

UNIVERSITY OF SCIENCE  
FACULTY OF INFORMATION TECHNOLOGY



Đồ án cuối kì  
Xây dựng và Triển khai Hệ thống Học máy Ứng  
dụng

# BÁO CÁO CUỐI KÌ

Supervisor: Bùi Tiên Lên

Students:	Full Name	ID
	Nguyễn Lê Quang	23127109
	Phan Hoàng Quang Nghị	23127436
	Nguyễn Tấn Văn	23127515

Ho Chi Minh City, 2025

# Mục lục

<b>1 Giới thiệu</b>	<b>4</b>
1.1 Phân tích Vấn đề (Problem Definition) . . . . .	4
1.2 Mục tiêu của Đồ án . . . . .	4
1.3 Tổng quan về Phương pháp . . . . .	5
<b>2 Thu thập và Phân tích Dữ liệu</b>	<b>6</b>
2.1 Nguồn và Phương pháp Thu thập . . . . .	6
2.2 Tiền xử lý và Làm sạch (Preprocessing & Cleaning) . . . . .	7
2.3 Phân tích Khám phá Dữ liệu (EDA) . . . . .	8
<b>3 Lựa chọn và Huấn luyện Mô hình</b>	<b>13</b>
3.1 Chuẩn bị Dữ liệu cho Mô hình . . . . .	13
3.2 Lựa chọn và Kiến trúc Mô hình . . . . .	14
3.3 Cấu hình Huấn luyện (Training Configuration) . . . . .	17
3.4 Phương pháp tinh chỉnh tham số . . . . .	18
<b>4 Kết quả và Thảo luận</b>	<b>19</b>
4.1 Kết quả Thực nghiệm . . . . .	19
4.2 So sánh và Thảo luận . . . . .	24
<b>5 Xây dựng và Triển khai Ứng dụng</b>	<b>27</b>
5.1 Kiến trúc Hệ thống . . . . .	27
5.2 Đóng gói và Tích hợp Mô hình . . . . .	30
5.3 Giao diện và Chức năng . . . . .	30
5.4 Triển khai . . . . .	37
<b>6 Kết luận</b>	<b>39</b>
6.1 Tóm tắt Kết quả . . . . .	39
6.2 Hạn chế . . . . .	39
6.3 Hướng phát triển . . . . .	40
<b>7 Tài liệu tham khảo</b>	<b>40</b>

# THÔNG TIN NHÓM VÀ PHÂN CÔNG CÔNG VIỆC

Họ và tên	MSSV	Phân công công việc
Nguyễn Lê Quang	23127109	<ul style="list-style-type: none"><li>– Tham gia tìm hiểu và lựa chọn đề tài.</li><li>– Thu thập dữ liệu từ Kaggle/Roboflow.</li><li>– Viết script tiền xử lý dữ liệu (Resize, chuẩn hóa nhãn).</li><li>– Soạn thảo Báo cáo Dữ liệu (Data Report).</li></ul>
Phan Hoàng Quang Nghị	23127436	<ul style="list-style-type: none"><li>– Nghiên cứu các kiến trúc mô hình (YOLOv8, YOLOv10, RT-DETR).</li><li>– Thực hiện huấn luyện mô hình (Training) và tinh chỉnh tham số.</li><li>– Vẽ biểu đồ và phân tích kết quả thực nghiệm.</li><li>– Soạn thảo Báo cáo Mô hình (Model Report).</li></ul>
Nguyễn Tân Văn	23127515	<ul style="list-style-type: none"><li>– Xây dựng ứng dụng Demo (Web App/Deployment).</li><li>– Tích hợp API cảnh báo (Telegram/Zalo).</li><li>– Tổng hợp tài liệu và biên soạn Slide báo cáo cuối kỳ.</li><li>– Kiểm tra lỗi (Debug) và hoàn thiện hệ thống.</li></ul>

# 1 Giới thiệu

## 1.1 Phân tích Vấn đề (Problem Definition)

- **Bối cảnh thực tiễn tại Việt Nam:**

Trong bối cảnh xã hội Việt Nam đang bước vào giai đoạn già hóa dân số nhanh chóng, số lượng người cao tuổi sống một mình hoặc sống riêng ngày càng gia tăng. Song song với đó, các vấn đề sức khỏe đột ngột như đột quỵ, té ngã, hoặc ngất xỉu trở thành mối lo ngại thường trực, đe dọa trực tiếp đến an toàn và tính mạng nếu không được phát hiện và can thiệp kịp thời.

- **Tính cấp thiết của vấn đề:**

Thời gian “vàng” trong cấp cứu đột quỵ và tai nạn té ngã là yếu tố cực kỳ quan trọng; sự chậm trễ trong việc phát hiện có thể dẫn đến những hậu quả nghiêm trọng về sức khỏe hoặc thậm chí tử vong. Tuy nhiên, việc giám sát thủ công 24/7 đối với người cao tuổi là điều bất khả thi đối với nhiều gia đình hiện nay.

- **Ý nghĩa của đề tài:**

Xuất phát từ nhu cầu thực tiễn đó, nhóm nhận thấy việc ứng dụng công nghệ để giải quyết bài toán này là vô cùng cần thiết và mang tính nhân văn sâu sắc. Ý tưởng về một “Hệ thống giám sát thông minh sử dụng camera và AI để phát hiện tư thế bất thường (ngã, nằm bất động) và tự động gửi cảnh báo” được hình thành nhằm mang lại sự an tâm cho các gia đình có người thân lớn tuổi. Đồng thời, giải pháp này cũng có tiềm năng đóng góp vào việc xây dựng các mô hình chăm sóc sức khỏe thông minh (Smart Healthcare) tại các bệnh viện và viện dưỡng lão.

Về mặt kỹ thuật, bài toán được xác định là bài toán Phát hiện vật thể (Object Detection) và Phân loại hành vi (Action Classification) với đầu vào là hình ảnh từ camera thời gian thực và đầu ra là tọa độ người cùng nhãn hành vi tương ứng (Fall Detected hoặc Non-Fall).

## 1.2 Mục tiêu của Đồ án

Đồ án này hướng tới việc đạt được các mục tiêu cụ thể sau:

- **Xây dựng hệ thống giám sát tự động:** Phát triển một mô hình học máy (Machine Learning/Computer Vision) có khả năng nhận diện chính xác các tư thế bất thường (ngã, nằm bất động) từ video camera trong thời gian thực.
- **Tối ưu hóa độ chính xác:** Mục tiêu quan trọng là mô hình cần phân biệt hiệu quả giữa sự cố ngã và các hành động sinh hoạt thường ngày (như nằm ngủ, tập yoga, ngồi) để giảm thiểu tỷ lệ báo động giả (False Positive), đảm bảo tính tin cậy của hệ thống.

- **Làm chủ quy trình End-to-End:** Nhóm thực hiện và làm chủ toàn bộ quy trình của một dự án AIoT thực tế: từ thu thập và xử lý dữ liệu video, huấn luyện và tinh chỉnh mô hình, đến việc tích hợp API và triển khai thành sản phẩm hoàn chỉnh.
- **Giải quyết bài toán thiết bị hạn chế:** Tìm kiếm và lựa chọn kiến trúc mô hình cân bằng tốt nhất giữa độ chính xác và tốc độ suy luận (inference time) để có thể triển khai trên các thiết bị biên hoặc máy chủ có tài nguyên vừa phải.
- **Hệ thống cảnh báo tin cậy:** Xây dựng cơ chế cảnh báo tự động gửi tin nhắn ngay lập tức đến người thân (qua SMS, Zalo, Telegram) khi phát hiện sự cố.

### 1.3 Tổng quan về Phương pháp

Để giải quyết bài toán trên, nhóm áp dụng cách tiếp cận tổng thể qua các bước sau:

- **Thu thập và Xử lý dữ liệu:**

Nhóm sử dụng bộ dữ liệu chuẩn hóa “Fall Detection Dataset” từ các nguồn công khai (Kaggle/Roboflow) để đảm bảo độ chính xác ban đầu. Quy trình tiền xử lý dữ liệu được tự động hóa bao gồm: khử trùng lặp bằng thuật toán băm ảnh (Perceptual Hashing), lọc nhiễu các nhãn sai lệch, và chuẩn hóa kích thước ảnh đầu vào về  $640 \times 640$  pixels phục vụ cho mô hình YOLO. Dữ liệu được quy hoạch về 3 lớp hành vi chính: Fall Detected, Walking, và Sitting.

- **Lựa chọn và Huấn luyện mô hình:**

Nhóm tập trung nghiên cứu các mô hình thuộc nhóm One-stage Object Detection do ưu điểm về tốc độ xử lý thời gian thực. Ba kiến trúc mô hình hiện đại được lựa chọn để thực nghiệm và so sánh là: YOLOv8n (Baseline - cân bằng tốc độ/chính xác), YOLOv10n (Tối ưu độ trễ bằng cách loại bỏ NMS), và RT-DETR (Sử dụng Transformer để nắm bắt ngữ cảnh toàn cục). Các mô hình được huấn luyện và đánh giá trên cùng một tập dữ liệu và cấu hình siêu tham số để đảm bảo tính công bằng.

- **Triển khai ứng dụng:**

Hệ thống được xây dựng với kiến trúc Client-Server. Luồng video từ camera (Webcam/IP Camera) được đưa vào mô hình học máy đã huấn luyện để phân tích từng khung hình. Khi phát hiện nhãn “Fall Detected”, hệ thống sẽ kích hoạt module cảnh báo để gửi thông báo đến người dùng thông qua API tích hợp.

## 2 Thu thập và Phân tích Dữ liệu

### 2.1 Nguồn và Phương pháp Thu thập

#### 2.1.1 Nguồn dữ liệu

Để đảm bảo tính chuẩn hóa và độ chính xác của nhãn ngay từ giai đoạn đầu, thay vì tự thu thập thủ công tốn kém thời gian và dễ sai sót, nhóm quyết định sử dụng **Bộ dữ liệu công khai (Public Dataset)**.

- **Tên bộ dữ liệu:** Fall Detection Dataset.
- **Nguồn gốc:** Bộ dữ liệu được tác giả UTTEJ KUMAR KANDAGATLA xây dựng và được lưu trữ trên các nền tảng chia sẻ dữ liệu uy tín là Kaggle và Roboflow.
- **Đặc điểm nổi bật:**
  - Dữ liệu bao gồm các hình ảnh đa dạng về bối cảnh môi trường như trong nhà, văn phòng, giúp mô hình học được tính tổng quát cao.
  - Dữ liệu đã được gán nhãn sẵn (Pre-labeled) theo định dạng chuẩn YOLO (Bounding Box + Class ID). Việc sử dụng dữ liệu gán nhãn sẵn giúp tiết kiệm thời gian gán nhãn thủ công và đảm bảo độ chính xác cao cho quá trình huấn luyện.

#### 2.1.2 Quy trình và Công cụ thu thập

Nhóm đã thực hiện quy trình thu thập dữ liệu thông qua các bước cụ thể sau:

- **Khảo sát:** Tiến hành khảo sát các nguồn dữ liệu mở trên Kaggle và Roboflow Universe để tìm kiếm bộ dữ liệu phù hợp với bài toán.
- **Lựa chọn:** Xác định và lựa chọn bộ dữ liệu có định dạng nhãn tương thích trực tiếp với bài toán Object Detection (cụ thể là định dạng YOLO).
- **Tải xuống:** Tải trực tiếp trên Kaggle với định dạng nén .zip để đảm bảo toàn vẹn dữ liệu.
- **Tổ chức dữ liệu:** Giải nén và tổ chức lại cấu trúc thư mục theo tiêu chuẩn để chuẩn bị cho các bước tiền xử lý tiếp theo.

#### 2.1.3 Định dạng lưu trữ

Dữ liệu thu thập được lưu trữ thống nhất theo hai định dạng tệp tin chính, đảm bảo tương thích với các framework huấn luyện phổ biến:

- **Hình ảnh:** Lưu trữ dưới định dạng .jpg (JPEG Image).
- **Nhãn (Labels):** Lưu trữ dưới dạng file văn bản .txt.
  - *Quy tắc:* Mỗi file ảnh sẽ có một file .txt tương ứng cùng tên.
  - *Cấu trúc nội dung:* Chứa thông tin bounding box theo chuẩn YOLO với 5 thông số: *class\_id, x\_center, y\_center, width, height*.

## 2.2 Tiền xử lý và Làm sạch (Preprocessing & Cleaning)

Nhóm đã xây dựng một quy trình tự động hóa (Automated Pipeline) bằng ngôn ngữ Python để chuyển đổi dữ liệu thô thành dữ liệu sạch, đảm bảo chất lượng đầu vào tốt nhất cho mô hình. Các bước thực hiện chi tiết như sau:

### 2.2.1 Xử lý nhiễu và trùng lặp

- **Khử trùng lặp (Deduplication):** Nhóm sử dụng thuật toán băm ảnh Perceptual Hashing (pHash) để mã hóa nội dung ảnh. Các cặp ảnh có mã hash tương đồng (độ giống nhau lớn hơn 95%) được xác định là trùng lặp và bị loại bỏ. Bước này giúp ngăn chặn hiện tượng rò rỉ dữ liệu (Data Leakage) giữa tập huấn luyện và tập kiểm thử.
- **Lọc nhiễu nhãn (Label Noise Filtering):** Hệ thống tự động quét và loại bỏ các bounding box được coi là nhiễu dựa trên các tiêu chí: – Kích thước quá nhỏ (diện tích nhỏ hơn 0.5% diện tích ảnh). – Tọa độ nằm ngoài phạm vi khung hình.

