BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC ĐẠI NAM

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO BÀI TẬP LỚN

CÔNG NGHỆ THÔNG TIN TRONG CHUYỂN ĐỔI SỐ

Giám sát an toàn tại nhà máy – camera phát hiện nhân viên không đội mũ bảo hộ

Sinh viên thực hiện : **Dương Đức Cường**
Trần Hoàng Công Tuyển
Nguyễn Văn Hội

Ngành : **Công nghệ thông tin**

Giảng viên hướng dẫn : **ThS. Lê Trung Hiếu**

Lời cảm ơn

Trong suốt quá trình học tập và rèn luyện tại Trường Đại học Đại Nam, chúng em đã nhận được rất nhiều sự quan tâm, chỉ dẫn tận tình của quý thầy cô trong Khoa Công nghệ Thông tin. Chính nhờ sự dìu dắt và truyền đạt kiến thức quý báu từ các thầy cô, chúng em mới có nền tảng vững chắc để vận dụng vào thực tiễn và hoàn thành bài tập lớn này. Chúng em nhận thấy rằng, thành quả đạt được hôm nay không chỉ là kết quả của riêng bản thân mà còn là sự đóng góp to lớn từ sự nỗ lực của tập thể giảng viên đã tận tâm giảng dạy và hỗ trợ sinh viên trong suốt thời gian qua.

Đặc biệt, chúng em xin bày tỏ lòng biết ơn sâu sắc tới Thạc sĩ Lê Trung Hiếu, giảng viên hướng dẫn, người đã dành nhiều thời gian, công sức và tâm huyết để định hướng, theo sát quá trình thực hiện của chúng em. Thầy không chỉ giúp chúng em định hình rõ ràng hơn về ý tưởng, cách triển khai đề tài mà còn khích lệ tinh thần, tạo động lực để chúng em vượt qua những khó khăn, hạn chế về kiến thức và kinh nghiệm thực tiễn. Sự tận tâm, kiên nhẫn và trách nhiệm của thầy là nguồn động lực rất lớn, giúp chúng em hoàn thành bài báo cáo này một cách tốt nhất.

Bên cạnh đó, chúng em cũng xin gửi lời cảm ơn chân thành đến các anh chị, bạn bè đã luôn đồng hành, hỗ trợ và chia sẻ những kinh nghiệm hữu ích trong quá trình nghiên cứu và thực hiện đề tài. Những lời động viên và sự hỗ trợ kịp thời từ mọi người đã giúp chúng em thêm tự tin, kiên trì để khắc phục những khó khăn và hoàn thiện báo cáo.

Mặc dù đã nỗ lực hết mình với tinh thần trách nhiệm cao, song do hạn chế về kiến thức và kinh nghiệm thực tế, bài báo cáo không thể tránh khỏi những thiếu sót. Chúng em rất mong nhận được những ý kiến đóng góp quý báu từ quý thầy cô để đề tài được hoàn thiện hơn trong tương lai.

Cuối cùng, chúng em xin kính chúc Ban Giám hiệu, quý thầy cô trong Khoa Công nghệ Thông tin và đặc biệt là thầy Lê Trung Hiếu luôn dồi dào sức khỏe, thành công trong sự nghiệp giảng dạy và nghiên cứu. Chúng em xin chân thành cảm ơn!

Mục lục

Danh sách bảng

Danh sách hình vẽ

Danh sách giải thuật

Chương 1

Giới thiệu

1.1 Đặt vấn đề

Trong môi trường công nghiệp, nơi mà các hệ thống giám sát, điều khiển, cũng như các thiết bị thông minh ngày càng được tích hợp sâu rộng, yêu cầu đảm bảo an toàn dữ liệu lại càng trở nên quan trọng hơn bao giờ hết. Dữ liệu không chỉ dừng lại ở văn bản hay hình ảnh mà còn có thể là dữ liệu âm thanh, dữ liệu cảm biến hoặc tín hiệu điều khiển. Một sự cố mất mát hoặc bị thay đổi dữ liệu trong quá trình truyền tải không chỉ làm sai lệch thông tin, mà còn có thể dẫn đến các quyết định sai lầm, gây ra rủi ro cho toàn bộ hệ thống sản xuất. Do đó, việc phân tích và tìm ra giải pháp bảo mật toàn diện cho dữ liệu trong quá trình truyền tải là nhiệm vụ trọng tâm của đề tài này.

1.1.1 Phân tích bài toán

Trước hết, cần đề cập đến vấn đề **mã hoá dữ liệu**. Đây là một trong những giải pháp cơ bản và quan trọng nhất để bảo vệ thông tin khi truyền tải qua môi trường mạng công cộng. Khi dữ liệu được mã hoá, nó sẽ được biến đổi thành một dạng mà những người không có khoá giải mã sẽ không thể hiểu được nội dung gốc. Điều này giúp ngăn chặn các hành vi nghe lén, đánh cắp hoặc khai thác thông tin trái phép trong quá trình truyền tải. Việc áp dụng các thuật toán mã hoá tiên tiến không chỉ nâng cao mức độ bảo mật mà còn tạo niềm tin cho người dùng khi tham gia vào hệ thống.

Tiếp theo, **xác thực người dùng** đóng vai trò bảo đảm rằng chỉ những cá nhân hoặc thiết bị có quyền hợp lệ mới có thể gửi và nhận dữ liệu. Trong thực tế, nếu không có một cơ chế xác thực chặt chẽ, hệ thống sẽ dễ dàng bị kẻ tấn công giả mạo để chiếm quyền truy cập. Việc xây dựng một mô hình xác thực đáng tin cậy, chẳng hạn như sử dụng chữ ký số hoặc cặp khoá công khai – bí mật, giúp hệ thống ngăn chặn được những nguy cơ truy cập trái phép. Đây chính là

nền tảng để duy trì tính an toàn và tin cậy của toàn bộ quá trình trao đổi thông tin.

Ngoài ra, một yếu tố không kém phần quan trọng là **đảm bảo tính toàn vẹn của dữ liệu**. Trong quá trình truyền tải, dữ liệu có thể bị thay đổi do sự cố kỹ thuật, lỗi truyền dẫn hoặc bị tác động bởi các hành vi tấn công có chủ đích. Nếu dữ liệu không còn giữ nguyên trạng thái ban đầu, người nhận sẽ không thể biết được thông tin đã bị sai lệch, dẫn đến việc đưa ra các quyết định sai lầm. Do đó, việc áp dụng các cơ chế kiểm tra toàn vẹn, chẳng hạn như sử dụng hàm băm mật mã kết hợp với chữ ký số, là giải pháp cần thiết để bảo đảm rằng dữ liệu khi đến tay người nhận hoàn toàn chính xác và không bị chỉnh sửa so với dữ liệu gốc.

