

BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP HÀ NỘI



ĐỒ ÁN TỐT NGHIỆP

NGÀNH: KHOA HỌC MÁY TÍNH

**ĐỀ TÀI: NGHIÊN CỨU VÀ PHÁT TRIỂN
ỨNG DỤNG NHẬN DIỆN BỆNH Ở LOÀI
MÈO BẰNG MÔ HÌNH YOLOV11**

Giảng viên hướng dẫn : TS. Nguyễn Mạnh Cường

Sinh viên thực hiện : Cù Đức Xuân

Mã sinh viên : 2021605437

Hà Nội – 2025

MỤC LỤC

MỤC LỤC.....	i
DANH MỤC HÌNH ẢNH	iv
DANH MỤC BẢNG BIỂU	vii
CÁC THUẬT NGỮ VÀ TỪ VIẾT TẮT.....	viii
LỜI CẢM ƠN	ix
LỜI NÓI ĐẦU	1
CHƯƠNG 1. PHÁT BIỂU BÀI TOÁN	3
1.1. Tổng quan về thị giác máy tính.....	3
1.1.1. Định nghĩa	3
1.1.2. Thực trạng	3
1.1.3. Các tác vụ hiện nay	4
1.2. Tổng quan về phát hiện đối tượng.....	6
1.2.1. Định nghĩa	6
1.2.2. Ứng dụng.....	7
1.2.3. Khó khăn và thách thức.....	7
1.3. Bài toán phát hiện bệnh ở mèo	8
1.3.1. Giới thiệu chung	8
1.3.2. Tình hình nghiên cứu trong và ngoài nước	9
1.3.3. Phát biểu chi tiết bài toán	9
1.3.4. Mục tiêu.....	10
1.3.5. Ứng dụng thực tiễn.....	10
1.3.6. Cơ hội và thử thách	11
CHƯƠNG 2. MỘT SỐ KỸ THUẬT GIẢI QUYẾT BÀI TOÁN.....	12
2.1. Phương hướng tiếp cận.....	12
2.2. Mạng nơ ron tích chập.....	12
2.2.1. Tổng quan.....	13
2.2.2. Cấu trúc	14
2.2.3. Phép tương quan chéo	15

2.2.4. Bộ lọc.....	17
2.2.5. Lớp tích chập.....	19
2.2.6. Lớp pooling	23
2.2.7. Lớp kết nối đầy đủ.....	25
2.2.8. Các hàm kích hoạt	26
2.3. Mô hình YOLOv11	29
2.3.1. Giới thiệu về thuật toán YOLO	29
2.3.2. YOLOv11	33
CHƯƠNG 3. THỰC NGHIỆM	41
3.1. Chuẩn bị dữ liệu	41
3.1.1. Thu thập dữ liệu	41
3.1.2. Gán nhãn dữ liệu	42
3.1.3. Phân tích dữ liệu.....	43
3.1.4. Tiền xử lý dữ liệu	45
3.1.5. Tăng cường dữ liệu.....	45
3.2. Huấn luyện mô hình	47
3.2.1. Lựa chọn siêu tham số.....	47
3.2.2. Kết quả sau khi huấn luyện	49
CHƯƠNG 4. XÂY DỰNG ỨNG DỤNG	53
4.1. Giới thiệu công cụ sử dụng	53
4.1.1. Streamlit Framework.....	53
4.1.2. Git	53
4.2. Phân tích thiết kế hệ thống	54
4.2.1. Biểu đồ use case	55
4.2.2. Đặc tả use case	55
4.2.3. Phân tích use case.....	59
4.3. Các chức năng sau khi cài đặt	63
4.4. Triển khai sản phẩm	66
KẾT LUẬN.....	68

TÀI LIỆU THAM KHẢO.....	69
-------------------------	----

DANH MỤC HÌNH ẢNH

Hình 1.1. Các tác vụ của thị giác máy tính	5
Hình 1.2. Minh họa bài toán	10
Hình 2.1. Kiến trúc CNN cơ bản.....	15
Hình 2.2. Minh họa phép tương quan chéo.....	16
Hình 2.3. Minh họa cách độ trượt hoạt động	16
Hình 2.4. Ví dụ về bộ lọc làm sắc nét.....	17
Hình 2.5. Ví dụ về bộ lọc tạo hiệu ứng nổi.....	18
Hình 2.6. Minh họa các kiểu tích chập phổ biến	19
Hình 2.7. Minh họa tích chập 2 chiều	20
Hình 2.8. Minh họa tích chập 1 chiều	21
Hình 2.9. Minh họa tích chập 3 chiều	22
Hình 2.10. Minh họa tích chập 3 chiều	22
Hình 2.11. Minh họa tích chập nhóm.....	23
Hình 2.12. Minh họa kiểu Max Pooling	24
Hình 2.13. Minh họa kiểu Average Pooling	24
Hình 2.14. Minh họa kiểu Global Max Pooling.....	25
Hình 2.15. Minh họa kiểu Global Average Pooling	25
Hình 2.16. Minh họa lớp kết nối đầy đủ	26
Hình 2.17. Minh họa hàm sigmoid	27
Hình 2.18. Minh họa hàm tanh	27
Hình 2.19. Minh họa hàm ReLU.....	28
Hình 2.20. Minh họa hàm Leaky ReLU	28
Hình 2.21. Minh họa hàm Noise ReLU	29
Hình 2.22. Lịch sử phát triển của YOLO.....	29
Hình 2.23. Kiến trúc YOLO ban đầu	30
Hình 2.24. Minh họa ý tưởng của thuật toán YOLO	31
Hình 2.25. Công thức tính Loss function của YOLO	32
Hình 2.26. Minh họa cách tính IOU.....	33

Hình 2.27. Kiến trúc YOLOv11.....	34
Hình 2.28. Khối tích chập chuẩn hiện nay.....	35
Hình 2.29. Khối tích chập không có hàm kích hoạt và khối tích chập theo chiều sâu.....	35
Hình 2.30. Khối bottleneck	36
Hình 2.31. Khối C2f.....	37
Hình 2.32. Khối C3k	37
Hình 2.33. Khối C3k2 phiên bản đầu tiên và thứ hai.	38
Hình 2.34. Khối SPPF	38
Hình 2.35. Khối C2PSA và PSA.....	39
Hình 2.36. Khối phát hiện.....	40
Hình 3.1. Minh họa bệnh mụn trứng cá ở mèo	41
Hình 3.2. Minh họa bệnh nhiễm trùng mắt ở mèo.....	41
Hình 3.3. Minh họa bệnh rụng lông ở mèo	42
Hình 3.4. Minh họa bệnh u ở mèo	42
Hình 3.5. Minh họa bệnh nấm da ở mèo.....	42
Hình 3.6. Minh họa gán nhãn với công cụ Roboflow.....	43
Hình 3.7. Kiểm tra số lượng nhãn phân bố	43
Hình 3.8. Kiểm tra số lượng ảnh chứa mỗi nhãn	44
Hình 3.9. Kết quả kiểm tra dữ liệu trong tập train và validation	44
Hình 3.10. Quy trình tiền xử lý	45
Hình 3.11. Kết quả tiền xử lý dữ liệu.....	45
Hình 3.12. Quy trình tăng cường ảnh	46
Hình 3.13. Minh họa dữ liệu sau khi biến đổi	46
Hình 3.14. Dữ liệu sau khi tăng cường	46
Hình 3.15. Minh họa lựa chọn siêu tham số	47
Hình 3.16. Đo lường loss trên tập train.....	49
Hình 3.17. Đo lường loss trên tập validation	50
Hình 3.18. Độ đo của mô hình	51

Hình 4.1. Biểu đồ use case tổng quát.....	55
Hình 4.2. Biểu đồ phân rã use case Execution.....	55
Hình 4.3. Biểu đồ trình tự use case Select Task	59
Hình 4.4. Biểu đồ lớp phân tích use case Select Task	60
Hình 4.5. Biểu đồ trình tự use case Select Model.....	60
Hình 4.6. Biểu đồ lớp phân tích use case Select Model	60
Hình 4.7. Biểu đồ trình tự use case Select Source.....	61
Hình 4.8. Biểu đồ lớp phân tích use case Select Source.....	61
Hình 4.9. Biểu đồ trình tự use case Execution	61
Hình 4.10. Biểu đồ lớp phân tích use case Execution	62
Hình 4.11. Biểu đồ trình tự use case Upload	62
Hình 4.12. Biểu đồ lớp phân tích use case Upload	62
Hình 4.13. Minh họa chức năng Select Task	63
Hình 4.14. Minh họa chức năng Select Model	63
Hình 4.15. Minh họa chức năng Select Source.....	63
Hình 4.16. Minh họa chức năng Execution với source là Image.....	64
Hình 4.17. Minh họa chức năng Execution với source là Video.....	64
Hình 4.18. Minh họa kết quả thực hiện nhận diện trên video.....	65
Hình 4.19. Minh họa chức năng Execution với source là Webcam	65
Hình 4.20. Kiểm thử định dạng khi upload image.....	66
Hình 4.21. Kiểm thử định dạng khi upload video.....	66
Hình 4.22. Luồng triển khai với streamlit cloud.....	67
Hình 4.23. Minh họa cấu hình trước triển khai.....	67

DANH MỤC BẢNG BIỂU

Bảng 4.1. Đặc tả use case Select Task	56
Bảng 4.2. Đặc tả use case Select Model	57
Bảng 4.3. Đặc tả use case Select Source.....	57
Bảng 4.4. Đặc tả use case Execution	58
Bảng 4.5. Đặc tả use case Upload.....	59

CÁC THUẬT NGỮ VÀ TỪ VIẾT TẮT

Thuật ngữ / từ viết tắt	Ý nghĩa
AI	Trí tuệ nhân tạo (Artificial Intelligence)
CNN	Mạng nơ-ron tích chập (Convolutional Neural Network)
Bouding box	Hộp giới hạn
Generative AI	Trí tuệ nhân tạo tạo sinh
Bounding box	Hộp giới hạn
IOU	Chỉ số đánh giá được sử dụng để đo độ chính xác của Object detector trên tập dữ liệu cụ thể (Intersection Over Union)

LỜI CẢM ƠN

Lời đầu tiên cho phép em xin được gửi lời cảm ơn sâu sắc tới các thầy cô trong Trường Công nghệ thông tin và Truyền thông - Trường Đại học Công Nghiệp Hà Nội, những người đã hết mình truyền đạt và chỉ dẫn cho chúng em những kiến thức, những bài học quý báu và bổ ích. Những kiến thức của thầy cô cho em nền tảng vững chắc để xác định hướng đi và phát triển phù hợp cho đê tài của mình.

Đặc biệt, cá nhân em xin được bày tỏ lòng chân thành biết ơn tới thầy giáo TS. Nguyễn Mạnh Cường, người trực tiếp hướng dẫn, chỉ bảo chúng em trong suốt quá trình làm **Đồ Án Tốt Nghiệp**. Nhờ sự tận tâm chỉ dạy và đưa ra những gợi ý quý báu đã giúp em hiểu rõ hơn về trí tuệ nhận tạo, đặc biệt là mạng nơ ron để ứng dụng thực hiện đê tài : “Nghiên cứu và phát triển ứng dụng nhận diện bệnh ở loài mèo bằng mô hình YOLOv11” . Sự kiên nhẫn và đồng hành của thầy đã giúp em hoàn thành đê tài một cách tốt nhất.

Cuối cùng, em xin gửi tình cảm sâu sắc tới những cá nhân, bạn bè và gia đình đã luôn bên cạnh khuyến khích, động viên, giúp đỡ cả về vật chất lẫn tinh thần cho em trong suốt quy trình nghiên cứu và thực hiện đê tài này. Những nhận xét, đóng góp của mọi người đã giúp báo cáo được hoàn thiện và chất lượng hơn.

Em xin chân thành cảm ơn!

Sinh viên

Cù Đức Xuân

LỜI NÓI ĐẦU

Trong những năm gần đây, công nghệ thị giác máy tính với sự phát triển mạnh của mạng nơ ron tích chập (Convolutional Neural Networks - CNN) đã trở thành một trong những tiến bộ đột phá nhất trong lĩnh vực trí tuệ nhân tạo và xử lý hình ảnh. CNN đã chứng minh hiệu quả vượt trội trong việc nhận dạng và phân loại hình ảnh, được ứng dụng rộng rãi trong nhiều lĩnh vực như sản xuất, y tế, và giao thông.

Đồng thời, con người ngày càng dành nhiều sự quan tâm và chăm sóc đến động vật, đặc biệt là mèo. Với các gia đình có điều kiện có họ không tiếc tiền để chi và chăm sóc thú cưng của họ, thậm chí một bộ phận người yêu thú cưng cũng không tiếc đổ tiền vào các dịch vụ chăm sóc chúng nhằm thể hiện sở thích cá nhân. Bên cạnh đó, các quán cà phê và các mô hình kinh doanh sử dụng mèo để thu hút khách hàng cũng đang phát triển mạnh mẽ.

Đề tài “**Nghiên cứu và phát triển ứng dụng nhận diện bệnh ở loài mèo bằng mô hình YOLOv11**” tập trung vào việc phát triển một hệ thống nhận diện bệnh ở loài mèo, cụ thể là các bệnh ngoài da dựa trên công nghệ mạng nơ ron tích chập (Convolutional Neural Network - CNN). CNN là một trong những phương pháp học sâu (deep learning) hiện đại và mạnh mẽ nhất, được sử dụng rộng rãi trong các bài toán về nhận diện hình ảnh và xử lý ảnh. Với khả năng tự động học các đặc trưng quan trọng từ dữ liệu hình ảnh, CNN đã chứng minh được hiệu quả vượt trội trong nhiều ứng dụng thực tế.

Đề tài được thực hiện hướng tới việc phát hiện bệnh trên mèo một cách sớm nhất, tránh các nguy cơ lây nhiễm ảnh hưởng tới cuộc sống con người, giảm thiểu các chi phí chữa bệnh do phát hiện bệnh muộn. Đồng thời mở rộng việc ứng dụng công nghệ trí tuệ nhân tạo trong ngành thú y tại Việt Nam và trên thế giới.

Để có thể thực hiện đề tài, ta sẽ cần các kiến thức cơ bản về xử lý ảnh, trí tuệ nhân tạo, học máy, học sâu. Các kiến thức về quy trình thực hiện như tiền xử lý dữ liệu, huấn luyện mô hình, tối ưu hóa mô hình là đặc biệt cần thiết.

Ngoài ra, các kiến thức về phân tích và thiết kế phần mềm, xây dựng phần mềm cũng là không thể thiếu để hoàn thành tốt đề tài.

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

Chương 1: Phát biểu bài toán

Giới thiệu tổng quan về bài toán, xác định dữ liệu đầu vào, dữ liệu đầu ra, đặc điểm và các kiến thức cơ bản phục vụ giải quyết bài toán.

Chương 2: Một số kĩ thuật giải quyết bài toán

Trình bày một số kĩ thuật nhằm phục vụ việc giải quyết bài toán. Cụ thể là một số kĩ thuật, phương pháp phục vụ việc tiền xử lý, xây dựng mô hình và triển khai sản phẩm.

Chương 3: Thực nghiệm

Trình bày quy trình thực nghiệm, các kết quả đạt được sau khi xây dựng chương trình và đánh giá dựa trên các độ đo cho mô hình đã được xây dựng.

Chương 4: Xây dựng ứng dụng

Trình bày các công cụ, quy trình được sử dụng để xây dựng một ứng dụng cụ thể nhằm phục vụ mục đích nhận diện bệnh trên mèo. Ngoài ra, trình bày kết quả sau khi ứng dụng mô hình đã xây dựng vào ứng dụng đã xây dựng.

Tổng kết lại, ta thấy đây là một đề tài khá rộng và cần có nhiều thời gian để nghiên cứu và thực nghiệm thêm. Ngoài ra vấn đề dữ liệu còn khá khan hiếm nên sẽ rất cần việc ứng dụng các kĩ thuật và xử lý dữ liệu một cách phù hợp nhằm mục đích tối ưu hóa kết quả đầu ra một cách chính xác. Cần đánh giá, xem xét các khó khăn của bài toán và đưa ra giải pháp phát triển cho tương lai.

CHƯƠNG 1. PHÁT BIỂU BÀI TOÁN

1.1. Tổng quan về thị giác máy tính

1.1.1. Định nghĩa

Thị giác máy tính hay computer vision là một lĩnh vực trong trí tuệ nhân tạo (AI) nhằm biến máy tính thành một công cụ có thể hiểu và diễn giải hình ảnh hoặc video giống như cách con người nhìn và phân tích thế giới xung quanh. Qua đó, hỗ trợ con người trong việc tự động hóa các nhiệm vụ hay công việc hàng ngày, đặc biệt là những công việc đơn giản nhưng yêu cầu hiệu suất cao. Để thực hiện điều đó, các thuật toán và mô hình học sâu, đặc biệt là những mô hình dựa trên mạng nơ-ron nhân tạo đã ra đời nhằm trích xuất, xử lý, phân tích nội dung và tạo ra các phép toán phù hợp với các bộ dữ liệu mà con người đưa vào.

1.1.2. Thực trạng

Hiện nay, thị giác máy tính đang được ứng dụng vào rất nhiều lĩnh vực, đặc biệt là lĩnh vực tự động hóa, robotic và y tế. Những lĩnh vực này đều đang phục vụ tích cực cho việc sản xuất, tự động hóa quy trình, giải quyết các vấn đề mà con người không thực hiện được và nâng cao chất lượng sống của con người.

Trên thế giới hiện nay, các nước như Mỹ, Trung Quốc, Nhật Bản, Hàn Quốc hay Singapore là các nước đi đầu về thị giác máy tính và đang liên tục chạy đua trong việc chiếm ưu thế trong lĩnh vực trí tuệ nhân tạo. Tiêu biểu nhất là Trung Quốc, khi là nước ứng dụng tốt nhất thị giác máy tính khi đưa chúng vào phục vụ đời sống. Tại Trung Quốc, taxi tự hành đang nổi lên như một xu hướng, phục vụ con người một cách an toàn và mang lại trải nghiệm thân thiện. Ngoài ra, xử lý camera tại biên cũng đang là xu hướng của toàn thế giới, đây là phương pháp giúp thực hiện quản lý qua camera một cách nhanh chóng, tiện lợi, thời gian thực và đặc biệt đảm bảo tính cá nhân hóa. Và đặc biệt, không thể không kể đến sự trợ giúp đắc lực của thị giác máy tính cho việc mô hình hóa 3D, điều này được thể hiện thông qua việc tái tạo chính xác các đối tượng trong

thế giới thực. Hay trong thiết kế game, kiến trúc thì thị giác máy tính đều hỗ trợ đắc lực trong việc cung cấp mô phỏng khán quan các đối tượng, không gian một cách phù hợp, từ đó trở thành công cụ đắc lực trong việc tăng trải nghiệm khách hàng.

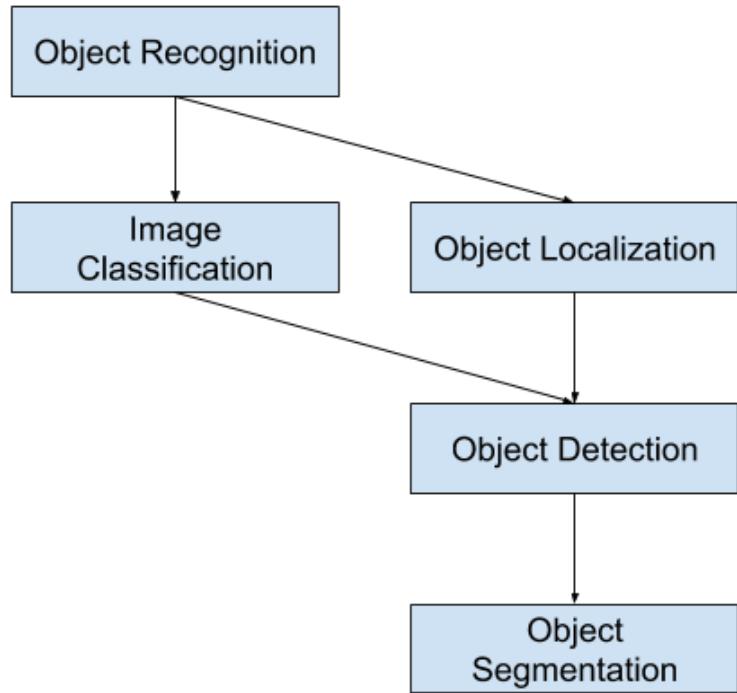
Tại Việt Nam, thị giác máy tính được ứng dụng rộng rãi nhất trong lĩnh vực sản xuất tự động hóa, y tế và logistic. Điều này thể hiện rõ nhất qua các dự án ứng dụng của VinBrain trong lĩnh vực y tế, chăm sóc sức khỏe hay nhà máy logistic hiện đại nhất Việt Nam của Viettel. Qua đó có thể thấy thị giác máy tính đang thực sự bùng nổ tại nước ta trong năm vừa qua như thế nào.

Tuy nhiên, hiện tại ngành công nghiệp thị giác máy tính đang có dấu hiệu đi chậm lại khi không có các nghiên cứu cách mạng giống như giai đoạn trước đó. Đa phần là hướng tới tối ưu hóa mô hình cũ hoặc tạo ra các thuật toán phù hợp cho việc tiền xử lý và sử dụng dữ liệu hợp lý. Điều này diễn ra là do sự thiếu hụt trầm trọng nguồn dữ liệu, các vấn đề đạo đức pháp lí đối với ngành hay chi phí cho việc huấn luyện là quá lớn. Ngoài ra, số lượng tài nguyên vi xử lý còn thiếu hụt cũng là một vấn đề cự kì nghiêm trọng.

Từ đó, một hướng đi mới đã được tạo ra đó là việc tích hợp thị giác máy tính vào các dạng mô hình trí tuệ nhân tạo khác. Tiêu biểu nhất là Midjourney, đây là ứng dụng bắt đầu cho sự bùng nổ cho con sốt Generative AI hay AI tạo sinh. Dưới góc nhìn của thị giác máy tính, ta thấy được tiềm năng to lớn trong việc sử dụng các kỹ thuật xử lý ảnh hay sử dụng các mô hình thị giác máy tính cho các tác vụ phục vụ cho trí tuệ nhân tạo tạo sinh. Hay việc ChatGPT hiện tại có thể sử dụng đầu vào là dữ liệu ảnh phục vụ phân tích, tạo sinh theo yêu cầu của người dùng. Những tiện ích đều đang giúp tối ưu hóa các công việc của con người hàng ngày, mang đến trải nghiệm thân thiện và hỗ trợ tối đa nhu cầu tìm hiểu và sáng tạo của con người.

1.1.3. Các tác vụ hiện nay

Hiện nay thị giác máy tính đang có 5 tác vụ phổ biến.



Hình 1.1. Các tác vụ của thị giác máy tính

Dựa vào sơ đồ trên ta có thể thấy sự tương quan giữa các tác vụ hiện có của thị giác máy tính. Vậy ta sẽ mô tả mục đích và ứng dụng của chúng trong đời sống như sau:

- **Image Classification:** phân loại hình ảnh phục vụ cho việc dự đoán nhãn của hình ảnh. Điều này phục vụ cho việc dự đoán các đối tượng thông qua dữ liệu ảnh được cung cấp.
 - + Đầu vào: Một hình ảnh với chỉ một đối tượng.
 - + Đầu ra: Nhãn tương ứng với đối tượng trong ảnh.
- **Object Localization:** Định vị đối tượng giúp xác định vị trí hiện diện của các đối tượng trong ảnh và cho biết vị trí của chúng bằng bounding box.
 - + Đầu vào: Một ảnh có một hoặc nhiều đối tượng.
 - + Đầu ra : Một hoặc nhiều bounding box được xác định bởi tọa độ tâm, chiều rộng và chiều cao.
- **Object Detection:** Phát hiện đối tượng sẽ thực hiện cả nhiệm vụ của Image Classification và Object Localization bao gồm thực hiện xác định vị trí các đối tượng bằng bounding box và gán nhãn cho chúng

+ Đầu vào: Một hình ảnh có một hoặc nhiều đối tượng, chẳng hạn như một bức ảnh.

+ Đầu ra: Một hoặc nhiều bounding box và nhãn cho mỗi bounding box.

- **Object Segmentation:** Phân vùng đối tượng thực hiện nhận dạng các đối tượng bằng cách làm nổi bật các pixel thay vì bounding box và thực hiện image captioning bằng cách kết hợp mô hình CNN và LSTM để đưa ra lý giải về hành động của đối tượng và nội dung của bức ảnh.

- **Object Recognition:** Nhận dạng đối tượng bao gồm tất cả các tác vụ đã nêu trên, thay vì chỉ sử dụng cho mục đích nhận diện thì Object Recognition còn dạy cho máy tính hiểu và học được nội dung của ảnh.

Chúng ta có thể hiểu Object Recognition và Object Detection tương tự nhau một cách tương đối. Tuy nhiên điểm khác biệt chính nằm ở đầu ra khi mà Object Detection cung cấp vị trí của đối tượng bằng bounding box trong khi Object Recognition xác định được chúng mà không cần định vị dựa trên Object Localization. Mặt khác, Object Detection có thể cung cấp đa dạng thông tin đối tượng trong bounding box còn Object Recognition tập trung vào sự hiểu biết của máy tính về đối tượng và các thông tin hữu ích hơn.

1.2. Tổng quan về phát hiện đối tượng

1.2.1. Định nghĩa

Phát hiện đối tượng hay Object Detection là tác vụ nằm trong lĩnh vực thị giác máy tính. Như đã nói ở trên, Object Detection thực hiện 2 nhiệm vụ chính là xác định đối tượng bằng bounding box và gán nhãn cho chúng.

Khác với các tác vụ khác trong thị giác máy tính, Object Detection thực hiện đánh giá và điều chỉnh mô hình dựa trên sai số giữa nhãn dự báo và sai số khung hình dự báo so với thực tế.

Dựa trên lịch sử hình thành, phát triển và đặc điểm cấu trúc của các thuật toán object detection bao gồm 2 nhóm chính:

- **Họ các mô hình R-CNN (Region-Based Convolutional Neural Networks):** giải quyết các nhiệm vụ định vị vật thể và nhận diện vật thể.

- **Hệ các mô hình YOLO (You Only Look Once):** là một nhóm kỹ thuật thứ hai để nhận dạng đối tượng được thiết kế để nhận diện vật thể thời gian thực.

1.2.2. Ứng dụng

Trong cuộc sống, nhận diện đối tượng được ứng dụng vào rất nhiều các mảng, ngành nghề khác nhau bao gồm:

- **An ninh và giám sát:** Giúp phát hiện và theo dõi các đối tượng như người, xe cộ, nhằm tăng cường an ninh trong hệ thống camera.
- **Giao thông thông minh:** Phân tích lưu lượng giao thông, nhận diện biển báo và phương tiện để hỗ trợ hệ thống điều khiển giao thông.
- **Chăm sóc sức khỏe:** Sử dụng trong chẩn đoán y tế, ví dụ như phát hiện các bất thường trong hình ảnh X-quang hoặc MRI.
- **Thương mại điện tử:** Tự động phân loại sản phẩm, tìm kiếm hình ảnh và gợi ý sản phẩm dựa trên nội dung hình ảnh.
- **Nông nghiệp thông minh:** Phát hiện sâu bệnh, đánh giá tình trạng cây trồng từ ảnh chụp từ drone hoặc các thiết bị cầm tay.
- **Robot và tự động hóa:** Giúp robot xác định và tương tác với các vật thể trong môi trường làm việc của chúng.

