ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phòng Đào tạo Sau đại học & Khoa học công nghệ

TỐNG DUY TÂN

LỚP: KHOA HỌC MÁY TÍNH KHÓA 12 (ĐỢT 1)

BÁO CÁO

ĐỒ ÁN CUỐI KHÓA MÔN HỌC

NHẬN DẠNG THỊ GIÁC VÀ ỨNG DỤNG

Giảng viên: TS. LÊ ĐÌNH DUY

TS. NGUYỄN TẤN TRẦN MINH KHANG

TP. Hồ Chí Minh – Tháng 12/2017

MỤC LỤC

MỞ ĐẦU

Kính chào thầy,

Lời mở đầu em xin gửi lời cảm ơn chân thành đến thầy và thầy Khang đã luôn tận tâm dạy dỗ và truyền đạt những kinh nghiệm vô cùng quý báu cho chúng em trong suốt thời gian học vừa qua. "Nhận dạng thị giác và ứng dụng" có thể nói là một môn học khá mới mẻ đối với cá nhân em, ban đầu em đã có một chút bối rối và cảm thấy thật khó để có thể hiểu được nó một cách thấu đáo. Nhưng sau khi được thầy giải thích cụ thể từng bước một trong qui trình qua lần lượt các bài giảng, bên cạnh đó là sự hướng dẫn tận tình từ thầy Khang qua các ví dụ minh hoạ cụ thể đã giúp những bạn từ chưa có kiến thức căn bản gì về Machine Learning như em cũng có thể dễ dàng tiếp cận với nó và nắm bắt một cách nhanh chóng những vấn đề cốt lõi như: quá trình thu thập dữ liệu, phân tích rút trích đặc trưng, xây dựng model, … bằng cách kết hợp giữa lý thuyết và các ví dụ thực tế thầy đã cho chúng em thấy được một bức tranh toàn diện về Machine Learning, nhưng đồng thời thầy cũng đi sâu vào giải thích một cách cụ thể những khía cạnh chính để tạo nên một hệ thống Machine Learning. Đó thật sự là những kiến thức nền tảng vô cùng bổ ích giúp em có thể tiếp tục đào sâu nghiên cứu hơn trong lĩnh vực này. Đây thật sự là một cơ hội rất tốt để nước ta có thể bắt kịp với sự phát triển nhanh chóng của khoa học, công nghệ từ các nước khác trên thế giới, các ứng dụng thông minh xuất hiện ngày càng nhiều đó cũng là cơ hội để em có thể phát triển bản thân, chính vì vậy việc nắm bắt và vận dụng tốt những kiến thức nền tảng mà thầy và thầy Khang đã truyền đạt sẽ giúp em có thể dễ dàng tiếp cận với hướng đi mới này hơn. Em sẽ tiếp tục cố gắng nghiên cứu, tìm hiểu để có thể xây dựng nên một ứng dụng thật sự thông minh và hữu ích trong cuộc sống. Trong khuôn khổ đồ án môn học này em đã cố gắng vận dụng hết những kiến thức mà thầy và thầy Khang đã truyền đạt, cũng như những gì mà em đã tìm hiểu được thêm trong thời gian vừa qua để xây dựng nên chương trình "Nhận diện tên của từng người trong một bức ảnh", đây có thể xem như là sản phẩm đầu tiên của em gửi đến các thầy để thay lời cảm ơn vì tất cả những gì mà em đã được học từ các thầy.

PHẦN 1. BÁO CÁO ĐỒ ÁN

1. CÁC THÔNG TIN CƠ BẢN

- Địa chỉ github lưu source code và các báo cáo: https://github.com/tanUIT/VRA

- Địa chỉ YouTube link đến video minh hoạ cho việc cài đặt, chạy chương trình, và hiển thị kết quả: https://www.youtube.com/watch?v=DXb2FpeuDnI

2. TỔNG QUAN VỀ ĐỒ ÁN

2.1. Mục tiêu

- Hiểu và vận dụng được những kiến thức đã học về 2 loại feature rút trích dữ liệu là HOG và LBP, cùng với 2 thuật toán machine learning là KNN và SVM vào việc xây dựng chương trình demo. Có thể thay đổi điều chỉnh các tham số để tăng hiệu suất chương trình.
- Bên cạnh đó là tìm hiểu, nghiên cứu thêm các hàm thư viện khác của Matlab để mở rộng chương trình demo.
- Nâng cao kỹ năng lập trình bằng ngôn ngữ Matlab.

2.2. Nội dung và cách thức thực hiện

Chương trình demo được xây dựng hoàn toàn bằng ngôn ngữ lập trình Matlab, với chức năng chính là khi ta đưa vào một bức ảnh màu với nhiều khuôn mặt người trong đó thì chương trình sẽ xử lý và trả về cho ta kết quả là bức ảnh ban đầu cùng với tên của từng người tương ứng trong bức ảnh đó.

Chương trình được xây dựng bao gồm 2 phần chính:

- Phần thứ nhất là khi ta đưa vào một tập các ảnh train (ở đây là các ảnh chân dung chỉ có duy nhất một khuôn mặt người) thì chương trình sẽ tự động detect và crop ra khuôn mặt tương ứng sau đó chuyển về ảnh grayscale và lưu vào cơ sở dữ liệu cùng với tên tương ứng của từng khuôn mặt. Tiếp theo đó ta có thể lựa chọn sử dụng feature HOG hoặc LBP để rút trích đặc trưng cho tập ảnh train mà ta đã lưu vào cơ sở dữ liệu. Cuối cùng là sử dụng thuật toán KNN hoặc SVM để xây dựng model cho phép nhận diện khuôn mặt của những người có trong tập dữ liệu train.

- Phần thứ hai là khi ta đưa vào một bức ảnh gồm nhiều khuôn mặt người khác nhau thì chương trình cũng sẽ tự động detect và crop ra khuôn mặt của từng người trong bức ảnh đó. Tiếp theo sẽ sử dụng model mà ta đã xây dựng để tiên đoán tên của từng khuôn mặt đó. Cuối cùng là ráp tên của từng người tương ứng vào trong bức ảnh ban đầu và hiển thị ra.

3. CHI TIẾT QUÁ TRÌNH THỰC HIỆN

3.1. Thiết kế chương trình

3.1.1. Cách tổ chức dữ liệu

Tất cả các file thực thi được đặt trong cùng một thư mục Project_Final. Gồm có tổng cộng 14 file bao gồm cả source code để xử lý các chức năng khác nhau trong chương trình và các file chứa source code xây dựng giao diện chương trình demo.

