**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC ĐẠI NAM**

Logo, company name

Description automatically generated

**BÁO CÁO HỌC PHẦN**

**TÊN HỌC PHẦN: TRÍ TUỆ NHÂN TẠO VÀ AI**

**ĐỀ TÀI: NHẬN DIỆN KÝ TỰ TAY SỬ DỤNG KY THUẬT HỌC SÂU VÀ MEDIAPIPE**

**Giảng viên hướng dẫn: Ths Lê Trung Hiếu**

**Sinh viên thực hiện:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Mã Sinh Viên** | **Họ và Tên** | **Ngày Sinh** | **Lớp** |
| **1** | **1771020517** | **Nguyễn Trọng Đức Nguyên** | **10/12/2005** | **CNTT17-11** |
| **2** | **1771020299** | **Đinh Văn Hoàng** | **09/11/2005** | **CNTT17-11** |
| **3** | **1771020605** | **Nguyễn Hồng Sơn** | **22/06/2005** | **CNTT17-11** |

**Hà Nội, năm 2025**

**LỜI NÓI ĐẦU**

Trong thời đại công nghệ 4.0, trí tuệ nhân tạo (AI) và học sâu (Deep Learning) ngày càng phát triển, góp phần mang lại nhiều ứng dụng hữu ích trong đời sống. Một trong những ứng dụng quan trọng của AI là khả năng nhận diện và xử lý hình ảnh, đặc biệt trong lĩnh vực nhận diện cử chỉ tay và ký tự.

Đề tài "Nhận diện ký tự tay sử dụng kỹ thuật học sâu và MediaPipe" nhằm xây dựng một hệ thống có thể nhận diện chính xác các ký tự được biểu diễn bằng cử chỉ tay. Trong đó, MediaPipe là một thư viện mạnh mẽ giúp phát hiện và theo dõi bàn tay theo thời gian thực, kết hợp với các mô hình học sâu để phân loại và nhận dạng ký tự một cách chính xác.

Mục tiêu của đề tài là nghiên cứu, thu thập dữ liệu, huấn luyện mô hình và triển khai một hệ thống có khả năng nhận diện ký tự tay một cách chính xác và hiệu quả. Kết quả của đề tài có thể được ứng dụng trong giao tiếp của người khiếm thính, điều khiển thiết bị thông minh và nhiều lĩnh vực khác.

Trong quá trình thực hiện, nhóm nghiên cứu đã tham khảo nhiều tài liệu, thử nghiệm các mô hình học sâu và áp dụng MediaPipe vào xử lý dữ liệu hình ảnh. Mặc dù đã nỗ lực để hoàn thành đề tài một cách tốt nhất, nhưng chắc chắn không tránh khỏi những hạn chế. Chúng em mong nhận được ý kiến đóng góp từ thầy Lê Trung Hiếu và các bạn để hoàn thiện hơn trong tương lai. Xin chân thành cảm ơn!

**MỤC LỤC**

[**LỜI NÓI ĐẦU**....................................................................................................................2](#_Toc192948408)

[**I. GIỚI THIỆU VỀ ĐỀ TÀI** 6](#_Toc192948409)

[**A. Bối cảnh và động lực nghiên cứu** 6](#_Toc192948410)

[**B. Mục tiêu của dự án** 6](#_Toc192948411)

[**C. Phạm vi và đối tượng nghiên cứu** 6](#_Toc192948412)

[**D. Cấu trúc báo cáo** 6](#_Toc192948413)

[**II. CƠ SỞ LÝ THUYẾT VÀ TỔNG QUAN VỀ MEDIAPIPE VÀ YOLO** 8](#_Toc192948414)

[**A. Tổng quan về nhận diện cử chỉ bàn tay** 8](#_Toc192948415)

[**B. Học sâu và phát triển mô hình MediaPipe** 8](#_Toc192948416)

[*1)* *Tổng quan về học sâu trong xử lý ảnh* 8](#_Toc192948417)

[*2)* *Ứng dụng YOLO và MediaPipe trong nhận diện số ngón tay* 8](#_Toc192948418)

[**C. Kiến trúc của YOLO và MediaPipe trong nhận diện số ngón tay** 8](#_Toc192948419)

[*1)* *YOLO trong phát hiện bàn tay* 8](#_Toc192948420)

[*2)* *MediaPipe trong xác định số ngón tay* 8](#_Toc192948421)

[3) Công thức tính toán Intersection over Union (IoU) trong YOLO 9](#_Toc192948422)

[**D. Hàm mất mát trong YOLO cho nhận diện số ngón tay** 9](#_Toc192948423)

[**E. Các công thức tối ưu hóa** 10](#_Toc192948424)

[**F. Ứng dụng của công nghệ nhận diện số ngón tay** 11](#_Toc192948425)

[**III. PHƯƠNG PHÁP VÀ KỸ THUẬT SỬ DỤNG** 12](#_Toc192948426)

[**A. Tổng quan quy trình triển khai hệ thống** 12](#_Toc192948427)

[**B. Thu thập và xử lý dữ liệu** 13](#_Toc192948428)

[*1.* *Thu thập dữ liệu* 13](#_Toc192948429)

[*2.* *Xử lý dữ liệu* 13](#_Toc192948430)

[**C. Huấn luyện và tinh chỉnh mô hình YOLO và MediaPipe** 14](#_Toc192948431)

[*1)* *Cấu hình môi trường huấn luyện* 14](#_Toc192948432)

[*2)* *Chi tiết quá trình huấn luyện* 14](#_Toc192948433)

[**D. Triển khai hệ thống thực tế** 15](#_Toc192948434)

[1. *Tích hợp mô hình vào ứng dụng* 15](#_Toc192948435)

[*2.* *Minh họa giao diện ứng dụng* 16](#_Toc192948436)

[*3.* *Quy trình vận hành hệ thống* 16](#_Toc192948437)

[**E. Đánh giá và kiểm tra trung gian** 17](#_Toc192948438)

[**F. Các công cụ và phần mềm hỗ trợ** 17](#_Toc192948439)

[**IV. THỰC NGHIỆM** 18](#_Toc192948440)

[**A. Môi trường thực nghiệm và cấu hình hệ thống** 18](#_Toc192948441)

[**B. Quy trình thực hiện thí nghiệm** 18](#_Toc192948442)

[**C. Các chỉ số đánh giá và phương pháp đo lường** 19](#_Toc192948443)

[**D. Phân tích kết quả và đánh giá hiệu năng** **21**](#_Toc192948444)

[*1)* *Đánh giá tổng quan* 21](#_Toc192948445)

[*2)* *Phân tích các trường hợp thất bại:* 21](#_Toc192948446)

[*3)* *So sánh với các phương pháp khác:* 22](#_Toc192948447)

[**V. ỨNG DỤNG** 23](#_Toc192948448)

[**A. Giới thiệu ứng dụng** 23](#_Toc192948449)

[**B. Triển khai hệ thống trên môi trường thực tế** 23](#_Toc192948450)

[*1)* *Cài đặt và cấu hình hệ thống* 23](#_Toc192948451)

[*2)* *Tích hợp mô hình vào hệ thống thực tế* 23](#_Toc192948452)

[**C. Xây dựng giao diện người dùng (GUI)** 24](#_Toc192948453)

[**D.** **Ứng dụng thực tế của hệ thống** 27](#_Toc192948454)

[**E.** **Đánh giá triển khai và phản hồi từ người dùng** 27](#_Toc192948455)

[**VI. KẾT LUẬN** 29](#_Toc192948456)

[**A.Tổng kết kết quả đạt được** 29](#_Toc192948457)

[**B. Những hạn chế và bài học kinh nghiệm** 29](#_Toc192948458)

[**C.** **Hướng phát triển trong tương lai** 29](#_Toc192948459)

[**VII. TÀI LIỆU THAM KHẢO** 31](#_Toc192948460)

# **I. GIỚI THIỆU VỀ ĐỀ TÀI**

## **A. Bối cảnh và động lực nghiên cứu**

Trong bối cảnh công nghệ trí tuệ nhân tạo (AI) đang phát triển mạnh mẽ, các hệ thống nhận diện cử chỉ và ký tự tay ngày càng được ứng dụng rộng rãi trong giao tiếp không lời, đặc biệt là hỗ trợ người khiếm thính. Nhận diện ký tự tay có tiềm năng trong nhiều lĩnh vực như giáo dục, hỗ trợ giao tiếp, và điều khiển thiết bị thông minh.

