



**TRƯỜNG ĐẠI HỌC BÁCH KHOA  
KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO**

**PBL5 – ĐỒ ÁN KỸ THUẬT MÁY TÍNH**

**HỆ THỐNG BÁO CHÁY TỰ ĐỘNG**

**GIẢNG VIÊN HƯỚNG DẪN: TS. HUỖNH HỮU HÙNG**

**DOANH NGHIỆP HƯỚNG DẪN: ENCLAVE**

**STT NHÓM: 11**

<b>HỌ VÀ TÊN SINH VIÊN</b>	<b>LỚP SINH HOẠT</b>	<b>LỚP HỌC PHẦN</b>
LÊ HỒNG ANH	20T1	20.Nh10A
LÊ ĐỨC HUY	20T1	20.Nh10A
HUỖNH BÁ THUẬN	20T1	20.Nh10A
HỒ NGỌC TÙNG	20T1	20.Nh10A

**ĐÀ NẴNG, 06/2023**

## TÓM TẮT ĐỒ ÁN

Hiện nay, vấn đề bảo đảm an toàn cháy nổ trong các tòa nhà là một vấn đề được chú trọng. Trong số các giải pháp để đảm bảo an toàn cháy nổ, hệ thống báo cháy đóng một vai trò quan trọng. Nếu không có hệ thống báo cháy hoặc hệ thống này không hoạt động đúng cách, rủi ro cho con người và tài sản sẽ rất lớn khi có cháy xảy ra.

Để giảm thiểu rủi ro và thuận tiện cho công tác chữa cháy và cứu người, nhóm của tôi đã thực hiện đồ án về hệ thống báo cháy tự động. Hệ thống này sử dụng loa để tự động báo cháy và gửi thông báo qua ứng dụng Android. Với ứng dụng này, người dùng có thể quan sát được đám cháy thông qua camera trên điện thoại.

Hiện tại, với hệ thống khả thi tối thiểu mà chúng tôi đã thực hiện, đã có thể xác định tình huống xảy ra cháy với độ chính xác và độ ổn định tương đối tốt. Các mô hình phối hợp với nhau cũng hoạt động ổn định.

Tuy nhiên, nếu có thời gian và nguồn lực lớn hơn, nhóm của chúng tôi có thể cải thiện lại mô hình để nó hoạt động tốt hơn và đáng tin cậy hơn trong việc bảo đảm an toàn cho tòa nhà khi xảy ra cháy nổ. Chúng tôi hy vọng rằng với việc cải thiện và phát triển tiếp hệ thống này, nó sẽ đóng góp vào việc giảm thiểu rủi ro cho con người và tài sản khi xảy ra cháy nổ.

Trong quá trình thực hiện đồ án, nhóm chúng tôi còn thiếu sót một số kiến thức và kinh nghiệm nên chúng tôi không thể tránh khỏi những sai sót trong quá trình xây dựng và triển khai đồ án. Vì vậy, nhóm chúng tôi mong nhận được sự thông cảm, chỉ bảo và giúp đỡ của thầy, cô và các bạn.

Chúng tôi xin cảm ơn các thầy, cô Khoa Công Nghệ Thông tin, trường Đại Học Bách Khoa - Đại Học Đà Nẵng vì đã cung cấp cho chúng tôi nhiều kiến thức mới mẻ. Đặc biệt nhóm chúng tôi xin chân thành cảm ơn thầy Huỳnh Hữu Hưng và anh Vương Nhật Quang (Doanh nghiệp Enclave) đã hỗ trợ và hướng dẫn chúng tôi những kiến thức cần thiết cũng như tinh thần học tập độc lập, tìm hiểu và sáng tạo nhằm phục vụ cho đồ án môn học này.

Nội dung quyền báo cáo gồm:

- Phần 1: Giới thiệu – Giới thiệu sơ qua về đề tài.
- Phần 2: Các vấn đề và giải pháp – Nêu ra các vấn đề cần giải quyết và các giải pháp để thực hiện các vấn đề trên.
- Phần 3: Kết quả - Kết quả thực hiện.
- Phần 4: Kết luận – Kết luận về mức độ hoàn thành, ưu điểm và nhược điểm của hệ thống và hướng phát triển trong tương lai.
- Phần 5: Tài liệu tham khảo – Tài liệu tham khảo đã sử dụng.

## DỰ KIẾN KẾ HOẠCH VÀ PHÂN CÔNG NHIỆM VỤ

STT	Nội dung công việc		Phân công	Thời gian
1	Phần cứng	Tìm hiểu phần cứng, linh kiện, cách lắp, cài đặt linh kiện và thành phần dự án	Huy, Thuận	Tuần 1
2		Kết nối các linh kiện phần cứng		Tuần 2 - 3
3		Cài đặt phần cứng		
4	Phần mềm	Xây dựng Cơ sở dữ liệu	Anh, Tùng	Tuần 4
5		Xây dựng Server		Tuần 5
6		Truyền nhận client-server - Gửi hình ảnh từ Rasp (WebCam) lên server - Nhận kết quả từ server về Raspi	Huy	Tuần 6 - 7
7		Thiết kế giao diện app mobile trên figma	Thuận	Tuần 6
8		Lập trình app mobile - Lập trình giao diện - Lấy dữ liệu từ server và hiển thị lên giao diện	Thuận, Tùng	Tuần 4 - 7
9	AI	Thu thập dữ liệu, xây dựng dataset	Anh, Huy	Tuần 7
10		Huấn luyện và thử nghiệm mô hình		Tuần 8 - 9
11	Thử nghiệm sản phẩm		Cả nhóm	Tuần 10
12	Viết báo cáo			Tuần 11
13	Làm slide			Tuần 11

## Mục lục

<b>1. GIỚI THIỆU .....</b>	<b>7</b>
<b>2. CÁC VẤN ĐỀ VÀ GIẢI PHÁP.....</b>	<b>7</b>
2.1. Giải pháp về phần cứng và truyền thông.....	8
2.1.1. Linh kiện sử dụng.....	8
2.1.2. Sơ đồ kết nối của hệ thống.....	9
2.1.3. Giải pháp truyền thông.....	10
2.2. Mô hình CNN.....	11
2.2.1. Giới thiệu về mô hình CNN.....	11
2.2.2. Kiến trúc của mô hình CNN .....	11
2.3 Mô hình YOLOv8.....	13
2.3.1 Giới thiệu mô hình YOLOv8.....	13
2.3.2 Kiến trúc của mô hình YOLOv8 .....	14
2.3.3 Cách YOLOv8 hoạt động .....	15
2.3.4 Hàm tính IoU (INTERSECTION OVER UNION) .....	16
2.3.5 Hàm mất mát (Loss Function).....	17
2.3.6 Tối ưu hóa (Optimizer).....	17
2.4 Firebase .....	18
2.5 Ứng dụng di động.....	19
<b>3. KẾT QUẢ .....</b>	<b>23</b>
3.1. Mô hình nhận diện.....	23
3.2. Lập trình ứng dụng.....	28
3.3. Thử nghiệm hệ thống .....	30
3.4. Đánh giá .....	31
<b>4. KẾT LUẬN.....</b>	<b>32</b>
4.1. Kết luận từ đồ án .....	32
4.2. Tồn tại những điểm yếu .....	32
4.3. Hướng phát triển .....	32
<b>5. TÀI LIỆU THAM KHẢO.....</b>	<b>33</b>

## MỤC LỤC HÌNH ẢNH

Hình 1. Sơ đồ tổng quan hệ thống .....	7
Hình 2. Raspberry Pi 2 Model B và các chân cắm.....	8
Hình 3. Webcam .....	8
Hình 4. Buzzer .....	9
Hình 5. Sơ đồ kết nối của hệ thống .....	9
Hình 6. Ví dụ về Convolution Layer .....	11
Hình 7. Ví dụ về Activation function .....	12
Hình 8. Ví dụ về Fully connected layer.....	13
Hình 9. Kiến trúc của mô hình YOLOv8 .....	14
Hình 10. Ví dụ về cách hoạt động của YOLO.....	16
Hình 11. Sơ đồ quá trình đăng nhập .....	20
Hình 12. Sơ đồ quá trình đăng ký.....	21
Hình 13. Sơ đồ quá trình xử lý cháy.....	22
Hình 14. Sơ đồ quá trình gửi vụ cháy đến Firebase .....	22
Hình 15. Cấu trúc tập dữ liệu .....	23
Hình 16. Kích thước tập dữ liệu .....	24
Hình 17. Đồ thị biểu diễn các hàm loss YOLOv8 trên tập train .....	25
Hình 18. Đồ thị biểu diễn các hàm loss YOLOv8 trên tập val.....	25
Hình 19. Đồ thị các chỉ số Precision và Recall .....	26
Hình 20. Đồ thị chỉ số mAP và mAP50-95 .....	26
Hình 21. Nhận diện trên tập thử nghiệm .....	28
Hình 22. Giao diện đăng nhập/đăng ký ứng dụng.....	28
Hình 23. Giao diện chính và giao diện khi mở một vụ cháy chi tiết.....	29
Hình 24. Thông báo nếu có cháy trong hiện trường.....	29
Hình 25. Giao diện tổng hợp các vụ cháy đã xảy ra trong quá khứ .....	30
Hình 26. Phát hiện cháy.....	31

## 1. GIỚI THIỆU

Hệ thống báo cháy là một phần thiết yếu của hệ thống phòng cháy chữa cháy hiện nay. Tuy nhiên, các hệ thống báo cháy truyền thống hiện chỉ có thể phát hiện được sự cố cháy thông thường trong một thời gian dài và đa số thông qua khói của ngọn lửa mà chưa thể cảnh báo trong một thời gian ngắn. Vì vậy, chúng tôi muốn phát triển hệ thống báo cháy tích hợp với công nghệ nhận diện hình ảnh.