Ngoài ra trong thư mục Project_Final còn chứa 2 thư mục con là thư mục DataTrain và thư mục DataTest:

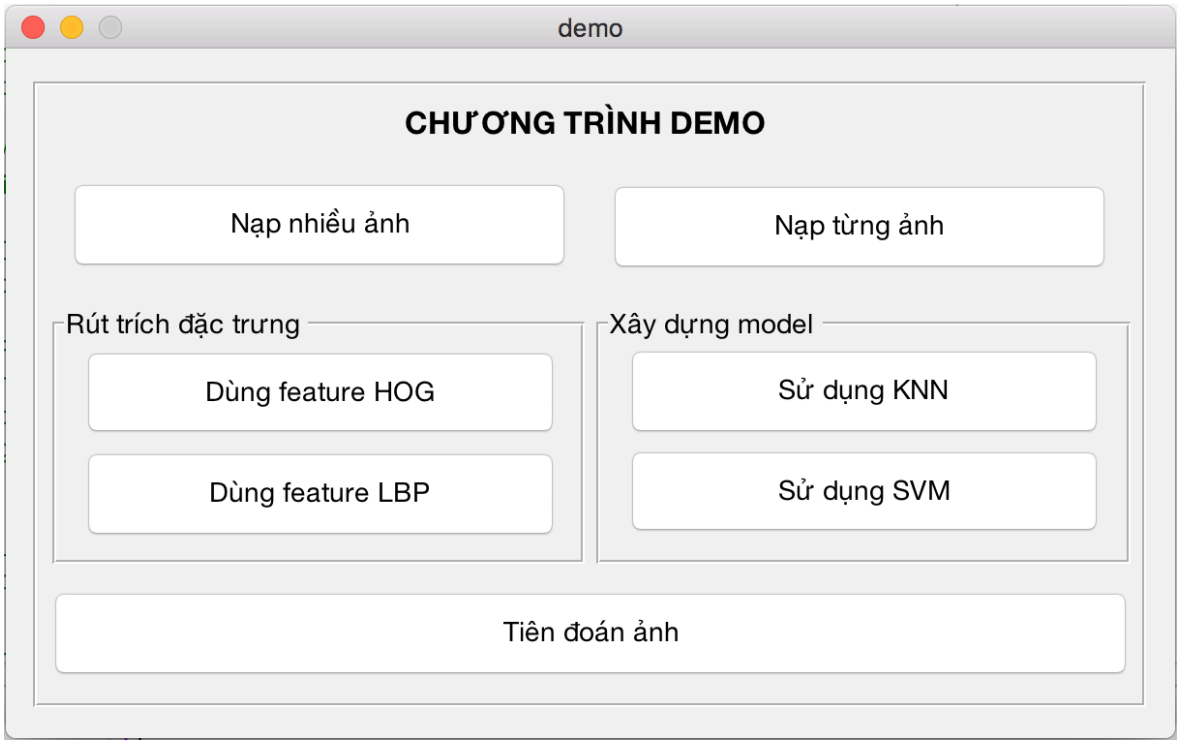
- Thư mục DataTrain: gồm có 8 thư mục con, mỗi thư mục con chứa 25 ảnh chân dung của một người nào đó dùng làm ảnh huấn luyện. Như vậy tổng số lượng ảnh huấn luyện mà ta có là 200 ảnh. Nhãn của mỗi bức ảnh train sẽ chính là tên của thư mục con chứa bức ảnh đó.
- Thư mục DataTest: gồm có tổng cộng 20 ảnh được dùng làm ảnh test, mỗi bức ảnh test sẽ bao gồm 2 hay nhiều người có trong tập ảnh train được chụp chung với nhau.

Mỗi ảnh train sau khi được import vào chương trình sẽ được crop ra từng khuôn mặt tương ứng trong đó và được chuyển thành ảnh grayscale với kích thước là 112 x 92. Sau đó lại được chuyển thành 1 vector cột 10304 chiều. Tất cả các ảnh huấn luyện sẽ được lưu vào tập tin dataTrainImage.mat là một ma trận 10304 x 200. Nhãn tương ứng cho các ảnh huấn luyện được lưu vào tập tin dataTrainLabel.mat là ma trận 200 x 1. Ảnh các khuôn mặt sau khi được crop ra sẽ được lưu vào từng thư mục con có tên tương ứng với tên các thư mục con trong thư mục DataTrain và tất cả được chứa trong 1 thư mục CroppedImages.

3.1.2. Chức năng các file thực thi

- File "selectImage.fig" được dùng để xây dựng giao diện chọn và thêm một ảnh đơn riêng lẻ vào cơ sở dữ liệu train.
- File "selectImage.m" chứa source code xử lý các controll trên giao diện thêm một ảnh train.
- File "readManyImage.m" thực hiện chức năng nạp tất cả ảnh train trong 1 thư mục con của tập DataTrain vào cơ sở dữ liệu train.
- File "cropFace.m" dùng để crop ra tất cả các khuôn mặt có trong bức ảnh đưa vào.
- File "storeNewFace.m" có nhiệm vụ lưu ảnh train mới vào tập tin dataTrainImage.mat và lưu nhãn tương ứng vào tập tin dataTrainLabel.mat
- File "loadData.m" dùng để load tất cả các ảnh train và nhãn tương ứng trong tập tin tin dataTrainImage.mat và dataTrainLabel.mat lên.
- File "extractFeaturesHOG.m" dùng để rút trích đặc trưng của ảnh sử dụng feature HOG.
- File "extractFeaturesLBP.m" dùng để rút trích đặc trưng của ảnh sử dụng feature LBP.
- File "buildModelKNN.m" dùng để xây dựng model cho việc nhận diện tên nhân vật sử dụng thuật toán KNN.
- File "buildModelSVM.m" dùng để xây dựng model cho việc nhận diện tên nhân vật sử dụng thuật toán SVM.
- File "predict.m" được dùng để tiên đoán tên của một nhân vật nào đó trong bức ảnh đưa vào dựa trên model đã được xây dựng.
- File "detectFace.m" nhận diện các khuôn mặt trong bức ảnh test và hiển thị lên với tên tương ứng của từng khuôn mặt.
- File "demo.fig" dùng để xây dựng giao diện chính cho chương trình demo.
- File "demo.m" chứa source code dùng để xử lý các controll trên giao diện chính của chương trình demo.

