BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC NHA TRANG
KHOA CÔNG NGHỆ THÔNG TIN

CHUYÊN ĐỀ TỐT NGHIỆP

SỬ DỤNG THỊ GIÁC MÁY TÍNH ĐỂ PHÁT HIỆN ĐỐI TƯỢNG NGƯỜI ĐI BỘ TRONG NHÀ VÀ CẢNH BÁO XÂM NHẬP

Giảng viên hướng dẫn: TS. Nguyễn Khắc Cường
Sinh viên thực hiện: Trương Tấn Nghĩa
Mã số sinh viên: 61131950

Khánh Hòa – 2023

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC NHA TRANG
KHOA CÔNG NGHỆ THÔNG TIN

CHUYÊN ĐỀ TỐT NGHIỆP

SỬ DỤNG THỊ GIÁC MÁY TÍNH ĐỂ PHÁT HIỆN ĐỐI TƯỢNG NGƯỜI ĐI BỘ TRONG NHÀ VÀ CẢNH BÁO XÂM NHẬP

GVHD: TS. Nguyễn Khắc Cường
SVTH: Trương Tấn Nghĩa
MSSV: 61131950

Khánh Hòa – Tháng 06/2023

TRƯỜNG ĐẠI HỌC NHA TRANG
Khoa Công Nghệ Thông Tin

PHIẾU THEO DÕI TIẾN ĐỘ VÀ ĐÁNH GIÁ CHUYÊN ĐỀ TỐT NGHIỆP

(Dùng cho CBHD và nộp cùng báo cáo CĐTN của sinh viên)

Tên đề tài: Sử dụng thị giác máy tính để phát hiện đối tượng người đi bộ trong nhà và cảnh báo xâm nhập

Chuyên ngành: Công nghệ thông tin

Họ và tên sinh viên: Trương Tấn Nghĩa

Mã sinh viên: 61131950

Người hướng dẫn (học hàm, học vị, họ và tên) : TS. Nguyễn Khắc Cường

Cơ quan công tác: Khoa CNTT, Trường ĐH Nha Trang

Phần đánh giá và cho điểm của người hướng dẫn (tính theo thang điểm 10)

Tiêu chí đánh giá	**Trọng số (%)**	**Mô tả mức chất lượng**				**Điểm**
		Giỏi	**Khá**	**Đạt yêu cầu**	**Không đạt**	
		9 - 10	**7 - 8**	**5 - 6**	**< 5**	
Xây dựng đề cương nghiên cứu	10					
Tinh thần và thái độ làm việc	10					
Kiến thức và kỹ năng làm việc	10					
Nội dung và kết quả đạt được	40					
Kỹ năng viết và trình bày báo cáo	30					
ĐIỂM TỔNG						

Ghi chú: Điểm tổng làm tròn đến 1 số lẻ.

Nhận xét chung (sau khi sinh viên hoàn thành CĐTN):

……..……

…….

……..……

Đồng ý cho sinh viên: Được chấm phản biện: □ Không được chấm phản biện: □

Khánh Hòa, ngày tháng năm 2023

Cán bộ hướng dẫn

(Ký và ghi rõ họ tên)

Nguyễn Khắc Cường

TRƯỜNG ĐẠI HỌC NHA TRANG
Khoa Công Nghệ Thông Tin

PHIẾU CHẤM ĐIỂM CHUYÊN ĐỀ TỐT NGHIỆP

(Dành cho cán bộ chấm phản biện)

Tên đề tài: Sử dụng thị giác máy tính để phát hiện đối tượng người đi bộ trong nhà và cảnh báo xâm nhập

Chuyên ngành: Công nghệ thông tin

Họ và tên sinh viên : Trương Tấn Nghĩa

Mã sinh viên: 61131950

Người phản biện (học hàm, học vị, họ và tên) :

.....

Cơ quan công tác:

.....

Phần đánh giá và cho điểm của người phản biện (tính theo thang điểm 10)

Tiêu chí đánh giá	**Trọng số (%)**	**Mô tả mức chất lượng**				**Điểm**
		Giỏi	**Khá**	**Đạt yêu cầu**	**Không đạt**	
		9 - 10	**7 - 8**	**5 - 6**	**< 5**	
Hình thức bản thuyết minh	10					
Nội dung bản thuyết minh	30					
Mức độ trích dẫn và sao chép	10					
Kết quả nghiên cứu đạt được	30					
Mức độ thể hiện kiến thức	20					
ĐIỂM TỔNG						

Ghi chú: Điểm tổng làm tròn đến 1 số lẻ.

Khánh Hòa, ngày……. tháng……năm…….

Cán bộ chấm phản biện

(Ký và ghi rõ họ tên)

LỜI CAM ĐOAN

Tôi xin được cam đoan: Đề tài tốt nghiệp "**SỬ DỤNG THỊ GIÁC MÁY TÍNH ĐỂ PHÁT HIỆN ĐỐI TƯỢNG NGƯỜI ĐI BỘ TRONG NHÀ VÀ CẢNH BÁO XÂM NHẬP**" là kết quả dựa trên sự cố gắng, nỗ lực của bản thân với sự hướng dẫn nhiệt tình và tận tâm của TS. NGUYỄN KHẮC CƯỜNG. Các số liệu và kết quả nghiên cứu trong đề tài là trung thực tự nghiên cứu và hoàn toàn không sao chép hoặc sử dụng kết quả của đề tài nghiên cứu nào tương tự.

Những phần sử dụng tài liệu tham khảo trong đề tài đã được trích dẫn đầy đủ.

Nếu phát hiện có sự sao chép kết quả nghiên cứu của đề tài khác, tôi xin chịu hoàn toàn trách nhiệm và chịu kỷ luật của Khoa và Nhà Trường đề ra.

Khánh Hòa, ngày 12 tháng 06 năm 2023

Sinh viên thực hiện

TRƯƠNG TẤN NGHĨA

LỜI CẢM ƠN

Lời đầu tiên, em muốn bày tỏ sự biết ơn chân thành đến Khoa Công nghệ Thông tin - Trường Đại học Nha Trang vì đã giúp đỡ em trong quá trình hoàn thành đề tài tốt nghiệp. Em cũng muốn gửi lời cảm ơn sâu sắc nhất đến thầy Nguyễn Khắc Cường vì đã tận tâm và nhiệt tình hướng dẫn và hỗ trợ em trong quá trình nghiên cứu và làm bài báo cáo tốt nghiệp.

Trong quá trình làm đề tài và báo cáo tốt nghiệp, em biết rằng có thể không tránh khỏi sai sót do kiến thức và kinh nghiệm của em còn hạn chế. Vì vậy, em rất mong các thầy, cô thông cảm và bỏ qua những thiếu sót này. Em rất mong nhận được ý kiến đóng góp từ quý thầy, cô để em có thể học hỏi thêm kinh nghiệm và kỹ năng cần thiết.

Em xin chân thành cảm ơn!

TÓM TẮT ĐỒ ÁN

Đề tài "**SỬ DỤNG THỊ GIÁC MÁY TÍNH ĐỂ PHÁT HIỆN ĐỐI TƯỢNG NGƯỜI ĐI BỘ TRONG NHÀ VÀ CẢNH BÁO XÂM NHẬP**" phát triển một ứng dụng có phát hiện đối tượng NGƯỜI ĐI BỘ TRONG NHÀ VÀ CẢNH BÁO XÂM NHẬP, video sử dụng mô hình Yolov4 để phát hiện đối tượng.

Rất mong nhận được sự thông cảm và đóng góp ý kiến từ Quý Thầy/Cô và các bạn.

MỤC LỤC

DANH MỤC CÁC HÌNH

MỞ ĐẦU

1. Lý do chọn đề tài:

Phát hiện đối tượng người đi bộ là một chủ đề rất được rất nhiều người quan tâm trong lĩnh vực thị giác máy tính và trí tuệ nhân tạo. Các ứng dụng của việc phát hiện đối tượng trong ảnh rất đa dạng và có sự ứng dụng rộng rãi trong cuộc sống hàng ngày, từ công nghiệp đến y tế và giải trí. Đặc biệt trong giám sát an ninh, phát hiện đối tượng trong ảnh được sử dụng rộng rãi để nhận diện và phát hiện các hành vi bất thường, giúp nâng cao hiệu quả giám sát an ninh.

Với đề tài này, chuyên đề sử dụng thị giác máy tính để phát hiện đối tượng **"NGƯỜI ĐI BỘ TRONG NHÀ VÀ ĐƯA RA CẢNH BÁO XÂM NHẬP"**, cụ thể là mô hình phát hiện đối tượng YOLOv4 để phát đối tượng người đi bộ và sẽ đưa ra cảnh báo xâm nhập khi đối tượng bước vào vùng cảnh báo. Cụ thể, mô hình sẽ phát hiện đối tượng người đi bộ đi vào vùng cấm được vẽ trên khung hình video và đưa ra cảnh báo trên màn hình cùng với gửi cảnh báo qua tin nhắn trên telegram.

2. Mục tiêu nghiên cứu:

- Trình bày được phương pháp phát hiện đối tượng của họ thuật toán YOLO, chi tiết về mô hình phát hiện đối tượng YOLOv4.
- Xây dựng ứng dụng phát hiện đối tượng "**NGƯỜI ĐI BỘ TRONG NHÀ VÀ CẢNH BÁO XÂM NHẬP"** với các chức năng: vẽ vùng cần cảnh báo trong video, phát hiện đối tượng đi vào vùng cảnh báo, gửi cảnh báo qua telegram, lưu lại video đã phát hiện.

3. Cấu trúc của báo cáo bao gồm:

Chương 1: Giới thiệu

Chương 2: Cơ sở lý thuyết

Chương 3: Cài đặt chương trình và kết quả

Chương 4: Kết luận

Chương 1. GIỚI THIỆU

1.1 Tổng quan nghiên cứu

Trong những năm gần đây, với sự phát triển của kinh tế xã hội việc phát hiện người đi bộ đóng một vai trò quan trọng trong các lĩnh vực giám sát khác nhau, chẳng hạn như lĩnh vực hệ thống giám sát an ninh, quản lý giao thông, xe tự động và an toàn người đi bộ. Qua các camera thu thập các hình ảnh khác nhau và phát hiện người đi bộ trong hình ảnh, an toàn của việc lái xe không người lái có thể được cải thiện [1]. Ví dụ, trong những nơi có lưu lượng giao thông lớn như trung tâm mua sắm và quảng trường hoặc những nơi chú trong việc an ninh như bảo tàng hoặc nơi cư trú, việc phát hiện người đi bộ đóng vai trò quan trọng trong việc phát hiện giao thông và giám sát an toàn an ninh.

Phát hiện mục tiêu có thể được chia thành các phương pháp phát hiện truyền thống và các phương pháp hiện đang thịnh hành dựa trên học sâu. Các phương pháp phát hiện truyền thống cần trích xuất đặc trưng nhân tạo và học máy để thực hiện phát hiện mục tiêu. Do ảnh hưởng của kiến thức trước đó bởi việc trích xuất đặc trưng nhân tạo, các phương pháp phát hiện mục tiêu truyền thống có độ ổn định kém và tốc độ phát hiện chậm [2]. So với các phương pháp truyền thống, phương pháp phát hiện mục tiêu dựa trên học sâu có hiệu suất phát hiện tốt hơn. Hiện nay, hai thuật toán phát hiện mục tiêu chính là thuật toán Two-stage và thuật toán One-stage, cả hai đều có nhiều ưu điểm khác nhau nhưng cũng đối mặt với các khó khăn khác nhau như che khuất và sự nhầm lẫn giữa người và hình chân dung. Thuật toán Two-stage là một phương pháp dựa trên các khu vực ứng cử viên. Trước tiên, cần chọn các khu vực ứng cử viên có thể chứa mục tiêu được đo, sau đó nhận dạng và điều chỉnh tiếp các khu vực ứng cử viên và thu được kết quả cuối cùng. Thuật toán đại diện chính là R-CNN [3], Fast R-CNN [4], Faster RCNN [5], và như vậy. Thuật toán One-stage kết hợp khung ứng cử viên và phân loại vào một giai đoạn, toàn bộ giai đoạn được chuyển thành một vấn đề hồi quy, từ đó cải thiện tốc độ phát hiện mục tiêu. Thuật toán đại diện chính là YOLO [6], SSD [7], YOLOv2 [8], YOLOv3 [9], YOLOv4 [10], Retina Net [11], và như vậy.