Từ những phân tích trên, có thể thấy rằng bài toán bảo mật trong quá trình truyền tải dữ liệu đòi hỏi phải kết hợp đồng thời nhiều giải pháp, trong đó ba trụ cột quan trọng nhất là: mã hoá dữ liệu, xác thực người dùng và kiểm tra tính toàn vẹn của thông tin. Chỉ khi ba yếu tố này được đảm bảo, hệ thống truyền thông mới có thể hoạt động một cách an toàn, tin cậy và đáp ứng được những yêu cầu khắt khe trong thực tiễn triển khai, đặc biệt là trong môi trường công nghiệp và chuyển đổi số [21].

1.2 Mục tiêu của đề tài

Đề tài được thực hiện với mong muốn xây dựng một hệ thống giám sát và truyền tải dữ liệu có tính bảo mật cao, nhằm ứng dụng trong bối cảnh công nghiệp và chuyển đổi số hiện nay. Việc nghiên cứu không chỉ dừng lại ở khía cạnh lý thuyết mà còn hướng tới việc triển khai thực tế, góp phần nâng cao khả năng bảo vệ thông tin, đảm bảo an toàn trong hoạt động sản xuất cũng như trong đời sống xã hội. Trên cơ sở đó, các mục tiêu cụ thể của đề tài được xác định như sau:

- **Mục tiêu 1:** Xây dựng cơ chế **mã hoá dữ liệu** nhằm bảo đảm thông tin trong quá trình truyền tải chỉ có thể được giải mã bởi những đối tượng được cấp quyền, qua đó ngăn chặn nguy cơ nghe lén hoặc đánh cắp dữ liệu.
- **Mục tiêu 2:** Thiết kế và triển khai phương thức **xác thực người dùng**, giúp hệ thống nhận diện và phân biệt người dùng hợp lệ với kẻ giả mạo, từ đó bảo đảm rằng chỉ những cá nhân hoặc thiết bị có quyền mới được phép truy cập và trao đổi dữ liệu.
- **Mục tiêu 3:** Đưa ra giải pháp **kiểm tra tính toàn vẹn dữ liệu**, bảo đảm rằng thông tin sau khi truyền tải không bị thay đổi, giả mạo hoặc hư hỏng. Điều này được thực hiện thông qua các thuật toán băm mật mã và chữ ký số.
- **Mục tiêu 4:** Xây dựng một **mô hình ứng dụng thử nghiệm** cho phép truyền tải dữ liệu âm thanh có kiểm tra bảo mật, từ đó minh chứng cho tính khả thi của hệ thống cũng như tạo nền tảng cho việc mở rộng nghiên cứu trong tương lai.

Với những mục tiêu này, đề tài hướng tới việc không chỉ giải quyết những thách thức đặt ra

trong quá trình truyền tải dữ liệu mà còn góp phần đề xuất một mô hình có thể áp dụng rộng rãi trong nhiều lĩnh vực khác nhau, từ giám sát công nghiệp cho đến các ứng dụng dịch vụ số trong đời sống hàng ngày.

1.3 Cấu trúc của báo cáo

Báo cáo gồm các phần như sau:

- Chương 1: Giới thiệu.
- Chương 2: Phân tích yêu cầu và thiết kế ứng dụng
- Chương 3: Kết luận và hướng phát triển

Chương 2

Phân tích yêu cầu và thiết kế ứng dụng

2.1 Mô tả thuật toán

Thuật toán giám sát an toàn tại nhà máy dựa trên việc **nhận dạng hình ảnh bằng học sâu (Deep Learning)** kết hợp với xử lý video thời gian thực. Hệ thống có nhiệm vụ phát hiện nhân viên không đội mũ bảo hộ khi làm việc trong khu vực sản xuất. Dưới đây là mô tả chi tiết quy trình thuật toán:

2.1.1 Các bước xử lý chính

1. **Thu nhận dữ liệu từ camera:** Ứng dụng sử dụng thư viện OpenCV để dò tìm và mở camera khả dụng trên hệ thống. Camera được cấu hình với độ phân giải 640×480 pixel và tốc độ 30 FPS nhằm cân bằng giữa độ chính xác và hiệu suất.

2. **Tiền xử lý khung hình:** Mỗi khung hình được chuyển sang dạng ma trận ảnh số. Nếu cần, hệ thống thực hiện resize để phù hợp với kích thước đầu vào của mô hình YOLOv8.

3. **Phát hiện đối tượng bằng YOLOv8:**

   - Chia ảnh đầu vào thành các lưới (grid).
   - Dự đoán bounding box và xác suất đối tượng trong từng ô.
   - Áp dụng thuật toán *Non-Maximum Suppression (NMS)* để loại bỏ các khung chồng chéo.
   - Trả về danh sách đối tượng gồm: tọa độ bounding box (x_1, y_1, x_2, y_2), độ tin cậy (confidence), và nhãn lớp ("person", "helmet").

4. **Xác định hành vi vi phạm:** Với mỗi đối tượng "person" được phát hiện:

- Nếu **không có lớp "helmet"** tại vùng đầu người → đánh dấu là vi phạm.
- Nếu có mũ bảo hộ → đánh dấu an toàn.

