

LỜI CẢM ƠN

Trong quá trình tự mình hoàn thành đồ án môn học 2 với đề tài, tôi đã nhận được rất nhiều sự tư vấn và lời khuyên từ Thầy Huỳnh Thế Thiện. Tôi xin chân thành cảm ơn!

Sinh viên thực hiện đồ án môn học 2
(Ký và ghi rõ họ tên)

Nguyễn Đặng Mai Thy

LỜI CAM ĐOAN

Sinh viên thực hiện đồ án môn học 2 cam đoan đề tài thực hiện dựa vào một số tài liệu trước đó và không sao chép nội dung, kết quả của đồ án khác. Các nội dung tham khảo đã được trích dẫn đầy đủ.

Sinh viên thực hiện đồ án môn học 2
(Ký và ghi rõ họ tên)

Nguyễn Đặng Mai Thy

TÓM TẮT NỘI DUNG

Đề tài "Thiết kế và xây dựng ứng dụng dựa trên thuật toán YOLOv8" tập trung vào việc phát triển một ứng dụng sử dụng thuật toán YOLOv8 để phân loại và phát hiện đối tượng trong ảnh và video. Mục tiêu của đề tài là xây dựng một ứng dụng có giao diện người dùng dễ sử dụng, nâng cao hiệu suất và độ chính xác, hỗ trợ đa nền tảng và đa nguồn dữ liệu, cũng như cung cấp tính năng đa nhiệm.

Trong đề tài này, một trong những mục tiêu quan trọng là xây dựng một giao diện người dùng thân thiện và trực quan. Giao diện này sẽ giúp người dùng tải lên dễ dàng ảnh hoặc video và thực hiện các nhiệm vụ phân loại và phát hiện đối tượng một cách dễ dàng. Giao diện nên được thiết kế sao cho người dùng có thể tùy chỉnh các tùy chọn và dễ dàng hiển thị kết quả phân loại và đánh dấu đối tượng trên hình ảnh hoặc video.

Một mục tiêu quan trọng khác của đề tài là nâng cao hiệu suất và độ chính xác của thuật toán YOLOv8. YOLOv8 đã được phát triển để cải thiện hiệu suất và độ chính xác của việc nhận dạng và phân loại đối tượng. Vì vậy, trong ứng dụng YOLOv8 Full Tasks, mục tiêu là tận dụng tối đa khả năng này để cung cấp cho người dùng một công cụ mạnh mẽ để phát hiện và phân loại đối tượng trong ảnh và video. Điều này đòi hỏi tối ưu hóa thuật toán, tăng cường quá trình huấn luyện mô hình và tăng cường khả năng xử lý của ứng dụng để đảm bảo rằng nó có thể nhận dạng và phân loại đối tượng một cách chính xác và nhanh chóng.

Hỗ trợ đa nền tảng và đa nguồn dữ liệu cũng là một mục tiêu quan trọng. Điều này đảm bảo rằng người dùng có thể sử dụng ứng dụng trên nhiều thiết bị và hệ điều hành khác nhau. Đồng thời, việc hỗ trợ đa nguồn dữ liệu, bao gồm ảnh và video từ nhiều nguồn khác nhau, cũng là một yếu tố quan trọng. Điều này cho phép người dùng làm việc với các tệp dữ liệu từ nhiều nguồn và tận dụng ứng dụng trên các nền tảng khác nhau mà không gặp rào cản.

Cuối cùng, mục tiêu cuối cùng của ứng dụng YOLOv8 Full Tasks là cung cấp tính năng đa nhiệm. Điều này cho phép người dùng thực hiện đồng thời nhiều nhiệm vụ như phát hiện đối tượng, phân loại và các tác vụ khác trong cùng một ứng dụng. Điều này

tạo ra sự linh hoạt và tiện ích cho người dùng, cho phép họ thực hiện nhiều tác vụ liên quan đến phân loại và phát hiện đối tượng một cách liền mạch trong một môi trường ứng dụng duy nhất.

Tóm lại, đề tài "Thiết kế và xây dựng ứng dụng dựa trên thuật toán YOLOv8" tập trung vào việc phát triển một ứng dụng sử dụng thuật toán YOLOv8 để phân loại và phát hiện đối tượng trong ảnh và video. Mục tiêu của đề tài bao gồm xây dựng một giao diện người dùng dễ sử dụng, nâng cao hiệu suất và độ chính xác của thuật toán, hỗ trợ đa nền tảng và đa nguồn dữ liệu, cũng như cung cấp tính năng đa nhiệm. Đề tài này đóng góp vào việc tạo ra một công cụ mạnh mẽ và tiện ích cho việc phân loại và phát hiện đối tượng trong ảnh và video, đồng thời đáp ứng yêu cầu đa dạng của người dùng trên nhiều nền tảng và tình huống sử dụng khác nhau.

Mục lục

1 TỔNG QUAN	1
1.1 Giới thiệu	1
1.2 Mục tiêu	3
1.3 TÌNH HÌNH NGHIÊN CỨU	3
1.3.1 Tình hình nghiên cứu ngoài nước:	4
1.4 Phương pháp nghiên cứu	4
1.5 Bố cục nội dung	5
2 CƠ SỞ LÝ THUYẾT	6
2.1 TỔNG QUAN VỀ TRÍ TUỆ NHÂN TẠO	6
2.1.1 Trí tuệ nhân tạo là gì?	6
2.1.2 Deep learning và Machine Learning	7
2.1.3 Ứng dụng trí tuệ nhân tạo	9
2.2 TỔNG QUAN VỀ THỊ GIÁC MÁY	9
2.2.1 Thị giác máy tính là gì?	9
2.2.2 Thị giác máy tính hoạt động như thế nào?	11
2.2.3 Một vài ví dụ về các tác vụ Thị giác máy tính	11
2.3 TỔNG QUAN VỀ MẠNG NƠ-RON	12
2.3.1 Mạng nơ-ron nhân tạo	13
2.3.2 Mạng nơ-ron tích chập	15
3 THUẬT TOÁN VÀ KỸ THUẬT	17
3.1 TỔNG QUAN VỀ YOLOV8	17

3.1.1	YOLOv8 là gì?	17
3.1.2	YOLO phát triển thành YOLOv8 như thế nào?	17
3.1.3	Tại sao nên sử dụng YOLOv8?	18
3.1.4	Kiến trúc YOLOv8: A Deep Dive	19
3.1.5	Cải thiện độ chính xác của YOLOv8	24
3.2	CÁC TASKS SỬ DỤNG TRONG YOLOV8	26
3.2.1	Object Detection	27
3.2.2	Instance Segmentation	28
3.2.3	Pose Estimation	28
3.2.4	Image Classification	29
3.2.5	Pose Estimation	29
4	CÔNG CỤ SỬ DỤNG	30
4.1	PHẦN MỀM STREAMLIT	30
4.1.1	Streamlit là gì?	30
4.1.2	Tại sao nên sử dụng Streamlit?	30
4.2	TRANG WEB OPEN IMAGES DATASET V7	32
4.3	TRANG WEB CVAT	33
5	THƯ VIỆN THU THẬP DỮ LIỆU	35
5.1	THƯ VIỆN OIDV4_TOOLKIT	35
5.2	THƯ VIỆN PYTUBE	36
5.3	THƯ VIỆN BING_IMAGE_DOWNLOADER	36
6	THIẾT KẾ DỮ LIỆU VÀ XÂY DỰNG MÔ HÌNH YOLOV8	38
6.1	THU THẬP DỮ LIỆU	38
6.1.1	OIDV4_TOOLKIT	38
6.1.2	BING_IMAGE_DOWNLOADER	39
6.1.3	PYTUBE	40
6.2	XỬ LÝ DỮ LIỆU	41
6.2.1	Object Detection	41

6.2.2	Instance Segmentation	43
6.2.3	Pose Estimation	43
6.2.4	Image Classification	44
6.3	CẤU TRÚC TẬP DỮ LIỆU	44
7	XÂY DỰNG MÔ HÌNH YOLOV8	46
7.1	GPU Check	46
7.2	Object Detection	46
7.2.1	config.yaml	46
7.2.2	train.py	47
7.2.3	confusion_matrix	47
7.3	Instance Segmentation	47
7.3.1	config.yaml	47
7.3.2	train.py	48
7.3.3	labels_correlogram	49
7.4	Pose Estimation	49
7.4.1	config.yaml	49
7.4.2	train.py	49
7.5	Image Classification	50
7.5.1	train.py	50
7.5.2	confusion_matrix	50
7.5.3	results	50
8	KIỂM TRA MÔ HÌNH ĐÃ XÂY DỰNG	52
8.1	Object Detection	52
8.2	Instance Segmentation	53
8.3	Pose Estimation	54
8.4	Image Classification	54
9	THIẾT KẾ VÀ XÂY DỰNG ỨNG DỤNG YOLOV8	56
9.1	Thiết kế giao diện	56

9.2 Cấu hình ứng dụng YOLOv8	60
9.3 Xây dựng ứng dụng YOLOv8	63
10 KẾT QUẢ, PHÂN TÍCH ĐÁNH GIÁ VÀ ĐỊNH HƯỚNG PHÁT TRIỂN TRONG TƯƠNG LAI	69
10.1 Kết quả sản phẩm ứng dụng YOLOv8	69
10.1.1 Giao diện	69
10.1.2 Kết quả	69
10.2 Phân tích và đánh giá kết quả	69
10.3 Định hướng phát triển trong tương lai	72

Danh sách hình vẽ

2.1	Hình ảnh minh họa AI	7
2.2	Sơ đồ mạng Deep neural network	8
2.3	Hình ảnh minh họa Computer vision	10
2.4	Hình ảnh minh họa Bộ não xử lý thông tin đầu vào và chuyển đổi nó thành thứ mà con người có thể hiểu được.	13
2.5	Hình ảnh minh họa ANN đơn giản	14
2.6	Hình ảnh minh họa CNN trong phân loại hình ảnh	16
3.1	Kiến trúc YOLOv8, trực quan hóa được thực hiện bởi người dùng GitHub RangeKing	20
3.2	Hình dung hộp neo trong YOLO	20
3.3	Đầu phát hiện của YOLOv5, được hiển thị bằng netron.app	21
3.4	Đầu phát hiện của YOLOv8, được hiển thị bằng netron.app	22
3.5	Mô-đun YOLOv8 C2f mới	23
3.6	Khảm tăng cường hình ảnh bàn cờ	23
3.7	Đánh giá YOLOv8 COCO	24
3.8	YOLOs mAP@.50 so với RF100	25
3.9	YOLO trung bình mAP@.50 so với các loại RF100	26
3.10	Các tasks có trong YOLOv8	27
3.11	Hình minh họa Object Detection	27
3.12	Hình minh họa Instance Segmentation	28
3.13	Hình minh họa Pose Estimation	28
3.14	Hình minh họa Image Classification	29

4.1	Hình minh họa phần mềm Streamlit	31
4.2	Hình minh họa trang web Open Images Dataset v7	32
4.3	Hình minh họa trang web CVAT	34
7.1	Hình minh họa confusion_matrix	47
7.2	Hình minh họa labels_correlogram	48
7.3	Hình minh họa confusion_matrix	50
7.4	Hình minh họa results	51
10.2	Giao diện ứng dụng YOLOv8	70
10.4	Kết quả	72
10.5	Kết quả	72
10.6	Kết quả	73
10.7	Kết quả	73
10.8	Kết quả	74
10.9	Kết quả	74
10.10	Kết quả	75
10.11	Kết quả	75

Danh mục các từ viết tắt

Dưới đây là danh mục các từ viết tắt được sử dụng trong luận văn.

Các từ viết tắt

Định nghĩa

DL	Deep Learning
YOLO	You Only Look Once
AI	Artificial Intelligence

Chương 1

TỔNG QUAN

1.1 Giới thiệu

YOLOv8 (You Only Look Once version 8) là một thuật toán nhận dạng đối tượng và phát hiện vật thể trong ảnh và video. Được phát triển dựa trên các phiên bản trước của YOLO, YOLOv8 là một phiên bản cải tiến với hiệu suất cao hơn và khả năng phát hiện đa nhiệm.

YOLOv8 có thể thực hiện nhiều tác vụ như nhận dạng vật thể, phát hiện khuôn mặt, phát hiện hành vi và phân loại đối tượng. Nó sử dụng một mạng nơ-ron tích chập (CNN) để trích xuất đặc trưng từ ảnh đầu vào và sau đó áp dụng các hộp giới hạn (bounding boxes) để xác định vị trí và loại đối tượng.

Các bước chính trong quá trình hoạt động của YOLOv8 bao gồm:

- **Trích xuất đặc trưng:** Mạng CNN được sử dụng để trích xuất các đặc trưng từ ảnh đầu vào. Các lớp tích chập được sử dụng để nhận biết các đặc điểm quan trọng của đối tượng.
- **Phát hiện đối tượng:** Sau khi trích xuất đặc trưng, YOLOv8 sử dụng các hộp giới hạn để xác định vị trí và loại đối tượng có mặt trong ảnh. Các hộp giới hạn này được dự đoán thông qua các lớp đầu ra của mạng.
- **Phân loại đối tượng:** YOLOv8 cũng có khả năng phân loại đối tượng, tức là xác định loại đối tượng cụ thể nếu nhận dạng được. Điều này đòi hỏi mô hình được huấn luyện trước với dữ liệu được gắn nhãn chính xác để nhận biết các lớp đối tượng khác

nhau.

- **Đa nhiệm:** YOLOv8 có khả năng thực hiện đồng thời nhiều tác vụ như nhận dạng đối tượng, phát hiện khuôn mặt và phân loại đối tượng. Điều này cho phép nó đáp ứng nhiều yêu cầu khác nhau trong việc phân tích hình ảnh và video.

Với hiệu suất và tính linh hoạt cao, YOLOv8 đã trở thành một trong những thuật toán nhận dạng đối tượng phổ biến và mạnh mẽ trong lĩnh vực thị giác máy tính. Nó có thể được áp dụng trong nhiều lĩnh vực như xe tự hành, giám sát an ninh, nhận dạng khuôn mặt, và nhiều ứng dụng hỗ trợ khác.

Ứng dụng YOLOv8 Full Tasks là một ứng dụng phân loại và phát hiện vật thể sử dụng thuật toán YOLOv8. Nó được thiết kế để nhận diện và phân loại đối tượng trong ảnh và video với khả năng đa nhiệm.

Với ứng dụng YOLOv8 Full Tasks, bạn có thể thực hiện các nhiệm vụ sau:

- **Phát hiện đối tượng:** Ứng dụng cho phép bạn xác định và phát hiện các vật thể trong ảnh và video. Bạn có thể tải lên một ảnh hoặc nhập video từ nguồn dữ liệu và YOLOv8 sẽ sử dụng mạng nơ-ron tích chập để tìm kiếm và đánh dấu các đối tượng có mặt trong hình ảnh.
- **Phân loại đối tượng:** YOLOv8 Full Tasks cũng cung cấp khả năng phân loại đối tượng. Sau khi phát hiện được đối tượng, ứng dụng sẽ áp dụng mô hình phân loại để xác định loại đối tượng cụ thể và hiển thị kết quả tương ứng.
- **Đa nhiệm:** Một trong những điểm mạnh của YOLOv8 Full Tasks là khả năng thực hiện đồng thời nhiều nhiệm vụ. Bạn có thể thực hiện phát hiện đối tượng, phân loại và thậm chí phát hiện khuôn mặt trong cùng một ứng dụng.
- **Giao diện người dùng đơn giản:** Ứng dụng được thiết kế với giao diện người dùng thân thiện và dễ sử dụng. Bạn có thể tải lên ảnh hoặc video, chọn các nhiệm vụ bạn muốn thực hiện và xem kết quả ngay lập tức.

Ứng dụng YOLOv8 Full Tasks có thể được áp dụng trong nhiều lĩnh vực như giám sát an ninh, xử lý ảnh y tế, xe tự hành, nhận dạng khuôn mặt và nhiều ứng dụng khác mà yêu cầu phát hiện và phân loại đối tượng trong thời gian thực.

1.2 Mục tiêu

Đối với đề tài trên, nhóm chúng tôi đã đề ra 4 mục tiêu chính:

Thứ nhất: Thiết kế một hệ thống định vị xe đưa rước học sinh để quản lý, giám sát và theo dõi vị trí của các xe đưa rước trong thời gian thực. Hệ thống này sẽ sử dụng công nghệ định vị GPS và các thông tin tương ứng để xác định vị trí chính xác của xe và cung cấp thông tin này cho người quản lý.

Thứ hai: Thiết kế một hệ thống nhận diện để ghi nhận việc điểm danh của học sinh khi lên xe và xuống xe. Hệ thống này sử dụng các công nghệ nhận diện khuôn mặt để xác định danh tính của học sinh và ghi nhận thời gian điểm danh.

Thứ ba: Tích hợp hệ thống định vị xe đưa rước và hệ thống nhận diện điểm danh vào một hệ thống quản lý tổng thể. Hệ thống này sẽ cung cấp giao diện quản lý để người quản lý có thể theo dõi vị trí của các xe, kiểm tra thông tin điểm danh và tạo báo cáo tổng hợp về việc vận chuyển học sinh.

Thứ tư: Tăng cường an ninh và sự an tâm cho phụ huynh: Mục tiêu cuối cùng là tạo một môi trường an toàn và đáng tin cậy cho học sinh và phụ huynh. Hệ thống định vị xe đưa rước và nhận diện điểm danh sẽ giúp phụ huynh an tâm hơn về việc con em mình được đưa đi và đón về trường một cách an toàn và đúng giờ.

1.3 TÌNH HÌNH NGHIÊN CỨU

Trong phần tình hình nghiên cứu chúng tôi muốn trình bày về các công trình đã được nghiên cứu có thể hỗ trợ cho việc xây dựng và phát triển đề tài từ đó đưa ra được phương án thực hiện tốt, tối ưu nhất cho đề tài. Góp phần cho việc có thể hoàn thành hoàn chỉnh cũng như chỉnh chu nhất cho toàn bộ hệ thống.

1.3.1 Tình hình nghiên cứu ngoài nước:

1.4 Phương pháp nghiên cứu

Để đạt được mục tiêu của đề tài, tôi sử dụng các phương pháp thu thập số liệu, thực nghiệm và phân tích tổng kết kinh nghiệm.

- Phương pháp thu thập số liệu:

- Sử dụng phương pháp quan sát: Tôi tiến hành quan sát trực tiếp các tình huống và thu thập số liệu về các đối tượng trong ảnh. Tôi ghi lại thông tin về vị trí, kích thước và đặc điểm của các đối tượng được phát hiện bằng YOLOv8.
- Tiến hành cuộc phỏng vấn: Tôi tiến hành cuộc phỏng vấn với người sử dụng ứng dụng YOLOv8 để thu thập ý kiến và phản hồi về trải nghiệm của họ. Tôi tìm hiểu về việc sử dụng các tính năng cụ thể, khả năng phát hiện và nhận dạng đối tượng, và hiệu suất của ứng dụng.

- Phương pháp thực nghiệm:

- Thiết kế và triển khai hệ thống: Tôi tiến hành thiết kế và triển khai hệ thống ứng dụng YOLOv8 với tất cả các tasks, bao gồm Instance Segmentation, Pose Estimation và Tracking Object. Tôi xây dựng một môi trường thử nghiệm chính xác để đảm bảo tính ổn định và hiệu suất của hệ thống.
- Tiến hành thử nghiệm: Tôi thực hiện các bài kiểm tra và thử nghiệm hệ thống ứng dụng YOLOv8 trên một tập dữ liệu đa dạng và phong phú. Tôi đánh giá khả năng phát hiện, nhận dạng và theo dõi các đối tượng trong ảnh. Tôi ghi lại kết quả và đánh giá hiệu suất của hệ thống.

- Phương pháp phân tích tổng kết kinh nghiệm:

- Đánh giá hiệu quả: Tôi tiến hành đánh giá hiệu quả của ứng dụng YOLOv8 dựa trên kết quả thử nghiệm. Tôi xem xét độ chính xác, độ tin cậy và tốc độ xử lý của hệ thống để đánh giá tính hiệu quả của nó trong việc phát hiện, nhận dạng và theo dõi các đối tượng.
- Phân tích dữ liệu: Tôi phân tích các dữ liệu thu thập được từ quá trình thử nghiệm và phân tích. Tôi xem xét các mẫu dữ liệu, xu hướng và biểu đồ để hiểu rõ hơn về

hiệu suất và khả năng của ứng dụng YOLOv8 trong từng tasks.

- Tổng kết kinh nghiệm: Dựa trên kết quả phân tích, tôi tổng kết kinh nghiệm và rút ra những kết luận về việc sử dụng YOLOv8 trong ứng dụng thực tế. Tôi đề xuất các cải tiến và khuyến nghị để nâng cao hiệu suất và tính ứng dụng của hệ thống YOLOv8 trong các tasks đã được thực hiện.

Trong quá trình này, tôi sử dụng phương pháp thu thập số liệu, thực nghiệm và phân tích tổng kết kinh nghiệm để đảm bảo rằng việc đánh giá và cải tiến ứng dụng YOLOv8 trong các tasks đã được thực hiện được thực hiện một cách toàn diện và có căn cứ dữ liệu đáng tin cậy.

