**ĐẠI HỌC DUY TÂN**

**KHOA CÔNG NGHỆ THÔNG TIN**

--------------🙖🙐✰🙖🙐-------------



**MÔN: ĐỒ ÁN MÔN – DS 423**

**TÊN ĐỀ TÀI :**

**DỰ ÁN PHÁT HIỆN CHÁY BẰNG YOLO**

|  |  |  |
| --- | --- | --- |
| **Giảng viên hướng dẫn** | **:** | **Lê Quý Vang** |
| **Nhóm** | **:** | **4** |
| **Thành Viên Nhóm** | **:** | **Phạm Đức Bình** |
|  |  | **Nguyễn Văn Chương** |
|  |  | **Nguyễn Doãn Thành Long** |
|  |  | **Trần Quốc Luận** |
|  |  | **Nguyễn Thị Cẩm Ly** |
|  |  | **Ngô Bùi Nhật Sang** |
|  |  | **Nguyễn Mai Thuần** |
|  |  | **Trần Văn Quốc Vương** |
|  |  | **Đỗ Thị Yến Vy** |

*Đà Nẵng, tháng 12 năm 2022*

Tt (19)

# **PHẦN I MỞ ĐẦU**

1. **LÝ DO CHỌN ĐỀ TÀI**

Những năm gần đây ở nước ta, cùng với sự phát triển kinh tế - xã hội, dẫn đến tốc độ đô thị hóa ngày càng nhanh. Cũng vì thế mà nhiều khu đô thị mới, khu dân cư hay thậm chí là nhiều nhà máy, xí nghiệp, cơ sở sản xuất, kinh doanh nhanh chóng được hình thành và phát triển. Nguy cơ về cháy, nổ xảy ra cao hơn bao giờ hết. Mặc dù đã có nhiều biện pháp thông tin, tuyên truyền và sự nỗ lực của các cấp, các ngành để chỉ đạo thực hiện công tác an toàn vệ sinh lao động, phòng chống cháy nổ nhưng vẫn còn một số đơn vị, cá nhân còn chủ quan chưa thực sự chú trọng công tác phòng cháy, chữa cháy (PCCC) và cứu nạn, cứu hộ, dẫn đến xảy ra các vụ cháy, nổ không được kiểm soát kịp thời gây hậu quả nghiêm trọng. Tai nạn về cháy nổ nghiêm trọng nhất xảy ra trong thời gian gần đây là vụ cháy chung cư cao cấp Carina Plaza, Quận 8, Sài Gòn, khiến 13 người chết, 91 người bị thương.

Cháy nổ thuộc loại tai nạn hết sức nghiêm trọng vì cùng lúc có thể gây thiệt hại lớn về tính mạng con người lẫn tài sản. Có nhiều nguyên nhân dẫn đến sự cố hỏa hoạn như chập điện, nổ các loại bình chứa khí, tai nạn giao thông,... Mà nguyên nhân chính là thiếu ý thức và kiến thức trong việc phòng cháy chữa cháy của toàn dân. Việc đảm bảo các trang thiết bị, điều kiện nhân lực, vật lực về an toàn phòng cháy, chữa cháy và cứu nạn, cứu hộ tại các công sở, cơ quan, đơn vị, khu dân cư một số nơi chỉ làm chiếu lệ, qua loa, mang tính đối phó. Cùng với xu thể thế giới, sự phát triển của cơ sở hạ tầng, kiến trúc thượng tần và các thiết bị khoa học kỹ thuật, việc phát triển và đầu tư vào công tác PCCC là cần thiết nhằm đảm bảo an toàn cho thiết bị, công trình, cũng như con người, tránh nhiều trường hợp đáng tiếc xảy ra cho cộng đồng và xã hội. Ở các đô thị lớn, đông dân cư thì cháy nổ càng dễ trở thành thảm họa, việc phòng cháy vì thế cần phải trở thành một nguyên tắc sống, ăn sâu bám rễ vào ý thức của từng người dân. Mỗi người hãy tạo cho mình lượng kiến thức nhất định về công tác phòng cháy và chữa cháy.

Khi phát hiện bằng hình ảnh đám cháy đã trở thành một điểm nóng trong nghiên cứu, phát hiện bằng hình ảnh có ưu điểm là phát hiện sớm, độ chính xác cao, cài đặt hệ thống linh hoạt và phát hiện hiệu quả các đám cháy trong không gian rộng lớn và cấu trúc tòa nhà phức tạp. Dựa trên học sâu và thuật toán được sử dụng để phát hiện tình huống cháy và đã đạt được kết quả chính xác hơn so với các phương pháp cổ điển trong thử nghiệm thực tế. Khả năng phát hiện bằng camera có thể đạt được khả năng phát hiện 7 × 24 giờ cả ngày, có thể ngăn chặn cháy rừng tốt hơn. Bằng cách phân tích thông tin hình ảnh thu được từ màn hình, đám cháy có thể được tìm thấy ở giai đoạn đầu, để tránh xảy ra nguy hiểm cháy.

1. **MỤC ĐÍCH VÀ Ý NGHĨA CỦA ĐỀ TÀI**
   1. **Mục đích của đề tài**

Mục tiêu chính của đề tài này là sử dụng kỹ thuật YOLO để xây dựng ứng dụng dự báo cháy bằng YOLO và xử dụng camera để phát hiện những vùng cháy.

* 1. **Ý nghĩa của đề tài**

Ý nghĩa chính của đề tài này là giúp cho công tác phòng cháy chữa cháy được hiệu quả hơn và giúp ngăn chặn, hạn chế đến mức thấp nhất những rủi ro đáng tiếc tránh những trường hợp xấu xảy ra làm thiệt hại người và tài sản của cá nhân, cộng đồng…

1. **ĐỐI TƯỢNG NGHIÊN CỨU**

- Các hành động có trong hình ảnh và video.

- Các phương pháp, giải thuật về nhận dạng.

- Kỹ thuật về YOLO.

1. **PHẠM VI NGHIÊN CỨU**

Nghiên cứu kỹ thuật xử lý ảnh và video

Nghiên cứu các thuật toán object detection

1. **PHƯƠNG PHÁP NGHIÊN CỨU**

- Các tài liệu về cơ sở lý thuyết: Xử lý ảnh số, lọc trích ảnh số, nhận diện cháy bằng camera.

- Phương pháp computer vision

- Các tài liệu liên quan tới lập trình

- Xây dựng chương trình thử nghiệm.

- Kiểm thử tính hiệu quả của chương trình với các nội dung khác nhau.

## **ĐỐI TƯỢNG SỬ DỤNG**

Người dùng: Sử dụng ứng dụng để phát hiện cháy nhằm phát hiện những rủi ro về tính mạng và con người.

# **PHẦN II CƠ SỞ LÝ THUYẾT**

Trong vài năm trở lại đây, Object detection là một trong những đề tài rất hót của deep learning bởi khả năng ứng dụng cao, dữ liệu dễ chuẩn bị và kết quả ứng dụng thì cực kì nhiều. Các thuật toán mới của Object detection như YOLO, SSD có tốc độ khá nhanh và độ chính xác cao nên giúp cho Object Detection cụ 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. Các mô hình cũng trở nên nhẹ hơn nên có thể hoạt động trên các thiết bị để tạo nên các thiết bị thông minh.

## **1. TỔNG QUAN VỀ YOLO**

YOLO (You only look once) là một mô hình mạng CNN cho việc phát hiện, nhận dạng, phân loại đối tượng. YOLO được tạo ra từ việc kết hợp giữa các convolutional layers và connected layers. Trong đó các convolutional layers sẽ trích xuất ra các feature của ảnh, còn full-connected layers sẽ dự đoán ra xác suất đã và tọa độ của đối tượng.