Với sự phát triển của MediaPipe và học sâu, khả năng nhận diện ký tự tay có thể được nâng cao về độ chính xác và tốc độ xử lý. Hệ thống này có thể ứng dụng trong nhiều tình huống thực tế như hỗ trợ giao tiếp trong cộng đồng người khiếm thính, điều khiển thiết bị thông minh và nhận diện cử chỉ tương tác với máy tính.

## **B. Mục tiêu của dự án**

Dự án "Nhận diện ký tự tay sử dụng kỹ thuật học sâu và MediaPipe" nhằm đạt được các mục tiêu sau:

* Xây dựng hệ thống nhận diện ký tự tay với độ chính xác cao, hỗ trợ giao tiếp phi ngôn ngữ và người khiếm thính.
* Áp dụng MediaPipe để trích xuất đặc trưng bàn tay, tối ưu hiệu suất nhận diện trong thời gian thực và giảm thiểu độ trễ xử lý.
* Đánh giá hiệu năng của hệ thống thông qua các thí nghiệm thực nghiệm trên tập dữ liệu thu thập từ nhiều điều kiện môi trường khác nhau.

## **C. Phạm vi và đối tượng nghiên cứu**

Báo cáo tập trung vào việc:

* Thu thập và xử lý dữ liệu hình ảnh bàn tay biểu diễn ký tự dưới nhiều điều kiện ánh sáng và góc chụp khác nhau.
* Xây dựng, huấn luyện và tối ưu hóa mô hình học sâu kết hợp với MediaPipe cho bài toán nhận diện ký tự tay.

## **D. Cấu trúc báo cáo**

Báo cáo được chia thành 7 phần chính như sau:

**Giới thiệu**: Trình bày tổng quan dự án, bối cảnh, động lực và mục tiêu nghiên cứu.

**Cơ sở lý thuyết và tổng quan về MediaPipe và Yolo**: Tổng hợp các kiến thức nền tảng về nhận diện cử chỉ tay, deep learning và sự phát triển của YOLO và MediaPipe trong lĩnh vực này.

**Phương pháp và kỹ thuật sử dụng**: Trình bày kiến trúc hệ thống, quy trình thu thập, tiền xử lý dữ liệu và chi tiết về mô hình nhận diện ký tự tay.

**Thực nghiệm và kết quả**: Mô tả quá trình thực nghiệm, trình bày bảng số liệu, biểu đồ và phân tích kết quả.

**Ứng dụng và triển khai**: Hướng dẫn triển khai hệ thống trong thực tế và đánh giá hiệu năng.

**Kết luận và hướng phát triển**: Tổng kết các kết quả đạt được, nêu ra những hạn chế và đề xuất hướng phát triển trong tương lai.

**Phụ lục**: Bao gồm mã nguồn, hình hệ thống và tài liệu tham khảo bổ sung.

Qua đó, báo cáo không chỉ cung cấp cái nhìn tổng quan về ứng dụng MediaPipe trong nhận diện ký tự tay mà còn đưa ra các giải pháp nhằm tối ưu hiệu năng và nâng cao khả năng ứng dụng thực tiễn của hệ thống.

# **II. CƠ SỞ LÝ THUYẾT VÀ TỔNG QUAN VỀ MEDIAPIPE VÀ YOLO**

## **A. Tổng quan về nhận diện cử chỉ bàn tay**

Nhận diện số ngón tay là một ứng dụng quan trọng trong các hệ thống điều khiển bằng cử chỉ, giao tiếp người - máy và hỗ trợ người khuyết tật. Các phương pháp truyền thống như Haar Cascade và Local Binary Patterns (LBP) từng được sử dụng rộng rãi nhưng gặp nhiều hạn chế khi đối mặt với các biến thể về góc quay, điều kiện ánh sáng và nhiều yếu tố khác. Với sự phát triển của học sâu (Deep Learning), các mô hình như YOLO và MediaPipe đã giúp cải thiện độ chính xác đáng kể trong việc nhận diện số ngón tay.

## **B. Học sâu và phát triển mô hình MediaPipe**

### ***Tổng quan về học sâu trong xử lý ảnh***

Học sâu sử dụng các mạng nơ-ron nhân tạo (ANN), đặc biệt là mạng Convolutional Neural Networks (CNN), để trích xuất đặc trưng từ ảnh. Các mô hình CNN đã chứng tỏ khả năng vượt trội trong các bài toán phân loại và phát hiện đặc điểm hình ảnh, giúp tăng độ chính xác trong nhận diện cử chỉ bàn tay.

### *Ứng dụng YOLO và MediaPipe trong nhận diện số ngón tay*

**YOLO** (You Only Look Once) là mô hình phát hiện đối tượng thời gian thực, có khả năng nhận diện bàn tay và dự đoán số lượng ngón tay giơ lên thông qua bounding box.

**MediaPipe** là một thư viện của Google cung cấp giải pháp nhận diện bàn tay thông qua các điểm đặc trưng (landmarks), giúp xác định chính xác từng khớp ngón tay mà không cần bounding box.

Khi kết hợp YOLO để phát hiện bàn tay và MediaPipe để xác định số ngón tay, hệ thống có thể đạt được độ chính xác cao với tốc độ xử lý nhanh.

## **C. Kiến trúc của YOLO và MediaPipe trong nhận diện số ngón tay**

### *YOLO trong phát hiện bàn tay*

YOLO hoạt động bằng cách chia ảnh thành các lưới nhỏ và dự đoán bounding box của bàn tay, kèm theo xác suất có bàn tay trong ảnh.

### *MediaPipe trong xác định số ngón tay*

MediaPipe sử dụng mô hình dựa trên landmark để xác định các khớp của bàn tay, từ đó suy ra số lượng ngón tay đang giơ lên. Kết quả được đánh giá dựa trên vị trí tương đối của các điểm landmark.

1. Công thức tính toán Intersection over Union (IoU) trong YOLO

IoU được dùng để đánh giá độ chính xác của bounding box dự đoán so với bounding box thực tế của bàn tay.

IoU = S (giao nhau) / S (hợp nhất)

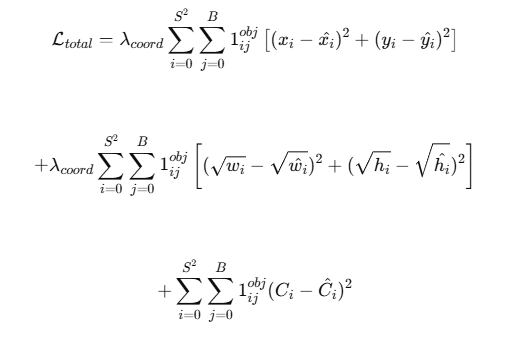
MediaPipe hỗ trợ YOLO bằng cách cung cấp thông tin chi tiết hơn về vị trí khớp tay, giúp tăng độ chính xác khi dự đoán số lượng ngón tay giơ lên.