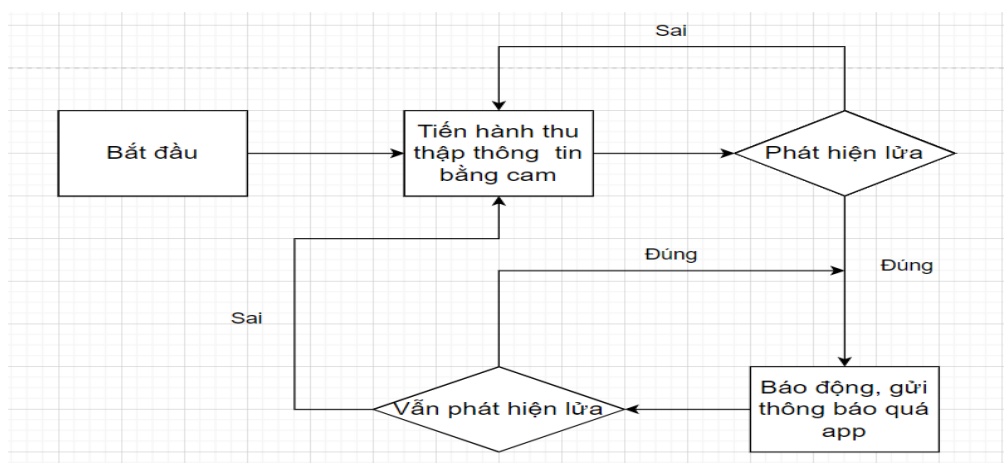
## 2. CÁC VẤN ĐỀ VÀ GIẢI PHÁP

Hệ thống sẽ tự động phát hiện khi có hoả hoạn xảy ra, sau đó gửi thông báo và hình ảnh tới cơ sở dữ liệu Firebase. Bằng cách sử dụng ứng dụng Android, người dùng có thể truy cập Firebase để xem thông báo và hình ảnh liên quan đến sự cố.

Các vấn đề cần giải quyết ở đây sẽ là:

- Cần có các thiết bị phần cứng để thu thập dữ liệu môi trường, hình ảnh.
- Cơ sở để lưu trữ, nhận các dữ liệu từ thiết bị phần cứng.
- Phát hiện cháy, báo cháy.
- Có ứng dụng để có thể theo dõi và nhận được tín hiệu.
- Hệ thống chạy trong thời gian thực.

Vì vậy, để giải quyết các vấn đề trên, chúng tôi cần đặt ra các giải pháp cho hệ thống của mình, với các chức năng và ứng dụng phù hợp.




Hình 1. Sơ đồ tổng quan hệ thống

## 2.1. Giải pháp về phần cứng và truyền thông

### 2.1.1. Linh kiện sử dụng

#### a. Raspberry Pi 2 Model B

CPU quad-core A53 (ARMv8) 64-bit, RAM 4GB. Kết nối Gigabit. Ethernet, Wifi. Bộ nhớ Micro-SD. 40-pin GPIO header. Camera Serial Interface (CSI). Kích thước 82mm x 56mm x 19.5mm, 50g.



Function	Physical Pins				Function
	BCM	pin#	pin#	BCM	
3.3 Volts		1	2		5 Volts
GPIO/SDA1 (I2C)	2	3	4		5 Volts
GPIO/SCL1 (I2C)	3	5	6		GND
GPIO/GCLK	4	7	8	14	TX UART/GPIO
GND		9	10	15	RX UART/GPIO
GPIO	17	11	12	18	GPIO
GPIO	27	13	14		GND
GPIO	22	15	16	23	GPIO
3.3 Volts		17	18	24	GPIO
MOSI (SPI)	10	19	20		GND
MISO(SPI)	9	21	22	25	GPIO
SCLK(SPI)	11	23	24	8	CEO_N (SPI)
GND		25	26	7	CE1_N (SPI)
RESERVED		27	28		RESERVED
GPIO	5	29	30		GND
GPIO	6	31	32	12	GPIO
GPIO	13	33	34		GND
GPIO	19	35	36	16	GPIO
GPIO	26	37	38	20	GPIO
GND		39	40	21	GPIO

Hình 2. Raspberry Pi 2 Model B và các chân cắm

#### b. Webcam Knup KP-CW100



Hình 3. Webcam

#### c. Buzzer

Còi Buzzer 5VDC có tuổi thọ cao, hiệu suất ổn định, chất lượng tốt, được sản xuất nhỏ gọn phù hợp thiết kế với các mạch còi buzzer nhỏ gọn, mạch báo động.



- Sử dụng nguồn ở ngưỡng 3.5V -> 5V.
- Biên độ âm thanh > 80dB.
- Nhiệt độ hoạt động: -20 °C đến 70 °C.

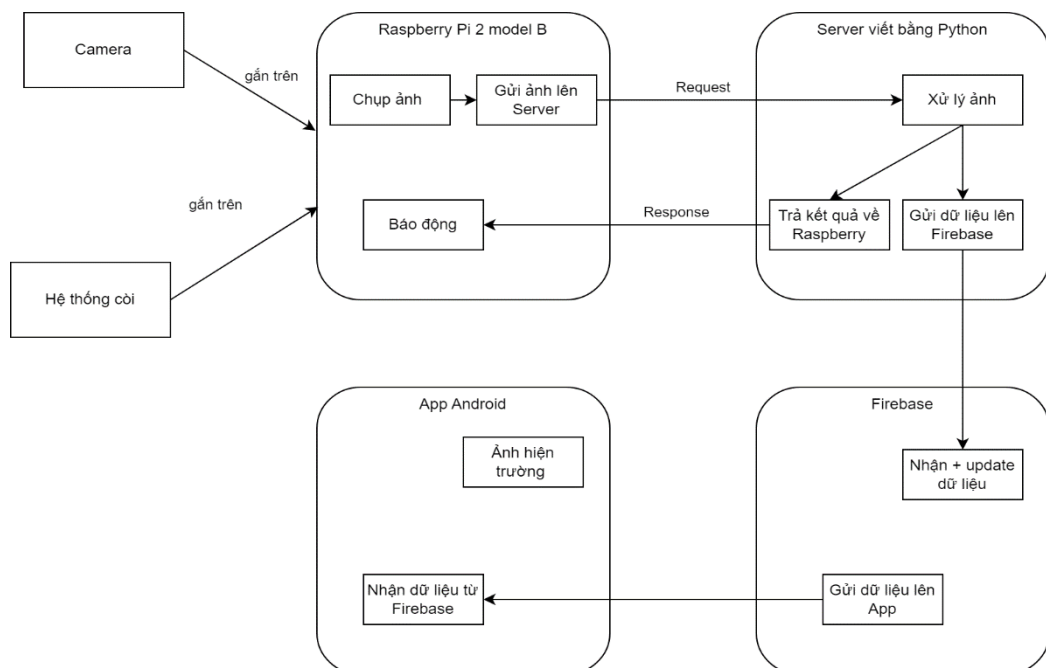


Hình 4. Buzzer

Kết nối toàn bộ phần cứng với Raspberry Pi

- Webcam:
  - Kết nối với Serial USB trên Raspberry pi.
- Buzzer:
  - Nối chân GND vào chân GND của Raspberry pi.
  - Nối chân DATA vào chân GPIO15 của Raspberry pi.

### 2.1.2. Sơ đồ kết nối của hệ thống



Hình 5. Sơ đồ kết nối của hệ thống

### 2.1.3. Giải pháp truyền thông

#### a. Restful API

RESTful API là một tiêu chuẩn được sử dụng trong việc thiết kế API cho các phần mềm, ứng dụng và dịch vụ web để tạo sự thuận tiện cho việc quản lý các resource. Các tài nguyên hệ thống như tệp văn bản, ảnh, video, âm thanh hay dữ liệu di động là mục tiêu mà nó hướng tới, bao gồm các trạng thái tài nguyên được định dạng và truyền tải qua HTTP. Có thể nói, RESTful API không phải là một loại công nghệ. Nó chỉ là một phương thức tạo ra API và nguyên lý tổ chức nhất định.

API (Application Programming Interface): Là tập hợp các quy tắc và cơ chế mà một ứng dụng hay một thành phần nào đó có khả năng tương tác với một ứng dụng với thành phần khác. API sẽ trả về những kiểu dữ liệu phổ biến như JSON hoặc XML mà ứng dụng của bạn cần sử dụng đến.

REST (Representational State Transfer): Là một dạng chuyển đổi cấu trúc hay kiểu kiến trúc để viết API. Nó có khả năng tạo ra sự tương tác giữa các máy với nhau thông qua phương thức HTTP đơn giản. Chức năng của REST là quy định sử dụng các phương thức HTTP và định dạng URL cho ứng dụng web.

#### b. Flask Framework

Flask là một web frameworks, nó thuộc loại micro-framework được xây dựng bằng ngôn ngữ lập trình Python. Flask cho phép bạn xây dựng các ứng dụng web từ đơn giản tới phức tạp. Nó có thể xây dựng các api nhỏ, ứng dụng web chẳng hạn như các trang web, blog, trang wiki hoặc một website dựa theo thời gian hay thậm chí là một trang web thương mại. Flask cung cấp cho bạn công cụ, các thư viện và các công nghệ hỗ trợ bạn làm những công việc trên.

Flask là một micro-framework. Điều này có nghĩa Flask là một môi trường độc lập, ít sử dụng các thư viện khác bên ngoài. Do vậy, Flask có ưu điểm là nhẹ, có rất ít lỗi do ít bị phụ thuộc cũng như dễ dàng phát hiện và xử lý các lỗi bảo mật. Còn Flask RESTful là một phần mở rộng cho Flask mà thêm hỗ trợ cho việc nhanh chóng xây dựng các API REST.

## 2.2. Mô hình CNN

### 2.2.1. Giới thiệu về mô hình CNN

Mô hình CNN (Convolutional Neural Networks) là một trong những mô hình deep learning phổ biến nhất và có ảnh hưởng nhiều nhất trong cộng đồng Computer Vision. CNN được dùng trong nhiều bài toán như nhận dạng ảnh, phân tích video, ảnh MRI, hoặc cho bài các bài của lĩnh vực xử lý ngôn ngữ tự nhiên.

Về kỹ thuật, mô hình CNN để training và kiểm tra, mỗi hình ảnh đầu vào sẽ chuyển nó qua 1 loạt các lớp tích chập với các bộ lọc (Kernels), tổng hợp lại các lớp được kết nối đầy đủ (Full Connected) và áp dụng hàm Softmax để phân loại đối tượng có giá trị xác suất giữa 0 và 1.

