

TRƯỜNG ĐẠI HỌC ĐÀ LẠT
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO KẾT QUẢ THỰC NGHIỆM
MÔN PHƯƠNG PHÁP HỌC MÁY
XÂY DỰNG ỨNG DỤNG NHẬN DIỆN BIỂN SỐ XE

Giáo viên:	Tạ Hoàng Thắng
Sinh viên thực hiện:	2115209 – Nguyễn Huy Hiếu
	2115287 – Hoàng Vũ Minh Trung
	2113024 – Nông Đức Trí
	2112997 – Đỗ Thanh Liêm

Đà Lạt, ngày 29, tháng 10, năm 2024

[illegible]

[Ký tên và ghi rõ họ tên]

ĐỀ CƯƠNG THỰC HIỆN BÁO CÁO

Tên đề tài: Xây dựng ứng dụng nhận diện biển số xe

Sinh viên thực hiện:

STT	Họ và Tên	MSSV	Lớp	Email Liên Hệ
1	Hoàng Vũ Minh Trung	2115287	CTK45A	2115287@dlu.edu.vn
2	Nguyễn Huy Hiếu	2115209	CTK45A	2115209@dlu.edu.vn
3	Nông Đức Trí	2113024	CTK45A	2113024@dlu.edu.vn
4	Đỗ Thanh Liêm	2112997	CTK45A	2112997@dlu.edu.vn

Giáo viên: Tạ Hoàng Thắng

1. Mục tiêu đề tài

- Hiểu về mô hình Yolov8, Inception Resnet V2, VGG16 cho phát hiện xe và biển số xe.
- Hiểu về mô hình PaddleOCR cho nhận dạng ký tự trong hình ảnh.
- Áp dụng được các framework UI để xây dựng giao diện ứng dụng.
- Áp dụng Tkinter và Utils để xây dựng ứng dụng hoàn chỉnh cho desktop.

2. Nội dung đề tài

- Tìm hiểu và bổ sung kiến thức về Yolov8, Inception Resnet V2, VGG16 và PaddleOCR.
- Phân tích yêu cầu, xác định các chức năng của Desktop App.
- Thiết kế Api.
- Các giai đoạn nhận diện biển số xe.
- Viết báo cáo tổng kết.

3. Phần mềm và công cụ sử dụng

- Ngôn ngữ lập trình: Python.

- Hệ quản trị cơ sở dữ liệu: SQLServer.
- Công cụ sử dụng: Visual Studio Code, Google Colab.
- Source Code: GitHub.

4. Kết quả dự kiến đạt được

- Nắm được mục tiêu và cách thức hoạt động của mô hình học máy Yolov8, Inception Resnet V2, VGG16.
- Nắm được mục tiêu và cách thức hoạt động của mô hình nhận dạng ký tự quang học PaddleOCR.
- Xây dựng ứng dụng dựa trên kết quả mô hình Yolov8 và PaddleOCR bằng Tkinter và Utils.
- Hoàn thiện khả năng thuyết trình, viết báo cáo, khả năng tìm kiếm và tổng hợp tài liệu, nghiên cứu khoa học.

MỤC LỤC

Mục lục

NHẬN XÉT CỦA GIÁO VIÊN	2
ĐỀ CƯƠNG THỰC HIỆN ĐỒ ÁN.....	3
MỤC LỤC	5
LỜI MỞ ĐẦU.....	Error! Bookmark not defined.
A. Tính cấp thiết của đề tài.....	Error! Bookmark not defined.
B. Mục tiêu.....	Error! Bookmark not defined.
C. Nhiệm vụ	Error! Bookmark not defined.
D. Đối tượng và phạm vi nghiên cứu..	Error! Bookmark not defined.
E. Phương pháp nghiên cứu	Error! Bookmark not defined.
CHƯƠNG 1: TỔNG QUAN VỀ NGHIÊN CỨU.....	8
1.1 Tổng quan về đề tài.....	8
1.1.1 Giới thiệu về quản lý bãi đỗ xe.....	8
1.1.2 Trí tuệ nhân tạo trong quản lý bãi đỗ xe.....	8
1.1.3 Lợi ích AI trong bãi đỗ xe.....	9
1.2 Tổng quan về nghiên cứu	9
1.2.1 Nghiên cứu liên quan.....	9
1.2.2 Nghiên cứu về ứng dụng.....	11
Chương 2. CƠ SỞ LÝ THUYẾT.....	13
2.1 Yolov8.....	13
2.1.1 Tổng Quan về mô hình Yolov8.....	13
2.1.2 Sử dụng mô hình YOLOv8 để phát hiện đối tượng	13
2.1.3 Tại sao sử dụng mô hình Yolov8.....	15
2.1.4 Cách thức hoạt động của yolov8	16

2.1.5 Mục tiêu của mô hình Yolov8.....	19
2.1.6 Quá trình thực hiện của mô hình Yolov8.....	19
2.1.7 Công thức toán học và Mã giả của mô hình Yolov8.....	20
2.2 PaddleOCR	21
2.2.1 Tổng quan về mô hình PaddleOCR.....	21
2.2.2 Sử dụng mô hình PaddleOCR để nhận dạng biển số xe.....	22
2.2.3 Tại sao sử dụng mô hình PaddleOCR.....	22
2.2.4 Cách thức hoạt động của PaddleOCR.....	23
2.2.5 Mục tiêu của mô hình PaddleOCR	23
2.2.5 Quá trình thực hiện của mô hình PaddleOCR.....	24
2.1.7 Công thức toán học và Mã giả của mô hình PaddleOCR.....	25
2.3 Tkinter và Utils	26
2.3.1 Tkinter.....	26
2.3.2 Utils.....	28
2.4 SQL Server.....	29
2.4.1 Đặc điểm chính của SQL Server	29
2.4.2 Lịch sử hình thành và phát triển.....	30
2.4.3 Ứng dụng của SQL Server.....	30
2.5 Các thư viện khác.....	31
2.5.1 OpenCV	31
2.5.2 CVZone.....	32
Chương 3. TRAIN MÔ HÌNH.....	34
3.1 Trước khi train mô hình.....	34
3.2 Quá trình train mô hình.....	34
3.2.1 Triển khai mô hình YoloV8	34

3.2.2	Sử dụng nền tảng hỗ trợ gắn dữ liệu hình ảnh.....	35
3.2.3	Xây dựng hệ thống nhận diện biển số xe	71
3.3	Sau khi train mô hình	75
3.4	Đánh giá mô hình.....	85

CHƯƠNG 1: TỔNG QUAN VỀ NGHIÊN CỨU

1.1 Tổng quan về đề tài

1.1.1 Giới thiệu về quản lý bãi đỗ xe

Quản lý bãi đỗ xe không chỉ là việc sắp xếp và giám sát các phương tiện, mà còn là một quá trình phức tạp đòi hỏi sự tích hợp của nhiều yếu tố khác nhau. Đầu tiên, nó bao gồm việc tích hợp các ngành công nghiệp khác nhau như công nghệ thông tin, quản lý môi trường, và quản lý kinh doanh để tạo ra một hệ thống bãi đỗ xe toàn diện và hiệu quả. Mục tiêu của quản lý bãi đỗ xe là tối ưu hóa không gian đỗ xe và nâng cao trải nghiệm người dùng, bằng cách giảm thiểu tình trạng ùn tắc và tạo điều kiện thuận lợi cho việc sử dụng bãi đỗ xe. Thêm vào đó, quản lý bãi đỗ xe cũng liên quan đến việc tạo ra trải nghiệm đỗ xe tiện lợi và an toàn cho người sử dụng. Quản lý bãi đỗ xe cũng đòi hỏi sự tương tác chặt chẽ với các cơ quan quản lý đô thị và cộng đồng địa phương. Việc hòa nhập và hỗ trợ cộng đồng địa phương không chỉ là yếu tố quan trọng trong việc xây dựng một hệ thống bãi đỗ xe hiệu quả mà còn giúp tạo ra lợi ích kép cho cả đô thị và cư dân. Cuối cùng, quản lý bãi đỗ xe cũng liên quan đến việc đánh giá và quản lý các rủi ro liên quan đến an ninh và an toàn. Các biện pháp an ninh và an toàn cần được thiết lập và thực hiện để đảm bảo môi trường bãi đỗ xe an toàn và bảo mật cho người sử dụng và cộng đồng địa phương. Tóm lại, quản lý bãi đỗ xe là một quá trình phức tạp và đa chiều, đòi hỏi sự tích hợp chặt chẽ của nhiều yếu tố khác nhau để tạo ra một hệ thống bãi đỗ xe bền vững và hiệu quả cho người sử dụng và cộng đồng địa phương.

1.1.2 Trí tuệ nhân tạo trong quản lý bãi đỗ xe

Trí tuệ nhân tạo (AI) là một lĩnh vực công nghệ mà máy tính và hệ thống máy tính được lập trình để thực hiện các nhiệm vụ thông minh mà trước đây chỉ có con người mới có khả năng thực hiện. Trong lĩnh vực quản lý bãi đỗ xe, AI đã mang lại nhiều ứng dụng đổi mới nhằm cải thiện trải nghiệm của người dùng và tối ưu hóa hoạt động quản lý bãi đỗ. AI được sử dụng để phát triển các hệ thống đặt chỗ thông minh, giúp dự đoán nhu cầu đỗ xe và tạo ra các trải nghiệm đặt chỗ cá nhân hóa và thuận tiện. Nó cũng được tích hợp vào các hệ thống tự động hướng dẫn và quản lý lưu lượng xe, giúp điều phối các phương tiện vào và ra khỏi bãi đỗ một cách hiệu quả. AI cũng giúp dự đoán xu hướng và nhu cầu đỗ xe, từ đó giúp các nhà quản lý lập kế hoạch vận hành và điều chỉnh

giá cả một cách hợp lý. Cuối cùng, AI có thể tạo ra các trải nghiệm đỗ xe cá nhân hóa cho người dùng bằng cách đề xuất các bãi đỗ gần nhất và cung cấp thông tin về tình trạng đỗ xe, phù hợp với thói quen và yêu cầu cụ thể của họ.

1.1.3 Lợi ích AI trong bãi đỗ xe

Trong thời đại công nghệ ngày nay, công nghệ nhận diện biển số xe đã trở thành một công cụ quan trọng trong việc quản lý bãi đỗ xe tại các thành phố và khu vực đông đúc. Công nghệ này sử dụng các thuật toán và hệ thống máy tính để nhận diện và xác định các biển số xe, từ đó cho phép tự động xác định danh tính của xe. Ứng dụng chính của công nghệ nhận diện biển số xe trong quản lý bãi đỗ là tạo ra một hệ thống quản lý đỗ xe tự động và chính xác. Thay vì phải sử dụng các phương pháp quản lý truyền thống như vé giấy hoặc thẻ từ, công nghệ nhận diện biển số xe cho phép xe chỉ cần đi vào phạm vi của camera hoặc thiết bị nhận diện, và hệ thống sẽ tự động nhận diện và ghi nhận thông tin về xe, bao gồm thời gian vào và ra. Việc áp dụng công nghệ nhận diện biển số xe trong quản lý bãi đỗ xe mang lại nhiều lợi ích, bao gồm tính chính xác cao, tiết kiệm thời gian, và ngăn chặn được các trường hợp gian lận đỗ xe. Ngoài ra, nó cũng tạo ra một môi trường đỗ xe hiện đại và tiện lợi cho người sử dụng, giúp tăng cường hiệu suất quản lý và tối ưu hóa không gian đỗ xe hiệu quả hơn. Tuy nhiên, cũng cần phải xem xét các vấn đề về quyền riêng tư và an ninh thông tin khi triển khai công nghệ nhận diện biển số xe trong các bãi đỗ.

1.2 Tổng quan về nghiên cứu

1.2.1 Nghiên cứu liên quan

Trong một bài viết về ứng dụng công nghệ trong kinh doanh của Đại học Văn Hiến hợp tác với Tổng cục Thống kê đã chỉ ra những tác động tích cực đối với nền kinh tế khi ứng dụng công nghệ vào quản lý bãi đỗ xe, không chỉ mang lại những kết quả tích cực mà còn giúp nâng cao hiệu quả giao thông và kinh tế đô thị của Việt Nam. Tổng quan về nội dung nghiên cứu như sau:

Trong bối cảnh Cách mạng Công nghiệp 4.0 đang lan tỏa rộng lớn trên nhiều lĩnh vực, quản lý bãi đỗ xe không chỉ được hưởng lợi mà còn phải đối mặt với áp lực cạnh tranh từ các quốc gia tiên tiến. Để đáp ứng được nhu cầu này, các bãi đỗ xe cần phải hành động kịp thời, tập trung vào việc áp dụng công nghệ và thực hiện quá trình

chuyển đổi số. Theo Tổng cục Thống kê, trong năm 2023, việc áp dụng công nghệ trong quản lý bãi đỗ xe tại các thành phố lớn của Việt Nam đã giúp giảm đáng kể tình trạng ùn tắc giao thông và tăng hiệu quả sử dụng không gian đỗ xe. Tuy nhiên, so với các quốc gia trong khu vực, Việt Nam vẫn cần nhiều cải tiến để đạt hiệu quả cao hơn trong quản lý đô thị. Để đạt được mục tiêu phát triển bền vững và hiệu quả, cần chú ý đến những giải pháp sau:

- Xây dựng và phát triển đội ngũ trí thức khoa học và công nghệ trong lĩnh vực quản lý đô thị: Đầu tư vào đào tạo và nâng cao trình độ cho đội ngũ cán bộ nghiên cứu. Ứng dụng nhanh chóng các kết quả nghiên cứu vào thực tiễn để nâng cao hiệu quả quản lý bãi đỗ xe.
- Phát triển, đổi mới hệ thống cơ quan nghiên cứu và chuyển giao công nghệ trong lĩnh vực giao thông đô thị: Nâng cao năng lực khoa học, công nghệ nội sinh cho các cơ quan nghiên cứu hiện có. Tăng cường sự phối hợp giữa các viện nghiên cứu, trường đại học và các tổ chức quốc tế.
- Xây dựng chương trình ứng dụng và phát triển công nghệ trong quản lý bãi đỗ xe: Tăng cường ứng dụng công nghệ thông tin trong quản lý và vận hành bãi đỗ xe. Phát triển các mô hình kinh doanh trực tuyến nhằm tạo điều kiện thuận lợi cho người dùng và doanh nghiệp.

Nghiên cứu "The Impact of Artificial Intelligence on Urban Traffic Management: A Literature Review" của Zhang et al. (2020) là một bước tiến quan trọng trong việc hiểu rõ về tác động của Trí tuệ nhân tạo (AI) trong quản lý giao thông đô thị. Từ việc phân tích các ứng dụng của AI trong dự đoán nhu cầu đỗ xe đến tối ưu hóa quá trình quản lý và cải thiện trải nghiệm người dùng, nghiên cứu đã đi sâu vào các khía cạnh quan trọng của việc áp dụng công nghệ này. Kết quả của nghiên cứu cung cấp cái nhìn tổng quan về những thách thức và cơ hội mà AI mang lại cho quản lý bãi đỗ xe, cũng như đề xuất hướng phát triển trong tương lai. Những phát hiện này không chỉ là cơ sở cho việc nghiên cứu tiếp theo mà còn cung cấp thông tin quan trọng cho các quyết định chiến lược trong quản lý và phát triển bãi đỗ xe.

Các tác giả đã đề xuất một số hướng phát triển và ứng dụng của Trí tuệ nhân tạo (AI) trong quản lý bãi đỗ xe:

- Nghiên cứu tiếp theo về ứng dụng cụ thể của AI trong từng khía cạnh quản lý bãi đỗ xe, như quản lý đặt chỗ, tối ưu hóa không gian đỗ, và cải thiện trải nghiệm người dùng.
- Phát triển các hệ thống AI dựa trên dữ liệu lớn (Big Data) để dự đoán và đáp ứng nhu cầu đỗ xe, từ dự báo tình hình đỗ xe đến cá nhân hóa dịch vụ cho từng người dùng.
- Nghiên cứu về việc tích hợp các công nghệ mới như Blockchain và Internet of Things (IoT) vào các ứng dụng AI trong quản lý bãi đỗ xe, nhằm tăng cường tính minh bạch, an toàn và hiệu quả cho các quy trình kinh doanh.
- Đào tạo và phát triển nhân lực với kỹ năng và kiến thức về AI trong lĩnh vực quản lý đô thị để đảm bảo rằng công nghệ mới được triển khai và sử dụng một cách hiệu quả.

Những đề xuất này không chỉ giúp mở ra các hướng nghiên cứu mới mà còn góp phần thúc đẩy sự phát triển và cải thiện của quản lý bãi đỗ xe trong tương lai.

1.2.2 Nghiên cứu về ứng dụng

Trí tuệ nhân tạo (AI) đã được áp dụng rộng rãi trong ngành quản lý bãi đỗ xe trên toàn cầu, đóng vai trò quan trọng trong việc cải thiện trải nghiệm đỗ xe, tối ưu hóa quy trình quản lý và tăng cường khả năng dự đoán. Dưới đây là một số ứng dụng tiêu biểu của công nghệ AI trong quản lý bãi đỗ xe:

- Hệ thống đặt chỗ đỗ xe thông minh: Công nghệ AI được sử dụng để phát triển các hệ thống đặt chỗ đỗ xe thông minh, giúp người dùng tìm kiếm và đặt chỗ đỗ một cách dễ dàng và nhanh chóng. Hệ thống này có thể dự đoán nhu cầu đỗ xe, đề xuất các bãi đỗ phù hợp và tạo ra trải nghiệm đặt chỗ cá nhân hóa.
- Hỗ trợ khách hàng tự động: Các hệ thống chatbot được tích hợp AI giúp cung cấp thông tin về bãi đỗ xe, hỗ trợ tìm kiếm chỗ trống, và trả lời các câu hỏi của khách hàng một cách tự động và nhanh chóng. Điều này giúp cải thiện dịch vụ khách hàng và giảm thời gian phản hồi.
- Dự đoán nhu cầu và tối ưu hóa không gian đỗ xe: AI được sử dụng để phân tích dữ liệu lớn và dự đoán nhu cầu đỗ xe, từ đó giúp các nhà quản lý lập kế

hoạch và điều chỉnh không gian đỗ xe một cách hiệu quả. Công nghệ này cũng giúp dự báo tình hình đỗ xe và xác định các mức giá phù hợp.

- Cải thiện trải nghiệm đỗ xe: AI được sử dụng để tạo ra các trải nghiệm đỗ xe cá nhân hóa cho người dùng, từ việc đề xuất bãi đỗ gần nhất đến việc cung cấp thông tin về tình trạng đỗ xe. Công nghệ này cũng có thể phân tích phản hồi của người dùng và điều chỉnh trải nghiệm dựa trên các yêu cầu và mong muốn cụ thể.

Chương 2. CƠ SỞ LÝ THUYẾT

2.1 YOLOv8

2.1.1 Tổng Quan về mô hình YOLOv8

YOLOv8 là phiên bản YOLO mới nhất của Ultralytics. Là một mô hình tiên tiến, hiện đại (SOTA), YOLOv8 được xây dựng dựa trên sự thành công của các phiên bản trước, giới thiệu các tính năng và cải tiến mới để nâng cao hiệu suất, tính linh hoạt và hiệu quả. YOLOv8 hỗ trợ đầy đủ các tác vụ AI về tầm nhìn, bao gồm phát hiện, phân đoạn, ước tính tư thế, theo dõi và phân loại. Tính linh hoạt này cho phép người dùng tận dụng các khả năng của YOLOv8 trên nhiều ứng dụng và miền khác nhau.

Giới thiệu Ultralytics YOLOv8, phiên bản mới nhất của mô hình phân đoạn hình ảnh và phát hiện đối tượng thời gian thực nổi tiếng. YOLOv8 được xây dựng dựa trên những tiến bộ tiên tiến trong học sâu và thị giác máy tính, mang lại hiệu suất vô song về tốc độ và độ chính xác. Thiết kế hợp lý của nó giúp nó phù hợp với nhiều ứng dụng khác nhau và dễ dàng thích ứng với các nền tảng phần cứng khác nhau, từ thiết bị biên cho đến API đám mây.

Khám phá Tài liệu YOLOv8, một tài nguyên toàn diện được thiết kế để giúp bạn hiểu và sử dụng các tính năng cũng như khả năng của nó. Cho dù bạn là một học viên máy học dày dạn kinh nghiệm hay là người mới trong lĩnh vực này, trung tâm này nhằm mục đích tối đa hóa tiềm năng của YOLOv8 trong các dự án của bạn.

2.1.2 Sử dụng mô hình YOLOv8 để phát hiện đối tượng

YOLOv8 của Ultralytics là một nhân tố thay đổi cuộc chơi thực sự. Nó giống như “Dao quân đội Thụy Sĩ” của các mô hình phân đoạn hình ảnh và phát hiện đối tượng. Với tính năng mở rộng, bạn có thể chuyển đổi giữa các phiên bản YOLO khác nhau mà không có vấn đề gì lớn. Nó giống như việc bạn có thể thay ống kính trên máy ảnh, thay vì ống kính, bạn đang hoán đổi các phiên bản của YOLO.

YOLOv8 còn có một mạng đường trục mới, đầu phát hiện không neo mới và chức năng mất mới, khiến nó trở thành lựa chọn hấp dẫn cho nhiều tác vụ phát hiện đối tượng và phân đoạn hình ảnh.

Sử dụng mạng YOLOv8 để giải quyết các vấn đề về phân loại, phát hiện đối tượng và phân đoạn hình ảnh. Tất cả các phương pháp này phát hiện các đối tượng trong hình ảnh hoặc video theo những cách khác nhau, gồm một số bước như sau:

Mạng thần kinh được tạo và đào tạo để phân loại hình ảnh sẽ xác định một loại đối tượng trên hình ảnh và trả về tên của nó cũng như xác suất của dự đoán này.

Mạng thần kinh để phát hiện đối tượng, ngoài loại đối tượng và xác suất, trả về tọa độ của đối tượng trên hình ảnh: x, y, chiều rộng và chiều cao, như được hiển thị trên hình ảnh thứ hai. Mạng thần kinh phát hiện đối tượng cũng có thể phát hiện một số đối tượng trong hình ảnh và các hộp giới hạn của chúng.

Cuối cùng, ngoài các loại đối tượng và hộp giới hạn, mạng thần kinh được đào tạo để phân đoạn hình ảnh sẽ phát hiện hình dạng của các đối tượng.

