BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN
NGÀNH CÔNG NGHỆ THÔNG TIN

BÁO CÁO ĐỒ ÁN CUỐI KỲ

ĐỒ ÁN CÔNG NGHỆ THÔNG TIN

ỨNG DỤNG AI NHẬN DIỆN HÀNG HÓA

Giảng viên hướng dẫn: Trần Nhật Quang

Danh sách sinh viên thực hiện

Mã số SV	Họ và tên	Mức độ đóng góp (%)
21110313	Nguyễn Dương Tiến Thông	100
21110308	Trần Xuân Thanh Thiện	100
21110747	Huỳnh Thiện Thọ	100

TP. Hồ Chí Minh, tháng 12 năm 2023

TIÊU CHÍ	NỘI DUNG	TRÌNH BÀY	TỔNG
ĐIỂM			

NHẬN XÉT

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Ký tên

MỤC LỤC

Danh mục các bảng

Danh mục các hình

LỜI MỞ ĐẦU

Sức mạnh của Trí tuệ Nhân tạo (AI) không chỉ là một khám phá khoa học mà còn là một sức mạnh biến đổi cách chúng ta tương tác với thế giới xung quanh. Trong hành trình không ngừng của sự tiến bộ, ứng dụng AI ngày càng trở nên hữu ích và đa dạng, mở ra những khả năng mới trong nhiều lĩnh vực của cuộc sống hàng ngày. Trong ngữ cảnh này, một trong những áp dụng độc đáo của công nghệ AI là khả năng nhận dạng hàng hóa, mang lại sự thuận tiện và tối ưu hóa trải nghiệm cho người tiêu dùng.

Việc sử dụng AI để nhận dạng hàng hóa không chỉ mang lại tiện ích mà còn là một bước nhảy vọt trong quá trình thanh toán. Khi người tiêu dùng có khả năng mua sắm một cách nhanh chóng và thuận tiện, sự chọn lựa hình thức mua trực tiếp bằng công nghệ AI trở nên hết sức hợp lý. Điều này không chỉ giảm bớt sự phức tạp và thời gian mà người tiêu dùng phải chi trả khi mua sắm, mà còn tạo ra một trải nghiệm thanh toán mượt mà và hiệu quả.

Bài báo cáo này sẽ mở rộng sự hiểu biết về cách công nghệ AI nhận dạng hàng hóa đang tích hợp vào hệ thống thanh toán, tăng cường tiện ích và tối ưu hóa trải nghiệm mua sắm. Chúng ta sẽ khám phá những phương pháp và thuật toán mà AI sử dụng để nhận diện sản phẩm một cách chính xác, từ quét mã vạch đến nhận dạng hình ảnh và sự kết hợp linh hoạt của chúng.

Qua sự tìm hiểu này, chúng ta sẽ thấy rõ hơn về cách công nghệ AI đã và đang thay đổi cách chúng ta thực hiện thanh toán hàng ngày. Bài báo cáo cũng sẽ giới thiệu về những lợi ích cụ thể mà ứng dụng AI trong việc nhận dạng hàng hóa mang lại, cũng như những thách thức và cơ hội mà nó đặt ra trong ngành thương mại điện tử và chuỗi cung ứng. Hãy cùng nhau đặt câu hỏi về tương lai của thanh toán và sự hình thành của một thế giới mua sắm thông minh, nơi mà AI không chỉ là công cụ hỗ trợ mà còn là người bạn đồng hành không thể thiếu.

Tuy nhiên, điều này cũng sẽ đưa chúng ta đối diện với những thách thức, cũng như những cơ hội mở rộng sự áp dụng của công nghệ này trong các lĩnh vực khác nhau của thương mại điện tử và chuỗi cung ứng

PHẦN I. ĐẶC TẢ

Đề tài của chúng tôi tập trung vào việc tích hợp Trí Tuệ Nhân Tạo (AI) vào quá trình nhận diện sản phẩm trong môi trường thương mại điện tử. Chúng tôi đặt ra mục tiêu xây dựng một hệ thống thông minh nhằm cải thiện trải nghiệm mua sắm trực tuyến của người tiêu dùng.

1. Sản phẩm dùng làm gì:

Sản phẩm có chức năng chính là ứng dụng Trí tuệ Nhân tạo (AI) để nhận dạng hàng hóa. Chủ yếu, phần mềm được thiết kế để tích hợp vào hệ thống thanh toán, bao gồm cả các ứng dụng di động và các trang web thương mại điện tử, trong đó AI sẽ tiến hành nhận diện các mặt hàng thông qua dữ liệu thu được từ camera và hiển thị lên giỏ hàng.

2. Dữ liệu, thông tin đầu vào:

Dữ liệu đầu vào sẽ là hình ảnh thu được từ camera, các thông tin của sản phẩm như tên sản phẩm, mô tả về sản phẩm, giá cả, hình ảnh của hàng hóa và danh mục hàng hóa.

3. Các tình huống sử dụng (mục đích, tính năng)

- Việc tích hợp AI nào nhận diện sản phẩm nhằm tối ưu hóa quá trình thanh toán trong môi trường mua sắm. Giúp cho việc thanh toán dễ dàng và thuận tiện hơn, giảm bớt được thời gian mua sắm.

- Tính năng:

  - Nhận diện sản phẩm: Phần mềm sẽ sử dụng công nghệ AI để nhận diện và định danh chính xác sản phẩm thông qua ảnh quét được từ camera
  - Tính toán tổng giá trị đơn hàng: Dựa trên thông tin về sản phẩm được nhận dạng, phần mềm sẽ tính toán tổng giá trị đơn hàng để tiến hành thanh toán.
  - Xác nhận thanh toán: Người dùng sẽ xác nhận và hoàn tất thanh toán sau khi xem xét thông tin đơn hàng.

4. Giao diện dự kiến:

- Giao diện sẽ được thiết kế đơn giản, dễ sử dụng và thân thiện với người dùng.

- Giao diện dự kiến của phần mềm sẽ bao gồm:

  - Khu vực hiển thị thông tin sản phẩm: Hiển thị thông tin chi tiết của sản phẩm được nhận diện, bao gồm tên sản phẩm, avatar của sản phẩm, số lượng, trọng lượng và tổng số tiền phải thanh toán của mỗi món hàng.

- Tổng giá trị đơn hàng: Hiển thị tổng giá trị của các sản phẩm trong đơn hàng.
- Nút thanh toán: Dùng để xác nhận và hoàn tất thanh toán.
- Trang VNPay: Dùng để thanh toán, khách hàng có thể lựa chọn giữa mã QR, thẻ thanh toán nội địa và thẻ quốc tế.

PHẦN II. PHÂN CÔNG CÔNG VIỆC

Bảng 1. Bảng phân công công việc của từng thành viên

TT	Tên sinh viên	Mô tả công việc	Phần trăm đóng góp
1	**Nguyễn Dương Tiến Thông**	- Triển khai thuật toán nhận diện AI. - Phát triển BoT-SORT và tích hợp Utralytics. - Sử dụng GPU của Google Colab để tối ưu hóa quá trình nhận diện. - Thiết kế giao diện giỏ hàng	**100%**
2	**Trần Xuân Thanh Thiện**	- Annotate và huấn luyện mô hình AI. - Nghiên cứu về YOLO V8, BoT-SORT, Utralytics để áp dụng vào dự án. - Kiểm tra và đảm bảo hiệu suất của thuật toán. - Sửa lỗi thuật toán - Thực hiện kiểm thử chất lượng.	**100%**
3	**Huỳnh Thiện Thọ**	- Annotate và huấn luyện mô hình AI. - Viết báo cáo. - Kiểm tra và sửa lỗi thuật toán. - So sánh hiệu suất giữa các thuật toán đã nghiên cứu. - Tích hợp VNPay vào hệ thống thanh toán. - Thực hiện kiểm thử và đánh giá tính ổn định.	**100%**

PHẦN III. THIẾT KẾ

1. Thiết kế giao diện

a. Giao diện giỏ hàng

Hình 1. Giao diện giỏ hàng

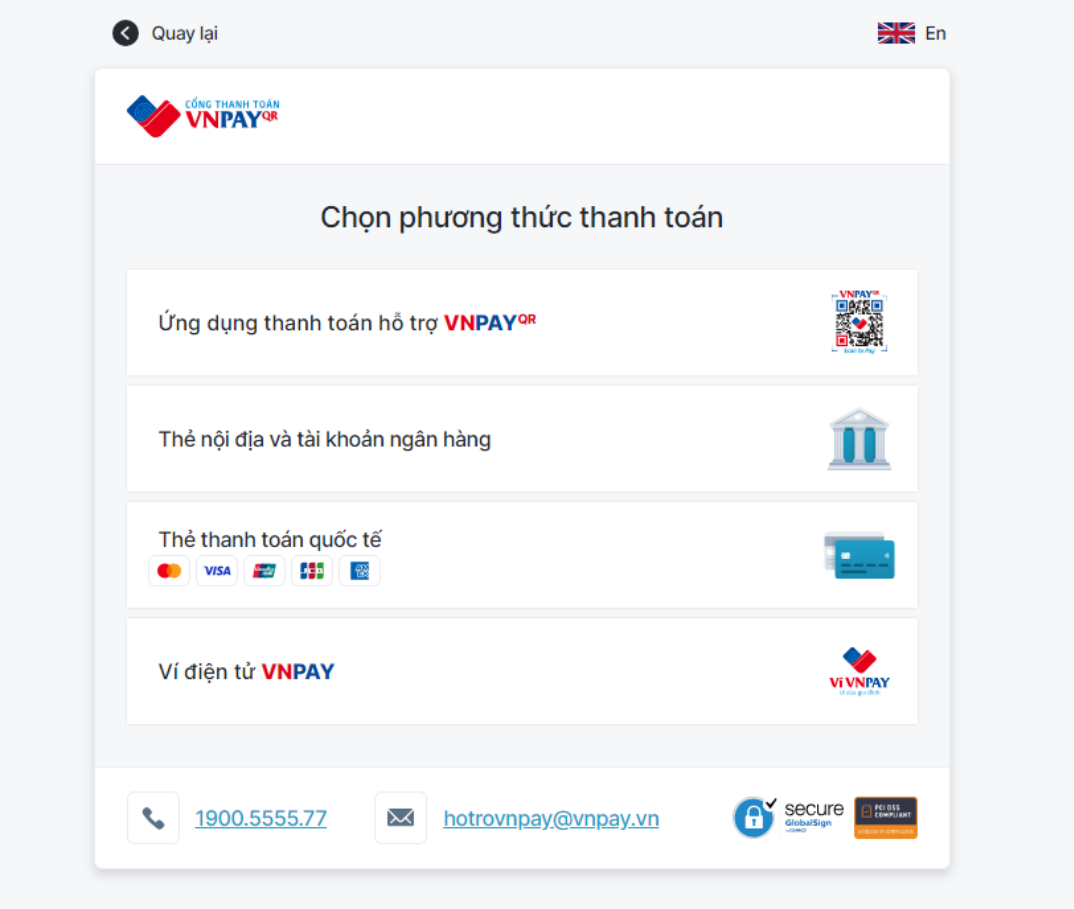
Mục đích: Giúp khách hàng có thể biết được thông tin những sản phẩm mình đã đặt lên bàn thanh toán để có thể tiến hành thanh toán.

Giải thích: Khi khách hàng đặt hàng hóa lên bàn thanh toán, hệ thống AI sẽ nhận diện các mặt hàng và hiện lên trang thanh toán này, các thông tin bao gồm Avatar sản phẩm, tên sản phẩm, số lượng, tổng số tiền của mỗi món hàng, nút thanh toán sẽ hiển thị tổng số tiền cần thanh toán và chuyển hướng sang phương thức thanh toán.

b. Giao diện Vnpay

Mục đích: Giúp khách hàng có thể thanh toán đơn hàng của mình qua nhiều phương thức thanh toán khác nhau.

Giải thích: Sau khi người dùng ấn nút "Thanh toán" ở giao diện giỏ hàng thì hệ thống sẽ chuyển hướng sang trang VNPay để người dùng lựa chọn phương thức thanh toán, có 4 phương thức thanh toán là thanh toán qua mã QR, thanh toán qua thẻ nội địa và tài khoản ngân hàng, thanh toán qua thẻ quốc tế và thanh toán qua ví điện tử VNPay. Khách hàng có thể tùy chọn phương thức thanh toán phù hợp với mình.