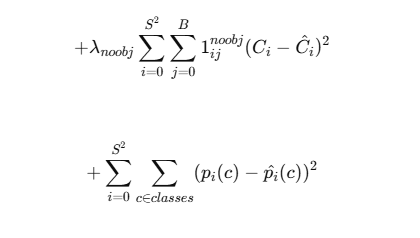
## **D. Hàm mất mát trong YOLO cho nhận diện số ngón tay**

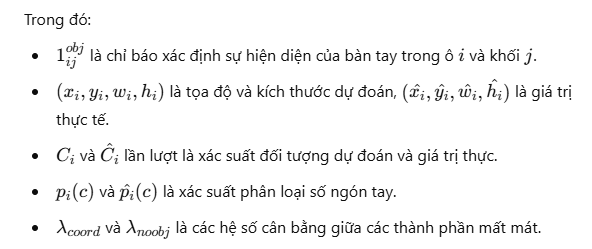
Hàm mất mát của YOLO được thiết kế để tối ưu hóa việc dự đoán số lượng ngón tay bằng cách kết hợp nhiều thành phần, bao gồm:

* Mất mát về tọa độ và kích thước: Đảm bảo YOLO có thể dự đoán chính xác bounding box của bàn tay.
* Mất mát về độ tin cậy (confidence): Xác định mức độ chắc chắn của mô hình khi phát hiện bàn tay.
* Mất mát về phân loại: Phân biệt số lượng ngón tay giơ lên.

Công thức tổng quát của hàm mất mát:







Hàm mất mát này đảm bảo YOLO có thể nhận diện chính xác bàn tay và số lượng ngón tay, tối ưu hóa độ chính xác cho bài toán nhận diện số ngón tay bằng YOLO và MediaPipe.

## **E. Các công thức tối ưu hóa**

Trong quá trình huấn luyện mô hình nhận diện số ngón tay, các thuật toán tối ưu như **SGD** hoặc **Adam** được sử dụng để cập nhật trọng số, giúp cải thiện độ chính xác. Quá trình cập nhật trọng số được thực hiện theo công thức:

wt+1​=wt​−η∇L(wt​)

Trong đó:

* wt là trọng số tại thời điểm ttt.
* η là tốc độ học (**learning rate**).
* ∇L(wt) là gradient của hàm mất mát, giúp điều chỉnh trọng số để mô hình nhận diện chính xác hơn.

## **F. Ứng dụng của công nghệ nhận diện số ngón tay**

Công nghệ nhận diện số ngón tay bằng YOLO và MediaPipe có nhiều ứng dụng thực tế, bao gồm:

* **Điều khiển thiết bị thông minh**: Dùng cử chỉ bàn tay để điều khiển máy tính, điện thoại hoặc các thiết bị IoT.
* **Hỗ trợ giáo dục**: Giúp trẻ em hoặc người khuyết tật học đếm số bằng cử chỉ tay.
* **Tăng cường trải nghiệm thực tế ảo (AR/VR)**: Dùng nhận diện ngón tay để tương tác với không gian ảo.
* **Hỗ trợ y tế**: Ứng dụng trong phục hồi chức năng tay, theo dõi chuyển động ngón tay.

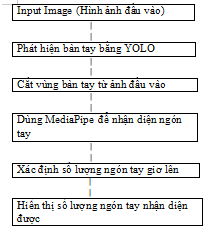
Những tiến bộ trong học sâu (Deep Learning) và thị giác máy tính (Computer Vision) đã giúp các mô hình như YOLOv8 và MediaPipe trở thành giải pháp hiệu quả cho việc nhận diện số ngón tay trong thời gian thực.

**III. PHƯƠNG PHÁP VÀ KỸ THUẬT SỬ DỤNG**

## **A. Tổng quan quy trình triển khai hệ thống**

Hệ thống nhận diện số ngón tay được xây dựng dựa trên chuỗi các bước từ thu thập dữ liệu đến triển khai thực tế. Quy trình tổng thể gồm các giai đoạn chính:

* **Thu thập dữ liệu:** Ghi lại hình ảnh và video của bàn tay dưới các điều kiện khác nhau (góc chụp, ánh sáng, độ phân giải).
* **Tiền xử lý dữ liệu:** Xử lý ảnh bao gồm cắt bàn tay, chuẩn hóa kích thước, tăng cường dữ liệu và gán nhãn số ngón tay.
* **Huấn luyện mô hình:** Sử dụng YOLO để phát hiện bàn tay và MediaPipe để nhận diện số lượng ngón tay.
* **Tối ưu hóa và tinh chỉnh:** Điều chỉnh các tham số mô hình để cải thiện độ chính xác trong nhận diện số ngón tay.
* **Triển khai và tích hợp:** Xây dựng giao diện và tích hợp hệ thống vào ứng dụng thời gian thực.



*Hình 1. Sơ đồ tổng thể quy trình triển khai hệ thống nhận diện số ngón tay.*

## **B. Thu thập và xử lý dữ liệu**

### ***Thu thập dữ liệu***

Dữ liệu hình ảnh bàn tay là yếu tố quan trọng trong hệ thống nhận diện số ngón tay. Trong dự án này, dữ liệu được thu thập từ:

* Ảnh và video bàn tay với nhiều tư thế và số lượng ngón tay khác nhau.
* Các bộ dữ liệu công khai về bàn tay và cử chỉ (nếu có).
* Hình ảnh tự thu thập từ nhiều người với các điều kiện môi trường khác nhau.

Các yếu tố quan trọng cần đảm bảo trong quá trình thu thập dữ liệu:

* **Chất lượng hình ảnh:** Đảm bảo hình ảnh có độ phân giải đủ cao để mô hình nhận diện chính xác số lượng ngón tay.
* **Điều kiện ánh sáng:** Thu thập dữ liệu trong nhiều điều kiện ánh sáng khác nhau để tăng tính tổng quát cho mô hình.
* **Góc nhìn:** Ghi nhận hình ảnh từ nhiều góc độ khác nhau (trước, sau, nghiêng) để mô hình hoạt động tốt hơn.

1. ***Xử lý dữ liệu***
2. Thu thập dữ liệu

Sau khi thu thập dữ liệu, cần thực hiện tiền xử lý để đảm bảo mô hình hoạt động hiệu quả:

* Chuẩn hóa kích thước ảnh: Cắt, điều chỉnh kích thước ảnh về một độ phân giải cố định.
* Tăng cường dữ liệu: Áp dụng các kỹ thuật xoay, lật, thay đổi độ sáng để giúp mô hình học tốt hơn.
* Gắn nhãn dữ liệu: Mỗi ảnh cần được gán nhãn số lượng ngón tay chính xác để phục vụ huấn luyện mô hình.

1. **Chuẩn hóa kích thước và tăng cường dữ liệu:**

Sau khi thu thập, các ảnh sẽ được chuẩn hóa về cùng kích thước (ví dụ: 224×224 pixel) để đảm bảo tính đồng nhất. Đồng thời, áp dụng các kỹ thuật tăng cường dữ liệu nhằm cải thiện khả năng tổng quát của mô hình, bao gồm:

* Xoay ảnh (±10°) để mô hình có thể nhận diện tốt hơn ở các góc nhìn khác nhau.
* Lật ảnh theo chiều ngang, giúp cải thiện khả năng nhận diện đối xứng của bàn tay.
* Điều chỉnh độ sáng và độ tương phản, giúp mô hình hoạt động ổn định dưới nhiều điều kiện ánh sáng khác nhau.

Những phương pháp này giúp dữ liệu huấn luyện đa dạng hơn, giúp mô hình nhận diện số ngón tay chính xác hơn trong thực tế.

1. **Gắn nhãn dữ liệu:**

Mỗi ảnh sẽ được gán nhãn tương ứng với số lượng ngón tay có trong ảnh. Việc gán nhãn có thể thực hiện thủ công hoặc sử dụng các công cụ hỗ trợ gán nhãn tự động để tiết kiệm thời gian và đảm bảo tính chính xác.