YOLO có thể không phải là thuật toán tốt nhất nhưng nó là thuật toán nhanh nhất trong các lớp mô hình object detection. Nó có thể đạt được tốc độ gần như real time mà độ chính xác không quá giảm so với các model thuộc top đầu.

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

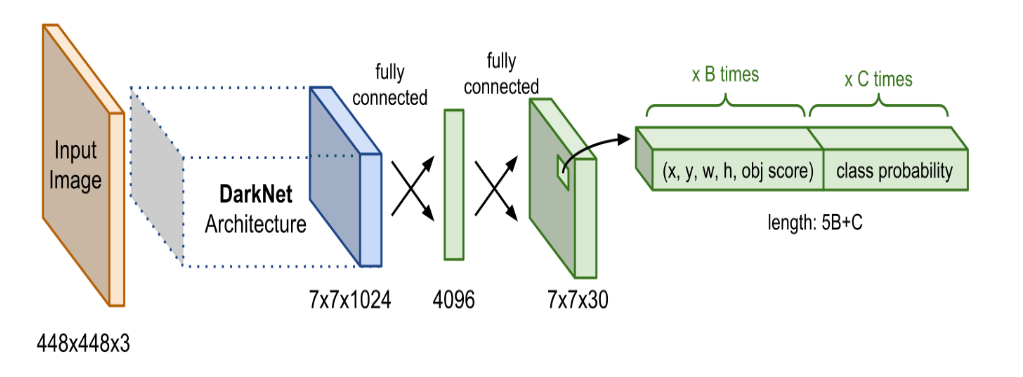
Một trong những ưu điểm mà YOLO đem lại đã là chỉ sử dụng thông tin toàn bộ bức ảnh một lần và dự đoán toàn bộ object box chứa các đối tượng, mô hình được xây dựng theo kiểu end-to-end nên được huấn luyện hoàn toàn bằng gradient descent.

## **2. KIẾN TRÚC MẠNG YOLO**

Kiến trúc YOLO bao gồm: base network là các mạng convolution làm nhiệm vụ trích xuất đặc trưng. Phần phía sau là những Extra Layers được áp dụng để phát hiện vật thể trên feature map của base network.

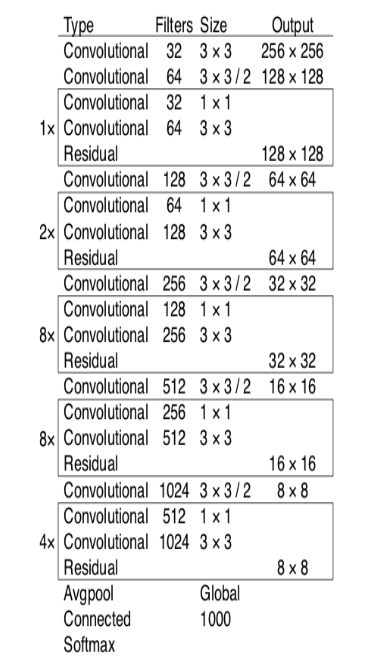
Thành phần Darknet Architechture được gọi là base network có tác dụng trích suất đặc trưng. Output của base network là một feature map có kích thước 7x7x1024 sẽ được sử dụng làm input cho các Extra layers có tác dụng dự đoán nhãn và tọa độ bounding box của vật thể.

Base network của YOLO sử dụng chủ yếu là các convolutional layer và các fully conntected layer. Các kiến trúc YOLO cũng khá đa dạng và có thể tùy biến thành các version cho nhiều input shape khác nhau.



Hình 2.1.Kiến trúc mạng YOLO

Trong YOLO version 3 tác giả áp dụng một mạng feature extractor là darknet-53. Mạng này gồm 53 convolutional layers kết nối liên tiếp, mỗi layer được theo sau bởi một batch normalization và một activation Leaky Relu. Để giảm kích thước của output sau mỗi convolution layer, tác giả down sample bằng các filter với kích thước là 2. Mẹo này có tác dụng giảm thiểu số lượng tham số cho mô hình.



Hình 2.2.Những lớp của convolutional

Các bức ảnh khi được đưa vào mô hình sẽ được scale để về chung một kích thước phù hợp với input shape của mô hình và sau đó được gom lại thành batch đưa vào huấn luyện.

Hiện tại YOLO đang hỗ trợ 2 đầu vào chính là 416x416 và 608x608. Mỗi một đầu vào sẽ có một thiết kế các layers riêng phù hợp với shape của input. Sau khi đi qua các layer convolutional thì shape giảm dần theo cấp số nhân là 2. Cuối cùng ta thu được một feature map có kích thước tương đối nhỏ để dự báo vật thể trên từng ô của feature map.

Kích thước của feature map sẽ phụ thuộc vào đầu vào. Đối với input 416x416 thì feature map có các kích thước là 13x13, 26x26 và 52x52. Và khi input là 608x608 sẽ tạo ra feature map 19x19, 38x38, 72x72.

## **3. NGUYÊN LÝ HOẠT ĐỘNG CỦA MẠNG YOLO**

Đầu vào của mô hình là một ảnh, mô hình sẽ nhận dạng ảnh đó có đối tượng nào hay không, sau đó sẽ xác định tọa độ của đối tượng trong bức ảnh. Ảnh đầu vào được chia thành thành S×S ô thường thì sẽ là 3×3, 7×7, 9×9... việc chia ô này có ảnh hưởng tới việc mô hình phát hiện đối tượng.

|  |
| --- |
|  |

Hình 2.3. Ví dụ minh họa

Ví dụ: Với Input là 1 ảnh, đầu ra mô hình là một ma trận 3 chiều có kích thước  S×S×(5×N+M) với số lượng tham số mỗi ô là (5×N+M) với N và M lần lượt là số lượng Box và Class mà mỗi ô cần dự đoán. Ví dụ với hình ảnh trên chia thành 7×7 ô, mỗi ô cần dự đoán 2 bounding box và 3 object : con chó, ô tô, xe đạp thì output là 7 ×7×13, mỗi ô sẽ có 13 tham số, kết quả trả về (7×7×2=98) bounding box. Chúng ta sẽ cùng giải thích con số (5×N+M) được tính như thế nào.

Dự đoán mỗi bounding box gồm 5 thành phần : (x, y, w, h, prediction) với (x, y ) là tọa độ tâm của bounding box, (w, h) lần lượt là chiều rộng và chiều cao của bounding box, prediction được định nghĩa Pr(Object)∗ IOU(pred,truth). Với hình ảnh trên như ta tính mỗi ô sẽ có 13 tham số, ta có thể hiểu đơn giản như sau tham số thứ 1 sẽ chỉ ra ô đó có chứa đối tượng nào hay không P(Object), tham số 2, 3, 4, 5 sẽ trả về x, y ,w, h của Box1. Tham số 6, 7, 8, 9, 10 tương tự sẽ Box2, tham số 11, 12, 13 lần lượt là xác suất ô đó có chứa object1( P(chó|object), object2(P(ô tô|object)), object3(P( xe đạp|object)). Lưu ý rằng tâm của bounding box nằm ở ô nào thì ô đó sẽ chứa đối tượng, cho dù đối tượng có thể ở các ô khác thì cũng sẽ trả về là 0. Vì vậy việc mà 1 ô chứa 2 hay nhiều tâm của bouding box hay đối tượng thì sẽ không thể detect được, đó là một hạn chế của mô hình YOLO, vậy ta cần phải tăng số lượng ô chia trong 1 ảnh lên đó là lí do vì sao mình nói việc chia ô có thể làm ảnh hưởng tới việc mô hình phát hiện đối tượng.

