**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG – CƠ SỞ TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN 2**

****

**BÁO CÁO CUỐI KÌ MÔN THỊ GIÁC MÁY TÍNH**

**ĐỀ TÀI : SỬ DỤNG PHƯƠNG PHÁP TRÍCH XUẤT ĐẶC TRƯNG TỪ MEDIAPIPE VÀ THUẬT TOÁN RANDOM FOREST ĐỂ NHẬN DIỆN NGÔN NGỮ KÍ HIỆU**

**GIẢNG VIÊN HƯỚNG DẪN: THẦY LÊ MINH HÓA**

**SINH VIÊN THỰC HIỆN**

|  |  |
| --- | --- |
| Trần Thế Bảo | N21DCPT008 |
| Nguyễn Thị Diệu Linh | N21DCPT046 |

# **MỤC LỤC**

[**MỤC LỤC** 1](#_Toc198172753)

[**MỤC LỤC HÌNH ẢNH** 3](#_Toc198172754)

[**PHÂN CÔNG NHIỆM VỤ** 4](#_Toc198172755)

[**CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI** 5](#_Toc198172756)

[**1.1 Lý do chọn đề tài** 5](#_Toc198172757)

[**1.2 Mục đích, ý nghĩa của đề tài** 5](#_Toc198172758)

[**1.2.1 Mục đích của đề tài** 5](#_Toc198172759)

[**1.2.2 Ý nghĩa của đề tài** 5](#_Toc198172760)

[**1.3 Ngôn ngữ sử dụng : Python** 5](#_Toc198172761)

[**1.4 Công cụ sử dụng** 6](#_Toc198172762)

[**CHƯƠNG 2: CƠ SỞ LÝ THUYẾT** 7](#_Toc198172763)

[**2.1 Khái niệm Landmark** 7](#_Toc198172764)

[**2.1.1 Định nghĩa chung** 7](#_Toc198172765)

[**2.1.2**  **Landmark trong Mediapipe Hands** 7](#_Toc198172766)

[**2.1.3 Vai trò, ý nghĩa của Landmark** 8](#_Toc198172767)

[**2.1.4 Ưu và nhược điểm** 8](#_Toc198172768)

[**2.2 Framework MediaPipe** 8](#_Toc198172769)

[**2.2.1 Kiến trúc tổng thể** 9](#_Toc198172770)

[**2.2.2 Các module chính** 9](#_Toc198172771)

[**2.2.3 Ưu và nhược điểm** 10](#_Toc198172772)

[**2.3 Thư viện Pickle** 10](#_Toc198172773)

[**2.3.1 Công cụ chính** 10](#_Toc198172774)

[**2.3.2 Cách hoạt động** 10](#_Toc198172775)

[**2.3.3 Ưu và nhược điểm** 10](#_Toc198172776)

[**2.4 Thư viện Scikit-learn** 11](#_Toc198172777)

[**2.4.1 Thuật toán** 12](#_Toc198172778)

[**2.4.2 Ưu và nhược điểm** 12](#_Toc198172779)

[**CHƯƠNG 3: XỬ LÝ DỮ LIỆU ĐẦU VÀO** 13](#_Toc198172780)

[**3.1 Dữ liệu đầu vào của hệ thống** 13](#_Toc198172781)

[**3.2 Xử lý dữ liệu** 14](#_Toc198172782)

[**3.2.1 Dữ liệu đầu vào** 14](#_Toc198172783)

[**3.2.2 Dữ liệu đầu ra :** 15](#_Toc198172784)

[**3.3 Lưu trữ dữ liệu** 15](#_Toc198172785)

[**3.4 Code xử lý dữ liệu đầu vào** 15](#_Toc198172786)

[**CHƯƠNG 4: HUẤN LUYỆN MÔ HÌNH SỬ DỤNG THUẬT TOÁN RANDOM FOREST** 17](#_Toc198172787)

