

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

TRẦN VIỆT TRƯỜNG

ĐỒ ÁN TỐT NGHIỆP
NGÀNH KHOA HỌC MÁY TÍNH

**XÂY DỰNG MÔ HÌNH NHẬN DIỆN VI PHẠM AN
TOÀN GIAO THÔNG**

Giáo viên hướng dẫn : TS. Trần Hùng Cường
Sinh viên thực hiện : Trần Việt Trường
Mã sinh viên : 2021600591

NGÀNH KHOA HỌC MÁY TÍNH

Hà Nội - 2025

MỤC LỤC

MỤC LỤC	i
DANH MỤC HÌNH ẢNH.....	iii
DANH MỤC BẢNG BIỂU	iv
DANH MỤC KÝ HIỆU – VIẾT TẮT	v
LỜI CẢM ƠN	vi
LỜI NÓI ĐẦU	1
CHƯƠNG 1: KHẢO SÁT HỆ THỐNG	4
1.1 Thực trạng giao thông tại Việt Nam	4
1.2 Hạn chế của phương pháp giám sát thủ công.....	5
1.3 Ứng dụng Trí tuệ nhân tạo, Học máy, Học sâu và Thị giác máy tính trong giám sát giao thông:	7
1.3.1 Trí tuệ nhân tạo (Artificial Intelligence – AI)	7
1.3.2 Học máy (Machine Learning – ML).....	8
1.3.3 Học sâu (Deep Learning – DL).....	9
1.3.4 Thị giác máy tính (Computer Vision – CV)	10
1.3.5 Mối quan hệ giữa Trí tuệ nhân tạo, Học máy, Học sâu và Thị giác máy tính trong giám sát giao thông	11
1.4 Khảo sát các công nghệ và mô hình liên quan	13
1.4.1 Các mô hình phát hiện đối tượng (Object Detection).....	13
1.4.2 Mô hình theo dõi đối tượng (Object Tracking)	14
1.4.3 Công nghệ nhận dạng biển số (License Plate Recognition)	14
1.4.4 Các mô hình nhận diện hành vi và trạng thái vi phạm	15
1.4.5 Một số công nghệ hỗ trợ và xử lý dữ liệu video.....	15
CHƯƠNG 2: CÁC THUẬT TOÁN VÀ MÔ HÌNH HỌC SÂU SỬ DỤNG TRONG HỆ THỐNG.....	16
2.1 Mạng nơ-ron nhân tạo (Artificial Neural Network – ANN).....	16
2.1.1 Cấu trúc cơ bản của mạng nơ-ron.....	17
2.1.2 Nguyên lý hoạt động của nơ-ron	19
2.1.3 Quá trình học của mạng nơ-ron	21

2.1.4 Ưu điểm và hạn chế của mạng nơ-ron.....	23
2.2 Mạng nơ-ron tích chập (Convolutional Neural Network – CNN)	24
2.2.1 Cấu trúc cơ bản của CNN	25
2.2.2 Nguyên lý hoạt động của CNN.....	30
2.2.3 Ưu điểm và hạn chế của mạng CNN	31
2.3 Mạng nơ-ron hồi quy (Recurrent Neural Network – RNN) và LSTM	32
2.3.1 Cấu trúc của mạng nơ-ron hồi quy	33
2.3.2 Nguyên lý hoạt động mạng nơ-ron hồi quy (RNN).....	35
2.3.3 Ưu điểm và Nhược điểm của mạng nơ-ron hồi quy (RNN).....	36
2.3.4 Mạng LSTM (Long Short-Term Memory).....	37
2.4 Kiến trúc YOLOv8 – Nền tảng chính của hệ thống	40
2.4.1 Tổng Quan Về Kiến trúc YOLOv8 :	40
2.4.2 Cấu Trúc Tổng Thể Của YOLOv8 Trong Quá Trình Huấn Luyện.	43
2.4.3 Cơ Chế Anchor-Free Detection	44
2.4.4 Cơ Chế Task-Aligned Assigner (TAL)	45
2.4.5 Hàm Mất Mát (Loss Function) trong YOLOv8.....	46
2.4.6 Chiến Lược Tối Ưu (Optimization Strategy)	47
2.4.7 Ưu điểm và hạn chế YOLOv8	47
CHƯƠNG 3. XÂY DỰNG TRIỂN KHAI HỆ THỐNG	50
3.1. Tổng quan hệ thống	50
3.2. Tiền xử lý dữ liệu	51
3.2.1. Thu thập và chú thích dữ liệu	51
3.2.2. Chuẩn hóa dữ liệu đầu vào	51
3.3. Môi trường sử dụng.....	54
3.4 Một số kết quả đạt được.....	57
KẾT LUẬN	68
TÀI LIỆU THAM KHẢO	70

DANH MỤC HÌNH ẢNH

<i>Hình 1.1: Tình trạng giao thông đông đúc tại Hà Nội giờ cao điểm</i>	<i>4</i>
<i>Hình 1.2: Biểu đồ số vụ tai nạn giao thông và số người tử vong giai đoạn 2018–2024.....</i>	<i>5</i>
<i>Hình 1.3: quy trình giám sát thủ công truyền thống.....</i>	<i>6</i>
<i>Hình 1.4: hệ thống giám sát giao thông thông minh</i>	<i>12</i>
<i>Hình 2.1 : hình ảnh minh họa mạng noron nhân tạo</i>	<i>16</i>
<i>Hình 2.2 Cấu trúc của mạng noron nhân tạo</i>	<i>17</i>
<i>Hình 2.3 Hình ảnh minh họa mạng noron tích chập (CNN).....</i>	<i>24</i>
<i>Hình 2.4 Cấu trúc cơ bản của mạng noron tích chập (CNN).....</i>	<i>25</i>
<i>Hình 2.5 Hình ảnh minh họa mạng no-ron hồi quy (RNN)</i>	<i>32</i>
<i>Hình 2.6 Hình ảnh minh họa mạng Long Short-Term Memory.....</i>	<i>37</i>
<i>Hình 2.9 Cấu Trúc Tổng Thể Của YOLOv8 Trong Quá Trình Huấn Luyện</i>	<i>44</i>
<i>Hình 3.1 Ứng dụng Python trong lĩnh vực Trí tuệ nhân tạo.....</i>	<i>55</i>
<i>Hình 3.2 Ứng dụng YOLOv8 trong nhận dạng thông minh trên đường phố.....</i>	<i>55</i>
<i>Hình 3.3 Môi trường thiết kế giao diện người dùng Qt Designer</i>	<i>56</i>
<i>Hình 3.4 Giao diện chính của hệ thống nhận diện vi phạm giao thông</i>	<i>57</i>
<i>Hình 3.5 Chức năng nhận diện và phân loại đối tượng giao thông</i>	<i>58</i>
<i>Hình 3.6 Hình ảnh chức năng nhận diện giao thông.....</i>	<i>59</i>
<i>Hình 3.7 Hình ảnh chức năng nhận diện biển số phương tiện</i>	<i>60</i>
<i>Hình 3.8 Hình ảnh chức năng nhận diện vi phạm vượt đèn đỏ</i>	<i>61</i>
<i>Hình 3.9 Hình ảnh minh họa khi đèn xanh</i>	<i>62</i>
<i>Hình 3.10 Hình ảnh minh họa khi đèn đỏ</i>	<i>62</i>
<i>Hình 3.11 Hình ảnh chức năng báo cáo vi phạm</i>	<i>63</i>
<i>Hình 3.12 Hình ảnh chi tiết lỗi vi phạm của từng phương tiện</i>	<i>64</i>

DANH MỤC BẢNG BIỂU

Bảng 1.1 Mối quan hệ các công nghệ AI trong giám sát giao thông	12
Bảng 1.2 So sánh giám sát thủ công và giám sát bằng AI.....	13
Bảng 2.1 Úng dụng ANN, CNN, RNN và LSTM trong bài toán giao thông.....	39
Bảng 2.2 So sánh các mô hình ANN, CNN, RNN và LSTM.....	40
Bảng 2.3 So sánh các mô hình YOLOv5 – YOLOv7 – YOLOv8.....	49
Bảng 3.1 : các thư viện sửa dụng trong mô hình.....	50
Bảng 3.2 cấu hình máy tính triển khai mô hình	51
Bảng 3.5 bảng kết quả kiểm thử hiệu suất	67

DANH MỤC KÝ HIỆU – VIẾT TẮT

Viết tắt	Thuật ngữ đầy đủ	Nghĩa
AI	Artificial Intelligence	Trí tuệ nhân tạo
ML	Machine Learning	Học máy
DL	Deep Learning	Học sâu
CV	Computer Vision	Thị giác máy tính
ANN	Artificial Neural Network	Mạng nơ-ron nhân tạo
CNN	Convolutional Neural Network	Mạng nơ-ron tích chập
RNN	Recurrent Neural Network	Mạng nơ-ron hồi quy
ROI	Region of Interest	Vùng quan tâm
FPS	Frames Per Second	Số khung hình mỗi giây
OCR	Optical Character Recognition	Nhận dạng ký tự quang học
ALPR	Automatic License Plate Recognition	Nhận dạng biển số tự động
IoU	Intersection over Union	Độ giao nhau giữa hai khung
BCE	Binary Cross Entropy	Hàm mất mát entropy nhị phân
CSV	Comma-Separated Values	Tệp dữ liệu dạng bảng

LỜI CẢM ƠN

Lời đầu tiên, em xin gửi lời cảm ơn chân thành và sâu sắc nhất tới toàn thể các thầy cô Trường Đại học Công nghệ Thông tin và Truyền thông – Đại học Công nghiệp Hà Nội, những người đã tận tình giảng dạy, truyền đạt cho em những kiến thức quý báu, tạo nền tảng vững chắc cho quá trình học tập và nghiên cứu của em trong suốt thời gian qua.

Đặc biệt, em xin bày tỏ lòng biết ơn sâu sắc tới giảng viên TS.Trần Hùng Cường, người đã trực tiếp hướng dẫn, chỉ bảo tận tình và giúp đỡ em trong suốt quá trình thực hiện đề tài “Nhận diện vi phạm giao thông”. Những ý kiến đóng góp quý báu của thầy là nguồn động lực to lớn giúp em hoàn thiện tốt đồ án này.

Em cũng xin gửi lời cảm ơn chân thành đến gia đình và bạn bè – những người luôn ở bên cạnh, động viên, khích lệ và hỗ trợ em cả về vật chất lẫn tinh thần trong suốt quá trình học tập và thực hiện đề tài.

Trong quá trình nghiên cứu và thực hiện đồ án, do thời gian có hạn cùng với kiến thức và kinh nghiệm của bản thân còn hạn chế, bài làm của em không tránh khỏi những thiếu sót. Em rất mong nhận được sự thông cảm, góp ý quý báu từ quý thầy cô và các bạn để đồ án của em được hoàn thiện hơn.

Em xin chân thành cảm ơn!

Sinh viên thực hiện:
Trần Việt Trường

LỜI NÓI ĐẦU

Trong những năm gần đây, cùng với sự phát triển mạnh mẽ của xã hội, số lượng phương tiện giao thông ngày càng gia tăng nhanh chóng. Điều này kéo theo tình trạng vi phạm luật giao thông diễn ra ngày càng phức tạp, gây mất trật tự an toàn giao thông và để lại nhiều hậu quả nghiêm trọng cho xã hội. Công tác giám sát, phát hiện và xử lý các hành vi vi phạm giao thông hiện nay vẫn còn gặp nhiều khó khăn do phụ thuộc phần lớn vào sức người, tốn thời gian và dễ xảy ra sai sót.

Trước thực tế đó, việc ứng dụng công nghệ nhận dạng hình ảnh và trí tuệ nhân tạo (AI) trong việc phát hiện và nhận diện vi phạm giao thông là hết sức cần thiết. Từ đó, em quyết định thực hiện đề tài “Nhận diện vi phạm giao thông” với mục tiêu xây dựng một hệ thống có khả năng phân tích hình ảnh, video giao thông, tự động phát hiện và nhận diện các hành vi vi phạm như vượt đèn đỏ, đi sai làn, không đội mũ bảo hiểm,... nhằm hỗ trợ cơ quan chức năng trong công tác quản lý và xử phạt một cách chính xác, nhanh chóng và hiệu quả. Hệ thống được xây dựng trên nền tảng trí tuệ nhân tạo (AI) và xử lý ảnh với khả năng học và nhận dạng đặc trưng của phương tiện cũng như hành vi vi phạm. Bên cạnh đó, hệ thống còn hướng tới việc lưu trữ, thống kê và báo cáo dữ liệu giúp cơ quan quản lý dễ dàng tra cứu và theo dõi tình hình vi phạm giao thông.

Đề tài không chỉ có ý nghĩa thực tiễn cao trong việc góp phần nâng cao hiệu quả giám sát giao thông, mà còn giúp em vận dụng kiến thức đã học về lập trình, xử lý ảnh và học máy, học sâu vào giải quyết một bài toán thực tế. Đồng thời, đây cũng là cơ hội để em tiếp cận với các công nghệ hiện đại, nâng cao kỹ năng nghiên cứu, phân tích và triển khai hệ thống thực tế.

Báo cáo này trình bày toàn bộ quá trình nghiên cứu, thiết kế, xây dựng và thử nghiệm hệ thống “Nhận diện vi phạm giao thông”, gồm ba chương chính và phần kết luận:

CHƯƠNG 1: KHẢO SÁT HỆ THỐNG

Trong chương đầu, em sẽ trình bày bối cảnh và sự cần thiết của việc xây dựng hệ thống nhận diện vi phạm giao thông. Cụ thể, chương này sẽ phân tích thực trạng tình hình giao thông hiện nay, những hạn chế của phương pháp giám

sát thủ công như tốn nhân lực, độ chính xác chưa cao, khó lưu trữ và xử lý dữ liệu. Từ đó, em xác định bài toán đặt ra là xây dựng một hệ thống có khả năng tự động phát hiện và nhận diện các hành vi vi phạm giao thông dựa trên hình ảnh và video thu được từ camera giám sát.

Ngoài ra, chương này cũng sẽ nêu rõ mục tiêu, phạm vi nghiên cứu, và khảo sát một số giải pháp, công nghệ tương tự đã được áp dụng trong và ngoài nước nhằm tìm ra hướng tiếp cận phù hợp cho đề tài.

CHƯƠNG 2 :CÁC THUẬT TOÁN VÀ MÔ HÌNH HỌC SÂU SỬ DỤNG TRONG HỆ THỐNG:

Chương này tập trung trình bày các thuật toán và mô hình học sâu được ứng dụng trong việc xây dựng hệ thống nhận diện vi phạm giao thông vượt đèn đỏ.

Trên cơ sở kết quả khảo sát ở chương trước, nội dung chương này đi sâu vào việc phân tích nguyên lý hoạt động, cấu trúc và vai trò của từng thành phần chính trong hệ thống, bao gồm mô hình học sâu YOLOv8 kết hợp thư viện xử lý ảnh OpenCV.

Mô hình YOLOv8 (You Only Look Once, phiên bản 8) được sử dụng để phát hiện và định vị các phương tiện giao thông (ô tô, xe máy, xe buýt, xe tải...) trong từng khung hình video. YOLOv8 hoạt động dựa trên cơ chế nhận diện một lần duy nhất – “You Only Look Once” – giúp đạt tốc độ xử lý cao và độ chính xác tốt, phù hợp cho các ứng dụng giám sát thời gian thực.

Bên cạnh đó, thư viện OpenCV được sử dụng để xác định trạng thái đèn tín hiệu giao thông thông qua việc phân tích màu sắc trong vùng quan tâm (ROI), đồng thời vẽ vạch dừng và xác định hành vi vượt đèn đỏ dựa trên vị trí của phương tiện so với tọa độ vạch..

CHƯƠNG 3: XÂY DỰNG TRIỂN KHAI HỆ THỐNG

Trong chương này, em sẽ trình bày quá trình cài đặt tiền sử lý dữ liệu và triển khai hệ thống dựa trên mô hình đã thiết kế ở chương trước.

Hệ thống được xây dựng sử dụng các công nghệ hiện đại như Python, OpenCV, TensorFlow hoặc YOLOv8 cho việc xử lý và nhận diện hình ảnh. Phần giao diện cơ bản được thiết kế nhằm giúp người dùng hoặc cơ quan chức năng dễ dàng theo dõi, thống kê và quản lý dữ liệu vi phạm.

Em sẽ trình bày một số kết quả thử nghiệm thực tế, minh họa khả năng nhận diện

của hệ thống qua các hình ảnh/video giao thông, đồng thời đánh giá hiệu suất và độ chính xác của mô hình.

KẾT LUẬN

Phần kết luận sẽ tổng hợp các kết quả đạt được, đánh giá hiệu quả thực tế của hệ thống nhận diện vi phạm giao thông, đồng thời rút ra bài học kinh nghiệm trong quá trình nghiên cứu và triển khai đề tài.

Bên cạnh đó, em cũng sẽ đề xuất các hướng phát triển trong tương lai, như nâng cấp mô hình AI để tăng độ chính xác, tích hợp nhận dạng biển số xe tự động, mở rộng phạm vi giám sát theo khu vực, và triển khai hệ thống trên nền tảng web hoặc ứng dụng di động phục vụ công tác quản lý giao thông thông minh.

CHƯƠNG 1: KHẢO SÁT HỆ THỐNG

1.1 Thực trạng giao thông tại Việt Nam

Hiện nay, Việt Nam có hơn 70 triệu phương tiện giao thông cơ giới, trong đó phần lớn là xe máy, chiếm tới gần 90% tổng số phương tiện. Sự gia tăng nhanh chóng của phương tiện trong nhiều năm qua, đặc biệt ở các đô thị lớn như Hà Nội và TP. Hồ Chí Minh, đã gây ra áp lực nặng nề lên hệ thống giao thông vốn phát triển chậm và chưa đáp ứng kịp nhu cầu. Trong khi số lượng phương tiện tăng đều mỗi năm, hạ tầng giao thông như đường sá, bãi đỗ xe, hệ thống giao thông công cộng và các công trình phụ trợ lại mở rộng hạn chế do thiếu quỹ đất và vốn đầu tư. Điều này dẫn đến tình trạng ùn tắc kéo dài, ô nhiễm không khí gia tăng và nguy cơ tai nạn giao thông luôn ở mức cao. Bối cảnh đó đặt ra thách thức lớn cho công tác quản lý, quy hoạch giao thông và đòi hỏi các giải pháp đồng bộ nhằm đảm bảo an toàn và tính bền vững cho hệ thống giao thông trong tương lai.

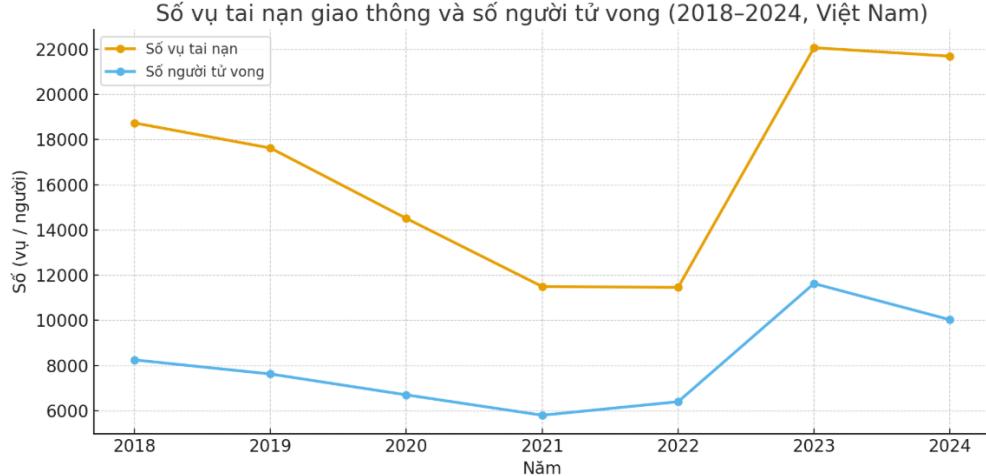


(Hình 1.1: Tình trạng giao thông đông đúc tại Hà Nội giờ cao điểm)

Theo Tổ chức Y tế Thế giới (WHO) và Ủy ban An toàn Giao thông Quốc gia, tình hình tai nạn giao thông ở Việt Nam vẫn đáng báo động:

- Trung bình mỗi năm xảy ra trên 16.000 vụ tai nạn giao thông, làm chết hơn 7.000 người và làm bị thương hơn 12.000 người.

- Mức thiệt hại kinh tế do tai nạn giao thông ước tính chiếm 2,5 – 3% GDP quốc gia mỗi năm.
- Trong đó, vượt đèn đỏ chiếm gần 15% các nguyên nhân trực tiếp gây tai nạn nghiêm trọng tại các ngã tư đô thị.



(Hình 1.2: Biểu đồ số vụ tai nạn giao thông và số người tử vong giai đoạn 2018–2024)

Thực tế tại các thành phố lớn như Hà Nội hay TP. Hồ Chí Minh, tình trạng vi phạm giao thông tại các ngã tư có đèn tín hiệu diễn ra khá phổ biến. Nhiều người điều khiển phương tiện cố tình vượt vạch dừng dù đèn đã chuyển sang đỏ, hoặc đi sai làn quy định khi chuẩn bị rẽ. Một số trường hợp thậm chí tăng tốc đột ngột khi thấy đèn vàng với tâm lý “kịp thì đi”, bất chấp nguy cơ va chạm với các phương tiện từ hướng khác.

