Trường Đại Học Điện Lực
Khoa Công Nghệ Thông Tin

BÁO CÁO CHUYÊN ĐỀ HỌC PHẦN
HỌC SÂU

ĐỀ TÀI :

NHẬN DIỆN CHÁY KHÓI SỬ DỤNG MÔ HÌNH MẠNG YOLOV5

Sinh viên thực hiện	**:**	**NGUYỄN MẠNH HƯNG** **PHẠM ĐỨC MẠNH**
Lớp	**:**	**D15TTNT&TGMT**
Giảng Viên Hướng Dẫn	**:**	**TS. ĐOÀN THỊ HƯƠNG GIANG**
Ngành	**:**	**CÔNG NGHỆ THÔNG TIN**
Chuyên Ngành	**:**	**TTNT & TGMT**
Khóa	**:**	**2020-2025**

Hà Nội, Tháng 5, Năm 2023

1. Mô tả tóm tắt đề tài

- Xây dựng chương trình phát hiện cháy nổ sử dụng mô hình YOLO:
  - Huấn luyện mô hình trên ba phiên bản: YOLOv3-tiny, YOLOv5s, YOLOv6s
  - Thử nghiệm và đánh giá kết quả 3 mô hình: YOLOv3-tiny, YOLOv5s, YOLOv6s

2. Nội dung thực hiện

- Chương 1: Tổng quan về bài toán
- Chương 2: Tổng quan về mô hình mạng YOLO
- Chương 3: Chương trình

3. Kết quả đạt được

- Hoàn thành báo cáo chuyên đề học phần Học Sâu.
- Xây dựng được phần mềm phát hiện cháy khói, đơn giản phù hợp.

Giảng viên hướng dẫn

TS. Đoàn Thị Hương Giang

Sinh viên thực hiện

Nguyễn Mạnh Hưng

Phạm Đức Mạnh

PHIẾU CHẤM ĐIỂM

Sinh viên thực hiện

Họ và tên	Điểm	Chữ ký	Ghi Chú
Nguyễn Mạnh Hưng 20810310347			
Phạm Đức Mạnh 20810310398			

Giảng viên chấm

Họ và tên	Chữ ký	Ghi chú
Giảng viên chấm 1		
Giảng viên chấm 2		

MỤC LỤC

DANH MỤC HÌNH ẢNH

LỜI NÓI ĐẦU

Hiện nay, trên thế giới cũng như ở Việt Nam, mỗi năm có hàng nghìn vụ cháy, gây thiệt hại lớn về người và kinh tế. Với tốc độ xây dựng cơ sở hạ tầng vô cùng mạnh mẽ, các toà nhà cao tầng, trung tâm thương mại, trụ sở văn phòng luôn là những địa điểm tiềm ẩn những nguy cơ về hỏa hoạn. Vì vậy, việc tìm kiếm và phát triển những phương pháp phát hiện sớm các khu vực sắp cháy, cháy nhỏ một cách chính xác, kịp thời là thực sự cấp thiết.

Hiện tại, đã có nhiều nghiên cứu đề xuất những biện pháp phát hiện và cảnh báo cháy như dùng các đầu báo cháy nhiệt, đầu báo cháy khói và đầu báo cháy lửa. Tuy vậy, phương pháp này vẫn có hạn chế là các đầu báo cháy chỉ làm việc khi nhiệt độ, khói đã lan truyền tới đầu cảm biến và đạt ngưỡng hoạt động, khi đó thường đám cháy đã phát triển lớn. Vì vậy, hệ thống chỉ hiệu quả trong không gian nhỏ và kín (như trong toà nhà), phát hiện cháy khi đám cháy bùng phát không nhanh, còn với vùng giám sát có không gian mở như các hành lang, phòng không kín, ảnh hưởng gió thì hệ thống hoạt động kém hiệu quả.

Chính vì vậy, chúng em chọn đề tài "Nhận diện cháy khói sử dụng mô hình mạng YOLOv5", nhằm xây dựng một chương trình nhận diện cháy nổ cũng như tìm hiểu rõ hơn về cách hoạt động, những lợi ích và độ chính xác mà mô hình mạng YOLO đem lại.

CHƯƠNG 1: TỔNG QUAN VỀ BÀI TOÁN

1.1. Đặt vấn đề

Trong những năm gần đây, hiện tượng cháy nổ tại các chung cư cao tầng, cũng như các khu công nghiệp, nhà ở riêng lẻ của người dân đã ngày càng gia tăng kể cả về số lượng lẫn số người bị thương vong. Theo thống kê của quý I năm 2023, trên địa bàn cả nước xảy ra 405 vụ cháy, làm chết 16 người, bị thương 11 người. Về tài sản ước tính sơ bộ thành tiền khoảng 22,44 tỷ đồng. Ngoài ra, xảy ra 1.311 vụ sự cố sự cố chạm chập thiết bị điện trên cột điện, trong nhà dân, cháy cỏ, rác.

Chính vì vậy, việc phát hiện đám cháy sớm và chính xác đang là yêu cầu cấp thiết đặt ra đối với các hệ thống cảnh báo cháy. Việc phát hiện và cảnh báo cháy kịp thời sẽ góp phần quan trọng trong việc đảm bảo an toàn, giảm thiểu thiệt hại cho con người. Hiện nay, cách tiếp cận chủ yếu được sử dụng nhằm giải quyết vấn đề trên là sử dụng các cảm biến nhiệt độ, cảm biến khói. Phương pháp này có hạn chế là khi hệ thống phát hiện ra đám cháy thì tình trạng cháy đã lớn và lan rộng.

Thông qua bài báo cáo này chúng em đề xuất một hệ thống phát hiện khói nhằm cảnh báo sớm đám cháy được xây dựng dựa trên nền tảng công nghệ học máy. Quá trình xây dựng, thử nghiệm cho thấy tính khả thi của hệ thống trong việc giải quyết vấn đề phát hiện sớm và cảnh báo đám cháy. Kết quả thử nghiệm đã chứng tỏ rằng hệ thống đề xuất có thể đáp ứng tốt mục tiêu cảnh báo sớm các đám cháy và phù hợp để triển khai với các hệ thống máy tính hiện tại.

1.2. Giới thiệu về bài toán nhận diện cháy khói

Việc phát hiện và cảnh báo cháy sử dụng đầu dò còn nhiều hạn chế do phụ thuộc hoàn toàn vào thời gian lan truyền nhiệt và khói kể từ khi bắt đầu xảy ra hiện tượng cháy cho đến khi khói hoặc nhiệt độ lan tỏa tới đầu dò. Những năm gần đây, một hướng mở ra trong nghiên cứu cảnh báo cháy là sử dụng trực tiếp dữ liệu hình ảnh, video từ các hệ thống camera giám sát, sau đó đưa qua các thuật toán xử lý hình ảnh, các thuật toán nhận dạng AI, từ đó đưa ra các cảnh báo khi phát hiện đám cháy. Ưu điểm của hệ thống là có thể tận dụng dữ liệu từ các hệ thống camera giám sát có sẵn, tuy nhiên độ chính xác hệ thống phụ thuộc phần lớn vào thuật toán xử lý, yêu cầu cấu hình phần cứng mạnh. Phần lớn các giải pháp được đề xuất cho bài toán phát hiện ngọn lửa sử dụng kỹ thuật xử lý ảnh, video số hiện đều dựa trên những tính chất có thể quan sát được của ngọn lửa như màu sắc, sự thay đổi về vị trí các điểm ảnh của ngọn lửa theo thời gian.

Trong các kỹ thuật nhận dạng hình ảnh, giải pháp áp dụng các mô hình mạng nơ ron học sâu được đánh giá đạt hiệu năng tốt nhất. Bài báo cáo này sẽ trình bày cơ sở lý thuyết mạng YOLO và quy trình huấn luyện mô hình mạng YOLOv5 để nhận dạng ngọn lửa. Sử dụng mô hình đã được huấn luyện để xây dựng chương trình nhận dạng ngọn lửa, sau đó thực hiện các thực nghiệm nhận dạng từ đó đưa ra đánh giá hiệu năng của mô hình.

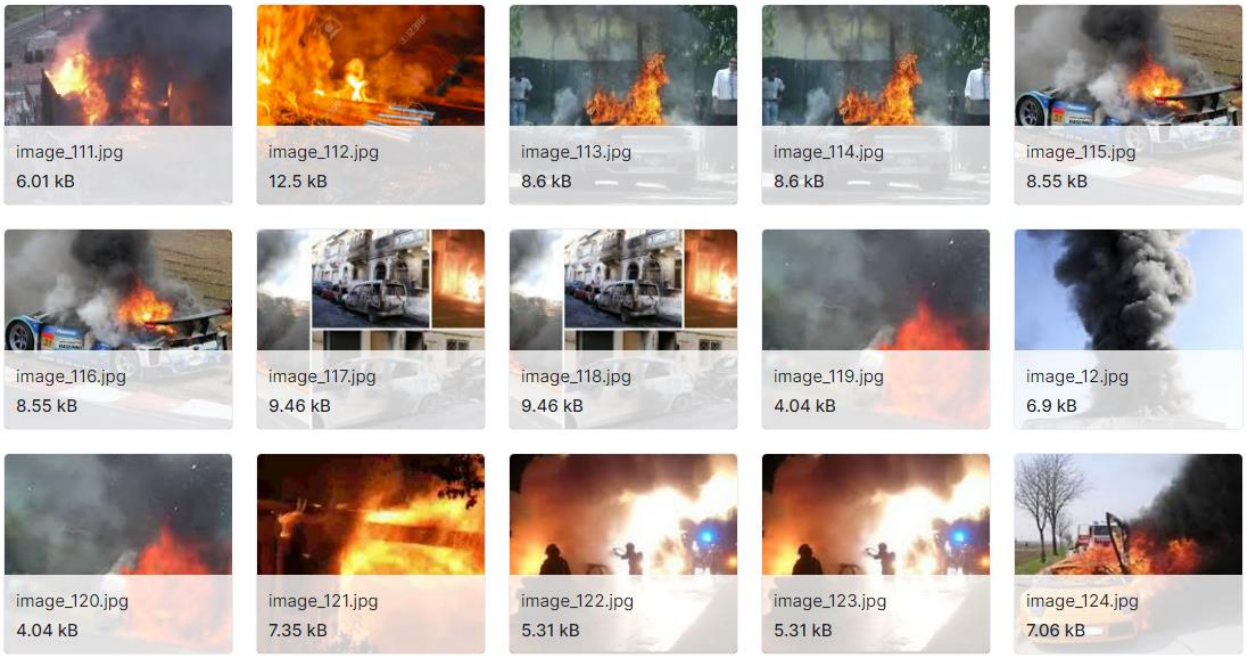
1.3. Một số CSDL về cháy khói đã được công bố

Qua quá trình tìm hiểu, chúng em đã thu thập được 04 bộ cơ sở dữ liệu (CSDL) về lửa và khói. Chi tiết được thể hiện trong Bảng 1 dưới đây:

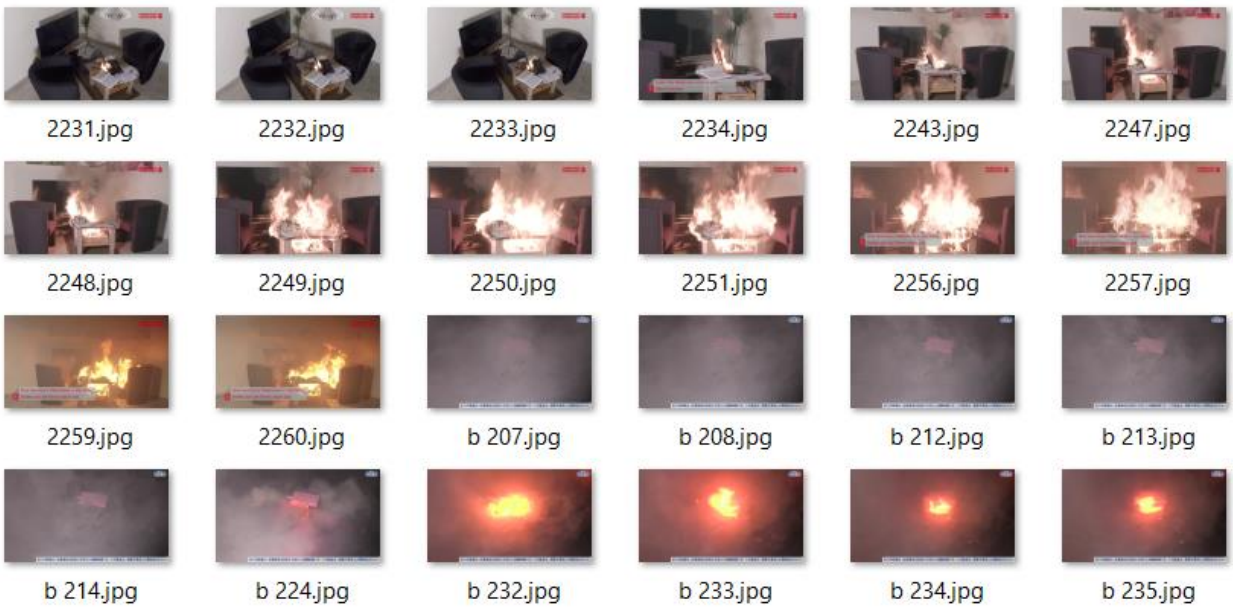
Bảng 1: CSDL về cháy khói

Tên CSDL	Link	Số lượng
Fire smoke and neutral [1]	https://www.kaggle.com/datasets/slirq123/fire-smoke-and-neutral	2000 ảnh
Fire and smoke bbox coco dateset [2]	https://www.kaggle.com/datasets/deeplearn1/fire-and-smoke-bbox-coco-dateset	1075 ảnh
Fire House [3]	https://zenodo.org/record/7649245#.ZBp0t8JBxhE	3320 ảnh
Fire on Road [4]	https://www.kaggle.com/datasets/tharakan684/urecamain	4000 ảnh

Các bộ cơ sở dữ liệu này chưa được gán nhãn và hình ảnh còn để riêng lẻ. Trong đó, bộ CSDL Fire smoke and neutral [1] có 1000 ảnh về lửa và 1000 ảnh về khói, với điều kiện nền, kích thước ảnh khác nhau như đã thể hiện trong hình 1.1 (a). Bộ CSDL Fire and smoke bbox coco dateset [3] có 3320 ảnh khác nhau về lửa, khói và ở nhiều vị trí có nền khác nhau, kích thước của các ảnh cũng khác nhau như thể hiện trong hình 1.1 (b)



a)



b)

Hình 1.1: CSDL cháy khói

Tuy nhiên, cả 04 bộ CSDL đều chưa được gán nhãn hoặc có gán nhãn, nhưng chưa được đầy đủ và lộn xộn. Do đó, trong phần chuẩn bị CSDL chúng em thực hiện lần lượt các bước:

- Download bốn bộ CSDL trên.

- Tiến hành chọn lựa kĩ càng những hình ảnh có chất lượng tốt với các vị trí và nền khác nhau.
- Sử dụng phần mềm MakesenseAI để gán nhãn cho lớp CSDL lửa là 0 và lớp CSDL khói là 1.

Sau khi tiến hành gộp và gán nhãn các hình ảnh, chúng em đặt tên cho bộ CSDL là OD-FireSmok-EPU. Bộ CSDL này sẽ được sử dụng để huấn luyện, thử nghiệm và đánh giá các mô hình YOLO trong Chương 3 của bài báo cáo.

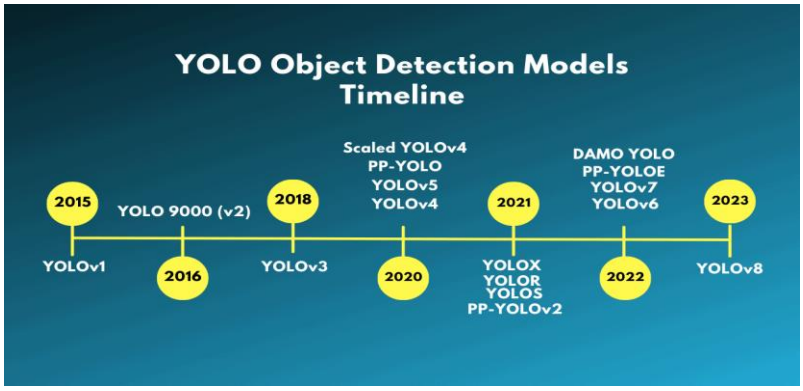
CHƯƠNG 2: TỔNG QUAN VỀ MÔ HÌNH MẠNG YOLO

2.1. Giới thiệu mạng YOLO

YOLO là một kiến trúc mạng CNN được sử dụng trong phát hiện, nhận dạng và phân loại đối tượng. Đối với bài toán phân loại (Classification) chỉ có khả năng phân loại đối tượng bằng cách dự đoán nhãn, thì YOLO có thể giải quyết bài toán phát hiện đối tượng (Object Detection), không chỉ có thể phát hiện nhiều đối tượng với nhiều nhãn khác nhau mà còn có thể xác định vị trí cụ thể của các đối tượng trong cùng một hình ảnh bằng các khung bao quanh đối tượng hình chữ nhật (Bounding Box).

YOLO là viết tắt của cụm từ "You only look once" nói nên khả năng về tốc độ nhận dạng của mô hình này, YOLO được đánh giá là mô hình cho tốc độ nhận dạng nhanh nhất có khả năng nhận dạng theo thời gian thực. Kiến trúc YOLO được xây dựng từ các lớp tích chập (Convolution layers) để trích xuất ra các đặc trưng của đối tượng và các lớp kết nối đầy đủ (full connected layer) để dự đoán nhãn và vị trí của đối tượng. Dữ liệu đầu vào là các hình ảnh, mô hình sẽ dự đoán vị trí, kích thước và nhãn của các Bounding Box.
Mô hình YOLO hiện tại có 8 phiên bản chính:

- YOLOv1, YOLO 9000 (v2), YOLOv3[5] do Joseph Redmon và Ali Farhadi phát triển từ 2015.
- YOLOv4[6] do Alexey Bochkovskiy phát triển năm 2020 và YOLOv5[7] do Glenn Jocher phát hành trong cùng năm 2020.
- YOLOv6[8] do Meituan Vision AI Department phát triển năm 2022 và YOLOv7[9] do Alexey Bochkovskiy phát triển cùng năm 2022.
- YOLOv8 được phát triển bởi Ultralytics – Công ty đã phát triển YOLOv5 vào năm 2023.



Hình 2.1: Thời gian ra đời của các phiên bản YOLO

Bài báo cáo này sẽ cung cấp một đánh giá toàn diện về sự phát triển của từng phiên bản, từ YOLOv1 - phiên bản ban đầu cho đến YOLOv8 - phiên bản mới nhất, nhằm làm sáng tỏ những đổi mới, điểm khác biệt và sự cải tiến qua mỗi phiên bản.

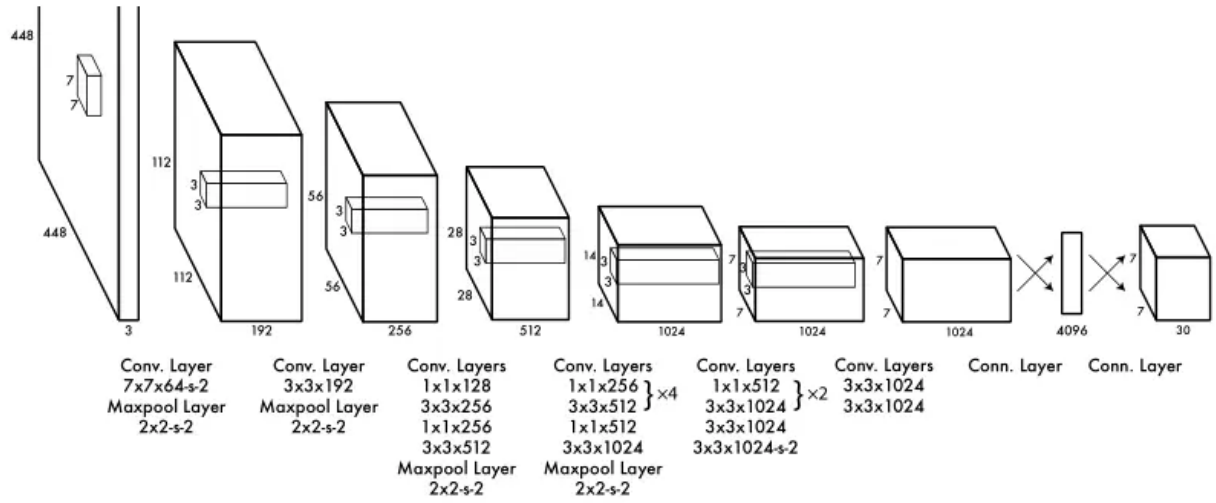
Ngoài việc thảo luận về những tiến bộ cụ thể của từng phiên bản YOLO, bài báo cáo còn nêu bật sự đánh đổi giữa tốc độ và độ chính xác đã xuất hiện trong suốt quá trình phát triển của khuôn khổ. Điều này nhấn mạnh tầm quan trọng của việc xem xét bối cảnh và yêu cầu của các ứng dụng cụ thể để lựa chọn mô hình YOLO phù hợp nhất.

2.2. Các phiên bản của YOLO v1 – v8

2.2.1. YOLOv1

Kiến trúc YOLOv1 coi bài toán phát hiện vật thể như một bài toán regression. Từ input là ảnh đầu vào, qua một mạng gồm các lớp convolution, pooling và fully connected là có thể ra được output. Kiến trúc này có thể được tối ưu để chạy trên GPU với một lần forward pass, và vì thế đạt được tốc độ rất cao. Tuy nhiên, YOLOv1 áp đặt các ràng buộc về không gian trên những bounding box, mỗi grid cell chỉ có thể predict rất ít bounding box (B) và duy nhất một class. Các ràng buộc này hạn chế khả năng nhận biết số object nằm gần nhau, cũng như đối với các object có kích thước nhỏ.

