ĐẠI HỌC ĐÀ NẴNG

TRƯỜNG ĐẠI HỌC BÁCH KHOA

KHOA ĐIỆN

ĐỒ ÁN TỐT NGHIỆP

CAPSTONE PROJECT

NGÀNH: KỸ THUẬT ĐIỀU KHIỂN VÀ TỰ ĐỘNG HÓA

ĐỀ TÀI:

NGHIÊN CỨU VÀ XÂY DỰNG HỆ THỐNG GIÁM SÁT TỰ ĐỘNG PHÁT HIỆN KHÓI LỬA QUA CAMERA GIÁM SÁT

Người hướng dẫn: **TS. NGUYỄN THỊ KIM TRÚC**

KS. NGUYỄN HOÀNG ANH KHOA

Sinh viên thực hiện:

NGUYỄN KIM HOÀNG – MSSV: 105200492 – LỚP: 20TDHCLC4

Đà Nẵng, 6/2026

TÓM TẮT

Tên đề tài:

NGHIÊN CỨU VÀ XÂY DỰNG HỆ THỐNG GIÁM SÁT TỰ ĐỘNG PHÁT HIỆN KHỎI LỬA QUA CAMERA GIÁM SÁT

Sinh viên thực hiện : Nguyễn Kim Hoàng

Số thẻ SV: 105200492 Lớp : 20TDHCLC4

Đề tài này tập trung nghiên cứu, thiết kế, chế tạo và thử nghiệm thiết bị tự động nhận dạng khói, lửa và cảnh báo đến người sử dụng. Cụ thể ở đây là ứng dụng thị giác máy tính để nhận dạng khói, lửa trong các khu vực mà thiết bị phát hiện được tới người dùng thông qua internet nhằm hỗ trợ việc quản lý PCCC trong các khu vực khó khăn. Tác giả áp dụng mô hình Resnet50 đối với bài toán phát hiện đối tượng (Object Detection). Kết quả thử nghiệm cho thấy thiết bị hoạt động ổn định và nhận dạng được các đóm cháy và khói khó phát hiện mà tác giả đã cho huấn luyện mô hình môi trường ánh sáng tự nhiên.

Tác giả đã tiến hành thu nhập dữ liệu hình ảnh, gán nhãn khói lửa trên Make Sense AI, huấn luyện thành công mô hình Resnet50 trên Google Colab, thực hiện kiểm tra mô hình vừa huấn luyện được bằng cách nhận dạng khói lửa qua công cụ VS code, sau đó viết chương trình giao diện người dùng và đưa ra cảnh báo kèm hình ảnh qua ứng dụng (app). Cuối cùng đưa chương trình vào máy tính nhúng Raspberry Pi 5. Tất cả các cơ sở lý thuyết, thiết kế, mô phỏng, chế tạo và thử nghiệm cho thấy đề tài hoàn thành được các mục tiêu đã đề ra từ ban đầu. Trong đề tài này, mô hình Resnet50, thư viện Open CV của Python được khai thác để tích hợp vào hệ thống dùng để nhận khói lửa. Đồng thời, streaming video lên App để người dùng có thể xem trực tiếp. Sử dụng giao thức HTTP/RTSP để truyền gửi cảnh báo kèm hình ảnh qua ứng dụng người dùng

Phụ lục 01

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

Họ tên sinh viên: Nguyễn Kim Hoàng Số thẻ sinh viên: 105200492

Lớp: 20TDHCLC4 Khoa: Điện Ngành: Kỹ thuật điều khiển và tự động hóa

1. *Tên đề tài đồ án:*

NGHIÊN CỨU VÀ XÂY DỰNG HỆ THỐNG GIÁM SÁT TỰ ĐỘNG PHÁT HIỆN KHÓI LỬA QUA CAMERA GIÁM SÁT

2. *Đề tài thuộc diện:* ☑ *Có ký kết thỏa thuận sở hữu trí tuệ đối với kết quả thực hiện*
3. *Các số liệu và dữ liệu ban đầu:*

Dựa trên đặc điểm và hành vi đặc trưng của hiện tượng cháy nổ (khói và lửa) trong môi trường trong nhà như tòa nhà, văn phòng và xưởng sản xuất, đề tài tiến hành thu thập và xử lý dữ liệu hình ảnh từ hệ thống camera IP thực tế để làm cơ sở xây dựng mô hình nhận dạng, để chế tạo thiết bị nhận dạng khói lửa, xây dựng giao diện người dùng và gửi cảnh báo về ứng dụng và thiết bị người dùng

Nội dung các phần thuyết minh và tính toán:

- Phân tích cơ sở lý thuyết về thị giác máy tính, CNN, kiến trúc ResNet, cơ chế hoạt động từng lớp (convolution, pooling, activation, fully connected).
- Mô hình hóa quy trình nhận dạng khói/lửa bằng ResNet50: xây dựng pipeline từ thu nhận ảnh → tiền xử lý → suy luận mô hình → cảnh báo.
- Đánh giá hiệu năng mô hình qua các chỉ số: accuracy, precision, recall, F1-score, loss.
- Phân tích thời gian xử lý: suy luận trung bình ~15 ms/ảnh với TFLite, tổng độ trễ toàn hệ thống ~30 ms .
- Thử nghiệm hiệu suất trên RPi khi xử lý nhiều camera đồng thời: theo dõi FPS, độ trễ và mức sử dụng CPU/RAM theo từng cấu hình .

4. *Các bản vẽ, đồ thị (ghi rõ các loại và kích thước bản vẽ):*

Các bản vẽ sơ đồ nguyên lý, sơ đồ hệ thống:
+ Hình 3.6: Mô hình kiến trúc hệ thống (khổ A4)
+ Hình 3.7: Sơ đồ kết nối thiết bị phần cứng (khổ A4)
+ Hình 3.15: Sơ đồ kết nối với module SIM900A (khổ A4)
+ Hình 3.16: Lưu đồ thuật toán chương trình nhận dạng và cảnh báo (khổ A4)

Đồ thị huấn luyện mô hình:
+ Hình 3.9: Biểu đồ Loss và Accuracy theo epochs (khổ A4)

5. Họ tên người hướng dẫn:	*Phần/ Nội dung:*
TS. Nguyễn Thị Kim Trúc	Theo dõi, hướng dẫn và góp ý kiến trong quá trình thực hiện đề tài Cung cấp thông tin, tài liệu liên quan đền đề tài
KS. Nguyễn Hoàng Anh Khoa	

6. *Ngày giao nhiệm vụ đồ án: 03/03/2025*
7. *Ngày hoàn thành đồ án: 16/06/2025*

Đà Nẵng, ngày 16 tháng 06 năm 2025

Trưởng Bộ môn Tự động hóa	**Người hướng dẫn 1**	**Người hướng dẫn 2**
TS. Giáp Quang Huy	TS. Nguyễn Thị Kim Trúc	KS. Nguyễn Hoàng Anh Khoa

ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA ĐIỆN

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

PHIẾU KIỂM SOÁT TIẾN ĐỘ LÀM ĐỒ ÁN TỐT NGHIỆP

(Phiếu dành cho người hướng dẫn/sinh viên)

Họ tên sinh viên: Nguyễn Kim Hoàng Số thẻ SV : 105200492

Tên đề tài ĐATN:

NGHIÊN CỨU VÀ XÂY DỰNG HỆ THỐNG GIÁM SÁT TỰ ĐỘNG PHÁT HIỆN KHÓI LỬA QUA CAMERA GIÁM SÁT

Họ tên người hướng dẫn:

TS. Nguyễn Thị Kim Trúc, Khoa Điện, Trường Đại học Bách Khoa – Đại học Đà Nẵng

KS. Nguyễn Hoàng Anh Khoa, Công ty TNHH HS Hyosung Quảng Nam

Tuần	**Ngày**	**Khối lượng**		**GVHD ký tên**
		đã thực hiện (%)	**tiếp tục thực hiện (%)**	
1	3/3/2025	Nhận đề tài	Gặp giảng viên hướng dẫn và trao đổi với đại diện doanh nghiệp	
2	10/3/2025	Tìm hiểu các kiến thức liên quan, xây dựng giải pháp cho vấn đề	Báo cáo hoạt động với đại diện doanh nghiệp	
3	17/3/2025	Báo cáo lại giải pháp cho doanh nghiệp, thống nhất phương án thiết kế	Viết báo cáo Chương 1	
4	24/3/2025	Duyệt lần 1: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
5	31/3/2025	Nghiên cứu ngôn ngữ lập trình Python, open CV	Nghiên cứu ngôn ngữ lập trình Python, open CV	
6	7/4/2025	Nắm được cơ bản ngôn ngữ lập trình Python, open CV	Nghiên cứu về mô hình mạng ResNet50	

7	14/4/2025	Thu thập mẫu dữ liệu hình ảnh từ đối tượng thực tế Nghiên cứu training dữ liệu trên Google Colab	Viết báo cáo chương 2	
8	21/4/2025	Duyệt lần 2: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
9	28/4/2025	Tiến hành training dữ liệu trên Google Colab. Nhận dạng được đối tượng khói lửa trên laptop	Lựa chọn phương án thiết kế phần cứng thiết bị tự động nhận dạng và cảnh báo khói lửa	
10	5/5/2025	Nghiên cứu về Raspberry Pi 5 và tiến hành nhận dạng đối tượng Thiết kế giao diện người dùng		
11	12/5/2025	Đưa video trực tiếp từ camera lên ứng dụng và cảnh báo đến thiết bị người dùng	Viết báo cáo chương 3	
12	19/5/2025	Duyệt lần 3: Đánh giá khối lượng hoàn thành _____ % : Được tiếp tục làm ĐATN □ Không tiếp tục thực hiện ĐATN □		
13	26/5/2025	Hoàn thiện thiết bị, tiến hành thực nghiệm	Hiệu chỉnh chương trình Viết báo cáo chương 4	
14	2/6/2025	Tiến hành thực nghiệm và hiệu chỉnh để hoàn thiện sản phẩm	Tổng hợp tài liệu, hoàn thiện báo cáo theo mẫu	
15	9/6/2025	Viết thuyết minh, hoàn thiện đề tài	In thuyết minh	

LỜI NÓI ĐẦU

Hiện nay khoa học kĩ thuật đang phát triển rất nhanh mang lại nhiều lợi ích to lớn cho con người cả vật chất lẫn tinh thần. Nhà nước ta đang đẩy mạnh cách mạng công nghiệp 4.0 để nâng cao đời sống của nhân dân và hoà nhập với sự phát triển chung của các nước trên thế giới. Nhằm thực hiện cuộc cách mạng phát triển công nghệ cao và xuất phát từ những khó khăn trong quá trình phòng cháy chữa cháy. Việc phát hiện khói lửa kịp thời rất quan trọng cùng với sự phát triển vượt bậc của trí tuệ nhân tạo. Nhận thấy được những yếu tố tiềm năng tác giả đã chọn đề tài "Nghiên cứu và xây dựng hệ thống giám sát tự động phát hiện khói lửa qua camera giám sát" để thực hiện nghiên cứu khoa học.

Trong suốt quá trình học tập và thực hiện nghiên cứu khoa học em luôn được sự quan tâm, hướng dẫn và giúp đỡ tận tình của các thầy, cô giáo trong Khoa Điện cùng với sự hỗ trợ Cty TNHH HS Hyosung. Lời đầu tiên em xin được bày tỏ lòng biết ơn sâu sắc đến Ban giám hiệu Trường Đại học Bách Khoa - Đại học Đà Nẵng và Ban chủ nhiệm Khoa Điện đã tận tình giúp đỡ cho em suốt thời gian học tại trường. Đặc biệt em xin bày tỏ lòng biết ơn chân thành sâu sắc tới thầy giáo TS. Nguyễn Thị Kim Trúc và anh KS. Nguyễn Hoàng Anh Khoa đã trực tiếp giúp đỡ, hướng dẫn em hoàn thành báo cáo này. Tuy nhiên do kiến thức còn hạn hẹp, chưa tiếp xúc nhiều với thực tiễn cũng như các tài liệu tham khảo còn quá ít, trong khi đó thời gian thực hiện cũng có hạn nên trong đồ án không tránh khỏi những thiếu sót, rất mong nhận được những lời chỉ dẫn thêm từ các thầy, cô. Cuối cùng em kính chúc quý thầy, cô dồi dào sức khỏe và thành công trong sự nghiệp cao quý.

CAM ĐOAN

Tôi xin cam đoan đây là công trình nghiên cứu của cá nhân tôi và được sự hướng dẫn khoa học của TS. Nguyễn Thị Kim Trúc, Kỹ sư. Nguyễn Hoàng Anh Khoa. Các nội dung nghiên cứu trong đề tài "Nghiên cứu và xây dựng hệ thống giám sát tự động phát hiện khói lửa qua camera giám sát" của tôi là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng thống kê phục vụ cho việc phân tích, nhận xét, đánh giá được cá nhân thu thập từ các nguồn khác nhau có ghi rõ nguồn gốc. Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung bài báo cáo của mình.

Sinh viên thực hiện

Nguyễn Kim Hoàng

\

MỤC LỤC

DANH SÁCH HÌNH ẢNH

DANH SÁCH BẢNG VẼ

DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT

CHỮ VIẾT TẮT

AI	Artificial Intelligence – Trí tuệ nhân tạo
CNN	Convolutional Neural Network – Mạng nơ-ron tích chập
CPU	Central Processing Unit – Bộ xử lý trung tâm
GPU	Graphics Processing Unit – Bộ xử lý đồ họa
RTSP	Real-Time Streaming Protocol – Giao thức truyền phát thời gian thực
HTTP	HyperText Transfer Protocol – Giao thức truyền siêu văn bản
RPi / RPi5	Raspberry Pi / Raspberry Pi 5 – Máy tính nhúng kích thước nhỏ
TFLite	TensorFlow Lite – Phiên bản nhẹ của thư viện học sâu TensorFlow
SMS	Short Message Service – Dịch vụ tin nhắn văn bản ngắn
SIM900A	Mô-đun GSM/GPRS dùng để gửi SMS và gọi điện
GUI	Graphical User Interface – Giao diện người dùng đồ họa
Kivy	Thư viện Python mã nguồn mở để xây dựng GUI đa nền tảng
FCM	Firebase Cloud Messaging – Dịch vụ gửi thông báo đẩy của Google
mAP	Mean Average Precision – Trung bình độ chính xác trung bình (detection)
IoU	Intersection over Union – Giao nhau trên hợp (độ trùng của bounding box)
ReLU	Rectified Linear Unit – Một loại hàm kích hoạt trong mạng nơ-ron
Pt / ONNX	Định dạng lưu trữ mô hình học sâu (PyTorch / Open Neural Network Exchange)
TensorFlow Lite	Thư viện mã nguồn mở dùng để xây dựng mô hình học sâu
OpenCV	Open Source Computer Vision Library – Thư viện thị giác máy tính mã nguồn mở
Colab	Google Colaboratory – Môi trường chạy Python trực tuyến của Google
VS Code	Visual Studio Code – Trình soạn thảo mã nguồn đa nền tảng của Microsoft

KÝ HIỆU

$l(x, y)$	Giá trị điểm ảnh tại vị trí (x, y) trong ảnh đầu vào
$f(x)$	Hàm kích hoạt (activation function)
$\delta(x)$	Hàm kích hoạt sigmoid
$ReLU(x)$	Hàm kích hoạt ReLU
$Conv$	Phép tích chập trong mạng CNN
W, H	Chiều rộng (Width), chiều cao (Height) của ảnh
C	Số kênh (Channels) trong ảnh màu (ví dụ: C = 3 với RGB)
N	Kích thước batch trong huấn luyện (số ảnh mỗi lần cập nhật trọng số)
Accuracy	Tỉ lệ dự đoán đúng
Precision	Tỉ lệ đúng trong số các dự đoán dương
Recall	Tỉ lệ phát hiện đúng
F1-score	Trung bình điều hòa của Precision và Recall
δ	Sai số nhỏ, dùng để tránh chia cho 0
θ	Tham số tổng quát (parameter) trong mô hình học sâu
TP	True Positive – Phát hiện đúng trường hợp có cháy
FP	False Positive – Phát hiện sai trường hợp không cháy
TN	True Negative – Dự đoán đúng trường hợp không cháy
FN	False Negative – Bỏ sót đám cháy

MỞ ĐẦU

1. Mục đích của đề tài

Trong bối cảnh các hệ thống camera giám sát ngày càng phổ biến tại các tòa nhà, khách sạn, văn phòng và khu công nghiệp, nhu cầu tận dụng dữ liệu hình ảnh để nâng cao hiệu quả phòng cháy chữa cháy trở nên cấp thiết. Đề tài này được thực hiện nhằm xây dựng một thiết bị nhúng thông minh có khả năng tự động nhận dạng khói và lửa từ hình ảnh camera theo thời gian thực, đồng thời phát cảnh báo kịp thời đến người dùng qua các kênh tin nhắn, cuộc gọi hoặc ứng dụng di động.

Thiết bị được thiết kế để có thể kết nối linh hoạt với nhiều camera IP sẵn có, xử lý đồng thời nhiều luồng hình ảnh, từ đó khắc phục những hạn chế của các cảm biến truyền thống như độ bao phủ thấp, dễ báo giả và không cung cấp thông tin hình ảnh sự cố. Thông qua việc tích hợp mô hình học sâu vào nền tảng nhúng, đề tài hướng đến mục tiêu xây dựng một giải pháp cảnh báo cháy hiệu quả, chi phí thấp và dễ triển khai thực tế, góp phần nâng cao an toàn cho người và tài sản trong môi trường đô thị hiện đại.

2. Mục tiêu đề tài

Xuất phát từ mục đích trên, đề tài hướng đến thực hiện các mục tiêu cụ thể như sau:

(i) Chế tạo thiết bị nhúng sử dụng Raspberry Pi 5 có khả năng kết nối nhiều camera IP, xử lý luồng hình ảnh theo thời gian thực để nhận dạng khói/lửa.

- Thiết bị được thiết kế nhằm tận dụng hạ tầng camera giám sát sẵn có trong các tòa nhà, văn phòng, nhà máy.
- Khả năng kết nối nhiều camera IP cho phép giám sát đồng thời nhiều khu vực, mở rộng vùng quan sát mà không cần thêm phần cứng xử lý riêng biệt.
- Việc xử lý thời gian thực đảm bảo phát hiện sớm nguy cơ cháy, hạn chế tối đa thiệt hại và rút ngắn thời gian phản ứng.

(ii) Huấn luyện và tích hợp mô hình học sâu ResNet50, tối ưu hóa để chạy được trên thiết bị nhúng.

- ResNet50 được lựa chọn nhờ vào sự cân bằng giữa độ chính xác và khả năng triển khai trên thiết bị tài nguyên thấp như Raspberry Pi.
- Mô hình được tinh chỉnh (fine-tune) trên tập dữ liệu khói/lửa thực tế nhằm nâng cao độ chính xác trong các tình huống đa dạng (ánh sáng yếu, nhiều vật cản).
- Quá trình chuyển mô hình sang định dạng tối ưu như TorchScript, ONNX giúp tăng tốc độ suy luận mà không làm suy giảm hiệu suất.

(iii) Xây dựng phần mềm cảnh báo thông minh, hỗ trợ người dùng theo dõi trực tiếp và nhận cảnh báo ngay khi có nguy cơ cháy xảy ra.

- Hệ thống phần mềm bao gồm giao diện giám sát trực tuyến, hiển thị luồng video và thông tin cảnh báo theo thời gian thực.
- Phần mềm hỗ trợ ghi log sự kiện, tạo thuận lợi cho kiểm tra và xử lý sau sự cố.

(iv) Thử nghiệm, đánh giá hiệu năng hệ thống và đề xuất hướng cải tiến phù hợp cho thực tế triển khai.

- Tiến hành đo lường tốc độ xử lý, độ trễ, độ chính xác và độ ổn định của hệ thống khi chạy ở chế độ một hoặc nhiều camera.
- Đánh giá khả năng hoạt động trong điều kiện môi trường thực tế như: nhiễu hình ảnh, rung lắc, thay đổi ánh sáng.
- Từ kết quả đánh giá, đề xuất các cải tiến kỹ thuật như sử dụng mô hình nhẹ hơn, tối ưu phần mềm, hoặc nâng cấp phần cứng nhằm đảm bảo khả năng mở rộng trong các ứng dụng quy mô lớn.

3. Phạm vi và đối tượng nghiên cứu

3.1 Đối tượng nghiên cứu

Đối tượng nghiên cứu bao gồm :

- Các loại môi trường có thể xảy ra khói, cháy và tương ứng với vật liệu đó tìm hiểu màu sắc của khói và lửa;

- Thuật toán xử lý ảnh nhận dạng khói, lửa bao gồm một số phiên bản khác nhau của mô hình YOLO và mô hình ResNet50;

- Các ngôn ngữ lập trình và giao thức truyền thông để xây dựng giao diện trên chương trình ứng dụng và gửi cảnh báo về người dùng.

3.2 Phạm vi nghiên cứu

- Nghiên cứu chế tạo thiết bị tự động với kích thước nhỏ gọn và có để ứng dụng trong mọi môi trường để có thể phát hiện khói và lửa, gửi cảnh báo về email đồng thời quan sát trên giao diện người dùng.

- Phạm vi nghiên cứu tập trung vào việc phát hiện khói và lửa trong môi trường trong nhà như tòa nhà, văn phòng, nhà xưởng có camera IP sẵn có; xử lý đồng thời 2–3 luồng camera, cảnh báo tức thời; không bao gồm hệ thống chữa cháy tự động.