Để thực hiện phát hiện người đi bộ một cách tốt hơn, nhiều người đã cải thiện và tối ưu hóa các thuật toán trong loạt YOLO. Zhang Mengge và đồng nghiệp [12] đề xuất mô hình MobileNet-Yolov3 và áp dụng thuật toán suy luận thích ứng video dựa

trên phương pháp khác biệt ba khung hình và bộ lọc hạt, từ đó cải thiện đáng kể tỷ lệ phát hiện video. Hao Xuzheng và đồng nghiệp [13] đề xuất một phương pháp phát hiện người đi bộ dựa trên mạng còn sót dư độ sâu, giúp mạng mô hình tốt hơn đối với các đặc trưng người đi bộ. Shi Ruijiao và đồng nghiệp [14] thiết kế một nhánh giảm tỷ lệ bốn lần để giải quyết vấn đề mất đặc trưng người đi bộ sau nhiều lần giảm tỷ lệ. Cơ chế chú ý không gian kênh được giới thiệu trong giai đoạn hợp nhất đặc trưng để giảm nhiễu nền. Đồng thời, hàm mất mát CIoU được giới thiệu để giải quyết vấn đề không tối ưu hóa nhất quán của hàm mất mát bình phương trung bình. Li Xiaoyan và đồng nghiệp [15] đưa ra thuật toán phát hiện G-MBNet, kết hợp CSPNet và ResNet18 để xây dựng mạng trích xuất đặc trưng nhẹ. Mô-đun chú ý kênh ECA được giới thiệu để cải thiện sự chú ý của mạng đối với các đặc trưng quan trọng. Cuối cùng, các tham số và kích thước của mạng trích xuất đặc trưng MBNet được giảm bằng cách kết hợp mô-đun tích chập Ghost. Để giải quyết vấn đề độ chính xác thấp của phát hiện người đi bộ trong cảnh quan phức tạp, Kang Shuai và đồng nghiệp [16] giới thiệu tích chập lỗ lai pha trộn vào mạng gốc YOLOv4, và cấu trúc tích chập lỗ có hình dạng răng cưa không gian được đề xuất để thay thế cấu trúc hồ bơi hình kim cương không gian, như vậy sẽ cho hiệu quả tốt hơn so với mạng YOLOv4 gốc.

Kết quả của nghiên cứu này sẽ đóng góp cho việc phát triển công nghệ phát hiện người đi bộ và ứng dụng của nó trong các tình huống thực tế. Hệ thống được phát triển có thể được sử dụng trong các lĩnh vực khác nhau, bao gồm hệ thống giám sát an ninh, quản lý giao thông, xe tự động và an toàn người đi bộ. Khả năng phát hiện chính xác người đi bộ trong thời gian thực sẽ nâng cao biện pháp an toàn, cho phép ra quyết định tích cực và cải thiện nhận thức tình huống tổng thể trong môi trường đô thị hiện đại.

Trong các phần tiếp theo, em sẽ trình bày một bài viết tổng quan về các tài liệu liên quan, thảo luận về phương pháp được áp dụng trong nghiên cứu này, trình bày kết quả thực nghiệm và cung cấp một phân tích chi tiết về các kết quả thu được. Tại cuối nghiên cứu này, em sẽ cố gắng đưa ra những đóng góp những hiểu biết và đề xuất có giá trị cho việc phát triển các hệ thống phát hiện người đi bộ hiệu quả và chính xác.

1.2 Thị giác máy tính là gì

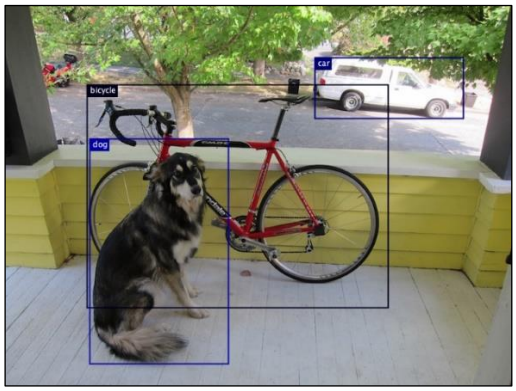
Thị giác máy tính là một lĩnh vực quan trọng trong trí tuệ nhân tạo, bao gồm các phương pháp thu thập, xử lý và phân tích hình ảnh kỹ thuật số, phát hiện đối tượng, tạo hình ảnh, siêu phân giải hình ảnh và nhiều hơn nữa. Trong các nhiệm vụ của thị giác máy tính, phát hiện đối tượng được coi là nhiệm vụ nhiều ứng dụng nhất.

Tuy nhiên, người mới bắt đầu có thể gặp khó khăn trong việc phân biệt các nhiệm vụ khác nhau của thị giác máy tính như phân loại hình ảnh, định vị đối tượng…Có thể hiểu đơn giản:

- Phân loại hình ảnh là việc gán nhãn cho hình ảnh, định vị đối tượng liên quan đến việc vẽ một hộp giới hạn xung quanh một hoặc nhiều đối tượng trong hình ảnh để khoanh vùng đối tượng.

- Phát hiện đối tượng là nhiệm vụ khó khăn hơn và là sự kết hợp của cả hai nhiệm vụ trên: vẽ một bounding box xung quanh từng đối tượng NGƯỜI ĐI BỘ TRONG NHÀ và gán cho chúng một nhãn.

Tất cả các vấn đề này được gọi chung là phát hiện đối tượng - object recognition hoặc **object detection**. Điều này có thể dẫn đến nhầm lẫn, đặc biệt là khi các nhiệm vụ này có liên quan với nhau.



Hình 1.1. Minh họa phát hiện đối tượng trong ảnh

Bài toán phát hiện đối tượng là khả năng của các hệ thống máy tính và phần mềm nhằm xác định vị trí các đối tượng trong hình ảnh và xác định từng đối tượng. Phát hiện đối tượng được áp dụng rộng rãi trong nhiều lĩnh vực như nhận diện khuôn mặt, phát hiện phương tiện, đếm người đi bộ, hệ thống an ninh và xe tự lái, và nhiều lĩnh vực khác….

1.3 Phát Biểu Bài Toán

Sử dụng mô hình YOLOv4 để phát hiện đối tượng người đi vào vùng cấm được vẽ trên khung hình video và đưa ra cảnh báo trên màn hình và gửi tin nhắn cảnh báo đến telegram của người sử dụng. Mô hình YOLOv4 được đào tạo trên tập dữ liệu COCO với 330 nghìn ảnh, 1.5 triệu đối tượng của 80 lớp. Trong đó với đối tượng người được sử dụng trong bài toán, tập dữ liệu có 250 nghìn ảnh. Tỷ lệ giữa 2 tập đào tạo (train set) và kiểm tra (test set) là 80%-20%.

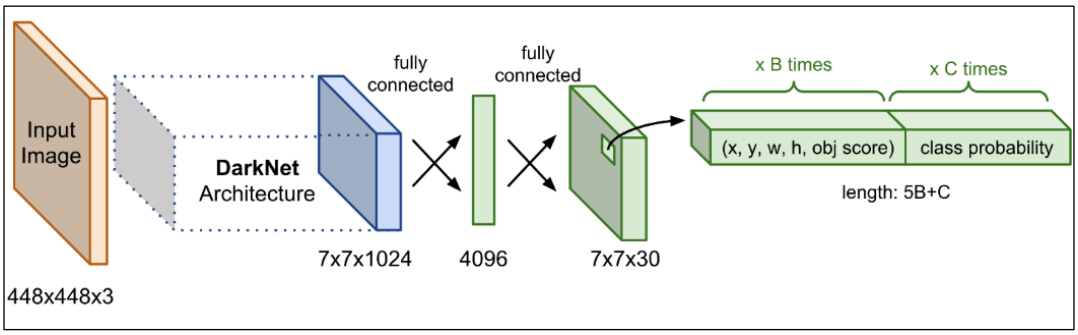
Chương 2. CƠ SỞ LÝ THUYẾT

2.1 Mạng YOLO-You only look once

2.1.1 Kiến trúc

YOLO trong object detection có nghĩa là "You only look once". Tức là chúng ta chỉ cần nhìn 1 lần là có thể phát hiện ra vật thể. Kiến trúc YOLO bao gồm 3 phần:

- Backbone: các mạng convolution làm nhiệm vụ trích xuất đặc trưng.
- Neck: những Extra Layers được áp dụng để phát hiện vật thể trên feature map của backbone.
- Head: bao gồm output layer, dự đoán nhãn và bounding box cho đối tượng.



Hình 2.1 Kiến trúc tổng quan YOLO

Các kiến trúc YOLO cũng khá đa dạng và có thể tùy biến thành các version cho nhiều input shape khác nhau.

Layers	YOLOv3	YOLOv4	YOLOv5
Neural Network Type	FCNN	FCNN	FCNN
Backbone	**Darknet53**	**CSPDarknet53** (CSPNet in Darknet)	**CSPDarkent53** Focus structure
Neck	**FPN** (Feature Pyramid Network)	**SPP** (Spatial Pyramid Pooling) and **PANet** (Path Aggregation Network)	**PANet**
Head	**B x (5 + C)** output layer **B**: No. of bounding boxes **C**: Class score	Same as Yolo v3	Same as Yolo v3
Loss Function	Binary Cross Entropy	Binary Cross Entropy	Binary Cross Entropy and Logit Loss Function

Hình 2.2 Các kiến trúc từ Yolo v3, Yolo v4, Yolo v5

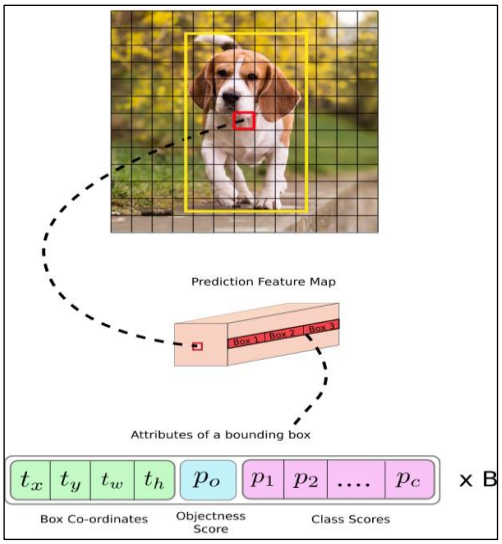
2.1.2 Output của YOLO

Output của mô hình YOLO là một véc tơ sẽ bao gồm các thành phần:

$$y^T = [p_0, \underbrace{\langle t_x, t_y, t_w, t_h \rangle}_{\text{bounding box}}, \underbrace{\langle p_1, p_2, \ldots, p_c \rangle}_{\text{scores of c classes}}]$$

- Po là xác suất dự báo vật thể xuất hiện trong bounding box
- (tx, ty, tw, th) giúp xác định bounding box. Trong đó tx, ty là tọa độ tâm và tw, th là kích thước rộng, dài của bounding box.
- (p1, p2, …, pc) là véc tơ phân phối xác suất dự báo của các classes

Như vậy output sẽ được xác định theo số lượng classes theo công thức (n_class +5). Nếu huấn luyện 80 classes thì ta sẽ có output là 85.



Hình 2.3 Kiến trúc output của YOLO

2.1.3 Dự báo trên nhiều feature map

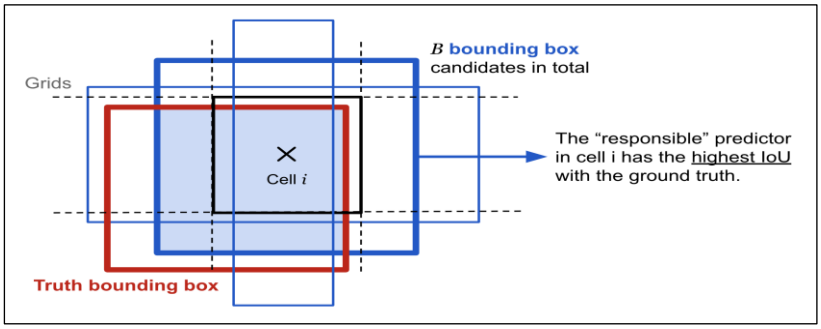
YOLO sử dụng nhiều feature map để dự báo. Các feature map ban đầu có kích thước nhỏ, giúp dự báo các đối tượng lớn. Các feature map sau có kích thước lớn hơn nhưng anchor box vẫn giữ nguyên kích thước, giúp dự báo các đối tượng nhỏ hơn.

2.1.4 Anchor box

Để xác định bounding box cho vật thể, YOLO sử dụng các anchor box để ước lượng. Các anchor box này được xác định trước và bao quanh vật thể một cách tương đối chính xác. Sau đó, thuật toán regression bounding box sẽ điều chỉnh lại các anchor box để tạo ra bounding box dự đoán cho vật thể. Trong một mô hình YOLO:

- Mỗi vật thể trong ảnh huấn luyện sẽ được phân bố vào một anchor box hoặc một cell trên feature map chứa điểm midpoint của vật thể.

- Nếu có nhiều hơn hai anchor boxes bao quanh vật thể, anchor box có IoU với ground truth bounding box cao nhất sẽ được sử dụng.

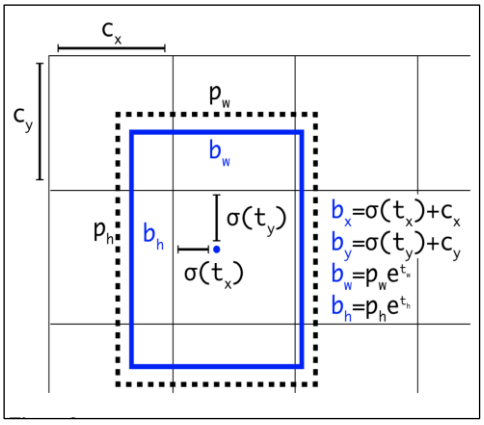


Hình 2.4 Xác định bounding box cho một vật thể từ các anchor box

- Các anchor boxes này đều giao nhau với bounding box của vật thể, tuy nhiên chỉ anchor box có đường viền dày nhất màu xanh được chọn làm anchor box cho vật thể vì nó có IoU so với ground truth bounding box cao nhất.