## **4. OUTPUT CỦA YOLO**

### **4.1. Output**

Output của mô hình YOLO là một véc tơ sẽ bao gồm các thành phần được mô tả qua hình dưới đây.

|  |
| --- |
|  |

Trong đó:

: Là xác suất dự báo vật thể xuất hiện trong bounding box.

<Giúp xác định bounding box. Trong đó , là tọa độ tâm còn là độ rộng và chiều cao của bounding box.

<là véc tơ phân phối xác suất dự báo của n classes.

Việc hiểu output khá là quan trọng để chúng ta cấu hình tham số chuẩn xác khi huấn luyện model qua các open source như darknet. Như vậy output sẽ được xác định theo số lượng classes theo công thức (n\_class+5). Nếu huấn luyện 80 classes thì bạn sẽ có output là 85. Trường hợp bạn áp dụng 3 anchors/cell thì số lượng tham số output sẽ là:

(n class+5) x 3 = 85 x 3 = 255

|  |
| --- |
|  |

Hình 2.4.Xác định đầu ra của bounding box

Kiến trúc một output của một model YOLO. Hình ảnh gốc là một feature map kích thước 13 x 13. Trên mỗi một cell của feature map chúng ta lựa chọn ra 3 anchor boxes với kích thước khác nhau lần lượt là box 1, box 2, box 3 sao cho tâm của các anchor boxes trùng với cell. Khi đó output của YOLO là một véc tơ concatenate ( ghép lại ) của 3 bounding boxes. Với các attributes ( thuộc tính ) tọa độ tâm, kích thước của một bounding box và các véc tơ phân phối xác suất dự báo của các classes.

### **4.2. Dự báo trên nhiều feature map**

YOLO dự báo trên nhiều feature map. Những feature map ban đầu có kích thước nhỏ giúp dự báo được các object kích thước lớn. Những feature map sau có kích thước lớn hơn trong khi anchor box được giữ cố định kích thước nên sẽ giúp dự báo các vật thể kích thước nhỏ. Hình 2.5 sẽ thể hiện rõ các feature máy của mạng YOLO.

|  |
| --- |
|  |

Hình 2.5. Tham số của feature maps

Với input shape là 416x416, output là 3 feature maps có kích thước lần lượt là 13x13, 26x26 và 52x52. Trên mỗi một cell của các feature map 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 featue map x 3 anchor box).

Đồng thời trên một feature map hình vuông S x S, mô hình YOLO sinh ra một số lượng anchor box là: S x S x 3. Như vậy số lượng anchor boxes trên một bức ảnh sẽ là: (13x13+26x26+52x52)x3=10647 (anchor boxes).

#### **4.3. Anchor box**

Anchor box là một bounding box cơ sở để xác định bounding box bao quanh vật thể dựa trên các phép dịch tâm và độ phóng đại kích thước chiều dài, rộng.

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. Sau này thuật toán regression bounding box sẽ 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. Nếu trường hợp có từ 2 anchor boxes trở lên cùng bao quanh vật thể thì ta sẽ xác định anchor box mà IoU với ground truth bounding box cao nhất.

|  |
| --- |
|  |

Nhìn vào hình ta có thể thấy, Từ Cell ita xác định được 3 anchor boxes viền xanh như trong hình. Cả 3 anchor boxes này đều giao nhau với bounding box của vật thể. Tuy nhiên chỉ anchor box có đường viền dày nhất màu xanh được lựa chọn làm anchor box cho vật thể bởi nó có IoU so với ground truth bounding box là cao nhất.

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

Một số trường hợp 2 vật thể bị trùng mid point, mặc dù rất hiếm khi xảy ra, nhưng khi gặp trường hợp này thuật toán sẽ rất khó xác định được class cho chúng.

## **5. HÀM MẤT MÁT (LOSS FUNCTION)**

Hàm loss function của YOLO chia thành 2 phần:  (localization loss) đo lường sai số của bounding box và (confidence loss) đo lường sai số của phân phối xác xuất các classes .

|  |
| --- |
|  |

+ : Hàm indicator có giá trị 0 ,1 nhằm xác định xem cell i có chứa vật thể hay không. Bằng 1 nếu chứa vật thể và 0 nếu không chứa.

+ : Cho biết bounding box thứ i có phải là bounding box của vật thể được dự đoán hay không.

+ : Điểm tin cậy của ô i. ( Pr(Object)∗ IOU(pred,truth) )

+ : Điểm tự tin dự đoán.

+ C: Tập hợp tất cả các lớp.

+ Xác suất có điều kiện, có hay không ô i có chứa một đối tượng của lớp c C.

+ (C) : Xác suất có điều kiện dự đoán.

Có thể ban đầu công thức trên khá khó hiểu với người bắt đầu. Chúng ta hãy hiểu đơn giản hóa mục đích của chúng:

Là hàm mất mát của bounding box dự báo so với thực tế.

Là hàm mất mát của phân phối xác suất. Trong đó tổng đầu tiên là mất mát của dự đoán có vật thể trong cell hay không? Và tổng thứ 2 là mất mát của phân phối xác suất nếu có vật thể trong cell.

Ngoài ra để điều chỉnh phạt loss function trong trường hợp dự đoán sai bounding box ta thông qua hệ số điều chỉnh và ta muốn giảm nhẹ hàm loss function trong trường hợp cell không chứa vật thể bằng hệ số điều chỉnh .

## **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.

Dự đoán bounding box sao cho nó sẽ không lệch khỏi vị trí trung tâm quá nhiều. Nếu bounding box dự đoán có thể đặt vào bất kỳ phần nào của hình ảnh, như trong mạng regional proposal network, việc huấn luyện mô hình có thể trở nên không ổn định.

Cho một anchor box có kích thước ( , ) tại cell nằm trên feature map với góc trên cùng bên trái của nó là (, ), mô hình dự đoán 4 tham số (,,, ) 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 (,) và kích thước , ) thông qua hàm sigmoid và hàm exponential như các công thức bên dưới:

|  |
| --- |
|  |

Ngoài ra do các tọa 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 dụng hàm sigmoid giúp ta giới hạn được tọa độ không vượt quá xa các ngưỡng này.

|  |
| --- |
|  |

  Công thức ước lượng bounding box từ anchor box. Hình chữ nhật nét đứt bên ngoài là anchor box có kích thước là (,). Tọa độ của một bounding box sẽ được xác định dựa trên đồng thời cả anchor box và cell mà nó thuộc về. Điều này giúp kiểm soát vị trí của bounding box dự đoán đâu đó quanh vị trí của cell và bounding box mà không vượt quá xa ra bên ngoài giới hạn này. Do đó quá trình huấn luyện sẽ ổn định hơn rất nhiều so với YOLO version 1.

## **7 . NON-MAX SUPPRESSION**

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 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 2.6. Hình ảnh minh họa non-max suppression

  Non-max suppression. Từ 3 bounding box ban đầu cùng bao quanh chiếc xe đã giảm xuống còn một bounding box cuối cùng.

Các bước của non-max suppression:

Step 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 suất chứa vật thể nhỏ hơn một ngưỡng threshold nào đó, thường là 0.5.

Step 2: Đối với các bouding 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 nhau rất cao. Ta sẽ xóa các bounding có có xác xuất thấp hơn và giữ lại bouding box có xác xuất cao nhất. Cuối cùng, ta thu được một bounding box duy nhất cho một vật thể.

