ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA CÔNG NGHỆ PHẦN MỀM

KHÓA LUẬN TỐT NGHIỆP

XÂY DỰNG ỨNG DỤNG NHẬN DẠNG BIỂN BÁO GIAO THÔNG TRÊN THIẾT BỊ DI ĐỘNG

Giáo viên hướng dẫn: **ThS. PHAN NGUYỆT MINH**
Sinh viên thực hiện: **NGUYỄN BÁ CHUNG - 07520031**
ĐỖ TRƯỜNG GIANG – 07520094
Lớp: **CNPM02**
Khóa: **02**

TP. Hồ Chí Minh, tháng 2 năm 2012

MỞ ĐẦU

Ngày nay với các tiến bộ của khoa học kỹ thuật thì mọi công việc hầu như đều có thể tiến hành trên máy tính một cách tự động hóa hoàn toàn hoặc một phần. Một trong những sự thay đổi lớn đó là cách thức chúng ta thu nhận và xử lý dữ liệu. Các công cụ nhập liệu như bàn phím hay máy scan dần bị thay thế bằng các thiết bị tiện lợi hơn như màn hình cảm ứng, camera…

Hơn thế nữa, các máy tính để bàn không còn là công cụ duy nhất có thể hỗ trợ cho con người. Chúng ta bước sang thế kỷ 21 với sự phát triển mạnh mẽ của các thiết bị di động, giải trí cầm tay hay smartphone. Với kích thước ngày càng nhỏ gọn và hiệu suất làm việc thì không ngừng được cải tiến, các công cụ mini này hứa hẹn sẽ là một phần không thể thiếu trong xã hội hiện đại. Và do đó, phát triển các ứng dụng trên các thiết bị này cũng là một xu thế tất yếu.

Công nghệ nhận dạng là một trong các công nghệ đang được áp dụng cho các thiết bị di động hiện nay. Nhận dạng có thể bao gồm nhận dạng âm thanh, hình ảnh. Các đối tượng nhận dạng có nhiều kiểu như tiếng nói, chữ viết, khuôn mặt, mã vạch … và biển báo giao thông cùng là một trong số đó. Chương trình nhận dạng biển báo giao thông thường phức tạp và được cài đặt trên những hệ thống có bộ xử lý lớn, camera chất lượng cao. Mục tiêu của khóa luận là cải tiến công nghệ nhận dạng này và mang nó cài đặt trên các thiết bị di động, giúp chúng ta phát hiện biển báo và nhận dạng nó một cách nhanh nhất.

Khóa luận **"Xây dựng ứng dụng nhận dạng biển báo giao thông trên thiết bị di động"** bao gồm tất cả 4 chương.

Chương I - Giới thiệu: Giới thiệu khái quát về khóa luận và mục đích của khóa luận.

Chương II - Nền tảng và công nghệ: Giới thiệu đầy đủ về các kiến thức nền tảng cũng như công nghệ và phần mềm được sử dụng trong khóa luận bao gồm kiến thức về xử lý ảnh, lý thuyết mạng noron, môi trường hệ điều hành dành cho di động Android, thư viện xử lý ảnh OpenCV.

Chương III - Xây dựng ứng dụng nhận dạng biển báo giao thông: Trình bày mô hình giải quyết bài toán nhận dạng trên thiết bị di động, các sơ đồ chức năng và thiết kế giao diện của chương trình.

Chương IV - Đánh giá kết quả và kết luận: Tổng kết quá trình thực hiện khóa luận và rút ra hướng phát triển sau này.

LỜI CẢM ƠN

Trong suốt thời gian thực hiện khóa luận tốt nghiệp, chúng em đã nhận được sự giúp đỡ, chỉ bảo tận tình của các thầy cô Trường ĐH CNTT – ĐHQGTPHCM. Chúngem xin gửi lời cảm ơn sâu sắc đến quý thầy cô. Đặc biệt xin chân thành cảm ơn cô **Phan Nguyệt Minh** – người đã trực tiếp hướng dẫn và tạo mọi điều kiện thuận lợi giúp đỡ chúng em hoàn thành khóa luận này.

Chúng em cũng xin cảm ơn chân thành tới gia đình và bạn bè, công ty đã tạo điều kiện, giúp đỡ và động viên chúng em hoàn thành khóa luận đúng thời hạn.

Mặc dù đã cố gắng hết khả năng nhưng khóa luận không thể nào tránh khỏi những thiếu xót. Rất mong nhận được sự góp ý quý báu của quý thầy cô để khóa luận có thể hoàn chỉnh hơn.

Nhóm sinh viên thực hiện
Nguyễn Bá Chung – Đỗ Trường Giang
Tháng 2 – 2012

NHẬN XÉT
(Của giảng viên hướng dẫn)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

NHẬN XÉT
(Của giảng viên phản biện)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

MỤC LỤC

DANH MỤC BẢNG BIỂU

DANH MỤC HÌNH VẼ

CHƯƠNG 1 : GIỚI THIỆU

Chương này trình bày các vấn đề sau:

1.1 Giới thiệu đề tài

1.2 Mục tiêu của đề tài

1.1 Giới Thiệu Đề Tài

Ngày nay, những tiến bộ mới trong khoa học kỹ thuật công nghệ đã giúp ích rất nhiều cho cuộc sống của con người. Mọi thứ hầu như đều được tự động và hiệu suất công việc được nâng cao hơn với sự trợ giúp của máy móc, thiết bị. Một trong những công nghệ tiên tiến đang được áp dụng rộng rãi trong đời sống chính là công nghệ nhận dạng.

Hình 1.1 – Một số thiết bị hay được sử dụng trong nhận dạng

Nhận dạng dữ liệu bao gồm có nhận dạng âm thanh và nhận dạng hình ảnh. Các đối tượng của bài toán nhận dạng thì rất phong phú, ví dụ như nhận dạng khuôn mặt, tiếng nói, nhận dạng chữ viết tay, nhận dạng mã vạch … Biển báo giao thông cũng là một trong số đó. Đây là kiểu đối tượng có tính chất hình học đặc trưng, thường bắt gặp trong đời sống hằng ngày với công dụng là đưa ra những cảnh báo thông tin cho người tham gia giao thông. Tuy nhiên các biển báo giao thông thì không

có quy luật mà chỉ là hệ thống các ký hiệu với ý nghĩa qui ước kèm theo. Việc ghi nhớ hình dạng và ý nghĩa của tất cả các loại biển báo đối với chúng ta sẽ là một khó khăn lớn, do đó chúng ta thường hay có nhu cầu tra cứu tìm hiểu trực quan.

Bài toán nhận dạng nói chung và nhận dạng biển báo giao thông nói riêng hiện vẫn còn là một trong những chủ đề được các nhà khoa học nghiên cứu. Hiện tại đã có một số hệ thống tiên tiến của nước ngoài có khả năng nhận dạng biển báo giao thông nhưng hầu hết các hệ thống này đều đòi hỏi một khả năng xử lý mạnh mẽ, đi kèm với nó là camera có chất lượng cao.

Quay trở lại vấn đề, ngày nay máy tính không còn là công cụ trợ giúp độc tôn dành cho con người. Hầu hết chúng ta ai cũng biết đến sự phát triển mạnh mẽ của các loại thiết bị giải trí cầm tay nhỏ gọn. Đó chính là smartphone. Với ưu điểm là kích thước bé, đi kèm với nó là các chíp xử lý thông minh tốc độ cao, smartphone có khả năng đảm đương rất nhiều tác vụ giống y như đang thao tác trên máy tính. Phát triển phần mềm cho smartphone hiện cũng là xu thế tất yếu.

Hình 1.2 – Smartphone đang là xu hướng phát triển mới trong giai đoạn này

Ứng dụng công nghệ nhận dạng trên smartphone chính là ý tưởng mà nhóm hướng tới khi thực hiện khóa luận này. Bài toán nhóm sẽ giải quyết là làm thế nào xây dựng một hệ thống thông minh cho phép phát hiện và nhận dạng biển báo giao thông trên thiết bị di động.

1.2 Mục Tiêu Của Đề Tài

Mục tiêu của đề tài là nghiên cứu bài toán nhận dạng nói chung và nhận dạng biển báo giao thông nói riêng, từ đó cải tiến áp dụng cho việc xây dựng hệ thống trên các thiết bị di động. Mặc dù smartphone có những cải tiến vượt trội nhưng tất nhiên những khác biệt về phần cứng như chip xử lý hay camera sẽ không thể so sánh với máy tính được. Do đó chương trình nhóm xây dựng sẽ tìm cách tối ưu hóa làm sao cho có thể tận dụng được những khả năng vốn có của smartphone.

Ngoài ra hệ thống biển báo giao thông của nước ta khá lớn, do đó nhóm sẽ xây dựng chương trình hoàn chỉnh nhưng sẽ thu nhỏ tập dữ liệu lại, coi đây như là một tập dữ liệu demo áp dụng cho khóa luận này.

Chương trình sẽ được xây dựng trên nền tảng Android, một trong những nền tảng di động phát triển mạnh nhất hiện nay. Ngoài ra nhóm sử dụng thư viện OpenCv hỗ trợ cho việc xử lý ảnh.

Hình 1.3 – Những nền tảng sẽ sử dụng trong khóa luận

CHƯƠNG 2 : NỀN TẢNG VÀ CÔNG NGHỆ

Chương này trình bày các vấn đề sau:

2.1 Lý thuyết xử lý ảnh

2.2 Lý thuyết mạng nơron

2.3 Nền tảng Android

2.4 Thư viện xử lý ảnh OpenCV

2.1 Lý Thuyết Xử Lý Ảnh

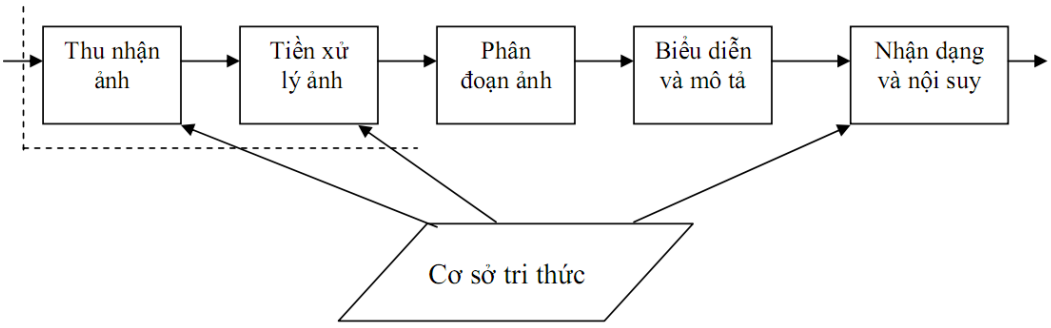
2.1.1 Tổng quan về xử lý ảnh

2.1.1.1 Xử lý ảnh là gì

Con người thu nhận thông tin qua các giác quan, trong đó thị giác đóng vai trò quan trọng nhất. Những năm trở lại đây với sự phát triển của phần cứng máy tính, xử lý ảnh và đồ hoạ đã phát triển một cách mạnh mẽ và có nhiều ứng dụng trong cuộc sống. Xử lý ảnh và đồ hoạ đóng một vai trò quan trọng trong tương tác người máy.

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

Có thể hiểu một cách khác, xử lý ảnh hay cao cấp hơn nữa là thị giác máy tính (Computer Vision) bao gồm tất cả các lý thuyết và kỹ thuật liên quan, cho phép tạo lập một hệ thống có khả năng tiếp nhận thông tin từ các hình ảnh thu được, lưu trữ và xử lý theo nhu cầu.



Hình 2.1 – Các bước cơ bản trong xử lý ảnh

- ***Thu nhận ảnh***: Quá trình tiếp nhận thông tin từ vật thể thông qua camera màu hoặc trắng đen, ảnh thu nhận được có thể là ảnh tương tự hoặc ảnh đã số hóa.
- ***Tiền xử lý ảnh***: Sau bộ thu nhận, ảnh có thể nhiễu, độ tương phản thấp nên cần đưa vào bộ tiền xử lý để nâng cao chất lượng. Chức năng chính của bộ tiền xử lý là lọc nhiễu, nâng độ tương phản để làm ảnh rõ hơn, nét hơn.
- ***Phân đoạn ảnh***: Là tách một ảnh đầu vào thành các vùng thành phần để biểu diễn phân tích, nhận dạng ảnh. Ví dụ: để nhận dạng chữ (hoặc mã vạch) trên phong bì thư cho mục đích phân loại bưu phẩm, cần chia các câu, chữ về địa chỉ hoặc tên người thành các từ, các chữ, các số (hoặc các vạch) riêng biệt để nhận dạng. Đây là phần phức tạp khó khăn nhất trong xử lý ảnh và cũng dễ gây lỗi, làm mất độ chính xác của ảnh. Kết quả nhận dạng ảnh phụ thuộc rất nhiều vào công đoạn này.
- ***Biểu diễn ảnh*:** Đầu ra ảnh sau phân đoạn chứa các điểm ảnh của vùng ảnh (ảnh đã phân đoạn) cộng với mã liên kết với các vùng lận cận. Việc biến đổi các số liệu này thành dạng thích hợp là cần thiết cho xử lý tiếp theo bằng máy tính. Việc chọn các tính chất để thể hiện ảnh gọi là trích chọn đặc trưng (Feature Selection) gắn với việc tách các đặc tính của ảnh dưới dạng các thông tin định lượng hoặc làm cơ sở để phân biệt lớp đối tượng này với đối tượng khác trong phạm vi ảnh nhận được. Ví dụ: trong nhận dạng ký tự trên phong bì thư, chúng ta miêu tả các đặc trưng của từng ký tự giúp phân biệt ký tự này với ký tự khác.
- ***Nhận dạng và nội suy ảnh*:** Nhận dạng ảnh là quá trình xác định ảnh. Quá trình này thường thu được bằng cách so sánh với mẫu chuẩn đã được học (hoặc lưu) từ trước. Nội suy là phán đoán theo ý nghĩa trên cơ sở nhận dạng. Ví dụ: một loạt chữ số và nét gạch ngang trên phong bì thư có thể được nội suy thành mã điện thoại.

2.1.1.2 Một số khái niệm trong xử lý ảnh

- ***Ảnh và điểm ảnh***: Gốc của ảnh (ảnh tự nhiên) là ảnh liên tục về không gian và độ sáng. Để xử lý bằng máy tính, ảnh cần phải được số hoá. Số hoá ảnh là sự biến đổi gần đúng một ảnh liên tục thành một tập điểm phù hợp với ảnh thật về vị trí (không gian) và độ sáng (mức xám). Khoảng cách giữa các điểm ảnh đó được thiết lập sao cho mắt người không phân biệt được ranh giới giữa chúng. Mỗi một điểm như vậy gọi là điểm ảnh và ảnh được xem như là 1 tập hợp các điểm ảnh.

- ***Độ phân giải của ảnh***: Độ phân giải (Resolution) của ảnh là mật độ điểm ảnh được ấn định trên một ảnh số được hiển thị.

- ***Mức xám của ảnh***: Mức xám của điểm ảnh là cường độ sáng của nó được gán bằng giá trị số tại điểm đó. Giá trị mức xám thông thường: 16, 32, 64, 128, 256.

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

2.1.2 Một số phương pháp biểu diễn ảnh

Sau khi thu nhận và số hóa, ảnh sẽ được lưu trữ hay chuyển sang giai đoạn phân tích. Trước khi đề cập đến vấn đề lưu trữ ảnh, cần xem xét ảnh sẽ được biểu diễn ra sao trong bộ nhớ máy tính.

2.1.2.1 Mã loạt dài

Mã loạt dài (Run-length Code) hay dùng để biểu diễn cho vùng ảnh hay ảnh nhị phân. Một vùng ảnh R có thể biểu diễn đơn giản nhờ một ma trận nhị phân:

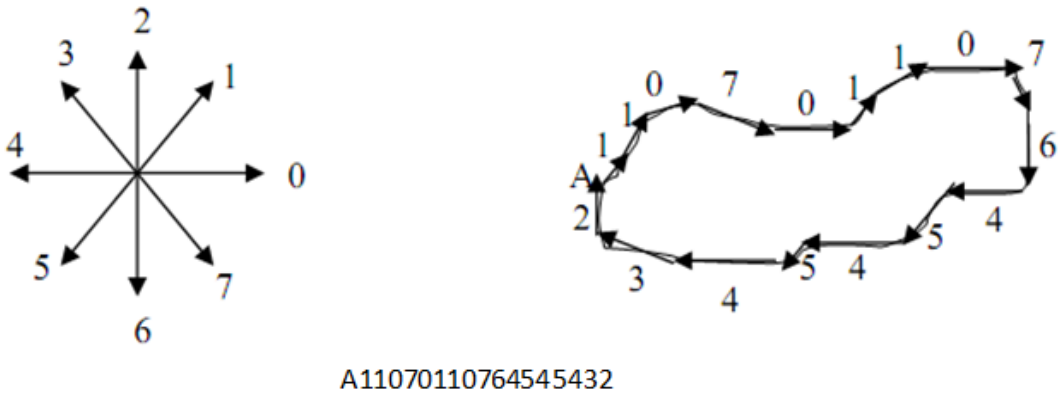
$$u(m,n) = \begin{cases} 1 & khi\ (m,n) \in R \\ 0 & khác \end{cases}$$

Với các biểu diễn trên, một vùng ảnh hay ảnh nhị phân được xem như chuỗi 0 hay 1 đan xen. Các chuỗi này được gọi là mạch (run). Theo phương pháp này, mỗi mạch sẽ

được biểu diễn bởi địa chỉ bắt đầu của mạch và chiều dài mạch theo dạng {<hàng,cột>, chiều dài}.

2.1.2.2 Mã xích

Mã xích (Chain Code) thường được dùng để biểu diễn biên của ảnh. Thay vì lưu trữ toàn bộ ảnh, người ta lưu trữ dãy các điểm ảnh như A, B…M. Theo phương pháp này, 8 hướng của vectơ nối 2 điểm biên liên tục được mã hóa. Khi đó ảnh được biểu diễn qua điểm ảnh bắt đầu A cùng với chuỗi các từ mã. Điều này được minh họa trong hình dưới đây:



Hình 2.2 – Hướng các điểm biên và mã tương ứng

2.1.2.3 Mã tứ phân

Theo phương pháp mã tứ phân (Quad Tree Code), một vùng ảnh coi như bao kín một hình chữ nhật. Vùng này được chia làm 4 vùng con (Quadrant). Nếu một vùng con gồm toàn điểm đen (1) hay toàn điểm trắng (0) thì không cần chia tiếp. Trong trường hợp ngược lại, vùng con gồm cả điểm đen và trắng gọi là vùng không đồng nhất, ta tiếp tục chia thành 4 vùng con tiếp và kiểm tra tính đồng nhất của các vùng con đó. Quá trình chia dừng lại khi mỗi vùng con chỉ chứa thuần nhất điểm đen hoặc điểm trắng. Quá trình đó tạo thành một cây chia theo bốn phần gọi là cây tứ phân. Như vậy, cây biểu diễn ảnh gồm một chuỗi các ký hiệu b (black), w (white) và g (grey) kèm theo ký hiệu mã hóa 4 vùng con. Biểu diễn theo phương pháp này ưu việt hơn so với các phương pháp trên, nhất là so với mã loạt dài. Tuy nhiên, để tính toán số đo các hình như chu vi, mô men là tương đối khó khăn.

2.1.3 Phương pháp phát hiện biên ảnh

- Điểm Biên: Một điểm ảnh được coi là điểm biên nếu có sự thay đổi nhanh hoặc đột ngột về mức xám (hoặc màu). Ví dụ trong ảnh nhị phân, điểm đen gọi là điểm biên nếu lân cận nó có ít nhất một điểm trắng.
- Đường biên (đường bao: boundary): tập hợp các điểm biên liên tiếp tạo thành một đường biên hay đường bao.
- Ý nghĩa của đường biên : đường biên là một loại đặc trưng cục bộ tiêu biểu trong phân tích, nhận dạng ảnh. Người ta sử dụng biên làm phân cách các vùng xám (màu) cách biệt.

2.1.3.1 Phát hiện biên trực tiếp

Phương pháp này làm nổi biên dựa vào sự biến thiên mức xám của ảnh. Kỹ thuật chủ yếu dùng để phát hiện biên ở đây là kỹ thuật lấy đạo hàm. Nếu lấy đạo hàm bậc nhất của ảnh ta có các kỹ thuật Gradient, nếu lấy đạo hàm bậc hai của ảnh ta có kỹ thuật Laplace. Ngoài ra còn có một số cách tiếp cận khác.

2.1.3.1.1 Kỹ thuật phát hiện biên Gradient

Gradient là một vec tơ f(x, y) có các thành phần biểu thị tốc độ thay đổi mức xám của điểm ảnh (theo hai hướng x, y trong bối cảnh xử lý ảnh hai chiều)

$$\frac{\partial f(x,y)}{\partial x} = f'_x \approx \frac{f(x+dx,y) - f(x,y)}{dx}$$

$$\frac{\partial f(x,y)}{\partial y} = f'_y \approx \frac{f(x,y+dy) - f(x,y)}{dy}$$

Trong đó, dx, dy là khoảng cách (tính bằng số điểm) theo hướng x và y. Tuy ta nói là lấy đạo hàm nhưng thực chất chỉ là mô phỏng và xấp xỉ đạo hàm bằng các kỹ thuật nhân chập vì ảnh số là tín hiệu rời rạc nên đạo hàm không tồn tại (thực tế chọn dx= dy=1).

Theo định nghĩa về Gradient, nếu áp dụng nó vào xử lý ảnh, việc tính toán sẽ rất phức tạp. Để đơn giản mà không mất tính chất của phương pháp Gradient, người ta sử dụng kỹ thuật Gradient dùng cặp mặt nạ H1, H2 trực giao (theo 2 hướng vuông góc).

➢ ***Mặt nạ Prewitt***

- Kỹ thuật sử dụng 2 mặt nạ nhân chập xấp xỉ đạo hàm theo 2 hướng x và y là:

$$H_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad H_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Hướng ngang (x) Hướng dọc (y)

- Tính $I \otimes H_x + I \otimes H_y$ để ra được kết quả

- Ví dụ:

$$I = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 5 & 5 & 5 & 0 & 0 \\ 5 & 5 & 5 & 5 & 0 & 0 \\ 5 & 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$I \otimes H_x + I \otimes H_y = \begin{pmatrix} 15 & 15 & 0 & -5 & * & * \\ 0 & 0 & -15 & -15 & * & * \\ -15 & -15 & -20 & -15 & * & * \\ -15 & -15 & -15 & -10 & * & * \\ * & * & * & * & * & * \\ * & * & * & * & * & * \end{pmatrix}$$

➢ ***Mặt nạ Sobel***

- Kỹ thuật sử dụng 2 mặt nạ nhân chập xấp xỉ đạo hàm theo 2 hướng x và y là:

$$H_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad H_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Hướng ngang (x) Hướng dọc (y)

- Tính $I \otimes H_x + I \otimes H_y$ để ra được kết quả.

➢ ***Kỹ thuật la bàn***

- Kỹ thuật sử dụng 8 mặt nạ nhân chập theo 8 hướng 0^0, 45^0, 90^0, 135^0, 180^0, 225^0, 270^0, 315^0.

$$H_1 = \begin{pmatrix} 5 & 5 & -3 \\ 5 & 0 & -3 \\ -3 & -3 & -3 \end{pmatrix} \qquad H_2 = \begin{pmatrix} 5 & 5 & 5 \\ -3 & 0 & -3 \\ -3 & -3 & -3 \end{pmatrix}$$

$$H_3 = \begin{pmatrix} -3 & 5 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & -3 \end{pmatrix} \qquad H_4 = \begin{pmatrix} -3 & -3 & 5 \\ -3 & 0 & 5 \\ -3 & -3 & 5 \end{pmatrix}$$

$$H_5 = \begin{pmatrix} -3 & -3 & -3 \\ -3 & 0 & 5 \\ -3 & 5 & 5 \end{pmatrix} \qquad H_6 = \begin{pmatrix} -3 & -3 & -3 \\ -3 & 0 & -3 \\ 5 & 5 & 5 \end{pmatrix}$$

$$H_7 = \begin{pmatrix} -3 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & 5 & -3 \end{pmatrix} \qquad H_8 = \begin{pmatrix} 5 & -3 & -3 \\ 5 & 0 & -3 \\ 5 & -3 & -3 \end{pmatrix}$$

- Kết quả thu được bằng cách tính $\sum_{i=1}^{8} I \otimes H_i$

2.1.3.1.2 Kỹ thuật phát hiện biên Laplace

Toán tử Laplace được định nghĩa như sau:

Ta có:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\frac{\partial^2 f}{\partial x^2} = \frac{\partial}{\partial x}\left(\frac{\partial f}{\partial x}\right) \approx \frac{\partial}{\partial x}\big(f(x+1,y) - f(x,y)\big)$$

$$\approx \big[f(x+1,y) - f(x,y)\big] - \big[f(x,y) - f(x-1,y)\big]$$

$$\approx f(x+1,y) - 2f(x,y) + f(x-1,y)$$

Tương tự

$$\frac{\partial^2 f}{\partial y^2} = \frac{\partial}{\partial y}\left(\frac{\partial f}{\partial y}\right) \approx \frac{\partial}{\partial y}\big(f(x,y+1) - f(x,y)\big)$$

$$\approx \big[f(x,y+1) - f(x,y)\big] - \big[f(x,y) - f(x,y-1)\big]$$

$$\approx f(x,y+1) - 2f(x,y) + f(x,y-1)$$

Vậy: ∇^2 f= f(x+1,y) + f(x,y+1) - 4f(x,y) + f(x-1,y) + f(x,y-1)

Dẫn tới

$$H = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Trong thực tế, người ta thường dùng nhiều kiểu mặt nạ khác nhau để xấp xỉ rời rạc đạo hàm bậc hai Laplace. Dưới đây là ba kiểu mặt nạ thường dùng:

$$H_1 = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix} \quad H_2 = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix} \quad H_3 = \begin{pmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{pmatrix}$$

2.1.3.1.3 Kỹ thuật phát hiện biên Canny

Đây là một thuật toán tương đối tốt, có khả năng đưa ra đường biên mảnh, và phát hiện chính xác điểm biên với điểm nhiễu.