Hình 2. Giao diện VNPay

2. Lựa chọn thuật toán

Trong quá trình đối mặt với nhiệm vụ nhận diện đối tượng và theo dõi trong môi trường thực tế, quá trình chọn lựa thuật toán đóng vai trò quan trọng đối với hiệu suất và độ chính xác của hệ thống. Chúng tôi đã xác định ba thuật toán chủ chốt để xem xét, đó là YOLO (V8), BoT-SORT và Utralytics. Mỗi thuật toán đều có những đặc tính riêng biệt và ưu điểm, và quyết định chọn lựa nên phản ánh các yếu tố cụ thể của dự án và yêu cầu kỹ thuật.

i. YOLO (V8):

YOLO (You Only Look Once) là một thuật toán nhận diện đối tượng đơn giai đoạn (single-shot detection), tức là nó có thể dự đoán vị trí và nhãn của tất cả các đối tượng trong một ảnh chỉ trong một lần chạy. YOLO (V8) là phiên bản mới nhất của YOLO, được phát

hành vào năm 2023. Phiên bản này có những cải tiến đáng kể về độ chính xác và tốc độ, nhờ sử dụng kiến trúc mạng Neural Network mới và các kỹ thuật giảm thiểu sai số.

- Ưu điểm của YOLO (V8)

  - Độ chính xác cao, đặc biệt là đối với các đối tượng lớn và trung bình.
  - Tốc độ nhanh, có thể đạt tới 45 FPS với GPU mạnh.
  - Hiệu quả sử dụng bộ nhớ, phù hợp với các thiết bị có khả năng tính toán hạn chế.

- Nhược điểm của YOLO (V8)

  - Khả năng phát hiện đối tượng nhỏ kém, đặc biệt là đối với các đối tượng có kích thước tương tự như nhiễu trong ảnh.

Khó khăn trong việc phát hiện đối tượng có kích thước khác nhau trên cùng một ảnh là một thách thức quan trọng trong lĩnh vực nhận diện đối tượng. YOLO (V8) là một trong những thuật toán được chọn lựa để giải quyết vấn đề này và có thể được so sánh với các thuật toán nhận diện đối tượng đơn giai đoạn khác như SSD (Single Shot Multibox Detector) và Faster R-CNN (Region-based Convolutional Neural Network).

Cả ba thuật toán đều chia sẻ ưu điểm là tốc độ nhanh, làm cho chúng rất phù hợp với các ứng dụng đòi hỏi đáp ứng nhanh chóng, như trong thời gian thực hay xử lý video. Tuy nhiên, có những sự khác biệt quan trọng giữa chúng.

YOLO (V8) nổi bật với độ chính xác cao hơn so với SSD và Faster R-CNN, đặc biệt là đối với các đối tượng lớn và trung bình. Phương pháp "You Only Look Once" giúp YOLO (V8) thực hiện việc nhận diện trong một lần chạy, giảm thiểu sự mất mát thông tin và tăng cường khả năng chính xác.

Trái ngược với đó, SSD chủ yếu được biết đến với khả năng chính xác cao trên các đối tượng nhỏ, nhưng có thể gặp khó khăn hơn khi đối mặt với đối tượng lớn. Trong khi đó, Faster R-CNN, mặc dù cung cấp độ chính xác tốt, nhưng thường yêu cầu nhiều giai đoạn trong quá trình nhận diện, làm tăng độ phức tạp và giảm tốc độ.

Trong ngữ cảnh này, mức độ độ chính xác trong việc nhận diện sản phẩm và tốc độ xử lý là yếu tố quyết định. YOLO (V8) với ưu điểm về độ chính xác cao và khả năng nhận diện đa đối tượng trong một lần chạy là sự lựa chọn lý tưởng cho ứng dụng của chúng tôi. Đặc biệt, với khả năng xử lý đối tượng có kích thước đa dạng trên một ảnh, YOLO (V8) đáp ứng đầy

đủ nhu cầu của chúng tôi trong việc nhận diện sản phẩm trong môi trường siêu thị.

ii. BoT-SORT:

BoT-SORT (Bounding Box Tracker with Simple Online and Realtime Tracking) là một thuật toán theo dõi đối tượng đơn giản và hiệu quả. BoT-SORT sử dụng phương pháp gia tăng tối thiểu (least squares minimization) để ước tính vị trí và kích thước của đối tượng trong các khung hình tiếp theo.

- Ưu điểm của BoT-SORT

  - Khả năng theo dõi đối tượng hiệu quả và linh hoạt trong các môi trường đa dạng.
  - Sự đơn giản và khả năng giảm sai số của thuật toán làm cho nó trở thành một lựa chọn hữu ích trong lĩnh vực xử lý video và giám sát an ninh.

- Nhược điểm của BoT-SORT

  - Khả năng phát hiện đối tượng mới kém, đặc biệt là đối với các đối tượng xuất hiện đột ngột trong khung hình.
  - Khó khăn trong việc theo dõi đối tượng có kích thước nhỏ hoặc bị che khuất.

BoT-SORT là một thuật toán theo dõi đối tượng nổi bật và có thể được so sánh với các đồng đội trong lĩnh vực như DeepSORT và MHT-SORT. Tất cả ba thuật toán đều sử dụng phương pháp gia tăng tối thiểu để ước tính vị trí và kích thước của đối tượng, tạo ra những ưu điểm chung về khả năng theo dõi đối tượng trong thời gian thực.

Trong khi cả ba thuật toán đều có các đặc tính tương tự, BoT-SORT đứng lên với khả năng giảm sai số cao hơn, đặc biệt là nhờ vào việc sử dụng kỹ thuật bình phương tối thiểu có trọng số (weighted least squares). Phương pháp này giúp cải thiện độ chính xác của quá trình ước tính, làm cho BoT-SORT trở thành một lựa chọn hữu ích cho các ứng dụng đòi hỏi sự chính xác cao trong việc theo dõi đối tượng.

So với DeepSORT, BoT-SORT có thể mang lại hiệu suất tốt hơn trong việc giảm thiểu sai số, điều này có ý nghĩa đặc biệt khi đối mặt với môi trường đa dạng và thay đổi. MHT-SORT, mặc dù cũng sử dụng phương pháp gia tăng tối thiểu, nhưng BoT-SORT có lợi thế với ưu điểm của kỹ thuật bình phương tối thiểu có trọng số, giúp nó đối mặt tốt hơn với các thách thức của quá trình theo dõi đối tượng.

Vì chúng tôi hoạt động trong lĩnh vực nhận diện sản phẩm trong các mặt hàng siêu thị bằng trí tuệ nhân tạo, quyết định chọn lựa giữa BoT-SORT, DeepSORT và MHT-SORT là

một quyết định chiến lược, chủ yếu dựa trên yêu cầu cụ thể của dự án và ưu tiên đặc biệt về độ chính xác và khả năng giảm thiểu sai số trong quá trình theo dõi đối tượng.

BoT-SORT, với khả năng giảm thiểu sai số cao hơn nhờ vào sử dụng kỹ thuật bình phương tối thiểu có trọng số, đã được chọn làm giải pháp phù hợp cho ứng dụng của chúng tôi. Trong môi trường siêu thị, nơi có sự biến động lớn về kích thước và hình dạng của sản phẩm, độ chính xác và sự linh hoạt trong việc theo dõi đối tượng là rất quan trọng.

Do đó, chúng tôi tin rằng sự chọn lựa BoT-SORT làm thuật toán theo dõi đối tượng chính cho dự án của mình sẽ giúp tối ưu hóa khả năng nhận diện sản phẩm trong môi trường siêu thị, mang lại trải nghiệm mua sắm hiệu quả và quản lý hàng tồn kho chính xác.

iii. Utralytics:

Utralytics, một thư viện và framework mã nguồn mở xuất phát từ PyTorch, đang thu hút sự chú ý lớn nhờ vào tính linh hoạt và đa nhiệm đa dạng của nó. Được thiết kế để đáp ứng nhiều nhu cầu trong lĩnh vực thị giác máy tính, Utralytics không chỉ giới hạn ở một nhiệm vụ cụ thể mà còn có khả năng thực hiện một loạt các tác vụ, bao gồm nhận diện đối tượng, phân loại, phát hiện keypoint và theo dõi đối tượng.

Utralytics không chỉ là một giải pháp đơn thuần cho một công việc cụ thể, mà còn mang lại một giải pháp toàn diện cho các dự án đa nhiệm. Sự linh hoạt của nó cho phép tích hợp và tùy chỉnh linh hoạt theo yêu cầu cụ thể của dự án, làm cho nó trở thành một lựa chọn mạnh mẽ cho các ứng dụng yêu cầu tính năng đa dạng và phức tạp.

Chúng tôi đã đưa ra quyết định chọn Utralytics làm thuật toán chính cho dự án của mình, không chỉ vì tính đa dạng và hiệu suất cao, mà còn vì Utralytics có những đặc điểm độc đáo và không giới hạn trong việc đối mặt với nhiều thách thức của dự án. Sự khả năng tùy chỉnh linh hoạt và khả năng xử lý tốt đối với dữ liệu lớn là những yếu tố quan trọng giúp Utralytics nổi bật và làm cho nó trở thành một giải pháp xuất sắc đối với dự án của chúng tôi.

PHẦN IV. CÀI ĐẶT VÀ KIỂM THỬ

1. Cài đặt

Đầu tiên, tiến hành cài đặt ba thư viện Python cần thiết cho dự án của mình. Tôi sử dụng lệnh pip để cài đặt Ultralytics, Super Gradients và phiên bản 0.6.12 của thư viện timm.

```
!pip install ultralytics
!pip install super-gradients
!pip install timm==0.6.12
```

Để có thể truy cập dữ liệu và mô hình từ Google Drive, tôi sử dụng thư viện google.colab để kết nối Google Drive với Colab.

```
from google.colab import drive
drive.mount('/content/drive')
```

Tiếp theo, tôi sử dụng thư viện Ultralytics để tương tác với mô hình YOLOv8. Trong đoạn mã này, tôi sẽ tiến hành kiểm tra xem YOLOv8 đã được cài đặt thành công hay chưa.

```
from ultralytics import YOLO

# Load the YOLOv8 model
model = YOLO('yolov8n.pt')
# Display model information (optional)
model.info()
```

Đoạn mã JavaScript và Python dưới đây tạo một video stream sử dụng webcam và hiển thị nó trong môi trường Colab. Đoạn mã này sẽ hiển thị video trực tiếp từ webcam trong Colab và trả về thông tin về thời gian tạo, hiển thị và chụp frame, cùng với base64-encoded data URL của frame đó. Dưới đây là mô tả ngắn về từng phần chính của mã:

- JavaScript Code:

removeDom(): Hàm này được sử dụng để xóa các phần tử DOM (Document Object Model) liên quan đến video stream khi cần.

onAnimationFrame(): Hàm này được gọi mỗi khi một frame mới của video được hiển thị. Nó sử dụng requestAnimationFrame để liên tục gọi nó cho đến khi biến

shutdown trở thành true. Nếu có một hàm pendingResolve, nó sẽ trả về ảnh hiện tại dưới dạng base64-encoded data URL.

createDom(): Hàm này tạo và hiển thị các phần tử DOM, bao gồm video, hình ảnh và các thông báo.

stream_frame(label, imgData): Hàm này được gọi từ Python để trả về frame hiện tại của video stream. Nó sẽ đợi cho đến khi có một frame mới, sau đó trả về thông tin về thời gian tạo, hiển thị và chụp frame, cùng với base64-encoded data URL của frame đó.

