

**BỘ THÔNG TIN VÀ TRUYỀN THÔNG  
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

-----



# **BÁO CÁO THỰC TẬP TỐT NGHIỆP ĐẠI HỌC**

*Đề tài: “Ứng dụng mô hình YOLO dự báo CHÁY thời gian thực”*

**Người hướng dẫn : HUỖNH TRỌNG THỨA**

**Sinh viên thực hiện : PHU DỰ THẮNG**

**Mã số sinh viên : N20DCCN146**

**Lớp : D20CQCNP02-N**

**Khóa : 2020 – 2025**

**Ngành : CÔNG NGHỆ PHẦN MỀM**

**Hệ : ĐẠI HỌC CHÍNH QUY**

**TP.HCM, tháng 08/2024**

**BỘ THÔNG TIN VÀ TRUYỀN THÔNG  
HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

-----



# **BÁO CÁO THỰC TẬP TỐT NGHIỆP ĐẠI HỌC**

*Đề tài: “Ứng dụng mô hình YOLO dự báo CHÁY thời gian thực”*

**Người hướng dẫn : HUỖNH TRỌNG THỨA**

**Sinh viên thực hiện : PHU DỰ THẮNG**

**Mã số sinh viên : N20DCCN146**

**Lớp : D20CQCNPĐ02-N**

**Khóa : 2020 – 2025**

**Ngành : CÔNG NGHỆ PHẦN MỀM**

**Hệ : ĐẠI HỌC CHÍNH QUY**

**TP.HCM, tháng 08/2024**

## PHIẾU NHẬN XÉT CỦA ĐƠN VỊ THỰC TẬP

### NHẬN XÉT QUÁ TRÌNH THỰC TẬP CỦA SINH VIÊN

Họ tên sinh viên: Phu Dư Thắng..... Lớp: D20CGCNPM02-N.....

Nơi thực tập: CÔNG TY TNHH AU-AVAGO.....

Họ tên người nhận xét: Huỳnh Hữu Ngọc Ý.....

Nội dung đánh giá	Xếp loại (đánh dấu vào ô được chọn)				
	Tốt	Khá	Trung bình	Trung bình yếu	Yếu
<b>I. Tính kỷ luật và tư chất</b>					
I.1. Thực hiện nội quy	X				
I.2. Thái độ làm việc	X				
I.3. Năng lực tiếp thu	X				
I.4. Khả năng vượt khó	X				
I.5. Giao tiếp và ứng xử	X				
<b>II. Khả năng chuyên môn</b>					
II.1. Kiến thức	X				
II.2. Kỹ năng thực hành	X				
II.3. Năng lực ngoại ngữ		X			
II.4. Kỹ năng làm việc nhóm		X			
II.5. Tính sáng tạo		X			
<b>III. Kết quả thực hiện đề tài được giao</b>					
I.1. Thực hiện yêu cầu về nội dung	X				
I.2. Thực hiện yêu cầu về tiến độ		X			
<b>IV. Nhận xét khác và lời khuyên cho sinh viên:</b> Chấp hành đúng các quy định về giờ giấc kỷ luật do công ty đề ra, hoàn thành tốt các nhiệm vụ được giao. Cần chủ động hơn trong công việc.					
<b>V. Điểm tổng kết thực tập tốt nghiệp (thang điểm 10):</b> <u>9.0</u> ..					

XÁC NHẬN CỦA TỔ CHỨC/DOANH NGHIỆP



NGƯỜI NHẬN XÉT  
(ký tên ghi rõ họ tên)

Huỳnh Hữu Ngọc Ý

- Ghi chú: Cơ quan xác nhận phải là cơ quan đăng ký thực tập, hoặc là cơ quan chủ quản của đơn vị thực tập.
- Dấu xác nhận là dấu tròn.

## **LỜI CẢM ƠN**

Lời đầu tiên, em xin gửi lời cảm ơn chân thành đến các Thầy Cô giáo tại Học viện Công nghệ Bru chính Viễn thông đã tận tình dẫn dắt và truyền đạt cho em nền tảng kiến thức vững chắc cùng với các kỹ năng cần thiết trong những năm học vừa qua. Em thật sự đã có cơ hội được trau dồi kiến thức và rèn luyện kỹ năng dưới sự hướng dẫn tận tâm của các Thầy Cô, và đó là điều vô cùng quý giá đối với em.

Đồng thời, em xin chân thành bày tỏ lòng biết ơn sâu sắc đến Thầy Huỳnh Trọng Thừa, người đã dành thời gian và công sức tận tình hướng dẫn và hỗ trợ em trong suốt quá trình nghiên cứu và thực hiện đề tài. Nhờ sự quan tâm, chỉ dẫn và những kinh nghiệm quý báu mà Thầy chia sẻ, em đã học tập thêm nhiều kiến thức mới, góp phần hoàn thiện nền tảng kiến thức và kỹ năng, cũng như phát triển bản thân. Thầy không chỉ truyền đạt kiến thức chuyên sâu mà còn truyền cho em tinh thần học tập và làm việc nghiêm túc. Điều này có ý nghĩa to lớn và sẽ trở thành những phẩm chất quý báu giúp em hoàn thiện bản thân trong học tập và công việc.

Bên cạnh đó, em ý thức rằng với lượng kiến thức và kinh nghiệm còn hạn chế, bài làm của em khó tránh khỏi những thiếu sót và sai lầm. Em rất mong nhận được sự thông cảm và những góp ý, nhận xét quý báu từ quý Thầy Cô. Những đề xuất này sẽ giúp em khắc phục và hoàn thiện đề tài một cách tốt nhất.

Cuối cùng, em chúc quý Thầy Cô luôn dồi dào sức khỏe để tiếp tục sứ mệnh cao cả là truyền đạt tri thức cho các thế hệ mai sau. Sự cống hiến của quý Thầy Cô không chỉ giúp đỡ riêng em mà còn góp phần hình thành và phát triển tài năng cho nhiều thế hệ sinh viên khác. Một lần nữa, em xin chân thành cảm ơn và hy vọng quý Thầy Cô sẽ luôn tràn đầy niềm vui và đạt được nhiều thành công trong sự nghiệp. Em rất mong sẽ có cơ hội được hợp tác và học hỏi từ quý Thầy Cô trong những dự án tương lai.

Em xin chân thành cảm ơn!

*TP. Hồ Chí Minh, ngày      tháng      năm 2024*

**SINH VIÊN THỰC HIỆN ĐỀ TÀI**

*(Sinh viên ký và ghi rõ họ tên)*

**PHU DỰ THẮNG**

# MỤC LỤC

LỜI CẢM ƠN .....	i
MỤC LỤC .....	ii
DANH MỤC CÁC HÌNH VẼ.....	iv
DANH MỤC CÁC KÝ HIỆU VÀ CHỮ VIẾT TẮT .....	vi
MỞ ĐẦU .....	1
CHƯƠNG 1. CƠ SỞ LÝ THUYẾT .....	2
1. NGHIÊN CỨU MÔ HÌNH FASTER R-CNN VÀ YOLO .....	2
1.1. MÔ HÌNH FASTER R-CNN .....	2
1.1.a. Kiến trúc mô hình.....	3
1.1.b. Cơ chế hoạt động.....	4
1.1.c. Chi tiết kỹ thuật .....	5
1.1.c.1. Intersection over Union.....	5
1.1.c.2. Non-Maximum Suppression .....	6
1.1.c.3. Region of Interest .....	7
1.1.c.4. Region Proposal Network .....	8
1.2. MÔ HÌNH YOLO .....	10
1.2.a. Kiến trúc mô hình.....	11
1.2.b. Cơ chế hoạt động.....	12
1.2.c. Chi tiết kỹ thuật .....	13
1.2.c.1. Cross Stage Partial Network.....	13
1.2.c.2. Path Aggregation Network.....	14
1.2.c.3. Spatial-Channel Decoupled Downspampling.....	15
1.2.c.4. NMS-Free Training.....	16
1.3. SO SÁNH TỔNG QUAN .....	17
2. TÌM HIỂU BÀI TOÁN DỰ BÁO CHÁY TRONG NHÀ .....	18
2.1. PHƯƠNG PHÁP .....	18
2.2. THÁCH THỨC .....	18

<b>3. TÌM HIỂU TẬP DỮ LIỆU CẢNH BÁO CHÁY BẰNG HÌNH ẢNH .....</b>	<b>19</b>
<b>3.1. PHÂN LOẠI DỮ LIỆU .....</b>	<b>19</b>
<b>3.2. GÁN NHÃN DỮ LIỆU .....</b>	<b>19</b>
<b>3.3. CHUẨN BỊ DỮ LIỆU .....</b>	<b>19</b>
<b>CHƯƠNG 2. THỰC NGHIỆM VÀ SO SÁNH MÔ HÌNH FASTER R-CNN VÀ YOLO TRÊN CÙNG TẬP DỮ LIỆU.....</b>	<b>20</b>
<b>1. CHUẨN BỊ TẬP DỮ LIỆU .....</b>	<b>20</b>
<b>1.1. THU THẬP HÌNH ẢNH .....</b>	<b>20</b>
<b>1.2. GÁN NHÃN DỮ LIỆU .....</b>	<b>21</b>
<b>1.3. TẠO TẬP DỮ LIỆU .....</b>	<b>22</b>
<b>2. HUẤN LUYỆN MÔ HÌNH FASTER R-CNN .....</b>	<b>24</b>
<b>3. HUẤN LUYỆN MÔ HÌNH YOLO .....</b>	<b>30</b>
<b>4. SO SÁNH THỰC NGHIỆM.....</b>	<b>35</b>
<b>CHƯƠNG 3. XÂY DỰNG VÀ THỬ NGHIỆM CÔNG CỤ DỰ BÁO CHÁY THỜI GIAN THỰC DỰA TRÊN HÌNH ẢNH.....</b>	<b>36</b>
<b>1. GIỚI THIỆU CÔNG NGHỆ .....</b>	<b>36</b>
<b>2. XÂY DỰNG CHƯƠNG TRÌNH.....</b>	<b>38</b>
<b>3. KẾT QUẢ THỬ NGHIỆM.....</b>	<b>41</b>
<b>KẾT LUẬN, KIẾN NGHỊ .....</b>	<b>42</b>
<b>TÀI LIỆU THAM KHẢO.....</b>	<b>43</b>

## DANH MỤC CÁC HÌNH VẼ

Hình 1.1.1.a.1 Kiến trúc mô hình Faster R–CNN .....	3
Hình 1.1.1.c.1.1 Hình ảnh minh họa thuật toán IoU.....	5
Hình 1.1.1.c.2.1 Hình ảnh minh họa thuật toán NMS.....	6
Hình 1.1.1.c.3.1 Hình ảnh minh họa thuật toán RoI Pooling .....	7
Hình 1.1.1.c.4.1 Hình ảnh minh họa Region Proposal Network.....	8
Hình 1.1.1.c.4.2 Hình ảnh minh họa kỹ thuật Anchors – 1 .....	9
Hình 1.1.1.c.4.3 Hình ảnh minh họa kỹ thuật Anchors – 2.....	9
Hình 1.1.2.a.1 Kiến trúc mô hình YOLO.....	11
Hình 1.1.2.c.1.1 Hình ảnh minh họa kiến trúc CSPNet .....	13
Hình 1.1.2.c.2.1 Hình ảnh minh họa kiến trúc PAN.....	14
Hình 1.1.2.c.3.1 Hình ảnh minh họa thao tác Depthwise và Pointwise.....	15
Hình 1.1.2.c.4.1 Hình ảnh minh họa chiến lược NMS–Free Training .....	16
Hình 2.1.1.1 Một số hình ảnh trong tập dữ liệu.....	20
Hình 2.1.2.1 Hình ảnh minh họa quy trình gán nhãn Roboflow .....	21
Hình 2.1.3.1 Bước 1 trong quy trình tạo tập dữ liệu Roboflow .....	22
Hình 2.1.3.2 Bước 2 trong quy trình tạo tập dữ liệu Roboflow .....	22
Hình 2.1.3.3 Bước 3 trong quy trình tạo tập dữ liệu Roboflow .....	23
Hình 2.1.3.4 Bước 4 trong quy trình tạo tập dữ liệu Roboflow .....	23
Hình 2.1.3.5 Bước 5 trong quy trình tạo tập dữ liệu Roboflow .....	23
Hình 2.2.1 Mã nguồn cài đặt môi trường Faster R–CNN.....	24
Hình 2.2.2 Mã nguồn tải tập dữ liệu từ Roboflow (COCO).....	24
Hình 2.2.3 Mã nguồn đăng ký tập dữ liệu với đối tượng COCO .....	25
Hình 2.2.4 Mã nguồn xây dựng lớp huấn luyện mô hình Faster R–CNN .....	25
Hình 2.2.5 Mã nguồn huấn luyện mô hình Faster R–CNN .....	26
Hình 2.2.6 Mã nguồn cấu hình đối tượng dự đoán mô hình Faster R–CNN .....	27
Hình 2.2.7 Mã nguồn thực hiện dự đoán mô hình Faster R–CNN.....	28
Hình 2.2.8 Kết quả huấn luyện mô hình Faster R–CNN .....	28
Hình 2.2.9 Một số kết quả kiểm thử bởi Faster R–CNN.....	29
Hình 2.3.1 Mã nguồn cài đặt môi trường YOLOv10.....	30
Hình 2.3.2 Mã nguồn tải trọng số mô hình YOLOv10 .....	30
Hình 2.3.3 Mã nguồn tải tập dữ liệu từ Roboflow (YOLOv9).....	31

Hình 2.3.4 Mã nguồn tệp tin .yaml .....	31
Hình 2.3.5 Mã nguồn huấn luyện mô hình YOLOv10 .....	32
Hình 2.3.6 Mã nguồn thực hiện dự đoán mô hình YOLOv10 .....	32
Hình 2.3.7 Kết quả huấn luyện mô hình YOLOv10.....	33
Hình 2.3.8 Một số kết quả kiểm thử bởi YOLOv10 .....	34
Hình 3.1.1 Hình ảnh logo ngôn ngữ Python .....	36
Hình 3.1.2 Hình ảnh logo thư viện Customtkinter .....	36
Hình 3.1.3 Hình ảnh logo thư viện Pygame .....	36
Hình 3.1.4 Hình ảnh logo thư viện OpenCV .....	37
Hình 3.1.5 Hình ảnh logo thư viện Ultralytics.....	37
Hình 3.2.1 Mã nguồn cấu hình âm thanh cảnh báo.....	38
Hình 3.2.2 Mã nguồn tải mô hình.....	38
Hình 3.2.3 Mã nguồn cấu hình Webcam.....	38
Hình 3.2.4 Mã nguồn xử lý hình ảnh.....	39
Hình 3.2.5 Mã nguồn trình bày hình ảnh lên giao diện .....	40
Hình 3.2.6 Mã nguồn kết thúc chương trình .....	40
Hình 3.3.1 Hình ảnh công cụ thử nghiệm .....	41



## DANH MỤC CÁC KÝ HIỆU VÀ CHỮ VIẾT TẮT

<i>Chữ viết tắt</i>	<i>Chữ đầy đủ</i>	<i>Giải thích</i>
<i>AP</i>	<i>Average Precision</i>	Độ chính xác trung bình, chỉ số đánh giá hiệu suất mô hình nhận diện đối tượng.
<i>API</i>	<i>Application Programming Interface</i>	Giao diện lập trình ứng dụng, một tập hợp các quy tắc giúp phần mềm giao tiếp với nhau.
<i>AR</i>	<i>Average Recall</i>	Độ nhớ trung bình, chỉ số đánh giá hiệu suất mô hình nhận diện đối tượng.
<i>CNN</i>	<i>Convolutional Neural Network</i>	Mạng nơ-ron được thiết kế đặc biệt để xử lý dữ liệu có cấu trúc lưới.
<i>COCO</i>	<i>Common Objects in Context</i>	Tên một tập dữ liệu chứa các đối tượng thông dụng trong ngữ cảnh.
<i>CSPNet</i>	<i>Cross-Stage Partial Network</i>	Mạng một phần xuyên giai đoạn, kiến trúc mạng đặc biệt giúp giảm mất mát thông tin.
<i>CVPR</i>	<i>Computer Vision and Pattern Recognition</i>	Hội nghị Thị giác máy tính và Nhận diện khuôn mẫu.
<i>FPN</i>	<i>Feature Pyramid Network</i>	Mạng kim tự tháp đặc trưng, kiến trúc mạng đặc biệt giúp giảm mất mát thông tin.
<i>ILSVRC</i>	<i>ImageNet Large Scale Visual Recognition Challenge</i>	Thử thách nhận diện hình ảnh quy mô lớn ImageNet.
<i>IoU</i>	<i>Intersection over Union</i>	Chỉ số đánh giá sự trùng khớp giữa hai vùng hình ảnh.
<i>NMS</i>	<i>Non-Maximum Suppression</i>	Kỹ thuật loại bỏ phần tử có giá trị ưu tiên không cực đại.
<i>NeurIPS</i>	<i>Neural Information Processing Systems</i>	Hội nghị Hệ thống Xử lý thông tin nơ-ron.
<i>mAP50</i>	<i>mean Average Precision at IoU 0.5</i>	Độ chính xác trung bình tại ngưỡng IoU là 0.5.

<i>mAP50–95</i>	<i>mean Average Precision at IoU from 0.5 to 0.95</i>	Độ chính xác trung bình tại các ngưỡng IoU từ 0.5 đến 0.95.
<i>OpenCV</i>	<i>Open Source Computer Vision Library</i>	Thư viện Thị giác máy tính mã nguồn mở.
<i>PAN</i>	<i>Path Aggregation Network</i>	Mạng tổng hợp đường dẫn, kiến trúc mạng đặc biệt giúp giảm mất mát thông tin.
<i>PASCAL VOC</i>	<i>Pattern Analysis, Statistical Modeling and Computational Learning Visual Object Classes</i>	Tên một tập dữ liệu chứa các lớp đối tượng thị giác phân tích mẫu, mô hình thống kê và học tính toán.
<i>R–CNN</i>	<i>Region-based Convolutional Neural Network</i>	Mạng nơ-ron tích chập dựa trên vùng.
<i>RoI</i>	<i>Region of Interest</i>	Vùng quan tâm, cách hiểu khác của vùng đề xuất.
<i>RPN</i>	<i>Region Proposal Network</i>	Mạng đề xuất vùng, mạng nơ-ron tạo ra vùng đề xuất có định hướng.
<i>SCDD</i>	<i>Spatial–Channel Decoupled Downspampling</i>	Kỹ thuật tách biệt thông tin không gian và kênh giúp giảm mất mát thông tin.
<i>YOLO</i>	<i>You Only Look Once</i>	Tên chung cho một dòng mô hình học sâu với ý nghĩa "Bạn chỉ nhìn một lần".
<i>yaml</i>	<i>YAML Ain't Markup Language</i>	Tên định dạng tệp tin để lưu trữ các thông tin cấu hình hoặc trao đổi dữ liệu giữa các ứng dụng.

