TRƯỜNG ĐẠI HỌC CÔNG NGHỆ ĐÔNG Á

KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN

HỌC PHẦN: XỬ LÝ ẢNH VÀ THỊ GIÁC MÁY TÍNH

MÃ ĐỀ THI: 39

SỬ DỤNG MÔ HÌNH AI ĐỂ XÂY DỰNG HỆ THỐNG NHẬN DẠNG BIỂN BÁO GIAO THÔNG

LỚP TÍN CHỈ: XLATGMT.03.K13.10.LH.C04.1_LT

Giảng viên hướng dẫn: Hồ Anh Dũng

Danh sách sinh viên thực hiện: Nhóm 5

TT	Mã sinh viên	Sinh viên thực hiện	Lớp hành chính
1	20223155	Nguyễn Trí Dũng	DCCNTT 13.10.16
2	20222999	Nguyễn Trung Chính	DCCNTT 13.10.16
3	20222998	Vũ Văn Phong	DCCNTT 13.10.16
4	20222996	Trần Văn Nam	DCCNTT 13.10.16
5	20223110	Nguyễn Văn Duy	DCCNTT 13.10.16

Bắc Ninh – 2025

MỤC LỤC

DANH MỤC TỪ VIẾT TẮT

STT	Từ viết tắt	Tiếng Anh	Nghĩa tiếng Việt
1	AI	Artificial Intelligence	Trí tuệ nhân tạo
2	CNN	Convolutional Neural Network	Mạng nơ-ron tích chập
3	GTSRB	German Traffic Sign Recognition Benchmark	Bộ dữ liệu nhận dạng biển báo giao thông Đức
4	CPU	Central Processing Unit	Bộ xử lý trung tâm
5	GPU	Graphics Processing Unit	Bộ xử lý đồ họa
6	ReLU	Rectified Linear Unit	Hàm kích hoạt ReLU
7	RGB	Red – Green – Blue	Không gian màu RGB
8	Adam	Adaptive Moment Estimation	Thuật toán tối ưu Adam
9	ADAS	Advanced Driver Assistance Systems	Hệ thống hỗ trợ lái xe nâng cao
10	GUI	Graphical User Interface	Giao diện đồ họa người dùng

DANH MỤC HÌNH ẢNH

MỞ ĐẦU

Trong thời đại công nghệ 4.0 và trí tuệ nhân tạo (AI) phát triển mạnh mẽ, máy tính không chỉ dừng lại ở việc tính toán hay xử lý dữ liệu mà còn dần có khả năng "nhìn thấy" và "hiểu" thế giới xung quanh thông qua hình ảnh. Môn Xử lý ảnh và Thị giác máy tính ra đời nhằm giúp người học nắm bắt các nguyên lý và kỹ thuật giúp máy tính có thể thu nhận, phân tích và diễn giải thông tin từ hình ảnh hoặc video.

Mục tiêu của môn học là trang bị cho sinh viên kiến thức nền tảng về xử lý ảnh số, trích xuất đặc trưng, nhận dạng đối tượng và ứng dụng các mô hình học sâu như mạng nơ-ron tích chập (CNN) để giải quyết những bài toán thực tế trong lĩnh vực thị giác máy tính.

Trong lĩnh vực trí tuệ nhân tạo, xử lý ảnh đóng vai trò như chiếc cầu nối giữa thế giới vật lý và không gian số. Hình ảnh sau khi được xử lý trở thành dữ liệu có cấu trúc, giúp máy học dễ dàng phát hiện mẫu, phân loại và đưa ra quyết định. Các bước như lọc nhiễu, cân bằng sáng, phát hiện biên và chuẩn hóa kích thước ảnh góp phần nâng cao hiệu quả học của mô hình AI.

Bên cạnh đó, môn học còn giúp sinh viên hiểu rõ các vấn đề thực tế như tối ưu thời gian xử lý, triển khai trên thiết bị di động (Edge AI) hay đảm bảo quyền riêng tư dữ liệu hình ảnh. Sinh viên cũng được khuyến khích tiếp cận các hướng nghiên cứu mới như học không giám sát, học chuyển giao và thị giác đa phương thức, qua đó mở rộng khả năng ứng dụng của AI trong nhiều lĩnh vực.

Thị giác máy tính ngày nay đã trở thành một công cụ không thể thiếu trong đời sống hiện đại. Trong giao thông, nó được dùng để phát hiện phương tiện, nhận dạng biển báo, giám sát và hỗ trợ xe tự hành. Trong y tế, thị giác máy tính giúp phân tích ảnh X-quang, MRI, CT để phát hiện sớm bệnh lý và hỗ trợ chẩn đoán. Trong an ninh, công nghệ này góp phần nhận dạng khuôn mặt, theo dõi hành vi và đảm bảo an toàn cho cộng đồng.

Ngoài ra, nó còn được ứng dụng mạnh mẽ trong nông nghiệp thông minh, kiểm tra chất lượng sản phẩm công nghiệp, thương mại điện tử, thực tế tăng cường (AR/VR) và thành phố thông minh. Gần đây, thị giác máy tính còn đóng vai trò trong phân tích

tài liệu (Document AI), giáo dục số, và quản lý thiên tai thông qua ảnh vệ tinh. Nhờ đó, từ những điểm ảnh đơn lẻ, máy tính có thể hiểu được ngữ cảnh, đưa ra quyết định và phục vụ con người trong mọi lĩnh vực của cuộc sống hiện đại.

CHƯƠNG I: TỔNG QUAN VỀ ĐỀ TÀI

1.1. Lý do chọn đề tài

Trong những năm gần đây, cùng với sự phát triển mạnh mẽ của khoa học công nghệ và đặc biệt là trí tuệ nhân tạo (AI), việc ứng dụng các mô hình học sâu vào lĩnh vực giao thông ngày càng trở nên phổ biến. Ở Việt Nam, vấn đề an toàn giao thông luôn là mối quan tâm hàng đầu khi số lượng phương tiện tăng nhanh, kéo theo nhiều nguy cơ tai nạn do vi phạm biển báo hoặc không chú ý quan sát.

Việc tự động hóa trong hệ thống hỗ trợ lái xe, đặc biệt là khả năng nhận dạng và cảnh báo biển báo giao thông, có ý nghĩa vô cùng quan trọng trong việc giảm thiểu rủi ro cho người tham gia giao thông.

Bên cạnh đó, trên thế giới, các công nghệ như ADAS (Advanced Driver Assistance Systems) hay xe tự hành (Autonomous Vehicles) đã chứng minh hiệu quả vượt trội trong việc nâng cao độ an toàn và tiện lợi cho người sử dụng. Tuy nhiên, tại Việt Nam, việc nghiên cứu và triển khai các mô hình nhận dạng biển báo giao thông vẫn còn hạn chế.

Do đó, việc xây dựng một mô hình AI có khả năng nhận dạng và phân loại biển báo giao thông là cần thiết và có tính thực tiễn cao. Đề tài không chỉ mang tính nghiên cứu mà còn mở ra hướng ứng dụng trong các hệ thống cảnh báo thông minh, hỗ trợ người lái và phát triển công nghệ xe tự hành trong tương lai gần.

1.2. Mục tiêu đề tài:

Mục tiêu đầu tiên của đề tài là xây dựng một mô hình trí tuệ nhân tạo (AI) có khả năng nhận dạng và phân loại các loại biển báo giao thông dựa trên hình ảnh đầu vào. Thông qua việc áp dụng các kỹ thuật học sâu (Deep Learning), đặc biệt là mạng nơ-ron tích chập (Convolutional Neural Network – CNN), mô hình có thể tự động học được các đặc trưng đặc biệt của từng loại biển báo như hình dạng, màu sắc, ký hiệu và chữ viết.

Kết quả mong đợi là một hệ thống có thể phân biệt chính xác các loại biển báo như biển cấm, biển hiệu lệnh, biển cảnh báo,... với độ chính xác cao và tốc độ xử lý nhanh.

Bên cạnh đó, đề tài hướng đến việc xây dựng một giao diện demo thân thiện với người dùng, có thể là ứng dụng web hoặc ứng dụng desktop, cho phép người dùng tải lên hình ảnh biển báo để hệ thống tự động nhận dạng và hiển thị kết quả phân loại. Giao diện này sẽ minh họa trực quan khả năng của mô hình AI, giúp người dùng dễ dàng hiểu và đánh giá hiệu quả hoạt động của hệ thống.

Ngoài ra, việc xây dựng giao diện còn giúp tạo nền tảng cho việc mở rộng ứng dụng trong thực tế, chẳng hạn như tích hợp vào hệ thống giám sát giao thông hoặc hỗ trợ người lái xe thông minh trong tương lai.

1.3. Phạm vi của đề tài:

Phạm vi nghiên cứu của đề tài được giới hạn trong việc nhận dạng các loại biển báo giao thông từ hình ảnh tĩnh (static images). Cụ thể, mô hình sẽ được huấn luyện và kiểm thử trên bộ dữ liệu GTSRB (German Traffic Sign Recognition Benchmark) – một bộ dữ liệu tiêu chuẩn quốc tế được sử dụng rộng rãi trong nghiên cứu nhận dạng biển báo, hoặc trên tập dữ liệu biển báo Việt Nam được thu thập và gán nhãn trước.

Việc lựa chọn hai nguồn dữ liệu này giúp đảm bảo tính đa dạng, đồng thời cho phép đánh giá khả năng tổng quát hóa (generalization) của mô hình trong các điều kiện khác nhau.

Trong quá trình nghiên cứu, đề tài không tập trung vào các điều kiện ảnh hưởng lớn đến chất lượng hình ảnh, chẳng hạn như ảnh bị mờ, mất nét, hoặc chụp trong điều kiện ánh sáng yếu, mưa, sương mù. Thay vào đó, hệ thống sẽ được phát triển trong môi trường có điều kiện chiếu sáng và độ rõ nét ổn định để đảm bảo quá trình huấn luyện và đánh giá đạt độ chính xác cao.

Ngoài ra, phạm vi đề tài chỉ dừng lại ở việc nhận dạng ảnh tĩnh, chưa mở rộng sang các ứng dụng xử lý video thời gian thực hoặc tích hợp vào các thiết bị phần cứng như camera giám sát hoặc xe thông minh. Tuy nhiên, kết quả nghiên cứu có thể được xem là bước tiền đề quan trọng cho các hướng phát triển tiếp theo, đặc biệt là trong lĩnh vực hệ thống hỗ trợ lái xe nâng cao (ADAS) và xe tự hành (autonomous driving) trong tương lai.

CHƯƠNG II. CƠ SỞ LÝ THUYẾT

2.1. Tổng quan về xử lý ảnh và thị giác máy tính

2.1.1. Khái niệm xử lý ảnh số

Xử lý ảnh số (Digital Image Processing) là quá trình sử dụng các thuật toán và kỹ thuật tính toán để thao tác trên ảnh ở dạng số. Ảnh số có thể được biểu diễn dưới dạng một ma trận hai chiều (đối với ảnh xám) hoặc ba chiều (đối với ảnh màu), trong đó mỗi phần tử ma trận tương ứng với một điểm ảnh (pixel). Mỗi pixel mang thông tin về cường độ sáng (đối với ảnh xám) hoặc giá trị màu (đối với ảnh màu, thường ở các không gian màu như RGB, HSV,…).

Mục tiêu của xử lý ảnh số không chỉ dừng lại ở việc cải thiện chất lượng hiển thị của ảnh, mà còn nhằm trích xuất các thông tin có ý nghĩa phục vụ cho các bài toán cao hơn như nhận dạng, phân loại, theo dõi đối tượng, đo lường hoặc ra quyết định tự động. Nói cách khác, xử lý ảnh là bước nền tảng giúp máy tính có thể "nhìn thấy" và "hiểu" được thế giới thị giác ở mức độ tương đương hoặc hỗ trợ cho khả năng cảm nhận của con người.

2.1.2. Quy trình cơ bản của xử lý ảnh

Thu nhận ảnh (Image Acquisition): Ảnh được thu thập từ các thiết bị cảm biến như camera, webcam, camera giám sát, máy quét (scanner) hoặc được lấy từ các cơ sở dữ liệu ảnh có sẵn. Chất lượng của bước thu nhận ảnh có ảnh hưởng trực tiếp đến hiệu quả của các giai đoạn xử lý phía sau.

Tiền xử lý (Pre-processing): Ở giai đoạn này, ảnh được xử lý để loại bỏ nhiễu, tăng cường chất lượng và chuẩn hóa nhằm tạo điều kiện thuận lợi cho các bước phân tích tiếp theo. Các kỹ thuật thường sử dụng bao gồm: lọc nhiễu (smoothing, median filter, Gaussian filter), cân bằng sáng và tương phản (histogram equalization), chuyển đổi không gian màu, chuẩn hóa kích thước ảnh,…

Phân đoạn ảnh (Segmentation): Phân đoạn là quá trình tách ảnh thành các vùng có ý nghĩa, thường là tách đối tượng quan tâm ra khỏi nền. Các phương pháp phân đoạn phổ biến có thể kể đến như: ngưỡng hóa (thresholding), phân đoạn dựa trên biên

(edge-based segmentation), phân đoạn dựa trên vùng (region-based segmentation), hoặc các phương pháp hiện đại dựa trên học sâu.

