TRƯỜNG ĐẠI HỌC HÀNG HẢI VIỆT NAM
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO BÀI TẬP LỚN
NHẬN DẠNG VÀ XỬ LÝ ẢNH

ĐỀ TÀI

XÂY DỰNG CHƯƠNG TRÌNH
NHẬN DẠNG BIỂN BÁO GIAO THÔNG

GVHD: ***TS. Nguyễn Hữu Tuân***
Sinh viên thực hiện: ***Tạ Thị Phương Thảo – 89773 – CNT61ĐH***

Hải Phòng, tháng 4/2023

MỤC LỤC

MỞ ĐẦU 6

1. Tính cấp thiết của vấn đề nghiên cứu 6

2. Tình hình nghiên cứu thuộc lĩnh vực đề tài 7

3. Mục tiêu, đối tượng, phạm vi nghiên cứu 8

4. Phương pháp nghiên cứu, kết cấu của công trình nghiên cứu 9

5. Kết quả đạt được của đề tài 9

CHƯƠNG 1 : BÀI TOÁN NHẬN DẠNG BIỂN BÁO GIAO THÔNG 10

1.1. Giới thiệu bài toán nhận dạng biển báo giao thông 10

1.2. Một số yếu tố tác động tới hiệu năng của một hệ thống nhận dạng biển báo giao thông 10

CHƯƠNG 2 : HỆ THỐNG NHẬN DẠNG BIỂN BÁO GIAO THÔNG 12

2.1. Kiến trúc của hệ thống 12

2.2. Mạng YOLO3 13

2.3. YOLO3 phát hiện đối tượng như thế nào? 14

2.4. Sử dụng mạng YOLO3 nhận dạng biển báo giao thông 20

CHƯƠNG 3 : XÂY DỰNG HỆ THỐNG NHẬN DẠNG BIỂN BÁO GIAO THÔNG 21

3.1. Tìm hiểu Python và các thư viện lập trình 21

3.2. Train mô hình nhận dạng biển báo giao thông 22

3.3. Chương trình nhận dạng biển báo giao thông 29

KẾT LUẬN 34

TÀI LIỆU THAM KHẢO 35

DANH SÁCH HÌNH ẢNH

Hình 2.1: Kiến trúc hệ thống nhận dạng biển báo giao thông *12*

Hình 2.2: Mô hình mạng YOLOV3 (Darknet53). 14

Hình 2.3: Cách thức mạng YOLOV3 thực hiện phát hiện đối tượng. 16

Hình 2.4: Kết quả bước đầu do YOLOV3 phát hiện đối tượng..... 17

Hình 2.5: Minh hoạ YOLO khi dự đoán đối tượng trong ảnh ở các tỉ lệ khác nhau. 17

Hình 2.6: Hàm IoU của YOLO..... 19

Hình 2.7: Minh họa kết quả của việc áp dụng kỹ thuật NMS. 20

Hình 3.2.1: Bộ dữ liệu...... *22*

Hình 3.2.2:Bộ dữ liệu..... *23*

Hình 3.2.3: Bộ dữ liệu...... *23*

Hình 3.2.4:Kết nối với google driver..... *23*

Hình 3.2.5: Tạo thư mục, tải xuống và giải nén giữ liệu...... *24*

Hình 3.2.6:Train *25*

Hình 3.2.7: Test..... *26*

Hình 3.2.8:Test..... *28*

Hình 3.2.9: Test..... *29*

Hình 3.3.1: Tải mô hình Yolo *30*

Hình 3.3.2: Hàm bắt đầu webcam. *30*

Hình 3.3.3: Hàm truy cập video *30*

Hình 3.3.4: Code tạo giao diện. *31*

Hình 3.3.5: Code nhận dạng đối tượng *31*

Hình 3.3.6: Code hiển thị kết quả..... *32*

Hình 3.3.7: Giao diện chính *33*

Hình 3.3.8: Form chọn video *33*

Hình 3.3.9: Video kết quả *34*

Hình 3.3.10: Ảnh kết quả khi nhận diện bằng webcam *34*

DANH SÁCH THUẬT NGỮ, CHỮ VIẾT TẮT

You Only Look Once	**YOLO**
Traffic-Sign Recognition	**TSR**
Convolutional Neural Network	**CNN**
Deep Neural Networks	**DNNs**
Single Shot MultiBox Detector	**SSD**

MỞ ĐẦU

1. Tính cấp thiết của vấn đề nghiên cứu

Phát hiện và nhận diện biển báo giao thông là một bài toán cần thiết và có tính ứng dụng cao. Các hệ thống nhận dạng biển báo giao thông là tiền đề quan trọng để xây dựng lên các giải pháp hỗ trợ người lái xe trên đường và là một thành phần không thể thiếu được trong các xe tự lái. Từ việc nhận dạng các biển báo giao thông sẽ giúp cho hệ thống định vị vị trí và hướng đi của phương tiện giao thông, cung cấp thông tin định hướng, giảm thiểu tai nạn giao thông, tăng tính an toàn và thuận tiện cho người lái xe.

Ứng dụng của việc nhận dạng biển báo giao thông trong thực tế:

- Hướng dẫn an toàn giao thông: Biển báo giao thông đóng vai trò cung cấp thông tin quan trọng về đường đi, tốc độ, các nguy hiểm tiềm tàng và các quy tắc giao thông. Bằng cách nhận dạng và hiểu biển báo, chúng ta có thể tuân thủ luật lệ giao thông và điều chỉnh hành vi lái xe hoặc điều hành an toàn khi tham gia giao thông.
- Giảm nguy cơ tai nạn: Nhận dạng chính xác biển báo giao thông giúp giảm nguy cơ xảy ra tai nạn giao thông. Khi chúng ta nhận ra các biển báo cảnh báo, biển báo nguy hiểm hoặc biển báo hạn chế, chúng ta có thể tăng cường cảnh giác, giảm tốc độ và thực hiện các biện pháp an toàn phù hợp.
- Hỗ trợ điều hướng và định vị: Biển báo giao thông cung cấp thông tin quan trọng về hướng đi và địa điểm. Khi chúng ta nhận dạng và hiểu biển báo, chúng ta có thể dễ dàng điều hướng đến đích mong muốn, đảm bảo không bị lạc đường và tiết kiệm thời gian di chuyển.
- Tạo trật tự giao thông: Nhận dạng biển báo giao thông giúp tạo ra một môi trường giao thông có trật tự. Việc tuân thủ biển báo giúp đảm bảo sự phân luồng giao thông, giảm ùn tắc, tăng khả năng dự đoán và đảm bảo an toàn cho mọi người tham gia giao thông.
- Hỗ trợ học lái xe: Việc nhận dạng và hiểu biển báo giao thông là một phần quan trọng trong quá trình học lái xe. Nắm vững kiến thức về biển báo giúp

người học lái xây dựng nhận thức giao thông và tuân thủ quy tắc, làm chủ các kỹ năng lái xe an toàn.

Các giải pháp cho bài toán phát hiện và nhận diện biển báo giao thông hiện nay chia làm 2 hướng chính:

- Hướng tiếp cận dựa trên phương pháp truyền thống (Traditional Methods) như xử lý ảnh, phân đoạn, rút trích đặc trưng và phân loại dựa trên các thuật toán truyền thống. Tuy nhiên, việc sử dụng các kỹ thuật xử lý ảnh đã phát triển từ rất lâu và bộc lộ nhược điểm của chúng là độ chính xác thấp, không tận dụng được hết các năng lực phần cứng và nguồn dữ liệu hình ảnh khổng lồ có thể thu thập được ngày nay.
- Hướng dựa trên các kỹ thuật học sâu (Deep Learning) sử dụng các mô hình mạng nơ ron phức tạp với số các lớp có quy mô lớn. Các mô hình mạng học sâu hiện nay đang là hướng đi chính nhận được nhiều sự quan tâm của các nhà nghiên cứu và các công ty công nghệ hàng đầu trên thế giới.

Do đó, đối với đề tài này, em quyết định thực hiện khảo sát các mô hình mạng học sâu có thể áp dụng cho bài toán nhận dạng biển báo giao thông, lựa chọn mô hình phù hợp nhất và tiến hành cài đặt, thử nghiệm hệ thống, từ đó rút ra kết luận và đánh giá.

2. Tình hình nghiên cứu thuộc lĩnh vực đề tài

Nhận dạng biển báo giao thông (Traffic-Sign Recognition) là một bài toán thị giác máy tính được nghiên cứu từ lâu, tuy nhiên cho tới nay vẫn chưa có giải pháp nào hoàn hảo. Ban đầu, Các kỹ thuật chủ yếu dựa trên các phương pháp xử lý ảnh như chuyển màu sắc chuyển đổi sang không gian màu HSV, sử dụng các bộ lọc và phép làm mờ như Gaussian Blur hoặc Median Filter để giảm nhiễu và làm mờ ảnh. Một kỹ thuật sử dụng các đặc trưng gradient đã được đề xuất bởi Arrospide và các cộng sự. Một phiên bản đầy đủ của các đặc trưng HOG cũng được áp dụng cho bài toán nhận dạng biển báo giao thông. Phương pháp này sử dụng các đặc trưng Haar-like cũng đã được áp dụng để giải quyết bài toán phát hiện biển báo giao thông và các thuật toán phân đoạn ảnh như Watershed, GrabCut

hoặc các mô hình học sâu như Fully Convolutional Networks (FCN) để phân đoạn và tách biển báo từ hình ảnh nền. Gần đây, phương pháp học máy sử dụng các Support Vector Machines (SVMs), Decision Trees, rừng ngẫu nhiên (Random Forest) hoặc K-nearest neighbors (KNN) cũng đã được nghiên cứu để phân loại và nhận dạng biển báo. Tuy nhiên, điểm chung của các cách tiếp cận trên là độ chính xác bị hạn chế, đặc biệt khi áp dụng trong các môi trường có các yếu tố ngoại cảnh thay đổi. Tuy nhiên các phương pháp này khá nhanh, có thể chạy theo thời gian thực và chi phí thấp, thường tốn ít tài nguyên tính toán.

Các mô hình mạng nơ ron học sâu (Deep neural networks - DNNs) ra đời vào khoảng những năm 1960 và được đặt tên là Convolutional Neural Networks (CNNs). Tuy nhiên phải tới những năm 2010 thì chúng mới được phục sinh khi giành được thứ hạng cao trong các cuộc thi về nhận dạng hình ảnh lớn trên thế giới. Điều này là do trước năm 2010, chưa có các cơ sở dữ liệu ảnh lớn và các hệ thống phần cứng song song đủ mạnh để có thể huấn luyện các mạng CNNs phức tạp. Kết quả là các mạng CNNs đã được áp dụng cho nhiều bài toán phát hiện đối tượng và nhận dạng trong lĩnh vực thị giác máy tính và nhận dạng biển báo giao thông là một trong những dạng bài toán đó.

