ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN, ĐHQG-HCM
KHOA MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

BÁO CÁO ĐỒ ÁN

Môn học: NT114.P21 – Đồ án chuyên ngành

ĐỀ TÀI

Phát Triển Hệ Thống Cảnh Báo Cháy Trong Nhà Máy Sử Dụng Kết Hợp Cảm Biến và Mô Hình Phát Hiện Cháy Dựa Trên YOLO

Giảng viên hướng dẫn: ThS. Đặng Lê Bảo Chương

Nhóm sinh viên thực hiện:

1. Huỳnh Ngọc Anh Kiệt 22520718
2. Nguyễn Thành Nhân 22521003

TP. HỒ CHÍ MINH, 2025

LỜI CẢM ƠN

Trước tiên, nhóm chúng em xin gửi lời cảm ơn chân thành và sâu sắc đến thầy Đặng Lê Bảo Chương, giảng viên phụ trách môn Đồ án chuyên ngành, người đã tận tình giảng dạy, hướng dẫn và hỗ trợ nhóm trong suốt quá trình thực hiện đồ án này. Những kiến thức chuyên môn quý báu cùng sự tận tâm của thầy đã giúp chúng em hiểu rõ hơn về đề tài và từng bước hoàn thành tốt nội dung được giao.

Nhóm cũng xin chân thành cảm ơn Khoa Mạng máy tính & Truyền thông đã tạo điều kiện thuận lợi về cơ sở vật chất và tài liệu tham khảo, giúp chúng em có môi trường học tập và nghiên cứu hiệu quả.

Đồng thời, chúng em xin gửi lời cảm ơn đến tất cả các thành viên trong nhóm vì tinh thần làm việc nghiêm túc, sự đoàn kết và nỗ lực không ngừng trong suốt quá trình thực hiện. Sự phối hợp nhịp nhàng và những ý tưởng đóng góp của từng thành viên đã góp phần quan trọng vào việc hoàn thành đồ án đúng tiến độ và đạt chất lượng tốt.

Cuối cùng, chúng em xin cảm ơn tất cả những người đã hỗ trợ và động viên nhóm trong quá trình thực hiện. Hy vọng qua đồ án này, chúng em sẽ góp phần cung cấp thêm kiến thức và tài liệu tham khảo hữu ích về đề tài này cho những ai quan tâm đến lĩnh vực này.

Thành phố Hồ Chí Minh, tháng 6 năm 2025

Nhóm sinh viên thực hiện

MỤC LỤC

I. NHỮNG MỤC VIẾT TẮT 4
II. MỤC TIÊU ĐỒ ÁN 4
III. CƠ SỞ LÝ THUYẾT 6
1. YOLOV11 6
2. Roboflow 14
3. Jupyter notebook 15
4. Computer Vision 16
5. Deep Learning 17
IV. PHẦN CỨNG 20
1. Raspberry Pi 5 20
2. ESP32 21
3. Cảm biến nhiệt độ DHT22 24
4. Cảm biến chất lượng không khí MQ-135 25
5. Camera 27
V. TRIỂN KHAI 28
VI. KẾT QUẢ VÀ ĐÁNH GIÁ 34
TÀI LIỆU THAM KHẢO 38

I. NHỮNG MỤC VIẾT TẮT

YOLO	You Only Look Once
SPFF	Spatial Pyramid Pooling Fast
NMS	Non-Maximum Suppression
MIPI CSI	Camera Serial Interface
CNN	Convolutional Neural Network
FPS	Frames Per Second
EDA	Electronic Design Automation
AI	Artificial Intelligence
GPU	Graphics Processing Unit
RTC	Real-Time Clock
UART	Universal Asynchronous Receiver Transmitter
GPIO	General Purpose Input/Output
CSI	Camera Serial Interface
ADC	Analog to Digital Converter
SPI	Serial Peripheral Interface
HSPI	High-speed SPI
VSPI	Virtual SPI
SDIO	Secure Digital Input Output
PWM	Pulse Width Modulation

II. MỤC TIÊU ĐỒ ÁN

Trong những năm qua, tình trạng hỏa hoạn của các nhà máy, các kho hàng hóa xảy ra khá đáng kể, gây ra những thiệt không đáng có cho các công ty, điển hình như vụ cháy bãi xe vào năm 2022 tại Thủ Đức, đây là một vụ cháy xảy ra vì một tác nhân từ ngọn lửa nhỏ lan ra ngọn lửa lớn, đây là một

tai nạn gây ra thiệt hại rất lớn về tài sản [1]. Những sự cố tai nạn này xảy thông thường là do những đám lửa nhỏ, hoặc những nguyên nhân khách quan khác như là sử dụng thuốc lá tại nơi không đúng nơi quy định, hay là xuất hiện những đám lửa nhỏ từ đó lan ra to…và nhiều tác nhân khách quan khác gây nên hậu quả đáng tiếc đến với tài sản.

Trước những vấn đề cấp thiết đó, đề tài "Phát triển hệ thống giám sát và cảnh báo nguy cơ cháy nổ trong nhà máy sử dụng cảm biến và mô hình YOLO" được xây dựng với mục tiêu thiết kế và triển khai một thiết bị camera có thể nhận diện những nguy cơ đó, từ đó có thể cảnh báo kịp thời đến với người dùng để có thể nhanh chóng xử lí, trước khi xảy ra hậu quả thiệt hại đến tài sản cũng như con người. Hệ thống giúp cho người dùng tránh những hậu quả đáng tiếc do cháy nổ trong nhà máy và các nơi chứa các tài sản lớn xảy ra, giúp cho người dùng phát hiện kịp thời và xử lí những nguy cơ đó.

Hệ thống tích hợp thị giác máy tính – giúp phân tích hình ảnh và phát hiện những nguy cơ xảy ra cháy nổ như thuốc lá, zippo, các ngọn lửa nhỏ…Kết hợp với các cảm biến nhiệt độ để có thể phát hiện kịp thời sự thay đổi thất thường trong kho hàng và nhà máy

Việc phát triển đề tài này là một bước đi mang tính ứng dụng cao, mở ra hướng giải quyết hiệu quả cho các tình huống khẩn cấp, đồng thời có mở rộng trong các lĩnh vực an ninh, công nghiệp và đô thị thông minh.

Mục tiêu tổng quan của đề tài: Xây dựng và phát triển một hệ thống thiết bị camera có thể phát hiện được các nguy cơ gây nên cháy nổ trong nhà máy, đồng thời nó có thể cảnh báo những ngọn lửa và khói để cho những người quản lý khu vực nhà máy đó có thể nhận biết và xử lí kịp thời. Hệ thống sử dụng thị giác máy tính, đồng thời cũng được tích hợp thêm cảm biến khói và nhiệt độ khi mà có sự thay đổi thất thường trong nhà máy hoặc các kho lưu trữ thì ngay lập tức sẽ báo động để xử lí kịp thời.

Mục tiêu cụ thể:

+ Thiết kế mô hình camera có khả năng quan sát và nhận diện có các nguy cơ nguy hiểm cháy nổ như thuốc lá, zippo, những ngọn lửa nhỏ (những tác nhân có thể gây ra một vụ cháy trong các kho hàng và gây ra ảnh hưởng lớn trong nhà máy sản xuất cần có môi trường sạch)

+ Ứng dụng thị giác máy tính để nhận diện hình ảnh các nguy cơ đó trong tình trạng có ánh sáng ổn định và trong cả tình trạng có ánh sáng yếu
+ Tích hợp các cảm biến nhiệt độ, cảm biến khói để có thể bổ trợ cho thị giác trong trường hợp nhiệt độ của các kho hàng bị thay đổi thất thường
+ Hiện thị màn hình camera và hiện thị các thông tin thời gian thực (Nhiệt độ phòng, vị trí mà nguy cơ cháy nổ sắp xảy ra) lên các giao diện như web và mobile app.
+ Thử nghiệm và đánh giá hiệu quả hệ thống trong các môi trường mô phỏng thực tế

III. CƠ SỞ LÝ THUYẾT

1. YOLOv11

1.1. Khái niệm

- YOLO 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 đó và tọa độ của đối tượng.
- YOLOv11 là phiên bản thứ 11 của YOLO được Ultralytics phát hành (giả định). Được cải tiến từ những phiên bản trước của YOLO, YOLOv11 giới thiệu các tính năng và tối ưu hóa mới khiến nó trở thành lựa chọn lý tưởng cho các bài toán Object Detection.

Hình 1.1.1. YOLOv11

- Các YOLOv11 series cung cấp nhiều mô hình đa dạng, mỗi mô hình chuyên biệt cho các tác vụ cụ thể trong thị giác máy tính. Các mô hình này được thiết kế để đáp ứng nhiều yêu cầu khác nhau, từ phát hiện đối tượng đến các tác vụ phức tạp hơn như phân đoạn thể hiện, phát hiện tư thế/điểm chính, phát hiện đối tượng theo hướng và phân loại.
- Mỗi biến thể của YOLOv11 series được tối ưu hóa cho nhiệm vụ tương ứng, đảm bảo hiệu suất và độ chính xác cao. Ngoài ra, các mô hình này tương thích với nhiều chế độ hoạt động khác nhau bao gồm Inference, Validation, Training và Export, tạo điều kiện thuận lợi cho việc sử dụng chúng trong các giai đoạn triển khai và phát triển khác nhau. [2]

- Các biến thể khác của YOLOv11:

Model	Filenames	Task	Inference	Validation	Training	Export
YOLO11	yolo11n.pt yolo11s.pt yolo11m.pt yolo11l.pt yolo11x.pt	Detection	✅	✅	✅	✅
YOLO11-seg	yolo11n-seg.pt yolo11s-seg.pt yolo11m-seg.pt yolo11l-seg.pt yolo11x-seg.pt	Instance Segmentation	✅	✅	✅	✅
YOLO11-pose	yolo11n-pose.pt yolo11s-pose.pt yolo11m-pose.pt yolo11l-pose.pt yolo11x-pose.pt	Pose/Keypoints	✅	✅	✅	✅
YOLO11-obb	yolo11n-obb.pt yolo11s-obb.pt yolo11m-obb.pt yolo11l-obb.pt yolo11x-obb.pt	Oriented Detection	✅	✅	✅	✅
YOLO11-cls	yolo11n-cls.pt yolo11s-cls.pt yolo11m-cls.pt yolo11l-cls.pt yolo11x-cls.pt	Classification	✅	✅	✅	✅

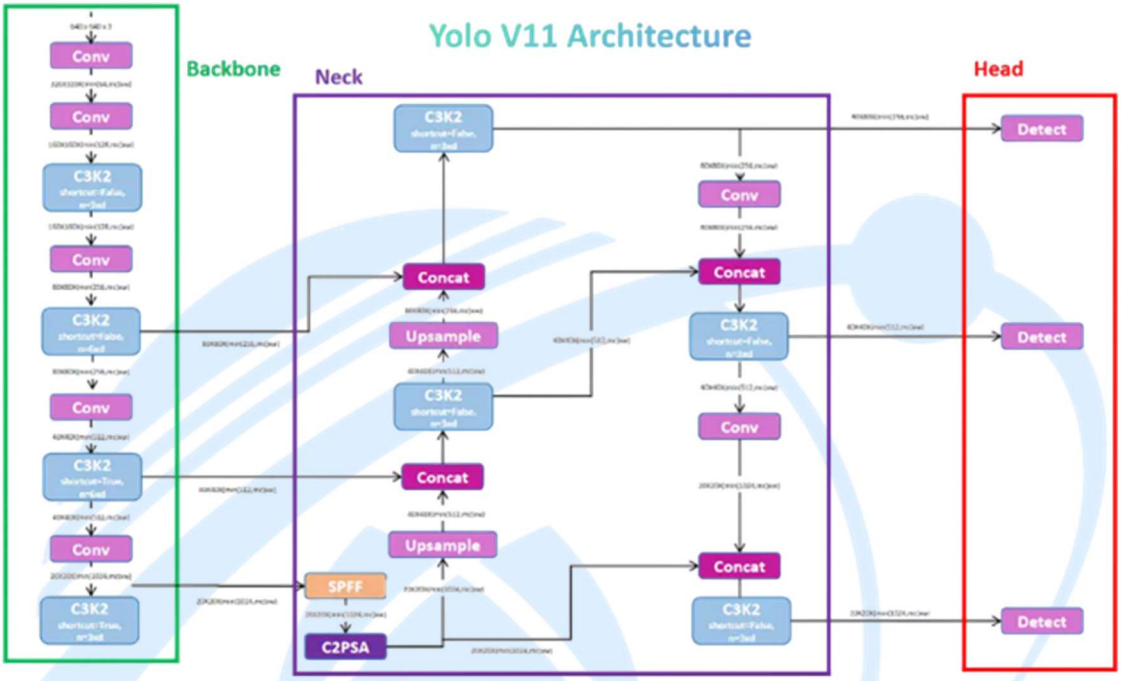
Hình 1.1.2. Biến thể YOLOv11

1.2. Thuật toán

- Thuật toán và phương pháp cốt lõi trong YOLOv11 gồm:
  + Anchor-Free Detection: Không cần anchor box và giảm độ phức tạp, trực tiếp dự đoán tọa độ box từ feature map.
  + NMS: Loại bỏ các box trùng lặp và giữ lại box có độ tin cậy cao nhất.
  + Mosaic Augmentation: Ghép 4 ảnh vào 1 ảnh để giúp mô hình tổng quát hóa tốt hơn.

