**TRƯỜNG ĐẠI HỌC HÀNG HẢI VIỆT NAM**

**VIỆN ĐÀO TẠO CHẤT LƯỢNG CAO**



**BÁO CÁO BÀI TẬP LỚN**

**HỌC PHẦN THỊ GIÁC MÁY TÍNH**

**ĐỀ TÀI**

**XÂY DỰNG CHƯƠNG TRÌNH PHÁT HIỆN NGƯỜI ĐEO KHẨU TRANG**

**Giảng viên hướng dẫn: Nguyễn Hữu Tuân**

**Sinh Viên Thực Hiện : Ngô Quốc Khánh - 92950**

**Hải Phòng - Năm 2024**

**MỤC LỤC**

[DANH MỤC HÌNH ẢNH ii](#_Toc186136336)

[LỜI MỞ ĐẦU iii](#_Toc186136337)

[CHƯƠNG I: NỘI DUNG LÝ THUYẾT 1](#_Toc186136338)

[Tổng quan về thị giác máy tính 1](#_Toc186136339)

[1. Giới thiệu về thị giác máy tính 1](#_Toc186136340)

[2. Cách thức hoạt động của Thị Giác Máy Tính 1](#_Toc186136341)

[3. Ứng dụng thực tiễn của thị giác máy tính 2](#_Toc186136342)

[CHƯƠNG II: PHÂN TÍCH BÀI TOÁN 4](#_Toc186136343)

[1. Tổng Quan về Bài Toán 4](#_Toc186136344)

[2. Kiến Trúc Chương Trình 4](#_Toc186136345)

[3. Ngôn Ngữ Thực Hiện 5](#_Toc186136346)

[3.1. Python là gì? 5](#_Toc186136347)

[3.2. Đặc điểm của ngôn ngữ lập trình Python 5](#_Toc186136348)

[4. YOLO và cơ chế hoạt động 6](#_Toc186136349)

[3.1. YOLO 6](#_Toc186136350)

[3.2. YOLOv8 9](#_Toc186136351)

[CHƯƠNG III: XÂY DỰNG CHƯƠNG TRÌNH 15](#_Toc186136352)

[1. Thư viện hỗ trợ 15](#_Toc186136353)

[2. Quá trình xây dựng chương trình 15](#_Toc186136354)

[3. Kết Quả Chương Trình 24](#_Toc186136355)

[4. Kết Luận Chương Trình 26](#_Toc186136356)

[5. Bổ Sung 2 Mô Hình Huấn Luyện 26](#_Toc186136357)

[KẾT LUẬN 31](#_Toc186136358)

[TÀI LIỆU THAM KHẢO 32](#_Toc186136359)

# DANH MỤC HÌNH ẢNH

[Hình 1 : Cách thức hoạt động của TGMT 2](#_Toc186136360)

[Hình 2 : Ứng dụng vào Y tế 3](#_Toc186136361)

[Hình 3 : ứng dụng vào xe tự hành 3](#_Toc186136362)

[Hình 4 : ứng dụng vào nông nghiệp 3](#_Toc186136363)

[Hình 5 : Kiến Trúc Chương Trình 4](#_Toc186136364)

[Hình 6 : Logo của Ultralytics 6](#_Toc186136365)

[Hình 7 : Cấu Trúc của YOLO 7](#_Toc186136366)

[Hình 8 : Cách YOLO hoạt động 7](#_Toc186136367)

[Hình 9 : các Phiên Bản YOLO 9](#_Toc186136368)

[Hình 10 : Logo YOLOv8 10](#_Toc186136369)

[Hình 11 : Cấu trúc của YOLOv8 11](#_Toc186136370)

[Hình 12 : Các Loại Phiên Bản YOLOv8 13](#_Toc186136371)

[Hình 13 : Số lượng ảnh gốc 16](#_Toc186136372)

[Hình 14 : Bộ dữ liệu 17](#_Toc186136373)

[Hình 15 : chỉnh sủa và dán nhãn 17](#_Toc186136374)

[Hình 16 : Đây là bộ Dữ Liệu 17](#_Toc186136375)

[Hình 17 : F1- confident Curve 21](#_Toc186136376)

[Hình 18 : Precision - Recall Curve 21](#_Toc186136377)

[Hình 19 : Confusion Matrix 21](#_Toc186136378)

[Hình 20 : Train\_batch 21](#_Toc186136379)

[Hình 21 : val\_batch\_label 21](#_Toc186136380)

[Hình 22 : valbatch\_pred 21](#_Toc186136381)

[Hình 23 : result 21](#_Toc186136382)

[Hình 24 : Giao diện chính của chương trình 24](#_Toc186136383)

[Hình 25 : Kết quả hiển thị nhận dạng bằng hình ảnh 25](#_Toc186136384)

[Hình 26 : Kết quả hiển thị nhận dạng bằng video 25](#_Toc186136385)

[Hình 27 : Kết quả hiển thị nhận dạng bằng camera 26](#_Toc186136386)

[Hình 28 : Dữ liệu Huấn luyện mô hình YOLOv8n 27](#_Toc186136387)

[Hình 29 : Dữ Liệu huấn luyện về mô hình YOLOv8s 28](#_Toc186136388)

# LỜI MỞ ĐẦU

Trong tình hình dịch Covid-19 và các mối đe dọa từ những bệnh về đường hô hấp, việc tuân thủ các biện pháp phòng ngừa, đặc biệt là đeo khẩu trang, đã trở thành yêu cầu thiết yếu trong bảo vệ sức khỏe cộng đồng. Công nghệ, đặc biệt là trí tuệ nhân tạo (AI) và học sâu (Deep Learning), đóng vai trò quan trọng trong việc hỗ trợ các giải pháp kiểm soát an toàn và quản lý sức khỏe cộng đồng.