4. Phương pháp nghiên cứu

Để chế tạo một thiết bị tự động thực hiện các chức năng trên cần phải giải quyết các vấn đề sau:

- Nghiên cứu lý thuyết:

+ Nghiên cứu tổng quan về công nghệ nhận dạng khói lửa bằng thị giác máy tính;

+ Thu thập và xử lý dữ liệu huấn luyện; Huấn luyện và triển khai mô hình ResNet50 trên Raspberry Pi;

- Nghiên cứu về phần mềm
    + Thuật toán Resnet50;
    + Hệ điều hành Linux;
    + Phần mền Google Colab để huấn luyện mô hình;
    + Phần mềm VS code là công cụ để viết chương trình trên máy tính
    + Thonny là công cụ để viết chương trình trên Raspberry PI
- Nghiên cứu phần cứng:
    + Camera IP;
    + Raspberry Pi 5;
    + Module sim 900A

5. Cấu trúc của đồ án tốt nghiệp

Để giải quyết bài toán đặt ra ở đây là thiết kế chế tạo thiết bị tự động nhận dạng khói & lửa trong khu vực nhất định. Tác giả xây dựng cấu trúc của bài báo cáo gồm bốn nội dung chính:

Chương 1: Tổng quan về đề tài

Chương 2: Đề xuất giải pháp nhận dạng và phát hiện khói lửa dựa và thị giác máy tính

Chương 3: Thiết kế chế tạo thiết bị tự động nhận dạng và cảnh báo khói lửa

Chương 4: Vận hành thử nghiêm và đánh giá

Chương 5: Kết luận và Hướng phát triển trong tương lai

CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI

1.1 Bối cảnh và nhu cầu thực tiễn

Cháy nổ là một trong những mối đe dọa nghiêm trọng đối với nhà ở, văn phòng, khách sạn, nhà máy và đặc biệt là các khu công nghiệp – nơi tập trung nhiều thiết bị điện, hóa chất và vật liệu dễ cháy. Theo thống kê từ Cục Cảnh sát Phòng cháy, chữa cháy và cứu nạn, cứu hộ – Bộ Công an Việt Nam, mỗi năm xảy ra trung bình hơn 3.000 vụ cháy, gây thiệt hại lớn về người và tài sản. Trong số đó, phần lớn các đám cháy đều không được phát hiện kịp thời ở giai đoạn khởi phát, khi mới xuất hiện khói hoặc ngọn lửa nhỏ, dẫn đến những hậu quả nghiêm trọng [2].

Hiện nay, trong phần lớn các vụ cháy, nguyên nhân dẫn đến thiệt hại nghiêm trọng là do phát hiện muộn hoặc thiết bị cảnh báo không hoạt động đúng cách [2]. Hầu hết các hệ thống cảnh báo cháy sử dụng cảm biến khói, khí hoặc nhiệt. Tuy nhiên, các thiết bị này vẫn còn nhiều hạn chế như:

- Dễ báo động giả trong môi trường nhiều bụi hoặc độ ẩm cao;
- Khó phát hiện cháy khi đám cháy nằm ngoài phạm vi của cảm biến;
- Việc triển khai mở rộng trong các khu vực rộng lớn gặp khó khăn nếu không có sẵn hệ thống dây dẫn tín hiệu.

Trong khi đó, tại các tòa nhà và nhà xưởng hiện đại đã được trang bị hệ thống camera giám sát IP nhằm phục vụ an ninh. Nếu có thể khai thác dữ liệu hình ảnh từ các camera này để tự động phát hiện dấu hiệu khói hoặc lửa, chúng ta không chỉ tận dụng được hạ tầng có sẵn, tiết kiệm chi phí đầu tư, mà còn nâng cao hiệu quả trong việc phát hiện sớm và cảnh báo cháy nổ một cách thông minh, kịp thời và chính xác hơn.

Hình 1.1 Vụ cháy chung cư mini ở Khương Hạ, Hà Hội 2023.

Trong thập kỷ qua, nhiều nghiên cứu đã tập trung phát triển các hệ thống phát hiện cháy dựa trên hình ảnh với sự hỗ trợ của học máy. Các phương pháp truyền thống như xử lý ảnh dựa trên ngưỡng màu sắc (color thresholding), phân tích chuyển động hoặc phát hiện vùng sáng bất thường có độ chính xác thấp và dễ bị ảnh hưởng bởi ánh sáng môi trường [3].

Việc ứng dụng học sâu đã chứng minh được hiệu quả vượt trội nhờ khả năng tự học đặc trưng phức tạp từ ảnh đầu vào, giúp tăng độ chính xác và khả năng khái quát hóa trong các môi trường khác nhau. Nhiều nghiên cứu gần đây đã đạt độ chính xác trên 90% khi sử dụng mạng nơ-ron tích chập (CNN) để phân loại hoặc phát hiện khói, lửa trên video thời gian thực [4][5].

1.2 Mục tiêu cụ thể của đề tài

Để đạt được mục đích trên, đề tài đặt ra các mục tiêu cụ thể như sau:

- Nghiên cứu và lựa chọn mô hình học sâu phù hợp cho bài toán nhận dạng khói/lửa trên ảnh camera;
- Huấn luyện mô hình ResNet50 với dữ liệu khói/lửa và triển khai mô hình trên thiết bị Raspberry Pi 5;
- Thiết kế hệ thống xử lý tích hợp camera IP truyền luồng ảnh về thiết bị nhúng;
- Xây dựng hệ thống cảnh báo đa kênh: gửi SMS, thực hiện cuộc gọi khẩn và hiển thị cảnh báo trên ứng dụng giám sát;
- Thử nghiệm và đánh giá hiệu năng hệ thống về độ chính xác, tốc độ phản hồi, khả năng xử lý đồng thời nhiều camera;
- Đề xuất hướng cải tiến để tối ưu mô hình, nâng cao khả năng nhận diện và tính mở rộng trong ứng dụng thực tế.

1.3 Hình thành và lựa chọn giải pháp

1.3.1 Phân tích lựa chọn mô hình học sâu

Trong các ứng dụng thị giác máy tính liên quan đến nhận dạng cháy, một số mô hình học sâu được sử dụng phổ biến gồm:

- VGG16/VGG19: là các mạng nơ-ron cổ điển với kiến trúc sâu và đơn giản, được ứng dụng sớm trong bài toán phân loại ảnh. Tuy nhiên, số lượng tham số lớn và yêu cầu tính toán cao khiến VGG không phù hợp với thiết bị nhúng [8].
- InceptionV3: cải tiến về tốc độ và hiệu quả với kiến trúc module “Inception”, có khả năng trích xuất đặc trưng đa cấp, nhưng cấu trúc khá phức tạp và khó triển khai trên phần cứng hạn chế [9].
- MobileNetV2: là mạng nhẹ được tối ưu cho thiết bị di động, có tốc độ nhanh nhưng độ chính xác thấp hơn trong các bài toán phức tạp như phân biệt khói/mây hoặc ánh sáng [10].

- YOLO (You Only Look Once): là mô hình phát hiện đối tượng thời gian thực, có khả năng định vị vùng cháy nhưng yêu cầu cấu hình phần cứng cao và khó triển khai đồng thời nhiều camera trên thiết bị nhúng [11].
- ResNet50: là mạng sâu có sử dụng các khối dư (residual blocks), giúp chống lại hiện tượng mất gradient và cải thiện hiệu năng huấn luyện. ResNet50 được đánh giá là cân bằng tốt giữa độ chính xác và khả năng triển khai thực tế trên Raspberry Pi hoặc Jetson Nano [6][12].

Từ các phân tích trên, đề tài lựa chọn ResNet50 làm mô hình chính do có tính ổn định, độ chính xác cao, và có thể dễ dàng tối ưu để chạy thời gian thực trên thiết bị nhúng.

1.3.2 Hình thành giải pháp hệ thống

Từ thực tế khảo sát cho thấy, phần lớn các tòa nhà, văn phòng và nhà xưởng hiện nay đều đã được trang bị hệ thống camera IP giám sát an ninh, thường hoạt động 24/7. Tuy nhiên, những camera này chủ yếu phục vụ mục đích quan sát thủ công, không có khả năng phát hiện nguy cơ cháy nổ tự động.

Trong khi đó, các hệ thống cảnh báo cháy hiện hành chủ yếu sử dụng cảm biến truyền thống như:

- Cảm biến khói quang học;
- Cảm biến nhiệt độ;
- Cảm biến khí gas.

Các thiết bị này có nhược điểm:

- Phụ thuộc vị trí lắp đặt: không phát hiện nếu cháy xảy ra ở nơi không có cảm biến;
- Dễ cảnh báo sai trong môi trường có bụi, độ ẩm, hơi nước hoặc khí nấu ăn;
- Không linh hoạt mở rộng, khó triển khai diện rộng trong tòa nhà lớn hoặc nhiều tầng;
- Không ghi lại hình ảnh sự kiện, gây khó khăn trong điều tra sau sự cố.

Từ đó, thực hiện nhận thấy có thể tận dụng chính nguồn dữ liệu video sẵn có từ hệ thống camera giám sát để:

- Phân tích và phát hiện khói/lửa theo thời gian thực bằng công nghệ thị giác máy tính;
- Kết nối linh hoạt nhiều camera IP qua mạng để mở rộng vùng giám sát mà không cần thêm cảm biến vật lý;
- Tích hợp vào thiết bị nhúng nhỏ gọn để dễ lắp đặt, tiêu thụ điện năng thấp và hoạt động ổn định liên tục;
- Kết hợp với các kênh cảnh báo nhanh như: ứng dụng di động, SMS, hoặc gọi điện đến người phụ trách.

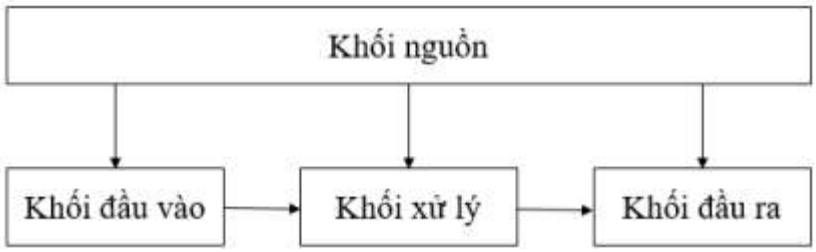
Từ các phân tích trên, hình thành ý tưởng chế tạo một thiết bị nhúng thông minh – đóng vai trò trung tâm giám sát cháy – với khả năng:

- Nhận luồng video từ nhiều camera;
- Xử lý bằng mô hình học sâu (ResNet50);
- Đưa ra cảnh báo tức thì nếu phát hiện khói hoặc lửa.

Ý tưởng này không chỉ xuất phát từ nhu cầu thực tiễn cấp bách mà còn tận dụng hiệu quả các thành phần hạ tầng sẵn có và phù hợp với xu hướng ứng dụng AI vào lĩnh vực phòng cháy chữa cháy thông minh hiện nay [1][4][6].

1.4 Định hướng thực hiện

Xuất phát từ 2 yêu cầu lớn của đề tài nghiên cứu là: nhận dạng khói lửa và cảnh báo khói lửa. Tác giả quyết định ứng dụng thị giác máy tính để nhận dạng khói lửa và đưa cảnh báo khói lửa nhận dạng được tới người dùng thông qua ứng dụng. Sơ đồ khối phần cứng của thiết bị được thể hiện như Hình



Hình 1.2 Sơ đồ phần cứng

Sơ đồ khối phần cứng của thiết bị gồm 4 khối sau:
- Khối đầu vào là khối thu thập hình ảnh khói lửa được phát hiện và đưa tín hiệu đến khối xử lý;
- Khối xử lý là khối xử lý ảnh, tính toán - xử lý và gửi tín hiệu đến khối đầu ra;
- Khối đầu ra là khối hiển thị hình ảnh khói lửa mà khối xử lý đã nhận dạng được và cảnh báo khói lửa lên ứng dụng người dùng;
- Khối nguồn là khối cung cấp điện năng cho toàn thiết bị hoạt động.

1.5 Kết luận

- Việc xây dựng thiết bị nhận dạng và cảnh báo cháy từ luồng camera IP không chỉ giải quyết bài toán kỹ thuật quan trọng trong giám sát an toàn mà còn có tiềm năng triển khai thực tế cao. Việc lựa chọn mô hình ResNet50 là có cơ sở lý luận và thực nghiệm rõ ràng, phù hợp với đặc thù của hệ thống xử lý đa camera trên nền tảng nhúng. Những nội dung tiếp theo sẽ trình bày chi tiết về giải pháp thiết kế phần cứng và phần mềm cho thiết bị cảnh báo khói lửa thông minh.

- Từ những vấn đề đặt ra, tác giả đã xác định được hướng nghiên cứu và phát triển cho đề tài: chế tạo thiết bị tự động nhận dạng và cảnh báo khói lửa sử dụng Raspberry Pi và mô hình học sâu. Qua đó, tác giả đề xuất giải pháp tổng thể và xây dựng sơ đồ khối phần cứng cho thiết bị, làm nền tảng cho các chương tiếp theo trong quá trình thiết kế và triển khai hệ thống.

CHƯƠNG 2: ĐỀ XUẤT GIẢI PHÁP NHẬN DẠNG KHÓI LỬA DỰA VÀO THỊ GIÁC MÁY TÍNH

2.1 Giới thiệu hướng tiếp cận

Như đã trình bày ở chương trước, hệ thống cảnh báo cháy truyền thống – chủ yếu dựa trên cảm biến khói, nhiệt hoặc khí gas – thường gặp nhiều hạn chế trong môi trường công nghiệp thực tế. Cụ thể, các cảm biến này có phạm vi hoạt động cố định, dễ bị ảnh hưởng bởi điều kiện bụi bẩn, độ ẩm cao hoặc thông gió mạnh; đồng thời việc triển khai mở rộng cũng gặp khó khăn tại các khu vực rộng lớn hoặc phân tầng phức tạp.

Trước những thách thức đó, sự phát triển mạnh mẽ của công nghệ thị giác máy tính (Computer Vision) kết hợp với các mô hình học sâu (Deep Learning) hiện đại đã mở ra một hướng tiếp cận mới, tiềm năng hơn: nhận dạng trực tiếp khói và lửa từ hình ảnh thu được qua camera giám sát. So với cảm biến vật lý, hình ảnh từ camera cung cấp thông tin phong phú hơn, bao quát toàn cảnh và có thể giám sát liên tục theo thời gian thực.

Đề tài này đề xuất một giải pháp mới, ứng dụng mô hình học sâu ResNet50 để phát hiện khói và lửa từ luồng video thu được bởi các camera IP sẵn có trong khu vực nhà xưởng hoặc khu công nghiệp. ResNet50 là một trong những kiến trúc mạng tích chập (Convolutional Neural Network – CNN) nổi bật, có khả năng trích xuất đặc trưng hình ảnh sâu, nhận diện chính xác ngay cả trong điều kiện ánh sáng hoặc môi trường thay đổi.

Mô hình được triển khai trên thiết bị nhúng Raspberry Pi 5, đóng vai trò là trung tâm xử lý. Raspberry Pi 5 sở hữu hiệu năng cao trong phân khúc vi điều khiển, đủ khả năng chạy mô hình học sâu ở tốc độ chấp nhận được trong thời gian thực. Thiết bị này sẽ xử lý hình ảnh từ camera, phân tích và nhận diện khói/lửa, sau đó gửi cảnh báo qua mạng (Wi-Fi/LAN) đến người dùng thông qua các kênh như điện thoại, máy tính hoặc hệ thống cảnh báo nội bộ (báo động còi, đèn nháy).

Ưu điểm chính của hướng tiếp cận này bao gồm:

- Phát hiện sớm và chính xác các đám cháy ngay từ khi xuất hiện khói/lửa nhỏ;
- Tận dụng hạ tầng camera có sẵn, tiết kiệm chi phí đầu tư mới;
- Dễ dàng mở rộng hệ thống mà không phụ thuộc vào dây tín hiệu hay cảm biến cố định;
- Tính linh hoạt cao nhờ thiết bị xử lý nhỏ gọn, có thể lắp đặt ở nhiều vị trí khác nhau.

Nhờ kết hợp giữa công nghệ hiện đại và thiết bị nhúng chi phí thấp, giải pháp mang tính ứng dụng thực tế cao, phù hợp với yêu cầu an toàn trong các khu công nghiệp hiện nay, đặc biệt là trong bối cảnh chuyển đổi số và xây dựng hệ thống nhà máy thông minh.

2.2 Cơ sở lý thuyết về nhận dạng ảnh bằng học sâu

2.2.1 Thị giác máy tính và mạng nơ-ron tích chập

Một trong những lĩnh vực quan trọng của trí tuệ nhân tạo (Artificial Intelligence) là thị giác máy tính (Computer Vision). Thị giác máy tính là một lĩnh vực bao gồm các phương pháp thu nhận, xử lý ảnh kỹ thuật số, phân tích và nhận dạng các hình ảnh, phát hiện các đối tượng, tạo ảnh, siêu phân giải hình ảnh và nhiều hơn vậy. Trong đó, bài toán phát hiện đối tượng (Object Detection) là khía cạnh sâu sắc nhất của thị giác máy tính do tính chất hiệu quả có khả năng triển khai trong thực tế.

Object Detection đề cập đến khả năng của hệ thống máy tính và phần mềm để định vị các đối tượng trong một hình ảnh và xác định từng đối tượng. Object Detection đã được sử dụng rộng rãi để phát hiện khuôn mặt, phát hiện xe, đếm số người đi bộ, hệ thống bảo mật, xe không người lái và các hệ thống phục vụ cho mục đích cảnh báo sớm từ xa. Object Detection đã bắt đầu sử dụng các phương pháp nhận dạng đối tượng hiện đại trong các ứng dụng và hệ thống, cũng như xây dựng các ứng dụng mới dựa trên các phương pháp này. Việc nhận dạng đối tượng sớm liên quan đến việc sử dụng các thuật toán cổ điển, giống như các thuật toán được hỗ trợ trong Open CV - thư viện Computer Vision phổ biến nhất hiện nay. Tuy nhiên, các thuật toán cổ điển này không thể đạt được hiệu suất đủ để làm việc trong các điều kiện khác nhau.

Việc áp dụng đột phá và nhanh chóng của việc học sâu (deep learning) vào năm 2012 đã đưa ra các thuật toán và phương pháp phát hiện đối tượng hiện đại, chính xác cao như: R - CNN, Fast-RCNN, Faster-RCNN, RetinaNet và nhanh hơn mà rất chính xác như SSD và YOLO. Sử dụng các phương pháp và thuật toán này, dựa trên deep learning và dựa trên việc học máy (machine learning) đòi hỏi rất nhiều kiến thức về toán học.

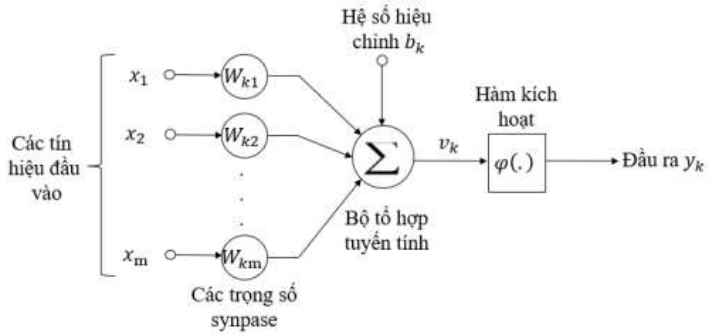
2.2.2 Mạng Neural

2.2.2.1 Khái niệm

Mạng neural nhân tạo (Artificial Neural Network) là mô hình xử lý thông tin theo cách thức xử lý thông tin của các hệ neural sinh học. Nó được tạo nên từ một số lượng lớn các phần tử kết nối với nhau thông qua các liên kết (gọi là trọng số liên kết) để cùng làm việc và giải quyết các bài toán đặt ra.

2.2.2.2 Các thành phần của mạng neural

Cấu trúc neural nhân tạo :



Hình 2.1 Mô hình tổng quát mạng neural

Các thành phần cơ bản của một neural nhân tạo bao gồm:

- Đầu vào: là các tín hiệu vào của neural, tín hiệu này thường được đưa vào dưới dạng một vector m chiều;

- Trọng số liên kết: mỗi liên kết được thể hiện bởi một trọng số (Synaptic weight). Trọng số liên kết giữa tín hiệu vào thứ j với neural k. Các trọng số này được khởi tạo một cách ngẫu nhiên ở điểm khởi tạo và được cập nhật liên tục trong quá trình huấn luyện;

- Bộ tổng (Summing function): tính tổng của tích các đầu vào với trọng số liên kết;

- Ngưỡng (một độ lệch - bias): được đưa vào như một thành phần của hàm truyền;

- Hàm kích hoạt (Activation function): hàm này được dùng để giới hạn phạm vi đầu ra của mỗi neural. Đầu vào của nó là kết quả của hàm tổng và ngưỡng đã cho. Đầu ra của mỗi neural giới hạn trong đoạn [0, 1] hoặc [-1, 1]. Các hàm truyền rất đa dạng được liệt kê trong Bảng 2.1. Người thiết kế mạng lựa chọn hàm truyền tuỳ thuộc vào từng bài toán và kinh nghiệm;

- Đầu ra: neural nhân tạo nhận tín hiệu đầu vào, xử lý và cho một tín hiệu đầu ra là kết quả của hàm truyền.

Bảng 2-1 Bảng một số hàm kích hoạt

STT	**Tên hàm**	Phương trình
1	Binary step	$f(x) = \begin{cases} 0, x < 0 \\ 1, x \geq 0 \end{cases}$
2	Logistic	$f(x) = \frac{1}{1 + e^{-x}}$
3	Softsign	$f(x) = \frac{1}{1 + \lvert x \rvert}$
4	Rectified linear unit (ReLU)	$f(x) = \begin{cases} 0, x < 0 \\ x, x \geq 0 \end{cases}$
5	Leaky rectified linear unit (Leaky ReLU)	$f(x) = \begin{cases} 0.01, x < 0 \\ x, x \geq 0 \end{cases}$
6	Parameteric rectified linear unit (PReLU)	$f(x) = \begin{cases} ax, x < 0 \\ x, x \geq 0 \end{cases}$
7	Exponential linear unit (ELU)	$f(a, x) = \begin{cases} a(e^x - 1), x < 0 \\ x, x \geq 0 \end{cases}$
8	Scaled exponential linear unit (SELU)	$f(a, x) = \gamma \begin{cases} a(e^x - 1), x < 0 \\ x, x \geq 0 \end{cases}, với\ i\ \gamma = 1.0507$ $f(x) = \max(0, x) + \sum_{s=1}^{s} (a_i{}^s \max(0, -x + b_i{}^s))$

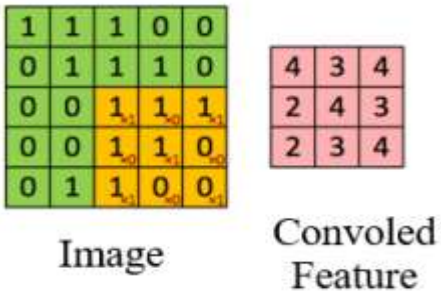
9	S-shaped rectified linear activation unit (SReLU)	$f(a,x)=\begin{cases} t_l + a_1(x-t_l), x \le t_l \\ x, t_l \le x \le t_r \\ t_r + a_l(x-t_r), x \ge t_r \end{cases}$
10	Adaptive Piecewise linear (APL)	$f(x)=\max(0,x)+\sum_{s=1}^{S}(a_i{}^s \max(0,-x+b_i{}^s))$
11	SoftPlus	$f(x)=\ln(1+e^x)$
12	Bent identity	$f(x)=\frac{\sqrt{(x^2+1)}-1}{2}+x$
13	SoftExponential	$f(a,x)=\begin{cases} -\frac{\ln(1-a(x+a))}{a}, a<0 \\ x, a=0 \\ \frac{e^{-ax}-1}{a}+a, a>0 \end{cases}$
14	Sinusoid	$f(x)=\sin(x)$
15	Sinc	$f(x)=\begin{cases} 1, x=0 \\ \frac{\sin(x)}{x}, x \ne 0 \end{cases}$
16	Gaussian	$f(x)=e^{-x^2}$

2.2.2.3 Mạng neural tính chập

A. Tổng quan về mạng neural tích chập

Convolutional Neural Network (CNN – Mạng neural tích chập) là mô hình Deep Learning. Nó xây dựng được những hệ thống có độ chính xác cao. CNN thường được sử dụng trong các bài toán phát hiện đối tượng (Object Detection).