Những hành vi này không chỉ tiềm ẩn nguy hiểm cho chính người điều khiển mà còn dễ dẫn đến tai nạn nghiêm trọng, đặc biệt trong điều kiện đường đông và tầm nhìn hạn chế. Ngoài ra, việc không tuân thủ đèn tín hiệu còn gây hỗn loạn lưu thông, khiến những người chấp hành đúng luật bị cản trở và làm giảm hiệu quả điều tiết giao thông của hệ thống đèn. Nếu tình trạng này không được kiểm soát, sẽ tạo ra thói quen xấu, ảnh hưởng lâu dài đến văn hóa giao thông đô thị và gây áp lực lớn lên lực lượng chức năng trong việc xử lý vi phạm.

1.2 Hạn chế của phương pháp giám sát thủ công

Hiện nay, phương pháp giám sát và phát hiện vi phạm giao thông tại các ngã tư chủ yếu dựa trên hệ thống camera an ninh kết hợp với quá trình quan sát thủ công của lực lượng chức năng. Mặc dù đây là cách tiếp cận truyền thống và

tương đối phổ biến, nó vẫn tồn tại nhiều hạn chế ảnh hưởng đến hiệu quả quản lý:

- Phụ thuộc lớn vào yếu tố con người: Việc nhân viên phải theo dõi hình ảnh liên tục trong thời gian dài khiến họ dễ rơi vào trạng thái mệt mỏi, giảm khả năng tập trung. Điều này dẫn đến nguy cơ bỏ sót các hành vi vi phạm hoặc đánh giá không chính xác do yếu tố chủ quan.
- Không thể xử lý khối lượng dữ liệu quá lớn: Tại các đô thị đông đúc, mỗi ngày hệ thống camera ghi lại hàng nghìn giờ video. Việc rà soát thủ công toàn bộ dữ liệu gần như bất khả thi, gây quá tải cho lực lượng giám sát và khiến nhiều trường hợp vi phạm không được phát hiện.
- Thiếu tính kịp thời trong phát hiện và phản ứng: Mặc dù camera ghi lại toàn bộ sự việc, nhưng nhân viên không thể theo dõi cùng lúc tất cả các điểm giao cắt. Do đó, khi một hành vi vi phạm xảy ra, hệ thống hiện tại thường không đưa ra cảnh báo ngay lập tức, khiến lực lượng chức năng khó can thiệp hoặc xử lý tại chỗ.
- Khó khăn trong lưu trữ và truy xuất bằng chứng: Dữ liệu video lớn yêu cầu hệ thống lưu trữ dung lượng cao và quy trình quản lý chặt chẽ. Tuy nhiên, trong thực tế, việc sắp xếp, tìm kiếm và trích xuất đoạn video phục vụ xử phạt hoặc điều tra thường mất nhiều thời gian. Hơn nữa, nhiều nơi lưu trữ không đồng bộ, thiếu chuẩn hóa, gây cản trở cho việc sử dụng bằng chứng khi cần thiết.



(Hình 1.3: quy trình giám sát thủ công truyền thống)

1.3 Ứng dụng Trí tuệ nhân tạo, Học máy, Học sâu và Thị giác máy tính trong giám sát giao thông:

Trong những năm gần đây, sự phát triển mạnh mẽ của Trí tuệ nhân tạo (AI) và các kỹ thuật liên quan như Học máy (Machine Learning), Học sâu (Deep Learning) và Thị giác máy tính (Computer Vision) đã tạo ra bước đột phá lớn trong lĩnh vực giám sát giao thông. Các hệ thống giám sát truyền thống phụ thuộc nhiều vào quan sát thủ công, dễ bị giới hạn về khả năng xử lý và tính chính xác. Việc ứng dụng AI cho phép tự động hóa toàn diện, từ phát hiện – phân tích – ghi nhận vi phạm đến lưu trữ và báo cáo dữ liệu.

1.3.1 Trí tuệ nhân tạo (Artificial Intelligence – AI)

Trí tuệ nhân tạo (AI) là một lĩnh vực khoa học liên ngành, kết hợp giữa khoa học máy tính, toán học, dữ liệu và các mô hình học máy nhằm xây dựng những hệ thống có khả năng thực hiện các chức năng vốn chỉ con người mới làm được. Những chức năng này bao gồm nhận thức, học hỏi từ kinh nghiệm, phân tích – suy luận, và ra quyết định dựa trên thông tin thu thập được. Mục tiêu của AI là tạo ra các thuật toán và hệ thống có khả năng tự động xử lý, thích ứng và cải thiện theo thời gian mà không cần sự can thiệp thủ công liên tục.

Trong lĩnh vực giám sát giao thông, trí tuệ nhân tạo giữ vai trò như “bộ não trung tâm” của toàn bộ hệ thống quan sát và xử lý dữ liệu. Nhờ vào các mô hình học sâu (deep learning) và thị giác máy tính (computer vision), AI có thể thực hiện nhiều nhiệm vụ quan trọng, bao gồm:

- Phân tích dữ liệu hình ảnh/video theo thời gian thực: AI thu thập và xử lý liên tục dòng dữ liệu từ camera quan sát giao thông, giúp hệ thống phát hiện biến động và phản ứng tức thời.
- Nhận diện tình huống giao thông: AI xác định các sự kiện xảy ra trên đường như ùn tắc, rẽ sai làn, dừng đỗ trái phép, chạy quá tốc độ, vượt đèn đỏ hoặc các tình huống bất thường khác.
- Đánh giá hành vi phương tiện dựa trên mô hình đã học: Hệ thống hiểu được “mẫu hành vi” của phương tiện, từ đó so sánh với các quy định giao thông để xác định xem hành vi đó có vi phạm hay không.

- Hỗ trợ ra quyết định: AI cung cấp các thông tin phân tích chính xác, làm cơ sở cho cơ quan chức năng xác định lỗi vi phạm, đưa ra cảnh báo, hoặc điều tiết giao thông hợp lý hơn.

Nhờ sự hỗ trợ của AI, hệ thống giám sát giao thông có thể vận hành liên tục 24/7, ổn định và chính xác, không bị ảnh hưởng bởi yếu tố thời tiết, điều kiện ánh sáng hay sự mệt mỏi của con người. Điều này giúp nâng cao hiệu quả quản lý giao thông, giảm thiểu tai nạn, phát hiện vi phạm kịp thời và góp phần xây dựng một môi trường giao thông an toàn, văn minh.

1.3.2 Học máy (Machine Learning – ML)

Học máy (Machine Learning – ML) là một nhánh quan trọng của trí tuệ nhân tạo, tập trung vào việc phát triển các thuật toán cho phép máy tính tự học từ dữ liệu, thay vì phải lập trình thủ công cho từng tình huống cụ thể. Thông qua việc xử lý các tập dữ liệu lớn, Học máy có khả năng nhận diện quy luật, mô hình hóa mối quan hệ giữa các yếu tố, và từ đó đưa ra kết luận, phân loại, hoặc dự đoán một cách tự động. Càng được cung cấp nhiều dữ liệu, các mô hình học máy càng trở nên chính xác và hiệu quả hơn.

Trong các hệ thống giám sát giao thông thông minh, Học máy là nền tảng giúp nâng cao khả năng hiểu và xử lý dữ liệu hình ảnh/video từ camera. Một số ứng dụng điển hình của học máy trong môi trường này bao gồm:

- Phân loại phương tiện: Học máy giúp nhận diện và phân chia chính xác các loại phương tiện như xe máy, ô tô con, xe tải, xe buýt... dựa trên đặc trưng hình dạng, kích thước và chuyển động.
- Nhận dạng mẫu hành vi: Dựa trên dữ liệu di chuyển của phương tiện, thuật toán có thể nhận biết các hành vi như dừng đột ngột, chuyển làn, rẽ, tăng tốc bất thường hoặc các hành vi gây nguy hiểm khác.
 - Dự đoán khả năng xảy ra vi phạm: Học máy có thể phân tích chuỗi hành vi để dự đoán nguy cơ vi phạm trước khi chúng thực sự xảy ra, hỗ trợ cơ quan quản lý phát hiện sớm và ngăn chặn rủi ro.
 - Liên tục cải thiện hiệu suất mô hình: Khi dữ liệu mới được bổ sung – ví dụ như các loại phương tiện mới hoặc các tình huống giao thông chưa từng thấy

– mô hình học máy có thể được đào tạo lại (training) để thích ứng, giúp hệ thống ngày càng chính xác và thông minh hơn.

Nhờ khả năng tự thích nghi và học hỏi liên tục, Học máy giúp hệ thống giám sát giao thông không chỉ hoạt động ổn định mà còn linh hoạt trong nhiều bối cảnh giao thông phức tạp. Điều này đặc biệt hữu ích tại các đô thị lớn, nơi mật độ phương tiện cao và hành vi tham gia giao thông rất đa dạng.

1.3.3 Học sâu (Deep Learning – DL)

Học sâu (Deep Learning – DL) là một lĩnh vực mở rộng của Học máy, tập trung vào việc xây dựng và huấn luyện các mạng nơ-ron nhân tạo nhiều lớp (Deep Neural Networks). Nhờ cấu trúc nhiều tầng xử lý liên tiếp, học sâu có khả năng tự động trích xuất đặc trưng từ dữ liệu thô mà không cần con người can thiệp nhiều vào quá trình thiết kế đặc trưng (feature engineering). Điều này giúp mô hình có thể xử lý những dạng dữ liệu phức tạp như hình ảnh, âm thanh, video, hoặc chuỗi thời gian với hiệu suất vượt trội.

Trong giám sát giao thông thông minh, Deep Learning đóng vai trò chủ đạo vì nó mang lại độ chính xác và tốc độ xử lý rất cao khi làm việc với hình ảnh và video. Một số ứng dụng quan trọng gồm:

- Phát hiện phương tiện: Deep Learning có thể nhận diện chính xác xe máy, ô tô, xe tải, xe buýt... ngay cả trong các điều kiện khó khăn như ánh sáng yếu, trời mưa, góc quay phức tạp hoặc môi trường giao thông đông đúc.
- Theo dõi đối tượng: Nhờ các thuật toán theo dõi (tracking) như DeepSORT, ByteTrack..., hệ thống có thể bám theo đường di chuyển của nhiều phương tiện cùng lúc, tạo ra chuỗi thông tin liên tục về vị trí và tốc độ của từng đối tượng.
- Nhận diện biển số: Mạng nơ-ron được huấn luyện để đọc và tách ký tự trong biển số, xử lý cả trường hợp biển số bị nghiêng, mờ hoặc bị che khuất một phần trong quá trình ghi hình.
- Nhận diện hành vi vi phạm: Deep Learning giúp mô hình phân tích chuỗi chuyển động của phương tiện để phát hiện các hành vi như vượt đèn đỏ, đi

sai làn đường, dừng đỗ không đúng vị trí, đi vào khu vực cấm hoặc các hành vi nguy hiểm khác.

Một số kiến trúc mạng nơ-ron phổ biến và đang được sử dụng rộng rãi bao gồm:

- YOLO (You Only Look Once): nổi bật với tốc độ xử lý cực nhanh, phù hợp cho giám sát thời gian thực.
- Faster R-CNN: mang lại độ chính xác cao trong nhận diện và phân vùng đối tượng.
- ResNet: nổi tiếng với khả năng học sâu mà không gặp vấn đề suy giảm gradient, giúp nâng cao hiệu quả trong phân loại hình ảnh.

Nhờ sự kết hợp giữa tốc độ, độ chính xác và khả năng tự học mạnh mẽ, Deep Learning đã trở thành công nghệ cốt lõi trong các hệ thống giám sát giao thông hiện đại, giúp tự động hóa quá trình theo dõi, nhận diện và phát hiện vi phạm một cách thông minh và hiệu quả.

1.3.4 Thị giác máy tính (Computer Vision – CV)

Thị giác máy tính (Computer Vision – CV) là một lĩnh vực quan trọng của trí tuệ nhân tạo, cho phép máy tính và hệ thống tự động “nhìn thấy”, phân tích, và hiểu nội dung của hình ảnh hoặc video tương tự như cách con người quan sát thế giới. Thông qua việc sử dụng các thuật toán xử lý ảnh kết hợp với học máy và học sâu, Computer Vision có khả năng trích xuất thông tin hữu ích từ dữ liệu hình ảnh thô, biến chúng thành các dữ liệu có cấu trúc phục vụ cho phân tích và ra quyết định.

Trong hệ thống giám sát giao thông, Thị giác máy tính đóng vai trò là công nghệ nền tảng, đảm bảo toàn bộ quá trình quan sát và phân tích diễn ra một cách chính xác. Một số nhiệm vụ chính của Computer Vision bao gồm:

- Phát hiện và định vị phương tiện trong khung hình: Thị giác máy tính xác định vị trí của xe máy, ô tô, xe tải... thông qua các hộp bao (bounding boxes) hoặc mặt nạ phân vùng (segmentation masks), giúp hệ thống biết chính xác phương tiện đang ở đâu trên ảnh.

- Nhận biết vạch kẻ đường, biển báo, đèn tín hiệu giao thông: Đây là cơ sở để đánh giá hành vi phương tiện, ví dụ như vượt qua vạch dừng, đi sai làn, hay không tuân thủ tín hiệu đèn.
- Theo dõi quỹ đạo chuyển động: Thị giác máy tính có khả năng ghi nhận đường di chuyển của từng phương tiện qua nhiều khung hình liên tiếp, từ đó phân tích tốc độ, hướng di chuyển và hành vi tổng thể.
- Tách đối tượng khỏi nền (Segmentation): Việc phân biệt rõ phương tiện và môi trường xung quanh giúp mô hình đưa ra kết quả chính xác hơn, đặc biệt trong môi trường giao thông đông đúc hoặc điều kiện ánh sáng phức tạp.
- Xác định thời điểm và vị trí vi phạm: Kết hợp nhiều thuật toán, Thị giác máy tính giúp xác định chính xác hành vi vi phạm xảy ra ở đâu, khi nào và với phương tiện nào.

Có thể xem Computer Vision như “đôi mắt” của hệ thống giám sát giao thông. Nó thu thập và cung cấp nguồn dữ liệu đầu vào quan trọng cho các mô hình AI và Deep Learning xử lý, nhận diện và đánh giá vi phạm. Nếu không có Thị giác máy tính, hệ thống sẽ không thể quan sát thế giới thực, cũng như không thể phân tích chính xác các tình huống giao thông.

1.3.5 Mối quan hệ giữa Trí tuệ nhân tạo, Học máy, Học sâu và Thị giác máy tính trong giám sát giao thông

Bốn công nghệ Trí tuệ nhân tạo, Học máy, Học sâu và Thị giác máy tính không tồn tại một cách tách biệt mà kết hợp chặt chẽ với nhau để tạo thành một hệ thống giám sát giao thông thông minh hoàn chỉnh. Mỗi thành phần đảm nhận một chức năng riêng, nhưng khi được tích hợp sẽ hình thành một quy trình xử lý khép kín từ việc thu nhận dữ liệu, phân tích đến đưa ra quyết định

Thành phần	Vai trò	Ý nghĩa trong giám sát giao thông
Trí tuệ nhân tạo	Tư duy, phân tích và ra quyết định	Đánh giá tình huống giao thông và xác định hành vi vi phạm
Học máy	Học từ dữ liệu	Nhận dạng các mẫu hành vi

		và tối ưu hiệu suất mô hình theo thời gian
Học sâu	Học trên dữ liệu lớn bằng nhiều tầng nơ-ron nhân tạo	Phát hiện, theo dõi phương tiện với độ chính xác và tốc độ cao
Thị giác máy tính	Nhìn và hiểu nội dung hình ảnh, video	Nhận diện đối tượng, vạch kẻ đường, biển báo, đèn tín hiệu trong thời gian thực

Bảng 1.1 Mối quan hệ các công nghệ AI trong giám sát giao thông

Việc ứng dụng Trí tuệ nhân tạo, học máy, học sâu và thị giác máy tính trong giám sát giao thông mang lại nhiều lợi ích vượt trội so với phương pháp thủ công. Hệ thống có khả năng tự động phát hiện vi phạm dựa trên việc phân tích hình ảnh và video theo thời gian thực, giúp giảm thiểu đáng kể sự phụ thuộc vào quan sát của con người và giảm tải nhân lực cho lực lượng chức năng. Đồng thời, các dữ liệu vi phạm được ghi nhận và xử lý một cách hoàn toàn tự động, hạn chế tối đa sai sót hoặc yếu tố cảm tính, từ đó tăng tính minh bạch và khách quan trong quá trình xử phạt. Bên cạnh đó, hệ thống hoạt động liên tục 24/7, không bị ảnh hưởng bởi điều kiện thời tiết hay thời gian, đảm bảo giám sát hiệu quả ở mọi thời điểm. Đặc biệt, nhờ khả năng tính toán mạnh mẽ của các mô hình học sâu, hệ thống còn xử lý được lượng lớn dữ liệu giao thông mỗi ngày, đáp ứng nhu cầu giám sát tại những khu vực đông phương tiện và có mật độ vi phạm cao.



(Hình 1.4: hệ thống giám sát giao thông thông minh)

So sánh giám sát thủ công với giám sát bằng trí tuệ nhân tạo

Tiêu chí	Giám sát thủ công	Giám sát bằng AI
Độ chính xác	Thấp – phụ thuộc con người	Cao – ổn định 24/7
Tốc độ xử lý	Chậm	Nhanh, thời gian thực
Tốn nhân lực	Nhiều	Ít
Khả năng lưu trữ	Khó khăn	Tự động – có hệ thống
Tính khách quan	Thấp	Cao

Bảng 1.2 So sánh giám sát thủ công và giám sát bằng AI

1.4 Khảo sát các công nghệ và mô hình liên quan

Trong những năm gần đây, sự phát triển mạnh mẽ của thị giác máy tính (Computer Vision) và học sâu (Deep Learning) đã mở ra nhiều hướng tiếp cận mới cho các bài toán giám sát và phân tích giao thông. Đặc biệt, các công nghệ như mạng nơ-ron tích chập (CNN), mô hình phát hiện đối tượng thời gian thực, mô hình theo dõi đa đối tượng hay nhận dạng biến số đã trở thành những thành phần cốt lõi trong các hệ thống nhận diện vi phạm giao thông hiện đại. Phần này trình bày khảo sát tổng quan về các công nghệ và mô hình đang được sử dụng rộng rãi.

1.4.1 Các mô hình phát hiện đối tượng (Object Detection)

Phát hiện đối tượng là bước nền tảng để nhận diện phương tiện, người đi bộ và các tác nhân liên quan. Các mô hình nổi bật gồm:

- Họ YOLO (You Only Look Once): bao gồm YOLOv5, YOLOv7, YOLOv8, YOLOv9 và YOLOv10. Các mô hình YOLO có ưu điểm tốc độ xử lý cao, độ chính xác tốt và khả năng hoạt động ổn định trong môi trường thời gian thực. YOLOv8 và YOLOv10 là các phiên bản mới, sử dụng kiến trúc tối ưu và cơ chế dự đoán anchor-free nhằm nâng cao hiệu suất phát hiện.

- Faster R-CNN: mô hình hai giai đoạn (two-stage) với độ chính xác cao, phù hợp môi trường ít yêu cầu thời gian thực.

- SSD (Single Shot Detector): mô hình một giai đoạn, gọn nhẹ, tốc độ tốt, thường được sử dụng trên các thiết bị IoT hoặc camera nhúng.

Các mô hình phát hiện đối tượng được dùng để xác định xe, người, biển báo... tạo nền tảng cho việc phân tích hành vi vi phạm.

1.4.2 Mô hình theo dõi đối tượng (Object Tracking)

Để nhận diện hành vi vi phạm, hệ thống cần theo dõi phương tiện theo thời gian. Một số mô hình tiêu biểu:

- DeepSORT: kết hợp thông tin hình dạng (appearance) và vị trí, theo dõi ổn định trong nhiều khung hình.
- ByteTrack: nổi bật ở môi trường đông phương tiện nhờ giữ lại các bounding box có độ tin cậy thấp.
- OC-SORT, StrongSORT: cải thiện độ mượt và tính ổn định của đường đi (trajectory).

Những mô hình này cho phép xác định chính xác đường di chuyển và hành động của phương tiện, phục vụ cho các bài toán như vượt đèn đỏ, lấn làn hay dừng đỗ sai quy định.

1.4.3 Công nghệ nhận dạng biển số (License Plate Recognition)

Giống như hệ thống phạt nguội, mô hình nhận dạng biển số giúp định danh phương tiện vi phạm:

- LPDNet, LPRNet: chuyên biệt cho nhận diện và đọc biển số.
- CRNN (Convolutional Recurrent Neural Network): trích xuất đặc trưng bằng CNN và giải mã chuỗi ký tự bằng RNN–CTC.
- PaddleOCR, EasyOCR: công cụ OCR mạnh, hỗ trợ nhiều định dạng biển số.

Công nghệ này giúp hệ thống tự động ghi nhận thông tin phương tiện vi phạm mà không cần can thiệp thủ công.

1.4.4 Các mô hình nhận diện hành vi và trạng thái vi phạm

Một số hành vi cần phân tích nâng cao:

- Không đội mũ bảo hiểm: nhận diện đầu + mũ bằng YOLO hoặc MobileNet kết hợp phân lớp.
- Vượt đèn đỏ: phân tích vị trí phương tiện qua thời gian bằng YOLO + tracking + nhận diện vùng cấm (ROI).
- Sử dụng điện thoại khi lái xe: nhận diện tư thế tay và điện thoại bằng pose estimation (BlazePose, MediaPipe).
- Chở quá số người: phát hiện người trên xe kết hợp mô hình pose estimation.