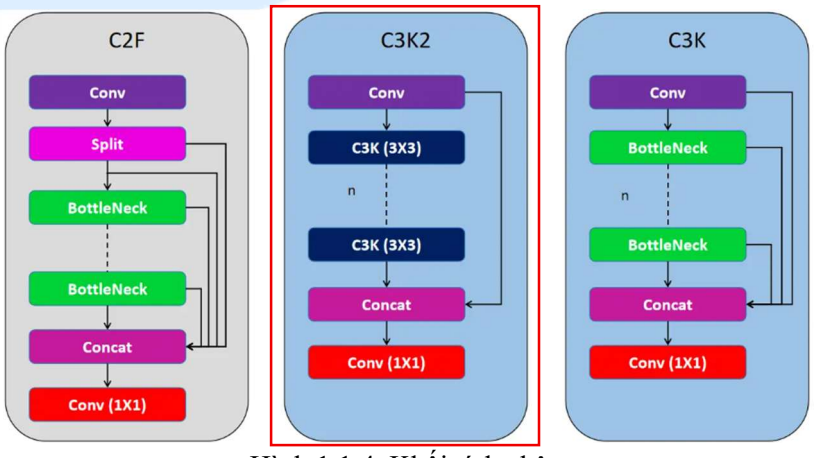
+ Auto Learning Rate và Cosine Scheduler: Tự động điều chỉnh learning rate giúp mô hình hội tụ nhanh và hiệu quả hơn.

- YOLOv11 gồm 3 phần chính:



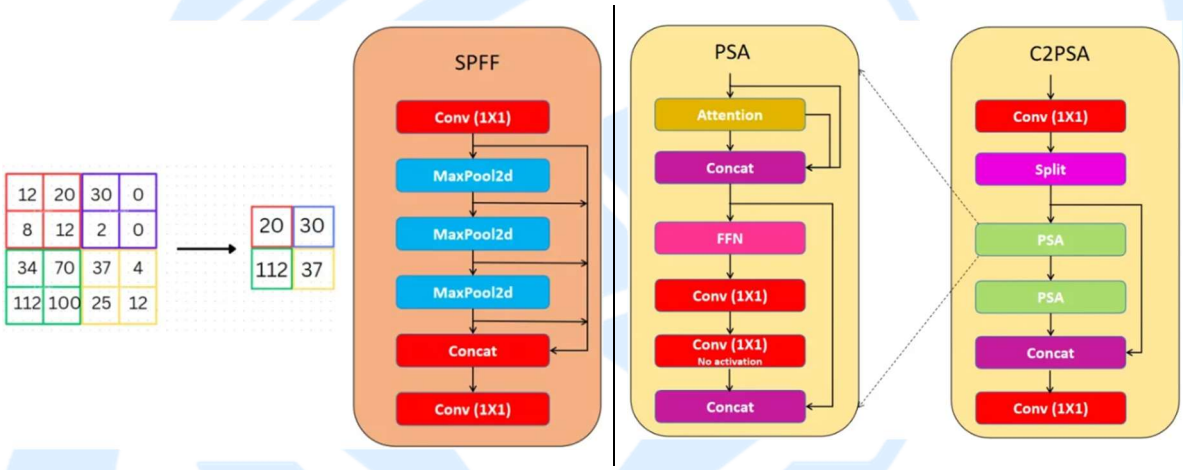
Hình 1.1.3. Cấu trúc YOLOv11

+ Backbone: Với chức năng trích xuất đặc trưng từ ảnh đầu vào, sử dụng các module C3K2 (Conv → C3K2 → Conv) thay vì C2F như YOLOv8.
  - Khối C3K2 tối ưu hóa luồng thông tin bằng cách chia nhỏ bản đồ đặc trưng và áp dụng các phép tích chập hạt nhân nhỏ (3×3), giúp giảm chi phí tính toán. Nhờ xử lý riêng lẻ và hợp nhất lại sau đó, C3K2 cải thiện khả năng biểu diễn đặc trưng với ít tham số hơn so với khối C2f của YOLOv8.



Hình 1.1.4. Khối tích chập

+ Neck: Với chức năng trộn và truyền thông tin từ nhiều cấp độ đặc trưng. YOLOv11 sử dụng mô-đun SPFF kết hợp với Upsampling và khối C2PSA để tăng cường khả năng phát hiện ở nhiều tỷ lệ khác nhau.
    - SPFF gom các đặc trưng từ các vùng khác nhau của ảnh bằng cách sử dụng nhiều lớp max-pooling với kích thước kernel khác nhau, giúp mô hình nắm bắt tốt các đối tượng có kích thước nhỏ hoặc đa dạng.
    - Khối C2PSA tích hợp cơ chế chú ý không gian, giúp mô hình tập trung vào các vùng quan trọng trong ảnh, đặc biệt là các đối tượng nhỏ hoặc bị che khuất. C2PSA sử dụng hai nhánh PSA riêng biệt để xử lý và hợp nhất đặc trưng, tối ưu hóa giữa hiệu năng và độ chính xác.

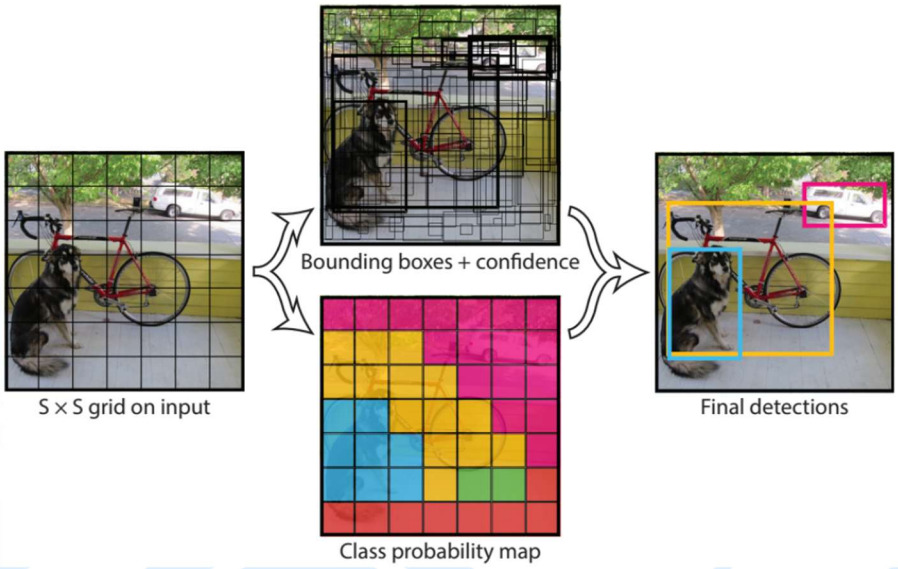


Hình 1.1.5. Mô-đun SPFF và C2PSA

+ Head: Với chức năng dự đoán bounding box, độ tin cậy và lớp của đối tượng. YOLOv11 sử dụng kiến trúc đầu phát hiện đa thang (multi-scale detection head), tương tự như các phiên bản trước, nhưng được tối ưu hóa cho độ chính xác và tốc độ:
    - Dự đoán được thực hiện từ ba bản đồ đặc trưng ở các cấp độ khác nhau (P3, P4, P5), tương ứng với các đối tượng nhỏ, trung bình và lớn.
    - Phần đầu sử dụng phương pháp anchor-free và kiến trúc decoupled head, trong đó:
        - Nhánh phân loại dự đoán lớp của từng đối tượng.
        - Nhánh hồi quy dự đoán tọa độ bounding box. [3]

1.3. Cách Yolo hoạt động

- 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$S\times S$ ô thường thì sẽ là 3×33×3,7×77×7,9×99×9... việc chia ô này có ảnh hưởng tới việc mô hình phát hiện đối tượng, mình xin trình bày ở phần sau.



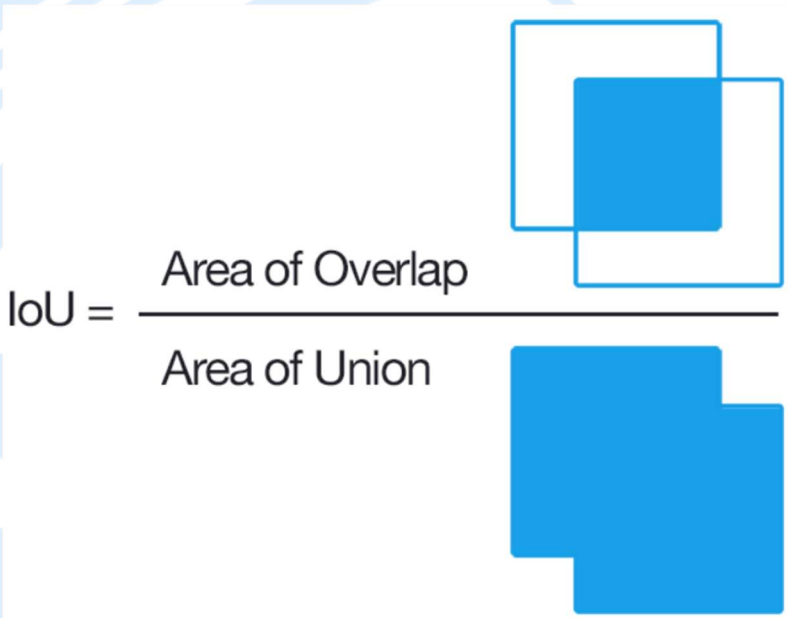
Hình 1.1.5. Mô hình hoạt động

- Với Input là 1 ảnh, đầu ra mô hình là một ma trận 3 chiều có kích thước S×S×(5×N+M)$S\times S\times(5\times N+M)$ với số lượng tham số mỗi ô là (5×N+M)$(5\times N+M)$ 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 7×77×7 ô, mỗi ô cần dự đóan 2 bounding box và 3 object : con chó, ô tô, xe đạp thì output là 7×7×137×7×13, mỗi ô sẽ có 13 tham số, kết quả trả về (7×7×2=98)(7×7×2=98) bounding box. Chúng ta sẽ cùng giải thích con số (5×N+M)$(5\times N+M)$ 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)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 YOLO1, 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.

1.4. Hàm tính IOU

- Trên ta có đề cập prediction được định nghĩa Pr(Object)∗ IOU(pred,truth)Pr(*Object*)∗ *IOU*(*pred*,*truth*), ta sẽ làm rõ hơn IOU(pred, truth) là gì. IOU (INTERSECTION OVER UNION) là hàm đánh giá độ chính xác của object detector trên tập dữ liệu cụ thể. IOU được tính bằng:



Hình 1.1.7. IOU

- Trong đó Area of Overlap là diện tích phần giao nhau giữa predicted bounding box với grouth-truth bouding box , còn Area of Union là diện tích phần hợp giữa predicted bounding box với grouth-truth bounding box. Những bounding box được đánh nhãn bằng tay trong tập traing set và test set. Nếu IOU > 0.5 thì prediction được đánh giá là tốt.

1.5. Hàm mất mát

- Hàm lỗi trong YOLO được tính trên việc dự đoán và nhãn mô hình để tính. Cụ thể hơn nó là tổng độ lõi của 3 thành phần con sau :

  - Độ lỗi của việc dự đoán loại nhãn của object - Classifycation loss

- Độ lỗi của dự đoán tọa độ tâm, chiều dài, rộng của boundary box (x, y ,w, h) - Localization loss
- Độ lỗi của việc dự đoán bounding box đó chứa object so với nhãn thực tế tại ô vuông đó - Confidence loss

- Mất phân loại:
  + Classifycation loss - độ lỗi của việc dự đoán loại nhãn cuả object, hàm lỗi này chỉ tính trên những ô vuông có xuất hiện object, còn những ô vuông khác ta không quan tâm. Classifycation loss được tính bằng công thức sau:

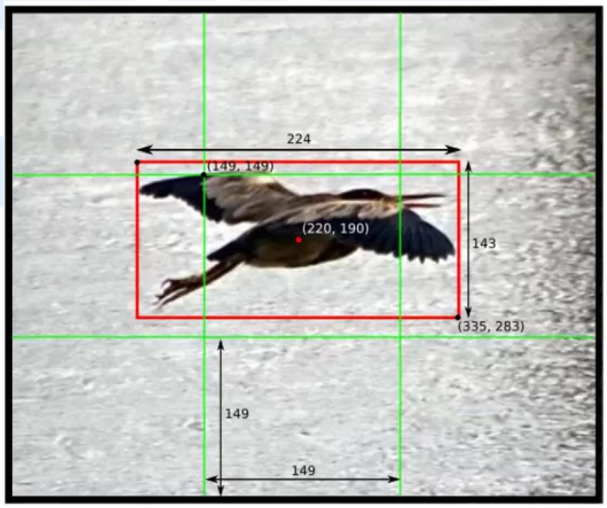
$$L_{classification} = \sum_{i=0}^{S^2} \mathbb{I}_i^{obj} \sum_{c \in class} (p_i(c) - \hat{p}_i(c))^2$$

Trong đó:

$\mathbb{I}_i^{obj}$: bằng 1 nếu ô vuông đang xét có object ngược lại bằng 0

$\hat{p}_i(c)$: là xác xuất có điều của lớp c tại ô vuông tương ứng mà mô hình dự đoán

- Mất vị trí địa lý:
  + Localization loss là hàm lỗi dùng để tính giá trị lỗi cho boundary box được dự đoán bao gồm tọa độ tâm, chiều rộng, chiều cao của so với vị trí thực tế từ dữ liệu huấn luyện của mô hình. Lưu ý rằng chúng ta không nên tính giá trị hàm lỗi này trực tiếp từ kích thức ảnh thực tế mà cần phải chuẩn hóa về [0, 1] so với tâm của bounding box. Việc chuẩn hóa này kích thước này giúp cho mô hình dự đoán nhanh hơn và chính xác hơn so với để giá trị mặc định của ảnh. Hãy cùng xem một ví dụ:



x = (220-149) / 149 = 0.48
y = (190-149) / 149 = 0.28
w = 224 / 448 = 0.50
h = 143 / 448 = 0.32

  + Giá trị hàm Localization loss được tính trên tổng giá trị lỗi dự đoán toạ độ tâm (x, y) và (w, h) của predicted bounding box với

grouth-truth bounding box. Tại mỗi ô có chưa object, ta chọn 1 boundary box có IOU (Intersect over union) tốt nhất, rồi sau đó tính độ lỗi theo các boundary box này.

+ Giá trị hàm lỗi dự đoán tọa độ tâm (x, y) của predicted bounding box và (x̂, ŷ) là tọa độ tâm của truth bounding box được tính như sau :

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2$$

+ GIá trị hàm lỗi dự đoán (w, h) của predicted bounding box so với truth bounding box được tính như sau :

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2$$

+ Với ví dụ trên thì S =7, B =2, còn λcoord là trọng số thành phần trong paper gốc tác giả lấy giá trị là 5.