2.1.5 Dự đoán bounding box

Trong một mô hình, cho trước một anchor box có kích thước (pw, ph) tại cell nằm trên feature map với góc trên cùng bên trái của nó là (cx, cy), mô hình sẽ dự đoán 4 tham số (tx, ty, tw, th). Hai tham số đầu tiên là độ lệch (offset) so với góc trên cùng bên trái của cell và hai tham số sau là tỷ lệ so với anchor box. Các tham số này sẽ giúp xác định bounding box dự đoán b, bao gồm tâm (bx, by) và kích thước (bw, bh), thông qua việc sử dụng hàm sigmoid và hàm exponential như các công thức dưới đây:

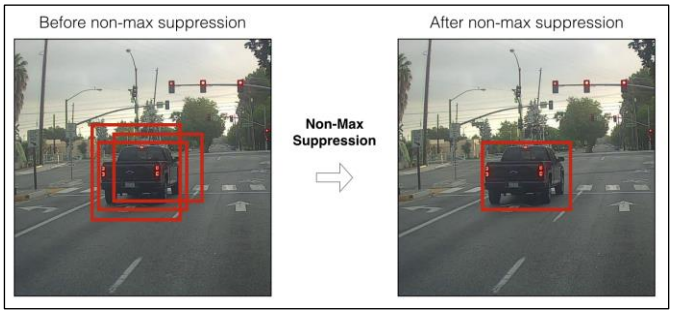


Hình 2.5 Công thức ước lượng bounding box từ anchor box

Các tọa độ trong mô hình đã được điều chỉnh theo chiều rộng và chiều cao của ảnh, vì vậy giá trị của chúng luôn nằm trong khoảng [0, 1]. Việc sử dụng hàm sigmoid giúp giới hạn các tọa độ trong phạm vi này và không vượt quá ngưỡng.

2.1.6 Non-max suppression

Vì YOLO dự báo rất nhiều bounding box trên một bức ảnh, các cell gần nhau có khả năng giao nhau rất cao, dẫn đến việc các khung hình overlap. Trong trường hợp này, để giảm số lượng khung hình được tạo ra đáng kể, thuật toán YOLO sử dụng phương pháp non-max suppression.



Hình 2.6 Sử dụng non-max suppression giảm bounding box bị overlap

Phương pháp non-max suppression bao gồm các bước sau:

1. Đầu tiên, cần phải tìm cách để loại bỏ các bounding box có xác suất chứa vật thể nhỏ hơn một ngưỡng threshold (thường là 0.5) để giảm số lượng bounding box.

2. Đối với các bounding-box giao nhau, non-max suppression sẽ chọn bounding box có xác suất chứa vật thể lớn nhất và tính chỉ số giao thoa IoU với các bounding box còn lại.

3. Nếu chỉ số IoU này lớn hơn ngưỡng threshold, nghĩa là hai bounding box overlap nhau rất cao, ta sẽ xóa các bounding box có xác suất thấp hơn và giữ lại bounding box có xác suất cao nhất. Cuối cùng, một bounding box duy nhất được giữ lại cho mỗi vật thể.

2.1.7 Hàm mất mát

Hàm loss function của YOLO được chia thành hai phần: phần đo lường sai số của bounding box (localization loss) và phần đo lường sai số của phân phối xác suất các classes (confidence loss).

$$\mathcal{L}_{\text{loc}} = \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2]$$

$$\mathcal{L}_{\text{cls}} = \underbrace{\sum_{i=0}^{S^2} \sum_{j=0}^{B} (\mathbb{1}_{ij}^{\text{obj}} + \lambda_{\text{noobj}}(1 - \mathbb{1}_{ij}^{\text{obj}}))(C_{ij} - \hat{C}_{ij})^2}_{\text{cell contain object}} + \underbrace{\sum_{i=0}^{S^2} \sum_{c \in \mathcal{C}} \mathbb{1}_{i}^{\text{obj}} (p_i(c) - \hat{p}_i(c))^2}_{\text{probability distribution classes}}$$

$$\mathcal{L} = \mathcal{L}_{\text{loc}} + \mathcal{L}_{\text{cls}}$$

Hàm mất mát của YOLO bao gồm hai phần:

- Lloc: đo lường sai số của bounding box dự đoán so với bounding box thực tế.
- Lcls: đo lường sai số của phân phối xác suất. Phần tổng đầu tiên tính mất mát của việc dự đoán có vật thể trong cell hay không, và phần tổng thứ hai tính mất mát của phân phối xác suất nếu cell chứa vật thể.

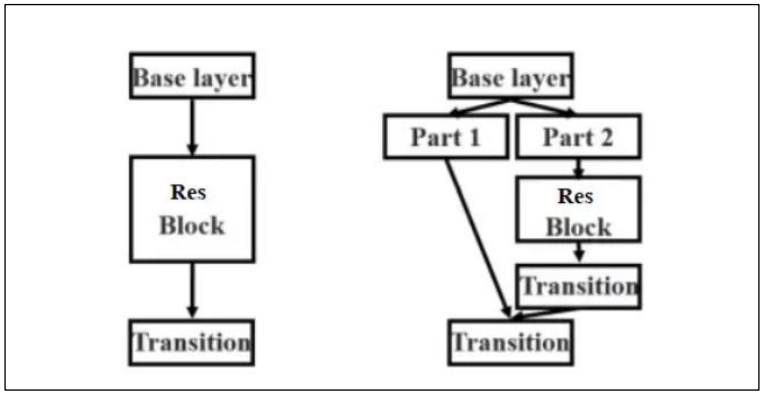
Để điều chỉnh hàm mất mát, ta sử dụng hệ số điều chỉnh λ để phạt loss function trong trường hợp dự đoán sai bounding box. Ngoài ra, có thể giảm nhẹ hàm mất mát trong trường hợp cell không chứa vật thể bằng cách sử dụng hệ số điều chỉnh.

2.2 YOLOv4: Optimal Speed and Accuracy of Object Detection

2.2.1 Backbone

2.2.1.1 CSP Block

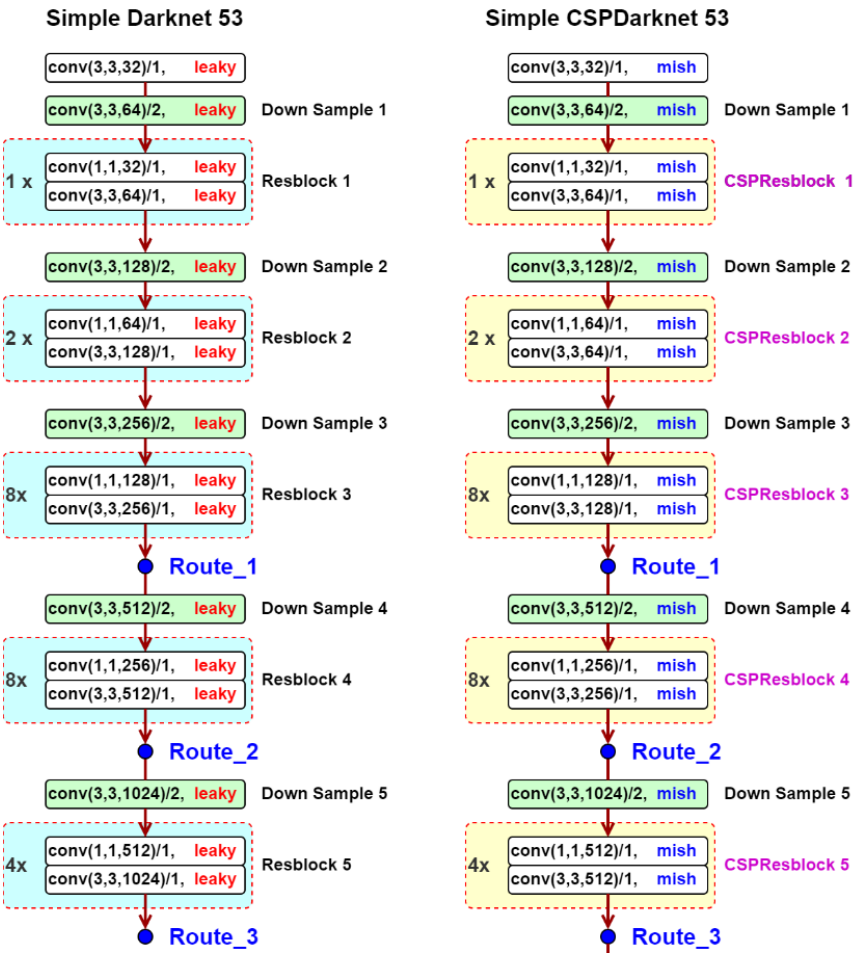
CSPBlock được áp dụng vào Residual Block với ý tưởng chính được thể hiện như Hình 8. Không giống như Residual Block chỉ có một đường đi từ đầu đến cuối, CSPBlock chia thành 2 đường đi. Việc chia thành 2 đường như vậy giúp loại bỏ việc tính toán lại gradient (đạo hàm), từ đó tăng tốc độ trong quá trình huấn luyện. Hơn nữa, việc tách thành 2 nhánh, mỗi nhánh lấy một phần từ feature map trước đó, giảm số lượng tham số đáng kể, giúp tăng tốc trong cả quá trình inference và huấn luyện.



Hình 2.7 Sự khác nhau giữa Residual Block và CSPResBlock

2.2.1.2 CSPDarkNet53

Trong YOLOv4, CSPBlock được áp dụng bằng cách thay thế Residual Block thông thường của YOLOv3 bằng CSPResBlock và thay đổi activation function từ LeakyReLU thành Mish, tạo thành CSPDarkNet53.



Hình 2.8 So sánh DarkNet53 với CSPDarkNet53

2.2.1.3. DropBlock

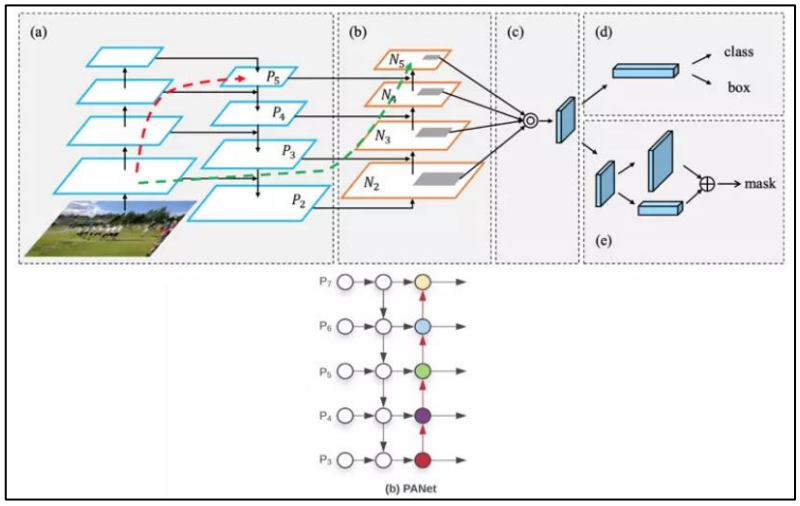
Trong bài toán Classification, ta thường sử dụng DropOut ở lớp cuối để giảm hiện tượng Overfitting. Tuy nhiên, việc loại bỏ ngẫu nhiên một số vị trí trong feature map trong Convolution không phải là phương pháp hiệu quả. Do các vị trí liền kề trong feature map có tương quan cao với nhau, vì vậy việc loại bỏ ngẫu nhiên các vị trí như vậy sẽ không đem lại nhiều hiệu quả. Thay vào đó, phương pháp DropBlock sẽ loại bỏ nhóm các vị trí trong feature map thay vì chỉ loại bỏ một vị trí.

2.2.2 Neck

YOLOv4 có thêm neck để thực hiện phát hiện vật thể ở trên những scale khác nhau. YOLOv4 sử dụng 2 thành phần cho neck: SPP và PAN.

2.2.2.1 PAN

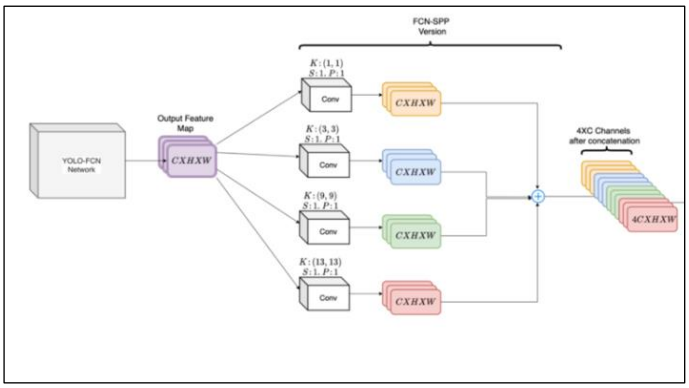
PAN là FPN được cải tiến. Trong FPN, một nhánh phụ được tạo ra để đưa thông tin từ các feature ở các layer sâu vào feature ở các layer hẹp. Tuy nhiên, FPN thiếu việc đưa feature từ các layer hẹp trở lại các layer sâu. Vì vậy, PAN có thêm một nhánh phụ, kiến trúc được mô tả cụ thể dưới đây:



Hình 2.9 Kiến trúc PAN

2.2.2.2 SPP

Với sự xuất hiện của SPP, ảnh đầu vào đã có thể có kích thước đa dạng hơn mà không cần phải giữ nguyên nữa, một vector có độ dài cố định vẫn được sinh ra khi đi qua SPP. YOLOv4 đổi kích thước của các kernel của Max Pooling, thay vì đưa ra các feature map có kích thước 4×4, 2×2, 1×1, YOLOv4 đưa ra 4 khối feature map có kích thước $H \times W$ (cùng kích thước với feature map lấy từ backbone), mỗi khối feature map $H \times W$ này được tạo ra từ Max Pooling với kernel có kích thước (1,3,9,13). Sau đó, chúng được concatenate lại với nhau, tạo thành một khối feature map $H \times W \times (4 \times C)$.