3. Mục tiêu, đối tượng, phạm vi nghiên cứu

Mục tiêu chính của đề tài là nghiên cứu các mạng nơ ron học sâu có thể áp dụng vào bài toán nhận dạng biển báo giao thông phổ biến (gồm biển báo rẽ phải, biển báo rẽ trái, giao với đường ưu tiên, giao với đường sắt, dừng lại, đoạn đường nguy hiểm, đi chậm, cấm rẽ, công trường đang thi công,…).

Đối tượng đề tài nghiên cứu là bài toán nhận dạng biển báo giao thông, các mô hình mạng học sâu nhân chập, các thành phần chính của một mô hình mạng nhân chập và các một số kỹ thuật xử lý ảnh.

Phạm vi đề tài là bài toán nhận dạng biển báo giao thông, kiến trúc mạng nơ ron nhân chập, tập dữ liệu được sử dụng để huấn luyện và kiểm tra mô hình nhận dạng vật thể COCO, mô hình nghiên cứu để giải quyết bài toán nhận dạng

vật thể là YOLOv3, ứng dụng chương trình nhận dạng biển báo giao thông bằng ngôn ngữ Python.

4. Phương pháp nghiên cứu, kết cấu của công trình nghiên cứu

Phương pháp nghiên cứu của đề tài gồm có:

- Tìm hiểu ngôn ngữ lập trình Python và các thư viện hỗ trợ như OpenCV, Tkinter, Numpy, PIL
- Huấn luyện mô hình YOLOv3 dựa trên tập dữ liệu tự xây dựng
- Cài đặt và huấn luyện các mô hình nhận dạng biển báo giao thông.
- Kiểm tra và đánh giá các mô hình nhận dạng biển báo giao thông
- Cài đặt chương trình, thử nghiệm với các dữ liệu hình ảnh tự thu thập
- Phân tích kết quả và rút ra kết luận

5. Kết quả đạt được của đề tài

Kết quả đạt được là một chương trình nhận dạng biển báo giao thông, có thể hoạt động trên các ảnh hoặc webcam với độ chính xác cao và thời gian xử lý nhanh. Chương trình cũng có thể mở rộng để nhận dạng các biển báo giao thông khác. Chương trình có thể được ứng dụng trong các lĩnh vực liên quan trong việc đảm bảo an toàn giao thông, hỗ trợ điều hướng và tạo ra một môi trường giao thông có trật tự,...

CHƯƠNG 1:
BÀI TOÁN NHẬN DẠNG BIỂN BÁO GIAO THÔNG

1.1. Giới thiệu bài toán nhận dạng biển báo giao thông

Bài toán nhận dạng biển báo giao thông từ dữ liệu hình ảnh input và dữ liệu hình ảnh thu được qua các camera thuộc loại bài toán phát hiện đối tượng (object detection) từ dữ liệu input là hình ảnh, kết quả output của bài toàn là danh sách các đối tượng và vị trí của chúng trong ảnh input.

Như vậy để làm được bài toán cần xây dựng bộ dữ liệu biển báo giao thông và sử dụng mô hình YOLOv3 để huấn luyện và kiểm tra các mô hình nhận dạng biển báo giao thông. Bộ dữ liệu bao gồm các ảnh và gán nhãn theo định dạng phù hợp với YOLOv3. Đồng thời tạo giao diện và hiển thị dữ liệu chương trình trên PyCharm.

Việc nhận dạng biển báo giao thông chính xác là tiền đề quan trọng cho việc xây dựng lên các hệ thống hỗ trợ người lái xe trên đường tham gia giao thông và giảm thiểu tai nạn giao thông, cảnh báo tài xế về các biển báo giao thông trên đường giúp tài xế nắm rõ các quy định và hạn chế tốc độ để bản an toàn, giúp tăng hiệu quả và độ chính xác cho hệ thống lái xe thông minh, xe tự hành.

1.2. Một số yếu tố tác động tới hiệu năng của một hệ thống nhận dạng biển báo giao thông

Tuy đây chỉ là một bài toán con của bài toán phát hiện đối tượng nhưng bài toán phát hiện các nhận dạng các biển báo giao thông không phải là một bài toán dễ do các yếu tố sau đây:

+ Ánh sáng ngoài trời: Ánh sáng mạnh và trực tiếp từ mặt trời có thể làm mất thông tin và gây mờ đi chi tiết trên biển báo giao thông, điều này ảnh hưởng lớn đến độ chính xác của hệ thống.

+ Độ nghiêng và hướng: Độ nghiêng và hướng của biển báo có thể làm thay đổi hình dạng và góc nhìn của biển báo, đặc biệt khi các mô hình huấn luyện chỉ được học trên các biển báo ở góc nhìn cụ thể.

+ Màu sắc: Điều kiện ánh sáng và độ tương phản có thể làm thay đổi màu sắc của biển báo, gây khó khăn trong việc nhận diện chính xác.

+ Kích thước và tỷ lệ: Biển báo quá nhỏ hoặc quá lớn so với kích thước mà mô hình đã được huấn luyện có thể gây khó khăn trong việc nhận dạng.

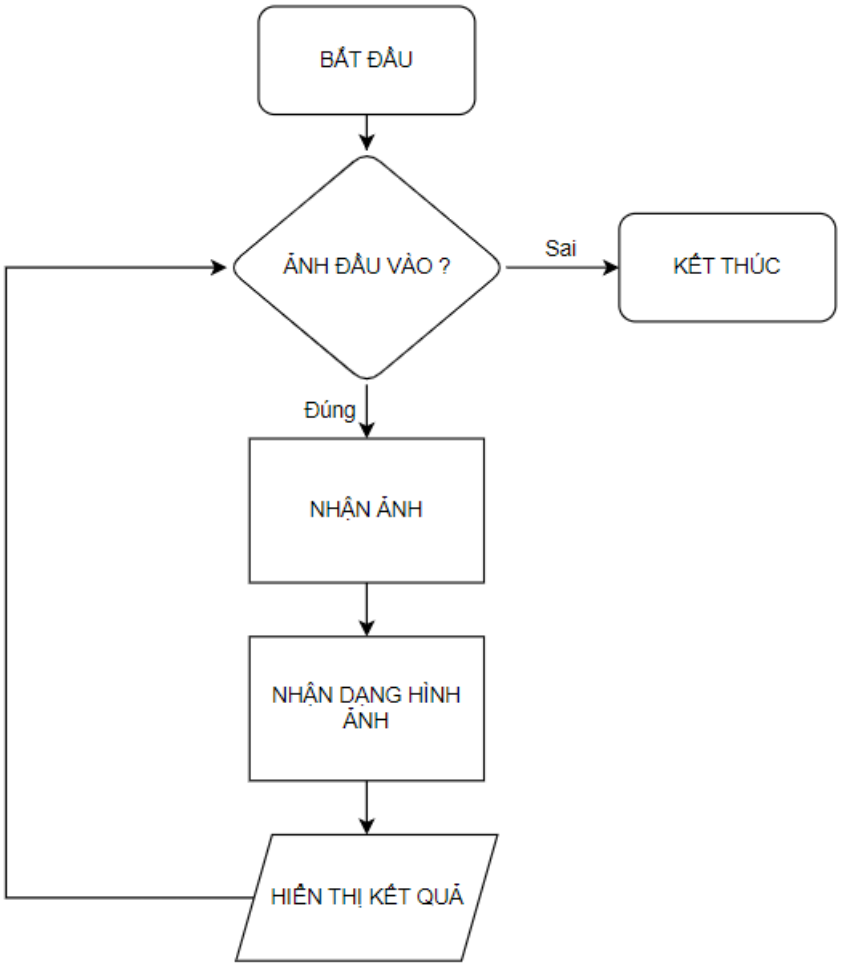
+ Nhiễu và nền: Sự hiện diện của nhiễu, bóng đổ, hoặc các vật thể khác trong khung hình có thể làm mất thông tin hoặc gây hiểu lầm trong quá trình nhận diện. Nền phức tạp, như các cành cây, bức tranh hoặc đèn đường, cũng có thể làm cho biển báo trở nên khó nhìn thấy.

+ Đa dạng biển báo: Một hệ thống nhận diện biển báo giao thông cần được huấn luyện và kiểm tra trên nhiều loại biển báo khác nhau để đảm bảo tính đa dạng và hiệu quả.

CHƯƠNG 2:

HỆ THỐNG NHẬN DẠNG BIỂN BÁO GIAO THÔNG

2.1. Kiến trúc chương trình



Hình 2.1: Kiến trúc chương trình nhận dạng biển báo giao thông

Như được minh hoạ ở hình 2.1, hệ thống nhận dạng biển báo giao thông mà đề tài xây dựng gồm các bước cụ thể sau:

+ Chọn ảnh/camera

+ Chương trình nhận ảnh đầu vào/ dữ liệu từ camera

+Nhận dạng biển báo giao thông bằng mô hình YOLOv3

+ Hiển thị kết quả nhận dạng là hình ảnh có chứa khung và tên biển báo (nếu có)

2.2. Mạng YOLO3

YOLO là gì?

"You only look once" (YOLO) [13] là một thuật toán sử dụng các mạng nơ ron nhân chập để phát hiện đối tượng (trong ảnh/video) với mục đích nhắm tới một kỹ thuật thống nhất và chạy ở thời gian thực cho các bài toán phát hiện đối tượng (object detection). Bạn chỉ nhìn 1 lần, hay YOLO, là một trong các thuật toán phát hiện đối tượng tốt nhất hiện nay. Mặc dù đây không phải là một thuật toán có độ chính xác nhất, nhưng nó là một lựa chọn rất tốt khi chúng ta cần thực hiện phát hiện đối tượng theo thời gian thực, với độ chính xác không bị suy giảm quá nhiều.

So sánh với các thuật toán nhận dạng khác, một thuật toán phát hiện đối tượng không chỉ dự đoán về nhãn của các lớp mà còn phát hiện vị trí của các đối tượng trong dữ liệu input. Do đó, nó không chỉ phân lớp ảnh thành một danh mục, mà còn phát hiện nhiều đối tượng có 1 trong ảnh. Thuật toán này ứng dụng một mạng nơ ron đơn đối với cả bức ảnh (chỉ 1 mô hình mạng nơ ron). Điều đó có nghĩa là mạng này sẽ chia bức ảnh thành các vùng và dự đoán biên (bounding boxes-bb) và sác xuất của mỗi vùng. Các biên này được đánh trọng số bởi các giá trị sác xuất.