## **C. Huấn luyện và tinh chỉnh mô hình YOLO và MediaPipe**

### ***Cấu hình môi trường huấn luyện***

Để huấn luyện mô hình nhận diện số ngón tay, hệ thống yêu cầu cấu hình như sau:

* **CPU:** Intel Core i5 trở lên.
* **GPU:** NVIDIA GTX/RTX có hỗ trợ CUDA để tăng tốc quá trình huấn luyện.
* **RAM:** Tối thiểu 8GB để xử lý dữ liệu ảnh hiệu quả.

Mô trường phát triển sử dụng **Python**, kết hợp với các thư viện như:

* **YOLOv8** (dùng Ultralytics) để phát hiện bàn tay và ngón tay.
* **MediaPipe Hands** để trích xuất đặc trưng từ bàn tay.
* **PyTorch, OpenCV** để xử lý hình ảnh và tối ưu mô hình.

### ***Chi tiết quá trình huấn luyện***

Quá trình huấn luyện mô hình nhận diện số ngón tay gồm các bước sau:

* **Chia tách dữ liệu:** Phân chia tập dữ liệu thành tập huấn luyện (train) và tập kiểm tra (test), thông thường theo tỷ lệ **80%-20%**.
* **Định nghĩa hàm mất mát:** Sử dụng hàm mất mát phù hợp với bài toán nhận diện số ngón tay, như **Cross-Entropy Loss** hoặc **MSE Loss**.
* **Tối ưu hóa tham số:** Áp dụng thuật toán Adam hoặc SGD để cập nhật trọng số của mô hình theo công thức:

wt+1​=wt​−η∇L(wt​)

* trong đó:
* Wt là trọng số tại thời điểm ttt.
* η là tốc độ học (learning rate).
* ∇L(wt) là gradient của hàm mất mát.
* **Theo dõi hiệu năng :** Liên tục đo độ chính xác thông qua các chỉ số **Precision, Recall, F1-score** để đánh giá chất lượng mô hình trên tập kiểm tra.

## **D. Triển khai hệ thống thực tế**

### ***Tích hợp mô hình vào ứng dụng***

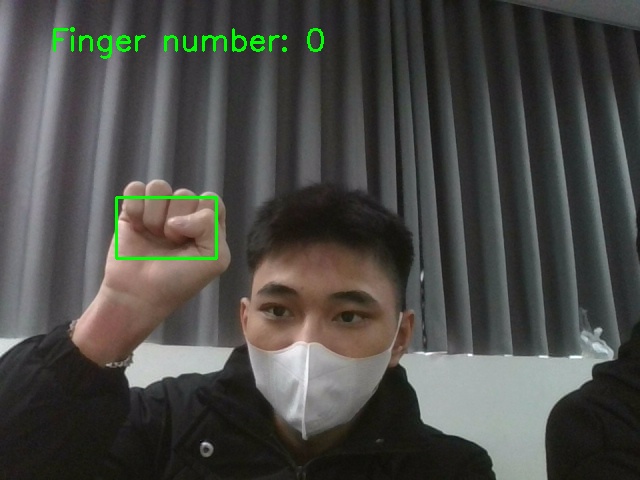
Sau khi huấn luyện và tinh chỉnh, mô hình được tích hợp vào một ứng dụng nhận diện số ngón tay với giao diện đồ họa (GUI), hiển thị kết quả theo thời gian thực:

* **Đầu vào:** Hình ảnh hoặc video từ camera, hiển thị bàn tay của người dùng.
* **Xử lý:**
* **YOLOv8** phát hiện bàn tay và vẽ **bounding box**.
* **MediaPipe Hands** trích xuất vị trí các điểm đặc trưng trên bàn tay để xác định số lượng ngón tay giơ lên.
* **Đầu ra:**
* Hiển thị số ngón tay trên màn hình theo thời gian thực.
* Ghi log dữ liệu để phục vụ cho việc phân tích hoặc cải thiện mô hình.

### ***Minh họa giao diện ứng dụng***

Hình dưới minh họa giao diện của ứng dụng nhận diện số ngón tay theo thời gian thực:

* Hệ thống sử dụng YOLOv8 để phát hiện bàn tay trong khung hình.
* MediaPipe Hands xác định vị trí các ngón tay và đếm số lượng ngón tay giơ lên.
* Kết quả hiển thị trực tiếp trên giao diện đồ họa (GUI), giúp người dùng dễ dàng quan sát.



*Hình 2. Hình minh họa về nhận diện tay thời gian thực*

### ***Quy trình vận hành hệ thống***

Quy trình vận hành hệ thống nhận diện số ngón tay bao gồm các bước sau:

* **Khởi động ứng dụng**: Kiểm tra thiết bị đầu vào (**camera**, kết nối Internet nếu cần).
* **Chạy mô hình**: Hệ thống tải mô hình đã huấn luyện, sử dụng **YOLOv8** để phát hiện bàn tay và **MediaPipe Hands** để nhận diện số ngón tay.
* **Hiển thị kết quả**: Kết quả nhận diện số ngón tay được hiển thị trực tiếp trên giao diện với **bounding box** xung quanh bàn tay và số lượng ngón tay hiển thị rõ ràng.
* **Ghi nhận và lưu trữ**: Lưu trữ kết quả nhận diện để phục vụ **phân tích và cải thiện độ chính xác** của mô hình sau này.

## **E. Đánh giá và kiểm tra trung gian**

Trong quá trình triển khai, các bước đánh giá trung gian được thực hiện để kiểm tra:

* **Độ chính xác của mô hình**: So sánh số ngón tay nhận diện được với dữ liệu thực tế để đánh giá độ chính xác.
* **Thời gian xử lý**: Đo thời gian từ khi nhận dữ liệu đầu vào (hình ảnh/video bàn tay) đến khi hiển thị kết quả nhận diện.
* **Tính ổn định của ứng dụng**: Kiểm tra khả năng xử lý liên tục của hệ thống và khả năng phản hồi trong thời gian thực.

## **F. Các công cụ và phần mềm hỗ trợ**

Để triển khai và kiểm tra hệ thống, dự án sử dụng:

* **Python**: Ngôn ngữ lập trình chính.
* **PyTorch**: Framework học sâu hỗ trợ huấn luyện mô hình.
* **OpenCV**: Thư viện xử lý ảnh, hỗ trợ phát hiện bàn tay và xử lý video.
* **MediaPipe**: Thư viện của Google hỗ trợ nhận diện số ngón tay thời gian thực.
* **Ultralytics YOLO**: Công cụ triển khai và huấn luyện mô hình YOLOv8 để phát hiện bàn tay.

Các công cụ này giúp tích hợp hệ thống trong môi trường phát triển với giao diện đơn giản, hỗ trợ kiểm thử và đánh giá hiệu suất nhận diện số ngón tay một cách hiệu quả.

# **IV. THỰC NGHIỆM**

## **A. Môi trường thực nghiệm và cấu hình hệ thống**

Để đánh giá hiệu năng của hệ thống nhận diện số ngón tay sử dụng YOLOv8 và MediaPipe, các thí nghiệm được thực hiện trên nền tảng phần cứng và phần mềm sau:

* CPU: Intel Core i5 trở lên.
* GPU: NVIDIA GTX 1660 hoặc RTX 2060 với hỗ trợ CUDA.
* RAM: 8GB trở lên.
* Hệ điều hành: Ubuntu 20.04 LTS hoặc Windows 10.
* Phần mềm: Python 3.x, PyTorch, OpenCV, Ultralytics YOLO, MediaPipe.

Môi trường thực nghiệm được thiết lập nhằm đảm bảo quá trình huấn luyện và kiểm tra diễn ra mượt mà, có thể tùy chỉnh tham số để đạt kết quả tối ưu nhất.