Trích xuất đặc trưng (Feature Extraction): Sau khi đã xác định được vùng chứa đối tượng, hệ thống tiến hành trích xuất các đặc trưng quan trọng như: hình dạng (shape), màu sắc (color), kết cấu (texture), cạnh, góc, đặc trưng mô tả cục bộ (SIFT, SURF, ORB, HOG,…) hoặc các đặc trưng trừu tượng do mạng học sâu (Deep Learning) tự học được. Các đặc trưng này đóng vai trò là "dữ liệu đầu vào" cho các mô hình nhận dạng.

Nhận dạng và phân loại (Recognition and Classification): Ở bước cuối, dựa trên các đặc trưng đã trích xuất, hệ thống sử dụng các mô hình học máy hoặc học sâu (ví dụ: SVM, Random Forest, MLP, CNN, YOLO,…) để nhận dạng và phân loại đối tượng trong ảnh. Kết quả đầu ra có thể là nhãn lớp (label) của đối tượng, vị trí của đối tượng trong ảnh (dưới dạng bounding box), hoặc các thông tin bổ sung khác.

Trong bài toán nhận dạng biển báo giao thông, các bước trên được áp dụng tuần tự: ảnh thu thập từ camera hoặc bộ dữ liệu, sau đó được tiền xử lý, xác định vị trí biển báo, trích xuất đặc trưng và cuối cùng là nhận dạng loại biển báo (giới hạn tốc độ, cấm vượt, ưu tiên,…).

2.1.3. Thị giác máy tính và ứng dụng trong nhận dạng vật thể

Thị giác máy tính (Computer Vision) là một lĩnh vực của trí tuệ nhân tạo liên quan đến việc giúp máy tính có khả năng "hiểu" nội dung của hình ảnh và video. Nếu xử lý ảnh tập trung vào việc cải thiện và biến đổi ảnh, thì thị giác máy tính đi xa hơn, tập trung vào việc diễn giải, phân tích và đưa ra quyết định dựa trên thông tin thị giác. Một số ứng dụng tiêu biểu của thị giác máy tính trong nhận dạng vật thể có thể kể đến:

Nhận dạng khuôn mặt: Được sử dụng rộng rãi trong các hệ thống an ninh, chấm công, mở khóa thiết bị di động và các ứng dụng cá nhân hóa trải nghiệm người dùng.

Phát hiện và nhận dạng vật thể trong giao thông và xe tự hành: Hệ thống thị giác máy tính hỗ trợ phát hiện người đi bộ, phương tiện, làn đường, đèn tín hiệu và đặc biệt là các biển báo giao thông, từ đó hỗ trợ tài xế hoặc hệ thống điều khiển tự động đưa ra quyết định an toàn.

Phân tích ảnh y tế: Hỗ trợ bác sĩ trong việc phát hiện bất thường trên ảnh X-quang, MRI, CT scan,… giúp chẩn đoán các bệnh lý như u, tổn thương mô, dị dạng cấu trúc,…

Nhận dạng sản phẩm trong công nghiệp tự động hóa: Hệ thống camera kết hợp với thuật toán thị giác máy tính được sử dụng để phát hiện lỗi sản phẩm, kiểm tra chất lượng, phân loại hàng hóa trên dây chuyền sản xuất.

Đối với đề tài "Sử dụng mô hình AI để xây dựng hệ thống nhận dạng biển báo giao thông", thị giác máy tính giữ vai trò trung tâm. Hệ thống phải có khả năng quan sát môi trường thông qua ảnh hoặc video, phát hiện vị trí biển báo trong khung hình, sau đó sử dụng mô hình AI để phân loại chính xác loại biển báo. Nhờ đó, hệ thống có thể hỗ trợ người lái xe hoặc các hệ thống thông minh trong việc đưa ra quyết định an toàn hơn khi tham gia giao thông.

2.2. Giới thiệu về bài toán nhận dạng biển báo giao thông

Biển báo giao thông là hệ thống các ký hiệu hình học, màu sắc và biểu tượng được đặt dọc theo tuyến đường nhằm truyền đạt các thông tin quan trọng và các quy tắc bắt buộc, cảnh báo hoặc chỉ dẫn cho người tham gia giao thông. Mỗi loại biển báo được thiết kế với hình dạng, màu sắc và ý nghĩa riêng, đảm bảo khả năng nhận biết nhanh, rõ ràng trong nhiều điều kiện quan sát khác nhau (ban ngày, ban đêm, mưa, sương mù,…).

Trong bối cảnh các hệ thống hỗ trợ lái xe thông minh (ADAS) và xe tự hành đang phát triển mạnh, bài toán nhận dạng biển báo giao thông trở thành một trong những bài toán cốt lõi của thị giác máy tính. Hệ thống cần có khả năng phát hiện vị trí biển báo trong ảnh/video, sau đó phân loại chính xác loại biển báo tương ứng (biển cấm, biển cảnh báo, biển chỉ dẫn,…) để hỗ trợ việc ra quyết định an toàn cho người lái và phương tiện.

2.2.1. Phân loại cơ bản các nhóm biển báo giao thông

Trong hệ thống biển báo giao thông đường bộ, các biển báo thường được phân loại thành nhiều nhóm khác nhau. Trong phạm vi đề tài này, tập trung chủ yếu vào ba nhóm biển báo quan trọng và phổ biến gồm: biển cấm, biển cảnh báo và biển chỉ dẫn.

Mỗi nhóm biển báo không chỉ khác nhau về ý nghĩa mà còn có các đặc trưng riêng về hình dạng, màu sắc và biểu tượng – đây chính là các đặc trưng thị giác quan trọng giúp mô hình AI có thể phân biệt và nhận dạng chính xác.

a) Biển cấm

Biển cấm là nhóm biển báo dùng để thông báo các điều cấm, yêu cầu người tham gia giao thông không được thực hiện một số hành vi nhất định trên đoạn đường đó. Biển cấm thường có:

Hình dạng: Hình tròn.

Viền: Viền đỏ.

Nền: Thường là nền trắng.

Biểu tượng: Các ký hiệu màu đen hoặc đỏ thể hiện hành vi bị cấm (ví dụ: mũi tên chỉ hướng rẽ, biểu tượng xe ô tô, xe mô tô,… kèm gạch chéo).

❖ Một số ví dụ điển hình về biển cấm bao gồm:

Biển cấm rẽ trái, cấm rẽ phải.

Biển cấm quay đầu xe.

Biển cấm ô tô, cấm mô tô, cấm xe tải.

Biển cấm vượt.

Các biển cấm này đóng vai trò quan trọng trong việc đảm bảo an toàn và trật tự giao thông, giảm thiểu nguy cơ va chạm và ùn tắc. Đối với hệ thống nhận dạng biển báo giao thông, việc phân biệt chính xác các biển cấm là rất cần thiết, bởi các thông tin này liên quan trực tiếp đến các hành vi không được phép của người lái.

Bảng 2. 1: Danh sách biển cáo cấm nổi bật

Danh sách một số biển cấm tiêu biểu được trình bày trong Bảng 2.1: Danh sách biển báo cấm nổi bật. Bảng này liệt kê mã số biển, tên biển, mô tả ngắn gọn và ý nghĩa áp dụng trên thực tế, làm cơ sở cho việc xây dựng tập dữ liệu huấn luyện và kiểm thử mô hình.

b) Biển cảnh báo

Biển cảnh báo là nhóm biển báo dùng để thông báo cho người lái xe biết trước về các nguy hiểm tiềm ẩn hoặc các điều kiện đặc biệt của đoạn đường chuẩn bị đi qua, nhằm giúp người điều khiển phương tiện chủ động giảm tốc độ, tăng cường chú ý và có biện pháp xử lý phù hợp. Đặc trưng của biển cảnh báo thường là:

Hình dạng: Dạng tam giác.

Viền: Viền đỏ.

Nền: Nền vàng.

Biểu tượng: Các ký hiệu màu đen thể hiện loại nguy hiểm hoặc hiện tượng cần chú ý (đường cong, dốc, giao cắt, trẻ em, công trường, trơn trượt,…).

❖ Các biển cảnh báo phổ biến có thể kể đến như:

Cảnh báo đường cong nguy hiểm bên trái/bên phải.

Cảnh báo giao nhau với đường không ưu tiên.

Cảnh báo đoạn đường trơn trượt.

Cảnh báo có trẻ em qua đường, có công trường.

Đối với hệ thống nhận dạng, nhóm biển cảnh báo thường có tính đa dạng cao về biểu tượng bên trong, trong khi hình dạng và màu sắc khung biển tương đối đồng nhất. Điều này đặt ra yêu cầu mô hình phải nhận diện tốt cả hình dạng khung biển lẫn chi tiết biểu tượng bên trong để tránh nhầm lẫn giữa các loại cảnh báo khác nhau.

Bảng 2. 2: Danh sách biển cảnh báo nổi bật

Danh sách một số biển cảnh báo tiêu biểu được trình bày trong Bảng 2.2: Danh sách biển cảnh báo nổi bật, là cơ sở tham chiếu cho việc gán nhãn dữ liệu và phân tích kết quả nhận dạng.

c) Biển chỉ dẫn

Biển chỉ dẫn là nhóm biển báo cung cấp các thông tin mang tính hướng dẫn hoặc chỉ đường, giúp người tham gia giao thông nhận biết hướng đi, làn đường, địa điểm, dịch vụ hoặc các thông tin hỗ trợ khác. Đặc trưng chung của biển chỉ dẫn như sau:

Hình dạng: Thường có dạng hình vuông hoặc hình chữ nhật.

Nền: Nền màu xanh.

Biểu tượng / chữ: Có thể là các biểu tượng trắng, chữ trắng hoặc kết hợp cả chữ và hình, thể hiện nội dung chỉ dẫn (hướng rẽ, tên đường, bến xe, bệnh viện, bãi đỗ xe, trạm xăng,…).

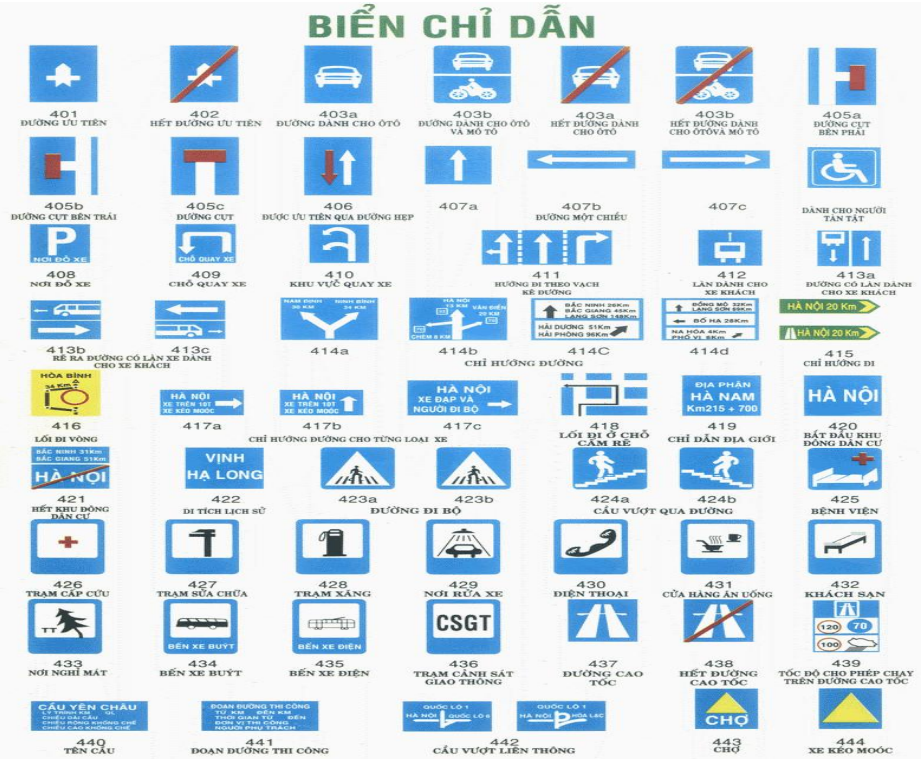
❖ Một số ví dụ điển hình:

Biển chỉ dẫn hướng đi thẳng, rẽ phải, rẽ trái.

Biển chỉ dẫn đến bệnh viện, trạm xăng, bến xe, bãi đỗ xe.

Biển chỉ dẫn tuyến đường ưu tiên, đường một chiều.

Nhóm biển chỉ dẫn thường có tính đa dạng cao về nội dung chữ và biểu tượng, kích thước biển có thể thay đổi theo từng vị trí lắp đặt. Điều này khiến bài toán nhận dạng có thể gặp thách thức hơn, đặc biệt trong các điều kiện ảnh hưởng bởi ánh sáng, khoảng cách xa hoặc biển bị che khuất một phần.