- Python Code:

video_stream(): Hàm này định nghĩa JavaScript code và sử dụng display(js) để hiển thị nó trong Colab.

video_frame(label, bbox): Hàm này gọi hàm stream_frame trong JavaScript thông qua eval_js để nhận thông tin về frame hiện tại của video stream từ webcam.

```
from IPython.display import display, Javascript, Image
# JavaScript to properly create our live video stream using our webcam
as input

def video_stream():
  js = Javascript('''
    var video;
    var div = null;
    var stream;
    var captureCanvas;
    var imgElement;
    var labelElement;

    var pendingResolve = null;
    var shutdown = false;

    function removeDom() {
       stream.getVideoTracks()[0].stop();
       video.remove();
```

```
        div.remove();
        video = null;
        div = null;
        stream = null;
        imgElement = null;
        captureCanvas = null;
        labelElement = null;
    }

    function onAnimationFrame() {
      if (!shutdown) {
        window.requestAnimationFrame(onAnimationFrame);
      }
      if (pendingResolve) {
        var result = "";
        if (!shutdown) {
          captureCanvas.getContext('2d').drawImage(video,  0,  0,  640,
480);
          result = captureCanvas.toDataURL('image/jpeg', 0.8)
        }
        var lp = pendingResolve;
        pendingResolve = null;
        lp(result);
      }
    }

    async function createDom() {
      if (div !== null) {
        return stream;
      }

      div = document.createElement('div');
      div.style.border = '2px solid black';
      div.style.padding = '3px';
```

```
    div.style.width = '100%';
    div.style.maxWidth = '600px';
    document.body.appendChild(div);

    const modelOut = document.createElement('div');
    modelOut.innerHTML = "Status:";
    labelElement = document.createElement('span');
    labelElement.innerText = 'No data';
    labelElement.style.fontWeight = 'bold';
    modelOut.appendChild(labelElement);
    div.appendChild(modelOut);

    video = document.createElement('video');
    video.style.display = 'block';
    video.width = div.clientWidth - 6;
    video.setAttribute('playsinline', '');
    video.onclick = () => { shutdown = true; };
    stream = await navigator.mediaDevices.getUserMedia(
        {video: { facingMode: "environment"}});
    div.appendChild(video);

    imgElement = document.createElement('img');
    imgElement.style.position = 'absolute';
    imgElement.style.zIndex = 1;
    imgElement.onclick = () => { shutdown = true; };
    div.appendChild(imgElement);

    const instruction = document.createElement('div');
    instruction.innerHTML =
        '' +
        'Bấm vào video để dừng';
    div.appendChild(instruction);
    instruction.onclick = () => { shutdown = true; };
```

```
    video.srcObject = stream;
    await video.play();

    captureCanvas = document.createElement('canvas');
    captureCanvas.width = 640; //video.videoWidth;
    captureCanvas.height = 480; //video.videoHeight;
    window.requestAnimationFrame(onAnimationFrame);

    return stream;
  }
  async function stream_frame(label, imgData) {
    if (shutdown) {
      removeDom();
      shutdown = false;
      return '';
    }

    var preCreate = Date.now();
    stream = await createDom();

    var preShow = Date.now();
    if (label != "") {
      labelElement.innerHTML = label;
    }

    if (imgData != "") {
      var videoRect = video.getClientRects()[0];
      imgElement.style.top = videoRect.top + "px";
      imgElement.style.left = videoRect.left + "px";
      imgElement.style.width = videoRect.width + "px";
      imgElement.style.height = videoRect.height + "px";
      imgElement.src = imgData;
    }
```

```
      var preCapture = Date.now();
      var result = await new Promise(function(resolve, reject) {
        pendingResolve = resolve;
      });
      shutdown = false;

      return {'create': preShow - preCreate,
              'show': preCapture - preShow,
              'capture': Date.now() - preCapture,
              'img': result};
    }
    ''')

  display(js)

def video_frame(label, bbox):
  data = eval_js('stream_frame("{}", "{}")'.format(label, bbox))
  return data
```

Tiếp theo, bổ sung các chức năng để chuyển đổi giữa đối tượng hình ảnh JavaScript và hình ảnh OpenCV, cũng như chuyển đổi bounding box (hộp giới hạn) của hình ảnh thành chuỗi byte base64 để có thể hiển thị trên video stream. Dưới đây là mô tả ngắn về từng hàm:

- Hàm js_to_image(js_reply): Chuyển đổi đối tượng hình ảnh từ JavaScript thành hình ảnh OpenCV.

+ Đầu Vào:

js_reply: Đối tượng hình ảnh từ JavaScript.

+ Đầu Ra:

img: Hình ảnh OpenCV (BGR).

- Hàm bbox_to_bytes(bbox_array): Chuyển đổi bounding box (hộp giới hạn) của hình ảnh thành chuỗi byte base64 để có thể hiển thị trên video stream.

+ Đầu Vào:

bbox_array: Numpy array (pixels) chứa hình chữ nhật để overlay lên video

stream.

+ Đầu Ra:

bbox_bytes: Chuỗi byte base64 của hình chữ nhật overlay.

```
import PIL
import io
from base64 import b64decode, b64encode

# function to convert the JavaScript object into an OpenCV image
def js_to_image(js_reply):
  """
  Params:
          js_reply: JavaScript object containing image from webcam
  Returns:
          img: OpenCV BGR image
  """
  # decode base64 image
  image_bytes = b64decode(js_reply.split(',')[1])
  # convert bytes to numpy array
  jpg_as_np = np.frombuffer(image_bytes, dtype=np.uint8)
  # decode numpy array into OpenCV BGR image
  img = cv2.imdecode(jpg_as_np, flags=1)

  return img

# function to convert OpenCV Rectangle bounding box image into base64
byte string to be overlayed on video stream
def bbox_to_bytes(bbox_array):
  """
  Params:
          bbox_array: Numpy array (pixels) containing rectangle to
overlay on video stream.
  Returns:
        bytes: Base64 image byte string
  """
```

```
  # convert array into PIL image
  bbox_PIL = PIL.Image.fromarray(bbox_array, 'RGBA')
  iobuf = io.BytesIO()
  # format bbox into png for return
  bbox_PIL.save(iobuf, format='png')
  # format return string
  bbox_bytes                                                    =
'{}'.format((str(b64encode(iobuf.getvalue()),
'utf-8')))

  return bbox_bytes
```

Đoạn mã tiếp theo sẽ test nhận diện, theo dõi bằng model pretrain dùng webcam máy tính tận dụng gpt của Google Colab. Các bước chính bao gồm:

1. Khởi tạo video stream từ webcam.
2. Trong mỗi vòng lặp, nhận frame mới từ video stream.
3. Sử dụng mô hình YOLOv8 để theo dõi vật thể trên frame.
4. Vẽ bounding box và track ID lên frame.
5. Sử dụng lịch sử theo dõi để vẽ đường dẫn theo dõi của đối tượng.
6. Overlay bounding box lên frame và hiển thị.

Đoạn mã này kết hợp giữa các thư viện như Ultralytics, OpenCV, và các kỹ thuật JavaScript để tạo ra một ứng dụng theo dõi vật thể trong thời gian thực trong môi trường Colab.

```
from google.colab.output import eval_js
from google.colab.patches import cv2_imshow
from collections import defaultdict
import numpy as np
import cv2

video_stream()

bbox = ' '
count = 0
```

```
track_history = defaultdict(lambda: [])

while True:
  js_reply = video_frame('test', bbox)
  if not js_reply:
      break

  # convert JS response to OpenCV Image
  frame = js_to_image(js_reply["img"])

  # create transparent overlay for bounding box
  bbox_array = np.zeros([480,640,4], dtype=np.uint8)

  #get results from model tracking
  results = model.track(frame, persist=True)



  # Get the boxes and track IDs
  boxes = results[0].boxes.xywh.cpu()
  track_ids = results[0].boxes.id.int().cpu().tolist()

  # Visualize the results on the frame
  annotated_frame = results[0].plot()

  # Plot the tracks
  for box, track_id in zip(boxes, track_ids):
    x, y, w, h = box
    track = track_history[track_id]
    track.append((float(x), float(y)))  # x, y center point
    if len(track) > 30:  # retain 90 tracks for 90 frames
        track.pop(0)

    # Draw the tracking lines
```

```
    points = np.hstack(track).astype(np.int32).reshape((-1, 1, 2))
    # bbox_array = cv2.polylines(bbox_array, [points], isClosed=False,
color=(230, 230, 230), thickness=10)


    left = int(int(x) - (int(w)/2))
    right = int (int(y) - (int(h)/2))
    width = int(w)
    height = int(h)



    bbox_array                                                          =
cv2.rectangle(bbox_array,(left,right),(left+width,right+height),(255,0
,0),2)
    bbox_array   =   cv2.putText(bbox_array,str(track_id),(left,right),
cv2.FONT_HERSHEY_SIMPLEX, 0.5,color=(0, 230,0), thickness=10)


    bbox_array[:,:,3] = (bbox_array.max(axis = 2) > 0 ).astype(int) *
255
    # convert overlay of bbox into bytes
    bbox_bytes = bbox_to_bytes(bbox_array)
    # update bbox so next frame gets new overlay
    bbox = bbox_bytes
```

Tiếp theo tôi định nghĩa một hàm take_photo để chụp ảnh từ webcam trong môi trường Google Colab. Dưới đây là giải thích ngắn gọn về cách nó hoạt động:

- JavaScript Code:

Một đoạn mã JavaScript được sử dụng để tạo giao diện cho việc chụp ảnh từ webcam. Nó tạo ra một video element, hiển thị video từ webcam, và một nút "Capture" để chụp ảnh.

Khi nút "Capture" được nhấp, ảnh được chụp và trả về dưới dạng base64-encoded data URL.

- Python Code:

Hàm take_photo hiển thị giao diện chụp ảnh bằng cách sử dụng mã JavaScript.

Nó sử dụng eval_js để gọi hàm JavaScript takePhoto và nhận kết quả (ảnh chụp)

dưới dạng base64-encoded data URL.

Chuyển đổi base64-encoded data URL thành hình ảnh OpenCV bằng hàm js_to_image.

Chạy mô hình YOLOv8 trên frame chụp ảnh để nhận diện vật thể.

Hiển thị frame chụp ảnh với bounding box và kết quả nhận diện.

Lưu frame chụp ảnh thành một tệp tin và trả về tên tệp.

Hàm này giúp người dùng chụp ảnh từ webcam và thực hiện nhận diện vật thể trên ảnh chụp đó bằng mô hình YOLOv8.

```
from IPython.display import display, Javascript
from google.colab.output import eval_js

def take_photo(filename='photo.jpg', quality=0.8):
  js = Javascript('''
    async function takePhoto(quality) {
      const div = document.createElement('div');
      const capture = document.createElement('button');
      capture.textContent = 'Capture';
      div.appendChild(capture);

      const video = document.createElement('video');
      video.style.display = 'block';
      const stream = await navigator.mediaDevices.getUserMedia({video:
true});

      document.body.appendChild(div);
      div.appendChild(video);
      video.srcObject = stream;
      await video.play();

      // Resize the output to fit the video element.

google.colab.output.setIframeHeight(document.documentElement.scrollHei
ght, true);
```

```
    // Wait for Capture to be clicked.
    await new Promise((resolve) => capture.onclick = resolve);

    const canvas = document.createElement('canvas');
    canvas.width = video.videoWidth;
    canvas.height = video.videoHeight;
    canvas.getContext('2d').drawImage(video, 0, 0);
    stream.getVideoTracks()[0].stop();
    div.remove();
    return canvas.toDataURL('image/jpeg', quality);
  }
  ''')
display(js)
data = eval_js('takePhoto({})'.format(quality))

frame = js_to_image(js_reply['img'])
# Run YOLOv8 inference on the frame
results = model(frame)

# Visualize the results on the frame
annotated_frame = results[0].plot()

# Display the annotated frame
cv2.imwrite(filename, annotated_frame)

return filename
```

Tôi sử dụng hàm take_photo để chụp ảnh từ webcam, sau đó hiển thị ảnh đã chụp lên môi trường Colab. Dưới đây là giải thích ngắn gọn:

- filename = take_photo('photo.jpg'): Gọi hàm take_photo để chụp ảnh từ webcam và lưu lại với tên tệp 'photo.jpg'.

- print('Saved to {}'.format(filename)): Hiển thị thông điệp cho biết ảnh đã được lưu và đường dẫn tới tệp.

- display(Image(filename)): Sử dụng thư viện IPython để hiển thị ảnh đã chụp trong môi trường Colab.