## MỞ ĐẦU

Trong những năm gần đây, tình hình cháy nổ tại các khu dân cư, nhà máy và khu rừng ngày càng trở nên phức tạp và nghiêm trọng, gây thiệt hại lớn về người và tài sản. Do đó, việc phát hiện sớm và xử lý kịp thời các vụ cháy là yếu tố then chốt giúp giảm thiểu những thiệt hại này. Vì thế, việc nghiên cứu và phát triển các hệ thống dự báo cháy sớm là một nhu cầu cấp thiết và quan trọng.

Qua đó, việc phát hiện và cảnh báo cháy sớm không chỉ giúp giảm thiểu thiệt hại mà còn góp phần bảo vệ an toàn cho môi trường và cộng đồng. Với sự phát triển mạnh mẽ của công nghệ trí tuệ nhân tạo và máy học, đặc biệt là các mô hình nhận diện đối tượng như YOLO, chúng ta giờ đây đã có thể xây dựng các hệ thống giám sát hiệu quả và nhanh chóng hơn bao giờ hết.

Đề tài này được nghiên cứu với mục đích là xây dựng và phát triển một ứng dụng dự báo cháy thời gian thực, với cốt lõi là mô hình YOLO để phân tích và nhận diện các đám cháy từ hình ảnh thực tế ghi nhận được. Mô hình này không chỉ sở hữu tốc độ xử lý nhanh chóng mà còn giúp cải thiện độ chính xác trong việc nhận diện đối tượng, từ đó hỗ trợ đắc lực cho công tác phòng chống và ứng phó kịp thời khi xảy ra cháy nổ.

Trước đây, các phương pháp truyền thống tập trung vào sử dụng cảm biến nhiệt và khói với nhược điểm có độ trễ nhất định và không phù hợp với một số môi trường đặc thù. Hiện tại, đã có nhiều nghiên cứu và ứng dụng các mô hình trí tuệ nhân tạo, và mô hình YOLO đã chứng minh được sự hiệu quả trong việc phát hiện đối tượng theo thời gian thực. Dù đã được áp dụng trong nhiều lĩnh vực khác nhau, mô hình này vẫn còn đang trong giai đoạn phát triển và hoàn thiện.

Trong bài nghiên cứu này, tôi sẽ sử dụng phương pháp thực nghiệm, bao gồm việc thu thập dữ liệu là hình ảnh các đám cháy, sau đó xây dựng mô hình YOLO để phân tích và nhận diện đám cháy. Tổng quan, quá trình nghiên cứu sẽ trải qua các bước: chuẩn bị dữ liệu, huấn luyện mô hình, đánh giá hiệu suất và tối ưu hóa mô hình.

Phạm vi nghiên cứu sẽ tập trung vào việc phát hiện các đám cháy theo thời gian thực từ nguồn dữ liệu là hình ảnh được ghi nhận từ thiết bị ghi hình, sau đó cảnh báo âm thanh đến người giám sát. Nghiên cứu cũng tập trung vào phân tích và tối ưu hóa mô hình để nâng cao độ chính xác nhằm đáp ứng tính khả thi của ứng dụng trong các điều kiện thực tế.

Cấu trúc của báo cáo sẽ trình bày 3 nội dung chính, bao gồm:

- Chương 1: Cơ sở lý thuyết.
- Chương 2: Thực nghiệm và so sánh mô hình Faster R-CNN và YOLO trên cùng tập dữ liệu.
- Chương 3: Xây dựng và thử nghiệm công cụ dự báo cháy thời gian thực dựa trên hình ảnh.

## CHƯƠNG 1. CƠ SỞ LÝ THUYẾT

### 1. NGHIÊN CỨU MÔ HÌNH FASTER R-CNN VÀ YOLO

#### 1.1. MÔ HÌNH FASTER R-CNN

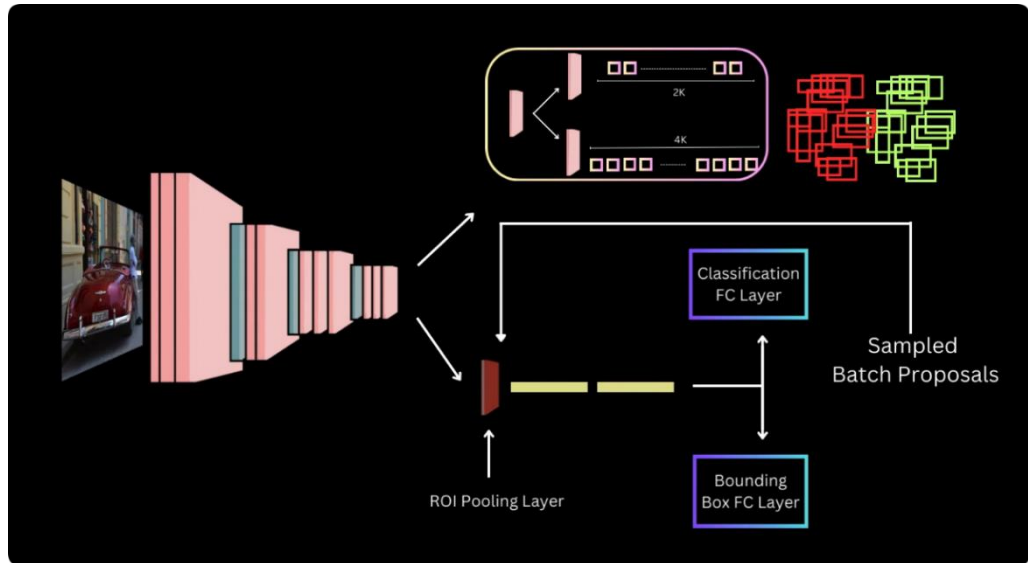
Cho đến phiên bản mới nhất là Fast R-CNN, mô hình nhận diện đối tượng này đã đạt được tốc độ gần như thời gian thực bằng cách sử dụng các mạng nơ-ron rất sâu, khi không tính đến thời gian dùng để tạo ra các vùng đề xuất. Lúc bấy giờ, các vùng đề xuất là điểm nghẽn then chốt trong quá trình tính toán tại thời điểm kiểm thử của những hệ thống nhận diện đối tượng tiên tiến nhất [1]. Vì vậy, mô hình Faster R-CNN được ra đời để cải tiến điều này.

Năm 2015, Shaoqing Ren, Kaiming He, Ross Girshick và Jian Sun đã giới thiệu mô hình Faster R-CNN tại hội nghị NeurIPS, một bước đột phá trong lĩnh vực phát hiện và nhận diện đối tượng qua hình ảnh. Mô hình này được phát triển nhằm giải quyết vấn đề hiệu suất của các mô hình nhận diện đối tượng trước đó. Cốt lõi của mô hình Faster R-CNN nằm ở sự thay đổi thuật toán tạo ra vùng đề xuất. Thay vì sử dụng Selective Search, một thuật toán dựa trên nguyên lý đồ thị để tạo ra các vùng đề xuất trong mô hình Fast R-CNN, họ đã thay thế bằng một mạng nơ-ron hoàn toàn mới.

Các tác giả của mô hình đã nhận thấy rằng bản đồ đặc trưng được sử dụng bởi các mô hình nhận diện dựa trên vùng cũng có thể dùng để tạo vùng đề xuất. Qua đó, họ đã tiến hành xây dựng một mạng nơ-ron mới, RPN (Region Proposal Network), với các lớp tích chập bổ sung mà đồng thời có thể phân loại và tinh chỉnh các vùng đề xuất [1]. Nhờ việc sử dụng RPN, mô hình Faster R-CNN đã có thể tạo ra các vùng đề xuất nhanh hơn nhiều so với một trong các phương pháp phổ biến nhất tại thời điểm đó là Selective Search, đồng thời vẫn đảm bảo được độ chính xác cao trong việc nhận diện đối tượng trong các bối cảnh phức tạp [1]. Điều này được xem là một thành tựu to lớn giúp Faster R-CNN có thể cạnh tranh với các mô hình tiên tiến khác vào thời điểm đó.

Mô hình Faster R-CNN đã chứng tỏ khả năng vượt trội trong các cuộc thi và ứng dụng thực tế. Các kiến trúc của RPN và Faster R-CNN đã được áp dụng và mở rộng sang nhiều phương pháp khác nhau như phát hiện đối tượng 3D, phát hiện dựa trên thành phần, phân đoạn đối tượng, và tạo chú thích ảnh. Bên cạnh đó, hệ thống phát hiện đối tượng nhanh và hiệu quả này đã được sử dụng trong các hệ thống thương mại như Pinterest, nơi nó đã cải thiện sự tương tác của người dùng. Ngoài ra, tại các cuộc thi nổi tiếng như ILSVRC và COCO năm 2015, Faster R-CNN và RPN là nền tảng cho nhiều bài dự thi giành được giải nhất ở nhiều hạng mục như ImageNet Detection, ImageNet Localization, COCO Detection, và COCO Segmentation [1].

### 1.1.a. Kiến trúc mô hình



Hình 1.1.1.a.1 Kiến trúc mô hình Faster R-CNN

Kiến trúc mô hình Faster R-CNN giúp tối ưu hóa quá trình nhận diện đối tượng bằng cách tích hợp cả bước trích xuất đặc trưng và tạo vùng đề xuất vào một mạng duy nhất. Điều này không chỉ hỗ trợ cải thiện hiệu suất tính toán của mô hình mà còn nâng cao độ hiệu quả trong việc sinh ra các vùng đề xuất so với các mô hình trước đó. Tổng quan, Faster R-CNN bao gồm 3 thành phần chính:

- **Mạng trích xuất đặc trưng (Feature Extraction Network):** Đây là một mạng nơ-ron tích chập có nhiệm vụ trích xuất các đặc trưng từ ảnh đầu vào và đầu ra là bản đồ đặc trưng. Điểm chung của các mô hình thuộc dòng R-CNN là việc ứng dụng kỹ thuật Transfer Learning để tận dụng các mô hình nổi tiếng trong việc trích xuất đặc trưng ảnh như VGG16, ResNet, v.v để làm Feature Extractor.
- **Mạng đề xuất vùng (Region Proposal Network):** Đây là một mạng nơ-ron tích chập nhỏ nằm sau mạng trích xuất đặc trưng, có nhiệm vụ tạo ra các vùng đề xuất một cách có định hướng với đầu vào là bản đồ đặc trưng. Là điểm cải tiến của mô hình Faster R-CNN, RPN không những có thể tạo ra các vùng đề xuất một cách nhanh chóng mà còn ngày càng cải thiện tính chính xác của các vùng đề xuất thông qua việc học từ dữ liệu, một trong những đặc trưng của mô hình máy học.
- **Mạng nhận diện đối tượng (Object Detection Network):** Đây là mạng nhận diện đối tượng được giữ nguyên từ mô hình Fast R-CNN, nằm ở phần cuối của kiến trúc, có nhiệm vụ phân loại hình ảnh và tinh chỉnh các hộp giới hạn từ đầu vào là sự kết hợp giữa bản đồ đặc trưng và các vùng đề xuất tạo ra bởi RPN thông qua một tầng trung gian đặc biệt là RoI (Region of Interest) Pooling.

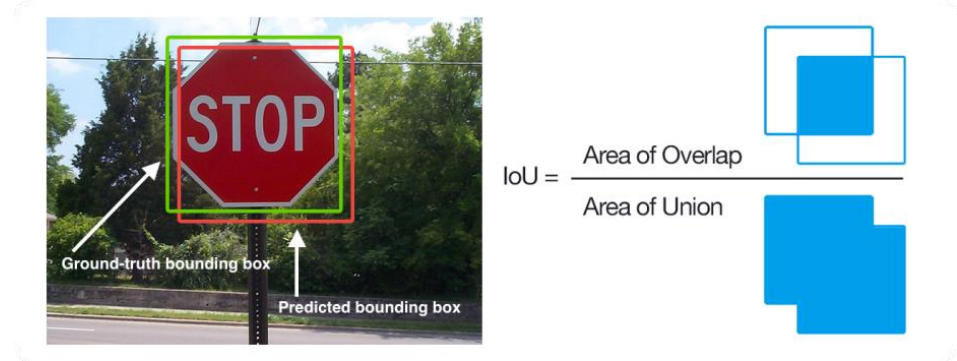
**1.1.b. Cơ chế hoạt động**

- *Trích xuất đặc trưng*
  - *Đầu vào:* Hình ảnh cần được nhận diện.
  - *Đầu ra:* Bản đồ đặc trưng.
  - *Xử lý:* Ứng dụng kỹ thuật Transfer Learning, hình ảnh đầu vào được đưa qua một mạng nơ-ron tích chập đã được huấn luyện sẵn. Mạng này được dùng làm Feature Extractor bằng cách chỉ giữ lại các tầng tích chập để trích xuất các đặc trưng quan trọng và loại bỏ các tầng kết nối đầy đủ. Đối với bài toán xử lý ảnh, việc sử dụng mạng huấn luyện sẵn như VGG16, ResNet, v.v. không chỉ giúp tiết kiệm thời gian và tài nguyên mà còn mang đến kết quả tốt hơn so với việc tự xây dựng từ đầu.
- *Tạo vùng đề xuất*
  - *Đầu vào:* Bản đồ đặc trưng.
  - *Đầu ra:* Tập hợp các vùng đề xuất gồm xác suất dự đoán đối tượng và tọa độ của vùng đề xuất.
  - *Xử lý:* Bản đồ đặc trưng được đưa qua RPN, mạng này là một mạng nơ-ron tích chập quy mô nhỏ. RPN hoạt động bằng cách trượt một ma trận kernel trên bản đồ đặc trưng đầu vào, tại mỗi vị trí kernel trượt qua, nó sử dụng các anchors, những hộp giới hạn cố định với nhiều kích thước và tỉ lệ khác nhau, để tạo ra vùng đề xuất. Sau đó, mỗi vùng đề xuất được phân thành 2 loại là chứa hoặc không chứa đối tượng, đồng thời được tinh chỉnh lại vị trí và kích thước để phù hợp hơn với các đối tượng được dự đoán trong ảnh.
- *Nhận diện đối tượng*
  - *Đầu vào:* Bản đồ đặc trưng và các vùng đề xuất.
  - *Đầu ra:* Các tập hợp chứa thông tin nhận diện.
  - *Xử lý:* Dựa vào thông tin của các vùng đề xuất sinh ra bởi RPN, mô hình thực hiện lấy ra các vùng đặc trưng tương ứng trên bản đồ đặc trưng. Tiếp theo, các vùng đặc trưng được đưa qua tầng RoI Pooling để chuẩn hóa về một kích thước cố định. Sau đó, các vùng đặc trưng được phân loại đối tượng tương ứng và tinh chỉnh lại vị trí và kích thước của hộp giới hạn để phù hợp hơn với các đối tượng được dự đoán. Cuối cùng, thuật toán NMS (Non-Maximum Suppression) là một kỹ thuật xử lý hậu kỳ giúp loại bỏ những hộp giới hạn dư thừa, các hộp giới hạn có độ trùng lặp cao với nhau nhưng có độ tin cậy thấp.

### 1.1.c. Chi tiết kỹ thuật

#### 1.1.c.1. Intersection over Union

*IoU* là một giá trị đo lường thể hiện mức độ trùng khớp giữa 2 hộp giới hạn, được dùng trong bài toán nhận diện đối tượng để đánh giá khả năng định vị chính xác giữa hộp giới hạn dự đoán và hộp giới hạn thực của đối tượng trong quá trình huấn luyện.



Hình 1.1.1.c.1.1 Hình ảnh minh họa thuật toán *IoU*

Công thức tính *IoU* yêu cầu đầu vào gồm 2 hộp giới hạn, mỗi hộp giới hạn được đại diện bởi một tập hợp tọa độ thể hiện vị trí của chúng trên bản đồ đặc trưng, biểu diễn thông qua 2 điểm tọa độ trên cùng bên trái và dưới cùng bên phải.

$$B_a = (x_{a1}, y_{a1}, x_{a2}, y_{a2})$$

$$B_b = (x_{b1}, y_{b1}, x_{b2}, y_{b2})$$

Đầu tiên, từ thông tin tọa độ, tính toán thông tin diện tích gồm  $S_a$  và  $S_b$  là diện tích các hộp giới hạn, và tọa độ phần giao nhau giữa 2 hộp giới hạn, gồm  $(x_{left}, y_{top}, x_{right}, y_{bottom})$ .

$$S_a = (x_{a2} - x_{a1}) \times (y_{a2} - y_{a1})$$

$$S_b = (x_{b2} - x_{b1}) \times (y_{b2} - y_{b1})$$

$$x_{left} = \max(x_{a1}, x_{b1})$$

$$x_{right} = \min(x_{a2}, x_{b2})$$

$$y_{top} = \max(y_{a1}, y_{b1})$$

$$y_{bottom} = \min(y_{a2}, y_{b2})$$

Tiếp theo, từ các thông tin diện tích, tính toán các thông tin diện tích gồm  $S_{in}$  là diện tích phần hợp và  $S_{un}$  là diện tích phần giao giữa 2 hộp giới hạn, sau cùng tính giá trị *IoU* từ các thông tin trên.

$$S_{in} = (x_{right} - x_{left}) \times (y_{bottom} - y_{top})$$

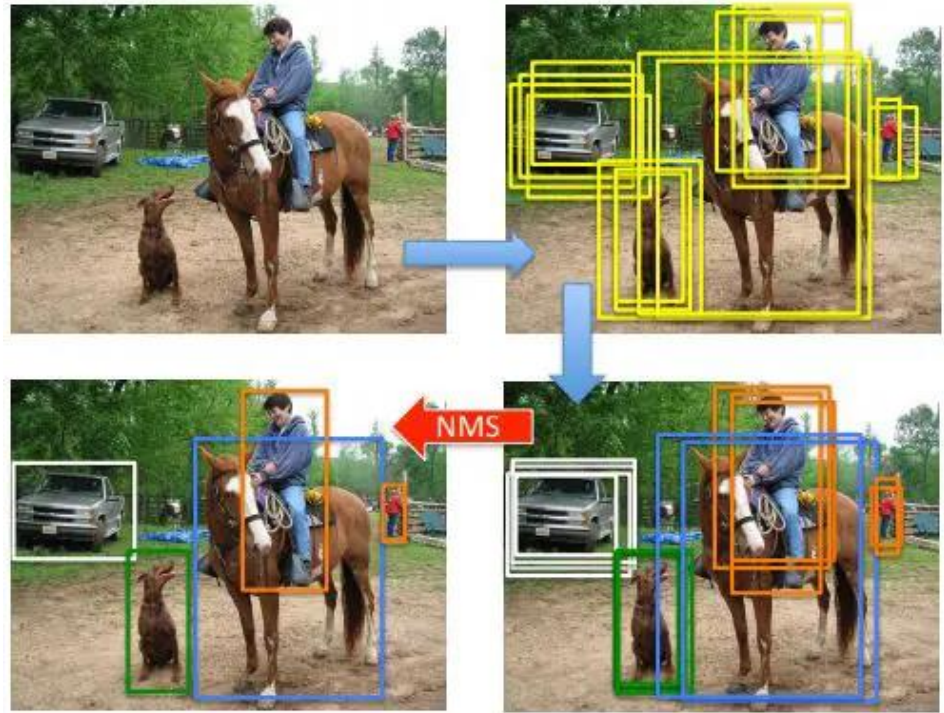
$$S_{un} = S_a + S_b - S_{in}$$

$$IoU = \frac{S_{in}}{S_{un}} \quad (1)$$



## 1.1.c.2. Non-Maximum Suppression

NMS là một thuật toán xử lý hậu kỳ được dùng phổ biến trong các hệ thống nhận diện và phát hiện đối tượng, đặc biệt là trong các mô hình thuộc dòng R-CNN. Kỹ thuật này giúp loại bỏ các hộp giới hạn dư thừa, chỉ giữ lại các hộp giới hạn có xác suất cao nhất đại diện cho một đối tượng cụ thể. Điều này giúp cải thiện tốc độ xử lý và độ chính xác của mô hình [2].