Trong đó, Convolution được sử dụng trong xử lý tín hiệu số dựa vào nguyên lý biến đổi thông tin. Có thể xem Convolution như một cửa sổ trượt (sliding window) lên một ma trận.

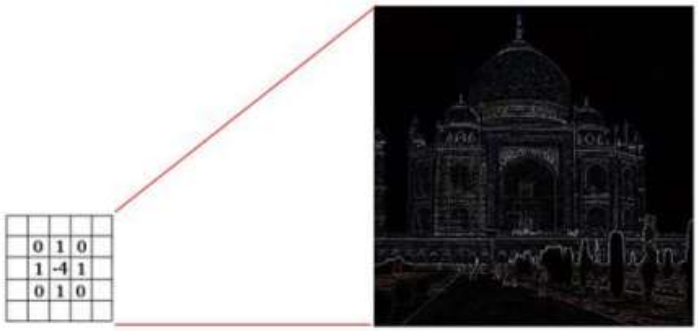


Hình 2.2 Hình ảnh minh họa Convolution

Convolutional layer có các thông số (parameter) được huấn luyện (training) để tự điều chỉnh lấy ra những thông tin chính xác nhất.

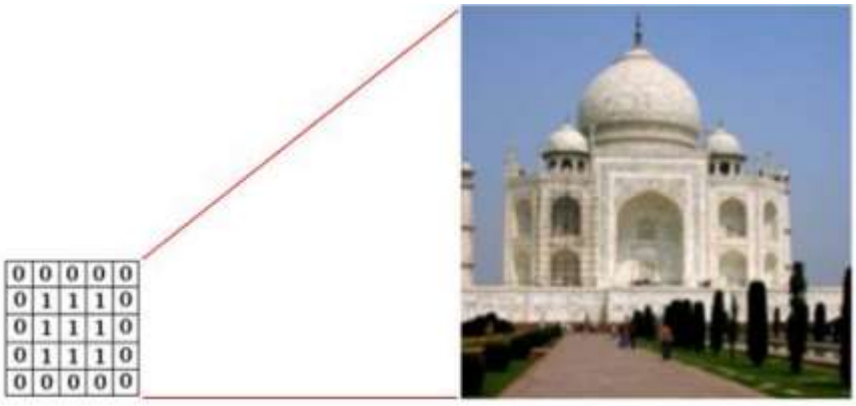
Sliding window được gọi là nhận (kernel) hoặc bộ lọc (filter). Theo ví dụ minh hoạ Hình 2.2, ta dùng một ma trận filter có dạng 3×3 nhân từng thành phần tương ứng với ma trận ảnh bên trái. Giá trị đầu ra do tích của các thành phần này cộng lại. Kết quả là một ma trận convoled feature sinh ra từ việc trượt ma trận filter và thực hiện tích chập cùng lúc lên toàn bộ ma trận ảnh bên trái. Từ đó, khi thực hiện Convolution ta có thể:

- Phát hiện được biên cạnh bằng cách tính vi phân giữa các điểm ảnh lân cận. Hình 2.3 là minh họa việc phát hiện biên;



Hình 2.3 Hình ảnh minh họa việc phát hiện biên

- Làm mờ bức ảnh nếu lấy giá trị trung bình của các điểm ảnh xung quanh cho vị trí điểm ảnh trung tâm. Hình 2.4 là minh họa cho việc làm mờ bức ảnh.



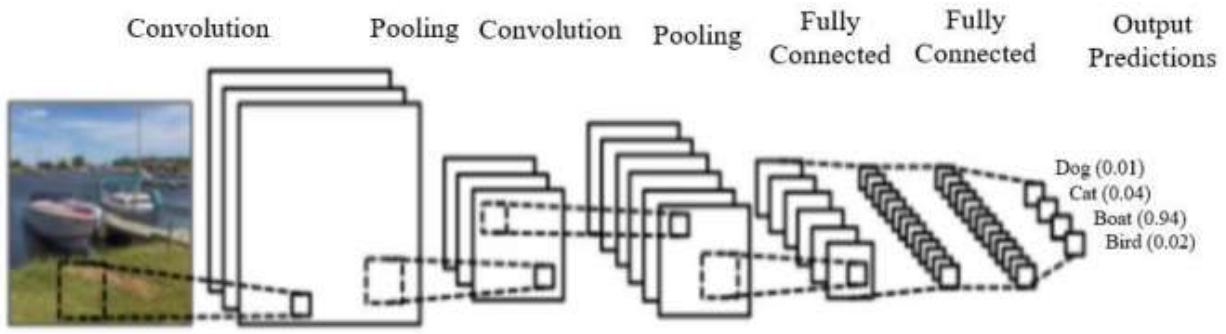
Hình 2.4 Hình ảnh minh họa việc làm mờ bức ảnh

B. Kiến trúc mạng CNN

Mạng CNN là một tập hợp của nhiều lớp Convolution chồng lên nhau và dùng các hàm kích hoạt để kích hoạt các trọng số. Các lớp liên kết với nhau thông qua cơ chế convolution (kết quả convolution từ lớp trước đó là đầu vào của lớp tiếp theo).

Mỗi lớp được sử dụng các filter khác nhau, có rất nhiều filter và kết hợp kết quả của chúng lại. Bên cạnh đó, một số lớp khác như lớp pooling hoặc lớp **upsample** dùng để lựa chọn lại các thông tin hữu ích và giảm các thông tin nhiễu.

Trong lúc training, CNN sẽ tự học các giá trị qua các lớp filter. Ví dụ trong tác vụ phân lớp ảnh, CNN sẽ cố gắng tìm ra thông số tối ưu cho các filter tương ứng theo thứ tự sau: **raw Pixel > edges > shapes > facial > high - level features**. Layer cuối cùng sẽ dùng để phân lớp ảnh



Hình 2.5 Cấu trúc mạng CNN cơ bản

Hình 2.5 biểu diễn cấu trúc mạng CNN cơ bản, trong đó bao gồm:

- Input: Lớp đầu vào;

- Convolution: Lớp tích chập;

- ReLu: Lớp biến đổi thông qua hàm kích hoạt ReLu để kích hoạt phi tuyến;

- Pooling: Lớp tổng hợp, thông thường là Max pooling hoặc có thể là Average pooling dùng để giảm chiều của ma trận đầu vào;

- Fully Connected: Lớp kết nối hoàn toàn, thông thường lớp này nằm ở sau cùng và kết nối với các đơn vị đại diện cho nhóm phân loại.

a) Lớp tích chập

CNN là một mạng neural được hình thành từ nhiều lớp khác nhau. Tuy nhiên, lớp tích chập (convolutional layer) được xem là có vai trò quan trọng nhất và sử dụng thuật toán convolution. Đối với lớp tích chập thì có những khái niệm cần nắm như sau:

- Filter map: CNN sử dụng các filter để áp dụng vào vùng của hình ảnh. Những filter map này được gọi là ma trận 3 chiều, mà bên trong nó là các con số và chúng là thông số (parameter);

- Stride: Khi dịch chuyển filter map theo pixel dựa vào giá trị từ trái sang phải, sự dịch chuyển này là stride;

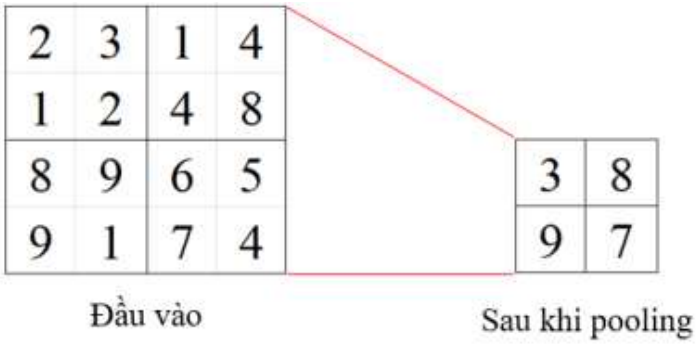
- Padding: Được hiểu là những giá trị 0 được thêm vào lớp input. Khi điều chỉnh padding = 1, tức là thêm 1 ô bọc xung quanh các cạnh input, các ô bọc này sẽ dày lên nếu ta tăng padding lên;

- Feature map: Là kết quả hiển thị sau mỗi lầ filter map được duyệt qua input, cứ mỗi lần quét như vậy chính là quá trình tính toán;

- Receptive field : Bộ filters sẽ trượt qua từng vị trí trên bức ảnh để tính tích chậpgiữa bộ filter và phần tương ứng trên bức ảnh. Phần tương ứng này trên bức ảnh gọi là receptive field.

b) Lớp pooling

Sau hàm kích hoạt sẽ sử dụng lớp pooling, một số loại lớp pooling phổ biến như là max pooling, average pooling với chức năng là giảm chiều của lớp trước đó. Với một lớp pooling có kích thước 2x2, cần phải trượt filter 2x2 này trên những vùng ảnh có kích thước tương tự rồi sau đó tính max hay average cho vùng ảnh đó.



Hình 2.6 Max pooling kernel 2x2, stride = 2

Trong đó, max pooling chọn giá trị lớn nhất làm giá trị đầu ra còn average pooling thì chọn giá trị trung bình cộng. Trong Hình 2.6, ta sử dụng filter với kernel size = (2, 2) và S = 2 để tính giá trị pooling.

Hầu hết khi dùng lớp pooling thì sẽ dùng kernel size = (2, 2), S = 2 và P = 0. Mục đích của tầng pooling là giảm thiểu ảnh hưởng của những đặc trưng trong không gian ảnh không còn cần thiết. Hơn nữa, lớp pooling có khả năng giảm chiều, làm hạn chế tràn (overfit) và giảm thời gian huấn luyện mô hình (training model).

c) Lớp fully connected

Tầng cuối cùng của mô hình CNN trong bài toán phân loại ảnh là tầng fully connected. Chức năng của lớp fully connected là chuyển ma trận đặc trưng ở tầng trước thành vector chứa xác suất của các đối tượng cần được dự đoán.

Sau khi ảnh được truyền qua nhiều lớp convolution và lớp pooling thì mô hình đã học được tương đối các đặc điểm của ảnh và output của lớp cuối cùng sẽ được là phẳng thành vector và đưa vào một lớp được kết nối như một mạng neural (lớp fully connected). Lớp fully connected kết hợp các tính năng lại với nhau để tạo ra một mô hình và cuối cùng sử dụng hàm kích hoạt softmax hoặc sigmoid để phân loại đầu ra.

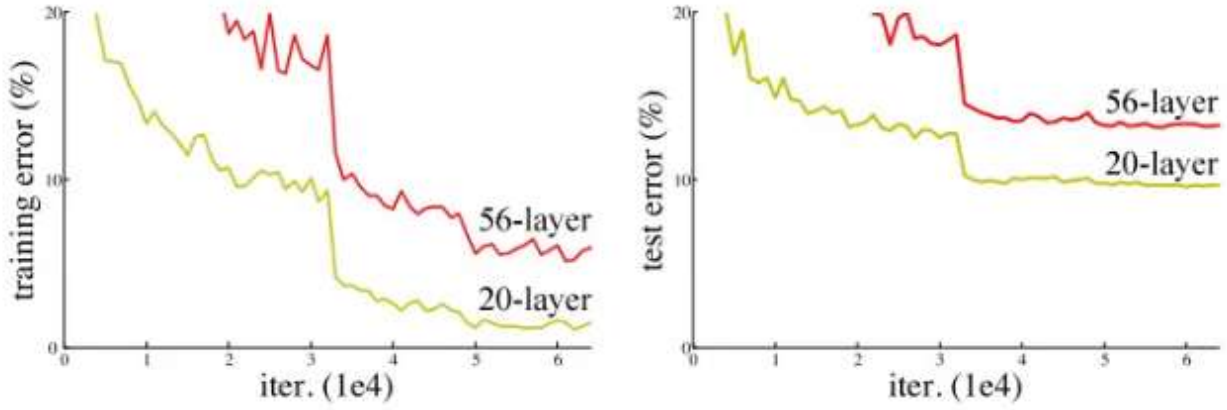
C. Mạng ResNet

a) Giới thiệu

Mạng ResNet (R) là một mạng CNN được thiết kế để làm việc với hàng trăm lớp. Một vấn đề xảy ra khi xây dựng mạng CNN với nhiều lớp chập sẽ xảy ra hiện tượng Vanishing Gradient dẫn tới quá trình học tập không tốt.

Vanishing Gradient

Trước hết thì Backpropagation Algorithm là một kỹ thuật thường được sử dụng trong quá trình tranining. Ý tưởng chung của thuật toán lá sẽ đi từ output layer đến input layer và tính toán gradient của cost function tương ứng cho từng parameter (weight) của mạng. Gradient Descent sau đó được sử dụng để cập nhật các parameter đó.



Hình 2.7 Hình ảnh minh họa cho hiện tượng degradation problem

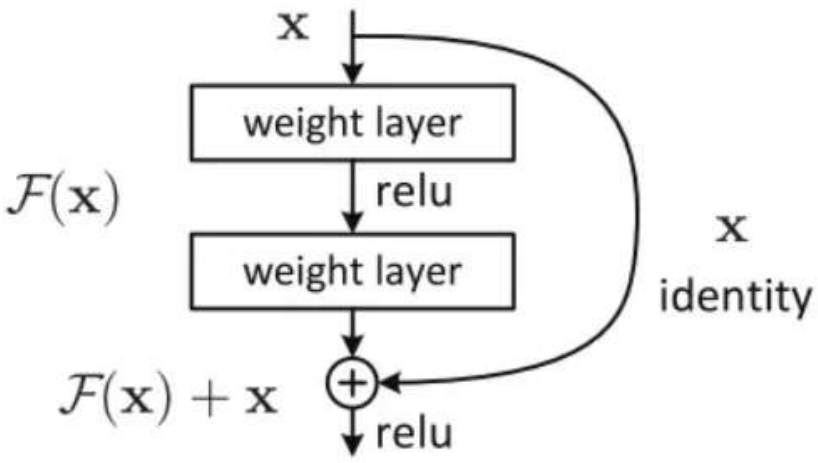
Toàn bộ quá trình trên sẽ được lặp đi lặp lại cho tới khi mà các parameter của network được hội tụ. Thông thường chúng ta sẽ có một hyperparametr (số Epoch - số lần mà traninig set được duyệt qua một lần và weights được cập nhật) định nghĩa cho số lượng vòng lặp để thực hiện quá trình này. Nếu số lượng vòng lặp quá nhỏ thì ta gặp phải trường hợp mạng có thể sẽ không cho ra kết quả tốt và ngược lại thời gian tranining sẽ lâu nếu số lượng vòng lặp quá lớn.

Tuy nhiên, trong thực tế Gradients thường sẽ có giá trị nhỏ dần khi đi xuống các layer thấp hơn. Dẫn đến kết quả là các cập nhật thực hiện bởi Gradients Descent không làm thay đổi nhiều weights của các layer đó và làm chúng không thể hội tụ và mạng sẽ không thu được kết quả tốt. Hiện tượng như vậy gọi là Vanishing Gradients.

- Mạng ResNet ra đời cũng giải quyết vấn đề đó.

b) Kiến trúc mạng ResNet

ResNet gần như tương tự với các mạng gồm có convolution, pooling, activation và fully-connected layer. Ảnh bên dưới hiển thị khối dư được sử dụng trong mạng. Xuất hiện một mũi tên cong xuất phát từ đầu và kết thúc tại cuối khối dư. Hay nói cách khác là sẽ bổ sung Input X vào đầu ra của layer, hay chính là phép cộng mà ta thấy trong hình minh họa, việc này sẽ chống lại việc đạo hàm bằng 0, do vẫn còn cộng thêm X.



Hình 2.8 Residual Block

Với H(x) là giá trị dự đoán, F(x) là giá trị thật (nhãn), chúng ta muốn H(x) bằng hoặc xấp xỉ F(x). Việc F(x) có được từ x như sau:

X->weight1-> ReLU -> weight2

Giá trị H(x) có được bằng cách:

F(x) + x -> ReLU

Như chúng ta đã biết việc tăng số lượng các lớp trong mạng làm giảm độ chính xác, nhưng muốn có một kiến trúc mạng sâu hơn có thể hoạt động tốt.

2.3 Mô hình ResNet50

2.3.1 Lý do lựa chọn ResNet50

Mô hình ResNet50 được lựa chọn vì các lý do sau:

- Đạt độ chính xác cao trong các nghiên cứu về phát hiện khói lửa, với F1-score thường trên 92% [19].
- Có thể được tối ưu hóa để chạy trên Raspberry Pi 5 thông qua công cụ như TensorRT hoặc ONNX Runtime [20].
- Phù hợp với bài toán phân loại khung hình thành hai lớp: "có cháy" và "bình thường"giúp giảm độ phức tạp so với các mô hình định vị đối tượng như YOLO.
- Khả năng triển khai thời gian thực: sau huấn luyện trên Google Colab hoặc PC, mô hình có thể suy luận trên ảnh độ phân giải 640x480 với tốc độ khoảng 5–10 FPS trên Raspberry Pi [21].

2.3.2 Giới thiệu ResNet50

ResNet-50 là một mạng nơ-ron tích chập (Convolutional Neural Network - CNN) thuộc họ ResNet (Residual Networks), được thiết kế để giải quyết các thách thức liên quan đến việc

huấn luyện các mạng nơ-ron sâu. Được phát triển bởi các nhà nghiên cứu tại Microsoft Research Asia, ResNet-50 nổi tiếng với độ sâu và hiệu quả trong các tác vụ phân loại hình ảnh.

2.3.3 Ứng dụng của ResNet-50

ResNet-50 đã được áp dụng rộng rãi trong các tác vụ thị giác máy tính, đặc biệt là phân loại hình ảnh và nhận dạng đối tượng. Khả năng học các đặc trưng phức tạp từ dữ liệu hình ảnh đã làm cho ResNet-50 trở thành lựa chọn phổ biến trong nhiều ứng dụng thực tế. Ngoài ra, ResNet-50 thường được sử dụng làm mô hình tiền huấn luyện trong học chuyển giao (transfer learning), giúp cải thiện hiệu suất của các mô hình trong các tác vụ cụ thể với dữ liệu hạn chế.

Tóm lại, ResNet-50 là một kiến trúc mạng nơ-ron sâu hiệu quả, giải quyết được các vấn đề liên quan đến việc huấn luyện các mạng sâu và đã đóng góp quan trọng trong lĩnh vực thị giác máy tính.

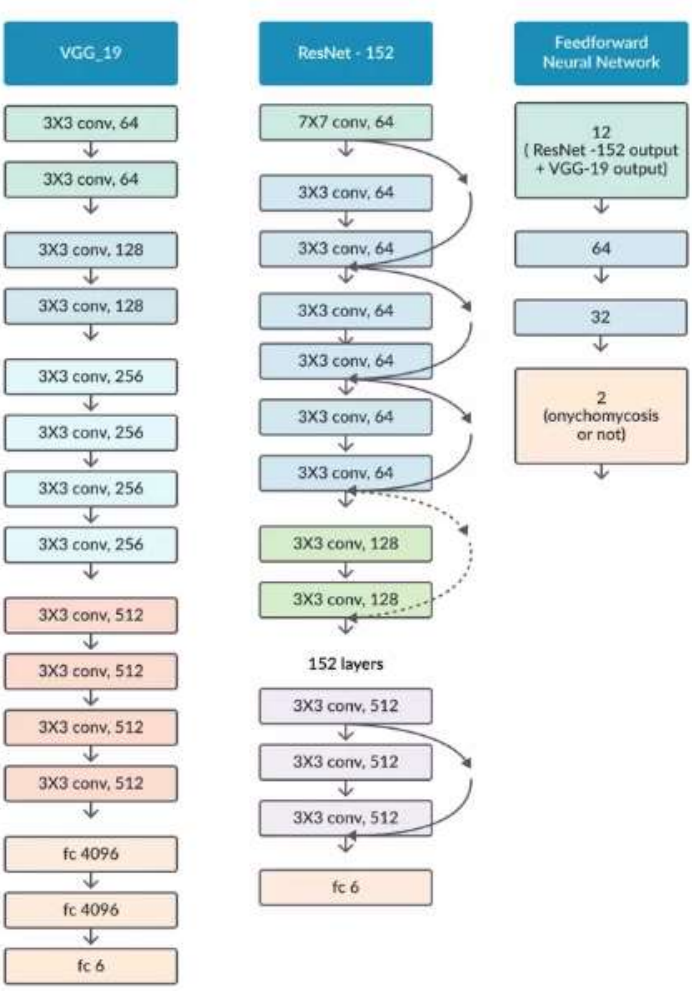
layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Hình 2.9 Cấu trúc Resnet50

Hình 1. VGG-19 là một mô hình CNN sử dụng kernel 3x3 trên toàn bộ mạng, VGG-19 cũng đã giành được ILSVRC năm 2014.

Hình 2. ResNet sử dụng các kết nối tắt (kết nối trực tiếp đầu vào của lớp (n) với (n+x) được hiển thị dạng mũi tên cong. Qua mô hình nó chứng minh được có thể cải thiện hiệu suất trong quá trình training model khi mô hình có hơn 20 lớp.