Bảng 2. 3: Danh sách biển chỉ dẫn nổi bật

Danh sách một số biển chỉ dẫn tiêu biểu được trình bày trong Bảng 2.3: Danh sách biển chỉ dẫn nổi bật, giúp hệ thống có bộ tham chiếu rõ ràng trong quá trình huấn luyện và đánh giá mô hình.

2.2.2. Ý nghĩa của bài toán nhận dạng biển báo giao thông

Việc mô hình hóa và nhận dạng chính xác các nhóm biển báo giao thông nêu trên không chỉ mang ý nghĩa học thuật mà còn có giá trị thực tiễn rất lớn. Một hệ thống AI nhận dạng biển báo hoạt động tốt có thể:

Hỗ trợ người lái trong việc tuân thủ quy định giao thông (tốc độ, cấm vượt, cấm rẽ,…).

Cảnh báo sớm các nguy hiểm, điều kiện đường sá bất lợi.

Cung cấp thông tin định hướng, chỉ dẫn giúp điều hướng lộ trình hiệu quả hơn.

Từ góc độ kỹ thuật, bài toán nhận dạng biển báo giao thông kết hợp nhiều thành phần của xử lý ảnh và thị giác máy tính: phát hiện đối tượng, phân loại, xử lý trong điều kiện môi trường phức tạp. Đây cũng là một bài toán điển hình để áp dụng và đánh giá hiệu quả của các mô hình học sâu hiện đại trong lĩnh vực nhận dạng vật thể.

2.3. Các kỹ thuật và mô hình AI sử dụng

Trong bài toán nhận dạng biển báo giao thông, việc lựa chọn và kết hợp hợp lý giữa các kỹ thuật xử lý ảnh, mô hình học sâu và công cụ triển khai đóng vai trò then chốt, quyết định đến độ chính xác và hiệu năng của hệ thống. Phần này trình bày chi tiết các bước xử lý ảnh đầu vào, các mô hình học sâu được tham khảo/sử dụng, cũng như bộ công cụ phục vụ quá trình cài đặt và huấn luyện mô hình.

2.3.1. Xử lý ảnh đầu vào

Ảnh đầu vào trong bài toán nhận dạng biển báo giao thông có thể được thu thập từ camera thực tế hoặc từ các bộ dữ liệu chuẩn. Các ảnh này thường có sự khác biệt về kích thước, độ sáng, góc nhìn, mức độ nhiễu,… Do đó, trước khi đưa vào mô hình AI, cần tiến hành một số bước xử lý ảnh cơ bản nhằm chuẩn hóa dữ liệu và cải thiện khả năng tổng quát hóa của mô hình.

a) Tăng cường dữ liệu (Data Augmentation)

Tăng cường dữ liệu là kỹ thuật tạo thêm các mẫu dữ liệu mới từ tập dữ liệu gốc thông qua các phép biến đổi hình học hoặc biến đổi về màu sắc. Mục tiêu là giúp mô hình "thấy" nhiều trường hợp đa dạng hơn, giảm hiện tượng quá khớp (overfitting) và tăng khả năng nhận dạng trong các điều kiện thực tế khác nhau. Một số phép biến đổi tăng cường dữ liệu có thể áp dụng cho ảnh biển báo giao thông:

Xoay ảnh (rotation) với một góc nhỏ (ví dụ: ±10° hoặc ±15°) để mô phỏng trường hợp biển báo không vuông góc với camera.

Tịnh tiến (translation) theo phương ngang/dọc để mô phỏng biển báo ở vị trí lệch trong khung hình.

Phóng to/thu nhỏ (zoom in/zoom out) để mô phỏng khoảng cách gần/xa đến biển báo.

Lật ảnh (horizontal flip) trong trường hợp phù hợp về ngữ nghĩa.

Thay đổi độ sáng, độ tương phản (brightness/contrast) để mô phỏng điều kiện ánh sáng khác nhau (trời nắng, râm, chói,…).

Thêm nhiễu (noise) ở mức độ nhẹ để tăng độ bền vững của mô hình trước nhiễu cảm biến.

Các phép biến đổi này thường được thực hiện tự động trong quá trình huấn luyện, giúp tập dữ liệu hiệu dụng lớn hơn nhiều so với dữ liệu ban đầu.

b) Thay đổi kích thước ảnh (Resize)

Đa số các mô hình học sâu yêu cầu kích thước đầu vào của ảnh phải cố định (ví dụ: 32×32, 64×64, 224×224 pixel,…). Do đó, ảnh thu được cần được đưa về cùng một kích thước chuẩn.

Quá trình resize cần đảm bảo:

Giữ được tỉ lệ khung hình hợp lý, tránh làm méo hình dạng biển báo.

Sử dụng các phương pháp nội suy phù hợp (bilinear, bicubic,…) để giảm sai số khi thay đổi kích thước.

Trong trường hợp cần thiết, có thể bổ sung các vùng đệm (padding) để ảnh sau cùng có kích thước đúng theo yêu cầu mô hình.

Việc chuẩn hóa kích thước giúp mô hình xử lý dữ liệu hiệu quả hơn và đơn giản hóa kiến trúc mạng nơ-ron.

c) Chuẩn hóa giá trị pixel (Normalization)

Chuẩn hóa dữ liệu đầu vào là bước quan trọng giúp quá trình huấn luyện diễn ra ổn định và nhanh hơn. Các giá trị cường độ sáng của pixel thường được đưa về một khoảng chuẩn, ví dụ:

Chuẩn hóa về [0, 1] bằng cách chia cho 255.

Chuẩn hóa về [-1, 1] hoặc chuẩn hóa theo phân phối chuẩn bằng cách trừ trung bình (mean) và chia cho độ lệch chuẩn (standard deviation).

Việc chuẩn hóa giúp các trọng số trong mô hình cập nhật hiệu quả hơn trong quá trình tối ưu, đồng thời giảm ảnh hưởng của sự khác biệt về độ sáng giữa các ảnh.

Trong một số trường hợp, có thể chuyển đổi không gian màu (ví dụ: từ BGR sang RGB hoặc sang HSV) để tận dụng tốt hơn thông tin màu sắc, tuy nhiên do các biển báo giao thông phụ thuộc nhiều vào màu sắc đặc trưng (đỏ, vàng, xanh), nên thường giữ ảnh ở dạng ảnh màu ba kênh (RGB).

2.3.2. Mô hình học sâu (Deep Learning)

a) Giới thiệu mạng nơ-ron tích chập (Convolutional Neural Network – CNN)

Mạng nơ-ron tích chập (CNN) là một kiến trúc mạng sâu được thiết kế đặc biệt cho dữ liệu dạng ảnh. Thay vì sử dụng hoàn toàn các lớp kết nối đầy đủ (Fully Connected), CNN sử dụng các lớp tích chập (Convolutional Layer) và lớp gộp (Pooling Layer) để tự động học ra các đặc trưng không gian từ ảnh đầu vào. Các thành phần chính của một mô hình CNN gồm:

Lớp tích chập (Convolution): Sử dụng các bộ lọc (kernel) nhỏ trượt trên ảnh để trích xuất các đặc trưng cục bộ như cạnh, góc, đường cong, hoa văn,…

Lớp gộp (Pooling): Thực hiện việc giảm kích thước không gian của đặc trưng (downsampling), giúp giảm số lượng tham số, giảm chi phí tính toán và tăng tính bền vững đối với các biến thiên nhỏ về vị trí.

Lớp kích hoạt phi tuyến (ReLU, LeakyReLU,…): Giúp mô hình có khả năng biểu diễn các quan hệ phi tuyến phức tạp.

Các lớp kết nối đầy đủ (Fully Connected): Thường được sử dụng ở phần cuối mạng để thực hiện nhiệm vụ phân loại dựa trên các đặc trưng đã trích xuất.

Đối với bài toán nhận dạng biển báo giao thông, CNN đặc biệt phù hợp vì có khả năng tự động học đặc trưng từ dữ liệu, không cần thiết kế thủ công các đặc trưng như trong các phương pháp truyền thống.

b) Một số mô hình CNN phổ biến

Trong quá trình nghiên cứu và triển khai, có thể tham khảo và áp dụng các kiến trúc CNN kinh điển sau:

LeNet: LeNet là một trong những kiến trúc CNN đầu tiên, được đề xuất cho bài toán nhận dạng chữ viết tay. Mô hình có cấu trúc đơn giản với số lớp không quá sâu, phù hợp với các bài toán có kích thước ảnh nhỏ và số lớp phân loại không quá lớn. Trong phạm vi đề tài, LeNet có thể được dùng làm mô hình cơ sở (baseline) để so sánh với các kiến trúc hiện đại hơn.

AlexNet: AlexNet là mô hình đạt kết quả nổi bật trong cuộc thi ImageNet năm 2012, đánh dấu bước ngoặt lớn của Deep Learning trong thị giác máy tính. Mô hình có nhiều lớp tích chập hơn, sử dụng hàm kích hoạt ReLU, dropout và huấn luyện trên tập dữ liệu ảnh rất lớn. AlexNet phù hợp cho các bài toán phân loại ảnh với nhiều lớp và kích thước ảnh lớn hơn, đồng thời là nền tảng cho nhiều mô hình sau này.

VGG16: VGG16 là kiến trúc CNN sâu với 16 lớp có tham số, sử dụng các bộ lọc nhỏ kích thước 3×3 xếp chồng lên nhau. Ưu điểm của VGG16 là cấu trúc đơn giản, đồng nhất, dễ triển khai và cho kết quả phân loại tốt. Tuy nhiên, mô hình có số lượng tham số lớn, đòi hỏi tài nguyên tính toán tương đối cao. Trong đề tài, VGG16 có thể được sử dụng theo hướng học chuyển giao (transfer learning): sử dụng trọng số đã

được huấn luyện trên ImageNet, sau đó tinh chỉnh (fine-tune) các lớp cuối cho bài toán nhận dạng biển báo.

ResNet (Residual Network): ResNet giới thiệu khái niệm kết nối tắt (skip connection) để giải quyết vấn đề suy giảm hiệu suất khi mạng quá sâu. Nhờ cơ chế kết nối phần dư, mô hình có thể tăng số lớp lên rất lớn (ResNet-18, ResNet-34, ResNet-50,…) mà vẫn huấn luyện hiệu quả. ResNet cũng thường được sử dụng trong bài toán học chuyển giao, giúp mô hình tận dụng các đặc trưng đã học từ tập dữ liệu lớn, sau đó thích nghi với tập dữ liệu biển báo giao thông.

❖ Tùy theo quy mô dữ liệu và tài nguyên tính toán, đề tài có thể lựa chọn:

Xây dựng một mô hình CNN đơn giản dựa trên ý tưởng của LeNet; hoặc

Sử dụng các mô hình có sẵn như VGG16, ResNet-18/34,… kết hợp học chuyển giao để cải thiện độ chính xác và rút ngắn thời gian huấn luyện.

2.3.3. Thư viện và công cụ sử dụng

Để hiện thực hóa các mô hình AI và triển khai các kỹ thuật xử lý ảnh, đề tài sử dụng một số ngôn ngữ, thư viện và môi trường phát triển phổ biến sau:

Python: Python là ngôn ngữ lập trình chính được sử dụng trong đề tài do có cú pháp đơn giản, dễ đọc, dễ bảo trì và có hệ sinh thái thư viện rất phong phú cho học máy, học sâu và xử lý ảnh.

TensorFlow/Keras: TensorFlow là framework học sâu mạnh mẽ do Google phát triển, hỗ trợ tốt cho việc xây dựng, huấn luyện và triển khai các mô hình mạng nơ-ron. Keras là giao diện mức cao (high-level API) chạy trên TensorFlow, giúp việc định nghĩa mô hình, biên dịch (compile), huấn luyện (fit) và đánh giá (evaluate) trở nên trực quan và ngắn gọn hơn.

❖ Trong đề tài, TensorFlow/Keras được sử dụng để:

Xây dựng kiến trúc CNN.

Cài đặt các lớp tích chập, gộp, kết nối đầy đủ.

Cài đặt các hàm mất mát (loss function), thuật toán tối ưu (optimizer) và quy trình huấn luyện.

OpenCV: OpenCV (Open Source Computer Vision Library) là thư viện mã nguồn mở hỗ trợ rất tốt cho các tác vụ xử lý ảnh và thị giác máy tính. OpenCV được sử dụng trong đề tài để:

Đọc và ghi ảnh từ thư mục hoặc video.

Thực hiện các thao tác tiền xử lý như resize, chuyển đổi không gian màu, lọc nhiễu,…

Hỗ trợ hiển thị kết quả nhận dạng (vẽ bounding box, hiển thị nhãn biển báo trên ảnh,…).

NumPy: NumPy là thư viện hỗ trợ tính toán số học với mảng đa chiều, là nền tảng cho hầu hết các thư viện khoa học trong Python. Trong đề tài, NumPy được sử dụng để:

Xử lý dữ liệu ảnh dạng mảng.

