

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KỸ THUẬT MÁY TÍNH



ĐỒ ÁN 2

**ỨNG DỤNG YOLOV8 VÀ OCR ĐỂ NHẬN
DIỆN BIỂN SỐ XE**

**YOLOV8 AND OCR APPLICATION
FOR LICENSE PLATE RECOGNITION**

NGÀNH KỸ THUẬT MÁY TÍNH

GIẢNG VIÊN HƯỚNG DẪN

TH.S LÊ HOÀI NGHĨA

TP. HỒ CHÍ MINH, 2025

LỜI CẢM ƠN

Đầu tiên, em xin gửi lời cảm ơn đến quý thầy cô giáo trường Đại Học Công Nghệ Thông Tin. Trong quá trình học tập và rèn luyện tại trường, với sự dạy dỗ, chỉ bảo tận tình của các quý thầy cô giáo đã trang bị cho em những kiến thức về chuyên môn cũng như kỹ năng mềm, tạo cho em hành trang vững chắc trong cuộc sống cũng như công việc sau này.

Tiếp theo, em xin cảm ơn khoa Kỹ Thuật Máy Tính vì đã luôn tạo mọi điều kiện thuận lợi cho em được học tập và phát triển.

Đặc biệt, để hoàn thành môn học này, em xin gửi lời cảm ơn tới Thầy Lê Hoài Nghĩa đã tận tình chỉ bảo, hướng dẫn và hỗ trợ trang thiết bị cho em trong suốt thời gian thực hiện đề tài.

Thành phố Hồ Chí Minh, tháng 1 năm
2025

Sinh viên thực hiện

Đỗ La Quốc Tuấn – 21521621

Trần Gia Kiệt – 2152264

MỤC LỤC

LỜI CẢM ƠN	2
MỤC LỤC	3
MỞ ĐẦU	1
CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI	2
1.1. Giới thiệu đề tài	2
1.2. Mục đích nghiên cứu	2
1.3. Nội dung nghiên cứu	2
1.5. Ý nghĩa thực tiễn	3
CHƯƠNG 2: CƠ SỞ LÝ THUYẾT	4
2.1. Mô hình YOLO.....	4
2.1.1. Giới thiệu	4
2.1.1. Lịch sử, cách thức học máy	4
2.1.2. Ưu và nhược điểm.....	7
2.2. Mô hình YOLOv8n.....	8
2.2.1. Đặc điểm chính	8
2.2.2. Ứng dụng.....	8
2.2.3. Ưu điểm và nhược điểm	9
2.3. Thư viện EasyOCR	9
2.4. Google Colab	10
CHƯƠNG 3: HUẤN LUYỆN MÔ HÌNH	12
3.1. Thu thập dữ liệu	12
3.1.1. Tìm kiếm và lựa chọn dữ liệu	12
3.1.2. Gán nhãn dữ liệu	12
3.1.4. Chia bộ dữ liệu.....	13
3.1.5. Xuất và tích hợp dữ liệu	14
3.2. Huấn luyện mô hình	14
3.2.1. Cài đặt môi trường.....	14
3.2.2. Tiến hành huấn luyện.....	15
CHƯƠNG 4: QUI TRÌNH NHẬN DIỆN BIỂN SỐ XE.....	17
4.1. Nhận diện vùng chứa biển số xe	17
4.2. Quá trình đọc biển số xe.....	19
4.3. Chương trình nhận diện biển số xe từ video	21

CHƯƠNG 5: KẾT QUẢ VÀ ĐÁNH GIÁ.....	24
5.1. Kết quả đạt được.....	24
5.2. Tổng kết	25
5.3. Kết quả thống kê	25
CHƯƠNG 6: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	27
6.1. Kết luận	27
6.2. Hướng phát triển	27
TÀI LIỆU THAM KHẢO	28

MỞ ĐẦU

Trong thời đại cách mạng công nghệ 4.0, việc ứng dụng trí tuệ nhân tạo (AI) và học sâu (Deep Learning) vào các lĩnh vực đời sống đã và đang mang lại những thay đổi to lớn, góp phần nâng cao hiệu quả và tự động hóa các quy trình. Một trong những ứng dụng nổi bật là nhận diện biển số xe, hỗ trợ quản lý giao thông, bãi đỗ xe thông minh và hệ thống an ninh.

Đề án “Nhận diện biển số xe bằng YOLOv8” được thực hiện với mục tiêu áp dụng các tiến bộ mới nhất trong lĩnh vực thị giác máy tính để phát triển một hệ thống nhận diện nhanh chóng, chính xác và hiệu quả. YOLOv8, một phiên bản cải tiến của mô hình YOLO (You Only Look Once), mang đến khả năng xử lý hình ảnh vượt trội nhờ tốc độ và độ chính xác cao, giúp nhận diện các đối tượng trong thời gian thực.

Trong báo cáo này, nhóm em sẽ trình bày quá trình nghiên cứu, triển khai và đánh giá hệ thống nhận diện biển số xe. Báo cáo bao gồm các nội dung từ tìm hiểu cơ sở lý thuyết, phương pháp luận, đến thử nghiệm thực tế, nhằm mang lại cái nhìn toàn diện về hiệu quả và tiềm năng ứng dụng của hệ thống.

Nhóm chúng em hy vọng rằng nghiên cứu này không chỉ góp phần giải quyết bài toán thực tế mà còn là nguồn tham khảo hữu ích cho các nghiên cứu liên quan trong tương lai.

CHƯƠNG 1: TỔNG QUAN ĐỀ TÀI

1.1. Giới thiệu đề tài

Nhận diện biển số xe là một lĩnh vực quan trọng trong các hệ thống giao thông thông minh, bãi đỗ xe tự động và quản lý phương tiện. Trong thời đại công nghệ 4.0, nhu cầu ứng dụng các mô hình AI hiện đại như YOLO (You Only Look Once) đang tăng cao nhờ khả năng nhận diện đối tượng nhanh chóng và chính xác.

Phiên bản YOLOv8, mới nhất trong dòng mô hình YOLO, đã mang đến những tiến bộ vượt bậc trong nhận diện đối tượng, đặc biệt trong các bài toán thời gian thực. Việc nghiên cứu và ứng dụng YOLOv8 trong nhận diện biển số xe mang lại nhiều tiềm năng cải thiện hiệu quả và tối ưu các quy trình trong giao thông và quản lý.

1.2. Mục đích nghiên cứu

Mục tiêu chính của nghiên cứu là ứng dụng YOLOv8 để nhận diện biển số xe một cách nhanh chóng, chính xác và hiệu quả.

Các mục đích cụ thể bao gồm:

- Nhận diện biển số xe thông qua video.
- Đánh giá hiệu suất của YOLOv8 trong bài toán nhận diện biển số xe.
- Đề xuất các giải pháp để tối ưu hệ thống trong các điều kiện khác nhau như ánh sáng, góc chụp và đối tượng bị che khuất.