Để triển khai ứng dụng học sâu và xử lý ảnh, đề tài " Xây dựng chương trình phát hiện người đeo khẩu trang " sẽ xây dựng một công cụ hỗ trợ phát hiện nhanh chóng và chính xác trạng thái đeo khẩu trang. Hệ thống được xây dựng trên nền tảng của mô hình tiên tiến YOLOv8 kết hợp với các công cụ mạnh mẽ như Python, OpenCV, Tkinter, và Pillow, cho phép nhận dạng các trạng thái: đeo khẩu trang đúng cách, không đeo khẩu trang, hoặc đeo khẩu trang sai cách.Nhận dạng được từ 3 chế độ : nhận dạng hình ảnh , nhận dạng video , nhận dạng camera .

Qua bài báo cáo trình bày các bước nghiên cứu, thiết kế và phát triển hệ thống, các khó khăn mà nhóm phải vượt qua và các giải pháp khắc phục. Đồng thời, em cũng phân tích những hạn chế và định hướng phát triển trong tương lai nhằm hoàn thiện hệ thống và mở rộng ứng dụng vào thực tế để nâng cao hiệu quả kiểm soát an toàn sức khỏe cộng đồng.

# CHƯƠNG I: NỘI DUNG LÝ THUYẾT

## Tổng quan về thị giác máy tính

### Giới thiệu về thị giác máy tính

Thị giác máy tính (Computer Vision) là một lĩnh vực nghiên cứu trong trí tuệ nhân tạo (AI) và khoa học máy tính, tập trung vào việc phát triển các phương pháp và thuật toán để máy tính có khả năng nhận biết, hiểu và xử lý thông tin từ dữ liệu hình ảnh hoặc video giống như cách mà con người nhìn và hiểu thế giới xung quanh.

Mục tiêu chính của thị giác máy tính là giúp máy tính có khả năng "nhìn" và "hiểu" hình ảnh một cách tự động, thậm chí có thể thực hiện các nhiệm vụ phức tạp như nhận diện đối tượng, phân loại hình ảnh, nhận diện khuôn mặt, đọc chữ viết, và nhiều ứng dụng khác.

### Cách thức hoạt động của Thị Giác Máy Tính

Nó phân tích hình ảnh và sau đó tạo ra các biểu diễn bằng số về những gì nó ‘nhìn thấy’ thông qua sử dụng mạng nơ-ron tích chập (CNN). CNN là một tầng của mạng thần kinh nhân tạo, sử dụng các lớp tích chập để lọc đầu vào cho ra thông tin hữu ích. Hoạt động tích chập bao gồm việc kết hợp dữ liệu đầu vào (feature map) với một kernel tích chập (filter) để tạo thành feature map được chuyển đổi.

Các bộ lọc trong lớp các tích chập (conv layers) được sửa đổi dựa trên các tham số đã học để trích xuất thông tin hữu ích nhất cho một tác vụ cụ thể. Mạng tích chập tự động điều chỉnh để tìm ra tính năng tốt nhất dựa trên nhiệm vụ. CNN sẽ lọc thông tin về hình dạng của một vật thể khi thực hiện nhiệm vụ nhận dạng đối tượng chung nhưng sẽ trích xuất màu sắc của con chim (bird) khi thực hiện nhiệm vụ nhận dạng chim. Điều này dựa trên sự hiểu biết của CNN rằng các loại vật thể khác nhau có hình dạng khác nhau, nhưng các loại chim khác nhau có nhiều khả năng khác nhau về màu sắc hơn là hình dạng.

|  |
| --- |
| Hình : Cách thức hoạt động của TGMT |

### Ứng dụng thực tiễn của thị giác máy tính

**Bảo mật và an toàn**

Chính phủ và doanh nghiệp sử dụng thị giác máy tính để tăng cường bảo mật tài sản , cơ sở và trang thiết bị . Ví dụ như về camera và cảm biến giám sát các không gian cộng đồng , cơ sở doanh nghiệp , cũng như các cơ quan cần có môi trưởng bảo mật cao . Các thiết bị này sẽ gửi cảnh báo tự động nếu có điều gì bất thường xảy ra , chẳng hạn như một người vào khu vực cấm.

**Hiệu quả hoạt động**

Thị giác máy tính có thể phân tích hình ảnh và trích xuất trong dữ liệu cho nghiệp vụ thông minh , tạo ra một hiệu quả hoạt động và hỗ trợ cho Trí Tuệ Nhân Tạo (AI) .

**Chăm sóc sức khỏe**

Thị giác máy tính đóng vai trò quan trọng trong việc chăm sóc sức khỏe cụ thể trong lĩnh vực y tế khám chữa bệnh cho bệnh nhân hỗ trợ cho bác sĩ . Đáng chú ý hơn là có thể phân tích hình ảnh y khoa tạo hiển thị trực quan cho các mô và cơ quan của con người , giúp cho việc chuẩn đoán nhanh và chính xác từ đó có thể cải thiện tình trạng chữa bệnh và cải thiện sức khỏe .

**Phương tiện tự hành**

Thị giác máy tính trong giao thông có thể nói giúp ích cho con người rất nhiều từ việc giám sát người tham gia giao thông khi điều khiển phương tiện cũng như tình hình giao thông xung quanh . Ở các phương tiện bán tự hành sử dụng máy học để giám sát hành vi con người .Ví dụ như nó sẽ tìm kiếm các dấu hiệu phân tâm , mệt mỏi . buồn ngủ , dấu hiệu đột quỵ dựa trên vị trí đầu , theo dõi mắt và chuyển động của phần trên cơ thể .

**Nông nghiệp**

Ứng dụng thị giác máy tính tăng cường hoạt động tổng thể của ngành nông nghiệp, từ tăng năng suất đến giảm chi phí với công nghệ tự động hóa thông minh. Hình ảnh vệ tinh cũng như cảnh quay UAV giúp phân tích các vùng đất rộng lớn và cải thiện các phương pháp canh tác

|  |  |  |
| --- | --- | --- |
| Hình 2 : Ứng dụng vào Y tế | Hình 3 : ứng dụng vào xe tự hành | Hình 4 : ứng dụng vào nông nghiệp |

# CHƯƠNG II: PHÂN TÍCH BÀI TOÁN

## Tổng Quan về Bài Toán

Đề tài : Xây dựng chương trình phát hiện người đeo khẩu trang .