Hình 2.10 Kiến trúc SPP

Chương 3. CÀI ĐẶT VÀ KẾT QUẢ

3.1 Cài đặt thư viện

- opencv-python: cung cấp các công cụ và hàm để thực hiện các tác vụ liên quan đến xử lý ảnh như: đọc và ghi ảnh và video…
- numpy: cung cấp các hàm và công cụ để thực hiện các phép tính số học, đại số tuyến tính, xử lý ảnh và xử lý tín hiệu.
- PyQt5: cung cấp các thành phần giao diện người dùng (UI) như các nút bấm, các hộp thoại, các trình chỉnh sửa văn bản và các cửa sổ đồ họa để xây dựng các chương trình python ứng dụng phức tạp.
- shapely: cung cấp tính năng vẽ hình, các đối tượng hình học, như điểm, đường thẳng, đa giác, v.v..
- python-telegram-bot: cung cấp một API Python cho các tính năng của Telegram Bot API, cho phép người dùng tạo ra các bot Telegram với các tính năng tùy chỉnh khác.
- telegram: được sử dụng để tương tác với ứng dụng Telegram thông qua API của nó. Ngoài ra, cung cấp các lớp và phương thức cho phép người dùng tạo, cấu hình và điều khiển các bot Telegram.
- asyncio: sử dụng để viết các ứng dụng bất đồng bộ (asynchronous) trong Python.
- datetime: cung cấp các lớp và phương thức để làm việc với dữ liệu ngày tháng.

3.2 Mô hình và dataset

3.2.1 Đào tạo mô hình

Để huấn luyện mô hình YOLOv4 để phát hiện người:

1. Chuẩn bị dữ liệu: Tập dữ liệu ảnh phải được gán nhãn đúng để mô hình có thể học được các đặc trưng của người. Có thể sử dụng các công cụ như LabelImg để gán nhãn.

2. Chuẩn bị tệp cấu hình: tải file yolov4-custom.cfg. Mở ra và và chỉnh sửa:

   -Tìm đến dòng 20, sửa max_batches=6000.

   -Đến dòng 22 sửa thành steps=80%, 90% của max_batches. Ví dụ ở đây là steps=4800,5400.

   -Thay thế toàn bộ các dòng có "classes=80" thành "classes=1.

   -Thay thế e toàn bộ các dòng có "filters=255" thành "filters=18".

   -Lưu lại là đã có thêm file yolov4-custom.cfg.

3. Chuẩn bị Makefile:

   Tải tiếp Makefile , mở ra và sửa như sau:

   Dòng 1, sửa thành GPU=1

   Dòng 2, sửa thành CUDNN=1

   Dòng 4, sửa thành OPENCV=1

   Save lại và có thêm file Makefile.

4. Tiến hành đào tạo mô hình với dữ liệu:

- Upload các file đã chuẩn bị, dữ liệu cùng source code mô hình YOLOv4 lên google drive, chúng ta sẽ sử dụng google colab để train với GPU.
- Tạo file yolo.names chứa tên các class:

```
%cd /content/gdrive/My\ Drive/darknet
!echo "person" > yolo.names
```

- Tạo hai file train.txt và val.txt chứa danh sách các file ảnh:

File train.txt chứa danh sách các file sẽ dùng để train

File val.txt thì tất nhiên là danh sách các file dùng để val

Danh sách cần chọn ngẫu nhiên đảm bảo tính phân phối dữ liệu.

- Tạo code block và chạy code để đọc dữ liệu

```
%cd /content/gdrive/My\ Drive/darknet

import glob2
import math
import os
import numpy as np

files = []
for ext in ["*.png", "*.jpeg", "*.jpg"]:
  image_files = glob2.glob(os.path.join("data/data/", ext))
  files += image_files

nb_val = math.floor(len(files)*0.2)
rand_idx = np.random.randint(0, len(files), nb_val)

# Tạo file train.txt
with open("train.txt", "w") as f:
  for idx in np.arange(len(files)):
    if (os.path.exists(files[idx][:-3] + "txt")):
      f.write(files[idx]+'\n')

# Tạo file vali.txt
with open("val.txt", "w") as f:
  for idx in np.arange(len(files)):
    if (idx in rand_idx) and (os.path.exists(files[idx][:-3] + "txt")):
      f.write(files[idx]+'\n')
```

5. Tạo file yolo.data chứa tham số đào tạo

   Tạo file yolo.data chứa tham số đào tạo để YOLOv4 đọc:

```
%cd /content/gdrive/My\ Drive/darknet
!mkdir backup
!echo classes=1 > yolo.data
!echo train=train.txt >> yolo.data
!echo valid=val.txt >> yolo.data
!echo names=yolo.names >> yolo.data
!echo backup=backup >> yolo.data
```

6. Biên dịch mã nguồn Darknet:

```
%cd /content/gdrive/My\ Drive/darknet
!rm darknet
!make
```

7. Tải pretrain weights:

```
%cd /content/gdrive/My\ Drive/darknet
!wget https://github.com/AlexeyAB/darknet/releases/download/darknet_yolo_v3_optimal/yolov4.conv.137
```

8. Đào tạo mô hình:

```
[ ]  %cd /content/gdrive/My\ Drive/darknet
     !./darknet detector train yolo.data cfg/yolov4-custom.cfg yolov4.conv.137 -dont_show
```

```
 1249: 2.061428, 2.191828 avg loss, 0.001000 rate, 23.764414 seconds, 79936 images, 25.588937 hours left
Loaded: 0.000039 seconds
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 139 Avg (IOU: 0.474586, GIOU: 0.434440), Class: 0.998237,
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 150 Avg (IOU: 0.757588, GIOU: 0.743850), Class: 0.981158,
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 161 Avg (IOU: 0.699273, GIOU: 0.693070), Class: 0.995428,
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 139 Avg (IOU: 0.000000, GIOU: 0.000000), Class: 0.000000,
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 150 Avg (IOU: 0.620572, GIOU: 0.580916), Class: 0.990169,
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 161 Avg (IOU: 0.660722, GIOU: 0.653763), Class: 0.992664,
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 139 Avg (IOU: 0.000000, GIOU: 0.000000), Class: 0.000000,
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 150 Avg (IOU: 0.566078, GIOU: 0.504852), Class: 0.997177,
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 161 Avg (IOU: 0.553811, GIOU: 0.495572), Class: 0.997952,
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 139 Avg (IOU: 0.847224, GIOU: 0.844552), Class: 0.988753,
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 150 Avg (IOU: 0.760316, GIOU: 0.758248), Class: 0.982622,
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 161 Avg (IOU: 0.709589, GIOU: 0.688823), Class: 0.994569,
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 139 Avg (IOU: 0.000000, GIOU: 0.000000), Class: 0.000000,
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 150 Avg (IOU: 0.000000, GIOU: 0.000000), Class: 0.000000,
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 161 Avg (IOU: 0.793285, GIOU: 0.787514), Class: 0.999252,
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 139 Avg (IOU: 0.561538, GIOU: 0.545014), Class: 0.998680,
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 150 Avg (IOU: 0.714267, GIOU: 0.701930), Class: 0.996065,
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 161 Avg (IOU: 0.793969, GIOU: 0.789605), Class: 0.998310,
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 139 Avg (IOU: 0.648545, GIOU: 0.601389), Class: 0.993009,
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 150 Avg (IOU: 0.723550, GIOU: 0.707183), Class: 0.992693,
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 161 Avg (IOU: 0.670116, GIOU: 0.641769), Class: 0.997253,
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 139 Avg (IOU: 0.000000, GIOU: 0.000000), Class: 0.000000,
v3 (iou loss, Normalizer: (iou: 0.07, cls: 1.00) Region 150 Avg (IOU: 0.876230, GIOU: 0.876230), Class: 0.995388,
```

3.2.2 Kết quả đào tạo mô hình

Với đối tượng người được sử dụng trong bài toán, tập dữ liệu có 250 nghìn ảnh. Tỷ lệ giữa 2 tập đào tạo (train set) và kiểm tra (test set) là 80%-20%.

Độ chính xác trên tập kiểm tra của mô hình:

Accuracy	Precision	Recall	F1-score
0.91	0.93	0.9	0.915

Ý nghĩa các thông số:

Accuracy: độ chính xác của một mô hình. Nó được định nghĩa là tỉ lệ giữa số lần dự đoán đúng và tổng số trường hợp được dự đoán.

- Precision (độ chính xác): tỉ lệ số lượng dự đoán đúng positive (TP - true positive) trên tổng số dự đoán positive (TP + FP - false positive). Nói cách khác, precision là khả năng của mô hình dự đoán chính xác các mẫu positive.
- Recall (độ phủ): tỉ lệ số lượng dự đoán đúng positive (TP) trên tổng số mẫu positive có thực (TP + FN - false negative). Recall đo lường khả năng của mô hình phát hiện được tất cả các mẫu positive trong tập dữ liệu.
- F1-score là trung bình điều hòa giữa precision và recall. Nó là một số đo kết hợp cả precision và recall để đưa ra một số liệu đánh giá tổng thể về hiệu suất của mô hình phân loại. F1-score được tính bằng công thức: F1-score = 2 * (precision * recall) / (precision + recall).

Ngoài ra mô hình còn được đánh giá với thông số khá theo các trường hợp:

Size ảnh đầu vào (width-height)	**mAP@0.5**	**AP@0.5:0.95**	**(R)FPS-(V)FPS**	**BFlops**
(608,608)	66.7%	43.5%	34-62	128.5
(512,512)	64.9%	43%	44-83	91.1
(419,419)	62.8%	41.2%	55-96	60.1
(320,320)	60%	38%	63-123	35.

Ý nghĩa các thông số trên:

- mAP@0.5 là trung bình của các giá trị Precision (độ chính xác) ở các mức ngưỡng khác nhau của IoU, từ 0 đến 1, với bước nhảy là 0.05. Tức là, mAP@0.5 tính toán trung bình của các giá trị Precision tại các mức ngưỡng IoU từ 0 đến 1, với bước nhảy là 0.05, và lấy giá trị Precision tại ngưỡng IoU bằng 0.5 để tính toán.
- AP@0.5:0.95 tính toán Precision (độ chính xác) ở các ngưỡng IoU từ 0.5 đến 0.95 với bước nhảy là 0.05, sau đó tính toán trung bình của các Precision đó. Chỉ số này có thể cung cấp một cái nhìn tổng thể hơn về hiệu suất của hệ thống phát hiện đối tượng, và phản ánh được khả năng phát hiện đối tượng với độ chính xác khác nhau.
- (R)FPS là viết tắt của "refresh rate" (tốc độ làm mới) và được đo bằng hertz (Hz). Đây là số lần màn hình của bạn cập nhật hình ảnh trên màn hình của bạn trong một

giây. Ví dụ, màn hình với tốc độ làm mới 60Hz sẽ cập nhật hình ảnh trên màn hình 60 lần trong một giây.

- (V)FPS là viết tắt của "frames per second" (khung hình trên giây) và là số lượng khung hình được hiển thị trên màn hình trong một giây. Đây là số lượng khung hình mà trò chơi hoặc ứng dụng đang xử lý và hiển thị trên màn hình của bạn trong một giây. Ví dụ, trò chơicó (V)FPS là 60 sẽ hiển thị 60 khung hình trên màn hình của bạn trong một giây.
- BFlops là viết tắt của "billion floating-point operations per second" (tỷ phép tính dấu chấm động trên giây) và là một đơn vị đo hiệu năng của các hệ thống xử lý. Nó được sử dụng để đo lường khả năng xử lý tín hiệu số và tính toán của mô hình.

3.3 Code chính

3.3.1 Cài đặt mô hình phát hiện với YOLOv4

- Viết file yolodetect.py với class YoloDetect() phát hiện đối tượng người xâm nhập trong hình ảnh.

  Bước 1: Viết các phương thức để khởi tạo lớp và load mô hình:

  Phương thức init() có ba tham: detect_class, frame_width và frame_height.

  -Tham số detect_class được thiết lập mặc định là "person", có nghĩa là thuật toán sẽ phát hiện chỉ người trong hình ảnh.

  -Tham số frame_width và frame_height xác định kích thước của hình ảnh đầu vào mà thuật toán YOLO sẽ xử lý.

  Hàm init() khởi tạo một số biến:
  - Các classnames_file, weights_file, config_file lần lượt là những biến chỉ đường dẫn tệp đến tệp tên 80 lớp đối mô hình phát hiện được, tệp trọng số của được huấn luyện trước và tệp thông số kiến trúc mô hình.
  -Các biến conf_threshold và nms_threshold xác định ngưỡng độ tin cậy tối thiểu để phát hiện đối tượng và ngưỡng khử các bounding box overlap.

  -Biến model là biến mô hình YOLOv4 được huấn luyện trước được tải bằng cách sử dụng phương thức cv2.dnn.readNet() từ thư viện OpenCV load những thông số từ các file thông qua các biến đường dẫn classnames_file, weights_file,

config_file.

- Biến classes và output_layers được khởi tạo là None và sau đó được thiết lập thành tên các lớp và các lớp đầu ra của mô hình YOLO bằng phương thức read_class_file() và get_output_layers().

- Hàm read_class_file() để đọc tệp chứa danh sách các lớp.

- Hàm get_output_layers() để lấy danh sách các lớp đầu ra của mô hình.