- Mất lòng tin:
  + Confidence loss là độ lỗi giữa dự đoán boundary box đó chứa object so với nhãn thực tế tại ô vuông đó. Độ lỗi này tính trên cả những ô vuông chứa object và không chứa object.

$$L_{confidence} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{obj} (C_i - \hat{C}_i)^2 + \lambda_{noobject} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{noobj} (C_i - \hat{C}_i)^2$$

+ Với ví dụ trên thì S =7, B =2, còn λnoobject là trọng số thành phần trong paper gốc tác giả lấy giá trị là 0.5. Đối với các hộp j của ô thứ i nếu xuất hiệu object thì Ci =1 và ngược lại. [4]

1.6. Kết luận

- Kết luận:
  + YOLOv11 là một bước tiến đáng kể trong lĩnh vực phát hiện đối tượng, kết hợp hài hòa giữa tốc độ, độ chính xác và hiệu quả tính toán. Những cải tiến như khối C3K2 giúp trích xuất đặc trưng hiệu quả hơn với ít tham số, trong khi cơ chế chú ý không gian C2PSA giúp mô hình tập trung tốt hơn vào các vùng quan trọng của ảnh, đặc biệt là các đối tượng nhỏ hoặc bị che khuất.
  + Với khả năng phát hiện đa thang đo, YOLOv11 xử lý tốt các đối tượng có kích thước và hình dạng đa dạng mà không làm chậm quá trình suy

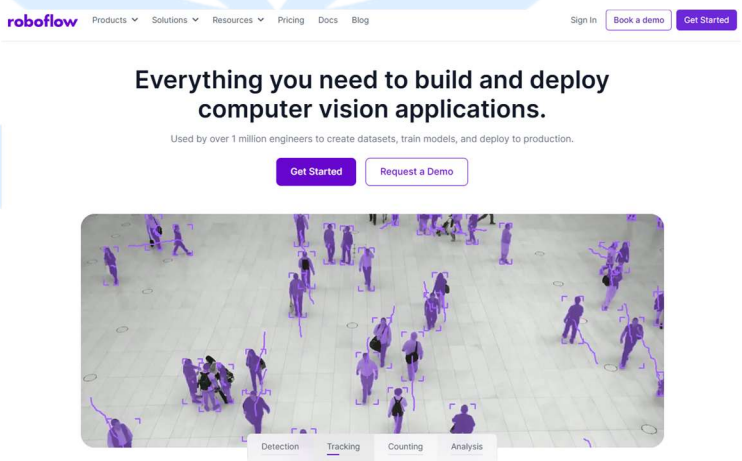
luận. Nhờ đó, mô hình đạt được điểm mAP cao và tốc độ FPS ấn tượng, phù hợp cho các ứng dụng thời gian thực như lái xe tự động, phân tích video, và chẩn đoán hình ảnh y tế.

+ YOLOv11 không chỉ nâng cao hiệu suất mà còn giữ được tính dễ triển khai, là lựa chọn lý tưởng cho cả nhà nghiên cứu, kỹ sư thị giác máy tính và học viên trong nhiều kịch bản ứng dụng – từ thiết bị biên đến hệ thống phân tích ở quy mô lớn.

- Những điểm nổi bật:
  + Tốc độ và độ chính xác vượt trội, cải thiện hơn so với YOLOv8 và YOLOv10.
  + Khối C3K2 và cơ chế chú ý C2PSA nâng cao khả năng trích xuất và tập trung đặc trưng.
  + Phù hợp với ứng dụng thực tế như xe tự hành và phân tích hình ảnh y tế.
  + Phát hiện đa thang đo hiệu quả, xử lý tốt các cấu trúc đối tượng phức tạp.
  + Thiết kế thân thiện và dễ sử dụng, thích hợp cho nhiều đối tượng người dùng trong lĩnh vực thị giác máy tính.

2. Roboflow

- Roboflow là một nền tảng thị giác máy tính (Computer Vision Platform) giúp người dùng xây dựng mô hình thị giác máy tính một cách nhanh chóng và chính xác hơn thông qua việc cung cấp các phương tiện thu thập dữ liệu, tiền xử lý và kỹ thuật huấn luyện mô hình tốt hơn.
- Roboflow cho phép người dùng tải lên bộ dữ liệu tùy chỉnh, vẽ chú thích, điều chỉnh hướng ảnh, thay đổi kích thước ảnh, điều chỉnh độ tương phản và thực hiện tăng cường dữ liệu. Nền tảng này cũng có thể được sử dụng để huấn luyện mô hình.



Hình 1.2.1. Roboflow

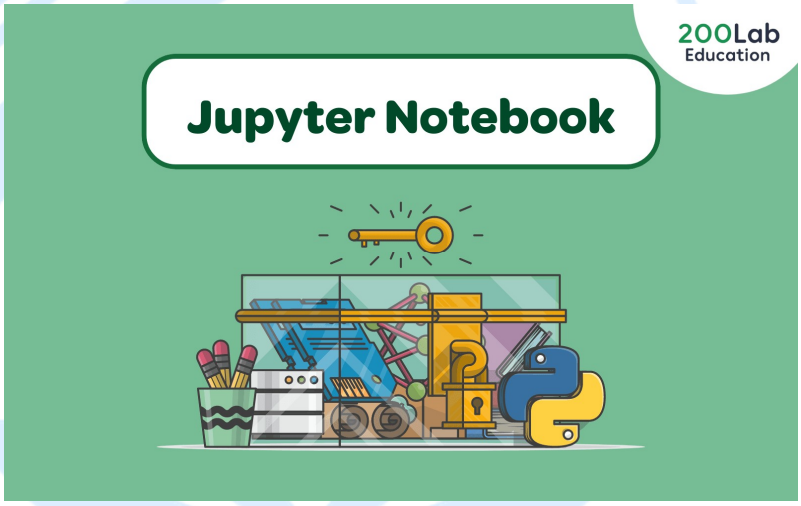
- Roboflow cũng có một công cụ chuyển chú thích chung cho phép người dùng tải lên và chuyển đổi chú thích từ một định dạng sang định dạng khác mà không

cần viết các kịch bản chuyển đổi cho bộ dữ liệu phát hiện đối tượng tùy chỉnh.

- Hiện nay nhiều quy trình kinh doanh – từ theo dõi hàng tồn kho đến phát hiện rò rỉ dầu và đếm viên thuốc – đều có thể được tự động hóa với thị giác máy tính. Roboflow cung cấp các công cụ giúp quá trình này trở nên đơn giản hơn nhiều, đồng thời nhanh chóng và chính xác hơn. Hơn 100,000 nhà phát triển và một nửa trong danh sách Fortune 100 đã và đang sử dụng các công cụ của Roboflow để xây dựng hệ thống. [5]

3. Jupyter notebook

- Jupyter Notebook là một nền tảng mã nguồn mở dành cho tính toán khoa học, cho phép người dùng tạo và chia sẻ tài liệu chứa mã lệnh, phương trình, biểu đồ và văn bản mô tả. Ban đầu gọi là IPython Notebook, đến năm 2014 được đổi tên thành Jupyter — ghép từ **Julia, Python và R**.
- Jupyter hỗ trợ hơn 40 ngôn ngữ lập trình, tích hợp Markdown để trình bày tài liệu tương tác, giúp người dùng vừa chạy mã vừa trình bày nội dung học thuật, nghiên cứu hay báo cáo một cách sinh động.

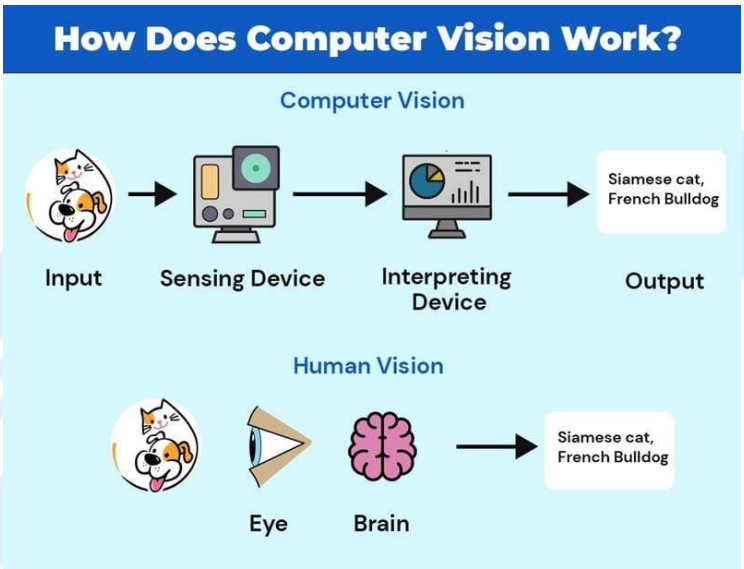


Hinh1.3.1. Jupyter Notebook

- Lợi ích của Jupyter notebook:
  + Phân tích khám phá dữ liệu (EDA): Cho phép chạy từng ô mã (code cell) độc lập và xem kết quả ngay lập tức, rất phù hợp với công việc khám phá dữ liệu.
  + Lưu trạng thái ô (cell): Tự động lưu kết quả và trạng thái của từng ô, thuận tiện trong việc xử lý dữ liệu lớn hoặc huấn luyện mô hình.
  + Độc lập ngôn ngữ: Với định dạng JSON, Jupyter hỗ trợ đa ngôn ngữ và có thể dễ dàng tích hợp với nhiều công cụ khác.
  + Trực quan hóa dữ liệu: Hỗ trợ biểu đồ và đồ họa thông qua thư viện như Matplotlib, Plotly, Bokeh,... giúp minh họa dữ liệu sinh động.
  + Tương tác trực tiếp: Thông qua **ipywidgets**, người dùng có thể tương tác trực tiếp với mã và dữ liệu trong trình duyệt, tạo ra trải nghiệm lập trình động và linh hoạt. [6]

4. Computer Vision

- **Computer Vision (Thị giác máy tính)** là một lĩnh vực con của trí tuệ nhân tạo (AI), nghiên cứu cách để máy tính có thể "nhìn" và "hiểu" thế giới xung quanh thông qua hình ảnh và video. Mục tiêu của thị giác máy tính là phát triển các hệ thống có thể nhận diện, phân tích và xử lý thông tin từ hình ảnh hoặc video, tương tự như cách con người xử lý thị giác của mình.



Hình 1.4.1. Computer vision

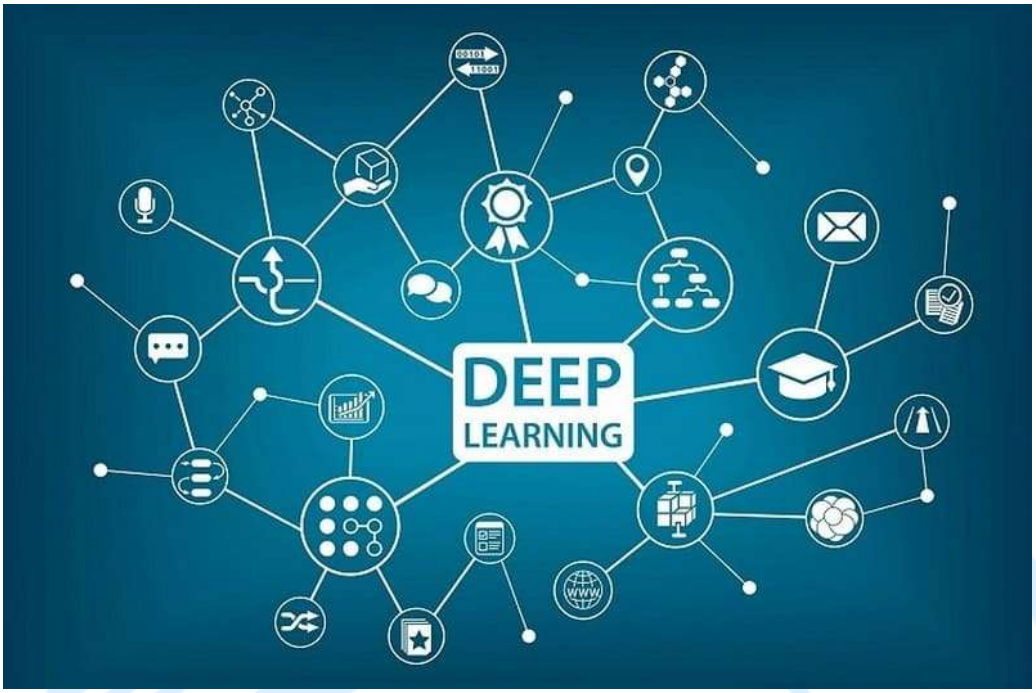
- **Computer Vision** đóng vai trò quan trọng trong nhiều ứng dụng hiện đại từ nhận diện khuôn mặt, phân loại hình ảnh đến các hệ thống lái xe tự động. Việc ứng dụng thị giác máy tính vào thực tiễn ngày càng trở nên phổ biến nhờ vào sự phát triển của các thuật toán học máy và các mạng nơ-ron nhân tạo.
- **Computer Vision** được phát triển và hoạt động dựa trên các kỹ thuật và thuật toán phức tạp nhằm mô phỏng quá trình nhận diện và xử lý hình ảnh của con người như sau:
    + **Thu thập hình ảnh (Image Acquisition):** Bước đầu tiên trong quá trình Computer Vision là thu thập dữ liệu hình ảnh hoặc video từ nhiều nguồn khác nhau như camera, cảm biến hoặc các cơ sở dữ liệu hình ảnh đã có sẵn.
    + **Tiền xử lý (Preprocessing):** Trước khi bắt đầu phân tích, các chuyên gia sẽ tiền xử lý hình ảnh để làm sạch dữ liệu, loại bỏ nhiễu, sửa các biến dạng, điều chỉnh độ sáng hoặc độ tương phản để cải thiện chất lượng hình ảnh. Mục tiêu của tiền xử lý là đảm bảo các thuật toán nhận được đầu vào hình ảnh chính xác và đáng tin cậy.
    + **Trích xuất đặc trưng (Feature Extraction):** Trong bước này, hệ thống sẽ nhận diện và trích xuất các đặc điểm đặc trưng của hình ảnh như các cạnh, góc, kết cấu, hình dạng hoặc phân bố màu sắc. Việc này giúp đơn giản hóa dữ liệu, giữ lại các thông tin quan trọng và loại bỏ những chi tiết không cần thiết, từ đó tạo ra đầu vào hiệu quả cho các bước phân tích

và phân loại sau.