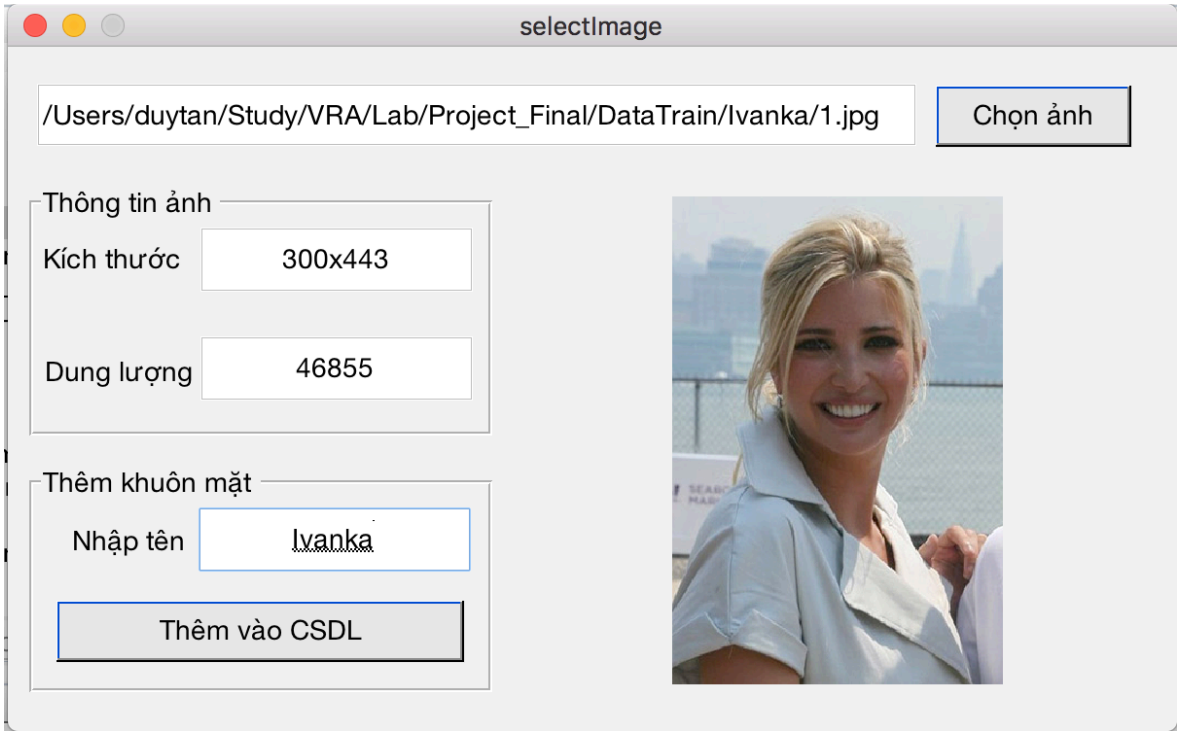
3.1.3. Qui trình hoạt động



Hình 1. Giao diện chính của chương trình demo

Trước tiên để tạo ra dữ liệu train từ các ảnh chân dung, tại giao diện chính của chương trình demo ta có 2 cách thực hiện:

- Cách thứ nhất là ta chọn vào nút nạp từng ảnh, khi đó một cửa số mới sẽ được hiện lên (như hình 2) cho phép ta chọn 1 ảnh chân dung bất kì dùng làm ảnh train, sau khi chọn ảnh chương trình sẽ hiển thị các thông tin cơ bản của bức ảnh đó lên, tiếp theo ta nhập vào tên của người trong bức ảnh đó cũng chính là nhãn tương ứng cho bức ảnh train đó. Sau khi nhập tên xong ta nhấn vào nút "Thêm vào CSDL" để thêm ảnh mới vào tập dữ liệu data train. Chương trình sẽ làm nhiệm vụ detect và crop ra khuôn mặt của người trong bức ảnh chân dung mà ta đã chọn, sau đó resize về kích thước 112 x 92 và biến đổi thành ảnh grayscale sau đó chuyển thành vector cột 10304 chiều. Cuối cùng là thêm nó vào trong tập tin dataTrainImage.mat và tên của người đó sẽ được thêm vào tập tin dataTrainLabel.mat



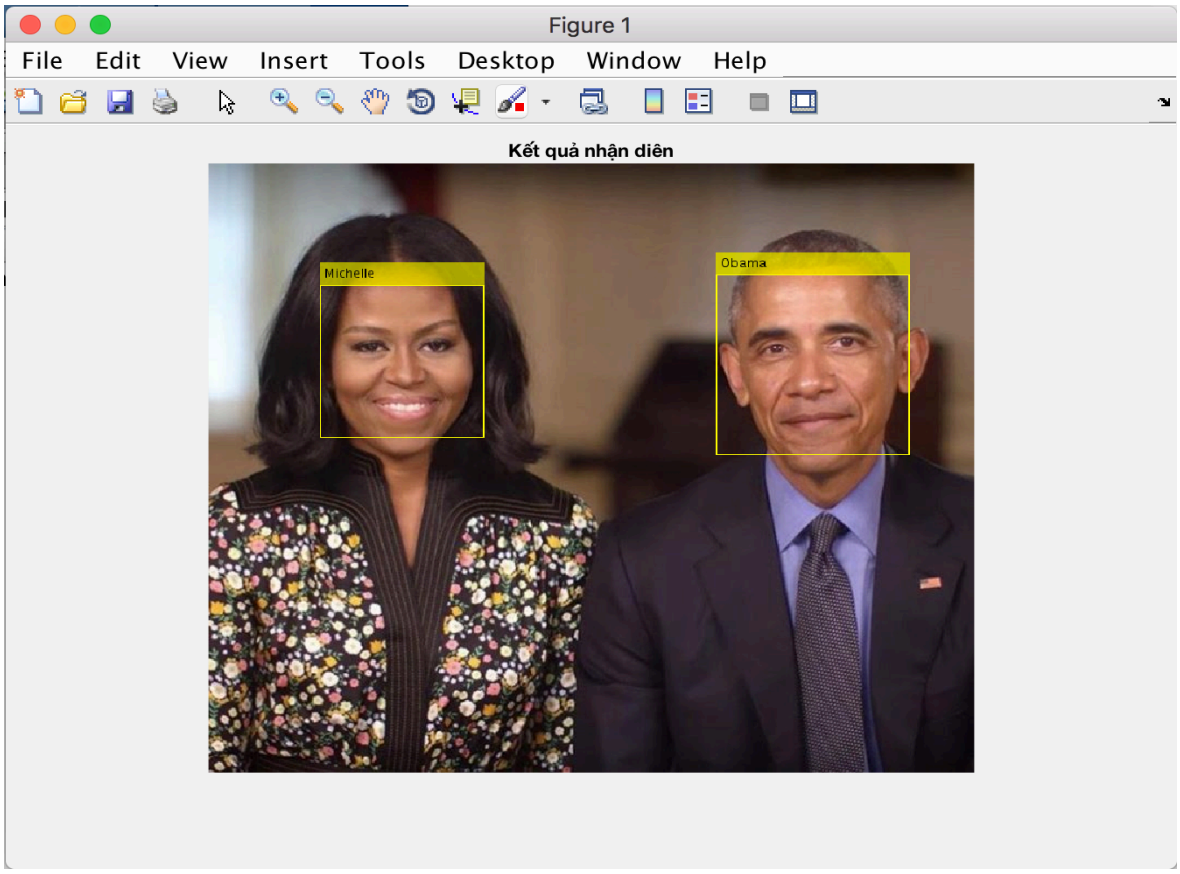
Hình 2. Thêm một ảnh đơn vào tập dữ liệu train

- Cách thứ hai là ta có thể thêm cùng lúc nhiều ảnh vào tập dữ liệu train bằng cách chọn vào nút "Nạp nhiều ảnh" trên giao diện chính của chương trình demo. Lập tức một của sổ mới (như hình 3) sẽ hiện lên cho phép ta chọn một thư mục con bất kỳ trong thư mục DataTrain, sau đó nhấn open. Một thanh tiến trình nhỏ hiện ra thông báo cho ta biết tiến trình nạp các ảnh vào trong tập dữ liệu train. Chương trình sẽ lần lượt duyệt qua từng bức ảnh trong thư mục con được chọn và sau đó thực hiện các bước tương tự như khi ta chọn nạp một ảnh để thêm ảnh mới vào tập dữ liệu train. Sau khi tiến trình hoàn tất thì tất cả các ảnh trong thư mục con sẽ được thêm vào trong tập tin dataTrainImage.mat và nhãn tương ứng của chúng chính là tên của thư mục con được chọn, sẽ được thêm vào trong tập tin dataTrainLabel.mat

Tiếp theo là đến quá trình rút trích đặc trưng, tại khu vực rút trích đặc trưng trên giao diện chính của chương trình demo chúng ta có thể thấy có 2 nút được sử dụng để hỗ trợ cho việc rút trích đặc trưng, ta có thể lựa chọn dùng feature HOG hoặc dùng feature LBP. Nếu sử dụng feature HOG để rút trích đặc trưng thì sao khi nhấn vào nút "Dùng feature HOG" chương trình sẽ tạo ra thêm 1 file FeatureDataHOG.mat trong thư mục gốc của chương trình, tương tự nếu ta chọn vào nút "Dùng feature LBP" thì chương

trình sẽ tạo ra thêm 1 file trong thư mục gốc của chương trình có tên là FeatureDataLBP.mat chứa dữ liệu rút trích tương ứng với feature mà ta đã lựa chọn.