1.3. Nội dung nghiên cứu

Nghiên cứu bao gồm các nội dung chính như sau:

- Tìm hiểu nguyên lý hoạt động và các điểm nổi bật của YOLOv8.
- Thu thập, xử lý và đánh giá bộ dữ liệu huấn luyện biển số xe.
- Xây dựng mô hình nhận diện biển số xe sử dụng YOLOv8, bao gồm các giai đoạn:
 - + Huấn luyện mô hình.

- + Kiểm tra và đánh giá hiệu suất.
- + Triển khai và thử nghiệm hệ thống trong các điều kiện thực tế.

1.4. Phương pháp nghiên cứu

Phương pháp thu thập dữ liệu: Thu thập các hình ảnh biển số xe từ nhiều nguồn khác nhau như camera giao thông, video giám sát.

Phương pháp xử lý dữ liệu: Xử lý hình ảnh bao gồm chuẩn hóa độ phân giải, lọc nhiễu và gán nhãn cho dữ liệu.

Phương pháp huấn luyện mô hình: Sử dụng YOLOv8n và tối ưu các tham số để đạt hiệu suất cao nhất.

Phương pháp đánh giá: Sử dụng các chỉ số đánh giá như độ chính xác (accuracy), độ bao phủ (recall), và tốc độ xử lý để đo lường hiệu quả của mô hình.

1.5. Ý nghĩa thực tiễn

Ứng dụng trong giao thông: Hỗ trợ các hệ thống giám sát giao thông, phát hiện và xử lý vi phạm như xe chạy quá tốc độ, vượt đèn đỏ.

Quản lý bãi đỗ xe: Tự động nhận diện biển số xe để kiểm soát vào/ra tại các bãi đỗ xe thông minh.

Cải thiện an ninh: Ứng dụng trong các hệ thống giám sát an ninh để theo dõi và định danh phương tiện trong thời gian thực.

Nâng cao hiệu quả vận hành: Giảm thiểu thời gian và nhân lực trong việc nhận diện biển số xe so với phương pháp thủ công.

Tóm lại: Nghiên cứu này không chỉ góp phần đẩy mạnh ứng dụng công nghệ AI vào thực tiễn mà còn mở ra tiềm năng phát triển các hệ thống thông minh trong tương lai.

- YOLOv3 được cải tiến dựa trên YOLOv2 bởi 2 tác giả ban đầu YOLOv2, họ cùng nhau xuất bản ‘YOLOv3: An incremental Improvement’ vào ngày 8 tháng 4 năm 2018.

- YOLOv4 được phát triển bởi Alexey Bochoknovskiy, Chien-Yao Wang, and HongYuan Mark Liao vì tác giả ban đầu của YOLO đã bế tắc khi phát hành phiên bản mới, 14 tờ báo có tiêu đề ‘YOLOv4: Optimal Speed and Accuracy of Object Detection’ vào ngày 23 tháng 4 năm 2020.

- YOLOv5 được phát hành ngay sau khi YOLOv4 được phát triển dựa vào framework Pytorch bởi tác giả Glenn Jocher, mã nguồn YOLOv5 có sẵn trên GitHub được phát hành ngày 18 tháng 5 năm 2020.

- YOLOv6 được phát triển bởi Meituan vào năm 2022, một công ty công nghệ lớn tại Trung Quốc. Đây là phiên bản YOLO được tối ưu hóa dành riêng cho môi trường sản xuất thực tế.

- YOLOv7 được phát hành bởi một nhóm nhà nghiên cứu độc lập, bao gồm Chien-Yao Wang, Alexey Bochkovskiy và Hong-Yuan Mark Liao. Đây là phiên bản cải tiến mạnh mẽ, được xem là một bước đột phá so với các phiên bản trước.

- YOLOv8 được phát triển và phát hành bởi Ultralytics, kế thừa tính năng từ YOLOv5 và bổ sung nhiều cải tiến mới, giúp tăng cường hiệu suất tổng thể. Đây là phiên bản tiên tiến nhất hiện nay, với nhiều cải tiến lớn cả về kiến trúc lẫn tính linh hoạt.



Hiệu suất						
Phát hiện (COCO) Phát hiện (Open Images V7) Phân đoạn (COCO) Phân loại (ImageNet) Tư thế (COCO) OBB (DOTAv1)						
Xem Tài liệu phát hiện để biết các ví dụ sử dụng với các mô hình được đào tạo trên COCO, bao gồm 80 lớp được đào tạo trước.						
Người mẫu	kích cỡ (điểm ảnh)	giá trị mAP 50-95	Tốc độ CPU ONNX (bệnh đa xơ cứng)	Tốc độ A100 TensorRT (bệnh đa xơ cứng)	tham số (M)	Thất bại (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Hình 2.2: Sự khác biệt của các phiên bản YOLOv8

Cách thức học máy của mô hình YOLO

Hình ảnh đầu vào sẽ được Thuật toán YOLO chia thành $S \times S$ thường thì sẽ $3 \times 3, 7 \times 7, 9 \times 9, \dots$

Với dữ liệu đầu vào là 1 ảnh, đầu ra mô hình sẽ là một ma trận 3 chiều có kích thước $S \times S \times (5 \times B + C)$ với số lượng tham số mỗi ô là $(5 \times B + C)$ với B là số lượng Bounding Box, C là lớp mà mỗi ô cần dự đoán.

Ta có bounding box gồm 5 thành phần dự đoán $(x, y, w, h, \text{confidence})$ với (x, y) là tọa độ tâm bounding box so với giới bound của ô lưới, (w, h) là chiều rộng, chiều cao được dự đoán so với toàn bộ hình ảnh, và cuối cùng dự đoán confidence đại diện cho IOU (Intersection over Union (tạm dịch: hợp cả 2) giữa vị trí được dự đoán và vị trí cố định cho trước (predicted box, ground truth box)).