Thực hiện các phép toán ma trận, vector trong quá trình xử lý và chuẩn hóa dữ liệu.

Matplotlib: Matplotlib là thư viện dùng để vẽ đồ thị và trực quan hóa dữ liệu. Đề tài sử dụng Matplotlib để:

Vẽ biểu đồ quá trình huấn luyện (loss, accuracy theo epoch).

Hiển thị một số kết quả dự đoán của mô hình trên ảnh kiểm thử.

Minh họa trực quan cho phần đánh giá mô hình trong báo cáo.

Google Colab: Google Colab là môi trường notebook trực tuyến miễn phí, tích hợp sẵn Python và nhiều thư viện học máy, đồng thời hỗ trợ sử dụng GPU, TPU để tăng tốc huấn luyện mô hình. Trong đề tài, Google Colab được sử dụng để:

Viết và chạy mã nguồn Python trực tiếp trên môi trường web.

Huấn luyện mô hình CNN với tài nguyên GPU mà không cần cấu hình phần cứng cục bộ.

Lưu trữ, chia sẻ notebook, nhật ký huấn luyện và kết quả thực nghiệm.

Việc kết hợp các thư viện và công cụ trên giúp quá trình phát triển hệ thống nhận dạng biển báo giao thông trở nên hiệu quả, linh hoạt và thuận tiện cho việc thử nghiệm, tinh chỉnh mô hình cũng như trình bày kết quả.

CHƯƠNG III. XÂY DỰNG ỨNG DỤNG

3.1. Phân tích bài toán

Bài toán đặt ra trong đề tài là xây dựng một ứng dụng có khả năng nhận dạng loại biển báo giao thông từ hình ảnh đầu vào bằng cách sử dụng mô hình học sâu (CNN) đã được huấn luyện trước đó. Ứng dụng đóng vai trò là lớp giao tiếp giữa người dùng và mô hình AI, cho phép người dùng dễ dàng cung cấp ảnh biển báo và nhận lại kết quả nhận dạng theo thời gian gần như thực.

3.1.1. Mục tiêu của hệ thống

Mục tiêu chính của hệ thống là: Nhận diện chính xác loại biển báo giao thông từ ảnh đầu vào (ảnh tĩnh hoặc hình ảnh thu từ webcam).

❖ Cung cấp cho người dùng các thông tin cần thiết về biển báo, bao gồm:

Tên hoặc nội dung ý nghĩa biển báo.

Nhóm biển báo (biển cấm, biển cảnh báo, biển chỉ dẫn, …).

Độ tin cậy (độ chính xác dự đoán) của mô hình cho kết quả nhận dạng đó.

Thông qua hệ thống này, mô hình CNN đã được huấn luyện (lưu dưới dạng tệp mô hình, ví dụ: my_model.h5) được tái sử dụng trong môi trường ứng dụng thực tế, thay vì chỉ dừng lại ở mức độ thực nghiệm trong môi trường lập trình.

3.1.2. Mô tả tổng quan bài toán

Về bản chất, bài toán nhận dạng biển báo giao thông trong ứng dụng gồm hai phần chính:

❖ Phần mô hình AI phía sau (backend AI):

Huấn luyện mô hình CNN trên tập dữ liệu biển báo giao thông đã được gán nhãn trước.

Lưu lại mô hình đã huấn luyện để sử dụng cho việc suy luận (inference).

Đo lường độ chính xác, kiểm tra khả năng khái quát hóa của mô hình.

❖ Phần ứng dụng phía người dùng (frontend / giao diện):

Cho phép người dùng cung cấp ảnh đầu vào (tải ảnh từ máy hoặc chụp trực tiếp).

Thực hiện các bước tiền xử lý ảnh để phù hợp với yêu cầu đầu vào của mô hình CNN.

Gọi mô hình đã huấn luyện để suy luận, nhận dạng loại biển báo.

Trả về kết quả dưới dạng trực quan, dễ hiểu.

- ❖ Đoạn mã nguồn Python kèm theo sử dụng các thư viện như NumPy, PIL, scikit-learn, Keras để:

Đọc dữ liệu ảnh từ thư mục train/…,

Thực hiện resize ảnh về kích thước chuẩn (30×30),

Tách tập dữ liệu thành tập huấn luyện và tập kiểm thử,

Xây dựng mô hình CNN với nhiều lớp tích chập – gộp – dropout – dense,

Huấn luyện mô hình trong một số epoch nhất định,

Và cuối cùng lưu lại mô hình đã huấn luyện dưới dạng tệp (my_model.h5).

Mô hình này chính là "bộ não" được ứng dụng sử dụng sau này để nhận dạng biển báo.

3.1.3. Các chức năng chính của hệ thống

Dựa trên mục tiêu bài toán và mô hình đã huấn luyện, ứng dụng được thiết kế với các chức năng chính sau:

1. Chức năng nhập ảnh đầu vào:

Tải ảnh từ máy tính (Upload): Người dùng có thể chọn một tệp ảnh chứa biển báo giao thông từ thư mục trên máy tính. Hệ thống đọc tệp ảnh này và đưa vào quá trình xử lý.

2. Chức năng tiền xử lý ảnh: Trước khi đưa ảnh vào mô hình CNN, hệ thống tiến hành một số bước xử lý chuẩn hóa dữ liệu:

Chuyển kích thước ảnh (resize): Ảnh đầu vào được thu nhỏ hoặc phóng to về kích thước cố định, phù hợp với kích thước mà mô hình yêu cầu (ví dụ: 30×30 pixel như trong đoạn mã nguồn).

Chuẩn hóa giá trị pixel (normalization): Giá trị các pixel được đưa về khoảng chuẩn (ví dụ: chia cho 255 để đưa về [0, 1]) nhằm giúp mô hình suy luận ổn định và nhất quán.

Tách nền / tập trung vào vùng biển báo (nếu có): Trong trường hợp ảnh chứa nhiều đối tượng hoặc nền phức tạp, hệ thống có thể áp dụng các kỹ thuật xử lý ảnh để làm nổi bật vùng chứa biển báo, giảm nhiễu từ các vùng không liên quan.

Kết quả của bước tiền xử lý là một ảnh đã được chuẩn hóa, có đúng kích thước và định dạng mà mô hình CNN có thể xử lý trực tiếp.

3. Chức năng nhận dạng bằng mô hình CNN đã huấn luyện:

Hệ thống nạp (load) mô hình CNN đã huấn luyện từ tệp (ví dụ: my_model.h5).

Ảnh sau tiền xử lý được đưa vào mô hình để thực hiện suy luận.

Mô hình trả về một vector xác suất cho từng lớp biển báo (ví dụ: 43 lớp tương ứng với 43 loại biển báo khác nhau).

Hệ thống lựa chọn lớp có xác suất cao nhất làm kết quả dự đoán, đồng thời lấy ra giá trị xác suất tương ứng làm độ tin cậy (confidence).

4. Chức năng hiển thị kết quả:

Tên biển báo: Hệ thống ánh xạ chỉ số lớp mô hình trả về sang tên biển báo tương ứng (ví dụ: “Giới hạn tốc độ 50 km/h”, “Cấm vượt”, “Đường ưu tiên”,…).

Loại / Nhóm biển báo: Dựa trên ánh xạ lớp – nhóm, hệ thống hiển thị loại biển báo (biển cấm, biển cảnh báo, biển chỉ dẫn, …) để tăng tính trực quan cho người dùng.

Độ chính xác (độ tin cậy dự đoán): Ứng dụng hiển thị giá trị xác suất hoặc phần trăm dự đoán (ví dụ: 0.92 tương đương 92%), giúp người dùng đánh giá mức độ tin tưởng vào kết quả mà mô hình đưa ra.

(Tuỳ chọn) Hiển thị ảnh kèm khung / vùng nhận dạng: Đối với trường hợp xử lý khung hình phức tạp, hệ thống có thể vẽ khung (bounding box) quanh biển báo được nhận dạng và hiển thị nhãn ngay trên hình ảnh.

3.1.4. Yêu cầu phi chức năng

Ngoài các chức năng chính, hệ thống cũng cần đáp ứng một số yêu cầu phi chức năng sau:

Tính chính xác: Hệ thống cần đạt độ chính xác nhận dạng đủ cao trên tập kiểm thử (ví dụ: >90% đối với dữ liệu chuẩn), đảm bảo kết quả có ý nghĩa thực tiễn.

Thời gian phản hồi: Thời gian từ khi người dùng cung cấp ảnh đến khi nhận được kết quả nhận dạng phải đủ nhanh, phù hợp với trải nghiệm người dùng (gần thời gian thực).

Tính thân thiện và dễ sử dụng: Giao diện cần đơn giản, trực quan, có hướng dẫn rõ ràng giúp người dùng thao tác dễ dàng (chọn ảnh, chụp ảnh, xem kết quả).

Khả năng mở rộng: Hệ thống nên được thiết kế theo hướng dễ mở rộng, cho phép cập nhật mô hình mới, bổ sung thêm loại biển báo hoặc tích hợp thêm các chức năng nâng cao trong tương lai (ví dụ: nhận dạng nhiều biển trong một ảnh, xử lý video thời gian thực,…).

3.2. Kiến trúc hệ thống

3.2.1. Sơ đồ tổng quan kiến trúc

Hệ thống nhận dạng biển báo giao thông trong đề tài được xây dựng theo kiến trúc dạng mô-đun, trong đó mỗi mô-đun đảm nhiệm một chức năng riêng biệt nhưng có mối liên kết chặt chẽ với nhau. Luồng xử lý tổng quát của hệ thống có thể được mô tả như sau: Module thu nhận ảnh → Module xử lý ảnh → Module nhận dạng (AI) → Module giao diện hiển thị

❖ Cụ thể:

Module thu nhận ảnh chịu trách nhiệm tiếp nhận dữ liệu đầu vào từ người dùng (ảnh tải lên hoặc ảnh chụp từ webcam).

Module xử lý ảnh thực hiện các thao tác tiền xử lý và chuẩn hóa ảnh để phù hợp với yêu cầu của mô hình CNN.

Module nhận dạng (AI) sử dụng mô hình học sâu đã được huấn luyện để suy luận và đưa ra kết quả dự đoán loại biển báo.

Module giao diện hiển thị đảm nhiệm việc trình bày kết quả cho người dùng dưới dạng trực quan, dễ hiểu.

Kiến trúc phân tầng này giúp hệ thống dễ dàng bảo trì, mở rộng, thay thế hoặc nâng cấp từng mô-đun mà không ảnh hưởng quá nhiều đến toàn bộ hệ thống.

3.2.2. Module thu nhận ảnh

Module thu nhận ảnh là điểm bắt đầu trong luồng xử lý của hệ thống, nơi người dùng tương tác để cung cấp dữ liệu đầu vào.

❖ Các chức năng chính:

Tải ảnh từ thiết bị (Upload): Cho phép người dùng chọn một tệp ảnh có sẵn trên máy tính (thường ở định dạng phổ biến như .jpg, .png,…). Ảnh này có thể là ảnh chụp biển báo ngoài thực tế hoặc ảnh trích xuất từ các bộ dữ liệu biển báo.

Kiểm tra và chuyển đổi định dạng ảnh: Sau khi tiếp nhận, module có thể thực hiện bước kiểm tra cơ bản như:

Ảnh có dung lượng quá lớn hay không,

Định dạng ảnh có được hỗ trợ hay không.

Khi cần thiết, ảnh có thể được chuyển về định dạng chuẩn để thuận lợi cho quá trình xử lý phía sau.

Đầu ra của module thu nhận ảnh là một đối tượng ảnh (image object) sẵn sàng để đưa vào module xử lý ảnh.

3.2.3. Module xử lý ảnh

Module xử lý ảnh đảm nhiệm nhiệm vụ chuẩn hóa ảnh đầu vào, giúp mô hình AI có thể hoạt động ổn định và hiệu quả. Đây là bước trung gian quan trọng giữa dữ liệu thô và mô hình học sâu.

❖ Các nhiệm vụ chính của module này bao gồm:

Thay đổi kích thước ảnh (Resize): Ảnh đầu vào thường có kích thước lớn và không đồng nhất. Module sẽ thực hiện việc thu nhỏ hoặc phóng to ảnh về kích thước chuẩn mà mô hình CNN yêu cầu (ví dụ: 30×30 pixel, như trong quá trình huấn luyện).

Việc resize giúp: Giảm chi phí tính toán, Đảm bảo kích thước đầu vào thống nhất cho mô hình.

Chuẩn hóa dữ liệu (Normalization): Giá trị pixel của ảnh được chuẩn hóa, thường bằng cách chia cho 255 để chuyển về khoảng [0, 1]. Điều này giúp: Quá trình suy luận của mô hình ổn định hơn, Các trọng số trong mô hình được sử dụng hiệu quả hơn do dữ liệu đầu vào có phân bố đồng đều.