Sau khi rút trích đặc trưng ta tiếp tục quá trình xây dựng model dựa trên dữ liệu đã được rút trích. Trên giao diện chính của chương trình demo, tại khu vực xây dựng model ta có thể thấy chương trình cũng hỗ trợ cho ta hai thuật toán được sử dụng cho việc xây dựng model đó là KNN và SVM. Nếu ta chọn vào nút "Sử dụng KNN" thì chương trình sẽ dựa trên trên dữ liệu rút trích (FeatureDataHOG.mat hoặc FeatureDataLBP.mat) và sử dụng thuật toán KNN để xây dựng model sử dụng cho việc tiên đoán tên của một khuôn mặt ai đó. Sau khi chương trình thực hiện xong việc xây dựng model thì sẽ tạo ra thêm 1 file ModelHOG.mat hoặc ModelLBP.mat tương ứng với loại feature mà ta đã chọn để rút trích đặc trưng. Đối với thuật toán SVM cũng được thực hiện một cách tương tự như thuật toán KNN.



Hình 3. Kết quả nhận diện của chương trình

Sau khi xây dựng xong model thì ta có thể tiến hành kiểm tra chương trình bằng cách đưa vào một bức ảnh gồm nhiều khuôn mặt của nhiều người khác nhau có trong

tập dữ liệu huấn luyện và chương trình sẽ cho chúng ta biết tên của từng người có mặt trong bức ảnh đó. Để thực hiện chức năng này ta chọn vào nút "Tiên đoán ảnh" trên giao diện chính của chương trình, khi đó một cửa số mới hiện ra cho phép ta chọn 1 ảnh test bất kỳ từ tập dữ liệu test. Sau khi chọn xong chương trình sẽ tự động detect và crop ra khuôn mặt của tất cả những người có trong bức ảnh đó, tiếp theo chương trình sẽ tiến hành rút trích đặc trưng theo feature mà ta đã chọn, sau đó sử dụng model mà ta đã xây dựng để tiên đoán tên của từng người trong bức ảnh. Cuối cùng chương trình sẽ hiển thị lên bức ảnh ma ta đã chọn kèm theo tên tương ứng của từng người trong đó như trong hình 3 ở trên.

3.2. Feature HOG

HOG (histogram of oriented gradients) là một feature descriptor được sử dụng trong computer vision và xử lý hình ảnh, dùng để detect một đối tượng. Các khái niệm về HOG được nêu ra từ năm 1986 tuy nhiên cho đến năm 2005 HOG mới được sử dụng rộng rãi sau khi Navneet Dalal và Bill Triggs công bố những bổ sung về HOG. HOG tương tự như các biểu đồ edge orientation, scale-invariant feature transform descriptors (như sift, surf,..), shape contexts nhưng HOG được tính toán trên một lưới dày đặc các cell và chuẩn hóa sự tương phản giữa các block để nâng cao độ chính xác.

Hog được sử dụng chủ yếu để mô tả hình dạng và sự xuất hiện của một object trong ảnh. Bài toán tính toán HOG thường gồm 5 bước sau:

3.2.1. Chuẩn hóa hình ảnh trước khi xử lý

Bước chuẩn hóa này hoàn toàn không bắt buộc, nhưng trong một số trường hợp, bước này có thể cải thiện hiệu suất của bộ mô tả HOG. Có ba phương pháp chuẩn hóa chính mà chúng ta có thể xem xét:

- Quy định về chuẩn Gamma/power : Trong trường hợp này, ta lấy log(p) của mỗi pixel p trong hình ảnh đầu vào.
- Chuẩn hoá gốc-vuông: Ở đây chúng ta lấy $\sqrt{(p)}$ của mỗi pixel p trong hình ảnh đầu vào. Theo định nghĩa, sự bình thường của các căn bậc hai nén các cường độ điểm ảnh đầu vào thấp hơn nhiều so với chuẩn bình thường của gamma.

- Variance normalization: Ở đây, chúng ta tính cần giá trị cường độ điểm ảnh trung bình μ và độ lệch tiêu chuẩn σ của hình ảnh đầu vào. Với mỗi điểm ảnh ta trừ đi giá trị trung bình của cường độ điểm ảnh và sau đó được chuẩn hóa bằng cách chia cho độ lệch chuẩn: p' = (p - μ) /σ

3.2.2. Tính toán gradient theo cả hướng x và y

Để lấy được hình ảnh gradient, chúng ta sẽ sử dụng tích chập (convolution):

Gx = I ⋆ Dx và Gy = I ⋆ Dy

với I là hình ảnh đầu vào, *Dx* là bộ lọc cho chiều x, và *Dy* là bộ lọc cho chiều y.

Sau khi có các ảnh gradient, chúng ta có thể tính toán cường độ gradient của hình ảnh:

$$|G| = \sqrt{Gx2 + Gy2}$$

Cuối cùng, định hướng của gradient cho mỗi pixel trong hình ảnh ban đầu được tính bằng cách:

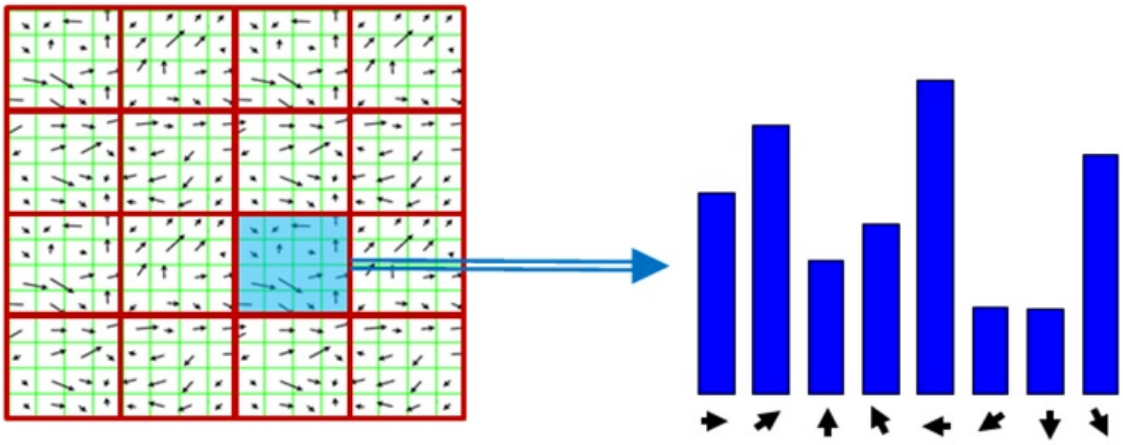
θ = arctan2(Gy, Gx)

Dự vào |G|4 và θ, chúng ta có thể tính được một biểu đồ cường độ gradient, trong đó cột của histogram dựa trên θ và trọng số của mỗi cột của biểu đồ được dựa trên |G|$.

3.2.3. Lấy phiếu bầu cùng trọng số trong các cell

Bây giờ chúng ta cần chia hình ảnh của chúng ta thành các cell và block. Một cell là một vùng hình chữ nhật được xác định bởi số điểm ảnh thuộc mỗi cell. Ví dụ: nếu ta có một hình ảnh 128 x 128 với pixel_per_cell = 4 x 4 thì sẽ có 32 x 32 = 1024 cell, pixel_per_cell = 32 x 32, sẽ có 4 x 4 = 16 cell.