### 2.2.2 Xử lý dữ liệu thiếu và không nhất quán

- **Kiểm tra tính toàn vẹn (Integrity Check):** Thực hiện rà soát song phương giữa thư mục chứa ảnh (images) và thư mục chứa nhãn (labels). Bất kỳ ảnh nào không có file nhãn tương ứng hoặc ngược lại đều bị loại bỏ nhằm đảm bảo tính đầy đủ tuyệt đối (Completeness).
- **Kiểm tra định dạng:** Đảm bảo tất cả các file nhãn đều tuân thủ đúng định dạng 5 giá trị của YOLO. Các file chứa ký tự lạ hoặc sai định dạng số học được tự động xử lý hoặc loại bỏ.

### 2.2.3 Chuẩn hóa và Biến đổi

- **Resize ảnh:** Toàn bộ ảnh đầu vào được đồng bộ kích thước về độ phân giải  $640 \times 640$  pixels. Đây là kích thước tiêu chuẩn (imgsz) cho các mô hình YOLOv8/v10, giúp cân bằng giữa tốc độ huấn luyện và khả năng nhận diện chi tiết.
- **Chuẩn hóa tọa độ:** Các tọa độ bounding box được chuẩn hóa về đoạn  $[0, 1]$  theo công thức tiêu chuẩn của YOLO:  $(X_{center}, Y_{center}, Width, Height)$ .
- **Quy hoạch nhãn (Class Mapping):** Dữ liệu gốc được ánh xạ và chuẩn hóa về 3 lớp hành vi duy nhất để phục vụ bài toán giám sát: – 0: Fall Detected (Sự cố ngã/đột quy). – 1: Walking (Di lại bình thường). – 2: Sitting (Ngồi bình thường).

**Kết quả thực nghiệm tiền xử lý:** Sau khi áp dụng quy trình trên, nhóm đã loại bỏ tổng cộng 12 mẫu nhiễu hoặc trùng lặp từ tập dữ liệu gốc, thu được bộ dữ liệu sạch gồm 473 mẫu, trong đó 363 mẫu cho tập huấn luyện (Train) và 110 mẫu cho tập kiểm tra (Validation), sẵn sàng cho quá trình huấn luyện mô hình.

## 2.3 Phân tích Khám phá Dữ liệu (EDA)

Sau khi hoàn tất quá trình tiền xử lý, nhóm tiến hành phân tích khám phá dữ liệu nhằm hiểu rõ các đặc trưng thống kê, phát hiện các mẫu tiềm ẩn (patterns) và đánh giá độ khó của bài toán.

### 2.3.1 Thông kê mô tả và Phân bố nhãm

Nhóm đã thực hiện thống kê số lượng mẫu (Bounding Boxes) theo từng lớp hành vi để đánh giá mức độ cân bằng dữ liệu.

- **Thông kê số lượng mẫu:** – Fall Detected (Ngã/Dột quy): 283 mẫu (chiếm khoảng 52.8%).
  - Walking (Di lại): 126 mẫu.
  - Sitting (Ngồi): 127 mẫu.
- **Đánh giá độ cân bằng:** Tổng số mẫu thuộc nhóm hành vi bình thường (Walking + Sitting) là 253 mẫu, so với 283 mẫu thuộc nhóm ngã. Tỷ lệ giữa lớp Dương (Fall) và lớp Âm (Non-Fall) xấp xỉ 1.1 : 1. Đây là tỷ lệ khá cân bằng, phù hợp cho việc huấn luyện mô hình mà không cần áp dụng các kỹ thuật oversampling hoặc undersampling phức tạp. Việc tách riêng lớp Sitting cũng giúp mô hình học rõ hơn về tư thế dễ gây nhầm lẫn này.
- **Phân bố theo tập Train/Validation:** Dữ liệu được chia tách theo chiến lược Hold-out (77/23). Kết quả cho thấy tỷ lệ các lớp giữa tập Train và Validation được giữ đồng nhất, đảm bảo kết quả kiểm thử phản ánh đúng hiệu năng thực tế.

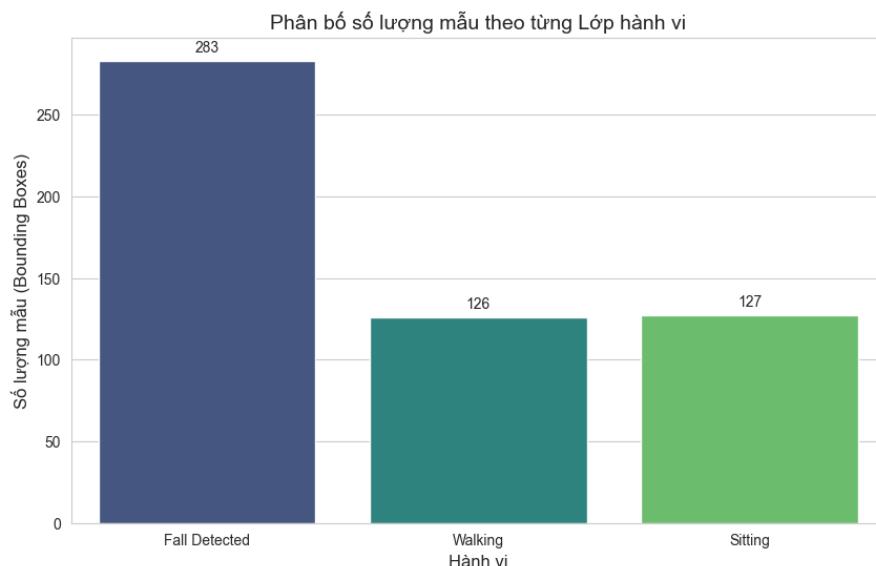


Figure 1: Biểu đồ phân bố số lượng mẫu theo từng lớp hành vi

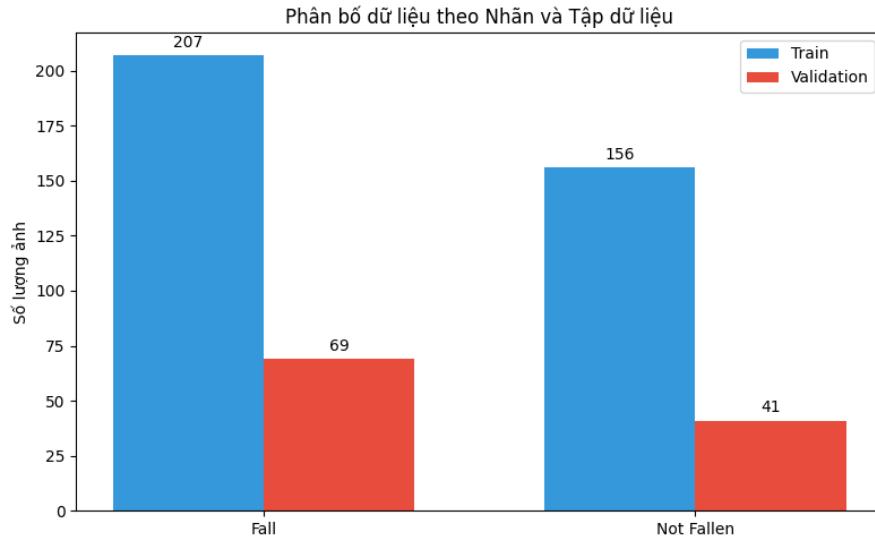


Figure 2: So sánh phân bố dữ liệu theo nhãn giữa tập Train và Validation

### 2.3.2 Phân tích mối quan hệ thuộc tính và Đặc trưng hình ảnh

#### a) Tương quan kích thước khung bao (Width vs. Height):

Nhóm sử dụng biểu đồ phân tán (Scatter Plot) giữa chiều rộng và chiều cao chuẩn hóa của bounding box để tìm kiếm các đặc trưng hình học phân biệt hành vi.

- *Cụm Walking:* Tập trung chủ yếu ở vùng  $Height > Width$ , phù hợp với dáng người cao và hẹp khi đứng hoặc đi lại.
- *Cụm Fall Detected:* Phân tán rộng nhưng có xu hướng nằm ở vùng  $Width \geq Height$ . Khi ngã, cơ thể thường nằm ngang hoặc co lại, làm tăng chiều rộng của bounding box.
- *Cụm Sitting:* Nằm xen kẽ giữa hai cụm trên và được xem là lớp khó (Hard Negative), do có chiều cao thấp giống ngã nhưng trạng thái ổn định giống đi lại.

*Kết luận:* Tỷ lệ khung hình (Aspect Ratio = W/H) là một đặc trưng quan trọng hỗ trợ phân loại hành vi.

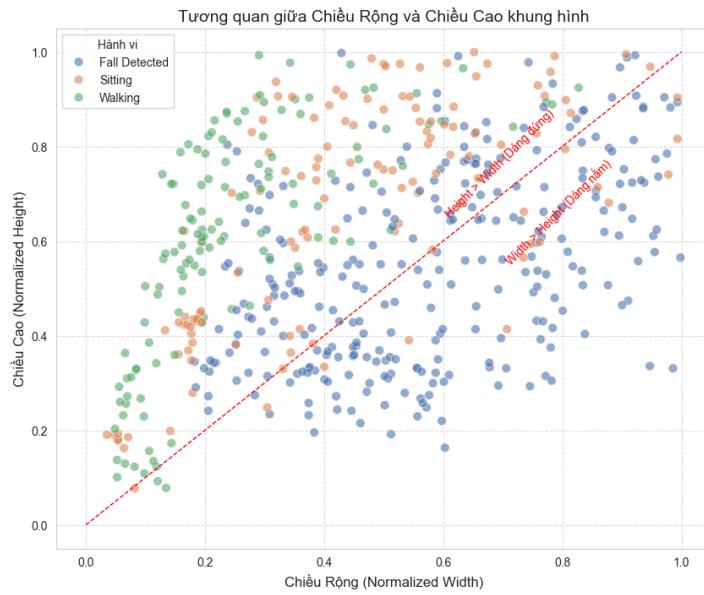


Figure 3: Tương quan chiều rộng và chiều cao bounding box của ba lớp hành vi

**b) Phân tích không gian xuất hiện (Positional Heatmap):**

Biểu đồ nhiệt cho thấy mật độ xuất hiện đối tượng tập trung cao nhất tại khu vực trung tâm ảnh ( $x \approx 0.5, y \approx 0.5$ ). Điều này cho thấy nguy cơ mô hình bị thiên vị vào vùng trung tâm. Để khắc phục, nhóm áp dụng các kỹ thuật tăng cường dữ liệu như dịch chuyển (Translation) và cắt ghép ngẫu nhiên (Mosaic) trong quá trình huấn luyện.

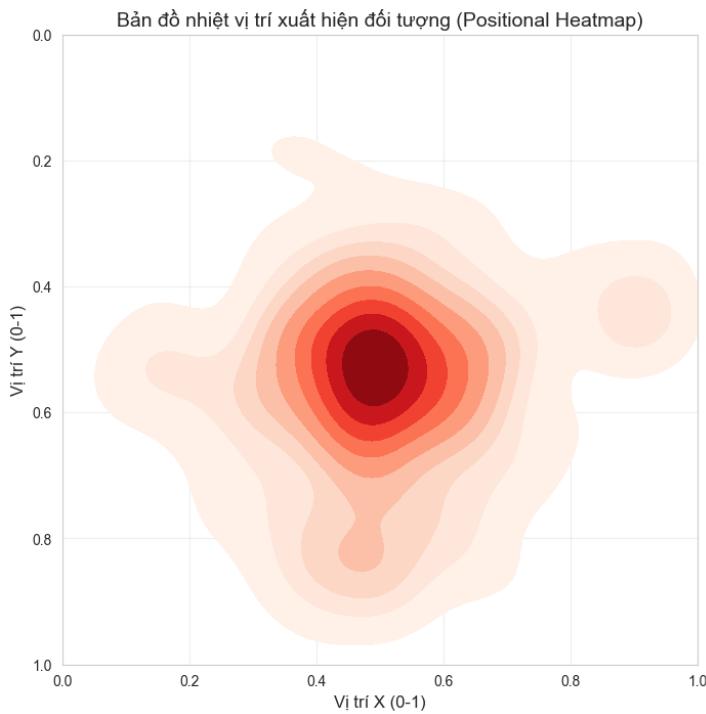


Figure 4: Bản đồ nhiệt thể hiện vị trí xuất hiện của đối tượng trong khung hình

**c) Đặc trưng dữ liệu phi cấu trúc:**

- *Dộ sáng*: Phân bố gần dạng hình chuông với giá trị trung tâm khoảng 150, cho thấy điều kiện ánh sáng nhìn chung ổn định.
- *Dộ sắc nét*: Phần lớn ảnh có độ sắc nét cao, tuy nhiên vẫn tồn tại một số ảnh bị nhòe do chuyển động (Motion Blur). Nhóm quyết định giữ lại các ảnh này nhằm tăng tính thực tế cho mô hình khi xử lý các tình huống ngã nhanh.
- *Đa dạng kích thước*: Diện tích bounding box trải rộng từ 0.05 đến 0.9, cho thấy dữ liệu bao phủ tốt các trường hợp từ xa đến gần camera.