Ngoài ra, trong quá trình training, loss function không có sự đánh giá riêng biệt giữa error của bounding box kích thước nhỏ so với error của bounding box kích thước lớn. Việc coi chúng như cùng loại và tổng hợp lại làm ảnh hưởng đến độ chính xác toàn cục của mạng. Error nhỏ trên box lớn nhìn chung ít tác hại, nhưng error nhỏ với box rất nhỏ sẽ đặc biệt ảnh hưởng đến giá trị IOU.



Hình 2.2: Kiến trúc mạng YOLOv1

2.2.2. YOLOv2 & YOLO 9000

YOLOv2 đặt tên là YOLO9000 là bản cải tiến của YOLOv1, phiên bản này tốt hơn, nhanh hơn, tiên tiến hơn để bắt kịp faster R-CNN (phương pháp sử dụng Region Proposal Network), xử lý được những vấn đề gặp phải của YOLOv1.

- Thêm Batch Normalization: Kĩ thuật Batch Normalization được đưa vào sau tất cả các lớp convolution của YOLOv2. Kĩ thuật này không những giảm được thời gian huấn luyện, mà còn có tác dụng tăng tính phổ quát (generalize) cho mạng. Ở YOLOv2, Batch Normalization giúp tăng mAP lên khoảng 2%. Mạng cũng không cần sử dụng thêm Dropout để tăng tính phổ quát.
- High resolution classifier: YOLO được huấn luyện với 2 pha. Pha đầu sẽ huấn luyện một mạng classifier với ảnh đầu vào kích thước nhỏ (224x224) và pha sau sẽ loại bỏ lớp fully connected và sử dụng mạng classifier này như phần khung xương (backbone) để huấn luyện mạng detection.
- Sử dụng kiến trúc anchorbox để đưa ra dự đoán: Trong YOLOv2, loại bỏ lớp fully connected ở giữa mạng và sử dụng kiến trúc anchorbox để predict các bounding box. Việc dự đoán các offset so với anchorbox sẽ dễ dàng hơn nhiều so với dự đoán toạ độ bounding box. Thay đổi này làm giảm mAP đi một chút nhưng làm recall tăng lên.
- K-mean clustering cho lựa chọn anchor: Thay vì phải chọn anchorbox bằng tay, YOLOv2 sử dụng thuật toán k-means để đưa ra các lựa chọn anchorbox tốt nhất cho mạng. Việc này tạo ra mean IoU tốt hơn.
- Direct location prediction: YOLOv2 sử dụng hàm sigmoid để hạn chế giá trị trong khoảng 0 đến 1, từ đó có thể hạn chế các dự đoán bounding box ở xung quanh grid cell, từ đó giúp mô hình ổn định hơn trong quá trình huấn luyện.
- Add fine-grained features: YOLOv2 sử dụng feature map 13x13 để đưa ra các dự đoán. Faster R-CNN và SSD đưa ra dự đoán ở nhiều tầng khác nhau trong mạng để tận dụng các feature map ở các kích thước khác nhau. YOLOv2 cũng kết hợp các feature ở các tầng khác nhau lại để đưa ra dự đoán, cụ thể kiến trúc nguyên bản của YOLOv2 kết hợp feature map 26x26 lấy từ đoạn gần cuối với feature map 13x13 ở cuối để đưa ra các dự đoán.
- Multi-Scale Training: Sau khi thêm kĩ thuật anchorbox cho YOLOv2, input của mạng được thay đổi thành 416x416 thay vì 448x448. Tuy vậy, YOLOv2 được thiết

kể chỉ gồm các lớp convolution và pooling nên có thể thích ứng với nhiều kích thước ảnh đầu vào khác nhau. Việc huấn luyện mạng trên nhiều kích thước ảnh khác nhau để tăng khả năng thích ứng của YOLOv2 với đa dạng kích thước ảnh.

- Light-weight backbone: Điểm cải tiến của YOLOv2 còn phải kể đến backbone mới có tên Darknet-19. Mạng này bao gồm 19 lớp convolution và 5 lớp maxpooling tạo ra tốc độ nhanh hơn phiên bản YOLO trước.

YOLOv2 đưa ra cách kết hợp các dataset khác với ImageNet để có thể phát hiện nhiều class hơn. Một directed graph được tạo ra, gọi là WordTree. Nó được dung để có thể merge được các label từ tập ImageNet (1000 class) với COCO/PASCAL (100 class), dựa vào WordNet để có thể xây dựng quan hệ giữa các class, từ đó có thể huấn luyện mạng nhận dạng các class có quan hệ với nhau.

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Hình 2.3: Kiến trúc mạng YOLOv2

2.2.3. YOLOv3

YOLOv3 có kiến trúc khá giống YOLOv2 nhưng đã được nghiên cứu cải tiến. Các cải tiến đó bao gồm:

- Logistic regression cho confidence score: YOLOv3 predict độ tự tin của bounding box (có chứa vật hay không) sử dụng logistic regression
- Thay softmax bằng các logistic classifier rời rạc: YOLOv3 sử dụng các logistic classifier thay vì softmax cho việc classify đối tượng. Việc này cho hiệu quả tốt hơn

nếu các label không "mutually exclusive", tức là có thể có đối tượng cùng thuộc 2 hay nhiều class khác nhau.

- Backbone mới - Darknet-53: Backbone được thiết kế lại với việc thêm các residual blocks (kiến trúc sử dụng trong ResNet).
- Multi-scale prediction: YOLOv3 sử dụng kiến trúc Feature Pyramid Networks (FPN) để đưa ra các dự đoán từ nhiều scale khác nhau của feature map. Việc này giúp YOLOv3 tận dụng các feature map với độ thô - tinh khác nhau cho việc dự đoán.
- Skip-layer concatenation: YOLOv3 cũng thêm các liên kết giữa các lớp dự đoán. Mô hình upsample các lớp dự đoán ở các tầng sau và sau đó concatenate với các lớp dự đoán ở các tầng trước đó. Phương pháp này giúp tăng độ chính xác khi predict các object nhỏ.

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

Hình 2.4: Kiến trúc backbone của YOLOv3

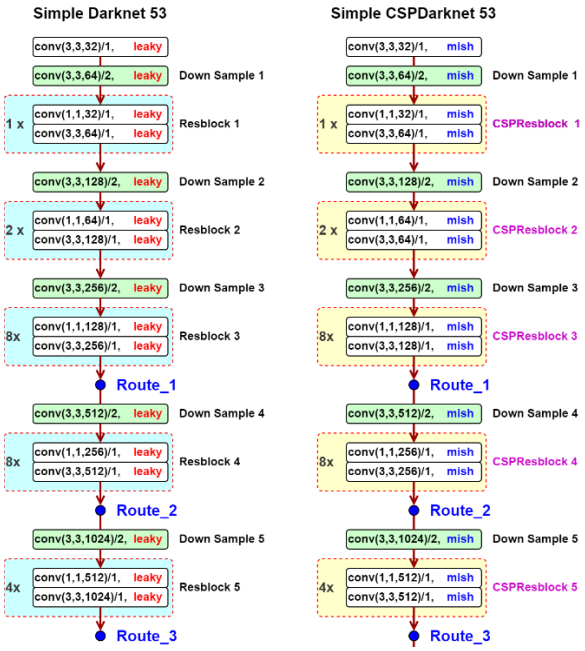
2.2.4. YOLOv4

Về kiến trúc, YOLOv4 bao gồm 3 phần chính:

- Backbone : CSPDarknet53
- Neck: SPP (Spatial Pyramid Pooling) và PAN (Path Aggregation Network)
- Head: YOLOv3

SPP (Spatial Pyramid Pooling) là một lớp gộp chung giúp loại bỏ ràng buộc kích thước cố định của mạng, tức là CNN không yêu cầu hình ảnh đầu vào có kích thước cố định. Cụ thể, chúng ta thêm một lớp SPP lên trên lớp tích chập cuối cùng. Lớp SPP tập hợp các tính năng và tạo ra các đầu ra có độ dài cố định, sau đó được đưa vào các lớp được kết nối đầy đủ (hoặc các bộ phân loại khác).

PAN (Path Aggregation Network) **nhằm mục đích thúc đẩy luồng thông tin trong khung phân đoạn phiên bản dựa trên đề xuất**. Cụ thể, hệ thống phân cấp tính năng được tăng cường với các tín hiệu bản địa hóa chính xác ở các lớp thấp hơn bằng cách tăng cường đường dẫn từ dưới lên, giúp rút ngắn đường dẫn thông tin giữa các lớp thấp hơn và tính năng trên cùng.



Hình 2.5: So sánh DarkNet53 với CSPDarkNet53

Ngoài ra, YOLOv4 còn sử dụng những BoF và BoS để tăng tốc quá trình training và cải thiện độ chính xác của mô hình như sau:

- Bag of Freebies (BoF) sử dụng cho phần Backbone:
  - CutMix và Mosaic data augmentation: Là hai kỹ thuật tăng cường dữ liệu được sử dụng để cải thiện hiệu suất của các mô hình học sâu. CutMix là kỹ thuật kết hợp hai hình ảnh bằng cách chọn ngẫu nhiên một phần của một hình ảnh và thay thế nó bằng một phần của một hình ảnh khác, trong khi Mosaic là kỹ thuật kết hợp nhiều hình ảnh thành một hình ảnh duy nhất. Cả hai kỹ thuật đều có

thể giúp ngăn chặn Overfitting và cải thiện hiệu suất tổng quát của các mô hình học sâu.

- DropBlock regularization: Trong bài toán Classification, để tránh hiện tượng Overfitting, người ta thường thêm lớp DropOut nhằm loại bỏ ngẫu nhiên 1 vài nơron ra khỏi quá trình training nhằm mục đích tránh cho các nơron quá phụ thuộc lẫn nhau dẫn đến mạng bị phức tạp. Tuy nhiên ở trong mạng Convolution thì bỏ đi ngẫu nhiên ở một số vị trí trong feature map không hợp lý vì các vị trí cạnh nhau trong feature map có tương quan cao với nhau nên bỏ đi những vị trí random như vậy sẽ không đem lại hiệu quả. Vì vậy DropBlock sẽ bỏ đi nhóm vị trí trong feature map thay vì chỉ bỏ đi một vị trí.
- Class label smoothing: Lớp cuối cùng của một mạng nơron thường là lớp Fully Connected dùng để dự đoán và đưa ra output. Ở lớp này người ta thường sử dụng softmax để normalize đầu ra và thu được xác xuất dự đoán của từng class. Ở đây, ý tưởng class-label smoothing được đưa ra để tránh việc khi training thì việc sử dụng nhãn mà trong đó có 1 class có giá trị bằng 1 còn lại tất cả class còn lại có giá trị bằng 0 thì hàm lỗi sẽ khuyến khích việc các class khác nhau trở nên phân biệt với nhau. Vì vậy người ta bỏ bớt 1 đại lượng ϵ từ class chính và thêm vào những class khác với công thức:

$$q_i = \begin{cases} 1 - \varepsilon \\ \dfrac{\varepsilon}{K-1} \end{cases} if\ i = y,\ \ otherwise \tag{2.1}$$

Điều này sẽ làm cho mô hình được khái quát hóa tốt hơn.