- except Exception as err: Bắt các ngoại lệ có thể xảy ra, chẳng hạn như nếu người dùng không có webcam hoặc không cho phép trang web truy cập vào webcam. Các lỗi sẽ được in ra màn hình

```
from IPython.display import Image
try:
  filename = take_photo('photo.jpg')
  print('Saved to {}'.format(filename))

  # Show the image which was just taken.
  display(Image(filename))
except Exception as err:
  # Errors will be thrown if the user does not have a webcam or if they
do not
  # grant the page permission to access it.
  print(str(err))
```

Đoạn mã Python bên dưới sử dụng mô hình YOLOv8 để thực hiện theo dõi đối tượng trong thời gian thực từ luồng video webcam trong môi trường Google Colab. Dưới đây là giải thích chi tiết:

1. Khởi Tạo:

   Thiết lập luồng video và biến.

   Chuỗi bbox lưu trạng thái overlay.

2. Vòng Lặp Chính:

   Lặp liên tục để lấy và xử lý các frame từ video.

3. Chụp và Chuyển Đổi Frame:

   js_reply = video_frame('test', bbox) - Lấy frame từ luồng video và chuyển đổi phản hồi JavaScript thành hình ảnh OpenCV.

4. Khởi Tạo Mảng Overlay:

   bbox_array = np.zeros([480, 640, 4], dtype=np.uint8) - Tạo mảng trống để vẽ bounding box.

5. Gọi Mô Hình Để Theo Dõi:

   - results = model.track(data='coco8.yaml') - Gọi mô hình để thực hiện theo dõi đối tượng. Có thể liên quan đến cấu hình COCO dataset.

   - results = model(frame) - Thực hiện dự đoán trên frame hiện tại bằng mô hình.

6. Lấy Kết Quả và Vẽ Bounding Box:

Lấy thông tin về bounding box và ID theo dõi từ kết quả mô hình và vẽ chúng lên frame.

7. Xử Lý Overlay và Cập Nhật Bbox:

   - Tạo một overlay trong suốt cho bounding box và chuyển đổi nó thành bytes để hiển thị trong môi trường Colab.

   - Cập nhật biến bbox để frame tiếp theo có thể nhận được overlay mới.

8. Lặp Lại Quá Trình Cho Từng Frame.

```
from google.colab.output import eval_js
from google.colab.patches import cv2_imshow
from collections import defaultdict
import numpy as np
import cv2

video_stream()

bbox = ' '
count = 0

track_history = defaultdict(lambda: [])

while True:
  js_reply = video_frame('test', bbox)
  if not js_reply:
      break

  # convert JS response to OpenCV Image
  frame = js_to_image(js_reply["img"])
```

```
  # create transparent overlay for bounding box
  bbox_array = np.zeros([480,640,4], dtype=np.uint8)

  #model call
  results = model.track(data='coco8.yaml')
  #get results from model tracking
  results = model(frame)



  # Get the boxes and track IDs
  boxes = results[0].boxes.xywh.cpu()
  track_ids = results[0].boxes.id.int().cpu().tolist()

  # Visualize the results on the frame
  annotated_frame = results[0].plot()

  # Plot the tracks
  for box, track_id in zip(boxes, track_ids):
    x, y, w, h = box
    track = track_history[track_id]
    track.append((float(x), float(y)))  # x, y center point
    if len(track) > 30:  # retain 90 tracks for 90 frames
        track.pop(0)

    # Draw the tracking lines
    points = np.hstack(track).astype(np.int32).reshape((-1, 1, 2))
    # bbox_array = cv2.polylines(bbox_array, [points], isClosed=False,
color=(230, 230, 230), thickness=10)

    left = int(int(x) - (int(w)/2))
    right = int (int(y) - (int(h)/2))
    width = int(w)
    height = int(h)
```

```
    bbox_array                                                    =
cv2.rectangle(bbox_array,(left,right),(left+width,right+height),(255,0
,0),2)
    bbox_array   =   cv2.putText(bbox_array,str(track_id),(left,right),
cv2.FONT_HERSHEY_SIMPLEX, 0.5,color=(0, 230,0), thickness=10)

    bbox_array[:,:,3] = (bbox_array.max(axis = 2) > 0 ).astype(int) *
255
    # convert overlay of bbox into bytes
    bbox_bytes = bbox_to_bytes(bbox_array)
    # update bbox so next frame gets new overlay
    bbox = bbox_bytes
```

Tiếp theo trong quá trình huấn luyện mô hình AI, chúng tôi sẽ bắt đầu quy trình đánh dấu nhãn cho ảnh. Đối với công đoạn này, đầu tiên chúng tôi sẽ tiến hành quay nhiều đoạn video các sản phẩm của chúng tôi với nhiều góc độ, độ sáng khác nhau. Tiếp theo chúng tôi sẽ thực hiện một đoạn mã để cắt đoạn video thành nhiều frame ảnh và resize lại các bức ảnh đó. Chi tiết đoạn mã như sau:

- Tôi sử dụng thư viện glob để lấy danh sách các tệp tin video có định dạng MOV từ đường dẫn /content/drive/MyDrive/data/. Danh sách này được lưu trữ trong biến videoList. Mỗi video được mở sử dụng cv2.VideoCapture và kiểm tra xem việc mở video có thành công hay không. Nếu không thành công, một thông báo lỗi sẽ được hiển thị. Nếu thành công, đối tượng cv2.VideoCapture sẽ được thêm vào danh sách caps. Sau đó kiểm tra xem thư mục /content/drive/MyDrive/data/image/ đã tồn tại hay chưa, nếu chưa, nó sẽ được tạo mới. Mỗi lần lặp, khung hình tiếp theo của mỗi video trong danh sách caps được đọc và chuyển đổi kích thước thành (640, 480) bằng cách sử dụng cv2.resize. Sau đó, ảnh được lưu vào thư mục đã được tạo trước đó với tên tệp tin được đặt dựa trên biến currentframe. Mỗi lần lưu ảnh, biến currentframe được tăng lên để đảm bảo tên tệp tin là duy nhất. Nếu không còn khung hình nào trong video, vòng lặp sẽ thoát.

```
import cv2
import os
import glob
```

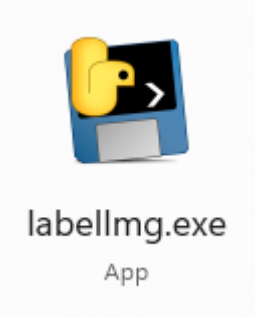
```
caps = []
videoList=glob.glob(r'/content/drive/MyDrive/data/*.MOV')
for path in videoList:
    cap = cv2.VideoCapture(path)
    if not cap.isOpened():
        print ("error opening ", path)
    else:
        caps.append(cap)

try:

    # creating a folder named data
    if not os.path.exists('/content/drive/MyDrive/data/image/'):
        os.makedirs('/content/drive/MyDrive/data/image/')

# if not created then raise error
except OSError:
    print ('Error: Creating directory of data')
currentframe = 0


while(True):

  # reading from frame
  for cap in caps:
    ret,frame=cap.read()
    frame = cv2.resize(frame, (640, 480))

    if ret:
        # if video is still left continue creating images
        name = '/content/drive/MyDrive/data/image/' +
str(currentframe) + '.jpg'
        print ('Creating...' + name)

        # writing the extracted images
        cv2.imwrite(name, frame)

        # increasing counter so that it will
        # show how many frames are created
        currentframe += 1
    else:
        break

# Release all space and windows once done
cam.release()
cv2.destroyAllWindows()
```

Sau khi thực hiện đoạn mã ta sẽ được kết quả như sau:

```
Creating.../content/drive/MyDrive/data/image/0.jpg
```

```
Creating.../content/drive/MyDrive/data/image/1.jpg
Creating.../content/drive/MyDrive/data/image/2.jpg
Creating.../content/drive/MyDrive/data/image/3.jpg
Creating.../content/drive/MyDrive/data/image/4.jpg
Creating.../content/drive/MyDrive/data/image/5.jpg
Creating.../content/drive/MyDrive/data/image/6.jpg
Creating.../content/drive/MyDrive/data/image/7.jpg
Creating.../content/drive/MyDrive/data/image/8.jpg
Creating.../content/drive/MyDrive/data/image/9.jpg
Creating.../content/drive/MyDrive/data/image/10.jpg

.....

Creating.../content/drive/MyDrive/data/image/4654.jpg
```

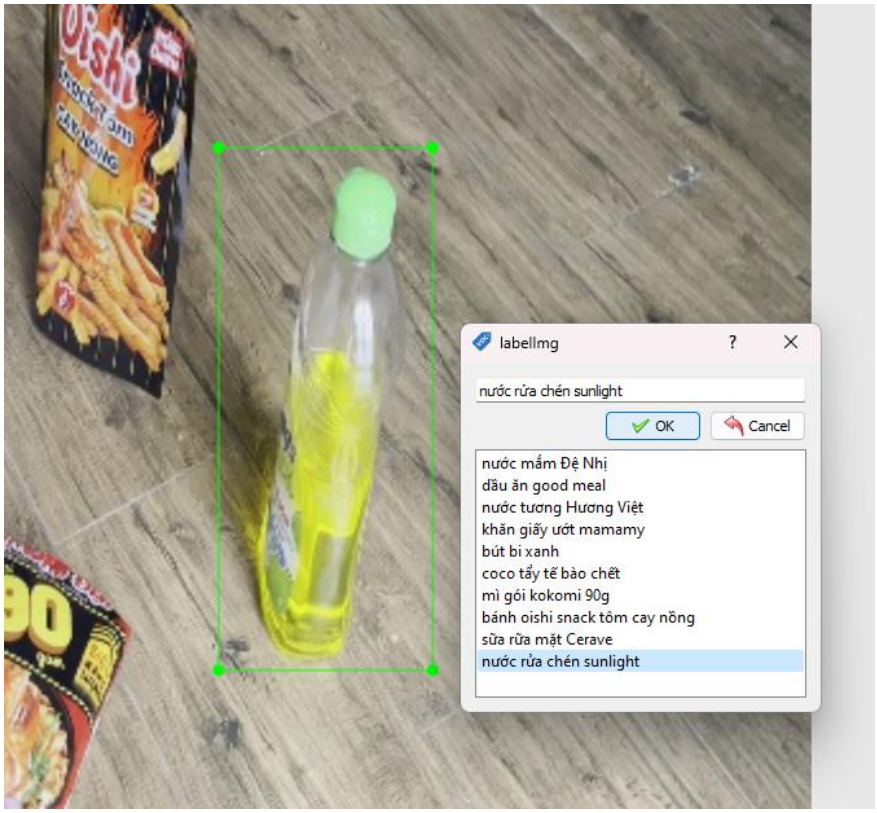
Kết quả ta thu được 4654 bức ảnh được cắt từ các video đã quay. Bước tiếp theo, để thực hiện quy trình đánh dấu, chúng tôi sử dụng một công cụ đánh dấu hình ảnh, giúp gắn nhãn mỗi sản phẩm trong hình. Công cụ chúng tôi chọn là "labelImg". Chúng tôi chọn công cụ này vì tính dễ cài đặt, giao diện cực kì đơn giản và dễ làm quen. Công cụ này mang lại khả năng đánh dấu nhãn cho ảnh rất hiệu quả, nhược điểm của công cụ này chính là không có khả năng tự động đánh dấu nhãn mà phải thực hiện hoàn toàn thủ công, vì vậy nên sẽ mất khá nhiều thời gian để đánh dấu hơn 4500 bức ảnh.



