BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC QUẢN LÝ VÀ CÔNG NGHỆ HẢI PHÒNG

ĐỒ ÁN TỐT NGHIỆP

NGÀNH: CÔNG NGHỆ THÔNG TIN

Sinh viên : Hoàng Đình Hiệp

Giảng viên hướng dẫn: TS. Hồ Thị Hương Thơm

HẢI PHÒNG - 2025

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC QUẢN LÝ VÀ CÔNG NGHỆ HẢI PHÒNG

XÂY DỰNG MÔ HÌNH PHÁT HIỆN NGƯỜI LÁI XE Ô TÔ NGỦ GẬT KHI ĐANG LÁI XE

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

NGÀNH: CÔNG NGHỆ THÔNG TIN

Sinh viên : Hoàng Đình Hiệp

Giảng viên hướng dẫn: TS. Hồ Thị Hương Thơm

HẢI PHÒNG – 2025

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC QUẢN LÝ VÀ CÔNG NGHỆ HẢI PHÒNG

NHIỆM VỤ ĐỀ TÀI TỐT NGHIỆP

Sinh viên: **Hoàng Đình Hiệp** **Mã SV**: **2112111022**

Lớp : CT2501C

Ngành : Công nghệ thông tin

Tên đề tài: ***"Xây dựng mô hình phát hiện người lái xe ô tô ngủ gật khi đang lái xe"***

NHIỆM VỤ ĐỀ TÀI

1. Nội dung và các yêu cầu cần giải quyết trong nhiệm vụ đề tài tốt nghiệp

a. Mô tả tóm tắt đề tài

- Mô hình hệ thống AI được thiết kế để phát hiện người lái xe ô tô có hành vi ngủ gật khi đang lái xe
- Cung cấp khả năng phát hiện đối tượng theo thời gian thực với độ chính xác cao,giúp đưa ra cảnh báo kịp thời đảm bảo an toàn khi tham gia giao thông

2. Nội dung hướng dẫn

- Nghiên cứu lý thuyết nhận dạng và phân loại đối tượng.
- Vận dụng lý thuyết xây dụng mô hình AI để phát hiện hành vi ngủ gật khi đang lái xe dẫn đến không đảm bảo an toàn khi lưu thông trên đường.
- Thử nghiệm và đánh giá mô hình

3. Kết quả cần đạt được

Xây dụng mô hình AI để phát hiện người lái xe ô tô có hành vi ngủ gật khi đang lái xe ô tô để đưa ra cảnh báo kịp thời đảm bảo an toàn giao thông

4. Địa điểm thực tập tốt nghiệp

Trường Đại học Quản lý và Công nghệ Hải Phòng

CÁN BỘ HƯỚNG DẪN ĐỀ TÀI TỐT NGHIỆP

Họ và tên : Hồ Thị Hương Thơm

Học hàm, học vị : Tiến sĩ

Cơ quan công tác : Trường Đại Học Hàng Hải Việt Nam

Nội dung hướng dẫn:

- Tìm hiểu về bài toán phát hiện đối tượng nói chung và phát hiện phát hiện đối tượng bằng YOLO11.
- Phân tích, thu thập dữ liệu hình ảnh người lái xe ngủ gật để huấn luyện phát hiện đối tượng.
- Cài đặt mô hình phát hiện đối tượng bằng YOLO11
- Thử nghiệm mô hình trên hình ảnh và video để phát hiện người lái xe ô tô ngủ gật khi tham gia giao thông
- Nhận xét đánh giá, kết luận và đưa ra khuyến nghị khi ứng dụng.

Đề tài tốt nghiệp được giao ngày 21 tháng 08 năm 2025

Yêu cầu phải hoàn thành xong trước ngày 20 tháng 11 năm 2025

Đã nhận nhiệm vụ ĐTTN	Đã giao nhiệm vụ ĐTTN
Sinh viên	Giảng viên hướng dẫn
Hoàng Đình Hiệp	Ts.Hồ Thị Hương Thơm

Hải Phòng, ngày tháng năm 2025

TRƯỞNG KHOA

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc

PHIẾU NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN TỐT NGHIỆP

Họ và tên giảng viên: Hồ Thị Hương Thơm

Đơn vị công tác: Trường Đại Học Hàng Hải Việt Nam

Họ và tên sinh viên : Hoàng Đình Hiệp

Ngành: Công Nghệ Thông Tin

1. Tinh thần thái độ của sinh viên trong quá trình làm đề tài tốt nghiệp

……
……
……
……

2. Đánh giá chất lượng của đồ án/khóa luận (so với nội dung yêu cầu đã đề ra trong nhiệm vụ Đ.T.T.N, trên các mặt lý luận, thực tiễn, tính toán số liệu...)

……
……
……
……

3. Ý kiến của giảng viên hướng dẫn tốt nghiệp

Được bảo vệ ☐ Không được bảo vệ ☐ Điểm hướng dẫn ☐

Hải Phòng, ngày .... tháng .... năm 2025

Giảng viên hướng dẫn

(*ký và ghi rõ họ tên)*

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc

PHIẾU NHẬN XÉT CỦA GIẢNG VIÊN CHẤM PHẢN BIỆN

Họ và tên giảng viên:……

Đơn vị công tác:……

Họ và tên sinh viên: …….. …..**Ngành**: ……

Đề tài tốt nghiệp: …….

……

1. Phần nhận xét của giảng viên chấm phản biện

……

……

……

……

2. Những mặt còn hạn chế

……

……

……

3. Ý kiến của giảng viên chấm phản biện

Được bảo vệ ☐ Không được bảo vệ ☐ Điểm phản biện ☐

Hải Phòng, ngày.....tháng ..... năm 2025

Giảng viên chấm phản biện

(*ký và ghi rõ họ tên*)

LỜI CẢM ƠN

Đồ án tốt nghiệp này được hoàn thành tại Trường Đại học Quản lý và Công nghệ Hải Phòng. Trong suốt quá trình học tập và thực hiện đề tài "Xây dựng mô hình phát hiện người lái xe oto ngủ gật khi đang lái xe ", em đã nhận được rất nhiều sự quan tâm, chỉ bảo và hỗ trợ quý báu từ các thầy cô, bạn bè và nhà trường.

Trước hết, em xin bày tỏ lòng biết ơn sâu sắc tới TS. Hồ Thị Hương Thơm, người đã tận tình hướng dẫn, định hướng, truyền đạt những kiến thức và kinh nghiệm quý báu, giúp em hoàn thiện đồ án này một cách tốt nhất.

Em cũng xin gửi lời cảm ơn chân thành đến Ban Giám hiệu cùng toàn thể giảng viên Trường Đại học Quản lý và Công nghệ Hải Phòng, đặc biệt là các thầy cô trong khoa Công nghệ Thông tin đã luôn tạo điều kiện thuận lợi trong suốt thời gian học tập và nghiên cứu.

Cuối cùng, em xin cảm ơn các bạn sinh viên lớp Công nghệ Thông tin khóa 2021 – 2025 đã luôn động viên, chia sẻ và giúp đỡ em trong quá trình thực hiện đề tài.

Trân trọng

Hải Phòng,ngày tháng năm 2025

Người viết

Hoàng Đình Hiệp

LỜI CAM ĐOAN

Sinh viên xin cam đoan đề tài "“ Xây dụng mô hình phát hiện người lái xe ô tô ngủ gật khi đang lái xe" là công trình nghiên cứu độc lập dưới sự hướng dẫn của TS. Hồ Thị Hương Thơm – Giảng viên Khoa Công nghệ thông tin Trường Đại Học Hàng Hải Việt Nam. Đề tài là sự cố gắng, quyết tâm của cá nhân sinh viên trong việc tiếp cận kiến thức chuyên môn mới để hoàn thành đạt mục tiêu của đề tài. Các dữ liệu đề nghiên cứu thực nghiệm là trung thức có nguồn gốc trích dẫn rõ ràng.

Sinh viên xin hoàn toàn chịu trách nghiệm với lời cam đoan này

Người cam đoan

Hoàng Đình Hiệp

MỤC LỤC

DANH SÁCH ẢNH

DANH SÁCH BẢNG

DANH SÁCH BẢNG CÁC TỪ VIẾT TẮT

STT	Từ viết tắt	Tên đầy đủ (English)	Ý nghĩa (Tiếng Việt)
1	AI	Artificial Intelligence	Trí tuệ nhân tạo
2	CV	Computer Vision	Thị giác máy tính
3	YOLO	You Only Look Once	Mô hình phát hiện đối tượng thời gian thực
4	SPPF	Spatial Pyramid Pooling – Fast	Khối trích xuất đa tỉ lệ cải tiến, tăng tốc so với SPP
5	C2PSA	C2-Parallel Spatial Attention	Khối chú ý không gian song song giúp tăng độ chính xác
6	C3k2	Cross-Stage Partial Conv Block (kernel 2)	Khối tích chập CSP cải tiến dùng kernel nhỏ giúp giảm chi phí tính toán

7	Rep / Rep Block	Re-parameterization Block	Khối tái tham số hóa giúp tăng tốc suy luận
8	Conv	Convolution	Tích chập – lớp cơ bản trong mạng CNN
9	UpSample	Upsampling	Tăng kích thước đặc trưng trong Neck
10	Concat	Concatenate	Nối đặc trưng theo chiều kênh
11	NMS	Non-Maximum Suppression	Lọc hộp dự đoán trùng lặp
12	GPU	Graphics Processing Unit	Bộ xử lý đồ họa
13	TPU	Tensor Processing Unit	Bộ xử lý tăng tốc AI của Google
14	EAR	Eye Aspect Ratio	Chỉ số hình học để đo mức độ mở của mắt
15	PERCLOS	Percentage of Eye Closure	Tỷ lệ thời gian mắt nhắm – dấu hiệu buồn ngủ

16	LSTM	Long Short-Term Memory	Mạng ghi nhớ chuỗi thời gian
17	mAP	Mean Average Precision	Độ chính xác trung bình
18	AP	Average Precision	Độ chính xác trung bình trên từng lớp
19	P	Precision	Độ chính xác dự đoán đúng trên số dự đoán
20	R	Recall	Tỷ lệ dự đoán đúng trên tổng số đối tượng thật
21	IDE	Integrated Development Environment	Môi trường lập trình tích hợp
22	VSC / VS Code	Visual Studio Code	Công cụ soạn thảo và lập trình
23	CPU	Central Processing Unit	Bộ xử lý trung tâm
24	pt	PyTorch Model File	Định dạng mô hình của PyTorch

25	TFLite	TensorFlow Lite	Định dạng mô hình dùng cho thiết bị di động
26	FPN	Feature Pyramid Network	Mạng kim tự tháp đặc trưng
27	PAN	Path Aggregation Network	Mạng tổng hợp đặc trưng
28	IoU	Intersection over Union	Mức độ trùng khớp giữa hộp dự đoán và hộp thật
29	ImgSz	Image Size	Kích thước ảnh khi huấn luyện

CHƯƠNG 1. TỔNG QUAN VỀ ĐỀ TÀI

Chương 1 giới thiệu vấn đề tai nạn giao thông do tài xế buồn ngủ và sự cần thiết của hệ thống giám sát bằng trí tuệ nhân tạo. Các kiến thức nền tảng về phát hiện đối tượng, những phương pháp phổ biến và ưu – nhược điểm của từng kỹ thuật được trình bày nhằm làm rõ cơ sở lựa chọn mô hình YOLO. Bên cạnh đó, chương cũng phân tích bài toán phát hiện tài xế ngủ gật và các chỉ số đánh giá độ tin cậy như Precision, Recall, AP và mAP. Mục tiêu là xây dựng nền tảng lý thuyết cho mô hình phát hiện buồn ngủ trong thời gian thực.

1.1. Đặt vấn đề

Trong những năm gần đây, tai nạn giao thông do người lái xe buồn ngủ ngày càng gia tăng và trở thành một trong những nguyên nhân hàng đầu gây thiệt hại về người và của. Đặc biệt, với các tài xế đường dài, tình trạng mệt mỏi, mất tập trung và ngủ gật khi điều khiển phương tiện tiềm ẩn nguy cơ vô cùng lớn.

Trên thế giới, nhiều giải pháp đã được nghiên cứu nhằm hỗ trợ giám sát tình trạng tỉnh táo của tài xế, từ cảm biến sinh học (nhịp tim, sóng não) cho đến hệ thống camera giám sát hành vi. Trong đó, việc ứng dụng trí tuệ nhân tạo [1] và thị giác máy tính [2] để tự động phát hiện dấu hiệu buồn ngủ từ hình ảnh/video khuôn mặt, mắt và tư thế lái xe đang được đánh giá là hướng đi hiệu quả, khả thi và ít xâm lấn.

Xuất phát từ thực tiễn này, đề tài "Xây dựng mô hình phát hiện người lái xe ô tô ngủ gật khi đang lái xe" được lựa chọn với kỳ vọng góp phần nâng cao an toàn giao thông và hỗ trợ nghiên cứu trong lĩnh vực thị giác máy tính [2].Trong đề tài này, sẽ tập trung nghiên cứu và áp dụng YOLO [9]11 một trong những phiên bản tiên tiến nhất của họ mô hình YOLO [9] (You Only Look Once) để phát hiện và cảnh báo tình trạng buồn ngủ, ngủ gật của người lái xe. YOLO11 [9] có ưu điểm vượt trội về tốc độ xử lý real-time và khả năng nhận diện các chi tiết nhỏ như trạng thái mắt nhắm, mắt mở, cử động đầu , từ đó đưa ra cảnh báo kịp thời nhằm giảm thiểu rủi ro tai nạn giao thông.

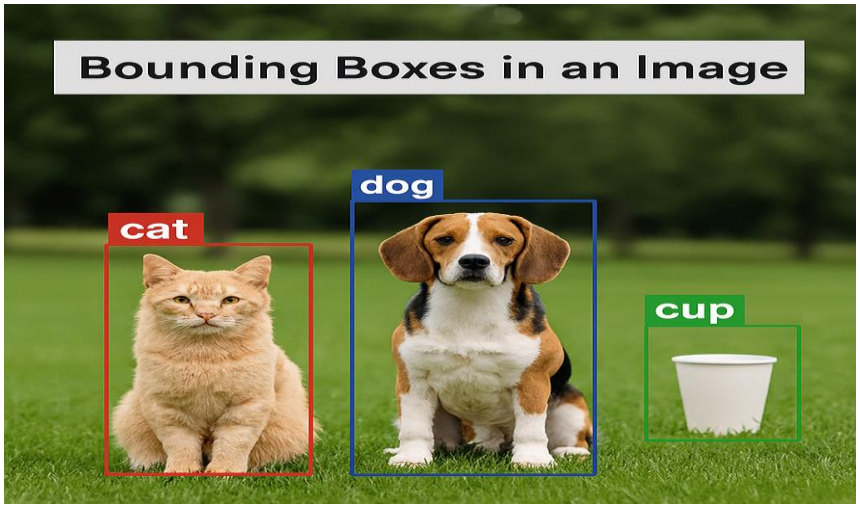
Hình 1.1. hình ảnh tài xế ngủ gật khi đang lái xe

1.2. Phát hiện đối tượng

1.2.1. Khái niệm

Phát hiện đối tượng là một lĩnh vực vô cùng quan trọng trong lĩnh vực xử lý ảnh và thị giác máy tính [2]. Phát hiện đối tượng liên quan đến việc xác định trường hợp cụ thể của đối tượng trong hình ảnh hoặc video. Các mô hình phát hiện đối tượng thường được dùng để xác định đối tượng trên hình ảnh hoặc video camera trong thời gian thực.

Phát hiện đối tượng đóng vai trò quan trọng trong học sâu và xử lý ảnh để phát triển các mô hình tính toán các thông tin cần thiết cho các ứng dụng phần mềm sử dụng thị giác máy tính [2].



Hình 1.2. hình ảnh hộp phát hiện đối tượng

Phát hiện đối tượng là quá trình xác định và định vị các đối tượng có trong hình ảnh hoặc video. Khác với phân loại ảnh chỉ trả lời câu hỏi "ảnh thuộc lớp nào", phát hiện đối tượng vừa cho biết loại đối tượng, vừa chỉ ra vị trí tọa độ của đối tượng đó.

