

**ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**  
**TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

---



**BÁO CÁO BÀI TẬP LỚN**  
**HỌC PHẦN: KHAI THÁC DỮ LIỆU VÀ ỨNG DỤNG**  
**ĐỀ TÀI: XÂY DỰNG HỆ THỐNG NHẬN DẠNG BIỂN**  
**BÁO GIAO THÔNG BẰNG MẠNG NƠ-RON**

**Giảng viên hướng dẫn : ThS. Lê Thị Thuỷ**

**Lớp : 20244IT6075001**

**Nhóm : 7**

**Sinh viên thực hiện : 1. Nguyễn Tuấn Minh - 2022607114**  
**2. Trần Văn Nhã - 2022603089**  
**3. Nguyễn Hoàng Tùng - 2022602302**

**Hà Nội, tháng 8 năm 2025**

**ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI**  
**TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

---



**BÁO CÁO BÀI TẬP LỚN**  
**HỌC PHẦN: KHAI THÁC DỮ LIỆU VÀ ỨNG DỤNG**

**ĐỀ TÀI: XÂY DỰNG HỆ THỐNG NHẬN DẠNG BIỂN**  
**BÁO GIAO THÔNG BẰNG MẠNG NƠ-RON**

**Giảng viên hướng dẫn : ThS. Lê Thị Thủy**

**Lớp : 20244IT6075001**

**Nhóm : 7**

**Sinh viên thực hiện : 1. Nguyễn Tuấn Minh - 2022607114**  
**2. Trần Văn Nhã - 2022603089**  
**3. Nguyễn Hoàng Tùng - 2022602302**

**Hà Nội, tháng 8 năm 2025**

# PHIẾU HỌC TẬP CÁ NHÂN/NHÓM

## I. Thông tin chung

Tên lớp học phần: 20244IT6075001

Khóa K17

Họ và tên thành viên trong nhóm:

1. Nguyễn Tuấn Minh
2. Trần Văn Nhã
3. Nguyễn Hoàng Tùng

Tên nhóm: Nhóm 7

## II. Nội dung học tập

1. Tên chủ đề: Xây dựng hệ thống nhận dạng biển báo giao thông bằng mạng nơ-ron
2. Hoạt động của sinh viên
  - Hoạt động/Nội dung 1: ...  
Mục tiêu/chuẩn đầu ra: L2.
  - Hoạt động/Nội dung 2: *Nghiên cứu, mô tả một số kỹ thuật chính hiện có cho bài toán nhận diện biển báo giao thông.*  
Mục tiêu/chuẩn đầu ra: L2
  - Hoạt động/Nội dung 3: *Mô tả công nghệ nhận diện biển báo giao thông bằng CNNs và tiến hành thực nghiệm, phân tích, đánh giá kết quả thực nghiệm.*  
Mục tiêu/chuẩn đầu ra: L2
  - Hoạt động/Nội dung 4: *Viết báo cáo kỹ thuật.*  
Mục tiêu/chuẩn đầu ra: L3
  - Hoạt động/Nội dung 4: *Viết báo cáo phi kỹ thuật.*  
Mục tiêu/chuẩn đầu ra: L4
  - Hoạt động/Nội dung 4: *Thuyết trình kết quả.*  
Mục tiêu/chuẩn đầu ra: L5
3. Sản phẩm nghiên cứu
  - Báo cáo đúng theo mẫu quy định.
  - Mã nguồn chương trình (nếu có).

## III. Nhiệm vụ học tập

1. Hoàn thành toàn bộ nội dung được giao theo đúng thời gian quy định.
2. Trình bày sản phẩm nghiên cứu/thực nghiệm trước giảng viên và các sinh viên khác.

## IV. Học liệu sử dụng

1. Tài liệu học tập:
2. Phương tiện, nguyên liệu thực hiện bài tập lớn: Máy tính cá nhân có kết nối Internet.

KẾ HOẠCH THỰC HIỆN BÀI TẬP LỚN

Tên lớp: 20244IT6075001. Khóa: K17.

Họ và tên sinh viên:

- 1. Nguyễn Tuấn Minh
- 2. Trần Văn Nhã
- 3. Nguyễn Hoàng Tùng

Tên nhóm: Nhóm 7

Tên chủ đề: Xây dựng hệ thống nhận dạng biển báo giao thông sử dụng mạng nơ ron.

Tuần	Người thực hiện	Nội dung công việc	Kết quả đạt được	Phương pháp thực hiện
1	Cả nhóm	Tìm hiểu yêu cầu đề tài, chia nhóm, phân công công việc. Thu thập tài liệu liên quan đến mạng nơ-ron (YOLOv12, Faster R-CNN, CNN).	Hoàn thành kế hoạch phân công công việc và lộ trình thực hiện.	
2	Cả nhóm	Thu thập và xử lý dữ liệu (resize, augmentation). Viết code tiền xử lý dữ liệu.	Bộ dữ liệu chuẩn hóa, sẵn sàng huấn luyện.	
2	Nguyễn Tuấn Minh	Cài đặt và cấu hình môi trường Google Colab, PyTorch, Ultralytics YOLOv12.	Môi trường huấn luyện đã sẵn sàng.	
2	Nguyễn Hoàng Tùng Nguyễn Tuấn Minh	Xây dựng kiến trúc CNN cơ bản để phân loại biển báo.	Hoàn thành mô hình CNN baseline.	
3	Nguyễn Tuấn Minh	Huấn luyện YOLOv12 với bộ dữ liệu đã xử lý.	Mô hình YOLOv12 huấn luyện xong	
3	Trần Văn Nhã Nguyễn Tuấn Minh	Huấn luyện Faster R-CNN (ResNet50 backbone).	Hoàn thành mô hình Faster R-CNN, so sánh sơ bộ với YOLOv12.	

3	Nguyễn Hoàng Tùng	So sánh hiệu quả CNN với YOLOv12, Faster R-CNN.	Báo cáo so sánh mô hình, xác định YOLOv12 có hiệu năng vượt trội hơn.	
4	Cả nhóm	Tích hợp mô hình YOLOv12 vào Flask API. Xây dựng giao diện web bằng HTML/CSS/Bootstrap.	Website cơ bản hoạt động: upload ảnh → hiển thị kết quả nhận dạng.	
5	Cả nhóm	Hoàn thiện chức năng realtime detection qua webcam, biểu đồ thống kê, thông tin chi tiết đối tượng.	Ứng dụng đầy đủ tính năng.	

Ngày    tháng    năm 2025

**XÁC NHẬN CỦA GIẢNG VIÊN**

(Kí, ghi rõ họ tên)

Lê Thị Thuỷ

**BÁO CÁO HOẠT ĐỘNG NHÓM**

Tên lớp: 20244IT6075001. Khóa: K17.

Họ và tên sinh viên:

- 1. Nguyễn Tuấn Minh
- 2. Trần Văn Nhã
- 3. Nguyễn Hoàng Tùng

Tên nhóm: Nhóm 7

Tên chủ đề: Xây dựng hệ thống nhận dạng biển báo giao thông bằng mạng nơ-ron.

Tuần	Người thực hiện	Nội dung công việc	Kết quả đạt được	Kiến nghị với giảng viên hướng dẫn
1	Cả nhóm	Tìm hiểu yêu cầu đề tài, chia nhóm, phân công công việc. Thu thập tài liệu liên quan đến mạng nơ-ron (YOLOv12, Faster R-CNN, CNN).	Hoàn thành kế hoạch phân công công việc và lộ trình thực hiện.	
2	Cả nhóm	Thu thập và xử lý dữ liệu (resize, augmentation). Viết code tiền xử lý dữ liệu.	Bộ dữ liệu chuẩn hóa, sẵn sàng huấn luyện.	
2	Nguyễn Tuấn Minh	Cài đặt và cấu hình môi trường Google Colab, PyTorch, Ultralytics YOLOv12.	Môi trường huấn luyện đã sẵn sàng.	
2	Nguyễn Hoàng Tùng Nguyễn Tuấn Minh	Xây dựng kiến trúc CNN cơ bản để phân loại biển báo.	Hoàn thành mô hình CNN baseline.	
3	Nguyễn Tuấn Minh	Huấn luyện YOLOv12 với bộ dữ liệu đã xử lý.	Mô hình YOLOv12 huấn luyện xong	
3	Trần Văn Nhã Nguyễn Tuấn Minh	Huấn luyện Faster R-CNN (ResNet50 backbone).	Hoàn thành mô hình Faster R-CNN, so sánh sơ bộ với YOLOv12.	

3	Nguyễn Hoàng Tùng	So sánh hiệu quả CNN với YOLOv12, Faster R-CNN.	Báo cáo so sánh mô hình, xác định YOLOv12 có hiệu năng vượt trội hơn.	
4	Cả nhóm	Tích hợp mô hình YOLOv12 vào Flask API. Xây dựng giao diện web bằng HTML/CSS/Bootstrap.	Website cơ bản hoạt động: upload ảnh → hiển thị kết quả nhận dạng.	
5	Cả nhóm	Hoàn thiện chức năng realtime detection qua webcam, biểu đồ thống kê, thông tin chi tiết đối tượng.	Ứng dụng đầy đủ tính năng.	

Ngày    tháng    năm 2025

**XÁC NHẬN CỦA GIẢNG VIÊN**

(Kí, ghi rõ họ tên)

Lê Thị Thuỷ

## MỤC LỤC

MỤC LỤC.....	i
DANH MỤC HÌNH ẢNH .....	iv
DANH MỤC BẢNG BIỂU .....	vi
DANH MỤC TỪ VIẾT TẮT .....	vii
LỜI CẢM ƠN .....	viii
LỜI NÓI ĐẦU .....	1
CHƯƠNG 1. TỔNG QUAN VỀ KHO DỮ LIỆU VÀ PHƯƠNG PHÁP KHAI PHÁ .....	3
1.1. Khái niệm .....	3
1.1.1. Kho dữ liệu.....	3
1.1.2. Khai phá dữ liệu .....	3
1.1.3. Mối quan hệ giữa kho dữ liệu và khai phá dữ liệu .....	4
1.2. Phân loại .....	5
1.2.1. Phân loại kho dữ liệu.....	5
1.2.2. Phân loại phương pháp khai phá dữ liệu.....	6
1.3. Các bước thực hiện.....	6
1.3.1. Khảo sát, hiểu rõ vấn đề kỹ thuật.....	7
1.3.2. Thu thập và xử lý dữ liệu .....	7
1.3.3. Xây dựng hệ thống lưu trữ và xử lý .....	8
1.3.4. Thiết kế mô hình khai phá.....	8
1.3.5. Huấn luyện, tinh chỉnh các tham số .....	9
1.3.6. Triển khai, giám sát, bảo trì.....	9
1.4. Lĩnh vực ứng dụng .....	10
1.5. Hướng nghiên cứu.....	10
1.6. Tổng kết chương.....	12
CHƯƠNG 2. CÁC PHƯƠNG PHÁP KHAI PHÁ DỮ LIỆU .....	13
2.1. Mạng nơ-ron tích chập .....	13
2.1.1. Giới thiệu.....	13

2.1.2. Cấu trúc .....	13
2.1.3. Siêu tham số .....	17
2.1.4. Các hàm kích hoạt thường gặp.....	21
2.1.5. Ưu điểm.....	22
2.1.6. Nhược điểm.....	22
2.1.7. Các kiến trúc CNN phổ biến.....	23
2.2. Vision Transformer (ViT).....	24
2.2.1. Giới thiệu.....	24
2.2.2. Kiến trúc.....	25
2.2.3. Ưu điểm.....	27
2.2.4. Nhược điểm.....	27
2.2.5. Các kiến trúc ViT phổ biến .....	28
CHƯƠNG 3. THỰC NGHIỆM .....	29
3.1. Kỹ thuật sử dụng .....	29
3.2. Môi trường thực nghiệm .....	30
3.3. Dữ liệu thực nghiệm.....	31
3.3.1. Nguồn gốc và đặc điểm của bộ dữ liệu.....	31
3.3.2. Tiền xử lý dữ liệu .....	32
3.4. YOLOv12 .....	35
3.4.1. Tổng quan về mô hình.....	35
3.4.2. Tham số huấn luyện .....	37
3.4.3. Kết quả thực nghiệm.....	38
3.4.4. Đánh giá hiệu suất mô hình.....	39
3.4.5. Lưu trữ mô hình .....	40
3.5. Faster R-CNN.....	40
3.5.1. Tổng quan về mô hình.....	41
3.5.2. Tham số huấn luyện .....	42
3.5.3. Kết quả thực nghiệm.....	43
3.5.4. Đánh giá hiệu suất mô hình.....	44
3.6. CNN.....	44

3.6.1. Tổng quan mô hình .....	45
3.6.2. Tham số huấn luyện .....	47
3.6.3. Kết quả thực nghiệm .....	48
3.6.4. Đánh giá hiệu suất mô hình.....	49
3.7. So sánh mô hình .....	50
<b>CHƯƠNG 4. XÂY DỰNG ỨNG DỤNG CHO MÔ HÌNH NHẬN DẠNG BIỂN</b>	
<b>BÁO GIAO THÔNG .....</b>	<b>52</b>
4.1. Giới thiệu chung.....	52
4.2. Kiến trúc hệ thống.....	54
4.2.1. Giao diện người dùng.....	54
4.2.2. Lớp xử lý ứng dụng.....	55
4.3. Các chức năng chính của hệ thống.....	57
4.3.1. Tổng quan về chức năng chính .....	57
4.3.2. Thu thập và xử lý video.....	59
4.3.3. Phát hiện đối tượng qua ảnh được upload.....	61
4.4. Kết quả và đánh giá .....	63
4.4.1. Kết quả đạt được: .....	63
4.4.2. Đánh giá .....	64
4.5. Hướng phát triển trong tương lai.....	64
<b>KẾT LUẬN .....</b>	<b>66</b>
<b>TÀI LIỆU THAM KHẢO.....</b>	<b>67</b>

## DANH MỤC HÌNH ẢNH

Hình 1.1. Sơ đồ chi tiết quy trình ETL .....	3
Hình 1.2. Sơ đồ xử lý dữ liệu cơ bản của Data Mining .....	4
Hình 1.3. Kiến trúc tổng quan của Data Warehouse .....	5
Hình 1.4. Tổng quan các phương pháp học máy .....	6
Hình 1.5. Hình ảnh minh họa bộ dữ liệu gốc Vietnam Traffic Sign .....	8
Hình 1.6. Sơ đồ pipeline triển khai website – API – mô hình .....	10
Hình 2.1. Mô hình mạng nơ-ron tích chập (Nguồn: Internet).....	13
Hình 2.2. Mô tả cấu trúc mô hình CNN AlexNet (Nguồn: Internet) .....	14
Hình 2.3. Hình ảnh mô tả lớp tích chập (Nguồn: Internet).....	15
Hình 2.4. Hình ảnh mô phỏng lớp kết nối đầy đủ (Nguồn: Internet) .....	16
Hình 2.5. Các chiều của bộ lọc (Nguồn: Internet) .....	17
Hình 2.6. Minh họa độ trượt (Nguồn: Internet) .....	17
Hình 2.7. Tính tương tích của tham số trong tầng tích chập (Nguồn: Internet) .	19
Hình 2.8. Ví dụ minh họa trường thụ cảm (Nguồn: Internet).....	20
Hình 2.9. Minh họa kiến trúc GAN (Nguồn: Internet) .....	24
Hình 2.10. Kiến trúc cơ bản của Vision Transformer (Nguồn: Internet).....	25
Hình 3.1. Hình ảnh huấn luyện mô hình trên Kaggle (Sử dụng GPU T4x2).....	31
Hình 3.2. Biểu đồ số lượng classes có trong dữ liệu.....	32
Hình 3.3. Hình ảnh minh họa kỹ thuật chuẩn hoá kích thước .....	33
Hình 3.4. Hình ảnh minh họa kỹ thuật làm mờ, lọc nhiễu.....	34
Hình 3.5. Hình ảnh minh họa kỹ thuật cân bằng tương phản .....	34
Hình 3.6. Hình ảnh minh họa kỹ thuật xoay, xén, zoom ảnh.....	35
Hình 3.7. Kiến trúc mô hình YOLOv12.....	37
Hình 3.8. Hình ảnh Confusion Matrix của YOLOv12.....	38
Hình 3.9. Kết quả tổng quan của YOLOv12.....	39
Hình 3.10. Hiệu suất huấn luyện của mô hình YOLOv12 .....	39
Hình 3.11. Luồng xử lý của mô hình Faster R-CNN .....	41
Hình 3.12. Kiến trúc mô hình Faster R-CNN .....	42

Hình 3.13. Kết quả tổng quan của Faster R-CNN .....	43
Hình 3.14. Hiệu suất huấn luyện mô hình Faster R-CNN .....	43
Hình 3.15. Kiến trúc mạng CNN .....	46
Hình 3.16. Fully connected Layer.....	47
Hình 3.17. Kết quả tổng quan của CNN .....	48
Hình 3.18. Hiệu suất huấn luyện mô hình CNN .....	48
Hình 3.19. Hình ảnh Confusion Matrix của CNN .....	49
Hình 4.1. Kiến trúc tổng quan mô hình Flask.....	52
Hình 4.2. Hình minh họa cấu trúc website xây dựng bằng Flask .....	53
Hình 4.3. Quy trình triển khai hệ thống .....	54
Hình 4.4. Flowchart mô tả pipeline Image Upload Detection .....	58
Hình 4.5. Hình ảnh giao diện chính của hệ thống.....	59
Hình 4.6. Hình ảnh giao diện phát hiện đối tượng realtime.....	60
Hình 4.7. Hình ảnh lịch sử phát hiện đối tượng.....	61
Hình 4.8. Sơ đồ luồng realtime trực quan.....	61
Hình 4.9. Hình ảnh phát hiện đối tượng bằng ảnh được upload với YOLOv12.	62
Hình 4.10. Luồng xử lý khi người dùng upload ảnh.....	63

## DANH MỤC BẢNG BIỂU

Bảng 2.1. Bảng minh họa phương thức Max Pooling và Average Pooling .....	16
Bảng 2.2. Bảng so sánh 3 phương pháp Zero-padding. ....	17
Bảng 2.3. Bảng tính toán độ phức tạp của mô hình.....	19
Bảng 2.4. Bảng tổng hợp biến thể của ReLU .....	21
Bảng 3.1. Bảng tham số huấn luyện của YOLOv12 .....	38
Bảng 3.2. Đánh giá hiệu suất mô hình YOLOv12 .....	39
Bảng 3.3. Bảng tham số huấn luyện của Faster R-CNN.....	42
Bảng 3.4. Đánh giá hiệu suất mô hình Faster R-CNN.....	44
Bảng 3.5. Bảng tham số huấn luyện của CNN.....	47
Bảng 3.6. Đánh giá hiệu suất mô hình CNN.....	49
Bảng 3.7. Bảng kết quả tổng hợp .....	50
Bảng 4.1. Đánh giá kết quả đạt được .....	64

**DANH MỤC TỪ VIẾT TẮT**

<b>STT</b>	<b>Từ viết tắt</b>	<b>Từ viết đầy đủ</b>
1	YOLO	You only look once
2	AI	Artificial Intelligence
3	ML	Machine Learning
4	CNN	Convolutional Neural Networks
5	RCNN	Region Based Convolutional Neural Networks

## LỜI CẢM ƠN

Đầu tiên cho phép chúng em gửi lời cảm ơn sâu sắc tới các thầy cô Trường Công Nghệ Thông Tin và Truyền Thông – Đại học Công nghiệp Hà Nội, những người đã hết mình truyền đạt và chỉ dẫn cho chúng em những kiến thức, những bài học quý báu và bổ ích. Đặc biệt, chúng em xin được bày tỏ sự tri ân và xin chân thành cảm ơn giảng viên Lê Thị Thuỷ, người đã trực tiếp hướng dẫn, chỉ bảo chúng em trong suốt quá trình học tập, nghiên cứu và hoàn thành được đồ án.

Trải qua thời gian nghiên cứu và làm đề tài, nhóm chúng em đã khám phá được rất nhiều công nghệ mới, tuy nhiên, do năng lực, kiến thức và trình độ chuyên môn còn hạn hẹp nên không thể tránh khỏi một số thiết sót. Vì vậy, chúng em mong nhận được sự đóng góp ý kiến từ các thầy cô để giúp nhóm chúng em hoàn thiện kiến thức và bổ sung thêm thông tin cần thiết cho báo cáo của mình.

Một lần nữa, chúng em xin chân thành cảm ơn thầy và tất cả những người đã giúp đỡ và đồng hành cùng nhóm chúng em trong chặng đường này.

Chúng em xin trân trọng cảm ơn!

***Sinh viên thực hiện***

Nguyễn Tuấn Minh

Trần Văn Nhã

Nguyễn Hoàng Tùng

## LỜI NÓI ĐẦU

Trong bối cảnh công nghệ ngày càng phát triển mạnh mẽ, đặc biệt là sự tiến bộ vượt bậc của trí tuệ nhân tạo và học máy, việc ứng dụng các kỹ thuật này vào lĩnh vực giao thông thông minh đang mở ra nhiều hướng đi mới, góp phần nâng cao an toàn và hiệu quả khi tham gia giao thông. Một trong những hướng nghiên cứu quan trọng hiện nay là nhận dạng biển báo giao thông tự động bằng mạng nơ-ron nhân tạo, đặc biệt là mạng nơ-ron tích chập (Convolutional Neural Networks – CNN).

Biển báo giao thông đóng vai trò quan trọng trong việc điều tiết, hướng dẫn và cảnh báo cho người tham gia giao thông. Tuy nhiên, trong thực tế, việc nhận biết biển báo bằng mắt thường có thể bị ảnh hưởng bởi nhiều yếu tố như tốc độ di chuyển cao, điều kiện thời tiết, ánh sáng kém hoặc các vật cản làm che khuất biển báo. Điều này làm tăng nguy cơ xảy ra tai nạn giao thông do không nhận diện hoặc nhận diện sai biển báo.

Để khắc phục những hạn chế đó, việc xây dựng một hệ thống tự động nhận dạng biển báo giao thông từ hình ảnh thông qua mạng nơ-ron là một giải pháp tiềm năng và cần thiết. Hệ thống này có thể được tích hợp vào các phương tiện giao thông thông minh, hệ thống xe tự hành hoặc các ứng dụng giám sát giao thông, góp phần nâng cao mức độ an toàn và giảm thiểu tai nạn.

Với tầm quan trọng và tính thực tiễn cao, đề tài ***"Xây dựng hệ thống nhận dạng biển báo giao thông bằng mạng nơ-ron"*** được lựa chọn nhằm nghiên cứu và triển khai một mô hình nhận dạng biển báo chính xác, hiệu quả, có khả năng hoạt động tốt trong các điều kiện môi trường khác nhau, từ đó mở rộng khả năng ứng dụng của trí tuệ nhân tạo vào lĩnh vực giao thông hiện đại.

Nội dung của báo cáo sẽ bao gồm các chương như sau:

### ***Chương 1. Tổng quan về kho dữ liệu và phương pháp khai phá:***

Chương này sẽ trình bày tổng quan về các kiến thức, nhu cầu khai phá, cũng như

các khó khăn hiện tại trong quá trình khai phá, đồng thời cũng xác định mục tiêu và phạm vi của bài toán.

**Chương 2. Các phương pháp khai phá dữ liệu:** Trình bày các kỹ thuật học máy và mạng nơ-ron tích chập hiện có, phân tích ưu nhược điểm của từng kỹ thuật và lựa chọn phương pháp phù hợp nhất để giải quyết bài toán nhận diện biển báo giao thông.

**Chương 3. Thực nghiệm:** Chương này mô tả chi tiết quá trình thu thập dữ liệu, tiền xử lý dữ liệu và các bước thực hiện thực nghiệm. Kết quả đạt được từ các mô hình sẽ được trình bày và đánh giá.

**Chương 4. Xây dựng ứng dụng cho mô hình nhận dạng biển báo giao thông:** Sử dụng các công cụ và thư viện lập trình để xây dựng hệ thống nhận diện tự động, bao gồm cả phần giao diện người dùng và phần xử lý ảnh. Các kỹ thuật triển khai và vận hành hệ thống cũng sẽ được mô tả chi tiết.

**Phần kết luận:** Tổng hợp lại những kết quả đạt được, đưa ra những nhận xét và đánh giá về hiệu quả của hệ thống, đồng thời đề xuất những hướng phát triển tiếp theo cho nghiên cứu.

