RESEARCH ARTICLE

CẢI TIẾN MÔ HÌNH NHẬN DIỆN HÌNH ẢNH YOLO DỰA TRÊN THAY ĐỔI KIẾN TRÚC CỦA MÔ HÌNH

Phan Thị Ngọc Hân

Khoa Công nghệ thông tin, Trường Đại học Ngoại ngữ -Tin học TP.HCM

hanptn@huflit.edu.vn

TÓM TẮT— Object detection [1](phát hiện đối tượng) là một trong những đề tài được quan tâm rất lớn của Deep Learning (học sâu) bởi khả năng ứng dụng cao, dữ liệu dể chuẩn bị và kết quả ứng dụng thì cực kỳ nhiều.

Các thuật toán mới của Object detection (phát hiện đối tượng) như YOLO, Single Shot Detector có tốc độ khá nhanh và độ chính xác cao nên giúp cho Object detection (phát hiện đối tượng) có thể thực hiện được các tác vụ dường như là real time, thậm chí là nhanh hơn so với con người mà độ chính xác không giảm.

Bài nghiên cứu này nhằm mục đích cải thiện độ chính xác của mô hình nhận diện hình ảnh thông qua việc kết hợp kiến trúc của hai phiên bản YOLOV09 và YOLOV10 của mô hình YOLO.

Từ khóa— YOLOV9, YOLO10, cải tiến, độ chính xác, kiến trúc.

I. GIỚI THIỆU

Deep Learning (học sâu) có thể xem là một lĩnh vực con của Machine Learning(học máy)-ở đó máy tính sẽ học và cải thiện chính nó thông qua các thuật toán. Deep Learning được xây dựng dựa trên các khái niệm phức tạp hơn rất nhiều, chủ yếu hoạt động với mạng nơ-ron nhân tạo để bắt chước khả năng tư duy và suy nghĩ của bộ não con người.

Object detection (phát hiện đối tượng) là một bài toán liên quan đến việc khoanh một vùng quan tâm trong ảnh và phân loại vùng này tương tự như phân loại hình ảnh.

YOLO [2] (You Only Look Once) là một mô hình phát hiện đối tượng (object detection) nổi tiếng trong lĩnh vực thị giác máy tính và học sâu (deep learning). Được giới thiệu lần đầu vào năm 2016 bởi Joseph Redmon và nhóm nghiên cứu tại Đại học Washington, YOLO đã trở thành một trong những công cụ quan trọng cho các ứng dụng nhận diện, phân loại và theo dõi đối tượng trong thời gian thực như : Đếm số lượng vật thể, thanh toán tiền tại quầy hàng, chấm công tự động, phát hiện vật thể nguy hiểm như súng, dao, …và rất nhiều các ứng dụng khác.Có thể nói dường như bất kì lình vực nào cũng đều có thể ứng dụng được object detection (phát hiện đối tượng).

Các phiên bản tiếp theo của YOLO đã liên tục cải tiến về độ chính xác và tốc độ, đồng thời giảm thiểu sự phụ thuộc vào tài nguyên máy tính.

YOLOV9 và YOLO10 là hai phiên bản mới nhất của dòng mô hình YOLO, được phát triển bởi cộng đồng OpenCV và Ultralytics, và ra mắt vào năm 2024.

II. CÁC CÔNG TRÌNH LIÊN QUAN

A. KHÁI NIỆM VỀ ỨNG DỤNG YOLO

1. GIỚI THIỆU CHUNG

YOLO là thuật toán object detection (phát hiện đối tượng) nên mục tiêu của mô hình không chỉ là dự báo nhãn cho vật thể như các bài toán classification (phân loại) mà nó còn xác định location (vị trí của vật thể). Do đó YOLO có thể phát hiện được nhiều vật thể có nhãn khác nhau trong một bức ảnh thay vì chỉ phân loại duy nhất một nhãn cho một bức ảnh.

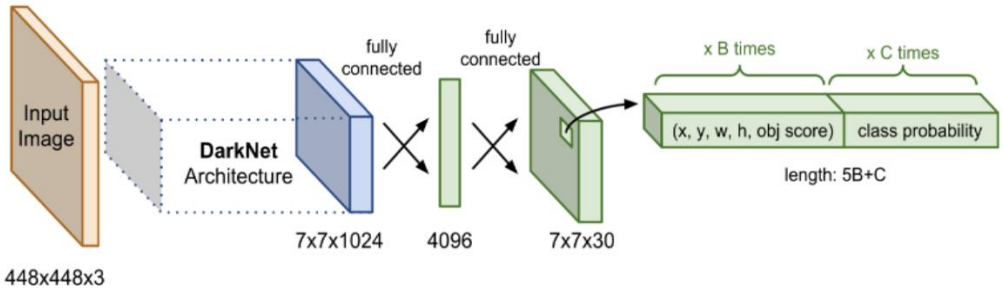
Các khái niệm liên quan bao gồm :

- Bounding box, amchor box: Bounding box là khung hình bao quanh vật thể. Anchor box là những khung hình có kích thước xác định trước, có tác dụng dự đoán bounding box.
- Fearture map (bản đồ đặc trưng) là một khối output mà ta sẽ chia nó thành một lưới ô vuông và áp dụng tìm kiếm và phát hiện vật thể trên từng cell.
- Non-max suppression: Phương pháp giảm thiểu nhiều boungding box chồng lấn nhau về 1 bounding box có xác xuất lớn nhất.

2. KIẾN TRÚC MẠNG YOLO

Kiến trúc YOLO bao gồm: base network (mạng cơ sở) là các mạng convolution (tích chập) làm nhiệm vụ trích xuất đặc trưng. Phần phía sau là những Extra Layers (lớp mở rộng) được áp dụng để phát hiện vật thể trên feature map (bản đồ đặc trưng) của base network (mạng cơ sở).

Base network (mạng cơ sở) của YOLO sử dụng chủ yếu là các convolutional layer (lớp tích chập) và các fully connected layer (lớp kết nối đầy đủ). Các kiến trúc YOLO cũng khá đa dạng và có thể tuỳ biến thành các phiên bản cho nhiều kích thước đầu vào khác nhau.



Hình 1. *Sơ đồ kiến trúc mạng YOLO. Thành phần Darknet Architechture được gọi là base netwwork (mạng cơ sở) có tác dụng trích xuất đặc trưng. Output của base network (mạng cơ sở) là một feature map (bản đồ đặc trưng) có kích thước 7x7x1024 sẽ được sử dụng làm cho input cho các Extra layers có tác dụng dự đoán*

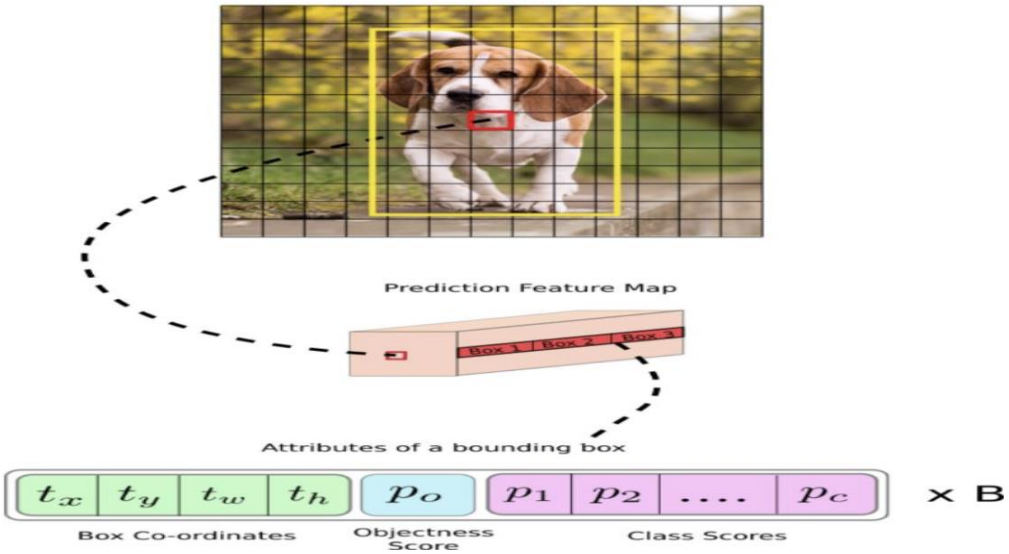
3. ĐẦU RA CỦA YOLO

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

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

Trong đó :