Ta có thuật toán như sau:

- **Bước 1**: Làm trơn ảnh

  Tính I ⊗ H, với:

$$H = \frac{1}{115}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

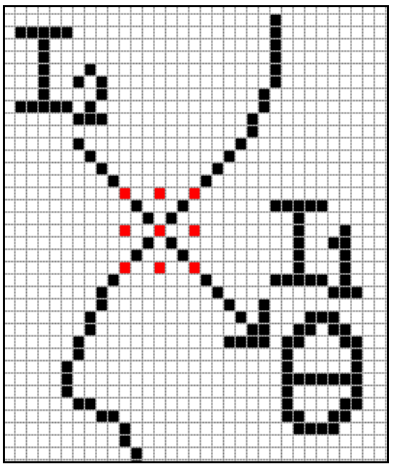
  Gọi G là kết quả lọc nhiễu: G= I⊗H

- **Bước 2**: Tính gradient của ảnh bằng mặt nạ Prewitt, kết quả đặt vào G_x,G_y.

  $G_x = G \otimes H_x$, $G_y = G \otimes H_y$

- **Bước 3**: Tính gradient hướng tại mỗi điểm (i,j) của ảnh. Hướng này sẽ được nguyên hóa để nằm trong 8 hướng [0..7], tương đương với 8 lân cận của một điểm ảnh.

- **Bước 4**: Dùng ràng buộc "loại bỏ những điểm không phải là cực đại" để xóa bỏ những điểm không là biên. Xét (i,j), θ là gradient hướng tại (i,j). I1, I2 là hai điểm lân cận của (i,j) theo hướng θ. Theo định nghĩa điểm biên cục bộ thì (i,j) là biên nếu I(i,j) cực đại địa phương theo hướng gradient → Nếu I(i,j) > I1 và I(i,j) > I2 thì mới giữ lại I(i,j), ngược lại xóa I(i,j) về điểm ảnh nền.



Hình 2.3 – Minh họa xác định điểm biên

- **Bước 5**: Phân ngưỡng. Với các điểm được giữ lại, thực hiện lấy ngưỡng gradient biên độ lần cuối để xác định các điểm biên thực sự.

2.1.3.2 Phát hiện biên gián tiếp

Nếu bằng cách nào đấy, chúng ta thu được các vùng ảnh khác nhau thì đường phân cách giữa các vùng đó chính là biên. Nói cách khác, việc xác định đường bao của ảnh được thực hiện từ ảnh đã được phân vùng. Phương pháp dò biên gián tiếp khó cài đặt nhưng áp dụng tốt khi sự biến thiên độ sáng nhỏ. Để có thể tiến hành xác định biên theo cách gián tiếp này, chúng ta cần giải quyết được bài toán phân vùng ảnh.

2.1.4 Phân vùng ảnh

Phân vùng ảnh là bước then chốt trong xử lý ảnh. Giai đoạn này nhằm phân tích ảnh thành những thành phần có cùng tính chất nào đó dựa theo biên hay các vùng liên thông. Tiêu chuẩn để xác định các vùng liên thông có thể là cùng mức xám, cùng màu hay cùng độ nhám…

Vùng ảnh là một chi tiết, một thực thể trông toàn cảnh. Nó là một tập hợp các điểm có cùng hoặc gần cùng một tính chất nào đó : mức xám, mức màu, độ nhám… Vùng ảnh là một trong hai thuộc tính của ảnh. Nói đến vùng ảnh là nói đến tính chất bề mặt. Đường bao quanh một vùng ảnh (Boundary) là biên ảnh. Các điểm trong một vùng ảnh có độ biến thiên giá trị mức xám tương đối đồng đều hay tính kết cấu tương đồng.

Dựa vào đặc tính vật lý của ảnh, người ta có nhiều kỹ thuật phân vùng : phân vùng dựa theo miền liên thông gọi là phân vùng dựa theo miền đồng nhất hay miền kề, phân vùng dựa vào biên gọi là phân vùng biên. Ngoài ra còn có các kỹ thuật phân vùng khác dựa vào biên độ, phân vùng dựa theo kết cấu.

2.1.4.1 Phân vùng theo ngưỡng biên độ

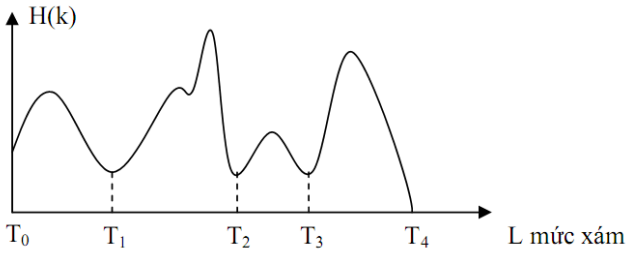
Đặc tính đơn giản nhất và có thể hữu ích nhất của ảnh đó là biên độ của các tính chất vật lý của ảnh như: độ tương phản, độ truyền sáng, màu sắc hoặc quang phổ.

Như vậy, có thể dùng ngưỡng biên độ để phân vùng khi biên độ đủ lớn đặc trưng cho ảnh. Thí dụ, biên độ trong bộ cảm biến ảnh hồng ngoại có thể phản ánh vùng có nhiệt độ thấp hay vùng có nhiệt độ cao. Kỹ thuật phân ngưỡng theo biên độ rất có lợi đối với ảnh nhị phân như văn bản in, đồ họa, ảnh màu hay ảnh X-quang.

Việc chọn ngưỡng rất quan trọng. Nó bao gồm các bước :

- Xem xét lược đồ xám của ảnh để xác định các đỉnh và các khe. Nếu ảnh có dạng rắn lượn (nhiều đỉnh và khe), các khe có thể dùng để chọn ngưỡng.
- Chọn ngưỡng t sao cho một phần xác định trước η của toàn bộ số mẫu là thấp hơn t.
- Điều chỉnh ngưỡng dựa trên lược đồ xám của các điểm lân cận.
- Chọn ngưỡng theo lược đồ xám của những điểm thỏa mãn tiêu chuẩn chọn. Thí dụ, với ảnh có độ tương phản thấp, lược đồ của những điểm có biên độ Laplace g(m,n) lớn hơn giá trị t định trước (sao cho từ 5% đến 10% số điểm ảnh với Gradient lớn nhất sẽ coi như biên) sẽ cho phép xác định các đặc tính ảnh lưỡng cực tốt hơn ảnh gốc.

Ta xét ví dụ sau về việc phân vùng dựa trên ngưỡng biên độ:



Hình 2.4 – Lược đồ rắn lượn và cách chọn ngưỡng

Giả sử ảnh có lược đồ xám và cách chọn các ngưỡng như hình trên với: T0=Lmin, …, T4=Lmax. Ta có 5 ngưỡng và phân ảnh thành 4 vùng, ký hiệu C_k là vùng thứ k của ảnh, k=1,2,3,4. Cách phân vùng theo nguyên tắc :

$$P(m,n) \in C_k \text{ nếu } Tk\text{-}1 \leq P(m,n) < T_k \text{ , } k=1,2,3,4.$$

Khi phân vùng xong, nếu ảnh rõ nét thì việc phân vùng coi như kết thúc. Nếu không, cần điều chỉnh ngưỡng.

2.1.4.2 Phân vùng theo miền đồng nhất

Kỹ thuật phân vùng ảnh thành các miền đồng nhất dựa vào các tính chất quan trọng nào đó của miền ảnh. Việc lựa chọn các tính chất của miền sẽ xác định tiêu chuẩn phân vùng. Tính đồng nhất của một miền ảnh là điểm chủ yếu xác định tính hiệu quả của việc phân vùng. Các tiêu chuẩn hay được dùng là sự thuần nhất về mức xám, màu sắc đối với ảnh màu, kết cấu sợi và chuyển động.

Các phương pháp phân vùng ảnh theo miền đồng nhất thường áp dụng là :

- Phương pháp tách cây tứ phân
- Phương pháp cục bộ
- Phương pháp tổng hợp

2.1.4.2.1 Phương pháp tách cây tứ phân

Về nguyên tắc, phương pháp này kiểm tra tính đúng đắn của tiêu chuẩn đề ra một cách tổng thể trên miền lớn của ảnh. Nếu tiêu chuẩn được thỏa mãn, việc phân đoạn coi như kết thúc. Trong trường hợp ngược lại, chia miền đang xét thành 4 miền nhỏ hơn. Với mỗi miền nhỏ, áp dụng một cách đệ quy phương pháp trên cho đến khi tất cả các miền đều thỏa mãn điều kiện.

Phương pháp này có thể mô tả bằng thuật toán sau :

```
Procedure PhanDoan(Mien)
    Begin
        If miền đang xét không thỏa Then
          Begin
            Chia miền đang xét thành 4 miền : Z1, Z2, Z3, Z4
            For i=1 to 4 do        PhanDoan (Zi)
          End
        Else  exit
    End
```

Tiêu chuẩn xét miền đồng nhất ở đây có thể dựa vào mức xám. Ngoài ra, có thể dựa vào độ lệch chuẩn hay độ chênh giữa giá trị mức xám lớn nhất và giá trị mức xám nhỏ nhất. Giả sử Max và Min là giá trị mức xám lớn nhất và nhỏ nhất trong miền đang xét.

Nếu |Max – Min| < T (ngưỡng) ta coi miền đang xét là đồng nhất. Trường hợp ngược lại, miền đang xét không là miền đồng nhất và sẽ được chia làm 4 phần.

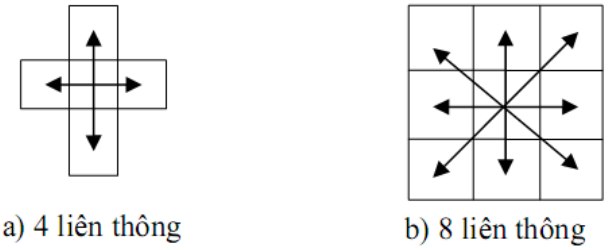
2.1.4.2.2 Phương pháp cục bộ

Ý tưởng của phương pháp là xét ảnh từ các miền nhỏ nhất rồi nối chúng lại nếu thỏa mãn tiêu chuẩn để được một miền đồng nhất lớn hơn. Tiếp tục với các miền thu được cho đến khi không thể nối thêm được nữa. Số miền còn lại cho ta kết quả phân đoạn. Như vậy, miền nhỏ nhất của bước xuất phát là điểm ảnh.

Phương pháp này hoàn toàn ngược với phương pháp tách. Song điều quan trọng ở đây là nguyên lý nối 2 vùng. Việc nối 2 vùng được thực hiện theo nguyên tắc sau :

- Hai vùng phải đáp ứng tiêu chuẩn, thí dụ như cùng màu hay cùng mức xám.
- Hai vùng phải kế cận nhau.

Trong xử lý ảnh, người ta dùng khái niệm liên thông để xác định tính chất kế cận. Có hai khái niệm về liên thông là 4 liên thông và 8 liên thông. Với 4 liên thông một điểm ảnh I(x,y) sẽ có 4 kế cận theo 2 hướng x và y ; trong khi đó với 8 liên thông, điểm I(x,y) sẽ có 4 liên thông theo 2 hướng x, y và 4 liên thông khác theo hướng chéo 45^o



a) 4 liên thông b) 8 liên thông

Hình 2.5 – Minh họa khái niệm liên thông

Dựa theo nguyên lý của phương pháp nối, ta có 2 thuật toán :

- Thuật toán tô màu (Blob Coloring) : sử dụng khái niệm 4 liên thông, dùng một cửa sổ di chuyển trên ảnh để so sánh với tiêu chuẩn nối.
- Thuật toán đệ quy cục bộ: sử dụng phương pháp tìm kiếm trong một cây để làm tăng kích thước vùng.

2.1.4.2.3 Phương pháp tổng hợp

Hai phương pháp nối (hợp) và tách đều có nhược điểm. Phương pháp tách sẽ tạo nên một cấu trúc phân cấp và thiết lập mối quan hệ giữa các vùng. Tuy nhiên, nó thực hiện việc chia quá chi tiết. Phương pháp hợp cho phép làm giảm số miền liên thông xuống tối thiểu, nhưng cấu trúc hàng ngang dàn trải, không cho ta thấy rõ mối liên hệ giữa các miền.

Vì nhược điểm này, người ta nghĩ đến phối hợp cả 2 phương pháp. Trước tiên, dùng phương pháp tách để tạo nên cây tứ phân, phân đoạn theo hướng từ gốc đến lá. Tiếp theo, tiến hành duyệt cây theo chiều ngược lại và hợp các vùng có cùng tiêu chuẩn. Với phương pháp này ta thu được một cấu trúc ảnh với các miền liên thông có kích thước tối đa.

2.1.4.3 Phân vùng theo kết cấu bề mặt

Kết cấu thường được nhận biết trên bề mặt của các đối tượng như gỗ, cát, vải vóc… Kết cấu là thuật ngữ phản ánh sự lặp lại của các phần tử sợi (texel) cơ bản. Sự lặp lại này có thể ngẫu nhiên hay có tính chu kì hoặc gần chu kì. Một texel chứa rất nhiều điểm ảnh. Trong phân tích ảnh, kết cấu được chia làm hai loại chính là: loại thống kê và loại cấu trúc.

Khi đối tượng xuất hiện trên một nền có tính kết cấu cao, việc phân đoạn dựa vào tính kết cấu trở nên quan trọng. Nguyên nhân là kết cấu sợi thường chứa mật độ cao các gờ (edge) làm cho phân đoạn theo biên kém hiệu quả, trừ phi ta loại tính kết cấu.

Nhìn chung, việc phân loại và phân vùng dựa vào kết cấu là một vấn đề phức tạp. Trong thực tế, chúng ta thường chỉ giải quyết vấn đề này bằng cách cho biết trước các loại kết cấu (dựa vào quy luật hay cách phân bố của nó).

2.1.5 Nhận dạng ảnh

Nhận dạng ảnh là giai đoạn cuối của các hệ thống xử lý ảnh. Nhận dạng là quá trình phân loại các đối tượng được biểu diễn theo một mô hình nào đó và gán chúng một tên (gán cho đối tượng một tên gọi, tức là một dạng) dựa theo những quy luật và mẫu chuẩn.

Trong lý thuyết về nhận dạng nói chung và nhận dạng ảnh nói riêng có ba cách tiếp cận khác nhau:

- Nhận dạng dựa vào phân hoạch không gian.
- Nhận dạng dựa vào cấu trúc.
- Nhận dạng dựa vào kỹ thuật mạng nơron.

Học có thầy: kỹ thuật phân loại nhờ kiến thức biết trước gọi là học có thầy. Đặc điểm cơ bản của kỹ thuật này là người ta có một thư viện các mẫu chuẩn. Mẫu cần nhận dạng sẽ được đem so sánh với mẫu chuẩn để xem nó thuộc loại nào. Vấn đề chủ yếu là thiết kế một hệ thống để có thể đối sánh đối tượng trong ảnh với mẫu chuẩn và quyết định gán cho chúng vào một lớp. Việc đối sánh nhờ vào các thủ tục ra quyết định dựa trên một công cụ gọi là hàm phân lớp hay hàm ra quyết định.

Học không có thầy: kỹ thuật này phải tự định ra các lớp khác nhau và xác định các tham số đặc trưng cho từng lớp. Học không có thầy đương nhiên là gặp khó khăn hơn. Một mặt, do số lớp không được biết trước, mặt khác những đặc trưng của lớp cũng không được biết trước. Kỹ thuật này nhằm tiến hành mọi cách gộp nhóm có thể và chọn lựa cách tốt nhất. Bắt đầu từ tập dữ liệu, nhiều thủ tục xử lý khác nhau nhằm phân lớp và nâng cấp dần để đạt được một phương án phân loại.

2.2 Lý Thuyết Mạng Nơron

2.2.1 Tổng quan về mạng nơron

2.2.1.1 Mạng nơron nhân tạo

Mạng noron nhân tạo (Artifical Neural Networks) mô phỏng lại mạng noron sinh học là một cấu trúc khối gồm các đơn vị tính toán đơn giản được liên kết chặt chẽ với nhau trong đó các liên kết giữa các noron quyết định chức năng của mạng.

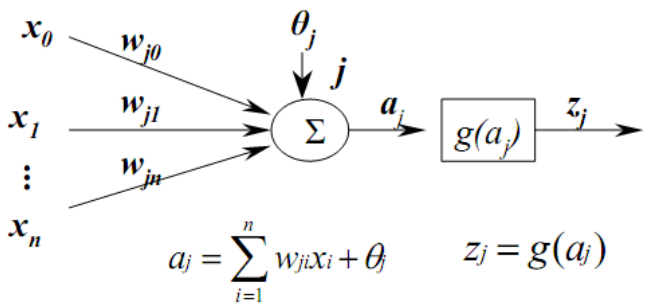
2.2.1.2 Các đặc trưng cơ bản của mạng nơron

- Gồm một tập các đơn vị xử lý (các noron nhân tạo)
- Trạng thái kích hoạt hay đầu ra của đơn vị xử lý
- Liên kết giữa các đơn vị. Xét tổng quát, mỗi liên kết được định nghĩa bởi một trọng số W_{jk} cho ta biết hiệu ứng mà tín hiệu của đơn vị j có trên đơn vị k
- Một luật lan truyền quyết định cách tính tín hiệu ra của từng đơn vị từ đầu vào của nó
- Một hàm kích hoạt, hay hàm chuyển (activation function, transfer function), xác định mức độ kích hoạt khác dựa trên mức độ kích hoạt hiện tại
- Một đơn vị điều chỉnh độ lệch (bias, offset) của mỗi đơn vị
- Phương pháp thu thập thông tin (luật học - learning rule)
- Môi trường hệ thống có thể hoạt động.

2.2.2 Các thành phần cơ bản của mạng nơron nhân tạo

2.2.2.1 Đơn vị xử lý

Còn được gọi là một nơron hay một nút (node), thực hiện một công việc rất đơn giản: nó nhận tín hiệu vào từ các đơn vị phía trước hay một nguồn bên ngoài và sử dụng chúng để tính tín hiệu ra sẽ được lan truyền sang các đơn vị khác.



Hình 2.6 – Đơn vị xử lý (Processing Unit)

Trong đó:

xi : các đầu vào

wji : các trọng số tương ứng với các đầu vào

θj : độ lệch (bias)

aj : đầu vào mạng (net-input)

zj : đầu ra của nơron

g(x): hàm chuyển (hàm kích hoạt).

Trong một mạng nơron có ba kiểu đơn vị:

- Các đơn vị đầu vào (Input units), nhận tín hiệu từ bên ngoài.
- Các đơn vị đầu ra (Output units), gửi dữ liệu ra bên ngoài.
- Các đơn vị ẩn (Hidden units), tín hiệu vào (input) và ra (output) của nó nằm trong mạng.

Mỗi đơn vị j có thể có một hoặc nhiều đầu vào: x0, x1, x2, … xn, nhưng chỉ có một đầu ra zj. Một đầu vào tới một đơn vị có thể là dữ liệu từ bên ngoài mạng, hoặc đầu ra của một đơn vị khác, hoặc là đầu ra của chính nó.

2.2.2.2 Hàm kết hợp

Mỗi một đơn vị trong một mạng kết hợp các giá trị đưa vào nó thông qua các liên kết với các đơn vị khác, sinh ra một giá trị gọi là net input. Hàm thực hiện nhiệm vụ này gọi là hàm kết hợp (combination function), được định nghĩa bởi một luật lan truyền cụ thể. Trong phần lớn các mạng nơron, chúng ta giả sử rằng mỗi một đơn vị cung cấp một bộ cộng như là đầu vào cho đơn vị mà nó có liên kết. Tổng đầu vào đơn vị j đơn giản chỉ là tổng trọng số của các đầu ra riêng lẻ từ các đơn vị kết nối cộng thêm ngưỡng hay độ lệch (bias) θj :

$$a_j = \sum_{i=1}^{n} w_{ji} x_i + \theta_j$$

Trường hợp wji > 0, nơron được coi là đang ở trong trạng thái kích thích. Tương tự, nếu như wji < 0, nơron ở trạng thái kiềm chế. Chúng ta gọi các đơn vị với luật lan truyền như trên là các sigma units.

Trong một vài trường hợp người ta cũng có thể sử dụng các luật lan truyền phức tạp hơn. Một trong số đó là luật sigma-pi, có dạng như sau:

$$a_j = \sum_{i=1}^{n} w_{ji} \prod_{k=1}^{m} x_{ik} + \theta_j$$

Rất nhiều hàm kết hợp sử dụng một "độ lệch" hay "ngưỡng" để tính net input tới đơn vị. Đối với một đơn vị đầu ra tuyến tính, thông thường, θj được chọn là hằng số và trong bài toán xấp xỉ đa thức θj = 1.

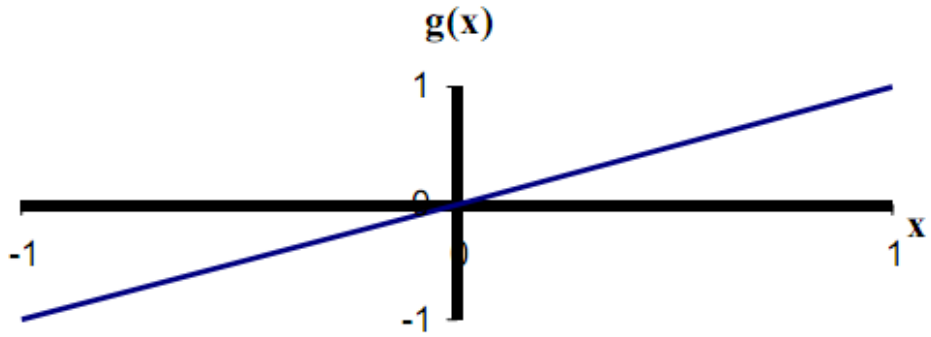
2.2.2.3 Hàm kích hoạt

Phần lớn các đơn vị trong mạng nơron chuyển net input bằng cách sử dụng một hàm vô hướng (scalar-to-scalar function) gọi là hàm kích hoạt, kết quả của hàm này là một giá trị gọi là mức độ kích hoạt của đơn vị (unit's activation). Loại trừ khả năng đơn vị đó thuộc lớp ra, giá trị kích hoạt được đưa vào một hay nhiều đơn vị khác. Các hàm kích hoạt thường bị ép vào một khoảng giá trị xác định, do đó thường được gọi là các hàm bẹp (squashing). Các hàm kích hoạt hay được sử dụng là:

- **Hàm đồng nhất (Linear function, Identity function)**

$$g(x) = x$$

Nếu coi các đầu vào là một đơn vị thì chúng sẽ sử dụng hàm này. Đôi khi một hằng số được nhân với net-input để tạo ra một hàm đồng nhất.



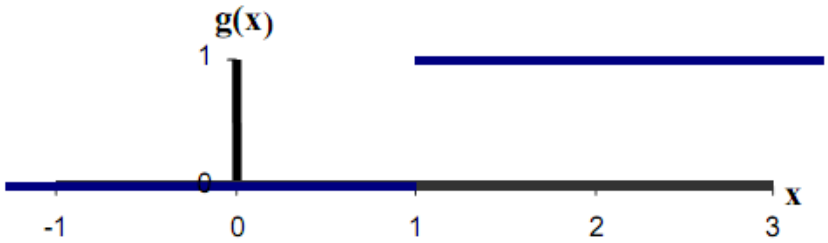
Hình 2.7 – Hàm đồng nhất (Identity function)

- **Hàm bước nhị phân (Binary step function, Hard limit function)**

  Hàm này cũng được biết đến với tên "Hàm ngưỡng" (Threshold function hay Heaviside function). Đầu ra của hàm này được giới hạn vào một trong hai giá trị:

$$g(x) = \begin{cases} 1, & nếu\ (x \geq \theta) \\ 0, & nếu\ (x < \theta) \end{cases}$$

  Dạng hàm này được sử dụng trong các mạng chỉ có một lớp. Trong hình vẽ sau, θ được chọn bằng 1.

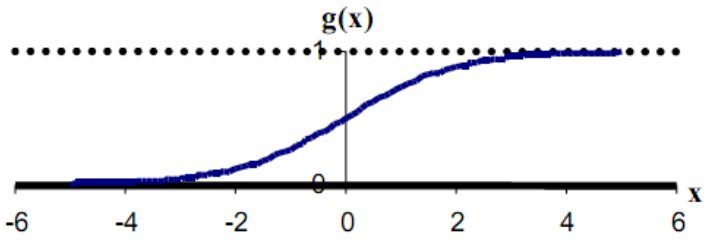


Hình 2.8 – Hàm bước nhị phân (Binary step function)

- **Hàm sigmoid (Sigmoid function (logsig))**

$$g(x) = \frac{1}{1+e^{-x}}$$

  Hàm này đặc biệt thuận lợi khi sử dụng cho các mạng được huấn luyện (trained) bởi thuật toán Lan truyền ngược (back-propagation), bởi vì nó dễ lấy đạo hàm, do đó có thể giảm đáng kể tính toán trong quá trình huấn luyện. Hàm này được ứng dụng cho các chương trình ứng dụng mà các đầu ra mong muốn rơi vào khoảng [0,1].