1.5 Bố cục nội dung

Báo cáo đồ án môn học 1 của tôi sẽ bao gồm 5 chương:

- Chương 1: Tổng quan
- Chương 2: Cơ sở lý thuyết
- Chương 3: Thuật toán sử dụng
- Chương 4: Công cụ sử dụng
- Chương 5: Thư viện sử dụng
- Chương 6: Thiết kế dữ liệu và xây dựng mô hình YOLOv8
- Chương 7: Thiết kế và xây dựng ứng dụng YOLOv8
- Chương 8: Kết quả, phân tích và định hướng phát triển trong tương lai

Chương 2

CƠ SỞ LÝ THUYẾT

2.1 TỔNG QUAN VỀ TRÍ TUỆ NHÂN TẠO

2.1.1 Trí tuệ nhân tạo là gì?

Stuart Russell và Peter Norvig đã từng xuất bản "Trí tuệ nhân tạo: Cách tiếp cận hiện đại trở thành một trong những sách giáo khoa hàng đầu trong nghiên cứu về AI". Trong đó, họ đi sâu vào bốn mục tiêu hoặc định nghĩa tiềm năng về AI, giúp phân biệt các hệ thống máy tính trên cơ sở tính hợp lý và suy nghĩ so với hành động:

Cách tiếp cận của con người:

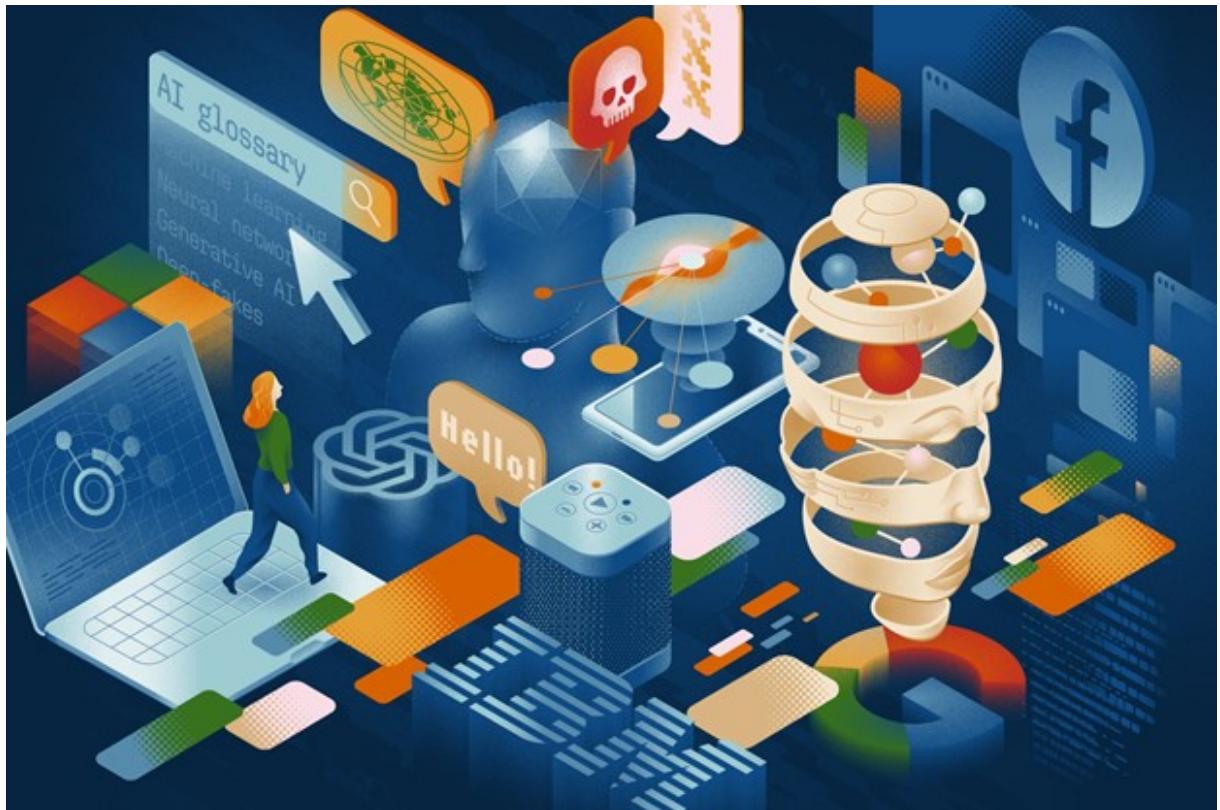
- Hệ thống suy nghĩ như con người
- Hệ thống hoạt động như con người

Cách tiếp cận lý tưởng:

- Hệ thống suy nghĩ hợp lý
- Hệ thống hành động hợp lý

Ở dạng đơn giản nhất, trí tuệ nhân tạo là một lĩnh vực, kết hợp khoa học máy tính và bộ dữ liệu mạnh mẽ, để cho phép giải quyết vấn đề. Nó cũng bao gồm các lĩnh vực phụ của học máy và học sâu, thường được đề cập cùng với trí tuệ nhân tạo. Các ngành này bao gồm các thuật toán AI tìm cách tạo ra các hệ thống chuyên gia đưa ra dự đoán hoặc phân loại dựa trên dữ liệu đầu vào.

Trong những năm qua, trí tuệ nhân tạo đã trải qua nhiều chu kỳ cường điệu, nhưng



Hình 2.1: Hình ảnh minh họa AI

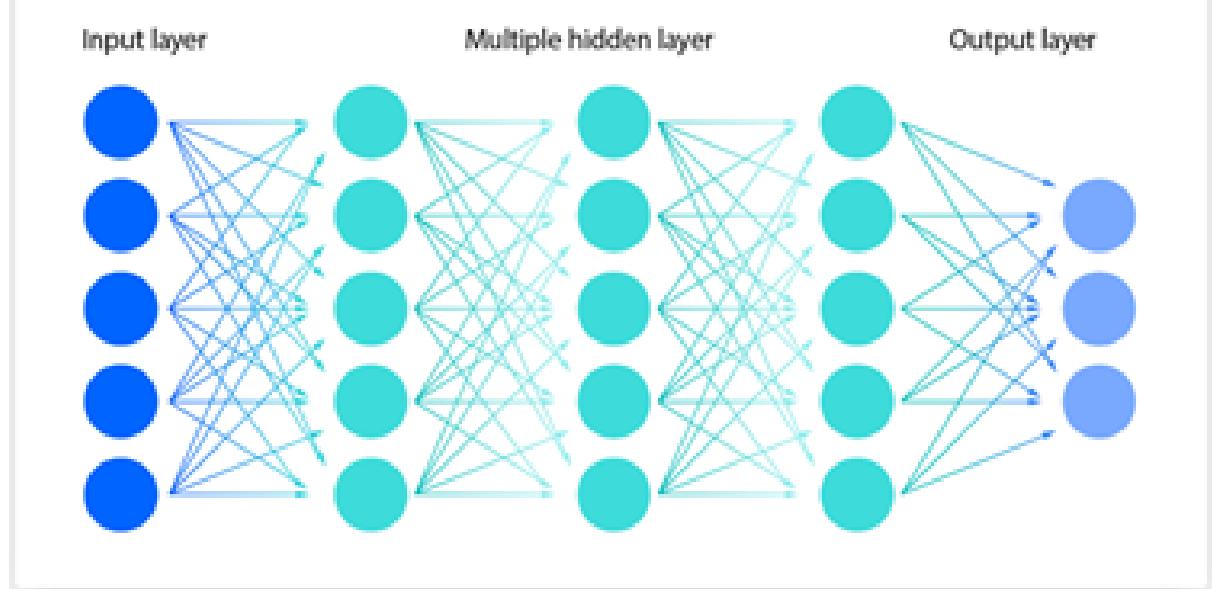
ngay cả với những người hoài nghi, việc phát hành ChatGPT của OpenAI dường như đánh dấu một bước ngoặt. Lần cuối cùng AI tạo ra xuất hiện lớn như vậy, những đột phá là trong thị giác máy tính, nhưng bây giờ bước nhảy vọt là trong xử lý ngôn ngữ tự nhiên. Và nó không chỉ là ngôn ngữ: Các mô hình tạo cũng có thể học ngữ pháp của mã phần mềm, phân tử, hình ảnh tự nhiên và nhiều loại dữ liệu khác.

Các ứng dụng cho công nghệ này đang phát triển mỗi ngày và chúng tôi chỉ mới bắt đầu khám phá các khả năng. Nhưng khi sự cường điệu xung quanh việc sử dụng AI trong kinh doanh cất cánh, các cuộc trò chuyện xung quanh đạo đức trở nên cực kỳ quan trọng. Để đọc thêm về vị trí của IBM trong cuộc trò chuyện xung quanh đạo đức AI, hãy đọc thêm tại [đây](#).

2.1.2 Deep learning và Machine Learning

Vì học sâu và học máy có xu hướng được sử dụng thay thế cho nhau, nên đáng chú ý là các sắc thái giữa hai loại. Như đã đề cập ở trên, cả deep learning và machine learning đều là các lĩnh vực con của trí tuệ nhân tạo và deep learning thực sự là một lĩnh vực con

Deep neural network



Hình 2.2: Sơ đồ mạng Deep neural network

của machine learning. Học sâu thực sự bao gồm các mạng lưới thần kinh. "Sâu" trong học sâu đề cập đến một mạng lưới thần kinh bao gồm hơn ba lớp — bao gồm đầu vào và đầu ra — có thể được coi là một thuật toán học sâu. Điều này thường được thể hiện bằng sơ đồ dưới đây.

Cách mà deep learning và machine learning khác nhau là ở cách mỗi thuật toán học. Học sâu tự động hóa phần lớn phần trích xuất tính năng của quy trình, loại bỏ một số can thiệp thủ công của con người cần thiết và cho phép sử dụng các tập dữ liệu lớn hơn. Bạn có thể nghĩ về deep learning như "học máy có thể mở rộng" như Lex Fridman đã lưu ý trong cùng một bài giảng của MIT từ trên. Học máy cổ điển, hay "không sâu", phụ thuộc nhiều hơn vào sự can thiệp của con người để học. Các chuyên gia con người xác định thứ bậc của các tính năng để hiểu sự khác biệt giữa các đầu vào dữ liệu, thường yêu cầu dữ liệu có cấu trúc hơn để tìm hiểu.

Học máy "sâu" có thể tận dụng các bộ dữ liệu được gắn nhãn, còn được gọi là học có giám sát, để thông báo cho thuật toán của nó, nhưng nó không nhất thiết phải yêu cầu tập dữ liệu được gắn nhãn. Nó có thể nhập dữ liệu phi cấu trúc ở dạng thô (ví dụ: văn bản, hình ảnh) và nó có thể tự động xác định thứ bậc của các tính năng phân biệt các loại

dữ liệu khác nhau với nhau. Không giống như học máy, nó không yêu cầu sự can thiệp của con người để xử lý dữ liệu, cho phép chúng ta mở rộng quy mô học máy theo những cách thú vị hơn.

2.1.3 Ứng dụng trí tuệ nhân tạo

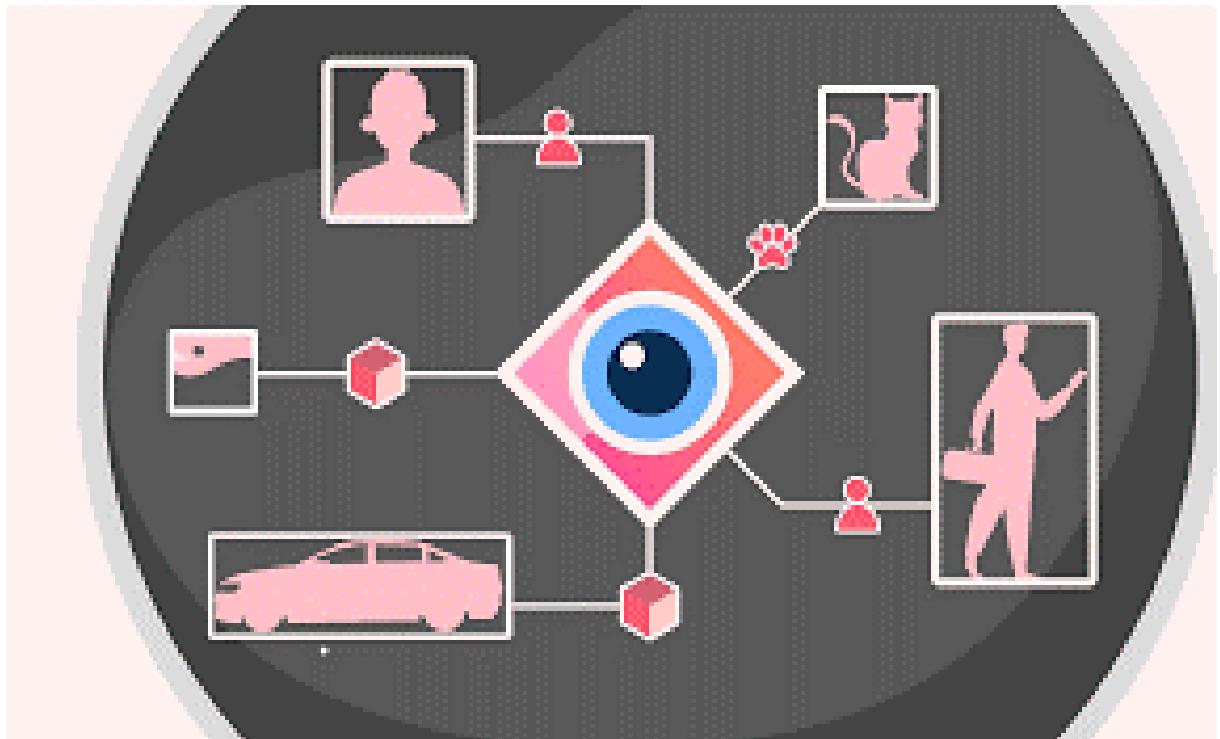
Có rất nhiều ứng dụng trong thế giới thực của các hệ thống AI ngày nay. Dưới đây là một số trường hợp sử dụng phổ biến nhất:

- **Nhận dạng giọng nói:** Ví dụ: Siri — hoặc cung cấp nhiều khả năng truy cập hơn xung quanh nhắm tin.
- **Dịch vụ khách hàng:** Ví dụ bao gồm các bot nhắm tin trên các trang web thương mại điện tử với các trợ lý ảo, ứng dụng nhắm tin, chẳng hạn như Slack và Facebook Messenger và các tác vụ thường được thực hiện bởi trợ lý ảo và trợ lý giọng nói.
- **Thị giác máy tính:** Được hỗ trợ bởi các mạng thần kinh tích chập, thị giác máy tính có các ứng dụng trong gắn thẻ ảnh trên phương tiện truyền thông xã hội, hình ảnh X quang trong chăm sóc sức khỏe và xe tự lái trong ngành công nghiệp ô tô.
- **Công cụ đề xuất:** Được sử dụng để đưa ra các đề xuất bổ sung có liên quan cho khách hàng trong quá trình thanh toán cho các nhà bán lẻ trực tuyến.
- **Giao dịch chứng khoán tự động:** Được thiết kế để tối ưu hóa danh mục đầu tư chứng khoán, các nền tảng giao dịch tần số cao do AI điều khiển thực hiện hàng nghìn hoặc thậm chí hàng triệu giao dịch mỗi ngày mà không cần sự can thiệp của con người.

2.2 TỔNG QUAN VỀ THỊ GIÁC MÁY

2.2.1 Thị giác máy tính là gì?

Thị giác máy tính là một lĩnh vực trí tuệ nhân tạo (AI) cho phép máy tính và hệ thống lấy thông tin có ý nghĩa từ hình ảnh kỹ thuật số, video và các đầu vào trực quan khác - và thực hiện hành động hoặc đưa ra đề xuất dựa trên thông tin đó. Nếu AI cho phép máy tính suy nghĩ, thị giác máy tính cho phép chúng nhìn, quan sát và hiểu.



Hình 2.3: Hình ảnh minh họa Computer vision

Thị giác máy tính hoạt động giống như thị giác của con người, ngoại trừ con người có một khởi đầu thuận lợi. Thị giác của con người có lợi thế về tuổi thọ của bối cảnh để đào tạo cách phân biệt các vật thể, chúng ở xa bao nhiêu, liệu chúng có đang di chuyển hay không và liệu có điều gì đó không ổn trong hình ảnh hay không.

Thị giác máy tính đào tạo máy móc thực hiện các chức năng này, nhưng nó phải làm điều đó trong thời gian ngắn hơn nhiều với máy ảnh, dữ liệu và thuật toán thay vì vông mạc, dây thần kinh thị giác và vỏ não thị giác. Bởi vì một hệ thống được đào tạo để kiểm tra sản phẩm hoặc xem tài sản sản xuất có thể phân tích hàng ngàn sản phẩm hoặc quy trình mỗi phút, nhận thấy các khiếm khuyết hoặc vẫn đề không thể nhận ra, nó có thể nhanh chóng vượt qua khả năng của con người.

Thị giác máy tính được sử dụng trong các ngành công nghiệp khác nhau, từ năng lượng và tiện ích đến sản xuất và ô tô - và thị trường đang tiếp tục phát triển. Dự kiến sẽ đạt 48,6 tỷ USD vào năm 2022.1

2.2.2 Thị giác máy tính hoạt động như thế nào?

Thị giác máy tính cần nhiều dữ liệu. Nó chạy các phân tích dữ liệu lặp đi lặp lại cho đến khi nó phân biệt được sự khác biệt và cuối cùng nhận ra hình ảnh. Ví dụ, để đào tạo một máy tính nhận dạng lốp ô tô, nó cần được cung cấp một lượng lớn hình ảnh lốp xe và các mặt hàng liên quan đến lốp xe để tìm hiểu sự khác biệt và nhận ra lốp xe, đặc biệt là lốp không có khuyết tật.

Hai công nghệ thiết yếu được sử dụng để thực hiện điều này: một loại máy học được gọi là học sâu và mạng nơ-ron tích chập (CNN).

Học máy sử dụng các mô hình thuật toán cho phép máy tính tự dạy về bối cảnh của dữ liệu trực quan. Nếu đủ dữ liệu được cung cấp thông qua mô hình, máy tính sẽ "nhìn" vào dữ liệu và tự dạy cách phân biệt hình ảnh này với hình ảnh khác. Các thuật toán cho phép máy tự học, thay vì ai đó lập trình nó để nhận ra hình ảnh.

CNN giúp mô hình học máy hoặc học sâu "nhìn" bằng cách chia nhỏ hình ảnh thành các pixel được gắn thẻ hoặc nhãn. Nó sử dụng các nhãn để thực hiện các phép tích chập (một phép toán trên hai hàm để tạo ra hàm thứ ba) và đưa ra dự đoán về những gì nó đang "nhìn thấy". Mạng lưới thần kinh chạy các kết cấu và kiểm tra tính chính xác của các dự đoán của nó trong một loạt các lần lặp lại cho đến khi các dự đoán bắt đầu trở thành sự thật. Sau đó, nó nhận ra hoặc nhìn thấy hình ảnh theo cách tương tự như con người.

Giống như một con người tạo ra một hình ảnh ở khoảng cách xa, CNN trước tiên phân biệt các cạnh cứng và hình dạng đơn giản, sau đó điền thông tin khi nó chạy lặp lại các dự đoán của nó. CNN được sử dụng để hiểu các hình ảnh đơn lẻ. Mạng nơ-ron tái phát (RNN) được sử dụng theo cách tương tự cho các ứng dụng video để giúp máy tính hiểu cách hình ảnh trong một loạt các khung hình có liên quan với nhau.

2.2.3 Một vài ví dụ về các tác vụ Thị giác máy tính

Dưới đây là một vài ví dụ về các tác vụ thị giác máy tính đã được thiết lập:

- **Phân loại hình ảnh** nhìn thấy một hình ảnh và có thể phân loại nó (một, một quả táo, khuôn mặt của một người). Chính xác hơn, nó có thể dự đoán chính xác rằng

một hình ảnh nhất định thuộc về một lớp nhất định. Ví dụ: một công ty truyền thông xã hội có thể muốn sử dụng nó để tự động xác định và tách biệt các hình ảnh phản cảm do người dùng tải lên.

- **Phát hiện đối tượng** có thể sử dụng phân loại hình ảnh để xác định một loại hình ảnh nhất định, sau đó phát hiện và lập bảng xuất hiện của chúng trong hình ảnh hoặc video. Ví dụ bao gồm phát hiện thiệt hại trên dây chuyền lắp ráp hoặc xác định máy móc cần bảo trì.
- **Theo dõi đối tượng theo dõi** hoặc theo dõi một đối tượng khi nó được phát hiện. Tác vụ này thường được thực hiện với hình ảnh được chụp theo trình tự hoặc nguồn cấp dữ liệu video thời gian thực. Ví dụ, các phương tiện tự trị không chỉ cần phân loại và phát hiện các đối tượng như người đi bộ, xe hơi khác và cơ sở hạ tầng đường bộ, họ cần theo dõi chúng trong chuyển động để tránh va chạm và tuân thủ luật giao thông.
- **Truy xuất hình ảnh dựa trên nội dung** sử dụng thị giác máy tính để duyệt, tìm kiếm và truy xuất hình ảnh từ các kho dữ liệu lớn, dựa trên nội dung của hình ảnh thay vì thẻ siêu dữ liệu được liên kết với chúng. Tác vụ này có thể kết hợp chú thích hình ảnh tự động thay thế việc gắn thẻ hình ảnh thủ công. Các tác vụ này có thể được sử dụng cho các hệ thống quản lý tài sản kỹ thuật số và có thể tăng độ chính xác của việc tìm kiếm và truy xuất.

