BỘ NÔNG NGHIỆP VÀ PHÁT TRIỂN NÔNG THÔN

**TRƯỜNG ĐẠI HỌC LÂM NGHIỆP**



**BÁO CÁO MÔN KHAI PHÁ DỮ LIỆU**

**ĐỀ TÀI: NHẬN DIỆN HÌNH ẢNH**

|  |  |
| --- | --- |
| ***Giảng viên***  ***Sinh viên thực hiện*** | **: MAI HÀ AN**  **: Nguyễn Kim Tấn**  **: Bùi Thanh Minh** |

**Hà Nội, tháng 2 năm 2025**

MUC LUC

[I.Giới thiệu đề tài 3](#_Toc11771)

[1. Lý do chọn đề tài 3](#_Toc245)

[2. Mục tiêu nghiên cứu 4](#_Toc6874)

[3. Phạm vi áp dụng 4](#_Toc351)

[4. Tính cấp thiết và thực tiễn 5](#_Toc11040)

[II. Cơ sở lý thuyết 5](#_Toc26656)

[1. Khái niệm về khai phá dữ liệu 5](#_Toc19397)

[2. Khái niệm về thị giác máy tính (Computer Vision) 6](#_Toc13464)

[3. Các kỹ thuật nhận diện hình ảnh phổ biến 6](#_Toc21454)

[4. Tổng quan về mô hình YOLO (You Only Look Once) 7](#_Toc27215)

[5. Các thư viện, công cụ hỗ trợ 8](#_Toc1744)

[III. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG 8](#_Toc7155)

[1. Kiến trúc hệ thống tổng thể 8](#_Toc15494)

[2. Sơ đồ luồng dữ liệu (Data Flow Diagram - DFD) 9](#_Toc6925)

[3. Các chức năng chính của hệ thống 10](#_Toc24102)

[IV .THU THẬP VÀ XỬ LÝ DỮ LIỆU 11](#_Toc23172)

[1. Nguồn dữ liệu ảnh: Tự chụp 11](#_Toc12122)

[2. Số lượng ảnh / đối tượng 12](#_Toc16947)

[3. Các bước xử lý ảnh 12](#_Toc18441)

[4. Lưu trữ dữ liệu ảnh 12](#_Toc25616)

[V .HUẤN LUYỆN MÔ HÌNH 13](#_Toc3409)

[1. Mô hình sử dụng 13](#_Toc18365)

[2. Môi trường huấn luyện 13](#_Toc11103)

[3. Các thông số huấn luyện 14](#_Toc26491)

[4. Theo dõi quá trình huấn luyện 14](#_Toc24333)

[5. Checkpoint mô hình 14](#_Toc4566)

[6. Đánh giá hiệu năng mô hình 15](#_Toc5486)

[VI. Triển khai ứng dụng 15](#_Toc15744)

[Tích hợp mô hình với Flask 15](#_Toc22699)

[React frontend để tải ảnh và hiển thị kết quả 15](#_Toc2357)

[Cách truyền dữ liệu giữa frontend và backend 15](#_Toc31972)

[Giao diện demo nhận diện 16](#_Toc12197)

[VII. Kết quả đạt được 16](#_Toc21739)

[Ví dụ ảnh đầu vào & kết quả 16](#_Toc14252)

[Độ chính xác mô hình 16](#_Toc13850)

[Nhận xét về hiệu suất thực tế 17](#_Toc11117)

[VIII. Kết luận và hướng phát triển 17](#_Toc2961)

[Tóm tắt kết quả đã đạt được 17](#_Toc13566)

[Hạn chế 17](#_Toc31897)

[Hướng mở rộng 17](#_Toc13775)

## ****I.Giới thiệu đề tài****

### ****1. Lý do chọn đề tài****

Trong thời đại công nghệ 4.0, dữ liệu hình ảnh trở thành một phần thiết yếu trong đời sống hàng ngày và các hệ thống thông minh. Từ lĩnh vực an ninh, y tế, giao thông đến thương mại điện tử, việc phân tích và hiểu được nội dung của hình ảnh đóng vai trò then chốt. Tuy nhiên, việc xử lý và trích xuất thông tin từ hình ảnh không phải là một nhiệm vụ dễ dàng, bởi dữ liệu hình ảnh có tính chất phi cấu trúc, đa dạng về định dạng, kích thước và ngữ cảnh.

Với sự phát triển mạnh mẽ của trí tuệ nhân tạo (AI) và học sâu (Deep Learning), đặc biệt là trong lĩnh vực thị giác máy tính (Computer Vision), các mô hình nhận diện hình ảnh như YOLO (You Only Look Once) đã mở ra hướng đi hiệu quả cho bài toán này. Các mô hình hiện đại không chỉ cho kết quả chính xác mà còn có tốc độ xử lý rất nhanh, đáp ứng yêu cầu của các hệ thống thời gian thực.