Ngoài ra, các mô hình phân tích video như I3D, TimeSformer, C3D có thể được sử dụng để nhận diện hành vi phức tạp trong chuỗi khung hình.

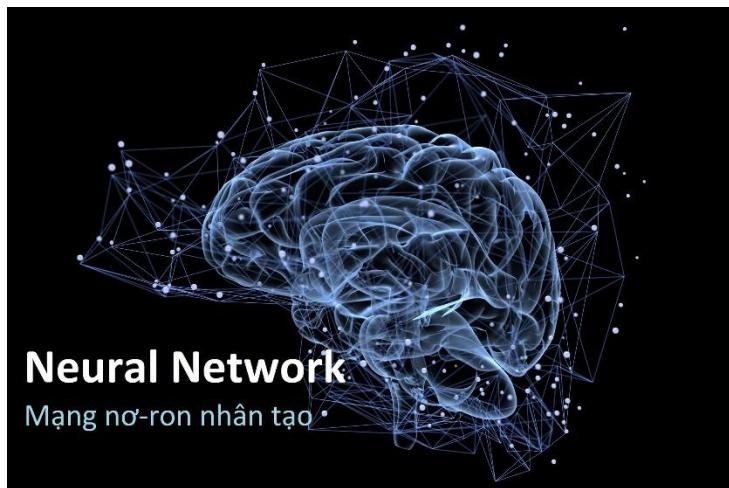
1.4.5 Một số công nghệ hỗ trợ và xử lý dữ liệu video

- OpenCV: Được sử dụng để xử lý ảnh và video ở tất cả các giai đoạn, từ việc đọc và trích xuất từng khung hình, chuyển đổi không gian màu, đến vẽ bounding box và hiển thị kết quả suy luận trực tiếp lên khung hình. OpenCV cũng hỗ trợ các thao tác tiền xử lý đầu vào giúp mô hình AI hoạt động ổn định và chính xác hơn.
- NumPy / Pandas:Hai thư viện này hỗ trợ mạnh trong việc phân tích và xử lý dữ liệu dưới dạng mảng và bảng. Chúng được dùng để thống kê kết quả mô hình, tính toán các chỉ số đánh giá, đồng thời lưu trữ dữ liệu vi phạm dưới dạng bảng để thuận tiện cho việc truy xuất và trực quan hóa hiệu suất.
- TensorRT / ONNX Runtime:Đây là các công cụ giúp tăng tốc quá trình suy luận của mô hình AI. Chúng tối ưu hóa mạng nơ-ron, giảm độ trễ và cải thiện tốc độ xử lý, đặc biệt quan trọng khi triển khai hệ thống trong môi trường thực tế yêu cầu tốc độ thời gian thực như giám sát giao thông.
- GStreamer / FFmpeg: Được sử dụng để xử lý và truyền tải luồng video trực tiếp từ camera IP hoặc các nguồn phát khác. Chúng giúp hệ thống ổn định khi làm việc với video thời gian thực, đảm bảo tốc độ khung hình ổn định và hỗ trợ tốt khi cần mở rộng quy mô hoặc tích hợp nhiều luồng camera cùng lúc.

CHƯƠNG 2: CÁC THUẬT TOÁN VÀ MÔ HÌNH HỌC SÂU SỬ DỤNG TRONG HỆ THỐNG

2.1 Mạng nơ-ron nhân tạo (Artificial Neural Network – ANN)

Mạng nơ-ron nhân tạo (Artificial Neural Network – ANN) là một mô hình tính toán mô phỏng cách hoạt động của hệ thần kinh sinh học, đặc biệt là não bộ con người. Trong hệ thần kinh sinh học, mỗi nơ-ron nhận tín hiệu từ nhiều nơ-ron khác, xử lý thông tin và quyết định có phát xung (tín hiệu) sang các nơ-ron tiếp theo hay không.

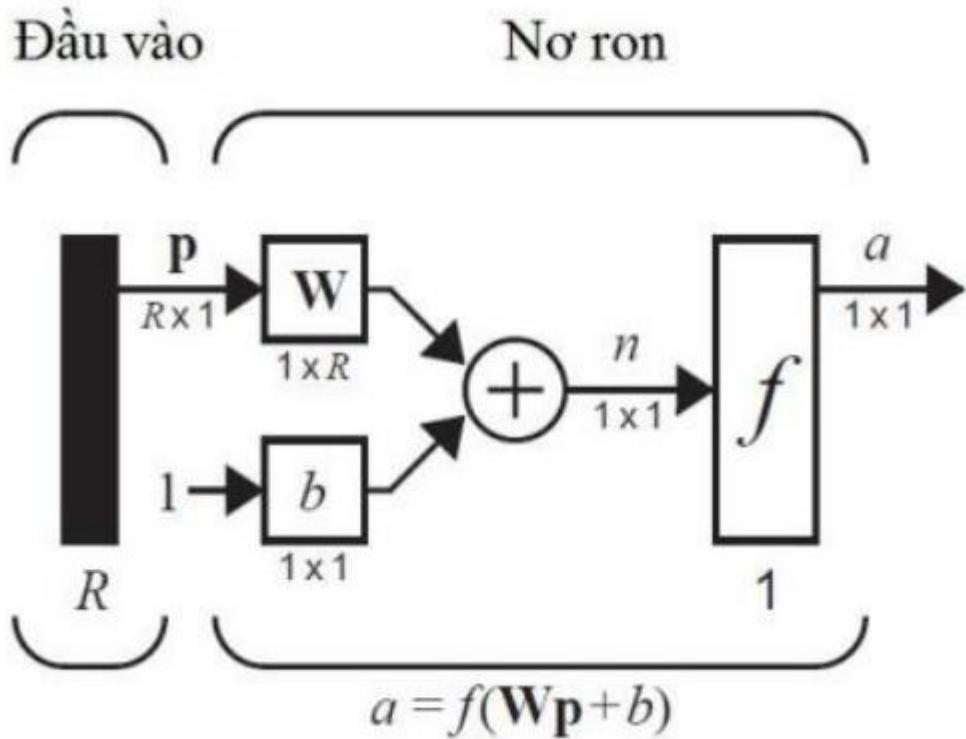


Hình 2.1 : hình ảnh minh họa mạng nơron nhân tạo

Tương tự, trong mạng nơ-ron nhân tạo, mỗi nơ-ron nhân tạo là một đơn vị xử lý đơn giản, nhận dữ liệu đầu vào, nhân với trọng số (weights), cộng với độ lệch (bias), sau đó đi qua một hàm kích hoạt (activation function) để tạo ra đầu ra. Các nơ-ron được tổ chức thành nhiều lớp (layers), liên kết với nhau tạo thành một mạng có khả năng:

- Học từ dữ liệu (learning from data),
- Nhận diện và ghi nhớ các mẫu (patterns),
- Thực hiện phân loại (classification), hồi quy (regression) hoặc các tác vụ ra quyết định khác.

2.1.1 Cấu trúc cơ bản của mạng nơ-ron



Hình 2.2 Cấu trúc của mạng nơron nhân tạo

Một mạng nơ-ron nhân tạo điển hình bao gồm ba nhóm lớp:

- Lớp đầu vào (Input Layer)
- Các lớp ẩn (Hidden Layers)
- Lớp đầu ra (Output Layer)

a) Lớp đầu vào (Input Layer)

Lớp đầu vào là tầng đầu tiên của mạng nơ-ron, có nhiệm vụ tiếp nhận trực tiếp dữ liệu thô từ môi trường hoặc từ bước tiền xử lý trước đó. Mỗi nút trong lớp này biểu diễn một đặc trưng của dữ liệu đầu vào, chẳng hạn như giá trị pixel của ảnh (grayscale hoặc RGB), các tọa độ hình học (x, y) , (x, y, z) , hay các thông số dạng số như tốc độ, hướng di chuyển và mật độ phuong tiện. Dữ liệu đầu vào thường được tổ chức dưới dạng một vector $p \in \mathbb{R}^n$ hoặc một tensor đối với ảnh, ví dụ như kích thước ảnh RGB được mô tả bởi tensor $[H \times W \times 3]$. Khác với các

lớp ẩn, lớp đầu vào không có trọng số, không có bias và không dùng hàm kích hoạt. Nó không thực hiện phép biến đổi nào mà chỉ đóng vai trò truyền dữ liệu sang lớp đầu tiên có trọng số, nơi các giá trị đầu vào sẽ bắt đầu được xử lý thông qua biểu thức cơ bản của một neuron:

$$n = Wp + b.$$

Nhờ đó, lớp đầu vào đảm bảo dữ liệu được đưa vào mạng đúng định dạng để các lớp phía sau có thể học và trích xuất đặc trưng một cách hiệu quả.

b) Lớp ẩn (Hidden Layers)

Lớp ẩn là nơi diễn ra phần lớn "trí thông minh" của mạng nơ-ron, bởi đây là tầng chịu trách nhiệm học và biểu diễn các đặc trưng từ dữ liệu. Khi nhận dữ liệu từ lớp đầu vào, các lớp ẩn đầu tiên thực hiện trích xuất đặc trưng cơ bản như cạnh, góc, đường cong hoặc các mẫu chuyển động từ dữ liệu thô (chẳng hạn ảnh từ camera). Ở những lớp sâu hơn, mạng thực hiện việc kết hợp các đặc trưng đơn giản này để tạo thành các đặc trưng phức tạp hơn, Nhờ các hàm kích hoạt phi tuyến, mạng có khả năng mô phỏng những quan hệ phi tuyến giữa đầu vào và đầu ra, điều mà các mô hình tuyến tính không thể thực hiện. Hoạt động của mỗi nơ-ron trong lớp ẩn tuân theo công thức tính toán:

$$z = \sum_{i=1}^n w_i x_i + b, a = f(z),$$

trong đó x_i là đầu vào thứ i , w_i là trọng số tương ứng, b là bias, $f(\cdot)$ là hàm kích hoạt và a là đầu ra của nơ-ron. Trong các mạng học sâu (Deep Neural Networks), số lượng lớp ẩn có thể từ vài lớp đến hàng chục hoặc hàng trăm lớp, và mỗi lớp có thể bao gồm hàng trăm đến hàng nghìn nơ-ron tùy thuộc vào độ phức tạp của bài toán cũng như khả năng tính toán của hệ thống. Chính nhờ cấu trúc nhiều lớp này mà mạng có thể học được những biểu diễn dữ liệu ngày càng trừu tượng và giàu ngữ nghĩa

c) Lớp đầu ra (Output Layer)

Lớp đầu ra là tầng cuối cùng của mạng nơ-ron, nơi tạo ra quyết định hoặc giá trị dự đoán dựa trên toàn bộ quá trình học của các lớp trước. Kiểu của lớp đầu

ra phụ thuộc trực tiếp vào loại bài toán. Với các bài toán phân loại (classification), nếu là phân loại nhị phân như “có vi phạm/không vi phạm” hoặc “có xe/không có xe”, lớp đầu ra thường chỉ có một nơ-ron và sử dụng hàm kích hoạt sigmoid để chuyển giá trị đầu ra thành xác suất, được mô tả bởi công thức:

$$a = \sigma(z) = \frac{1}{1 + e^{-z}}.$$

Trong khi đó, với bài toán phân loại đa lớp như nhận diện loại phương tiện (xe máy, ô tô, xe tải, xe buýt...), số nơ-ron ở lớp cuối sẽ bằng số lớp và thường sử dụng hàm softmax để chuẩn hóa đầu ra thành phân phối xác suất trên các lớp, với công thức:

$$a_i = \frac{e^{z_i}}{\sum_{j=1}^K e^{z_j}}.$$

Đối với các bài toán hồi quy (regression) – chẳng hạn dự đoán vận tốc, khoảng cách, thời gian đến ngã tư hay các đại lượng liên tục khác – lớp đầu ra sử dụng hàm kích hoạt tuyến tính (linear) để mô hình có thể trả về giá trị thực bất kỳ, theo biểu thức:

$$a = z.$$

Như vậy, cấu trúc của lớp đầu ra luôn được thiết kế phù hợp với bản chất của bài toán nhằm đảm bảo mô hình đưa ra dự đoán chính xác và có ý nghĩa.

2.1.2 Nguyên lý hoạt động của nơ-ron

Một nơ-ron nhân tạo có thể được xem như đơn vị tính toán cơ bản nhất của mạng nơ-ron, hoạt động tương tự như tế bào thần kinh trong não bộ con người. Về bản chất, nơ-ron này thực hiện một phép biến đổi dữ liệu từ không gian đầu vào (tập hợp các đặc trưng mô tả sự vật hoặc hiện tượng) sang một giá trị đầu ra duy nhất dùng để biểu diễn mức độ kích hoạt của nơ-ron đó.

Mỗi nơ-ron tiếp nhận nhiều tín hiệu đầu vào, ký hiệu là x_1, x_2, \dots, x_n . Mỗi đầu vào được gắn với một trọng số học được tương ứng w_1, w_2, \dots, w_n . Những trọng số này thể hiện độ quan trọng của từng đặc trưng trong quá trình dự đoán. Một trọng số lớn (dương hoặc âm) cho thấy đầu vào đó có ảnh hưởng mạnh đến quyết định của nơ-ron — trọng số dương làm tăng khả năng kích hoạt, trong khi

trọng số âm làm giảm mức kích hoạt. Trong quá trình huấn luyện mô hình, các trọng số không cố định mà được điều chỉnh liên tục thông qua thuật toán lan truyền ngược và tối ưu hóa. Mục tiêu của việc điều chỉnh này là làm cho mô hình “học” được mối quan hệ đúng đắn giữa dữ liệu đầu vào và nhãn đầu ra trong thực tế. Khi các đầu vào được đưa vào nơ-ron, chúng được kết hợp theo tổng có trọng số, được tính bằng công thức:

$$z = \sum_{i=1}^n w_i x_i + b$$

Trong đó x_i là đầu vào, w_i là trọng số thể hiện độ quan trọng của từng đầu vào, b là độ lệch giúp điều chỉnh ngưỡng kích hoạt, và toàn bộ phép cộng này tạo ra giá trị z tín hiệu thuần mà nơ-ron sử dụng trước khi đưa qua hàm kích hoạt.

Giá trị z sau đó được đưa qua một hàm kích hoạt $f(\cdot)$ để tạo ra đầu ra cuối cùng:

$$a = f(z)$$

Hàm kích hoạt đóng vai trò cực kỳ quan trọng. Nếu không có hàm kích hoạt phi tuyến, toàn bộ mạng nơ-ron dù có nhiều lớp đến đâu cũng chỉ tương đương một mô hình tuyến tính đơn giản, không thể học các quan hệ phức tạp có nhiều chiều của dữ liệu thực tế.

Một số hàm kích hoạt phổ biến:

a) Hàm sigmoid

Hàm sigmoid nén giá trị đầu ra vào khoảng $(0, 1)$, rất phù hợp cho các bài toán phân loại nhị phân hoặc dự đoán xác suất. Tuy nhiên, sigmoid có nhược điểm là dễ gây mất gradient ở các giá trị lớn.

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Hàm ReLU (Rectified Linear Unit)

Đây là một trong những hàm kích hoạt phổ biến nhất trong mạng nơ-ron sâu vì tính đơn giản, tốc độ tính toán cao và khả năng giảm thiểu hiện tượng biến mất gradient. ReLU giúp mạng có khả năng học tốt hơn trên dữ liệu lớn và nhiều tầng.

$$f(z) = \max(0, z)$$

b) Hàm tanh

Hàm tanh cho đầu ra trong khoảng (-1, 1), đối xứng quanh gốc nên thường hoạt động hiệu quả hơn sigmoid trong một số bài toán yêu cầu dữ liệu cân bằng quanh 0. Tanh cũng là lựa chọn phổ biến trong các mô hình xử lý chuỗi.

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

c) Hàm softmax

Softmax được dùng trong lớp đầu ra của các bài toán phân loại đa lớp. Hàm này biến đổi một vector bất kỳ thành một phân phối xác suất, trong đó tổng các phần tử bằng 1, giúp mô hình diễn giải mức độ thuộc về từng lớp.

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Về mặt bản chất, một nơ-ron nhân tạo chỉ kích hoạt mạnh khi giá trị z đủ lớn để vượt qua ngưỡng mà hàm kích hoạt quy định. Điều này tương đương với cơ chế neuron sinh học trong não: tế bào thần kinh chỉ phát ra tín hiệu điện khi điện thế màng vượt ngưỡng kích hoạt.

Nhờ cơ chế này, mạng nơ-ron nhân tạo có khả năng mô phỏng linh hoạt các mối quan hệ phức tạp, phi tuyến trong dữ liệu, từ đó giải quyết được nhiều bài toán phức tạp như nhận diện hình ảnh, xử lý tiếng nói, phân tích video và nhiều ứng dụng khác trong trí tuệ nhân tạo hiện đại.

2.1.3 Quá trình học của mạng nơ-ron

Quá trình học của mạng nơ-ron là quá trình tối ưu các trọng số và độ lệch dựa trên dữ liệu huấn luyện, nhằm giảm thiểu sai số giữa dự đoán của mô hình và giá trị thực tế. Đây là một chu trình lặp đi lặp lại gồm bốn bước: lan truyền xuôi, tính hàm mất mát, lan truyền ngược và cập nhật trọng số. Mỗi vòng lặp như vậy được gọi là một "iteration", còn tập hợp nhiều iteration đi hết toàn bộ dữ liệu huấn

luyện được gọi là một "epoch". Trong thực tế, mạng nơ-ron thường cần huấn luyện hàng chục đến hàng nghìn epoch để đạt được sự hội tụ.

Quá trình này thường gồm các bước lặp đi lặp lại:

- Lan truyền xuôi (Forward Propagation)
- Tính hàm mất mát (Loss Function)
- Lan truyền ngược (Backpropagation)
- Cập nhật trọng số (Weight Update)

a) Lan truyền xuôi (Forward Propagation)

$$a^{(l)} = f^{(l)}(W^{(l)}a^{(l-1)} + b^{(l)})$$

Trong bước lan truyền xuôi, dữ liệu đầu vào được đưa vào lớp đầu tiên của mạng và di chuyển qua tất cả các lớp ẩn cho đến lớp đầu ra. Ở mỗi lớp, mô hình thực hiện phép nhân giữa ma trận trọng số $W^{(l)}$ và vector đầu vào từ lớp trước $a^{(l-1)}$, cộng với độ lệch $b^{(l)}$, rồi truyền kết quả qua hàm kích hoạt $f^{(l)}$. Sau khi đi hết các lớp, mạng tạo ra dự đoán cuối cùng \hat{y} . Quá trình này mô phỏng hoạt động truyền tín hiệu trong hệ thần kinh sinh học và là cơ sở để mô hình đưa ra dự đoán.

b) Hàm mất mát (Loss Function)

Hàm mất mát đo lường mức độ sai lệch giữa dự đoán \hat{y} và nhãn thật y . Giá trị mất mát càng lớn chứng tỏ mô hình dự đoán càng sai. Một số hàm mất mát phổ biến gồm:

MSE (Mean Squared Error) cho hồi quy

$$L = \frac{1}{m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)})^2$$

Cross-Entropy Loss cho phân loại

$$L = - \sum_{i=1}^m y^{(i)} \log(\hat{y}^{(i)})$$

Với mọi mô hình học sâu, mục tiêu cốt lõi là tối thiểu hóa giá trị hàm mất mát trên toàn bộ dữ liệu huấn luyện, giúp mô hình học được mối quan hệ chính xác nhất.

c) Lan truyền ngược (Backpropagation)

$$\frac{\partial L}{\partial w_{ij}}$$

Lan truyền ngược là bước tính toán gradient của hàm mất mát theo từng trọng số và độ lệch trong mạng. Dựa vào đạo hàm và quy tắc chuỗi (chain rule), gradient được lan truyền từ lớp cuối cùng ngược về lớp đầu vào. Điều này cho biết trọng số nào góp phần gây ra sai số và mức độ đóng góp lớn hay nhỏ. Nhờ đó, mô hình biết cần chỉnh trọng số theo hướng nào để giảm sai số. Backpropagation là bước quan trọng nhất trong quá trình học vì nó cung cấp thông tin cần thiết cho bước tối ưu hóa.

d) Cập nhật trọng số (Weight Update)

$$w := w - \eta \frac{\partial L}{\partial w}$$

Ở bước này, các thuật toán tối ưu sử dụng gradient để cập nhật trọng số. Trong Gradient Descent cơ bản, trọng số được cập nhật theo hướng giảm của gradient, với η là tốc độ học, quyết định lượng thay đổi của trọng số mỗi lần cập nhật. Ngoài ra, các thuật toán hiện đại như Adam, Momentum, và RMSprop giúp quá trình học ổn định hơn, nhanh hơn và ít bị mắc kẹt tại các điểm tối thiểu cục bộ. Adam đặc biệt hiệu quả do khả năng tự điều chỉnh tốc độ học cho từng tham số dựa trên động lượng và độ lớn gradient qua thời gian.