- Bag of Specials (BoS) sử dụng cho phần Backbone:
  - Mish activation: Tiền thân của hàm Mish này là hàm Swish, hàm Swish có công thức toán học như sau:

$$f(x) = x * sigmoid(x) \tag{2.2}$$

Hàm Mish kế thừa được những tính chất của hàm Swich như: mượt, liên tục, tự chỉnh lưu, đơn điệu. Với công thức như sau:

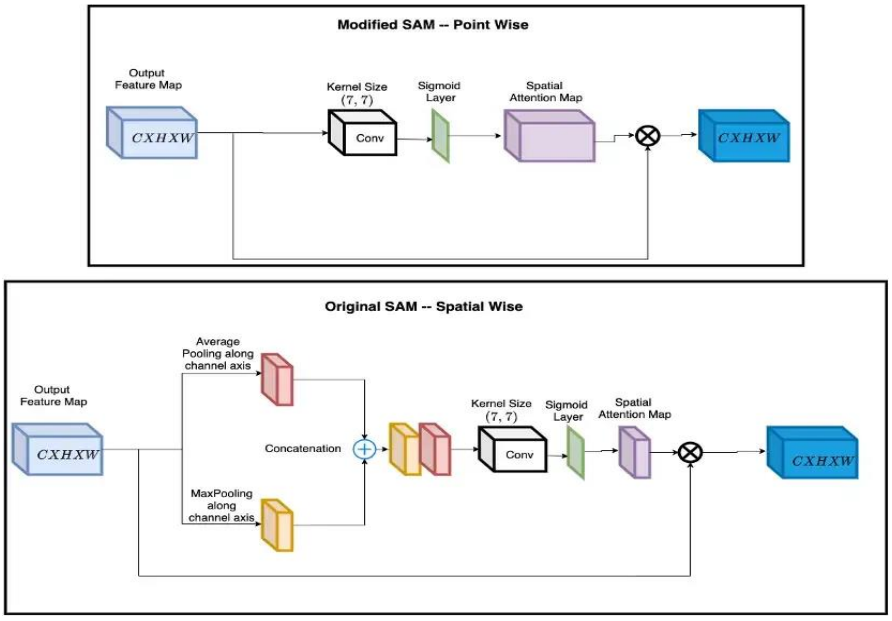
$$\begin{aligned} f(x) &= xtanh\big(softplus(x)\big) \\ &= xtanh(\ln(1 + e^x)) \end{aligned} \tag{2.3}$$

Khi training, hàm Mish có khả năng loại bỏ hiện tượng Dying ReLU, không bị chặn trên, tránh bão hòa, ngoài ra còn có đạo hàm liên tục. Vì những tính chất này, hàm Mish có khả năng tăng tốc quá trình training và hoạt động tốt hơn các hàm kích hoạt khác.

- CSP block: Ta có ý tưởng chính của CSPDenseBlock là tách feature map ra thành 2 phần, một phần sẽ cho đi qua những khối dense block với nhiều dense layer và một transition layer; phần còn lại sẽ được kết hợp với với transmitted feature map để mang sang stage tiếp theo. Vì vậy, với điều này thì trong mạng sẽ có 2 luồng gradient và feature map được độc lập với nhau để thực hiện những mục đích riêng.

  Mạng CSP không chỉ bảo toàn được lợi ích của việc sử dụng lại những feature characteristics của mạng DenseNet mà còn thêm khả năng ngăn chặn lượng thông tin gradient trùng lặp quá mức bằng cách cắt bớt dòng gradient. Nhờ vậy ta có thể tăng tốc quá trình training và còn giảm số lượng tham số cần tính toán để tăng tốc độ inference.

- SAM-block (Spatial Attention Module): Tập trung vào điểm thay vào việc biến đổi feature map bằng cách thêm những lớp average và max pooling. Tác giả sử dụng sigmoid activation function sau lớp CNN để tập trung hẳn vào những thông tin ngữ cảnh của vật thể, những giá trị không có vai trò trong dectection/classification sẽ bị loại bỏ. Cuối cùng là phép element-wise để thêm tham số attention vào trong ảnh.



Hình 2.6: SAM block trong YOLOv4

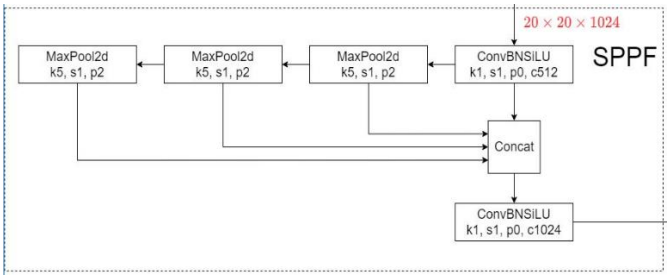
- PAN (Path Aggregation Network): một phiên bản cải tiến của FPN (Feature Pyramid Networks). Nếu FPNs rất linh hoạt và có tác dụng để lưu chuyển và kết hợp những local (low-level) features và global (high-level) features. Nó giúp kết hợp những semantic features đáng giá từ những level cao của pyramid và localization features từ những level thấp hơn để cho kết quả dự đoán cuối cùng.
- DIoU-NMS (Distance IoU): Vấn đề của IoU là khi 2 bounding box không trùng nhau thì mạng sẽ không học được gì vì loss=0. Ý tưởng của DIoU loss là sẽ đưa 2 bounding box không trùng lặp lại gần nhau, khoảng cách giữa tâm của các bounding box như 1 đại lượng phạt them trong khi IoU loss sẽ không cải thiện gì trong trường hợp này.

$$\mathcal{L}_{DIoU} = 1 - IoU + \frac{\rho^2(b, b^{gt})}{c^2} \tag{2.7}$$

2.2.5. YOLOv5

YOLOv5 không có quá nhiều thay đổi so với YOLOv4. YOLOv5 tập trung vào tốc độ và độ dễ sử dụng. Các thay đổi trong YOLOv5 như sau:

- Backbone:
  - C3 module: YOLOv5 cải tiến CSPResBlock của YOLOv4 thành một module mới, ít hơn một lớp Convolution gọi là C3 module.
  - Activation function: YOLOv4 sử dụng Mish còn ở phiên bản YOLOv5, activation function được sử dụng là SiLU.
- Neck:
  - SPPF: YOLOv5 áp dụng một module giống với SPP, nhưng nhanh hơn gấp đôi và gọi đó là SPP - Fast (SPPF). Thay vì sử dụng MaxPooling song song như trong SPP, SPPF của YOLOv5 sử dụng MaxPooling tuần tự. Hơn nữa, kernel size trong MaxPooling của SPPF toàn bộ là 5 thay vì là [5,9,13] [5,9,13] như SPP của YOLOv4.



Hình 2.7: Kiến trúc của module SPPF

- YOLOv5 trong việc giải quyết Grid Sensitivity lại sử dụng công thức

$$\begin{aligned} b_x &= 2\sigma(t_x) - 0.5 + c_x \\ b_y &= 2\sigma(t_y) - 0.5 + c_y \\ b_w &= p_w(2\sigma(t_w))^2 \\ b_h &= p_h(2\sigma(t_h))^2 \end{aligned} \tag{2.8}$$

- Trong xử lý data thì các kỹ thuật Data Augmentation được áp dụng là:
  - Mosaic Augmentation
  - Copy-paste Augmentation
  - Random Affine transform
  - MixUp Augmentation
  - Và các thay đổi về màu sắc cũng như là Random Flip của Albumentations
- EMA (Exponential Moving Average) Weight: Thông thường, khi training, ta sẽ ngay lập tức cập nhất weight của model trong quá trình backward pass. Tuy nhiên, trong EMA, ta tạo ra một model y hệt với model chúng ta sử dụng trong training gọi là momentum model, nhưng cách cập nhật weight sẽ tuân theo công thức Exponential Moving Average (EMA).

$$\theta' = \beta\theta' + (1-\beta)\theta \tag{2.9}$$

  Trong đó weight của mô hình gốc θ được coi như student và weight của mô hình momentum θ' được gọi là mean teachers.
- Loss scaling: YOLOv5 có sử dụng 3 đầu ra từ PAN Neck, để phát hiện objects tại 3 scale khác nhau. Tuy nhiên, sự ảnh hưởng của các object tại mỗi scale đến Objectness Loss là khác nhau, do đó, công thức của Objectness Loss được thay đổi như sau:

$$\mathcal{L}_{obj} = 4.0 * \mathcal{L}_{obj}^{small} + 1.0 * \mathcal{L}_{obj}^{medium} + 4.0 * \mathcal{L}_{obj}^{large} \tag{2.10}$$

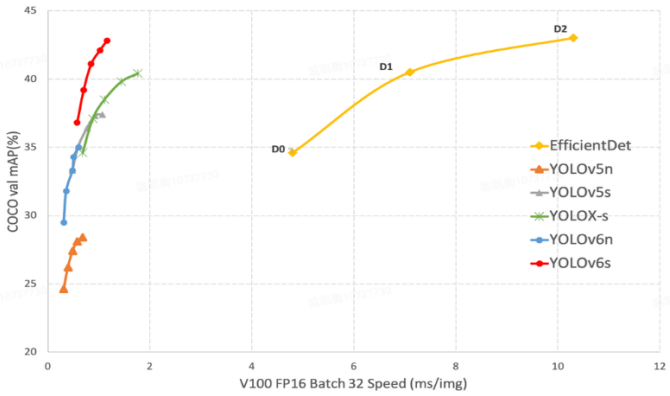
- Anchor Box: Anchor Box trong YOLOv5 nhận được 2 sự thay đổi lớn.
  - Đầu tiên là sử dụng Auto Anchor, một kĩ thuật áp dụng giải thuật di truyền (GA) vào Anchor Box ở sau bước k-means để Anchor Box hoạt động tốt hơn với những custom dataset của người dùng, chứ không còn hoạt động tốt ở mỗi COCO nữa.

- Thứ hai đó chính là offset tâm của object để lựa chọn nhiều Anchor Box cho một object.

2.2.6. YOLOv6

YOLOv6 được lấy cảm hứng từ kiến trúc YOLO giai đoạn ban đầu. YOLOv6 là khung phát hiện đối tượng một giai đoạn dành riêng cho các ứng dụng công nghiệp, với thiết kế hiệu quả thân thiện với phần cứng và hiệu suất cao. Nó vượt trội hơn YOLOv5 về độ chính xác phát hiện và tốc độ suy luận, khiến nó trở thành phiên bản hệ điều hành tốt nhất của kiến trúc YOLO cho các ứng dụng sản xuất.