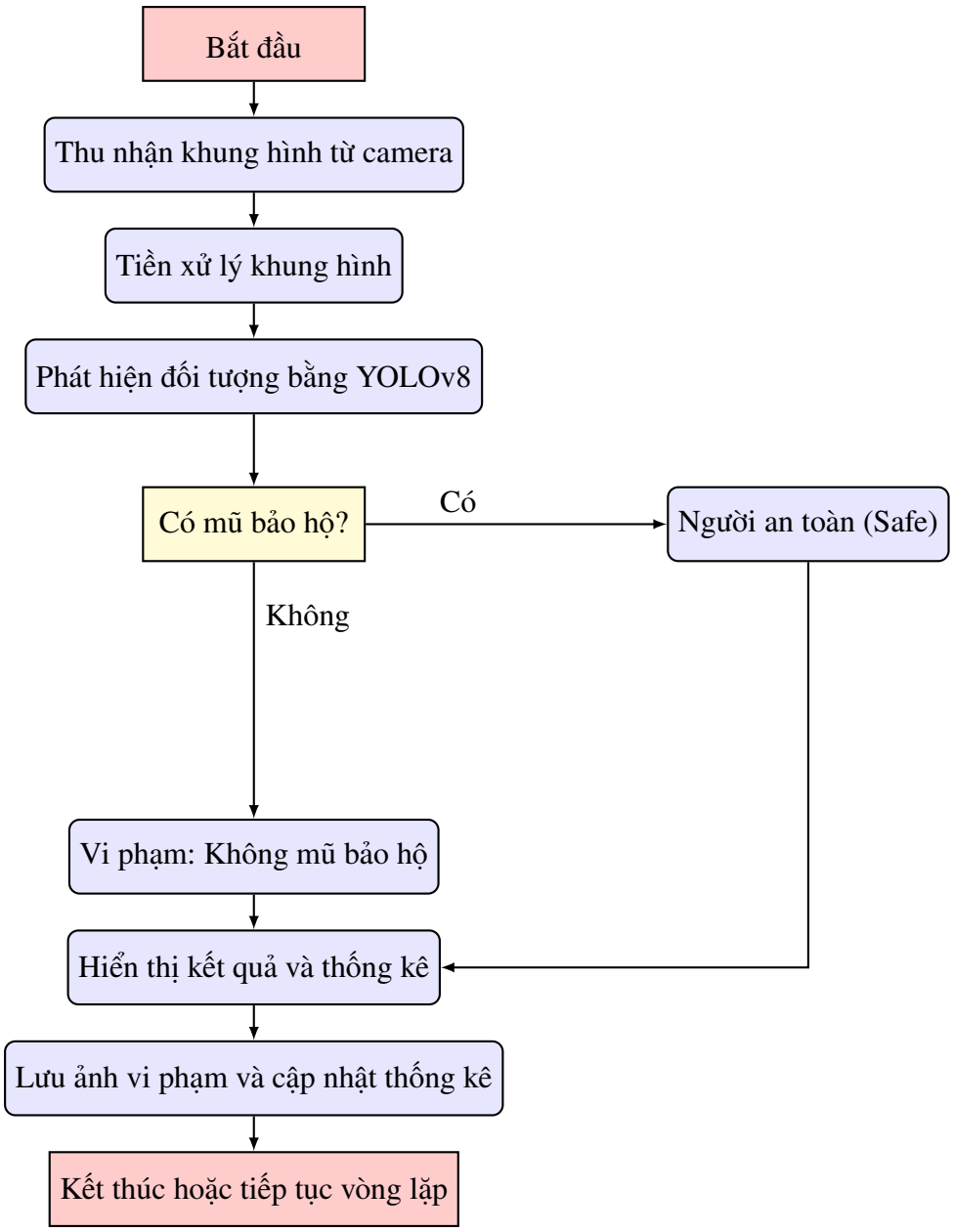
5. **Hiển thị kết quả trên video:**
   - Người an toàn: khung màu xanh, hiển thị nhãn *Safe*.
   - Người vi phạm: khung màu đỏ, hiển thị nhãn *NO HELMET*.
   - Góc màn hình có thống kê: tổng số phát hiện, số vi phạm, tỉ lệ phần trăm vi phạm.
6. **Lưu trữ và cảnh báo:** Khi phát hiện vi phạm, hệ thống:
   - Lưu ảnh bằng chứng vào thư mục `violations/` kèm timestamp.
   - Cập nhật thống kê vi phạm.
   - Có thể mở rộng để gửi cảnh báo (âm thanh, email, hệ thống SCADA...).

2.1.2 Bảng mô tả chi tiết chức năng

Bước	Mô tả	Công cụ / Thuật toán
1	Thu nhận khung hình từ camera	OpenCV
2	Tiền xử lý ảnh	Chuyển ma trận ảnh, resize, chuẩn hóa
3	Phát hiện đối tượng	YOLOv8, Non-Maximum Suppression
4	Xác định vi phạm	So sánh "person" với "helmet"
5	Hiển thị kết quả	OpenCV: vẽ bounding box, thống kê
6	Lưu trữ và cảnh báo	Ghi file ảnh vi phạm, cập nhật thống kê

Bảng 2.1: Các bước chính của thuật toán giám sát an toàn

2.1.3 Lưu đồ thuật toán



Hình 2.1: Lưu đồ thuật toán giám sát an toàn

2.2 Phân tích mã nguồn

Trong phần này, chúng ta sẽ đi sâu phân tích mã nguồn của hệ thống giám sát an toàn nhằm làm rõ cấu trúc chương trình, cơ chế hoạt động của từng thành phần và mối quan hệ giữa chúng. Hệ thống được xây dựng bằng ngôn ngữ **Python**, sử dụng mô hình học sâu YOLOv8 để nhận dạng đối tượng trong ảnh và thư viện **Flask** để xây dựng giao diện web phục vụ giám sát. Toàn bộ mã nguồn được chia thành hai tệp chính: `app.py` và `testcamera.py`. Trong đó, `app.py` giữ vai trò trung tâm, còn `testcamera.py` hỗ trợ kiểm tra và chẩn đoán tình trạng camera.

2.2.1 Cấu trúc chương trình

File `app.py` triển khai toàn bộ ứng dụng giám sát an toàn. Tệp này khởi tạo một ứng dụng Flask, định nghĩa các đường dẫn (endpoint) để người dùng truy cập qua trình duyệt, đồng thời quản lý quá trình kết nối camera, xử lý từng khung hình và phát hiện vi phạm. Ngoài ra, `app.py` còn quản lý việc lưu trữ các khung hình vi phạm vào thư mục riêng (`violations/`) cũng như xử lý các tệp ảnh được tải lên bởi người dùng thông qua giao diện web (`static/uploads/`).

Bên cạnh đó, file `testcamera.py` có chức năng kiểm tra sự hoạt động của camera trên hệ thống. Khi chạy chương trình này, hệ thống sẽ tự động dò tìm các ID camera khả dụng, thử nhiều backend khác nhau (DirectShow, Video4Linux2, AVFoundation...) để đảm bảo tính tương thích trên nhiều hệ điều hành. Kết quả trả về gồm danh sách camera hoạt động, độ phân giải hỗ trợ, đồng thời hiển thị một đoạn video ngắn để xác nhận camera có thể sử dụng. File này đóng vai trò hỗ trợ quan trọng, giúp người triển khai hệ thống nhanh chóng phát hiện các lỗi cấu hình hoặc quyền truy cập camera.

2.2.2 Phân tích lớp SafetyDetectionApp

Trái tim của hệ thống chính là lớp `SafetyDetectionApp`, được định nghĩa trong file `app.py`. Lớp này đóng vai trò là **bộ điều phối trung tâm** (central controller), gói gọn toàn bộ logic giám sát, quản lý kết nối camera, xử lý hình ảnh thông qua mô hình YOLOv8 và cung cấp kết quả cho giao diện Flask. Nhờ thiết kế hướng đối tượng, toàn bộ quá trình giám sát được tổ chức thành các phương thức rõ ràng, dễ mở rộng và bảo trì.