1.2.2. Cách phát hiện đối tượng

Để phát hiện đối tượng chúng ta có thể sử dụng cách phát hiện truyền thống hoặc sử dụng theo học sâu hiện đại. Mỗi phương pháp đều có ưu điểm và nhược điểm riêng nên mỗi cách phát hiện sẽ phù hợp với từng yêu cầu và điều kiện khác nhau.

Xử lý ảnh theo phương pháp truyền thống

Kỹ thuật xử lý ảnh truyền thống không yêu cầu về dữ liệu để đào tạo và tự giám sát, OpenCV là một ví dụ điển hình cho phương pháp truyền thống

Ưu điểm: Không yêu cầu dữ liệu được gán nhãn thủ công, giúp chúng ta giảm bớt số lượng công việc cần phải thực hiện và chi phí liên quan đến đào tạo dữ liệu

Nhược điểm:

- Hạn chế trong những tình huống phức tạp như điều kiện ánh sáng, đối tượng bị che mất một nửa, ảnh bị mờ không rõ nét...v.v
- Độ chính xác và độ tin cậy thấp
- Xử lý ảnh theo phương pháp học sâu hiện đại

Các phương pháp học sâu hiện đại đa số đều phụ thuộc vào học có giám sát, các phương pháp học có giám sát là tiêu chuẩn trong các nhiệm vụ thực hiện thị giác máy tính [2]. Tuy nhiên phương pháp này bị hạn chế bởi sự giới hạn sức mạnh tính toán của GPU

Ưu điểm:

- Phát hiện đối tượng mạnh mẽ hơn so với phương pháp phát hiện truyền thống với các đối tượng bị che khuất, thiếu ánh sáng, bị mờ..v.v..
- Độ chính xác cao hơn so với các phương pháp truyền thống

Nhược điểm:

- Tốn lượng lớn dữ liệu dùng để đào tạo
- Cần nhiều sức mạnh của GPU để xử lý và đào tạo mô hình

1.2.3 Các phương pháp phát hiện phổ biến hiện nay.

1.2.3.1. Phương pháp mô tả đặc trưng

Phương pháp mô tả đặc trưng (Histogram of Oriented Gradient – HOG) được xem là một trong những kỹ thuật phát hiện đối tượng cổ điển và được sử dụng từ khá sớm. Ý tưởng ban đầu được giới thiệu vào năm 1986, tuy nhiên phải đến năm 2005 phương pháp này mới thực sự phổ biến và được ứng dụng rộng rãi trong các bài toán thị giác máy tính [2]. HOG đóng vai trò là một công cụ trích xuất đặc trưng, hỗ trợ nhận dạng và xác định vị trí đối tượng trong hình ảnh.

Nguyên lý chính của HOG dựa trên việc phân tích sự phân bố của gradient hoặc hướng biên trong ảnh để mô tả các đặc trưng cục bộ. Cách tiếp cận này được thực hiện bằng cách chia ảnh thành nhiều ô nhỏ (cell). Với mỗi cell, hệ thống sẽ tính toán histogram của các hướng gradient tại những điểm ảnh trong cell đó. Sau đó, các histogram từ nhiều cell sẽ được kết hợp lại để hình thành một mô tả tổng quát cho toàn bộ bức ảnh.

Để tăng độ chính xác, các histogram thường được chuẩn hóa trong các khối (block), giúp giảm ảnh hưởng của sự thay đổi ánh sáng hoặc độ tương phản. Nhờ cơ chế này, HOG đạt hiệu quả tốt trong các tác vụ nhận dạng đối tượng, đặc biệt là những đối tượng có biên và hình dạng rõ ràng.

Trước khi tìm hiểu chi tiết về cấu trúc tổng thể của HOG, chúng ta hãy xem cơ chế hoạt động của nó. Đối với một điểm ảnh cụ thể trong hình ảnh, biểu đồ tần suất của gradient được xác định bằng cách xét các giá trị theo chiều dọc và ngang để hình thành nên các vector đặc trưng. Nhờ vào độ lớn gradient và góc gradient, ta có thể xác định rõ giá trị tại điểm ảnh hiện tại thông qua việc phát hiện những đặc trưng khác trong khu vực xung quanh theo cả chiều ngang lẫn chiều dọc.

Dựa trên biểu diễn hình ảnh này, chúng ta sẽ phân tích một phần nhỏ của hình ảnh có kích thước xác định. Trước tiên, gradient được tính toán bằng cách chia toàn bộ hình ảnh thành các ô nhỏ 8×8 để tạo ra các biểu diễn gradient. Với 64 vector gradient thu được, ta tiếp tục chia nhỏ mỗi ô theo góc và tính toán biểu đồ theo diện tích ô. Quá trình này giúp giảm số lượng 64 vector ban đầu xuống còn 9 giá trị đại diện.

Khi đã có được các giá trị biểu đồ gồm 9 ô cho mỗi phần tử, ta có thể kết hợp chồng lấn để hình thành các khối. Bước cuối cùng là tạo ra các khối đặc trưng, chuẩn hóa các vector đặc trưng thu được, và tổng hợp tất cả để hình thành đặc trưng HOG cuối cùng.

1.2.3.2. Mạng nơ-ron

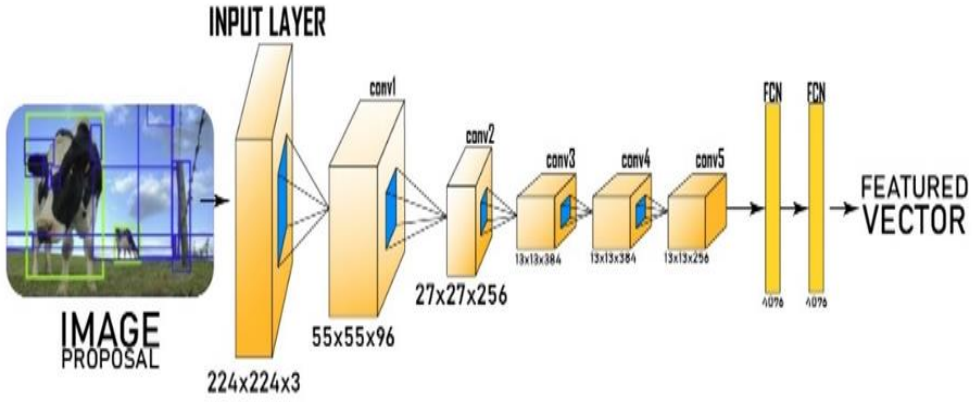
Mạng nơ-ron tích chập theo vùng (R-CNN)

Trong quá trình phát hiện đối tượng, mạng nơ-ron tích chập dựa trên vùng (R-CNN) được xem như một cải tiến quan trọng so với các phương pháp truyền thống như HOG hay SIFT. Thay vì khai thác toàn bộ đặc trưng của hình ảnh, R-CNN tập trung vào việc trích xuất các đặc trưng quan trọng nhất (thường khoảng 2000 đặc trưng) thông qua việc sử dụng các đặc trưng được lựa chọn. Quá trình này được hỗ trợ bởi các thuật toán tìm kiếm chọn lọc, nhằm đưa ra những vùng đề xuất có khả năng chứa đối tượng cao hơn.

Cụ thể, thuật toán tìm kiếm chọn lọc sẽ tạo ra nhiều phân đoạn nhỏ trong một hình ảnh và chọn các vùng ứng cử viên tiềm năng. Sau đó, một thuật toán tham lam được sử dụng để kết hợp các vùng nhỏ này thành các phân đoạn lớn

hơn, nhằm giảm sự trùng lặp và giữ lại những vùng quan trọng nhất phục vụ cho bước xử lý tiếp theo.

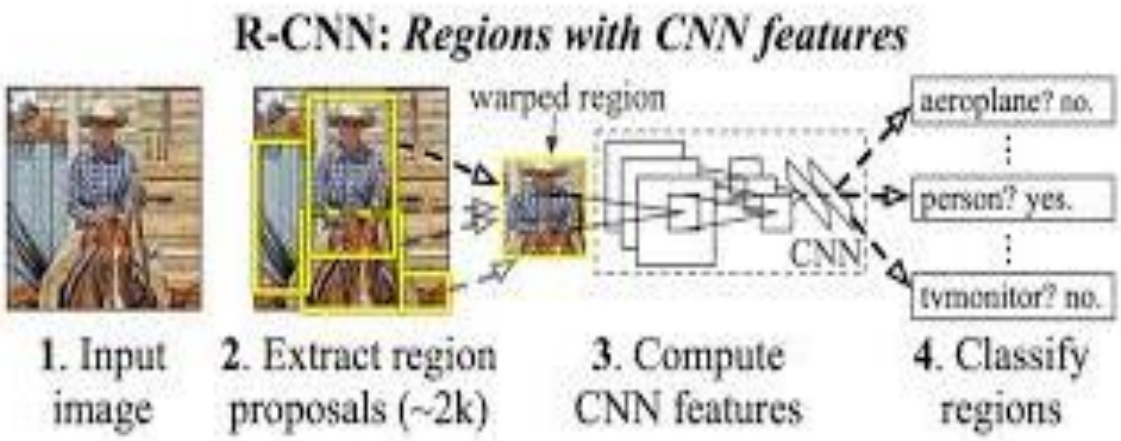
Ở giai đoạn cuối, R-CNN tiến hành phân loại các vùng được đề xuất để dự đoán nhãn đối tượng tương ứng và gắn hộp giới hạn bao quanh. Mỗi dự đoán được tính toán dựa trên một mô hình phân loại riêng, đồng thời sử dụng thêm một mô hình hồi quy để tinh chỉnh hộp giới hạn, giúp tăng độ chính xác của việc xác định vị trí đối tượng trong hình ảnh.



Hình 1.3. ảnh mô tả hoạt động của R-CNN

a. Faster R-CNN

Faster R-CNN là một bước tiến vượt bậc trong chuỗi cải tiến từ R-CNN sang Fast R-CNN sang Faster R-CNN. Mục tiêu chính của nó là tăng tốc độ và độ chính xác trong phát hiện đối tượng bằng cách tích hợp trực tiếp quá trình đề xuất vùng vào mạng nơ-ron, thay vì dựa vào các thuật toán bên ngoài như Selective Search.



Hình 1.4 Ảnh miêu tả phương pháp phát hiện đối tượng bằng Faster R-CNN

b. YOLO

YOLO [9] (You Only Look Once) là một trong những phương pháp phát hiện đối tượng được sử dụng rộng rãi trong lĩnh vực thị giác máy tính [2]. Thuật toán này dựa trên mạng nơ-ron tích chập (CNN) để xử lý dữ liệu hình ảnh cũng như video.

Cơ chế hoạt động của YOLO [9] là chia hình ảnh đầu vào thành một lưới các ô, trong đó mỗi ô có nhiệm vụ phát hiện và nhận diện đối tượng xuất hiện bên trong. Mô hình được đánh giá cao nhờ tốc độ xử lý nhanh và độ chính xác tốt. Kể từ khi Joseph Redmon và cộng sự giới thiệu lần đầu vào năm 2016, YOLO [9] đã trải qua nhiều phiên bản cải tiến, với phiên bản gần đây nhất là YOLO [9]11.

Tuy nhiên, một hạn chế đáng chú ý của YOLO [9] là khả năng nhận diện các đối tượng có kích thước nhỏ còn hạn chế, đồng thời độ chính xác cũng bị ảnh hưởng khi đối tượng xuất hiện trong bối cảnh phức tạp, đông đúc hoặc ở khoảng cách xa so với camera.

Một số mô hình YOLO [9] nổi bật:

- YOLO1 [9]: Phiên bản đầu tiên của YOLO [9], được Joseph Redmon giới thiệu năm 2016. Đây là bước đột phá khi lần đầu tiên đưa ra ý

tưởng xử lý toàn bộ hình ảnh chỉ trong một lần (You Only Look Once), giúp phát hiện đối tượng nhanh hơn so với các phương pháp trước đó.

- YOLO 2 [9] (YOLO [9]9000): Ra mắt năm 2017 bởi Joseph Redmon, được cải tiến để nhận diện hơn 9000 lớp đối tượng khác nhau. Phiên bản này kết hợp giữa phát hiện và phân loại đối tượng, nâng cao độ chính xác so với YOLO1 .

- YOLO v3 [9]: Công bố vào năm 2018, cũng bởi Joseph Redmon, với khả năng phát hiện đối tượng đa cấp độ nhờ mạng darknet-53. YOLO [9]3 nổi bật với hiệu quả cao trong phát hiện đối tượng thời gian thực.

- YOLO 4 [9]: Được phát triển bởi Alexey Bochkovskiy và phát hành vào năm 2020. Đây là phiên bản kế thừa YOLOv3 với nhiều cải tiến quan trọng trong huấn luyện và tối ưu hóa, mang lại tốc độ xử lý nhanh và độ chính xác cao.

- YOLO 5 [9]: Ra mắt vào giữa năm 2020 bởi Ultralytics. YOLO 5 không phải do nhóm tác giả gốc phát triển nhưng nhanh chóng trở thành một trong những phiên bản phổ biến nhất nhờ tính tiện dụng, tốc độ cao và dễ triển khai trên nhiều nền tảng.

- YOLO 6 [9]: Được Meituan phát hành năm 2022, tập trung vào hiệu năng trong ứng dụng công nghiệp và robot giao hàng tự động. YOLO6 cung cấp tốc độ suy luận nhanh và tối ưu hóa tốt cho các hệ thống thực tế.

- YOLO 7 [9]: Ra mắt vào năm 2022, YOLO7 được xem là một trong những phiên bản mạnh mẽ nhất với khả năng cân bằng giữa tốc độ và độ chính xác, vượt trội trong các bài toán phát hiện đối tượng thời gian thực.

- YOLO 8 [9]: Phát hành đầu năm 2023 bởi Ultralytics, YOLO8 giới thiệu nhiều cải tiến về kiến trúc và thuật toán huấn luyện, hỗ trợ đa nhiệm (phân loại, phát hiện, phân đoạn). Đây là phiên bản rất linh hoạt trong ứng dụng.

- YOLO 9 [9]: Công bố năm 2024, YOLO 9 nâng cao hiệu quả với các cải tiến trong thuật toán huấn luyện và kiến trúc mạng, mang lại độ chính xác cao hơn trên dữ liệu phức tạp.

- YOLO 10 [9]: Ra đời giữa năm 2024, YOLO 10 tập trung vào việc phát hiện đối tượng trong môi trường đông đúc và phức tạp, đồng thời giảm chi phí tính toán, giúp mô hình hoạt động hiệu quả ngay cả trên thiết bị phần cứng hạn chế.

- YOLO11 [9]: Phiên bản mới nhất, phát hành năm 2025. YOLO11 tích hợp các cải tiến vượt bậc cả về tốc độ và độ chính xác, đồng thời mở rộng khả năng xử lý cho nhiều tác vụ thị giác máy tính [2] khác nhau, trở thành chuẩn mực mới trong lĩnh vực phát hiện đối tượng.

1.3. Bài toán phát hiện người ngủ gật

Bài toán phát hiện người lái xe ngủ gật từ hình ảnh là một hướng nghiên cứu quan trọng trong lĩnh vực thị giác máy tính [2], với mục tiêu nâng cao an toàn giao thông. Việc nhận diện trạng thái buồn ngủ hoặc ngủ gật của tài xế thông qua hình ảnh từ camera cabin cho phép hệ thống đưa ra cảnh báo sớm, giúp ngăn ngừa nguy cơ mất kiểm soát phương tiện. Đây là yếu tố then chốt trong việc bảo vệ tính mạng của tài xế, hành khách và người tham gia giao thông.

Giải pháp này có tính ứng dụng cao, đặc biệt trong các hệ thống giám sát thông minh trên ô tô, xe tải, taxi, xe buýt và các phương tiện đường dài. Bằng cách kết hợp các mô hình học sâu như YOLO [9] với logic thời gian và đặc trưng hình học (EAR, PERCLOS, head pose), hệ thống có thể nhận diện chính xác trạng thái hành vi như mắt nhắm, gật đầu, mất tập trung. Việc triển khai mô hình nhẹ, tốc độ cao giúp đảm bảo khả năng hoạt động thời gian thực, phù hợp với môi trường vận hành liên tục và yêu cầu phản ứng nhanh. Đây là bước tiến quan trọng hướng tới giao thông an toàn và thông minh.