[**4.1 Thuật toán huấn luyện mô hình Random Forest** 17](#_Toc198172788)

[**4.1.1 Nguyên lý hoạt động** 17](#_Toc198172789)

[**4.1.2 Các đặc điểm nổi bật** 17](#_Toc198172790)

[**4.1.3 Ưu và nhược điểm** 18](#_Toc198172791)

[**4.2 Thực hiện huấn luyện mô hình** 18](#_Toc198172792)

[**4.2.1 Tìm siêu tham số tốt nhất** 18](#_Toc198172793)

[**4.2.2 Huấn luyện và lưu trữ mô hình** 20](#_Toc198172794)

[**CHƯƠNG 5: XÂY DỰNG CHƯƠNG TRÌNH NHẬN DIỆN NGÔN NGỮ KÍ HIỆU THEO THỜI GIAN THỰC** 22](#_Toc198172795)

[**5.1 Các bước thực hiện** 22](#_Toc198172796)

[**5.2 Code nhận diện ngôn ngữ kí hiệu theo thời gian thực** 22](#_Toc198172797)

[**5.3 Kết quả** 25](#_Toc198172798)

[**CHƯƠNG 6: KẾT LUẬN** 26](#_Toc198172799)

[**6.1 Ưu và nhược điểm** 26](#_Toc198172800)

[**6.2 Phương hướng phát triển** 26](#_Toc198172801)

# **MỤC LỤC HÌNH ẢNH**

[*Hình 1 Landmark trên bàn tay 7*](#_Toc198172833)

[*Hình 2 Ứng dụng của Mediapipe 9*](#_Toc198172834)

[*Hình 3 Thư viện Scikit-learn 11*](#_Toc198172835)

[*Hình 4 Dữ liệu đầu vào 13*](#_Toc198172836)

[*Hình 5 Vị trí các landmark 14*](#_Toc198172837)

[*Hình 6 file lưu trữ dữ liệu đầu vào đã xử lý 15*](#_Toc198172838)

[*Hình 7 Thuật toán Random Forest 17*](#_Toc198172839)

[*Hình 8 Kết quả của chương trình 25*](#_Toc198172840)

# **PHÂN CÔNG NHIỆM VỤ**

|  |  |
| --- | --- |
| Trần Thế Bảo | Nguyễn Thị Diệu Linh |
| -Thu thập và xử lý dữ liệu huấn luyện  -Train mô hình để tìm ra mô hình tối ưu  - Thu thập lý thuyết,tài liệu về thuật toán Random Forest  - Thực hiện việc nhận dạng chữ cái và hiển thị lên màn hình  -Hỗ trợ viết báo cáo | - Thu thập và xử lý dữ liệu huấn luyện  - Train mô hình để tìm ra mô hình tối ưu  - Thu thập lý thuyết, tài liệu về thuật toán trích xuất đặc trưng từ mediapipe  -Thực hiện việc ghép chữ thành câu và hiển thị lên màn hình  -Viết báo cáo |

# **CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI**

## **1.1 Lý do chọn đề tài**

Trong bối cảnh xã hội hiện đại, việc xây dựng môi trường giao tiếp bình đẳng cho người khiếm thính đang trở thành nhu cầu cấp thiết nhằm xóa bỏ rào cản ngôn ngữ và thúc đẩy hòa nhập cộng đồng. Các giải pháp hỗ trợ hiện nay, dù đã phát triển nhiều thiết bị và phần mềm nhận diện cử chỉ, vẫn còn gặp hạn chế về chi phí, tính di động và độ chính xác khi áp dụng trong thực tế. Mediapipe, với khả năng trích xuất tọa độ và đặc trưng hình học của bàn tay theo thời gian thực, kết hợp cùng thuật toán Random Forest mạnh mẽ trong phân lớp dữ liệu nhiều chiều và chống overfitting, được kỳ vọng sẽ khắc phục các nhược điểm này. Việc lựa chọn đề tài “Sử dụng phương pháp trích xuất đặc trưng từ Mediapipe và thuật toán Random Forest để nhận diện ngôn ngữ ký hiệu” xuất phát từ mong muốn tận dụng ưu thế của hai công nghệ để tạo ra hệ thống nhận diện hiệu quả, chi phí thấp và dễ dàng triển khai.

