

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN



THỰC TẬP CƠ SỞ
ĐỀ TÀI: GIÁM SÁT AN NINH, NHẬN DẠNG ĐỐI
TƯỢNG XÂM NHẬP BẰNG OpenCV, YOLOv8

Họ và tên: HOÀNG ĐỨC HƯNG

Mã sinh viên: B22DCCN408

Giáo viên: Nguyễn Mạnh Hùng

Hà Nội - 2025

MỤC LỤC

Chương 1: GIỚI THIỆU CHUNG	5
1. 1. Giới thiệu đề tài	5
1. 1. 1. Bối cảnh	5
1. 1. 2. Các giải pháp hiện có	5
1. 1. 3. Hệ thống giám sát an ninh và nhận dạng đối tượng	6
Chương 2: CƠ SỞ LÝ THUYẾT VÀ CÁC THÀNH PHẦN HỆ THỐNG	7
2. 1. Các Thành phần của Hệ thống Giám sát An ninh	7
2. 1. 1. Giới thiệu về YOLOv8	7
2. 1. 2. OPENCV	8
2. 1. 3. Numpy	9
2. 1. 4. Caer Library	9
2. 1. 5. Pytorch	10
2. 1. 6. Flask	11
2. 2. Các Chỉ số Đánh giá Hiệu năng	12
2. 2. 1. Ma trận nhầm lẫn(Confusion Matrix)	12
2. 2. 2. Accuracy, Precision, Recall, F1 Score	13
2. 2. 3. Mean Average Precision (mAP)	15
Chương 3: Hệ thống giám sát an ninh, nhận dạng đối tượng xâm nhập	17
3. 1. Đặt lại bài toán	17

3. 2. Mô tả bộ dữ liệu sử dụng.....	17
3. 2. 1. Xây dựng công cụ trích xuất frame.....	18
3. 2. 2. Công cụ RoboFlow trong việc gán nhãn Dataset.....	18
3. 3. Train Model.....	20
3. 4. Đánh giá mô hình.....	21
3. 5. Xây dựng hệ thống giám sát và phát hiện đối tượng.....	23
3. 5. 1. Các thành phần chính:.....	23
3. 5. 2. Thiết kế hệ thống kiến trúc phần mềm.....	23
3. 5. 3. Thiết kế cơ sở dữ liệu.....	24
3. 5. 4. Mô tả nghiệp vụ.....	25
3. 5. 5. Triển khai Hệ thống.....	32
3. 5. 6. Đề xuất hướng cải tiến Hệ thống.....	34

MỞ ĐẦU

Việc áp dụng trí tuệ nhân tạo trong hệ thống giám sát an ninh ngày càng trở nên cần thiết trong bối cảnh công nghệ hiện đại phát triển nhanh chóng. Các hệ thống giám sát truyền thống như CCTV mặc dù có vai trò quan trọng trong việc theo dõi nhưng lại gặp hạn chế khi không thể phản ứng kịp thời đối với các tình huống xâm nhập. Điều này dẫn đến nhu cầu cấp bách trong việc tích hợp các công nghệ nhận dạng đối tượng tự động, giúp cải thiện khả năng phát hiện và cảnh báo, từ đó nâng cao hiệu quả bảo vệ tài sản và người sử dụng.

Phương hướng giải quyết được đề xuất trong luận văn không chỉ dựa trên việc triển khai mô hình YOLOv8 – một trong những công nghệ tiên tiến nhất trong lĩnh vực nhận dạng đối tượng thời gian thực – mà còn chú trọng đến việc tích hợp một hệ thống phần mềm toàn diện. Qua việc nghiên cứu và so sánh với các dự án tương tự, luận văn hướng tới một giải pháp tổng hợp, trong đó việc xử lý hình ảnh qua OpenCV và Numpy được kết hợp với khả năng học sâu của PyTorch để huấn luyện mô hình. Hơn thế nữa, hệ thống còn tích hợp các công cụ hỗ trợ như Flask để xây dựng giao diện quản lý và Twilio nhằm đảm bảo quá trình cảnh báo tự động khi phát hiện đối tượng xâm nhập. Cách tiếp cận này không chỉ giúp nâng cao hiệu suất nhận dạng trong điều kiện thực tế mà còn mở ra hướng phát triển tích hợp các thành phần mới như theo dõi đối tượng (DeepSort) trong tương lai, đảm bảo tính mở rộng và linh hoạt của hệ thống.

Luận văn được chia thành ba chương chính. Chương 1 sẽ trình bày các nội dung giới thiệu chung, vấn đề và các giải pháp hiện có trên thế giới, từ đó đưa ra đề xuất giải pháp dựa trên các nền tảng công nghệ hiện đại. Chương 2 trình bày chi tiết các thành phần công nghệ sẽ được ứng dụng, bao gồm lịch sử ra đời, nguyên lý hoạt động, ứng dụng cụ thể và đánh giá ưu nhược điểm của từng công cụ; qua đó, phần này sẽ giúp người đọc nắm được cơ sở lý thuyết và thực tiễn của các thành phần được lựa chọn. Chương 3 sẽ đặt lại bài toán một cách cụ thể hơn dựa trên những thành phần đã trình bày, sau đó tiến hành thiết kế hệ thống (có thể thông qua sơ đồ UML) cùng với quy trình chuẩn bị dữ liệu kiểm thử, kết quả cài đặt và đánh giá thực nghiệm, đưa ra hướng cải tiến.

Chương 1: GIỚI THIỆU CHUNG

1. 1. Giới thiệu đề tài

1. 1. 1. Bối cảnh

Trong môi trường ngân hàng, nơi an ninh và bảo mật luôn được đặt lên hàng đầu, các hệ thống giám sát truyền thống dựa vào camera CCTV không còn đáp ứng đủ yêu cầu. Mặc dù có khả năng ghi nhận hình ảnh liên tục, những hệ thống này lại thiếu cơ chế xử lý và phân tích dữ liệu tự động, dẫn đến việc phát hiện hành vi xâm nhập hoặc các tình huống bất thường xảy ra chậm trễ. Trong bối cảnh này, bài toán cần giải quyết là xây dựng một hệ thống giám sát thông minh dành riêng cho ngân hàng, có khả năng tự động phân tích video, nhận diện các đối tượng có nguy cơ xâm nhập một cách chính xác và nhanh chóng, từ đó đưa ra các phản ứng kịp thời để bảo vệ tài sản và đảm bảo an ninh cho toàn bộ cơ sở ngân hàng.

1. 1. 2. Các giải pháp hiện có

Trên thế giới, đã có nhiều dự án giám sát an ninh ứng dụng công nghệ AI được triển khai với những thành tựu đáng kể. Ví dụ, hệ thống Spot AI được áp dụng tại các trung tâm thương mại lớn đã tự động phân tích video để nhận diện các hành vi bất thường và gửi cảnh báo ngay lập tức, góp phần cải thiện đáng kể hiệu quả giám sát. Tương tự, IntelliVision phát triển các giải pháp nhận diện khuôn mặt và theo dõi chuyển động liên tục, giúp quản lý an ninh tại các khu vực công cộng. Các sản phẩm của Hikvision cũng tích hợp thuật toán nhận dạng đối tượng để phục vụ cho cả môi trường thương mại và công nghiệp, mặc dù những hệ thống này thường tập trung vào quy mô lớn và mục tiêu chung. Trong bối cảnh này, dự án này hướng đến việc phát triển một hệ thống giám sát an ninh chuyên dụng cho môi trường ngân hàng. Hệ thống được đề xuất sẽ tối ưu hóa quy trình xử lý hình ảnh và nhận diện đối tượng thông qua việc kết hợp YOLOv8, OpenCV, Numpy, và PyTorch, cùng với việc tích hợp giao diện quản lý và cơ chế cảnh báo tự động. Mục tiêu không phải là để đề cao bản thân mà là học hỏi, cải tiến dựa trên những kinh nghiệm quý báu từ các dự án đã có, và từ đó góp phần tạo ra một giải pháp thiết thực, phù hợp với điều kiện và nhu cầu an ninh đặc thù của các cơ sở ngân hàng.

1. 1. 3. Hệ thống giám sát an ninh và nhận dạng đối tượng

Hệ thống sẽ được xây dựng với mục tiêu tạo ra một giải pháp giám sát an ninh tự động, chuyên dụng cho môi trường ngân hàng. Đầu tiên, mô hình YOLOv8 sẽ được áp dụng để nhận diện đối tượng trong các luồng video thời gian thực, nhờ vào khả năng xử lý nhanh và độ chính xác cao của nó. Trước khi dữ liệu video được đưa vào mô hình, các khung hình sẽ trải qua một quy trình tiền xử lý kỹ lưỡng sử dụng OpenCV và Numpy, qua đó loại bỏ nhiễu, cân bằng ánh sáng và chuẩn hóa dữ liệu, giúp nâng cao chất lượng đầu vào và đảm bảo rằng mô hình học sâu sẽ nhận diện chính xác hơn các đối tượng trong môi trường có điều kiện ánh sáng và độ nhiễu phức tạp.

Quá trình huấn luyện và triển khai mô hình sẽ được thực hiện bằng PyTorch kết hợp với CUDA, tận dụng sức mạnh xử lý của GPU để giảm thời gian huấn luyện và suy luận, đảm bảo hệ thống có thể phản hồi ngay lập tức khi phát hiện sự cố. Mặt khác, một giao diện quản lý trực quan sẽ được xây dựng trên nền tảng Flask, cho phép người quản trị theo dõi liên tục các luồng video, ngay khi phát hiện đối tượng xâm nhập, hệ thống sẽ gửi cảnh báo tới người dùng.

Ngoài ra, để mở rộng khả năng phân tích hành vi đối tượng, trong giai đoạn phát triển sau này, dự án còn có tiềm năng tích hợp module theo dõi đối tượng như DeepSort. Module này sẽ cung cấp thông tin chi tiết về hành vi và chuyển động của các đối tượng được phát hiện, từ đó hỗ trợ quá trình phân tích và đánh giá các tình huống nguy cơ, góp phần tăng cường khả năng dự báo và phản ứng của hệ thống trong môi trường ngân hàng. Tổng hợp lại, hệ thống không chỉ tập trung vào việc nhận diện đối tượng một cách chính xác và nhanh chóng mà còn được thiết kế để tích hợp một cách liền mạch với giao diện quản lý và hệ thống cảnh báo, nhằm đảm bảo an ninh tối đa cho các cơ sở ngân hàng.

Chương 2: CƠ SỞ LÝ THUYẾT VÀ CÁC THÀNH PHẦN HỆ THỐNG

2. 1. Các Thành phần của Hệ thống Giám sát An ninh

2. 1. 1. Giới thiệu về YOLOv8

a. Lịch sử ra đời

YOLO (You Only Look Once) ban đầu được giới thiệu vào năm 2016 bởi Joseph Redmon và cộng sự với mục tiêu thực hiện phát hiện đối tượng chỉ qua một lần duyệt qua ảnh. Sau đó, các phiên bản tiếp theo (YOLOv2, YOLOv3, YOLOv4, YOLOv5) đã cải tiến về độ chính xác và tốc độ. YOLOv8, được phát triển bởi Ultralytics và ra mắt vào năm 2023, là bước tiến mới nhất trong dòng mô hình này. Nó tích hợp những cải tiến về kiến trúc và tối ưu hóa, cho phép xử lý các bài toán nhận diện trong thời gian thực với hiệu suất cao.