- Biến last_alert được sử dụng để theo dõi thời điểm cuối cùng mà thông báo được gửi, và biến alert_telegram_each xác định khoảng thời gian (tính bằng giây) giữa các thông báo khi phát hiện đối tượng được gửi đến Telegram.

```
class YoloDetect():
    def __init__(self, detect_class="person", frame_width=960, frame_height=540):
        # Parameters
        self.classnames_file = "model/coco-classes.txt"
        self.weights_file = "model/yolov4.weights"
        self.config_file = "model/yolov4.cfg"
        self.conf_threshold = 0.8
        self.nms_threshold = 0.5
        self.detect_class = detect_class
        self.frame_width = frame_width
        self.frame_height = frame_height
        self.scale = 1 / 255
        self.model = cv2.dnn.readNet(self.weights_file, self.config_file)
        self.classes = None
        self.output_layers = None
        self.read_class_file()
        self.get_output_layers()
        self.last_alert = None
        self.alert_telegram_each = 15  # seconds

    def read_class_file(self):
        with open(self.classnames_file, 'r') as f:
            self.classes = [line.strip() for line in f.readlines()]

    def get_output_layers(self):
        layer_names = self.model.getLayerNames()
        self.output_layers = [layer_names[i - 1] for i in self.model.getUnconnectedOutLayers()]
```

Bước 2: Viết các hàm vẽ khung khi phát hiện người và tính điểm trung tâm của đối tượng người, nếu điểm trung tâm đi vào vùng cấm thì đưa ra cảnh báo.

-Hàm draw_prediction() nhận tham số số là hình ảnh
(img), class_id, conf, x, y, x_plus_w, y_plus_h và points. Hàm này vẽ một hình chữ nhật xung quanh đối tượng được phát hiện trên hình ảnh (img), đặt nhãn và độ tin cậy của đối tượng được phát hiện, tính toán trung tâm của đối tượng và vẽ một hình tròn tại đó. Nếu trung tâm của đối tượng nằm trong một vùng cụ thể được định nghĩa

bởi points, hàm sẽ gọi hàm alert để cảnh báo.

-Hàm alert() nhận tham số là hình ảnh (img) và sẽ cảnh báo bằng cách thêm văn bản "ALARM !!!!" vào hình ảnh. Nếu thời gian giữa lần cảnh báo cuối cùng và lần cảnh báo hiện tại lớn hơn giá trị được xác định bởi biến alert_telegram_each, hàm sẽ tạo một luồng mới để gửi một thông báo cảnh báo đến tài khoản Telegram. Hình ảnh được lưu vào tệp "alert.png" và được thay đổi kích thước trước khi gửi. Cuối cùng, hàm sẽ trả về hình ảnh được cảnh báo.

```
def draw_prediction(self, img, class_id, conf, x, y, x_plus_w, y_plus_h, points):
    label = str(self.classes[class_id])
    conf = str(round(conf, 2))
    color = (0, 255, 0)
    cv2.rectangle(img, (x, y), (x_plus_w, y_plus_h), color, 2)
    cv2.putText(img, label, (x - 10, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)
    cv2.putText(img, conf, (x - 10, y - 25), cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)

    # Tinh toan centroid
    centroid = ((x + x_plus_w) // 2, (y + y_plus_h) // 2)
    cv2.circle(img, centroid, 5, (color), -1)

    if isInside(points, centroid):
        img = self.alert(img)

    return isInside(points, centroid)

def alert(self, img):
    cv2.putText(img, "ALARM!!!!", (10, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0, 255), 2)
    # New thread to send telegram after 15 seconds
    if (self.last_alert is None) or (
            (datetime.datetime.utcnow() - self.last_alert).total_seconds() > self.alert_telegram_each):
        self.last_alert = datetime.datetime.utcnow()
        cv2.imwrite("alert.png", cv2.resize(img, (1920,1080)))
        thread = threading.Thread(target=send_telegram)
        thread.start()
    return img
```

<u>Bước 3</u>: Viết hàm send_telegram() gửi thông báo ảnh phát hiện xâm nhập đến telegram

-Tham số photo_path được đặt mặc định là "alert.png", đây là đường dẫn của ảnh được gửi cùng với thông báo.

-Biến my_token chứa token của bot Telegram được sử dụng để gửi tin nhắn. Sau đó, chúng ta sử dụng thư viện telegram để tạo một đối tượng bot với token vừa khai báo.

- Hàm asyncio.run() để chạy một coroutine (hàm bất đồng bộ) bot.sendPhoto() gửi một tin nhắn chứa ảnh đến chat với chat_id là "1919079406" (id của người dùng)

-Tham số photo của hàm bot.sendPhoto() được truyền bằng cách mở tập tin ảnh với

đường dẫn được chỉ định trong tham số photo_path.

-Tham số caption được sử dụng để thêm một chú thích cho ảnh được gửi đi.

```
import telegram
import asyncio

def send_telegram(photo_path="alert.png"):
    try:
        my_token = "6295457754:AAEW5ljWVrZYRRpPJXzflLuYKVilrxhH9hY"
        bot = telegram.Bot(token=my_token)
        asyncio.run(bot.sendPhoto(chat_id="1919079406", photo=open(photo_path, "rb"), caption="Có xâm nhập, nguy hiểm!'
        print("Send sucess")
    except Exception as ex:
        print("Can not send message telegram ", ex)
```

Bước 4: Viết hàm detect() phát hiện đối tượng, hàm này nhận vào một khung hình frame và một danh sách các điểm points được sử dụng để vẽ các dự đoán lên khung hình. Sau đó, phương thức này thực hiện các bước sau:

1. Tạo một blob từ khung hình đầu vào sử dụng hàm cv2.dnn.blobFromImage(). Blob được sử dụng để chuẩn bị dữ liệu đầu vào cho mô hình deep learning.
2. Đưa blob vào mô hình deep learning sử dụng phương thức setInput().
3. Thực hiện việc dự đoán các đối tượng trong khung hình bằng cách chạy mô hình sử dụng phương thức forward() và lấy ra các kết quả dự đoán (outs).
4. Lọc các đối tượng được dự đoán bằng cách duyệt qua các kết quả dự đoán và chỉ chọn ra những đối tượng có độ tin cậy (confidence) lớn hơn ngưỡng (confidence threshold) và thuộc lớp cần phát hiện (detect class).
5. Áp dụng thuật toán NMS (Non-Maximum Suppression) để loại bỏ các dự đoán trùng lặp và chỉ giữ lại dự đoán có độ tin cậy cao nhất.
6. Vẽ các hình chữ nhật (bounding boxes) xung quanh các đối tượng được phát hiện bằng cách sử dụng phương thức draw_prediction(). Các hình chữ nhật này được vẽ lên khung hình frame và được xác định bởi tọa độ của điểm góc trái trên và điểm góc phải dưới của hình chữ nhật.
7. Trả về khung hình frame đã được vẽ các hình chữ nhật xung quanh các đối tượng được phát hiện.

```
detect(self, frame, points):
blob = cv2.dnn.blobFromImage(frame, self.scale, (896, 896), (0, 0, 0), True, crop=False)
self.model.setInput(blob)
outs = self.model.forward(self.output_layers)

# Loc cac object trong khung hinh
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 >= self.conf_threshold) and (self.classes[class_id] == self.detect_class):
            center_x = int(detection[0] * self.frame_width)
            center_y = int(detection[1] * self.frame_height)
            w = int(detection[2] * self.frame_width)
            h = int(detection[3] * self.frame_height)
            x = center_x - w / 2
            y = center_y - h / 2
            class_ids.append(class_id)
            confidences.append(float(confidence))
            boxes.append([x, y, w, h])

indices = cv2.dnn.NMSBoxes(boxes, confidences, self.conf_threshold, self.nms_threshold)
for i in indices:
    box = boxes[i]
    x = box[0]
    y = box[1]
    w = box[2]
    h = box[3]
    self.draw_prediction(frame, class_ids[i],confidences[i], round(x), round(y), round(x + w), round(y + h), points)

return frame
```

3.3.2 Cài đặt xử lý phát hiện đối tượng trên video

Viết file intrusion_warning.py để phát hiện đối tượng cần NGƯỜI ĐI BỘ TRONG NHÀ VÀ CẢNH BÁO XÂM NHẬP trên video.

Bước 1: Viết hàm vẽ các điểm của vùng xâm nhập trên khung hình video.
- Hàm handle_left_click() được sử dụng để xử lý sự kiện click trái chuột trên khung hình (frame) và hàm draw_polygon() được sử dụng để vẽ một đa giác (polygon) trên khung hình.

-Hàm handle_left_click() nhận vào các tham số sau:

- event: sự kiện được kích hoạt (ở đây là sự kiện click trái chuột)
- x: tọa độ x của điểm được click trên khung hình
- y: tọa độ y của điểm được click trên khung hình
- flags: các cờ thêm vào (ở đây không được sử dụng)
- points: danh sách các điểm được lưu trữ

Nếu sự kiện là click trái chuột (cv2.EVENT_LBUTTONDOWN), hàm sẽ thêm một điểm mới vào danh sách các điểm points. Điểm mới này được tạo ra bằng cách sử

dụng tọa độ x và y của điểm được click trên khung hình.

-Hàm draw_polygon() nhận vào hai tham số:

- frame: khung hình cần vẽ đa giác lên
- points: danh sách các điểm của đa giác

Hàm này sử dụng một vòng lặp để duyệt qua danh sách các điểm của đa giác và vẽ các điểm này bằng hàm cv2.circle(), với bán kính 5 và màu đỏ (0, 0, 255). Sau đó, hàm sử dụng hàm cv2.polylines() để vẽ đa giác lên khung hình. Đa giác này được xác định bằng danh sách các điểm points, và được vẽ bằng màu xanh lam (255, 0, 0) và độ dày 2 pixel.

```
6   def handle_left_click(event, x, y, flags, points):
7       if event == cv2.EVENT_LBUTTONDOWN:
8           points.append([x, y])
9
10  def draw_polygon (frame, points):
11      for point in points:
12          frame = cv2.circle( frame, (point[0], point[1]), 5, (0,0,255), -1)
13
14      frame = cv2.polylines(frame, [np.int32(points)], False, (255,0, 0), thickness=2)
15      return frame
```

Bước 2: Viết hàm intrusion_warning() xử lý các frame của video, gọi tới hàm phát hiện đối tượng qua từng frame ở phần trên. Ghi ra file video kết quả.

-Hàm nhận vào hai tham số:

- path_video: đường dẫn tới file video cần phát hiện xâm nhập
- path_save_folder: đường dẫn tới thư mục để lưu trữ video kết quả sau khi phát hiện xâm nhập

-Hàm này bao gồm các bước sau:

1. Khởi tạo danh sách các điểm được lưu trữ trong biến points.
2. Khởi tạo một mô hình phát hiện đối tượng YOLO bằng cách tạo một đối tượng YoloDetect().
3. Mở video từ đường dẫn được cung cấp sử dụng hàm cv2.VideoCapture().
4. Khởi tạo một VideoWriter để ghi lại video kết quả sau khi phát hiện xâm nhập. Thông tin về video kết quả bao gồm định dạng video (ở đây là mp4), kích thước

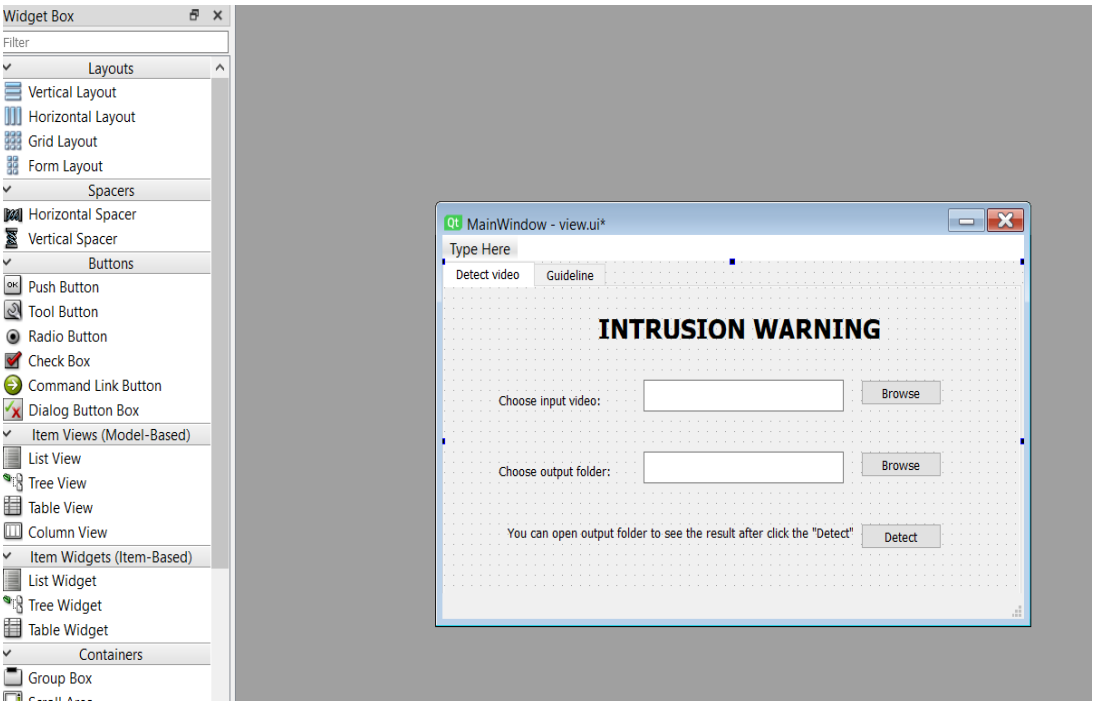
khung hình (960x540) và tốc độ khung hình (50 fps). Video kết quả được lưu trữ trong thư mục được chỉ định bởi biến path_save_folder.