YOLO chỉ sử dụng các lớp nhân chập, do đó nó là một mạng nơ ron nhân chập đầy đủ (FCN). So với phiên bản đầu tiên (YOLO [13]) YOLO3 [11] (hay YOLOv3), sử dụng một kiến trúc mới, sâu hơn để trích chọn đặc trưng gọi là Darknet-53 (xem hình 2.2). Như cái tên của nó, Darknet-53 sử dụng 53 lớp nhân chập, mỗi lớp được theo sau bởi một lớp chuẩn hoá theo lô và một lớp kích hoạt sử dụng hàm Leaky ReLU. Không có lớp tổng hợp (pooling) nào được sử dụng, và một lớp nhân chập với giá trị bước (stride) bằng 2 được sử dụng để giảm kích thước của các feature map. Điều này giúp cho việc ngăn chặn các mất mát về các đặc trưng ở mức thấp thường gây ra do các lớp tổng hợp.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Hình 2.2: Mô hình mạng YOLO3 (Darknet53).

Mô hình mạng sẽ giảm kích thước của ảnh theo một tỉ lệ gọi là stride (sải/bước) của mạng. Chẳng hạn, nếu bước của mạng là 32, thì kích thước ảnh input từ 416x416 sẽ cho một ảnh kết quả có kích thước 13x13. Nói chung, bước của bất cứ một lớp nào trong mạng sẽ bằng với giá trị tỉ lệ theo đó giá trị output của lớp sẽ nhỏ hơn giá trị ảnh input của mạng.

2.3. YOLO3 phát hiện đối tượng như thế nào?

Trước hết, chúng ta cần hiểu một số thông tin cơ bản về mô hình mạng YOLO3 như sau:

+ Kích thước input là một lô các ảnh có kích thước (m, 416, 416, 3).

+ Kết quả đầu ra là một danh sách các bb cùng với lớp nhận dạng. Mỗi bb được biểu diễn bởi 6 giá trị (pc, bx, by, bh, bw, c). Nếu chúng ta muốn mở rộng giá trị c thành một vectỏ có 80 chiều thì mỗi bb sẽ được biểu diễn thành 85 số.

Tương tự như với tất cả các bộ phát hiện đối tượng, các đặc trưng học được qua các lớp nhân chập sẽ được truyền vào một bộ phân lớp/hồi quy để thực hiện việc dự đoán phát hiện (vị trí toạ độ của các bb, nhãn của lớp, …).

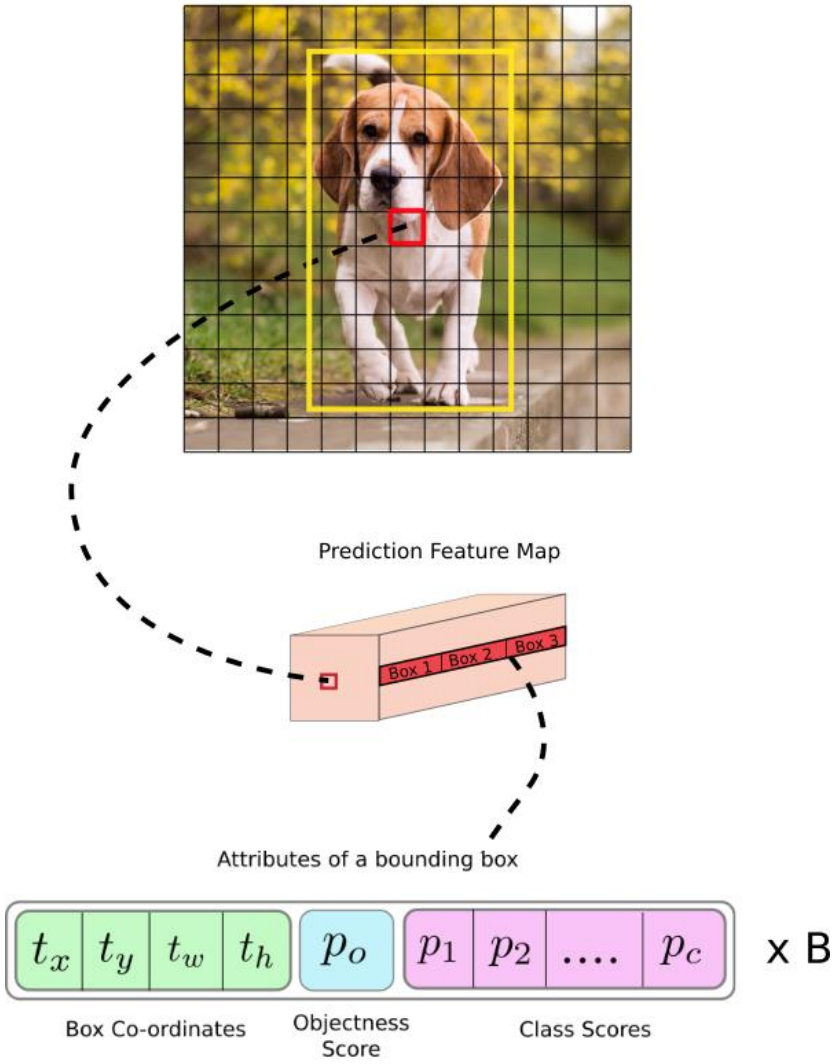
Trong YOLO, việc dự đoán được thực hiện bằng cách sử dụng một lớp nhân chập sử dụng các phép nhân chập 1x1. Do đó, điều đầu tiên cần chú ý là kết quả đầu ra sẽ là các feature map. Vì chúng ta chỉ có các phép nhân chập 1x1, kích thước của prediction map sẽ bằng với kích thước của feature map ngay trước nó. Trong YOLO v3, cách thức mà chúng ta diễn giải prediction mao này là mỗi cell sẽ có thể predict một số lượng cố định các bb.

Chẳng hạn nếu chúng ta có B*(5+C) giá trị tong feature map. B là số bb mà mỗi cell có thể dự đoán. Theo nội dung bài báo, mỗi một trong số B bb này có thể chuyên biệt cho việc dự đoán một loại đối tượng cụ thể. Mỗi bb phải có 5+C thuộc tính, mô tả các giá trị về toạ độ trung tâm, kích thước, giá trị điểm số và C độ tin cậy cho mỗi bb. YOLO v3 dự đoán 3 BB cho mỗi cell.

Chúng ta sẽ mong muốn mỗi cell của feature map sẽ dự đoán một đối tượng qua một trong số các bb nếu như trung tâm của đối tượng nằm trong trường nhận thức của cell đó.

Điều này phải được thực hiện qua việc YOLO huấn luyện mô hình mạng của nó, trong đó chỉ một BB sẽ chịu trách nhiệm cho việc phát hiện bất cứ đối tượng nào. Đầu tiên, chúng ta cần phải xác minh (ascertain) các ô mà bb này thuộc về.

Để làm điều đó, chúng ta sẽ chia bức ảnh input thành các lưới kích thước bằng với feature map cuối cùng. Hãy xem ví dụ minh hoạ 2.3 bên dưới, trong đó ảnh input là 416x416, và bước của mạng là 32. Như đã đề cập trước đó, kích thước của feature map cuối cùng sẽ là 13x13. Do đó chúng ta sẽ chia bức ảnh thành các ô 13x13.

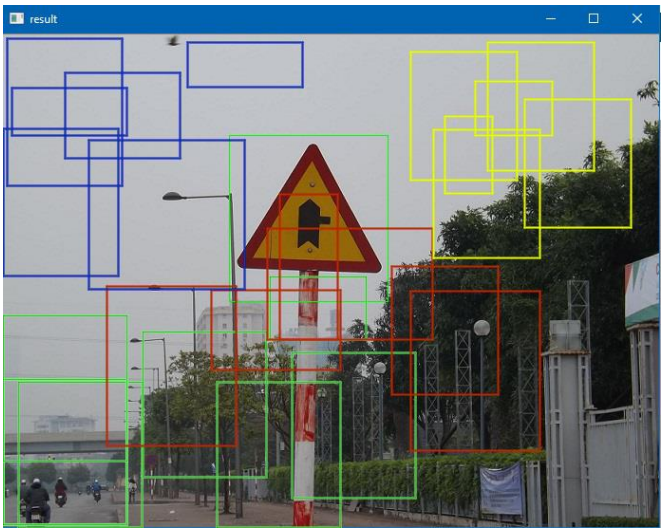


Hình 2.3: Cách thức mạng YOLO3 thực hiện phát hiện đối tượng.

Sau đó, ô - cell (trên bức ảnh input) chứa trung tâm của bb đúng của một đối tượng sẽ được chọn là ô chịu trách nhiệm cho việc dự đoán đối tượng. Trong bức ảnh, chính là cell được đánh dấu đỏ, chứa trung tâm của bb đúng (đánh dấu vàng).

Bây giờ, ô đỏ là ô số 7 trong hàng số 7 của lưới. Chúng ta sẽ gán ô số 7 trong hàng số 7 trên feature map (ô tương ứng trên feature map) là ô chịu trách nhiệm phát hiện đối tượng con chó. Ô đó có thể phát hiện được 3 bb. Vậy bb này sẽ được gán cho nhãn là con chó? Để có thể hiểu điều này, chúng ta cần phải nắm được khái niệm về các mỏ neo (anchors) (chú ý là ô mà chúng ta đang đề cập tới ở đây là ô nằm trên feature map dự đoán. Chúng ta chia ảnh input thành một lưới

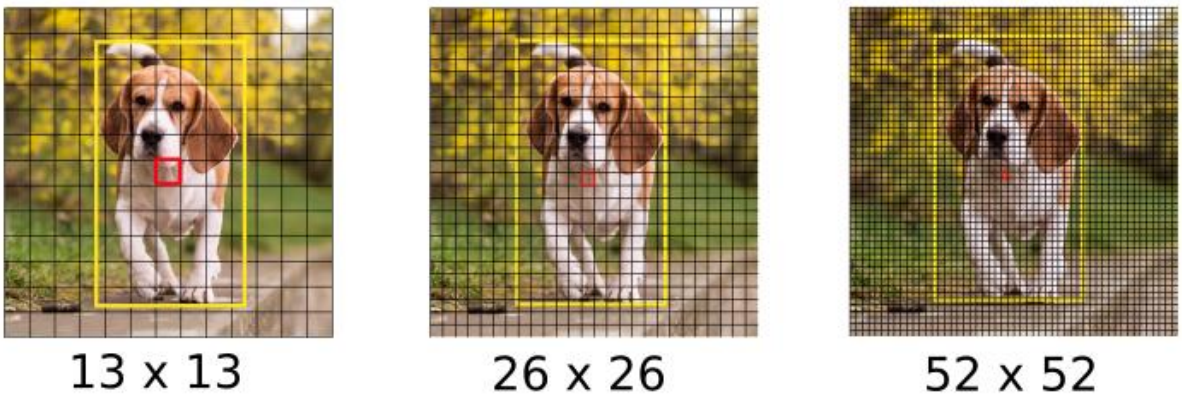
chỉ để xác định ô nào của feature map dự đoán sẽ chịu trách nhiệm cho việc dự doán).