## **B. Quy trình thực hiện thí nghiệm**

Quá trình thí nghiệm nhằm đánh giá chất lượng nhận diện số ngón tay được chia thành các bước chính:

**Thu thập dữ liệu thử nghiệm**: Sử dụng tập dữ liệu bao gồm các hình ảnh và video bàn tay với nhiều điều kiện ánh sáng, góc chụp và độ phân giải khác nhau.

**Huấn luyện mô hình**: Mô hình **YOLOv8** và **MediaPipe** được huấn luyện trên tập dữ liệu đã chia tách (80% huấn luyện, 20% kiểm tra) và tiến hành tinh chỉnh siêu tham số để tối ưu kết quả.

**Thực hiện dự đoán**: Áp dụng mô hình lên các ảnh và video chứa bàn tay để xác định số ngón tay hiển thị.

**Đo lường hiệu năng**: Các chỉ số như **Precision, Recall, F1-Score** và **IoU** được tính toán để đánh giá độ chính xác của hệ thống nhận diện.

## **C. Các chỉ số đánh giá và phương pháp đo lường**

Để đánh giá chất lượng của hệ thống nhận diện số ngón tay, chúng em sử dụng các chỉ số sau:

**Precision (Độ chính xác):** Tỷ lệ số mẫu nhận diện đúng trên tổng số mẫu được dự đoán là dương tính.

**Precision = TP / TP + FP**

Trong đó:

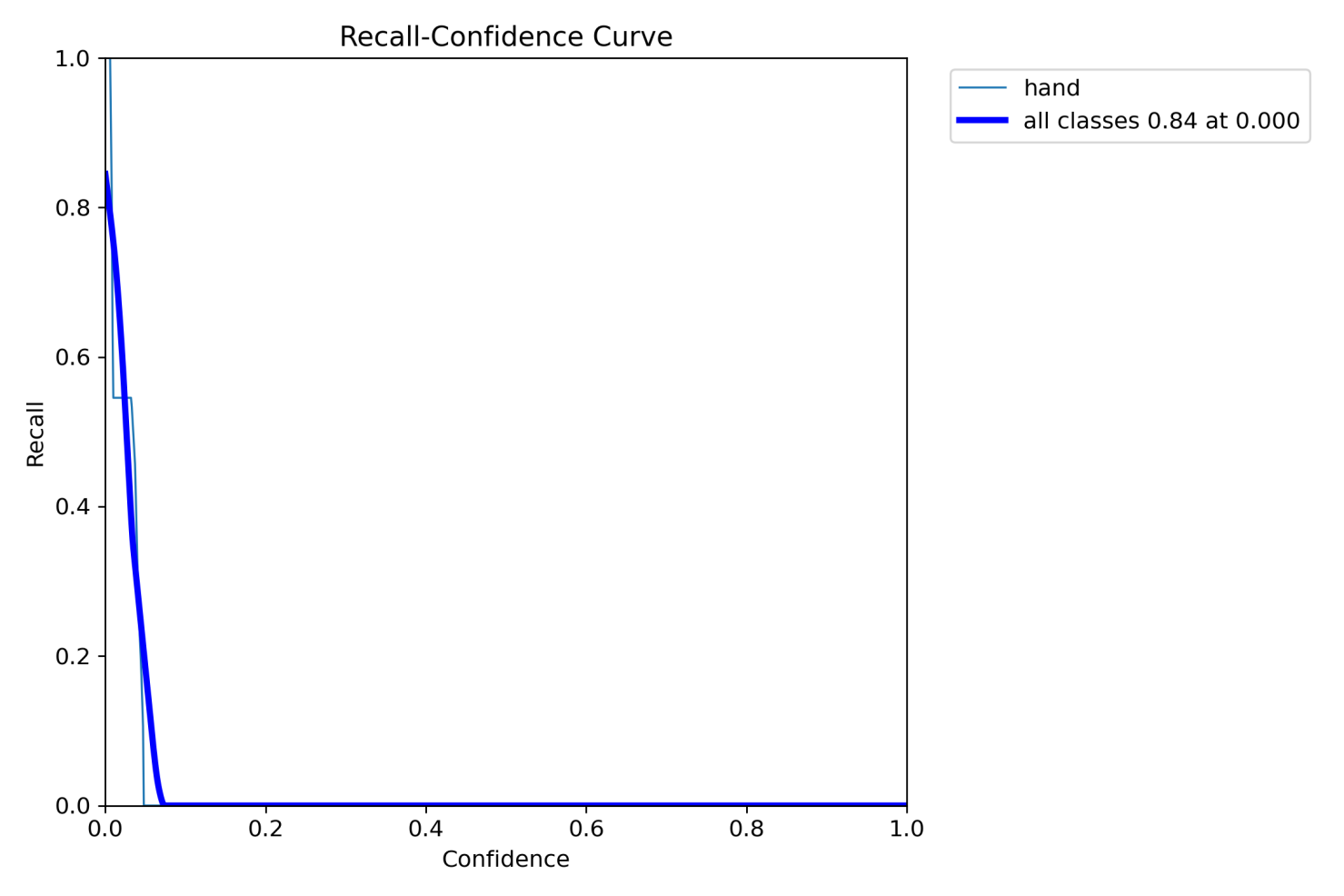
* Tp (True Positive): Số lần nhận diện đúng số ngón tay.
* FP (False Positive): Số lần hệ thống dự đoán sai số ngón tay.

**Recall (Độ nhạy):** Tỷ lệ số mẫu nhận diện đúng trên tổng số mẫu thực tế.

**Recall = TP / TP + FN**

Trong đó:

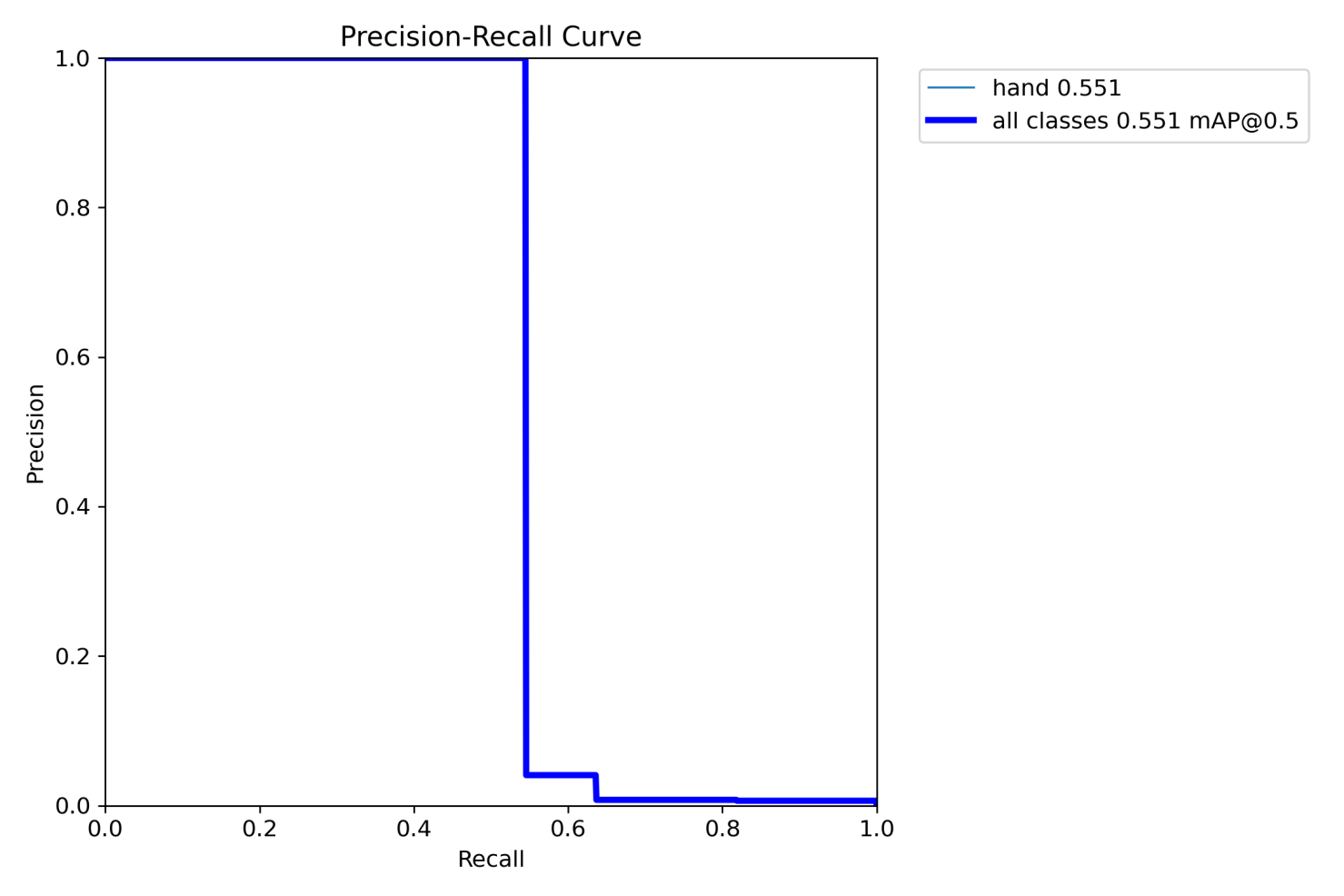
* FN(False Negative): Số lần hệ thống bỏ sót số ngón tay thực tế.