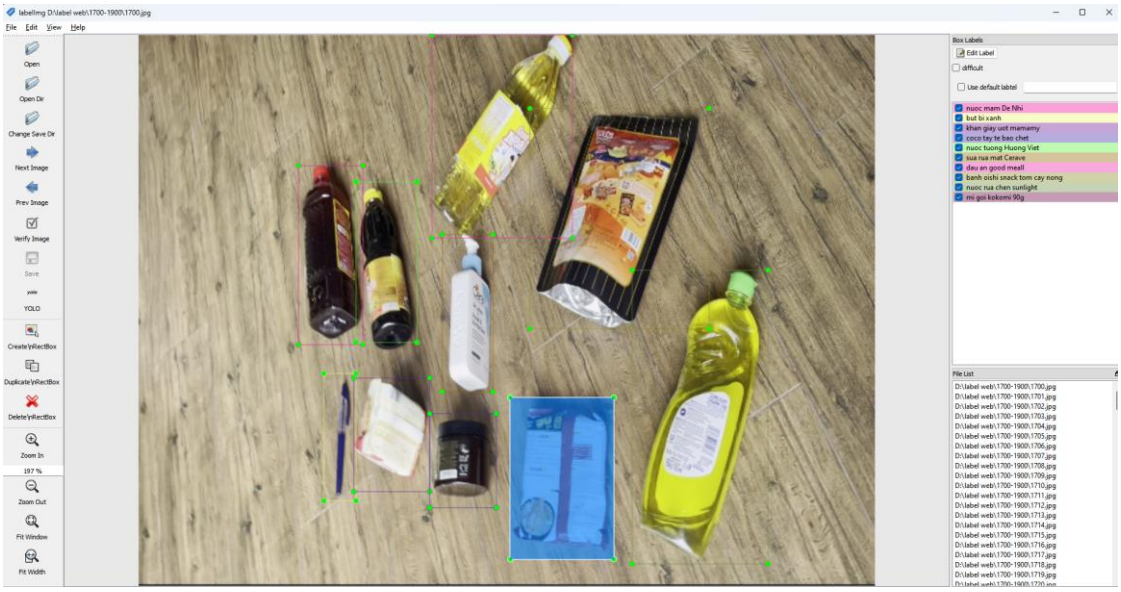
Hình 3. Công cụ labelImg.exe

Quy trình đánh dấu nhãn cho ảnh:

1. Đầu tiên chúng tôi sẽ chọn thu mục gốc lưu trữ các bức ảnh cần đánh dấu ở mục "Open Dir".
2. Tiếp theo chúng tôi sẽ đánh dấu nhãn cho từng ảnh bừng cách ấn phím "W", khi này chúng tôi sẽ có thể kéo chuột tạo ra một ô vuông bao quanh lấy sản phẩm cần đánh dấu
3. Nhập tên của sản phẩm vào ô trống và ấn "OK" để lưu nhãn.
4. Tiếp tục lặp lại đối với các sản phẩm khác.



Hình 4. Quy trình đánh dấu nhãn cho một sản phẩm



Hình 5. Công cụ đánh dấu nhãn

Sau khi hoàn thành quy trình đánh dấu cho tất cả các ảnh, chúng ta sẽ có được một tập dữ liệu nhãn phong phú, đặc trưng cho các đối tượng cần mô hình nhận diện nhằm tạo ra bộ dữ liệu huấn luyện đa dạng và đầy đủ để mô hình có thể học từ.

Quá trình này không chỉ quan trọng để nâng cao độ chính xác của mô hình, mà còn giúp xây dựng sự đa dạng và phong phú trong dữ liệu huấn luyện, tạo ra một mô

hình AI mạnh mẽ và linh hoạt hơn."

Tiếp theo tôi sử dụng đoạn mã Python trên dùng thư viện Ultralytics và mô hình YOLOv8 để thực hiện việc huấn luyện mô hình. Dưới đây là giải thích:

- Khởi Tạo Mô Hình:

model = YOLO('yolov8x.pt'): Tạo một đối tượng mô hình YOLOv8 sử dụng trọng số được đặt tên là 'yolov8x.pt'.

- Thiết Lập và Huấn Luyện:

results = model.train(data='config.yaml', epochs=100, imgsz=640): Thực hiện quá trình huấn luyện mô hình YOLOv8.

data='config.yaml': Sử dụng cấu hình từ tệp 'config.yaml', chứa thông tin về dữ liệu huấn luyện, kiến trúc mô hình, và các thông số cần thiết khác.

epochs=100: Thiết lập số lượng epochs (chu kỳ huấn luyện) là 100.

imgsz=640: Kích thước ảnh đầu vào được resize lại thành 640x640 pixel trong quá trình huấn luyện.

```
from ultralytics import YOLO


# highest reliability model
model = YOLO('yolov8x.pt')


# train vs file setup khoản 1 100 epochs, với kích thức ảnh được resize lại: 640
results = model.train(data='config.yaml', epochs=100, imgsz=640)
```

Tiếp theo tôi sẽ tiếp tục train cho AI từ file tốt nhất

```
from ultralytics import YOLO



model = YOLO('/content/drive/MyDrive/data/runs/detect/train/weights/best.pt')


results = model.train(data='config.yaml', epochs=50, imgsz=640)
```

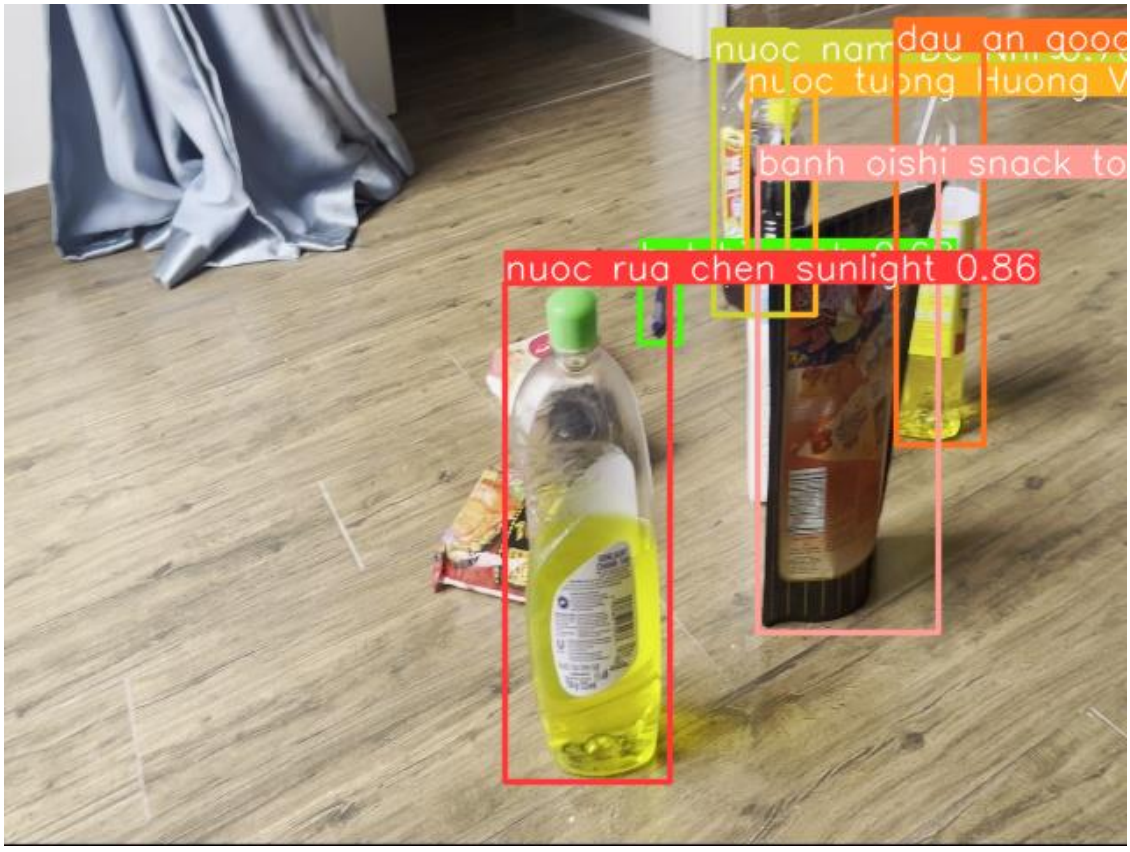
Tiếp tục train cho AI qua nhiều góc độ khác nhau, càng được train qua nhiều góc độ thì độ chính xác của hệ thống nhận diện càng tăng cao.

Hình 6. Quá trình train cho AI

Tôi sẽ tiến hành kiểm thử bằng cách cho load một mô hình nhận diện đối tượng YOLO đã được huấn luyện từ đường dẫn '/content/drive/MyDrive/data/runs/detect/train/weights/best.pt'. Sau đó, nó sử dụng mô hình này để dự đoán và tạo ra một hình ảnh đã được chú thích với các bounding box và nhãn cho đối tượng trên ảnh '500.jpg'.

```
from ultralytics import YOLO
from google.colab.patches import cv2_imshow
import cv2



model                                                                    =
YOLO('/content/drive/MyDrive/data/runs/detect/train/weights/best.pt')


results = model('/content/drive/MyDrive/data/image/500.jpg')
# Visualize the results on the frame
annotated_frame = results[0].plot()
```

```
# # Display the annotated frame
cv2_imshow(annotated_frame)
```




```
image 1/1 /content/drive/MyDrive/data/image/500.jpg: 480x640 1 nuoc rua chen sunlight, 1 banh oishi snack tom cay nong, 1 dau an good meal, 1 nuoc tuong Huong Viet, 1 nuoc nam De Nhi, 1 but bi xanh, 62.1ms
Speed: 1.6ms preprocess, 62.1ms inference, 1.4ms postprocess per image at shape (1, 3, 480, 640)
```

Hình 7. Kết quả của quá trình train AI

Kết quả trả về từ đoạn mã là một hình ảnh đã được chú thích (annotated) bởi mô hình YOLO. Hình ảnh này bao gồm các bounding box (khung giới hạn) và nhãn cho đối tượng mà mô hình đã nhận diện được trên ảnh '500.jpg'. Cụ thể, đối tượng results là một đối tượng Ultralytics chứa thông tin chi tiết về dự đoán của mô hình.

Qua kết quả thu được ta có thể thấy mô hình YOLO cho ra kết quả nhận diện với độ chính xác cao, tốc độ cực kì nhanh, hầu hết các sản phẩm đều cho ra kết quả nhận diện đúng với sản phẩm thực tế, mặc dù vậy vẫn còn một số sản phẩm bị che khuất bởi các sản phẩm khác như “mì gói kokomi 90g” , “khăn giấy ướt mamamy” và

“coco tẩy tế bào chết” là vẫn không nhận diện được.

Tiếp theo tôi sẽ tiếp tục thử sử dụng mô hình YOLOv8 từ Ultralytics để thực hiện việc theo dõi đối tượng trong video thông qua việc sử dụng một model Sort khác chính là ByteTrack. Đoạn mã này sử dụng YOLOv8 với ByteTrack để theo dõi và đánh dấu đối tượng trong video, sau đó ghi lại video đầu ra với thông tin đánh dấu.

Dưới đây là giải thích công dụng và vai trò của đoạn mã:

1. Load Model: Sử dụng YOLOv8 để tải mô hình đã được huấn luyện. Trong trường hợp này, mô hình được sử dụng để theo dõi đối tượng trong video.
2. Mở Video: Mở video từ đường dẫn đã được xác định (video_path).
3. Xác định Thuộc Tính Video: Lấy thông tin chiều cao, chiều rộng và số kênh màu của frame video.
4. Định nghĩa VideoWriter: Sử dụng OpenCV để định nghĩa đối tượng VideoWriter để ghi video đầu ra. Các thông số của VideoWriter được thiết lập dựa trên thuộc tính của video nguồn.
5. Vòng Lặp Video: Duyệt qua từng frame trong video.
6. Chạy YOLOv8 Tracking: Áp dụng thuật toán theo dõi của YOLOv8 sử dụng ByteTrack cho từng frame. Kết quả trả về là một frame đã được đánh dấu với các hộp giới hạn và nhãn của các đối tượng.
7. Ghi Video Đầu Ra: Ghi frame đã được đánh dấu vào video đầu ra.
8. Dừng Lặp: Nếu người dùng nhấn phím 'q', quá trình lặp sẽ kết thúc.
9. Giải phóng Tài Nguyên: Giải phóng bộ nhớ và đóng cửa sổ hiển thị nếu đã mở.

```
import cv2
from ultralytics import YOLO

# Load the YOLOv8 model

model = YOLO('runs/detect/train/weights/best.pt')  # load a trained
model

# Open the video file
video_path = 'for_test.mp4'
```

```
cap = cv2.VideoCapture(video_path)
ret, frame = cap.read()

# Get the video properties
H, W, _ = frame.shape

# Define the codec and create VideoWriter object
output_path = 'output_ByteTrack_test.mp4'
out = cv2.VideoWriter(output_path, cv2.VideoWriter_fourcc(*'MP4V'), int(cap.get(cv2.CAP_PROP_FPS)), (W, H))



# Loop through the video frames
while cap.isOpened():
    # Read a frame from the video
    success, frame = cap.read()



    if success:
        # Run YOLOv8 tracking on the frame, persisting tracks between frames
        results = model.track(frame, persist=True, conf=0.5, iou=0.5, tracker="bytetrack.yaml")

        # Visualize the results on the frame
        annotated_frame = results[0].plot()
        out.write(annotated_frame)

        # Break the loop if 'q' is pressed
        if cv2.waitKey(1) & 0xFF == ord("q"):
            break
    else:
        # Break the loop if the end of the video is reached
        break
```

```
# Release the video capture object and close the display window
cap.release()
out.release()
cv2.destroyAllWindows()
```

Tiếp theo, triển khai một hàm post_to_api để thực hiện yêu cầu POST tới một API. Mã nguồn này có thể được sử dụng như là một bản cơ sở để thực hiện yêu cầu POST tới các API khác nhau bằng cách thay đổi url và data. Dưới đây là giải thích ngắn gọn về công dụng của mã nguồn:

1. post_to_api Function:

   Tham số:

   - url (str): Địa chỉ URL của điểm cuối API mà yêu cầu sẽ được gửi tới.
   - data (dict): Dữ liệu sẽ được gửi trong phần thân của yêu cầu.
   - headers (dict, tùy chọn): Các tiêu đề bổ sung cho yêu cầu.