5. Vòng lặp chạy qua từng khung hình của video được mở bằng hàm cap.read().
6. Trong vòng lặp, trước khi phát hiện xâm nhập, hàm sử dụng hàm cv2.resize() để thay đổi kích thước khung hình thành 960x540, sau đó vẽ đa giác (polygon) trên khung hình sử dụng hàm draw_polygon().
7. Nếu phát hiện xâm nhập (detect = True), hàm sử dụng mô hình YOLO để phát hiện các đối tượng trên khung hình sử dụng phương thức detect() của đối tượng model.
8. Hàm sử dụng hàm cv2.waitKey() để chờ đợi phản hồi từ bàntiện. Nếu phím 'q' được nhấn, vòng lặp sẽ kết thúc và video sẽ được lưu vào đường dẫn được chỉ định bởi biến path_save_folder. Nếu phím 'd' được nhấn, danh sách các điểm points được thêm một điểm cuối cùng giống với điểm đầu tiên để hoàn thành đa giác, và biến detect được thiết lập thành True để bắt đầu phát hiện xâm nhập.
9. Khung hình được ghi lại vào video kết quả bằng hàm result.write().
10. Khung hình được hiển thị lên màn hình sử dụng hàm cv2.imshow().
11. Hàm cv2.setMouseCallback() được sử dụng để gọi hàm handle_left_click() khi có sự kiện click trái chuột trên khung hình.
12. Khi vòng lặp kết thúc, video được giải phóng và cửa sổ hiển thị khung hình được đóng bằng hàm cap.release(), result.release() vàcv2.destroyAllWindows().

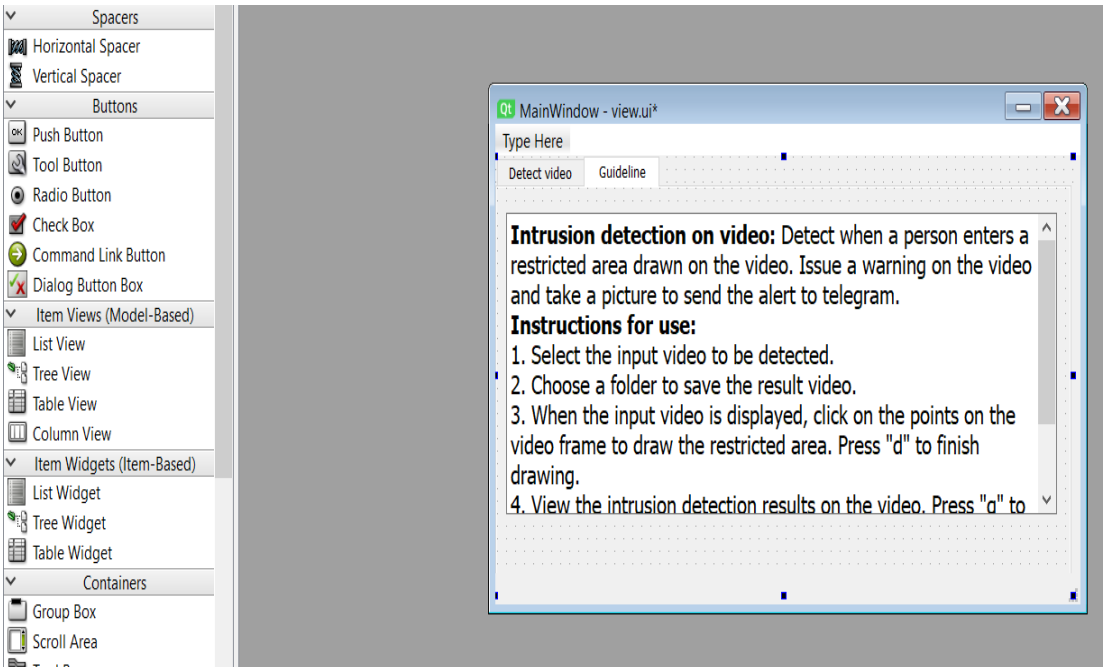
```python
def intrusion_warning(path_video, path_save_folder):
    # Chua cac diem nguoi dung chon de tao da giac
    points = []
    # new model Yolo
    model = YoloDetect()
    detect = False
    cap = cv2.VideoCapture(path_video)
    # datetime object containing current date and time
    result = cv2.VideoWriter(path_save_folder+"/"+datetime.now().strftime("%d-%m-%Y_%Hh%Mp%Ss_")+'video_result.mp4',
                             cv2.VideoWriter_fourcc(*'mp4v'),
                             50, (960,540))

    # Check if camera opened successfully
    if (cap.isOpened()== False):
        print("Error opening video stream or file")

    while (cap.isOpened()):
        ret,frame = cap.read()
        if ret == True:

            # Ve ploygon
            frame = cv2.resize(frame, (960, 540))
            frame = draw_polygon(frame, points)

            if detect:
                frame = model.detect(frame= frame, points= points)

            key = cv2.waitKey(1)
            if key == ord('q'):
                break
            elif key == ord('d'):
                points.append(points[0])
                detect = True

            # Hien anh ra man hinh
            result.write(frame)
            cv2.imshow("Intrusion Warning", frame)
            cv2.setMouseCallback('Intrusion Warning', handle_left_click, points)
        else:
            break

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

Bước 3: Sử dụng Qt Designer để vẽ tạo giao diện cho ứng dụng.

Tab 1- Detect video : Vẽ giao diện để ứng dụng có các trường dữ liệu nhập và nút bấm để phát hiện đối tượng trên video.



Tab 2- Guideline: Ghi hướng dẫn sử dụng ứng dụng:



Bước 4: Viết file main.py bắt các sự kiện trên giao diện đến code xử lý chúng:

- Hàm chooseVideoInput() được sử dụng để mở hộp thoại để chọn file video đầu vào. Hộp thoại được tạo bằng hàm QtWidgets.QFileDialog.getOpenFileName() và chỉ cho phép chọn các file có định dạng .mp4 hoặc .avi. Đường dẫn tới file được chọn sẽ được gán vào ô văn bản self.input để lưu trữ.

- Hàm chooseFolderOutput() được sử dụng để mở hộp thoại để chọn thư mục đầu ra. Hộp thoại được tạo bằng hàm QtWidgets.QFileDialog.getExistingDirectory(). Đường dẫn tới thư mục được chọn sẽ được gán vào ô văn bản self.output để lưu trữ.

- Hàm intrusion_warning() được sử dụng để xác nhận việc phát hiện xâm nhập trên video được chọn. Hộp thoại xác nhận được đào tạo bằng hàm. QtWidgets. QMessageBox.question(). Nếu người dùng chọn Yes, phương thức sẽ lấy đường dẫn tới video đầu vào và thư mục đầu ra từ hai ô văn bản self.input và self.output. Nếu một trong hai đường dẫn không được cung cấp, hộp thoại thông báo lỗi sẽ được hiển thị. Nếu cả hai đường dẫn đều được cung cấp, phương thức sẽ gọi hàm intrusion_warning() để phát hiện xâm nhập trên video và hiển thị hộp thoại thông báo thành công sau khi việc lưu trữ video kết quả hoàn tất.

```
    #function for events

    def chooseVideoInput(self):
        video_path = QtWidgets.QFileDialog.getOpenFileName(None, "Select choose video",
                                filter = "files (*.mp4 *.avi)")
        self.input.setText(video_path[0])

    def chooseFolderOutput(self):
        folder_path = QtWidgets.QFileDialog.getExistingDirectory(None, "Select choose folder")
        self.output.setText(folder_path)

    def intrusion_waring(self):
        ans = QtWidgets.QMessageBox.question(None, 'Confirm', "Intrusion detection on this video?", QtWidgets.QMe
        if ans == QtWidgets.QMessageBox.Yes:
            folder_data_in = self.input.text()
            folder_data_out = self.output.text()
            if folder_data_in=="" or folder_data_out=="":
                QtWidgets.QMessageBox.about(None,"Error","Empty folder path/video path")
            else:
                intrusion_warning(folder_data_in,folder_data_out)
                QtWidgets.QMessageBox.about(None,"Successful","Save result video successful!")
if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

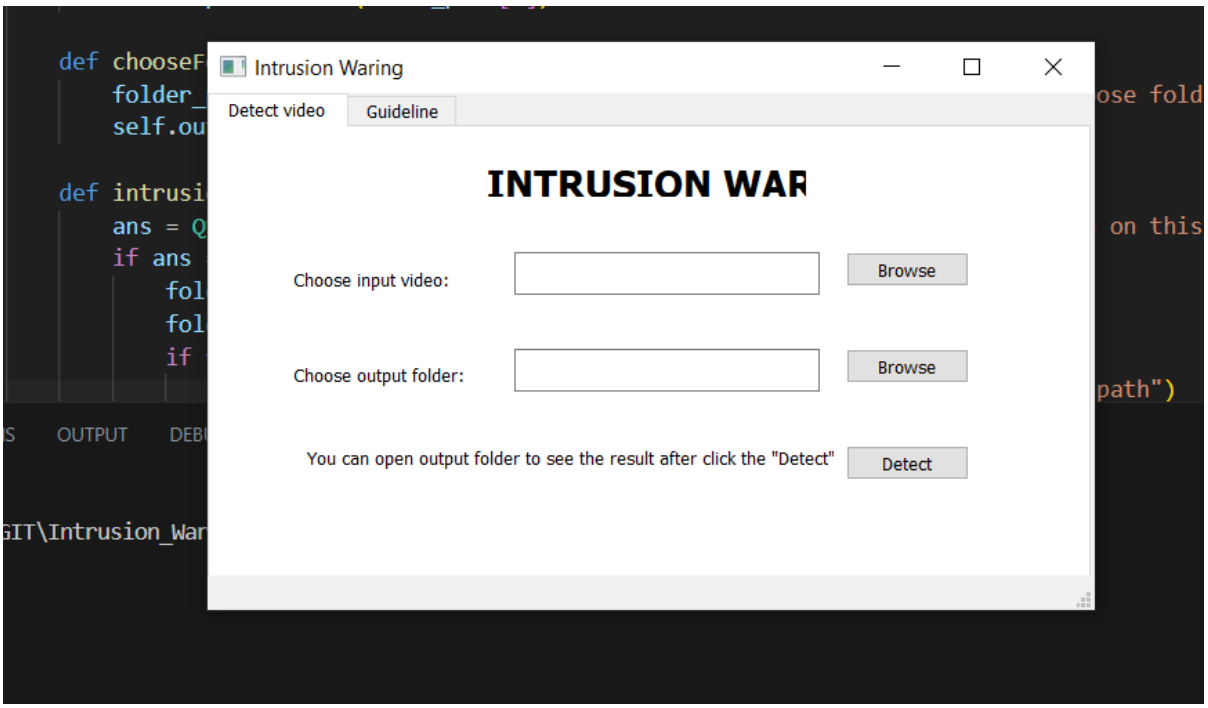
3.4 Kết quả

3.4.1 Giao diện của ứng dụng và đầu vào

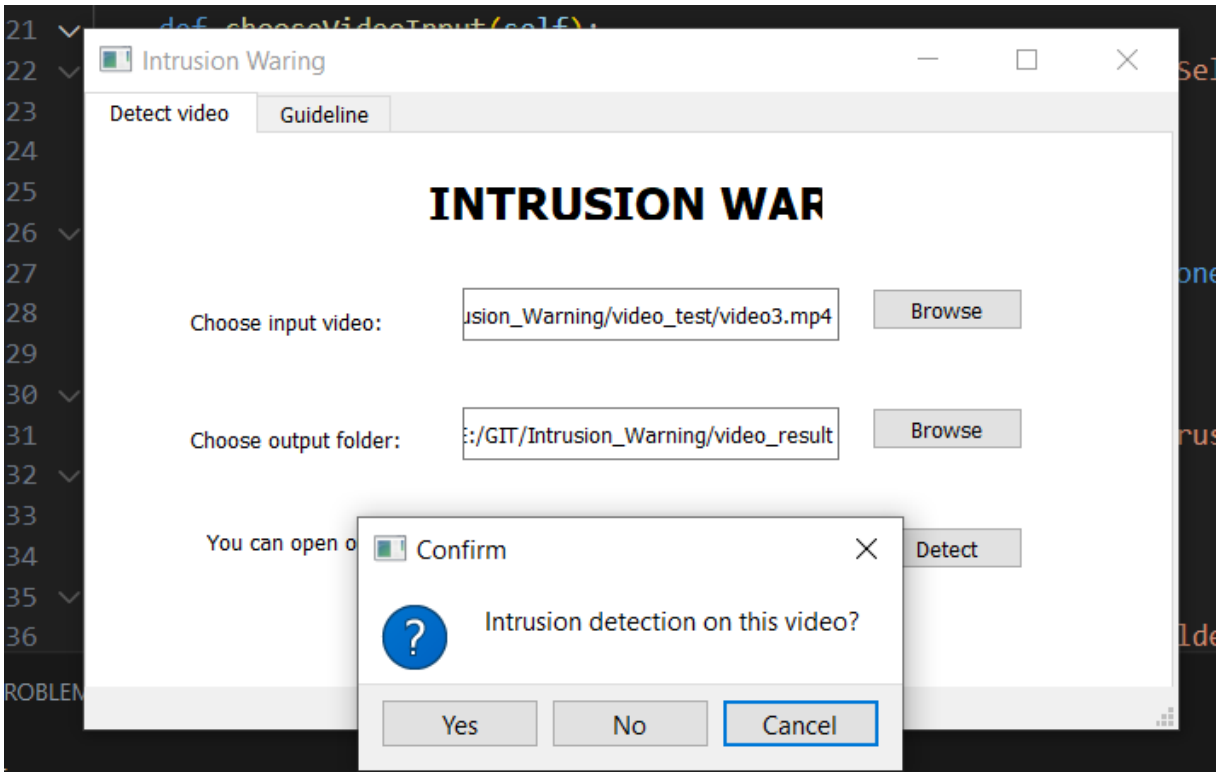
Giao diện của ứng dung có 2 tab

Tab 1: Phát hiện đối tượng cần NGƯỜI ĐI BỘ TRONG NHÀ VÀ CẢNH BÁO XÂM NHẬP

- Input 1: chọn video cần phát hiện đối tượng.
- Input 2: chọn thư mục lưu video kết quả sau khi phát hiện.
- Nút chọn video, nút chọn thư mục và nút để bắt đầu phát hiện.