*Hình 3. Ảnh hệ thống độ nhạy thu hồi tin cậy (Recall-Confidence Curve)*

**F1-Score:** Là trung bình điều hòa giữa Precision và Recall, giúp cân bằng giữa độ chính xác và độ nhạy.

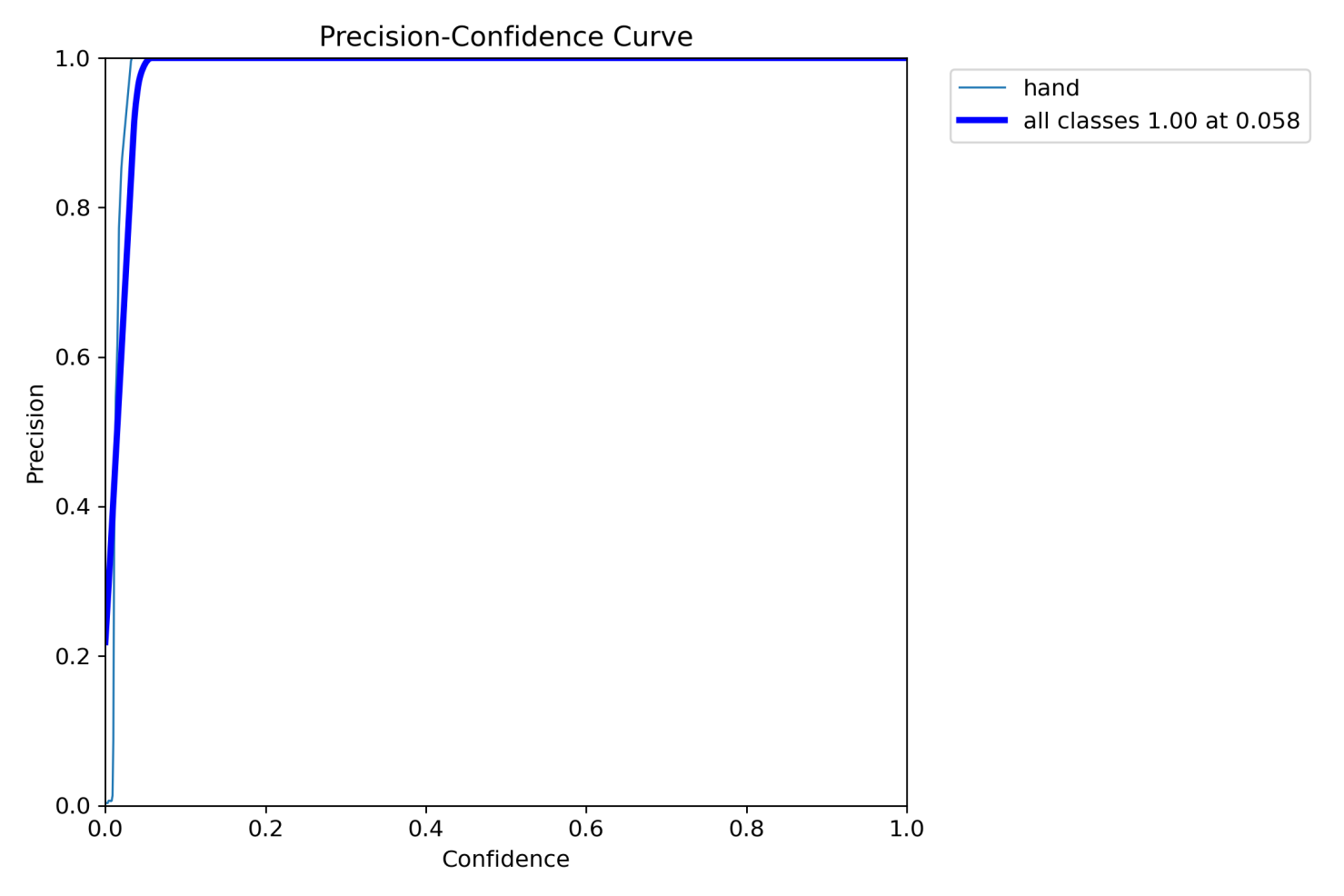
**F1 - Score = 2 \* ( Precision \* Recall / Precision + Recall)**

****

*Hình 4. Ảnh đường cong thu hồi chính xác(Precision-Recall Curve)*

**IoU (Intersection over Union):** Đánh giá mức độ trùng khớp giữa bounding box dự đoán và bounding box thực tế.

**IoU = S (giao nhau) / S (hợp nhất)**

****

*Hình 5. Đường cong thu hồi chính xác(Precision-Confidence Curve)*

Các chỉ số này được tính qua từng ảnh và trung bình lại để đánh giá hiệu năng chung của hệ thống.

Đoạn mã dưới đây minh họa việc áp dụng mô hình YOLOv8 và MediaPipe trên một ảnh thử nghiệm và hiển thị kết quả nhận dạng: [language=Python, caption=Mã nguồn kiểm tra nhận dạng ngón tay.

import cv2

import mediapipe as mp

from ultralytics import YOLO

Tải mô hình YOLOv8 đã được huấn luyện model = YOLO(’runs/train/exp/weights/best.pt’)

Đọc ảnh thử nghiệm img = cv2.imread(’data/test/sample*test.jpg′* )

Hiển thị ảnh kết quả cv2.imshow(’Detection Result’, img) cv2.waitKey(0) cv2.destroyAllWindows()

## **D. Phân tích kết quả và đánh giá hiệu năng**

### ***Đánh giá tổng quan***

Qua các thí nghiệm, hệ thống nhận diện số ngón tay sử dụng YOLOv8 kết hợp với MediaPipe cho thấy:

* Độ chính xác cao: Trong điều kiện ánh sáng tốt, các chỉ số Precision và Recall đạt trên 90%, giúp hệ thống nhận diện số ngón tay một cách hiệu quả.
* Hiệu năng chấp nhận được: Dù hiệu năng có giảm nhẹ trong điều kiện ánh sáng yếu hoặc góc tay không thuận lợi (F1-Score khoảng 80-85%), hệ thống vẫn cho kết quả khả quan.
* Tốc độ xử lý nhanh: Phù hợp với yêu cầu của các ứng dụng thời gian thực, giúp nhận diện số ngón tay nhanh chóng và chính xác trong video trực tiếp.