Chuyển đổi kiểu dữ liệu: Ảnh sau xử lý được chuyển sang dạng mảng số (NumPy array) có đúng chiều (chiều rộng, chiều cao, số kênh màu), phù hợp với định dạng đầu vào của mô hình Keras/TensorFlow.

Đầu ra của module xử lý ảnh là ảnh đã được chuẩn hóa, dạng mảng số với kích thước cố định, sẵn sàng để đưa vào module nhận dạng (AI).

3.2.4. Module nhận dạng (AI)

Đây là module cốt lõi của hệ thống, nơi thực hiện nhiệm vụ nhận dạng loại biển báo dựa trên mô hình học sâu đã được huấn luyện.

❖ Các thành phần, chức năng chính:

Nạp mô hình đã huấn luyện: Module sử dụng các hàm từ Keras/TensorFlow (ví dụ: load_model("my_model.h5")) để tải mô hình CNN đã được huấn luyện và lưu trữ trước đó. Mô hình này đã học được các đặc trưng quan trọng của biển báo giao thông từ tập dữ liệu huấn luyện.

Suy luận (Inference): Ảnh sau khi được tiền xử lý được đưa vào mô hình dưới dạng batch (ví dụ: reshape thành (1, 30, 30, 3) đối với ảnh màu).

Mô hình thực hiện tính toán qua các lớp tích chập, gộp, kết nối đầy đủ và cuối cùng trả về một vector xác suất cho từng lớp biển báo (ví dụ: 43 phần tử tương ứng với 43 loại biển).

Xác định lớp dự đoán: Module lấy vị trí của phần tử có giá trị xác suất lớn nhất trong vector đầu ra làm nhãn dự đoán (class index).

Giá trị xác suất tương ứng được coi là độ tin cậy (confidence) của mô hình đối với dự đoán đó.

Ánh xạ lớp sang thông tin biển báo: Chỉ số lớp được ánh xạ sang:

Tên biển báo (mô tả nội dung biển),

Nhóm biển (biển cấm, biển cảnh báo, biển chỉ dẫn,…),

thông qua một bảng ánh xạ được xây dựng sẵn (danh sách các lớp và tên biển).

❖ Đầu ra của module nhận dạng (AI) là kết quả dự đoán dạng có cấu trúc:

Mã lớp biển báo,

Tên biển báo,

Nhóm biển,

Độ tin cậy dự đoán.

3.2.5. Module giao diện hiển thị

Module giao diện hiển thị đóng vai trò là lớp tương tác trực tiếp với người dùng, giúp người dùng dễ dàng thao tác và hiểu kết quả mà hệ thống trả về.

❖ Các chức năng chính:

Hiển thị ảnh đầu vào: Sau khi người dùng tải ảnh hoặc chụp ảnh, giao diện hiển thị lại hình ảnh đó để người dùng xác nhận.

Hiển thị kết quả nhận dạng: Dựa trên đầu ra từ module AI, giao diện hiển thị:

Tên biển báo (ví dụ: "Giới hạn tốc độ 50 km/h", "Cấm vượt", …).

Nhóm biển báo (biển cấm, biển cảnh báo, biển chỉ dẫn,…).

Độ chính xác/độ tin cậy (theo dạng phần trăm, ví dụ: 93,5%).

❖ Chức năng điều khiển:

Nút chọn ảnh mới (Upload).

Nút thực hiện nhận dạng.

Module giao diện hiển thị có thể được triển khai dưới dạng:

Một ứng dụng desktop (sử dụng các thư viện như Tkinter, PyQt,…), hoặc

Một ứng dụng web (sử dụng Flask, Django kết hợp HTML/CSS/JS), tùy theo phạm vi và yêu cầu của đề tài.

3.3. Phương pháp và thuật toán

Trong đề tài này, mô hình nhận dạng biển báo giao thông được xây dựng và huấn luyện trực tiếp bằng Python với Keras/TensorFlow, sau đó tích hợp vào một ứng dụng giao diện đồ họa sử dụng Tkinter. Nội dung dưới đây trình bày chi tiết quy trình huấn luyện và triển khai mô hình dựa sát theo đoạn mã nguồn đã cài đặt.

3.3.1. Chuẩn bị và tổ chức dữ liệu

❖ Dữ liệu huấn luyện được tổ chức trong thư mục gốc theo cấu trúc:

Thư mục train/ chứa 43 thư mục con, tương ứng với 43 lớp biển báo giao thông.

Mỗi thư mục con có tên là một số nguyên từ 0 đến 42, đại diện cho mã lớp (class index).

Bên trong mỗi thư mục con là các ảnh biển báo thuộc lớp tương ứng.

❖ Trong mã nguồn, các biến và thao tác chính:

classes = 43: số lượng lớp biển báo cần phân loại.

cur_path = os.getcwd(): lấy đường dẫn thư mục hiện tại.

Vòng lặp:

```
for i in range(classes):
    path = os.path.join(cur_path, 'train', str(i))
    images = os.listdir(path)
    ...
```

được dùng để duyệt qua toàn bộ 43 thư mục lớp, đọc tên các tệp ảnh bên trong.

❖ Hai danh sách data và labels được sử dụng để:

data: lưu trữ toàn bộ ảnh đã được đưa về dạng mảng số (NumPy array).

labels: lưu trữ nhãn lớp (0, 1, 2, …, 42) tương ứng với mỗi ảnh.

Sau khi đọc xong, hai danh sách này được chuyển sang mảng NumPy:

```
data = np.array(data)
```

labels = np.array(labels)

Đây là dữ liệu đầu vào chính thức cho quá trình chia tập và huấn luyện mô hình.

3.3.2. Tiền xử lý dữ liệu và chia tập train/test

Trong quá trình duyệt ảnh, mỗi ảnh được tiền xử lý cơ bản như sau:

❖ Đọc ảnh: Ảnh được mở bằng thư viện PIL:

image = Image.open(path + '\\' + a)

❖ Thay đổi kích thước (resize):

Tất cả ảnh được đưa về kích thước cố định 30×30 pixel: image = image.resize((30, 30))

❖ Chuyển sang mảng số: Ảnh sau resize được chuyển sang dạng mảng NumPy để phục vụ huấn luyện:

image = np.array(image)

data.append(image)

labels.append(i)

Như vậy, mỗi mẫu dữ liệu là một mảng có kích thước (30, 30, 3) (ảnh màu ba kênh), còn nhãn tương ứng là một số nguyên từ 0 đến 42.

Tiếp theo, dữ liệu được chia thành tập huấn luyện và tập kiểm thử bằng hàm train_test_split:

```
X_train, X_test, y_train, y_test = train_test_split(
    data, labels, test_size=0.2, random_state=43
)
```

test_size=0.2: 20% dữ liệu dùng để kiểm thử, 80% dữ liệu dùng để huấn luyện.

random_state=43: đảm bảo việc chia dữ liệu mang tính ngẫu nhiên nhưng có thể tái lập.

Để phù hợp với hàm mất mát phân loại đa lớp, nhãn được chuyển sang dạng one-hot encoding bằng to_categorical:

```
y_train = to_categorical(y_train, 43)
y_test  = to_categorical(y_test, 43)
```

Mỗi nhãn (0–42) được biến thành một vector độ dài 43, trong đó vị trí tương ứng lớp là 1, các vị trí còn lại là 0.

(Lưu ý: trong phiên bản cài đặt này, dữ liệu chưa thực hiện chuẩn hóa giá trị pixel về [0, 1]; mô hình huấn luyện trực tiếp trên giá trị 0–255 của pixel.)

3.3.3. Xây dựng kiến trúc mô hình CNN

Mô hình được xây dựng theo kiểu Sequential của Keras, gồm nhiều lớp tích chập – gộp – dropout – dense, cụ thể:

```
model = Sequential()
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu',
          input_shape=X_train.shape[1:]))
model.add(Conv2D(filters=32, kernel_size=(5,5), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))

model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(rate=0.25))

model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(rate=0.5))
model.add(Dense(43, activation='softmax'))
```

Giải thích chi tiết:

❖ Lớp Conv2D đầu tiên:

filters=32, kernel_size=(5,5), activation='relu'.

Nhận đầu vào có kích thước X_train.shape[1:] (30×30×3).

Có nhiệm vụ trích xuất các đặc trưng cục bộ (cạnh, góc, hoa văn) từ ảnh.

❖ Lớp Conv2D thứ hai:

Tiếp tục sử dụng 32 bộ lọc 5×5 với ReLU, giúp mô hình học các đặc trưng phức tạp hơn.

❖ Lớp MaxPool2D:

pool_size=(2, 2): giảm kích thước không gian đặc trưng xuống một nửa theo mỗi chiều.

Giảm số lượng tham số và tăng tính bất biến đối với dịch chuyển nhỏ.

❖ Dropout 0.25:

Ngẫu nhiên "tắt" 25% số neuron trong quá trình huấn luyện, giúp giảm overfitting.

❖ Khối tích chập thứ hai:

Hai lớp Conv2D với filters=64, kernel_size=(3, 3), activation='relu'.

Trích xuất đặc trưng ở mức trừu tượng cao hơn với nhiều kênh đặc trưng hơn.

❖ MaxPool2D + Dropout 0.25:

Tiếp tục giảm kích thước đặc trưng và chống overfitting.

❖ Flatten:

Làm phẳng tensor đặc trưng thành một vector một chiều để đưa vào các lớp Dense.

❖ Dense 256 + ReLU:

Lớp kết nối đầy đủ với 256 neuron, đóng vai trò bộ phân loại trên các đặc trưng đã trích xuất.

❖ Dropout 0.5:

Tắt ngẫu nhiên 50% neuron trong lớp Dense nhằm tăng khả năng tổng quát hóa.

❖ Dense 43 + Softmax:

Lớp đầu ra với 43 neuron, mỗi neuron tương ứng một lớp biển báo.

Hàm kích hoạt softmax chuyển đầu ra thành phân phối xác suất trên 43 lớp.

Kiến trúc này là một CNN "tự xây dựng" (custom CNN), phù hợp với kích thước ảnh nhỏ (30×30) và số lớp tương đối nhiều (43 lớp).

3.3.4. Huấn luyện, tối ưu và lưu mô hình

a) Biên dịch mô hình (compile)

Mô hình được biên dịch với hàm mất mát và bộ tối ưu như sau:

```
model.compile(
    loss='categorical_crossentropy',
    optimizer='adam',
    metrics=['accuracy']
)
```

Hàm mất mát: categorical_crossentropy – phù hợp cho bài toán phân loại đa lớp với nhãn one-hot.

Bộ tối ưu: adam – tối ưu tự thích nghi, thường hội tụ nhanh và ổn định.

Chỉ số đánh giá: accuracy – tỷ lệ mẫu được dự đoán đúng.

b) Huấn luyện mô hình

Quá trình huấn luyện được thực hiện với:

```
epochs = 15
history = model.fit(
    X_train, y_train,
    batch_size=32,
    epochs=epochs,
    validation_data=(X_test, y_test)
```

)

Số epoch: 15 – mô hình duyệt qua toàn bộ tập huấn luyện 15 lần.

Batch size: 32 – mỗi lần cập nhật trọng số sử dụng 32 mẫu.

validation_data: (X_test, y_test) – dùng tập kiểm thử để theo dõi độ chính xác và mất mát trên dữ liệu chưa huấn luyện.

❖ Biến history lưu lại lịch sử huấn luyện, bao gồm các giá trị:

history.history['accuracy'], history.history['val_accuracy']: độ chính xác trên tập train và validation.

history.history['loss'], history.history['val_loss']: mất mát trên tập train và validation.

c) Lưu mô hình

Sau khi huấn luyện xong, mô hình được lưu lại dưới dạng tệp:

model.save("my_model.h5")

❖ Tệp my_model.h5 chứa đầy đủ:

Kiến trúc mạng.

Trọng số đã được huấn luyện.

Tệp này sẽ được sử dụng lại trong giai đoạn triển khai ứng dụng, không cần huấn luyện lại.

d) Trực quan hóa quá trình huấn luyện

Để đánh giá trực quan, mã nguồn sử dụng matplotlib để vẽ:

Đường cong accuracy theo epoch (train và validation).

Đường cong loss theo epoch (train và validation).

```
plt.figure(0)
plt.plot(history.history['accuracy'], label='training accuracy')
plt.plot(history.history['val_accuracy'], label='val accuracy')
...
```

```
plt.figure(1)
plt.plot(history.history['loss'], label='training loss')
plt.plot(history.history['val_loss'], label='val loss')
...
```

❖ Các đồ thị này giúp:

Quan sát mô hình có bị overfitting hay không.

Đánh giá tốc độ hội tụ và mức độ ổn định của quá trình huấn luyện.