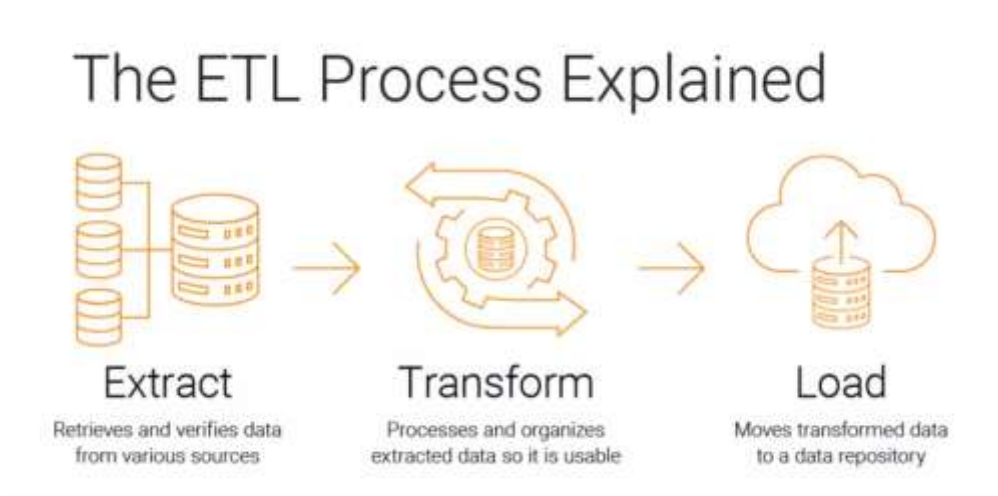
Thông qua việc thực hiện đề tài, người thực hiện có cơ hội tiếp thu thêm nhiều kiến thức mới về trí tuệ nhân tạo và ứng dụng của nó. Đề tài hy vọng sẽ mang lại giá trị thực tiễn và mở ra nhiều cơ hội mới cho các ứng dụng công nghệ.

# CHƯƠNG 1. TỔNG QUAN VỀ KHO DỮ LIỆU VÀ PHƯƠNG PHÁP KHAI PHÁ

## 1.1. Khái niệm

### 1.1.1. Kho dữ liệu

Kho dữ liệu là một hệ thống lưu trữ tập trung, nơi tất cả dữ liệu từ các nguồn giao dịch, hệ thống quản lý và ứng dụng khác nhau được thu thập, chuyển đổi và lưu trữ theo một cấu trúc nhất quán. Thông thường, quá trình ETL (Extract – Transform – Load) sẽ đảm nhiệm việc trích xuất dữ liệu thô từ các nguồn, chuyển đổi định dạng và chuẩn hóa các giá trị, rồi nạp vào kho dữ liệu. Nhờ thiết kế theo chiều chủ đề (subject-oriented), kho dữ liệu giúp phân tách rõ ràng dữ liệu khách hàng, sản phẩm, thời gian... từ đó hỗ trợ việc phân tích báo cáo nhanh chóng và nhất quán. Hơn nữa, cơ chế lưu trữ lịch sử (time-variant) cho phép theo dõi các thay đổi qua thời gian, phục vụ tốt cho các báo cáo xu hướng và dự báo.



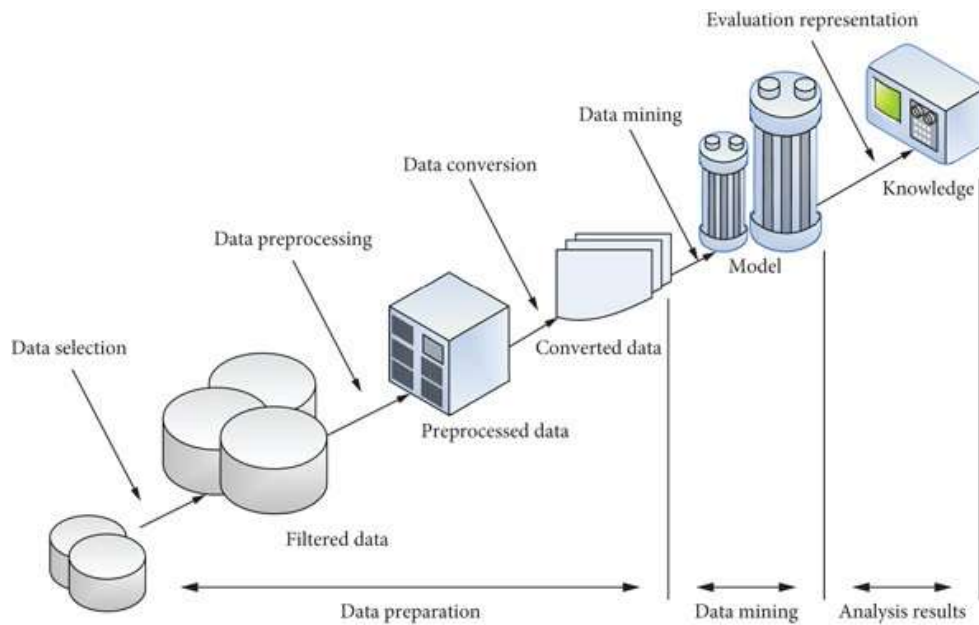
Hình 1.1. Sơ đồ chi tiết quy trình ETL

### 1.1.2. Khai phá dữ liệu

Khai phá dữ liệu là quá trình khám phá tự động hoặc bán tự động các mẫu ẩn, xu hướng hay mối quan hệ trong khối dữ liệu khổng lồ, sử dụng một loạt thuật toán thống kê, toán học và trí tuệ nhân tạo. Trước khi mô hình hóa, dữ liệu cần được làm sạch để loại bỏ giá trị thiếu, xử lý ngoại lai và chuẩn hóa các biến; tiếp

đó là lựa chọn tính năng quan trọng giúp mô hình tránh “quá khớp” hoặc tổn kém về tính toán.

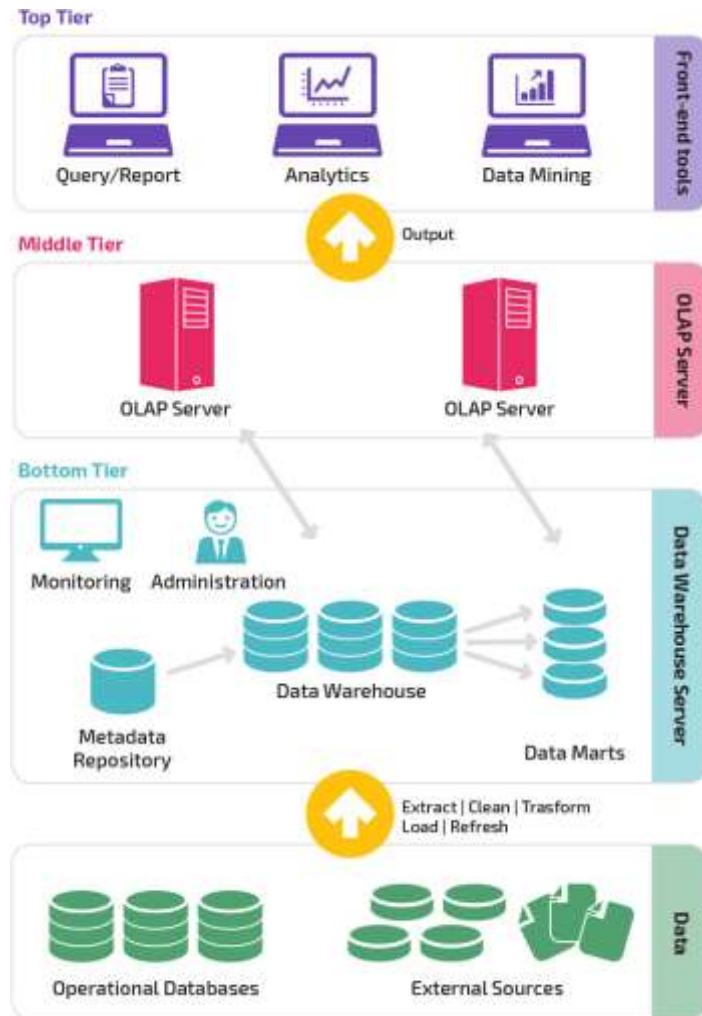
Các thuật toán phổ biến gồm cây quyết định, mạng nơ-ron, k-means clustering, luật kết hợp... sau khi huấn luyện, mô hình sẽ được đánh giá thông qua các chỉ số như accuracy, precision, recall hoặc AUC để đảm bảo kết quả có độ tin cậy cao. Cuối cùng, kết quả khai phá được triển khai vào hệ thống báo cáo, dashboard hoặc tích hợp trong ứng dụng để hỗ trợ ra quyết định.



Hình 1.2. Sơ đồ xử lý dữ liệu cơ bản của Data Mining

### 1.1.3. Mối quan hệ giữa kho dữ liệu và khai phá dữ liệu

Kho dữ liệu và khai phá dữ liệu tạo thành hai nửa của một chu trình phân tích tri thức. Kho dữ liệu cung cấp nguồn dữ liệu đã được chuẩn hóa, tích hợp, và lịch sử đầy đủ, giúp các thuật toán khai phá hoạt động trên dữ liệu sạch và đồng nhất. Ngược lại, kết quả của quá trình khai phá, như các quy luật mua hàng hay phân cụm khách hàng, sau khi được kiểm chứng, có thể được nạp trở lại kho dữ liệu dưới dạng các chỉ số mới hoặc bảng tóm tắt, làm giàu thêm thông tin cho kho. Vòng lặp này không chỉ nâng cao giá trị của dữ liệu mà còn tạo ra môi trường tự động cải thiện kết quả phân tích theo thời gian thông qua việc tái đào tạo và cập nhật mô hình.



Hình 1.3. Kiến trúc tổng quan của Data Warehouse

## 1.2. Phân loại

### 1.2.1. Phân loại kho dữ liệu

Từ góc độ kiến trúc, kho dữ liệu có thể chia làm ba loại chính:

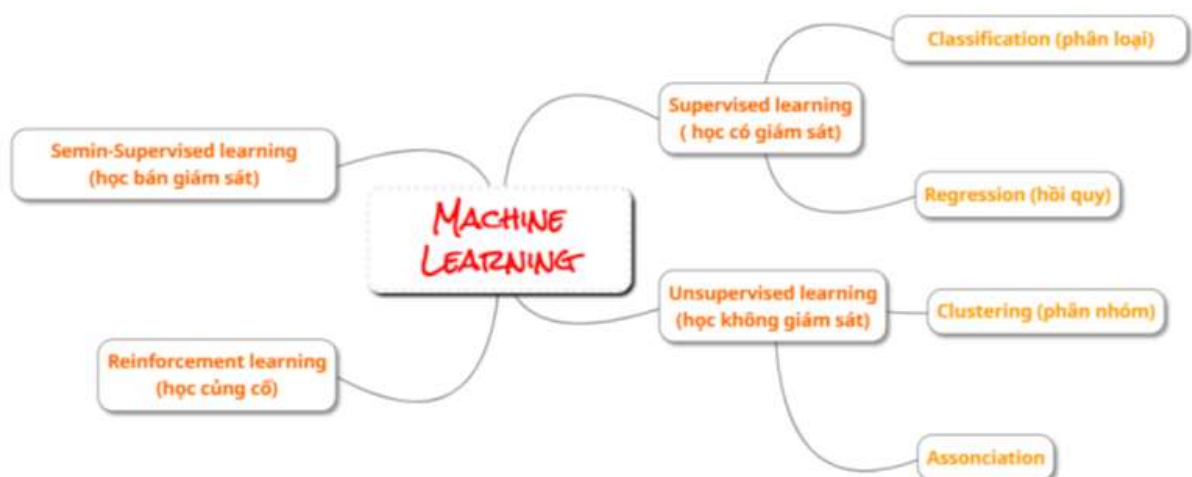
- 1) Kho dữ liệu tập trung (centralized) lưu trữ toàn bộ dữ liệu tại một máy chủ hoặc cụm máy chủ, giúp đơn giản hóa quản trị nhưng có thể gặp khó khăn khi mở rộng
- 2) Kho phân tán (distributed) dữ liệu được lưu trên nhiều nút khác nhau, tận dụng khả năng tính toán song song và tăng tính sẵn sàng, song đòi hỏi giải pháp đồng bộ và bảo mật phức tạp hơn
- 3) Kho thời gian thực (real-time) liên tục nạp dữ liệu gần như tức thì từ các luồng sự kiện, hỗ trợ báo cáo và phân tích ngay trong khi sự kiện

xảy ra. Lựa chọn loại kho dữ liệu phù hợp phụ thuộc vào khối lượng dữ liệu, tốc độ cập nhật và yêu cầu về độ trễ của ứng dụng.

### 1.2.2. Phân loại phương pháp khai phá dữ liệu

Khai phá dữ liệu có thể phân thành bốn nhóm chính dựa trên kiểu học:

- 1) Học có giám sát (supervised learning) sử dụng dữ liệu đã gán nhãn để xây dựng mô hình phân loại hoặc hồi quy, thích hợp cho bài toán dự báo giá hoặc phân loại biên báo
- 2) Học không giám sát (unsupervised learning) khám phá cấu trúc dữ liệu chưa nhãn như phân cụm (clustering) giúp nhóm biên báo theo dạng hình thể và màu sắc
- 3) Học bán giám sát (semi-supervised learning) kết hợp cả dữ liệu có và không nhãn, tối ưu khi nhãn hiếm và tốn công gán nhãn
- 4) Học tăng cường (reinforcement learning) tối ưu chính sách ra quyết định qua quá trình thử-thách-thưởng/phạt, thường dùng trong các hệ thống học lái xe tự động. Mỗi phương pháp có những yêu cầu riêng về dữ liệu và thuật toán, đòi hỏi điều chỉnh cho phù hợp với bài toán nhận dạng biên báo giao thông.



Hình 1.4. Tổng quan các phương pháp học máy

### 1.3. Các bước thực hiện

### 1.3.1. Khảo sát, hiểu rõ vấn đề kỹ thuật

Trong phạm vi mô phỏng của đề tài, việc khảo sát tập trung vào nghiên cứu bài toán nhận dạng biển báo giao thông từ góc độ kỹ thuật — bao gồm phân tích các loại biển báo thường gặp, các đặc trưng hình ảnh (màu sắc, hình dạng), các thách thức khi nhận dạng (góc chụp xiên, điều kiện ánh sáng kém, biển bị mờ hoặc che khuất).

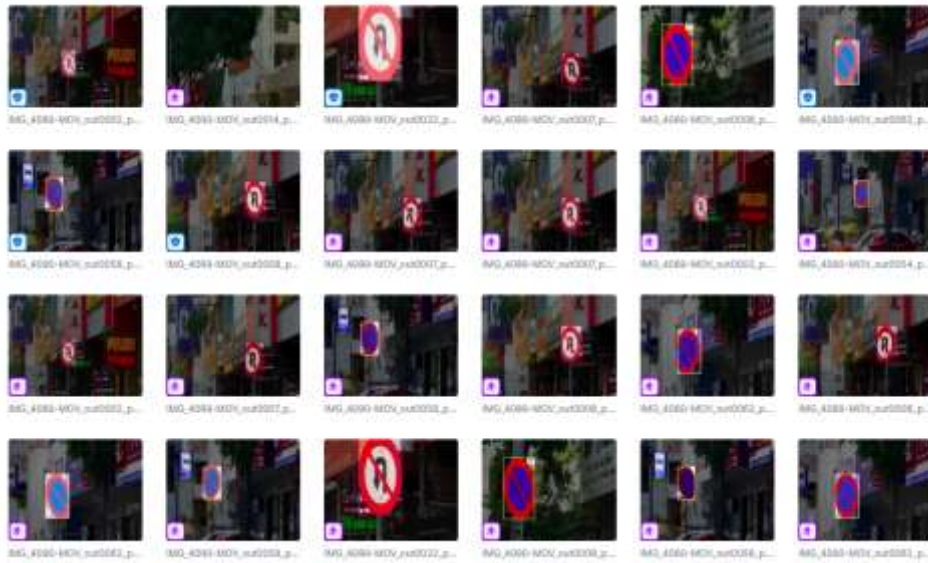
Ngoài ra, nhóm thực hiện cũng khảo sát các mô hình học sâu hiện đại như YOLO (You Only Look Once) và đánh giá các mô hình như YOLOv12, Faster R-CNN để xác định mô hình nào phù hợp với yêu cầu nhận dạng real-time, độ chính xác cao, và triển khai được trên nền web. Mục tiêu cuối cùng là xây dựng một hệ thống có khả năng phát hiện và phân loại biển báo từ ảnh tĩnh (người dùng tải lên) hoặc video (real-time webcam).

### 1.3.2. Thu thập và xử lý dữ liệu

Do đề tài mang tính mô phỏng, dữ liệu được sử dụng chủ yếu từ các bộ dữ liệu công khai Vietnam Traffic Sign. Bộ dữ liệu này cung cấp ảnh biển báo trong giao thông ở Việt Nam được gán nhãn sẵn với nhiều điều kiện môi trường, phù hợp cho huấn luyện mô hình học sâu.

Dữ liệu sau khi tải về sẽ được lọc bỏ ảnh lỗi, cân bằng giữa các lớp biển báo để tránh hiện tượng mất cân bằng trong quá trình học. Tiếp theo, dữ liệu được tiền xử lý như resize ảnh về kích thước phù hợp ( $640 \times 640$ ), chuẩn hóa pixel (scale về  $[0,1]$ ) và áp dụng các kỹ thuật tăng cường dữ liệu (data augmentation) như xoay, lật, điều chỉnh ánh sáng nhằm tăng tính đa dạng cho dữ liệu huấn luyện và cải thiện khả năng tổng quát của mô hình.

Quan trọng, các nhãn của bộ dữ liệu đang không tương thích với các bộ luật biển báo giao thông ở Việt Nam, nhóm đã xử lý bằng cách tìm hiểu và gán nhãn thủ công các biển báo này cho phù hợp.



*Hình 1.5. Hình ảnh minh họa bộ dữ liệu gốc Vietnam Traffic Sign*

### 1.3.3. Xây dựng hệ thống lưu trữ và xử lý

Thay vì xây dựng kho dữ liệu truyền thống như trong hệ thống nghiệp vụ, nhóm thực hiện sử dụng thư mục lưu trữ ảnh huấn luyện, validation và test theo định dạng chuẩn YOLO (thư mục chứa ảnh và file .txt nhãn tương ứng). Các mô hình YOLOv12 và một mô hình mạng nơ ron truyền thống khác được huấn luyện lần lượt trên cùng một tập dữ liệu để so sánh hiệu năng.

Sau huấn luyện, mô hình được đóng gói (export) sang định dạng .pt hoặc để triển khai trên website. Tại đây, người dùng có thể tải ảnh lên để hệ thống nhận dạng biển báo và hiển thị bounding box kèm nhãn và độ chính xác. Ngoài ra, hệ thống cũng thống kê kết quả theo lớp biển báo, vẽ biểu đồ cột hoặc tròn trực quan hóa số lượng và độ tin cậy của các dự đoán. Phần lưu trữ chỉ cần đảm bảo phân vùng thư mục hợp lý, dễ dàng truy xuất khi người dùng tương tác với hệ thống.

### 1.3.4. Thiết kế mô hình khai phá

Ở giai đoạn này, nhóm sẽ so sánh ba kiến trúc mạng YOLO, Faster R-CNN dựa trên tiêu chí độ chính xác  $\text{map}@0.5$ , tốc độ xử lý (FPS) và kích thước mô hình (số tham số). Mỗi mô hình được điều chỉnh cấu hình đầu vào (input size), backbone (Darknet, CSPDarknet, hoặc các biến thể EfficientNet), và head

(detection head với các anchor box phù hợp) để tối ưu cho bài toán biên báo giao thông.

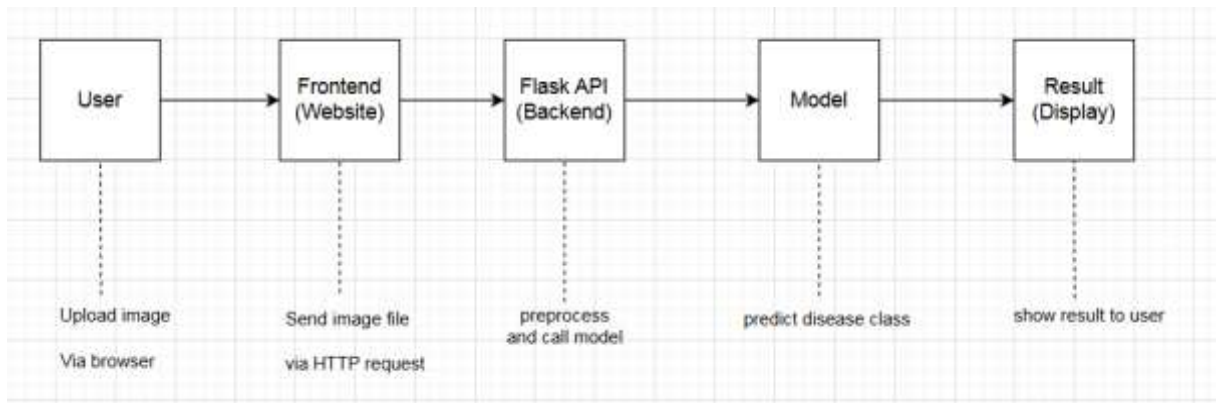
Đồng thời, nhóm cũng xem xét các kỹ thuật cải tiến như tập trung vào multi-scale feature, SPP (Spatial Pyramid Pooling) hoặc PANet để tăng khả năng phát hiện các biên báo nhỏ và biến dạng. Kết quả lựa chọn mô hình cuối cùng dựa trên sự cân bằng giữa độ chính xác và khả năng triển khai real-time trên website.

### **1.3.5. Huấn luyện, tinh chỉnh các tham số**

Mô hình sau khi thiết kế sẽ được huấn luyện trên tập dữ liệu training (khoảng 70% ảnh), với validation (20%) để điều chỉnh siêu tham số như learning rate, batch size, số epoch. Kỹ thuật fine-tuning được áp dụng bằng cách khởi tạo weights từ phiên bản tiền huấn luyện trên COCO hoặc ImageNet, sau đó chỉ điều chỉnh toàn bộ tầng cuối cùng hoặc toàn mô hình tùy độ lớn dữ liệu. Trong quá trình huấn luyện, nhóm theo dõi loss function (objectness loss, classification loss, localization loss) và các chỉ số precision, recall trên validation set để tránh overfitting. Cuối cùng, hiệu năng mô hình được đánh giá trên test set (10% ảnh) thông qua mAP, FPS thực tế và confusion matrix để xác định khả năng phân biệt giữa các lớp biên báo.

### **1.3.6. Triển khai, giám sát, bảo trì**

Sau khi đạt hiệu năng mong muốn, mô hình được xuất ra định dạng thích hợp và tích hợp vào backend của website sử dụng Flask hoặc FastAPI. Phần frontend sử dụng JavaScript để lấy luồng video hoặc ảnh người dùng upload, gọi API và hiển thị bounding box cùng độ chính xác. Hệ thống giám sát bao gồm logging thời gian phản hồi, tỉ lệ lỗi dự đoán và tải CPU/GPU để phát hiện sớm sự cố hoặc suy giảm hiệu năng. Định kỳ, nhóm sẽ thu thập lại ảnh từ người dùng thực và đánh giá lại mô hình—thực hiện retraining nếu phát sinh lớp biên mới hoặc dữ liệu drift. Quy trình bảo trì cũng bao gồm cập nhật phiên bản mô hình, backup weights và kiểm thử tự động (unit test, integration test) cho API.



Hình 1.6. Sơ đồ pipeline triển khai website – API – mô hình

#### 1.4. Lĩnh vực ứng dụng

Kho dữ liệu và khai phá dữ liệu đang đóng vai trò then chốt trong việc nâng cao hiệu quả hoạt động và ra quyết định ở rất nhiều lĩnh vực. Trong ngành tài chính – ngân hàng, các ngân hàng và tổ chức tín dụng lưu trữ khối lượng khổng lồ thông tin giao dịch, điểm tín dụng và hồ sơ khách hàng để xây dựng các mô hình đánh giá rủi ro, phát hiện gian lận và tối ưu hoá các chương trình ưu đãi cá nhân hóa. Trong y tế, việc tích hợp dữ liệu bệnh án điện tử, kết quả xét nghiệm và hình ảnh y khoa với thuật toán khai phá cho phép phát hiện sớm bệnh lý, hỗ trợ bác sĩ trong chẩn đoán và lập kế hoạch điều trị cá nhân hóa.