YOLOv6s (màu đỏ) cung cấp Độ chính xác trung bình (mAP) tốt hơn tất cả các phiên bản trước của YOLOv5, với thời gian suy luận nhanh hơn khoảng 2 lần. Chúng ta cũng có thể thấy một khoảng cách hiệu suất rất lớn giữa kiến trúc dựa trên YOLO và EfficientDet, dựa trên phát hiện đối tượng hai giai đoạn.

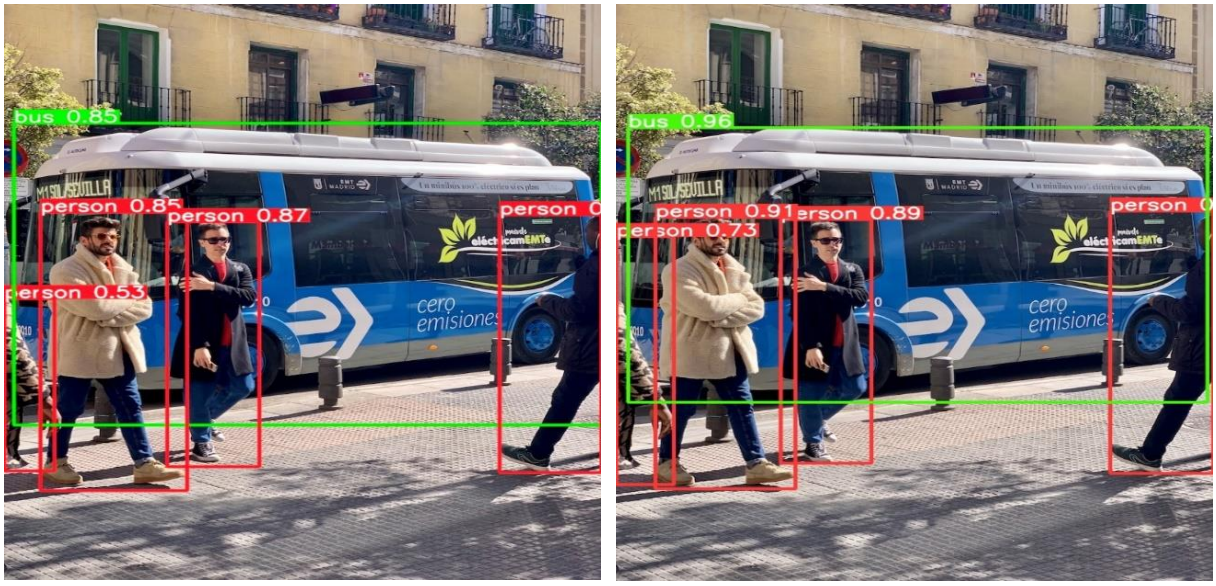


Hình 2.8: Biểu đồ so sánh suy luận ảnh đơn

Giống như trong suy luận hình ảnh đơn, YOLOv6 cung cấp kết quả tốt hơn cho video trên tất cả các phổ FPS.

Hình 2.9: Biểu đồ so sánh suy luận video

Ta tiến hành so sánh 2 phiên bản YOLOv5 và YOLOv6 để thấy được sự khác biệt:



YOLOv5s YOLOv6s

Chúng ta có thể thấy rõ rằng YOLOv6s phát hiện nhiều đối tượng hơn trong hình ảnh và có độ tin cậy cao hơn về nhãn của chúng.

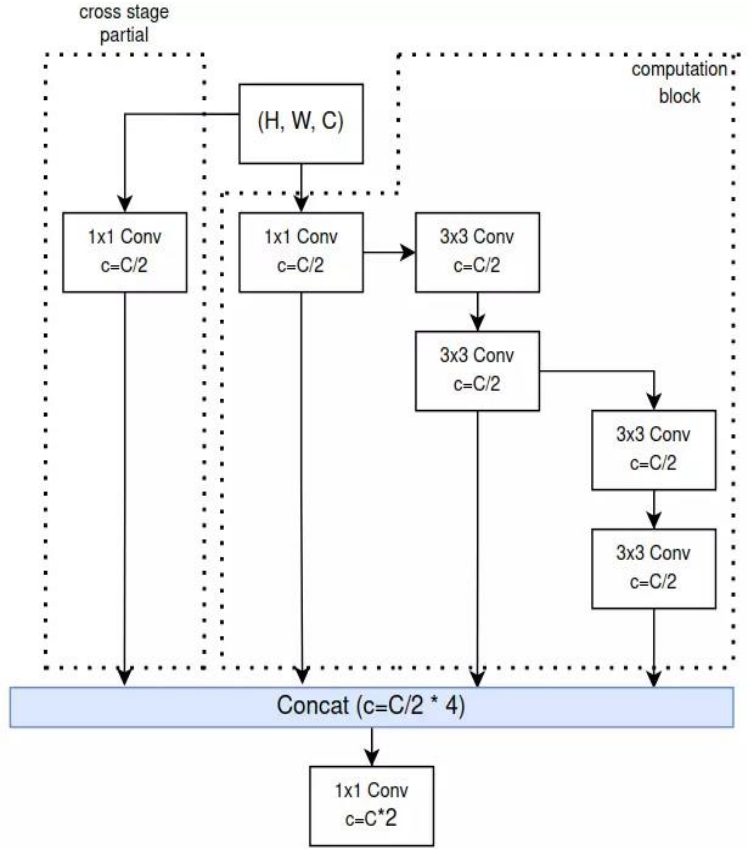
Nhìn chung YOLOv6 cung cấp kết quả tương đối tốt và cải thiện đáng kể trên tất cả các mặt trận so với các phiên bản YOLO trước đó.

2.2.7. YOLOv7

YOLOv7 đã đánh dấu sự trở lại nổi bật của YOLO. Theo phần tóm tắt của YOLOv7 ta có đoạn như sau: "YOLOv7 vượt qua mọi model Object Detection trong cả tốc độ và độ chính xác từ 5 FPS tới 160 FPS và đạt độ chính xác cao nhất với 56.8% AP trong số toàn bộ các model Object Detection real-time, có tốc độ 30 FPS hoặc hơn trên GPU V100. Và đương nhiên là YOLOv7 cũng vượt qua cả: YOLOR, YOLOX, Scaled-YOLOv4, YOLOv5, DETR, Deformable DETR, DINO-5scale-R50, ViT-Adapter-B cũng như là rất nhiều các mạng Object Detection khác cả về mặt tốc độ cũng như là độ chính xác. Hơn nữa, YOLOv7 được train trên COCO từ đầu mà không sử dụng bất kì pretrained nào."
Kiến trúc của YOLOv7 như sau:

- Backbone:

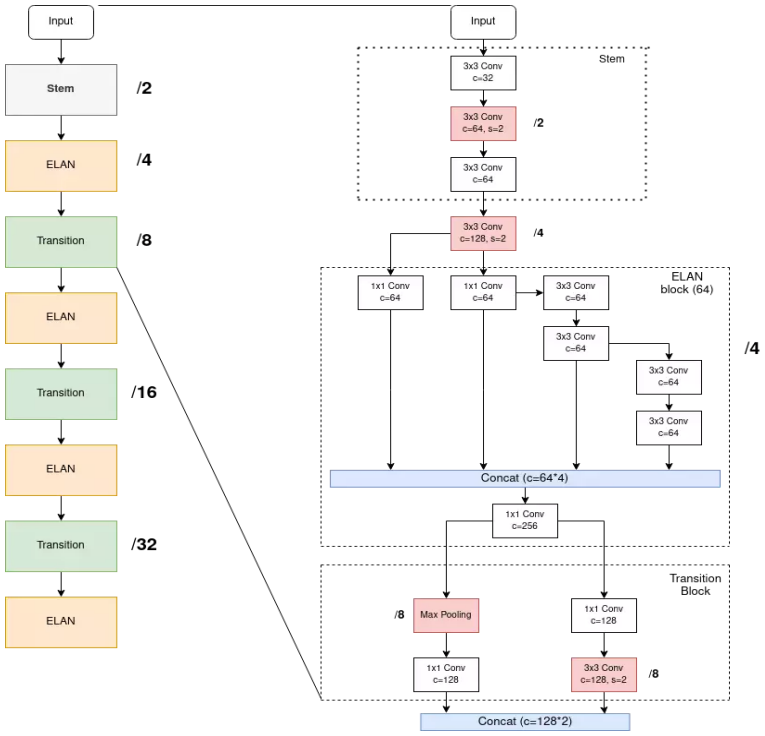
- ELAN (Efficient Layer Aggregation Network): Một ELAN block gồm 3 phần: Cross Stage Partial, Computation Block và phép PointWiseConv. Thiết kế của ELAN Block chịu ảnh hưởng từ 2 nghiên cứu trước đó là CSPNet và VoVNet. CSP hóa một block là việc tạo thêm một nhánh "cross stage partial". Computation Block là block chứa các lớp Conv được tính toán để sinh ra các feature mới thông qua các 3×33×3 Conv. Cuối cùng, các feature maps được tổng hợp lại ở cuối sử dụng toán tử concatenate trên chiều channel như VoVNet, và đưa qua PointWiseConv (1×11×1 Conv).



Hình 2.10: Cấu tạo của 1 ELAN Block

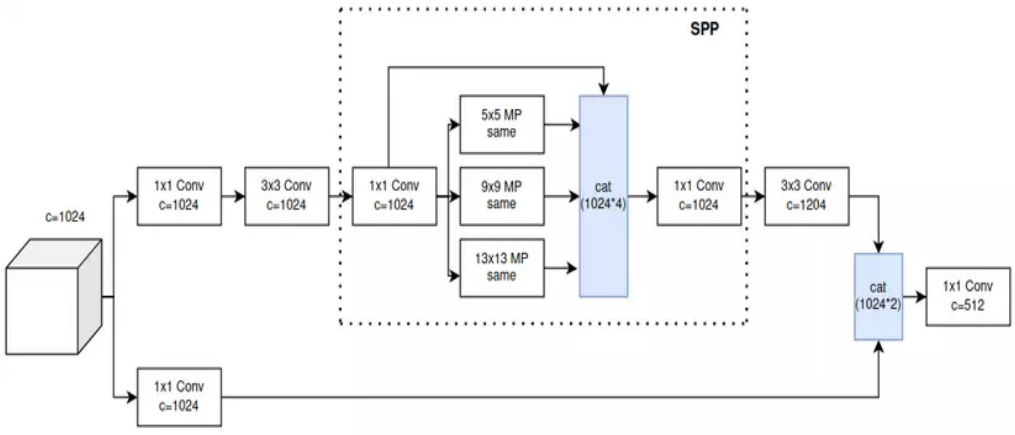
- Stem: Trước khi tiến vào ELAN Block đầu tiên trong backbone, ảnh đầu vào sẽ đi qua Stem Block
- Transition Block: Các ELAN Block được kết nối với nhau thông qua các Transition Block. Mỗi Transition Block là một lần giảm kích cỡ của feature maps đi 2 lần.