Hình 1.1.1.c.2.1 Hình ảnh minh họa thuật toán NMS

Thuật toán NMS yêu cầu đầu vào là: giá trị ngưỡng *threshold* và tập hợp các hộp giới hạn dự đoán  $[[x_1, x_2, x_3, x_4, score], \dots]$ , mỗi phần tử là một tập hợp các giá trị gồm  $(x_1, x_2)$  là tọa độ tâm và  $(x_3, x_4)$  chiều rộng và chiều dài của hộp giới hạn, và *score* là xác suất dự đoán chứa đối tượng. Thuật toán được thực hiện trên mỗi lớp đối tượng riêng biệt, gồm các bước được thực hiện như sau:

Bước 1: Sắp xếp các phần tử trong tập hợp theo giá trị *score*.

Bước 2: Lưu trữ phần tử có giá trị *score* cao nhất vào một tập hợp đầu ra và loại bỏ phần tử đó ra khỏi tập hợp đầu ra.

Bước 3: Loại bỏ các phần tử trong tập hợp đầu vào có mức trùng lặp so với phần tử vừa được lưu trữ cao hơn giá trị *threshold* đã cho. Mức trùng lặp được tính thông qua giá trị *IoU*.

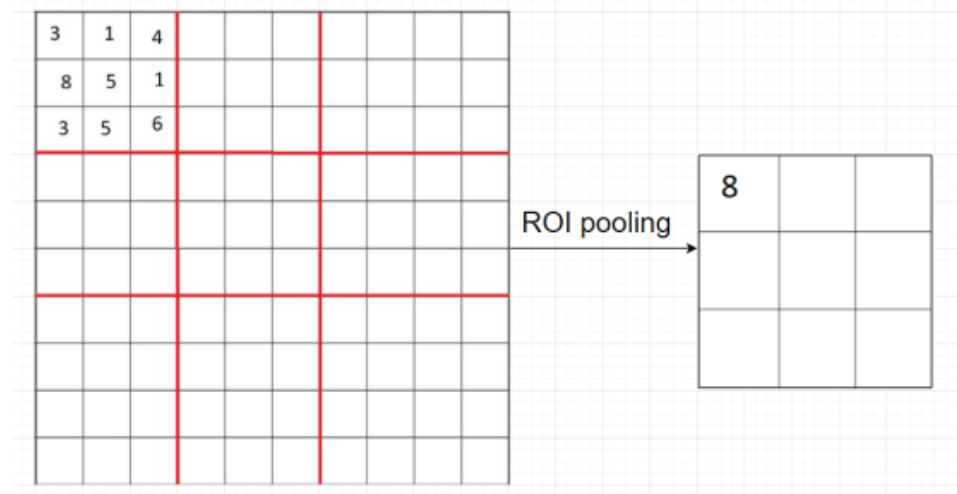
Bước 4: Lặp lại các bước trên cho đến khi tập hợp đầu vào rỗng.

Bước 5: Trả về danh sách các phần tử có giá trị *score* cao nhất tương ứng với mỗi lớp đối tượng.



### 1.1.c.3. Region of Interest

RoI là một dạng Pooling Layer được phát triển trong giai đoạn xây dựng mô hình Fast R-CNN với mục đích chuẩn hóa kích thước các Region Proposal sinh ra bởi thuật toán Selective Search, sau này là mạng RPN ở mô hình Faster R-CNN. Điểm khác biệt so với Max Pooling hay Average Pooling là bất kể kích thước của ma trận đầu vào thì RoI Pooling luôn cho đầu ra với kích thước cố định được định nghĩa trước [3].



Hình 1.1.1.c.3.1 Hình ảnh minh họa thuật toán RoI Pooling

Giả sử đầu vào là một ma trận có kích thước  $m \times n$  ( $10 \times 10$ ) và đầu ra là ma trận có kích thước  $h \times k$  ( $3 \times 3$ ) như hình trên.

Ta chia chiều rộng ma trận đầu vào thành  $h$  phần, gồm  $h - 1$  phần có kích thước  $\lfloor m \div h \rfloor$  và phần cuối có kích thước  $\lfloor m \div h \rfloor + m \% h$ . Tương tự ta chia chiều dài thành  $k$  phần, gồm  $k - 1$  phần có kích thước  $\lfloor n \div k \rfloor$ , phần cuối có kích thước  $\lfloor n \div k \rfloor + n \% k$ .

$$\begin{aligned} Width &= (3 - 1) \times (10 \div 3) + 1 \times (10 \div 3 + 10 \% 3) \\ &= 2 \times 3 + 1 \times (3 + 1) = 2 \times 3 + 1 \times 4 \end{aligned}$$

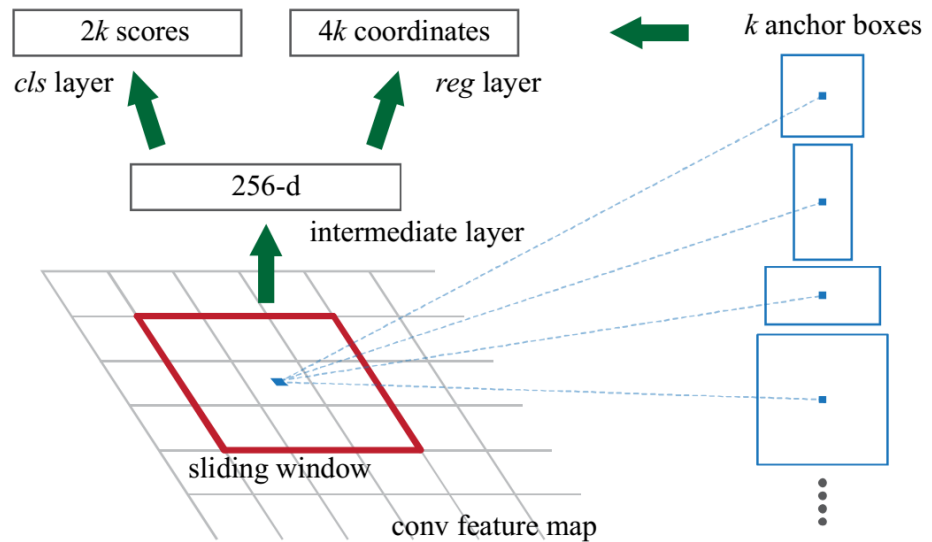
$$\begin{aligned} Height &= (3 - 1) \times (10 \div 3) + 1 \times (10 \div 3 + 10 \% 3) \\ &= 2 \times 3 + 1 \times (3 + 1) = 2 \times 3 + 1 \times 4 \end{aligned}$$

Từ đó, ta được chiều rộng có 3 phần, gồm 2 phần có kích thước 3 và 1 phần có kích thước 4, tương tự với chiều cao. Cuối cùng, với mỗi khối được tạo ra bằng các đường đỏ như hình minh họa, thực hiện phép Max Pooling để lấy ra từng giá trị cho ma trận đầu ra. Giả sử  $Part$  là phần ma trận được chia ra bởi thuật toán, mỗi phần lúc này được xem xét như một phần tử riêng biệt. Khi đó, từng phần tử trong ma trận đầu ra được tính toán như sau:

$$Output_{i \times i} = \max(Part_{i \times i}) \quad (2)$$

#### 1.1.c.4. Region Proposal Network

RPN là một mạng nơ-ron tích chập được sử dụng để sinh ra các vùng đề xuất một cách có định hướng. RPN nhận đầu vào là một hình ảnh có kích thước bất kỳ với đầu ra là một tập hợp các đề xuất vùng có dạng hình chữ nhật, mỗi vùng gồm 4 giá trị tọa độ cho biết vị trí của chúng trên bản đồ đặc trưng và 2 giá trị xác suất dự đoán cho biết vùng này có chứa đối tượng hay không [1].



Hình 1.1.1.c.4.1 Hình ảnh minh họa Region Proposal Network

Mạng RPN hoạt động theo quy luật cửa sổ trượt, bằng cách trượt mạng này trên bản đồ đặc trưng với ma trận kernel kích thước  $n \times n$ . Tại mỗi vùng đặc trưng được trượt qua, RPN thực hiện trích xuất đặc trưng để phân loại và tinh chỉnh các vùng đề xuất sao cho khớp với đối tượng được dự đoán [1]. Cuối cùng, được một tập hợp các vùng đề xuất đã được phân loại và tinh chỉnh.

Ý tưởng của mạng RPN là thay vì dự đoán tọa độ ở 2 góc, mô hình sẽ dự đoán điểm trung tâm, chiều rộng và chiều dài của hình chữ nhật. Qua đó, thuật ngữ anchors đã ra đời để biểu diễn các vùng đề xuất sinh ra bởi mạng RPN.

Trong quá trình thực nghiệm, các tác giả của mô hình đã áp dụng 3 kích thước ( $128 \times 128$ ,  $256 \times 256$  và  $512 \times 512$ ) và 3 tỉ lệ (1 : 1, 1 : 2 và 2 : 1) khác nhau. Từ đó, sinh ra 9 anchors, mỗi anchor đều lấy vị trí trung tâm của ma trận kernel làm tọa độ trung tâm để hình thành anchor box, tại mỗi vị trí mà RPN trượt qua trên bản đồ đặc trưng [1]. Như vậy, đối với một bản đồ đặc trưng có kích thước  $W \times H$ , thì tổng cộng sẽ có  $W \times H \times k$  anchor boxes khi lược bỏ đi khoảng cách giữa các anchors.



Hình 1.1.1.c.4.2 Hình ảnh minh họa kỹ thuật Anchors – 1

Giả sử hình ảnh minh họa trên có một anchor với tâm ở giữa ảnh, ta sẽ có 9 hình chữ nhật khác nhau được tổ hợp từ 3 kích thước và 3 tỉ lệ, tương ứng với các màu sắc là: màu da cam ( $128 \times 128$ ), màu xanh lục ( $256 \times 256$ ) và màu xanh lam ( $512 \times 512$ ).



Hình 1.1.1.c.4.3 Hình ảnh minh họa kỹ thuật Anchors – 2

Giả sử hình ảnh minh họa trên có kích thước  $400 \times 600 \text{ pixel}$ , các tâm anchors cách nhau  $16 \text{ pixel}$  thì sẽ có khoảng  $(400 \times 600) \div (16 \times 16) \approx 938$  tâm, và kết hợp với 9 anchors sinh ra ở mỗi tâm thì ta sẽ được  $938 \times 9 = 8442 \text{ anchor boxes}$ . Tuy nhiên, sau RPN, do có rất nhiều anchor box nằm chồng chéo lên nhau nên sẽ được loại bỏ bớt đi bằng thuật toán NMS.

## 1.2. MÔ HÌNH YOLO

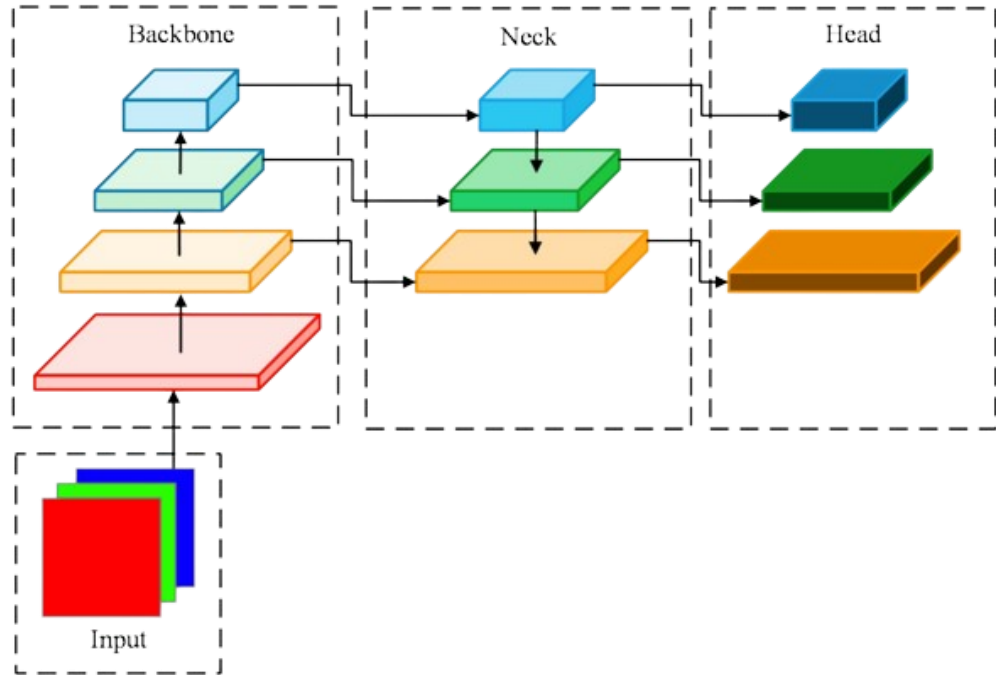
Trong những năm gần đây, YOLO đã trở thành một trong những mô hình hàng đầu trong lĩnh vực nhận diện đối tượng thời gian thực, nhờ vào sự cân bằng hiệu quả giữa chi phí tính toán và hiệu suất nhận diện. Các nhà nghiên cứu đã khám phá nhiều thiết kế kiến trúc, mục tiêu tối ưu hóa, chiến lược tăng cường dữ liệu và các khía cạnh khác, đem lại những tiến bộ đáng kể cho YOLO. Tuy nhiên, sự phụ thuộc vào thuật toán NMS để xử lý hậu kỳ đã gây khó khăn trong việc triển khai mô hình và ảnh hưởng đến độ trễ trong quá trình nhận diện. Ngoài ra, thiết kế của các thành phần trong YOLO thiếu sự kiểm tra toàn diện và kỹ lưỡng, dẫn đến sự dư thừa tính toán đáng kể và giới hạn khả năng của mô hình. Điều này làm cho hiệu suất của YOLO chưa được tối ưu, mặc dù có tiềm năng cải thiện đáng kể [4].

Tại hội nghị CVPR vào năm 2024, các nhà nghiên cứu đã giới thiệu mô hình YOLOv10, đánh dấu một bước tiến lớn trong lĩnh vực phát hiện và nhận diện đối tượng qua hình ảnh. Mô hình này được phát triển nhằm giải quyết các thách thức về hiệu suất và độ chính xác mà các phiên bản YOLO trước đây gặp phải. Cốt lõi của YOLOv10 nằm ở sự thay đổi trong kiến trúc và các phương pháp tối ưu hóa. Thay vì tuân theo các phương pháp truyền thống, mô hình đã áp dụng một loạt các kỹ thuật học sâu tiên tiến và cải tiến quá trình xử lý hậu kỳ, từ đó cải thiện đáng kể độ chính xác và hiệu suất của quá trình nhận diện đối tượng.

Các nhà nghiên cứu đã nhận thấy sự dư thừa trong quá trình tính toán của các phiên bản YOLO trước đây, do đó đã tiến hành cải tiến để phát triển phiên bản mới là YOLOv10. Mô hình này nhằm giải quyết cả những thiếu sót về kiến trúc và xử lý hậu kỳ được phát hiện trong quá trình thử nghiệm các phiên bản trước đó. Bằng cách loại bỏ triệt để thuật toán NMS và tối ưu hóa các thành phần mô hình khác nhau, YOLOv10 đã đạt được hiệu suất tối ưu với chi phí tính toán giảm đáng kể [4].

YOLOv10 đã được thử nghiệm một cách rộng rãi trên các tiêu chuẩn chất lượng như COCO, và đã thể hiện sự vượt trội về hiệu suất và hiệu quả. Mô hình đã đạt được kết quả khả quan trên nhiều biến thể khác nhau, chứng tỏ những cải tiến đáng kể cả về độ trễ và độ chính xác so với các phiên bản trước và các mô hình hiện đại khác. Nhờ vào những thay đổi này, YOLOv10 đã đạt được nhiều thành tựu khả quan trong cả ứng dụng thực tế và các cuộc thi. Mô hình đã xuất sắc đứng đầu trong nhiều cuộc thi quốc tế về phát hiện đối tượng như COCO và PASCAL VOC, và đã chứng minh khả năng phát hiện đối tượng nhanh chóng và chính xác, vượt trội hơn hẳn các phiên bản trước đó và nhiều đối thủ cạnh tranh khác.

### 1.2.a. Kiến trúc mô hình



Hình 1.1.2.a.1 Kiến trúc mô hình YOLO

Kiến trúc mô hình YOLO đã trải qua quá trình cải tiến liên tục để đạt được sự cân bằng hiệu quả giữa độ chính xác và hiệu suất. Nhờ những thay đổi mới này, phiên bản mới nhất của YOLO đã đạt được hiệu suất tối ưu và độ chính xác vượt trội khi nhận diện đối tượng theo thời gian thực. Tổng quan, kiến trúc YOLOv10 bao gồm 3 thành phần chính:

- **Mạng xương sống (Backbone):** Đây là một mạng nơ-ron tích chập có nhiệm vụ trích xuất các đặc trưng từ ảnh đầu vào và đầu ra là các đặc trưng được trích xuất từ nhiều mức độ phân giải khác nhau [5]. Trong YOLOv10, mô hình này sử dụng kiến trúc CSPNet (Cross Stage Partial Network) để cải thiện luồng gradient và giảm sự dư thừa tính toán.
- **Phần kết nối (Neck):** Đây là một phần quan trọng trong kiến trúc mạng tổng thể, được thiết kế để tổng hợp các đặc trưng đầu vào thuộc nhiều mức độ phân giải khác nhau và đầu ra là các đặc trưng đã được kết hợp và nâng cao. Phần này sử dụng PAN (Path Aggregation Network), một mạng nơ-ron kết hợp các đặc trưng đa tỉ lệ một cách hiệu quả [5].
- **Phần đầu ra (Head):** Đây là phần nhận diện đối tượng từ các đặc trưng tổng hợp từ phần kết nối, gồm 2 mạng nhận diện khác nhau được sử dụng cho 2 mục tiêu cụ thể, mạng OneToMany sử dụng trong quá trình huấn luyện và mạng OneToOne sử dụng trong quá trình kiểm thử, để giảm thiểu sự phụ thuộc vào các thuật toán xử lý hậu kỳ và nâng cao khả năng thực thi thời gian thực của mô hình.