## **8. THUẬT TOÁN SỬ DỤNG TRONG YOLOv7**

## **8.1. Khái niệm YOLOv7**

YOLOv7 là mô hình phát hiện đối tượng thời gian thực nhanh nhất và chính xác nhất cho các tác vụ thị giác máy tính. Bài báo chính thức của YOLOv7 có tên “YOLOv7: Trainable bag-of-freebies thiết lập công nghệ mới cho máy phát hiện vật thể thời gian thực” được phát hành vào tháng 7 năm 2022 bởi Chien-Yao Wang, Alexey Bochkovskiy và Hong-Yuan Mark Liao .

## **8.2. Sự khác biệt giữa các phiên bản YOLOv7**

Các mô hình YOLOv7 cơ bản khác nhau bao gồm YOLOv7, YOLOv7-tiny và YOLOv7-W6:

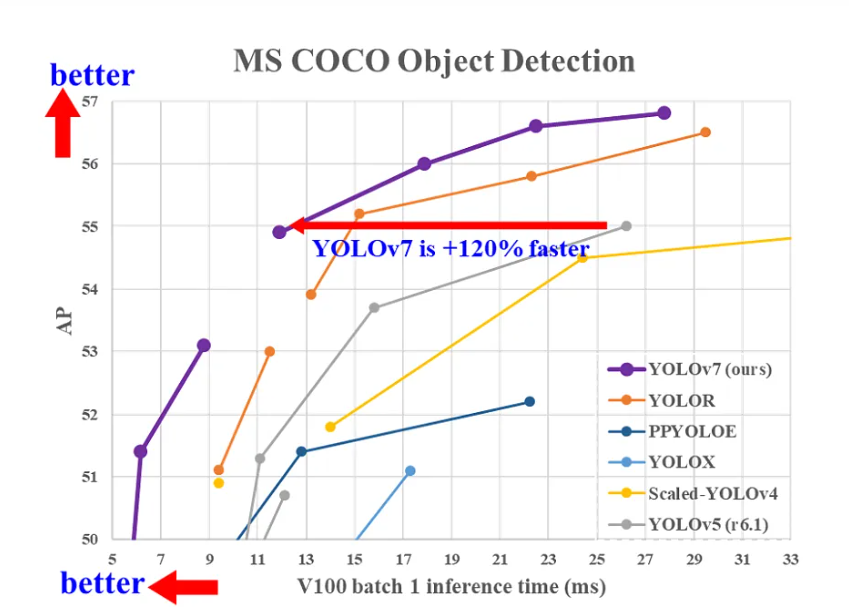
+ YOLOv7 là mô hình cơ bản được tối ưu hóa cho tính toán GPU thông thường.

+ YOLOv7-tiny là một mô hình cơ bản được tối ưu hóa cho GPU cạnh. Hậu tố “nhỏ” của mô hình thị giác máy tính có nghĩa là chúng được tối ưu hóa cho Edge AI và khối lượng công việc học sâu, đồng thời nhẹ hơn để chạy ML trên thiết bị điện toán di động hoặc máy chủ và thiết bị biên phân tán. Mô hình này rất quan trọng đối với các ứng dụng thị giác máy tính trong thế giới thực phân tán. So với các phiên bản khác, YOLOv7-tiny được tối ưu hóa cạnh sử dụng ReLU rò rỉ làm chức năng kích hoạt, trong khi các mô hình khác sử dụng SiLU làm chức năng kích hoạt.

+ YOLOv7-W6 là một mô hình cơ bản được tối ưu hóa cho điện toán đám mây GPU. Các đơn vị đồ họa đám mây (GPU) như vậy là các phiên bản máy tính để chạy các ứng dụng nhằm xử lý khối lượng công việc khổng lồ của AI và học sâu trên đám mây mà không yêu cầu phải triển khai GPU trên thiết bị người dùng cục bộ. Các biến thể khác bao gồm YOLOv7-X, YOLOv7-E6 và YOLOv7-D6, có được bằng cách áp dụng phương pháp chia tỷ lệ ghép được đề xuất (xem thêm kiến ​​trúc YOLOv7 bên dưới) để mở rộng chiều sâu và chiều rộng của toàn bộ mô hình.

## **8.3. Hiệu suất phát hiện đối tượng YOLOv7** Hiệu suất YOLOv7 được đánh giá dựa trên các phiên bản YOLO trước đó (YOLOv4 và YOLOv5) và YOLOR là đường cơ sở. Các mô hình đã được đào tạo với các cài đặt giống nhau. YOLOv7 mới cho thấy sự cân bằng giữa tốc độ và độ chính xác tốt nhất so với các máy dò vật thể hiện đại.

Nhìn chung, YOLOv7 vượt qua tất cả các thiết bị dò tìm vật thể trước đây về cả tốc độ và độ chính xác, dao động từ 5 FPS đến 160 FPS . Thuật toán YOLO v7 đạt được độ chính xác cao nhất trong số tất cả các mô hình phát hiện đối tượng thời gian thực khác - trong khi đạt được 30 FPS hoặc cao hơn khi sử dụng GPU V100.



Hình 2.7.Hình ảnh so sánh các mô hình YOLO khác

So với các mô hình Cascade- Mask R-CNN hoạt động tốt nhất , YOLOv7 đạt được độ chính xác cao hơn 2% với tốc độ suy luận tăng đáng kể (nhanh hơn 509%). Điều này rất ấn tượng vì các phiên bản R-CNN như vậy sử dụng kiến ​​trúc nhiều bước mà trước đây đã đạt được độ chính xác phát hiện cao hơn đáng kể so với kiến ​​trúc máy dò một giai đoạn. YOLOv7 vượt trội hơn YOLOR , YOLOX, Scaled-YOLOv4, YOLOv5, DETR, ViT Adapter-B và nhiều thuật toán phát hiện đối tượng khác về tốc độ và độ chính xác.



Hình 2.8.Tham số của các mô hình YOLO

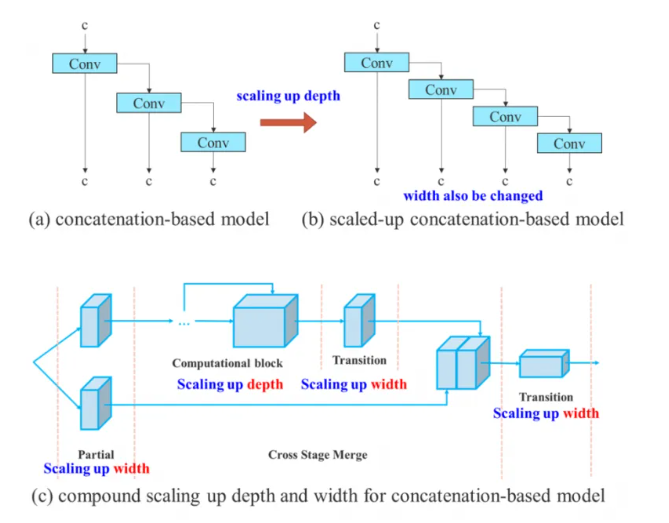
### **8.4.Kiến trúc YOLOv7** Kiến trúc YOLOv7 dựa trên kiến ​​trúc mô hình YOLO trước đó, cụ thể là YOLOv4, Scaled YOLOv4 và YOLO-R. **Tỷ lệ mô hình phức hợp YOLOv7**

Mục đích chính của việc mở rộng mô hình là điều chỉnh các thuộc tính chính của mô hình để tạo ra các mô hình đáp ứng nhu cầu của các yêu cầu ứng dụng khác nhau. Ví dụ, chia tỷ lệ mô hình có thể tối ưu hóa chiều rộng mô hình (số kênh), chiều sâu (số giai đoạn) và độ phân giải (kích thước hình ảnh đầu vào).