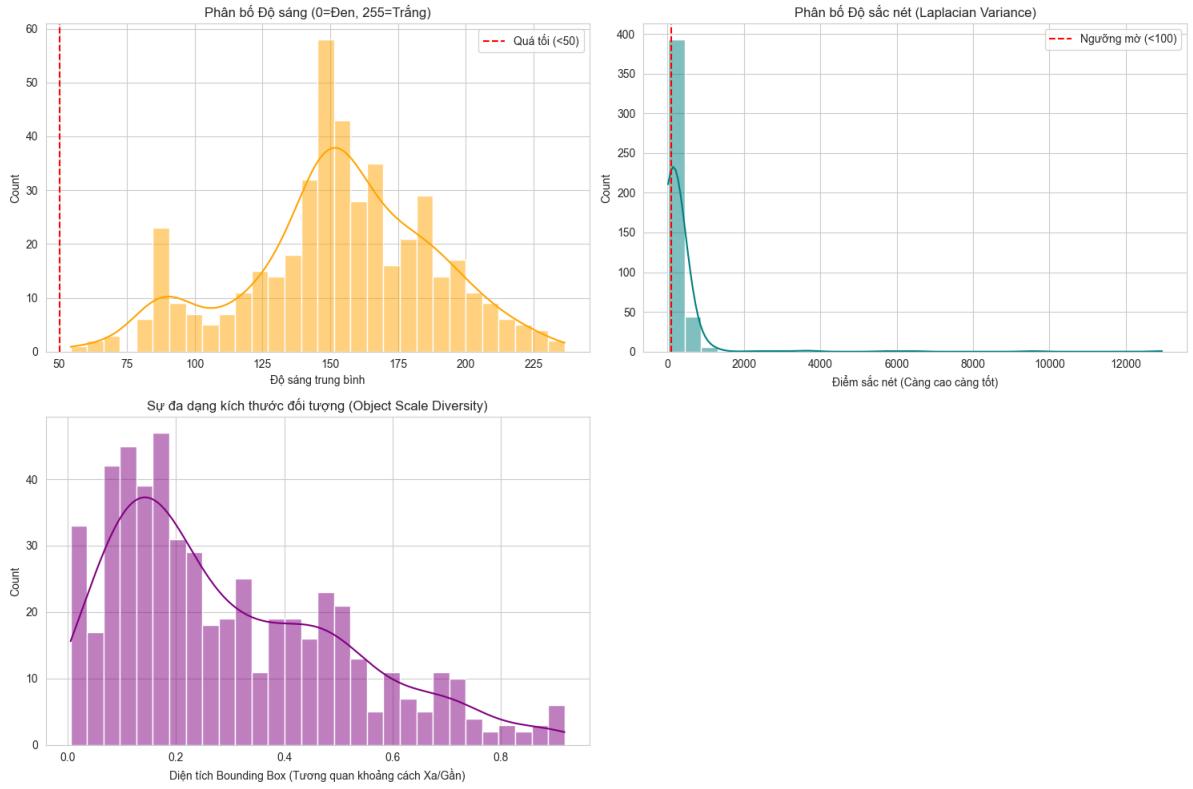


Figure 5: Phân bố độ sáng, độ sắc nét và kích thước đối tượng

### 2.3.3 Phân tích chất lượng dữ liệu và Xử lý vấn đề

Trước khi huấn luyện, nhóm thiết lập các tiêu chí định lượng về mức độ sẵn sàng dữ liệu (Data Readiness) và xử lý triệt để các vấn đề phát sinh.

- Xử lý trùng lặp và rò rỉ dữ liệu:** Sử dụng thuật toán pHash để phát hiện ảnh trùng lặp. Kết quả loại bỏ 11 ảnh trong tập Train và 1 ảnh trong tập Validation có độ tương đồng lớn hơn 95%, đảm bảo không xảy ra hiện tượng rò rỉ dữ liệu (Data Leakage).
- Xử lý ngoại lệ và nhiễu:** Loại bỏ các bounding box có diện tích quá nhỏ (nhỏ hơn 0.5% diện tích ảnh) hoặc có tọa độ nằm ngoài khung hình. Đồng thời, kiểm tra để đảm bảo mọi tọa độ nhãn đều nằm trong khoảng chuẩn hóa  $[0, 1]$ , tránh gây lỗi trong quá trình lan truyền gradient.
- Kiểm tra tự động (Sanity Check):** Tích hợp script kiểm tra tính toàn vẹn cặp ảnh-nhãn với điều kiện `len(images) == len(labels)`, đảm bảo dữ liệu đầu vào đạt mức đầy đủ tuyệt đối trước khi huấn luyện mô hình.

### 3 Lựa chọn và Huấn luyện Mô hình

#### 3.1 Chuẩn bị Dữ liệu cho Mô hình

##### 3.1.1 Phân chia tập dữ liệu (Train/Validation Split)

Để đánh giá khách quan hiệu năng mô hình, nhóm áp dụng chiến lược phân chia dữ liệu theo phương pháp **hold-out** với tỷ lệ cụ thể như sau:

- **Tỷ lệ phân chia:** xấp xỉ 77% cho tập Huấn luyện và 23% cho tập Kiểm định.
- **Tập Huấn luyện (Training Set):** 363 ảnh.
- **Tập Kiểm định (Validation Set):** 110 ảnh.
- **Tổng số ảnh sau làm sạch:** 473 ảnh.

**Lý do lựa chọn tỷ lệ phân chia.** Do tổng kích thước bộ dữ liệu tương đối nhỏ (dưới 500 mẫu), việc dành phần lớn dữ liệu cho huấn luyện là cần thiết nhằm giúp mô hình học được các đặc trưng đa dạng của các tư thế *ngã, đi và ngồi*. Với 77% dữ liệu cho huấn luyện, mô hình có đủ tư liệu để học các mẫu hành vi quan trọng.

Bên cạnh đó, tập Validation gồm 110 ảnh (chiếm 23%) vẫn đảm bảo đủ lớn về mặt thống kê để đại diện cho phân phối dữ liệu thực tế. Tỷ lệ này cho phép giám sát hiệu quả hiện tượng **overfitting** và hỗ trợ quá trình tinh chỉnh siêu tham số (*hyperparameter tuning*) một cách tin cậy, đồng thời không làm lãng phí dữ liệu huấn luyện.

##### 3.1.2 Tóm tắt các bước tiền xử lý cuối cùng (Final Preprocessing)

Trước khi dữ liệu được nạp vào mạng nơ-ron, nhóm thực hiện quy trình chuẩn hóa và tăng cường dữ liệu một cách nghiêm ngặt nhằm đảm bảo chất lượng đầu vào tối ưu.

##### Làm sạch và chuẩn hóa (Cleaning & Normalization).

- **Khử trùng lặp:** Sử dụng thuật toán *pHash* để loại bỏ các ảnh có độ tương đồng trên 95%, qua đó ngăn chặn hiện tượng rò rỉ dữ liệu (*data leakage*) giữa tập Train và Validation.
- **Lọc nhiễu nhăn:** Loại bỏ các bounding box có diện tích quá nhỏ (nhỏ hơn 0.5% diện tích ảnh) hoặc có tọa độ sai lệch.
- **Chuẩn hóa kích thước ảnh:** Toàn bộ ảnh được đồng bộ về kích thước  $640 \times 640$  pixels. Đây là kích thước tối ưu cho các dòng mô hình *YOLOv8/v10* và *RT-DETR*, giúp cân bằng giữa tốc độ xử lý và khả năng phát hiện vật thể nhỏ.
- **Quy hoạch nhăn:** Dữ liệu được ánh xạ thông nhất về ba lớp hành vi:
  - 0: Fall Detected

- 1: Walking
- 2: Sitting

**Tăng cường dữ liệu (Data Augmentation).** Nhằm nâng cao khả năng tổng quát hóa của mô hình trong các điều kiện môi trường phức tạp, nhóm áp dụng các chiến lược tăng cường dữ liệu sau:

- **Mosaic Augmentation (100%):** Ghép ngẫu nhiên bốn ảnh thành một mẫu huấn luyện, giúp mô hình học cách phát hiện đối tượng ở nhiều tỷ lệ kích thước khác nhau.
- **Biến đổi quang học (Photometric Distortions):** Áp dụng *Blur* và *Median Blur* với xác suất  $p = 0.01$  để mô phỏng hiện tượng nhòe do chuyển động nhanh khi ngã. Ngoài ra, chuyển đổi ảnh sang thang xám (*ToGray*) nhằm giúp mô hình tập trung vào hình dáng đối tượng thay vì thông tin màu sắc.
- **Biến đổi hình học (Geometric Distortions):** Áp dụng *Scale* trong khoảng  $\pm 50\%$  và *Translation* 10% để tăng khả năng nhận diện đối tượng ở các khoảng cách khác nhau và giảm hiện tượng mô hình học lệch vào vùng trung tâm ảnh.

## 3.2 Lựa chọn và Kiến trúc Mô hình

Dựa trên đặc thù của bài toán là phát hiện hành động ngã trong thời gian thực (*Real-time Fall Detection*) trên các thiết bị biến hoặc máy chủ có tài nguyên hạn chế, nhóm nghiên cứu tập trung vào các mô hình thuộc nhóm **One-stage Object Detection**. Nhóm lựa chọn ba kiến trúc đại diện để thực nghiệm, bao gồm **YOLOv8n**, **YOLOv10n** và **RT-DETR**.

### 3.2.1 Lý do lựa chọn mô hình

Ba kiến trúc được lựa chọn đại diện cho các hướng tiếp cận khác nhau trong bài toán phát hiện vật thể, nhằm đánh giá toàn diện sự đánh đổi giữa *độ chính xác*, *tốc độ suy luận* và *độ phức tạp mô hình*.

- **YOLOv8 (Nano version):** Đây là kiến trúc *State-of-the-art* phổ biến, nổi bật với sự cân bằng hiệu quả giữa tốc độ xử lý và độ chính xác. Phiên bản Nano (YOLOv8n) được lựa chọn nhằm tối ưu khả năng triển khai trên các thiết bị nhúng và hệ thống chi phí thấp.
- **YOLOv10 (Nano version):** Là phiên bản cải tiến mới nhất (năm 2024), YOLOv10 loại bỏ hoàn toàn bước *Non-Maximum Suppression (NMS)* trong quá trình suy luận. Mô hình này được lựa chọn để kiểm chứng giả thuyết rằng việc loại bỏ NMS giúp giảm độ trễ (*latency*) tối đa cho các hệ thống thời gian thực.

- **RT-DETR (Real-Time Detection Transformer)**: Đại diện cho hướng tiếp cận dựa trên cơ chế *Attention* của Transformer. Việc lựa chọn RT-DETR nhằm khai thác khả năng nắm bắt ngữ cảnh toàn cục (*Global Context*), giúp mô hình phân biệt tốt hơn giữa các hành động có hình thái tương đồng như *ngã* và *ngồi* – một trong những thách thức lớn của bài toán.

### 3.2.2 Kiến trúc chi tiết các mô hình

a) **Kiến trúc YOLOv8n (Baseline).** YOLOv8n được sử dụng làm mô hình nền tảng (baseline) trong quá trình so sánh. Kiến trúc tổng thể gồm ba thành phần chính:

- **Backbone:** Sử dụng mạng *CSPDarknet* cải tiến với module *C2f* (*Cross Stage Partial bottleneck with two convolutions*), giúp cải thiện khả năng truyền gradient và trích xuất đặc trưng so với module C3 trong các phiên bản trước.
- **Neck:** Áp dụng cấu trúc *PANet* (*Path Aggregation Network*) nhằm kết hợp đặc trưng ở nhiều mức phân giải, từ đó tăng khả năng phát hiện đối tượng có kích thước đa dạng.
- **Head:** Thiết kế theo hướng *anchor-free* với *Decoupled Head*, tách biệt nhánh phân loại (*classification*) và nhánh hồi quy khung bao (*bounding box regression*), giúp quá trình học ổn định và hiệu quả hơn.

Mô hình YOLOv8n có khoảng **3.0 triệu tham số**, phù hợp cho các bài toán yêu cầu suy luận nhanh với tài nguyên hạn chế.

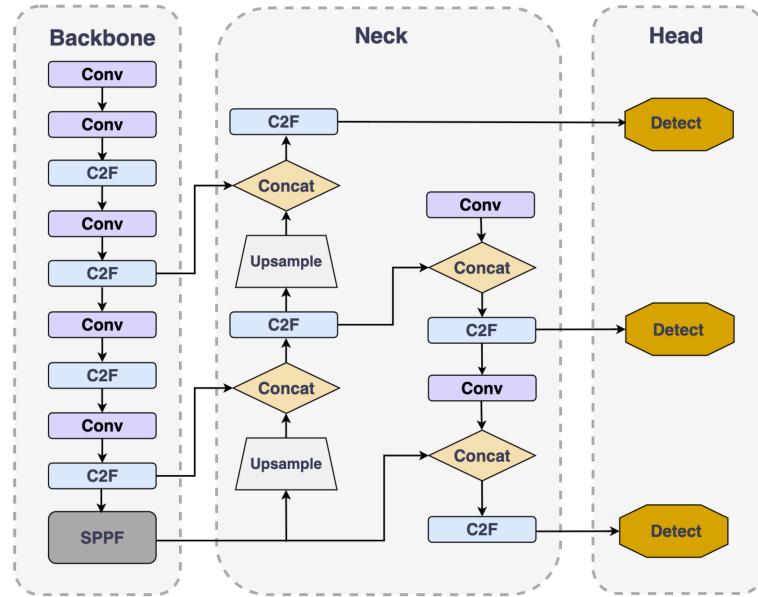


Figure 6: Kiến trúc mạng YOLOv8 với module C2f và Decoupled Head (Nguồn: Luo et al., 2024 [2]).

**b) Kiến trúc YOLOv10n.** YOLOv10n được thiết kế với mục tiêu tối ưu hóa tốc độ suy luận:

- **Cải tiến cốt lõi:** Thay thế cơ chế gán nhãn truyền thống bằng *Dual Label Assignments*. Trong đó, nhánh *one-to-one* được sử dụng khi suy luận, cho phép loại bỏ hoàn toàn bước hậu xử lý *NMS*, vốn tiêu tốn nhiều thời gian.
- **Module kiến trúc:** Tích hợp các module *C2fCIB* và *PSA* (*Partial Self-Attention*) nhằm mở rộng vùng nhận thức (*receptive field*) mà không làm tăng đáng kể chi phí tính toán.

YOLOv10n có khoảng **2.7 triệu tham số**, là mô hình nhẹ nhất trong ba kiến trúc được lựa chọn.