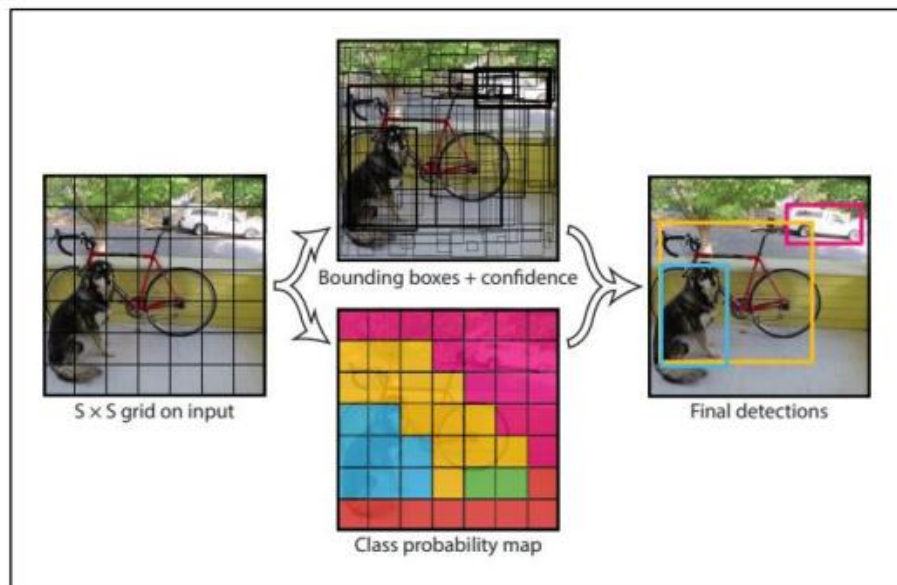
Từng ô lưới dự đoán xác suất lớp (class): $\text{Pr}(\text{Class}_i | \text{Object})$

Confidence score: nếu không có đối tượng trong ô, thì điểm sẽ là 0, còn không thì bằng $\text{Pr}(\text{Object}) * \text{IOU}$

$$\text{Pr}(\text{Class}_i | \text{Object}) * \text{Pr}(\text{Object}) * \text{IOU}_{\text{pred}}^{\text{truth}} = \text{Pr}(\text{Class}_i) * \text{IOU}_{\text{pred}}^{\text{truth}}$$

Hình 2.3: Công thức tính độ chính xác phát hiện ra đối tượng

Theo như công thức trên ta sẽ ra xác suất final detection sẽ dự đoán và bao các đối tượng trong hình ảnh đó như hình minh họa bên dưới:



Hình 2.4: Phát hiện nhận diện đối tượng theo công thức

2.1.2. Ưu và nhược điểm

Ưu điểm:

- Một trong những ưu điểm mà YOLO đem lại đó là chỉ sử dụng thông tin toàn bộ bức ảnh một lần và dự đoán toàn bộ object box chứa các đối tượng, mô hình được xây dựng theo kiểu end-to-end nên được huấn luyện hoàn toàn bằng gradient descen.
- Xử lý khung hình ở tốc độ 45 FPS đến 150 FPS tốt hơn so với thời gian thực.
- Khả năng tổng quát hình ảnh tốt hơn.

Nhược điểm

- Hiệu suất không cao trong việc phát hiện các đối tượng có kích thước nhỏ hoặc đối tượng cần phát hiện nhỏ hơn 1 ô lưới được phân cắt theo mô hình.
- Việc phát hiện một nhóm các đối tượng ở chung một ô lưới được phân cắt theo mô hình cũng trở nên khó khăn.
- Bị giới hạn về tính linh hoạt trong việc phát hiện các vật thể do mô hình chỉ phát hiện được các vật thể dựa trên dữ liệu đã được học từ trước (được phân vùng theo một kích thước cụ thể) nên khi có một vật thể với tỉ lệ kích thước

khác thường dễ dàng dẫn đến việc phát hiện sai vật thể hoặc làm giảm tỷ lệ chính xác.

2.2. Mô hình YOLOv8n

2.2.1. Đặc điểm chính

YOLOv8n là một phiên bản thu nhỏ của mô hình YOLOv8, được thiết kế để cân bằng giữa hiệu suất (speed) và kích thước mô hình (size). Nó rất phù hợp cho các ứng dụng yêu cầu tốc độ xử lý cao hoặc phải hoạt động trên thiết bị có tài nguyên hạn chế như IoT, thiết bị edge, hoặc thiết bị nhúng.

Đặc điểm chính của YOLOv8n:

- Nhỏ gọn:
 - + Kích thước mô hình rất nhỏ, giúp tiết kiệm bộ nhớ (RAM/VRAM) và giảm thời gian tải dữ liệu.
 - + Được thiết kế để chạy mượt mà trên các CPU/GPU có hiệu suất thấp.
- Tốc độ cao:
 - + Tối ưu hóa cho xử lý thời gian thực, có thể đạt tốc độ inference nhanh trên các thiết bị như NVIDIA Jetson Nano, Raspberry Pi hoặc laptop thông thường.
 - + Tốc độ nhanh hơn so với các phiên bản lớn như YOLOv8m, YOLOv8l, đặc biệt trong các bài toán cần phản hồi tức thì.
- Anchor-free detection: Không sử dụng anchor boxes, làm giảm độ phức tạp của mô hình và tăng tốc độ xử lý.
- Đa tác vụ: Hỗ trợ phát hiện đối tượng (object detection), phân loại hình ảnh (image classification) và phân đoạn đối tượng (instance segmentation).
- Khả năng tùy chỉnh: Người dùng có thể điều chỉnh mô hình dễ dàng thông qua các framework phổ biến như PyTorch hoặc tích hợp sẵn của Ultralytics.

2.2.2. Ứng dụng

- Nhận diện biển số xe: Triển khai trên các thiết bị có cấu hình thấp để xử lý nhanh trong thời gian thực.
- Nhận diện khuôn mặt hoặc vật thể: Tích hợp trong các hệ thống camera giám sát hoặc robot.
- Thiết bị IoT và edge: Các ứng dụng như nhà thông minh, giao thông thông minh, hoặc nông nghiệp thông minh.
- Ứng dụng di động: Triển khai trên smartphone hoặc thiết bị di động với hạn chế về tài nguyên.

2.2.3. Ưu điểm và nhược điểm

Ưu điểm:

- Tốc độ cao: Đảm bảo thời gian phản hồi nhanh, phù hợp với các bài toán thời gian thực.
- Tiêu thụ tài nguyên thấp: Dễ dàng triển khai trên các thiết bị với cấu hình khiêm tốn.
- Dễ sử dụng: Tích hợp tốt với các công cụ huấn luyện của Ultralytics.2.2.4. Sử dụng app chấm công có kết nối internet.

Nhược điểm:

- Độ chính xác thấp hơn: So với các phiên bản lớn như YOLOv8l, độ chính xác có thể thấp hơn, đặc biệt với các bài toán phức tạp hoặc dữ liệu khó.
- Giới hạn cho các đối tượng nhỏ: Khả năng nhận diện các đối tượng nhỏ hoặc bị che khuất có thể không bằng các phiên bản lớn hơn.

2.3. Thư viện EasyOCR

EasyOCR là một thư viện Python mạnh mẽ, mã nguồn mở, được thiết kế để thực hiện Nhận dạng ký tự quang học (OCR) một cách dễ dàng và nhanh chóng. Thư viện này đặc biệt hữu ích trong việc trích xuất văn bản từ hình ảnh hoặc tài liệu dạng hình ảnh.

Đặc điểm chính:

- Dễ sử dụng.
- Hỗ trợ đa ngôn ngữ.