   Trả về: Đối tượng phản hồi (response) từ API.

   Hoạt Động:

   - Hàm này sử dụng thư viện requests để thực hiện yêu cầu POST với dữ liệu JSON tới API.
   - Trả về đối tượng phản hồi, bao gồm thông tin như mã trạng thái (status_code) và nội dung phản hồi (json).

2. Example Usage:

- Sử dụng URL của JSONPlaceholder, một dịch vụ thử nghiệm API.
- Tạo một đối tượng JSON (payload_data) chứa dữ liệu yêu cầu.
- Gọi hàm post_to_api để gửi yêu cầu và nhận phản hồi từ API.
- In ra mã trạng thái của phản hồi và nội dung phản hồi.

```
import requests


def post_to_api(url, data, headers=None):
    """
    Sends a POST request to the specified API.
```

```
    Parameters:
    - url (str): The URL of the API endpoint.
    - data (dict): The data to be sent in the request body.
    - headers (dict, optional): Additional headers for the request.

    Returns:
    - response: The response object returned by the API.
    """
    response = requests.post(url, json=data, headers=headers)
    return response

# Example usage:
api_url = "https://jsonplaceholder.typicode.com/posts"
payload_data = {"title": "foo", "body": "bar", "userId": "1"}
response = post_to_api(api_url, payload_data)

print(response.status_code)
print(response.json())
```

Đoạn mã Python tiếp theo định nghĩa một hàm post để gửi yêu cầu POST tới một API với dữ liệu được truyền vào. Nó in ra mã trạng thái HTTP và nội dung phản hồi từ API sau khi yêu cầu được gửi đi.

```
def post(id, cls):
  # api_url = "https://jsonplaceholder.typicode.com/posts"
  api_url = "http://myapi/posts"
  payload_data = {"Productid": id, "name": cls}
  response = post_to_api(api_url, payload_data)

  print(response.status_code)
  print(response.json())
```

Dưới đây là một đoạn mã định nghĩa hàm post để gửi dữ liệu lên một API. Dưới đây là giải thích từng phần của đoạn mã:

- Hàm post:

- Hàm này nhận ba tham số: id, cls, và taken.

- id: ID của đối tượng được theo dõi.
- cls: Lớp của đối tượng được theo dõi.
- taken: Số lượng đối tượng đã được theo dõi.

- Định nghĩa API URL và Dữ liệu Payload:

- api_url: Đường dẫn URL của API, trong trường hợp này là "http://myapi/posts".
- prices: Một mảng chứa giá của các đối tượng tương ứng với các lớp.
- unit: Một mảng chứa đơn vị tính của các đối tượng tương ứng với các lớp.
- payload_data: Dữ liệu cần gửi lên API, bao gồm id, name (tên của đối tượng), price (giá), units (đơn vị tính), taken (số lượng), và payable (số lượng cần thanh toán).

- Gọi Hàm post_to_api và In Kết Quả:

- Hàm post_to_api chưa được định nghĩa trong đoạn mã bạn đã cung cấp, nhưng giả sử nó là một hàm khác được sử dụng để thực hiện gửi POST request đến API với dữ liệu payload.
- In ra mã trạng thái HTTP và phản hồi từ API.

```
def post(id, cls, taken):
  # api_url = "https://jsonplaceholder.typicode.com/posts"
  api_url = "http://myapi/posts"
  # payload_data = {"Productid": id, "name": cls}
  # class_names = ['nuoc rua chen sunlight', 'banh oishi snack tom cay
nong', 'dau an good meal', 'nuoc tuong Huong Viet', 'nuoc nam De Nhi',
'but bi xanh,', ' khan giay uot mamamy', 'coco tay te bao chet', 'mi
goi kokomi 90g', 'duong am Cerave']
  prices = ['24,300', '12.000', '41.000', '6.600', '25.000', '7,650',
'21,000', '115.000', '4,500', '370.000']
  unit = ['chai', 'bich', 'chai', 'chai', 'chai', 'cay', 'bich', 'hu',
'goi', 'hu']
  payload_data =
{"id":id,"name":cls,"price":price[cls],"units":unit[cls],"taken":taken
,"payable":taken}
  response = post_to_api(api_url, payload_data)

  print(response.status_code)
  print(response.json())
```

Tiếp theo tôi sử dụng mô hình YOLOv8 để theo dõi đối tượng trong video và hiển thị kết quả trên video đầu ra. Dưới đây là giải thích các phần chính của mã:

1. Load YOLOv8 Model:

   Sử dụng thư viện Ultralytics để tải mô hình YOLOv8 từ đường dẫn 'runs/detect/train/weights/best.pt'.

2. Xử lý Video:

   Mở video từ đường dẫn 'for_test.mp4' và lấy kích thước khung hình.

3. Ghi Video Đầu Ra:

   Tạo video đầu ra với cùng định dạng và tốc độ khung hình như video đầu vào.

4. Theo Dõi và Vẽ Đường Theo Dõi:

   - Sử dụng mô hình để theo dõi và nhận diện đối tượng trong mỗi khung hình.
   - Lấy thông tin về hộp giới hạn (boxes) và ID của các đối tượng được theo dõi.
   - Sử dụng các thông tin này để vẽ các đường dẫn theo dõi trên video.

5. Gửi Dữ Liệu Lên API:

   Nếu một đối tượng mới được phát hiện, sử dụng hàm post để gửi thông tin của đối tượng (ID và tên) lên một API thông qua URL 'http://myapi/posts'.

6. Ghi Video và Hiển Thị:

   Viết video đầu ra và hiển thị các khung hình đã được vẽ.

7. Dừng Vòng Lặp:

   Nếu phím 'q' được nhấn, vòng lặp sẽ dừng lại.

Điều này làm cho đoạn mã này là một chương trình theo dõi đối tượng trực quan, đồng thời gửi thông tin về đối tượng mới được phát hiện lên một API.

```
import cv2
from ultralytics import YOLO
from collections import defaultdict
import numpy as np

# Load the YOLOv8 model

model = YOLO('runs/detect/train/weights/best.pt')  # load a trained model

# Open the video file
video_path = 'for_test.mp4'
```

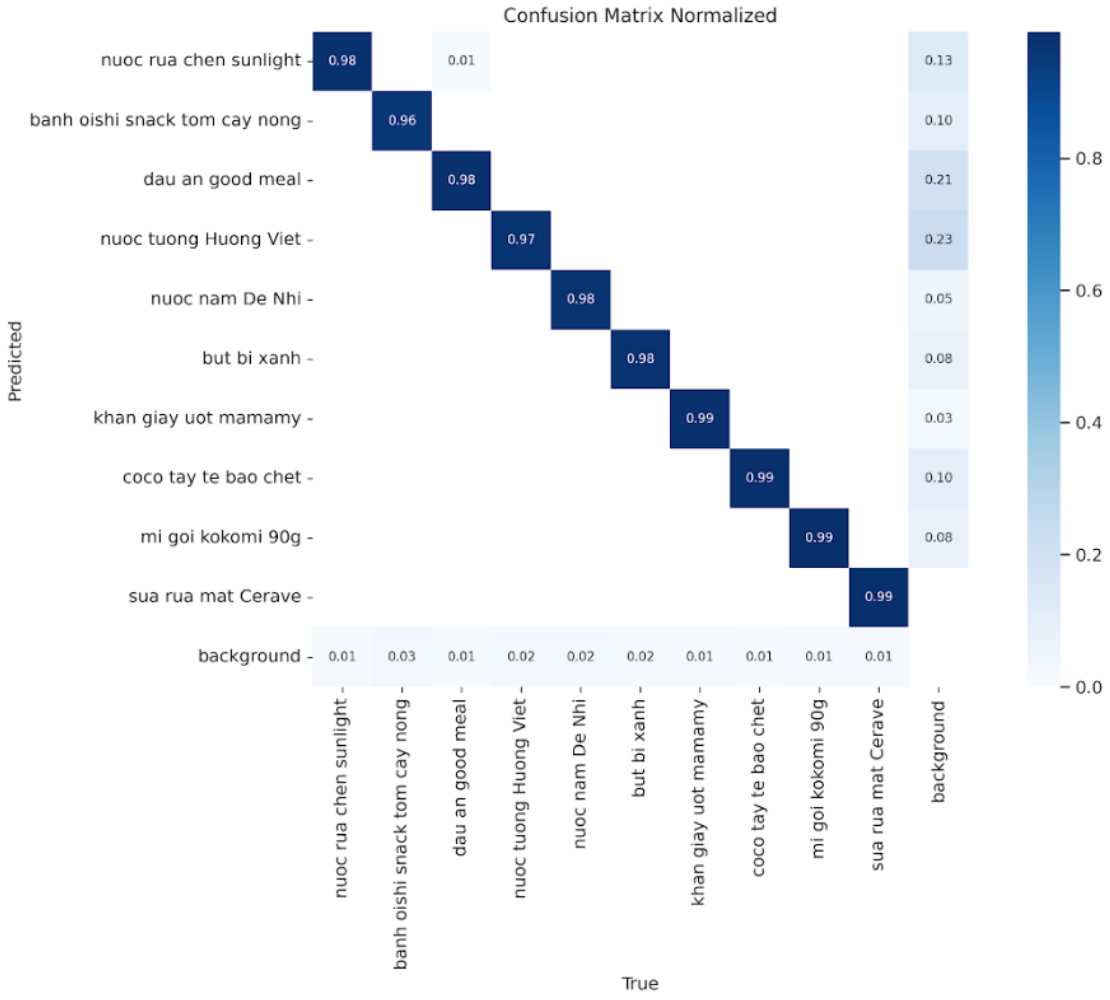
```
cap = cv2.VideoCapture(video_path)
ret, frame = cap.read()

# Get the video properties
H, W, _ = frame.shape

# Define the codec and create VideoWriter object
output_path = 'output_BoT-SORT_test.mp4'
out = cv2.VideoWriter(output_path, cv2.VideoWriter_fourcc(*'MP4V'),
int(cap.get(cv2.CAP_PROP_FPS)), (W, H))

# Store the track history
track_history = defaultdict(lambda: [])

# item store
track_history_items = defaultdict(lambda: [])

# Loop through the video frames
while cap.isOpened():
    # Read a frame from the video
    success, frame = cap.read()


    if success:
        # Run YOLOv8 tracking on the frame, persisting tracks between frames
        results = model.track(frame, persist=True, conf=0.7, iou=0.5)

        # Get the boxes and track IDs
        boxes = results[0].boxes.xywh.cpu()

        # Visualize the results on the frame
        annotated_frame = results[0].plot()

        #class name
        class_names = ['nuoc rua chen sunlight', 'banh oishi snack tom cay
nong', 'dau an good meal', 'nuoc tuong Huong Viet', 'nuoc nam De Nhi', 'but
bi xanh,', ' khan giay uot mamamy', 'coco tay te bao chet', 'mi goi kokomi
90g', 'duong am Cerave']

       if results[0].boxes.id is not None:
          track_ids = results[0].boxes.id.int().cpu().tolist()
          for track_id in track_ids:
              if track_id not in track_history:
                  cls = results[0].boxes.cls.int().cpu().tolist()
                  for i in range(len(cls)):
                      current_cls = cls[i]
                      current_track_id = track_id
                      current_item = (current_cls, current_track_id)

                      if current_item not in track_history_items:
```

```
                            store = track_history_items[current_item]  # Using
current_item as a key
                            class_name = class_names[current_cls]

                            # Check if the class name is not already in the
store
                            if class_name not in store:
                                store.append(class_name)

                                # Assuming you want to print the content of
track_history_items
                                for key, value in track_history_items.items():
                                    print(f"Key: {key}, Value: {value}")

                                # Assuming you want to post something with
current_cls, class_name, and the count of items in store
                                post(current_cls, class_name, len(store))


          # Plot the tracks
          for box, track_id in zip(boxes, track_ids):
              x, y, w, h = box
              track = track_history[track_id]
              track.append((float(x), float(y)))  # x, y center point
              if len(track) > 30:  # retain 90 tracks for 90 frames
                  track.pop(0)

              # Draw the tracking lines
              points = np.hstack(track).astype(np.int32).reshape((-1, 1, 2))
              cv2.polylines(annotated_frame, [points], isClosed=False,
color=(230, 230, 230), thickness=10)

          # Display the annotated frame
          out.write(annotated_frame)

          # Break the loop if 'q' is pressed
          if cv2.waitKey(1) & 0xFF == ord("q"):
              break
      else:
          # Break the loop if the end of the video is reached
          break

# Release the video capture object and close the display window
cap.release()
out.release()
cv2.destroyAllWindows()
```