Hình 2.4: Kết quả bước đầu do YOLO3 phát hiện đối tượng.

Trong hình vẽ trên, chúng ta chỉ vẽ các hộp mà mô hình đã gán các xác suất cao, nhưng vẫn còn có quá nhiều hộp. Chúng ta sẽ mong muốn thực hiện lọc kết quả output của thuật toán để có số lượng íthơn các đối tượng được phát hiện. Để đạt được điều đó, chúng ta cần sử dụng thuật toán loại bỏ các đối tượng không cực trị (Non Maximum Suppression – NMS) mà chúng tôi sẽ trình bày ở phần sau của tài liệu này.

Dự đoán ở các tỉ lệ khác nhau



Hình 2.5: Minh hoạ YOLO khi dự đoán đối tượng trong ảnh ở các tỉ lệ khác nhau.

YOLO v3 dự đoán ở 3 tỉ lệ khác nhau. Lớp dự đoán được sử dụng để dự đoán ở các feature map tại các kích thước khác nhau, có các bước khác nhau là 32, 16, 8 tương ứng. Điều này có nghĩa là với một bức ảnh đầu vào kích thước 416x416, chúng ta sẽ thực hiện detect ở các tỉ lệ 13x13, 26x26 và 52x52.

Mạng sẽ thực hiện giảm kích thước của ảnh input xuống tới lớp đầu tiên, nơi mà một phát hiện sẽ được thực hiện sử dụng feature map của một lớp với bước giảm bằng 32. Thêm nữa, các lớp sẽ được tăng tỉ lệ với một tỉ lệ bằng 2 và ghép vào các feature map của các lớp trước có cùng kích thước. Một phát hiện nữa sẽ được thực hiện với một bước giảm bằng 16. Quá trình tăng kích thước sẽ được lặp lại và phát hiện cuối cùng sẽ được thực hiện với bước giảm bằng 8.

Kỹ thuật loại bỏ các kết quả trùng lặp bằng lọc ngưỡng dựa trên độ tin cậy (hay điểm số dự đoán) và NMS

Với một bức ảnh input kích thước 416x416, YOLO sẽ phát hiện tổng số ((52x52) + (26x26) + (13x13))*3 = 10647 BB. Tuy nhiên, trong trường hợp ảnh của chúng ta, sẽ chỉ có 1 đối tượng, 1 con chó. Vậy làm thế nào để giảm từ 10647 xuống 1.

Đầu tiên chúng ta lọc các BB dựa trên điểm số (được YOLO dự đoán) của đối tượng. Nói chung các hộp sẽ có điểm số dưới 1 ngưỡng (chẳng hạn 0.5) sẽ bị bỏ qua. Tiếp đó, loại bỏ các đối tượng không cực trị sẽ được sử dụng để loại bỏ bớt các box. Chẳng hạn, tất cả 3 BB của ô màu đỏ có thể phát hiện 1 box hoặc các ô liền kề có thể phát hiện cùng đối tượng, khi đó NMS sẽ loại bỏ nhiều phát hiện.

Cụ thể, chúng ta sẽ thực hiện các bước sau:

+ Loại bỏ các box có điểm số thấp (không đủ độ tin cậy trong việc phát hiện 1 lớp).

+ Chỉ chọn một hộp khi có một vài hộp chồng lấn lên nhau và cùng phát hiện 1 đối tượng (sử dụng kỹ thuật loại bỏ các phần tử không phải cực trị).

Để hiểu rõ hơn, tôi sẽ sử dụng ví dụ với ảnh chiếc xe ô tô đã được đề cập ở trên. Đầu tiên, chúng ta sẽ áp dụng một bộ lọc thứ nhất bằng cách lọc theo ngưỡng. Điều này đơn giản là loại bỏ các hộp có giá trị điểm số nhỏ hơn 1 ngưỡng đã chọn.

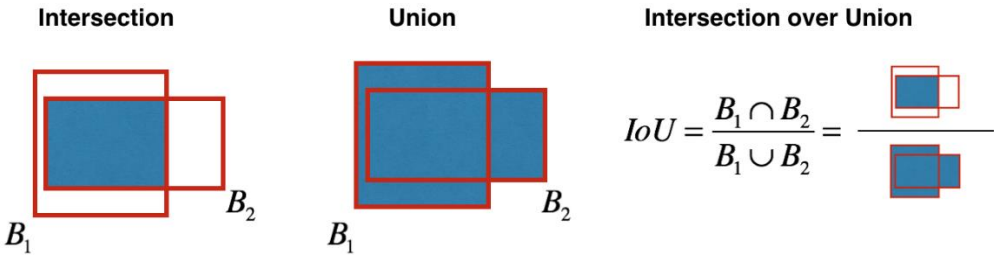
Mô hình cho chúng ta 19x19x5x85 giá trị, với mỗi hộp có 85 số. Tô sẽ thấy thuận tiện hơn khi sắp xếp 1 tensor (19, 19, 5, 85) hoặc (19, 19, 425) thành các biến sau:

+ độ tin cậy của hộp được dự đoán (box confidence): tensor (19x19, 5, 1) chứa pc xác suất cho mỗi một trong số 5 hộp được dự đoán bởi mỗi 19x19 ô.

+ các hộp (boxes): tensor (19, 19, 5, 4) chứa các giá trị (bx, by, bh, bw) cho mỗi một trong số 5 hộp của mỗi ô.

+ box_class_probs: tensor (19x19, 5, 80) chứa xác suất phát hiện cho mỗi một trong số 80 lớp với mỗi 1 ô dự đoán (trong số 5 ô).

Sau khi lọc bằng ngưỡng trên các điểm số lớp, chúng ta sẽ vẫn có nhiều hộp chồng lấn lên nhau. Một bộ lọc thứ hai để lựa chọn các hộp đúng được gọi là NMS. NMS sử dụng một hàm rất quan trọng là hàm "Intersection over Union" (IoU).



Hình 2.6: Hàm IoU của YOLO

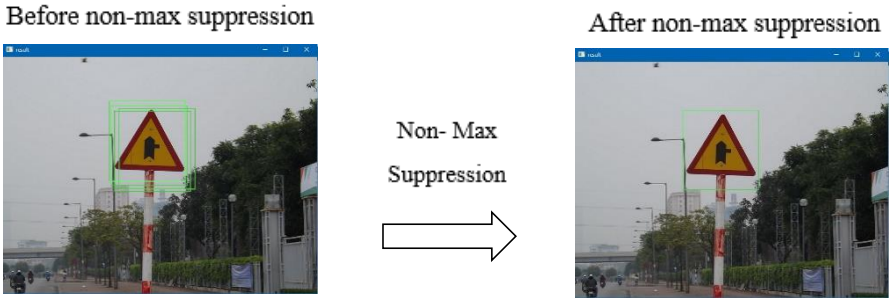
Các bước chính của kỹ thuật NMS là:

1. Lựa chọn hộp có giá trị điểm cao nhất.

2. Tính giá trị chồng lấn với tất cả các hộp khác, loại bỏ các hộp có độ chồng lấn lớn hơn giá trị iou_threshold.

3. Lặp lại bước 1 cho tới khi không có hộp nào có giá trị điểm nhỏ hơn ngưỡng.

Các bước này sẽ loại bỏ các hộp có độ chồng lấn lớn với các hộp được lựa chọn. Chỉ có các hộp tốt nhất được giữ lại (xem hình 2.7)



Hình 2.7: Minh họa kết quả của việc áp dụng kỹ thuật NMS.

2.4. Sử dụng mạng YOLO3 nhận dạng biển báo giao thông

Em đã thực hiện nghiên cứu và thấy rằng có 3 cách để có thể sử dụng một mô hình YOLOV3 để áp dụng vào bài toán của đề tài:

1. Xây dựng mô hình mạng YOLO3 sử dụng Tensorflow.
2. Sử dụng module dnn của thư viện OpenCV phiên bản 4.3
3. Sử dụng Darknet.

CHƯƠNG 3:
XÂY DỰNG HỆ THỐNG NHẬN DẠNG BIỂN BÁO GIAO THÔNG

3.1. Tìm hiểu Python và các thư viện lập trình

Chương trình được cài đặt bằng ngôn ngữ lập trình Python 3.9 và các thư viện hỗ trợ như OpenCV, Tkinter, Numpy, YOLOv3. Mô hình nhận dạng biển báo giao thông được huấn luyện dựa trên mô hình YOLOv3.

Python là một ngôn ngữ lập trình bậc cao thường được sử dụng để xây dựng trang web và phần mềm, tự động hóa các tác vụ và tiến hành phân tích dữ liệu. Python là ngôn ngữ có mục đích chung, nghĩa là nó có thể được sử dụng để tạo nhiều chương trình khác nhau và không chuyên biệt cho bất kỳ vấn đề cụ thể nào. Python cũng là một trong những ngôn ngữ lập trình phổ biến nhất hiện nay.

Python có nhiều ứng dụng với nhận dạng ảnh, chẳng hạn như phân tích dữ liệu và học máy, thị giác máy tính, các công nghệ nhận dạng (vân tay, khuôn mặt,…), xử lý ảnh cơ bản…

Các thư viện hỗ trợ

- OpenCV là một thư viện xử lý ảnh và thị giác máy tính, hỗ trợ nhiều thuật toán và ứng dụng như nhận dạng khuôn mặt, phát hiện đối tượng, xử lý video và nhiều hơn nữa. OpenCV có thể dùng để đọc, ghi, hiển thị và xử lý ảnh đầu vào, áp dụng các thuật toán nhận dạng khuôn mặt, phát hiện đối tượng, trích xuất đặc trưng và phân loại ảnh.
- Tkinter là một thư viện giao diện người dùng đồ họa (GUI) cho Python, cho phép tạo ra các cửa sổ, nút, menu và các thành phần khác. Tkinter có thể dùng để tạo giao diện người dùng cho chương trình, cho phép người dùng chọn ảnh đầu vào, điều khiển các chức năng và xem kết quả nhận dạng ảnh.
- Numpy là một thư viện tính toán khoa học cho Python, cung cấp các đối tượng mảng nhiều chiều và các hàm toán học để thao tác với chúng. Numpy có

thể dùng để lưu trữ và thao tác với dữ liệu ảnh dưới dạng mảng nhiều chiều, cũng như thực hiện các phép tính toán học cần thiết cho quá trình nhận dạng ảnh.