Mục tiêu bài toán : Xây dựng một chương trình để nhận dạng và phát hiện người có hoặc không đeo khẩu tang , đeo khẩu trang sai cách từ ảnh , video hoặc camera . Hệ thống có thể giúp cho người quản lí phát hiện , cảnh báo và báo cáo tình trạng tuân thủ quy định đeo khẩu trang tại các khu vực cần đeo .

Giải pháp thực hiện và Quy Trình Thực hiện:

Thu nhập và chuẩn bị dữ liệu : Thu nhập hình ảnh và video bao gồm những người đeo khẩu trang đúng và sai hoặc không đeo . Sau đó gán nhãn dữ liệu thành các lớp như : “ đeo khẩu trang” , “đeo khẩu trang sai cách ” , không đeo khẩu trang . Sử dụng robotflow để tổng hợp dữ liệu và gán nhãn .

Huấn Luyện Mô Hình : Sử dụng mô hình YOLO để huẩn luyện với dữ liệu đã được gán nhãn thông qua kaggle để Huấn Luyện .

Triển khai mô hình chương trình nhận dạng người đeo khẩu trang và đưa ra các thông báo .

Các công cụ sử dụng là : Mô hình mạng YOLO , ngôn ngữ python , các thư viện hỗ trợ của python .

## Kiến Trúc Chương Trình



Hình : Kiến Trúc Chương Trình

Theo Hình Ảnh Minh Họa ở trên , Chương trình có 3 chế độ : Nhận dạng bằng hình ảnh , nhận dạng bằng video , nhận dạng bằng camera .

Khi chọn 1 trong 3 chế độ và chọn dữ liệu đầu vào tương ứng : Chương Trình sẽ nhận dạng từ mô hình đã huấn luyện ra 3 trạng thái đeo khẩu trang : Đeo Khẩu Trang , Không Đeo Khẩu Trang , Đeo Khẩu Trang Sai Cách .

Kết Quả nhận dạng sẽ được hiển thị lên màn hình .

## Ngôn Ngữ Thực Hiện

Python là một ngôn ngữ lập trình dễ học và đọc, điều này giúp cho việc phát triển và duy trì mã nguồn một cách dễ dàng. Đặc biệt có nhiều thư viện và framework trong Python được thiết kế đặc biệt cho xử lý hình ảnh và máy học, chẳng hạn như OpenCV (Open Source Computer Vision Library) và TensorFlow. Những thư viện này cung cấp các công cụ mạnh mẽ để thực hiện phân tích hình ảnh và máy học. Vì vậy em lựa chọn ngôn ngữ python là ngôn ngữ lập trình cho bài toán “Xây dựng hệ thống đếm lưu lượng ô tô dựa trên phát hiện đối tượng”.

### 3.1. Python là gì?

Python là một ngôn ngữ lập trình thông dịch (interpreted), hướng đối tượng (object-oriented), và là một ngôn ngữ bậc cao (high-level) ngữ nghĩa động (dynamic semantics). Python hỗ trợ các module và gói (packages), khuyến khích chương trình module hóa và tái sử dụng mã. Trình thông dịch Python và thư viện chuẩn mở rộng có sẵn dưới dạng mã nguồn hoặc dạng nhị phân miễn phí cho tất cả các nền tảng chính và có thể được phân phối tự do.

### 3.2. Đặc điểm của ngôn ngữ lập trình Python

Ngôn ngữ lập trình mức cao: Python là một ngôn ngữ lập trình mức cao, điều này có nghĩa là cú pháp của nó rất dễ đọc và hiểu, giúp người lập trình tập trung vào logic của chương trình thay vì chi tiết cú pháp phức tạp.

Thông dịch và biên dịch: Python được thực thi thông dịch (interpreter) trực tiếp từ mã nguồn, giúp đơn giản hóa quá trình phát triển và kiểm thử.

Đa năng: Python được sử dụng rộng rãi trong nhiều lĩnh vực, bao gồm phát triển web, trí tuệ nhân tạo, phân tích dữ liệu, tự động hóa, scripting hệ thống, và nhiều ứng dụng khác.

Thư viện và framework phong phú: Python có một cộng đồng lớn và đa dạng, với nhiều thư viện và framework hỗ trợ cho nhiều mục đích khác nhau.

Hỗ trợ lập trình hàm và lập trình đối tượng: Python hỗ trợ cả lập trình hàm và lập trình đối tượng. Người lập trình có thể sử dụng cả hai phong cách lập trình tùy thuộc vào yêu cầu của dự án.

Cộng đồng mạnh mẽ: Python có một cộng đồng lớn, năng động và tích cực. Điều này giúp cung cấp nguồn lực học, hỗ trợ và các dự án mã nguồn mở.

Dễ học và sử dụng: Python thường được chọn làm ngôn ngữ đầu tiên cho người mới học lập trình do cú pháp đơn giản và dễ học.