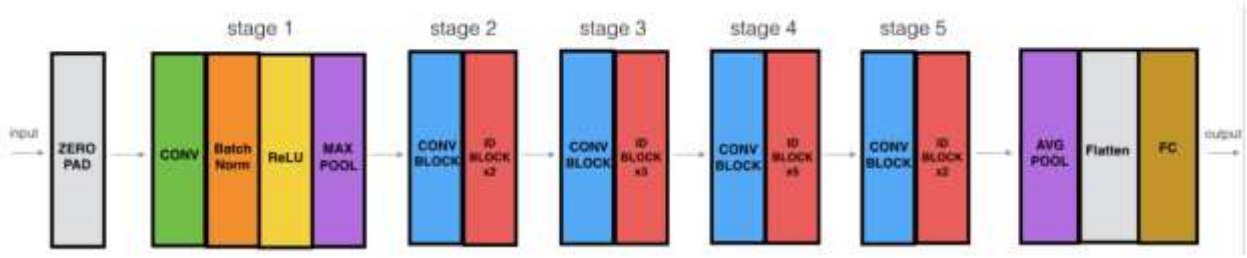
Hình 3. Tổng cộng có 12 đầu ra từ ResNet-152 và VGG-19 đã được sử dụng làm đầu vào cho mạng có 2 lớp hidden. Đầu ra cuối cùng được tính toán thông qua hai lớp ẩn (hidden). Việc xếp chồng các lớp sẽ không làm giảm hiệu suất mạng.Với kiến trúc này các lớp phía trên có được thông tin trực tiếp hơn từ các lớp dưới nên sẽ điều chỉnh trọng số hiệu quả hơn.



Hình 2.10 Các mô hình CNN

2.3.4 Xây dựng mạng ResNet-50

Hình dưới đây mô tả chi tiết kiến trúc mạng nơ ron ResNet :



Hình 2.11 Hình ảnh mô tả kiến trúc mạng nơ ron Resnet

"ID BLOCK" trong hình trên là viết tắt của từ Identity block và ID BLOCK x3 nghĩa là có 3 khối Identity block chồng lên nhau. Nội dung hình trên như sau :

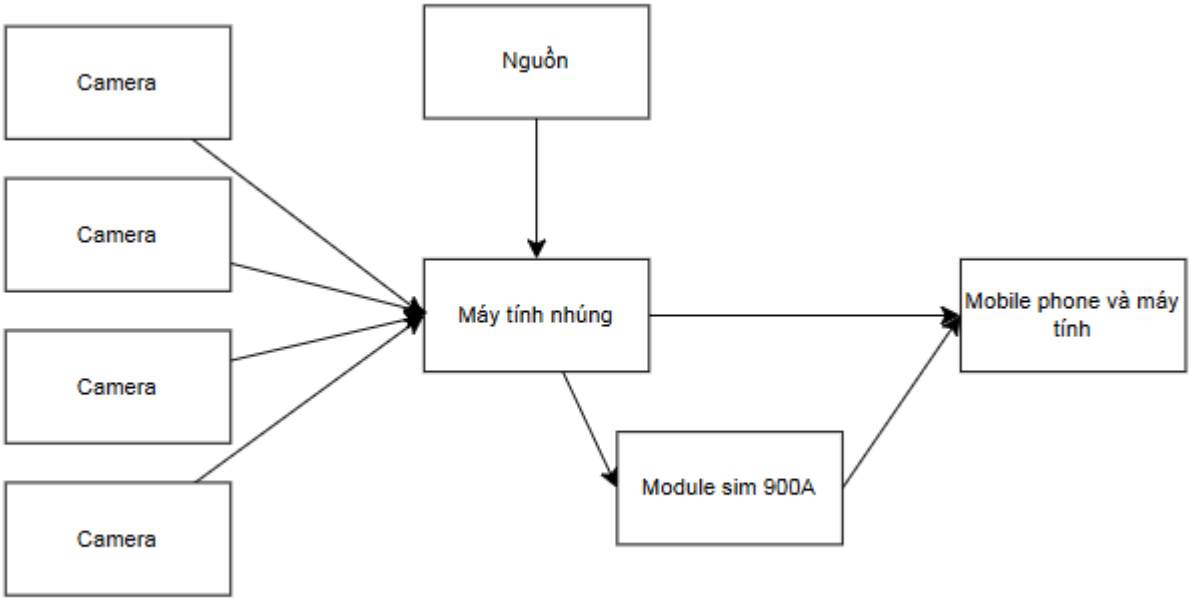
- Zero-padding : Input với (3,3)

- Stage 1 : Tích chập (Conv1) với 64 filters với shape(7,7), sử dụng stride (2,2). BatchNorm, MaxPooling (3,3).
- Stage 2 : Convolutiontal block sử dụng 3 filter với size 64x64x256, f=3, s=1. Có 2 Identity blocks với filter size 64x64x256, f=3.
- Stage 3 : Convolutional sử dụng 3 filter size 128x128x512, f=3,s=2. Có 3 Identity blocks với filter size 128x128x512, f=3.
- Stage 4 : Convolutional sử dụng 3 filter size 256x256x1024, f=3,s=2. Có 5 Identity blocks với filter size 256x256x1024, f=3.
- Stage 5 :Convolutional sử dụng 3 filter size 512x512x2048, f=3,s=2. Có 2 Identity blocks với filter size 512x512x2048, f=3.
- The 2D Average Pooling : sử dụng với kích thước (2,2).
- The Flatten.
- Fully Connected (Dense) : sử dụng softmax activation.

2.4 Đề xuất giải pháp

2.4.1 Phương án thiết kế phần cứng

Dựa vào những nghiên cứu chung về việc nhận dạng khói lửa bằng thị giác máy tính và cảnh báo qua mạng internet tới người dùng, tác giả thiết kế ra mô hình phần ucnwgs của thiết bị như Hình:



Hình 2.12 Sơ đồ thiết kế phần cứng

Sơ đồ khối thiết kế phần cứng bao gồm các thiết bị điện chính sau:

- Camera IP: Thu nhận hình ảnh từ khu vực giám sát và truyền video về bộ xử lý trung tâm. Được lắp đặt tại các vị trí trọng yếu trong nhà xưởng để quan sát liên tục 24/7.
- Raspberry Pi 5: Đóng vai trò là máy tính nhúng xử lý trung tâm, tiếp nhận hình ảnh từ camera và thực hiện nhận dạng khói/lửa bằng mô hình học sâu ResNet50. Thiết bị có hiệu năng cao, nhỏ gọn, tiêu thụ điện thấp và dễ dàng mở rộng kết nối.
- Điện thoại di động: Nhận cảnh báo cháy thông qua SMS, cuộc gọi hoặc thông báo từ các nền tảng trực tuyến. Đồng thời, người dùng có thể giám sát hình ảnh trực tiếp qua địa chỉ IP của camera.
- Module SIM900A: Được kết nối với Raspberry Pi để gửi tin nhắn SMS hoặc thực hiện cuộc gọi tự động khi phát hiện sự cố cháy, đảm bảo khả năng cảnh báo ngay cả khi mất kết nối Internet.
- Nguồn điện: Cung cấp năng lượng cho toàn bộ hệ thống, có thể kết hợp với bộ lưu điện (UPS) để đảm bảo hoạt động liên tục.

Ngoài ra, sau khi hoàn thiện thiết kế phần cứng và các thiết bị điện tử, tác giả tiến hành thiết kế phần cơ khí, vỏ bảo vệ và bố trí lắp đặt các thành phần một cách hợp lý để đảm bảo thiết bị vận hành an toàn, bền bỉ trong điều kiện thực tế như nhà xưởng hoặc khu vực dễ xảy ra cháy nổ.

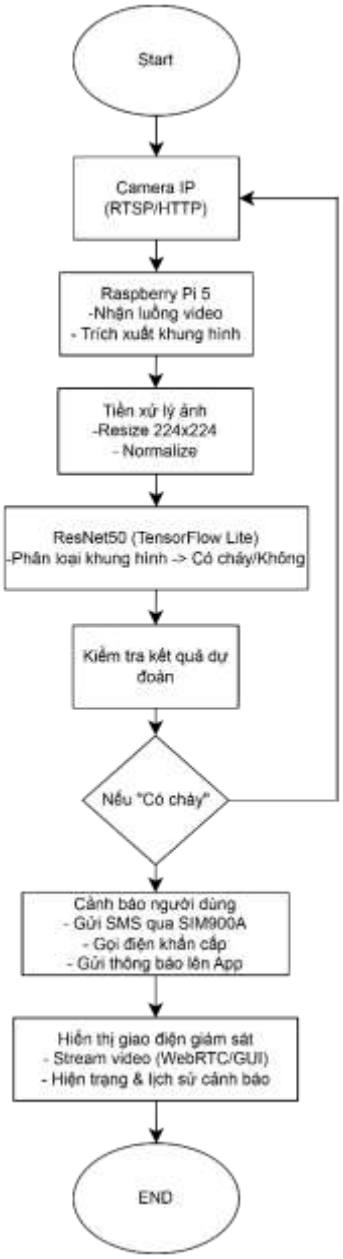
2.4.2 Phương án thiết kế phần mền

Phần mềm điều khiển hệ thống gồm các thành phần chính:

- Nhận dữ liệu video từ camera IP qua RTSP;
- Tiền xử lý ảnh: resize, chuẩn hóa trước khi đưa vào mô hình;
- Suy luận bằng mô hình ResNet50;
- Gửi cảnh báo đến người dùng khi phát hiện khói/lửa;
- Giao diện giám sát: hiển thị luồng video và trạng thái cảnh báo.

Luồng xử lý được lập trình bằng Python, sử dụng các thư viện như:

- OpenCV (xử lý ảnh),
- VS code (chạy mô hình),
- Firebase (lưu trữ và ltruyền dữ liệu cảnh báo),
- Thonny IDE để triển khai trên Raspberry Pi.



Hình 2.13 Sơ đồ luồng xử lý hệ thống nhận dạng và cảnh báo khói lửa

Hệ thống nhận dạng và cảnh báo khói lửa được xây dựng theo mô hình xử lý tuần tự, từ việc thu nhận hình ảnh đầu vào đến khi đưa ra cảnh báo cho người dùng. Các bước chính được mô tả chi tiết như sau:

Bước 1: Thu nhận dữ liệu từ camera IP

Hệ thống sử dụng một hoặc nhiều camera IP lắp đặt tại các vị trí cần giám sát. Các camera này truyền luồng video liên tục về thiết bị xử lý trung tâm thông qua giao thức RTSP hoặc HTTP. Ưu điểm của việc sử dụng camera IP là có thể dễ dàng mở rộng hệ thống và tận dụng cơ sở hạ tầng sẵn có trong các tòa nhà, văn phòng, hoặc nhà xưởng.

Bước 2: Trích xuất khung hình và tiền xử lý

Raspberry Pi 5, đóng vai trò là bộ điều khiển nhúng trung tâm, sẽ thu nhận luồng video và tiến hành trích xuất các khung hình (frame) định kỳ. Mỗi khung hình được xử lý sơ bộ thông qua các bước tiền xử lý như thay đổi kích thước (resize về 224x224 pixels) và chuẩn hóa giá trị điểm ảnh (normalize), đảm bảo định dạng đầu vào phù hợp với mô hình học sâu ResNet50.

Bước 3: Phân loại hình ảnh bằng mô hình ResNet50

Sau khi tiền xử lý, ảnh đầu vào sẽ được đưa vào mô hình học sâu ResNet50 đã được huấn luyện trước để phân loại. Mô hình sẽ xác định xem khung hình có chứa dấu hiệu khói/lửa hay không. Đây là bước quan trọng nhất, nơi mà các đặc trưng hình ảnh được học sâu trong mạng nơ-ron giúp hệ thống nhận biết chính xác tình huống cháy trong nhiều điều kiện môi trường khác nhau.

Bước 4: Kiểm tra kết quả và ra quyết định

Dựa vào kết quả phân loại, hệ thống sẽ đánh giá nếu có sự xuất hiện của khói hoặc lửa thì thực hiện các hành động cảnh báo tương ứng. Trường hợp không có dấu hiệu bất thường, hệ thống tiếp tục quay lại vòng lặp giám sát với các khung hình mới.

Bước 5: Kích hoạt cảnh báo

Khi phát hiện khói/lửa, hệ thống sẽ đồng thời kích hoạt ba kênh cảnh báo:

- Gửi tin nhắn SMS và thực hiện cuộc gọi đến số điện thoại người phụ trách qua module SIM900A.
- Gửi thông báo đẩy (push notification) đến ứng dụng giám sát trên điện thoại di động thông qua dịch vụ Firebase Cloud Messaging (FCM).
- Ghi nhận thời điểm và hình ảnh sự kiện để lưu trữ trong hệ thống quản lý.

Bước 6: Hiển thị và giám sát trên giao diện người dùng

Người dùng có thể giám sát tình trạng khu vực thông qua giao diện đồ họa hiển thị trên màn hình kết nối với Raspberry Pi hoặc qua ứng dụng di động. Giao diện cung cấp hình ảnh camera theo thời gian thực, thông tin về vị trí camera đang cảnh báo và lịch sử các sự kiện cháy đã phát hiện. Việc này giúp người vận hành kịp thời đưa ra phản ứng phù hợp hoặc xác minh sự cố.

2.4.2.1 Công cụ xây dựng hệ thống

a) Roboflow

Roboflow là là một công cụ trực tuyến miễn phí để dán nhãn ảnh. Nhờ sử dụng trình duyệt, nó không yêu cầu bất kỳ cài đặt phức tạp nào, chỉ cần truy cập trang web và máy tính sẵn sàng làm việc. Máy tính đang chạy trên hệ điều hành nào cũng không quan trọng. Nó hoàn hảo cho các dự án học sâu về thị giác máy tính nhỏ, giúp quá trình chuẩn bị tập dữ liệu dễ dàng và nhanh hơn nhiều. Có thể tải xuống các nhãn đã chuẩn bị ở một trong nhiều định dạng được hỗ trợ

b) Google Colab

Google Colab là một sản phẩm được nghiên cứu và phát triển bởi Google Research. Nó cho phép xây dựng chương trình Python và biên dịch thông qua hình thức trình duyệt, nó hiệu quả và phù hợp với Data analysis.

Bảng 2-2 Bảng cấu hình phần cứng Google Colab cung cấp

CPU	GPU	TPU
Intel Xeon Processor with two cores @2.30 GHz and 13GB Ram	Up to Tesla K80 with 12 GB of GDDR5 VRAM, Intel Xeon Processor with two cores @2.30 GHz and 13GB Ram	Cloud TPU with 180 teraflops of computation, Intel Xeon Processor with two cores @2.30 GHz and 13GB Ram

c) Visual Studio Code

Visual Studio Code (VS Code) là một trình soạn thảo mã nguồn miễn phí và mã nguồn mở, được sử dụng phổ biến để lập trình đa ngôn ngữ, trong đó có Python. VS Code cung cấp nhiều tính năng mạnh mẽ như tô màu cú pháp, gợi ý mã thông minh (IntelliSense), tích hợp trình gỡ lỗi, kiểm soát phiên bản Git và một kho tiện ích mở rộng phong phú giúp mở rộng chức năng cho các ngôn ngữ và công cụ khác nhau. VS Code được phát triển bởi Microsoft và hoạt động đa nền tảng trên Windows, Mac OS và Linux. Thiết kế nhẹ, khả năng tùy biến cao và hỗ trợ đa dạng tiện ích mở rộng

d) Thư viện Open CV

Project Open CV được bắt đầu từ Intel năm 1999 bởi Gary Bradsky. OpenCV viết tắt cho Open Source Computer Vision Library. OpenCV là thư viện nguồn mở hàng đầu cho Computer Vision và Machine Learning và hiện có thêm tính năng tăng tốc GPU cho các hoạt động theo thời gian thực. OpenCV được phát hành theo giấy phép BSD, do đó nó miễn phí cho cả học tập và sử dụng với mục đích thương mại. Nó có trên các giao diện C, C++, Python, Java và hỗ trợ Windows, Linux, Mac OS, iOS và Android. OpenCV được thiết kế để hỗ trợ hiệu quả về tính toán và chuyên dùng cho các ứng dụng thời gian thực. Nếu được viết trên C/C++ tối ưu, thư viện này có thể tận dụng được bộ xử lý đa lõi (multi-core processing). Các tính năng và các modul phổ biến của Open CV như sau:

- Xử lý và hiển thị Hình ảnh/ Video/ I/O (core, imgproc, highgui);
- Phát hiện các vật thể (objdetect, features2d, nonfree);
- Geometry-based monocular hoặc stereo computer vision (calib3d, stitching, videostab);

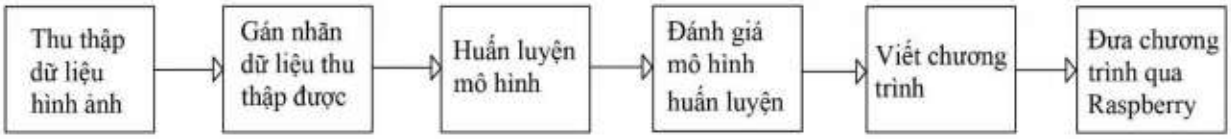
- Computational photography (photo, video, superres);
- Machine learning & clustering (ml, flann);
- CUDA acceleration (gpu).

e) Thonny IDE python

Thonny IDE python là một môi trường phát triển tích hợp cho Python được thiết kế cho người mới bắt đầu. Nó được tạo ra bởi Aivar Annamaa, một lập trình viên người Estonia. Nó hỗ trợ các cách khác nhau để duyệt mã, đánh giá biểu thức từng bước, trực quan hóa chi tiết ngăn xếp cuộc gọi và chế độ giải thích các khái niệm về tham chiếu và đống.

2.4.2.2. Kết luận phương án thiết kế phần mềm

Sau khi nghiên cứu và tìm hiểu các công cụ, thư viện và nền tảng phần mềm tác giả đưa ra phương án thiết kế phần mềm như sau:



Hình 2.14 Phương án thiết kế phần mền

Trong Hình 2.14 là các bước thiết kế phần mềm cho bài toán nhận dạng khói lửa như sau:

- Tiến hành thu thập hình ảnh dữ liệu chụp được từ điện thoại, máy ảnh hoặc có thể tìm kiếm trên internet;
- Khi đã thu thập đầy đủ dữ liệu hình ảnh của mô hình tác giả tiến hành gán nhãn của từng loại bằng công cụ Roboflow;
- Thực hiện huấn luyện mô hình trên Google Colab khi gán nhãn xong;
- Khi huấn luyện mô hình, tác giả lưu các kết quả thu được vào đồ thị để thực hiện bước đánh giá mô hình;
- Lấy file huấn luyện được đưa vào VS Code để viết chương trình nhận dạng khói lửa trên laptop đồng thời kiểm nghiệm độ chính xác mô hình rồi đưa vào Raspberry Pi;
- Đưa chương trình xuống Raspberry Pi, chương trình viết trên công cụ Thonny IDE

2.5 Kết luận

Ở chương 2, dựa trên cơ sở lý thuyết nghiên cứu tác giả xây dựng phương án thiết kế phần cứng cũng như phương án thiết kế phần mềm đối với đề tài.Tác giả, lựa chọn mô hình Resnet50 đối với loại bài toán Object Detection và đưa ra các công cụ để huấn luyện mô hình.

CHƯƠNG 3: THIẾT KẾ CHẾ TẠO THIẾT BỊ TỰ ĐỘNG NHẬN DẠNG VÀ CẢNH BÁO KHÓI LỬA

3.1 Mục tiêu thiết kế

Hệ thống được thiết kế nhằm phát hiện sớm và chính xác các hiện tượng khói hoặc lửa xảy ra tại nhiều khu vực giám sát, với mục tiêu giảm thiểu thiệt hại về tài sản và con người, đồng thời hỗ trợ kịp thời công tác phòng cháy chữa cháy.

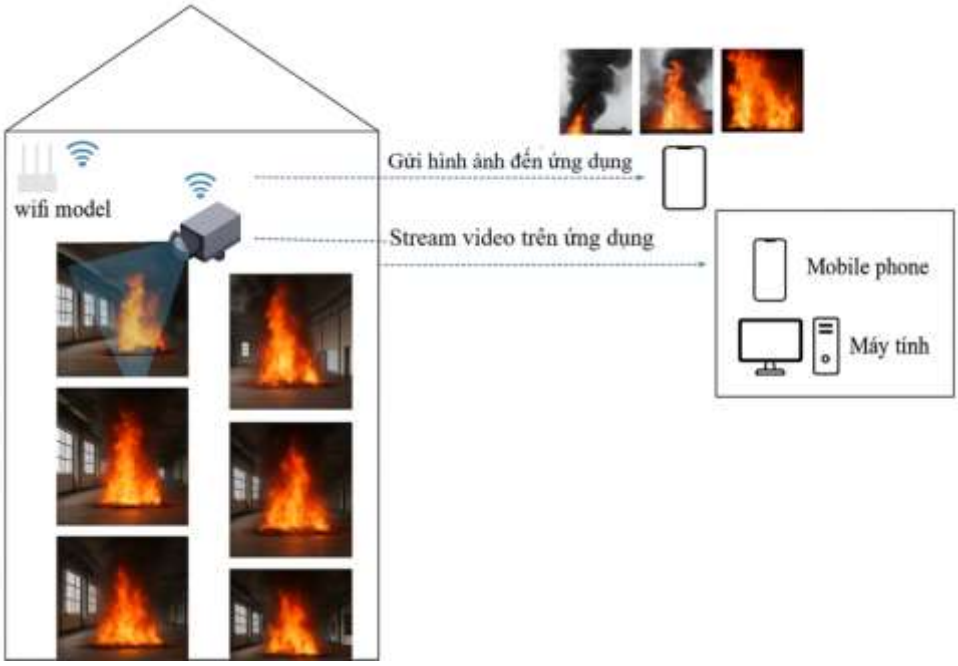
Các mục tiêu cụ thể bao gồm:

1. Tận dụng hạ tầng sẵn có:
   - Sử dụng camera IP hiện có, hỗ trợ các chuẩn truyền video như RTSP/HTTP, để tránh chi phí triển khai mới.
   - Hệ thống không yêu cầu thay đổi lớn về hạ tầng giám sát.
2. Xử lý ảnh thời gian thực trên thiết bị nhúng:
- Dữ liệu hình ảnh từ camera sẽ được trích xuất và xử lý trực tiếp bởi các thiết bị nhúng.
   - Ứng dụng mô hình học sâu ResNet50 (TensorFlow Lite) để phân loại ảnh theo 3 lớp: "Có lửa" “Có khói” và "Không cháy".
   - Tiến hành tiền xử lý ảnh (resize, normalize) để đảm bảo tương thích và tối ưu hiệu suất dự đoán.
3. Phát cảnh báo nhanh chóng qua nhiều kênh:
   - Gửi tin nhắn SMS thông báo vị trí và thời điểm xảy ra cháy thông qua mô-đun SIM900A.
   - Tự động gọi điện đến người phụ trách để đảm bảo nhận được thông báo khẩn cấp.
4. Giao diện giám sát thời gian thực:
   - Cung cấp giao diện Web/GUI để người dùng theo dõi hình ảnh trực tiếp từ camera.
   - Hiển thị trạng thái hệ thống, danh sách cảnh báo đã xảy ra và ảnh minh chứng đi kèm.
   - Hỗ trợ xem lại lịch sử cảnh báo và trạng thái xử lý.
5. Tối ưu hiệu suất và dễ triển khai:
   - Hệ thống được tối ưu để tiêu thụ điện năng thấp, hoạt động ổn định 24/7 trên thiết bị nhúng.
   - Thiết kế mô-đun linh hoạt, dễ mở rộng, dễ lắp đặt tại nhiều điểm giám sát như: nhà kho, phòng điện, hành lang chung cư, xưởng sản xuất...