- PIL (Python Imaging Library) là một thư viện xử lý ảnh cho Python, cung cấp các hàm và phương thức để tạo, đọc, ghi, hiển thị và biến đổi ảnh. Pillow là một bản fork của PIL với nhiều cải tiến và bảo trì thường xuyên. PIL hoặc Pillow có thể dùng để chuyển đổi định dạng ảnh, điều chỉnh kích thước, độ sáng, độ tương phản và các thuộc tính khác của ảnh. Cũng có thể dùng để vẽ lên ảnh để hiển thị kết quả nhận dạng ảnh.

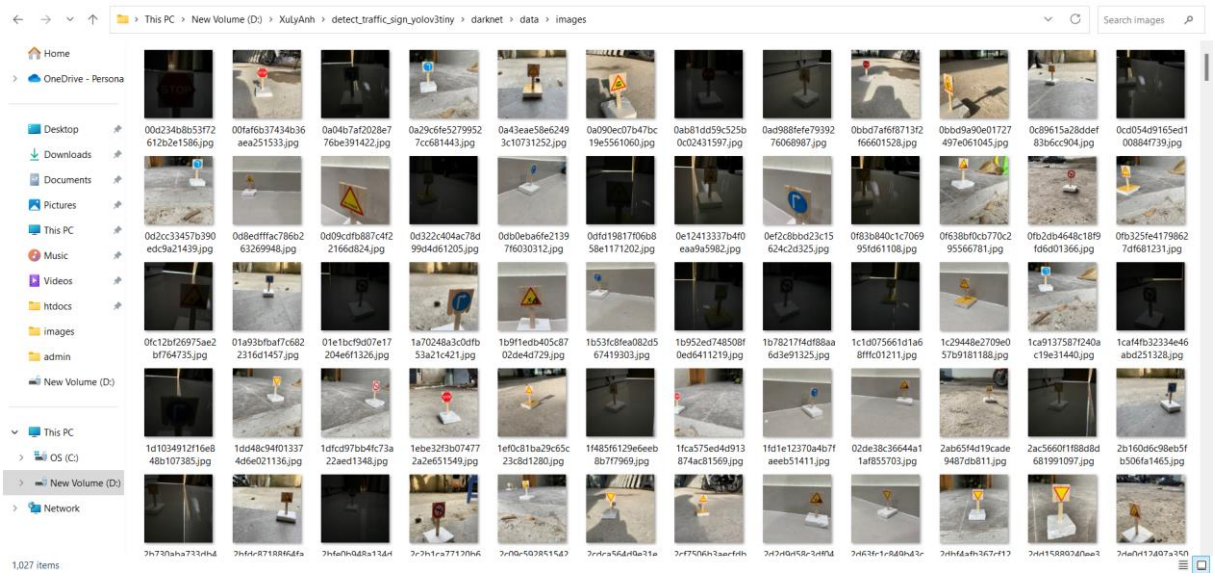
3.2. Train mô hình nhận dạng biển báo giao thông

3.2.1. Bộ dữ liệu

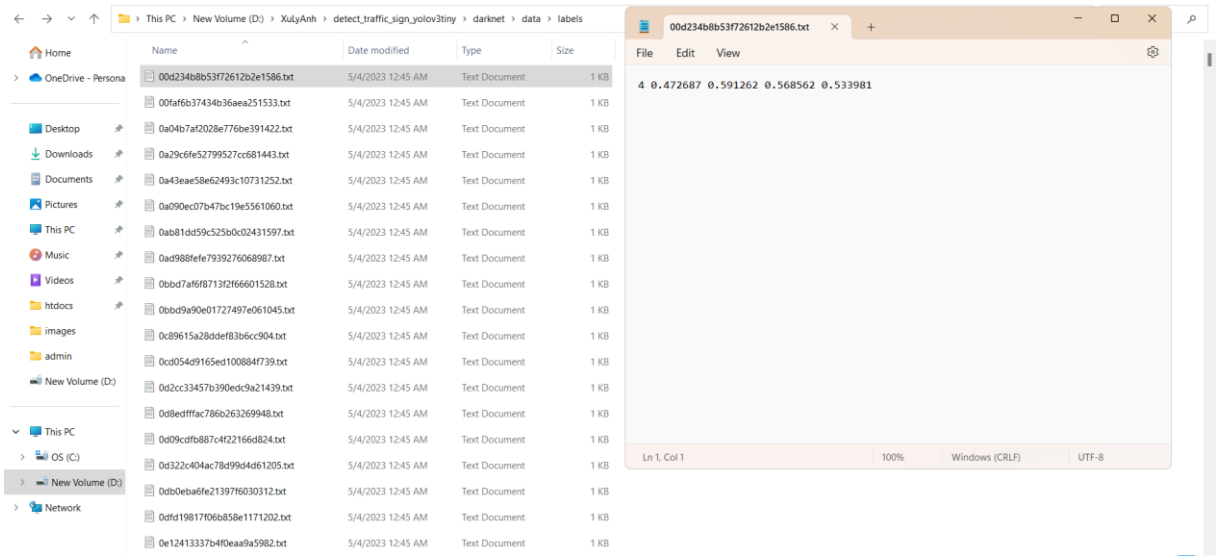
Bộ dữ liệu mới về logo 8 loại biển báo giao thông được xây dựng dựa trên cấu trúc COCO128 bao gồm images (chứa các ảnh) và labels (chứa các nhãn tọa độ của ảnh tương ứng).

Bao gồm: biển báo rẽ phải, biển báo rẽ trái, giao với đường ưu tiên, giao với đường sắt, dừng lại, đoạn đường nguy hiểm, đi chậm, cấm rẽ, công trường đang thi công.

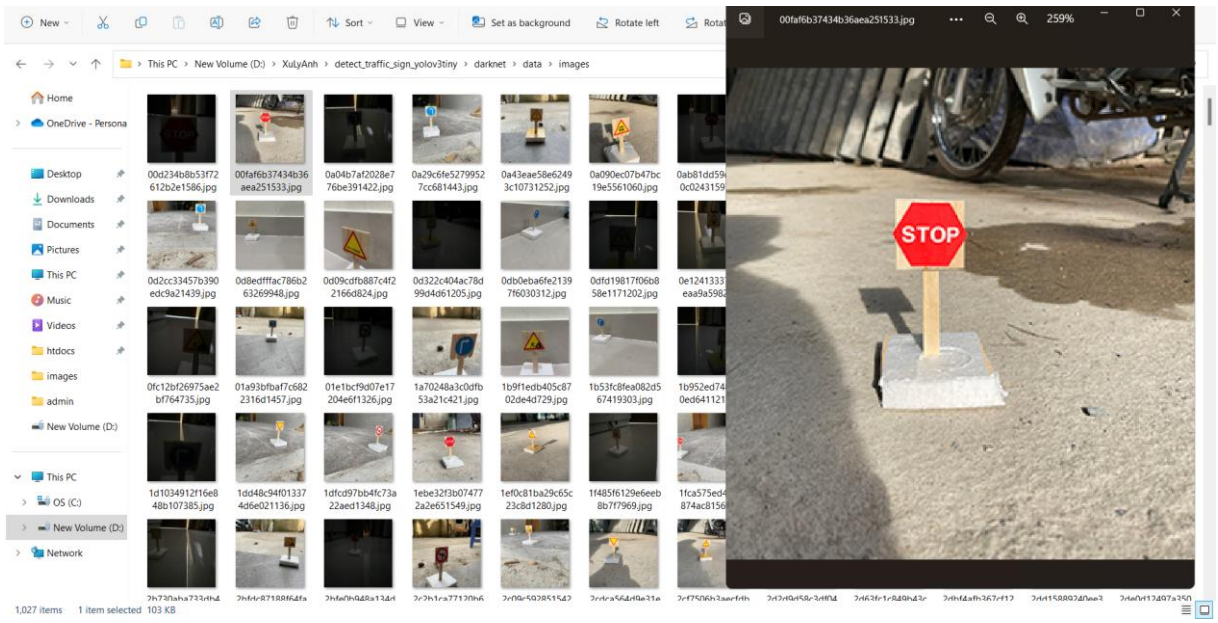
Kho dữ liệu mới dùng để train mô hình gồm 1027 file ảnh chứa 8 loại biển báo giao thông và 1027 file .txt chứa nhãn tương ứng.



Hình 3.2.1: Bộ dữ liệu



Hình 3.2.2: Bộ dữ liệu



Hình 3.2.3: Bộ dữ liệu

3.2.2. Tiến hành train

Mở YOLOv3 trên Google Colab

Tiến hành kết nối với google driver

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

Hình 3.2.4: Kết nối với google driver

Tiến hành tạo thư mục, tải xuống và giải nén dữ liệu cần thiết để train

```
!mkdir /content/drive/MyDrive/data
!mkdir /content/drive/MyDrive/data/yolo
!mkdir /content/drive/MyDrive/data/yolo/backup
!mkdir /content/drive/MyDrive/data/yolo/obj
!pip install gdown
import gdown
url1 = 'https://drive.google.com/uc?id=1SbWsfFltuy5SYwyq3ABT1InE3CC9DBnC'
url2 = 'https://drive.google.com/uc?id=1RuPzVY8xsVcSg-Yd_stC1exeVUGxYYeI'
url3 = 'https://drive.google.com/uc?id=1gglycNcqYKLsoQWdwhU2bBZtPtNKhlS7'
url4 = 'https://drive.google.com/uc?id=1O_Ay9XZ41ejDPEWpXrb1q4w5JuU1L_V8'
url5 = 'https://drive.google.com/uc?id=1BMyEtSm1GdKoKGJ4Ln7P6KBh-VFbbI03'
url6 = 'https://drive.google.com/uc?id=1G2oVkg0_SQjoHcsQUtbA2qi4vVBFXw5Q'
url7 = 'https://drive.google.com/uc?id=1srqfTf6YsnsDontoFkuRnErFIP4xj62U'

output1 = '/content/drive/MyDrive/data/obj.zip'
output2 = '/content/drive/MyDrive/data/yolo/obj.data'
output3 = '/content/drive/MyDrive/data/yolo/obj.names'
output4 = '/content/drive/MyDrive/data/yolo/train.txt'
output5 = '/content/drive/MyDrive/data/yolo/backup/yolov3-tiny-obj_last.weights'
output6 = '/content/drive/MyDrive/data/yolo/yolov3-tiny-obj.cfg'
output7 = '/content/drive/MyDrive/data/yolo/yolov3-tiny.conv.15'

gdown.download(url1, output1)
gdown.download(url2, output2)
gdown.download(url3, output3)
gdown.download(url4, output4)
gdown.download(url5, output5)
gdown.download(url6, output6)
gdown.download(url7, output7)
!unzip /content/drive/MyDrive/data/obj.zip -d /content/drive/MyDrive/data/yolo/obj/
```