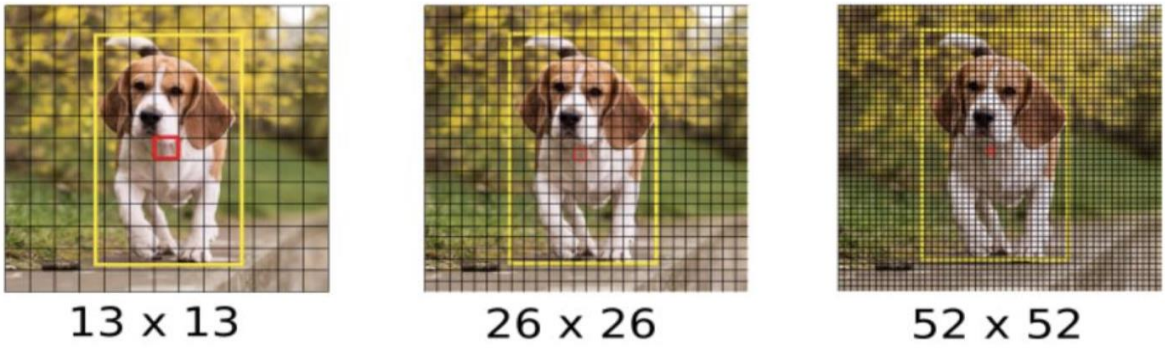
- p_0 là xác suất dự báo vật thể xuất hiện trong bounding box.
- $<t_x,t_y,t_w,t_h>$ giúp xác định bounding box. Trong đó t_x,t_y là toạ độ tâm và t_w,t_h là kích thước rộng, dài của bounding box.
- $<p_1, p_2 ,\ldots,p_c>$ là véc tơ phân phối xác xuất dự báo của các lớp.



Hình 2. *Kiến trúc một output của model YOLO*

4. DỰ BÁO TRÊN NHIỀU FEATURE MAP (BẢN ĐỒ ĐẶC TRƯNG)

YOLO dự báo trên nhiều feature map (bản đồ đặc trưng). Những feature map (bản đồ đặc trưng) ban đầu có kích thước nhỏ giúp dự báo được các object kích thước lớn. Những feature map (bản đồ đặc trưng) sau có kích thước lớn hơn trong khi anchor box được giữ cố định kích thước nên sẽ giúp dự báo được các vật thể kích thước nhỏ.



Hình 3. *Các feature maps của một mạng YOLO với input shape là 416x416, output là 3 feature maps có kích thước lần lượt là 13x13, 26x26 và 52x52* [3].

Trên mỗi một cell của các feature map (bản đồ đặc trưng) chúng ta sẽ áp dụng 3 anchor box để dự đoán vật thể. Như vậy số lượng các anchor box khác nhau trong một mô hình YOLO sẽ là 9 (3 feature map x 3 anchor box).

Đồng thời trên một feature map (bản đồ đặc trưng) hình vuông SxS, mô hình YOLOv3 sinh ra một số lượng anchor boxes là : SxSx3. Như vậy số lượng anchor boxes trên một bức ảnh sẽ là :

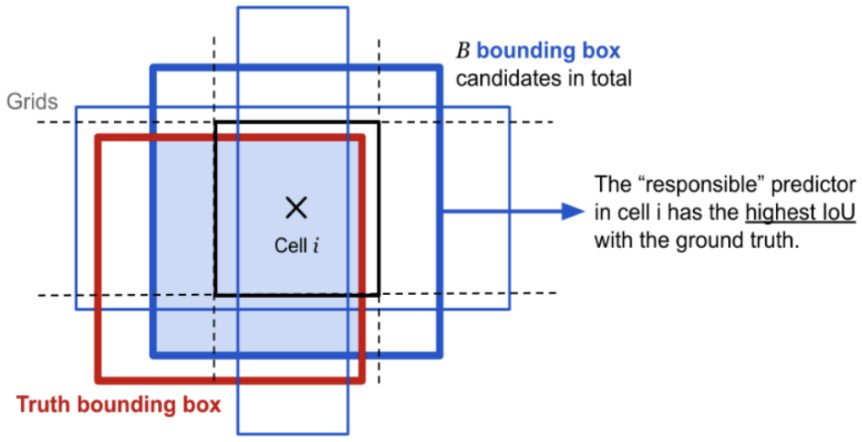
(13x13+26x26+52x52)=10647(anchor boxes)

Đây là một số lượng rất lớn và là nguyên nhân khiến quá trình huấn luyện mô hình YOLO vô cùng chậm bởi cần dự báo đồng thời nhãn và bounding box trên đồng thời 10647 anchor boxes.

5. ANCHOR BOX

Để tìm được bounding box cho vật thể, YOLO sẽ cần các anchor box làm cơ sở ước lượng. Những anchor box này sẽ được xác định trước và sẽ bao quanh vật thể một cách tương đối chính xác. Hiện nay thuật toán regression bounding box đã tinh chỉnh lại anchor box để tạo ra bounding box dự đoán cho vật thể. Trong một mô hình YOLO

mỗi một vật thể trong hình ảnh huấn luyện được phân bố về một anchor box. Trong trường hợp có từ 2 anchor boxes trở lên cùng bao quanh vật thể thì ta sẽ xác định anchor box mà có Intersection over Union (IoU) với ground truth bounding box là cao nhất.



Hình 4. *Xác định anchor box cho một vật thể*

Mỗi vật thể trong hình ảnh huấn luyện được phân bố về một cell trên feature map (bản đồ đặc trưng) mà chứa điểm mid point của vật thể. Chẳng hạn như hình chú chó trong hình 3 sẽ được phân về cho cell màu đỏ vì điểm

mid point của ảnh chú chó rơi vào đúng cell này. Từ cell ta sẽ xác định các anchor boxes bao quanh hình ảnh chú chó.

Như vậy khi xác định một vật thể ta sẽ cần xác định 2 thành phần gắn liền với nó là cell và anchor box. Không chỉ riêng mình cell hoặc chỉ mình anchor box.

6. DỰ BÁO BOUNDING BOX

Để dự báo bounding box cho một vật thể, chúng ta dựa trên một phép biến đổi từ anchor box và cell.

