tham số; mô hình không đưa ra giả định nào về việc phân phối dữ liệu. Hơn nữa, nó có thể được sử dụng trực tiếp để phân loại đa lớp.

Thuật toán KNN có nhiều ứng dụng trong ngành đầu tư, bao gồm dự đoán phá sản, dự đoán giá cổ phiếu, phân bổ xếp hạng tín dụng trái phiếu doanh nghiệp, tạo ra chỉ số vốn và trái phiếu tùy chỉnh.

IV. Phương pháp rút trích đặc trưng hình ảnh HOG

Lược đồ mức xám (histogram) của một ảnh, từ nay về sau ta qui ước gọi là lược đồ xám, là một hàm cung cấp tần suất xuất hiện của mỗi mức xám (grey level). Lược đồ xám được biểu diễn trong một hệ tọa độ vuông góc x, y . Trong hệ tọa độ này, trục hoành biểu diễn số mức xám từ 0 đến N , N là số mức xám (256 mức trong trường hợp chúng ta xét). Trục tung biểu diễn số điểm ảnh cho một mức xám (số điểm ảnh cú cùng mức xám). Cũng có thể biểu diễn khác một chút: trục tung là tỷ lệ số điểm ảnh có cùng mức xám trên tổng số điểm ảnh



Hình 2.1 Lược đồ xám của ảnh

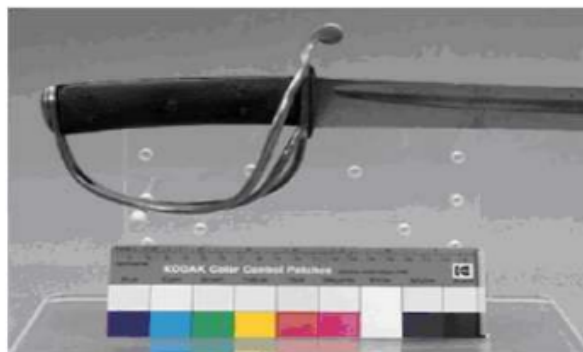


Figure 1. Object Image

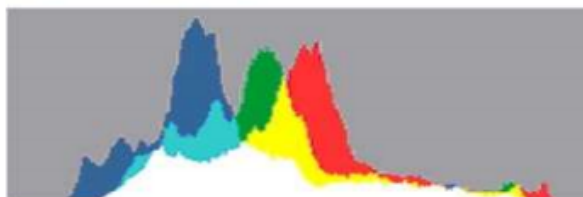
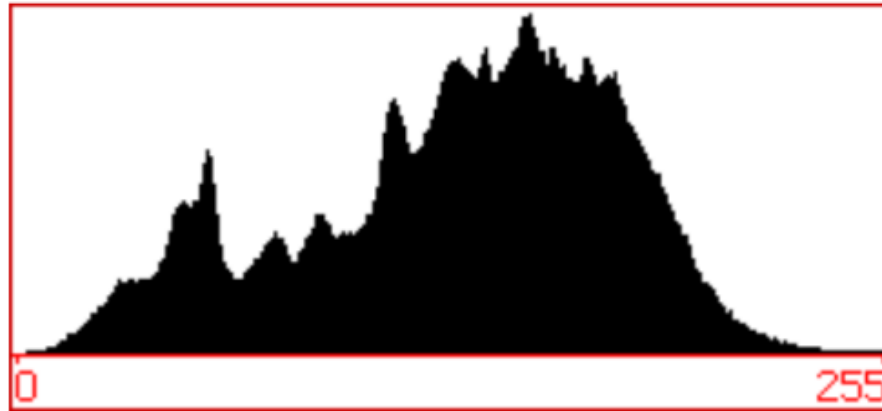


Figure 2. Histogram of the Object in Figure 1

Hình 2.2: Một ví dụ về biểu đồ tần suất histogram

- Histogram cung cấp cho những thông số cơ bản, như độ sáng và độ tương phản (contrast) của ảnh. Độ tương phản đặc trưng cho sự thay đổi độ sáng của đối tượng so với nền. Có thể nói, độ tương phản là độ nổi của điểm ảnh hay vùng ảnh so với nền. Ta có một vài nhận xét về histogram:

+ NX1. Histogram tốt có hình ngọn núi với độ cao tăng dần từ trái, cao nhất ở giữa và thấp nhất ở bên phải. Điều đó chứng tỏ số lượng điểm ảnh nhiều nhất là ở độ sáng trung bình. (Xem Hình 2.3).



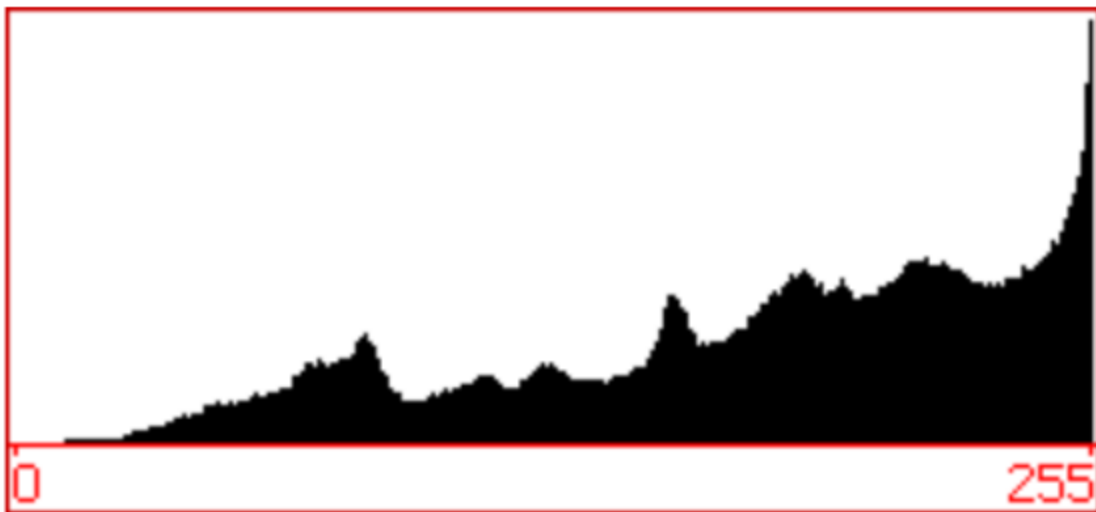
Hình 2.3: Histogram tốt

+ NX2. Ảnh quá tối: histogram bị nghiêng về bên trái, có một cái cột gần như thẳng đứng sát trái (Xem Hình 2.4).