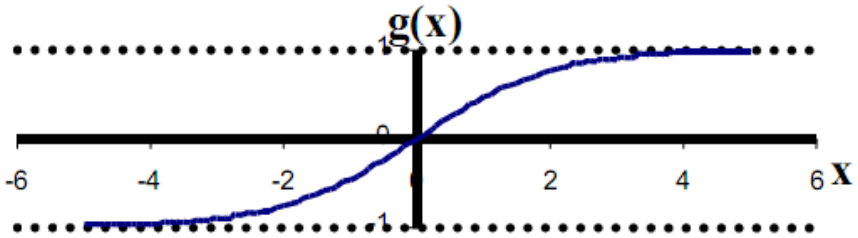


Hình 2.9 – Hàm Sigmoid

- **Hàm sigmoid lưỡng cực (Bipolar sigmoid function (tansig))**

$$g(x) = \frac{1 - e^{-x}}{1 + e^{-x}}$$

Hàm này có các thuộc tính tương tự hàm sigmoid. Nó làm việc tốt đối với các ứng dụng có đầu ra yêu cầu trong khoảng [-1,1].



Hình 2.10 – Hàm Sigmoid lưỡng cực

Các hàm chuyển của các đơn vị ẩn (hidden units) là cần thiết để biểu diễn sự phi tuyến vào trong mạng. Lý do là hợp thành của các hàm đồng nhất là một hàm đồng nhất. Mặc dù vậy nhưng nó mang tính chất phi tuyến (nghĩa là, khả năng biểu diễn các hàm phi tuyến) làm cho các mạng nhiều tầng có khả năng rất tốt trong biểu diễn các ánh xạ phi tuyến. Tuy nhiên, đối với luật học lan truyền ngược, hàm phải khả vi (differentiable) và sẽ có ích nếu như hàm được gắn trong một khoảng nào đó. Do vậy, hàm sigmoid là lựa chọn thông dụng nhất.

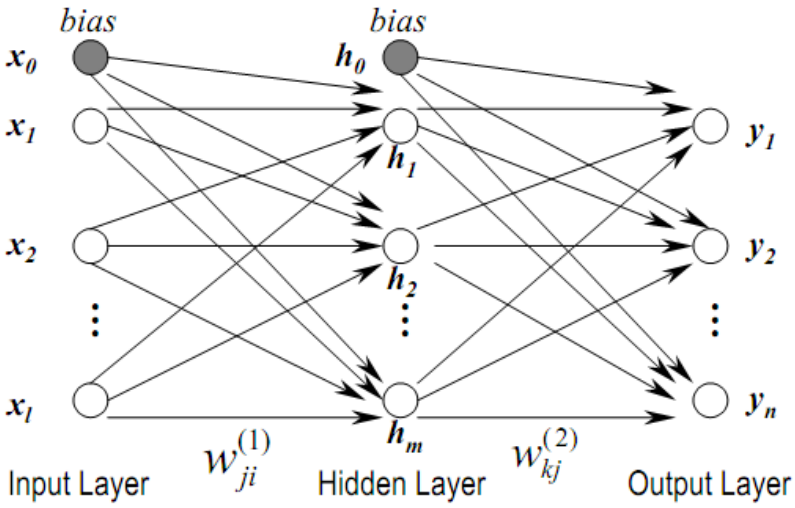
Đối với các đơn vị đầu ra (output units), các hàm chuyển cần được chọn sao cho phù hợp với sự phân phối của các giá trị đích mong muốn. Chúng ta đã thấy rằng đối với các giá trị ra trong khoảng [0,1], hàm sigmoid là có ích; đối với các giá trị đích mong muốn là liên tục trong khoảng đó thì hàm này cũng vẫn có ích, nó có thể cho ta các giá trị ra hay giá trị đích được căn trong một khoảng của hàm kích hoạt đầu ra. Nhưng nếu các giá trị đích không được biết trước khoảng xác định thì hàm hay được sử dụng nhất là hàm đồng nhất (identity function). Nếu giá trị mong muốn là dương nhưng không biết cận trên thì nên sử dụng một hàm kích hoạt dạng mũ (exponential output activation function).

2.2.2.4 Các hình trạng của mạng

Hình trạng của mạng được định nghĩa bởi: số lớp (layers), số đơn vị trên mỗi lớp, và sự liên kết giữa các lớp như thế nào. Các mạng về tổng thể được chia thành hai loại dựa trên cách thức liên kết các đơn vị:

2.2.2.4.1 Mạng truyền thẳng (Feed-forward neural network):

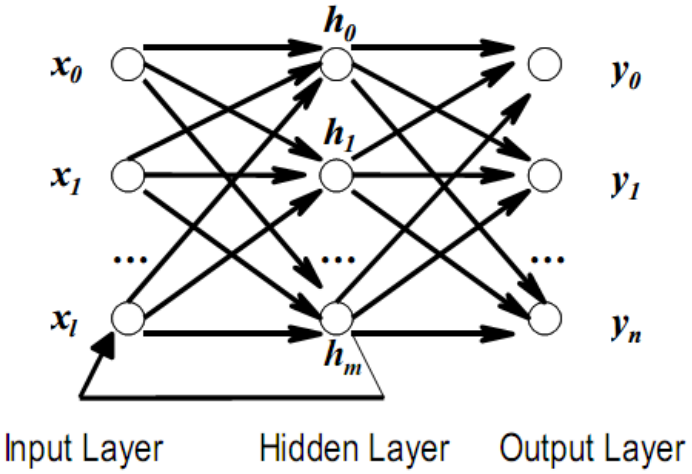
Dòng dữ liệu từ đơn vị đầu vào đến đơn vị đầu ra chỉ được truyền thẳng. Việc xử lý dữ liệu có thể mở rộng ra nhiều lớp, nhưng không có các liên kết phản hồi. Nghĩa là, các liên kết mở rộng từ các đơn vị đầu ra tới các đơn vị đầu vào trong cùng một lớp hay các lớp trước đó là không cho phép.



Hình 2.11 – Mạng nơron truyền thẳng nhiều lớp (Feed-forwardneural network)

2.2.2.4.2 Mạng hồi quy (Recurrent neural network):

Có chứa các liên kết ngược. Khác với mạng truyền thẳng, các thuộc tính động của mạng mới quan trọng. Trong một số trường hợp, các giá trị kích hoạt của các đơn vị trải qua quá trình nới lỏng (tăng giảm số đơn vị và thay đổi các liên kết) cho đến khi mạng đạt đến một trạng thái ổn định và các giá trị kích hoạt không thay đổi nữa. Trong các ứng dụng khác mà cách chạy động tạo thành đầu ra của mạng thì những sự thay đổi các giá trị kích hoạt là đáng quan tâm.



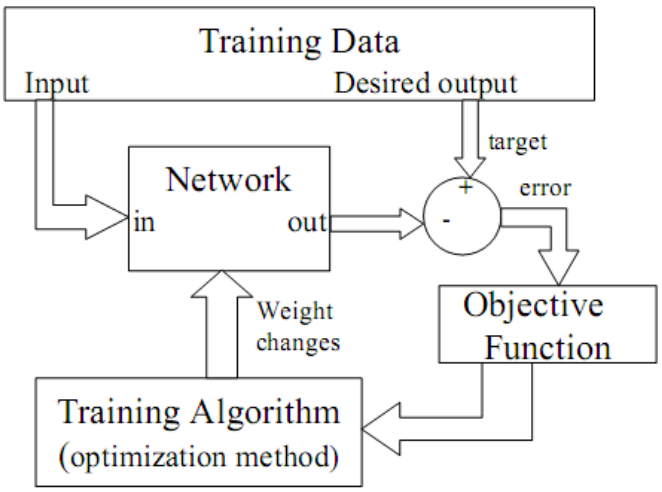
Hình 2.12 – Mạng nơron hồi quy (Recurrent neural network)

2.2.2.5 Huấn luyện mạng

Chức năng của một mạng nơron được quyết định bởi các nhân tố như: hình trạng mạng (số lớp, số đơn vị trên mỗi tầng, và cách mà các lớp được liên kết với nhau) và các trọng số của các liên kết bên trong mạng. Hình trạng của mạng thường là cố định, và các trọng số được quyết định bởi một thuật toán huấn luyện (training algorithm). Tiến trình điều chỉnh các trọng số để mạng "nhận biết" được quan hệ giữa đầu vào và đích mong muốn được gọi là học (learning) hay huấn luyện (training). Rất nhiều thuật toán học đã được phát minh để tìm ra tập trọng số tối ưu làm giải pháp cho các bài toán. Các thuật toán đó có thể chia làm hai nhóm chính: Học có thầy (Supervised learning) và Học không có thầy (Unsupervised Learning).

2.2.2.5.1 Học có thầy (Supervised learning):

Mạng được huấn luyện bằng cách cung cấp cho nó các cặp mẫu đầu vào và các đầu ra mong muốn (target values). Các cặp được cung cấp bởi "thầy giáo", hay bởi hệ thống trên đó mạng hoạt động. Sự khác biệt giữa các đầu ra thực tế so với các đầu ra mong muốn được thuật toán sử dụng để thích ứng các trọng số trong mạng. Điều này thường được đưa ra như một bài toán xấp xỉ hàm số - cho dữ liệu huấn luyện bao gồm các cặp mẫu đầu vào x, và một đích tương ứng t, mục đích là tìm ra hàm f(x) thoả mãn tất cả các mẫu học đầu vào.



Hình 2.13 – Mô hình Học có thầy (Supervised learning model)

2.2.2.5.2 Học không có thầy (Unsupervised Learning):

Với cách học không có thầy, không có phản hồi từ môi trường để chỉ ra rằng đầu ra của mạng là đúng. Mạng sẽ phải khám phá các đặc trưng, các điều chỉnh, các mối tương quan, hay các lớp trong dữ liệu vào một cách tự động. Trong thực tế, đối với phần lớn các biến thể của học không có thầy, các đích trùng với đầu vào. Nói một cách khác, học không có thầy luôn thực hiện một công việc tương tự như một mạng tự liên hợp, cô đọng thông tin từ dữ liệu vào.

2.2.2.6 Hàm mục tiêu

Để huấn luyện một mạng và xét xem nó thực hiện tốt đến đâu, ta cần xây dựng một hàm mục tiêu (hay hàm giá) để cung cấp cách thức đánh giá khả năng hệ thống một cách không nhập nhằng. Việc chọn hàm mục tiêu là rất quan trọng bởi vì hàm này thể hiện các mục tiêu thiết kế và quyết định thuật toán huấn luyện nào có thể được áp dụng. Để phát triển một hàm mục tiêu đo được chính xác cái chúng ta muốn không phải là việc dễ dàng. Một vài hàm cơ bản được sử dụng rất rộng rãi. Một trong số chúng là hàm tổng bình phương lỗi (sum of squares error function)

$$E = \frac{1}{NP}\sum_{p=1}^{P}\sum_{i=1}^{N}(t_{pi} - y_{pi})^2$$

Trong đó

p: số thứ tự mẫu trong tập huấn luyện

i : số thứ tự của đơn vị đầu ra

t_{pi} và y_{pi} : tương ứng là đầu ra mong muốn và đầu ra thực tế của mạng cho đơn vị đầu ra thứ i trên mẫu thứ p.

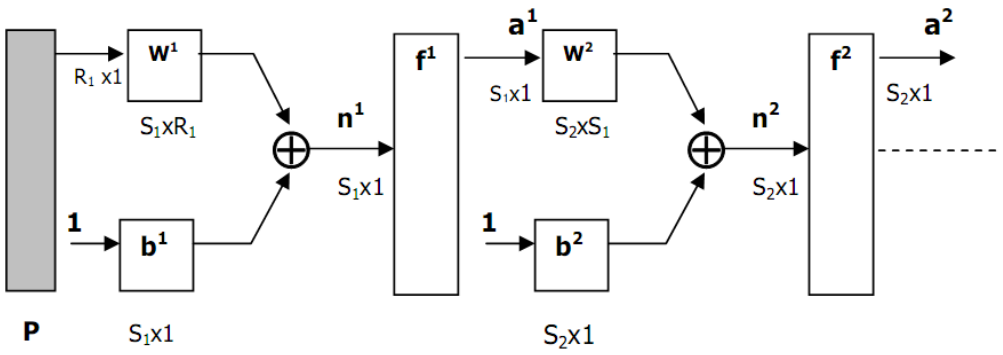
Trong các ứng dụng thực tế, nếu cần thiết có thể làm phức tạp hàm số với một vài yếu tố khác để có thể kiểm soát được sự phức tạp của mô hình.

2.2.3 Mạng truyền thẳng và thuật toán lan truyền ngược

Để đơn giản và tránh hiểu nhầm, mạng truyền thẳng trình bày trong phần này là mạng truyền thẳng có nhiều lớp (MLP - MultiLayer Perceptron). Đây là một trong những mạng truyền thẳng điển hình, thường được sử dụng trong các hệ thống nhận dạng.

2.2.3.1 Mạng truyền thẳng MLP

Một mạng truyền thẳng nhiều lớp bao gồm một lớp vào, một lớp ra và một hoặc nhiều lớp ẩn. Các nơron đầu vào thực chất không phải các nơron theo đúng nghĩa, bởi lẽ chúng không thực hiện bất kỳ một tính toán nào trên dữ liệu vào, đơn giản nó chỉ tiếp nhận các dữ liệu vào và chuyển cho các lớp kế tiếp. Các nơron ở lớp ẩn và lớp ra mới thực sự thực hiện các tính toán, kết quả được định dạng bởi hàm đầu ra (hàm chuyển). Cụm từ "truyền thẳng" (feed forward) (không phải là trái nghĩa của lan truyền ngược) liên quan đến một thực tế là tất cả các nơron chỉ có thể được kết nối với nhau theo một hướng: tới một hay nhiều các nơron khác trong lớp kế tiếp (loại trừ các nơron ở lớp ra).



Hình 2.14 – Mạng nơron truyền thẳng nhiều lớp

Trong đó

P: Vector đầu vào (vector cột)

$\mathbf{W^i}$: Ma trận trọng số của các nơron lớp thứ i.

($\mathbf{S^i x R^i}$: S hàng (nơron) - R cột (số đầu vào))

$\mathbf{b^i}$: Vector độ lệch (bias) của lớp thứ i (S^ix1: cho S nơron)

$\mathbf{n^i}$: net input (S^ix1)

$\mathbf{f^i}$: Hàm chuyển (hàm kích hoạt)

$\mathbf{a^i}$: net output (S^ix1)

$\oplus$: Hàm tổng thông thường.

Mỗi liên kết gắn với một trọng số, trọng số này được thêm vào trong quá trình tín hiệu đi qua liên kết đó. Các trọng số có thể dương, thể hiện trạng thái kích thích, hay âm, thể hiện trạng thái kiềm chế. Mỗi nơron tính toán mức kích hoạt của chúng bằng cách cộng tổng các đầu vào và đưa ra hàm chuyển. Một khi đầu ra của tất cả các nơron trong một lớp mạng cụ thể đã thực hiện xong tính toán thì lớp kế tiếp có thể bắt đầu thực hiện tính toán của mình bởi vì đầu ra của lớp hiện tại tạo ra đầu vào của lớp kế tiếp. Khi tất cả các nơron đã thực hiện tính toán thì kết quả được trả lại bởi các nơron đầu ra. Tuy nhiên, có thể là chưa đúng yêu cầu, khi đó một thuật toán huấn luyện cần được áp dụng để điều chỉnh các tham số của mạng.

Xét trường hợp mạng có hai lớp như hình 2.14, công thức tính toán cho đầu ra như sau:

$$\mathbf{a^2 = f^2(W^2(f^1(W^1P + b^1)) + b^2)}$$

Mạng có nhiều lớp có khả năng tốt hơn là các mạng chỉ có một lớp, chẳng hạn như mạng hai lớp với lớp thứ nhất sử dụng hàm sigmoid và lớp thứ hai dùng hàm đồng nhất có thể áp dụng để xấp xỉ các hàm toán học khá tốt, trong khi các mạng chỉ có một lớp thì không có khả năng này.

2.2.3.2 Thiết kế cấu trúc mạng

Mặc dù, về mặt lý thuyết, có tồn tại một mạng có thể mô phỏng một bài toán với độ chính xác bất kỳ. Tuy nhiên, để có thể tìm ra mạng này không phải là điều đơn giản. Để định nghĩa chính xác một kiến trúc mạng như: cần sử dụng bao nhiêu lớp ẩn,

mỗi lớp ẩn cần có bao nhiêu đơn vị xử lý cho một bài toán cụ thể là một công việc hết sức khó khăn.

2.2.3.2.1 Số lớp ẩn:

Vì các mạng có hai lớp ẩn có thể thể hiện các hàm với dáng điệu bất kỳ, nên, về lý thuyết, không có lý do nào sử dụng các mạng có nhiều hơn hai lớp ẩn. Người ta đã xác định rằng đối với phần lớn các bài toán cụ thể, chỉ cần sử dụng một lớp ẩn cho mạng là đủ. Các bài toán sử dụng hai lớp ẩn hiếm khi xảy ra trong thực tế. Thậm chí đối với các bài toán cần sử dụng nhiều hơn một lớp ẩn thì trong phần lớn các trường hợp trong thực tế, sử dụng chỉ một lớp ẩn cho ta hiệu năng tốt hơn là sử dụng nhiều hơn một lớp. Việc huấn luyện mạng thường rất chậm khi mà số lớp ẩn sử dụng càng nhiều.

2.2.3.2.2 Số đơn vị trong lớp ẩn:

Một vấn đề quan trọng trong việc thiết kế một mạng là cần có bao nhiêu đơn vị trong mỗi lớp. Sử dụng quá ít đơn vị có thể dẫn đến việc không thể nhận dạng được các tín hiệu đầy đủ trong một tập dữ liệu phức tạp, hay thiếu ăn khớp (underfitting). Sử dụng quá nhiều đơn vị sẽ tăng thời gian luyện mạng, có lẽ là quá nhiều để luyện khi mà không thể luyện mạng trong một khoảng thời gian hợp lý. Số lượng lớn các đơn vị có thể dẫn đến tình trạng thừa ăn khớp (overfitting), trong trường hợp này mạng có quá nhiều thông tin, hoặc lượng thông tin trong tập dữ liệu mẫu (training set) không đủ các dữ liệu đặc trưng để huấn luyện mạng.

Số lượng tốt nhất của các đơn vị ẩn phụ thuộc vào rất nhiều yếu tố - số đầu vào, đầu ra của mạng, số trường hợp trong tập mẫu, độ nhiễu của dữ liệu đích, độ phức tạp của hàm lỗi, kiến trúc mạng và thuật toán luyện mạng.

Trong phần lớn các trường hợp, không có một cách để có thể dễ dàng xác định được số tối ưu các đơn vị trong lớp ẩn mà không phải luyện mạng sử dụng số các đơn vị trong lớp ẩn khác nhau và dự báo lỗi tổng quát hóa của từng lựa chọn. Cách tốt nhất là sử dụng phương pháp thử sai (trial-and-error). Trong thực tế, có thể sử dụng phương pháp lựa chọn tiến (forward

selection) hay lựa chọn lùi (backward selection) để xác định số đơn vị trong lớp ẩn.

Lựa chọn tiến bắt đầu với việc chọn một luật hợp lý cho việc đánh giá hiệu năng của mạng. Sau đó, ta chọn một số nhỏ các đơn vị ẩn, luyện và thử mạng; ghi lại hiệu năng của mạng. Sau đó, tăng một chút số đơn vị ẩn; luyện và thử lại cho đến khi lỗi là chấp nhận được, hoặc không có tiến triển đáng kể so với trước.

Lựa chọn lùi, ngược với lựa chọn tiến, bắt đầu với một số lớn các đơn vị trong lớp ẩn, sau đó giảm dần đi. Quá trình này rất tốn thời gian nhưng sẽ giúp ta tìm được số lượng đơn vị phù hợp cho lớp ẩn.

2.2.3.3 Thuật toán lan truyền ngược (Back-Propagation)

Cần có một sự phân biệt giữa kiến trúc của một mạng và thuật toán học của nó, các mô tả trong các mục trên mục đích là nhằm làm rõ các yếu tố về kiến trúc của mạng và cách mà mạng tính toán các đầu ra từ tập các đầu vào. Sau đây là mô tả của thuật toán học sử dụng để điều chỉnh hiệu năng của mạng sao cho mạng có khả năng sinh ra được các kết quả mong muốn.

Về cơ bản có hai dạng thuật toán để luyện mạng: học có thầy và học không có thầy. Các mạng nơron truyền thẳng nhiều lớp được luyện bằng phương pháp học có thầy. Phương pháp này căn bản dựa trên việc yêu cầu mạng thực hiện chức năng của nó và sau đó trả lại kết quả, kết hợp kết quả này với các đầu ra mong muốn để điều chỉnh các tham số của mạng, nghĩa là mạng sẽ học thông qua những sai sót của nó.

Thuật toán lan truyền ngược là dạng tổng quát của thuật toán trung bình bình phương tối thiểu (Least Means Square-LMS). Thuật toán này thuộc dạng thuật toán xấp xỉ để tìm các điểm mà tại đó hiệu năng của mạng là tối ưu. Chỉ số tối ưu (performance index) thường được xác định bởi một hàm số của ma trận trọng số và các đầu vào nào đó mà trong quá trình tìm hiểu bài toán đặt ra.

Bỏ qua sự phức tạp về mặt toán học, thuận toán có thể phát biểu đơn giản như sau:

- **Bước 1**: ***Lan truyền xuôi các tính toán trong mạng truyền thẳng***
  - Khi đó, đầu ra của một lớp trở thành đầu vào của lớp kế tiếp. Phương trình thể hiện hoạt động này như sau (trong đó M là số lớp trong mạng) :

$$\mathbf{a}^{m+1} = \mathbf{f}^{m+1}(\mathbf{W}^{m+1}\mathbf{a}^{m} + \mathbf{b}^{m+1}) \text{ với } m = 0, 1, ..., M-1,$$

  - Các nơron trong lớp thứ nhất nhận các tín hiệu từ bên ngoài (với p chính là điểm bắt đầu của phương trình hình 2.14)

$$\mathbf{a}^{0} = \mathbf{p},$$

  - Đầu ra của lớp cuối cùng được xem là đầu ra của mạng:

$$\mathbf{a} = \mathbf{a}^{M}.$$

- **Bước 2**: ***Lan truyền lỗi (hay độ nhạy cảm) ngược lại qua mạng***
  - Thuật toán lan truyền ngược sử dụng chỉ số hiệu năng là trung bình bình phương lỗi của đầu ra so với giá trị đích. Đầu vào của thuật toán chính là tập các cặp mô tả hoạt động đúng của mạng:

$$\{(\mathbf{p}_1, \mathbf{t}_1), (\mathbf{p}_2, \mathbf{t}_2), ..., (\mathbf{p}_Q, \mathbf{t}_Q)\},$$

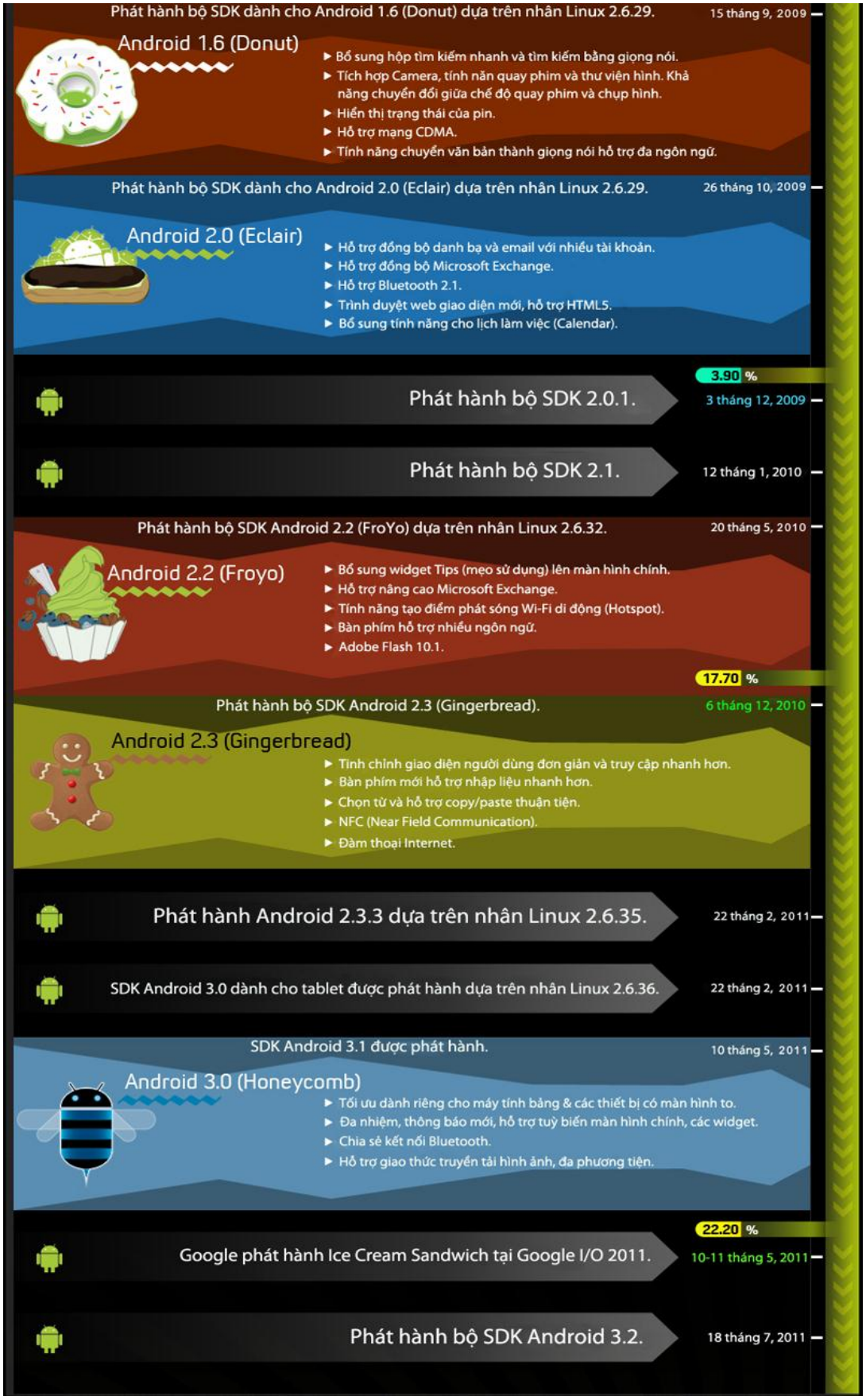
    Trong đó $\mathbf{p}_i$ là một đầu vào và $\mathbf{t}_i$ là đầu ra mong muốn tương ứng, với i=1..Q.
  - Mỗi đầu vào đưa vào mạng, đầu ra của mạng đối với nó được đem so sánh với đầu ra mong muốn.Thuật toán sẽ điều chỉnh các tham số của mạng để tối thiểu hóa trung bình bình phương lỗi.
- **Bước 3**: ***Cập nhật lại các trọng số và độ lệch tương ứng***

2.3 Nền Tảng Android

Android là tên gọi của một hệ điều hành mã nguồn mở dựa trên nhân linux, ban đầu được Google xây dựng dành cho các thiết bị di động nhưng hiện tại đã vươn ra TV, HD Player, Tablet…). Khởi đầu, Android được phát triển bởi công ty cùng tên, sau này được Google mua lại nhằm cạnh tranh với RIM và sau đó là Apple. Hiện tại, đây là nền tảng có mức tăng trưởng nhanh và cũng là hệ điều hành có thị phần lớn nhất. Với bản chất là hệ điều hành mã nguồn mở, Android hoàn toàn thích hợp cho mục đích nghiên cứu, học tập của sinh viên và nhân lực công nghệ thông tin. Đó chính là lý do Android được chọn để hoàn thành đồ án này.