Nguồn tham khảo: <https://www.ultralytics.com/>

b. Mô tả hoạt động

YOLOv8 hoạt động theo nguyên tắc "một lần duyệt" (single-shot detection) trong đó toàn bộ quá trình nhận diện đối tượng – bao gồm trích xuất đặc trưng, định vị và phân loại – được thực hiện qua một mạng neural duy nhất. Cấu trúc của YOLOv8 thường bao gồm:

- ✧ Backbone: Trích xuất các đặc trưng cơ bản từ ảnh đầu vào bằng các lớp convolution tiên tiến, tối ưu hóa thông qua các kỹ thuật như residual connections và CSP (Cross Stage Partial connections).
- ✧ Head: Giai đoạn dự đoán, nơi các khối dự đoán xác định vị trí (bounding boxes) và lớp của đối tượng. YOLOv8 có thể sử dụng các kỹ thuật anchor-free detection để giảm thiểu phức tạp và tăng tốc độ suy luận.
- ✧ Hơn nữa, YOLOv8 tích hợp các kỹ thuật hiện đại như data augmentation nâng cao và post-processing tối ưu (ví dụ: non-max suppression) để đảm bảo kết quả nhận diện ổn định ngay cả trong điều kiện ánh sáng yếu hoặc môi trường phức tạp.

Nguồn tham khảo: <https://docs.ultralytics.com/models/yolov8/#key-features-of-yolov8>

c. Ứng dụng

Trong dự án này, YOLOv8 đóng vai trò trong việc:

- ✧ Phát hiện đối tượng thời gian thực: Mỗi khung hình video từ camera sẽ được xử lý qua YOLOv8 để phát hiện các đối tượng xâm nhập (như người, xe hoặc vật thể bất thường).

- ✧ Định vị và phân loại chính xác: Sau khi phát hiện, YOLOv8 sẽ xác định chính xác vị trí của đối tượng trong khung hình, cho phép kích hoạt các hành động tiếp theo như lưu lại hình ảnh, gửi cảnh báo hoặc kích hoạt module theo dõi (DeepSort).
- ✧ Hỗ trợ các module phụ trợ: Kết quả của YOLOv8 được truyền đến các hệ thống giao diện (qua Flask) để hiển thị trực tiếp, cũng như đến hệ thống cảnh báo để thông báo kịp thời cho đội ngũ an ninh.
- ✧ Điều này đảm bảo rằng hệ thống không chỉ nhận diện đối tượng một cách nhanh chóng mà còn có thể xử lý và phản hồi tức thời với các tình huống bất thường.

2. 1. 2. OPENCV

a. Lịch sử ra đời

OpenCV được phát triển bởi Intel vào năm 1999 với mục tiêu tạo ra một thư viện mã nguồn mở phục vụ cho xử lý ảnh và thị giác máy tính. Vào năm 2000, OpenCV được phát hành công khai và nhanh chóng trở thành tiêu chuẩn trong lĩnh vực này nhờ vào khả năng xử lý hiệu quả và hỗ trợ đa nền tảng.

Nguồn tham khảo: opencv.org/about

b. Mô tả hoạt động

OpenCV cung cấp hàng trăm hàm xử lý ảnh và video, bao gồm các phép biến đổi không gian màu, lọc nhiễu, phát hiện biên, và trích xuất đặc trưng. Thư viện này được viết chủ yếu bằng C/C++ và tối ưu hóa cho hiệu năng cao, hỗ trợ tích hợp với GPU (ví dụ: CUDA) để xử lý video thời gian thực. Quy trình xử lý thường bắt đầu bằng việc thu thập dữ liệu từ camera, sau đó thực hiện các bước tiền xử lý như chuyển đổi màu, cân bằng sáng và loại bỏ nhiễu trước khi chuyển sang các bước nhận diện đối tượng.

c. Ứng dụng

Trong dự án này, OpenCV sẽ thực hiện các bước tiền xử lý hình ảnh:

- ✧ Thu thập và chuẩn hóa dữ liệu: Lấy khung hình từ camera, chuyển đổi sang định dạng phù hợp (ví dụ, chuyển sang grayscale để giảm bớt dữ liệu không cần thiết).
- ✧ Cải thiện chất lượng ảnh: Áp dụng các thuật toán lọc nhiễu và cân bằng ánh sáng để đảm bảo hình ảnh đầu vào đạt chất lượng tối ưu, từ đó tăng độ chính xác khi phát hiện đối tượng.
- ✧ Phát hiện chuyển động: Sử dụng các kỹ thuật trích xuất biên và phân tích chuyển động để xác định các vùng có khả năng xuất hiện đối tượng xâm nhập, làm cơ sở cho bước nhận diện tiếp theo.
- ✧ Các bước này không chỉ giúp nâng cao chất lượng đầu vào cho mô hình YOLOv8 mà còn giảm thiểu lỗi do ảnh không rõ nét hoặc bị nhiễu, từ đó tối ưu hóa quá trình nhận diện đối tượng.

d. Đánh giá

So với MATLAB Image Processing Toolbox – một công cụ thương mại với khả năng xử lý mạnh mẽ nhưng đòi hỏi chi phí cao và môi trường vận hành chuyên biệt – OpenCV mang lại lợi thế về tính mã nguồn mở, khả năng tùy chỉnh linh hoạt và dễ dàng tích hợp vào các hệ thống phần mềm khác, đặc biệt là các dự án sử dụng Python cho xử lý dữ liệu.

2. 1. 3. Numpy

a. Lịch sử ra đời

Numpy được phát triển dựa trên các dự án Numeric và Numarray nhằm cung cấp khả năng xử lý mảng đa chiều và các phép tính số học hiệu quả cho Python. Nó đã trở thành nền tảng cho hầu hết các thư viện khoa học dữ liệu và học máy hiện nay.

Nguồn tham khảo: numpy.org

b. Mô tả hoạt động

Numpy cho phép tạo và thao tác với các mảng đa chiều (ndarray) thông qua các phép toán vector hóa, từ đó tối ưu hóa quá trình tính toán số học. Các hàm toán học và thống kê của Numpy được viết bằng C, giúp giảm thiểu vòng lặp Python không cần thiết và tăng tốc độ xử lý dữ liệu. Điều này đặc biệt hữu ích khi xử lý các khung hình ảnh có kích thước lớn, nơi việc tính toán từng pixel một cách thủ công sẽ gây ra sự chậm trễ.

c. Ứng dụng

Trong dự án này, Numpy sẽ được sử dụng vào mục đích:

- ✧ Chuyển đổi dữ liệu ảnh: Biến đổi các khung hình từ định dạng hình ảnh sang mảng số, làm nền tảng cho việc xử lý tiếp theo bằng OpenCV và các mô hình học sâu.
- ✧ Tiền xử lý số liệu: Thực hiện các phép tính thống kê như tính trung bình, chuẩn hóa độ sáng và đối chiếu các đặc trưng của ảnh trước khi đưa vào mô hình nhận diện.
- ✧ Hỗ trợ xử lý batch: Khi làm việc với video, Numpy giúp xử lý tập hợp các khung hình dưới dạng batch, tối ưu hóa hiệu suất khi truyền dữ liệu vào mạng neural.

2. 1. 4. Caer Library

a. Lịch sử ra đời

Caer được phát triển nhằm đơn giản hóa các thao tác xử lý ảnh khi sử dụng OpenCV và Numpy. Mục tiêu chính của nó là giảm thiểu lượng code cần thiết và giúp lập trình viên dễ dàng triển khai các thao tác xử lý ảnh cơ bản.

b. Mô tả hoạt động

Caer cung cấp các API bao bọc (wrapper) cho các hàm của OpenCV và Numpy, cho phép thực hiện các thao tác như chuyển đổi không gian màu, thay đổi kích thước, lọc nhiễu, và tăng cường dữ liệu (data augmentation) chỉ với vài dòng lệnh. Các hàm này được thiết kế để dễ đọc, dễ bảo trì và tích hợp liền mạch vào quy trình xử lý dữ liệu của dự án.

c. Ứng dụng

Trong dự án này, Caer sẽ đảm nhận vai trò:

- ✧ Tiền xử lý tự động: Thực hiện các bước chuẩn hóa ảnh, chuyển đổi không gian màu và lọc nhiễu trước khi gửi ảnh cho mô hình YOLOv8.
- ✧ Tăng cường dữ liệu: Tạo ra các biến thể của ảnh (ví dụ: xoay, phóng to, giảm độ sáng) nhằm cải thiện khả năng tổng quát của mô hình khi huấn luyện.
- ✧ Tích hợp nhanh: Giúp các nhà phát triển nhanh chóng triển khai và kiểm thử các giải pháp xử lý ảnh mà không cần phải viết lại các hàm từ đầu, tiết kiệm thời gian và giảm thiểu lỗi phát sinh.

d. Đánh giá

So với việc tự viết các hàm xử lý ảnh dựa trên OpenCV, Caer giúp giảm đáng kể lượng code cần viết và rút ngắn thời gian phát triển. Mặc dù khả năng tùy biến của Caer có thể hạn chế hơn so với việc sử dụng trực tiếp OpenCV, nhưng với các ứng dụng đòi hỏi triển khai nhanh và ổn định, Caer là lựa chọn lý tưởng.

2. 1. 5. Pytorch

a. Lịch sử ra đời

PyTorch được phát triển bởi Facebook AI Research (FAIR) và ra mắt vào năm 2016. CUDA của NVIDIA, ra đời từ đầu những năm 2000, cung cấp nền tảng xử lý song song cho các GPU, giúp tăng tốc các phép tính toán trong học sâu.

Nguồn tham khảo: pytorch.org, developer.nvidia.com/cuda-zone

b. Mô tả hoạt động

PyTorch cho phép xây dựng các mô hình học sâu thông qua việc tạo ra đồ thị tính toán động (dynamic computation graph), hỗ trợ tính toán tự động đạo hàm (automatic differentiation). Khi tích hợp với CUDA, PyTorch có thể chuyển các phép tính nặng sang GPU, thực hiện xử lý song song và giảm thời gian huấn luyện đáng kể. Quy trình hoạt động bao gồm việc truyền dữ liệu qua các lớp neural, tính toán loss, và cập nhật trọng số thông qua các thuật toán tối ưu hóa như Adam hoặc SGD.

c. Ứng dụng

Trong dự án này, PyTorch sẽ đảm nhận vai trò:

- ✧ Huấn luyện mô hình YOLOv8: Sử dụng dữ liệu ảnh thu được từ hệ thống giám sát để huấn luyện mô hình nhận diện đối tượng.
- ✧ Tối ưu hóa hiệu suất: Tận dụng CUDA để xử lý các phép tính phức tạp, giúp giảm thời gian phản hồi và cho phép hệ thống nhận diện đối tượng trong thời gian thực.
- ✧ Chuyển giao mô hình: Sau khi huấn luyện, mô hình sẽ được tích hợp vào hệ thống để thực hiện suy luận (inference) liên tục trên các luồng video từ camera.

d. Đánh giá

So với TensorFlow, PyTorch được đánh giá cao về tính linh hoạt nhờ vào đồ thị tính toán động, thuận tiện cho việc thử nghiệm và tinh chỉnh mô hình. Trong khi TensorFlow mạnh về triển khai quy mô lớn, PyTorch lại phù hợp hơn với quá trình nghiên cứu và phát triển, đặc biệt trong môi trường học thuật và thử nghiệm.

2. 1. 6. Flask

a. Lịch sử ra đời

Flask được tạo ra bởi Armin Ronacher và ra mắt vào năm 2010 như một microframework cho Python, nhằm cung cấp giải pháp xây dựng ứng dụng web đơn giản, nhẹ nhàng và dễ mở rộng.