1.2.3. Khó khăn và thách thức

Đối với con người, việc nhận dạng trong ảnh không phải là việc khó khăn, nhưng đối với một hệ thống nhân tạo thì việc nhận dạng đòi hỏi phải giải quyết rất nhiều vấn đề như:

- **Sự xuất hiện hoặc thiếu một số thành phần:** Các thành phần miêu tả đối tượng có thể xuất hiện hoặc không trong hình ảnh làm cho bài toán nhận dạng trở nên khó khăn hơn nhiều.
- **Tư thế, góc chụp:** Ảnh chụp có thể thay đổi rất nhiều khi thay đổi góc chụp giữa camera với đối tượng. Chẳng hạn như chụp thẳng, chụp xéo...
- **Sự biến dạng của đối tượng:** Biến dạng của đối tượng cũng có thể làm ảnh hưởng đến các thông số của đối tượng đó.

- **Sự che khuất:** Đối tượng có thể bị che khuất bởi các đối tượng khác.
- **Sự phức tạp của hình nền:** Hình nền phức tạp cũng sẽ khiến cho việc nhận dạng trở nên khó khăn.
 - **Điều kiện của ảnh:** Ảnh chụp trong các điều kiện khác nhau về ánh sáng, camera,... ảnh hưởng rất nhiều đến chất lượng của ảnh

1.3. Bài toán phát hiện bệnh ở mèo

1.3.1. Giới thiệu chung

Thế giới đang ngày càng trở nên văn minh và phát triển hơn, điều đó được biểu hiện qua việc con người càng ngày càng gần gũi với thiên nhiên và ngay cả những loài động vật xung quanh. Đặc biệt là giới trẻ đang ngày càng có xu hướng nuôi thú cưng bạn hay thậm chí chỉ là một thú vui để giải trí. Điều đó có nghĩa rằng, nhu cầu chăm sóc và quản lý sức khỏe của thú cưng ngày càng trở nên thiết yếu và phổ biến, loài mèo cũng không ngoại lệ.

Như chúng ta thấy, loài mèo là một trong những loài thú cưng được yêu thích nhất trên thế giới. Những người sở hữu mèo thường sẵn sàng chi trả những khoản phí cho việc chăm sóc và nuôi dưỡng chúng. Ngoài ra còn có rất nhiều mô hình kinh doanh được mọc lên nhờ sự thu hút của loài mèo như những quán cà phê, khách sạn hay các khu du lịch sử dụng loài mèo làm yếu tố để thu hút khách hàng.

Từ những nhu cầu đó, kết hợp với sự bùng nổ của trí tuệ nhân tạo và sự phổ biến rộng rãi của các mô hình CNN hay cụ thể hơn là các mô hình phục vụ cho nhận diện đối tượng trong đời sống, việc nghiên cứu và phát triển một sản phẩm liên quan đến chăm sóc loài mèo là rất phù hợp với bối cảnh như vậy. Đề tài này được tạo ra để phục vụ nhu cầu trên.

Các mô hình CNN đã trở nên ngày càng phổ biến và cực kì dễ áp dụng, tuy nhiên để có thể dự đoán chính xác thì cần có các kỹ thuật, kinh nghiệm phù hợp. Đặc biệt là việc xử lý dữ liệu một cách tốt nhất, nghiên cứu và tìm hiểu các phương pháp tối ưu phù hợp. Vì vậy, cần có sự sáng tạo và phân tích một cách kỹ lưỡng để có thể tiếp cận và giải quyết bài toán toàn diện và hiệu quả.

1.3.2. Tình hình nghiên cứu trong và ngoài nước

Hiện tại, số lượng nghiên cứu trong ngành thú y là rất lớn và số lượng nghiên cứu ứng dụng các công cụ phục vụ tự động hóa phù hợp các nghiên cứu đó như AI cũng rất nhiều. Tuy nhiên hiện tại trên thế giới số lượng nghiên cứu ứng dụng AI vào việc chăm sóc sức khỏe thú cưng còn rất ít. Đặc biệt là loài mèo.

Hiện tại, trên thế giới có công ty Catsme thực hiện ứng dụng AI trong việc phát hiện các triệu chứng bệnh thông qua biểu cảm của mèo. Tuy nhiên thì đó vẫn là chưa đủ khi mà số lượng tìm kiếm được trên các diễn đàn báo uy tín vẫn còn rất ít và chưa đa dạng.

Tại Việt Nam, gần như không có các báo cáo về việc ứng dụng AI phục vụ cho loài mèo. Qua đó ta có thể thấy được tiềm năng nghiên cứu có thể khai phá với chủ đề này còn rất lớn.

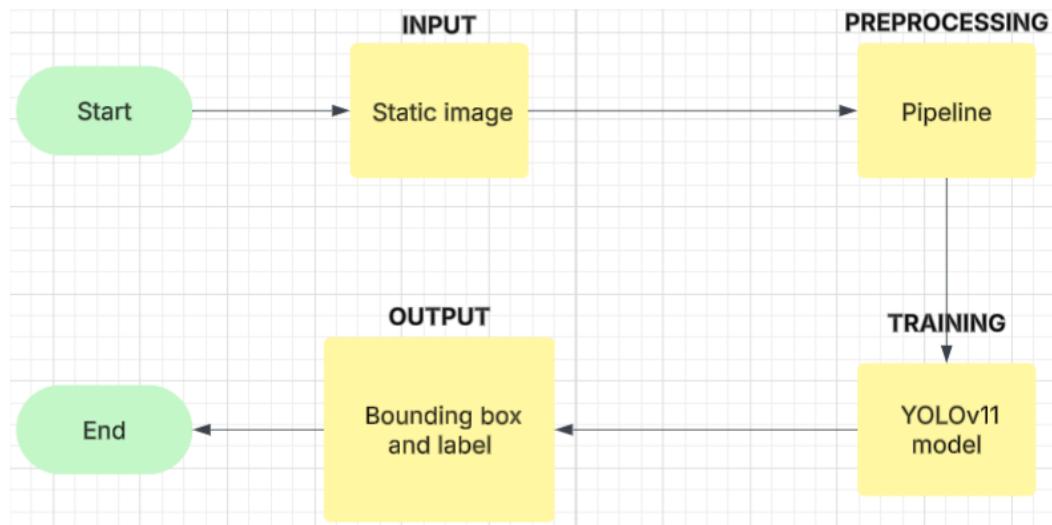
1.3.3. Phát biểu chi tiết bài toán

- **Đầu vào:** Đầu vào để thực hiện xây dựng bài toán và huấn luyện mô hình sẽ là các ảnh về các bệnh phổ biến trên loài mèo, cụ thể là các bệnh về da và dễ phát hiện. Các ảnh này được lấy tại các tổ chức y tế thú y hoặc các trang web của các tổ chức có uy tín trên internet. Ngoài ra, khi thực hiện thực nghiệm với thời gian thực, ngoài ảnh thì có thể là các video và hình ảnh được nhận trực tiếp từ camera.

- **Đầu ra:** Đầu ra cho bài toán này là vùng bệnh trên cơ thể của mèo và nhãn, tức là tên của bệnh đó trên vùng bệnh đã khoanh.

- **Các ràng buộc cụ thể:** Dữ liệu được sử dụng cho mục đích huấn luyện phải là ảnh tĩnh, được định dạng jpg. Ngoài ra, dữ liệu phải được tiền xử lý trước rồi mới thực hiện huấn luyện.

- **Minh họa:**



Hình 1.2. Minh họa bài toán

1.3.4. Mục tiêu

- Nghiên cứu và xây dựng một hệ thống cảnh báo dựa trên CNN hay cụ thể là mô hình YOLOv11 .
- Lựa chọn các thuật toán, các kỹ thuật phục vụ tiền xử lý một cách hiệu quả.
- Nghiên cứu và tối ưu mô hình được huấn luyện sao cho đạt kết quả đầu ra là chính xác nhất, qua đó đề xuất và cải tiến mô hình.
- Xây dựng hoàn thiện và triển một ứng dụng dựa trên các công nghệ phù hợp.

1.3.5. Ứng dụng thực tiễn

- **Nâng cao chất lượng chăm sóc sức khỏe cho thú cưng:** Phát hiện sớm các dấu hiệu bất thường trên cơ thể mèo giúp bác sĩ thú y can thiệp kịp thời, giảm thiểu tác động của bệnh lý và cải thiện hiệu quả điều trị. Điều này góp phần đảm bảo sức khỏe và tuổi thọ của vật nuôi.
- **Hỗ trợ chủ nuôi trong việc quản lý và theo dõi sức khỏe:** Với hệ thống tự động nhận diện bệnh dựa trên hình ảnh hoặc video, chủ nuôi có thể dễ dàng theo dõi tình trạng sức khỏe của mèo một cách định kỳ, từ đó đưa ra các biện pháp chăm sóc hợp lý mà không cần phải phụ thuộc hoàn toàn vào lịch trình khám bệnh truyền thống.

- **Giảm thiểu chi phí và thời gian chẩn đoán:** Việc ứng dụng công nghệ AI giúp tự động hóa quá trình phân tích hình ảnh, giảm bớt thời gian chẩn đoán và chi phí so với các phương pháp truyền thống, đồng thời giảm khả năng sai sót do con người gây ra.
- **Khuyến khích nghiên cứu và ứng dụng công nghệ trong lĩnh vực thú y:** Bài toán không chỉ mang tính ứng dụng cao trong thực tiễn mà còn mở ra cơ hội nghiên cứu, phát triển các giải pháp AI tiên tiến trong lĩnh vực chăm sóc sức khỏe động vật, góp phần thúc đẩy sự đổi mới và chuyển giao công nghệ trong ngành thú y.

1.3.6. Cơ hội và thử thách

- **Đa dạng và phức tạp của dữ liệu hình ảnh:** Hình ảnh mèo có thể có chất lượng, góc chụp, điều kiện ánh sáng và phông nền khác nhau, gây khó khăn cho mô hình trong việc học được các đặc trưng tổng quát.
- **Yêu cầu về tài nguyên tính toán:** Huấn luyện và triển khai YOLOv11, đặc biệt trong môi trường thời gian thực, đòi hỏi GPU mạnh mẽ và hệ thống tính toán hiệu quả. Điều này có thể là thách thức đối với các hệ thống có nguồn lực hạn chế.
- **Thu thập và gán nhãn dữ liệu:** Việc thu thập một tập dữ liệu đa dạng, chất lượng cao và sau đó gán nhãn chính xác (ví dụ: tạo bounding boxes cho các dấu hiệu bệnh) thường tốn kém thời gian và công sức.
- **Khan hiếm nguồn dữ liệu:** Vì đây là đề tài còn rất mới nên vẫn chưa có nhiều nghiên cứu hay các nguồn dữ liệu phổ biến để thực hiện nên dữ liệu thu thập được rất ít.

CHƯƠNG 2. MỘT SỐ KỸ THUẬT GIẢI QUYẾT BÀI TOÁN

2.1. Phương hướng tiếp cận

Để có thể nhận diện bệnh trên loài mèo một cách hiệu quả và đặc biệt là ở thời gian thực thì đầu vào sẽ là các ảnh thông qua camera hoặc ảnh được tải lên. Để phù hợp với dữ liệu ảnh thì sử dụng mô hình dựa mạng nơ-ron tích chập (CNN) với khả năng trích chọn đặc trưng và học từ dữ liệu sẽ là hiệu quả nhất. Từ đó ta có quá trình thực hiện bài toán bao gồm các bước sau:

- **Thu thập và chuẩn bị dữ liệu:** Thu thập hình ảnh mèo từ nhiều nguồn (ví dụ: camera giám sát, thiết bị di động, bộ dữ liệu công khai) và gán nhãn chính xác các dấu hiệu bệnh lý. Quá trình này bao gồm việc sử dụng các công cụ annotation để tạo ra các bounding box cho từng dấu hiệu bất thường. Cụ thể ở đây là Roboflow.
- **Tiền xử lý và tăng cường dữ liệu:** Áp dụng các kỹ thuật tiền xử lý như resize, normalization và data augmentation (lật, xoay, thay đổi độ sáng, v.v.) nhằm làm phong phú tập dữ liệu và giảm hiện tượng overfitting.
- **Xây dựng và huấn luyện mô hình:** Cấu hình các siêu tham số quan trọng (learning rate, batch size, số epoch, ...) và áp dụng các kỹ thuật như regularization (dropout, weight decay) và early stopping để tối ưu hóa quá trình huấn luyện cho mô hình YOLOv11 đã được pre-trained với bộ dữ liệu COCO.
- **Đánh giá và hiệu chỉnh:** Sử dụng các chỉ số như mAP, precision, recall, F1-score để đánh giá hiệu năng của mô hình trên tập validation. Dựa vào kết quả đánh giá, tiếp tục hiệu chỉnh hyperparameter hoặc áp dụng thêm các kỹ thuật fine-tuning nếu cần.
- **Xây dựng ứng dụng:** Triển khai model được huấn luyện thông qua framework Stremlit.

Phương pháp tiếp cận này đảm bảo rằng từ việc thu thập dữ liệu đến triển khai mô hình, mỗi bước đều được tối ưu nhằm xây dựng một hệ thống nhận diện bệnh ở loài mèo hiệu quả, chính xác và ổn định.

2.2. Mạng nơ-ron tích chập

2.2.1. Tổng quan

Mạng nơ ron tích chập hay CNN (Convolution Neural Networks) là mạng nơ ron nhân tạo được xây dựng phục vụ cho các tác vụ học sâu (Deep Learning), được thiết kế đặc biệt cho dữ liệu có cấu trúc dạng lưới (grid-like topology) hay dạng bảng. Ví dụ như dữ liệu chuỗi thời gian (time-series) được xem như dữ liệu dạng lưới 1 chiều với các mẫu được lấy theo khoảng thời gian đều đặn và liên tục, hay dữ liệu dạng ảnh là dữ liệu dạng lưới 2 chiều bao gồm các điểm ảnh (pixel)...

Mạng nơ ron tích chập được xây dựng nhằm giải quyết các hạn chế của các mạng nơ ron nhân tạo truyền thống như MLP (Multi-layer Preceptron) khi làm việc với các dạng dữ liệu có cấu trúc không gian như hình ảnh. Các hạn chế đó bao gồm:

- **Số lượng tham số:** Các mạng nơ ron kết nối đầy đủ (full-connected) thực hiện biến đổi các loại dữ liệu không gian thành một vector mà không qua xử lý, điều này khiến số lượng tham số trên các mô hình mạng nơ ron tương ứng với số lượng pixel trên ảnh là rất lớn. Thay vào đó, mạng nơ ron tích chập thực hiện giảm số lượng tham số bằng cách sử dụng các bộ lọc (filters) và trích xuất các đặc trưng cần thiết.
- **Khai thác cấu trúc không gian:** Các mạng nơ ron kết nối đầy đủ không khai thác không gian từ dữ liệu. Để giải quyết vấn đề đó, mạng nơ ron duy trì các cấu trúc dữ liệu này và thực hiện trích xuất các đặc trưng cục bộ như cạnh, góc, các hình khối. Qua đó dần xây dựng thành đối tượng phục vụ cho các tác vụ của thị giác máy tính.
- **Xử lý đặc trưng:** Thay vì trích xuất đặc trưng một cách thủ công bằng các kỹ thuật như HOG, mạng nơ ron tích chập thực hiện trích xuất và xử lý đặc trưng một cách tự động bằng cách cấu hình các bộ lọc và học chúng. Ngoài ra, mô hình mạng nơ ron tích chập học các đặc trưng một cách phân cấp, học từ cơ bản đến nâng cao, học từ những đặc trưng cơ bản nhất tới các chi tiết. Điều này được thể hiện thông qua lớp (layer).

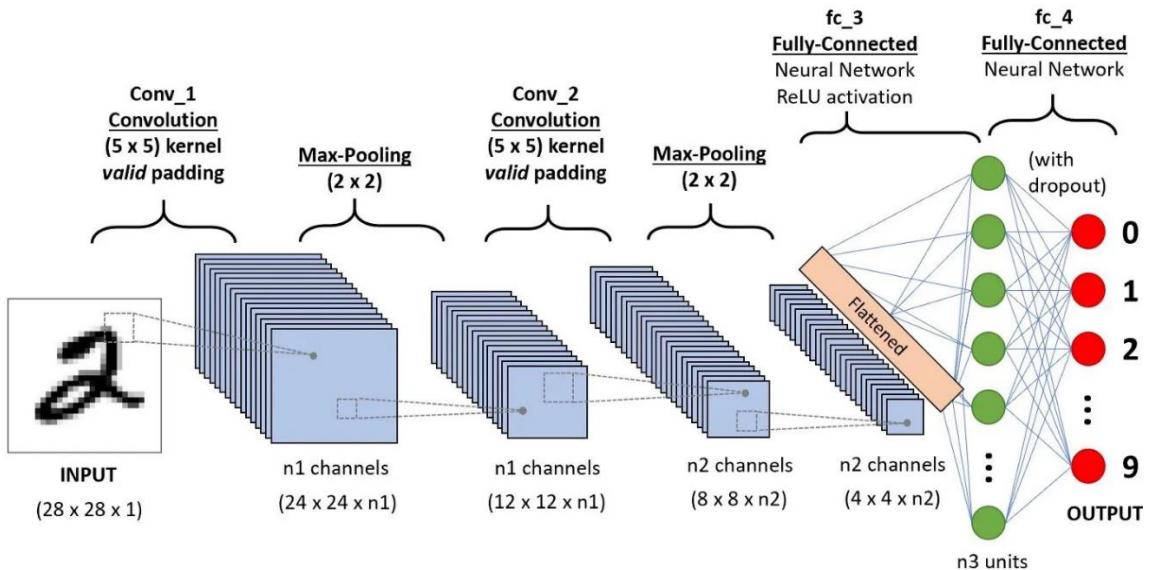
- **Xử lý vấn đề quá khóp:** Các mô hình mạng kết nối đầy đủ do số lượng tham số lớn sẽ bị vấn đề quá khóp (overfitting), điều này ảnh hưởng rất lớn tới kết quả đầu ra. Để xử lý vấn đề này, mạng nơ ron tích chập thực hiện việc ngắt kết nối giữa mô số nơ ron trong mạng một cách ngẫu nhiên (dropout) để thực hiện giảm tình trạng này.
- **Không ổn định với sự dịch chuyển:** Khi các đặc trưng trong ảnh bị dịch chuyển tức là bị lệch đi thì mạng nơ ron kết nối đầy đủ sẽ ảnh hưởng tới việc phát hiện đặc trưng trong ảnh. Mạng nơ ron sử dụng padding để giải quyết vấn đề này. Bằng cách thêm các giá trị pixel vào biên của ảnh để khi đổi tượng di chuyển hay hành động, bộ lọc sẽ vẫn có thể nhận diện các đặc trưng của đổi tượng ở vị trí mới mà không bị ảnh hưởng bởi sự thay đổi vị trí.
- **Khó mở rộng với dữ liệu ảnh có kích thước lớn:** Khi ảnh có kích thước càng lớn tức là số lượng pixel càng lớn, điều này sẽ làm tăng số lượng tham số của các mạng nơ ron kết nối đầy đủ. Ngược lại với điều đó, mạng nơ ron tích chập sử dụng các lớp kết nối cục bộ bao gồm lớp tích chập, lớp pooling và lớp kết nối đầy đủ với khả năng chia sẻ trọng số, giúp giảm đáng kể số tham số và xử lý hiệu quả hơn với ảnh lớn.

Mạng nơ ron tích chập đã và đang chiếm lĩnh vai trò cực kì quan trọng trong lĩnh vực thị giác máy tính, đặc biệt là khi con người ngày càng ứng dụng mạnh mẽ trí tuệ nhân tạo vào đời sống và sản xuất như hiện nay. Vì vậy, việc phát triển và ứng dụng mạng nơ ron tích chập là một trong những nhiệm vụ cơ bản và thiết yếu trong ngành công nghệ, đặc biệt là trí tuệ nhân tạo.

2.2.2. Cấu trúc

Kiến trúc truyền thống của mạng nơ ron tích chập bao gồm các lớp sau:

- Lớp tích chập
- Lớp pooling
- Lớp kết nối đầy đủ

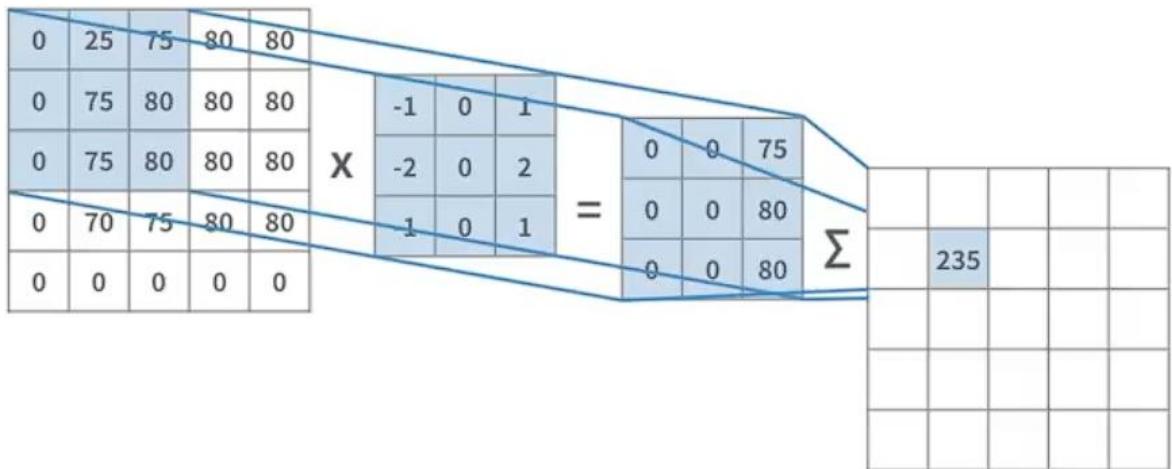


Hình 2.1. Kiến trúc CNN cơ bản

Lớp tích chập và lớp pooling được hiệu chỉnh theo các siêu tham số (hyperparameters)

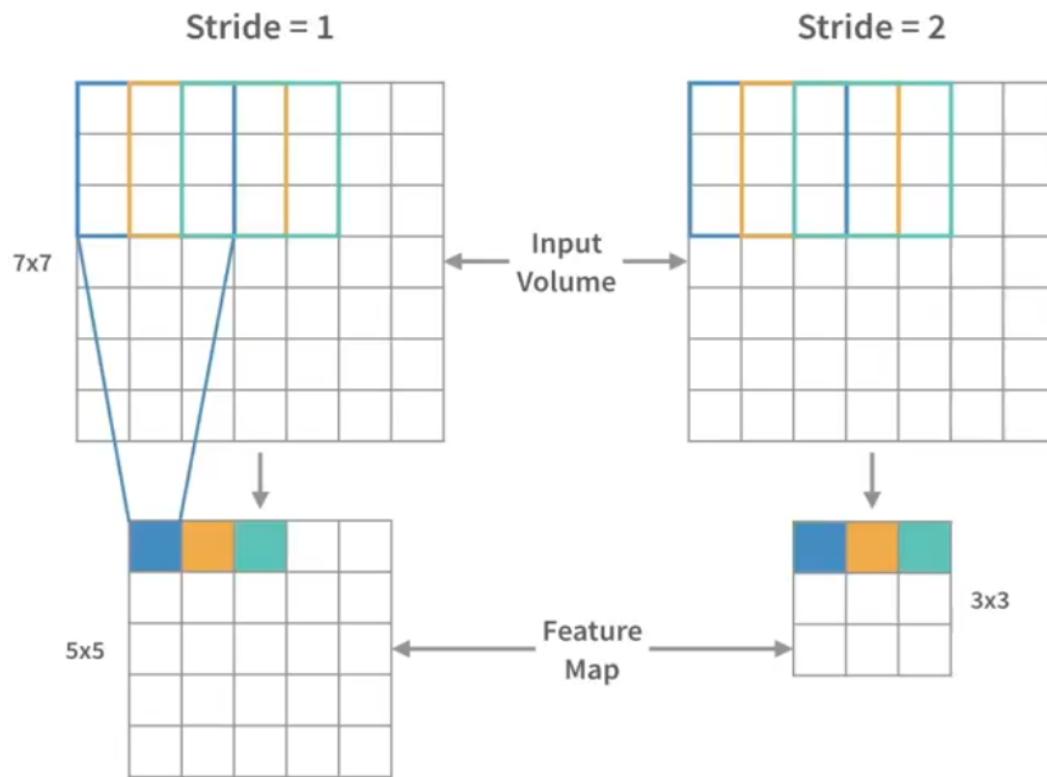
2.2.3. Phép tương quan chéo

Mạng nơ ron tích chập được xây dựng dựa trên phép tương quan chéo (cross correlation). Cụ thể phép này được thực hiện trên một mảng dữ liệu đầu vào 2 chiều và một mảng hạt nhân tương quan (kernel correlation) hay bộ lọc có kích thước cố định với điều kiện kích thước của mảng nhỏ hơn hoặc bằng kích thước của dữ liệu đầu vào. Phép này được thực hiện bằng cách di chuyển bộ lọc từ trái qua phải, từ trên xuống dưới dựa trên một độ trượt (stride) nhất định qua các vùng ảnh. Tại mỗi nơi mà bộ lọc đi qua sẽ thực được nhân theo từng phần tử tại vị trí tương ứng, rồi sau đó ta lấy tổng các phần tử trong mảng kết quả để có được một giá trị số vô hướng duy nhất. Giá trị này được ghi vào mảng đầu ra tại vị trí tương ứng. Toàn bộ quá trình này sinh ra một ma trận đặc trưng đầu ra (feature map), thể hiện mức độ trùng khớp giữa bộ lọc và các vùng của ảnh đầu vào. Sau đây là minh họa về phép tương quan chéo với đầu vào là một ma trận 5×5 và bộ lọc có kích thước 3×3 :



Hình 2.2. Minh họa phép tương quan chéo

Độ trượt hay Stride là số pixel mà bộ lọc di chuyển qua dữ liệu đầu vào, ở đây là ma trận. Một độ trượt có nghĩa là bộ lọc trượt từng pixel một, dẫn đến bản đồ đặc trưng đầu ra được tính toán dày đặc. Một bước tiến lớn hơn hai hoặc nhiều pixel bỏ qua, dẫn đến bản đồ đặc trưng đầu ra thưa thớt hơn và giảm kích thước không gian. Một bước tiến lớn hơn có thể làm tăng hiệu quả tính toán nhưng cũng có thể dẫn đến mất thông tin không gian.



Hình 2.3. Minh họa cách độ trượt hoạt động

2.2.4. Bộ lọc

Bộ lọc (Filter) hay Hạt nhân (Kernel) là một trong những thành phần cực kì quan trọng trong mạng nơ ron tích chập. Bộ lọc là một ma trận vuông nhỏ thực hiện các phép tương quan chéo trên dữ liệu đầu vào giúp trích xuất các đặc trưng cần thiết thông qua việc quét các vùng trên ma trận dữ liệu đầu vào. Điều này giúp giảm số lượng tham số và loại bỏ những đặc trưng không cần thiết. Các đặc trưng được trích xuất ra tạo thành bản đồ đặc trưng (feature map) trở thành đầu vào cho các lớp tiếp theo. Mỗi bộ lọc sẽ có các nhiệm vụ trích xuất từng đặc trưng khác nhau.