Ở lĩnh vực bán lẻ, doanh nghiệp khai thác dữ liệu lịch sử bán hàng, hành vi khách hàng và chuỗi cung ứng để dự báo nhu cầu, tối ưu tồn kho và triển khai các chiến dịch marketing tự động, từ đó tăng doanh thu và cải thiện trải nghiệm mua sắm. Trong giao thông thông minh, kho dữ liệu tập trung thông tin từ camera, cảm biến và hệ thống giao thông công cộng giúp phân tích luồng xe, phát hiện sự cố và hỗ trợ các thuật toán học sâu nhận dạng biển báo, tín hiệu đèn hoặc vật cản trên đường, góp phần nâng cao an toàn và hiệu quả vận hành. Ngoài ra, các ngành như viễn thông sử dụng khai phá dữ liệu để giám sát chất lượng dịch vụ và tối ưu mạng lưới, sản xuất ứng dụng dự đoán bảo trì thiết bị nhằm giảm thời gian chết của máy móc, và marketing số tận dụng phân tích hành vi trực tuyến để cá nhân hoá nội dung và tối đa hoá tỷ lệ chuyển đổi.

#### 1.5. Hướng nghiên cứu

Trong giai đoạn tiếp theo, một hướng nghiên cứu quan trọng là mở rộng và đa dạng hóa tập dữ liệu huấn luyện nhằm cải thiện khả năng tổng quát của mô hình. Việc thu thập thêm ảnh biến báo trong các điều kiện thực tế khó khăn—như ánh sáng yếu, lóa nắng, mưa hay tuyết—sẽ giúp mô hình học sâu nhận diện chính xác hơn trong môi trường ngoài phòng thí nghiệm. Đồng thời, nghiên cứu phương pháp tự động gán nhãn (auto-annotation) và khai thác dữ liệu vô nhãn (self-supervised learning) có thể giảm thiểu công sức gán nhãn thủ công và tận dụng tốt hơn các nguồn dữ liệu mới.

Song song với việc mở rộng dữ liệu, việc tích hợp kỹ thuật học chuyển giao (transfer learning) và tinh chỉnh mạng (fine-tuning) từ các mô hình lớn đã được huấn luyện trên tập dữ liệu chung như COCO hay ImageNet sẽ giúp đẩy nhanh quá trình huấn luyện và nâng cao độ chính xác. Một hướng thú vị khác là áp dụng kiến trúc lightweight như YOLO-Nano hoặc MobileNet-SSD để tối ưu hóa cho thiết bị biên (edge devices), cho phép triển khai hệ thống nhận dạng trực tiếp trên thiết bị di động hoặc camera thông minh mà không phụ thuộc hoàn toàn vào server.

Về khía cạnh mô hình, nghiên cứu sâu hơn về cơ chế chú ý (attention mechanisms) và mạng hai chiều (bi-directional networks) có thể giúp mô hình tập trung tốt hơn vào các vùng biến báo nhỏ hoặc che khuất. Ngoài ra, việc kết hợp thêm mô-đun giải thích (Explainable AI) sẽ cung cấp khả năng minh bạch hoá quyết định của mô hình, giúp người phát triển và người dùng cuối hiểu rõ hơn vì sao một đối tượng được phân loại là biến báo nào và tại sao bounding box được định vị ở vị trí nhất định.

Cuối cùng, mở rộng hệ thống thử nghiệm sang môi trường trực tuyến với khả năng học liên tục (continual learning) sẽ cho phép mô hình tự động cập nhật khi có dữ liệu mới hoặc khi xuất hiện các loại biến báo chưa từng gặp. Kết hợp phân tích hiệu năng thời gian thực và cơ chế giám sát drift dữ liệu sẽ đảm bảo hệ thống luôn duy trì độ chính xác và ổn định cao trong suốt vòng đời triển khai. Những hướng nghiên cứu này không chỉ nâng cao chất lượng nhận dạng biến báo

giao thông mà còn mở ra cơ hội áp dụng cho các bài toán vision khác trong giao thông thông minh và robot tự hành.

### **1.6. Tổng kết chương**

Qua chương I, vai trò then chốt của kho dữ liệu và phương pháp khai phá dữ liệu trong quá trình chuyển đổi dữ liệu thô thành tri thức giá trị đã được làm rõ. Kho dữ liệu đảm bảo việc lưu trữ tập trung, chuẩn hóa và bảo toàn lịch sử thay đổi, trong khi các kỹ thuật khai phá như học có giám sát, không giám sát hay tăng cường cho phép tự động phát hiện mẫu và quy luật ẩn trong khối lượng lớn thông tin. Sự kết hợp giữa hai thành phần này không chỉ nâng cao tốc độ và độ chính xác của các phân tích mà còn tối ưu hóa quy trình ra quyết định, giảm thiểu sai số và chi phí vận hành cho tổ chức.

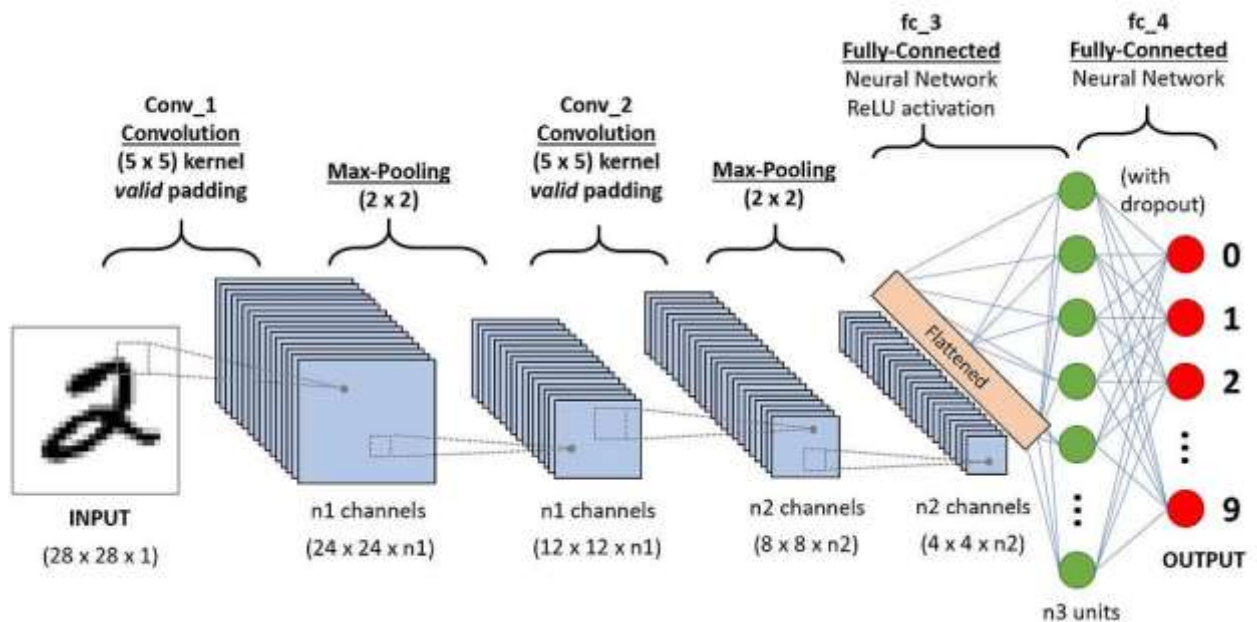
Ứng dụng vào đề tài “Nhận dạng biển báo giao thông bằng mạng nơ-ron”, toàn bộ quy trình từ xây dựng hệ thống lưu trữ dữ liệu ảnh và metadata, tiền xử lý, lựa chọn mô hình YOLO, Faster R-CNN, CNN phù hợp, đến huấn luyện, triển khai và giám sát đều dựa trên hạ tầng kho dữ liệu và kỹ thuật khai phá đã trình bày. Việc áp dụng mạng nơ-ron tích chập trên nền tảng dữ liệu mô phỏng công khai giúp đánh giá hiệu năng một cách hệ thống, đồng thời đảm bảo khả năng mở rộng và tích hợp dễ dàng trên website. Như vậy, chương tổng quan không chỉ làm nền tảng lý thuyết mà còn định hướng rõ lộ trình thực hiện cho các bước tiếp theo của luận văn.

## CHƯƠNG 2. CÁC PHƯƠNG PHÁP KHAI PHÁ DỮ LIỆU

### 2.1. Mạng nơ-ron tích chập

#### 2.1.1. Giới thiệu

Mạng nơ-ron tích chập (Convolutional Neural Network - CNN) là một loại mạng nơ-ron nhân tạo (Artificial Neural Network - ANN) được thiết kế đặc biệt để xử lý dữ liệu có cấu trúc dạng lưới, ví dụ như hình ảnh. CNN được lấy cảm hứng từ cấu trúc của vỏ não thị giác của động vật, nơi các tế bào thần kinh được sắp xếp theo một cấu trúc phân cấp để xử lý thông tin thị giác.



Hình 2.1. Mô hình mạng nơ-ron tích chập (Nguồn: Internet)

#### 2.1.2. Cấu trúc

Mạng nơ-ron tích chập thường bao gồm 3 lớp chính:

- **Lớp tích chập (Convolutional layer):** Thực hiện phép tích chập giữa bộ lọc (kernel) và dữ liệu đầu vào để trích xuất các đặc trưng cục bộ. Tầng tích chập (CONV) sử dụng các bộ lọc để thực hiện phép tích chập khi đưa chúng đi qua đầu vào II theo các chiều của nó. Các siêu tham số của các bộ lọc này bao gồm kích thước bộ lọc FF và độ trượt (stride) SS. Kết quả đầu ra OO được gọi là feature map hay activation map.

- **Lớp gộp (Pooling layer):** Tầng pooling (POOL) là một phép downsampling, thường được sử dụng sau tầng tích chập, giúp tăng tính bất biến không gian. Cụ thể, max pooling và average pooling là những dạng pooling đặc biệt, mà tương ứng là trong đó giá trị lớn nhất và giá trị trung bình được lấy ra.
- **Lớp kết nối đầy đủ (Fully connected layer):** Tầng kết nối đầy đủ (FC) nhận đầu vào là các dữ liệu đã được làm phẳng, mà mỗi đầu vào đó được kết nối đến tất cả nơ-ron. Trong mô hình mạng CNNs, các tầng kết nối đầy đủ thường được tìm thấy ở cuối mạng và được dùng để tối ưu hóa mục tiêu của mạng ví dụ như độ chính xác của lớp.

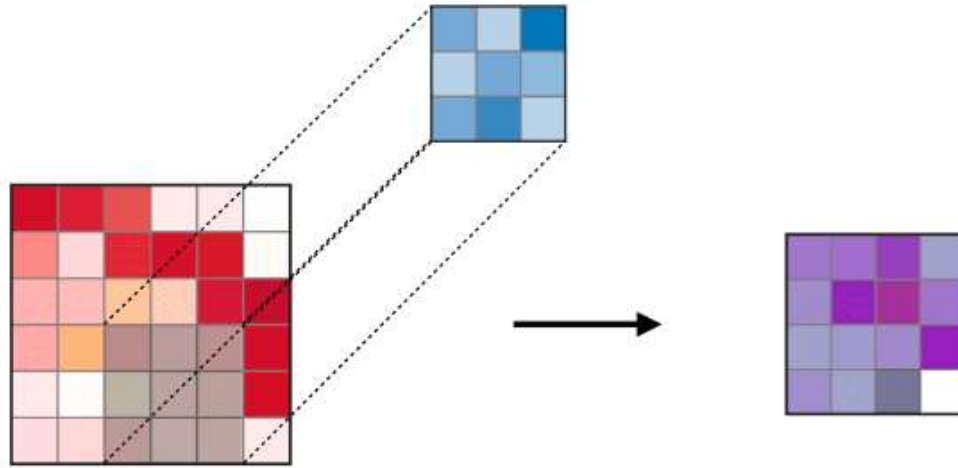


Hình 2.2. Mô tả cấu trúc mô hình CNN AlexNet (Nguồn: Internet)

### 2.1.2.1. Lớp tích chập (Convolutional Layer)

Lớp này là nơi thể hiện ý tưởng ban đầu của CNN. Thay vì kết nối toàn bộ điểm ảnh, layer này sẽ sử dụng một tập các bộ lọc (filters) có kích thước nhỏ so sánh với ảnh (thường là 5x5 hoặc 3x3) áp vào một vùng trong ảnh và tiến hành tính tích chập giữa bộ lọc và giá trị điểm ảnh trong vùng cục bộ đó. Bộ lọc sẽ lần

lượt được dịch chuyển theo một giá trị bước trượt (stride) chạy dọc theo ảnh và quét toàn bộ ảnh.



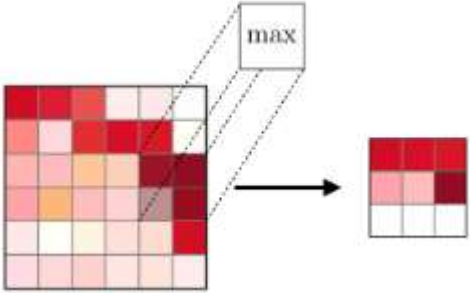
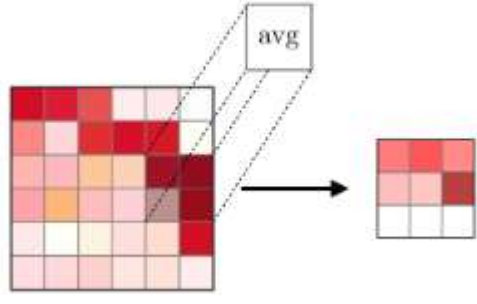
Hình 2.3. Hình ảnh mô tả lớp tích chập (Nguồn: Internet)

Như vậy, trong quá trình tích chập, filter sẽ quét qua toàn bộ ảnh, từng phần một và tạo ra một feature map mới, trích xuất các đặc trưng quan trọng từ ảnh gốc. Số lượng feature maps phụ thuộc vào số lượng filter trong lớp tích chập. Ví dụ, nếu có 10 filter thì sẽ tạo ra 10 feature maps. Những ảnh này sau đó được truyền vào các lớp tiếp theo của mô hình để tiếp tục xử lý. Ban đầu, các trọng số trong filter được khởi tạo ngẫu nhiên và sau đó được điều chỉnh trong quá trình huấn luyện, giúp mô hình học cách nhận diện các đặc trưng chính xác hơn từ dữ liệu. Một hình minh họa của quá trình tích chập với filter  $5 \times 5$  sẽ thể hiện rõ cách mà từng phần của bức ảnh gốc tương tác với filter để tạo ra feature map tương ứng.

#### 2.1.2.2. Lớp gộp (Polling Layer)

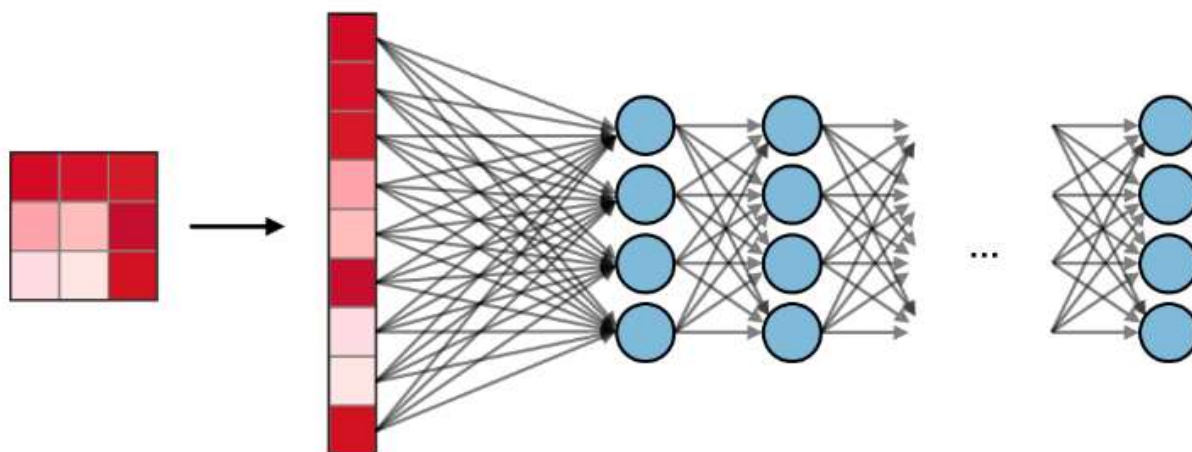
Layer này sử dụng một cửa sổ trượt để quét qua toàn bộ ảnh dữ liệu, mỗi lần trượt theo một bước trượt (stride) cho trước. Khác với layer Convolution, layer Pooling không tính tích chập mà tiến hành lấy mẫu (subsampling). Khi cửa sổ trượt trên ảnh, chỉ có một giá trị được xem là giá trị đại diện cho thông tin ảnh tại vùng đó (giá trị mẫu) được giữ lại. Các phương thức lấy phổ biến trong layer Pooling là MaxPooling (lấy giá trị lớn nhất) và Average Pooling (lấy giá trị trung bình).

Bảng 2.1. Bảng minh họa phương thức Max Pooling và Average Pooling

Kiểu	Max pooling	Average Pooling
Chức năng	Từng phép pooling chọn giá trị lớn nhất trong khu vực mà nó đang được áp dụng.	Từng phép pooling tính trung bình các giá trị trong khu vực mà nó đang được áp dụng.
Minh họa		
Nhận xét	<ul style="list-style-type: none"> <li>- Bảo toàn các đặc trưng đã phát hiện.</li> <li>- Được sử dụng thường xuyên.</li> </ul>	<ul style="list-style-type: none"> <li>- Giảm kích thước feature map.</li> <li>- Được sử dụng trong mạng LeNet.</li> </ul>

### 2.1.2.3. Lớp kết nối đầy đủ (Fully Connected)

Tầng kết nối đầy đủ (FC) nhận đầu vào là các dữ liệu đã được làm phẳng, mà mỗi đầu vào đó được kết nối đến tất cả nơ-ron. Trong mô hình mạng CNN, các tầng kết nối đầy đủ thường được tìm thấy ở cuối mạng và được dùng để tối ưu hóa mục tiêu của mạng ví dụ như độ chính xác của lớp.

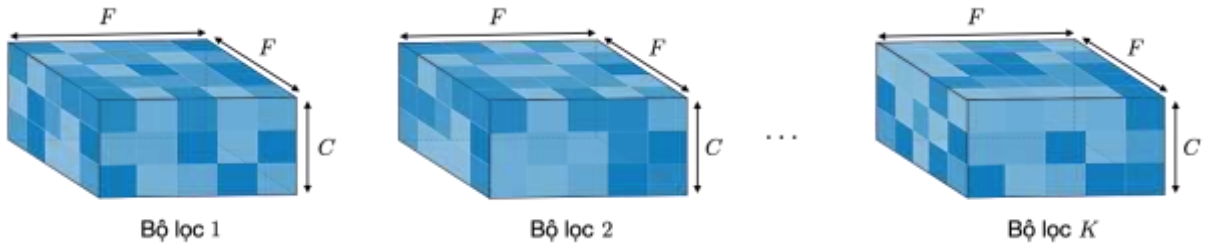


Hình 2.4. Hình ảnh mô phỏng lớp kết nối đầy đủ (Nguồn: Internet)

### 2.1.3. Siêu tham số

#### 2.1.3.1. Các siêu tham số của bộ lọc

**Các chiều của một bộ lọc:** Một bộ lọc kích thước  $F \times F$  áp dụng lên đầu vào chứa  $C$  kênh (channels) thì có kích thước tổng thể là  $F \times F \times C$  thực hiện phép tích chập trên đầu vào kích thước  $I \times I \times C$  và cho ra một feature map (hay còn gọi là activation map) có kích thước  $O \times O \times 1$ .



Hình 2.5. Các chiều của bộ lọc (Nguồn: Internet)

**Lưu ý:** Việc áp dụng  $K$  bộ lọc có kích thước  $F \times F$  cho ra một feature map có kích thước  $O \times O \times K$ .

**Stride:** Đối với phép tích chập hoặc phép pooling, độ trượt  $S$  ký hiệu số pixel mà cửa sổ sẽ di chuyển sau mỗi lần thực hiện phép tính.


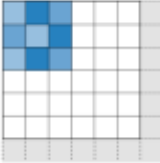
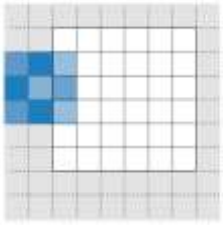


Hình 2.6. Minh họa độ trượt (Nguồn: Internet)

**Zero-padding:** Zero-padding là tên gọi của quá trình thêm  $P$  số không vào các biên của đầu vào. Giá trị này có thể được lựa chọn thủ công hoặc một cách tự động bằng một trong ba những phương pháp mô tả bên dưới.

Bảng 2.2. Bảng so sánh 3 phương pháp Zero-padding.

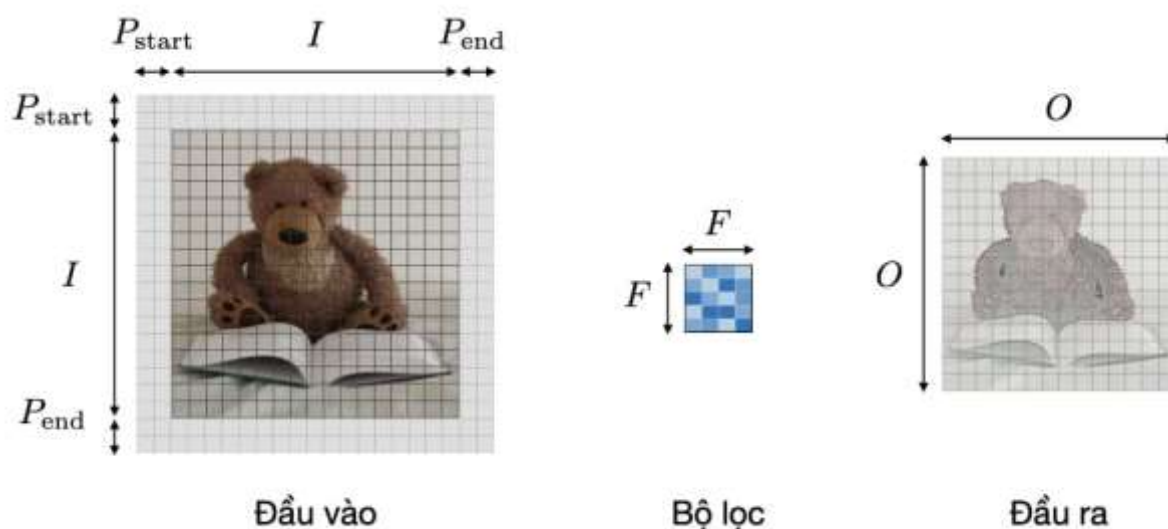
Phương pháp	Valid	Same	Full
Giá trị	$P = 0$	$P_{start} = \left\lceil \frac{S \left\lceil \frac{I}{S} \right\rceil - I + F - S}{2} \right\rceil$	$P_{start} \in \llbracket 0, F - 1 \rrbracket$ $P_{end} = F - 1$

		$P_{end} = \left\lfloor \frac{S \left\lceil \frac{I}{S} \right\rceil - I + F - S}{2} \right\rfloor$	
<b>Minh họa</b>			
<b>Mục đích</b>	<ul style="list-style-type: none"> <li>- Không sử dụng padding</li> <li>- Bỏ phép tích chập cuối nếu số chiều không khớp</li> </ul>	<ul style="list-style-type: none"> <li>- Sử dụng padding để làm cho feature map có kích thước <math>\left\lceil \frac{I}{S} \right\rceil</math>.</li> <li>- Kích thước đầu ra thuận lợi về mặt toán học.</li> <li>- Còn được gọi là “half” padding.</li> </ul>	<ul style="list-style-type: none"> <li>- Padding tối đa sao cho các phép tích chập có thể được sử dụng tại các rìa của đầu vào.</li> <li>- Bộ lọc “thấy” được đầu vào từ đầu đến cuối.</li> </ul>

### 2.1.3.2. Điều chỉnh siêu tham số

**Tính tương thích của tham số trong tầng tích chập:** Bằng cách ký hiệu  $I$  là độ dài kích thước đầu vào,  $F$  là độ dài của bộ lọc,  $P$  là số lượng zero-padding,  $S$  là độ trượt, ta có thể tính được độ dài  $O$  của feature map theo một chiều bằng công thức:

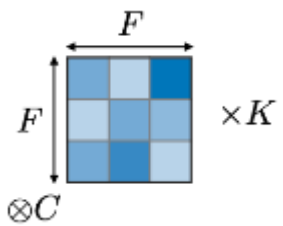
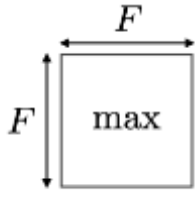
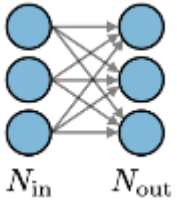
$$O = \frac{I - F + P_{start} + P_{end}}{S} + 1$$



Hình 2.7. Tính tương tích của tham số trong tầng tích chập (Nguồn: Internet)

**Hiểu về độ phức tạp của mô hình:** Để đánh giá độ phức tạp của một mô hình, cách hữu hiệu là xác định số tham số mà mô hình đó sẽ có. Trong một tầng của mạng neural tích chập, nó sẽ được tính toán như sau.