Nguồn tham khảo: flask.palletsprojects.com

b. Mô tả hoạt động

Flask hoạt động theo mô hình WSGI, cung cấp các tính năng cơ bản như định tuyến URL, xử lý request/response, tích hợp với template engine (Jinja2) và các module mở rộng để xây dựng các ứng dụng web linh hoạt. Các thành phần của Flask giúp tạo nên một backend mạnh mẽ cho việc quản lý giao diện người dùng và xử lý dữ liệu theo thời gian thực.

c. Ứng dụng

Trong dự án này, Flask sẽ đảm nhận vai trò:

- ✧ Xây dựng giao diện dashboard: Phát triển trang web cho phép người quản trị theo dõi luồng video, xem cảnh báo và truy xuất dữ liệu lưu trữ từ hệ thống.
- ✧ Tạo API giao tiếp: Cung cấp các API để giao tiếp giữa hệ thống backend (xử lý hình ảnh, nhận diện đối tượng) và giao diện người dùng, đảm bảo việc truyền tải dữ liệu diễn ra nhanh chóng và chính xác.
- ✧ Tích hợp với các hệ thống cảnh báo: Kết nối với các module khác như Twilio để kích hoạt các cảnh báo qua SMS hay email ngay khi phát hiện sự cố.

d. Đánh giá

So với Django – một framework web toàn diện – Flask mang lại sự đơn giản và dễ triển khai hơn cho các dự án nhỏ đến vừa. Trong khi Django cung cấp nhiều tính năng tích hợp sẵn nhưng cũng làm tăng độ phức tạp, Flask cho phép tập trung vào các chức năng cần thiết cho giao diện giám sát, thuận tiện cho việc mở rộng hoặc tùy chỉnh theo yêu cầu.

2. 2. Các Chỉ số Đánh giá Hiệu năng.

2. 2. 1. Ma trận nhầm lẫn(Confusion Matrix)

		Actual Values		
		Positive (1)	Negative (0)	
Predicted Values	Positive (1)	TP Type 1 Error	FP Type 1 Error	
	Negative (0)	FN Type 2 Error	TN	

TP: True positive (dương tính thật)
FP: False positive (dương tính giả)
TN: True negative (âm tính thật)
FN: False negative (âm tính giả)

Image 1: Confusion Matrix, nguồn: <https://www.youtube.com/watch?v=AfBfJ2imgOE&t=1s>

a. Ý nghĩa các phần tử trong ma trận

- ✧ TP: Số lượng mẫu thực tế là Positive và được mô hình dự đoán đúng là Positive.
- ✧ TN: Số lượng mẫu thực tế là Negative và được mô hình dự đoán đúng là Negative.
- ✧ FP: Số lượng mẫu thực tế là Negative nhưng bị mô hình dự đoán nhầm thành Positive (đây là lỗi loại 1).
- ✧ FN: Số lượng mẫu thực tế là Positive nhưng bị mô hình dự đoán nhầm thành Negative (đây là lỗi loại 2).

b. Ví dụ về Ma trận Phát hiện gian lận giao dịch.

- ✧ Positive: Các giao dịch hợp lệ, Negative: Các giao dịch không hợp lệ.
- ✧ TP = 99, FP = 1, FN = 0, TN = 0.
- ✧ Actual values: 99 trường hợp giao dịch hợp lệ, 1 trường hợp giao dịch gian lận.
- ✧ Predicted values: 100 trường hợp hợp lệ.

		Actual values	
		positive	negative
Predicted values	positive	99	1
	negative	0	0

Image 2: Ví dụ Confusion Matrix, nguồn: <https://www.youtube.com/watch?v=AfBfJ2imgOE&t=1s>

Nhìn qua thì thấy Độ chính xác là 99%, độ chính xác rất cao. Nhưng mà chỉ có 1 trường hợp gian lận mà không phát hiện được, cho nên mô hình này hoàn toàn không dùng được. Dữ liệu ở trong trường hợp này gọi là Dữ liệu không cân bằng.

2. 2. 2. Accuracy, Precision, Recall, F1 Score

a. Accuracy

Accuracy là tỷ lệ phần trăm của các dự đoán đúng so với tổng số dự đoán:

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

b. Precision

✧ Đo tỷ lệ dự đoán đúng trong số các dự đoán dương tính:

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

Ứng dụng trong giám sát:

✧ Precision cao: Giảm cảnh báo sai (false alarm) – ví dụ: camera không nhầm bóng cây thành người đột nhập.

c. Recall

✧ Đo tỷ lệ đối tượng thật được phát hiện chính xác:

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

Ứng dụng trong giám sát:

- ✧ Recall cao: Đảm bảo không bỏ sót sự kiện quan trọng (ví dụ: phát hiện kẻ xâm nhập ẩn nấp).
- ✧ Cân bằng: Trong giám sát y tế (phát hiện bệnh nhân ngã), cần Recall cao để không bỏ sót sự cố, nhưng cũng phải duy trì Precision hợp lý để tránh quá tải cảnh báo.

d. F1-Score

Là trung bình điều hòa của Precision và Recall, cân bằng giữa hai chỉ số:

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

Trong đó:

- ✧ TP (True Positives): Số đối tượng được phát hiện đúng.
- ✧ TN (True Negatives): Số trường hợp âm tính được dự đoán đúng.
- ✧ FP (False Positives): Số trường hợp báo động giả.
- ✧ FN (False Negatives): Số trường hợp đối tượng bị bỏ sót.

Ứng dụng trong giám sát:

- ✧ Tối ưu hệ thống giám sát đa nhiệm (ví dụ: camera vừa đếm người, vừa phát hiện hành vi đáng ngờ).
- ✧ Phù hợp khi dữ liệu mất cân bằng lớp (ví dụ: số sự kiện đột nhập ít hơn hoạt động bình thường).

Precision Recall Curve

Là đồ thị biểu diễn mối quan hệ giữa precision(Oy) và recall(Ox), được sử dụng để đánh giá hiệu suất của một mô hình nhận dạng đối tượng.

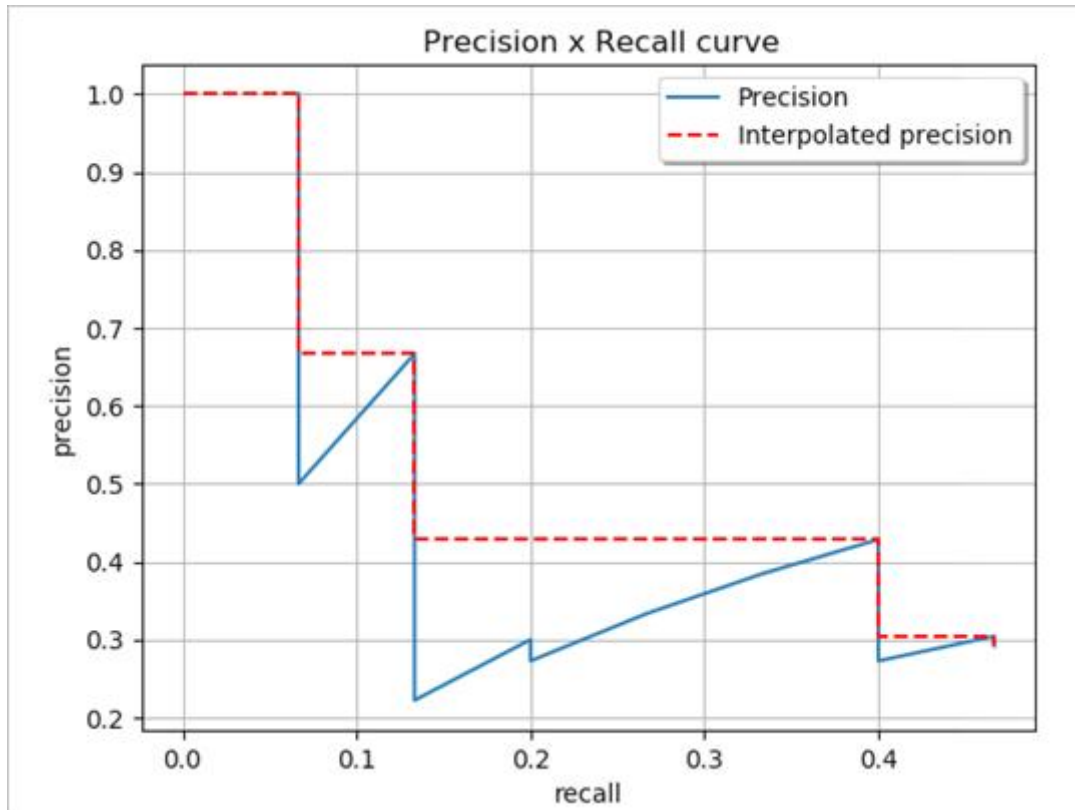


Image 3: P-R curve, nguồn: chính chủ

Một mô hình được coi là tốt nếu giá trị precision vẫn giữ ở mức cao khi recall tăng, nghĩa là khi mô hình phát hiện nhiều đối tượng(recall tăng) nó vẫn duy trì được độ chính xác cao(precision không giảm nhiều). Điều này cho thấy mô hình có khả năng nhận diện đúng đối tượng một cách ổn định, ít báo động giả.

2. 2. 3. Mean Average Precision (mAP)

a. Average Precision(AP)

AP được tính bằng diện tích dưới đường PR Curve cho một lớp đối tượng (xem hình trên):

$$AP = \int_0^1 \rho(R) dR$$

Ứng dụng trong giám sát:

✧ Khi AP nhỏ thì cả Precision và Recall đều khá thấp, suy ra model không tốt.

b. Mean Average Precision(mAP)

Là trung bình các AP của tất cả các lớp đối tượng:

$$mAP = \frac{1}{N} \times \sum_1^N AP_i$$

Trong đó:

N: Số lớp đối tượng

Ứng dụng trong giám sát:

mAP là chỉ số toàn diện, đánh giá đồng thời khả năng định vị và phân loại của mô hình, từ đó phản ánh được hiệu năng tổng thể của hệ thống giám sát.

Chương 3: Hệ thống giám sát an ninh, nhận dạng đối tượng xâm nhập.

3. 1. Đặt lại bài toán

Trong môi trường ngân hàng, nơi mà an ninh và bảo mật là yếu tố sống còn, việc phát hiện kịp thời các hành vi xâm nhập và những tình huống bất thường là điều cần thiết nhưng vẫn còn là một thách thức lớn đối với các hệ thống giám sát truyền thống. Để khắc phục vấn đề này, bài toán được đặt ra cần tái xác định theo hướng phát triển một hệ thống giám sát an ninh tự động chuyên dụng, có khả năng phân tích video thời gian thực và phản ứng nhanh với các mối đe dọa. Cụ thể, hệ thống sẽ áp dụng mô hình YOLOv8 để nhận diện các đối tượng có nguy cơ xâm nhập ngay lập tức từ các khung hình video thu được từ camera giám sát. Trước đó, dữ liệu hình ảnh sẽ được xử lý qua các bước tiền xử lý với OpenCV và Numpy, giúp loại bỏ nhiễu, cân bằng ánh sáng và chuẩn hóa dữ liệu, nhằm tạo ra đầu vào chất lượng cao cho mô hình học sâu. Quá trình huấn luyện và suy luận của mô hình sẽ được tối ưu hóa bằng PyTorch kết hợp với CUDA, đảm bảo tốc độ xử lý đáp ứng yêu cầu thời gian thực ngay trong điều kiện ánh sáng và chất lượng hình ảnh biến đổi của môi trường ngân hàng. Hệ thống cũng sẽ tích hợp giao diện quản lý trực quan xây dựng trên nền tảng Flask, giúp người quản trị theo dõi và điều phối các phản ứng khi hệ thống tự động phát hiện sự cố, đồng thời gửi cảnh báo âm thanh tới người dùng. Ngoài ra, để mở rộng khả năng phân tích và theo dõi hành vi của đối tượng sau khi nhận diện, module theo dõi như DeepSort có thể được tích hợp vào hệ thống, cung cấp dữ liệu chi tiết về hành trình và chuyển động của đối tượng qua các khung hình video. Từ đó, bài toán được đặt lại không chỉ là việc nhận diện đối tượng mà còn là xây dựng một hệ thống giám sát toàn diện, liên tục cập nhật và phản ứng với mọi tình huống bất thường, đảm bảo an toàn tối đa cho các cơ sở ngân hàng.

3. 2. Mô tả bộ dữ liệu sử dụng.

Dữ liệu hình ảnh và video được thu thập từ các camera an ninh chất lượng, đảm bảo phản ánh được điều kiện ánh sáng và môi trường thực tế trong nội bộ ngân hàng. Bên cạnh đó, một số dữ liệu bổ trợ được lấy từ các bộ dữ liệu công khai có tính chất tương đồng (COCO, Roboflow).