Backbone hoàn chỉnh của YOLOv7 là tập hợp của các ELAN Block và các Transition block



Hình 2.11: Backbone của YOLOv7

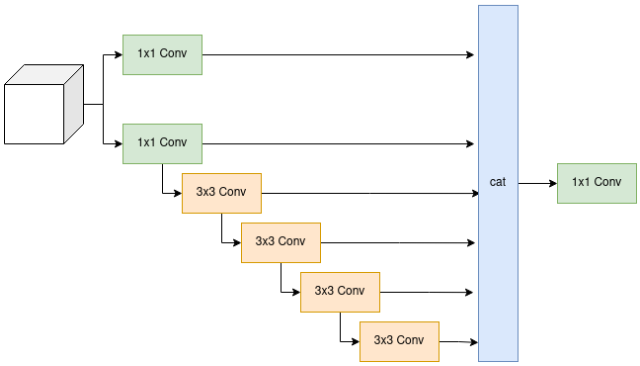
- Neck:
  - SPPCSPC: Trong YOLOv7, SPP tiếp tục được CSP hóa. SPPCSPC có 2 đường đều có số channels bằng với số channels trong feature maps ban đầu thay vì là một nửa như CSP thông thường.
    FLOPs và tốc độ của SPPCSPC so với SPP và tốc độ cho thấy SPPCSPC cực kì nặng và chậm.



Hình 2.12: Kiến trúc của SPPCSPC

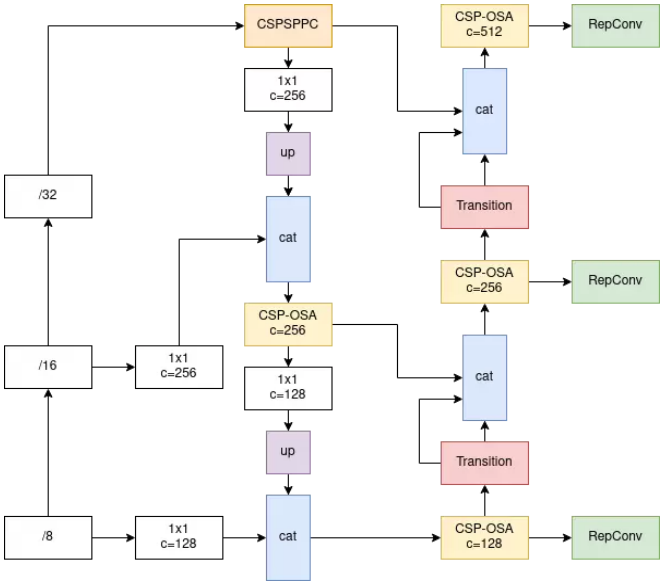
- (CSP-OSA) PANet: Ở YOLOv7, toán tử cộng để kết hợp 2 feature maps từ 2 scale vào với nhau đã được thay thế bằng toán tử concatenate. Hơn nữa, sau khi kết hợp với feature maps từ scale trên, sinh ra một feature maps trung gian, feature maps trung gian này tiếp tục được xử lý bằng CSP-OSA module thay vì được tiếp tục kết hợp thẳng với feature maps từ scale dưới như FPN hay PANet thông thường.

  OSA module được CSP hóa có cấu trúc:



Hình 2.13: CSP-OSA module sử dụng trong neck của YOLOv7

- RepConv: Là 1 module với tốc độ của 3×33×3 Conv nhưng độ chính xác thì cao hơn rất nhiều. RepConv sử dụng kĩ thuật Re-param để có một module tốc độ cao mà độ chính xác cũng cao



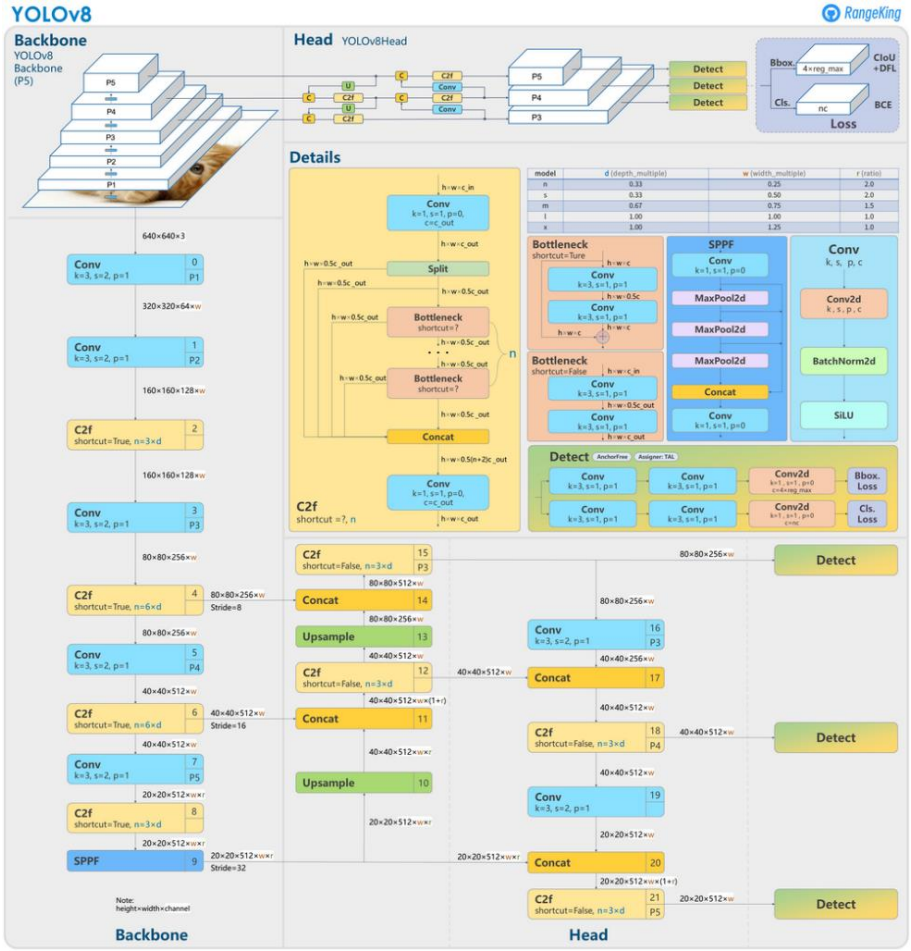
Hình 2.14: Neck trong YOLOv7

- Head: YOLOR và Auxiliary head

2.2.8. YOLOv8

YOLOv8 là phiên bản mới nhất của mô hình phát hiện đối tượng và phân đoạn hình ảnh YOLO. Là một mô hình tiên tiến, hiện đại (SOTA), YOLOv8 được xây dựng dựa trên sự thành công của các phiên bản trước, giới thiệu các tính năng và cải tiến mới để nâng cao hiệu suất, tính linh hoạt và hiệu quả.

YOLOv8 được thiết kế tập trung mạnh vào tốc độ, kích thước và độ chính xác, làm cho nó trở thành một lựa chọn hấp dẫn cho các tác vụ AI tầm nhìn khác nhau. Nó hoạt động tốt hơn các phiên bản trước bằng cách kết hợp các cải tiến như backbone network mới, anchor-free split head mới và loss functions mới. Những cải tiến này cho phép YOLOv8 mang lại kết quả vượt trội, trong khi vẫn duy trì kích thước nhỏ gọn và tốc độ vượt trội.



Hình 2.15: Cấu trúc YOLOv8

- Anchor Free Detection: YOLOv8 là một mô hình anchor-free. Điều này có nghĩa là nó dự đoán trực tiếp tâm của một đối tượng thay vì độ lệch từ một anchor box đã biết. Với công thức tính:

$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$
$$b_w = p_w e^{t_w} \quad (2.11)$$
$$b_h = p_h e^{t_h}$$

- Closing the Mosaic Augmentation

2.2.9. So sánh giữa các phiên bản

Chúng ta đã cùng nhau tìm hiểu về 8 phiên bản của YOLO, từ mô hình ban đầu cho đến mô hình mới nhất. Thông qua bảng dưới đây, chúng ta sẽ có một cái nhìn tổng quan hơn để có thể đưa ra đánh giá giữa các phiên bản:

Bảng 2: Tóm tắt kiến trúc 8 phiên bản YOLO

Version	Date	Anchor	Framework	Backbone	mAP (%)
YOLO	2015	No	Darknet	Darknet24	63.4
YOLOv2	2016	Yes	Darknet	Darknet24	63.4
YOLOv3	2018	Yes	Darknet	Darknet53	36.2
YOLOv4	2020	Yes	Darknet	CSPDarknet53	43.5
YOLOv5	2020	Yes	Pytorch	Modified CSP v7	55.8
YOLOv6	2022	No	Pytorch	EfficientRep	52.5
YOLOv7	2022	No	Pytorch	RepConvN	56.8
YOLOv8	2023	No	Pytorch	YOLO v8	53.9

- **Anchors**: Mô hình YOLO ban đầu tương đối đơn giản và không sử dụng Anchor, trong khi mô hình hiện đại nhất dựa vào máy dò hai giai đoạn có Anchor. YOLOv2 kết hợp các Anchor, giúp cải thiện độ chính xác của dự đoán từ Bounding box. Xu hướng này tồn tại trong 5 năm cho đến khi YOLOX giới thiệu một cách tiếp cận không cần Anchor để đạt được kết quả tiên tiến nhất. Kể từ đó, các phiên bản YOLO tiếp theo đã bỏ việc sử dụng Anchor.
- **Framework**: Ban đầu, YOLO được phát triển bằng cách sử dụng khung Darknet, và nó phù hợp với các phiên bản tiếp theo. Tuy nhiên, khi Ultralytics chuyển YOLOv3 sang PyTorch, các phiên bản YOLO còn lại được phát triển bằng PyTorch, dẫn đến sự gia tăng đột biến về các cải tiến. Một ngôn ngữ học sâu khác được sử dụng là PaddlePaddle, một khung nguồn mở ban đầu được phát triển bởi Baidu.

- **Backbone**: Kiến trúc Backbone của các mô hình YOLO đã trải qua những thay đổi đáng kể theo thời gian. Bắt đầu với kiến trúc Darknet, bao gồm các lớp tổng hợp tối đa và tích chập đơn giản, các mô hình sau này đã kết hợp các kết nối từng phần giữa các giai đoạn (CSP) trong YOLOv4 và tham số hóa lại trong YOLOv6 và YOLOv7.
- **Performance**: Mặc dù hiệu suất của các mô hình YOLO đã được cải thiện theo thời gian, nhưng điều đáng chú ý là họ thường ưu tiên cân bằng tốc độ và độ chính xác hơn là chỉ tập trung vào độ chính xác. Sự đánh đổi này là một khía cạnh thiết yếu của khuôn khổ YOLO, cho phép phát hiện đối tượng theo thời gian thực trên các ứng dụng khác nhau.