Với mỗi cell trong bức ảnh, ta cần xây dựng 1 biểu đồ cường độ gradient. Mỗi pixcel sẽ được vote vào vào biểu đồ, trọng số của mỗi vote chính là cường độ gradient tại pixel đó. Cuối cùng, mỗi pixel đóng góp một phiếu bầu có trọng số vào biểu đồ - trọng lượng của phiếu chỉ đơn giản là cường độ gradient |G| tại pixel đó. Lúc này, chúng ta có thể thu thập và ghép các biểu đồ này để tạo ra feature vector cuối cùng. Tuy nhiên, ta sẽ chuẩn hóa các block để có được kết quả tốt hơn.



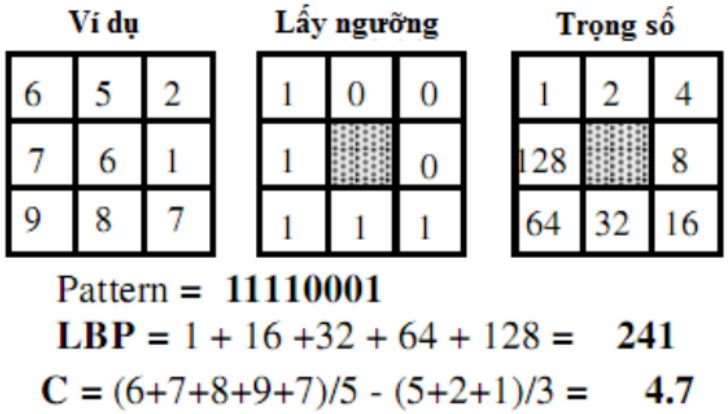
3.2.4. Chuẩn hóa các block

Một lần nữa, ta cần chia các block giống như chia cell ở phía trên. Đơn vị của ta không còn là các điểm ảnh nữa mà là các cell. Người ta thường sử dụng hoặc 2 x 2 hoặc 3 x 3 cell_per_block có được độ chính xác hợp lý trong hầu hết các trường hợp.Các block này sẽ chồng lên nhau. Ví du: ta có 3x3 cells và cell_per_block= 2x2 thì ta sẽ có 4 block. Tiếp đến, ta sẽ tiến hành thu thập và ghép các histogram của cell trong block.

3.2.5. Thu thập tất cả các biểu đồ cường độ gradient định hướng để tạo ra feature vector cuối cùng.

3.3. Feature LBP

LBP là viết tắt của Local Binary Pattern hay là mẫu nhị phân địa phương được Ojala trình bày vào năm 1996 như là một cách đo độ tương phản cục bộ của ảnh. Phiên bản đầu tiên của LBP được dùng với 8 điểm ảnh xung quanh và sử dụng giá trị của điểm ảnh ở trung tâm làm ngưỡng. Giá trị LBP được xác định bằng cách nhân các giá trị ngưỡng với trọng số ứng với mỗi điểm ảnh sau đó cộng tổng lại. Hình 4 minh họa cách tính độ tương phản trực giao (C) là hiệu cấp độ xám trung bình của các điểm ảnh lớn hơn hoặc bằng ngưỡng với các điểm ảnh thấp hơn ngưỡng.



Hình 4. Ví dụ về LBP và độ tương phản cục bộ C

Kể từ khi được đưa ra, theo định nghĩa là bất biến với những thay đổi đơn điệu trong ảnh đen trắng. Để cải tiến phương pháp, bổ sung thêm phương pháp tương phản trực giao địa phương. Hình 4 minh họa cách tính độ tương phản trực giao (C) là ký hiệu cấp độ xám trung bình của các điểm ảnh lớn hơn hoặc bằng ngưỡng với các điểm ảnh thấp hơn ngưỡng. Phân phối hai chiều của mã LBP và độ tương phản cục bộ được lấy làm đặc trưng gọi là LBP/C.

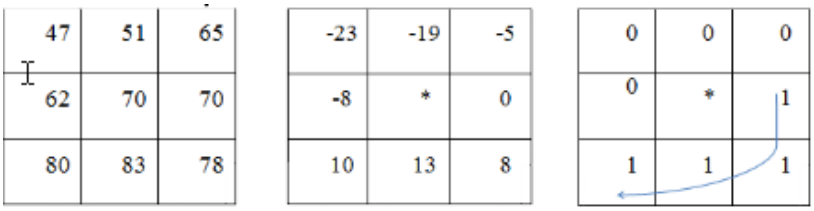
Thuật toán LBP:

Thông tin LBP của pixel tại trung tâm của mỗi khối ảnh sẽ được tính dựa trên thông tin của các pixel lận cận. Có thể tóm tắt các bước tiến hành như sau:

- Bước 1: Xác định bán kính làm việc.
- Bước 2: Tính giá trị LBP cho pixel ở trung tâm (x_c, y_c) khối ảnh dựa trên thông tin của các pixel lân cận:

Trong đó, (g_p) là giá trị grayscale của các pixel lân cận, (g_c) là giá trị grayscale của các trung tâm và (s) là hàm nhị phân được xác định như sau: s(z) = 1 nếu giá trị z ≥0.

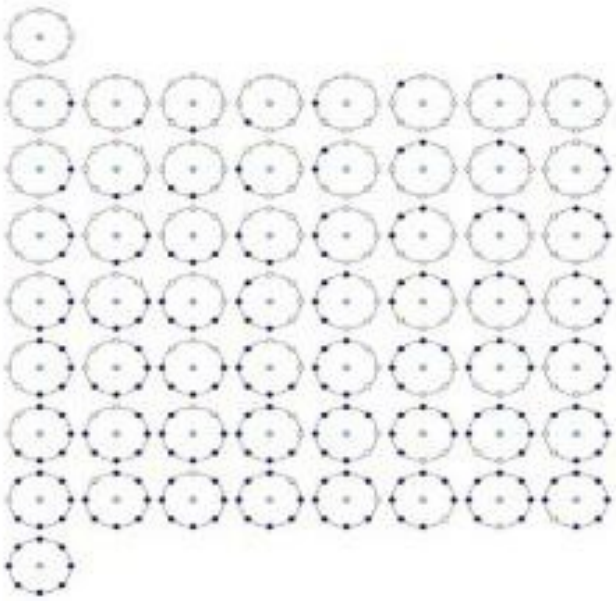
Ví dụ:



Hình 5. Các biến thể của LBP LBP đồng dạng

Một mẫu nhị phân được gọi là đồng dạng khi xét chuỗi bit xoay vòng thì có nhiều nhất là 2 lần thay đổi (transitions) từ giá trị bit 0 sang 1 hoặc từ giá trị bit 1 sang 0. Ví dụ: 00000000 có 0 transitions, 01110000 có 2 transitions, 11001111 có 2 transitions nên đây là uniform LBP. 11001001 có 4 transitions, 01010011 có 6 transitions nên không phải là uniform LBP.

Dựa trên định nghĩa này, bảng ánh xạ cho bán kính làm việc P -neighbours sẽ có P(P-1) + 3 nhãn. Có nghĩa là có 59 nhãn trong trường hợp làm việc với 8-neighbour. Hình vẽ sau đây thể hiện 59 nhãn (mẫu) và minh họa về histogram của đặc trưng LBP đồng dạng.