Có 6 loại bộ lọc bao gồm:

- **Bộ lọc phát hiện cạnh (Edge Detection Filter):** Bộ lọc này có nhiệm vụ làm nổi bật các cạnh của đối tượng trong ảnh bằng cách phát hiện những thay đổi mạnh trong cường độ điểm ảnh. Một số bộ lọc phổ biến bao gồm:
 - + **Bộ lọc Sobel:** Dùng để phát hiện các cạnh theo chiều ngang hoặc dọc bằng cách nhấn mạnh các vùng có sự thay đổi cường độ lớn.
 - + **Bộ lọc Prewitt:** Tương tự như Sobel nhưng sử dụng trọng số khác trong ma trận, cũng nhằm mục tiêu phát hiện cạnh.
 - + **Bộ lọc Laplacian:** Dựa trên đạo hàm bậc hai, giúp phát hiện các cạnh một cách tổng quát hơn, bao gồm cả cạnh chéo.
- **Bộ lọc làm sắc nét (Sharpening Filter):** Giúp tăng độ nét của hình ảnh bằng cách làm nổi bật các đường biên và chi tiết. Bộ lọc này nhấn mạnh các thành phần tần số cao, từ đó cải thiện độ rõ ràng của ảnh. Bộ lọc này khuếch đại sự chênh lệch giữa điểm ảnh trung tâm và các điểm lân cận, giúp làm rõ các chi tiết. Ví dụ:

$$\begin{bmatrix} 0, & -1, & 0 \\ -1, & 5, & -1 \\ 0, & -1, & 0 \end{bmatrix}$$

Hình 2.4. Ví dụ về bộ lọc làm sắc nét

- **Bộ lọc làm mờ (Smoothing/Blurring Filter):** Dùng để giảm nhiễu và làm mượt ảnh. Bộ lọc này thường được sử dụng ở giai đoạn tiền xử lý trước khi trích xuất đặc trưng. Một số loại phổ biến:
 - + **Bộ lọc Box Blur:** Tính trung bình giá trị các điểm ảnh lân cận với trọng số bằng nhau.
 - + **Bộ lọc Gaussian Blur:** Sử dụng hàm Gaussian để tính trung bình có trọng số, với điểm ảnh gần tâm có trọng số cao hơn, giúp làm mờ mượt mà hơn.
- **Bộ lọc tạo hiệu ứng nổi (Embossing Filter):** Tạo hiệu ứng 3D bằng cách làm nổi bật cạnh và thêm bóng đổ ở phía đối diện. Thường được sử dụng trong phân tích kết cấu hoặc tạo hiệu ứng thị giác. Bộ lọc này làm cho hình ảnh giống như được khắc nổi lên khỏi bề mặt. Ví dụ:

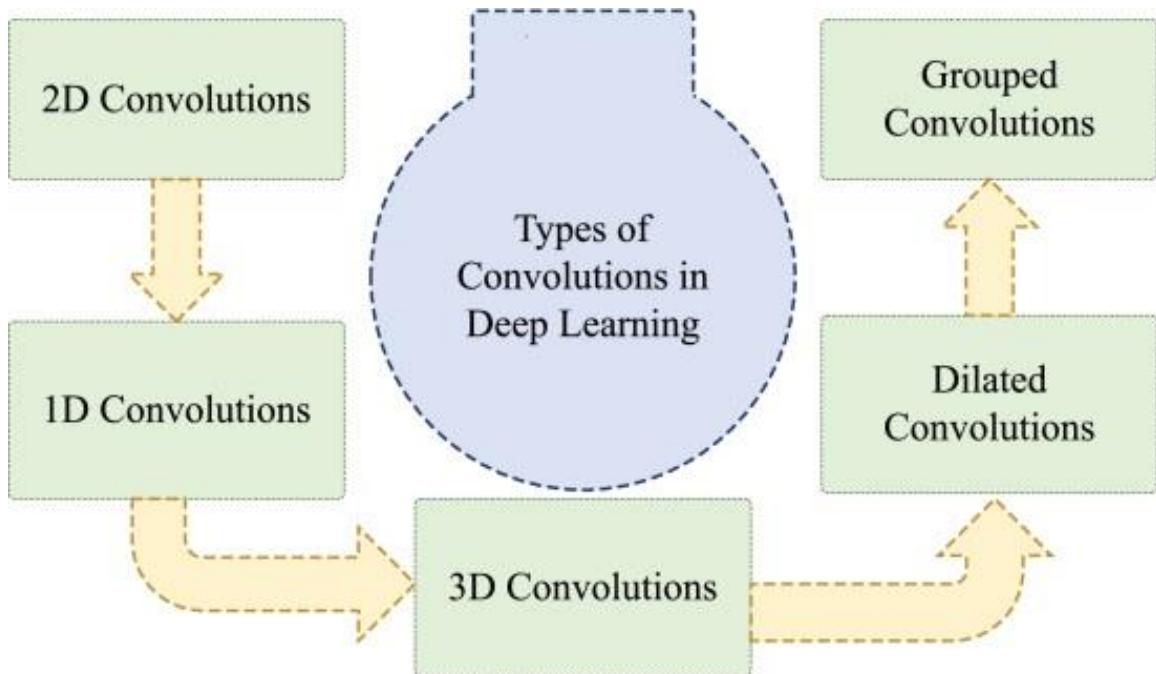
$$\begin{bmatrix} -2, & -1, & 0 \\ -1, & 1, & 1 \\ 0, & 1, & 2 \end{bmatrix}$$

Hình 2.5. Ví dụ về bộ lọc tạo hiệu ứng nổi

- **Bộ lọc tùy chỉnh (Custom Filter):** Trong các mô hình học sâu, đặc biệt là CNN, các bộ lọc không được xác định trước mà được học tự động từ dữ liệu trong quá trình huấn luyện. Các kernel này được cập nhật thông qua thuật toán lan truyền ngược (backpropagation) để trích xuất các đặc trưng tối ưu phục vụ cho nhiệm vụ cụ thể như nhận diện khuôn mặt, phân loại ảnh, phát hiện vật thể,...
- **Bộ lọc theo tần số (Frequency-Specific Filter):** Được thiết kế để nhắm đến các dải tần số cụ thể trong ảnh.
 - + **Bộ lọc thông cao (High-pass Filter):** Làm nổi bật các chi tiết sắc nét và đường biên.
 - + **Bộ lọc thông thấp (Low-pass Filter):** Loại bỏ các chi tiết nhỏ, giữ lại các vùng có sự thay đổi chậm trong ảnh, giúp làm mượt và giảm nhiễu.

2.2.5. Lớp tích chập

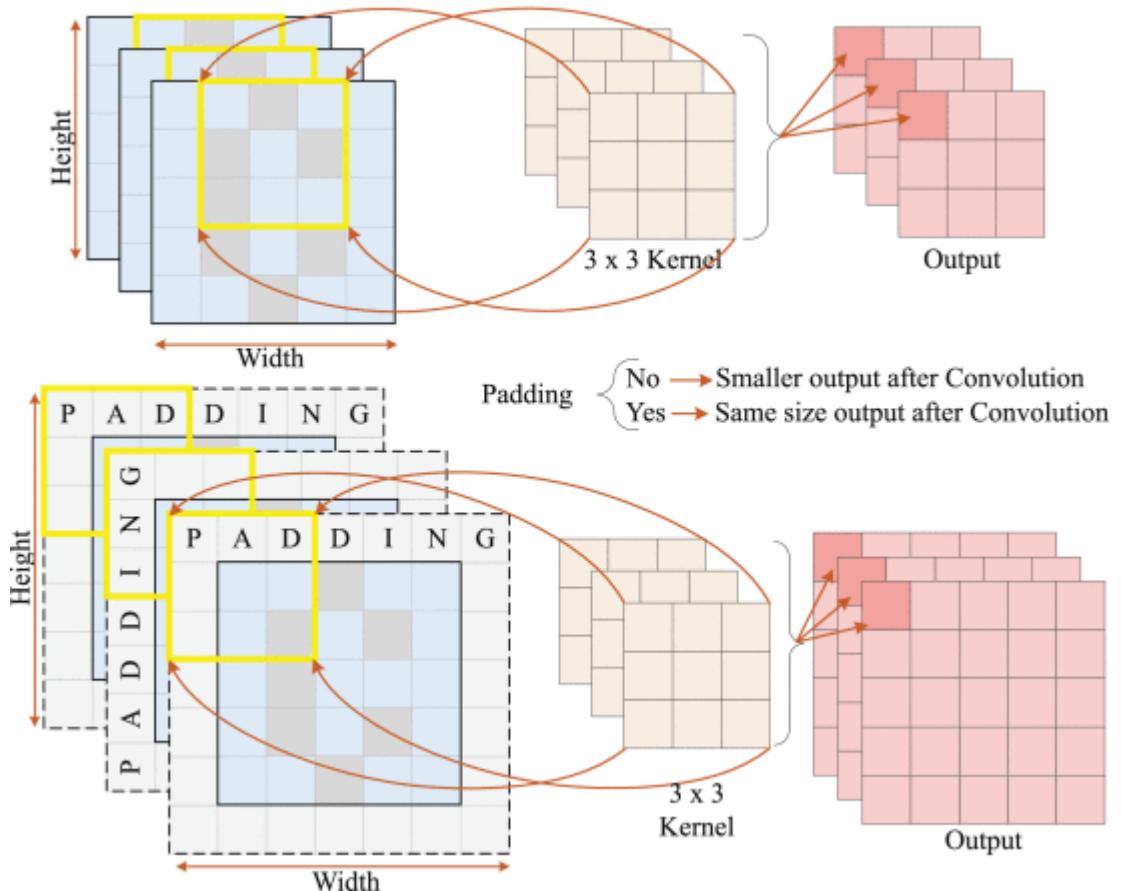
Lớp tích chập sử dụng phép tích chập hay chính xác hơn là phép tương quan chéo với dữ liệu đầu vào và bộ lọc cùng với sử dụng độ trượt (stride) để trích xuất đặc trưng và tạo thành ma trận đặc trưng. Có 5 kiểu tích chập phổ biến bao gồm:



Hình 2.6. Minh họa các kiểu tích chập phổ biến [17]

- **Tích chập 2 chiều (2D Convolutions):** là kỹ thuật sử dụng bộ lọc 2D trượt trên dữ liệu đầu vào để trích xuất các đặc trưng cục bộ. Đây là kỹ thuật cơ bản và phổ biến nhất. Tích chập 2 chiều có các ưu điểm sau:

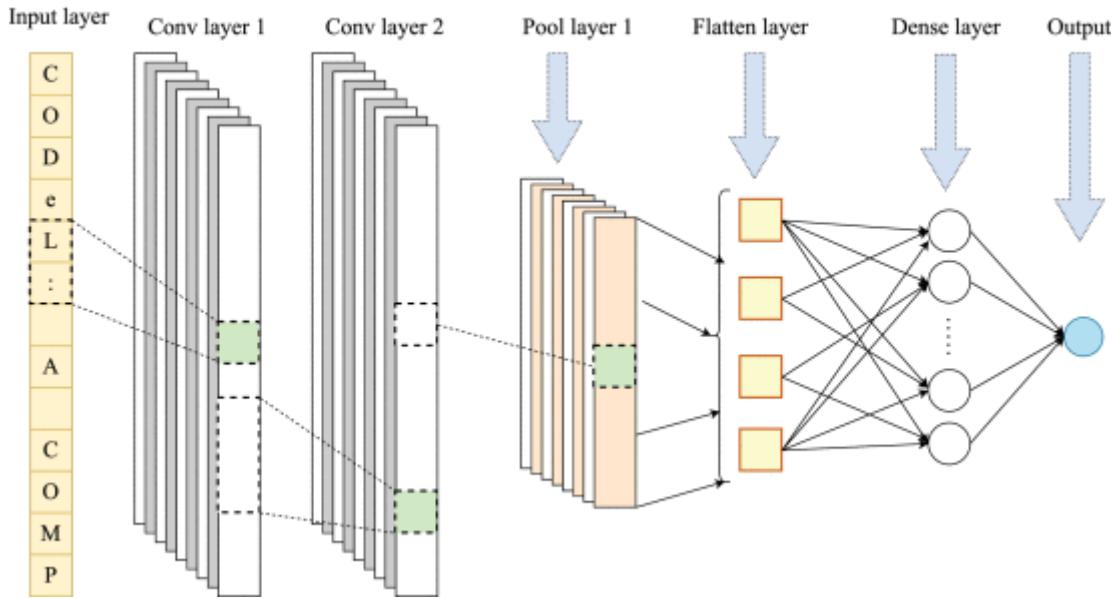
- + **Xử lý dữ liệu 2D:** Đặc biệt phù hợp với hình ảnh được biểu diễn dưới dạng lưới điểm ảnh.
- + **Trích xuất đặc trưng:** Học các mẫu quan trọng như cạnh, kết cấu, thành phần đối tượng.
- + **Ứng dụng đa dạng:** Áp dụng trong nhận diện hình ảnh, phát hiện đối tượng, phân đoạn ảnh, nhận diện khuôn mặt.
- + **Tăng hiệu quả phân tích:** Giúp mạng CNN hiểu mối quan hệ không gian và cấu trúc phân cấp trong hình ảnh.
- + **Mở rộng sang các lĩnh vực khác:** Ngoài thị giác máy tính, còn dùng trong xử lý tín hiệu và xử lý ngôn ngữ tự nhiên (NLP).



Hình 2.7. Minh họa tích chập 2 chiều [17]

- **Tích chập 1 chiều (1D Convolutions):** là kiểu tích chập được thiết kế chuyên biệt để xử lý dữ liệu tuần tự như chuỗi thời gian (time series), tín hiệu âm thanh và ngôn ngữ tự nhiên. Khác với tích chập 2 chiều, tích chập 1 chiều hoạt động trên một đường thẳng, cho phép chúng phát hiện ra các mẫu (patterns) phát triển theo thời gian. Tích chập 1 chiều chủ yếu được ứng dụng trong xử lý ngôn ngữ tự nhiên và xử lý tín hiệu âm thanh. Một số ứng dụng phổ biến bao gồm:

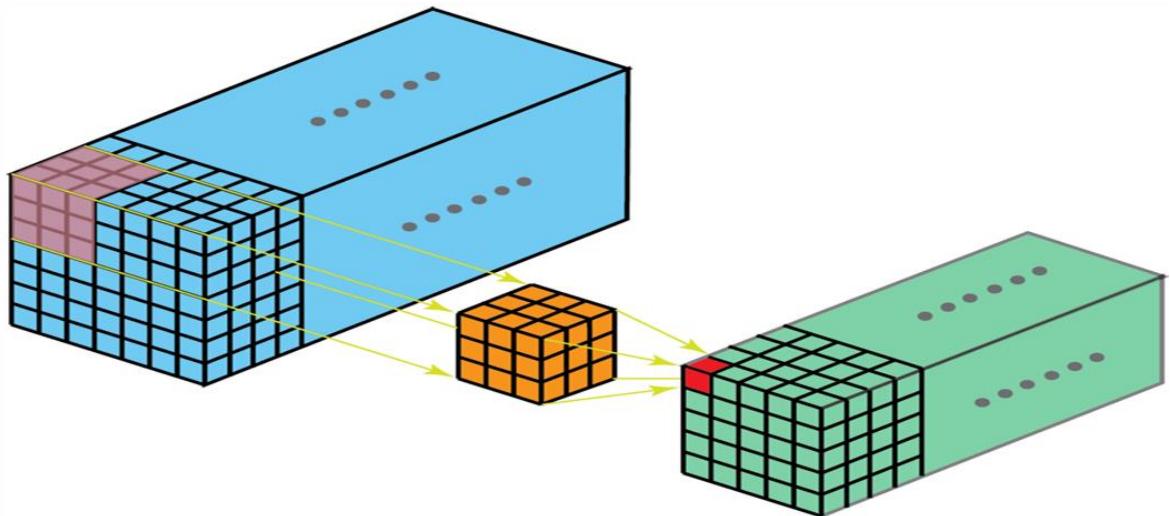
- + **Nhận diện người nói (Speaker Identification):** 1D convolutions phân tích mẫu giọng nói của từng cá nhân và học cách liên kết các mẫu đó với từng người cụ thể.
- + **Nhận diện cảm xúc (Emotion Recognition):** 1D convolutions phân tích sự thay đổi theo thời gian về cao độ (pitch), ngữ điệu (tone), và cường độ (intensity) của tín hiệu âm thanh để phân loại trạng thái cảm xúc (vui, buồn, tức giận...).



Hình 2.8. Minh họa tích chập 1 chiều [17]

– **Tích chập 3 chiều (3D Convolutions):** là kỹ thuật được thiết kế để xử lý dữ liệu thể tích như hình ảnh y tế 3D hoặc dữ liệu video. Các lớp tích chập này cho phép mô hình xử lý đồng thời các chiều không gian và thời gian, từ đó nắm bắt được các đặc trưng phức tạp trong cả ba chiều. Tích chập 3 chiều có một số ưu điểm sau:

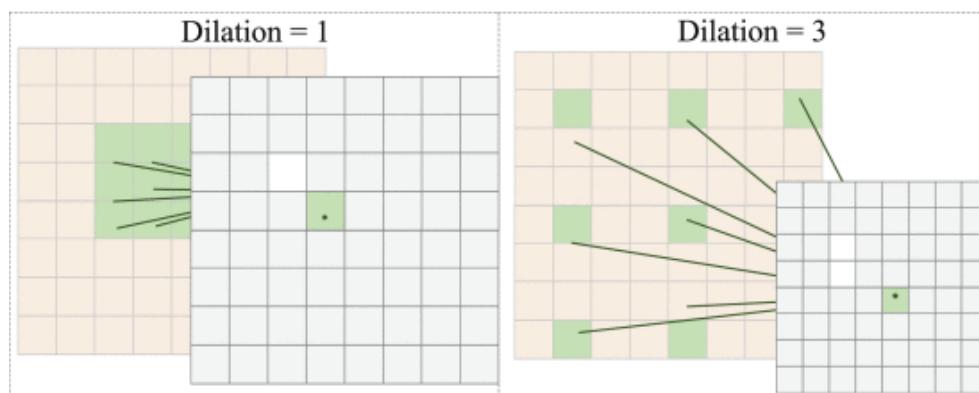
- + **Xử lý đồng thời không gian và thời gian:** Nắm bắt cả thông tin không gian 3 chiều và diễn biến theo thời gian.
- + **Hiểu rõ cấu trúc 3D tổng thể:** Phân tích toàn bộ thể tích thay vì lát cắt rời rạc.
- + **Cải thiện độ chính xác:** Tăng hiệu quả trong phân đoạn, nhận diện, chẩn đoán.
- + **Ứng dụng đa dạng:** Áp dụng cho ảnh y tế, video, dữ liệu đám mây điểm (point cloud).
- + **Bảo toàn mối quan hệ không gian:** Học được các đặc trưng theo cả ba trục (x, y, z).
- + **Hỗ trợ tối ưu hóa hiện đại:** Tương thích với sparse convolution, dilated convolution và multi-scale voxel.



Hình 2.9. Minh họa tích chập 3 chiều

- **Tích chập giãn (Dilated Convolutions hay Atrous Convolutions):** là một biến thể của tích chập truyền thống, trong đó các phần tử bộ lọc được xen kẽ bằng các khoảng trống nhằm mở rộng vùng cảm nhận (receptive field) mà không tăng số lượng tham số hay giảm độ phân giải. Kỹ thuật này có các ưu điểm bao gồm:

- + **Mở rộng vùng cảm nhận:** Giúp mô hình thu thập thông tin ngữ cảnh rộng hơn mà không cần tăng số lớp hoặc kích thước kernel.
- + **Hiệu quả trong các bài toán phân đoạn ngữ nghĩa và phát hiện đối tượng:** Bắt được thông tin tổng thể và chi tiết trong ảnh.
- + **Chống nhiễu và che khuất:** Giúp mô hình duy trì hiệu suất tốt ngay cả khi đầu vào bị che khuất hoặc nhiễu.

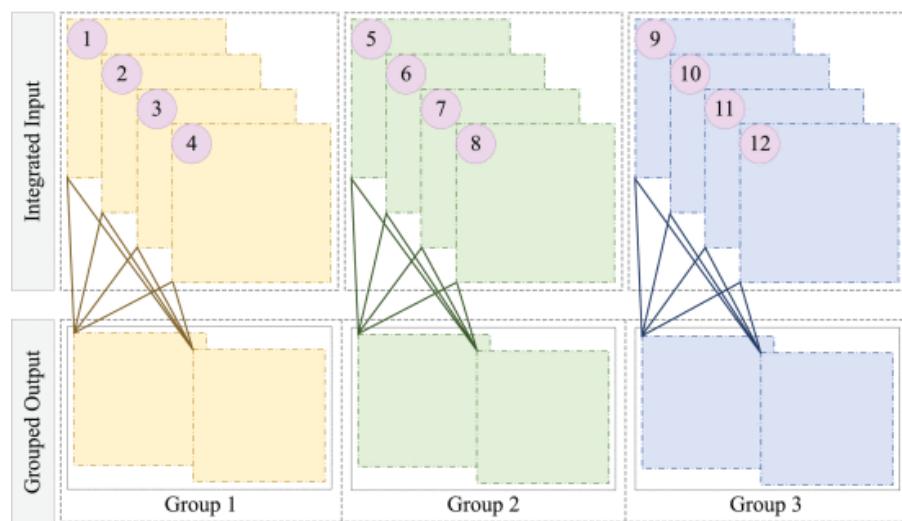


Hình 2.10. Minh họa tích chập 3 chiều [17]

- **Tích chập nhóm:** là kỹ thuật chia các kênh đầu vào và đầu ra của một lớp tích chập thành nhiều nhóm nhỏ. Trong mỗi nhóm, các phép tích chập riêng

biệt sẽ được thực hiện, sau đó kết quả từ các nhóm sẽ được ghép nối để tạo thành đầu ra cuối cùng. Kỹ thuật này có các ưu điểm bao gồm:

- + **Giảm chi phí tính toán:** Ít phép toán hơn so với tích chập toàn bộ.
- + **Tiết kiệm bộ nhớ:** Giảm dung lượng cần thiết để lưu trữ trọng số.
- + **Tăng khả năng song song hóa:** Các nhóm có thể tính toán đồng thời, tận dụng tối đa GPU hoặc hệ thống phân tán.
- + **Tăng khả năng mở rộng:** Dễ dàng huấn luyện mô hình trên tập dữ liệu lớn và trong các bài toán phức tạp.



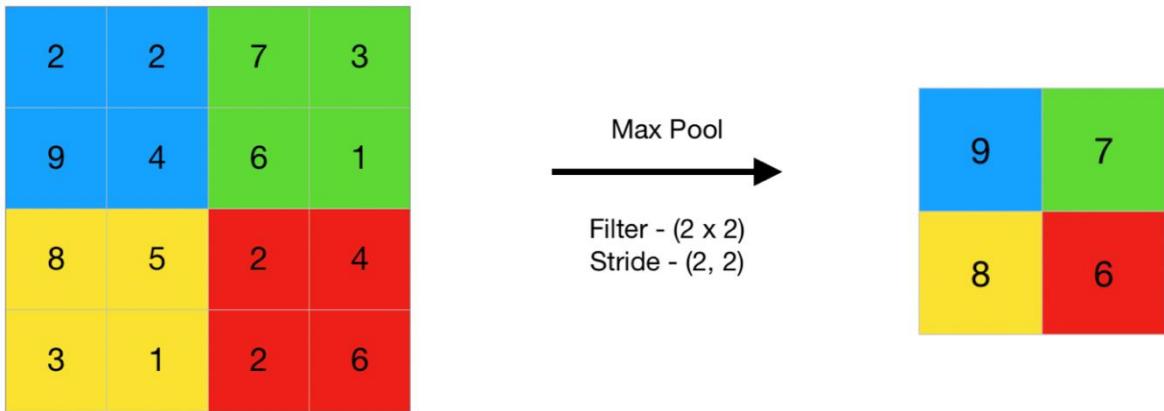
Hình 2.11. Minh họa tích chập nhóm [17]

2.2.6. Lớp pooling

Các lớp pooling được sử dụng để lấy mẫu lại (sub-sample) các bản đồ đặc trưng (feature maps) được tạo ra sau các phép toán tích chập. Cụ thể, pooling lấy các bản đồ đặc trưng có kích thước lớn và thu nhỏ chúng thành các bản đồ có kích thước nhỏ hơn. Trong quá trình thu nhỏ này, pooling giữ lại những đặc trưng (hoặc thông tin) nổi bật nhất trong mỗi vùng pooling.

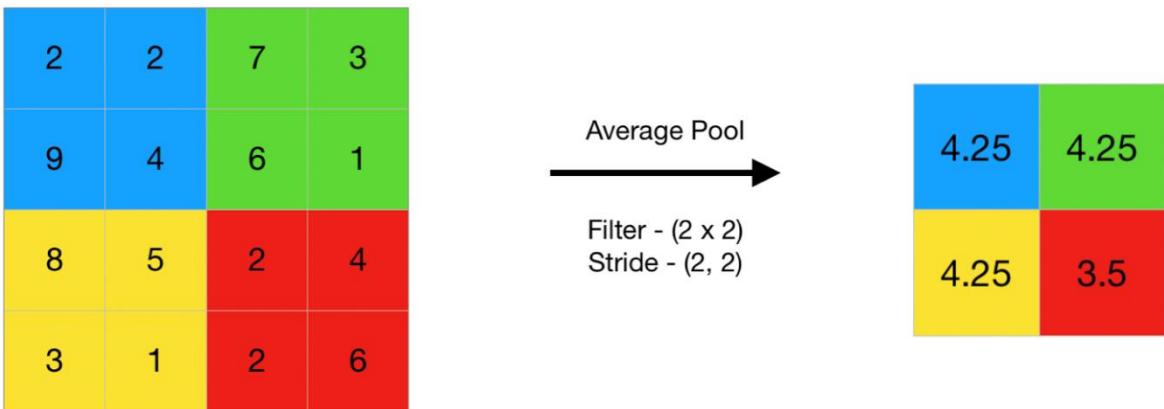
Phép toán pooling được thực hiện bằng cách xác định kích thước vùng pooling và stride của phép toán, tương tự như trong phép tích chập. Có nhiều kỹ thuật pooling khác nhau được sử dụng tại các lớp pooling khác nhau, nhưng có 3 loại phổ biến nhất bao gồm:

- **Max Pooling:** lấy ra phần tử lớn nhất từ vùng bén đồ đặc trưng được bộ lọc bao phủ. Kiểu pooling này sẽ bắt đặc trưng nổi bật và giúp mạng nhạy với sự hiện diện của đặc trưng mạnh.



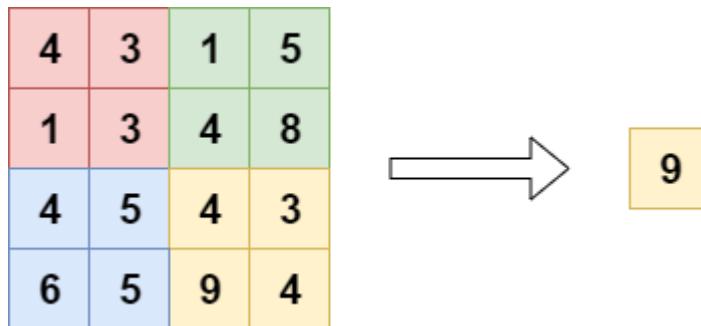
Hình 2.12. Minh họa kiểu Max Pooling

- **Average Pooling:** tính giá trị trung bình tại vùng mà bộ lọc bao phủ cho đến khi bộ lọc trượt hết bén đồ đặc trưng. Average pooling sẽ bắt thông tin tổng thể tốt hơn và ít nhạy cảm với nhiễu.

