HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA AN TOÀN THÔNG TIN

BÀI TẬP LỚN THỰC TẬP CƠ SỞ

ĐỀ TÀI:NHẬN DIỆN BIỂN BÁO GIAO THÔNG

Các sinh viên thực hiện:

B22DCCN041 Phạm Thị Minh Anh

B22DCCN090 Nguyễn Thành Công

B22DCCN702 Bùi Thái Sỹ

B21DCCN210 Đinh Bá Đạt

Giảng viên hướng dẫn: PGS.TS Nguyễn Trọng Khánh

HÀ NỘI 2025

MỤC LỤC

DANH MỤC CÁC HÌNH VẼ

DANH MỤC CÁC BẢNG BIỂU

DANH MỤC CÁC TỪ VIẾT TẮT

Từ viết tắt	Thuật ngữ tiếng Anh/Giải thích	Thuật ngữ tiếng Việt/Giải thích
SVM	Support Vector Machine	Mô hình học có giám sát dùng để phân loại và hồi quy.
HOG	Histogram of Oriented Gradients	Histogram hướng của gradient – Bộ mô tả đặc trưng dùng trong xử lý ảnh và thị giác máy tính.
CVXOPT	Convex Optimization	Tối ưu hóa lồi
KKT	Karush-Kuhn-Tucker conditions	Điều kiện Karush-Kuhn-Tucker – Điều kiện cần và đủ trong bài toán lồi để nghiệm là tối ưu trong bài toán tối ưu phi tuyến.

CHƯƠNG 1. GIỚI THIỆU

1.1 Đặt vấn đề

Trong những năm gần đây, việc ứng dụng công nghệ thị giác máy tính (computer vision) vào lĩnh vực giao thông đã thu hút nhiều sự quan tâm, đặc biệt là trong việc phát triển các hệ thống hỗ trợ lái xe thông minh và phương tiện tự hành. Một trong những chức năng quan trọng của các hệ thống này là khả năng nhận diện biển báo giao thông tự động và chính xác. Việc phát hiện và phân loại chính xác biển báo giúp hỗ trợ người lái đưa ra quyết định kịp thời, đồng thời đảm bảo an toàn cho tất cả người tham gia giao thông, nâng cao tính thông minh cho phương tiện di chuyển.

1.2 Lý do chọn đề tài

Đề tài "Nhận diện biển báo giao thông" bằng SVM kết hợp với HOG được lựa chọn dựa trên tính ứng dụng cao trong thực tiễn, đồng thời có thể tiếp cận bằng các kỹ thuật học máy truyền thống với chi phí tính toán hợp lý. Mặc dù hiện nay các phương pháp học sâu như CNN đang đạt được độ chính xác vượt trội trong các bài toán phân loại hình ảnh, chúng thường yêu cầu lượng dữ liệu lớn, thời gian huấn luyện dài và tài nguyên phần cứng mạnh, điều này gây khó khăn khi triển khai trên các hệ thống nhúng hoặc các thiết bị giới hạn về tính toán. Trong khi đó, các phương pháp truyền thống như kết hợp HOG và SVM tuy đơn giản nhưng vẫn mang lại hiệu quả tốt trong nhiều trường hợp, đặc biệt là các ứng dụng yêu cầu tốc độ xử lý nhanh và tài nguyên hạn chế.

1.3 Mục tiêu nghiên cứu

Mục tiêu chính của đề tài là xây dựng một hệ thống có khả năng phát hiện và phân loại tự động các biển báo giao thông đường bộ từ ảnh đầu vào, dựa trên việc kết hợp các kỹ thuật HOG, SVM. Cụ thể, hệ thống cần đạt được những yêu cầu sau:

- Trích xuất đặc trưng hình ảnh hiệu quả bằng kỹ thuật HOG.
- Phân loại chính xác các loại biển báo sử dụng mô hình SVM.
- Phát hiện biển báo ở nhiều kích thước khác nhau trong ảnh bằng kỹ thuật Pyramid Image kết hợp Sliding Window.

1.4 Phạm vi nghiên cứu và phương pháp tiếp cận

Trong khuôn khổ đề tài, hệ thống được xây dựng nhằm nhận diện một số loại biển báo giao thông phổ biến, bao gồm: biển dừng (Stop), biển qua đường dành cho người đi bộ (Crosswalk), biển giới hạn tốc độ (Speed Limit) và sử dụng tập dữ liệu biển báo giao thông từ các nguồn có sẵn.

Về mặt phương pháp, kỹ thuật Histogram of Oriented Gradients (HOG) được sử dụng để trích xuất đặc trưng hình học từ ảnh, giúp mô tả tốt các cạnh, góc và cấu trúc của biển báo. Sau đó, các đặc trưng này được đưa vào mô hình Support Vector Machine (SVM) để phân loại. Để hệ thống có thể phát hiện biển báo với nhiều kích thước khác nhau trong ảnh, kỹ thuật Pyramid Image kết hợp với cửa sổ trượt (Sliding Window) được áp dụng. Điều này giúp tăng khả năng phát hiện biển báo ở cả khoảng cách gần và xa. Ngôn ngữ lập trình Python được lựa chọn để

xây dựng hệ thống, kết hợp các thư viện như OpenCV và scikit-learn, đồng thời sử dụng các file XML để đọc dữ liệu nhãn trong tập huấn luyện.

Tuy nhiên, hệ thống vẫn còn một số hạn chế. Phương pháp HOG kết hợp với SVM có thể gặp khó khăn trong các điều kiện ánh sáng yếu, biển báo bị che khuất hoặc khi xuất hiện nhiều yếu tố gây nhiễu trong ảnh. Bên cạnh đó, nếu muốn triển khai trong các môi trường thời gian thực, tốc độ xử lý cần được tối ưu thêm. Dù còn một số điểm cần cải thiện, phương pháp này vẫn được đánh giá là một giải pháp khả thi, đạt được sự cân bằng giữa độ chính xác, hiệu năng và khả năng hiện thực hóa, đặc biệt phù hợp với các bài toán yêu cầu tính gọn nhẹ và dễ triển khai trong thực tế.

CHƯƠNG 2. CÁC NGHIÊN CỨU LIÊN QUAN

2.1 Đóng góp nền tảng từ Dalal và Triggs (2005)

Bài báo "Histograms of Oriented Gradients for Human Detection" của Dalal và Triggs (2005) là một trong những tài liệu có giá trị nền tảng. Đây là một bài báo khoa học được công bố tại hội nghị CVPR, với nội dung tập trung vào việc đề xuất một đặc trưng mới – HOG và đánh giá hiệu quả của nó trong việc phát hiện người trong ảnh tĩnh. Tuy chủ đề là phát hiện người, nhưng phương pháp HOG + SVM được trình bày trong bài báo có tính khái quát cao và có thể áp dụng cho các bài toán tương tự, trong đó có nhận diện biển báo giao thông.

Mục tiêu chính của bài báo là xây dựng một mô tả đặc trưng hình dạng hiệu quả, phù hợp với các ứng dụng phát hiện đối tượng trong ảnh. Tác giả nhận thấy rằng việc phát hiện người trong ảnh đòi hỏi mô tả tốt về đường biên, cấu trúc cục bộ và tính ổn định trước các nhiễu loạn từ môi trường. Từ đó, họ đề xuất đặc trưng HOG – thống kê hướng gradient trong từng vùng nhỏ của ảnh, kết hợp chuẩn hóa cục bộ, để mô tả chính xác hình dạng của đối tượng. Phương pháp này được kết hợp với SVM tuyến tính để phân loại ảnh chứa người và ảnh không chứa người. Một điểm nổi bật của bài báo là quy trình nghiên cứu bài bản, hệ thống. Tác giả tiến hành nhiều thí nghiệm để đánh giá tác động của các thông số như kích thước ô, số lượng hướng, cách chuẩn hóa và loại phân loại, từ đó tối ưu hiệu suất mô hình. Ngoài ra, bài báo còn đóng góp bộ dữ liệu thử nghiệm INRIA Person Dataset, đồng thời thể hiện tính khả chuyển cao của phương pháp. HOG + SVM được trình bày theo hướng dễ áp dụng, không đòi hỏi các kỹ thuật phức tạp, rất phù hợp với các bài toán như nhận diện biển báo giao thông – nơi đặc trưng hình dạng đóng vai trò then chốt.

Tuy nhiên, bài báo cũng có một số hạn chế. Thứ nhất, phạm vi ứng dụng chủ yếu giới hạn trong phát hiện người, chưa mở rộng sang các đối tượng khác. Thứ hai, chưa phân tích rõ hiệu quả trong điều kiện ảnh kém chất lượng, ánh sáng yếu hay đối tượng bị che khuất – những yếu tố phổ biến trong môi trường thực tế. Cuối cùng, bài báo ít đề cập đến chi phí tính toán và thời gian xử lý – yếu tố quan trọng nếu triển khai trong các hệ thống thời gian thực như giao thông thông minh.

2.2 Phát hiện biển báo quy mô lớn với biến thể đặc trưng HOG

Bài báo *"Large Scale Sign Detection Using HOG Feature Variants"* của Overett và Petersson (NICTA, Úc) là một công trình nghiên cứu thực tiễn, được thiết kế nhằm phục vụ nhu cầu triển khai hệ thống phát hiện biển báo giao thông ở quy mô lớn. Mặc dù không mang tính nền tảng như công trình của Dalal và Triggs, bài báo này lại có đóng góp đáng kể trong việc tối ưu hóa đặc trưng HOG để ứng dụng hiệu quả hơn trong các hệ thống thương mại, nơi tốc độ xử lý và tài nguyên tính toán là yếu tố then chốt.

Mục tiêu chính của bài báo là cải tiến đặc trưng HOG – vốn nổi bật trong việc mô tả hình dạng – để đáp ứng các yêu cầu về thời gian và tài nguyên trong các hệ thống quét hàng loạt hàng nghìn kilomet dữ liệu video từ mạng lưới đường bộ. Tác giả đề xuất hai biến thể là IHOG (Integral HOG) và BHOG (Binary HOG). IHOG tận dụng biểu diễn tích phân ảnh nhằm tăng tốc độ tính toán vector HOG, còn BHOG chuyển đổi các đặc trưng liên tục thành nhị phân để giảm chi phí lưu trữ và tính toán. Cả hai biến thể đều nhằm duy trì hiệu quả phát hiện mà vẫn giảm thiểu độ phức tạp xử lý.

Một điểm mạnh của bài báo là cách tiếp cận rõ ràng, gắn chặt với ứng dụng thực tế. Tác giả đã thực hiện các thử nghiệm trên tập dữ liệu lớn và đánh giá chi tiết về tỷ lệ phát hiện, tốc độ và độ chính xác. Kết quả cho thấy IHOG là lựa chọn cân bằng giữa tốc độ và độ chính xác, trong khi BHOG đặc biệt phù hợp với các thiết bị nhúng hoặc hệ thống có tài nguyên hạn chế, nhờ dung lượng và chi phí tính toán thấp. Bài báo cho thấy hiệu quả giảm đáng kể tỷ lệ dương tính giả mà không làm suy giảm độ chính xác tổng thể – điều rất quan trọng trong môi trường giao thông thật.

Tuy nhiên, bài báo cũng có một số hạn chế. Trước hết, hệ thống vẫn phụ thuộc vào các chiến lược điều chỉnh thủ công các tham số – điều có thể gây khó khăn khi áp dụng cho nhiều loại biển báo hoặc điều kiện môi trường khác nhau. Thứ hai, phương pháp không tận dụng khả năng học đặc trưng tự động như các mô hình học sâu hiện đại. Cuối cùng, tuy bài báo tối ưu hóa HOG về mặt xử lý, nhưng không đề cập sâu đến khả năng phát hiện trong điều kiện thời tiết xấu, biển báo bị mờ, hoặc các tình huống đặc biệt trong giao thông thực tế.

2.3 Phân loại bằng SVM - Maldonado Bascon

Bài báo *"Road-sign Detection and Recognition Based on Support Vector Machines*" của Maldonado-Bascón và cộng sự (2007) là một trong những công trình có ý nghĩa nền tảng trong lĩnh vực nhận diện biển báo giao thông. Được công bố trên tạp chí IEEE Transactions on Industrial Electronics, bài báo tập trung vào việc xây dựng một hệ thống hai giai đoạn: phát hiện vùng nghi ngờ chứa biển báo và phân loại vùng đó bằng mô hình SVM sử dụng đặc trưng HOG.

Cụ thể, ở bước đầu tiên, hệ thống khai thác thông tin màu sắc (màu đỏ, xanh – vốn là các màu đặc trưng của biển báo giao thông) và hình dạng hình học cơ bản như hình tròn, tam giác, để trích xuất các vùng khả nghi trong ảnh. Sau đó, đặc trưng HOG được tính toán cho từng vùng này, và đưa vào bộ phân loại SVM để xác định chính xác loại biển báo.

Phương pháp SVM được lựa chọn vì khả năng phân tách tuyến tính hiệu quả trong không gian đặc trưng cao chiều, đặc biệt khi kết hợp với HOG – vốn mô tả tốt hình dạng và cấu trúc cạnh của đối tượng. Kết quả thực nghiệm cho thấy hệ thống có thể nhận diện biển báo với độ chính xác lên tới 94% trên ảnh thực tế, đồng thời thời gian xử lý cũng đủ nhanh để đáp ứng các yêu cầu của hệ thống thời gian thực.

Một điểm đáng chú ý là tính thực tiễn của nghiên cứu – hệ thống được thử nghiệm trên ảnh chụp từ môi trường thực, và cho thấy khả năng triển khai vào các ứng dụng giao thông thông minh. Mặc dù vậy, bài báo cũng tồn tại một số hạn chế, trong đó nổi bật là khả năng mở rộng đối với các bộ dữ liệu quy mô lớn và đa dạng hơn. Ngoài ra, phương pháp chưa khai thác được các kỹ thuật học đặc trưng tự động, vốn đang là xu hướng chủ đạo trong các mô hình học sâu hiện đại.

2.4 Khả năng phát hiện biển báo ở nhiều kích thước với Image Pyramid

Trong bài báo "*Multi-view traffic sign detection, recognition and 3D localisation*" của Timofte và Van Gool (2009), nhóm tác giả đã đề xuất một quy trình phát hiện và nhận dạng biển báo giao thông hiệu quả, đặc biệt trong môi trường thực tế với nhiều yếu tố thay đổi như ánh sáng, góc nhìn và kích thước biển báo.

Một trong những kỹ thuật quan trọng được sử dụng trong nghiên cứu này là Image Pyramid. Kỹ thuật này tạo ra nhiều phiên bản của ảnh gốc ở các độ phân giải khác nhau, cho phép mô hình quét và phát hiện đối tượng ở mọi tỷ lệ. Khi kết hợp với phương pháp quét cửa sổ trượt (sliding window) và trích xuất đặc trưng HOG trên từng cửa sổ, Image Pyramid giúp

tăng khả năng phát hiện biển báo ở xa hoặc bị thu nhỏ, đồng thời duy trì hiệu suất nhận diện ổn định ở các tỷ lệ khác nhau.

Việc áp dụng Image Pyramid là thành phần thiết yếu giúp hệ thống đạt được tính tổng quát cao hơn trong môi trường thực tế. Tuy nhiên, việc quét nhiều tầng ảnh có thể làm tăng thời gian xử lý, do đó cần có chiến lược tối ưu hóa hợp lý để cân bằng giữa độ chính xác và tốc độ xử lý.