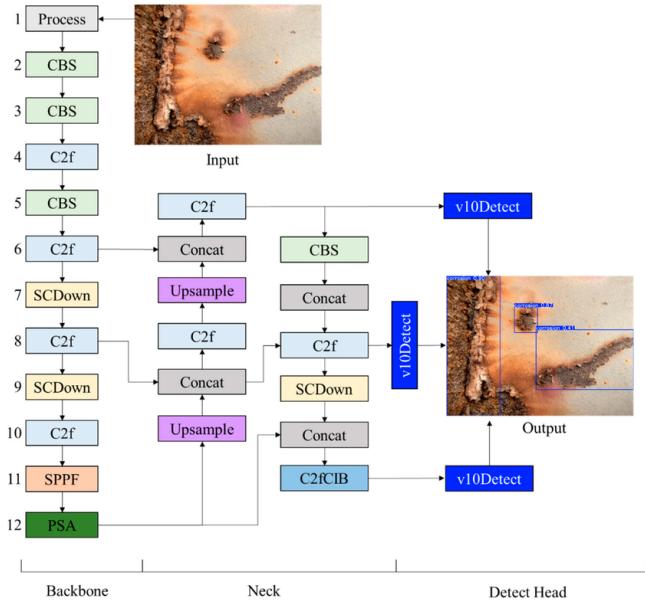


Figure 7: Kiến trúc YOLOv10 với cơ chế suy luận không cần NMS (Nguồn: Cheng & Kang, 2024 [3]).

**c) Kiến trúc RT-DETR.** RT-DETR tiếp cận bài toán phát hiện vật thể theo hướng Transformer thay vì thuần CNN:

- **Hybrid Encoder:** Kết hợp backbone CNN để trích xuất đặc trưng không gian cục bộ và Transformer Encoder (AIFI) nhằm mô hình hóa mối quan hệ toàn cục giữa các vùng trong ảnh.
- **Decoder:** Sử dụng cơ chế *Query Selection*, cho phép dự đoán trực tiếp các bounding box mà không cần sử dụng anchor box truyền thống.

RT-DETR có khoảng **32 triệu tham số**, lớn hơn đáng kể so với các mô hình YOLO phiên bản Nano, nhưng bù lại mang lại khả năng biểu diễn ngữ cảnh toàn cục mạnh mẽ hơn.

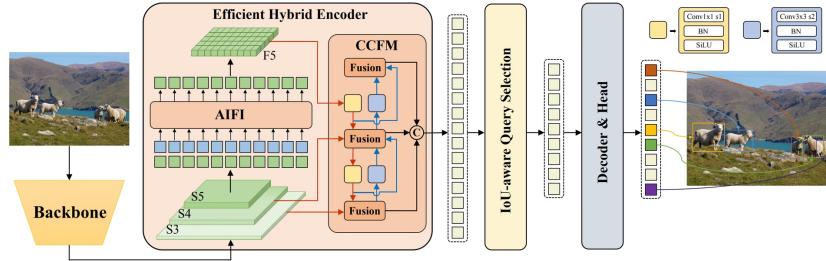


Figure 8: Kiến trúc lai giữa CNN và Transformer của RT-DETR (Nguồn: Lv et al., CVPR 2024 [1]).

### 3.3 Cấu hình Huấn luyện (Training Configuration)

Nhằm đảm bảo tính công bằng và nhất quán trong quá trình so sánh hiệu năng, tất cả các mô hình (**YOLOv8n**, **YOLOv10n**, **RT-DETR**) đều được huấn luyện trên cùng một bộ siêu tham số thống nhất theo chiến lược *Manual Tuning Best Practices*.

#### 3.3.1 Hàm mất mát (Loss Function)

Các mô hình được huấn luyện với tổ hợp ba hàm mất mát tiêu chuẩn, trong đó mỗi thành phần đảm nhiệm một vai trò riêng trong việc tối ưu hóa mạng nơ-ron:

- **Box Loss (IoU / CIoU):** Được sử dụng để đánh giá mức độ chồng lấn giữa khung bao dự đoán và khung bao thực tế (*Ground Truth*). Hàm mất mát này giúp mô hình học cách định vị chính xác vị trí của đối tượng.
- **Classification Loss (BCE – Binary Cross Entropy):** Đo lường độ chính xác trong việc phân loại xác suất hành động giữa các lớp *Fall Detected*, *Walking* và *Sitting*.
- **Distribution Focal Loss (DFL):** Giúp tinh chỉnh ranh giới của bounding box bằng cách mô hình hóa phân phối xác suất của các tọa độ, từ đó cải thiện độ chính xác định vị.

#### 3.3.2 Thuật toán tối ưu và các siêu tham số chính

- **Thuật toán tối ưu (Optimizer):** Sử dụng **AdamW**. Đây là thuật toán được framework *Ultralytics* tự động lựa chọn dựa trên kích thước và đặc thù của tập dữ liệu, giúp quá trình hội tụ ổn định và giảm thiểu hiện tượng *overfitting*.
- **Tỷ lệ học (Learning Rate):** Tỷ lệ học khởi tạo ( $lr_0$ ) được thiết lập là **0.001429**, kết hợp với hệ số động lượng (*momentum*) là **0.9**.

- **Weight Decay:** Thiết lập ở mức **0.0005**. Tham số này đóng vai trò như một cơ chế điều chỉnh (regularization), giúp hạn chế mô hình học quá khớp với dữ liệu huấn luyện.
- **Số vòng lặp (Epochs): 60.** Giá trị này được lựa chọn nhằm đảm bảo mô hình có đủ thời gian hội tụ với tập dữ liệu nhỏ (khoảng 500 ảnh).
- **Batch size: 16.** Phù hợp với giới hạn bộ nhớ VRAM của GPU NVIDIA T4 trên nền tảng Google Colab.
- **Patience (Early Stopping): 20.** Quá trình huấn luyện sẽ tự động dừng nếu không có sự cải thiện về chỉ số đánh giá trong 20 epochs liên tiếp.

### 3.3.3 Chiến lược huấn luyện đặc biệt

Nhóm áp dụng kỹ thuật **Close Mosaic = 10**. Cụ thể, kỹ thuật tăng cường dữ liệu *Mosaic* (ghép bốn ảnh ngẫu nhiên thành một mẫu huấn luyện) sẽ được tắt trong 10 epochs cuối cùng (từ epoch 51 đến 60). Mục tiêu của chiến lược này là giúp mô hình ổn định lại các tham số (*batch normalization statistics*) và học trực tiếp từ các ảnh gốc tự nhiên thay vì ảnh ghép nhân tạo, từ đó cải thiện độ chính xác trong giai đoạn chốt mô hình.

## 3.4 Phương pháp tinh chỉnh tham số

Thay vì sử dụng các phương pháp tìm kiếm siêu tham số tự động tốn kém tài nguyên tính toán như *Grid Search* hay *Random Search*, nhóm nghiên cứu áp dụng phương pháp **Manual Tuning** dựa trên quan sát thực nghiệm.

Quy trình tinh chỉnh được thực hiện theo các bước sau:

- **Thiết lập Baseline:** Khởi đầu bằng việc huấn luyện mô hình với cấu hình mặc định do framework *Ultralytics* cung cấp nhằm thiết lập mốc hiệu năng so sánh ban đầu.
- **Điều chỉnh số Epochs:** Thông qua việc quan sát biểu đồ *Training Loss* và *Validation Loss* của các lần chạy thử, nhóm nhận thấy mô hình có xu hướng hội tụ ổn định trong khoảng 40–50 epochs. Do đó, số epochs được chốt ở mức 60 để đảm bảo hội tụ hoàn toàn mà vẫn tiết kiệm thời gian huấn luyện.
- **Điều chỉnh chiến lược Augmentation:** Qua thực nghiệm, nhóm nhận thấy dữ liệu gốc phản ánh sát thực tế hơn so với dữ liệu ghép từ *Mosaic* trong giai đoạn cuối. Vì vậy, tham số `close_mosaic = 10` đã được bổ sung nhằm tinh chỉnh và tối ưu hóa chỉ số *mAP*.

## 4 Kết quả và Thảo luận

### 4.1 Kết quả Thực nghiệm

Để đảm bảo tính công bằng, tất cả các mô hình (**YOLOv8n**, **YOLOv10n**, **RT-DETR**) đều được huấn luyện trong **60 epochs** với cùng một cấu hình siêu tham số. Các kết quả dưới đây được ghi nhận trên tập *Validation* (đóng vai trò là tập *Test* trong nghiên cứu này).

#### 4.1.1 Kết quả mô hình YOLOv8n (Baseline)

Biểu đồ quá trình học (Learning Curves).

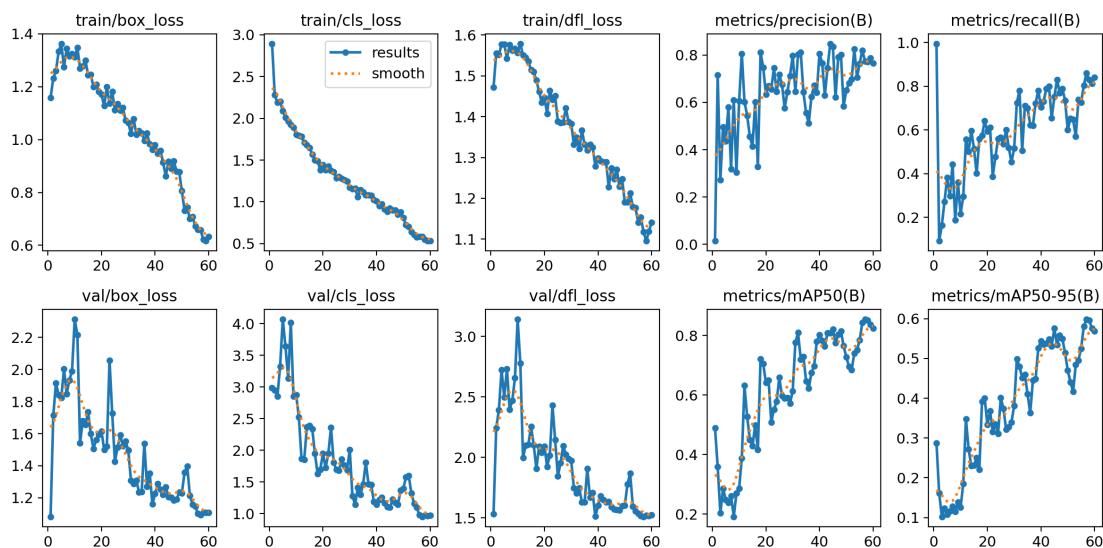


Figure 9: Biểu đồ Loss và Metrics của mô hình YOLOv8n qua 60 epochs

Nhận xét:

- **Hội tụ:** Các hàm mất mát (*train/box\_loss*, *train/cls\_loss*) giảm đều đặn và mượt mà. Đặc biệt, *train/cls\_loss* giảm mạnh từ 2.89 xuống 0.53, cho thấy mô hình học rất hiệu quả.
- **Ôn định:** Trên tập Validation, trong 30 epochs đầu xuất hiện dao động mạnh (giai đoạn tìm kiếm), nhưng từ epoch 50 trở đi, các chỉ số đi vào quỹ đạo ổn định và giảm sâu, chứng tỏ mô hình không bị *overfitting* nghiêm trọng.

Các chỉ số đánh giá (Metrics). Kết quả tốt nhất được ghi nhận tại epoch 57:

Chỉ số	Giá trị	Nhận xét
Precision	0.774	Độ chính xác ở mức khá
Recall	0.861	Độ nhạy cao, mô hình ít bỏ sót đối tượng
mAP@50	0.854	Hiệu năng tổng thể tốt nhất
mAP@50–95	0.598	Định vị khung bao chặt chẽ

### Phân tích Ma trận nhầm lẫn (Confusion Matrix).



Figure 10: Ma trận nhầm lẫn chuẩn hóa của mô hình YOLOv8n

- **Lớp Fall Detected:** Đạt Recall rất cao (0.93). Chỉ có khoảng 1% trường hợp bị nhận nhầm thành nền (*Background*), đảm bảo tính an toàn cho ứng dụng cảnh báo.
- **Lớp Walking:** Nhận diện tốt với độ chính xác đạt 0.917.
- **Lớp Sitting:** Là lớp yếu nhất (Recall 0.74), thường bị nhầm lẫn với Walking (16%) hoặc bị bỏ sót (11%) do sự tương đồng về hình thái phần thân trên.

#### 4.1.2 Kết quả mô hình YOLOv10n

Biểu đồ quá trình học (Learning Curves).

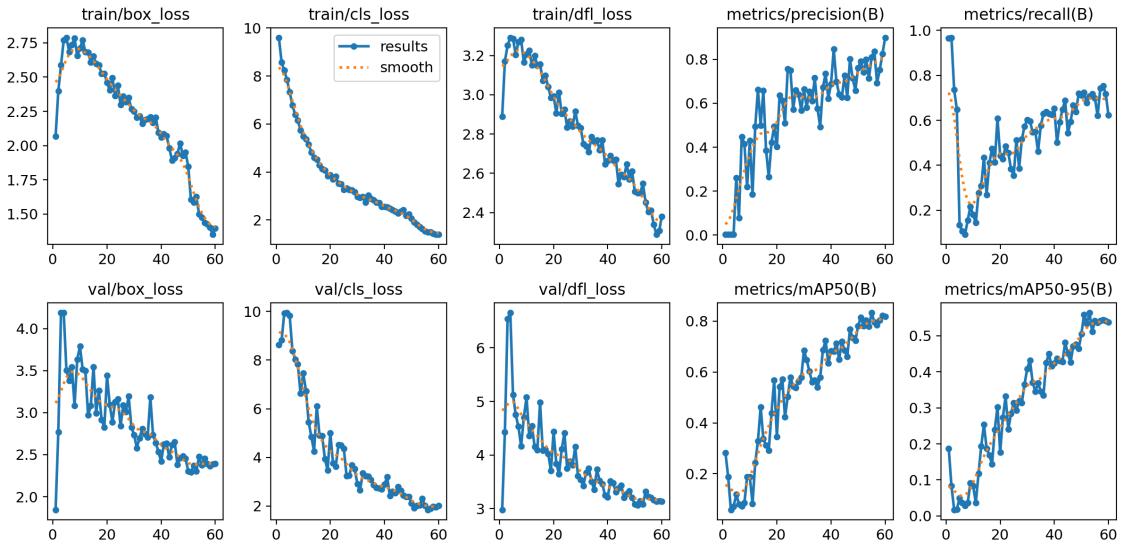


Figure 11: Biểu đồ Loss và Metrics của mô hình YOLOv10n qua 60 epochs

**Nhận xét:** Mô hình hội tụ rất nhanh trong giai đoạn đầu nhờ cơ chế *Dual Assignments*. Tuy nhiên, *val/cls\_loss* có dấu hiệu bão hòa sớm ở mức khoảng 2.00 sau epoch 50, cho thấy mô hình gặp khó khăn trong việc tối ưu thêm và xuất hiện hiện tượng quá khớp cục bộ.