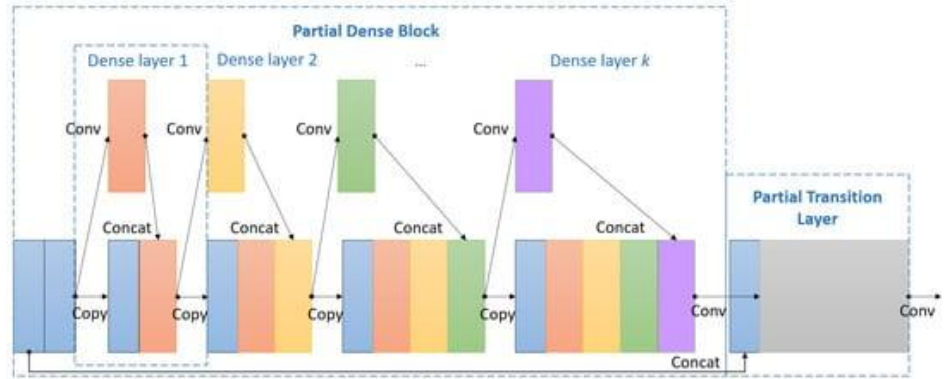
### 1.2.b. Cơ chế hoạt động

- *Trích xuất đặc trưng*
  - *Đầu vào:* Hình ảnh cần được nhận diện.
  - *Đầu ra:* 3 bản đồ đặc trưng được phân giải ở các mức khác nhau.
  - *Xử lý:* Ứng dụng kỹ thuật xử lý trong kiến trúc mạng CSPNet và kỹ thuật SCDD (Spatial–Channel Decoupled Downspampling) giúp trích xuất hiệu quả các đặc trưng quan trọng từ ảnh đầu vào. Mạng CSPNet chia quá trình trích xuất đặc trưng thành nhiều cụm xử lý. Trong mỗi cụm, bản đồ đặc trưng được tách thành 2 phần để xử lý và sau đó tổng hợp lại [6]. Qua đó, cải tiến quá trình xử lý bằng kỹ thuật SCDD bằng cách chia thành 2 giai đoạn gồm giảm kích thước không gian và tăng chiều sâu. Điều này giúp giảm chi phí tính toán và tối đa hóa việc giữ lại các thông tin quan trọng của ảnh [7].
- *Tổng hợp đặc trưng*
  - *Đầu vào:* 3 bản đồ đặc trưng được phân giải.
  - *Đầu ra:* 3 bản đồ đặc trưng được tổng hợp ở các mức khác nhau.
  - *Xử lý:* Ứng dụng kỹ thuật xử lý trong kiến trúc PAN, một mạng nơ-ron được cải tiến nhằm cải thiện khả năng tổng hợp đặc trưng bằng cách bổ sung các đường kết nối tắt nhằm tối ưu hóa dòng thông tin từ tầng thấp lên tầng cao. Cụ thể, trong quá trình nghiên cứu, nhận thấy rằng đặc trưng ở các mức thấp hữu ích cho việc nhận diện các đối tượng lớn, tuy nhiên, lượng thông tin sẽ mất dần trong quá trình xử lý, gây khó khăn trong việc truy cập thông tin định vị chính xác [8]. Như vậy, kiến trúc này là một chiến lược tối ưu để giảm thiểu mất mát thông tin trong quá trình tổng hợp đặc trưng.
- *Nhận diện đối tượng*
  - *Đầu vào:* 3 bản đồ đặc trưng được tổng hợp.
  - *Đầu ra:* Các tập hợp chứa thông tin nhận diện.
  - *Xử lý:* Dựa vào các bản đồ đặc trưng được tổng hợp ở phần kết nối, phần đầu ra sẽ thực hiện dự đoán tập hợp các nhãn và hộp giới hạn sau khi bản đồ đặc trưng đã qua 2 bước xử lý gồm chia lưới, mỗi ô lưới là một vùng đặc trưng, và sinh anchors, mỗi anchor hình thành một anchor box có khả năng chứa đối tượng. Đồng thời, ứng dụng kỹ thuật cải tiến NMS–Free Training, YOLOv10 tận dụng các ưu điểm đến từ 2 phương pháp gán nhãn OneToMany và OneToOne, qua đó cải thiện độ chính xác trong quá trình huấn luyện và giảm chi phí tính toán trong quá trình dự đoán [7].

### 1.2.c. Chi tiết kỹ thuật

#### 1.2.c.1. Cross Stage Partial Network

CSPNet là một kiến trúc mạng CNN được thiết kế với mục đích đạt được sự kết hợp gradient chất lượng hơn trong khi giảm lượng tính toán của mô hình. Mục tiêu này đạt được bằng cách phân chia bản đồ đặc trưng của lớp cơ sở thành 2 phần và sau đó kết hợp chúng lại thông qua một kiến trúc phân cấp xuyên tầng [6].



Hình 1.1.2.c.1.1 Hình ảnh minh họa kiến trúc CSPNet

Theo các tác giả của YOLOv10, cốt lõi của CSPNet là làm cho luồng gradient lan truyền qua các đường khác nhau trong mạng nơ-ron bằng cách phân tách luồng gradient. Bằng cách này, họ đã khẳng định rằng thông tin gradient được truyền có thể có sự khác biệt lớn về tương quan bằng cách chuyển đổi các bước nối kết và chuyển tiếp. Thêm vào đó, CSPNet có thể giảm đáng kể lượng tính toán, cải thiện tốc độ suy luận cũng như độ chính xác của mô hình [6].

Mỗi giai đoạn trong kiến trúc CSPNet bao gồm một khối Partial Dense và một tầng Partial Transition. Trong một khối Partial Dense, bản đồ đặc trưng của tầng cơ sở được chia thành 2 phần dựa trên giá trị kênh  $x_0 = [x'_0, x''_0]$ . Qua đó, phần đầu tiên sẽ được liên kết trực tiếp với cuối giai đoạn, trong khi phần còn lại sẽ đi qua khối Dense, và sau cùng sẽ được kết hợp lại ở tầng Partial Transition [6].

$$\text{Dense Layer Output: } x_k = w_k * [x''_0, x_1, \dots, x_{k-1}]$$

Partial Dense Block Output undergo a Transition Layer:

$$x_T = w_T * [x'_0, x_1, \dots, x_k]$$

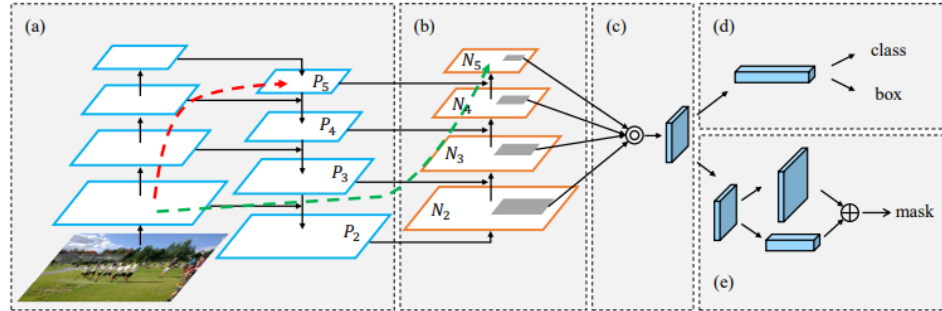
Partial Transition Layer Output:

$$x_U = w_U * [x'_0, x_T] \quad (3)$$

Ta có thể thấy rằng gradient đến từ các lớp Dense được tích hợp riêng biệt. Mặt khác, phần đầu bản đồ đặc trưng không đi qua các lớp Dense cũng được tích hợp riêng biệt. Qua đó, gradient ở cả hai phía đều không chứa thông tin trùng lặp của phía còn lại.

### 1.2.c.2. Path Aggregation Network

PAN là một mạng nơ-ron tích chập, được cải tiến dựa trên kiến trúc mạng FPN (Feature Pyramid Network), được thiết kế để tăng cường luồng thông tin trong quá trình tổng hợp đặc trưng, qua đó cải thiện chất lượng các bản đồ đặc trưng đầu vào để phân đầu ra có thể nhận diện một cách tốt nhất.



Hình 1.1.2.c.2.1 Hình ảnh minh họa kiến trúc PAN

Cụ thể, các tác giả của kiến trúc PAN thực hiện tăng cường toàn bộ hệ thống đặc trưng với các tín hiệu định vị chính xác ở các tầng thấp hơn bằng cách bổ sung đường dẫn từ dưới lên, điều này rút ngắn đường dẫn thông tin giữa các tầng thấp hơn và đặc trưng ở tầng cao nhất. Bên cạnh đó, họ cũng giới thiệu Adaptive Feature Pooling, tầng này liên kết lưới đặc trưng và đặc trưng ở mọi mức để làm cho thông tin trở nên hữu ích ở mỗi mức đặc trưng, lan truyền trực tiếp đến các mạng con đề xuất tiếp theo [8].

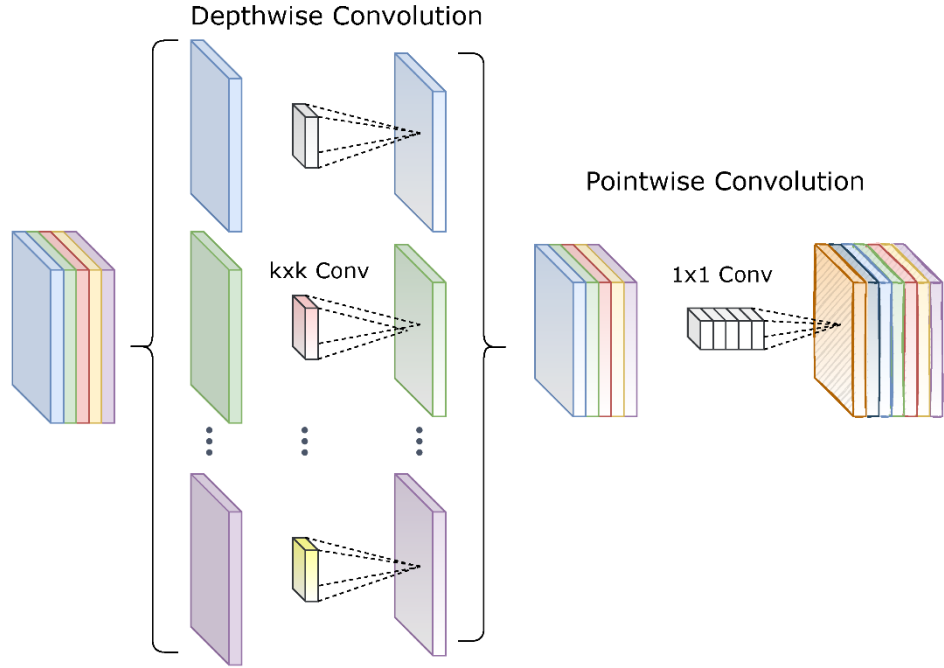
Trong kiến trúc FPN, thông tin đặc trưng chủ yếu di chuyển từ các tầng cao đến các tầng thấp, theo hướng từ trên xuống. Tuy nhiên, thông tin chi tiết và tín hiệu định vị chính xác thường nằm ở các tầng thấp. Qua đó, PAN đã cải tiến FPN bằng cách thêm một phần mạng gọi là Bottom-Up Path Augmentation, phần này bổ sung các kết nối từ các tầng thấp lên các tầng cao hơn, từ đó thông tin ở các tầng thấp có thể truyền trực tiếp lên các tầng cao hơn để cung cấp thông tin chi tiết hơn cho các tầng cao. Điều này đặc biệt hữu ích trong việc phát hiện các đối tượng nhỏ và các chi tiết phức tạp.

Bên cạnh đó, mạng PAN kết thúc quá trình tổng hợp đặc trưng với một tầng Adapter Feature Pooling, tầng này là một kỹ thuật quan trọng giúp kết nối và tích hợp thông tin từ các đặc trưng ở các cấp độ khác nhau. Nó đảm bảo rằng các đặc trưng từ các tầng khác nhau có kích thước đồng nhất, giúp cải thiện khả năng tổng hợp thông tin và nâng cao hiệu suất của mạng trong các tác vụ phát hiện và phân đoạn đối tượng.



### 1.2.c.3. Spatial–Channel Decoupled Downspampling

SCDD là một kỹ thuật hỗ trợ cho quá trình trích xuất đặc trưng, giúp giảm độ phức tạp tính toán và tối đa hóa thông tin trích xuất của quá trình xử lý, góp phần nâng cao hiệu suất nhận diện của mô hình và làm cho mô hình phù hợp hơn cho các ứng dụng yêu cầu tính toán nhanh và hiệu quả.



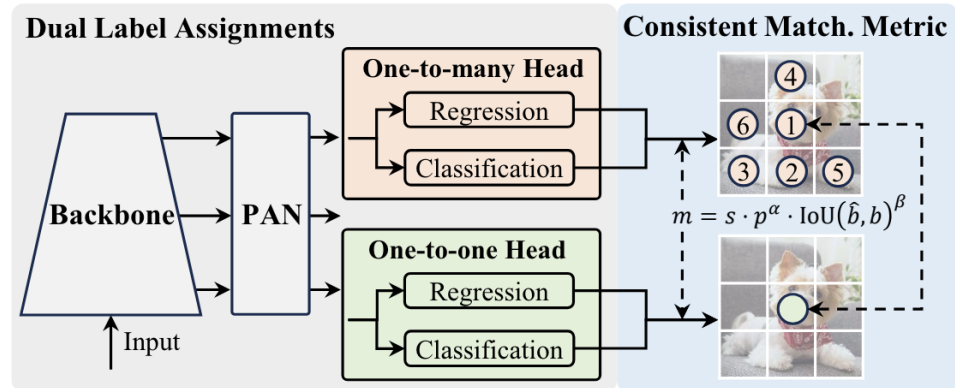
Hình 1.1.2.c.3.1 Hình ảnh minh họa thao tác Depthwise và Pointwise

Theo các tác giả của YOLOv10, việc tách biệt các thao tác giảm kích thước không gian và tăng chiều sâu, hỗ trợ trích xuất đặc trưng nhanh chóng và hiệu quả hơn so với các phương pháp truyền thống là thực hiện xen kẽ các thao tác trên. Với phương pháp truyền thống, mô hình sử dụng các phép tích chập tiêu chuẩn với ma trận  $3 \times 3$  và bước nhảy là 2 giúp giảm không gian từ  $W \times H$  thành  $\frac{W}{2} \times \frac{H}{2}$ , và phép biến đổi kênh giúp tăng chiều sâu từ  $C$  thành  $2C$ . Điều này ước tính độ phức tạp rất lớn, cụ thể với chi phí tính toán là  $O(\frac{9}{2}HWC^2)$  và số lượng tham số là  $O(18C^2)$  [4].

Vì vậy, các tác giả đã thay đổi quy trình xử lý đặc trưng thông qua việc đảo ngược và tách biệt quá trình xử lý như sau: tăng chiều sâu đầu tiên rồi mới giảm kích thước. Theo ước tính của họ, điều này giúp giảm chi phí tính toán xuống  $O(2HWC^2 + \frac{9}{2}HWC)$  và số lượng tham số xuống  $O(2C^2 + 18C)$ . Đồng thời, nó tối đa hóa việc giữ lại thông tin trong quá trình trích xuất đặc trưng, dẫn đến hiệu suất được tăng cường với giảm độ trễ [4].

#### 1.2.c.4. NMS-Free Training

NMS-Free Training là một chiến lược huấn luyện không dùng thuật toán NMS, NMS mặc dù là một thuật toán phổ biến được dùng trong các mô hình tiên tiến nhưng lại làm tăng đáng kể thời gian dự đoán dẫn đến không đạt được hiệu suất cao, cho các mô hình YOLO bằng việc gán nhãn kép và sử dụng độ đo mức độ khớp nhất quán để đạt được cả hiệu suất cao trong cả huấn luyện và dự đoán [4].



Hình 1.1.2.c.4.1 Hình ảnh minh họa chiến lược NMS-Free Training

Phương pháp gán nhãn kép trong NMS-Free Training sử dụng 2 phương pháp gán nhãn OneToMany và OneToOne nhằm tận dụng các ưu điểm để bù trừ các nhược điểm của 2 phương pháp này.

##### ○ Gán nhãn OneToMany

- **Ưu điểm:** Cung cấp nhiều thông tin tín hiệu giám sát, giúp mô hình học tốt hơn, tốc độ hội tụ nhanh hơn.
- **Nhược điểm:** Cần phải dùng NMS để xử lý sau huấn luyện, tốn thời gian triển khai và tính toán ở bước hậu xử lý.

##### ○ Gán nhãn OneToOne

- **Ưu điểm:** Tránh phải dùng NMS sau huấn luyện, giúp đơn giản hóa và tăng tốc độ triển khai mô hình.
- **Nhược điểm:** Cung cấp ít tín hiệu giám sát hơn, dẫn đến độ chính xác và tốc độ hội tụ không tối ưu.

Cách chiến lược NMS-Free Training vận hành trong quá trình huấn luyện gồm phương pháp gán nhãn kép sử dụng 2 đầu ra để tận dụng những ưu điểm của 2 phương pháp gán nhãn trên nhưng khi dự đoán thì chỉ sử dụng một đầu ra OneToOne để tránh sử dụng NMS, và độ đo mức độ khớp nhất quán là chỉ số đánh giá mức độ phù hợp giữa kết quả dự đoán và kết quả thực tế, giúp cải thiện độ chính xác và hiệu suất mô hình bằng cách duy trì sự nhất quán trong tất cả các dự đoán khi so sánh với các đối tượng thực tế trong ảnh.

### 1.3. SO SÁNH TỔNG QUAN

Dựa trên các lý thuyết nghiên cứu từ Faster R-CNN và YOLOv10, cả hai mô hình này đều là các kiến trúc nền tảng và là các mô hình phổ biến trong bài toán nhận diện đối tượng, nhưng giữa chúng có các đặc điểm và cách tiếp cận khác nhau về mặt kiến trúc dẫn đến sự khác biệt giữa hiệu suất và tốc độ.

	<i>Faster R-CNN</i>	<i>YOLO (v10)</i>
<i>Kiến trúc</i>	<p><i>Được thiết kế theo kiến trúc 2 giai đoạn:</i></p> <ul style="list-style-type: none"> <li><i>Mạng RPN: Sinh vùng các đề xuất.</i></li> <li><i>Mạng CNN: Phân loại và tinh chỉnh hộp giới hạn.</i></li> </ul>	<p><i>Được thiết kế theo kiến trúc 1 giai đoạn:</i></p> <ul style="list-style-type: none"> <li><i>Mạng CNN: Phân loại và tinh chỉnh hộp giới hạn trực tiếp trên các vùng đề xuất.</i></li> </ul>
<i>Độ chính xác</i>	<i>Do được tính toán chi tiết và tinh chỉnh kỹ lưỡng các vùng đề xuất trước khi truyền qua mạng CNN nên độ chính xác đạt được rất cao.</i>	<i>Do được thiết kế để đạt được tốc độ tối ưu bằng cách giảm độ phức tạp trong tính toán nên độ chính xác đạt được ở mức tương đối.</i>
<i>Tốc độ</i>	<i>Tốc độ chậm do các thao tác tính toán chi tiết và phức tạp.</i>	<i>Tốc độ rất nhanh do các thao tác tính toán được đơn giản hóa.</i>
<i>Ứng dụng</i>	<p><i>Các ứng dụng yêu cầu có độ chính xác cao.</i></p> <ul style="list-style-type: none"> <li><i>Y tế.</i></li> <li><i>An ninh.</i></li> <li><i>...</i></li> </ul>	<p><i>Các ứng dụng yêu cầu có tốc độ cao.</i></p> <ul style="list-style-type: none"> <li><i>Xe tự hành.</i></li> <li><i>Giám sát.</i></li> <li><i>...</i></li> </ul>

Tổng kết, Faster R-CNN và YOLO đều là các mô hình phổ biến hàng đầu trong lĩnh vực thị giác máy tính, cụ thể như phát hiện đối tượng. Mô hình Faster R-CNN nổi bật về độ chính xác và khả năng phân loại chi tiết, trong khi YOLO nổi bật về tốc độ và khả năng xử lý thời gian thực. Do đó, việc lựa chọn mô hình phù hợp phụ thuộc vào các yêu cầu cụ thể mà ứng dụng hướng đến, như độ chính xác hay tốc độ phản hồi.