## **1.2 Mục đích, ý nghĩa của đề tài**

### **1.2.1 Mục đích của đề tài**

Mục đích của đề tài là xây dựng một hệ thống nhận diện ngôn ngữ ký hiệu theo thời gian thực, sử dụng Mediapipe để trích xuất chính xác các đặc trưng hình học của bàn tay và thuật toán Random Forest để phân lớp các ký hiệu, từ đó cung cấp giải pháp hỗ trợ giao tiếp hiệu quả, tiết kiệm chi phí và linh hoạt trong nhiều môi trường khác nhau. Hệ thống nhằm cải thiện đáng kể độ chính xác nhận diện so với các phương pháp truyền thống, đồng thời đảm bảo khả năng mở rộng cho việc bổ sung thêm ngôn ngữ ký hiệu hoặc tích hợp vào các thiết bị di động, ứng dụng trợ giúp học tập và giao tiếp dành cho người khiếm thính. Thông qua đó, đề tài kỳ vọng góp phần xóa bỏ rào cản ngôn ngữ, thúc đẩy hòa nhập xã hội và nâng cao chất lượng cuộc sống cho cộng đồng người khiếm thính.

### **1.2.2 Ý nghĩa của đề tài**

Về mặt khoa học, đề tài góp phần khẳng định tính khả thi khi kết hợp các kỹ thuật trích xuất đặc trưng hình học với thuật toán Random Forest trong bài toán nhận diện ký hiệu, từ đó gợi mở hướng nghiên cứu mới cho các bài toán xử lý ảnh và học máy. Về mặt thực tiễn, hệ thống được phát triển sẽ là công cụ đắc lực giúp người khiếm thính giao tiếp hiệu quả hơn, giảm thiểu rào cản ngôn ngữ và nâng cao chất lượng cuộc sống. Bên cạnh đó, giải pháp cũng có thể mở rộng ứng dụng trong các lĩnh vực như trợ lý ảo, giáo dục đặc biệt và tương tác người-máy, góp phần thúc đẩy bình đẳng và hòa nhập xã hội.

## **1.3 Ngôn ngữ sử dụng : Python**

Nhóm lựa chọn ngôn ngữ Python cho đề tài này vì vì đây là một ngôn ngữ lập trình mạnh mẽ, phổ biến trong lĩnh vực trí tuệ nhân tạo và học máy. Python không chỉ hỗ trợ tốt cho việc xử lý dữ liệu, huấn luyện mô hình mà còn tương thích tốt với thư viện Mediapipe – công cụ chính mà nhóm sử dụng để trích xuất đặc trưng từ bàn tay. Bên cạnh đó, Python có cú pháp đơn giản, dễ học, phù hợp với môi trường nghiên cứu và phát triển của sinh viên. Việc sử dụng Python giúp nhóm tiết kiệm thời gian, thuận lợi hơn trong việc thử nghiệm, đánh giá và cải tiến hệ thống.

Ưu điểm nổi bật của python :

* Hỗ trợ nhiều thư viện học máy như scikit-learn, numpy, pandas, rất cần thiết cho huấn luyện và xử lý dữ liệu.
* Tương thích tốt với thư viện Mediapipe để trích xuất đặc trưng từ video theo thời gian thực.
* Cộng đồng người dùng đông đảo, dễ tìm kiếm tài liệu, ví dụ và cách khắc phục lỗi.
* Cú pháp rõ ràng, dễ hiểu, giúp đẩy nhanh tiến độ phát triển và giảm lỗi lập trình.
* Hỗ trợ trực quan hóa dữ liệu với matplotlib, seaborn, giúp theo dõi và đánh giá kết quả mô hình dễ dàng.