✧ Đặc điểm dữ liệu:

Số lượng ảnh/video thu thập được khoảng 600 mẫu, với định dạng ảnh JPEG và video MP4.

Các nhãn gán chính gồm: “dangerous object”, “object” và “weapon”.

Dữ liệu được chia theo tỷ lệ: 80% dành cho huấn luyện, 10% dùng để kiểm tra (validation) và 10% dùng cho thử nghiệm (test), nhằm đảm bảo tính ngẫu nhiên và đại diện của từng tập mẫu.

❖ Phương pháp gán nhãn:

Việc gán nhãn được thực hiện thủ công trên Roboflow. Điều này giúp giảm thiểu sai sót trong quá trình huấn luyện và nâng cao hiệu quả của mô hình trong việc nhận diện các đối tượng cần theo dõi.

3. 2. 1. Xây dựng công cụ trích xuất frame

- ❖ Sử dụng OpenCV Python để xây dựng công cụ trích xuất frame từ video. Điểm đặc biệt của đoạn code này là chỉ lưu lại các frame có thứ tự chia hết cho 10 (cứ mỗi frame sẽ lưu 1 ảnh). Frame sẽ được lưu trữ trong thư mục chỉ định.
- ❖ Sử dụng công cụ này vào mục đích trích xuất frame để gán nhãn.

3. 2. 2. Công cụ RoboFlow trong việc gán nhãn Dataset

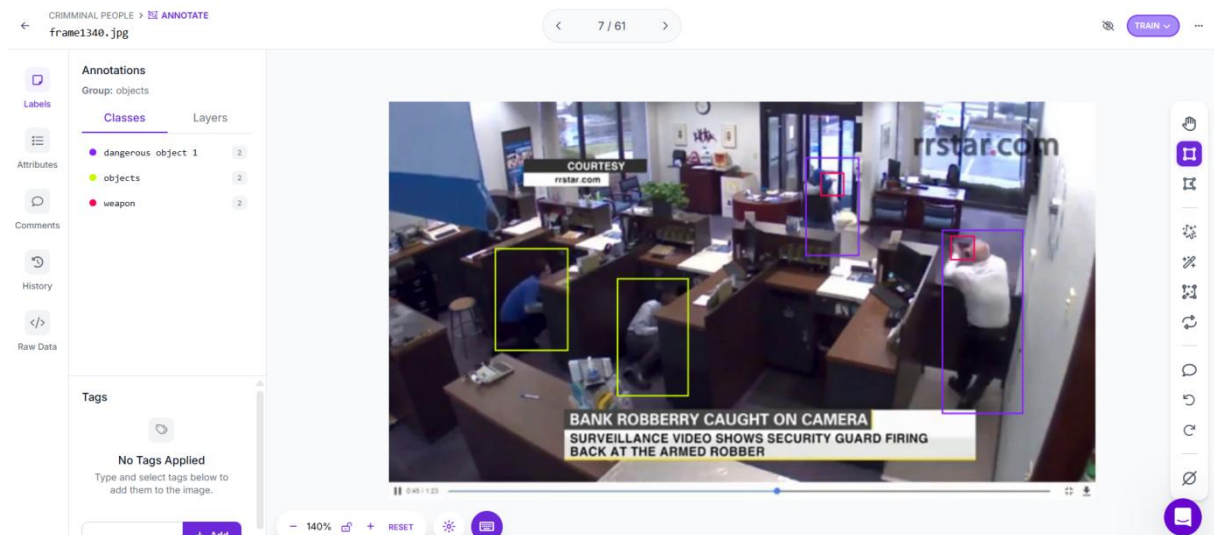


Image 4: Chuẩn bị Dataset, nguồn: chính chủ

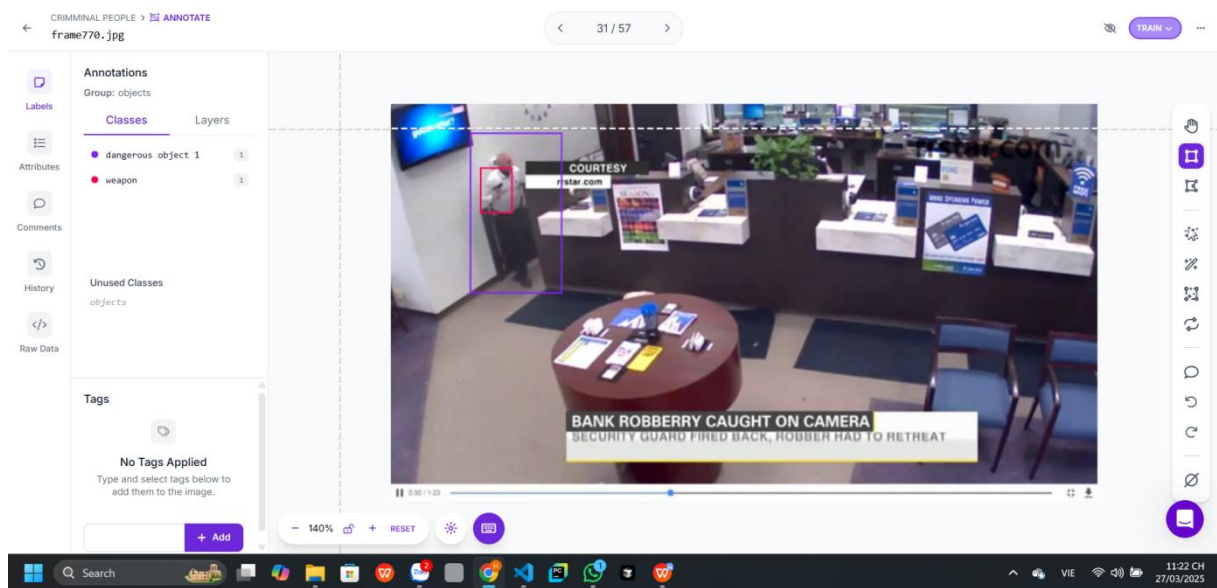


Image 5: Chuẩn bị dataset, nguồn: chính chủ

Check lại dữ liệu bằng thủ công

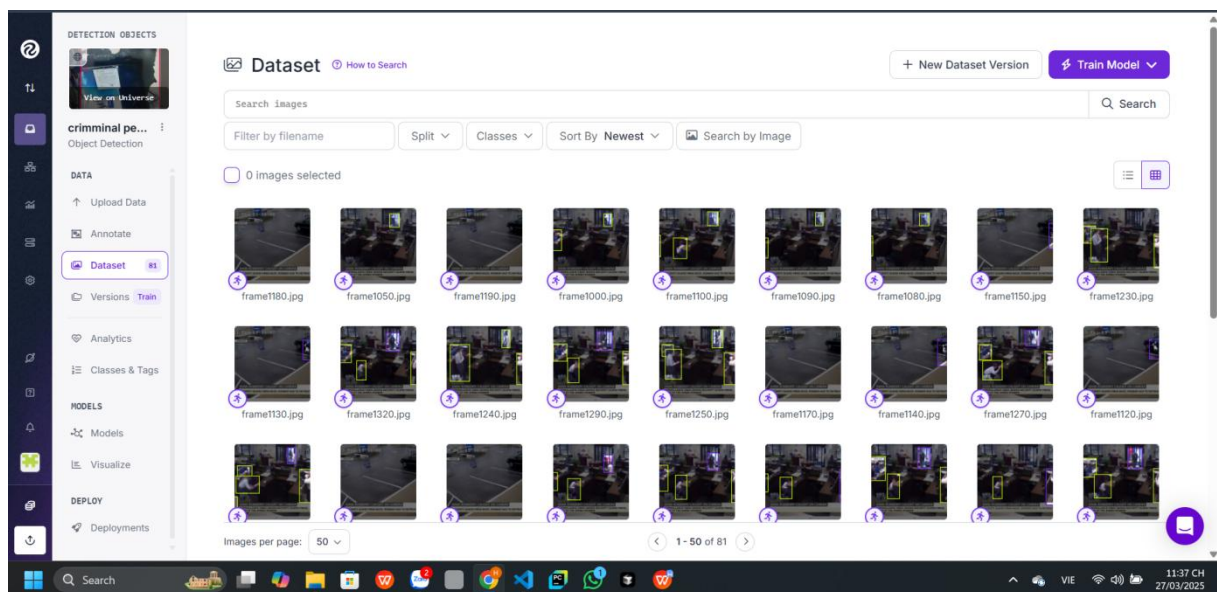


Image 6: Chuẩn bị dataset, nguồn: chính chủ

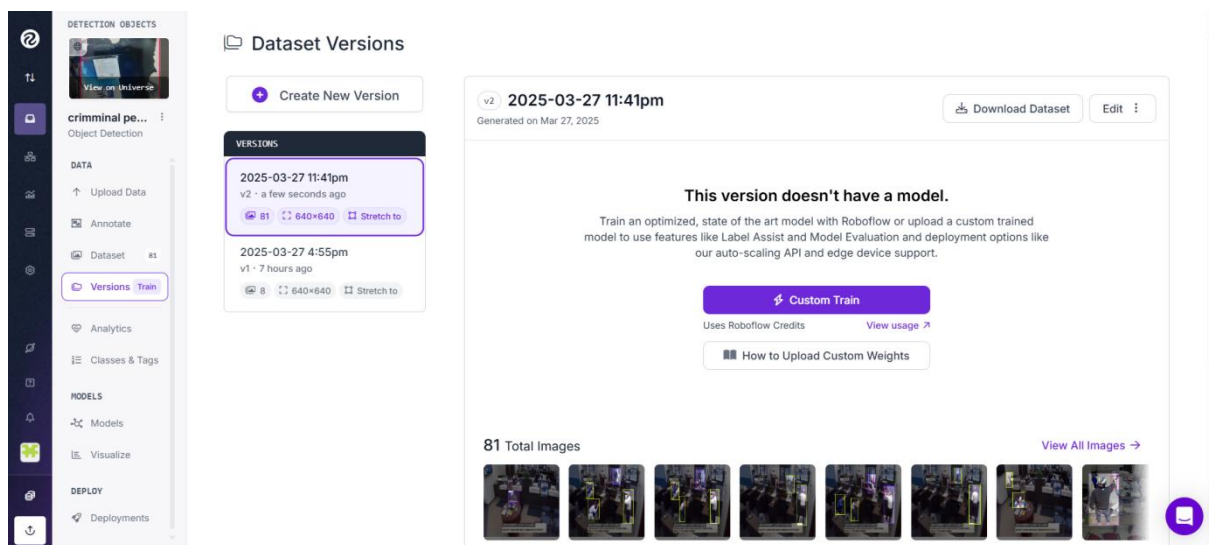


Image 7: Chuẩn bị dataset, nguồn: chính chủ

3.3. Train Model

Quá trình train diễn ra nhiều lần, sử dụng local computer và google colab:

✧ Local computer

Thiết lập môi trường kiểm thử:

Các mô hình YOLOv8n và YOLOv8s được huấn luyện với số epoch cố định (ví dụ: 50 epochs) trên GPU cấu hình AMD Ryzen 5 5600H (6 nhân/12 luồng), RAM 16GB, SSD 512GB, Radeon™ Graphics, GeForce RTX 3050, với batch size được đặt là 16 và learning rate 0.01.

Môi trường chạy được thiết lập trên hệ điều hành Windows, sử dụng thư viện OpenCV để xử lý hình ảnh, kết hợp với PyTorch để huấn luyện và suy diễn mô hình.

Kịch bản kiểm thử:

Kịch bản 1: Kiểm thử với ảnh đầu vào rõ nét, ánh sáng đầy đủ

Mô hình được chạy trên các khung hình thu được từ video giám sát, trong đó các đối tượng được chụp với chất lượng cao. Mục tiêu là đánh giá khả năng nhận diện chính xác các lớp đối tượng “dangerous object”, “object” và “weapon”.