3.2 Thiết kế phần cứng

3.2.1 Sơ đồ nguyên lý hệ thống

Nguyên lý hoạt động của hệ thống ứng dụng trong nhà xưởng, văn phòng, tòa nhà được thể hiện như Hình 3.2.



Hình 3.1 Mô hình tổng thể hệ thống làm việc trong nhà xưởng, văn phòng

Trong Hình 3.1, camera IP của thiết bị thực hiện thu nhận hình ảnh tại khu vực cần giám sát và truyền dữ liệu trực tiếp đến Raspberry Pi 5 thông qua mạng Wi-Fi nội bộ. Tại đây, hình ảnh được xử lý bằng mô hình học sâu ResNet50 để nhận dạng sự xuất hiện của khói hoặc lửa. Khi phát hiện có dấu hiệu bất thường, hệ thống lập tức gửi cảnh báo đến người dùng thông qua tin nhắn SMS và thực hiện cuộc gọi tự động bằng module SIM900A.

3.2.2 Lựa chọn thiết bị điện tử

3.2.2.1 Lựa chọn camera

Hình 3.2 Camera IP hỗ trợ theo dõi

Camera IP có những thông số kỹ thuật sau:

Thiết bị thu hình chính của hệ thống, có nhiệm vụ ghi nhận hình ảnh từ môi trường để gửi về máy tính nhúng xử lý. Hệ thống có thể sử dụng bất kỳ loại camera IP nào hỗ trợ giao thức RTSP hoặc HTTP streaming và có độ phân giải phù hợp.
Yêu cầu kỹ thuật tối thiểu:

- Độ phân giải: tối thiểu 1280x720 (HD)
- Giao thứ hỗ trợ: RTSP hoặc HTTP
- Khả năng kết nối: LAN hoặc Wi-Fi nội bộ
- Tốc độ khung hình: từ 15 FPS trở lên

Hệ thống không giới hạn loại hoặc hãng camera cụ thể, chỉ cần camera đáp ứng các tiêu chí trên để đảm bảo khả năng tích hợp và truyền dữ liệu ổn định đến Raspberry Pi 5 phục vụ cho quá trình nhận dạng khói/lửa.

3.2.2.2 Lựa chọn router

Hình 3.3 Router

Thông số kỹ thuật

- Chuẩn Wifi : Wi-Fi 6 (802.11ax)
- Độ mạnh của sóng (các thiết bị mạng) : 5 GHz: 1201 Mb/giây (802.11ax) và 2,4 GHz: 300 Mb/giây (802.11n)
- Bằng tần sóng : 2.4GHz & 5GHz
- Độ phủ sóng : 15 – 20m
- Cổng kết nối : 1 × Cổng WAN Gigabit và 3 × Cổng LAN Gigabit
- Hãng sản xuất : TP Link

3.2.2.3 Lựa chọn máy tính nhúng

Raspberry Pi 5 là một máy tính nhúng siêu nhỏ gọn, tích hợp đầy đủ các thành phần cần thiết để hoạt động như một máy tính độc lập. Thiết bị có hiệu năng cao, phù hợp cho các ứng dụng thị giác máy tính và xử lý học sâu trong thời gian thực.

Hình 3.4 Hình ảnh Raspberry Pi 5

Raspberry Pi 5 với thông số kỹ thuật chính:

- Bộ xử lý: Broadcom BCM2712, CPU 4 nhân ARM Cortex-A76 @ 2.4 GHz
- GPU: VideoCore VII
- RAM: 8 GB LPDDR4X
- Lưu trữ: khe cắm thẻ microSD, hỗ trợ SSD qua cổng USB 3.0
- Kết nối: 2 cổng USB 3.0, 2 cổng USB 2.0, Gigabit Ethernet, WiFi 802.11ac, Bluetooth 5.0
- Cổng hiển thị: 2 cổng micro-HDMI (hỗ trợ độ phân giải 4K)
- Hệ điều hành: Raspberry Pi OS hoặc hệ điều hành khác tương thích ARM64

Raspberry Pi 5 được lựa chọn trong đề tài vì có hiệu năng đủ mạnh để xử lý mô hình học sâu như ResNet50 phục vụ cho nhiệm vụ nhận dạng và cảnh báo khói lửa theo thời gian thực.

3.2.2.4 Lựa chọn nguồn điện

Nhằm đảm bảo hệ thống hoạt động liên tục, tăng độ tin cậy và tính an toàn cho hệ thống để hệ thống hoạt động 24/7, không được phép ngắt quảng ngay cả khi mất điện

Hình 3.5 Pin dự phòng

Nguồn Pin dự phòng AVA như Hình với các thông số kĩ thuật sau:

- Hiệu suất sạc: 64%;
- Dung lượng Pin:10.000 mAh;
- Nguồn vào: Micro USB: 5V - 2A;
- Nguồn ra: USB: 5V - 2.4A;
- Lõi Pin: Polymer;
- Công nghệ/Tiện ích: Đèn LED báo hiệu;
- Trọng lượng 182 g và kích thước: Dài 9 cm - Ngang 5.9 cm - Dày 2.56 cm.

3.2.2.5 Lựa chọn module truyền thông qua SMS

Hình 3.6 Module SIM900A

Thông số kỹ thuật module SIM900A

- Băng tần kép 900/1900 MHz
- Trên bo mạch có hai bộ giao tiếp cấp nguồn VCC5 5V, cấp nguồn VCC4 3,5V-4,5V, tùy chọn tự khởi động (mặc định) và khởi động điều khiển.
- Onboard SMA (mặc định) và giao diện ăng-ten IPXmini, đặt lại giao diện SIM900A.
- Đầu SMA cái,bao gồm ăng-ten.
- Tiêu thụ điện năng thấp: 1.5mA

- Điều khiển thông qua các lệnh AT (Lệnh GSM 07,07, 07,05 và SIMCOM được tăng cường)
- Giao diện cấp độ UART TTL (3.3V hoặc 5V) và RS232
- Khe cắm thẻ SIM (Lật)
- Nhiệt độ hoạt động: -40°C đến +85°C
- Kích thước: 49 x 50mm

3.2.2.6 Lựa chọn mạch chuyển đổi tín hiệu

Hình 3.7 Mạch chuyển USB UART TTL FT232RL

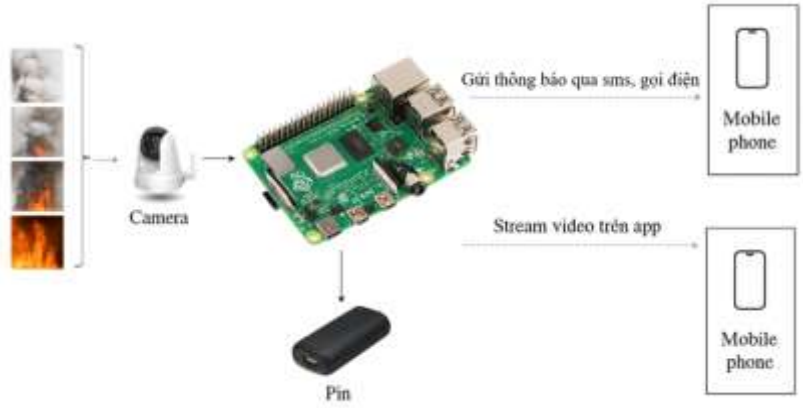
Thông số kỹ thuật:

– Có ngõ ra nguồn có thể điều chỉnh 3V3 hoặc 5VDC
– Chuyển giao tiếp từ USB sang UART TTL
– Drive hỗ trợ Windows Mac, Linux
– Có cầu chì tự phục hồi: 500mA
– Tốc độ Baudrate: tùy chỉnh

3.3 Xây dựng chương trình nhận dạng và cảnh báo khói lửa

3.3.1 Nguyên lý hoạt động của thiết bị

Nguyên lý hoạt động của thiết bị tự động nhận dạng và cảnh báo khói lửa như Hình

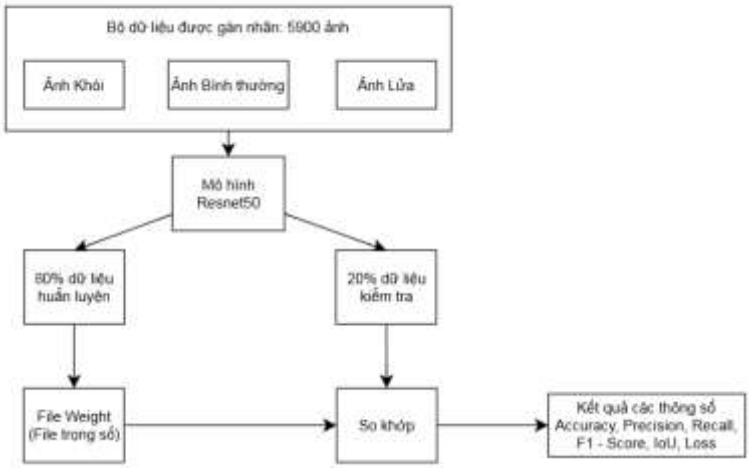


Hình 3.8 Sơ đồ cấu trúc các thiết bị phần cứng của hệ thống

Trong Hình 3.7, đường mũi tên nét đứt màu xanh là đường truyền tín hiệu kết nối không dây và đường mũi tên màu đen là đường truyền tín hiệu kết nối có dây. Camera thu nhận tín hiệu hình ảnh khói lửa và truyền dữ liệu trực tiếp đến Raspberry Pi 5. Tại đây, dữ liệu về hình ảnh được nhận dạng ra khói lửa đã xác định và gửi cảnh báo qua điện thoại người dùng. Ngoài ra, video trực tiếp cũng được phát trên App cho người giám sát có thể truy cập xem trực tiếp. Để duy trì mọi hoạt động của thiết bị tự động nhận dạng và cảnh báo khói lửa cần có một nguồn Pin dự phòng để cấp điện cho Rasppberry Pi 5 khi có sự cố mất điện.

3.3.2 Chương trình nhận dạng khói lửa

3.3.2.1 Huấn luyện và đánh giá mô hình Resnet50



Hình 3.9 Quá trình huấn luyện và đánh giá mô hình

Trong Hình là minh họa bộ dữ liệu được sử dụng trong quá trình huấn luyện và kiểm tra mô hình nhận dạng khói lửa. Bộ dữ liệu bao gồm 5.900 ảnh thuộc 2 lớp: "khói/lửa" và "bình thường", được thu thập từ nhiều nguồn khác nhau như camera giám sát, internet và các cảnh quay thực tế, với độ phân giải và điều kiện ánh sáng khác nhau. Tất cả ảnh đều đã được gán nhãn và chia thành hai phần: 80% dùng để huấn luyện mô hình, 20% còn lại dùng để kiểm tra và đánh giá độ chính xác.

Mô hình học sâu ResNet50 được huấn luyện trên tập dữ liệu này để nhận dạng sự xuất hiện của khói hoặc lửa. Sau khi huấn luyện, mô hình tạo ra một file trọng số (weight file) chứa các tham số học được. Tập dữ liệu kiểm tra được sử dụng để đánh giá hiệu quả của mô hình đã huấn luyện. Các chỉ số đánh giá như độ chính xác (accuracy), độ nhạy (recall), độ đặc hiệu (precision), F1-score và độ mất mát (loss) sẽ được tự động tính toán để kiểm chứng chất lượng của mô hình ResNet50 trong việc phát hiện khói và lửa.

Mô hình ResNet50 đã được huấn luyện trước trên bộ dữ liệu có chứa ảnh khói/lửa và ảnh bình thường. Mô hình được xuất ra dưới định dạng .pt hoặc .onnx để triển khai trên Raspberry Pi. Sau khi huấn luyện, mô hình đạt độ chính xác 96.5% và F1-score 0.93.

3.3.3 Chương trình cảnh báo khói lửa

3.3.3.1 Thiết kế giao diện người dùng

a) Tổng quan về Python

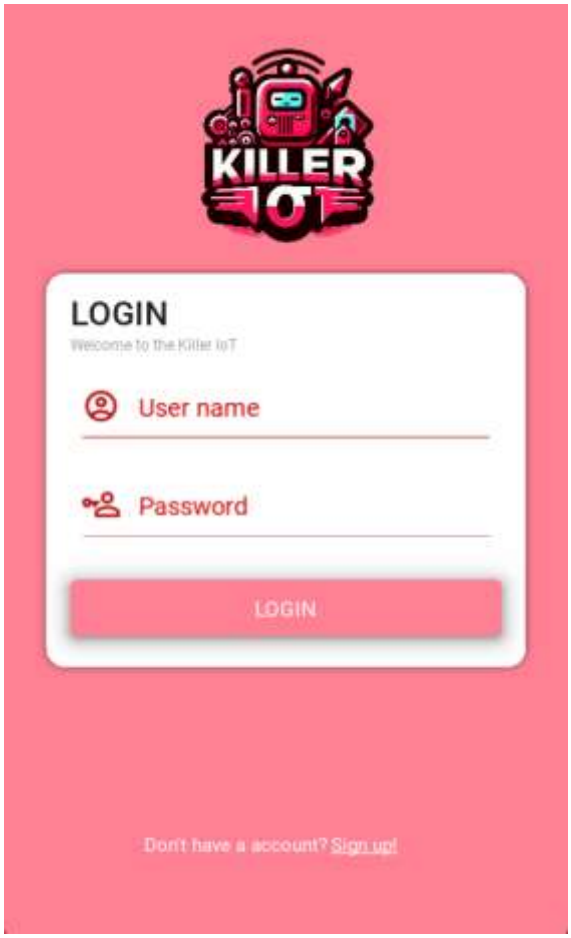
Python là ngôn ngữ lập trình bậc cao được phát triển bởi Guido van Rossum và ra mắt lần đầu vào năm 1991. Python là ngôn ngữ lập trình, được thiết kế nhằm viết ra các chương trình với cú pháp đơn giản, dễ đọc và dễ hiểu. Python hỗ trợ nhiều mục đích sử dụng như phát triển web, xử lý dữ liệu, trí tuệ nhân tạo, tự động hóa, v.v., với cú pháp rõ ràng giúp trình thông dịch hiểu và thực thi mã lệnh một cách hiệu quả.

b) Giao diện người dùng

Để giúp người dùng dễ dàng theo dõi trạng thái hệ thống và phản ứng kịp thời khi có sự cố xảy ra, đề tài xây dựng một giao diện ứng dụng đơn giản, trực quan sử dụng ngôn ngữ lập trình Python kết hợp với thư viện Kivy – một công cụ mã nguồn mở hỗ trợ thiết kế giao diện đồ họa đa nền tảng.

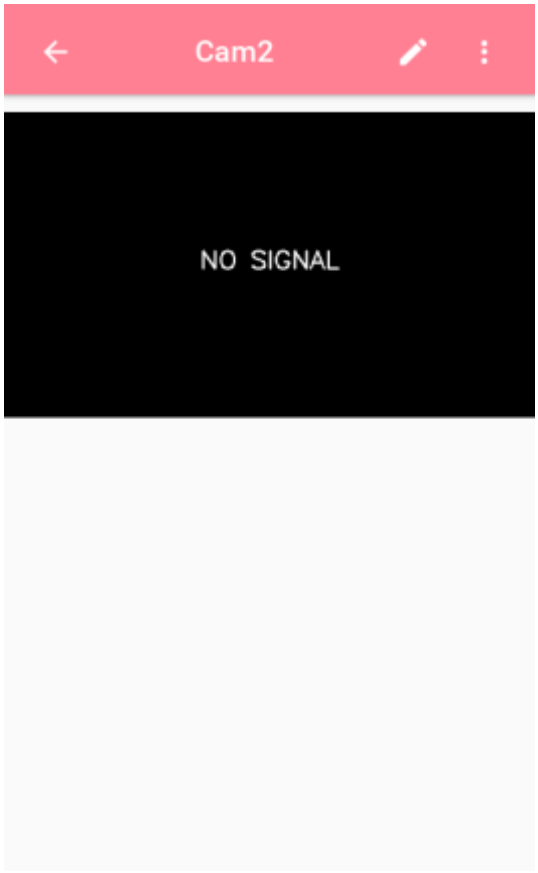
Giao diện đăng nhập

- Mục đích chính là đảm bảo tính bảo mật cho hệ thống, chỉ cho phép người dùng được cấp quyền truy cập vào ứng dụng.
- Yêu cầu nhập tên đăng nhập và mật khẩu để truy cập các chức năng chính.



Hình 3.10 Giao diện đăng nhập tài khoản

Hình 3.11 Màn hình hiển thị stream trên ứng dụng nếu có tính hiệu từ cam



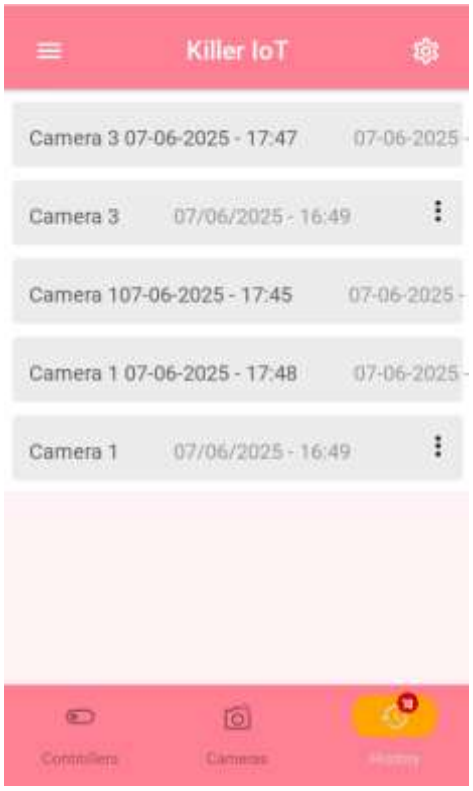
Hình 3.12 Màn hình hiển thị stream trên ứng dụng nếu không tính hiệu từ cam

Hiển thị video thời gian thực từ camera IP

- Giao diện chính hiển thị luồng hình ảnh trực tiếp từ camera IP đặt tại khu vực cần giám sát.
- Hình ảnh được cập nhật liên tục, hỗ trợ người dùng phát hiện sớm các dấu hiệu khói hoặc lửa.
- Giao diện đơn giản, dễ thao tác, phù hợp với môi trường công nghiệp.

Hiển thị trạng thái hệ thống

- Giao diện cung cấp thông tin cập nhật về tình trạng hoạt động của hệ thống, bao gồm:
  + Kết nối với camera IP (đang hoạt động / mất kết nối);
  + Trạng thái xử lý mô hình nhận dạng;
  + Tình trạng kết nối mạng và các module cảnh báo.



Hình 3.13 Lịch sử phát hiện khói lửa

Hình 3.14 Hình ảnh xảy ra đám cháy và thời gian phát hiện cảnh báo

Lịch sử các cảnh báo

- Là nơi lưu trữ và hiển thị toàn bộ các cảnh báo đã xảy ra.

Mỗi cảnh báo bao gồm:

+ Thời điểm phát hiện khói/lửa;

+ Ảnh snapshot của khung hình tại thời điểm phát hiện bất thường.

- Các thông tin này được trình bày rõ ràng trong danh sách, giúp người dùng dễ dàng tra cứu và đánh giá mức độ nghiêm trọng của các sự cố đã qua.

Cảnh báo khẩn cấp qua SMS và cuộc gọi

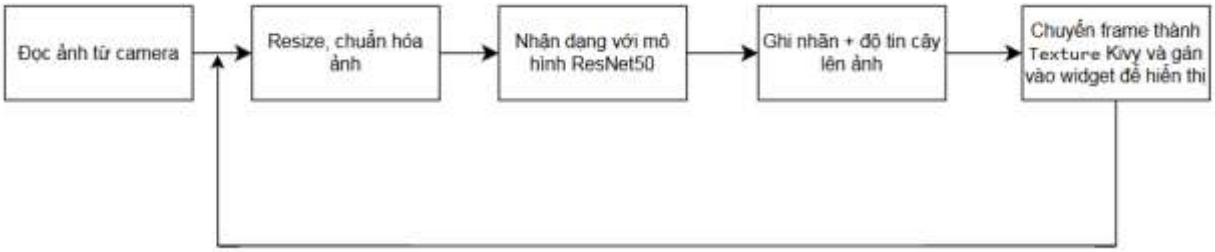
- Khi phát hiện sự cố, hệ thống sẽ kích hoạt các phương thức cảnh báo khẩn cấp:

+ Gửi tin nhắn SMS với nội dung cảnh báo đến số điện thoại của người phụ trách;

+ Thực hiện cuộc gọi tự động nhằm đảm bảo người nhận được thông báo ngay cả khi không có kết nối mạng hoặc không theo dõi ứng dụng.

c) Streaming Video lên app

Cách thứ streaming video lên app như Hình :



Hình 3.15 Cách thức streaming video lên ứng dụng

Giao diện người dùng được thiết kế bằng thư viện Kivy, hiển thị trực tiếp trên màn hình của thiết bị.