## YOLO và cơ chế hoạt động

### YOLO

|  |
| --- |
| Hình : Logo của Ultralytics |

Yolo (You only look once) là một mô hình mạng CNN cho việc phát hiện, nhận dạng, phân loại đối tượng. Yolo được tạo ra từ việc kết hợp giữa các convolutional layers và connected layers. Trong đó các convolutional layers sẽ trích xuất ra các feature của ảnh, còn full-connected layers sẽ dự đoán ra xác suất đó và tọa độ của đối tượng.

|  |
| --- |
| Hình : Cấu Trúc của YOLO |

***Cách yolo hoạt động***

Đầu vào của mô hình là một ảnh, mô hình sẽ nhận dạng ảnh đó có đối tượng nào hay không, sau đó sẽ xác định tọa độ của đối tượng trong bức ảnh. ẢNh đầu vào được chia thành thành *S*×*S* ô thường thì sẽ là 3×3, 7×7, 9×9... việc chia ô này có ảnh hưởng tới việc mô hình phát hiện đối tượng.

|  |
| --- |
| Hình : Cách YOLO hoạt động |

Với Input là 1 ảnh, đầu ra mô hình là một ma trận 3 chiều có kích thước *S*×*S*×(5×*N*+*M*) với số lượng tham số mỗi ô là (5×*N*+*M*) với N và M lần lượt là số lượng Box và Class mà mỗi ô cần dự đoán. Ví dụ với hình ảnh trên chia thành 7×7 ô, mỗi ô cần dự đóan 2 bounding box và 3 object : con chó, ô tô, xe đạp thì output là 7×7×13, mỗi ô sẽ có 13 tham số, kết quả trả về (7×7×2=98) bounding box. Chúng ta sẽ cùng giải thích con số (5×*N*+*M*) được tính như thế nào.  
Dự đoán mỗi bounding box gồm 5 thành phần : (x, y, w, h, prediction) với (x, y ) là tọa độ tâm của bounding box, (w, h) lần lượt là chiều rộng và chiều cao của bounding box, prediction được định nghĩa Pr(*Object*)∗ *IOU*(*pred*,*truth*) xin trình bày sau. Với hình ảnh trên như ta tính mỗi ô sẽ có 13 tham số, ta có thể hiểu đơn giản như sau tham số thứ 1 sẽ chỉ ra ô đó có chứa đối tượng nào hay không P(Object), tham số 2, 3, 4, 5 sẽ trả về x, y ,w, h của Box1. Tham số 6, 7, 8, 9, 10 tương tự sẽ Box2, tham số 11, 12, 13 lần lượt là xác suất ô đó có chứa object1( P(chó|object), object2(P(ô tô|object)), object3(P( xe đạp|object)). Lưu ý rằng tâm của bounding box nằm ở ô nào thì ô đó sẽ chứa đối tượng, cho dù đối tượng có thể ở các ô khác thì cũng sẽ trả về là 0. Vì vậy việc mà 1 ô chứa 2 hay nhiều tâm của bouding box hay đối tượng thì sẽ không thể detect được, đó là một hạn chế của mô hình YOLO1.

Sau đó Yolo nhanh chóng được phát triển trong những năm tiếp theo:

* YOLOv2 được phát hành vào năm 2016, đã cải thiện mô hình ban đầu bằng cách kết hợp chuẩn hóa hàng loạt, hộp neo và cụm thứ nguyên.
* YOLOv3 ra mắt vào năm 2018, đã nâng cao hơn nữa hiệu suất của mô hình bằng cách sử dụng mạng đường trục hiệu quả hơn, nhiều neo và tổng hợp kim tự tháp không gian.
* YOLOv4 được phát hành vào năm 2020, giới thiệu những cải tiến như tăng cường dữ liệu khảm, đầu phát hiện không neo mới và chức năng mất mát mới.
* YOLOv5 đã cải thiện hơn nữa hiệu suất của mô hình và thêm các tính năng mới như tối ưu hóa siêu tham số, theo dõi thử nghiệm tích hợp và xuất tự động sang các định dạng xuất phổ biến.
* YOLOv6 là mã nguồn mở nhờ Meituan vào năm 2022 và đang được sử dụng trong nhiều robot giao hàng tự động của công ty.
* YOLOv7 đã thêm các tác vụ bổ sung như ước tính tư thế trên bộ dữ liệu chính COCO.
* YOLOv8 là một mô hình tiên tiến, hiện đại (SOTA), YOLOv8 được xây dựng dựa trên sự thành công của các phiên bản trước, giới thiệu các tính năng và cải tiến mới để nâng cao hiệu suất, tính linh hoạt và hiệu quả.
* YOLOv9 : giới thiệu một kiến ​​trúc mô hình thị giác máy tính mới: YOLOv9. Mã nguồn đã được công bố, cho phép bất kỳ ai cũng có thể đào tạo các mô hình YOLOv9 của riêng mình.
* YOLOv10 : Được phát hành vào ngày 23 tháng 5 năm 2024, YOLOv10 là mô hình phát hiện vật thể theo thời gian thực do các nhà nghiên cứu từ Đại học Thanh Hoa phát triển.

|  |
| --- |
| Hình : các Phiên Bản YOLO |

### YOLOv8

Giới Thiệu

YOLOv8 là một mô hình nhận dạng đối tượng dựa trên mạng convolutional neural network (CNN) được phát triển bởi Joseph Redmon và nhóm nghiên cứu của ông tại Đại học Washington .

|  |
| --- |
| Hình : Logo YOLOv8 |

YOLOv8 là một thuật toán thị giác máy tính được sử dụng để phát hiện đối tượng. YOLOv8 là phiên bản nâng cấp của YOLOv7, với khả năng nhận diện đối tượng nhanh hơn và chính xác hơn. Điều này được đạt được thông qua một số cải tiến, bao gồm mạng kim tự tháp đặc trưng, các mô-đun chú ý không gian và các kỹ thuật tăng cường dữ liệu tiên tiến.

Mô hình YOLOv8 sử dụng một mạng neural kiến trúc darknet-53 để trích xuất đặc trưng của hình ảnh và áp dụng thuật toán nhận dạng đối tượng YOLOv8 trên các đặc trưng đó.

YOLOv8 là một phiên bản nâng cáp với phiên bản trước, với nhiều tính năng mới như hỗ trợ nhiều độ phân giải khác nhau, sử dụng các mô hình backbone khác nhau, và áp dụng các kỹ thuật như attention mechanism, label smoothing, mixup augmentation.

Mô hình backbone là một phần của một mô hình học sâu, thường là một mạng nơ-ron tích chập (CNN), có chức năng trích xuất các đặc trưng (features) từ ảnh.

|  |
| --- |
| Hình : Cấu trúc của YOLOv8 |

Attention mechanism là một kỹ thuật giúp mô hình tập trung vào những phần quan trọng của dữ liệu đầu vào, bằng cách gán các trọng số khác nhau cho các phần khác nhau của dữ liệu.