Trong các cách tiếp cận truyền thống với các kiến ​​trúc dựa trên nối (ví dụ: ResNet hoặc PlainNet), các yếu tố tỷ lệ khác nhau không thể được phân tích độc lập và phải được xem xét cùng nhau. Ví dụ, độ sâu mô hình mở rộng quy mô sẽ gây ra sự thay đổi tỷ lệ giữa kênh đầu vào và kênh đầu ra của lớp chuyển tiếp, do đó có thể dẫn đến việc giảm mức sử dụng phần cứng của mô hình.

Đây là lý do tại sao YOLOv7 giới thiệu tỷ lệ mô hình ghép cho một mô hình dựa trên nối. Phương pháp chia tỷ lệ kết hợp cho phép duy trì các thuộc tính mà mô hình có ở thiết kế ban đầu và do đó duy trì cấu trúc tối ưu.

Và đây là cách hoạt động của việc mở rộng mô hình phức hợp: Ví dụ, việc mở rộng hệ số chiều sâu của một khối tính toán cũng yêu cầu sự thay đổi trong kênh đầu ra của khối đó. Sau đó, tỷ lệ hệ số chiều rộng được thực hiện với cùng một mức độ thay đổi trên các lớp chuyển tiếp.

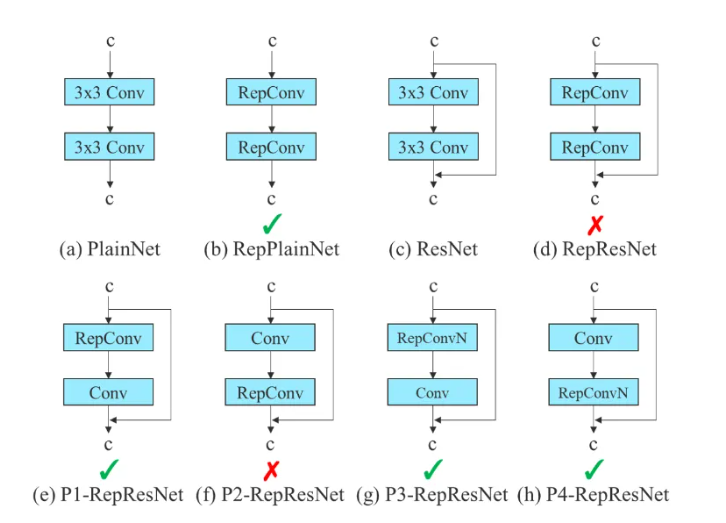


Hình 2.9. Convolutional

**Tích chập được tham số hóa lại được lập kế hoạch**

Trong khi RepConv đã đạt được hiệu suất tuyệt vời trong các kiến ​​trúc VGG , việc áp dụng trực tiếp trong ResNet hoặc DenseNet dẫn đến mất độ chính xác đáng kể. Trong YOLOv7, kiến ​​trúc của tích chập được tham số hóa lại đã lên kế hoạch sử dụng RepConv mà không có kết nối nhận dạng (RepConvN).

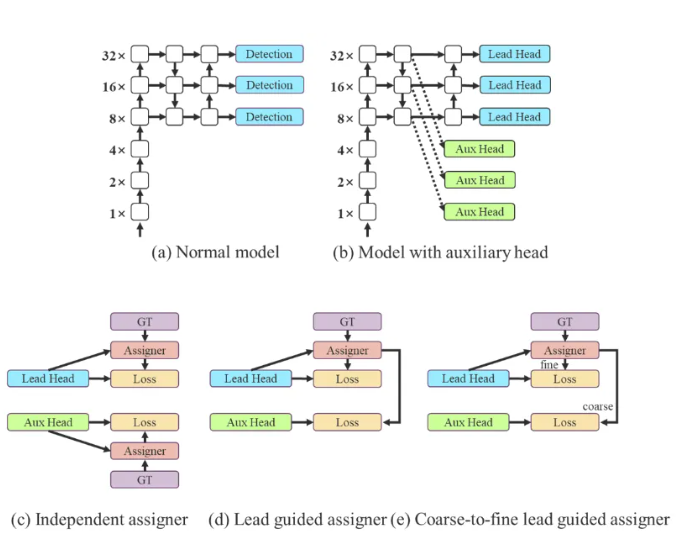
Ý tưởng là để tránh có một kết nối đồng nhất khi một lớp chập có dư hoặc nối được thay thế bằng tích chập được tham số hóa lại.



Hình 2.10.Mô hình convolutional phù hợp

**Thô cho phụ trợ và tốt cho mất chì**

Một kiến ​​trúc YOLO bao gồm xương sống, cổ và đầu. Phần đầu chứa các đầu ra mô hình dự đoán. Lấy cảm hứng từ Deep Supervision, một kỹ thuật thường được sử dụng để đào tạo mạng nơ-ron sâu , YOLOv7 không giới hạn ở một đầu duy nhất. Đầu chịu trách nhiệm cho đầu ra cuối cùng được gọi là đầu dẫn , và đầu được sử dụng để hỗ trợ đào tạo ở các lớp giữa được gọi là đầu phụ .

Ngoài ra, và để tăng cường đào tạo mạng sâu, một cơ chế Gán nhãn đã được giới thiệu để xem xét các kết quả dự đoán mạng cùng với sự thật cơ bản và sau đó chỉ định các nhãn mềm. So với cách gán nhãn truyền thống đề cập trực tiếp đến chân lý cơ bản để tạo nhãn cứng dựa trên các quy tắc đã cho, nhãn mềm đáng tin cậy sử dụng các phương pháp tính toán và tối ưu hóa cũng xem xét chất lượng và phân phối đầu ra dự đoán cùng với giá trị thực cơ bản.  
 

Hình 2.11.Đầu vào và đầu ra của YOLO

PHẦN III

XÂY DỰNG VÀ TRIỂN KHAI CHƯƠNG TRÌNH

## **1. BÀI TOÁN NHẬN DẠNG**

Nhiệm vụ chính của dự án là nhận dạng được ngọn lửa trong hình ảnh và video. Và đặc biệt hơn là nhận diện trực tiếp từ wedcam hoặc trên camera giám sát.

## **2. CHUẨN BỊ DỮ LIỆU**

Dữ liệu phục vụ cho dự án bao gồm dữ liệu hình ảnh được lưu dưới dạng jpg và dữ liệu số được lưu dưới dạng txt. Bao gồm 3591 file hình ảnh và 3591 file dữ liệu số.

Ví dụ mình họa 1 file hình ảnh và 1 file số



Hình 3.1

## **3. XÂY DỰNG CHƯƠNG TRÌNH THỬ NGHIỆM**

## **3.1.Demo chương trình**

Thực hiện lệnh kết nối tới drive

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

|  |
| --- |
| %cd /content/drive/MyDrive/YOLOV7\_Train/  !git clone https://github.com/augmentedstartups/yolov7 |

Thực hiện truy cập tới trang website và sao chép mã nguồn YOLOv7 về drive

Thực hiện cài đặt các thư viện cần thiết để train YOLOv7

|  |  |
| --- | --- |
| |  | | --- | | %cd /content/drive/MyDrive/YOLOV7\_Train/yolov7/  !pip install -r requirements.txt | |

Thực hiện tải weight pretrain về để nhận diện thử

|  |
| --- |
| %cd /content/drive/MyDrive/YOLOV7\_Train/yolov7/  !mkdir pretrain  !git clone https://github.com/WongKinYiu/yolov7/releases/download/v0.1/yolov7.pt pt |

Thực hiện giải nén dữ liệu Train model

|  |
| --- |
| %cd /content/drive/MyDrive/YOLOV7\_Train/  !mkdir train\_data  %cd train\_data  !unzip ../firedata.zip |