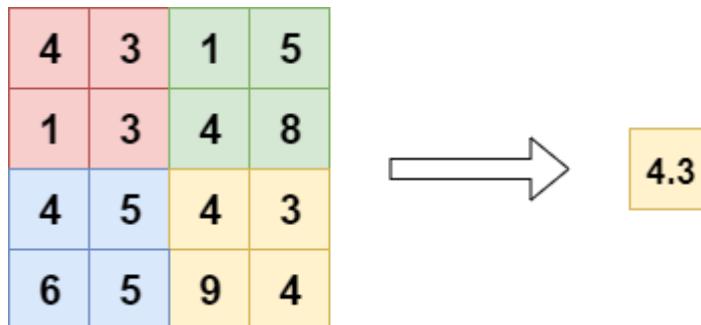


Hình 2.13. Minh họa kiểu Average Pooling

- **Global Pooling:** có thể chia thành 2 loại là Global Max Pooling và Global Average Pooling. Global Max Pooling lấy giá trị lớn nhất và Global Average Pooling tính giá trị trung bình tại toàn bén đồ đặc trưng. Điều này giúp giảm số lượng tham số lớn và tránh overfitting. Tuy nhiên có thể gây mất chi tiết quan trọng.



Hình 2.14. Minh họa kiểu Global Max Pooling



Hình 2.15. Minh họa kiểu Global Average Pooling

Tuy nhiên, hạn chế chính của lớp pooling là đôi khi nó làm giảm hiệu suất tổng thể của mạng CNN. Nguyên nhân là do lớp pooling chỉ giúp CNN xác định liệu một đặc trưng cụ thể có xuất hiện trong ảnh đầu vào hay không, mà không quan tâm đến vị trí chính xác của đặc trưng đó.

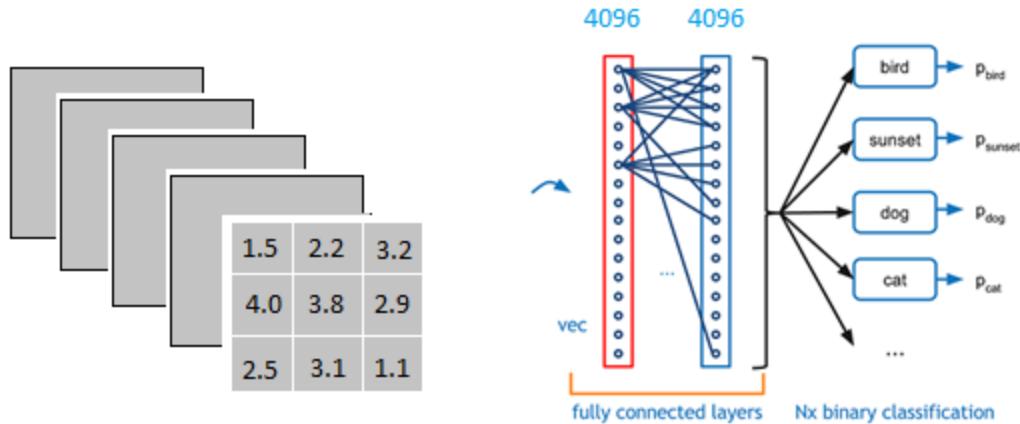
2.2.7. Lớp kết nối đầy đủ

Thông thường, phần cuối (hoặc các lớp cuối) của mỗi kiến trúc CNN (dùng cho phân loại) sẽ bao gồm các lớp fully-connected (FC), trong đó mỗi neuron trong một lớp sẽ kết nối với tất cả các neuron của lớp trước đó. Lớp fully-connected cuối cùng thường được sử dụng như lớp đầu ra (classifier) của toàn bộ kiến trúc CNN.

Các lớp fully-connected thực chất là một loại mạng nơ-ron nhân tạo (ANN) dạng feed-forward, và nó tuân theo nguyên lý của mạng perceptron nhiều lớp truyền thống (MLP).

Cụ thể, các lớp kết nối đầy đủ sẽ lấy đầu vào từ lớp tích chập hoặc pooling cuối cùng, vốn ở dạng một tập hợp các bản đồ đặc trưng (feature maps). Các bản đồ đặc trưng này sẽ được làm phẳng (flatten) thành một vector một

chiều, sau đó vector này được đưa vào lớp kết nối đầy đủ để tạo ra kết quả đầu ra cuối cùng của mô hình CNN.



Hình 2.16. Minh họa lớp kết nối đầy đủ

2.2.8. Các hàm kích hoạt

Hàm kích hoạt trong mạng nơ-ron có nhiệm vụ ánh xạ đầu vào thành đầu ra bằng cách xử lý tổng có trọng số của các đầu vào (kèm theo bias nếu có).

Nó quyết định nơ-ron có được kích hoạt hay không bằng cách tạo ra một đầu ra tương ứng.

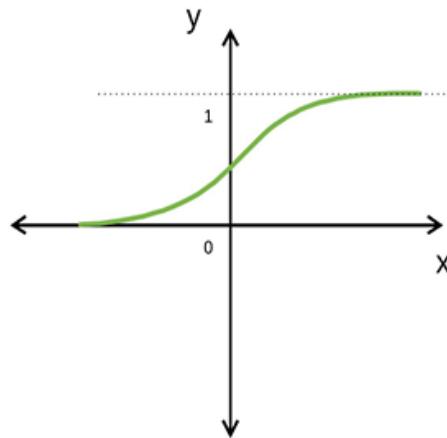
Trong kiến trúc CNN, hàm kích hoạt phi tuyến được sử dụng sau các lớp học được (như lớp tích chập và lớp fully-connected) để giúp mạng học được các mối quan hệ phức tạp giữa đầu vào và đầu ra.

Để hỗ trợ quá trình lan truyền ngược (backpropagation) khi huấn luyện, hàm kích hoạt bắt buộc phải khả vi (differentiable).

Một số hàm kích hoạt phổ biến nhất trong CNN bao gồm:

- **Sigmoid:** là hàm kích hoạt nhận đầu vào là các số thực bất kỳ và ánh xạ đầu ra vào khoảng giá trị $[0, 1]$. Nó có dạng đường cong hình chữ S (S-shaped curve), biểu diễn theo công thức:

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

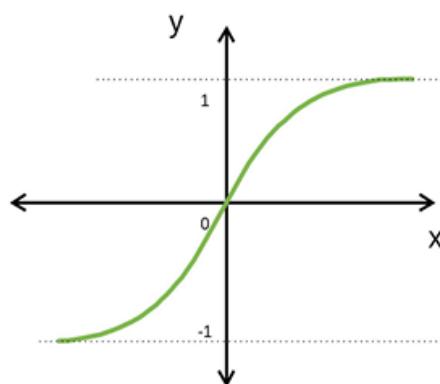


Hình 2.17. Minh họa hàm sigmoid

Hàm sigmoid thường được sử dụng cho các bài toán cần dự đoán xác suất, thường dùng ở lớp cuối của CNN khi bài toán yêu cầu phân loại 2 lớp.

- **Tanh:** là một hàm kích hoạt nhận đầu vào là các số thực bất kỳ và ánh xạ đầu ra vào khoảng giá trị $[-1, 1]$. Nó có dạng cong giống chữ S, nhưng khác sigmoid ở chỗ trung tâm của nó là 0 thay vì 0.5, biểu diễn công thức:

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

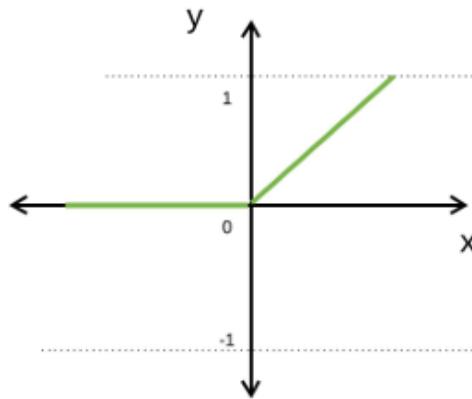


Hình 2.18. Minh họa hàm tanh

Hàm tanh thường được sử dụng để giúp đầu ra cân bằng quanh 0, điều này giúp việc học nhanh hơn, vì dữ liệu đầu vào cho các lớp tiếp theo sẽ có phân bố tốt hơn. Đặc biệt, thường được dùng ở các tầng giữa để cải thiện khả năng hội tụ của quá trình huấn luyện.

- **ReLU:** là một hàm kích hoạt đơn giản nhưng cực kỳ hiệu quả. Công thức của hàm:

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



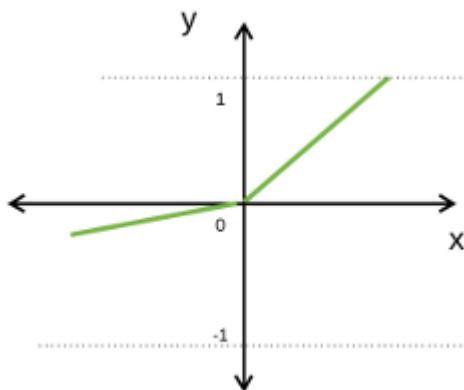
Hình 2.19. Minh họa hàm ReLU

ReLU thường được sử dụng để tăng tốc độ huấn luyện vì hội tụ nhanh hơn so với sigmoid hay tanh, nhờ vào việc đơn giản hóa phép tính và tránh hiện tượng gradient nhỏ. Ngoài ra, ReLU có khả năng giảm hiện tượng biến mất đạo hàm (vanishing gradient) tức là nguy cơ đạo hàm tiến về 0, giúp mạng sâu học tốt hơn.

- **Leaky ReLU:** là một biến thể của ReLU có công thức:

$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ mx, & x \leq 0 \end{cases}$$

Trong đó m là một số dương cực nhỏ nhằm khắc phục điểm yếu chết neuron khi $f(x) = 0$



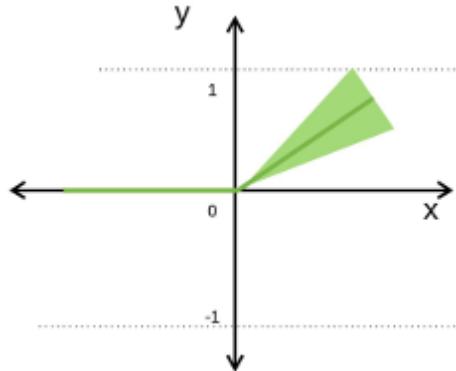
Hình 2.20. Minh họa hàm Leaky ReLU

Leaky ReLU giúp mạng học tốt hơn khi dữ liệu có nhiều giá trị âm, Leaky ReLU giữ được thông tin thay vì chặn hoàn toàn.

- **Noisy ReLU:** một biến thể của ReLU, trong đó ta thêm một thành phần nhiễu ngẫu nhiên (Y) vào đầu vào trước khi áp dụng hàm ReLU.

$$f(x) = \max(x + Y), \text{with } Y \sim N(0, \sigma(x))$$

Y thường là một số ngẫu nhiên lấy từ phân phối chuẩn (Gaussian distribution) hoặc một phân phối khác, giúp mạng học được những đặc trưng ổn định hơn thay vì chỉ ghi nhớ dữ liệu huấn luyện.



Hình 2.21. Minh họa hàm Noise ReLU

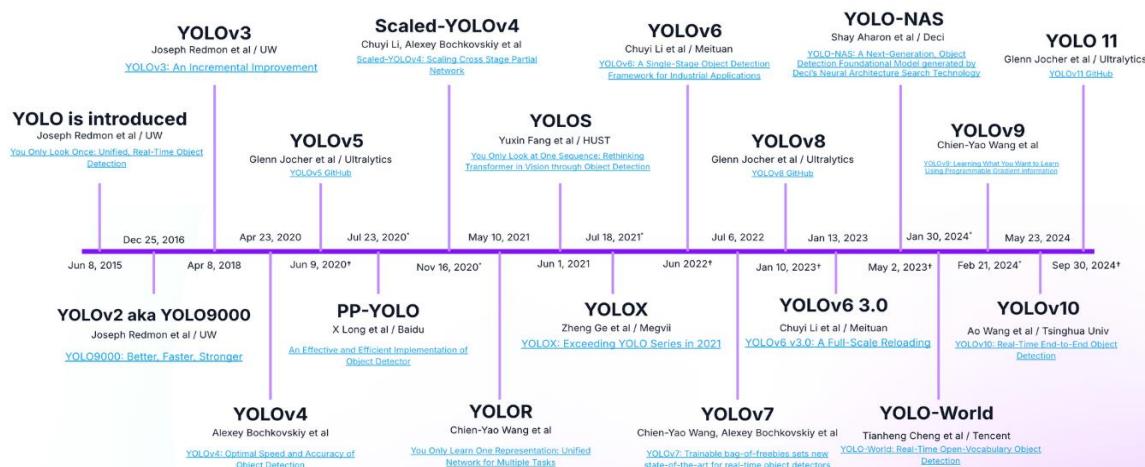
Noisy ReLU đặc biệt hữu ích trong việc giảm overfitting và giảm hiện tượng chết nơ ron.

2.3. Mô hình YOLOv11

2.3.1. Giới thiệu về thuật toán YOLO

YOLO hay You Only Look Once là một trong những họ thuật toán nổi tiếng trong lĩnh vực phát hiện đối tượng (Object Detection) bởi độ chính xác cũng như xử lý nhanh ở thời gian thực.

YOLO được giới thiệu từ năm 2015 bởi Joseph Redmon và cộng sự với tiêu đề “**You Only Look Once: Unified, Real-Time Object Detection**”. Kể từ đó đến nay YOLO đã có 12 phiên bản được ra mắt với các cải tiến khác nhau. Phiên bản mới nhất hiện tại là YOLOv12.

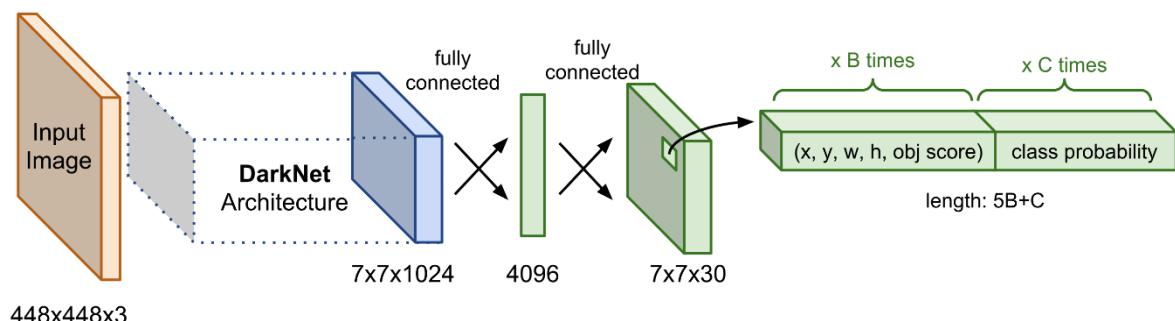


Hình 2.22. Lịch sử phát triển của YOLO

Thuật toán YOLO là một thuật toán single-shot object detection, tức là sử dụng một lần quét duy nhất đối với hình ảnh đầu vào để dự đoán sự hiện diện và vị trí của các đối tượng trong ảnh. Phương pháp này xử lý toàn bộ hình ảnh chỉ trong một lần, giúp nó có hiệu suất tính toán cao.

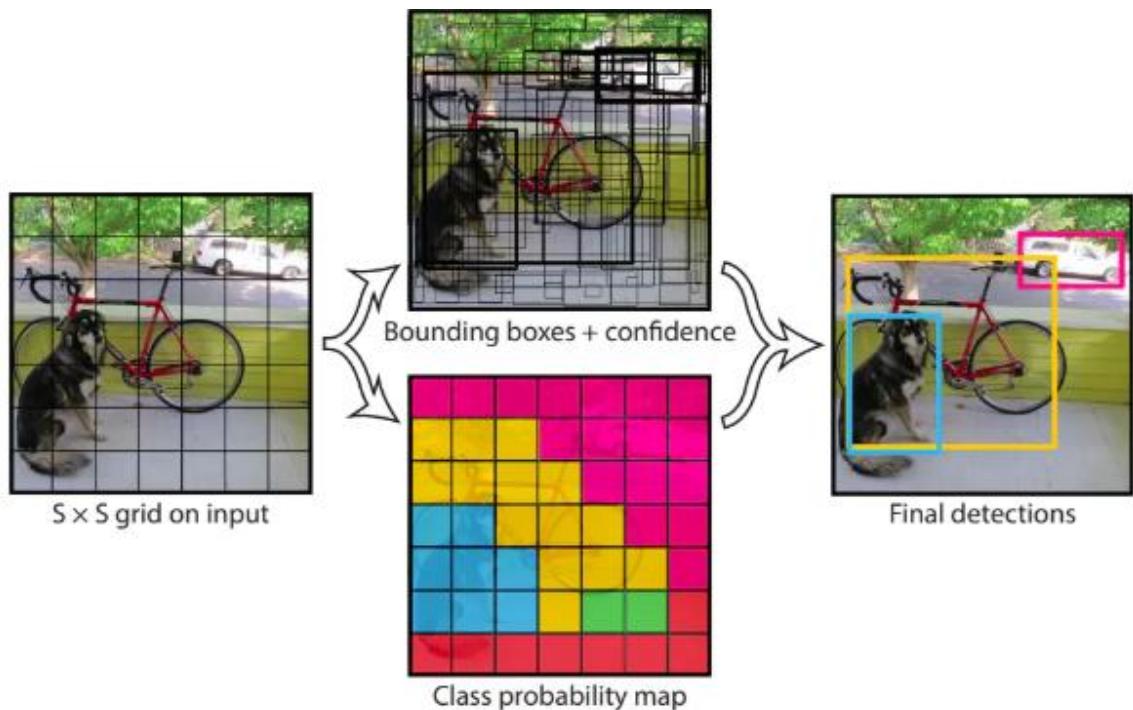
Kiến trúc ban đầu của YOLO bao gồm:

- Thành phần Darknet Architecture là các mạng tích chập được gọi là base network có tác dụng trích suất đặc trưng. Base network của YOLO sử dụng chủ yếu là các lớp tích chập và các lớp kết nối đầy đủ. Các kiến trúc YOLO cũng khá đa dạng và có thể tùy biến thành các phiên bản cho nhiều đầu vào có kích thước khác nhau.
- Output của base network là một bản đồ đặc trưng có kích thước $7 \times 7 \times 1024$ sẽ được sử dụng làm đầu vào cho các Extra layers có tác dụng dự đoán nhãn và tọa độ hộp giới hạn của vật thể.



Hình 2.23. Kiến trúc YOLO ban đầu

Từ input là ảnh đầu vào, qua một mạng gồm các lớp tích chập, pooling và lớp kết nối đầy đủ là có thể ra được output. Kiến trúc này có thể được tối ưu để chạy trên GPU với một lần truyền thăng (forward pass), và vì thế đạt được tốc độ rất cao.



Hình 2.24. Minh họa ý tưởng của thuật toán YOLO

Ý tưởng chính của YOLO là chia ảnh thành một lưới các ô (grid cell) với kích thước SxS (mặc định là 7x7). Với mỗi grid cell, mô hình sẽ đưa ra dự đoán cho B bounding box. Ứng với mỗi box trong B bounding box này sẽ là 5 tham số x, y, w, h, confidence, lần lượt là tọa độ tâm (x, y), chiều rộng w, chiều cao h và độ tự tin của dự đoán. Với grid cell trong lưới SxS kia, mô hình cũng dự đoán xác suất rơi vào mỗi class.

Độ tự tin của dự đoán ứng với mỗi bounding box được định nghĩa là $p(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$, trong đó $p(\text{Object})$ là xác suất có vật trong cell và $\text{IOU}_{\text{pred}}^{\text{truth}}$ là tỷ lệ giao nhau trên hợp (Intersection over Union) giữa vùng dự đoán (predicted box) và vùng thực tế (ground truth box).

Xác suất rơi vào mỗi class cho một grid cell được ký hiệu $p(\text{Class}_i | \text{Object})$. Các giá trị xác suất cho C class sẽ tạo ra C output cho mỗi grid cell. B bounding box của cùng một grid cell sẽ chia sẻ chung một tập các dự đoán về class của vật, đồng nghĩa với việc tất cả các bounding box trong cùng một grid cell sẽ chỉ có chung một class. Tổng số output của mô hình sẽ là $S \times S \times (5 * B + C)$ (dưới dạng ma trận).

Loss function: YOLO chia loss function thành 2 phần:

- **Localization loss** dùng để đo lường sai số của bounding box
- **Confidence loss** đo lường sai số của phân phối xác suất các classes

$$\begin{aligned}\mathcal{L}_{\text{loc}} &= \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \\ \mathcal{L}_{\text{cls}} &= \underbrace{\sum_{i=0}^{S^2} \sum_{j=0}^B (\mathbb{1}_{ij}^{\text{obj}} + \lambda_{\text{noobj}}(1 - \mathbb{1}_{ij}^{\text{obj}})) (C_{ij} - \hat{C}_{ij})^2}_{\text{cell contain object}} + \underbrace{\sum_{i=0}^{S^2} \sum_{c \in \mathcal{C}} \mathbb{1}_i^{\text{obj}} (p_i(c) - \hat{p}_i(c))^2}_{\text{probability distribution classes}} \\ \mathcal{L} &= \mathcal{L}_{\text{loc}} + \mathcal{L}_{\text{cls}}\end{aligned}$$

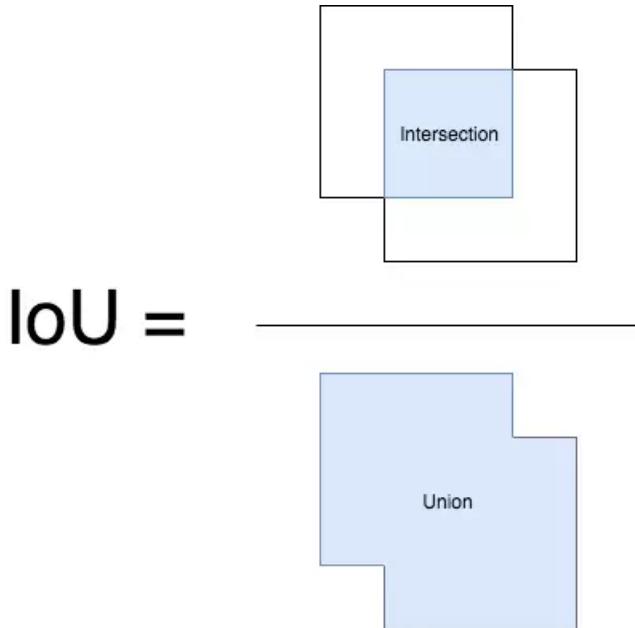
Hình 2.25. Công thức tính Loss function của YOLO

Trong đó:

- + $\mathbb{1}_{ij}^{\text{obj}}$ là một hàm chỉ thị có giá trị bằng 0 hoặc 1, bằng 1 nếu cell i chứa object, bằng 0 nếu không chứa object
- + $\mathbb{1}_i^{\text{obj}}$ xác định xem cell i có chứa vật thể hay không. Bằng 1 nếu chứa vật thể và 0 nếu không chứa.
- + $\mathbb{1}_i^{\text{noobj}}$ ngược lại với $\mathbb{1}_i^{\text{obj}}$
- + λ_{coord} và λ_{noobj} là hệ số điều chỉnh
- + C_{ij} là điểm tin cậy của bounding box i trong cell j = $p(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}}$
- + \hat{C}_{ij} là điểm tin cậy dự đoán của bounding box i trong cell j
- + C là số lượng class
- + p_{ij} là xác suất có vật thể trong cell i
- + \hat{p}_{ij} là xác suất dự đoán có vật thể trong cell i

Trong YOLO nếu chỉ dừng lại ở việc đặt các bounding box cho từng vật thể thì có thể sẽ xảy ra hiện tượng bị dự đoán nhiều bounding box cho cùng 1 vật thể. Kĩ thuật Non-Maximum Suppression (NMS) được sử dụng để giải quyết vấn đề trên. Ý tưởng sẽ là: Các box có confidence_score được xếp hạng từ cao xuống thấp. $[box_1, box_2, box_3, \dots, box_n]$ duyệt từ đầu danh sách, với từng box_i , loại đi các box_j có $\text{IOU}(box_i, box_j) > \text{threshold}$. Trong đó $j > i$.

threshold là một giá trị ngưỡng được lựa chọn sẵn. IOU là công thức tính độ overlap - giao thoa giữa hai bounding box.



Hình 2.26. Minh họa cách tính IOU

Nếu chỉ số này lớn hơn ngưỡng threshold thì điều đó chứng tỏ 2 bounding boxes đang overlap nhau rất cao. Ta sẽ xóa các bounding có xác xuất thấp hơn và giữ lại bounding box có xác xuất cao nhất. Cuối cùng, ta thu được một bounding box duy nhất cho một vật thể.

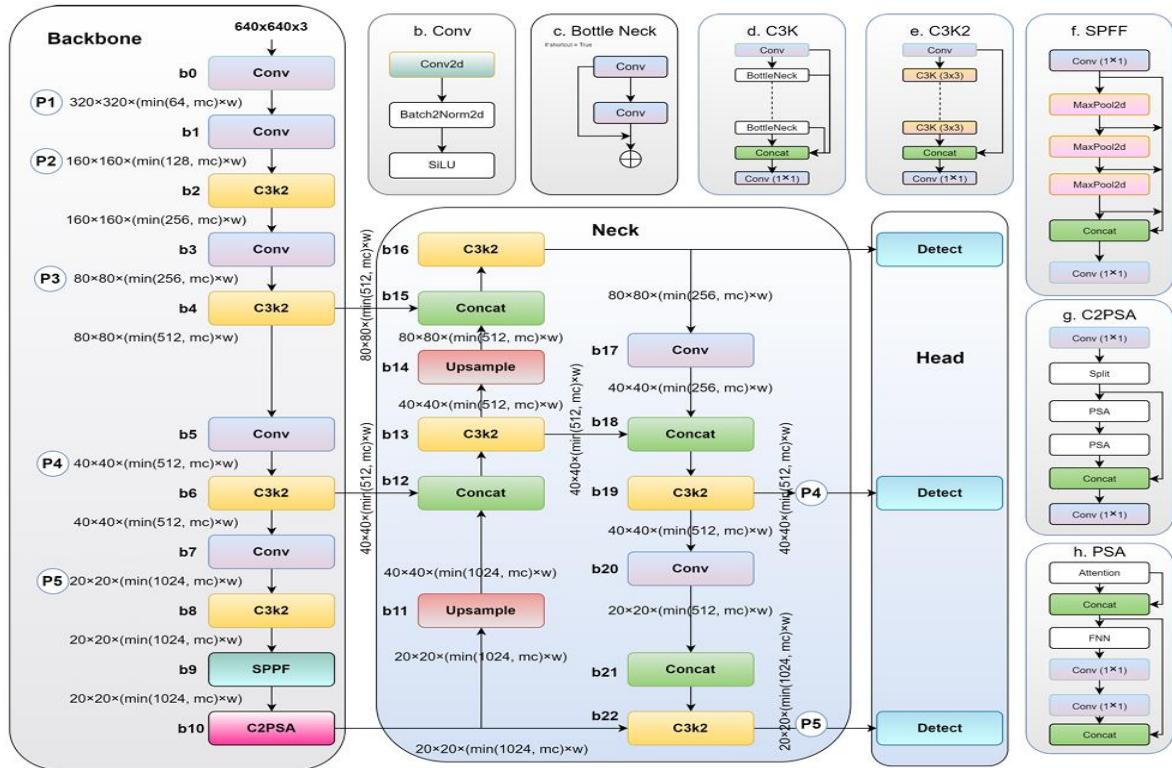
2.3.2. YOLOv11

So với kiến trúc ban đầu, YOLOv11 đã được cải tiến rất nhiều cả về mặt hiệu năng lẫn độ chính xác.