### 2.2.2. Kiến trúc của mô hình CNN

#### a. Lớp tích chập - Convolution Layer:

Lớp tích chập (Convolution Layer) là một phần quan trọng trong kiến trúc của mô hình CNN. Tầng tích chập sử dụng các bộ lọc để thực hiện phép tích chập khi đưa chúng đi qua đầu vào theo các chiều của nó. Các siêu tham số của các bộ lọc này bao gồm kích thước bộ lọc và độ trượt (stride). Kết quả đầu ra được gọi là feature map hay activation map. Nó là 1 phép toán có 2 đầu vào như ma trận hình ảnh và 1 bộ lọc hoặc hạt nhân.

Ví dụ: Xét 1 ma trận kích thước 5 x 5 có giá trị pixel là 0 và 1. Ma trận bộ lọc 3 x 3 như dưới đây:

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

5 x 5 – Image Matrix

\*

1	0	1
0	1	0
1	0	1

3 x 3 – Filter Matrix

Hình 6. Ví dụ về Convolution Layer

Các siêu tham số của bộ lọc trong tầng tích chập (Convolution Layer) bao gồm kích thước bộ lọc (F) và độ trượt (stride - S). Độ trượt ký hiệu số pixel mà cửa sổ sẽ di chuyển sau mỗi lần thực hiện phép tính.

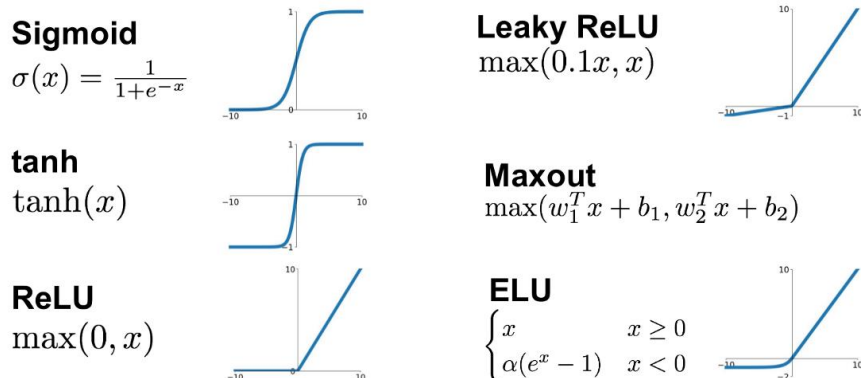
Ngoài ra còn có siêu tham số Zero-padding là tên gọi của quá trình thêm P số không vào các biên của đầu vào. Giá trị này có thể được lựa chọn thủ công hoặc một cách tự động bằng một trong ba những phương pháp: Valid, Same và Full.

b. Lớp gộp - Pooling Layer:

Lớp pooling sẽ giảm bớt số lượng tham số khi hình ảnh quá lớn. Không gian pooling còn được gọi là lấy mẫu con hoặc lấy mẫu xuống làm giảm kích thước của mỗi map nhưng vẫn giữ lại thông tin quan trọng. Cụ thể, max pooling và average pooling là những dạng pooling đặc biệt, mà tương ứng là trong đó giá trị lớn nhất và giá trị trung bình được lấy ra.

c. Hàm kích hoạt - Activation Function:

Mô hình CNN sử dụng các hàm kích hoạt để tạo ra các đặc trưng phi tuyến tính từ các giá trị đầu vào. Các hàm kích hoạt như ReLU giúp mô hình có khả năng học được các đặc trưng phi tuyến tính, cũng như giảm thiểu sự triệt tiêu đặc trưng khi áp dụng các phép tích chập. Ngoài ReLU, các hàm kích hoạt khác được sử dụng trong CNN bao gồm Sigmoid, Tanh, v.... Tùy thuộc vào vấn đề cụ thể và cấu trúc của mô hình, một hàm kích hoạt phù hợp sẽ được chọn để đạt được hiệu quả tốt nhất.



Hình 7. Ví dụ về Activation function

d. Flatten Layer:

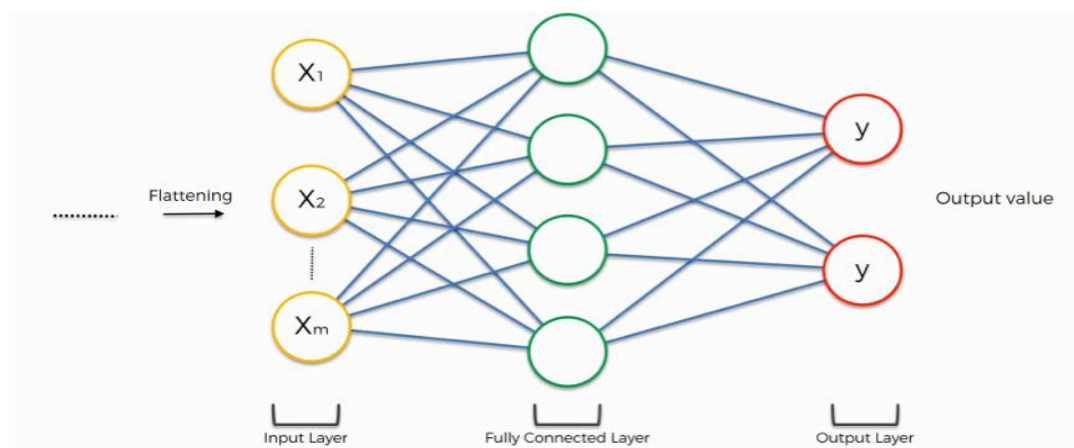
Lớp làm phẳng (Flatten Layer) được sử dụng để làm phẳng đầu vào. Ví dụ, nếu lớp làm phẳng được áp dụng cho lớp có đầu vào có kích thước (batch\_size, 2,2), thì kích thước đầu ra của lớp sẽ là (batch\_size, 4).

e. Lớp kết nối đầy đủ - Fully connected layer:

Tầng kết nối đầy đủ (Fully Connected Layer) nhận đầu vào là các dữ liệu đã được làm phẳng, mà mỗi đầu vào đó được kết nối đến tất cả neuron. Trong mô hình mạng CNNs, các tầng kết nối đầy đủ thường được tìm thấy ở cuối mạng và được dùng để tối ưu hóa mục tiêu của mạng ví dụ như độ chính xác của lớp.

Sau khi ảnh được truyền qua nhiều lớp tích chập và lớp gộp thì model đã học được tương đối các đặc điểm của ảnh (ví dụ mắt, mũi, khung mặt, ...) thì tensor của output của layer cuối cùng, kích thước  $H*W*D$ , sẽ được chuyển về 1 vector kích thước  $(H*W*D)$ .

Sau đó ta dùng các fully connected layer để kết hợp các đặc điểm của ảnh để ra được output của model.



Hình 8. Ví dụ về Fully connected layer

## 2.3 Mô hình YOLOv8

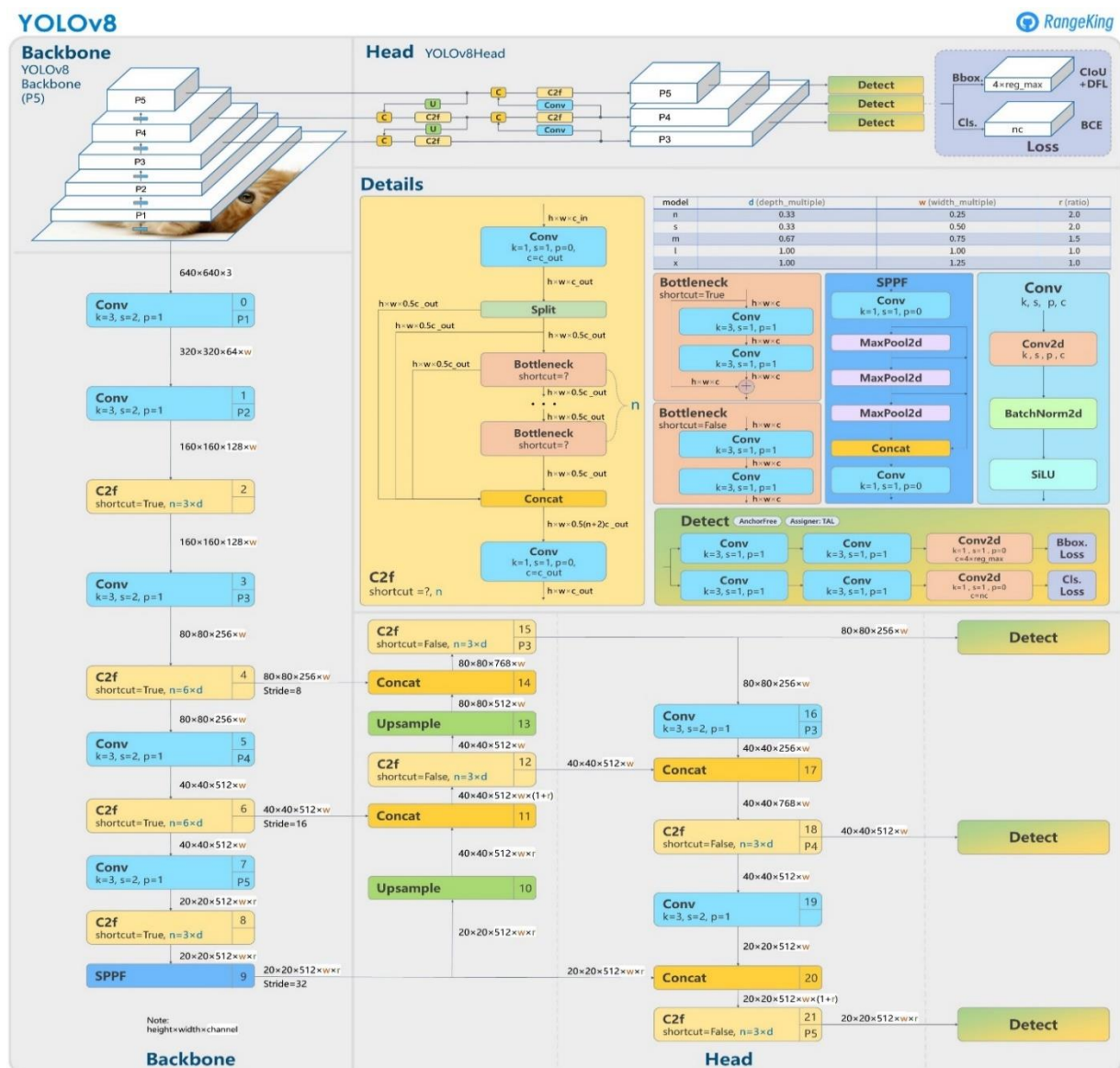
### 2.3.1 Giới thiệu mô hình YOLOv8

YOLOv8 là một mô hình tiên tiến, hàng đầu thế giới (SOTA), xây dựng trên thành

công của các phiên bản trước đó và giới thiệu các tính năng và cải tiến mới để tăng cường hiệu suất, linh hoạt và hiệu quả. YOLOv8 hỗ trợ đầy đủ các nhiệm vụ AI thị giác, bao gồm phát hiện, phân đoạn, ước lượng tư thế, theo dõi và phân loại. Sự đa dạng này cho phép người dùng tận dụng khả năng của YOLO trong nhiều ứng dụng và lĩnh vực khác nhau.