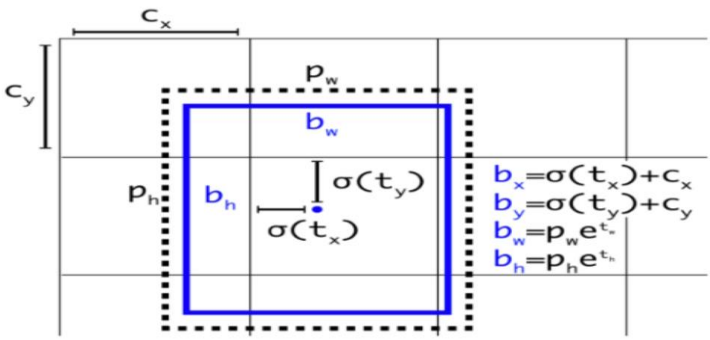
Cho một anchor box có kích thước (p_w,p_h) tại cell nằm trên feature map (bản đồ đặc trưng) với góc trên cùng bên trái của nó là (c_x,c_y), mô hình dự đoán 4 tham số (t_x,t_y,t_w,t_h) trong đó 2 tham số đầu là độ lệch (offset) so với góc trên cùng bên trái của cell và 2 tham số sau là tỷ lệ so với anchor box. Và các tham số này sẽ giúp xác định bounding box dự đoán b có tâm (b_x,b_y), và kích thước (b_w,b_h) thông qua hàm sigmoid và hàm exponential như các công thức bên dưới:

$$\mathbf{B_{x=\sigma(t_x)+C_x}}$$

$$\mathbf{b_{y=\sigma(t_y)+C_y}}$$

$$\mathbf{b_{w=p_w e^{t_w}}}$$

$$\mathbf{b_{h=p_h e^{t_h}}}$$



Hình 5. *Công thức ước lượng bounding box từ anchor box*

Ngoài ra do các toạ độ đã được hiệu chỉnh theo width và height của bức ảnh nên luôn có giá trị nằm trong ngưỡng [0,1]. Do đó khi áp hàm sigmoid giúp giới hạn được toạ độ không vượt quá xa các ngưỡng này.

7. NON-MAX SUPPRESSION (LOẠI BỎ CÁC GIÁ TRỊ KHÔNG CỰC ĐẠI)

Do thuật toán YOLO dự báo ra rất nhiều bounding box trên một bức ảnh nên đối với những cell có vị trí gần nhau, khả năng các khung hình bị overlap (chồng lấn) là rất cao. Trong trường hợp đó, YOLO sẽ cần đến non-max suppression để giảm bớt số lượng các khung hình được sinh ra một cách đáng kể.

Hình 6. *Non-max suppression. Từ 3 bounding box ban đầu cùng bao quanh chiếc xe đã giảm còn một bounding box cuối cùng* [4].

Các bước của non-max suppression:

Bước 1: Đầu tiên chúng ta sẽ tìm cách giảm bớt số lượng các bounding box bằng cách lọc bỏ toàn bộ những bounding box có xác xuất chứa vật thể nhỏ hơn một ngưỡng threshold nào đó, thường là 0.5.

Bước 2: Đối với các bounding box giao nhau, non-max suppression sẽ lựa chọn ra một bounding box có xác xuất chứa vật thể là lớn nhất. Sau đó tính toán chỉ số giao thoa IoU với các bounding box còn lại.

Nếu chỉ số này lớn hơn ngưỡng threshold thì điều đó chứng tỏ 2 bounding boxes đang overlap (chồng lấn) nhau rất cao. Ta sẽ xoá các bounding box có xác xuất thấp hơn và giữ lại bounding box có xác xuất cao nhất. Cuối cùng, ta thu được một bounding box duy nhất cho một vật thể.

B. CÁC PHIÊN BẢN CỦA YOLO

Phiên bản YOLO	**Giới thiệu chung**	**Đặc điểm nổi bậc**
YOLO (You Only Look Once) - 2015	Joseph Redmon và cộng sự phát triển	Phương pháp thống nhất: sử dụng một mạng CNN duy nhất để dự đoán bounding boxes và class probabilities. Xử lý tốc độ cao: 45 FPS.
YOLOv2 (YOLO9000) - 2016	Cải thiện độ chính xác và tốc độ	Chuẩn hóa theo batch. Anchor boxes. Bộ phân loại có độ phân giải cao. YOLO9000: Kết hợp phát hiện đối tượng trên cả COCO và ImageNet.
YOLOv3 - 2018	Tăng cường khả năng phát hiện đối tượng nhỏ	Dự đoán đa kích thước. Darknet-53. Dự đoán bounding box bằng logistic regression.
YOLOv4 - 2020	Cải tiến đáng kể về tốc độ và độ chính xác	CSPDarknet53. Bag of freebies và bag of specials. Sử dụng nhiều kỹ thuật tối ưu hóa và tăng cường dữ liệu.
Scaled-YOLOv4 - 2020	Phiên bản mở rộng của YOLOv4	Kiến trúc đa mô hình (P5, P6, P7). Hợp nhất đường dẫn đặc trưng (PAN). Tăng cường Mosaic, đào tạo tự đối kháng.
YOLOv5 - 2020	Tối ưu hóa cho sử dụng trong thực tế	Triển khai trong PyTorch. Cải thiện tăng cường dữ liệu. Nhiều kích thước mô hình (s, m, l, x).
PP-YOLO - 2020	Được phát triển bởi Baidu	ResNet50-vd. Tối ưu hóa phần cứng. Tăng cường dữ liệu MixUp, CutMix.
PP-YOLOv2 - 2021	Phiên bản cải tiến của PP-YOLO	ResNet50-vd và CSPDarknet. IoU Loss, Grid Sensitive, Matrix NMS. Hiệu suất vượt trội.
YOLOR - 2021	Kết hợp phát hiện đối tượng và phân loại	Học đại diện hợp nhất. Phân loại và phát hiện đối tượng. Hiệu suất cao.
YOLOX - 2021	Mô hình YOLO tiên tiến	Không có neo. Gán nhãn động. Đa quy mô (s, m, l, x).
YOLOv6 - 2022	Tập trung vào ứng dụng công nghiệp	Tăng cường dữ liệu nâng cao. Cải thiện kiến trúc mạng. Tập trung vào hiệu quả triển khai.
YOLOv7 - 2022	Được cho là nhanh nhất và chính xác nhất trong dòng YOLO	E-ELAN. Hàm mất mát nâng cao. Độ chính xác cao và hiệu suất thời gian thực.
PP-YOLOE - 2022	Phiên bản mới của PP-YOLOv2	Kiến trúc mở rộng. Mạng tổng hợp đường dẫn. Hợp nhất tính năng không gian thích ứng. Hiệu suất cao.
YOLOv8 - 2023	Phiên bản nổi bật với khả năng kết hợp giữa tốc độ và độ chính xác	Cải tiến trong thiết kế mạng. Tăng cường dữ liệu hiện đại. Tập trung vào triển khai th›ực tế.
YOLOv9-2024	Giảm suy giảm thông tin	Tối ưu hoá tham số (GELAN)

	trong mạng nơ-ron sâu	Cảit hiện khả năng học tập (PGI)
YOLOv10-2024	Hiệu suất hiện đại với chi phí tính toán giảm đáng kể (độ chính xác vượt trội)	Loại bỏ tối đa NMS. Tối ưu hoá các thành phần mô hình khác nhau.

III. CÁC CẢI TIẾN ĐỀ XUẤT

A. CẢI TIẾN DỰA TRÊN PHIÊN BẢN YOLOV9

1. GIỚI THIỆU VỀ YOLOV9

YOLOv9 [5] là một phiên bản cải tiến, tập trung vào việc giảm suy giảm thông tin trong mạng nơ-ron sâu.

Đặc điểm nổi bật:

GELAN (Generalized Efficient Layer Aggregation Network): giúp tối ưu hóa sử dụng tham số và tăng cường khả năng tổng quát hóa.

PGI (Programmable Gradient Information): cải thiện khả năng học tập bằng cách giảm sự mất mát thông tin qua các tầng.

Tăng tốc độ huấn luyện: giảm độ trễ mà không làm suy giảm độ chính xác.

2. GIỚI THIỆU VỀ PHIÊN BẢN MỚI NHẤT YOLOV10

YOLOv10 [6] là phiên bản mới nhất của dòng mô hình YOLO, được thiết kế để cải thiện tốc độ và độ chính xác bằng cách loại bỏ Non-Maximum Suppression (NMS) và tối ưu hóa toàn bộ pipeline huấn luyện. Kiến trúc này tập trung vào việc đơn giản hóa việc gán nhãn và cải thiện hiệu suất suy luận, giúp giảm độ trễ đáng kể so với các phiên bản trước.