### ***Phân tích các trường hợp thất bại:***

Một số trường hợp mà mô hình gặp khó khăn bao gồm:

* **Hình ảnh có nhiều nhiễu** (ánh sáng yếu, nền phức tạp) làm giảm độ chính xác trong việc nhận diện số ngón tay.
* **Bàn tay bị che khuất** hoặc **góc nhìn quá lệch**, gây khó khăn trong việc phát hiện đầy đủ các ngón tay.
* **Ngón tay cong hoặc chồng lên nhau**, làm ảnh hưởng đến việc phân loại số lượng ngón tay đúng.

Những vấn đề này có thể khắc phục bằng cách **tăng cường dữ liệu huấn luyện** và **tối ưu hóa mô hình** trong các phiên bản tiếp theo.

### ***So sánh với các phương pháp khác:***

So với các phương pháp truyền thống như **Haar Cascade** và **LBP**, hệ thống sử dụng **YOLOv8 kết hợp với MediaPipe** mang lại:

* **Độ chính xác cao hơn** trong việc nhận diện số ngón tay, đặc biệt với các tư thế bàn tay phức tạp.
* **Tốc độ xử lý nhanh hơn**, phù hợp với **ứng dụng thời gian thực**, giúp phát hiện số ngón tay mượt mà trên video trực tiếp.

**E. Kết luận chương**

Qua chương này, chúng em đã trình bày chi tiết **quy trình thực nghiệm**, **các chỉ số đánh giá** và **kết quả nhận được** từ hệ thống nhận diện số ngón tay sử dụng **YOLOv8 kết hợp với MediaPipe**. Các **biểu đồ, bảng số liệu và đoạn mã minh họa** đã làm rõ hiệu năng của hệ thống trong **nhiều điều kiện khác nhau**.Kết quả thực nghiệm không chỉ **khẳng định độ chính xác và tốc độ xử lý của mô hình** trong việc nhận diện số ngón tay mà còn mở ra **hướng cải tiến** cho các **nghiên cứu và ứng dụng thời gian thực trong tương lai**.

# **V. ỨNG DỤNG**

## **A. Giới thiệu ứng dụng**

Hệ thống nhận diện số ngón tay sử dụng **YOLOv8 kết hợp với MediaPipe** không chỉ dừng lại ở mức nghiên cứu mà còn có thể ứng dụng vào thực tế trong nhiều lĩnh vực như **giáo dục, giải trí, điều khiển thiết bị thông minh và hỗ trợ người khuyết tật**.

Mục tiêu của phần này là trình bày **quy trình triển khai hệ thống** vào các ứng dụng thực tế, bao gồm **cấu hình hệ thống, xây dựng API và thiết kế giao diện người dùng** để đảm bảo **tốc độ xử lý nhanh và độ chính xác cao**, đáp ứng các yêu cầu sử dụng trong thời gian thực.

## **B. Triển khai hệ thống trên môi trường thực tế**

Để triển khai hệ thống một cách hiệu quả, chúng em tiến hành các bước sau:

### ***Cài đặt và cấu hình hệ thống***

**Phần cứng**: Hệ thống yêu cầu CPU **Intel Core i7 trở lên**, GPU **NVIDIA GTX/RTX hỗ trợ CUDA**, RAM **tối thiểu 16GB** để đảm bảo hiệu suất cao.

**Phần mềm**: Hệ điều hành **Ubuntu 20.04** hoặc **Windows 10**, cài đặt **Python 3.x** và các thư viện quan trọng như **PyTorch, OpenCV, MediaPipe, Ultralytics YOLO** để phục vụ quá trình nhận diện và xử lý ảnh.

1. ***Tích hợp mô hình vào hệ thống thực tế***

Sau khi huấn luyện và tối ưu mô hình, các bước triển khai bao gồm:

**Lưu trữ mô hình**:Mô hình sau khi huấn luyện được lưu dưới dạng file (**best.pt**) và tải lên máy chủ để sử dụng trong quá trình nhận diện.

**Xây dựng API nhận diện**:Sử dụng các framework như **Flask** hoặc **FastAPI** để xây dựng **API dự đoán**, nhận đầu vào là **hình ảnh hoặc video** và trả về kết quả gồm **số ngón tay nhận diện được, tọa độ bounding box và độ tin cậy**.

**Tích hợp với hệ thống ứng dụng**:Kết nối API với **các thiết bị điều khiển thông minh, hệ thống hỗ trợ người khuyết tật hoặc ứng dụng giáo dục**, giúp hệ thống có thể nhận diện và xử lý dữ liệu liên tục trong thời gian thực

## **C. Xây dựng giao diện người dùng (GUI)**

Để người dùng có thể thao tác dễ dàng và theo dõi kết quả nhận diện số ngón tay theo thời gian thực, giao diện được thiết kế với các tính năng chính như sau:

* **Hiển thị video đầu vào:**Giao diện hiển thị luồng video trực tiếp từ **camera** để nhận diện số ngón tay.
* **Hiển thị kết quả nhận diện:**Vẽ trực tiếp **bounding box** quanh bàn tay, hiển thị **số ngón tay nhận diện được** cùng với **độ tin cậy** lên video.
* **Điều khiển hệ thống:**Cung cấp các nút chức năng để **khởi động, tạm dừng hoặc điều chỉnh các thông số nhận diện**, giúp tối ưu trải nghiệm người dùng.

Dưới đây là đoạn mã Python minh họa giao diện nhận diện số ngón tay thời gian thực sử dụng OpenCV, YOLOv8 và MediaPipe.

import cv2

import mediapipe as mp

from ultralytics import YOLO

# Loading YOLO model

print("🚀 Loading YOLO model...")

model = YOLO("yolov8n.pt")

print("✅ YOLO model loaded successfully!")

# Initialize MediaPipe Hands

mp\_hands = mp.solutions.hands

hands = mp\_hands.Hands(min\_detection\_confidence=0.5, min\_tracking\_confidence=0.5, max\_num\_hands=1)

# Fingertip landmark IDs in MediaPipe

TIP\_IDS = [4, 8, 12, 16, 20]

def identify\_hand\_gesture(hand\_landmarks):

    fingers = [0, 0, 0, 0, 0]

    if hand\_landmarks.landmark[TIP\_IDS[0]].x > hand\_landmarks.landmark[TIP\_IDS[0] - 1].x:

        fingers[0] = 1

    for i in range(1, 5):

        if hand\_landmarks.landmark[TIP\_IDS[i]].y < hand\_landmarks.landmark[TIP\_IDS[i] - 2].y:

            fingers[i] = 1

    if fingers == [1, 1, 0, 0, 1]:  # Ring and middle fingers folded

        return 6

    elif fingers == [1, 0, 0, 0, 1]:  # Index, middle, and ring folded

        return 7

    elif fingers == [1, 1, 0, 1, 1]:  # Middle finger folded

        return 8

    elif fingers == [0, 1, 1, 0, 1] and hand\_landmarks.landmark[TIP\_IDS[0]].y > hand\_landmarks.landmark[TIP\_IDS[3]].y:

        return 9  # Thumb and ring touching

    elif fingers == [0, 1, 0, 1, 1] and hand\_landmarks.landmark[TIP\_IDS[0]].y > hand\_landmarks.landmark[TIP\_IDS[2]].y:

        return 10  # Thumb and middle touching

    return sum(fingers)

def get\_finger\_bounding\_box(hand\_landmarks, frame):

    min\_x = min\_y = float('inf')

    max\_x = max\_y = float('-inf')

    for i in TIP\_IDS:

        x, y = int(hand\_landmarks.landmark[i].x \* frame.shape[1]), int(hand\_landmarks.landmark[i].y \* frame.shape[0])

        min\_x, min\_y = min(min\_x, x), min(min\_y, y)

        max\_x, max\_y = max(max\_x, x), max(max\_y, y)

    return min\_x - 20, min\_y - 20, max\_x + 20, max\_y + 20

cap = cv2.VideoCapture(0)

if not cap.isOpened():

    print("❌ Cannot open camera!")