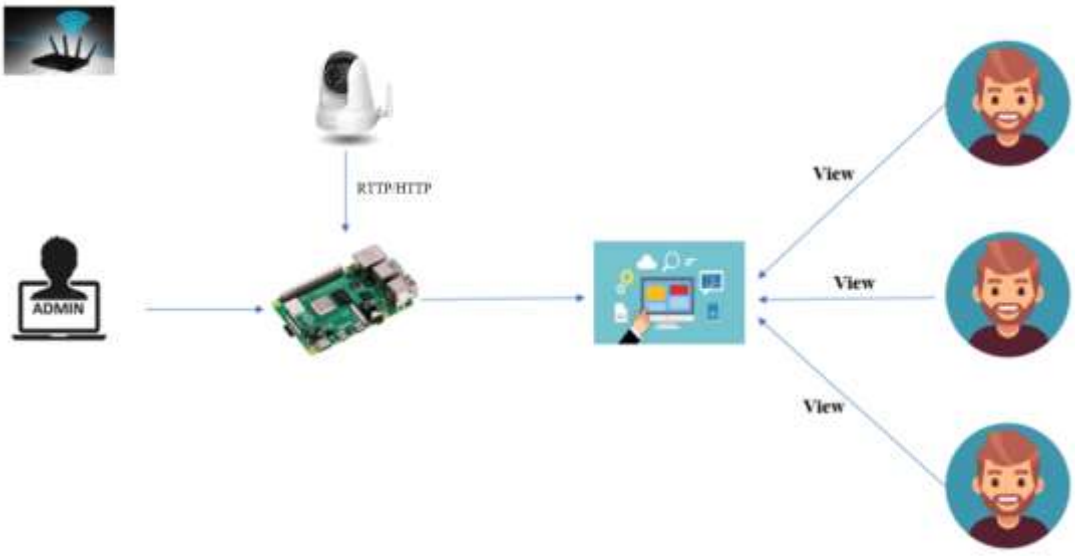
-Camera IP thu thập hình ảnh từ khu vực giám sát và Raspberry Pi đọc ảnh từ camera qua địa chỉ IP;

-Raspberry Pi xử lý từng khung hình và thực hiện nhận dạng khói/lửa bằng mô hình học sâu ResNet50;

-Sau khi nhận dạng, ảnh được gán nhãn và hiển thị ngay trên giao diện của ứng dụng Kivy;

-Quá trình này được thực hiện liên tục với từng ảnh (frame) đọc được từ camera, tạo thành một luồng video thời gian thực hiển thị trực tiếp trên ứng dụng.

Hệ thống hoạt động streaming video kết hợp xử lý và hiển thị được thể hiện trong Hình



Hình 3.16 Hệ thống hoạt động streaming video lên ứng dụng

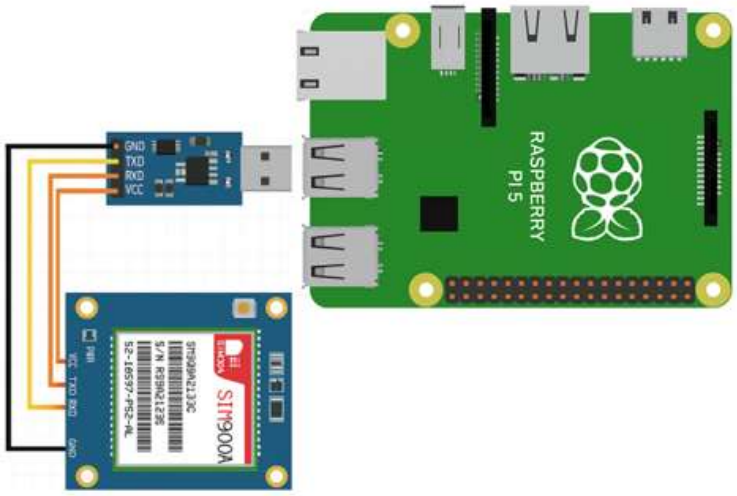
Admin: Sử dụng các camera IP được kết nối với mạng nội bộ để phát hiện khói/lửa. Raspberry Pi 5 sẽ đọc luồng video từ các camera này, xử lý khung hình theo thời gian thực bằng mô hình học sâu ResNet50 để nhận diện sự cố. Kết quả nhận dạng và video được hiển thị trực tiếp trên ứng dụng giao diện Kivy cài đặt trên Raspberry Pi.

User: Có thể theo dõi luồng video trực tiếp tại khu vực giám sát thông qua màn hình kết nối với Raspberry Pi, nơi giao diện Kivy hiển thị liên tục quá trình giám sát và phát hiện.

3.3.3.2 Thiết kế chương trình cảnh báo

Chương trình cảnh báo thông qua SMS hoặc cuộc gọi điện

a) Sơ đồ kết nối



Hình 3.17 Sơ đồ kết nối với module sim900A

b) Cách thức hoạt động

Cách thức hoạt động của hệ thống cảnh báo thông qua SMS hoặc cuộc gọi điện bao gồm các bước sau:

Bước 1: Phát hiện khói/lửa

- Hệ thống sử dụng mô hình học sâu ResNet50 trên Raspberry Pi 5 để phân tích hình ảnh từ camera IP. Khi mô hình xác định có dấu hiệu khói hoặc lửa, nó sẽ đánh dấu là sự cố cần cảnh báo.

Bước 2: Kích hoạt module SIM900A

- Raspberry Pi gửi các lệnh AT thông qua giao tiếp UART (qua USB-TTL converter) tới module SIM900A.

Bước 3: Gửi cảnh báo SMS hoặc gọi điện

- Gửi tin nhắn SMS, Pi sẽ gửi lệnh AT+CMGS cùng nội dung cảnh báo đến số điện thoại đã cài đặt, gọi điện, Pi gửi lệnh ATD<số>; để gọi đến người dùng.

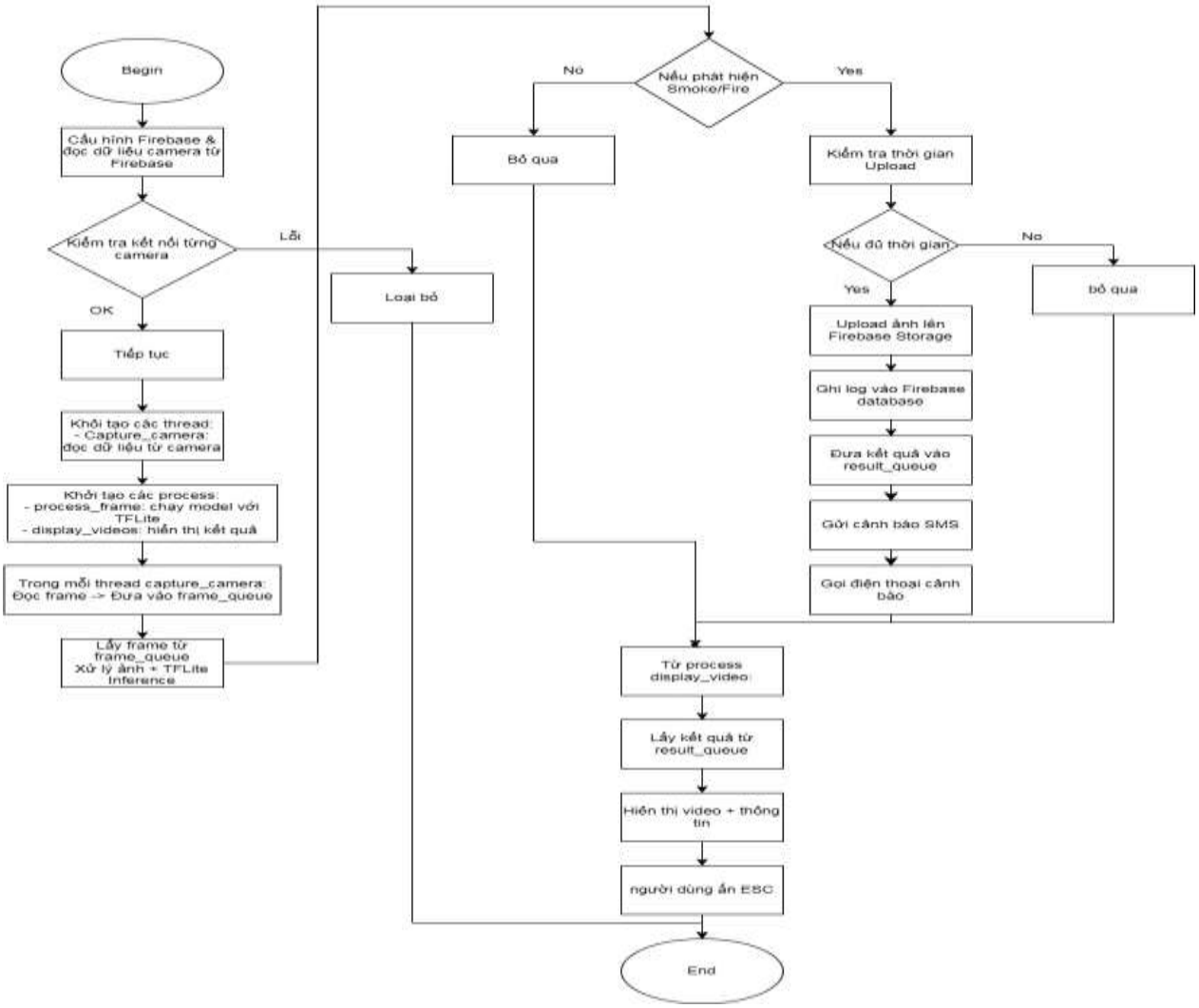
Bước 4: Hoàn tất cảnh báo

- Sau khi gửi SMS hoặc gọi điện thành công, hệ thống ghi nhận thời điểm và trạng thái cảnh báo để tránh gửi lặp lại nếu không có thay đổi mới.

Bước 5: Tiếp tục giám sát

- Hệ thống quay lại trạng thái giám sát để phát hiện và xử lý các tình huống cháy mới

3.3.3.3 Lưu đồ thuật toán xử lý phát hiện và cảnh báo khói/lửa trên Raspberry Pi



Hình 3.18 Lưu đồ thuật toán xử lý phát hiện và cảnh báo khói/lửa trên Raspberry Pi

Giải thích lưu đồ:

- **Khởi tạo hệ thống**: Raspberry Pi được cấu hình kết nối với Firebase để lấy danh sách camera từ cơ sở dữ liệu thời gian thực (Realtime Database) và khởi tạo SDK cho Firebase Storage và Database.

- **Kiểm tra kết nối camera**: Hệ thống lần lượt kiểm tra kết nối của từng camera. Camera nào lỗi sẽ bị loại bỏ, chỉ giữ lại những camera hoạt động tốt để xử lý.

- **Khởi tạo các luồng và tiến trình**:

  - Mỗi camera sẽ chạy trong một **luồng riêng** để đọc dữ liệu liên tục.
  - Một **tiến trình chính** dùng để xử lý hình ảnh bằng mô hình học sâu (AI).
  - Một **tiến trình phụ** dùng để hiển thị kết quả.

- **Xử lý AI**: Mỗi ảnh từ camera sẽ được đưa vào mô hình ResNet50 TFLite để dự đoán. Kết quả phân loại gồm 3 lớp: "Fire", "Smoke" và "No".

- **Cảnh báo**:

- Nếu kết quả là "No" → bỏ qua.
- Nếu phát hiện "Fire" hoặc "Smoke":
    - Hệ thống kiểm tra xem đã đủ thời gian giữa các lần cảnh báo chưa (tối thiểu 2 phút để tránh trùng lặp).
    - Nếu đủ điều kiện:
        - Upload ảnh lên Firebase Storage.
        - Ghi log thời gian, loại cảnh báo và link ảnh lên Firebase Realtime Database.
        - Gửi tin nhắn SMS và gọi điện thoại qua module SIM900 đến số điện thoại người giám sát đã cài đặt.

- **Hiển thị kết quả**: Tiến trình hiển thị sẽ lấy kết quả dự đoán từ hàng đợi (result queue) để gắn nhãn lên hình ảnh và hiển thị trực tiếp bằng OpenCV.

- **Kết thúc**: Người dùng nhấn phím ESC để kết thúc chương trình và đóng tất cả các tiến trình liên quan.

3.3.3.4 Cấu trúc cơ sở dữ liệu và giải pháp tối ưu lưu trữ

Trong hệ thống cảnh báo cháy, mỗi lần thiết bị nhận dạng phát hiện khói hoặc lửa đều sinh ra một bản ghi sự kiện đi kèm với ảnh minh họa được trích xuất từ luồng video. Để lưu trữ các sự kiện này một cách hiệu quả, đảm bảo vừa truy xuất nhanh vừa tiết kiệm dung lượng đề tài đề xuất thiết kế cơ sở dữ liệu theo hai hướng song song:

- Lưu trữ ảnh cảnh báo trên Firebase Storage;

- Lưu metadata sự kiện (thời gian, loại cảnh báo, đường dẫn ảnh...) trong Firestore hoặc Realtime Database.

1. Cấu trúc metadata lưu trên Firebase Database

Mỗi sự kiện được lưu dưới dạng một bản ghi JSON có cấu trúc như sau:

Hình 3.19 Hình ảnh sự kiện được lưu trữ trong Firebase

Trong đó:

- Camera 1 07-06-2025 – 17:48 : mã định danh duy nhất của sự kiện

- Data_time: thời gian xảy ra

- Link: đường dẫn tới ảnh đã lưu trên Firebase Storage
- Log: trạng thái gửi cảnh báo

2. Lưu trữ ảnh snapshot

Ảnh snapshot tại thời điểm phát hiện khói/lửa được nén định dạng JPEG và lưu vào thư mục con trên Firebase Storage:

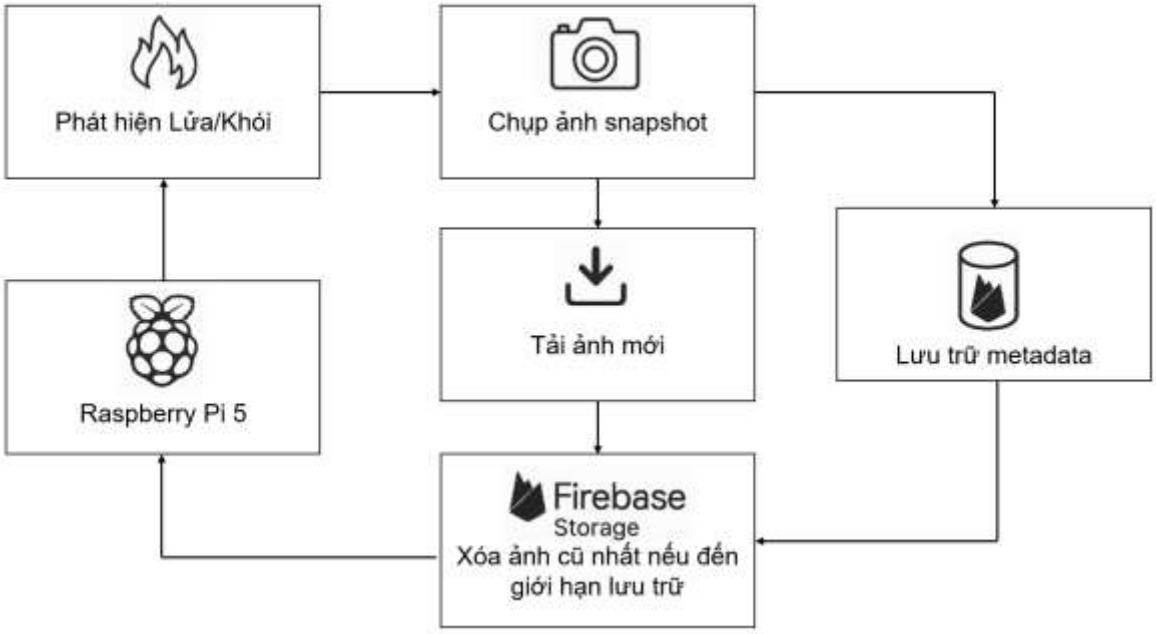
https://storage.googleapis.com/iot-camera3caf1.appspot.com/Image/Camera%201%2007-06-2025%20-%2017%3A48_20250607_174800.jpg

3. Giới hạn số ảnh lưu trữ (FIFO – First In First Out)

Vì Firebase Storage chỉ cho phép lưu tối đa 1 GB, hệ thống được thiết kế để chỉ lưu một số lượng ảnh mới nhất .Mỗi khi có ảnh mới được lưu, hệ thống sẽ:

- Kiểm tra tổng số ảnh hiện có trong thư mục snapshots;
- Nếu vượt quá ngưỡng, tự động xóa ảnh cũ nhất (dựa trên timestamp hoặc thứ tự lưu);
- Sau đó upload ảnh mới và cập nhật metadata vào database.

4. Sơ đồ mô tả luồng xử lý ảnh

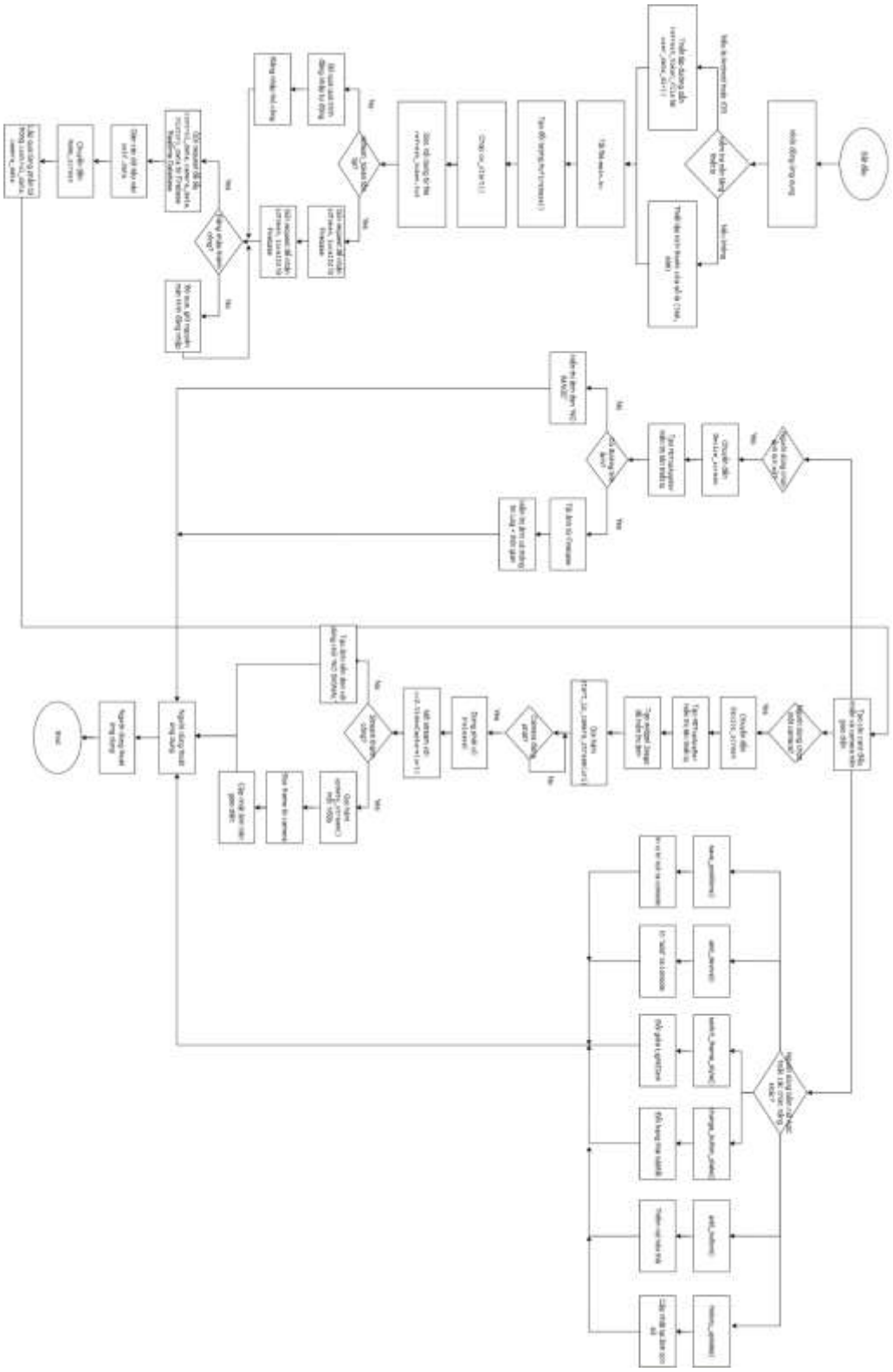


Hình 3.20 Sơ đồ mô tả luồng xử lý ảnh

5. Lợi ích của giải pháp đề xuất

- Giảm chi phí lưu trữ
- Dễ truy xuất theo thời gian hoặc theo thiết bị cảnh báo
- Đảm bảo hệ thống hoạt động ổn định liên tục mà không bị lỗi do đầy bộ nhớ

3.4 Lưu đồ thuật toán của hệ thống



Hình 3.21 Lưu đồ thuật toán chương trình nhận dạng và cảnh báo khói lửa

Lưu đồ thuật toán chương trình nhận dạng và cảnh báo khói lửa được thể hiện như Hình bao gồm các bước sau:

Bước 1. Khởi động ứng dụng

- Kiểm tra nền tảng thiết bị (Android/iOS hoặc desktop).
- Thiết lập kích thước cửa sổ:
  + Mobile: Full màn hình.
  + Desktop: Kích thước mặc định (360, 600).
- Tải file giao diện main.kv.
- Khởi tạo Firebase để kết nối với cơ sở dữ liệu.

Bước 2. Xử lý đăng nhập

- Kiểm tra file refresh_token.txt:
  + Nếu tồn tại: Đăng nhập tự động bằng token.
  + Nếu không: Yêu cầu đăng nhập thủ công.
- Sau khi đăng nhập thành công, lấy dữ liệu từ Firebase Realtime Database:
  + control_data: Danh sách thiết bị điều khiển.
  + camera_data: Danh sách camera.
  + history_data: Lịch sử hình ảnh.

Bước 3. Hiển thị màn hình chính (HomeScreen)

- Tạo các card hiển thị:
  + Thiết bị điều khiển: Hiển thị từ control_data.
  + Camera: Hiển thị từ camera_data.
  + Lịch sử: Hiển thị từ history_data (ảnh và thông tin log).

Bước 4. Xử lý chức năng chính

a. Xem camera

- Khi người dùng chọn một camera:
  + Chuyển đến DeviceScreen.
  + Hiển thị thanh toolbar với tên camera.
  + Mở stream RTSP từ URL camera bằng OpenCV:
    Nếu thành công: Hiển thị video real-time (60 FPS).
    Nếu thất bại: Hiển thị ảnh đen "NO SIGNAL".

b. Xem lịch sử

- Khi người dùng chọn một mục lịch sử:
  + Chuyển đến DeviceScreen.
  + Hiển thị ảnh từ Firebase Storage (nếu có) hoặc ảnh đen "NO IMAGE".
  + Hiển thị thông tin log và thời gian.