YOLOv10 giới thiệu phương pháp Consistent Dual Assignments (CDA), giúp tăng cường tính nhất quán của nhãn và loại bỏ các bước xử lý hậu kỳ, giúp mô hình đạt được tốc độ nhanh hơn mà không làm giảm độ chính xác.

Thành phần chính:

Backbone – CSPNet cải tiến với Large-Kernel Convolutions

YOLOv10 vẫn giữ lại cấu trúc CSPNet, nhưng được tối ưu bằng large-kernel convolutions (LKC) để mở rộng vùng cảm thụ mà không làm tăng đáng kể số tham số.

Việc sử dụng convolution kernel lớn (9x9, 11x11) giúp cải thiện khả năng học ngữ cảnh không gian mà không làm mất đi chi tiết cục bộ.

Độ sâu của mạng được tối ưu hóa để phù hợp với cấu trúc head mới.

Neck – Rank-Guided Block Design

Neck của YOLOv10 được thiết kế theo phương pháp Rank-Guided Block Design, giúp giảm số lượng phép toán dư thừa.

Tích hợp BiFPN (Bidirectional Feature Pyramid Network) để tăng cường sự hợp nhất của đặc trưng giữa các tầng.

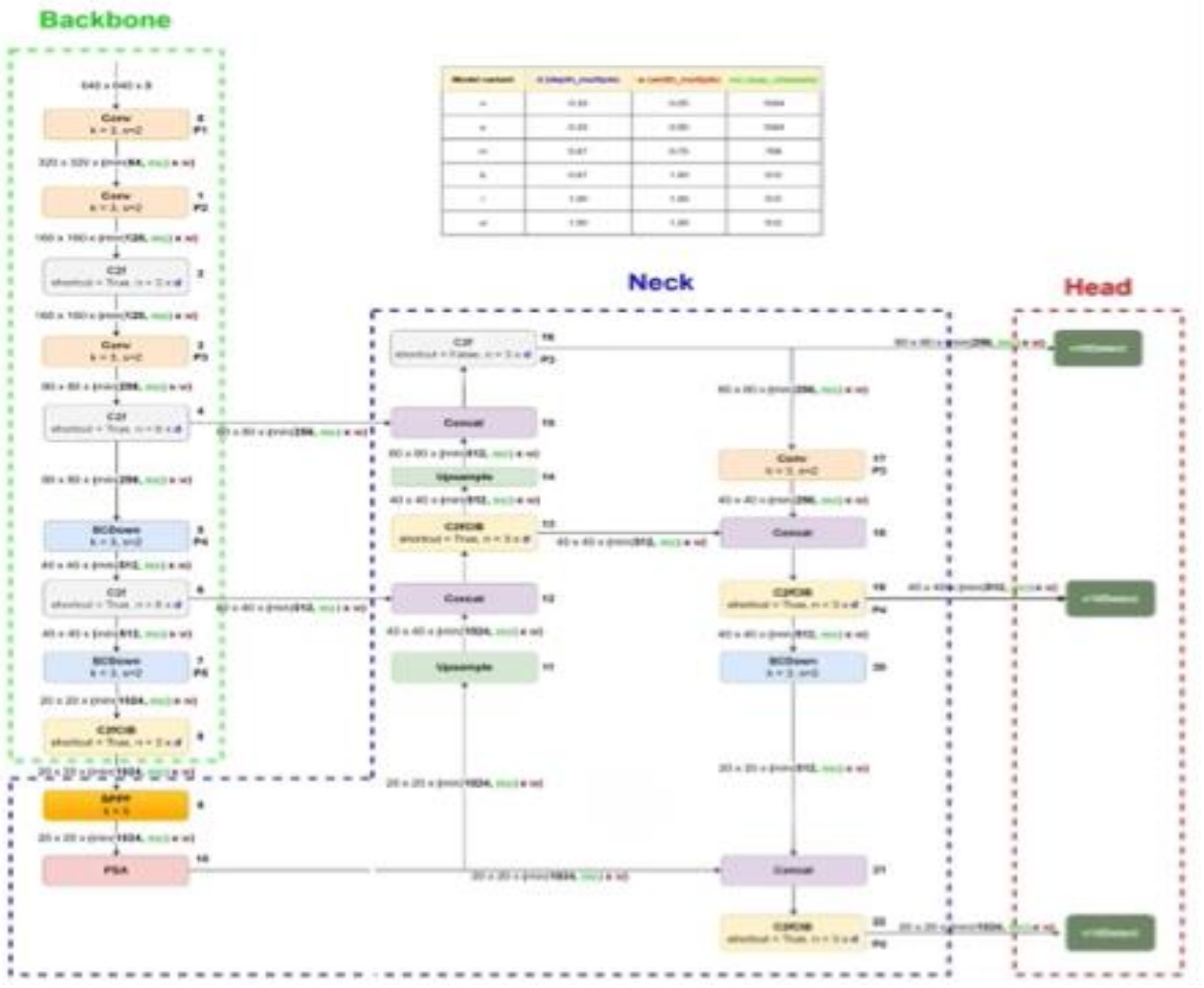
Giúp mô hình duy trì độ chính xác cao hơn trong khi giảm độ trễ.

Head – Consistent Dual Assignments (CDA) thay thế NMS

Loại bỏ hoàn toàn Non-Maximum Suppression (NMS), giảm độ trễ suy luận đáng kể.

Áp dụng Consistent Dual Assignments (CDA), cho phép mỗi vật thể trong ảnh được gán nhãn một cách tối ưu mà không cần hậu xử lý.

Đảm bảo rằng các bounding boxes được dự đoán một cách nhất quán và tối ưu hơn so với phương pháp NMS truyền thống.



Hình 7. *Cấu trúc của VOLOV10*

3. NHỮNG CẢI TIẾN NỔI BẬC CỦA YOLOV10

Cải tiến	**YOLOv10**	**Lợi ích**
Backbone	CSPNet + Large-Kernel Conv	Cải thiện vùng cảm thụ, giảm số tham số
Neck	Rank-Guided Block Design	Giảm phép toán dư thừa, tăng tốc độ inference
Head	Consistent Dual Assignments (CDA)	Loại bỏ NMS, giảm độ trễ suy luận
Gán nhãn	Dual Assignments	Gán nhãn chính xác hơn, tăng tốc huấn luyện
Độ trễ	Giảm 46% so với YOLOv9	Tăng tốc suy luận
Số lượng tham số	Giảm 25%	Giảm yêu cầu phần cứng
Hiệu suất tổng thể	Tốt hơn	Giữ cân bằng giữa tốc độ và độ chính xác

Ưu điểm:

Tốc độ suy luận cao hơn: Loại bỏ NMS giúp giảm đáng kể thời gian xử lý đầu ra.

Độ chính xác cao hơn trên tập COCO: Nhờ kiến trúc head tối ưu và phương pháp gán nhãn cải tiến.

Tăng cường khả năng nhận diện vật thể nhỏ: Với convolution kernel lớn, YOLOv10 có thể giữ lại nhiều thông tin ngữ cảnh hơn.

Hiệu suất phần cứng tốt hơn: Số lượng tham số ít hơn giúp YOLOv10 chạy nhanh hơn trên các thiết bị edge như NVIDIA Jetson hoặc TPU.

Nhược điểm:

Yêu cầu kỹ thuật cao trong huấn luyện: Do loại bỏ NMS, việc tinh chỉnh siêu tham số trở nên quan trọng hơn.

Khả năng tổng quát hóa phụ thuộc vào CDA: Việc áp dụng gán nhãn kép cần kiểm tra kỹ lưỡng để tránh sai sót.