2.5 Tập dữ liệu chuẩn GTSRB và vai trò trong đánh giá mô hình

Tập dữ liệu GTSRB (*German Traffic Sign Recognition Benchmark*), được giới thiệu bởi Stallkamp và cộng sự vào năm 2011, là một bộ dữ liệu tiêu chuẩn được sử dụng rộng rãi trong nghiên cứu nhận diện biển báo giao thông. GTSRB bao gồm hơn 50,000 hình ảnh biển báo giao thông, được thu thập trong các điều kiện thực tế đa dạng, với nhiều loại biển và góc chụp khác nhau.

Mục tiêu chính của việc xây dựng tập dữ liệu này là nhằm tạo ra một bộ mẫu đại diện phong phú cho các tình huống thực tế mà hệ thống nhận diện biển báo giao thông phải đối mặt. Những hình ảnh trong GTSRB thể hiện sự biến đổi lớn về kích thước, góc nhìn, ánh sáng, và các yếu tố gây nhiễu như che khuất hay mờ nhòe, giúp mô hình có thể học và đánh giá một cách toàn diện về khả năng nhận diện trong các điều kiện phức tạp.

Ngoài ra, GTSRB còn đóng vai trò là một tiêu chuẩn chung để so sánh hiệu suất giữa các thuật toán khác nhau. Nhờ đó, nó tạo điều kiện thuận lợi cho việc đánh giá khách quan và thúc đẩy sự phát triển liên tục của các phương pháp nhận diện biển báo, từ các kỹ thuật truyền thống đến các mô hình học sâu hiện đại.

Tuy nhiên, do quy mô lớn và tính đa dạng cao của tập dữ liệu, việc huấn luyện và đánh giá trên GTSRB cũng đặt ra những thách thức nhất định về mặt tính toán, đòi hỏi tài nguyên phần cứng và thời gian xử lý đáng kể. Dù vậy, đây vẫn là một công cụ thiết yếu, góp phần nâng cao độ chính xác và tính ứng dụng thực tiễn của các hệ thống nhận diện biển báo giao thông hiện nay.

CHƯƠNG 3. PHƯƠNG PHÁP ĐỀ XUẤT

Trong đề tài này, nhóm thực hiện đề xuất một hệ thống nhận diện biển báo giao thông dựa trên hai kỹ thuật chính: Histogram of Oriented Gradients (HOG) để trích xuất đặc trưng và Support Vector Machine (SVM) để phân loại. Đây là một hướng tiếp cận cổ điển nhưng vẫn rất hiệu quả đối với các bài toán nhận dạng hình ảnh trong trường hợp số lượng lớp không quá lớn, dữ liệu có chất lượng tương đối và yêu cầu mô hình có khả năng triển khai trên phần cứng hạn chế.

HOG là một phương pháp trích xuất đặc trưng dựa trên việc mô tả hình dạng và đường viền của đối tượng thông qua hướng của gradient cục bộ. Kỹ thuật này đã được chứng minh là rất hiệu quả trong việc mô tả đặc trưng hình học, đặc biệt là trong môi trường có ánh sáng thay đổi hoặc nhiễu ảnh nhất định. Khi áp dụng vào bài toán nhận diện biển báo, HOG giúp trích xuất các thông tin quan trọng về hình dạng đặc trưng của từng loại biển (ví dụ như hình bát giác của biển "STOP", hình tròn của biển giới hạn tốc độ, hoặc hình người đi bộ của biển qua đường).

Sau khi trích xuất đặc trưng bằng HOG, các vector đặc trưng sẽ được đưa vào mô hình phân loại SVM. SVM là một mô hình học máy có khả năng phân tách dữ liệu tuyến tính hoặc phi tuyến bằng cách tìm ra siêu phẳng tối ưu trong không gian đặc trưng. Nhờ đó, SVM có thể phân biệt rõ ràng giữa các loại biển báo dựa trên các đặc trưng được cung cấp từ HOG.

Hệ thống đề xuất có thể tóm gọn theo pipeline như sau: Ảnh đầu vào → Chuẩn hóa → Trích xuất đặc trưng HOG → Phân loại bằng SVM → Dự đoán nhãn biển báo.

Hướng tiếp cận HOG + SVM có ưu điểm là nhẹ, dễ triển khai, ít phụ thuộc vào tài nguyên phần cứng, đồng thời vẫn đạt được độ chính xác cao trong môi trường có kiểm soát. Đây là một lựa chọn hợp lý cho các ứng dụng thực tế như hệ thống hỗ trợ lái xe hoặc thiết bị IoT giao thông thông minh.

3.1 Tổng quan về xử lý ảnh

3.1.1 Khái niệm về xử lý ảnh

Xử lý ảnh là một trong những mảng nghiên cứu quan trọng nhất trong lĩnh vực thị giác máy tính (Computer Vision), là tiền đề cho nhiều ứng dụng mới thuộc lĩnh vực này. Hai nhiệm vụ cơ bản của quá trình xử lý ảnh là nâng cao chất lượng thông tin hình ảnh và xử lý số liệu cung cấp cho các quá trình khác trong đó có việc ứng dụng thị giác vào điều khiển.

Con người thu nhận thông tin qua các giác quan trong đó thị giác đóng vai trò quan trọng nhất. Sự phát triển nhanh của phần cứng máy tính, xử lý ảnh và đồ hoạ đã phát triển mạnh mẽ và ngày càng có nhiều ứng dụng trong cuộc sống. Xử lý ảnh đóng một vai trò quan trọng trong tương tác người - máy.

Quá trình xử lý nhận dạng ảnh là một quá trình thao tác nhằm biến đổi một ảnh đầu vào để cho ra một kết quả mong muốn. Kết quả đầu ra của một quá trình xử lý ảnh có thể là một ảnh chất lượng tốt hơn hoặc một kết luận.

3.1.2 Các bước cơ bản trong xử lý ảnh

Quá trình xử lý một ảnh đầu vào nhằm thu được một ảnh đầu ra mong muốn thường phải trải qua rất nhiều bước khác nhau. Các bước cơ bản của một quá trình xử lý ảnh được thể hiện thông qua hình sau:

Hình 1 – Các giai đoạn cơ bản của xử lý ảnh

Dựa vào sơ đồ trên, có thể nhận thấy quá trình xử lý ảnh thông qua 5 bước chính:

- Thu nhận ảnh đầu vào: đây là bước đầu tiên trong quá trình xử lý ảnh. Thực hiện điều này, cần có bộ thu ảnh, bộ thu ảnh ở đây có thể là máy chụp ảnh đơn sắc hay màu, máy quét ảnh, máy quay... Để lấy ra các bức ảnh cần thiết phục vụ cho quá trình xử lý ảnh.
- Tiền xử lý: ảnh sau khi thu được từ các thiết bị cần được xử lý trước khi đưa vào phục vụ cho quá trình trích chọn đặc điểm. Quá trình cải thiện về độ tương phản, khử nhiễu, khôi phục ảnh, nắn chỉnh hình học… Nhằm tăng chất lượng ảnh và giảm sai số cho việc phân tích bức ảnh sau này.
- Trích chọn đặc trưng: bức ảnh thu được bao gồm nhiều vùng và thành phần. Trong khi đó chỉ cần lấy ra những vùng nhất định để xử lý và lấy ra kết quả mong muốn. Ví dụ: trích ra vùng chứa biển số xe trong một bức ảnh. Điều này làm giúp cho việc xử lý ảnh được nhanh hơn, loại bỏ những phần dư thừa không cần thiết, bức ảnh sẽ được thu gọn tối ưu.
- Hậu xử lý: sau khi có được ảnh tối ưu cần thiết, ảnh sẽ được qua các quá trình chuyển đổi.
- So sánh, đối chiếu và kết luận: đây là quá trình quan trọng nhất. Ảnh sẽ được so sánh, đối chiếu với ảnh mẫu trong trường hợp có mẫu so sánh hoặc dùng phương pháp nội suy đối với trường hợp không có mẫu sẵn để đưa ra kết luận hay một kết quả nào đó trong bức ảnh đó.

3.1.3 Một số khái niệm cơ bản về ảnh

3.1.3.1 Ảnh và điểm ảnh

Điểm ảnh (pixel) được xem như là dấu hiệu hay cường độ sáng tại một tọa độ trong không gian của đối tượng.

Ảnh được xem như là một tập hợp các điểm ảnh, có thể coi ảnh là mảng số thực hai chiều $I_{m\times n}$, có kích thước m×n bao gồm m×n điểm ảnh, trong đó mỗi điểm ảnh có một giá trị xác định biểu thị mức xám tại vị trí m×n tương ứng.

3.1.3.2 Mức xám

Một điểm ảnh có hai đặc trưng cơ bản là vị trí (x,y) của điểm ảnh và độ xám của nó. Dưới đây xem xét một số khái niệm và thuật ngữ thường dùng trong xử lý ảnh.

Mức xám của điểm ảnh là cường độ sáng của nó được gán bằng giá trị số tại điểm đó. Các thang giá trị mức xám thông thường: 16, 32, 64, 128, 256 (mức 256 là mức phổ dụng. Lý do từ kỹ thuật máy tính dùng 1 byte (8 bit) để biểu diễn mức xám. Mức xám dùng 1 byte biểu diễn: 2^8 = 256 mức, tức là từ 0 đến 255).

Ảnh đen trắng: là ảnh có hai màu đen, trắng (không chứa màu khác) với mức xám ở các điểm ảnh có thể khác nhau.

Ảnh nhị phân: ảnh chỉ có 2 mức đen trắng phân biệt tức dùng 1 bit mô tả 2 mức khác nhau. Nói cách khác mỗi điểm ảnh của ảnh nhị phân chỉ có thể là 0 hoặc 1.

Ảnh màu: Trong hệ màu RGB (Red, Blue, Green) để tạo nên thế giới màu, thường dùng 3 byte để mô tả mức màu, khi đó các giá trị màu:

$$2^{8*3} = 2^{24} \approx 16{,}7 \text{ triệu màu.}$$

3.1.3.3 Độ phân giải của ảnh

Độ phân giải của ảnh là mật độ điểm ảnh được ấn định trên một ảnh số được hiển thị. Khoảng cách giữa các điểm ảnh phải được chọn sao cho mắt người vẫn thấy được sự liên tục của ảnh. Việc lựa chọn khoảng cách thích hợp tạo nên một mật độ phân bố, đó chính là độ phân giải và được phân bố theo trục x và y trong không gian hai chiều.

3.1.3.4 Ngưỡng

Ngưỡng là một khái niệm quen thuộc trong xử lý ảnh cũng như rất nhiều giải thuật khác. Ngưỡng được dùng khá hiệu quả trong việc nhận dạng và tìm biên ảnh. Nó dùng để chỉ một giá trị mà người ta dựa vào để phân hoạch một tập hợp thành các miền phân biệt. Trong xử lý ảnh các vùng có giá trị thỏa mãn ngưỡng thường được quy về một giá trị hay nói cách khác là giá trị mức xám tại điểm xét thỏa mãn điều kiện của ngưỡng sẽ được phân hoạch lại cùng một vùng, các vùng không thỏa mãn sẽ được phân hoạch vào vùng còn lại.

Giá trị ngưỡng thường được xác định dựa vào những điểm đặc biệt hoặc dựa vào kinh nghiệm khảo sát. Nếu dựa vào số lượng ngưỡng áp dụng cho cùng một tập dữ liệu người ta sẽ phân ra các phương pháp ứng dụng ngưỡng đơn, ngưỡng kép, hay đa ngưỡng. Nếu dựa vào sự biến thiên của giá trị ngưỡng, trong cùng phạm vi ứng dụng người ta sẽ phân ra các phương pháp dùng ngưỡng cố định và không cố định. Ngưỡng không cố định nghĩa là giá trị của nó sẽ thay đổi tùy theo sự biến thiên của tập dữ liệu theo không gian và thời gian. Thông thường giá trị này được xác định thông qua khảo sát tập dữ liệu bằng phương pháp thống kê.

Hình 2 - Ảnh gốc (trái) và ảnh sau phân ngưỡng (phải)

Xét ví dụ như hình 2, ảnh gốc là một ảnh grayscale có giá trị mức xám của các điểm ảnh nằm trong khoảng từ 0-255, với giá trị ngưỡng là 127, các điểm ảnh nào có giá trị lớn hơn ngưỡng thì cho nó thành màu đen (0), ngược lại là trắng (1) và cho ảnh kết quả.

3.1.4 Các bài toán thường gặp trong xử lý ảnh

3.1.4.1 Khử nhiễu

Có 2 loại nhiễu cơ bản trong quá trình thu nhận ảnh:

- Nhiễu hệ thống: nhiễu có quy luật và có thể khử bằng các phép biến đổi.

- Nhiễu ngẫu nhiên: nhiễu không rõ nguyên nhân, điều này được khắc phục bằng các phép lọc.

3.1.4.2 Trích chọn đặc điểm

Các đặc điểm của đối tượng được trích chọn tùy theo mục đích nhận dạng trong quá trình xử lý ảnh. Có thể nêu ra một số đặc điểm của ảnh sau đây:

- Đặc điểm không gian: Phân bổ mức xám, phân bổ xác suất, phân bổ biên độ.
- Đặc điểm biến đổi: Các đặc điểm loại này được trích chọn bằng việc thực hiện lọc vùng (zonal filtering). Các bộ vùng được gọi là mặt nạ đặc trưng (feature mask) thường là các khe hẹp với hình dạng khác nhau (chữ nhật, tam giác, cung tròn v.v..).
- Đặc điểm biên và đường biên: đặc trưng cho đường biên của đối tượng và do vậy rất hữu ích trong việc trích chọn các thuộc tính bất biến được dùng khi nhận dạng đối tượng. Các đặc điểm này có thể được trích chọn nhờ toán tử gradient, toán tử la bàn, toán tử Laplace, toán tử "chéo không" (zero crossing), …
- Việc trích chọn hiệu quả các đặc điểm giúp cho việc nhận dạng các đối tượng ảnh chính xác, với tốc độ tính toán cao và dung lượng nhớ lưu trữ giảm xuống.

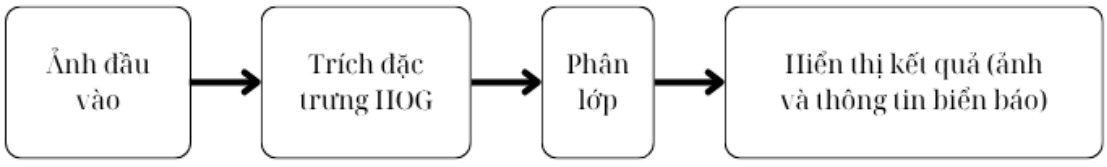
3.1.4.3 Nhận dạng

Nhận dạng tự động mô tả đối tượng, phân loại và phân nhóm các mẫu là những vấn đề quan trọng trong thị giác máy tính, được ứng dụng trong nhiều ngành khoa học khác nhau. Quá trình nhận dạng thường đi kèm với mẫu. Mẫu có thể là ảnh của vân tay, ảnh của một vật nào đó được chụp, một chữ viết, khuôn mặt người hoặc một ký đồ tín hiệu tiếng nói. Khi biết một mẫu nào đó, để nhận dạng hoặc phân loại mẫu đó, có thể làm theo hai hướng:

- Phân loại có mẫu: chẳng hạn phân tích phân biệt trong đó mẫu đầu vào được định danh như một thành phần của một lớp đã xác định, khi đó chỉ cần so sánh đối tượng đầu vào với mẫu để đưa ra kết luận.
- Phân loại không có mẫu: trong đó các mẫu được gán vào các lớp khác nhau dựa trên một tiêu chuẩn đồng dạng nào đó. Các lớp này cho đến thời điểm phân loại vẫn chưa biết hay chưa được định danh. Việc giải quyết bài toán nhận dạng trong những ứng dụng mới, nảy sinh trong cuộc sống không chỉ tạo ra những thách thức về thuật giải, mà còn đặt ra những yêu cầu về tốc độ tính toán. Đặc điểm chung của tất cả những ứng dụng đó là cần nhiều những đặc trưng cần thiết, không thể do chuyên gia đề xuất mà phải được trích chọn dựa trên các thủ tục phân tích dữ liệu.