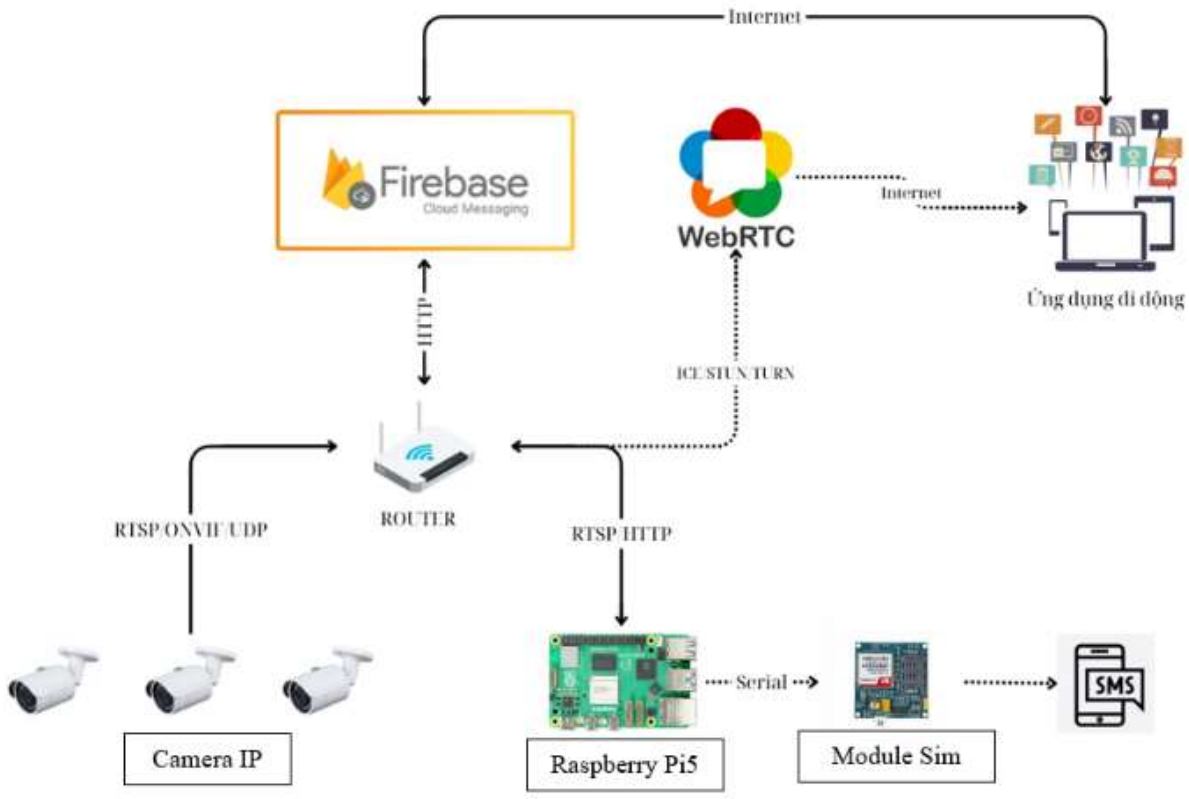
c. Các chức năng khác

- Thêm nút điều khiển: Kéo thả nút mới vào giao diện.
- Đổi chế độ sáng/tối: Toggle giữa Light/Dark theme.
- Lưu vị trí nút: In vị trí các nút ra console.
- Cập nhật lịch sử: Tải lại dữ liệu từ Firebase.

5. Thoát ứng dụng

- Dừng stream camera (nếu đang chạy).
- Giải phóng tài nguyên.
- Lưu trạng thái giao diện (nếu cần).

3.5 Kiến trúc tổng thể của hệ thống



Hình 3.22 Mô hình kiến trúc hệ thống

Mô hình kiến trúc hệ thống cảnh báo khói lửa :

- Camera IP

  + Các camera IP sử dụng giao thức RTSP / ONVIF / UDP để truyền dữ liệu video liên tục về mạng nội bộ.

  + Camera hoạt động trong thời gian thực, giám sát liên tục khu vực cần theo dõi.

- Router (mạng Wi-Fi nội bộ)

  + Là trung tâm kết nối giữa camera IP và Raspberry Pi 5.

  + Router chịu trách nhiệm phân phối dữ liệu video qua mạng nội bộ đến Raspberry Pi

  + Đồng thời, router cũng kết nối với Internet để truyền cảnh báo và dữ liệu tới người dùng.

- Raspberry Pi 5

  + Thiết bị xử lý trung tâm của hệ thống.

  + Nhận luồng video từ camera IP thông qua giao thức RTSP/HTTP.

  + Phân tích hình ảnh bằng mô hình học sâu ResNet50 để nhận dạng khói/lửa.

  + Khi phát hiện bất thường, Pi thực hiện 2 nhiệm vụ:

    ✓ Gửi cảnh báo SMS qua module SIM thông qua giao tiếp serial UART.
    ✓ Gửi thông báo đẩy lên ứng dụng di động thông qua Firebase Cloud Messaging (FCM).

- Firebase Cloud Messaging (FCM)

  + Nhận cảnh báo từ Raspberry Pi qua HTTP.

  + Chuyển tiếp cảnh báo đến ứng dụng di động đã đăng ký.

- WebRTC

  + Hỗ trợ truyền video trực tiếp (streaming) đến ứng dụng di động qua Internet.

  + Sử dụng các giao thức ICE, STUN, TURN để thiết lập kết nối ngang hàng giữa Raspberry Pi và thiết bị của người dùng.

  + Cho phép người dùng giám sát camera từ xa ngay trên điện thoại hoặc máy tính bảng.

- Ứng dụng di động

  + Nhận thông báo cảnh báo cháy từ Firebase.

  + Cho phép người dùng xem video thời gian thực thông qua WebRTC.

  + Hiển thị hình ảnh, thông tin cảnh báo (thời gian, vị trí camera, ảnh snapshot…).

- Module SIM900A

+ Kết nối với Raspberry Pi qua cổng serial (TX/RX).

+ Khi có cảnh báo, module gửi tin nhắn SMS hoặc thực hiện cuộc gọi tự động đến người dùng theo số điện thoại đã cấu hình.

3.6 Tích hợp và thử nghiệm

Hệ thống được tiến hành kiểm thử thực tế bằng cách sử dụng các nguồn video mô phỏng cháy, bao gồm: đốt giấy, khói thuốc và dữ liệu thu từ camera IP trong các môi trường khác nhau. Mục tiêu nhằm đánh giá khả năng nhận diện và phản ứng của hệ thống trong các tình huống đa dạng.

Các tình huống thử nghiệm bao gồm:

- Khói nhẹ thoát ra từ vật thể đang cháy âm ỉ (mô phỏng cháy khởi phát);
- Ngọn lửa rõ rệt xuất hiện nhanh (mô phỏng cháy lớn đột ngột);
- Ánh sáng mạnh từ mặt trời hoặc đèn điện (kiểm tra độ nhạy và khả năng loại nhiễu);
- Môi trường có chuyển động hoặc bụi (đánh giá độ ổn định và chống báo động giả).

Kết quả ban đầu:

- Hệ thống phát hiện chính xác các trường hợp có khói và/hoặc lửa;
- Thời gian phản hồi nhanh, kích hoạt cảnh báo trong vòng 1–2 giây sau khi nhận diện;
- Không xảy ra cảnh báo giả trong các thử nghiệm ánh sáng mạnh và chuyển động thông thường.

Kết quả này cho thấy tiềm năng ứng dụng thực tế của hệ thống trong việc giám sát cháy nổ tại các khu vực công nghiệp, đặc biệt là những nơi yêu cầu phản ứng nhanh và cảnh báo sớm.

3.7 Kết luận chương

Chương này đã trình bày chi tiết về thiết kế tổng thể của hệ thống cảnh báo cháy ứng dụng công nghệ học sâu, bao gồm: kiến trúc phần cứng, phần mềm và quy trình tích hợp giữa các thành phần.

Giải pháp sử dụng mô hình ResNet50 để nhận dạng khói và lửa theo thời gian thực, được triển khai trực tiếp trên thiết bị nhúng Raspberry Pi 5. Hệ thống cho phép kết nối với nhiều camera IP, đảm bảo khả năng giám sát linh hoạt và mở rộng cho các khu vực công nghiệp có quy mô lớn.

Nhờ vào cấu trúc tối ưu và cơ chế cảnh báo đa kênh (hiển thị trực quan, gửi SMS, thực hiện cuộc gọi), hệ thống cho thấy hiệu quả cao trong việc phát hiện sớm và cảnh báo kịp thời các tình huống cháy nổ, đáp ứng yêu cầu ứng dụng trong thực tế với độ tin cậy và tốc độ phản hồi nhanh.

CHƯƠNG 4: VẬN HÀNH THỬ NGHIỆM VÀ ĐÁNH GIÁ

Chương này sẽ trình bày chi tiết kết quả thử nghiệm và đánh giá thiết bị trong quá trình nhận dạng và cảnh báo khói lửa, sử dụng mô hình đã được nghiên cứu và triển khai trên Raspberry Pi 5. Do điều kiện thực tế tại thời điểm thực nghiệm không xảy ra sự cố cháy hoặc có hiện tượng khói rõ ràng trong môi trường xưởng hoặc hộ gia đình, việc tiến hành thử nghiệm trực tiếp với các tình huống cháy thật gặp nhiều khó khăn và tiềm ẩn rủi ro an toàn.

Vì vậy, trong quá trình kiểm thử, chúng tôi đã sử dụng các video và hình ảnh mô phỏng khói/lửa được thu thập từ internet, cũng như tự tạo các tình huống giả định bằng cách đốt giấy, tạo khói nhỏ trong môi trường có kiểm soát. Các hình ảnh và video này được chiếu lên màn hình hoặc camera ghi lại trực tiếp để hệ thống nhận dạng và xử lý như trong tình huống thực tế. Thử nghiệm cho thấy mô hình có thể phát hiện và phản hồi tương đối.

4.1 Mục tiêu thử nghiệm

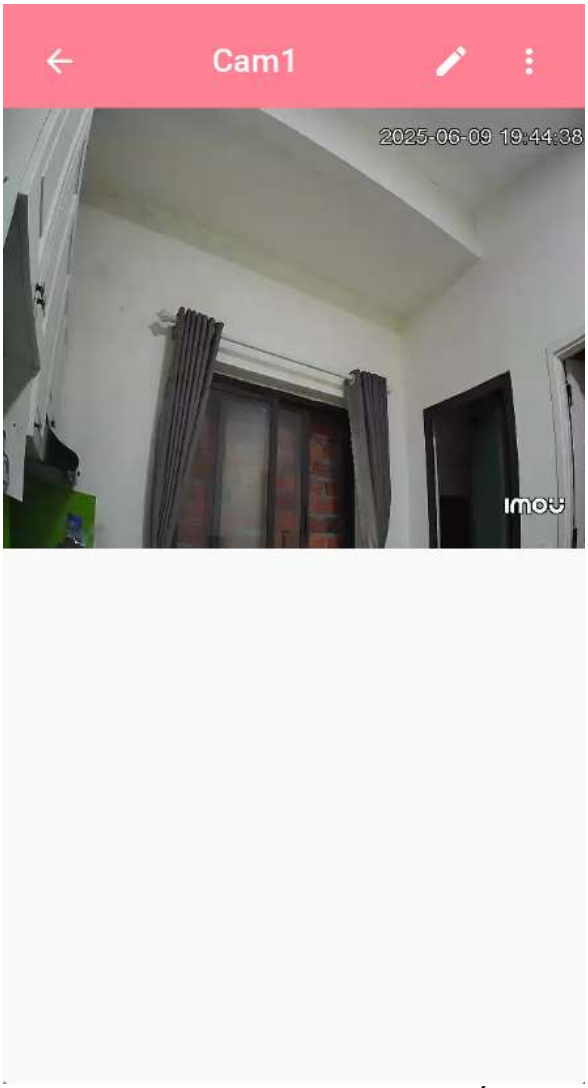
Mục tiêu của quá trình thử nghiệm là đánh giá hiệu quả hoạt động của hệ thống cảnh báo cháy trong các điều kiện thực tế khác nhau. Cụ thể, thử nghiệm được thiết kế nhằm kiểm tra:

- Khả năng nhận dạng khói và lửa của mô hình học sâu khi được triển khai trên nền tảng phần cứng nhúng Raspberry Pi 5;
- Hiệu năng xử lý, thông qua việc đo lường tốc độ phản hồi và độ trễ từ lúc phát hiện đến khi kích hoạt cảnh báo;
- Độ tin cậy của hệ thống trong các môi trường có yếu tố gây nhiễu như ánh sáng mạnh, bụi, hoặc chuyển động phức tạp trong khung hình;
- Tính hiệu quả của cơ chế cảnh báo, bao gồm khả năng gửi tin nhắn SMS, thực hiện cuộc gọi tự động và hiển thị thông báo trên ứng dụng di động, nhằm đảm bảo thông tin được truyền đến người quản lý một cách kịp thời và chính xác.

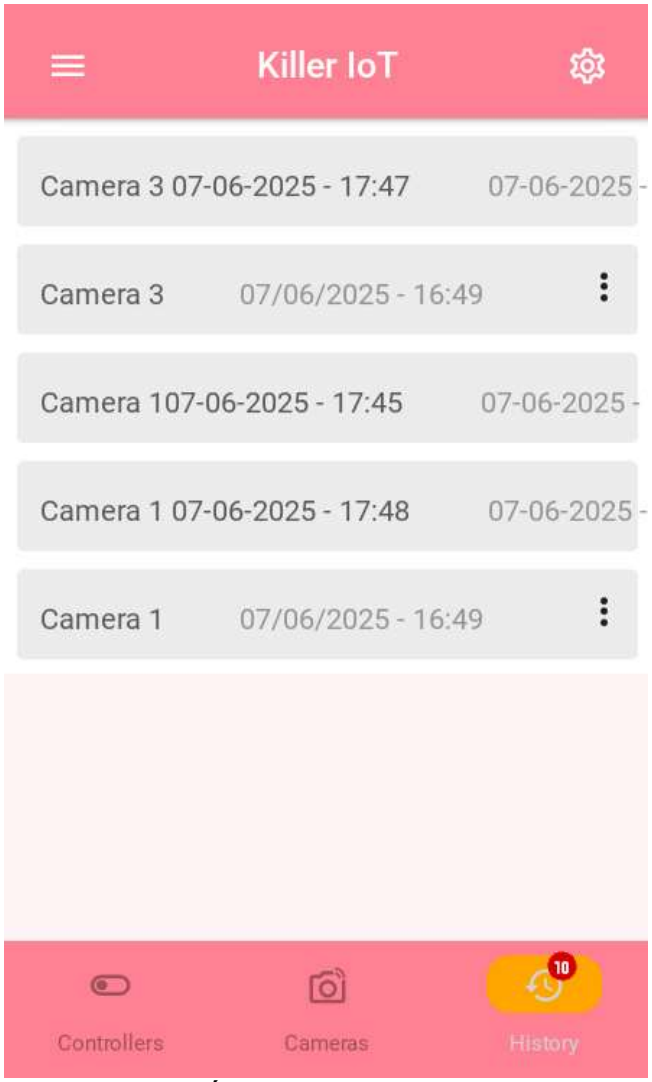
Thông qua quá trình kiểm thử này, hệ thống được đánh giá toàn diện về tính chính xác, độ ổn định và khả năng ứng dụng thực tiễn trong môi trường công nghiệp, góp phần làm rõ tính khả thi của giải pháp được đề xuất.

4.2 Thử nghiệm thiết bị thực tế

Thử nghiện thiết bị thông qua một đám cháy do tác giả thực hiện



Hình 4.1 Màn hình stream trực tiếp trên ứng dụng



Hình 4.2 Kết quả lịch sử được thu thập

Hình 4.3 Kết quả nhận dạng khói được hiển thị trên ứng dụng

Hình 4.4 Kết quả nhận dạng lửa được hiển thị trên ứng dụng

- Thông qua quá trình thử nghiệm thực tế, tác giả đã tiến hành kiểm tra hệ thống với các đám cháy nhỏ được tạo ra trong môi trường có kiểm soát (ví dụ như đốt giấy, tạo khói nhẹ) nhằm đảm bảo an toàn trong quá trình đánh giá.

- Kết quả thử nghiệm cho thấy mô hình ResNet50 tích hợp trên Raspberry Pi 5 có khả năng phát hiện khói và lửa chính xác trong hầu hết các tình huống. Thời gian xử lý trung bình cho mỗi khung hình khoảng 0,2 giây, và tốc độ xử lý đạt khoảng 8 FPS, tùy thuộc vào kích thước ảnh đầu vào và tình trạng hệ thống.

- Trong các lần thử nghiệm, mô hình cho độ chính xác trung bình đạt khoảng 85–90%. Thiết bị nhận diện tốt các đám cháy có đặc điểm rõ ràng về màu sắc và hình dạng, tuy nhiên khi khói mờ hoặc lan quá nhẹ, hệ thống đôi khi bỏ sót.

- Sau khi nhận diện được khói hoặc lửa, hệ thống thực hiện gửi cảnh báo kèm hình ảnh qua tin nhắn SMS và cuộc gọi điện thoại thông qua module SIM900A. Quá trình cảnh báo diễn

ra ổn định và kịp thời, đảm bảo thông tin được truyền đến người dùng ngay khi phát hiện sự cố.

4.3 Môi trường thử nghiệm

Các thử nghiệm được thực hiện trong các điều kiện môi trường thực tế và bán thực tế, bao gồm:

- Nguồn dữ liệu**:**
  + Video thực tế từ camera IP tại phòng thí nghiệm, hành lang tòa nhà;
  + Dữ liệu video mô phỏng khói (đốt giấy, khói nhang) và lửa nhỏ (bật lửa, nến cháy);
  + Một phần dữ liệu lấy từ bộ dữ liệu công khai [1] để huấn luyện và đánh giá bổ sung.
- Thiết bị phần cứng:
  + Raspberry Pi 5 (8GB RAM);
  + Camera IP độ phân giải 720p, giao thức RTSP;
  + Module SIM900A gắn kèm;
  + Mạng LAN nội bộ và router Wi-Fi.

4.4 Các tình huống thử nghiệm

Để đánh giá hiệu quả nhận dạng khói và lửa của hệ thống trong điều kiện thực tế, một loạt tình huống thử nghiệm đã được xây dựng mô phỏng các hoàn cảnh có thể xảy ra tại môi trường công nghiệp. Các tình huống này bao gồm cả trường hợp có cháy thật, khói nhẹ, nhiễu sáng, chuyển động và cả tình huống dễ gây báo động giả.

Mục tiêu của quá trình thử nghiệm là kiểm tra khả năng nhận dạng chính xác, tốc độ phản hồi, độ ổn định cũng như độ nhạy của mô hình trong nhiều điều kiện khác nhau. Ngoài ra, hệ thống cũng được kiểm tra khả năng xử lý đa luồng khi hoạt động với nhiều camera cùng lúc.

Bảng dưới đây mô tả các tình huống thử nghiệm chính được sử dụng:

STT	Mô tả tình huống	Mục tiêu kiểm tra
1	Lửa rõ ràng, gần camera	Kiểm tra khả năng phát hiện nhanh
2	Lửa nhỏ (nến cháy)	Phân biệt lửa nhỏ với ánh sáng thường
3	Chuyển động của người/vật thể	Kiểm tra độ ổn định nhận diện
4	Nhiều camera đồng thời	Kiểm tra đa luồng xử lý

4.5 Đánh giá độ chính xác và tốc độ nhận dạng

4.5.1 Đánh giá độ chính xác với model ResNet50

Để đánh giá hiệu quả của hệ thống nhận dạng khói và lửa, thực hiện đã sử dụng bộ dữ liệu thử nghiệm gồm 5.900 ảnh được gán nhãn chính xác, trong đó:

- 4500 ảnh chứa dấu hiệu khói/lửa (positive),

- 1400 ảnh không có dấu hiệu cháy (negative),

Dữ liệu lấy từ các nguồn gồm: video thực tế thu tại hiện trường, ảnh từ Internet, và ảnh cắt từ video huấn luyện.

Việc đánh giá được thực hiện dựa trên các chỉ số:

Accuracy (độ chính xác tổng thể)**:**

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision (độ chính xác dương tính)**:**

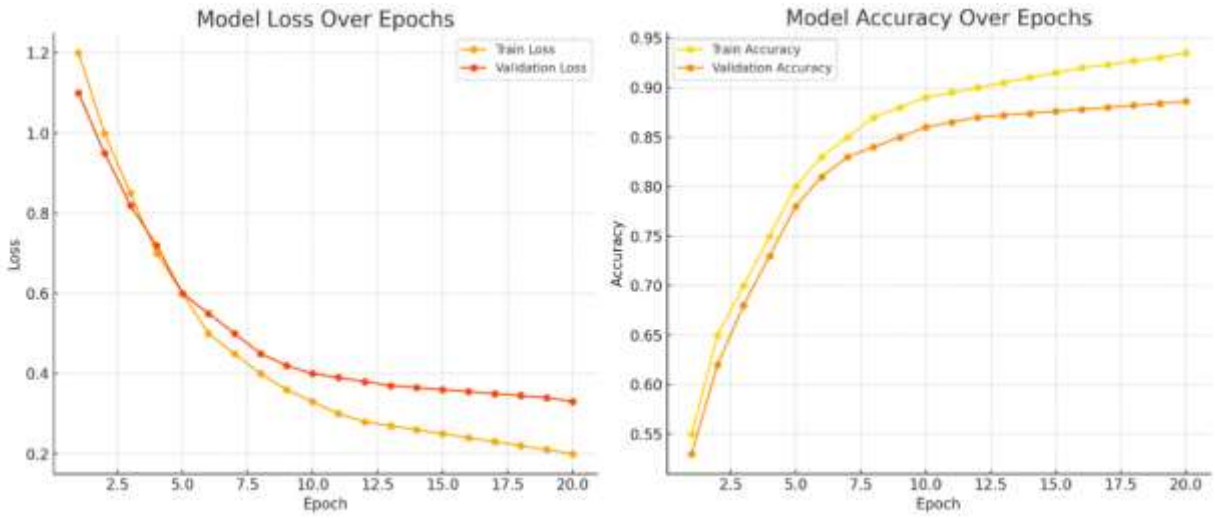
$$Precision = \frac{TP}{TP + FN}$$

Recall (tỷ lệ phát hiện đúng)**:**

$$Recall = \frac{TP}{TP + FN}$$

F1-Score (chỉ số cân bằng precision và recall)**:**

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$



Hình 4.5 Hình biểu đồ Training vs Validation Loss và Training Validation Accuracy

Bảng 4-1 Kết quả các thông số đánh giá mô hình Resnet50

Model	Accuracy	Loss	Precision	recall	F1 - score	Sizes (MB)
Resnet50	0.965	0.12	0.927	0.934	0.93	97

Hiệu năng thời gian thực

Hệ thống được triển khai trên Raspberry Pi 5 với mô hình ResNet50 chạy qua thư viện TensorFlow Lite . Kết quả cho thấy:

- Tốc độ xử lý trung bình: ~6.8 FPS (frame per second) cho ảnh kích thước 224×224.
- Thời gian trích xuất, xử lý và đưa ra quyết định cho 1 ảnh: ~150–180 ms.
- Tổng thời gian từ lúc có cháy đến lúc gửi cảnh báo: khoảng 1.8–2.3 giây.