Cấu trúc và thuộc tính chính

Lớp `SafetyDetectionApp` được khởi tạo với các thuộc tính quan trọng sau:

- `model`: đối tượng YOLOv8 được nạp sẵn khi ứng dụng khởi động. Đây là thành phần cốt lõi, chịu trách nhiệm phân tích từng khung hình và phát hiện các đối tượng như người, mũ bảo hộ, hoặc tình trạng không đội mũ.
- `camera`: đối tượng kết nối tới thiết bị ghi hình. Hệ thống hỗ trợ nhiều loại camera, từ webcam tích hợp đến camera IP. Việc khởi tạo thành công camera là điều kiện tiên quyết để bắt đầu giám sát.
- `is_detecting`: biến trạng thái (flag) cho biết hệ thống có đang thực hiện phát hiện theo thời gian thực hay không. Điều này cho phép người dùng dễ dàng bật/tắt giám sát chỉ bằng một thao tác trên giao diện.
- `violation_count` và `total_detections`: các bộ đếm thống kê, lần lượt ghi nhận số vi phạm phát hiện được và tổng số lần phân tích khung hình. Từ hai biến này, hệ thống có thể

tính toán tỷ lệ vi phạm và hiển thị trực quan trên giao diện web.

- `detection_results`: danh sách lưu trữ lịch sử các phát hiện. Mỗi phần tử trong danh sách chứa thông tin về thời gian, loại phát hiện, số lượng vi phạm, độ tin cậy trung bình, và đường dẫn đến ảnh minh chứng. Đây là cơ sở cho module thống kê và báo cáo.

Cấu trúc thuộc tính này cho thấy hệ thống được thiết kế vừa đảm bảo **xử lý thời gian thực**, vừa có khả năng **lưu vết lịch sử**, giúp cho công tác giám sát an toàn lao động minh bạch và có thể truy cứu khi cần.

Các nhóm phương thức chính

Các phương thức của lớp được tổ chức thành ba nhóm chức năng:

(1) Quản lý camera. Hai phương thức quan trọng trong nhóm này là `find_working_camera()` và `open_camera()`. Phương thức đầu tiên duyệt qua danh sách các chỉ số thiết bị (device index) để tìm camera hoạt động được, đảm bảo hệ thống có thể tự động thích ứng nếu thay đổi phần cứng. Phương thức `open_camera()` chịu trách nhiệm mở luồng video và kiểm tra trạng thái. Thiết kế này giúp hệ thống trở nên **linh hoạt và bền vững** trong nhiều môi trường triển khai khác nhau.

(2) Xử lý và phân tích hình ảnh. Đây là nhóm phương thức quan trọng nhất, bao gồm:

- `detect_in_frame(frame)`: gọi mô hình YOLO để phân tích từng khung hình, trả về danh sách các đối tượng phát hiện, vị trí và xác suất.
- `draw_detections(frame, detections)`: vẽ các khung chữ nhật (bounding boxes) lên ảnh gốc, gắn nhãn "Safe" hoặc "No Helmet" cùng độ tin cậy. Đây là bước **trực quan hóa** giúp người vận hành dễ dàng theo dõi.
- `fake_detection()`: phương thức giả lập phát hiện, dùng trong chế độ demo khi không có mô hình hoặc camera. Điều này đặc biệt hữu ích cho mục đích trình bày và thử nghiệm nhanh.

(3) Cập nhật kết quả và lưu trữ. Sau mỗi lần phát hiện, hệ thống kiểm tra xem có vi phạm hay không. Nếu có, ảnh được lưu vào thư mục `violations/` và thông tin chi tiết được thêm vào danh sách `detection_results`. Các phương thức thống kê sẽ truy xuất dữ liệu này để hiển thị số liệu tổng quan và danh sách chi tiết trên giao diện.

Thuật toán phát hiện và xử lý

Quá trình phát hiện có thể mô tả bằng giải thuật sau:

Giải thuật 1: Giải thuật phát hiện vi phạm an toàn

Input: Luồng video từ camera hoặc ảnh tải lên

Output: Kết quả phát hiện: Safe / No Helmet, lưu ảnh vi phạm nếu có

while *is_detecting* = *True* **do**
  frame ← đọc khung hình từ camera results ← YOLOv8.predict(frame) **foreach** *object trong results* **do**
    if *object* = *person* ***và*** *không có helmet* **then**
      Violation ← True violation_count++ Lưu frame vào thư mục `violations/`
    else
      Violation ← False
    end
    Vẽ bounding box và nhãn lên frame
  end
  total_detections++ Cập nhật danh sách detection_results Trả về frame đã được xử lý để hiển thị
end

Đoạn code minh họa

```python
class SafetyDetectionApp:
    def __init__(self, model_path="yolov8n.pt"):
        self.model = YOLO(model_path)
        self.camera = None
        self.is_detecting = False
        self.violation_count = 0
        self.total_detections = 0
        self.detection_results = []

    def open_camera(self, index=0):
        self.camera = cv2.VideoCapture(index)
        return self.camera.isOpened()

    def detect_in_frame(self, frame):
        results = self.model(frame)
        detections = []
        for r in results:
            for box in r.boxes:
                cls = int(box.cls[0])
                conf = float(box.conf[0])
                detections.append((cls, conf, box.xyxy[0]))
        return detections

    def draw_detections(self, frame, detections):
```

```
        for cls, conf, box in detections:
            x1, y1, x2, y2 = map(int, box)
            label = "Safe" if cls == 1 else "No Helmet"
            color = (0,255,0) if cls == 1 else (0,0,255)
            cv2.rectangle(frame, (x1,y1), (x2,y2), color, 2)
            cv2.putText(frame, f"{label}: {conf:.2f}",
                        (x1,y1-10), cv2.FONT_HERSHEY_SIMPLEX,
                        0.6, color, 2)
        return frame
```

Listing 2.1: Trích đoạn lớp SafetyDetectionApp trong file app.py

Nhận xét tổng quan

Có thể thấy lớp `SafetyDetectionApp` đã được thiết kế với tư tưởng hướng đối tượng rõ ràng: phân tách nhiệm vụ, sử dụng các biến trạng thái để quản lý hoạt động, đồng thời gói gọn toàn bộ logic phát hiện vào trong một cấu trúc thống nhất. Điều này giúp mã nguồn **dễ đọc, dễ bảo trì và có khả năng mở rộng**. Trong tương lai, lớp này hoàn toàn có thể được mở rộng thêm các chức năng như gửi cảnh báo qua email, tích hợp hệ thống IoT, hoặc bổ sung nhiều lớp nhận dạng bảo hộ khác mà không phá vỡ kiến trúc hiện tại.

2.2.3 Các API của Flask

Để phục vụ người dùng cuối, hệ thống xây dựng một loạt endpoint bằng Flask. Các endpoint này đóng vai trò như cầu nối giữa lớp xử lý nền (SafetyDetectionApp) và giao diện web. Người dùng có thể khởi động hoặc dừng camera, theo dõi luồng video trực tiếp, tải ảnh lên để phân tích, hay xem thống kê an toàn lao động theo thời gian thực.