- Hiệu suất cao.
- Mã nguồn mở.
- Tích hợp GPU.

Ứng dụng:

- Nhận diện biển số xe: Trích xuất văn bản từ biển số xe trong các dự án như hệ thống kiểm soát giao thông hoặc quản lý bãi đỗ xe.
- Số hóa tài liệu: Chuyển đổi các tài liệu dạng ảnh (PDF, ảnh chụp) thành văn bản có thể chỉnh sửa.
- Nhận diện văn bản từ video: Kết hợp EasyOCR với OpenCV để nhận diện văn bản trong video theo thời gian thực.
- Dịch thuật hình ảnh: Tích hợp với thư viện dịch thuật để trích xuất và dịch văn bản từ ảnh.
- Nhận diện viết tay: Với độ chính xác cao, EasyOCR cũng có thể được sử dụng để nhận diện chữ viết tay.

2.4. Google Colab

Colaboratory hay còn gọi là Google Colab, là một sản phẩm từ Google Research, nó cho phép thực thi các câu lệnh Python trên nền tảng đám mây, đặc biệt phù hợp với những bạn nào làm dự án (project) theo nhóm, hoặc muốn chia sẻ file code Python của mình với người khác.

Colab không yêu cầu cài đặt hay cần cấu hình máy tính mạnh, mọi thứ có thể chạy thông qua trình duyệt, bạn có thể sử dụng tài nguyên máy tính từ CPU tốc độ cao, GPUs hay TPUs đều được cung cấp cho bạn và đặc biệt là hoàn toàn miễn phí nếu bạn chọn option mặc định là CPU. Cá nhân mình rất thích Google Colab vì kể cả máy mình không đủ mạnh vẫn có thể sử dụng nó để viết code Python mà không cần phải cài đặt thêm bất kì phần mềm nào vào máy.

Giao diện của Google Colab rất giống với Jupyter Notebook - một công cụ giúp bạn chạy từng dòng lệnh Python một cách trực quan và kiểm tra kết quả câu lệnh ngay tại chỗ.

Welcome To Colab

File Edit View Insert Runtime Tools Help

Table of contents

- Getting started
- Data science
- Machine learning
- More Resources

Featured examples

+ Section

```
[ ] import numpy as np
import IPython.display as display
from matplotlib import pyplot as plt
import io
import base64

ys = 200 + np.random.randn(100)
x = [x for x in range(len(ys))]

fig = plt.figure(figsize=(4, 3), facecolor='w')
plt.plot(x, ys, '-')
plt.fill_between(x, ys, 195, where=(ys > 195), facecolor='g', alpha=0.6)
plt.title("Sample Visualization", fontsize=10)

data = io.BytesIO()
plt.savefig(data)
image = F"data:image/png;base64,{base64.b64encode(data.getvalue()).decode()}"
alt = "Sample Visualization"
display.display(display.Markdown(F"!!!{alt}!!!({image})"))
plt.close(fig)
```

Sample Visualization

202
201
200
199
198
197
196
195

Resources

You are not subscribed. [Learn more](#)

You currently have zero compute units available. Resources offered free of charge are not guaranteed. Purchase more units [here](#).

At your current usage level, this runtime may last up to 84 hours 40 minutes.

[Manage sessions](#)

Want more memory and disk space?

[Upgrade to Colab Pro](#)

Python 3 Google Compute Engine backend

Showing resources from 9:49 AM to 9:56 AM

System RAM	Disk
1.0 / 12.7 GB	27.1 / 107.7 GB

[Change runtime type](#)

Hình 2.5: Giao diện Google Colab

CHƯƠNG 3: HUẤN LUYỆN MÔ HÌNH

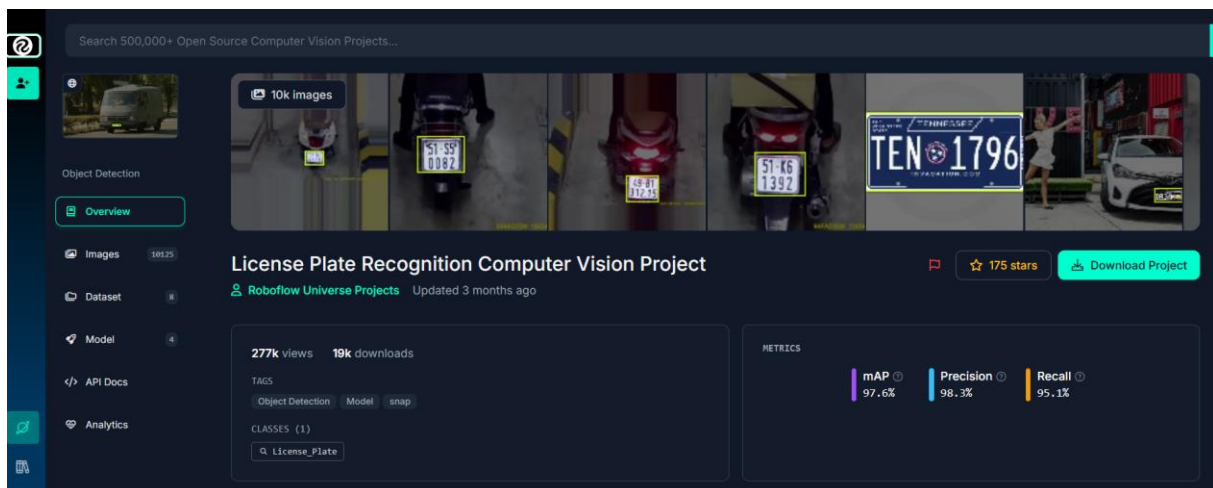
3.1. Thu thập dữ liệu

Để huấn luyện mô hình YOLOv8 trong bài toán nhận diện biển số xe, việc thu thập và chuẩn bị dữ liệu chất lượng cao đóng vai trò quan trọng. Quy trình thu thập dữ liệu được thực hiện qua các bước sau:

3.1.1. Tìm kiếm và lựa chọn dữ liệu

Dữ liệu hình ảnh biển số xe được thu thập từ nhiều nguồn khác nhau như: camera giao thông, video giám sát, hoặc các tập dữ liệu sẵn có trên nền tảng **Roboflow**.

Ưu tiên các hình ảnh đa dạng về góc chụp, điều kiện ánh sáng, và kích thước biển số để đảm bảo tính bao quát cho mô hình.

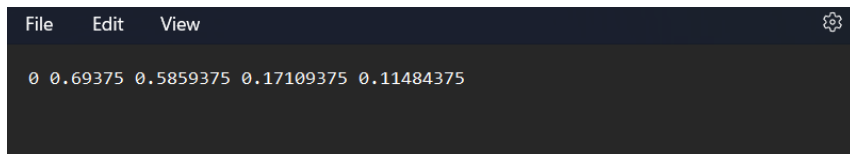


3.1.2. Gán nhãn dữ liệu

Công cụ gán nhãn trên Roboflow được sử dụng để đánh dấu vị trí biển số xe trong từng hình ảnh bằng cách vẽ các hộp giới hạn (bounding boxes).