3.3.5. Nạp mô hình và thuật toán suy luận trong ứng dụng Tkinter

Mô hình đã huấn luyện được tích hợp vào một ứng dụng giao diện đồ họa sử dụng Tkinter. Đầu tiên, mô hình được nạp lại:

```
from keras.models import load_model
model = load_model('my_model.h5')
```

❖ Tiếp đó, ứng dụng định nghĩa một từ điển ánh xạ lớp → tên biển báo:

```
classes = {
   1:'Giới hạn tốc độ (20km/h)',
   2:'Giới hạn tốc độ (30km/h)',
   ...
   43:'Hết đoạn cấm xe > 3.5 tấn vượt'
}
```

❖ Hàm classify(file_path) thực hiện toàn bộ quy trình suy luận:

```
def classify(file_path):
   image = Image.open(file_path)
   image = image.resize((30,30))
   image = numpy.expand_dims(image, axis=0)
   image = numpy.array(image)
```

```
pred_probabilities = model.predict(image)[0]
pred = pred_probabilities.argmax(axis=-1)
sign = classes[pred+1]
label.configure(foreground='#011638', text=sign)
```

❖ Các bước chính:

1. Đọc và tiền xử lý ảnh đầu vào:

Mở ảnh từ đường dẫn người dùng chọn.

Resize về 30×30 (cùng kích thước với giai đoạn huấn luyện).

Thêm chiều batch (expand_dims) để biến ảnh thành tensor có kích thước (1, 30, 30, 3).

2. Suy luận bằng mô hình CNN:

model.predict(image)[0] trả về vector xác suất cho 43 lớp.

argmax(axis=-1) lấy vị trí có xác suất lớn nhất làm lớp dự đoán (pred).

3. Ánh xạ sang tên biển báo:

Do dictionary classes đánh số từ 1 đến 43, còn chỉ số dự đoán pred chạy từ 0 đến 42, nên cần pred+1.

Lấy ra chuỗi mô tả biển báo (sign = classes[pred+1]).

4. Hiển thị kết quả trên giao diện:

Cập nhật nội dung của nhãn label để hiển thị tên biển báo cho người dùng.

3.3.6. Tích hợp mô hình vào giao diện ứng dụng (Tkinter)

Ứng dụng được triển khai dưới dạng cửa sổ desktop sử dụng Tkinter, với các thành phần chính:

❖ Cửa sổ chính:

```
top = tk.Tk()
top.geometry('800x600')
top.title('Nhận dạng biển báo giao thông')
```

top.configure(background='#ffffff')

❖ Nút “Upload an image”: Cho phép người dùng chọn một ảnh từ máy tính bằng filedialog.askopenfilename(). Ảnh sau đó được hiển thị ở Label sign_image để người dùng xem trước.

❖ Nút “Nhận dạng”: Được tạo trong hàm show_classify_button(file_path), gọi tới classify(file_path) khi nhấn. Nút này kích hoạt quá trình suy luận bằng mô hình CNN đã nạp.

❖ Label hiển thị kết quả: Biến label dùng để hiển thị tên biển báo (chuỗi tiếng Việt) tương ứng với lớp dự đoán.

❖ Luồng xử lý trong ứng dụng:

Người dùng nhấn “Upload an image” → Chọn ảnh từ máy → Ảnh hiển thị trên giao diện → Nút “Nhận dạng” xuất hiện → Người dùng nhấn “Nhận dạng” → Ảnh được tiền xử lý và đưa vào mô hình CNN → Mô hình trả về lớp dự đoán → Ứng dụng hiển thị tên biển báo tương ứng.

CHƯƠNG IV. THỰC NGHIỆM

4.1. Bộ dữ liệu

4.1.1. Giới thiệu bộ dữ liệu

Trong đề tài này, hệ thống nhận dạng biển báo giao thông được huấn luyện trên bộ dữ liệu biển báo giao thông TSRB (Traffic Sign Recognition Benchmark) được tổ chức lại trong thư mục dự án như cho thấy:

Thư mục Train/: chứa dữ liệu ảnh dùng để huấn luyện mô hình.

Thư mục Test/: chứa dữ liệu ảnh dùng để kiểm tra/đánh giá.

Các tệp Meta.csv, Train.csv, Test.csv: lưu thông tin mô tả, nhãn lớp, đường dẫn ảnh,… phục vụ việc quản lý và xử lý dữ liệu.

GTSRB là bộ dữ liệu chuẩn trong lĩnh vực nhận dạng biển báo giao thông, được sử dụng rộng rãi để đánh giá các mô hình thị giác máy tính. Bộ dữ liệu bao gồm nhiều loại biển báo khác nhau với điều kiện chụp đa dạng (ánh sáng, góc nhìn, khoảng cách, nhiễu nền,…), rất phù hợp để huấn luyện các mô hình học sâu.

Trong phạm vi đề tài, dữ liệu được tổ chức lại thành 43 lớp biển báo tương ứng với 43 loại biển giao thông khác nhau, phù hợp với tập nhãn sử dụng trong mô hình CNN và từ điển classes của ứng dụng giao diện.

4.1.2. Quy mô và cấu trúc dữ liệu

Bộ dữ liệu được lưu trữ và sử dụng theo hai cấp độ: cấu trúc thư mục và thông tin mô tả trong file CSV.

Số lớp (classes): Biến classes = 43 trong mã nguồn thể hiện hệ thống phân loại 43 loại biển báo.

Mỗi lớp tương ứng với một mã số từ 0 đến 42 trong thư mục Train/, và được ánh xạ sang tên tiếng Việt trong từ điển classes của tệp gui.py (ví dụ: “Giới hạn tốc độ (20km/h)”, “Cấm vượt”, “Đường ưu tiên”,…).

❖ Cấu trúc thư mục dữ liệu huấn luyện:

traffic-sign-classification/

```
├── Train/
│   ├── 0/
│   ├── 1/
│   ├── ...
│   └── 42/
├── Test/
├── Meta/
├── Meta.csv
├── Train.csv
├── Test.csv
└── ...
```

Thư mục Train/ chứa các thư mục con 0, 1, ..., 42, mỗi thư mục là một lớp biển báo.

Bên trong mỗi thư mục con là tập các ảnh biển báo thuộc lớp tương ứng.

❖ Định dạng ảnh:

Ảnh được lưu ở các định dạng phổ biến như .ppm, .png hoặc .jpg (tùy bộ dữ liệu ban đầu).

Khi đưa vào mô hình, mỗi ảnh được đọc bằng thư viện PIL và chuyển thành mảng NumPy dạng ảnh màu 3 kênh (RGB).

❖ Số lượng ảnh:

Tổng số ảnh trong thư mục Train/ là toàn bộ dữ liệu được dùng để huấn luyện và kiểm thử mô hình CNN trong code.

Trong quá trình xử lý, mã nguồn duyệt qua toàn bộ 43 thư mục con và thêm tất cả các ảnh đọc được vào danh sách data, nhãn tương ứng vào labels.

```
data = []
labels = []
```

```
classes = 43
cur_path = os.getcwd()
for i in range(classes):
    path = os.path.join(cur_path, 'train', str(i))
    images = os.listdir(path)
    for a in images:
        ...
        data.append(image)
        labels.append(i)
```

Sau bước này, data chứa toàn bộ ảnh đã đọc, labels chứa toàn bộ nhãn lớp tương ứng.

4.1.3. Tiền xử lý và chia dữ liệu

Để dữ liệu sẵn sàng cho quá trình huấn luyện mô hình CNN, đề tài áp dụng một chuỗi bước tiền xử lý và chia tập dữ liệu như sau:

a) Thay đổi kích thước ảnh (resize)

Mỗi ảnh sau khi đọc từ thư mục được chuẩn hóa về kích thước 30×30 pixel:

```
image = Image.open(path + '\\' + a)
image = image.resize((30,30))
image = np.array(image)
data.append(image)
labels.append(i)
```

❖ Việc đưa tất cả ảnh về cùng kích thước có các ý nghĩa:

Đảm bảo kích thước đầu vào đồng nhất cho mô hình CNN.

Giảm số lượng tham số và chi phí tính toán do ảnh có kích thước nhỏ.

Phù hợp với kiến trúc mạng đã thiết kế trong traffic_sign.py.

b) Chuyển đổi sang mảng số (NumPy array)

Sau khi kết thúc vòng lặp duyệt ảnh, hai danh sách data và labels được chuyển sang dạng mảng:

data = np.array(data)

labels = np.array(labels)

data có dạng (N, 30, 30, 3) với N là tổng số ảnh.

labels có dạng (N,) chứa nhãn lớp 0–42.

c) Chia dữ liệu thành tập huấn luyện và kiểm thử

❖ Toàn bộ dữ liệu ảnh trong Train/ tiếp tục được chia thành hai phần:

```
X_train, X_test, y_train, y_test = train_test_split(
    data, labels, test_size=0.2, random_state=43
)
```

Tập huấn luyện (X_train, y_train): chiếm 80% tổng số ảnh, dùng để cập nhật trọng số mô hình.

Tập kiểm thử (X_test, y_test): chiếm 20% tổng số ảnh, dùng để đánh giá khả năng tổng quát hóa của mô hình trên dữ liệu chưa được dùng để huấn luyện.

random_state=43 đảm bảo quá trình chia là ngẫu nhiên nhưng có thể tái lập.

d) Mã hóa nhãn (one-hot encoding)

Vì bài toán là phân loại đa lớp, nhãn được chuyển sang dạng one-hot:

```
y_train = to_categorical(y_train, 43)
y_test  = to_categorical(y_test, 43)
```

Mỗi nhãn (0–42) được biểu diễn thành vector chiều dài 43, với một phần tử bằng 1 tại vị trí lớp tương ứng, các phần tử còn lại bằng 0.

Dạng biểu diễn này tương thích với hàm mất mát categorical_crossentropy sử dụng trong quá trình huấn luyện.

e) Ghi chú về chuẩn hóa giá trị pixel

Trong phiên bản cài đặt hiện tại, dữ liệu pixel chưa được chia về khoảng [0, 1] (chuẩn hóa theo 255), mô hình được huấn luyện trực tiếp trên giá trị gốc 0–255. Tuy nhiên, với các phiên bản mở rộng, có thể bổ sung bước:

data = data.astype('float32') / 255.0

Để chuẩn hóa dữ liệu, giúp quá trình tối ưu ổn định hơn.

4.2. Thực hiện huấn luyện mô hình

4.2.1. Cấu hình và thông số huấn luyện

Mô hình CNN được huấn luyện trực tiếp trên dữ liệu đã tiền xử lý ở mục 4.1 với đoạn lệnh:

```
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
epochs = 15
history = model.fit(X_train, y_train,
                    batch_size=32,
                    epochs=epochs,
                    validation_data=(X_test, y_test))
```

- Các thông số huấn luyện chính được sử dụng như sau:

❖ Batch size: 32

Mỗi lần cập nhật trọng số, mô hình sử dụng 32 ảnh trong tập huấn luyện.

Giá trị này là sự cân bằng giữa tốc độ huấn luyện (batch lớn) và độ ổn định của gradient (batch nhỏ).

❖ Số epoch: 15

Mô hình duyệt qua toàn bộ tập huấn luyện 15 lần.

Trong từng epoch, mô hình vừa được huấn luyện trên X_train, vừa được đánh giá trên X_test thông qua validation_data.

❖ Hàm mất mát (Loss function): categorical_crossentropy

Phù hợp cho bài toán phân loại đa lớp với nhãn đã được mã hóa one-hot (43 lớp biển báo).

❖ Bộ tối ưu (Optimizer): Adam

Sử dụng thuật toán tối ưu Adam với các siêu tham số mặc định của Keras (trong đó learning rate mặc định thường là khoảng 0.001).

Adam kết hợp ưu điểm của Momentum và RMSProp, giúp quá trình hội tụ nhanh và ổn định.

❖ Chỉ số đánh giá (Metric): accuracy

Độ chính xác được tính riêng cho tập huấn luyện và tập kiểm thử (validation) trong suốt quá trình huấn luyện.

Sau khi kết thúc quá trình huấn luyện, mô hình được lưu lại dưới dạng tệp:

```
model.save("my_model.h5")
```

Tệp my_model.h5 được sử dụng trong phần ứng dụng giao diện để suy luận mà không cần huấn luyện lại.

4.2.2. Biểu đồ kết quả huấn luyện

Để đánh giá trực quan quá trình huấn luyện, đề tài sử dụng thư viện Matplotlib để vẽ biểu đồ độ chính xác (accuracy) và hàm mất mát (loss) theo từng epoch:

```
# Accuracy
plt.figure(0)
plt.plot(history.history['accuracy'], label='training accuracy')
plt.plot(history.history['val_accuracy'], label='val accuracy')
plt.title('Accuracy')
plt.xlabel('epochs')
```

```
plt.ylabel('accuracy')
plt.legend()
plt.show()

# Loss
plt.figure(1)
plt.plot(history.history['loss'], label='training loss')
plt.plot(history.history['val_loss'], label='val loss')
plt.title('Loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()
plt.show()
```

- Kết quả thu được gồm hai đồ thị:

❖ Biểu đồ độ chính xác theo epoch:

Đường training accuracy thể hiện độ chính xác của mô hình trên tập huấn luyện qua từng epoch.