Bài toán phát hiện người lái xe ngủ gật từ hình ảnh đặt ra mục tiêu:

- Phát hiện sớm và cảnh báo kịp thời, xác định trạng thái buồn ngủ hoặc ngủ gật để cảnh báo, giảm nguy cơ mất kiểm soát phương tiện.
- Bảo vệ an toàn con người, ngăn ngừa tai nạn, đảm bảo tính mạng cho tài xế và hành khách.
- Ứng dụng thực tiễn vào hệ thống giám sát thông minh trên ô tô, hỗ trợ lái xe đường dài, lái xe tải, taxi, xe bus,…

1.4. Đánh giá độ tin cậy của bài toán phân loại

Độ tin cậy của mô hình phát hiện người lái xe ngủ gật là yếu tố quan trọng nhằm đánh giá mức độ chính xác, khả năng phản ứng và tính ổn định của hệ thống trong môi trường thực tế. Việc đánh giá này giúp xác định xem mô hình có thể hoạt động hiệu quả khi triển khai trên các thiết bị giám sát trong xe hay không.

Các chỉ số được sử dụng phổ biến trong quá trình đánh giá bao gồm Precision**,** Recall**,** AP (Average Precision) và mAP (mean Average Precision)**.** Dưới đây là các tiêu chí và cách tính cụ thể.

1.4.1. Precision và Recall

Precision – đại diện cho độ tin cậy của mô hình, cho biết trong số các đối tượng được dự đoán là *Positive* (ví dụ: buồn ngủ), có bao nhiêu phần trăm thực sự là *Positive*.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (1\text{-}1)$$

Recall – đại diện cho độ nhạy của mô hình, cho biết trong số các đối tượng thực sự *Positive*, mô hình phát hiện đúng được bao nhiêu phần trăm.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (1\text{-}20$$

Trong đó:

- True Positive (TP): Mô hình dự đoán đúng đối tượng.
  Ví dụ: Có một tài xế buồn ngủ trong ảnh và mô hình dự đoán đúng là "buồn ngủ".
- False Positive (FP): Mô hình dự đoán sai, tức là nhận diện một đối tượng không tồn tại.

  Ví dụ: Không có tài xế buồn ngủ trong ảnh nhưng mô hình lại dự đoán là "buồn ngủ".

- True Negative (TN): Mô hình dự đoán đúng rằng không có đối tượng nào.
  Ví dụ: Không có tài xế buồn ngủ và mô hình cũng không phát hiện ra.
- False Negative (FN): Mô hình bỏ sót đối tượng thật sự tồn tại.
  Ví dụ: Có tài xế buồn ngủ trong ảnh nhưng mô hình không phát hiện được.

Precision và Recall có mối quan hệ nghịch đảo: khi Precision cao thì Recall có thể giảm và ngược lại. Vì vậy, để đánh giá tổng thể, người ta thường sử dụng thêm chỉ số F1-score – là trung bình điều hòa giữa Precision và Recall, giúp cân bằng giữa hai yếu tố này.

1.4.2. AP (Average Precision)

Average Precision (AP) là giá trị trung bình của Precision tại các mức Recall khác nhau.

AP thể hiện sự cân bằng giữa độ chính xác và độ bao phủ của mô hình.

Công thức nội suy 11 điểm được sử dụng để tính AP như sau:

$$AP = \frac{1}{11} \sum r \in \{0, 0.1, \ldots, 1\} \; Pinterp(r) \quad (1\text{-}3)$$

Trong đó: Pinterp(r)=r~≥rmaxP(r~)

Với P(r~) P(\tilde{r})P(r~) là độ chính xác tại mức thu hồi r~\tilde{r}r~.

AP được tính bằng cách lấy trung bình độ chính xác tại 11 mức Recall khác nhau (0, 0.1, 0.2, ..., 1), trong đó tại mỗi mức, ta chọn giá trị Precision lớn nhất ứng với mức Recall bằng hoặc lớn hơn.

Ví dụ minh họa**:**

Giả sử mô hình có đồ thị Precision–Recall như hình 1.9, sau khi áp dụng phương pháp nội suy 11 điểm, ta có:

$$\text{AP}=\frac{1}{11}\ (1 + 0.6666 + 0.4285 + 0.4285 + 0.4285 + 0 + 0 + 0 + 0 + 0 + 0) = 26.84\%$$

1.4.3. mAP (mean Average Precision)

Mean Average Precision (mAP) là giá trị trung bình của tất cả các AP tương ứng với từng lớp đối tượng được phát hiện. Đây là chỉ số phổ biến trong các bài toán *Object Detection* (phát hiện đối tượng).

Công thức tính như sau:

$$\text{mAP}= \frac{\text{NAPa+APb+APc+}\cdots}{N} \qquad (1-5)$$

Trong đó:

- APa,APb,APcAP_a, AP_b, AP_cAPa,APb,APc là các giá trị AP của từng lớp (ví dụ: *tỉnh táo*, *buồn ngủ*, *ngủ gật*).
- NNN là tổng số lớp cần đánh giá.

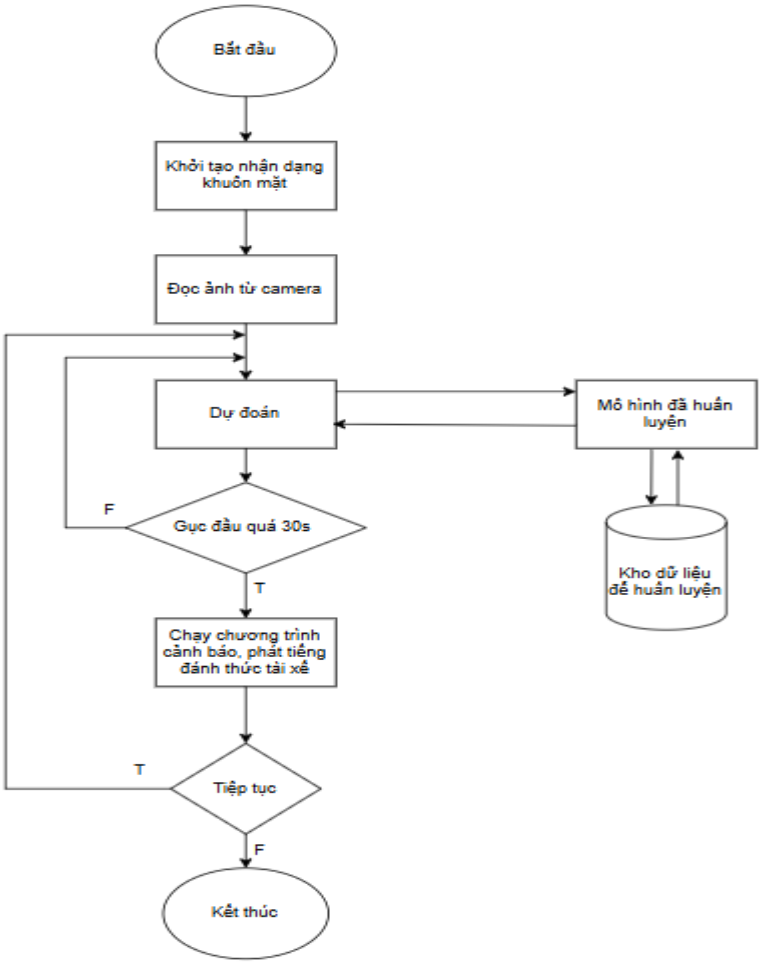
Giá trị mAP càng cao thể hiện mô hình có khả năng phát hiện chính xác và ổn định trên nhiều loại đối tượng khác nhau.

Thông thường, trong các mô hình phát hiện người lái xe ngủ gật, mAP đạt từ 0.8 – 0.95 được xem là đáng tin cậy để ứng dụng trong thực tế.

CHƯƠNG 2. PHƯƠNG PHÁP PHÁT HIỆN NGƯỜI LÁI XE NGỦ GẬT

Chương 2 trình bày quy trình xây dựng mô hình phát hiện tài xế ngủ gật, bao gồm sơ đồ hoạt động, chuẩn bị dữ liệu và quá trình huấn luyện. Bộ dữ liệu được thu thập, gán nhãn và xử lý để đưa vào mô hình YOLO11 – phiên bản cải tiến mạnh về tốc độ và độ chính xác trong phát hiện đối tượng. Chương cũng mô tả các thành phần kiến trúc của YOLO11, những điểm mạnh – yếu và lý do lựa chọn mô hình này cho bài toán. Cuối cùng, chương làm rõ cách YOLO11 được áp dụng để nhận diện hành vi mệt mỏi như mắt nhắm, gục đầu nhằm cảnh báo nguy hiểm khi lái xe.

2.1. Sơ đồ hoạt động của mô hình



Hình 2.1. hình vẽ sơ đồ hoạt động của mô hình

Mô hình có đầu vào là các video quan sát người lái xe được thu thập từ camera đặt trước vị trí người lái. Việc tính toán và phát hiện sẽ được thực hiện

liên tục trong các ảnh đầu vào, nếu tài xế nhắm mắt, ngáp, không tập trung lái xe quá một ngưỡng thời gian nhất định thì sẽ phát ra âm thanh cảnh bảo.Thuật toán YOLO sẽ làm nhiệm vụ nhận diện các trạng thái khuôn mặt để kết luận việc tài xế có ngủ gật hay không. Không chỉ nhận diện ngủ gật dựa trên mặt, thuật toán YOLO cũng sẽ xác định ngủ gật thông qua gục đầu trong một khoảng thời gian nhất định cũng sẽ đưa ra cảnh báo.

Các thành phần chính của mô hình:

- Khởi tạo tham số: là quá trình gán giá trị ban đầu cho các tham số. Khi tham số đã được khởi tạo, nếu người dùng không truyền giá trị tương ứng khi gọi hàm thì chương trình sẽ sử dụng giá trị mặc định này.
- Đọc hình ảnh từ camera: Thu nhận dữ liệu ảnh trực tiếp từ thiết bị camera để có thể bắt được khuôn mặt nhằm phục vụ bài toán phát hiện tài xế lái xe ngủ gật.
- Dự đoán: là quá trình ước lượng trước một kết quả hoặc sự kiện chưa xảy ra, dựa trên dữ liệu hiện tại, kiến thức có sẵn, hoặc các mô hình phân tích.
- Chạy chương trình**:** là quá trình máy tính thực thi các lệnh và câu lệnh

2.2. Phương pháp phát hiện bằng YOLO11

2.2.1 YOLO11 là gì

Sự phát triển của thuật toán YOLO [9] đạt đến tầm cao mới với sự ra đời của YOLO11, đại diện cho một bước tiến đáng kể trong công nghệ phát hiện đối tượng theo thời gian thực. Phiên bản mới nhất này xây dựng dựa trên thế mạnh của phiên bản trước đó đồng thời giới thiệu các khả năng mới giúp mở rộng tiện ích của nó trên nhiều ứng dụng CV khác nhau.

YOLO11 nổi bật nhờ khả năng thích ứng được cải tiến, hỗ trợ phạm vi mở rộng các tác vụ CV ngoài việc phát hiện đối tượng truyền thống.

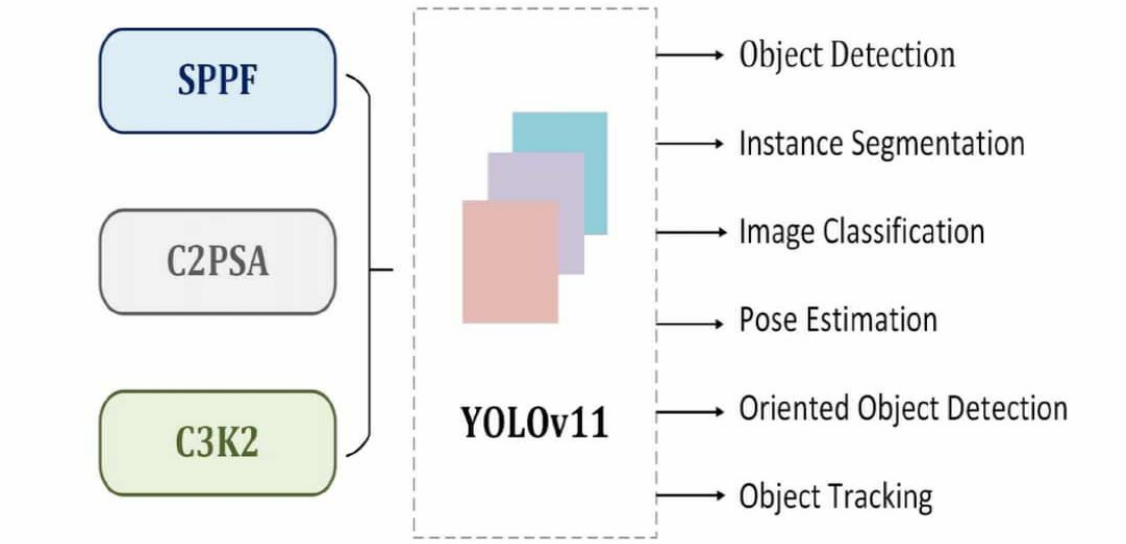
Đáng chú ý trong số này là ước tính tư thế và phân đoạn trường hợp, mở rộng khả năng ứng dụng của mô hình trong nhiều lĩnh vực khác nhau. Thiết kế

củaYOLO11 tập trung vào việc cân bằng sức mạnh và tính thực tiễn, nhằm giải quyết các thách thức cụ thể trong nhiều ngành công nghiệp khác nhau với độ chính xác và hiệu quả cao hơn.

Mô hình mới nhất này chứng minh sự phát triển liên tục của công nghệ phát hiện đối tượng theo thời gian thực, mở rộng ranh giới của những gì cóthể trong các ứng dụng CV. Tính linh hoạt và cải tiến hiệu suất của nó định vị YOLO11 là một tiến bộ đáng kể trong lĩnh vực này, có khả năng mở ra những con đường mới cho việc triển khai thực tế trên nhiều lĩnh vực khác nhau

Dựa trên kiến trúc đã được thiết lập này, YOLO11 mở rộng và nâng cao nền tảng được đặt ra bởi YOLOv8, giới thiệu những cải tiến về kiến trúc và tối ưu hóa tham số để đạt được hiệu suất phát hiện vượt trội như minh họa trong hình dưới đây.

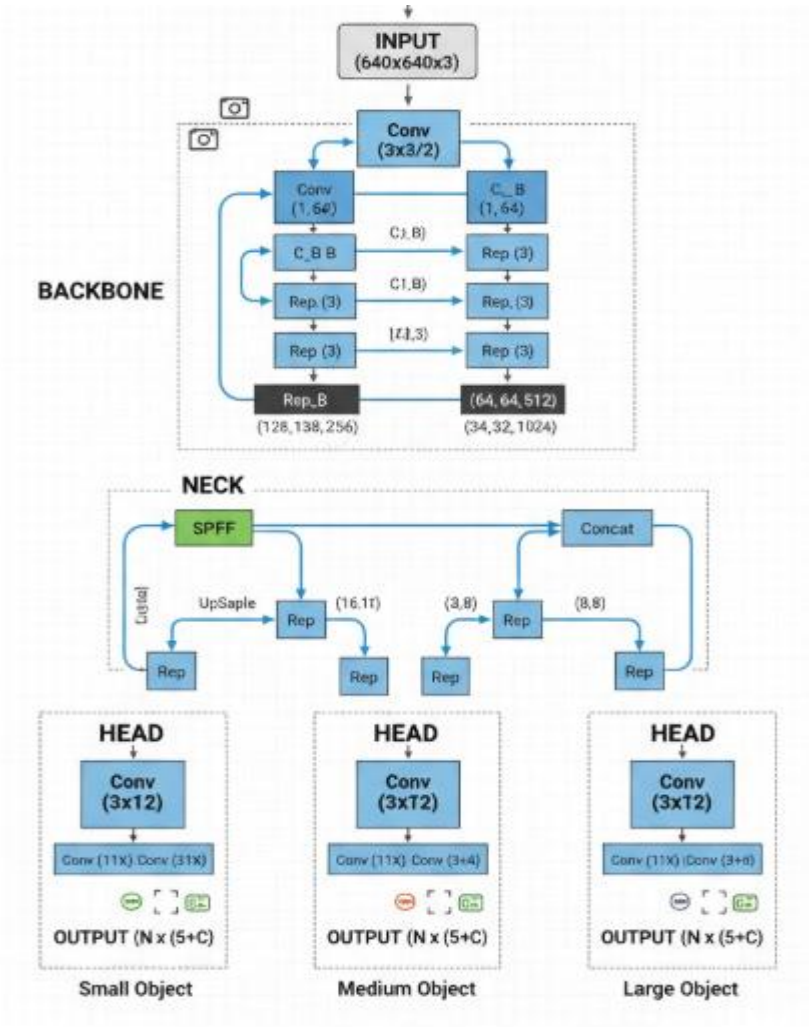
Các phần sau đây trình bày chi tiết các sửa đổi kiến trúc chính được triển khai trong YOLO11:



Hình 2.2.Các mô-đun chính trong YOLO11

2.2.2. Cấu trúc của YOLO11

Để minh họa rõ hơn cách mô hình YOLO11 hoạt động từ giai đoạn trích xuất đặc trưng đến dự đoán đầu ra, hình dưới đây trình bày cấu trúc tổng thể của mô hình. Kiến trúc này bao gồm ba phần chính: Backbone, Neck và Head, được tổ chức theo dạng pipeline nhằm tối ưu tốc độ và độ chính xác.