### 2.3.2 Kiến trúc của mô hình YOLOv8

Chưa có một bài báo chính thức nào về kiến trúc mô hình Yolov8. Tài liệu dưới đây được nghiên cứu dựa trên mã nguồn và được tác giả đồng ý. Cấu trúc nhận diện vật thể của Yolov8 có 2 phần: Backbone và Head.



Hình 9. Kiến trúc của mô hình Yolov8

Backbone là một loạt các lớp tích chập trích xuất các tính năng có liên quan từ hình ảnh đầu vào. Lớp SPPF và các lớp tích chập tiếp theo xử lý các đối tượng ở nhiều tỷ lệ khác nhau, trong khi các lớp Upsample tăng độ phân giải của bản đồ đối tượng. Mô-đun C2f kết hợp các tính năng cấp cao với thông tin theo ngữ cảnh để cải thiện độ chính xác của phát hiện. Cuối cùng, Head sử dụng một tập hợp các lớp tích chập và tuyến tính để ánh xạ các đặc trưng nhiều chiều tới các hộp giới hạn đầu ra và các lớp đối tượng. Đối với chú thích sơ đồ, các hình chữ nhật đại diện cho các lớp, với các nhãn mô tả loại lớp (Conv, Upsample, v.v.) và mọi tham số có liên quan (kích thước hạt nhân, số lượng kênh, v.v.). Các mũi tên biểu thị luồng dữ liệu giữa các lớp, với hướng của mũi tên biểu thị luồng dữ liệu từ lớp này sang lớp tiếp theo.

### 2.3.3 Cách Yolov8 hoạt động

Thuật toán phát hiện đối tượng YOLO (You Only Look Once) hoạt động theo nhiều giai đoạn. Hoạt động của YOLO có thể được mô tả theo các bước sau:

Hình ảnh đầu vào: Thuật toán lấy một hình ảnh làm đầu vào.

Chia lưới: Hình ảnh được chia thành một lưới các ô. Mỗi ô chịu trách nhiệm phát hiện các đối tượng có mặt trong đó. Kích thước của lưới phụ thuộc vào kích thước đầu vào của hình ảnh và kích thước của bản đồ tính năng tích chập cuối cùng của mạng.

Trích xuất tính năng: Mỗi ô được truyền qua CNN để trích xuất các tính năng. CNN này được đào tạo trước trên một tập dữ liệu hình ảnh lớn để tìm hiểu các tính năng có thể được sử dụng để phát hiện đối tượng.

Điểm đối tượng: Mỗi ô dự đoán một điểm đối tượng, cho biết liệu một đối tượng có hiện diện trong ô đó hay không. Điều này được thực hiện bằng cách sử dụng hàm hồi quy logistic, dự đoán xác suất xuất hiện của một đối tượng trong ô.

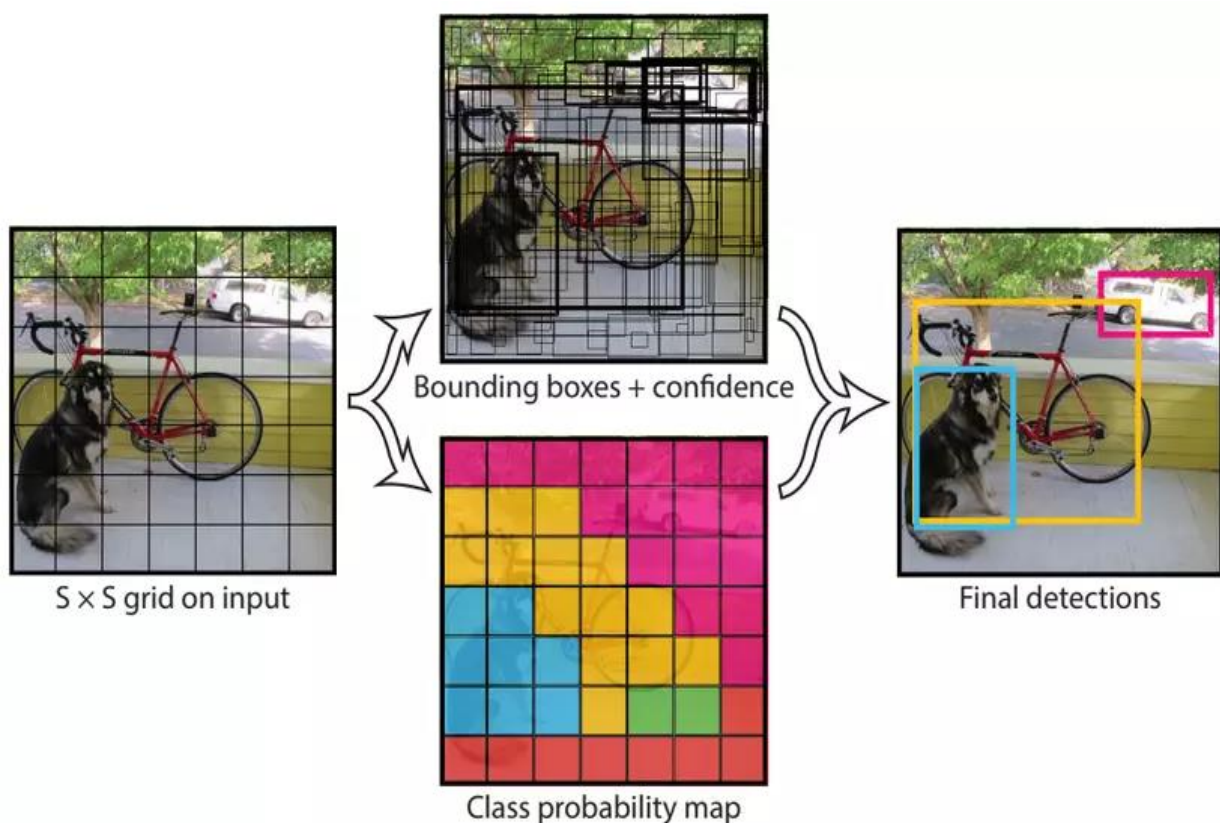
Xác suất lớp: Đối với mỗi ô dự đoán một đối tượng, YOLO dự đoán lớp của đối tượng và xác suất của nó. Điều này được thực hiện bằng cách sử dụng hàm softmax, tính toán xác suất có điều kiện của đối tượng thuộc mỗi lớp.

Hộp giới hạn: Đối với mỗi ô dự đoán một đối tượng, YOLO cũng dự đoán hộp giới hạn bao quanh đối tượng. Hộp giới hạn được biểu thị bằng tọa độ tâm, chiều rộng và chiều cao của nó và được dự đoán tương ứng với kích thước ô.



Loại bỏ không tối đa: Để loại bỏ các hộp giới hạn dư thừa và cải thiện độ chính xác của các phát hiện, YOLO thực hiện loại bỏ không tối đa (NMS) trên các hộp giới hạn được dự đoán. NMS loại bỏ tất cả các hộp giới hạn chồng chéo với điểm tin cậy thấp hơn.

Đầu ra: Đầu ra cuối cùng của YOLO là một danh sách các hộp giới hạn với lớp liên quan và điểm tin cậy, đại diện cho các đối tượng được phát hiện trong hình ảnh đầu vào.




Hình 10. Ví dụ về cách hoạt động của Yolo

#### 2.3.4 Hàm tính IoU (INTERSECTION OVER UNION)

IOU (INTERSECTION OVER UNION) là hàm đánh giá độ chính xác của object detector trên tập dữ liệu cụ thể. IOU được tính bằng:



$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


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

### 2.3.5 Hàm mất mát (Loss Function)

Hàm lỗi trong Yolov8 được tính trên việc dự đoán hộp giới hạn, nhãn mô hình để tính:

- Độ lỗi của việc dự đoán loại nhãn của object - Cls loss
- Độ lỗi của dự đoán tọa độ tâm, chiều dài, rộng của hộp giới hạn (x, y, w, h) - BBox loss.

### 2.3.6 Tối ưu hóa (Optimizer)

Theo mặc định, Yolov8 sử dụng Stochastic Gradient Descent (SGD) với momentum là 0.937, weight decay là 0.0005, và learning rate là 0.01. Bên cạnh đó Yolov8 còn sử dụng Adaptive Moment Estimation (Adam), Root Mean Square Propagation (RMSprop), Adam with weight decay regularization (AdamW).

```
if name == 'Adam':
    optimizer = torch.optim.Adam(g[2], lr=lr, betas=(momentum, 0.999)) # adjust beta1 to momentum
elif name == 'AdamW':
    optimizer = torch.optim.AdamW(g[2], lr=lr, betas=(momentum, 0.999), weight_decay=0.0)
elif name == 'RMSProp':
    optimizer = torch.optim.RMSprop(g[2], lr=lr, momentum=momentum)
elif name == 'SGD':
    optimizer = torch.optim.SGD(g[2], lr=lr, momentum=momentum, nesterov=True)
else:
    raise NotImplementedError(f'Optimizer {name} not implemented.')
```

## 2.4 Firebase

### a. Giới thiệu về Firebase

Firebase là dịch vụ cơ sở dữ liệu hoạt động trên nền tảng đám mây – cloud. Kèm theo đó là hệ thống máy chủ cực kỳ mạnh mẽ của Google. Chức năng chính là giúp người dùng lập trình ứng dụng bằng cách đơn giản hóa các thao tác với cơ sở dữ liệu. Cụ thể là những giao diện lập trình ứng dụng API đơn giản.