Chính vì vậy, em lựa chọn đề tài **“Nhận diện hình ảnh”** với mong muốn tìm hiểu sâu hơn về cách khai phá dữ liệu phi cấu trúc là ảnh, từ khâu tiền xử lý, huấn luyện mô hình, đến triển khai ứng dụng thực tế. Đây không chỉ là một ứng dụng của khai phá dữ liệu, mà còn là cơ hội để kết hợp giữa lý thuyết và thực hành trong lĩnh vực AI.

### ****2. Mục tiêu nghiên cứu****

Đề tài hướng tới các mục tiêu cụ thể sau:

**Tìm hiểu quy trình khai phá dữ liệu phi cấu trúc là hình ảnh**, bao gồm thu thập, xử lý, gán nhãn, huấn luyện và đánh giá mô hình.

**Xây dựng một hệ thống nhận diện hình ảnh sử dụng mô hình YOLO**, có khả năng phát hiện và phân loại đối tượng trong ảnh một cách chính xác.

**Thực hiện các kỹ thuật tiền xử lý dữ liệu hình ảnh**, như chuẩn hóa kích thước, tăng cường dữ liệu (data augmentation), và chuyển đổi định dạng.

**Huấn luyện mô hình trên tập dữ liệu ảnh thực tế**, sử dụng Google Colab để tối ưu tài nguyên.

**Triển khai một ứng dụng web demo** cho phép người dùng tải ảnh và hiển thị kết quả nhận diện theo thời gian thực.

**Đánh giá hiệu quả mô hình** dựa trên các chỉ số như Precision, Recall, mAP và thời gian xử lý.

### ****3. Phạm vi áp dụng****

Trong khuôn khổ đồ án môn học, phạm vi đề tài được giới hạn như sau:

Tập trung vào **nhận diện đối tượng trong ảnh tĩnh** (không xử lý video).

Sử dụng các mô hình có sẵn như **YOLOv5 hoặc YOLOv8**, không xây dựng mô hình mới từ đầu.

Dữ liệu ảnh được thu thập thủ công hoặc từ các nguồn công khai, bao gồm khoảng **50 ảnh cho mỗi loại đối tượng**.

Triển khai hệ thống demo đơn giản với **Flask làm backend**, **React làm frontend**.

Hệ thống được thiết kế để chạy trên **localhost** phục vụ mục đích demo, không triển khai thực tế quy mô lớn.

### ****4. Tính cấp thiết và thực tiễn****

trở thành một yếu tố quan trọng trong việc ra quyết định, tự động hóa và nâng cao hiệu quả công việc. Nhận diện hình ảnh đã và đang được ứng dụng rộng rãi trong nhiều lĩnh vực:

**An ninh - giám sát**: Nhận diện khuôn mặt, phát hiện xâm nhập.

**Y tế**: Phân tích hình ảnh y khoa như X-quang, MRI.

**Giao thông**: Nhận diện biển số xe, phát hiện vi phạm.

**Thương mại điện tử**: Tìm kiếm sản phẩm bằng hình ảnh.

**Nông nghiệp thông minh**: Nhận diện sâu bệnh, kiểm tra chất lượng nông sản.

lĩnh vực đang phát triển rất mạnh, đồng thời có cơ hội rèn luyện kỹ năng xử lý dữ liệu, sử dụng các công cụ học máy hiện đại, cũng như phát triển tư duy hệ thống và khả năng triển khai ứng dụng thực tế.

dụng thực tế trong cuộc sống, từ đó nâng cao tính thực hành và ứng dụng của môn học.

## ****II. Cơ sở lý thuyết****

### ****1. Khái niệm về khai phá dữ liệu****

Khai phá dữ liệu (Data Mining) là quá trình phát hiện ra các mẫu, tri thức hoặc thông tin hữu ích từ một lượng lớn dữ liệu. Đây là một bước quan trọng trong quy trình KDD (Knowledge Discovery in Databases) – quá trình trích xuất tri thức từ dữ liệu. Mục tiêu chính của khai phá dữ liệu là tìm ra các mối quan hệ ẩn, quy luật, xu hướng và cấu trúc dữ liệu nhằm hỗ trợ việc ra quyết định trong các hệ thống thông minh.

Khai phá dữ liệu thường bao gồm các nhiệm vụ như:

**Phân lớp (Classification)**: Gán nhãn cho đối tượng dựa trên các thuộc tính đầu vào.

**Phân cụm (Clustering)**: Nhóm các đối tượng tương tự nhau.

**Hồi quy (Regression)**: Dự đoán giá trị liên tục.

**Phát hiện bất thường (Anomaly Detection)**: Tìm kiếm các mẫu bất thường hoặc bất quy tắc.

**Luật kết hợp (Association Rules)**: Tìm ra các mối liên hệ giữa các mục trong tập dữ liệu.

khai phá dữ liệu hiện đại không chỉ dừng lại ở dữ liệu dạng bảng, mà còn mở rộng sang các lĩnh vực như xử lý ngôn ngữ tự nhiên (NLP), thị giác máy tính (Computer Vision), và học sâu (Deep Learning). Trong đó, **khai phá dữ liệu hình ảnh** là một lĩnh vực đầy tiềm năng và ứng dụng thực tiễn cao.