## 2. TÌM HIỂU BÀI TOÁN DỰ BÁO CHÁY TRONG NHÀ

Bài toán dự báo cháy trong nhà là việc phát hiện sớm các dấu hiệu của một vụ cháy và cảnh báo kịp thời để ngăn chặn và giảm thiểu thiệt hại về người và tài sản. Mục tiêu của bài toán là nhận diện các dấu hiệu đặc trưng của vụ cháy như sự hiện diện của khói, lửa và sự gia tăng nhiệt độ nhanh chóng bất thường trong môi trường, đồng thời đưa ra cảnh báo sớm cho người dùng.

### 2.1. PHƯƠNG PHÁP

- *Cảm biến nhiệt và khói:* Đây là một phương pháp truyền thống và phổ biến nhất để phát hiện cháy. Cảm biến khói phát hiện sự hiện diện của các hạt khói trong không khí, thường sử dụng công nghệ ion hóa hoặc quang điện, trong khi cảm biến nhiệt phát hiện nhiệt độ cao hoặc sự gia tăng nhiệt độ bất thường, và thực hiện cảnh báo khi nhiệt độ vượt quá ngưỡng nhất định. Các cảm biến này thường được kết nối với hệ thống báo động để cảnh báo người trong nhà.
- *Thiết bị giám sát và phân tích hình ảnh:* Đây là một phương pháp truyền thống sử dụng hình ảnh trực quan. Thông qua việc kết hợp các thiết bị ghi hình giám sát với các thuật toán xử lý hình ảnh để phát hiện sự hiện diện của lửa hoặc khói. Các hệ thống này có thể phân tích hình ảnh trong thời gian thực và nhận diện các dấu hiệu của cháy, chẳng hạn như ánh sáng đặc trưng của ngọn lửa hay các đặc điểm của khói.
- *Trí tuệ nhân tạo:* Đây là một phương pháp hiện đại và tối ưu hóa khả năng cho các phương pháp truyền thống khi mà công nghệ trí tuệ nhân tạo đang phát triển mạnh mẽ. Sử dụng các mô hình máy học với khả năng tính toán ưu việt để phân tích dữ liệu từ các thiết bị ghi hình hoặc cảm biến nhằm phát hiện các dấu hiệu của cháy sớm hơn và chính xác hơn. Các mô hình này có thể học từ dữ liệu quá khứ để cải thiện khả năng dự báo.

### 2.2. THÁCH THỨC

- *Báo động giả:* Một trong những thách thức lớn nhất của bài toán dự báo cháy là giảm thiểu báo động sai. Điều này đòi hỏi các hệ thống phải phân biệt được các nguồn thông tin gây nhiễu (như ánh sáng từ các thiết bị điện tử hay khói thuốc, hơi nước, v.v.) với các dấu hiệu cháy thực sự.
- *Phát hiện sớm và chính xác:* Để đảm bảo an toàn tối đa, hệ thống cần phải phát hiện cháy ở giai đoạn sớm nhất có thể. Điều này đòi hỏi các hệ thống dự báo cháy phải nhạy bén và chính xác. Đây là một thành tựu khó đạt được đối với các phương pháp truyền thống vì chúng có độ trễ nhất định.
- *Khả năng mở rộng và tích hợp:* Hệ thống cần có thể tích hợp linh hoạt với các hệ thống an ninh khác. Điều này không chỉ hỗ trợ mở rộng phạm vi giám sát, tránh các điểm mù của hệ thống, mà còn tối ưu hóa các hiệu quả và tiện ích mà hệ thống dự báo cháy mang lại.

### 3. TÌM HIỂU TẬP DỮ LIỆU CẢNH BÁO CHÁY BẰNG HÌNH ẢNH

Đối với bài toán dự báo cháy sử dụng phương pháp trí tuệ nhân tạo thì nguồn dữ liệu huấn luyện là vô cùng quan trọng. Đặc biệt đối với mô hình đặc thù cho việc nhận diện đối tượng như YOLO, hình ảnh là nguồn dữ liệu không thể thay thế, không chỉ yêu cầu độ chính xác cao và đa dạng trong tập dữ liệu nhằm nâng cao khả năng nhận diện của mô hình mà còn phải tuân theo những tiêu chuẩn nhất định.

#### 3.1. PHÂN LOẠI DỮ LIỆU

- *Hình ảnh có cháy*: Bao gồm các hình ảnh với sự hiện diện rõ ràng của lửa hoặc khói. Các hình ảnh này cần có sự đa dạng về kích thước, màu sắc và độ sáng của ngọn lửa, cũng như bối cảnh khác nhau (ngoài trời, trong nhà, ban ngày, ban đêm, v.v.) để mô hình có nhận diện một cách chính xác.
- *Hình ảnh không có cháy*: Bao gồm các hình ảnh không chứa lửa hoặc khói, nhưng có thể chứa các nguồn sáng khác (ánh đèn, ánh sáng phản chiếu, v.v.) để huấn luyện mô hình phân biệt chính xác giữa đám cháy thực sự và các nguồn sáng khác.

#### 3.2. GÁN NHÃN DỮ LIỆU

- *Hộp giới hạn (Bounding Box)*: Tập giá trị tọa độ để xác định phạm vi của một khu vực chứa đối tượng trong ảnh. Định dạng của tập giá trị có thể khác nhau tùy theo tiêu chuẩn mà mô hình sử dụng. Trực quan hóa, một đường viền hình chữ nhật bao quanh khu vực có sự hiện diện của đám cháy trong hình ảnh.
- *Nhãn (Label)*: Giá trị đại diện ("fire" cho lửa, "smoke" cho khói, v.v.) để xác định rằng đó là một đối tượng cụ thể trong bối cảnh bài toán mà mô hình phải nhận diện trong phạm vi của hộp giới hạn.

#### 3.3. CHUẨN BỊ DỮ LIỆU

- *Tiền xử lý dữ liệu*: Các thao tác biến đổi dữ liệu như cân bằng màu sắc, chuẩn hóa kích thước, cắt ảnh, v.v., và các kỹ thuật tăng cường dữ liệu như xoay ảnh, lật ảnh, làm mịn ảnh, thay đổi độ sáng, v.v.
- *Phân chia dữ liệu*: Tập dữ liệu gốc được chia thành 3 tập dữ liệu con phục vụ cho từng mục tiêu cụ thể. Có rất nhiều cách phân chia dữ liệu và việc lựa chọn phương pháp phụ thuộc vào bản chất của dữ liệu và mục tiêu của bài toán.
  - *Tập huấn luyện (Training)*: Tập con chiếm tỉ lệ lớn nhất, được sử dụng để mô hình học các đặc trưng của đối tượng cần nhận diện.
  - *Tập kiểm tra (Validation)*: Tập con được sử dụng để đánh giá mô hình trong quá trình huấn luyện nhằm tối ưu hóa các siêu tham số và ngăn tình trạng quá khớp dữ liệu.
  - *Tập kiểm thử (Testing)*: Tập con được sử dụng để đánh giá hoàn toàn độc lập về hiệu suất của mô hình sau khi huấn luyện xong.



## CHƯƠNG 2. THỰC NGHIỆM VÀ SO SÁNH MÔ HÌNH FASTER R-CNN VÀ YOLO TRÊN CÙNG TẬP DỮ LIỆU

### 1. CHUẨN BỊ TẬP DỮ LIỆU

Trong phần nghiên cứu này, với mục đích đánh giá một cách toàn diện và trực quan về lý thuyết đã được trình bày, tôi sẽ thực hiện so sánh 2 mô hình Faster R-CNN và YOLO trên cùng một tập dữ liệu. Vì vậy, tôi đã tự chuẩn bị một tập dữ liệu bằng Roboflow, một nền tảng cung cấp các công cụ mạnh mẽ để quản lý và xử lý dữ liệu hình ảnh.

#### 1.1. THU THẬP HÌNH ẢNH

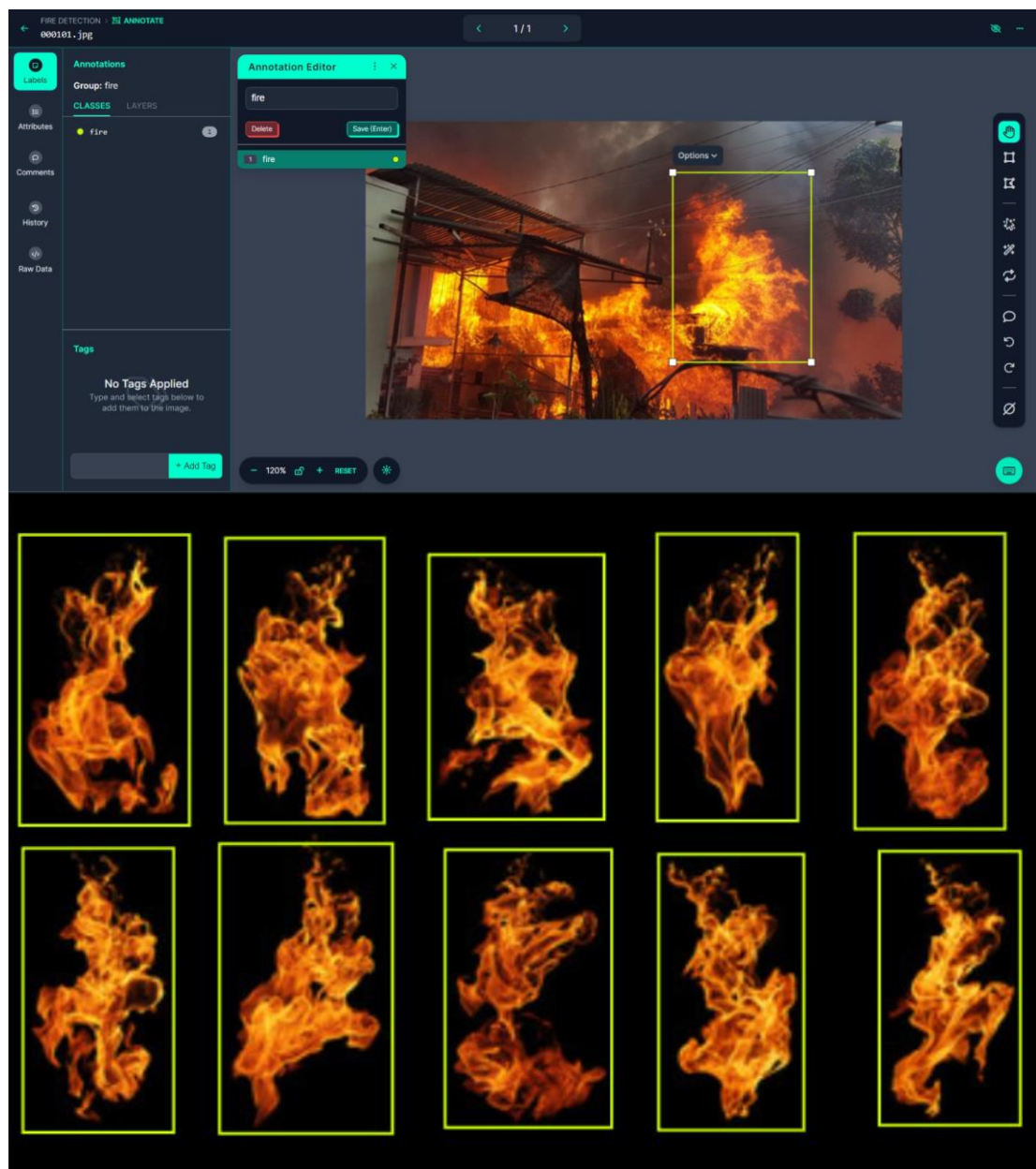
Tôi đã thu thập một lượng lớn hình ảnh từ nhiều nguồn khác nhau, bao gồm hình ảnh các ngọn lửa cụ thể và từ các vụ cháy rừng, cháy nhà, v.v. cũng như các hình ảnh gây nhiễu khác. Điều này đảm bảo rằng dữ liệu huấn luyện đủ phong phú và đa dạng, phản ánh được nhiều trường hợp thực tế. Tổng số lượng hình ảnh thu thập được là 1350.



Hình 2.1.1.1 Một số hình ảnh trong tập dữ liệu

## 1.2. GÁN NHÃN DỮ LIỆU

Sử dụng công cụ gán nhãn cung cấp bởi Roboflow, tôi đã thực hiện khoanh vùng các phần hình ảnh chứa đối tượng lửa bằng hộp giới hạn và thực hiện gán nhãn cho chúng. Quá trình này là rất quan trọng vì nó giúp mô hình học được các đặc trưng của lửa. Đối với bài toán nhận diện đối tượng, nhãn là một tập giá trị và có thể được định dạng theo các tiêu chuẩn khác nhau, cụ thể trong nghiên cứu này sử dụng định dạng COCO để huấn luyện Faster R-CNN và định dạng YOLOv9 để huấn luyện YOLOv10. Bên cạnh đó, Roboflow hỗ trợ đóng gói các hình ảnh vào các tập dữ liệu con. Qua đó, tập dữ liệu tổng thể được chia theo tỷ lệ 80 – 15 – 5, gồm 1080 mẫu trên tập huấn luyện, 203 mẫu trên tập kiểm tra và 67 mẫu trên tập kiểm thử.



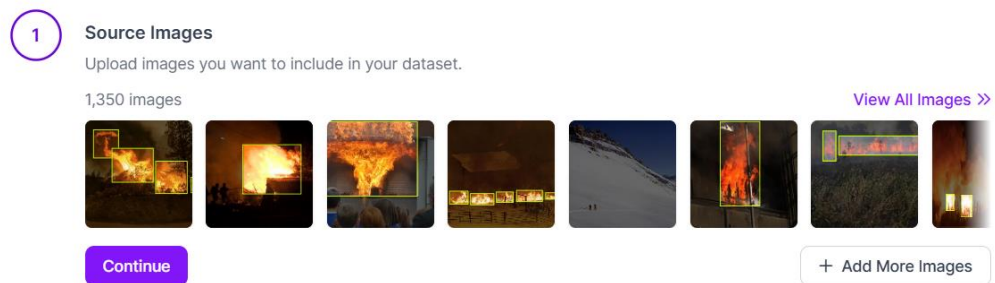
Hình 2.1.2.1 Hình ảnh minh họa quy trình gán nhãn Roboflow

### 1.3. TẠO TẬP DỮ LIỆU

Sau khi hoàn tất quá trình gán nhãn cho dữ liệu, tôi sử dụng công cụ mạnh mẽ do Roboflow cung cấp để tạo ra một tập dữ liệu phù hợp với mục tiêu nghiên cứu của bản thân. Quy trình này bao gồm 5 bước, đảm bảo rằng mỗi giai đoạn đều được thực hiện một cách cẩn thận và chính xác để tối ưu hóa hiệu quả của tập dữ liệu. Các bước chi tiết như sau:

**Bước 1:** *Lựa chọn hình ảnh đã gán nhãn cho tập dữ liệu.*

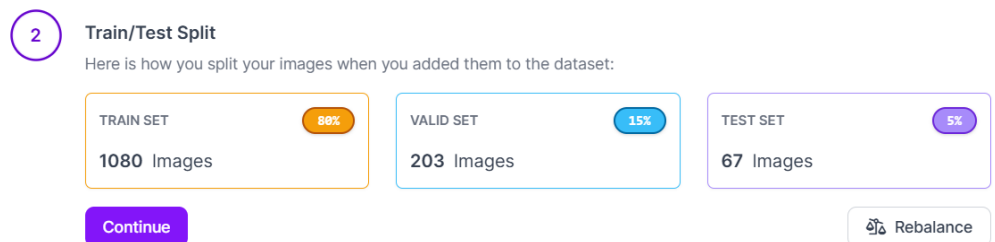
Đầu tiên, tôi cẩn thận xem xét và chọn lọc những hình ảnh đã được gán nhãn chính xác để đảm bảo rằng tập dữ liệu có thể phản ánh đầy đủ các trường hợp cần phát hiện. Việc lựa chọn một cách cẩn thận sẽ giúp tối ưu hóa chất lượng của tập dữ liệu.



Hình 2.1.3.1 Bước 1 trong quy trình tạo tập dữ liệu Roboflow

**Bước 2:** *Xác định tỷ lệ chia tách cho các tập dữ liệu con.*

Tiếp theo, tôi quyết định tỷ lệ chia tách tập dữ liệu tổng thể thành 3 tập dữ liệu con, gồm tập huấn luyện, tập kiểm tra và tập kiểm thử theo tỷ lệ 80 – 15 – 5. Thông thường, bước này sẽ không thay đổi nếu đã đóng gói khi gán nhãn dữ liệu. Sự phân chia hợp lý này đảm bảo rằng mô hình được huấn luyện hiệu quả và được đánh giá chính xác.

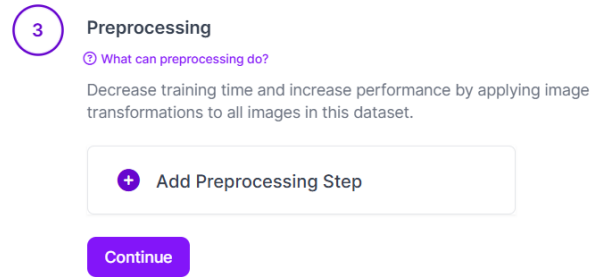


Hình 2.1.3.2 Bước 2 trong quy trình tạo tập dữ liệu Roboflow

**Bước 3:** *Áp dụng các thao tác tiền xử lý dữ liệu với Preprocessing Options.*

Trong bước này, tôi tận dụng các tùy chọn tiền xử lý của Roboflow để cải thiện chất lượng hình ảnh trong tập dữ liệu. Các thao tác như thay đổi kích thước, điều chỉnh độ sáng, và cân bằng màu sắc được áp dụng để đảm bảo dữ liệu đầu vào có chất lượng tốt nhất.

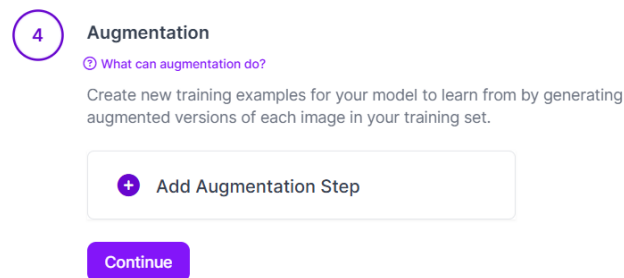




Hình 2.1.3.3 Bước 3 trong quy trình tạo tập dữ liệu Roboflow

**Bước 4:** Thực hiện các thao tác tăng cường dữ liệu với Augmentation Options.

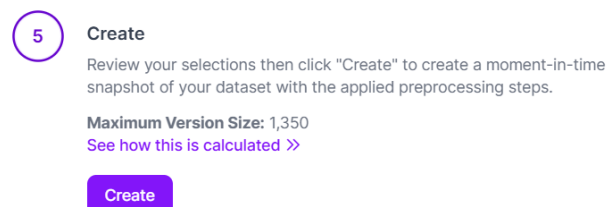
Trong bước này, Roboflow hỗ trợ các phương pháp tăng cường dữ liệu để làm phong phú và đa dạng hơn tập dữ liệu. Các thao tác như xoay ảnh, lật ảnh và thêm nhiễu giúp mô hình trở nên linh hoạt và mạnh mẽ hơn khi đối mặt với các tình huống thực tế đa dạng. Tuy nhiên, tôi không áp dụng phương pháp tăng cường dữ liệu vì sự giới hạn về tài nguyên, và tập dữ liệu gốc đã đủ để đánh giá phần lý thuyết 2 mô hình Faster R-CNN và YOLO.