Kịch bản 2: Kiểm thử với ảnh đầu vào có điều kiện ánh sáng yếu hoặc bị nhiễu

Các khung hình được xử lý qua bước tiền xử lý với OpenCV để loại bỏ nhiễu và cân bằng ánh sáng. Từ đó, đánh giá khả năng của mô hình trong việc nhận diện đối tượng khi gặp phải các điều kiện thực tế không lý tưởng.

Kịch bản 3: Kiểm thử với dữ liệu ngoài tập huấn luyện

Các mẫu ảnh chưa từng được mô hình “thấy” trong quá trình huấn luyện được đưa vào thử nghiệm nhằm đánh giá khả năng tổng quát hóa của mô hình. Kịch bản này giúp phát hiện các hiện tượng overfitting và kiểm tra tính ổn định của mô hình khi triển khai thực tế.

Chỉ số đánh giá:

Các chỉ số chính được sử dụng để đánh giá bao gồm: Precision, Recall, F1-Score và mAP@0.5. Bên cạnh đó, ma trận nhầm lẫn (confusion matrix) cũng được phân tích để xác định tỷ lệ dự đoán đúng và các lỗi do mô hình gây ra.

3. 4. Đánh giá mô hình

a. Độ hội tụ của Loss

- ✧ **train/box_loss, train/cls_loss, train/dfl_loss** đều có xu hướng giảm ổn định từ epoch đầu đến cuối (từ ~1.8 xuống ~0.9–1.0 với box_loss, và từ ~3.2 xuống ~2.7 với cls_loss).
- ✧ Tương tự **val/box_loss, val/cls_loss, val/dfl_loss** cũng giảm theo, chứng tỏ mô hình không bị overfit nghiêm trọng.

Kết luận: quá trình fine-tune diễn ra tốt, mô hình học dần cách tối ưu vị trí, phân lớp và phân phối bbox (DFL).

b. Precision & Recall

- ✧ **Precision(B)** nhảy múa khá nhiều (từ ~0.0 lên ~0.8 rồi dao động quanh 0.6–0.7), nghĩa là đôi lúc mô hình vẫn còn dự đoán “chìm” (false positives) khi chưa được điều chỉnh chắc.
- ✧ **Recall(B)** ổn định hơn, tăng từ ~0.6 lên ~0.7–0.8 sau epoch 18, cho thấy khả năng “bắt” được target (true positives) khá tốt.

Kết luận: bạn có thể thêm kỹ thuật giảm false positives (ví dụ tăng IoU threshold hoặc NMS strictness) để cải thiện Precision, hoặc nâng batch size/learning rate schedule để ổn định hơn.

c. mAP 0.5 & mAP 0.5 - 0.95

- ✧ **mAP@0.5** đạt ngưỡng ~0.6 và duy trì ổn định quanh đó sau epoch 10.

- ✧ **mAP@0.5:0.95** dao động quanh $\sim 0.3\text{--}0.4$, có một đợt peak lên ~ 0.38 ở epoch 20–25.

Kết luận:

- ✧ mAP@0.5 đạt $\sim 60\%$ là khá chấp nhận được với một task nhiều class (3 lớp).
- ✧ mAP@0.5:0.95 thấp hơn ($\sim 40\%$) vì metric này khắt khe hơn—mô hình gặp khó với các bbox chính xác cao (>0.75 IoU).

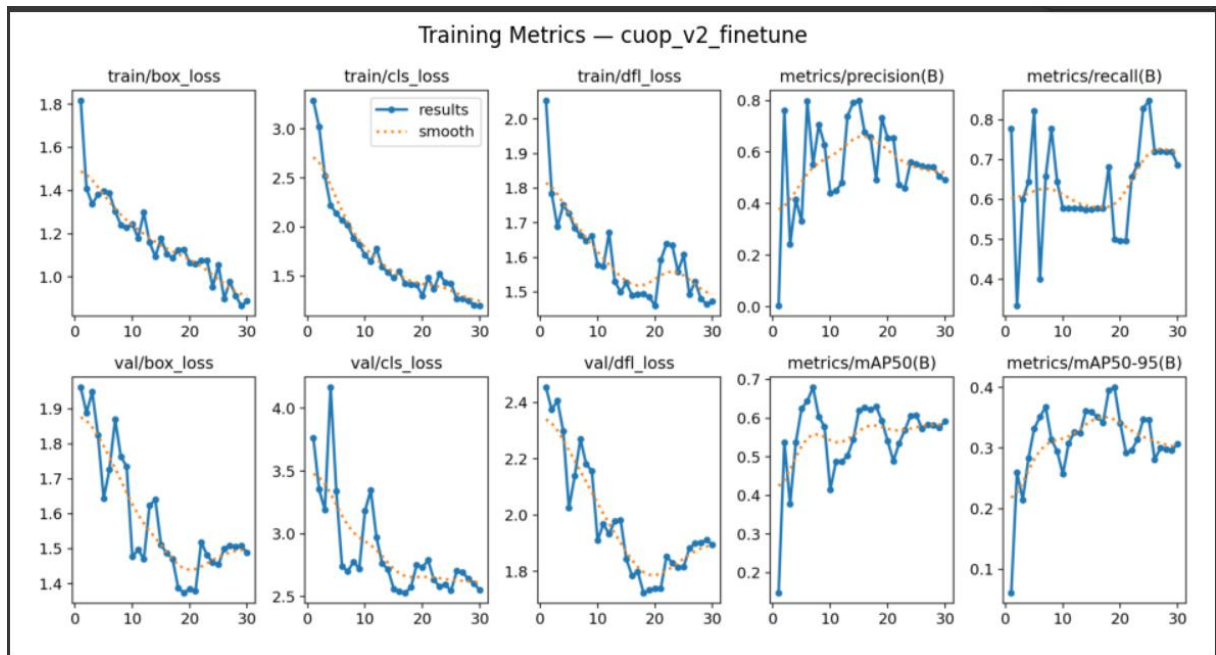


Image 8: Kết quả train model, nguồn: chính chủ



Image 9: kết quả train model, nguồn: chính chủ

3. 5. Xây dựng hệ thống giám sát và phát hiện đối tượng

Hệ thống giám sát an ninh được xây dựng nhằm đáp ứng nhu cầu phát hiện và cảnh báo kịp thời các hành vi xâm nhập trong môi trường thực tế, đặc biệt là tại các cơ sở ngân hàng. Ứng dụng sử dụng mô hình YOLOv8 tiên tiến để nhận diện đối tượng trong thời gian thực qua các khung hình video thu được từ camera. Các kết quả xử lý được hiển thị trên giao diện web xây dựng bằng Flask, đồng thời tích hợp chức năng gửi cảnh báo âm thanh khi phát hiện đối tượng nguy hiểm. Hệ thống không chỉ đảm bảo hiệu suất nhận diện cao mà còn mang lại trải nghiệm trực quan, giúp quản trị viên dễ dàng theo dõi và xử lý các tình huống bất thường ngay lập tức.

3. 5. 1. Các thành phần chính:

Client: Người dùng nhập địa chỉ webcam để hệ thống lấy dữ liệu theo thời gian thực và email để hệ thống gửi thông báo.

Server: Sẽ sử dụng Flask và SocketIO. Bao gồm:

- Video Stream Manager: Kết nối đến webcam qua OpenCV.
- Model Cache: Load YOLOv8 vào RAM khi khởi động server để tăng tốc độ.
- Alert Engine: So sánh kết quả YOLOv8 với “Object dangerous”, gửi cảnh báo âm thanh tới người dùng.

Database: Lưu trữ lịch sử cảnh báo, thông tin người dùng.

3. 5. 2. Thiết kế hệ thống kiến trúc phần mềm.

✧ Sử dụng Client-Server architecture.

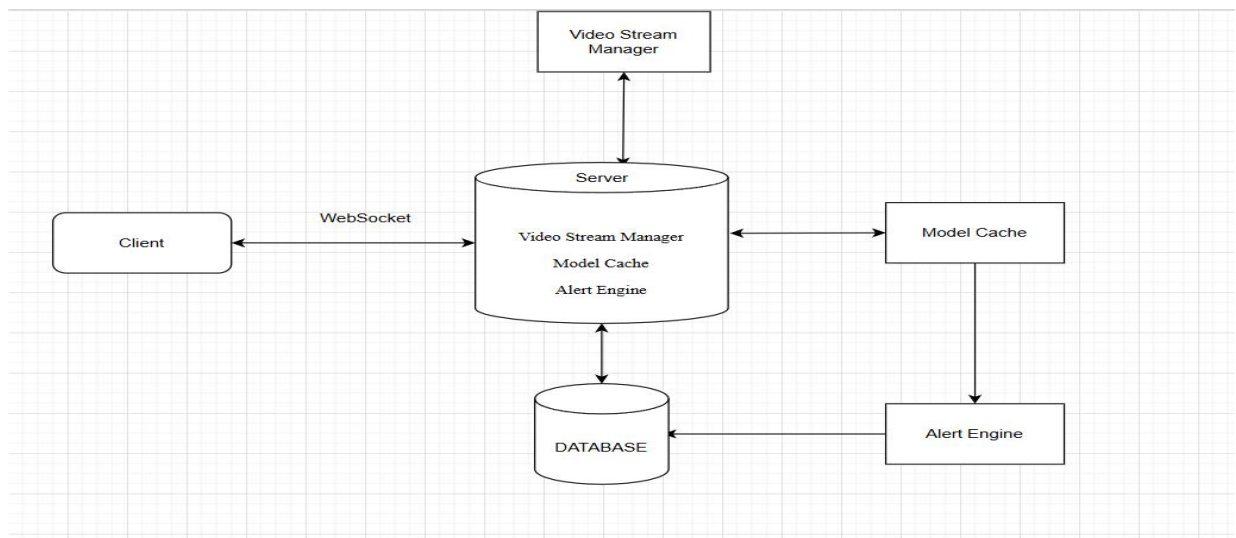


Image 10: Kiến trúc hệ thống tổng thể, nguồn: chính chủ

3. 5. 3. Thiết kế cơ sở dữ liệu

✧ Entity Relationship Diagram

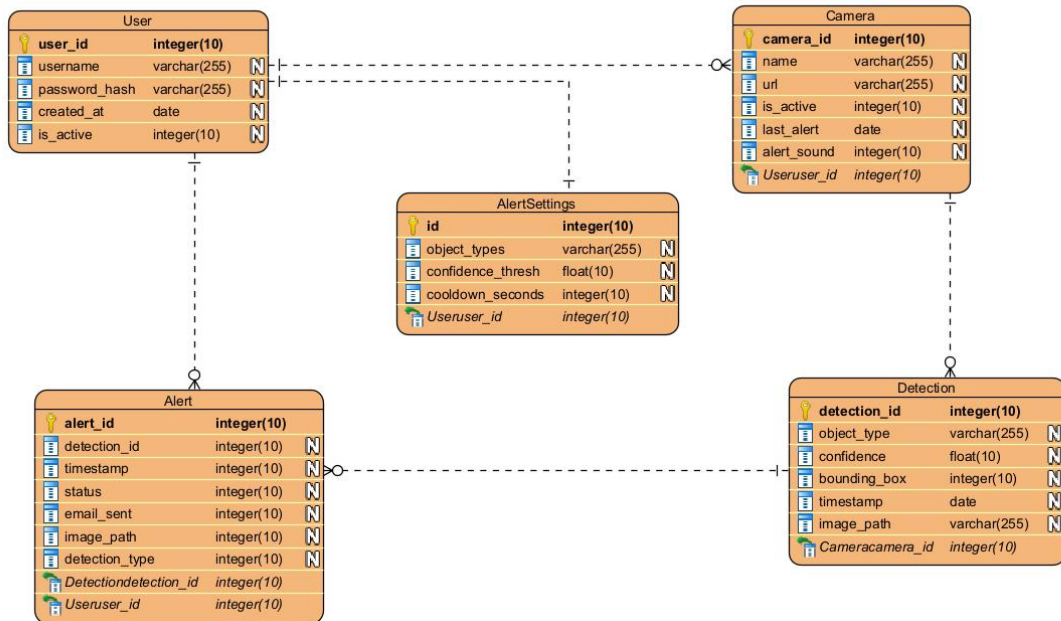


Image 11: Entity Relationship Diagram, nguồn: chính chủ

✧ Mối quan hệ giữa các thực thể chính:

a. User (Người dùng) - Camera:

Mối quan hệ một-nhiều: Một người dùng có thể sở hữu/quản lý nhiều camera

User chứa user_id (khóa chính)

Camera chứa user_id (khóa ngoại) liên kết với User

b. Camera - Detection (Phát hiện):

Mối quan hệ một-nhiều: Một camera có thể có nhiều phát hiện đối tượng

Camera chứa camera_id (khóa chính)

Detection chứa camera_id (khóa ngoại) liên kết với Camera

c. Detection - Alert (Cảnh báo):

Mối quan hệ một-nhiều: Một phát hiện có thể tạo ra nhiều cảnh báo

Detection chứa detection_id (khóa chính)

Alert chứa detection_id (khóa ngoại) liên kết với Detection

d. User - Alert:

Mối quan hệ một-nhiều: Một người dùng có thể nhận nhiều cảnh báo

User chứa user_id (khóa chính)

Alert chứa user_id (khóa ngoại) liên kết với User

e. User - AlertSettings (Cài đặt cảnh báo):

Mối quan hệ một-một: Mỗi người dùng có một bộ cài đặt cảnh báo

User chứa user_id (khóa chính)

AlertSettings chứa user_id (khóa ngoại) liên kết với User.