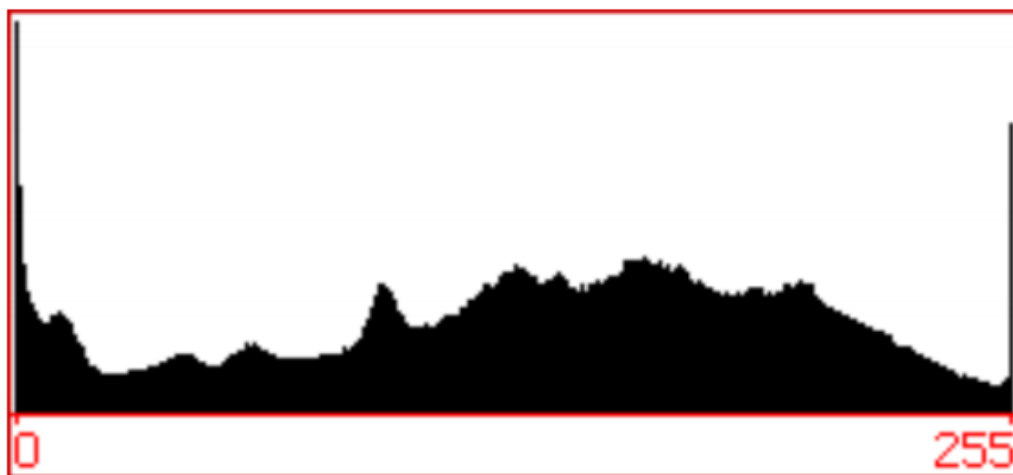
Hình 2.4: Histogram của ảnh quá tối

+ NX3. Ảnh quá sáng: histogram bị nghiêng về bên phải, có một cái cột gần như thẳng đứng sát phải (Xem Hình 2.5)



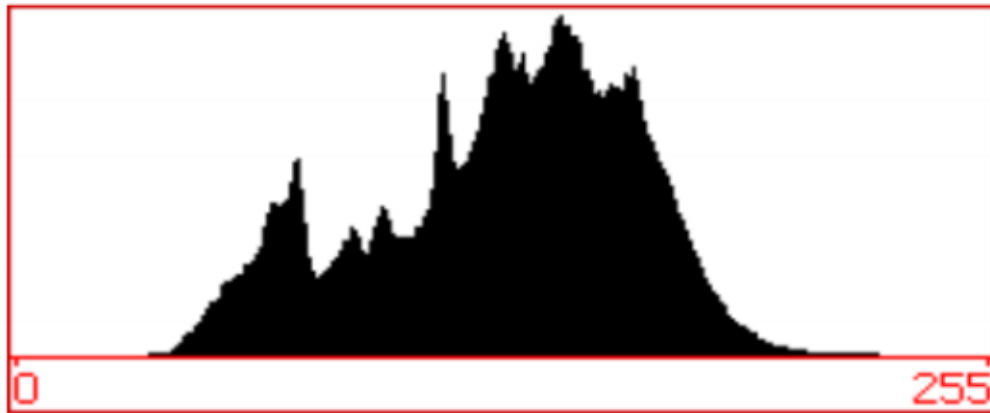
Hình 2.5: Histogram của ảnh quá sáng

+ NX4. Ảnh quá tương phản: có hai cái cột nằm ở 2 đầu trái phải (Xem Hình 2.6)



Hình 2.6: Histogram của ảnh quá tương phản

+ NX5. Ảnh kém tương phản: dải màu bị dồn vào giữa, hai đầu không có gì. (Xem Hình 2.7)



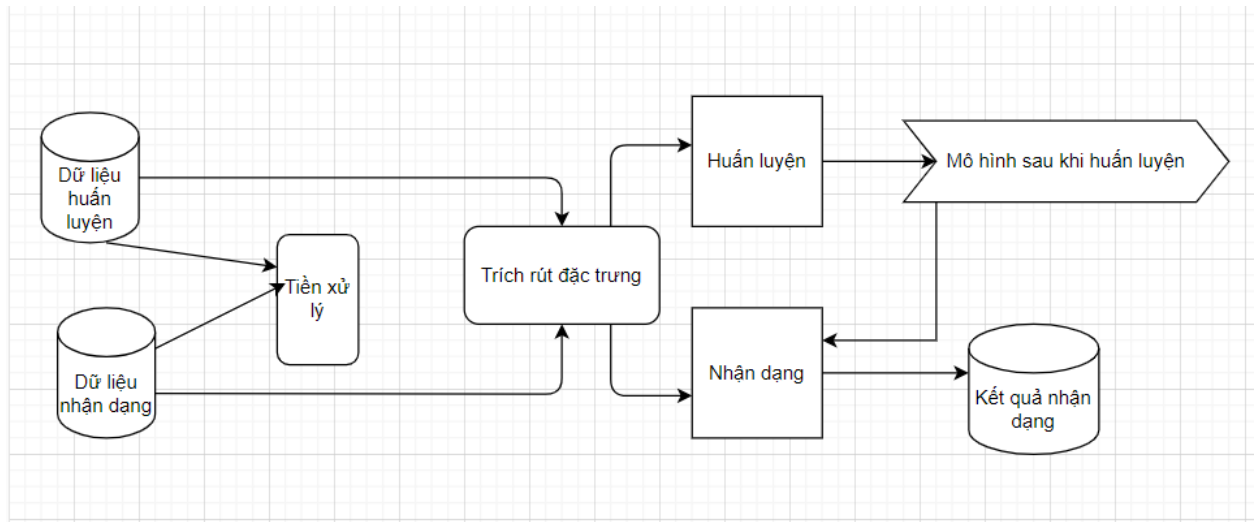
Hình 2.7: Histogram của ảnh kém tương phản

- Từ lược đồ xám ta có thể suy diễn ra các tính chất quan trọng của ảnh như giá trị xám trung bình hoặc độ tần mạn. Qua cách tác động lên điểm ảnh, sự phân bố của biểu đồ cột được thay đổi theo mục đích. Dựa vào lược đồ xám chúng ta có thể xác định được ngưỡng thích hợp cho quá trình phân đoạn hoặc tính được các đại lượng đặc trưng của một ảnh.