Sự khác biệt lớn nhất là kiến trúc hiện tại được chia thành 3 phần với nhiều lớp khác nhau bao gồm:

- **Lõi chính (Backbone):** là một thành phần quan trọng trong kiến trúc YOLO, chịu trách nhiệm trích xuất các đặc trưng từ hình ảnh đầu vào ở nhiều tỷ lệ khác nhau. Quá trình này bao gồm việc xếp chồng các lớp tích chập và các khối chuyên biệt để tạo ra các bản đồ đặc trưng ở các độ phân giải khác nhau.
- **Cỗ (Neck):** là phần kết hợp các đặc trưng ở các tỷ lệ khác nhau từ lõi chính và truyền chúng tới phần đầu (head) để dự đoán. Quá trình này thường bao gồm việc tăng tỷ lệ và kết hợp các bản đồ đặc trưng từ các mức độ khác nhau, giúp mô hình thu thập thông tin đa tỷ lệ hiệu quả.

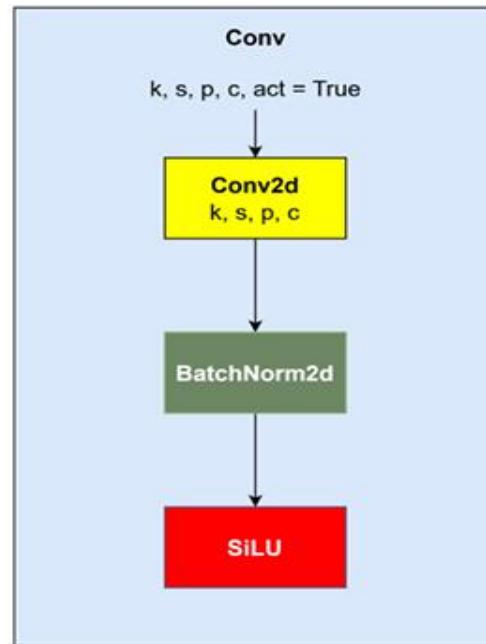
- Đầu (Head):** Phần đầu của YOLOv11 chịu trách nhiệm tạo ra các dự đoán cuối cùng về phát hiện và phân loại đối tượng. Nó xử lý các bản đồ đặc trưng từ cổ và cuối cùng xuất ra các hộp giới hạn và nhãn lớp cho các đối tượng trong hình ảnh.



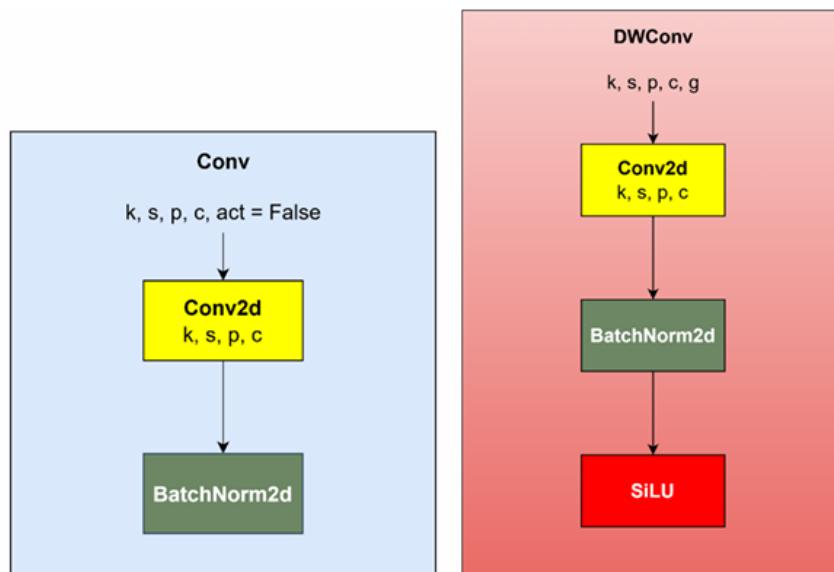
Hình 2.27. Kiến trúc YOLOv11 [18]

Dựa vào kiến trúc trên *hình 2.35* ta có thể thấy trong 1 phần của YOLOv11 có nhiều thành phần khác nhau bao gồm:

- Khối tích chập:** YOLOv11 duy trì cấu trúc tương tự như các phiên bản trước, sử dụng các lớp tích chập ban đầu để giảm kích thước hình ảnh. Những lớp này tạo thành nền tảng của quá trình trích xuất đặc trưng, dần dần giảm kích thước không gian trong khi tăng số lượng kênh (channel). Trong phiên bản YOLOv11, có 2 khối tích chập mới được giới thiệu so với khối tích chập chuẩn hiện nay đó là khối tích chập không có hàm kích hoạt và khối tích chập theo chiều sâu.

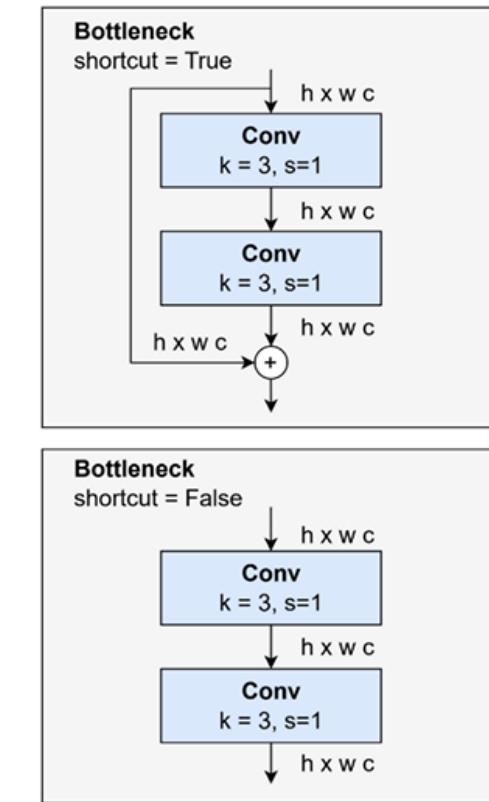


Hình 2.28. Khối tích chập chuẩn hiện nay [19]



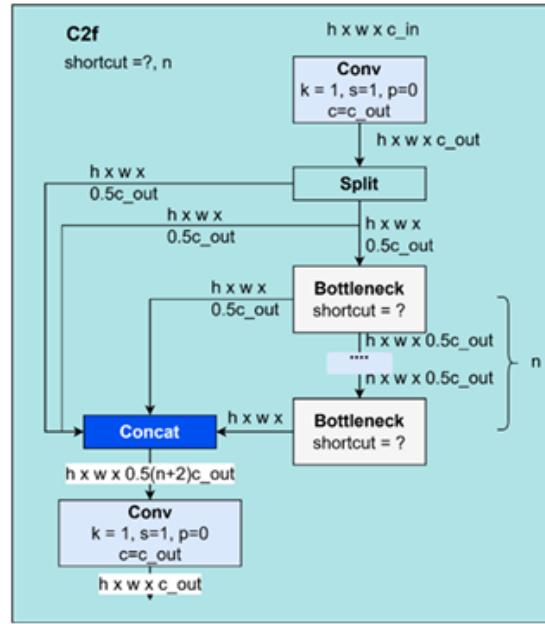
Hình 2.29. Khối tích chập không có hàm kích hoạt và khối tích chập theo chiều sâu [19]

- **Khối Bottleneck:** giúp tạo ra các mạng sâu hơn bằng cách xếp chồng các lớp bổ sung mà không làm tăng đáng kể chi phí tính toán. Có 2 kiểu bottleneck là sử dụng shortcut và không sử dụng shortcut.



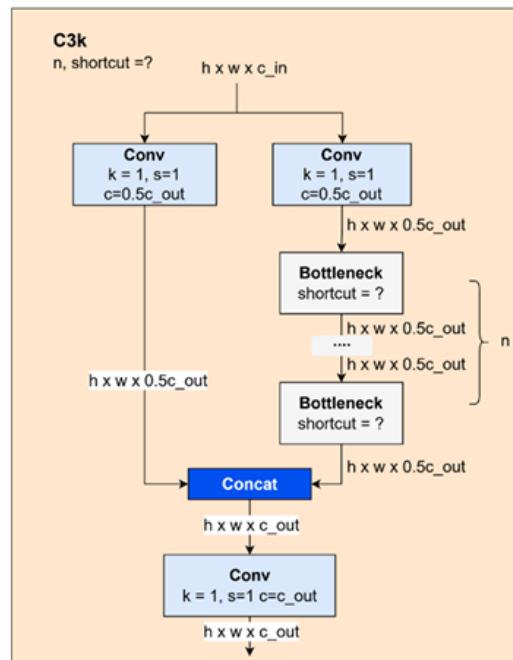
Hình 2.30. Khối bottleneck [19]

- **Khối C2f:** là một triển khai nhanh hơn của CSP (Cross-Stage Partial) Bottleneck với 2 phép tích chập.
- + Trong đó, CSP Bottleneck là một cấu trúc mạng được thiết kế để cải thiện hiệu suất tính toán và giảm thiểu độ trễ trong các mô hình phát hiện đối tượng, cho phép chia nhỏ các phép toán và giảm thiểu sự phụ thuộc giữa các lớp, từ đó tăng tốc quá trình huấn luyện và suy luận.
- + C2f được sử dụng để trích xuất đặc trưng ở tất cả các giai đoạn.
- + Các module của C2f có thể học thành công các đặc trưng đa tỷ lệ và mở rộng phạm vi của các trường tiếp xúc bằng cách sử dụng chuyển đổi vector đặc trưng và tích chập nhiều lớp.

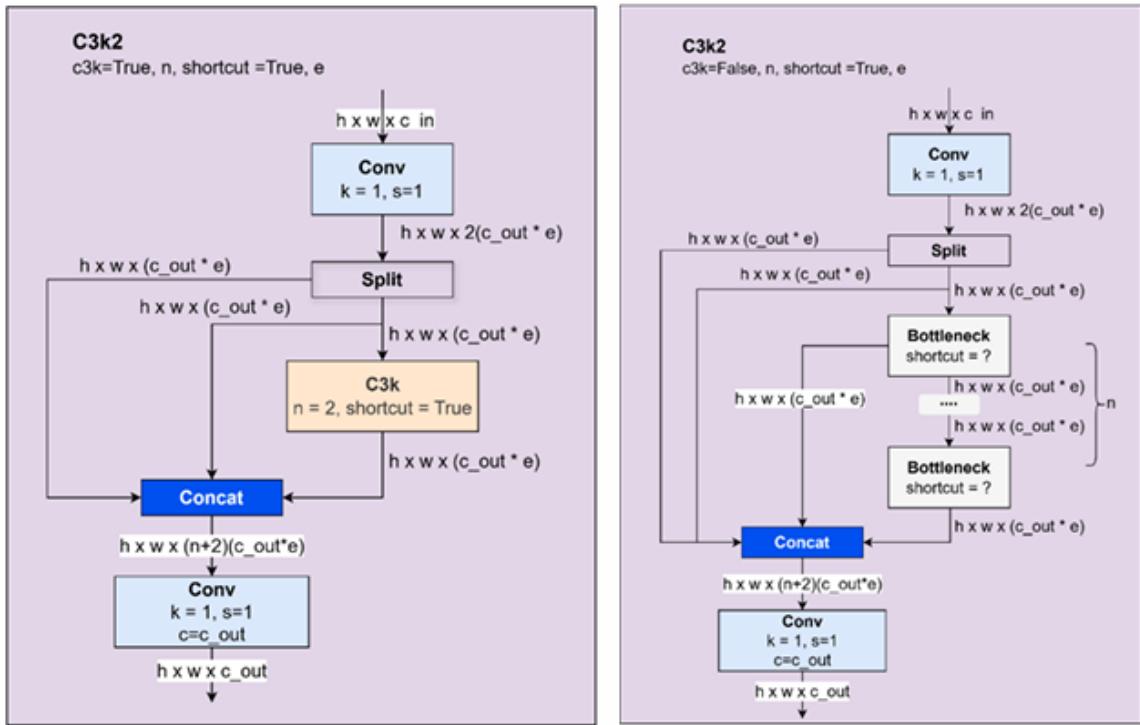


Hình 2.31. Khối C2f[19]

– **Khối C3k2:** là một khối mới được giới thiệu trong YOLOv11 để thay thế cho khối C2f. C3k2 có cấu trúc tương tự như C2f. Hơn nữa, phiên bản đầu tiên của C3k2 tương đương với C2f. Phiên bản thứ hai của C3k2 thay thế các bottleneck bằng một khối C3k. Khối C3k tự nó là một khối chứa 3 khối tích chập và một số bottleneck. Khối C3k2 giúp nắm bắt các đặc trưng phức tạp hơn, làm cho nó phù hợp với các đối tượng có kích thước thay đổi.

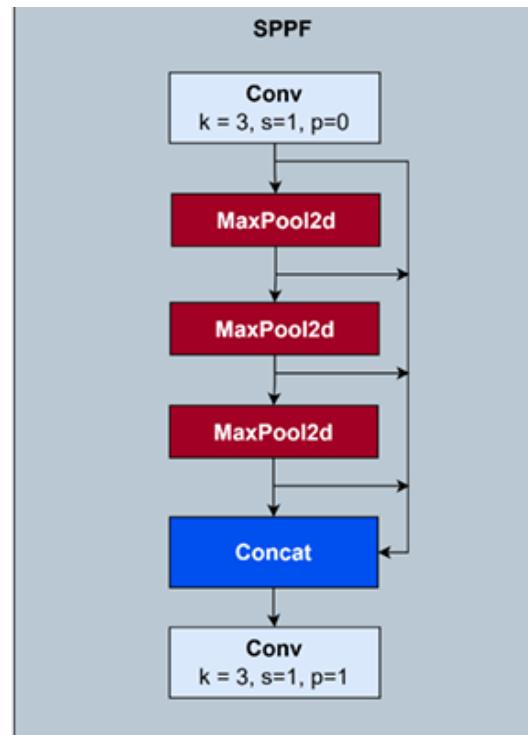


Hình 2.32. Khối C3k [19]



Hình 2.33. Khối C3k2 phiên bản đầu tiên và thứ hai. [19]

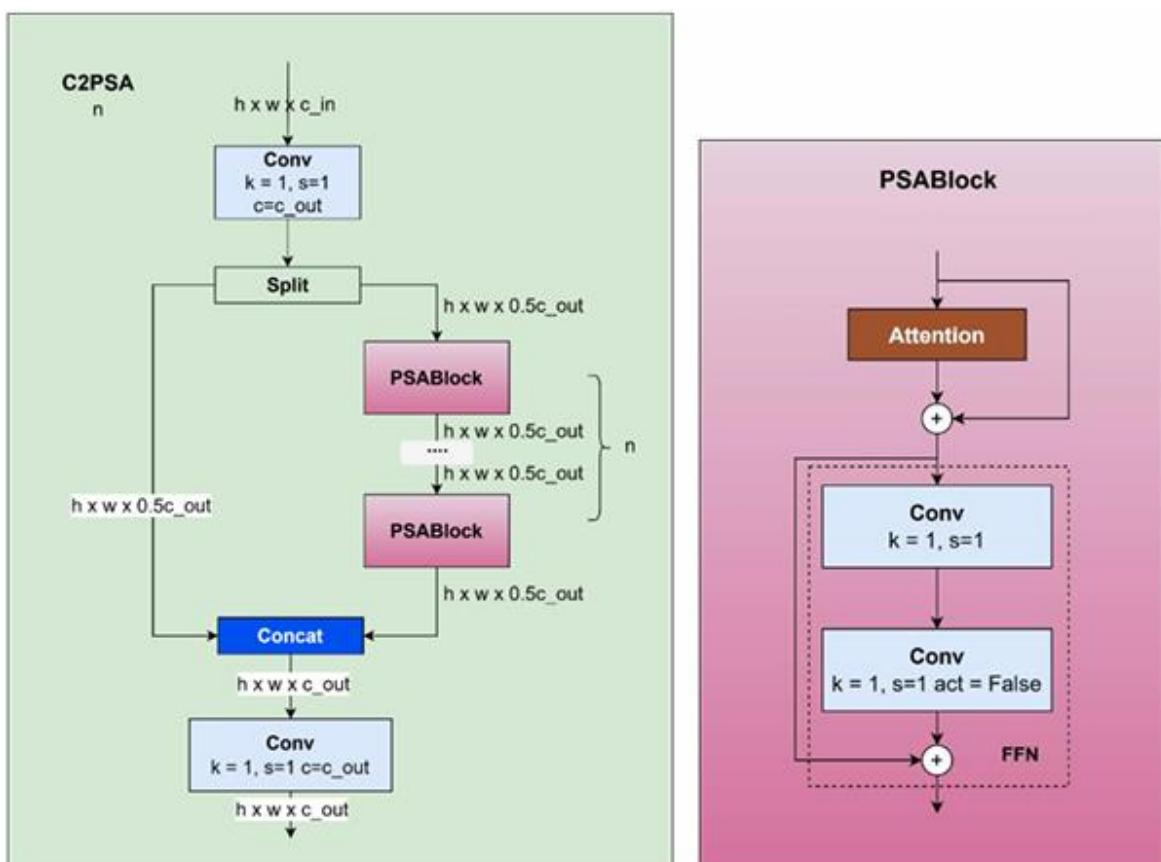
– **Khối SPPF:** Spatial Pyramid Pooling – Fast (SPPF) là một khối cho phép các bản đồ đặc trưng được đại diện ở nhiều tỷ lệ khác nhau. Bằng cách nhóm các đặc trưng ở các tỷ lệ khác nhau, SPPF giúp mô hình bắt được các đặc trưng ở các mức độ trừu tượng khác nhau.



Hình 2.34. Khối SPPF [19]

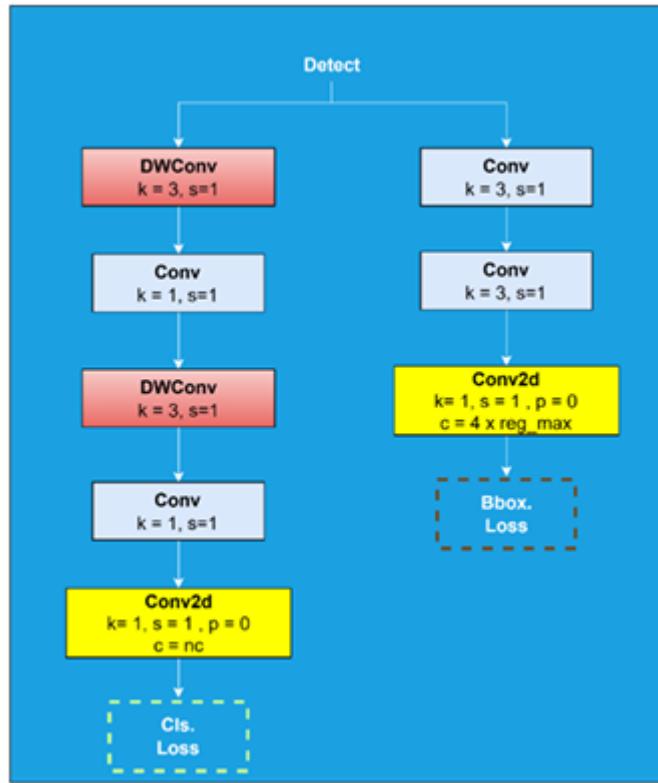
- Khối C2PSA và PSA

- + Trong phiên bản YOLOv10 và YOLOv11 có giới thiệu một khái niệm về khối chú ý (Attention Block) được sử dụng để xác định cách các điểm ảnh trong một hình ảnh hoặc khung video tương tác với nhau.
- + Khối PSA (Partial Self Attention) bao gồm 1 khối chú ý và 1 mạng truyền thẳng (Feed Forward Network) bao gồm 2 khối tích chập liên tiếp, khối tích chập thứ 2 không có hàm kích hoạt.
- + Khối C2PSA gồm nhiều khối PSA cho phép thực hiện thao tác PSA nhiều lần. qua đó bắt được các mối quan hệ đặc trưng sâu hơn. C2PSA hiệu quả trong việc bắt các đặc trưng sâu mà không tốn thêm thời gian, vì đầu ra chia ban đầu của phép tích chập chia đều vào thành hai nhánh: một nhánh ngắn và một nhánh xử lý sâu.



Hình 2.35. Khối C2PSA và PSA [19]

- **Khối phát hiện (Detect Block):** khối phát hiện bao gồm hai thành phần, cụ thể là đầu phân loại và đầu hồi quy. Xác suất lớp được tạo ra bởi đầu phân loại, trong khi các dự đoán tọa độ hộp giới hạn được tạo ra bởi đầu hồi quy.



Hình 2.36. Khởi phát hiện [19]

CHƯƠNG 3. THỰC NGHIỆM

3.1. Chuẩn bị dữ liệu

3.1.1. Thu thập dữ liệu

Dữ liệu được thu thập thủ công trên internet thông qua các website có bài viết liên quan đến bệnh cần thu thập. Các bệnh được lựa chọn thu thập để thực nghiệm là các bệnh dễ phát hiện bằng mắt thường bao gồm: nấm da, nhiễm trùng mắt, mụn trứng cá, u và rụng lông. Một số ví dụ về các bệnh phổ biến đã nêu trên như sau:



Hình 3.1. Minh họa bệnh mụn trứng cá ở mèo



Hình 3.2. Minh họa bệnh nhiễm trùng mắt ở mèo



Hình 3.3. Minh họa bệnh rụng lông ở mèo



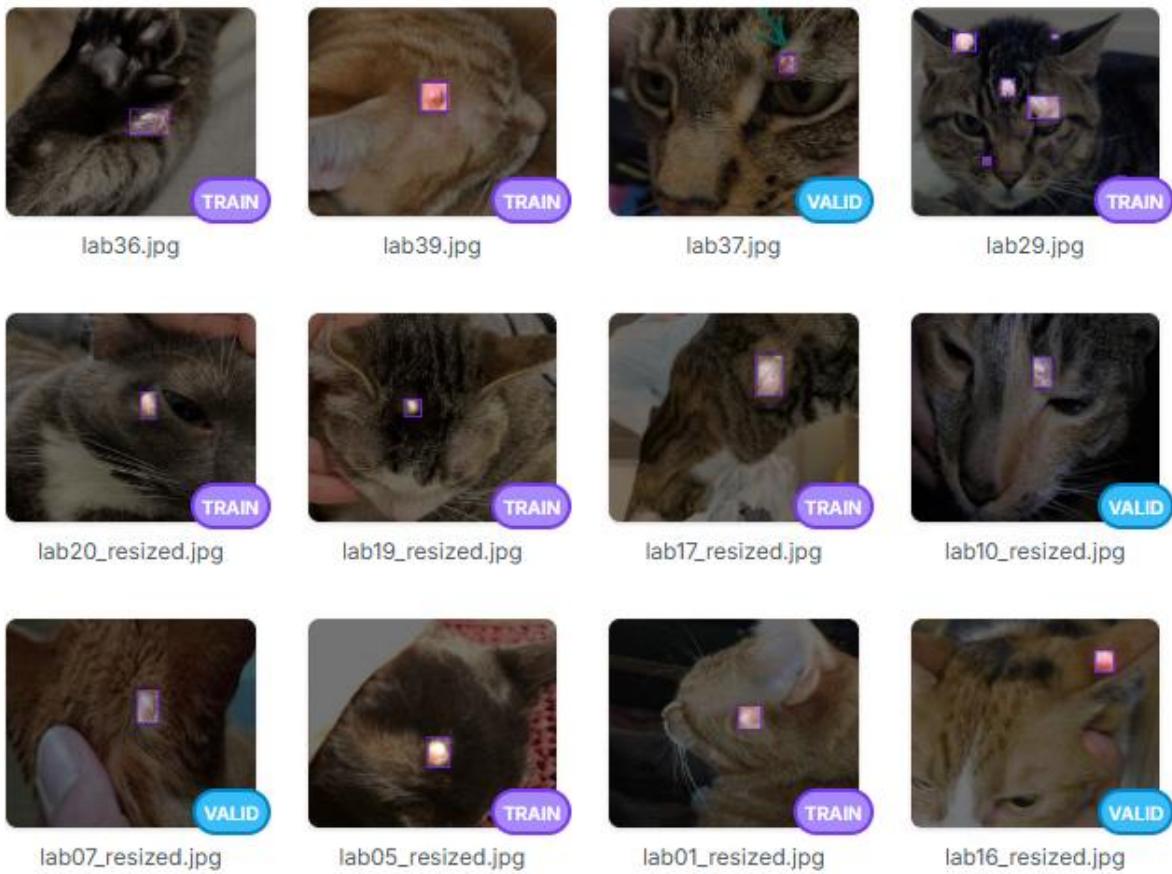
Hình 3.4. Minh họa bệnh u ở mèo



Hình 3.5. Minh họa bệnh nấm da ở mèo

3.1.2. Gán nhãn dữ liệu

Công cụ gán nhãn dữ liệu được sử dụng trong đề tài này là Roboflow. Đây là một công cụ gán nhãn có nhiều tiện ích hỗ trợ người dùng trong gán nhãn và chia tập dữ liệu với nhiều cách thức gán nhãn phục vụ cho nhiều mô hình khác nhau.