### b. Cách thức hoạt động Firebase bao gồm việc

- Lưu trữ dữ liệu thời gian thực.
- Xác thực người dùng.
- Firebase hosting.

Khi đăng ký một tài khoản trên Firebase để tạo ứng dụng, bạn đã có một cơ sở dữ liệu thời gian thực. Dữ liệu bạn nhận được dưới dạng JSON. Đồng thời nó cũng luôn được đồng bộ thời gian thực đến mọi kết nối client.

Trong trường hợp bị mất mạng, dữ liệu được lưu lại ở local. Vì thế khi có mọi sự thay đổi nào đều được tự động cập nhật lên Server của Firebase. Bên cạnh đó, đối với các dữ liệu ở local cũ hơn với Server thì cũng tự động cập nhật để được dữ liệu mới nhất.

### c. Xây dựng và ứng dụng

Ưu điểm của Firebase:

- Tạo tài khoản và sử dụng dễ dàng.
- Tốc độ phát triển nhanh.
- Nhiều dịch vụ trong một nền tảng.
- Được cung cấp bởi Google.
- Tập trung vào phát triển giao diện người dùng.
- Firebase không có máy chủ.

- Học máy (Machine Learning).
- Tạo lưu lượng truy cập.
- Theo dõi lỗi.
- Sao lưu.

Realtime Database có cấu trúc:

- Devices - Thiết bị
  - token: Device token. - Mã thiết bị.
  - name: Device name. - Tên thiết bị.
    - date: Date in unix timestamp. - Tem đánh dấu thời gian.
    - image\_url: Image with fire detected. - Hình ảnh khi có cảnh báo cháy.

Firebase Storage:

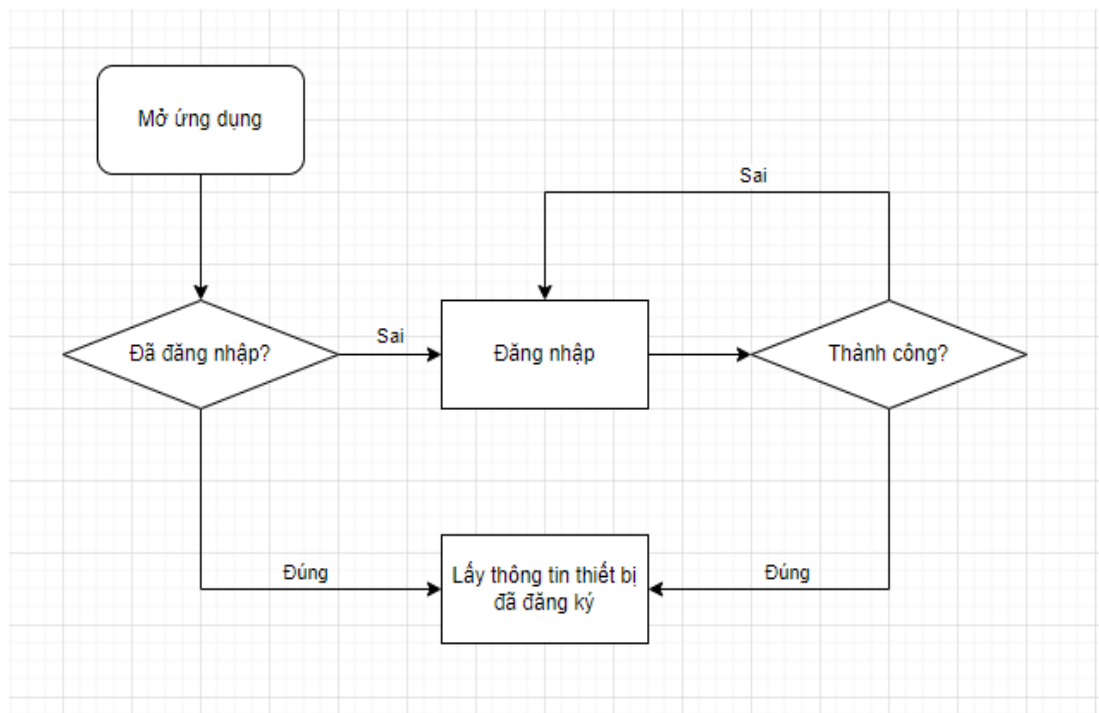
- Hình ảnh tải lên từ Raspberry Pi.
  - uid: là token của thiết bị đã đăng ký
  - date : thời gian mà ảnh đẩy lên Firebase Storage
  - Ext: có thể là đuôi jpg, png, ...

## 2.5 Ứng dụng di động

Ứng dụng di động này được xây dựng trên nền tảng Android. Ứng dụng này hướng đến người dùng là người chủ sở hữu camera. Ứng dụng sẽ có các chức năng chính sau:

### a. Đăng nhập

Khi bật ứng dụng lần đầu tiên, người dùng sẽ được yêu cầu đăng nhập. Thông tin đăng nhập của người dùng sẽ được cung cấp bởi người chủ camera gồm email và mật khẩu. Thông tin đăng nhập này sẽ được lưu lại để không cần đăng nhập vào các lần mở ứng dụng sau này.

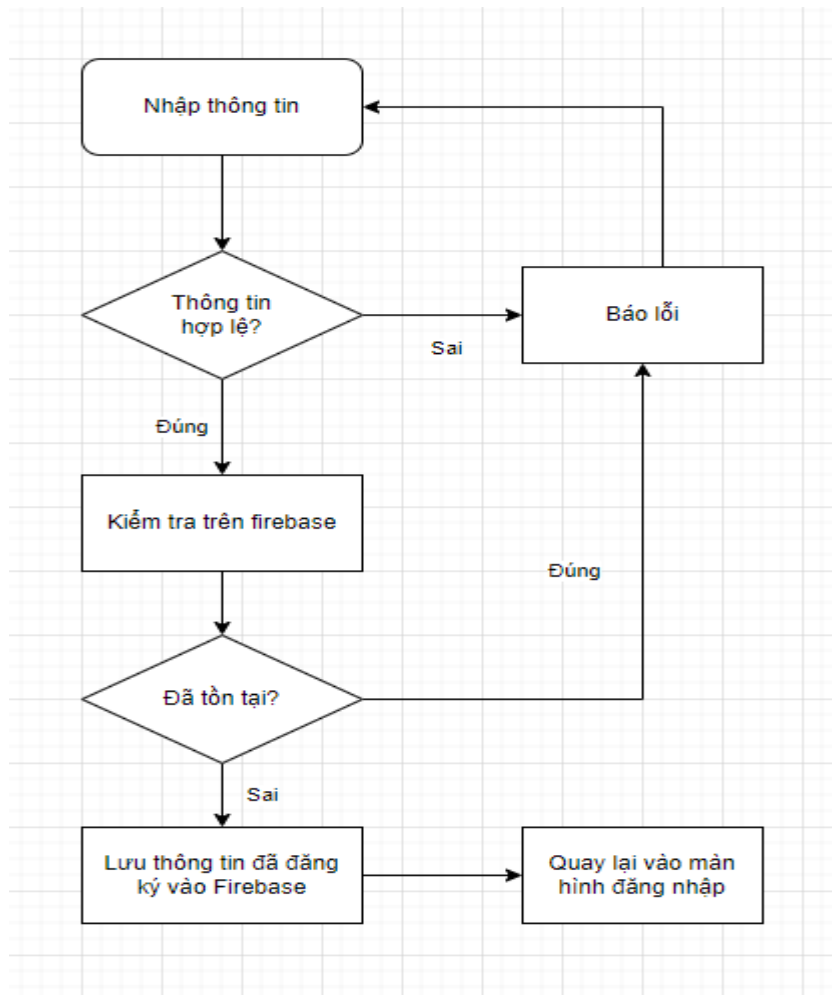


Hình 11. Sơ đồ quá trình đăng nhập

b. Tạo tài khoản mới

Những thông tin trên được gửi lên Firebase nếu hợp lệ thì Firebase sẽ tự động tạo ID cho tài khoản để cung cấp cho người dùng với mục đích xác thực lúc đăng nhập và các thông tin này được lưu vào Firebase Authentication table và Firestore Realtime Database.

Sơ đồ thể hiện quá trình tạo tài khoản và lưu dữ liệu được thể hiện như sau:

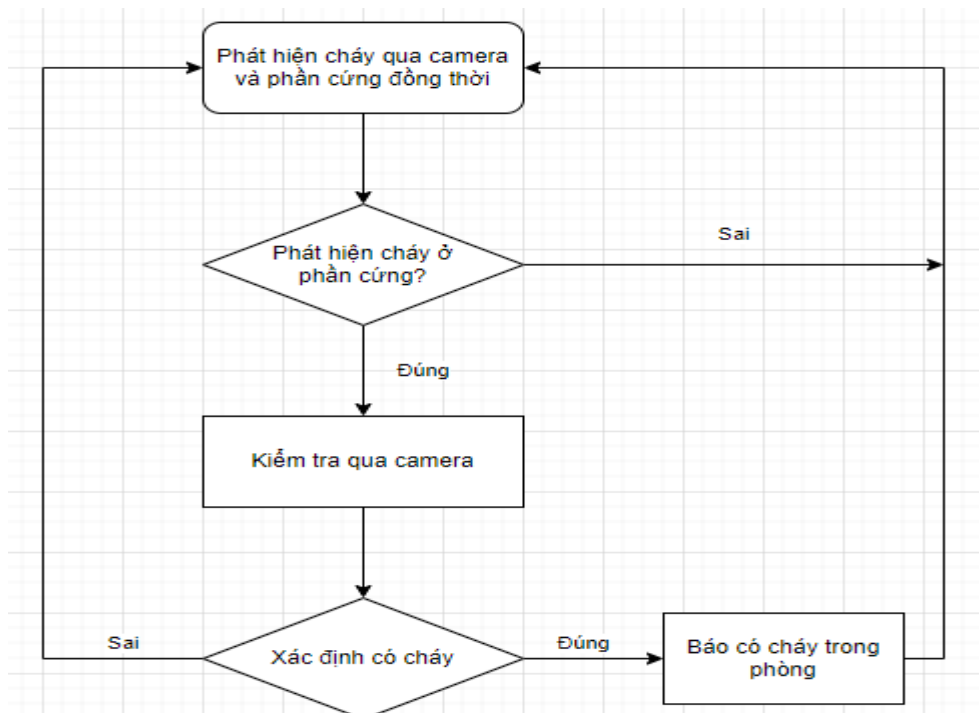


Hình 12. Sơ đồ quá trình đăng ký

c. Nhận thông báo nếu có dấu hiệu bất thường

Khi đã đăng nhập và đăng ký thiết bị vào hệ thống, người dùng có thể nhận thông báo khi có cháy.