2.1.4 Ưu điểm và hạn chế của mạng nơ-ron

Ưu điểm:

Mạng nơ-ron có khả năng học và mô hình hóa các quan hệ phi tuyến phức tạp giữa đầu vào và đầu ra, giúp phát hiện những đặc trưng ẩn mà mô hình tuyến tính khó nắm bắt. Khi được cung cấp dữ liệu đủ lớn và đa dạng, cùng với các kỹ thuật điều chuẩn phù hợp, mạng nơ-ron có thể tổng quát hóa tốt và xử lý hiệu quả những dạng dữ liệu phi cấu trúc như hình ảnh, video hay âm thanh. Đặc biệt, khi

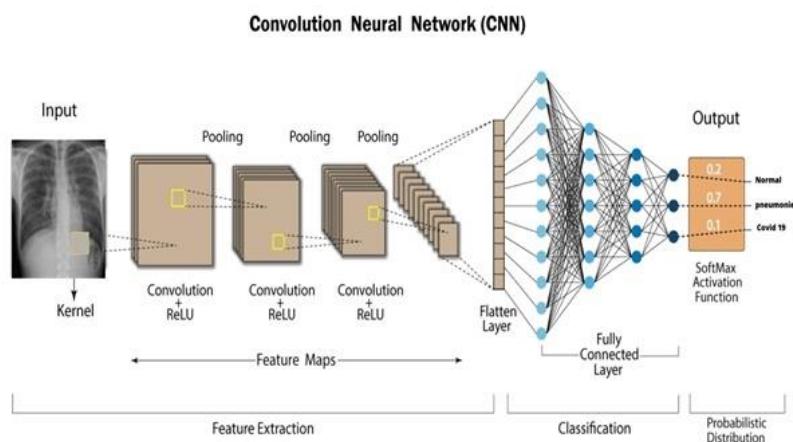
chạy trên GPU hoặc TPU, mạng có thể hoạt động gần như theo thời gian thực, phục vụ tốt cho các hệ thống giám sát giao thông thông minh.

Hạn chế:

Việc huấn luyện mạng nơ-ron đòi hỏi lượng dữ liệu rất lớn và phải được gán nhãn chính xác, gây tốn kém thời gian và chi phí. Mô hình cũng yêu cầu tài nguyên tính toán mạnh, cần GPU/TPU và bộ nhớ lớn, khiến quá trình triển khai phức tạp hơn. Ngoài ra, mạng nơ-ron thường khó giải thích do cấu trúc nhiều tầng và số lượng lớn tham số, gây hạn chế trong các ứng dụng cần tính minh bạch. Cuối cùng, mô hình dễ mắc lỗi overfitting nếu quá phức tạp so với dữ liệu, vì vậy phải áp dụng các kỹ thuật như điều chỉnh, dropout, tăng cường dữ liệu hoặc dừng sớm để tránh học thuộc dữ liệu.

2.2 Mạng nơ-ron tích chập (Convolutional Neural Network – CNN)

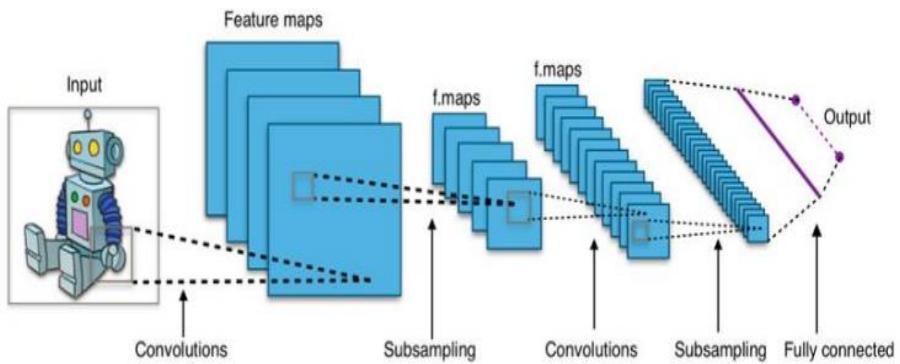
Mạng nơ-ron tích chập (CNN) là một kiến trúc mạng nơ-ron chuyên biệt được phát triển để xử lý dữ liệu có cấu trúc dạng lưới, đặc biệt là ảnh và video. Đặc trưng nổi bật của CNN là khả năng tự động trích xuất đặc trưng (feature extraction) mà không cần sự can thiệp thủ công, đồng thời giảm đáng kể số lượng tham số so với các mạng kết nối đầy đủ truyền thống.



Hình 2.3 Hình ảnh minh họa mạng nơron tích chập (CNN)

Nhờ tính hiệu quả trong xử lý ảnh và khả năng học được các đặc trưng mạnh mẽ, CNN trở thành mô hình nền tảng trong các hệ thống giám sát giao thông thông minh, nhận diện đối tượng và phân tích hình ảnh/video.

2.2.1 Cấu trúc cơ bản của CNN



Hình 2.4 Cấu trúc cơ bản của mạng nơron tích chập (CNN)

Một mạng CNN điển hình gồm bốn nhóm lớp chính: Lớp tích chập, Lớp gộp, Lớp kích hoạt phi tuyến, và Lớp kết nối đầy đủ. Mỗi nhóm đảm nhận một vai trò riêng trong việc trích xuất và xử lý đặc trưng của ảnh.

a) Lớp tích chập (Convolutional Layer)

Lớp tích chập là thành phần quan trọng nhất trong CNN chức năng chính:

- Trích xuất đặc trưng cục bộ từ ảnh thông qua phép tích chập (convolution) giữa ảnh và các bộ lọc (kernel/filter).
- Mỗi kernel là một ma trận nhỏ (thường 3×3 hoặc 5×5), trượt qua ảnh để tạo ra một feature map, thể hiện mức độ xuất hiện của một đặc trưng tại các vị trí trong ảnh.

Ý nghĩa học của kernel

Mỗi kernel đại diện cho một loại đặc trưng mà mạng học được:

- Kernel 1 thường được tối ưu để phát hiện các cạnh dọc trong ảnh. Khi kernel này quét qua các vùng có sự thay đổi cường độ theo phương ngang, nó tạo ra phản hồi mạnh, giúp mô hình nhận diện biên dọc của vật thể. Đây là loại đặc trưng cơ bản nhất, đóng vai trò xây dựng nền nhận thức hình dạng theo chiều đứng, đặc biệt quan trọng khi mô hình cần phân biệt các vật thể có cấu trúc đứng rõ ràng như thân xe, trụ biển báo hoặc biển số.

- Kernel 2 lại nhạy với sự thay đổi cường độ theo phương dọc, nên được tối ưu để phát hiện cạnh ngang. Các cạnh này biểu thị những thay đổi đột ngột

theo chiều đứng của ảnh, thường xuất hiện ở mép trên – dưới của vật thể. Trong ngữ cảnh giao thông, cạnh ngang giúp nhận biết các thành phần như nóc xe, mặt đường, thanh biển số, hoặc đường kẻ vạch. Sự bổ sung giữa Kernel 1 và Kernel 2 giúp CNN có khả năng phân tích cấu trúc hai chiều của hình ảnh một cách đầy đủ hơn.

- Kernel 3 có trọng số phức tạp và bất đối xứng hơn, cho phép nó nhạy với các đặc trưng như góc, đường cong hoặc vùng giao nhau của cạnh. Đây là các đặc trưng trung gian quan trọng để mô hình có thể nhận diện những hình dạng phức tạp hơn, chẳng hạn như góc biển số, cạnh cong của bánh xe, hoặc các giao điểm tạo thành khối cứng của phương tiện. Nhờ những đặc trưng này, mô hình bắt đầu hiểu không chỉ vị trí đường biên mà cả hình khối sơ cấp của vật thể trong ảnh.

Công thức tích chập

$$y(i, j) = \sum_{m=1}^k \sum_{n=1}^k x(i+m, j+n) W(m, n) + b$$

Trong đó:

- x : ảnh đầu vào
- W : kernel (trọng số)
- b : bias
- y : giá trị feature map

Việc sử dụng chung một kernel cho toàn bộ ảnh giúp CNN: Giảm số tham số, khai thác tính chất bất biến theo tịnh tiến (đối tượng xuất hiện bất kỳ vị trí nào vẫn được nhận diện).

b) Lớp gộp (Pooling Layer)

Pooling Layer giúp giảm kích thước feature map sau convolution, từ đó giảm số tính toán nhưng vẫn giữ lại đặc trưng quan trọng nhất. Lớp này lấy thông tin nổi bật trong từng vùng nhỏ và bỏ đi chi tiết không cần thiết, giúp mô hình ổn

định hơn trước các biến đổi nhỏ của ảnh. Nhờ vậy, mạng CNN tập trung vào đặc trưng chính và xử lý nhanh, hiệu quả hơn.

Hai loại Pooling phổ biến

Max Pooling:

Max Pooling là loại pooling phổ biến nhất trong các mạng CNN. Phép toán này chọn giá trị lớn nhất trong mỗi vùng con (patch) của feature map, thường có kích thước 2×2 hoặc 3×3 với bước nhảy (stride) tương ứng. Việc lấy giá trị cực đại giúp Max Pooling giữ lại những tín hiệu mạnh nhất, thể hiện sự hiện diện rõ rệt của các đặc trưng quan trọng như cạnh, góc hoặc các điểm sáng nổi bật.

Đặc trưng nổi bật mà Max Pooling mang lại là khả năng phát hiện và bảo toàn các mẫu mạnh, bất kể vị trí của chúng trong vùng xét. Điều này giúp mô hình trở nên bất biến với dịch chuyển nhỏ (translation invariance). Ví dụ, nếu cạnh xuất hiện hơi lệch trong vùng 2×2 , Max Pooling vẫn giữ được tín hiệu mạnh nhất và tránh làm mất thông tin. Do khả năng tập trung vào đặc trưng nổi bật, Max Pooling được sử dụng rộng rãi trong các kiến trúc CNN hiện đại như VGG, ResNet, Inception và cả các mô hình detection như YOLO.

Ngoài ra, Max Pooling hỗ trợ giảm kích thước feature map khá mạnh, từ đó giúp giảm lượng tham số và chi phí tính toán. Mặc dù điều này dẫn đến một phần mất mát thông tin, nhưng trong hầu hết các bài toán nhận dạng, việc ưu tiên “đặc trưng mạnh nhất” lại mang đến hiệu quả học tốt hơn và tăng khả năng tổng quát hóa.

Average Pooling:

Average Pooling thay vì chọn giá trị lớn nhất, sẽ tính trung bình tất cả các phần tử trong vùng xét. Điều này mang đến hiệu ứng làm mượt (smoothing) feature map, khiến các thay đổi nhỏ, nhiễu cục bộ hay những điểm bất thường bị giảm thiểu. Average Pooling giúp mô hình giữ lại bức tranh tổng quát của đặc trưng thay vì tập trung vào các điểm nổi bật nhất.

Trong các bài toán nhận dạng đối tượng hoặc phân loại hình ảnh, Average Pooling thường kém hiệu quả hơn Max Pooling, vì nó làm giảm độ sắc nét của đặc trưng và không nhấn mạnh các mẫu mạnh. Tuy nhiên, trong một số tình

huống, Average Pooling lại có vai trò quan trọng. Một ví dụ điển hình là Global Average Pooling (GAP) – phép tính trung bình toàn bộ feature map thành một giá trị duy nhất cho mỗi kênh. GAP được dùng rộng rãi trong các kiến trúc hiện đại như ResNet hoặc MobileNet để thay thế các lớp Fully Connected, giúp giảm đáng kể số lượng tham số, hạn chế overfitting và tăng tính diễn giải (interpretability).

Average Pooling cũng hữu ích trong các hệ thống cần ổn định với nhiễu, hoặc khi ta muốn mô hình chú ý đến mức độ xuất hiện trung bình của đặc trưng thay vì một điểm cực đại.

Lợi ích của Pooling Layer:

Pooling Layer giúp giảm kích thước feature map, từ đó giảm số tham số và lượng tính toán, giúp mô hình nhẹ và nhanh hơn. Việc giảm độ phức tạp cũng hạn chế overfitting, cho phép mạng học những đặc trưng ổn định hơn. Đồng thời, pooling tăng khả năng khai quát hóa nhờ giữ lại các đặc trưng quan trọng thay vì chi tiết dư thừa. Ngoài ra, pooling tạo ra tính bất biến với dịch chuyển nhỏ, giúp mô hình nhận dạng tốt ngay cả khi đối tượng bị lệch vị trí, xoay nhẹ hoặc xuất hiện nhiều.

c) Lớp kích hoạt phi tuyến (Activation Function)

Sau mỗi lớp tích chập, ta thường áp dụng một hàm kích hoạt ReLU để tạo phi tuyến cho mạng.

$$f(x) = \max(0, x)$$

Nếu $x > 0 \rightarrow$ hàm giữ nguyên giá trị x .

Nếu $x \leq 0 \rightarrow$ hàm trả về 0.

Hàm ReLU (Rectified Linear Unit) là một trong những hàm kích hoạt phi tuyến được sử dụng phổ biến nhất trong các mạng nơ-ron sâu, đặc biệt là các kiến trúc CNN hiện đại. Về bản chất, ReLU hoạt động theo cơ chế rất đơn giản: mọi giá trị âm của đầu vào đều được đưa về 0, trong khi các giá trị dương được giữ nguyên. Nhờ đặc tính này, ReLU vừa tạo ra tính phi tuyến cần thiết để mô hình học được các quan hệ phức tạp trong dữ liệu, vừa đảm bảo chi phí tính toán thấp hơn nhiều so với các hàm truyền thống như sigmoid hay tanh, vốn yêu cầu tính

toán hàm mũ. Chính vì không phải thực hiện các phép toán nặng, ReLU đặc biệt phù hợp với những mô hình có quy mô lớn hoặc yêu cầu xử lý thời gian thực.

Một ưu điểm nổi bật khác của ReLU là khả năng giảm thiểu hiện tượng mất gradient (vanishing gradient), vốn là vấn đề nghiêm trọng trong huấn luyện mạng sâu. Khi giá trị đầu vào lớn hơn 0, đạo hàm của ReLU luôn bằng 1, giúp gradient được truyền ngược một cách hiệu quả và ổn định qua nhiều lớp. Ngược lại, với giá trị âm, đạo hàm bằng 0 khiến các nơ-ron tương ứng bị “tắt”, nhờ đó mô hình có khả năng loại bỏ những đường kích hoạt không cần thiết và tăng tính chọn lọc trong học đặc trưng. Đặc điểm này giúp mạng trở nên ổn định hơn, tránh tràn thông tin và hỗ trợ việc hội tụ nhanh hơn. Với sự đơn giản nhưng hiệu quả rõ rệt, ReLU đã trở thành lựa chọn mặc định trong phần lớn các mô hình CNN hiện đại như ResNet, YOLO, VGG hay EfficientNet, đóng góp quan trọng vào hiệu suất và khả năng tổng quát hóa của các mạng học sâu.

e) Lớp kết nối đầy đủ (Fully Connected Layer – FC)

Lớp kết nối đầy đủ (Fully Connected Layer) nằm ở cuối mạng nơron tích chập và đóng vai trò quyết định trong việc đưa ra kết quả cuối cùng của mô hình. Sau khi các tầng tích chập và tầng gộp trích xuất được những đặc trưng quan trọng từ dữ liệu đầu vào, toàn bộ thông tin này sẽ được “phẳng hóa” thành một vector một chiều trước khi đưa vào lớp kết nối đầy đủ. Trong lớp này, mỗi nơron được kết nối với toàn bộ nơron ở tầng phía trước, cho phép mạng học được các mối quan hệ phức tạp giữa các đặc trưng đã thu được. Hoạt động của một nơron trong lớp kết nối đầy đủ được mô tả bởi công thức:

$$y = f(Wx + b)$$

trong đó W là ma trận trọng số, b là giá trị điều chỉnh (bias), x là đầu vào đã được phẳng hóa và f là hàm kích hoạt như ReLU hoặc sigmoid. Trong các bài toán phân loại, lớp kết nối đầy đủ cuối cùng thường kết hợp với hàm Softmax để biến đầu ra thành xác suất cho từng lớp, được tính theo công thức:

$$P(y_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Nhờ cơ chế này, mô hình có thể xác định lớp có xác suất cao nhất, từ đó đưa ra kết luận cuối cùng

2.2.2 Nguyên lý hoạt động của CNN

Bước 1: Tiền xử lý ảnh đầu vào

Ở bước đầu tiên, ảnh thu được từ camera thường có kích thước lớn và chứa nhiều loại nhiễu như ánh sáng không ổn định, độ mờ hoặc lệch màu. Vì vậy, ảnh được thay đổi kích thước về chuẩn như 224×224 hoặc 416×416 và được chuẩn hóa giá trị điểm ảnh về khoảng $0-1$ hoặc $-1-1$ theo công thức:

$$x_{\text{chuẩn hoá}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Trong một số trường hợp, ảnh còn được lọc nhiễu hoặc cân bằng sáng để giúp dữ liệu đồng nhất và làm mô hình học hiệu quả hơn.

Bước 2: Tích chập để trích xuất đặc trưng mức thấp

Ở bước này, ảnh đã được tiền xử lý sẽ đi qua các lớp tích chập để trích xuất những đặc trưng cơ bản như cạnh, đường thẳng và họa tiết nhỏ. Quá trình tích chập được mô tả bằng công thức:

$$y(i, j) = \sum_m \sum_n x(i + m, j + n) \cdot k(m, n)$$

trong đó x là ảnh đầu vào, k là bộ lọc tích chập và y là bản đồ đặc trưng thu được. Kết quả là nhiều bản đồ đặc trưng khác nhau đại diện cho các thông tin thị giác mức thấp.

Bước 3: Gộp đặc trưng bằng các lớp gộp

Sau tích chập, các bản đồ đặc trưng đi qua lớp gộp, thường là gộp cực đại. Lớp này có nhiệm vụ giảm chiều rộng và chiều cao của bản đồ đặc trưng, giữ lại giá trị quan trọng nhất trong mỗi vùng nhỏ. Gộp cực đại được mô tả bằng công thức:

$$y(i, j) = \max_{(m, n) \in \Omega} x(i + m, j + n)$$

trong đó Ω là vùng gộp (ví dụ 2×2). Bước này giúp giảm số lượng tham số, hạn chế học quá mức và làm mô hình ổn định hơn trước các dịch chuyển nhỏ của đối tượng trong ảnh.

Bước 4: Trích xuất đặc trưng mức cao ở các lớp tích chập sâu hơn

Ở các tầng sâu hơn, các bộ lọc tích chập không còn chỉ nhận diện đặc trưng đơn giản mà bắt đầu học những đặc trưng phức tạp như bánh xe, thân xe, biển báo hoặc con người. Quá trình tích chập vẫn tuân theo công thức ở bước 2 nhưng với nhiều bộ lọc hơn và kích thước bản đồ đặc trưng thường nhỏ hơn. Đây là bước hình thành các đặc trưng mức cao, rất quan trọng cho việc phân loại và nhận dạng đối tượng.

Bước 5: Đưa vào các lớp kết nối đầy đủ

Cuối cùng, các bản đồ đặc trưng được làm phẳng thành một vector một chiều và đưa vào mạng kết nối đầy đủ. Mỗi nơron trong lớp này tính toán theo công thức:

$$y = f(Wx + b)$$

trong đó W là ma trận trọng số, b là giá trị điều chỉnh và f là hàm kích hoạt như ReLU hoặc sigmoid.

Ở tầng cuối cùng, mô hình thường sử dụng hàm Softmax cho bài toán phân loại nhiều lớp, được tính như sau:

$$P(y_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

Nhờ đó, mô hình đưa ra xác suất cho từng lớp, ví dụ như loại phương tiện hoặc tình huống vi phạm.

2.2.3 Ưu điểm và hạn chế của mạng CNN

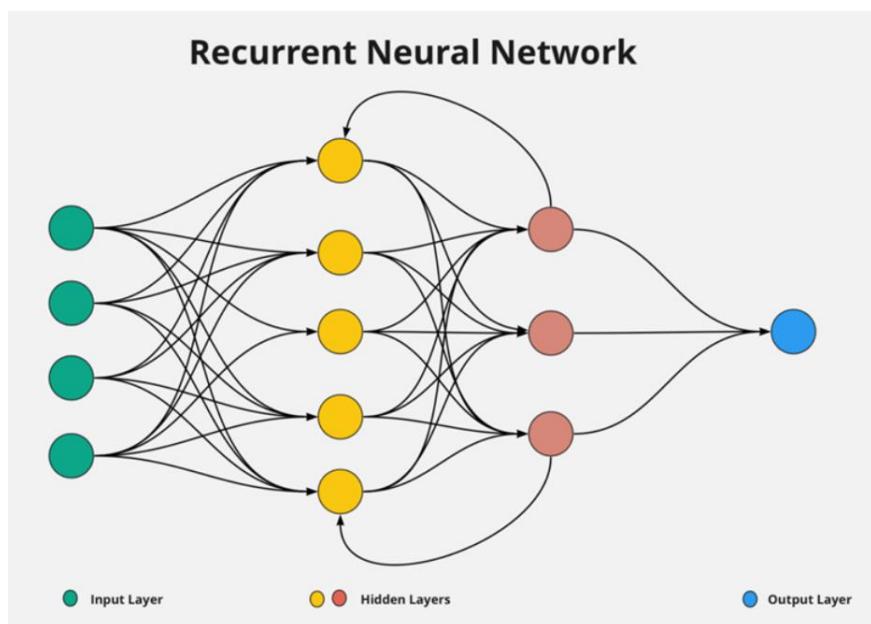
Mạng nơ-ron tích chập (CNN) có khả năng tự động trích xuất đặc trưng từ hình ảnh và video nhờ các lớp tích chập và lớp gộp, giúp mô hình không cần thiết kế đặc trưng thủ công và học được những quan hệ không gian phức tạp. Điều này làm cho CNN đạt hiệu suất cao trong các nhiệm vụ như nhận diện phương tiện, phát hiện biển báo, phân đoạn làn đường hay phân tích cảnh giao thông. Ngoài ra,

việc sử dụng các bộ lọc lặp lại giúp giảm số lượng tham số, hạn chế học quá mức và hỗ trợ xử lý thời gian thực khi kết hợp với phần cứng chuyên dụng.