3. 5. 4. Mô tả nghiệp vụ

a. Các actor chính:

Người dùng: người trực tiếp sử dụng hệ thống để quản lý và giám sát camera

b. Các Use Case chính:

- ✧ **Thêm camera mới:** Người dùng nhập tên camera và url camera. Người dùng chọn nút “Thêm camera” để thêm camera vào database, hoặc chọn nút “Hủy” để hủy thêm camera. Người dùng có thể chọn âm thanh cảnh báo hoặc sử dụng âm thanh mặc định.

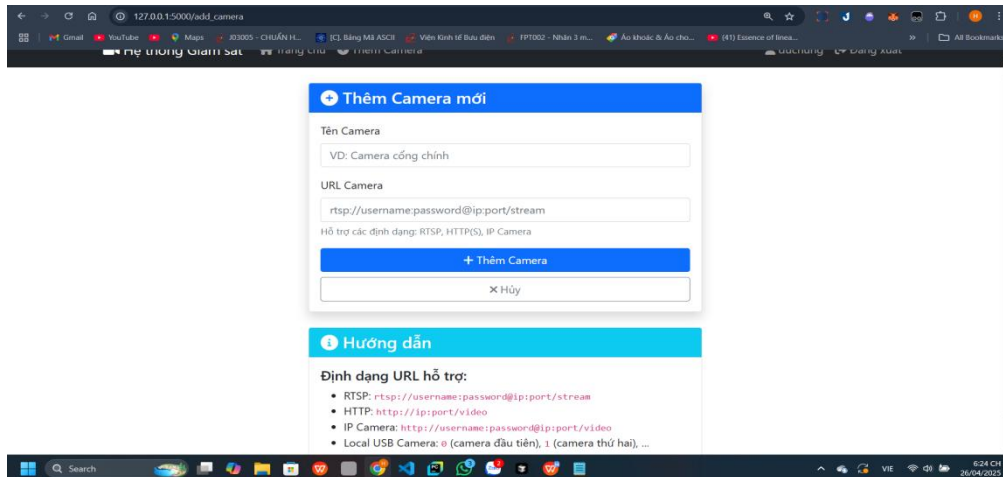


Image 12: Giao diện Thêm camera, nguồn chính chủ

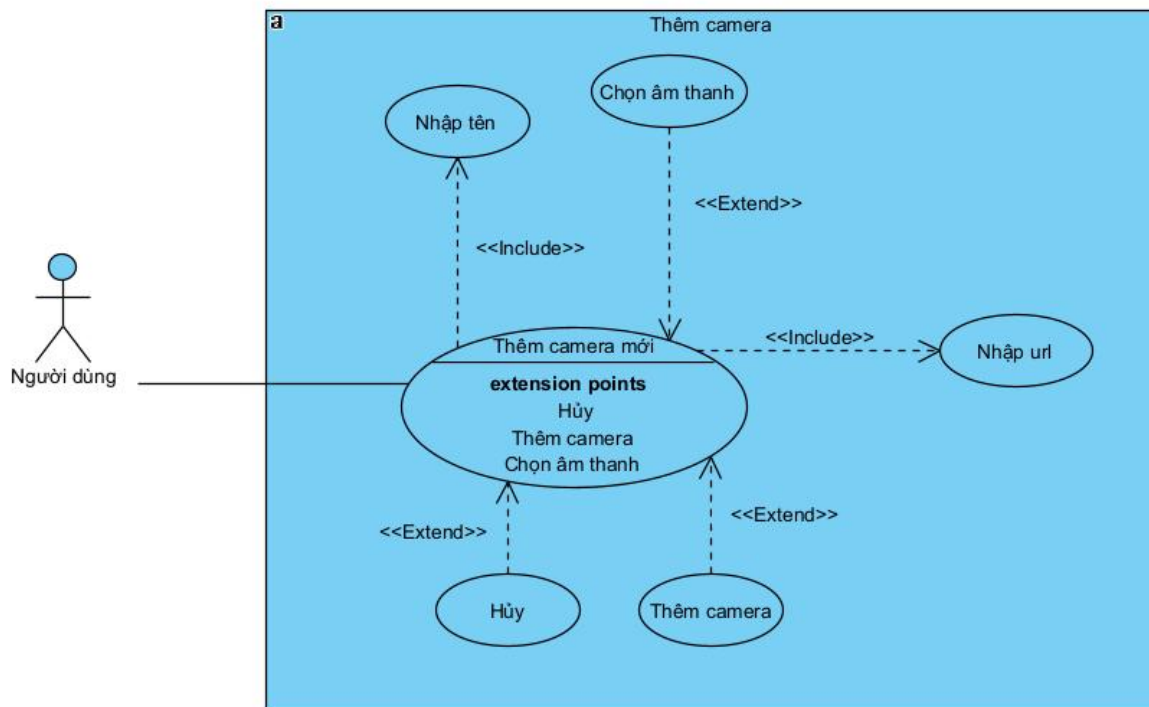



Image 13: Use case diagram "Thêm camera", nguồn: chính chủ

- ✧ **Cài đặt cảnh báo:** Người dùng chọn đối tượng cảnh báo, điều chỉnh ngưỡng tin cậy (confidence), điều chỉnh Thời gian chờ. Sau khi cài đặt xong, người dùng chọn Lưu cài đặt


Cài đặt cảnh báo

Đổi tượng cần cảnh báo:

☒ Dangerous Object
☒ Weapon

Ngưỡng tin cậy (0.1 - 1.0):

Chỉ gửi cảnh báo khi độ tin cậy vượt quá ngưỡng này

Thời gian chờ (giây):

Thời gian tối thiểu giữa các cảnh báo



Lưu cài đặt

Image 14: Giao diện "Cài đặt cảnh báo", nguồn: chính chủ

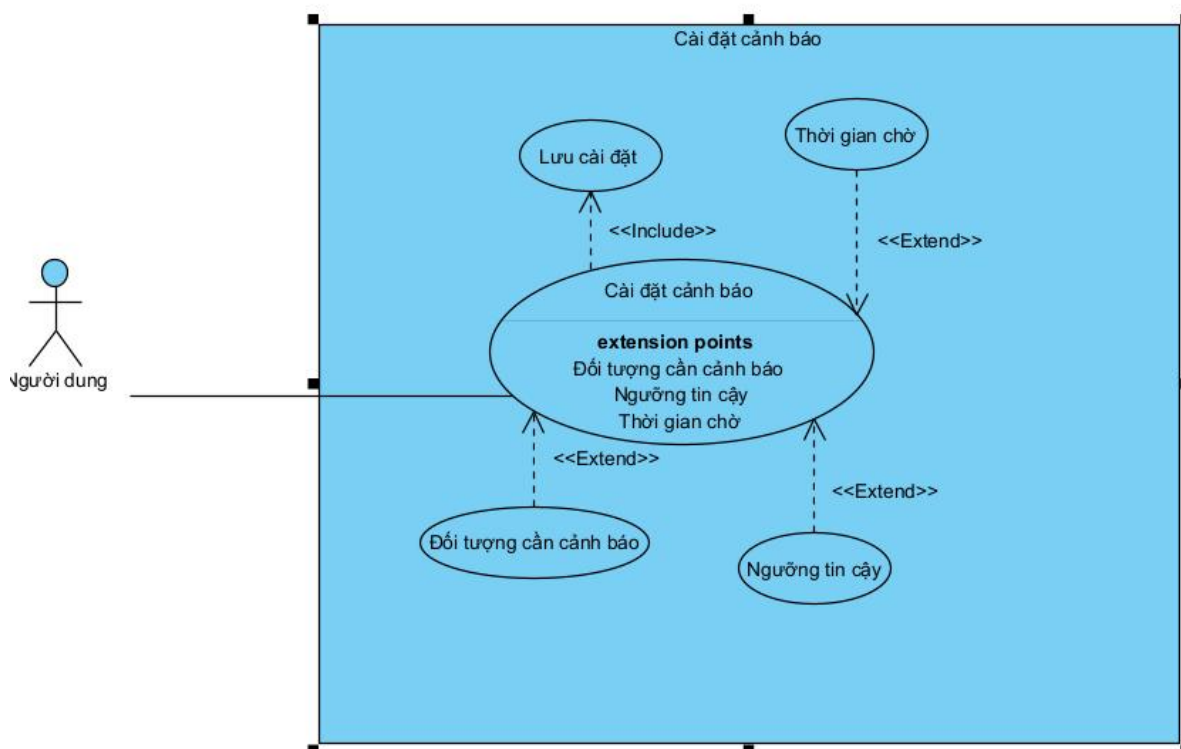


Image 15: Use case diagram "Cài đặt cảnh báo", nguồn: chính chủ

- Xem danh sách camera:** xem tất cả các camera đã thêm, xem trạng thái hoạt động, bật tắt bounding box.