Hình 2.1.3.4 Bước 4 trong quy trình tạo tập dữ liệu Roboflow

**Bước 5:** Xác nhận để Roboflow tiến hành tạo tập dữ liệu.

Cuối cùng, tôi xác nhận các thiết lập tạo tập dữ liệu. Roboflow sẽ tự động xử lý và chuẩn bị tập dữ liệu dựa trên các lựa chọn đã thực hiện ở các bước trước, cung cấp một tập dữ liệu hoàn chỉnh, sẵn sàng để sử dụng trong quá trình huấn luyện và đánh giá mô hình.



Hình 2.1.3.5 Bước 5 trong quy trình tạo tập dữ liệu Roboflow

## 2. HUẤN LUYỆN MÔ HÌNH FASTER R-CNN

### 1) Cài đặt môi trường



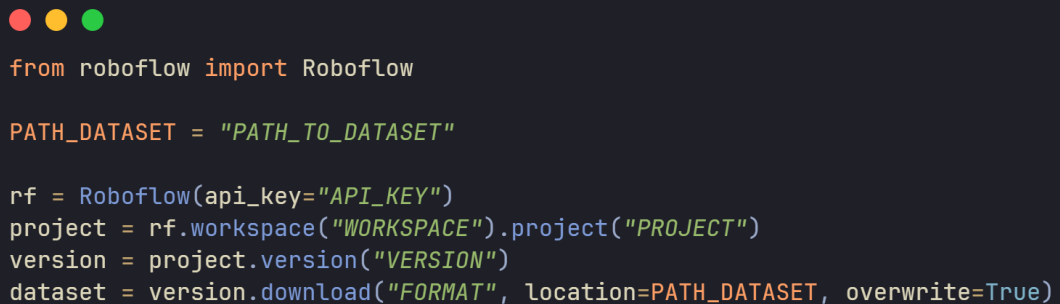
```
!pip install git+https://github.com/facebookresearch/detectron2.git
!pip install roboflow
```

Hình 2.2.1 Mã nguồn cài đặt môi trường Faster R-CNN

Đây là các lệnh thực thi tải xuống các gói thư viện cần thiết vào môi trường Python trên Google Colab, hoạt động ở chế độ dòng lệnh.

- `!pip install git + https://github.com/facebookresearch/detectron2.git`: Lệnh cài đặt tự động các thư viện cần thiết từ kho lưu trữ Github detectron2 để có thể xây dựng mô hình Faster R-CNN.
- `!pip install roboflow`: Lệnh cài đặt gói thư viện roboflow, một thư viện giúp dễ dàng tích hợp với Roboflow, một nền tảng cung cấp dịch vụ gán nhãn dữ liệu và huấn luyện mô hình máy học.

### 2) Tải tập dữ liệu



```
from roboflow import Roboflow

PATH_DATASET = "PATH_TO_DATASET"

rf = Roboflow(api_key="API_KEY")
project = rf.workspace("WORKSPACE").project("PROJECT")
version = project.version("VERSION")
dataset = version.download("FORMAT", location=PATH_DATASET, overwrite=True)
```

Hình 2.2.2 Mã nguồn tải tập dữ liệu từ Roboflow (COCO)

Đoạn mã Python sử dụng thư viện *roboflow* để thực thi tải xuống một tập dữ liệu từ Roboflow, định dạng mà thư viện *detectron2* yêu cầu là COCO. Các thông tin cần thiết để tải xuống bao gồm:

- `PATH_TO_DATASET`: Đường dẫn lưu trữ tập dữ liệu.
- `API_KEY`: Thông tin khóa API để truy cập vào các tài nguyên của Roboflow.
- `WORKSPACE`: Thông tin tên Workspace nơi chứa Project.
- `PROJECT`: Thông tin tên Project nơi chứa tập dữ liệu.
- `VERSION`: Thông tin phiên bản tập dữ liệu.
- `FORMAT`: Thông tin định dạng tập dữ liệu. Tùy chọn `overwrite = True` cho phép ghi đè dữ liệu nếu nó đã tồn tại trong đường dẫn thư mục lưu trữ.

```
from detectron2.data.datasets import register_coco_instances

ANN_TRAIN, IMG_TRAIN = f"{PATH_DATASET}/train/_annotations.coco.json", f"{PATH_DATASET}/train"
ANN_VAL, IMG_VAL = f"{PATH_DATASET}/valid/_annotations.coco.json", f"{PATH_DATASET}/valid"
ANN_TEST, IMG_TEST = f"{PATH_DATASET}/test/_annotations.coco.json", f"{PATH_DATASET}/test"

register_coco_instances("my_dataset_train", {}, ANN_TRAIN, IMG_TRAIN)
register_coco_instances("my_dataset_val", {}, ANN_VAL, IMG_VAL)
register_coco_instances("my_dataset_test", {}, ANN_TEST, IMG_TEST)
```

Hình 2.2.3 Mã nguồn đăng ký tập dữ liệu với đối tượng COCO

Đoạn mã Python này dùng hàm `register_coco_instances` để đăng ký các đường dẫn đến thư mục hình ảnh và tệp tin nhãn với thư viện `detectron2`. Thông qua các tên đã đăng ký, ta sẽ dùng chúng thay thế cho các đường dẫn trong suốt quá trình xây dựng mô hình.

- `PATH_DATASET`: Đường dẫn đến thư mục tập dữ liệu đã tải trước đó.
- `ANN_TRAIN, IMG_TRAIN`: Các đường dẫn đến dữ liệu huấn luyện.
- `ANN_TRAIN, IMG_TRAIN`: Các đường dẫn đến dữ liệu kiểm tra.
- `ANN_TRAIN, IMG_TRAIN`: Các đường dẫn đến dữ liệu kiểm thử.
- `my_dataset_train`: Tên đại diện cho tập dữ liệu huấn luyện.
- `my_dataset_val`: Tên đại diện cho tập dữ liệu kiểm tra.
- `my_dataset_test`: Tên đại diện cho tập dữ liệu kiểm thử.

### 3) Huấn luyện mô hình

```
from detectron2.engine import DefaultTrainer
from detectron2.evaluation import COCOEvaluator

class COCOTrainer(DefaultTrainer):
    @classmethod
    def build_evaluator(cls, cfg, dataset_name, output_folder=None):
        if output_folder is None:
            os.makedirs("coco_eval", exist_ok=True)
            output_folder = "coco_eval"
        return COCOEvaluator(dataset_name, cfg, False, output_folder)
```

Hình 2.2.4 Mã nguồn xây dựng lớp huấn luyện mô hình Faster R-CNN

Đoạn mã Python này xây dựng một lớp `COCOTrainer` kế thừa từ nguyên mẫu của thư viện `detectron2` là lớp `DefaultTrainer` để hiện thực phương thức `build_evaluator` thuộc về lớp để có thực hiện đánh giá mô hình trong quá trình huấn luyện. Phương thức này sẽ thực hiện kiểm tra mô hình dựa trên tập dữ liệu kiểm tra đã đăng ký trước đó và lưu lại các kết quả đánh giá vào thư mục chỉ định hoặc mặc định là `coco_eval`.

```

PRETRAINED_MODEL = "PATH_TO_PRETRAINED_MODEL"

cfg = get_cfg()
cfg.merge_from_file(model_zoo.get_config_file(PRETRAINED_MODEL))
cfg.DATASETS.TRAIN = ("my_dataset_train",)
cfg.DATASETS.TEST = ("my_dataset_val",)

cfg.DATALOADER.NUM_WORKERS = 2
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url(PRETRAINED_MODEL)
cfg.SOLVER.IMS_PER_BATCH = 4      # BATCH SIZE.
cfg.SOLVER.BASE_LR = 0.001        # LEARNING RATE.
cfg.SOLVER.MAX_ITER = 3000        # ITERATION.
cfg.TEST.EVAL_PERIOD = 200        # TEST PERIOD.
cfg.SOLVER.WARMUP_ITERS = 1000
cfg.SOLVER.WARMUP_FACTOR = 0.01

cfg.SOLVER.STEPS = [1000, 2000]
cfg.SOLVER.GAMMA = 0.05

cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 64  # ROI HEAD BATCH SIZE.
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 1           # NUMBER OF CLASS.

os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)
trainer = COCOTrainer(cfg)
trainer.resume_or_load(resume=False)
trainer.train()

```

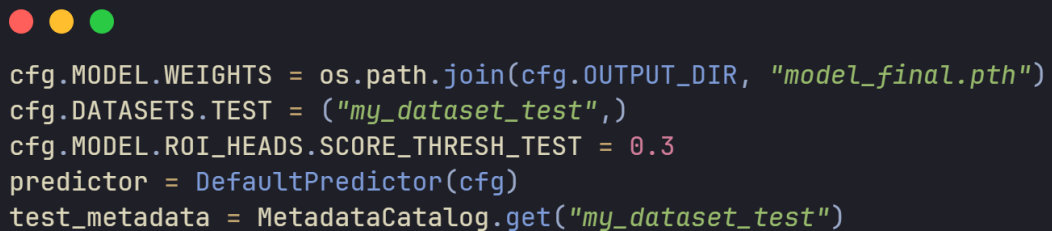
Hình 2.2.5 Mã nguồn huấn luyện mô hình Faster R-CNN

Đoạn mã Python này cấu hình và huấn luyện mô hình Faster R-CNN dựa vào trọng số mô hình đã được huấn luyện sẵn bởi thư viện *detectron2*. Các thông số cần thiết cho quá trình huấn luyện bao gồm:

- *PRETRAINED\_MODEL*: Đường dẫn tương đối đến tệp tin trọng số mô hình trên Github của thư viện *detectron2*.
- *cfg = get\_cfg()*: Khai báo biến lưu trữ thông tin cấu hình.
- *cfg.merge\_from\_file*: Hàm thực hiện kết hợp các cấu hình từ mô hình của thư viện *detectron2* vào cấu hình hiện tại.
- *cfg.DATASET.TRAIN*: Xác định tập dữ liệu huấn luyện.
- *cfg.DATASET.TEST*: Xác định tập dữ liệu kiểm tra.
- *cfg.MODEL.WEIGHTS*: Xác định trọng số mô hình được sử dụng.
- *cfg.SOLVER.IMS\_PER\_BATCH*: Xác định số lượng mẫu dùng 1 đợt huấn luyện mô hình.
- *cfg.SOLVER.BASE\_LR*: Xác định giá trị learning rate khởi đầu.
- *cfg.SOLVER.MAX\_ITER*: Xác định tổng số đợt huấn luyện.

- *cfg.TEST.EVAL\_PERIOD*: Xác định mốc huấn luyện mà mô hình dựa vào đó để đánh giá khả năng học trong quá trình huấn luyện khi số đợt huấn luyện hiện tại là bội số của mốc đó.
- *cfg.SOVER.WARMUP\_ITEERS*: Xác định mốc huấn luyện mà trước đó, mô hình sẽ giảm giá trị learning rate theo hệ số tỷ lệ chỉ định và tăng dần trở về giá trị gốc khi mô hình học đến mốc đó.
- *cfg.SOLVER.WARMUP\_FACTOR*: Xác định hệ số tỷ lệ để giảm giá trị gốc của learning rate khi mô hình bắt đầu học.
- *cfg.SOLVER.STEPS*: Xác định danh sách các mốc huấn luyện mà tại đó giá trị learning rate sẽ giảm đi theo hệ số tỷ lệ chỉ định.
- *cfg.SOLVER.GAMMA*: Xác định hệ số tỷ lệ để giảm giá trị learning rate khi mô hình học đến mốc chỉ định.
- *cfg.MODEL.ROI\_HEADS.BATCH\_SIZE\_PER\_IMAGE*: Xác định số lượng vùng đề xuất trong mỗi ảnh để huấn luyện các đầu ra của mô hình.
- *cfg.MODEL.ROI\_HEADS.NUM\_CLASSES*: Xác định số lượng lớp (nhãn) đối tượng mà mô hình cần nhận diện trong quá trình huấn luyện.

#### 4) Thực hiện dự đoán



```

cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR, "model_final.pth")
cfg.DATASETS.TEST = ("my_dataset_test",)
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.3
predictor = DefaultPredictor(cfg)
test_metadata = MetadataCatalog.get("my_dataset_test")
    
```

Hình 2.2.6 Mã nguồn cấu hình đối tượng dự đoán mô hình Faster R-CNN

Đoạn mã Python này cấu hình đối tượng dự đoán mô hình Faster R-CNN với trọng số mô hình đã huấn luyện. Các thông số cần thiết cho quá trình dự đoán bao gồm:

- *cfg.MODEL.WEIGHTS*: Thay thế trọng số mô hình hiện tại bằng trọng số mô hình đã được huấn luyện.
- *cfg.DATASETS.TEST*: Thay thế tập dữ liệu kiểm tra bằng tập dữ liệu kiểm thử dùng cho dự đoán.
- *cfg.MODEL.ROI\_HEADS.SCORE\_THRESH\_TEST*: Xác định giá trị độ tin cậy dùng làm ngưỡng để yêu cầu mô hình chỉ giữ lại các phát hiện mà độ tin cậy cao hơn hoặc bằng ngưỡng đó.
- *test\_metadata = MetadataCatalog.get(\_\_)*: Lệnh lấy thông tin siêu dữ liệu của tập dữ liệu kiểm thử.

```
import glob

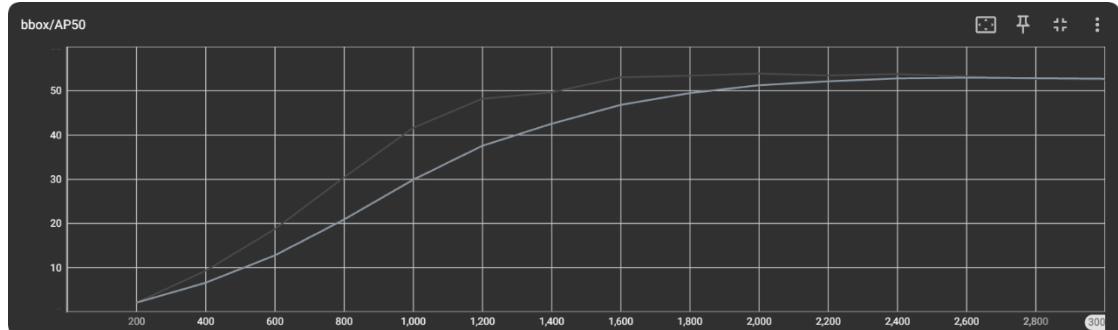
for image_name in glob.glob("PATH_TO_TEST_DATA"):
    image = cv2.imread(image_name)
    outputs = predictor(image)
    v = Visualizer(image[:, :, ::-1], metadata=test_metadata, scale=0.8)
    out = v.draw_instance_predictions(outputs["instances"].to("cpu"))
    cv2.imshow(out.get_image()[:, :, ::-1])
```

Hình 2.2.7 Mã nguồn thực hiện dự đoán mô hình Faster R-CNN

Đoạn mã Python này dự đoán và hiển thị các kết quả dự đoán trên tập dữ liệu kiểm thử. Các bước thực hiện cụ thể như sau:

- `image = cv2.imread(__)`: Đọc tệp tin hình ảnh.
- `outputs = predictor(__)`: Dự đoán với mô hình đã huấn luyện.
- `v = Visualizer(__)`: Chuyển đổi dữ liệu dự đoán đầu ra sang dạng đối tượng mà có thể hiển thị được thông qua cv2.
- `out = v.draw_instance_predictions(__)`: Vẽ các dự đoán lên hình ảnh.
- `cv2.imshow(__)`: Hiển thị hình ảnh đã được xử lý.

### 5) Đánh giá kết quả huấn luyện



Hình 2.2.8 Kết quả huấn luyện mô hình Faster R-CNN

Mô hình Faster R-CNN đã được huấn luyện dựa trên trọng số sẵn có từ mô hình X101 – FPN được tải từ Github *detectron2*, với số lượng vòng huấn luyện là 20 (3000 đợt huấn luyện) và số lượng mẫu trong mỗi đợt huấn luyện là 4. Mô hình đạt được giá trị *mAP50* khoảng 55% và *mAP50 – 95* khoảng 25%.

Dựa trên các biểu đồ được thống kê trong quá trình huấn luyện, các giá trị đo lường độ mất mát như *total\_loss*, *loss\_cls*, *loss\_box\_reg*, *loss\_rpn\_cls*, *loss\_rpn\_loc* giảm dần và các giá trị đo lường độ chính xác như AP & AR tăng dần trong quá trình huấn luyện. Tổng kết, xu hướng tổng thể là tích cực, gợi ý rằng mô hình đang hội tụ.





Hình 2.2.9 Một số kết quả kiểm thử bởi Faster R-CNN

### 3. HUẤN LUYỆN MÔ HÌNH YOLO

#### 1) Cài đặt môi trường




```
!pip install -q git+https://github.com/THU-MIG/yolov10
!pip install -q roboflow
```

Hình 2.3.1 Mã nguồn cài đặt môi trường YOLOv10

Đây là các lệnh thực thi tải xuống các gói thư viện cần thiết vào môi trường Python trên Google Colab, hoạt động ở chế độ dòng lệnh.

- `!pip install -q git + https://github.com/THU - MIG/yolov10`: Lệnh cài đặt tự động các thư viện cần thiết từ kho lưu trữ Github yolov10 để có thể xây dựng mô hình YOLOv10.
- `!pip install -q roboflow`: Lệnh cài đặt gói thư viện roboflow, một thư viện giúp dễ dàng tích hợp với Roboflow, một nền tảng cung cấp dịch vụ gán nhãn dữ liệu và huấn luyện mô hình máy học.

#### 2) Tải trọng số mô hình



```
!wget -P '/pretrained' 'https://github.com/jameslahm/yolov10/releases/download/v1.1/yolov10n.pt'
!wget -P '/pretrained' 'https://github.com/jameslahm/yolov10/releases/download/v1.1/yolov10s.pt'
!wget -P '/pretrained' 'https://github.com/jameslahm/yolov10/releases/download/v1.1/yolov10m.pt'
!wget -P '/pretrained' 'https://github.com/jameslahm/yolov10/releases/download/v1.1/yolov10b.pt'
!wget -P '/pretrained' 'https://github.com/jameslahm/yolov10/releases/download/v1.1/yolov10l.pt'
!wget -P '/pretrained' 'https://github.com/jameslahm/yolov10/releases/download/v1.1/yolov10x.pt'
```

Hình 2.3.2 Mã nguồn tải trọng số mô hình YOLOv10

Đây là các lệnh Linux thực thi tải xuống các phiên mô hình YOLOv10 khác nhau và lưu vào thư mục chỉ định. Các mô hình này đại diện cho các phiên bản với kích thước, độ chính xác và độ trễ tăng dần từ  $N \sim X$ .