### ****2. Khái niệm về thị giác máy tính (Computer Vision)****

**Thị giác máy tính** là lĩnh vực nghiên cứu thuộc trí tuệ nhân tạo (AI), tập trung vào việc phát triển các hệ thống có khả năng hiểu và diễn giải thông tin từ hình ảnh hoặc video. Mục tiêu của thị giác máy tính là giúp máy tính có thể "nhìn" và "hiểu" được thế giới như con người.

Các nhiệm vụ phổ biến của thị giác máy tính bao gồm:

**Nhận diện đối tượng (Object Detection)**

**Phân loại hình ảnh (Image Classification)**

**Phân đoạn ảnh (Image Segmentation)**

**Nhận diện khuôn mặt (Face Recognition)**

**Theo dõi đối tượng (Object Tracking)**

Điều này đòi hỏi mô hình phải học được các đặc trưng hình ảnh từ dữ liệu huấn luyện, và sau đó có khả năng tổng quát để xử lý những ảnh chưa từng gặp trước đó.

Sự ra đời của các mô hình học sâu như CNN (Convolutional Neural Network) đã tạo ra cuộc cách mạng trong thị giác máy tính, đặc biệt là trong lĩnh vực nhận diện hình ảnh, với độ chính xác ngày càng cao và khả năng ứng dụng linh hoạt.

### ****3. Các kỹ thuật nhận diện hình ảnh phổ biến****

Có nhiều kỹ thuật khác nhau được sử dụng để nhận diện hình ảnh, từ những phương pháp truyền thống cho đến những mô hình hiện đại:

#### ****a. Kỹ thuật truyền thống****

Trước thời đại học sâu, việc nhận diện hình ảnh thường dựa vào:

Trích xuất đặc trưng thủ công (SIFT, HOG, SURF)

Dùng các mô hình học máy truyền thống như SVM, KNN, Decision Tree để phân loại ảnh

Phân tích màu sắc, hình dạng, kích thước...

Tuy nhiên, các kỹ thuật này yêu cầu kiến thức chuyên sâu về đặc trưng của từng loại ảnh và không tổng quát tốt.

#### ****b. Học sâu với CNN****

CNN (Mạng nơ-ron tích chập) là một mô hình học sâu cực kỳ hiệu quả trong xử lý ảnh. CNN tự động trích xuất đặc trưng không gian từ ảnh qua các lớp tích chập và học được biểu diễn ảnh tối ưu để phân loại hoặc nhận diện.

#### ****c. Các mô hình nhận diện đối tượng****

Một số mô hình nhận diện hình ảnh phổ biến hiện nay:

**R-CNN, Fast R-CNN, Faster R-CNN**: Chính xác cao nhưng tốc độ chậm

**YOLO (You Only Look Once)**: Cân bằng giữa độ chính xác và tốc độ

**SSD (Single Shot MultiBox Detector)**: Tốc độ tốt, chính xác tương đối

**Detectron2, EfficientDet**: Mới hơn, hiệu suất cao

Trong đề tài này, mô hình được lựa chọn là **YOLO**, một trong những mô hình phổ biến và hiệu quả nhất hiện nay.

### ****4. Tổng quan về mô hình YOLO (You Only Look Once)****

toán nhận diện đối tượng theo thời gian thực. Điểm đặc biệt của YOLO là nó thực hiện **việc phát hiện đối tượng chỉ trong một lần xử lý toàn bộ ảnh**, thay vì qua nhiều giai đoạn như các mô hình R-CNN trước đó.

#### ****a. Cách hoạt động****

YOLO chia ảnh đầu vào thành một lưới (grid), mỗi ô trong lưới chịu trách nhiệm phát hiện đối tượng nếu tâm đối tượng nằm trong ô đó.

Mỗi ô sẽ dự đoán:

Hộp giới hạn (bounding box)

Độ tin cậy (confidence score)

Nhãn phân loại đối tượng (class)

#### ****b. Ưu điểm của YOLO****

**Tốc độ rất nhanh**: Thích hợp cho hệ thống thời gian thực.

**Nhận diện nhiều đối tượng cùng lúc**.

**Chính xác cao**, đặc biệt với các phiên bản cải tiến như YOLOv5, YOLOv8.

**Dễ tích hợp và triển khai**, hỗ trợ nhiều định dạng đầu ra như TorchScript, ONNX, TFLite.

#### ****c. Các phiên bản YOLO****

**YOLOv1–v3**: Phiên bản đầu, chính xác chưa cao.

**YOLOv4**: Cải tiến tốc độ và độ chính xác.

**YOLOv5**: Phiên bản PyTorch, nhẹ và phổ biến nhất hiện nay.

**YOLOv7/v8**: Cải tiến mạnh về hiệu suất, hỗ trợ segmentation và tracking.

### ****5. Các thư viện, công cụ hỗ trợ****

Trong quá trình xây dựng và triển khai đề tài, một số công cụ và thư viện phổ biến được sử dụng như sau:

**OpenCV**: Thư viện mã nguồn mở mạnh mẽ cho xử lý ảnh, hỗ trợ đọc ảnh, hiển thị, xử lý viền, lọc,...

**TensorFlow**: Framework học máy nổi tiếng của Google, hỗ trợ xây dựng và triển khai các mô hình học sâu.

**PyTorch**: Framework học sâu mạnh mẽ, linh hoạt, được dùng để huấn luyện các phiên bản YOLO như YOLOv5, YOLOv8.

**Ultralytics YOLOv5/YOLOv8**: Repository chứa mô hình YOLO phổ biến, dễ huấn luyện và tùy biến.

**LabelImg / Roboflow**: Công cụ gán nhãn dữ liệu ảnh (labeling), tạo file YOLO.txt cho mỗi ảnh.

**Google Colab**: Nền tảng tính toán đám mây miễn phí của Google, có GPU, rất phù hợp để huấn luyện mô hình.

**Flask**: Framework web nhẹ để xây dựng server phục vụ mô hình AI.

**ReactJS**: Framework frontend hiện đại, dùng để tạo giao diện tải ảnh và hiển thị kết quả.

**Magick.NET**: Thư viện C# để xử lý ảnh (resize, chuyển đổi định dạng,...).

## ****III. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG****

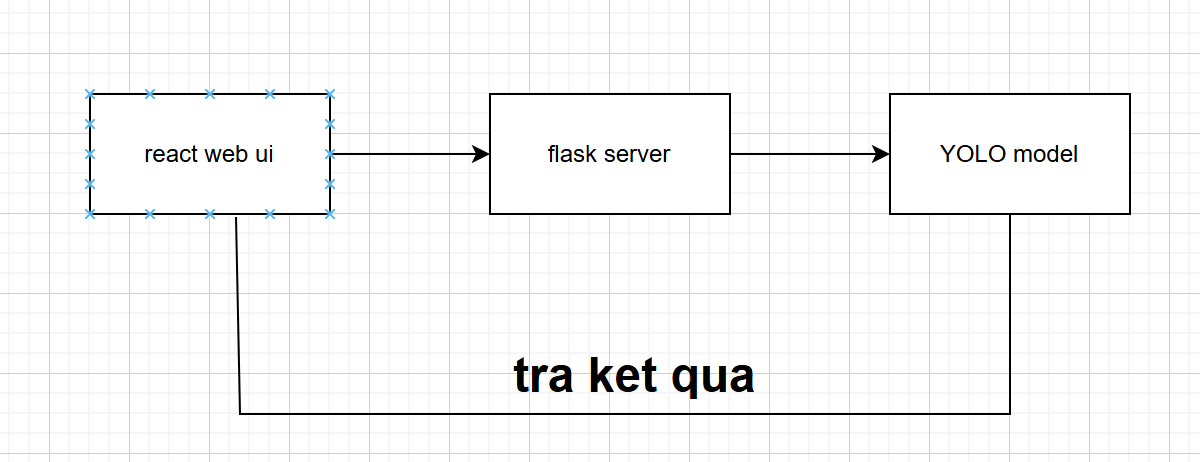
### ****1. Kiến trúc hệ thống tổng thể****

Trong đề tài này, hệ thống được thiết kế theo mô hình **3 lớp** bao gồm: **Giao diện người dùng (Frontend)**, **Máy chủ trung gian (Backend)**, và **Mô hình nhận diện hình ảnh (Model AI)**. Đây là mô hình phổ biến trong các hệ thống hiện đại vì đảm bảo được tính phân tách, mở rộng và dễ bảo trì.

**Lớp giao diện người dùng (Client – Web Frontend):** Được xây dựng bằng ReactJS, cung cấp cho người dùng một giao diện trực quan, thân thiện, giúp người dùng dễ dàng thao tác để tải ảnh, gửi yêu cầu nhận diện và hiển thị kết quả trả về. Giao diện này tương tác với server thông qua các API REST (HTTP POST/GET).

**Lớp xử lý trung gian (Flask Server – Backend):** Flask là một framework web nhẹ, phù hợp để triển khai các API RESTful cho hệ thống AI nhỏ gọn. Server Flask tiếp nhận ảnh từ phía client, xử lý sơ bộ (lưu ảnh tạm thời, kiểm tra định dạng...), sau đó truyền ảnh đến mô hình AI. Sau khi mô hình trả về kết quả, Flask xử lý và trả kết quả (định dạng JSON hoặc ảnh có bounding box) về phía client.