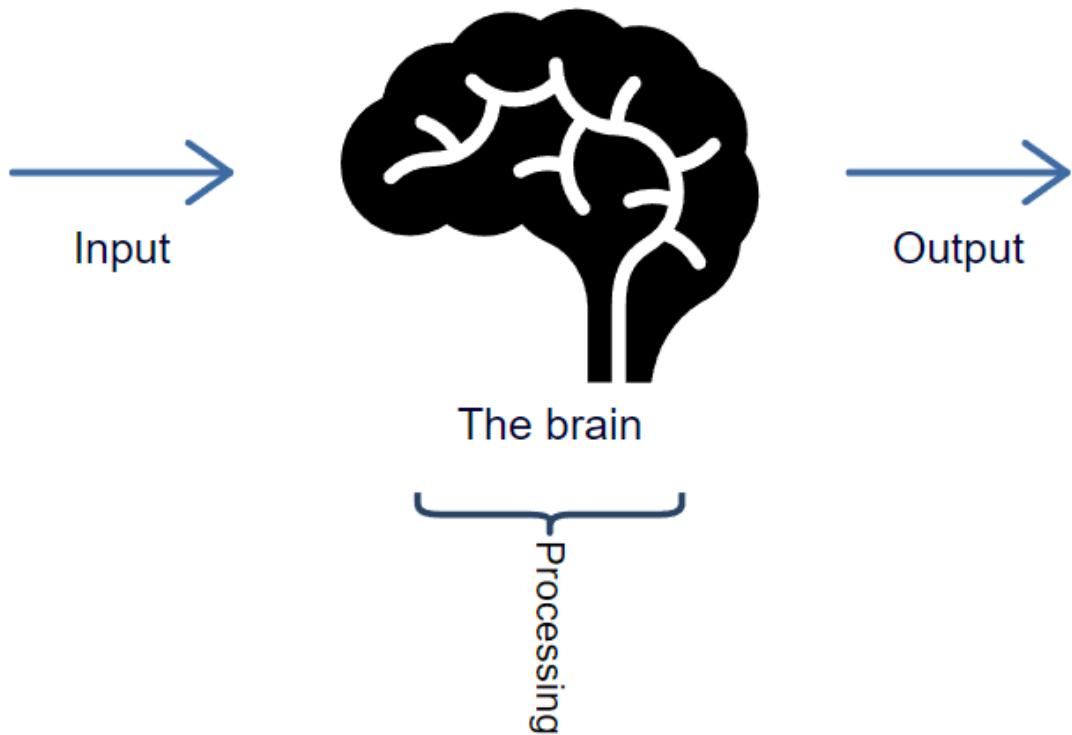
2.3 TỔNG QUAN VỀ MẠNG NƠ-RON

Mạng lưới thần kinh là các thuật toán lấy cảm hứng từ hoạt động của bộ não con người để đưa ra quyết định và thực hiện các nhiệm vụ. Họ đạt được điều này bằng cách nhận ra các mối quan hệ trong dữ liệu như bộ não con người diễn giải kiến thức.

Một mạng lưới thần kinh bao gồm các nút được kết nối với nhau thông qua các khớp thần kinh trong một cấu trúc phân lớp. Các nút này tương tự như tế bào thần kinh trong não. Họ liên tục học hỏi từ dữ liệu được cung cấp cho họ và nhằm mục đích tạo ra kết quả chính xác nhất có thể.

Bộ não xử lý đầu vào và biến nó thành một cái gì đó mà người đó hiểu.

Mạng nơ-ron là một điều cần thiết để có thể tạo ra các mô hình trong học sâu.



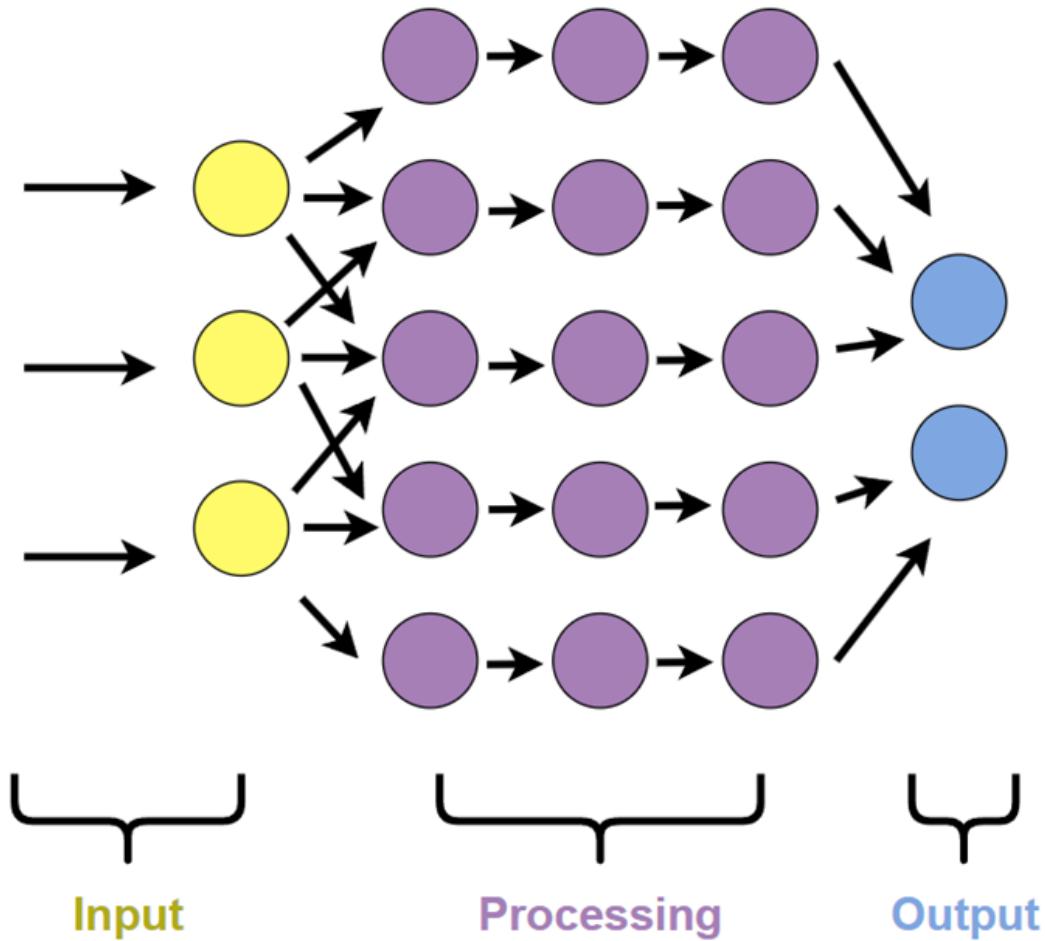
Hình 2.4: Hình ảnh minh họa Bộ não xử lý thông tin đầu vào và chuyển đổi nó thành thứ mà con người có thể hiểu được.

ANN và CNN là hai trong số các mạng thần kinh được sử dụng phổ biến nhất.

2.3.1 Mạng nơ-ron nhân tạo

Mạng nơ-ron nhân tạo là mạng nơ-ron đa năng có thể được áp dụng cho các nhiệm vụ khác nhau. Họ làm việc bằng cách sử dụng các kỹ thuật feedforward và backpropagation.

- **Feedforward:** ANN lấy dữ liệu thông qua một lớp nút đầu vào và làm cho nó đi qua một hoặc nhiều lớp bên trong cho đến khi đạt được lớp nút đầu ra. Các lớp bên trong chịu trách nhiệm xử lý dữ liệu thông minh và được ẩn. Thông tin sau đó biến đổi sau khi vượt qua mỗi lớp và cuối cùng đạt đến trạng thái đầu ra của nó. Vì dữ liệu di chuyển về phía trước, điều này được gọi là feedforward.
- **Backpropagation:** Vì các mạng liên tục học hỏi, backpropagation hỗ trợ trong quá trình này. Trọng số được gán cho các nút. Bất cứ khi nào gặp lỗi trong quá trình đào tạo, thông tin sẽ được gửi trở lại nút trước đó và trọng số được điều chỉnh dựa trên



Hình 2.5: Hình ảnh minh họa ANN đơn giản

đóng góp của chúng vào lõi.

Ứng dụng Mạng nơ-ron nhân tạo

- **Xử lý ngôn ngữ tự nhiên:** Chúng tôi sử dụng ANN trong các tác vụ NLP như phân loại văn bản và nhận dạng giọng nói vì chúng tôi có thể giúp các mô hình học cách hiểu và tạo ra ngôn ngữ của con người.
- **Tài chính:** Chúng ta có thể sử dụng ANN để dự đoán giá cổ phiếu và xu hướng thị trường tài chính cũng như đưa ra quyết định đầu tư dựa trên các mẫu.
- **Y tế:** Chúng tôi có thể áp dụng ANN trong chẩn đoán y tế và dự đoán bệnh vì chúng có thể giúp phân tích dữ liệu bệnh nhân để chẩn đoán và hỗ trợ y học cá nhân hóa.
- **Hệ thống khuyến nghị:** ANN cung cấp năng lượng cho các công cụ đề xuất trong các lĩnh vực khác nhau, ví dụ, các nền tảng phát trực tuyến. Điều này được thực hiện bằng cách phân tích sở thích của người dùng để cung cấp các đề xuất được cá nhân

hóa.

- **Phát triển trò chơi:** Phát triển trò chơi sử dụng ANN để tạo ra các nhân vật không phải người chơi thông minh, thường được viết tắt là NPC và tối ưu hóa chiến lược trò chơi.

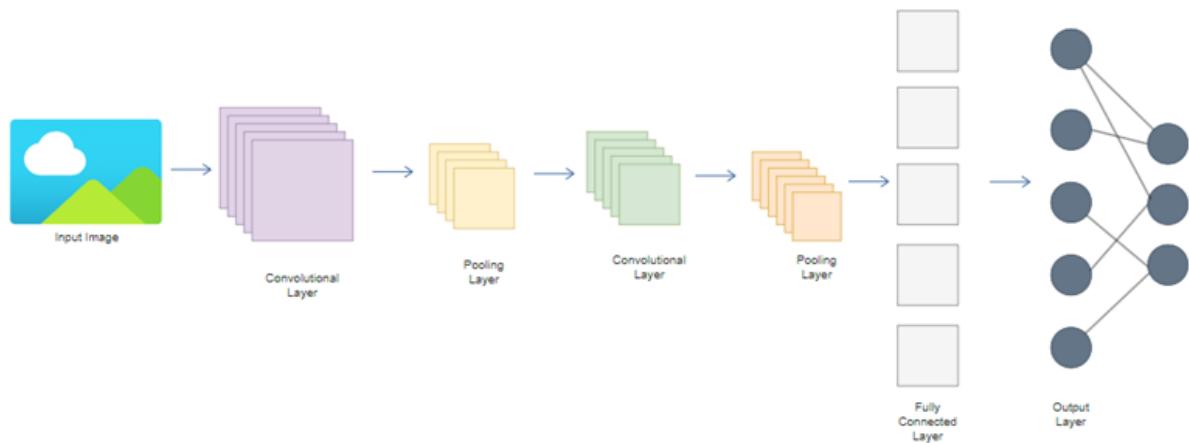
2.3.2 Mạng nơ-ron tích chập

Mạng nơ-ron tích chập là một loại mạng thần kinh chuyên dụng được thiết kế để xử lý dữ liệu giống như lưới như hình ảnh hoặc chuỗi. Chúng hoạt động bằng cách sử dụng các chức năng kích hoạt và các lớp khác nhau.

- **Các lớp tích chập:** Dữ liệu được truyền qua một hoặc nhiều lớp tích chập và một tập hợp các bộ lọc có thể học được được áp dụng. Các lớp này nắm bắt các mẫu bằng cách phát hiện các cạnh, góc và kết cấu.
- **Chức năng kích hoạt:** Tiếp theo, một chức năng kích hoạt, ví dụ: ReLU được áp dụng để thêm phi tuyến tính để giúp làm cho mạng tìm hiểu mối quan hệ phức tạp giữa dữ liệu và các tính năng được trích xuất. Một chức năng kích hoạt, nói một cách đơn giản, giống như một cánh cổng xác định xem một tế bào thần kinh có nên "bắn" hay không dựa trên tổng trọng số của các đầu vào của nó.
- **Các lớp gộp:** Các lớp gộp chủ yếu chịu trách nhiệm cho việc lấy mẫu xuống bản đồ tính năng. Nói một cách dễ dàng hơn, điều này có nghĩa là giữ lại dữ liệu quan trọng trong khi giảm kích thước để giảm yêu cầu tính toán.
- **Các lớp được kết nối đầy đủ:** Các bản đồ tính năng được làm phẳng sau đó được chuyển qua các lớp được kết nối đầy đủ. Họ có thể xử lý các tính năng cấp cao để dự đoán hoặc phân loại.

Ứng dụng Mạng nơ-ron nhân tạo

- **Phân loại hình ảnh:** CNN xuất sắc trong các nhiệm vụ phân loại hình ảnh, nơi họ có thể phân loại chính xác hình ảnh thành nhiều danh mục khác nhau.
- **Phát hiện đối tượng:** Tiếp tục với điểm đầu tiên, CNN cũng được sử dụng rộng rãi để xác định và bản địa hóa nhiều đối tượng. Hãy lấy R-CNN và YOLO làm ví dụ. Các kiến trúc dựa trên CNN này có hiệu suất phát hiện đối tượng tiên tiến.



Hình 2.6: Hình ảnh minh họa CNN trong phân loại hình ảnh

- **Nhận dạng khuôn mặt:** Một cách khác mà CNN có thể góp phần phát hiện là họ có thể tìm hiểu các đặc điểm khuôn mặt và kết hợp chúng với các danh tính đã biết.
- **Hình ảnh y tế:** CNN đã cho thấy kết quả đầy hứa hẹn trong phân tích hình ảnh y tế. Điều này nổi bật trong các nhiệm vụ như phát hiện khối u, phân đoạn các cơ quan hoặc mô và thậm chí chẩn đoán bệnh từ quét y tế.
- **Phân tích video:** Chúng tôi cũng có thể áp dụng CNN cho các tác vụ phân tích video như nhận dạng hành động, tóm tắt video và theo dõi đối tượng video.

Chương 3

THUẬT TOÁN VÀ KỸ THUẬT

3.1 TỔNG QUAN VỀ YOLOv8

3.1.1 YOLOv8 là gì?

YOLOv8 là mô hình YOLO hiện đại mới nhất có thể được sử dụng để phát hiện đối tượng, phân loại hình ảnh và các tác vụ phân đoạn phiên bản. YOLOv8 được phát triển bởi Ultralytics, người cũng đã tạo ra mô hình YOLOv5 có ảnh hưởng và xác định ngành. YOLOv8 bao gồm nhiều thay đổi và cải tiến về trải nghiệm kiến trúc và nhà phát triển so với YOLOv5.

YOLOv8 đang được phát triển tích cực khi viết bài đăng này, vì Ultralytics hoạt động trên các tính năng mới và trả lời phản hồi từ cộng đồng. Thực vậy, khi Ultralytics phát hành một mô hình, nó được hỗ trợ lâu dài: tổ chức làm việc với cộng đồng để làm cho mô hình trở nên tốt nhất có thể.

3.1.2 YOLO phát triển thành YOLOv8 như thế nào?

Dòng mô hình YOLO (You Only Look Once) đã trở nên nổi tiếng trong thế giới thị giác máy tính. Sự nổi tiếng của YOLO là do độ chính xác đáng kể của nó trong khi vẫn duy trì kích thước mô hình nhỏ. Các mô hình YOLO có thể được đào tạo trên một GPU duy nhất, giúp nhiều nhà phát triển có thể truy cập được. Các học viên học máy có thể triển khai nó với chi phí thấp trên phần cứng biên hoặc trên đám mây.

YOLO đã được nuôi dưỡng bởi cộng đồng thị giác máy tính kể từ khi ra mắt lần đầu tiên vào năm 2015 bởi Joseph Redmond. Trong những ngày đầu (phiên bản 1-4), YOLO được duy trì bằng mã C trong một khung học sâu tùy chỉnh được viết bởi Redmond có tên là Darknet.

YOLOv5 suy luận trên xe đạp

Tác giả YOLOv8, Glenn Jocher tại Ultralytics, đã theo dõi repo YOLOv3 trong PyTorch (một khung học tập sâu từ Facebook). Khi việc đào tạo trong repo bóng tối trở nên tốt hơn, Ultralytics cuối cùng đã tung ra mô hình của riêng mình: YOLOv5.

YOLOv5 nhanh chóng trở thành kho lưu trữ SOTA của thế giới nhờ cấu trúc Python linh hoạt. Cấu trúc này cho phép cộng đồng phát minh ra các cải tiến mô hình mới và nhanh chóng chia sẻ chúng trên kho lưu trữ với các phương pháp PyTorch tương tự.

Cùng với các nguyên tắc cơ bản về mô hình mạnh mẽ, những người bảo trì YOLOv5 đã cam kết hỗ trợ một hệ sinh thái phần mềm lành mạnh xung quanh mô hình. Họ tích cực khắc phục sự cố và thúc đẩy khả năng của kho lưu trữ theo yêu cầu của cộng đồng.

Trong hai năm qua, nhiều mô hình khác nhau đã phân nhánh khỏi kho lưu trữ YOLOv5 PyTorch, bao gồm Scaled-YOLOv4, YOLOR và YOLOv7. Các mô hình khác xuất hiện trên khắp thế giới từ các triển khai dựa trên PyTorch của riêng họ, chẳng hạn như YOLOX và YOLOv6. Trên đường đi, mỗi mô hình YOLO đã mang đến các kỹ thuật SOTA mới tiếp tục thúc đẩy độ chính xác và hiệu quả của mô hình.

Trong sáu tháng qua, Ultralytics đã làm việc để nghiên cứu phiên bản SOTA mới nhất của YOLO, YOLOv8. YOLOv8 được ra mắt vào ngày 10/1/2023.

3.1.3 Tại sao nên sử dụng YOLOv8?

Dưới đây là một vài lý do chính tại sao bạn nên cân nhắc sử dụng YOLOv8 cho dự án thị giác máy tính tiếp theo của mình:

- YOLOv8 có tỷ lệ chính xác cao được đo bằng Microsoft COCO và Roboflow 100
- YOLOv8 đi kèm với rất nhiều tính năng tiện lợi cho nhà phát triển, từ CLI dễ sử dụng đến gói Python có cấu trúc tốt.
- Có một cộng đồng lớn xung quanh YOLO và một cộng đồng đang phát triển xung quanh mô hình YOLOv8, có nghĩa là có rất nhiều người trong giới thị giác máy tính

có thể hỗ trợ bạn khi bạn cần hướng dẫn.

YOLOv8 đạt được độ chính xác mạnh mẽ trên COCO. Ví dụ, mô hình YOLOv8m - mô hình trung bình - đạt được 50,2% mAP khi đo trên COCO. Khi được đánh giá dựa trên Roboflow 100, một bộ dữ liệu đánh giá cụ thể hiệu suất mô hình trên các miền tác vụ cụ thể khác nhau, YOLOv8 đạt điểm cao hơn đáng kể so với YOLOv5. Thông tin thêm về điều này được cung cấp trong phân tích hiệu suất của chúng tôi sau trong bài viết.

Hơn nữa, các tính năng tiện lợi cho nhà phát triển trong YOLOv8 là rất đáng kể. Trái ngược với các mô hình khác, nơi các tác vụ được phân chia trên nhiều tệp Python khác nhau mà bạn có thể thực thi, YOLOv8 đi kèm với CLI giúp đào tạo mô hình trực quan hơn. Đây là phần bổ sung cho gói Python cung cấp trải nghiệm mã hóa liền mạch hơn so với các mô hình trước đó.

Cộng đồng xung quanh YOLO rất đáng chú ý khi bạn đang xem xét một mô hình để sử dụng. Nhiều chuyên gia thị giác máy tính biết về YOLO và cách thức hoạt động của nó, và có rất nhiều hướng dẫn trực tuyến về việc sử dụng YOLO trong thực tế. Mặc dù YOLOv8 là mới khi viết phần này, nhưng có rất nhiều hướng dẫn trực tuyến có thể giúp ích.

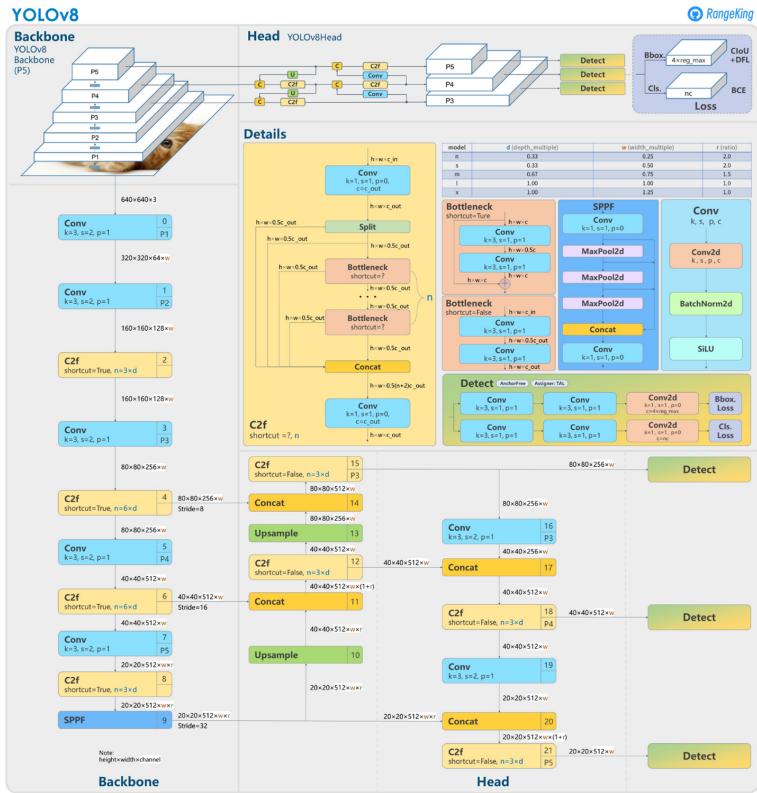
3.1.4 Kiến trúc YOLOv8: A Deep Dive

YOLOv8 chưa có bài báo được xuất bản, vì vậy chúng tôi thiếu cái nhìn sâu sắc trực tiếp về phương pháp nghiên cứu trực tiếp và nghiên cứu cắt bỏ được thực hiện trong quá trình tạo ra nó. Như đã nói, chúng tôi đã phân tích kho lưu trữ và thông tin có sẵn về mô hình để bắt đầu ghi lại những gì mới trong YOLOv8.