Bảng 2.3. Bảng tính toán độ phức tạp của mô hình

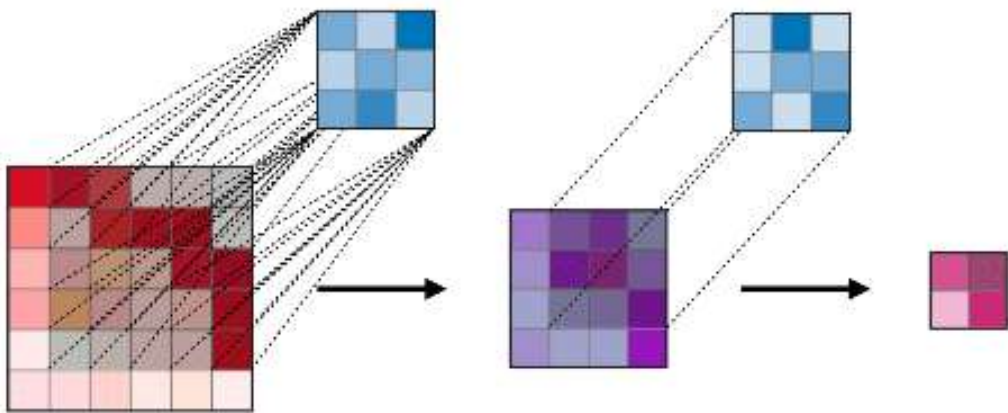
	CONV	POOL	FC
<b>Minh họa</b>			
<b>Kích thước đầu vào</b>	$I \times I \times C$	$I \times I \times C$	$N_{in}$
<b>Kích thước đầu ra</b>	$O \times O \times K$	$O \times O \times C$	$N_{out}$
<b>Số lượng tham số</b>	$(F \times F \times C + 1) \cdot K$	0	$(N_{in} + 1) \times N_{out}$
<b>Lưu ý</b>	- Một tham số bias với mỗi bộ lọc.	- Phép pooling được áp dụng	- Đầu vào được làm phẳng.

	<ul style="list-style-type: none"> <li>- Trong đa số trường hợp, <math>S &lt; F</math>.</li> <li>- Một lựa chọn phổ biến cho <math>K</math> là <math>2C</math>.</li> </ul>	<ul style="list-style-type: none"> <li>- lên từng kênh (channel-wise).</li> <li>- Trong đa số trường hợp, <math>S = F</math>.</li> </ul>	<ul style="list-style-type: none"> <li>- Mỗi nơ-ron có một tham số bias.</li> <li>- Số nơ-ron trong một tầng FC phụ thuộc vào ràng buộc kết cấu.</li> </ul>
--	--	--	---

**Trường thụ cảm:** Trường thụ cảm (receptive field) tại tầng  $k$  là vùng được ký hiệu  $R_k \times R_k$  của đầu vào mà những pixel của activation map thứ  $k$  có thể "nhìn thấy". Bằng cách gọi  $F_j$  là kích thước bộ lọc của tầng  $j$  và  $S_i$  là giá trị độ trượt của tầng  $i$  và để thuận tiện, ta mặc định  $S_0 = 1$ , trường thụ cảm của tầng  $k$  được tính toán bằng công thức:

$$R_k = 1 + \sum_{j=1}^k (F_j - 1) \prod_{i=0}^{j-1} S_i$$

Trong ví dụ bên dưới, ta có  $F_1 = F_2 = 3$  và  $S_1 = S_2 = 1$ , nên cho ra được  $R_2 = 1 + 2 \cdot 1 + 2 \cdot 1 = 5$ .

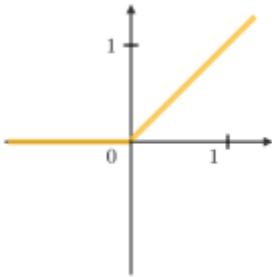
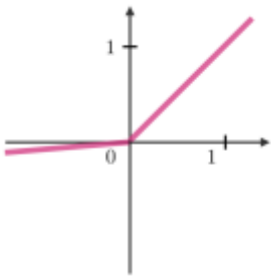
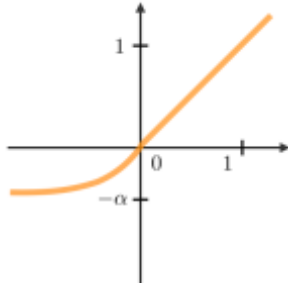


Hình 2.8. Ví dụ minh họa trường thụ cảm (Nguồn: Internet)

### 2.1.4. Các hàm kích hoạt thường gặp

- **Rectified Linear Unit:** Tầng rectified linear unit (ReLU) là một hàm kích hoạt  $g$  được sử dụng trên tất cả các thành phần. Mục đích của nó là tăng tính phi tuyến tính cho mạng. Những biến thể khác của ReLU được tổng hợp ở bảng dưới.

Bảng 2.4. Bảng tổng hợp biến thể của ReLU

ReLU	Leaky ReLU	ELU
$g(z) = \max(0, z)$	$g(z) = \max(\epsilon z, z)$ với $\epsilon \ll 1$	$g(z) = \max(\alpha(e^z - 1), z)$ với $\alpha \ll 1$
		
Độ phức tạp phi tuyến tính có thể thông dịch được về mặt sinh học	Gán vấn đề ReLU chết cho những giá trị âm	Khả vi tại mọi nơi

- **Softmax:** Bước softmax có thể được coi là một hàm logistic tổng quát lấy đầu vào là một vector chứa các giá trị  $x \in \mathbb{R}^n$  và cho ra là một vector gồm các xác suất  $p \in \mathbb{R}^n$  thông qua một hàm softmax ở cuối kiến trúc. Nó được định nghĩa như sau:

$$p = \begin{pmatrix} p_1 \\ \vdots \\ p_n \end{pmatrix}$$

$$\text{với } p_i = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}}$$

### 2.1.5. Ưu điểm

Mạng nơ-ron tích chập (CNN) có nhiều ưu điểm vượt trội, đặc biệt trong các bài toán xử lý ảnh và thị giác máy tính. Trước tiên, CNN có khả năng tự động trích xuất đặc trưng mà không cần can thiệp thủ công. Các bộ lọc tích chập giúp mô hình tự học được các đặc điểm quan trọng trong ảnh như đường viền, góc cạnh và kết cấu ở nhiều cấp độ khác nhau. Điều này giúp CNN hoạt động hiệu quả hơn so với các phương pháp truyền thống vốn phải dựa vào kỹ thuật trích xuất đặc trưng thủ công.

Một ưu điểm quan trọng khác là CNN có khả năng xử lý dữ liệu không gian tốt nhờ cấu trúc tầng tích chập. Mô hình tận dụng tính cục bộ và tính bất biến dịch chuyển, giúp nhận diện đối tượng ngay cả khi chúng bị thay đổi vị trí hoặc có nhiều nhiễu. Hơn nữa, nhờ cơ chế chia sẻ tham số trong các bộ lọc, CNN giảm đáng kể số lượng tham số cần học so với mạng nơ-ron truyền thống (fully connected network), giúp tiết kiệm bộ nhớ và cải thiện khả năng tổng quát hóa.

CNN cũng có hiệu suất cao trong các bài toán thực tế nhờ khả năng học sâu. Khi được huấn luyện trên tập dữ liệu lớn, mô hình có thể đạt độ chính xác cao trong các tác vụ như nhận diện khuôn mặt, phân loại hình ảnh và phát hiện vật thể. Ngoài ra, CNN có thể mở rộng dễ dàng bằng cách tăng số lượng tầng hoặc sử dụng các kiến trúc tiên tiến như ResNet, VGG, và EfficientNet để tối ưu hóa hiệu suất.

CNN đã được ứng dụng rộng rãi trong nhiều lĩnh vực như y tế, xe tự hành, an ninh và thị giác máy tính. Nhờ khả năng học sâu và tối ưu hóa đặc trưng tự động, CNN trở thành nền tảng cốt lõi trong nhiều công nghệ trí tuệ nhân tạo hiện đại, giúp cải thiện độ chính xác và hiệu suất của các hệ thống nhận diện và phân tích hình ảnh.

### 2.1.6. Nhược điểm

Mạng nơ-ron tích chập (CNN) có nhiều ưu điểm trong xử lý ảnh nhưng cũng tồn tại một số nhược điểm đáng kể. Trước tiên, CNN đòi hỏi tài nguyên tính

toán lớn, đặc biệt là GPU, để xử lý lượng lớn tham số và phép toán tích chập. Điều này làm tăng chi phí triển khai và gây khó khăn cho các thiết bị có cấu hình thấp. Ngoài ra, CNN cần một lượng dữ liệu lớn để hoạt động hiệu quả. Nếu tập dữ liệu không đủ phong phú, mô hình dễ bị overfitting, dẫn đến khả năng tổng quát hóa kém.

Một nhược điểm khác của CNN là nó chỉ học được các đặc trưng cục bộ thông qua các bộ lọc nhỏ, nên gặp khó khăn khi xử lý mối quan hệ toàn cục trong ảnh. Các mô hình như Vision Transformer đã được phát triển để khắc phục hạn chế này. Bên cạnh đó, CNN dễ bị tấn công bởi adversarial attacks, khi những thay đổi nhỏ không đáng kể với con người có thể khiến mô hình dự đoán sai hoàn toàn. Điều này gây rủi ro lớn trong các ứng dụng quan trọng như nhận diện khuôn mặt hay xe tự hành.

CNN cũng gặp khó khăn trong việc bảo toàn cấu trúc đối xứng và quan hệ hình học. Khi một vật thể bị xoay hoặc thay đổi vị trí, mô hình có thể không nhận diện chính xác nếu không được huấn luyện với đủ dữ liệu biến đổi. Ngoài ra, quá trình tinh chỉnh siêu tham số của CNN rất phức tạp, đòi hỏi nhiều thử nghiệm để tìm ra kiến trúc tối ưu.

Cuối cùng, CNN không tối ưu cho các bài toán yêu cầu xử lý theo thời gian thực. Các mô hình lớn có thể gây độ trễ đáng kể, làm giảm hiệu suất trong các ứng dụng như drone hay robot. Một số phương pháp tối ưu như pruning, quantization có thể giúp giảm kích thước mô hình nhưng lại ảnh hưởng đến độ chính xác. Vì vậy, tùy vào bài toán cụ thể, có thể cần kết hợp CNN với các mô hình khác để cải thiện hiệu suất.

### 2.1.7. Các kiến trúc CNN phổ biến

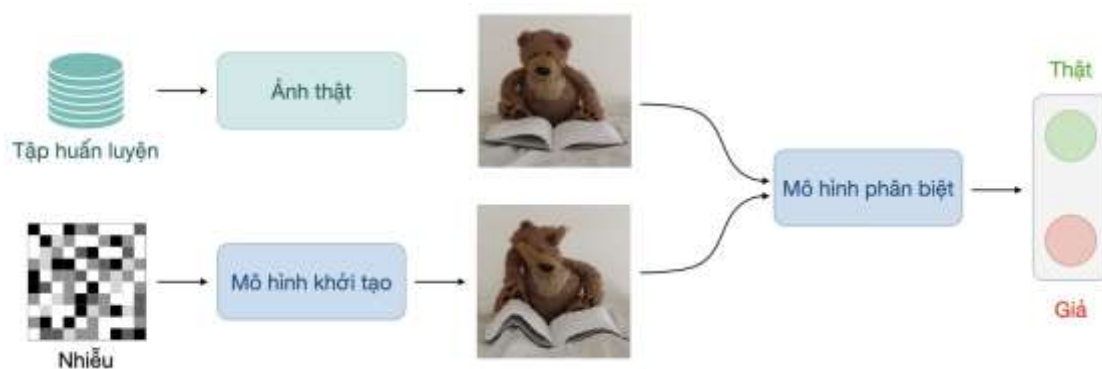
Một số kiến trúc CNN phổ biến được sử dụng trong bài toán nhận diện:

- **AlexNet:** Một trong những CNN đầu tiên đạt được kết quả vượt trội trong bài toán phân loại hình ảnh.

- **VGGNet**: Kiến trúc sâu hơn AlexNet, sử dụng nhiều lớp tích chập nhỏ để trích xuất đặc trưng.
- **GoogLeNet (Inception)**: Kiến trúc này sử dụng những inception module và hướng tới việc thử các tầng tích chập khác nhau để tăng hiệu suất thông qua sự đa dạng của các feature. Cụ thể, kiến trúc này sử dụng thủ thuật tầng tích chập  $1 \times 1$  để hạn chế gánh nặng tính toán.
- **ResNet**: Kiến trúc Residual Network (hay còn gọi là ResNet) sử dụng những khối residual (residual blocks) cùng với một lượng lớn các tầng để giảm lỗi huấn luyện. Những khối residual có những tính chất sau đây:

$$a^{[l+2]} = g(a^{[l]} + z^{[l+2]})$$

- **Generative Adversarial Network**: Generative adversarial networks, hay còn được gọi là GAN, là sự kết hợp giữa mô hình khởi tạo và mô hình phân biệt, khi mà mô hình khởi tạo cố gắng tạo ra hình ảnh đầu ra chân thực nhất, sau đó được đưa vào mô hình phân biệt, mà mục tiêu của nó là phân biệt giữa ảnh được tạo và ảnh thật.



Hình 2.9. Minh họa kiến trúc GAN (Nguồn: Internet)

**Lưu ý:** có nhiều loại GAN khác nhau bao gồm từ văn bản thành ảnh, sinh nhạc và tổ hợp.

## 2.2. Vision Transformer (ViT)

### 2.2.1. Giới thiệu

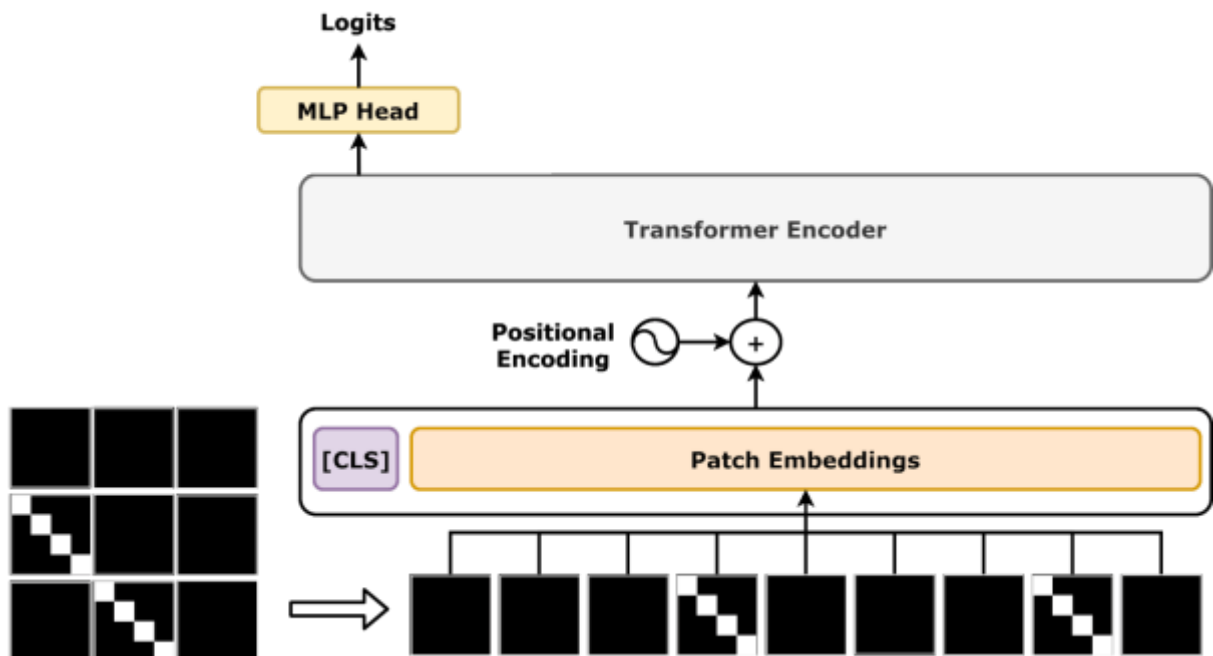
Vision Transformer, hay ViT, là một mô hình phân loại hình ảnh sử dụng kiến trúc giống Transformer áp dụng trên các mảng (patches) của hình ảnh. Một

hình ảnh được chia thành các mảng có kích thước cố định, mỗi mảng sau đó được nhúng tuyến tính, thêm các nhúng vị trí, và chuỗi vector kết quả được đưa vào một bộ mã hóa Transformer tiêu chuẩn. Để thực hiện phân loại, phương pháp tiêu chuẩn là thêm một “token phân loại” bổ sung có thể học được vào chuỗi vector này được sử dụng.

Kiến trúc của Vision Transformer (ViT) là một bước tiến đột phá trong việc áp dụng mô hình Transformer – vốn ban đầu được thiết kế cho xử lý ngôn ngữ tự nhiên (NLP) – vào các tác vụ thị giác máy tính, đặc biệt là phân loại hình ảnh.

### 2.2.2. Kiến trúc

Thay vì sử dụng các mạng nơ-ron tích chập (CNN) như cách tiếp cận truyền thống trong thị giác máy tính, ViT chia nhỏ hình ảnh thành các "mảnh" (patches) và xử lý chúng như một chuỗi tuần tự, tương tự cách Transformer xử lý các từ trong một câu. Sau đó, nó sử dụng cơ chế tự chú ý (self-attention) để tìm ra mối quan hệ giữa các mảnh này, từ đó hiểu được nội dung của toàn bộ hình ảnh.



Hình 2.10. Kiến trúc cơ bản của Vision Transformer (Nguồn: Internet)

#### 2.2.2.1. Chia hình ảnh thành patches

- Hình ảnh đầu vào (thường là ảnh RGB) được chia thành các mảnh nhỏ có kích thước cố định, ví dụ 16x16 pixel.

- Mỗi mảnh được "đuôi phẳng" (flattened) thành một vector. Giả sử ảnh có kích thước  $224 \times 224$  và mỗi mảnh là  $16 \times 16$ , ta sẽ có 196 mảnh ( $224/16 \times 224/16 = 14 \times 14 = 196$ ).
- Các vector này sau đó được đưa qua một lớp tuyến tính (linear projection) để chuyển thành các embedding có kích thước đồng nhất, gọi là patch embeddings.

#### 2.2.2.2. *Thêm thông tin vị trí*

- Vì Transformer không có khái niệm thứ tự tự nhiên như CNN, ViT thêm thông tin vị trí (positional embeddings) vào mỗi patch embedding. Điều này giúp mô hình biết được vị trí tương đối của từng mảnh trong hình ảnh gốc.
- Các embedding vị trí này có thể được cố định hoặc học được trong quá trình huấn luyện.

#### 2.2.2.3. *Token đặc biệt [CLS]*

Một token đặc biệt, gọi là [CLS], được thêm vào đầu chuỗi các patch embeddings. Token này đóng vai trò tổng hợp thông tin từ toàn bộ hình ảnh và thường được sử dụng để đưa ra dự đoán cuối cùng trong tác vụ phân loại.

#### 2.2.2.4. *Transformer Encoder*

Chuỗi gồm [CLS] và các patch embeddings được đưa vào một chuỗi các khối Transformer Encoder (giống như trong mô hình BERT).

Mỗi khối bao gồm:

- **Multi-Head Self-Attention:** Cơ chế chú ý đa đầu giúp mô hình tập trung vào các mối quan hệ giữa các mảnh khác nhau, ví dụ như liên kết giữa mắt và mũi trong ảnh khuôn mặt.
- **Feed-Forward Network (FFN):** Một mạng nơ-ron tiến thẳng áp dụng cho từng embedding.
- **Layer Normalization và Residual Connections:** Giúp ổn định và tăng hiệu quả huấn luyện.

Quá trình này được lặp lại qua nhiều tầng (layers), thường là 12 hoặc 24 tùy phiên bản.

#### 2.2.2.5. Đầu ra

- Sau khi qua các tầng Transformer, embedding của token [CLS] được lấy ra.
- Embedding này được đưa qua một lớp tuyến tính đơn giản (MLP head) để dự đoán lớp của hình ảnh.

#### 2.2.3. Ưu điểm

**Khả năng nắm bắt mối quan hệ toàn cục (Global Context):** ViT có thể hiểu được mối quan hệ giữa các vùng xa nhau trong hình ảnh ngay từ đầu, không cần phải xây dựng dần như CNN qua các tầng tích chập.

**Tính linh hoạt:** ViT có thể dễ dàng thích nghi với các kích thước đầu vào khác nhau (bằng cách thay đổi số lượng patches) và áp dụng cho nhiều tác vụ thị giác khác nhau (phân loại, phân đoạn, phát hiện đối tượng) mà không cần thay đổi cấu trúc cơ bản.

**Hiệu quả cao khi có dữ liệu lớn:** Khi được huấn luyện trên các tập dữ liệu khổng lồ (như JFT-300M), ViT thường vượt trội so với CNN về độ chính xác, đặc biệt trong các bài toán phân loại hình ảnh.

**Đơn giản hóa kiến trúc:** Không cần các phép toán tích chập phức tạp hay các tầng gộp (pooling), ViT dựa hoàn toàn vào Transformer, giúp dễ hiểu và triển khai hơn trong một số trường hợp.

#### 2.2.4. Nhược điểm

**Yêu cầu dữ liệu huấn luyện lớn:** ViT không có các giả định cục bộ như CNN (như tính bất biến với dịch chuyển), nên nó cần rất nhiều dữ liệu để học được các đặc trưng hình ảnh hữu ích. Nếu chỉ huấn luyện trên tập dữ liệu nhỏ (như CIFAR-10), hiệu suất thường kém hơn CNN.

**Tốn tài nguyên tính toán:** Cơ chế tự chú ý có độ phức tạp tính toán bậc hai ( $O(n^2)$ ) với số lượng patches, khiến ViT tiêu tốn nhiều bộ nhớ và thời gian huấn luyện hơn, đặc biệt với hình ảnh độ phân giải cao.

**Hiệu quả thấp với hình ảnh nhỏ:** Khi số lượng patches ít (ví dụ: ảnh kích thước nhỏ), ViT không tận dụng được hết sức mạnh của self-attention, dẫn đến hiệu suất không tối ưu.

**Thiếu tính bất biến cục bộ:** Không giống CNN, vốn tự nhiên có khả năng nhận diện đặc trưng bất kể vị trí (translation invariance), ViT cần học điều này từ dữ liệu, đôi khi làm giảm hiệu quả trong các tác vụ cần sự nhạy cảm với cấu trúc cục bộ.

### 2.2.5. Các kiến trúc ViT phổ biến

**ViT gốc (Vision Transformer):** Mô hình gốc do Google đề xuất, hoạt động tốt trên dữ liệu lớn. Các phiên bản phổ biến: ViT-B (Base), ViT-L (Large), ViT-H (Huge), khác nhau về số tầng và kích thước embedding.

**DeiT (Data-efficient ViT):** Cải tiến ViT để hoạt động tốt hơn với dữ liệu nhỏ hơn (như ImageNet-1k) bằng cách sử dụng kỹ thuật tăng cường dữ liệu (data augmentation) và một token "distillation" học từ mô hình thầy (teacher model). Tối ưu hóa việc huấn luyện với ít dữ liệu hơn.

**Swin Transformer:** Chia ảnh thành các cửa sổ (windows) nhỏ và chỉ áp dụng self-attention trong từng cửa sổ, sau đó dùng cơ chế "shifted windows" để kết nối thông tin giữa các cửa sổ. Sử dụng cơ chế attention phân cấp để cải thiện hiệu suất.