2.3.1 Sự phát triển của Android

Lịch sử phát triển của hệ điều hành này có thể được tóm tắt như sau:

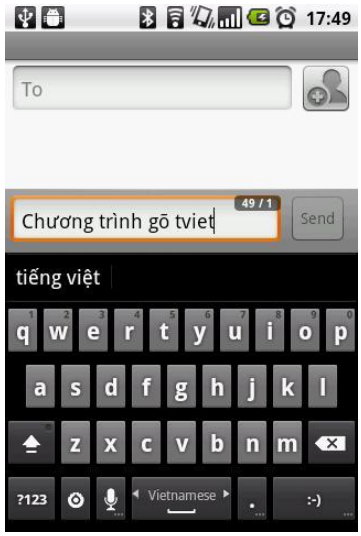


Hình 2.15 – Lịch sử phát triển Android

2.3.2 Những đặc điểm khác biệt của Android

Android được phát triển theo hướng mở hết mức có thể, người dùng hay các nhà phát triển có toàn quyền thay đổi giao diện, các thành phần hệ thống hay thậm chí là mã nguồn của hệ điều hành này tùy thích. Điều này khiến cho hệ điều hành này trở nên rất dễ tùy biến nhưng kèm theo đó là khó làm quen đối với những người chưa có kinh nghiệm. Tuy vậy, sự thành công của nó trên thị trường phần nào đã khẳng định được chất lượng so với các đối thủ.

Hình 2.16 – Một số giao diện của Android



Hình 2.17 – Bàn phím của Android

2.3.3 Máy ảo Dalvik

Trên Android, các nhà phát triển phần mềm có thể lựa chọn giữa ngôn ngữ C/C++ hoặc Java để viết ứng dụng. Đối với các ứng dụng viết bằng Java, Android thực thi chúng thông qua máy ảo – tương tự JVM trên máy bàn – gọi là Dalvik. Dalvik là một phần mềm mã nguồn mở được thiết kế và viết bởi Dan Bornstein dựa trên JVM nhưng được cải tiến để có thể hoạt động hiệu quả trên các thiết bị Android.

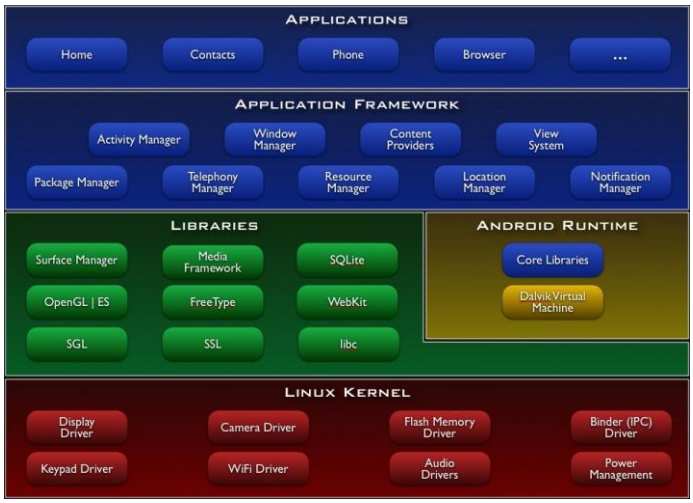
Điểm khác biệt của Dalvik so với JVM (hoạt động dựa trên thanh ghi – stack), chính là việc Dalvik dựa theo kiến trúc các register.

Mặc dù dùng chung ngôn ngữ Java, nhưng các file .class sẽ được dịch sang định dạng .dex nhằm làm giảm tối đa dung lượng của các chương trình. Các hằng hay chuỗi trùng nhau sẽ được tinh giảm trong file .dex. Các bytecode của Java cũng sẽ được chuyển thành một bộ chỉ thị của Dalvik. Điều này đem đến sự khác biệt về dung lượng cho các file .dex so với trước khi chuyển đổi.

Ngoài ra, máy ảo Dalvik còn được tối ưu hóa rất nhiều để có khả năng chạy cùng lúc nhiều ‘instance’ mà không tốn quá nhiều bộ nhớ và năng lực xử lý. Trong các phiên bản của Android, Dalvik liên tục được cải tiến để đem đến khả năng hoạt động hiệu quả hơn cho các ứng dụng Android.

2.3.4 Kiến trúc của Android

Mô hình sau thể hiện một cách tổng quát các thành phần của hệ điều hành Android. Mỗi một phần sẽ được đặc tả một cách chi tiết dưới đây.



Hình 2.18 – Kiến trúc của Android

2.3.4.1 Applications (Tầng ứng dụng)

Đây là tầng cao nhất của Android, bao gồm các ứng dụng chạy trên nền máy ảo Dalvik. Chủ yếu các chương trình trên tầng này đều được viết bằng ngôn ngữ Java, một số ít có thể kèm theo native code (C/C++) nếu như có yêu cầu đặc biệt về tốc độ xử lý hay khả năng can thiệp trực tiếp phần cứng.

2.3.4.2 Application Framework

Bằng cách cung cấp một nền tảng phát triển mở, Android cung cấp cho các nhà phát triển khả năng xây dựng các ứng dụng cực kỳ phong phú và sáng tạo. Nhà phát triển được tự do tận dụng các thiết bị phần cứng, thông tin địa điểm truy cập, các dịch vụ chạy nền, thiết lập hệ thống báo động, thêm các thông báo để các thanh trạng thái, và nhiều, nhiều hơn nữa. Để tăng khả năng re-use các thông tin cần thiết, cũng như khả năng trao đổi dữ liệu giữa các thành phần ứng dụng, Android đề ra một khái niệm là Framework, bao gồm tất cả các tài nguyên cần thiết cho việc trình bày giao diện, các API riêng của nhà sản xuất, cách quản lý phần cứng của Android…

Nhà phát triển có thể truy cập vào các API cùng một khuôn khổ được sử dụng bởi các ứng dụng lõi. Các kiến trúc ứng dụng được thiết kế để đơn giản hóa việc sử dụng lại các thành phần; bất kỳ ứng dụng có thể xuất bản khả năng của mình và ứng dụng nào khác sau đó có thể sử dụng những khả năng (có thể hạn chế bảo mật được thực thi bởi khuôn khổ). Cơ chế này cho phép các thành phần tương tự sẽ được thay thế bởi người sử dụng.

Cơ bản tất cả các ứng dụng là một bộ các dịch vụ và các hệ thống, bao gồm:

- Một tập hợp rất nhiều các **View** có khả năng kế thừa lẫn nhau dùng để thiết kế phần giao diện ứng dụng như: gridview, tableview, linearlayout,…
- Một "**Content Provider**" cho phép các ứng dụng có thể truy xuất dữ liệu từ các ứng dụng khác (chẳng hạn như Contacts) hoặc là chia sẻ dữ liệu giữa các ứng dụng đó.
- Một "**Resource Manager**" cung cấp truy xuất tới các tài nguyên không phải là mã nguồn, chẳng hạn như: localized strings, graphics, and layout files.

- Một "**Notification Manager**" cho phép tất cả các ứng dụng hiển thị các custom alerts trong status bar.
- **Activity Manager** được dùng để quản lý chu trình sống của ứng dụng và điều hướng các activity.

2.3.4.3 Library (Thư viện)

Android bao gồm một tập hợp các thư viện C/C++ được sử dụng bởi nhiều thành phần khác nhau trong hệ thống Android. Điều này được thể hiện thông qua nền tảng ứng dụng Android. Một số các thư viện cơ bản được liệt kê dưới đây:

- **System C library**: một BSD-derived triển khai các thư viện hệ thống ngôn ngữ C chuẩn, được nhúng vào các thiết bị dựa trên hệ điều hành Linux.
- **Media Libraries** – Dựa trên PacketVideo's OpenCORE; thư viện này hỗ trợ cho việc chơi nhạc, quay phim, chụp hình theo các định dạng file MPEG4, H.264, MP3, AAC, AMR, JPG, và PNG
- **Surface Manager** – Quản lý truy cập đến các hệ thống con hiển thị cũng như các lớp đồ họa 2D, 3D từ tầng ứng dụng.
- **LibWebCore** – Thư viện được dùng để tạo nên thành phần webview trong Android và có thể nhúng được vào nhiều ứng dụng.
- **SGL** – Thư viện hỗ trợ đồ họa 2D.
- **3D libraries** – Thư viện đồ họa 3D (chủ yếu là OpenGL ES.
- **FreeType** – thư viện render font chữ.
- **SQLite** – Một cơ sở dữ liệu nhỏ được dùng cho các thiết bị cầm tay có bộ nhớ hạn chế. SQLite không có "quan hệ" như các cơ sở dữ liệu khác.

2.3.4.4 Android Runtime (Môi trường thực thi)

Android bao gồm một tập hợp các thư viện cơ bản mà cung cấp hầu hết các chức năng có sẵn trong các thư viện lõi của ngôn ngữ lập trình Java. Tất cả các ứng dụng Android đều chạy trong tiến trình riêng. Máy ảo Dalvik đã được viết để cho một thiết bị có thể chạy nhiều máy ảo hiệu quả. Các VM Dalvik thực thi các tập tin thực thi Dalvik (dex). Định dạng được tối ưu hóa cho bộ nhớ tối thiểu. VM là dựa trên register-based, và chạy các lớp đã được biên dịch bởi một trình biên dịch Java để

chuyển đổi thành các định dạng dex. Các VM Dalvik dựa vào nhân Linux cho các chức năng cơ bản như luồng và quản lý bộ nhớ cấp thấp.

2.3.4.5 Linux Kernel

Đây là nền tảng cơ bản nhất của hệ điều hành Android, đảm nhiệm vai trò giao tiếp phần cứng, điều khiển các chức năng cơ bản nhất của thiết bị và cung cấp các tính năng thiết yếu như quản lý bộ nhớ, quản lý luồng, kết nối mạng… bằng hàng loạt các driver do nhà sản xuất viết cho thiết bị của họ.

2.3.5 Các thành phần trong một dự án ứng dụng Android

2.3.5.1 AndroidManifest.xml

Trong bất kì một dự án Android nào khi tạo ra đều có một file AndroidManifest.xml, file này được dùng để định nghĩa các màn hình sử dụng, các quyền cũng như các giao diện cho ứng dụng. Đồng thời nó cũng chứa thông tin về phiên bản SDK cũng như màn hình chính sẽ chạy đầu tiên.

File này được tự động sinh ra khi tạo một dự án Android. Trong file manifest bao giờ cũng có 3 thành phần chính đó là: *application*, *permission* và *version*.

Dưới đây là nội dung của một file AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="nhox.example.gridview"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" />

    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".HelloGridViewActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>
</manifest>
```

Hình 2.19 – Kiến trúc file XML

- **Application**

Thẻ <*application*>, bên trong thẻ này chứa các thuộc tính được định nghĩa cho ứng dụng Android như:

- **android:icon** = **"**drawable resource**"** →Ở đây đặt đường dẫn đến file icon của ứng dụng khi cài đặt. VD: android:icon = "@drawable/icon".
- **android:name** = **"**string**"** →thuộc tính này để đặt tên cho ứng dụng Android. Tên này sẽ được hiển thị lên màn hình sau khi cài đặt ứng dụng.
- **android:theme** = **"**drawable theme**"** → thuộc tính này để đặt theme cho ứng dụng. Các theme là các cách để hiển thị giao diện ứng dụng.

Ngoài ra còn nhiều thuộc tính khác…

- **Permission**

Bao gồm các thuộc tính chỉ định quyền truy xuất và sử dụng tài nguyên của ứng dụng. Khi cần sử dụng một loại tài nguyên nào đó thì trong file manifest của ứng dụng cần phải khai báo các quyền truy xuất như sau:

```
<uses-permission
android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission
android:name="android.permission.ACCOUNT_MANAGER"/>
<uses-permission
android:name="android.permission.VIBRATE" />
<uses-permission
android:name="android.permission.CALL_PHONE"/>
```

- **SDK version**

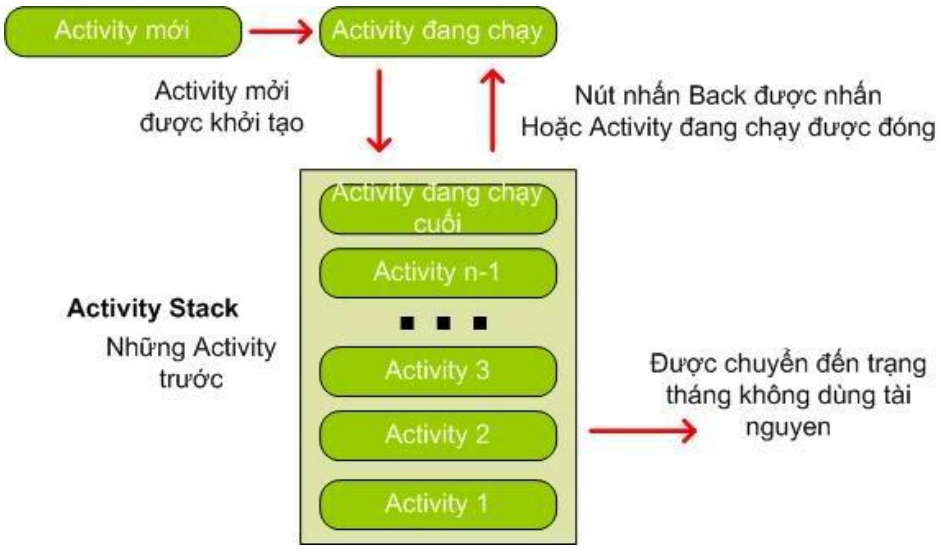
Thẻ xác định phiên bản SDK được khai báo để chỉ ra phiên bản SDK thấp nhất có thể sử dụng, như sau:

<uses-sdk android:`minSdkVersion`="7" />.

2.3.5.2 Activity

Activity có thể được xem như một window trên môi trường Windows phổ biến như hiện tại. Tại mỗi thời điểm chỉ có duy nhất 1 Activity có thể ghi nhận các thao tác trên màn hình cảm ứng từ người dùng, các activity khác đều được đặt vào trạng thái chờ hoặc bị xóa khỏi bộ nhớ. Khi người dùng kết thúc phiên làm việc với Activity

nào, các activity đang xếp hàng phía sau sẽ được đẩy lên đầu trở thành Running Activity. Cấu trúc này được gọi là Activity Stack.

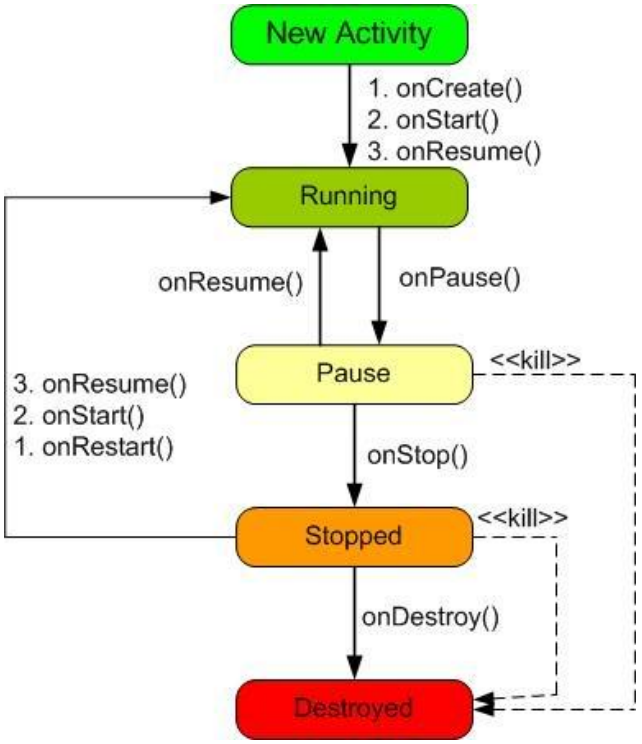


Hình 2.20 – Activity Stack

Một Activity chủ yếu có 3 chu kỳ chính sau:

- **Active hoặc running**: Khi Activity là được chạy trên màn hình. Activity này tập trung vào những thao tác của người dùng trên ứng dụng.
- **Paused**: Activity là được tạm dừng (paused) khi mất tiêu điểm (focus) nhưng người dùng vẫn trông thấy. Có nghĩa là một Activity mới ở trên nó nhưng không bao phủ đầy màn hình. Một Activity tạm dừng là còn sống nhưng có thể bị kết thúc bởi hệ thống trong trường hợp thiếu vùng nhớ.
- **Stopped**: Nếu nó hoàn toàn bao phủ bởi Activity khác. Nó vẫn còn trạng thái và thông tin thành viên trong nó. Người dùng không thấy nó và thường bị loại bỏ trong trường hợp hệ thống cần vùng nhớ cho tác vụ khác.

Ba chu kỳ của Activity được thể hiện rõ hơn bằng biểu đồ như hình sau:



Hình 2.21 – Vòng đời của một Activity

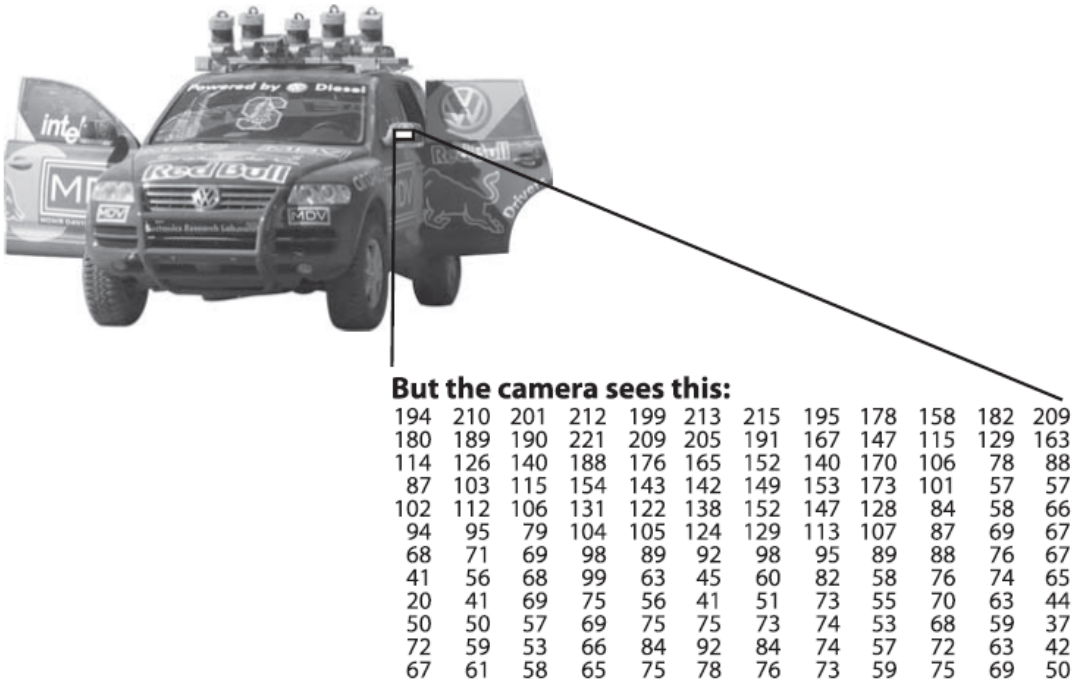
2.4 Thư Viện Xử Lý Ảnh OpenCv

2.4.1 Vài nét về Computer Vision

Computer Vision (hay CV) là một phương thức hay phương cách được dùng để chuyển hóa các dữ liệu hình ảnh dạng tĩnh hay các đoạn video từ camera thành một dạng "nhận thức" hay "đặc tả" – như 'xe máy', 'có 3 chiếc xe máy đang chạy', 'mặt trời đang lên'… Việc nhận thức này có vẻ rất dễ dàng với con người, nhưng thực chất lại không hề đơn giản trong thế giới số. Bởi, thực chất, khi xử lý hình ảnh, não người thực hiện hàng loạt công đoạn phân tích, xử lý, tái tạo, liên kết… các luồng thông tin khác nhau để đi đến kết luận cuối cùng về sự vật. Các công đoạn này được thực hiện hoàn toàn nhờ vào các cơ quan đặc thù của vỏ não, không cần sự can thiệp của con người.

Nhưng, đối với máy tính, tất cả các cơ quan trên đều không tồn tại, tất cả những gì máy tính có thể nhận biết và xử lý là các tín hiệu số. Các hình ảnh khi chuyển về định dạng số sẽ là một ma trận các con số, không có biên giới rõ ràng, không có tri thức nào về hình ảnh đó trong quá khứ, không có khả năng liên hệ hay

làm rõ nét một chi tiết nào của ảnh…, các lập trình viên phải thực hiện tất cả những việc này bằng tay. Thêm vào đó, dữ liệu hình ảnh trên máy tính nhận được là những dữ liệu đã bị nhiễu và ảnh hưởng rất nhiều từ những sự thay đổi rất nhỏ như thay đổi độ sáng, thay đổi phông nền, hay thay đổi góc nhìn đối với sự vật. Chính những điều này đã khiến cho việc nhận dạng hình ảnh trên máy tính trở nên rất khó khăn.



Hình 2.22 – Ví dụ về Computer Vision

Tuy không dễ dàng, nhưng nhờ có bề dày lịch sử và kế thừa, Computer Vision cũng đạt được một số bước tiến nhất định với một số bộ thư viện xử lý hình ảnh hiện tại, nhưng vẫn còn rất xa so với mục tiêu cuối cùng. Tuy vậy, tương lai của CV vẫn rất tươi sáng.

2.4.2 Một số thư viện xử lý ảnh tiêu biểu

Nhằm mục đích giảm nhẹ gánh nặng cho các lập trình viên trong các tác vụ xử lý hình ảnh, các thư viện Computer Vision đã ra đời. Nhờ các thư viện này, lập trình viên có thể tránh được việc phải viết lại từ những hàm cơ bản nhất, mà chỉ cần tập trung vào việc sử dụng chúng sao cho hiệu quả nhất cho mục đích của mình. Một số thư viện xử lý hình ảnh có thể kể đến như sau:

- **VXL**(http://vxl.sourceforge.net/)

Đây là thư viện xử lý hình ảnh ở mức cơ bản, chuyên về các thao tác chỉnh sửa hình ảnh như đổi màu sắc, thay đổi kích thước ảnh… VXL được xây dựng trên nền tảng ngôn ngữ ANSI/ISO C++ nhằm tăng khả năng tương thích và tốc độ hoạt động của tác vụ xử lý ảnh. Hiện tại VXL đã xuất hiện trên Linux, Windows và Mac OS nhưng vẫn chưa xuất hiện trên nền tảng di động nào.

- **Camellia** (http://camellia.sourceforge.net/)

Bộ thư viện này được viết hoàn toàn trên ngôn ngữ C và được tối ưu hóa rất nhiều nhằm đem đến tốc độ hoạt động thời gian thực. Camellia xuất hiện trên cả Linux, Windows, Mac OS nhưng chủ yếu hỗ trợ cho ngôn ngữ Ruby và chưa có kế hoạch xuất hiện trên nền tảng di động nào.

- **OpenVIDIA**
  (http://openvidia.sourceforge.net/index.php/OpenVIDIA)

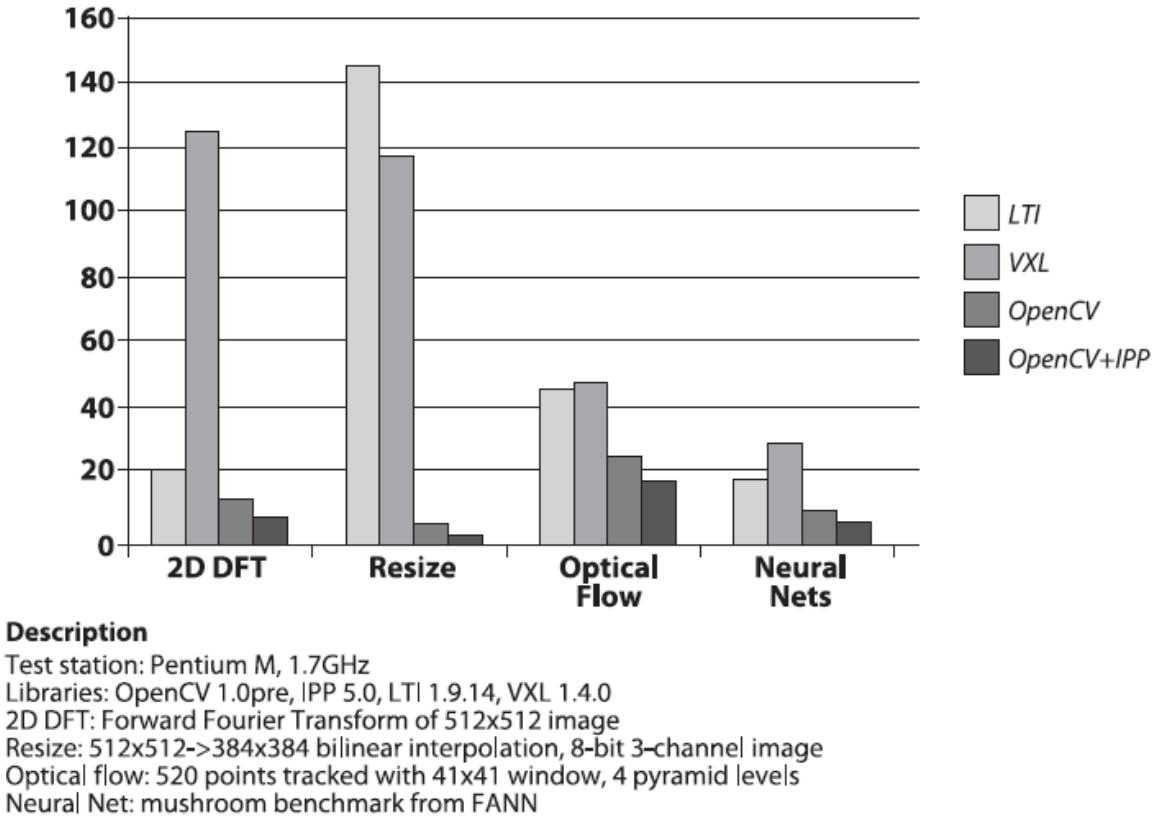
OpenVIDIA là bộ thư viện xử lý hình ảnh được nVidia khởi xướng nhằm mục đích tận dụng sức mạnh xử lý vốn rất dư dả từ các GPU (mạnh về xử lý song song). Bộ thư viện này tận dụng rất nhiều công nghệ của nVidia như CUDA, NPP và hỗ trợ các thư viện mã nguồn mở như OpenGL, OpenCL… Nhờ tận dụng khả năng xử lý mạnh mẽ của GPU (vốn mạnh hơn nhiều so với CPU), bộ thư viện này cho tốc độ xử lý nhanh hơn khá nhiều so với các giải pháp hiện tại. Bộ thư viện này hỗ trợ đầy đủ các giải pháp nhận dạng hình ảnh, xử lý hình ảnh thời gian thực…

OpenVIDIA hiện đã có các giải pháp phần mềm xuất hiện trên nền tảng Tegra 2 của di động, nhưng nhóm không chọn bộ thư viện này vì nó không thể phổ biến rộng rãi lên các nền tảng không sử dụng GPU của nVidia.

- **NCV – Nokia Computer Vision**
  (http://research.nokia.com/page/221)

Bộ thư viện xử lý hình ảnh được Nokia xây dựng dành riêng cho hệ điều hành Symbian của mình, cung cấp các khả năng xử lý hình ảnh như thay đổi kích thước, thay đổi khung hình, dựng hình, dựng giao diện người dùng… nhưng lại thiếu đi các hàm mang tính cần thiết cho mục tiêu của đề tài – nhận dạng hình ảnh. Bộ thư viện này cũng không hỗ trợ các nền tảng khác như Android, iOS.

Nhóm đã tham khảo qua rất nhiều các thư viện xử lý ảnh khác nhau, nhưng hầu hết đều bị giới hạn ở khả năng hoạt động trên nền tảng di động - ở đây là Android. Trong quá trình tìm tòi, nhóm chỉ phát hiện được OpenCV là bộ thư viện hỗ trợ đầy đủ nhất cho Android, đồng thời cũng có sức mạnh xử lý không hề thua kém các thư viện đã nêu. Nên, trong phạm vi đồ án lần này, nhóm đã chọn thực hiện dựa trên OpenCV.



Hình 2.23 – Tốc độ xử lý của OpenCV so với LTI và VXL

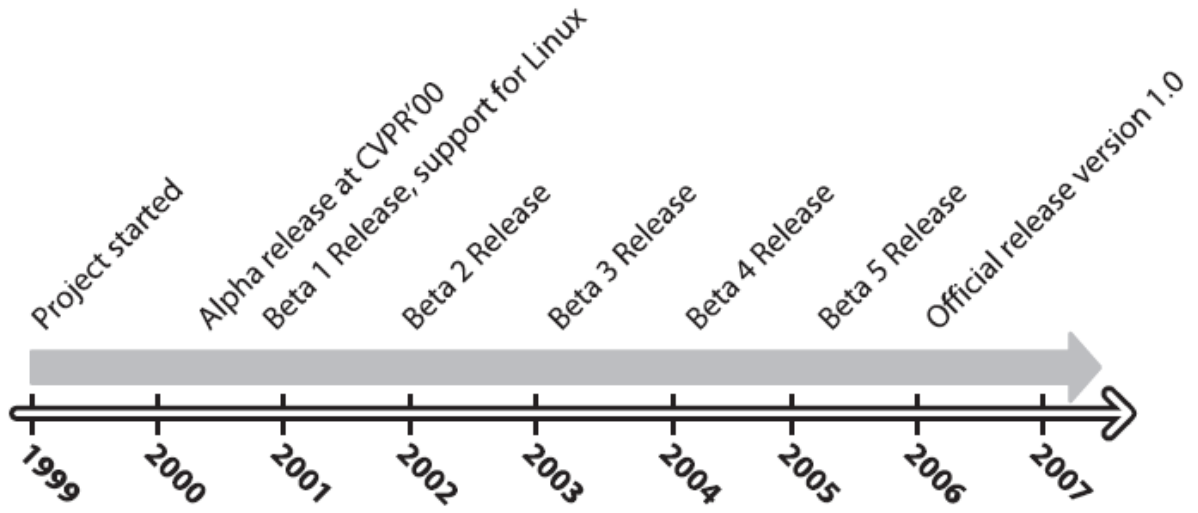
2.4.3 Thư viện OpenCV

OpenCV là một bộ thư viện xử lý hình ảnh mã nguồn mở được host trên http://SourceForge.net/projects/opencvlibrary. Bộ thư viện được viết trên ngôn ngữ C/C++ nhờ đó nó có khả năng hoạt động trên gần như mọi nền tảng như Linux, Windows và MacOS. Ngoài ra, nhờ sự ủng hộ nhiệt tình của cộng đồng mã nguồn mở, OpenCV còn có khả năng hoạt động trên các nền tảng khác như Java, nhúng,... và đặc biệt là Android.

OpenCV ban đầu được thiết kế cho các thiết bị điện toán với sức mạnh rất lớn, được tối ưu hóa trên nền ngôn ngữ C và khả năng tận dụng các bộ vi xử lý đa luồng – đa nhân nhằm hiện thực hóa các ứng dụng xử lý hình ảnh thời gian thực. Do đó, khi được chuyển lên các nền tảng di động, bộ thư viện này bị thuyên giảm khá nhiều sức mạnh so với phiên bản gốc, nhưng vẫn là một trong những bộ thư viện tốt nhất cho thế giới mobile.

2.4.3.1 Lịch sử của OpenCV

OpenCV có một khởi đầu khá khiêm nhường ở Media Lab MIT, chỉ là một bộ thư viện được chuyền tay giữa các sinh viên, đóng vai trò là một nền tảng ban đầu cho các ứng dụng xử lý hình ảnh của họ, để họ không mất thời gian viết lại các hàm mang tính căn bản mà có thể dành nhiều thời gian để tập trung vào các chức năng cao cấp hơn. Bộ thư viện này được bộ phận nghiên cứu của Intel chú ý và đầu tư phát triển. Tính từ khi bản thử nghiệm Alpha đầu tiên của OpenCV xuất hiện năm 1999, đến nay bộ thư viện này đã có hơn 500 hàm chức năng bao trùm hầu như toàn bộ các lĩnh vực xử lý hình ảnh như theo dõi sản phẩm công nghiệp, các hình ảnh y tế, camera an ninh, giao diện người dùng, tinh chỉnh camera, robotic… Ngoài ra, OpenCV còn được kèm theo thư viện Machine Learning (MLL) nhằm tăng cường khả năng học và nhận dạng hình ảnh chính xác hơn.



Hình 2.24 – Lịch sử phát triển của OpenCV

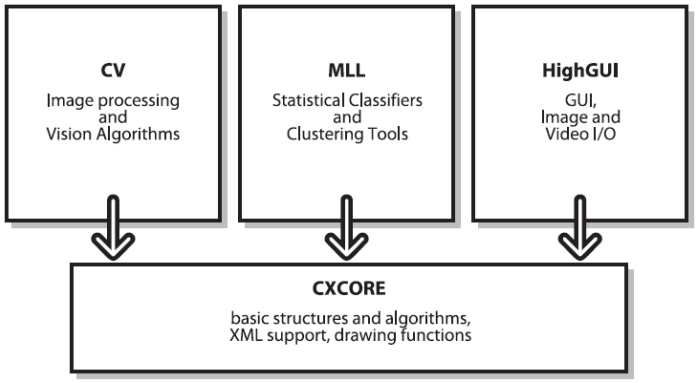
Do bản chất là một bộ thư viện mã nguồn mở, nhưng giấy phép của OpenCV lại cho phép người dùng có thể sử dụng một phần hay toàn bộ mã nguồn của nó cho các ứng dụng có thu phí của họ mà không phải đóng góp gì lại cho cộng đồng, kể cả việc công bố mã nguồn của họ. Chính nhờ điều này nên OpenCV nhận được rất nhiều sự quan tâm của các công ty phần mềm lớn như IBM, Intel, Microsoft,… và hiện tại lượng thành viên của project này đã lên đến hơn 20,000 người trên khắp các châu lục.

2.4.3.2 Cấu trúc của OpenCV

OpenCV là một bộ thư viện rất lớn, được cấu trúc thành 5 phần chính, trong đó có 4 phần được phân chia như hình 2.25 dưới đây. Trong đó, CV chứa các hàm xử lý hình ảnh căn bản và các thuật toán nhận dạng cấp cao; MLL là bộ thư viện phục vụ cho ngành máy học. HighGUI chứa các lệnh nhập xuất và hàm để truy xuất hình ảnh, video; CXCore chứa các cấu trúc cơ bản, các hàm vẽ và các hàm dùng chung cho toàn bộ thư viện.

Trong hình bên dưới, không bao gồm CvAux, thành phần chứa các thuật toán còn trong giai đoạn thử nghiệm hoặc các hàm bị ngừng phát triển. Trong CvAux, người dùng có thể tìm thấy khá nhiều hàm có khả năng sẽ xuất hiện chính thức trong OpenCV trong tương lai. Có thể kể đến như:

- Nhận dạng cử chỉ từ camera
- Đặc tả các texture (vân bề mặt)
- Theo dõi mắt và miệng
- Theo dõi 3D
- Tìm khung xương của vật thể



Hình 2.25 – Cấu trúc thư viện OpenCV

2.4.3.3 OpenCV trên Android

Hiện tại, OpenCV đã cung cấp sẵn các wrapper dành cho hệ điều hành Android trên trang chủ của mình (http://opencv.willowgarage.com/wiki/Android) và cho người dùng chọn lựa giữa việc sử dụng thư viện đã biên dịch sẵn hay dùng trực tiếp mã nguồn của OpenCV. Nếu không có yêu cầu đặc biệt về các tính năng hay tối ưu thêm nữa các hàm của OpenCV, lựa chọn đầu tiên sẽ dễ dàng hơn cho đại đa số người dùng.

Sau khi tải và cài đặt OpenCV lên môi trường lập trình cho Android, người dùng đã có thể sử dụng các tính năng đặc biệt của bộ thư viện này một cách nhanh chóng.

CHƯƠNG 3 : ỨNG DỤNG NHẬN DẠNG BIỂN BÁO GIAO THÔNG

Chương này trình bày các vấn đề sau:

3.1 Mô tả bài toán

3.2 Mô hình giải quyết bài toán

3.3 Thiết kế chương trình

3.4 Thực nghiệm

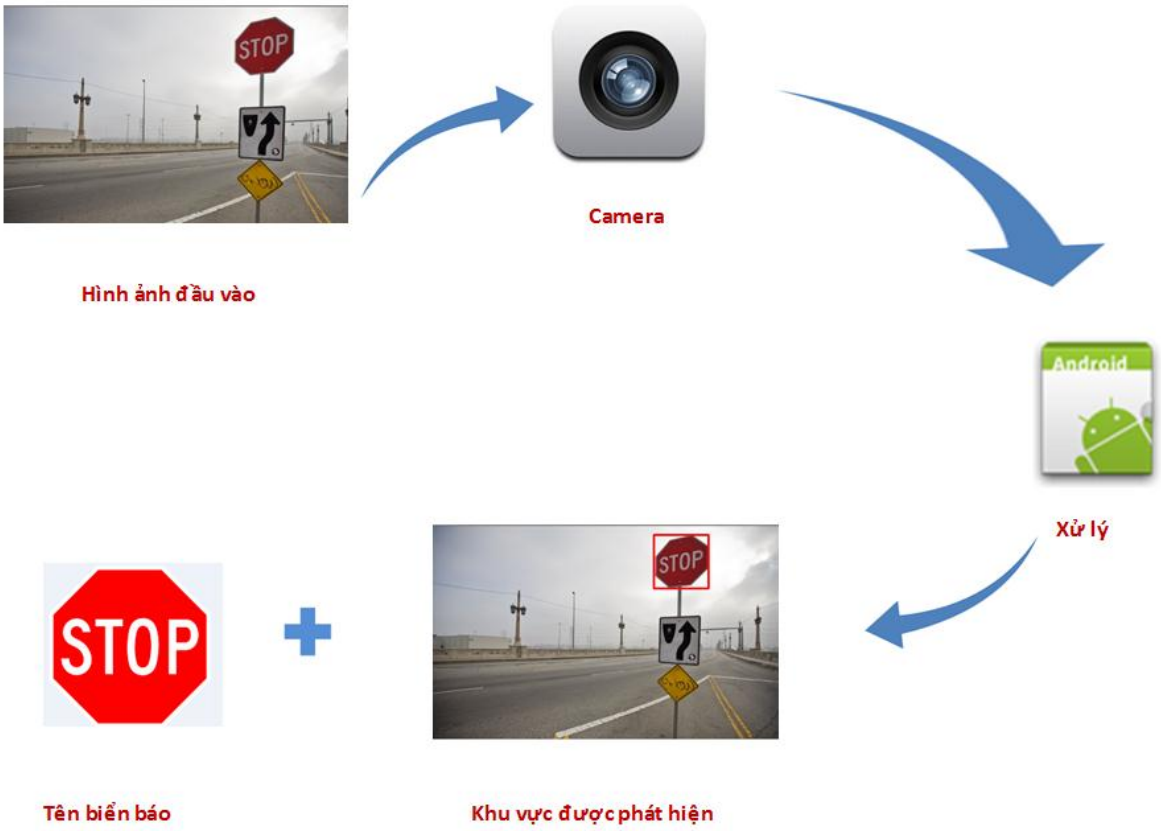
3.1 Mô Tả Bài Toán

3.1.1 Đặt vấn đề

Biển báo giao thông là cách thức để thông báo cho người tham gia giao thông tình trạng đường, đưa ra những chỉ dẫn, hay cảnh báo giúp người tham gia giao thông xử lý và đưa ra những hành vi hợp lý, đảm bảo an toàn giao thông. Mỗi người tham gia giao thông đòi hỏi phải nắm bắt và hiểu hết toàn bộ ý nghĩa của tất cả các biển báo giao thông. Việc này trở nên khá khó khăn khi số lượng biển báo quá lớn. Hơn thế nữa việc phát hiện biển báo trở nên phức tạp khi mà các yếu tố ngoại cảnh tác động. Biển báo có thể bị hư hại khi tiếp xúc một thời gian dài dưới ánh nắng mặt trời. Màu sắc bị thay đổi trong các điều kiện sương mù, ánh sáng yếu ban đêm, biển báo bị che khuất bởi nhà cửa, cây cối …

Vì những lý do trên, nhóm mong muốn xây dựng một ứng dụng nhỏ gọn, cầm tay, đơn giản cho phép người dùng có thể tra cứu trực quan thông tin của biển báo khi không nhớ nội dung biển báo này, phát triển hơn nữa về sau có thể là tích hợp hoặc kết nối tới các phương tiện giao thông và đưa ra các cảnh báo thích hợp.

Chức năng chính cơ bản của ứng dụng là tra cứu thông tin trực quan. Người dùng sử dụng điện thoại có cài ứng dụng, chụp hình hoặc quét camera qua khung cảnh có hình biển báo. Ứng dụng sẽ tự động phát hiện (detect) các khu vực có hình dạng biển báo và để xuất hình ảnh cùng thông tin biển báo mà ứng dụng nhận ra.



Hình 3.1 – Bài toán nhận dạng biển báo giao thông

3.1.2 Đối tượng của bài toán

Đối tượng của bài toán là các biển báo giao thông của Việt Nam. Trong giới hạn của luận văn này chúng ta sẽ xét tới các biển báo giao thông đường bộ. Số lượng biển báo giao thông đường bộ của Việt Nam khoảng hơn 200 biển báo và chia làm nhiều dạng:

- **Biển báo cấm đường bộ Việt Nam:** Nhóm biển báo cấm gồm có 39 kiểu được đánh số thứ tự từ biển số 101 đến biển số 139 nhằm báo điều cấm hoặc hạn chế mà người sử dụng đường bộ phải tuyệt đối tuân theo. Các biển báo loại này thường có hình dạng tròn với viền bao xung quanh màu đỏ đặc trưng, cá biệt có một số biển báo không thuộc dạng tròn nhưng số lượng không đáng kể.

Số hiệu biển báo:101
Tên biển báo: Đường cấm
Chi tiết báo hiệu: Đường cấm tất cả các loại phương tiện (cơ giới và thô sơ) đi lại cả hai hướng, trừ các xe được ưu tiên theo luật lệ nhà nước quy định.

Số hiệu biển báo: 102
Tên biển báo: Cấm đi ngược chiều
Chi tiết báo hiệu: Đường cấm tất cả các loại xe (cơ giới và thô sơ) đi vào theo chiều đặt biển, trừ các xe được ưu tiên theo luật lệ nhà nước quy định.

Số hiệu biển báo: 103a
Tên biển báo: Cấm ôtô
Chi tiết báo hiệu: Đường cấm tất cả các loại xe cơ giới kể cả môtô 3 bánh có thùng đi qua, trừ môtô hai bánh, xe gắn máy và các xe được ưu tiên theo luật lệ nhà nước quy định.

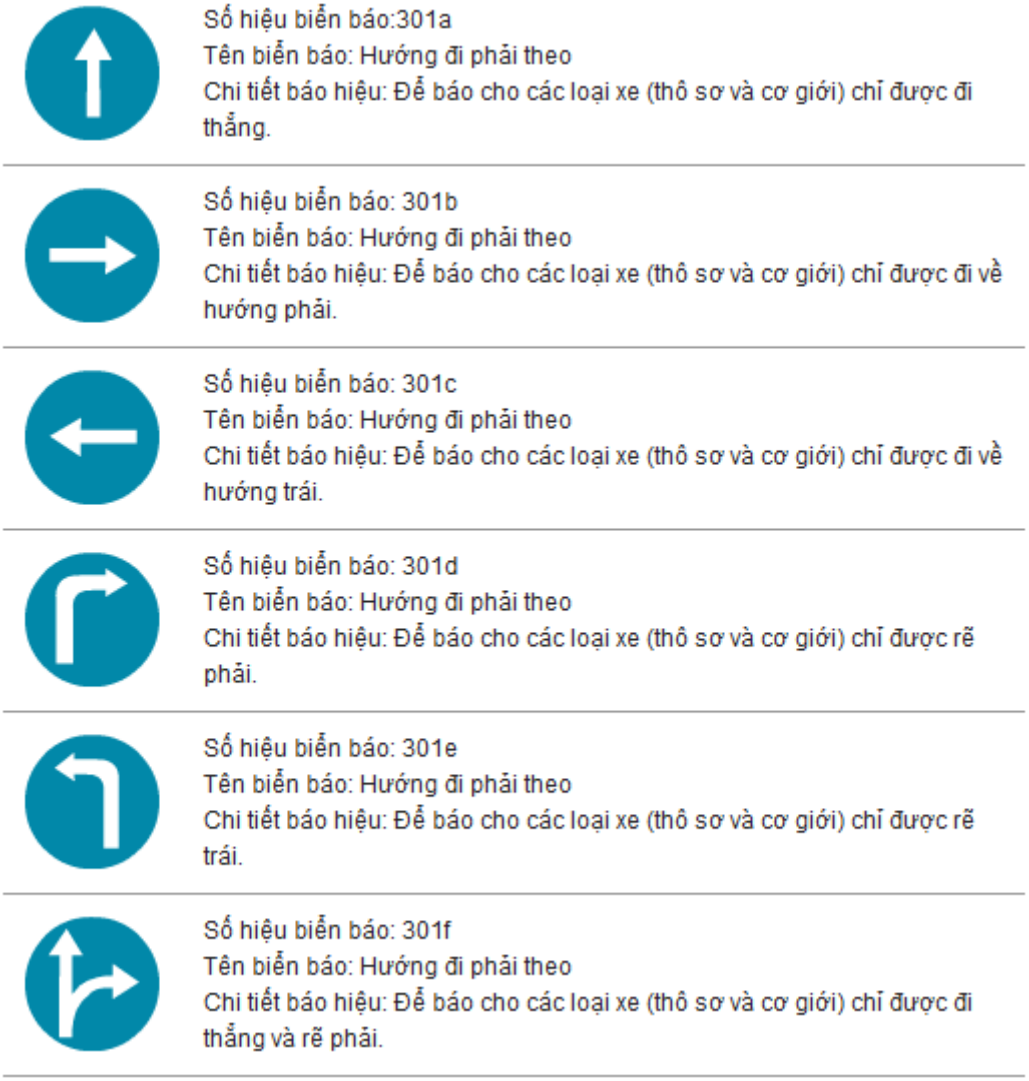
Số hiệu biển báo: 103b
Tên biển báo: Cấm ôtô rẽ phải
Chi tiết báo hiệu: Đường cấm tất cả các loại xe cơ giới kể cả môtô 3 bánh có thùng xe rẽ phải, trừ môtô hai bánh, xe gắn máy và các xe được ưu tiên theo luật lệ nhà nước.

Số hiệu biển báo: 103c
Tên biển báo: Cấm ôtô rẽ trái
Chi tiết báo hiệu: Đường cấm tất cả các loại xe cơ giới kể cả môtô 3 bánh có thùng xe rẽ trái, trừ môtô hai bánh, xe gắn máy và các xe được ưu tiên theo luật lệ nhà nước.

Số hiệu biển báo: 104
Tên biển báo: Cấm môtô