Câu 3. Xây dựng hệ thống

V. Xây dựng hệ thống

1. Sơ đồ khối



2. Quá trình thực hiện

1. Bước 1: Tiền xử lý

1.1 Thực hiện giảm nhiễu cho ảnh (sử dụng kỹ thuật lọc trung bình)

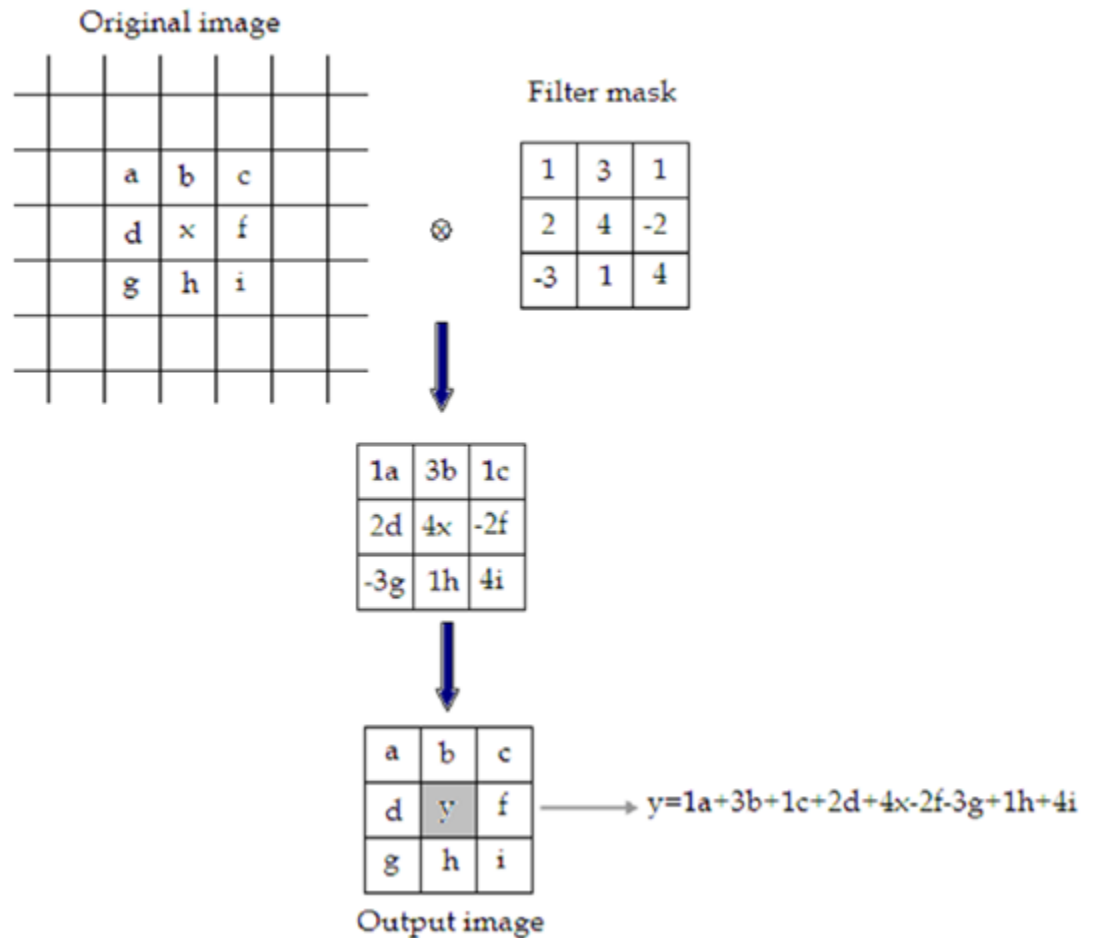
- Lọc trung bình là kỹ thuật lọc tuyến tính đơn giản trong việc tính toán.
- Kỹ thuật này thường ưu tiên cho các hướng để bảo vệ biên ảnh khỏi bị mờ khi làm trơn ảnh.
- Lọc trung bình giúp cân bằng màu cho các pixel trong 1 vùng ảnh nhất định sao cho sự chênh lệch màu sẽ giảm xuống sau khi lọc. Ma trận đầu ra sẽ có trị số tại các node có sự chênh lệch ít hơn so với ma trận đầu vào.
- Mỗi 1 điểm ảnh (Pixel) được thay thế bằng trung bình trọng số của các điểm lân cận.

- Các bước thực hiện:

Bước 1: Quét cửa sổ lọc (ma trận kích thước 3×3) lần lượt lên các thành phần của ảnh đầu vào, điền các giá trị được quét vào cửa sổ lọc

Bước 2: Tính giá trị trung bình các thành phần trong cửa sổ lọc

Bước 3: Gán giá trị trung bình này cho ảnh đầu ra.



1.2 Xóa nền cho ảnh.

2. Bước 2: Trích rút đặc trưng

- HOG là viết tắt của Histogram of Oriented Gradient - một loại “feature descriptor”.
- Mục đích của HOG 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.

- HOG 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 cells
- 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 4 bước cơ bản để xây dựng một vector HOG cho hình ảnh, bao gồm:
 - Bước 1: Tính gradient
 - Bước 2: Tính vector đặc trưng histogram
 - Bước 3: Chuẩn hóa khối (blocks)
 - Bước 4: Tính toán vector HOG

2.1. Tính Gradient

- Gradient là 1 vector có thành phần biểu thị tốc độ thay đổi giá trị của điểm ảnh theo 2 hướng x và y.
- Hình ảnh được chia thành 1 lưới ô vuông và trên đó chúng ta xác định được các vùng cục bộ liền kề hoặc chồng lên nhau

- 1 vùng cục bộ bao gồm nhiều ô cục bộ (cell) có kích thước 4×4 . (Ảnh kích thước $64 \times 64 \Rightarrow$ có 256 cell).
- Mỗi 1 cell có 16 điểm ảnh (16 pixel). Trên mỗi điểm ảnh ta cần tính 2 giá trị là độ lớn của gradient và phương của gradient.

Tổng: 1 cell sẽ có 2 ma trận là ma trận độ lớn gradient và ma trận phương của gradient với kích thước 4×4 .

- Thực hiện tính đạo hàm cho mỗi điểm trong 1 cell để tìm ra 2 ma trận:

Để tính bộ lọc sobel, phép tích chập của kernel kích thước 3×3 được thực hiện với hình ảnh ban đầu. Nếu chúng ta kí hiệu I là ma trận ảnh gốc và G_x, G_y là 2 ma trận ảnh mà mỗi điểm trên nó lần lượt là đạo hàm theo trục x trục y . Chúng ta có thể tính toán được kernel như sau:

- Đạo hàm theo chiều ngang:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * I$$

- Đạo hàm theo chiều dọc:

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * I$$

Kí hiệu * tương tự như phép tích chập giữa bộ lọc bên trái và ảnh đầu vào bên phải.

Giá trị độ lớn gradient (gradient magnitude) và phương gradient (gradient direction) có thể được tạo ra từ 2 đạo hàm G_x và G_y theo công thức bên dưới:

- Độ lớn gradient

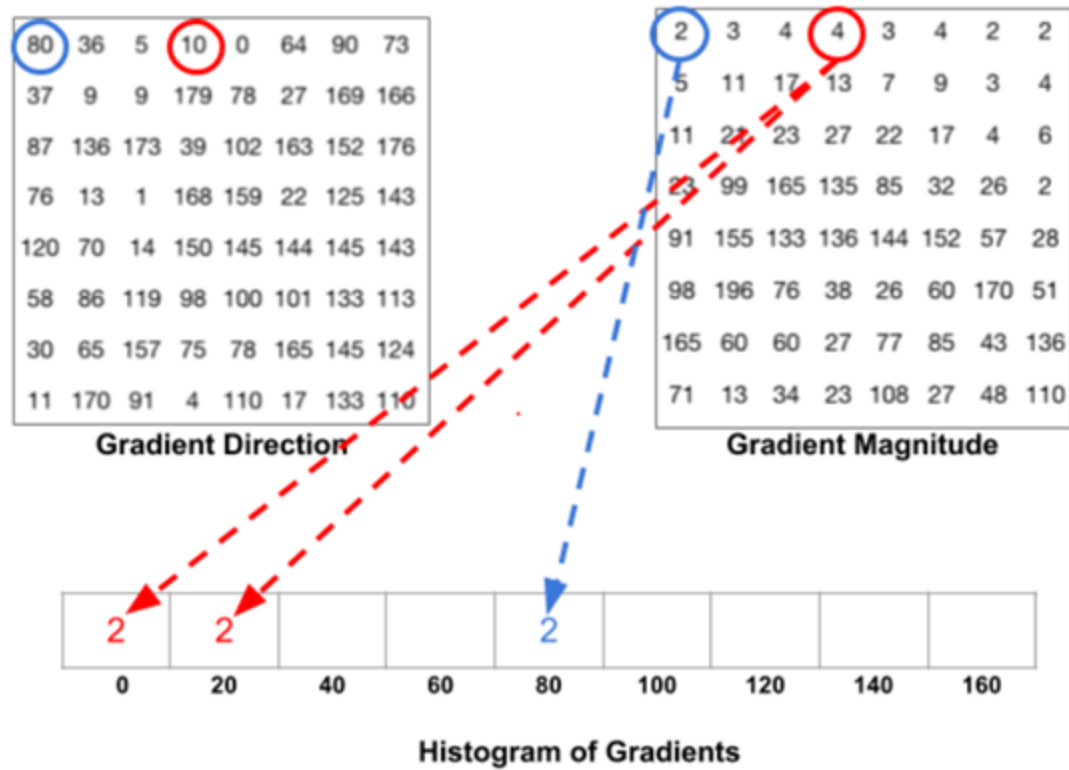
$$G = \sqrt{G_x^2 + G_y^2}$$

- Phương gradient:

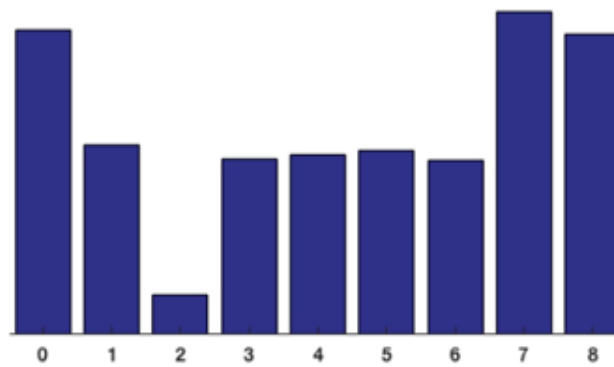
$$\theta = \arctan\left(\frac{G_y}{G_x}\right)$$

2.2. Tính vector đặc trưng histogram.

- Thực hiện mapping độ lớn gradient vào các bins tương ứng của phương gradient.
- Sắp xếp các giá trị phương gradient theo thứ tự từ nhỏ đến lớn và chia chúng vào 9 bins. Độ lớn của phương gradient sẽ nằm trong khoảng $[0, 180]$ nên mỗi bins sẽ có độ dài là 20 như hình bên dưới.
- 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.
- Trong hình bên dưới ô được khoanh trong hình tròn viền xanh tương ứng với phương gradient là 80 và độ lớn gradient là 2. Khi đó tại véc tơ bins của HOG, phương gradient bằng 80 sẽ rơi vào vị trí thứ 5 nên tại ô này chúng ta điền giá trị 2 ứng với độ lớn gradient.



Tính tổng tất cả các độ lớn gradient thuộc cùng 1 bins của véc tơ bins ta thu được biểu đồ Histogram of Gradients như bên dưới:



Hình 4: Biểu đồ Histogram of Gradient gồm 9 bins tương ứng với một ô vuông trong lưới ô vuông.

2.3. Chuẩn hóa khối Block

- Một block gồm nhiều cell, block 4*4 nghĩa là ta có vùng diện tích của 16 cell liên kề → block này sẽ phủ trên diện tích = 16x16 pixel.
- Trong quá trình chuẩn hóa, ta sẽ lần lượt chuẩn hóa block 4*4 đầu tiên, rồi dịch block đó sang 1 cell và cũng thực hiện chuẩn hóa cho block này. Như vậy, giữa block đầu tiên và block liền kề đã có sự chồng lấn cell lẫn nhau (2 cell). Tổng 1 ảnh có $13 \times 13 = 169$ khối block.

Thao tác cụ thể chuẩn hóa cho mỗi block sẽ dùng L2-Norm.

Cách làm: Lấy tất cả vector của 16 cell trong block đang xét nối lại với nhau thành vector v . Vector v có $9 \times 16 = 144$ phần tử. Sau đó ta chuẩn hóa (tính toán lại vector v) theo công thức bên dưới:

$$L2\text{-norm}, v \rightarrow v / \sqrt{\|v\|_2^2 + \epsilon^2};$$

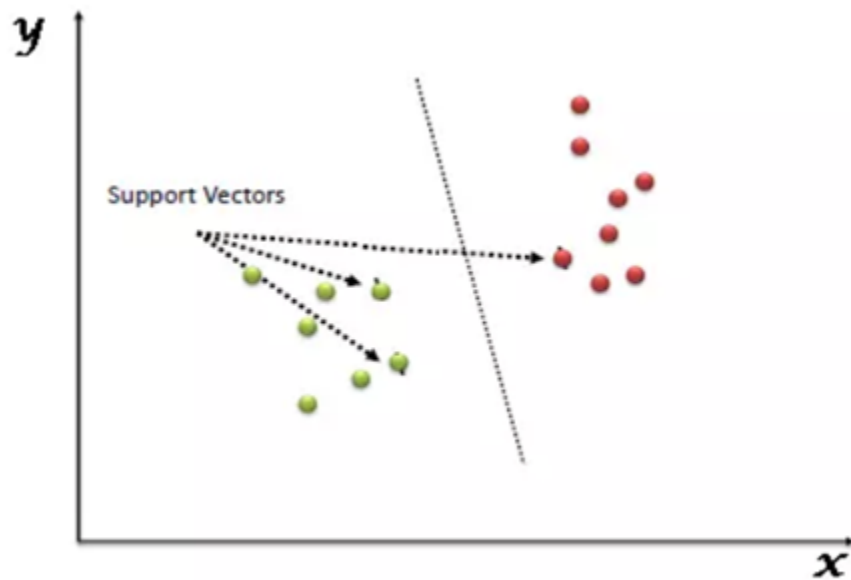
Sau khi chuẩn hóa tất cả các Block ta sẽ nối các vector này lại thành 1 vector đại diện cho toàn bộ ảnh

2.4. Tính toán vector HOG.

- Với mỗi hình ảnh kích thước 64x64, chia thành các block 16x16 chồng nhau, sẽ có 13 block ngang và 13 block dọc, nên sẽ có $13 \times 13 = 169$ blocks.
- Mỗi block gồm 16 cell. Khi áp dụng biểu đồ 9 bin cho mỗi cell, mỗi block sẽ được đại diện bởi một vector có kích thước 144x1.
- 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 $144 \times 169 = 24336$ phần tử.