Tổ chức lại thư mục Train vì YOLOv7 yêu cầu images và labels

|  |
| --- |
| %cd /content/drive/MyDrive/YOLOV7\_Train/train\_data  !mkdir train  !mkdir train/images  !mkdir train/labels  !mv \*.jpg train/images  !mv \*.txt train/labels |

Khai báo 1 file yaml để YOLOv7

|  |
| --- |
| %cd /content/drive/MyDrive/YOLOV7\_Train/yolov7  !rm data/mydataset.yaml # nếu có  !echo 'train: ../train\_data/train' >> data/mydataset.yaml  !echo 'val: ../train\_data/train' >> data/mydataset.yaml  !echo 'nc: 1' >> data/mydataset.yaml  !echo "names: ['fire']" >> data/mydataset.yaml |

Train model YOLOv7 với bộ dữ liệu

|  |
| --- |
| %cd /content/drive/MyDrive/YOLOV7\_Train/yolov7  !python train.py --batch 8 --cfg cfg/training/yolov7.yaml --epochs 200 --data data/mydataset.yaml --weights 'pretrain/yolov7.pt' |

Tải ảnh từ internet về drive

|  |
| --- |
| %cd /content/drive/MyDrive/YOLOV7\_Train/yolov7  !mkdir test\_images  %cd test\_images  !wget https://cdn.theculturetrip.com/wp-content/uploads/2017/11/15271255494\_fcc90d3f9b\_k.jpg  !wget https://cdnimgen.vietnamplus.vn/t620/uploaded/wbxx/2020\_05\_13/85896\_3585300525545474\_a1.jpg  !wget https://media-cdn-v2.laodong.vn/Storage/NewsPortal/2020/4/22/800234/Chay-Pho-Co-9.jpg  !wget https://i-vnexpress.vnecdn.net/2019/03/23/chay-1-3737-1553317413.jpg |

|  |
| --- |
| #Nhận diện thử Lửa với weights vừa train sau 200 epochs  %cd /content/drive/MyDrive/YOLOV7\_Train/yolov7  !python detect.py --weights /content/drive/MyDrive/YOLOV7\_Train/yolov7/runs/train/exp/weights/last.pt --source test\_images/Chay-Pho-Co-9.jpg |

Nhận diện thử với lửa với weights vừa train

Xem ảnh đã nhận diện

|  |
| --- |
| from IPython.display import Image, display  display(Image(filename="/content/drive/MyDrive/YOLOV7\_Train/yolov7/runs/detect/exp21/Chay-Pho-Co-9.jpg")) |

Thêm các thư viện cần thiết cho dự án

|  |
| --- |
| import os  import sys  sys.path.append('/content/gdrive/MyDrive/yolov7')  #là một thư viện có sẵn trong Python dùng để viết CLI.  #Khi viết, chương trình của bạn sẽ định nghĩa các tham số cần thiết và argparse sẽ tìm cách truyền các tham số đó từ sys.argv.  import argparse  import time  from pathlib import Path  import cv2  import torch  import numpy as np  import torch.backends.cudnn as cudnn  from numpy import random  from models.experimental import attempt\_load  from utils.datasets import LoadStreams, LoadImages  from utils.general import check\_img\_size, check\_requirements, check\_imshow, non\_max\_suppression, apply\_classifier, \  scale\_coords, xyxy2xywh, strip\_optimizer, set\_logging, increment\_path  from utils.plots import plot\_one\_box  from utils.torch\_utils import select\_device, load\_classifier, time\_synchronized, TracedModel |

Xậy dựng lớp kích thước cho hình ảnh

|  |
| --- |
| def letterbox(img, new\_shape=(640, 640), color=(114, 114, 114), auto=True, scaleFill=False, scaleup=True, stride=32):  # Thay đổi kích thước và đệm hình ảnh trong khi đáp ứng các ràng buộc nhiều bước  shape = img.shape[:2]  if isinstance(new\_shape, int):#kiểm tra xem môt đối tượng (tham số thứ nhất) là một instance hay là một lớp con của Classinfo (tham số thứ hai).  new\_shape = (new\_shape, new\_shape)  # Tỷ lệ tỷ lệ (mới / cũ)  r = min(new\_shape[0] / shape[0], new\_shape[1] / shape[1])  if not scaleup: # chỉ giảm quy mô, không tăng quy mô (để kiểm tra mAP tốt hơn)  r = min(r, 1.0)    # Lớp đệm máy tính  ratio = r, r # tỷ lệ chiều rộng, chiều cao  new\_unpad = int(round(shape[1] \* r)), int(round(shape[0] \* r))  dw, dh = new\_shape[1] - new\_unpad[0], new\_shape[0] - new\_unpad[1] # wh padding  if auto:  # hình chữ nhật tối thiểu  dw, dh = np.mod(dw, stride), np.mod(dh, stride) # wh padding  elif scaleFill: # căng ra  dw, dh = 0.0, 0.0  new\_unpad = (new\_shape[1], new\_shape[0])  ratio = new\_shape[1] / shape[1], new\_shape[0] / shape[0] # tỷ lệ chiều rộng, chiều cao  dw /= 2 # chia đệm thành 2 bên  dh /= 2  if shape[::-1] != new\_unpad: # thay đổi kích thước  img = cv2.resize(img, new\_unpad, interpolation=cv2.INTER\_LINEAR)  top, bottom = int(round(dh - 0.1)), int(round(dh + 0.1))  left, right = int(round(dw - 0.1)), int(round(dw + 0.1))  img = cv2.copyMakeBorder(img, top, bottom, left, right, cv2.BORDER\_CONSTANT, value=color) # add border  return img, ratio, (dw, dh) |

Kích hoạt những tham số cho mô hình

|  |
| --- |
| classes\_to\_filter = None # Bạn có thể cung cấp danh sách các lớp để lọc theo tên,  source\_image\_path = "/content/drive/MyDrive/YOLOV7\_Train/yolov7/test\_images/Chay-Pho-Co-9.jpg"  opt = {    "weights": "/content/drive/MyDrive/YOLOV7\_Train/yolov7/runs/train/exp/weights/last.pt", # Đường dẫn đến trọng số của tệp trọng số mặc định dành cho mô hình  "yaml" : "/content/drive/MyDrive/YOLOV7\_Train/yolov7/data/mydataset.yaml",  "img-size": 640, # kích thước hình ảnh mặc định  "conf-thres": 0.1,# ngưỡng tin cậy để suy luận.  "iou-thres" : 0.45, # NMS IoU ngưỡng cho suy luận.  "device" : '0', # thiết bị để chạy mô hình của chúng tôi, tức là 0 hoặc 0,1,2,3 hoặc cpu  "classes" : classes\_to\_filter # danh sách các lớp để lọc hoặc Không có  } |