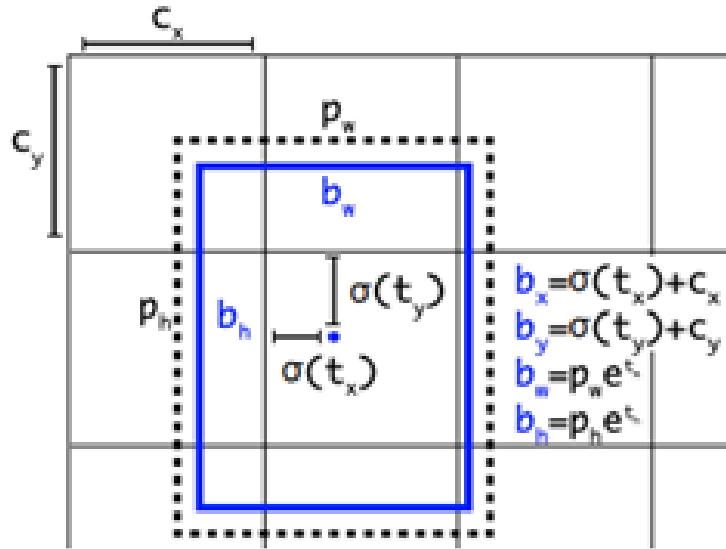
Nếu bạn muốn tự mình nhìn vào mã, hãy kiểm tra kho lưu trữ YOLOv8 và bạn xem sự khác biệt mã này để xem một số nghiên cứu đã được thực hiện như thế nào.

Ở đây chúng tôi cung cấp một bản tóm tắt nhanh về các bản cập nhật mô hình có tác động và sau đó chúng tôi sẽ xem xét đánh giá của mô hình, điều này đã nói lên điều đó. Hình ảnh sau đây được thực hiện bởi người dùng GitHub RangeKing cho thấy một vizualization chi tiết của kiến trúc mạng.

Anchor Free Detection YOLOv8 là một mô hình không neo. Điều này có nghĩa là



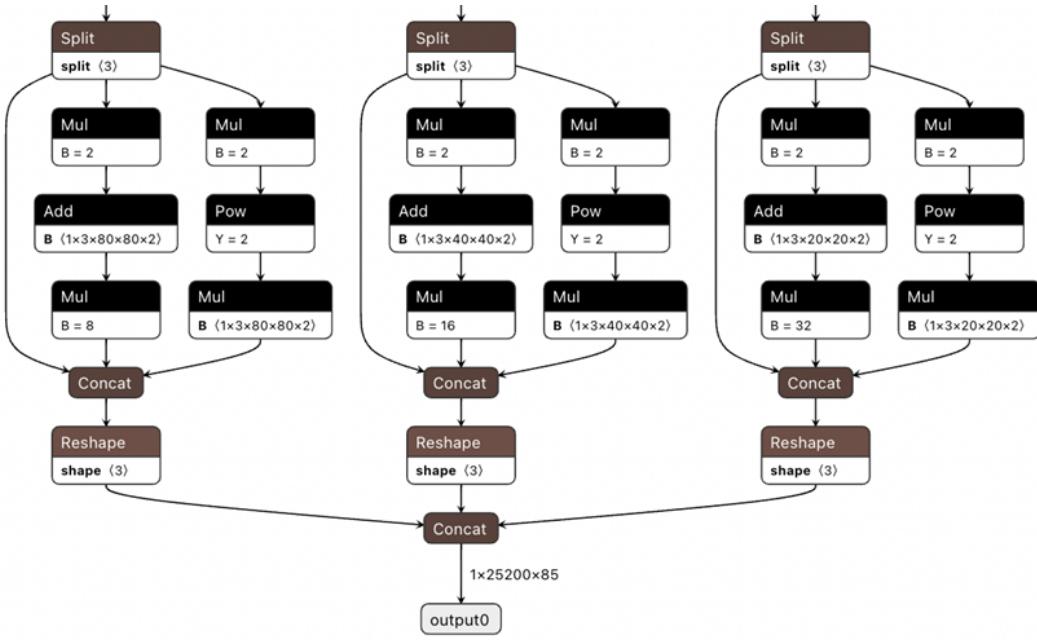
Hình 3.1: Kiến trúc YOLOv8, trực quan hóa được thực hiện bởi người dùng GitHub RangeKing



Hình 3.2: Hình dung hộp neo trong YOLO

nó dự đoán trực tiếp tâm của một đối tượng thay vì độ lệch từ một hộp neo đã biết

Anchor box nổi tiếng là một phần phức tạp của các mô hình YOLO trước đó, vì chúng có thể đại diện cho sự phân bố các hộp của điểm chuẩn mục tiêu nhưng không đại diện cho sự phân phối của tập dữ liệu tùy chỉnh.



Hình 3.3: Đầu phát hiện của YOLOv5, được hiển thị bằng netron.app

Phát hiện không neo làm giảm số lượng dự đoán hộp, giúp tăng tốc độ Triết tiêu không tối đa (NMS), một bước xử lý bài đăng phức tạp sàng lọc thông qua các phát hiện ứng cử viên sau khi suy luận

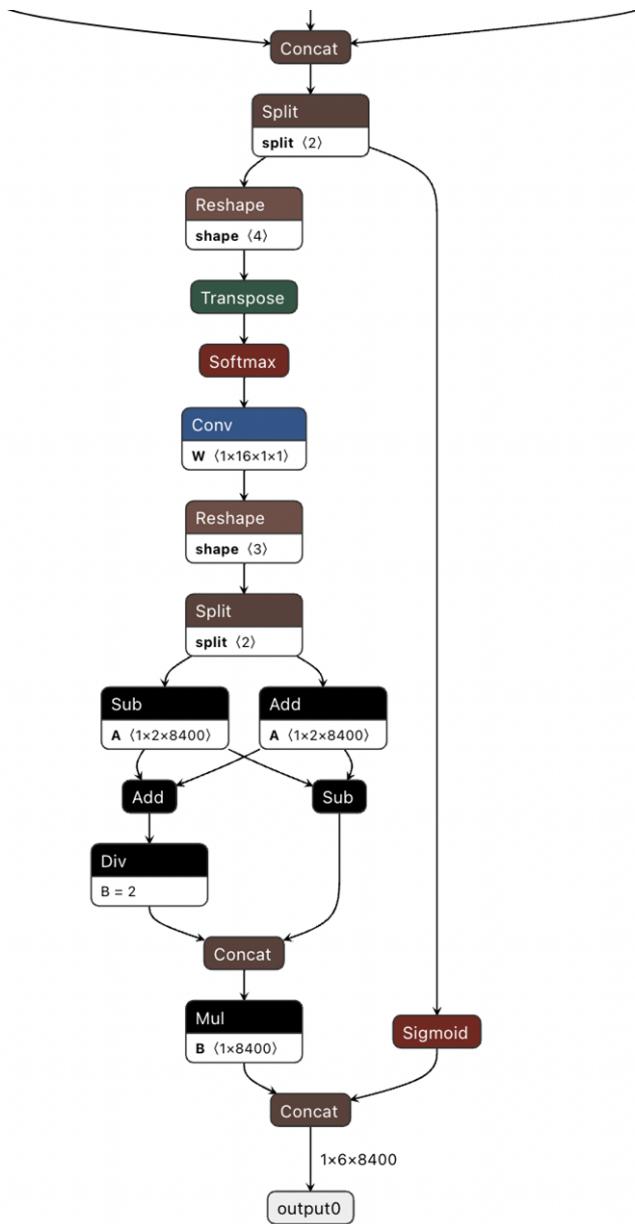
Các kết cấu mới Conv 6x6 đầu tiên của thân cây được thay thế bằng 3x3, khối xây dựng chính đã được thay đổi và C2f thay thế C3. Mô-đun được tóm tắt trong hình dưới đây, trong đó "f" là số lượng tính năng, "e" là tốc độ mở rộng và CBS là một khối bao gồm Conv, BatchNorm và SiLU sau này.

Trong C2f, tất cả các đầu ra từ Bottleneck (tên ưa thích cho hai conv 3x3 với các kết nối còn lại) được nối nối. Trong khi ở C3 chỉ có đầu ra của Bottleneck cuối cùng được sử dụng.

Bottleneck giống như trong YOLOv5 nhưng kích thước kernel của conv đầu tiên đã được thay đổi từ 1x1 thành 3x3. Từ thông tin này, chúng ta có thể thấy rằng YOLOv8 đang bắt đầu trở lại khôi ResNet được xác định vào năm 2015.

Ở cổ, các tính năng được nối trực tiếp mà không buộc cùng kích thước kênh. Điều này làm giảm số lượng thông số và kích thước tổng thể của tensor.

Closing the Mosaic Augmentation Nghiên cứu học sâu có xu hướng tập trung vào kiến trúc mô hình, nhưng thói quen đào tạo trong YOLOv5 và YOLOv8 là một phần



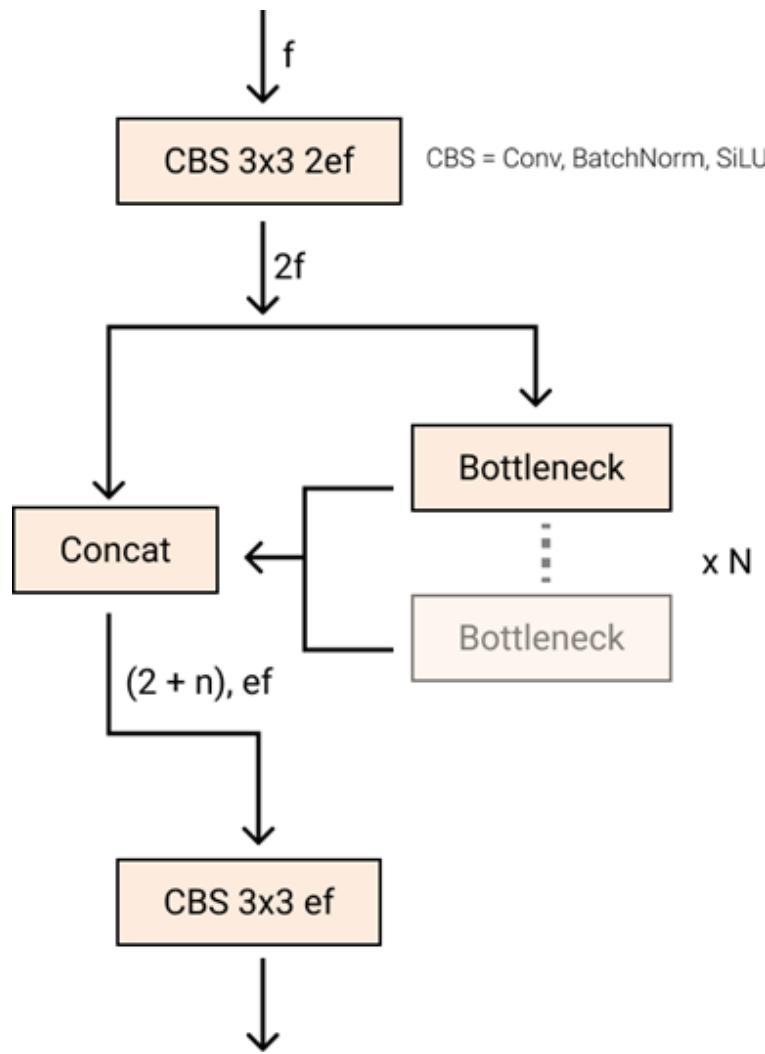
Hình 3.4: Đầu phát hiện của YOLOv8, được hiển thị bằng netron.app

thiết yếu trong thành công của họ.

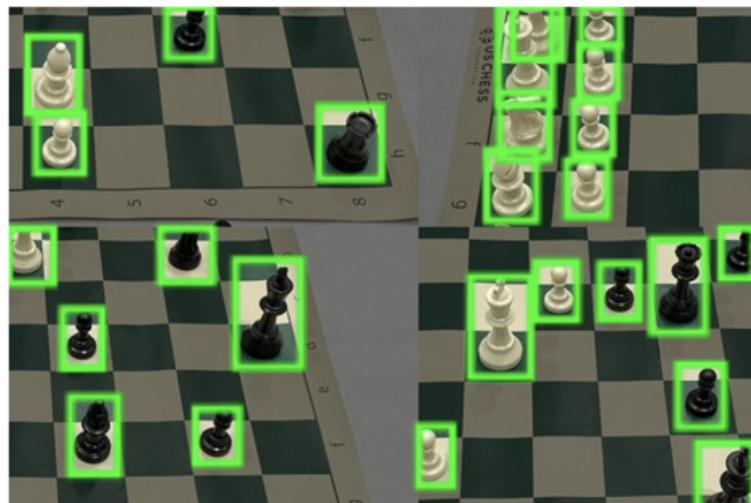
YOLOv8 tăng cường hình ảnh trong quá trình đào tạo trực tuyến. Ở mỗi ký nguyên, mô hình thấy một biến thể hơi khác nhau của hình ảnh mà nó đã được cung cấp.

Một trong những tăng cường đó được gọi là tăng cường khám. Điều này liên quan đến việc ghép bốn hình ảnh lại với nhau, buộc mô hình phải tìm hiểu các đối tượng ở các vị trí mới, bị tách một phần và chồng lại các pixel xung quanh khác nhau.

Tuy nhiên, việc tăng cường này được chứng minh theo kinh nghiệm là làm giảm hiệu suất nếu được thực hiện thông qua toàn bộ thói quen đào tạo. Thật thuận lợi khi tắt nó



Hình 3.5: Mô-đun YOLOv8 C2f mới



Hình 3.6: Khảm tăng cường hình ảnh bàn cờ

▼ Detection

Model	size (pixels)	mAP ^{val} 50-95	Speed CPU (ms)	Speed T4 GPU (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	-	-	3.2	8.7
YOLOv8s	640	44.9	-	-	11.2	28.6
YOLOv8m	640	50.2	-	-	25.9	78.9
YOLOv8l	640	52.9	-	-	43.7	165.2
YOLOv8x	640	53.9	-	-	68.2	257.8

- mAP^{val} values are for single-model single-scale on [COCO val2017](#) dataset.
Reproduce by `yolo mode=val task=detect data=coco.yaml device=0`
- Speed averaged over COCO val images using an [Amazon EC2 P4d](#) instance.
Reproduce by `yolo mode=val task=detect data=coco128.yaml batch=1 device=0/cpu`

Hình 3.7: Đánh giá YOLOv8 COCO

trong mười kỷ nguyên đào tạo cuối cùng.

Loại thay đổi này là minh chứng cho sự chú ý cẩn thận của mô hình YOLO đã được đưa ra ngoài giờ trong repo YOLOv5 và trong nghiên cứu YOLOv8.

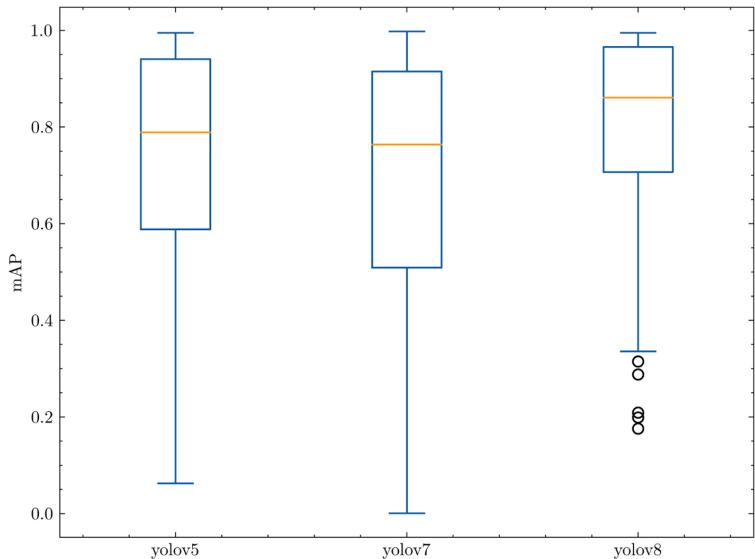
3.1.5 Cải thiện độ chính xác của YOLOv8

Nghiên cứu YOLOv8 chủ yếu được thúc đẩy bởi đánh giá thực nghiệm về điểm chuẩn COCO. Khi mỗi phần của mạng lưới và thói quen đào tạo được tinh chỉnh, các thử nghiệm mới được chạy để xác nhận hiệu ứng thay đổi đối với mô hình COCO.

Độ chính xác của YOLOv8 COCO (Common Objects in Context) là tiêu chuẩn công nghiệp để đánh giá các mô hình phát hiện đối tượng. Khi so sánh các mô hình trên COCO, chúng tôi xem xét giá trị mAP và phép đo FPS để biết tốc độ suy luận. Các mô hình nên được so sánh ở tốc độ suy luận tương tự.

Hình ảnh dưới đây cho thấy độ chính xác của YOLOv8 trên COCO, sử dụng dữ liệu được thu thập bởi nhóm Ultralytics và được công bố trong YOLOv8 README của họ:

Độ chính xác của YOLOv8 COCO là hiện đại cho các mô hình ở độ trễ suy luận tương đương khi viết bài đăng này.



Hình 3.8: YOLOs mAP@.50 so với RF100

Độ chính xác RF100 Tại Roboflow, chúng tôi đã rút ra 100 bộ dữ liệu mẫu từ Vũ trụ Roboflow, một kho lưu trữ hơn 100.000 bộ dữ liệu, để đánh giá mức độ khai quát hóa các mô hình cho các miền mới. Điểm chuẩn của chúng tôi, được phát triển với sự hỗ trợ của Intel, là một chuẩn mực cho các học viên thị giác máy tính được thiết kế để cung cấp câu trả lời tốt hơn cho câu hỏi: "mô hình này sẽ hoạt động tốt như thế nào trên tập dữ liệu tùy chỉnh của tôi?"

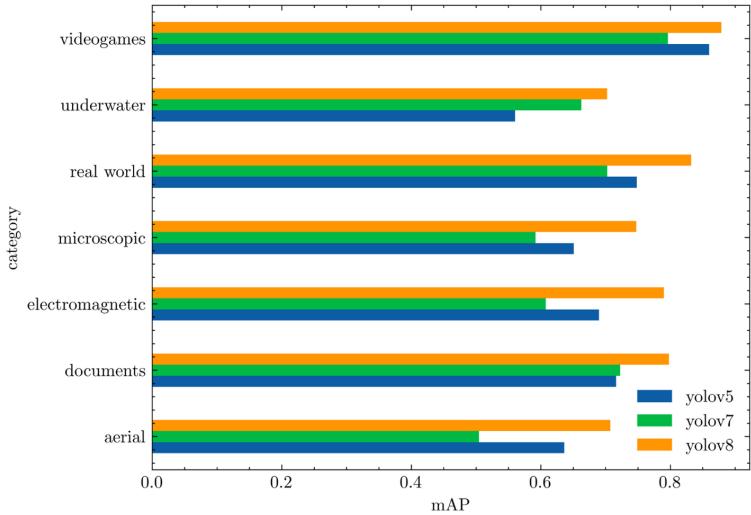
Chúng tôi đã đánh giá YOLOv8 trên điểm chuẩn RF100 của chúng tôi cùng với YOLOv5 và YOLOv7, các ô hộp sau đây hiển thị mAP@.50 của mỗi mô hình.

Chúng tôi chạy phiên bản nhỏ của mỗi mô hình trong 100 kỷ nguyên, chúng tôi đã chạy một lần với một hạt giống duy nhất vì vậy do xổ số gradient lấy kết quả này với một hạt muối.

Biểu đồ hộp dưới đây cho chúng ta biết YOLOv8 có ít ngoại lệ hơn và mAP tổng thể tốt hơn khi được đo so với điểm chuẩn Roboflow 100.

Biểu đồ thanh sau đây cho thấy mAP@.50 trung bình cho mỗi loại RF100. Một lần nữa, YOLOv8 vượt trội hơn tất cả các mô hình trước đó.

Liên quan đến đánh giá YOLOv5, mô hình YOLOv8 tạo ra kết quả tương tự trên mỗi tập dữ liệu hoặc cải thiện kết quả đáng kể.

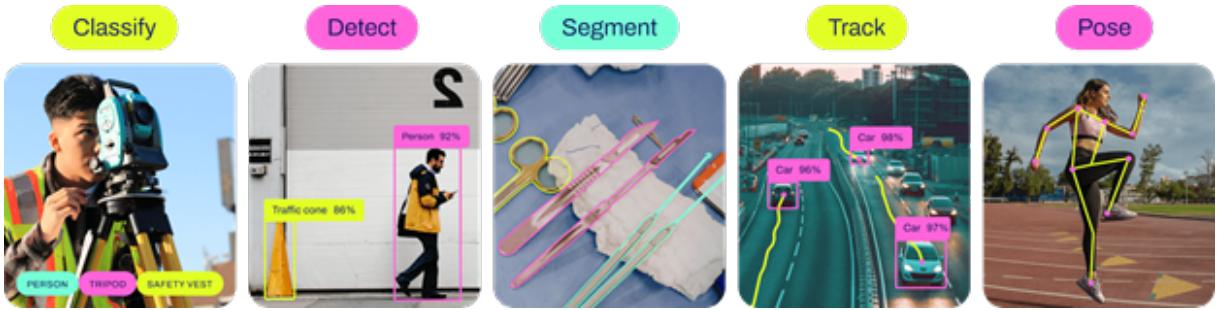


Hình 3.9: YOLO trung bình mAP@,.50 so với các loại RF100

3.2 CÁC TASKS SỬ DỤNG TRONG YOLOV8

YOLOv8 là một khung AI hỗ trợ nhiều tác vụ thị giác máy tính. Khung có thể được sử dụng để thực hiện phát hiện, phân đoạn, phân loại và ước tính tư thế. Mỗi nhiệm vụ này có một mục tiêu và trường hợp sử dụng khác nhau.