2. Kiểm thử

a. Biểu đồ

Trong quá trình triển khai hệ thống AI nhận diện sản phẩm, việc đánh giá hiệu suất là một phần quan trọng để đảm bảo rằng mô hình đã được huấn luyện đáp ứng đúng nhu cầu và đạt được kết quả mong muốn. Dưới đây là một số biểu đồ thể hiện mức độ nhận diện của sản phẩm dựa trên các bộ dữ liệu kiểm thử và thực tế.



Hình 8. Ma trận đánh giá độ chính xác nhận diện sản phẩm

Ma trận trên là một bảng hiển thị kết quả dự đoán của một mô hình phân loại so với kết quả thực tế. Ma trận này được sử dụng để đánh giá hiệu suất nhận diện của một hệ thống.

Phần trung tâm của ma trận là các ô vuông với các giá trị biểu hiện mức độ chính xác của khả năng nhận diện. Trong trường hợp này, các nhãn dự đoán là các sản phẩm khác nhau.

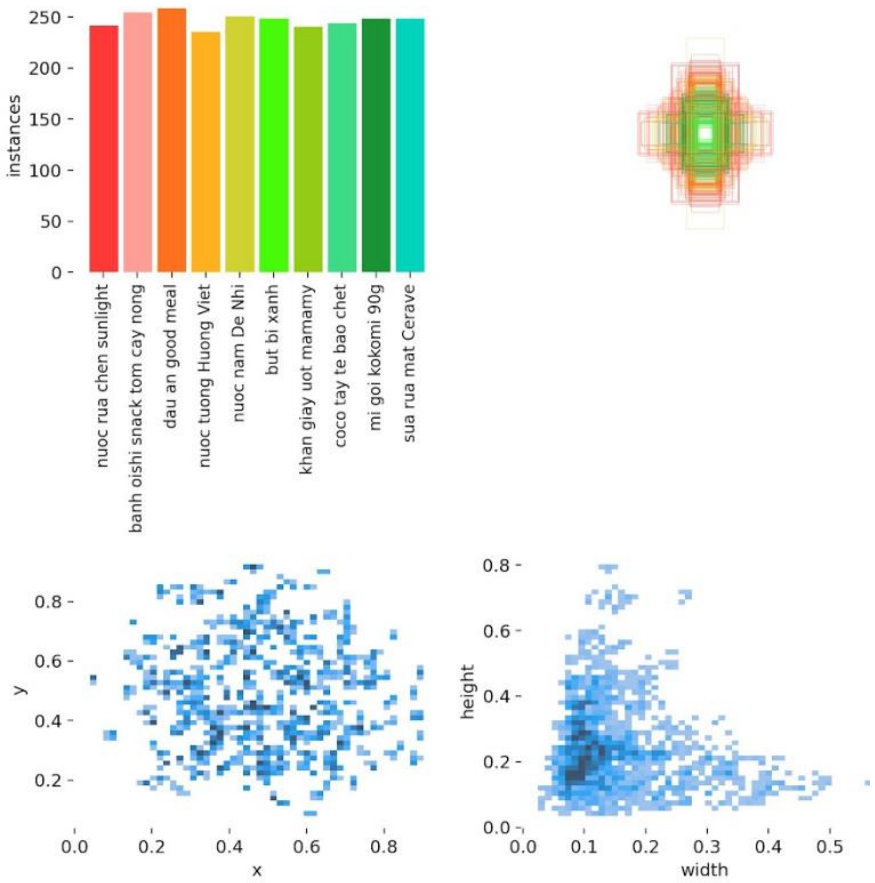
Dọc theo bên trái của ma trận là các nhãn dự đoán của các sản phẩm. Bên dưới là sản phẩm thực tế chính xác. Trong trường hợp này, các nhãn cũng là các sản phẩm khác nhau.

Các số ở giao điểm của các hàng và cột cho biết số lượng mẫu được dự đoán là thuộc loại đó nhưng thực sự thuộc loại khác.

Ví dụ, ở hàng thứ nhất, cột thứ ba, có số 0,1. Điều này có nghĩa là có 0,01 mẫu được dự đoán sai là "dầu ăn good meal" nhưng thực sự là "nước rửa chén sunlight" và số dự đoán đúng là 0.98.

Tổng quan, ma trận này cho thấy rằng mô hình đã có thể phân loại chính xác hầu hết các sản phẩm, tỉ lệ chính xác trung bình từ 98-99%. Tuy nhiên, vẫn còn một số lượng cực kì nhỏ sản phẩm bị phân loại sai (1%).

Các sản phẩm bị phân loại sai thường là các sản phẩm có độ tương đồng cao với các sản phẩm khác. Ví dụ, trong ma trận này, "nước rửa chén Sunlight" bị phân loại sai thành dầu ăn good meal" vì hai sản phẩm này có hình dạng và màu sắc gần giống như nhau.



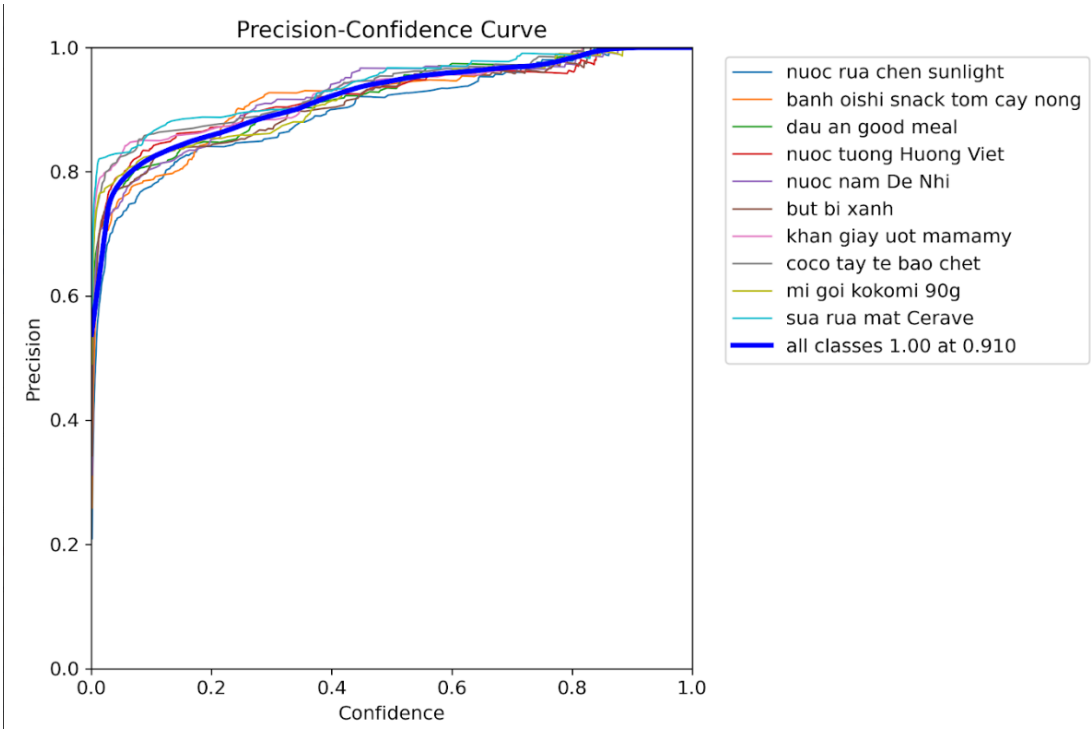
Hình 9. Phân bố khả năng nhận dạng sản phẩm của AI

Đây là biểu đồ thể hiện sự phân bố của các sản phẩm được AI nhận diện chính xác sau khi đào tạo. Biểu đồ cho thấy có sự phân bố khác nhau giữa các sản phẩm, đa số các sản phẩm có kích thước khoảng 0.25 về chiều cao và khoảng 0.4 về chiều rộng, trong khi đó có một vài

sản phẩm có kích thước lớn hơn, cao nhất là 0.8 và rộng nhất là 0.5.

Các sản phẩm có kích thước lớn bao gồm các sản phẩm được đánh dấu bởi các cột màu nóng như đỏ, cam,... các sản phẩm bao gồm "nước rửa chén sunlight", "bánh oishi snack tôm cay nồng", "dầu ăn good meal", "nước tương Hương Việt", "nước mắm Đệ Nhị". Trong khi đó các sản phẩm có kích thước nhỏ hơn sẽ được đánh dấu bởi cột màu lạnh như lục, lam,... các sản phẩm bao gồm "bút bi xanh", "khăn giấy ướt mamamy", "coco tẩy tế bào chết", "mì gói kokomi 90g", "sữa rửa mặt Cerave".

Nhìn chung, biểu đồ này cho thấy rằng AI đã có thể nhận diện chính xác hầu hết các sản phẩm sau khi đào tạo thông qua việc phân biệt cách thước của chúng.



Hình 10. Biểu đồ F1-Confidence Curve

Biểu đồ F1-Confidence Curve thể hiện mối quan hệ giữa F1 Score và Confidence Score của một mô hình.

F1 Score là một chỉ số đánh giá hiệu suất của một mô hình phân loại, trong khi Confidence Score là một chỉ số đánh giá độ tin cậy của kết quả phân loại của mô hình.

Trong biểu đồ này, trục X thể hiện Confidence Score, trong khi trục Y thể hiện F1 Score. Các điểm trên biểu đồ thể hiện kết quả phân loại của mô hình đối với các sản phẩm khác nhau.

Các điểm càng nằm ở phía trên đường cong F1-Confidence thể hiện các sản phẩm được

phân loại chính xác với độ tin cậy cao. Các điểm nằm ở phía dưới đường cong F1-Confidence tối ưu thể hiện các sản phẩm được phân loại chính xác với độ tin cậy thấp.

Nhìn chung, biểu đồ này cho thấy rằng mô hình đã có thể phân loại chính xác hầu hết các sản phẩm. Tuy ban đầu tỉ lệ chính xác chỉ nằm ở khoảng 60%, nhưng sau khi tiến hành train qua nhiều góc độ khác nhau thì tỉ lệ chính xác đã tăng lên gần như là 100%.

b. Thực nghiệm

Tôi sẽ tiến hành mô phỏng lại tình huống đi mua hàng ở siêu thị để kiểm tra khả năng nhận diện của thuật toán BoT-SORT và ByteTrack.

Tình huống: Mô phỏng lại quá trình mua hàng ở siêu thị và tiến hành đặt các sản phẩm lên bàn thanh toán.