## **1.4 Công cụ sử dụng**

PyCharm là một môi trường phát triển tích hợp (IDE) mạnh mẽ dành cho ngôn ngữ Python, được phát triển bởi JetBrains. Việc sử dụng PyCharm giúp nhóm làm việc hiệu quả hơn nhờ vào giao diện thân thiện, dễ sử dụng và các tính năng hỗ trợ lập trình hiện đại.

Ưu điểm nổi bật của pycharm:

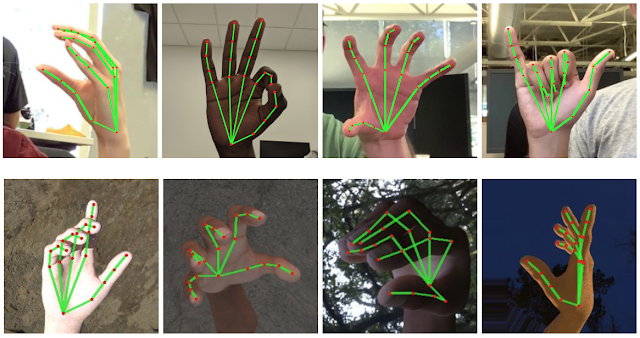
* Hỗ trợ quản lý mã nguồn thông minh: Gợi ý cú pháp, kiểm tra lỗi theo thời gian thực và tự động hoàn thành mã giúp tiết kiệm thời gian và giảm lỗi lập trình.
* Tích hợp môi trường ảo (virtual environment): Hỗ trợ dễ dàng cài đặt và quản lý các thư viện cần thiết cho dự án mà không ảnh hưởng đến hệ thống.
* Tích hợp Git: Hỗ trợ làm việc nhóm hiệu quả qua hệ thống quản lý mã nguồn Git, giúp theo dõi, đồng bộ và quản lý phiên bản code thuận tiện.
* Tích hợp công cụ debug mạnh mẽ: Giúp nhóm dễ dàng theo dõi luồng chạy của chương trình, kiểm tra giá trị biến và phát hiện lỗi trong quá trình xử lý.
* Hỗ trợ Jupyter Notebook và trực quan hóa dữ liệu: Phù hợp cho việc thử nghiệm và kiểm tra nhanh kết quả mô hình học máy.

# **CHƯƠNG 2: CƠ SỞ LÝ THUYẾT**

## **2.1 Khái niệm Landmark**

### **2.1.1 Định nghĩa chung**

Trong thị giác máy tính (computer vision), một landmark (điểm mốc hay keypoint) là một vị trí cố định, có ý nghĩa hình thái trên đối tượng cần phân tích. Ví dụ, đối với khuôn mặt có thể là góc mắt, đỉnh mũi; đối với bàn tay, đó là các khớp và đầu ngón.



Hình 1 Landmark trên bàn tay

### **2.1.2 Landmark trong Mediapipe Hands**

* MediaPipe Hands xác định 21 điểm landmark cho mỗi bàn tay, đánh số từ 0 đến 20 theo thứ tự cố định:
  + 0: cổ tay (wrist)
  + 1–4: khớp ngón cái (thumb) từ gốc đến đỉnh
  + 5–8: khớp ngón trỏ (index)
  + 9–12: khớp ngón giữa (middle)
  + 13–16: khớp ngón áp út (ring)
  + 17–20: khớp ngón út (pinky)
* Mỗi landmark có 3 trị số:
  + x, y: tọa độ chuẩn hóa (normalized) trong khung hình (∈ [0,1]), thể hiện vị trí ngang và dọc.
  + z: độ sâu tương đối so với mặt phẳng cổ tay (có thể âm nếu nằm trước mặt phẳng).