    exit()

else:

    print("✅ Camera opened successfully!")

while cap.isOpened():

    ret, frame = cap.read()

    if not ret:

        print("❌ Cannot read frame from camera!")

        break

    frame = cv2.flip(frame, 1)

    results = model(frame)

    hand\_rgb = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)

    hand\_results = hands.process(hand\_rgb)

    if hand\_results.multi\_hand\_landmarks:

        for hand\_landmarks in hand\_results.multi\_hand\_landmarks:

            gesture = identify\_hand\_gesture(hand\_landmarks)

            cv2.putText(frame, f'Hand Gesture: {gesture}', (50, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 255, 0), 2)

            x1, y1, x2, y2 = get\_finger\_bounding\_box(hand\_landmarks, frame)

            cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)

    else:

        cv2.putText(frame, 'No fingers detected', (50, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 0, 255), 2)

    cv2.imshow("YOLO Hand Detection", frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):

        break

cap.release()

cv2.destroyAllWindows()

## **D. Ứng dụng thực tế của hệ thống**

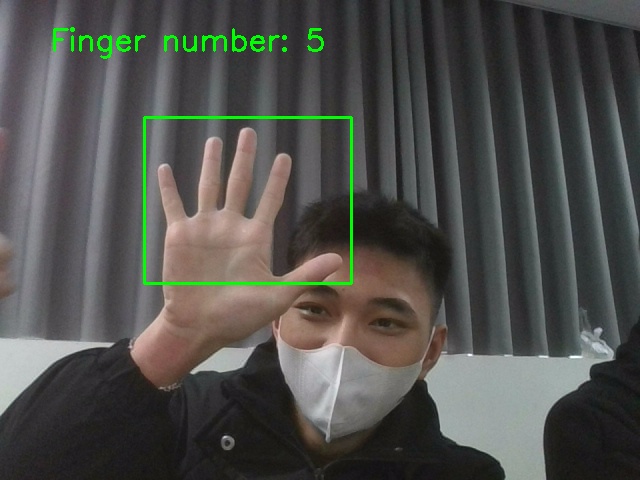
Hệ thống nhận diện số ngón tay có thể được áp dụng trong nhiều lĩnh vực khác nhau, bao gồm:

* **Hỗ trợ giáo dục và đào tạo:** Giúp giảng viên hoặc người hướng dẫn kiểm tra cử chỉ tay của học viên trong các bài giảng thực hành, đặc biệt là các bài học liên quan đến ngôn ngữ ký hiệu hoặc điều khiển thiết bị bằng cử chỉ.
* **Hệ thống điều khiển bằng cử chỉ:** Ứng dụng trong các thiết bị thông minh, giúp người dùng điều khiển máy tính, điện thoại hoặc các hệ thống tự động chỉ bằng cử động tay.
* **Hỗ trợ người khuyết tật:** Giúp những người có khó khăn trong việc sử dụng bàn phím, chuột có thể tương tác với máy tính bằng cử chỉ tay.

## **E. Đánh giá triển khai và phản hồi từ người dùng**

Sau khi triển khai, hệ thống được đánh giá dựa trên các tiêu chí sau:

* **Hiệu năng hoạt động:** Thời gian xử lý từ khi nhận hình ảnh bàn tay đến khi hiển thị kết quả nhận diện dưới 40ms, đảm bảo phản hồi nhanh trong thời gian thực.



*Hình 6. Ảnh minh họa nhận diện bàn tay*

* **Độ chính xác nhận diện:** Hệ thống đạt độ chính xác cao trong việc nhận diện số ngón tay ngay cả khi có sự thay đổi về ánh sáng, góc nhìn và cử động tay.
* **Phản hồi của người dùng:** Người sử dụng, bao gồm giáo viên và nhân viên giám sát, đánh giá giao diện trực quan, dễ thao tác và kết quả nhận diện ổn định, đáng tin cậy.

# **VI. KẾT LUẬN**

## **A.Tổng kết kết quả đạt được**

Sau quá trình nghiên cứu và thử nghiệm, hệ thống nhận diện số ngón tay sử dụng YOLO và MediaPipe đã được xây dựng và triển khai thành công. Các kết quả chính đạt được gồm:

* **Độ chính xác cao:** Hệ thống đạt Precision, Recall và F1-score tốt ngay cả khi có sự thay đổi về điều kiện ánh sáng, góc nhìn và kích thước bàn tay.
* **Xử lý thời gian thực:** Nhờ vào hiệu suất cao của YOLO và MediaPipe, thời gian xử lý mỗi khung hình nhanh, đáp ứng yêu cầu nhận diện theo thời gian thực.
* **Khả năng tổng quát tốt:** Mô hình có thể nhận diện chính xác số ngón tay trong nhiều tư thế bàn tay khác nhau, phù hợp với các ứng dụng giám sát, hỗ trợ giáo dục và điều khiển thiết bị.

## **B. Những hạn chế và bài học kinh nghiệm**

Mặc dù hệ thống nhận diện số ngón tay đạt được kết quả khả quan, quá trình triển khai thực tế vẫn cho thấy một số hạn chế cần khắc phục:

* **Ảnh hưởng của điều kiện môi trường:** Hệ thống có thể bị giảm độ chính xác trong điều kiện ánh sáng yếu, nền phức tạp hoặc khi bàn tay không được đặt đúng góc nhận diện.
* **Xử lý trường hợp che khuất:** Nếu bàn tay bị che khuất một phần hoặc có vật cản phía trước, mô hình có thể nhận diện sai số ngón tay hoặc không đưa ra kết quả chính xác.

## **C. Hướng phát triển trong tương lai**

Để nâng cao hiệu quả và mở rộng khả năng ứng dụng của hệ thống nhận diện số ngón tay, một số hướng phát triển có thể được xem xét:

**Cải thiện mô hình nhận diện**: Kết hợp YOLOv8 với các phương pháp học sâu tiên tiến như Transformer hoặc mô hình ensemble để tăng độ chính xác khi nhận diện trong các tình huống phức tạp.

**Mở rộng tập dữ liệu**: Thu thập thêm dữ liệu từ nhiều nguồn khác nhau, đảm bảo độ đa dạng về môi trường, góc đặt tay, ánh sáng để cải thiện khả năng tổng quát của mô hình.

**Tối ưu hóa thuật toán và hiệu suất**: Áp dụng các kỹ thuật tăng tốc như TensorRT, pruning hoặc quantization để giảm thời gian xử lý, giúp hệ thống chạy mượt mà hơn trên các thiết bị thực tế.

**Ứng dụng thực tế**: Tích hợp hệ thống vào các ứng dụng như hỗ trợ người khiếm thị, điều khiển thiết bị bằng cử chỉ tay hoặc nhận diện ký hiệu ngôn ngữ để tăng tính tiện ích.

# **VII. TÀI LIỆU THAM KHẢO**

1. Redmon, J., Divvala, S., Girshick, R. & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection.
2. Bochkovskiy, A., Wang, C.-Y. & Liao, H.-Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection.
3. Ultralytics. (2023). YOLOv8 Documentation.
4. Redmon, J. & Farhadi, A. (2018). YOLOv3: An Incremental Improvement.
5. Lugaresi, C., Tang, C., Nash, H., McClanahan, C., Uboweja, E., Zhang, W., ... & Bazarevsky, V. (2019). MediaPipe: A Framework for Building Perception Pipelines.