Hình 2.3 hình cấu trúc của YOLO11

2.2.2.1. Backbone — trích xuất đặc trưng

Mục tiêu của module này là trích xuất biểu diễn đa tỉ lệ (multi-scale) chứa cả thông tin cấu trúc (edges, corners) và ngữ nghĩa (khuôn mặt, mắt). YOLO11 giữ cấu trúc backbone truyền thống nhưng thay thế/tiếp thêm các khối có attention để cải thiện bắt đặc trưng nhỏ.

Khối SPFF và C2PSA

YOLO11 vẫn giữ nguyên khối Spatial Pyramid Pooling - Fast (SPPF) từ các phiên bản trước nhưng giới thiệu một khối Cross Stage Partial with

Spatial Attention (C2PSA) mới. Khối C2PSA là một bổ sung đáng chú ý giúp tăng cường sự chú ý không gian trong các bản đồ đặc điểm. Cơ chếchú ý không gian này cho phép mô hình tập trung hiệu quả hơn vào các vùng quan trọng trong hình ảnh. Bằng cách gộp các đặc điểm theo không gian,khối C2PSA cho phép YOLO [9]11 tập trung vào các khu vực quan tâm cụ thể, có khả năng cải thiện độ chính xác phát hiện đối với các đối tượng có kích thước và vị trí khác nhau.

C2PSA (Cross Stage Partial with Spatial Attention) là thành phần quan trọng nhất trong Backbone của YOLO11, đóng vai trò tăng cường khả năng biểu diễn đặc trưng đa không gian. Khối này bao gồm hai nhánh xử lý song song theo cơ chế CSP và một nhánh chú ý không gian PSA.

Nhánh trích xuất đặc trưng (Feature Extraction Branch)

Nhánh này tiếp nhận một phần của tensor đầu vào và thực hiện chuỗi các lớp tích chập 1×1 và 3×3.

- Tích chập 1×1 có nhiệm vụ điều chỉnh số kênh đặc trưng.
- Các tích chập 3×3 được sử dụng để học đặc trưng không gian sâu hơn. Sự kết hợp này cho phép khối học được các mẫu phức tạp trong ảnh như đường viền khuôn mặt, cấu trúc mắt và vùng chuyển động nhỏ.

Nhánh chú ý không gian (Spatial Attention Branch – PSA)

Nhánh PSA xử lý phần còn lại của đặc trưng đầu vào và tạo ra bản đồ chú ý nhằm làm nổi bật các vùng quan trọng trong ảnh. Quá trình này bao gồm các bước:

- giảm số kênh đầu vào bằng tích chập 1×1 để tối ưu chi phí tính toán;
- trích xuất ngữ cảnh không gian bằng các phép pooling kích thước 5×5 cùng với tích chập giãn cách (dilated convolution);
- tổng hợp thông tin từ nhiều nguồn đặc trưng để tạo ra bản đồ chú ý không gian;
- chuẩn hóa bản đồ chú ý bằng hàm kích hoạt sigmoid và nhân trực tiếp vào tensor đặc trưng.

Nhánh PSA giúp mô hình tập trung vào các vùng đặc biệt quan trọng như mắt, miệng và đầu người – những chi tiết rất nhỏ nhưng đóng vai trò then chốt trong bài toán phát hiện trạng thái buồn ngủ.

Hợp nhất hai nhánh

Sau khi hai nhánh xử lý xong, tensor đặc trưng được ghép nối và đưa qua một lớp tích chập 1×1 để kết hợp thông tin và ổn định số kênh. Việc hợp nhất này giúp mô hình vừa giữ được đặc trưng chi tiết cục bộ vừa tập trung được vào các vùng quan trọng nhờ cơ chế chú ý.

Các lớp giảm kích thước đặc trưng

Giữa các tầng backbone, YOLO11 sử dụng các lớp tích chập 3×3 với bước nhảy 2 để giảm kích thước đặc trưng theo từng mức. Việc giảm kích thước này giúp mô hình trích xuất được các đặc trưng ngữ nghĩa ở mức cao hơn, đồng thời giảm chi phí tính toán ở các tầng sâu.

Khối SPPF – Spatial Pyramid Pooling Fast

SPPF được đặt tại cuối Backbone nhằm mở rộng vùng cảm thụ (receptive field) của mô hình. Khối này sử dụng kỹ thuật stacking các lớp pooling 5×5 liên tiếp để mô phỏng pooling 9×9 và 13×13 mà không cần sử dụng kernel lớn.

Cấu trúc SPPF bao gồm:

- một lớp tích chập 1×1 để nén thông tin ban đầu;

ba tầng pooling nối tiếp nhau;

- ghép nối kết quả của bốn tầng đặc trưng;
- một lớp tích chập 1×1 cuối để hợp nhất thông tin.

Khối SPPF giúp Backbone hiểu được ngữ cảnh rộng của khuôn mặt và hình thể, đặc biệt hiệu quả khi đối tượng có nhiều biến dạng về tư thế.

2.2.2.2 Cấu tạo nội bộ của Neck

Neck của YOLO11 đảm nhiệm nhiệm vụ tổng hợp đặc trưng từ nhiều mức độ khác nhau để tăng cường khả năng phát hiện vật thể nhỏ, trung bình và lớn. Cấu trúc Neck bao gồm hai luồng chính: FPN (Feature Pyramid Network) và PAN (Path Aggregation Network), cùng với khối C3k2 được tối ưu hóa.

Luồng FPN (top–down)

Luồng FPN truyền thông tin từ tầng đặc trưng sâu xuống các tầng nông hơn. Quá trình này bao gồm:

- phóng to đặc trưng bằng upsampling;
- ghép nối với đặc trưng có cùng kích thước từ Backbone;
- trộn đặc trưng bằng khối C3k2.

Nhờ cơ chế này, các tầng có độ phân giải cao hơn được bổ sung thông tin ngữ nghĩa, giúp phát hiện các đối tượng nhỏ (như mắt hoặc mí mắt) hiệu quả hơn.

Luồng PAN (bottom–up)

Luồng PAN thực hiện chiều ngược lại:

- giảm kích thước đặc trưng bằng lớp tích chập stride 2;
- ghép nối với đặc trưng ở mức sâu hơn;
- xử lý bằng khối C3k2.

PAN giúp tăng cường thông tin chi tiết cho các tầng sâu, cải thiện khả năng mô hình phân biệt giữa các trạng thái tương tự (như chớp mắt và ngủ gật).

Khối C3k2

YOLO11 giới thiệu một thay đổi đáng kể bằng cách thay thế khối C2f trong cổ bằng khối C3k2. Khối C3k2 được thiết kế để nhanh hơn và hiệu

quả hơn, nâng cao hiệu suất tổng thể của quy trình tổng hợp tính năng. Sau khi lấy mẫu và nối, cổ trong YOLO11 kết hợp khối cải tiến này, dẫn đến tốc độ và hiệu suất được nâng cao

Các đặc điểm chính của khối C3k2:

- Xử lý nhanh hơn: Việc sử dụng hai phép tích chập nhỏ hơn giúp giảm chi phí tính toán so với một phép tích chập lớn, giúp trích xuất tính năng nhanh hơn.
- Hiệu quả tham số: C3k2 là phiên bản nhỏ gọn hơn của nút thắt CSP, giúp kiến trúc hiệu quả hơn về số lượng tham số có thể đào tạo

2.2.2.3. Cấu tạo nội bộ của Detection Head

Detection Head là thành phần cuối của mô hình, chịu trách nhiệm dự đoán hộp giới hạn và nhãn lớp trên ba mức đặc trưng đầu ra

Kiến trúc anchor-free

Khác với các phiên bản YOLO cũ sử dụng anchor cố định, YOLO11 dự đoán trực tiếp tọa độ hộp bao thông qua các điểm đặc trưng (feature points). Điều này giúp mô hình:

- giảm lỗi dự đoán do anchor không phù hợp;
- tăng độ chính xác đối với các đối tượng nhỏ;
- đơn giản hóa quá trình huấn luyện và tính toán.

Các đầu ra của một điểm đặc trưng

Mỗi điểm đặc trưng dự đoán các giá trị sau:

- vị trí tương đối của tâm hộp;
- chiều rộng và chiều cao của hộp;
- độ tin cậy (objectness);
- xác suất lớp;
- giá trị chất lượng (quality) nhằm cải thiện quá trình NMS.

Detection Head xử lý đặc trưng qua một số lớp tích chập nhẹ, sau đó chia thành các nhánh dự đoán riêng biệt.

Cơ chế Loss và tối ưu hóa nội bộ

YOLO11 sử dụng tổ hợp các hàm Loss nâng cao:

- SIoU, EIoU hoặc CIoU cho dự đoán hộp bao;
- Varifocal Loss để khớp nhãn mềm và giá trị chất lượng;
- Distribution Focal Loss giúp dự đoán biên hộp chính xác hơn.

Những hàm mất mát này giúp YOLO11 đạt độ chính xác cao hơn, đặc biệt ở các đối tượng nhỏ hoặc bị che phủ một phần.

Tổng kết cấu tạo nội bộ YOLO11

Cấu trúc nội bộ của YOLO11 được tổ chức theo hướng module hóa, trong đó:

- Backbone chịu trách nhiệm trích xuất đặc trưng đa cấp;
- C2PSA giúp mô hình tập trung vào các vùng quan trọng nhờ cơ chế chú ý không gian;
- SPPF mở rộng vùng nhận thông tin trong khi vẫn giữ tốc độ;
- C3k2 trong Neck tối ưu quá trình tổng hợp đa đặc trưng;
- Detection Head anchor-free đơn giản hóa dự đoán và nâng cao hiệu quả.
- Các hàm Loss hiện đại giúp mô hình hội tụ nhanh và ổn định.

Nhờ kiến trúc tối ưu và chặt chẽ này, YOLO11 trở thành một trong những mô hình hiệu quả nhất trong họ YOLO, đặc biệt phù hợp cho các bài toán nhận dạng thời gian thực như phát hiện tài xế ngủ gật.

2.2.3. Những tính năng chính của YOLO11

Dưới đây là những tính năng của YOLO11:

1. Độ chính xác được cải thiện với độ phức tạp được giảm bớt: Biến thể YOLO11 đạt được điểm Độ chính xác trung bình (mAP) vượt trội trên tập dữ liệu COCO trong khi sử dụng ít hơn 22% tham số so với biến thể YOLOv8, chứng minh hiệuquả tính toán được cải thiện mà không ảnh hưởng đến độ chính xác .

2. Tính linh hoạt trong các tác vụ CV: YOLO11 [3] thể hiện sự thành thạo trong nhiều ứng dụng CV khác nhau, bao gồm ước tính tư thế, nhận dạng đối tượng, phân loại hình ảnh, phân đoạn thể hiện và phát hiện hộp giới hạn định hướng (OBB).

3. Tốc độ và hiệu suất được tối ưu hóa: Thông qua các thiết kế kiến trúc tinh tế và quy trình đào tạo hợp lý, YOLO11 đạt được tốc độ xử lý nhanh hơn trong khi vẫn duy trì sự cân bằng giữa độ chính xác và hiệu quả tính toán.

4. Số lượng tham số được sắp xếp hợp lý: Việc giảm các tham số góp phần làm cho hiệu suất mô hình nhanh hơn mà không ảnh hưởng đáng kể đến độ chính xác tổng thể của YOLO11 [3].

5. Trích xuất tính năng nâng cao: YOLO11 kết hợp các cải tiến trong cả kiến trúc xương sống và cổ, mang lại khả năng trích xuất tính năng nâng cao và do đó, phát hiện đối tượng chính xác hơn.

6. Khả năng thích ứng theo ngữ cảnh: YOLO11 chứng minh tính linh hoạt trong nhiều tình huống triển khai khác nhau,bao gồm nền tảng đám mây, thiết bị biên và hệ thống được tối ưu hóa cho GPU NVIDIA.

2.2.4. Diểm mạnh và điểm yếu của YOLO11

Điểm mạnh:

- Cân bằng hiệu suất vượt trội: Đạt được sự cân bằng vượt trội giữa tốc độ và độ chính xác, vượt trội hơn các mô hình khác trên nhiều nền tảng phần cứng khác nhau.

- Tính Linh hoạt Vượt trội: Một họ mô hình duy nhất xử lý năm tác vụ AI thị giác chính, đơn giản hóa quá trình phát triển cho các ứng dụng phức tạp.

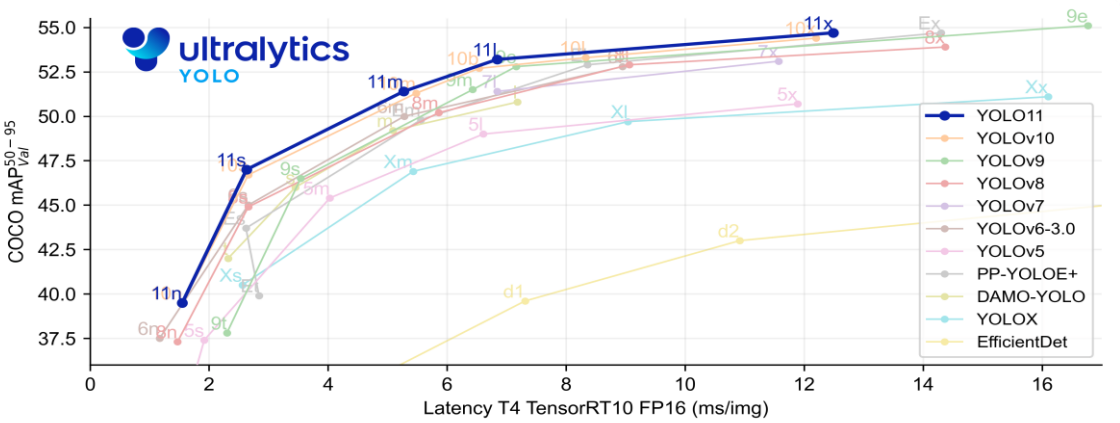
- Hệ sinh thái được duy trì tốt: Được hỗ trợ bởi sự phát triển tích cực, một cộng đồng lớn mạnh, cập nhật thường xuyên và các tài nguyên toàn diện, đảm bảo độ tin cậy và hỗ trợ.

- Dễ sử dụng: Được thiết kế cho trải nghiệm người dùng được tối ưu hóa, cho phép cả người mới bắt đầu và chuyên gia huấn luyện và triển khai mô hình với mức độ phức tạp tối thiểu.

- Hiệu quả đào tạo và triển khai: Được tối ưu hóa cho thời gian đào tạo nhanh hơn và sử dụng bộ nhớ thấp hơn, lý tưởng cho nhiều loại phần cứng từ thiết bị điện đến máy chủ đám mây.

Điểm Yếu:

- Là một mô hình hiện đại, các biến thể YOLO11lớn nhất đòi hỏi tài nguyên tính toán đáng kể để đạt được độ chính xác tối đa, mặc dù chúng vẫn rất hiệu quả so với lớp hiệu suất của chúng.



Hình 2.4. YOLO11 so với các phiên bản trước

2.3. Phát hiện tài xế ngủ gật bằng YOLO11

Phát hiện tình trạng buồn ngủ hoặc ngủ gật của tài xế là một trong những ứng dụng quan trọng của các mô hình học sâu trong lĩnh vực thị giác máy tính [2]. Trong đó, YOLO11 được đánh giá là một trong những kiến trúc mạng nơ-ron tích chập mạnh mẽ và hiệu quả, có khả năng phát hiện đối tượng trong thời gian thực.