3. Bước 3: Huấn luyện mô hình (Phân loại ảnh)

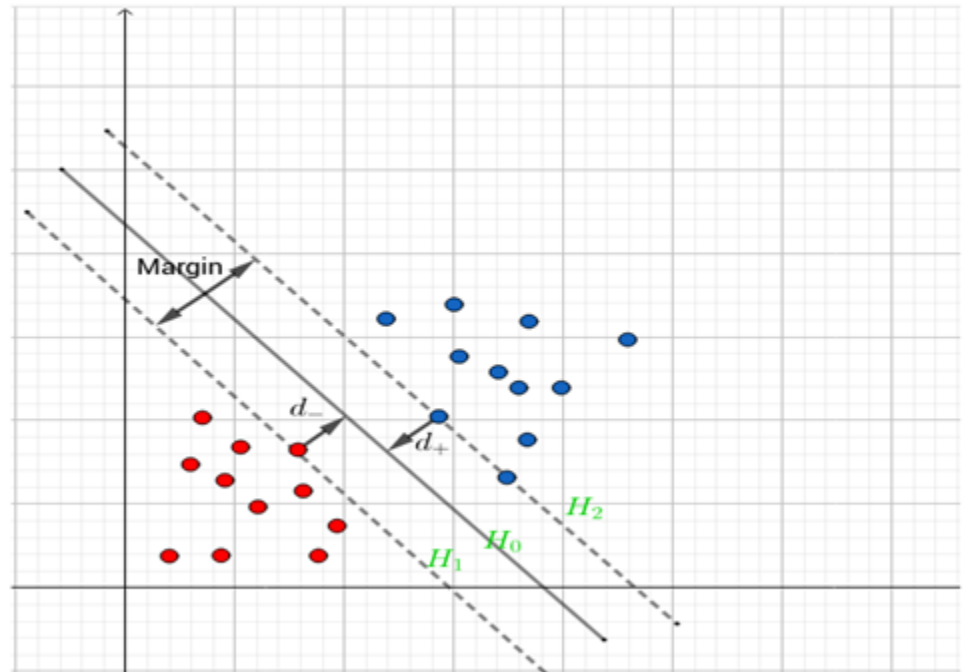
- SVM là một thuật toán giám sát, nó có thể sử dụng cho cả việc phân loại hoặc đệ quy. Tuy nhiên nó được sử dụng chủ yếu cho việc phân loại. Trong thuật toán này, chúng ta vẽ đồ thị dữ liệu là các điểm trong n chiều (ở đây n là số lượng các tính năng) với giá trị của mỗi tính năng sẽ là một phần liên kết. Sau đó chúng ta thực hiện tìm "đường bay" (hyper-plane) phân chia các lớp.
- Hyper-plane nó chỉ hiểu đơn giản là 1 đường thẳng có thể phân chia các lớp ra thành hai phần riêng biệt.



- Cách làm SVM:
 - SVM là tìm một siêu phẳng (hyper plane) để phân tách các điểm dữ liệu. Siêu phẳng này sẽ chia không gian thành các miền khác nhau và mỗi miền sẽ chứa một loại dữ liệu.
 - Siêu phẳng được biểu diễn bằng hàm số (W và X là các vector $\langle W.X \rangle$ là tích vô hướng) Hay (W là ma trận chuyển vị)

- Cách chọn siêu phẳng tối ưu:
 - Siêu phẳng phân tách hai lớp dữ liệu thỏa mã $\langle W.X \rangle + b = 0$. Siêu phẳng này tạo ra hai nửa không gian (half space) dữ liệu:

Không gian các dữ liệu lớp âm thỏa mã và không gian dữ liệu lớp dương thỏa mã



Tiếp theo ta chọn hai siêu phẳng lề đi qua điểm thuộc lớp âm và đi qua điểm thuộc lớp dương đều song song với

- : $\langle W.X \rangle + b = -1$

- : $\langle W.X \rangle + b = 1$

Khoảng cách từ đến là

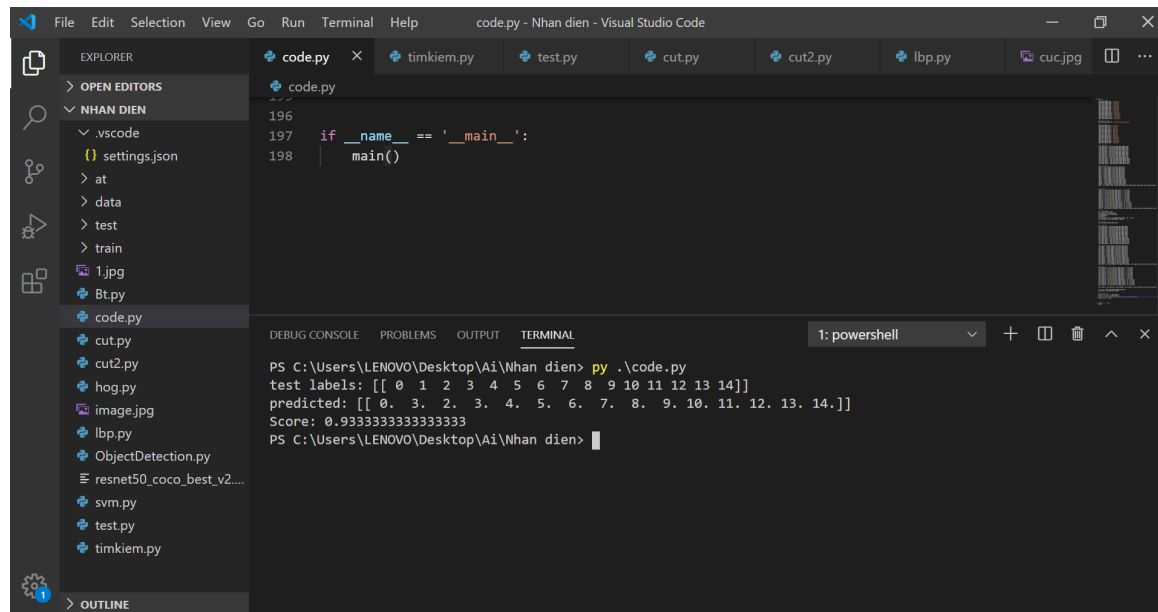
Khoảng cách từ đến là

$m = +$ được gọi là mức lẻ

Siêu phẳng tối ưu mà chúng ta cần chọn là siêu phẳng phân tách có lề lớn nhất. Lý thuyết học máy đã chỉ ra rằng một siêu phẳng như vậy sẽ cực tiểu hóa giới hạn lỗi mắc phải.

4. Bước 4: Nhận dạng ảnh

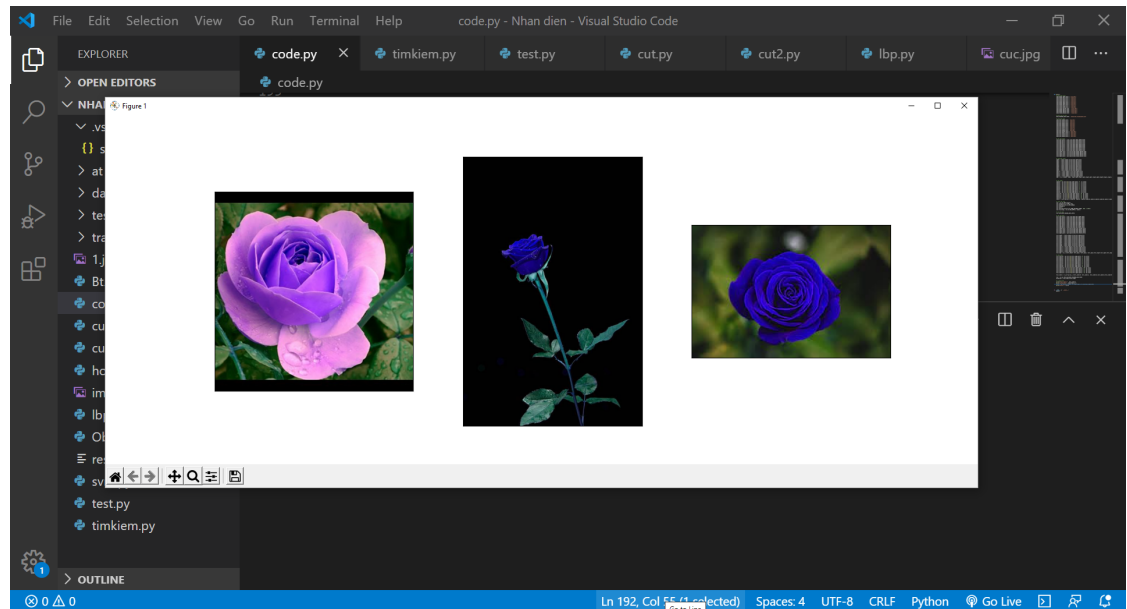
4.1 Kết quả train và test:



The screenshot shows the Visual Studio Code interface with a file explorer on the left and a terminal at the bottom. The file explorer shows a project named 'NHAN DIEN' with a folder 'data' containing 'train' and 'test' subfolders, and several image files. The terminal shows the execution of a Python script named 'code.py' in a PowerShell window. The output of the script is as follows:

```
PS C:\Users\LENOVO\Desktop\Ai\Nhan dien> py .\code.py
test labels: [[ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14]]
predicted: [[ 0. 3. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13. 14.]]
Score: 0.9333333333333333
PS C:\Users\LENOVO\Desktop\Ai\Nhan dien>
```


4.2 Kết quả tìm kiếm hình ảnh từ 1 hình ảnh:



4.3 Code

Code.py

```
import glob

from cv2 import cv2

import numpy as np


def create_images_array(load_img_paths):

    imgs = []

    # dn
    win_size = (64, 64)
    block_size = (16, 16)
    block_stride = (4, 4)
    cell_size = (4, 4)

    bins = 9

    # load anh

    for load_img_path in load_img_paths:
```

```

        img = cv2.imread(load_img_path)

        # loc trung binh

        img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

        figure_size = 9

        img = cv2.blur(img, (figure_size, figure_size))

        # hog

        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

        gray = cv2.resize(gray, win_size)

        hog = cv2.HOGDescriptor(win_size, block_size,
block_stride, cell_size, bins)

        img = hog.compute(gray)

        imgs.append(img)

    return np.array(imgs, np.float32)

def main():

    # path anh train

    LOAD_TRAIN_IMG0S_PATH = 'train/0/*'

    LOAD_TRAIN_IMG1S_PATH = 'train/1/*'

    LOAD_TRAIN_IMG2S_PATH = 'train/2/*'

    LOAD_TRAIN_IMG3S_PATH = 'train/3/*'

    LOAD_TRAIN_IMG4S_PATH = 'train/4/*'

    LOAD_TRAIN_IMG5S_PATH = 'train/5/*'

    LOAD_TRAIN_IMG6S_PATH = 'train/6/*'

    LOAD_TRAIN_IMG7S_PATH = 'train/7/*'

    LOAD_TRAIN_IMG8S_PATH = 'train/8/*'

    LOAD_TRAIN_IMG9S_PATH = 'train/9/*'

    LOAD_TRAIN_IMG10S_PATH = 'train/10/*'

```

```
LOAD_TRAIN_IMG11S_PATH = 'train/11/*'

LOAD_TRAIN_IMG12S_PATH = 'train/12/*'

LOAD_TRAIN_IMG13S_PATH = 'train/13/*'

LOAD_TRAIN_IMG14S_PATH = 'train/14/*'


# path luu du lieu

SAVE_TRAINED_DATA_PATH = 'data/svm_trained_data.xml'


# path anh test

LOAD_TEST_IMG0S_PATH = 'test/0/*'

LOAD_TEST_IMG1S_PATH = 'test/1/*'

LOAD_TEST_IMG2S_PATH = 'test/2/*'

LOAD_TEST_IMG3S_PATH = 'test/3/*'

LOAD_TEST_IMG4S_PATH = 'test/4/*'

LOAD_TEST_IMG5S_PATH = 'test/5/*'

LOAD_TEST_IMG6S_PATH = 'test/6/*'

LOAD_TEST_IMG7S_PATH = 'test/7/*'

LOAD_TEST_IMG8S_PATH = 'test/8/*'

LOAD_TEST_IMG9S_PATH = 'test/9/*'

LOAD_TEST_IMG10S_PATH = 'test/10/*'

LOAD_TEST_IMG11S_PATH = 'test/11/*'

LOAD_TEST_IMG12S_PATH = 'test/12/*'

LOAD_TEST_IMG13S_PATH = 'test/13/*'

LOAD_TEST_IMG14S_PATH = 'test/14/*'


# lay anh train

load_img0_paths = glob.glob(LOAD_TRAIN_IMG0S_PATH)
```

```
load_img1_paths = glob.glob(Load_TRAIN_IMG1S_PATH)
load_img2_paths = glob.glob(Load_TRAIN_IMG2S_PATH)
load_img3_paths = glob.glob(Load_TRAIN_IMG3S_PATH)
load_img4_paths = glob.glob(Load_TRAIN_IMG4S_PATH)
load_img5_paths = glob.glob(Load_TRAIN_IMG5S_PATH)
load_img6_paths = glob.glob(Load_TRAIN_IMG6S_PATH)
load_img7_paths = glob.glob(Load_TRAIN_IMG7S_PATH)
load_img8_paths = glob.glob(Load_TRAIN_IMG8S_PATH)
load_img9_paths = glob.glob(Load_TRAIN_IMG9S_PATH)
load_img10_paths = glob.glob(Load_TRAIN_IMG10S_PATH)
load_img11_paths = glob.glob(Load_TRAIN_IMG11S_PATH)
load_img12_paths = glob.glob(Load_TRAIN_IMG12S_PATH)
load_img13_paths = glob.glob(Load_TRAIN_IMG13S_PATH)
load_img14_paths = glob.glob(Load_TRAIN_IMG14S_PATH)

# load anh train

imgs0 = create_images_array(load_img0_paths)
imgs1 = create_images_array(load_img1_paths)
imgs2 = create_images_array(load_img2_paths)
imgs3 = create_images_array(load_img3_paths)
imgs4 = create_images_array(load_img4_paths)
imgs5 = create_images_array(load_img5_paths)
imgs6 = create_images_array(load_img6_paths)
imgs7 = create_images_array(load_img7_paths)
imgs8 = create_images_array(load_img8_paths)
imgs9 = create_images_array(load_img9_paths)
imgs10 = create_images_array(load_img10_paths)
```

```

    imgs11 = create_images_array(load_img11_paths)

    imgs12 = create_images_array(load_img12_paths)

    imgs13 = create_images_array(load_img13_paths)

    imgs14 = create_images_array(load_img14_paths)

    imgs = np.r_[imgs0, imgs1,
    imgs2, imgs3, imgs4, imgs5, imgs6, imgs7, imgs8, imgs9, imgs10, imgs11,
    imgs12, imgs13, imgs14]

# tao label

labels0 = np.full(len(load_img0_paths), 0, np.int32)
labels1 = np.full(len(load_img1_paths), 1, np.int32)
labels2 = np.full(len(load_img2_paths), 2, np.int32)
labels3 = np.full(len(load_img3_paths), 3, np.int32)
labels4 = np.full(len(load_img4_paths), 4, np.int32)
labels5 = np.full(len(load_img5_paths), 5, np.int32)
labels6 = np.full(len(load_img6_paths), 6, np.int32)
labels7 = np.full(len(load_img7_paths), 7, np.int32)
labels8 = np.full(len(load_img8_paths), 8, np.int32)
labels9 = np.full(len(load_img9_paths), 9, np.int32)
labels10 = np.full(len(load_img10_paths), 10, np.int32)
labels11 = np.full(len(load_img11_paths), 11, np.int32)
labels12 = np.full(len(load_img12_paths), 12, np.int32)
labels13 = np.full(len(load_img13_paths), 13, np.int32)
labels14 = np.full(len(load_img14_paths), 14, np.int32)

labels = np.array([np.r_[labels0, labels1,
labels2, labels3, labels4, labels5, labels6, labels7, labels8, labels
9, labels10, labels11, labels12, labels13, labels14,]])