B. PHƯƠNG PHÁP ĐỀ XUẤT-THAY ĐỔI KIẾN TRÚC CỦA YOLOV9 VÀ YOLOV10

Trong phần này, nghiên cứu đề xuất một kiến trúc YOLO được sửa đổi bằng cách tích hợp các thành phần từ YOLOv9 và YOLOv10. Cụ thể:

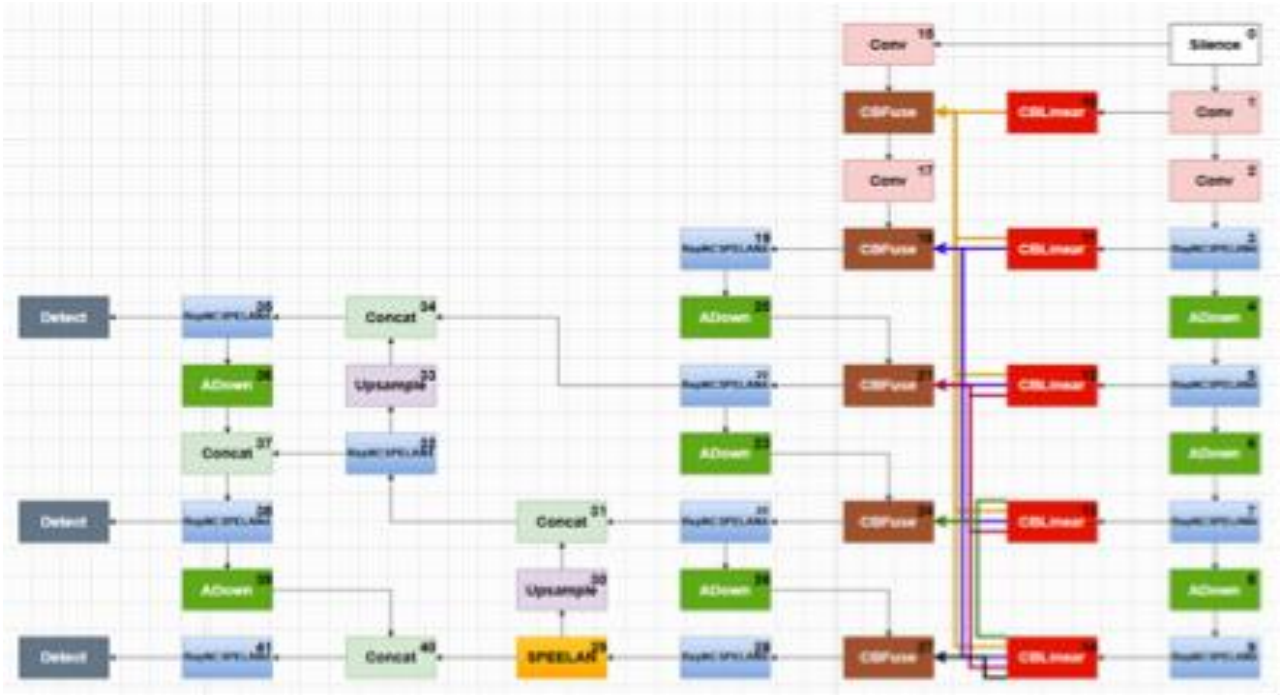
- Sử dụng backbone và neck của YOLOv9.
- Thay thế phần detection head của YOLOv9 bằng detection head của YOLOv10.

Sự kết hợp này giúp tận dụng khả năng trích xuất và tổng hợp đặc trưng mạnh mẽ của YOLOv9, đồng thời kế thừa chiến lược Consistent Dual Assignments (CDA) của YOLOv10, giúp loại bỏ nhu cầu sử dụng Non-Maximum Suppression (NMS), từ đó giảm độ trễ và tăng tốc độ suy luận.

Kiến trúc mô hình được sửa đổi bao gồm các thành phần sau:

Backbone:

Sử dụng Generalized Efficient Layer Aggregation Network (GELAN) từ YOLOv9 (bản yolov9e) để tăng hiệu suất tham số và cải thiện dòng chảy gradient



Hình 8. *Kiến trúc YOLOV9e*

Giữ nguyên cải tiến CSPNet để tối ưu hóa khả năng trích xuất đặc trưng.

Neck:

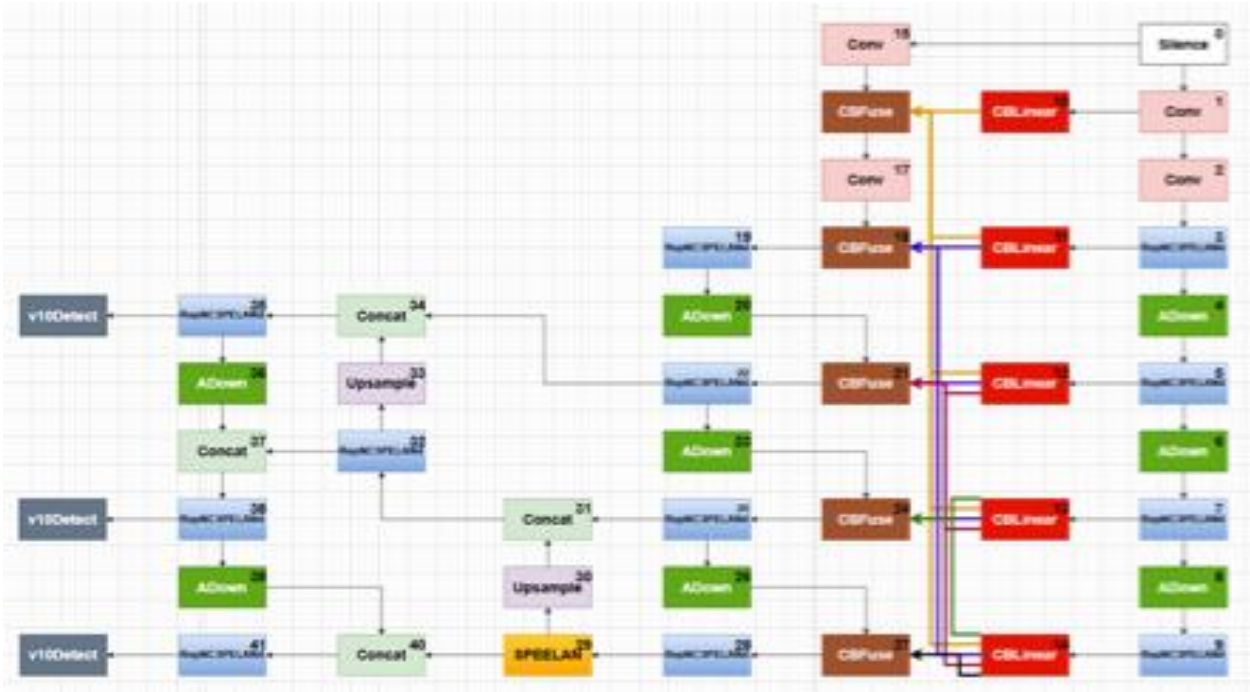
Tích hợp cấu trúc BiFPN từ YOLOv9 để tăng cường hợp nhất đặc trưng trên nhiều cấp độ.

Giảm bớt tính toán dư thừa bằng cách tối ưu hóa chia sẻ thông tin giữa các tầng.

Head:

Thay thế detection head truyền thống của YOLOv9 bằng Consistent Dual Assignments (CDA) head của YOLOv10.

Loại bỏ bước xử lý NMS, giúp giảm độ trễ và tăng tốc độ suy luận đáng kể.