Label smoothing là một kỹ thuật giúp giảm hiện tượng overfitting (là hiện tượng khi mô hình học máy quá khớp với dữ liệu huấn luyện, tức là mô hình có độ chính xác cao trên dữ liệu huấn luyện nhưng thấp trên dữ liệu mới) của mô hình, bằng cách thay đổi các nhãn đích (target labels) từ các giá trị nhị phân (0 hoặc 1) thành các giá trị liên tục trong khoảng [0, 1]. Label smoothing giúp mô hình trở nên ít chắc chắn về các dự đoán của nó và tránh bị ảnh hưởng quá nhiều bởi các nhãn sai hoặc nhiễu.

Mixup augmentation là một kỹ thuật tăng cường dữ liệu (data augmentation) bằng cách tạo ra các điểm dữ liệu mới từ sự kết hợp tuyến tính của hai điểm dữ liệu ngẫu nhiên và nhãn tương ứng của chúng. Mixup augmentation giúp mô hình học được các biểu diễn phức tạp và linh hoạt hơn, và giảm thiểu hiện tượng overfitting.

YOLOv8 hỗ trợ đầy đủ các tác vụ AI về thị giác, bao gồm phát hiện, phân đoạn, ước tính tư thế, theo dõi và phân loại. Tính linh hoạt này cho phép người dùng tận dụng các khả năng của YOLOv8 trên nhiều ứng dụng và miền khác nhau.

*Kiến trúc YOLOv8 có thể được chia thành ba thành phần chính:*

Backbone: Đây là mạng nơ-ron tích chập (CNN) chịu trách nhiệm trích xuất các đặc điểm từ hình ảnh đầu vào. YOLOv8 sử dụng xương sống CSPDarknet53 tùy chỉnh, sử dụng các kết nối một phần giữa các giai đoạn để cải thiện luồng thông tin giữa các lớp và tăng độ chính xác.

Neck: Neck, còn được gọi là feature extractor, hợp nhất các bản đồ feature từ các giai đoạn khác nhau của backbone để nắm bắt thông tin ở nhiều quy mô khác nhau. YOLOv8 Architecture sử dụng một module C2f mới thay vì Feature Pyramid Network (FPN) truyền thống. Module này kết hợp các feature ngữ nghĩa cấp cao với thông tin không gian cấp thấp, dẫn đến cải thiện độ chính xác phát hiện, đặc biệt là đối với các đối tượng nhỏ.

Head: Head chịu trách nhiệm đưa ra dự đoán. YOLOv8 sử dụng nhiều mô-đun phát hiện để dự đoán các hộp giới hạn, điểm đối tượng và xác suất lớp cho mỗi ô lưới trong bản đồ đặc điểm. Sau đó, các dự đoán này được tổng hợp để có được các phát hiện cuối cùng.

*Ưu , nhược điểm của YOLOv8*

|  |  |
| --- | --- |
| Ưu điểm | Nhược điểm |
| Tốc độ: YOLOv8 được đánh giá là nhanh chóng và thời gian phản hồi thấp, giúp xử lý các tác vụ nhận diện đối tượng và phân-segment ảnh trong thời gian thực.  Độ chính xác: YOLOv8 được xây dựng trên các tiến bộ về học sâu và thị giác máy tính, đảm bảo độ chính xác cao trong việc nhận diện đối tượng.  Sự linh hoạt: YOLOv8 hỗ trợ việc nhận diện đối tượng và phân-segment trên cả GPU và CPU, tận dụng các công nghệ như TensorRT của Nvidia và OpenVino của Intel. | Để sử dụng YOLOv8 hiệu quả cần phải:  Có kiến thức chuyên sâu về Machine Learning, Deep Learning và các thuật toán liên quan.  Cần phải được huấn luyện trên một tập dữ liệu đủ lớn và đa dạng để đạt được hiệu quả cao nhất.  Yêu cầu các tài nguyên tính toán cao để đạt được tốc độ xử lý nhanh và chính xác.  Thuật toán YOLOv8 không phải là mã nguồn mở và chỉ có sẵn thông qua các thỏa thuận cấp phép với người tạo ra nó, Joseph Redmon.  YOLOv8 có thể không hoạt động tốt trong tất cả các môi trường và có thể cần thêm điều chỉnh hoặc tối ưu hóa để đạt được hiệu suất tối ưu. |

|  |
| --- |
| Hình : Các Loại Phiên Bản YOLOv8 |

YOLOv8n (Nano): Nhẹ, nhanh, tiết kiệm tài nguyên, phù hợp cho các thiết bị hạn chế tài nguyên.

YOLOv8s (Small): Cân bằng giữa tốc độ và độ chính xác, dùng trong ứng dụng thực tế trung bình.

YOLOv8m (Medium): Hiệu năng cao hơn, phù hợp với các bài toán phức tạp hơn.

YOLOv8l (Large): Tăng cường độ chính xác, dùng cho các hệ thống mạnh như máy chủ.

YOLOv8x(Extra Large): Hiệu năng cao nhất, phù hợp với bài toán yêu cầu độ chính xác tối đa, nhưng cần tài nguyên phần cứng lớn.

*Ứng dụng YOLOv8*

YOLOv8 là một mô hình đa năng mà bạn có thể sử dụng trong nhiều ứng dụng thực tế. Dưới đây là một số trường hợp sử dụng phổ biến.

Đếm người: Các nhà bán lẻ có thể đào tạo mô hình để phát hiện lưu lượng người đi bộ theo thời gian thực trong cửa hàng của họ, phát hiện độ dài hàng đợi, v.v.

Phân tích thể thao: Các nhà phân tích có thể sử dụng mô hình để theo dõi chuyển động của người chơi trên sân thể thao nhằm thu thập thông tin chi tiết có liên quan về động lực của đội (Xem AI trong thể thao ).