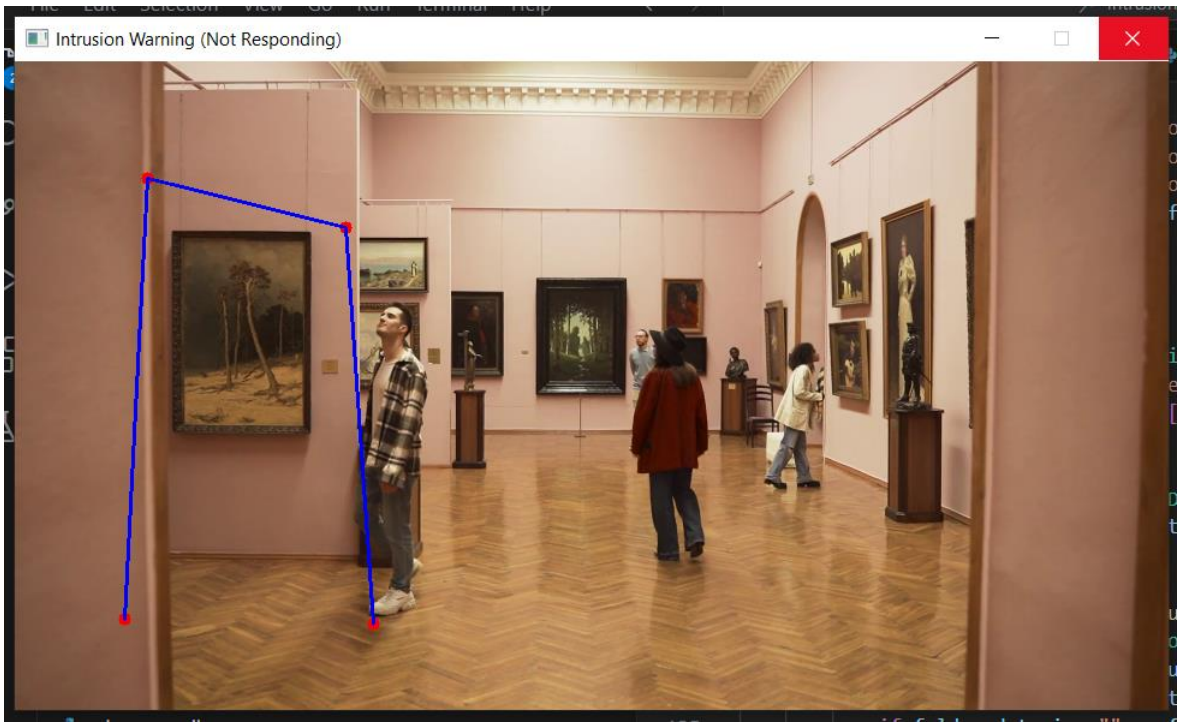


Sau khi ấn “detect” hiện ra hộp thoại xác nhận bắt đầu.

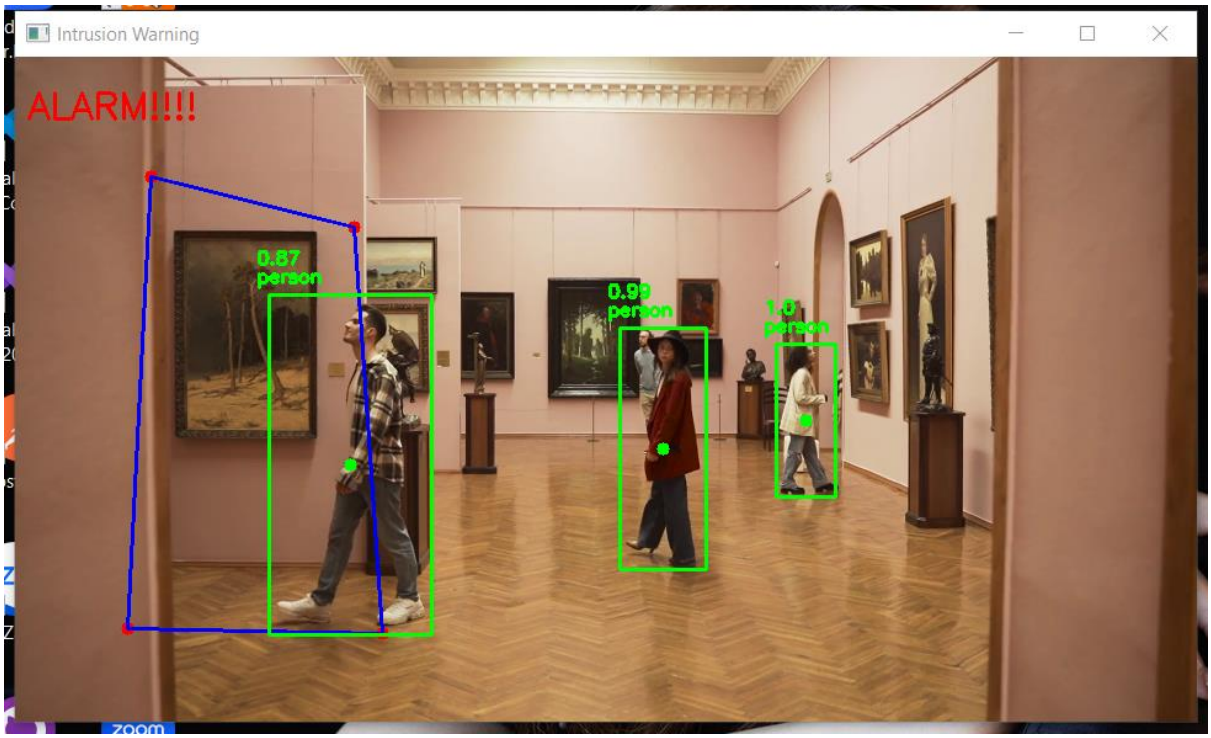


3.4.2 Phát hiện và đưa ra cảnh báo trên màn hình

Video hiện ra, vẽ các điểm để tạo ra 1 hình tứ giác trên khung hình và hình tứ giác đó được xem như là vùng cảnh báo khi có người đi vào.

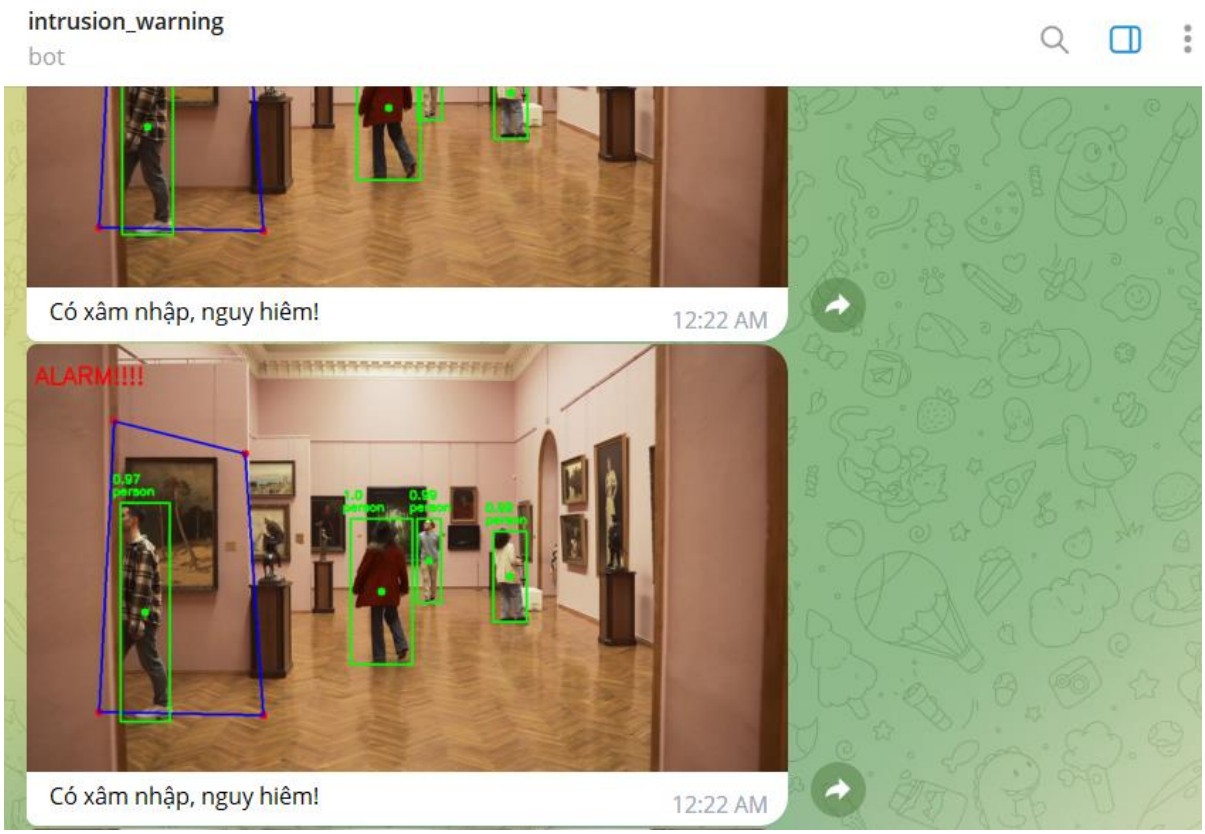


Khi đối tượng đi vào vùng cảnh báo đó thì trên màn hình sẽ hiển thị cảnh báo "ALARM" ngay trên khung hình.



3.4.3 Gửi tin nhắn thông báo có người xâm nhập qua Telegram

Khi có đối tượng đi vào vùng cảnh báo được vẽ trước đó ứng dụng sẽ chụp hình lại đối tượng xâm nhập vào vùng cảnh báo và gửi một tin nhắn cảnh báo kèm hình ảnh đã chụp tới telegram của người dùng thông qua ID của tài khoản telegram và token của tobot-chat được tạo trên telegram của tài khoản đó sau mỗi 15s một lần gửi:

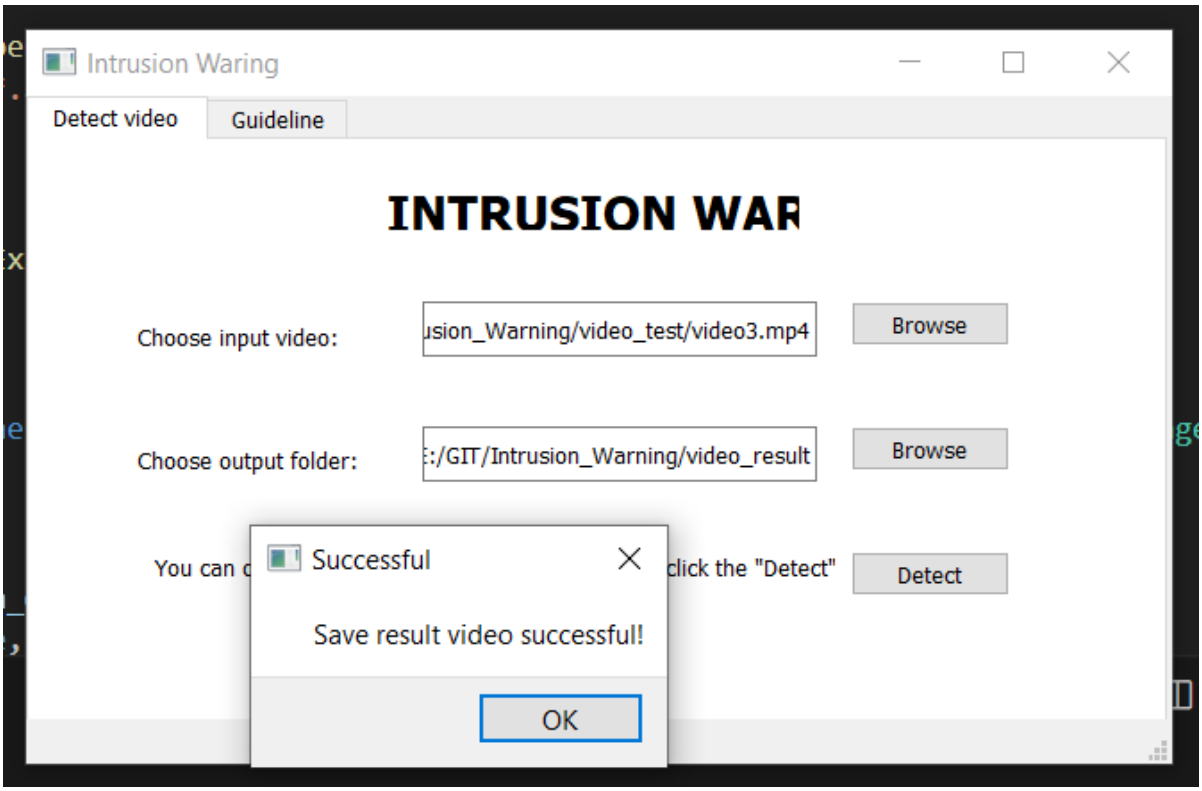


Ứng dụng chỉ đưa ra cảnh báo “ALARM” và chụp hình đối tượng xâm nhập gửi thông báo “Có xâm nhập, nguy hiểm” qua telegram chỉ khi đối tượng đối tượng nào đó xâm nhập vào vùng cảnh báo. Còn nếu đối tượng không xâm nhập vào vùng cảnh báo thì ứng dụng sẽ không đưa ra cảnh báo “ALARM” và cũng sẽ không gửi thông báo “Có xâm nhập, nguy hiểm” qua telegram.