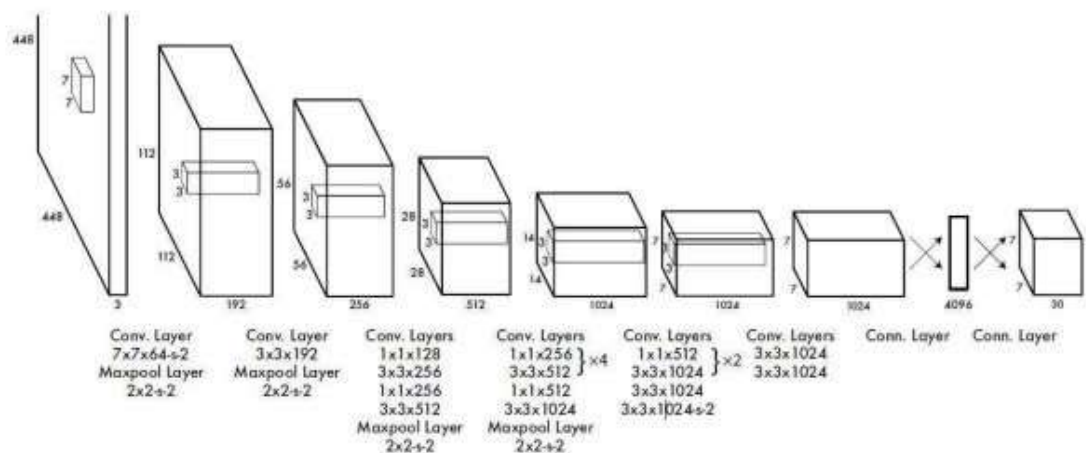


Figure 10. YOLO network structure [11]

YOLO hoạt động bằng cách chia hình ảnh đầu vào thành một lưới các ô và dự đoán các bounding box (hộp bao quanh đối tượng) và xác suất lớp cho mỗi ô.

Kiến trúc mạng YOLO: Mạng này được chia thành nhiều khối (block), mỗi khối bao gồm một hoặc nhiều lớp:

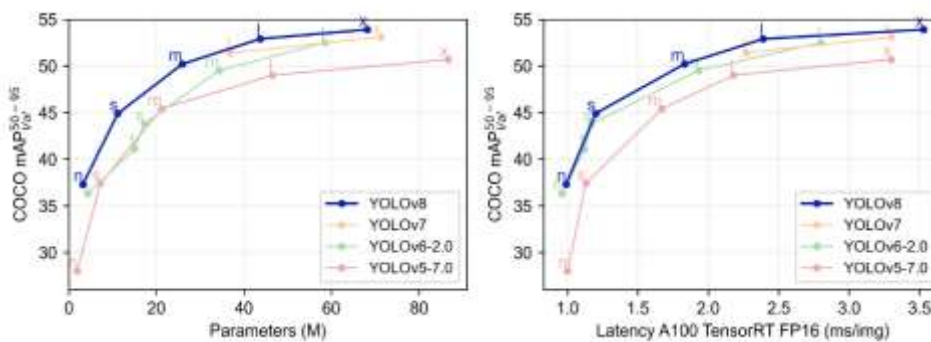
- Khối 1: Lớp chập (Conv) với 64 kernel kích thước 7x7, stride = 2. Lớp này trích xuất các đặc trưng cơ bản từ ảnh đầu vào như cạnh, góc. Lớp gộp tối đa (Maxpool) với kích thước 2x2, stride = 2. Lớp này giảm kích

thước ảnh xuống còn 112×112 , đồng thời giúp loại bỏ nhiễu và tăng tính bất biến với vị trí.

- Khối 2: Lớp chập với 192 kernel kích thước 3×3 , stride = 1. Lớp này tiếp tục trích xuất các đặc trưng phức tạp hơn từ ảnh. Lớp gộp tối đa với kích thước 2×2 , stride = 2. Kích thước ảnh giảm xuống còn 56×56 .
- Khối 3: Lớp chập với 128 kernel kích thước 1×1 , stride = 1. Lớp này giúp giảm số lượng kênh (channels) của ảnh đặc trưng.
 - o Lớp chập với 256 kernel kích thước 3×3 , stride = 1.
 - o Lớp chập với 256 kernel kích thước 1×1 , stride = 1.
 - o Lớp chập với 512 kernel kích thước 3×3 , stride = 1.
 - o Lớp gộp tối đa với kích thước 2×2 , stride = 2. Kích thước ảnh giảm xuống còn 28×28 .
- Khối 5: Tương tự khối 3 và 4, với số lượng kernel là 1024. Kích thước ảnh giảm xuống còn 7×7 .
- Khối 6: Lớp chập với 1024 kernel kích thước 3×3 , stride = 1. Lớp chập với 1024 kernel kích thước 3×3 , stride = 2. Kích thước ảnh giảm xuống còn 3×3 .
- Khối 7: Lớp chập với 1024 kernel kích thước 3×3 , stride = 1. Lớp chập với 1024 kernel kích thước 3×3 , stride = 1.
- Khối 8: Lớp kết nối đầy đủ với 4096 nơ-ron. Lớp kết nối đầy đủ với 30 nơ-ron. Lớp này đưa ra dự đoán cuối cùng về vị trí, kích thước, lớp và độ tin cậy của các đối tượng trong ảnh.

2.1.3 Tại sao sử dụng mô hình YOLOv8

Bảng so sánh các mô phiên bản mô hình Yolo:



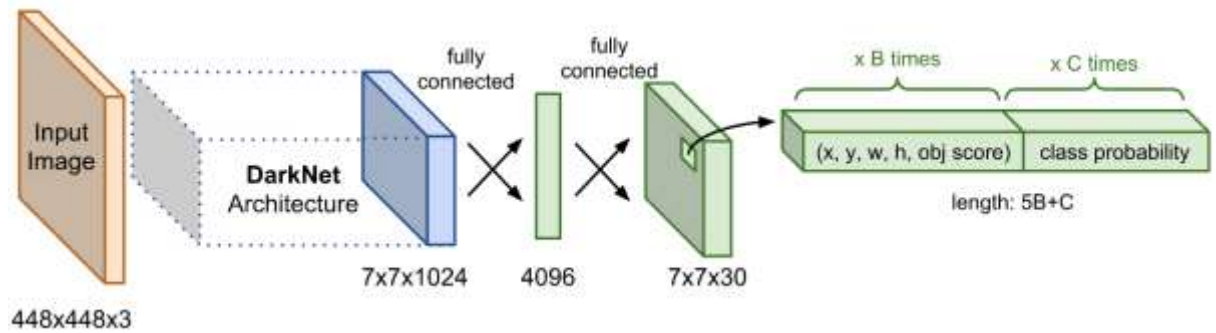
Khi so sánh với các mô hình YOLO khác được đào tạo ở độ phân giải hình ảnh 640, tất cả các mô hình YOLOv8 đều có thông lượng tốt hơn với cùng một số tham số.

Bây giờ, hãy cùng xem chi tiết hiệu suất của các mô hình YOLOv8 mới nhất khi kết hợp với các mô hình YOLOv5 từ Ultralytics. Các bảng sau đây cho thấy sự so sánh toàn diện giữa YOLOv8 và YOLOv5.

Dưới đây là một số tính năng chính về bản phát hành mới:

- API thân thiện với người dùng (Dòng lệnh + Python).
- Nhanh hơn và chính xác hơn.
- Hỗ trợ:
- Phát hiện đối tượng,
- Phân đoạn sơ thảo,
- Phân loại hình ảnh.
- Mở rộng cho tất cả các phiên bản trước.
- Mạng đường trục mới.
- Đầu neo mới.
- Chức năng mát mát mới.

2.1.4 Cách thức hoạt động của yolov8



- Khối đầu vào:Hình ảnh đầu vào:
 - Hình ảnh đầu vào có kích thước 416x416 pixel được chia thành 16x16 ô lưới (grid). Mỗi ô lưới có kích thước 26x26 pixel.
 - Khối Split: Khối Split chia hình ảnh đầu vào thành 4 phần, mỗi phần có kích thước 104x104 pixel.
- Khối Cross Stage Partial (CSP):
 - Khối CSP: Khối CSP là một cấu trúc mạng nơ-ron tích chập được thiết kế để tăng tốc độ xử lý và cải thiện độ chính xác.
 - Khối Residual: Khối Residual kết hợp đầu ra của các lớp trước với đầu ra hiện tại để giúp mạng học tốt hơn các đặc trưng dài hạn.
 - Khối Concat: Khối Concat kết hợp các đặc trưng từ các nhánh khác nhau của khối CSP.
- Khối Feature Pyramid Network (FPN):
 - Khối FPN: Khối FPN kết hợp các đặc trưng từ các độ phân giải khác nhau để cải thiện khả năng phát hiện các đối tượng ở nhiều kích thước.
 - Khối Upsample: Khối Upsample tăng kích thước của các đặc trưng có độ phân giải thấp để phù hợp với các đặc trưng có độ phân giải cao.
 - Khối Concat: Khối Concat kết hợp các đặc trưng từ các độ phân giải khác nhau.
- Khối dự đoán:

- Khối CONV: Khối CONV sử dụng một lớp tích chập để biến đổi các đặc trưng.
- Khối YOLO: Khối YOLO dự đoán các hộp giới hạn (bounding box) và xác suất cho các đối tượng trong hình ảnh.
- Khối Sigmoid: Khối Sigmoid biến đổi đầu ra của khối YOLO thành giá trị nằm trong khoảng 0 và 1, đại diện cho xác suất của đối tượng.
- Khối xử lý hậu kỳ:
 - Khối Non-Maximum Suppression (NMS): Khối NMS loại bỏ các hộp giới hạn trùng lặp và chọn ra hộp giới hạn chính xác nhất cho mỗi đối tượng.
 - Khối Lọc: Khối Lọc loại bỏ các dự đoán có xác suất thấp hơn ngưỡng nhất định.
- Kết quả: YOLOv8 trả về danh sách các đối tượng được phát hiện trong hình ảnh, cùng với vị trí, kích thước và xác suất của mỗi đối tượng.

2.1.5 Mục tiêu của mô hình Yolov8

Mục tiêu của Yolov8 là **phân loại** và **định vị** các đối tượng trong hình ảnh hoặc video. Nó thuộc loại mô hình học có giám sát, yêu cầu tập dữ liệu được gán nhãn trước để huấn luyện:

- **Phát hiện đối tượng:** Xác định vị trí và kích thước của các đối tượng trong ảnh (ví dụ: xe máy, ô tô, người đi bộ) bằng cách vẽ các hộp giới hạn (bounding boxes) xung quanh chúng.
- **Phân loại đối tượng:** Gán nhãn cho mỗi đối tượng được phát hiện (ví dụ: "xe máy", "ô tô", "người").
- **Thực hiện trong thời gian thực:** Yolov8 được thiết kế để xử lý hình ảnh với tốc độ cao, đáp ứng yêu cầu của các ứng dụng thời gian thực như camera giám sát hoặc hệ thống tự lái.

2.1.6 Quá trình thực hiện của mô hình Yolov8

Chuẩn bị dữ liệu:

- **Thu thập dữ liệu:** Thu thập hình ảnh từ camera giám sát bãi đỗ xe, đảm bảo đa dạng về góc chụp, điều kiện ánh sáng,...
- **Gán nhãn:** Sử dụng công cụ gán nhãn (như LabelImg, Roboflow, VGG Image Annotator (VIA)) để vẽ bounding boxes xung quanh các đối tượng (xe máy, ô tô, người,...) và gán nhãn tương ứng cho mỗi đối tượng. Lưu nhãn theo định dạng YOLO (txt) hoặc COCO (json).
- **Tăng cường dữ liệu:** Áp dụng các kỹ thuật tăng cường dữ liệu (data augmentation) như xoay, lật, phóng to, thu nhỏ, thay đổi độ sáng,... để tăng số lượng và đa dạng hóa dữ liệu huấn luyện, giúp mô hình tổng quát hóa tốt hơn.
- **Chia tập dữ liệu:** Chia dữ liệu thành 3 tập: huấn luyện (train), kiểm tra (validation) và kiểm thử (test) theo tỷ lệ phù hợp (ví dụ: 80% train, 10% validation, 10% test).

Huấn luyện mô hình:

- **Lựa chọn mô hình YOLOv8:** Chọn phiên bản YOLOv8 phù hợp (YOLOv8n, YOLOv8s, YOLOv8m,...) dựa trên yêu cầu về độ chính xác và tốc độ xử lý.
- **Cấu hình tham số:** Thiết lập các tham số huấn luyện như learning rate, batch size, số lượng epoch, optimizer (ví dụ: Adam, SGD), hàm mất mát,...
- **Huấn luyện:** Sử dụng tập dữ liệu huấn luyện để huấn luyện mô hình YOLOv8. Theo dõi các chỉ số đánh giá trên tập kiểm tra (validation) để điều chỉnh tham số và ngăn chặn overfitting.
- **Lưu mô hình:** Lưu mô hình đã huấn luyện để sử dụng cho việc dự đoán.

Điều chỉnh tham số:

- **Tìm kiếm siêu tham số:** Sử dụng các kỹ thuật như grid search, random search, hoặc Bayesian optimization để tìm kiếm các giá trị tối ưu cho các siêu tham số (hyperparameter) của mô hình.
- **Theo dõi các chỉ số:** Quan sát các chỉ số đánh giá như mAP, precision, recall, F1-score trên tập kiểm tra (validation) để đánh giá hiệu suất của mô hình và điều chỉnh tham số cho phù hợp.
- **Kiểm tra trên tập kiểm thử:** Sau khi hoàn thành huấn luyện, đánh giá mô hình trên tập kiểm thử (test) để có được đánh giá khách quan về hiệu suất thực tế.

2.1.7 Công thức toán học và Mã giả của mô hình YOLOv8

Hàm mất mát: YOLOv8 sử dụng một hàm mất mát kết hợp, bao gồm các thành phần cho phân loại, định vị đối tượng, và độ tự tin. Các thành phần này có thể bao gồm:

- **Phân loại:** Cross-entropy loss, focal loss.
- **Định vị:** CIOU loss, GIoU loss, DIOU loss.
- **Độ tự tin:** Binary cross-entropy loss.

Mã giả:

Hàm huấn_luyện_YOLOv8(tập_dữ_liệu, tham_số_mô_hình):

1. Khởi tạo mô hình YOLOv8.

2. Nạp tập dữ liệu đã gán nhãn.
3. Chia tập dữ liệu thành các batch.
4. Lặp qua từng epoch:
 - Lặp qua từng batch:
 - Đưa batch dữ liệu vào mô hình.
 - Tính toán hàm mất mát.
 - Cập nhật trọng số mô hình bằng thuật toán lan truyền ngược.
5. Lưu mô hình đã huấn luyện

2.2 PaddleOCR

2.2.1 Tổng quan về mô hình PaddleOCR

PaddleOCR là một bộ công cụ mã nguồn mở mạnh mẽ được phát triển bởi PaddlePaddle, cung cấp các giải pháp nhận dạng ký tự quang học (OCR) toàn diện và hiệu quả. Nó được sử dụng rộng rãi trong nhiều lĩnh vực như:

- Xử lý tài liệu: trích xuất văn bản từ hình ảnh, PDF, hóa đơn, chứng minh thư,...
- Nhận dạng biển số xe: tự động nhận dạng biển số xe từ camera giao thông.
- Nhận dạng mã vạch và QR code: giải mã thông tin từ mã vạch và QR code.
- Chuyển đổi văn bản sang giọng nói: đọc văn bản thành giọng nói tự nhiên.

Thành phần chính của PaddleOCR:

- Mô hình nhận dạng ký tự: PaddleOCR cung cấp nhiều mô hình nhận dạng ký tự, bao gồm mô hình CNN, Transformer và mô hình lai.
- Hệ thống xử lý trước và sau: PaddleOCR cung cấp các công cụ xử lý trước và sau như lọc nhiễu, binarization, phân chia dòng văn bản và nhận dạng ngôn ngữ.
- API: PaddleOCR cung cấp API đơn giản để người dùng dễ dàng tích hợp vào các ứng dụng của họ.

2.2.2 Sử dụng mô hình PaddleOCR để nhận dạng biển số xe

PaddleOCR là một mã nguồn mở được phát triển bởi Baidu PaddlePaddle, hỗ trợ nhận dạng và trích xuất thông tin từ hình ảnh. PaddleOCR có thể nhận dạng tiếng Anh, tiếng Trung, chữ số và hỗ trợ nhận dạng các văn bản dài.

Sử dụng mô hình SAST trong PaddleOCR để xác định vị trí biển số trong ảnh. Mô hình SAST được huấn luyện trên tập dữ liệu ICDAR 2013 và 2015, có khả năng xác định vị trí chính xác các vùng văn bản trong ảnh.

Sử dụng mô hình CRNN trong PaddleOCR để nhận dạng các ký tự trên biển số. Mô hình CRNN được huấn luyện trên tập dữ liệu Street View Text và IIIT5K, có khả năng nhận dạng chính xác các ký tự trong ảnh.

Ưu điểm:

- PaddleOCR có khả năng nhận dạng biển số xe với độ chính xác cao.
- PaddleOCR có tốc độ xử lý nhanh chóng.
- PaddleOCR có thể được sử dụng để nhận dạng biển số xe của nhiều quốc gia khác nhau.

2.2.3 Tại sao sử dụng mô hình PaddleOCR

Có nhiều lý do để sử dụng PaddleOCR để nhận dạng biển số xe thay vì các mô hình khác:

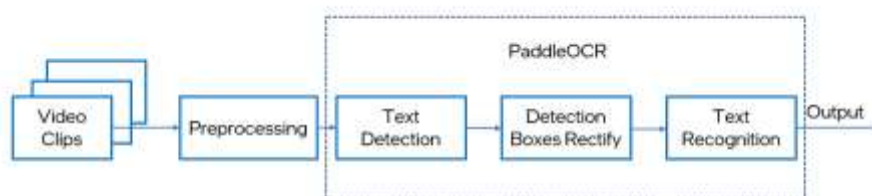
- Khả năng nhận dạng chính xác cao: PaddleOCR được huấn luyện trên tập dữ liệu lớn với nhiều loại biển số xe khác nhau, do đó nó có khả năng nhận dạng chính xác cao.
- Tốc độ xử lý nhanh chóng: PaddleOCR được tối ưu hóa cho hiệu suất, do đó nó có tốc độ xử lý nhanh chóng.
- Khả năng mở rộng: PaddleOCR có thể được mở rộng để hỗ trợ nhận dạng biển số xe của nhiều quốc gia khác nhau.
- Mã nguồn mở: PaddleOCR là mã nguồn mở, do đó bạn có thể dễ dàng sử dụng và chỉnh sửa nó cho phù hợp với nhu cầu của mình.

- Hỗ trợ cộng đồng: PaddleOCR được hỗ trợ bởi một cộng đồng lớn các nhà phát triển, do đó bạn có thể dễ dàng tìm kiếm sự trợ giúp nếu gặp bất kỳ vấn đề nào.

Mô hình	Khả năng nhận dạng chính xác	Tốc độ xử lý	Khả năng mở rộng	Mã nguồn mở	Hỗ trợ cộng đồng
PaddleOCR	Cao	Nhanh	Cao	Có	Cao
EasyOCR	Trung bình	Nhanh	Trung bình	Có	Trung bình
Tesseract	Trung bình	Chậm	Thấp	Có	Thấp
Google Cloud Vision API	Cao	Nhanh	Cao	Không	Cao

Kết luận: PaddleOCR là một lựa chọn tốt cho việc nhận dạng biển số xe vì nó có khả năng nhận dạng chính xác cao, tốc độ xử lý nhanh chóng, khả năng mở rộng cao, mã nguồn mở và được hỗ trợ bởi một cộng đồng lớn các nhà phát triển.

2.2.4 Cách thức hoạt động của PaddleOCR



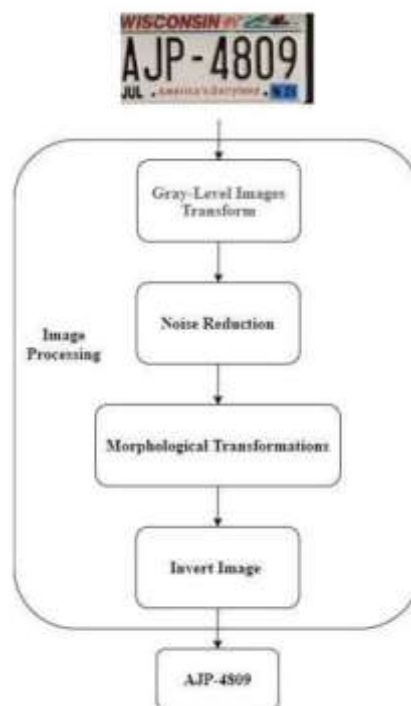
- Nhận hình ảnh, video
- Cải thiện chất lượng hình ảnh(giảm nhiễu hoặc làm sắc nét hình ảnh)
- Phát hiện văn bản(xác định các khu vực của hình ảnh chứa văn bản)
- Chỉnh sửa các vùng chứa văn bản(sửa chữa các biến dạng trong các vùng chứa văn bản)
- Nhận dạng văn bản(nhận dạng các ký tự trong các vùng chứa văn bản và chuyển chúng thành văn bản kỹ thuật số)
- Output(văn bản được nhận dạng)

2.2.5 Mục tiêu của mô hình PaddleOCR

Mục tiêu của PaddleOCR là **nhận dạng ký tự quang học (OCR)**, tức là trích xuất văn bản từ hình ảnh. Đây cũng là một bài toán học có giám sát, cần tập dữ liệu gồm hình ảnh và văn bản tương ứng.:

- **Phát hiện vùng văn bản:** Xác định vị trí của các vùng chứa văn bản trong hình ảnh.
- **Nhận dạng ký tự:** Chuyển đổi các ký tự trong vùng văn bản thành dạng văn bản kỹ thuật số có thể chỉnh sửa và lưu trữ.
- **Hỗ trợ nhiều ngôn ngữ:** PaddleOCR có khả năng nhận dạng nhiều ngôn ngữ, bao gồm tiếng Anh, tiếng Trung, tiếng Việt và các chữ số.