CHƯƠNG 3: CHƯƠNG TRÌNH

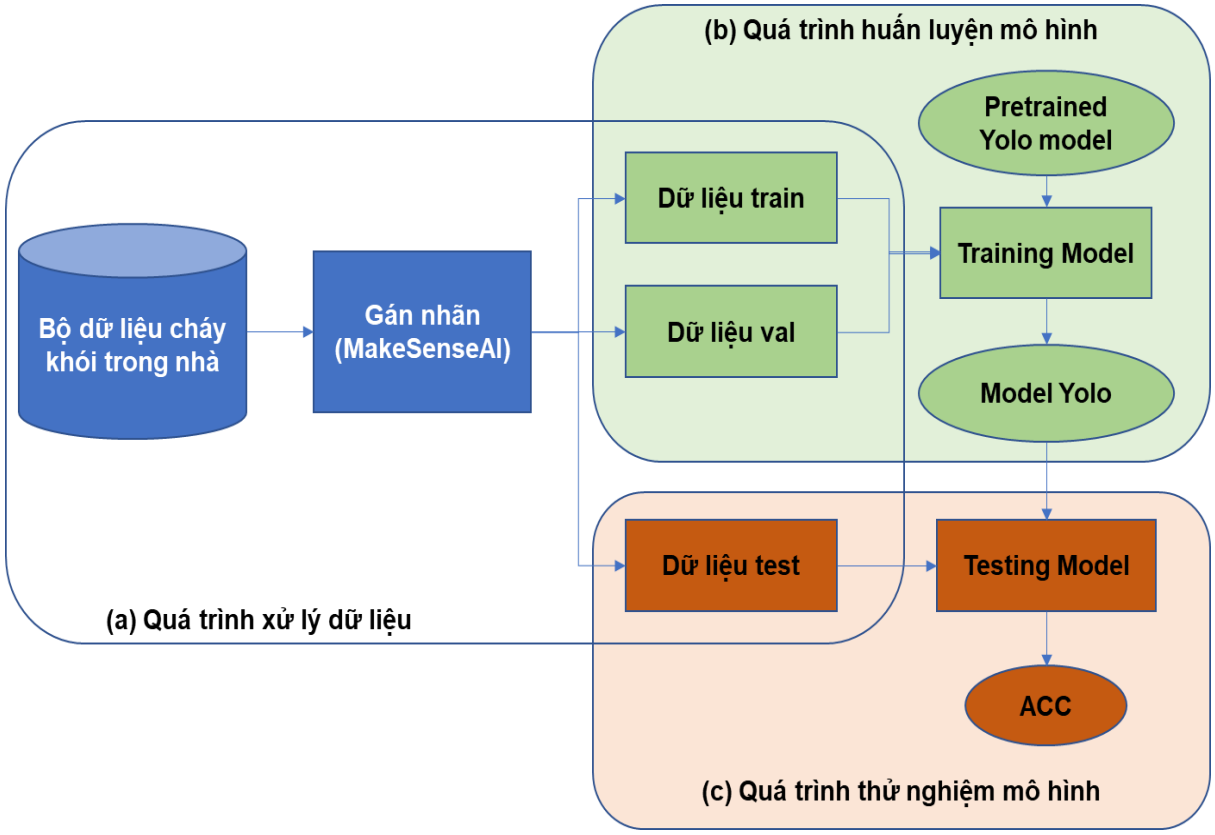
3.1. Đánh giá các phiên bản YOLO cho bài toán phát hiện cháy nổ

3.1.1. Giao thức thực hiện đánh giá mô hình

Chia toàn bộ tập dữ liệu thành 10 phần bằng nhau một cách ngẫu nhiên. Sử dụng phương pháp KFold với K = 10 để thực hiện training và testing các phiên bản YOLOv3, 4, 5. Với mô hình đánh giá được biểu diễn ở mục 3.1.2. Trong đó, mỗi lần thử nghiệm sẽ lấy 8 phần dữ liệu để train, 1 phần để val, 1 phần để test. Kết quả của 1 phần test là $ACC_i (i = 1,10)$. Như vậy, kết quả đánh giá của toàn bộ mô hình được tính như sau:

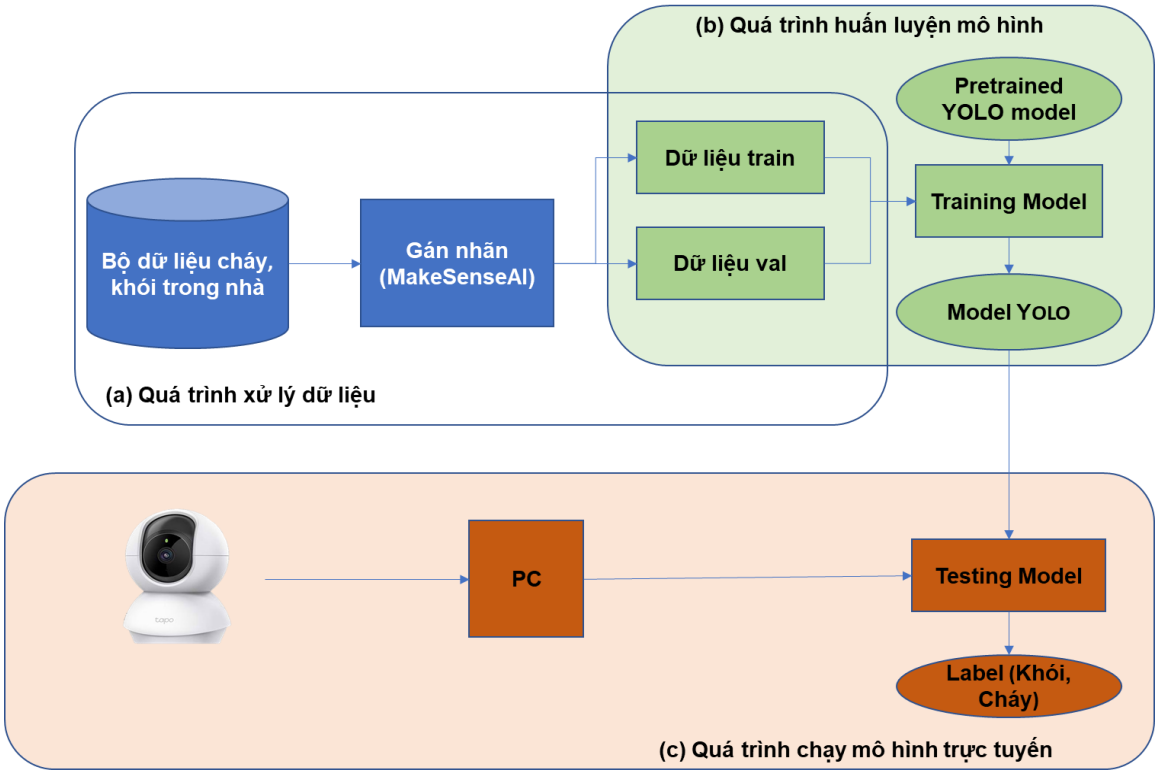
$$ACC_{YOLO} = \frac{\sum_{i=1}^{n=10} ACC_i}{10} \quad (3.1)$$

3.1.2. Mô hình quá trình đánh giá



Hình 3.1: Mô hình quá trình đánh giá

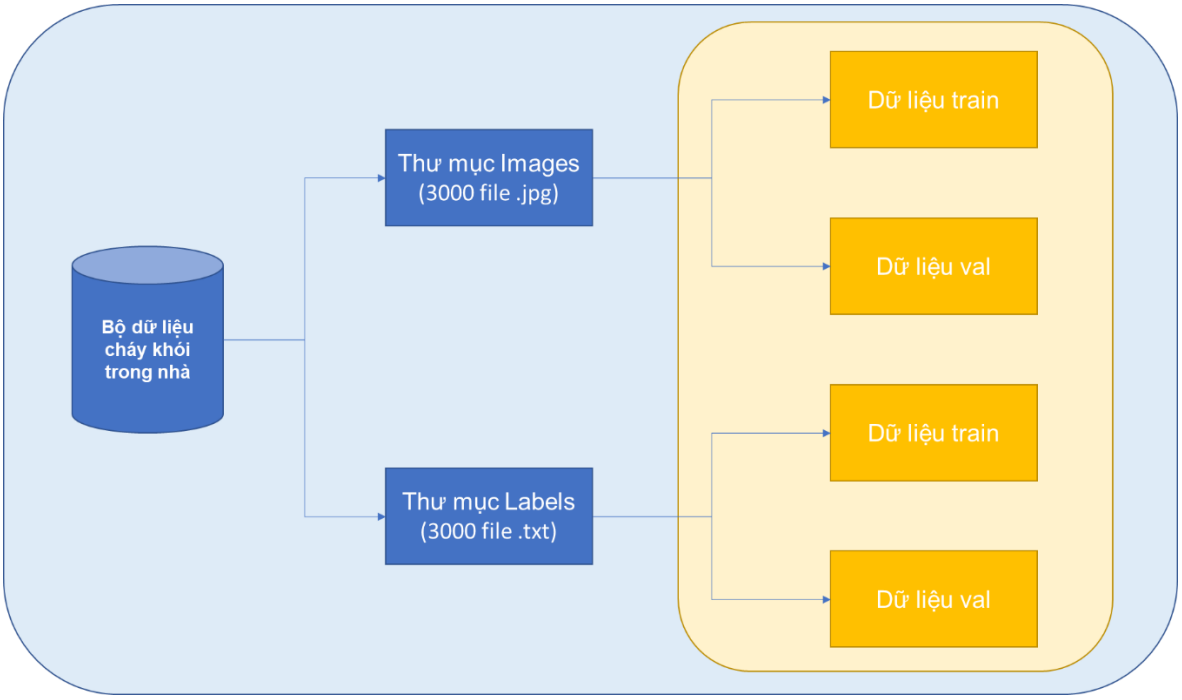
3.2. Triển khai mô hình hệ thống



Hình 3.2: Mô hình hệ thống khi triển khai

- Bộ dữ liệu hình ảnh (dataset) được chia thành 2 tập dữ liệu bao gồm tập huấn luyện (training) và tập xác minh (validate).
- Các hình ảnh sau đó được tiến hành gán nhãn nhằm xác định vị trí các ngọn lửa và khói.
- Tiến hành cấu hình các thông số cần thiết cho Model, sử dụng tập dữ liệu huấn luyện và tập validate để huấn luyện cho mô hình. Mô hình sau khi được huấn luyện thành công sẽ trả về kết quả là bộ trọng số được lưu dưới định dạng ".pt".
- Sau khi có mô hình huấn luyện, chúng ta sẽ sử dụng chúng cho chương trình nhận dạng ngọn lửa, khói với dữ liệu đầu vào lấy từ các hình ảnh, video hoặc camera.
- Kết quả của quá trình nhận dạng là hiển thị trực tiếp kết quả phân loại "cháy", "khói" và vị trí của ngọn lửa, khói trên màn hình. Đồng thời lưu lại kết quả nhận dạng.