- **Detect:** Phát hiện là nhiệm vụ chính được YOLOv8 hỗ trợ. Nó liên quan đến việc phát hiện các đối tượng trong khung hình ảnh hoặc video và vẽ các hộp giới hạn xung quanh chúng. Các đối tượng được phát hiện được phân loại thành các loại khác nhau dựa trên các tính năng của chúng. YOLOv8 có thể phát hiện nhiều đối tượng trong một khung hình ảnh hoặc video với độ chính xác và tốc độ cao.
- **SSegment:** Phân đoạn là một nhiệm vụ liên quan đến việc phân đoạn hình ảnh thành các khu vực khác nhau dựa trên nội dung của hình ảnh. Mỗi khu vực được gán một nhãn dựa trên nội dung của nó. Nhiệm vụ này rất hữu ích trong các ứng dụng như phân đoạn hình ảnh và hình ảnh y tế. YOLOv8 sử dụng một biến thể của kiến trúc U-Net để thực hiện phân đoạn.
- **Classify:** Phân loại là một nhiệm vụ liên quan đến việc phân loại hình ảnh thành các danh mục khác nhau. YOLOv8 có thể được sử dụng để phân loại hình ảnh dựa trên nội dung của chúng. Nó sử dụng một biến thể của kiến trúc EfficientNet để thực hiện phân loại.



Hình 3.10: Các tasks có trong YOLOv8



Hình 3.11: Hình minh họa Object Detection

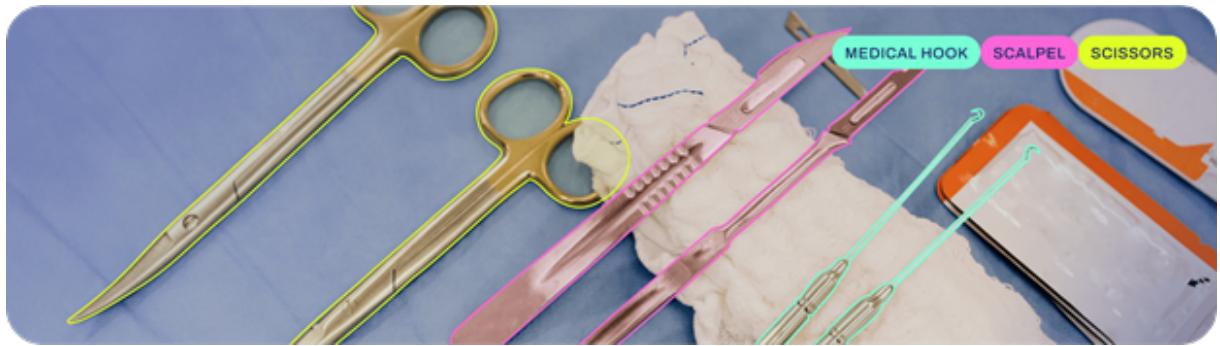
- Pose:** Phát hiện tư thế / điểm chính là một nhiệm vụ liên quan đến việc phát hiện các điểm cụ thể trong khung hình ảnh hoặc video. Những điểm này được gọi là điểm chính và được sử dụng để theo dõi chuyển động hoặc ước tính tư thế. YOLOv8 có thể phát hiện các điểm chính trong khung hình ảnh hoặc video với độ chính xác và tốc độ cao.

YOLOv8 hỗ trợ nhiều tác vụ, bao gồm phát hiện, phân đoạn, phân loại và phát hiện điểm chính. Mỗi nhiệm vụ này có các mục tiêu và trường hợp sử dụng khác nhau. Bằng cách hiểu sự khác biệt giữa các tác vụ này, bạn có thể chọn tác vụ thích hợp cho ứng dụng thị giác máy tính của mình.

3.2.1 Object Detection

Phát hiện đối tượng là một nhiệm vụ liên quan đến việc xác định vị trí và lớp đối tượng trong luồng hình ảnh hoặc video.

Đầu ra của máy dò đối tượng là một tập hợp các hộp giới hạn bao quanh các đối tượng trong hình ảnh, cùng với nhãn lớp và điểm tin cậy cho mỗi hộp. Phát hiện đối tượng là



Hình 3.12: Hình minh họa Instance Segmentation



Hình 3.13: Hình minh họa Pose Estimation

một lựa chọn tốt khi bạn cần xác định các đối tượng quan tâm trong một cảnh, nhưng không cần biết chính xác đối tượng ở đâu hoặc hình dạng chính xác của nó.

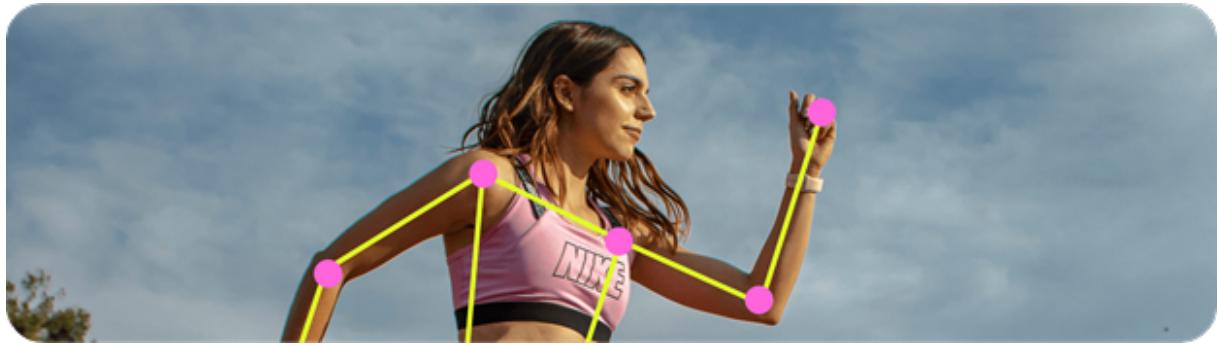
3.2.2 Instance Segmentation

Phân đoạn phiên bản đi một bước xa hơn so với phát hiện đối tượng và liên quan đến việc xác định các đối tượng riêng lẻ trong một hình ảnh và phân đoạn chúng từ phần còn lại của hình ảnh.

Đầu ra của mô hình phân đoạn phiên bản là một tập hợp các mặt nạ hoặc đường viền phác thảo từng đối tượng trong hình ảnh, cùng với nhãn lớp và điểm tin cậy cho từng đối tượng. Phân đoạn phiên bản rất hữu ích khi bạn cần biết không chỉ vị trí của các đối tượng trong hình ảnh mà còn cả hình dạng chính xác của chúng.

3.2.3 Pose Estimation

Ước tính tư thế là một nhiệm vụ liên quan đến việc xác định vị trí của các điểm cụ thể trong ảnh, thường được gọi là các điểm chính. Các điểm chính có thể đại diện cho



Hình 3.14: Hình minh họa Image Classification

các phần khác nhau của đối tượng như khớp, mốc hoặc các tính năng đặc biệt khác. Vị trí của các điểm chính thường được biểu diễn dưới dạng một tập hợp các tọa độ 2D [x, y] hoặc 3D [x, y, visible].

Đầu ra của mô hình ước tính tư thế là một tập hợp các điểm đại diện cho các điểm chính trên một đối tượng trong ảnh, thường cùng với điểm tin cậy cho mỗi điểm. Ước tính tư thế là một lựa chọn tốt khi bạn cần xác định các phần cụ thể của một đối tượng trong một cảnh và vị trí của chúng trong mối quan hệ với nhau.

3.2.4 Image Classification

3.2.5 Pose Estimation

Phân loại hình ảnh là nhiệm vụ đơn giản nhất trong ba nhiệm vụ và liên quan đến việc phân loại toàn bộ hình ảnh thành một trong một tập hợp các lớp được xác định trước.

Đầu ra của bộ phân loại hình ảnh là một nhãn lớp duy nhất và điểm tin cậy. Phân loại hình ảnh rất hữu ích khi bạn chỉ cần biết hình ảnh thuộc lớp nào và không cần biết các đối tượng của lớp đó nằm ở đâu hoặc hình dạng chính xác của chúng là gì.

Chương 4

CÔNG CỤ SỬ DỤNG

4.1 PHẦN MỀM STREAMLIT

4.1.1 Streamlit là gì?

Streamlit là một khung công tác mã nguồn mở và miễn phí để nhanh chóng xây dựng và chia sẻ các ứng dụng web học máy và khoa học dữ liệu tuyệt đẹp. Nó là một thư viện dựa trên Python được thiết kế đặc biệt cho các kỹ sư học máy. Các nhà khoa học dữ liệu hoặc kỹ sư học máy không phải là nhà phát triển web và họ không quan tâm đến việc dành hàng tuần để học cách sử dụng các khung này để xây dựng các ứng dụng web. Thay vào đó, họ muốn một công cụ dễ học và dễ sử dụng hơn, miễn là nó có thể hiển thị dữ liệu và thu thập các tham số cần thiết để mô hình hóa. Streamlit cho phép bạn tạo một ứng dụng trông tuyệt đẹp chỉ với một vài dòng mã.

4.1.2 Tại sao nên sử dụng Streamlit?

Điều tốt nhất về Streamlit là bạn thậm chí không cần biết những điều cơ bản về phát triển web để bắt đầu hoặc tạo ứng dụng web đầu tiên của mình. Vì vậy, nếu bạn là người đam mê khoa học dữ liệu và bạn muốn triển khai các mô hình của mình một cách dễ dàng, nhanh chóng và chỉ với một vài dòng mã, Streamlit là một lựa chọn phù hợp.

Một trong những khía cạnh quan trọng của việc làm cho một ứng dụng thành công là cung cấp nó với giao diện người dùng hiệu quả và trực quan. Nhiều ứng dụng nặng

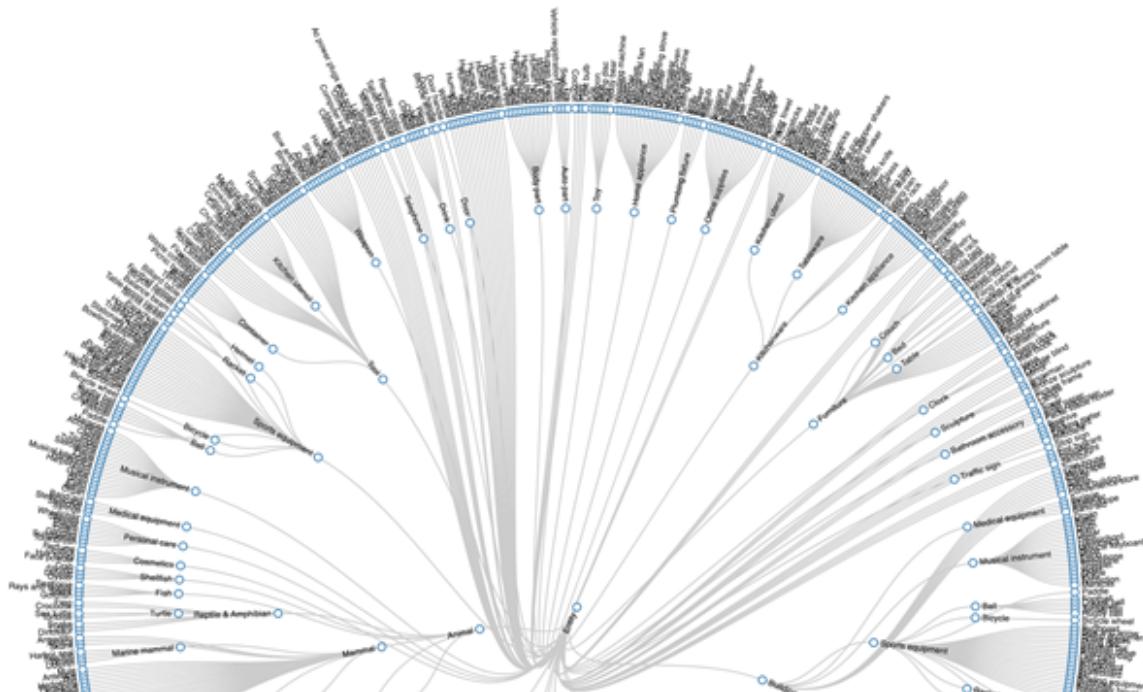


Hình 4.1: Hình minh họa phần mềm Streamlit

về dữ liệu hiện đại phải đối mặt với thách thức xây dựng giao diện người dùng hiệu quả một cách nhanh chóng mà không cần thực hiện các bước phức tạp. Streamlit là một thư viện Python mã nguồn mở đầy hứa hẹn, cho phép các nhà phát triển xây dựng giao diện người dùng hấp dẫn ngay lập tức.

Streamlit là cách dễ nhất đặc biệt đối với những người không có kiến thức về giao diện người dùng để đưa mã của họ vào ứng dụng web:

- Không yêu cầu kinh nghiệm hoặc kiến thức front-end (html, js, css).
- Bạn không cần phải mất nhiều ngày hoặc hàng tháng để tạo một ứng dụng web, bạn có thể tạo một ứng dụng học máy hoặc khoa học dữ liệu thực sự đẹp mắt chỉ trong vài giờ hoặc thậm chí vài phút.
- Nó tương thích với phần lớn các thư viện Python (ví dụ: giao trúc, matplotlib, seaborn, plotly, Keras, PyTorch, SymPy (latex)).
- Cần ít mã hơn để tạo các ứng dụng web tuyệt vời.
- Bộ nhớ đệm dữ liệu đơn giản hóa và tăng tốc các đường ống tính toán.



Hình 4.2: Hình minh họa trang web Open Images Dataset v7

4.2 TRANG WEB OPEN IMAGES DATASET V7

Open Images V7 là một bộ dữ liệu linh hoạt và mở rộng được Google bảo vệ. Nhằm mục đích thúc đẩy nghiên cứu trong lĩnh vực thị giác máy tính, nó tự hào có một bộ sưu tập lớn các hình ảnh được chú thích với rất nhiều dữ liệu, bao gồm nhãn cấp hình ảnh, hộp giới hạn đối tượng, mặt nạ phân đoạn đối tượng, mối quan hệ trực quan và tường thuật cuộn bô.

Các tính năng chính

- Bao gồm 9M hình ảnh được chú thích theo nhiều cách khác nhau để phù hợp với nhiều tác vụ thị giác máy tính.
 - Có 16 triệu hộp giới hạn đáng kinh ngạc trên 600 lớp đối tượng trong 1,9 triệu hình ảnh. Những chiếc hộp này chủ yếu được vẽ tay bởi các chuyên gia đảm bảo độ chính xác cao.
 - Chú thích mối quan hệ trực quan tổng cộng 3,3 triệu có sẵn, chi tiết 1.466 bộ ba mối quan hệ duy nhất, thuộc tính đối tượng và hoạt động của con người.
 - V7 giới thiệu 66,4 triệu nhãn cấp điểm trên 1,4 triệu hình ảnh, trải dài trên 5.827 lớp.
 - Bao gồm 61,4 triệu nhãn cấp hình ảnh trên một bộ đa dạng gồm 20.638 lớp.

- Cung cấp một nền tảng thống nhất để phân loại hình ảnh, phát hiện đối tượng, phát hiện mối quan hệ, phân đoạn phiên bản và mô tả hình ảnh đa phương thức.

Cấu trúc tập dữ liệu Open Images V7 được cấu trúc thành nhiều thành phần phục vụ cho các thách thức thị giác máy tính khác nhau:

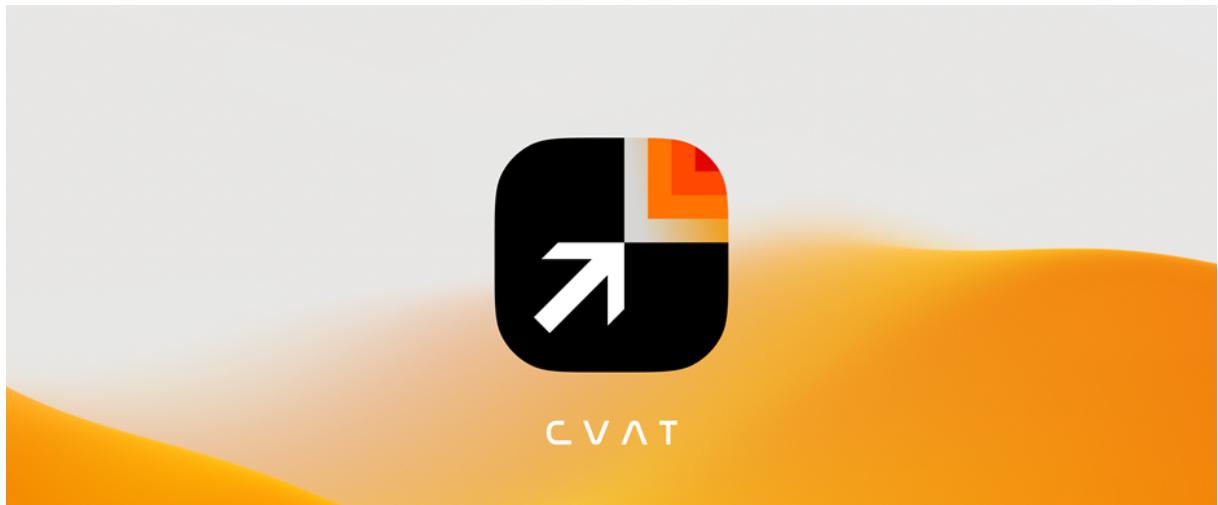
- **Hình ảnh:** Khoảng 9 triệu hình ảnh, thường thể hiện các cảnh phức tạp với trung bình 8,3 đối tượng trên mỗi hình ảnh.
- **Hộp giới hạn:** Hơn 16 triệu hộp phân định ranh giới các đối tượng trên 600 danh mục.
- **Mặt nạ phân đoạn:** Chúng nêu chi tiết ranh giới chính xác của 2.8M đối tượng trên 350 lớp.
- **Mối quan hệ trực quan:** 3,3 triệu chú thích cho biết mối quan hệ, thuộc tính và hành động của đối tượng.
- **Tường thuật được bản địa hóa:** 675k mô tả kết hợp dấu vết giọng nói, văn bản và chuột.
- **Nhãn cấp điểm:** 66,4 triệu nhãn trên 1,4 triệu hình ảnh, phù hợp với phân đoạn ngữ nghĩa không / ít ảnh.

4.3 TRANG WEB CVAT

Các nhà nghiên cứu có thể có được những hiểu biết vô giá về một loạt các thách thức thị giác máy tính mà bộ dữ liệu giải quyết, từ phát hiện đối tượng cơ bản đến xác định mối quan hệ phức tạp.

CVAT là một công cụ chú thích video và hình ảnh tương tác cho thị giác máy tính. Nó được sử dụng bởi hàng chục ngàn người dùng và các công ty trên khắp thế giới. Nhiệm vụ của chúng tôi là giúp các nhà phát triển, công ty và tổ chức trên toàn thế giới giải quyết các vấn đề thực tế bằng cách sử dụng phương pháp AI lấy dữ liệu làm trung tâm.

Bắt đầu sử dụng CVAT trực tuyến: cvat.ai. Bạn có thể sử dụng nó miễn phí, hoặc Đăng ký để nhận dữ liệu, tổ chức, chú thích tự động không giới hạn và Tích hợp Roboflow và HuggingFace.



Hình 4.3: Hình minh họa trang web CVAT

Hoặc thiết lập CVAT như một giải pháp tự lưu trữ: Hướng dẫn cài đặt tự lưu trữ. Chúng tôi cung cấp hỗ trợ Doanh nghiệp cho các cài đặt tự lưu trữ với các tính năng cao cấp: tích hợp SSO, LDAP, Roboflow và HuggingFace cũng như phân tích nâng cao (sắp ra mắt). Chúng tôi cũng thực hiện các khóa đào tạo và hỗ trợ tận tâm với SLA 24 giờ.

Chương 5

THƯ VIỆN THU THẬP DỮ LIỆU

5.1 THƯ VIỆN OIDV4_TOOLKIT

Thư viện OIDv4 Toolkit là một công cụ hỗ trợ trong việc làm việc với bộ dữ liệu Open Images Dataset (OID). Bộ dữ liệu Open Images là một tập dữ liệu rất lớn và đa dạng với hàng triệu hình ảnh được gắn nhãn cho nhiều loại đối tượng khác nhau.

OIDv4 Toolkit cung cấp các chức năng để tải xuống, xử lý và trích xuất dữ liệu từ bộ dữ liệu Open Images. Các chức năng chính của thư viện bao gồm:

- **Tải xuống dữ liệu:** Cho phép tải xuống các tệp tin hình ảnh và tệp tin nhãn (annotations) từ bộ dữ liệu Open Images.
- **Xử lý dữ liệu:** Cung cấp các chức năng để chuyển đổi và xử lý dữ liệu hình ảnh và nhãn. Ví dụ: chuyển đổi định dạng hình ảnh, chia tách tệp tin nhãn, lọc dữ liệu theo loại đối tượng, v.v.
- **Trích xuất dữ liệu:** Hỗ trợ trích xuất thông tin từ bộ dữ liệu Open Images như danh sách các lớp đối tượng, thống kê số lượng hình ảnh và nhãn, v.v.

5.2 THƯ VIỆN PYTUBE

YouTube là nền tảng chia sẻ video phổ biến nhất trên thế giới và là một hacker, bạn có thể gặp phải tình huống muốn viết kịch bản thứ gì đó để tải xuống video. Đối với điều này, tôi trình bày cho bạn: pytube.

pytube là một thư viện nhẹ được viết bằng Python. Nó không có sự phụ thuộc của bên thứ ba và nhắm mục đích có độ tin cậy cao.

PyTube cũng làm cho đường ống dễ dàng, cho phép bạn chỉ định các chức năng gọi lại cho các sự kiện tải xuống khác nhau, chẳng hạn như trên tiến trình hoặc khi hoàn thành.

Hơn nữa, pytube bao gồm một tiện ích dòng lệnh, cho phép bạn tải xuống video ngay từ thiết bị đầu cuối.