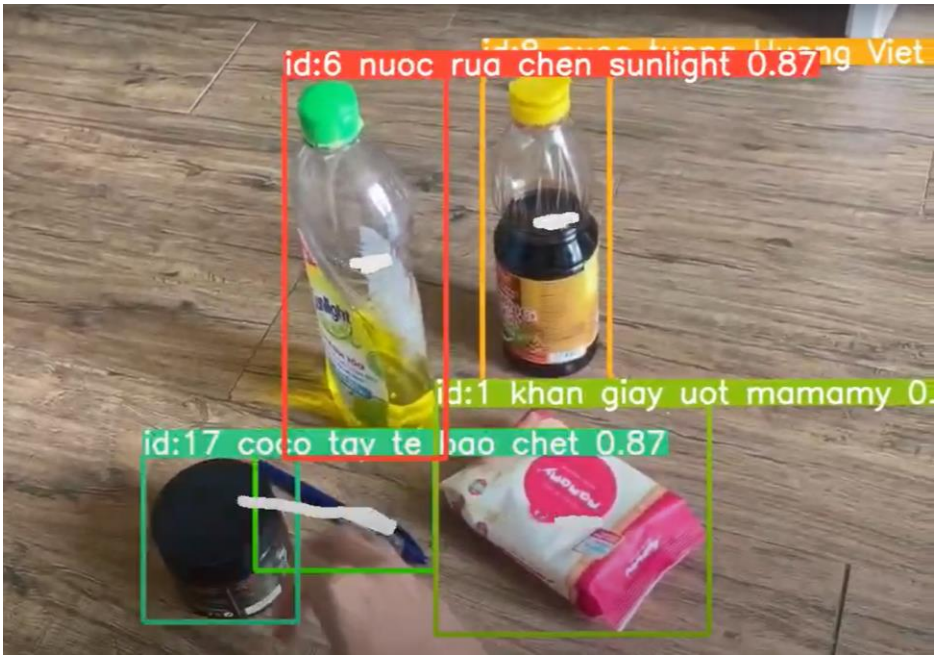
Dữ liệu đầu vào: Hình ảnh thu được từ camera, các sản phẩm mua bao gồm “nước rửa chén sunlight”, “nước tương Hương Việt”, “khăn giấy ướt mamamy”, “bút bi xanh”, “coco tẩy tế bào chết”.

Kết quả dự kiến: Cả hai thuật toán sẽ đều có khả năng nhận diện sản phẩm và có độ chính xác cao.

Mục đích: Kiểm tra và đánh giá hiệu suất hoạt động của hệ thống trí tuệ nhân tạo trong ngữ cảnh của quá trình mua sắm hàng ngày.

Hình 11. Hình ảnh camera thu được trong quá trình đặt hàng lên bàn thanh toán

Kết quả: Qua quá trình thực nghiệm có thể thấy cả hai thực toán đều có thể nhận diện được hàng hóa khi đặt từng món lên bàn, với độ chính xác cao, hầu hết đều từ 87%-98% và sự chênh lệch giữa hai thuộc toán là không đáng kể.

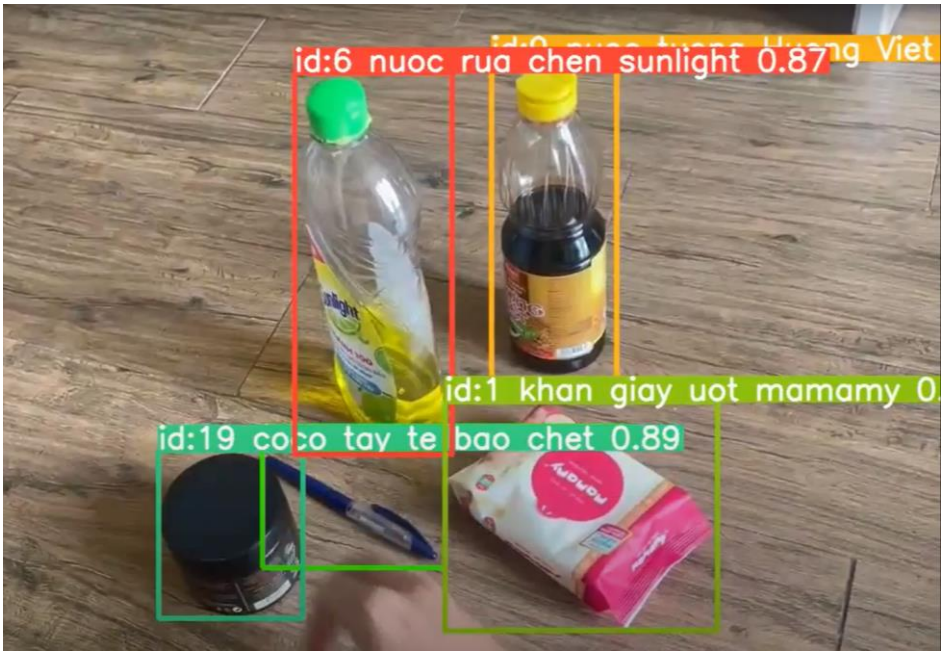


Hình 12. Mô phỏng thuật toán BoT-SORT

Dưới đây là log của kết quả chạy thuật toán BoT-SORT:

```
0: 384x640 (no detections), 107.9ms
Speed: 2.7ms preprocess, 107.9ms inference, 1.1ms postprocess per image at
shape (1, 3, 384, 640)

0: 384x640 (no detections), 51.3ms
Speed: 1.6ms preprocess, 51.3ms inference, 0.8ms postprocess per image at
shape (1, 3, 384, 640)

0: 384x640 (no detections), 48.5ms
Speed: 2.4ms preprocess, 48.5ms inference, 0.9ms postprocess per image at
shape (1, 3, 384, 640)

0: 384x640 (no detections), 48.6ms
Speed: 2.5ms preprocess, 48.6ms inference, 0.8ms postprocess per image at
shape (1, 3, 384, 640)

0: 384x640 (no detections), 34.7ms
Speed: 2.3ms preprocess, 34.7ms inference, 1.4ms postprocess per image at
shape (1, 3, 384, 640)

0: 384x640 (no detections), 35.3ms
Speed: 2.5ms preprocess, 35.3ms inference, 0.8ms postprocess per image at
shape (1, 3, 384, 640)

0: 384x640 (no detections), 34.8ms
```

```
Speed: 2.6ms preprocess, 34.8ms inference, 0.9ms postprocess per image at
shape (1, 3, 384, 640)

.....
0: 384x640 1 nuoc rua chen sunlight, 1 nuoc tuong Huong Viet, 1 but bi xanh,
1 khan giay uot mamamy, 1 coco tay te bao chet, 35.1ms
Speed: 2.6ms preprocess, 35.1ms inference, 2.0ms postprocess per image at
shape (1, 3, 384, 640)
```



Hình 1. Mô phỏng thuật toán ByteTrack

Dưới đây là log của kết quả chạy thuật toán ByteTrack:

```
0: 384x640 (no detections), 2714.6ms
Speed: 2.7ms preprocess, 2714.6ms inference, 3.4ms postprocess per image at
shape (1, 3, 384, 640)

0: 384x640 (no detections), 2838.3ms
Speed: 2.3ms preprocess, 2838.3ms inference, 5.9ms postprocess per image at
shape (1, 3, 384, 640)

0: 384x640 (no detections), 4337.6ms
Speed: 3.7ms preprocess, 4337.6ms inference, 7.1ms postprocess per image at
shape (1, 3, 384, 640)

0: 384x640 (no detections), 2634.9ms
Speed: 2.6ms preprocess, 2634.9ms inference, 3.5ms postprocess per image at
shape (1, 3, 384, 640)

.....
0: 384x640 1 nuoc rua chen sunlight, 1 nuoc tuong Huong Viet, 1 but bi xanh,
1 khan giay uot mamamy, 1 coco tay te bao chet, 2780.7ms
Speed: 4.7ms preprocess, 2780.7ms inference, 7.0ms postprocess per image at
shape (1, 3, 384, 640)
```

PHẦN V. KẾT LUẬN

1. Kết luận

Trong quá trình nghiên cứu và triển khai ứng dụng AI vào nhận diện sản phẩm, chúng tôi đã thu được những kết quả đáng kể, điều này thể hiện rõ sự tiềm năng và ưu việt của công nghệ này trong việc tối ưu hóa quy trình nhận diện và quản lý sản phẩm. BoT-SORT, với khả năng theo dõi hiệu quả, đặc biệt là trong các môi trường đa dạng, đem lại sự đơn giản và linh hoạt trong quá trình theo dõi đối tượng. Sự ổn định và giảm thiểu sai số trong quá trình theo dõi làm cho BoT-SORT trở thành một lựa chọn hiệu quả cho các hệ thống xử lý video và giám sát an ninh. Utralytics, một thư viện và framework đa nhiệm đa dạng, mở rộng khả năng ứng dụng AI từ nhận diện đối tượng đến phân loại và theo dõi đối tượng đã giúp cho sản phẩm trở trên hoàn thiện hơn. Hệ thống AI nhận diện sản phẩm của chúng tôi đã hoạt động rất tốt và cho ra độ chính xác cực kì cao, đem lại nhiều ưu điểm, bên cạnh đó vẫn còn tồn tại một vài hạn chế của sản phẩm.

a. Ưu điểm

- Độ chính xác cao: Các sản phẩm đều được nhận diện và chính xác cao nhất lên đến 98%.

- Hiệu suất cao: Nhận diện ngay lập tức khi có sản phẩm được đặt vào khung hình

- Dễ dàng tích hợp: Hệ thống AI có thể dễ dàng tích hợp vào các hệ thống quản lý sản phẩm hiện tại, mang lại sự linh hoạt và thuận tiện trong việc triển khai.

b. Nhược điểm

- Chưa tích hợp được cân khối lượng.

- Giao diện thanh toán vẫn còn khá thô sơ

- Yêu cầu dữ liệu lớn: Để đạt được hiệu suất tốt, hệ thống AI yêu cầu một lượng lớn dữ liệu đa dạng để huấn luyện, dữ liệu của nhóm vẫn còn ít so với một cửa hàng.

2. Hướng phát triển

Dựa trên những kết quả tích cực và nhận định được những hạn chế cần được vượt qua, chúng tôi sẽ đề ra một số hướng phát triển để nâng cao và hoàn thiện hệ thống AI nhận diện sản phẩm:

- Tích hợp cân khối lượng: Một trong những hướng phát triển quan trọng là tích hợp

khả năng đo lường cân khối lượng của sản phẩm. Điều này sẽ giúp cải thiện tính toàn vẹn của quy trình nhận diện, đồng thời hỗ trợ trong việc quản lý hàng hóa và dự báo tồn kho.

- Tối ưu hóa giao diện thanh toán: Đặc biệt là cần tập trung vào việc phát triển một giao diện thanh toán thân thiện với người dùng, giúp tối ưu hóa trải nghiệm của khách hàng trong quá trình mua sắm. Điều này có thể bao gồm cải thiện tính tương tác và tích hợp các phương thức thanh toán tiện lợi.

- Mở rộng cơ sở dữ liệu: Tăng cường việc thu thập và sử dụng dữ liệu huấn luyện là yếu tố quyết định cho độ chính xác của hệ thống AI. Việc mở rộng và đa dạng hóa nguồn dữ liệu sẽ giúp cải thiện hiệu suất của mô hình.

- Nghiên cứu và áp dụng công nghệ mới: Liên tục theo dõi và áp dụng các tiến bộ trong lĩnh vực AI để nâng cao hiệu suất và tính ứng dụng của hệ thống. Công nghệ mới như học máy tăng cường hoặc xử lý ngôn ngữ tự nhiên có thể được khám phá để mở rộng khả năng của sản phẩm.

- Hợp tác và đối tác: Tìm kiếm cơ hội hợp tác với các đối tác và nhà cung cấp dịch vụ chuyên nghiệp trong lĩnh vực AI để nâng cao chất lượng và tính đồng nhất của sản phẩm.

Chúng tôi tin rằng việc triển khai những hướng phát triển này sẽ giúp sản phẩm không chỉ đáp ứng được những yêu cầu hiện tại mà còn tạo ra sự đột phá và sẵn sàng cho thách thức của tương lai trong lĩnh vực nhận diện sản phẩm sử dụng trí tuệ nhân tạo.

PHẦN VI: TÀI LIỆU THAM KHẢO

[1] VNPay: https://sandbox.vnpayment.vn/apis/docs/huong-dan-tich-hop/

[2] https://www.hackster.io/coderscafe/autobill-042d29#overview

[3] https://docs.ultralytics.com/reference/engine/results/#ultralytics.engine.results.Boxes

[4] https://docs.ultralytics.com/modes/predict/#boxes