**Các chỉ số đánh giá (Metrics).** Kết quả tốt nhất được ghi nhận tại epoch 53:

Chỉ số	Giá trị	Nhận xét
Precision	0.799	Cao hơn YOLOv8n
Recall	0.707	Thấp nhất, bỏ sót nhiều sự kiện
mAP@50	0.804	Thấp hơn YOLOv8n
Tốc độ suy luận	3.4 ms	Nhanh nhất nhờ loại bỏ NMS

**Phân tích Ma trận nhầm lẫn (Confusion Matrix).**

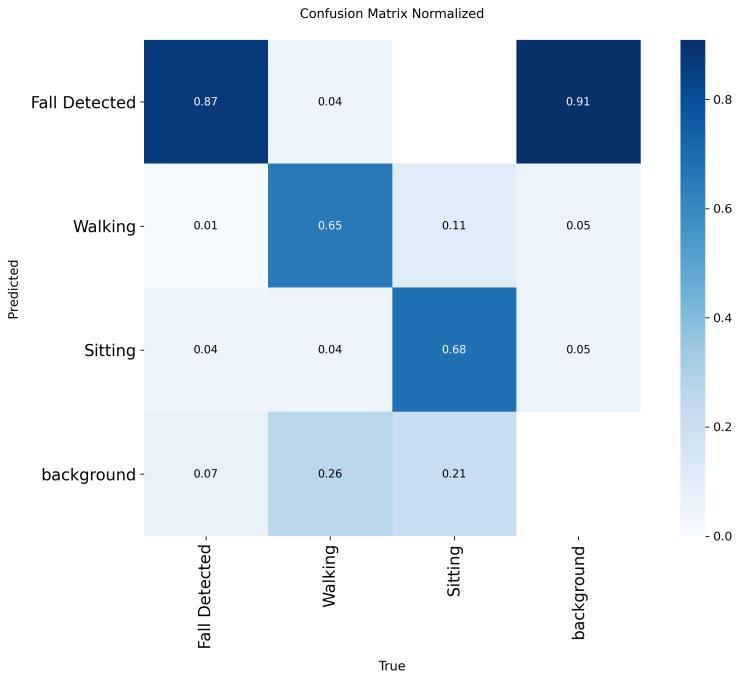


Figure 12: Ma trận nhầm lẫn chuẩn hóa của mô hình YOLOv10n

- **Báo động giả nghiêm trọng:** Có tới 91% các mẫu thuộc lớp nền bị dự đoán nhầm thành *Fall Detected*. Điều này khiến hệ thống liên tục phát cảnh báo sai khi không có người.
- **Lớp Fall Detected:** Recall hiển thị ở mức 0.87, tuy nhiên chỉ số này chịu ảnh hưởng bởi hiện tượng mô hình dự đoán thiên lệch quá mức vào lớp ngã.

#### 4.1.3 Kết quả mô hình RT-DETR (Large)

Biểu đồ quá trình học (Learning Curves).

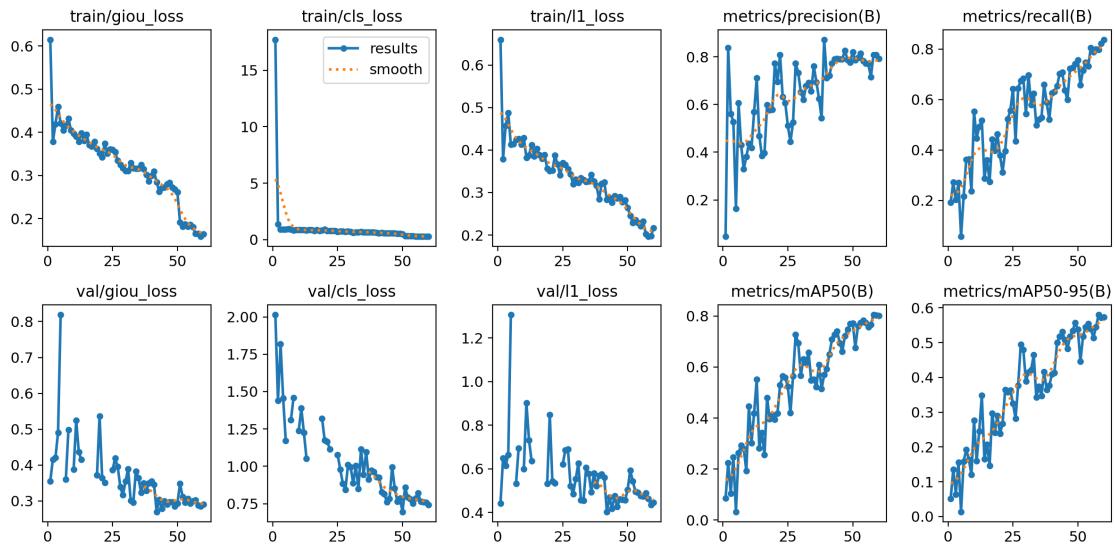


Figure 13: Biểu đồ Loss và Metrics của mô hình RTDETR qua 60 epochs

**Nhận xét:** Đường cong Loss giảm ổn định và ít dao động hơn so với các mô hình CNN thuần túy nhờ cơ chế *Attention* toàn cục. Khoảng cách giữa Train Loss và Validation Loss nhỏ, cho thấy khả năng tổng quát hóa tốt.

**Các chỉ số đánh giá (Metrics).** Kết quả tốt nhất được ghi nhận tại epoch 58:

Chỉ số	Giá trị	Nhận xét
Precision	0.808	Cao nhất, độ tin cậy dự đoán tốt
Recall	0.797	Mức bao phủ sự kiện khá
mAP@50	0.805	Tương đương YOLOv10n
mAP@50–95	0.580	Dịnh vị tốt nhờ Transformer

Phân tích Ma trận nhầm lẫn (Confusion Matrix).

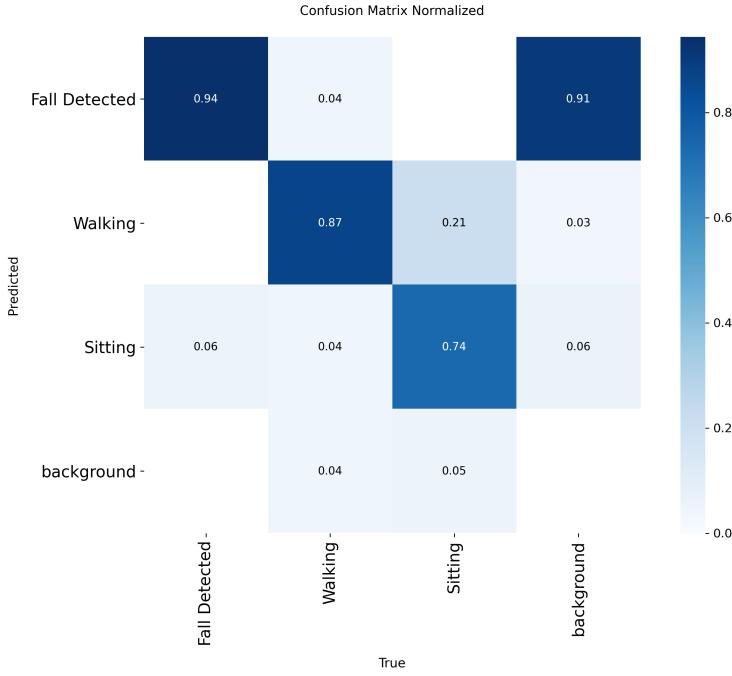


Figure 14: Ma trận nhầm lẫn chuẩn hóa của mô hình RTDETR

- **Lớp Fall Detected:** Đạt Recall rất cao (0.94), tương đương YOLOv8n.
- **Vấn đề Background:** Tương tự YOLOv10n, RT-DETR gặp khó khăn nghiêm trọng trong việc phân loại lớp nền, với khoảng 91% mẫu nền bị nhận nhầm thành *Fall Detected*. Đây là hạn chế lớn và cần được khắc phục bằng cách tăng cường dữ liệu mẫu âm (*Negative Samples*).

## 4.2 So sánh và Thảo luận

### 4.2.1 Bảng so sánh hiệu năng các mô hình

Để đảm bảo tính công bằng, các mô hình được đánh giá trên cùng tập Validation với phần cứng GPU Tesla T4. Bảng dưới đây tổng hợp các chỉ số hiệu năng cốt lõi.

Tiêu chí	YOLOv8n	YOLOv10n	RT-DETR (L)
Số tham số (Params)	3.0 M	2.7 M (Nhẹ nhất)	32.8 M
Độ phức tạp (GFLOPs)	8.1	6.5	103.4
Thời gian suy luận	4.6 ms	3.4 ms (Nhanh nhất)	17.9 ms
Precision	0.774	0.799	0.809
Recall (Tổng thể)	0.861	0.708	0.801
mAP@50	0.854	0.803	0.805
Recall (Lớp Fall Detected)	0.930	0.803	0.915

#### Nhận xét:

**YOLOv8n (Sự cân bằng tối ưu):** Đạt mAP@50 và Recall lớp Ngã cao nhất (93%). Đây là ứng cử viên sáng giá nhất cho bài toán an toàn y tế vì giảm thiểu tối đa rủi ro bỏ sót sự cố.

**YOLOv10n (Tối ưu tốc độ):** Nhờ loại bỏ NMS, mô hình đạt tốc độ xử lý nhanh nhất ( $3.4\text{ ms/nh}$ ). Tuy nhiên, sự đánh đổi về độ nhạy (Recall thấp hơn khoảng 13% so với YOLOv8n) là quá lớn đối với yêu cầu giám sát người cao tuổi.

**RT-DETR (Kiến trúc Transformer):** Có Precision cao nhất, ít báo động giả hơn về lý thuyết, nhưng chi phí tính toán quá lớn (gấp khoảng 12 lần YOLOv8n), không phù hợp cho các thiết bị biên hạn chế tài nguyên.

#### 4.2.2 Phân tích hiện tượng Overfitting/Underfitting

Dựa trên sự biến thiên của hàm mất mát (Loss) và chỉ số mAP giữa tập Train và Validation, nhóm đưa ra các phân tích sau:

**YOLOv8n – Trạng thái Good Fit (Tốt):** Hiện tượng: Đường cong Loss giảm đều đặn trên cả hai tập dữ liệu. Độ chênh lệch (gap) giữa Train và Val nhỏ. Dấu hiệu: Có dấu hiệu Overfitting rất nhẹ sau epoch 50 (Val loss đi ngang). Biện pháp xử lý: Nhóm đã áp dụng cơ chế Early Stopping (patience = 20) và các kỹ thuật regularization (như Weight Decay 0.0005) để kiểm soát, giúp mô hình duy trì khả năng tổng quát hóa tốt.

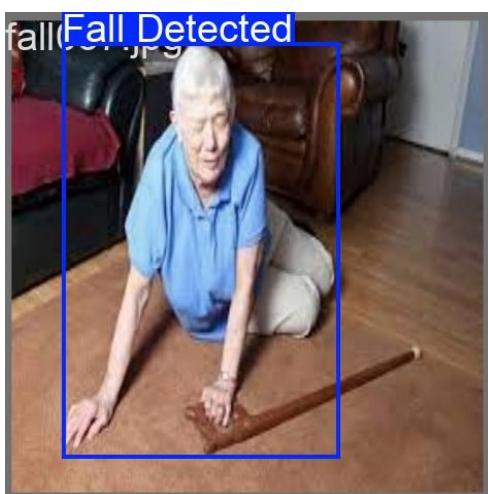
**YOLOv10n – Overfitting cục bộ:** Hiện tượng: train/cls\_loss giảm rất sâu (xuống 1.39) nhưng val/cls\_loss lại bão hòa ở mức cao (2.00) ngay từ epoch 50. Phân tích: Mô hình có xu hướng “học vẹt” các đặc trưng của hành động ngã, dẫn đến việc nhận diện sai nhiều vùng nền (background) thành hành vi ngã. Nguyên nhân: Kiến trúc Dual-Assignments đường như yêu cầu lượng dữ liệu lớn hơn mức 500 ảnh hiện có để hoạt động hiệu quả.

**RT-DETR – Khả năng tổng quát hóa tốt (Robust):** Hiện tượng: Đường cong Validation Loss ổn định và ít dao động nhất. Khoảng cách giữa Train và Val là nhỏ nhất trong ba mô hình. Nguyên nhân: Cơ chế Attention toàn cục của Transformer giúp mô hình học được ngữ nghĩa thực sự thay vì chỉ ghi nhớ các chi tiết cục bộ.