3.2 Tổng quan về hệ thống phát hiện và phân lớp biển báo giao thông

Quy trình xử lý tổng quát của phương pháp được trình bày như sau:



Hình 3 – Quy trình tổng quát của quá trình phát hiện và phân loại biển báo

Đầu tiên, ảnh đầu vào thu được từ camera sẽ được trích xuất đặc trưng HOG, các đặc trưng này sẽ được đưa qua mô hình SVM đã được huấn luyện trước đó để nhận dạng, đưa ra kết quả cuối cùng về phân lớp của biển báo này.

Nhìn chung, hệ thống có thể chia làm hai quá trình lớn:

- Phát hiện biển báo trong khung hình sử dụng các thuật toán xử lý ảnh.
- Phân lớp biển báo phát hiện được bằng mô hình SVM.

3.3 Trích xuất đặc trưng HOG

3.3.1 Tổng quan về đặc trưng HOG

HOG, viết tắt của Histogram of Oriented Gradient là một phương pháp giúp 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 directions) 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 được gọi là cells, và với mỗi cell sẽ tính toán một histogram về các hướng của gradients cho các điểm nằm trong cell. Ghép các histogram lại với nhau 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.

Đặc trưng HOG cho phép miêu tả tốt các loại biển báo giao thông có hình dạng khác nhau. Vì vậy trong nghiên cứu này sẽ sử dụng đặc trưng HOG để trích xuất các đặc trưng của biển để phục vụ cho thao tác nhận dạng.

3.3.2 Quy trình trích xuất đặc trưng HOG trên ảnh

Bước 1: Tính cường độ và hướng biến thiên tại mỗi pixel theo công thức.

Cường độ: $|G| = \sqrt{G_x^2 + G_y^2}$

Hướng: $\theta = \arctan(\frac{G_y}{G_x})$

Trong đó, G_x và G_y là đạo hàm theo trục x và y.

Bước 2: Phân chia ảnh

Chia ảnh đầu ra ở bước trên thành nhiều khối (block), mỗi khối có số cell bằng nhau, mỗi cell có số pixels bằng nhau. Số khối được tính bằng công thức:

$$n_{block/image} = \left(\frac{W_{image} - W_{block} \times W_{cell}}{W_{cell}} + 1\right) \times \left(\frac{H_{image} - H_{block} \times H_{cell}}{H_{cell}} + 1\right)$$

Trong đó, W_{image}, H_{image}, W_{block}, H_{block}, W_{cell}, H_{cell} lần lượt là chiều rộng, chiều cao của ảnh, khối và ô.

Bước 3: Tính vectơ đặc trưng cho từng khối

Sau khi xác định số block và kích thước mỗi block, cell, để tính toán vector đặc trưng cho từng cell, 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 với độ lớn α(x,y).

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

$$B(x,y) = round\left(\frac{p \times \alpha(x,y)}{\pi}\right) mod\, p$$

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

$$B(x,y) = round\left(\frac{p \times \alpha(x,y)}{2\pi}\right) mod\, p$$

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, nối các vector đặc trưng ô để thu được vector đặc trưng khối. Số chiều vector đặc trưng khối tính theo công thức :

$$size_{block} = n \times size_{cell}$$

Trong đó, n cells là số ô trong khối và size feature/cell là số chiều vectơ đặc trưng của ô bằng 9 (unsigned-HOG) hoặc 18 (signed-HOG).

Bước 4: Chuẩn hóa khối

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. v là vectơ đặc trưng ban đầu của khối, vk là k-norm của v(k = 1, 2), e là hằng số nhỏ.. Khi đó, các giá trị chuẩn hóa có thể tính bằng một trong những công thức sau:

L2-norm:

$$f = \frac{v}{\sqrt{||v_2||^2 + e^2}}$$

L1-norm:

$$f = \frac{v}{(||v_1|| + e)}$$

L1-sqrt:

$$f = \sqrt{\frac{v}{(||v_1||^2 + e)}}$$

Ghép các vector đặc trưng khối sẽ thu được vector đặc trưng HOG cho ảnh. Số chiều vector đặc trưng ảnh tính theo công thức với n là số khối của hình ảnh, size block là số chiều của vectơ đặc trưng khối:

$$size_{image} = n \times size_{block}$$

3.3.3 Áp dụng quy trình trích xuất đặc trưng HOG trên ảnh

Mỗi ảnh được đưa về kích thước 32x32 và tiến hành các bước rút trích đặc trưng HOG. Cụ thể, vùng ảnh đó được chia thành 49 khối, mỗi khối chứa 2*2 cell, mỗi cell chứa 4*4 pixels. Số chiều vectơ đặc trưng tại mỗi ô là 9 (sử dụng 9 bin) và số chiều vectơ đặc trưng mỗi khối là: 9x2x2 = 36 chiều (vì mỗi khối có 2x2 ô). Do đó, số chiều vectơ đặc trưng của ảnh là 49x36 = 1764 chiều.

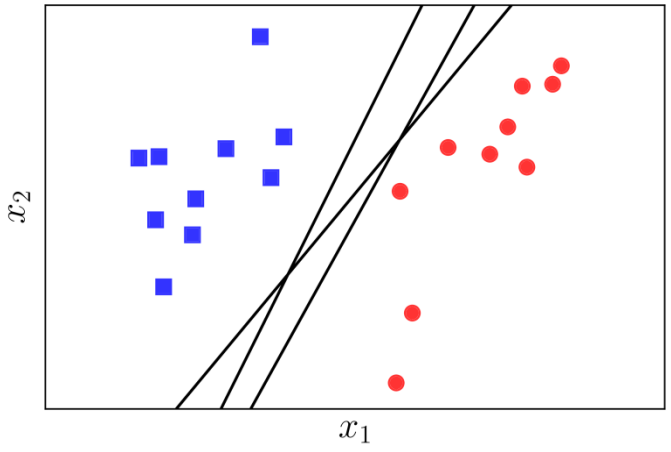
Tất cả các hình ảnh sau khi trải qua quá trình rút trích đặc trưng HOG sẽ được đưa vào mô hình SVM đã được huấn luyện để thực hiện việc dự đoán.

3.4 Thuật toán Support Vector Machine

Support Vector Machine (SVM) là phương pháp phân lớp dựa trên lý thuyết thống kê của Vapnik và Alexei Chervonenkis xây dựng vào năm 1960 và được sử dụng nhiều trong các ứng dụng nhận dạng chữ viết tay, nhận dạng khuôn mặt, phân loại tài liệu, tin sinh học…So với các phương pháp phân loại khác, khả năng phân loại của SVM là tương đương hoặc tốt hơn đáng kể.

Thuật toán SVM hoạt động bằng cách tìm một siêu mặt phẳng tối ưu (optimal hyperplane) để phân chia các điểm dữ liệu thuộc hai lớp khác nhau sao cho khoảng cách từ các điểm gần nhất đến siêu mặt phẳng là lớn nhất. Những điểm dữ liệu gần nhất này được gọi là support vectors.

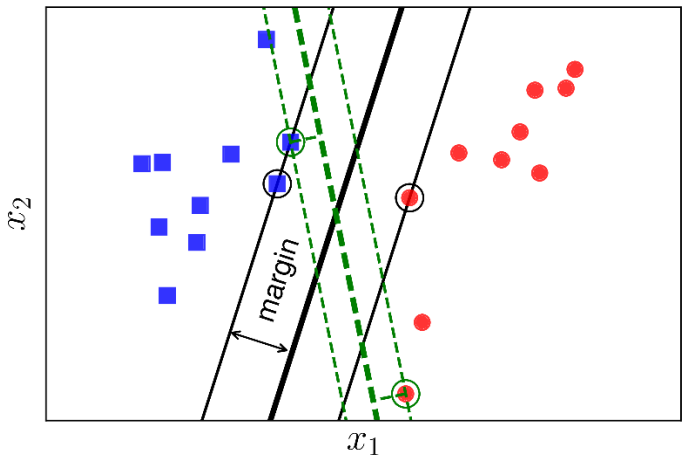
Tiến hành xem xét một bài toán đơn giản: Giả sử có hai class khác nhau được mô tả bởi các điểm trong không gian nhiều chiều, hai classess này là linearly separable, tức tồn tại một mặt siêu phẳng phân chia chính xác hai classes đó. Hãy tìm một siêu mặt phẳng phân chia hai classes đó, tức tất cả các điểm thuộc một class nằm về cùng một phía của siêu mặt phẳng đó và ngược phía với toàn bộ các điểm thuộc class còn lại.



Hình 4 – Các mặt phân cách hai classes linearly separable.

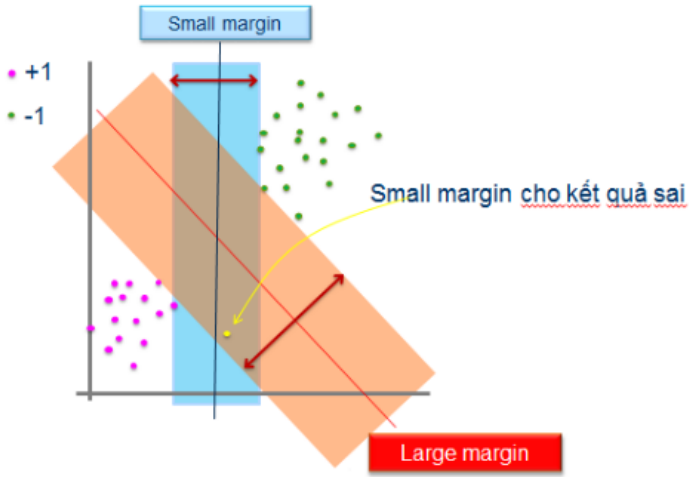
Như Hình 4 có thể thấy, bài toán trên có thể có vô số nghiệm. Vậy trong vô số các mặt phân chia đó, đâu là mặt phân chia tối ưu nhất theo một tiêu chuẩn nào đó?

Vậy cần một đường phân chia sao cho khoảng cách từ điểm gần nhất của mỗi class (các điểm được khoanh tròn) tới đường phân chia là như nhau. Khoảng cách như nhau này được gọi là *margin* (*lề*).



Hình 5 – Margin của hai classes là bằng nhau và lớn nhất có thể

Khi khoảng cách từ đường phân chia tới các điểm gần nhất của mỗi class là như nhau. Xét hai cách phân chia bởi đường nét liền màu đen và đường nét đứt màu lục, rõ ràng đường nét liền màu đen vì nó tạo ra một *margin* rộng hơn. Việc *margin* rộng hơn sẽ mang lại hiệu ứng phân lớp tốt hơn vì *sự phân chia giữa hai classes là rạch ròi hơn*. Margin nhỏ hơn thì khả năng cho kết quả sai sẽ cao hơn

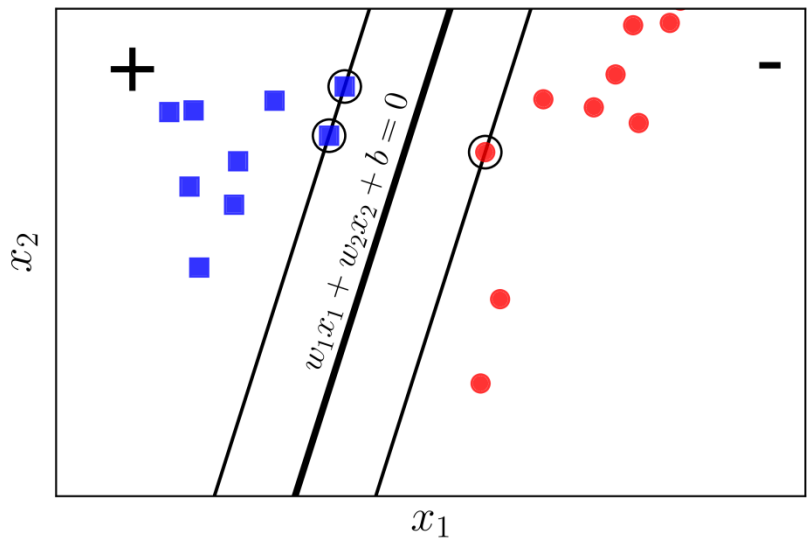


Hình 6 – Ví dụ minh hoạ về độ lớn của lề (margin)

3.4.1 Xây dựng bài toán tối ưu cho SVM

Giả sử rằng các cặp dữ liệu của training set là (x_1, y_1), (x_2, y_2), …, (x_N, y_N) với vector $x_i \in R^d$ thể hiện đầu vào của một điểm dữ liệu và y_i là nhãn của điểm dữ liệu đó. d là số chiều của dữ liệu và N là số điểm dữ liệu. Giả sử rằng nhãn của mỗi điểm dữ liệu được xác định bởi $y_i = 1$ (class 1) hoặc $y_i = -1$ (class 2).

Để giúp dễ hình dung, hãy xét trường hợp trong không gian hai chiều. *Không gian hai chiều để dễ hình dung, các phép toán hoàn toàn có thể được tổng quát lên không gian nhiều chiều.*



Hình 7 – Phân tích bài toán SVM

Giả sử rằng các điểm vuông xanh thuộc class 1, các điểm tròn đỏ thuộc class -1 và mặt $w^T x + b = w_1 x_1 + w_2 x_2 + b = 0$ là mặt phân chia giữa hai classes. Class 1 nằm về *phía dương*, class -1 nằm về *phía âm* của mặt phân chia. Để xác định mặt phân chia cần đi tìm các hệ số w và b.

Có thể thấy rằng với cặp dữ liệu (x_n, y_n) bất kỳ, d là số chiều của không gian, khoảng cách từ điểm đó tới mặt phân chia là:

$$\frac{y_n(w^T x_n + b)}{||W||_2} \text{ với } ||w||_2 = \sqrt{\sum_{i=1}^{d} w_i^2}$$

Điều này có thể dễ nhận thấy vì theo giả sử ở trên, y_n luôn cùng dấu với *phía* của x_n. Từ đó suy ra y_n cùng dấu với $w^T x_n + b$ và tử số luôn là 1 số không âm.