+ **Phân loại đối tượng (Object Classification):** Sau khi trích xuất các đặc trưng và hoàn thiện huấn luyện mô hình học sâu (deep learning), hệ thống có thể phân loại và nhận diện các đối tượng trong hình ảnh bằng cách so sánh các đặc trưng đã trích xuất với các mẫu mà nó đã học được trong quá trình huấn luyện. Đây là quá trình giúp máy tính phân biệt các đối tượng khác nhau, chẳng hạn như nhận diện hình ảnh của một con mèo hay con chó.
+ **Nhận diện đối tượng (Object Identification):** Nhận diện đối tượng nhằm xác định chính xác các đối tượng cụ thể trong một bức tranh. Nhiệm vụ của giai đoạn này là nhận diện các đối tượng hoặc cá thể trong một cảnh giúp hệ thống không chỉ phân loại mà còn biết chính xác đối tượng nào đang xuất hiện trong hình ảnh.
+ **Theo dõi đối tượng (Object Tracking):** Quá trình theo dõi đối tượng trong thị giác máy tính liên quan đến việc theo dõi sự chuyển động của một đối tượng qua các khung hình liên tiếp trong video. Hệ thống sẽ xác định vị trí của đối tượng trong mỗi khung hình và duy trì sự liên tục trong suốt quá trình chuyển động. Các thuật toán theo dõi sử dụng nhiều kỹ thuật khác nhau để đảm bảo việc theo dõi chính xác, ngay cả trong các điều kiện khó khăn như ánh sáng yếu hoặc khi đối tượng bị che khuất. [7]

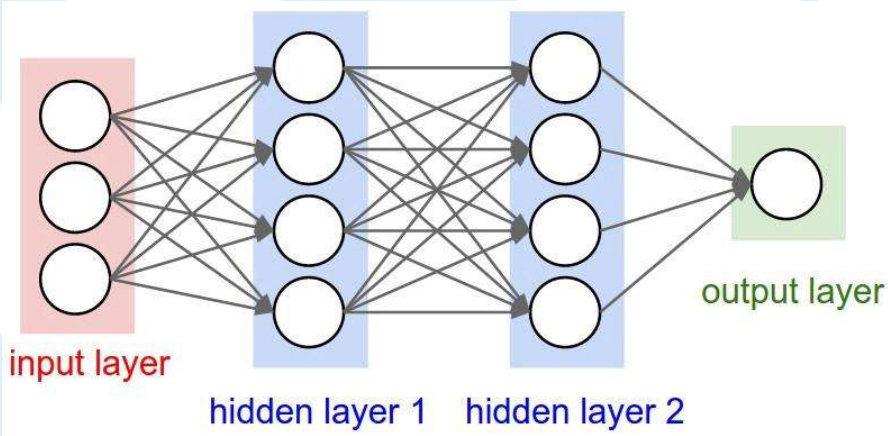
5. Deep Learning

- **Deep Learning** (Học sâu) là một phần quan trọng của trí tuệ nhân tạo (AI), đặc biệt nổi tiếng với khả năng mô phỏng cách con người học. Điểm đặc biệt của Deep Learning là sự sâu và phức tạp của mạng neuron nhân tạo mà nó sử dụng. Mạng này được thiết kế để tự động học từ dữ liệu và trích xuất các đặc trưng quan trọng để giải quyết các nhiệm vụ phức tạp.



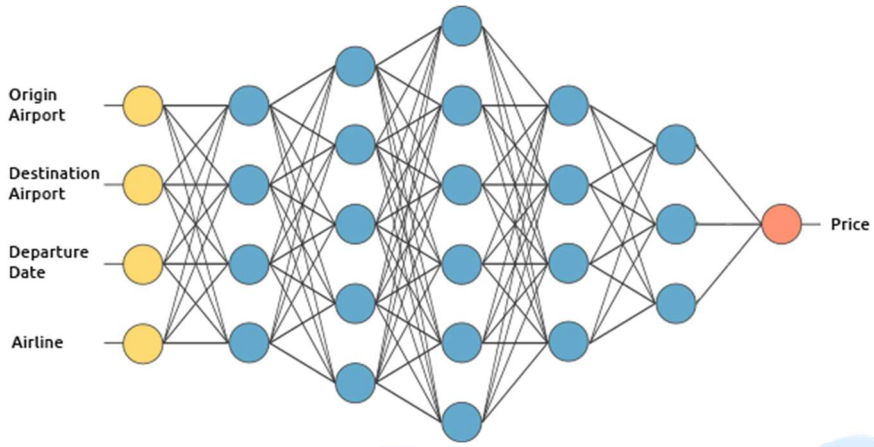
Hình 1.5.1. Deep Learning

- Cách hoạt động:
  + Một neural network bao gồm nhiều layer khác nhau, càng nhiều layer thì mạng sẽ càng "sâu". Trong mỗi lớp là các node nút mạng và được liên kết với các layer khác bên cạnh.



Hình 1.5.2. Neural network

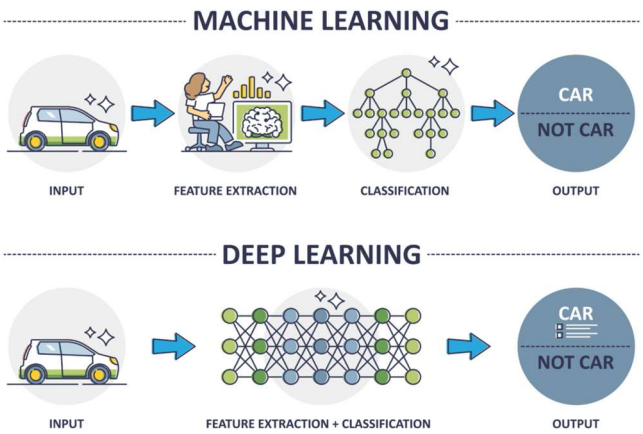
  + Mỗi kết nối giữa các node sẽ có một trọng số tương ứng, trọng số càng cao thì mức độ ảnh hưởng của kết nối này đến mạng neural càng lớn.
  + Mỗi neural sẽ có một chức năng kích hoạt, về cơ bản chịu trách nhiệm "chuẩn hóa" đầu ra từ neural này.
  + Dữ liệu được người dùng nhập vào neural network đi qua tất cả các layer và trả về kết quả ở layer cuối cùng, được gọi là output layer.



Hình 1.5.3. Deep Learning Neural Network

+ Trong quá trình đào tạo mô hình neural network, các trọng số sẽ được thay đổi và nhiệm vụ của mô hình là tìm tập giá trị của các trọng số sao cho phán đoán đúng nhất.
+ Các hệ thống Deep Learning yêu cầu phần cứng rất mạnh để có thể xử lý lượng lớn dữ liệu và thực hiện các phép tính phức tạp. Nhiều mô hình Deep Learning có thể mất hàng tuần hoặc thậm chí hàng tháng để triển khai trên phần cứng tiên tiến nhất hiện nay.

- Mối liên hệ giữa Deep Learning và Machine Learning: Deep Learning là một phân nhánh của Machine Learning. Cả hai đều liên quan đến việc máy tính học hỏi từ dữ liệu để thực hiện các nhiệm vụ mà trước đây đòi hỏi sự can thiệp của con người. Machine Learning bao gồm nhiều phương pháp và thuật toán, trong khi Deep Learning tập trung vào việc sử dụng các mạng neuron nhân tạo sâu và phức tạp để thực hiện học tập tự động. Deep Learning có thể coi là một phân nhánh tiến xa hơn của Machine Learning, đặc biệt hiệu quả trong việc xử lý dữ liệu lớn và phức tạp, như trong việc nhận diện hình ảnh, xử lý ngôn ngữ tự nhiên và nhiều ứng dụng khác. [8]

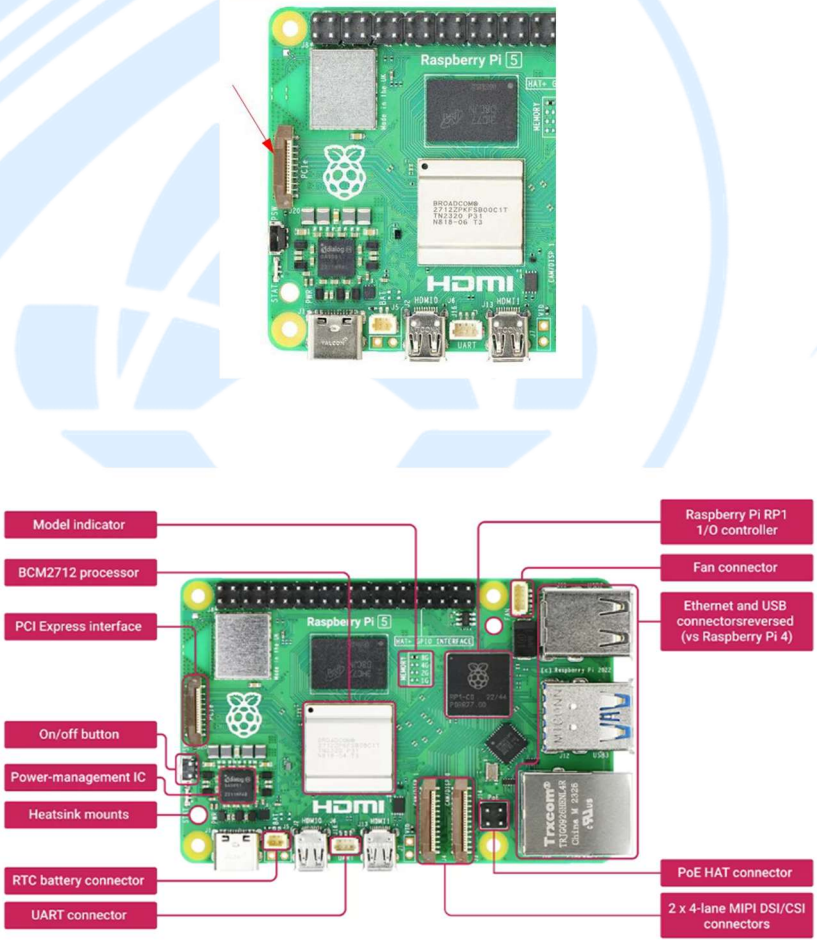


Hình 1.5.4. Deep Learning vs Machine Learning

IV. PHẦN CỨNG

1. Raspberry Pi 5

- Raspberry Pi là máy tính bo mạch đơn được phát triển bởi Raspberry Pi Foundation ở Anh với mục đích ban đầu phục vụ giảng dạy khoa học máy tính cơ bản trong trường học và các nước đang phát triển. Sau đó, Raspberry Pi đã trở thành một máy vi tính rất đáng tin cậy ngay cả ngoài lĩnh vực giáo dục. Khoa học và kỹ thuật, CNTT và thậm chí cả các ngành công nghiệp đã và đang sử dụng Raspberry Pi trong hệ thống của họ.
- RPi 5 được trang bị CPU ARM Cortex-A76 tốc độ lên tới 2.4 GHz và GPU VideoCore VII. Theo hãng sản xuất, Pi 5 có hiệu suất cao hơn khoảng từ 2-3 lần so với người tiền nhiệm là Pi 4! Raspberry Pi 5 có thể là lựa chọn tốt hơn cho dự án đòi hỏi sức mạnh xử lý tốc độ cao như các chương trình AI hoặc xử lý hình ảnh đồ họa chuyên sâu.



Hình 2.1.1. Raspberry Pi 5

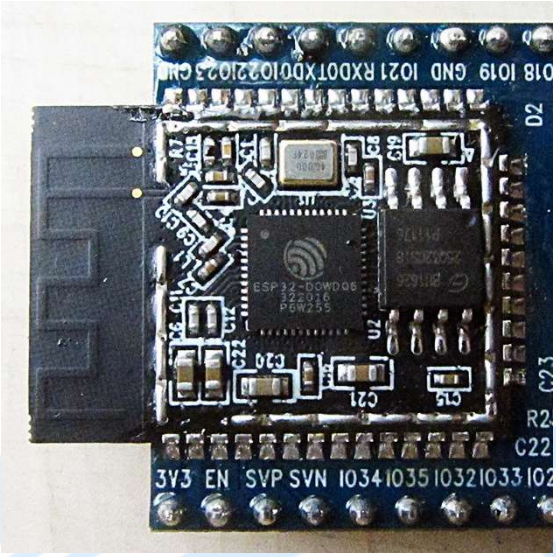
- Các thành phần chính của Raspberry Pi 5 gồm có :
  + <u>Model Indicator :</u> Chỉ thị phiên bản bo mạch, giúp người dùng dễ dàng nhận diện model trong quá trình sử dụng hoặc kiểm tra.