2.2.5 Quá trình thực hiện của mô hình PaddleOCR



Chuẩn bị dữ liệu:

- **Thu thập dữ liệu:** Thu thập hình ảnh biển số xe từ camera giám sát, đảm bảo đa dạng về loại biển số, góc chụp, điều kiện ánh sáng,...
- **Gán nhãn:** Sử dụng công cụ gán nhãn (như PPOCRLabel, LabelImg) để vẽ bounding boxes xung quanh vùng chứa biển số và gán nhãn văn bản tương ứng. Lưu nhãn theo định dạng phù hợp với PaddleOCR.

- **Tăng cường dữ liệu:** Tương tự như YOLOv8, áp dụng các kỹ thuật tăng cường dữ liệu để tăng số lượng và đa dạng hóa dữ liệu.
- **Chia tập dữ liệu:** Chia dữ liệu thành 3 tập: huấn luyện, kiểm tra và kiểm thử.

Huấn luyện mô hình:

- **Lựa chọn mô hình:** PaddleOCR cung cấp nhiều mô hình phát hiện và nhận dạng khác nhau. Chọn mô hình phù hợp dựa trên yêu cầu về độ chính xác, tốc độ và ngôn ngữ.
- **Cấu hình tham số:** Thiết lập các tham số huấn luyện cho cả mô hình phát hiện và nhận dạng.
- **Huấn luyện:** Huấn luyện mô hình PaddleOCR theo 2 giai đoạn: phát hiện vùng văn bản và nhận dạng ký tự.
- **Lưu mô hình:** Lưu mô hình đã huấn luyện.

Điều chỉnh tham số:

- **Tối ưu hóa tham số:** Điều chỉnh các tham số huấn luyện (learning rate, batch size,...) cho cả mô hình phát hiện và nhận dạng để đạt được hiệu suất tốt nhất.
- **Theo dõi các chỉ số:** Theo dõi độ chính xác ký tự, tỷ lệ nhận dạng chính xác, tỷ lệ lỗi ký tự trên tập kiểm tra.
- **Kiểm tra trên tập kiểm thử:** Đánh giá mô hình trên tập kiểm thử để có đánh giá cuối cùng về hiệu suất.

2.1.7 Công thức toán học và Mã giả của mô hình PaddleOCR

Phát hiện vùng văn bản:

- **Công thức:** PaddleOCR sử dụng các phương pháp như DB (Differentiable Binarization) hoặc SAST (Small and Accurate Scene Text) để phát hiện vùng văn bản. Các phương pháp này dựa trên CNN và có thể sử dụng các hàm mất mát như Dice loss hoặc Balanced Cross-Entropy loss.

Nhận dạng ký tự (CRNN):

- **Công thức:** CRNN kết hợp CNN để trích xuất đặc trưng và RNN (LSTM hoặc GRU) để xử lý chuỗi. Hàm mất mát thường được sử dụng là Connectionist Temporal Classification (CTC) loss.

Mã giả:

Hàm huấn_luyện_PaddleOCR(tập_dữ_liệu, tham_số_mô_hình):

1. Khởi tạo mô hình PaddleOCR (phát hiện văn bản và nhận dạng ký tự).
2. Nạp tập dữ liệu hình ảnh và văn bản.
3. Lặp qua từng epoch:
 - Lặp qua từng hình ảnh:
 - Phát hiện vùng văn bản trong hình ảnh.
 - Nhận dạng ký tự trong vùng văn bản.
 - Tính toán hàm mất mát (cho cả phát hiện và nhận dạng).
 - Cập nhật trọng số mô hình.
4. Lưu mô hình đã huấn luyện

2.3 Tkinter và Utils

2.3.1 Tkinter



Tkinter là một thư viện Python mã nguồn mở được sử dụng để xây dựng giao diện người dùng (UI) cho các ứng dụng desktop. Được tích hợp sẵn trong Python, Tkinter cho phép phát triển các ứng dụng với giao diện đồ họa một cách nhanh chóng và dễ dàng.

Với Tkinter, người phát triển có thể tạo ra các cửa sổ ứng dụng và thêm các thành phần giao diện như nút bấm, nhãn, hộp văn bản, và các khung chứa. Tkinter cung cấp

một bộ công cụ phong phú và linh hoạt để xây dựng giao diện người dùng, giúp việc thiết kế và bố trí các thành phần trở nên đơn giản.

Tkinter sử dụng một cách tiếp cận hướng đối tượng, nơi mà các thành phần giao diện được đại diện bởi các đối tượng. Các thuộc tính và hành vi của các thành phần này có thể được tùy chỉnh và quản lý thông qua các phương thức và thuộc tính của các đối tượng đó.

Một điểm mạnh của Tkinter là sự đơn giản và tính dễ sử dụng. Người phát triển có thể nhanh chóng tạo ra các ứng dụng GUI mà không cần phải học một ngôn ngữ lập trình mới hay cài đặt thêm các gói phần mềm phức tạp. Điều này làm cho Tkinter trở thành lựa chọn phổ biến cho các dự án nhỏ hoặc các bài tập học tập về lập trình GUI.

Ngoài ra, cộng đồng Python cũng đóng góp nhiều tài liệu và ví dụ hữu ích, giúp người mới bắt đầu dễ dàng tiếp cận và học cách sử dụng Tkinter. Các thư viện bổ sung như `'ttk'` cũng được cung cấp để mở rộng khả năng và cải thiện giao diện của các ứng dụng Tkinter.

Tóm lại, Tkinter là một thư viện Python mạnh mẽ và tiện lợi cho phát triển giao diện người dùng trên các ứng dụng desktop. Với tính dễ sử dụng và sự hỗ trợ mạnh mẽ từ cộng đồng, Tkinter là một trong những công cụ phổ biến nhất cho việc xây dựng các ứng dụng GUI đơn giản và hiệu quả.

2.3.2 Utils

wolph/python-utils

Python Utils is a module with some convenient utilities not included with the standard Python install



8

Contributors



15k

Used by



91

Stars



37

Forks



Utils (viết tắt của Utilities) là một tập hợp các hàm và công cụ tiện ích được sử dụng để thực hiện các tác vụ chung và thường xuyên trong lập trình. Được thiết kế để tăng hiệu quả và giảm thiểu sự lặp lại mã nguồn, Utils giúp các lập trình viên tập trung vào việc giải quyết các vấn đề cụ thể trong ứng dụng của họ mà không cần phải tái tạo lại các chức năng cơ bản.

Trong nhiều ngôn ngữ lập trình, Utils bao gồm các hàm và công cụ để xử lý chuỗi ký tự, làm việc với ngày giờ, thực hiện các phép tính toán học, và quản lý cấu trúc dữ liệu như danh sách, mảng và bản đồ. Chúng cũng có thể bao gồm các hàm để xử lý đầu vào và đầu ra, làm việc với tệp tin, và thực hiện các tác vụ liên quan đến mạng.

Một điểm mạnh của Utils là tính tái sử dụng. Các hàm và công cụ tiện ích được viết một lần và có thể sử dụng lại trong nhiều dự án khác nhau, giúp giảm thiểu sự lặp lại và tăng tính nhất quán trong mã nguồn. Điều này không chỉ tiết kiệm thời gian mà còn giúp cải thiện chất lượng mã nguồn bằng cách sử dụng các giải pháp đã được kiểm thử và chứng minh là hiệu quả.

Utils thường được tổ chức thành các thư viện hoặc module riêng biệt, giúp dễ dàng quản lý và tích hợp vào các dự án khác nhau. Các ngôn ngữ lập trình phổ biến như Python, JavaScript, Java, và C# đều có các thư viện Utils mạnh mẽ và phong phú, cung cấp một loạt các công cụ tiện ích để hỗ trợ lập trình viên.

Tóm lại, Utils là một phần không thể thiếu trong lập trình, cung cấp các công cụ tiện ích và hàm xử lý chung giúp tăng hiệu quả, giảm sự lặp lại, và nâng cao chất lượng mã nguồn. Với sự hỗ trợ mạnh mẽ từ các thư viện và cộng đồng lập trình, Utils giúp đơn giản hóa quá trình phát triển và bảo trì phần mềm, đồng thời giúp các lập trình viên tập trung vào việc giải quyết các vấn đề cụ thể trong ứng dụng của họ.

2.4 SQL Server

SQL Server là một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) của Microsoft, được thiết kế để hỗ trợ và quản lý dữ liệu doanh nghiệp một cách hiệu quả và an toàn. SQL Server được sử dụng rộng rãi trong các doanh nghiệp và tổ chức lớn nhờ vào khả năng mở rộng, tính linh hoạt và các tính năng bảo mật mạnh mẽ.



2.4.1 Đặc điểm chính của SQL Server

SQL Server mang lại nhiều lợi ích cho các lập trình viên và quản trị viên cơ sở dữ liệu nhờ những đặc điểm nổi bật sau đây:

- Khả năng mở rộng và hiệu suất cao: SQL Server có thể xử lý một lượng lớn dữ liệu và các giao dịch phức tạp nhờ vào khả năng mở rộng theo chiều ngang và dọc. Tính năng này cho phép tăng cường hiệu suất và đáp ứng nhu cầu kinh doanh ngày càng tăng mà không làm giảm hiệu suất.

- Tính sẵn sàng cao và khắc phục sự cố: SQL Server cung cấp các giải pháp như Always On Availability Groups và Database Mirroring để đảm bảo tính sẵn sàng cao của dữ liệu. Những tính năng này giúp bảo vệ dữ liệu khỏi các sự cố và đảm bảo rằng hệ thống có thể phục hồi nhanh chóng sau sự cố.
- Bảo mật mạnh mẽ: SQL Server được trang bị các tính năng bảo mật tiên tiến như mã hóa dữ liệu, kiểm soát truy cập, và các biện pháp bảo vệ dữ liệu nhạy cảm. Tính năng Transparent Data Encryption (TDE) giúp mã hóa toàn bộ cơ sở dữ liệu, bảo vệ dữ liệu ở trạng thái nghỉ.
- Hỗ trợ phân tích và báo cáo: SQL Server Integration Services (SSIS), SQL Server Analysis Services (SSAS), và SQL Server Reporting Services (SSRS) cung cấp các công cụ mạnh mẽ để tích hợp, phân tích và tạo báo cáo dữ liệu. Những công cụ này giúp doanh nghiệp đưa ra các quyết định dựa trên dữ liệu một cách nhanh chóng và chính xác.

2.4.2 Lịch sử hình thành và phát triển

SQL Server được phát triển lần đầu vào năm 1989 bởi Microsoft, hợp tác với Sybase và Ashton-Tate. Ban đầu, SQL Server được thiết kế cho các hệ điều hành UNIX và OS/2, nhưng sau đó đã chuyển sang hỗ trợ các hệ điều hành Windows. Qua nhiều phiên bản, SQL Server đã không ngừng cải tiến và bổ sung các tính năng mới như hỗ trợ XML, dịch vụ phân tích dữ liệu và khả năng tích hợp với các công cụ phát triển của Microsoft như Visual Studio.

Trong suốt những năm qua, SQL Server đã trở thành một phần không thể thiếu của hạ tầng dữ liệu doanh nghiệp, được sử dụng rộng rãi trong nhiều lĩnh vực như tài chính, y tế, giáo dục và thương mại điện tử.

2.4.3 Ứng dụng của SQL Server

SQL Server được ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau nhờ vào khả năng quản lý dữ liệu mạnh mẽ và linh hoạt. Trong phát triển web, SQL Server thường được sử dụng để lưu trữ và quản lý dữ liệu người dùng, xử lý các giao dịch trực tuyến và quản lý nội dung. Trong lĩnh vực tài chính, SQL Server giúp quản lý các giao dịch,

theo dõi dòng tiền và phân tích dữ liệu tài chính. Trong y tế, SQL Server được sử dụng để quản lý hồ sơ bệnh nhân, theo dõi lịch sử điều trị và phân tích dữ liệu y tế.

Với những ưu điểm vượt trội về hiệu suất, bảo mật và khả năng mở rộng, SQL Server đã khẳng định vị thế của mình như một giải pháp cơ sở dữ liệu hàng đầu cho các doanh nghiệp và tổ chức. Cùng với sự phát triển không ngừng của công nghệ và nhu cầu xử lý dữ liệu ngày càng tăng, SQL Server hứa hẹn sẽ tiếp tục là lựa chọn ưu tiên của các nhà phát triển và quản trị viên cơ sở dữ liệu trong việc xây dựng các hệ thống thông tin hiện đại và hiệu quả.

2.5 Các thư viện khác

2.5.1 OpenCV

OpenCV, hay còn gọi là Open Source Computer Vision Library, đã từng bước khẳng định vị thế của mình từ khi ra đời vào năm 1999, do Willow Garage phát triển. Được thiết kế với mục tiêu làm cho công nghệ thị giác máy tính trở nên dễ dàng tiếp cận và áp dụng vào các ứng dụng thực tế, OpenCV đã trở thành một công cụ không thể thiếu trong nghiên cứu và phát triển trong nhiều lĩnh vực, từ xe tự lái đến công nghệ y tế.

Thư viện này cung cấp một loạt các thuật toán và công cụ để xử lý và phân tích hình ảnh số. Các tính năng chính bao gồm nhận dạng đối tượng, phát hiện và theo dõi chuyển động, nhận dạng khuôn mặt và mắt, xử lý ảnh y tế, nhận diện biển báo giao thông, và nhiều ứng dụng khác.

Điểm mạnh của OpenCV không chỉ là sự đa dạng và linh hoạt của các thuật toán và công cụ có sẵn, mà còn là khả năng hỗ trợ nhiều ngôn ngữ lập trình như C++, Python và Java. Điều này giúp cho các nhà phát triển có thể sử dụng thư viện này trên nhiều nền tảng khác nhau một cách thuận tiện và linh hoạt.

Các thuật toán quan trọng tích hợp trong OpenCV bao gồm Haar Cascade để nhận diện đối tượng, thuật toán Shi-Tomasi và Harris Corner Detection để phát hiện góc, thuật toán Lucas-Kanade để theo dõi chuyển động, và nhiều thuật toán khác như SIFT, SURF, FAST và ORB.

Với sự phát triển và mở rộng liên tục, OpenCV không chỉ là một công cụ hữu ích trong quá trình nghiên cứu, mà còn là một phần không thể thiếu trong các ứng dụng thực tế như nhận dạng khuôn mặt, xử lý hình ảnh y tế và hệ thống an ninh. Sự ảnh hưởng của OpenCV đã và đang mở ra nhiều cơ hội mới trong lĩnh vực công nghệ và khoa học máy tính.

Tóm lại, OpenCV không chỉ là một công cụ mạnh mẽ cho việc xử lý hình ảnh và thị giác máy tính, mà còn là một nguồn tài nguyên quan trọng đối với cộng đồng nghiên cứu và phát triển. Tuy nhiên, việc sử dụng OpenCV cũng đặt ra một số thách thức mà các nhà phát triển cần vượt qua để tận dụng được toàn bộ tiềm năng của thư viện này.

2.5.2 CVZone

CvZone là một thư viện mã nguồn mở mạnh mẽ được phát triển nhằm hỗ trợ các dự án liên quan đến thị giác máy tính và trí tuệ nhân tạo. Ra đời với mục tiêu làm cho việc triển khai các ứng dụng thị giác máy tính trở nên dễ dàng và hiệu quả hơn, CvZone đã nhanh chóng trở thành một công cụ hữu ích trong cộng đồng phát triển phần mềm và nghiên cứu.

Thư viện này cung cấp nhiều công cụ và thuật toán để xử lý và phân tích hình ảnh cũng như video, bao gồm nhận diện khuôn mặt, phát hiện đối tượng, theo dõi chuyển động, và nhiều ứng dụng khác trong thị giác máy tính. CvZone không chỉ đơn giản hóa việc tích hợp các chức năng phức tạp mà còn tối ưu hóa hiệu suất cho các dự án liên quan đến trí tuệ nhân tạo.

Một trong những điểm mạnh của CvZone là tính tương thích cao với nhiều ngôn ngữ lập trình và nền tảng khác nhau, đặc biệt là Python. Điều này giúp các nhà phát triển có thể sử dụng thư viện này một cách linh hoạt và thuận tiện trên các hệ điều hành khác nhau. Bên cạnh đó, CvZone còn tích hợp tốt với các thư viện khác như OpenCV, giúp mở rộng khả năng và hiệu suất của các ứng dụng.

Các tính năng quan trọng của CvZone bao gồm:

- Nhận diện và theo dõi đối tượng: CvZone cung cấp các công cụ để phát hiện và theo dõi các đối tượng trong thời gian thực, giúp phát triển các ứng dụng như giám sát an ninh, phân tích video và hệ thống hỗ trợ lái xe.

- Phát hiện và nhận diện khuôn mặt: Với các thuật toán tiên tiến, CvZone có khả năng nhận diện và phân tích khuôn mặt, hỗ trợ các ứng dụng như nhận diện người dùng, phân tích cảm xúc và kiểm soát truy cập.

- Tích hợp dễ dàng với OpenCV: CvZone có thể được sử dụng kết hợp với OpenCV, tận dụng các thuật toán và công cụ mạnh mẽ của OpenCV để tăng cường khả năng xử lý hình ảnh và video.

CvZone không chỉ là một công cụ hữu ích trong việc phát triển các dự án thị giác máy tính mà còn là một nguồn tài nguyên quý giá cho các nhà nghiên cứu và phát triển trong lĩnh vực trí tuệ nhân tạo. Với các tính năng vượt trội và sự linh hoạt trong ứng dụng, CvZone đang mở ra nhiều cơ hội mới trong công nghệ và khoa học máy tính.

Chương 3. TRAIN MÔ HÌNH

3.1 Trước khi train mô hình.

```
1 # Load a model
2 model = YOLO("yolov8n.yaml") # build a new model from YAML
3 model = YOLO("yolov8n.pt") # load a pretrained model (recommended for training)
4 model = YOLO("yolov8n.yaml").load("yolov8n.pt") # build from YAML and transfer weights
5
6 # Train the model
7 results = model.train(data="coco8.yaml", epochs=100, imgsz=640)
```

Train mô hình

Đoạn code trên được sử dụng để khởi tạo mô hình YOLOv8 và xây dựng mô hình từ một tệp .YAML, sau đó tải một mô hình được huấn luyện trước. Tiếp theo là xây dựng mô hình từ YAML và chuyển trọng số từ mô hình đã được huấn luyện. Cuối cùng là *results* chứa kết quả của quá trình huấn luyện mô hình. Các thông số như số epoch, kích thước hình ảnh (imgsz), và tệp dữ liệu (data) được cung cấp để huấn luyện mô hình.

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
1/200	4.08G	1.069	2.589	1.39	33	640: 100% 16/16 [00:11<00:00, 1.35it/s]
	Class	Images	Instances	Box(P	R	mAP50 mAP50-95): 100% 3/3 [00:02<00:00, 1.45it/s]
	all	70	218	0.793	0.682	0.74 0.549
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
2/200	4.17G	0.8919	1.139	1.245	17	640: 100% 16/16 [00:04<00:00, 3.34it/s]
	Class	Images	Instances	Box(P	R	mAP50 mAP50-95): 100% 3/3 [00:00<00:00, 3.26it/s]
	all	70	218	0.771	0.729	0.796 0.596
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
3/200	4.17G	0.9677	1.158	1.308	24	640: 100% 16/16 [00:06<00:00, 2.31it/s]
	Class	Images	Instances	Box(P	R	mAP50 mAP50-95): 100% 3/3 [00:00<00:00, 3.27it/s]
	all	70	218	0.652	0.47	0.508 0.319
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size
4/200	4.15G	1.023	1.17	1.33	29	640: 100% 16/16 [00:04<00:00, 3.69it/s]
	Class	Images	Instances	Box(P	R	mAP50 mAP50-95): 100% 3/3 [00:01<00:00, 2.93it/s]
	all	70	218	0.597	0.481	0.478 0.285

3.2 Quá trình train mô hình.

3.2.1 Triển khai mô hình YoloV8

```

1  import cv2
2  import cvzone
3  import math
4  from ultralytics import YOLO
5
6  vid1 = r'D:\Code\Đồ án chuyên ngành\New folder\video.mp4'
7
8  cap = cv2.VideoCapture(vid1)
9  #none
10
11  model = YOLO('last.pt')
12  classnames = ['license-plate','vehicle']
13
14  while True:
15      ret, frame = cap.read()
16      frame = cv2.resize(frame, (550,720))
17      results = model(frame)
18
19      for info in results:
20          parameters = info.boxes
21          for box in parameters:
22              x1, y1, x2, y2 = box.xyxy[0]
23              x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
24              confidence = box.conf[0]
25              class_detect = box.cls[0]
26              class_detect = int(class_detect)
27
28              # Check for valid index before accessing classnames
29              if 0 <= class_detect < len(classnames):
30                  class_detect = classnames[class_detect]
31              else:
32                  # Handle the case where class_detect is out of range (e.g., print a message)
33                  print("Warning: Class index out of range")
34                  continue
35
36              conf = math.ceil(confidence * 100)
37              if conf > 50 and class_detect == 'license-plate':
38                  cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
39                  cvzone.putTextRect(frame, f'{class_detect}', [x1 + 8, y1 - 12], thickness=2, scale=1)
40
41              cv2.imshow('frame', frame)
42              if cv2.waitKey(1) & 0xFF == ord('t'):
43                  break
44
45  cap.release()
46  cv2.destroyAllWindows()