Hình 3.2.5: Tạo thư mục, tải xuống và giải nén dữ liệu

Tiến hành train

```
# use this to download a file
def download(path):
  from google.colab import files
  files.download(path)


!./darknet detect cfg/yolov3-tiny.cfg yolov3-tiny.weights data/dog.jpg
imShow('predictions.jpg')

!cp /content/drive/MyDrive/data/yolo/obj.names ./data/
!cp /content/drive/MyDrive/data/yolo/obj.data  ./data/
!cp /content/drive/MyDrive/data/yolo/train.txt  ./data/
!cp /content/drive/MyDrive/data/yolo/yolov3-tiny-obj.cfg ../darknet/
```

```
/mydrive
Cloning into 'darknet'...
remote: Enumerating objects: 15514, done.
remote: Total 15514 (delta 0), reused 0 (delta 0), pack-reused 15514
Receiving objects: 100% (15514/15514), 14.17 MiB | 11.98 MiB/s, done.
Resolving deltas: 100% (10412/10412), done.
/content/darknet
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2022 NVIDIA Corporation
Built on Wed_Sep_21_10:33:58_PDT_2022
Cuda compilation tools, release 11.8, V11.8.89
Build cuda_11.8.r11.8/compiler.31833905_0
mkdir -p ./obj/
```

Sau khi train, sẽ thu được file backup.

Hình 3.2.6: Train

3.2.3. Thử nghiệm kết quả

Dưới đây là một số kết quả thử nghiệm, kết quả 29/32 ảnh nhận dạng đúng biển báo giao thông.

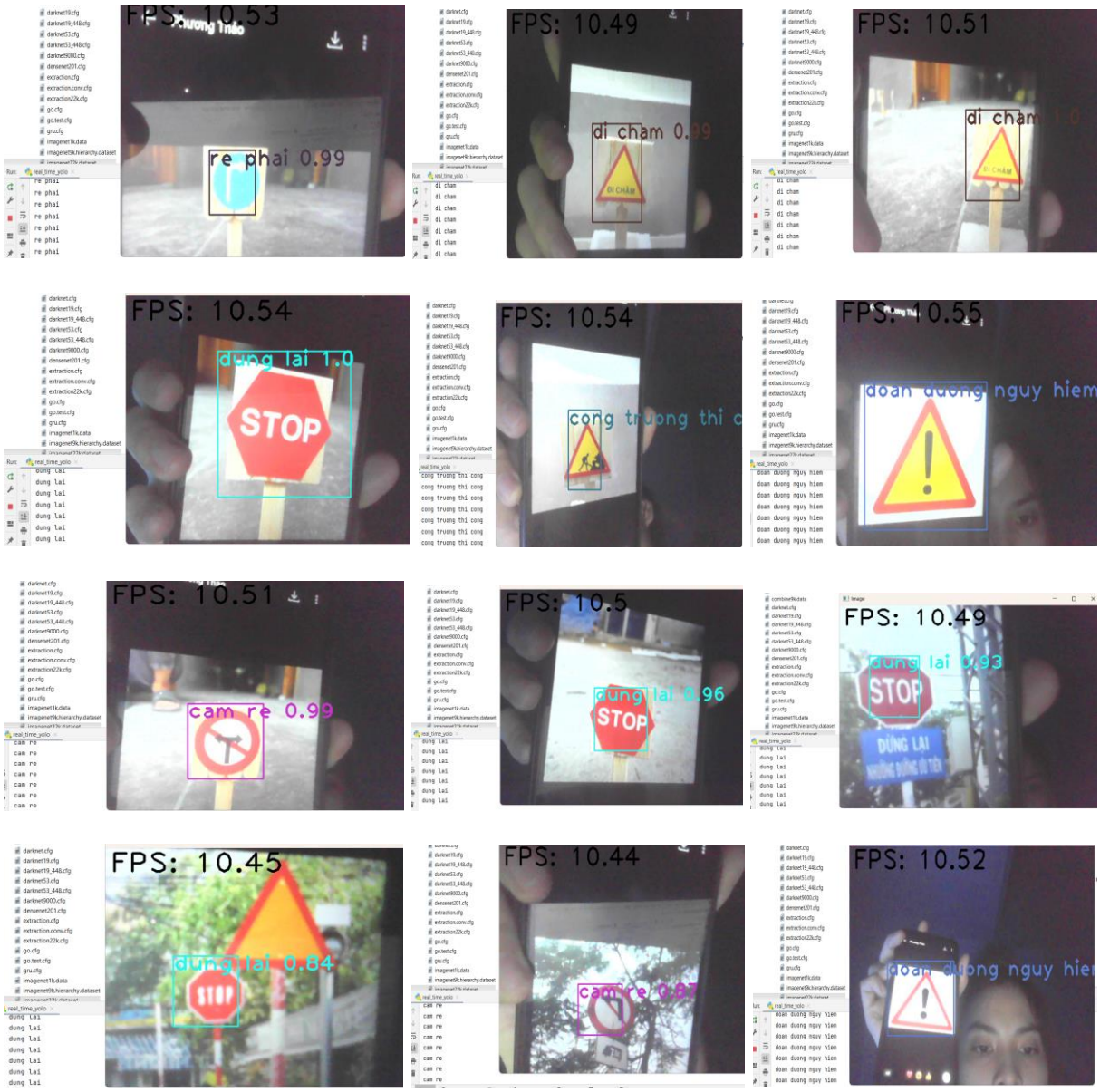
Trường hợp sai đầu, nhầm lẫn biển báo rẽ trái thành biển báo đoạn đường nguy hiểm và biển báo đi chậm với phần trăm chính xác là 64%, do ảnh nhỏ, biển báo bị mờ, ánh sáng chói từ màn hình dẫn đến nhận dạng không chính xác. Trường hợp 2, chương trình không nhận dạng được biển báo rẽ trái, lí do nhầm lẫn hướng rẽ với biển báo rẽ phải. Trường hợp 3, đối với hình ảnh có 2 biển báo trong cùng 1 hình thì nhận diện được 1 biển báo ví dụ hình ảnh có biển báo dừng lại và biển báo giao với đường ưu tiên, lí do vì biển báo giao với đường ưu tiên bị lệch màu, có một vài vết chấm bị mất màu.

Phương án khắc phục:

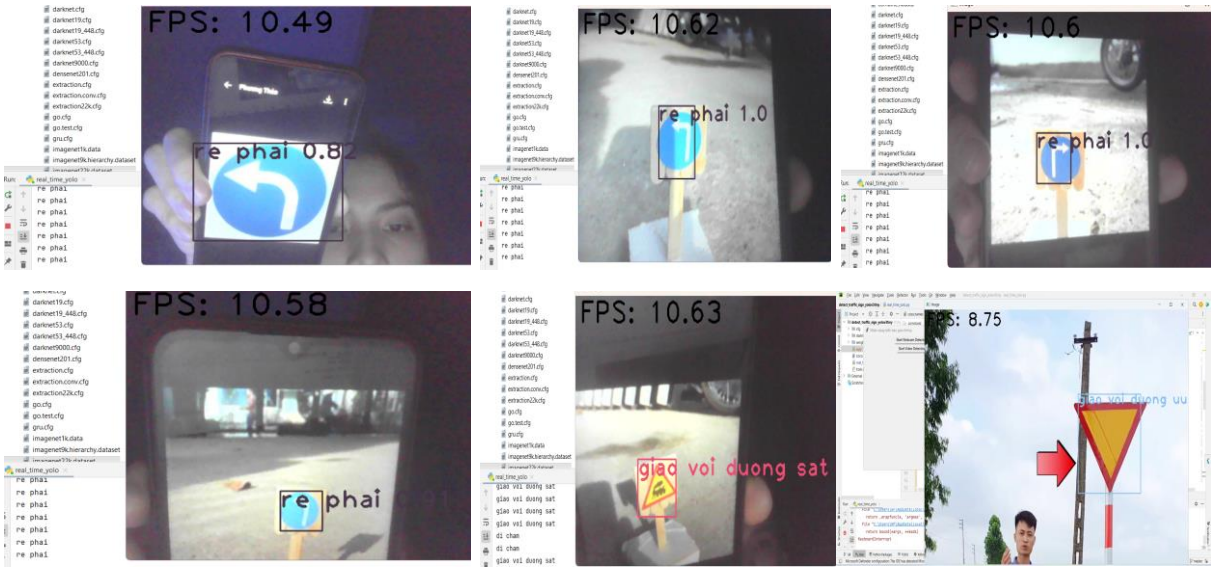
- Bổ sung nhiều ảnh biển báo với nhiều góc độ, khía cạnh giúp phân biệt giữa các biển báo với nhau vào tập dữ liệu đào tạo. Đồng thời tăng cường độ tương phản và sáng ảnh để tăng độ nhận diện và tổng quát hóa của mô hình.

- Kiểm tra lại tập dữ liệu xem có bị sai nhãn trong quá trình gán tọa độ hay không

Hình 3.2.7: Test



Hình 3.2.8: Test



Hình 3.2.9: Test

Để đánh giá kết quả của hệ thống test trên, em đã thu thập dữ liệu là các hình ảnh của các biển báo giao thông từ 2 nguồn: Từ Internet và tự chụp một số ảnh trên tuyến đường Bạch Thái Bưởi, Hải Phòng.