+ BCM2712 Processor : CPU chính: Broadcom BCM2712, ARM Cortex-A76 (4 nhân, 64-bit, xung nhịp 2.4GHz), hiệu năng xử lý cao hơn gấp 2–3 lần so với Raspberry Pi 4, phù hợp cho các ứng dụng AI, server nhẹ, desktop, ...
+ PCI Express Interface: Giao tiếp mở rộng PCIe Gen 2 x1 (qua cáp FFC), cho phép kết nối với SSD NVMe, card mạng, các thiết bị mở rộng tốc độ cao.
+ On/Off Button: Nút nguồn vật lý: hỗ trợ tắt/mở bo mạch tiện lợi, cải tiến đáng kể so với các đời trước vốn không có nút nguồn.
+ Power-Management IC: IC quản lý nguồn điện cho toàn hệ thống, Đảm bảo hiệu quả năng lượng và bảo vệ linh kiện.
+ Heatsink Mounts: Vị trí lắp bộ tản nhiệt, hỗ trợ làm mát chủ động khi hệ thống hoạt động với hiệu suất cao
+ RTC Battery Connector: Cổng kết nối pin RTC, duy trì thời gian thực kể cả khi mất nguồn chính
+ UART Connector: Giao tiếp UART phục vụ debug và lập trình hệ thống, dễ dàng kết nối với thiết bị ngoài qua chuẩn serial.
+ Raspberry Pi RP1 I/O Controller: Vi điều khiển RP1 chịu trách nhiệm điều khiển I/O (GPIO, USB, camera, v.v.), giảm tải cho CPU chính và cải thiện hiệu suất I/O.
+ Fan Connector: Cổng kết nối quạt làm mát (active cooling), hỗ trợ kiểm soát nhiệt độ tốt hơn cho các ứng dụng tải nặng.
+ Ethernet and USB Connectors: 2 cổng USB 3.0, 2 cổng USB 2.0, cổng Ethernet Gigabit.
+ PoE HAT Connector: Giao tiếp cấp nguồn qua Ethernet (Power over Ethernet), tương thích với PoE HAT để cấp nguồn mà không cần adapter riêng.
+ 13 2 x 4-lane MIPI DSI/CSI Connectors: 2 đầu cắm đa năng cho camera (CSI) và màn hình cảm ứng (DSI), mỗi đầu gồm 4 lane dữ liệu → băng thông cao hơn đời cũ.

2. ESP32

- ESP32 là một series các vi điều khiển trên một vi mạch giá rẻ, năng lượng thấp có hỗ trợ WiFi và dual-mode Bluetooth (tạm dịch: Bluetooth chế độ kép). Dòng ESP32 sử dụng bộ vi xử lý Tensilica Xtensa LX6 ở cả hai biến thể lõi kép và lõi đơn, và bao gồm các công tắc antenna tích hợp, RF balun, bộ khuếch đại công suất, bộ khuếch đại thu nhiễu thấp, bộ lọc và module quản lý năng lượng. ESP32 được chế tạo và phát triển bởi Espressif Systems, một công ty Trung Quốc có trụ sở tại Thượng Hải, và được sản xuất bởi TSMC bằng cách sử dụng công nghệ 40 nm.ESP32 là sản phẩm kế thừa từ vi điều khiển

ESP8266.

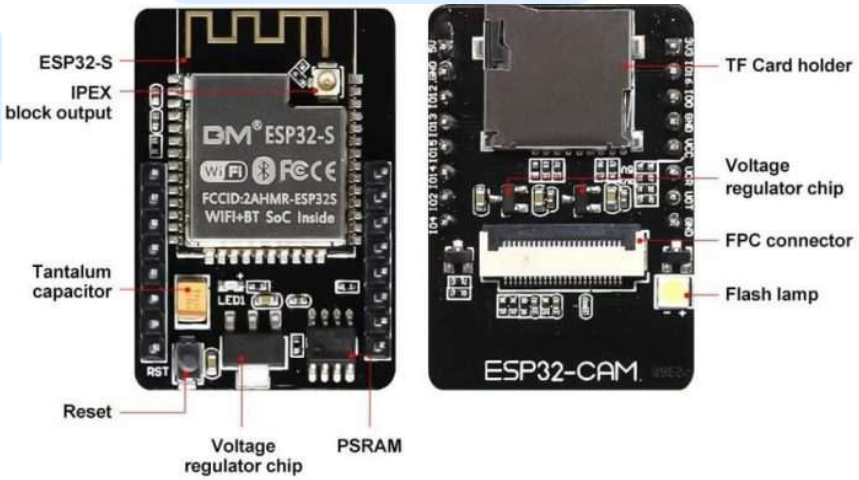


Hình 2.2.1. ESP 32

- Các tính năng của ESP32 bao gồm:
  + CPU: Bộ vi xử lý Xtensa lõi kép (hoặc lõi đơn) 32-bit LX6, hoạt động ở tần số 240 MHz (160 MHz cho ESP32-S0WD và ESP32-U4WDH) và hoạt động ở tối đa 600 MIPS (200 MIPS với ESP32-S0WD/ESP32-U4WDH)
  + Bộ đồng xử lý (co-processor) công suất cực thấp (Ultra low power, viết tắt: ULP) hỗ trợ việc đọc ADC và các ngoại vi khi bộ xử lý chính (main processor) vào chế độ deep sleep.
  + Hệ thống xung nhịp: CPU Clock, RTC Clock và Audio PLL Clock
  + Bộ nhớ nội:
    - 448 KB bộ nhớ ROM cho việc booting và các tính năng lõi
    - 520 KB bộ nhớ SRAM trên chip cho dữ liệu và tập lệnh
  + Kết nối không dây:
    - Wi-Fi: 802.11 b/g/n
    - Bluetooth: v4.2 BR/EDR và BLE (chia sẻ sóng vô tuyến với Wi-Fi)
  + 34 GPIO pad vật lý với các ngoại vi:
    - ADC SAR 12 bit, 18 kênh
    - DAC 2 × 8-bit
    - 10 cảm biến cảm ứng (touch sensor) (GPIO cảm ứng điện dung)
    - 3 SPI (SPI, HSPI và VSPI) hoạt động ở cả 2 chế độ master/slave. Module ESP32 hỗ trợ 4 ngoại vi SPI với SPI0 và SPI1 kết nối đến bộ nhớ flash của ESP32 còn SPI2 và SPI3 tương ứng với HSPI và VSPI. Các GPIO đều có thể được dùng để triển khai HSPI và VSPI.
    - 2 I^2S
    - 2 I^2C, hoạt động được ở cả chế độ master và slave, với chế độ Standard mode (100 Kbit/s) và Fast mode (400 Kbit/s). Hỗ trợ 2 chế độ định địa chỉ là 7-bit và 10-bit. Các GPIO đều có thể được

dùng để triển khai I²C.
- 3 UART (UART0, UART1, UART2) với tốc độ lên đến 5 Mbps
- SD/SDIO/CE-ATA/MMC/eMMC host controller
- SDIO/SPI slave controller
- Ethernet MAC interface cho DMA và IEEE 1588 Precision Time Protocol (tạm dịch: Giao thức thời gian chính xác IEEE 1588)
- CAN bus 2.0
- Bộ điều khiển hồng ngoại từ xa (TX/RX, lên đến 8 kênh)
- PWM cho điều khiển động cơ
- LED PWM (lên đến 16 kênh)
- Cảm biến hiệu ứng Hall
- Bộ tiền khuếch đại analog công suất cực thấp (Ultra low power analog pre-amplifier)

+ Bảo mật:
  - Hỗ trợ tất cả các tính năng bảo mật chuẩn IEEE 802.11, bao gồm WFA, WPA/WPA2 và WAPI.
  - Secure boot (tạm dịch: khởi động an toàn)
  - Mã hóa flash
  - 1024-bit OTP, lên đến 768-bit cho khách hàng
  - Tăng tốc mã hóa phần cứng: AES, SHA-2, RSA, elliptic curve cryptography (ECC, tạm dịch: mật mã đường cong ellip), random number generator (viết tắt: RNG, tạm dịch: trình tạo số ngẫu nhiên)
+ Quản lý năng lượng:
  - Hỗ trợ 5 chế độ hoạt động với mức tiêu thụ năng lượng khác nhau: Active, Modem¬-sleep, Light-¬sleep, Deep-¬sleep và Hibernation
  - Bộ ổn áp nội với điện áp rơi thấp (internal low-dropout regulator)
  - Individual power domain (tạm dịch: Miền nguồn riêng) cho RTC
  - Trở lại hoạt động từ ngắt GPIO, timer, đo ADC, ngắt với cảm ứng điện dung



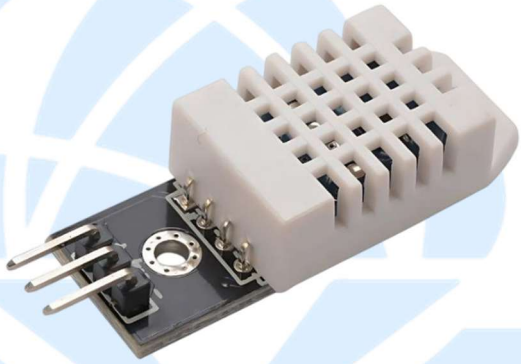
Hình 2.2.2. Chi tiết phần cứng ESP32

- Lưu trữ tích hợp:
  - ESP32 bao gồm bộ nhớ tích hợp sau:

Kích thước	**Bộ Nhớ**
SRAM	520 KiBit
Bộ nhớ flash	448 KiBit
NVRAM	16 KiBit

3. Cảm biến nhiệt độ DHT22

- Cảm biến DHT22 (AM2302) là thiết bị đo nhiệt độ và độ ẩm môi trường, trả về tín hiệu số (digital) thông qua giao tiếp 1 dây (1-Wire). Nó tích hợp cảm biến độ ẩm điện dung và cảm biến nhiệt độ NTC (nhiệt điện trở), kết hợp với vi mạch xử lý để chuyển đổi dữ liệu analog thành digital, phù hợp cho các ứng dụng giám sát thời gian thực.



Hình 2.3.1. DHT22

- Các tính năng của cảm biến DHT22:
  + Đo đồng thời nhiệt độ và độ ẩm: Cung cấp dữ liệu độ ẩm tương đối (0-100% RH) và nhiệt độ (-40 đến 80°C) trong một thiết bị.
  + Tín hiệu số: Truyền dữ liệu qua giao thức 1-Wire, đảm bảo độ ổn định và chính xác, không cần ADC bên ngoài.
  + Phạm vi đo rộng: Độ ẩm 0-100% RH (±2% sai số), nhiệt độ -40 đến 80°C (±0.5°C sai số).
  + Tần số lấy mẫu: Lên đến 0.5Hz (2 giây/lần), phù hợp cho giám sát môi trường thời gian thực.

  + Nguồn điện linh hoạt: Hoạt động ở điện áp 3.3-5.5V, tiêu thụ dòng thấp (tối đa 2.5mA khi truyền dữ liệu).
  + Tích hợp điện trở pull-up: Trong mô-đun, giảm phức tạp khi kết nối với vi điều khiển.
  + Kích thước nhỏ gọn: Dễ dàng tích hợp vào các dự án IoT, thiết bị cầm tay, hoặc hệ thống giám sát môi trường.
  + Hỗ trợ nhiều nền tảng: Tương thích với Arduino, ESP8266, ESP32, Raspberry Pi thông qua các thư viện lập trình (như DHT_sensor_library).
  + Độ bền cao: Hoạt động ổn định trong môi trường hơi nước thuần, phù hợp cho các ứng dụng như trạm thời tiết, nhà kính, kho lưu trữ.
- Phương thức hoạt động:
  + Cấu trúc: DHT22 gồm cảm biến độ ẩm (đo độ ẩm tương đối), cảm biến nhiệt độ NTC, và vi mạch xử lý tín hiệu. Dữ liệu được truyền qua chân DATA sử dụng giao thức 1-Wire.
  + Giao tiếp:
    ▪ Vi điều khiển (như Arduino, ESP8266, ESP32) gửi tín hiệu bắt đầu (pull-down chân DATA trong ~1ms, sau đó pull-up).
    ▪ DHT22 đáp ứng bằng cách gửi 40 bit dữ liệu (5 byte): 2 byte cho độ ẩm (phần nguyên và thập phân), 2 byte cho nhiệt độ (phần nguyên và thập phân), 1 byte kiểm tra lỗi (checksum).
    ▪ Tần số lấy mẫu tối đa 0.5Hz (2 giây/lần).
  + Nguồn và đo lường: Hoạt động ở điện áp 3.3-5.5V, dòng tối đa 2.5mA khi truyền dữ liệu. Đo độ ẩm từ 0-100% RH (±2% sai số), nhiệt độ từ -40 đến 80°C (±0.5°C sai số).

4. Cảm biến chất lượng không khí MQ-135

- Cảm biến MQ-135 là một cảm biến khí dùng để phát hiện và đo nồng độ các loại khí trong không khí như NH3 (amoniac), NOx, ancol, benzen, khói, CO, CO2, v.v. Nó hoạt động dựa trên sự thay đổi điện trở của vật liệu cảm biến (Thiếc Dioxide - SnO2) khi tiếp xúc với các khí, chuyển đổi thành tín hiệu analog hoặc digital để giám sát chất lượng không khí trong môi trường, thường được dùng trong nhà, văn phòng, hoặc công nghiệp.

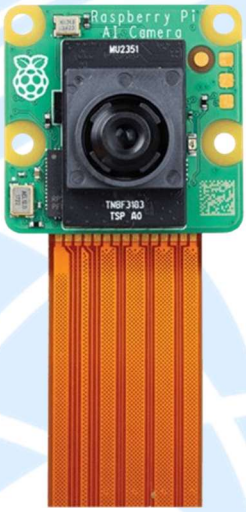
Hình 2.4.1. MQ-135

- Tính năng:
  + Phát hiện đa khí: NH3, NOx, ancol, benzen, khói, CO, CO2 (10-1000 ppm).
  + Độ nhạy điều chỉnh: Biến trở tích hợp để tinh chỉnh ngưỡng phát hiện.
  + Tín hiệu đầu ra: Analog (điện áp tăng theo nồng độ khí) và digital (ngưỡng logic TTL).
  + Nguồn điện: 5V DC, công suất tiêu thụ thấp (heater 5V ± 0.1V AC/DC).
  + Ứng dụng đa dạng: Giám sát không khí trong nhà, văn phòng, hệ thống cảnh báo khí độc, IoT, nhà thông minh.
  + Tích hợp dễ dàng: Hỗ trợ thư viện Arduino, tương thích với nhiều vi điều khiển.
  + Kích thước nhỏ gọn: Phù hợp cho các dự án điện tử, robot, và tự động hóa.
- Phương thức hoạt động:
  + Cấu trúc: MQ-135 gồm phần tử cảm biến SnO2, lõi gia nhiệt (heater), và mạch xử lý tín hiệu. SnO2 thay đổi điện trở khi tiếp xúc với khí.
  + Nguyên lý: Khi khí (như CO, NH3, benzen) tiếp xúc với SnO2 được làm nóng, electron được giải phóng, làm giảm điện trở. Cảm biến đo sự thay đổi này và xuất tín hiệu analog (A0) hoặc digital (D0) dựa trên ngưỡng điều chỉnh bằng biến trở.
  + Giao tiếp: Kết nối với vi điều khiển (Arduino, ESP32) qua chân A0 (analog) hoặc D0 (digital). Giá trị analog (0-1023) được chuyển thành nồng độ khí (ppm) thông qua thư viện (như MQ135 library). Điện áp nguồn: 5V DC, điện trở heater: 33Ω ± 5%.