Hình 6. Bảng thống kê các mẫu của uniform LBP

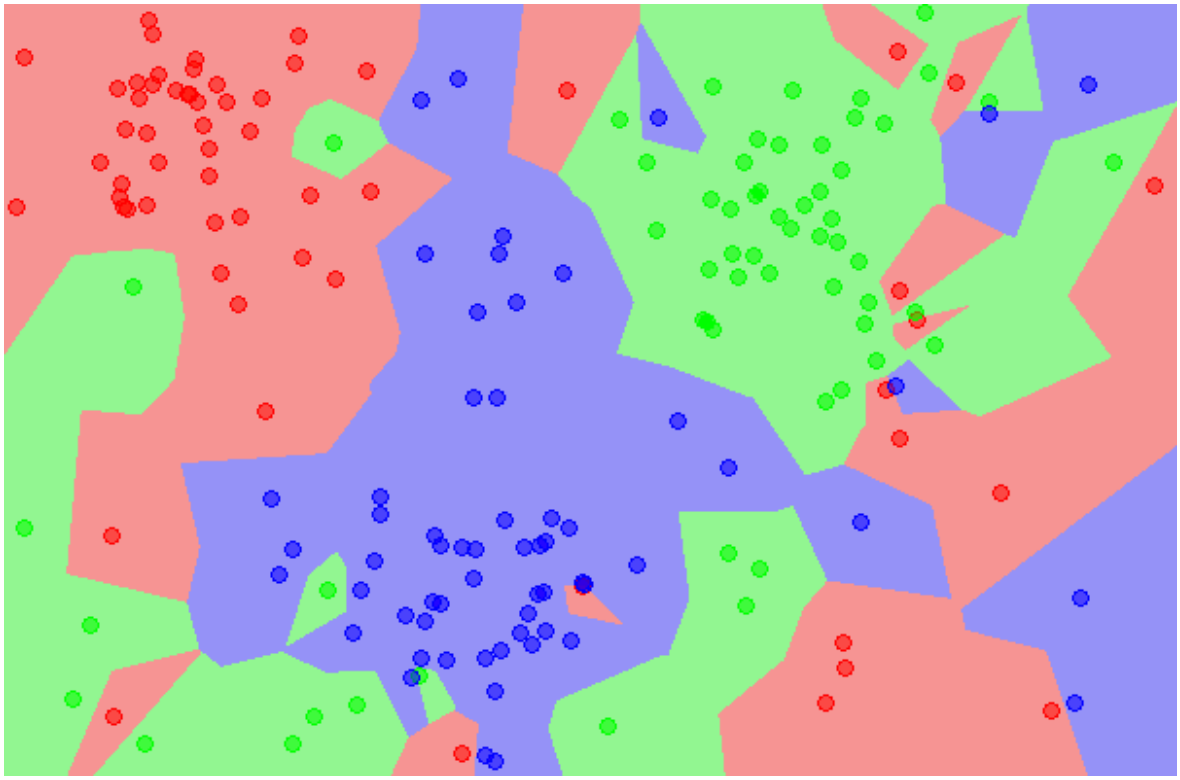
3.4. Thuật toán KNN

K-nearest neighbor là một trong những thuật toán supervised-learning đơn giản nhất (mà hiệu quả trong một vài trường hợp) trong Machine Learning. Khi training, thuật toán này không học một điều gì từ dữ liệu training (đây cũng là lý do thuật toán này được xếp vào loại lazy learning), mọi tính toán được thực hiện khi nó cần dự đoán kết quả của dữ liệu mới. K-nearest neighbor có thể áp dụng được vào cả hai loại của bài toán Supervised learning là Classification và Regression. KNN còn được gọi là một thuật toán Instance-based hay Memory-based learning.

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. Chi tiết sẽ được nêu trong phần tiếp theo.

Trong bài toán Regresssion, 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.

Một cách ngắn gọn, 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. Hình 7 là một ví dụ về KNN trong classification với K = 1.



Hình 7. Bản đồ của 1NN

Ví dụ trên đây là bài toán Classification với 3 classes: Đỏ, Lam, Lục. Mỗi điểm dữ liệu mới (test data point) sẽ được gán label theo màu của điểm mà nó thuộc về. Trong hình này, có một vài vùng nhỏ xem lẫn vào các vùng lớn hơn khác màu. Ví dụ có một điểm màu Lục ở gần góc 11 giờ nằm giữa hai vùng lớn với nhiều dữ liệu màu Đỏ và Lam. Điểm này rất có thể là nhiễu. Dẫn đến nếu dữ liệu test rơi vào vùng này sẽ có nhiều khả năng cho kết quả không chính xác.

3.5. Thuật toán SVM

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 tinh vi nhất cho đến nay. Các hàm công cụ đa dạng của SVM cho phép tạo không gian chuyên đổi để xây dựng mặt phẳng phân lớp.

3.5.1. 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 (supervied learning) được sử dụng cho phân lớp dữ liệu.

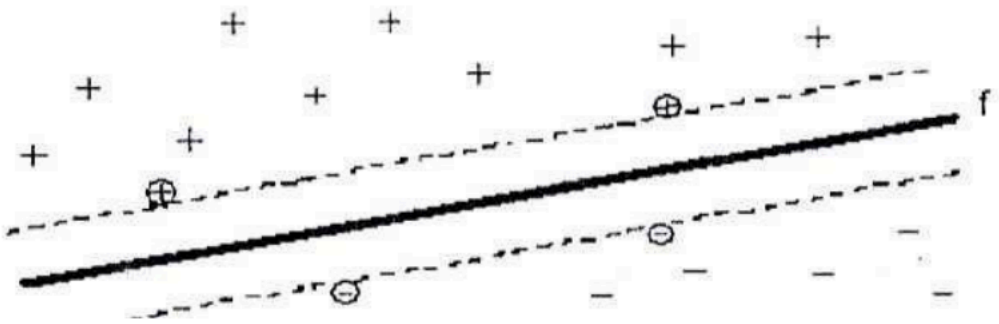
Là 1 phương pháp thử nghiệm, đưa ra 1 trong những phương pháp mạnh và chính xác nhất trong số các thuật toán nổi tiếng về phân lớp dữ liệu.

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.

3.5.2. Ý tưởng

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.