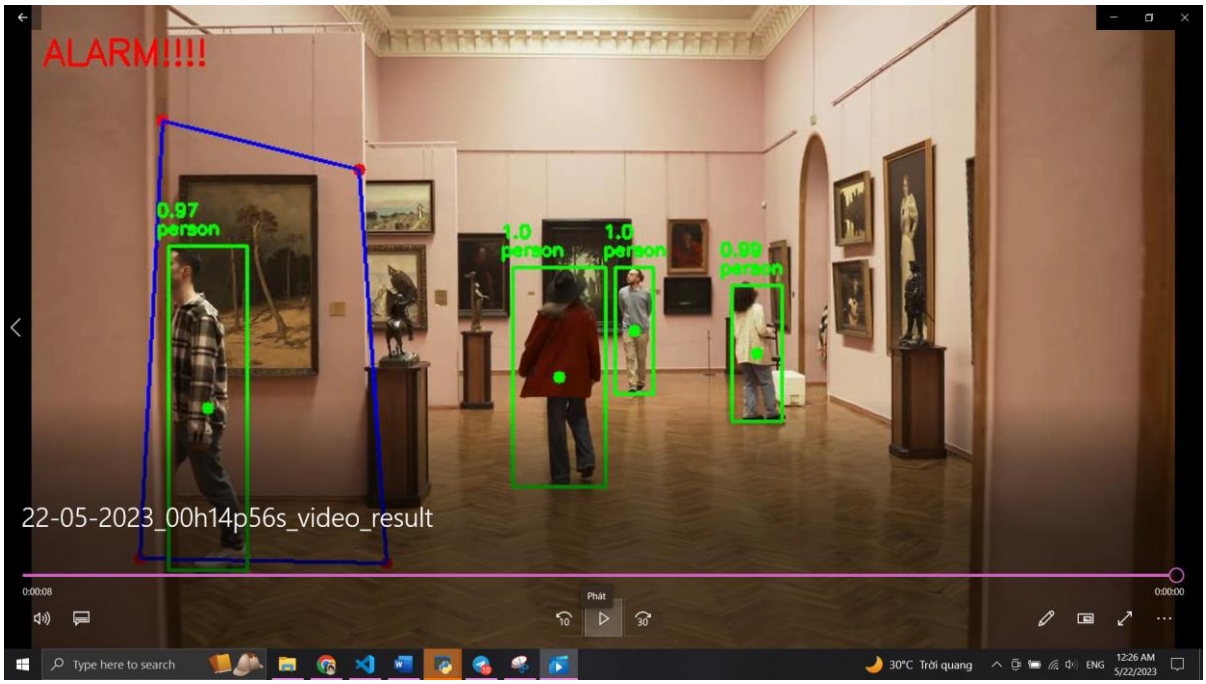
3.4.4 Lưu video kết quả

Sau khi chạy xong video, hoặc nhất phím “q” để kết thúc quá trình phát hiện trên video. Ứng dụng sẽ lưu kết quả video vào thư mục kết quả đã chọn. Video kết quả lưu dưới dạng mp4 với tên có kèm ngày giờ lưu.

Thông báo video kết quả đã được lưu



Video kết quả khi được mở lên xem, được lưu với tên kèm ngày giờ.



3.4.5 Độ chính xác của mô hình

- Độ chính xác của ứng dụng khi test với 100 video và hình ảnh an ninh thực tế (mỗi video tầm 30 giây). Kết quả cho thấy:

Data Test	Độ chính xác %
100 Video an ninh thực tế	94%

a. 94/100 video đưa ra cảnh báo đúng ngay khi có người đi vào vùng, không bị bắt nhầm, bỏ sót người.

- 3/100 video đưa ra cảnh báo chậm hơn khi người đã vào sâu hơn vùng cấm (trễ 3-4 giây).
- 2/100 video có phát hiện thiếu người (trường hợp 3-4 người đi quá sát nhau bắt thành 2 người).

- Độ chính xác của ứng dụng khi test với hình ảnh thực tế có 100, 50, 20 người

Data Test	Độ chính xác %
Hình ảnh có 100 người	94%
Hình ảnh có 50 người	96%
Hình ảnh có 20 người	99%

- Hình ảnh 100 người model bắt được 94/100 người 6 người bắt hụt vì 3 4 người đứng quá sát nhau bắt thành 2 người.
- Hình ảnh có 50 người model bắt được 48/50 người 2 người bắt hụt vì bị che khuất do đứng quá sát nhau.
- Hình ảnh có 20 người model bắt được 19/20 người 1 người bắt hụt vì 2 người đứng sát nhau bắt thành 1 người.

3.5 Thông tin đóng góp đề tài

Với đề tài sử dụng thị giác máy tính để phát hiện đối tượng NGƯỜI ĐI BỘ TRONG NHÀ VÀ CẢNH BÁO XÂM NHẬP, em đã có những đóng góp vào đề tài:

- Tổng quan bài toán phát hiện đối tượng trong thị giác máy tính và bài toán của đề tài.

- Trình bày chi tiết phương pháp phát hiện đối tượng họ nhà YOLO và mô hình YOLOv4 được sử dụng.
- Xây dựng ứng dụng phát hiện đối tượng người đi vào vùng NGƯỜI ĐI BỘ TRONG NHÀ VÀ CẢNH BÁO XÂM NHẬP trong video.
- Ứng dụng được cài đặt bằng ngôn ngữ lập trình Python, viết code logic kết hợp với mô hình AI phát hiện đối tượng tượng người YOLOv4 được đào tạo trên tập COCO. Với độ chính xác phát hiện người lên đến trên 94%.
- Ứng dụng gồm các chức năng:
    - Chọn video và thư mục lưu kết quả. Xử lý đầu vào video và vẽ vùng NGƯỜI ĐI BỘ TRONG NHÀ VÀ CẢNH BÁO XÂM NHẬP trên video đầu vào.
    - Phát hiện đối tượng người di chuyển vào vùng NGƯỜI ĐI BỘ TRONG NHÀ VÀ CẢNH BÁO XÂM NHẬP qua từng frame của video. Hiển thị loại đối tượng và xác suất dự đoán.
    - Đưa ra cảnh báo trên màn hình khi có đối tượng người đi vào vùng NGƯỜI ĐI BỘ TRONG NHÀ VÀ CẢNH BÁO XÂM NHẬP.
    - Gửi cảnh báo bằng bot-chat qua telegram cho người sử dụng.
    - Lưu lại video kết quả với tên kèm theo ngày giờ.

Chương 4. KẾT LUẬN

Trong báo cáo này, em đã nghiên cứu và trình bày chi tiết phương pháp phát hiện đối tượng với mô hình họ nhà YOLO nói chung và YOLOv4 nói riêng. Em đã kết hợp giữa viết code logic và mô hình AI (YOLOv4) để xây dựng nên ứng dụng phát hiện đối tượng người đi bộ đi vào vùng cảnh báo được vẽ trên video và phát ra cảnh báo xâm nhập trên màn hình và gửi thông báo đến telegram.

Tuy nhiên, vì những hạn chế về mặt thời gian và tài nguyên nên em mới chỉ phát hiện đối tượng được trên video, chưa thử nghiệm được video trực tiếp với camera kết nối bên ngoài. Tuy nhiên đây cũng là hướng đi để em nghiên cứu và phát triển đề tài về sau.

TÀI LIỆU THAM KHẢO

[1] Xin Yu, Gao Hongbo, Zhao Jianhui, & Zhou Mo (2018). Overview of deep learning intelligent driving methods. Journal of Tsinghua University (Natural Science Edition) 58(04), 438-444.

https://www.researchgate.net/publication/327801383_Overview_of_deep_learning_intelligent_driving_methods

[2] Geng Yining, Liu Shuaishi, Liu Taiting, Yan Wenyang, & Lian Yufeng. (2021). Survey of pedestrian detection technology based on computer vision. Journal of Computer Applications 41(S1), 43-50.

https://www.researchgate.net/publication/356838324_A_Survey_on_Pedestrian_Detection_System_Using_Computer_Vision_and_Deep_Learning

[3] Girshick, Donahue, Darrell T & Malik J (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 580-587).

https://ieeexplore.ieee.org/document/6909475

[4] Girshick, R. (2015). Fast r-CNN. In Proceedings of the IEEE international conference on computer vision (pp. 1440-1448).

https://arxiv.org/abs/1504.08083

[5] Ren, He, Girshick, & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems.

https://arxiv.org/pdf/1506.01497.pdf

[6] Redmon, Divvala, S, Girshick & Farhadi (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

https://arxiv.org/abs/1506.02640

[7] Liu, Anguelov, Erhan, Szegedy, Reed, Fu, C. Y, & Berg (2016, October). SSD: Single shot multibox detector. In European conference on computer vision (pp. 21-37). Springer, Cham.

https://arxiv.org/pdf/1512.02325.pdf

[8] Redmon, & Farhadi (2017). YOLO9000: better, faster, stronger. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7263-7271).

https://ieeexplore.ieee.org/document/8100173

[9] Redmon & Farhadi (2018). Yolov3: An incremental improvement. arXiv preprint arXiv: 1804.02767.

https://arxiv.org/pdf/1804.02767.pdf

[10] Bochkovskiy A, Wang & Liao (2020). Yolov4: Optimal speed and precision of object detection. arXiv preprint arXiv: 2004.10934.

https://arxiv.org/abs/2004.10934

[11] Lin, Goyal, Girshick, He, K., & Dollár, P. (2017). Focal loss for dense object detection. In Proceedings of the IEEE international conference on computer vision (pp. 2980-2988).

https://openaccess.thecvf.com/content_ICCV_2017/papers/Lin_Focal_Loss_for_ICCV_2017_paper.pdf

[12] Zhang Mengge., Li Jian., & Chen Huiwen. (2022). Real-time video pedestrian detection algorithm based on YoloV3. Microcontrollers & Embedded Systems, 22(06), 29-32.

https://www.researchgate.net/publication/354094757_Realtime_Pedestrian_Detection_Algorithm_Based_on_Improved_YOLOv3

[13] Hao Xuzheng., & Chai Zhengyi. (2019). Research on Improved YOLOv4-based Pedestrian Detection Model

https://drpress.org/ojs/index.php/HSET/article/view/1090

[14] Shi Ruijiao., Chen Houjin., Li Jupeng., Li Yanfeng., Li Feng., & Wan Chengkai. (2022). Small-scale pedestrian detection algorithm in railway scene based on attention and multi-level feature fusion. Journal of the China Railway Society, 44(05),76-83.

https://ietresearch.onlinelibrary.wiley.com/doi/epdf/10.1049/cvi2.12125

[15] Li Xiaoyan, Fu Huitong, Niu Wentao, Wang Peng, Lv Zhigang & Wang Weiming.

(2022). Multimodal pedestrian detection algorithm based on deep learning. Journal of Xi'an Jiaotong University, (10), 76-83.

https://www.sciencedirect.com/science/article/pii/S0167739X21001813

[16] Kang Shuai, Zhang Jianwu, Zhu Zunjie & Tong Guofeng(2021). Improved YOLOv4 algorithm for dense object pedestrian detection in complex visual scenes. Telecommunications Science, 37(08), 46-56

https://www.sciencedirect.com/science/article/pii/S0045790621002445

[17] YOLOv4: Optimal Speed and Accuracy of Object Detection.23 Apr 2020. Prof: Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao.

https://arxiv.org/abs/2004.10934

[18] YOLO You Only Look Once. 09 Mar 2020.Author: Pham Dinh Khanh

https://phamdinhkhanh.github.io/2020/03/09/DarknetAlgorithm.html

[19] Real-time object detection method based on improved YOLOv4-tiny. Zicong Jiang, Liquan Zhao, Shuaiyang Li, Yanfei Jia

https://arxiv.org/abs/2011.04244

[20] Research on Improved YOLOv4-based Pedestrian Detection Model. Yujun Li, Ruiqi Hu.

https://www.researchgate.net/publication/362624610_Research_on_Improved_YOLOv4-based_Pedestrian_Detection_Model