+ Hiệu chuẩn: Cần hiệu chuẩn trong không khí sạch để xác định R0 (điện trở cơ sở). Tỷ số RS/R0 được dùng để tính nồng độ khí dựa trên biểu đồ đặc tính của nhà sản xuất.

5. Camera

- Camera Raspberry Pi 5 là các mô-đun camera được thiết kế để kết nối với bo mạch Raspberry Pi 5 thông qua cổng MIPI CSI, cho phép chụp ảnh, quay video và xử lý hình ảnh. Các mô-đun này, do Raspberry Pi Foundation sản xuất hoặc bên thứ ba cung cấp, hỗ trợ nhiều ứng dụng như giám sát, AI, chụp ảnh time-lapse, hoặc robot.



Hình 2.5.1. Camera Raspberry Pi 5

- Tính năng:
  + Độ phân giải cao: Lên đến 12MP (Camera Module 3, HQ Camera), hỗ trợ video 1080p@30fps hoặc 4K với một số mô-đun.
  + Tùy chọn ống kính: Camera Module 3 có phiên bản chuẩn/rộng, HQ Camera hỗ trợ ngàm C/CS/M12, cho phép thay ống kính linh hoạt.
  + Chụp ảnh chuyên dụng: Hỗ trợ HDR, chụp ảnh ban đêm (NoIR), chụp chuyển động nhanh (Global Shutter), và time-lapse.
  + Hỗ trợ AI: AI Camera (IMX500) cung cấp xử lý thị giác máy tính thời gian thực với độ trễ thấp.
  + Đầu ra linh hoạt: Lưu ảnh (JPEG, PNG, BMP), video (H.264), hoặc dữ liệu thô (DNG) cho xử lý hậu kỳ.
  + Kích thước nhỏ gọn: Phù hợp cho dự án IoT, robot, camera giám sát, hoặc drone.
  + Tương thích đa nền tảng: Hỗ trợ Raspberry Pi OS, Ubuntu, và các thư viện như OpenCV, Halcon.

- Phương thức hoạt động:
  + Cấu trúc: Bao gồm cảm biến hình ảnh (như IMX708, IMX477, IMX500), ống kính (cố định hoặc thay đổi), và cáp ribbon kết nối với cổng MIPI CSI trên Raspberry Pi 5. Raspberry Pi 5 có 2 cổng MIPI (4 lane), hỗ trợ 2 camera đồng thời.
  + Giao tiếp:
    - Kết nối vật lý: Cáp ribbon 22 chân (khác với 15 chân của Raspberry Pi 4) hoặc cáp chuyển đổi được cắm vào cổng MIPI, với mặt tiếp xúc vàng hướng về cổng Ethernet.
    - Phần mềm: Raspberry Pi OS (Bookworm) tự động nhận diện camera chính hãng qua camera_auto_detect=1. Camera bên thứ ba cần chỉnh sửa /boot/firmware/config.txt (tắt auto-detect, thêm dòng dtoverlay như dtoverlay=imx219).
    - Lệnh: Sử dụng libcamera-hello, rpicam-still, hoặc rpicam-vid để kiểm tra, chụp ảnh, quay video. Hỗ trợ Python với thư viện picamera2.
  + Xử lý: Raspberry Pi 5 (CPU Cortex-A76 2.4GHz, GPU VideoCore VII) xử lý hình ảnh/video hiệu quả, hỗ trợ AI (YOLO11) và độ phân giải cao (lên đến 12MP).

V. TRIỂN KHAI

- Phần triển khai hệ thống sẽ được làm trong 3 giai đoạn là:
  + Nhận diện các vật thể có thể gây ra cháy nổ và lửa bằng model AI YOLOv11 thông qua các dataset sưu tầm được.
  + Thiết kế một nền tảng (website/ ứng dụng di động) để có thể nhận thông báo và ghi nhận những gì mà camera đang giám sát.
  + Thiết kế hệ thống phần cứng, lắp ráp thiết bị để chạy thực tế

1. Train model nhận diện

- Ở giai đoạn này, để camera có thể nhận diện được các vật thể và nguy cơ gây ra những vấn đề tai nạn cháy nổ, thì chương trình sẽ cần phải nhận diện các hình ảnh liên quan qua một bộ dataset được làm và sưu tầm qua Roboflow.
- Dataset với 4 class:
  + Fire: class chứa các data liên quan đến lửa.
  + Smoke: class chứa các data liên quan đến khói .
  + Smoking: class chứa các data liên quan đến việc người dùng hút thuốc và điếu thuốc.
  + Zippo: Class chứa các data liên quan đến hột quẹt, đồ đánh lửa…

- Các dữ liệu hình ảnh trong dataset sẽ từ 400-500 ảnh cho phù hợp tiêu chuẩn của YOLOv11 và các dataset đó sẽ liên quan đến những nguy cơ cháy nổ trong tất cả các tình huống xảy ra trong thực tế.

Hình ảnh 3.1.1. Ảnh được label với class là fire

- Để quá trình train model AI diễn ra một cách thuận lợi, thì Jupyter Notebook chính là công cụ được sử dụng để train trong môi trường python.

```
[2]: !pip install ultralytics

Requirement already satisfied: ultralytics in /usr/local/lib/python3.12/dist-packages (8.3.128)
Requirement already satisfied: numpy>=1.23.0 in /usr/local/lib/python3.12/dist-packages (from ultralytics) (2.2.5)
Requirement already satisfied: matplotlib>=3.3.0 in /usr/local/lib/python3.12/dist-packages (from ultralytics) (3.10.1)
Requirement already satisfied: opencv-python>=4.6.0 in /usr/local/lib/python3.12/dist-packages (from ultralytics) (4.11.0.86)
Requirement already satisfied: pillow>=7.1.2 in /usr/local/lib/python3.12/dist-packages (from ultralytics) (11.2.1)
Requirement already satisfied: pyyaml>=5.3.1 in /usr/lib/python3/dist-packages (from ultralytics) (6.0.1)
Requirement already satisfied: requests>=2.23.0 in /usr/local/lib/python3.12/dist-packages (from ultralytics) (2.32.3)
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.12/dist-packages (from ultralytics) (1.15.2)
Requirement already satisfied: torch>=1.8.0 in /usr/local/lib/python3.12/dist-packages (from ultralytics) (2.7.0)
Requirement already satisfied: torchvision>=0.9.0 in /usr/local/lib/python3.12/dist-packages (from ultralytics) (0.22.0)
Requirement already satisfied: tqdm>=4.64.0 in /usr/local/lib/python3.12/dist-packages (from ultralytics) (4.67.1)
Requirement already satisfied: psutil in /usr/local/lib/python3.12/dist-packages (from ultralytics) (7.0.0)
Requirement already satisfied: py-cpuinfo in /usr/local/lib/python3.12/dist-packages (from ultralytics) (9.0.0)
Requirement already satisfied: pandas>=1.1.4 in /usr/local/lib/python3.12/dist-packages (from ultralytics) (2.2.3)
Requirement already satisfied: seaborn>=0.11.0 in /usr/local/lib/python3.12/dist-packages (from ultralytics) (0.13.2)
Requirement already satisfied: ultralytics-thop>=2.0.0 in /usr/local/lib/python3.12/dist-packages (from ultralytics) (2.0.14)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=3.3.0->ultralytics) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=3.3.0->ultralytics) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=3.3.0->ultralytics) (4.57.0)
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=3.3.0->ultralytics) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=3.3.0->ultralytics) (25.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=3.3.0->ultralytics) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.12/dist-packages (from matplotlib>=3.3.0->ultralytics) (2.9.0.po
st0)
```

Hình ảnh 3.1.2. Cài đặt ultralytics

- Để YOLOv11 có thể chạy được thì ultralytics cần phải được cài đặt trên môi trường của các máy thực hiện huấn luyện.

```
[5]: !wget https://github.com/ultralytics/assets/releases/download/v8.3.0/yolo11n.pt

--2025-05-19 02:40:52--  https://github.com/ultralytics/assets/releases/download/v8.3.0/yolo11n.pt
Resolving github.com (github.com)... 20.205.243.166
Connecting to github.com (github.com)|20.205.243.166|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: https://objects.githubusercontent.com/github-production-release-asset-2e65be/521807533/34b70ade-b6eb-4179-a60f-d6494307226b?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=releaseassetproduction%2F20250518%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20250518T194052Z&X-Amz-Expires=300&X-Amz-Signature=389cac6ab1816b52869aa832a791a7197b34eb9407fae27387e380bd72553581&X-Amz-SignedHeaders=host&response-content-disposition=attachment%3B%20filename%3Dyolo11n.pt&response-content-type=application%2Foctet-stream [following]
--2025-05-19 02:40:52--  https://objects.githubusercontent.com/github-production-release-asset-2e65be/521807533/34b70ade-b6eb-4179-a60f-d6494307226b?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=releaseassetproduction%2F20250518%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20250518T194052Z&X-Amz-Expires=300&X-Amz-Signature=389cac6ab1816b52869aa832a791a7197b34eb9407fae27387e380bd72553581&X-Amz-SignedHeaders=host&response-content-disposition=attachment%3B%20filename%3Dyolo11n.pt&response-content-type=application%2Foctet-stream
Resolving objects.githubusercontent.com (objects.githubusercontent.com)... 185.199.110.133, 185.199.111.133, 185.199.108.133, ...
Connecting to objects.githubusercontent.com (objects.githubusercontent.com)|185.199.110.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5613764 (5.4M) [application/octet-stream]
Saving to: 'yolo11n.pt'

yolo11n.pt          100%[===================>]   5.35M  7.19MB/s    in 0.7s

2025-05-19 02:40:55 (7.19 MB/s) - 'yolo11n.pt' saved [5613764/5613764]
```

Hình ảnh 3.1.3. Tải file cài đặt của Yolo11n

- YOLO11n là một phiên bản YOLO được sử dụng khác phổ biến đối với YOLOv11 tuy nhiên để việc huấn luyện mô hình trở nên chuẩn xác hơn thì nên phải sử dụng YOLO11 với những phiên bản cao hơn một chút, ở đây nhóm có sử dụng một phiên bản có nét nhỉn hơn là YOLO11m.

```
import gdown

file_id = "1dBRBd2fTh-duxVhl16N6isNrmyzeIr9H"
url = f"https://drive.google.com/uc?id=1dBRBd2fTh-duxVhl16N6isNrmyzeIr9H"
gdown.download(url, "detect-data8.zip", quiet=False)

Downloading...
From: https://drive.google.com/uc?id=1dBRBd2fTh-duxVhl16N6isNrmyzeIr9H
To: /home/kiethuynh/detect_dataset4/detect/detect-data8.zip
100%|██████████| 15.4M/15.4M [00:00<00:00, 26.3MB/s]
'detect-data8.zip'
```

Hình ảnh 3.1.4. Tải file dataset từ google drive về máy

- Jupyter Notebook là một ứng dụng không có giao diện, để có thể import hoặc tải các file dataset thì cần sẽ phải sử dụng các câu lệnh trên với file id là token của link được gán vào và url là địa chỉ của đường link đó.

```
import zipfile
with zipfile.ZipFile("detect-data8.zip", 'r') as zip_ref:
    zip_ref.extractall("detect_dataset8")
```

Hình ảnh 3.1.5. Giải nén file vừa download

- Để có thể thực hiện train cần phải giải nén file để dễ truy cập hơn vào các mục

runs	3 seconds ago
test	5 minutes ago
train	1 second ago
valid	7 seconds ago
data.yaml	4 minutes ago
README.dataset.txt	5 minutes ago
README.roboflow.txt	8 minutes ago

Hình ảnh 3.1.6. File dataset

- Một File dataset sẽ gồm có các thành phần gồm có test, train, valid là nơi chứa các dữ liệu hình ảnh và chúng đóng một vai trò quan trọng khi quá trình huấn luyện mô hình được diễn ra.
- Tại file Yaml là một file đường dẫn cũng như chứa các thông tin từ số lượng class và các class cụ thể.

```
train: /home/kiethuynh/detect_dataset8/train/images
val: /home/kiethuynh/detect_dataset8/valid/images
test: /home/kiethuynh/detect_dataset8/test/images

nc: 4
names: ['fire', 'smoke', 'smoking', 'zippo']
```

Hình ảnh 3.1.7. File Yaml

- File Yaml là một file đóng vai trò rất quan trọng trong quá trình huấn luyện mô hình qua YOLO với nội dung file gồm số lượng class cần nhận diện và 3 đường dẫn đến các file hình ảnh của dataset.