Khác với các phương pháp truyền thống tách riêng bước trích xuất đặc trưng và phân loại, YOLO11 xử lý toàn bộ bức ảnh chỉ qua một lần lan truyền mạng, đồng thời dự đoán trực tiếp vị trí và nhãn của đối tượng. Cơ chế này giúp mô hình đạt được tốc độ xử lý rất cao, đáp ứng yêu cầu vận hành trong môi trường giao thông thực tế – nơi mà độ trễ trong cảnh báo có thể dẫn đến tai nạn.

Trong bài toán phát hiện tài xế ngủ gật, YOLO11 [9] có thể được huấn luyện để nhận diện các đặc trưng hành vi liên quan đến sự mệt mỏi như:

- Mắt nhắm trong thời gian dài (PERCLOS): Dấu hiệu tài xế buồn ngủ.
- Tần suất chớp mắt bất thường.
- Gật đầu hoặc thay đổi tư thế đầu (head pose).
- Mất tập trung, không quan sát phía trước.

Dữ liệu hình ảnh cabin được đưa vào mô hình YOLO [9] sau khi gán nhãn (bounding box) cho các vùng quan tâm như mắt, mũi, miệng, khuôn mặt. Quá

trình huấn luyện giúp mô hình học được mối liên hệ giữa các đặc trưng hình ảnh này với trạng thái tỉnh táo hoặc buồn ngủ. Khi triển khai thực tế, hệ thống camera trong xe sẽ truyền liên tục hình ảnh về mô hình YOLO [9] để đưa ra dự đoán trong thời gian thực.

Ưu điểm nổi bật của việc sử dụng YOLO11 trong phát hiện tài xế ngủ gật bao gồm:

- Tốc độ cao: Đảm bảo cảnh báo tức thời cho tài xế.
- Độ chính xác tốt: Nhờ khả năng học sâu từ dữ liệu lớn.
- Khả năng triển khai linh hoạt: Có thể áp dụng trên các thiết bị phần cứng giới hạn như camera giám sát trong ô tô.

Tuy nhiên, thách thức đặt ra là sự thay đổi điều kiện môi trường (ánh sáng yếu, ban đêm, kính mắt, khẩu trang che mặt) có thể làm giảm hiệu quả phát hiện. Do đó, việc cải tiến mô hình, kết hợp thêm các đặc trưng hình học (EAR, PERCLOS) hoặc các mô hình theo dõi chuỗi thời gian (LSTM) sẽ giúp nâng cao độ tin cậy.

Như vậy, việc áp dụng YOLO11 trong phát hiện tài xế ngủ gật không chỉ mang lại tính khả thi về mặt kỹ thuật mà còn mở ra tiềm năng triển khai thực tế trong các hệ thống giám sát an toàn giao thông thông minh.

CHƯƠNG 3 CÀI ĐẶT, THỬ NGHIỆM VÀ ĐÁNH GIÁ

Chương 3 trình bày môi trường cài đặt và các công cụ sử dụng để xây dựng mô hình như Python, Visual Studio Code và Google Colab. Bộ dữ liệu được chuẩn bị, tăng cường và phân chia phục vụ huấn luyện mô hình YOLO11. Quá trình đào tạo mô hình được thực hiện với nhiều giá trị epoch để tìm ra cấu hình tối ưu, đồng thời các biểu đồ học và ma trận nhầm lẫn được phân tích để đánh giá hiệu suất. Kết quả thử nghiệm cho thấy mô hình đạt độ chính xác cao và hoạt động tốt trên ảnh, video, góp phần phát hiện tài xế ngủ gật theo thời gian thực

3.1 Môi trường thử nghiệm

3.1.1 Python

3.1.1.1. Khái niệm

Python [5] là một ngôn ngữ lập trình bậc cao được thiết kế phục vụ nhiều mục đích khác nhau, do Guido van Rossum phát triển và ra mắt lần đầu vào năm 1991. Ngôn ngữ này được biết đến nhờ cú pháp rõ ràng, dễ đọc, dễ hiểu và dễ ghi nhớ, giúp người mới học lập trình có thể tiếp cận nhanh chóng. Python [5] sử dụng hình thức trình bày mạch lạc, có cấu trúc hợp lý và được xem là một trong những ngôn ngữ thân thiện nhất với người học.

Python [5] là ngôn ngữ lập trình hướng đến tính dễ học và hiệu quả, được ứng dụng rộng rãi trong nhiều lĩnh vực, đặc biệt là trí tuệ nhân tạo [1], khoa học dữ liệu, phân tích dữ liệu, và phát triển web. Cấu trúc của Python [5] cho phép người lập trình viết mã ngắn gọn hơn so với nhiều ngôn ngữ khác mà vẫn đảm bảo hiệu suất và độ chính xác cao.

Vào tháng 7 năm 2018, sau hơn 30 năm đóng góp, Van Rossum tuyên bố rời vị trí lãnh đạo trong cộng đồng Python [5], để lại một di sản lớn cho thế giới lập trình. Python [5] hiện được quản lý và phát triển bởi Python [5] Software Foundation một tổ chức phi lợi nhuận điều phối việc phát triển và định hướng của ngôn ngữ này.

Python [5] là một ngôn ngữ động, có khả năng cấp phát bộ nhớ tự động, tương tự như các ngôn ngữ hiện đại khác như Ruby, Perl, Scheme hay Smalltalk.

Ban đầu, Python [5] được phát triển để hoạt động trên hệ điều hành Unix, nhưng theo thời gian, phạm vi hỗ trợ của nó đã mở rộng sang nhiều nền tảng khác như Windows, macOS, Linux, và nhiều hệ điều hành tương tự Unix.

Mặc dù quá trình phát triển của Python [5] có sự đóng góp từ cộng đồng toàn cầu, Guido van Rossum vẫn được xem là người có vai trò then chốt trong việc định hình tư tưởng và triết lý của ngôn ngữ này. Nhờ tính linh hoạt, dễ học và mạnh mẽ, Python [5] liên tục nằm trong nhóm những ngôn ngữ lập trình phổ biến và được ưa chuộng nhất thế giới hiện nay.

3.1.1.2. Đặc tính của Python [5]

Python [5] ngày càng được sử dụng rộng rãi trong cộng đồng lập trình nhờ sở hữu nhiều đặc tính nổi bật, hỗ trợ tối đa cho cả người mới học lẫn lập trình viên chuyên nghiệp. Một số đặc trưng chính của ngôn ngữ Python [5] có thể kể đến như sau:

- Ngôn ngữ hướng đối tượng: Python [5] hỗ trợ hoàn chỉnh các kỹ thuật lập trình hướng đối tượng, cho phép tổ chức và quản lý mã nguồn theo mô hình lớp và đối tượng, giúp chương trình dễ mở rộng và bảo trì hơn.

- Lập trình tương tác: Python [5] cho phép người dùng làm việc trực tiếp với trình thông dịch, thử nghiệm và chạy từng dòng lệnh một cách linh hoạt, rất thuận tiện trong quá trình kiểm thử và phát triển phần mềm.

- Dễ học, dễ sử dụng: Cú pháp của Python [5] gần gũi với ngôn ngữ tự nhiên, giúp người học nhanh chóng nắm bắt, đặc biệt phù hợp với những ai mới bắt đầu làm quen với lập trình.

- Cấu trúc cú pháp đơn giản: Việc loại bỏ dấu ngoặc nhọn và sử dụng thụt đầu dòng để phân chia khối lệnh khiến Python [5] trở nên trực quan, dễ đọc và dễ hiểu, từ đó giảm đáng kể lỗi cú pháp trong quá trình viết mã.

- Mã nguồn mở: Python [5] được phát hành theo giấy phép mã nguồn mở, cho phép người dùng tự do sử dụng, chỉnh sửa và phân phối lại. Điều

này góp phần thúc đẩy sự phát triển mạnh mẽ của cộng đồng Python [5] toàn cầu.

- Tính đa nền tảng: Python [5] có thể hoạt động trên nhiều hệ điều hành và kiến trúc phần cứng khác nhau như Windows, macOS, Linux hay Unix mà không cần thay đổi mã nguồn.

- Khả năng mở rộng: Python [5] hỗ trợ thêm các mô-đun hoặc thư viện mở rộng, giúp người dùng dễ dàng tích hợp các chức năng nâng cao vào chương trình mà không cần viết lại từ đầu.

- Cấu trúc linh hoạt: Python [5] cung cấp nhiều cơ chế giúp tổ chức mã nguồn logic và hiệu quả, đồng thời cho phép kết hợp với các ngôn ngữ khác như C/C++ hoặc Java để tăng hiệu suất.

- Cộng đồng phát triển mạnh: Python [5] có một cộng đồng người dùng đông đảo trên toàn thế giới, luôn tích cực đóng góp vào việc phát triển, cải tiến và mở rộng hệ sinh thái thư viện. Nhờ đó, người học và lập trình viên có thể dễ dàng tìm thấy tài liệu, hỗ trợ kỹ thuật và lời giải cho các vấn đề thực tế.

Nhờ những ưu điểm trên, Python [5] không chỉ trở thành một trong những ngôn ngữ phổ biến nhất hiện nay mà còn được xem là công cụ nền tảng trong nhiều lĩnh vực như trí tuệ nhân tạo [1], khoa học dữ liệu, và phát triển ứng dụng web.

3.1.1.3 Các phiên bản của Python [5]

Python [5] được Guido van Rossum khởi tạo vào cuối những năm 1980 tại Trung tâm Toán học – Tin học (CWI) ở Hà Lan, với mong muốn tạo ra một ngôn ngữ đơn giản, dễ hiểu nhưng vẫn mạnh mẽ, kế thừa những ưu điểm từ ngôn ngữ ABC. Python [5] được thiết kế để có khả năng xử lý ngoại lệ, tương tác với hệ điều hành Amoeba, và hỗ trợ tốt việc lập trình hướng cấu trúc. Phiên bản đầu tiên

của Python [5] được hoàn thiện vào tháng 12 năm 1989, đánh dấu sự ra đời của một trong những ngôn ngữ lập trình có sức ảnh hưởng lớn nhất hiện nay.

Phiên bản Python [5] 2.0, phát hành vào ngày 16 tháng 10 năm 2000, mang lại nhiều cải tiến đáng kể như bộ thu gom rác tự động (garbage collection) và khả năng xử lý ký tự Unicode, giúp ngôn ngữ này hỗ trợ tốt hơn cho các ứng dụng toàn cầu hóa.

Tiếp đó, Python [5] 3.0 ra mắt vào ngày 3 tháng 12 năm 2008, được xem là cột mốc quan trọng trong lịch sử phát triển của Python [5]. Đây là bản nâng cấp lớn và không tương thích ngược với các phiên bản 2.x, do đó một số mã nguồn cũ cần được chuyển đổi lại để sử dụng được. Tuy nhiên, nhiều tính năng mới của Python [5] 3 đã được chuyển mã ngược (backport) về các phiên bản 2.6.x và 2.7.x để đảm bảo quá trình chuyển đổi diễn ra thuận lợi hơn. Python [5] 3 đi kèm công cụ 2to3, giúp tự động chuyển đổi mã Python [5] 2 sang Python [5] 3.

Các phiên bản nâng cấp sau đó như Python [5] 3.9.2 và 3.8.8 tiếp tục được phát hành nhằm vá lỗi bảo mật nghiêm trọng (bao gồm cả trong Python [5] 2.7), khắc phục các lỗ hổng có thể dẫn đến nguy cơ thực thi mã độc từ xa và rò rỉ bộ nhớ đệm.

Trong năm 2022, Python [5] tiếp tục được cải tiến mạnh mẽ với các bản cập nhật 3.10.4, 3.9.12, và 3.8.13, đồng thời vá một số lỗi bảo mật còn tồn tại ở phiên bản 3.7.13. Đến cuối năm 2022, Python [5] 3.9.13 được phát hành, đánh dấu một bước tiến mới trong việc nâng cao tính an toàn và ổn định của ngôn ngữ. Vào tháng 9 năm 2022, bốn bản cập nhật tiếp theo cũng được công bố (3.10.7, 3.9.14, 3.8.14, và 3.7.14) nhằm khắc phục các lỗ hổng có thể bị khai thác trong tấn công từ chối dịch vụ (DoS).

Tính đến tháng 10 năm 2023, Python [5] 3.12 được công nhận là phiên bản ổn định và tối ưu nhất, mang đến hiệu năng cao hơn, cú pháp nhất quán hơn, cùng nhiều cải tiến về thư viện tiêu chuẩn. Một số thay đổi đáng chú ý từ phiên bản

3.11 cũng được giữ lại, như nâng cấp trình biên dịch, cải thiện tốc độ thực thi và quản lý bộ nhớ hiệu quả hơn.

3.1.2. Visual studio code

3.1.2.1 Khái niệm

- Visual Studio Code (VS Code) là một môi trường phát triển tích hợp (IDE – Integrated Development Environment) được phát triển bởi Microsoft, ra mắt lần đầu vào năm 2015. Đây là một trình soạn thảo mã nguồn mạnh mẽ, đa nền tảng, hỗ trợ lập trình trên Windows, macOS và Linux.

VS Code được thiết kế hướng đến sự nhẹ, linh hoạt và mở rộng cao, cho phép lập trình viên có thể cài đặt thêm các tiện ích mở rộng (extensions) để phục vụ nhiều ngôn ngữ khác nhau, trong đó có Python [5]. Với giao diện trực quan, tốc độ xử lý nhanh và khả năng tùy chỉnh mạnh mẽ, VS Code nhanh chóng trở thành một trong những công cụ lập trình phổ biến nhất trên thế giới.

Trình soạn thảo mã nguồn này này không chỉ hỗ trợ viết và chạy mã Python [5] mà còn tích hợp các chức năng phát hiện lỗi, gợi ý cú pháp, gỡ lỗi (debug) và quản lý môi trường ảo (virtual environment), giúp người dùng dễ dàng phát triển và thử nghiệm chương trình. Ngoài ra, VS Code còn tích hợp sẵn Git, giúp kiểm soát phiên bản mã nguồn thuận tiện ngay trong giao diện làm việc.

3.1.2.2 Các đặc trưng của visual studio code

VS code hỗ trợ các tính năng sau:

- Hỗ trợ đa ngôn ngữ: Ngoài Python [5], VS Code còn hỗ trợ nhiều ngôn ngữ lập trình khác như C/C++, Java, JavaScript, HTML, CSS, và Go.

- Giao diện thân thiện**:** VS Code có thiết kế tối giản, dễ thao tác, đồng thời cho phép người dùng tùy chỉnh giao diện, màu sắc, phím tắt và bố cục theo sở thích cá nhân.

- Hệ thống tiện ích mở rộng mạnh mẽ: Người dùng có thể cài đặt hàng nghìn tiện ích mở rộng từ Marketplace để hỗ trợ các tính năng nâng cao như linting, kiểm thử, kết nối cơ sở dữ liệu hoặc phát triển web.

- Tích hợp Git**:** VS Code cung cấp sẵn các công cụ quản lý mã nguồn như commit, push, pull và xem lịch sử thay đổi mà không cần rời khỏi IDE.

- Gỡ lỗi trực tiếp**:** Cho phép lập trình viên đặt breakpoint, theo dõi biến, và kiểm tra luồng thực thi của chương trình ngay trong giao diện.

- Hỗ trợ môi trường ảo: VS Code giúp người dùng tạo và quản lý môi trường ảo Python [5] dễ dàng, đảm bảo tính độc lập của các gói thư viện cho từng dự án.

- Hiệu năng cao, nhẹ và miễn phí**:** Dù có nhiều tính năng mạnh mẽ, VS Code vẫn chạy mượt mà trên hầu hết cấu hình máy tính, đặc biệt hoàn toàn miễn phí và có mã nguồn mở.

Nhờ những ưu điểm nổi bật này, Visual Studio Code đã trở thành lựa chọn hàng đầu cho các lập trình viên Python [5], đặc biệt trong các dự án nghiên cứu, thử nghiệm mô hình học máy hoặc phát triển ứng dụng nhỏ đến trung bình.