Tuy vậy, CNN cũng có những hạn chế như cần lượng lớn dữ liệu gán nhãn chính xác và đòi hỏi tài nguyên tính toán mạnh trong quá trình huấn luyện. Mô hình có tính khó giải thích, hoạt động như “hộp đen”, gây khó khăn trong các ứng dụng cần minh bạch. CNN cũng dễ học quá mức khi dữ liệu không đủ đa dạng và dễ suy giảm hiệu suất khi điều kiện môi trường thay đổi. Vì thế, các kỹ thuật như tăng cường dữ liệu, điều chuẩn, loại bỏ ngẫu nhiên nơ-ron và dừng sớm thường được áp dụng để cải thiện độ ổn định và khả năng thích nghi của mô hình.

2.3 Mạng nơ-ron hồi quy (Recurrent Neural Network – RNN) và LSTM

Mạng nơ-ron hồi quy (RNN) là một mô hình học sâu được đào tạo để xử lý và chuyển đổi đầu vào dữ liệu tuần tự thành đầu ra dữ liệu tuần tự cụ thể. Dữ liệu tuần tự là dữ liệu, chẳng hạn như từ, câu hoặc dữ liệu chuỗi thời gian, trong đó các thành phần tuần tự tương quan với nhau dựa trên ngữ nghĩa phức tạp và quy tắc cú pháp. RNN là một hệ thống phản mềm gồm nhiều thành phần được kết nối với nhau theo cách con người thực hiện chuyển đổi dữ liệu tuần tự, chẳng hạn như dịch văn bản từ ngôn ngữ này sang ngôn ngữ khác. Phần lớn RNN đang được thay thế bằng trí tuệ nhân tạo (AI) dựa trên công cụ biến đổi và các mô hình ngôn ngữ lớn (LLM), hiệu quả hơn nhiều trong việc xử lý dữ liệu tuần tự.



Hình 2.5 Hình ảnh minh họa mạng nơ-ron hồi quy (RNN)

2.3.1 Cấu trúc của mạng nơ-ron hồi quy

Mạng thần kinh hồi quy có cấu trúc khác biệt so với các loại mạng thần kinh truyền thống. Điểm đặc trưng của nó nằm ở vòng lặp hồi quy, cho phép thông tin từ bước thời gian trước được đưa trở lại để phục vụ việc xử lý bước thời gian sau.

Các thành phần chính của mạng nơ-ron hồi quy:

a) Dữ liệu đầu vào tại thời điểm hiện tại (x_t)

Tại mỗi bước thời gian t , mạng thần kinh hồi quy nhận một giá trị đầu vào x_t . Đây là thông tin mà mô hình cần xử lý ngay tại thời điểm đó. Tùy theo bài toán, x_t có thể mang nhiều dạng khác nhau, chẳng hạn như:

- Một điểm dữ liệu trong chuỗi thời gian, ví dụ: giá cổ phiếu ngày hôm nay, nhiệt độ hiện tại, lưu lượng giao thông theo giờ.
- Một bộ đặc trưng được rút trích từ hình ảnh, chẳng hạn vector đặc trưng do CNN tạo ra tại mỗi khung hình trong video.
- Một từ trong câu, thường được biểu diễn dưới dạng one-hot vector hoặc embedding.
- Một tín hiệu thu được từ cảm biến, chẳng hạn dữ liệu từ cảm biến gia tốc, cảm biến áp suất hoặc cảm biến chuyển động.
- Vị trí hoặc vận tốc của một phương tiện giao thông, phục vụ dự báo hành trình hoặc mô hình hóa chuyển động.

b, Trạng thái ẩn tại thời điểm hiện tại (h_t)

Trạng thái ẩn h_t là thành phần cốt lõi của mạng thần kinh hồi quy, đóng vai trò như bộ nhớ tạm thời giúp mô hình duy trì và xử lý thông tin theo thời gian. Tại mỗi bước t , trạng thái ẩn thực hiện hai nhiệm vụ quan trọng:

- Lưu trữ thông tin tóm tắt về toàn bộ dữ liệu đã đi qua h_t chứa thông tin được “tích lũy” từ các thời điểm trước đó. Vì vậy, nó giúp mạng hiểu được ngữ cảnh và mối quan hệ giữa các phần tử trong chuỗi.

- Tham gia trực tiếp vào việc tính toán đầu ra tại thời điểm t

Đầu ra y_t của mạng không chỉ phụ thuộc vào đầu vào hiện tại x_t mà còn phụ thuộc vào trạng thái ẩn h_t . Điều này cho phép mạng phản ánh đúng bối cảnh trong chuỗi dữ liệu.

- Kích thước của trạng thái ẩn là cố định, không thay đổi theo độ dài của chuỗi.

Nghĩa là dù chuỗi gồm 10 phần tử hay 1000 phần tử, kích thước vector h_t vẫn giữ nguyên, giúp mô hình gọn nhẹ hơn và tránh bùng nổ số lượng tham số.

c. Đầu ra tại thời điểm hiện tại (y_t)

Đầu ra y_t là kết quả mà mạng thần kinh hồi quy tạo ra tại bước thời gian t . Giá trị này được tính trực tiếp từ trạng thái ẩn h_t , vì trạng thái ẩn đã tổng hợp thông tin của cả đầu vào hiện tại và ngữ cảnh từ các thời điểm trước. Tùy theo mục tiêu của bài toán, y_t có thể mang nhiều dạng khác nhau, chẳng hạn:

- Giá trị dự đoán

Ví dụ: dự báo nhiệt độ ngày mai, giá trị cảm biến kế tiếp, mức tiêu thụ năng lượng,...

- Xác suất phân loại

Thường dùng trong các mô hình ngôn ngữ, nhận dạng chuỗi hành động, phân loại cảm xúc,...

- Nhãn cho từng thời điểm (sequence labeling)

Chẳng hạn: gán nhãn từ loại (POS tagging), nhận dạng thực thể (NER), hoặc phân đoạn video theo từng khung hình.

- Tín hiệu điều khiển

Dùng trong robot, điều khiển tự động, dự đoán chuyển động,...

d. Hệ thống trọng số của mạng

Trong mạng thần kinh hồi quy, các phép biến đổi từ đầu vào và trạng thái ẩn sang trạng thái và đầu ra được thực hiện thông qua ba ma trận trọng số chính:

- Ma trận trọng số từ đầu vào sang trạng thái ẩn (W_{xh})

Xác định cách thông tin hiện tại x_t ảnh hưởng đến trạng thái ẩn mới h_t .

- Ma trận trọng số từ trạng thái ẩn trước sang trạng thái ẩn hiện tại (W_{hh})

Mô tả cách bộ nhớ từ bước trước h_{t-1} được kết hợp với đầu vào hiện tại để tạo ra bộ nhớ mới. Đây chính là “vòng lặp hồi quy” đặc trưng của RNN.

- Ma trận trọng số từ trạng thái ẩn sang đầu ra (W_{hy})

Quy định cách mạng chuyển đổi thông tin lưu trong trạng thái ẩn h_t thành đầu ra y_t .

2.3.2 Nguyên lý hoạt động mạng nơ-ron hồi quy (RNN)

Mạng nơ-ron hồi quy (RNN) là một dạng mạng nơ-ron đặc biệt được thiết kế để xử lý dữ liệu tuần tự, nơi thứ tự và ngữ cảnh của các phần tử trong chuỗi giữ vai trò quan trọng. Khác với mạng truyền thẳng (Feedforward Neural Network) vốn xử lý từng đầu vào độc lập, RNN có khả năng tái sử dụng thông tin quá khứ nhờ một cấu trúc hồi quy bên trong.

Cơ chế hồi quy — bộ nhớ ngắn hạn

RNN hoạt động dựa trên việc duy trì một trạng thái ẩn (hidden state), đóng vai trò là “bộ nhớ tạm thời”, giúp mạng ghi nhớ thông tin từ các bước thời gian trước đó.

Tại bước thời gian t , mạng nhận:

- Đầu vào hiện tại: x_t
- Trạng thái ẩn trước đó: h_{t-1}

và cập nhật trạng thái ẩn mới:

$$h_t = f(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$$

Trong đó:

- W_{xh} : trọng số kết nối đầu vào → trạng thái ẩn
- W_{hh} : trọng số hồi quy ($\text{ẩn} \rightarrow \text{ẩn}$)
- f : hàm kích hoạt (tanh, ReLU)

Trạng thái ẩn h_t tóm tắt toàn bộ thông tin từ bước 1 đến t .

Chia sẻ tham số qua các bước thời gian

Tất cả các bước thời gian sử dụng chung 1 tập trọng số:

$$W_{xh}, W_{hh}, W_{hy}$$

Điều này giúp RNN học được quy luật thời gian không phụ thuộc vị trí, làm cho mô hình gọn nhẹ và hiệu quả hơn.

Đầu ra tại mỗi bước thời gian

Đầu ra y_t được tính từ trạng thái ẩn:

$$y_t = g(W_{hy}h_t + b_y)$$

Trong quá trình xử lý chuỗi, cách RNN sinh đầu ra phụ thuộc trực tiếp vào yêu cầu của bài toán. Đối với các nhiệm vụ cần dự đoán tại từng bước thời gian như gán nhãn từ (POS tagging), phân tích thực thể (NER) hay dịch máy theo từng từ, mô hình sẽ xuất ra đầu ra tại mỗi bước thời gian, được gọi là kiến trúc Many-to-Many. Ở dạng này, mỗi trạng thái ẩn h_t sẽ được ánh xạ thành một đầu ra y_t , phản ánh thông tin của toàn bộ chuỗi từ bước 1 đến thời điểm t.

Ngược lại, với những bài toán chỉ cần một kết quả duy nhất cho toàn bộ chuỗi đầu vào chẳng hạn như phân tích cảm xúc của một câu, nhận diện thẻ loại văn bản hay dự báo giá trị của chuỗi thời gian dựa trên toàn bộ lịch sử RNN sẽ chỉ tạo đầu ra ở bước thời gian cuối cùng, được gọi là kiến trúc Many-to-One. Trong trường hợp này, trạng thái ẩn cuối cùng h_T đóng vai trò như một biểu diễn tổng quát của toàn bộ chuỗi, và được sử dụng để sinh ra đầu ra cuối y_T .

2.3.3 Ưu điểm và Nhược điểm của mạng nơ-ron hồi quy (RNN)

Ưu điểm của RNN:

Một trong những ưu điểm quan trọng nhất của RNN là khả năng mô hình hóa dữ liệu tuần tự, nơi thông tin tại một thời điểm phụ thuộc vào các phần tử xuất hiện trước đó. Nhờ cơ chế duy trì trạng thái ẩn, RNN có thể lưu trữ và tích lũy thông tin ngắn hạn trong chuỗi, giúp mạng hiểu được ngữ cảnh cục bộ như quan hệ giữa các từ trong câu hoặc xu hướng gần trong chuỗi thời gian.

Bên cạnh đó, RNN sử dụng cơ chế chia sẻ trọng số qua mọi bước thời gian, giúp giảm đáng kể số lượng tham số cần học. Điều này không chỉ làm cho mô hình trở nên gọn nhẹ mà còn cho phép mạng áp dụng cùng một quy tắc học cho

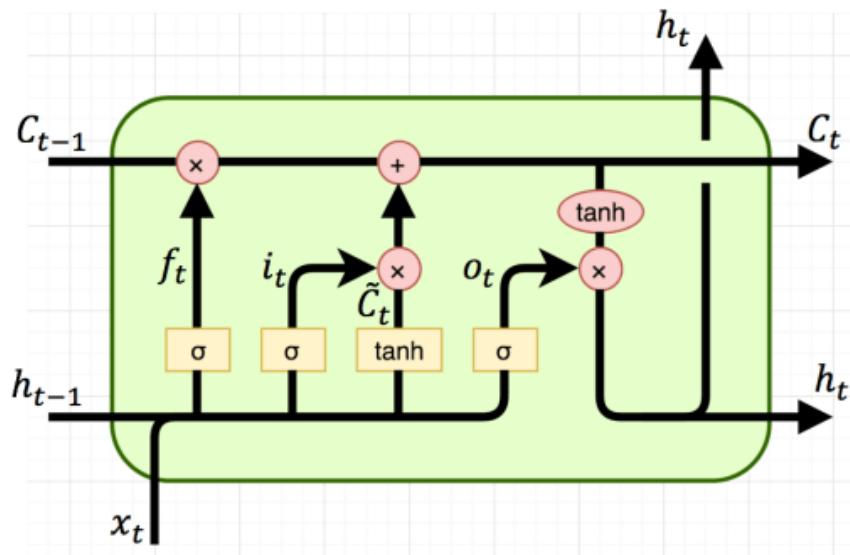
mọi vị trí trong chuỗi, bất kể độ dài của chuỗi thay đổi như thế nào. Ngoài ra, RNN có thể xử lý các chuỗi có độ dài tùy ý mà không cần thay đổi cấu trúc mạng, tạo sự linh hoạt trong nhiều ứng dụng như xử lý ngôn ngữ tự nhiên, dự đoán chuỗi thời gian và nhận dạng giọng nói.

Nhược điểm của RNN

Mặc dù có nhiều ưu điểm, RNN cơ bản lại gặp phải những hạn chế lớn trong thực tế. Vấn đề nghiêm trọng nhất là hiện tượng vanishing gradient khi lan truyền ngược qua nhiều bước thời gian, khiến gradient dần trở nên quá nhỏ và gần như không thể cập nhật trọng số một cách hiệu quả. Điều này khiến RNN gần như không thể học được các mối quan hệ dài hạn trong chuỗi, làm cho mô hình dễ “quên” thông tin quan trọng nếu nó xuất hiện quá xa trong quá khứ. Ngược lại, đôi khi RNN cũng có thể gặp exploding gradient, khiến giá trị gradient tăng đột biến và làm quá trình huấn luyện mất ổn định nếu không áp dụng các biện pháp như gradient clipping.

Ngoài ra, do đặc tính xử lý tuần tự, RNN không thể song song hóa tốt, dẫn đến tốc độ huấn luyện chậm hơn đáng kể so với các mô hình hiện đại dựa trên cơ chế tự chú ý (Attention). RNN cũng kém hiệu quả khi làm việc với các chuỗi rất dài, tiêu tốn nhiều thời gian tính toán và khó nắm bắt được các phụ thuộc xa. Cuối cùng, chất lượng của RNN thường kém hơn LSTM và GRU, vì thiếu cơ chế kiểm soát việc ghi nhớ hoặc quên thông tin trong chuỗi.

2.3.4 Mạng LSTM (Long Short-Term Memory)



Hình 2.6 Hình ảnh minh họa mạng Long Short-Term Memory

Mạng LSTM (Long Short-Term Memory) là một biến thể nâng cao của mạng thần kinh hồi quy RNN, được thiết kế nhằm khắc phục các hạn chế có hưu của RNN truyền thống, đặc biệt là vấn đề vanishing gradient. Khác với RNN cơ bản chỉ dựa vào một trạng thái ẩn duy nhất để lưu trữ thông tin, LSTM được trang bị một cơ chế bộ nhớ dài hạn nhờ vào cấu trúc các “cổng” (gates) được điều khiển bởi các hàm kích hoạt sigmoid và tanh. Những cổng này cho phép mạng chủ động quyết định thông tin nào sẽ được lưu lại, thông tin nào bị quên đi, và thông tin nào được sử dụng để sinh trạng thái ẩn mới.

Cụ thể, mỗi tế bào LSTM (LSTM cell) bao gồm ba cổng chính: cổng quên Forget Gate, cổng đầu vào Input Gate và cổng đầu ra Output Gate. Cổng quên chịu trách nhiệm loại bỏ những thông tin không cần thiết từ trạng thái ô nhớ của bước trước; cổng đầu vào kiểm soát lượng thông tin mới được lưu vào bộ nhớ; trong khi cổng đầu ra điều chỉnh phần thông tin trong ô nhớ được dùng để tạo ra trạng thái ẩn hiện tại. Nhờ đó, LSTM duy trì hai trạng thái: trạng thái ô nhớ (C_t) – đóng vai trò là bộ nhớ dài hạn, và trạng thái ẩn (h_t) – đại diện cho thông tin ngắn hạn.

Quá trình cập nhật một cell LSTM tuân theo các bước sau:

- Cổng quên: Cổng quên quyết định phần thông tin nào từ trạng thái ô nhớ trước đó C_{t-1} sẽ được giữ lại hay loại bỏ:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$

- Cổng đầu vào: Cổng đầu vào xác định phần thông tin mới sẽ được thêm vào ô nhớ:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i), \tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C)$$

- Cập nhật trạng thái ô nhớ: Trạng thái ô nhớ mới C_t được tính bằng cách kết hợp thông tin cần quên và thông tin mới học được:

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

- Cổng đầu ra và trạng thái ẩn: Cổng đầu ra xác định phần nào của ô nhớ sẽ được đưa ra ngoài làm trạng thái ẩn:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o), h_t = o_t \odot \tanh(C_t)$$

Nhờ cấu trúc điều khiển thông tin tinh vi này, LSTM có khả năng học và lưu trữ các mối quan hệ dài hạn trong chuỗi dữ liệu, vượt trội hơn hẳn so với RNN truyền thống. Vì vậy, LSTM được sử dụng rộng rãi trong các bài toán như dịch máy, mô hình hóa ngôn ngữ, nhận dạng giọng nói, phân tích cảm xúc và dự đoán chuỗi thời gian – nơi yêu cầu mô hình có khả năng ghi nhớ thông tin xuất hiện từ rất xa trong quá khứ.

Ứng dụng ANN – CNN – RNN – LSTM trong giao thông:

Các mô hình học sâu như ANN, CNN, RNN và LSTM được ứng dụng rộng rãi trong lĩnh vực giao thông để nâng cao hiệu quả quản lý và điều khiển. ANN thường được dùng để dự báo lưu lượng phương tiện và tối ưu hóa tín hiệu đèn giao thông. CNN xử lý hình ảnh và video, hỗ trợ nhận dạng biển báo, phát hiện phương tiện và giám sát vi phạm. Trong khi đó, RNN và đặc biệt là LSTM làm việc tốt với dữ liệu chuỗi thời gian, giúp dự báo tắc nghẽn, thời gian di chuyển và phân tích diễn biến giao thông theo thời gian thực. Nhờ các mô hình này, hệ thống giao thông trở nên thông minh, chính xác và tự động hơn.

Mô hình	Đặc điểm chính	Ứng dụng trong giao thông
ANN	Mạng nơ-ron truyền thống, kết nối đầy đủ	Phân loại cơ bản, nhận dạng tín hiệu đơn giản
CNN	Tích chập, trích xuất đặc trưng mạnh từ ảnh	Nhận diện phương tiện, biển báo, vạch đường
RNN	Xử lý chuỗi, ghi nhớ ngữ cảnh ngắn hạn	Phân tích hành vi theo chuỗi video
LSTM	Ghi nhớ dài hạn, khắc phục vanishing gradient	Dự đoán chuyển động, phân tích chuỗi dài

Bảng 2.1 *Ứng dụng ANN, CNN, RNN và LSTM trong bài toán giao thông*

Các mô hình ANN, CNN, RNN và LSTM đều có những ưu và nhược điểm riêng khi ứng dụng trong giao thông. ANN có ưu điểm là cấu trúc đơn giản, dễ triển khai và cho kết quả dự báo khá tốt, nhưng lại phụ thuộc nhiều vào chất lượng dữ liệu và dễ bị quá khớp. CNN vượt trội trong xử lý hình ảnh và video, giúp nhận diện phương tiện và biển báo rất chính xác, tuy nhiên yêu cầu tài nguyên tính toán lớn và cần dữ liệu gán nhãn nhiều. RNN phù hợp với dữ liệu chuỗi thời gian như

lưu lượng giao thông, nhưng dễ gặp vấn đề mất mát hoặc nổ gradient khiến mô hình khó học các quan hệ dài hạn. LSTM khắc phục được hạn chế này và dự báo chuỗi tốt hơn, song cấu trúc phức tạp khiến thời gian huấn luyện dài và đòi hỏi nhiều tài nguyên. Do đó, việc lựa chọn mô hình tùy thuộc mục tiêu và nguồn lực của hệ thống giao thông.