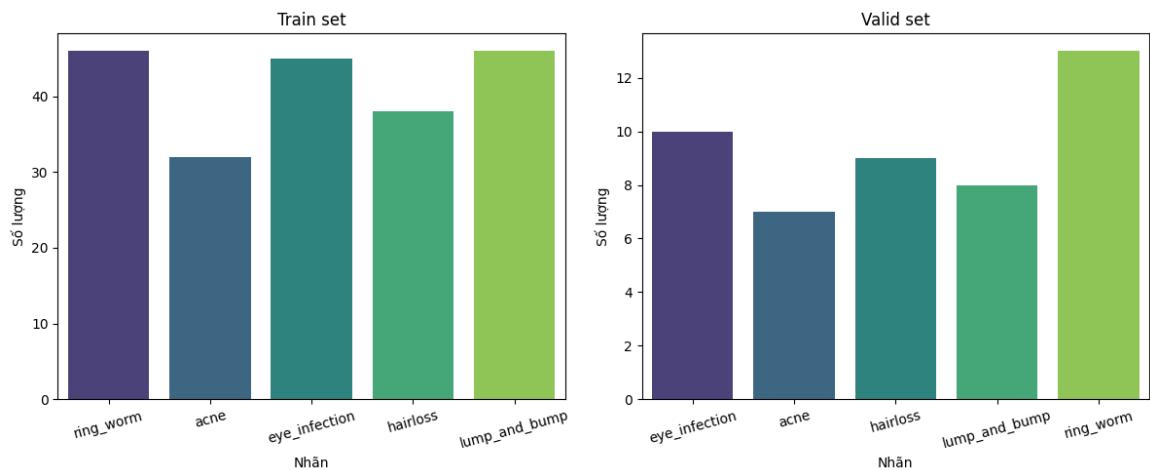


Hình 3.6. Minh họa gán nhãn với công cụ Roboflow

3.1.3. Phân tích dữ liệu

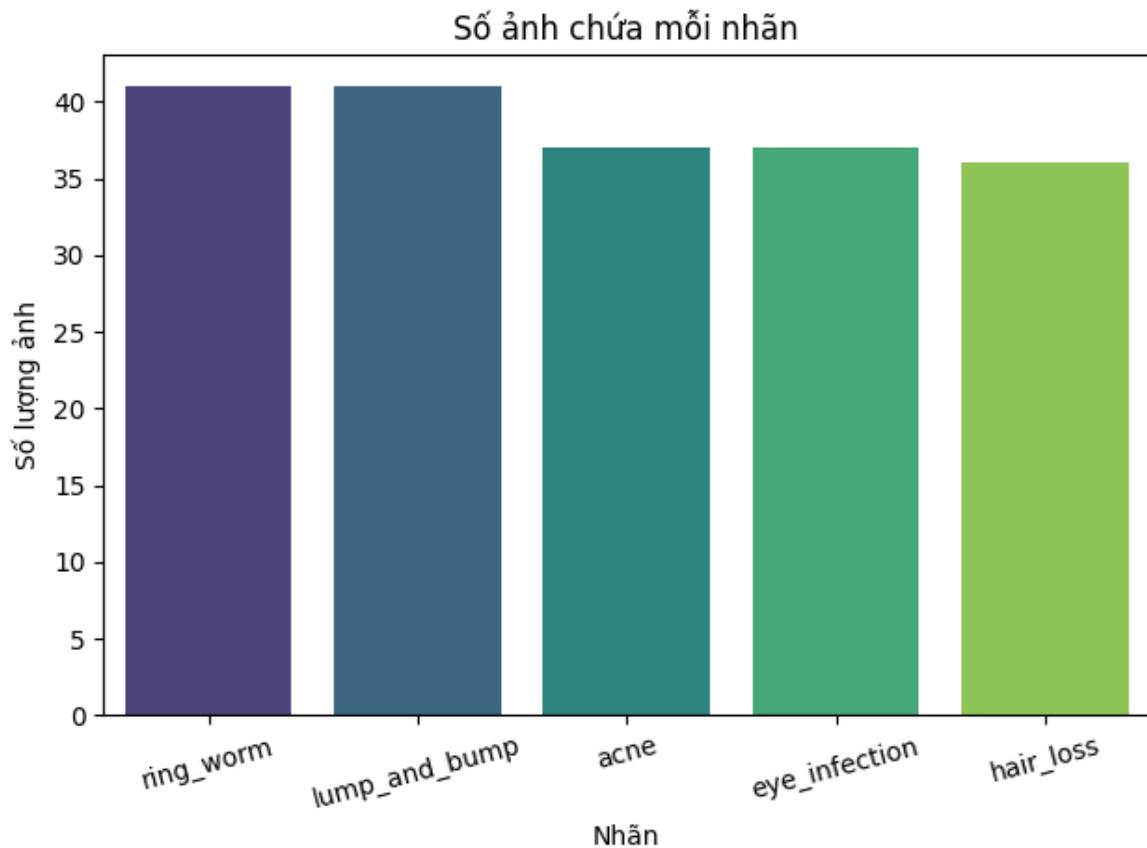
Sử dụng Google Colab để phân tích dựa trên tập dữ liệu đã gán nhãn một cách trực quan và hiệu quả.

- Kiểm tra số lượng nhãn phân bố trên tập train và validation.



Hình 3.7. Kiểm tra số lượng nhãn phân bố

- Kiểm tra số lượng ảnh chứa mỗi nhãn.



Hình 3.8. Kiểm tra số lượng ảnh chứa mỗi nhãn

- Kiểm tra số lượng ảnh không được gán nhãn và không đúng định dạng

```
#check train dataset
image_dir = path + "/train/images"
label_dir = path + "/train/labels"

check_dataset_validity(image_dir, label_dir)
```

🔍 Phát hiện 0 ảnh không hợp lệ.
[]

```
#check validation dataset
image_dir = path + "/valid/images"
label_dir = path + "/valid/labels"

check_dataset_validity(image_dir, label_dir)
```

🔍 Phát hiện 0 ảnh không hợp lệ.
[]

Hình 3.9. Kết quả kiểm tra dữ liệu trong tập train và validation

Dựa vào việc kiểm tra dữ liệu thông qua phân tích, ta thấy dữ liệu sau khi gán nhãn không bị mất cân bằng và không có dữ liệu không hợp lệ.

3.1.4. Tiền xử lý dữ liệu

Việc tiền xử lý dữ liệu ảnh sẽ gồm 3 bước bao gồm: giảm nhiễu, cân bằng histogram trên ảnh và resize theo kích cỡ phù hợp.

Để tiền xử lý dữ liệu một cách hiệu quả, ta sử dụng thư viện albumentations. Đây là một thư viện hỗ trợ việc tiền xử lý và tăng cường dữ liệu một cách hiệu quả và vẫn đảm bảo rằng nhãn được gán sẽ biến đổi phù hợp theo ảnh.

```
#build pipeline
def preprocessing_pipeline(target_size=(640, 640)):
    return A.Compose([
        A.Lambda(image=denoise_image),           # xử lý nhiễu
        A.Lambda(image=equalize_histogram),       # cân bằng histogram
        A.Resize(*target_size),                  # resize ảnh
    ], bbox_params=A.BboxParams(format='yolo', label_fields=['class_labels']))
```

Hình 3.10. Quy trình tiền xử lý



Hình 3.11. Kết quả tiền xử lý dữ liệu

3.1.5. Tăng cường dữ liệu

Việc tăng cường dữ liệu được thực hiện nhằm đa dạng hóa và làm giàu dữ liệu. Từ đó tăng khả năng học của mô hình khi huấn luyện và độ chính xác tối ưu hơn.

Để thực hiện việc tăng cường dữ liệu, ta sử dụng thư viện albumentations để hỗ trợ.

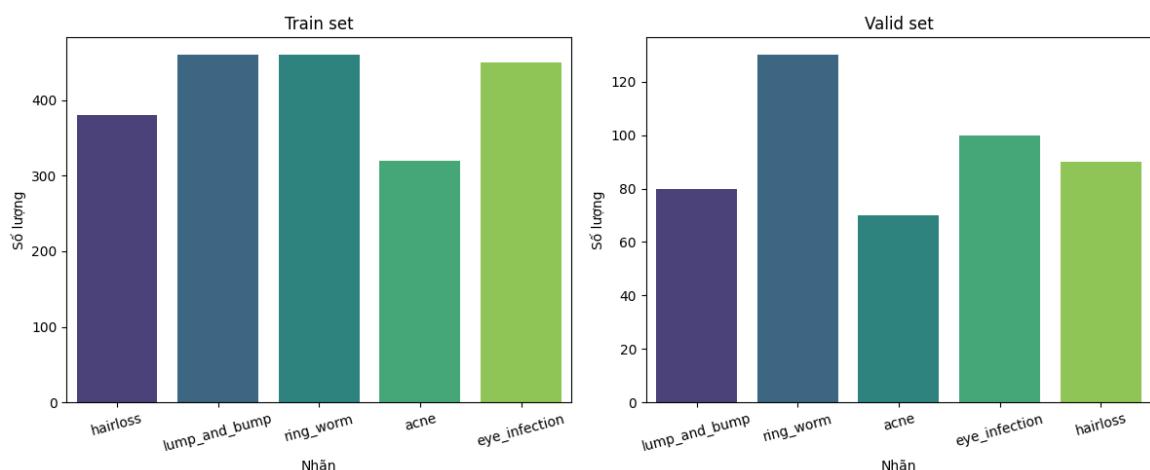
Các kỹ thuật thực hiện tăng cường dữ liệu được thực hiện bao gồm: xoay ảnh (rotate), biến đổi hình học (shear), điều chỉnh độ sáng (brightness), điều chỉnh không gian màu (saturation).

```
def augmentation_pipeline(self):
    return A.Compose([
        A.OneOf([
            A.Rotate(limit=(90, 90), p=0.33),
            A.Rotate(limit=(180, 180), p=0.33),
            A.Rotate(limit=(270, 270), p=0.34)
        ], p=1.0),
        A.Affine(shear={"x": (-20, 20)}, p=0.8),
        A.RandomBrightnessContrast(brightness_limit=0.2, contrast_limit=0.2, p=0.7),
        A.HueSaturationValue(hue_shift_limit=0, sat_shift_limit=30, val_shift_limit=0, p=0.7),
    ],
    bbox_params=A.BboxParams(format='yolo', label_fields=['class_labels']))
```

Hình 3.12. Quy trình tăng cường ảnh



Hình 3.13. Minh họa dữ liệu sau khi biến đổi



Hình 3.14. Dữ liệu sau khi tăng cường

3.2. Huấn luyện mô hình

3.2.1. Lựa chọn siêu tham số

Việc lựa chọn siêu tham số phù hợp đóng vai trò cực kì quan trọng trong quá trình huấn luyện mô hình. Điều này có thể tiết kiệm thời gian huấn luyện đồng thời giúp mô hình hội tụ nhanh hơn. Một siêu tham số tốt sẽ giúp mô hình học hiệu quả và tránh overfitting.

```
model.train(
    data='/content/drive/MyDrive/data.yaml',
    epochs=100,
    batch=32,
    lr0=0.0001,
    weight_decay=0.0005,
    optimizer='adamw',
    lrf=0.1,
    mosaic=True,
    mixup=0.1,
    imgsz=416
)
```

Hình 3.15. Minh họa lựa chọn siêu tham số

Các siêu tham số được lựa chọn bao gồm:

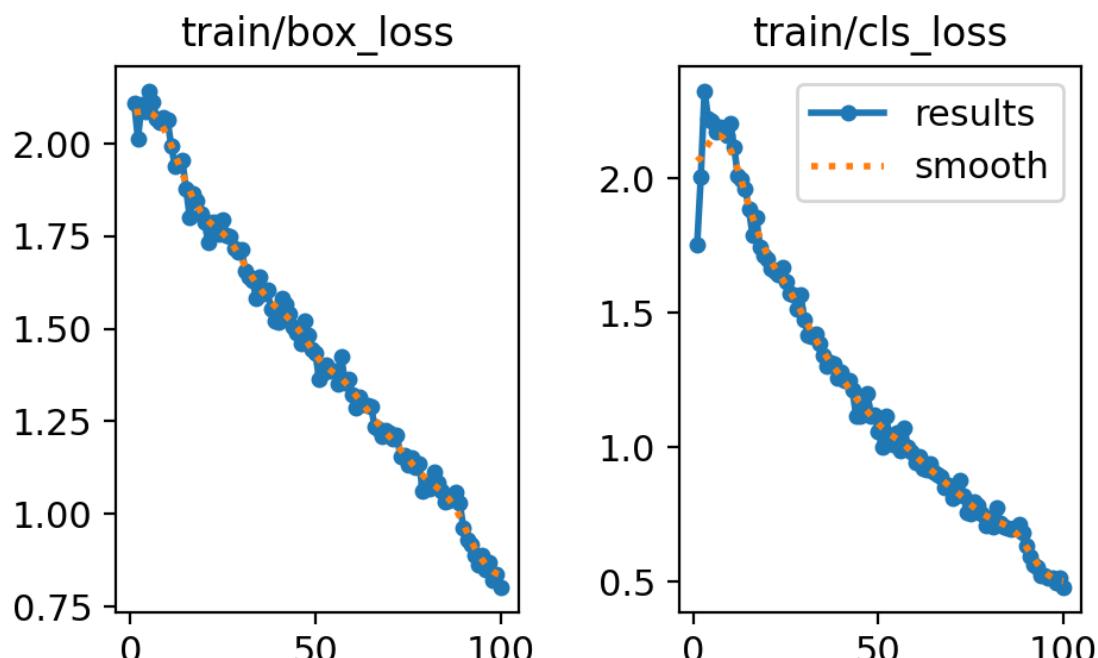
- Epochs: Tham số này xác định số lượng epochs (vòng lặp) huấn luyện. Một epoch là một lần duyệt qua toàn bộ tập dữ liệu huấn luyện. Huấn luyện với nhiều epochs hơn có thể giúp mô hình học được các đặc trưng phức tạp hơn từ dữ liệu, nhưng cũng có nguy cơ overfitting (mô hình học quá tốt trên dữ liệu huấn luyện nhưng kém hiệu quả trên dữ liệu mới).
- Batch: Tham số này thiết lập kích thước batch (batch size). Trong quá trình huấn luyện, mô hình sẽ xử lý đồng thời một nhóm gồm 32 mẫu dữ liệu (ảnh) trước khi cập nhật các trọng số của nó. Kích thước batch ảnh hưởng đến tốc độ huấn luyện, việc sử dụng bộ nhớ và độ ổn định của quá trình học.
- Lr0: Tham số này đặt tỷ lệ học ban đầu (initial learning rate) cho trình tối ưu hóa. Tỷ lệ học quyết định kích thước của các bước điều chỉnh trọng số của mô hình trong mỗi lần cập nhật. Một tỷ lệ học quá lớn có thể khiến quá trình huấn luyện không ổn định, trong khi một tỷ lệ học quá nhỏ có thể làm cho quá trình huấn luyện diễn ra rất chậm.

- Weight_decay: Tham số này quy định hệ số weight decay (l2 regularization). Weight decay là một kỹ thuật regularization để ngăn chặn overfitting bằng cách thêm một hình phạt vào hàm mất mát dựa trên độ lớn của các trọng số của mô hình.
- Optimizer: Tham số này chọn trình tối ưu hóa (optimizer) được sử dụng để cập nhật trọng số của mô hình trong quá trình huấn luyện. 'adamw' là một biến thể của thuật toán Adam, thường mang lại hiệu suất tốt hơn Adam trong nhiều trường hợp, đặc biệt là khi kết hợp với weight decay.
- Lrf: Tham số này xác định hệ số giảm tỷ lệ học cuối cùng (learning rate final factor). Nó kiểm soát mức độ mà tỷ lệ học sẽ giảm xuống vào cuối quá trình huấn luyện so với tỷ lệ học ban đầu (lr0).
- Mosaic: Tham số này kích hoạt kỹ thuật Mosaic augmentation. Mosaic là một kỹ thuật tăng cường dữ liệu bằng cách ghép bốn ảnh huấn luyện khác nhau thành một ảnh mới. Điều này giúp mô hình học được cách phát hiện các đối tượng ở các tỷ lệ và ngũ cành khác nhau, đồng thời làm giảm sự phụ thuộc vào vị trí tuyệt đối của đối tượng trong ảnh.
- Mixup: Tham số này chỉ định xác suất áp dụng kỹ thuật MixUp augmentation. MixUp là một kỹ thuật tăng cường dữ liệu bằng cách tạo ra các mẫu huấn luyện mới là sự kết hợp tuyến tính của hai mẫu huấn luyện khác nhau và nhãn tương ứng của chúng. Giá trị 0.1 có nghĩa là có 10% khả năng một batch dữ liệu sẽ được áp dụng MixUp. MixUp giúp mô hình trở nên mượt mà hơn trong việc đưa ra dự đoán giữa các lớp và cải thiện khả năng chống lại nhiễu.
- Imgsz: Tham số này đặt kích thước ảnh đầu vào cho mô hình. Tất cả các ảnh huấn luyện sẽ được resize về kích thước được truyền vào trước khi được đưa vào mô hình. Kích thước ảnh đầu vào ảnh hưởng đến tốc độ huấn luyện và hiệu suất của mô hình. Kích thước lớn hơn thường đòi hỏi nhiều bộ nhớ hơn và có thể cải thiện độ chính xác nhưng làm chậm quá trình huấn luyện.

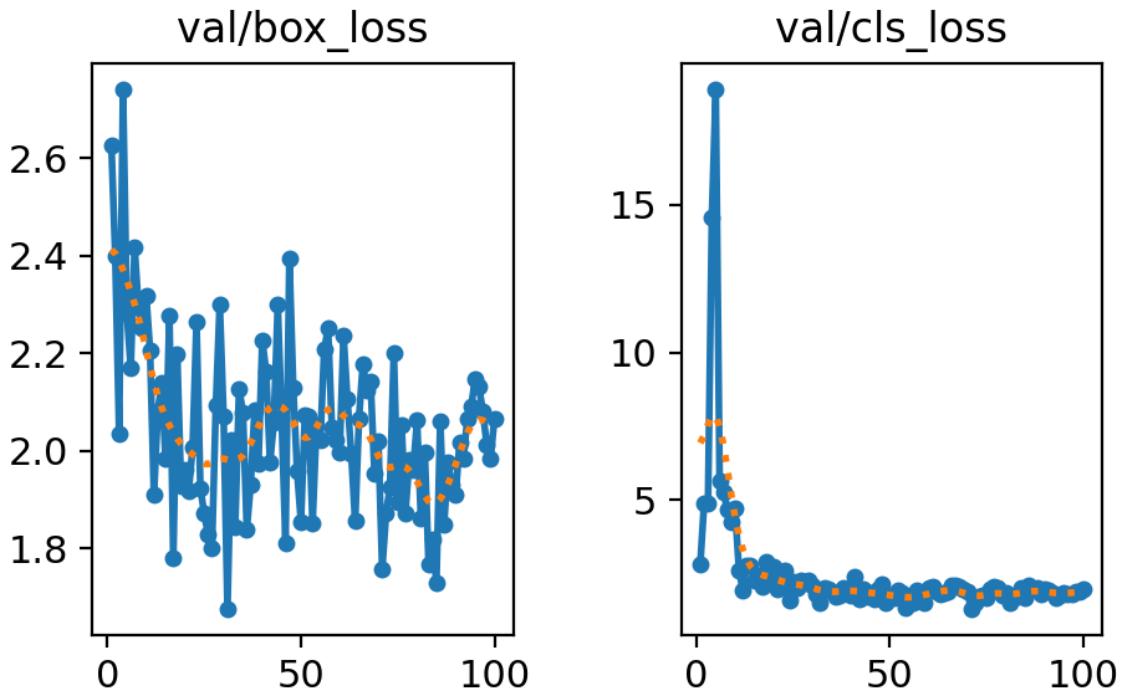
3.2.2. Kết quả sau khi huấn luyện

Trong quá trình huấn luyện các hàm loss sẽ được tối ưu hóa để mô hình hội tụ tham số một cách nhanh và chuẩn nhất. Có 2 hàm loss cần phải tối ưu đối với bài toán phát hiện đối tượng (object detection) bao gồm:

- Box loss: Đo lường mức độ chính xác của các hộp giới hạn mà mô hình dự đoán so với các hộp giới hạn thực tế của các đối tượng trong ảnh. Nói cách khác, nó cho biết mô hình định vị các đối tượng tốt như thế nào. Đối với mô hình YOLO là tối ưu cho IOU loss.
- Classification loss: Đo lường mức độ chính xác của mô hình trong việc phân loại các đối tượng được phát hiện vào đúng lớp của chúng. Nói cách khác, nó cho biết mô hình nhận diện các đối tượng thuộc loại nào tốt như thế nào.



Hình 3.16. Đo lường loss trên tập train



Hình 3.17. Đo lường loss trên tập validation

Dựa vào các biểu đồ trực quan về việc tối ưu lỗi trong quá trình huấn luyện và validation, ta có thể thấy sự thay đổi liên tục về mặt giá trị. Điều này chứng tỏ mô hình vẫn còn bị overfit khi gặp dữ liệu mới, gây ra dự đoán sai so với thực tế ảnh hưởng tới giá trị của loss tại mỗi thời điểm.

Mô hình sau khi huấn luyện được đánh giá dựa trên các độ đo (metrics) sau:

- Precision: là độ chính xác hay độ tin cậy của model dựa trên công thức sau:

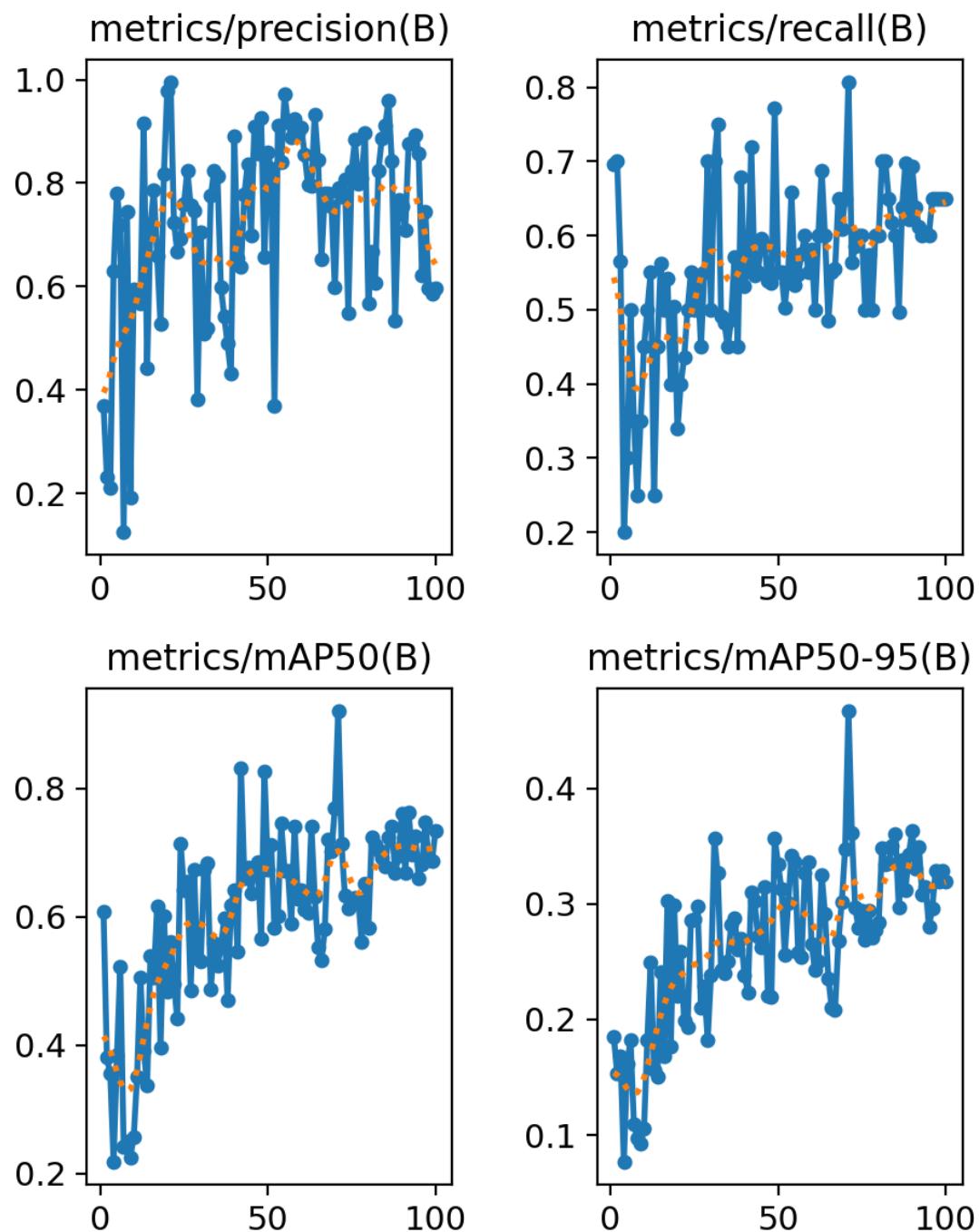
$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

- Recall: là độ nhạy của model được tính dựa trên công thức sau:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

- mAP50: là thước đo để đo lường lường độ chính xác trung bình của mô hình khi ngưỡng IoU được đặt ở 50%. Độ đo này đại diện cho việc mô hình chỉ xem xét các phát hiện "dễ dàng".

- mAP50-95: là thước đo để đo lường lường độ chính xác trung bình của mô hình tại các ngưỡng IoU từ 0.5 đến 0.95 (bước 0.05). Độ đo này cung cấp một cái nhìn toàn diện về hiệu suất của mô hình qua các mức độ khó phát hiện khác nhau.



Hình 3.18. Độ đo của mô hình

Thông qua biểu đồ biến đổi trong quá trình huấn luyện của các độ đo, ta thấy mô hình chưa thực sự nhạy với các dữ liệu có đặc điểm nhận dạng khó. Điều này thể hiện thông qua thông số mAP50-95 và recall.

Thử nghiệm trên tập test với các ảnh chưa được tăng cường ta được kết quả như sau:

Class	Images	Instances	Box(P)	R	mAP50	mAP50-95):
all	227	267	0.588	0.478	0.488	0.227
acne	42	42	0.649	0.643	0.599	0.312
eye_infection	42	60	0.857	0.717	0.828	0.406
hairloss	42	59	0.312	0.22	0.139	0.044
lump_and_bump	53	53	0.585	0.509	0.549	0.247
ringworm	48	53	0.537	0.302	0.323	0.127

Hình 3.19. Điểm đánh giá của mô hình trên tập test

Như vậy, thông qua tập test ta có thể thấy được sự mất cân bằng khá rõ, đặc biệt là đối với bệnh rụng lông khi precision và recall khá thấp, điều này xảy ra do sự khác biệt về màu da và lông trên mỗi vật thể có sự khác biệt và không đồng nhất. Ngoài ra các mức điểm trung bình mAP50 chỉ ở khoảng 0.5 cho thấy mô hình vẫn còn chưa thực sự tốt đối với việc đặt các hộp giới hạn. Nhìn chung về tổng quan mô hình thực sự ổn đối với đặc điểm của nhiễm trùng mắt khi precision lên tới 0.875 và 0.717.

Để mô hình có thể tăng độ chính xác là một quá trình dài liên tục điều chỉnh tham số và xem xét các phương pháp tối ưu. Ngoài ra vấn đề dữ liệu cũng là một vấn đề nghiêm trọng ảnh hưởng đến độ chính xác của mô hình khi mà bộ dữ liệu chưa thực sự đa dạng và các đặc điểm nhận biết không tập trung

CHƯƠNG 4. XÂY DỰNG ỨNG DỤNG

4.1. Giới thiệu công cụ sử dụng

4.1.1. Streamlit Framework

Streamlit là một framework mã nguồn mở dành cho Python, giúp các nhà khoa học dữ liệu và lập trình viên dễ dàng xây dựng các ứng dụng web trực quan để phân tích dữ liệu và triển khai mô hình machine learning (ML). Ra mắt lần đầu vào năm 2019, Streamlit nhanh chóng trở thành lựa chọn phổ biến nhờ tính đơn giản, nhanh chóng và khả năng tạo giao diện người dùng mà không cần kinh nghiệm lập trình web.

Streamlit giúp lập trình viên chuyển đổi các mô hình machine learning thành ứng dụng web một cách dễ dàng, cho phép hiển thị các biểu đồ, hình ảnh, bảng và văn bản ngay trên giao diện của ứng dụng mà không cần cấu hình phức tạp.

Streamlit nổi bật với nhiều ưu điểm khiến nó trở thành công cụ hàng đầu cho các dự án phân tích dữ liệu và ML:

- **Dễ Sử Dụng:** Với Streamlit, bạn chỉ cần biết Python để tạo ứng dụng web. Không yêu cầu kiến thức về HTML, CSS hay JavaScript.
- **Hiệu Suất Cao:** Streamlit tự động cập nhật dữ liệu trong ứng dụng mỗi khi mã nguồn thay đổi, giúp tiết kiệm thời gian phát triển và bảo trì.
- **Tích Hợp Dữ Liệu Dễ Dàng:** Streamlit hỗ trợ trực tiếp các thư viện phân tích dữ liệu như Pandas, NumPy và Matplotlib, giúp dễ dàng tích hợp và hiển thị dữ liệu.
- **Mã Nguồn Mở Và Miễn Phí:** Streamlit là mã nguồn mở, nên cộng đồng người dùng luôn có thể đóng góp và phát triển thêm tính năng mới.