Quản lý hàng tồn kho: Mô hình phát hiện đối tượng có thể giúp phát hiện mức tồn kho sản phẩm để đảm bảo đủ lượng hàng tồn kho và cung cấp thông tin liên quan đến hành vi của người tiêu dùng.

Xe tự hành: Lái xe tự hành sử dụng mô hình phát hiện vật thể để giúp xe tự lái di chuyển an toàn trên đường.

# CHƯƠNG III: XÂY DỰNG CHƯƠNG TRÌNH

## Thư viện hỗ trợ

***Các thư viện hỗ trợ***

* Tkinter là một thư viện giao diện người dùng đồ họa (GUI) cho Python, cho phép tạo ra các cửa sổ, nút, menu và các thành phần khác. Tkinter có thể dùng để tạo giao diện người dùng cho chương trình, cho phép người dùng chọn ảnh đầu vào, điều khiển các chức năng và xem kết quả xử lý ảnh.
* Pillow là một thư viện xử lý ảnh cho Python, cung cấp các hàm và phương thức để tạo, đọc, ghi, hiển thị và biến đổi ảnh. Pillow là một bản fork của PIL với nhiều cải tiến và bảo trì thường xuyên. PIL hoặc Pillow có thể dùng để chuyển đổi định dạng ảnh, điều chỉnh kích thước, độ sáng, độ tương phản và các thuộc tính khác của ảnh. Cũng có thể dùng để vẽ lên ảnh để hiển thị kết quả mô hình.
* Datetime : Thư viện được tích hợp sẵn trong python . Dùng để hiển thị thời gian hiển thị trong giao diện .
* Ultralytics: cung cấp các công cụ để huấn luyện, kiểm tra, dự đoán và triển khai mô hình YOLOv8.
* OpenCV : OpenCV là tên viết tắt của open source computer vision library – có thể được hiểu là một thư viện nguồn mở cho máy tính. Cụ thể hơn OpenCV là kho lưu trữ các mã nguồn mở được dùng để xử lý hình ảnh, phát triển các ứng dụng đồ họa trong thời gian thực

## Quá trình xây dựng chương trình

**B1 : Sưu tầm và chuẩn bị dữ liệu**

Sử dụng Roboflow để tạo bộ dứ liệu bao gồm thu nhập ảnh thủ công bằng cách chụp , hoặc lưu về từ nhiều chỗ và sử dụng trực tiếp 1 số bộ dữ liệu từ Roboflow. Đây là nguồn dữ liệu trên Roboflow mà em đã làm :

https://universe.roboflow.com/object-mfpha/facemask-detection-nyuzn/dataset/4

Dán nhãn các ảnh đã lưu về máy và chỉnh sửa lại các nhãn đã dán sai khi sử dụng 1 số bộ dữ liệu từ Roboflow và xuất ra file data.yaml sau :

|  |
| --- |
| train: ../train/images  val: ../valid/images  test: ../test/images  nc: 5  names: ['deo khau trang', 'deo khau trang sai cach', 'khau trang', 'ko deo khau trang', 'undefined'] |

Đây là bộ dữ liệu b

|  |  |
| --- | --- |
| Số Dữ Liệu ảnh gốc trước khi dùng các phép để tăng cường dữ liệu để ra được Như Hình 13 | Hình : Số lượng ảnh gốc |

|  |
| --- |
| Hình : Bộ dữ liệu |
| Hình : chỉnh sủa và dán nhãn   |  | | --- | | Hình : Đây là bộ Dữ Liệu | |

**B2 : Huấn luyện mô hình với bộ dữ liệu trên**

Cài đặt thư viện Ultralytics

|  |
| --- |
| !pip install ultralytics  from IPython import display  display.clear\_output()  import ultralytics  ultralytics.checks() |
|  |

Tải dữ liệu từ Roboflow

|  |
| --- |
| !pip install roboflow  from roboflow import Roboflow  rf = Roboflow(api\_key="DSliZq2pHPerHad2ElT2")  project = rf.workspace("object-mfpha").project("facemask-detection-nyuzn")  version = project.version(4)  dataset = version.download("yolov8") |
|  |

Huấn luyện mô hình YOLOv8 cụ thẻ là mô hình medium

|  |
| --- |
| !yolo task=detect mode=train model=yolov8m.pt data=/kaggle/working/facemask-detection-4/data.yaml epochs=100 batch=32 imgsz=640 device=0 patience=10 |
|  |
|  |

Nén và lưu kết quả

|  |
| --- |
| !zip -r facemask.zip /kaggle/working/runs/detect |
|  |

**Kết quả đánh giá chương trình**

**chỉ số chính**

Precision (Độ chính xác) mô hình đạt 80.38% cho thấy khả năng xác định đúng với các đối tượng mục tiêu và lớp chính xác nhất là deo khau trang

Recall (Độ nhạy ) : Mô hình đạt 72.34%, thể hiện khả năng phát hiện đầy đủ các đối tượng mục tiêu.

F1-Score ( Trung bình giữa Precision và Recall) : đạt 75.68%, phản ánh sự cân bằng tương đối giữa độ chính xác và độ nhạy.

mAP@50 (Điểm trung bình độ chính xác ở ngưỡng IOU50%) :Mô hình đạt 78.93%, cho thấy hiệu suất phân loại và định vị đối tượng ở mức cơ bản là tốt.

mAP@50-95% (Điểm trung bình độ chính xác ở ngưỡng IOU50%-90% ) : Mô hình đạt 61.43%, phản ánh hiệu suất toàn diện khi yêu cầu độ chính xác cao hơn.

|  |  |  |
| --- | --- | --- |
| Hình 17 : F1- confident Curve | Hình 18 : Precision - Recall Curve | Hình : Confusion Matrix |
| Hình : Train\_batch | Hình : val\_batch\_label | Hình : valbatch\_pred |
| Hình : result | | |