3.3. Chuẩn bị bộ dữ liệu



Hình 3.3: Dữ liệu đầu vào

Bằng việc sử dụng MakeSenseAI để thực hiện gán nhãn 3000 hình ảnh, kết quả chúng em thu được là 3000 file nhãn (.txt). Sau đó, chúng em tiến hành chia bộ dữ liệu thành 2 thư mục: Thư mục Images (.jpg) chứa 3000 hình ảnh và Thư mục Labels (.txt) chứa 3000 file nhãn.

Với bộ dữ liệu trên, chúng em sẽ tiến hành thử nghiệm trên 3 phiên bản YOLOv3, YOLOv5 và YOLOv6.

3.4. Huấn luyện mô hình

Trong quá trình huấn luyện, mạng nơ ron sẽ tính toán lần lượt tất cả các ảnh của tệp dữ liệu đầu vào và sử dụng lặp lại các ảnh này nhiều lần, mục đích là để tối ưu hàm mất mát. Quá trình tối ưu này sẽ giúp cho mạng nơ ron tìm được bộ trọng số tốt nhất, giúp cho quá trình nhận dạng được chính xác nhất.

Mô hình sau khi huấn luyện thành công sẽ cho ra kết quả là bộ trọng số được lưu trữ trong file exp có định dạng ".pt". YOLOv5 còn cho phép người dùng sử dụng bộ trọng số có sẵn để tiếp tục quá trình huấn luyện, nhằm giúp cho quá trình huấn luyện nhanh hơn thay vì phải huấn luyện lại từ đầu.

Quá trình huấn luyện mô hình sử dụng một lượng lớn tài nguyên của máy tính, vì vậy nếu máy tính có cấu hình không đủ mạnh, thời gian huấn luyện sẽ diễn ra rất dài. Ta có thể huấn luyện trên Google Colab để giúp giảm thời gian huấn luyện, đồng thời đảm bảo hiệu quả và chất lượng của mô hình.

Trước khi thực hiện huấn luyện mô hình, chúng ta cần phải cấu hình lại các tham số cho phù hợp. Dưới đây là bảng danh mục các tham số cần phải cấu hình của mạng:

Bảng 3: Danh mục các tham số cần phải cấu hình của mạng

Tên tham số	Giá trị	Ý nghĩa
img	640x480	Kích thước ảnh đầu vào
Batch size	16	Số lượng ảnh sử dụng cho mỗi vòng huấn luyện
epochs	50	Số lượt huấn luyện trên toàn bộ dữ liệu
data	dataset	Đường dẫn đến file chứa tập dữ liệu
weight	yolov3-tiny.pt, yolov5s.pt, yolov6s.pt	Lựa chọn phiên bản model yolov5s
device	0	Chọn GPU để training, chọn 'CPU' nếu không có GPU
name	defaut	Đường dẫn thư mục lưu mode

3.5. Thử nghiệm và đánh giá kết quả

Trước khi thực hiện nhận dạng, chúng ta cần phải cấu hình lại các tham số. Dưới đây là bảng danh mục các tham số cần phải cấu hình trước khi bắt đầu thử nghiệm:

Bảng 4: Danh mục các tham số cần phải cấu hình để phục vụ cho việc nhận dạng

Tên tham số	Giá trị	Ý nghĩa
img	640x480	Kích thước ảnh đầu vào
source	0	Chọn '0' đối với dữ liệu nhận dạng từ camera. Đường dẫn đối với dữ liệu là hình ảnh, video có sẵn.

conf-thres	0.25	Thiết lập ngưỡng độ tin cậy dự đoán
iou-thres	0.45	Thiết lập ngưỡng IoU
weight	best.pt	Lựa chọn bộ trọng số tốt nhất đạt được từ kết quả training model

Kết thúc quá trình huấn luyện, kết quả trả về là 2 bộ trọng số được mô hình lưu lại bao gồm: Bộ trọng số tốt nhất (best.pt) và Bộ trọng số cuối cùng của quá trình huấn luyện (last.pt). Trong đó. bộ trọng số best.pt sẽ được sử dụng cho chương trình nhận dạng.

3.5.1. Thử nghiệm và đánh giá kết quả

Để đánh giá hiệu năng của mô hình, chúng em đánh giá hai thông số bao gồm độ chính xác trong nhận dạng và tốc độ nhận dạng. Đánh giá độ chính xác trong nhận dạng chúng ta cần phải sử dụng hai thông số gồm:

- Precision: Biểu thị độ chính xác trong dự đoán tên và vị trí đối tượng.
- Recall: Biểu thị khả năng phát hiện đối tượng trong dữ liệu đầu vào.

Đối với hệ thống này, phát hiện ngọn lửa là nhiệm vụ quan trọng nhất, do đó tham số Recall được quan tâm hơn Precision. Recall càng cao thì khả năng phát hiện lửa càng cao đồng nghĩa là khả năng bỏ sót lửa càng thấp.

Độ chính xác dự đoán đối tượng:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.2}$$

Khả năng phát hiện đối tượng:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3.3}$$

Trong đó:

- TP - True Positive: Thực tế có đối tượng, dự đoán có đối tượng.
- FN - False Negative: Thực tế có đối tượng, dự đoán không có đối tượng.
- TN - True Negative: Thực tế không có đối tượng, dự đoán không có đối tượng.
- FP - False Positive: Thực tế không có đối tượng, dự đoán có đối tượng.

Ở bài báo cáo này, chúng em sẽ tiến hành thử nghiệm và so sánh kết quả trên 3 phiên bản YOLOv3, YOLOv5 và YOLOv6. Kết quả so sánh được thực hiện với các điều kiện giống nhau như: sử dụng cùng 1 bộ dữ liệu huấn luyện, cài đặt cùng 1 bộ tham số (IoU, image size), thực thi trên cùng một nền tảng phần cứng.

Hình 3.4: Nhận diện cháy, khói từ ảnh, video

Hình 3.5: Nhận diện cháy, khói trực tiếp từ camera

3.5.2. So sánh kết quả thử nghiệm trên 3 phiên bản

Bảng 5: So sánh kết quả giữa các phiên bản

Nhận diện	YOLOv3-tiny		YOLOv5s		YOLOv6s	
	Precision	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
Cháy	89%	87%	91%	93%	52%	50%
Khói	85%	73%	88%	82%	38%	25%

Kết quả trên cho ta thấy hiệu năng của phiên bản YOLOv5s tốt hơn so với phiên bản YOLOv3-tiny và YOLOv6s trong khả năng phát hiện thể hiện qua tham số Recall. Lưu ý rằng, trong ứng dụng phát hiện và cảnh báo cháy, tham số Recall cần được ưu tiên cao hơn.

Bảng 6: Bảng so sánh thời gian huấn luyện và chạy 3 mô hình chế độ offline

STT	Phiên bản	Thời gian huấn luyện model (tiếng)	Thời gian chạy một frame
1	YOLOv3-tiny	2.5	31ms
2	YOLOv5s	4.7	19ms
3	YOLOv6s	5.5	13ms

Bảng 7: Bảng so sánh tốc độ nhận dạng giữa các phiên bản ở chế độ online

Phiên bản	CPU	Tốc độ (fps)	GPU	Tốc độ (fps)
YOLOv3-tiny	11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz	**50,2**	NVIDIA GeForce RTX 3050 Laptop GPU	**31,6**
YOLOv5s		**70,4**		**51,8**
YOLOv6s		**90,3**		**73,5**

Bảng 7 cho ta thấy giải pháp sử dụng phiên bản YOLOv5s cho tốc độ xử lý nhanh hơn nhiều so với giải pháp sử dụng YOLOv3-tiny; YOLOv6s tuy cho tốc độ xử lý nhanh hơn YOLOv5s nhưng tốn nhiều thời gian huấn luyện hơn khi sử dụng chung một bộ dữ

liệu và tham số. Qua đó, cho thấy mô hình sử dụng YOLOv5s hoàn toàn có thể ứng dụng trong các hệ thống báo cháy dân dụng sử dụng phần cứng có cấu hình trung bình.

=> **Kết luận**: Từ kết quả thực nghiệm, ta thấy giải pháp sử dụng phiên bản YOLOv5s cho hiệu năng tốt hơn so với sử dụng YOLOv3-tiny và YOLOv6s về độ chính xác và tốc độ nhận dạng trong các điều kiện khác nhau. Hệ thống có khả năng phát hiện, nhận dạng tốt với điều kiện trong nhà và ngoài trời, đặc biệt với điều kiện trong nhà, hệ thống cho kết quả nhận dạng với độ chính xác lên tới Precision ≥ 95% và Recall ≥ 93%. Tốc độ nhận dạng nhanh 19ms/frame ảnh cho phép triển khai nhận dạng theo thời gian thực. Hệ thống hoạt động hiệu quả hơn với các đối tượng có kích thước lớn.

KẾT LUẬN

Thông qua bài báo này, chúng em đã mô tả hệ thống nhận dạng, cảnh báo sớm đám cháy thông qua phát hiện đám khói sử dụng công nghệ học máy. Hệ thống được xây dựng dựa trên nhu cầu thực thế hiện nay, bằng việc tận dụng hệ thống phần cứng phổ thông như camera giám sát, cảm biến nhiệt độ độ ẩm... Công nghệ học máy được ứng dụng nhằm tăng khả năng cũng như độ chính xác cho hệ thống.

Với thực trạng về cháy nổ hiện tại, việc sử dụng hệ thống này hứa hẹn sẽ cải thiện được hiệu quả, giảm thiểu thiệt hại. Mong rằng trong tương lai, hệ thống sẽ tiếp tục được phát triển kết hợp nhận diện khói và lửa đồng thời nhằm tăng khả năng phát hiện đám cháy trong những điều kiện khác nhau.

TÀI LIỆU THAM KHẢO

[1] https://www.kaggle.com/datasets/slirq123/fire-smoke-and-neutral
[2] https://www.kaggle.com/datasets/deeplearn1/fire-and-smoke-bbox-coco-dateset
[3] https://zenodo.org/record/7649245#.ZBp0t8JBxhE
[4] https://www.kaggle.com/datasets/tharakan684/urecamain
[5] Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." *arXiv preprint arXiv:1804.02767* (2018).
[6] Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao, 2020. YOLOv4: Optimal Speed and Accuracy of Object Detection.arXiv:2004.10934v1, 2020
[7] Jocher Glenn. YOLOv5 release v6.1. https://github.com/ ultralytics/yolov5/releases/tag/v6.1, 2022.
[8] C. Li, L. Li, H. Jiang, K. Weng, Y. Geng, L. Li, Z. Ke, Q. Li, M. Cheng, W. Nie, et al., “Yolov6: A single-stage object detection framework for industrial applications,” arXiv preprint arXiv:2209.02976, 2022.
[9] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, “Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors,” arXiv preprint arXiv:2207.02696, 2022.