4.1.2. Git

Git là một hệ thống quản lý phiên bản phân tán (Distributed Version Control System - DVCS) được thiết kế để theo dõi các thay đổi trong tệp tin và phối hợp làm việc nhóm trong quá trình phát triển phần mềm. Nó ghi lại lịch

sử các thay đổi, cho phép bạn quay lại các phiên bản trước đó, so sánh các thay đổi và hợp nhất công việc từ nhiều người.

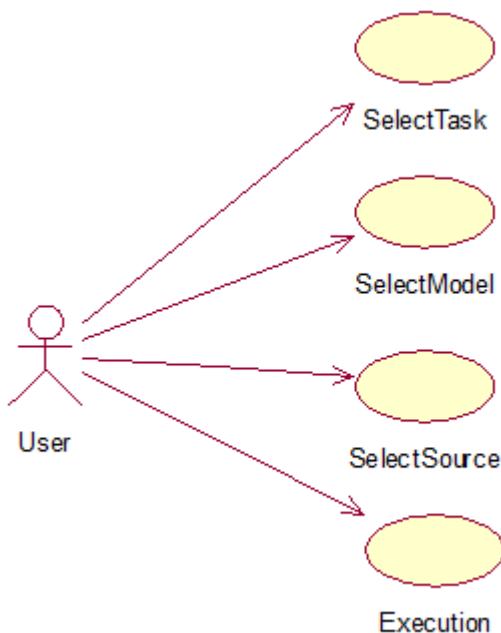
Git đóng vai trò quan trọng trong việc triển khai sản phẩm, đặc biệt là trong các quy trình DevOps và CI/CD. Dưới đây là một số ứng dụng chính:

- **Nguồn lưu trữ duy nhất:** Git repository chứa phiên bản mã nguồn cuối cùng và đã được kiểm thử, sẵn sàng để triển khai. Đây là nơi mọi thay đổi đều được ghi lại và có thể kiểm soát.
- **Quản lý phiên bản triển khai:** Mỗi lần triển khai thành công có thể được đánh dấu bằng một tag trong Git (ví dụ: v1.0.0, release-2023-10-27). Điều này giúp dễ dàng theo dõi các phiên bản đã triển khai và quay lại nếu cần.
- **Kích hoạt quy trình CI/CD:** Các nền tảng CI/CD thường tích hợp chặt chẽ với Git. Khi có một commit mới được đẩy lên một nhánh cụ thể (ví dụ: nhánh chính hoặc nhánh phát hành), hệ thống CI/CD sẽ tự động build, kiểm thử và triển khai ứng dụng.
- **Triển khai dựa trên nhánh:** Các nhánh Git khác nhau có thể tương ứng với các môi trường triển khai khác nhau (ví dụ: develop cho môi trường phát triển, staging cho môi trường thử nghiệm, main cho môi trường sản xuất). Việc hợp nhất mã vào một nhánh cụ thể sẽ kích hoạt quy trình triển khai cho môi trường đó.
- **GitOps:** Đây là một phương pháp triển khai hiện đại, trong đó trạng thái mong muốn của hệ thống (bao gồm cả ứng dụng và cơ sở hạ tầng) được khai báo và quản lý trong Git. Các công cụ tự động hóa sẽ theo dõi kho lưu trữ Git và đảm bảo rằng môi trường thực tế khớp với trạng thái đã khai báo.
- **Sử dụng Git Hooks cho tự động hóa:** Git hooks là các script có thể được kích hoạt trước hoặc sau các sự kiện Git (ví dụ: pre-commit, post-receive). Chúng có thể được sử dụng để tự động hóa các tác vụ liên quan đến triển khai, chẳng hạn như chạy script build, sao chép tệp lên máy chủ, hoặc thông báo về việc triển khai thành công.

4.2. Phân tích thiết kế hệ thống

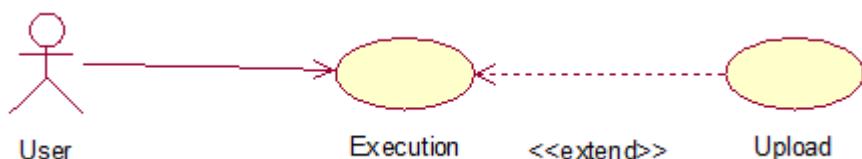
4.2.1. Biểu đồ use case

Biểu đồ use case tổng quan



Hình 4.1. Biểu đồ use case tổng quát

Phân rã use case Execution



Hình 4.2. Biểu đồ phân rã use case Execution

4.2.2. Đặc tả use case

1. Đặc tả use case Select Task

Mã use case	UC1
Tên use case	Select Task
Tóm tắt	Use case cho phép người dùng chọn nhiệm vụ mà mình muốn thực hiện
Actor	User
Tiền điều kiện	Trình duyệt được bật
Đảm bảo tối thiểu	Người dùng mở được Select Task

Đảm bảo thành công	Người dùng chọn nhiệm vụ cần thực hiện thành công
Kích hoạt	Người dùng nhập chọn nhiệm vụ được hiển thị ra
Luồng các sự kiện:	
<ul style="list-style-type: none"> - Luồng cơ bản: <ol style="list-style-type: none"> 1) Người dùng nhấp chọn nút “Select Task” và màn hình hiển thị ra danh sách các nhiệm vụ được cấu hình sẵn. 2) Người dùng lựa chọn tác vụ được hiển thị trên màn hình. 3) Hệ thống ghi nhận nhiệm vụ. Use case kết thúc. 	
Ngoại lệ: Không có	
Hậu điều kiện: Không có	

Bảng 4.1. ĐẶC TẢ USE CASE SELECT TASK

2. ĐẶC TẢ USE CASE SELECT MODEL

Mã use case	UC2
Tên use case	Select Model
Tóm tắt	Use case cho phép người dùng chọn mô hình phù hợp dựa trên nhiệm vụ đã chọn
Actor	User
Tiền điều kiện	Trình duyệt được bật và use case Select Task đã được thực hiện
Đảm bảo tối thiểu	Người dùng mở được Select Model
Đảm bảo thành công	Người dùng chọn mô hình muốn sử dụng thành công
Kích hoạt	Người dùng nhập chọn mô hình được hiển thị ra
Luồng sự kiện:	
<ul style="list-style-type: none"> - Luồng cơ bản: <ol style="list-style-type: none"> 1) Người dùng nhấp chọn nút “Select Model” và màn hình hiển thị ra danh sách các mô hình được cấu hình sẵn. 2) Người dùng lựa chọn mô hình được hiển thị trên màn hình. 	

3) Hệ thống ghi nhận mô hình. Use case kết thúc.
Ngoại lệ: Không có
Hậu điều kiện: Không có

Bảng 4.2. Đặc tả use case Select Model

3. Đặc tả use case Select Source

Mã use case	UC3
Tên use case	Select Source
Tóm tắt	Use case cho phép người dùng chọn nguồn dữ liệu dựa trên nhiệm vụ đã chọn
Actor	User
Tiền điều kiện	Trình duyệt được bật
Đảm bảo tối thiểu	Người dùng mở được Select Source
Đảm bảo thành công	Người dùng chọn nguồn muốn tải thành công
Kích hoạt	Người dùng nhập chọn các nguồn được hiển thị ra

Luồng sự kiện:

- Luồng cơ bản:
 - 1) Người dùng nhập chọn nút “Select Source” và màn hình hiển thị ra danh sách các nguồn dữ liệu được cấu hình sẵn.
 - 2) Người dùng lựa chọn nguồn được hiển thị trên màn hình.
 - 3) Hệ thống ghi nhận nguồn. Use case kết thúc.

Ngoại lệ: Không có
Hậu điều kiện: Không có

Bảng 4.3. Đặc tả use case Select Source

4. Đặc tả use case Execution

Mã use case	UC4
Tên use case	Execution
Tóm tắt	Use case cho phép người dùng thực hiện nhận diện bệnh trên mèo dựa trên dữ liệu đưa vào
Actor	User

Tiền điều kiện	Trình duyệt được bật và use case Select Source đã được thực hiện
Đảm bảo tối thiểu	Người dùng upload source theo nguồn dữ liệu đã lựa chọn
Đảm bảo thành công	Người dùng upload dữ liệu thành công
Kích hoạt	Người dùng nhấp chọn nút “Execution” để thực hiện
Luồng sự kiện:	
<ul style="list-style-type: none"> - Luồng cơ bản: <ol style="list-style-type: none"> 1) Use case thực hiện khi người dùng thực hiện nhấp nút “Execution” hoặc hệ thống tự động thực hiện đối với nguồn đã chọn là webcam. 2) Hệ thống thực hiện dựa trên gán nhãn và đánh dấu bounding box dựa trên bệnh mà hệ thống nhận diện được từ dữ liệu đầu vào. 3) Hệ thống hiển thị kết quả. Use case kết thúc. 	
<ul style="list-style-type: none"> - Luồng rẽ nhánh: <ol style="list-style-type: none"> 1) Tại bước 2, nếu hệ thống không ghi nhận dữ liệu đầu vào thì hệ thống không thực hiện và use case kết thúc. 	
Ngoại lệ: Không có	
Hậu điều kiện: Không có	

Bảng 4.4. ĐẶC TẢ USE CASE EXECUTION

5. Đặc tả use case Upload

Mã use case	UC5
Tên use case	Upload
Tóm tắt	Use case cho phép người dùng tải lên nguồn dữ liệu đã chọn từ Select Source
Actor	User
Tiền điều kiện	Trình duyệt được bật và use case Select Source đã được thực hiện với lựa chọn là Image hoặc Video
Đảm bảo tối thiểu	Người dùng upload source theo nguồn dữ liệu đã lựa chọn

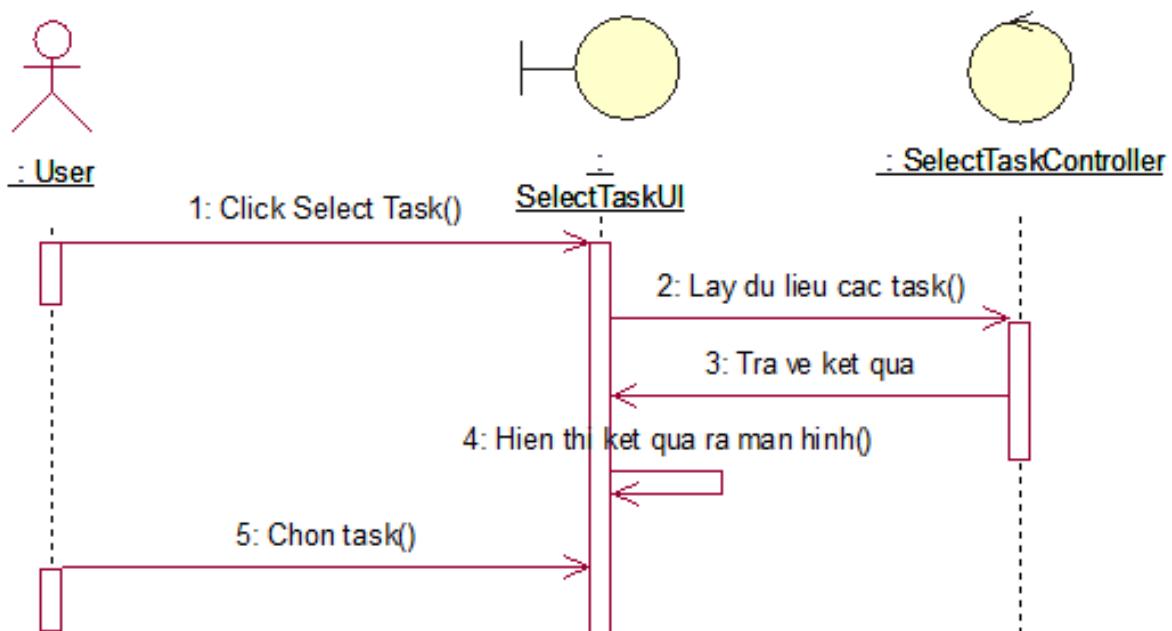
Đảm bảo thành công	Người dùng mở được nguồn dữ liệu
Kích hoạt	Người dùng nhấp chọn nút “Open” sau khi chọn được dữ liệu
Luồng sự kiện:	
<ul style="list-style-type: none"> - Luồng cơ bản: <ol style="list-style-type: none"> 1) Use case thực hiện sau khi người dùng thực hiện “Browse files”. 2) Người dùng cần chọn các tệp cần tảo tin có định dạng phù hợp. Sau đó, kích vào nút “open” để thực hiện upload dữ liệu. Use case kết thúc. 	
<ul style="list-style-type: none"> - Luồng rẽ nhánh: <ol style="list-style-type: none"> 1) Tại bước 2, nếu người dùng tải sai định dạng của dữ liệu. Hệ thống sẽ không hiển thị và thông báo lỗi. 2) Tại bước 2, nếu người dùng bấm “Cancel” thì use case kết thúc. 	
Ngoại lệ: Không có	
Hậu điều kiện: Không có	

Bảng 4.5. Đặc tả use case Upload

4.2.3. Phân tích use case

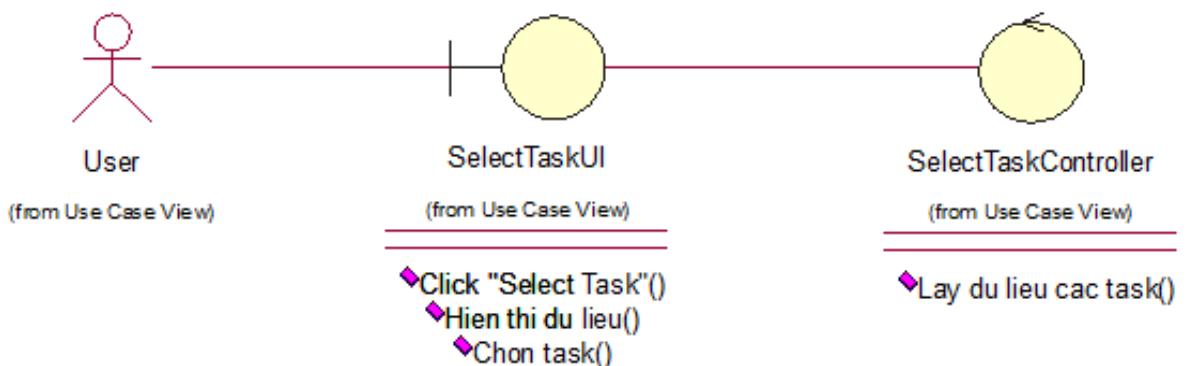
1. Use case Select Task

- Biểu đồ trình tự use case



Hình 4.3. Biểu đồ trình tự use case Select Task

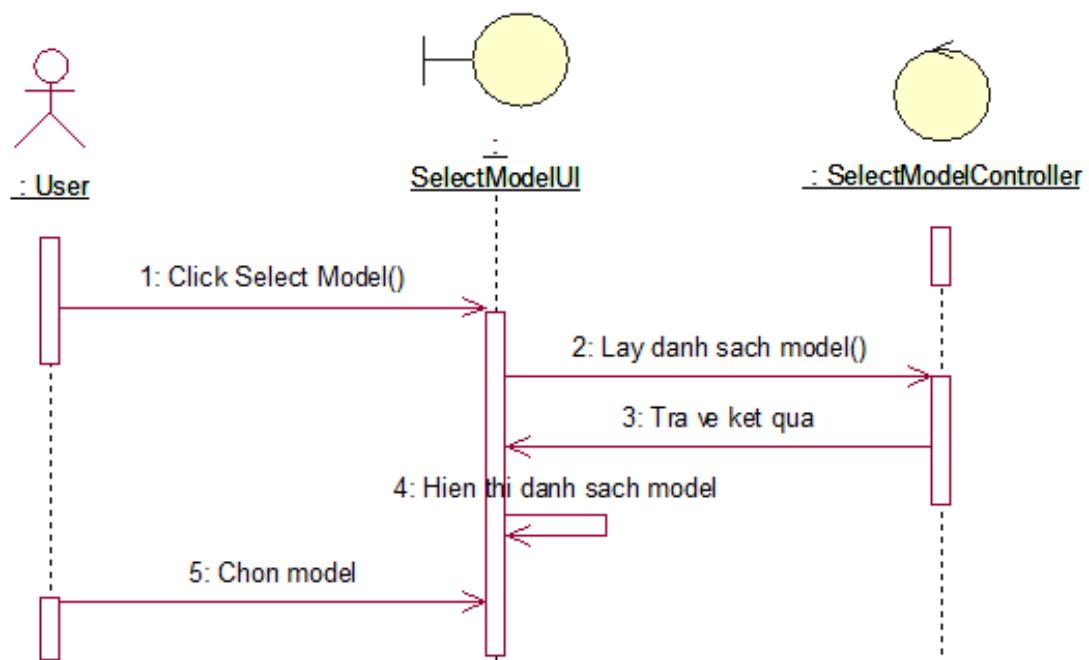
- Biểu đồ lớp phân tích



Hình 4.4. Biểu đồ lớp phân tích use case Select Task

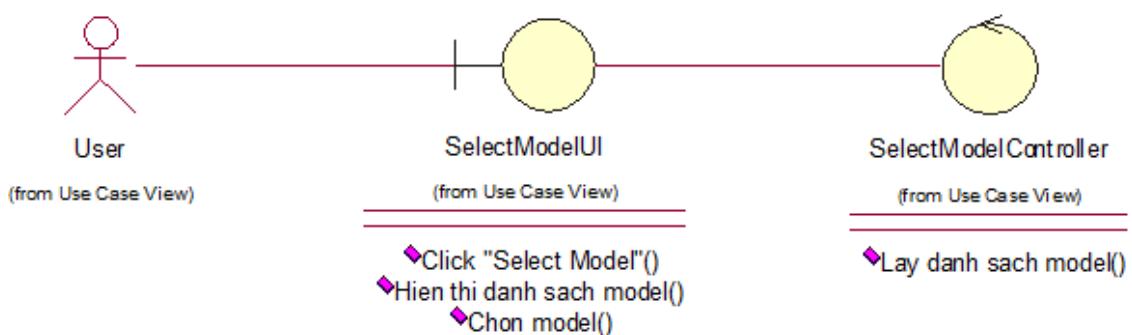
2. Use case Select Model

- Biểu đồ trình tự use case



Hình 4.5. Biểu đồ trình tự use case Select Model

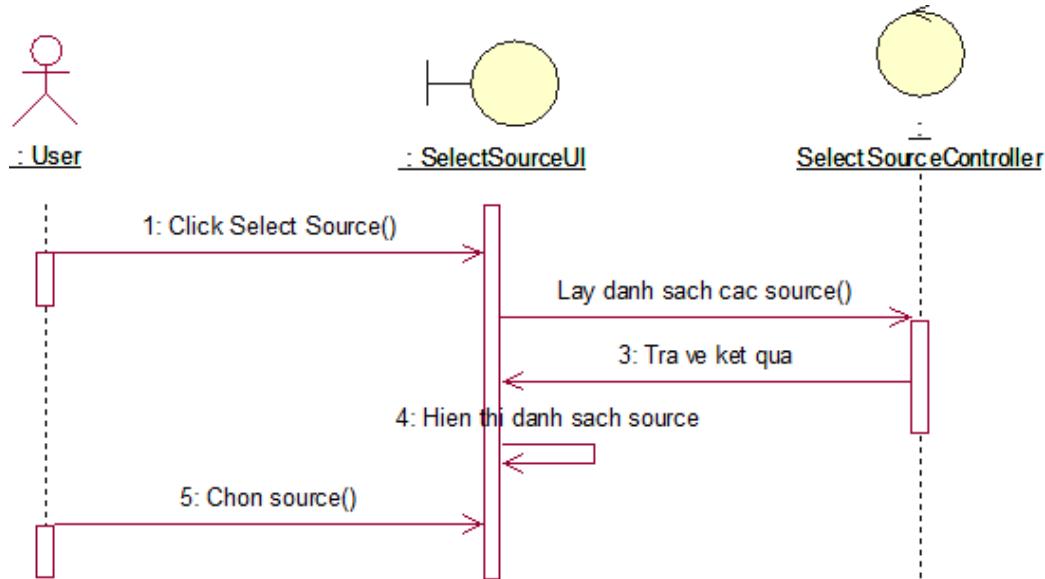
- Biểu đồ lớp phân tích



Hình 4.6. Biểu đồ lớp phân tích use case Select Model

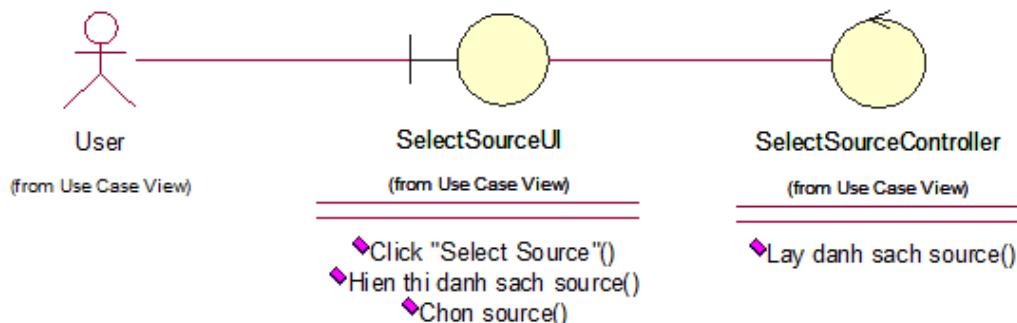
3. Use case Select Source

- Biểu đồ trình tự use case



Hình 4.7. Biểu đồ trình tự use case Select Source

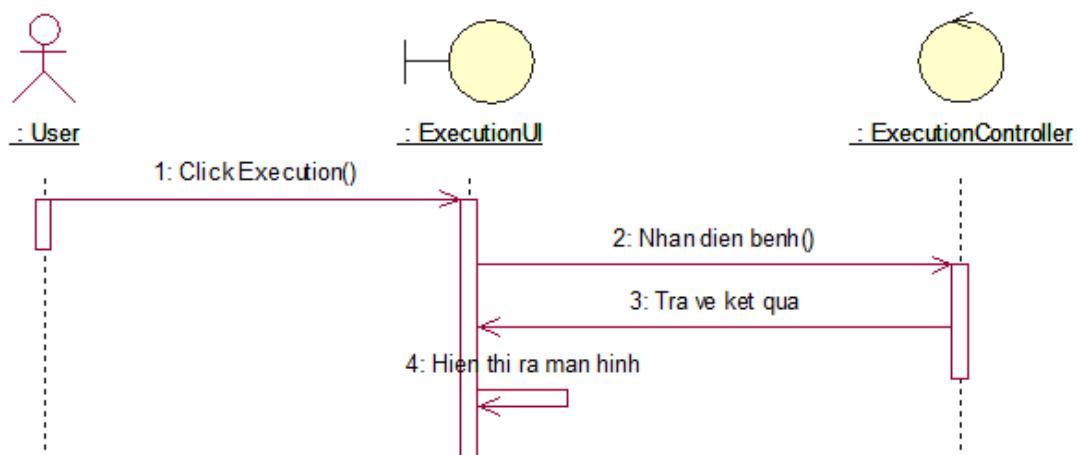
- Biểu đồ lớp phân tích



Hình 4.8. Biểu đồ lớp phân tích use case Select Source

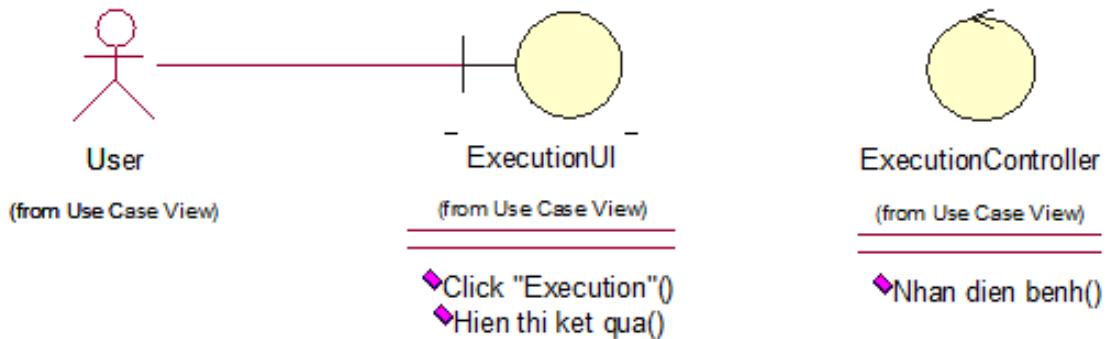
4. Use case Execution

- Biểu đồ trình tự use case



Hình 4.9. Biểu đồ trình tự use case Execution

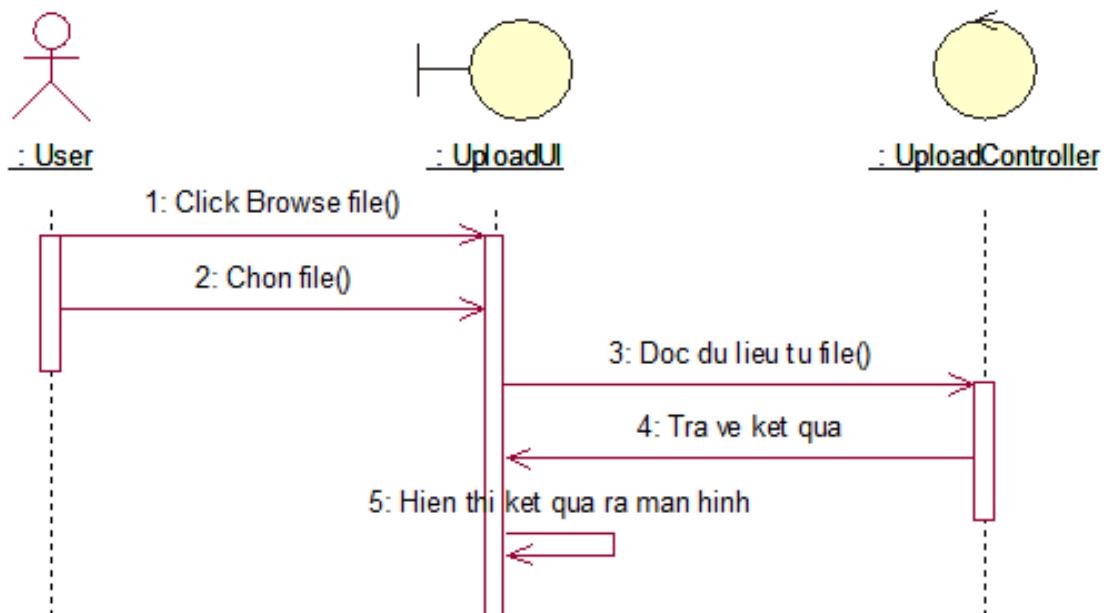
- Biểu đồ lớp phân tích



Hình 4.10. Biểu đồ lớp phân tích use case Execution

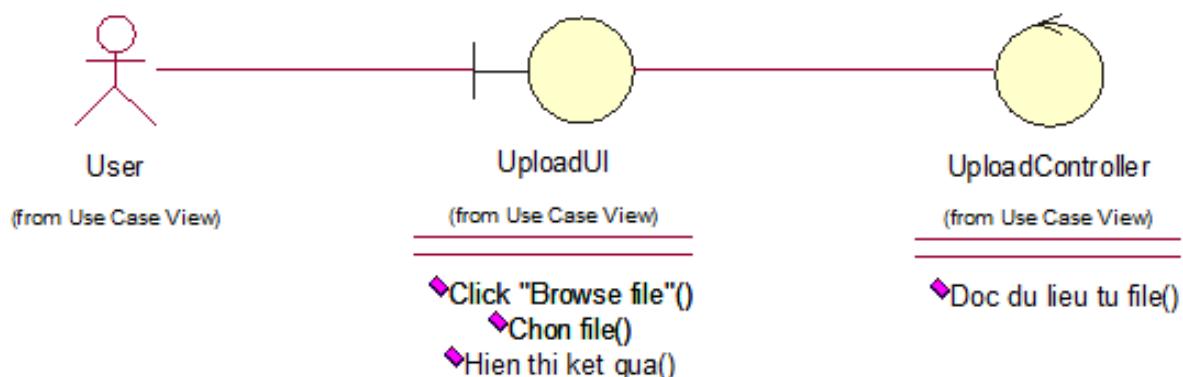
5. Use case Upload

- Biểu đồ trình tự use case



Hình 4.11. Biểu đồ trình tự use case Upload

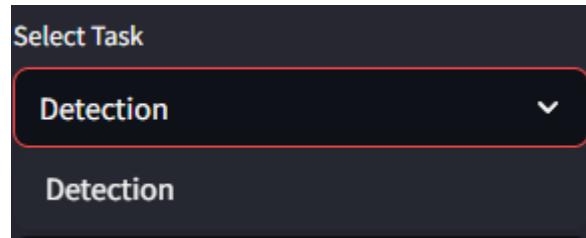
- Biểu đồ lớp phân tích



Hình 4.12. Biểu đồ lớp phân tích use case Upload

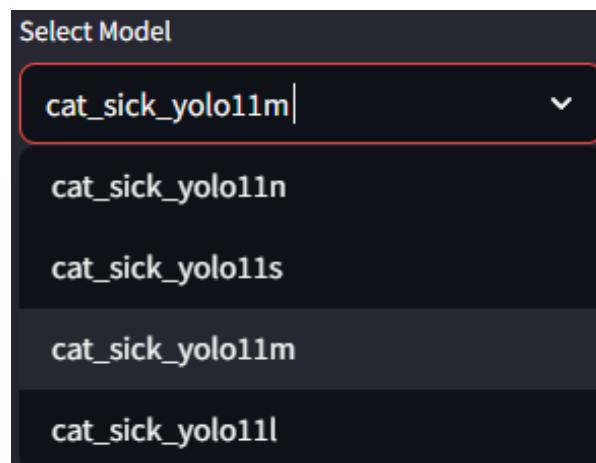
4.3. Các chức năng sau khi cài đặt

- Chức năng “Select Task”