Do đó *margin* được tính là khoảng cách gần nhất từ 1 điểm tới mặt đó (bất kể điểm nào trong hay classes):

$$margin = \min_n \frac{y_n(w^T x_n + b)}{||w||_2}$$

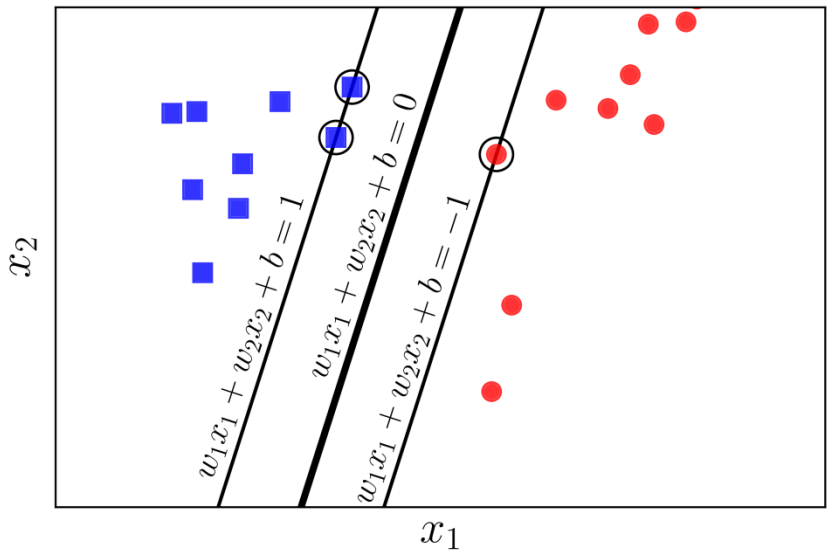
Bài toán tối ưu trong SVM chính là bài toán tìm w và b sao cho *margin* này đạt giá trị lớn nhất:

$$(w, b) = arg \max_{w,b} \left\{ \min_n \frac{y_n(w^T x_n + b)}{||w||_2} \right\} = arg \max_{w,b} \left\{ \frac{1}{||w||_2} \min_n y_n(w^T x_n + b) \right\} (1)$$

Việc giải trực tiếp bài toán này sẽ rất phức tạp, nhưng có cách để đưa nó về bài toán đơn giản hơn. Có thể thấy nếu thay vector hệ số w thành kw và b thành kb trong đó k là một hằng số dương thì mặt phân chia không thay đổi, tức là khoảng cách từ từng điểm đến mặt phân chia không đổi, tức là *margin* không đổi. Dựa trên tính chất này, có thể giả sử:

$$y_n(w^T x_n + b) = 1$$

với những điểm nằm gần mặt phân chia nhất



Hình 8 – Các điểm gần mặt phân cách nhất của hai classes được khoanh tròn.

Như vậy. với mọi n, có:

$$y_n(w^T x_n + b) \geq 1$$

Vậy bài toán tối ưu (1) có thể đưa về bài toán ràng buộc sau:

$$(w, b) = arg \max_{w,b} \frac{1}{||w||_2}$$

$$\text{Subject to: } y_n(w^T x_n + b) \geq 1, \forall n = 1, 2, \dots, N \quad (2)$$

Bằng 1 phép biến đổi đơn giản: nghịch đảo hàm mục tiêu, bình phương nó để được một hàm khả vi và nhân với $\frac{1}{2}$ để biểu thức đạo hàm đẹp hơn thì sẽ đưa bài toán về:

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

$$\text{Subject to: } 1 - y_n(w^T x_n + b) \leq 0, \forall n = 1, 2, \dots, N \quad (3)$$

Trong bài toán (3), hàm mục tiêu là một norm, nên là một hàm lồi. Các hàm bất đẳng thức ràng buộc là các hàm tuyến tính theo w và b, nên chúng cũng là các hàm lồi. Vậy bài toán tối ưu (3) có hàm mục tiêu là lồi, và các hàm ràng buộc cũng là lồi, nên nó là một bài toán lồi. Hơn nữa, nó là một Quadratic Programming. Thậm chí, hàm mục tiêu là *strictly convex* vì $||w||_2^2 = w^T I w$ và I là ma trận đơn vị - là một ma trận xác định dương. Từ đây có thể suy ra nghiệm cho SVM là *duy nhất*.

Đến đây thì bài toán này có thể giải được bằng các công cụ hỗ trợ tìm nghiệm cho Quadratic Programing, ví dụ CVXOPT. Tuy nhiên, việc giải bài toán này trở nên phức tạp khi số chiều d của không gian dữ liệu và số điểm dữ liệu N tăng lên cao.

Người ta thường giải bài toán đối ngẫu của bài toán này. Thứ nhất, bài toán đối ngẫu có những tính chất thú vị hơn khiến nó được giải hiệu quả hơn. Thứ hai, trong quá trình xây dựng bài toán đối ngẫu, người ta thấy rằng SVM có thể được áp dụng cho những bài toán mà dữ liệu không *linearly separable*, tức các đường phân chia không phải là một mặt phẳng mà có thể là các mặt có hình thù phức tạp hơn.

Sau khi tìm được mặt phân cách $w^T x + b = 0$, class của bất kỳ một điểm nào sẽ được xác định đơn giản bằng cách:

$$class(x) = sgn(w^T x + b)$$

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

3.4.2 Bài toán đối ngẫu cho SVM

Nếu một bài toán lồi thoả mãn tiêu chuẩn Slater thì strong duality thoả mãn. Và nếu strong duality thoả mãn thì nghiệm của bài toán chính là nghiệm của hệ điều kiện KKT.

Mà bài toán tối (3) là một bài toán lồi. Hãy chứng minh nó thoả mãn điều kiện Slater. Điều kiện Slater nói rằng, nếu tồn tại w, b thoả mãn:

$$1 - y_n(w^T x_n + b) \leq 0, \forall n = 1, 2, \dots, N$$

thì strong duality thoả mãn.

Việc kiểm tra này tương đối đơn giản. Vì luôn có một siêu mặt phẳng phân chi hai classes nếu hay class đó là linearly separable, tức bài toán có nghiệm, nên feasible set của bài toán tối ưu (3) phải khác rỗng. Tức luôn luôn tồn tại cặp (w_0, b_0) sao cho:

$$1 - y_n(w_0^T x_n + b_0) \leq 0, \forall n = 1, 2, \dots, N$$

$$\Leftrightarrow 2 - y_n(2w_0^T x_n + 2b_0) \leq 0, \forall n = 1, 2, \dots, N$$

Vậy chỉ cần chọn $w_1 = 2w_0$ và $b_1 = 2b_0$, ta sẽ có:

$$1 - y_n(w_1^T x_n + b_1) \leq -1 \ < 0, \forall n = 1, 2, \dots, N$$

Từ đó suy ra điều kiện Slater thoả mãn. Theo lý thuyết tối ưu thì strong duality được đảm bảo. Điều đó cho phép giải bài toán gốc thông qua bài toán đối ngẫu. Để làm được điều này, cần xây dựng hàm Lagrangian cho bài toán (3) để tiến hàng chuyển sang bài toán đối ngẫu và áp dụng điều kiện KKT.

Với bài toán tối ưu tổng quát:

$$x^* = arg\min_x f_0(x)$$

$$subject\ to: f_i(x) \leq 0, i = 1, 2, \dots, m$$
$$h_j(x) = 0, j = 1, 2, \dots, p$$

Với miền xác định $D = (\bigcap_{i=0}^{m} domf_i) \cap \left(\bigcap_{j=0}^{p} domh_j\right)$. Giả sử $D \neq \emptyset$. Lagrangian được xây dựng:

$$L(w, b, v) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{j=1}^{p} v_j h_j(x)$$

Với $\lambda = [\lambda_1,\ \lambda_2, \dots, \lambda_m]$; $v = \left[v_1, v_2, \dots, v_p\right]$ là các vectors và được gọi là dual variables (biến đối ngẫu) hoặc Lagrange multiplier vectors (vector nhân tử Lagrange). Lúc này nếu biến chính $x \in R^n$ thì tổng số biến của hàm số này sẽ là $n + m + p$.

Suy ra Lagrangian của bài toán (3) là:

$$L(w,b,\lambda) = \frac{1}{2}||w||_2^2 + \sum_{n=1}^{N} \lambda_n \big(1 - y_n(w^T x_n + b)\big) \quad (4)$$

Với $\lambda = [\lambda_1, \lambda_2, \dots, \lambda_n]^T$ và $\lambda_n \geq 0, \forall n = 1, 2, \dots, N$.

Hàm đối ngẫu Lagrange được định nghĩa là:

$$g(\lambda) = \min_{w,b} L(w,b,\lambda)$$

Với $\lambda \geq 0$

Việc tìm giá trị nhỏ nhất của hàm này theo w và b có thể được thực hiện bằng cách giải hệ phương trình đạo hàm của $L(w,b,\lambda)$ theo w và b bằng 0:

$$\frac{\partial L(w,b,\lambda)}{\partial w} = w - \sum_{n=1}^{N} \lambda_n y_n x_n = 0 \Rightarrow w = \sum_{n=1}^{N} \lambda_n y_n x_n \quad (5)$$

$$\frac{\partial L(w,b,\lambda)}{\partial b} = -\sum_{n=1}^{N} \lambda_n y_n = 0 \quad (6)$$

Thay (5) và (6) vào (4) thu được $g(\lambda)$:

$$g(\lambda) = \sum_{n=1}^{N} \lambda_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} \lambda_n \lambda_m y_n y_m x_n^T x_m \quad (7)$$

Xét ma trận:

$$V = [y_1 x_1, y_2 x_2, \dots, y_N x_N]$$

Và vector $1 = [1, 1, \dots, 1]^T$, có thể viết lại $g(\lambda)$ dưới dạng:

$$g(\lambda) = -\frac{1}{2} \lambda^T V^T V \lambda + 1^T \lambda \quad (8)$$

Đặt $K = V^T V$, có thể thấy K là một ma trận nửa xác định dương. Thật vậy, với mọi λ, có:

$$\lambda^T K \lambda = \lambda^T V^T V \lambda = ||V\lambda||_2^2 \geq 0$$

Vậy $g(\lambda) = -\frac{1}{2} \lambda^T K \lambda + 1^T \lambda$ là một hàm concave.

Từ đó, kết hợp với hàm đỗi ngẫu Lagrange và các điều kiện ràng buộc của λ sẽ thu được bài toán đối ngẫu Lagrange:

$$\lambda = arg \max_{\lambda} g(\lambda)$$

$$subject\ to: \ \lambda \geq 0 \quad (9)$$

$$\sum_{n=1}^{N} \lambda_n y_n = 0 \quad (6)$$

Đây là một bài toán lồi vì đang đi tìm giá trị lớn nhất của một hàm mục tiêu là *concave* trên một *polyhedron.* Bài toán này cũng được là một Quadratic Programming và cũng có thể được giải bằng các thư viện như CVXOPT.

Trong bài toán đối ngẫu này, số tham số (parameters) phải tìm là N, là chiều của λ, tức là số điểm dữ liệu. Trong khi với bài toán gốc (3), số tham số phải tìm là $d + 1$, là tổng chiều dài của w và b, tức số chiều của mỗi điểm dữ liệu cộng với 1. Trong rất nhiều trường hợp, số điểm dữ liệu có được trong *training set* lớn hơn số chiều dữ liệu rất nhiều. Nếu giải trực tiếp bằng các công cụ giải Quadratic Programming, có thể bài toán đối ngẫu còn phức tạp hơn (tốn thời gian hơn) so với bài toàn gốc. Tuy nhiên, điều hấp dẫn của bài toán đối ngẫu này đến từ phần *Kernel Support Vector Machine (Kernel SVM)*, tức cho các bài toán mà dữ liệu không phải là *linearly separable* hoặc *gần linearly separable*. Ngoài ra, dựa vào tính chất đặc biệt của hệ điều kiện KKT mà SVM có thể được giải bằng nhiều phương pháp hiệu quả hơn.

Vì đây là một bài toán lồi và *strong duality* thoả mãn, nghiệm của bài toán sẽ thoả mãn hệ điều kiện KKT sau đây với biến số là w, b và λ:

$$1 - y_n(w^T x_n + b) \leq 0, \forall n = 1, 2, \dots, N \quad (10)$$

$$\lambda_n \geq 0, \forall n = 1, 2, \dots, N$$

$$\lambda_n(1 - y_n(w^T x_n + b)) = 0, \forall n = 1, 2, \dots, N \quad (11)$$

$$w = \sum_{n=1}^{N} \lambda_n y_n x_n \quad (12)$$

$$\sum_{n=1}^{N} \lambda_n y_n = 0 \quad (13)$$

Từ điều kiện (11) có thể suy ra, với n bất kỳ, hoặc $\lambda_n = 0$ hoặc $y_n(w^T x_n + b) = 0$. Trường hợp thứ hai chính là:

$$w^T x_n + b = y_n \quad (14)$$

do $y_n^2 = 1, \forall n$.

Những điểm thoả mãn (14) chính là những điểm nằm gần mặt phân chia nhất, là những điểm được khoanh tròn trong Hình 8 phía trên. Hai đường thẳng $w^T x_n + b = \pm 1$ *tựa* lên các điểm thoả mãn (14). Vậy nên những điểm (vectors) thoả mãn (14) còn được gọi là các *Support Vectors*.

Một quan sát khác, số lượng những điểm thoả mãn (14) thường chiếm số lượng rất nhỏ trong số N điểm. Chỉ cần dựa trên những *support vectors* này, hoàn toàn có thể xác định được mặt phân cách cần tìm. Nhìn theo một cách khác, hầu hết các λ_n bằng 0. Vậy mặc dù vector $\lambda \in R^N$ có số chiều có thể rất lớn, nhưng số lượng các phần tử khác 0 của nó rất ít. Nói cách khác, vector λ là một sparse vector. Support Vector Machine vì vậy còn được xếp vào *Sparse Models*. Các *Sparse Models* thường có cách giải hiệu quả (nhanh) hơn các mô hình tương tự với nghiệm là *dense* (hầu hết khác 0). Đây chính là lý do thứ hai của việc bài toán đối ngẫu SVM được quan tâm nhiều hơn là bài toán gốc.

Với những bài toán có số điểm dữ liệu N nhỏ, có thể giải hệ điều kiện KKT phía trên bằng cách xét các trường hợp $\lambda_n = 0$ hoặc $\lambda_n \neq 0$. Tổng số trường hợp phải xét là 2^N. Với $N > 50$ (thường là như thế), đây là một con số rất lớn, giải bằng các này sẽ không khả thi. Thay vào đó, sẽ giải bài toán tối ưu (9) bằng CVXOPT và bằng thư viện sklearn.

Sau khi tìm được λ từ bài toán (9), có thể suy ra được w dựa vào (12) và b dựa vào (11) và (13). Rõ ràng chỉ cần quan tâm tới $\lambda_n \neq 0$.

Gọi tập hợp $S = \{n: \lambda_n \neq 0\}$ và N_S là số phần tử của tập S. với mỗi $n \in S$, có:

$$1 = y_n(w^T x_n + b) \Leftrightarrow b + w^T x_n = y_n$$

Mặc dù từ chỉ một cặp (x_n, y_n), có thể suy ra ngay được b nếu đã biết w, một phiên bản khác để tính b thường được sử dụng và được cho là ổn định hơn trong tính toán (numerically more stable) là:

$$b = \frac{1}{N_S}\sum_{n\in S} y_n - w^T x_n + b = \frac{1}{N_S}\sum_{n\in S}\left(y_n - \sum_{m\in S}\lambda_m y_m x_m^T x_n\right) \quad (15)$$

tức trung bình cộng của mọi cách tính b.

Trước đó, w được tính bằng:

$$w = \sum_{m\in S}\lambda_m y_m x_m \quad (16)$$

theo (12)

Để xác định một điểm x mới thuộc vào class nào, cần xác định dấu của biểu thức:

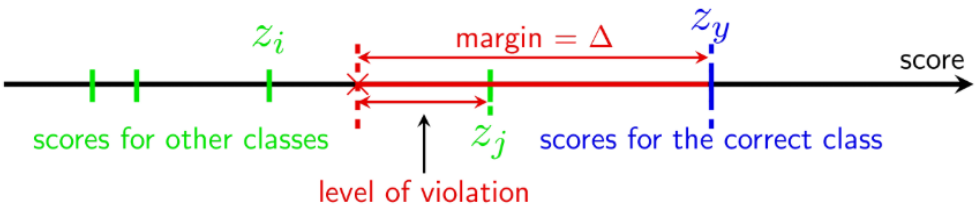
$$w^T x_n + b = \sum_{m\in S}\lambda_m y_m x_m^T x_n + \frac{1}{N_S}\sum_{n\in S}\left(y_n - \sum_{m\in S}\lambda_m y_m x_m^T x_n\right)$$

Biểu thức này phụ thuộc vào cách tính tích vô hướng giữa các cặp vector x và từng $x_n \in S$.