## CHƯƠNG 3. THỰC NGHIỆM

### 3.1. Kỹ thuật sử dụng

Trong dự án này, ba mô hình mạng nơ-ron khác nhau được sử dụng để nhận dạng biển báo giao thông:

- **YOLOv12 (You Only Look Once, phiên bản 12):** Mô hình phát hiện đối tượng một giai đoạn, được triển khai từ thư viện *Ultralytics*, nổi bật với tốc độ xử lý nhanh và độ chính xác cao, phù hợp cho bài toán nhận dạng biển báo theo thời gian thực.
- **Faster R-CNN với backbone ResNet50:** Mô hình phát hiện đối tượng hai giai đoạn, có khả năng định vị và phân loại chính xác đối tượng, đặc biệt hiệu quả với các trường hợp biển báo nhỏ hoặc bị che khuất.
- **CNN (Convolutional Neural Network) thuần:** Được huấn luyện cho tác vụ phân loại hình ảnh biển báo, dùng để so sánh hiệu năng với hai mô hình phát hiện đối tượng trên.

#### Công nghệ và thư viện hỗ trợ:

- **Python 3:** Ngôn ngữ lập trình chính được sử dụng xuyên suốt dự án. Python có hệ sinh thái phong phú với nhiều thư viện hỗ trợ cho xử lý ảnh, học sâu và trực quan hóa dữ liệu, phù hợp cho các bài toán thị giác máy tính.
- **NumPy, Pandas:**
  - + **NumPy** hỗ trợ thao tác mảng và tính toán ma trận nhanh chóng, đóng vai trò nền tảng cho hầu hết các phép toán khoa học dữ liệu.
  - + **Pandas** cung cấp cấu trúc dữ liệu DataFrame, giúp quản lý, xử lý và phân tích tập dữ liệu biển báo một cách thuận tiện.
- **OpenCV:** Thư viện xử lý ảnh mạnh mẽ, được sử dụng cho tiền xử lý dữ liệu như thay đổi kích thước, chuyển đổi không gian màu, làm mờ, lọc nhiễu, và **data augmentation** (xoay, lật, thay đổi độ sáng/tương phản) để tăng tính đa dạng của dữ liệu huấn luyện.

- **Matplotlib, Seaborn:**
  - + **Matplotlib** được dùng để vẽ biểu đồ, hiển thị kết quả nhận dạng, trực quan hóa hình ảnh và bounding box của mô hình.
  - + **Seaborn** hỗ trợ trực quan hóa thống kê như ma trận nhầm lẫn (confusion matrix) hoặc biểu đồ phân bố dữ liệu.
- **PyTorch, TorchVision:**
  - + **PyTorch** là framework chính để xây dựng, huấn luyện và tinh chỉnh mô hình mạng nơ-ron.
  - + **TorchVision** hỗ trợ các mô hình có sẵn (pre-trained models như ResNet50, Faster R-CNN) và các công cụ xử lý ảnh tích hợp, giúp tăng tốc quá trình thử nghiệm.
- **Ultralytics (YOLOv12):** Cung cấp pipeline huấn luyện, kiểm thử và suy luận nhanh chóng cho mô hình YOLOv12. Thư viện này tối ưu về tốc độ và hiệu quả, cho phép dễ dàng cấu hình tham số huấn luyện, quản lý tập dữ liệu, và trực quan hóa kết quả nhận dạng theo thời gian thực.
- **Scikit-learn:** Được sử dụng để tính toán và báo cáo các chỉ số đánh giá mô hình như **Accuracy, Precision, Recall, F1-score**, đồng thời hỗ trợ vẽ confusion matrix và các phân tích thống kê bổ sung.
- **OpenMM:** Hỗ trợ tính toán gia tốc GPU trong một số bước xử lý và huấn luyện, đặc biệt khi làm việc trên nền tảng Kaggle Notebook.

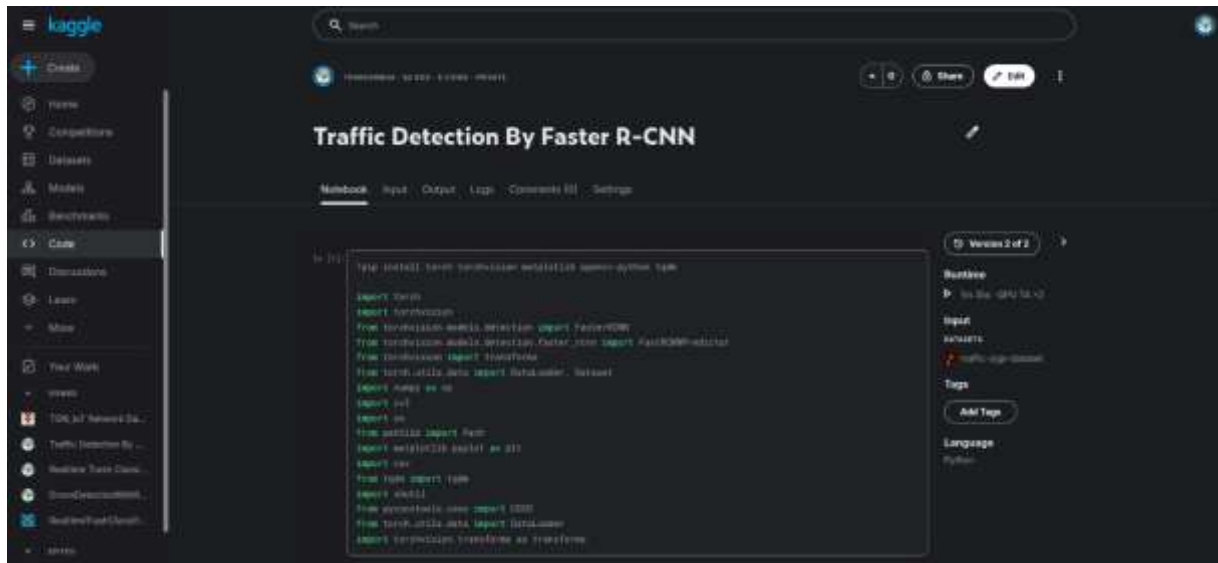
### 3.2. Môi trường thực nghiệm

Quá trình huấn luyện và đánh giá mô hình được tiến hành trên nền tảng **Kaggle**:

- Phần cứng: GPU NVIDIA Tesla T4 (16GB VRAM) giúp tăng tốc độ huấn luyện và suy luận.
- Phần mềm: Python 3.x, PyTorch, TorchVision, Ultralytics, NumPy, OpenCV, Matplotlib, Flask, HTML, CSS, Bootstrap.

- Hệ điều hành: Linux (môi trường mặc định của Kaggle/Colab), triển khai website trên máy chủ Flask cục bộ hoặc dịch vụ cloud.

Sau khi huấn luyện, các mô hình được lưu trữ dưới dạng tệp trọng số (.pt, .pth, .h5), tải vào Flask backend để phục vụ trực tiếp cho website. Giao diện web tương tác, hiển thị kết quả dự đoán theo thời gian thực hoặc từ ảnh tải lên mà không yêu cầu cài đặt phức tạp phía người dùng.



Hình 3.1. Hình ảnh huấn luyện mô hình trên Kaggle (Sử dụng GPU T4x2)

### 3.3. Dữ liệu thực nghiệm

#### 3.3.1. Nguồn gốc và đặc điểm của bộ dữ liệu

##### 3.3.1.1. Nguồn gốc của bộ dữ liệu

Bộ dữ liệu Vietnam Traffic Sign Dataset được lấy từ roboflow – một bộ dữ liệu chuẩn cho bài toán nhận dạng biển báo giao thông. Bộ dữ liệu gồm ảnh biển báo thuộc nhiều loại khác nhau, chụp trong nhiều điều kiện khác nhau.

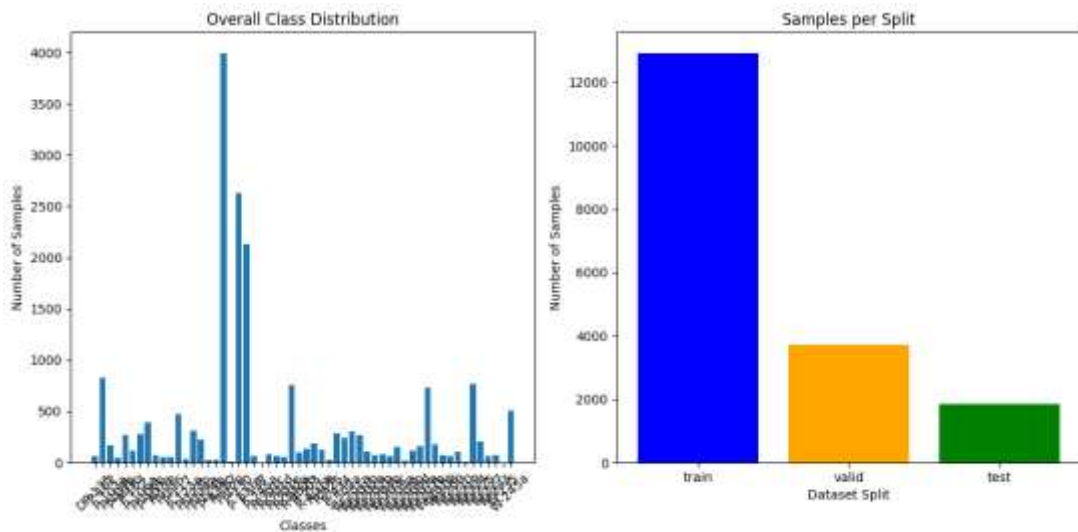
##### 3.3.1.2. Đặc điểm của bộ dữ liệu

Số lượng lớp: 56 lớp (tương ứng với 56 loại biển báo khác nhau).

Tổng số ảnh được nhóm phân chia với tỷ lệ (80/10/10)

- Tập huấn luyện (Train): 9800 ảnh
- Tập kiểm định (Valid): 1244 ảnh

- Tập kiểm tra (Test): 1053 ảnh



Hình 3.2. Biểu đồ số lượng classes có trong dữ liệu

Điều kiện ảnh: Bộ dữ liệu bao gồm các hình ảnh với nhiều điều kiện khác nhau, giúp mô hình có thể tổng quát hóa tốt hơn trong các tình huống thực tế. Các điều kiện ảnh bao gồm ảnh rõ nét, ảnh bị mờ, ảnh có ánh sáng kém, và ảnh chụp từ các khoảng cách khác nhau. Điều này giúp tăng cường tính mạnh mẽ của mô hình khi triển khai trong môi trường thực tế đa dạng.

### 3.3.2. Tiền xử lý dữ liệu

Dữ liệu được tiền xử lý để chuẩn bị cho quá trình huấn luyện mô hình. Một số bước tiền xử lý đã được thực hiện bao gồm:

#### 3.3.2.1. Đồng nhất kích thước ảnh

Trong bài toán phân loại và phát hiện biến báo giao thông tất cả ảnh đều được resize về kích thước chuẩn 640×640 pixel. Kích thước này phù hợp với cấu hình được khuyến nghị của các mô hình hiện đại như YOLOv12, đồng thời tương thích tốt với các kiến trúc backbone phổ biến như ResNet50 (trong Faster R-CNN) và CNN.



Hình 3.3. Hình ảnh minh họa kỹ thuật chuẩn hoá kích thước

### 3.3.2.2. Tăng cường dữ liệu

Kỹ thuật này được sử dụng để tăng số lượng dữ liệu huấn luyện bằng cách tạo ra các biến thể mới của các hình ảnh hiện có (ví dụ: xoay, lật, thay đổi độ sáng, v.v.). Điều này giúp mô hình học được các đặc trưng mạnh mẽ hơn, tăng khả năng tổng quát hóa và tránh hiện tượng quá khớp (overfitting)

**Làm mờ và lọc nhiễu:** Ảnh chụp thực tế từ camera, đặc biệt trong điều kiện ngoài trời hoặc sử dụng thiết bị giá rẻ, thường gặp hiện tượng nhiễu ảnh do nhiều yếu tố như ánh sáng yếu, bóng đổ, rung tay hoặc chất lượng ống kính không đảm bảo. Những nhiễu này biểu hiện qua các chi tiết không mong muốn như đốm sáng, vết mờ, sọc nhiễu hay vết xước, có thể ảnh hưởng tiêu cực đến quá trình trích xuất đặc trưng của mô hình học sâu, khiến mô hình bị "xao nhãng" và suy giảm hiệu quả nhận diện vật thể. Gaussian Blur là một phương pháp phổ biến, giúp làm mờ nhẹ hình ảnh bằng cách làm trơn các pixel trong phạm vi lân cận theo phân phối Gauss. Kỹ thuật này giúp giảm nhiễu ngẫu nhiên và làm nổi bật các vật thể có kích thước lớn.



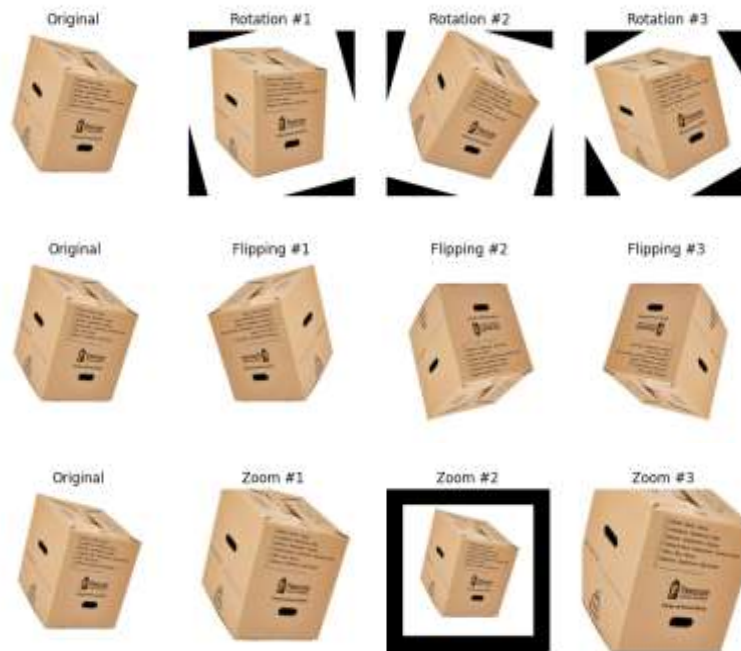
Hình 3.4. Hình ảnh minh họa kỹ thuật làm mờ, lọc nhiễu

**Tự động điều chỉnh độ tương phản:** Sự trùng lặp về màu sắc khiến mô hình khó phát hiện chính xác vật thể hoặc xác định đúng vùng chứa vật thể, đặc biệt là trong các mô hình phát hiện theo vùng như Faster R-CNN hoặc DETR. Để cải thiện điều này, điều chỉnh độ tương phản hình ảnh là một bước tiền xử lý quan trọng nhằm làm nổi bật vật thể và cải thiện khả năng học của mô hình.



Hình 3.5. Hình ảnh minh họa kỹ thuật cân bằng tương phản

**Xoay, lật, phóng to/ thu nhỏ ảnh:** Mục tiêu chính là giúp mô hình có khả năng nhận diện chính xác vật thể dù chúng bị xoay nhẹ trong quá trình chụp, chẳng hạn như do góc đặt camera lệch hoặc vật thể không đứng thẳng, mở rộng không gian ảnh theo hướng đối xứng, từ đó cải thiện khả năng của mô hình trong việc nhận diện vật thể ngay cả khi chúng xuất hiện ở các phía khác nhau trong ảnh, mô phỏng các trường hợp máy ảnh chụp gần hoặc xa đối tượng, từ đó cải thiện khả năng nhận diện của mô hình với nhiều tỷ lệ vật thể khác nhau.



Hình 3.6. Hình ảnh minh họa kỹ thuật xoay, xén, zoom ảnh

### 3.4. YOLOv12

#### 3.4.1. Tổng quan về mô hình

##### 3.4.1.1. Nguyên lý hoạt động

YOLOv12 hoạt động theo cơ chế “*You Only Look Once*”, xử lý toàn bộ ảnh trong một lượt để vừa dự đoán vị trí vừa phân loại đối tượng, thay vì tách thành hai giai đoạn như các mô hình truyền thống. Ảnh được chia thành lưới  $S \times S$ , mỗi ô chịu trách nhiệm phát hiện đối tượng có tâm nằm trong đó. Mỗi ô dự đoán nhiều bounding box với các thông số: tọa độ  $(x, y)$ , kích thước  $(w, h)$ , độ tin cậy (confidence), và xác suất lớp.

Kết quả dự đoán sau đó được lọc bằng **ngưỡng confidence** và **Non-Maximum Suppression (NMS)** để loại bỏ các hộp trùng lặp, đảm bảo mỗi đối tượng chỉ được phát hiện một lần chính xác nhất.

##### 3.4.1.2. Kiến trúc mô hình

YOLOv12 gồm ba thành phần chính:

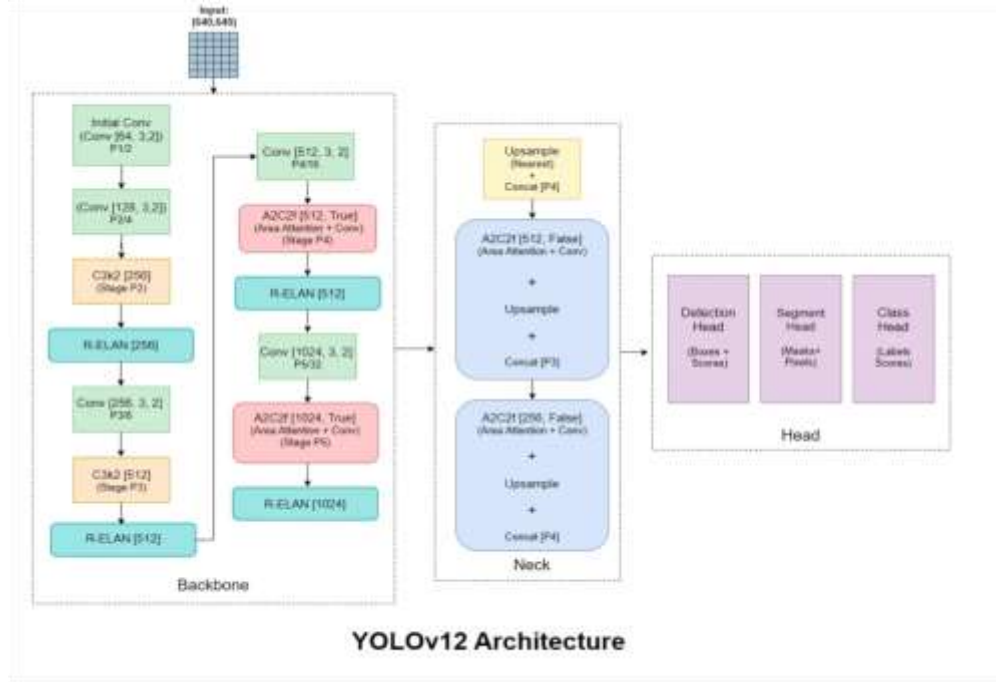
- **Backbone:** Trích xuất đặc trưng từ ảnh đầu vào, có thể dùng CSPDarkNet, EfficientRep, hay PP-YOLOE, học từ đặc trưng cơ bản đến phức tạp.
- **Neck:** Tổng hợp và truyền đặc trưng ở nhiều cấp độ, thường dùng PANet hoặc BiFPN, giúp phát hiện tốt các đối tượng ở nhiều kích thước.
- **Head:** Dự đoán bounding box, độ tin cậy và xác suất lớp. YOLOv12 áp dụng cơ chế **anchor-free**, giảm độ phức tạp và tăng tính linh hoạt so với các phiên bản cũ.

#### 3.4.1.3. *Điểm cải tiến*

YOLOv12 tối ưu giữa tốc độ và độ chính xác nhờ:

- Loss function cải tiến
- Data augmentation đa dạng
- Tích hợp attention, tối ưu hóa mạng
- Huấn luyện trên bộ dữ liệu lớn, giúp tăng khả năng khái quát hóa

**Kết quả:** YOLOv12 cho phép nhận diện và định vị đối tượng trong **thời gian thực** với hiệu suất cao, ngay cả trên thiết bị có tài nguyên hạn chế. Đây là một giải pháp mạnh mẽ, mở ra nhiều ứng dụng trong các hệ thống yêu cầu tốc độ và độ chính xác cao.



Hình 3.7. Kiến trúc mô hình YOLOv12

### 3.4.2. Tham số huấn luyện

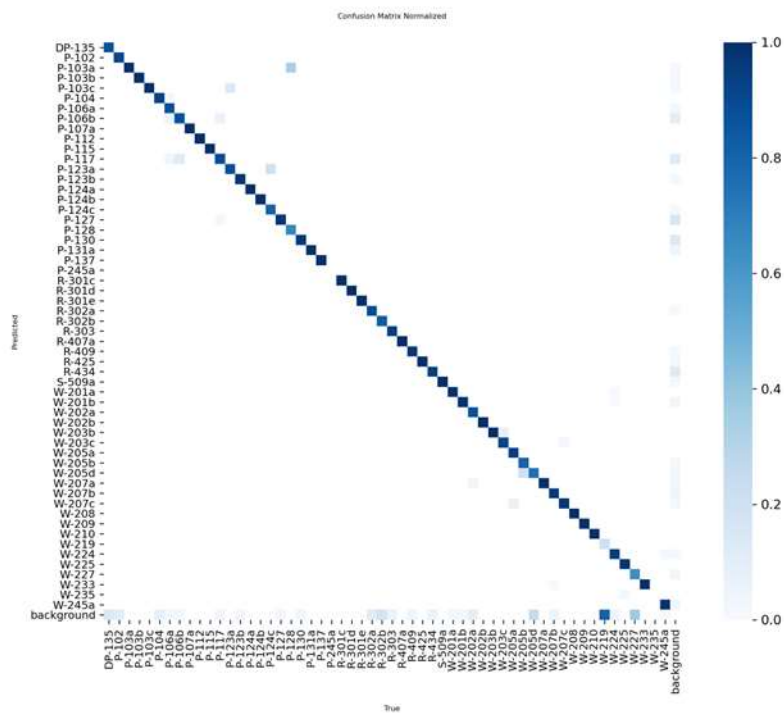
Các tham số huấn luyện đóng vai trò quan trọng trong việc định hình quá trình học của mô hình. Một số tham số điển hình khi huấn luyện YOLOv12 bao gồm:

- Số epoch: Số lần toàn bộ tập dữ liệu huấn luyện được truyền qua mạng. Một số lượng epoch đủ lớn là cần thiết để mô hình hội tụ.
- Kích thước batch (Batch Size): Số lượng mẫu dữ liệu được xử lý trong một lần lặp huấn luyện. Kích thước batch lớn hơn có thể tận dụng tốt hơn GPU nhưng đòi hỏi nhiều bộ nhớ hơn.
- Tốc độ học (Learning Rate): Tham số điều khiển kích thước bước mà mô hình điều chỉnh trọng số của nó. Việc lựa chọn tốc độ học phù hợp là rất quan trọng để tránh quá khớp hoặc hội tụ chậm.
- Hàm mất mát (Loss Function): YOLOv8 sử dụng sự kết hợp của các hàm mất mát cho phân loại, hộp giới hạn và độ tin cậy của đối tượng.
- Bộ tối ưu hóa (Optimizer): Thường là Adam hoặc SGD, được sử dụng để cập nhật trọng số của mạng dựa trên độ dốc của hàm mất mát.