```

3.2.2 Sử dụng nền tảng hỗ trợ gán dữ liệu hình ảnh.

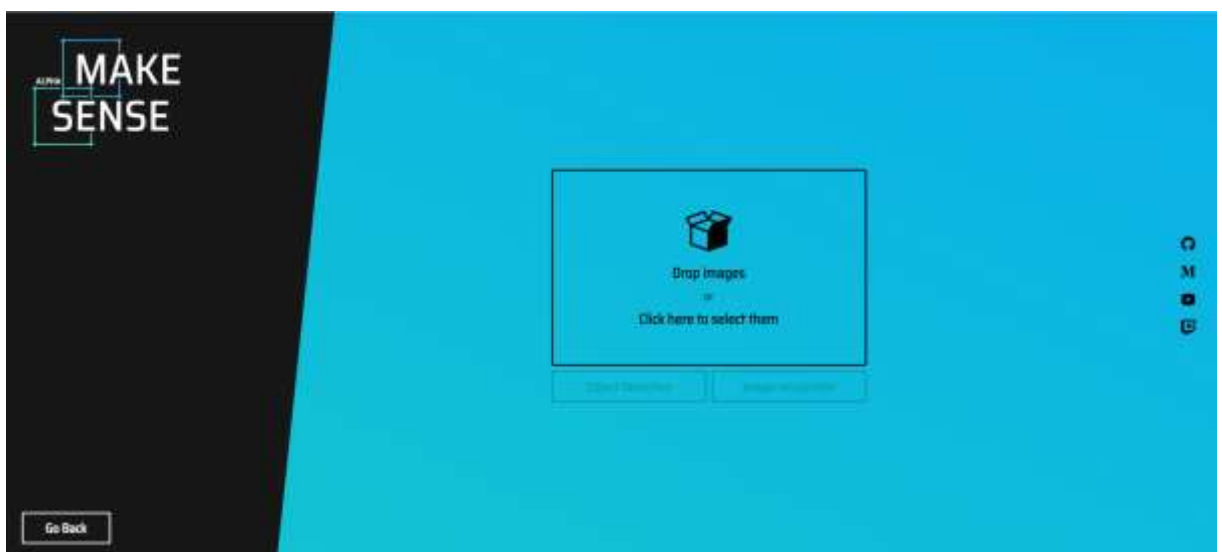
3.2.2.1 Về Makesense.ai

- Makesense.ai là một nền tảng hỗ trợ gán nhãn dữ liệu hình ảnh bằng sự hỗ trợ của trí tuệ nhân tạo (AI). Dựa trên công nghệ AI, Makesense.ai cho phép người dùng gán nhãn cho các đối tượng trong hình ảnh một cách dễ dàng và hiệu quả. Dưới đây là một số điểm nổi bật về Makesense.ai:
- Gán nhãn dữ liệu hình ảnh: Makesense.ai giúp người dùng

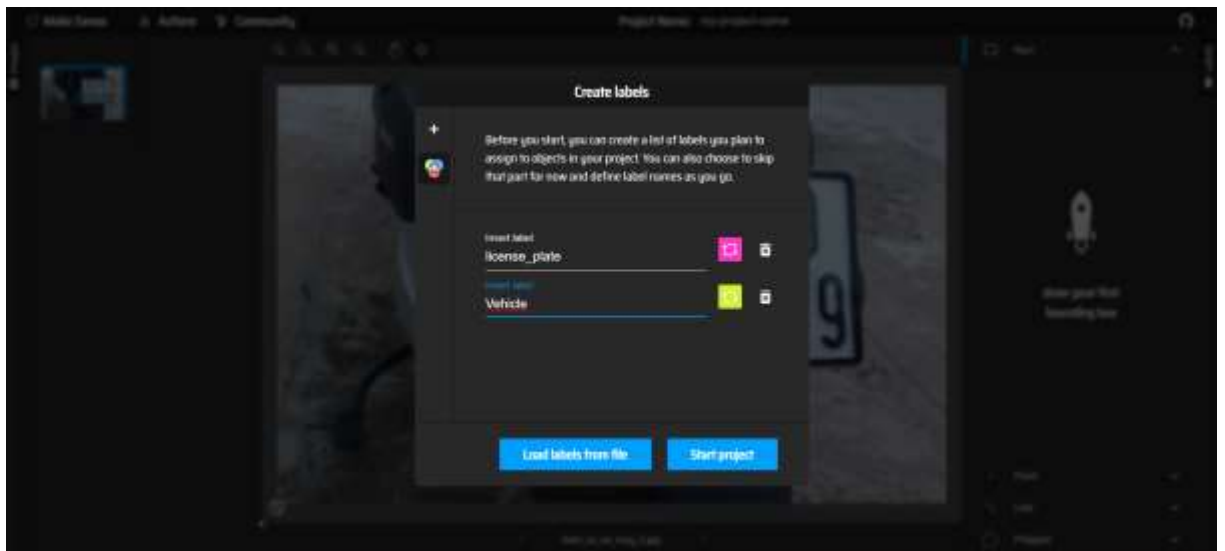
gán nhãn cho các đối tượng trong hình ảnh. Điều này hỗ trợ trong việc xây dựng các tập dữ liệu được sử dụng cho huấn luyện mô hình học máy.

- Dễ sử dụng: Với giao diện thân thiện và các tính năng kéo và thả, Makesense.ai giúp người dùng dễ dàng gán nhãn và tạo dữ liệu chuẩn để phát triển các ứng dụng AI.
- Ứng dụng rộng rãi: Makesense.ai có thể được sử dụng trong nhiều lĩnh vực, bao gồm phân loại ảnh, nhận diện đối tượng, xử lý ngôn ngữ tự nhiên và nhiều hơn nữa. Định dạng lại dữ liệu nhận được từ PaddleOCR.

3.2.2.2 Lấy dữ liệu hình ảnh từ Makesense.ai



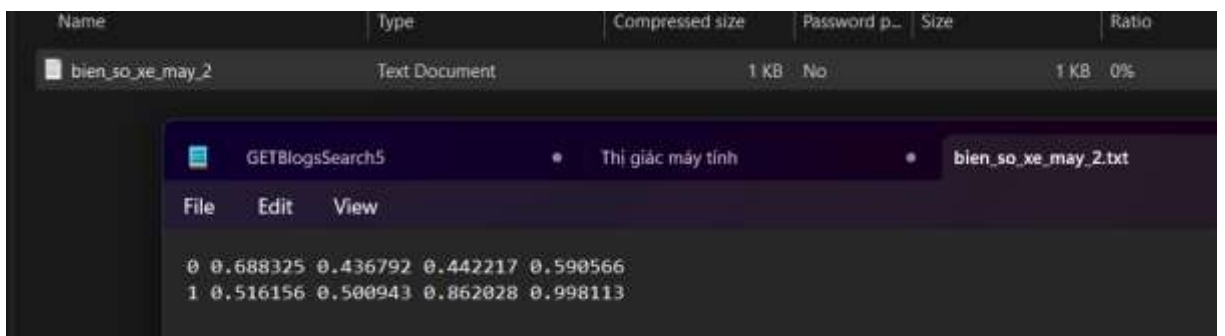
Nền tảng hỗ trợ Makesense.ai



Gán giá trị cho dữ liệu hình ảnh



Sau khi xong các bước trên thì ta sẽ xuất dữ liệu hình ảnh về và di chuyển vào thư mục chứa dữ liệu hình ảnh

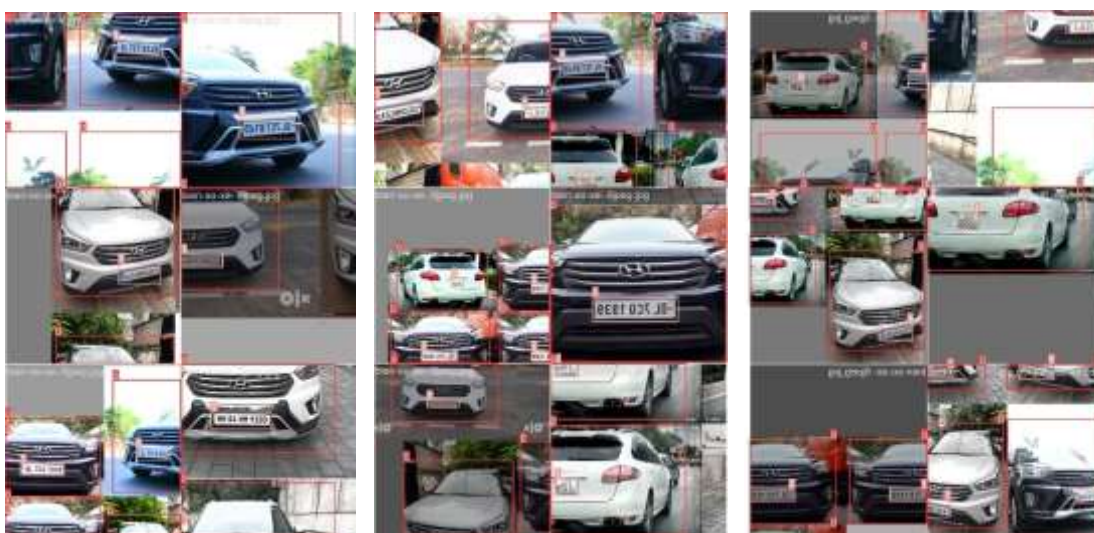


Dữ liệu hình ảnh

Tiếp tục thực hiện với nhiều hình ảnh khác nhau để lấy dữ liệu:



Dưới đây là kết quả sau khi train mô hình với hơn 500 ảnh



3.2.3 Triển khai mô hình khác

3.2.3.1 Resnet

3.2.3.1.1 Tổng quan về mô hình resnet

ResNet (Residual Network) là một mô hình mạng nơ-ron sâu (Deep Neural Network) nổi bật trong lĩnh vực học sâu, được giới thiệu bởi Kaiming He và cộng sự trong bài báo "Deep Residual Learning for Image Recognition" (2015). ResNet đã đạt được thành công lớn trong các bài toán nhận diện hình ảnh và học sâu nói chung nhờ vào việc giải quyết vấn đề độ sâu của mạng và gradient biến mất khi huấn luyện mạng rất sâu.

Tổng quan chính:

a. Vấn đề cần giải quyết

Khi mạng trở nên quá sâu, hiệu suất thường giảm đi dù chúng có khả năng biểu diễn cao hơn. Điều này xảy ra do:

Gradient biến mất hoặc phát nổ trong quá trình lan truyền ngược.

Mạng học chậm hơn và dễ rơi vào cạm bẫy tối ưu kém.

b. Ý tưởng chính của ResNet

Thay vì huấn luyện trực tiếp mạng sâu, ResNet sử dụng một kiến trúc *Residual Learning*.

Thay vì học ánh xạ đầu vào $H(x)$, nó học hàm dư $F(x) = H(x) - x$

Điều này được thực hiện bằng cách sử dụng các kết nối tắt (shortcut connections), cho phép tín hiệu đầu vào bỏ qua một hoặc nhiều lớp.

c. Residual Block

Một khối residual cơ bản bao gồm:

Hai hoặc ba lớp nơ-ron thông thường (convolutional, batch normalization, activation).

Kết nối tắt, cộng trực tiếp đầu vào ban đầu vào đầu ra của các lớp.

Công thức:

$$y = F(x, W) + x$$

Trong đó $F(x, W)$ là phép biến đổi học được, và x là đầu vào ban đầu.

d. Kiến trúc mạng

ResNet được xây dựng theo tầng:

ResNet-18, ResNet-34: Sử dụng residual block đơn giản (2 lớp).

ResNet-50, ResNet-101, ResNet-152: Sử dụng bottleneck block (3 lớp) để tăng hiệu quả tính toán.

e. Ưu điểm ResNet

Huấn luyện mạng rất sâu (lên tới hàng trăm hoặc thậm chí hàng nghìn lớp) mà không gặp vấn đề gradient biến mất.

Hiệu suất vượt trội trong các bài toán như nhận diện hình ảnh (ImageNet), phát hiện vật thể, và phân đoạn hình ảnh.

f. Ứng dụng

Nhận diện hình ảnh, phân loại.

Xử lý video.

Thị giác máy tính (computer vision) tổng quát.

Mô hình hóa ngôn ngữ và các bài toán ngoài thị giác nhờ ý tưởng kết nối dư.

3.2.3.1.2 Sử dụng mô hình ResNet để phát hiện đối tượng

Sử dụng ResNet để phát hiện đối tượng (object detection) là một cách tiếp cận phổ biến trong thị giác máy tính. Tuy ResNet được thiết kế ban đầu cho nhiệm vụ phân loại hình ảnh, nhưng nó có thể được tích hợp vào các hệ thống phát hiện đối tượng bằng cách sử dụng như một mạng cơ sở (backbone) để trích xuất đặc trưng từ hình ảnh đầu vào.

Các bước chính để sử dụng ResNet cho phát hiện đối tượng

a. Sử dụng ResNet làm backbone

ResNet hoạt động như một mạng cơ sở trích xuất đặc trưng quan trọng từ hình ảnh.

Các lớp cuối của ResNet (fully connected layers hoặc các lớp phân loại) thường được loại bỏ, chỉ giữ lại các lớp convolution để tạo ra bản đồ đặc trưng (feature maps).

b. Kết hợp với các mô-đun phát hiện đối tượng

ResNet được sử dụng trong các mô hình phát hiện đối tượng phổ biến như:

Faster R-CNN: Sử dụng ResNet làm backbone để tạo ra các bản đồ đặc trưng, sau đó kết hợp với Region Proposal Network (RPN) để tạo ra các đề xuất vùng quan tâm (Region of Interest - ROI) và dự đoán nhãn.

YOLO (You Only Look Once) hoặc SSD (Single Shot MultiBox Detector): Một số phiên bản của các mô hình này tích hợp ResNet hoặc một biến thể làm mạng cơ sở.

Mask R-CNN: Phát triển từ Faster R-CNN để phát hiện đối tượng và phân đoạn ảnh.

c. Fine-tuning ResNet

Khởi tạo ResNet với trọng số đã được huấn luyện trước trên ImageNet (pretrained weights).

Fine-tune các lớp cao (hoặc toàn bộ mạng) để phù hợp với tập dữ liệu phát hiện đối tượng cụ thể.

Quy trình triển khai sử dụng InceptionResNet để phát hiện đối tượng

a. Chuẩn bị dữ liệu

Tải dữ liệu gốc và các tệp XML chú thích (annotation):

Dữ liệu được lưu trong thư mục ../images/ với tệp ảnh và chú thích trong định dạng XML.

Sử dụng thư viện xml.etree.ElementTree để trích xuất thông tin bounding box từ tệp XML (tọa độ xmin, xmax, ymin, ymax).

```
1  from glob import glob
2  import xml.etree.ElementTree as xet
3  import pandas as pd
4
5  # Đọc các tệp XML
6  path = glob('../images/*.xml')
7  labels_dict = dict(filepath=[], xmin=[], xmax=[], ymin=[], ymax=[])
8
9  # Duyệt qua từng tệp XML
10 for filename in path:
11     info = xet.parse(filename)
12     root = info.getroot()
13     member_object = root.find('object')
14     labels_info = member_object.find('bndbox')
15     xmin = int(labels_info.find('xmin').text)
16     xmax = int(labels_info.find('xmax').text)
17     ymin = int(labels_info.find('ymin').text)
18     ymax = int(labels_info.find('ymax').text)
19
20     labels_dict['filepath'].append(filename)
21     labels_dict['xmin'].append(xmin)
22     labels_dict['xmax'].append(xmax)
23     labels_dict['ymin'].append(ymin)
24     labels_dict['ymax'].append(ymax)
25
26 # Lưu dữ liệu bounding box vào DataFrame
27 df = pd.DataFrame(labels_dict)
28 df.to_csv('labels.csv', index=False)
29 df.head()
```

Liên kết ảnh với dữ liệu bounding box:

Trích xuất tên tệp ảnh từ tệp XML và ánh xạ với đường dẫn ảnh.



```
1 import os
2
3 def getFilename(filename):
4     filename_image = xet.parse(filename).getroot().find('filename').text
5     filepath_image = os.path.join('../images', filename_image)
6     return filepath_image
7
8 # Tạo danh sách đường dẫn đến ảnh
9 image_path = list(df['filepath'].apply(getFilename))
```

b. Xử lý dữ liệu

Đọc ảnh và chuẩn hóa dữ liệu đầu vào:

Resize tất cả các ảnh về kích thước chuẩn (224x224).

Chuẩn hóa giá trị điểm ảnh về khoảng [0, 1].

Chuẩn hóa bounding box:

Tọa độ bounding box (xmin, xmax, ymin, ymax) được chuẩn hóa dựa trên kích thước của ảnh.

```

1 import cv2
2 from tensorflow.keras.preprocessing.image import load_img, img_to_array
3 import numpy as np
4
5 labels = df.iloc[:, 1:].values
6 data, output = [], []
7
8 for ind in range(len(image_path)):
9     image = image_path[ind]
10    img_arr = cv2.imread(image)
11    h, w, d = img_arr.shape
12
13    # Resize và chuẩn hóa ảnh
14    load_image = load_img(image, target_size=(224, 224))
15    load_image_arr = img_to_array(load_image)
16    norm_load_image_arr = load_image_arr / 255.0
17
18    # Chuẩn hóa tọa độ bounding box
19    xmin, xmax, ymin, ymax = labels[ind]
20    nxmin, nxmax = xmin / w, xmax / w
21    nymin, nymin = ymin / h, ymax / h
22    label_norm = (nxmin, nxmax, nymin, nymin)
23
24    data.append(norm_load_image_arr)
25    output.append(label_norm)
26
27 # Chuyển đổi dữ liệu thành array
28 X = np.array(data, dtype=np.float32)
29 y = np.array(output, dtype=np.float32)

```

Chia tập dữ liệu:

Chia dữ liệu thành tập huấn luyện (80%) và tập kiểm tra (20%).

```

1 from sklearn.model_selection import train_test_split
2
3 x_train, x_test, y_train, y_test = train_test_split(X, y, train_size=0.8, random_state=0)

```

c. Xây dựng mô hình

Sử dụng InceptionResNetV2 làm backbone:

Lấy các đặc trưng từ tầng convolutional của mô hình đã được huấn luyện trước.

Xây dựng đầu ra bounding box:

Sử dụng các lớp Dense để dự đoán 4 tọa độ của bounding box.

```
1 from tensorflow.keras.applications import InceptionResNetV2
2 from tensorflow.keras.layers import Dense, Dropout, Flatten, Input
3 from tensorflow.keras.models import Model
4
5 # Tải mô hình InceptionResNetV2 đã pretrained
6 inception_resnet = InceptionResNetV2(weights='imagenet', include_top=False, input_tensor=Input(shape=(224, 224, 3)))
7
8 # Thêm các lớp Fully Connected cho đầu ra bounding box
9 headmodel = inception_resnet.output
10 headmodel = Flatten()(headmodel)
11 headmodel = Dense(500, activation='relu')(headmodel)
12 headmodel = Dense(250, activation='relu')(headmodel)
13 headmodel = Dense(4, activation='sigmoid')(headmodel)
14
15 # Kết hợp mô hình
16 model = Model(inputs=inception_resnet.input, outputs=headmodel)
17
18 # Compile mô hình
19 model.compile(loss='mse', optimizer=tf.keras.optimizers.Adam(learning_rate=1e-4))
20 model.summary()
21
```

d. Huấn luyện mô hình

Sử dụng TensorBoard để theo dõi quá trình huấn luyện.

Lưu mô hình sau khi hoàn thành.

```
1 from tensorflow.keras.callbacks import TensorBoard
2
3 # Cấu hình TensorBoard
4 tfb = TensorBoard(log_dir='object_detection')
5
6 # Huấn luyện mô hình
7 history = model.fit(
8     x=x_train, y=y_train,
9     batch_size=10,
10    epochs=15,
11    validation_data=(x_test, y_test),
12    callbacks=[tfb]
13 )
14
15 # Lưu mô hình
16 model.save('./object_detection.h5')
```

e. Dự đoán và hiển thị kết quả

Sử dụng mô hình đã huấn luyện để dự đoán bounding box trên tập kiểm tra và hiển thị kết quả.



```
1 import matplotlib.pyplot as plt
2
3 # Dự đoán trên một ảnh
4 idx = 0 # chỉ số ảnh
5 test_image = x_test[idx]
6 predicted_bbox = model.predict(np.expand_dims(test_image, axis=0))[0]
7 true_bbox = y_test[idx]
8
9 # Hiển thị ảnh với bounding box dự đoán
10 fig, ax = plt.subplots(1, 1, figsize=(8, 8))
11 ax.imshow(test_image)
12 ax.add_patch(plt.Rectangle(
13     (predicted_bbox[0] * 224, predicted_bbox[2] * 224),
14     (predicted_bbox[1] - predicted_bbox[0]) * 224,
15     (predicted_bbox[3] - predicted_bbox[2]) * 224,
16     edgecolor='red', fill=False, linewidth=2, label='Predicted'
17 ))
18 plt.show()
```

3.2.3.1.3 Phân tích thông tin từ XML



```
1 <annotation>
2   <folder>images</folder>
3   <filename>N1.jpeg</filename>
4   <path>/Users/asik/Desktop/ANPR/imagesN1.jpeg</path>
5   <source>
6     <database>Unknown</database>
7   </source>
8   <size>
9     <width>1920</width>
10    <height>1080</height>
11    <depth>3</depth>
12  </size>
13  <segmented>0</segmented>
14  <object>
15    <name>number_plate</name>
16    <pose>Unspecified</pose>
17    <truncated>0</truncated>
18    <difficult>0</difficult>
19    <bndbox>
20      <xmin>1093</xmin>
21      <ymin>645</ymin>
22      <xmax>1396</xmax>
23      <ymax>727</ymax>
24    </bndbox>
25  </object>
26 </annotation>
```

Khi đã hoàn tất quá trình ghi nhãn, bây giờ chúng ta cần thực hiện một số bước xử lý trước dữ liệu. Vì đầu ra của nhãn là XML để sử dụng cho quá trình đào tạo nên chúng tôi cần dữ liệu ở định dạng mảng. Để làm được điều đó, chúng ta sẽ lấy thông tin hữu ích từ nhãn là các điểm chéo của hộp hình chữ nhật hoặc hộp giới hạn lần lượt là xmin, ymin, xmax, ymax. Điều này có sẵn trong XML. Vì vậy, chúng ta cần trích xuất thông tin và lưu nó ở bất kỳ định dạng thuận tiện nào, ở đây tôi sẽ chuyển đổi thông tin giới hạn thành CSV và sau này, tôi sẽ chuyển đổi thông tin đó thành một mảng bằng Pandas. Bây giờ hãy xem cách phân tích thông tin bằng Python.