Cụ thể, `/start_camera` và `/stop_camera` quản lý trạng thái camera; `/video_feed` cung cấp stream video đã qua xử lý, trong đó mỗi khung hình được mã hóa thành JPEG và truyền tới trình duyệt theo chuẩn MJPEG. Endpoint `/stats` và `/reset_stats` cho phép truy xuất và làm mới thống kê. Đặc biệt, `/upload` và `/capture_frame` cho phép phân tích ảnh tĩnh, đáp ứng tình huống người quản lý muốn kiểm tra một bức ảnh cụ thể thay vì video trực tiếp. Nhờ vào cơ chế RESTful API, các chức năng này có thể dễ dàng được mở rộng sang ứng dụng di động hoặc hệ thống quản lý sản xuất thông minh.

Luồng hoạt động của API Flask có thể mô tả bằng giải thuật sau:

Giải thuật 2: Giải thuật tổng quát cho các API Flask

Input: Yêu cầu HTTP từ client (trình duyệt web hoặc ứng dụng khác)

Output: Kết quả trả về (HTML, JSON, hoặc MJPEG stream)

switch *endpoint* **do**
- **case** */start_camera* **do**
  - Gọi `open_camera()` trong SafetyDetectionApp Đặt `is_detecting = True` Trả về thông báo JSON: "Camera started"
- **end**
- **case** */stop_camera* **do**
  - Giải phóng tài nguyên camera Đặt `is_detecting = False` Trả về thông báo JSON: "Camera stopped"
- **end**
- **case** */video_feed* **do**
  - **while** *is_detecting* **do**
    - Đọc frame từ camera Gọi YOLOv8.detect(frame) Vẽ bounding box và nhãn lên frame Mã hoá frame thành JPEG Truyền về client theo chuẩn MJPEG
  - **end**
- **end**
- **case** */upload* **do**
  - Nhận file ảnh từ POST request Phân tích ảnh bằng YOLOv8 Nếu có vi phạm → lưu ảnh vào `violations/` Trả về JSON chứa số lượng vi phạm và độ tin cậy
- **end**
- **case** */stats* **do**
  - Trả về JSON chứa `violation_count`, `total_detections`, `tỉ lệ vi phạm`
- **end**

end

Đoạn code Flask minh họa

```python
from flask import Flask, Response, request, jsonify, render_template
app = Flask(__name__)
safety_app = SafetyDetectionApp("yolov8n.pt")

@app.route('/start_camera')
def start_camera():
    if safety_app.open_camera():
        safety_app.is_detecting = True
        return jsonify({"status": "Camera started"})
    return jsonify({"status": "Failed to open camera"})

@app.route('/stop_camera')
def stop_camera():
```

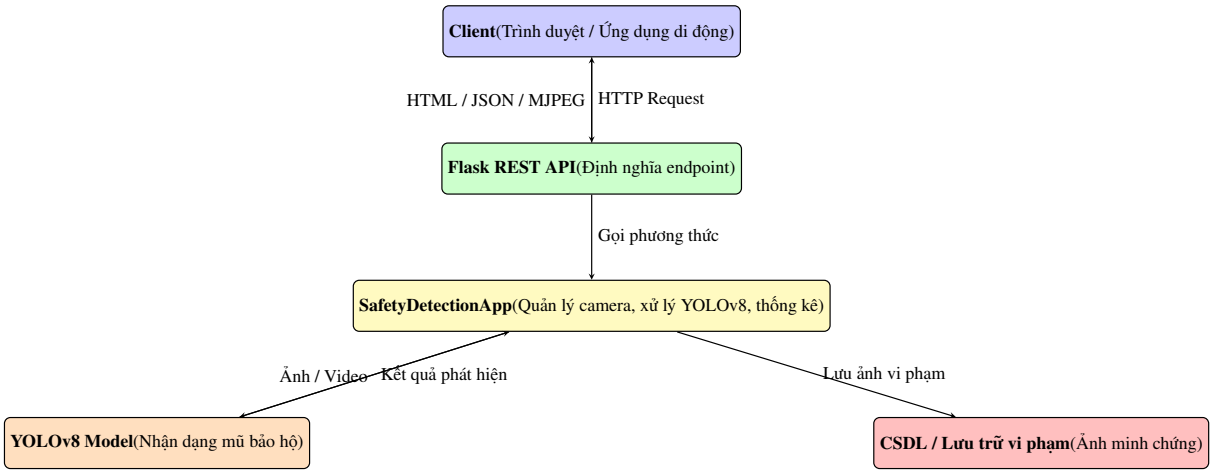
```
    safety_app.is_detecting = False
    if safety_app.camera:
        safety_app.camera.release()
    return jsonify({"status": "Camera stopped"})

@app.route('/video_feed')
def video_feed():
    def generate():
        while safety_app.is_detecting:
            ret, frame = safety_app.camera.read()
            if not ret: break
            detections = safety_app.detect_in_frame(frame)
            frame = safety_app.draw_detections(frame, detections)
            _, buffer = cv2.imencode('.jpg', frame)
            yield (b'--frame\r\n'
                   b'Content-Type: image/jpeg\r\n\r\n' + buffer.tobytes
                      () + b'\r\n')
    return Response(generate(),
                    mimetype='multipart/x-mixed-replace; boundary=frame'
                       )

@app.route('/upload', methods=['POST'])
def upload():
    if 'file' not in request.files:
        return jsonify({"error": "No file uploaded"})
    file = request.files['file']
    frame = cv2.imdecode(np.frombuffer(file.read(), np.uint8), cv2.
       IMREAD_COLOR)
    detections = safety_app.detect_in_frame(frame)
    return jsonify({"detections": len(detections)})
```

Listing 2.2: Một số endpoint Flask trong file app.py

Sơ đồ kiến trúc API REST



Hình 2.2: Sơ đồ kiến trúc REST API theo hướng từ trên xuống

2.3 Thử nghiệm

Để đánh giá tính khả thi và hiệu quả của hệ thống, nhóm tiến hành triển khai thử nghiệm trực tiếp trên môi trường thực tế. Hệ thống được cài đặt trên máy tính cá nhân sử dụng hệ điều hành Windows, tích hợp camera laptop độ phân giải HD, với thư viện OpenCV phiên bản 4.12 và mô hình YOLOv8n. Giao diện web được cung cấp thông qua Flask, cho phép truy cập qua địa chỉ `http://localhost:5000`.