3.3. Chương trình nhận dạng biển báo giao thông

3.3.1. Code

Dưới đây là một số đoạn code chính trong chương trình:

```
net = cv2.dnn.readNet("weights/yolov3-tiny-obj_last.weights", "cfg/yolov3-tiny-obj.cfg")
classes = []
with open("coco.names", "r") as f:
    classes = [line.strip() for line in f.readlines()]
layer_names = net.getLayerNames()
output_layers = [layer_names[i - 1] for i in net.getUnconnectedOutLayers()]
colors = np.random.uniform(0, 255, size=(len(classes), 3))
```

Hình 3.3.1: Code tải mô hình YOLO

Đoạn code trên được sử dụng để tải mô hình YOLO và các tệp cấu hình liên quan, danh sách các lớp và mảng màu sắc. Và được chạy trước khi bắt đầu nhận dạng biển báo.

```
def start_webcam_detection():
    cap = cv2.VideoCapture(0)  # Open webcam
    start_detection(cap)
```

Hình 3.3.2: Hàm bắt đầu webcam

```
def start_image_detection():
    file_path = filedialog.askopenfilename(initialdir=".", title="Chọn file ảnh", filetypes=(("Image files", "*.jpg;*.jpeg;*.png"), ("All files", "
    if file_path:
        image = cv2.imread(file_path)
        detect_objects(image)
```

Hình 3.3.3: Hàm bắt đầu ảnh

```
def start_video_detection():
    file_path = filedialog.askopenfilename(initialdir=".", title="Select Video File", filetypes=(("Video files", "*.mp4"), ("All files", "*.*")))
    if file_path:
        cap = cv2.VideoCapture(file_path)  # Open video file
        if not cap.isOpened():
            print("Failed to open video file.")
            return
        start_detection(cap)
```

Hình 3.3.3: Hàm truy cập video

Đoạn code đã trên dùng để mở hộp thoại truy cập folder và cho phép người dùng chọn tệp video từ hệ thống tệp của họ. Hộp thoại sẽ chỉ hiển thị các tệp có phần mở rộng là .mp4 hoặc tất cả các tệp (*.*). Sau khi chọn một tệp video, hàm

start_video_detection() sẽ tiếp tục kiểm tra xem video có được mở thành công không. Nếu không sẽ thông báo "Failed to open video file.". Nếu tệp video được mở thành công, hàm start_detection() sẽ được gọi để bắt đầu quá trình nhận dạng trên video đã chọn.

```
# Tạo cửa sổ chính
window = tk.Tk()
window.title("Nhận dạng biển báo giao thông")
window.geometry("400x400")

# Tạo các nút
webcam_button = tk.Button(window, text="Nhận dạng bằng webcam", command=start_webcam_detection)
webcam_button.pack()

video_button = tk.Button(window, text="Nhận dạng bằng video", command=start_video_detection)
video_button.pack()

image_button = tk.Button(window, text="Nhận dạng bằng ảnh", command=start_image_detection)
image_button.pack()

# Chạy vòng lặp sự kiện Tkinter
window.mainloop()
```

Hình 3.3.4: Code tạo giao diện

Đoạn code này sẽ tạo ra một cửa sổ có tiêu đề "Nhận dạng biển báo giao thông" với kích thước 400x400 pixel. Cửa sổ sẽ có hai nút: "Bắt đầu phát hiện webcam" và "Bắt đầu phát hiện video". Nhấp vào các nút này sẽ kích hoạt các chức năng tương ứng để bắt đầu phát hiện webcam hoặc video để nhận dạng biển báo giao thông.

```
def detect_objects(frame):
    font = cv2.FONT_HERSHEY_PLAIN

    height, width, channels = frame.shape

    # Detecting objects
    blob = cv2.dnn.blobFromImage(frame, 0.00392, (416, 416), (0, 0, 0), True, crop=False)

    net.setInput(blob)
    outs = net.forward(output_layers)
```

```
def start_detection(cap):
    font = cv2.FONT_HERSHEY_PLAIN
    starting_time = time.time()
    frame_id = 0
    while True:
        _, frame = cap.read()
        frame_id += 1

        height, width, channels = frame.shape

        # Detecting objects
        blob = cv2.dnn.blobFromImage(frame, 0.00392, (416, 416), (0, 0, 0), True, crop=False)

        net.setInput(blob)
        outs = net.forward(output_layers)
```

Hình 3.3.5: Code nhận dạng đối tượng

Ở đoạn code trên các biến font, starting_time, và frame_id được khởi tạo để tính toán FPS (Frames Per Second). Trong vòng lặp while True, nó sẽ đọc từng khung hình từ đối tượng cap bằng cách sử dụng phương thức cap.read(). Số lượng khung hình được đếm bởi biến frame_id. Kích thước (chiều cao, chiều rộng, số kênh) của khung hình được lấy bằng cách sử dụng frame.shape. Tiếp theo sẽ thực hiện quá trình nhận dạng đối tượng bằng cách chuyển khung hình thành dữ liệu đầu vào phù hợp với mô hình YOLO thông qua cv2.dnn.blobFromImage(). Dữ liệu này sẽ được đưa vào mạng neural để nhận dạng đối tượng trong khung hình. Sau đó, chúng ta sử dụng net.setInput() để đưa dữ liệu vào mạng neural và sử dụng net.forward() để lấy các đầu ra dự đoán từ mạng. Các đầu ra này chứa thông tin về các đối tượng được phát hiện trong khung hình.