3.2.3.1.4 Phân tích dữ liệu từ XML và chuyển đổi nó thành CSV

Trước tiên, hãy tải tất cả các thư viện sẽ sử dụng trong dự án này cùng một lúc. Ngoài ra, sẽ sử dụng thư viện python xml.etree để phân tích dữ liệu từ XML và cũng nhập gấu trúc và toàn cầu. Sử dụng glob trước tiên hãy lấy tất cả các tệp XML được tạo trong quá trình ghi nhãn.

```
1 import os
2 import cv2
3 import numpy as np
4 import pandas as pd
5 import tensorflow as tf
6 import pytesseract as pt
7 import plotly.express as px
8 import matplotlib.pyplot as plt
9 import xml.etree.ElementTree as xet
10
11 from glob import glob
12 from skimage import io
13 from shutil import copy
14 from tensorflow.keras.models import Model
15 from tensorflow.keras.callbacks import TensorBoard
16 from sklearn.model_selection import train_test_split
17 from tensorflow.keras.applications import InceptionResNetV2
18 from tensorflow.keras.layers import Dense, Dropout, Flatten, Input
19 from tensorflow.keras.preprocessing.image import load_img, img_to_array
```



```

1 path = glob('../input/number-plate-detection/images/*.xml')
2 labels_dict = dict(filepath=[],xmin=[],xmax=[],ymin=[],ymax=[])
3 for filename in path:
4
5     info = xet.parse(filename)
6     root = info.getroot()
7     member_object = root.find('object')
8     labels_info = member_object.find('bndbox')
9     xmin = int(labels_info.find('xmin').text)
10    xmax = int(labels_info.find('xmax').text)
11    ymin = int(labels_info.find('ymin').text)
12    ymax = int(labels_info.find('ymax').text)
13
14    labels_dict['filepath'].append(filename)
15    labels_dict['xmin'].append(xmin)
16    labels_dict['xmax'].append(xmax)
17    labels_dict['ymin'].append(ymin)
18    labels_dict['ymax'].append(ymax)

```

Trong đoạn code trên, lấy từng tệp riêng lẻ và phân tích thành xml.etree và tìm đối tượng -> bndbox. Sau đó, trích xuất xmin,xmax,ymin,ymax và lưu các giá trị đó vào từ điển. Sau khi chuyển đổi nó thành khung dữ liệu gấu trúc và lưu nó vào tệp CSV và lưu nó vào thư mục dự án như hiển thị bên dưới.

```

1 df = pd.DataFrame(labels_dict)
2 df.to_csv('labels.csv',index=False)
3 df.head()

```


	filepath	xmin	xmax	ymin	ymax
0	../input/number-plate-detection/images/N148.xml	244	369	240	293
1	../input/number-plate-detection/images/N177.xml	331	538	263	317
2	../input/number-plate-detection/images/N173.xml	80	335	150	243
3	../input/number-plate-detection/images/N213.xml	131	209	129	153
4	../input/number-plate-detection/images/N119.xml	180	559	216	314

Với đoạn code trên, đã trích xuất thành công vị trí đường chéo của mỗi hình ảnh và chuyển đổi dữ liệu từ định dạng không có cấu trúc sang định dạng có cấu trúc. Có thể có dữ liệu A look ở trên. Bây giờ cũng trích xuất tên tệp hình ảnh tương ứng của XML.

```

1 filename = df['filepath'][0]
2 def getFilename(filename):
3     filename_image = xet.parse(filename).getroot().find('filename').text
4     filepath_image = os.path.join('../input/number-plate-detection/images',filename_image)
5     return filepath_image
6 getFilename(filename)

```

'../input/number-plate-detection/images/N148.jpeg'

```

1 image_path = list(df['filepath'].apply(getFilename))
2 image_path[:10]#random check

```

['../input/number-plate-detection/images/N148.jpeg',

'../input/number-plate-detection/images/N177.jpeg',

'../input/number-plate-detection/images/N173.jpeg',

'../input/number-plate-detection/images/N213.jpeg',

'../input/number-plate-detection/images/N119.jpeg',

'../input/number-plate-detection/images/N103.jpeg',

```
'../input/number-plate-detection/images/N166.jpeg',  
'../input/number-plate-detection/images/N127.jpeg',  
'../input/number-plate-detection/images/N162.jpeg',  
'../input/number-plate-detection/images/N198.jpeg']
```

3.2.3.1.5 Xác minh dữ liệu

xác minh hộp giới hạn có xuất hiện chính xác đối với một hình ảnh nhất định hay không. Ở đây xem xét hình ảnh N2.jpeg và vị trí đường chéo tương ứng có thể tìm thấy trong df.

```
1 file_path = image_path[87] #path of our image N2.jpeg  
2 img = cv2.imread(file_path) #read the image  
3 # xmin=1804/ymax=1734/xmax=2493/ymax=1882  
4 img = io.imread(file_path) #read the image  
5 fig = px.imshow(img)  
6 fig.update_layout(width=600, height=500, margin=dict(l=10, r=10, b=10, t=10), xaxis_title='Figure 8 - N2.jpeg with bounding box')  
7 fig.add_shape(type='rect', x8=1804, x1=2493, y8=1734, y1=1882, xref='x', yref='y', line_color='cyan')
```

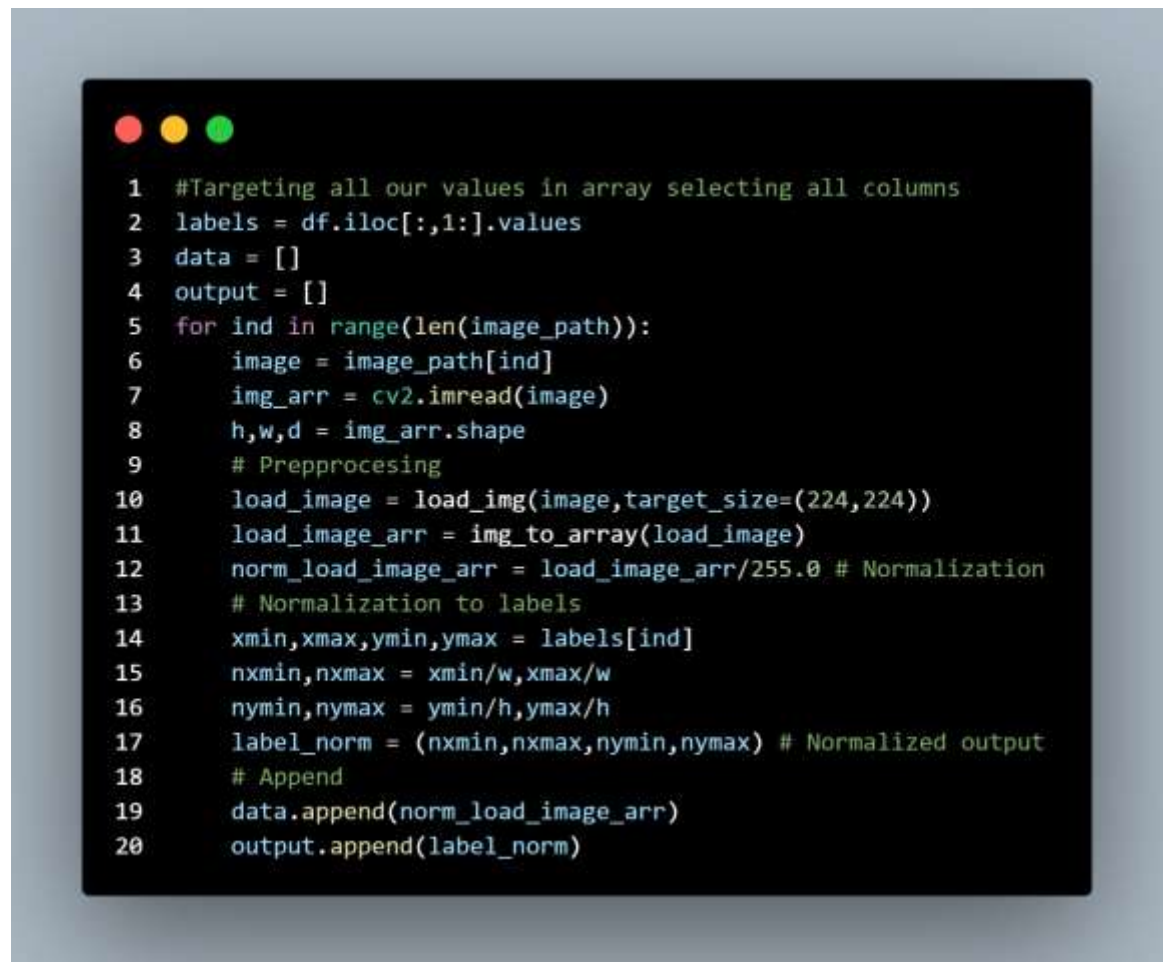


3.2.3.1.6 Xử lý dữ liệu

- Đọc dữ liệu

Đây là một bước rất quan trọng, trong quá trình này, chúng ta sẽ lấy từng hình ảnh và chuyển đổi nó thành một mảng bằng OpenCV và thay đổi kích thước hình ảnh thành 224 x 224, đây là kích thước tương thích tiêu chuẩn của mô hình

học chuyển giao được đào tạo trước.



Sau đó, sẽ chuẩn hóa hình ảnh chỉ bằng cách chia cho số tối đa vì số tối đa cho hình ảnh 8 bit là $28 - 1 = 255$. Đó là lý do sẽ chia hình ảnh của mình 255.0 . Cách lặn mảng có giá trị lớn nhất gọi là Chuẩn hóa (Min-Max Scaler). Chúng ta cũng cần bình thường hóa nhãn của mình. Bởi vì đối với mô hình deep learning, phạm vi đầu ra phải nằm trong khoảng từ 0 đến 1. Để chuẩn hóa nhãn, chúng ta cần chia các điểm chéo với chiều rộng và chiều cao của hình ảnh. Và cuối cùng là giá trị trong danh sách python.

- Tách và kiểm tra

Trong bước tiếp theo, chúng ta sẽ chuyển đổi danh sách thành một mảng bằng Numpy.

```
1 # Convert data to array
2 X = np.array(data,dtype=np.float32)
3 y = np.array(output,dtype=np.float32)
```

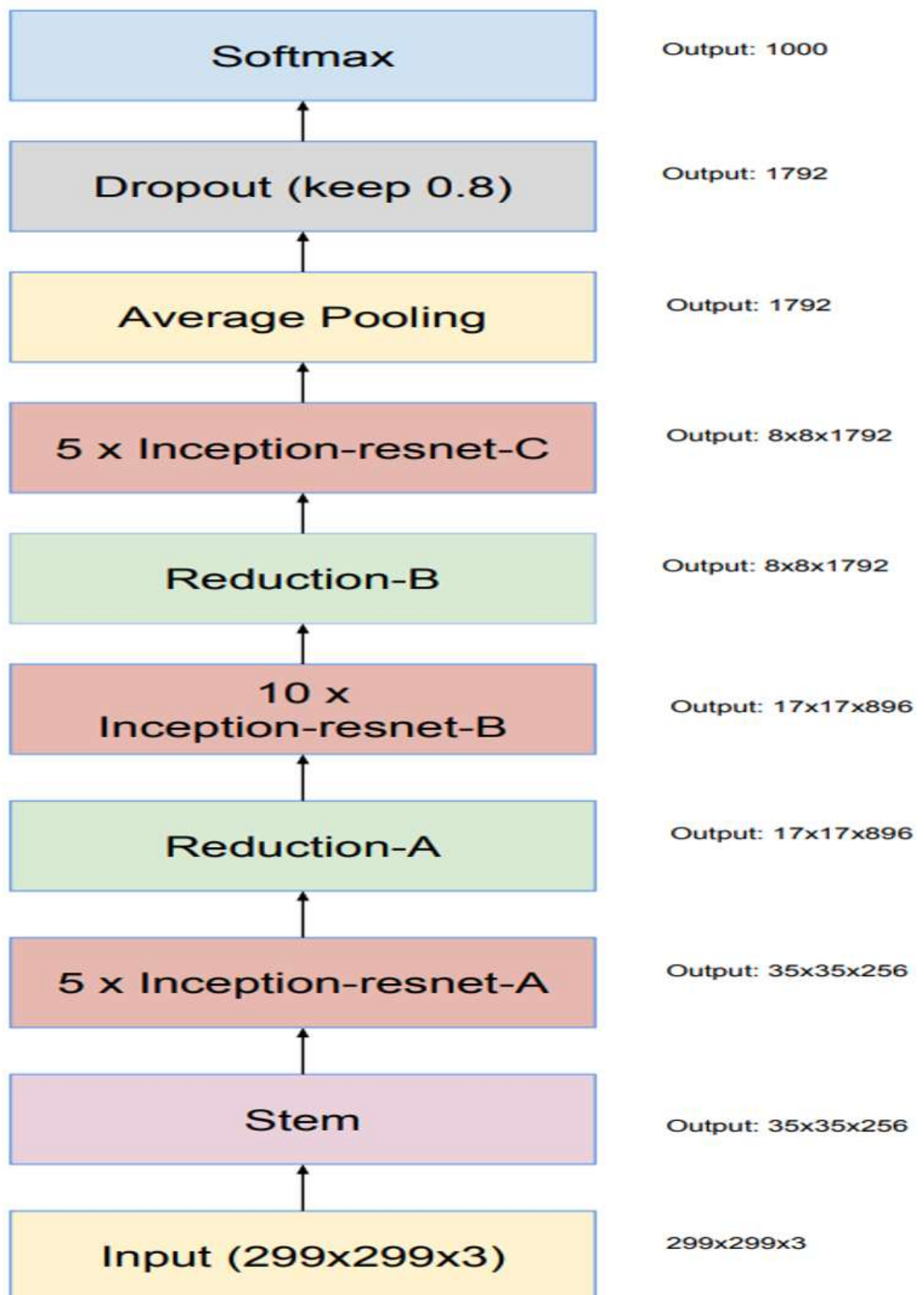
Bây giờ hãy chia dữ liệu thành tập huấn luyện và tập kiểm tra bằng sklearn.

```
1 # Split the data into training and testing set using sklearn.
2 x_train,x_test,y_train,y_test = train_test_split(X,y,train_size=0.8,random_state=0)
3 x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Out: ((180, 224, 224, 3), (45, 224, 224, 3), (180, 4), (45, 4))

3.2.3.1.7 *Học sâu để phát hiện đối tượng*

Inception-ResNet-v2 là mạng thần kinh tích chập được đào tạo trên hơn một triệu hình ảnh từ cơ sở dữ liệu ImageNet. Mạng sâu 164 lớp và có thể phân loại hình ảnh thành 1000 loại đối tượng, chẳng hạn như bàn phím, chuột, bút chì và nhiều loài động vật. Kết quả là mạng đã học được cách biểu diễn tính năng phong phú cho nhiều loại hình ảnh. Inception-ResNet-v2 được sử dụng cho nhiệm vụ phân loại. Inception-Resnet-v2 được xây dựng dựa trên sự kết hợp giữa cấu trúc Inception và kết nối Residual. Trong khối Inception-Resnet, nhiều bộ lọc tích chập có kích thước được kết hợp bằng các kết nối dư. Việc sử dụng các kết nối residual không chỉ tránh được vấn đề xuống cấp do các cấu trúc sâu gây ra mà còn giảm thời gian huấn luyện.



Ở đây, chúng ta sẽ sử dụng mô hình Inception-ResNet-v2 với các trọng số được huấn luyện trước và huấn luyện mô hình này cho dữ liệu của chúng tôi. Trước đây chúng ta đã nhập các thư viện cần thiết từ TensorFlow, hãy tiếp tục.


```

1 Model: "model"
2
3 Layer (type)                Output Shape                Param #    Connected to
4 -----
5 input_1 (InputLayer)        [(None, 224, 224, 3)] 0
6
7 conv2d (Conv2D)             (None, 111, 111, 32) 864    input_1[0][0]
8
9 batch_normalization (BatchNorm (None, 111, 111, 32) 96    conv2d[0][0]
10
11 activation (Activation)      (None, 111, 111, 32) 0    batch_normalization[0][0]
12
13 conv2d_1 (Conv2D)           (None, 109, 109, 32) 9216    activation[0][0]
14
15 batch_normalization_1 (BatchNor (None, 109, 109, 32) 96    conv2d_1[0][0]
16
17 activation_1 (Activation)    (None, 109, 109, 32) 0    batch_normalization_1[0][0]
18
19 conv2d_2 (Conv2D)           (None, 109, 109, 64) 18432    activation_1[0][0]
20
21 batch_normalization_2 (BatchNor (None, 109, 109, 64) 192    conv2d_2[0][0]
22
23 activation_2 (Activation)    (None, 109, 109, 64) 0    batch_normalization_2[0][0]
24
25 max_pooling2d (MaxPooling2D) (None, 54, 54, 64) 0    activation_2[0][0]
26
27 conv2d_3 (Conv2D)           (None, 54, 54, 80) 5120    max_pooling2d[0][0]
28
29 batch_normalization_3 (BatchNor (None, 54, 54, 80) 240    conv2d_3[0][0]
30
31 activation_3 (Activation)    (None, 54, 54, 80) 0    batch_normalization_3[0][0]
32
33 conv2d_4 (Conv2D)           (None, 52, 52, 192) 138240    activation_3[0][0]
34
35 batch_normalization_4 (BatchNor (None, 52, 52, 192) 576    conv2d_4[0][0]
36
37 activation_4 (Activation)    (None, 52, 52, 192) 0    batch_normalization_4[0][0]

```

```

1 tfb = TensorBoard('object_detection')
2 history = model.fit(x=x_train,y=y_train,batch_size=10,epochs=180,
3                     validation_data=(x_test,y_test),callbacks=[tfb])

```

```

1 2022-08-17 11:07:09.212581: I tensorflow/core/profiler/lib/profiler_session.cc:131] Profiler session initializing.
2 2022-08-17 11:07:09.212638: I tensorflow/core/profiler/lib/profiler_session.cc:146] Profiler session started.
3 2022-08-17 11:07:09.214679: I tensorflow/core/profiler/internal/gpu/cupti_tracer.cc:1614] Profiler found 1 GPUs
4 2022-08-17 11:07:09.557052: I tensorflow/core/profiler/lib/profiler_session.cc:164] Profiler session tear down.
5 2022-08-17 11:07:09.557231: I tensorflow/core/profiler/internal/gpu/cupti_tracer.cc:1748] CUPTI activity buffer flushed.
6 2022-08-17 11:07:10.429360: I tensorflow/compiler/mlir/mlir_graph_optimization_pass.cc:185] None of the MLIR Optimization Passes are enabled (registered 2)

```

[illegible]

```
1 model.save('./object_detection.h5')
```


3.2.3.1.8 TENSORBOARD

Hãy xem xét các đại lượng vô hướng trên TensorBoard. Chúng ta có thể thấy trên các biểu đồ vô hướng. Tập huấn luyện và xác thực không có hành vi phù hợp quá mức và sự mất đối với các kỷ nguyên là ít hơn.



3.2.3.1.9 Mô hình phát hiện đối tượng

Đây là bước cuối cùng trong việc phát hiện đối tượng. Trong bước này, chúng ta sẽ kết hợp tất cả lại với nhau và đưa ra dự đoán cho một hình ảnh nhất định.

```
1 # Load model
2 model = tf.keras.models.load_model('./object_detection.h5')
3 print('Model loaded Sucessfully')
```

Tiếp theo là tải hình ảnh THỬ NGHIỆM với đường dẫn phù hợp tới nó. Một số hình ảnh - thư mục TEST.

```

1 path = '../input/number-plate-detection/TEST/TEST.jpeg'
2 image = load_img(path) # PIL object
3 image = np.array(image,dtype=np.uint8) # 8 bit array (0,255)
4 image1 = load_img(path,target_size=(224,224))
5 image_arr_224 = img_to_array(image1)/255.0 # Convert into array and get the normalized output
6
7 # Size of the original image
8 h,w,d = image.shape
9 print('Height of the image =',h)
10 print('Width of the image =',w)

```


```

1 fig = plt.imshow(image)
2 fig.update_layout(width=700, height=500, margin=dict(l=10, r=10, b=10, t=10), xaxis_title='Figure 13 - TEST Image')

```




Đầu ra



```
1 # Make predictions
2 coords = model.predict(test_arr)
3 coords
```

Out: array([[0.35251912, 0.63733816, 0.65963215, 0.7349673]], dtype=float32)

Chúng ta đã nhận được đầu ra từ mô hình và đầu ra nhận được là đầu ra được chuẩn hóa. Vì vậy, điều cần làm là chuyển đổi trở lại các giá trị dạng ban đầu, điều các giá trị mà thực sự đã làm trong quá trình đào tạo, trong quá trình đào tạo, chúng ta có các giá trị dạng ban đầu và chuyển đổi giá trị chuẩn hóa đó. Vì vậy, về cơ bản, chúng ta sẽ phi chuẩn hóa lại.



```
1 # Denormalize the values
2 denorm = np.array([w,w,h,h])
3 coords = coords * denorm
4 coords
```

Out : array([[317.6197314 , 574.24168348, 480.8718347 , 535.7911554]])

Giới hạn

Bây giờ chúng ta sẽ vẽ giới hạn lên trên hình ảnh.