Danh sách camera

Tên camera	URL	Trạng thái	Hành động
cam1	rtsp://184.72.239.149/vod/mp4:BigBuckBunny_175k.mov	Đang hoạt động	Xem Bật/Tắt khung
cam2	rtsp://807e9439d5ca.entrypoint.cloud.wowza.com:1935/app-rC94792j/068b9c9a_stream2	Đang hoạt động	Xem Bật/Tắt khung

Image 16: Giao diện "Danh sách camera", nguồn: chính chủ

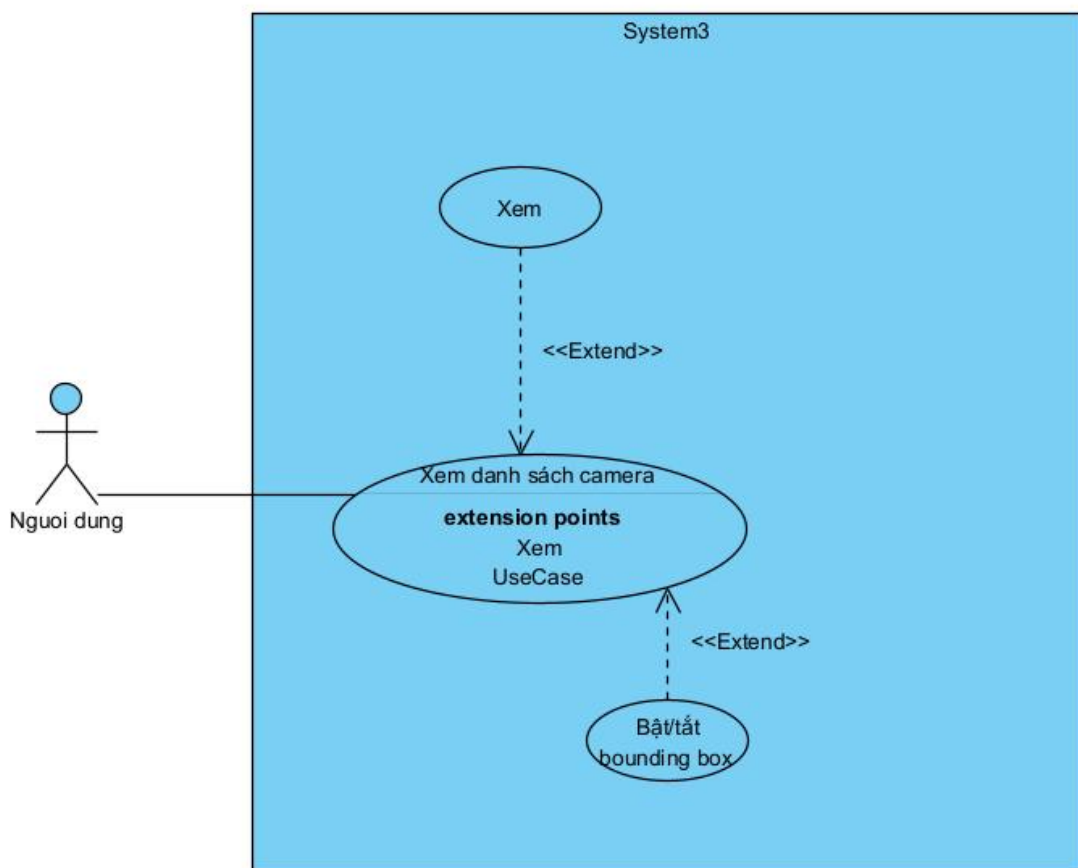


Image 17: Use case diagram "Danh sách camera", nguồn: chính chủ

✧ **Quản lý camera:** bật tắt camera, sửa thông tin camera, xóa camera, bật/tắt bounding box, xem cảnh báo gần đây.

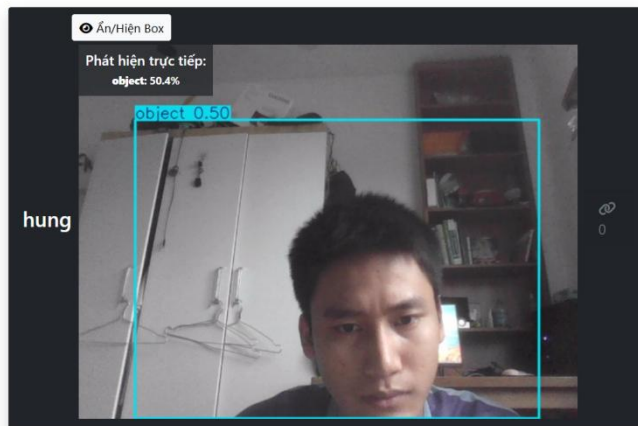


Image 18: Giao diện "Camera Detail", nguồn: chính chủ

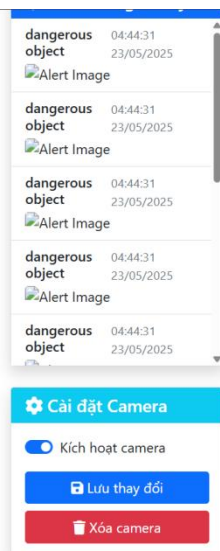


Image 19 Giao diện "Camera Detail", nguồn: chính chủ

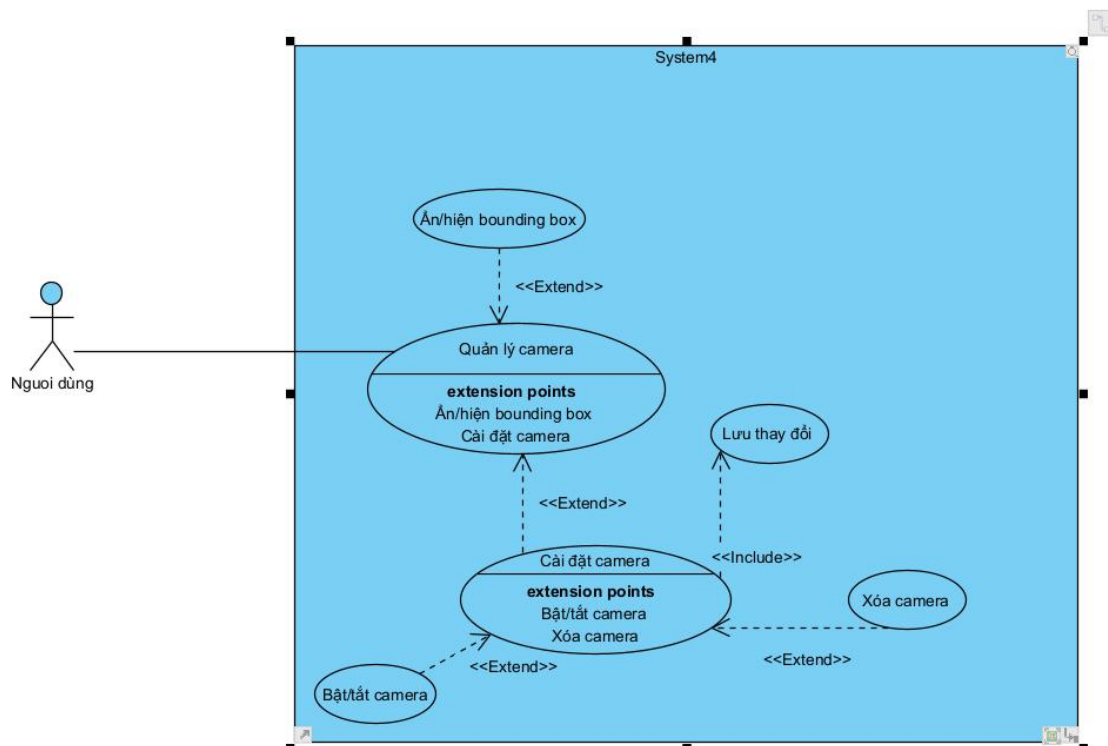


Image 20: Use case diahram "Cài đặt camera", nguồn: chính chủ

- ✧ **Xem cảnh báo:** Người dùng xem các cảnh báo gần đây, nút Xem chi tiết sẽ mở ra 1 pop-up bao gồm thông tin cơ bản của cảnh báo và frame. Người dùng có thể xóa cảnh báo đó.

Danh sách camera

Tên camera	URL	Trạng thái	Hành động
hung	0	Đang hoạt động	Xem. Bật/Tắt khung

Cảnh báo gần đây

dangerous object

Camera: hung

Thời gian: 03:23:53 23/05/2025

Xem chi tiết

dangerous object

Camera: hung

Thời gian: 03:23:53 23/05/2025

Xem chi tiết

dangerous object

Image 21: Giao diện "Danh sách camera", nguồn: chính chủ

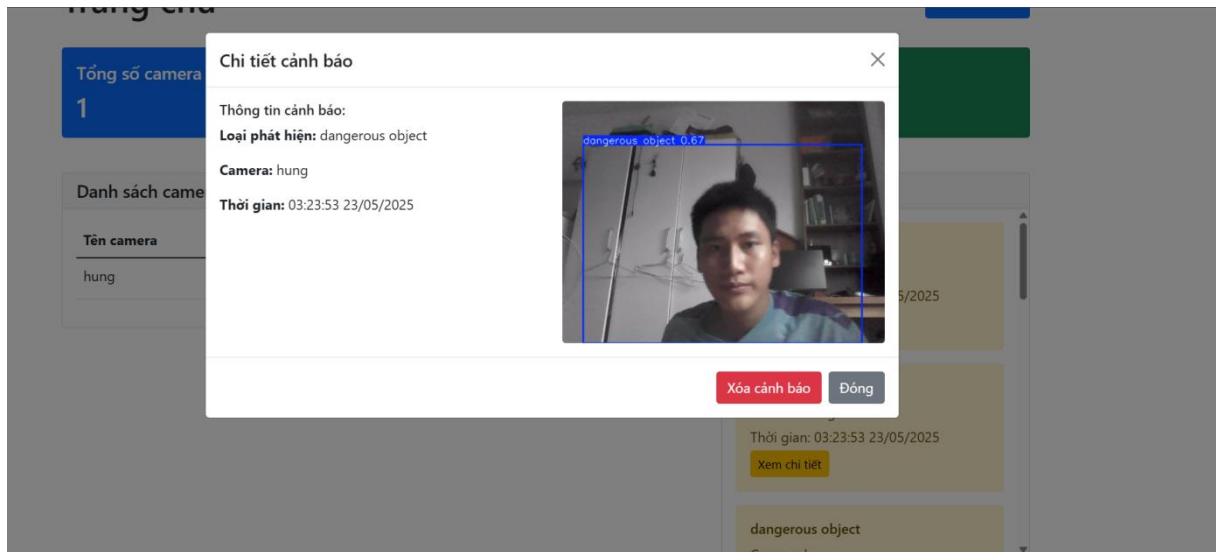


Image 22: Giao diện pop-up Xem chi tiết cảnh báo, nguồn: chính chủ

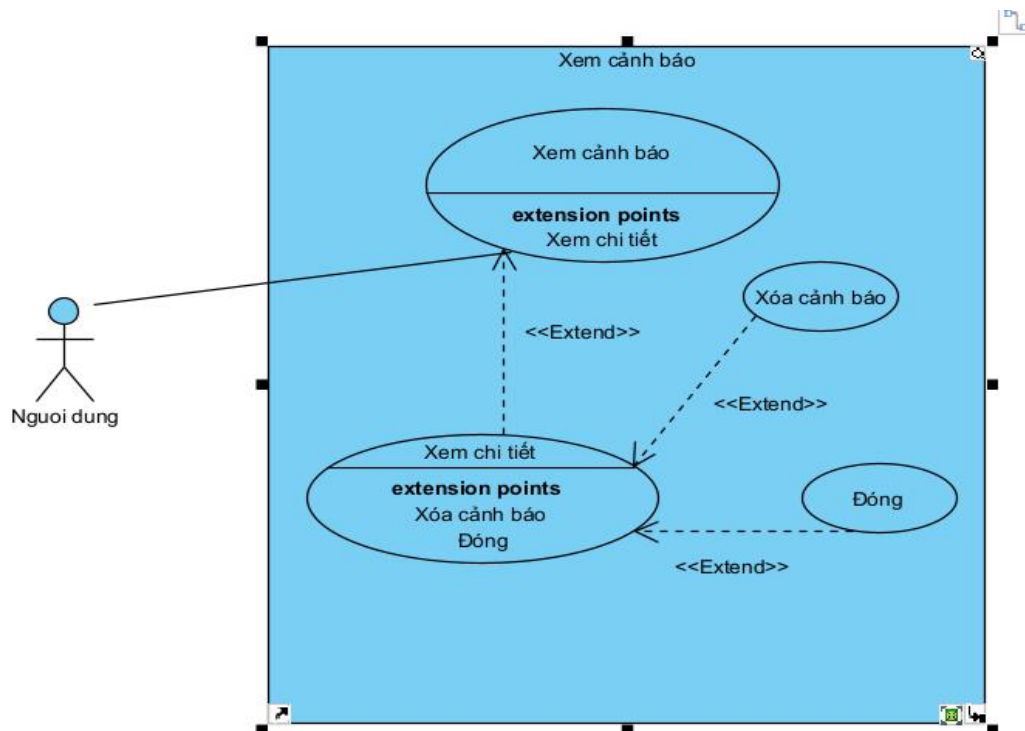


Image 23: Use case diagram "Xem chi tiết cảnh báo", nguồn: chính chủ

c. Sơ đồ tuần tự

Sẽ chỉ có 2 chức năng có sự tương tác chính là người dùng.