```

```
# train du lieu

svm = cv2.ml.SVM_create()

svm.setType(cv2.ml.SVM_C_SVC)

svm.setKernel(cv2.ml.SVM_LINEAR)

svm.setGamma(1)

svm.setC(1)

svm.setTermCriteria((cv2.TERM_CRITERIA_COUNT, 100,
1.e-06))

svm.train(imgs, cv2.ml.ROW_SAMPLE, labels)

# luu du lieu

svm.save(SAVE_TRAINED_DATA_PATH)

# lay anh test

test_img0_paths = glob.glob(LOAD_TEST_IMG0S_PATH)
test_img1_paths = glob.glob(LOAD_TEST_IMG1S_PATH)
test_img2_paths = glob.glob(LOAD_TEST_IMG2S_PATH)
test_img3_paths = glob.glob(LOAD_TEST_IMG3S_PATH)
test_img4_paths = glob.glob(LOAD_TEST_IMG4S_PATH)
test_img5_paths = glob.glob(LOAD_TEST_IMG5S_PATH)
test_img6_paths = glob.glob(LOAD_TEST_IMG6S_PATH)
test_img7_paths = glob.glob(LOAD_TEST_IMG7S_PATH)
test_img8_paths = glob.glob(LOAD_TEST_IMG8S_PATH)
test_img9_paths = glob.glob(LOAD_TEST_IMG9S_PATH)
test_img10_paths = glob.glob(LOAD_TEST_IMG10S_PATH)
test_img11_paths = glob.glob(LOAD_TEST_IMG11S_PATH)
```

```
test_img12_paths = glob.glob(Load_Test_Image12s_Path)
test_img13_paths = glob.glob(Load_Test_Image13s_Path)
test_img14_paths = glob.glob(Load_Test_Image14s_Path)

test_imgs0 = create_images_array(test_img0_paths)
test_imgs1 = create_images_array(test_img1_paths)
test_imgs2 = create_images_array(test_img2_paths)
test_imgs3 = create_images_array(test_img3_paths)
test_imgs4 = create_images_array(test_img4_paths)
test_imgs5 = create_images_array(test_img5_paths)
test_imgs6 = create_images_array(test_img6_paths)
test_imgs7 = create_images_array(test_img7_paths)
test_imgs8 = create_images_array(test_img8_paths)
test_imgs9 = create_images_array(test_img9_paths)
test_imgs10 = create_images_array(test_img10_paths)
test_imgs11 = create_images_array(test_img11_paths)
test_imgs12 = create_images_array(test_img12_paths)
test_imgs13 = create_images_array(test_img13_paths)
test_imgs14 = create_images_array(test_img14_paths)

test_imgs = np.r_[test_imgs0, test_imgs1, test_imgs2,
test_imgs3,test_imgs4,test_imgs5,test_imgs6,test_imgs7,test_im
gs8,test_imgs9,test_imgs10,test_imgs11,test_imgs12,test_imgs13
,test_imgs14]

# chay test

test_labels0 = np.full(len(test_img0_paths), 0, np.int32)
test_labels1 = np.full(len(test_img1_paths), 1, np.int32)
test_labels2 = np.full(len(test_img2_paths), 2, np.int32)
```

```
test_labels3 = np.full(len(test_img3_paths), 3, np.int32)
test_labels4 = np.full(len(test_img4_paths), 4, np.int32)
test_labels5 = np.full(len(test_img5_paths), 5, np.int32)
test_labels6 = np.full(len(test_img6_paths), 6, np.int32)
test_labels7 = np.full(len(test_img7_paths), 7, np.int32)
test_labels8 = np.full(len(test_img8_paths), 8, np.int32)
test_labels9 = np.full(len(test_img9_paths), 9, np.int32)
test_labels10 = np.full(len(test_img10_paths), 10,
np.int32)

test_labels11 = np.full(len(test_img11_paths), 11,
np.int32)

test_labels12 = np.full(len(test_img12_paths), 12,
np.int32)

test_labels13 = np.full(len(test_img13_paths), 13,
np.int32)

test_labels14 = np.full(len(test_img14_paths), 14,
np.int32)

test_labels = np.array([np.r_[test_labels0, test_labels1,
test_labels2,test_labels3,test_labels4,test_labels5,test_label
s6,test_labels7,test_labels8,test_labels9,test_labels10,test_l
abels11,test_labels12,test_labels13,test_labels14]])

svm = cv2.ml.SVM_load(SAVE_TRAINED_DATA_PATH)

predicted = svm.predict(test_imgs)

# in man hinh

print("test labels:", test_labels)

print("predicted:", predicted[1].T)
```



```
        score = np.sum(test_labels ==
predicted[1].T)/len(test_labels[0])

        print("Score:", score)

if __name__ == '__main__':
    main()
```