Hình 9. *Kiến trúc sau khi kết hợp YOLOV9 và YOLOV10*

IV. KẾT QUẢ THỰC NGHIỆM

1. THIẾT LẬP THỰC NGHIỆM:

Thực hiện các thực nghiệm trên các tập dữ liệu là Fruit Freshness Detection Dataset với 12410 ảnh thuộc 4 lớp gồm có 'Fresh Apple', 'Fresh Banana', 'Rotten Apple', 'Rotten Banana' được thu thập bởi Brac University [5]. Các bước thực hiện chủ yếu thông qua API của Ultralystics [6] và triển khai trên Google Colab bao gồm:

2. CHUẨN BỊ DỮ LIỆU:

Tạo tập dữ liệu nhỏ hơn từ Fruit Freshness Detection Dataset, đặt tên là data_subset với 10% hình ảnh lấy ngẫu nhiên từ tập huấn luyện, xác thực và kiểm tra của tập dữ liệu gốc.

Dữ liệu của data_subet được chuẩn bị với các đặc trưng sau:

+ Huấn luyện: 70% mẫu

+ Xác thực: 20% mẫu

+ Kiểm tra: 10% mẫu

B. HUẤN LUYỆN MÔ HÌNH NGUYÊN BẢN CỦA YOLOV10

Sử dụng tập dữ liệu data_subset để huấn luyện mô hình nguyên bản của YOLOv10l, đặt tên là fruit_original_subset_v10:

1.
```
#yolov10l
!yolo detect train model=yolov10l.yaml data=datasets/data_subset.yaml workers=2 batch=16 device=0 epochs=100 patience=50 name=fruit_original_subset_v10
```

Hình 10. *Sử dụng API để huấn luyện fruit_original_subset_v10*

Kết quả huấn luyện:

```
100 epochs completed in 1.632 hours.
Optimizer stripped from runs/detect/fruit_original_subset_v10/weights/last.pt, 52.2MB
Optimizer stripped from runs/detect/fruit_original_subset_v10/weights/best.pt, 52.2MB

Validating runs/detect/fruit_original_subset_v10/weights/best.pt...
WARNING ⚠️ validating an untrained model YAML will result in 0 mAP.
Ultralytics 8.3.49 🚀 Python-3.10.12 torch-2.5.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
YOLOv10l summary (fused): 461 layers, 25,722,536 parameters, 0 gradients, 126.3 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 6/6 [00:05<00:00,  1.10it/s]
                   all        186        247      0.836      0.854      0.895       0.75
           Fresh Apple         58         63      0.889      0.937      0.955      0.856
          Fresh Banana         45         90      0.682      0.656      0.719      0.518
          Rotten Apple         35         38      0.971      0.947      0.984      0.912
         Rotten Banana         47         56      0.804      0.877      0.923      0.712
Speed: 0.3ms preprocess, 18.7ms inference, 0.0ms loss, 0.5ms postprocess per image
Results saved to runs/detect/fruit_original_subset_v10
💡 Learn more at https://docs.ultralytics.com/modes/train
```

Hình 11. *Kết quả huấn luyện của fruit_original_subset_v10*

Các chỉ số đo độ chính xác:

```
#yolov10l
!yolo detect val model=runs/detect/fruit_original_subset_v10/weights/best.pt data=datasets/data_subset.yaml iou=0.5 imgsz=640 half=True name=val_fruit_original_subset_v10

Ultralytics 8.3.49 🚀 Python-3.10.12 torch-2.5.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
YOLOv10l summary (fused): 461 layers, 25,722,536 parameters, 0 gradients, 126.3 GFLOPs
val: Scanning /content/gdrive/MyDrive/yolov8_KLTN/datasets/fruit_subset/labels/val.cache... 186 images, 1 backgrounds, 0 corrupt: 100% 186/186 [00:00<?, ?it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 12/12 [00:05<00:00,  2.13it/s]
                   all        186        247      0.836      0.854      0.895       0.75
           Fresh Apple         58         63      0.889      0.937      0.955      0.856
          Fresh Banana         45         90      0.682      0.656      0.719      0.518
          Rotten Apple         35         38      0.971      0.947      0.984      0.912
         Rotten Banana         47         56      0.804      0.877      0.923      0.712
Speed: 2.1ms preprocess, 18.3ms inference, 0.0ms loss, 0.5ms postprocess per image
Results saved to runs/detect/val_fruit_original_subset_v10
💡 Learn more at https://docs.ultralytics.com/modes/val
```

Hình 12. *Các chỉ số đo độ chính xác của fruit_original_subset_v10*

Tốc độ:

```
!yolo detect predict model=runs/detect/fruit_original_subset_v10/weights/best.pt source=inference/fruit_img_test.jpg save=True name=fruit_img_test project=result/fruit_original_subset_v10_2
!yolo detect predict model=runs/detect/fruit_original_subset_v10/weights/best.pt source=inference/fruit_img_test2.jpg save=True name=fruit_img_test2 project=result/fruit_original_subset_v10_2
!yolo detect predict model=runs/detect/fruit_original_subset_v10/weights/best.pt source=inference/fruit_img_test3.jpg save=True name=fruit_img_test3 project=result/fruit_original_subset_v10_2
!yolo detect predict model=runs/detect/fruit_original_subset_v10/weights/best.pt source=inference/fruit_img_test4.jpg save=True name=fruit_img_test4 project=result/fruit_original_subset_v10_2

Ultralytics 8.3.75 🚀 Python-3.11.11 torch-2.5.1+cu124 CUDA:0 (NVIDIA A100-SXM4-40GB, 40507MiB)
YOLOv10l summary (fused): 461 layers, 25,722,536 parameters, 0 gradients, 126.3 GFLOPs

image 1/1 /content/gdrive/MyDrive/yolov8_KLTN/inference/fruit_img_test.jpg: 640x640 1 Fresh Apple, 19.1ms
Speed: 3.9ms preprocess, 19.1ms inference, 22.7ms postprocess per image at shape (1, 3, 640, 640)
Results saved to result/fruit_original_subset_v10_2/fruit_img_test5
💡 Learn more at https://docs.ultralytics.com/modes/predict
Ultralytics 8.3.75 🚀 Python-3.11.11 torch-2.5.1+cu124 CUDA:0 (NVIDIA A100-SXM4-40GB, 40507MiB)
YOLOv10l summary (fused): 461 layers, 25,722,536 parameters, 0 gradients, 126.3 GFLOPs

image 1/1 /content/gdrive/MyDrive/yolov8_KLTN/inference/fruit_img_test2.jpg: 512x640 1 Rotten Apple, 60.2ms
Speed: 3.7ms preprocess, 60.2ms inference, 22.9ms postprocess per image at shape (1, 3, 512, 640)
Results saved to result/fruit_original_subset_v10_2/fruit_img_test22
💡 Learn more at https://docs.ultralytics.com/modes/predict
Ultralytics 8.3.75 🚀 Python-3.11.11 torch-2.5.1+cu124 CUDA:0 (NVIDIA A100-SXM4-40GB, 40507MiB)
YOLOv10l summary (fused): 461 layers, 25,722,536 parameters, 0 gradients, 126.3 GFLOPs

image 1/1 /content/gdrive/MyDrive/yolov8_KLTN/inference/fruit_img_test3.jpg: 448x640 2 Rotten Bananas, 61.4ms
Speed: 3.5ms preprocess, 61.4ms inference, 23.3ms postprocess per image at shape (1, 3, 448, 640)
Results saved to result/fruit_original_subset_v10_2/fruit_img_test32
💡 Learn more at https://docs.ultralytics.com/modes/predict
Ultralytics 8.3.75 🚀 Python-3.11.11 torch-2.5.1+cu124 CUDA:0 (NVIDIA A100-SXM4-40GB, 40507MiB)
YOLOv10l summary (fused): 461 layers, 25,722,536 parameters, 0 gradients, 126.3 GFLOPs

image 1/1 /content/gdrive/MyDrive/yolov8_KLTN/inference/fruit_img_test4.jpg: 448x640 1 Fresh Banana, 63.7ms
Speed: 3.4ms preprocess, 63.7ms inference, 23.8ms postprocess per image at shape (1, 3, 448, 640)
Results saved to result/fruit_original_subset_v10_2/fruit_img_test42
💡 Learn more at https://docs.ultralytics.com/modes/predict
```