3.1.2.3. Các phiên bản của Visual Studio code

Visual Studio Code (VS Code) được phát triển bởi Microsoft và công bố phiên bản đầu tiên vào tháng 4 năm 2015. Ngay từ khi ra mắt, VS Code đã nhanh chóng thu hút sự quan tâm lớn từ cộng đồng lập trình nhờ tính năng gọn nhẹ, tốc độ nhanh và khả năng mở rộng cao thông qua các tiện ích (extensions).

VS Code là một dự án mã nguồn mở, được phát hành theo giấy phép MIT và có mã nguồn công khai trên nền tảng GitHub. Microsoft phát hành bản cập nhật hàng tháng, trong đó liên tục bổ sung tính năng mới, vá lỗi và cải thiện hiệu suất. Sau đây là các phiên bản:

- Phiên bản 1.0 (2016): Đây là bản phát hành ổn định đầu tiên, hỗ trợ cơ bản các ngôn ngữ lập trình phổ biến như JavaScript, TypeScript, và C#.
- Phiên bản 1.20–1.30 (2018–2019): Giai đoạn này tập trung nâng cao hiệu suất, cải thiện khả năng gỡ lỗi và tích hợp mạnh mẽ hơn với hệ thống Git.
- Phiên bản 1.40–1.50 (2020): VS Code bổ sung nhiều tiện ích hỗ trợ Python [5], Jupyter Notebook, và Remote Development, giúp phát triển ứng dụng từ xa và xử lý dữ liệu trực tiếp trên server.
- Phiên bản 1.60–1.70 (2021–2022): Các tính năng như Auto Save, Inline Suggestion (gợi ý mã thông minh), và Live Share (lập trình cộng tác thời gian thực) được cải thiện đáng kể.
- Phiên bản 1.80 trở đi (2023–2025): Tập trung vào việc tối ưu hóa hiệu năng, giao diện và tích hợp AI (ví dụ như GitHub Copilot), đồng thời cải thiện hỗ trợ ngôn ngữ Python [5], C/C++, và Rust.

Tính đến tháng 10 năm 2025, phiên bản mới nhất của VS Code đã trở nên ổn định, với hiệu suất cao hơn, tốc độ khởi động nhanh, và khả năng tùy biến sâu hơn thông qua các extension hiện đại. Các phiên bản mới vẫn giữ triết lý ban đầu của Microsoft: "nhẹ, linh hoạt, miễn phí và dành cho mọi lập trình viên."

3.1.3. Google Colab

3.1.3.1. Khái niệm

Google Colab [6]oratory, thường được gọi tắt là Google Colab [6], là một sản phẩm do Google Research phát triển. Đây là một nền tảng cho phép người dùng viết và thực thi mã nguồn Python [5] trực tiếp trên trình duyệt web mà không cần cài đặt hay cấu hình môi trường lập trình trên máy tính cá nhân. Google Colab [6] đặc biệt phù hợp với các lĩnh vực như phân tích dữ liệu (Data Analysis), học máy (Machine Learning) và giáo dục.

Một trong những ưu điểm nổi bật của Colab là khả năng chạy mã Python [5] thông qua tài nguyên phần cứng mạnh mẽ trên nền tảng điện toán đám mây của Google. Người dùng có thể dễ dàng lựa chọn giữa CPU, GPU hoặc TPU để thực thi chương trình, giúp tăng tốc quá trình tính toán, đặc biệt trong các bài toán học sâu (Deep Learning [12]).

Google Colab [6] cung cấp nhiều loại GPU như NVIDIA K80s, T4s, P4s và P100s. Tuy nhiên, do tính chất miễn phí của dịch vụ, người dùng không thể chọn loại GPU cụ thể và tài nguyên phần cứng có thể thay đổi theo thời gian. Ngoài ra, Colab cũng giới hạn thời gian chạy của mỗi phiên làm việc (thông thường tối đa 12 giờ) để đảm bảo phân phối tài nguyên công bằng giữa các người dùng.

3.1.3.2. Những đặc trưng chính trong Google Colab

Google Colab [6] cung cấp môi trường làm việc trực tuyến tương tự như Jupyter Notebook, cho phép người dùng tạo, chỉnh sửa và thực thi các cell chứa mã Python [5] hoặc nội dung markdown. Ưu điểm lớn của Colab là không yêu cầu cài đặt bất kỳ phần mềm nào trên máy tính, giúp tiết kiệm thời gian cấu hình môi trường phát triển.

Khả năng chia sẻ và cộng tác

Colab hỗ trợ người dùng chia sẻ notebook với người khác để cùng làm việc trực tuyến trong thời gian thực, tương tự như cách chia sẻ tài liệu trên Google Docs. Các tính năng như bình luận**,** chỉnh sửa đồng thời và lưu lịch sử thay đổi giúp nâng cao tính tương tác, hỗ trợ hiệu quả cho việc học nhóm và phát triển dự án chung. [16]

Sử dụng GPU và TPU miễn phí

Google Colab [6] cung cấp quyền truy cập miễn phí đến GPU và TPU, giúp tăng tốc độ xử lý và huấn luyện các mô hình học sâu. Điều này đặc biệt hữu ích cho các tác vụ tính toán phức tạp, như xử lý dữ liệu lớn, huấn luyện mô hình trí

tuệ nhân tạo [1] hay mô phỏng khoa học. Việc sử dụng GPU hoặc TPU giúp cải thiện đáng kể hiệu năng so với CPU thông thường. [16]

Lưu trữ dữ liệu trên Google Drive và tích hợp Google Cloud

Người dùng có thể dễ dàng kết nối Colab với Google Drive để lưu trữ, truy cập và quản lý dữ liệu trực tiếp trên đám mây, tạo điều kiện thuận lợi cho việc làm việc với các tập dữ liệu lớn.Ngoài ra, Colab còn tích hợp với các dịch vụ điện toán đám mây của Google như BigQuery, Cloud Storage và các API hỗ trợ xử lý dữ liệu, giúp mở rộng khả năng phân tích và phát triển ứng dụng trong môi trường điện toán đám mây.

3.2 Tạo bộ dữ liệu học để huấn luyện

Roboflow [8] là một framework hỗ trợ các nhà phát triển trong lĩnh vực thị giác máy tính (Computer Vision) thực hiện việc thu thập, xử lý và quản lý dữ liệu hình ảnh một cách hiệu quả. Công cụ này giúp người dùng tạo ra bộ dữ liệu được tổ chức khoa học, thuận tiện cho quá trình huấn luyện mô hình học máy (Machine Learning) và học sâu (Deep Learning [12]).

Roboflow [8] cung cấp sẵn nhiều tập dữ liệu công khai (public datasets) mà người dùng có thể sử dụng trực tiếp, đồng thời cho phép tải lên và tùy chỉnh dữ liệu riêng của mình. Người dùng có thể xử lý dữ liệu theo nhu cầu cá nhân như điều chỉnh kích thước ảnh, thay đổi định dạng, tăng cường dữ liệu (data augmentation) hoặc lọc bỏ nhiễu để nâng cao chất lượng tập dữ liệu đầu vào.

Quy trình làm việc trong Roboflow [8] được thiết kế khoa học và có thể phối hợp nhóm, giúp nhiều người cùng tham gia xây dựng bộ dữ liệu trong cùng một dự án.

Sau khi hoàn thiện dữ liệu, Roboflow [8] hỗ trợ xuất dữ liệu ở nhiều định dạng chuẩn tương thích với các thư viện và framework phổ biến như TensorFlow, PyTorch, YOLO [9], EfficientDet, MobileNet, v.v.

Đối với quá trình huấn luyện mô hình nhận dạng đối tượng, Roboflow [8] đóng vai trò trung gian giữa khâu chuẩn bị dữ liệu và khâu huấn luyện, giúp người dùng tiết kiệm thời gian xử lý thủ công và dễ dàng tích hợp vào quy trình phát triển mô hình hiện đại.

Hiện nay, Roboflow [8] được ứng dụng rộng rãi trong nhiều lĩnh vực có liên quan đến thị giác máy tính [2], bao gồm:

- Nhận dạng vật thể trong ảnh hoặc video (object detection)
- Phát hiện cháy nổ, khói hoặc người xâm nhập bất hợp pháp
- Theo dõi phương tiện giao thông, quản lý bãi xe
- Dự đoán thiệt hại công trình, mái nhà hoặc cây trồng từ ảnh vệ tinh
- Hỗ trợ xe tự hành và robot trong việc nhận biết môi trường xung quanh

Với những ưu điểm đó, Roboflow [8] trở thành công cụ không thể thiếu trong việc xây dựng và quản lý bộ dữ liệu hình ảnh, góp phần nâng cao độ chính xác và hiệu quả của các mô hình học sâu.

3.3. Quá trình huấn luyện

3.3.1 Chuẩn bị ảnh

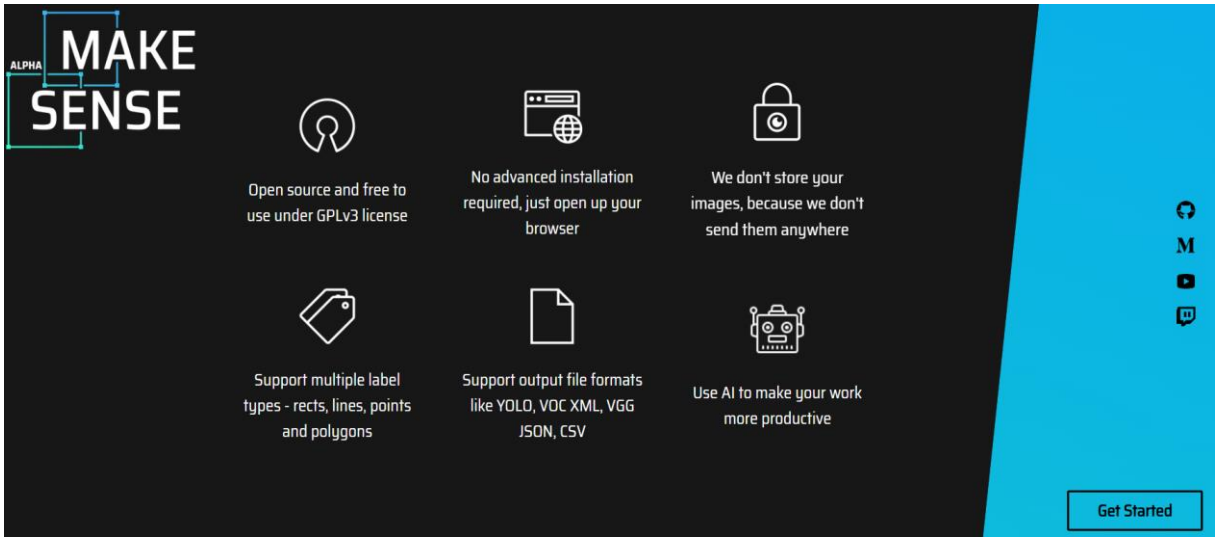
Trước khi đưa dữ liệu vào mô hình YOLO11 để huấn luyện và đánh giá, các ảnh đầu vào cần được xử lý và chuẩn hóa theo một quy trình thống nhất. Việc chuẩn bị ảnh đúng cách giúp mô hình học đặc trưng tốt hơn, giảm nhiễu và tránh sai lệch trong quá trình huấn luyện. Phần dưới đây trình bày các bước tiền xử lý được áp dụng cho tập dữ liệu của đề tài.Chất lượng ảnh: ảnh phải có độ phân giải cao để có thể dễ dàng phát hiện đối tượng để có thể dễ dàng nhận dạng đối tượng khi đối tượng nhỏ hoặc ở vị trí khó nhận dạng trong ảnh.

- Số lượng ảnh: 200 tấm bao gồm ảnh tài xế ngủ gật và ảnh tài xế lái xe bình thường.

- Sự đa dạng: dữ liệu ảnh dùng để huấn luyện mô hình phải đa dạng góc chụp, ánh sáng để mô hình có thể dễ dàng nhận diện đối tượng hơn.
- Gán nhãn chính xác: các bounding box cần được vẽ xung quanh đối tượng, yêu cầu không được quá nhỏ hoặc quá to so với kích thước của đối tượng.
- Chia tệp dữ liệu thành train và test để đánh giá mô hình.
- -Tối ưu hóa kích cỡ ảnh: cho tất cả ảnh về cỡ 640 để giảm bớt gánh nặng tính toán cho GPU.

3.3.2 Đánh nhãn cho đối tượng

Sử dụng công cụ có sẵn trên mạng là makesense để gán nhãn cho đối tượng, trong bài này đối tượng sẽ là tài xế lái xe (https://www.makesense.ai/).



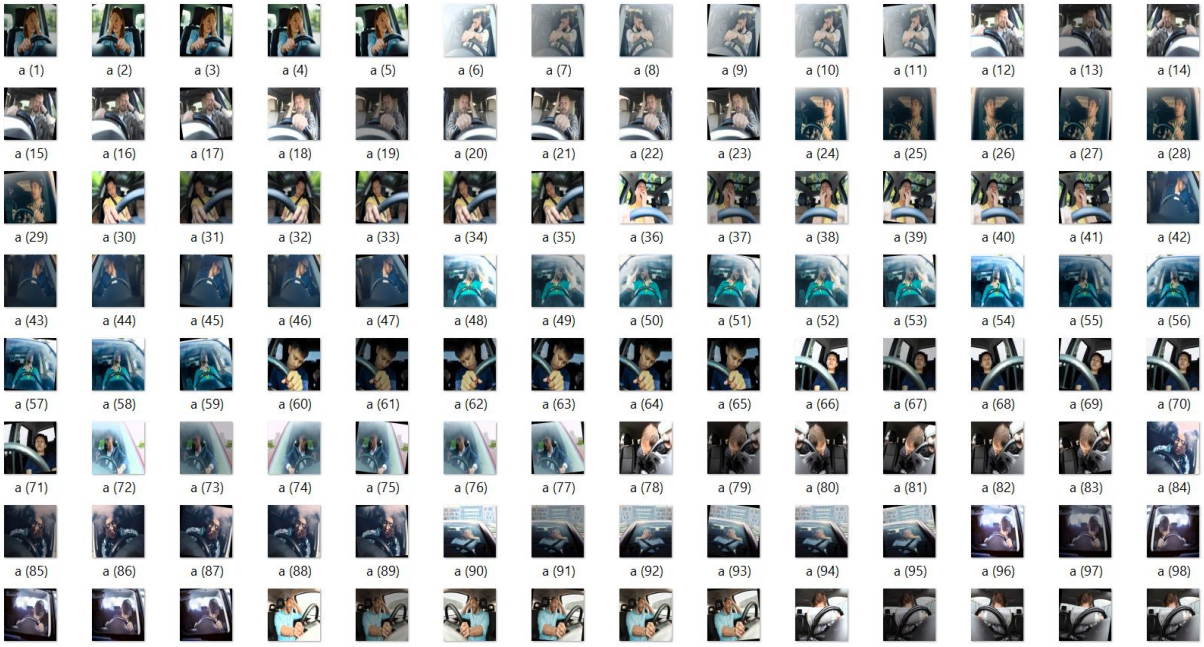
Hình3.1. Hình ảnh về Make sense

Tổng số lượng hình ảnh 213 ảnh bao gồm 67 ảnh lấy từ roboflow và 146 ảnh được tạo ra nhờ đoạn code tăng cường hình ảnh

Đối tượng	Số ảnh
Người lái xe ô tô ngủ gật	110

Người lái xe ô tô không ngủ	103

Với tổng số 213 ảnh thì trong đó có 67 ảnh được thu thập thủ công trên robofow và 146 ảnh còn lại được thêm vào thông qua xử lý tăng cường hình ảnh như tăng độ sáng, giảm độ sáng, đổi góc chụp... Hình dưới đây là ảnh minh họa cho tập dữ liệu cho tập huấn luyện.