### **2.1.3 Vai trò, ý nghĩa của Landmark**

* Mô tả hình dạng: 21 landmark tạo thành “bộ xương” số học của bàn tay, giúp mô hình học máy hiểu được cấu trúc và tư thế.
* Tiền xử lý dữ liệu: Thay vì xử lý toàn bộ ảnh (hình ảnh có hàng triệu pixel), chúng ta chỉ cần vectơ hóa 21 điểm (x, y) => 42 giá trị. Việc này giảm chiều dữ liệu, giảm nhiễu nền và tăng tốc độ huấn luyện.
* Theo dõi động tác: Khi làm việc với video, thứ tự và biến thiên của các landmark qua các khung hình cho phép xác định chuyển động, cử chỉ liên tục.

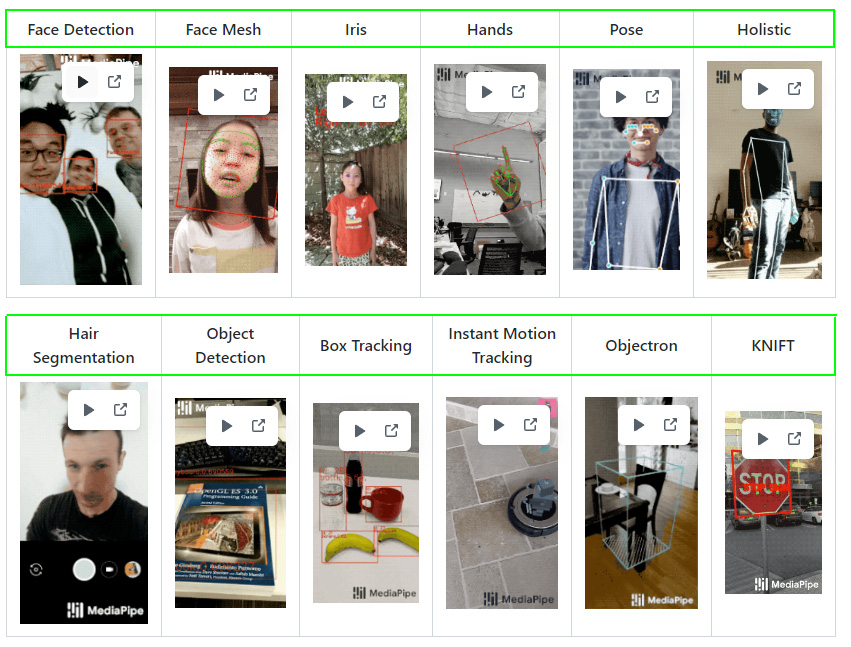
### **2.1.4 Ưu và nhược điểm**

* Ưu điểm:
  + Giảm đáng kể số chiều đầu vào so với xử lý ảnh gốc.
  + Ổn định với biến đổi ánh sáng, nền phức tạp.
  + Dễ kết hợp với các thuật toán học máy truyền thống (SVM, Random Forest, k‑NN…).
* Nhược điểm:
  + Mất thông tin “texture” và “màu sắc” (đôi khi cần thiết với một số bài toán).
  + Phụ thuộc độ chính xác của bước phát hiện and trích xuất landmark (nếu Mediapipe không tìm đủ 21 điểm, phải bỏ mẫu hoặc xử lý đặc biệt).

## **2.2 Framework MediaPipe**

Về tổng quan, [MediaPipe](https://github.com/google/mediapipe) là tập hợp của một loạt các giải pháp Machine Learning đa nền tảng, có thể can thiệp được và cực kỳ lightweight. Một số ưu điểm có thể kể tới của giải pháp này bao gồm:

* Cung cấp một giải pháp inference nhanh chóng: Google khẳng định rằng bộ công cụ này có thể chạy ổn định trên hầu hết các cấu hình phần cứng thông dụng.
* Dễ dàng cài đặt và triển khai: Việc cài đặt cực kỳ dễ dàng và tiện lợi, có thể triển khai trên nhiều nền tảng khác nhau như Mobile (Android/iOS), Desktop/Cloud, Web và IoT devices.
* Mã nguồn mở và miễn phí: Toàn bộ source code được công khai trên [MediaPipe](https://github.com/google/mediapipe), người dùng hoàn toàn có thể sử dụng và tùy chỉnh trực tiếp để phù hợp với bài toán của mình.