Khởi tạo mô hình

|  |
| --- |
| # khởi tạo mô hình và thiết lập nó để suy luận  with torch.no\_grad():  weights, imgsz = opt['weights'], opt['img-size']  set\_logging()  device = select\_device(opt['device'])  half = device.type != 'cpu'  model = attempt\_load(weights, map\_location=device) # tải mô hình FP32  stride = int(model.stride.max()) # người mẫu sải bước  imgsz = check\_img\_size(imgsz, s=stride) # kiểm tra img\_size  if half:  model.half()  names = model.module.names if hasattr(model, 'module') else model.names  colors = [[random.randint(0, 255) for \_ in range(3)] for \_ in names]  if device.type != 'cpu':  model(torch.zeros(1, 3, imgsz, imgsz).to(device).type\_as(next(model.parameters())))# chạy một lần  img0 = cv2.imread(source\_image\_path)  img = letterbox(img0, imgsz, stride=stride)[0]  img = img[:, :, ::-1].transpose(2, 0, 1) # BGR to RGB, to 3x416x416  img = np.ascontiguousarray(img)#Trả về một mảng liền kề  img = torch.from\_numpy(img).to(device) #Chuyển đổi mảng numpy thành tensor trên GPU  img = img.half() if half else img.float() # uint8 đến fp16 / 32  img /= 255.0 # 0 - 255 to 0.0 - 1.0  if img.ndimension() == 3:  img = img.unsqueeze(0)#Thêm 1 chiều với dim bằng 0 cho Tensor  # Sự suy luận  t1 = time\_synchronized()  pred = model(img, augment= False)[0]  # Áp dụng NMS  classes = None  if opt['classes']:  classes = []  for class\_name in opt['classes']:  classes.append(opt['classes'].index(class\_name))  pred = non\_max\_suppression(pred, opt['conf-thres'], opt['iou-thres'], classes= classes, agnostic= False)  t2 = time\_synchronized()  for i, det in enumerate(pred):  s = ''  s += '%gx%g ' % img.shape[2:] # print string  gn = torch.tensor(img0.shape)[[1, 0, 1, 0]]  if len(det):  det[:, :4] = scale\_coords(img.shape[2:], det[:, :4], img0.shape).round()  for c in det[:, -1].unique():  n = (det[:, -1] == c).sum() # phát hiện cho mỗi lớp  s += f"{n} {names[int(c)]}{'s' \* (n > 1)}, " # thêm vào chuỗi    for \*xyxy, conf, cls in reversed(det):  label = f'{names[int(cls)]} {conf:.2f}'  plot\_one\_box(xyxy, img0, label=label, color=colors[int(cls)], line\_thickness=3) |

Tiến hành show hình ảnh nhận đã nhận diện lên

|  |
| --- |
| from google.colab.patches import cv2\_imshow  cv2\_imshow(img0) |

Uploaded video lên drive

|  |
| --- |
| %cd /content/drive/MyDrive/YOLOV7\_Train/yolov7  !mkdir test\_video  %cd test\_video  from google.colab import files  uploaded =files.upload() |

Lấy đường dẫn của video trong drive

|  |
| --- |
| video\_path="/content/drive/MyDrive/YOLOV7\_Train/yolov7/test\_video/chay1.mp4" |

Khởi tạo các đối tượng để nhận dạng lửa trong video

|  |
| --- |
| # khởi tạo đối tượng video  video=cv2.VideoCapture(video\_path)  # thông tin video  fps=video.get(cv2.CAP\_PROP\_FPS)  w=int(video.get(cv2.CAP\_PROP\_FRAME\_WIDTH))  h =int(video.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))  nframes=int(video.get(cv2.CAP\_PROP\_FRAME\_COUNT))  # khởi tạo đối tượng để ghi đầu ra video  output=cv2.VideoWriter('output.mp4',cv2.VideoWriter\_fourcc(\*'DIVX'),fps,(w,h))  torch.cuda.empty\_cache()  # khởi tạo mô hình và thiết lập nó để suy luận  with torch.no\_grad():  weights, imgsz = opt['weights'], opt['img-size']  set\_logging()  device = select\_device(opt['device'])  half = device.type != 'cpu'  model = attempt\_load(weights, map\_location=device) # load FP32 model  stride = int(model.stride.max()) # model stride  imgsz = check\_img\_size(imgsz, s=stride) # check img\_size  if half:  model.half()  names = model.module.names if hasattr(model, 'module') else model.names  colors = [[random.randint(0, 255) for \_ in range(3)] for \_ in names]  if device.type != 'cpu':  model(torch.zeros(1, 3, imgsz, imgsz).to(device).type\_as(next(model.parameters())))    classes = None  if opt['classes']:  classes = []  for class\_name in opt['classes']:  classes.append(opt['classes'].index(class\_name))  for j in range(nframes):  ret,img0= video.read()  if ret:  img = letterbox(img0, imgsz, stride=stride)[0]  img = img[:, :, ::-1].transpose(2, 0, 1) # BGR to RGB, to 3x416x416  img = np.ascontiguousarray(img)  img = torch.from\_numpy(img).to(device)  img = img.half() if half else img.float() # uint8 to fp16/32  img /= 255.0 # 0 - 255 to 0.0 - 1.0  if img.ndimension() == 3:  img = img.unsqueeze(0)  # Inference  t1 = time\_synchronized()  pred = model(img, augment= False)[0]  pred = non\_max\_suppression(pred, opt['conf-thres'], opt['iou-thres'], classes= classes, agnostic= False)  t2 = time\_synchronized()  for i, det in enumerate(pred):  s = ''  s += '%gx%g ' % img.shape[2:] # print string  gn = torch.tensor(img0.shape)[[1, 0, 1, 0]]  if len(det):  det[:, :4] = scale\_coords(img.shape[2:], det[:, :4], img0.shape).round()  for c in det[:, -1].unique():  n = (det[:, -1] == c).sum() # detections per class  s += f"{n} {names[int(c)]}{'s' \* (n > 1)}, " # add to string  for \*xyxy, conf, cls in reversed(det):  label = f'{names[int(cls)]} {conf:.2f}'  plot\_one\_box(xyxy, img0, label=label, color=colors[int(cls)], line\_thickness=3)  print(f"{j+1} /{nframes} frames processed")  output.write(img0)  else:  break  output.release()  video.release() |