- *YOLOv10 – N*: Thích hợp cho môi trường cực kỳ hạn chế về tài nguyên.
- *YOLOv10 – S*: Cân bằng tốc độ và độ chính xác.
- *YOLOv10 – M*: Sử dụng cho mục đích chung.
- *YOLOv10 – B*: Độ chính xác cao hơn với chiều rộng tăng.
- *YOLOv10 – L*: Độ chính xác cao với chi phí tài nguyên tính toán.
- *YOLOv10 – X*: Độ chính xác và hiệu suất tối đa.

Mỗi biến thể của YOLOv10 được thiết kế cho các nhu cầu tính toán và yêu cầu độ chính xác khác nhau, làm cho mô hình trở nên linh hoạt và phù hợp hơn với nhiều ứng dụng khác nhau.



### 3) Tải tập dữ liệu

```
from roboflow import Roboflow

PATH_DATASET = "PATH_TO_DATASET"

rf = Roboflow(api_key="API_KEY")
project = rf.workspace("WORKSPACE").project("PROJECT")
version = project.version("VERSION")
dataset = version.download("FORMAT", location=PATH_DATASET, overwrite=True)
```

Hình 2.3.3 Mã nguồn tải tập dữ liệu từ Roboflow (YOLOv9)

Đoạn mã Python sử dụng thư viện *roboflow* để thực thi tải xuống một tập dữ liệu từ Roboflow, định dạng mà thư viện *ultralytics* yêu cầu là YOLOv9. Các thông tin cần thiết để tải xuống bao gồm:

- *PATH\_TO\_DATASET*: Đường dẫn lưu trữ tập dữ liệu.
- *API\_KEY*: Thông tin khóa API để truy cập vào các tài nguyên của Roboflow.
- *WORKSPACE*: Thông tin tên Workspace nơi chứa Project.
- *PROJECT*: Thông tin tên Project nơi chứa tập dữ liệu.
- *VERSION*: Thông tin phiên bản tập dữ liệu.
- *FORMAT*: Thông tin định dạng tập dữ liệu. Tùy chọn *overwrite = True* cho phép ghi đè dữ liệu nếu nó đã tồn tại trong đường dẫn thư mục lưu trữ.

Khi thực thi thành công, một tệp tin *.yaml* được đính kèm và cấu hình sẵn bởi Roboflow. Đây là tệp tin cấu hình cho quá trình huấn luyện mô hình YOLO.

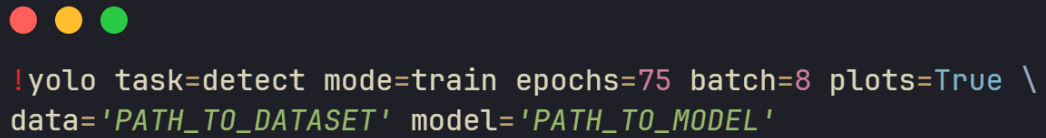
```
train: PATH_TO_IMAGES_TRAIN
val: PATH_TO_IMAGES_VALID
test: PATH_TO_IMAGES_TEST

nc: NUMBER_OF_CLASS
names: LIST_OF_CLASS_NAME
```

Hình 2.3.4 Mã nguồn tệp tin *.yaml*

- *PATH\_TO\_IMAGES\_TRAIN*: Đường dẫn đến thư mục hình ảnh huấn luyện.
- *PATH\_TO\_IMAGES\_VALID*: Đường dẫn đến thư mục hình ảnh kiểm tra.
- *PATH\_TO\_IMAGES\_TEST*: Đường dẫn đến thư mục hình ảnh kiểm thử.
- *NUMBER\_OF\_CLASS*: Thông tin số lượng lớp đối tượng dự đoán.
- *LIST\_OF\_CLASS\_NAME*: Thông tin danh sách tên lớp đối tượng dự đoán.

#### 4) Huấn luyện mô hình



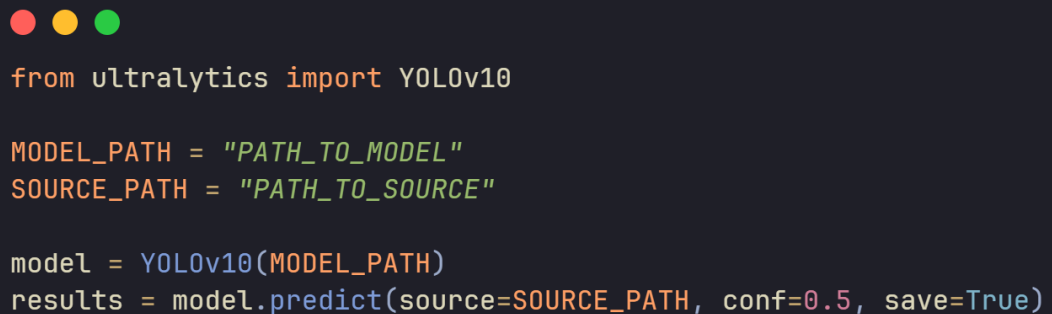
```
!yolo task=detect mode=train epochs=75 batch=8 plots=True \
data='PATH_TO_DATASET' model='PATH_TO_MODEL'
```

Hình 2.3.5 Mã nguồn huấn luyện mô hình YOLOv10

Đây là lệnh thực thi huấn luyện mô hình với các tham số chỉ định, hoạt động ở chế độ dòng lệnh. Các thông tin cần thiết cho quá trình huấn luyện bao gồm:

- *task*: Xác định nhiệm vụ của mô hình (*detect*: nhiệm vụ phát hiện đối tượng, *segment*: phân đoạn đối tượng, *export*: xuất mô hình thành dạng tệp tin).
- *mode*: Xác định chế độ thực thi nhiệm vụ (*train*: huấn luyện, *val*: kiểm tra, *predict*: dự đoán).
- *epochs*: Xác định số lượng vòng huấn luyện. Một vòng là một lượt duyệt qua toàn bộ dữ liệu huấn luyện, chia làm nhiều đợt huấn luyện.
- *batch*: Xác định số lượng mẫu được đưa vào mô hình trong mỗi đợt của một vòng huấn luyện.
- *plots*: Xác định tắt/mở tính năng vẽ đồ thị. Điều này cho phép YOLO tạo ra các đồ thị về quá trình huấn luyện, chẳng hạn như đường cong mất mát và độ chính xác.
- *PATH\_TO\_DATASET*: Đường dẫn đến dữ liệu huấn luyện. Thường dùng một tệp tin *.yaml* để cấu hình thông tin cho tất cả quá trình huấn luyện, kiểm tra và kiểm thử mô hình.
- *PATH\_TO\_MODEL*: Đường dẫn đến mô hình cơ sở để huấn luyện. Thường dùng một mô hình đã được huấn luyện sẵn của YOLOv10 để cải tiến lại nhằm phục vụ cho mục tiêu cụ thể.

#### 5) Thực hiện dự đoán



```
from ultralytics import YOLOv10

MODEL_PATH = "PATH_TO_MODEL"
SOURCE_PATH = "PATH_TO_SOURCE"

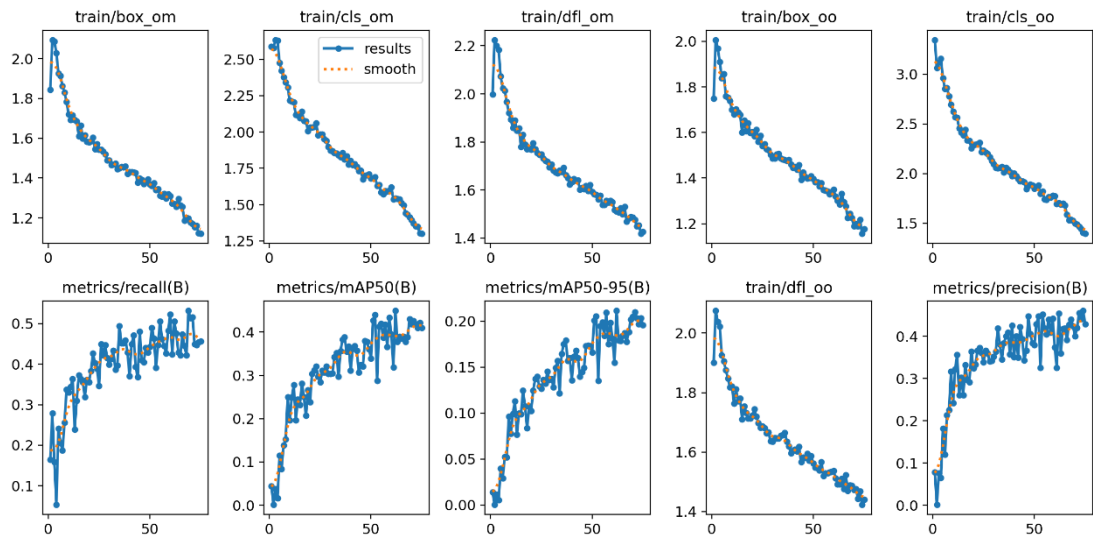
model = YOLOv10(MODEL_PATH)
results = model.predict(source=SOURCE_PATH, conf=0.5, save=True)
```

Hình 2.3.6 Mã nguồn thực hiện dự đoán mô hình YOLOv10

Đoạn mã Python này sử dụng thư viện ultralytics để thực hiện dự đoán với mô hình YOLOv10 đã được huấn luyện. Các thông tin cần thiết cho quá trình dự đoán bao gồm:

- *MODEL\_PATH*: Đường dẫn đến mô hình đã được huấn luyện.
- *SOURCE\_PATH*: Đường dẫn đến dữ liệu dự đoán, có thể là một tệp tin dạng hình ảnh, tệp tin dạng video, thư mục ảnh hoặc tệp tin .yaml.
- *model = YOLOv10(—)*: Lệnh tạo đối tượng YOLOv10. Đây là cách tải mô hình YOLOv10 đã huấn luyện.
- *results = model.predict(—)*: Lệnh dự đoán với mô hình YOLOv10 được tải lên. Tùy chọn *conf = 0.5* yêu cầu mô hình chỉ giữ lại các phát hiện mà độ tin cậy cao hơn hoặc bằng 0.5 và *save = True* yêu cầu chương trình lưu lại các kết quả dự đoán.

## 6) Đánh giá kết quả huấn luyện



Hình 2.3.7 Kết quả huấn luyện mô hình YOLOv10

Mô hình YOLOv10 đã được huấn luyện dựa trên trọng số sẵn có từ mô hình YOLOv10 – B được tải từ Github *yolov10*, với số lượng vòng huấn luyện là 75 và số lượng mẫu trong mỗi đợt huấn luyện là 8. Mô hình đạt được giá trị *mAP50* khoảng 45% và *mAP50 – 95* khoảng 20%.

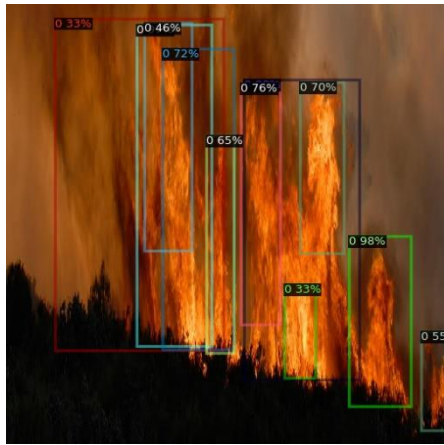
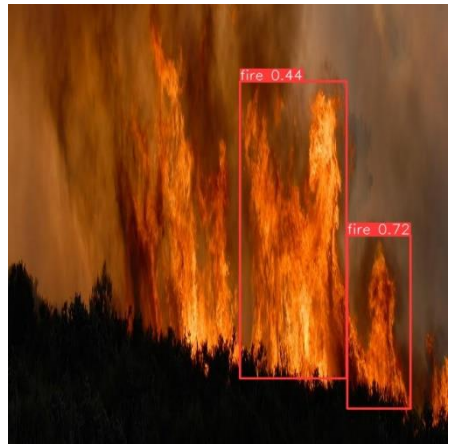
Dựa trên các biểu đồ được thống kê trong quá trình huấn luyện, các giá trị đo lường độ mất mát như *box\_om*, *cls\_om*, *dfl\_om*, *box\_oo*, *cls\_oo*, *dfl\_oo* giảm dần và các giá trị đo lường độ chính xác như *precision*, *recall*, *mAP50*, *mAP50 – 95* tăng dần trong quá trình huấn luyện. Tổng kết, xu hướng tổng thể là tích cực, gợi ý rằng mô hình đang hội tụ.





Hình 2.3.8 Một số kết quả kiểm thử bởi YOLOv10

#### 4. SO SÁNH THỰC NGHIỆM

	<i>Faster R-CNN</i>	<i>YOLOv10</i>
<i>Thông số</i>	<ul style="list-style-type: none"> <li><i>epoch</i>: 20 vòng</li> <li><i>batch</i>: 4 mẫu</li> </ul>	<ul style="list-style-type: none"> <li><i>epoch</i>: 75 vòng</li> <li><i>batch</i>: 8 mẫu</li> </ul>
<i>Độ chính xác</i>	<ul style="list-style-type: none"> <li><i>mAP50</i>: <math>\approx 55\%</math></li> <li><i>mAP50 - 95</i>: <math>\approx 25\%</math></li> </ul>	<ul style="list-style-type: none"> <li><i>mAP50</i>: <math>\approx 45\%</math></li> <li><i>mAP50 - 95</i>: <math>\approx 20\%</math></li> </ul>
<i>Tốc độ</i>	<ul style="list-style-type: none"> <li><i>Huấn luyện</i>: <math>\approx 3h</math></li> <li><i>Dự đoán</i>: <math>\approx 87.1\text{ ms/mẫu}</math></li> </ul>	<ul style="list-style-type: none"> <li><i>Huấn luyện</i>: <math>\approx 1h</math></li> <li><i>Dự đoán</i>: <math>\approx 33.4\text{ ms/mẫu}</math></li> </ul>
<i>Dự đoán</i>		

Dựa trên các kết quả thống kê từ quá trình thực nghiệm huấn luyện và dự đoán của 2 mô hình Faster R-CNN và YOLOv10 trên cùng tập dữ liệu gồm 1350 ảnh trên Google Colab, tôi tổng kết được như sau:

- Faster R-CNN*: Mô hình này đạt được độ chính xác cao hơn mô hình YOLOv10 trong khi số vòng huấn luyện chỉ  $\approx 1/3$  và số lượng mẫu mỗi đợt chỉ  $1/2$ . Mặc dù Faster R-CNN có thể giảm đáng kể thông số huấn luyện nhưng sự phức tạp trong quá trình tính toán là rất lớn dẫn đến thời gian huấn luyện và dự đoán của mô hình cần gấp  $\approx 3$  lần so với YOLOv10. Tổng thể, mô hình Faster R-CNN có khả năng dự đoán vượt trội ở mọi kích thước đối tượng nhưng thời gian yêu cầu là rất lớn, không đáp ứng tiêu chí thời gian thực.
- YOLOv10*: Mô hình đạt được tốc độ xử lý vượt trội so với mô hình Faster R-CNN nhưng độ chính xác của YOLOv10 chỉ ở mức tương đối. Mặc dù YOLOv10 dùng số lượng vòng huấn luyện gấp  $\approx 3$  lần và số lượng mẫu mỗi đợt gấp 2 lần so với mô hình Faster R-CNN nhưng vì được đơn giản hóa quá trình tính toán nên thời gian huấn luyện và dự đoán là ngắn hơn đáng kể, cụ thể là giảm đi  $\approx 3$  lần. Tổng thể, mô hình YOLOv10 có khả năng dự đoán kém hơn nhưng thời gian yêu cầu rất ít, phù hợp tiêu chí thời gian thực.

## CHƯƠNG 3. XÂY DỰNG VÀ THỬ NGHIỆM CÔNG CỤ DỰ BÁO CHÁY THỜI GIAN THỰC DỰA TRÊN HÌNH ẢNH

### 1. GIỚI THIỆU CÔNG NGHỆ

#### a) Ngôn ngữ lập trình Python



Hình 3.1.1 Hình ảnh logo ngôn ngữ Python

Python là một ngôn ngữ lập trình bậc cao, được sử dụng cho các mục đích lập trình đa chức năng, được Guido van Rossum phát triển và lần đầu ra mắt vào năm 1991 tại Hà Lan. Python có nguồn gốc từ nhiều ngôn ngữ khác, bao gồm ABC, Modula-3, C, C++, Algol-68, SmallTalk, Unix shell và các ngôn ngữ script khác. Python được thiết kế với ưu điểm mạnh là dễ đọc, dễ học, dễ nhớ, có hình thức rất sáng sủa, cấu trúc rõ ràng, thuận tiện cho việc phát triển và bảo trì.

#### b) Thư viện Customtkinter



Hình 3.1.2 Hình ảnh logo thư viện Customtkinter

Customtkinter là một thư viện giao diện đồ họa hiện đại dựa trên nền tảng Tkinter, được phát triển bởi Tom Schimansky. Thư viện này cung cấp các thành phần giao diện mới mẻ, hiện đại và hoàn toàn có thể tùy chỉnh, có thể được sử dụng độc lập hoặc kết hợp với Tkinter. Ngoài ra, màu sắc của những thành phần giao diện được hỗ trợ bởi CustomTkinter có thể thích ứng linh hoạt với giao diện hệ thống người dùng, và đều hỗ trợ khả năng mở rộng độ phân giải.

#### c) Thư viện Pygame



Hình 3.1.3 Hình ảnh logo thư viện Pygame



Pygame là một thư viện mã nguồn mở trên nền tảng Python, được phát triển bởi Pete Shinnars và ra mắt vào năm 2000. Được thiết kế theo hướng dễ sử dụng và hoạt động trên nhiều hệ điều hành, Pygame tích hợp với Simple DirectMedia Layer (SDL), cung cấp các chức năng đa phương tiện mạnh mẽ, bao gồm hỗ trợ âm thanh, hình ảnh và xử lý sự kiện. Pygame chủ yếu được sử dụng để phát triển các trò chơi 2D và ứng dụng đa phương tiện, đồng thời cũng là một công cụ giáo dục tuyệt vời cho việc dạy và học lập trình Python.

**d) Thư viện OpenCV**



*Hình 3.1.4 Hình ảnh logo thư viện OpenCV*

OpenCV là một thư viện mã nguồn mở mạnh mẽ cho Python và nhiều ngôn ngữ lập trình khác, được phát triển bởi Intel và ra mắt vào năm 2000. Được thiết kế để hỗ trợ các ứng dụng thị giác máy tính và xử lý ảnh, OpenCV cung cấp hàng loạt hàm và công cụ cho các nhiệm vụ như lọc và biến đổi hình ảnh, phát hiện và theo dõi đối tượng, nhận diện khuôn mặt và phân tích video. Thư viện này hoạt động trên nhiều hệ điều hành, bao gồm Windows, MacOS và Linux, và được ứng dụng rộng rãi trong nhiều lĩnh vực từ nghiên cứu khoa học, phát triển sản phẩm, đến giáo dục.