Nhãn được lưu trữ dưới định dạng phù hợp với YOLO (như file.txt chứa tọa độ hộp giới hạn và nhãn đối tượng).



3.1.3. Tiền xử lí dữ liệu

Chuẩn hóa hình ảnh: Thay đổi kích thước ảnh để đảm bảo tương thích với yêu cầu đầu vào của YOLOv8.

Tăng cường dữ liệu (Data Augmentation): Sử dụng các kỹ thuật như xoay, lật, thay đổi độ sáng và độ tương phản để tăng kích thước và sự đa dạng của bộ dữ liệu.

Lọc dữ liệu: Loại bỏ các hình ảnh không rõ ràng hoặc chứa nhiều nhiễu ảnh hưởng đến chất lượng nhận diện.

3.1.4. Chia bộ dữ liệu

Bộ dữ liệu được chia thành 3 tập riêng biệt để đảm bảo hiệu suất huấn luyện:

- Tập huấn luyện (Training set): Chiếm khoảng 70-80% tổng dữ liệu, dùng để huấn luyện mô hình.

- Tập kiểm tra (Test set): Chiếm khoảng 10-15%, dùng để đánh giá hiệu suất mô hình.
- Tập xác thực (Validation set): Chiếm khoảng 10-15%, dùng để theo dõi hiệu suất trong quá trình huấn luyện.

Name	Date modified	Type	Size
test	12/3/2024 10:54 AM	File folder	
train	12/3/2024 10:55 AM	File folder	
valid	12/3/2024 10:56 AM	File folder	
data	12/3/2024 12:41 AM	Yaml Source File	1 KB
README.dataset	1/14/2023 1:28 AM	Text Document	1 KB
README.roboflow	1/14/2023 1:28 AM	Text Document	2 KB

3.1.5. Xuất và tích hợp dữ liệu

Sau khi xử lý, bộ dữ liệu được xuất dưới định dạng tương thích với YOLOv8 (như định dạng YOLOv5/YOLOv8).

Dữ liệu có thể được tích hợp trực tiếp vào Google Colab hoặc các môi trường huấn luyện khác thông qua API của Roboflow.

3.2. Huấn luyện mô hình

3.2.1. Cài đặt môi trường

- Sử dụng Google Colab với GPU được kích hoạt để tăng tốc huấn luyện.
- Cài đặt các thư viện cần thiết, bao gồm ultralytics (hỗ trợ YOLOv8), opencv-python, và matplotlib. Sau khi cài đặt thành công, chương trình sẽ thông báo:

```

Requirement already satisfied: roboflow in /usr/local/lib/python3.10/dist-packages (1.1.49)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from roboflow) (2024.8.30)
Requirement already satisfied: idna==3.7 in /usr/local/lib/python3.10/dist-packages (from roboflow) (3.7)
Requirement already satisfied: cycler in /usr/local/lib/python3.10/dist-packages (from roboflow) (0.12.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.4.7)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from roboflow) (3.8.0)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.26.4)
Requirement already satisfied: opencv-python-headless==4.10.0.84 in /usr/local/lib/python3.10/dist-packages (from roboflow) (4.10.0.84)
Requirement already satisfied: Pillow>=7.1.2 in /usr/local/lib/python3.10/dist-packages (from roboflow) (11.0.0)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from roboflow) (2.8.2)
Requirement already satisfied: python-dotenv in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.0.1)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from roboflow) (2.32.3)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.16.0)
Requirement already satisfied: urllib3>=1.26.6 in /usr/local/lib/python3.10/dist-packages (from roboflow) (2.2.3)
Requirement already satisfied: tqdm>=4.41.0 in /usr/local/lib/python3.10/dist-packages (from roboflow) (4.66.6)
Requirement already satisfied: PyYAML>=5.3.1 in /usr/local/lib/python3.10/dist-packages (from roboflow) (6.0.2)
Requirement already satisfied: requests-toolbelt in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.0.0)
Requirement already satisfied: filetype in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.2.0)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->roboflow) (1.3.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->roboflow) (4.55.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib->roboflow) (24.2)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib->roboflow) (3.2.0)
Requirement already satisfied: charset-normalizer<4.0 in /usr/local/lib/python3.10/dist-packages (from requests->roboflow) (3.4.0)

```

3.2.2. Tiến hành huấn luyện

- Tải bộ dữ liệu đã được xử lý từ roboflow

```
[ ] pip install roboflow

from roboflow import RoboFlow
rf = RoboFlow(api_key="xxBKci0frUwh0oWIPy40")
project = rf.workspace("roboflow-universe-projects").project("license-plate-recognition-rxg4e")
version = project.version(4)
dataset = version.download("yolov8")

Requirement already satisfied: roboflow in /usr/local/lib/python3.10/dist-packages (1.1.49)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from roboflow) (2024.8.30)
Requirement already satisfied: idna==3.7 in /usr/local/lib/python3.10/dist-packages (from roboflow) (3.7)
Requirement already satisfied: cycler in /usr/local/lib/python3.10/dist-packages (from roboflow) (0.12.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.4.7)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (from roboflow) (3.8.0)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.10/dist-packages (from roboflow) (1.26.4)
Requirement already satisfied: opencv-python-headless==4.10.0.84 in /usr/local/lib/python3.10/dist-packages (from roboflow) (4.10.0.84)
```