Sơ đồ thể hiện quá trình này được mô tả như sau:

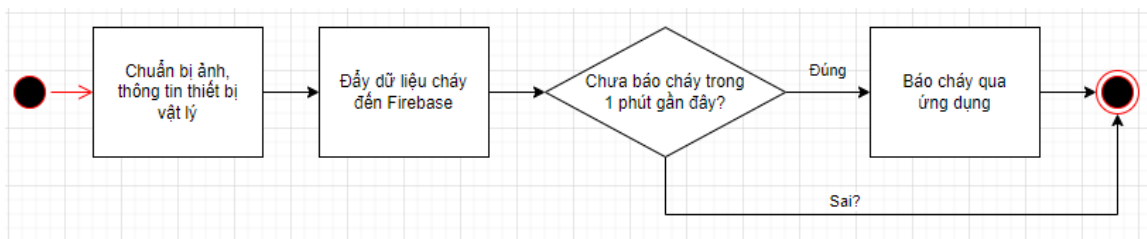


Hình 13. Sơ đồ quá trình xử lý cháy

#### d. Xử lý dữ liệu và gửi qua Firebase

Khi đã phát hiện cháy, cần xử lý qua server để nó gửi dữ liệu vụ cháy lên Firebase. Nếu có cháy, nó sẽ gửi dữ liệu kèm theo báo cháy cho người dùng. Thông báo sẽ gửi liên tục cho đến khi hết cháy.

Sơ đồ thể hiện quá trình này được mô tả như sau:



Hình 14. Sơ đồ quá trình gửi vụ cháy đến Firebase

### 3. KẾT QUẢ

#### 3.1. Mô hình nhận diện

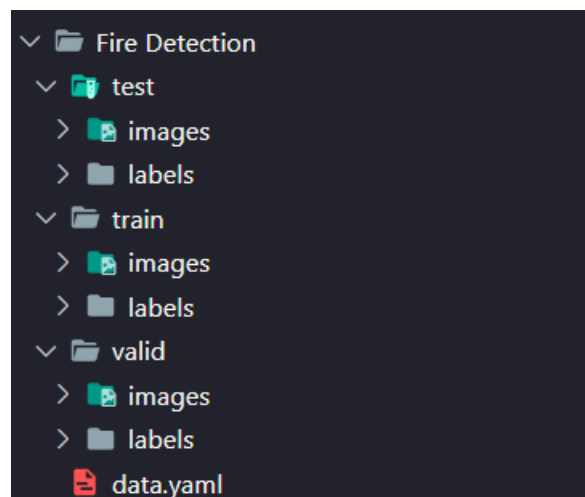
##### a. Dữ liệu sử dụng để train

Trong phần huấn luyện nhận diện và phân lớp lửa với khói, nhóm đã sử dụng tập dữ liệu có sẵn. Thông tin về tập dữ liệu:

<https://universe.roboflow.com/fypfirerooster/fire-detection-classification/dataset/4>

Mô tả: Tập dữ liệu trên có 4521 hình ảnh và gán nhãn 8473 đối tượng trong 3 lớp: nf, fire, smoke. Tất cả các ảnh trong tập dữ liệu đều có cùng kích thước là 416 x 416.

Trong đó, tập dữ liệu đã được chia sẵn thành 3 tập: Train, Val, Test và tệp YAML nhằm mục đích huấn luyện mô hình. Mỗi tập lại chứa 2 tập con là: images chứa danh sách ảnh và labels chứa danh sách nhãn ứng với mỗi ảnh. Phân chia dữ liệu: Train: 2959 (67%), Val: 1032(23%), Test: 444 (10%).



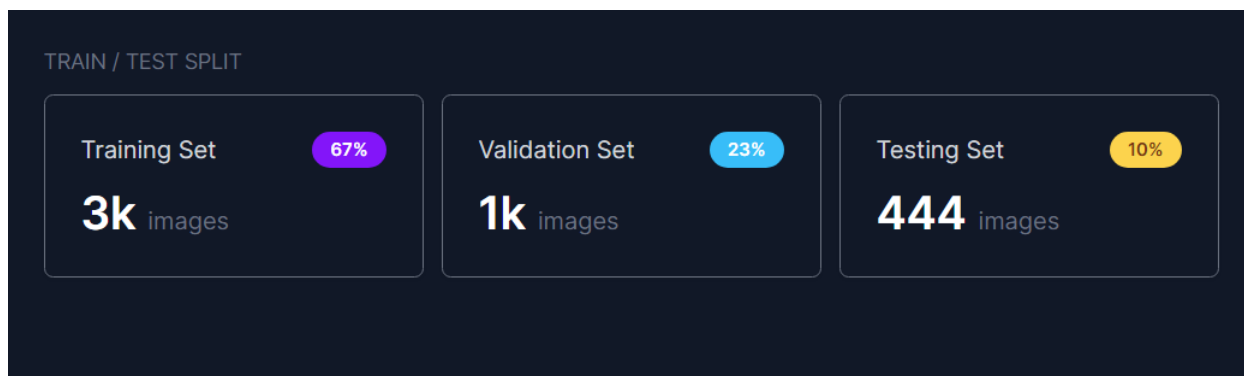
Hình 15. Cấu trúc tập dữ liệu

Mỗi dòng trong file nhãn bao gồm: <object-class> <x> <y> <width> <height>. Trong đó:

<object-class>: Chỉ số đánh dấu các class. Trong bài toán này chỉ có 3 classes là fire ứng với giá trị 0, nf tương ứng với giá trị 1 và smoke ứng với giá trị 2.

<x><y>: tâm tọa độ của bounding boxes.

<width><height>: kích thước của đối tượng.



Hình 16. Kích thước tập dữ liệu

b. Kết quả huấn luyện

Sau khi có được dữ liệu và phân chia dữ liệu theo tỉ lệ phù hợp. Nhóm đã tiến hành tải mô hình YOLOv8 để đi huấn luyện trên Google Colab. Khi huấn luyện nhóm đã thay đổi các tham số trong mô hình như: epochs, lr0, batch size, imgsz, patience, optimizer.

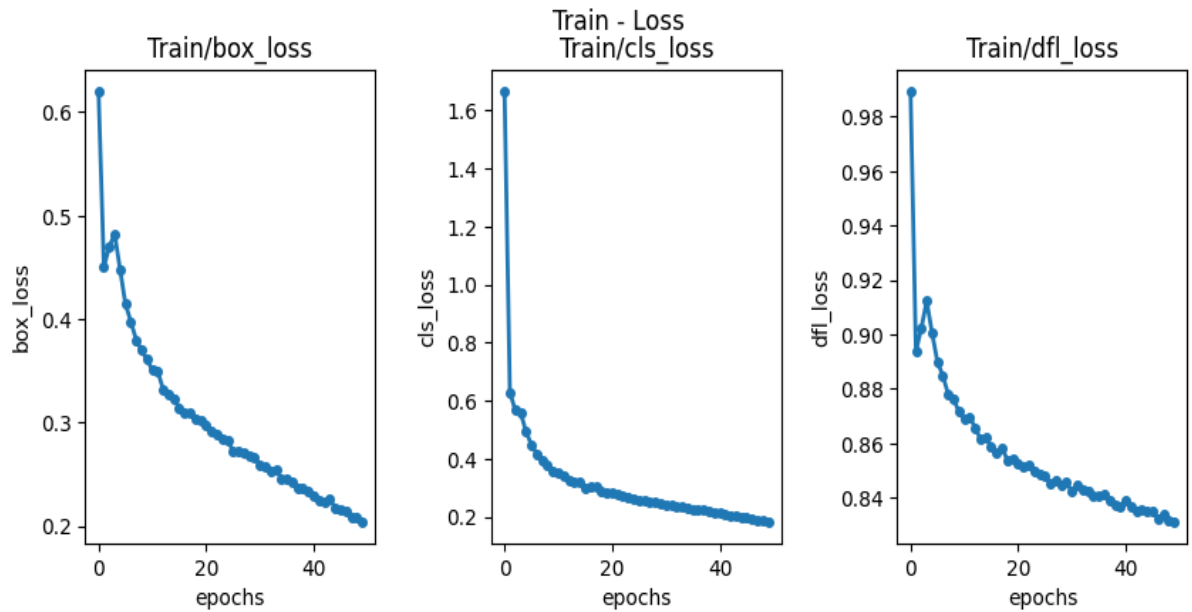
**Training setting:**

- Epochs: 50
- Patience: 40
- Batch: 16
- Imgsz: 416
- Lr0: 0.001
- Optimizer: Adam