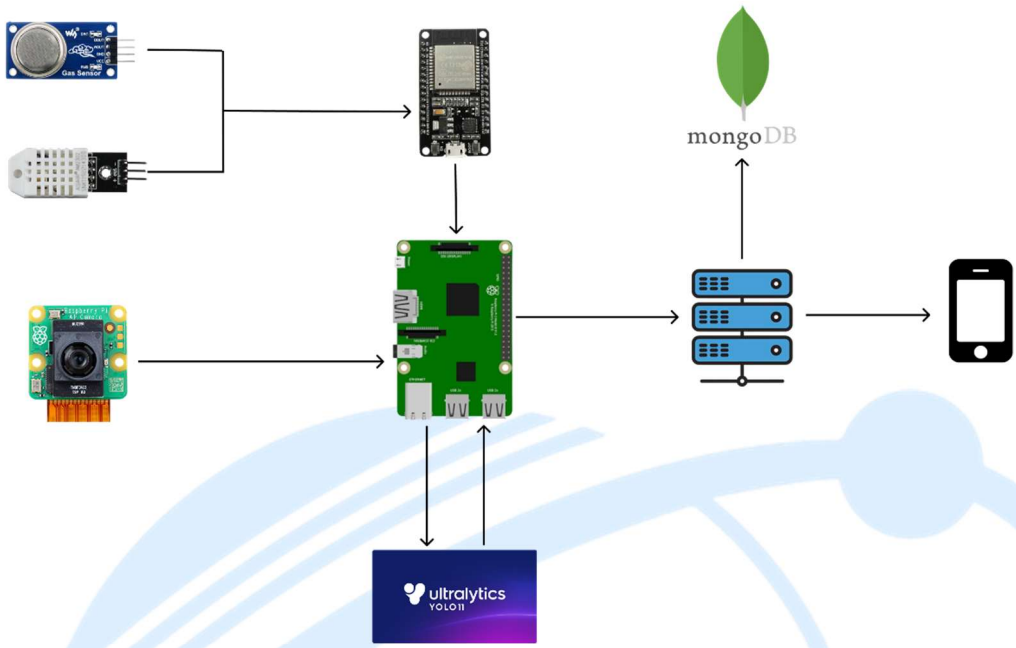
```
[ ]: !yolo task=detect mode=train model=yolo11m.pt data=data.yaml epochs=300 imgsz=640 batch=-1
```

Hình ảnh 3.1.8. Train model

- Ở đây, mô hình sẽ tiến hành được huấn luyện với model là YOLO11m và số epochs để mô hình được học là 300.

2. Thiết kế hệ thống phần cứng

- Để mô hình có thể hoạt động được cũng như là nhận diện được các dữ liệu đã được học, thì hệ thống phần cứng sẽ được thiết kế như sau:



Hình ảnh 3.2.1. Mô hình reference

- **Các phần cứng:**
  + Các phần cứng cảm biến được sủ dụng là cảm biến chất lượng không khí và cảm biến nhiệt độ, với mục đích bổ trợ cho quá trình phòng chống các nguy cơ cháy nổ trong nhà máy.
  + Camera được sử dụng với mục đích nhận diện các vật thể gây nguy cơ cháy nổ đồng thời còn sử dụng để giám sát tình hình trong nhà máy, chúng cũng đóng vai trò là một camera giám sát an ninh cho nhà máy.
  + ESP32 với vai trò điều khiển 2 thiết bị cảm biến, ESP32 cũng là một phần cứng đóng vai trò là cầu nối giữa thiết bị cảm biến và máy tính nhúng
  + Raspberry Pi 5 được sử dụng để làm máy tính trung tâm để giao tiếp các thiết bị và ứng dụng di động.
    - Khi camera bắt được những nguy cơ và tác nhân gây ra cháy nổ thì ngay lập tức Raspberry Pi 5 sẽ gửi về server các thông tin hình ảnh của các tác nhân đó.
    - Sau khi nhận diện được các tác nhân thì server sẽ thông tin cho ứng dụng di động rằng các tác nhân đó đang gây nguy cơ đến cháy nổ, hoặc tác nhân đang xảy ra để người dùng nhanh chóng xử lí.
    - Các hình ảnh, dữ liệu của các tác nhân sẽ được lưu trữ trên database của MongoDB
    - Ý tưởng có thể trích xuất được hình ảnh thứ 3-5 của hiện trường để của dữ liệu lưu trữ và gửi đến app di động để hiện thị trên ứng dụng, nhằm xem lại nguyên nhân xảy ra sự việc.

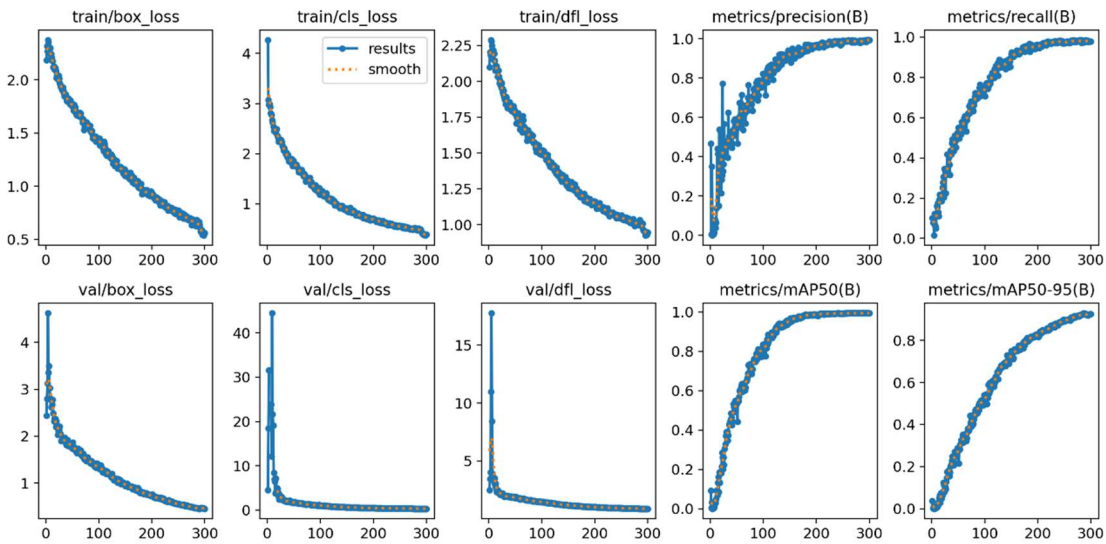
- **Phần mềm:**
  + Ứng dụng di động được xây dựng gồm các chức năng có thể giám sát được an ninh của một nhà máy.
  + Ứng dụng có chức năng cảnh báo khi có tác nhân xảy đến, hiển thị một thông báo khẩn cấp cho người dùng nếu phát hiện điều bất thường.
  + Ứng dụng còn có thể trích xuất được hình ảnh để người dùng có thể xem được nguyên nhân sự việc xảy ra (hiện thị ngày, tháng năm, giờ xảy ra)
  + Ứng dụng còn có chức năng SignUp/SignIn và nhập mã số của camera, điều này nhằm đảm bảo thông tin bảo mật của người dùng và thiết bị khi sử dụng, tránh xâm nhập bất hợp pháp.
  + MongoDB là nơi lưu trữ các thông tin hình ảnh tác nhân, đồng thời nó sẽ lưu trữ thông tin của người dùng đăng kí để sử dụng camera, đồng thời MongoDB còn lưu trữ các thông tin mà cảm biến của hệ thống thu được.

- **Workflow:**
  + Khi gặp một tầm nhìn khuất nhưng các cảm biến phát hiện sự bất thường: Cảm biến phát hiện thất thường → ESP32 gửi tín hiệu về Raspberry Pi 5 → Thông tin thông số thu được gửi về Server → MongoDB lưu lại → Ứng dụng di dộng nhận dữ liệu realtime.
  + Khi bắt được nguy cơ và tác nhân xảy ra: Camera luôn hoạt động → Raspberry Pi 5 ghi nhận hình ảnh → YOLOv11 nhận diện các tác nhân → kết quả bất thường được gửi về Server → MongoDB lưu lại → Ứng dụng di động nhận dữ liệu realtime.

VI. KẾT QUẢ VÀ ĐÁNH GIÁ

1. Kết quả huấn luyện mô hình

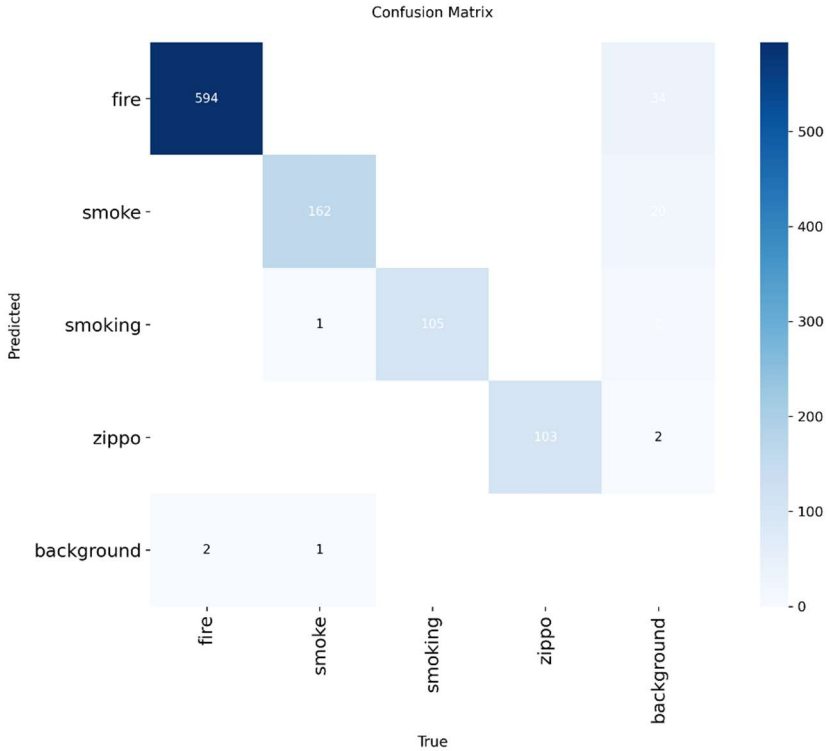
- Trong quá trình thực hiện huấn luyện cho mô hình có thể học được những vật thể cũng như các tác nhân cần được nhận diện, mô hình sẽ cho ra các kết quả sau:



Hình ảnh 4.1.1. Training log chart

- Với hình ảnh trên là một biểu đồ kết quả của mô hình huấn luyên qua 300 epochs.
- Về Hiệu quả huấn luyện (Training Effectiveness):
  + train/box_loss, train/cls_loss, train/dfl_loss đều giảm mượt về cuối, cho thấy rằng mô hình không bị stuck ở local minimum và học vô cùng ổn định
  + Loss của validation giảm tương đồng với training. Không có hiện tượng loss training giảm nhưng loss validation tăng (dấu hiệu overfitting).
- Hiệu suất mô hình (Model Performance):
  + Precision & Recall đều cao (~1.0) => Mô hình phát hiện đúng và gần như đầy đủ các đối tượng.
  + mAP50 ≈ 1.0, mAP50-95 ≈ 0.9 => Mô hình không chỉ dự đoán đúng mà còn rất chính xác về vị trí (localization).
- Độ hội tụ (Convergence:
  + Đường loss (train + val) đều mượt, không có spikes lớn.
  + Các metric (precision, recall, mAP) không dao động mạnh về cuối → mô hình đã hội tụ tốt.
- Khả năng tổng quát hóa (Generalization):
  + Loss validation giảm song song với training => mô hình học tốt trên tập chưa thấy trước.
  + Có sự cải thiện liên tục trong precision, recall, mAP của validation, không chỉ training.

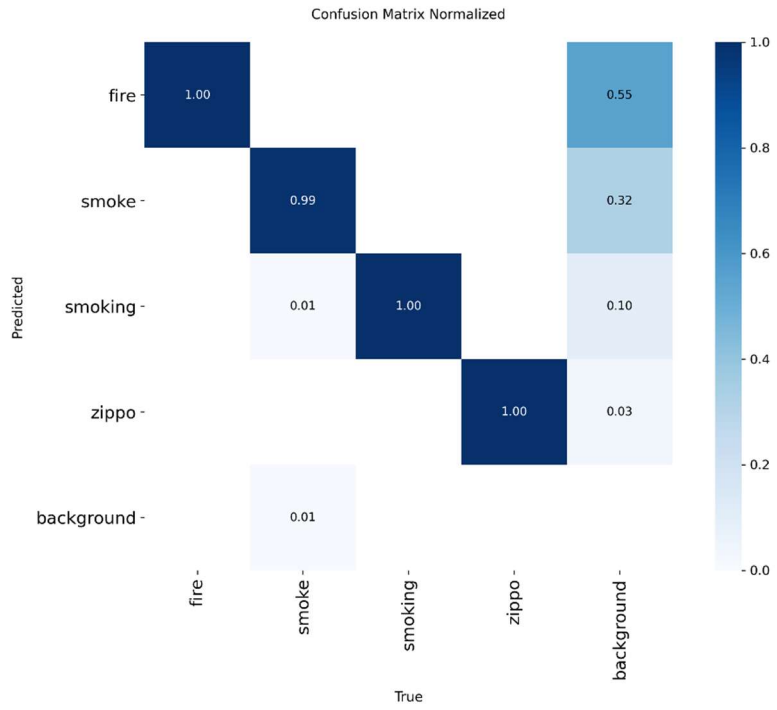
➔ Mô hình hoạt động ổn định, hoạt động cực kỳ tốt, học hiệu quả và tổng quát hóa tốt, hoàn toàn đạt chuẩn để triển khai. Biểu đồ càng đi xuống thì mô hình được huấn luyện sẽ càng ổn định.



Hình ảnh 4.1.2. Confusion Matrix

- Confusion Matrix dùng để đánh giá chất lượng của mô hình phân loại (classification), cho thấy mô hình dự đoán các nhãn như thế nào so với thực tế.
- Phân tích chi tiết từng lớp kết quả cho ra:
  + Fire: 594 ảnh dự đoán đúng là fire, 34 ảnh fire bị dự đoán nhầm là background. Kết quả trên đã cho một kết luận chính xác rất cao, nhưng có 1 ít mẫu nhầm là background.
  + Smoke: 162 ảnh được dự đoán đúng là smoke, 6 ảnh bị nhầm sang background, 1 ảnh bị nhầm sang smoke. Kết quả trên được đánh giá tốt, nhưng vẫn có vài tường hợp nhầm lẫn nhẹ.
  + Zippo: 103 ảnh được dự đoán đúng là zippo, 2 ảnh bị nhầm là background.. Kết quả trên được đánh giá gần như hoàn hảo và tỉ lệ nhầm cực thấp.
  + Background: Có 2 mẫu background bị nhầm là fire, 1 mẫu bị nhầm là smoke. Điều này cho thấy mô hình có một số ít nhầm lẫn background là vật thể, nhưng tỷ lệ rất thấp.