Mô hình	Ưu điểm	Nhược điểm
CNN	Mạnh trong xử lý ảnh; đặc trưng học tự động	Cần nhiều dữ liệu huấn luyện
RNN	Phân tích chuỗi tốt, xử lý thời gian	Dễ bị vanishing gradient
LSTM	Ghi nhớ dài hạn, ổn định hơn RNN	Tốn tài nguyên, chậm hơn
ANN	Đơn giản, nhanh	Không phù hợp xử lý ảnh phức tạp

Bảng 2.2 So sánh các mô hình ANN, CNN, RNN và LSTM

2.4 Kiến trúc YOLOv8 – Nền tảng chính của hệ thống

2.4.1 Tổng Quan Về Kiến trúc YOLOv8 :



Trên cơ sở các mạng nơ-ron tích chập (CNN), nhiều kiến trúc phát hiện đối tượng đã được đề xuất và phát triển. Trong số đó, họ mô hình YOLO (You Only

Look Once) đã trở thành một trong những phương pháp mạnh mẽ và phổ biến nhất nhờ khả năng phát hiện nhanh, độ chính xác cao, và tính phù hợp với các ứng dụng thời gian thực. Khác với các mô hình hai giai đoạn như R-CNN hay Faster R-CNN, YOLO sử dụng cơ chế phát hiện một lần (single-stage detection), thực hiện đồng thời hai nhiệm vụ: Phân loại đối tượng, Dự đoán vị trí khung chứa (bounding box) ngay trong một lần lan truyền (forward pass) qua mạng. Điều này giúp giảm độ trễ và tối ưu hiệu năng trong quá trình suy luận.

Một trong những phiên bản tiên tiến nhất của họ YOLO là YOLOv8, được huấn luyện theo phương pháp học có giám sát (Supervised Learning). Trong đó, mỗi ảnh đầu vào I được gán tập nhãn thật (Ground Truth) Y , bao gồm:

- Loại đối tượng (Class Label): biểu thị lớp mà đối tượng thuộc về.
- Tọa độ khung chứa (Bounding Box): được mô tả bởi bộ tham số (x, y, w, h) , trong đó (x, y) là tâm bounding box và (w, h) là chiều rộng và chiều cao.
- Mức độ tin cậy (Confidence Score): cho biết khả năng tồn tại của một đối tượng tại vị trí đó (thường gán là 1 đối với nhãn thật).

Tập nhãn có thể được biểu diễn dưới dạng:

$$Y = \{(c_i, b_i, s_i)\}_{i=1}^N$$

Trong đó c_i là nhãn lớp, $b_i = (x_i, y_i, w_i, h_i)$ là bounding box và s_i là điểm tin cậy của đối tượng thứ i .

Quy trình huấn luyện YOLOv8

Quá trình huấn luyện diễn ra như một vòng lặp tối ưu hóa nhằm tìm ra tập trọng số tối ưu W^* . Mỗi vòng lặp (epoch) bao gồm các bước:

1. Dự đoán (Forward Propagation)

Mô hình với trọng số hiện tại W sinh ra dự đoán:

$$\hat{Y} = f(I; W)$$

Trong đó \hat{Y} chứa các giá trị dự đoán về lớp, bounding box và điểm tin cậy.

2. So sánh với nhãn thật (Matching)

Các dự đoán \hat{Y} được ghép nối (match) với nhãn thật Y . Mục tiêu là xác định mức độ sai lệch giữa dự đoán và dữ liệu đúng.

3. Tính toán sai số (Loss Function)

YOLOv8 sử dụng hàm mất mát tổng hợp:

$$L_{\text{total}} = \lambda_{cls} L_{cls} + \lambda_{box} L_{box} + \lambda_{obj} L_{obj}$$

Trong đó:

- Loss phân loại (Classification Loss):

$$L_{cls} = - \sum_i \sum_k y_{ik} \log(\hat{p}_{ik})$$

- Loss hồi quy bounding box (CIoU Loss):

$$L_{box} = 1 - \text{CIoU}(b_i, \hat{b}_i)$$

- Loss điểm tin cậy (Objectness Loss):

$$L_{obj} = -[s_i \log(\hat{s}_i) + (1 - s_i) \log(1 - \hat{s}_i)]$$

4. Cập nhật trọng số bằng Backpropagation và AdamW

Sau khi tính toán sai số, mô hình thực hiện lan truyền ngược (Backpropagation) để tính gradient:

$$g_t = \nabla_W L_{\text{total}}(W_t)$$

Trong YOLOv8, thuật toán tối ưu AdamW được sử dụng để cập nhật trọng số, với các bước:

Cập nhật moment:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1 - \beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \end{aligned}$$

Hiệu chỉnh moment:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

Cập nhật trọng số:

$$W_{t+1} = W_t - \eta \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} + \lambda W_t \right)$$

Mục tiêu cuối cùng quá trình huấn luyện nhằm tìm trọng số tối ưu:

$$W^{*} = \arg \min_W \sum_{i=1}^M L_{total}^{(i)}(W)$$

Trong đó M là tổng số mẫu hoặc batch trong tập huấn luyện.

2.4.2 Cấu Trúc Tổng Thể Của YOLOv8 Trong Quá Trình Huấn Luyện

YOLOv8 được xây dựng dựa trên ba khối chính, với cải tiến so với các phiên bản trước:

1. Backbone:

Trong YOLOv8, Backbone đóng vai trò trích xuất các đặc trưng quan trọng từ ảnh đầu vào và được xây dựng trên nền tảng CSPDarknet đã được cải tiến. Thành phần nổi bật nhất của Backbone là khói C2f – phiên bản nâng cấp của C3 trong YOLOv5. Khối C2f được thiết kế với cấu trúc chia luồng dữ liệu thành nhiều nhánh song song, sau đó tổng hợp lại ở cuối, giúp mô hình học được các đặc trưng ngũ cảnh phong phú hơn nhưng vẫn duy trì tốc độ suy luận cao. Backbone tạo ra các Feature Map ở nhiều mức độ khác nhau, từ nông đến sâu, giúp mô hình nắm bắt đầy đủ cả chi tiết vị trí lẫn thông tin ngũ nghĩa.

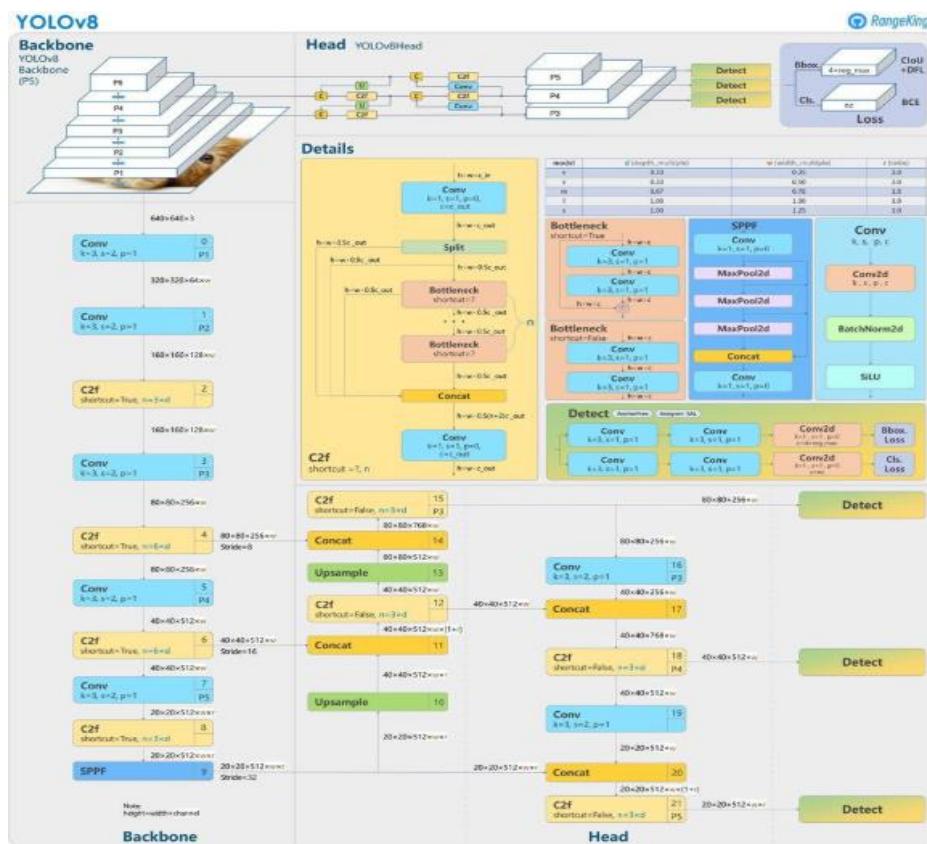
2. Neck:

Neck là phần kết nối giữa Backbone và Head, đồng thời đảm nhiệm vai trò tổng hợp các Feature Map đa tỷ lệ để mô hình có thể xử lý hiệu quả các đối tượng có kích thước khác nhau. YOLOv8 sử dụng kiến trúc kết hợp giữa FPN (Feature Pyramid Network) và PAN (Path Aggregation Network). FPN truyền thông tin ngũ nghĩa từ các lớp sâu xuống các lớp nông, trong khi PAN truyền thông tin vị trí chi tiết từ các lớp nông lên các lớp sâu. Sự kết hợp song song hai luồng truyền

này giúp mô hình vừa giữ được độ chính xác về vị trí, vừa đảm bảo khả năng phân loại mạnh mẽ, đặc biệt hiệu quả trong việc phát hiện các vật thể nhỏ.

3. Head:

Head của YOLOv8 là thành phần thực hiện dự đoán cuối cùng về hộp giới hạn và lớp đối tượng. YOLOv8 áp dụng kiến trúc Anchor-Free kết hợp với Decoupled Head, mang lại hiệu suất cao hơn so với các phiên bản trước. Nhờ cấu trúc tách rời, Head gồm hai nhánh độc lập: nhánh phân loại (Classification) dự đoán lớp của đối tượng và nhánh hồi quy (Regression) dự đoán tọa độ cùng kích thước hộp giới hạn. Cách tiếp cận này cải thiện tốc độ hội tụ và tăng độ chính xác của mô hình. Đồng thời, việc loại bỏ Anchor Box giúp mô hình dự đoán trực tiếp tâm và kích thước hộp giới hạn, giảm độ phức tạp tính toán và tránh các vấn đề không khớp giữa vật thể và các anchor đặt sẵn.



(Hình 2.9 Cấu Trúc Tổng Thể Của YOLOv8 Trong Quá Trình Huấn Luyện)

2.4.3 Cơ Chế Anchor-Free Detection

Anchor-Free Detection là cơ chế dự đoán đối tượng của YOLOv8 mà không sử dụng các Anchor Box định sẵn như các thế hệ YOLO trước. Thay vì dự

đoán độ lệch so với các anchor, mô hình trực tiếp dự đoán tọa độ tâm và kích thước của bounding box tại mỗi cell trên Feature Map. Điều này giúp giảm độ phức tạp, loại bỏ nhu cầu tinh chỉnh anchor, đồng thời tăng độ chính xác và khả năng tổng quát hóa. Anchor-Free giúp mô hình học dễ dàng hơn, đặc biệt hiệu quả trong việc phát hiện các vật thể nhỏ hoặc có hình dạng đa dạng, và mang lại tốc độ suy luận cao hơn.

Nguyên tắc hoạt động:

Mỗi điểm $p = (p_x, p_y)$ trên Feature Map là tâm dự đoán tiềm năng của một đối tượng.

Mô hình dự đoán:

- Độ lệch từ tâm điểm đến tâm thật của Bounding Box: $(\Delta x, \Delta y)$
- Kích thước: (t_w, t_h) dưới dạng logarit

Công thức dự đoán Bounding Box (x_c, y_c, w, h) :

$$(x_c, y_c, w, h) = (p_x + \Delta x, p_y + \Delta y, e^{t_w}, e^{t_h})$$

Việc sử dụng hàm mũ e^t đảm bảo $w, h > 0$.

2.4.4 Cơ Chế Task-Aligned Assigner (TAL)

Task-Aligned Assigner (TAL) là cơ chế gán nhãn trong YOLOv8, dùng để chọn ra những điểm dự đoán phù hợp nhất cho quá trình huấn luyện. Thay vì chỉ dựa vào IoU hoặc chỉ dựa vào điểm phân loại, TAL kết hợp cả khả năng phân loại (classification score) và khả năng định vị (IoU) thành một chỉ số chung gọi là Task-Aligned Score. Những prediction có điểm số cao nhất trong vùng hợp lệ sẽ được chọn làm positive samples. Nhờ đó, TAL giúp giảm gán nhầm, tăng chất lượng điểm học và cải thiện hiệu suất cả về phân loại lẫn định vị đối tượng.

Điểm ưu tiên s_{ij} cho cặp (Dự đoán i , Nhãn thật j):

$$s_{ij} = P_{cls}^\alpha \cdot (\text{IoU}_{ij})^\beta$$

- P_{cls} : Độ tin cậy phân loại của dự đoán i
- IoU_{ij} : Intersection over Union giữa dự đoán và nhãn thật

- $\alpha, \beta \geq 1$: siêu tham số tăng trọng số cho các dự đoán chất lượng cao
Những dự đoán có s_{ij} cao nhất được gán làm mẫu dương, giúp mô hình học nhanh và ít nhiễu.

2.4.5 Hàm Mất Mát (Loss Function) trong YOLOv8

Hàm Mất Mát (Loss Function) YOLOv8 sử dụng là hàm mất mát tổng hợp, bao gồm 3 thành phần chính:

$$L_{\text{total}} = L_{\text{box}} + L_{\text{cls}} + L_{\text{dfl}}$$

Mỗi thành phần được thiết kế để tối ưu hóa một phần khác nhau của bài toán detection:

- Box Loss – L_{box} sử dụng CIoU Loss:

$$L_{\text{box}} = 1 - \text{CIoU}$$

CIoU (Complete IoU) đánh giá chất lượng khung dự đoán dựa trên 3 yếu tố:

- IoU – mức độ chồng lấp : đo xem predicted box và ground-truth box trùng nhau bao nhiêu.
- Khoảng cách tâm hai box: hạn chế trường hợp IoU cao nhưng box lệch tâm.
- Tỉ lệ hình dạng (aspect ratio): tách các box dự đoán có tỉ lệ chiều rộng/chiều cao khác xa ground truth.

- Classification Loss – L_{cls}

Dùng Binary Cross-Entropy (BCE) cho mỗi lớp:

$$L_{\text{cls}} = - \sum_i [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)]$$

$y_i \in \{0,1\}$: nhãn thật cho lớp i

p_i : xác suất dự đoán

Mỗi lớp được đánh giá độc lập tốt cho các dataset có nhiều lớp.

- Distribution Focal Loss – L_{dfl}

- Đây là thành phần đặc biệt và rất quan trọng trong YOLOv8.
- YOLOv8 → không dự đoán trực tiếp, mà học phân phối rời rạc (discrete distribution) cho từng tọa độ.
 - Mỗi cạnh của box được biểu diễn bằng một phân phối xác suất (bin). Sau đó mô hình lấy expected value để suy ra tọa độ cuối cùng.

2.4.6 Chiến Lược Tối Ưu (Optimization Strategy)

YOLOv8 sử dụng một số kỹ thuật tối ưu hóa tiên tiến:

- AdamW Optimizer: AdamW giúp mô hình học nhanh như Adam nhưng vẫn có khả năng regularize tốt như SGD. Điều này đặc biệt quan trọng với các mạng lớn như YOLOv8, giúp tránh overfitting khi huấn luyện trên dữ liệu nhỏ hoặc nhiễu.
- Learning Rate Schedule: Warmup tránh gradient đột ngột ở đầu training, còn Cosine Annealing giúp LR giảm tự nhiên, tránh “giảm sốc”. Kết quả là mô hình hội tụ mượt, ổn định và đạt điểm mAP cao hơn.
- Mixed Precision (FP16): FP16 đặc biệt hiệu quả với GPU RTX/Ampere. Nhờ loss scaling, chất lượng không giảm mà tốc độ tăng đáng kể. Điều này giúp YOLOv8 huấn luyện nhanh hơn mà không cần phàn cứng quá mạnh.
- EMA (Exponential Moving Average): Trọng số mô hình thường dao động mạnh trong lúc huấn luyện. EMA làm mượt những dao động này, giúp mô hình cuối cùng dự đoán ổn định hơn, đặc biệt hữu ích khi validation noise cao hoặc batch nhỏ.
- Data Augmentation: Augmentation mạnh giúp mô hình học tốt hơn trong môi trường thực tế. Mosaic và MixUp giúp mô hình nhận biết vật thể trong nhiều bối cảnh và tỷ lệ khác nhau, từ đó tăng khả năng tổng quát hóa rõ rệt.

2.4.7 Ưu điểm và hạn chế YOLOv8

Ưu điểm:

YOLOv8 sở hữu nhiều ưu điểm vượt trội so với các phiên bản trước. Mô hình sử dụng kiến trúc backbone cải tiến và cơ chế anchor-free, giúp tăng độ chính xác trong việc phát hiện đối tượng, đặc biệt đối với các vật thể nhỏ hoặc có hình dạng

phức tạp. Tốc độ xử lý của YOLOv8 rất cao, đáp ứng tốt yêu cầu thời gian thực trong các hệ thống giám sát và nhận dạng thông minh. Bên cạnh đó, YOLOv8 hỗ trợ đa dạng nhiệm vụ như phát hiện đối tượng, phân vùng ảnh, phân loại và ước lượng pose, mang lại tính linh hoạt cao trong nhiều ứng dụng. Thư viện Ultralytics tích hợp cũng giúp quá trình cài đặt và huấn luyện trở nên đơn giản hơn, dễ triển khai trên nhiều nền tảng phần cứng khác nhau.

Nhược điểm:

Tuy mang lại nhiều lợi ích, YOLOv8 cũng tồn tại một số hạn chế. Việc huấn luyện mô hình yêu cầu phần cứng mạnh, đặc biệt là GPU, khiến quá trình huấn luyện trên CPU trở nên rất chậm và kém hiệu quả. Các phiên bản lớn như YOLOv8l hoặc YOLOv8x có kích thước mô hình lớn, tiêu tốn nhiều bộ nhớ và không phù hợp với các thiết bị nhúng có tài nguyên hạn chế. Ngoài ra, mô hình phụ thuộc mạnh vào chất lượng dữ liệu đầu vào; nếu dữ liệu gán nhãn không đầy đủ hoặc thiếu đa dạng, hiệu suất sẽ giảm đáng kể. YOLOv8 cũng gặp khó khăn trong các trường hợp có mức độ che khuất cao, nơi nhiều đối tượng chồng lên nhau, dẫn đến khả năng nhận diện giảm. Vì vậy, khi triển khai thực tế, cần cân nhắc kỹ giữa mục tiêu ứng dụng, tài nguyên phần cứng và chất lượng dữ liệu để đạt hiệu quả tối ưu.

So sánh các mô hình YOLO: YOLOv5 – YOLOv7 – YOLOv8

Tiêu chí	YOLOv5	YOLOv7	YOLOv8
Năm ra mắt	2020	2022	2023
Kiến trúc Backbone	Focus + CSP	E-ELAN	C2f (CSP cải tiến)
Kiểu dự đoán (Head)	Decoupled Head	Decoupled Head (tối ưu hơn)	Decoupled + Task-Aligned Assigner
Cơ chế Anchor	Anchor-based	Anchor-based	Anchor-free
Khả năng mở rộng mô hình	Có (n, s, m, l, x)	Có (tiny → E6)	Có đầy đủ + phiên bản

			segmentation
Tốc độ xử lý	Nhanh	Rất nhanh	Nhanh nhất trong ba mô hình
Độ chính xác (mAP)	Cao	Rất cao	Cao nhất (tối ưu nhất)
Hỗ trợ phân đoạn ảnh	Có	Có	Có (cải thiện mạnh)
Ưu điểm nổi bật	Dễ huấn luyện, tài liệu phong phú	Hiệu năng mạnh, phù hợp hệ thống lớn	Anchor-free, chính xác cao, đa nhiệm
Nhược điểm	Anchor-based khó tối ưu	Model lớn, tốn tài nguyên	Yêu cầu GPU khi huấn luyện
Tối ưu triển khai thực tế	Rất tốt (ONNX, TensorRT)	Tốt	Rất tốt (hỗ trợ nhiều nền tảng nhất)

Bảng 2.3 So sánh các mô hình YOLOv5 – YOLOv7 – YOLOv8

CHƯƠNG 3. XÂY DỰNG TRIỀN KHAI HỆ THỐNG

3.1. Tổng quan hệ thống

Hệ thống được xây dựng nhằm tự động phát hiện và ghi nhận hành vi vượt đèn đỏ trong thời gian thực. Mục tiêu chính của hệ thống là hỗ trợ giám sát giao thông một cách tự động, giảm tải cho lực lượng chức năng và tăng tính minh bạch trong quá trình xử phạt vi phạm. Ứng dụng sử dụng kết hợp các thư viện :

Thư viện	Phiên bản	Chức năng
Python	3.11.8	Ngôn ngữ lập trình chính
OpenCV	4.8	Xử lý ảnh/video
Ultralytics YOLO	v8	Phát hiện đối tượng
EasyOCR	1.7	Nhận dạng ký tự
PyQt6	6.2.5	Giao diện phần mềm

Bảng 3.1 : các thư viện sử dụng trong mô hình

Luồng hoạt động chính gồm 7 bước:

1. Thu nhận khung hình từ camera hoặc video.
2. Tiền xử lý khung hình (resize và cắt ROI đèn).
3. Nhận diện trạng thái đèn giao thông bằng HSV masking.
4. Phát hiện phương tiện bằng YOLOv8m.
5. Kiểm tra xem xe có vượt qua vạch dừng khi đèn đỏ.
6. Lưu thông tin vi phạm (ảnh, tọa độ, thời gian, loại xe).
7. Hiển thị kết quả trực tiếp qua OpenCV và PyQt6.