2.3.1 Môi trường thử nghiệm

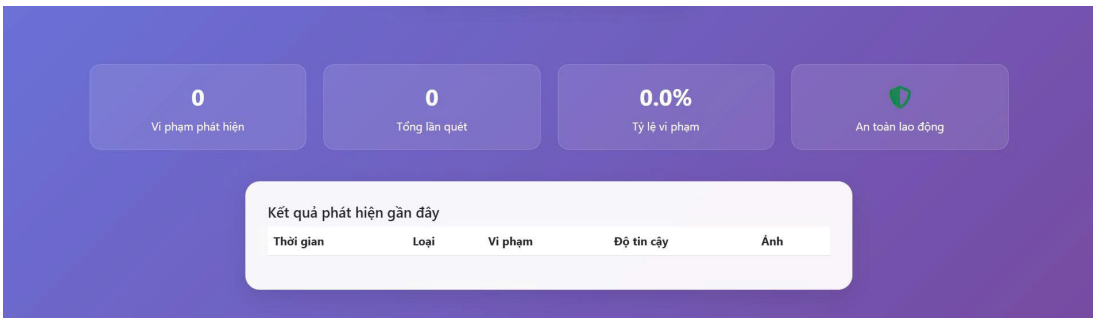
- **Phần cứng:** Máy tính xách tay Core i5, RAM 8GB, camera tích hợp độ phân giải 720p.
- **Phần mềm:** Python 3.10, thư viện OpenCV 4.12, Flask 2.2, Ultralytics YOLOv8.
- **Hệ điều hành:** Windows 10 (64-bit).
- **Điều kiện thử nghiệm:** Phòng học/nhà máy có ánh sáng tự nhiên, một số trường hợp ánh sáng huỳnh quang.

Các kịch bản thử nghiệm bao gồm: (1) kiểm tra giao diện ban đầu và kết nối camera, (2) phát hiện đối tượng an toàn (có đội mũ bảo hộ), (3) phát hiện vi phạm (không đội mũ bảo hộ), (4) upload ảnh để phân tích ngoại tuyến, (5) kiểm tra thống kê và lưu trữ vi phạm.

2.3.2 Kết quả thử nghiệm giao diện hệ thống

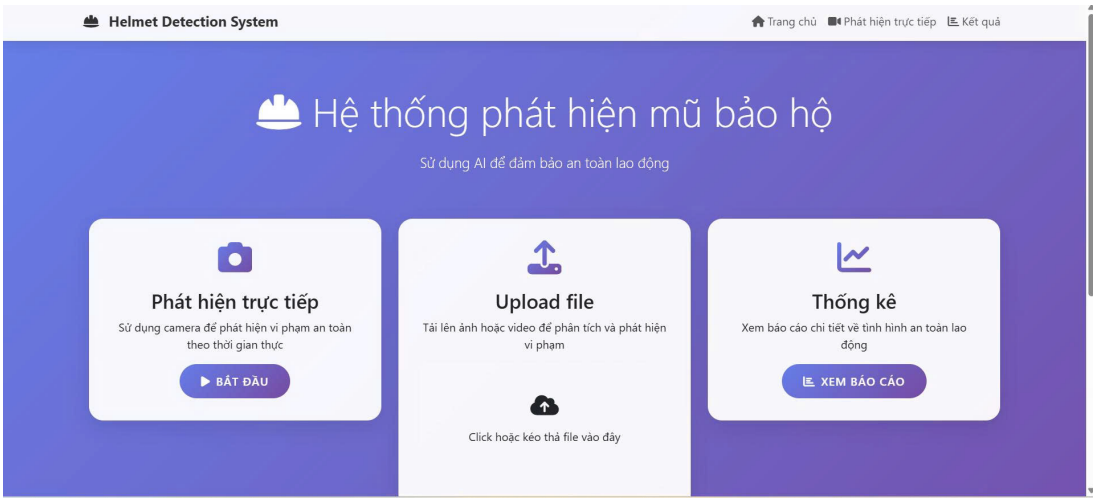
Ngay khi khởi động ứng dụng, trang chủ hiển thị ba chức năng chính: *Phát hiện trực tiếp*, *Upload file*, và *Thống kê*. Giao diện có thiết kế trực quan, với tông màu xanh tím dịu mắt, biểu

tượng dễ hiểu. Người dùng chỉ cần nhấn nút *Bắt đầu* để kích hoạt camera hoặc kéo-thả ảnh để phân tích.



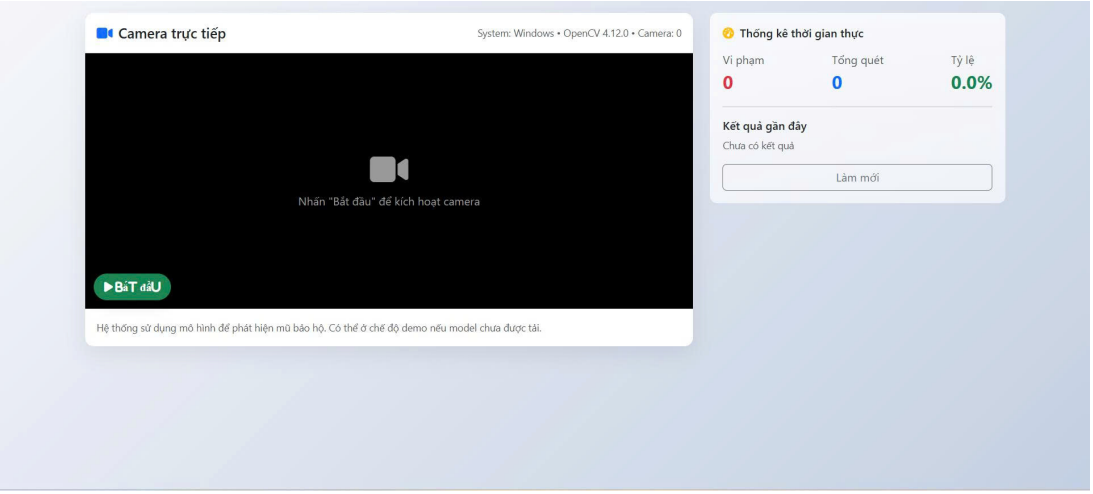
Hình 2.3: Trang chủ của hệ thống phát hiện mũ bảo hộ

Khi nhấn *Bắt đầu*, hệ thống tiến hành kết nối với camera và bắt đầu truyền video trực tiếp lên giao diện web. Ở trạng thái ban đầu khi chưa có đối tượng trong khung hình, thống kê hiển thị số vi phạm bằng 0, tổng số lần quét bằng 0 và tỷ lệ vi phạm 0%. Điều này thể hiện hệ thống khởi tạo thành công và sẵn sàng hoạt động.