3.4.3 Bài toán Support Vector Machine cho multi-class

Các phương pháp Support Vector Machine như (Hard Margin, Soft Margin, Kernel) đều được xây dựng nhằm giải quyết bài toán Binary Classification, tức bài toàn phân lớp chỉ với hai classes. Các mô hình làm việc với bài toán có 2 classes còn được gọi là Binary Classifier. Một cách tự nhiên để mở rộng các mô hình này áp dụng cho các bài toán multi-classes classification, tức có nhiều class dữ liệu khác nhau là có thể sử dụng nhiều Binary classifier nhưng cách làm này có những hạn chế.

Đối với multi-class trong SVM, trong khi test, class của một input cũng được xác định bởi thành phần có giá trị lớn nhất trong vector. Điều này giống với Softmax Regression sử dụng cross-entropy để ép hai vector xác suất bằng nhau, tức ép phần tử tương với với correct class trong vector xác suất gần với 1, đồng thời, các phần tử còn lại trong vector đó gần với 0. Nói cách khác việc làm này khiến cho phần tử tương ứng với correct class càng lớn hơn các phần tử còn lại càng tốt. Trong khi đó, multi-class SVM xây dựng hàm mất mát dựa trên định nghĩa của biên an toàn, giống như Hard/Soft Margin. Multi-class SVM muốn thành phần ứng với correct class của score vector lớn hơn các phần tử khác, không những thế, nó còn lớn hơn một đại lượng $\Delta > 0$ gọi là biên an toàn.



Hình 9 – Mô tả correct class lớn hơn các class còn lại một khoảng là Δ

Với các xác định biên như trên, multi-class SVM sẽ cho qua những scores nằm về phía trước màu đỏ. Những điểm có score nằm phía màu đỏ sẽ bị xử phạt, và càng vi phạm nhiều thì càng bị xử phạt càng cao.

Để mô tả các mức vi phạm này dưới dạng toán học, trước hết giả sử rằng các thành phần của score vector được đánh số thứ tự từ 1. Các classes cũng được đánh số thứ tự từ 1. Giả sử rằng điểm dữ liệu x đang xét thuộc class y và score vector của nó là vector z. Thế thì score của correct class là z_y và score của các class khác là z_i, $i \neq y$. Xét ví dụ như trong Hình 9 với hai score z_i trong vùng an toàn và z_j trong vùng vi phạm.

- Với mỗi score z_i trong vùng an toàn, loss bằng 0
- Với mỗi score z_j vượt quá điểm an toàn, loss do nó gây ra được tính bằng $z_j - (z_y - \Delta)$

Hay nói một cách đơn giản thì với mỗi score z_j, $j \neq y$, loss của nó gây ra có thể được viết gọn thành:

$$\max(0, \Delta - z_y + z_j)$$

Như vậy, với một điểm dữ liệu $x_n, n = 1, 2, \dots, N$, tổng cộng loss do nó gây ra là:

$$L_n = \sum_{j \neq y_n} \max(0, \Delta - z_{y_n}^n + z_j^n)$$

Với toàn bộ các điểm dữ liệu $X = [x_1, x_2, \dots, x_N]$ loss tổng cộng là:

$$L(X, y, W) = \frac{1}{N} \sum_{n=1}^{N} \sum_{j \neq y_n} \max(0, \Delta - z_{y_n}^n + z_j^n) \qquad (1)$$

Với $y = [y_1, y_2, \dots, y_N]$ là các vector chứa các correct class của toàn bộ các điểm dữ liệu trong training set. Hệ số $\frac{1}{N}$ tính trung bình loss để tránh việc biểu thức này quá lớn gây tràn số máy tính.

Trong trường hợp nếu tìm được w hoàn hảo, tức không có score nào vi phạm thì biểu thức (1) đạt giá trị bằng 0. Hay nói cách khác kW cũng là một nghiệm của bài toán với $k > 1$ bất kì. Việc bài toán có vô số nghiệm và có những nghiệm có những phần tử tiến tới vô cùng khiến cho bài toán rất khó giải. Một phương pháp giải quyết vấn đề này là Regularization vào hàm mất mát giúp việc ngăn chặn hệ số của w trở nên quá lớn. Và người ta nhận thấy rằng Δ có thể được chọn bằng 1 mà không ảnh hưởng nhiều tới chất lượng của nghiệm. Thực tế cho thấy cả hai tham số Δ và λ đều giúp cân bằng giữa *data loss* và *regularization loss*. Thực vậy, độ lớn của các hệ số trong W có tác động trực tiếp lên các *score vectors*, và vì vậy ảnh hưởng tới sự khác nhau giữa chúng. Khi giảm các hệ số của W, sự khác nhau giữa các scores cũng giảm một

tỉ lệ tương tự; và khi tăng các hệ số của W, sự khác nhau giữa các scores cũng tăng lên. Bởi vậy, giá trị chính xác Δ của *margin* giữa các scores trở nên không quan trọng vì có thể tăng hoặc giảm W một cách tùy ý. Việc quan trọng hơn là hạn chế việc W trở nên quá lớn. Việc này đã được điều chỉnh bởi tham số λ.

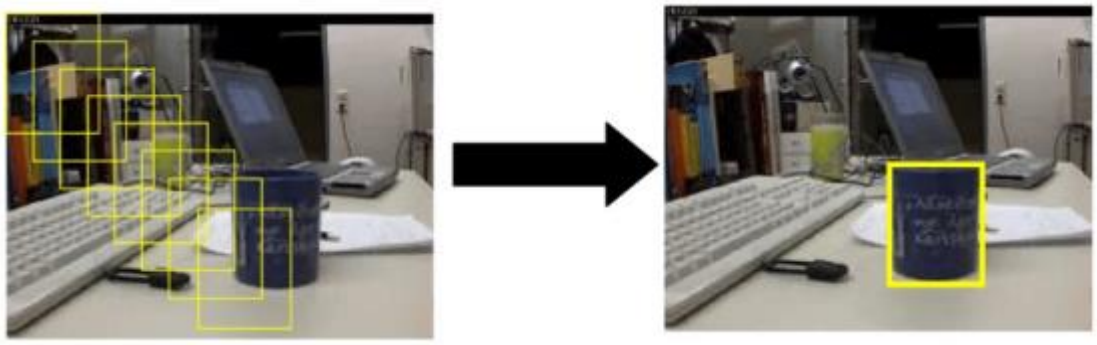
Cuối cùng tối ưu hàm mất mát cho multi-class SVM:

$$L(X, y, W) = \frac{1}{N}\sum_{n=1}^{N}\sum_{j\neq y_n}\max(0, 1 - w_{y_n}^T x_n + w_j^T x_n) + \frac{\lambda}{2}||W||_F^2$$

3.5 Traffic Sign Localization (tạm dịch: Định vị biển báo)

3.5.1 Sliding window

Sliding Window: Một kỹ thuật được sử dụng trong thị giác máy tính và xử lý tín hiệu, nơi một "cửa sổ" có kích thước cố định (hoặc một hình chữ nhật con) được di chuyển trên một hình ảnh (hoặc tín hiệu) để phân tích từng vùng cục bộ tại một thời điểm.



Hình 10 – Sử dụng sliding window để tìm chiếc cốc trong ảnh

Để phát hiện chính xác vị trí biển báo trong ảnh, kỹ thuật Sliding Window sử dụng nhiều kích thước cửa sổ và điều chỉnh bước nhảy (stride) sao cho đạt được sự cân bằng giữa tốc độ xử lý và độ chính xác phát hiện.

- Kích thước cửa sổ khác nhau để phát kiện đối tượng có kích thước khác nhau

Hình 11 – Kích thước cửa sổ khác nhau

- Bước nhảy nhỏ sẽ phát hiện đối tượng với độ chính xác cao hơn nhưng tốn thời gian xử lý, bước nhau lớn thì tốc độ xử lý sẽ cao hơn nhưng có thể bỏ sót vật thể nhỏ.

Hình 12 – Bước nhảy khác nhau

Như vậy cần lựa chọn kích thước cửa sổ phù hợp để phát hiện vật thể ở các kích cỡ khác nhau và bước nhảy (stride) phù hợp để cân bằng giữa tốc độ xử lý và độ chính xác.

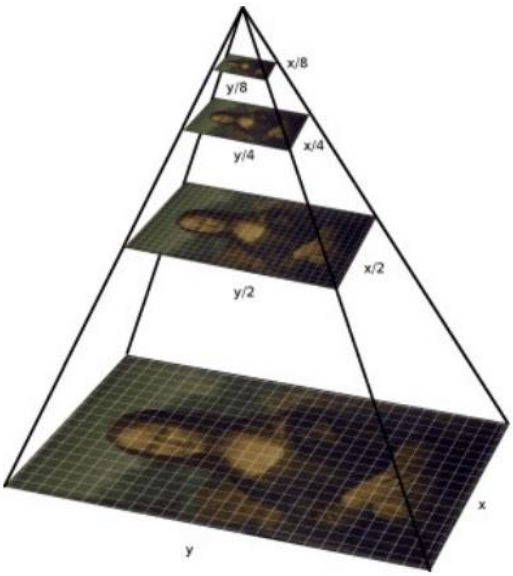
3.5.2 Pyramid Images

Trong nhiều trường hợp, các đối tượng cần quan tâm thường xuất hiện rất nhỏ trong ảnh. Việc áp dụng trực tiếp một cửa sổ có kích thước cố định có thể sẽ không xử lý hiệu quả điều này.

Pyramid Image là một biểu diễn đa tỉ lệ của ảnh đầu vào. Thay vì xử lý ảnh gốc duy nhất, kỹ thuật này tạo ra nhiều phiên bản của ảnh ở các độ phân giải (resolutions) khác nhau, tạo thành một cấu trúc dạng kim tự tháp.

- Ảnh gốc ở đáy kim tự tháp.
- Ảnh càng lên cao thì càng có độ phân giải thấp hơn (nhỏ hơn).

Mỗi mức (level) trong kim tự tháp cho phép mô hình "nhìn" ảnh ở kích thước khác, từ đó phát hiện đối tượng nhỏ hoặc lớn mà ảnh gốc có thể bỏ sót.



Hình 13 – Pyramind Image

Ví dụ với biển báo ban đầu

Ảnh khi scale ở 0.512000000000001.

Hình 14 – scale ở 0.512000000000001.

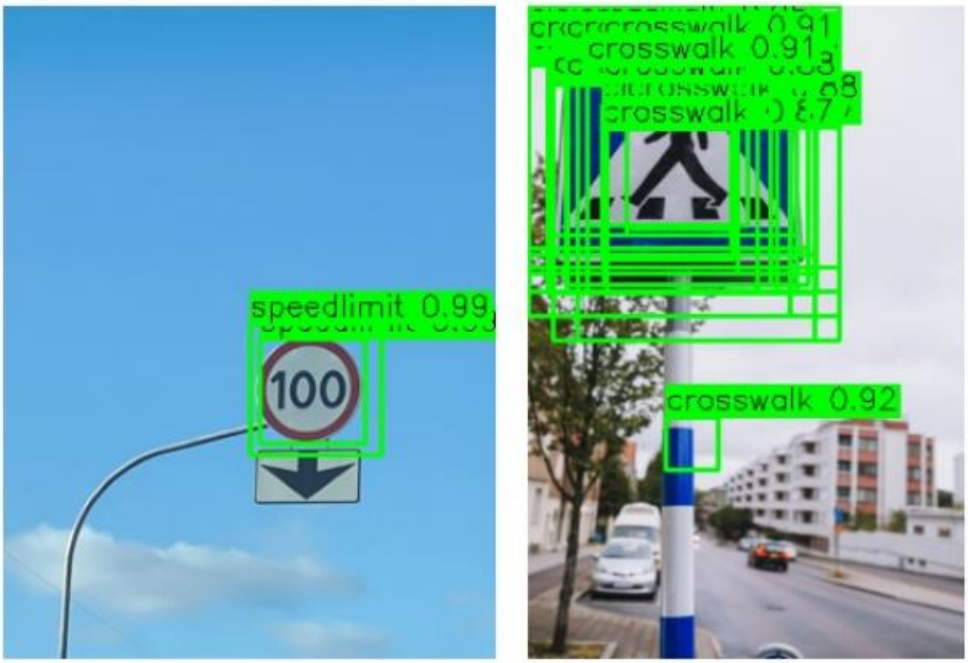
Ảnh khi scale ở 0.327680000000000014.

Hình 15 – scale ở 0.327680000000000014.

Pyramid Image là một công cụ mạnh mẽ trong thị giác máy tính, đặc biệt hữu ích trong các bài toán mà kích thước đối tượng không cố định. Việc sử dụng pyramid giúp mô hình tổng quát hơn và hoạt động hiệu quả trong các tình huống thực tế.

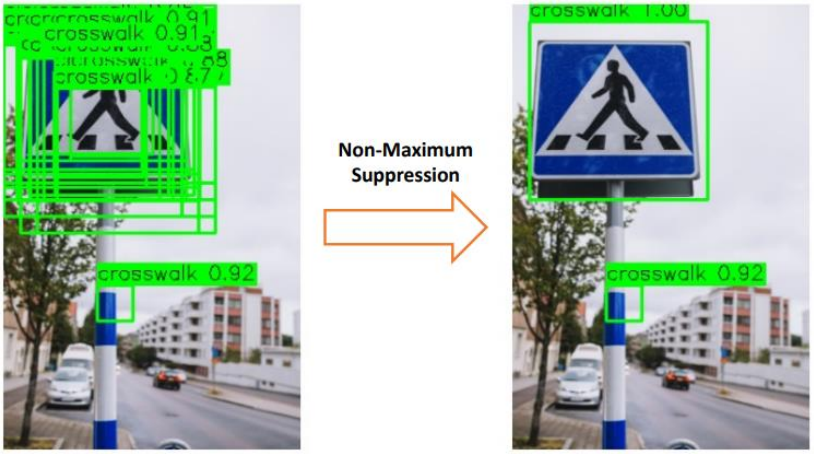
3.5.3 Non-maximum suppression

Thông thường, kết quả đầu ra thô của việc phát hiện đối tượng có thể có rất nhiều hộp giới hạn (bounding boxes) chồng lấp lên nhau, với điểm tin cậy (confidence score) gần như giống nhau. Cần nén (compress) lại thành hộp tốt nhất.



Hình 16 – Vấn đề nhiều hộp giới hạn

Non-maximum suppression (NMS): Một kỹ thuật được sử dụng trong các tác vụ phát hiện đối tượng để loại bỏ (prune) nhiều hộp giới hạn (bounding boxes) bị chồng lấp và cùng chỉ đến một đối tượng.