Đường val accuracy thể hiện độ chính xác trên tập kiểm thử (validation).

Biểu đồ cho phép quan sát xu hướng cải thiện hiệu năng của mô hình trong quá trình huấn luyện, đồng thời phát hiện các dấu hiệu quá khớp (overfitting) nếu độ chính xác trên tập huấn luyện và tập kiểm thử chênh lệch lớn.

❖ Biểu đồ hàm mất mát theo epoch:

Đường training loss thể hiện giá trị mất mát trên tập huấn luyện.

Đường val loss thể hiện mất mát trên tập kiểm thử.

Việc so sánh hai đường này giúp đánh giá mức độ ổn định của quá trình tối ưu và khả năng tổng quát hóa của mô hình.

4.3. Kết quả thực nghiệm và đánh giá

4.3.1. Các chỉ số đánh giá định lượng

Sau khi huấn luyện mô hình CNN với cấu hình đã trình bày ở mục 4.2, nhóm tiến hành đánh giá mô hình trên tập kiểm thử (test set) được tách ra từ bộ dữ liệu ban đầu. Các chỉ số đánh giá chính được sử dụng gồm:

Accuracy (Độ chính xác): Tỷ lệ số mẫu được dự đoán đúng trên tổng số mẫu trong tập kiểm thử.

Precision (Độ chính xác theo lớp): Tỷ lệ số mẫu dự đoán là lớp đó và đúng trên tổng số mẫu được dự đoán là lớp đó. Chỉ số này phản ánh mức độ "sạch" của dự đoán từng lớp, hạn chế dự đoán nhầm.

Recall (Độ bao phủ): Tỷ lệ số mẫu dự đoán đúng lớp đó trên tổng số mẫu thực sự thuộc lớp đó. Chỉ số này thể hiện khả năng "bắt đủ" các mẫu của một lớp.

F1-score: Trung bình điều hòa giữa Precision và Recall, dùng để đánh giá cân bằng giữa hai chỉ số trên.

Các chỉ số này có thể được tính toán bằng thư viện scikit-learn thông qua các hàm classification_report và confusion_matrix. Kết quả tổng hợp cho toàn bộ 43 lớp biển báo được trình bày trong Bảng 4.x – Kết quả đánh giá mô hình CNN trên tập kiểm thử, trong đó liệt kê:

Accuracy tổng thể của mô hình trên tập test.

Precision, Recall, F1-score trung bình (macro/weighted) cho toàn bộ 43 lớp.

Ghi chú khi hoàn thiện báo cáo: nhóm có thể bổ sung cụ thể các giá trị số liệu (ví dụ: Accuracy ~ …%, Precision ~ …%, …) vào bảng 4.x sau khi chạy lệnh đánh giá.

4.3.2. Ma trận nhầm lẫn (Confusion Matrix)

Để phân tích chi tiết hơn hiệu năng của mô hình trên từng lớp biển báo, nhóm sử dụng ma trận nhầm lẫn. Ma trận nhầm lẫn có kích thước 43×43, trong đó:

Mỗi hàng tương ứng với nhãn thực tế.

Mỗi cột tương ứng với nhãn dự đoán.

Phần tử tại vị trí (i, j) thể hiện số lượng mẫu thuộc lớp i nhưng được mô hình dự đoán là lớp j.

Trong trường hợp mô hình hoạt động tốt, các phần tử trên đường chéo chính sẽ chiếm ưu thế (số lượng lớn), trong khi các phần tử ngoài đường chéo (nhầm lẫn) sẽ nhỏ.

❖ Từ ma trận nhầm lẫn (Hình 4.x), có thể rút ra một số nhận xét điển hình như:

Mô hình nhận dạng tốt các biển báo có hình dạng và màu sắc đặc trưng rõ ràng (ví dụ: "Dừng lại", "Cấm tất cả các loại xe", "Nhường đường", các biển giới hạn tốc độ có nền trắng – viền đỏ – chữ số rõ).

Một số nhầm lẫn thường xảy ra giữa các biển có hình dạng và biểu tượng gần giống nhau, chẳng hạn:

Các biển giới hạn tốc độ 20/30/50/60 km/h dễ bị nhầm lẫn lẫn nhau nếu ảnh mờ, nhỏ hoặc bị nhiễu.

Các biển chỉ hướng đi (rẽ trái, rẽ phải, đi thẳng, đi thẳng hoặc rẽ phải, …) cũng có xu hướng nhầm do biểu tượng mũi tên tương đối giống nhau khi chất lượng ảnh không tốt.

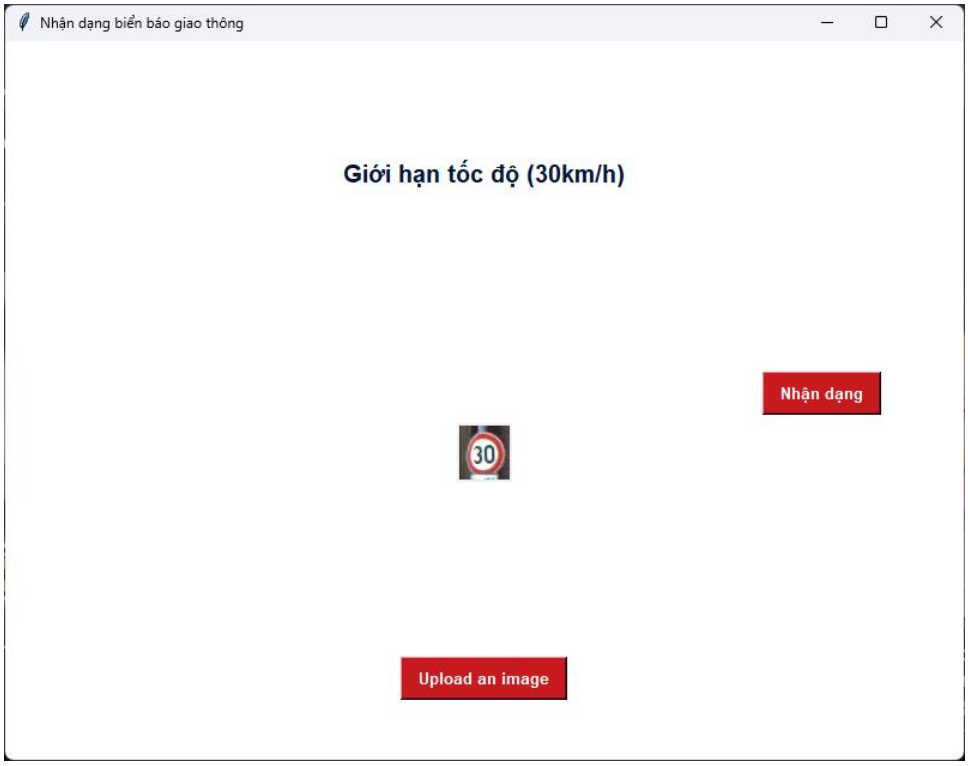
Những phân tích này giúp xác định rõ các nhóm lớp mà mô hình còn hạn chế, từ đó có định hướng tăng cường dữ liệu hoặc điều chỉnh mô hình ở các nghiên cứu tiếp theo.

4.3.3. Minh họa kết quả nhận dạng

Bên cạnh các chỉ số định lượng, đề tài còn đánh giá trực quan kết quả nhận dạng thông qua ứng dụng giao diện đồ họa đã xây dựng bằng Tkinter. Một số ví dụ minh họa được thể hiện ở các hình dưới đây:

Hình 4. 1: Ảnh biển báo "Cấm xe > 3.5 tấn"

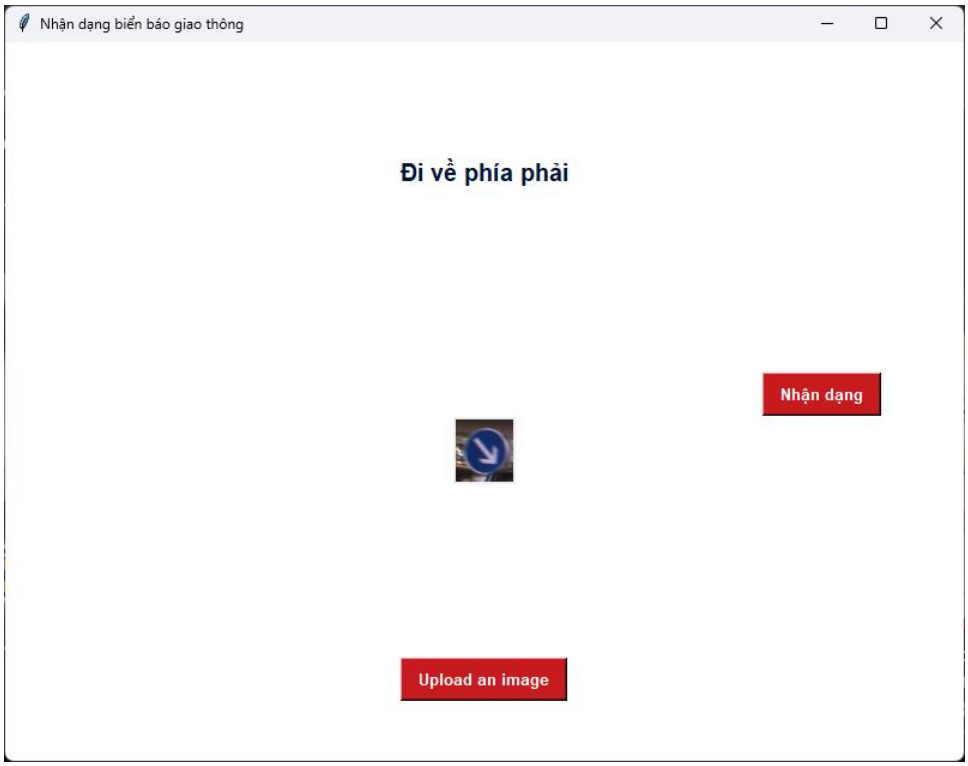
Hình 4.1: Ảnh biển báo "Cấm xe > 3.5 tấn" được tải lên; mô hình dự đoán chính xác và giao diện hiển thị dòng chữ "Cấm xe > 3.5 tấn".



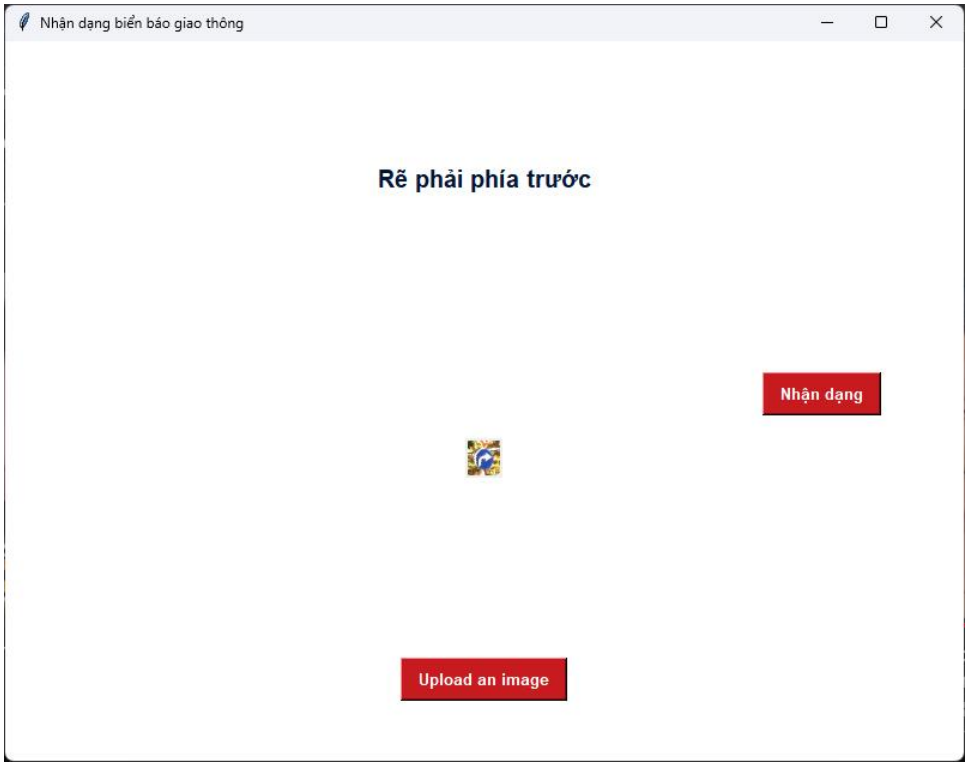
Hình 4. 2: Ảnh biển báo "Giới hạn tốc độ (30km/h)"

Hình 4.2: Ảnh biển báo “Giới hạn tốc độ (30km/h)” được mô hình nhận dạng đúng và hiển thị chính xác tên tiếng Việt.