**B3 : Xây dựng chương trình**

Một số đoạn mã tiêu biểu trong chương trình

Hàm nhận dạng từ ảnh

|  |
| --- |
| def result\_image():  if image\_path:  image = cv2.imread(image\_path)  results = model.predict(source=image, conf=confidence\_threshold)  labels = extract\_labels(results)  show\_image\_with\_results(image, results)  update\_status(f"Nhận dạng hoàn tất: {labels}") |

**Trích xuất nhãn từ kết quả nhận dạng**

|  |
| --- |
| def extract\_labels(results):  labels = []  for result in results:  boxes = result.boxes  for box in boxes:  cls = int(box.cls[0])  confidence = box.conf[0]  if confidence >= confidence\_threshold:  label = model.names[cls]  labels.append(label)  return ', '.join(labels) |

**Hàm vẽ ảnh**

|  |
| --- |
| def draw\_results(image, results):  for result in results:  boxes = result.boxes  for box in boxes:  confidence = box.conf[0]  if confidence >= confidence\_threshold:  x1, y1, x2, y2 = map(int, box.xyxy[0])  cls = int(box.cls[0])  label = model.names[cls]  cv2.rectangle(image, (x1, y1), (x2, y2), color, 2)  text = f"{label} {confidence:.2f}"  font\_scale = max(0.5, min((x2 - x1) / 300, 2))  font\_thickness = max(1, int(font\_scale))  (text\_width, text\_height), baseline = cv2.getTextSize(text, cv2.FONT\_HERSHEY\_SIMPLEX, font\_scale, font\_thickness)  text\_x = x1  text\_y = y1 - 10 if y1 - 10 > text\_height else y1 + text\_height + 10  cv2.rectangle(image, (text\_x, text\_y - text\_height - baseline),  (text\_x + text\_width, text\_y + baseline), color, cv2.FILLED)  cv2.putText(image, text, (text\_x, text\_y), cv2.FONT\_HERSHEY\_SIMPLEX, font\_scale, (255, 255, 255),  font\_thickness, cv2.LINE\_AA)  return image |

**Nhận dạng bằng cam**

|  |
| --- |
| def start\_camera():  global cap, running  cap = cv2.VideoCapture(0)  running = True  update\_status("Camera đã khởi động")  def update\_camera\_frame():  if running and cap.isOpened():  ret, frame = cap.read()  if ret:  if trigger\_recognition:  results = model.predict(source=frame, conf=confidence\_threshold)  frame = draw\_results(frame, results)  show\_image(frame, img\_label\_left)  img\_label\_left.after(10, update\_camera\_frame)  else:  stop\_camera()  else:  stop\_camera()  update\_camera\_frame() |

## Kết Quả Chương Trình

**Giao diện chính của chương trình**

|  |
| --- |
| Hình : Giao diện chính của chương trình |

**Kết quả hiển thị nhận dạng bằng hình ảnh**

|  |
| --- |
| Hình : Kết quả hiển thị nhận dạng bằng hình ảnh |

**Kết quả hiển thị nhận dạng bằng video**

|  |
| --- |
| Hình : Kết quả hiển thị nhận dạng bằng video |

**Kết quả hiển thị nhận dạng bằng camera**

|  |
| --- |
| Hình : Kết quả hiển thị nhận dạng bằng camera |

## Kết Luận Chương Trình

**Ưu điểm :** Chương trình có giao diện đơn giản , dễ thao tác . Sử dụng 3 chế độ nhận dạng bằng ảnh, video , camera . Nhận dạng 3 kiểu trạng thái tiêu biểu là Đeo Khẩu Trang , Không Đeo Khẩu Trang , Đeo Khẩu Trang sai cách . Nhận dạng độ chính xác đạt 87.32 ở ảnh . Nhận dạng độ chính xác ở video 89,78% dựa vào xác định từng khung hình một trong video .

**Nhược điểm :** Bộ dữ liệu huấn luyện hiện tại chưa đủ lớn và đa dạng, dẫn đến độ chính xác chưa tối ưu trong một số trường hợp phức tạp hoặc đặc thù. Ngoài ra, chương trình có thể gặp phải các lỗi nhận dạng khi ánh sáng hoặc chất lượng hình ảnh đầu vào không đảm bảo.Một số hạn chế khi nhận dạng một người thì chính xác cao còn nhiều người thì còn thấp .

## Bổ Sung 2 Mô Hình Huấn Luyện

* 1. YOLOv8n

|  |
| --- |
| !yolo task=detect mode=train model=yolov8n.pt data=/kaggle/working/facemask-detection-4/data.yaml epochs=100 batch=64 imgsz=640 device=0 patience=10 |
| Hình : Dữ liệu Huấn luyện mô hình YOLOv8n |

Thời gian hoàn thành Huấn Luyện Mô Hình của YOLOv8n đạt thời gian hoàn thành 1,324 tiếng nhanh hơn so với mô hình của YOLOv8m hoàn thành là 4.508 tiếng Độ chính xác của mô hình đạt độ chính xác 83,09 thấp hơn nhiều so với độ chính xác của YOLOv8 medium là 87,32%. Tóm tắt về mô hình là 225 layers,3,011,823 parameters, 3,011,807 gradients, 8.2 GFLOPs so với YOLOv8m là 295 layers, 25,859,215 parameters, 25,859,199 gradients, 79.1 GFLOPs .Số epoch 58 không thể tăng độ chính xác thì dừng lại so với YOLOv8m là 53. Điều đó thể hiện rằng do YOLOv8n là một mô hình nhẹ hơn , ít tham số hơn và yêu cầu tài nguyên tính toán thấp hơn . Còn được thiết kế nhỏ gọn , số lượng tham số và độ phức tạp kiến trúc thấp hơn dẫn đến khả năng học tập kém hơn . Tuy nhiên độ chính xac vẫn đủ cao đủ đáp ứng mà chương trình đưa ra .