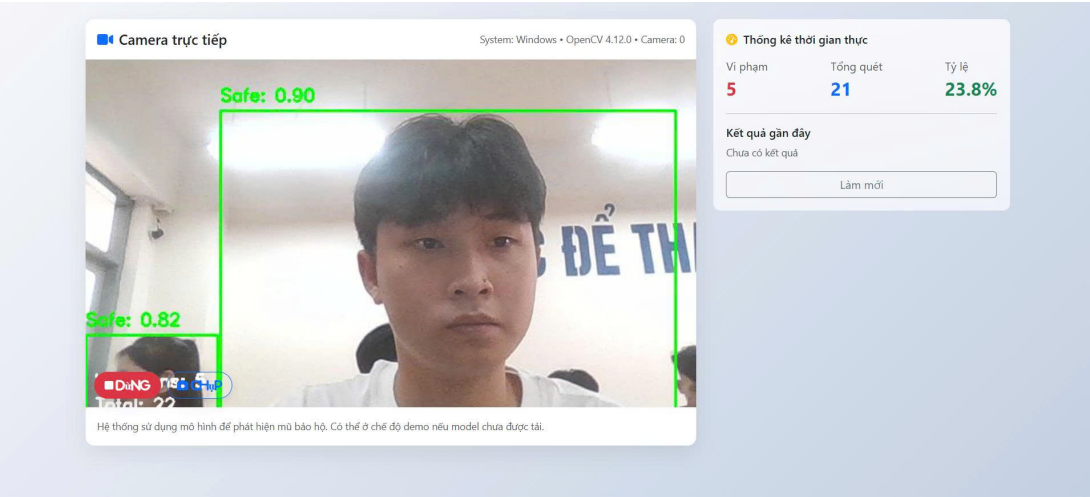
Hình 2.4: Giao diện phát hiện trực tiếp khi camera chưa ghi nhận đối tượng

Khi có người xuất hiện trong khung hình nhưng không đội mũ bảo hộ, hệ thống sử dụng YOLOv8 để phát hiện đối tượng "person", sau đó kiểm tra sự hiện diện của lớp "helmet". Nếu không phát hiện mũ, khung hình sẽ được viền đỏ với nhãn "NO HELMET" và vi phạm được cộng vào thống kê. Ngược lại, nếu phát hiện đối tượng an toàn, khung được hiển thị màu xanh kèm nhãn "Safe" và độ tin cậy.



Hình 2.5: Camera trực tiếp khi chưa có đối tượng vi phạm (tất cả đều Safe)

Khi thử nghiệm nhiều tình huống khác nhau, hệ thống đã phân biệt rõ ràng giữa người có mũ bảo hộ và người không có mũ. Các vi phạm được lưu lại dưới dạng hình ảnh trong thư mục `violations/`, kèm timestamp để dễ dàng tra cứu sau này.



Hình 2.6: Ví dụ hệ thống phát hiện vi phạm: 5/21 lần quét (23.8%) không đội mũ bảo hộ

2.3.3 Thử nghiệm chức năng Upload file

Bên cạnh phát hiện trực tiếp, hệ thống còn cho phép người dùng tải lên ảnh hoặc video để phân tích. Khi một ảnh được upload, hệ thống đọc file, giải mã bằng OpenCV và đưa vào mô hình YOLOv8. Kết quả phân tích được hiển thị ngay trên giao diện web, đồng thời ảnh kết quả được lưu lại trong thư mục vi phạm nếu phát hiện trường hợp không đội mũ bảo hộ. Chức năng này đặc biệt hữu ích trong tình huống người quản lý muốn kiểm tra nhanh một bức ảnh hiện trường thay vì phải dùng camera trực tiếp.

2.3.4 Thống kê và báo cáo

Hệ thống có module thống kê theo thời gian thực, hiển thị số vi phạm đã phát hiện, tổng số lần quét và tỷ lệ phần trăm vi phạm. Thông tin này được hiển thị trực quan trên giao diện bằng số liệu và biểu tượng. Ngoài ra, bảng "Kết quả phát hiện gần đây" ghi lại thời gian, loại phát hiện (trực tiếp hoặc upload), số vi phạm và độ tin cậy của mô hình. Tính năng này giúp nhà quản lý dễ dàng theo dõi tình hình an toàn lao động trong ca sản xuất.

2.3.5 Đánh giá kết quả thử nghiệm

Qua nhiều lần thử nghiệm, hệ thống cho thấy:

- **Tính ổn định:** Camera kết nối ổn định, tốc độ xử lý trung bình đạt 15–20 FPS, đủ để giám sát thời gian thực.
- **Độ chính xác:** Trong điều kiện ánh sáng tốt, hệ thống nhận dạng đúng đa số trường hợp. Một số sai số xảy ra khi đối tượng di chuyển nhanh hoặc bị che khuất.
- **Tính tiện dụng:** Giao diện web trực quan, dễ sử dụng, phù hợp cả cho người dùng không có chuyên môn CNTT.
- **Khả năng mở rộng:** Hệ thống có thể tích hợp thêm nhiều lớp nhận dạng khác (áo bảo hộ, găng tay, kính bảo hộ) và kết nối với cơ sở dữ liệu để lưu trữ lâu dài.

2.3.6 Kết luận thử nghiệm

Các thử nghiệm cho thấy hệ thống hoạt động đúng như kỳ vọng: phát hiện chính xác người lao động có hoặc không đội mũ bảo hộ, hiển thị kết quả ngay trên giao diện, đồng thời lưu trữ bằng chứng vi phạm. Mặc dù còn một số hạn chế nhỏ trong điều kiện ánh sáng yếu hoặc nhiều người đứng gần nhau, hệ thống đã chứng minh được tính ứng dụng cao trong việc giám sát an toàn lao động tại nhà máy, phân xưởng hoặc công trường xây dựng.

Chương 3

Kết luận và hướng phát triển

Sau quá trình nghiên cứu, thiết kế và triển khai hệ thống giám sát an toàn lao động bằng camera nhận dạng mũ bảo hộ, đề tài đã chứng minh được tính khả thi cũng như hiệu quả trong việc áp dụng công nghệ học sâu vào thực tiễn sản xuất. Hệ thống không chỉ hoạt động ổn định trong môi trường thử nghiệm mà còn cung cấp giao diện trực quan, dễ sử dụng và có khả năng mở rộng trong tương lai. Dưới đây là phân tích chi tiết về hiệu quả của hệ thống, đặc điểm của thuật toán sử dụng và một số đề xuất cải tiến.

3.1 Phân tích hiệu quả

Hệ thống đã đáp ứng được các mục tiêu đề ra ban đầu, bao gồm:

- Phát hiện tự động và theo thời gian thực các trường hợp người lao động không đội mũ bảo hộ.
- Hiển thị kết quả phát hiện trực tiếp trên giao diện web, đồng thời lưu trữ bằng chứng hình ảnh cho việc kiểm tra và báo cáo.
- Cung cấp thống kê về số vi phạm, tổng số lần quét, tỷ lệ phần trăm vi phạm và danh sách chi tiết các lần phát hiện gần đây.

Thử nghiệm thực tế cho thấy hệ thống hoạt động với tốc độ trung bình từ 15–20 FPS, đảm bảo yêu cầu giám sát thời gian thực. Độ chính xác trong điều kiện ánh sáng bình thường đạt mức khả quan, hầu hết các trường hợp vi phạm đều được phát hiện. Giao diện thân thiện giúp người dùng thao tác dễ dàng mà không cần nhiều kiến thức chuyên sâu về công nghệ. Nhìn chung, hệ thống đã chứng minh được khả năng ứng dụng trong môi trường thực tế như nhà máy hoặc công trường xây dựng.