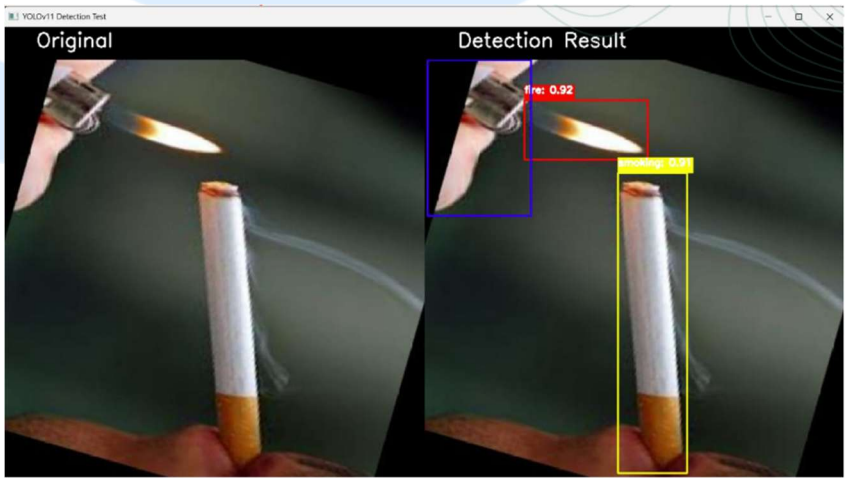
➔ Đây là một mô hình phân loại rất tốt, với tỷ lệ nhầm lẫn thấp và hiệu suất cao, mô hình tổng quát hóa tốt, cần cải thiện một ít phần nhầm lẫn.



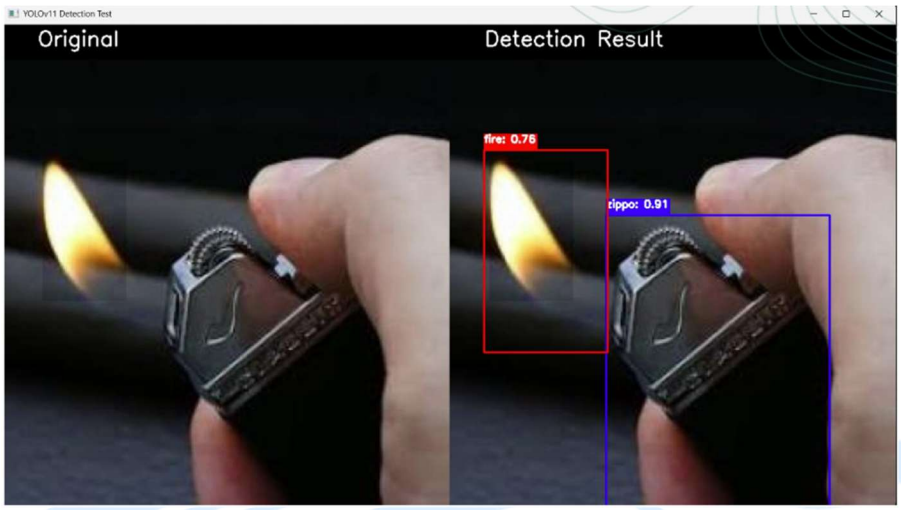
Hình ảnh 4.1.3. Confusion Matrix Normalized

- Confusion Matrix Normalized là ma trận nhầm lẫn đã được chuẩn hóa, được dùng trong đánh giá mô hình phân loại đa lớp (multi-class classification).
- Tổng hợp các kết quả cho ra:
  + Fire: dự đoán rất tốt cho ảnh fire thật (1.00), nhưng lại hay nhầm background là fire (0.55) → cần cải thiện.
  + Smoke : dự đoán tốt (0.99), nhưng bị nhầm 32% background thành smoke → cũng là dấu hiệu cần điều chỉnh.
  + Smoking: cực kỳ chính xác (1.00), gần như không nhầm.
  + Zippo: cũng rất chính xác (1.00).

Kết quả:



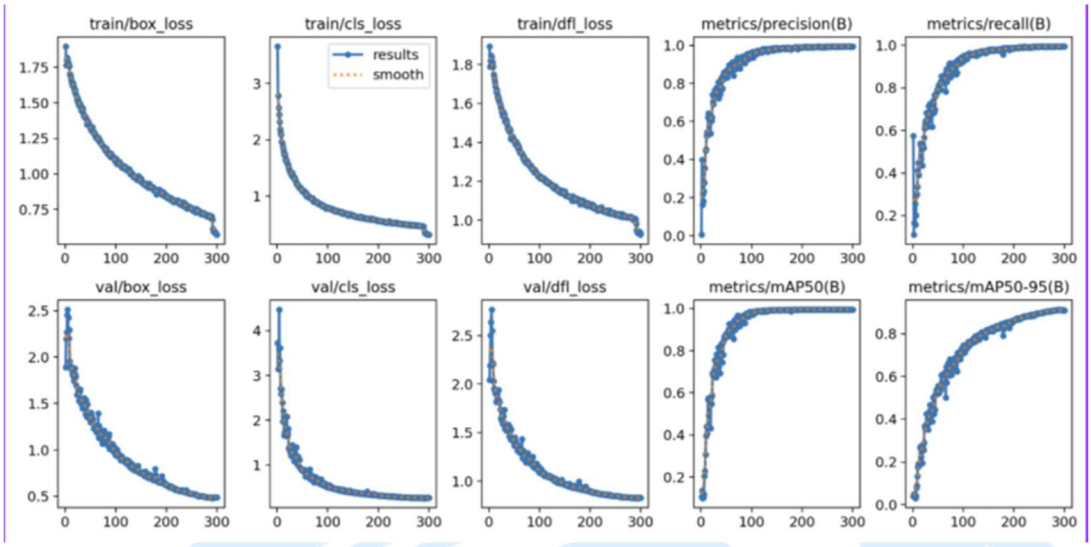
Hình ảnh 4.1.4. Kết quả demo 1



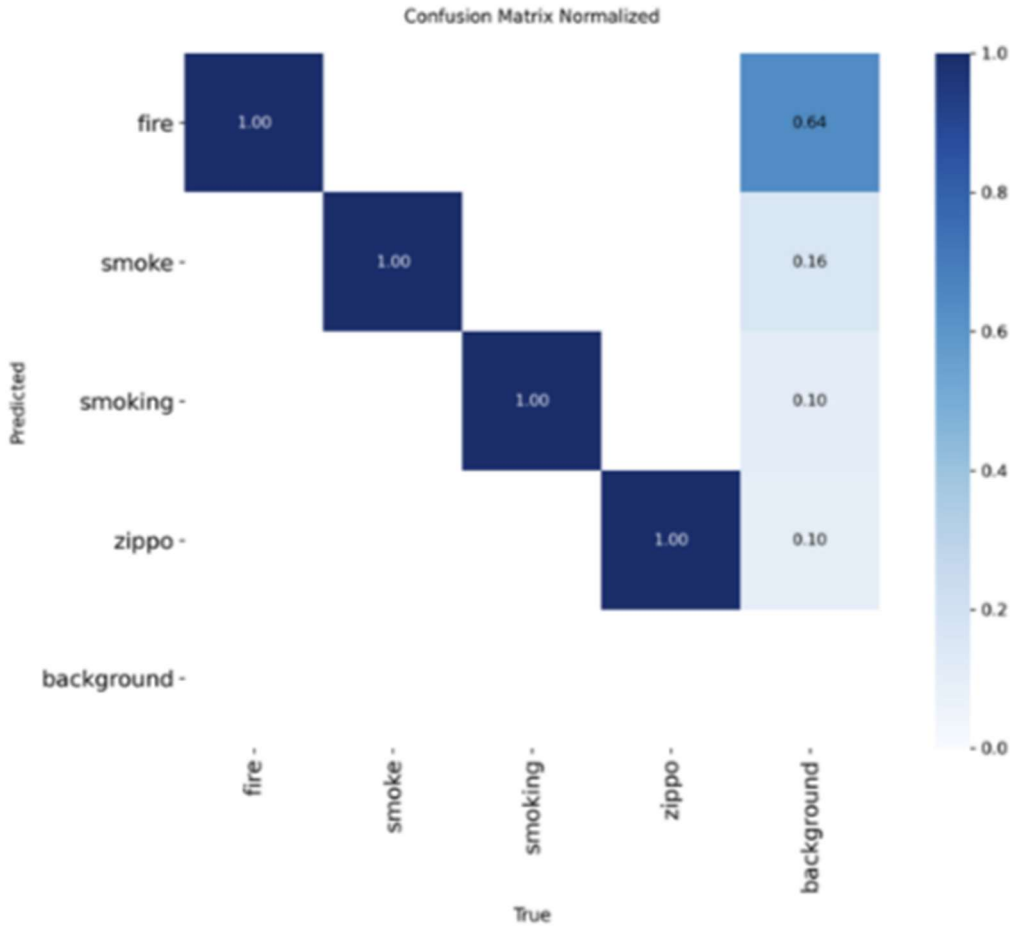
Hình ảnh 4.1.5. Kết quả demo 2

2. SO SÁNH 2 PHIÊN BẢN YOLO

- Ở đồ án này, nhóm có sử dụng 2 phiên bản YOLO khác nhau là YOLO11m và YOLOv8m, lý do nhóm chọn phiên bản YOLOv8m còn lại là do khi chạy thử demo cho model trên máy tính thật thì số FPS của model trên YOLOv8m nhỉn hơn rất nhiều so với YOLO11m, ngoài ra nó còn có sự khác nhau về kết quả. Chính vì vậy, ta sẽ có sự so sánh của 2 phiên bản này khác gì nhau.
- Tại YOLOv8m:
  + Sẽ có số lượng tham số nhỏ hơn so với YOLO11m, tuy nhiên sau khi train thì kết quả mà biểu đồ cho thấy, YOLOv8m sẽ hoạt động vẫn rất ổn định, và quá trình train không bị chập chờn cũng như bị lỗi trong quá trình cho model học.
  + Về tốc độ train thì YOLOv8m sẽ train lâu hơn khá nhiều so với YOLO11m, với tốc độ train model mà nhóm thu được với lượng dataset như nhau thì YOLO11m train nhanh hơn so với YOLOv8m tầm 30 phút đến 1 tiếng.
  + YOLOv8m khi chạy demo detect trên một máy tính thật mượt hơn so với YOLO11m có thể là do một số lý do:
    - Số tham số ít hơn so với YOLO11m
    - Số FLOPS ít hơn so với YOLO11m
    - Tải model vào GPU ít hơn so với YOLO11m
  + Dưới đây là một số kết quả thu được khi train bằng YOLOv8m (có sự update về dataset nhiều hơn vì YOLO11m nhóm đã detect ra kết quả mong đợi, nhưng khi test trên YOLOv8m thì kết quả bị giảm đi kha khá nên nhóm đã update dataset lại và xóa đi kết quả cũ thì cho ra được kết quả mới với số lượng dataset nhiều hơn)



Hình ảnh 4.2.1. Training log chart của YOLOv8m



Hình ảnh 4.2.2. Confusion Matrix Normalized By YOLOv8m

TÀI LIỆU THAM KHẢO

[1] M. Hòa và C. Tuấn, "Cháy lớn bãi xe ở Thủ Đức," 6 6 2022. [Trực tuyến]. Available: https://web.archive.org/web/20220606093444/https://tuoitre.vn/chay-lon-bai-xe-o-thu-duc-20220606140416842.htm. [Đã truy cập 10 5 2025].

[2] ultralytics, "Ultralytics YOLO11," 26 2 2025. [Trực tuyến]. Available: https://docs.ultralytics.com/vi/models/yolo11/#can-yolo11-be-deployed-on-edge-devices. [Đã truy cập 10 5 2025].

[3] N. R. Sulake, "A Comprehensive Guide to YOLOv11 Object Detection," 1 5 2025. [Trực tuyến]. Available: https://www.analyticsvidhya.com/blog/2024/10/yolov11-object-detection/. [Đã truy cập 11 5 2025].

[4] V. Hoàng, "Tìm hiểu về YOLO trong bài toán real-time object detection," 16 11 2019. [Trực tuyến]. Available: https://viblo.asia/p/tim-hieu-ve-yolo-trong-bai-toan-real-time-object-detection-yMnKMdvr57P. [Đã truy cập 11 5 2025].

[5] ultralytics, "Roboflow Tích hợp," 30 3 2025. [Trực tuyến]. Available: https://docs.ultralytics.com/vi/integrations/roboflow/#how-can-i-manage-and-analyze-my-yolo11-dataset-using-roboflow. [Đã truy cập 11 5 2025].

[6] Pum, "Jupyter Notebook là gì? Hướng dẫn cài đặt và sử dụng Jupyter Notebook," 29 7 2022. [Trực tuyến]. Available: https://200lab.io/blog/jupyter-notebook-la-gi?srsltid=AfmBOorQD2OqBBm1Tm4hp2RO6BrKdTZcnarpo0Eby-ZJgKDLlMDkaQOj. [Đã truy cập 11 5 2025].

[7] FPT.AI, "Computer Vision – Thị giác máy tính là gì? Ứng dụng & Ví dụ," 13 12 2024. [Trực tuyến]. Available: https://fpt.ai/vi/bai-viet/computer-vision-va-nhung-ung-dung-trong-nganh-tai-chinh-ngan-hang-bao-hiem/. [Đã truy cập 12 5 2025].

[8] H. Nguyễn, "Deep Learning là gì? Tổng quan về Deep Learning từ A-Z," 21 4 2022. [Trực tuyến]. Available: https://vietnix.vn/deep-learning-la-gi/. [Đã truy cập 12 5 2025].