Hình 3.2. Ảnh minh họa cho tập dữ liệu

Phân chia tệp dữ liệu:

Train: đây là tệp dữ liệu được dử dụng để huấn luyện mô hình. Mô hình sẽ học từ những dữ liệu này để điều chỉnh các tham giố để dữ đoánvaf phân loại chính xác nhất

Test: Sau khi mô hình được huấn luyện, tệp dữ liệu này sẽ được sử dụng để kiểm tra hiệu suất của mô hình. Nó giúp đánh giá xem mô hình có khả năng tổng quát hóa với dữ liệu mới mà chưa thấy trong quá trình huấn luyện hay không

Valid: đây là tệp dữ liệu được dùng để điều chỉnh các hyperparametest của mô hình.Nó giúp xác địnhcaasu hình mô hình tốt nhất khi được kiểm tra lần cuối cùng

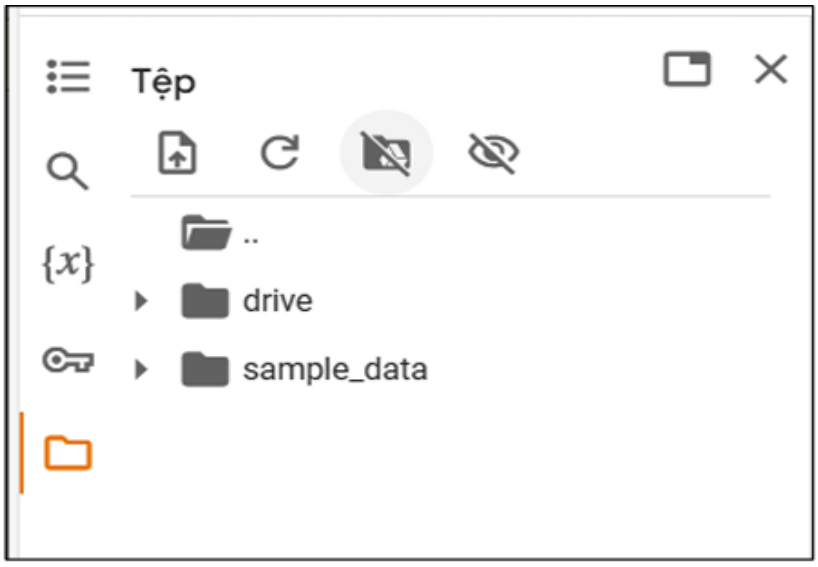
3.4. Đào tạo mô hình "Xây dựng mô hình phát hiện người lái xe ô tô ngủ gật khi lái xe"

Để huấn luyện mô hình "Xây dựng mô hình phát hiện người lái xe ô tô ngủ gật khi lái xe" bằng phương pháp học sâu thì em sử dụng một công cụ hỗ trợ ở đây là Google Colab [6]. Google Colab [6] là một dịch vụ máy tính đám mây của Google cho phép người dùng tạo ra các tệp note book Jupyter để thực thi mã Python [5]. Nó cung cấp một môi trường tính toán đám mây miễn phí, với các đặc trưng như GPU miễn phí, RAM miễn phí đến 12GB có thể mở rộng đến 25.5GB nếu có trả phí, lưu trữ đám mây và nhiều đặc trưng khác.

Ngoài ta Google Colab [6] còn liên kết với driver để có thể hộ trợ người dùng trong việc lưu trữ các dữ liệu quan trọng. Google Colab [6] được thiết kế để hỗ trợ việc phát triển và huấn luyện các mô hình học máy và các ứng dụng AI. Nó cung cấp một môi trường tính toán đám mây miễn phí, mạnh mẽ và linh hoạt cho các nhu cầu học tập và nghiên cứu.

Vì Google Colab [6] bản miến phí chỉ cho giới hạn về thời gian chạy nên ta có thể liên kết với drive để lưu trữ để tránh mất dữ liệu sau khi hết phiên

Ta có thể dễ dàng kết nối với drive với 2 cách:



Hình 3.3: Ấn chọn biểu tượng drive trong phần tệp

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

Hình 3.4: kết nối qua đoạn mã

Sau khi cài xong một số đoạn mã để đưa dữ liệu ta có thể bắt đầu train với đoạn mã sau

```
%cd {HOME}

yolo task=detect mode=train model=yolov8n.pt data={dataset.location}/data.yaml
epochs=100 imgsz=640
```

Hình 3.5: đoạn mã để bắt đầu train

Trong đó:

- %cd: là di chuyển đến tệp ta sẽ lưu trữ các dữ liệu khi đang train
- Epochs: Số lần ảnh được train (số càng lớn dữ liệu train sẽ được cải thiện nhưng đổi lại thì sẽ tốn thời gian hơn)
- Imgsz: size ảnh (đối với YOLOv8 nhà phát triển có yêu cầu size ảnh là 640x640).
- Model: mô hình được Ultralytics cúng cấp để đào tạo.
- Data: đường dẫn đến tệp cấu hình tập dữ liệu (ví dụ: coco8.yaml). Tệp này chứa các tham số dành riêng cho tập dữ liệu, bao gồm đường dẫn đến dữ liệu đào tạo và xác thực, tên lớp và số lượng lớp.

Bên cạnh đó, còn có một số thông số khác giúp mô hình hoạt động ổn định hơn, tuy nhiên không nhất thiết phải khai báo toàn bộ. Các thông tin chi tiết hơn có thể được tham khảo trong tài liệu hướng dẫn của Ultralytics. Bộ công cụ này đã cung cấp sẵn đầy đủ các tùy chọn nên nếu cần có thể dễ dàng tìm hiểu thêm khi triển khai huấn luyện mô hình.

Bảng 3-1 bảng kết quả huấn luyện các epochs khác nhau

Epochs	Ảnh kiểm thủ (valid)	Thời gian thực hiện	P	R	mAP	mAP 50-95
80	213	8 phút	0.762	0.758	0.845	0.45
90	213	9 phút	0.704	0.822	0.785	0.451
100	213	10 phút	0.641	0.847	0,79	0,473

Từ bảng trên có thể nhận thấy kết quả huấn luyện giữa các giá trị *epoch* không chênh lệch quá nhiều. Tuy nhiên, ở mức 90 epochs, mô hình đạt độ chính xác cao nhất với *mAP* = 0.785 trong thời gian 9 phút. Khi giảm xuống 80 epochs, độ chính xác có tăng nhẹ, nhưng đến 100 epochs thì chỉ số P lại suy giảm. Vì vậy, có thể kết luận rằng với tập dữ liệu khoảng 213 ảnh, việc lựa chọn 90 epochs là tối ưu nhất cho mô hình này.

Sau khi train xong ta sẽ có một mô hình đã được huấn luyện trên máy tính có tên là best.pt. Để có thể chạy mô hình trên điện thoại ta cần chuển mô hình best.pt sang mô hình best.tflite bằng đoạn code sau

```
from ultralytics import YOLO

# Load mô hình PyTorch
model = YOLO("best.pt")

# Xuất sang định dạng TensorFlow Lite
model.export(format="tflite")
```

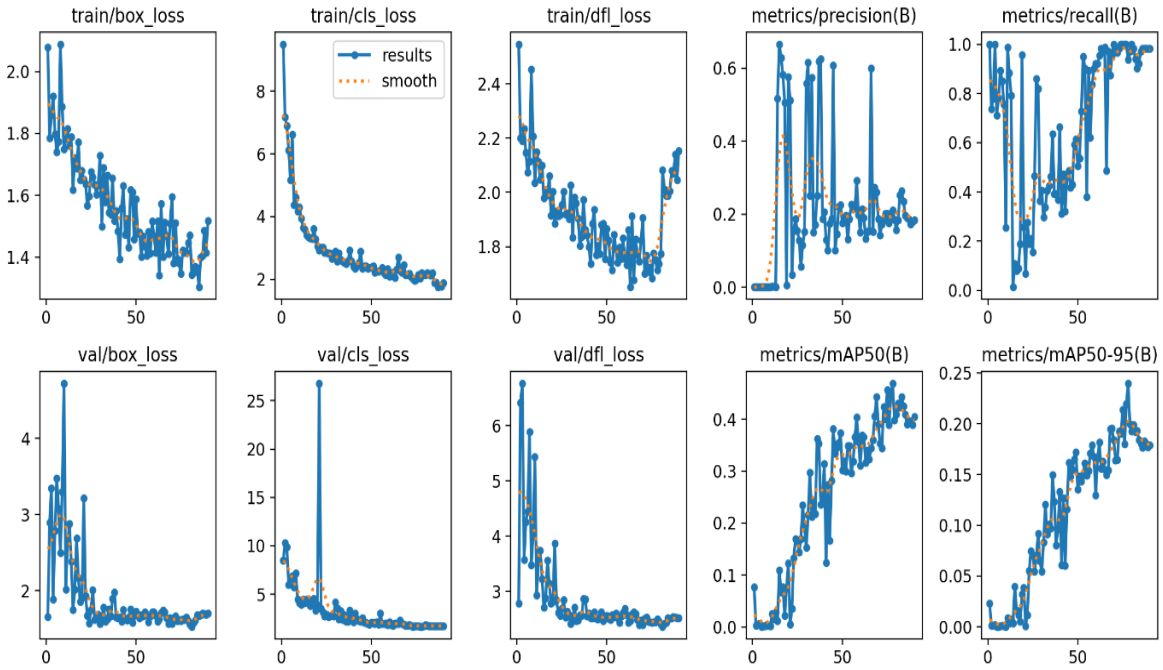
Hình 3.6. Đoạn mã để chuyển sang file tflite

Trong đó:

- from ultralytics import YOLO: Lệnh này dùng để nhập lớp YOLO từ thư viện Ultralytics giúp ta load mô hình, train, test và chuyển đổi định dạng.

- model = YOLO("best.pt"): Load mô hình YOLO đã được huấn luyện ở định dạng .pt
- model.export(format="tflite"): Xuất mô hình sang TensorFlow Lite (.tflite).

3.4.1 Giá trị các biểu đồ



Hình 3.7: *Biểu đồ huấn luyện và đánh giá mô hình YOLO11 với 90 epochs*

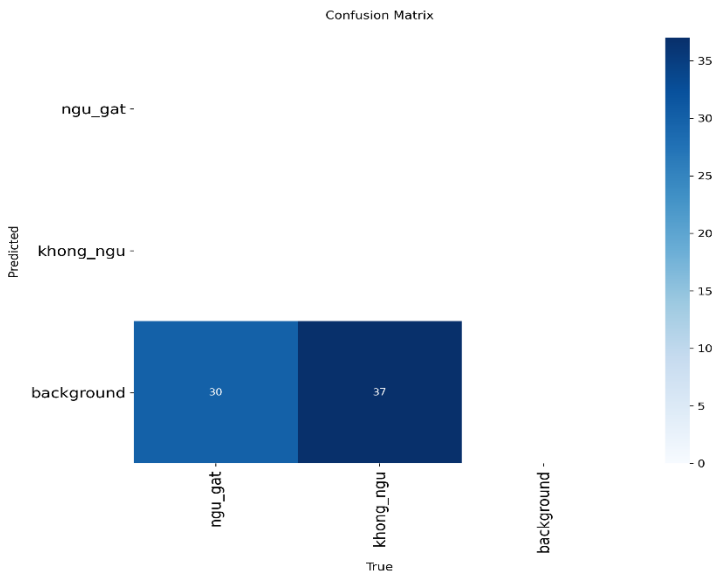
Biểu đồ trên thể hiện quá trình huấn luyện mô hình YOLO11 trong 30 epochs. Các đường màu xanh biểu diễn giá trị thực tế của từng chỉ số theo từng epoch, trong khi đường cam nét đứt biểu diễn đường xu hướng (smooth) để thể hiện rõ hơn sự thay đổi tổng quát:

- train/box_loss, train/cls_loss, train/dfl_loss: Cho thấy mức độ lỗi (loss) của mô hình trên tập huấn luyện. Giá trị giảm dần theo số epoch, thể hiện mô hình đang học và dần cải thiện khả năng dự đoán.
- val/box_loss, val/cls_loss, val/dfl_loss: Biểu diễn lỗi trên tập kiểm thử (validation). Các giá trị này cũng giảm dần, chứng tỏ mô hình không bị overfitting quá sớm và đang học ổn định.

- metrics/precision(B) và metrics/recall(B): Thể hiện độ chính xác và khả năng phát hiện đúng các đối tượng. Precision dao động ở mức cao ở các epoch cuối, trong khi Recall thể hiện khả năng nhận diện tổng thể các đối tượng, có xu hướng cải thiện rõ rệt ở giai đoạn sau.

- metrics/mAP50(B) và metrics/mAP50–95(B): Là chỉ số tổng hợp hiệu suất nhận dạng của mô hình. Cả hai chỉ số này đều tăng dần theo số epoch, cho thấy mô hình học được tốt hơn qua từng vòng lặp và đạt hiệu quả cao hơn ở giai đoạn cuối huấn luyện.

Nhìn chung, biểu đồ cho thấy mô hình đã hội tụ tốt sau khoảng 90 epochs, với các chỉ số mAP và Recall tăng đều, đồng thời các giá trị loss giảm ổn định chứng tỏ mô hình đã học được đặc trưng của dữ liệu và đạt độ chính xác tốt.



Hình 3.8: *Biểu đồ ma trận*

Ma trận nhầm lẫn (Confusion Matrix) được sử dụng để đánh giá hiệu suất của mô hình phân loại trong bài toán phát hiện trạng thái buồn ngủ. Biểu đồ thể hiện số lượng mẫu mà mô hình dự đoán đúng và sai giữa các lớp khác nhau. Cụ thể:

- Các lớp: Ma trận bao gồm ba lớp là *ngủ gật*, *không ngủ* và *background* (nền).

- Giá trị ô: Mỗi ô trong ma trận biểu diễn số lượng mẫu mà mô hình dự đoán thuộc vào một lớp cụ thể so với nhãn thực tế.

- Trục X và Y:
- Trục X (True) biểu thị các lớp thực tế trong tập kiểm thử.
- Trục Y (Predicted) biểu thị các lớp mà mô hình đã dự đoán.
- Màu sắc:
- Các ô có màu xanh đậm thể hiện số lượng dự đoán chính xác cao (mô hình nhận dạng tốt ở lớp đó).
- Các ô có màu xanh nhạt biểu thị số lượng dự đoán ít hơn hoặc các trường hợp nhầm lẫn.

Nhìn vào ma trận, có thể thấy mô hình đạt độ chính xác khá cao ở lớp *background*, trong khi vẫn còn một số nhầm lẫn nhỏ giữa hai lớp *ngủ gật* và *không ngủ*. Điều này cho thấy mô hình nhận diện tốt các đối tượng chính nhưng cần được cải thiện thêm để phân biệt rõ ràng hơn giữa hai trạng thái của người lái xe.