- Bắt đầu huấn luyện mô hình. Ở đây nhóm em huấn luyện mô hình với số lần duyệt qua mô hình là epochs = 20.

```
!-Detection-using-YOLO-V8/ultralytics/yolo/v8/detect/train.py model=yolov8n.pt data=/content/Licence-Plate-Detection-using-YOLO-V8/L

0      -1  1      464  ultralytics.nn.modules.Conv      [3, 16, 3, 2]
1      -1  1      4672 ultralytics.nn.modules.Conv      [16, 32, 3, 2]
2      -1  1      7360 ultralytics.nn.modules.C2f      [32, 32, 1, True]
3      -1  1     18560 ultralytics.nn.modules.Conv      [32, 64, 3, 2]
4      -1  2     49664 ultralytics.nn.modules.C2f      [64, 64, 2, True]
5      -1  1     73984 ultralytics.nn.modules.Conv      [64, 128, 3, 2]
6      -1  2     197632 ultralytics.nn.modules.C2f      [128, 128, 2, True]
7      -1  1     295424 ultralytics.nn.modules.Conv      [128, 256, 3, 2]
8      -1  1     460288 ultralytics.nn.modules.C2f      [256, 256, 1, True]
9      -1  1     164608 ultralytics.nn.modules.SPPF      [256, 256, 5]
10     -1  1           0 torch.nn.modules.upsampling.Upsample [None, 2, 'nearest']
11     [-1, 6] 1           0 ultralytics.nn.modules.Concat      [1]
12     -1  1     148224 ultralytics.nn.modules.C2f      [384, 128, 1]
13     -1  1           0 torch.nn.modules.upsampling.Upsample [None, 2, 'nearest']
14     [-1, 4] 1           0 ultralytics.nn.modules.Concat      [1]
15     -1  1     37248  ultralytics.nn.modules.C2f      [192, 64, 1]
16     -1  1     36992  ultralytics.nn.modules.Conv      [64, 64, 3, 2]
17     [-1, 12] 1           0 ultralytics.nn.modules.Concat      [1]
18     -1  1     123648 ultralytics.nn.modules.C2f      [192, 128, 1]
19     -1  1     147712 ultralytics.nn.modules.Conv      [128, 128, 3, 2]
20     [-1, 9] 1           0 ultralytics.nn.modules.Concat      [1]
21     -1  1     493056 ultralytics.nn.modules.C2f      [384, 256, 1]
22     [-1, 10, 31] 1     751507 ultralytics.nn.modules.Detect      [1, [64, 128, 256]]

Transferred 319/355 items from pretrained weights
optimizer: SGD(lr=0.01) with parameter groups 57 weight(decay=0.0), 64 weight(decay=0.0005), 63 bias
train: Scanning /content/Licence-Plate-Detection-using-YOLO-V8/Licence-Plate-Recognition-4/train/labels... 21173 images, 28 backgrounds,
Signal received. 15 <frame at 0x7f4bf7994780, file '/usr/lib/python3.10/concurrent/futures/thread.py', line 23, code _python_exit>
train: New cache created: /content/Licence-Plate-Detection-using-YOLO-V8/Licence-Plate-Recognition-4/train/labels.cache
/usr/local/lib/python3.10/dist-packages/albumentations/__init__.py:24: UserWarning: A new version of Albumentations is available: 1.4.21
check_for_updates()
/usr/local/lib/python3.10/dist-packages/albumentations/core/composition.py:205: UserWarning: Got processor for bboxes, but no transform to
self.set_keys()
albumentations: Blur(p=0.01, blur_limit=(3, 7)), MedianBlur(p=0.01, blur_limit=(3, 7)), ToGray(p=0.01, num_output_channels=3, method='wei
val: Scanning /content/Licence-Plate-Detection-using-YOLO-V8/Licence-Plate-Recognition-4/valid/labels... 2046 images, 3 backgrounds, 0 co
Signal received. 15 <frame at 0x7f4d1181f2e0, file '/usr/lib/python3.10/threading.py', line 810, code <listcomp>>
val: New cache created: /content/Licence-Plate-Detection-using-YOLO-V8/Licence-Plate-Recognition-4/valid/labels.cache
Image sizes 640 train, 640 val
Using 0 dataloader workers
Logging results to runs/detect/train3
Starting training for 20 epochs...
```

Sau khi mô hình được huấn luyện thành công sẽ tạo ra hai file là file best.pt và last.pt. Chúng ta sẽ dùng file best.pt để nhận diện biển số xe.

Name	Date modified	Type	Size
 best.pt	12/8/2024 4:49 PM	PT File	5,419 KB
 last.pt	12/8/2024 4:49 PM	PT File	5,419 KB

Sau quá trình huấn luyện chúng ta có mô hình nhận diện biển số xe. Làm tương tự để tạo mô hình nhận diện xe máy, ô tô nhằm cho chương trình nhận diện biển số xe chính xác hơn.

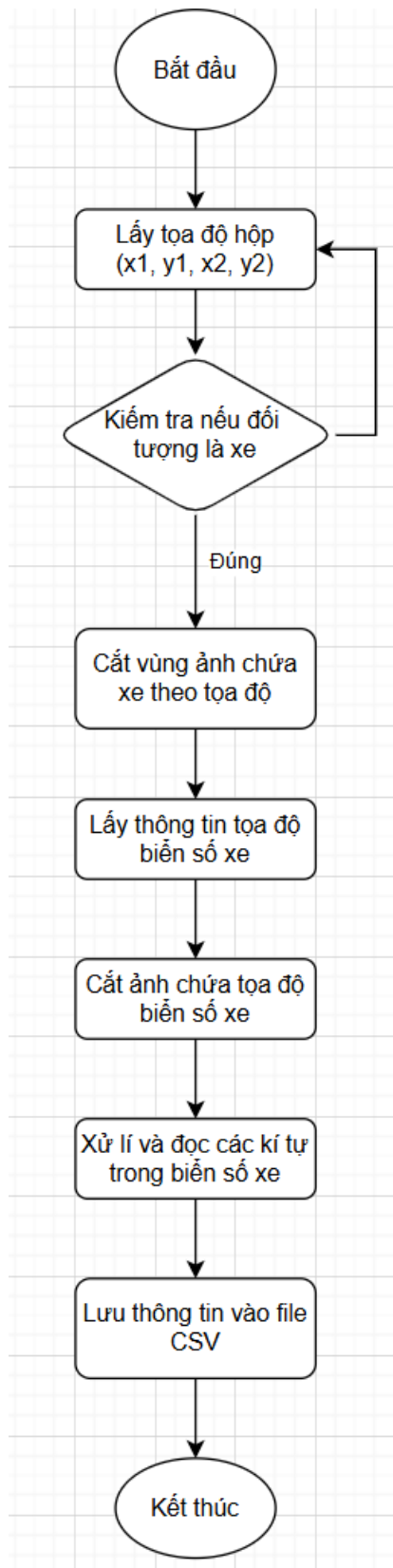
CHƯƠNG 4: QUI TRÌNH NHẬN DIỆN BIỂN SỐ XE

Để có thể tiến hành nhận diện biển số, ta có thể chia giai đoạn nhận diện biển số thành hai giai đoạn chính, đó là:

- Giai đoạn 1: Nhận diện vùng chứa biển số xe.
- Giai đoạn 2: Nhận diện các ký tự có trong biển số.