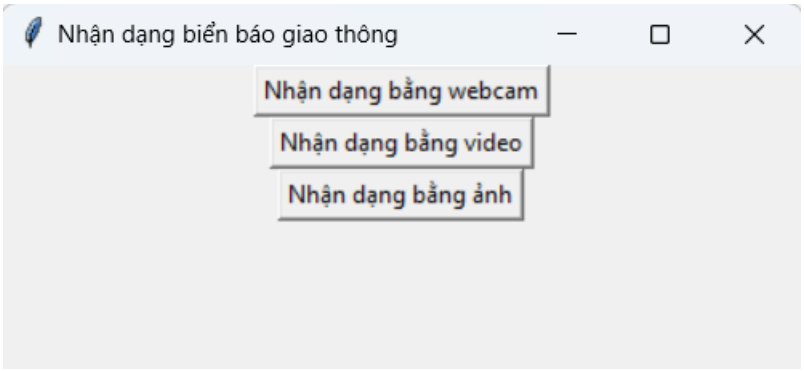
```
    class_ids = []
    confidences = []
    boxes = []
    for out in outs:
        for detection in out:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]
            if confidence > 0.2:
                # Object detected
                center_x = int(detection[0] * width)
                center_y = int(detection[1] * height)
                w = int(detection[2] * width)
                h = int(detection[3] * height)

                # Rectangle coordinates
                x = int(center_x - w / 2)
                y = int(center_y - h / 2)

                boxes.append([x, y, w, h])
                confidences.append(float(confidence))
                class_ids.append(class_id)

    indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.8, 0.3)
```

```
    for i in range(len(boxes)):
        if i in indexes:
            x, y, w, h = boxes[i]
            label = str(classes[class_ids[i]])
            confidence = confidences[i]
            color = colors[class_ids[i]]
            cv2.rectangle(frame, (x, y), (x + w, y + h), color, 2)
            cv2.putText(frame, label + " " + str(round(confidence, 2)), (x, y + 30), font, 3, color, 3)

    elapsed_time = time.time() - starting_time
    fps = frame_id / elapsed_time
    cv2.putText(frame, "FPS: " + str(round(fps, 2)), (10, 50), font, 4, (0, 0, 0), 3)
    cv2.imshow("Image", frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Hình 3.3.6: Code hiển thị kết quả

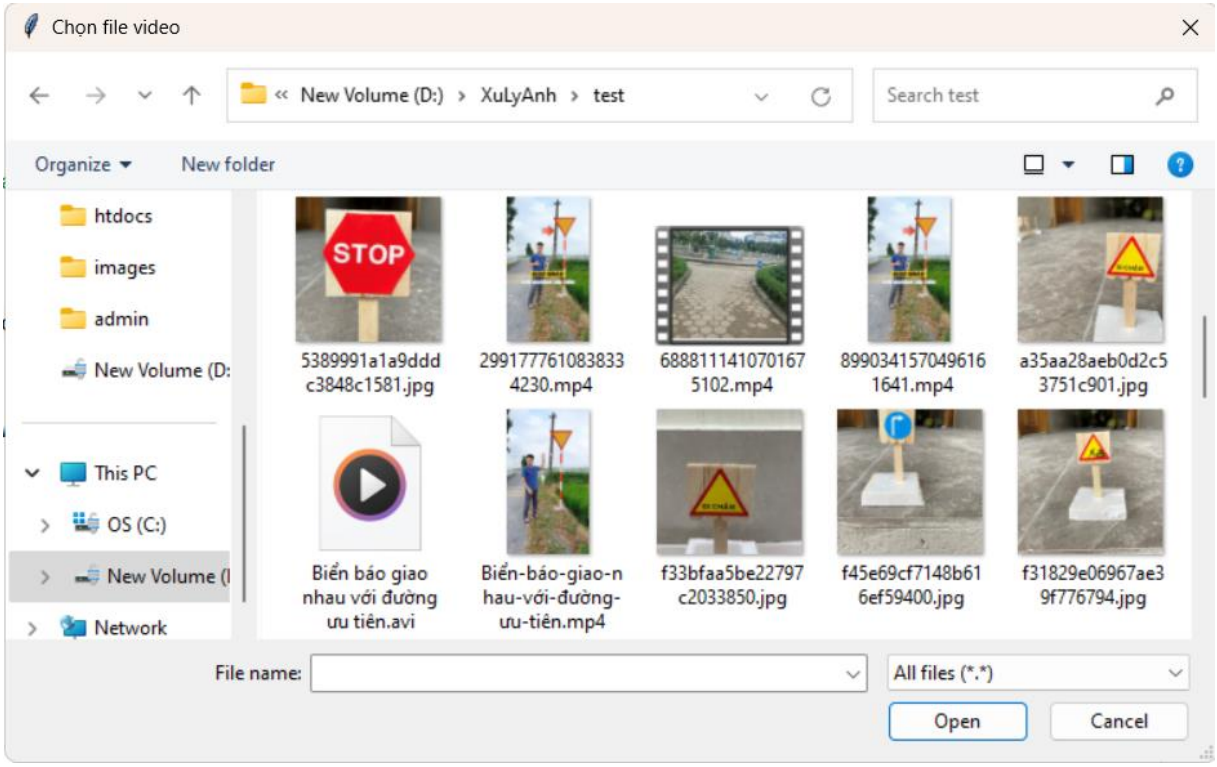
Đoạn code này dùng để hiện ảnh kết quả bao gồm khung bao quanh biển báo và nhãn chứa tên biển báo giao thông cùng mức độ chính xác.

3.3.2. Giao diện

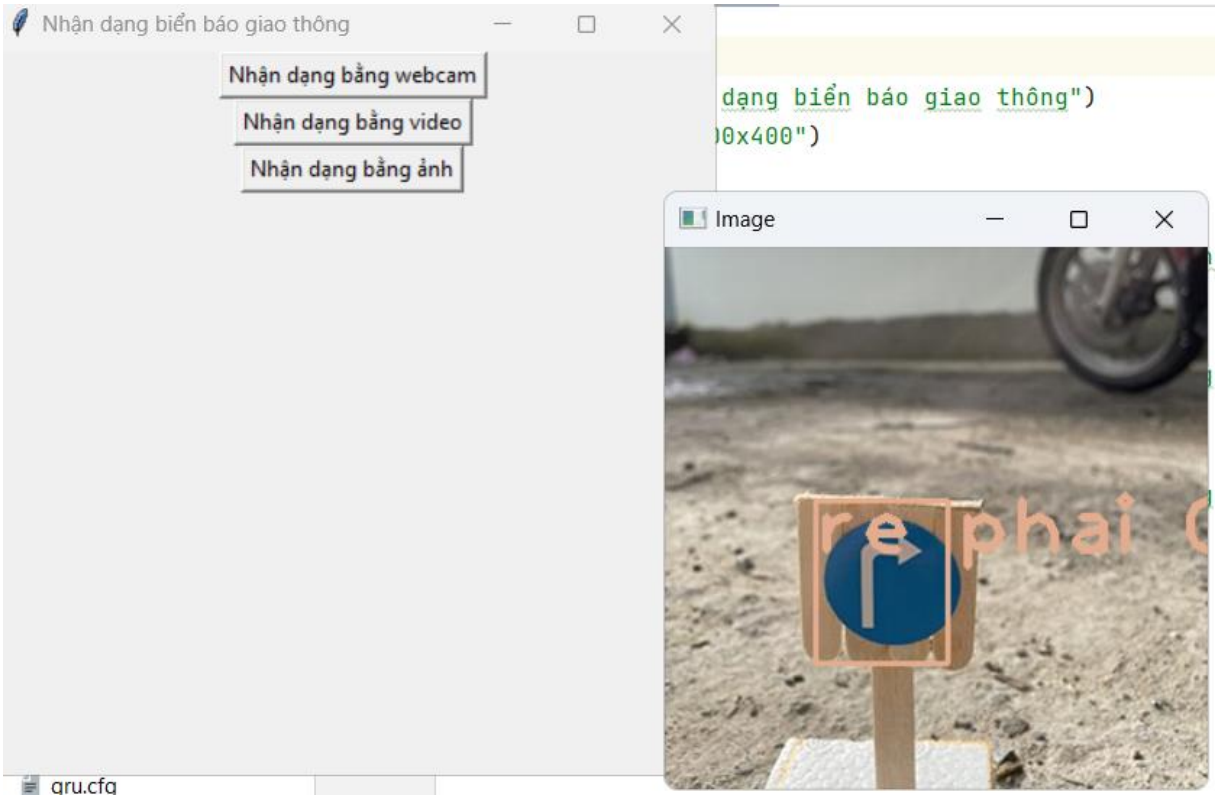


Hình 3.3.7: Giao diện chính

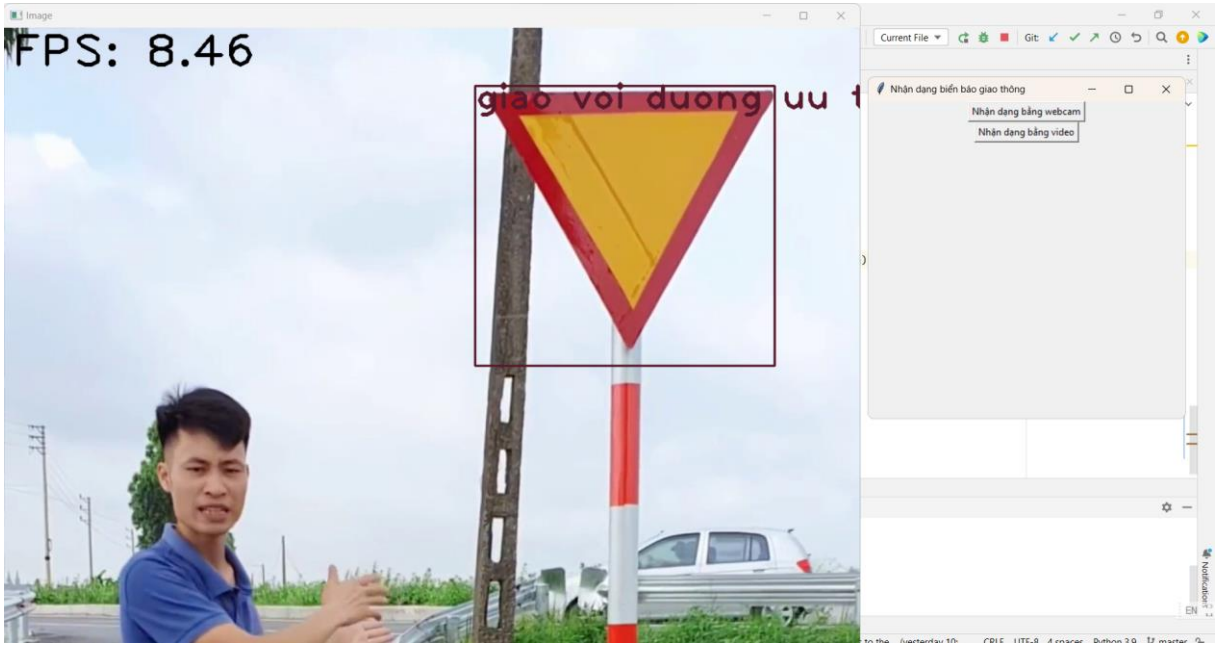
Sau khi truy cập vào giao diện chính, kích button "Nhận dạng bằng video". Tại đây, chọn "Video" -> "Kiểm tra" để mở file video trong máy tính. Có thể lựa chọn các file .mp4, .avi, và một số định dạng khác mà python version 4.x trong from chọn file video.



Hình 3.3.8: Form chọn file video hoặc file ảnh



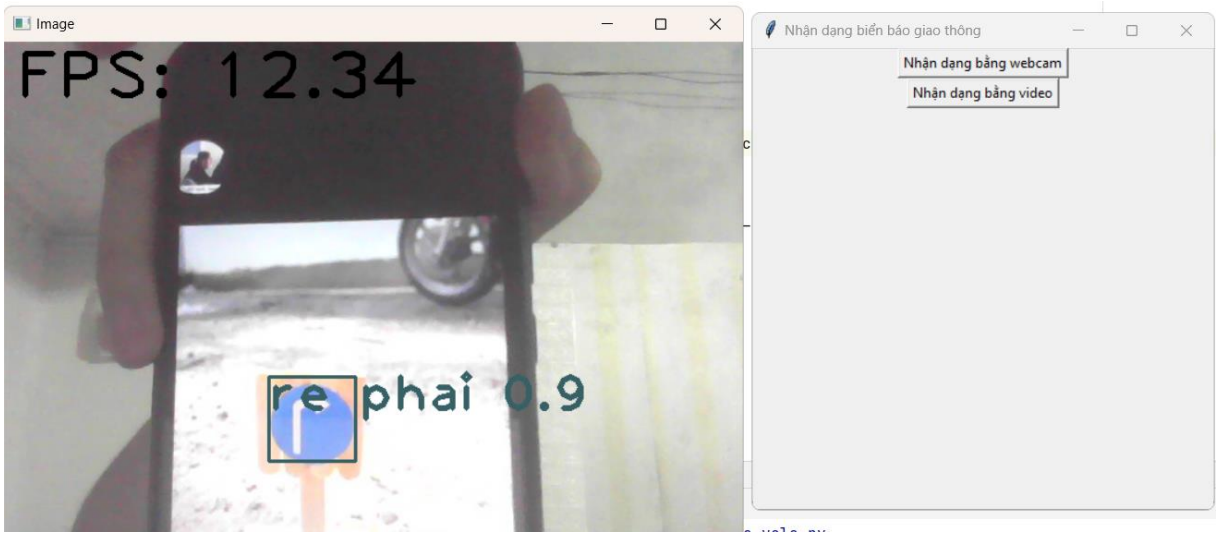
Hình 3.3.9: Kết quả khi nhận diện bằng ảnh



Hình 3.3.9: Video kết quả

Sau khi ấn "Open" video đã chọn được hiện lên trong from "Image" đây là form kết quả sau khi mô hình truy cập được video và nhận diện biển báo:

- Giao nhau với đường ưu tiên: Tên biển báo giao thông



Hình 3.3.10: Ảnh kết quả khi nhận diện bằng webcam

Chương trình sẽ nhận dạng biển báo giao thông sẽ hiển thị khung và nhãn khi độ chính xác nhận diện lớn hơn hoặc bằng 0.4 , kèm theo FPS (Số khung hình trên giây).

KẾT LUẬN

Kết thúc nghiên cứu, đề tài đã thực hiện được những vấn đề là:

- Mô hình nhận dạng biển báo giao thông được huấn luyện dựa trên mô hình YOLOv3.
- Xây dựng một chương trình nhận dạng biển báo giao thông sử dụng mô hình YOLOv3.
- Chương trình được cài đặt bằng ngôn ngữ lập trình Python và các thư viện hỗ trợ như OpenCV, Tkinter, Numpy, YOLOv3, time.

Chương trình chưa hoàn toàn được ưu việt, thời gian chạy còn lâu và nhận dạng sai đối với các ảnh nhiễu hay mờ. Để bổ sung cần thêm nhiều dữ liệu ảnh khác nhau để mô hình có thể nhận dạng chính xác hơn.

Em trong tương lai mong muốn bổ sung thêm dữ liệu để học tăng cường nhằm tránh các nhầm lẫn có thể xảy ra.

Em xin chân thành cảm ơn!

TÀI LIỆU THAM KHẢO

[1] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *ArXiv180402767 Cs*, Apr. 2018, Accessed: May 21, 2020. [Online]. Available: http://arxiv.org/abs/1804.02767.

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," Jun. 2015, Accessed: May 22, 2020. [Online]. Available: https://arxiv.org/abs/1506.02640v5.

[3] *opencv/opencv*. OpenCV, 2020.

[4] Alexey, *AlexeyAB/darknet*. 2020.

[5] Object detection: Chapter 2 PyimageSearch Gurus - Adrian Rosebrock