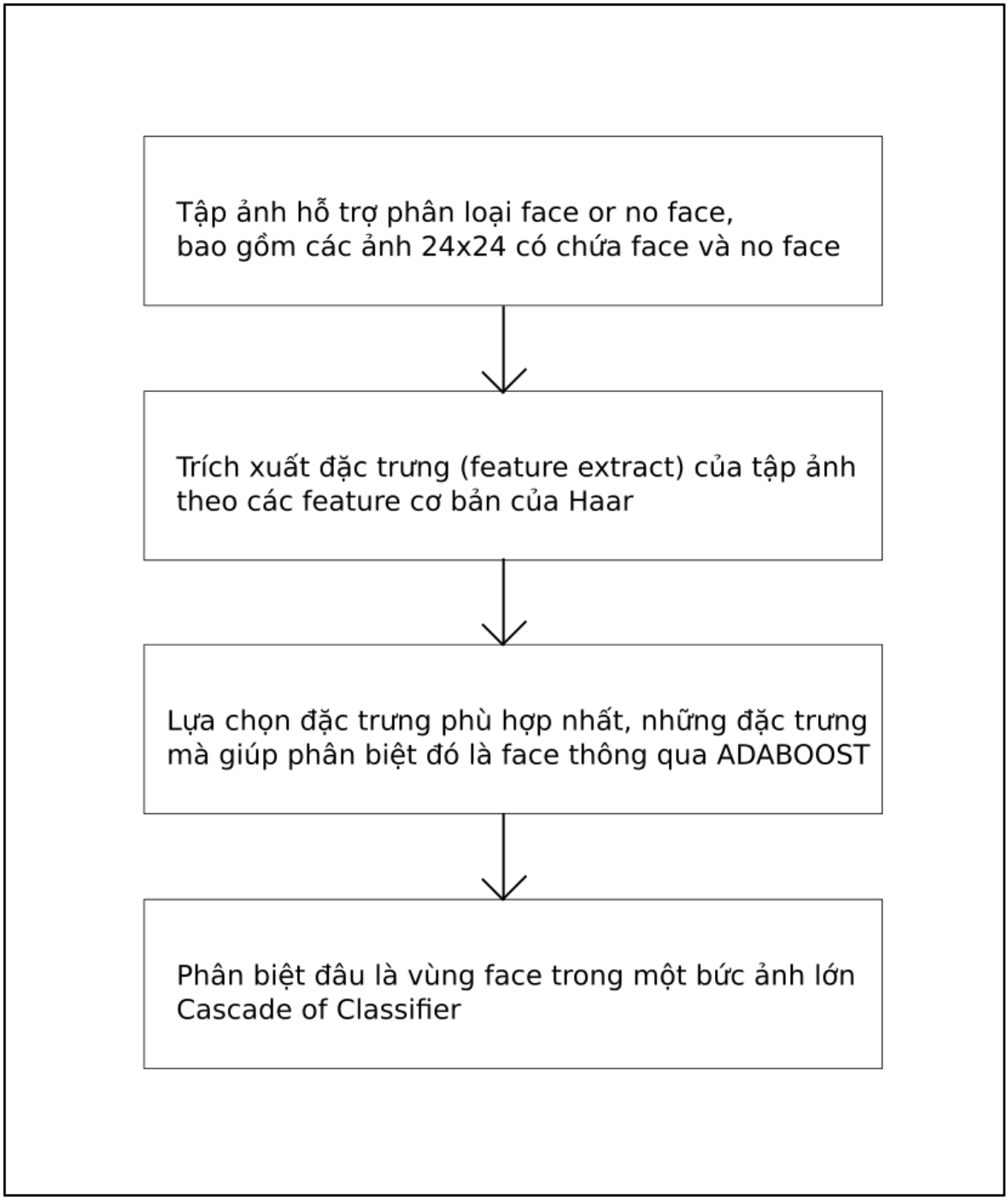
Mục đích của phương pháp SVM là tìm được khoảng cách biên lớn nhất, điều này được minh họa như sau:



Hình 8. Siêu phẳng phân chia dữ liệu học thành 2 lớp + và - với khoảng cách biên lớn nhất. Các điểm gần nhất (điểm được khoanh tròn) là các Support Vector.

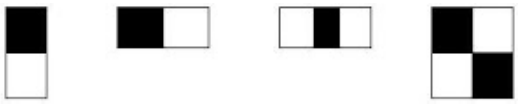
3.6. Thuật toán Viola-Jones

Để phát hiện ra các khuôn mặt có trong bức ảnh em đã sử dụng đối tượng vision.CascadeObjectDetector được hỗ trợ bởi Matlab. Thư viện này sử dụng thuật toán Viola-Jones, đây là phương pháp phát hiện khuôn mặt của Paul Viola và Michael Jones đề xuất vào năm 2001. Phương pháp sử dụng đặc trưng Haar-Like kết hợp với máy phân lớp Ada Boost giúp tăng tốc độ của chương trình.



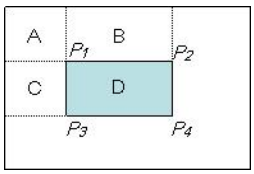
Hình 9. Tiến trình nhận diện khuôn mặt dùng thuật toán Viola-Jones

Các đặc trưng Haar-like là các hình chữ nhật đen trắng để xác định khuôn mặt người. Gồm 4 đặc trưng cơ bản sau:



Hình 10. Đặc trưng Haar-Like

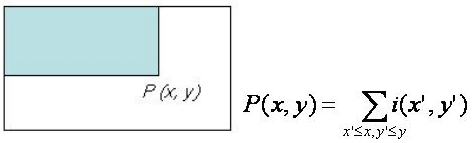
Để tăng tốc độ tính toán và xử lí, Viola-Jones đề xuất một khái niệm mới là Intergral Image (tích phân ảnh). Intergral Image là một mảng hai chiều có kích thước bằng kích thước của ảnh đang xét. Khi đó, tổng mức xám của 1 vùng được tính như sau:



Hình 11. Tích phân ảnh

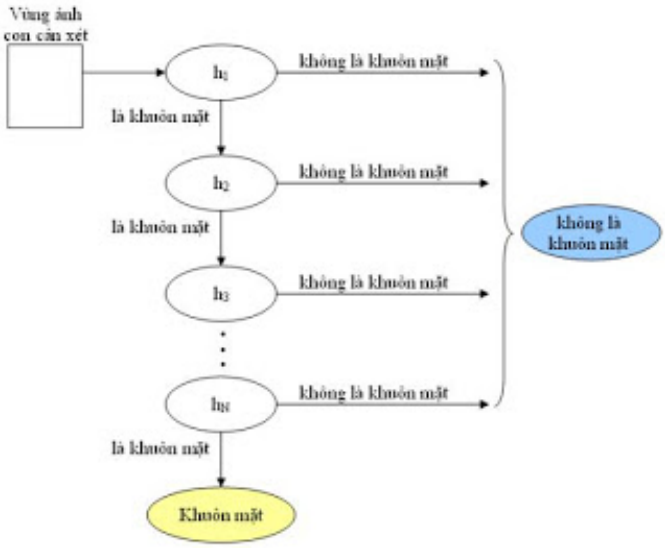
D = A + B + C + D – (A+B) – (A+C) + A = P1(x1, y1) + P2(x2, y2) + P3(x3, y3) + P4(x4, y4) – (P1(x1, y1)+P2(x2, y2)) – (P1(x1, y1)+P3(x3, y3)) + P1(x1, y1)

Trong đó, P(x, y) được tính như hình 11:



Hình 12. Cách tính P(x, y)

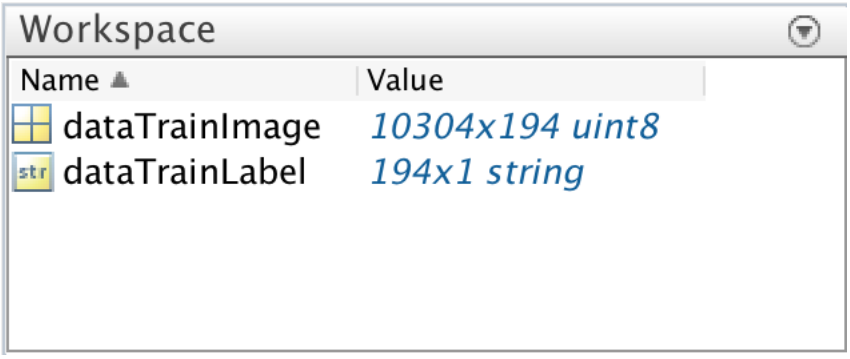
Sau khi tính được tích phân ảnh từ vùng ảnh cần xét, thuật toán Viola-Jones sử dụng bộ phân lớp AdaBoost như hình 12 để loại bỏ các đặc trưng không cần thiết.