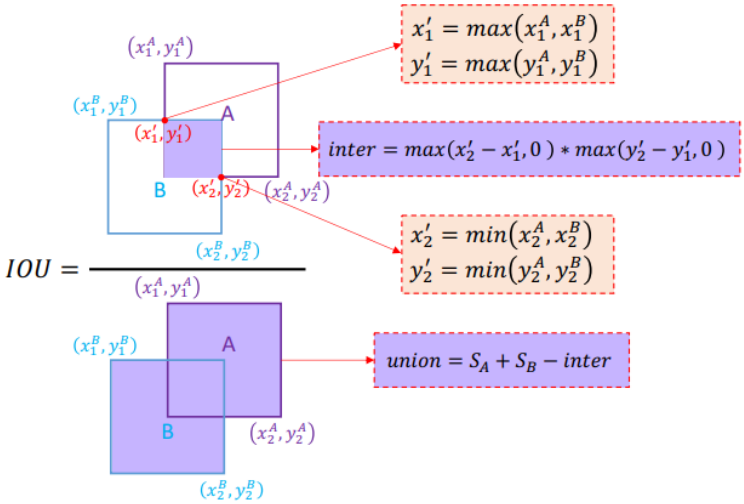
Hình 17 - Non-maximum suppression

Phương pháp loại bỏ không tối đa (Non-maximum suppression):

- Chọn hộp giới hạn (bounding box) có điểm số tin cậy cao nhất (A).
- Tính toán IoU giữa (A) và các hộp giới hạn khác.
- Chỉ giữ lại những hộp giới hạn có IoU nhỏ hơn một ngưỡng (iou_threshold). Nếu không, loại bỏ tất cả các hộp giới hạn còn lại.

Hình 18 - Non-maximum suppression

Trong đó IoU (Intersection over Union) là một chỉ số dùng để đo lường sự chồng lấn giữa hai hộp giới hạn.



Hình 19 Intersection over Union (IoU)

CHƯƠNG 4. CÀI ĐẶT TRIỂN KHAI VÀ ĐÁNH GIÁ

4.1 Các thư viện cần import

```
import time
import os
import cv2
import numpy as np
import matplotlib.pyplot as plt
import xml.etree.ElementTree as ET

from skimage.transform import resize
from skimage import feature
from sklearn.svm import SVC
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

Hình 20 – Các thư viện cần import

import time

- Dùng tính thời gian test các file.

import os

- Dùng để làm việc với hệ thống file: duyệt thư mục, nối đường dẫn, liệt kê file.

import cv2

- Thư viện OpenCV: xử lý ảnh.
- Các chức năng phổ biến:
    - Đọc/hiển thị ảnh (cv2.imread, cv2.imshow).
    - Resize, chuyển đổi ảnh (cv2.resize, cv2.cvtColor).
    - Vẽ bounding box (cv2.rectangle).

import numpy as np

- Thư viện mảng số học hiệu năng cao.
- Dùng để thao tác dữ liệu hình ảnh dưới dạng ma trận (array), xử lý vector.

import matplotlib.pyplot as plt

- Vẽ đồ thị và hiển thị ảnh.
- Dùng để:
    - Hiển thị ảnh sau khi xử lý.
    - Vẽ biểu đồ đánh giá mô hình (accuracy, confusion matrix...).

import xml.etree.ElementTree as ET

- Dùng để phân tích file XML chứa thông tin bounding boxes (dạng PASCAL VOC).
- Trích xuất tọa độ bounding box và nhãn tương ứng từ file annotation.

from skimage.transform import resize

- Thư viện skimage dùng resize ảnh (có thể dùng thay cv2.resize).
- Đảm bảo ảnh có cùng kích thước trước khi trích đặc trưng.

from skimage import feature

- Trích xuất đặc trưng HOG (Histogram of Oriented Gradients).
- Dùng feature.hog (image) để lấy vector đặc trưng từ ảnh.

from sklearn.svm import SVC

- SVM (Support Vector Machine) classifier.
- Là mô hình chính dùng để phân loại có/không có đối tượng trong ảnh.

from sklearn.preprocessing import LabelEncoder, StandardScaler

- LabelEncoder: Chuyển nhãn dạng text (VD: "cat", "dog") thành số (0, 1).
- StandardScaler: Chuẩn hóa dữ liệu đầu vào (mean = 0, std = 1) giúp SVM hoạt động hiệu quả hơn.

from sklearn.model_selection import train_test_split

- Chia dữ liệu thành tập huấn luyện và kiểm thử (train/test hoặc train/validation).

from sklearn.metrics import accuracy_score

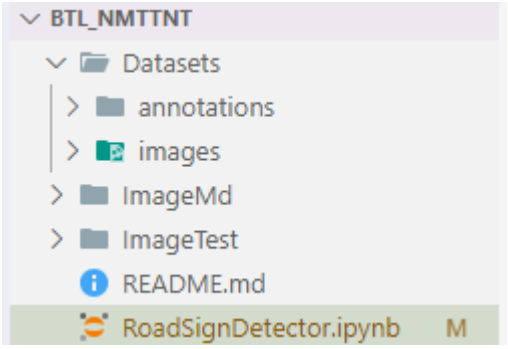
- Tính độ chính xác (accuracy) của mô hình SVM.

4.2 Dữ liệu và tiền xử lý

4.2.1 Chuẩn bị dữ liệu

Đây là một bước rất quan trọng, vì dữ liệu đầu vào chất lượng cao sẽ giúp mô hình hoạt động chính xác hơn.

Dữ liệu được thu thập từ https://aivietnam.edu.vn/, một kho dữ liệu phục vụ cho các bài toán liên quan đến thị giác máy tính. Cụ thể, bộ dữ liệu bao gồm một thư mục gốc có tên là **Datasets**, bên trong chứa hai thư mục con: **images** và **annotations**. Mỗi thư mục chứa khoảng 800 file.



Hình 21 – Cấu trúc thư mục Datasets

Thư mục images chứa các ảnh đầu vào định dạng .png, là dữ liệu đầu vào mà mô hình cần xử lý.

Hình 22 - Ảnh trong thư mục images

Thư mục annotations chứa các file .xml tương ứng với từng ảnh, theo định dạng chuẩn của **PASCAL VOC**. Mỗi file XML mô tả chi tiết thông tin của các đối tượng có trong ảnh, bao gồm:

- Tên ảnh và kích thước ảnh (chiều rộng, chiều cao, số kênh màu) được khai báo trong thẻ **<filename>** và **<size>.**
- Thông tin đối tượng (object) bao gồm:
    - Nhãn (label) của đối tượng, nằm trong thẻ **<name>**. Ví dụ: **"stop".**
    - Vị trí hộp giới hạn (bounding box) xác định vùng chứa đối tượng trong ảnh, gồm các tọa độ:
        - **<xmin>:** hoành độ góc trên bên trái
        - **<ymin>:** tung độ góc trên bên trái
        - **<xmax>:** hoành độ góc dưới bên phải
        - **<ymax>:** tung độ góc dưới bên phải

```
<annotation>
    <folder>images</folder>
    <filename>road54.png</filename>
    <size>
        <width>300</width>
        <height>400</height>
        <depth>3</depth>
    </size>
    <segmented>0</segmented>
    <object>
        <name>stop</name>
        <pose>Unspecified</pose>
        <truncated>0</truncated>
        <occluded>0</occluded>
        <difficult>0</difficult>
        <bndbox>
            <xmin>43</xmin>
            <ymin>60</ymin>
            <xmax>258</xmax>
            <ymax>273</ymax>
        </bndbox>
    </object>
</annotation>
```

Hình 23 – Ví dụ file .xml

Trong ví dụ trên, ảnh road54.png có kích thước 300x400 pixels và chứa một đối tượng có nhãn là "stop", được bao quanh bởi hộp giới hạn với tọa độ (43, 60, 258, 273).

Thông tin này sẽ được sử dụng để huấn luyện mô hình phát hiện đối tượng, giúp mô hình học cách xác định vị trí và phân loại các đối tượng trong ảnh đầu vào.

4.2.2 Tiền xử lý dữ liệu

Tiền xử lý dữ liệu là bước quan trọng nhằm chuẩn bị đầu vào phù hợp cho quá trình huấn luyện mô hình học máy. Trong đề tài này, dữ liệu được xử lý thông qua các bước chính sau.

4.2.2.1 Đọc và trích xuất đối tượng từ ảnh

Từ thư mục **annotations**, các file XML được duyệt qua để lấy thông tin vị trí và nhãn của các đối tượng. Dựa vào tọa độ **xmin, ymin, xmax, ymax**, các vùng chứa đối tượng được cắt trực tiếp từ ảnh gốc trong thư mục images.

Những đối tượng có nhãn là **trafficlight** được **loại bỏ** khỏi dữ liệu huấn luyện.

Mỗi đối tượng được cắt ra sẽ được lưu vào danh sách **list_image**, còn nhãn tương ứng sẽ được lưu vào **list_label**.

```
label_dir = 'Datasets/annotations'
image_dir = 'Datasets/images'

list_image = []
list_label = []

for xml_file in os.listdir(label_dir):
    xml_path = os.path.join(label_dir, xml_file)

    tree = ET.parse(xml_path)
    root = tree.getroot()

    folder = root.find('folder').text
    img_filename = root.find('filename').text
    img_filepath = os.path.join(image_dir, img_filename)
    img = cv2.imread(img_filepath)

    for obj in root.findall('object'):
        classname = obj.find('name').text
        if classname == 'trafficlight':
            continue

        xmin = int(obj.find('bndbox/xmin').text)
        ymin = int(obj.find('bndbox/ymin').text)
        xmax = int(obj.find('bndbox/xmax').text)
        ymax = int(obj.find('bndbox/ymax').text)

        object_img = img[ymin:ymax, xmin:xmax]

        list_image.append(object_img)
        list_label.append(classname)
```

Hình 24 – Đọc và trích xuất đối tượng từ ảnh

Ở trên, đoạn mã đã thực hiện:

- Đọc tất cả file XML trong thư mục **annotations**.
- Với mỗi file XML:
    - Lấy tên ảnh, load ảnh từ thư mục **images.**
    - Bỏ qua nếu ảnh không tồn tại.
    - Duyệt từng đối tượng (<object>) trong XML:
        - Bỏ qua nếu là **trafficlight**.
        - Lấy tọa độ **bounding box (xmin, ymin, xmax, ymax).**

- Cắt ảnh theo vùng **bounding box**.
- Lưu ảnh đã cắt vào **list_image**, lưu nhãn vào **list_label.**

```
plt.imshow(list_image[0])
plt.axis('off')
plt.title(list_label[0])
plt.show()

print(set(list_label))
```

✓ 0.2s


```
{'stop', 'crosswalk', 'speedlimit'}
```

Hình 25 – Ảnh đầu tiên và danh sách các label

Nhận được ảnh đầu tiên với label là **speedlimit** và danh sách các gồm 3 label: **speedlimit, crosswalk, stop.**

4.2.2.2 Trích xuất đặc trưng HOG

Tất cả các ảnh đối tượng được chuyển sang ảnh xám, sau đó được resize về kích thước 32x32. Tiếp theo, đặc trưng HOG (Histogram of Oriented Gradients) được tính toán để biểu diễn đặc trưng hình dạng và cạnh của đối tượng.

- Đặc trưng HOG giúp mô hình nhận diện được cấu trúc và hình khối đặc trưng của từng đối tượng.
- Kết quả HOG được lưu dưới dạng vector, dùng làm đầu vào cho mô hình phân loại.

```python
def preprocess_image(image):
    if(len(image) > 2):
        image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    image = image.astype(np.float32)

    resized_img = resize(
        image,
        output_shape=(32, 32),
        anti_aliasing=True
    )

    hog_features = feature.hog(
        resized_img,
        orientations=9,
        pixels_per_cell=(8, 8),
        cells_per_block=(2, 2),
        transform_sqrt=True,
        block_norm='L2',
        feature_vector=True
    )

    return hog_features
```

Hình 26 – Trích xuất đặc trưng HOG

Ở trên, đoạn mã đã thực hiện:

- Chuyển ảnh sang grayscale (nếu là ảnh màu tức là có $\boldsymbol{len(image) > 2)}$.
- Chuyển kiểu dữ liệu ảnh về **float32**.
- Resize ảnh về kích thước 32×32 sử dụng $\boldsymbol{anti\_aliasing = True}$ để làm mịn ảnh trước khi resize.
- Trích xuất đặc trưng HOG bằng cách sử dụng **feature.hog**. Trong đó các tham số được truyền vào gồm:
    - $\boldsymbol{orientations = 9}$**:**
        - Số hướng gradient được chia trong mỗi cell.
        - Ảnh sẽ được phân tích theo 9 hướng (0° đến 180°).
        - Mỗi hướng ghi lại độ mạnh của gradient tại cell đó.
    - $\boldsymbol{pixels\_per\_cell = (8, 8)}$**:**
        - Kích thước mỗi ô (cell) trong ảnh (tính bằng pixel).
        - Ảnh được chia thành các ô 8x8 pixel để tính histogram hướng.
    - $\boldsymbol{cells\_per\_block = (2, 2)}$**:**
        - Kích thước của mỗi khối (block) – tổ hợp nhiều cell.
        - Mỗi block gồm 2x2 cell, tức là (2 * 8)x(2 * 8) = 16x16 pixel.
        - HOG sẽ được chuẩn hóa theo từng block để tăng tính ổn định.
    - $\boldsymbol{transform\_sqrt = True}$**:**
        - Áp dụng căn bậc 2 cho độ tương phản ảnh trước khi tính HOG.
        - Giúp cải thiện độ ổn định ánh sáng (light invariance).

- $\boldsymbol{block\_norm = 'L2'}$:
  - Phương pháp chuẩn hóa vector đặc trưng trong mỗi block.
  - 'L2' nghĩa là dùng chuẩn Euclidean.
- $\boldsymbol{feature\_vector = True}$:
  - Trả về kết quả dưới dạng vector 1 chiều (flatten).
  - Nếu **False**, sẽ trả về dưới dạng ma trận theo khối.

- Trả về vector HOG.

```
img_processed_list = []

for img in list_image:
    img_features = preprocess_image(img)
    img_processed_list.append(img_features)

img_processed_list = np.array(img_processed_list)

print(list_image[0].shape)
print(img_processed_list[0].shape)
```

Hình 27 – Trích xuất đặc trưng HOG cho các ảnh đã cắt từ bounding box ở trên

Ở trên, đoạn mã đã thực hiện:

- Duyệt qua từng ảnh sau khi đã cắt từ **bounding box**.
- Trích xuất đặc trưng HOG.
- Thêm vector HOG vào **img_processed_list**.
- Chuyển danh sách các vector thành mảng numpy 2D, dạng:
  - (số ảnh, số đặc trưng mỗi ảnh)
- Sau đó in ra kích thước ảnh gốc sau cắt **list_image[0]** và kích thước vector đặc trưng HOG **img_processed_list[0].**

```
(321, 328, 3)
(324,)
```

Hình 28 – Kết quả nhận được khi trích xuất đặc trưng HOG

Kết quả nhận được là (321, 328, 3) kích thước ảnh đầu tiên của **list_image** là (h, w, 3) và (324,) là kích thước vector đặc trưng HOG với 324 đặc trưng.

4.2.2.3 Mã hóa nhãn

Các nhãn văn bản được chuyển đổi thành giá trị số bằng **LabelEncoder** để mô hình dễ dàng xử lý.

```
label_encoder = LabelEncoder()
encoded_label = label_encoder.fit_transform(list_label)

print(np.unique(encoded_label, return_counts=True))
print(encoded_label)
print(label_encoder.classes_)
```

Hình 29 – Mã hóa nhãn

Dùng hàm **fit_transform** của **LabelEncoder** để mã hóa các nhãn.