timkiem.py

```
import glob

from cv2 import cv2

import numpy as np

from matplotlib import pyplot as plt

from PIL import Image, ImageFilter

path0 = glob.glob("train/0/*.jpg")
path1 = glob.glob("train/1/*.jpg")
path2 = glob.glob("train/2/*.jpg")
path3 = glob.glob("train/3/*.jpg")
path4 = glob.glob("train/4/*.jpg")
path5 = glob.glob("train/5/*.jpg")
path6 = glob.glob("train/6/*.jpg")
path7 = glob.glob("train/7/*.jpg")
path8 = glob.glob("train/8/*.jpg")
path9 = glob.glob("train/9/*.jpg")
path10 = glob.glob("train/10/*.jpg")
```

```
path11 = glob.glob("train/11/*.jpg")
path12 = glob.glob("train/12/*.jpg")
path13 = glob.glob("train/13/*.jpg")
path14 = glob.glob("train/14/*.jpg")

cv_img = []

SAVE_TRAINED_DATA_PATH = 'data/svm_trained_data.xml'

imgs = []
img = cv2.imread('1.jpg')
win_size = (64, 64)
block_size = (16, 16)
block_stride = (4, 4)
cell_size = (4, 4)
bins = 9
img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
figure_size = 9
img = cv2.blur(img, (figure_size, figure_size))
# hog
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
gray = cv2.resize(gray, win_size)
hog = cv2.HOGDescriptor(win_size, block_size, block_stride,
cell_size, bins)
img = hog.compute(gray)

imgs.append(img)
```

```
imgs = np.array(imgs,np.float32)

svm = cv2.ml.SVM_load(SAVE_TRAINED_DATA_PATH)
predicted = svm.predict(imgs)
print(predicted[1].T)
kq = predicted[1].T
if kq == 0:
    for img in path0:
        n = cv2.imread(img)
        cv_img.append(n)

    plt.figure(figsize=(15,6))
    plt.subplot(131)
    plt.imshow(cv_img[0])
    plt.xticks([], plt.yticks([]))
    plt.subplot(132)
    plt.imshow(cv_img[1])
    plt.xticks([], plt.yticks([]))
    plt.subplot(133)
    plt.imshow(cv_img[2])
    plt.xticks([], plt.yticks([]))
    plt.show()
elif kq == 1:
    for img in path1:
        n = cv2.imread(img)
        cv_img.append(n)

    plt.figure(figsize=(15,6))
```

```
plt.subplot(131)

plt.imshow(cv_img[0])

plt.xticks([], plt.yticks([]))

plt.subplot(132)

plt.imshow(cv_img[1])

plt.xticks([], plt.yticks([]))

plt.subplot(133)

plt.imshow(cv_img[2])

plt.xticks([], plt.yticks([]))

plt.show()

elif kq == 2:

    for img in path3:

        n = cv2.imread(img)

        cv_img.append(n)

    plt.figure(figsize=(15,6))

    plt.subplot(131)

    plt.imshow(cv_img[0])

    plt.xticks([], plt.yticks([]))

    plt.subplot(132)

    plt.imshow(cv_img[1])

    plt.xticks([], plt.yticks([]))

    plt.subplot(133)

    plt.imshow(cv_img[2])

    plt.xticks([], plt.yticks([]))

    plt.show()

elif kq == 3:

    for img in path3:
```

```
        n = cv2.imread(img)

        cv_img.append(n)

    plt.figure(figsize=(15,6))

    plt.subplot(131)

    plt.imshow(cv_img[0])

    plt.xticks([], plt.yticks([]))

    plt.subplot(132)

    plt.imshow(cv_img[1])

    plt.xticks([], plt.yticks([]))

    plt.subplot(133)

    plt.imshow(cv_img[2])

    plt.xticks([], plt.yticks([]))

    plt.show()

elif kq == 4:

    for img in path4:

        n = cv2.imread(img)

        cv_img.append(n)

    plt.figure(figsize=(15,6))

    plt.subplot(131)

    plt.imshow(cv_img[0])

    plt.xticks([], plt.yticks([]))

    plt.subplot(132)

    plt.imshow(cv_img[1])

    plt.xticks([], plt.yticks([]))

    plt.subplot(133)

    plt.imshow(cv_img[2])

    plt.xticks([], plt.yticks([]))
```

```
plt.show()

elif kq == 5:

    for img in path5:

        n = cv2.imread(img)

        cv_img.append(n)

    plt.figure(figsize=(15,6))

    plt.subplot(131)

    plt.imshow(cv_img[0])

    plt.xticks([], plt.yticks([]))

    plt.subplot(132)

    plt.imshow(cv_img[1])

    plt.xticks([], plt.yticks([]))

    plt.subplot(133)

    plt.imshow(cv_img[2])

    plt.xticks([], plt.yticks([]))

    plt.show()

elif kq == 6:

    for img in path6:

        n = cv2.imread(img)

        cv_img.append(n)

    plt.figure(figsize=(15,6))

    plt.subplot(131)

    plt.imshow(cv_img[0])

    plt.xticks([], plt.yticks([]))

    plt.subplot(132)

    plt.imshow(cv_img[1])

    plt.xticks([], plt.yticks([]))
```

```
plt.subplot(133)

plt.imshow(cv_img[2])

plt.xticks([], plt.yticks([]))

plt.show()

elif kq == 7:

    for img in path7:

        n = cv2.imread(img)

        cv_img.append(n)

    plt.figure(figsize=(15,6))

    plt.subplot(131)

    plt.imshow(cv_img[0])

    plt.xticks([], plt.yticks([]))

    plt.subplot(132)

    plt.imshow(cv_img[1])

    plt.xticks([], plt.yticks([]))

    plt.subplot(133)

    plt.imshow(cv_img[2])

    plt.xticks([], plt.yticks([]))

    plt.show()

elif kq == 8:

    for img in path8:

        n = cv2.imread(img)

        cv_img.append(n)

    plt.figure(figsize=(15,6))

    plt.subplot(131)

    plt.imshow(cv_img[0])

    plt.xticks([], plt.yticks([]))
```

```
plt.subplot(132)

plt.imshow(cv_img[1])

plt.xticks([], plt.yticks([]))

plt.subplot(133)

plt.imshow(cv_img[2])

plt.xticks([], plt.yticks([]))

plt.show()

elif kq == 9:

    for img in path9:

        n = cv2.imread(img)

        cv_img.append(n)

    plt.figure(figsize=(15,6))

    plt.subplot(131)

    plt.imshow(cv_img[0])

    plt.xticks([], plt.yticks([]))

    plt.subplot(132)

    plt.imshow(cv_img[1])

    plt.xticks([], plt.yticks([]))

    plt.subplot(133)

    plt.imshow(cv_img[2])

    plt.xticks([], plt.yticks([]))

    plt.show()

elif kq == 10:

    for img in path10:

        n = cv2.imread(img)

        cv_img.append(n)

    plt.figure(figsize=(15,6))
```



```
plt.subplot(131)

plt.imshow(cv_img[0])

plt.xticks([], plt.yticks([]))

plt.subplot(132)

plt.imshow(cv_img[1])

plt.xticks([], plt.yticks([]))

plt.subplot(133)

plt.imshow(cv_img[2])

plt.xticks([], plt.yticks([]))

plt.show()

elif kq == 11:

    for img in path11:

        n = cv2.imread(img)

        cv_img.append(n)

    plt.figure(figsize=(15,6))

    plt.subplot(131)

    plt.imshow(cv_img[0])

    plt.xticks([], plt.yticks([]))

    plt.subplot(132)

    plt.imshow(cv_img[1])

    plt.xticks([], plt.yticks([]))

    plt.subplot(133)

    plt.imshow(cv_img[2])

    plt.xticks([], plt.yticks([]))

    plt.show()

elif kq == 12:

    for img in path12:
```

```
        n = cv2.imread(img)

        cv_img.append(n)

    plt.figure(figsize=(15,6))

    plt.subplot(131)

    plt.imshow(cv_img[0])

    plt.xticks([], plt.yticks([]))

    plt.subplot(132)

    plt.imshow(cv_img[1])

    plt.xticks([], plt.yticks([]))

    plt.subplot(133)

    plt.imshow(cv_img[2])

    plt.xticks([], plt.yticks([]))

    plt.show()

elif kq == 13:

    for img in path13:

        n = cv2.imread(img)

        cv_img.append(n)

    plt.figure(figsize=(15,6))

    plt.subplot(131)

    plt.imshow(cv_img[0])

    plt.xticks([], plt.yticks([]))

    plt.subplot(132)

    plt.imshow(cv_img[1])

    plt.xticks([], plt.yticks([]))

    plt.subplot(133)

    plt.imshow(cv_img[2])

    plt.xticks([], plt.yticks([]))
```

```
plt.show()

elif kq == 14:

    for img in path14:

        n = cv2.imread(img)

        cv_img.append(n)

    plt.figure(figsize=(15,6))

    plt.subplot(131)

    plt.imshow(cv_img[0])

    plt.xticks([], plt.yticks([]))

    plt.subplot(132)

    plt.imshow(cv_img[1])

    plt.xticks([], plt.yticks([]))

    plt.subplot(133)

    plt.imshow(cv_img[2])

    plt.xticks([], plt.yticks([]))

    plt.show()

else:

    print("k tim dc anh")

print(type(kq))

print(kq)
```