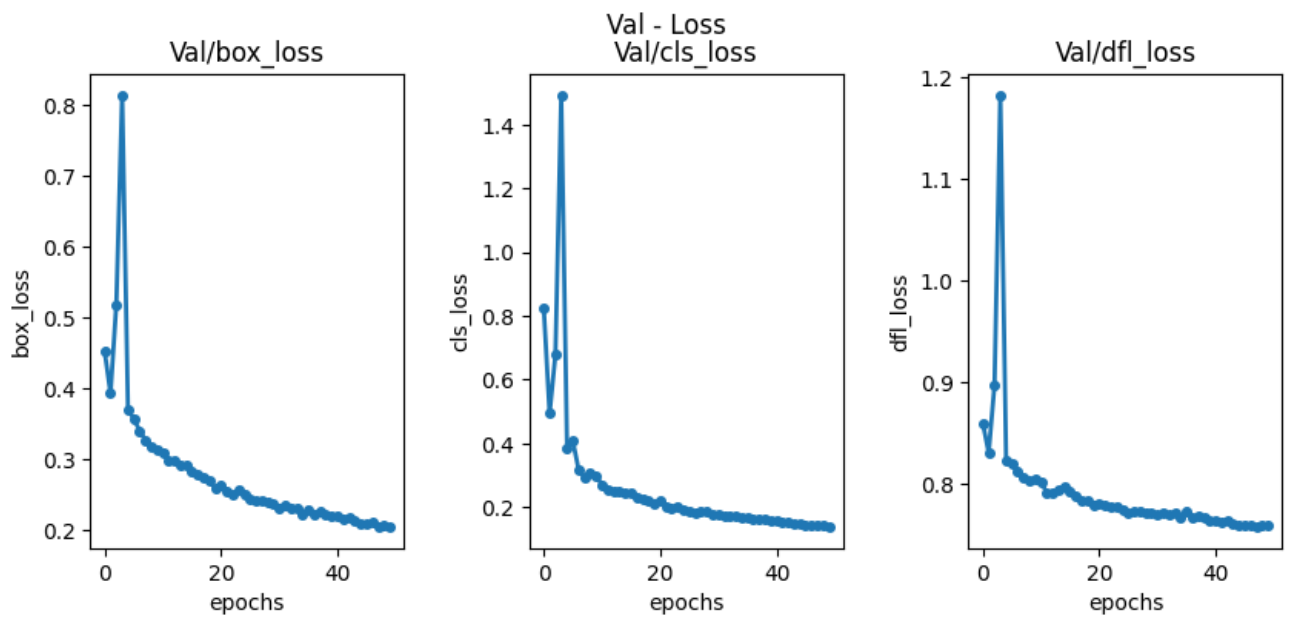
Sau mỗi lượt train xong, YOLOv8 sẽ lưu trữ model tốt nhất thông qua file “best.pt”. File này lưu trữ trọng số tại một điểm tốt nhất trong quá trình huấn luyện. Sau đó tiếp tục huấn luyện mô hình với file trọng số “best.pt”.

Trong quá trình train sẽ ngừng huấn luyện khi không có sự thay đổi đáng kể hay nói cách khác hàm loss giảm theo thời gian cho đến khi hội tụ.

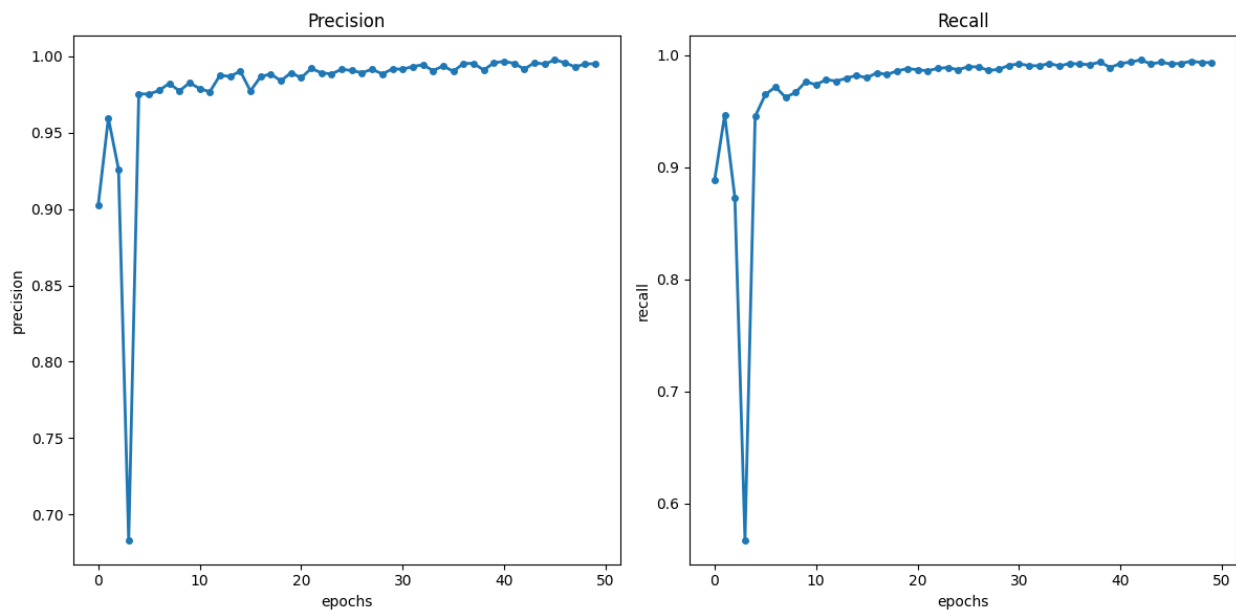




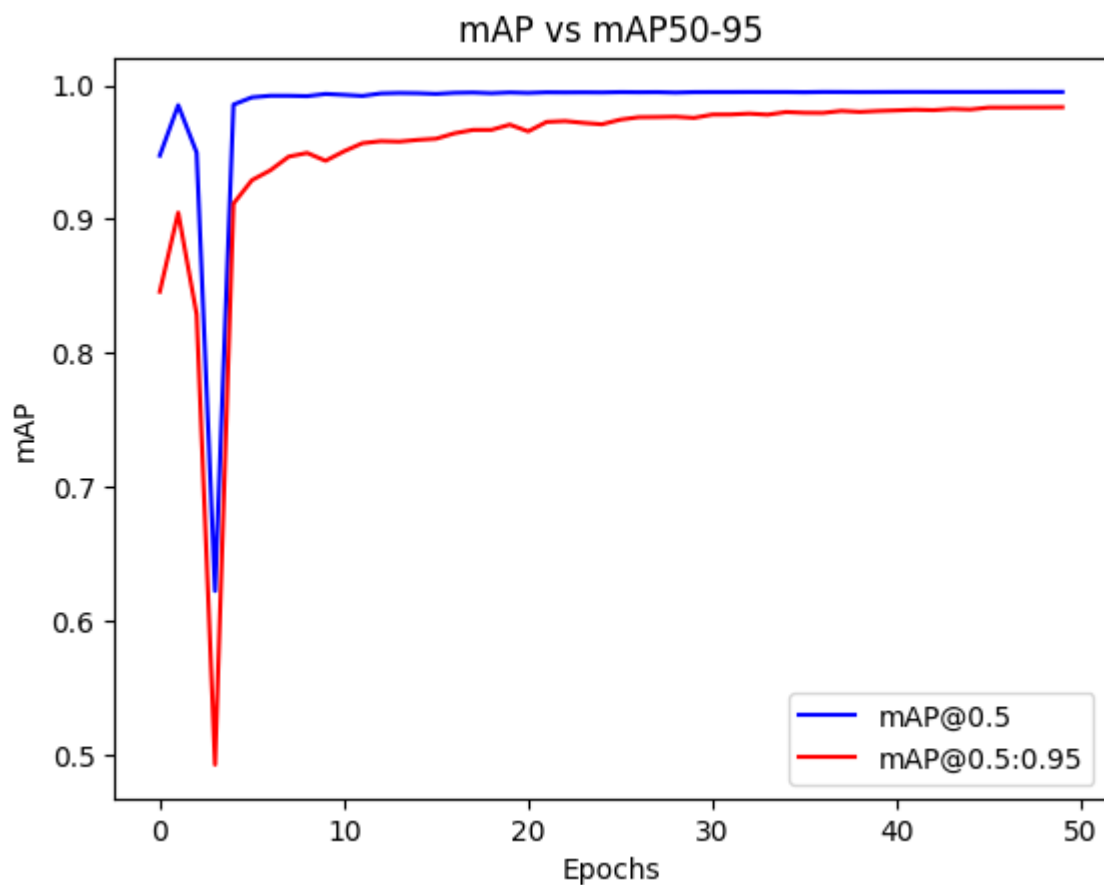
Hình 17. Đồ thị biểu diễn các hàm loss YOLOv8 trên tập train



Hình 18. Đồ thị biểu diễn các hàm loss YOLOv8 trên tập val



Hình 19. Đồ thị các chỉ số Precision và Recall



Hình 20. Đồ thị chỉ số mAP và mAP50-95

Precision: Là độ đo đánh giá độ tin cậy của kết luận đưa ra (bao nhiêu % lời kết luận của model là chính xác).

Recall: Là độ đo đánh giá khả năng tìm kiếm toàn bộ các ground truth của mô hình (bao nhiêu % positive sample mà model nhận diện được).

- True Positive (TP): Phát hiện chính xác. Phát hiện với  $\text{IOU} \geq \text{ngưỡng}$ .
- False Positive (FP): Phát hiện sai. Phát hiện với  $\text{IOU} < \text{ngưỡng}$ .
- False Negative (FN): Không phát hiện ra đối tượng
- True Negative (TN): Tất cả các hộp giới hạn có thể không được phát hiện.