Chi tiết báo hiệu: Đường cấm tất cả các loại môtô đi qua, trừ các xe môtô được ưu tiên theo luật lệ nhà nước quy định.

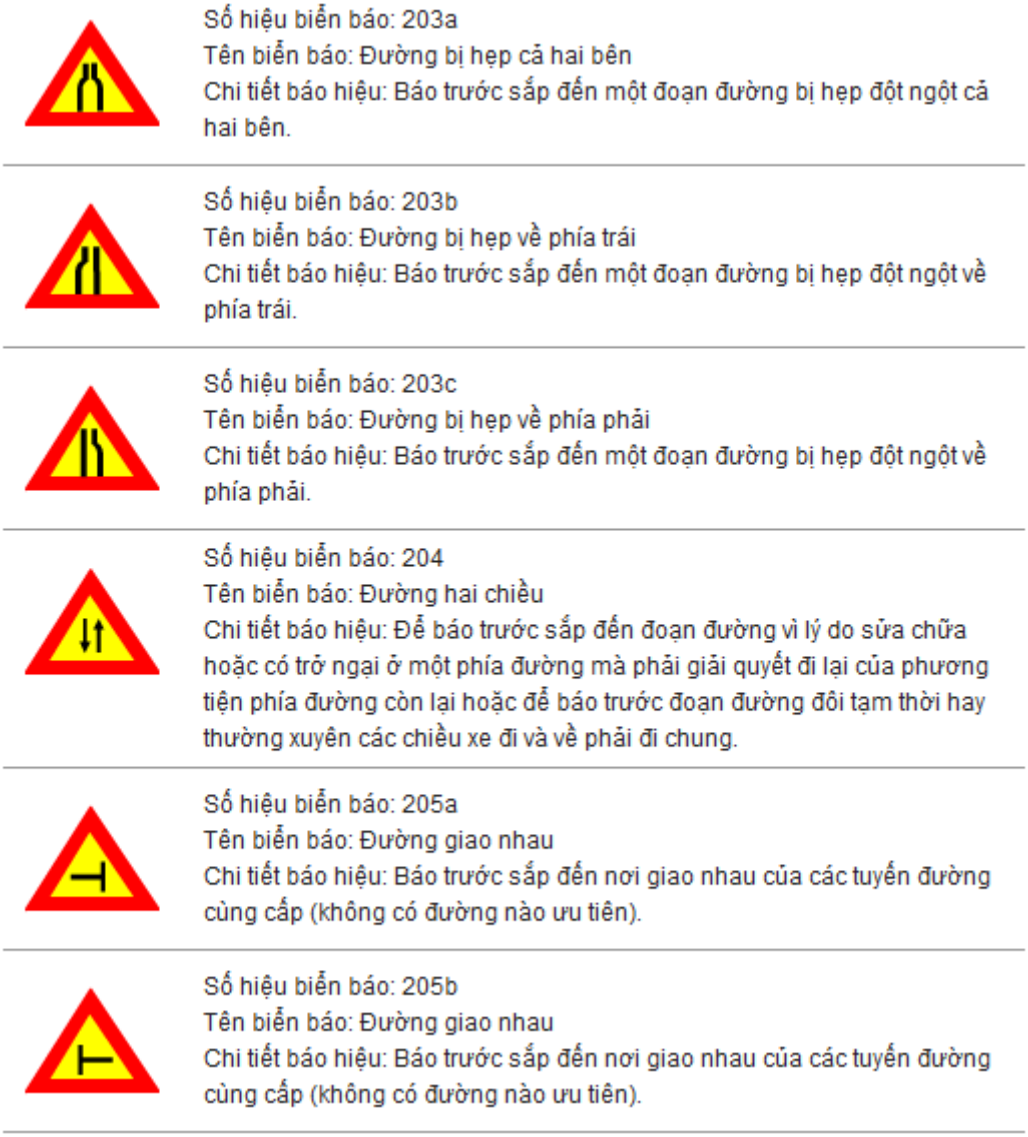
Hình 3.2 – Một số mẫu biển báo cấm

- **Biển hiệu lệnh đường bộ Việt Nam:** Nhóm biển hiệu lệnh gồm có 9 kiểu được đánh số thứ tự từ biển số 301 đến biển số 309 nhằm báo cho người sử dụng đường biết hiệu lệnh phải thi hành. Các biển báo loại này thường có hình dạng tròn với nền màu xanh đặc trưng, cá biệt có một số biển báo có đường kẻ sọc cắt ngang.



Hình 3.3 – Một số mẫu biển báo hiệu lệnh

- **Biển báo nguy hiểm đường bộ Việt Nam:** Biển báo nguy hiểm gồm 46 kiểu được đánh số thứ tự từ biển số 201 đến biển số 246 nhằm báo cho người sử dụng đường bộ biết trước tính chất của sự nguy hiểm trên đường để có biện pháp phòng ngừa, xử trí. Các biển báo loại này thường có hình dạng tam giác với viền bao xung quanh màu đỏ đặc trưng, nền vàng, cá biệt có một số biển báo không phải hình tam giác.



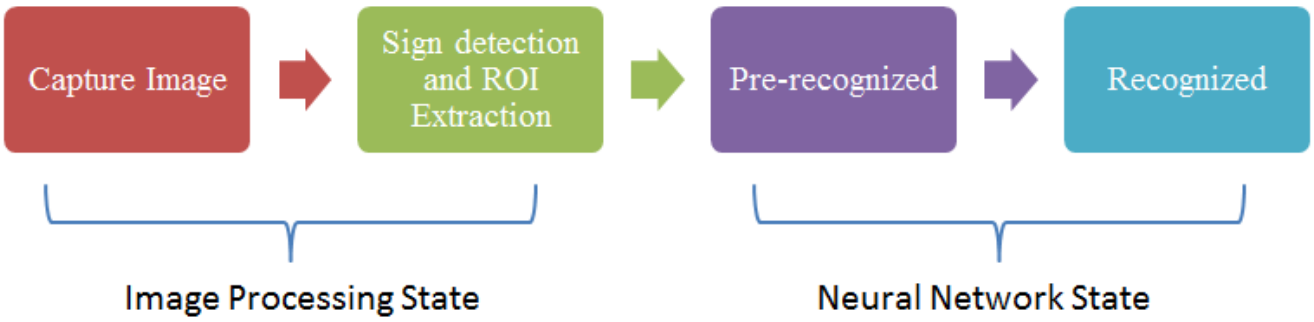
Hình 3.4 – Một số mẫu biển báo nguy hiểm

Ngoài các dạng biển báo trên thì còn có biển chỉ dẫn hình chữ nhật, biển phụ đường bộ, vạch kẻ đường … Vì tính chất phức tạp của các loại biển báo này mà luận văn sẽ không nghiên cứu đến. Giới hạn luận văn chỉ tập trung vào nhận biết 3 loại biển báo phía trên.

3.2 Mô Hình Giải Quyết Bài Toán

3.2.1 Mô hình tổng quát

Để giải quyết bài toán phát hiện và nhận dạng biển báo giao thông, nhóm đề xuất mô hình 4 bước chia làm 2 giai đoạn như sau:



Hình 3.5 – Mô hình giải quyết bài toán

- **Giai đoạn xử lý ảnh (Image Processing State)**:

  Mục đích cuối cùng của giai đoạn này là thu được hình ảnh của biển báo giao thông cần nhận dạng. Bước đầu tiên là thu thập dữ liệu hình ảnh nền khung cảnh (**Capture Image**). Hình ảnh thu được thông qua camera của thiết bị có thể là dữ liệu tĩnh (chụp hình) hoặc dạng real-time (xử lý trên từng frame của video). Vì thiết bị sử dụng là thiết bị kỹ thuật số nên ảnh thu được sẽ là ảnh đã được số hóa. Trên dữ liệu ảnh này tùy theo điều kiện môi trường mà có biện pháp thích hợp để nâng cao chất lượng ảnh, lọc nhiễu, chỉnh độ sáng …

  Bước thứ hai trong giai đoạn này là phát hiện biển báo và trích xuất vùng đặc trưng (**Sign detection and ROI Extraction**). Bằng các thuật toán xử lý ảnh thích hợp và dựa trên đặc trưng cụ thể của biển báo giao thông ở Việt Nam mà ứng dụng sẽ xử lý trên ảnh thu nhận từ bước một, lọc bỏ ảnh nền, phát hiện và trích xuất các khu vực đặc trưng có khả năng là hình ảnh của biển báo. Kết quả thu được sau giai đoạn này là 1 tập các hình ảnh biển báo giao thông đã lọc bỏ hình nền mà ứng dụng phát hiện được.

- **Giai đoạn tính toán mạng Nơron (Neutral Network State)**:

  Ở giai đoạn này hình ảnh có khả năng là biển báo giao thông thu được ở giai đoạn trước sẽ được xử lý và nhận dạng. Trước hết các hình ảnh này sẽ được thay đổi kích thước lại theo một mẫu chuẩn (thuật ngữ tiếng anh gọi là "blob"). Mẫu chuẩn này sẽ có kích thước 30*30 pixel. Từ các mẫu này, dữ liệu được xử lý và chuyển thành một dạng khác mà mạng noron có thể xử lý (bước này gọi là tiền nhận dạng – Pre_recognized). Kết quả thu được sẽ là một tập các giá trị input đầu vào cho mạng noron.

  Bước cuối cùng trong giai đoạn này chính là tính toán trong mạng noron. Tập giá trị input sẽ được đưa vào mạng, các tính toán sẽ được thực hiện và lan truyền trong mạng cho đến khi tới đầu ra của mạng. Bản chất mạng noron này đã được huấn luyện dựa trên một tập mẫu các biển báo giao thông trước đó. Từ kết quả của đầu ra của mạng noron ta có thể xác định được tên của biển báo giao thông nếu biển báo này đã được mạng noron học trước đó.

  Khó khăn chủ yếu trong giai đoạn này là khả năng nhận dạng biển báo sẽ là không cao nếu hình ảnh cần nhận dạng khác biệt quá nhiều so với tập mẫu mà mạng noron đã được học. Mạng noron được học càng nhiều mẫu của một loại biển báo thì khả năng nhận dạng ra biển báo đó càng cao, điều này đồng nghĩa với việc số lượng noron của mạng lớn, tốc độ xử lý chậm hơn và thời gian huấn luyện mạng lâu hơn.

3.2.2 Thu nhận hình ảnh (Capture Image)

Giai đoạn đầu tiên trong hệ thống phát hiện và nhận dạng biển báo là bước thu nhận ảnh. Như đã trình bày, dù là chụp hình hay quay phim thì thực chất đối tượng ta thu được từ camera chính là frame hình. Hình ảnh này là dữ liệu đã được số hóa. Có thể do một số yếu tố ngoại cảnh dẫn tới hình ảnh thu được bị mờ hoặc nhiễu, vì vậy trong bước này đòi hỏi cần thực hiện một số thao tác nâng cao chất lượng ảnh.

Các thuật toán có thể áp dụng để nâng cao chất lượng ảnh bao gồm như làm mịn ảnh bằng thuật toán Smooth Gaussian, giảm mẫu (downsample) và tăng mẫu (upsample) bằng Gaussian Pyramid … Việc cài đặt các giải thuật này sẽ được thư viện xử lý ảnh OpenCV hỗ trợ.

3.2.3 Phát hiện biển báo và trích xuất vùng đặc trưng

Mục đích của giai đoạn này là tìm ra có hay không khu vực có đặc trưng giống như mô tả của một biển báo trên nền khung cảnh. Sau đó từ dữ liệu này sẽ lọc bỏ toàn bộ hình nền, chỉ trích xuất lại hình ảnh của mỗi biển báo.

3.2.3.1 Đặc trưng phát hiện biển báo

Để phát hiện biển báo chúng ta dựa trên đặc trưng của biển báo giao thông Việt Nam

- **Biển báo cấm**: thường có hình dạng tròn, viền đỏ; cá biệt cũng có một số ngoại lệ nhưng không đáng kể

Hình 3.6 – Mẫu biển báo cấm

- **Biển báo nguy hiểm**: có dạng hình tam giác, viền đỏ, nền vàng; cá biệt cũng có một số ngoại lệ nhưng không đáng kể

Hình 3.7 – Mẫu biển nguy hiểm

- **Biển hiệu lệnh**: có dạng hình tròn, nền xanh; cá biệt cũng có một số ngoại lệ nhưng không đáng kể

Hình 3.8 – Mẫu biển hiệu lệnh

Tóm lại đặc trưng dùng để nhận dạng biển báo là đối tượng có viền bao màu đỏ hoặc màu xanh, dạng hình học là hình tròn (hoặc eclipse nếu lệch góc nhìn), hình tam giác. Ngoài ra những biển báo hướng dẫn dạng hình chữ nhật, hay các biển báo có tính cá biệt quá cao sẽ không được nghiên cứu trong khóa luận này.

Hình 3.9 – Mẫu một số biển báo quá cá biệt

3.2.3.2 Phương pháp phát hiện biển báo

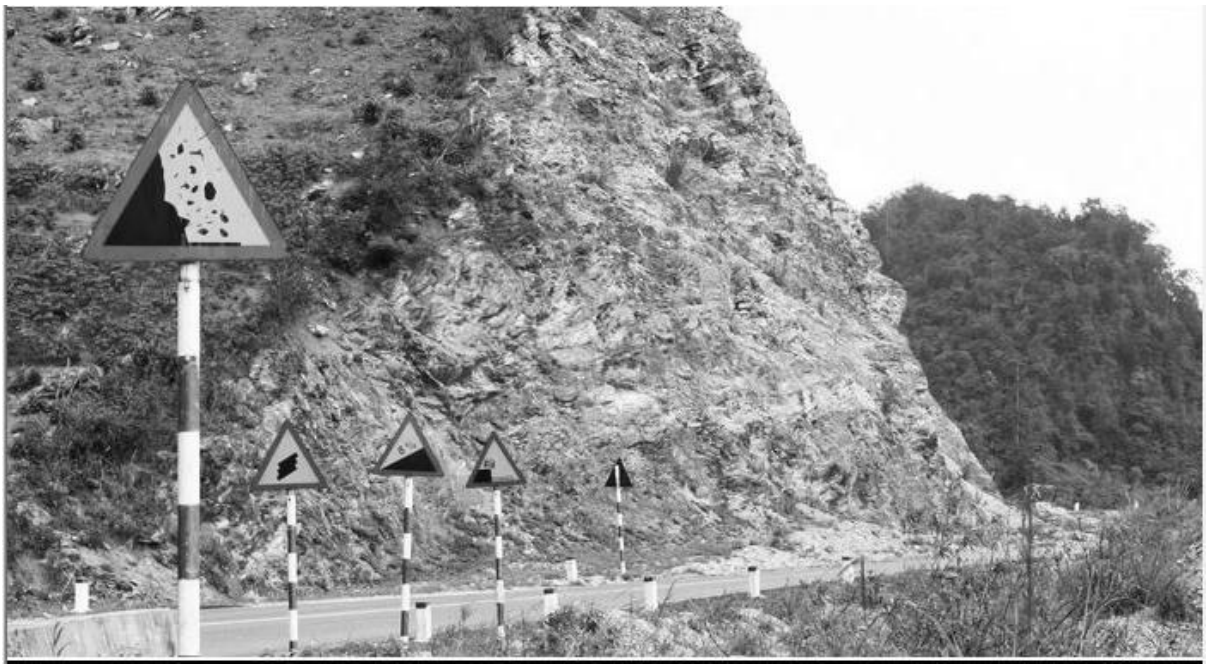
Dựa trên các đặc trưng của biển báo, nhóm sử dụng phương pháp phát hiện biên Canny kết hợp với đặc trưng màu để tìm ra biên ảnh của biển báo, sau đó dùng đặc trưng nhận dạng hình học để giữ lại chính xác các biên ảnh đúng, loại bỏ các biên giả.

3.2.3.2.1 Phát hiện biên ảnh bằng Canny:

Chi tiết thuật toán Canny đã được trình bày trong phần lý thuyết xử lý ảnh. Chúng ta có thể hiểu đơn giản về phương pháp Canny như sau:

- Phương pháp canny sử dụng một ngưỡng màu (threshold) nhất định cho việc phát hiện biên.
- Biên ảnh được hiểu là vùng ảnh mà ở đó có sự chênh lệch cường độ màu rõ rệt.
- Các ảnh áp dụng thuật toán Canny đều là ảnh được chuyển sang mức xám (Gray scale) với cường độ từ 0 - 255

Hình 3.10 – Ảnh ban đầu thu từ camera



Hình 3.11 – Ảnh sau khi đã chuyển sang ảnh mức xám

Hình 3.12 – Ảnh sau khi dùng Canny để tìm biên

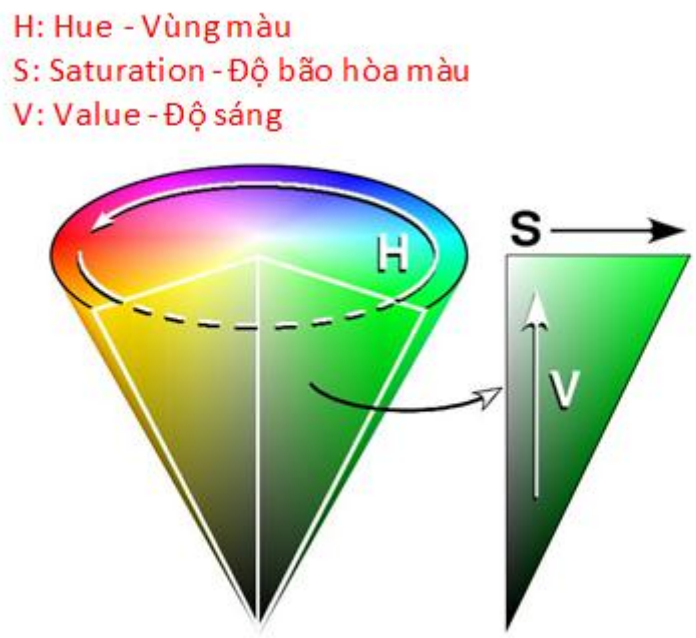
Nhận xét: Khi chỉ sử dụng phương pháp Canny thông thường thì kết quả thu được chứa rất nhiều "nhiễu", sẽ gây khó khăn cho việc tìm ra biên ảnh của biển báo sau này nếu chị dựa vào đặc trưng hình học của biển báo (hình tròn hay hình tam giác).

3.2.3.2.2 Kết hợp đặc trưng màu với phương pháp Canny:

Nhận thấy 3 loại biển báo cần phát hiện đều có màu đặc trưng là viền đỏ hoặc xanh nước biển, vì thế ta có thể lợi dụng điểm này để loại bớt các chi tiết nhiễu trên hình.

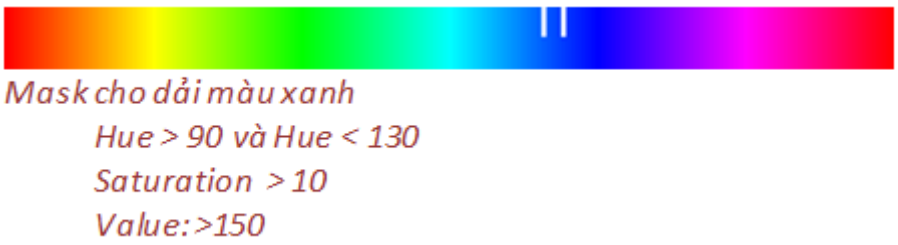
Ý tưởng của phương pháp này như sau:

- Ta xây dựng một mặt nạ (mask) màu đỏ (hoặc xanh) có kích thước bằng ảnh gốc bằng cách loại bỏ tất cả các điểm ảnh không thuộc dải màu đỏ (hoặc xanh) khi chuyển ảnh gốc thành ảnh mức xám.
- Để lập mặt nạ màu, chúng ta chuyển ảnh về xử lý trên không gian màu HSV



Hình 3.13 – Không gian màu HSV

Hình 3.14 – Khoảng giá trị ứng với dải màu đỏ

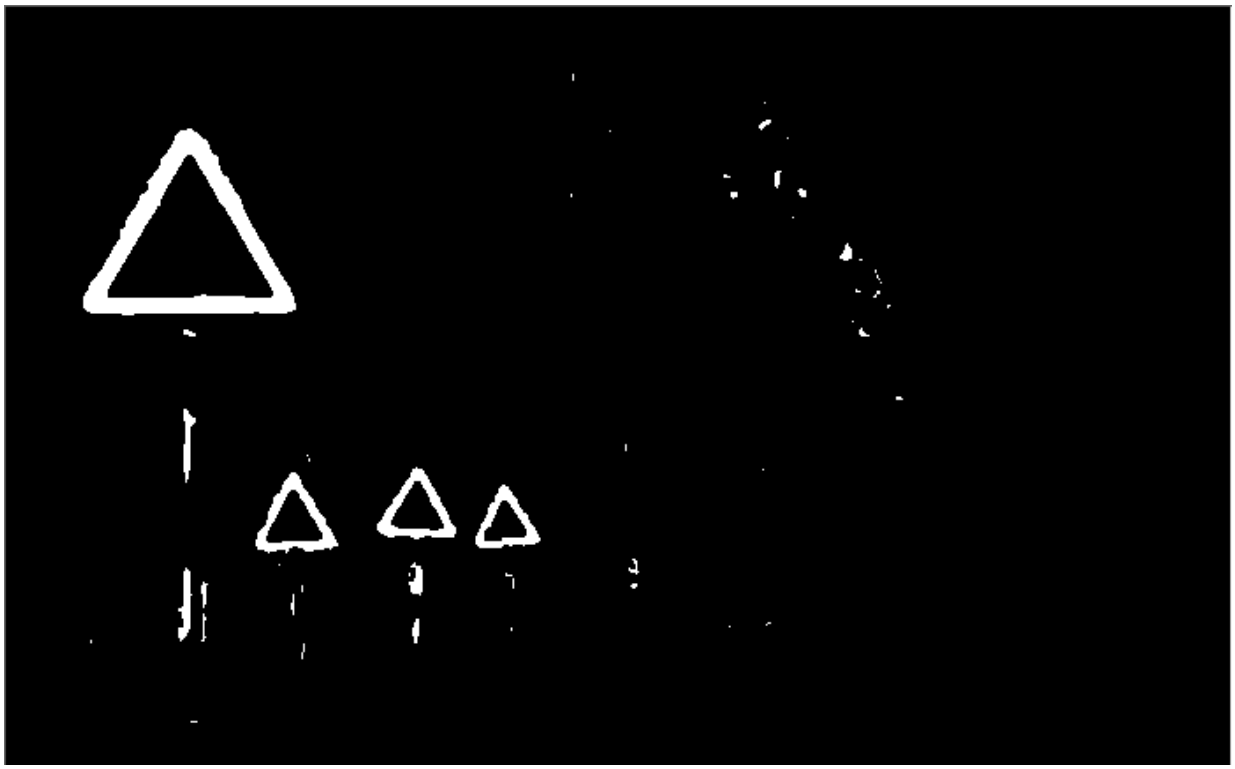


Hình 3.15 – Khoảng giá trị ứng với dải màu xanh

- Với mỗi giá trị H, S, V ta lập một mask tương ứng dựa theo dải màu xanh hay đỏ. Duyệt qua từng pixel của ảnh. Pixel nào nằm trong mask, ta giữ nguyên giá trị mức xám (gray scale)

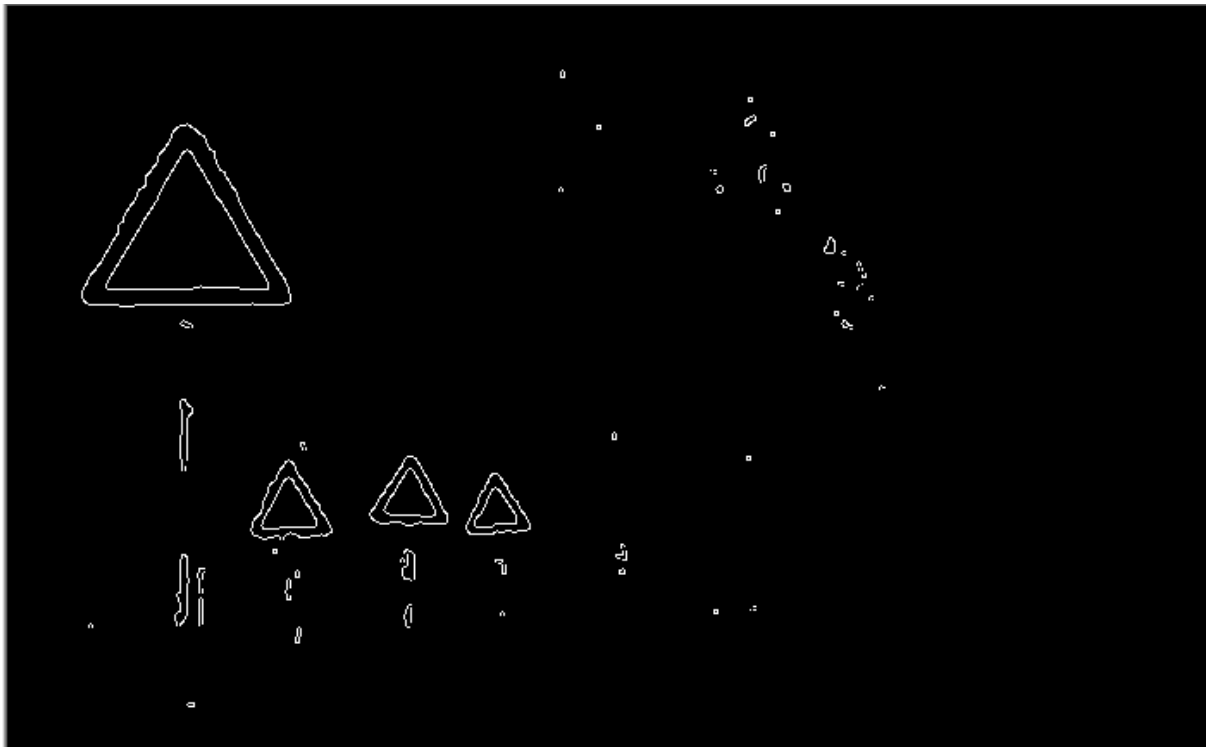
khi chuyển qua ảnh mức xám, ngược lại thì gán cho giá trị mức xám là 0.

- Khi dùng phép toán And dành cho 3 mặt nạ của H, S, V trên thì kết quả thu được chính là ảnh mức xám của ảnh gốc nhưng đã được lọc lại chỉ còn những điểm ảnh thuộc miền giá tri màu đỏ hoặc xanh ban đầu.



Hình 3.16 – Ảnh sau khi đã lọc qua mặt nạ màu

- Việc lập khoảng giá trị cho H, S, V khi làm mặt nạ có ý nghĩa quan trọng tới độ chính xác khi xử lý sau này. Rõ ràng là khi khoảng giá trị càng hẹp thì nhiễu càng ít nhưng có nguy cơ bị mất hình cao hơn, còn khoảng giá trị rộng thì độ ăn mòn giảm nhưng số nhiễu còn lại vẫn đáng kể.
- Khi có kết quả là ảnh mức xám đã lọc qua mặt nạ thì ta dùng phương pháp Canny để tìm lại biên của ảnh.



Hình 3.17 – Ảnh sau khi dùng Canny phát hiện biên

Nhận xét: Khi kết hợp dùng mặt nạ màu để lọc điểm ảnh sau đó mới áp dụng phương pháp Canny để tìm biên, ta thấy rõ ráng kết quả được cải thiện hơn rất nhiều, các biên tìm được có độ chính xác cao hơn và nhiễu ít hơn so với nếu chỉ dùng Canny đơn thuần.

3.2.3.2.3 Dùng đặc trưng hình học để loại bỏ biên giả:

Sau khi có ảnh biên thu được từ việc áp dụng phương pháp Canny, chúng ta sẽ tiến hành loại bỏ các biên giả.

Trong xử lý ảnh, có một thuật ngữ gọi là Contour (đường viền), có thể hiểu tạm hiểu là đường viền bao quanh khép kín.

Để loại bỏ các biên giả, ta tiến hành duyệt qua lần lượt tất cả các Contour thu được từ phương pháp Canny (hàm FindContour đã được OpenCV hỗ trợ). Với mỗi Contour này ta kiểm tra đặc trưng hình học của nó để xác định xem có phải là biên của biển báo hay ko?

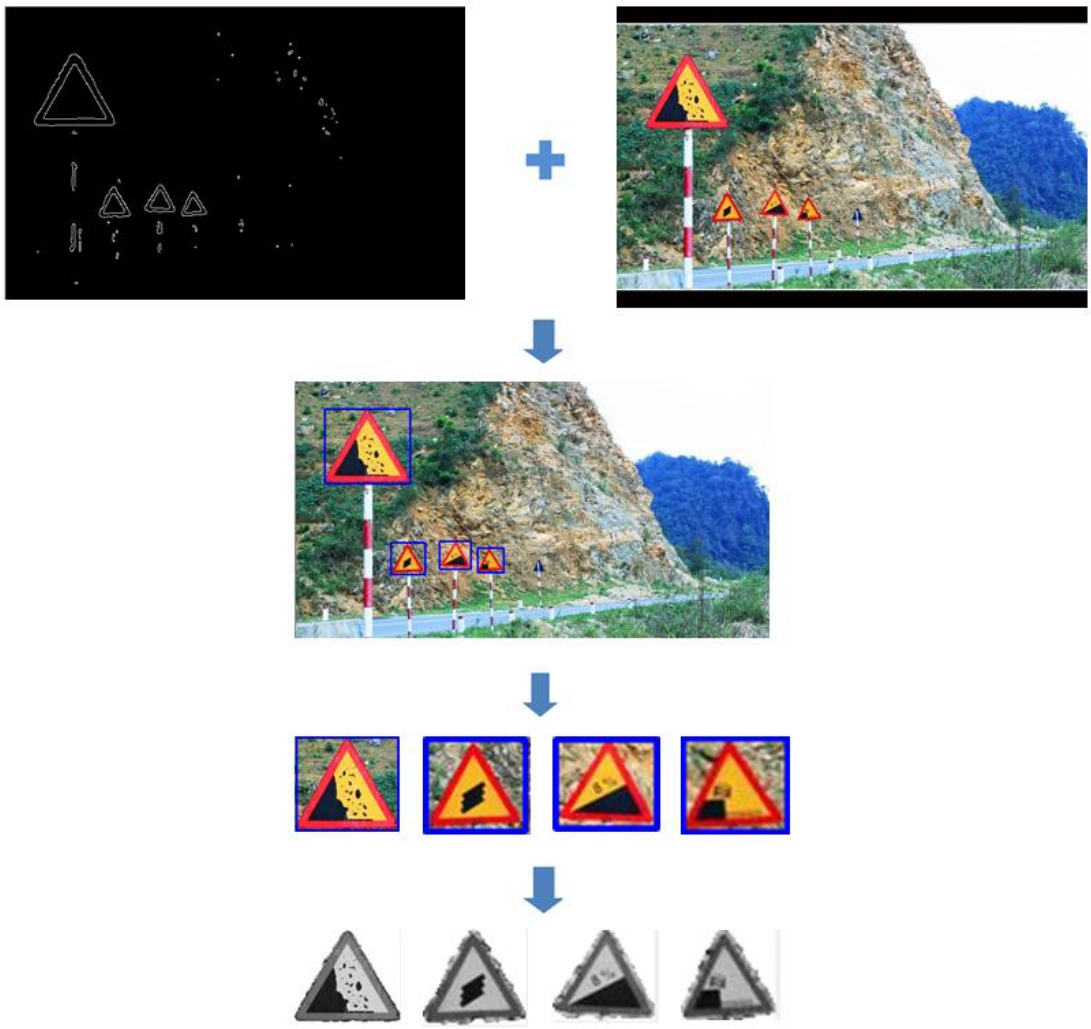
- Tính chất tam giác: Kiểm tra xem contour có phải được tạo thành từ 3 vector hay không ?

- Tính chất hình tròn (hoặc eclipse): Tính toán bán kính trục lớn, trục nhỏ, diện tích của Contour, từ đó suy ra giá trị PI. Nếu Pi ~3.14 thì đó thỏa mãn tính chất hình tròn.

Kết thúc quá trình duyệt, ta loại bỏ tất cả các Contour không thỏa mãn đặc trưng hình học, những Contour còn lại chính là biên của các biển báo.

3.2.3.3 Trích xuất vùng đặc trưng (ROI Extraction)

Sau khi có được các biên của các biển báo, ta xác định các hình chữ nhật bao đóng ngoại tiếp các biên này. Trích xuất các khu vực tương ứng với các hình chữ nhật này trên ảnh gốc, loại bỏ tất cả các điểm màu nằm ngoài biên của biển báo, sau đó chuyển ảnh màu sang ảnh mức xám.



Hình 3.18 – Kết quả của ROI Extraction

3.2.4 Xử lý trước khi nhận dạng (Pre-recognized)

Đây là quá trình chúng ta biến đổi dữ liệu ảnh thu ở bước ROI extraction thành một kiểu dữ liệu hợp lý, làm input đầu vào cho mạng nơron.

- Sau khi có dữ liệu ảnh, chúng ta thay đổi kích thước ảnh về cùng kích thước mẫu 30*30 pixel
- Lưu ý rằng khi ảnh còn ở dạng ảnh màu thì mỗi pixel chứa thông tin của 3 màu R, G, B.
- Ta có công thức chuyển ảnh RGB thành ảnh mức xám (Gray scale) như sau:

Gray = (0.299*R + 0.587*G + 0.114*B)

- Dữ liệu chúng ta sẽ cung cấp cho input của mạng noron ở giai đoạn sau sẽ là một dãy 63 tham số, đại diện cho dữ liệu của bức ảnh, trong đó:
  - *3 tham số đại diện cho giá trị trung bình của ba màu R, G, B của bức hình*

$$MR = \frac{1}{256}\left(\frac{1}{900}\sum_{i=1}^{30}\sum_{j=1}^{30} b_{i,j}\right)$$

$$MG = \frac{1}{256}\left(\frac{1}{900}\sum_{i=31}^{60}\sum_{j=1}^{30} b_{i,j}\right)$$

$$MB = \frac{1}{256}\left(\frac{1}{900}\sum_{i=61}^{90}\sum_{j=1}^{30} b_{i,j}\right)$$

$b_{i,j}$ đại diện cho giá trị màu ở pixel tại vị trí i*j

  - *30 tham số đại diện cho cường độ sáng của ảnh Gray theo chiều dọc*

$$\frac{1}{30}\sum_{j=1}^{30}(b'_{i,j} > T), \quad i = 1, 2, ..., 30$$

$b'_{i,j}$ đại diện cho giá trị cường độ xám tại vị trí i*j

- *30 tham số đại diện cho cường độ sáng của ảnh Gray theo chiều ngang*

$$\frac{1}{30}\sum_{i=1}^{30}(b'_{i,j} > T) \quad , \; j = 1,2,...,30$$

$b'_{i,j}$ đại diện cho giá trị cường độ xám tại vị trí i*j

- *Trong đó T là đại diện cho giá trị trung bình của cường độ sáng của ảnh mức xám*

$$T = \frac{1}{900}\sum_{i=1}^{30}\sum_{j=1}^{30} b'_{i,j}$$

Mảng 63 phần tử này sẽ được chuyển cho bước kế tiếp, cũng là giai đoạn cuối; quá trình nhận dạng. Về lý thuyết, khi chúng ta sử dụng ảnh có kích thước 30*30 pixel thì trường hợp lý tưởng nhất vẫn là sử dụng 30*30=900 giá trị làm input cho mạng noron, tuy nhiên điều này đòi hỏi một số lượng noron quá lớn, đi kèm với nó là cấu trúc mạng và khả năng huấn luyện thành công hay không ? Vì vậy, nhóm sử dụng một cấu trúc rút gọn bằng cách dùng 63 giá trị phần tử trên, đại diện cho thông tin của bức ảnh để mạng noron xử lý.

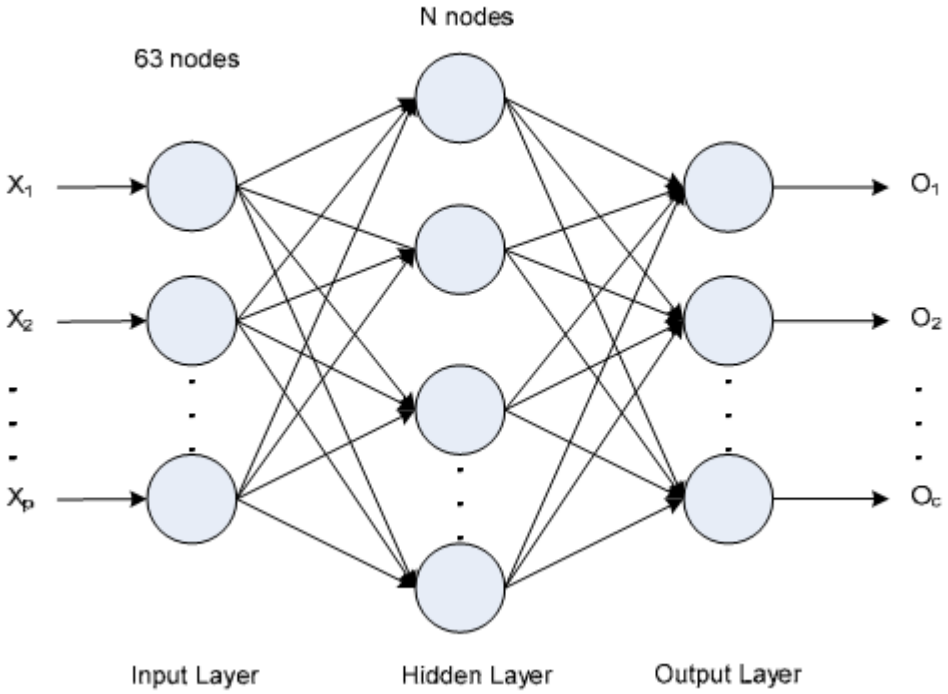
3.2.5 Quá trình nhận dạng (Recognized)

Đây là giai đoạn cuối cùng của hệ thống. Chúng ta cung cấp 1 tập dữ liệu đầu vào cho một mạng noron đã được huấn luyện từ trước. Mạng sẽ tính toán và trả kết quả output. Từ các kết quả này chúng ta có thể xác định mạng có thể nhận dạng ra biển báo đó hay không và nếu có thì biển báo đó là gì.

3.2.5.1 Cấu trúc mạng noron

- Mạng noron được nhóm lựa chọn ở đây là mạng truyền thẳng MLP (Multi layer Perception).
- Giải thuật huấn luyện mạng là giải thuật lan truyền ngược lỗi Back-propagation
- Hàm activation là hàm sigmoid.

- Mạng gồm có 3 layer:
    - **Input Layer**: Số noron là 63, đại diện cho 63 tham số thể hiện của bức ảnh
    - **Hidden layer**: lớp ẩn, lớp ở giữa của mạng. Số noron lớp này được xác định bằng thực nghiệm để đo hiệu năng của mạng
    - **Output Layer**: Số noron bằng chính lượng biển báo mà mạng có thể nhận diện, ví dụ: output có 5 noron thì mạng chỉ có thể nhận diện 5 biển báo.



Hình 3.19 – Cấu trúc mạng noron để nhận dạng biển báo

3.2.5.2 Tập huấn luyện

Để có thể sử dụng được mạng noron thì mạng này cần phải được huấn luyện. Do thời gian khóa luận ngắn và nhóm không đủ điều kiện để sưu tầm toàn bộ hệ thống mẫu biển báo Việt nam cho mạng noron học nên nhóm sẽ xây dựng một tập mẫu demo gồm 5 loại biển báo với tổng cộng 30 mẫu.