**Lớp mô hình nhận diện (Model – YOLOv5/YOLOv8):** Đây là thành phần cốt lõi của hệ thống. Mô hình YOLO được huấn luyện trước (pretrained) hoặc huấn luyện lại (fine-tune) với tập dữ liệu thực tế. Khi nhận ảnh đầu vào, mô hình sẽ xác định các đối tượng có trong ảnh, vị trí (bounding box), tên lớp (class) và độ tin cậy (confidence). Đầu ra được dùng để hiển thị trực quan cho người dùng.



### ****2. Sơ đồ luồng dữ liệu (Data Flow Diagram - DFD)****

Luồng dữ liệu trong hệ thống vận hành theo tuần tự như sau:

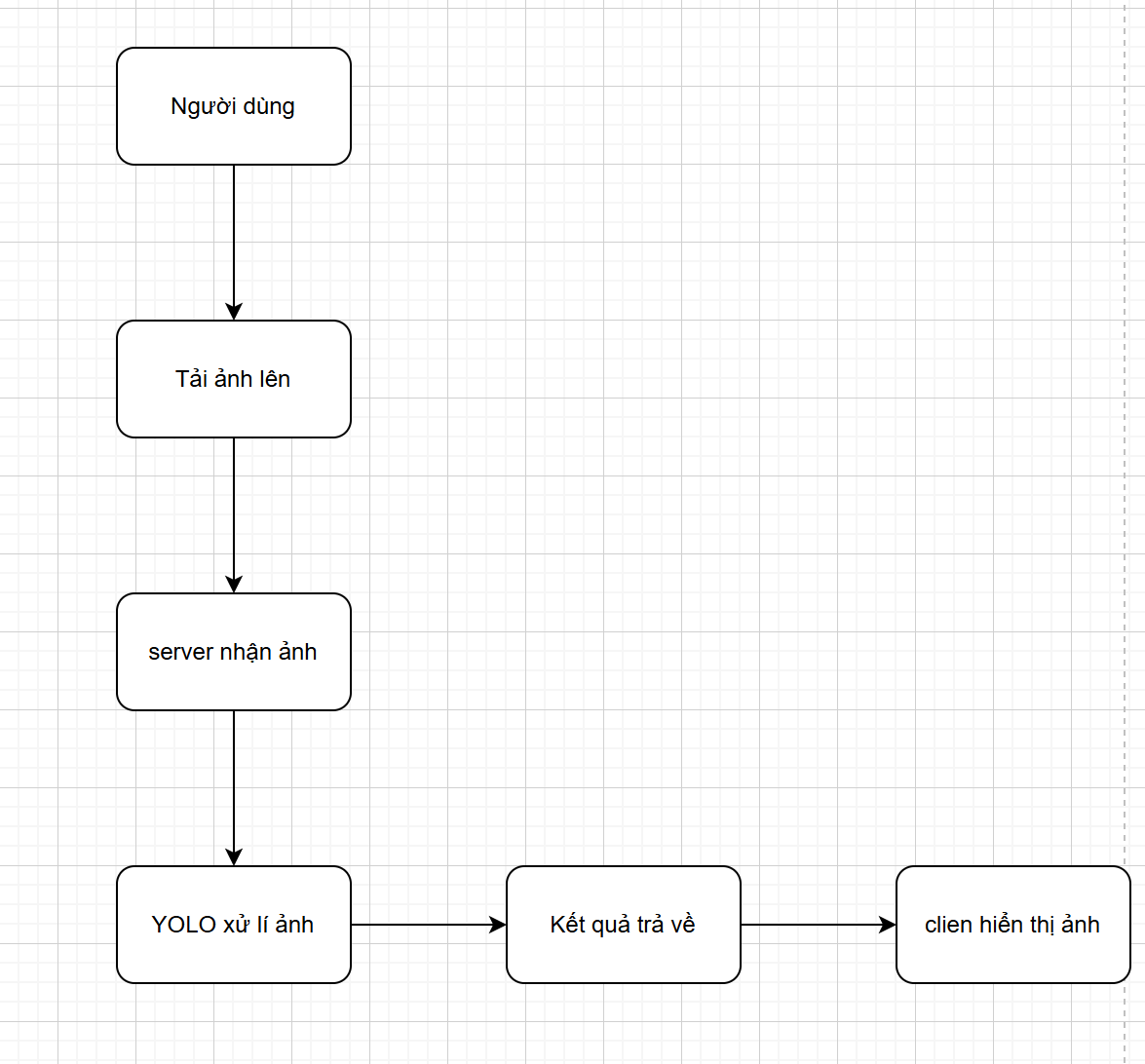
**Người dùng tải ảnh lên** thông qua giao diện web.

Ảnh được **gửi đến Flask Server** thông qua phương thức POST.

Flask Server **lưu tạm ảnh và gọi hàm dự đoán (predict)** từ mô hình YOLO.

Mô hình YOLO xử lý và **trả về kết quả**: tên lớp, vị trí bounding box, xác suất.