Xậy dựng các thành phần để chuyển đổi video thành hình ảnh để nhận dạng

|  |
| --- |
| # nhập phần phụ thuộc  from IPython.display import display, Javascript, Image  from google.colab.output import eval\_js  from google.colab.patches import cv2\_imshow  from base64 import b64decode, b64encode  import PIL  import io  import html  # hàm để chuyển đổi đối tượng JavaScript thành hình ảnh OpenCV  def js\_to\_image(js\_reply):  """  Params:  js\_reply: JavaScript object containing image from webcam  Returns:  img: OpenCV BGR image  """  # giải mã hình ảnh base64  image\_bytes = b64decode(js\_reply.split(',')[1])    # chuyển đổi byte thành mảng numpy  jpg\_as\_np = np.frombuffer(image\_bytes, dtype=np.uint8)  # giải mã mảng numpy thành hình ảnh OpenCV BGR  img = cv2.imdecode(jpg\_as\_np, flags=1)  return img  # hàm để chuyển đổi hình ảnh hộp giới hạn Hình chữ nhật OpenCV thành chuỗi byte base64 để được phủ trên luồng video  def bbox\_to\_bytes(bbox\_array):  """  Params:  bbox\_array: Numpy array (pixels) containing rectangle to overlay on video stream.  Returns:  bytes: Base64 image byte string  """  # chuyển đổi mảng thành hình ảnh PIL  bbox\_PIL = PIL.Image.fromarray(bbox\_array, 'RGBA')  iobuf = io.BytesIO()  # định dạng bbox thành png để trả lại  bbox\_PIL.save(iobuf, format='png')    # chuỗi trả về định dạng  bbox\_bytes = 'data:image/png;base64,{}'.format((str(b64encode(iobuf.getvalue()), 'utf-8')))  return bbox\_bytes  # JavaScript để tạo đúng luồng video trực tiếp của chúng tôi bằng cách sử dụng webcam làm đầu vào  def video\_stream():  js = Javascript('''  var video;  var div = null;  var stream;  var captureCanvas;  var imgElement;  var labelElement;    var pendingResolve = null;  var shutdown = false;    function removeDom() {  stream.getVideoTracks()[0].stop();  video.remove();  div.remove();  video = null;  div = null;  stream = null;  imgElement = null;  captureCanvas = null;  labelElement = null;  }    function onAnimationFrame() {  if (!shutdown) {  window.requestAnimationFrame(onAnimationFrame);  }  if (pendingResolve) {  var result = "";  if (!shutdown) {  captureCanvas.getContext('2d').drawImage(video, 0, 0, 640, 480);  result = captureCanvas.toDataURL('image/jpeg', 0.8)  }  var lp = pendingResolve;  pendingResolve = null;  lp(result);  }  }    async function createDom() {  if (div !== null) {  return stream;  }  div = document.createElement('div');  div.style.border = '2px solid black';  div.style.padding = '3px';  div.style.width = '100%';  div.style.maxWidth = '600px';  document.body.appendChild(div);    const modelOut = document.createElement('div');  modelOut.innerHTML = "<span>Status:</span>";  labelElement = document.createElement('span');  labelElement.innerText = 'No data';  labelElement.style.fontWeight = 'bold';  modelOut.appendChild(labelElement);  div.appendChild(modelOut);    video = document.createElement('video');  video.style.display = 'block';  video.width = div.clientWidth - 6;  video.setAttribute('playsinline', '');  video.onclick = () => { shutdown = true; };  stream = await navigator.mediaDevices.getUserMedia(  {video: { facingMode: "environment"},  deviceId: 0  });  div.appendChild(video);  imgElement = document.createElement('img');  imgElement.style.position = 'absolute';  imgElement.style.zIndex = 1;  imgElement.onclick = () => { shutdown = true; };  div.appendChild(imgElement);    const instruction = document.createElement('div');  instruction.innerHTML =  '<span style="color: red; font-weight: bold;">' +  'When finished, click here or on the video to stop this demo</span>';  div.appendChild(instruction);  instruction.onclick = () => { shutdown = true; };    video.srcObject = stream;  await video.play();  captureCanvas = document.createElement('canvas');  captureCanvas.width = 640; //video.videoWidth;  captureCanvas.height = 480; //video.videoHeight;  window.requestAnimationFrame(onAnimationFrame);    return stream;  }  async function stream\_frame(label, imgData) {  if (shutdown) {  removeDom();  shutdown = false;  return '';  }  var preCreate = Date.now();  stream = await createDom();    var preShow = Date.now();  if (label != "") {  labelElement.innerHTML = label;  }    if (imgData != "") {  var videoRect = video.getClientRects()[0];  imgElement.style.top = videoRect.top + "px";  imgElement.style.left = videoRect.left + "px";  imgElement.style.width = videoRect.width + "px";  imgElement.style.height = videoRect.height + "px";  imgElement.src = imgData;  }    var preCapture = Date.now();  var result = await new Promise(function(resolve, reject) {  pendingResolve = resolve;  });  shutdown = false;    return {'create': preShow - preCreate,  'show': preCapture - preShow,  'capture': Date.now() - preCapture,  'img': result};  }  ''')  display(js)    def video\_frame(label, bbox):  data = eval\_js('stream\_frame("{}", "{}")'.format(label, bbox))  return data |

Bắt đầu nhận dạng bằng video trực tuyến từ webcam

|  |
| --- |
| # bắt đầu phát trực tuyến video từ webcam  video\_stream()  # nhãn cho video  label\_html = 'Capturing...'  # hộp giới hạn khởi tạo để trống  bbox = ''  count = 0  with torch.no\_grad():  weights, imgsz = opt['weights'], (480,640)  set\_logging()  device = select\_device(opt['device'])  half = device.type != 'cpu'  model = attempt\_load(weights, map\_location=device) # load FP32 model  stride = int(model.stride.max()) # model stride  if half:  model.half()  names = model.module.names if hasattr(model, 'module') else model.names  colors = [[random.randint(0, 255) for \_ in range(3)] for \_ in names]  if device.type != 'cpu':  model(torch.zeros(1, 3, imgsz[0], imgsz[1]).to(device).type\_as(next(model.parameters())))  classes = None  if opt['classes']:  classes = []  for class\_name in opt['classes']:  classes.append(opt['classes'].index(class\_name))    while True:  js\_reply = video\_frame(label\_html, bbox)  if not js\_reply:  break    img0 = js\_to\_image(js\_reply["img"])  bbox\_array = np.zeros([480,640,4], dtype=np.uint8)  img = letterbox(img0, imgsz, stride=stride)[0]  img = img[:, :, ::-1].transpose(2, 0, 1) # BGR to RGB, to 3x416x416  img = np.ascontiguousarray(img)  img = torch.from\_numpy(img).to(device)  img = img.half() if half else img.float() # uint8 to fp16/32  img /= 255.0 # 0 - 255 to 0.0 - 1.0  if img.ndimension() == 3:  img = img.unsqueeze(0)  # Inference  t1 = time\_synchronized()  pred = model(img, augment= False)[0]  # Apply NMS  pred = non\_max\_suppression(pred, opt['conf-thres'], opt['iou-thres'], classes= classes, agnostic= False)  t2 = time\_synchronized()  for i, det in enumerate(pred):  s = ''  s += '%gx%g ' % img.shape[2:] # print string  gn = torch.tensor(img0.shape)[[1, 0, 1, 0]]  if len(det):  det[:, :4] = scale\_coords(img.shape[2:], det[:, :4], img0.shape).round()  for c in det[:, -1].unique():  n = (det[:, -1] == c).sum() # detections per class  s += f"{n} {names[int(c)]}{'s' \* (n > 1)}, " # add to string    for \*xyxy, conf, cls in reversed(det):  label = f'{names[int(cls)]} {conf:.2f}'  plot\_one\_box(xyxy, bbox\_array, label=label, color=colors[int(cls)], line\_thickness=3)    bbox\_array[:,:,3] = (bbox\_array.max(axis = 2) > 0 ).astype(int) \* 255  bbox\_bytes = bbox\_to\_bytes(bbox\_array)    bbox = bbox\_bytes |

### **3.2.** **Kết quả đạt được**

Về kết quả thử nghiệm chúng tôi thử nghiệm trên 2 phần đó là hình ảnh và video

+ Hình ảnh:

****

Hình 3.2.Train hình ảnh

+ Về video thì chúng tôi nhận dạng trực tiếp bằng webcam và nhận dạng video tải về từ internet.



Hình 3.3.Train video

# **KẾT LUẬN**

* + - 1. **NHỮNG VẤN ĐỀ ĐẠT ĐƯỢC**
* Tìm hiểu ngôn ngữ lập trình Python.
* Nắm được cách hoạt động của mô hình YOLO
* Nắm được kiến thức về object detection về nhận diện hành động trong hình ảnh và trong video thực tế.
* Tìm hiểu được cách thu thập nguồn dữ liệu để phục vụ cho bài toán YOLO.
  + - 1. **NHỮNG VẤN ĐỀ CHƯA ĐẠT ĐƯỢC**
* Chưa phân tích sâu về vấn đề đặt ra.
* Chưa hoàn thiện được tất cả các dự kiến của dự án.
  + - 1. **HƯỚNG PHÁT TRIỂN**
* Xây dựng chương trình giám cháy rừng từ webcam vệ tinh nhằm khắc phát hiện những ngọn lửa lúc nhỏ nhằm tránh rủi ro về tài nguyên và con người.

# **TÀI LIỆU THAM KHẢO**

<https://viso.ai/deep-learning/yolov7-guide/>

<https://viblo.asia/p/tim-hieu-ve-yolo-trong-bai-toan-real-time-object-detection-yMnKMdvr57P>

https://phamdinhkhanh.github.io/2020/03/10/DarknetGoogleColab.html