**e) Thư viện Ultralytics**



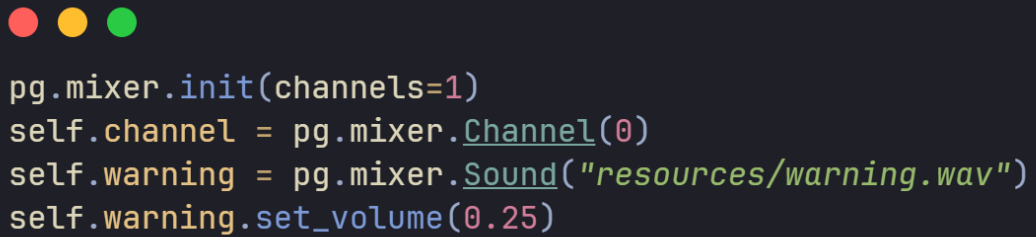
*Hình 3.1.4 Hình ảnh logo thư viện Ultralytics*

Ultralytics là một thư viện mã nguồn mở trên nền tảng Python, nổi tiếng với các mô hình YOLO tiên tiến có thể phát hiện và nhận diện đối tượng trong hình ảnh. Được thiết kế với tính dễ sử dụng và hiệu quả cao, Ultralytics cung cấp các công cụ tiên tiến để huấn luyện, đánh giá và triển khai các mô hình YOLO. Ngoài ra, thư viện này hỗ trợ nhiều nhiệm vụ thị giác máy tính, từ phát hiện đối tượng, phân loại hình ảnh đến theo dõi đối tượng thời gian thực, và được ứng dụng rộng rãi trong nhiều lĩnh vực như an ninh, y tế, và công nghiệp.



## 2. XÂY DỰNG CHƯƠNG TRÌNH

### 1) Phần âm thanh cảnh báo

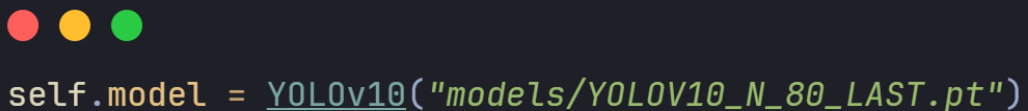


```
pg.mixer.init(channels=1)
self.channel = pg.mixer.Channel(0)
self.warning = pg.mixer.Sound("resources/warning.wav")
self.warning.set_volume(0.25)
```

Hình 3.2.1 Mã nguồn cấu hình âm thanh cảnh báo

- `pg.mixer.init(__)`: Hàm khai báo những thông tin cần thiết để phần xử lý âm thanh của thư viện Pygame có thể hoạt động. Đồng thời, chỉ định số lượng kênh âm thanh dùng trong chương trình là 1.
- `self.channel`: Biến lưu trữ kênh âm thanh mà đối tượng âm thanh sử dụng.
- `self.warning`: Thực hiện tải tệp tin âm thanh theo đường dẫn tương đối trên bằng lớp `Sound` và lưu trữ đối tượng âm thanh vào biến `warning`.
- `self.warning.set_volume(__)`: Hàm điều chỉnh âm lượng của đối tượng âm thanh xuống 1/4 âm lượng gốc.

### 2) Phần mô hình nhận diện

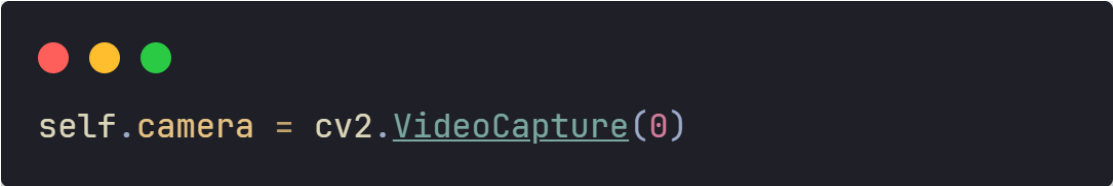


```
self.model = YOLOv10("models/YOLOV10_N_80_LAST.pt")
```

Hình 3.2.2 Mã nguồn tải mô hình YOLO

- `self.model`: Biến lưu trữ đối tượng mô hình `YOLOv10`. Mô hình sử dụng được huấn luyện dựa trên trọng số có kích thước nhỏ nhất và độ trễ thấp nhất là `YOLOV10 – N` với tập huấn luyện gồm  $\approx 6000$  mẫu, và đạt được độ chính xác là  $\approx 80\%$ .

### 3) Phần ghi hình đối tượng



```
self.camera = cv2.VideoCapture(0)
```

Hình 3.2.3 Mã nguồn cấu hình Webcam

- `self.camera`: Biến lưu trữ đối tượng ghi hình, thiết lập tham số chỉ mục là 0 để chuyển đổi sang chế độ ghi hình thời gian thực với Webcam.

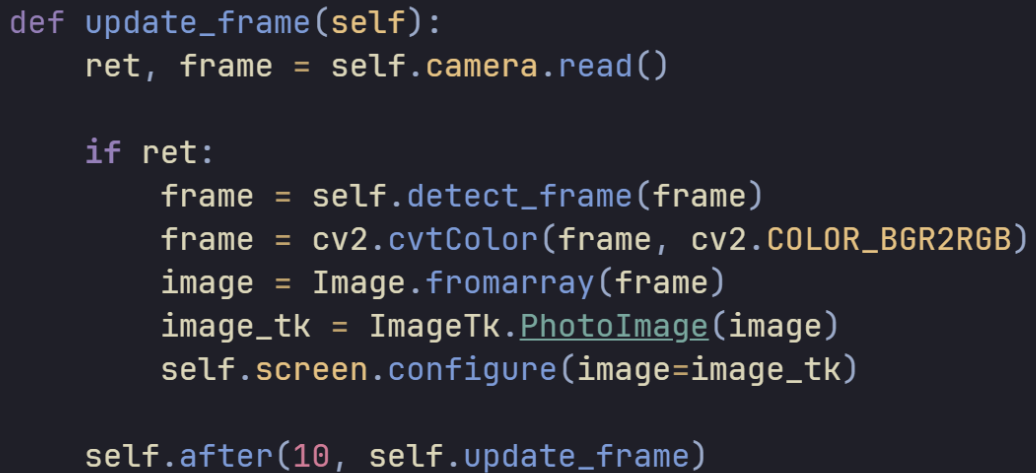
#### 4) Phần nhận diện hình ảnh

```
def detect_frame(self, frame):
    result = self.model.predict(source=frame, conf=self.conf_thresh.get())[0]
    # LẤY THÔNG TIN HỘ GIỚI HẠN.
    boxes = result.bboxes.xyxy
    if len(boxes) > 0:
        # LẤY THÔNG TIN XÁC SUẤT DỰ ĐOÁN.
        confs = result.bboxes.conf
        for box, conf in zip(boxes, confs):
            t, l, r, b = map(int, box.tolist())
            # VẼ HỘ GIỚI HẠN.
            cv2.rectangle(
                img=frame,
                pt1=(t, l),
                pt2=(r, b),
                color=self.FRAME_COLOR,
                thickness=2,
            )
            # VẼ XÁC SUẤT DỰ ĐOÁN.
            cv2.putText(
                img=frame,
                text=f"{conf:.2f}",
                org=(t, l - 10),
                fontFace=cv2.FONT_HERSHEY_SIMPLEX,
                fontScale=0.5,
                color=self.FRAME_COLOR,
            )
        # CẢNH BÁO ÂM THANH.
        if not self.channel.get_busy():
            self.channel = self.warning.play()
    return frame
```

Hình 3.2.4 Mã nguồn xử lý hình ảnh

- `result = self.model.predict(__)[0]`: Biến lưu trữ kết quả dự đoán của mô hình với tham số độ tin cậy được thiết lập từ người sử dụng.
- `boxes = result.bboxes.xyxy`: Biến lưu trữ danh sách thông tin tọa độ của các hộp giới hạn được dự đoán từ đối tượng hình ảnh theo định dạng là tọa độ điểm ở góc trên cùng bên trái và góc dưới cùng bên phải.
- `confs = result.bboxes.conf`: Biến lưu trữ danh sách thông tin xác suất dự đoán của các hộp giới hạn được dự đoán từ đối tượng hình ảnh.
- `cv2.rectangle(__)`: Hàm thực hiện vẽ lên đối tượng hình ảnh các hộp giới hạn thông qua tọa độ được trích xuất từ kết quả dự đoán.
- `cv2.putText(__)`: Hàm thực hiện vẽ lên đối tượng hình ảnh các xác suất dự đoán thông qua tọa độ được tính toán từ tọa độ của hộp giới hạn.
- `channel = warning.play()`: Hàm thực hiện phát âm thanh cảnh báo và lưu lại thông tin kênh âm thanh hoạt động vào biến `channel` để kiểm tra xem kênh đó đã được giải phóng hay chưa nhằm tránh phát âm thanh trùng lặp.

### 5) Phần giao diện chương trình



```
def update_frame(self):
    ret, frame = self.camera.read()

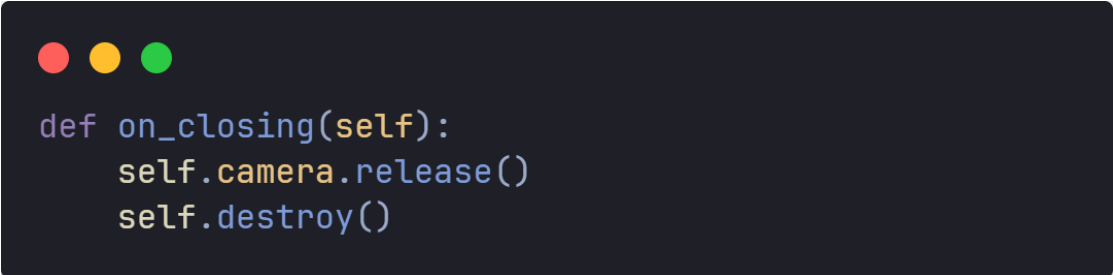
    if ret:
        frame = self.detect_frame(frame)
        frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        image = Image.fromarray(frame)
        image_tk = ImageTk.PhotoImage(image)
        self.screen.configure(image=image_tk)

    self.after(10, self.update_frame)
```

Hình 3.2.5 Mã nguồn trình bày hình ảnh lên giao diện

- `frame = self.detect_frame(__)`: Thực hiện gọi hàm nhận diện hình ảnh và lưu trữ lại hình ảnh vừa được xử lý.
- `frame = cv2.cvtColor(__)`: Chuyển đổi hình ảnh thành dạng mà thư viện xử lý hình ảnh của Python có thể xử lý.
- `image = Image.fromarray(__)`: Chuyển đổi hình ảnh từ dạng ma trận thành đối tượng hình ảnh.
- `image_tk = ImageTk.PhotoImage(__)`: Chuyển đổi đối tượng hình ảnh thành dạng mà thư viện Customtkinter có thể xử lý.
- `self.screen.configure(image = image_tk)`: Ghi đè hình ảnh đang được hiển thị bởi đối tượng giao diện.
- `self.after(__)`: Hàm thực hiện gọi lại một hàm khác sau một khoảng thời gian chỉ định. Mã này dùng để thực hiện đệ quy lên hàm `update_frame`.

### 6) Kết thúc chương trình



```
def on_closing(self):
    self.camera.release()
    self.destroy()
```

Hình 3.2.6 Mã nguồn kết thúc chương trình

Hàm `on_closing` thực hiện thu hồi quyền sử dụng Webcam và giải phóng các tài nguyên mà chương trình ứng dụng đã được cấp phát.

### 3. KẾT QUẢ THỬ NGHIỆM



Hình 3.3.1 Hình ảnh công cụ thử nghiệm

Tổng quan, ứng dụng hoạt động với sự kết hợp của 3 thư viện là *Pygame* để xử lý âm thanh, *OpenCV* để xử lý hình ảnh, *Ultralytics* để nhận diện đối tượng, và tất cả được điều khiển bởi giao diện ứng dụng của thư viện *Customtkinter*. Ngoài ra, chương trình cho phép thiết lập các mức độ tin cậy khác nhau giúp khả năng nhận diện đối tượng của mô hình trở nên linh hoạt hơn.

Ứng dụng sử dụng *OpenCV* để nhận hình ảnh từ Webcam, sau đó chuyển dữ liệu hình ảnh này vào mô hình YOLOv10 của *Ultralytics* để thực hiện nhận diện đối tượng. Sau đó, thông tin nhận diện được *OpenCV* xử lý, trình bày lên giao diện, và có thể phát âm thanh cảnh báo bằng *Pygame* nếu cần.

Các vùng chứa đối tượng nhận diện bởi mô hình được khoanh vùng bằng các đường viền hình chữ nhật màu đỏ, là các khu vực có xác suất dự đoán được biểu diễn trên góc trái của hình chữ nhật vượt qua độ tin cậy đã chỉ định. Khi đó, ứng dụng sẽ phát ra âm thanh cảnh báo để thông báo cho người giám sát.

Tổng kết, ứng dụng cung cấp một giải pháp giám sát hiệu quả bằng sự kết hợp 3 thư viện *Pygame*, *OpenCV* và *Ultralytics*, được quản lý thư viện *Customtkinter*. Không chỉ nhận diện đối tượng từ hình ảnh Webcam và cung cấp cảnh báo âm thanh khi cần thiết mà còn hỗ trợ điều chỉnh mức độ tin cậy, ứng dụng có thể linh hoạt đáp ứng nhu cầu giám sát trong nhiều tình huống khác nhau trong thực tiễn.

## KẾT LUẬN, KIẾN NGHỊ

Ứng dụng mô hình YOLO trong dự báo cháy thời gian thực là một hướng nghiên cứu quan trọng nhằm tăng cường khả năng phát hiện và phản ứng nhanh chóng với các tình huống cháy nổ. Tuy nhiên, bài nghiên cứu này mặc dù đã sử dụng mô hình tiên tiến của dòng YOLO, là YOLOv10, nhưng vẫn đang gặp phải một số thách thức lớn về hiệu suất của mô hình, bao gồm độ chính xác còn thấp và độ trễ còn hạn chế, làm giảm hiệu quả trong việc nhận diện.

- *Độ chính xác:* Mô hình được sử dụng đạt độ chính xác  $\approx 80\%$ . Mặc dù, mô hình có thể nhận diện được các ngữ cảnh cơ bản nhưng đối với các ngữ cảnh phức tạp, gồm các đối tượng chi tiết nhỏ, vẫn còn hạn chế.
- *Độ trễ:* Mô hình được sử dụng có độ trễ  $\approx 33.4ms$  trong quá trình huấn luyện và kiểm thử nhưng khi kết hợp với ứng dụng thì độ trễ  $\approx 150ms$ . Qua đó, ứng dụng vẫn còn hạn chế với tiêu chí thời gian thực.

Qua đó, để cải thiện hiệu suất của hệ thống, việc phân tích và đánh giá các yếu tố ảnh hưởng đến hiệu suất của mô hình là cần thiết, từ đó đưa ra các giải pháp nâng cấp và tối ưu hóa. Điều này có thể bao gồm việc cải thiện dữ liệu huấn luyện, tối ưu hóa mô hình, nâng cấp phần cứng và phát triển giao diện người dùng hiệu quả hơn.

- *Cải thiện dữ liệu huấn luyện:* Tập dữ liệu huấn luyện được sử dụng không đủ lớn và đa dạng, dẫn đến mô hình không được học đầy đủ các đặc điểm của đám cháy. Vì vậy, cần tăng cường và làm phong phú dữ liệu huấn luyện bằng cách thu thập thêm các hình ảnh cháy từ nhiều nguồn và trong nhiều điều kiện khác nhau.
- *Nâng cấp phần cứng:* Hiện tại, mô hình thiếu các kỹ thuật tối ưu hóa và cũng như phần cứng chưa đáp ứng được yêu cầu thời gian thực. Vì vậy, cần điều chỉnh và thử nghiệm các siêu tham số khác nhau để tìm ra cấu hình tối ưu nhất. Đồng thời, cần nâng cấp hệ thống phần cứng để giảm thiểu độ trễ trong quá trình dự đoán.
- *Phát triển giao diện người dùng:* Hiện tại, ứng dụng tạo ra chủ yếu chỉ dùng để mô phỏng quá trình nhận diện của mô hình nên phần giao diện còn nhiều hạn chế. Vì vậy, để có thể sử dụng trong môi trường thực tế, cần bổ sung thiết kế giao diện người dùng trực quan và dễ sử dụng hơn.

Tổng kết, ứng dụng YOLO trong dự báo cháy thời gian thực đã chỉ ra những tiềm năng và thách thức sẽ phải đối mặt. Mặc dù đã sử dụng mô hình tiên tiến như YOLOv10, mô hình vẫn chưa đạt được độ chính xác mong muốn và ứng dụng có độ trễ cao, chưa đáp ứng tiêu chí thời gian thực. Để khắc phục các hạn chế này, cần thực hiện cải tiến về dữ liệu huấn luyện, tối ưu hóa mô hình và nâng cấp phần cứng. Ngoài ra, việc phát triển giao diện người dùng trực quan cũng là một yếu tố quan trọng giúp nâng cao tính khả dụng của ứng dụng. Qua những phân tích và đề xuất trên, ứng dụng có thể đạt được hiệu suất cao hơn, đáp ứng yêu cầu thời gian thực trong thực tiễn.

## TÀI LIỆU THAM KHẢO

- [1] S. Ren; K. He; R. Girshick; J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137-1149, 2017.
- [2] J. Hosang; R. Benenson; B. Schiele, "Learning Non-maximum Suppression," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6469-6477, 2017.
- [3] Nttuan8, "Bài 11: Object detection với Faster R-CNN," 4 April 2019. [Online]. Available: <https://nttuan8.com/bai-11-object-detection-voi-faster-r-cnn/>. [Accessed 2024 July 18].
- [4] A. Wang; H. Chen; L. Liu; K. Chen; Z. Lin; J. Han; G. Ding, "YOLOv10: Real-Time End-to-End Object Detection," *arXiv preprint arXiv:2405.14458*, 2024.
- [5] glenn-jocher; zhixuwei; abirami-vina; RizwanMunawar; Burhan-Q, "YOLOv10 - Ultralytics YOLO Documents," Ultralytics, 25 May 2024. [Online]. Available: <https://docs.ultralytics.com/vi/models/yolov10/>. [Accessed 15 July 2024].
- [6] C. -Y. Wang; H. -Y. Mark Liao; Y. -H. Wu; P. -Y. Chen; J. -W. Hsieh; I. -H. Yeh, "CSPNet: A New Backbone that can Enhance Learning Capability of CNN," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1571-1580, 2020.
- [7] N. Hoàng, "Tóm Tắt : YOLOv10: Real-Time End-to-End Object Detection," Viblo, 26 May 2024. [Online]. Available: <https://viblo.asia/p/tom-tat-yolov10-real-time-end-to-end-object-detection-WR5JRZldJGv>. [Accessed 21 July 21].
- [8] Shu Liu; Lu Qi; Haifang Qin; Jianping Shi; Jiaya Jia, "Path Aggregation Network for Instance Segmentation," *arXiv:1803.01534*, 2018.