Các tính năng chính

- Hỗ trợ cho cả luồng tiến bộ & DASH
- Hỗ trợ tải xuống danh sách phát hoàn chỉnh
- Dễ dàng đăng ký on_download_progress &on_download_complete callback
- Hỗ trợ theo dõi phụ đề
- Xuất các bản nhạc phụ đề sang định dạng .srt (Phụ đề SubRip)
- Khả năng chụp URL hình thu nhỏ
- Mã nguồn được ghi chép rộng rãi
- Không có sự phụ thuộc của bên thứ ba

5.3 THƯ VIỆN BING_IMAGE_DOWNLOADER

Thư viện Bing Image Downloader là một công cụ Python cho phép tải xuống hình ảnh từ công cụ tìm kiếm Bing. Nó cung cấp cách dễ dàng để lấy hình ảnh từ Bing bằng cách tìm kiếm theo từ khóa và tải xuống các kết quả tìm kiếm.

Đây là một số tính năng và cách sử dụng cơ bản của thư viện Bing Image Downloader:

- **Tìm kiếm hình ảnh:** Bạn có thể sử dụng thư viện để tìm kiếm hình ảnh trên Bing bằng cách chỉ định từ khóa tìm kiếm.

- **Tải xuống hình ảnh:** Thư viện cho phép bạn tải xuống hình ảnh từ kết quả tìm kiếm Bing. Bạn có thể chỉ định số lượng hình ảnh cần tải xuống.

Chương 6

THIẾT KẾ DỮ LIỆU VÀ XÂY DỤNG MÔ HÌNH YOLOV8

6.1 THU THẬP DỮ LIỆU

6.1.1 OIDV4_TOOLKIT

```
from sys import exit
from textwrap import dedent
from modules.parser import *
from modules.utils import *
from modules.downloader import *
from modules.show import *
from modules.csv_downloader import *
from modules.bounding_boxes import *
from modules.image_level import *

ROOT_DIR = '...'
DEFAULT_OID_DIR = os.path.join(ROOT_DIR, 'OID')

if __name__ == '__main__':
    args = parser_arguments()
```

```
if args.command == 'downloader_ill':  
    image_level(args, DEFAULT_OID_DIR)  
else:  
    bounding_boxes_images(args, DEFAULT_OID_DIR)
```

Để chạy chương trình, bạn cần mở cửa sổ Command Prompt (cmd) và thực hiện lệnh sau:

```
python main.py downloader --classes Cat --type_csv train --limit 4
```

Trong đó:

- python là lệnh để chạy trình thông dịch Python.
- main.py là tên của file script bạn muốn chạy.
- downloader là đối số command để xác định hành động cụ thể của script.
- –classes Cat là đối số classes để chỉ định lớp (class) của các đối tượng bạn muốn tải xuống. Trong trường hợp này, lớp là "Cat" (Mèo).
- –type_csv train là đối số type_csv để chỉ định loại dữ liệu bạn muốn tải xuống từ file CSV. Trong trường hợp này, loại dữ liệu là "train" (huấn luyện).
- –limit 4 là đối số limit để chỉ định số lượng hình ảnh tối đa bạn muốn tải xuống. Trong trường hợp này, giới hạn là 4 hình ảnh.

Dữ liệu thu được bao gồm các hình ảnh và nhãn. Định dạng của nhãn là "Dog 174.14056200000002 96.0 482.726968 451.84". Tuy nhiên, để huấn luyện với YOLOv8, bạn cần sửa "Dog" thành "0" trong nhãn.

Phương pháp thu thập dữ liệu này phù hợp cho các nhiệm vụ như Phân loại Hình ảnh (Image Classification) và Phát hiện Đối tượng (Object Detection), khi dữ liệu không yêu cầu xử lý phức tạp.

6.1.2 BING_IMAGE_DOWNLOADER

```
from bing_image_downloader import downloader  
downloader.download("Cats", limit=100,  
output_dir='Data_Bing', adult_filter_off=True, force_replace=False,
```

```
timeout=60)
```

Trong đó:

- “Cats”: là nội dung, vật thể muốn tải xuống
- limit=100: Chỉ định số lượng hình ảnh tối đa mà bạn muốn tải xuống. Trong trường hợp này, giới hạn là 100 hình ảnh.
- output_dir='Data_Bing': Chỉ định thư mục đầu ra cho việc lưu trữ các hình ảnh đã tải xuống. Trong trường hợp này, thư mục đầu ra được đặt là 'Data_Bing'.
- adult_filter_off=True: Tắt bộ lọc nội dung người lớn để cho phép tải xuống các hình ảnh có thể liên quan đến người lớn (tùy thuộc vào kết quả tìm kiếm).
- force_replace=False: Không bắt buộc thay thế các hình ảnh đã tồn tại trong thư mục đầu ra. Nếu tùy chọn này được đặt là True, các hình ảnh đã tồn tại sẽ bị ghi đè.
- timeout=60: Đặt thời gian chờ tối đa (tính bằng giây) cho mỗi yêu cầu tải xuống hình ảnh.

Dữ liệu thu được từ việc tải xuống bao gồm các hình ảnh liên quan. Tuy nhiên, để huấn luyện với YOLOv8, bạn cần có các nhãn phù hợp với từng nhiệm vụ.

Phương pháp thu thập dữ liệu này phù hợp với hầu hết các nhiệm vụ của YOLOv8

6.1.3 PYTUBE

```
from pytube import YouTube
yt=YouTube(' ')
video = yt.streams.get_highest_resolution()
video.download()
```

Dữ liệu thu được từ việc tải xuống là video theo url đã cung cấp. Dữ liệu này được dùng để kiểm tra lại mô hình đã xây dựng.

6.2 XỬ LÝ DỮ LIỆU

6.2.1 Object Detection

Dữ liệu tải bằng OIVD4_TOOLKIT Do dữ liệu thu được từ thư viện trên có định dạng của nhãn là "Dog 174.14056200000002 96.0 482.726968 451.84". Tuy nhiên, để huấn luyện với YOLOv8, bạn cần sửa "Dog" thành "0" trong nhãn.

Phương trình chuyển đổi:

```
import os
import cv2
import numpy as np
from tqdm import tqdm
import argparse
import fileinput

ROOT_DIR = os.getcwd()

# create dict to map class names to numbers for yolo
classes = {}
with open("classes.txt", "r") as myFile:
    for num, line in enumerate(myFile, 0):
        line = line.rstrip("\n")
        classes[line] = num
myFile.close()

# step into dataset directory
os.chdir('C:/Users/hvthy/Desktop/DoAn2/SEGMENTATION/Data/Data_v7/Dataset/labels')
DIRS = os.getcwd()
DIRS1 = os.listdir(os.getcwd())

for DIR in DIRS1:
    if os.path.isdir(DIR):
```

```

os.chdir(DIR)

DIRS2 = os.getcwd()

print("Currently in subdirectory:", DIR)

for filename in tqdm(os.listdir(os.getcwd())):
    filename_path=os.path.join(DIRS2, filename)
    annotations = []
    with open(filename_path, 'r') as f:
        for line in f:
            for class_type in classes:
                line = line.replace(class_type,
                                    str(classes.get(class_type)))
            labels = line.split()
            coords = np.asarray([float(labels[1]),
                                float(labels[2]), float(labels[3]),
                                float(labels[4])])
            labels[1], labels[2], labels[3], labels[4] = coords[0],
            coords[1], coords[2], coords[3]
            newline = str(labels[0]) + " " + str(labels[1]) + " " +
            str(labels[2]) + " " + str(labels[3]) + " " +
            str(labels[4])
            line = line.replace(line, newline)
            annotations.append(line)
    f.close()
    with open(filename_path, "w", encoding='utf-8') as outfile:
        for line in annotations:
            outfile.write(line)
    outfile.close()
os.chdir(DIRS)

```

Dữ liệu tải bằng BING_IMAGE_DOWNLOADER Sử dụng web CVAT để tạo các label từ image nhập vào: Bước 1: Truy cập vào giao diện web của CVAT (Computer Vision Annotation Tool): <https://cvat.org/>

Bước 2: Tạo một dự án mới trên CVAT và tải lên các hình ảnh đã tải xuống từ bước 1.

Bước 3: Sử dụng các công cụ có sẵn trong CVAT để tạo các nhãn cho Pose Estimation:

- Sử dụng các công cụ vẽ hoặc đánh dấu để bao quanh các đối tượng trong hình ảnh mà bạn muốn nhận dạng.
- Với mỗi đối tượng, vẽ một hộp giới hạn (bounding box) xung quanh nó bằng cách kéo chuột từ góc trên bên trái đến góc dưới bên phải của đối tượng.
- Đảm bảo rằng bạn đã tạo các nhãn (bounding box) chính xác và đầy đủ cho tất cả các đối tượng trong hình ảnh.

Bước 4: Xuất khẩu nhãn từ CVAT: Sau khi hoàn thành việc tạo nhãn trên CVAT, xuất khẩu nhãn dưới dạng tệp tin YOLO 1.1

6.2.2 Instance Segmentation

Sử dụng web CVAT để tạo các label từ image nhập vào:

Bước 1: Truy cập vào giao diện web của CVAT (Computer Vision Annotation Tool):

<https://cvat.org/>

Bước 2: Tạo một dự án mới trên CVAT và tải lên các hình ảnh đã tải xuống từ bước 1.

Bước 3: Sử dụng các công cụ có sẵn trong CVAT để tạo các nhãn cho Instance Segmentation:

- Sử dụng công cụ vẽ để xác định vùng của các đối tượng và gán nhãn tương ứng cho mỗi vùng đó.
- Điều này sẽ tạo ra các nhãn cho mỗi đối tượng trong hình ảnh.

Bước 4: Xuất khẩu nhãn từ CVAT: Sau khi hoàn thành việc tạo nhãn trên CVAT, xuất khẩu nhãn dưới dạng tệp tin SEG MASK 1.1

6.2.3 Pose Estimation

Sử dụng web CVAT để tạo các label từ image nhập vào:

Bước 1: Truy cập vào giao diện web của CVAT (Computer Vision Annotation Tool):

<https://cvat.org/>

Bước 2: Tạo một dự án mới trên CVAT và tải lên các hình ảnh đã tải xuống từ bước 1.

Bước 3: Sử dụng các công cụ có sẵn trong CVAT để tạo các nhãn cho Pose Estimation:

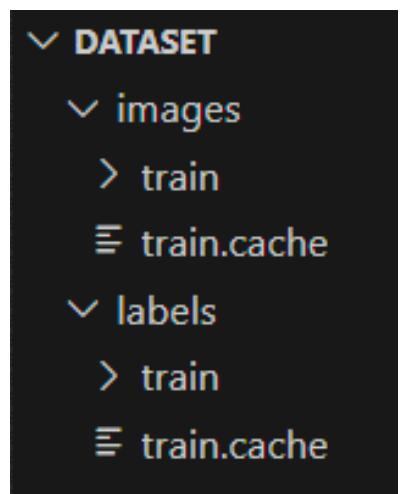
- Sử dụng các công cụ vẽ hoặc đánh dấu để chỉ định các điểm quan trọng trên cơ thể trong hình ảnh.
- Đảm bảo rằng bạn đã đánh dấu các điểm quan trọng chính xác và đầy đủ trong hình ảnh.

Bước 4: Xuất khẩu nhãn từ CVAT: Sau khi hoàn thành việc tạo nhãn trên CVAT, xuất khẩu nhãn dưới dạng tệp tin YOLO 1.1

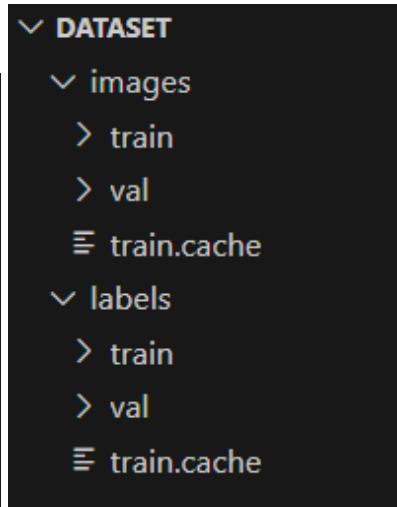
6.2.4 Image Classification

Để huấn luyện mô hình YOLOv8 cho nhiệm vụ Phân loại Hình ảnh (Image Classification), bạn chỉ cần sắp xếp dữ liệu thành các thư mục, mỗi thư mục đại diện cho một lớp (class) và chứa các hình ảnh tương ứng.

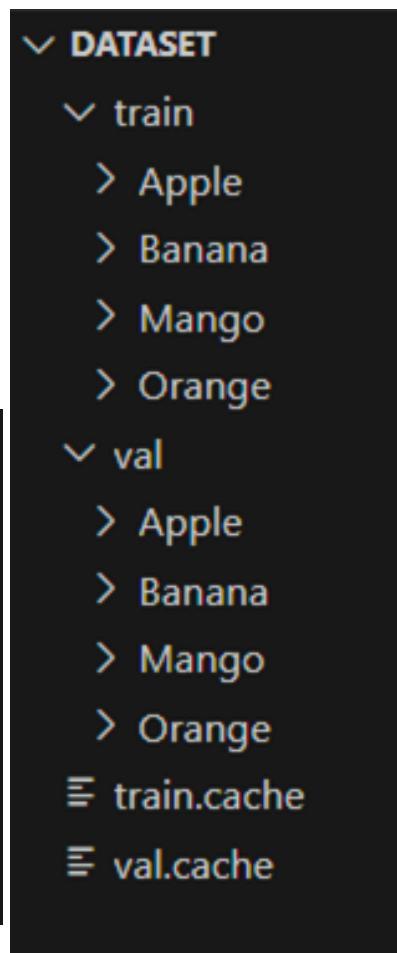
6.3 CẤU TRÚC TẬP DỮ LIỆU



(a) Object Detection



(b) Instance Segmentation,
Pose Estimation



(c) • Image Classification

Chương 7

XÂY DỰNG MÔ HÌNH YOLOV8

7.1 GPU Check

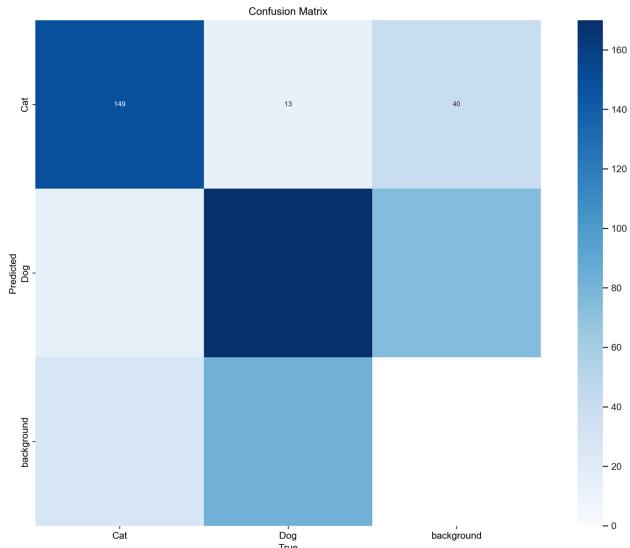
```
import torch
if torch.cuda.is_available():
    print('ok')
```

7.2 Object Detection

7.2.1 config.yaml

```
train: images/train # train images (relative to 'path')
val: images/val # val images (relative to 'path')

# Classes
names:
  0: Cat
  1: Dog
```



Hình 7.1: Hình minh họa confusion_matrix

7.2.2 train.py

```
from ultralytics import YOLO

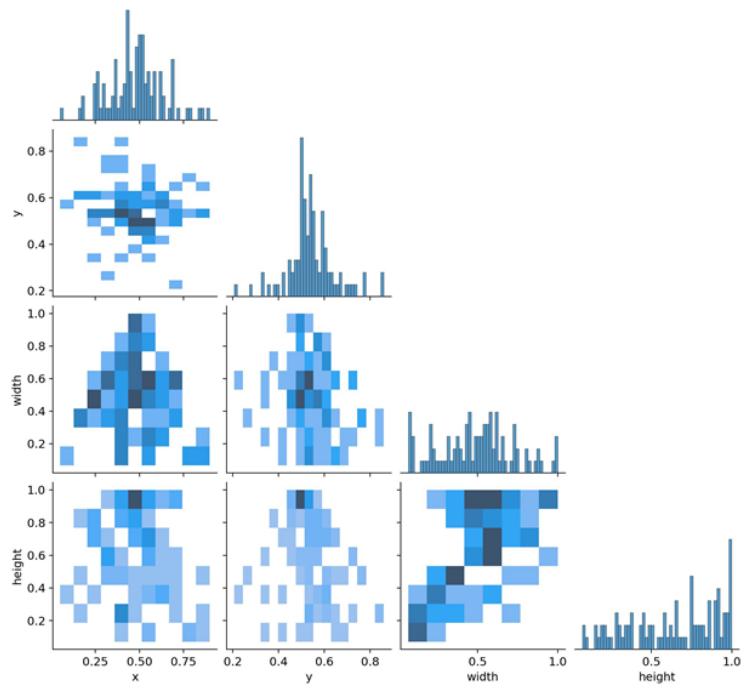
if __name__ == '__main__':
    # build a new model from scratch
    model = YOLO("yolov8n.yaml")
    results= model.train(data="config.yaml", epochs=100, device='0') #
        train the model
```

7.2.3 confusion_matrix

7.3 Instance Segmentation

7.3.1 config.yaml

```
path: ../Dataset
train: images/train
val: images/val
```



Hình 7.2: Hình minh họa labels_correlogram

```
nc: 1
names: ["Dog"]
```

7.3.2 train.py

```
from ultralytics import YOLO

if __name__ == '__main__':
    model = YOLO('yolov8n-seg.pt') # load a pretrained model (recommended
                                    for training)
    model.train(data='config.yaml', epochs=100, imgsz=640)
```

7.3.3 labels_correlogram

7.4 Pose Estimation

7.4.1 config.yaml

```
# Data
path: /media/veracrypt2/computer_vision/47_pose_detection_yolov8/code/data
train: images/train # train images (relative to 'path')
val: images/val # val images (relative to 'path')

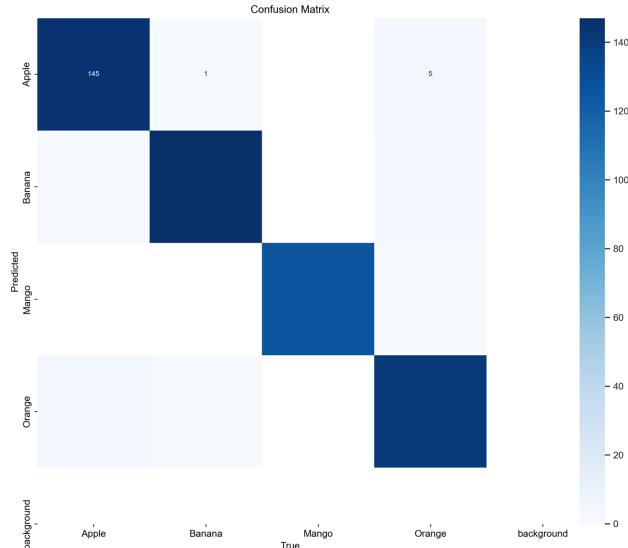
# Keypoints
kpt_shape: [39, 3] # [number of keypoints, number of dim]
flip_idx: [0, 1, 2, 4, 3, 10, 11, 12, 13, 14, 5, 6, 7, 8, 9, 15, 16, 17,
18, 19, 20, 21, 22, 23, 27, 28, 29, 24, 25,
26, 33, 34, 35, 30, 31, 32, 36, 38, 37]

# Classes
names:
0: quadruped
```

7.4.2 train.py

```
from ultralytics import YOLO

model = YOLO('yolov8n-pose.pt') # load a pretrained model (recommended
for training)
model.train(data='config.yaml', epochs=1, imgsz=640)
```



Hình 7.3: Hình minh họa confusion_matrix

7.5 Image Classification

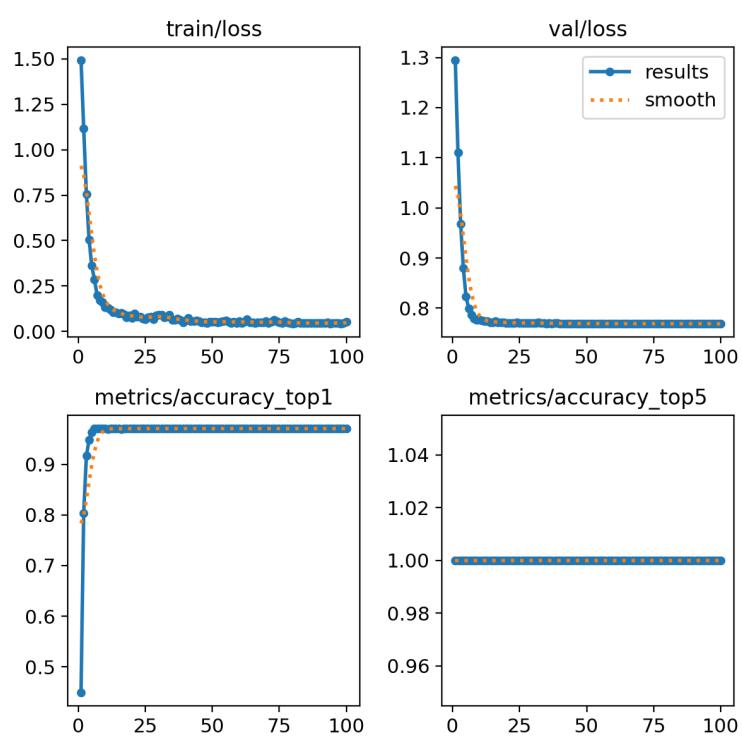
7.5.1 train.py

```
from ultralytics import YOLO

if __name__ == "__main__":
    model = YOLO('yolov8n-cls.pt') # load a pretrained model (recommended
                                    for training)
    model.train(data='C:/Users/hvthy/Desktop/DoAn2/CLASSIFICATION/Data/Dataset', epoch
                imgsz=64)
```