#### 4.2.3 Phân tích các trường hợp sai điển hình (Error Analysis)

Nhóm đã trực quan hóa kết quả dự đoán và xác định ba nhóm nguyên nhân lỗi chính:

**Trường hợp 1: Lỗi bỏ sót ngã (False Negative) do tư thế phục hồi**



(a) Nhãn thật (Ground Truth): Fall Detected



(b) Dự đoán (Prediction): Sitting (0.50)

Figure 15: Ví dụ lỗi False Negative do tư thế phục hồi sau ngã

- **Mô tả:** Nạn nhân sau khi ngã đang cố gắng ngồi dậy hoặc với tay lấy đồ.
- **Dự đoán sai:** Mô hình nhận diện là Sitting (Ngồi) thay vì Fall Detected.
- **Giả thuyết nguyên nhân:** Do mô hình 2D chủ yếu dựa vào tỷ lệ khung hình (Aspect Ratio) và hướng trực cơ thể. Khi nạn nhân ngồi dậy, phần thân trên thẳng đứng giống hệt tư thế ngồi thông thường, gây nhầm lẫn về mặt hình học.

#### Trường hợp 2: Lỗi do môi trường phức tạp (Cầu thang)



(a) Nhãn thật (Ground Truth): Fall Detected



(b) Dự đoán (Prediction): Sitting (0.70)

Figure 16: Ví dụ lỗi False Negative do môi trường cầu thang

- **Mô tả:** Người ngã trên khu vực cầu thang.

- **Dự đoán sai:** Mô hình nhận diện là Sitting.
- **Giả thuyết nguyên nhân:** Cấu trúc bậc thang gấp khúc tạo ra nhiễu nền (Background Clutter). Tư thế người bị gập theo bậc thang (đầu gối co, lưng tựa) có sự chồng lấn đặc trưng (feature overlap) lớn với tư thế ngồi nghỉ.

### Trường hợp 3: Lỗi phân loại sai hành động (Misclassification)



(a) Nhãn thật (Ground Truth): Sitting

(b) Dự đoán (Prediction): Walking (0.90)

Figure 17: Ví dụ lỗi phân loại sai hành động sinh hoạt thường nhật

- **Mô tả:** Người đang ngồi trên ghế đá, vắt chéo chân.
- **Dự đoán sai:** Mô hình nhận diện là Walking (Di lại).
- **Giả thuyết nguyên nhân:** Vị trí hai đầu gối so le khi vắt chân tạo ra hình dáng giống bước chân đang di chuyển. Ngoài ra, việc ghế ngồi bị khuất hoặc hòa lẫn vào nền khiến mô hình thiếu ngữ cảnh “đang ngồi trên ghế”.

**Tổng kết nguyên nhân:** Các sai số chủ yếu đến từ hạn chế của ảnh tĩnh 2D (thiếu thông tin thời gian/chuyển động) và sự chồng lấn về đặc trưng hình học giữa các tư thế trong bối cảnh phức tạp.

## 5 Xây dựng và Triển khai Ứng dụng

### 5.1 Kiến trúc Hệ thống

Để hiện thực hóa giải pháp giám sát an toàn, nhóm thiết kế kiến trúc hệ thống tổng thể theo mô hình AIoT (AI + Internet of Things). Tuy nhiên, trong phạm vi đồ án và do giới hạn về thời gian cũng như kinh phí đầu tư phần cứng chuyên dụng, hệ thống được triển khai ở quy mô **thực nghiệm mô phỏng** (Experimental Simulation).

Mô hình này tận dụng tài nguyên sẵn có của máy tính cá nhân để giả lập các thành phần của một hệ thống thực tế, tập trung vào việc chứng minh **tính đúng đắn** của **giải thuật** và **độ chính xác** của **mô hình AI**.

### 5.1.1 Sơ đồ kiến trúc thực nghiệm

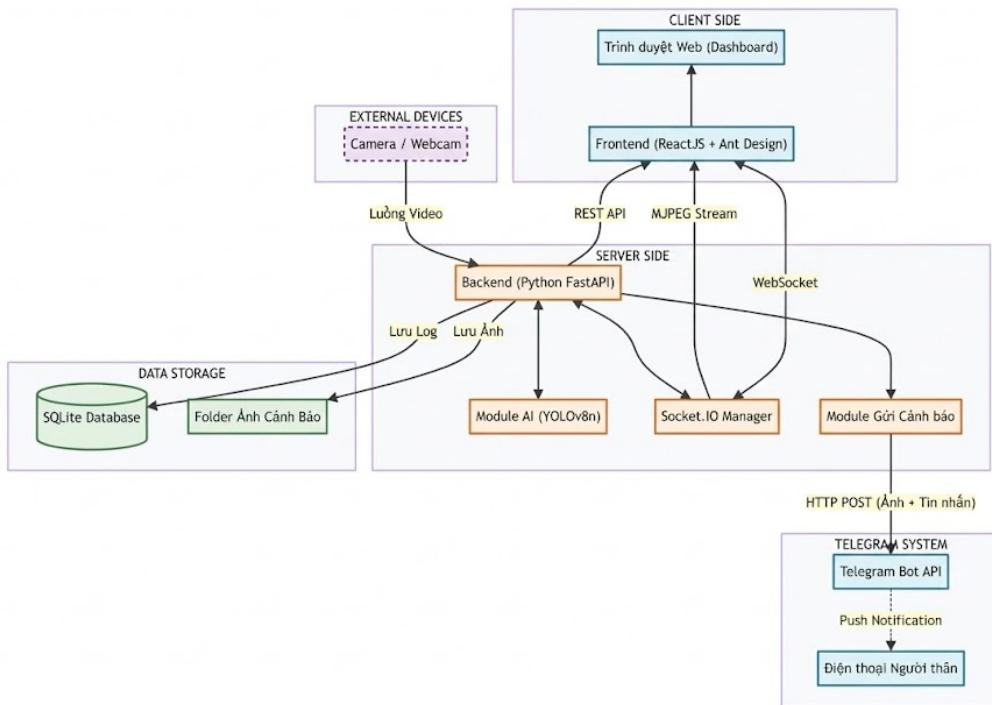


Figure 18: Sơ đồ kiến trúc hệ thống

Kiến trúc tổng thể của hệ thống được thiết kế theo mô hình phân lớp (Layered Architecture), bao gồm các khối chức năng chính như sau:

#### a) Khối Thiết bị Ngoại vi (External Devices)

**Camera/Webcam** đóng vai trò là “mắt” của hệ thống. Trong môi trường thực nghiệm, nhóm sử dụng Webcam kết nối trực tiếp qua cổng USB để thu nhận luồng video (Video Stream) ổn định và truyền dữ liệu hình ảnh về máy chủ xử lý trung tâm theo thời gian thực.

#### b) Khối Máy chủ Xử lý (Server Side – Backend)

Đây là trung tâm điều phối và xử lý logic chính của hệ thống, được xây dựng trên nền tảng **Python FastAPI**. Backend bao gồm các module chính sau:

- **Module AI (YOLOv8n):** Thực hiện suy luận (inference) trên từng khung hình (frame) video để phát hiện hành vi ngã.
- **Socket.IO Manager:** Quản lý kết nối thời gian thực giữa Backend và Frontend, cho phép đẩy thông báo cảnh báo (Popup) ngay lập tức mà không cần tải lại trang.

- **Module Gửi Cảnh báo:** Khi phát hiện sự cố, module này tự động chụp ảnh hiện trường và kích hoạt quy trình gửi cảnh báo đến hệ thống Telegram.

#### c) Khối Lưu trữ Dữ liệu (Data Storage)

Hệ thống sử dụng cơ chế lưu trữ lai nhằm tối ưu hiệu năng và tính linh hoạt:

- **SQLite Database:** Lưu trữ nhật ký hoạt động, thông tin người dùng và lịch sử các lần cảnh báo (thời gian, độ tin cậy).
- **File System:** Lưu trữ vật lý các hình ảnh bằng chứng (Evidence Images) phục vụ việc xem lại và kiểm tra sự cố.

#### d) Khối Giao diện Người dùng (Client Side – Frontend)

Khối này là nơi người dùng trực tiếp tương tác với hệ thống, được phát triển bằng **ReactJS**. Các chức năng chính bao gồm:

- **Dashboard:** Hiển thị luồng video trực tiếp (MJPEG Stream) và các thông báo trạng thái của hệ thống.
- **Tương tác người dùng:** Cho phép xem lại lịch sử cảnh báo và quản lý các cài đặt hệ thống.

#### e) Khối Hệ thống Cảnh báo Từ xa (Telegram Notification System)

Đây là thành phần quan trọng giúp người giám hộ nhận được thông báo ngay cả khi không trực tiếp theo dõi hệ thống.

- **Telegram Bot API:** Hệ thống tích hợp với nền tảng đám mây của Telegram để gửi thông báo.
- **Cơ chế hoạt động:** Khi Backend gửi yêu cầu HTTP POST chứa hình ảnh sự cố và nội dung cảnh báo, Telegram Server sẽ xử lý và đẩy thông báo dạng Push Notification trực tiếp đến điện thoại của người dùng đã đăng ký với Bot. Cơ chế này đảm bảo việc truyền tải thông tin gần như tức thời và có độ tin cậy cao.

### 5.1.2 Luồng dữ liệu (Data Flow)

Quy trình xử lý dữ liệu trong hệ thống diễn ra tuần tự theo các bước sau:

1. **Capture:** Webcam thu hình liên tục ở độ phân giải VGA ( $640 \times 480$ ) nhằm tối ưu tốc độ truyền tải.
2. **Inference:** Mỗi khung hình được đưa vào mô hình YOLOv8n để suy luận.
3. **Analysis:** Nếu nhận định là *Fall Detected* trong một khoảng thời gian xác định, hệ thống kích hoạt trạng thái cảnh báo.
4. **Action:** Gửi thông báo đến Web, Telegram và lưu log sự kiện, hình ảnh vào lịch sử.

## 5.2 Đóng gói và Tích hợp Mô hình

Việc chuyển giao mô hình từ giai đoạn huấn luyện sang giai đoạn triển khai được thực hiện thông qua quy trình đóng gói và tích hợp chuẩn hóa.

### 5.2.1 Đóng gói trọng số

Sau khi hoàn tất quá trình huấn luyện và đánh giá trên tập Validation, file trọng số tốt nhất `yolov8n.pt` được trích xuất với các đặc điểm sau:

- **Định dạng:** PyTorch Serialized Model (.pt).
- **Nội dung:** Chứa toàn bộ kiến trúc mạng nơ-ron và các tham số đã được tối ưu để nhận diện ba lớp hành vi: *Fall*, *Walking*, và *Sitting*.

File này đóng vai trò là **hạt nhân (kernel)** của module nhận diện và có thể được nạp lại dễ dàng trong mọi môi trường Python hỗ trợ thư viện Ultralytics.

### 5.2.2 Tích hợp vào Ứng dụng

Mô hình được tích hợp vào Backend thông qua thư viện `ultralytics` kết hợp với OpenCV. Quy trình tích hợp xử lý sự khác biệt giữa môi trường huấn luyện và môi trường triển khai như sau:

- **Khởi tạo:** Hệ thống nạp file `yolov8n.pt` vào bộ nhớ (RAM/VRAM) ngay khi khởi động ứng dụng nhằm giảm độ trễ cho các lần suy luận tiếp theo.
- **Tiền xử lý tự động:** Mặc dù Webcam cung cấp ảnh ở độ phân giải  $640 \times 480$  (tỷ lệ 4:3), mô hình được huấn luyện trên ảnh vuông  $640 \times 640$ . Nhóm sử dụng kỹ thuật **Letterboxing** tích hợp sẵn trong thư viện dự đoán để tự động thêm viền đậm, đảm bảo giữ nguyên tỷ lệ khung hình và không làm méo ảnh. Điều này giúp mô hình hoạt động nhất quán so với giai đoạn huấn luyện.
- **Hậu xử lý:** Kết quả đầu ra từ mô hình (tọa độ bounding box, class ID và confidence) được giải mã và vẽ trực tiếp lên khung hình video để hiển thị trên Frontend.

## 5.3 Giao diện và Chức năng

Hệ thống cung cấp một giao diện web trực quan, thân thiện và tập trung vào trải nghiệm **giám sát thời gian thực**.

### 5.3.1 Màn hình Đăng nhập và Đăng ký

**Chức năng:** Dảm bảo tính bảo mật, chỉ người dùng đã đăng ký mới được quyền truy cập vào hệ thống giám sát.

**Mô tả:** Giao diện đơn giản gồm các trường nhập **Username**, **Password**. Hệ thống có xác thực dữ liệu đầu vào và thông báo lỗi nếu sai thông tin.