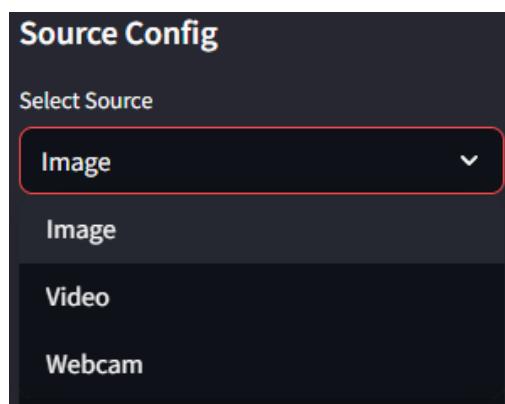
Hình 4.13. Minh họa chức năng Select Task

- Chức năng “Select Model”



Hình 4.14. Minh họa chức năng Select Model

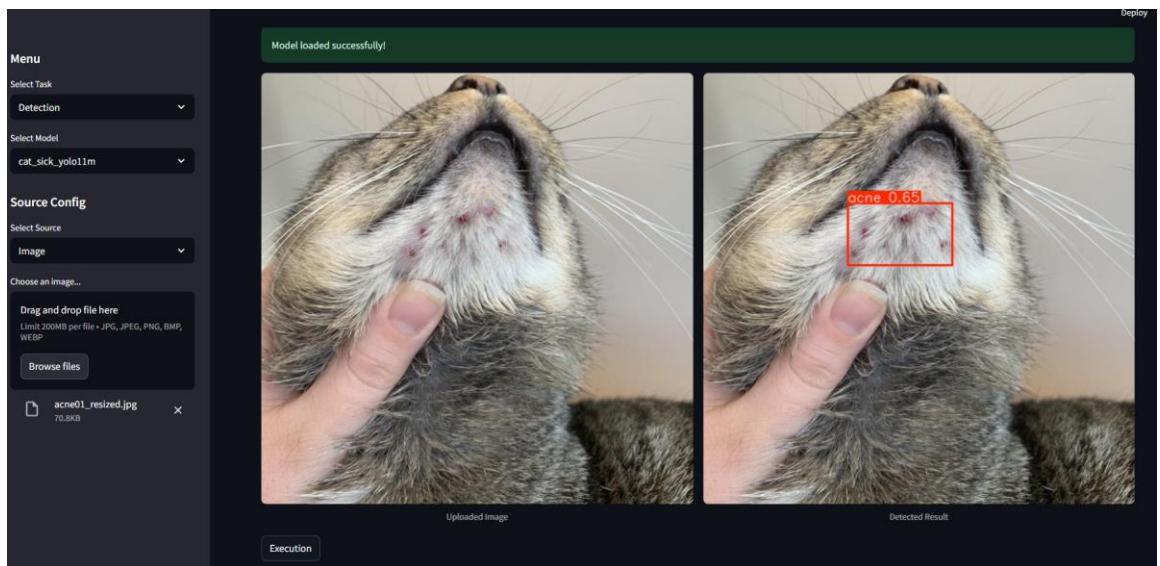
- Chức năng “Select Source”



Hình 4.15. Minh họa chức năng Select Source

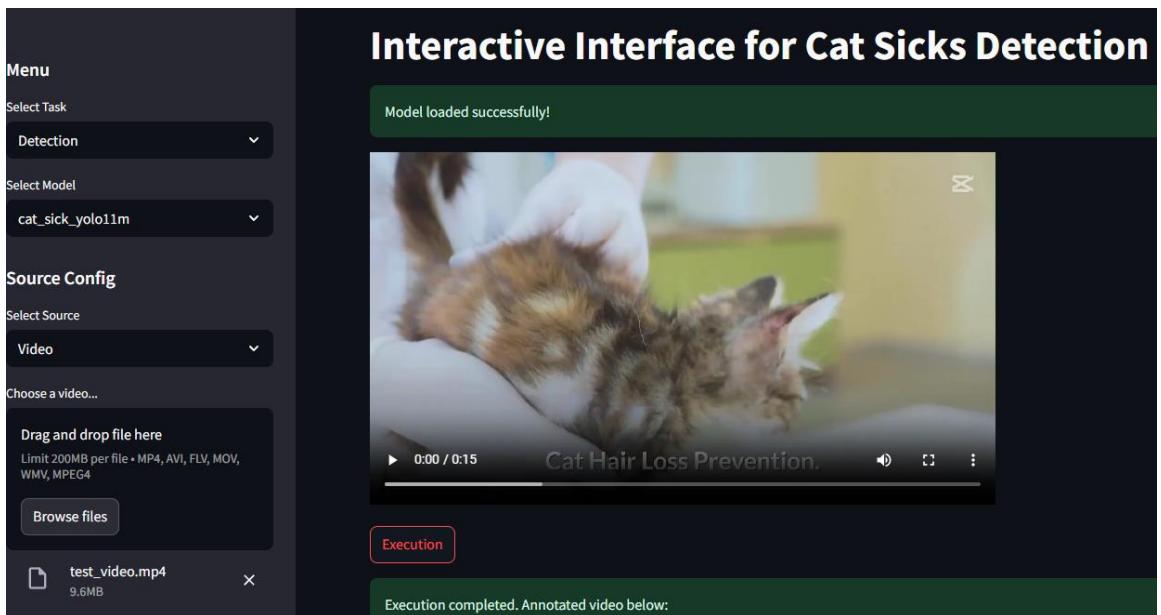
- Chức năng “Execution”

- + Source là Image

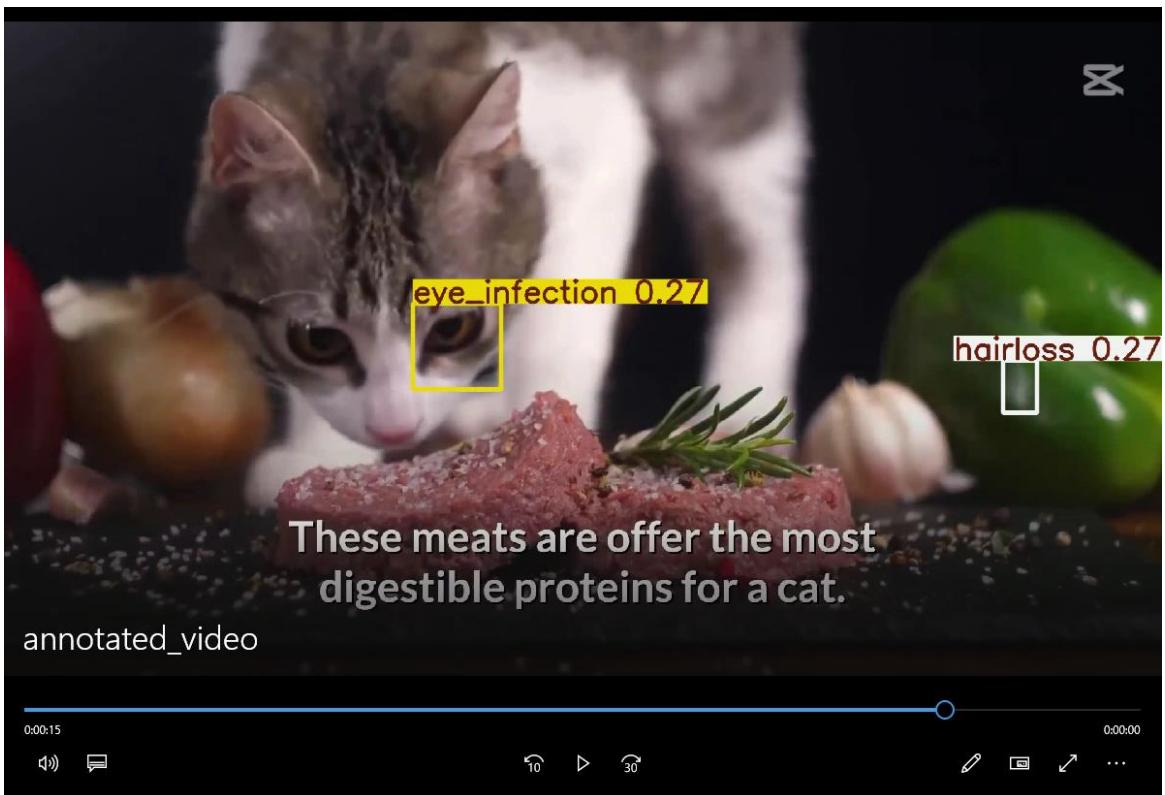


Hình 4.16. Minh họa chức năng Execution với source là Image

+ Source là Video

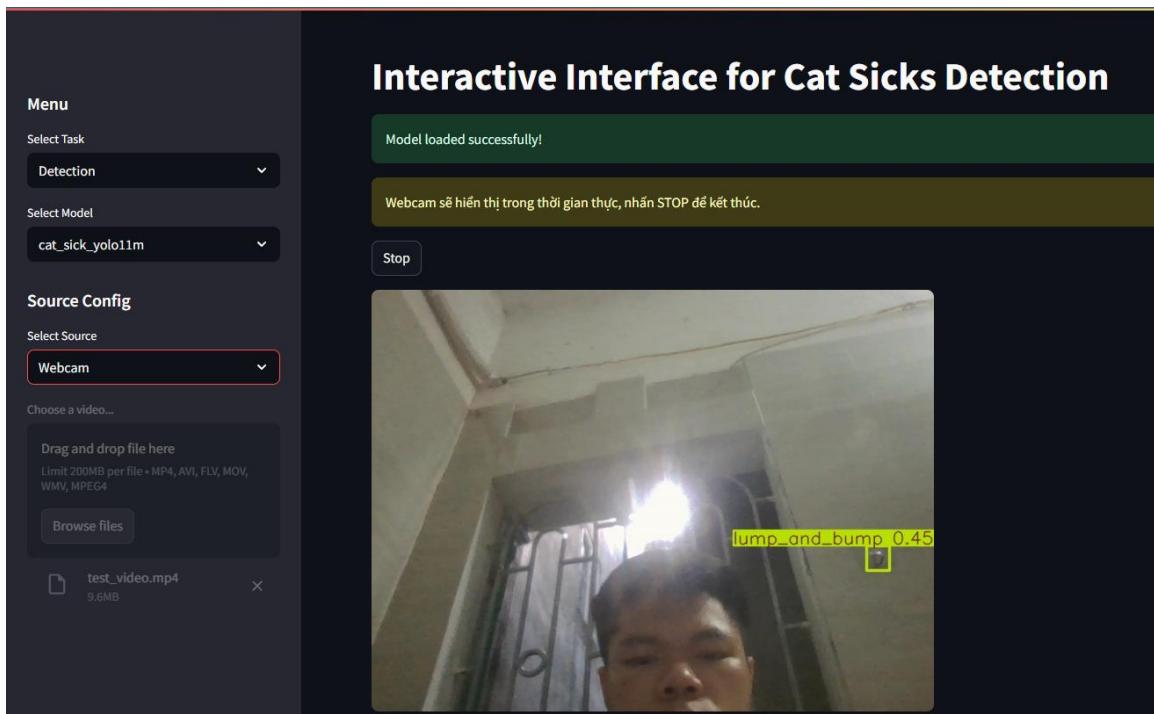


Hình 4.17. Minh họa chức năng Execution với source là Video



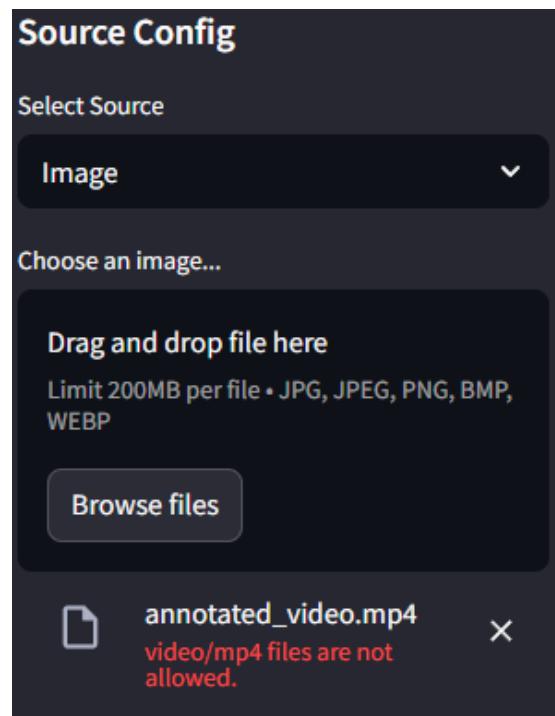
Hình 4.18. Minh họa kết quả thực hiện nhận diện trên video

+ Source là Webcam



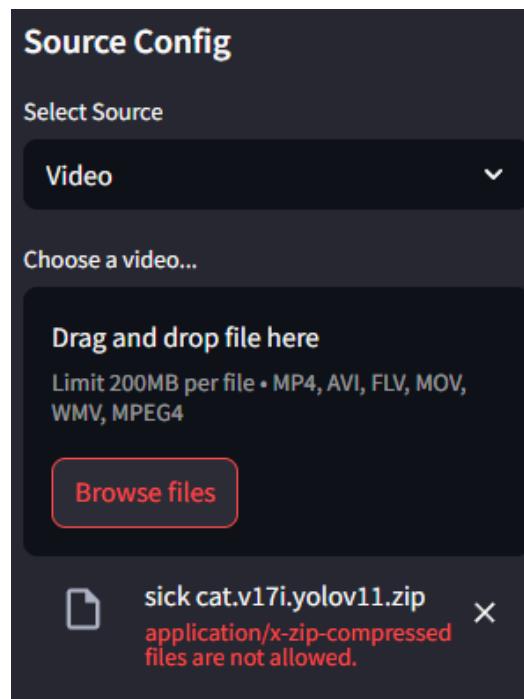
Hình 4.19. Minh họa chức năng Execution với source là Webcam

- Chức năng Upload
 - + Upload sai định dạng Image



Hình 4.20. Kiểm thử định dạng khi upload image

+ Upload sai định dạng Video

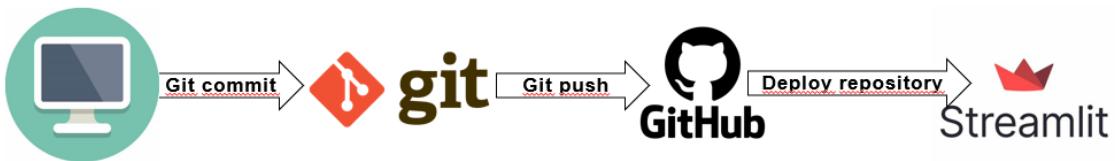


Hình 4.21. Kiểm thử định dạng khi upload video

4.4. Triển khai sản phẩm

Để triển khai sản phẩm lên môi trường thực tế thì cần có cloud và hosting. Đối với ứng dụng được phát triển từ streamlit framework, sự hỗ trợ từ github và streamlit cloud sẽ hỗ trợ việc triển khai sản phẩm.

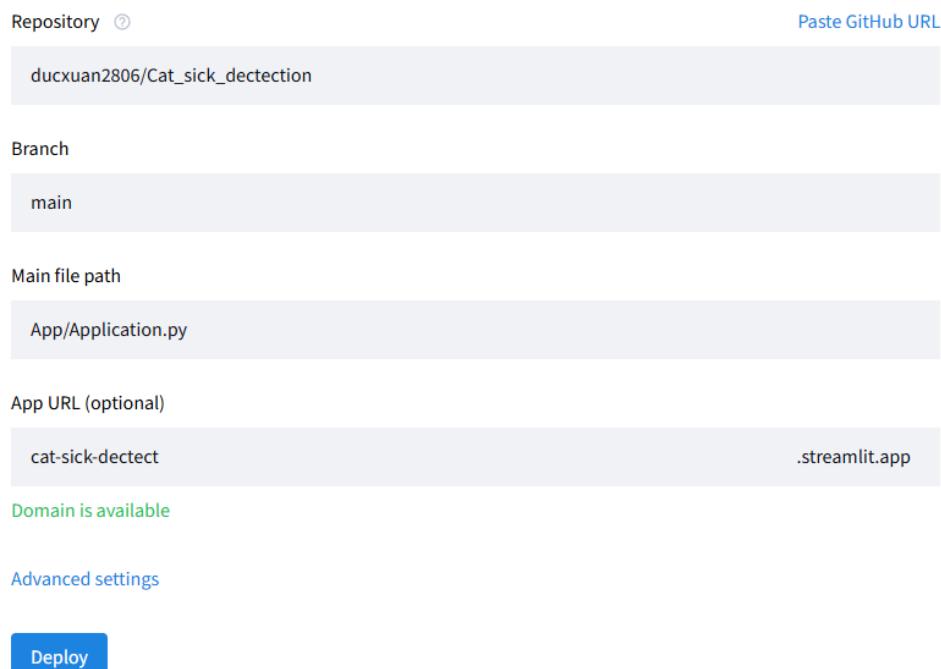
Luồng triển khai sản phẩm được thực hiện như sau:



Hình 4.22. Luồng triển khai với streamlit cloud

Để thực hiện triển khai lên streamlit cloud, trước tiên cần phải kết nối với github và chỉ định repository cần triển khai. Sau đây là ví dụ về cấu hình các thành phần cần thiết trước khi triển khai trên cloud:

Deploy an app



Repository	Paste GitHub URL
ducxuan2806/Cat_sick_dectection	
Branch	
main	
Main file path	
App/Application.py	
App URL (optional)	.streamlit.app
cat-sick-dectect	.streamlit.app
Domain is available	
Advanced settings	
Deploy	

Hình 4.23. Minh họa cấu hình trước triển khai

KẾT LUẬN

Thời gian làm đồ án tốt nghiệp vừa qua là trải nghiệm vô cùng thú vị và đáng giá với bản thân em. Em đã tìm hiểu các kỹ thuật khác nhau của trí tuệ nhân tạo nhằm giải quyết bài toán nhận dạng bệnh trên loài mèo, qua đó không chỉ tích lũy được các kinh nghiệm về chuyên môn mà còn học được cái kĩ năng làm việc độc lập, quản lý thời gian. Đây chắc chắn sẽ là hành trang quý giá trong sự nghiệp tương lai của em.

Em đã tìm hiểu, nghiên cứu, ứng dụng mô hình YOLO cũng như tận dụng các nghiên cứu từng được công bố về nhận dạng bệnh trên loài mèo để hoàn thành đề tài. Ngoài ra, em đã tự xây dựng và thực nghiệm một bộ dữ liệu mới dành cho bài toán nhận dạng bệnh trên mèo. Dưới sự hướng dẫn của thầy Nguyễn Mạnh Cường, em cũng đã tự xây dựng được một mô hình mạng học sâu dựa trên mô hình YOLOv11 thông qua việc huấn luyện và điều chỉnh lại trọng số trong suốt quá trình. Điều này giúp nhận diện bệnh một cách hiệu quả hơn và là một bước quan trọng trước khi xây dựng và triển khai sản phẩm cho người dùng trải nghiệm.

Hệ thống đã cho thấy khả năng nhận dạng với những bệnh cơ bản trên mèo với precision và recall đều ở mức ổn. Với những bệnh dễ nhận diện như bệnh rụng lông thì kết quả trên tập test khá cao. Tuy nhiên do tài nguyên và thời gian hạn chế, nên hệ thống mới chỉ nhận dạng được 5 bệnh cơ bản. Em mong rằng, nghiên cứu của mình có thể đóng góp một phần công sức đối với vấn đề nghiên cứu ứng dụng mô hình YOLO trong việc nhận diện bệnh trên động vật, đặc biệt là với loài mèo.

Trong tương lai, em muốn tiếp tục phát triển và đóng góp thêm dữ liệu cho vấn đề đối với các bệnh trên mèo thông qua việc tìm kiếm phương pháp tiền xử lý hiệu quả và phù hợp. Đặc biệt là em muốn nghiên cứu kĩ hơn trong việc ứng dụng mô hình YOLO cho dữ liệu bệnh y khoa, không chỉ đối với động vật như mèo.

TÀI LIỆU THAM KHẢO

- [1] H. Sahota, “Linkedin,” LinkedIn Corporation, 23 10 2023. [Trực tuyến]. Available: <https://www.linkedin.com/learning/computer-vision-for-data-scientists>. [Truy cập gần nhất ngày 18 tháng 5, 2025].
- [2] “Ultralytics,” Ultralytics Inc, [Trực tuyến]. Available: <https://www.ultralytics.com>. [Truy cập gần nhất ngày 27 tháng 4, 2025].
- [3] Perry Gibson, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani và Manohar Paluri, “Optimizing Grouped Convolutions on Edge Devices,” *ResearchGate*, số 2242494, pp. 31st IEEE International Conference on Application-specific Systems, Architectures and Processors, 2020.
- [4] Anirudha Ghosh, A. Sufian, Farhana Sultana và Amlan Chakrabarti, “Fundamental Concepts of Convolutional Neural Network,” *Researchgate*, số 519-567, p. Recent Trends and Advances in Artificial Intelligence and Internet of Things, 2020.
- [5] Aston Zhang, Zack C. Lipton, Mu Li và Alex J. Smola, *Dive into deep learning*, Cambridge University Press, 2021.
- [6] Du Tran, Lubomir Bourdev, Rob Fergus và Lorenzo Torr, “Learning Spatiotemporal Features with 3D Convolutional Networks,” *arXiv*, tập Computer Vision and Pattern Recognition, số 1412.0767, p. Computer Science, 2014.
- [7] Gudala Lavanya và Sagar Dhanraj Pande, “Enhancing Real-time Object Detection with YOLO Algorithm,” *Researchgate*, số 10.4108/eetiot.4541, pp. Computer vision, Image processing, Object detection, CNN, accuracy, 2023.
- [8] Ian Goodfellow, Yoshua Bengio và Aaron Courville, *Deep Learning*, Massachusetts: MIT Press, 2016.
- [9] Jia-Hao Xu , Jian-Ping Li, Zheng-Ran Zhou và Qing Lv, “A Survey of the Yolo Series of Object Detection Algorithms,” *IEEE*, số 10873779, p.

2024 21st International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), 2024.

- [10] J. Terven và D. Cordova-Esparza, “A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS,” *arXiv*, tập Computer Vision and Pattern Recognition, số 2304.00501, p. Computer Science, 2024.
- [11] Peiyuan Jiang, Daji Ergu, Fangyao Liu và Ying Cai, “A Review of Yolo Algorithm Development,” *Sciencedirect*, số 1066-1073, pp. Yolo, Object Detection, Public Data Analysis, 2022.
- [12] Rahima Khanam và Muhammad Hussain, “YOLOv11: An Overview of the Key Architectural Enhancements,” *arXiv*, tập Computer Vision and Pattern Recognition, số 2410.17725, p. Computer Science, 2024.
- [13] Wang và Chuqi, “A Review on 3D Convolutional Neural Network,” *IEEE*, số 10075760, pp. 2023 IEEE 3rd International Conference on Power, Electronics and Computer Applications (ICPECA), 2023.
- [14] Tausif Diwan, G. Anirudh và Jitendra V. Tembhurne, “Object detection using YOLO: challenges, architectural successors, datasets and applications,” *Springer Nature*, 2022.
- [15] "Streamlit," Snowflake Inc., [Online]. Available: <https://streamlit.io/cloud>. [Truy cập gần nhất ngày 27 tháng 5, 2025].
- [16] Ranjan Sapkota, Rizwan Qureshi, Marco Flores-Caler, Chetan Badjugar, Upesh Nepal, Alwin Poulose, Peter Zeno, Uday Bhanu Prakash Vaddevolu, Sheheryar Khan, Maged Shoman, Hong Yan và Manoj Karkee, “YOLOv12 to Its Genesis: A Decadal and Comprehensive Review of The You Only Look Once (YOLO) Series,” *arXiv*, tập Computer Vision and Pattern Recognition, số 2406.19407, p. Computer Science, 2025.

- [17] Abolfazl Younesi, Mohsen Ansari, Mohammadamin Fazli, Alireza Ejlali, Muhammad Shafique và Jörg Henkel, “A Comprehensive Survey of Convolutions in Deep Learning: Applications, Challenges, and Future Trends,” *arXiv*, số 2402.15490, p. Machine Learning , 2024.
- [18] Areeg Fahad Rasheed và M. Zarkoosh, “YOLOv11 Optimization for Efficient Resource Utilization,” *arXiv*, số 2412.14790, p. Computer Vision and Pattern Recognition, 2024.
- [19] Priyanto Hidayatullah, Nurjannah Syakrani, Muhammad Rizqi Sholahuddin, Trisna Gelar và Refdinal Tubagus, “YOLOv8 to YOLO11: A Comprehensive Architecture In-depth Comparative Review,” *arXiv*, tập Computer Vision and Pattern Recognition, số 2501.13400, p. Computer Science, 2025.