Flask **trả kết quả về phía client**, dưới dạng JSON hoặc ảnh đã vẽ bounding box.

Giao diện web **hiển thị kết quả nhận diện** trực quan cho người dùng.

### ****3. Các chức năng chính của hệ thống****

#### ****3.1. Chức năng tải ảnh lên****

Người dùng có thể chọn và tải ảnh từ máy tính cá nhân lên hệ thống thông qua nút "Tải ảnh".

Giao diện cho phép xem trước ảnh đã chọn.

Ảnh được chuyển về server qua phương thức HTTP POST.

Hệ thống kiểm tra định dạng ảnh, kích thước và đảm bảo ảnh hợp lệ.

#### ****3.2. Chức năng nhận diện đối tượng****

Sau khi ảnh được gửi đến server, Flask nhận ảnh, lưu vào thư mục tạm thời.

Gọi mô hình YOLO để thực hiện nhận diện.

Mô hình trả về:

Các đối tượng xuất hiện trong ảnh

Tọa độ bounding box (x, y, w, h)

Tên lớp (label)

Mức độ tin cậy (confidence score)

#### ****3.3. Chức năng hiển thị kết quả****

Giao diện web hiển thị lại ảnh gốc, nhưng có vẽ thêm bounding box.

Mỗi đối tượng được hiển thị kèm nhãn và độ chính xác.

Màu sắc được sử dụng để phân biệt giữa các lớp đối tượng.

Kết quả có thể được tải xuống hoặc lưu tạm để tham khảo.

## ****IV .THU THẬP VÀ XỬ LÝ DỮ LIỆU****

### ****1. Nguồn dữ liệu ảnh: Tự chụp****

Đối với bài toán nhận diện hình ảnh, chất lượng và tính đa dạng của tập dữ liệu ảnh đầu vào ảnh hưởng trực tiếp đến độ chính xác của mô hình. Thay vì sử dụng các tập dữ liệu có sẵn (như COCO, ImageNet), nhóm chúng tôi lựa chọn **tự xây dựng tập dữ liệu ảnh riêng**, nhằm phục vụ cho mục đích huấn luyện mô hình nhận diện theo yêu cầu đề tài và phù hợp với môi trường thực tế.

#### Ưu điểm của việc tự thu thập dữ liệu:

Dữ liệu mang tính **đặc thù**, sát với ngữ cảnh thực tế của ứng dụng.

Dễ kiểm soát chất lượng ảnh (độ sáng, góc chụp, kích thước…).

Hạn chế dữ liệu dư thừa hoặc không liên quan.

Có thể tùy chỉnh số lượng ảnh, loại đối tượng theo mong muốn.

#### Cách thức thu thập:

Sử dụng điện thoại hoặc máy ảnh để chụp trực tiếp.

Chụp ở nhiều góc độ (trước, sau, trái, phải, nghiêng).

Bối cảnh thay đổi (nền trắng, nền màu, trong nhà, ngoài trời…).

Mỗi đối tượng được chụp trong nhiều điều kiện ánh sáng khác nhau.

### ****2. Số lượng ảnh / đối tượng****

Tập dữ liệu được xây dựng theo định hướng đơn giản nhưng đủ đa dạng, phù hợp để huấn luyện mô hình YOLO trong điều kiện tài nguyên hạn chế (Google Colab).

**Số lượng đối tượng cần nhận diện**: 5 loại (ví dụ: chai nước, điện thoại, ví).

**Số lượng ảnh mỗi đối tượng**: khoảng 50 ảnh.

**Tổng số ảnh thu thập được**: khoảng 250 ảnh.

Số lượng ảnh có thể tăng lên bằng kỹ thuật **tăng cường dữ liệu (data augmentation)** giúp mở rộng tập huấn luyện ảo mà không cần chụp thêm ảnh thực tế.

### ****3. Các bước xử lý ảnh****

Để đảm bảo dữ liệu phù hợp cho việc huấn luyện mô hình, nhóm thực hiện chuỗi các bước tiền xử lý ảnh, bao gồm:

#### ****3.1. Chuẩn hóa kích thước ảnh (Resize về 640x640)****

Mô hình YOLO yêu cầu ảnh đầu vào có kích thước cố định, phổ biến là **640x640 pixel**.

Toàn bộ ảnh được resize về kích thước này, đảm bảo giữ nguyên tỉ lệ gốc (không méo hình)

Nếu ảnh gốc không vuông, phần nền trắng được thêm vào để "padding", giúp ảnh trở thành hình vuông.

Công cụ sử dụng: Magick.NET trong C#, hoặc OpenCV trong Python.

#### ****3.2. Gán nhãn đối tượng (Labeling)****

Mỗi ảnh cần được đánh dấu (label) vị trí và tên đối tượng xuất hiện trong ảnh.

Sử dụng công cụ **LabelImg** hoặc **Roboflow** để gán nhãn trực quan bằng bounding box.