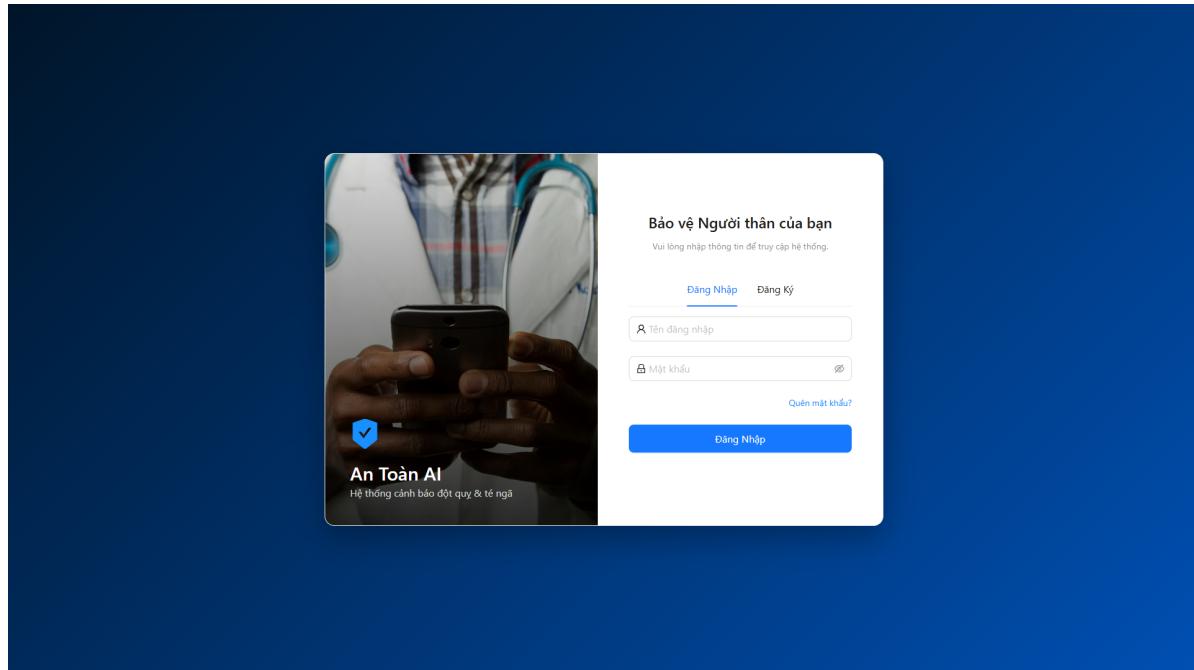


Figure 19: Trang Đăng nhập

### 5.3.2 Màn hình Dashboard (Giám sát trung tâm)

**Chức năng:** Đây là màn hình chính của ứng dụng.

- Hiển thị luồng video trực tiếp từ camera.
- Thông kê nhanh trạng thái hệ thống và số lượng cảnh báo trong ngày.
- Cung cấp liên kết nhanh đến bot **Telegram** để nhận cảnh báo từ xa.

**Mô tả:** Bố cục chia thành các thẻ (**Card**) thông kê phía trên và khung hình video lớn ở trung tâm.

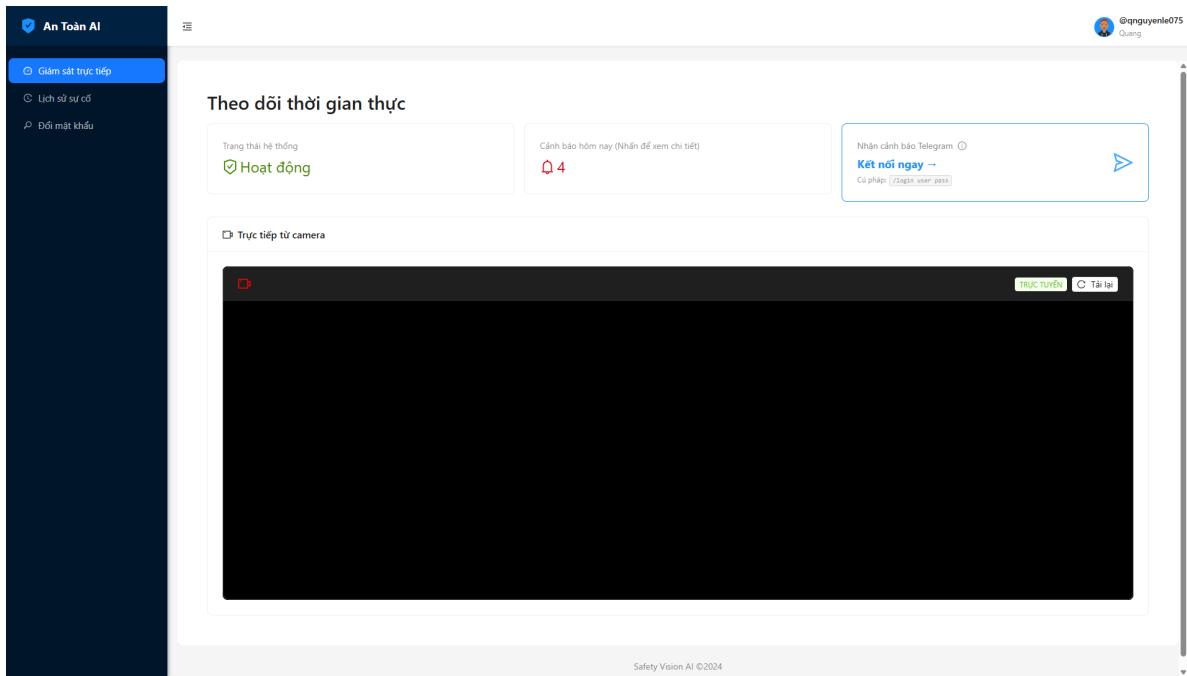


Figure 20: Dashboard Giám sát Trung tâm

### 5.3.3 Chức năng Cảnh báo Đột quy/Ngã (Alert Popup)

**Chức năng:** Đây là tính năng quan trọng nhất của hệ thống. Khi AI phát hiện hành động ngã:

- Một cửa sổ (Popup) màu đỏ bật lên ngay lập tức, đè lên mọi giao diện khác.
- Phát âm thanh cảnh báo liên tục để thu hút sự chú ý.
- Hiển thị hình ảnh chụp lại khoảnh khắc người bị ngã và thời gian xảy ra.

**Mô tả:** Giao diện cảnh báo được thiết kế với màu đỏ chủ đạo, nút tắt cảnh báo lớn để người giám sát dễ dàng thao tác.



Figure 21: Cửa sổ Cảnh báo Đột quy/Ngã

#### 5.3.4 Màn hình khác

- **Lịch sử:** Hiển thị danh sách các lần cảnh báo trong quá khứ dưới dạng bảng (Table), cho phép xem lại thời gian và hình ảnh.
- **Đổi mật khẩu:** Cho phép người dùng cập nhật mật khẩu để bảo vệ tài khoản.
- **Thông tin tài khoản:** Cho phép người dùng xem và thay đổi Họ tên và số điện thoại cùng với việc xem tên đăng nhập.
- **Hỗ trợ:** Cho phép người dùng xem các thông tin liên quan tới ứng dụng.