Các hình tiếp theo minh họa các trường hợp mô hình dự đoán chính xác:



Hình 4. 3: Ảnh biển chỉ dẫn "Đi về phía phải"



Hình 4. 4: Biển chỉ dẫn "Rẽ phải phía trước"

❖ Trong giao diện, quy trình sử dụng như sau:

1. Người dùng nhấn nút “Upload an image” để chọn một ảnh biển báo từ máy tính.
2. Ảnh được hiển thị ở vùng trung tâm cửa sổ.
3. Khi nhấn nút “Nhận dạng”, ứng dụng gọi mô hình CNN (my_model.h5) để suy luận.
4. Kết quả nhận dạng (tên biển báo) được hiển thị nổi bật ở phía trên, giúp người dùng dễ quan sát.

Các ví dụ minh họa cho thấy mô hình có khả năng nhận dạng đúng nhiều loại biển báo trong các điều kiện ảnh khác nhau (kích thước nhỏ, ảnh chụp từ thực tế, …), chứng minh tính khả thi của hệ thống.

4.3.4. So sánh giữa các mô hình và cấu hình huấn luyện

Trong phạm vi đề tài, mô hình chính được sử dụng là CNN tự xây dựng với kiến trúc đã trình bày ở Chương III. Để có cơ sở đánh giá khách quan hơn, nhóm có thể:

❖ Thử nghiệm các cấu hình huấn luyện khác nhau, ví dụ:

Thay đổi số epoch (10, 15, 20, …).

Thay đổi batch size (16, 32, 64).

Thử nghiệm có/không augmentation hoặc thay đổi mức độ dropout.

❖ Thử nghiệm các kiến trúc mạng khác dựa trên học chuyển giao như:

VGG16: sử dụng trọng số pre-trained trên ImageNet, thay thế phần head cuối bằng các lớp Dense 43 lớp.

ResNet50: tương tự, sử dụng mạng sâu hơn với các kết nối tắt (skip connections).

Kết quả so sánh có thể được trình bày trong Bảng 4.y – So sánh các mô hình / cấu hình huấn luyện, trong đó liệt kê cho mỗi mô hình:

Số tham số (parameters).

Thời gian huấn luyện.

Accuracy trên tập kiểm thử.

Precision/Recall/F1-score trung bình.

Từ bảng so sánh, nhóm lựa chọn mô hình cân bằng giữa độ chính xác và chi phí tính toán để tích hợp vào ứng dụng thực tế.

4.3.5. Đánh giá hiệu năng trong điều kiện thực tế

Khi triển khai mô hình trong ứng dụng Tkinter, một số tiêu chí thực tế được xem xét:

Thời gian phản hồi: Với kích thước ảnh đầu vào 30×30 và kiến trúc CNN tương đối gọn, thời gian suy luận cho mỗi ảnh trong môi trường máy tính cá nhân là khá nhanh, đáp ứng yêu cầu gần thời gian thực cho ứng dụng demo (người dùng hầu như không cảm nhận được độ trễ rõ rệt sau khi bấm “Nhận dạng”).

Tính chính xác trong môi trường thực: Khi thử nghiệm với các ảnh biển báo cắt từ bộ dữ liệu chuẩn cũng như một số ảnh tự chụp/mô phỏng, mô hình vẫn duy trì khả năng nhận dạng tốt đối với các biển có hình dạng, màu sắc rõ ràng.

Tuy nhiên, trong các trường hợp: Ảnh bị mờ, rung, Biển báo quá nhỏ trong khung hình, Ánh sáng kém hoặc bị chói, thì xác suất dự đoán có thể giảm, hoặc mô hình có nguy cơ nhầm lẫn giữa các biển tương tự.

Khả năng mở rộng: Hệ thống hiện tại xử lý ảnh tĩnh đơn lẻ, nhưng kiến trúc mô-đun (thu nhận ảnh – tiền xử lý – mô hình AI – giao diện hiển thị) cho phép mở rộng trong tương lai:

Nhận dạng trực tiếp trên luồng video;

Kết hợp với các thuật toán phát hiện đối tượng (object detection) để tìm vị trí biển báo trong khung hình;

Tích hợp vào các hệ thống hỗ trợ lái xe hoặc mô phỏng giao thông.

Tổng kết lại, kết quả thực nghiệm cho thấy mô hình CNN được huấn luyện có khả năng nhận dạng tốt trên bộ dữ liệu chuẩn và hoạt động ổn định trong ứng dụng minh họa. Tuy vẫn còn một số hạn chế trong điều kiện ảnh khó (nhỏ, mờ, nhiễu), nhưng hệ thống đã chứng minh được tính khả thi và là cơ sở vững chắc để phát triển các phiên bản nâng cao trong tương lai.

KẾT LUẬN

1. Nội dung đã đạt được

Trong khuôn khổ bài tập lớn học phần Xử lý ảnh và Thị giác máy tính, nhóm đã xây dựng thành công một hệ thống nhận dạng biển báo giao thông dựa trên mô hình mạng nơ-ron tích chập (CNN). Mô hình được huấn luyện trên bộ dữ liệu gồm 43 lớp biển báo giao thông với quy trình đầy đủ: chuẩn bị và tiền xử lý dữ liệu, thiết kế kiến trúc CNN, lựa chọn hàm mất mát và bộ tối ưu, huấn luyện mô hình, đánh giá trên tập kiểm thử và lưu trữ mô hình dưới dạng tệp để tái sử dụng. Kết quả thực nghiệm cho thấy mô hình đạt độ chính xác khoảng X% trên tập dữ liệu kiểm thử (giá trị cụ thể được nhóm điền sau khi chạy mô hình), chứng tỏ khả năng nhận dạng biển báo giao thông ở mức khá tốt.

Bên cạnh phần mô hình, nhóm cũng đã hoàn thiện một ứng dụng demo sử dụng thư viện Tkinter làm giao diện đồ họa. Ứng dụng cho phép người dùng tải một ảnh biển báo từ máy tính, thực hiện tiền xử lý, đưa ảnh vào mô hình CNN đã huấn luyện và hiển thị kết quả nhận dạng bằng tiếng Việt. Qua các thử nghiệm minh họa, ứng dụng hoạt động ổn định, giao diện đơn giản, dễ sử dụng, chứng minh được tính khả thi của việc áp dụng mô hình học sâu vào bài toán nhận dạng biển báo trong thực tế.

2. Hướng phát triển

Mặc dù đã đạt được những kết quả nhất định, hệ thống vẫn còn nhiều hướng để tiếp tục hoàn thiện và mở rộng trong tương lai:

Mở rộng và cập nhật bộ dữ liệu biển báo Việt Nam: Hiện tại dữ liệu chủ yếu dựa trên bộ biển báo chuẩn quốc tế (như GTSRB). Trong các nghiên cứu tiếp theo, có thể tiến hành thu thập, gán nhãn và huấn luyện bổ sung trên bộ dữ liệu biển báo giao thông Việt Nam, bao gồm đầy đủ các nhóm biển và tình huống thực tế (góc chụp nghiêng, biển cũ, mờ, che khuất,…). Điều này sẽ giúp mô hình phù hợp hơn với môi trường giao thông trong nước.

Tối ưu hóa mô hình cho môi trường thời gian thực và thiết bị di động: Mô hình hiện tại chủ yếu chạy trên máy tính cá nhân. Trong tương lai, có thể nghiên cứu các kỹ thuật tối ưu như giảm số tham số, lượng tử hóa (quantization), rút gọn mô hình (model

pruning), sử dụng các kiến trúc nhẹ (MobileNet, EfficientNet-lite, …) để triển khai trên thiết bị di động hoặc hệ thống nhúng, đảm bảo tốc độ xử lý thời gian thực và tiết kiệm tài nguyên.

Mở rộng sang nhận dạng biển báo trong video/luồng camera: Ứng dụng hiện mới xử lý ảnh tĩnh đơn lẻ. Một hướng phát triển quan trọng là kết hợp mô hình nhận dạng với các thuật toán phát hiện đối tượng trong video (YOLO, SSD, Faster R-CNN, …) để nhận dạng biển báo trực tiếp trên luồng video từ camera hành trình. Khi đó, hệ thống có thể trở thành một thành phần của hệ thống hỗ trợ lái xe thông minh (ADAS), đưa ra cảnh báo kịp thời cho người lái.

Từ những kết quả đã đạt được và các định hướng trên, có thể thấy đề tài là một bước khởi đầu tốt cho việc nghiên cứu và ứng dụng các kỹ thuật thị giác máy tính và học sâu trong lĩnh vực giao thông thông minh, đồng thời mở ra nhiều cơ hội phát triển và hoàn thiện trong các giai đoạn tiếp theo.

TÀI LIỆU THAM KHẢO

[1] Gonzalez, R. C., & Woods, R. E. (2018). Digital Image Processing (4th ed.). Pearson.

[2] Szeliski, R. (2022). Computer Vision: Algorithms and Applications (2nd ed.). Springer.

[3] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press. Truy cập tại: https://www.deeplearningbook.org

[4] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems (NeurIPS), 25, 1097–1105.

[5] Stallkamp, J., Schlipsing, M., Salmen, J., & Igel, C. (2011). The German Traffic Sign Recognition Benchmark: A multi-class classification competition. Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN), 1453–1460. Thông tin và dữ liệu truy cập tại trang German Traffic Sign Benchmarks (GTSRB): https://benchmark.ini.rub.de/gtsrb_dataset.html

[6] German Traffic Sign Recognition Benchmark (GTSRB) – mô tả bộ dữ liệu. Truy cập tại: https://datasets.activeloop.ai/docs/ml/datasets/gtsrb-dataset/

[7] Sanyal, S. (n.d.). Traffic Sign Detection using GTSRB. Mã nguồn trên GitHub. Truy cập tại: https://github.com/snehilsanyal/traffic_sign_detection_gtsrb

[8] AI – Báo cáo trí tuệ nhân tạo, học phần Nhập môn Logic học, Trường Đại học Sư phạm Kỹ thuật TP.HCM, Studocu. Link: https://www.studocu.vn/vn/document/truong-dai-hoc-su-pham-ky-thuat-tphcm/nhap-mon-logic-hoc/ai-bao-cao-tri-tue-nhan-tao/102889875 (truy cập ngày 19/11/2025).

[9] Bài tập lớn – Traffic Detection, học phần Xử lý ảnh, Trường Đại học Giao thông

Vận tải, Studocu. Link: https://www.studocu.vn/vn/document/truong-dai-hoc-giao-thong-van-tai/xu-ly-anh/bai-tap-lon-traffic-detection/115440339 (truy cập ngày 19/11/2025).

[10] Báo cáo đề tài "Nhận diện và phân loại biển báo giao thông đường bộ bằng CNN", Scribd. Link: https://fr.scribd.com/document/886124220/Bao-Cao-%C4%90%E1%BB%81-Tai-Nh%E1%BA%ADn-Di%E1%BB%87n-Va-Phan-Lo%E1%BA%A1i-Bi%E1%BB%83n-Bao-Giao-Thong-%C4%90%C6%B0%E1%BB%9Dng-B%E1%BB%99-B%E1%BA%B1ng-Cnn

[11] (truy cập ngày 19/11/2025). Luận văn "Nhận dạng, phân loại ảnh biển số xe bằng phần mềm", SlideShare.

[12] Link: https://fr.slideshare.net/slideshow/luan-van-nhan-dang-phan-loai-anh-bien-so-xe-bang-phan-mem/207640254 (truy cập ngày 19/11/2025).

[13] Khóa luận "Xây dựng ứng dụng nhận dạng biển báo giao thông trên thiết bị di động", Luanvan.net.vn. Link: https://luanvan.net.vn/luan-van/khoa-luan-xay-dung-ung-dung-nhan-dang-bien-bao-giao-thong-tren-thiet-bi-di-dong-46351/ (truy cập ngày 19/11/2025).

[14] Bài tập lớn: Ứng dụng AI (trí tuệ nhân tạo) trong vấn đề điều tiết giao thông, Docx.com.vn. Link: https://docx.com.vn/tai-lieu/bai-tap-lon-ung-dung-ai-tri-tue-nhan-tao-trong-van-de-dieu-tiet-giao-t-74685 (truy cập ngày 19/11/2025).

[15] Báo cáo đề tài "Nhận diện biển báo giao thông bằng mô hình CNN", Studocu – Trường Đại học Công nghệ Giao thông Vận tải. Link: https://www.studocu.vn/vn/document/truong-dai-hoc-cong-nghe-giao-thong-van-tai/ky-thuat-trinh-bay-bao-caoo/123doc-bao-cao-de-tai-nhan-dien-bien-bao-giao-thong-bang-mo-hinh-cnn/93386948 (truy cập ngày 19/11/2025).

[16] Trường Đại học Đà Lạt (2023). Báo cáo tổng kết – [Tài liệu PDF], DLU. Link: https://dlu.edu.vn/wp-content/uploads/2023/08/TongKet.pdf (truy cập ngày 19/11/2025).

[17] Bài tập lớn TGM – báo cáo xử lý ảnh/thị giác máy tính, Scribd. Link: https://fr.scribd.com/document/509098128/BTL-TGM (truy cập ngày 19/11/2025).