Hình 2 Ứng dụng của Mediapipe

### **2.2.1 Kiến trúc tổng thể**

* **Graph‑based Pipeline**:  
  Mediapipe xây dựng các “đồ thị” (graph) gồm các **nodes** (processor) và **tensors/packets** (dữ liệu) luân chuyển giữa các node. Mỗi node thực hiện một bước xử lý (ví dụ: decode ảnh, chạy inference mạng nơ‑ron, vẽ kết quả).
* **Calculator**: Mỗi node được gọi là một “calculator” – có thể viết bằng C++ hoặc Python – đảm nhận một tác vụ riêng biệt.
* **Data Flow**: Kết quả của calculators được đẩy dưới dạng timestamped packets, giúp xử lý video hiệu quả và đồng bộ.

### **2.2.2 Các module chính**

Mediapipe cung cấp sẵn các giải pháp (pre‑built graphs) cho nhiều bài toán:

* **Hands**: Phát hiện và trích xuất 21 landmark bàn tay.
* **Face Mesh**: Phát hiện 468 điểm landmark trên khuôn mặt.
* **Pose**: Theo dõi 33 điểm khớp cơ thể.
* **Holistic**: Kết hợp Face Mesh, Hands & Pose trong một graph.
* **Objectron**: Phát hiện và theo dõi đối tượng 3D (hộp 3D).
* **Selfie Segmentation**: Phân tách vùng chủ thể khỏi nền.

### **2.2.3 Ưu và nhược điểm**

* Ưu điểm
  + **Thời gian thực**: Tối ưu cho cả CPU lẫn GPU, chạy mượt trên nhiều thiết bị.
  + **Modular & Extensible**: Bạn có thể kết hợp các calculators hoặc tự viết ones mới để phục vụ bài toán riêng.
  + **Cross‑platform**: Hỗ trợ Android, iOS, Linux, Windows, macOS, WebAssembly.
* Nhược điểm
  + **Độ phức tạp**: Với graph lớn và nhiều calculators, việc debug có thể khó khăn.
  + **Kích thước thư viện**: Một số module nặng, cần cân nhắc khi deploy lên mobile.

## **2.3 Thư viện Pickle**

Pickle là module chuẩn của Python dùng để **tuần tự hóa** (serialization) và **khôi phục** (deserialization) các đối tượng Python, giúp lưu trữ hoặc truyền đạt chúng dưới dạng nhị phân.

### **2.3.1 Công cụ chính**

* **Lưu trữ đối tượng Python**: Bạn có thể lưu bất kỳ đối tượng Python nào (list, dict, class instance, mô hình học máy, v.v.) vào file để dùng lại sau này mà không phải tái tạo lại từ đầu.
* **Truyền dữ liệu giữa các tiến trình hoặc máy**: Nhị phân mà pickle tạo ra có thể gửi qua mạng, IPC, rồi unpickle ở đầu kia để tái tạo đối tượng.

### **2.3.2 Cách hoạt động**

* **Pickling** (tuần tự hóa): Chuyển đổi đối tượng Python thành chuỗi bytes.
* **Unpickling** (phục hồi): Từ chuỗi bytes, tái sinh lại đối tượng Python gốc.