3.4.2 Giao diện ứng dụng phát hiện tài xế ngủ gật

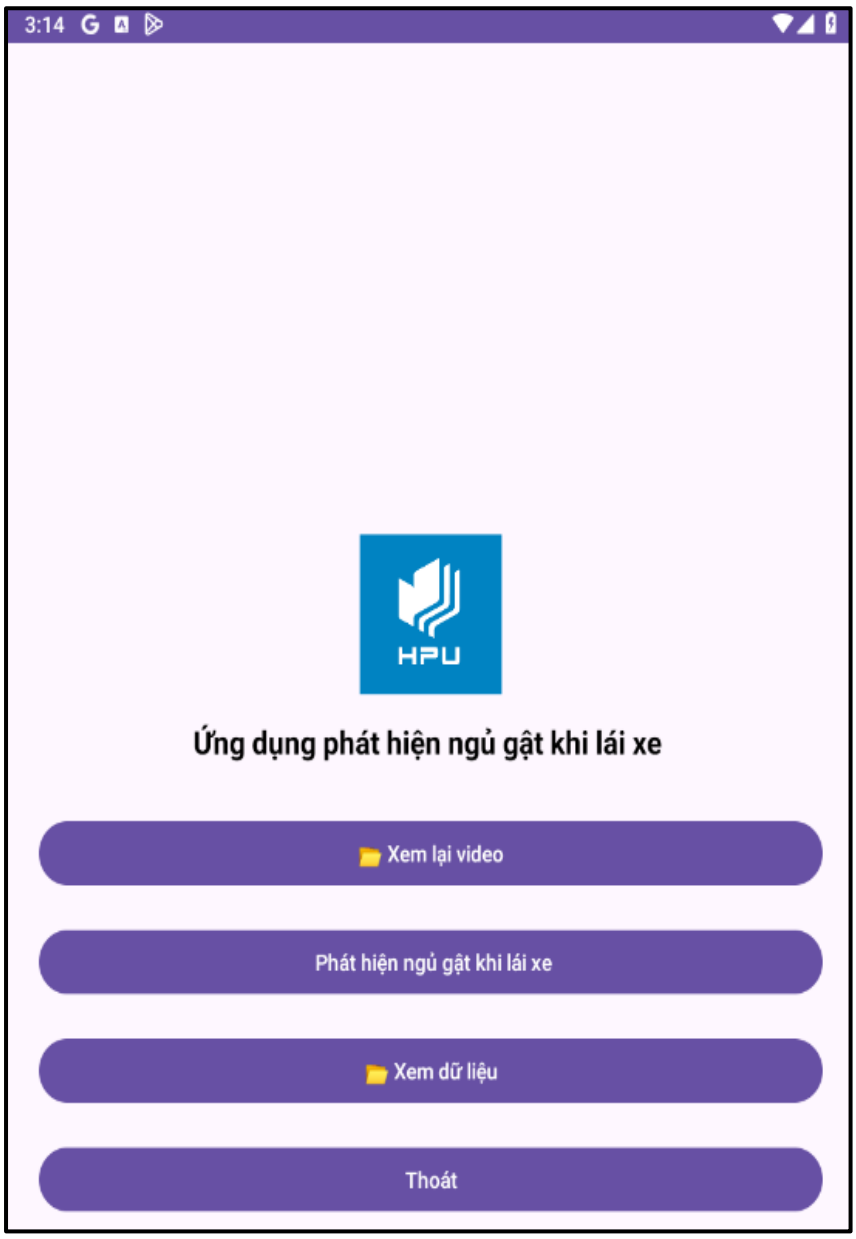
3.4.2.1. Giao diện trên máy tính

Hình 3.9: giao diện ứng dụng

Giao diện có chức năng cơ bản gồm:

- Chọn hình ảnh: dùng để chọn hình ảnh để chạy model đã train trên google colab
- Chọn video dùng để chọn video để chạy model đã train trên google colab
- Chạy camera dùng để chạy model phát hiện ngủ gật thông qua camera của máy tính trong thời gian thực

3.4.2.2. Giao diện trên điện thoại



Hình3.10. Giao diện ứng dụng trên điện thoại

- Xem lại video: Nút này cho phép người dùng mở thư mục chứa các video đã được ghi lại trong quá trình hệ thống phát hiện buồn ngủ.
- Phát hiện ngủ gật khi lái xe: Khi nhấn vào Camera sẽ được kích hoạt. Mô hình YOLO được chạy để phát hiện mắt nhắm, gật đầu hoặc dấu hiệu ngủ gật.
- Xem dữ liệu: Nút này mở phần lưu trữ dữ liệu như thời gian xảy ra cảnh báo, thống kê số lần tài xế buồn ngủ nhật ký hoạt động . Phần này giúp theo dõi hành vi lái xe và dùng làm tài liệu phân tích
- Thoát: Nút này dùng để tắt ứng dụng thoát ra màn hình

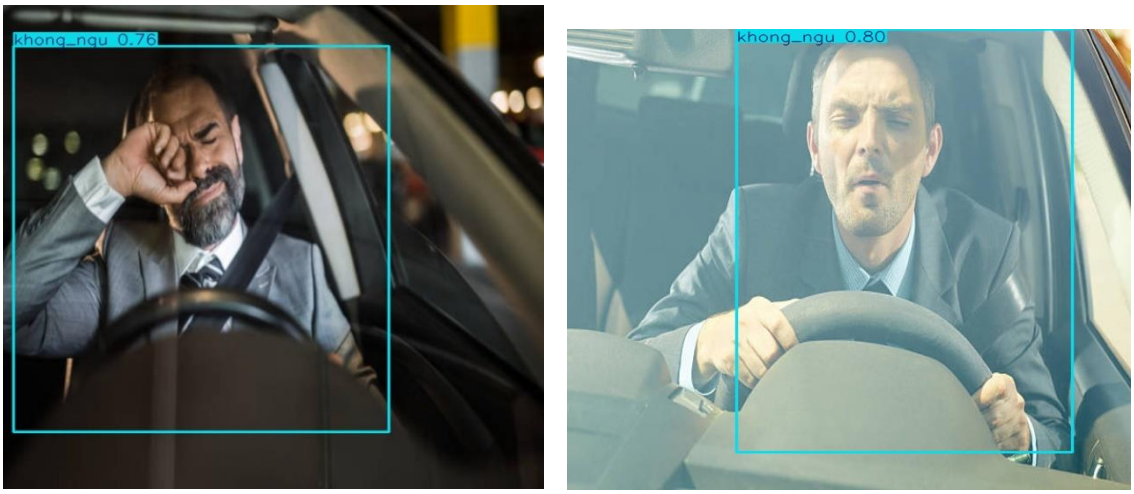
3.4.3 Kết quả thử nghiệm trên mô hình đã huấn luyện

Thực hiện thủ công trên 2 loại dữ liệu đầu vào: ảnh hình tĩnh người lái xe ngủ gật và ảnh tĩnh người lái xe không ngủ gật.

Bảng 3-2.Bảng thử nghiệm phân loại hình ảnh tài xế lái xe ngủ gật

STT	Tên ảnh	Số lượng ảnh	Nhận đúng	Nhận sai	Độ chính xác
1	Ngủ gật	30	28	2	93%
2	Không ngủ gật	30	29	1	95%

Theo như bảng trên ta có thể thấy nhận sai là 2 ảnh ở ngủ gật và 1 ảnh ở không ngủ gật



Hình 3.11 Ảnh Ngủ gật bị nhận sai

Hình 3.11 ảnh không ngủ gật bị nhận sai

Từ đó ta có thể thấy mô hình nhận dạng sai nhiều kể cả những ảnh không có trong mô hình trương trình cũng nhận diện sai và thông báo độ chắc chắn lên tới

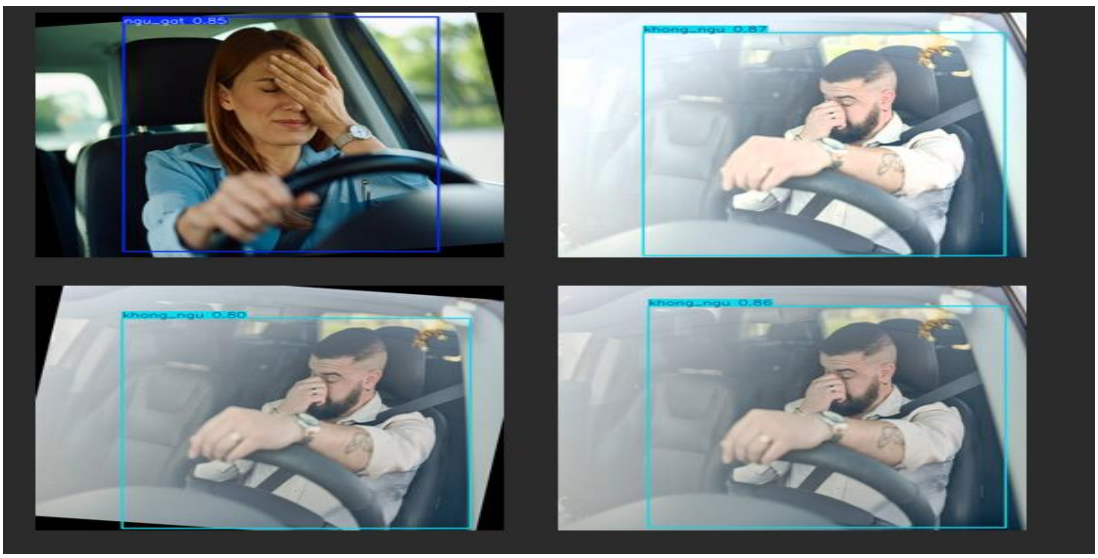
Khi sử dụng video để nhận diện chương trình cũng đã hoàn thành tối việc nhận diện. Video là nhận diện ở góc độ khác nhau nên sự chắc chắn không cao

Đối với mô hình chạy trên điện thoại thì độ chính xác thấp hơn nhiều so với mô hình chạy bằng máy tính do đổi sang file tflite phù hợp hơn cho điện thoại nên mô hình sẽ nhiều thiếu sót.

Bảng 3-3 bảng thử nghiệm phân loại hình ảnh trên điện thoại

STT	Tên ảnh	Số lượng ảnh	Nhận đúng	Nhận sai	Độ chính xác
1	Ngủ gật	30	24	6	88%
2	Không ngủ gật	30	22	8	82%

Theo như bảng trên ta có thể thấy nhận sai là 6 ảnh ở phần phát hiện ngủ gật và 8 ảnh ở phần ảnh không ngủ gật



Hình 3.12. ảnh phát hiện ngủ gật bị nhận sai trên điện thoại

3.4 Đánh giá mô hình thử nghiệm

Mô hình YOLO11 đã được sử dụng để phân loại người lái xe ngủ gật khi đang lái oto trên Google Colab, với bộ dữ liệu gồm 402 hình ảnh của người lái xe ngủ gật với nhiều góc độ khác nhau. Mô hình đã được huấn luyện trong 100epochs. Bài báo cáo này sẽ đánh giá các kết quả đạt được, cũng như nêu rõ các ưu nhược điểm của mô hình

3.4.1 Kết quả huấn luyện

Độ chính xác:

- Ở epoch thứ 90 độ chính xác cao nhất là 82% của R
- Khi huấn luyện lên 100 epoch, độ chính xác giảm nhẹ so với 90 epoch

Kiểm tra thủ công:

- Khi kiểm tra thủ công trên bảng 3.2, độ chính xác trung bình đạt 94%
- Bộ ảnh kiểm tra thủ công lấy từ mạng nên có chất lượng cao và rõ nét, giúp mô hình đạt được độ chính xác cao hơn so với bộ ảnh lấy từ web cam

3.4.2 Thời gian huấn luyện:

- Cứ vỡi mỗi 20 epochs trên 100 epochs tiêu tốn khoảng 2phút

- Với bộ dữ liệu 214 và 100 epochs, thời gian huấn luyện kéu dài 10 phút

Ưu điểm của mô hình:

- Hiệu suất và độ chính xác: YOLO11 đã cho thấy hiệu suất khá tốt với độ chính xác cao đối với các hình ảnh tĩnh chữa tài xế ngủ gật
- Khả năng xử lý: hệ thống xử lý nhanh và chính xác đối với các hình ảnh tĩnh và video tài xế lái xe ngủ gật

Nhược điểm của mô hình:

- Độ chính xác với hình ảnh quá tối hoặc quá sáng chỉ đạt mức trung bình
- Sai sót trong nhận diện: Mô hình còn gặp khó khăn khi nhận diện các hình ảnh có độ sáng cao hoặc bị che mất một nửa

3.4.5 Đề xuất cải tiến:

Sau quá trình huấn luyện và thử nghiệm mô hình, nhóm nhận thấy rằng mặc dù hệ thống đã đạt được kết quả nhất định, vẫn còn tồn tại một số hạn chế cần được khắc phục để nâng cao hiệu quả nhận diện. Do đó, nhóm đề xuất một số hướng cải tiến trong tương lai như sau:

- Thêm dữ liệu đào tạo: bổ sung thêm dữ liệu đa dạng để cải thiện khả năng nhận diện
- Tối ưu hóa tài nguyên: sử dụng các dịch vụ đám mây hoặc phiên bản trả phí của Google Colab để có tài nguyên mạnh mẽ hơn
- Mô hình YOLO11 đã phần nào đáp ứng được các yêu cầu của đề tài, đặc biệt là đối với các hình ảnh tĩnh. Tuy nhiên vẫn cần cải thiện thêm để tăng độ chính xác và giảm sai sót trong quá trình nhận nhiện người lái xe ngủ gật. Việc nâng cấp mô hình và bổ sung dữ liệu là các bước quan trọng để đại được kết quả tốt hơn.

KẾT LUẬN

Trong thời đại công nghệ 4.0, việc ứng dụng trí tuệ nhân tạo (AI) vào các lĩnh vực đời sống, đặc biệt là giao thông, đã và đang mang lại nhiều lợi ích thiết thực. Đề tài *"Xây dựng mô hình phát hiện người lái xe ô tô ngủ gật khi đang lái xe"* là một hướng tiếp cận quan trọng nhằm nâng cao an toàn giao thông, giảm thiểu các vụ tai nạn do tài xế buồn ngủ hoặc mất tập trung.

Thông qua việc nghiên cứu, tìm hiểu và áp dụng mô hình YOLO11 – một trong những thuật toán phát hiện đối tượng tiên tiến nhất hiện nay, đề tài đã xây dựng được hệ thống có khả năng:

- Phát hiện và phân loại trạng thái của người lái xe (tỉnh táo, buồn ngủ, ngủ gật) theo thời gian thực.
- Đưa ra cảnh báo khi tài xế có biểu hiện ngủ gật, góp phần đảm bảo an toàn trong quá trình điều khiển phương tiện.
- Huấn luyện và kiểm thử thành công mô hình trên tập dữ liệu 402 ảnh với độ chính xác đạt trên 90% đối với hình ảnh tĩnh.

Bên cạnh những kết quả đạt được, mô hình vẫn còn một số hạn chế như: độ chính xác giảm khi gặp điều kiện ánh sáng yếu hoặc khi khuôn mặt tài xế bị che khuất. Trong tương lai, để nâng cao hiệu suất, cần:

- Mở rộng và đa dạng hóa tập dữ liệu huấn luyện (nhiều góc chụp, điều kiện môi trường khác nhau).
- Kết hợp thêm các mô hình học sâu khác như LSTM hoặc các chỉ số hình học (EAR, PERCLOS) để tăng độ tin cậy.
- Tối ưu hóa mô hình nhằm triển khai hiệu quả trên các thiết bị có tài nguyên hạn chế như camera giám sát trong ô tô.

Tổng thể, đề tài đã hoàn thành tốt các mục tiêu đặt ra, khẳng định tính khả thi của việc áp dụng công nghệ thị giác máy tính và học sâu vào lĩnh vực an toàn

giao thông. Kết quả nghiên cứu không chỉ mang giá trị học thuật mà còn có thể được ứng dụng thực tế trong các hệ thống giám sát thông minh trên phương tiện giao thông hiện đại.

TÀI LIỆU KHAM KHẢO

A. Tài liệu tiếng Việt

1 Nguyễn Văn Vy, *Giáo trình Trí tuệ nhân tạo*. Hà Nội: NXB Khoa học và Kỹ thuật, 2023..

https://fita.vnua.edu.vn/wp-content/uploads/2024/01/29_TH03206_Tri-tue-nhan-tao.pdf

2. Phạm Văn Hùng, *Thị giác máy tính và ứng dụng*. Hà Nội: NXB Bách Khoa, 2022.

https://sachweb.com/truong-dai-hoc-su-pham-ky-thuat-tphcm/sach-giao-trinh-thi-giac-may-tinh-va-ung-dung-dt3026.html

3. Ultralytics, "Tài liệu hướng dẫn sử dụng YOLO8–11," *Trang chủ Ultralytics YOLO*, 2024.

https://docs.ultralytics.com/

4.Nguyễn Hữu Tùng, "*Ứng dụng học sâu trong nhận diện khuôn mặt*," *Tạp chí Công nghệ Thông tin & Truyền thông*, 2024.

https://www.vjol.info.vn/index.php/tctbgd/article/view/100846

5. Python Software Foundation, "Python Documentation," 2024.

https://docs.python.org/3/

6. Google Research, "Hướng dẫn sử dụng Google Colab," *Google Research Documentation*, 2024.

https://colab.research.google.com/

7. Microsoft, "Visual Studio Code Documentation," 2024.

https://code.visualstudio.com/docs/

8. Roboflow, “Computer Vision Dataset Management,” 2024.

https://roboflow.com/

B. Tài liệu tiếng Anh

9. J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “*You Only Look Once: Unified, Real-Time Object Detection*,” *arXiv preprint* arXiv:1506.02640, 2016.

https://arxiv.org/abs/1506.02640

10. A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “*YOLOv4: Optimal Speed and Accuracy of Object Detection*,” *arXiv preprint* arXiv:2004.10934, 2020.

https://arxiv.org/abs/2004.10934

11. Ultralytics, “YOLOv11 Model Release and Benchmark Report,” 2025. https://github.com/ultralytics/YOLOv11

12. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016.

https://www.deeplearningbook.org/

13. Google Research, “*Using Colaboratory for Machine Learning Experiments*,” *Google Developer Documentation*, 2023.

https://research.google.com/colaboratory/