```
1 coords = coords.astype(np.int32)
2 coords
```

Out : array([[317, 574, 480, 535]], dtype=int32)



```
1 # Draw bounding on top the image
2 xmin, xmax,ymin,ymax = coords[0]
3 pt1 =(xmin,ymin)
4 pt2 =(xmax,ymax)
5 print(pt1, pt2)
```



```
1 cv2.rectangle(image,pt1,pt2,(0,255,0),3)
2 fig = px.imshow(image)
3 fig.update_layout(width=700, height=500, margin=dict(l=10, r=10, b=10, t=10))
```



Tạo Pipeline

```
1 # Create pipeline
2 path = '../input/number-plate-detection/TEST/TEST.jpeg'
3 def object_detection(path):
4
5     # Read image
6     image = load_img(path) # PIL object
7     image = np.array(image,dtype=np.uint8) # 8 bit array (0,255)
8     image1 = load_img(path,target_size=(224,224))
9
10    # Data preprocessing
11    image_arr_224 = img_to_array(image1)/255.0 # Convert to array & normalized
12    h,w,d = image.shape
13    test_arr = image_arr_224.reshape(1,224,224,3)
14
15    # Make predictions
16    coords = model.predict(test_arr)
17
18    # Denormalize the values
19    denorm = np.array([w,w,h,h])
20    coords = coords * denorm
21    coords = coords.astype(np.int32)
22
23    # Draw bounding on top the image
24    xmin, xmax,ymin,ymax = coords[0]
25    pt1 =(xmin,ymin)
26    pt2 =(xmax,ymax)
27    print(pt1, pt2)
28    cv2.rectangle(image,pt1,pt2,(0,255,0),3)
29    return image, coords
30
31 image, cods = object_detection(path)
32
33 fig = px.imshow(image)
34 fig.update_layout(width=700, height=500, margin=dict(l=10, r=10, b=10, t=10),xaxis_title='Figure 14')
```



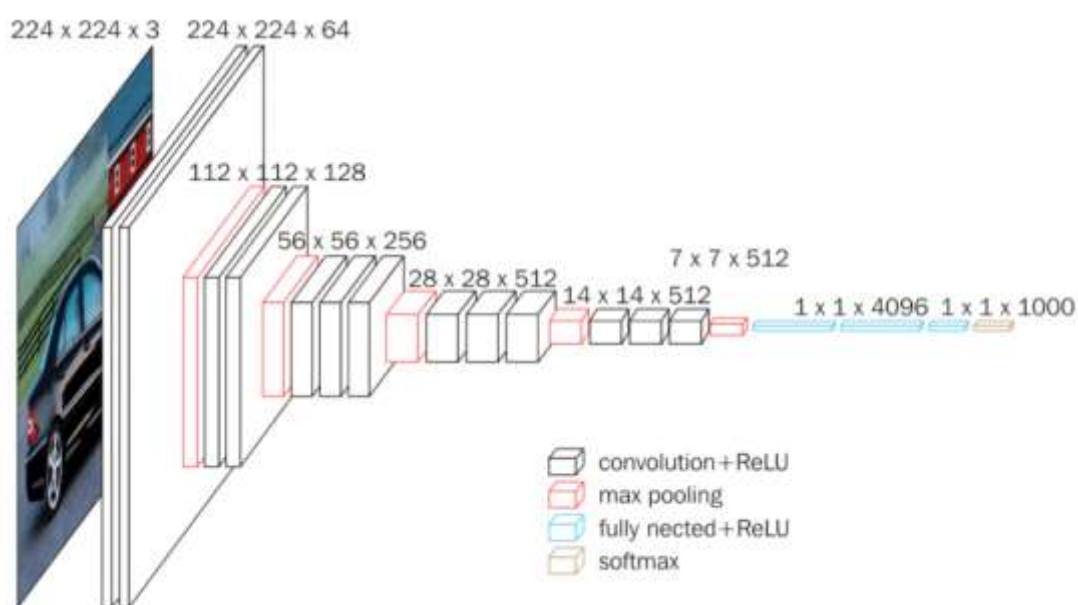

3.2.3.1.10 So sánh YOLOv8 vs Inception ResNet V2

Tiêu chí	YOLOv8	Inception ResNet V2
Mục đích chính	Phát hiện đối tượng	Phân loại hình ảnh
Kiến trúc	CNN với các head phát hiện đối tượng và các thành phần FPN/PAFPN cho việc hợp nhất đặc trưng	Kết hợp Inception Module và Residual Connections
Tốc độ	Rất nhanh (real-time detection)	Chậm hơn nhiều, chủ yếu cho classification
Độ chính xác	Cao đối với phát hiện đối tượng (mAP cao)	Cao đối với phân loại hình ảnh (Top-1 Accuracy)
Kích thước mô hình	Nhỏ hơn, tối ưu cho các thiết bị có giới hạn tài nguyên	Lớn hơn, đòi hỏi tài nguyên tính toán cao
Ứng dụng	Phát hiện đối tượng thời gian thực trong video, hình ảnh	Phân loại ảnh, nhận diện đối tượng trong các tập dữ liệu lớn
Độ phức tạp huấn luyện	Đơn giản hơn, có thể huấn luyện từ đầu hoặc fine-tune trên các tập dữ liệu phát hiện đối tượng	Phức tạp hơn, yêu cầu nhiều tài nguyên để huấn luyện từ đầu
Residual Connections	Có, nhưng trong kiến trúc khác, không tập trung như trong Inception ResNet V2	Sử dụng nhiều, làm nổi bật khả năng truyền tải thông tin ở mạng sâu
Chức năng chính	Bounding box detection và phân loại đối tượng	Phân loại hình ảnh với độ chính xác cao
Khả năng mở rộng	Cao, có thể tùy chỉnh kích thước mô hình (YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, YOLOv8x)	Không được thiết kế cho nhiều phiên bản mở rộng
Yêu cầu tài nguyên tính	Thấp hơn, phù hợp với các ứng dụng nhúng hoặc thiết bị di động	Cao hơn, cần GPU mạnh cho huấn luyện và inference

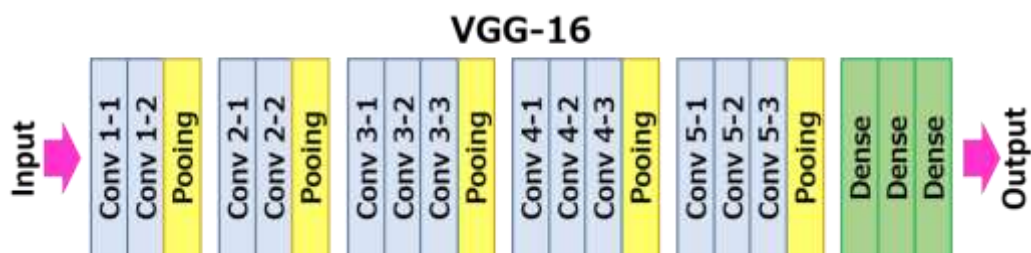
3.2.3.2 VGG16

3.2.3.2.1 Giới thiệu về VGG16

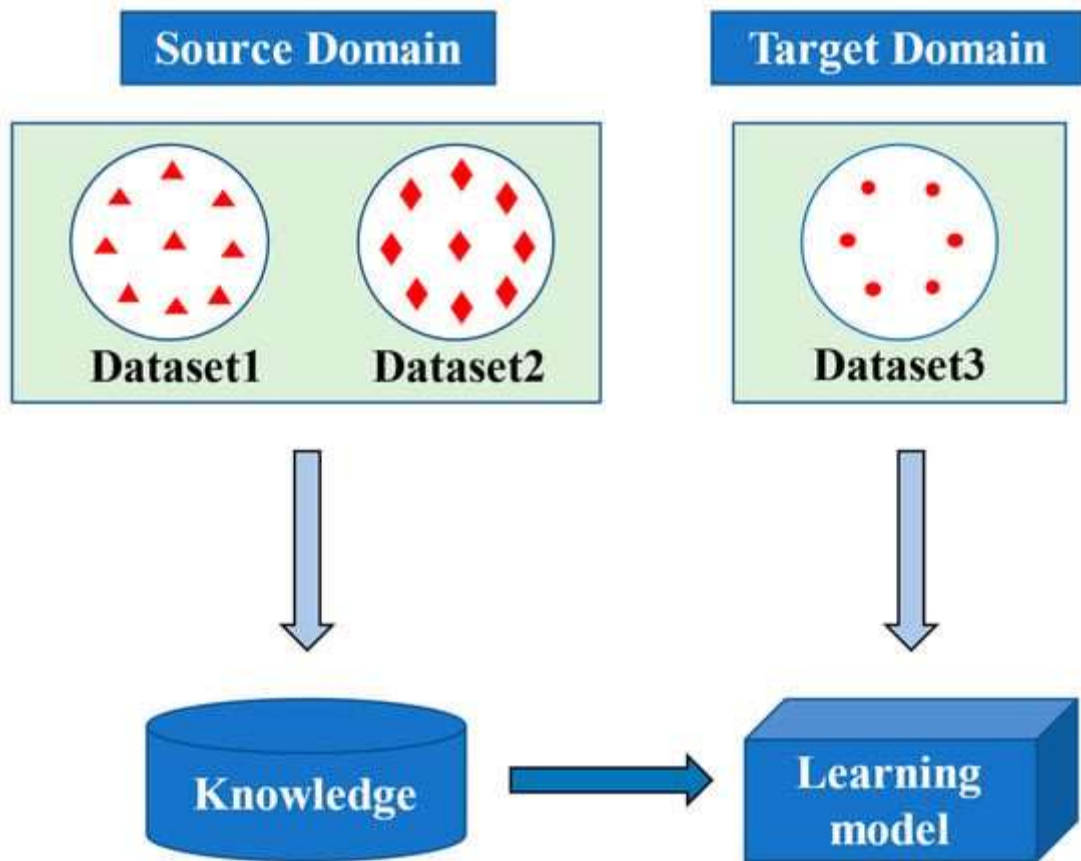
VGG16, vốn nổi tiếng với khả năng nhận diện hình ảnh, có thể được điều chỉnh để nhận diện biển số xe. Mô hình này cải tiến từ AlexNet bằng cách thay thế các bộ lọc lớn (11 và 5) bằng nhiều bộ lọc 3x3 nhỏ hơn, giúp thu thập thông tin chi tiết về các ký tự và con số trên biển số. Các bộ lọc 1x1 được sử dụng để kết hợp thông tin từ các kênh khác nhau, tăng cường khả năng nhận diện các mẫu phức tạp. Cấu trúc của VGG16, với 5 lớp tổng hợp tối đa xen kẽ với các lớp tích chập, cho phép mô hình giảm dần kích thước dữ liệu đồng thời giữ lại thông tin quan trọng về vị trí và hình dạng của biển số. Hai lớp fully connected cuối cùng, kết hợp với softmax, giúp phân loại biển số dựa trên các đặc trưng đã học. Học chuyển tiếp, kỹ thuật tận dụng kiến thức từ các mô hình đã được huấn luyện trên tập dữ liệu lớn (ví dụ, nhận diện chữ số), có thể được áp dụng để tăng hiệu suất nhận diện biển số, đặc biệt khi tập dữ liệu biển số hạn chế. Bằng cách "mượn" kiến thức từ các mô hình nhận diện ký tự, VGG16 có thể học cách nhận diện biển số hiệu quả hơn với ít dữ liệu huấn luyện hơn.



Mô hình VGG16



Mô hình VGG16 tóm tắt



Minh họa chuyển giao

Sau khi trích xuất đặc trưng bằng VGG16, lớp fully connected với 1024 đơn vị được sử dụng để phân loại biến số. Số lượng đơn vị này đủ lớn để học đặc trưng, nhưng không quá lớn gây overfitting (mô hình học quá chi tiết tập huấn luyện, không khái quát tốt dữ liệu mới). Để giảm overfitting, lớp Dropout được thêm vào sau lớp fully connected. Dropout ngẫu nhiên "tắt" một số đơn vị trong quá trình huấn luyện, buộc mô hình học đặc trưng phân tán hơn, tăng khả năng khái quát hóa.

3.2.3.2.2 Thực nghiệm kết quả.

```
# Create the model
# Using the functional API to specify input shape explicitly
input_tensor = Input(shape=(IMAGE_SIZE, IMAGE_SIZE, 3))
vgg_output = VGG16(weights="imagenet", include_top=False)(input_tensor)
flatten_output = Flatten()(vgg_output)

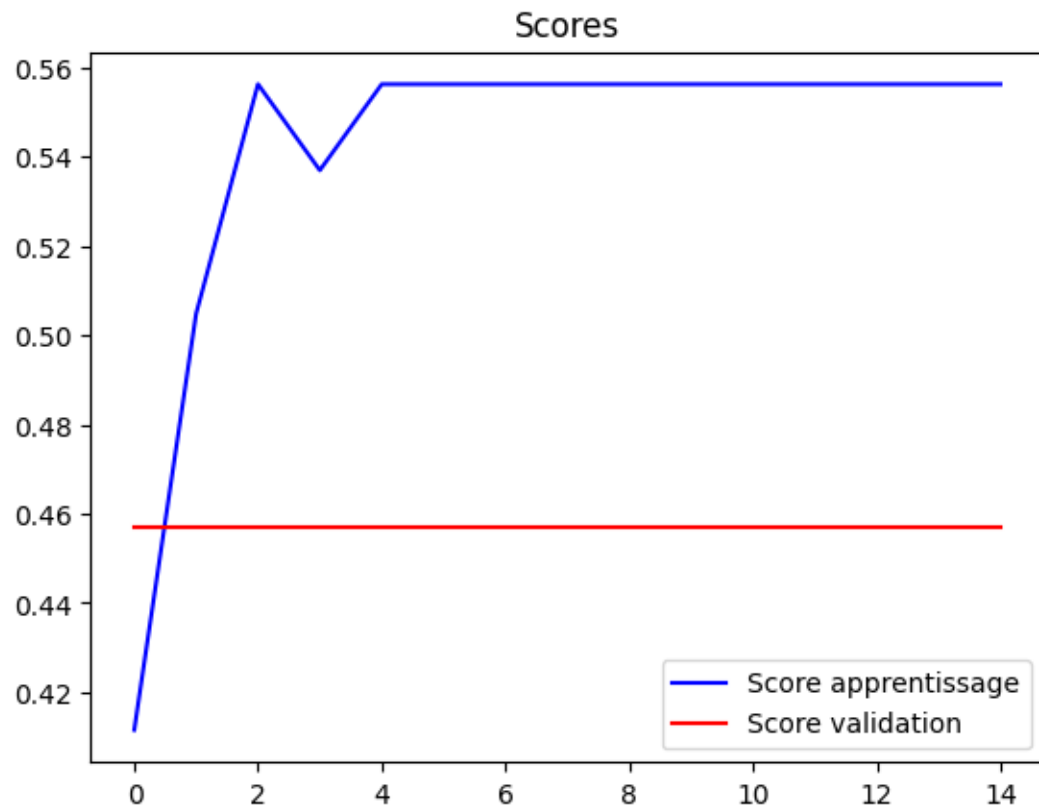
model = Sequential()
model.add(input_tensor)
model.add(VGG16(weights="imagenet", include_top=False, input_shape=(IMAGE_SIZE, IMAGE_SIZE, 3)))
model.add(Flatten()) # The Flatten layer will now receive a defined input shape from VGG16
model.add(Dense(128, activation="relu"))
model.add(Dense(128, activation="relu"))
model.add(Dense(64, activation="relu"))
model.add(Dense(4, activation="sigmoid"))

model.layers[1].trainable = False # Adjust the index to match the VGG16 layer

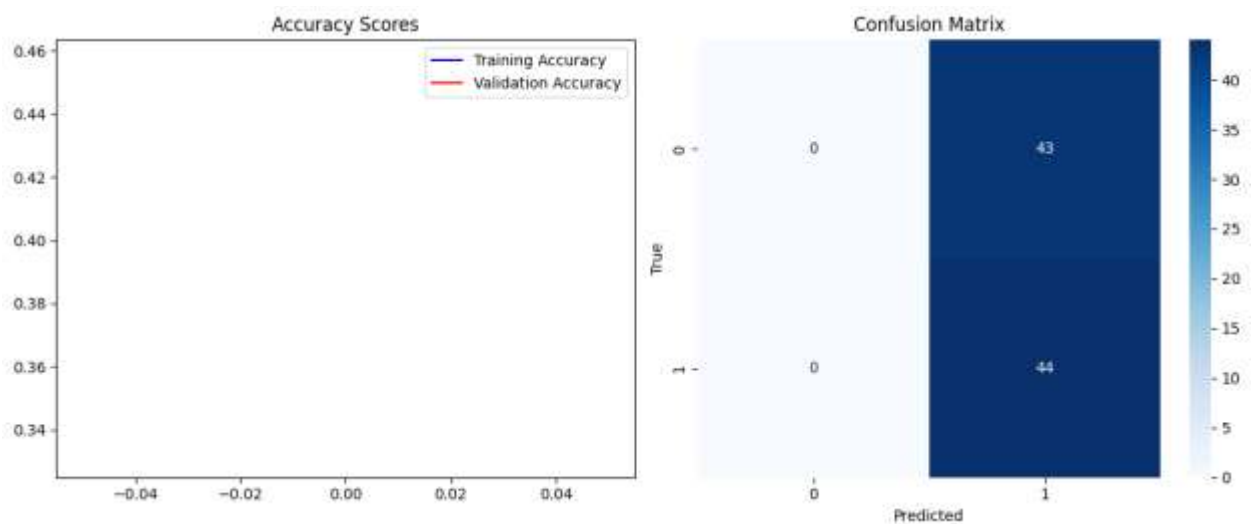
model.summary()
model.compile(loss='mean_squared_error', optimizer='adam', metrics=['accuracy'])
train = model.fit(X_train, y_train, validation_data=(X_val, y_val), epochs=15, batch_size=32, verbose=1)
```

Định nghĩa mô hình VGG16

Quá trình thử nghiệm và huấn luyện được thực hiện trên nền tảng Google Colaboratory (Colab). Colab cung cấp môi trường máy chủ đám mây miễn phí với cấu hình mạnh mẽ, bao gồm CPU, RAM, và GPU (như Tesla T4), đáp ứng tốt nhu cầu tính toán cho các tác vụ học máy. Hệ thống được cài đặt môi trường Python mới nhất, cùng các frameworks và thư viện phổ biến như numpy, matplotlib, tensorflow, keras,... giúp đơn giản hóa việc tổ chức dữ liệu, chạy thử nghiệm và lưu trữ kết quả. Chương trình thử nghiệm được xây dựng trên môi trường Python và sử dụng frameworks TensorFlow với giao diện thư viện Keras, một thư viện mạnh mẽ hỗ trợ xử lý ảnh và xây dựng mô hình mạng nơ-ron.

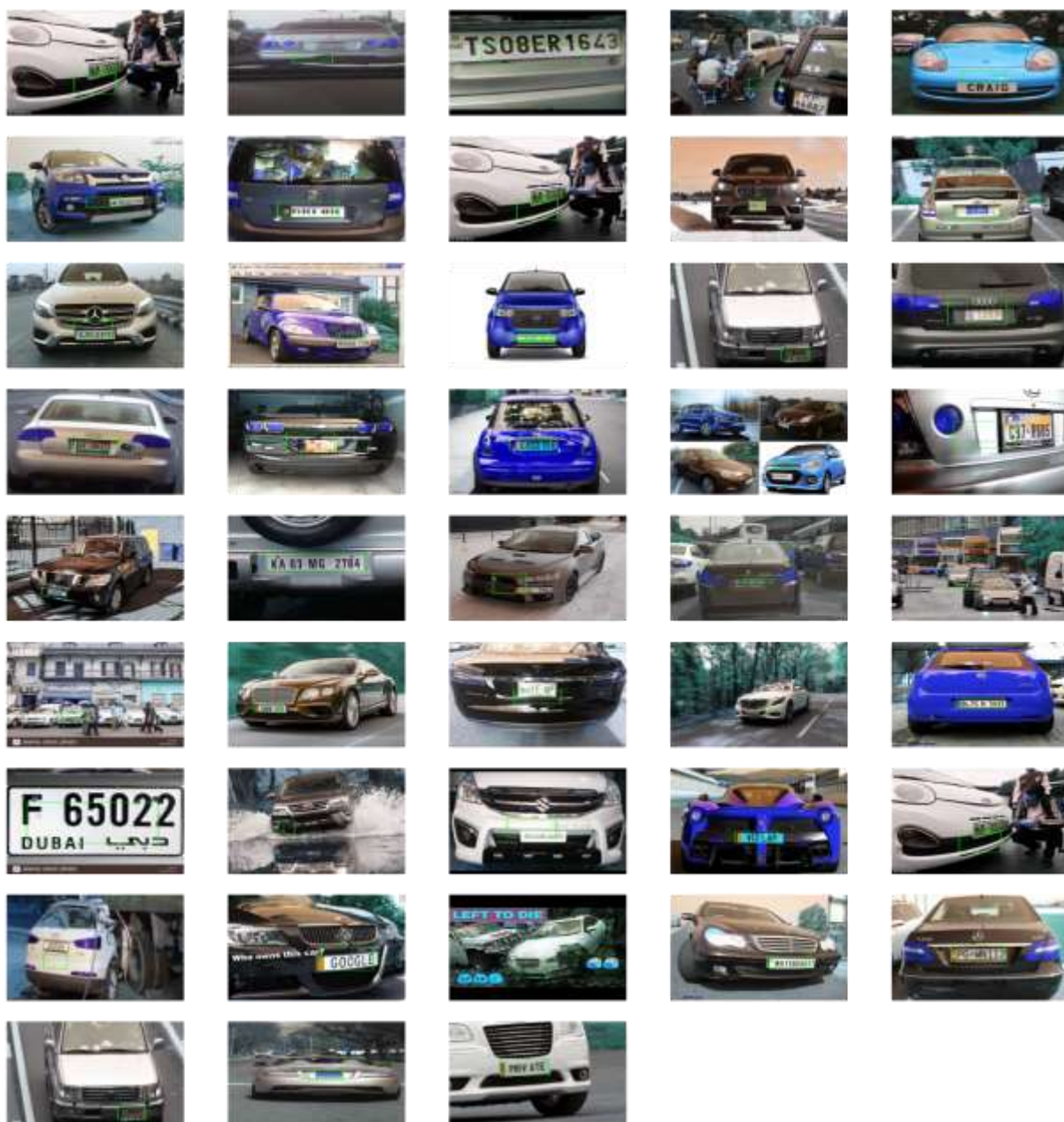


Biểu đồ VGG16



Biểu đồ Confusion Matrix

Kết quả quá trình huấn luyện VGG16 trên tập dữ liệu được thể hiện trong hình 13. Đồ thị thể hiện khi huấn luyện dữ liệu sử dụng VGG16. Kết quả thể hiện được hiệu quả trên số epoch và độ chính xác trong quá trình huấn luyện. Đây là kết quả 15 lần chạy thử nghiệm. Kết quả trên tập dữ liệu để cho kết quả độ chính xác phân lớp (accuracy) tốt hơn so với mô hình VGG16.



Kết quả sau khi chạy thử

3.2.3.2.3 Kết luận

Trong lĩnh vực giao thông thông minh, nhận diện biển số xe đóng vai trò quan trọng trong việc quản lý và giám sát phương tiện. Hệ thống nhận diện biển số xe tự động trích xuất thông tin từ hình ảnh biển số, chuyển đổi thành dạng văn bản để xử lý và lưu trữ. Nghiên cứu này tập trung vào việc xây dựng hệ thống nhận diện biển số xe thời gian thực bằng các phương pháp phân loại hình ảnh. Hệ thống bao gồm ba bước chính: (1) thu thập hình ảnh biển số xe, (2) xử lý và trích xuất đặc trưng từ ảnh, (3) phân loại biển số bằng các mô hình học sâu. Hai mô hình được đánh giá là CNN và VGG16 với học chuyển tiếp. VGG16, với kiến trúc sâu và khả năng học chuyển tiếp từ các tập dữ liệu lớn, được kỳ vọng mang lại hiệu suất cao hơn. Học

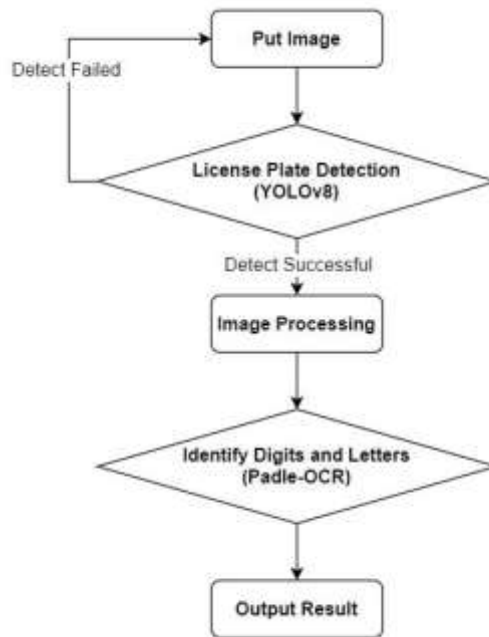
chuyển tiếp cho phép mô hình tận dụng kiến thức từ các bài toán nhận diện ký tự, từ đó cải thiện khả năng nhận diện biển số. Kết quả thử nghiệm trên tập dữ liệu gồm nhiều loại biển số xe cho thấy cả hai mô hình đều đạt độ chính xác cao. VGG16 với học chuyển tiếp cho kết quả vượt trội hơn, chứng minh hiệu quả của việc tận dụng kiến thức từ các bài toán liên quan. Nghiên cứu này là tiền đề cho việc phát triển các ứng dụng nhận diện biển số xe trong thực tiễn, chẳng hạn như:

- **Hệ thống quản lý bãi đỗ xe thông minh:** tự động nhận diện biển số, quản lý ra vào, tính phí.
- **Hệ thống thu phí giao thông tự động:** nhận diện biển số để thu phí không dừng.
- **Hệ thống giám sát an ninh:** phát hiện và cảnh báo các phương tiện vi phạm giao thông, phương tiện bị truy nã.

Với sự phát triển của công nghệ học sâu, hệ thống nhận diện biển số xe sẽ ngày càng chính xác và hiệu quả, góp phần nâng cao hiệu quả quản lý giao thông và an ninh đô thị.

3.2.4 Xây dựng hệ thống nhận diện biển số xe

Kiến trúc hệ thống:



Quy trình nhận dạng biển số xe, bao gồm các bước sau:

- Put Image: Hình ảnh đầu vào chứa biển số xe.
- License Plate Detection (YOLOv8): Sử dụng mô hình YOLOv8 để phát hiện biển số xe trong hình ảnh.
 - o YOLOv8 là một mô hình học sâu được huấn luyện để phát hiện đối tượng, trong trường hợp này là biển số xe.
 - o Mô hình này sẽ xác định vị trí của biển số trong ảnh và khoanh vùng nó.
- Detect Successful/Detect Failed:
 - o Nếu YOLOv8 phát hiện được biển số, quy trình sẽ chuyển sang bước tiếp theo ("Detect Successful").
 - o Nếu không phát hiện được, quy trình sẽ kết thúc ("Detect Failed").
- Image Processing: Xử lý hình ảnh biển số đã được khoanh vùng ở bước trước để chuẩn bị dữ liệu cho bước nhận dạng ký tự tiếp theo. Các bước xử lý có thể bao gồm:

- Cắt (crop) vùng biển số từ ảnh gốc.
 - Chuyển đổi ảnh sang ảnh xám.
 - Nâng cao độ tương phản.
 - Loại bỏ nhiễu.
- Identify Digits and Letters (Paddle-OCR): Sử dụng công cụ Paddle-OCR để nhận dạng các ký tự (chữ số và chữ cái) trên biển số.
 - Paddle-OCR là một công cụ mã nguồn mở phổ biến để nhận dạng ký tự quang học (OCR).
 - Công cụ này sẽ phân tích hình ảnh biển số và chuyển đổi nó thành chuỗi văn bản.
- Output Result: Kết quả cuối cùng là chuỗi văn bản chứa thông tin biển số xe.