7.5.2 confusion_matrix

7.5.3 results



Hình 7.4: Hình minh họa results

Chương 8

KIỂM TRA MÔ HÌNH ĐÃ XÂY DỤNG

8.1 Object Detection

```
import os
from ultralytics import YOLO
import cv2

VIDEOS_DIR = 'C:/Users/hvthy/Desktop/DoAn2/DETECTION/Data/Data_mp4'
video_path = os.path.join(VIDEOS_DIR, 'Cat1.mp4')
video_path_out = '{}_out.mp4'.format(video_path)
cap = cv2.VideoCapture(video_path)
ret, frame = cap.read()
H, W, _ = frame.shape
out = cv2.VideoWriter(video_path_out, cv2.VideoWriter_fourcc('m', 'p',
    '4', 'v'), int(cap.get(cv2.CAP_PROP_FPS)), (W, H))

# Load a model
model = YOLO('../weights/best.pt') # load a custom model
threshold = 0.5
```

```

while ret:

    results = model(frame)[0]
    for result in results.boxes.data.tolist():
        x1, y1, x2, y2, score, class_id = result
        if score > threshold:
            cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)),
                          (0, 255, 0), 4)
            cv2.putText(frame, results.names[int(class_id)].upper(),
                        (int(x1), int(y1 - 10)),
                        cv2.FONT_HERSHEY_SIMPLEX, 1.3, (0, 255, 0), 3,
                        cv2.LINE_AA)

    out.write(frame)
    ret, frame = cap.read()

cap.release()
out.release()
cv2.destroyAllWindows()

```

8.2 Instance Segmentation

```

from ultralytics import YOLO
import cv2

model_path = '../weights/last.pt'
image_path = '... /images/val/1be566eccffe9561.png'
img = cv2.imread(image_path)
H, W, _ = img.shape
model = YOLO(model_path)
results = model(img)

for result in results:

```

```
for j, mask in enumerate(result.masks.data):
    mask = mask.numpy() * 255
    mask = cv2.resize(mask, (W, H))
    cv2.imwrite('./output.png', mask)
```

8.3 Pose Estimation

```
from ultralytics import YOLO
import cv2

model_path = '../weights/last.pt'
image_path = './samples/wolf.jpg'
img = cv2.imread(image_path)
model = YOLO(model_path)
results = model(image_path)[0]

for result in results:
    for keypoint_idx, keypoint in enumerate(result.keypoints.tolist()):
        cv2.putText(img, str(keypoint_idx), (int(keypoint[0]),
                                             int(keypoint[1])),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

cv2.imshow('img', img)
cv2.waitKey(0)
```

8.4 Image Classification

```
from ultralytics import YOLO
import numpy as np
```

```
model = YOLO('..../weights/last.pt') # load a custom model
results = model('..../Banana.jpg') # predict on an image
names_dict = results[0].names
probs = results[0].probs.data.tolist()

print(names_dict)
print(probs)
print(names_dict[np.argmax(probs)])
```

Chương 9

THIẾT KẾ VÀ XÂY DỰNG ỨNG DỤNG YOLOV8

9.1 Thiết kế giao diện

```
from pathlib import Path
from PIL import Image
import streamlit as st

import config
from utils import load_model, infer_uploaded_image, infer_uploaded_video,
infer_uploaded_webcam

# setting page layout
st.set_page_config(
    page_title="Interactive Interface for YOLOv8",
    page_icon="",
    layout="wide",
    initial_sidebar_state="expanded"
)
```

```

# main page heading
st.title("Interactive Interface for YOLOv8")

# sidebar
st.sidebar.header("DL Model Config")

# model options
task_type = st.sidebar.selectbox(
    "Select Task",
    ["Detection",
     "Classification",
     "Segmentation",
     "Tracking",
     "Pose Estimation"]
)

model_type = None
if task_type == "Detection":
    model_type = st.sidebar.selectbox(
        "Select Model",
        config.DETECTION_MODEL_LIST
    )
elif task_type == "Classification":
    model_type = st.sidebar.selectbox(
        "Select Model",
        config.CLASS_MODEL_LIST
    )
elif task_type == "Segmentation":
    model_type = st.sidebar.selectbox(
        "Select Model",
        config.SEG_MODEL_LIST
    )
elif task_type == "Tracking":

```

```

model_type = st.sidebar.selectbox(
    "Select Model",
    config.DETECTION_MODEL_LIST
)

elif task_type == "Pose Estimation":
    model_type = st.sidebar.selectbox(
        "Select Model",
        config.POSE_MODEL_LIST
    )

else:
    st.error("Currently only 'Detection' function is implemented")

confidence = float(st.sidebar.slider(
    "Select Model Confidence", 30, 100, 50)) / 100

model_path = ""
if model_type == 'best.pt':
    model_path =
        'C:/Users/hvthy/Desktop/DoAn2/DETECTION/TrainYOLO/runs/detect/train/weights/best.pt'
else:
    model_path = Path(config.MODEL_DIR, str(model_type))

# load pretrained DL model
model = load_model(model_path)

# image/video options
st.sidebar.header("Image/Video Config")
source_selectbox = st.sidebar.selectbox(
    "Select Source",
    config.SOURCES_LIST
)

source_img = None

```

```

if source_selectbox == config.SOURCES_LIST[0]: # Image
    infer_uploaded_image(confidence, model, task_type)
elif source_selectbox == config.SOURCES_LIST[1]: # Video
    infer_uploaded_video(confidence, model)
elif source_selectbox == config.SOURCES_LIST[2]: # Webcam
    infer_uploaded_webcam(confidence, model, task_type)
else:
    st.error("Currently only 'Image' and 'Video' source are implemented")

```

Mã trên triển khai một ứng dụng Streamlit cho giao diện tương tác với mô hình YOLOv8.

Dưới đây là mô tả về cách thiết kế giao diện trong mã trên:

1. Thiết lập giao diện trang:

- Sử dụng st.set_page_config() để thiết lập tiêu đề trang, biểu tượng trang, bố cục và trạng thái thanh bên (mở rộng hoặc thu gọn).
- Sử dụng st.title() để hiển thị tiêu đề chính của trang.

2. Thanh bên:

- Sử dụng st.sidebar.header() để hiển thị tiêu đề thanh bên.
- Sử dụng st.sidebar.selectbox() để tạo một hộp chọn cho người dùng chọn loại nhiệm vụ (Detection, Classification, Segmentation, Tracking, Pose Estimation).
- Dựa vào loại nhiệm vụ được chọn, sử dụng st.sidebar.selectbox() để tạo một hộp chọn khác để người dùng chọn mô hình tương ứng.
- Sử dụng st.sidebar.slider() để tạo một thanh trượt cho người dùng chọn ngưỡng độ tin cậy của mô hình.

3. Tải và khởi tạo mô hình:

- Dựa vào mô hình được chọn, xác định đường dẫn tới mô hình và sử dụng load_model() để tải mô hình YOLOv8.

4. Cấu hình hình ảnh/video:

- Sử dụng st.sidebar.header() để hiển thị tiêu đề thanh bên.
- Sử dụng st.sidebar.selectbox() để tạo một hộp chọn cho người dùng chọn nguồn (Image, Video, Webcam).

- Dựa vào nguồn được chọn, gọi các hàm infer_uploaded_image(), infer_uploaded_video(), hoặc infer_uploaded_webcam() để thực hiện dự đoán trên ảnh, video hoặc webcam.

Mã trên cung cấp một giao diện tương tác cho người dùng chọn loại nhiệm vụ, mô hình, ngưỡng độ tin cậy và nguồn dữ liệu để thực hiện các tác vụ như phát hiện, phân loại, phân đoạn, theo dõi hoặc ước tính tư thế bằng mô hình YOLOv8.

9.2 Cấu hình ứng dụng YOLOv8

```
from pathlib import Path
import sys

# Get the absolute path of the current file
file_path = Path(__file__).resolve()

# Get the parent directory of the current file
root_path = file_path.parent

# Add the root path to the sys.path list if it is not already there
if root_path not in sys.path:
    sys.path.append(str(root_path))

# Get the relative path of the root directory with respect to the current
# working directory
ROOT = root_path.relative_to(Path.cwd())

# Source
SOURCES_LIST = ["Image", "Video", "Webcam"]

# DL model config
MODEL_DIR = ROOT

YOLOv8n = MODEL_DIR / "weights/detection/yolov8n.pt"
YOLOv8s = MODEL_DIR / "weights/detection/yolov8s.pt"
YOLOv8m = MODEL_DIR / "weights/detection/yolov8m.pt"
YOLOv8l = MODEL_DIR / "weights/detection/yolov8l.pt"
YOLOv8x = MODEL_DIR / "weights/detection/yolov8x.pt"
CustomDatapose = "../runs/detect/train/weights/best.pt"
DETECTION_MODEL_LIST = [
```

```

"yolov8n.pt",
"yolov8s.pt",
"yolov8m.pt",
"yolov8l.pt",
"yolov8x.pt",
"best.pt"]

YOL0v8ncls = MODEL_DIR / "weights/detection/yolov8n-cls.pt"
YOL0v8scls = MODEL_DIR / "weights/detection/yolov8s-cls.pt"
YOL0v8mcls = MODEL_DIR / "weights/detection/yolov8m-cls.pt"
YOL0v8lcls = MODEL_DIR / "weights/detection/yolov8l-cls.pt"
YOL0v8xcls = MODEL_DIR / "weights/detection/yolov8x-cls.pt"
CustomDatapose = "../runs/detect/train/weights/best.pt"
CLASS_MODEL_LIST = [
    "yolov8n-cls.pt",
    "yolov8s-cls.pt",
    "yolov8m-cls.pt",
    "yolov8l-cls.pt",
    "yolov8x-cls.pt",
    "best-cls.pt"]

YOL0v8nseg = MODEL_DIR / "weights/detection/yolov8n-seg.pt"
YOL0v8sseg = MODEL_DIR / "weights/detection/yolov8s-seg.pt"
YOL0v8mseg = MODEL_DIR / "weights/detection/yolov8m-seg.pt"
YOL0v8lseg = MODEL_DIR / "weights/detection/yolov8l-seg.pt"
YOL0v8xseg = MODEL_DIR / "weights/detection/yolov8x-seg.pt"
CustomDataseg = "../runs/detect/train/weights/best.pt"
SEG_MODEL_LIST = [
    "yolov8n-seg.pt",
    "yolov8s-seg.pt",
    "yolov8m-seg.pt",
    "yolov8l-seg.pt",
    "yolov8x-seg.pt",
]

```

```

"best-seg.pt"]

YOL0v8npose = MODEL_DIR / "weights/detection/yolov8n-pose.pt"
YOL0v8spose = MODEL_DIR / "weights/detection/yolov8s-pose.pt"
YOL0v8mpose = MODEL_DIR / "weights/detection/yolov8m-pose.pt"
YOL0v8lpose = MODEL_DIR / "weights/detection/yolov8l-pose.pt"
YOL0v8xpose = MODEL_DIR / "weights/detection/yolov8x-pose.pt"
CustomDatapose = "../../runs/detect/train/weights/best.pt"
POSE_MODEL_LIST = [
    "yolov8n-pose.pt",
    "yolov8s-pose.pt",
    "yolov8m-pose.pt",
    "yolov8l-pose.pt",
    "yolov8x-pose.pt",
    "best-pose.pt"]

```

Mã trên định nghĩa một số biến cấu hình được sử dụng trong ứng dụng Streamlit.

Dưới đây là mô tả của các phần chính trong mã:

1. Đường dẫn và thư mục:

- Sử dụng Path(__file__) để lấy đường dẫn tuyệt đối của tệp hiện tại.
- Sử dụng .parent để lấy thư mục cha của tệp hiện tại.
- Sử dụng sys.path.append() để thêm đường dẫn gốc vào danh sách sys.path nếu nó chưa tồn tại.
- Sử dụng .resolve() để đảm bảo đường dẫn tệp là tuyệt đối.
- Sử dụng .relative_to(Path.cwd()) để lấy đường dẫn tương đối của thư mục gốc đối với thư mục làm việc hiện tại.

2. Danh sách nguồn dữ liệu:

- SOURCES_LIST là một danh sách chứa các tùy chọn nguồn dữ liệu (Image, Video, Webcam).

3. Cấu hình mô hình Deep Learning:

- MODEL_DIR là đường dẫn đến thư mục chứa các mô hình.

- Các biến như YOLOv8n, YOLOv8s,..., CustomData là đường dẫn tuyệt đối đến các mô hình trong thư mục MODEL_DIR/weights/detection.
- DETECTION_MODEL_LIST là một danh sách chứa tên các mô hình phát hiện.
- Tương tự, có các biến và danh sách tương ứng cho mô hình phân loại (CLASS_MODEL_LIST), mô hình phân đoạn (SEG_MODEL_LIST), và mô hình ước tính tư thế (POSE_MODEL_LIST). Các biến và danh sách này được sử dụng trong mã trước để xác định đường dẫn tới mô hình được chọn từ giao diện người dùng.

9.3 Xây dựng ứng dụng YOLOv8

```

from ultralytics import YOLO
import streamlit as st
import cv2
from PIL import Image
import tempfile

def _display_detected_frames(conf, model, st_frame, image, type):
    # Resize the image to a standard size
    image = cv2.resize(image, (720, int(720 * (9 / 16))))
    # Predict the objects in the image using YOL0v8 model
    res = model.predict(image, conf=conf)
    # Plot the detected objects on the video frame
    res_plotted = res[0].plot()
    if type == "Detection":
        st_frame.image(res_plotted,
                        caption='Detected Video',
                        channels="BGR",
                        use_column_width=True
                    )
    elif type == "Classification":
        st_frame.image(res_plotted,
                        caption='Classified Video',

```

```

        channels="BGR",
        use_column_width=True
    )

elif type == "Segmentation":
    st_frame.image(res_plotted,
                    caption='Segmented Video',
                    channels="BGR",
                    use_column_width=True
    )

elif type == "Tracking":
    st_frame.image(res_plotted,
                    caption='Tracked Video',
                    channels="BGR",
                    use_column_width=True
    )

elif type == "Pose Estimation":
    st_frame.image(res_plotted,
                    caption='Estimated Video',
                    channels="BGR",
                    use_column_width=True
    )

@st.cache_resource
def load_model(model_path):
    model = YOLO(model_path)
    return model

def infer_uploaded_image(conf, model, type):
    source_img = st.sidebar.file_uploader(
        label="Choose an image...",
        type=(".jpg", ".jpeg", ".png", '.bmp', '.webp')
    )
    col1, col2 = st.columns(2)

```

```

with col1:

    if source_img:
        uploaded_image = Image.open(source_img)
        # adding the uploaded image to the page with caption
        st.image(
            image=source_img,
            caption="Uploaded Image",
            use_column_width=True
        )

if source_img:
    if st.button("Execution"):
        with st.spinner("Running..."):
            res = model.predict(uploaded_image,
                                conf=conf)

            boxes = res[0].boxes
            res_plotted = res[0].plot()[:, :, ::-1]

with col2:
    if type == "Detection":
        st.image(res_plotted,
                  caption="Detected Image",
                  use_column_width=True)

    try:
        with st.expander("Detection Results"):
            for box in boxes:
                st.write(box.xywh)

    except Exception as ex:
        st.write("No image is uploaded yet!")
        st.write(ex)

elif type == "Classification":
    st.image(res_plotted,

```

```

        caption="Classified Image",
        use_column_width=True)

    elif type == "Segmentation":
        st.image(res_plotted,
                  caption="Segmented Image",
                  use_column_width=True)

    elif type == "Tracking":
        st.image(res_plotted,
                  caption="Tracked Image",
                  use_column_width=True)

    elif type == "Pose Estimation":
        st.image(res_plotted,
                  caption="Estimated Image",
                  use_column_width=True)

    else:
        st.image(res_plotted,
                  caption="Trained Image",
                  use_column_width=True)

def infer_uploaded_video(conf, model):
    source_video = st.sidebar.file_uploader(
        label="Choose a video...")
    if source_video:
        st.video(source_video)

    if source_video:
        if st.button("Execution"):

```

```

with st.spinner("Running..."):

    try:
        tfile = tempfile.NamedTemporaryFile()
        tfile.write(source_video.read())
        vid_cap = cv2.VideoCapture(
            tfile.name)
        st_frame = st.empty()
        while (vid_cap.isOpened()):
            success, image = vid_cap.read()
            if success:
                _display_detected_frames(conf,
                                         model,
                                         st_frame,
                                         image
                                         )
            else:
                vid_cap.release()
                break
        except Exception as e:
            st.error(f"Error loading video: {e}")

def infer_uploaded_webcam(conf, model, type):
    try:
        flag = st.button(
            label="Stop running"
        )
        vid_cap = cv2.VideoCapture(0) # local camera
        st_frame = st.empty()
        while not flag:
            success, image = vid_cap.read()
            if success:
                _display_detected_frames(
                    conf,

```

```

        model,
        st_frame,
        image,
        type
    )
else:
    vid_cap.release()
    break
except Exception as e:
    st.error(f"Error loading video: {str(e)}")

```

Trong mã trên, có một số hàm và chức năng được triển khai để thực hiện việc nhận dạng đối tượng, phân loại ảnh, phân đoạn đối tượng, theo dõi đối tượng và ước tính tư thế bằng mô hình YOLOv8. Dưới đây là mô tả của từng hàm:

- conf: Ngưỡng độ tin cậy (confidence threshold) để nhận dạng đối tượng.
- model: Một đối tượng YOLOv8 chứa mô hình YOLOv8.
- st_frame: Đối tượng Streamlit để hiển thị video đã phát hiện.
- image: Một mảng numpy biểu diễn khung hình video.
- type: Loại nhiệm vụ, bao gồm "Detection", "Classification", "Segmentation", "Tracking", "Pose Estimation".

1. `_display_detected_frames(conf, model, st_frame, image, type)`: Đây là một hàm hỗ trợ để hiển thị các đối tượng đã được phát hiện trên một khung hình video bằng cách sử dụng mô hình YOLOv8 2. `load_model(model_path)`: Hàm này được sử dụng để tải mô hình YOLOv8 từ đường dẫn `model_path`. Hàm trả về một đối tượng mô hình YOLOv8. 3. `infer_uploaded_image(conf, model, type)`: Hàm này thực hiện việc dự đoán trên ảnh đã được tải lên. 4. `infer_uploaded_video(conf, model)`: Hàm này thực hiện việc dự đoán trên video đã được tải lên. 5. `infer_uploaded_webcam(conf, model, type)`: Hàm này thực hiện việc dự đoán trên webcam.

Các hàm trên có thể được sử dụng trong ứng dụng Streamlit để hiển thị kết quả từ mô hình YOLOv8 thông qua giao diện người dùng.

Chương 10

KẾT QUẢ, PHÂN TÍCH ĐÁNH GIÁ VÀ ĐỊNH HƯỚNG PHÁT TRIỂN TRONG TƯƠNG LAI

10.1 Kết quả sản phẩm ứng dụng YOLOv8

10.1.1 Giao diện

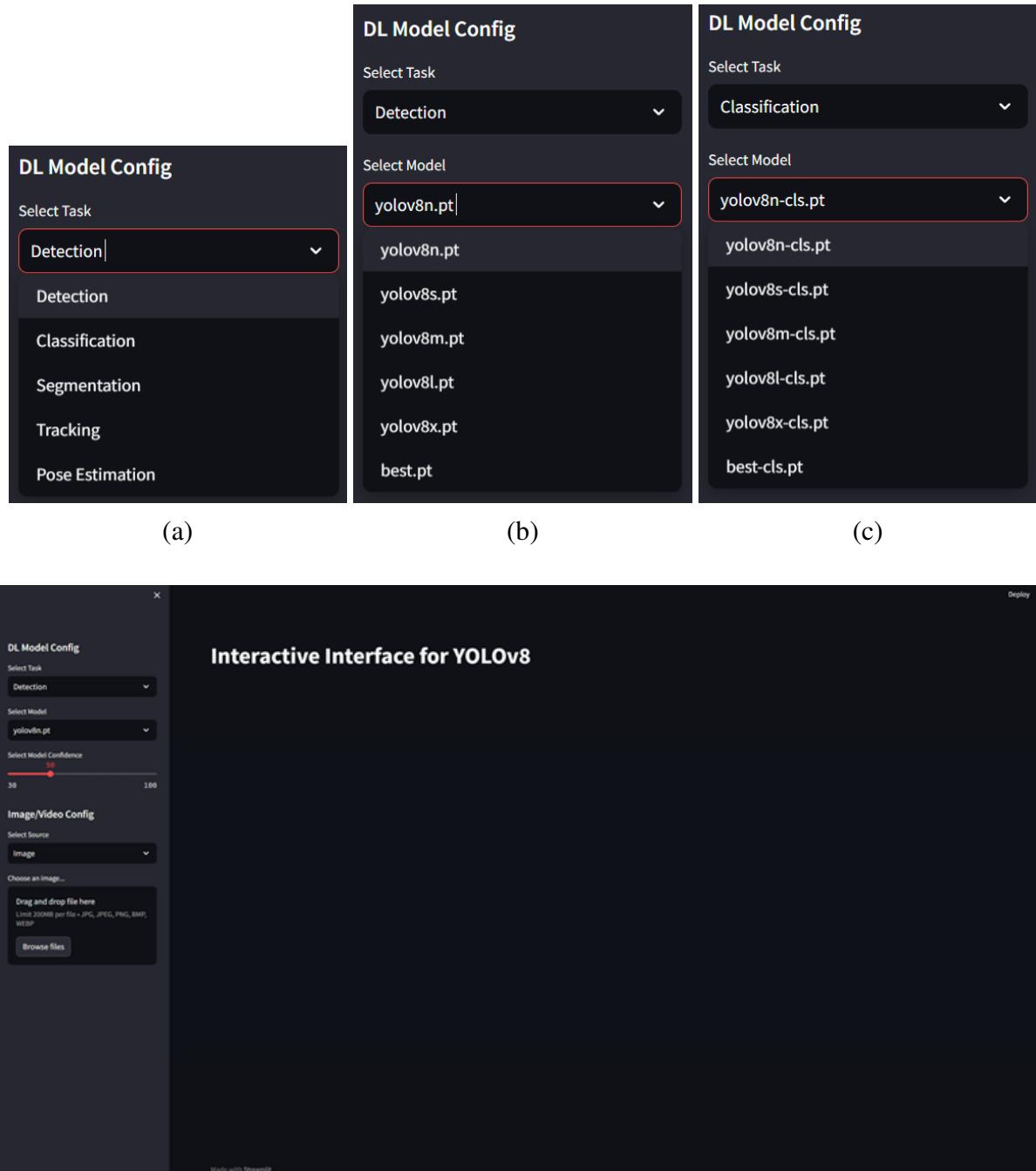
10.1.2 Kết quả

10.2 Phân tích và đánh giá kết quả

Dù vẫn còn ba phần chưa thực hiện được trong quá trình thu thập dữ liệu, bao gồm Instance Segmentation, Pose Estimation và Tracking Object, tuy nhiên, phần xử lý dữ liệu và cấu trúc dữ liệu đã được hoàn thành tốt trên tất cả các tác vụ.