```
(array([0, 1, 2], dtype=int64), array([200, 783,  90], dtype=int64))
[1 1 1 ... 2 2 2]
['crosswalk' 'speedlimit' 'stop']
```

Hình 30 – Kết quả trả về khi in ra các nhãn sau khi mã hóa

Kết quả gồm 3 label tương ứng sau khi **fit_transform** là:

- 0: 200 mẫu **crosswalk**
- 1: 783 mẫu **speedlimit**
- 2: 90 mẫu **stop**

4.2.2.4 Tách tập huấn luyện và chuẩn hóa dữ liệu

Dữ liệu được chia thành hai tập:

- Tập huấn luyện (70%) dùng để học.
- Tập kiểm tra (30%) dùng để đánh giá độ chính xác mô hình.

Đặc trưng HOG được chuẩn hóa bằng StandardScaler, giúp mô hình học ổn định hơn và tránh bị chi phối bởi những đặc trưng có giá trị lớn.

```
X_train, X_test, y_train, y_test = train_test_split(img_processed_list, encoded_label, test_size=0.3, random_state=42,shuffle=True)

scaler_standard = StandardScaler()
X_train = scaler_standard.fit_transform(X_train)
X_test = scaler_standard.transform(X_test)
```

Hình 31 – Tách tập huấn luyện và chuẩn hóa dữ liệu

Ở trên, đoạn mã đã thực hiện:

- $\boldsymbol{test\_size = 0.3}$ **:** tức 30% là test và 70% để train.
- Chia dữ liệu ngẫu nhiên với $\boldsymbol{random\_state = 42}$ đảm bảo kết quả có thể tái lập (42 là một "con số vui" trong khoa học máy tính, xuất phát từ truyện *The Hitchhiker's Guide to the Galaxy* – nơi 42 được gọi là "câu trả lời cho sự sống, vũ trụ và mọi thứ".). Có thể dùng bất kỳ số nguyên nào, miễn là cố định.
- $\boldsymbol{shuffle = True}$: để trộn tập ảnh khi huấn luyện. Nếu không trộn dữ liệu sẽ được chia theo thứ tự ban đầu. Điều này có thể gây học lệch nếu dữ liệu đã được sắp xếp theo nhãn, thời gian, ...

- SVM rất nhạy với tỉ lệ đặc trưng nên cần đưa tất cả các đặc trưng về cùng thang đo. Nên sử dụng **StandardScaler** để chuẩn hóa theo công thức:

$$x' = \frac{x - \mu}{\sigma}$$

  - Trong đó:
    - μ: giá trị trung bình của đặc trưng.
    - σ: độ lệch chuẩn.
  - **fit_transform(X_train):** Tính trung bình và độ lệch chuẩn từ **X_train**, và chuẩn hóa luôn.
  - **transform(X_test):** Áp dụng cùng thông số từ **X_train** để chuẩn hóa **X_test.**

4.3 Mô hình và huấn luyện

4.3.1 Mô hình phân loại

Trong đề tài này, mô hình phân loại sử dụng là SVM (Support Vector Machine) - một thuật toán học máy có khả năng phân biệt tốt các lớp trong không gian đặc trưng HOG.

Cấu hình mô hình như sau:

model = SVC(kernel='rbf', random_state=42, probability=True, C=0.5)

Trong đó:

- $\boldsymbol{kernel = 'rbf'}$: sử dụng hàm kernel Gaussian (Radial Basis Function) - phù hợp với dữ liệu phi tuyến tính.
- $\boldsymbol{C = 0.5}$: tham số điều chỉnh độ phạt của sai số - giúp cân bằng giữa việc tối ưu biên và tránh overfitting.
- $\boldsymbol{probability = True}$: cho phép mô hình trả về xác suất dự đoán của từng lớp.
- $\boldsymbol{random\_state = 42}$: để đảm bảo tính tái lập.

4.3.2 Huấn luyện mô hình

Mô hình được huấn luyện trên tập dữ liệu đã tiền xử lý và chuẩn hóa:

model.fit(X_train, y_train)

Sau khi huấn luyện, mô hình đã học được cách phân biệt giữa các loại biển báo dựa trên đặc trưng hình dạng của chúng (HOG).

4.4 Kết quả và đánh giá mô hình

4.4.1 Đánh giá độ chính xác

Mô hình được đánh giá trên tập kiểm tra (**X_test, y_test**) bằng độ chính xác:

```
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Hình 32 – Đánh giá kết quả mô hình

Ở trên, đoạn mã đã thực hiện:

- $\boldsymbol{y\_pred = model.predict(X\_test)}$:
  - Dùng model để dự đoán nhãn cho tập kiểm tra **X_test.**
  - Kết quả là một mảng **y_pred** chứa các nhãn số như **0, 1, 2** tương ứng với:
    - **'crosswalk'**
    - **'speedlimit'**
    - **'stop'**
- $\boldsymbol{accuracy = accuracy\_score(y\_test, y\_pred)}$**:**
  - So sánh nhãn dự đoán (**y_pred**) với nhãn thật (**y_test**).
  - Tính độ chính xác **(accuracy)** theo công thức:

$$accuracy = \frac{Số\ mẫu\ dự\ đoán\ đúng}{Tổng\ số\ mẫu}$$

```
Accuracy: 0.9751552795031055
```

Hình 33 – Độ chính xác khi đánh giá

Sau khi đánh giá nhận thấy độ chính xác lên tới 97,5%.

4.4.2 Kiểm thử trên ảnh mới

Sau khi huấn luyện xong, mô hình được áp dụng vào phát hiện đối tượng trong ảnh mới theo pipeline sau:

- Tạo ảnh kim tự tháp (image pyramid) để dò ở nhiều tỷ lệ khác nhau.
- Trượt cửa sổ (sliding window) trên ảnh ở các kích thước khác nhau.
- Tiền xử lý ảnh cửa sổ, tính HOG, và đưa vào mô hình để dự đoán.
- Nếu xác suất dự đoán vượt qua ngưỡng threshold = 0.95 thì giữ lại.
- Áp dụng Non-Maximum Suppression (NMS) để loại bỏ các hộp bị chồng lắp.

```python
def pyramid_image(image, scale=0.8, min_size=(30, 30)):
    acc_scale = 1.0
    pyramid_imgs = [(image, acc_scale)]

    i = 0
    while True:
        acc_scale = acc_scale * scale
        h = int(image.shape[0] * acc_scale)
        w = int(image.shape[1] * acc_scale)
        if h < min_size[1] or w < min_size[0]:
            break
        image = cv2.resize(image, (w, h))
        pyramid_imgs.append((image, acc_scale))
        i+=1

    return pyramid_imgs
```

Hình 34 – Hàm pyramid_image

Ở trên, đoạn mã đã thực hiện thu nhỏ ảnh bằng cách nhân với scale cho đến khi kích thước nhỏ hơn **min_size** thì dừng lại:

- Lưu danh sách ảnh ở các tỉ lệ khác nhau trong **pyramid_imgs** với ảnh đầu tiên là ảnh gốc với scale là 1.
- Lặp liên tục để tạo các ảnh thu nhỏ dần.
- Giảm h với w theo từng lần scale là 0.8.
- Dùng **cv2.resize** để resize ảnh lại theo kích thước mới.
- Thêm ảnh đã resize và tỉ lệ mới vào **pyramid_imgs.**

```python
def sliding_window(image, window_sizes, stride):

    image_height, image_width = image.shape[:2]

    windows = []

    for window_size in window_sizes:
        window_width, window_height = window_size
        for y in range(0, image_height - window_height + 1, stride):
            for x in range(0, image_width - window_width + 1, stride):
                windows.append([x, y, x + window_width, y + window_height])

    return windows
```

Hình 35 – Hàm sliding_window

Ở trên, đoạn mã đã thực hiện:

- Lặp qua các kích thước cửa sổ.
- Duyệt ảnh từ góc trái trên đến góc phải dưới.
- Tạo các cửa sổ (bounding boxes) có kích thước đã cho và bước nhảy xác định.

```python
def compute_iou(bbox, bboxes, bbox_area, bboxes_area):
    xxmin = np.maximum(bbox[0], bboxes[:, 0])
    yymin = np.maximum(bbox[1], bboxes[:, 1])
    xxmax = np.minimum(bbox[2], bboxes[:, 2])
    yymax = np.minimum(bbox[3], bboxes[:, 3])

    w = np.maximum(0, xxmax-xxmin + 1)
    h = np.maximum(0, yymax-yymin + 1)

    intersection = w * h
    iou = intersection / (bbox_area + bboxes_area-intersection)

    return iou
```

Hình 36 – Hàm compute_iou

Ở trên, đoạn mã đã thực hiện tính IoU (Intersection over Union) giữa 1 bounding box với nhiều bounding boxes khác – đây là một chỉ số quan trọng trong object detection để xác định mức độ trùng lặp giữa các hộp:

$$IoU = \frac{Diện\ tích\ giao\ nhau}{Diện\ tích\ hợp\ nhất}$$

- Tìm tọa độ giao nhau của bbox với từng bbox trong danh sách:
  - **xxmin, yymin**: điểm bắt đầu của phần giao.
  - **xxmax, yymax**: điểm kết thúc của phần giao.
- Tính chiều rộng và chiều cao phần giao nhau. Nếu không giao nhau thì w hoặc h sẽ bằng 0.
- Tính IoU bằng công thức

$$IoU = \frac{Diện\ tích\ giao\ nhau}{Tổng\ diện\ tích - diện\ tích\ giao}$$

```python
def nms(bboxes, iou_threshold):
    if not bboxes:
        return []

    scores = np.array([bbox[5] for bbox in bboxes])
    indices = np.argsort(scores)[::-1]

    x1 =  np.array([bbox[0] for bbox in bboxes])
    y1 =  np.array([bbox[1] for bbox in bboxes])
    x2 =  np.array([bbox[2] for bbox in bboxes])
    y2 =  np.array([bbox[3] for bbox in bboxes])

    area = (x2 - x1 + 1) * (y2 - y1 + 1)

    keep = []

    while indices.size > 0:
        i = indices[0]
        keep.append(i)

        iou = compute_iou([x1[i],y1[i],x2[i],y2[i]],
                          np.array([
                              x1[indices[1:]],
                              y1[indices[1:]],
                              x2[indices[1:]],
                              y2[indices[1:]]

                              ]).T,area[i],area[indices[1:]]
                          )

        idx_keep = np.where(iou <= iou_threshold)[0]
        indices = indices[idx_keep + 1]
    return [bboxes[i] for i in keep]
```

Hình 37 – Hàm nms

Ở trên, đoạn mã đã thực hiện loại bỏ các bounding boxes trùng lặp (gần giống nhau) dựa trên độ chồng lấp IoU, giữ lại box có điểm số cao nhất:

- Lấy các **score** (độ tin cậy), sắp xếp từ cao xuống thấp, để ưu tiên **box** có độ tin cậy cao nhất.
- Trích riêng các tọa độ để tính diện tích và IoU dễ hơn.
- Tính diện tích từng box.
- Lấy box có điểm cao nhất (đầu tiên trong danh sách) và giữ lại.
- Tính IoU giữa box hiện tại và tất cả các box còn lại.
- Chỉ giữ lại các box có IoU nhỏ hơn threshold và cập nhật lại danh sách box còn xét.

Ví dụ:

Giả sử có 3 box cùng nằm gần nhau, cùng class "stop", điểm số:

- Box 1: 0.9
- Box 2: 0.85 (gần trùng Box 1)
- Box 3: 0.4 (xa)

→ NMS sẽ giữ lại Box 1 và 3, loại Box 2 nếu IoU(Box 1, Box 2) > threshold.

```python
def draw_rectangle(image, bounding_box, label_encoder, save_path):

    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    if not bounding_box:
        print("No bounding boxes to draw.")
        return
    for box in bounding_box:
        x1, y1, x2, y2, class_idx, prob = box

        cv2.rectangle(image, (x1, y1), (x2, y2), (0, 255, 0), 2)

        class_name = label_encoder.inverse_transform([class_idx])[0]
        label = f'{class_name}: {prob:.2f}'

        (w,h), _ = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, 0.6, 1)
        cv2.rectangle(image, (x1, y2), (x1 + w, y2 + h + 5), (0, 255, 0), -1)
        cv2.putText(image, label, (x1, y2 + h), cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 0), 1)

    image_bgr = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
    cv2.imwrite(save_path, image_bgr)

    plt.imshow(image)
    plt.axis('off')
    plt.show()
```

Hình 38 – Hàm draw_rectangle

Ở trên, đoạn mã đã thực hiện vẽ hình chữ nhật xung quanh biển báo:

- Chuyển màu từ BGR (OpenCV) về RGB (matplotlib), để hiển thị màu đúng.
- Lặp qua từng bounding box, lấy tọa độ, thông tin lớp và xác suất dự đoán:
    - Vẽ hình chữ nhật (box) màu xanh lá xung quanh vật thể.
    - Chuyển nhãn từ chỉ số (0, 1, 2) về tên (crosswalk, stop,...).
    - Tạo chuỗi label có cả tên lớp và xác suất (ví dụ: "stop: 0.87").
    - Tính kích thước label text để biết cần vẽ bao nhiêu pixel nền phía sau chữ.
    - Vẽ nền RGB xanh lá phía dưới box để hiển thị chữ rõ ràng hơn.
    - Ghi label (tên + xác suất) lên ảnh với font đen.
- Chuyển ảnh về BGR rồi lưu vào save_path.
- Dùng plt để show ảnh.

```
img_test_dir = 'ImageTest'
output_dir = 'OutputTest'
img_test_list = os.listdir(img_test_dir)
threadhold = 0.95
stride = 16
window_sizes = [
    (32, 32),
    (64, 64),
    (128, 128),
    (256, 256),
    (400, 400),
    (512, 512),
]

for img in img_test_list:
    start_time = time.time()
    img_file_path = os.path.join(img_test_dir, img)
    print(img_file_path)
    bounding_box = []
    image = cv2.imread(img_file_path)
    pyramid_images = pyramid_image(image)

    for pyramid_info in pyramid_images:
        pyramid_img, scale = pyramid_info

        window_list = sliding_window(pyramid_img, window_sizes, stride)
        for window in window_list:
            x1, y1, x2, y2 = window
            window_img = pyramid_img[y1:y2, x1:x2]

            preprocess_img = preprocess_image(window_img)

            normalize_img = scaler_standard.transform([preprocess_img])[0]

            label = model.predict_proba([normalize_img])[0]

            if np.all(label < threadhold):
                continue
            else:
                class_idx = np.argmax(label)
                prob = label[class_idx]

                x1 = int(x1 / scale)
                y1 = int(y1 / scale)
                x2 = int(x2 / scale)
                y2 = int(y2 / scale)

                bounding_box.append([x1, y1, x2, y2, class_idx, prob])
    bounding_box = nms(bounding_box, 0.01)
    draw_rectangle(image, bounding_box, label_encoder, os.path.join(output_dir, img))


    print('Time process: ', time.time() - start_time)
```

Hình 39 – Test thử với các ảnh trong thư mục ImageTest

Ở trên, đoạn mã đã thực hiện kiểm tra mô hình SVM với Sliding Window + Image Pyramid để phát hiện đối tượng trong ảnh test:

- Lấy danh sách ảnh test từ thư mục ImageTest.
- Khởi tạo biến **output_dir** là thư mục cho ra output của ảnh.
- **threshold** dùng để bỏ qua vùng không chắc chắn.
- $\boldsymbol{stride = 16}$: bước nhảy của sliding window.
- **window_sizes**: quét với nhiều kích thước khác nhau.
- Đọc từng ảnh trong thư mục ImageTest:

- Tạo nhiều ảnh từ lớn đến nhỏ để phát hiện cả vật nhỏ và lớn với hàm **pyramid_image**.
- Duyệt qua từng tầng của **pyramid_images:**
  - Lấy tất cả cửa sổ quét cho từng tầng của pyramid bằng hàm **sliding_window**.
  - Chuẩn hóa và trích HOG.
  - Dự đoán xác suất từng class bằng SVM
  - Nếu xác suất của mọi class đều nhỏ hơn ngưỡng thì bỏ qua. Ngược lại, lưu box + class + xác suất.
  - Scale lại tọa độ: Do ảnh đã bị thu nhỏ theo pyramid nên cần scale lại về kích thước gốc ảnh.
- Loại bỏ các box trùng nhau quá nhiều (IoU > 0.01).
- Hiển thị ảnh với các bounding box đã phát hiện + tên class + độ tin cậy.