Hình 13. *Tốc độ của fruit_original_subset_v10*

C. KẾT HỢP KIẾN TRÚC YOLOV9 VÀ YOLOV10 :

Tạo file YOLOv9e10.yaml:

#yolov9e10

!yolo detect train model=yolov9e10.yaml data=datasets/data_subset.yaml workers=2 batch=16 device=0 epochs=100 patience=50 name=fruit_original_subset_v9e10

```yaml
# Parameters
nc: 80 # number of classes

# GELAN backbone
backbone:
  - [-1, 1, nn.Identity, []]
  - [-1, 1, Conv, [64, 3, 2]] # 1-P1/2
  - [-1, 1, Conv, [128, 3, 2]] # 2-P2/4
  - [-1, 1, C2fCIB, [256, 128, 64, 2]] # 3
  - [-1, 1, SCDown, [256, 3, 2]] # 4-P3/8
  - [-1, 1, C2fCIB, [512, 256, 128, 2]] # 5
  - [-1, 1, SCDown, [512, 3, 2]] # 6-P4/16
  - [-1, 1, C2fCIB, [1024, 512, 256, 2]] # 7
  - [-1, 1, SCDown, [1024, 3, 2]] # 8-P5/32
  - [-1, 1, C2fCIB, [1024, 512, 256, 2]] # 9

  - [1, 1, CBLinear, [[64]]] # 10
  - [3, 1, CBLinear, [[64, 128]]] # 11
  - [5, 1, CBLinear, [[64, 128, 256]]] # 12
  - [7, 1, CBLinear, [[64, 128, 256, 512]]] # 13
  - [9, 1, CBLinear, [[64, 128, 256, 512, 1024]]] # 14

  - [0, 1, Conv, [64, 3, 2]] # 15-P1/2
  - [[10, 11, 12, 13, 14, -1], 1, CBFuse, [[0, 0, 0, 0, 0]]] # 16
  - [-1, 1, Conv, [128, 3, 2]] # 17-P2/4
  - [[11, 12, 13, 14, -1], 1, CBFuse, [[1, 1, 1, 1]]] # 18
  - [-1, 1, C2fCIB, [256, 128, 64, 2]] # 19
  - [-1, 1, SCDown, [256, 3, 2]] # 20-P3/8
  - [[12, 13, 14, -1], 1, CBFuse, [[2, 2, 2]]] # 21
  - [-1, 1, C2fCIB, [512, 256, 128, 2]] # 22
  - [-1, 1, SCDown, [512, 3, 2]] # 23-P4/16
  - [[13, 14, -1], 1, CBFuse, [[3, 3]]] # 24
  - [-1, 1, C2fCIB, [1024, 512, 256, 2]] # 25
  - [-1, 1, SCDown, [1024, 3, 2]] # 26-P5/32
  - [[14, -1], 1, CBFuse, [[4]]] # 27
  - [-1, 1, C2fCIB, [1024, 512, 256, 2]] # 28
  - [-1, 1, SPPELAN, [512, 256]] # 29

# GELAN head
head:
  - [-1, 1, nn.Upsample, [None, 2, "nearest"]]
  - [[-1, 25], 1, Concat, [1]] # cat backbone P4
  - [-1, 1, C2fCIB, [512, 512, 256, 2]] # 32

  - [-1, 1, nn.Upsample, [None, 2, "nearest"]]
  - [[-1, 22], 1, Concat, [1]] # cat backbone P3
  - [-1, 1, C2fCIB, [256, 256, 128, 2]] # 35 (P3/8-small)
  - [-1, 1, SCDown, [256, 3, 2]]
  - [[-1, 32], 1, Concat, [1]] # cat head P4
  - [-1, 1, C2fCIB, [512, 512, 256, 2]] # 38 (P4/16-medium)

  - [-1, 1, SCDown, [512, 3, 2]]
  - [[-1, 29], 1, Concat, [1]] # cat head P5
  - [-1, 1, C2fCIB, [512, 1024, 512, 2]] # 41 (P5/32-large)

  - [[35, 38, 41], 1, v10Detect, [nc]] # Detect(P3, P4, P5)
```

Hình 14. *Nội dung file YOLOv9e10*

Kết quả huấn luyện:

```
100 epochs completed in 0.871 hours.
Optimizer stripped from runs/detect/fruit_original_subset_v9e10/weights/last.pt, 61.7MB
Optimizer stripped from runs/detect/fruit_original_subset_v9e10/weights/best.pt, 61.7MB

Validating runs/detect/fruit_original_subset_v9e10/weights/best.pt...
Ultralytics 8.3.75 🚀 Python-3.11.11 torch-2.5.1+cu124 CUDA:0 (NVIDIA A100-SXM4-40GB, 40507MiB)
YOLOv9e10 summary (fused): 434 layers, 30,352,744 parameters, 0 gradients, 138.3 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 6/6 [00:02<00:00,  2.75it/s]
                   all        186        247      0.909      0.832      0.907        0.8
           Fresh Apple         58         63      0.963      0.921      0.958      0.867
          Fresh Banana         45         90       0.79      0.628       0.74       0.59
          Rotten Apple         35         38          1      0.957      0.993      0.945
         Rotten Banana         47         56      0.882      0.821      0.936      0.798
Speed: 0.1ms preprocess, 7.6ms inference, 0.0ms loss, 0.1ms postprocess per image
Results saved to runs/detect/fruit_original_subset_v9e10
💡 Learn more at https://docs.ultralytics.com/modes/train
```

Hình 15. *Kết quả huấn luyện sau khi kết hợp kiến trúc 2 phiên bản YOLOV9 và YOLOV10*

Các chỉ số đo độ chính xác:

```
#yolov9e10
!yolo detect val model=runs/detect/fruit_original_subset_v9e10/weights/best.pt data=datasets/data_subset.yaml iou=0.5 imgsz=640 half=True name=val_fruit_original_subset_v9e10

Ultralytics 8.3.75 🚀 Python-3.11.11 torch-2.5.1+cu124 CUDA:0 (NVIDIA A100-SXM4-40GB, 40507MiB)
YOLOv9e10 summary (fused): 434 layers, 30,352,744 parameters, 0 gradients, 138.3 GFLOPs
val: Scanning /content/gdrive/MyDrive/yolov8_KLTN/datasets/fruit_subset/labels/val.cache... 186 images, 1 backgrounds, 0 corrupt: 100% 186/186 [00:00<?, ?it/s]
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 12/12 [00:02<00:00,  4.44it/s]
                   all        186        247      0.909      0.832      0.907        0.8
           Fresh Apple         58         63      0.963      0.921      0.958      0.867
          Fresh Banana         45         90       0.79      0.628      0.741       0.59
          Rotten Apple         35         38          1      0.957      0.993      0.945
         Rotten Banana         47         56      0.882      0.821      0.936      0.798
Speed: 0.7ms preprocess, 7.9ms inference, 0.0ms loss, 0.1ms postprocess per image
Results saved to runs/detect/val_fruit_original_subset_v9e10
💡 Learn more at https://docs.ultralytics.com/modes/val
```

Hình 16. *Kết quả và độ hính xác sau khi kết hợp kiến trúc 2 phiên bản YOLOV9 và YOLOV10*