✧ Thêm camera

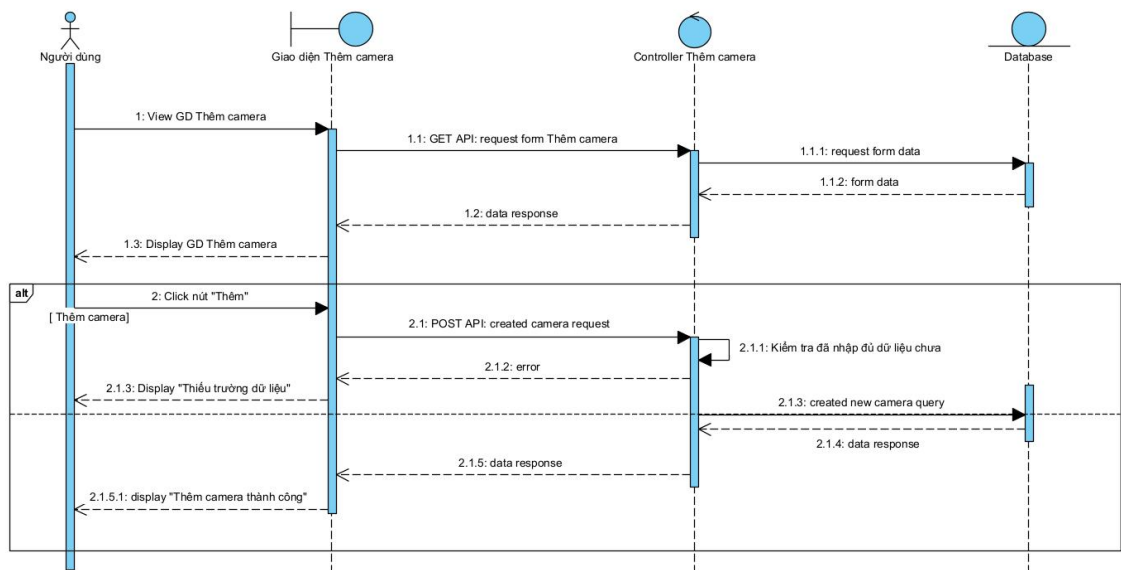


Image 24: Sequence Diagram "Thêm camera", nguồn: chính chủ

✧ Cài đặt cảnh báo

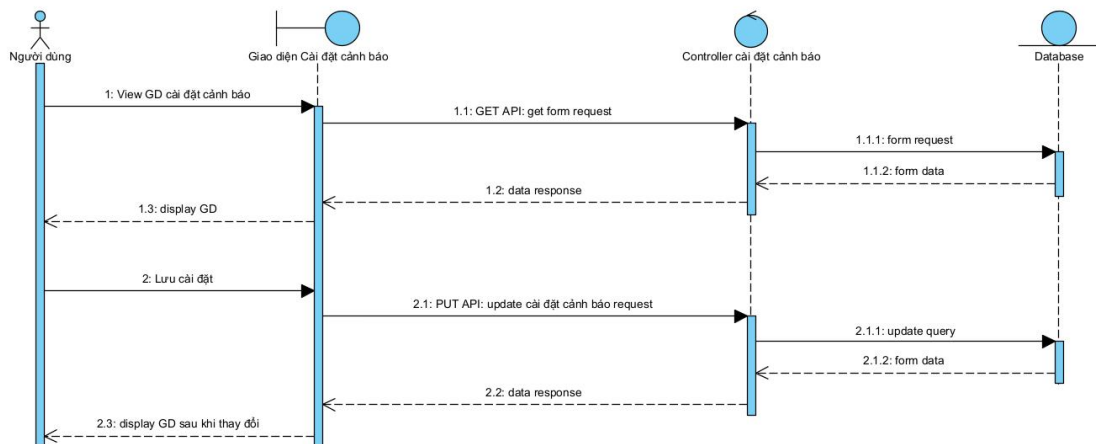


Image 25: Sequence Diagram "Cài đặt cảnh báo", nguồn: chính chủ

3. 5. 5. Triển khai Hệ thống

a. Backend Framework và Database:

- ✧ Sử dụng Flask (Python) làm web framework chính
- ✧ SQLAlchemy làm ORM (Object-Relational Mapping) để tương tác với database
- ✧ Cấu trúc database được thiết kế với các model chính:
 - User: Quản lý người dùng và xác thực
 - Camera: Quản lý thông tin camera

- Detection: Lưu trữ các phát hiện đối tượng
- Alert: Quản lý cảnh báo
- AlertSettings: Cấu hình cảnh báo cho từng người dùng

b. Frontend Technologies:

- ✧ Bootstrap 5 cho UI/UX hiện đại và responsive
- ✧ Bootstrap Icons cho các biểu tượng
- ✧ JavaScript thuần cho xử lý tương tác người dùng
- ✧ Modal dialogs cho hiển thị chi tiết cảnh báo
- ✧ AJAX/Fetch API cho các tương tác bất đồng bộ

c. Xử lý Video và Computer Vision:

- ✧ Hỗ trợ nhiều loại camera khác nhau:
 - RTSP streams
 - HTTP/HTTPS streams
 - IP cameras
 - Local USB cameras

d. Khả năng phát hiện đối tượng (object detection) với model YOLO:

- ✧ Lưu trữ bounding boxes
- ✧ Độ tin cậy (confidence) của phát hiện
- ✧ Lưu trữ hình ảnh phát hiện

e. Hệ thống Cảnh báo:

- ✧ Âm thanh cảnh báo có thể tùy chỉnh
- ✧ Cấu hình cảnh báo linh hoạt:
 - Loại đối tượng cần cảnh báo
 - Ngưỡng độ tin cậy
 - Thời gian cooldown giữa các cảnh báo

f. Bảo mật:

- ✧ Xác thực người dùng với Flask-Login

- ✧ Mật khẩu được mã hóa với Werkzeug Security
- ✧ Hỗ trợ HTTPS cho kết nối an toàn
- ✧ Phân quyền người dùng và camera

g. Tính năng Quản lý:

- ✧ Dashboard tổng quan với thống kê
- ✧ Quản lý camera:
 - Thêm/xóa camera
 - Bật/tắt camera
 - Bật/tắt khung phát hiện
- ✧ Quản lý cảnh báo:
 - Xem chi tiết cảnh báo
 - Xóa cảnh báo
 - Cấu hình cảnh báo

h. Tối ưu hóa:

- ✧ Lazy loading cho các relationship trong database
- ✧ Responsive design cho mọi thiết bị
- ✧ Xử lý bất đồng bộ cho các tác vụ nặng
- ✧ Caching cho các tài nguyên tĩnh
- ✧ Khả năng Mở rộng:
- ✧ Kiến trúc module hóa cho dễ dàng thêm tính năng mới
- ✧ Hỗ trợ nhiều loại camera và stream
- ✧ Có thể tích hợp thêm các model AI khác
- ✧ Dễ dàng thêm các kênh cảnh báo mới

3. 5. 6. Đề xuất hướng cải tiến Hệ thống

a. Cải thiện Độ chính xác và Linh hoạt của Phát hiện:

- ✧ Hỗ trợ nhiều loại đối tượng hơn: Mở rộng khả năng phát hiện không chỉ giới hạn ở "dangerous object" và "weapon". Cho phép người dùng tùy chọn các loại đối tượng khác và âm thanh cảnh báo.

- ✧ Vùng quan tâm (Region of Interest - ROI): Cho phép người dùng định nghĩa các khu vực cụ thể trong khung hình camera mà họ muốn giám sát. Cảnh báo chỉ được kích hoạt khi đối tượng xuất hiện trong vùng ROI này. Điều này giúp giảm thiểu cảnh báo sai và tập trung vào các khu vực quan trọng.
- ✧ Phát hiện hành vi: Thay vì chỉ phát hiện đối tượng tĩnh, phát triển khả năng nhận diện các hành vi bất thường (ví dụ: đột nhập, vật thể bị bỏ rơi, tụ tập đông người).

b. Mở rộng Khả năng Xử lý Camera:

- ✧ Xử lý đa luồng hiệu quả hơn: Với nhiều camera, việc xử lý từng luồng có thể tốn tài nguyên. Cân nhắc sử dụng các kỹ thuật xử lý song song hoặc phân tán (ví dụ: sử dụng hàng đợi công việc như Celery, hoặc thậm chí triển khai trên nhiều server nếu cần) để xử lý đồng thời nhiều luồng video.
- ✧ Tùy chọn chất lượng stream: Cho phép người dùng chọn chất lượng video stream (ví dụ: độ phân giải, tốc độ khung hình) để cân bằng giữa chất lượng hình ảnh và tải xử lý/mạng. Tính năng này có thể phân level người dùng, những người đăng ký tài khoản vip sẽ có thể lựa chọn chất lượng stream tốt hơn.

c. Nâng cao Trải nghiệm Người dùng:

- ✧ Giao diện Dashboard tương tác hơn: Hiện thị trạng thái camera trực tiếp (online/offline) một cách rõ ràng, biểu đồ hoặc thống kê về số lượng cảnh báo theo thời gian, hoặc bản đồ hiển thị vị trí camera (nếu áp dụng).
- ✧ Quản lý cảnh báo nâng cao: Thêm chức năng lọc, tìm kiếm cảnh báo theo loại, thời gian, camera. Cho phép đánh dấu cảnh báo là đã xem/chưa xem hoặc phân loại chúng.
- ✧ Push Notifications: Thay vì chỉ dựa vào cảnh báo âm thanh trên web, tích hợp Push Notifications thông qua trình duyệt hoặc phát triển ứng dụng di động để người dùng nhận cảnh báo ngay lập tức trên điện thoại.
- ✧ Lịch sử video: Lưu trữ các đoạn video ngắn khi cảnh báo xảy ra (thay vì chỉ lưu frame ảnh) hoặc cung cấp tùy chọn ghi hình liên tục.

d. Cải thiện model

- ✧ Do chưa có đủ tài nguyên, quá trình train model gặp nhiều vấn đề nên độ chính xác model chưa được như mong đợi. Sẽ cần cải thiện model nhiều hơn.

e. Các tính năng bổ sung khác:

- ✧ Hỗ trợ đa người dùng/đa vai trò: Nếu hệ thống được sử dụng bởi nhiều người trong một tổ chức, thêm các vai trò (admin, user) với quyền hạn khác nhau.
- ✧ Tích hợp với các hệ thống khác: Kết nối với các hệ thống nhà thông minh hoặc các dịch vụ thông báo khác (ví dụ: Telegram, Slack) để gửi cảnh báo.

Kết luận

Qua quá trình nghiên cứu, xây dựng và thử nghiệm, hệ thống giám sát an ninh ứng dụng mô hình YOLOv8 đã chứng minh được hiệu quả trong việc nhận diện đối tượng xâm nhập tại môi trường ngân hàng. Việc kết hợp các công nghệ hiện đại như OpenCV, Numpy, PyTorch và Flask không chỉ giúp tối ưu hóa quy trình xử lý hình ảnh, mà còn đảm bảo khả năng phát hiện nhanh chóng, chính xác các mối nguy hiểm ẩn trong điều kiện thực tế đa dạng.

Các kết quả đánh giá cho thấy mô hình đạt độ chính xác và khả năng tổng quát hóa tốt, với các chỉ số mAP, Precision, Recall ổn định, đáp ứng, hệ thống đã đáp ứng được tốc độ tạm ổn. Song, chưa có đủ tài nguyên để train model tốt hơn nên việc nhận dạng chưa đạt được kết quả như mong đợi.

Bên cạnh đó, giao diện quản lý trực quan, chức năng cảnh báo đa dạng và kiến trúc phần mềm mở rộng giúp hệ thống dễ dàng triển khai, vận hành và nâng cấp trong tương lai. Tuy nhiên, vẫn còn nhiều hướng cải tiến tiềm năng như nâng cao độ chính xác nhận diện, hỗ trợ nhiều loại đối tượng/phân vùng giám sát, tối ưu xử lý đa luồng và tích hợp các kênh cảnh báo mới nhằm đáp ứng tốt hơn nhu cầu thực tế của từng đơn vị sử dụng.