Hình 40 – Kết quả test thử của ảnh crosswalk trong ImageTest

Hình 41 – Kết quả test thử của ảnh gồm 3 biển speedlimit trong ImageTest

Hình 42 – Kết quả test thử của ảnh gồm 2 biển speedlimit và crosswalk trong ImageTest

4.4.3 Đánh giá

4.4.3.1 Tổng quan dữ liệu đánh giá

Tập dữ liệu dùng để đánh giá mô hình gồm 50 ảnh, được lựa chọn nhằm phản ánh tính đa dạng và độ phức tạp của các tình huống trong thực tế. Các ảnh trong tập này bao gồm nhiều loại khác nhau như ảnh chứa biển báo rõ ràng, ảnh mà biển báo bị che khuất một phần hoặc bị cắt, ảnh có nhiều vật thể khác gây nhiễu xung quanh biển báo, và cả những ảnh có biển báo ở xa hoặc bị mờ do chất lượng hình ảnh hoặc điều kiện môi trường.

Việc đánh giá mô hình được thực hiện dựa trên ba tiêu chí chính:

- Nhận diện đúng: Mô hình xác định chính xác loại biển báo và khoanh vùng (bounding box) đúng vị trí trong ảnh.
- Không nhận diện được: Trong ảnh có biển báo nhưng mô hình không phát hiện ra.

- Nhận diện sai: Bao gồm các trường hợp mô hình phát hiện sai loại biển báo hoặc khoanh vùng sai vị trí (bounding box không khớp với thực tế).

Thông qua việc sử dụng tập dữ liệu này, quá trình đánh giá giúp phản ánh hiệu quả thực tế của mô hình trong các điều kiện khác nhau, từ dễ đến khó, và là cơ sở để xem xét các hạn chế còn tồn tại cũng như định hướng cải tiến trong các bước tiếp theo.

4.4.3.2 Kết quả nhận diện

Sau khi chạy mô hình trên tập đánh giá gồm 50 ảnh, kết quả thu được như sau:

Tiêu chí đánh giá	**Số lượng ảnh**	**Tỷ lệ (%)**
Ảnh được mô hình nhận diện đúng (biển báo và bounding box chính xác)	44 ảnh	88%
Ảnh không được mô hình phát hiện ra biển báo	5 ảnh	10%
Ảnh mô hình nhận diện sai (nhầm vật thể khác với biển báo, hoặc bounding box sai lệch vị trí)	1 ảnh	2%

Bảng 1. Kết quả nhận diện

Các kết quả trên được phân tích chi tiết theo các khái niệm trong lĩnh vực nhận diện đối tượng (object detection):

- True Positive (TP): Mô hình dự đoán đúng biển báo, bao gồm cả loại biển và vị trí (bounding box) gần đúng với thực tế. Những trường hợp này được tính là ảnh nhận diện đúng.
- False Positive (FP): Mô hình dự đoán có biển báo ở vị trí không tồn tại trong ảnh, hoặc nhận nhầm vật thể khác là biển báo. Những trường hợp này rơi vào nhóm ảnh nhận diện sai.
- False Negative (FN): Ảnh có chứa biển báo nhưng mô hình không phát hiện được. Những ảnh này thuộc nhóm không nhận diện được.
- True Negative (TN): Ảnh không chứa biển báo và mô hình cũng không phát hiện gì. Tuy nhiên, trong bài toán phát hiện đối tượng (object detection), các trường hợp TN thường ít được xét đến do mô hình chủ yếu tập trung vào các vùng có khả năng chứa đối tượng.

Từ kết quả này, có thể thấy mô hình đạt tỷ lệ nhận diện đúng tương đối cao, tuy nhiên vẫn tồn tại một số hạn chế như bỏ sót biển báo trong những ảnh phức tạp hoặc nhầm lẫn khi xuất hiện nhiều vật thể tương tự trong ảnh. Những điểm này sẽ được phân tích kỹ hơn ở các phần sau để làm rõ nguyên nhân và đưa ra hướng cải thiện.

4.4.3.3 Phân tích kết quả

Kết quả đánh giá cho thấy mô hình hoạt động khá ổn định trong điều kiện ảnh rõ ràng, tuy nhiên vẫn còn tồn tại một số vấn đề đáng chú ý. Phân tích theo từng nhóm ảnh như sau:

Ảnh được nhận diện đúng:

Các ảnh được mô hình nhận diện chính xác thường có chất lượng tốt, biển báo rõ nét, ít bị nhiễu và có kích thước đủ lớn trong ảnh. Biển báo thường xuất hiện gần trung tâm hoặc trong vùng dễ phát hiện, với độ tương phản tốt so với nền. Điều này cho thấy đặc trưng HOG có khả năng mô tả hiệu quả khi các yếu tố đầu vào thuận lợi.

Ảnh không nhận diện được:

Một số ảnh có chứa biển báo nhưng mô hình hoàn toàn không phát hiện được. Nguyên nhân có thể do biển báo bị mờ, bị che khuất một phần, nằm ở xa hoặc ảnh có ánh sáng yếu. Đặc biệt, những ảnh có biển báo nhỏ cũng gây khó khăn vì đặc trưng HOG khó mô tả được các chi tiết quan trọng trên vùng ảnh quá nhỏ. Ngoài ra, số lượng ảnh huấn luyện còn hạn chế, chẳng hạn như biển SpeedLimit chỉ có khoảng 200 objects, và biển Stop chỉ có 91 objects, cũng ảnh hưởng đáng kể đến khả năng tổng quát của mô hình trong việc nhận diện các loại biển báo này trong điều kiện phức tạp.

Ảnh nhận diện sai (bounding box sai hoặc nhầm lẫn vật thể):

Một số trường hợp mô hình phát hiện sai vị trí biển báo hoặc nhận nhầm các vật thể không liên quan như biển quảng cáo, logo, bảng thông tin,... là biển báo. Những sai lệch này có thể do:

- Sliding window chưa tối ưu: Khung dò tìm có thể không khớp hoàn toàn với kích thước và vị trí thực của biển báo.
- False Positive: Mô hình dự đoán nhầm vùng có đặc điểm giống biển báo.
- Thiếu bước tinh chỉnh bounding box: Sau khi phát hiện, nếu không có bước hiệu chỉnh lại vị trí hộp (fine-tuning), bounding box có thể bị lệch, sai kích thước hoặc bao không đúng vùng.

Những lỗi này phản ánh rằng mô hình còn bị giới hạn bởi chất lượng và độ đa dạng của tập dữ liệu huấn luyện, đặc biệt là số lượng ảnh huấn luyện còn ít cho một số lớp. Đồng thời, việc xử lý ảnh có biển báo nhỏ hoặc trong điều kiện phức tạp (che khuất, mờ, xa) vẫn chưa thực sự hiệu quả. Để cải thiện, cần tăng cường dữ liệu huấn luyện, bổ sung thêm ảnh ở nhiều điều kiện khác nhau, và cân nhắc áp dụng các bước xử lý nâng cao như tăng độ phân giải ảnh đầu vào, sử dụng sliding window đa tỉ lệ hoặc tích hợp thêm mạng tinh chỉnh bounding box (bounding box regressor).

4.4.3.4 Hạn chế và đề xuất cải tiến

Hạn chế:

Mô hình sử dụng kết hợp đặc trưng HOG và SVM tuy có hiệu quả nhất định trong điều kiện đơn giản, nhưng bộc lộ nhiều điểm yếu khi xử lý các tình huống thực tế phức tạp:

- HOG không phù hợp với ảnh có background phức tạp: Vì HOG chỉ tập trung mô tả cạnh và gradient cục bộ, nên khi ảnh có nhiều chi tiết nhiễu, vật thể phức tạp hoặc ánh sáng không đồng đều, đặc trưng trích xuất có thể bị sai lệch, khiến mô hình dễ nhầm lẫn.

- Không tối ưu trong việc xác định vị trí đối tượng: Phương pháp dò tìm bằng sliding window có hiệu suất thấp, vừa tốn thời gian vừa dễ bỏ sót đối tượng nhỏ hoặc bị lệch vị trí. Mô hình không tự động học được khái niệm không gian hoặc cấu trúc hình học của biển báo.
- Khó mở rộng: Khi tăng số lượng lớp biển báo cần phân loại, việc thiết kế hệ thống thủ công với HOG + SVM trở nên cồng kềnh và không hiệu quả.

Đề xuất cải tiến:

Để nâng cao hiệu quả nhận diện trong các tình huống phức tạp hơn, có thể áp dụng một số hướng cải tiến sau:

- Tiền xử lý ảnh tốt hơn: Áp dụng các kỹ thuật như cân bằng sáng (histogram equalization), lọc nhiễu, tăng cường độ tương phản, hoặc làm rõ ảnh mờ có thể giúp HOG mô tả tốt hơn các đặc trưng cốt lõi của biển báo.
- Dùng phương pháp học sâu hiện đại: Các mô hình như Convolutional Neural Networks (CNNs) có khả năng học đặc trưng từ dữ liệu mà không cần thủ công thiết kế, và cho kết quả vượt trội trong các bài toán nhận dạng hình ảnh.
- Thử nghiệm với các kiến trúc phát hiện đối tượng mạnh hơn: Ví dụ như YOLO (You Only Look Once) hoặc SSD (Single Shot Detector) — những mô hình này tích hợp cả phát hiện và phân loại trong một mạng duy nhất, cho tốc độ nhanh và độ chính xác cao, đặc biệt hiệu quả với ảnh có nhiều vật thể hoặc biển báo nhỏ.
- Tăng cường dữ liệu: Sử dụng các kỹ thuật data augmentation như xoay, thu phóng, thay đổi ánh sáng để tăng đa dạng dữ liệu huấn luyện, giúp mô hình học được nhiều tình huống thực tế hơn.

HOG + SVM là một giải pháp đơn giản, dễ triển khai nhưng có giới hạn rõ rệt trong môi trường thực tế nhiều biến đổi. Việc chuyển sang các giải pháp học sâu và áp dụng thêm các bước xử lý ảnh thông minh sẽ là hướng đi cần thiết để nâng cao độ chính xác, tính linh hoạt và hiệu suất của hệ thống nhận diện biển báo giao thông.

4.4.3.5 Hình ảnh minh họa

Nhận diện đúng:

Hình 43 – Ảnh nhận diện đúng

Không nhận diện được:

Hình 44 – Ảnh không nhận diện được

Nhận diện sai:

Hình 45 – Ảnh nhận diện sai

Link mã nguồn: https://github.com/sybui2004/RoadSignDetector

KẾT LUẬN

Trong khuôn khổ đề tài, nhóm đã xây dựng một hệ thống nhận diện biển báo giao thông sử dụng phương pháp trích xuất đặc trưng HOG kết hợp với mô hình phân loại SVM. Quy trình thực hiện gồm các bước: thu thập dữ liệu, tiền xử lý ảnh, trích xuất đặc trưng HOG, huấn luyện mô hình SVM, đánh giá mô hình.

Mô hình có khả năng nhận diện một số loại biển báo giao thông phổ biến với độ chính xác tương đối tốt trong điều kiện ảnh rõ ràng. Kết quả thử nghiệm cho thấy mô hình hoạt động khá ổn định trong các điều kiện ảnh rõ ràng, với các biển báo có kích thước phù hợp và độ tương phản tốt. Tuy nhiên, hệ thống vẫn còn một số hạn chế khi xử lý các trường hợp ảnh phức tạp như biển báo bị che khuất, mờ, nhỏ hoặc nằm trong bối cảnh có nhiều vật thể gây nhiễu. Ngoài ra, hiện tượng dự đoán sai vị trí hoặc nhầm lẫn vật thể không phải biển báo cũng được ghi nhận.

Định hướng phát triển:

- Mở rộng bộ dữ liệu huấn luyện đa dạng hơn về điều kiện thời tiết, ánh sáng, góc chụp.
- Sử dụng thêm các kỹ thuật tiền xử lý nâng cao như cân bằng sáng, tăng cường ảnh (augmentation).
- Kết hợp với các phương pháp học sâu (Deep Learning) như CNN, YOLO để tăng hiệu quả phát hiện và nhận diện.

TÀI LIỆU THAM KHẢO

[1] Support Vector Machine - https://machinelearningcoban.com/

[2] Bishop, Christopher M. “Pattern recognition and Machine Learning.”, Springer (2006)

[3] Duda, Richard O., Peter E. Hart, and David G. Stork. Pattern classification. John Wiley & Sons, 2012.

[4] SVC, https://scikit-learn.org/

[5] LIBSVM – A Library for Support Vector Machines

[6] Treml, M., et al.: Speeding up Semantic Segmentation for Autonomous Driving. Proceedings of the Conference on Neural Information Processing Systems (NIPS), Barcelona, Spain, 1–7 (2016).

[7] Takacs, Arpad & Rudas, Imre & Bosl, Dominik & Haidegger, Tamas. (2018). Highly Automated Vehicles and Self-Driving Cars [Industry Tutorial]. IEEE Robotics & Automation Magazine. 25. 106-112. 10.1109/MRA.2018.2874301.

[8] Yuan, Chang & Chen, Hui & Liu, Ju & Zhu, Di & Xu, Yanyan, Robust Lane Detection for Complicated Road Environment Based on Normal Map. IEEE Access. PP. 1-1. 10.1109/ACCESS.2018.2868976, 2018.

[9] Hà Thị Kim Duyên, Lê Mạnh Long, Nguyễn Đức Duy, Phan Sỹ Thuần, Nguyễn Ngọc Hải, Nguyễn Thị Tú Uyên, “NGHIÊN CỨU VÀ PHÁT TRIỂN HỆ THỐNG XE TỰ HÀNH ỨNG DỤNG TRÍ TUỆ NHÂN TẠO”, hội nghị HAUI, 6/2020.

[10] Nguyễn Văn Long,“Tìm hiểu và đề xuất phương pháp nhận dạng và phân loại biển báo giao thông ở Việt Nam”, Luận văn Thạc sĩ, Đại học Duy Tân, 2016.

[11] Hiep Do Quang, Tien Ngo Manh, Cuong Nguyen Manh, Dung Pham Tien, Manh Tran Van, Kiem Nguyen Tien, Duy Nguyen Duc, “An Approach to Design Navigation System for Omnidirectional Mobile Robot Based on ROS”, Robotics and Informatics (ICCRI 2020).

[12] Van Nguyen Thi Thanh, Tien Ngo Manh, Cuong Nguyen Manh, Dung Pham Tien, Manh Tran Van, Duyen Ha Thi Kim, Duy Nguyen Duc, “Autonomous Navigation for Omnidirectional Robot Based on Deep Reinforcement Learning”, Robotics and Informatics (ICCRI 2020).

[13] Maître, Henri. (2013). Image Processing: Overview and Perspectives. Intelligent Video Surveillance Systems.

[14] Botchkarev, Alexei. (2018). Performance Metrics (Error Measures) in Machine Learning Regression, Forecasting and Prognostics: Properties and Typology.

[15] Kingma, Diederik & Ba, Jimmy. (2014). Adam: A Method for Stochastic Optimization. International Conference on Learning Representations.