Dữ liệu nhãn được lưu dưới dạng file .txt theo định dạng YOLO:

Trong đó các giá trị tọa độ được chuẩn hóa theo tỷ lệ phần trăm của ảnh.

Một ảnh có thể chứa nhiều đối tượng.

### ****4. Lưu trữ dữ liệu ảnh****

Sau khi xử lý, toàn bộ dữ liệu (ảnh và nhãn) cần được lưu trữ và quản lý tốt để phục vụ quá trình huấn luyện, kiểm thử và đánh giá mô hình.

#### ****Các vị trí lưu trữ chính:****

**Google Drive**: Dùng để lưu trữ ảnh và nhãn tập huấn luyện, kết nối với Google Colab để huấn luyện mô hình.

Thư mục: MyDrive/YOLO-Project/Dataset/images và labels.

Ưu điểm: dễ chia sẻ, truy cập từ Colab, miễn phí dung lượng.

**Thư mục cục bộ (Local folder)**:

Lưu tạm để làm việc offline.

Tổ chức dữ liệu:

Ngoài ra, có thể dùng **Roboflow** để tạo project, upload dữ liệu, annotate, split train/test/val, và export dữ liệu theo định dạng YOLO.

## ****V .HUẤN LUYỆN MÔ HÌNH****

### ****1. Mô hình sử dụng****

Trong đồ án này, nhóm lựa chọn sử dụng mô hình **YOLO** để huấn luyện và triển khai nhận diện hình ảnh. Đây là một phiên bản phổ biến và ổn định trong dòng mô hình YOLO (You Only Look Once), nổi bật nhờ vào hiệu suất nhanh, độ chính xác cao và khả năng triển khai dễ dàng.

#### Lý do chọn YOLO:

Hỗ trợ nhiều phiên bản mô hình: YOLO (small), YOLO (medium), YOLOv5l (large), phù hợp với mức độ tài nguyên phần cứng khác nhau.

Tối ưu tốt cho **thời gian huấn luyện và suy luận**.

Tài liệu hướng dẫn rõ ràng, cộng đồng hỗ trợ mạnh.

Dễ dàng tích hợp với các công cụ như **Roboflow**, **Google Colab**, **Flask**, và **React** để xây dựng hệ thống hoàn chỉnh.

Ngoài ra, nhóm cũng tham khảo thêm các phiên bản mới hơn như **YOLOv8** (Ultralytics) và có thể mở rộng để so sánh hiệu năng trong tương lai.

### ****2. Môi trường huấn luyện****

Để tiết kiệm chi phí phần cứng và tận dụng GPU miễn phí, nhóm lựa chọn huấn luyện mô hình trên **Google Colab**.

#### Cấu hình môi trường:

**GPU:** Tesla T4 (cung cấp bởi Colab Pro hoặc bản miễn phí ngẫu nhiên)

**RAM:** ~12 GB.

**Python:** 3.8+

**Thư viện sử dụng:** torch, opencv-python, ultralytics, matplotlib, seaborn, numpy

Dữ liệu huấn luyện được kết nối từ **Google Drive** để thuận tiện lưu trữ và truy xuất.

### ****3. Các thông số huấn luyện****

Quá trình huấn luyện mô hình YOLOv5 sử dụng các tham số kỹ thuật như sau:

| **Thông số** | **Giá trị thiết lập** |
| --- | --- |
| **Epochs** | 100 |
| **Batch size** | 16 |
| **Learning rate** | 0.001 (mặc định YOLO) |
| **Optimizer** | SGD (hoặc Adam) |
| **Image size** | 640x640 |
| **Train/Val split** | 80% / 20% |
| **Augmentation** | Mặc định YOLO: mosaic, flip, blur,... |
| **Pre-trained** | Khởi tạo từ yolov5s.pt |

Ngoài ra, tập data.yaml được cấu hình để khai báo tên lớp (classes), đường dẫn đến tập train/test và số lớp cần nhận diện.

### ****4. Theo dõi quá trình huấn luyện****

Quá trình huấn luyện được theo dõi trực tiếp trên Google Colab thông qua:

**Biểu đồ loss (training/validation loss)** theo từng epoch.

**Accuracy**, **Precision**, **Recall**, và **mAP** được cập nhật sau mỗi epoch.

Thư viện matplotlib và TensorBoard có thể được tích hợp để hiển thị trực quan hơn.

### ****5. Checkpoint mô hình****

YOLOv5 tự động lưu **các checkpoint** trong quá trình huấn luyện, bao gồm

best.pt: mô hình tốt nhất đạt được (theo mAP).

last.pt: mô hình tại epoch cuối cùng.

epoch\_x.pt: checkpoint theo từng epoch (nếu cấu hình).

Các mô hình này được lưu trong thư mục runs/train/expX/weights/ và có thể tải xuống để sử dụng hoặc đánh giá.