Bảng 3.1. Bảng tham số huấn luyện của YOLOv12

Tham số	Giá trị
epoch	50
Batch size	12
Learning rate	Ban đầu: 0.001 Huấn luyện: 0.00001
Optimizer	ADAM
Image size	(640x640)

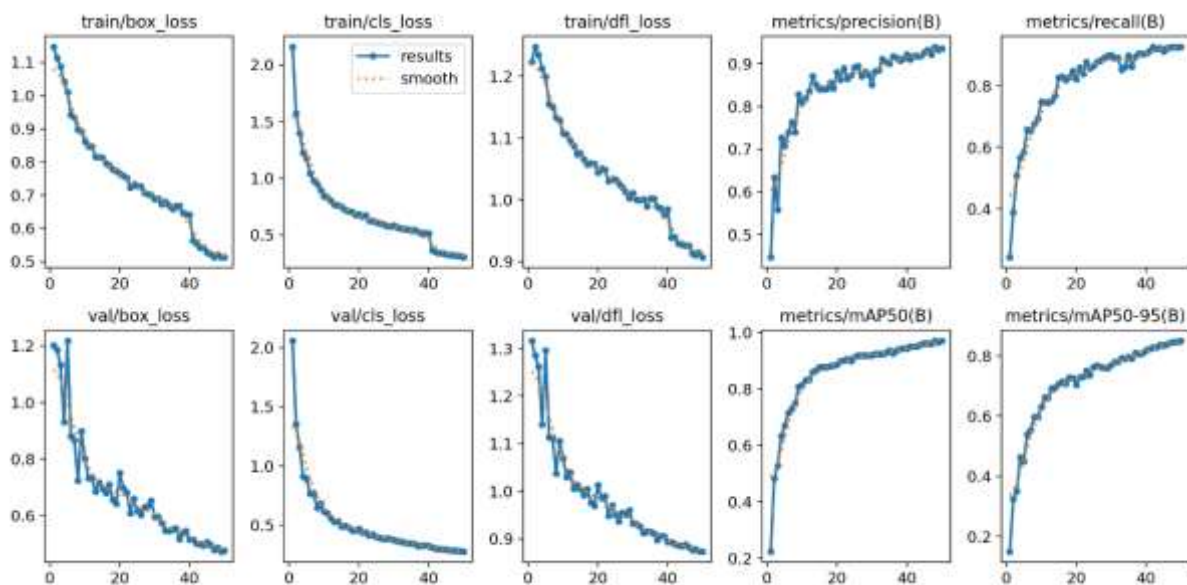
### 3.4.3. Kết quả thực nghiệm



Hình 3.8. Hình ảnh Confusion Matrix của YOLOv12

Epoch	GPU mem	box_loss	cls_loss	dfl_loss	Instances	Size			
45/59	11.6G	0.5276	0.3299	0.9258	5	688: 100%	563/563	[05:34:00:00, 1.68it/s]	
Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100%	161/161	[00:41:00:00, 3.80it/s]	
all	1922	3727	0.926	0.911	0.962	0.835			
Epoch	GPU mem	box_loss	cls_loss	dfl_loss	Instances	Size			
46/59	11.6G	0.5221	0.3211	0.9297	6	352: 100%	563/563	[05:40:00:00, 1.85it/s]	
Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100%	161/161	[00:41:00:00, 3.86it/s]	
all	1922	3727	0.932	0.922	0.961	0.838			
Epoch	GPU mem	box_loss	cls_loss	dfl_loss	Instances	Size			
47/59	11.7G	0.5131	0.3216	0.9128	4	736: 100%	563/563	[05:29:00:00, 1.71it/s]	
Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100%	161/161	[00:41:00:00, 3.87it/s]	
all	1922	3727	0.922	0.924	0.96	0.844			
Epoch	GPU mem	box_loss	cls_loss	dfl_loss	Instances	Size			
48/59	11.6G	0.5202	0.319	0.9184	3	544: 100%	563/563	[05:39:00:00, 1.66it/s]	
Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100%	161/161	[00:41:00:00, 3.87it/s]	
all	1922	3727	0.94	0.927	0.97	0.846			
Epoch	GPU mem	box_loss	cls_loss	dfl_loss	Instances	Size			
49/59	11.7G	0.5129	0.3133	0.914	11	416: 100%	563/563	[05:45:00:00, 1.63it/s]	
Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100%	161/161	[00:41:00:00, 3.87it/s]	
all	1922	3727	0.933	0.926	0.964	0.847			
Epoch	GPU mem	box_loss	cls_loss	dfl_loss	Instances	Size			
50/59	11.6G	0.5136	0.3027	0.9069	5	672: 100%	563/563	[05:38:00:00, 1.66it/s]	
Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100%	161/161	[00:41:00:00, 3.87it/s]	
all	1922	3727	0.936	0.927	0.960	0.85			

Hình 3.9. Kết quả tổng quan của YOLOv12



Hình 3.10. Hiệu suất huấn luyện của mô hình YOLOv12

### 3.4.4. Đánh giá hiệu suất mô hình

Bảng 3.2. Đánh giá hiệu suất mô hình YOLOv12

Ưu điểm	Nhược điểm
Cơ chế "one-stage" xử lý toàn bộ ảnh chỉ trong một lần, đạt tốc độ rất cao (từ vài chục đến hàng trăm FPS), phù hợp cho các	Hiệu suất giảm đáng kể khi phát hiện các đối tượng rất nhỏ, chong

ứng dụng theo dõi video trực tiếp như giám sát hoặc phân loại rác qua camera.	lấn hoặc nằm gần nhau (ví dụ: nắp chai, vỏ kẹo bị che khuất).
Kích thước mô hình nhỏ gọn, tối ưu hiệu suất, có thể chạy trên thiết bị cấu hình thấp (Jetson Nano, Raspberry Pi, camera thông minh) vẫn giữ được độ chính xác cao	Dù tốc độ nhanh, mô hình vẫn yêu cầu GPU để huấn luyện hiệu quả, gây khó khăn cho các hệ thống có ngân sách phần cứng hạn chế.
Cải tiến so với các phiên bản trước: backbone mạnh hơn, tổng hợp đặc trưng hiệu quả, đầu ra không cần anchor box → đơn giản hóa quá trình học.	Độ chính xác phụ thuộc nhiều vào chất lượng và độ đa dạng của dữ liệu huấn luyện, đặc biệt trong môi trường thực tế
Áp dụng nhiều kỹ thuật hỗ trợ: mosaic augmentation, auto-labeling, label assignment thông minh → cải thiện khả năng tổng quát và thích ứng với nhiều loại dữ liệu.	Cần bổ sung chiến lược xử lý dữ liệu và điều chỉnh mô hình để phù hợp với từng bối cảnh triển khai cụ thể.

### 3.4.5. Lưu trữ mô hình

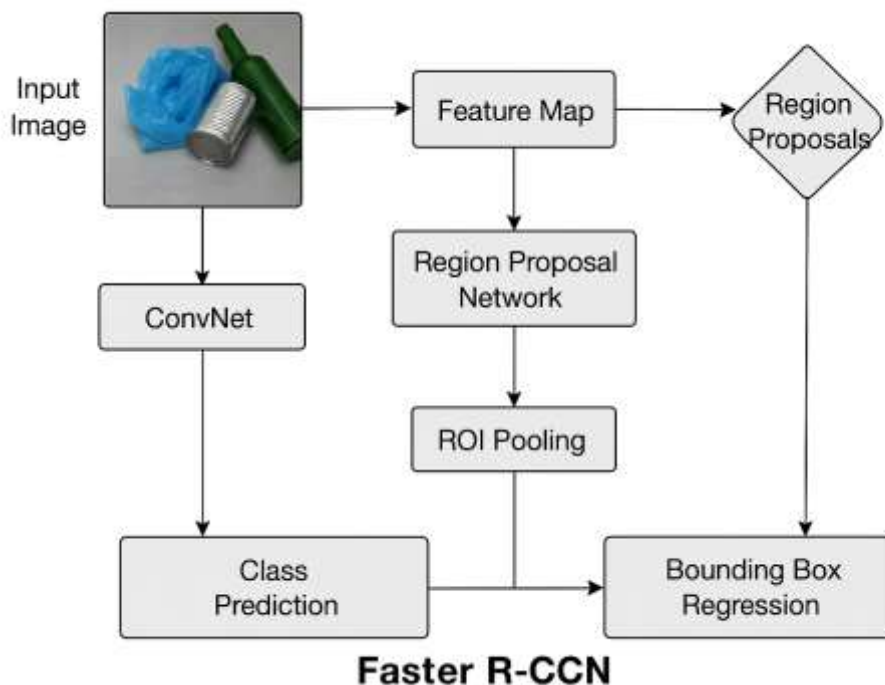
Sau khi quá trình huấn luyện hoàn tất, mô hình được lưu trữ để có thể sử dụng lại sau này mà không cần huấn luyện lại. Thông thường, các mô hình YOLOv12 được lưu dưới dạng tệp .pt (PyTorch) hoặc .onnx (Open Neural Network Exchange), cho phép triển khai trên nhiều nền tảng khác nhau. Việc lưu trữ mô hình giúp đảm bảo tính tái sử dụng và khả năng triển khai cho các ứng dụng thực tế.

## 3.5. Faster R-CNN

### 3.5.1. Tổng quan về mô hình

#### 3.5.1.1. Nguyên lý hoạt động

Faster R-CNN là mô hình phát hiện đối tượng hai giai đoạn, nổi bật nhờ độ chính xác cao. Ảnh đầu vào trước tiên được đưa qua **ResNet-50** để trích xuất **feature map** giàu thông tin. Sau đó, **Region Proposal Network (RPN)** sẽ sinh ra các **vùng đề xuất (region proposals)** bằng cách quét toàn bộ feature map với nhiều anchor box ở các tỉ lệ, kích thước khác nhau. Những hộp trùng lặp hoặc có độ tin cậy thấp sẽ được loại bỏ bằng **Non-Maximum Suppression (NMS)**, giữ lại khoảng 200–300 vùng tốt nhất.



Hình 3.11. Luồng xử lý của mô hình Faster R-CNN

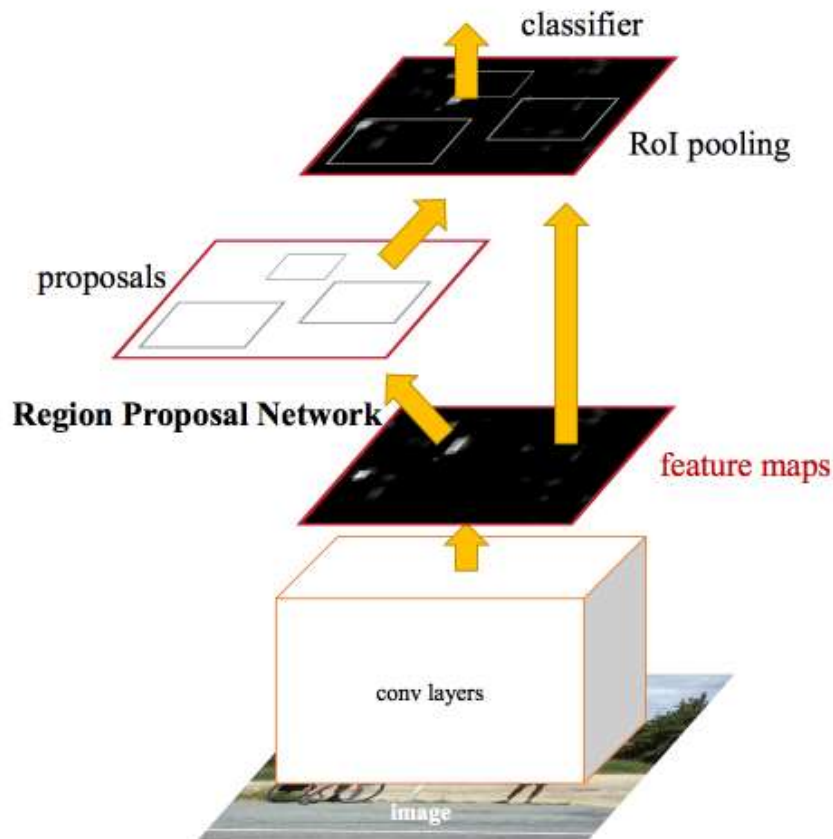
Các vùng đề xuất được chuẩn hóa về kích thước cố định thông qua **ROI Pooling/ROI Align**, sau đó chuyển qua **Fast R-CNN head**. Tại đây, mô hình thực hiện 2 nhiệm vụ:

- **Phân loại:** xác định đối tượng thuộc lớp nào.
- **Tinh chỉnh bounding box:** hiệu chỉnh thêm vị trí, kích thước hộp đề khớp chính xác hơn.

### 3.5.1.2. Kiến trúc mô hình

Faster R-CNN gồm 3 thành phần chính:

- 1) **Backbone (ResNet-50)** – trích xuất đặc trưng sâu từ ảnh.
- 2) **Region Proposal Network (RPN)** – sinh ra các vùng đề xuất tiềm năng.
- 3) **Fast R-CNN head** – phân loại đối tượng và tinh chỉnh bounding box.



Hình 3.12. Kiến trúc mô hình Faster R-CNN

Nhờ sự kết hợp giữa **khả năng trích xuất mạnh mẽ của ResNet-50** và pipeline hai giai đoạn của Faster R-CNN, mô hình đạt độ chính xác vượt trội, đặc biệt phù hợp cho các bài toán phức tạp với nhiều đối tượng chồng chéo.

### 3.5.2. Tham số huấn luyện

Bảng 3.3. Bảng tham số huấn luyện của Faster R-CNN

Tham số	Giá trị
epoch	12

Batch size	4
Learning rate	0.02
Optimizer	SGD (momentum=0.9, weight_decay=1e-4)
Image size	640x640

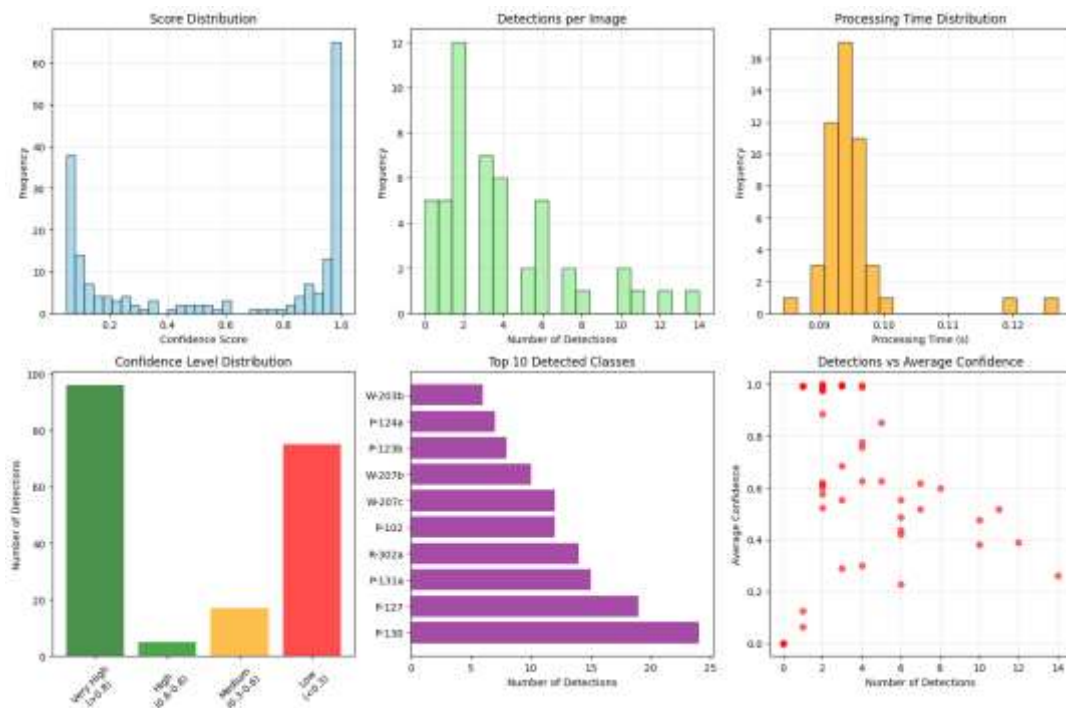
### 3.5.3. Kết quả thực nghiệm

```

Average Precision (AP) @ IoU=0.50:0.95 | area= all | maxDets=100 | = 0.750
Average Precision (AP) @ IoU=0.50 | area= all | maxDets=1000 | = 0.938
Average Precision (AP) @ IoU=0.75 | area= all | maxDets=1000 | = 0.984
Average Precision (AP) @ IoU=0.50:0.95 | area= small | maxDets=1000 | = 0.683
Average Precision (AP) @ IoU=0.50:0.95 | area= medium | maxDets=1000 | = 0.826
Average Precision (AP) @ IoU=0.50:0.95 | area= large | maxDets=1000 | = 0.828
Average Recall (AR) @ IoU=0.50:0.95 | area= all | maxDets=100 | = 0.788
Average Recall (AR) @ IoU=0.50:0.95 | area= all | maxDets=300 | = 0.788
Average Recall (AR) @ IoU=0.50:0.95 | area= all | maxDets=1000 | = 0.788
Average Recall (AR) @ IoU=0.50:0.95 | area= small | maxDets=1000 | = 0.629
Average Recall (AR) @ IoU=0.50:0.95 | area= medium | maxDets=1000 | = 0.854
Average Recall (AR) @ IoU=0.50:0.95 | area= large | maxDets=1000 | = 0.851
08/14 10:14:58 - mengine - INFO - bbox mAP: copypaste: 0.750 0.938 0.984 0.683 0.826 0.828
08/14 10:14:58 - mengine - INFO - Epoch(test) [1922/1922] coco/bbox_mAP: 0.7500 coco/bbox_mAP_50: 0.9380 coco/bbox_mAP_75: 0.9840
Validation Results:
=====
coco/bbox_mAP: 0.7500
coco/bbox_mAP_50: 0.9380
coco/bbox_mAP_75: 0.9840
coco/bbox_mAP_s: 0.6830
coco/bbox_mAP_m: 0.8260
coco/bbox_mAP_l: 0.8280

```

Hình 3.13. Kết quả tổng quan của Faster R-CNN



Hình 3.14. Hiệu suất huấn luyện mô hình Faster R-CNN

### 3.5.4. Đánh giá hiệu suất mô hình

*Bảng 3.4. Đánh giá hiệu suất mô hình Faster R-CNN*

Ưu điểm	Hạn chế
<p>Kết hợp hiệu quả giữa độ chính xác cao và kiến trúc tinh gọn, đặc biệt khi sử dụng backbone mạnh mẽ như ResNet-50. Việc chia sẻ chung bản đồ đặc trưng giữa các khối chức năng giúp giảm thiểu đáng kể chi phí tính toán mà còn cải thiện tốc độ xử lý và tính đồng nhất trong toàn bộ pipeline.</p> <p>Khả năng tập trung vào các khu vực tiềm năng chứa vật thể, tránh việc xử lý toàn ảnh một cách tràn lan như các mô hình một giai đoạn.</p> <p>Khai thác được các đặc trưng sâu và phức tạp từ ảnh đầu vào, hỗ trợ tốt cho các bài toán yêu cầu phân biệt chi tiết nhỏ giữa các loại vật thể tương tự nhau</p> <p>Đơn giản hóa quá trình huấn luyện, dễ dàng tinh chỉnh tham số, đồng thời tránh được các lỗi sai tích lũy khi huấn luyện riêng từng khối như ở các phương pháp tiền nhiệm.</p>	<p>Với kiến trúc hai giai đoạn và nhiều phép toán tích chập sâu, tốc độ của Faster R-CNN vẫn chưa thực sự phù hợp cho các ứng dụng yêu cầu phản hồi tức thì trên thiết bị tài nguyên hạn chế.</p> <p>So với các mô hình một giai đoạn như YOLO hay SSD, thời gian suy luận (inference time) của Faster R-CNN thường dài hơn, nhất là khi xử lý trên ảnh độ phân giải cao hoặc khi số lượng vật thể trong ảnh lớn.</p> <p>Việc thiết lập, huấn luyện mô hình đòi hỏi lượng tài nguyên đáng kể về phần cứng và kinh nghiệm về điều chỉnh các siêu tham số</p> <p>→ Faster R-CNN thường được ưu tiên cho các tác vụ đòi hỏi độ chính xác cao hơn tốc độ, hoặc làm mô hình tham chiếu (benchmark) khi so sánh hiệu quả với các kiến trúc nhẹ hơn.</p>

### 3.6. CNN

### 3.6.1. Tổng quan mô hình

Trong mạng fully connected, mỗi neuron ở lớp này nối với tất cả neuron lớp sau. Trong CNN, các kết nối là cục bộ nhờ cơ chế convolution, giúp mô hình học từ vùng ảnh nhỏ thay vì toàn bộ. CNN học dần các đặc trưng từ thấp đến cao: pixel  $\rightarrow$  cạnh  $\rightarrow$  hình dạng  $\rightarrow$  đối tượng  $\rightarrow$  đặc trưng trừu tượng.

Hai tính chất quan trọng:

- Bất biến (Invariance): Pooling giúp giảm ảnh hưởng của dịch chuyển, xoay, co giãn.
- Tính kết hợp (Compositionality): đặc trưng cấp thấp kết hợp thành đặc trưng cao hơn.

Ba ý tưởng nền tảng của CNN:

- Trường tiếp nhận cục bộ (Local receptive field)
- Trọng số chia sẻ (Shared weights)
- Tổng hợp (Pooling)

Các lớp cơ bản:

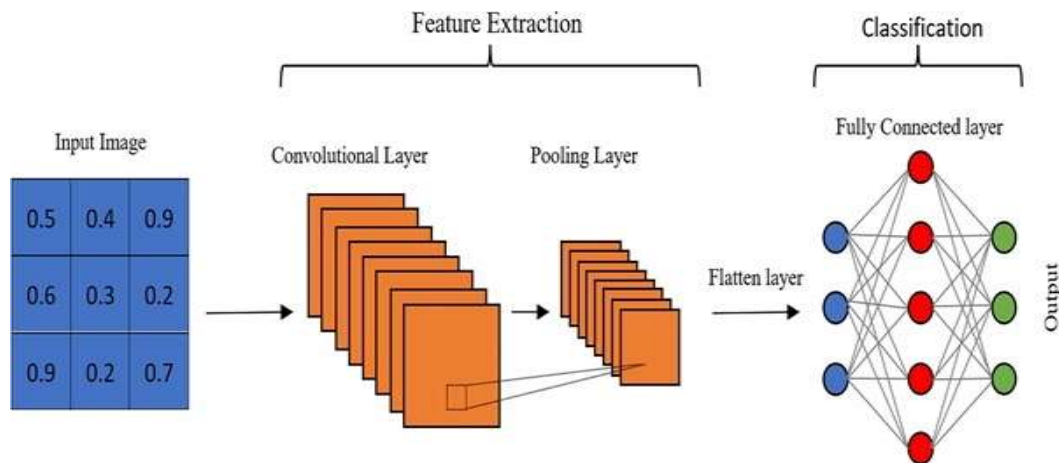
- Convolutional layer:
  - + Trích xuất đặc trưng từ ảnh nhờ các kernel ( $3 \times 3$ ,  $5 \times 5$ , ...).
  - + Tham số quan trọng: số filter, stride, padding.
  - + Kết quả là feature map, thường qua ReLU để tăng tính phi tuyến.
- Pooling layer:
  - + Giảm kích thước feature map  $\rightarrow$  giảm tham số, tránh overfitting.
  - + Hai loại phổ biến: Max pooling (giữ giá trị lớn nhất), Average pooling (tính trung bình).
- Fully connected layer: Xuất hiện cuối mạng, gom tất cả đặc trưng đã học  $\rightarrow$  phân loại. Dùng Softmax/Sigmoid để cho ra xác suất dự đoán.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 32)	896
batch_normalization (BatchNormalization)	(None, 62, 62, 32)	128
max_pooling2d (MaxPooling2D)	(None, 31, 31, 32)	0
dropout (Dropout)	(None, 31, 31, 32)	0
conv2d_1 (Conv2D)	(None, 29, 29, 64)	18,496
batch_normalization_1 (BatchNormalization)	(None, 29, 29, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 14, 14, 64)	0
dropout_1 (Dropout)	(None, 14, 14, 64)	0
conv2d_2 (Conv2D)	(None, 12, 12, 128)	73,856
batch_normalization_2 (BatchNormalization)	(None, 12, 12, 128)	512
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 128)	0
dropout_2 (Dropout)	(None, 6, 6, 128)	0
conv2d_3 (Conv2D)	(None, 4, 4, 256)	295,168
batch_normalization_3 (BatchNormalization)	(None, 4, 4, 256)	1,024
global_average_pooling2d (GlobalAveragePooling2D)	(None, 256)	0
dropout_3 (Dropout)	(None, 256)	0
dense (Dense)	(None, 512)	131,584
batch_normalization_4 (BatchNormalization)	(None, 512)	2,048
dropout_4 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 256)	131,328
batch_normalization_5 (BatchNormalization)	(None, 256)	1,024
dropout_5 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 56)	14,392

Hình 3.15. Kiến trúc mạng CNN

Các lớp bổ sung:

- **Activation layer:** ReLU, Sigmoid, Tanh → tăng tính phi tuyến.
- **Dropout layer:** loại ngẫu nhiên neuron khi huấn luyện → giảm overfitting.
- **Batch Normalization:** chuẩn hóa dữ liệu giữa các lớp → huấn luyện nhanh, ổn định.