Đối với việc xây dựng mô hình, vẫn còn một phần Tracking Object chưa hoàn thiện. Mặc dù đã xây dựng các mô hình YOLOv8, nhưng Instance Segmentation vẫn gặp lỗi khi hiển thị quá nhiều kết quả cùng một lúc. Còn Pose Estimation thì chưa thể thực hiện được với dữ liệu ảnh.

Mặc dù còn những khuyết điểm này, nhưng những thành tựu đã đạt được trong việc



Hình 10.2: Giao diện ứng dụng YOLOv8

xử lý dữ liệu và cấu trúc dữ liệu trên các tác vụ khác đáng được khen ngợi. Cần tiếp tục nỗ lực để giải quyết các vấn đề còn lại và hoàn thiện các phần chưa được thực hiện.

DL Model Config

Select Task

Segmentation

Select Model

yolov8n-seg.pt

yolov8n-seg.pt

yolov8s-seg.pt

yolov8m-seg.pt

yolov8l-seg.pt

yolov8x-seg.pt

best-seg.pt

DL Model Config

Select Task

Tracking

Select Model

yolov8n.pt

yolov8n.pt

yolov8s.pt

yolov8m.pt

yolov8l.pt

yolov8x.pt

best.pt

DL Model Config

Select Task

Pose Estimation

Select Model

yolov8n-pose.pt

yolov8n-pose.pt

yolov8s-pose.pt

yolov8m-pose.pt

yolov8l-pose.pt

yolov8x-pose.pt

best-pose.pt

(a)

(b)

(c)

Image/Video Config

Select Source

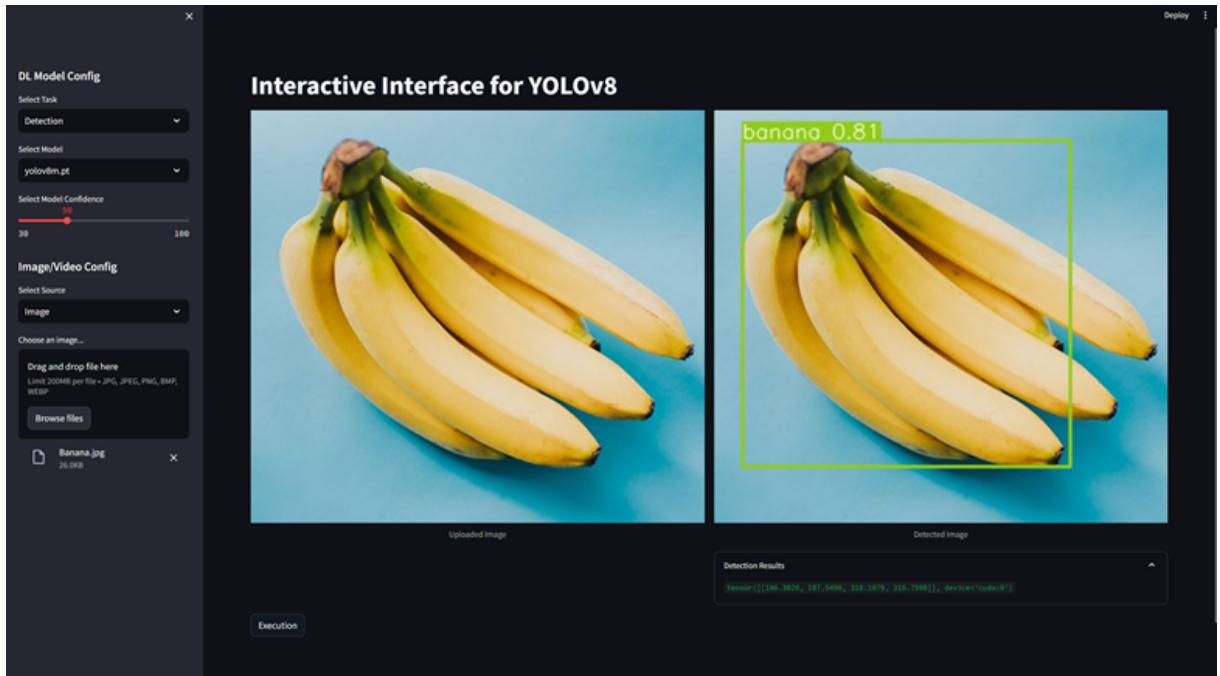
Image

Image

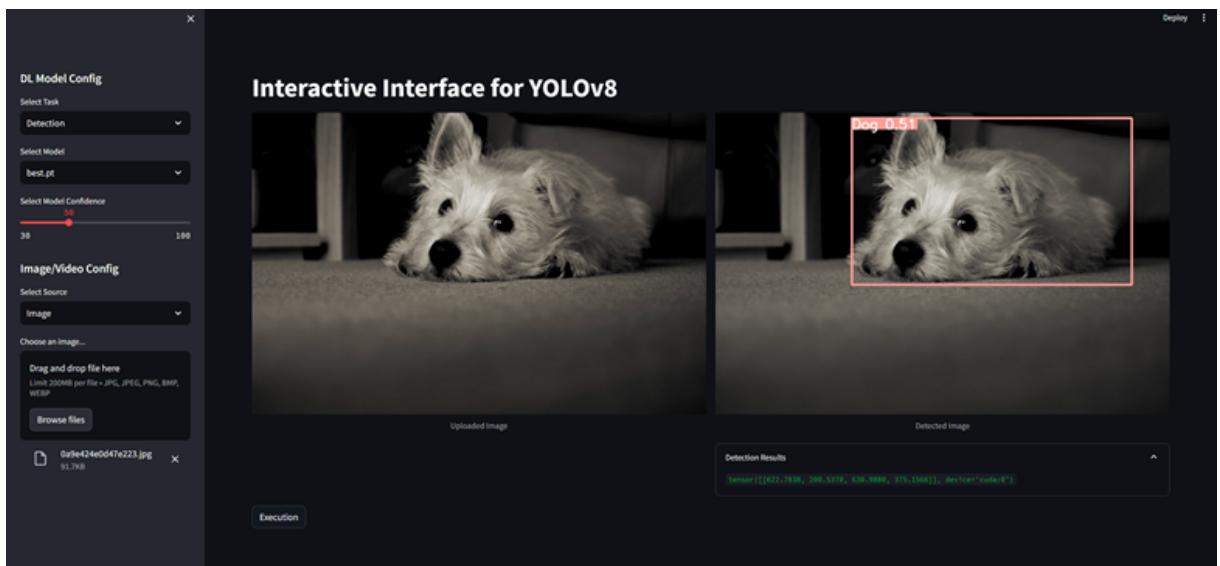
Video

Webcam

(d)



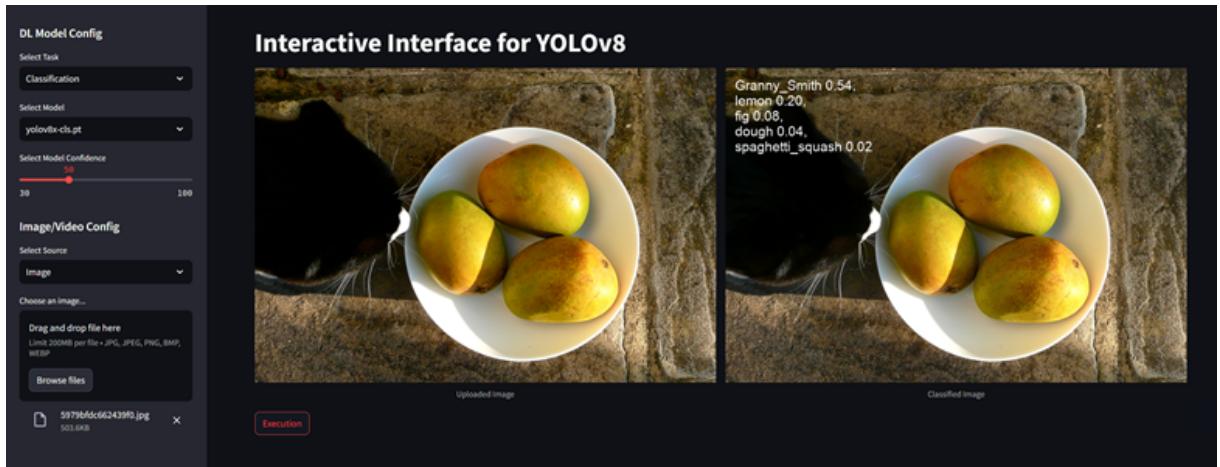
Hình 10.4: Kết quả



Hình 10.5: Kết quả

10.3 Định hướng phát triển trong tương lai

Hoàn thiện Instance Segmentation: Tiếp tục nghiên cứu và phát triển mô hình để giải quyết vấn đề hiển thị quá nhiều kết quả cùng một lúc trong Instance Segmentation. Cải thiện độ chính xác và hiệu suất của mô hình để đảm bảo tính đáng tin cậy trong việc phân đoạn các đối tượng trong ảnh.



Hình 10.6: Kết quả

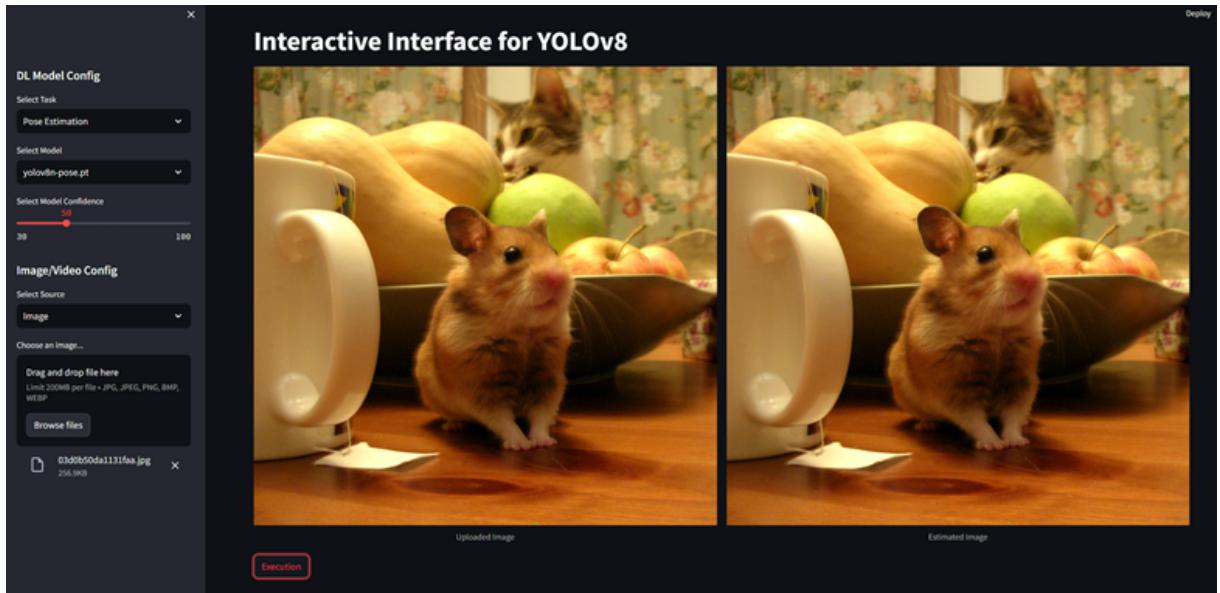


Hình 10.7: Kết quả

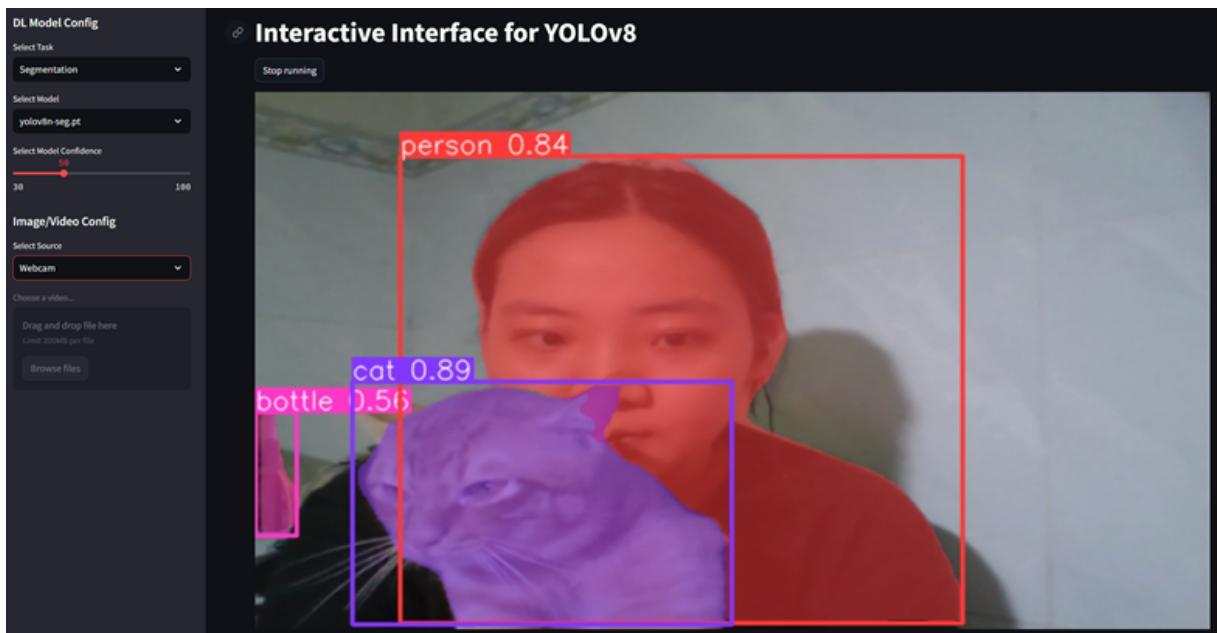
Đạt được Pose Estimation với dữ liệu ảnh: Tiếp tục nghiên cứu và phát triển các phương pháp và mô hình để thực hiện Pose Estimation trên dữ liệu ảnh. Điều này có thể đòi hỏi việc sử dụng các mô hình nâng cao và tập dữ liệu phong phú để học và dự đoán vị trí và góc xoay của các đối tượng trong ảnh.

Hoàn thiện Tracking Object: Để hoàn thiện phần Tracking Object, bạn có thể nghiên cứu và áp dụng các phương pháp và thuật toán để theo dõi và nhận diện đối tượng trong các khung hình liên tiếp. Điều này có thể bao gồm việc sử dụng các mô hình theo dõi đối tượng cơ bản như Kalman Filter hoặc sử dụng các mô hình học sâu như DeepSORT để đảm bảo tính chính xác và ổn định trong việc theo dõi đối tượng.

Nâng cao mô hình YOLOv8: Tiếp tục nghiên cứu và phát triển mô hình YOLOv8 để cải thiện hiệu suất và độ chính xác của Instance Segmentation. Bạn có thể xem xét việc



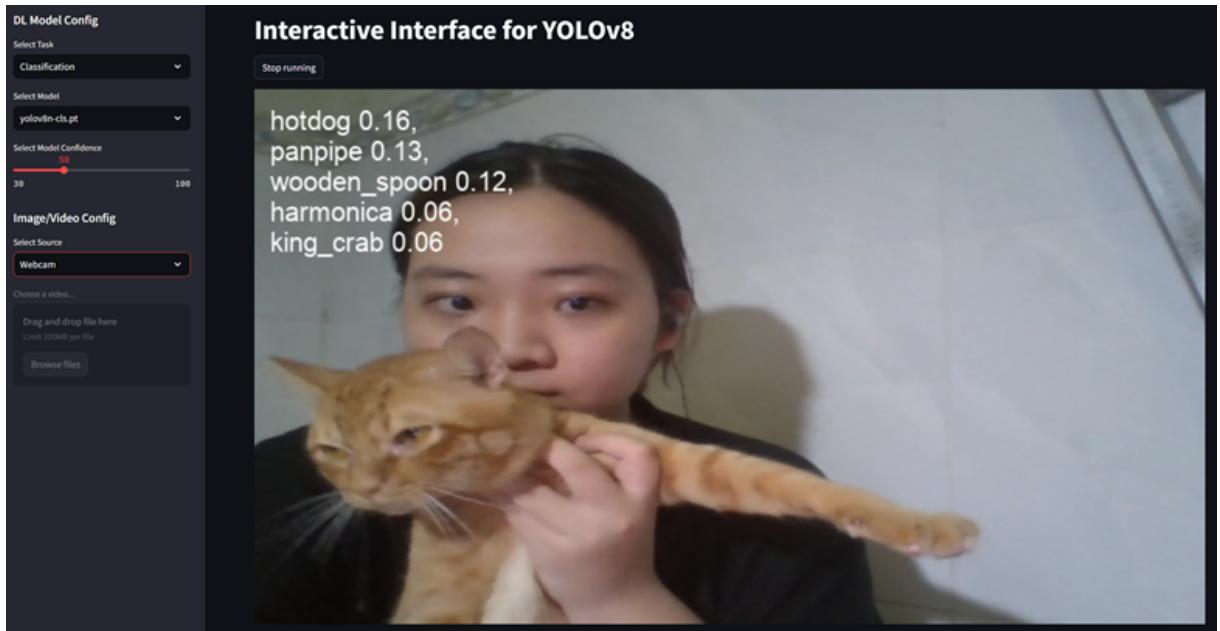
Hình 10.8: Kết quả



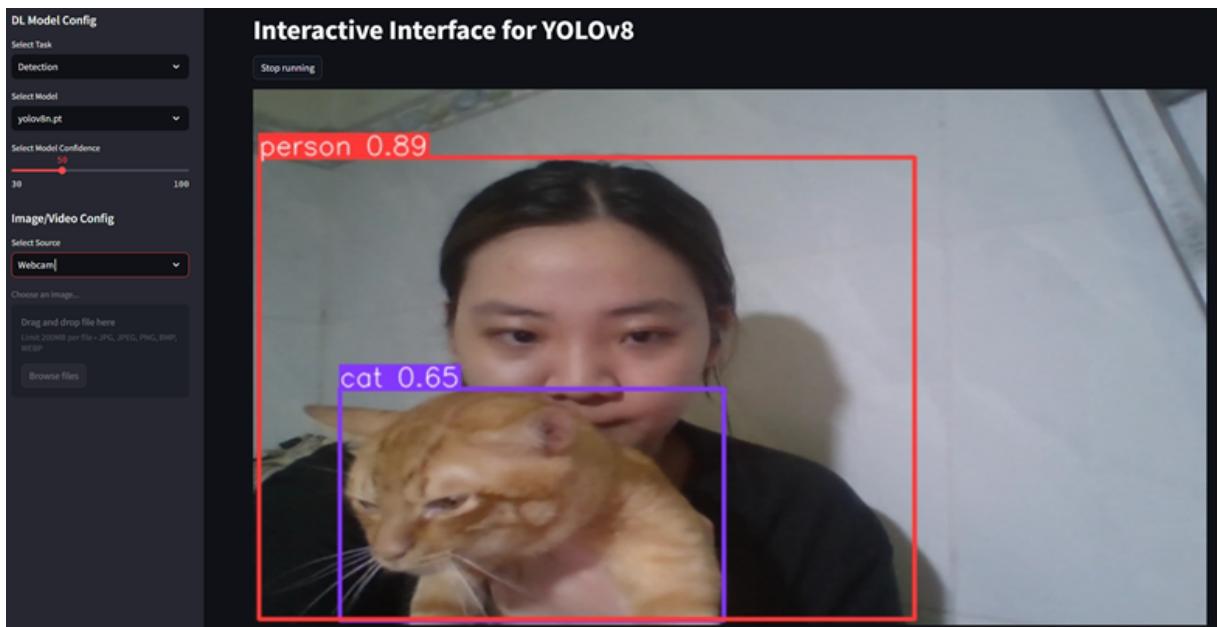
Hình 10.9: Kết quả

tăng cường kiến trúc mô hình, tối ưu hóa quá trình huấn luyện và thử nghiệm, và áp dụng các kỹ thuật tiên tiến để nâng cao khả năng phân đoạn đối tượng trong ảnh.

Kiểm tra và đánh giá mô hình: Tiến hành các bài kiểm tra và đánh giá chi tiết trên mô hình đã xây dựng, bao gồm cả phần Tracking Object. Điều này giúp xác định độ chính xác, hiệu suất và sự ổn định của mô hình trong các tình huống thực tế và tìm ra các điểm yếu cần cải thiện.



Hình 10.10: Kết quả



Hình 10.11: Kết quả

Cần lưu ý rằng việc phát triển trong tương lai đòi hỏi sự nghiên cứu tiếp tục, thử nghiệm và cải tiến liên tục. Đồng thời, việc sử dụng tập dữ liệu đa dạng và chất lượng cao cũng là yếu tố quan trọng để đảm bảo tính đáng tin cậy và hiệu suất của các mô hình.