Hình 3.20 – Tập dữ liệu mẫu để huấn luyện mạng

Hình 3.21 – Tập biển báo chuẩn

3.3 Thiết Kế Chương Trình

3.3.1 Yêu cầu phần mềm

3.3.1.1 Yêu cầu chức năng

STT	Tên yêu cầu	Mô tả	Ghi chú
1	Phát hiện biển báo bằng tay	Người dùng sử dụng chế độ chụp hình để phát hiện biển báo	
2	Phát hiện biển báo tự động	Người dùng sử dụng chế độ quay video để ứng dụng tự phát hiện biển báo	
3	Nhận dạng biển báo	Ứng dụng nhận dạng các biển báo được phát hiện theo yêu cầu của người dùng	
4	Thay đổi chế độ phát hiện biển báo	Thay đổi giữa chế độ phát hiện bằng tay và phát hiện tự động	
5	Zoom hình ảnh	Thay đổi kích thước hình ảnh thu được qua camera	Chỉ hỗ trợ trong chế độ phát hiện biển báo bằng tay

Bảng3.1 – Danh sách yêu cầu chức năng

3.3.1.2 Yêu cầu hiệu quả

Áp dụng với thiết bị di động :

- Cài đặt hệ điều hành Android 2.2 trở lên.
- Chíp ARM tốc độ 600MHZ trở lên.
- Bộ nhớ 512MB trở lên
- Màn hình độ phân giải 3.2 trở lên
- Camera 5 Megapixel trở lên

STT	Chức năng	Tốc độ xử lý	Ghi chú
1	Phát hiện biển báo bằng tay	Ngay lập tức	
2	Phát hiện biển báo tự động	10Fps (Frame/giây)	
3	Nhận dạng biển báo	Ngay lập tức	
4	Thay đổi chế độ phát hiện biển báo	Ngay lập tức	
5	Zoom hình ảnh	3Fps (Frame/giây)	Tốc độ Zoom chủ yếu phụ thuộc vào khả năng hỗ trợ của phần cứng điện thoại

Bảng3.2 – Danh sách yêu cầu hiệu quả

3.3.1.3 Yêu cầu tương thích

Ứng dụng chạy ổn định trên các thiết bị :

- Cài đặt hệ điều hành Android 2.2 trở lên.
- Chíp ARM tốc độ 600MHZ trở lên.
- Bộ nhớ 512MB trở lên
- Màn hình độ phân giải 3.2 trở lên
- Camera 5 Megapixel trở lên

3.3.1.4 Yêu cầu tiện dụng

STT	Chức năng	Mức độ dễ sử dụng	Ghi chú
1	Phát hiện biển báo bằng tay	Thao tác cảm ứng trên màn hình	
2	Phát hiện biển báo tự động	Không cần thao tác	
3	Nhận dạng biển báo	Thao tác cảm ứng trên màn hình	
4	Thay đổi chế độ phát hiện biển báo	Thao tác cảm ứng trên màn hình	
5	Zoom hình ảnh	Thao tác cảm ứng trên màn hình	Tùy thuộc phần cứng hỗ trợ hay không

Bảng3.3 – Danh sách yêu cầu tiện dụng

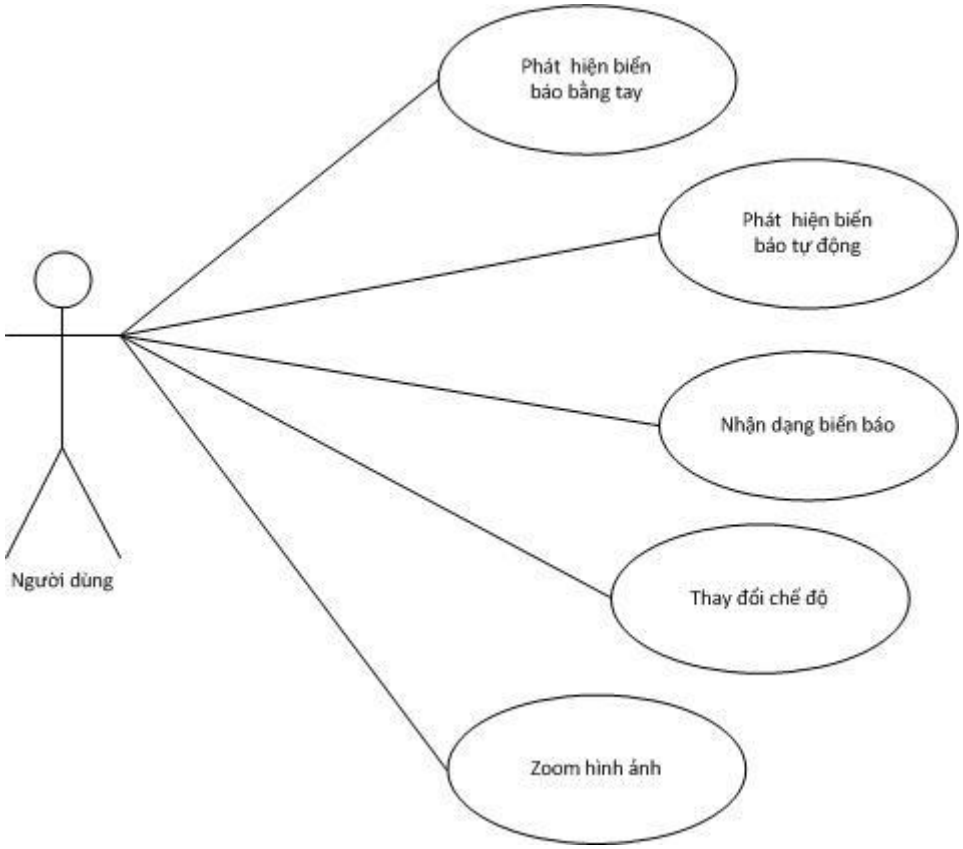
3.3.1.5 Yêu cầu tiến hóa

STT	Chức năng	Giá trị cần thay đổi	Ghi chú
1	Lưu trữ và phát hiện biển báo mới	Ma trận trọng số của mạng noron	Để nhận biết biển báo mới, dữ liệu cần cho mạng noron học để điều chỉnh lại ma trận trọng số cho thích hợp

Bảng3.4 – Danh sách yêu cầu tiến hóa

3.3.2 Thiết kế Use-Case

3.3.2.1 Sơ đồ Use-case tổng quát



Hình 3.20 – Sơ đồ Use-case tổng quát

3.3.2.2 Danh sách các Actor

STT	Tên Actor	Ý nghĩa/Ghi chú
1	Người dùng	Người trực tiếp tương tác với ứng dụng, có nhu cầu tra cứu thông tin về biển báo

Bảng3.5 – Danh sách các Actor

3.3.2.3 Danh sách các Use-Case

STT	Tên Use-Case	Ý nghĩa/Ghi chú
1	Phát hiện biển báo bằng tay	Người dùng tự điều chỉnh camera, lựa chọn khu vực hình ảnh có biển báo giao thông sau đó chụp hình để ứng

		dụng có thể phát hiện biển báo
2	Phát hiện biển báo tự động	Người dùng không cần phải chụp hình mà chỉ cần dùng camera quay hình khung cảnh, ứng dụng tự động phát hiện khu vực có biển báo giao thông
3	Nhận dạng biển báo	Sau khi phát hiện biển báo, nếu người dùng yêu cầu, ứng dụng sẽ nhận dạng biển báo được người dùng lựa chọn
4	Thay đổi chế độ	Người dùng lựa chọn thay đổi chế độ giữa phát hiện bằng tay và phát hiện tự động
5	Zoom hình ảnh	Cho phép người dùng thay đổi kích thước hình ảnh thu được qua camera

Bảng3.6 – Danh sách các Use-Case

3.3.2.4 Đặc tả Use-Case

3.3.2.4.1 Đặc tả Use case "Phát hiện biển báo bằng tay":

a. Tóm tắt

- Use case cho phép người dùng có thể tự phát hiện biển báo giao thông bằng tay

b. Dòng sự kiện

- Dòng sự kiện chính:
  - Dùng camera, lựa chọn khung cảnh muốn phát hiện biển báo.
  - Click nút chụp hình.
- Dòng sự kiện khác:
  - Không có.

c. Các yêu cầu đặc biệt

- Không có.

d. Trạng thái hệ thống khi bắt đầu thực hiện Use-case

- Ứng dụng đang ở chế độ phát hiện biển báo bằng tay.

e. Trạng thái hệ thống sau khi thực hiện Use-case

- Ứng dụng thông báo kết quả của việc phát hiện biển báo
- Các khu vực biển báo (nếu có) sẽ được khoanh vùng

f. Điểm mở rộng

- Không có.

3.3.2.4.2 Đặc tả Use case "Phát hiện biển báo tự động":

a. Tóm tắt

- Use case cho phép người dùng có thể tự phát hiện biển báo giao thông một cách tự động.

b. Dòng sự kiện

- Dòng sự kiện chính:
  - Dùng camera, quét qua khung cảnh chứa biển báo giao thông.
  - Ứng dụng tự động phát hiện khu vực có biển báo.
- Dòng sự kiện khác:
  - Không có.

c. Các yêu cầu đặc biệt

- Không có.

d. Trạng thái hệ thống khi bắt đầu thực hiện Use-case

- Ứng dụng đang ở chế độ phát hiện biển báo tự động.

e. Trạng thái hệ thống sau khi thực hiện Use-case

- Ứng dụng tự động khoanh vùng khu vực có biển báo nếu có.

f. Điểm mở rộng

- Không có.

3.3.2.4.3 Đặc tả Use case "Nhận dạng biển báo":

a. Tóm tắt

- Use case cho phép người dùng nhận dạng biển báo đã được phát hiện.

b. Dòng sự kiện

- Dòng sự kiện chính:
  - Lựa chọn khu vực có biển báo đã được phát hiện.
  - Ứng dụng tự động nhận dạng biển báo và thông báo kết quả.
- Dòng sự kiện khác:
  - Thông báo trường hợp biển báo không có trong cơ sở dữ liệu mẫu của ứng dụng.

c. Các yêu cầu đặc biệt

- Không có.

d. Trạng thái hệ thống khi bắt đầu thực hiện Use-case

- Hệ thống đã phát hiện khu vực biển báo và đang sẵn sàng cho việc nhận dạng.

e. Trạng thái hệ thống sau khi thực hiện Use-case

- Sẵn sàng cho việc nhận dạng biển báo khác.
- Sẵn sàng cho việc quay trở lại chế độ phát hiện biển báo.

f. Điểm mở rộng

- Không có.

3.3.2.4.4 Đặc tả Use case "Thay đổi chế độ":

a. Tóm tắt

- Use case cho phép người dùng thay đổi chế độ phát hiện biển báo.

b. Dòng sự kiện

- Dòng sự kiện chính:
  - Kích hoạt menu lựa chọn chế độ.

- Chọn lựa chế độ phát hiện biển báo mà người dùng muốn.

- Dòng sự kiện khác:
  - Không có.

c. Các yêu cầu đặc biệt

- Không có.

d. Trạng thái hệ thống khi bắt đầu thực hiện Use-case

- Hệ thống không đang ở chế độ nhận dạng.

e. Trạng thái hệ thống sau khi thực hiện Use-case

- Hệ thống ở chế độ phát hiện biển báo mà người dùng lựa chọn và sẵn sàng làm việc.

f. Điểm mở rộng

- Không có.

3.3.2.4.5 Đặc tả Use case "Zoom hình ảnh":

a. Tóm tắt

- Use case cho phép người dùng thay đổi kích thước hình ảnh thu được qua camera.

b. Dòng sự kiện

- Dòng sự kiện chính:
  - Thay đổi kích thước hình bằng việc tương tác với thanh kéo trên màn hình.
- Dòng sự kiện khác:
  - Hình ảnh không thay đổi kích thước nếu camera không hỗ trợ zoom.

c. Các yêu cầu đặc biệt

- Thiết bị phần cứng hỗ trợ zoom.

d. Trạng thái hệ thống khi bắt đầu thực hiện Use-case

- Hệ thống đang ở chế độ phát hiện biển báo giao thông bằng tay.

e. Trạng thái hệ thống sau khi thực hiện Use-case

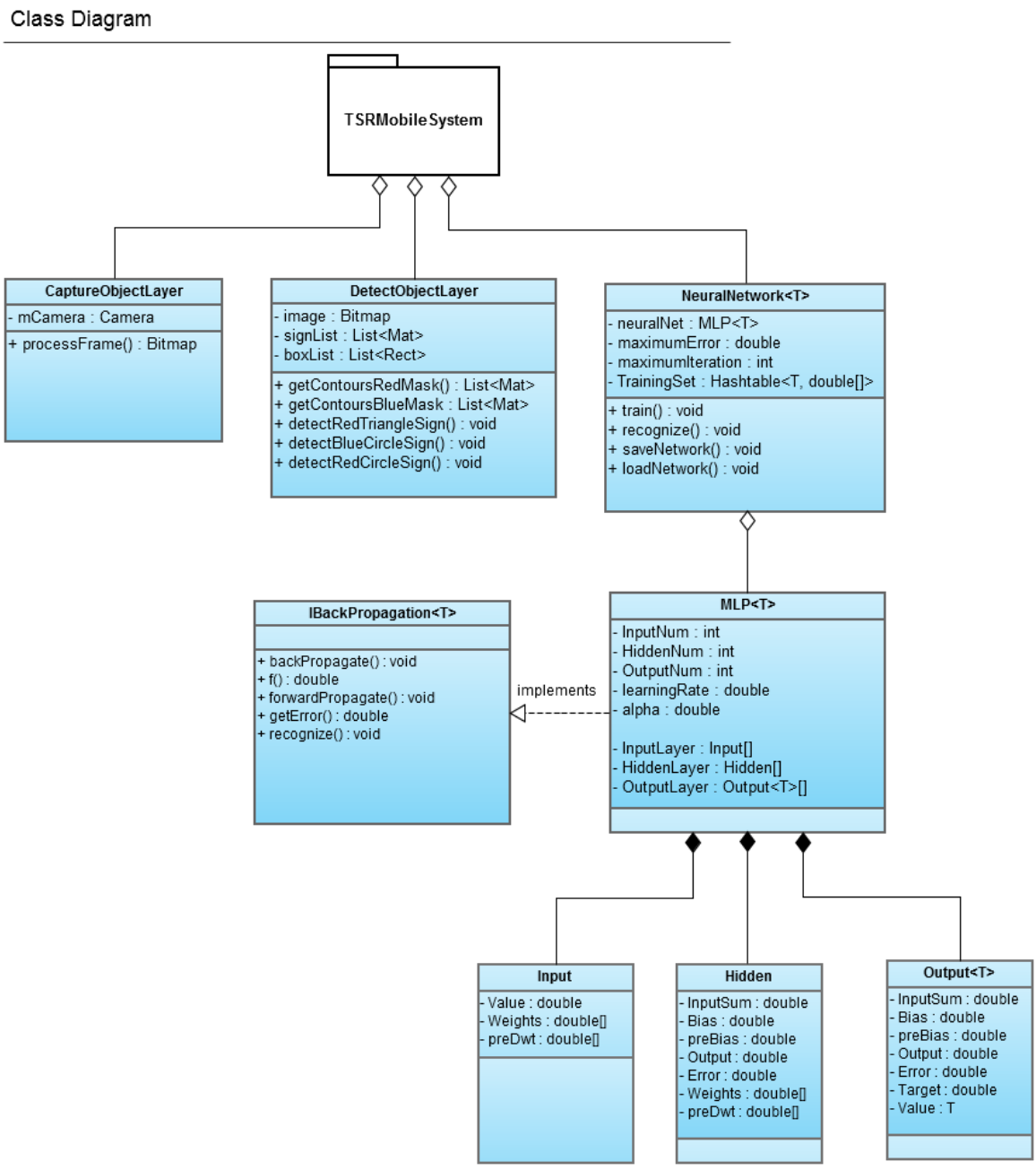
- Hình ảnh thu được có kích thước mong muốn theo nhu cầu của người dùng.

f. Điểm mở rộng

- Không có.

3.3.3 Thiết kế sơ đồ lớp (mức phân tích)

3.3.3.1 Sơ đồ lớp mức phân tích



Hình 3.21 – Sơ đồ lớp mức phân tích

3.3.3.2 Danh sách các lớp đối tượng và quan hệ

STT	Tên lớp/quan hệ	Loại	Ý nghĩa/Ghi chú
1	Input	Public	Lớp Input của mạng noron
2	Hidden	Public	Lớp Hidden của mạng noron
3	Output	Public	Lớp Output của mạng noron
4	IbackPropagation<T>	Public	Interface bao đóng các chức năng của mạng noron
5	MLP<T>	Public	Mạng noron truyền thẳng nhiều lớp MLP
6	CaptureObjectLayer	Public	Chịu trách nhiệm thu lấy dữ liệu từ camera của thiết bị
7	DetectObjectLayer	Public	Đối tượng dùng để phát hiện biển báo
8	NeuralNetwork<T>	Public	Mạng noron dùng để nhận dạng biển báo

Bảng3.7 – Danh sách các lớp đối tượng quan hệ

3.3.3.3 Mô tả chi tiết từng lớp đối tượng

3.3.3.3.1 Lớp Input:

STT	Tên thuộc tính	Kiểu	Truy cập	Ý nghĩa/Ghi chú
1	Value	double	Public	Giá trị output của noron
2	Weights	double[]	Public	Mảng chứa trọng số ứng với noron này
3	preDwt	double[]	Public	Mảng chứa trọng số trước khi sửa lỗi

Bảng3.8 – Danh sách thuộc tính lớp Input

3.3.3.3.2 Lớp Hidden:

STT	Tên thuộc tính	Kiểu	Truy cập	Ý nghĩa/Ghi chú
1	InputSum	double	Public	Tổng trọng số
2	Bias	double	Public	Độ lệch hiện tại của noron

3	preBias	double	Public	Độ lệch của noron trước khi sửa lỗi
4	Output	double	Public	Giá trị đầu ra của noron
5	Error	double	Public	Giá trị lỗi
6	Weights	double[]	Public	Mảng chứa trọng số ứng với noron này
7	preDwt	double[]	Public	Mảng chứa trọng số trước khi sửa lỗi

Bảng3.9 – Danh sách thuộc tính lớp Hidden

3.3.3.3.3 Lớp Output:

STT	Tên thuộc tính	Kiểu	Truy cập	Ý nghĩa/Ghi chú
1	InputSum	double	Public	Tổng trọng số
2	Bias	double	Public	Độ lệch hiện tại của noron
3	preBias	double	Public	Độ lệch của noron trước khi sửa lỗi
4	Output	double	Public	Giá trị đầu ra của noron
5	Error	double	Public	Giá trị lỗi
6	Target	double	Public	Giá trị đầu ra mong muốn
7	Value	T	Public	Kiểu trị kiểu T tương ứng với đầu ra

Bảng3.10 – Danh sách thuộc tính lớp Output

3.3.3.3.4 Lớp IbackPropagation<T>:

STT	Tên phương thức	Kiểu	Truy cập	Ý nghĩa/Ghi chú
1	backPropagate	void	Public	Lan truyền ngược lỗi và hiệu chỉnh trọng số trong mạng
2	f	double	Public	Hàm tính giá trị kích hoạt cho noron

3	forwardPropagate	void	Public	Lan truyền tính toán trong mạng
4	recognize	void	Public	Tiến hành nhận dạng một mẫu
5	initializeNetwork	void	Public	Khởi tạo mạng noron

Bảng3.11 – Danh sách phương thức lớp IbackPropagation<T>

3.3.3.3.5 Lớp MLP<T>:

STT	Tên thuộc tính	Kiểu	Truy cập	Ý nghĩa/Ghi chú
1	InputNum	int	Private	Số noron lớp Input
2	HiddenNum	int	Private	Số noron lớp Hidden
3	OutputNum	int	Private	Số noron lớp Output
4	learningRate	double	Private	Tỷ lệ thay đổi trọng số
5	alpha	double	Private	Thông số alpha của mạng noron
6	OutputValueHight	double	Private	Giá trị tính toán được gần đúng với giá trị mong muốn nhất
7	MatchedHigh	T	Private	Giá trị kiểu T tương ứng với OutputValueHight
8	OutputValueLow	double	Private	Giá trị tính toán được gần đúng xếp thứ 2 so với giá trị mong muốn nhất
9	MatchedLow	T	Private	Giá trị kiểu T tương ứng với OutputValueLow
10	InputLayer	Input[]	Private	Mảng chứa các noron lớp Input
11	HiddenLayer	Hidden[]	Private	Mảng chứa các noron lớp Hidden

12	OutputLayer	Output<T>[]	Private	Mảng chứa các noron lớp Output

Bảng3.12 – Danh sách thuộc tính lớp MLP<T>

STT	Tên phương thức	Kiểu	Truy cập	Ý nghĩa/Ghi chú
1	reset	void	Public	Khởi tạo lại các tham số của mạng
2	getMatchedHigh	T	Public	Lấy kết quả kiểu T gần chính xác nhất
3	getOutputValueHight	double	Public	Lấy kết quả gần chính xác nhất
4	getMatchedLow	T	Public	Lấy kết quả kiểu T gần chính xác thứ hai
5	getOutputValueLow	double	Public	Lấy kết quả gần chính xác thứ hai

Bảng3.13 – Danh sách phương thức lớp MLP<T>

3.3.3.3.6 Lớp CaptureObjectLayer:

STT	Tên thuộc tính	Kiểu	Truy cập	Ý nghĩa/Ghi chú
1	mCamera	Camera	Private	Camera của thiết bị
2	mFrameWidth	int	Private	Độ rộng của Frame hình
3	mFrameHeight	int	Private	Độ cao của Frame hình
4	mFrame	Byte[]	Private	Dữ liệu thu từ camera

Bảng3.14 – Danh sách thuộc tính lớp CaptureObjectLayer

STT	Tên phương thức	Kiểu	Truy cập	Ý nghĩa/Ghi chú
1	processFrame	Bitmap	Public	Xử lý ảnh thu được từ camera
2	run	Void	Public	Khởi chạy luồng phụ song song với luồng chính, liên

				tục lấy dữ liệu từ camera
3	getFrameWidth	int	Public	Lấy độ rộng của Frame hình
4	getFrameHeight	int	Public	Lấy độ cao của Frame hình

Bảng3.15 – Danh sách phương thức lớp CaptureObjectLayer

3.3.3.3.7 Lớp DetectObjectLayer:

STT	Tên thuộc tính	Kiểu	Truy cập	Ý nghĩa/Ghi chú
1	image	Bitmap	Private	Chứa hình thu từ camera để xử lý
2	signList	List<Mat>	Private	Danh sách các biển báo tìm được
3	boxList	List<Rect>	Private	Danh sách các viền bao chữ nhật của biển báo

Bảng3.16 – Danh sách thuộc tính lớp DetectObjectLayer

STT	Tên phương thức	Kiểu	Truy cập	Ý nghĩa/Ghi chú
1	getContoursRedMask	List<Mat>	Public	Lấy đường biên sau khi lọc mặt nạ màu đỏ
2	getContoursBlueMask	List<Mat>	Public	Lấy đường biên sau khi lọc mặt nạ màu xanh
3	detectRedTriangleSign	void	Public	Tìm các biển báo tam giác đỏ
4	detectBlueCircleSign	void	Public	Tìm các biển báo tròn xanh
5	detectRedCircleSign	void	Public	Tìm các biển báo tròn đỏ
6	getSignList	List<Mat>	Public	Lấy danh sách biển báo
7	getBoxList	List<Rect>	Public	Lấy sanh sách đường bao

				chữ nhật của biển báo

Bảng3.17 – Danh sách phương thức lớp DetectObjectLayer

3.3.3.3.8 Lớp NeuralNetwork<T>:

STT	Tên thuộc tính	Kiểu	Truy cập	Ý nghĩa/Ghi chú
1	neuralNet	MLP<T>	Private	Mạng MLP
2	maximumError	double	Private	Giới hạn tối đa của lỗi
3	maximumIteration	int	Private	Số vòng lặp huấn luyện mạng
4	TrainingSet	Hashtable<T, double[]>	Public	Tập dữ liệu huấn luyện mạng

Bảng3.18 – Danh sách thuộc tính lớp NeuralNetwork<T>

STT	Tên phương thức	Kiểu	Truy cập	Ý nghĩa/Ghi chú
1	train	void	Public	Huấn luyện mạng
2	recognize	Void	Public	Nhận dạng một đầu vào
3	saveNetwork	void	Public	Lưu cấu hình mạng
4	loadNetwork	void	Public	Load cấu hình mạng

Bảng3.19 – Danh sách phương thức lớp NeuralNetwork<T>

3.3.4 Thiết kế dữ liệu

Do tích chất của ứng dụng là phần mềm nghiên cứu về giải thuật phát hiện và nhận dạng biển báo giao thông nên cơ sở dữ liệu ứng dụng sử dụng còn có phần đơn giản.

Nhóm sử dụng một file XML để lưu cấu trúc mã, tên, nội dung và hình ảnh của biển báo giao thông.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<data>
    <sign>
        <id>00_101</id>
        <name>Biển báo 101</name>
        <title>Đường cấm</title>
        <content>
            Đường cấm tất cả các loại phương tiện (cơ giới và thô sơ)
            đi lại cả hai hướng, trừ các xe được ưu tiên theo luật lệ nhà nước quy định.
        </content>
        <image>sign_101</image>
    </sign>
    <sign>
        <id>00_102</id>
        <name>Biển báo 102</name>
        <title>Cấm đi ngược chiều</title>
        <content>
            Đường cấm tất cả các loại xe (cơ giới và thô sơ)
            đi vào theo chiều đặt biển, trừ các xe được ưu tiên theo luật lệ nhà nước quy định.
        </content>
        <image>sign_102</image>
    </sign>
    ...
    ...
    ...
</data>
```