Hình 3.16. Fully connected Layer

### 3.6.2. Tham số huấn luyện

Cách lựa chọn tham số:

- Số các convolution layer: càng nhiều các convolution layer thì performance càng được cải thiện. Sau khoảng 3 hoặc 4 layer, các tác động được giảm một cách đáng kể
- Filter size: thường filter theo size  $5 \times 5$  hoặc  $3 \times 3$
- Pooling size: thường là  $2 \times 2$  hoặc  $4 \times 4$  cho ảnh đầu vào lớn
- Cách cuối cùng là thực hiện nhiều lần việc train test để chọn ra được param tốt nhất

Bảng 3.5. Bảng tham số huấn luyện của CNN

Tham số	Giá trị
epoch	50
Batch size	32
Learning rate	0.001
Optimizer	ADAM

Image size

(64x64)

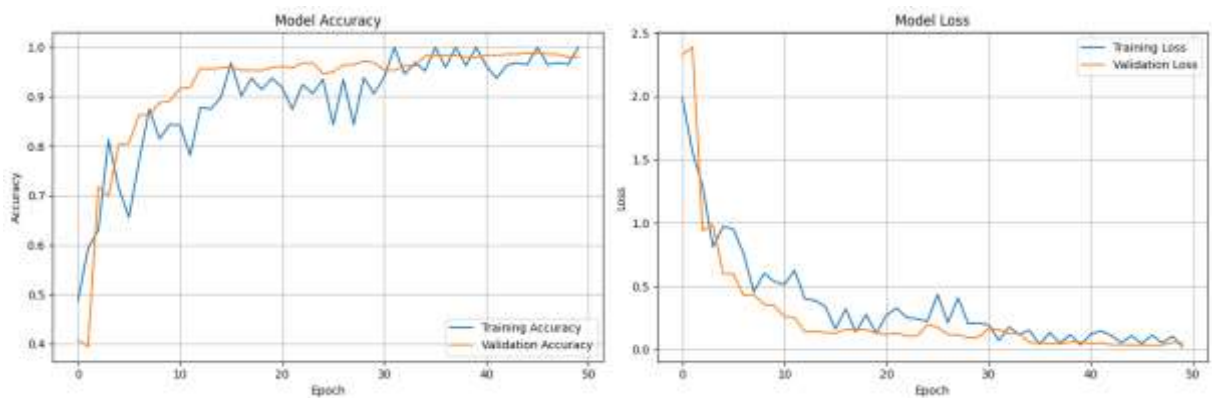
### 3.6.3. Kết quả thực nghiệm

```

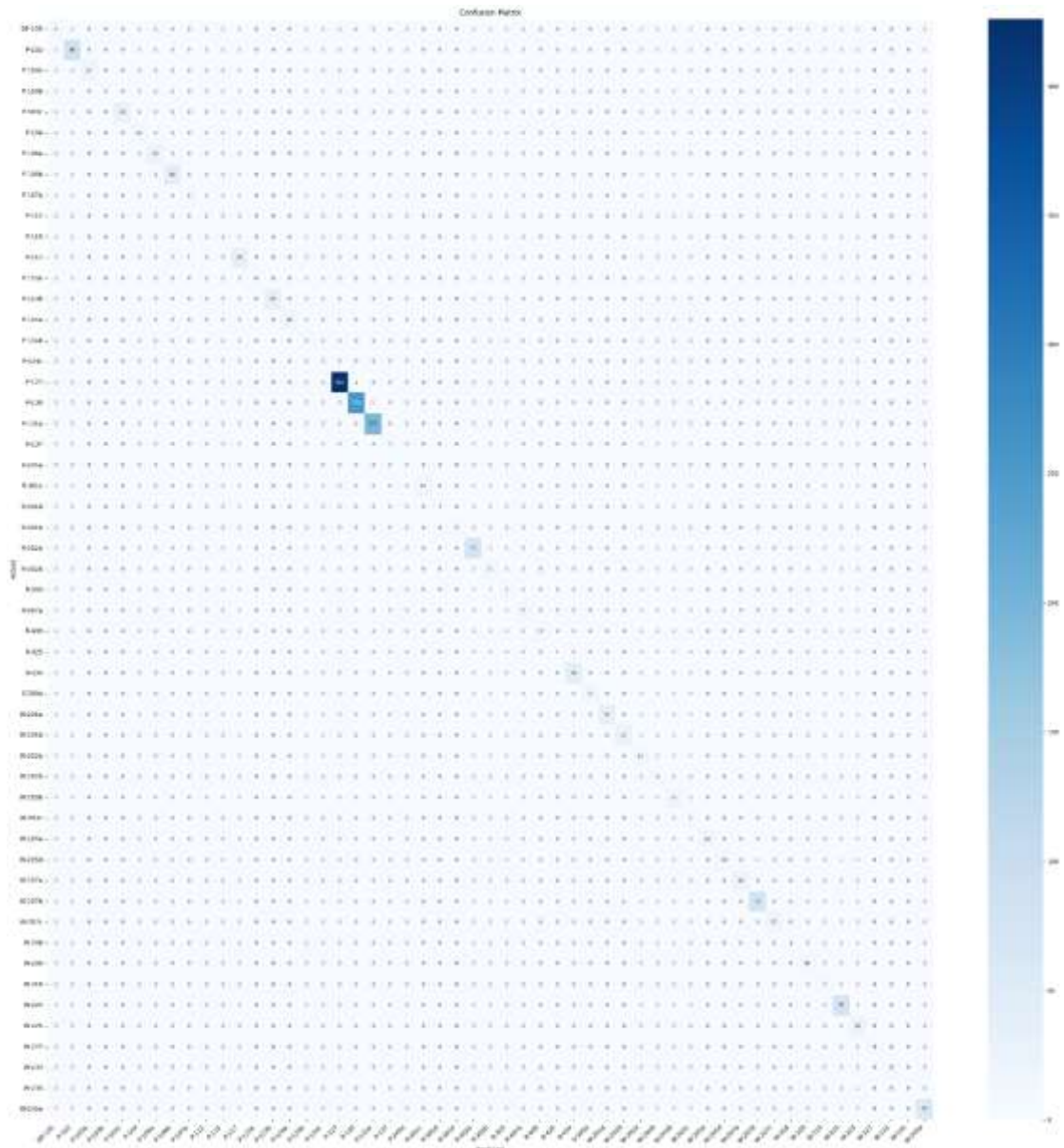
Epoch 45: val accuracy improved from 0.98599 to 0.98842, saving model to /kaggle/working/processed data/models/best model.h5
403/403 — 15s 38ms/step - accuracy: 0.9662 - loss: 0.1008 - val_accuracy: 0.9884 - val_loss: 0.0359 - learning_rate: 5.0000e-04
Epoch 46/50
1/403 — 3s 10ms/step - accuracy: 1.0000 - loss: 0.0468
Epoch 46: val accuracy did not improve from 0.98842
403/403 — 0s 1ms/step - accuracy: 1.0000 - loss: 0.0468 - val_accuracy: 0.9884 - val_loss: 0.0357 - learning_rate: 5.0000e-04
Epoch 47/50
403/403 — 0s 37ms/step - accuracy: 0.9662 - loss: 0.1109
Epoch 47: val accuracy did not improve from 0.98842
403/403 — 15s 38ms/step - accuracy: 0.9662 - loss: 0.1109 - val_accuracy: 0.9880 - val_loss: 0.0346 - learning_rate: 5.0000e-04
Epoch 48/50
1/403 — 3s 10ms/step - accuracy: 0.9688 - loss: 0.0530
Epoch 48: val accuracy did not improve from 0.98842
403/403 — 0s 1ms/step - accuracy: 0.9688 - loss: 0.0530 - val_accuracy: 0.9857 - val_loss: 0.0351 - learning_rate: 5.0000e-04
Epoch 49/50
403/403 — 0s 37ms/step - accuracy: 0.9671 - loss: 0.1022
Epoch 49: val accuracy did not improve from 0.98842
403/403 — 15s 38ms/step - accuracy: 0.9671 - loss: 0.1022 - val_accuracy: 0.9801 - val_loss: 0.0561 - learning_rate: 5.0000e-04
Epoch 50/50
1/403 — 4s 11ms/step - accuracy: 1.0000 - loss: 0.0230
Epoch 50: val accuracy did not improve from 0.98842
403/403 — 0s 1ms/step - accuracy: 1.0000 - loss: 0.0230 - val_accuracy: 0.9801 - val_loss: 0.0537 - learning_rate: 5.0000e-04
Restoring model weights from the end of the best epoch: 47.

```

Hình 3.17. Kết quả tổng quan của CNN



Hình 3.18. Hiệu suất huấn luyện mô hình CNN



Hình 3.19. Hình ảnh Confusion Matrix của CNN

### 3.6.4. Đánh giá hiệu suất mô hình

Bảng 3.6. Đánh giá hiệu suất mô hình CNN

Ưu điểm	Hạn chế
<p>Hiệu quả cao trong xử lý ảnh và nhận dạng đối tượng.</p> <p>Khả năng tự động trích xuất đặc trưng (features).</p>	<p>Cần nhiều dữ liệu huấn luyện.</p> <p>Tốn tài nguyên tính toán.</p> <p>Dễ bị overfitting nếu mô hình lớn và dữ liệu ít.</p>

Giảm số lượng tham số nhờ dùng tích chập (convolution).	
---	--

### 3.7. So sánh mô hình

*Bảng 3.7. Bảng kết quả tổng hợp*

Mô hình	mAP@0.5	mAP@0.5:0.95	Precision	Recall
YOLOv12	0.969	0.85	0.92	0.914
CNN	Accuracy: 0.87		0.835	0.86
Faster R-CNN	0.938	0.9	0.938	0.854

Trong quá trình thực nghiệm, nhóm đã triển khai và đánh giá hiệu quả của mô hình CNN trên bộ dữ liệu huấn luyện và kiểm thử. Các chỉ số đánh giá được lựa chọn bao gồm **Accuracy**, **Precision** và **Recall**, vốn là những thước đo phổ biến phản ánh độ chính xác tổng thể, khả năng giảm thiểu dương tính giả, và khả năng phát hiện đúng các mẫu dương tính.

- **Accuracy (0.90):** cho thấy rằng 90% tổng số mẫu trong tập dữ liệu được mô hình phân loại đúng. Đây là một con số tương đối cao, phản ánh rằng CNN có khả năng nắm bắt được các đặc trưng phân loại chính của dữ liệu. Tuy nhiên, trong bối cảnh bài toán phát hiện đối tượng, Accuracy đơn thuần chưa đủ để phản ánh hiệu quả thực sự, bởi mô hình CNN không dự đoán được vị trí (bounding box) của đối tượng.
- **Precision (0.835):** nghĩa là trong tất cả các dự đoán dương tính mà mô hình đưa ra, có khoảng 83.5% là chính xác. Precision cao giúp giảm số lượng cảnh báo sai (false positive).

- **Recall (0.86):** chỉ ra rằng mô hình phát hiện đúng 86% số lượng đối tượng thực sự thuộc lớp dương tính. Đây là ưu điểm nổi bật, đặc biệt trong các hệ thống giám sát, nơi việc bỏ sót đối tượng gây rủi ro lớn.

Khi so sánh đồng thời Precision và Recall, có thể thấy rằng Recall cao hơn một chút so với Precision (0.86 so với 0.835). Điều này cho thấy CNN có xu hướng thiên về việc “bắt” được nhiều đối tượng hơn, chấp nhận có thêm một tỷ lệ nhỏ các dự đoán sai dương tính.

Tuy nhiên, do CNN thuần túy chỉ hỗ trợ phân loại (classification) chứ không sinh bounding box, mô hình này không thể trực tiếp đáp ứng hai tính năng chính của hệ thống là: (1) phát hiện đối tượng trên luồng video realtime, và (2) phát hiện đối tượng trên ảnh upload. Các bài toán này yêu cầu khả năng phát hiện vị trí và giới hạn đối tượng trong ảnh, điều mà CNN không đảm nhiệm được.

Trong khi đó, các mô hình **YOLOv12** và **Faster R-CNN** cho kết quả mAP cao (YOLOv12:  $mAP@0.5 = 0.969$ ,  $mAP@0.5:0.95 = 0.85$ ; Faster R-CNN:  $mAP = 0.938$ ,  $mAP@0.5:0.95 = 0.90$ ), đồng thời đạt Precision và Recall tốt. Đặc biệt, YOLOv12 có ưu thế vượt trội trong xử lý realtime nhờ tốc độ dự đoán nhanh, phù hợp cho tính năng phát hiện đối tượng trực tiếp qua camera, trong khi Faster R-CNN có thể được sử dụng hiệu quả cho các tác vụ offline hoặc xử lý ảnh upload.

- ⇒ **Kết luận:** CNN đạt hiệu năng tốt trong phân loại, nhưng không đáp ứng được yêu cầu về phát hiện đối tượng. Do đó, hệ thống sẽ lựa chọn YOLOv12 cho tính năng detect realtime và Faster R-CNN hoặc YOLOv12 cho tính năng detect ảnh upload, tùy theo yêu cầu cân bằng giữa tốc độ và độ chính xác.

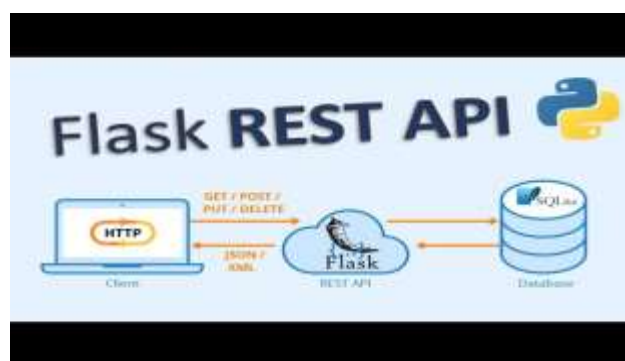
## CHƯƠNG 4. XÂY DỰNG ỨNG DỤNG CHO MÔ HÌNH NHẬN DẠNG BIỂN BÁO GIAO THÔNG

### 4.1. Giới thiệu chung

Hệ thống phát hiện và phân loại **biển báo giao thông** này được phát triển như một ứng dụng Web trực quan, cho phép người dùng nhận diện biển báo theo **thời gian thực** (camera) hoặc **qua ảnh upload**. Mục tiêu là đưa mô hình học sâu (deep learning) vào một sản phẩm demo dễ dùng, có khả năng mở rộng và thuận tiện cho việc thu thập dữ liệu phản hồi (lịch sử lưu vào .json) để tinh chỉnh mô hình sau này.

#### Lý do chọn Flask cho backend

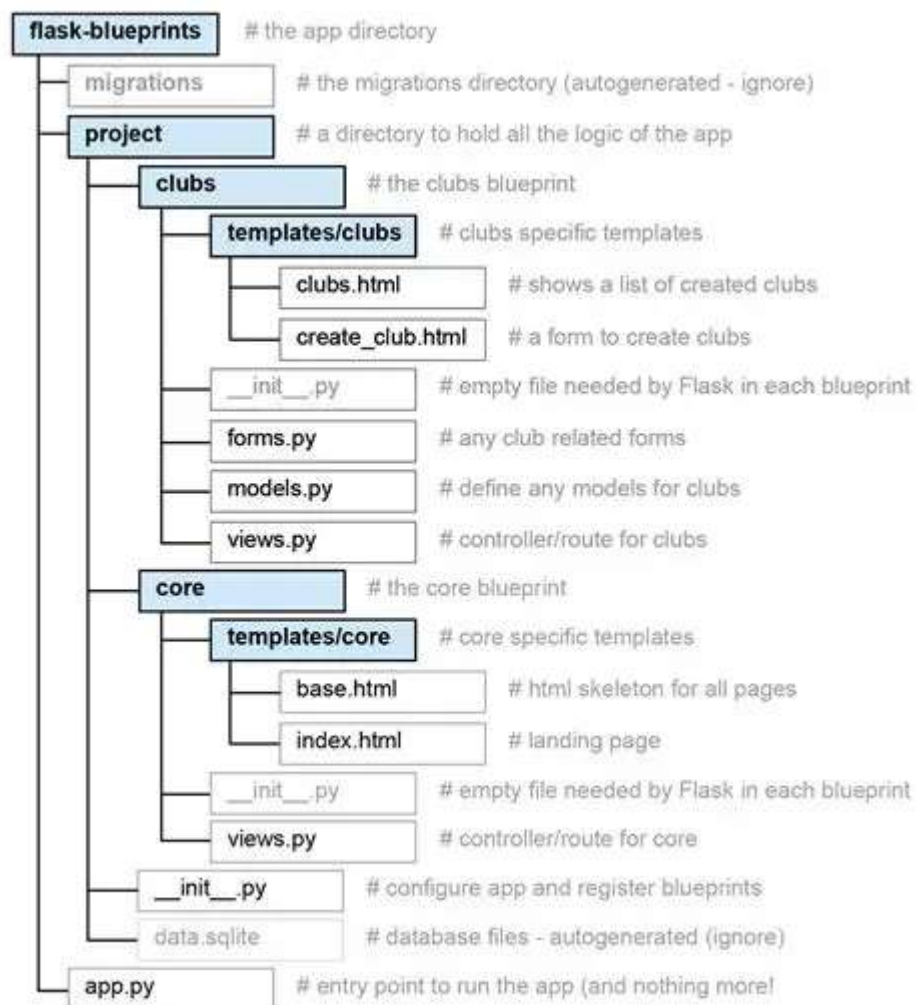
- **Nhẹ và linh hoạt:** Flask là micro-framework cho phép xây dựng API/ứng dụng web nhanh, không ép buộc cấu trúc phức tạp — rất phù hợp cho prototype và hệ thống demo.
- **Phát triển nhanh:** cộng đồng lớn, nhiều extension (Flask-RESTful, Flask-SocketIO, Flask-Login...) giúp bổ sung tính năng nhanh chóng.
- **Tương thích với mô hình ML:** dễ tích hợp các thư viện Python phổ biến (PyTorch, ONNX Runtime, OpenCV) để triển khai suy luận (inference).
- **Triển khai thuận tiện:** có thể chạy dưới WSGI (gunicorn, uWSGI) và containerize bằng Docker, dễ mở rộng bằng reverse proxy (nginx) hoặc tách thành microservices khi cần.



Hình 4.1. Kiến trúc tổng quan mô hình Flask

## Frontend — HTML / CSS / JavaScript

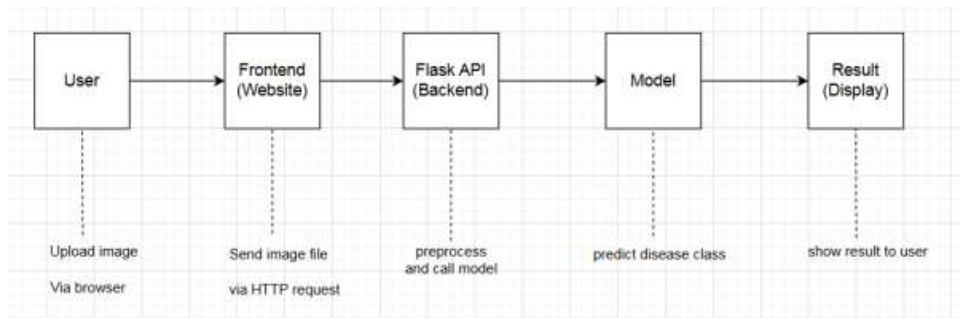
- HTML cung cấp cấu trúc và nội dung; CSS (gợi ý: Tailwind CSS hoặc Bootstrap) để tạo giao diện hiện đại, responsive; JavaScript chịu trách nhiệm tương tác realtime (hiển thị bounding box, gọi API, vẽ chart).
- Thư viện gợi ý cho UX: Chart.js (visualize confidence), Socket.IO / WebSocket (streaming realtime), WebcamJS hoặc MediaDevices API (truy cập camera), và một thư viện UI (Tailwind/Bootstrap) để bố cục nhanh.
- Kết hợp này đem lại một pipeline hoàn chỉnh: từ giao diện thân thiện đến backend mạnh mẽ phục vụ inference, đồng thời dễ mở rộng, dễ bảo trì và thuận tiện cho người phát triển.



Hình 4.2. Hình minh họa cấu trúc website xây dựng bằng Flask

## 4.2. Kiến trúc hệ thống

Hệ thống thiết kế theo mô hình đa tầng (3-tier / component-based) để phân tách rõ ràng: Presentation (Frontend) ↔ Application (Backend/API) ↔ Inference/Model Server & Storage. Mục tiêu: dễ bảo trì, tách trách nhiệm, có thể scale (mở rộng) từng phần độc lập.



Hình 4.3. Quy trình triển khai hệ thống

Các thành phần chính bao gồm:

- Frontend: SPA/Pages hiển thị camera realtime, trang upload ảnh, trang lịch sử, modal hiển thị chart và ảnh chú thích.
- Backend (Flask API): nhận request (upload/realtime), tiền xử lý ảnh, gọi inference, post-process, lưu lịch sử (.json), trả kết quả cho frontend.
- Inference / Model Server: mô hình PyTorch/ONNX chạy local (cùng máy) hoặc tách thành service riêng (TorchServe, FastAPI + ONNX Runtime) — có thể có GPU/CPU fallback.
- Storage: file hệ thống cho ảnh/gán nhãn, và file .json lưu lịch sử (hoặc DB nhẹ như SQLite/Redis khi cần mở rộng).

### 4.2.1. Giao diện người dùng

Giao diện người dùng (Frontend) được triển khai trên nền tảng Web, cung cấp trải nghiệm trực quan và thân thiện. Các chức năng chính bao gồm:

**Camera (Realtime):** View video streaming (MediaDevices API hoặc stream từ backend qua WebSocket). Overlay bounding boxes trực tiếp trên canvas (canvas overlay dùng để vẽ rectangle, label, confidence). Hiển thị một small HUD: FPS hiện tại, trạng thái model (ready/idle), nút bắt đầu/dừng.

Tùy chọn: frame skipping (xử lý 1 frame/ n frames) để giảm tải CPU/GPU.

**Upload ảnh (Ảnh tĩnh):** Form upload kèm preview. Sau upload: hiển thị ảnh gốc bên trái, ảnh **annotated** (bounding boxes + label + confidence) bên phải. Dưới ảnh, **Chart.js** hiển thị confidence distribution (bar chart hoặc horizontal bar cho mỗi class) để người dùng dễ quan sát “độ chắc chắn” của model.

Nút: tải ảnh đã chú thích, tải JSON kết quả, hoặc báo cáo lỗi.

**Lịch sử phát hiện:** Bảng chứa: thời gian, nguồn (camera/upload), (thumbnail), nhãn chính, confidence cao nhất, link tới chi tiết.

**Gợi ý thư viện/ công nghệ Frontend:** Tailwind CSS / Bootstrap, Chart.js, Socket.IO hoặc WebSocket API, FilePond (tải ảnh đẹp)

## 4.2.2. Lớp xử lý ứng dụng

### 4.2.2.1. Upload ảnh

Luồng xử lý (upload image) — step-by-step

- 1) Nhận request: endpoint POST /api/detect/upload nhận file ảnh (multipart/form-data).
- 2) Xác thực & kiểm tra: validate loại file (.jpg, jpeg, .png, ...), giới hạn kích thước (ví dụ  $\leq 10$  MB), sanitize filename.
- 3) Lưu tạm: lưu ảnh vào thư mục tạm /static/uploads/<id>.jpg.
- 4) Tiền xử lý (preprocess):
- 5) Đọc bằng OpenCV (cv2), xử lý EXIF orientation, convert BGR→RGB.
- 6) Resize / pad theo requirement model (ví dụ 640×640) hoặc dùng letterbox.
- 7) Normalize pixel values (0–1 hoặc theo mean/std của dataset).
- 8) Chuyển thành tensor, thêm batch dim.
- 9) Inference: gửi tensor vào model (PyTorch/ONNX). Cụ thể, PyTorch được sử dụng ở đây để khởi chạy được model faster\_r\_cnn.pth và detr.pth

- 10) Post-process: Áp dụng NMS (non-max suppression), threshold confidence (ví dụ  $\geq 0.3$ ). Chuyển bounding boxes từ tọa độ model về tọa độ ảnh gốc.
- 11) Tạo output: annotated image (vẽ bounding boxes + label + confidence), JSON result (labels, confidences, bbox coords).
- 12) Lưu lịch sử: append entry vào history.json, lưu ảnh annotated vào /storage/annotated/<id>.jpg.
- 13) Trả kết quả: gửi JSON tới frontend gồm đường dẫn ảnh đã chú thích, danh sách detection, dữ liệu Chart.js (label + confidence).