3.1.1 Phân tích và nhận xét đặc điểm của các thuật toán sử dụng

Thuật toán chính được sử dụng trong hệ thống là mô hình **YOLOv8** (You Only Look Once, phiên bản 8). Đây là một trong những mô hình phát hiện đối tượng hiện đại, nổi bật bởi sự cân bằng giữa tốc độ và độ chính xác. YOLOv8 hoạt động theo cơ chế chia ảnh thành các lưới, sau đó dự đoán đồng thời vị trí (bounding box) và xác suất thuộc lớp của đối tượng trong từng ô. Ưu điểm của cách tiếp cận này là tốc độ xử lý rất nhanh, phù hợp cho các ứng dụng thời gian thực như giám sát an toàn.

So với các thuật toán phát hiện truyền thống (Haar Cascade, HOG + SVM), YOLO có ưu điểm vượt trội về khả năng nhận dạng nhiều loại đối tượng khác nhau trong cùng một khung hình, đồng thời xử lý tốt các tình huống có nhiều đối tượng chồng chéo. So với các mô hình hiện đại khác như Faster R-CNN hay SSD, YOLOv8 có ưu thế về tốc độ, trong khi độ chính xác ở mức tương đương, đủ để đáp ứng nhu cầu phát hiện vi phạm an toàn lao động.

Tuy nhiên, YOLOv8 cũng có những hạn chế nhất định. Thứ nhất, độ chính xác phụ thuộc mạnh vào dữ liệu huấn luyện. Nếu bộ dữ liệu chưa đủ đa dạng (ví dụ các góc chụp, điều kiện ánh sáng, loại mũ bảo hộ khác nhau), mô hình có thể bỏ sót hoặc nhầm lẫn. Thứ hai, trong điều kiện ánh sáng yếu hoặc đối tượng di chuyển quá nhanh, kết quả phát hiện có thể không ổn định. Thứ ba, mô hình cần tài nguyên phần cứng nhất định, do đó khi triển khai trên các thiết bị nhúng hoặc camera thông minh cần tối ưu thêm.

3.1.2 Đề xuất cải tiến

Mặc dù hệ thống hiện tại đã đạt được kết quả khả quan, vẫn còn nhiều hướng phát triển có thể nâng cao tính hiệu quả và giá trị ứng dụng trong thực tế:

- **Cải thiện mô hình học sâu:** Huấn luyện lại YOLOv8 với bộ dữ liệu chuyên biệt cho môi trường công nghiệp Việt Nam, bao gồm nhiều loại mũ bảo hộ, màu sắc, và góc nhìn khác nhau. Ngoài ra, có thể kết hợp các kỹ thuật tăng cường dữ liệu (data augmentation) để cải thiện độ bền vững của mô hình.

- **Mở rộng đối tượng nhận dạng:** Hiện tại hệ thống mới tập trung vào phát hiện mũ bảo hộ. Trong tương lai có thể bổ sung thêm các thiết bị bảo hộ khác như áo phản quang, găng tay, kính bảo hộ, khẩu trang... nhằm xây dựng một hệ thống giám sát an toàn toàn diện.

- **Tối ưu hiệu năng:** Triển khai các phiên bản nhẹ hơn của YOLO (YOLOv8n, YOLOv8-tiny) hoặc sử dụng TensorRT, ONNX để tăng tốc xử lý trên GPU/TPU. Điều này giúp hệ thống đạt tốc độ cao hơn ngay cả khi chạy trên phần cứng hạn chế.

- **Tích hợp cơ sở dữ liệu:** Kết hợp hệ thống với một cơ sở dữ liệu tập trung để lưu trữ lâu dài các kết quả phát hiện, từ đó tạo điều kiện cho việc phân tích xu hướng vi phạm, thống

kê theo thời gian, theo ca làm việc hoặc theo khu vực sản xuất.

- **Tích hợp hệ thống cảnh báo:** Phát triển module cảnh báo tức thời thông qua còi báo động, gửi thông báo đến điện thoại của quản lý hoặc tích hợp với các hệ thống SCADA trong nhà máy.
- **Ứng dụng trên thiết bị IoT:** Triển khai hệ thống trên các thiết bị nhúng như Jetson Nano, Raspberry Pi 4 hoặc camera AI để giám sát trực tiếp tại hiện trường mà không cần máy tính trung gian, từ đó tăng tính linh hoạt và giảm chi phí.

Kết luận

Qua quá trình triển khai và thử nghiệm, hệ thống giám sát an toàn bằng camera phát hiện mũ bảo hộ đã chứng minh được hiệu quả và tính khả thi. Đây là một minh chứng rõ ràng cho việc áp dụng công nghệ học sâu vào lĩnh vực an toàn lao động, góp phần giảm thiểu tai nạn và nâng cao ý thức chấp hành quy định bảo hộ của công nhân. Với các cải tiến và mở rộng trong tương lai, hệ thống có thể trở thành một công cụ hữu ích, không chỉ trong các nhà máy mà còn ở công trường xây dựng, kho bãi, và các môi trường lao động khác đòi hỏi tuân thủ nghiêm ngặt quy định an toàn.

Tài liệu tham khảo

[1] Gary Bradski et al. Opencv library, 2000. Available at https://opencv.org/.

[2] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

[3] Piotr Dollár, Zhuowen Tu, Pietro Perona, and Serge Belongie. Integral channel features. 2009.

[4] Martín Abadi et al. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[5] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part based models, 2010.

[6] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, 2013.

[7] Ian J. Goodfellow, Yaroslav Bulatov, Julian Ibarz, Sacha Arnoud, and Vinay Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. In *In ICLR, 2014b*.

[8] Chunhui Gu, Joseph J. Lim, Pablo Arbeláez, and Jitendra Malik. Recognition using regions.

[9] Hugo Krawczyk, Mihir Bellare, and Ran Canetti. Hmac: Keyed-hashing for message authentication. *RFC 2104, Internet Engineering Task Force (IETF)*, 1997.

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012.

[11] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. *CoRR*, 2015.

[12] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. Handbook of applied cryptography. *CRC Press*, 1996.

[13] Pallets Projects. Flask documentation, 2023. Available at https://flask.palletsprojects.com/.

[14] Florent Perronnin, Jorge Sánchez, and Thomas Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*, 2010.

[15] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, 2015.

[16] Shaoquing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with regional proposal networks. *CoRR*, 2015.

[17] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. In *Communications of the ACM*, volume 21, pages 120–126. ACM, 1978.

[18] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, 2014.

[19] William Stallings. *Cryptography and Network Security: Principles and Practice*. Pearson, 7th edition, 2017.

[20] Ultralytics. Yolov8: Real-time object detection, 2023. Available at https://github.com/ultralytics/ultralytics.

[21] Timothy Van Zandt. Documentation for fancybox.sty: Box tips and tricks for , 2010.

[22] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *In Computer Vision - ECCV 2014*, 2014.