Tốc độ:

```
#val_fruit_original_subset_v9e10
!yolo detect predict model=runs/detect/fruit_original_subset_v9e10/weights/best.pt source=inference/fruit_img_test.jpg save=True name=fruit_img_test project=result/fruit_original_subset_v9e10
!yolo detect predict model=runs/detect/fruit_original_subset_v9e10/weights/best.pt source=inference/fruit_img_test2.jpg save=True name=fruit_img_test2 project=result/fruit_original_subset_v9e10
!yolo detect predict model=runs/detect/fruit_original_subset_v9e10/weights/best.pt source=inference/fruit_img_test3.jpg save=True name=fruit_img_test3 project=result/fruit_original_subset_v9e10
!yolo detect predict model=runs/detect/fruit_original_subset_v9e10/weights/best.pt source=inference/fruit_img_test4.jpg save=True name=fruit_img_test4 project=result/fruit_original_subset_v9e10

Ultralytics 8.3.75 🚀 Python-3.11.11 torch-2.5.1+cu124 CUDA:0 (NVIDIA A100-SXM4-40GB, 40507MiB)
YOLOv9e10 summary (fused): 434 layers, 30,352,744 parameters, 0 gradients, 138.3 GFLOPs

image 1/1 /content/gdrive/MyDrive/yolov8_KLTN/inference/fruit_img_test.jpg: 640x640 1 Fresh Apple, 15.8ms
Speed: 3.8ms preprocess, 15.8ms inference, 22.3ms postprocess per image at shape (1, 3, 640, 640)
Results saved to result/fruit_original_subset_v9e10/fruit_img_test
💡 Learn more at https://docs.ultralytics.com/modes/predict
Ultralytics 8.3.75 🚀 Python-3.11.11 torch-2.5.1+cu124 CUDA:0 (NVIDIA A100-SXM4-40GB, 40507MiB)
YOLOv9e10 summary (fused): 434 layers, 30,352,744 parameters, 0 gradients, 138.3 GFLOPs

image 1/1 /content/gdrive/MyDrive/yolov8_KLTN/inference/fruit_img_test2.jpg: 512x640 1 Rotten Apple, 1 Rotten Banana, 75.7ms
Speed: 3.7ms preprocess, 75.7ms inference, 23.7ms postprocess per image at shape (1, 3, 512, 640)
Results saved to result/fruit_original_subset_v9e10/fruit_img_test2
💡 Learn more at https://docs.ultralytics.com/modes/predict
Ultralytics 8.3.75 🚀 Python-3.11.11 torch-2.5.1+cu124 CUDA:0 (NVIDIA A100-SXM4-40GB, 40507MiB)
YOLOv9e10 summary (fused): 434 layers, 30,352,744 parameters, 0 gradients, 138.3 GFLOPs

image 1/1 /content/gdrive/MyDrive/yolov8_KLTN/inference/fruit_img_test3.jpg: 448x640 2 Rotten Bananas, 75.6ms
Speed: 3.2ms preprocess, 75.6ms inference, 24.0ms postprocess per image at shape (1, 3, 448, 640)
Results saved to result/fruit_original_subset_v9e10/fruit_img_test3
💡 Learn more at https://docs.ultralytics.com/modes/predict
Ultralytics 8.3.75 🚀 Python-3.11.11 torch-2.5.1+cu124 CUDA:0 (NVIDIA A100-SXM4-40GB, 40507MiB)
YOLOv9e10 summary (fused): 434 layers, 30,352,744 parameters, 0 gradients, 138.3 GFLOPs

image 1/1 /content/gdrive/MyDrive/yolov8_KLTN/inference/fruit_img_test4.jpg: 448x640 1 Fresh Banana, 77.5ms
Speed: 3.7ms preprocess, 77.5ms inference, 24.0ms postprocess per image at shape (1, 3, 448, 640)
Results saved to result/fruit_original_subset_v9e10/fruit_img_test4
💡 Learn more at https://docs.ultralytics.com/modes/predict
```

Hình 17. *Kết quả về tốc độ sau khi kết hợp kiến trúc 2 phiên bản YOLOV9 và YOLOV10*

D. KẾT QỦA CỦA SỰ KẾT HỢP HAI KIẾN TRÚC

Kết quả các phiên bản của mô hình đào tạo từ tập dữ liệu data_subset:

Bảng so sánh các chỉ số đo độ chính xác:

Model	Precision	Recall	mAP50	mAP50-95
fruit_original_subset_v10	0.836	0.854	0.895	0.75
fruit_original_subset_v9e10	0.909 (+0.063)	0.832 (-0.022)	0.907 (+0.012)	0.8 (+0.05)

Tốc độ (trung bình từ 4 lần detect ảnh):

Model	Preprocess + Inference + Postprocess
fm_ original_subset_v10	77.9ms
fm_ original_subset_v9e10	88.25ms (-10.35ms)

V. KẾT LUẬN

Từ kết quả thực nghiệm, chúng ta có thể kết luận rằng thay đổi kiến trúc YOLOv10 cho sự cải thiện về độ chính xác (chỉ có Recall là bị giảm) nhưng bù lại tốc độ cũng chậm đi (-10.35ms).

Huấn luyện mô hình mới tiêu tốn một lượng tài nguyên lớn hơn rất nhiều so với mô hình nguyên bản (13.8G < 25G).

Tóm lại, mặc dù sự kết hợp của hai kiến trúc đã chứng minh hiệu quả về độ chính xác, nhưng việc tiêu tốn nhiều tài nguyên vẫn là một hạn chế cần được khắc phục.

Việc tìm ra giải pháp cân bằng giữa hiệu suất và hiệu quả tài nguyên sẽ là chìa khóa cho sự phát triển trong tương lai của lĩnh vực đang nghiên cứu.

VI. LỜI CẢM ƠN

Nghiên cứu được tài trợ bởi Trường Đại học Ngoại ngữ -Tin học Thành phố Hồ Chí Minh trong khuôn khổ Đề tài mã số H2024-11.

VII. TÀI LIỆU THAM KHẢO

[1] Wang Zhiquiang, Liu Jun, 2017, "A review of object detection based on convolutional neural network," trong *Chinese Control Conference (CCC)*, Dalian, China.

[2] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, 2016, "You Only Look Once: Unified, Real-Time Object Detection," trong *Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA.

[3] Daniel Tan Wei Xun, Yoke Lin Lim, Sutthiphong Srigrarom, 2021, "Drone detection using YOLOv3 with transfer learning on NVIDIA Jetson TX2," trong *Second International Symposium on Instrumentation, Control, Artificial Intelligence, and Robotics (ICA-SYMP)*, Bangkok, Thailand.

[4] CH. Ranga Reddy, Angshuman Roy, Ms. G. Priyanka , Dr. G. Victo Sudha George, 2022, "VEHICLE DETECTION USING YOLO V3 FOR COUNTING THE VEHICLES AND TRAFFIC ANALYSIS," *nternational Research Journal of Engineering and Technology (IRJET),* tập 9, số 03, pp. 411-412.

[5] Linhan Song, Yuhang Wu, Wenbo Zhang, 2024, "Research on Fire Detection Based on the Yolov9 Algorithm," Changchun, China.

[6] Ammar Ahmed, Abdu Manaf, 2024, "Pediatric Wrist Fracture Detection in X-rays via YOLOv10 Algorithm and Dual Label Assignment System," *ResearchGate.*

IMPROVING YOLO IMAGE RECOGNITION MODEL BASED ON CHANGING THE MODEL'S ARCHITECTURAL ARCHITECTURE

Phan Thi Ngoc Han

ABSTRACT— Object detection is one of the most prominent topics in deep learning due to its high applicability, ease of data preparation, and a wide range of practical applications.

Object detection is one of the most prominent topics in deep learning due to its high applicability, ease of data preparation, and a wide range of practical applications.

This research aims to improve the accuracy and speed of the YOLO application by modifying the model's architecture of the latest version.

Keywords—YOLO, Achitecture, Improving, Accuracy, Speed

Phan Thị Ngọc Hân – Giảng viên khoa Công nghệ thông tin – HUFLIT.

Lĩnh vực nghiên cứu: Nhận diện hình ảnh, Trí tuệ nhân tạo