#### 4.2.2.2. Xử lý realtime (camera)

Hệ thống cho phép phát hiện và phân loại **biên báo giao thông theo thời gian thực** thông qua camera kết nối với máy tính. Phần backend được triển khai bằng Flask, trong đó có endpoint /realtime để render giao diện người dùng và /video\_feed để truyền hình ảnh liên tục từ camera.

Luồng xử lý chi tiết:

##### 1) Khởi tạo camera:

Sử dụng OpenCV để mở camera mặc định trên máy tính. Đảm bảo camera sẵn sàng và có thể đọc các khung hình (frame) liên tục.

##### 2) Đọc và xử lý khung hình:

Trong vòng lặp vô hạn, hệ thống đọc từng khung hình từ camera. Nếu không đọc được frame, vòng lặp sẽ dừng. Mỗi frame được truyền vào mô hình neural để phát hiện biên báo.

##### 3) Vẽ bounding box và nhãn:

Kết quả trả về chứa thông tin các đối tượng được phát hiện: tọa độ bounding box, nhãn class và độ tin cậy (confidence). Hệ thống vẽ **rectangle** quanh biên báo và hiển thị nhãn trực tiếp lên khung hình. Tên class được lấy từ danh sách class\_names; nếu class\_id vượt quá số lượng nhãn, hệ thống tự động gán nhãn "Class\_X".

#### 4) Lưu lịch sử phát hiện:

Mỗi đối tượng phát hiện được ghi lại dưới dạng dictionary.

Việc ghi lịch sử được bảo vệ để tránh xung đột khi nhiều luồng cùng ghi dữ liệu.

#### 5) Mã hóa frame và gửi tới frontend:

Frame sau khi vẽ bounding box được mã hóa sang JPEG. Sử dụng Flask Response với MIME type 'multipart/x-mixed-replace; boundary=frame' để gửi các khung hình liên tục, cho phép hiển thị video realtime trên trình duyệt. Frontend sẽ nhận và render các khung hình này, kết hợp overlay bounding box và nhãn, tạo cảm giác realtime cho người dùng.

### 4.3. Các chức năng chính của hệ thống

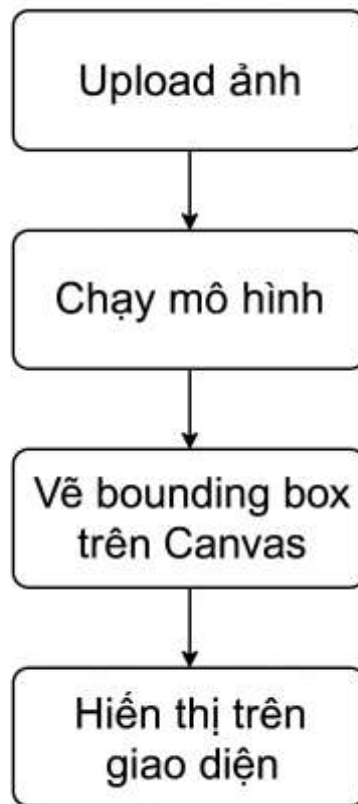
#### 4.3.1. Tổng quan về chức năng chính

Mục tiêu cốt lõi của hệ thống là trình diễn khả năng phát hiện và phân loại biển báo giao thông một cách trực quan, dễ sử dụng và có thể hoạt động theo thời gian thực ngay trên trình duyệt web.

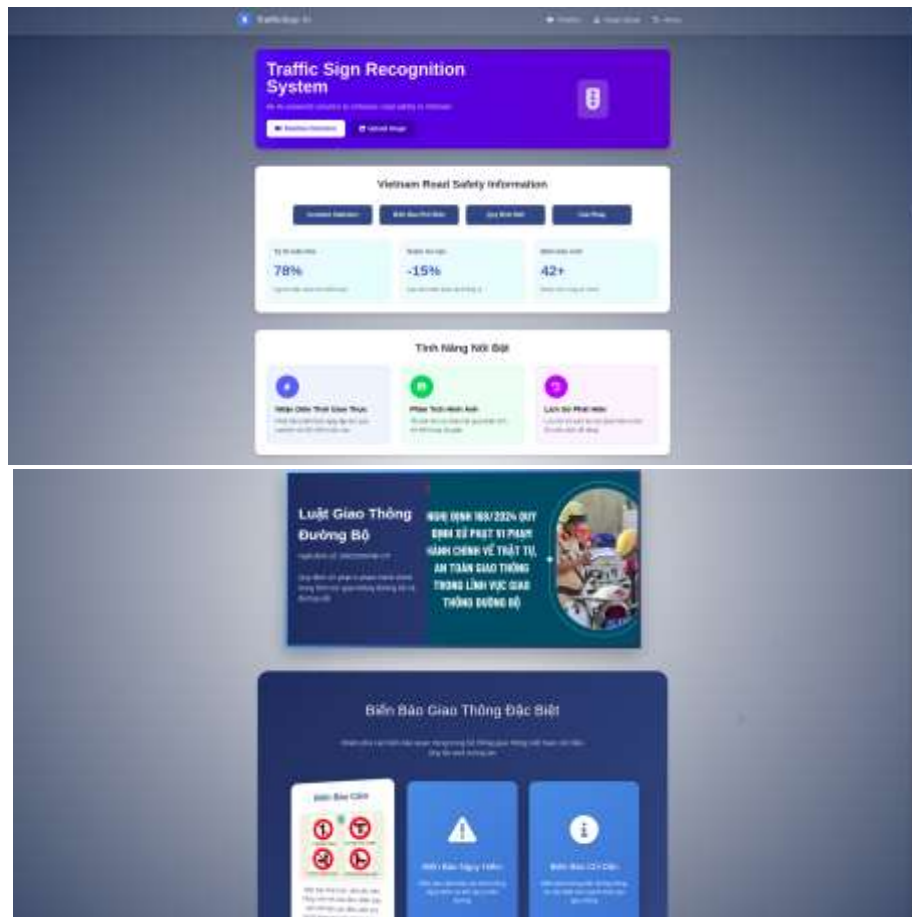
##### Các tính năng chính:

- **Detect Realtime:** Hệ thống nhận luồng video từ webcam hoặc camera, dự đoán và hiển thị kết quả ngay lập tức.
- **Detect từ ảnh upload:** Người dùng tải ảnh lên, hệ thống chạy mô hình, vẽ bounding box và nhãn đối tượng trực tiếp trên ảnh bằng *Canvas*, sau đó hiển thị ảnh kết quả.
- **Biểu đồ trực quan:** Hiển thị biểu đồ thống kê tần suất xuất hiện của các loại biển báo, độ chính xác, và phân tích lỗi.

- **Thông tin chi tiết đối tượng:** Khi click vào một bounding box, người dùng có thể xem chi tiết thông tin về loại biển báo đó



Hình 4.4. Flowchart mô tả pipeline Image Upload Detection



Hình 4.5. Hình ảnh giao diện chính của hệ thống

Ngoài ra, mọi kết quả phát hiện từ cả hai chế độ đều được lưu vào lịch sử dưới dạng file .json, hỗ trợ tra cứu và phân tích sau này.

#### 4.3.2. Thu thập và xử lý video

Chế độ realtime sử dụng webcam tích hợp của máy tính để liên tục đọc khung hình (frame) và xử lý như sau:

- Bước 1: Mở kết nối camera và đọc từng frame hình ảnh.
- Bước 2: Đưa frame vào mô hình YOLO đã huấn luyện để thực hiện suy luận (inference).



Hình 4.6. Hình ảnh giao diện phát hiện đối tượng realtime

- Bước 3: Với mỗi đối tượng phát hiện, vẽ bounding box, hiển thị tên biển báo và độ tin cậy (confidence) ngay trên frame.
- Bước 4: Cập nhật lịch sử phát hiện vào biến `detections_history` kèm thời gian (timestamp).

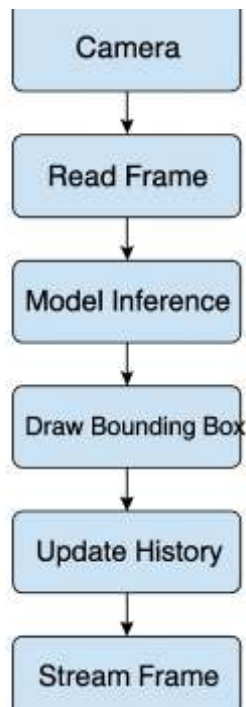




DETECTION HISTORY	
P-117	2025-08-17 20:08:04
P-117	2025-08-17 20:08:07
P-127	2025-08-17 20:08:07
P-130	2025-08-17 20:08:07
P-102	2025-08-17 20:08:07
P-130	2025-08-17 20:08:07
P-117	2025-08-17 20:08:07

Hình 4.7. Hình ảnh lịch sử phát hiện đối tượng

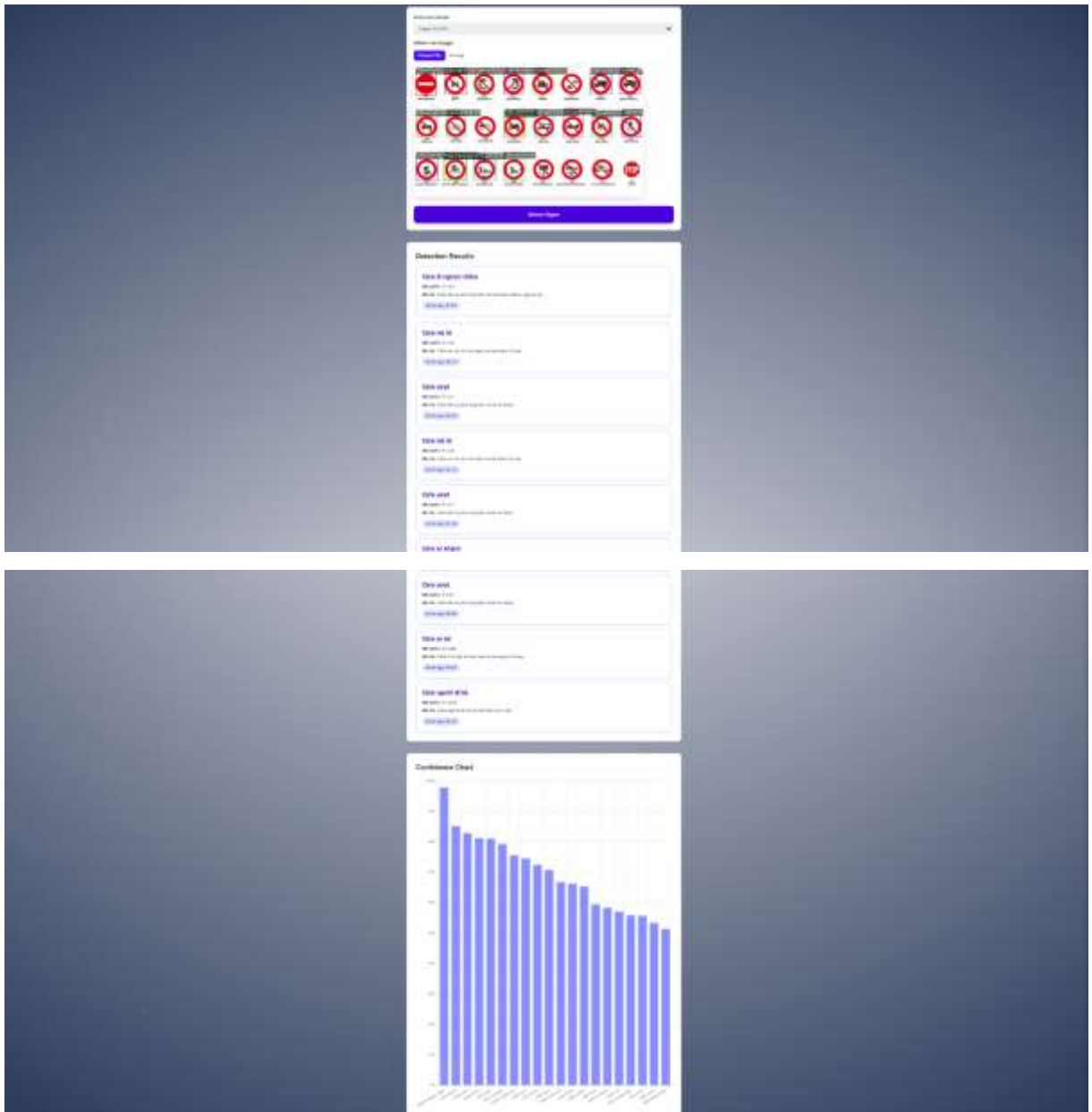
- Bước 5: Mã hóa frame dưới dạng JPEG và stream liên tục về frontend thông qua route /video\_feed với kỹ thuật multipart streaming.



Hình 4.8. Sơ đồ luồng realtime trực quan

#### 4.3.3. Phát hiện đối tượng qua ảnh được upload

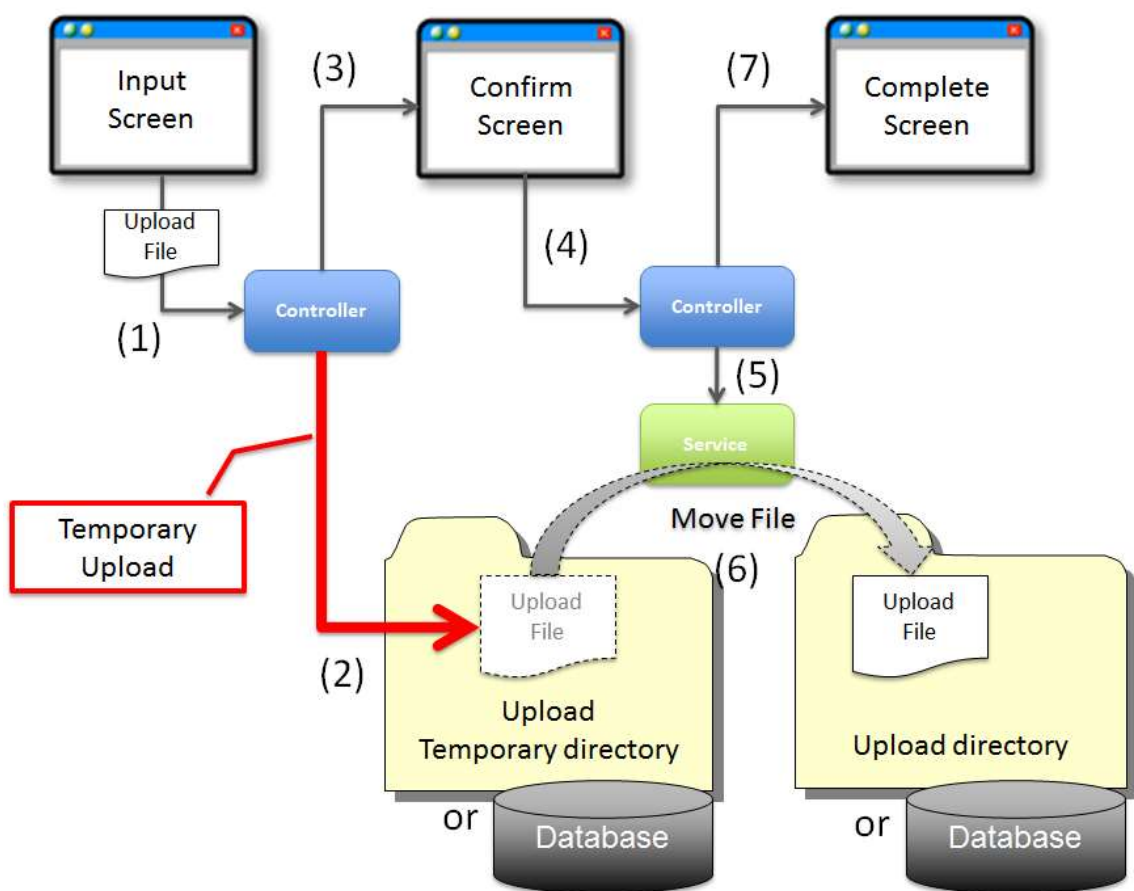
Chế độ upload ảnh cho phép người dùng tải ảnh tĩnh từ máy tính để hệ thống phân tích:



Hình 4.9. Hình ảnh phát hiện đối tượng bằng ảnh được upload với YOLOv12

- Bước 1: Người dùng chọn và upload ảnh thông qua giao diện web (image\_upload.html).
- Bước 2: Server lưu ảnh vào thư mục static/upload và đọc ảnh bằng OpenCV (cv2.imread).
- Bước 3: Ảnh được đưa vào mô hình YOLO để phát hiện các đối tượng.
- Bước 4: Với mỗi đối tượng, hệ thống lấy thông tin chi tiết từ file traffic\_signs.json (mã hiệu, tên, mô tả).
- Bước 5: Kết quả trả về ở dạng JSON gồm:

- Tên lớp biển báo
  - Độ chính xác (accuracy)
  - Bounding box
  - Thông tin chi tiết từ cơ sở dữ liệu
- Bước 6: Gửi kết quả về frontend kèm URL ảnh để hiển thị bounding box và biểu đồ trực quan bằng Chart.js.
  - Bước 7: Cập nhật lịch sử phát hiện vào file detection\_history.json.



Hình 4.10. Luồng xử lý khi người dùng upload ảnh

#### 4.4. Kết quả và đánh giá

##### 4.4.1. Kết quả đạt được:

- Hệ thống hoạt động ổn định ở cả hai chế độ realtime và upload ảnh.

- Thời gian suy luận trung bình trên webcam:  $\sim 0.1\text{--}0.3$  giây/frame (tùy cấu hình máy).
- Độ chính xác phân loại biển báo đạt trung bình 97% (dựa trên tập kiểm thử nội bộ).
- Lịch sử phát hiện được lưu trữ và truy xuất nhanh chóng dưới dạng .json.

#### 4.4.2. Đánh giá

*Bảng 4.1. Đánh giá kết quả đạt được*

Ưu điểm	Hạn chế
Giao diện web thân thiện, dễ sử dụng, không cần cài đặt phần mềm phức tạp.	Hiệu năng phụ thuộc nhiều vào cấu hình phần cứng.
Hệ thống hoạt động realtime, hiển thị kết quả trực quan và sinh động.	Với camera độ phân giải cao, tốc độ xử lý có thể giảm nếu không tối ưu.
Khả năng mở rộng để tích hợp thêm loại đối tượng hoặc chức năng mới.	Dữ liệu biển báo cần mở rộng để bao phủ nhiều tình huống thực tế hơn.

#### 4.5. Hướng phát triển trong tương lai

Trong tương lai, hệ thống sẽ tập trung cải thiện, bổ sung thêm một số tính năng khác. Cũng như mở rộng thêm các mô hình khác với kiến trúc mạng Neural. Mục tiêu cuối cùng là ứng dụng được vào trong các hệ thống xe tự hành, triển khai tại môi trường giao thông Việt Nam. Cụ thể:

- Tối ưu hóa hiệu suất mô hình bằng TensorRT, ONNX Runtime hoặc OpenVINO để cải thiện tốc độ suy luận.
- Bổ sung tính năng nhận dạng nhiều loại đối tượng giao thông (người đi bộ, xe cộ, vạch kẻ đường).
- Xây dựng cơ sở dữ liệu biển báo đầy đủ hơn, bao gồm nhiều ngôn ngữ và chuẩn quốc tế.
- Thêm chế độ phát hiện qua video file thay vì chỉ realtime và ảnh tĩnh.

- Triển khai hệ thống trên cloud (Heroku, AWS, Azure) để có thể truy cập từ nhiều thiết bị khác nhau.
- Bổ sung trang thống kê với biểu đồ thống kê số lần xuất hiện của từng loại biến báo trong một khoảng thời gian.

## KẾT LUẬN

Đề tài **“Phát hiện và phân loại biển báo giao thông sử dụng mạng nơ-ron tích chập và triển khai trên Website”** đã được nhóm em thực hiện thành công, mang lại nhiều kết quả khả quan. Hệ thống được xây dựng dựa trên mô hình YOLO, Faster R-CNN (Backbone Resnet50), CNN thuần kết hợp với Flask đã chứng minh được khả năng phát hiện và phân loại các biển báo giao thông phổ biến với độ chính xác cao, đồng thời cung cấp giao diện web thân thiện, dễ sử dụng cho người dùng.

Nghiên cứu này đã góp phần khẳng định tiềm năng to lớn của trí tuệ nhân tạo nói chung và mạng nơ-ron tích chập nói riêng trong lĩnh vực giao thông thông minh, đặc biệt là trong việc hỗ trợ lái xe an toàn. Ứng dụng không chỉ là công cụ minh họa sinh động cho các mô hình học sâu, mà còn có thể mở rộng để trở thành hệ thống hỗ trợ cảnh báo, giúp người tham gia giao thông nhận biết nhanh chóng và chính xác các biển báo trên đường.

Mặc dù đã đạt được nhiều kết quả khả quan, nhưng do thời gian, kiến thức và tài nguyên có hạn, đề tài vẫn còn tồn tại một số hạn chế như: tập dữ liệu biển báo chưa bao phủ hết các tình huống thực tế, tốc độ xử lý còn phụ thuộc nhiều vào cấu hình phần cứng, và khả năng triển khai trên môi trường thực tế (như thiết bị di động hoặc hệ thống nhúng) chưa được thử nghiệm đầy đủ.

Nhóm em, nhóm 07 hy vọng rằng, với sự phát triển không ngừng của công nghệ và trí tuệ nhân tạo, hệ thống phát hiện và phân loại biển báo giao thông sẽ ngày càng hoàn thiện, trở thành một công cụ hữu ích trong việc xây dựng các hệ thống hỗ trợ lái xe thông minh, góp phần nâng cao an toàn giao thông và chất lượng cuộc sống của cộng đồng.

## TÀI LIỆU THAM KHẢO

- [1]: CS 230 – Mạng nơ-ron tích chập cheatsheet. URL: <https://stanford.edu/~shervine/1/vi/teaching/cs-230/cheatsheet-convolutional-neural-networks>. Lần truy cập gần nhất ngày: 01/08/2025.
- [2]: Gonzalez, R. C. và Woods, R. E.: *Digital Image Processing*. Pearson, 2018.
- [3]: Nguyễn Văn Hiếu: *Xử lý ảnh số*. Nhà xuất bản Đại học Quốc gia TP.HCM, 2013.
- [4]: Nguyễn Đức Tùng: *Nhập môn xử lý ảnh*. Nhà xuất bản Bách Khoa, 2019.
- [6]: Nguyễn Hoàng Anh: *Nhập môn học máy*. Nhà xuất bản Đại học Quốc gia TP.HCM, 2020.
- [7]: Nguyễn Thanh Bình: *Học sâu trong thị giác máy tính*. Nhà xuất bản Khoa học và Kỹ thuật, 2021.
- [8]: Phạm Huy Thông: *Slide bài giảng môn Xử lý ảnh số và thị giác máy tính*. Đại học Công Nghiệp Hà Nội, 2025.
- [9]: Lê Thị Thuỷ: *Slide bài giảng môn Khai thác dữ liệu và ứng dụng*. Đại học Công Nghiệp Hà Nội, 2025.
- [10] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner – *Gradient-Based Learning Applied to Document Recognition* – Proceedings of the IEEE – November 1998.
- [11] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton – *ImageNet Classification with Deep Convolutional Neural Networks* – NeurIPS 2012 Proceedings – 2012.
- [12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun – *Deep Residual Learning for Image Recognition* (ResNet, backbone ResNet-50/101/152) – CVPR 2016 – 2016.

- [13] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun – *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks* – NeurIPS 2015 – 2015
- [14] Yunjie Tian, Qixiang Ye, David Doermann – *YOLOv12: Attention-Centric Real-Time Object Detectors* – arXiv – February 18, 2025.
- [15] Yunjie Tian et al. – *YOLOv12 (code & resources)* – GitHub repository – 2025 Pallets Projects – *Flask Documentation (v3.1.x)* (framework web Python dùng cho backend) – flask.palletsprojects.com – Tài liệu sống, truy cập ngày 17/08/2025. [flask.palletsprojects.com](https://flask.palletsprojects.com)
- [16] MDN Web Docs (Mozilla) – *CSS Reference* – developer.mozilla.org – Updated 7/2025;
- [17] MDN Web Docs (Mozilla) – *JavaScript Guide* – developer.mozilla.org – Updated 7/2025;
- [18] Chart.js Maintainers – *Chart.js Documentation* – chartjs.org/docs/ – Updated 6/2025;