Nhận xét : YOLOv8n là lựa chọn tốt cho các bài toán yêu cầu :triển khai nhanh chóng, thiết bị có tài nguyên hạn chế ,độ chính xác không yêu cầu mức tuyệt đối.

* 1. YOLOv8s

|  |
| --- |
| !yolo task=detect mode=train model=yolov8s.pt data=/kaggle/working/facemask-detection-4/data.yaml epochs=100 batch=64 imgsz=640 device=0 patience=10 |
| Hình : Dữ Liệu huấn luyện về mô hình YOLOv8s |

Thời gian hoàn thành Huấn Luyện Mô Hình của YOLOv8n đạt thời gian hoàn thành 1,411 tiếng nhanh hơn so với mô hình của YOLOv8m hoàn thành là 4.508 tiếng Độ chính xác của mô hình đạt độ chính xác 85,91 thấp hơn nhiều so với độ chính xác của YOLOv8 medium là 87,32%. Tóm tắt về mô hình là 225 layers,11,137,535 parameters, 11,137,519 gradients,28,7 GFLOPs so với YOLOv8m là 295 layers, 25,859,215 parameters, 25,859,199 gradients, 79.1 GFLOPs . Điều đó thể hiện rằng do YOLOv8n là một mô hình nhẹ hơn , ít tham số hơn và yêu cầu tài nguyên tính toán thấp hơn . Còn được thiết kế nhỏ gọn , số lượng tham số và độ phức tạp kiến trúc thấp hơn dẫn đến khả năng học tập kém hơn . Tuy nhiên độ chính xac vẫn đủ cao đủ đáp ứng mà chương trình đưa ra .

Nhận xét : YOLOv8s là lựa chọn tốt khi yêu cầu:độ chính xác cao hơn YOLOv8n nhưng không quá lớn như YOLOv8m,cân bằng giữa tốc độ và tài nguyên sử dụng.

5.3. Kết Luận

YOLOv8n: Tối ưu hóa tốc độ huấn luyện, phù hợp với các ứng dụng triển khai nhanh, tài nguyên hạn chế, và yêu cầu độ chính xác vừa phải.

YOLOv8s: Cung cấp sự cân bằng tốt giữa tốc độ và độ chính xác, phù hợp với các bài toán yêu cầu hiệu suất cao hơn nhưng không cần tài nguyên lớn như YOLOv8m.

YOLOv8m: Lựa chọn tối ưu khi cần độ chính xác cao và có sẵn tài nguyên phần cứng mạnh.

Đặc điểm tài nguyên:Mô hình nhỏ hơn (YOLOv8n và YOLOv8s) có ít tham số và tính toán hơn, phù hợp cho các hệ thống edge hoặc thiết bị di động.Mô hình lớn hơn (YOLOv8m) yêu cầu phần cứng mạnh nhưng mang lại hiệu suất cao hơn.

Ứng dụng thực tế:Với các bài toán như phát hiện khẩu trang, YOLOv8n hoặc YOLOv8s đều đủ đáp ứng trong môi trường thông thường.Trong các trường hợp yêu cầu độ chính xác tuyệt đối (như bệnh viện hoặc nơi có yêu cầu cao), YOLOv8m sẽ phù hợp hơn.

Tóm tắt so sánh

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Mô hình | Thời gian huấn luyện | Độ chính xác (%) | Tham số | GFLOPs | Ứng dụng phù hợp |
| YOLOv8n | 1.324 tiếng | 83.09 | 3,011,823 | 8.2 | Triển khai nhanh, thiết bị hạn chế |
| YOLOv8s | 1.411 tiếng | 85.91 | 11,137,535 | 28.7 | Cân bằng giữa độ chính xác và tốc độ |
| YOLOv8m | 4.508 tiếng | 87.32 | 25,859,215 | 79.1 | Yêu cầu độ chính xác cao, tài nguyên mạnh |

# KẾT LUẬN

Đề tài : “Xây dựng chương trình phát hiện người đeo khẩu trang” đã triển khai hệ thống nhận dạng một cách hiệu quả bằng mô hình YOLOv8, có ba chế độ hoạt động của hệ thống hỗ trợ: hình ảnh, video và camera thời gian thực; phân loại ba trạng thái chính: đeo khẩu trang đúng cách, không đeo khẩu trang, đeo khẩu trang sai cách.

Được xây dựng bằng ngôn ngữ Python và các thư viện mạnh mẽ như OpenCV, Tkinter, và Ultralytics, hệ thống đảm bảo độ chính xác và tốc độ xử lý tốt. Tuy nhiên, hạn chế của hệ thống là bộ dữ liệu huấn luyện chưa đủ lớn và gặp khó khăn trong điều kiện ánh sáng yếu hoặc hình ảnh chất lượng kém.

Tuy nhiên, trong quá trình thực hiện hệ thống còn tồn tại một số hạn chế như sau:Bộ dữ liệu tập train chưa đủ lớn và phong phú để hệ có thể nhận dạng một cách tốt nhất trong nhiều điều kiện phức tạp như điều kiện thiếu sáng, hình ảnh bị nhiễu, hay các góc quay không dễ lấy trích xuất.

Hướng phát triển trong tương lai là Nâng cấp dữ liệu huấn luyện: Tăng cường số lượng và chất lượng dữ liệu.Cải tiến mô hình: Tinh chỉnh thuật toán để tăng độ chính xác và giảm lỗi.Tích hợp thời gian thực: Hoàn thiện nhận dạng qua camera trực tiếp.Mở rộng ứng dụng: Phát hiện người trong đám đông hoặc theo dõi hành vi không tuân thủ quy định y tế.

Hệ thống hiện tại đã đạt được nền tảng vững chắc để tiếp tục phát triển, hứa hẹn trở thành công cụ hiệu quả trong việc hỗ trợ các nỗ lực bảo vệ sức khỏe cộng đồng.

# TÀI LIỆU THAM KHẢO

Website :

<https://aws.amazon.com/vi/what-is/computer-vision/>