```

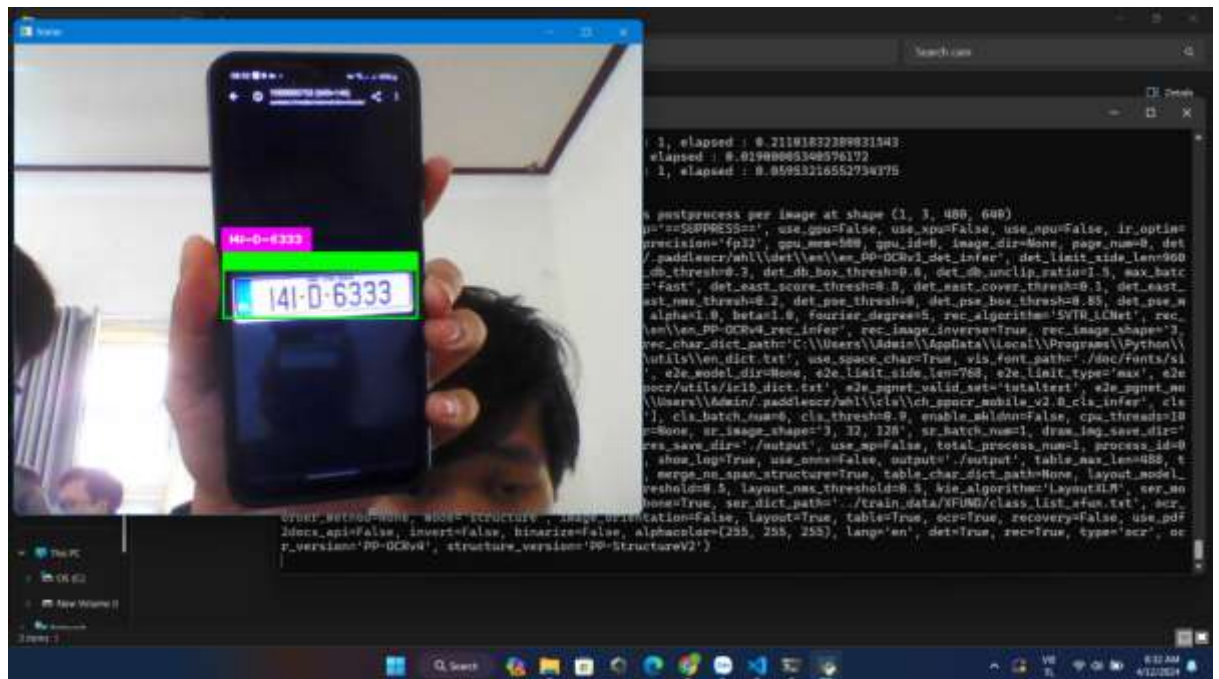
1 from ultralytics import YOLO
2 from paddleocr import PaddleOCR
3 app = Flask(__name__)
4 # Initialize YOLO
5 trained = YOLO('best.pt')
6 # Initialize PaddleOCR
7 ocr = PaddleOCR(use_angle_cls=True, lang='en')
8 def gen_frames():
9     vid = cv2.VideoCapture(0)
10    vid.set(3, 1000)
11    vid.set(4, 480)
12    vid.set(10, 150)
13
14    while True:
15        success, frame = vid.read()
16        if not success:
17            break
18        else:
19            result = trained(frame)
20            frame = draw_box(result, frame)
21            ret, buffer = cv2.imencode('.jpg', frame)
22            frame = buffer.tobytes()
23            yield (b'--frame\r\n'
24                  + b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
25
26 def get_plates(result, img):
27     images = [] # Store all license plates
28     boxes = result[0].boxes # List of all coordinates of license plates
29     img = img.copy()
30     for b in boxes:
31         x1 = int(b.xyxy[0][0])
32         y1 = int(b.xyxy[0][1])
33         x2 = int(b.xyxy[0][2])
34         y2 = int(b.xyxy[0][3])
35         images.append(img[y1:y2, x1:x2].copy())
36     return images
37
38 def format_ocr(plate_text):
39     result = ''
40
41     if len(plate_text[0]) == 2:
42         first_line = plate_text[0][0][1][0]
43         second_line = plate_text[0][1][1][0]
44         results = first_line + second_line
45         result = result.join(results)
46
47     elif len(plate_text[0]) == 1:
48         second_line = plate_text[0][0][1][0]
49         result = second_line
50
51     if len(result) < 8:
52         return None
53     return result
54
55 def paddleocr_predict(plate):
56     result = ocr.ocr(plate, cls=True)
57     number = result
58     return number
59
60 def get_IP_number(result, img):
61     plates = get_plates(result, img)
62     plate_numbers = [] # Store all IP number
63
64     for plate in plates:
65         number = paddleocr_predict(plate)
66         plate_numbers.append(number)
67
68     return plate_numbers
69
70 def draw_box(result, img):
71     boxes = result[0].boxes # All coordinates of plates
72     plate_numbers = get_IP_number(result, img) # All predicted IP number
73
74     # For each IP coordinates and each predicted IP number of that IP
75     for b, pnum in zip(boxes, plate_numbers):
76         x1 = int(b.xyxy[0][0])
77         y1 = int(b.xyxy[0][1]) - 20 # Small adjust to make it looks better
78         x2 = int(b.xyxy[0][2])
79         y2 = int(b.xyxy[0][3])
80
81         # Draw rectangle around the IP
82         cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)
83
84         # Fill background of the predicted IP number
85         cv2.rectangle(img, (x1, y1 + 22), (x2, (y1)), (0, 255, 0), -1)
86         text_xy = (x1 + 2, y1 + 18) # Coordinate of predicted IP number
87         text = format_ocr(pnum)
88         cv2.putText(img, f'{text}', text_xy, cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 0), 2)
89
90     return img
91
92 @app.route('/')
93 def index():
94     return render_template('cam.html')

```

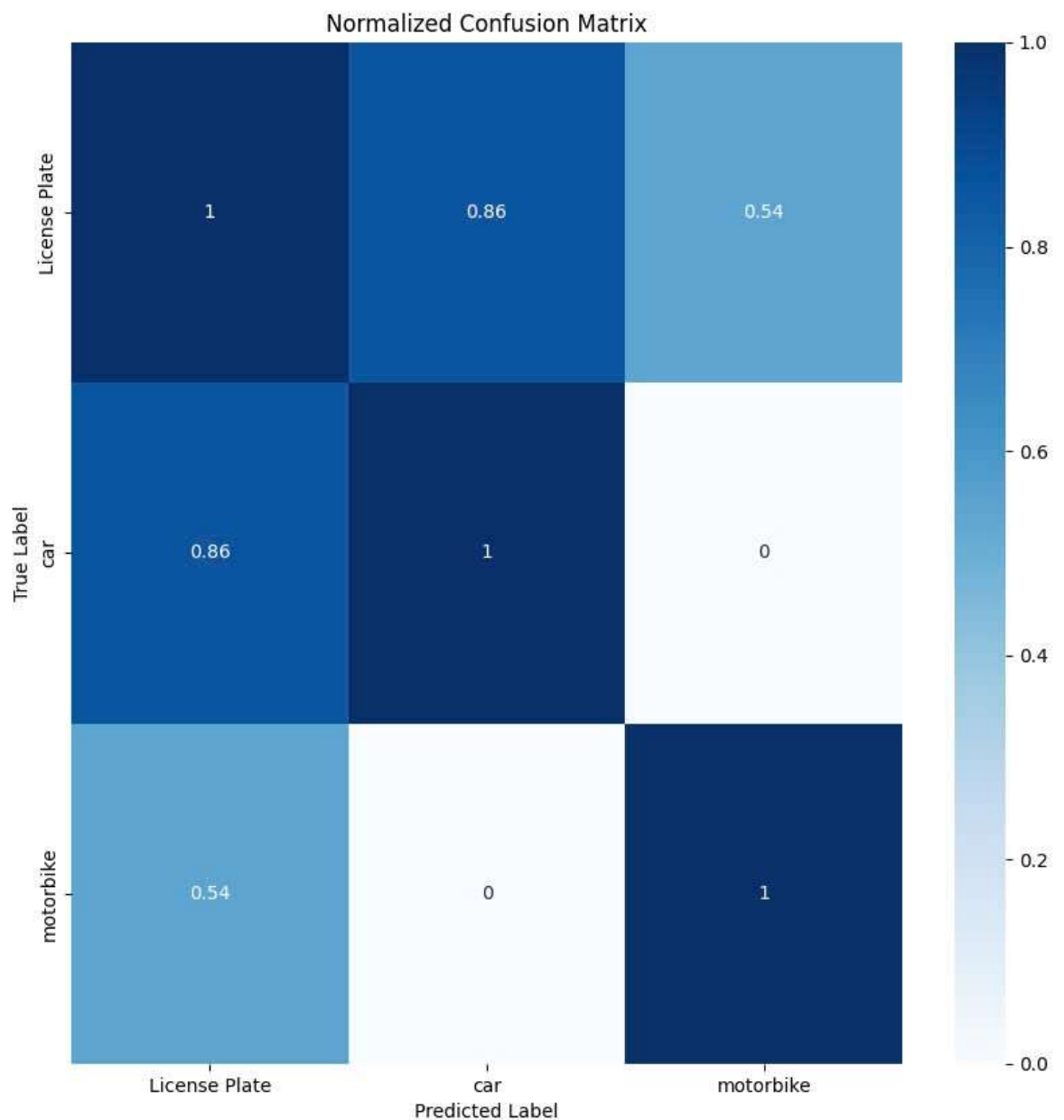
Sau khi đã xác định được cấu trúc hệ thống, bọn em sẽ xây dựng một chương trình nhận diện biển số xe cơ bản:

Đầu tiên, code khởi tạo mô hình YOLOv8 đã được huấn luyện trước đó và công cụ PaddleOCR. Sau đó, nó truy cập webcam và liên tục đọc các khung hình video. Với mỗi khung hình, code sử dụng YOLOv8 để phát hiện biển số xe, trích xuất vùng biển số và sử dụng PaddleOCR để nhận dạng các ký tự trên biển số. Cuối cùng, nó vẽ bounding box xung quanh biển số và hiển thị kết quả nhận dạng trên khung hình video. Đoạn code này có thể được sử dụng để xây dựng một ứng dụng nhận dạng biển số xe thời gian thực, chẳng hạn như trong hệ thống quản lý bãi đỗ xe hoặc kiểm soát giao thông.





3.3 Sau khi train mô hình



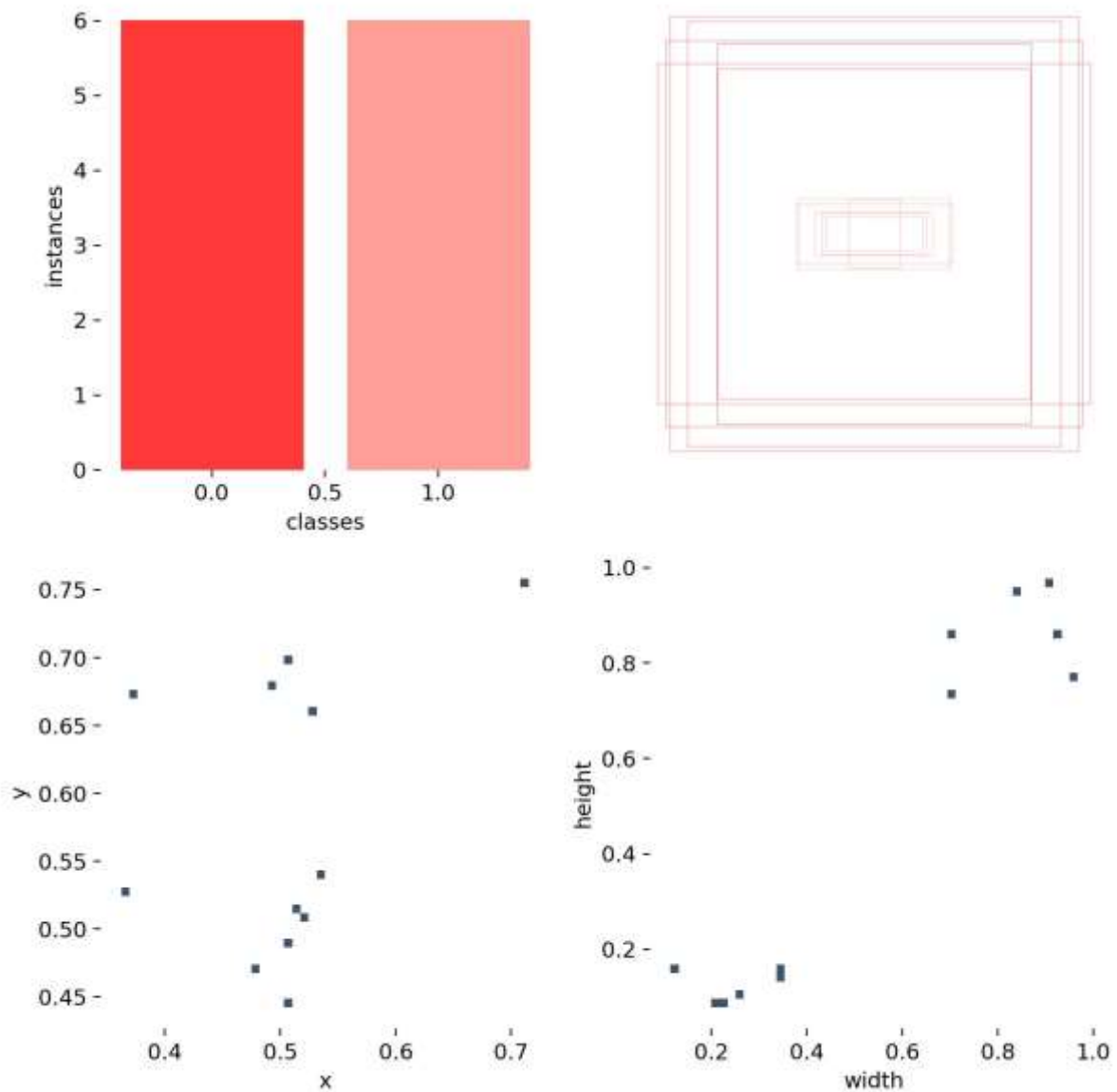
Confusion matrix

Biểu đồ trên hiển thị một ma trận (confusion matrix), cho thấy hệ thống nhận dạng biển số xe hoạt động rất tốt trong việc nhận dạng chính xác từng loại đối tượng: biển số, xe hơi và xe máy, với tỷ lệ chính xác 100% cho mỗi loại. Tuy nhiên, hệ thống vẫn còn gặp khó khăn trong việc phân biệt biển số xe với xe hơi và xe máy. Cụ thể, 86% trường hợp biển số xe bị nhầm lẫn với xe hơi và 54% bị nhầm lẫn với xe máy. Điều này cho thấy hệ thống có thể gặp khó khăn khi biển số xe có hình dạng hoặc đặc điểm tương đồng với hai loại xe này. Để cải thiện hiệu suất, cần tập trung vào việc tăng cường dữ liệu huấn luyện, đặc biệt là những

trường hợp biển số xe dễ gây nhầm lẫn, và tinh chỉnh mô hình để tăng khả năng phân biệt giữa các lớp đối tượng.

Công thức toán học để đánh giá mô hình trên:

- TP (Dương tính thật): Mô hình dự đoán có biển số và thực tế đúng là có.
- FP (Dương tính giả): Mô hình dự đoán có biển số nhưng thực tế không có.
- FN (Âm tính giả): Mô hình dự đoán không có biển số nhưng thực tế có.
- TN (Âm tính thật): Mô hình dự đoán không có biển số và thực tế đúng là không có.
- Độ chính xác (Accuracy) (3): Tỷ lệ phần trăm các kết quả dự đoán đúng (cả có biển số và không có biển số) trên tổng số dự đoán.
 - $\text{Độ chính xác} = (TP + TN) / (TP + FP + FN + TN)$
- Độ chuẩn xác (Precision) (4): Tỷ lệ phần trăm các trường hợp dự đoán đúng là có biển số trên tổng số lần mô hình dự đoán có biển số.
 - $\text{Độ chuẩn xác} = TP / (TP + FP)$
- Tỷ lệ phát hiện (Detection rate) (5): Tỷ lệ phần trăm các trường hợp dự đoán đúng là có biển số trên tổng số trường hợp thực tế có biển số trong tập dữ liệu.
 - $\text{Tỷ lệ phát hiện} = TP / (TP + FN)$



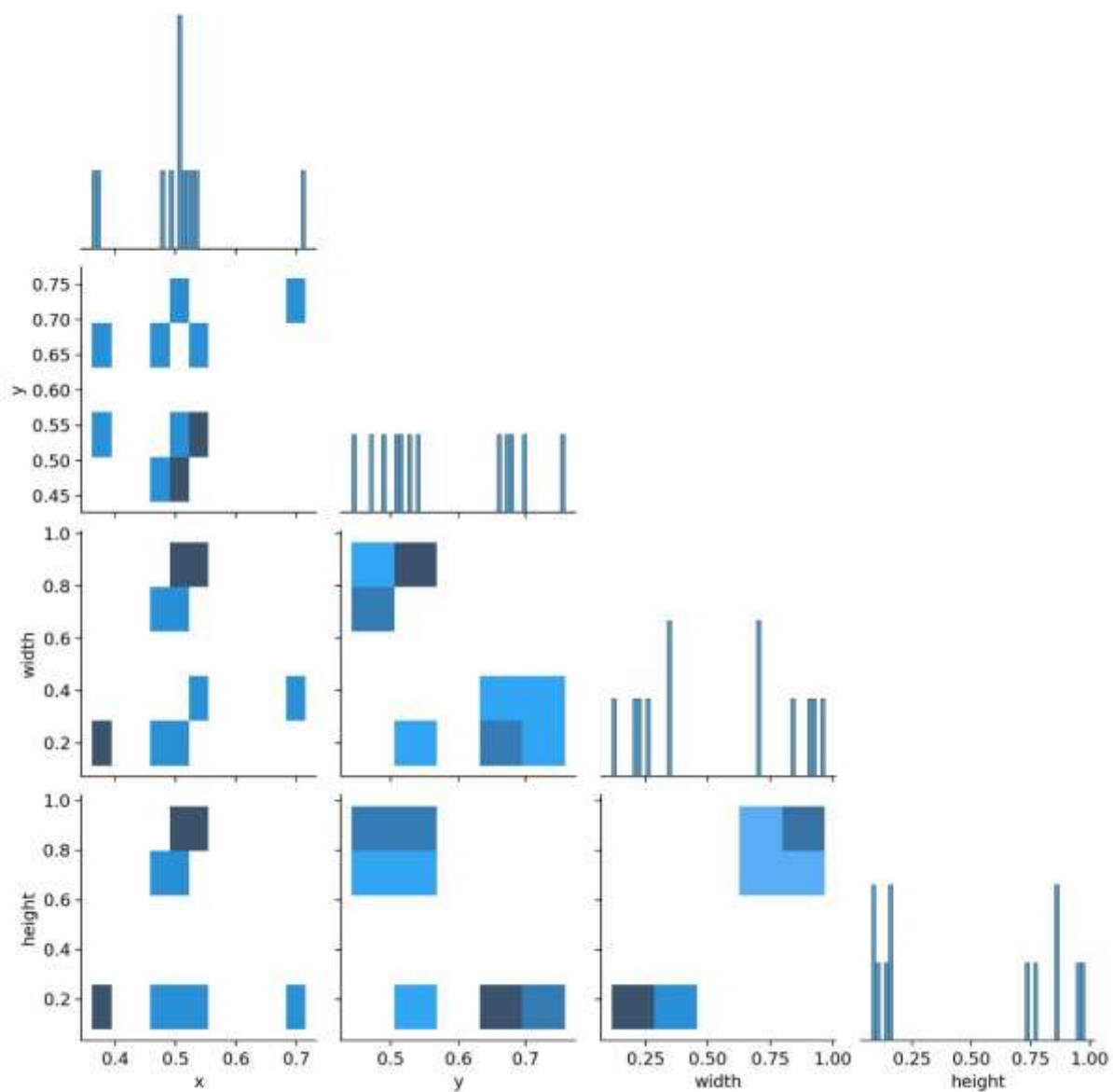
Labels

Biểu đồ có hai trục:

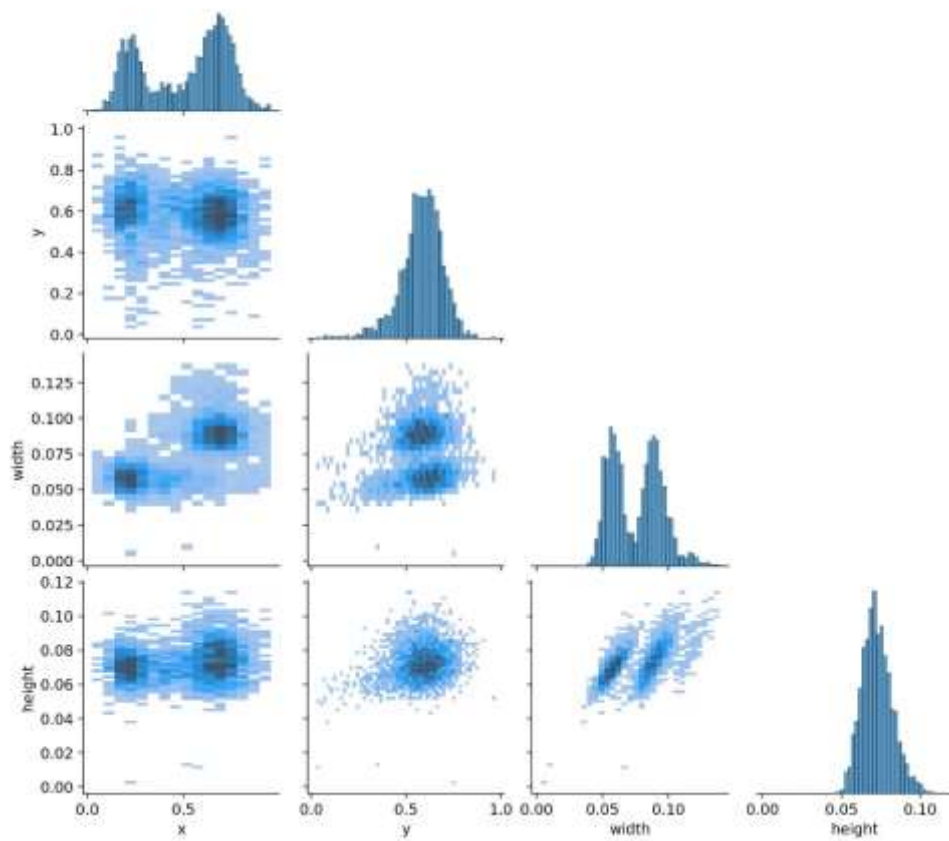
Trục x: Biểu thị độ rộng của các đối tượng trong ảnh (width).

Trục y: Biểu thị số lượng các đối tượng có độ rộng tương ứng (cases).

Biểu đồ cho thấy số lượng các đối tượng trong tập dữ liệu. Biểu đồ cũng cho thấy các đối tượng trong tập dữ liệu có độ rộng đa dạng. Vùng tô màu xanh lam và đỏ cho thấy tỷ lệ phần trăm các đối tượng có độ rộng nằm trong khoảng từ 0 đến 1. Điều này cho thấy rằng hầu hết các đối tượng trong cả hai tập dữ liệu đều có độ rộng nằm trong khoảng này.

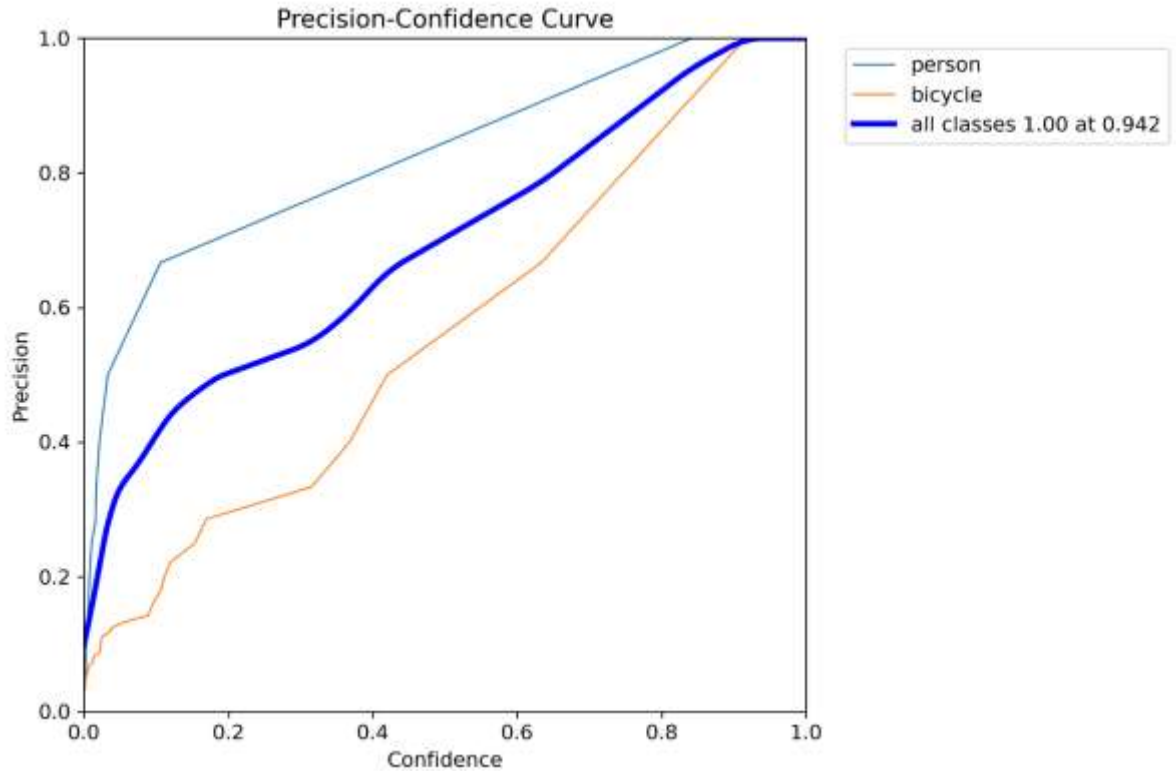


Labels correlogram



matrix scatter plot

Biểu đồ hiển thị có thể nhận diện bien số xe rõ nhất khi có chiều rộng là 0.6 và chiều cao là 0.75



P curve

Biểu đồ cho thấy độ chính xác của mô hình đối với người cao hơn so với phương tiện di chuyển. Biểu đồ cũng cho thấy độ chính xác của mô hình tăng lên khi độ tự tin của mô hình tăng lên. Đường cong màu đen cho thấy độ chính xác của mô hình đối với tất cả các lớp nằm giữa đường cong màu xanh lam và đường cong màu đỏ. Điều này là do độ chính xác của mô hình đối với tất cả các lớp là trung bình của độ chính xác của mô hình đối với từng lớp.

Độ chính xác cao ở mức độ tự tin cao: Điều này cho thấy mô hình có thể dự đoán chính xác biển số xe.

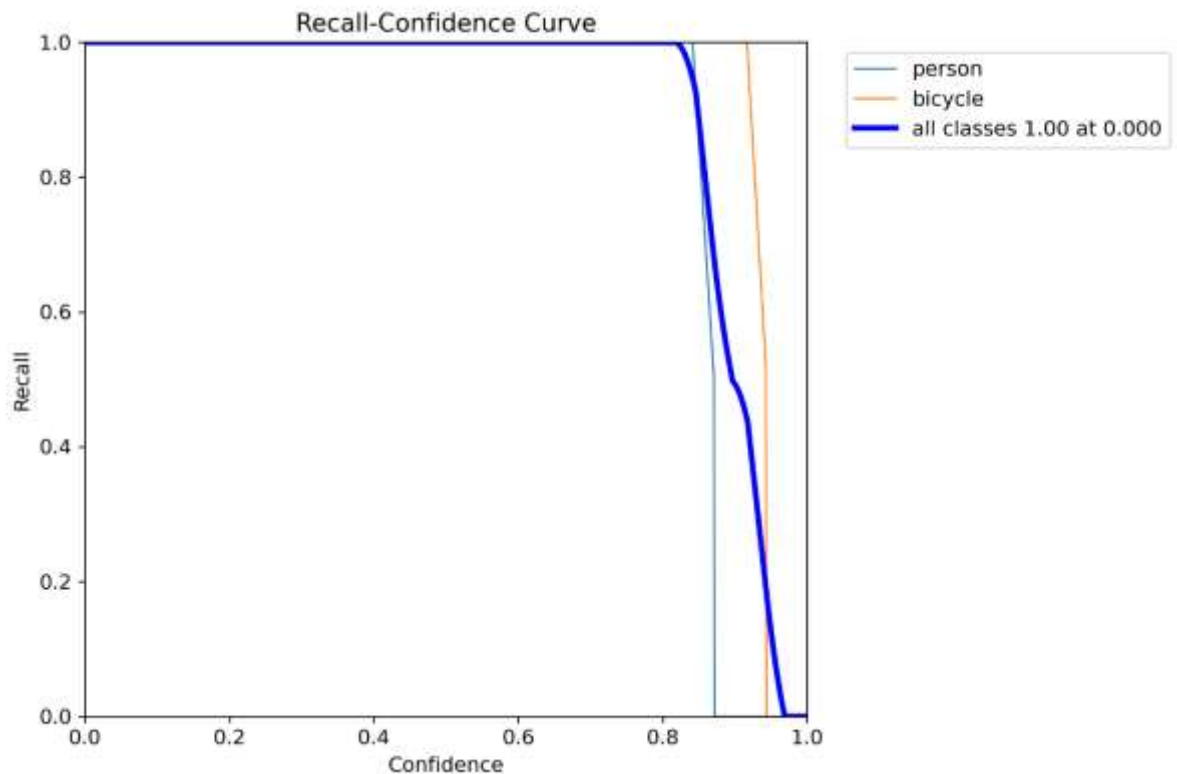
Độ chính xác thấp ở mức độ tự tin thấp: Điều này cho thấy mô hình có thể dự đoán sai biển số xe.

Biểu đồ Precision-Confidence Curve cũng có thể được sử dụng để điều chỉnh ngưỡng dự đoán của mô hình. Ngưỡng dự đoán là một giá trị được sử dụng để quyết định xem một dự đoán có được coi là chính xác hay không. Ví dụ, nếu ngưỡng dự đoán được đặt thành 0,9, thì mô hình chỉ sẽ dự đoán biển số xe khi nó tự tin 90% về dự đoán của mình.

Độ chính xác (Precision): Là tỷ lệ số lượng dự đoán chính xác trong số tất cả các dự đoán được thực hiện. Ví dụ, nếu mô hình dự đoán 100 biển số xe và 90 trong số đó là chính xác, thì độ chính xác của mô hình là 90%.

Độ tự tin (Confidence): Là mức độ chắc chắn của mô hình về dự đoán của mình. Độ tự tin thường được thể hiện dưới dạng một số từ 0 đến 1, với 0 là không chắc chắn và 1 là chắc chắn.

Ngưỡng dự đoán (Prediction Threshold): Là một giá trị được sử dụng để quyết định xem một dự đoán có được coi là chính xác hay không. Ví dụ, nếu ngưỡng dự đoán được đặt thành 0,9, thì mô hình chỉ sẽ dự đoán biển số xe khi nó tự tin 90% về dự đoán của mình.



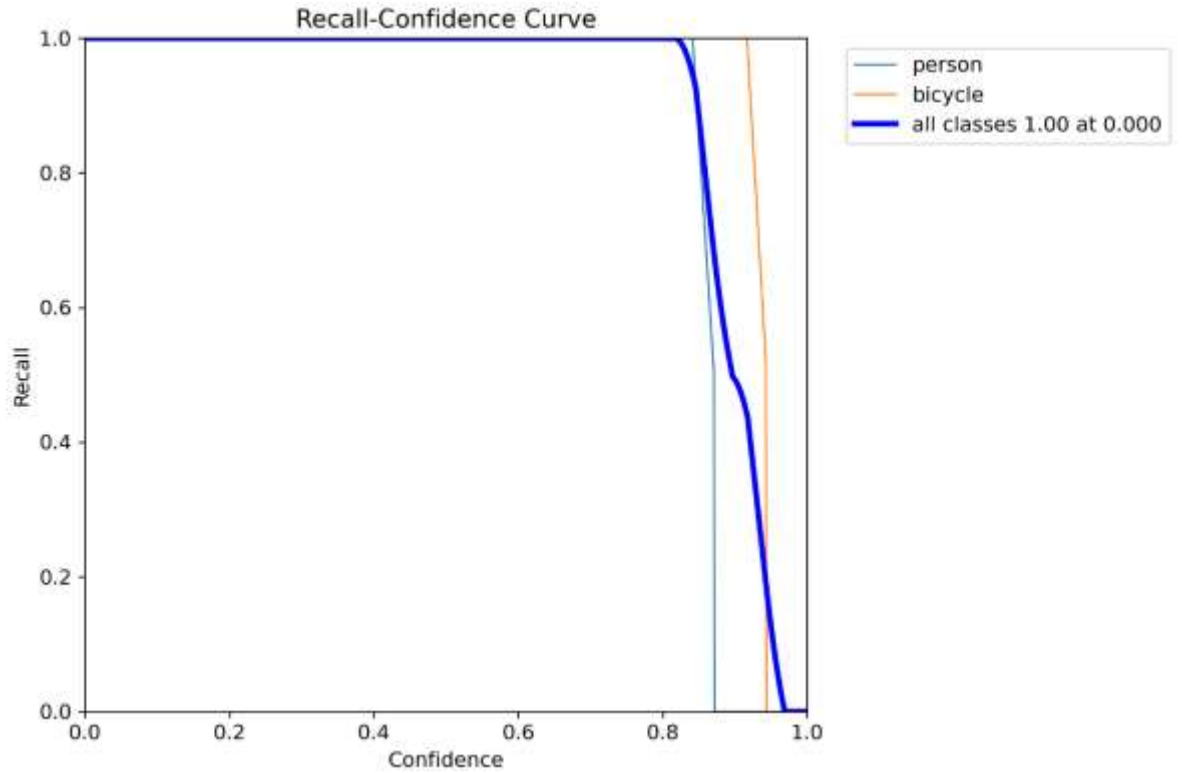
R curve

Biểu đồ Recall-Confidence Curve được sử dụng để đánh giá hiệu suất của mô hình nhận diện biển số xe:

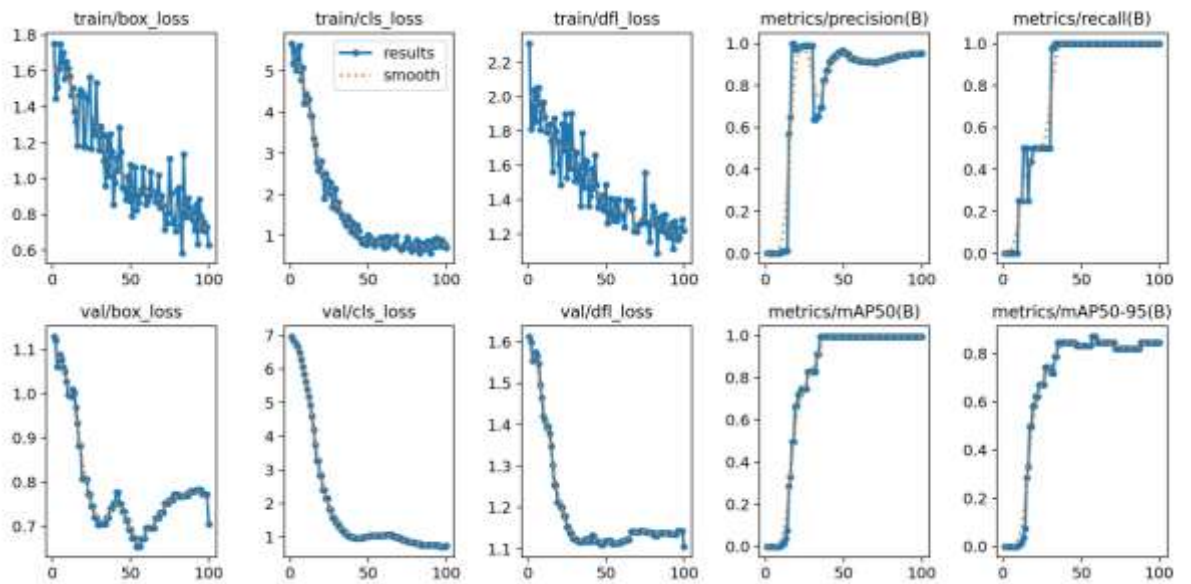
Độ bao quát cao ở mức độ tự tin cao: Điều này cho thấy mô hình có thể phát hiện hầu hết các biển số xe khi nó tự tin về dự đoán của mình.

Độ bao quát thấp ở mức độ tự tin thấp: Điều này cho thấy mô hình có thể bỏ sót một số biển số xe khi nó không tự tin về dự đoán của mình.

Biểu đồ Recall-Confidence Curve cũng được sử dụng để điều chỉnh ngưỡng dự đoán của mô hình. Ngưỡng dự đoán là một giá trị được sử dụng để quyết định xem một dự đoán có được coi là chính xác hay không. Ví dụ, nếu ngưỡng dự đoán được đặt thành 0,9, thì mô hình chỉ sẽ dự đoán biển số xe khi nó tự tin 90% về dự đoán của mình.



Results



training curves

Các hàm mất mát (loss function) như train/box_loss, train/cls_loss và train/dfl_loss đều giảm dần, cho thấy mô hình đang học cách xác định vị trí và phân loại biển số xe chính xác hơn. Đồng thời, các chỉ số đánh giá như metrics/precision(B), metrics/recall(B), metrics/mAP50(B) và metrics/mAP50-95(B) đều tăng dần, cho thấy khả năng phát hiện và nhận dạng biển số của mô

hình ngày càng được cải thiện. Tuy nhiên, đường cong val/box_loss cho thấy có thể có hiện tượng overfitting (quá khớp) nhẹ, khi mất mát trên tập kiểm tra tăng lên sau một khoảng thời gian giảm. Điều này có thể cần được xử lý bằng các kỹ thuật như regularization (chính quy hóa) hoặc tăng cường dữ liệu huấn luyện. Nhìn chung, các đường cong huấn luyện cho thấy mô hình đang học tốt và có tiềm năng trở thành một hệ thống nhận diện biển số xe hiệu quả.

3.4 Đánh giá mô hình

Trong quá trình xây dựng hệ thống quản lý khu du lịch, ba mô hình chính được sử dụng bao gồm YOLOv8 và PaddleOCR. Mỗi mô hình có các độ chính xác và ứng dụng khác nhau, phù hợp với các mục đích cụ thể trong hệ thống. Dưới đây là đánh giá chi tiết về các mô hình này.

Mô hình	Độ chính xác	Ứng dụng
Yolov8	76.15	Nhận diện biển số xe
PaddleOCR	99	Nhận diện ký tự trong hình ảnh

- YOLOv8
 - Độ chính xác: 76.15%
 - Ứng dụng: Nhận diện biển số xe
 - Đánh giá: YOLOv8 là phiên bản cải tiến của dòng mô hình YOLO, được sử dụng cho nhiệm vụ nhận diện bãi đỗ xe. Với độ chính xác 76.15%, YOLOv8 thể hiện khả năng mạnh mẽ trong việc nhận diện và đếm số lượng phương tiện trong thời gian thực. Mô hình này được ứng dụng để theo dõi và đếm số lượng phương tiện ra vào bãi đỗ xe thông qua các camera giám sát, cung cấp dữ liệu quan trọng cho việc lập kế hoạch và quản lý. Khả năng xử lý nhanh và độ chính xác cao của YOLOv8 giúp cải thiện hiệu quả quản lý và đáp ứng yêu cầu của các hoạt động kinh doanh trong khu du lịch.
- PaddleOCR
 - Độ chính xác: 99%
 - Ứng dụng: Nhận diện biển số xe
 - Đánh giá: PaddleOCR là một mô hình nhận diện ký tự quang học (OCR) mạnh mẽ, đặc biệt hiệu quả trong việc nhận diện biển số xe với độ chính xác lên tới 99%. Trong hệ thống quản lý bãi đỗ xe, PaddleOCR giúp tự động hóa quá trình kiểm tra và ghi nhận biển số xe khi vào và ra khỏi bãi đỗ, giảm thiểu sai sót và tăng cường hiệu quả vận hành. Khả năng nhận diện nhanh và

chính xác của mô hình này không chỉ giúp tiết kiệm thời gian mà còn cung cấp dữ liệu đáng tin cậy cho việc theo dõi và quản lý bãi đỗ xe.