Cấu hình sử dụng :

Thành phần	Giá trị
CPU	Intel i7-12700F
GPU	RTX 3060 (12GB)

RAM	16GB
Ổ cứng	SSD Nvme 512GB
Môi trường	Windows 11

Bảng 3.2 cấu hình máy tính triển khai mô hình

3.2. Tiền xử lý dữ liệu

3.2.1. Thu thập và chú thích dữ liệu

Dữ liệu bao gồm ảnh và video ghi lại giao thông thực tế. Các phương tiện được gán nhãn theo chuẩn YOLO với 4 lớp:

- Car
- Motorcycle
- Bus
- Truck

```
VEHICLE_CLASSES = [2, 3, 5, 7] # car, motorcycle, bus, truck
```

Việc giới hạn số lớp giúp mô hình tập trung vào các đối tượng quan trọng trong bài toán giao thông, đồng thời giảm thời gian huấn luyện.

3.2.2. Chuẩn hóa dữ liệu đầu vào

Chuẩn hóa các dữ liệu video, ảnh đầu vào

Toàn bộ khung hình được resize về 1280×720 trước khi đưa vào mô-đun xử lý. Việc chuẩn hóa kích thước này mang lại nhiều lợi ích quan trọng:

```
TARGET_W, TARGET_H = 1280, 720
```

1. Đảm bảo ROI của đèn luôn có định

Khi tất cả các khung hình đều có cùng độ phân giải, vùng quan tâm (ROI) của đèn giao thông sẽ không bị thay đổi bởi sự khác biệt kích thước giữa các video đầu vào. Điều này giúp hệ thống định vị đèn chính xác và ổn định hơn, tránh trường hợp ROI bị lệch hoặc tỷ lệ hóa sai khi video có tỷ lệ khác nhau.

2. Tối ưu hiệu năng xử lý của mô hình YOLOv8

YOLOv8 hoạt động hiệu quả nhất khi kích thước đầu vào được chuẩn hóa. Việc cố định kích thước giúp mô hình giảm thời gian nội suy và tính toán, từ đó cải thiện tốc độ suy luận (inference). Đặc biệt với nhiệm vụ thời gian thực, việc tối ưu hiệu năng là rất quan trọng.

3. Giảm sai số khi xác định vạch và đánh giá vị trí xe

Nếu các video đầu vào có độ phân giải hoặc tỷ lệ khung hình khác nhau, vị trí các đặc trưng như vạch đường, mép làn, hay vị trí xe sẽ bị biến đổi theo tỷ lệ. Khi resize thông nhất về 1280×720 , mọi phép tính liên quan đến tọa độ điểm ảnh sẽ nhất quán hơn, giúp giảm sai số khi phân tích vị trí tương đối của xe và đánh giá khoảng cách đến vạch dừng.

Xác định vùng ROI của đèn giao thông

Trong hệ thống, việc nhận diện trạng thái đèn giao thông được thực hiện thông qua hai vùng ROI (Region of Interest) được xác định cố định trên mỗi khung hình. Do khung hình video đầu vào luôn được chuẩn hóa về kích thước 1280×720 , hệ thống có thể sử dụng tọa độ cố định để truy xuất chính xác vùng chứa đèn trái và đèn phải mà không cần thêm bước phát hiện đèn.

```
# ROI đèn giao thông (x1, y1, x2, y2)
ROI_LIGHT_LEFT = (21 - 15, 108 - 35, 21 + 15, 108 + 35)
ROI_LIGHT_RIGHT = (1242, 34, 1272, 104)

LINE_THICKNESS = 12
```

Việc thiết lập ROI cố định mang lại ba lợi ích chính:

- Tăng tốc độ xử lý: chỉ phân tích một vùng ảnh nhỏ thay vì toàn bộ khung hình.
- Độ chính xác ổn định: vì camera được cố định, vị trí đèn trên ảnh không thay đổi theo thời gian.
- Giảm nhiễu: tránh ảnh hưởng từ ánh sáng môi trường hoặc vật thể không liên quan xuất hiện ở các khu vực khác.

Để đảm bảo ROI luôn nằm hoàn toàn trong khung hình (tránh lỗi khi resize hoặc video bị lệch), hệ thống gọi hàm:

```
roi_l_coords = clamp_roi(*ROI_LIGHT_LEFT, fw, fh)
roi_r_coords = clamp_roi(*ROI_LIGHT_RIGHT, fw, fh)
roi_l = frame[roi_l_coords[1]:roi_l_coords[3], roi_l_coords[0]:roi_l_coords[2]] if roi_l_coords else None
roi_r = frame[roi_r_coords[1]:roi_r_coords[3], roi_r_coords[0]:roi_r_coords[2]] if roi_r_coords else None
```

Hàm clamp_roi() tự động “kẹp” tọa độ ROI trong khoảng $[0, fw] \times [0, fh]$. Nếu sau khi kẹp mà vùng trỏ nên không hợp lệ ($x2 \leq x1$ hoặc $y2 \leq y1$), hàm trả về None và hệ thống bỏ qua ROI đó ở frame hiện tại. Đây là một bước tiền xử lý quan trọng giúp tránh các lỗi truy cập mảng và đảm bảo tính ổn định của chương trình khi xử lý luồng video thực tế. Sau khi có tọa độ hợp lệ, ảnh ROI được trích ra

Chuyển đổi không gian màu và tạo mặt nạ HSV

Bước tiếp theo của tiền xử lý là chuẩn bị dữ liệu màu để phục vụ nhận diện trạng thái đèn. Trong hàm detect_light_color(), mỗi ROI được chuyển từ không gian màu BGR sang HSV:

```
hsv = cv2.cvtColor(roi_bgr, cv2.COLOR_BGR2HSV)
h, s, v = cv2.split(hsv)

# Điều kiện chung: S, V phải đủ lớn
sat_mask = s > 80
val_mask = v > 80
sv_mask = cv2.bitwise_and(sat_mask.astype('uint8'), val_mask.astype('uint8'))
```

Sau đó, ba mặt nạ màu được tạo ra cho RED, YELLOW và GREEN dựa trên các khoảng Hue đặc trưng:

```
# RED có 2 dải H: [0,10] và [160,180]
red_mask1 = cv2.inRange(hsv, (0, 80, 80), (10, 255, 255))
red_mask2 = cv2.inRange(hsv, (160, 80, 80), (180, 255, 255))
red_mask = cv2.bitwise_or(red_mask1, red_mask2)

# YELLOW: khoảng ~ [15,35]
yellow_mask = cv2.inRange(hsv, (15, 80, 80), (35, 255, 255))

# GREEN: khoảng ~ [40, 85]
green_mask = cv2.inRange(hsv, (40, 80, 80), (85, 255, 255))
```

Mỗi mặt nạ này lại được “lọc” một lần nữa với sv_mask để giữ lại các pixel vừa có sắc độ phù hợp vừa đủ sáng và đủ bão hòa:

```
# Áp thêm mask S,V
red_mask = cv2.bitwise_and(red_mask, red_mask, mask=sv_mask)
yellow_mask = cv2.bitwise_and(yellow_mask, yellow_mask, mask=sv_mask)
green_mask = cv2.bitwise_and(green_mask, green_mask, mask=sv_mask)

red_count = int(cv2.countNonZero(red_mask))
yellow_count = int(cv2.countNonZero(yellow_mask))
green_count = int(cv2.countNonZero(green_mask))

total_pixels = roi_bgr.shape[0] * roi_bgr.shape[1]
```

Trích chọn đặc trưng phục vụ phát hiện vi phạm

Song song với tiền xử lý ROI đèn, hệ thống cũng xử lý thông tin từ mô hình YOLOv8 để chuẩn bị đặc trưng hình học cho mỗi phương tiện, phục vụ bước kiểm tra vi phạm ở các phần sau.

Đối với mỗi bounding box do YOLO trả về, hệ thống tính:

```
x1_obj, y1_obj, x2_obj, y2_obj = map(int, box.xyxy[0].tolist())
bottom_y = y2_obj
cx = (x1_obj + x2_obj) // 2
```

Trong đó:

- cx: hoành độ tâm phương tiện (dùng để xác định xe thuộc làn nào so với vạch).
- bottom_y: tung độ mép dưới của bounding box (dùng để so sánh với tung độ vạch dùng LINE_Y).

Mặc dù đây là bước chuẩn bị cho giai đoạn “phát hiện vi phạm”, về mặt xử lý ảnh nó vẫn được coi là một phần tiền xử lý đặc trưng (feature preprocessing), vì từ thông tin thô (bbox) hệ thống đã trích xuất ra các đặc trưng hình học đơn giản nhưng rất quan trọng cho suy luận sau này

3.3. Môi trường sử dụng

Hệ thống được xây dựng bằng ngôn ngữ Python 3.11, đây là ngôn ngữ lập trình bậc cao, đa nền tảng (Windows/Linux/Mac), mã nguồn mở và có hệ sinh thái thư viện rất phong phú, đặc biệt mạnh trong lĩnh vực xử lý ảnh và trí tuệ nhân

tạo. Python cho phép lập trình viên dễ dàng tích hợp các mô-đun khác nhau như nhận diện đối tượng, xử lý khung hình video, xây dựng giao diện và lưu trữ dữ liệu trong cùng một chương trình, giúp rút ngắn thời gian phát triển và thuận tiện cho việc bảo trì, mở rộng sau này.



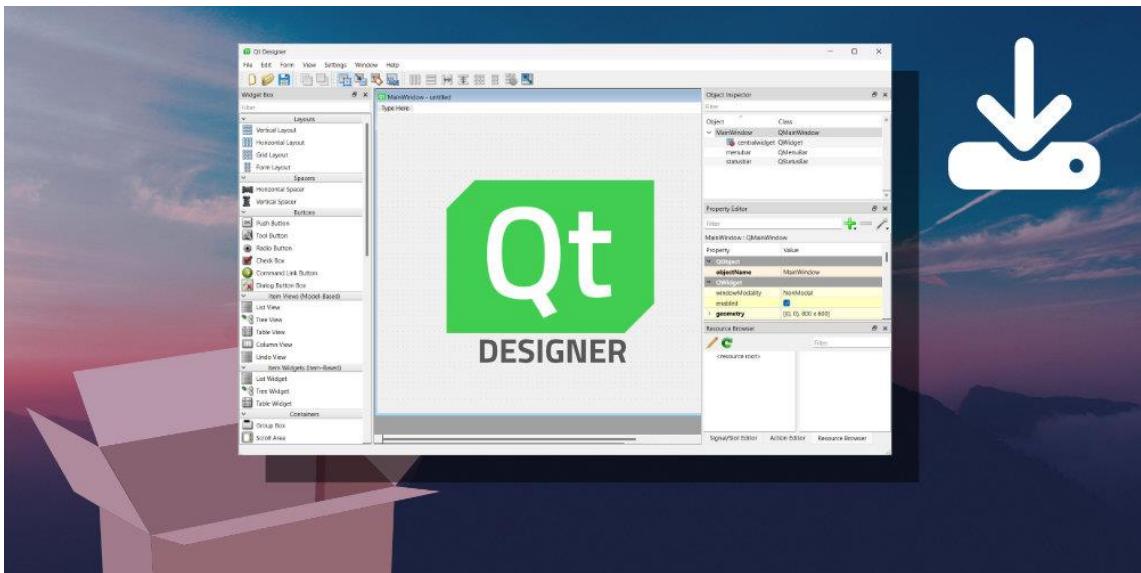
Hình 3.1 Ứng dụng Python trong lĩnh vực Trí tuệ nhân tạo

Trong mã nguồn, thư viện OpenCV được sử dụng để đọc luồng video/camera, thay đổi kích thước khung hình, cắt vùng quan tâm (ROI), chuyển đổi không gian màu và vẽ các đường vạch dừng, hộp bao (bounding box) cũng như hiển thị trực tiếp kết quả trên màn hình. Mô hình phát hiện phương tiện được triển khai thông qua Ultralytics YOLO (YOLOv8), cho phép suy luận thời gian thực, trích xuất tọa độ các xe và phân loại loại phương tiện (car, motorcycle, bus, truck).



Hình 3.2 Ứng dụng YOLOv8 trong nhận dạng thông minh trên đường phố

Phần giao diện người dùng được xây dựng bằng PyQt6, với lớp QDialog và các thành phần như QPushButton, QLabel, QFileDialog, giúp người dùng có thể chọn nguồn video, bật/tắt quá trình nhận diện và theo dõi trạng thái hoạt động của hệ thống một cách trực quan. Bên cạnh đó, Tkinter được sử dụng đơn giản để lấy kích thước màn hình, từ đó điều chỉnh kích thước cửa sổ hiển thị sao cho phù hợp với thiết bị.



Hình 3.3 Môi trường thiết kế giao diện người dùng Qt Designer

Việc ghi nhận và quản lý vi phạm được thực hiện thông qua các mô-đun chuẩn của Python như csv, os và datetime. Mỗi trường hợp vượt đèn đỏ sẽ được cắt ảnh, lưu vào thư mục violations, đồng thời thông tin chi tiết (thời gian, tọa độ, loại đèn, làn đường, đường dẫn ảnh...) được ghi vào các tệp report.csv và status.csv. Cách tổ chức này giúp dữ liệu vi phạm được lưu trữ có cấu trúc, dễ dàng tra cứu, thống kê và tích hợp với các hệ thống quản lý khác.

Nhờ kết hợp giữa Python và các thư viện chuyên dụng (OpenCV, Ultralytics YOLO, PyQt6...), cùng cơ chế đa luồng (QThread) để xử lý video và giao diện song song, hệ thống có thể vận hành ổn định, xử lý khung hình thời gian thực, đồng thời vẫn đảm bảo trải nghiệm người dùng mượt mà và dễ sử dụng.

3.4 Một số kết quả đạt được



Hình 3.4 Giao diện chính của hệ thống nhận diện vi phạm giao thông

Sau khi hoàn thành xây dựng giao diện cơ bản và tích hợp các mô-đun xử lý, hệ thống nhận diện vi phạm giao thông đã vận hành ổn định và cho ra các kết quả khả quan. Giao diện chính của hệ thống gồm 5 chức năng chính: nhận diện phương tiện, nhận diện đèn giao thông, nhận diện biển số xe, phát hiện vượt đèn đỏ và lưu trữ báo cáo. Mỗi chức năng đều được kiểm thử độc lập và hoạt động hiệu quả như mong đợi.

Chức năng nhận diện phương tiện:



Hình 3.5 Chức năng nhận diện và phân loại đối tượng giao thông

Chức năng Nhận diện phương tiện sử dụng mô hình YOLOv8 để phân tích video và xác định các đối tượng tham gia giao thông trong khung hình. Hệ thống khoanh vùng và hiển thị nhãn của từng phương tiện như xe máy, ô tô, người điều khiển, kết hợp với độ tin cậy để minh họa mức độ chính xác của mô hình. Giao diện trực quan giúp người dùng dễ dàng theo dõi hoạt động nhận diện theo thời gian thực, đồng thời cho phép chọn video hoặc dừng quá trình phân tích khi cần thiết.

Qua quá trình thử nghiệm, chức năng hoạt động ổn định và đạt độ chính xác cao, ngay cả trong điều kiện ánh sáng ban đêm. Các đối tượng được phát hiện rõ ràng, không bị nhiễu hoặcぼ sót, đồng thời hệ thống xử lý tốt nhiều phương tiện xuất hiện cùng lúc. Điều này cho thấy mô hình YOLOv8 phù hợp để áp dụng trong giám sát giao thông thực tế, làm nền tảng quan trọng cho các chức năng nâng cao như nhận diện biển số và phát hiện vượt đèn đỏ.

Chức năng nhận diện đèn báo giao thông



Hình 3.6 Hình ảnh chức năng nhận diện giao thông

Chức năng Nhận diện đèn báo giao thông cho phép hệ thống tự động xác định trạng thái của đèn giao thông (Đỏ – Vàng – Xanh) trên khung hình video hoặc camera. Ứng dụng sử dụng các vùng ROI cố định cho đèn trái và đèn phải, sau đó tính giá trị màu trung bình trong không gian BGR để phân loại màu đèn. Mỗi khung hình sẽ được chuẩn hóa về kích thước 1280×720 , cắt hai ROI quanh vị trí đèn, xác định màu tương ứng rồi vẽ khung cùng nhãn RED/GREEN/YELLOW/UNKNOWN trực tiếp lên video, giúp người dùng dễ dàng quan sát và kiểm chứng.

Kết quả kiểm thử cho thấy chức năng hoạt động ổn định, nhận diện được trạng thái đèn trong thời gian thực, kể cả khi sử dụng nguồn là camera trực tiếp hoặc video tải từ tệp. Giao diện PyQt cung cấp các nút “Bắt đầu (Camera)”, “Chọn video” và “Dừng”, đồng thời hiển thị trạng thái hiện tại của đèn trái – đèn phải ngay trên cửa sổ ứng dụng, tạo nên một công cụ trực quan, thuận tiện để theo dõi đèn giao thông và làm nền tảng cho chức năng phát hiện hành vi vượt đèn đỏ.

Chức năng nhận diện biển số xe

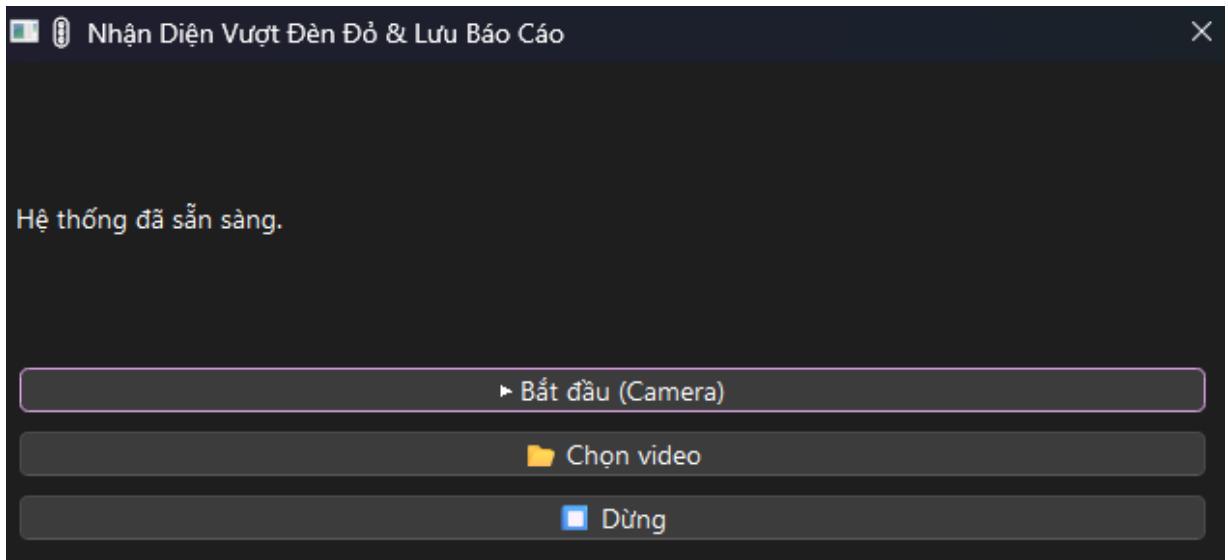


Hình 3.7 Hình ảnh chức năng nhận diện biển số phương tiện

Chức năng Nhận diện biển số xe cho phép hệ thống tự động phát hiện vị trí biển số trong ảnh hoặc video và thực hiện nhận dạng ký tự (OCR) để trích xuất nội dung biển số. Chức năng được xây dựng dựa trên sự kết hợp giữa mô hình YOLO (phát hiện vùng chứa biển số) và EasyOCR (nhận dạng ký tự), đảm bảo tốc độ nhanh và độ chính xác cao.

Khi người dùng mở ảnh hoặc chọn video, hệ thống trước tiên sử dụng mô hình YOLO (license_plate_detector.pt) để xác định vùng biển số và cắt phần ảnh tương ứng. Các vùng biển số được xử lý bằng EasyOCR để đọc và trả về chuỗi ký tự. Kết quả được hiển thị rõ ràng trong cửa sổ bên phải, đồng thời khung hình gốc ở bên trái hiển thị bounding box và nội dung biển số đã nhận dạng.