Ngưỡng: tùy thuộc vào số liệu, nó thường được đặt thành 50%, 75% hoặc 95%.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} = \frac{\text{TP}}{\text{all detections}}$$
$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} = \frac{\text{TP}}{\text{all ground truths}}$$

Mean Average Precision (mAP): mAP bằng trung bình của chỉ số Average Precision (AP) trên tất cả các lớp trong một mô hình.

$$mAP = \frac{1}{n} \sum_{k=1}^{k=n} AP_k$$

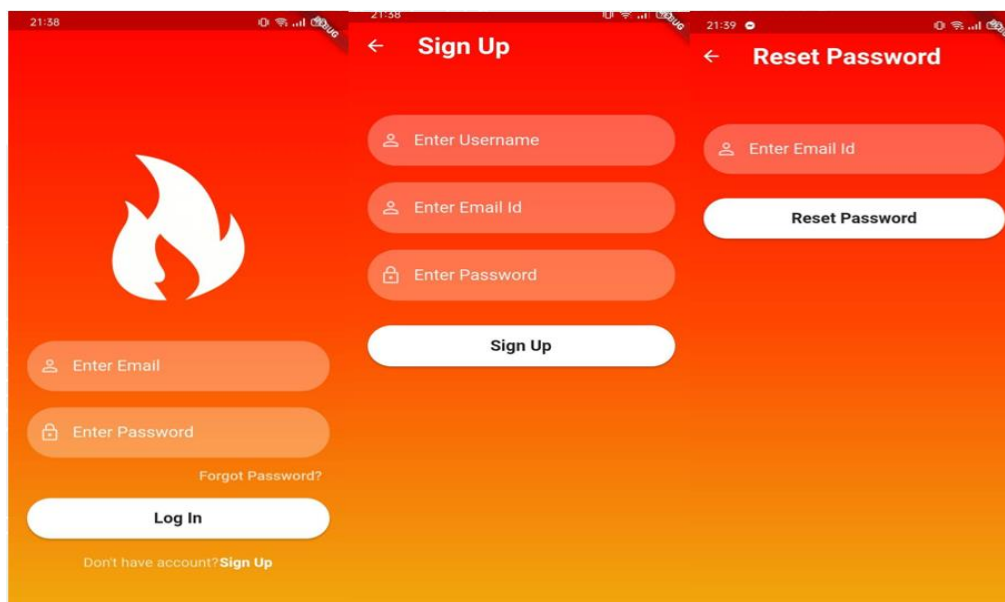
**$AP_k = \text{the AP of class } k$**   
 **$n = \text{the number of classes}$**

c. Kết quả nhận diện bằng ảnh trên tập kiểm thử

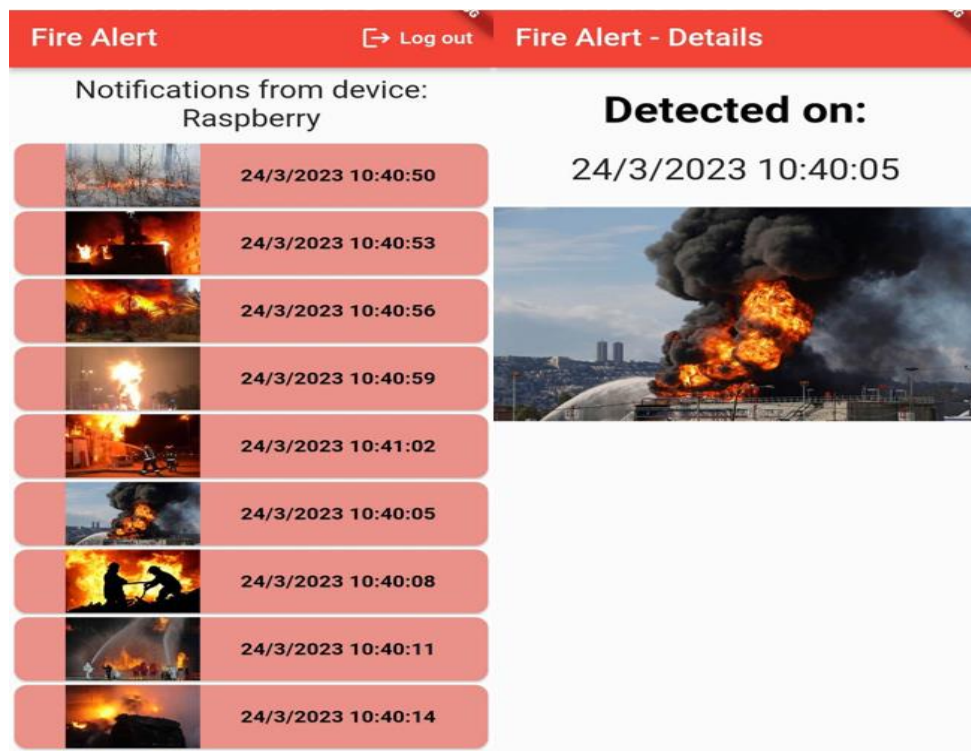


Hình 21. Nhận diện trên tập thử nghiệm

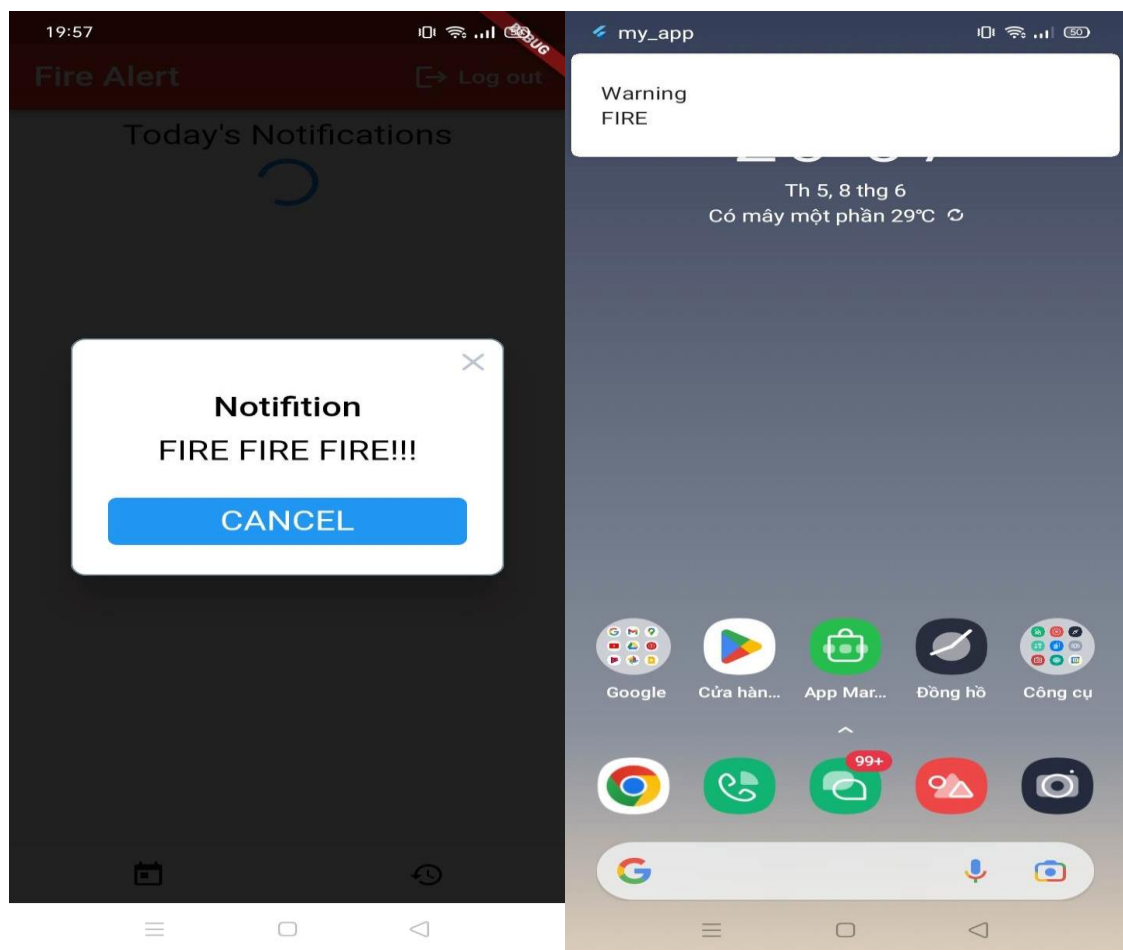
3.2. Lập trình ứng dụng



Hình 22. Giao diện đăng nhập/đăng ký ứng dụng



Hình 23. Giao diện chính và giao diện khi mở một vụ cháy chi tiết



Hình 24. Thông báo nếu có cháy trong hiện trường



Hình 25. Giao diện tổng hợp các vụ cháy đã xảy ra trong quá khứ

### 3.3. Thử nghiệm hệ thống

Thử nghiệm hệ thống thực tế cho kết quả khá tốt. Khả năng phát hiện nhanh ở khoảng cách gần. Nhưng vẫn có một số trường hợp phát hiện không tốt do các yếu tố bên ngoài như thiết bị, model để train chưa chính xác. Mặc dù FPS khi nhận diện không cao nhưng cho kết quả nhận diện khá ổn định.





Hình 26. Phát hiện cháy

### 3.4. Đánh giá

#### a. Ưu điểm

- Các phần của hệ thống kết nối, giao tiếp với nhau ổn định.
- Thực hiện đầy đủ các chức năng đề ra.
- Khả năng mở rộng tốt.

#### b. Khuyết điểm

- Ứng dụng di động còn sơ sài.
- Một vài thiết bị bị hỏng hóc, khi tiến hành còn một chút trục trặc.
- Tốc độ xử lý khung hình không cao.
- Chất lượng camera không được tốt, bị ám xanh -> Độ chính xác phát hiện giảm.

## 4. KẾT LUẬN

### 4.1. Kết luận từ đồ án

- Nhóm đã hoàn thành tương đối các yêu cầu ban đầu.
- Tăng khả năng làm việc nhóm.
- Học được lập trình vi điều khiển, huấn luyện mô hình học máy.
- Học được lập trình Android, vận hành giao tiếp giữa Android và cơ sở dữ liệu Firebase.
- Chuyên nghiệp hơn trong tư duy, xây dựng định hướng công việc.

### 4.2. Tồn tại những điểm yếu

- Cần cải thiện và thiết kế lại đề mọt mà, bắt mắt hơn.
- Cần có kế hoạch phân chia công việc tốt hơn từ ban đầu.
- Cần học thêm về quản lý thời gian.
- Hệ thống bị chậm.

### 4.3. Hướng phát triển

- Tăng độ hoàn thiện ứng dụng Android
- Làm thêm mô hình nhận diện người và tăng độ chính xác.
- Thêm các tính năng như kích hoạt hệ thống chữa cháy nước.
- Cải thiện về khả năng phát hiện có lửa.



## 5. TÀI LIỆU THAM KHẢO

[1] Brief summary of YOLOv8 model structure

<https://github.com/ultralytics/ultralytics/issues/189>

[2] Object-Detection-Metrics

<https://github.com/rafaelpadilla/Object-Detection-Metrics>

[3] How to Detect Objects in Images Using the YOLOv8 Neural Network

<https://www.freecodecamp.org/news/how-to-detect-objects-in-images-using-yolov8/>

[4] Evolution of YOLO Object Detection Model From V5 to V8

<https://www.labellerr.com/blog/evolution-of-yolo-object-detection-model-from-v5-to-v8/>

Link mã nguồn: [HuynhBaThuan/FireAlert \(github.com\)](https://github.com/HuynhBaThuan/FireAlert)