Hình 13. Máy phân lớp AdaBoost

4. KẾT QUẢ THỰC NGHIỆM

- Tổng số lượng ảnh dùng để huấn luyện là 200 ảnh chân dung, nhưng sau khi nạp vào chương trình thì thực tế tổng số khuôn mặt được detect và crop ra là 194 khuôn mặt. Như vậy trong quá trình nạp ảnh huấn luyện đã có 6 ảnh bị mất, nguyên nhân là do sử dụng thư viện vision.CascadeObjectDetector với Detection threshold = 6 nhằm mục đích tăng độ chính xác cho việc nhận diện khuôn mặt (loại bỏ các điểm gây nhiễu), nhưng ngược lại một số khuôn mặt không nhìn trực diện sẽ khó được phát hiện ra vì vậy mà chương trình đã bỏ qua 6 khuôn mặt từ các ảnh chân dung này.



Hình 14. Kết quả nạp dữ liệu huấn luyện

- Để kiểm tra chương trình em đã thực hiện thử nghiệm với 20 ảnh chụp chung của nhiều người trong tập dữ liệu huấn luyện, sử dụng 2 feature HOG và LBP kết hợp xen kẽ với 2 thuật toán machine learning là KNN và SVM. Dưới đây là bảng kết quả cụ thể số lượng khuôn mặt được nhận diện đúng với tên của nhân vật, tương ứng với 8 thư mục con trong tập dữ liệu huấn luyện.

Tên	Mark	Obama	Trump	Priscilla	Michelle	Melania	Putin	Ivanka	
Số khuôn mặt	4	9	9	4	5	3	3	3	
HOG+KNN	3	9	8	4	5	3	3	3	95%
HOG+SVM	4	9	8	4	4	3	3	3	95%
LBP+KNN	3	7	8	2	4	2	2	2	75%
LBP+SVM	4	9	9	4	5	3	1	3	95%

Bảng 1. Kết quả thực nghiệm chương trình demo

PHẦN 2. BÁO CÁO MỞ RỘNG

5. Giới thiệu về deep learning

Deep learning là một nhánh nhỏ của machine learning tập trung giải quyết các vấn đề liên quan đến mạng thần kinh nhân tạo nhằm nâng cấp các công nghệ như nhận diện giọng nói, computer vision và xử lý ngôn ngữ tự nhiên. Deep learning đang trở thành một trong những lĩnh vực nổi bậc nhất trong khoa học máy tính. Chỉ trong vài năm, deep learning đã thúc đẩy tiến bộ trong đa dạng các lĩnh vực như nhận thức sự vật (object perception), dịch tự động (machine translation), nhận diện giọng nói,… - những vấn đề từng rất khó khăn với các nhà nghiên cứu trí tuệ nhân tạo.

Trí tuệ nhân tạo có thể được hiểu đơn giản là được cấu thành từ các lớp xếp chồng lên nhau, trong đó mạng thần kinh nhân tạo nằm ở dưới đáy, machine learning nằm ở tầng tiếp theo và deep learning nằm ở tầng trên cùng.

6. Mạng thần kinh nhân tạo

Mạng thần kinh nhân tạo xét trong khía cạnh công nghệ tin có thể được hiểu là một hệ thống các chương trình và cấu trúc dữ liệu mô phỏng cách vận hành của não người. Một mạng thần kinh như vậy thường bao gồm một lượng lớn các vi xử lý hoạt động song song, mỗi vi xử lý chứa đựng một vùng kiến thức riêng và có thể truy cập vào các dữ liệu trong bộ nhớ riêng của mình (đôi khi chúng không nhất thiết phải là phần cứng mà có thể là các phần mềm và giải thuật).

Nếu ví mạng thần kinh nhân tạo với não người thì các neuron thần kinh chính là các node (node là đơn vị thần kinh trong mạng thần kinh nhân tạo – mỗi chiếc máy tính trong mạng thần kinh có thể được xem như 1 node) được kết nối với nhau trong một mạng lưới lớn. Bản thân từng node này chỉ trả lời được những câu hỏi hết sức cơ bản chứ không hề thông minh, nhưng khi được gộp chung với nhau thì chúng lại có sức mạnh xử lý được cả những tác vụ khó. Và điều quan trọng ở đây là bằng những thuật toán phù hợp, chúng ta có thể dạy và huấn luyện được chúng.

7. Machine learning và deep learning

Machine learning là chương trình chạy trên một mạng thần kinh nhân tạo, có khả năng huấn luyện máy tính "học" từ một lượng lớn dữ liệu được cung cấp để giải quyết

những vấn đề cụ thể. Ví dụ nếu ta muốn dạy máy tính nhận biết người đi bộ qua đường thì ta phải đưa ra cho nó một tập các quy tắc chỉ ra mặt đường, biển giao thông, vạch kẻ dành cho người qua đường, người đi bộ, …Nhưng đối với machine learning, ta chỉ cần đưa cho máy tính xem 10.000 ảnh có ảnh người đi bộ qua đường và 10.000 ảnh không có cảnh người đi bộ qua đường để nó tự học theo.

Ngày nay, một phương pháp dạy máy tính mới đang nhanh chóng trở nên phổ biến là deep learning – một loại machine learning sử dụng nhiều lớp thần kinh nhân tạo để phân tích dữ liệu về nhiều chi tiết khác nhau. Chẳng hạn để dạy máy tính nhận diện hình ảnh một con mèo thì chúng ta sẽ lập trình ra nhiều lớp trong mạng thần kinh nhân tạo, mỗi lớp có khả năng xác định một đặc điểm cụ thể của con mèo như râu, vuốt, chân,… rồi cho máy tính xem hàng nghìn bức ảnh mèo (chỉ ra rằng "đây là con mèo") cùng hàng nghìn bức ảnh không phải mèo (chỉ ra rằng "đây không phải mèo"). Khi mạng thần kinh nhân tạo này xem hết các bức ảnh, các lớp node của nó sẽ dần nhận ra râu, vuốt, chân,..., biết lớp nào là quan trọng, lớp nào không. Nó cũng sẽ nhận ra rằng mèo luôn có chân nhưng những con vật không phải mèo cũng có chân nên khi cần xác định mèo, chúng sẽ tìm chân đi kèm những đặc điểm khác như vuốt hay râu.

8. Một số ứng dụng của deep learning

Trên các nền tảng lớn hiện nay như Facebook, Amazon, Netflix,… đều có những hệ thống gợi ý mạnh mẽ giúp tăng cường khả năng tương tác với người dùng. Cụ thể là chúng dựa trên các dữ liệu người dùng phát sinh ra khi dùng để gợi ý thêm những sản phẩm họ sẽ thích (trên các nền tảng mua sắm), những bộ phim họ sẽ muốn xem (như trên Netflix), gợi ý các bài quảng cáo/được tài trợ (trên Facebook) hay các khóa học người học quan tâm (trên các nền tảng học online).

Gần đây, trí tuệ nhân tạo Watson của IBM đã phát hiện ra một loại bệnh mà các bác sĩ đã bó tay không thể tìm ra ở một nữ bệnh nhân. Bằng cách so sánh bộ gen của người phụ nữ này với hơn 20 triệu kết quả nghiên cứu bệnh khác, Watson đã đưa ra kết quả là một chứng leukemia cực kỳ hiếm gặp chỉ trong 10 phút.