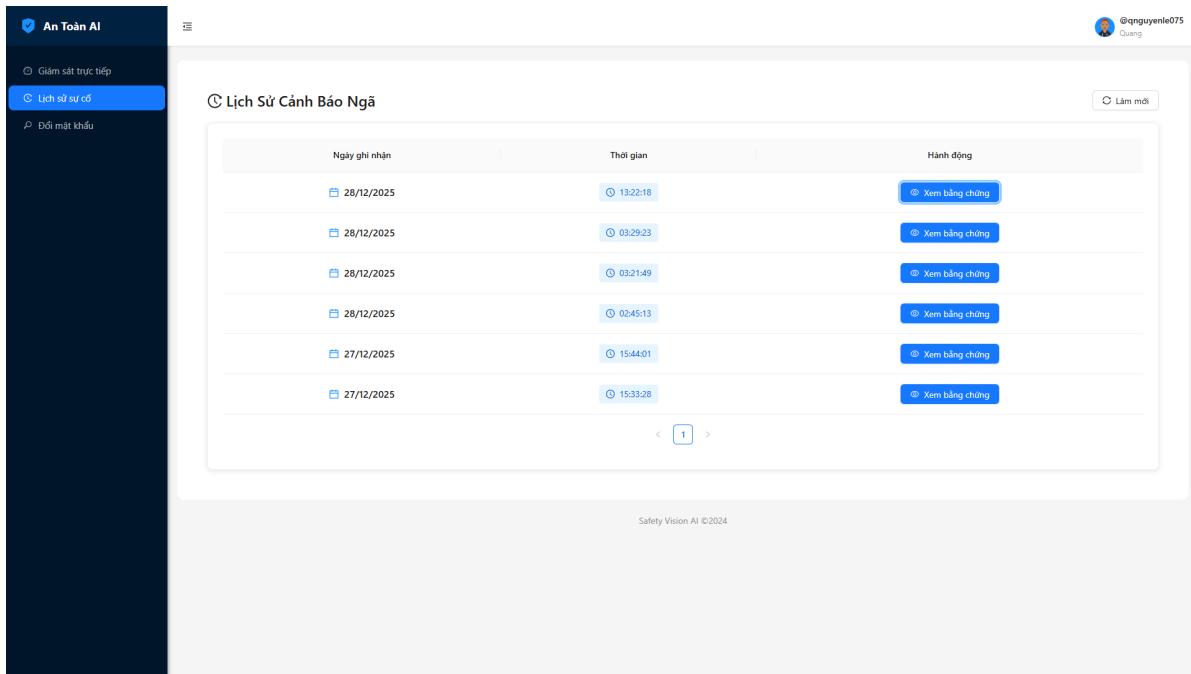


Figure 22: Màn hình Lịch sử Cảnh báo

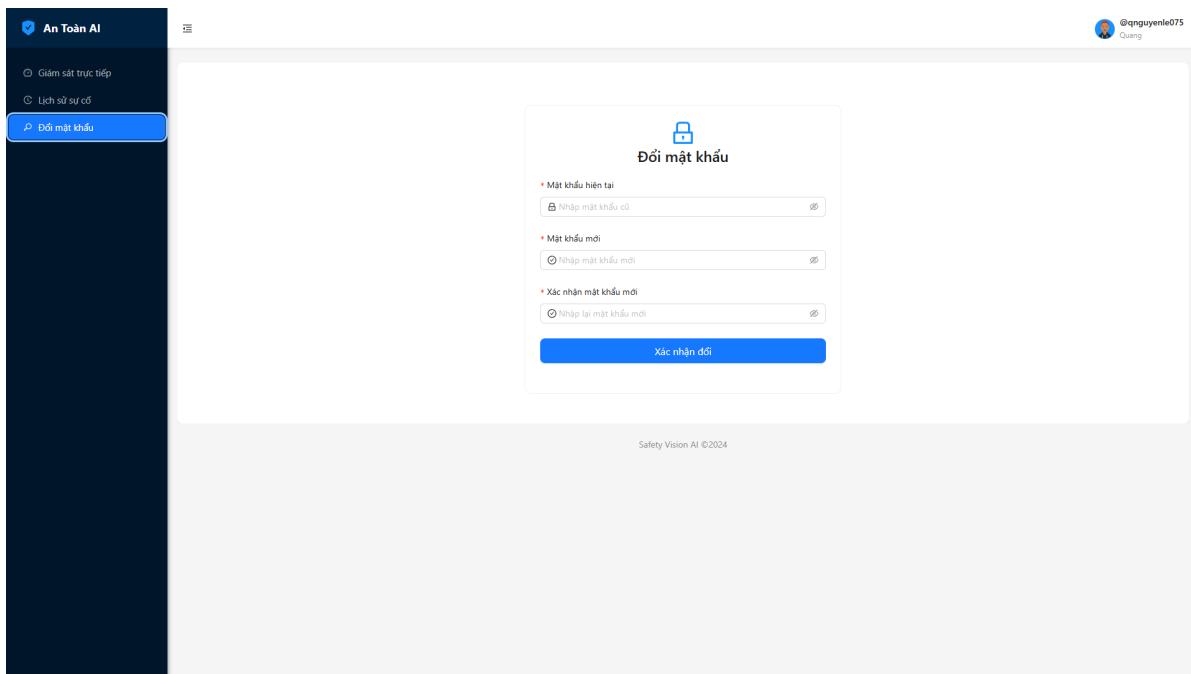


Figure 23: Màn hình Đổi Mật khẩu

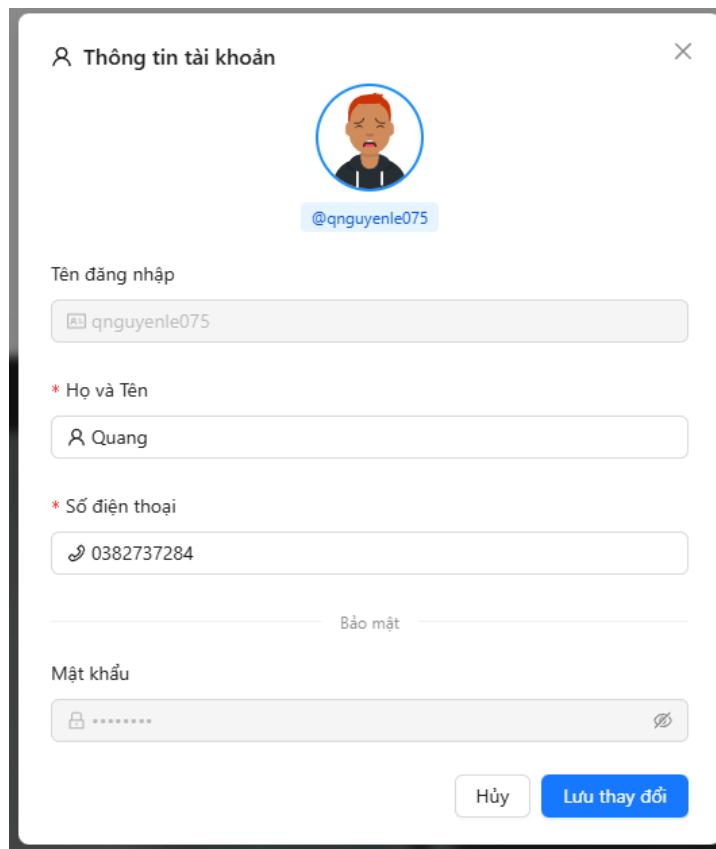


Figure 24: Màn hình Thông tin Tài khoản

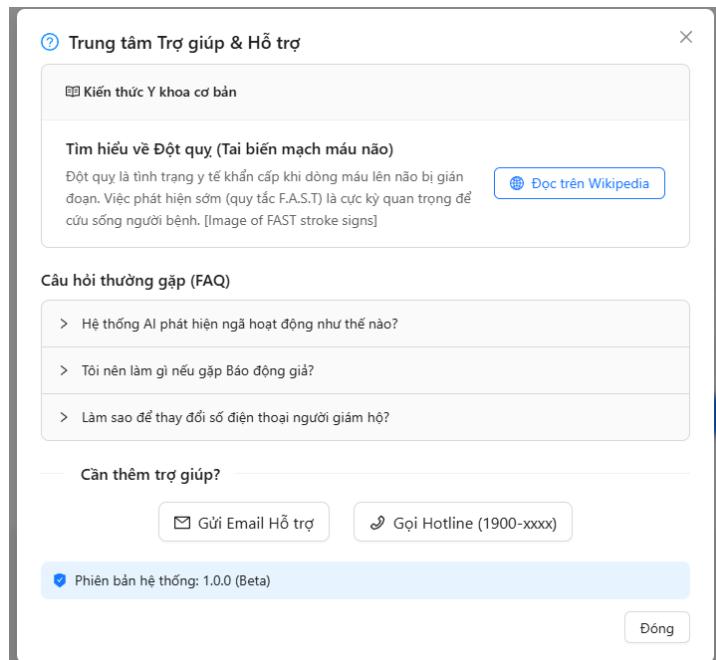


Figure 25: Màn hình Hỗ trợ

### 5.3.5 Thông báo Bot Telegram

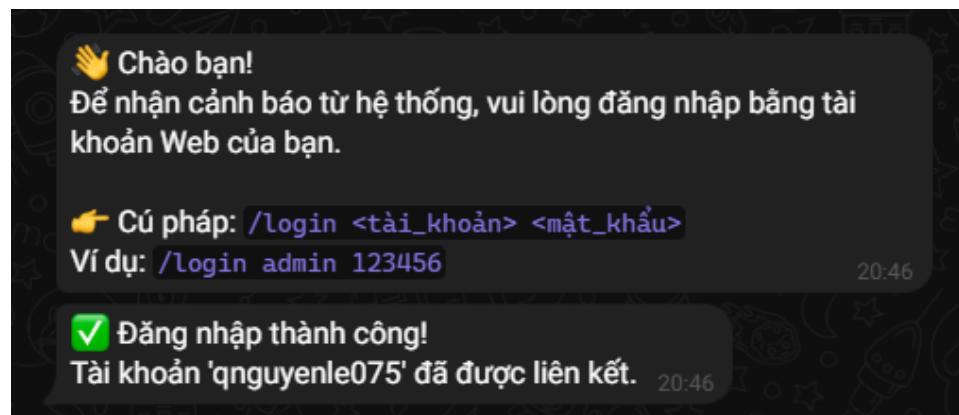


Figure 26: Thông báo Kết nối Bot Telegram

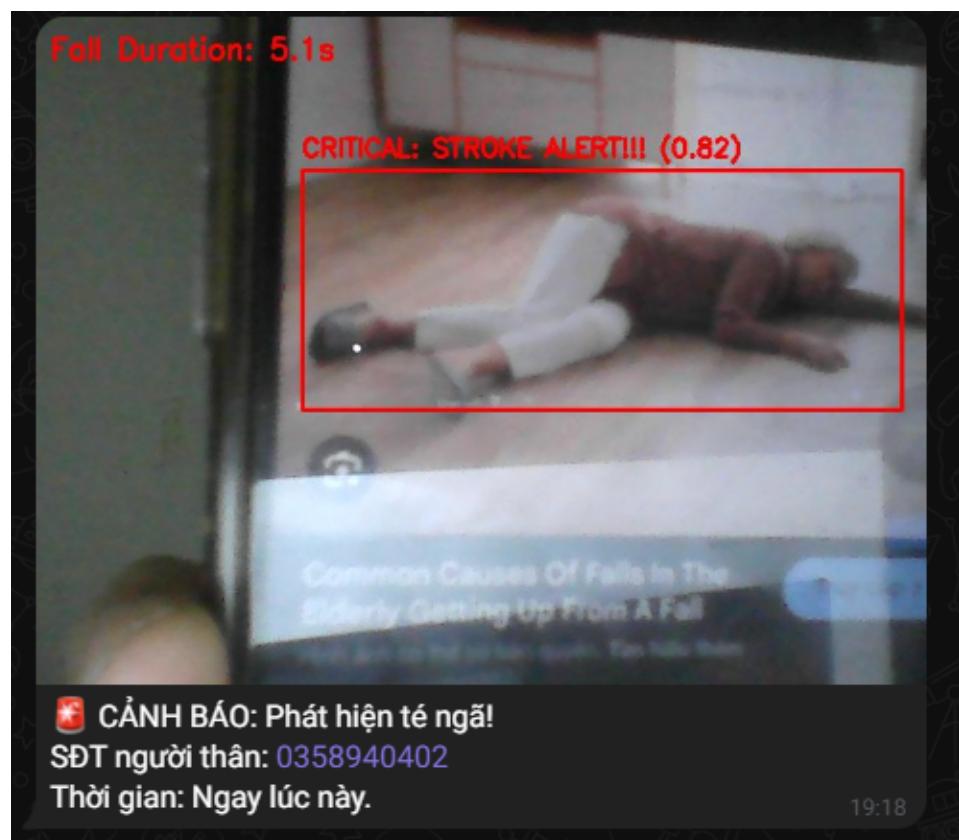


Figure 27: Thông báo Bot Telegram khi phát hiện ngã

## 5.4 Triển khai

Hiện tại, hệ thống được triển khai và vận hành trên môi trường **Local (Máy cục bộ)** để phục vụ mục đích thử nghiệm và báo cáo đồ án. Dưới đây là hướng dẫn chi tiết các bước để cài đặt và chạy ứng dụng.

### 5.4.1 Yêu cầu môi trường

Để chạy hệ thống, máy tính cần được cài đặt sẵn:

- **Python (3.9 trở lên):** Để chạy Backend.
- **Node.js (16.x trở lên) & NPM:** Để chạy Frontend.
- **Git:** Để quản lý mã nguồn.
- **Webcam:** Tích hợp sẵn hoặc rời (để input video).

### 5.4.2 Quy trình cài đặt và kích hoạt

- **Bước 1: Chuẩn bị mã nguồn**

Tải mã nguồn từ GitHub hoặc copy thư mục dự án vào máy. Cấu trúc app được tổ chức trong thư mục **fall-detection-system** bao gồm hai phần chính: **backend** và **frontend**.

- **Bước 2: Cài đặt và chạy Backend (Server)**

Mở Terminal (Command Prompt) và thực hiện các lệnh sau:

– Di chuyển vào thư mục backend:

```
cd fall-detection-system  
cd backend
```

– Tạo môi trường ảo (khuyến nghị) và kích hoạt:

```
python -m venv venv  
# Windows:  
.\\venv\\Scripts\\activate  
# MacOS/Linux:  
source venv/bin/activate
```

– Cài đặt các thư viện phụ thuộc:

```
pip install -r requirements.txt
```

– Khởi động Server:

```
uvicorn app.main:app --reload
```

**Kết quả thành công:** Server sẽ chạy tại địa chỉ `http://localhost:8000`.

- **Bước 3: Cài đặt và chạy Bot Telegram cảnh báo (Server)**

Mở một cửa sổ Terminal khác và thực hiện các lệnh sau:

- Di chuyển vào thư mục backend:

```
cd fall-detection-system  
cd backend
```

- Kích hoạt lại môi trường ảo ở bước trước (khuyến nghị):

```
# Windows:  
.\\venv\\Scripts\\activate  
# MacOS/Linux:  
source venv/bin/activate
```

- Khởi động Bot Telegram:

```
python bot_listener.py
```

**Kết quả thành công:** Bot báo cháy thành công trên Terminal.

- **Bước 4: Cài đặt và chạy Frontend (Client)**

Mở một cửa sổ Terminal mới và thực hiện:

- Di chuyển vào thư mục frontend:

```
cd fall-detection-system  
cd frontend
```

- Cài đặt các gói thư viện Node modules:

```
npm install
```

- Khởi động ứng dụng React:

```
npm run dev
```

**Kết quả thành công:** Trình duyệt sẽ tự động mở địa chỉ <http://localhost:5173>.

#### 5.4.3 Kết quả triển khai

Sau khi thực hiện các bước trên, hệ thống hoạt động ổn định trên môi trường **Localhost**.

- Người dùng truy cập vào <http://localhost:5173> để sử dụng.
- Camera được kích hoạt và truyền hình ảnh về Server xử lý với độ trễ thấp (< 500 ms).
- Cảnh báo được gửi tức thì qua giao thức **WebSocket**.

## 6 Kết luận

### 6.1 Tóm tắt Kết quả

Đồ án đã hoàn thành các mục tiêu cốt lõi đề ra ban đầu, xây dựng thành công quy trình **End-to-End** cho bài toán giám sát an toàn và phát hiện sự cố đột quy/té ngã. Cụ thể:

#### 6.1.1 Về Dữ liệu:

Nhóm đã xây dựng và chuẩn hóa thành công bộ dữ liệu gồm **473 mẫu ảnh chất lượng cao**, được gán nhãn chính xác với **3 lớp hành vi** (Fall Detected, Walking, Sitting), đảm bảo tính cân bằng và đa dạng về kích thước đối tượng.

#### 6.1.2 Về Mô hình:

- Đã thực nghiệm và so sánh **3 kiến trúc hiện đại** (YOLOv8n, YOLOv10n, RT-Detr).
- Lựa chọn thành công mô hình tối ưu là **YOLOv8n** với độ chính xác trung bình **mAP@50 đạt 85.4%**.
- Đặc biệt, mô hình đạt độ nhạy (**Recall**) rất cao đối với lớp sự cố “**Fall Detected**” là **93%**, giảm thiểu tối đa rủi ro bỏ sót các trường hợp nguy hiểm.
- Tốc độ suy luận đạt trung bình **4.6 ms/ảnh**, hoàn toàn đáp ứng yêu cầu xử lý thời gian thực.

#### 6.1.3 Về Ứng dụng:

Xây dựng hoàn thiện hệ thống giám sát tích hợp (**Web App + Backend AI**), cho phép xem video trực tiếp, lưu trữ lịch sử và tự động gửi cảnh báo tức thì qua **Telegram Bot** khi phát hiện người ngã.

## 6.2 Hạn chế

Mặc dù đã đạt được những kết quả khả quan, đồ án vẫn còn tồn tại một số hạn chế nhất định về cả mặt thuật toán và triển khai hệ thống:

#### 6.2.1 Hạn chế về Mô hình:

- Báo động giả (False Positives):** Hệ thống vẫn còn hiện tượng nhận diện nhầm các vật thể trong nền (background) hoặc các tư thế ngồi phức tạp thành hành động ngã.

- **Thiếu thông tin thời gian:** Do sử dụng mô hình phát hiện vật thể trên ảnh tĩnh (2D CNN), hệ thống gặp khó khăn trong việc phân biệt các hành động có tính chất động.
- **Môi trường phức tạp:** Độ chính xác giảm khi hoạt động trong các môi trường có cấu trúc hình học gây nhiễu như cầu thang hoặc góc khuất.

### **6.2.2 Hạn chế về Triển khai (Deployment):**

- **Môi trường mô phỏng:** Hệ thống chưa được triển khai trên các thiết bị phần cứng chuyên dụng, hiện chỉ dừng ở mức **Proof of Concept** sử dụng Webcam kết nối trực tiếp với máy tính Backend.
- **Dánh giá độ trễ mạng:** Hệ thống chưa được kiểm thử đầy đủ về độ trễ truyền dẫn và độ ổn định khi vận hành qua giao thức mạng trong điều kiện thực tế.

## **6.3 Hướng phát triển**

Để nâng cao chất lượng và tính ứng dụng của đồ án trong tương lai, nhóm đề xuất các hướng cải thiện sau:

### **6.3.1 Cải tiến thuật toán:**

- Nghiên cứu tích hợp các mạng nơ-ron hồi quy (như **LSTM**) hoặc sử dụng **Video Vision Transformers** để xử lý dữ liệu chuỗi thời gian.
- Bổ sung thêm lớp hành vi “**Lying**” (Nầm) và tăng cường dữ liệu nền (Negative Samples) nhằm giảm tỷ lệ báo động giả.

### **6.3.2 Tối ưu hóa triển khai:**

- Chuyển đổi mô hình sang định dạng **TensorRT** hoặc **ONNX** để tối ưu tốc độ suy luận trên các thiết bị biên.
- Thủ nghiệm kết nối với hệ thống **Camera IP** thực tế để đánh giá và xử lý các vấn đề về băng thông và độ trễ mạng.

## **7 Tài liệu tham khảo**

- 1 W. Lv et al., "DETRs Beat YOLOs on Real-time Object Detection," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2024, pp. 16965–16974.
- 2 Y. Luo, N. Pan, Y. Zhang, Y. Zhang, and X. Wu, "YOLO-Drone: An Optimized YOLOv8 Network for Tiny UAV Object Detection," Sensors, vol. 24, no. 15, p. 4858, Jul. 2024. doi: 10.3390/s24154858.

- 3 H. Cheng and F. Kang, "Corrosion Detection and Grading Method for Hydraulic Metal Structures Based on an Improved YOLOv10 Sequential Architecture," Applied Sciences, vol. 14, no.[1] 24, p. 12009, Dec. 2024.[1] doi: 10.3390/app142412009.
- 4 G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLO," Jan. 2023. [Online].
- 5 N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-End Object Detection with Transformers," in European Conference on Computer Vision (ECCV), 2020, pp. 213–229.
- 6 N. Lu, Y. Wu, L. Feng, and J. Song, "Deep Learning for Fall Detection: Three-Dimensional CNN Combined With LSTM on Video Kinematic Data," IEEE Journal of Biomedical and Health Informatics, vol. 23, no. 1, pp. 314–323, Jan. 2019.
- 7 "Artificial intelligence: a modern approach", 2020, Russell, S. and Norvig, P.
- 8 "Deep learning", 2016, Goodfellow, I., Bengio, Y., and Courville, A