Điều này chứng tỏ mô hình đủ nhanh để ứng dụng giám sát thời gian thực và gửi cảnh báo gần như tức thời.

4.5.2 So sánh với model Yolo v11

Bảng 4-2 Bảng so sánh Model ResNet50 và YOLOv11

Tiêu chí	**ResNet50 (classification)**	**YOLOv11 (detection)**	**Nhận xét tổng hợp**
Task	Phân loại khói/lửa	Phát hiện đối tượng	YOLO dư thừa chức năng cho bài toán
Mục tiêu đầu ra	Có/Không có khói/lửa	Xác định vị trí và loại đối tượng	YOLO mạnh trong định vị, nhưng không cần thiết cho bài toán này
Yêu cầu dữ liệu huấn luyện	Ít hơn (ảnh nhãn nhị phân)	Cần ảnh và nhãn bounding box	Huấn luyện YOLO phức tạp, cần nhiều dữ liệu đánh nhãn chi tiết
F1-score	0.93	0.91	ResNet50 cân bằng tốt precision/recall
Dung lượng mô hình	~97 MB	~160 MB	YOLO lớn hơn, khó tối ưu trên raspberry pi 5
Tốc độ xử lý trên Raspberry Pi	~15ms/ảnh (với TFLite)	~30ms/ảnh (chưa gồm tạo luồng, xử lý)	ResNet nhanh gấp đôi khi triển khai trên thiết bị nhúng không có GPU
Yêu cầu phần cứng	CPU (TFLite)	GPU (tốt nhất với CUDA)	Raspberry pi 5 không có GPU, bất lợi cho YOLO
Tối ưu hóa	Dễ tối ưu qua TFLite	Cần chuyển sang YOLOv5n, TensorRT...	YOLO cần giảm kiến trúc mới chạy tốt trên raspberry pi 5
Khả năng triển khai thực tế	Cao	Hạn chế nếu không tối ưu	ResNet phù hợp hơn cho thời gian thực
Độ trễ toàn hệ thống	~30ms (trọn quy trình)	~60ms+ (ảnh + xử lý box + gửi cảnh báo)	ResNet đảm bảo luồng liên tục hơn

Từ bảng 4-2 có thể thấy rằng việc sử dụng mô hình ResNet50 cho bài toán nhận dạng khói/lửa là lựa chọn hợp lý vì các lý do sau:

- Độ chính xác cao, đủ để phát hiện sớm nguy cơ cháy;
- Thời gian suy luận nhanh, đảm bảo hệ thống phản hồi thời gian thực (<2 giây);
- Kích thước mô hình nhỏ, dễ triển khai trên Raspberry Pi 5 sử dụng CPU;
- Không yêu cầu dữ liệu huấn luyện phức tạp như bounding box;
- Đơn giản hóa hệ thống xử lý và cảnh báo.

Ngược lại, mặc dù YOLOv11 là mô hình mạnh trong phát hiện đối tượng, nhưng trong bối cảnh này lại gây ra nhiều bất lợi:

- Kích thước mô hình lớn, tốc độ xử lý chậm hơn;
- Cần GPU hoặc tối ưu chuyên sâu mới chạy mượt;
- Cần dữ liệu gán nhãn bounding box khó thu thập;
- Độ trễ tăng cao khi xử lý luồng video nhiều camera trên Raspberry pi 5.

Vì vậy, mô hình ResNet50 là lựa chọn phù hợp nhất, vừa đáp ứng tốt về độ chính xác, vừa đảm bảo hiệu năng hệ thống trên thiết bị nhúng có tài nguyên hạn chế.

4.6 Kết quả đạt được

Sau quá trình khảo sát những vấn đề đặt ra ngoài thực tế, tác giả đã tiến hành

Bảng 4-3 Bảng tổng hợp các đặc tính và kết quả đạt được

STT	Kết quả	Ưu điểm	Nhược điểm	Hướng khắc phục (nếu có)
1	Chế tạo thành công thiết bị nhận dạng khói lửa	Thiết bị hoạt động đúng yêu cầu đề ra, có khả năng nhận dạng được khói và lửa	Số lượng dữ liệu hình ảnh khói/lửa còn chưa đa dạng, mô hình chưa đủ tổng quát trong một số trường hợp	Cần bổ sung thêm dữ liệu ảnh ở nhiều góc, điều kiện ánh sáng và môi trường khác nhau để tăng độ chính xác mô hình
2	Hệ thống phát hiện được khói/lửa qua hình ảnh từ camera IP theo thời gian thực	Phù hợp với môi trường giám sát nhà xưởng, dễ triển khai	Độ trễ còn phụ thuộc vào độ phân giải và tốc độ xử lý	Có thể giảm độ phân giải khung hình
3	Mô hình học sâu ResNet50 hoạt động ổn	Cho kết quả nhận dạng tương đối chính	Mô hình nặng, tốc độ xử lý còn chậm (FPS ~8)	Có thể chuyển sang mô hình nhẹ hơn nếu muốn tăng tốc

	định trên Raspberry Pi 5	xác, không yêu cầu GPU rời		độ; hoặc xử lý theo batch/lượt
4	Khi phát hiện khói/lửa, hệ thống lưu ảnh, gửi SMS và thực hiện cuộc gọi	Đã gửi được cảnh báo kèm hình ảnh về thiết bị người dùng	Thời gian gửi ảnh lên ứng dụng phụ thuộc vào tốc độ mạng. Khi mạng yếu, quá trình bị gián đoạn	Cần bổ sung cơ chế retry (thử lại), hoặc hỗ trợ kết nối mạng di động qua SIM900A nếu mất Wi-Fi
5	Thiết bị sử dụng Raspberry Pi 5 để xử lý và điều khiển	Kích thước nhỏ gọn, giá thành rẻ, đủ khả năng xử lý cơ bản	Hiệu suất xử lý chưa cao với mô hình lớn	Chấp nhận được vì mục tiêu là phát hiện chính xác và cảnh báo. Có thể cân nhắc tối ưu mã nguồn và dùng mô hình nhẹ hơn nếu cần
6	Thiết bị có thể hoạt động liên tục	Thiết kế gọn nhẹ, dễ lắp đặt	Khi mất điện, cần cấp nguồn riêng cho camera và Raspberry Pi	Sử dụng sạc dự phòng có dung lượng lớn hoặc tích hợp pin lithium + mạch sạc để cấp nguồn liên tục, cấp nguồn điện dự phòng cho thiết bị
7	Hệ thống xử lý dữ liệu từ nhiều camera IP	Hệ thống có thể mở rộng để xử lý từ 2–4 camera IP tùy theo cấu hình	Raspberry Pi 5 bị giới hạn tài nguyên (CPU, RAM, băng thông USB), dẫn đến tình trạng lag hoặc treo khi xử lý từ 5 camera IP trở lên cùng lúc	Chia quá trình xử lý theo lượt (batch), mỗi lần xử lý tối đa 2–4 camera. Có thể dùng đa tiến trình (multiprocessing) hoặc phân chia khung thời gian để xử lý từng nhóm camera một cách luân phiên.

nghiên cứu lý thuyết, thiết kế, mô phỏng, chế tạo và tiến hành các thí nghiệm cần thiết. Hệ thống đã được chế tạo hoàn thiện, tổng hợp các đặc điểm cũng như các kết quả đạt được trong quá trình nghiên cứu tạo ra hệ thống được trình bày ở Bảng 4.3.

4.7 Đánh giá thực nghiệm xử lý nhiều camera cùng lúc

Mô hình thử nghiệm

- Sử dụng 4 camera IP truyền luồng RTSP vào Raspberry Pi 5.

- Thiết lập mỗi luồng xử lý riêng biệt (multi-thread).
- Cấu hình hệ thống xử lý song song các khung hình từ từng camera.
- Mỗi camera có thể hướng về một nguồn khói/lửa khác nhau để đánh giá khả năng phân biệt.

Bảng 4-4 Bảng các chỉ số cần đánh giá

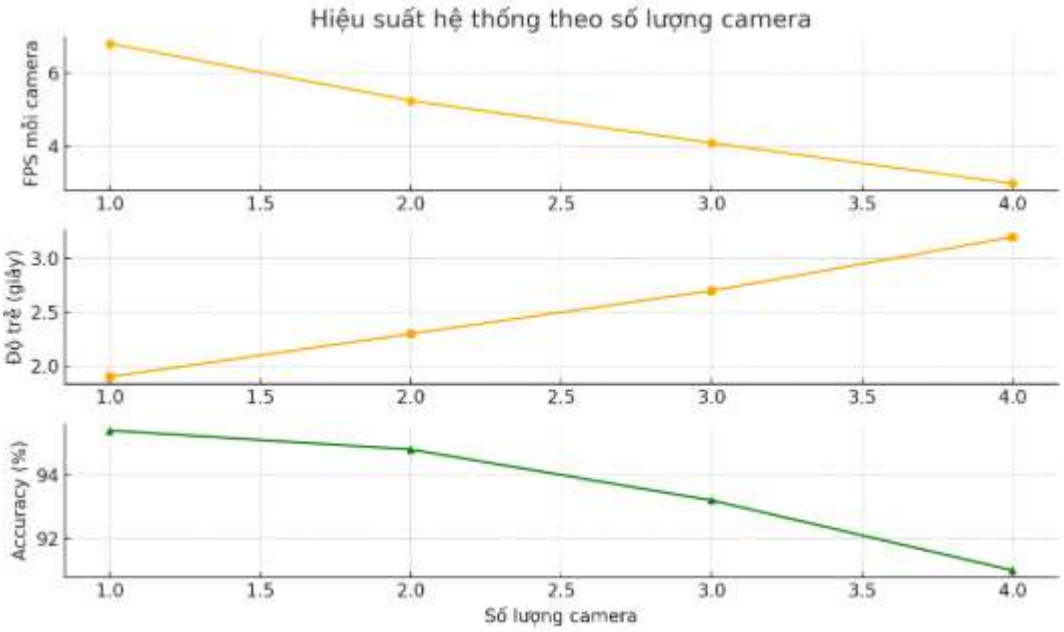
Tiêu chí	Mô tả đánh giá
Tốc độ xử lý trung bình (FPS)	Đo FPS trên mỗi camera và toàn bộ hệ thống khi số camera tăng từ 1→2 →3 →4
Độ trễ nhận diện	Thời gian từ lúc khói lửa xuất hiện đến khi cảnh báo kích hoạt
CPU và RAM sử dụng (%)	Giám sát mức sử dụng tài nguyên để xác định ngưỡng tối đa thiết bị có thể xử lý
Tỷ lệ bỏ sót (FN)	Kiểm tra xem có camera nào bị bỏ qua sự kiện cháy do tải nặng hoặc trễ không
Tỷ lệ cảnh báo (FP)	Kiểm tra xem khi tải cao hệ thống có nhầm lẫn không
Ổn định hoạt động (Stability)	Hệ thống có bị crash, treoe khi xử lý nhiều camera không

Bảng 4-5 Bảng kết quả thử nghiệm

Số camera	Tổng FPS	Độ trễ trung bình	CPU(%)	RAM(%)
1	7	2	40%	900MB
2	11	2.5	68%	1.3GB
3	13	3	70%	1.7GB
4	15	3.5	80%	2.1GB
5	20	4	50%	1GB

Nhận xét và đánh giá

- Khả năng mở rộng: Raspberry Pi 5 xử lý tốt với 2–3 camera, nhưng khi vượt quá 3, hiệu năng suy giảm rõ rệt.
- Tài nguyên giới hạn: Raspberry Pi 5 chỉ xử lý giới hạn trong 4 luồng liên tục, nếu có camera thứ 5 thì sẽ tự động giải phóng CPU và RAM để lấy dữ liệu của camera thứ 5 dẫn đến độ trễ tăng lên
- Khuyến nghị:
  + Sử dụng xử lý bất đồng bộ hoặc multi-thread để tối ưu tải;
  + Nếu cần >3 camera, nên phân tải sang nhiều thiết bị hoặc nâng cấp phần cứng;



Hình 4.6 Biểu đồ đường mô tả độ trể theo số lượng camera

4.8 Hướng cải tiến

Với những vấn đề còn tồn tại của thiết bị, hướng khắc phục ta sẽ chia làm 3 phần: dữ liệu hình ảnh, xử lý ảnh, phần cứng.

4.8.1 Đối với dữ liệu hình ảnh

Chất lượng và độ đa dạng của dữ liệu huấn luyện ảnh hưởng trực tiếp đến khả năng nhận dạng của mô hình. Do đó, cần cải thiện các vấn đề sau:

- Cần thu nhập nhiều hơn hình ảnh, dữ liệu phong phú của nhiều loại đám cháy, dạng khói lửa khác nhau thường có ở nhiều loại môi trường khác nhau;

- Tăng cường dữ liệu, áp dụng các kỹ thuật như xoay ảnh, thay đổi độ sáng/tối, thêm nhiễu, làm mờ, thay đổi góc nhìn,... nhằm mô phỏng nhiều tình huống thực tế hơn.

- Trong quá trình sử dụng, cần liên tục bổ sung thêm dữ liệu thực tế từ hệ thống để mô hình ngày càng thích nghi tốt hơn với các tình huống mới.

Để thiết bị có thể được huấn luyện nhiều trường hợp khác nhau và tăng độ nhận dạng, phát hiện đối tượng.

4.8.2 Đối với xử lý ảnh

Để mô hình hoạt động ổn định trong nhiều điều kiện ánh sáng và môi trường khác nhau, cần cải tiến các bước tiền xử lý hình ảnh:

- Xử lý ảnh đầu vào tốt hơn: Nghiên cứu các kỹ thuật loại bỏ nhiễu, cân bằng sáng trong trường hợp ảnh quá tối hoặc quá sáng, để mô hình nhận dạng ổn định hơn.

- Giảm độ phân giải hợp lý: Tối ưu kích thước ảnh đầu vào để giảm tải xử lý nhưng vẫn đảm bảo đủ chi tiết cho nhận dạng.

4.8.3 Đối với phần cứng

Để nâng cao hiệu suất và độ tin cậy của hệ thống, các hướng cải tiến phần cứng bao gồm:

- Nâng cấp nguồn điện: Sử dụng sạc dự phòng dung lượng lớn, pin lithium hoặc cấp nguồn điện dự phòng riêng để đảm bảo thiết bị hoạt động liên tục khi mất điện.

- Nâng cấp máy tính nhúng: Nếu yêu cầu tốc độ xử lý cao hơn, có thể thay thế Raspberry Pi 5 bằng thiết bị nhúng có hiệu năng cao hơn như Jetson Nano, Jetson Xavier NX,...

- Tối ưu phần cứng theo nhu cầu thực tế: Nếu chỉ giám sát ít khu vực, giữ nguyên Pi 5 là hợp lý. Nhưng nếu giám sát nhiều khu vực với nhiều camera, cần:

  + Giới hạn số camera xử lý đồng thời từ 2–4 để tránh quá tải CPU/RAM.

  + Xử lý theo nhóm (batch): Chia camera thành các nhóm, xử lý luân phiên.

  + Hoặc chuyển mô hình xử lý sang máy chủ trung tâm, Raspberry Pi chỉ làm nhiệm vụ thu nhận hình ảnh và gửi về server để xử lý.

4.9 Kết luận

Hệ thống đã được thử nghiệm thành công trong nhiều điều kiện thực tế khác nhau và cho kết quả nhận dạng khả quan. Mô hình học sâu ResNet50 kết hợp với phần cứng nhúng Raspberry Pi đã chứng minh được tính ứng dụng cao, đáp ứng tốt yêu cầu cảnh báo cháy sớm trong nhà xưởng. Các kết quả thử nghiệm giúp xác nhận hiệu quả của giải pháp đề xuất, đồng thời mở ra tiềm năng triển khai thực tế trên diện rộng.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Kết luận:

Đề tài này tập trung vào việc nghiên cứu, thiết kế, chế tạo và thử nghiệm một hệ thống nhận dạng và cảnh báo khói lửa trong môi trường nhà xưởng hoặc không gian dân dụng, sử dụng mô hình học sâu ResNet50 tích hợp trên máy tính nhúng Raspberry Pi 5.Các nội dung từ lý thuyết, mô hình hóa, huấn luyện mô hình, thiết kế phần cứng, cho đến thử nghiệm thực tế đều đã được thực hiện. Kết quả thử nghiệm cho thấy hệ thống có thể nhận dạng được khói và lửa với độ chính xác tương đối tốt, đồng thời gửi cảnh báo kèm hình ảnh thông qua tin nhắn SMS và cuộc gọi tự động bằng module SIM900A.Hệ thống sử dụng các thư viện OpenCV và tích hợp giao tiếp qua UART với tập lệnh AT để thực hiện cảnh báo. Ngoài ra, các khung hình có thể được lưu trữ và xem lại nhằm phục vụ mục đích giám sát hoặc xác minh sau cảnh báo.

Hướng phát triển:

Trong thời gian tới, tác giả sẽ tiếp tục cải tiến hệ thống theo các hướng sau:

- Bổ sung dữ liệu huấn luyện thực tế, bao gồm nhiều tình huống cháy trong môi trường khác nhau (ban ngày, ban đêm, nhà kho, ngoài trời...) để nâng cao độ chính xác của mô hình.

- Tối ưu tốc độ xử lý và khả năng xử lý đồng thời nhiều luồng hình ảnh từ nhiều camera IP bằng cách áp dụng phương pháp xử lý theo batch hoặc phân cụm luồng dữ liệu theo thời gian.

- Tích hợp thêm các cảm biến môi trường (nhiệt độ, khí gas...) nhằm hỗ trợ nhận dạng nguy cơ cháy từ nhiều nguồn tín hiệu khác nhau.

- Trong tương lai, có thể phát triển phiên bản mở rộng hoạt động trên server hoặc thiết bị nhúng mạnh hơn để phục vụ cho các hệ thống giám sát quy mô lớn.

TÀI LIỆU THAM KHẢO

[1] World Health Organization (WHO), *Fire safety in buildings*, 2021.

[2] Bộ Công an Việt Nam, *Báo cáo tình hình cháy nổ toàn quốc năm 2023*.

[3] Kim, Y., et al., "Limitations of conventional fire sensors and need for AI-based early warning systems," *Sensors*, 2021.

[4] M. Muhammad, A. S. Malik, et al., "Early Fire Detection Using Surveillance Videos: A Survey," *Digital Signal Processing*, vol. 96, 2020.

[5] Celik, T. et al., "Fire detection using statistical color model in video sequences," *Journal of Visual Communication and Image Representation*, vol. 18, no. 2, pp. 176–185, 2007.

[6] Tao, L. et al., "Fire and smoke detection using ResNet-50 deep learning model," *IEEE Access*, vol. 8, pp. 181749–181760, 2020.

[7] Zhang, L., et al., "Vision-based fire detection using deep learning," *Fire Safety Journal*, vol. 91, pp. 61–71, 2017.

[8] Simonyan, K., & Zisserman, A. (2015). "Very deep convolutional networks for large-scale image recognition," *ICLR 2015*.

[9] Szegedy, C. et al., "Rethinking the inception architecture for computer vision," *CVPR 2016*.

[10] Howard, A. et al., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv:1704.04861*, 2017.

[11] Redmon, J. et al., "You Only Look Once: Unified, Real-Time Object Detection," *CVPR 2016*.

[12] He, K., et al., "Deep Residual Learning for Image Recognition," *CVPR 2016*.

[13] LeCun, Y., Bengio, Y., & Hinton, G. (2015). "Deep learning," *Nature*.

[14] Simonyan, K., & Zisserman, A. (2015). "Very deep convolutional networks for large-scale image recognition," *ICLR*.

[15] Szegedy, C. et al. (2016). "Rethinking the inception architecture for computer vision," *CVPR*.

[16] Howard, A. et al. (2017). "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv:1704.04861*.

[17] Redmon, J. et al. (2016). "You Only Look Once: Unified, Real-Time Object Detection," *CVPR*.

[18] He, K., Zhang, X., Ren, S., & Sun, J. (2016). "Deep Residual Learning for Image Recognition," *CVPR*.

[19] Tao, L. et al. (2020). "Fire and smoke detection using ResNet-50 deep learning model," *IEEE Access*.

[20] NVIDIA, “Deploying Deep Learning Models on Edge Devices with TensorRT,” 2022.

[21] Raza, A. et al. (2021). “Real-time fire detection using deep CNN on Raspberry Pi,” *Procedia Computer Science*.

[22] K. Fridman, "Keras ResNet: Building, Training, and Scaling Residual Nets in Keras," *MissingLink.ai*. [Online]. Available: https://missinglink.ai/guides/keras/keras-resnet-building-training-scaling-residual-nets-keras/. [Accessed: 09-Jun-2025].

[22] R. Brownlee, "Introduction to ResNets," *Towards Data Science*, 2018. [Online]. Available:https://towardsdatascience.com/introduction-to-resnets-c0a830a288a4. [Accessed: 09-Jun-2025].

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv preprint arXiv:1512.03385*, 2015. [Online]. Available: https://arxiv.org/pdf/1512.03385.pdf.

[24] A. Nagrani, "An Overview of ResNet and Its Variants," *Towards Data Science*, 2020. [Online]. Available: https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035. [Accessed: 09-Jun-2025].

[25] Wang Z., Li Y., Wu H., “Enhancing Fire and Smoke Detection Using Deep Learning Techniques,” *MDPI Sensors*, vol. 22, no. 15, pp. 5022–5036, 2022.

[26] Ahmad Z., Akhter N., “A Real-Time Forest Fire and Smoke Detection System Using Deep Learning,” *Environmental Monitoring and Assessment*, vol. 193, no. 12, pp. 1–13, 2021.