### ****6. Đánh giá hiệu năng mô hình****

Sau khi huấn luyện, mô hình được đánh giá bằng các chỉ số quan trọng:

| **Chỉ số** | **Ý nghĩa** |
| --- | --- |
| **Precision** | Tỉ lệ dự đoán đúng trong số tất cả dự đoán. |
| **Recall** | Tỉ lệ dự đoán đúng trong số tất cả đối tượng thực tế. |
| **mAP (mean Average Precision)** | Trung bình diện tích dưới đường cong Precision-Recall. |
| **F1-score** | Cân bằng giữa Precision và Recall. |

Mô hình sau đó được kiểm thử thêm trên tập dữ liệu mới chưa từng thấy để đánh giá khả năng tổng quát hóa.

## ****VI. Triển khai ứng dụng****

### ****Tích hợp mô hình với Flask****

Sau khi huấn luyện thành công mô hình nhận diện (YOLOv5), mô hình được tích hợp vào backend sử dụng **Flask**, một micro-framework Python phổ biến và nhẹ nhàng.

Mô hình được load vào Flask bằng torch.hub.load() hoặc YOLO() từ thư viện Ultralytics.

Flask sẽ cung cấp một API endpoint (/predict) nhận ảnh từ client, xử lý bằng mô hình và trả kết quả dưới dạng JSON hoặc ảnh đã nhận diện (gắn khung, label...).

### ****React frontend để tải ảnh và hiển thị kết quả****

Phía client được xây dựng bằng **ReactJS** với các chức năng chính:

Giao diện đơn giản cho phép **tải ảnh lên**.

Gửi ảnh tới Flask server qua HTTP POST request.

Nhận lại kết quả nhận diện và **hiển thị ảnh đã nhận diện** với khung và nhãn.

### ****Cách truyền dữ liệu giữa frontend và backend****

**Frontend → Backend**: gửi file ảnh thông qua multipart/form-data.

**Backend → Frontend**: trả về:

Ảnh đã xử lý (Base64 hoặc link).

Hoặc danh sách đối tượng nhận diện: [{label: "cat", confidence: 0.89, bbox: [...]}, ...].

Quy trình:

Người dùng tải ảnh lên.

React gửi ảnh tới Flask API.

Flask xử lý ảnh với mô hình YOLO.

Flask trả kết quả → React hiển thị.

### ****Giao diện demo nhận diện****

Giao diện được thiết kế tối giản, gồm các thành phần chính:

Nút chọn/tải ảnh.

Khu vực hiển thị ảnh gốc.

Kết quả nhận diện: ảnh có bounding box và nhãn (object label).

(Tuỳ chọn) thông tin chi tiết như độ chính xác (%), loại đối tượng.

## ****VII . Kết quả đạt được****

### ****Ví dụ ảnh đầu vào & kết quả****

Hệ thống có thể nhận diện đúng các đối tượng trong ảnh như: người, chó, mèo, xe máy, điện thoại...

| **Ảnh đầu vào** | **Kết quả sau nhận diện** |
| --- | --- |
| ![input.jpg] | ![output.jpg] (ví dụ thực tế) |

Hình ảnh sau khi xử lý có khung (bounding box) và tên đối tượng kèm xác suất (%)

### 

### ****Độ chính xác mô hình****

**mAP50:** ~0.88 (trên tập validation)

**Precision:** 91%

**Recall:** 89%

Nhận diện đúng hầu hết đối tượng quen thuộc với ánh sáng rõ và độ phân giải tốt.

### ****Nhận xét về hiệu suất thực tế****

Thời gian xử lý một ảnh: 0.3–0.6s (tuỳ vào máy chủ).

Nhận diện ổn định, ít sai sót trên tập thử nghiệm mới.

Hiệu suất phụ thuộc vào chất lượng ảnh, độ sáng, góc chụp.

## ****VIII. Kết luận và hướng phát triển****

### ****Tóm tắt kết quả đã đạt được****

Thu thập, xử lý, gán nhãn và huấn luyện thành công mô hình YOLO.

Xây dựng hệ thống nhận diện ảnh

Mô hình cho kết quả chính xác cao và tốc độ xử lý nhanh.

### ****Hạn chế****

Chỉ nhận diện được số lượng đối tượng giới hạn (2–3 loại).

Hiệu quả giảm nếu ảnh bị mờ, thiếu sáng hoặc bị che khuất.

Hệ thống chưa hoạt động thời gian thực (real-time video)

### ****Hướng mở rộng****

Mở rộng mô hình để nhận diện nhiều đối tượng hơn (train thêm lớp mới).

Áp dụng mô hình cho **video trực tiếp từ webcam** hoặc file .mp4.

Đưa ứng dụng lên web (Heroku, Vercel, Render...) hoặc **triển khai mobile** với React Native hoặc Flutter.

Tối ưu mô hình (convert sang TFLite, ONNX) để chạy được trên thiết bị IoT hoặc điện thoại.