4.1. Nhận diện vùng chứa biển số xe

Sơ đồ giải thuật:

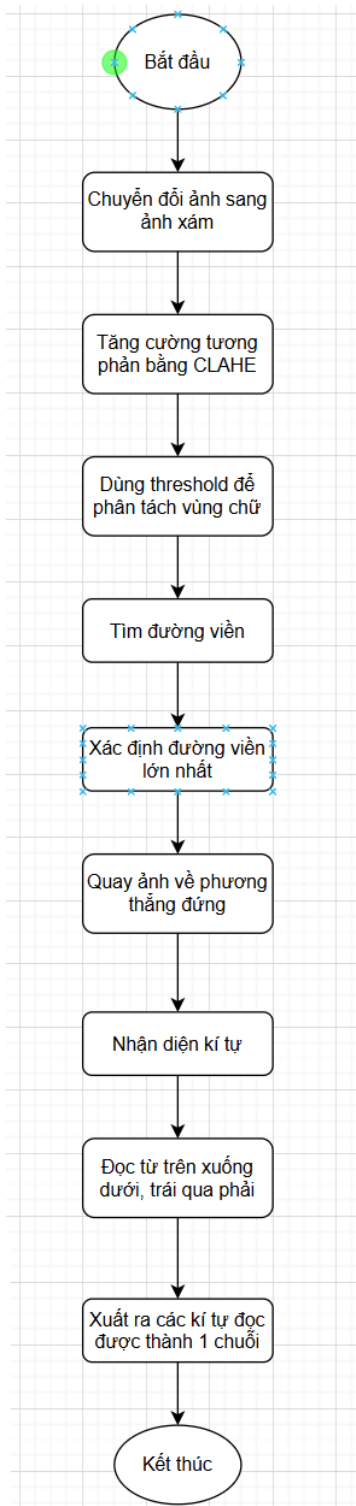


Hình 4.1: Sơ đồ thuật toán nhận diện biển số xe

- Quá trình cắt biên số xe gồm các bước như sau: Đầu tiên, từ file đã huấn luyện, chương trình sẽ nhận diện các phương tiện có trong ảnh như xe máy, ô tô, xe tải. Sau khi đã nhận diện được phương tiện, chương trình tiến hành lấy tọa độ $(x1, y1, x2, y2)$ ứng với phương tiện đó và tiến hành cắt ảnh chứa các tọa độ đó. Sau khi đã có các ảnh phương tiện, chương trình tiếp tục nhận diện biển số xe có trong ảnh chứa phương tiện để lấy các tọa độ tương ứng và cắt hình ảnh chứa tọa độ ứng với tọa độ đọc được. Tiếp đến chương trình tiến hành đọc các kí tự có trong ảnh chứa biển số xe sau đó xuất ra chuỗi và lưu vào file.csv.

4.2. Quá trình đọc biển số xe

- Sơ đồ giải thuật:



Hình 4.2: Sơ đồ thuật toán đọc các kí tự biển số xe

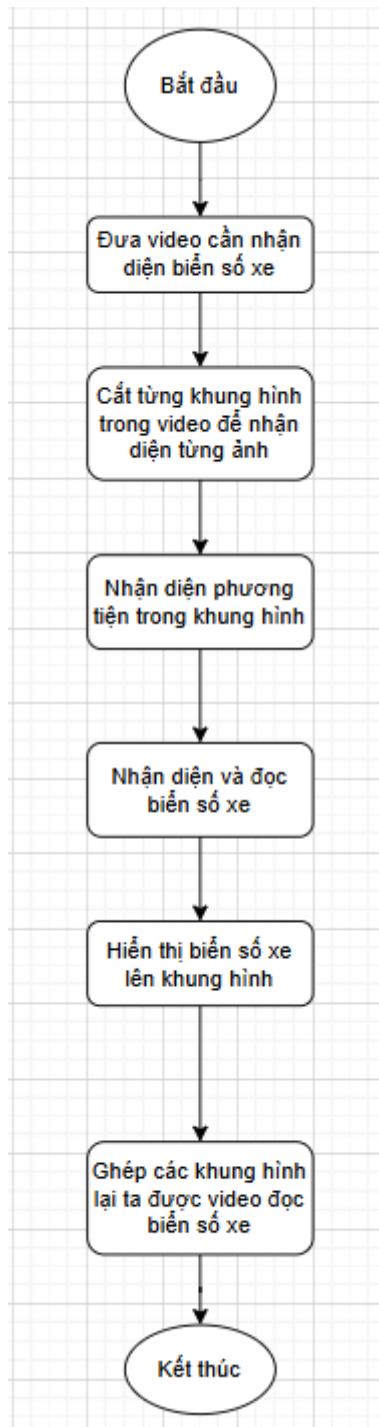
- Khi chương trình đã cắt được ảnh chứa biển số xe, đầu tiên chương trình sẽ chuyển ảnh biển số qua ảnh xám, sau đó dùng phương pháp CLAHE để tăng cường tương phản. Dùng phương pháp threshold để các ký tự hiển thị rõ nét trắng đen hơn nhằm nhận diện chính xác hơn. Tiếp tới chương trình sẽ kiểm tra xem biển số có nằm đúng theo phương thẳng đứng hay chưa, nếu rồi thì giữ nguyên, chưa sẽ tiến hành quay biển số về đúng theo phương thẳng đứng. Trước tiên, chương trình sẽ xác định đường viền bao xung quanh biển số và lấy đường bao nào có diện tích lớn nhất(chứa biển số), sau đó dựng trục chính giữa ảnh và bắt đầu kiểm tra xem ảnh có đang bị lệch hay không để quay về đúng trục. Sau bước kiểm tra vị trí ảnh, chương trình bắt đầu nhận diện ký tự có trong ảnh. Chương trình sẽ đọc ký tự theo thứ tự từ trên xuống dưới, từ trái qua phải và in ra thành chuỗi ký tự và đó là biển số xe cần nhận dạng. Trong chương trình này, nhóm em định dạng biển số xe gồm 8 hoặc 9 ký tự.



Hình 4.3: Hình ảnh sau khi threshold và quay biển số về phương thẳng đứng

4.3. Chương trình nhận diện biển số xe từ video

Sơ đồ giải thuật:



Hình 4.4: Sơ đồ thuật toán đọc biển số xe từ video

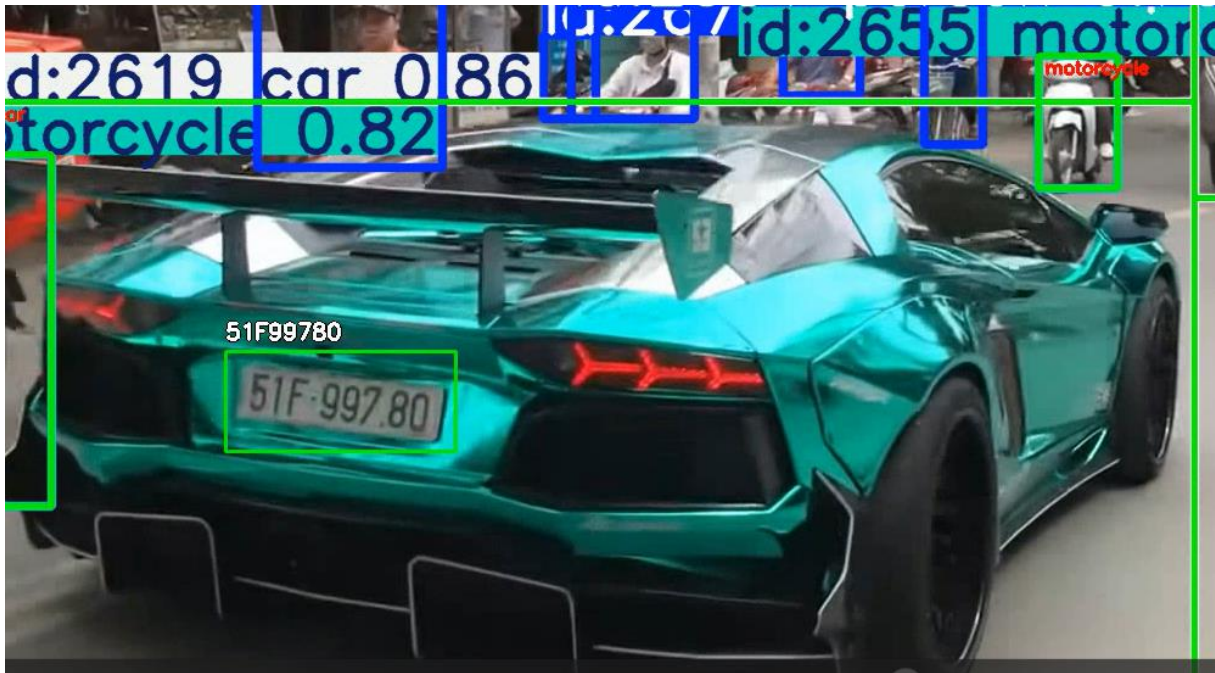
- Để hiển thị biển số xe đọc được lên video, đầu tiên chương trình sẽ cắt video thành các khung hình liên tiếp nhau và tiến hành đọc biển số từng khung

hình. Sau khi đã hiển thị thành công biển số của từng xe lên khung hình, chương trình sẽ ghép các khung hình lại thành video và xuất.

CHƯƠNG 5: KẾT QUẢ VÀ ĐÁNH GIÁ

5.1. Kết quả đạt được

Hệ thống đã nhận diện thành công biển số xe tuy nhiên vẫn có lúc hệ thống nhận diện sai biển vì điều kiện chất lượng video bị mờ hoặc do biển số bị lệch so với khung hình quá. Video hiển thị biển số xe xuất ra trong thư mục output.



Hình 5.1: Chương trình hiển thị thành công biển số xe



Hình 5.2: Đọc thành công biển số xe taxi

5.2. Tổng kết

Ưu điểm:

- Hệ thống đã có thể nhận diện thành công biển số xe và hiển thị trên video.
- Tuy một số biển số bị nghiêng hoặc lệch nhưng hệ thống vẫn nhận diện được.

Nhược điểm:

- Hệ thống bị hạn chế trong điều kiện ánh sáng kém.
- Nếu biển số bị che khuất hoặc hư hỏng thì hệ thống không nhận diện được.
- Hệ thống đôi khi bị nhầm lẫn giữa kí tự “0” và “O” hoặc là kí tự “I” và “1”.

5.3. Kết quả thống kê

Sau khi hoàn thành sản phẩm, nhóm em có chạy thử và lập ra được bảng thống kê số lần thực hiện chính xác để kiểm tra tính ổn định của hệ thống như sau:

Kết quả được thể hiện trong bảng 5.2.

Bảng 5.2: Kết quả thống kê số lần thực hiện chính xác của hệ thống.

STT	SỐ LẦN THỬ NGHIỆM	SỐ LẦN THỰC HIỆN ĐÚNG	XÁC SUẤT (%)
1	102	88	86%
2	76	70	92%
3	168	131	78%

CHƯƠNG 6: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1. Kết luận

Nghiên cứu này đã xây dựng một hệ thống nhận diện biển số xe ứng dụng mô hình YOLOv8 kết hợp với OCR, mang lại những kết quả khả quan trong việc phát hiện và nhận dạng ký tự trên biển số. Hệ thống đạt được những thành tựu nổi bật như:

- Hiệu quả trong nhận diện đối tượng: Với sự hỗ trợ của YOLOv8n, hệ thống có khả năng phát hiện biển số xe nhanh chóng và chính xác, phù hợp cho các ứng dụng thời gian thực.

- Tích hợp OCR trích xuất ký tự hiệu quả: Công nghệ OCR đã hỗ trợ tốt trong việc nhận diện nội dung ký tự trên biển số, đảm bảo tính toàn diện cho hệ thống.

- Ứng dụng thực tế rộng rãi: Hệ thống có thể áp dụng vào nhiều lĩnh vực như quản lý giao thông, bãi đỗ xe thông minh, và giám sát an ninh.

Tuy nhiên, vẫn còn tồn tại một số hạn chế, chẳng hạn như giảm hiệu suất trong điều kiện ánh sáng yếu, biển số bị che khuất, hoặc sự nhầm lẫn ký tự trong bước OCR. Những thách thức này sẽ là cơ sở để cải tiến hệ thống trong tương lai.

6.2. Hướng phát triển

Hệ thống nhận diện biển số xe có tiềm năng phát triển và mở rộng để đáp ứng các yêu cầu thực tế cao hơn. Trong tương lai, hệ thống có thể được nâng cấp theo các hướng sau:

- Cải thiện khả năng nhận diện trong điều kiện khó khăn.
- Tối ưu hóa mô hình OCR.
- Triển khai thực tế trên thiết bị nhúng.
- Hỗ trợ nhiều loại biển số.
- Phát triển giao diện người dùng (UI/UX).

TÀI LIỆU THAM KHẢO

- [1] <https://dlu.edu.vn/wp-content/uploads/2023/08/TongKet.pdf>
- [2] <https://docs.ultralytics.com/vi/models/yolov8/>
- [3] [https://aws.amazon.com/vi/what-is/ocr/#:~:text=Nh%E1%BA%ADn%20d%E1%BA%A1ng%20k%C3%BD%20t%E1%BB%B1%20quang%20h%E1%BB%8Dc%20\(OCR\)%20l%C3%A0%20q%C3%A1%20tr%C3%ACnh,d%C6%B0%E1%BB%9Bi%20d%E1%BA%A1ng%20t%E1%BB%87p%20h%C3%ACnh%20%E1%BA%A3nh](https://aws.amazon.com/vi/what-is/ocr/#:~:text=Nh%E1%BA%ADn%20d%E1%BA%A1ng%20k%C3%BD%20t%E1%BB%B1%20quang%20h%E1%BB%8Dc%20(OCR)%20l%C3%A0%20q%C3%A1%20tr%C3%ACnh,d%C6%B0%E1%BB%9Bi%20d%E1%BA%A1ng%20t%E1%BB%87p%20h%C3%ACnh%20%E1%BA%A3nh)