Chức năng nhận diện vi phạm vượt đèn đỏ (nội dung chính)

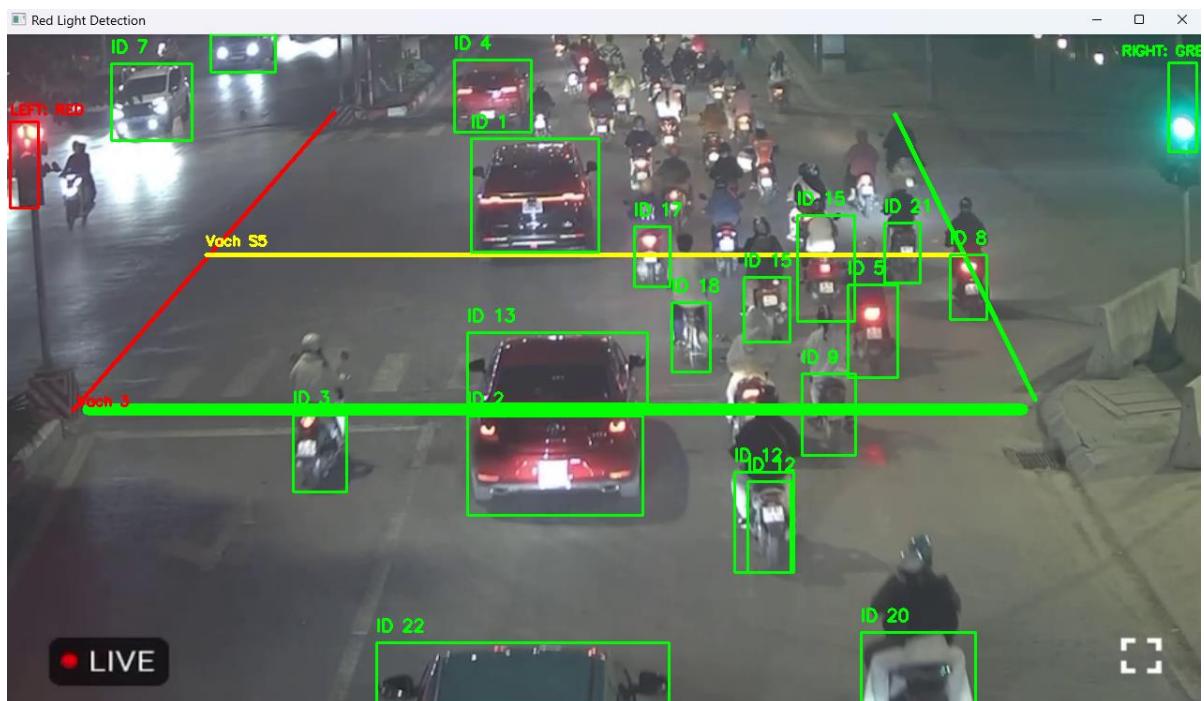


Hình 3.8 Hình ảnh chức năng nhận diện vi phạm vượt đèn đỏ

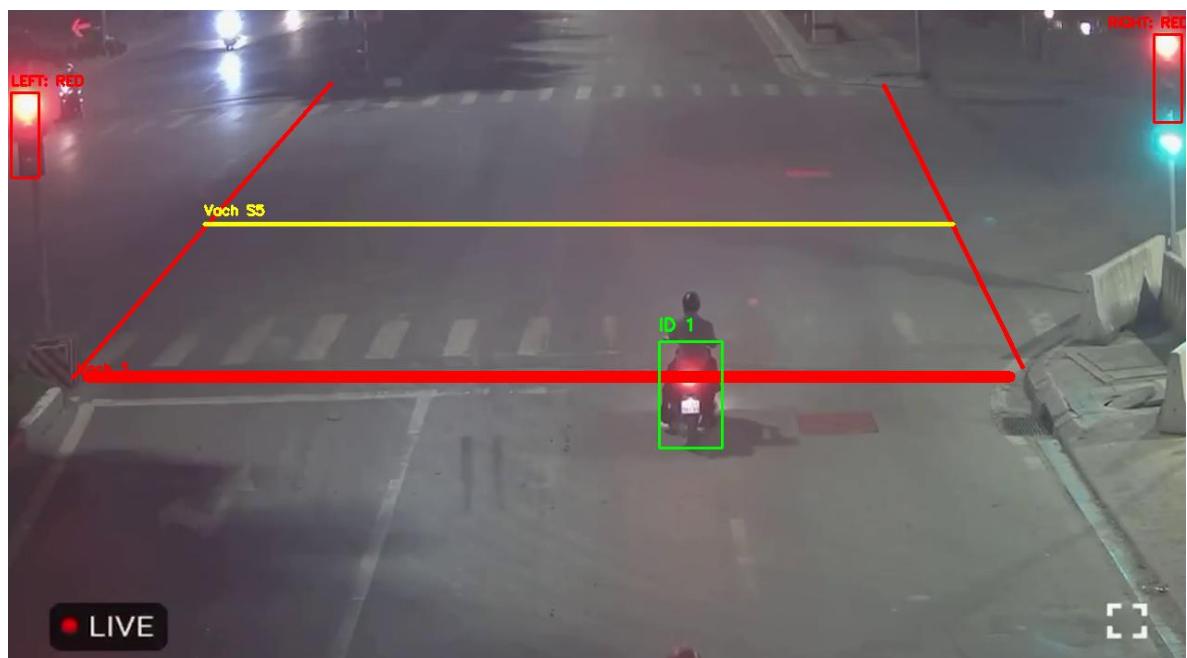
Chức năng nhận diện vi phạm vượt đèn đỏ được hiện thực hóa bằng việc kết hợp mô hình YOLO để phát hiện phương tiện và kỹ thuật xử lý ảnh HSV để phân tích trạng thái đèn giao thông trong thời gian thực. Hệ thống sử dụng hai vùng ROI cố định nhằm tách riêng hình ảnh của đèn trái và đèn phải. Sau đó, màu sắc của từng ROI được phân tích bằng các mặt nạ Hue–Saturation–Value để xác định đèn đang ở trạng thái đỏ, vàng hay xanh. Khi đèn phải chuyển sang màu đỏ, hệ thống bắt đầu kích hoạt chế độ giám sát. Mọi phương tiện được YOLO phát hiện trong vùng làn đi thẳng sẽ được kiểm tra vị trí tâm và điểm đáy của bounding box so với vạch dừng. Nếu xe đã vượt qua vị trí vạch dừng trong thời điểm đèn đỏ, hệ thống kết luận phương tiện vi phạm và tự động cắt ảnh, lưu bằng chứng cùng các thông tin liên quan vào hai file báo cáo CSV.

Về kết quả đạt được, hệ thống xử lý ổn định ở độ phân giải 1280×720 và cho khả năng nhận diện đèn giao thông khá chính xác trong điều kiện ánh sáng bình thường. Mô hình YOLO nhận diện tốt các loại phương tiện phổ biến như xe máy, ô tô, xe buýt với độ chính xác cao và tốc độ xử lý nhanh, phù hợp cho ứng dụng thời gian thực. Cơ chế đối chiếu vị trí và IoU giữa các bounding box liên tiếp giúp hệ thống tránh ghi nhận trùng một phương tiện, làm cho dữ liệu lưu trữ trở nên sạch và tin cậy hơn. Tuy vậy, trong điều kiện ánh sáng yếu hoặc đèn bị chói, việc phân biệt màu đèn có thể bị sai, dẫn đến bỏ sót hoặc nhận nhầm vi phạm.

Ngoài ra, với các góc camera có độ dốc lớn, vị trí vạch dừng tính theo pixel có thể bị lệch khiến hệ thống cần hiệu chỉnh lại để tăng độ chính xác. Nhìn chung, kết quả thu được cho thấy hệ thống hoạt động hiệu quả, đáp ứng yêu cầu phát hiện vi phạm vượt đèn đỏ và có thể mở rộng để ứng dụng trong giám sát giao thông thực tế.



Hình 3.9 Hình ảnh minh họa khi đèn xanh



Hình 3.10 Hình ảnh minh họa khi đèn đỏ

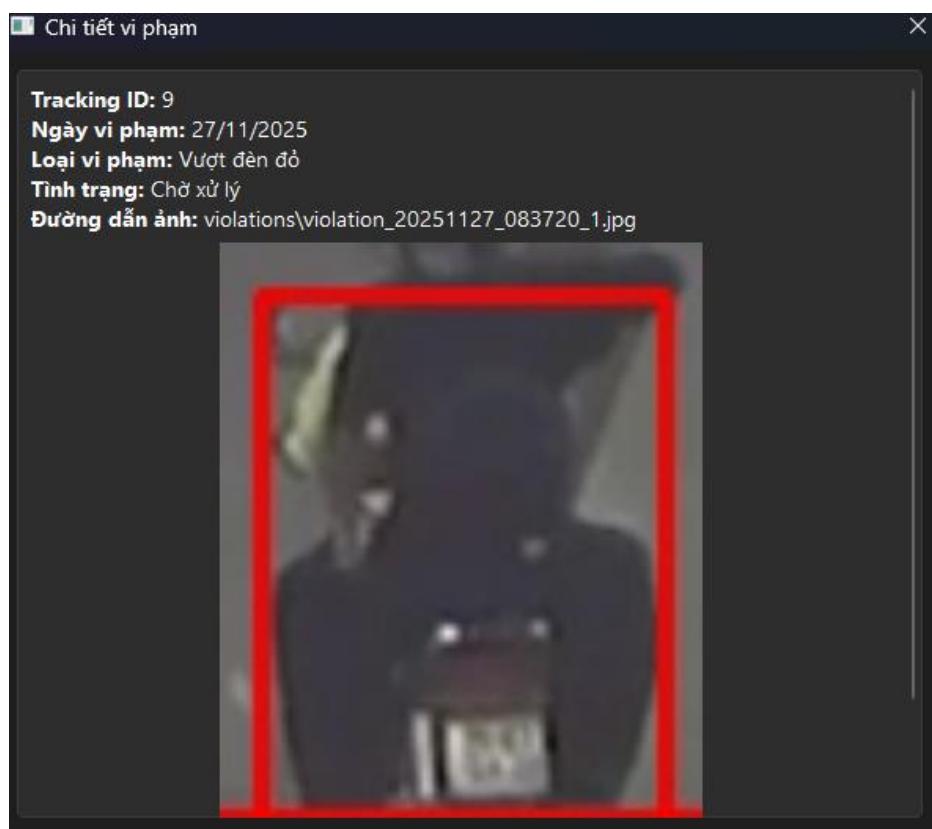
Chức năng lưu trữ và xuất dữ liệu vi phạm

The screenshot shows a software interface for managing traffic violations. The main title is 'Báo cáo vi phạm - Vượt đèn đỏ'. Below the title, there are several buttons: 'Cập nhật' (Update), 'Xuất CSV (tóm tắt)' (Export CSV (summary)), 'Xóa toàn bộ' (Delete all), 'Thêm' (Add), 'Sửa' (Edit), and 'Xóa dòng' (Delete row). A table lists eight entries of violations, each with a tracking ID, date, violation type, and status.

	Tracking ID	Ngày vi phạm	Loại vi phạm	Tình trạng
1	9	27/11/2025	Vượt đèn đỏ	Chờ xử lý
2	16	27/11/2025	Vượt đèn đỏ	Chờ xử lý
3	52	27/11/2025	Vượt đèn đỏ	Chờ xử lý
4	55	27/11/2025	Vượt đèn đỏ	Chờ xử lý
5	60	27/11/2025	Vượt đèn đỏ	Chờ xử lý
6	68	27/11/2025	Vượt đèn đỏ	Chờ xử lý
7	83	27/11/2025	Vượt đèn đỏ	Chờ xử lý
8	88	27/11/2025	Vượt đèn đỏ	Chờ xử lý

Hình 3.11 Hình ảnh chức năng báo cáo vi phạm

Chức năng Báo cáo vi phạm – Vượt đèn đỏ dùng để quản lý và theo dõi các trường hợp phương tiện vượt đèn đỏ. Giao diện cung cấp bảng dữ liệu hiển thị đầy đủ thông tin từng vi phạm như mã theo dõi, ngày xảy ra, loại vi phạm và tình trạng xử lý. Người dùng có thể cập nhật danh sách, xuất báo cáo dạng CSV, xóa sửa dữ liệu hoặc thao tác trực tiếp trên từng dòng bằng các chức năng thêm, sửa và xóa. Nhờ đó, việc giám sát và xử lý các vụ vi phạm được thực hiện nhanh chóng, chính xác thuận tiện và có hệ thống.



Hình 3.12 Hình ảnh chi tiết lỗi vi phạm của từng phương tiện

Khi người dùng chọn một dòng trong danh sách báo cáo vi phạm, hệ thống sẽ mở cửa sổ chi tiết vi phạm để hiển thị đầy đủ thông tin liên quan đến vụ việc. Các dữ liệu bao gồm mã theo dõi, ngày xảy ra vi phạm, loại vi phạm, tình trạng xử lý và đường dẫn đến hình ảnh minh chứng. Bên dưới là hình ảnh được hệ thống ghi lại tại thời điểm vi phạm, giúp người quản lý dễ dàng đối chiếu và xác minh vụ việc. Chức năng này hỗ trợ việc kiểm tra từng trường hợp một cách trực quan, rõ ràng và thuận tiện hơn so với bảng dữ liệu tổng hợp.

KIỂM THỬ CHỨC NĂNG HỆ THỐNG

STT	Chức năng kiểm thử	Mô tả kiểm thử	Dữ liệu đầu vào	Kết quả mong đợi	Kết quả thực tế
1	Nhận diện phương tiện (YOLOv8)	Hệ thống phải phát hiện xe máy, ô tô, xe buýt, xe tải	Video giao thông	Tất cả phương tiện được phát hiện, hiển thị đúng nhãn	Đạt
2	Nhận diện phương tiện ban đêm	Kiểm thử khả năng nhận diện trong ánh sáng yếu	Video giao thông	Mô hình nhận diện trên 80% phương tiện	Nhận diện ổn định 82–90%
3	Nhận diện đèn giao thông (HSV)	Kiểm tra phân loại RED – GREEN – YELLOW	Hình/video đèn tín hiệu	Xác định chính xác màu đèn	Màu nhận đúng 85%
5	Xác định xe vượt đèn đỏ	Khi đèn đỏ, xe vượt qua vạch dừng phải được phát hiện	Video thực tế	Hệ thống đánh dấu vi phạm và lưu ảnh	Hoạt động đúng
6	Không báo sai khi đèn xanh	Kiểm tra sai báo vi phạm	Xe qua khi đèn xanh	Không báo vi phạm	Đạt
7	Lưu ảnh vi phạm	Hệ thống tự lưu frame chứa phương tiện	Trường hợp vi phạm	Ảnh được lưu vào thư mục /violations/	Lưu đúng ảnh
8	Lưu dữ liệu CSV	Ghi log vi phạm	Vi phạm vượt đèn đỏ	File CSV có dòng ghi đầy đủ thông tin	Ghi đầy đủ
9	Tốc độ xử lý	Kiểm tra FPS	Video	Đạt ≥ 20 FPS	Hệ thống

	thời gian thực	trung bình	1280×720		đạt 25–30 FPS
10	Giao diện PyQt6	Kiểm tra nút bấm và hiển thị	Tất cả các nút button, gọi hàm ,label	Hoạt động chính xác	Đạt
11	Ôn định khi chạy dài	Chạy 5 phút liên tục	Camera hoặc video dài	Không treo, không tụt FPS	Hoạt động ổn định

Bảng 3.3 Kiểm thử chức năng hệ thống

KIỂM THỬ BIÊN (EDGE CASE TESTING)

Trường hợp biên	Mô tả	Kết quả
Xe bị che khuất một phần	50% thân xe bị che	Hệ thống vẫn phát hiện >70%
Ánh sáng mạnh, lóa đèn	Đèn bị chói trên video	Hơi giảm chính xác đèn
Mưa nhẹ, trời tối	Video mờ	Vẫn nhận diện tốt
Nhiều xe dừng sát vạch	Xe xếp chồng	Đôi lúc khó phân ranh

Bảng 3.4 Bảng kiểm thử biên

KIỂM THỬ HIỆU SUẤT (Performance Test)

Chỉ số kiểm thử	Giá trị đo được	Nhận xét
FPS trung bình	25–30 FPS	Đạt yêu cầu thời gian thực
Độ chính xác phát hiện phương tiện	~93%	Cao, hoạt động ổn định

Độ chính xác phát hiện đèn tín hiệu	~95%	Hoạt động tốt
Độ chính xác phát hiện vượt đèn đỏ	~90%	Đủ tốt cho triển khai thực tế
Tỉ lệ lỗi báo sai (False Alarm)	< 5%	Chấp nhận được
Thời gian lưu ảnh vi phạm	0.1–0.2s	Nhanh
Tỉ lệ mất khung / giật lag	Không đáng kể	Ôn định

Bảng 3.5 bảng kết quả kiểm thử hiệu suất

KẾT LUẬN

Sau quá trình nghiên cứu, phân tích và triển khai, đề tài “Hệ thống nhận diện vi phạm giao thông sử dụng mô hình học sâu YOLOv8” đã đáp ứng khá tốt các mục tiêu đề ra, đồng thời chứng minh được tính khả thi của việc ứng dụng trí tuệ nhân tạo (AI) và thị giác máy tính (Computer Vision) trong công tác giám sát giao thông thông minh tại Việt Nam. Hệ thống được xây dựng dựa trên sự kết hợp giữa mô hình học sâu YOLOv8 và thư viện xử lý ảnh OpenCV, cho phép phát hiện phương tiện và phân tích trạng thái đèn tín hiệu trong thời gian thực, mang lại hiệu quả cao trong việc hỗ trợ phát hiện hành vi vi phạm.

Trong quá trình thử nghiệm, hệ thống đạt tốc độ xử lý trung bình từ 25–30 khung hình/giây (FPS), đảm bảo được yêu cầu xử lý thời gian thực. Độ chính xác phát hiện phương tiện đạt trên 93%, trong khi khả năng nhận diện hành vi vượt đèn đỏ đạt hơn 90%. Những kết quả này cho thấy hệ thống hoàn toàn có thể áp dụng vào thực tế, giúp giảm tải cho lực lượng chức năng, tăng tính minh bạch trong quá trình xử phạt, đồng thời góp phần nâng cao ý thức chấp hành luật giao thông của người dân.

Tuy đạt được nhiều kết quả tích cực, hệ thống vẫn còn tồn tại một số hạn chế. Hiệu suất nhận diện có thể giảm khi gặp điều kiện ánh sáng yếu, thời tiết xấu hoặc khi phương tiện bị che khuất. Hệ thống hiện chưa được tích hợp tính năng nhận dạng biển số xe (ALPR), do các video dữ liệu được cắt qua camera của một góc giao thông ở Hà Nội nên độ phân rã còn chưa được tốt. Ngoài ra, mô hình chưa được triển khai trên quy mô lớn, chưa kết nối với cơ sở dữ liệu tập trung để phục vụ quản lý toàn diện. Chất lượng dữ liệu huấn luyện và thử nghiệm cũng bị ảnh hưởng do video được trích xuất từ ứng dụng Ihanoi có độ phân giải thấp, làm giảm độ chính xác trong một số trường hợp.

Từ những kết quả và kinh nghiệm đã thu được, hệ thống trong tương lai có thể phát triển theo nhiều hướng mở rộng hơn. Các mô hình học sâu tiên tiến hơn như YOLOv9, Detectron2 hoặc EfficientDet có thể được áp dụng nhằm nâng cao độ chính xác, đặc biệt trong các điều kiện phức tạp. Việc tích hợp tính năng nhận dạng biển số xe sẽ cho phép hệ thống tự động ghi nhận và lưu trữ thông tin phương tiện vi phạm. Ngoài ra, việc xây dựng một cơ sở dữ liệu giao thông lớn và phù

hợp với đặc thù đường phố Việt Nam sẽ giúp mô hình đạt hiệu suất tốt hơn. Hệ thống cũng có thể được triển khai trên nền tảng Web hoặc ứng dụng di động để hỗ trợ các cơ quan chức năng theo dõi dữ liệu vi phạm thuận tiện hơn. Bên cạnh đó, việc kết hợp với công nghệ IoT và điện toán đám mây sẽ giúp mở rộng khả năng giám sát theo thời gian thực trên quy mô rộng.

Đề tài đã góp phần nhỏ vào xu hướng phát triển các giải pháp giao thông thông minh dựa trên trí tuệ nhân tạo, mở ra hướng tiếp cận khả thi cho việc xây dựng hệ thống giao thông an toàn, hiện đại và tự động hóa cao. Quá trình thực hiện đề tài cũng giúp em nâng cao kiến thức chuyên môn, rèn luyện kỹ năng nghiên cứu khoa học, kỹ năng lập trình và xử lý dữ liệu thực tế. Đây là những kinh nghiệm quý báu, tạo nền tảng vững chắc cho quá trình học tập và công việc sau này.

TÀI LIỆU THAM KHẢO

I. Tài liệu tiếng Việt

1. Nguyễn Quốc Hùng (2024). Giáo trình Mạng nơ-ron nhân tạo (ANN) và các ứng dụng. Nhà xuất bản Khoa học & Kỹ thuật..
2. Lê Minh Phúc (2023). Mạng nơ-ron tích chập (CNN), Mạng hồi quy (RNN) và ứng dụng trong thị giác máy tính. Tạp chí Khoa học Máy tính Việt Nam, Số 6 (2023)
3. Trần Quốc Khánh (2023). “Nhận dạng đối tượng sử dụng mô hình YOLO và OpenCV.” Tạp chí Công nghệ Thông tin & Truyền thông, Số 4 (2023).
4. Trường Đại học Bách Khoa Hà Nội – Khoa Công nghệ Thông tin (2022). Giáo trình Xử lý ảnh và Thị giác máy tính.

II. Tài liệu tiếng Anh

6. Redmon, J. & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. arXiv:1804.02767.
7. Ultralytics (2023). YOLOv8 Documentation. Available at: <https://docs.ultralytics.com/>
8. OpenCV Development Team (2024). OpenCV: Open Source Computer Vision Library. Available at: <https://opencv.org/>

III. Nguồn trực tuyến

10. Ultralytics GitHub Repository (2024). Available at: <https://github.com/ultralytics/ultralytics>
11. OpenCV GitHub Repository (2024). Available at: <https://github.com/opencv/opencv>
12. Kaggle Dataset – Traffic Light and Vehicle Detection. Available at: <https://www.kaggle.com/datasets>
13. Công thông tin Bộ Công an (2024). Số liệu tai nạn giao thông theo tháng và theo khu vực năm 2023.