Hình 3.22 – Cấu trúc file dữ liệu XML

3.3.5 Thiết kế giao diện

3.3.5.1 Danh sách các màn hình

STT	Tên màn hình	Ý nghĩa/Ghi chú
1	Màn hình chính	Giao diện chính, cho phép người dùng chọn lựa thay đổi chế độ phát hiện biển báo
2	Phát hiện biển báo bằng tay	Chế độ phát hiện biển báo bằng tay, cho phép người dụng tự lựa chọn khung cảnh và chụp hình
3	Phát hiện biển báo tự động	Chế độ tự động phát hiện biển báo khi quét camera qua khung cảnh.
4	Màn hình kết quả detect	Hiển thị kết quả phát hiện được, khoanh vùng các biển báo
5	Màn hình nhận dạng	Hiển thị kết quả nhận dạng biển báo theo nhu cầu

Bảng3.20 – Danh sách màn hình

3.3.5.2 Mô tả chi tiết các màn hình

3.3.5.2.1 Màn hình chính:

Hình 3.23 – Màn hình chính

STT	Tên	Loại	Ý Nghĩa	Kích hoạt	Ghi chú
1	cusZoomSlider	CustomSlider	Cho phép kéo trượt để zoom hình ảnh	Drag	
2	imgFocusImage	ImageView	Focus hình ảnh		
3	btnCaptureButton	ImageButton	Chụp hình	Click	
4	mItemManual	MenuItem	Chọn chế độ phát hiện bằng tay	Click	
5	mItemAuto	MenuItem	Chọn chế độ phát hiện tự động	Click	

Bảng3.21 – Chi tiết màn hình chính

[cusZoomSlider drag]

- Thay đổi hình ảnh theo tỷ lệ Zoom thích hợp

[btnCaptureButton click]

- Chụp hình sau đó thông báo kết quả:
  - Hiện thông báo nếu không có phát hiện biển báo
  - Chuyển đến màn hình kết quả detect nếu phát hiện biển báo

[mItemManual click]

- Chuyển sang chế độ phát hiện bằng tay

[mItemAuto click]

- Chuyển sang chế độ phát hiện tự động

3.3.5.2.2 Màn hình phát hiện biển báo bằng tay:

Hình 3.24 – Màn hình phát hiện biển báo bằng tay

STT	Tên	Loại	Ý Nghĩa	Kích hoạt	Ghi chú
1	cusZoomSlider	CustomSlider	Cho phép kéo trượt để zoom hình ảnh	Drag	
2	imgFocusImage	ImageView	Focus hình ảnh		
3	btnCaptureButton	ImageButton	Chụp hình	Click	

Bảng3.22 – Chi tiết màn hình phát hiện biển báo bằng tay

[cusZoomSlider drag]

- Thay đổi hình ảnh theo tỷ lệ Zoom thích hợp

[btnCaptureButton click]

- Chụp hình sau đó thông báo kết quả:
  - Hiện thông báo nếu không có phát hiện biển báo
  - Chuyển đến màn hình kết quả detect nếu phát hiện biển báo

3.3.5.2.3 Màn hình phát hiện biển báo tự động:

Hình 3.25 – Màn hình phát hiện biển báo tự động

STT	Tên	Loại	Ý Nghĩa	Kích hoạt	Ghi chú
1	tvFPS	TextView	Hiển thị số Frame/giây		
2	imgFocusImage	ImageView	Focus hình ảnh		

Bảng3.23 – Chi tiết màn hình phát hiện biển báo tự động

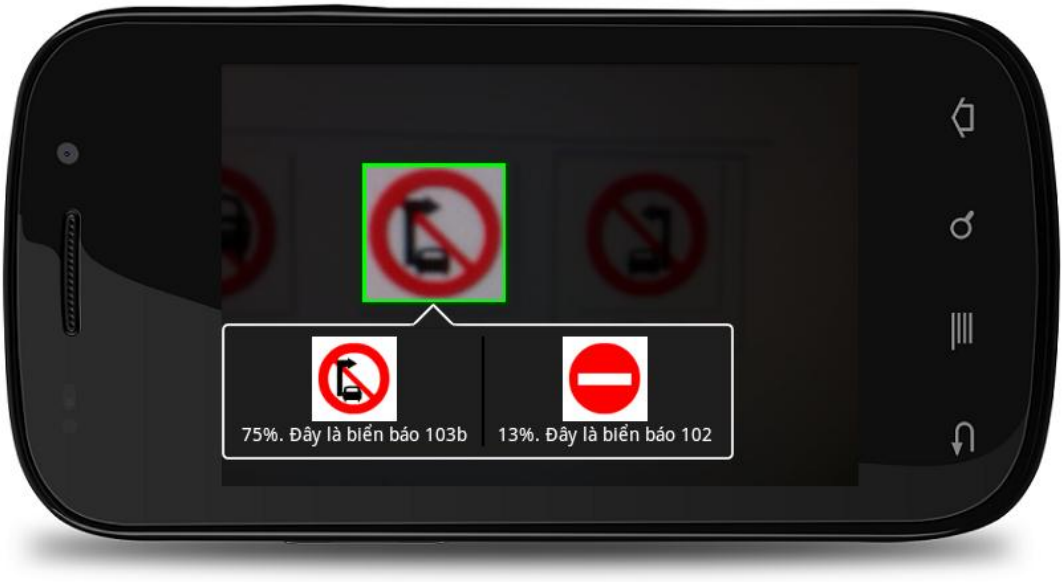
3.3.5.2.4 Màn hình kết quả detect:

Hình 3.26 – Màn hình kết quả detect

STT	Tên	Loại	Ý Nghĩa	Kích hoạt	Ghi chú
1	cam	FrameViewLayer	Hiện thị hình ảnh thu từ camera đã xử lý và khoanh vùng biển báo		

Bảng3.24 – Chi tiết màn hình kết quả detect

3.3.5.2.5 Màn hình kết quả nhận dạng:



Hình 3.27 – Màn hình kết quả nhận dạng

STT	Tên	Loại	Ý Nghĩa	Kích hoạt	Ghi chú
1	cam	FrameViewLayer	Hiện thị hình ảnh thu từ camera đã xử lý và khoanh vùng biển báo		
2	quickAction	PopupMenu	Hiện thị kết quả nhận dạng tương ứng với biển báo được chọn		

Bảng3.25 – Chi tiết màn hình kết quả nhận dạng

3.4 Thực Nghiệm

Như đã trình bày trong phần cấu trúc mạng noron, tập huấn luyện dành cho mạng sẽ là một tập 5 loại biển báo với tất cả 30 mẫu biến thể. Mục đích của việc thực nghiệm là nhằm xác định số lượng noron ở lớp ẩn một cách hợp lý, làm sao cho mạng có khả năng nhận biết cao nhất, không bỏ xót thông tin và thời gian huấn luyện mạng trong giới hạn cho phép.

Chúng ta sẽ sử dụng đại lượng trung bình bình phương lỗi tối thiểu và số vòng lặp huấn luyện mạng để đo lường quá trình thực nghiệm, lựa chọn số noron lớp ẩn hợp lý. Giới hạn trung bình bình phương tối thiểu sẽ là 0.0001 và số vòng lặp huấn luyện tối đa là 500.000 vòng.

Số noron lớp ẩn	Số lượng mẫu huấn luyện	Trung bình bình phương lỗi tối thiểu	Số vòng lặp huấn luyện	Kết quả huấn luyện
10	5	0.317	500000	Thất bại
20	5	0.240	500000	Thất bại
30	5	0.237	500000	Thất bại
40	5	0.153	500000	Thất bại
45	**5**	**0.0001**	**20882**	**Thành công**
45	10	0.161	500000	Thất bại
50	**10**	**0.0001**	**39388**	**Thành công**
50	**15**	**0.0001**	**59349**	**Thành công**
50	**20**	**0.0001**	**64497**	**Thành công**
50	**25**	**0.0001**	**111311**	**Thành công**
50	30	0.407	500000	Thất bại
60	30	0.329	500000	Thất bại
70	30	0.089	500000	Thất bại
80	**30**	**0.0001**	**47092**	**Thành công**

Bảng3.26 – Bảng kết quả thực nghiệm

Qua kết quả thực nghiệm, mạng noron sẽ được chọn số noron lớp ẩn là 80 và đại lượng ước lượng trung bình bình phương tối thiểu là nhỏ hơn 0.0001.

CHƯƠNG 4 : ĐÁNH GIÁ KẾT QUẢ VÀ KẾT LUẬN

Chương này trình bày các vấn đề sau:

4.1 Đánh giá luận văn

4.2 Đánh giá chương trình

4.3 Hướng phát triển

4.4 Kết luận

4.1 Đánh Giá Luận Văn

Báo cáo lý thuyết đã trình bày đầy đủ và cụ thể những điểm then chốt về xử lý ảnh, mạng noron cũng như sức mạnh của nó. Ngoài ra báo cáo cũng giới thiệu một nền tảng di động mới dành cho các nhà phát triển ứng dụng di động đó là hệ điều hành Android. Cùng với nó là thư viện xử lý ảnh OpenCV, một trong các thư viện thường hay được sử dụng khi làm việc với các bài toán nhận dạng.

Về mặt áp dụng, khóa luận đã giải quyết được bài toán phát hiện và nhận dạng biển báo giao thông ở Việt Nam. Mô hình giải quyết bài toán đơn giản, dễ cài đặt và thích hợp với các thị bị di động không cần đòi hỏi quá nhiều về phần cứng.

4.2 Đánh Giá Chương Trình

4.2.1 Kết quả đạt được

Chương trình tương đối hoàn chỉnh cho phép người dùng thực hiện các chức năng sau:

- Phát hiện biển báo trong khung cảnh bằng tay thông qua thao tác chụp hình
- Phát hiện biển báo tự động khi quay phim khung cảnh
- Nhận dạng biển báo phát hiện được dựa theo cơ sở dữ liệu có sẵn

4.2.2 Các hạn chế

- Chương trình hiện tại vẫn còn mang tính học thuật cao, chỉ được xem như một ứng dụng để tra cứu thông tin biển báo tức thời chứ chưa có

chức năng như một hệ thống nhận dạng và tự cảnh báo cho người tham gia giao thông để có thể xử lý kịp thời khi tham gia giao thông.

- Vì mang tính nghiên cứu nên hệ thống chỉ mới làm việc trên tập dữ liệu thử nghiệm với 5 loại biển báo khác nhau.
- Ứng dụng còn bị hạn chế khi làm việc với một số phần cứng không đáp ứng được các yêu cầu về xử lý hoặc chất lượng camera
- Ứng dụng chưa giải quyết triệt để bài toán về xử lý lỗi góc nhìn (perspective projection) khi chụp ảnh, bài toán làm việc với biển báo trong điều kiện môi trường phực tạp như ánh sáng yếu, che khuất…

4.3 Hướng Phát Triển

Nâng cấp khả năng hệ thống, trở thành một hệ thống nhận dạng và đưa ra cảnh báo tức thời cho người tham gia giao thông. Để làm việc này có thể phát triển ứng dụng theo hướng client – server, hoặc kết nối thiết bị với camera của các phương tiện ô tô, tiếp nhận dữ liệu từ camera của ô tô sau đó đưa ra cảnh báo cho người điều khiển.

Cải thiện khả năng làm việc, giải quyết triệt để các vấn đề còn mắc phải như phát hiện chưa chính xác trong điều kiện môi trường phực tạp, sửa lỗi góc nhìn khi quay phim, chụp hình…

4.4 Kết Luận

Luận văn xây dựng thành công hệ thống phát hiện và nhận dạng biển báo giao thông trên thiết bị di động, kết hợp được sức mạnh của công nghệ xử lý ảnh với nền tảng di động tiên tiến, hứa hẹn sẽ là hướng đi mới dành cho các ứng dụng smartphone trong tương lai.

TÀI LIỆU THAM KHẢO

[Tiếng Việt]

1. Giáo trình xử lý ảnh , TS Phạm Việt Bình – TS Đỗ Năng Toàn, Đại Học Thái Nguyên, 2007.
2. Giáo trình xử lý ảnh, tập thể tác giả, Học Viện Công Nghệ Bưu Chính Viễn Thông, lưu hành nội bộ, 2006.
3. Lý thuyết mạng noron, Nguyễn Thanh Cẩm.

[Tiếng Anh]

1. A Guided Tour of Computer Vision, Vishvjit S. Nalwa.
2. An introduction to neural networks: Pattern learning with the back-propagation algorithm - http://www.ibm.com/developerworks/library/l-neural/
3. Android technical resources http://developer.android.com/resources/browser.html?tag=tutorial
4. Color-Based Road Sign Detection and Tracking, Luis David Lopez and Olac Fuentes, Computer Science Department University of Texas
5. Learning OpenCV: Computer Vision with the OpenCV Library, Gary Bradski-Adrian Kaehler.
6. OpenCV Tutorials - http://opencv.itseez.com/trunk/doc/tutorials/tutorials.html
7. Principles of Artificial Neural Networks, Daniel Graupe.
8. Traffic Sign Recognition Using Neural network on OpenCV: Toward Intelligent Vehicle/Driver Assistance System, Auranuch Lorsakul - Jackrit Suthakorn