### **2.3.3 Ưu và nhược điểm**

* Ưu điểm
  + **Dễ dùng**: Chỉ vài dòng code là đã lưu và load được.
  + **Hỗ trợ hầu hết các đối tượng Python**: Từ dữ liệu cơ bản đến class instance, mô hình scikit‑learn…
* Nhược điểm
  + **Bảo mật**: Không an toàn nếu unpickle dữ liệu từ nguồn không tin cậy, vì nó có thể thực thi code độc hại.
  + **Tương thích phiên bản**: File pickle tạo từ Python 3 có thể không unpickle trên Python 2, hoặc ngược lại.
  + **Kích thước file**: File pickle có thể lớn hơn so với định dạng JSON, CSV.

## **2.4 Thư viện Scikit-learn**

Scikit-learn (Sklearn) là thư viện mạnh mẽ nhất dành cho các thuật toán học máy được viết trên ngôn ngữ Python. Thư viện cung cấp một tập các công cụ xử lý các bài toán machine learning và statistical modeling gồm: classification, regression, clustering, và dimensionality reduction.

Thư viện được cấp phép bản quyền chuẩn FreeBSD và chạy được trên nhiều nền tảng Linux. Scikit-learn được sử dụng như một tài liệu để học tập.

Scikit-learn hỗ trợ mạnh mẽ trong việc xây dựng các sản phẩm. Nghĩa là thư viện này tập trung sâu trong việc xây dựng các yếu tố: dễ sử dụng, dễ code, dễ tham khảo, dễ làm việc, hiệu quả cao.

Mặc dù được viết cho Python nhưng thực ra các thư viện nền tảng của scikit-learn lại được viết dưới các thư viện của C để tăng hiệu suất làm việc. Ví dụ như: Numpy(Tính toán ma trận), LAPACK, LibSVM và Cython.



Hình 3 Thư viện Scikit-learn

### **2.4.1 Thuật toán**

Thư viện tập trung vào việc mô hình hóa dữ liệu. Nó không tập trung vào việc truyền tải dữ liệu, biến đổi hay tổng hợp dữ liệu. Những công việc này dành cho thư viện Numpy và Pandas.

Sau đây là một số nhóm thuật toán được xây dựng bởi thư viện scikit-learn:

* Clustering: Nhóm thuật toán Phân cụm dữ liệu không gán nhãn. Ví dụ thuật toán KMeans
* Cross Validation: Kiểm thử chéo, đánh giá độ hiệu quả của thuật toán học giám sát sử dụng dữ liệu kiểm thử (validation data) trong quá trình huấn luyện mô hình.
* Datasets: Gồm nhóm các Bộ dữ liệu được tích hợp sẵn trong thư viện. Hầu như các bộ dữ liệu đều đã được chuẩn hóa và mang lại hiêu suất cao trong quá trình huấn luyện như iris, digit, ...
* Dimensionality Reduction: Mục đích của thuật toán này là để Giảm số lượng thuộc tính quan trọng của dữ liệu bằng các phương pháp như tổng hợp, biểu diễn dữ liệu và lựa chọn đặc trưng. Ví dụ thuật toán PCA (Principal component analysis).
* Ensemble methods: Các Phương pháp tập hợp sử dụng nhiều thuật toán học tập để có được hiệu suất dự đoán tốt hơn so với bất kỳ thuật toán học cấu thành nào.
* Feature extraction: Trích xuất đặc trưng. Mục đích là để định nghĩa các thuộc tình với dữ liệu hình ảnh và dữ liệu ngôn ngữ.
* Feature selection: Trích chọn đặc trưng. Lựa chọn các đặc trưng có ý nghĩa trong việc huấn luyện mô hình học giám sát.
* Parameter Tuning: Tinh chỉnh tham số. Các thuật toán phục vụ việc lựa chọn tham số phù hợp để tối ưu hóa mô hình.
* Manifold Learning: Các thuật toán học tổng hợp và Phân tích dữ liệu đa chiều phức tạp.
* Supervised Models: Học giám sát. Mảng lớn các thuật toán học máy hiện nay. Ví dụ như linear models, discriminate analysis, naive bayes, lazy methods, neural networks, support vector machines và decision trees.

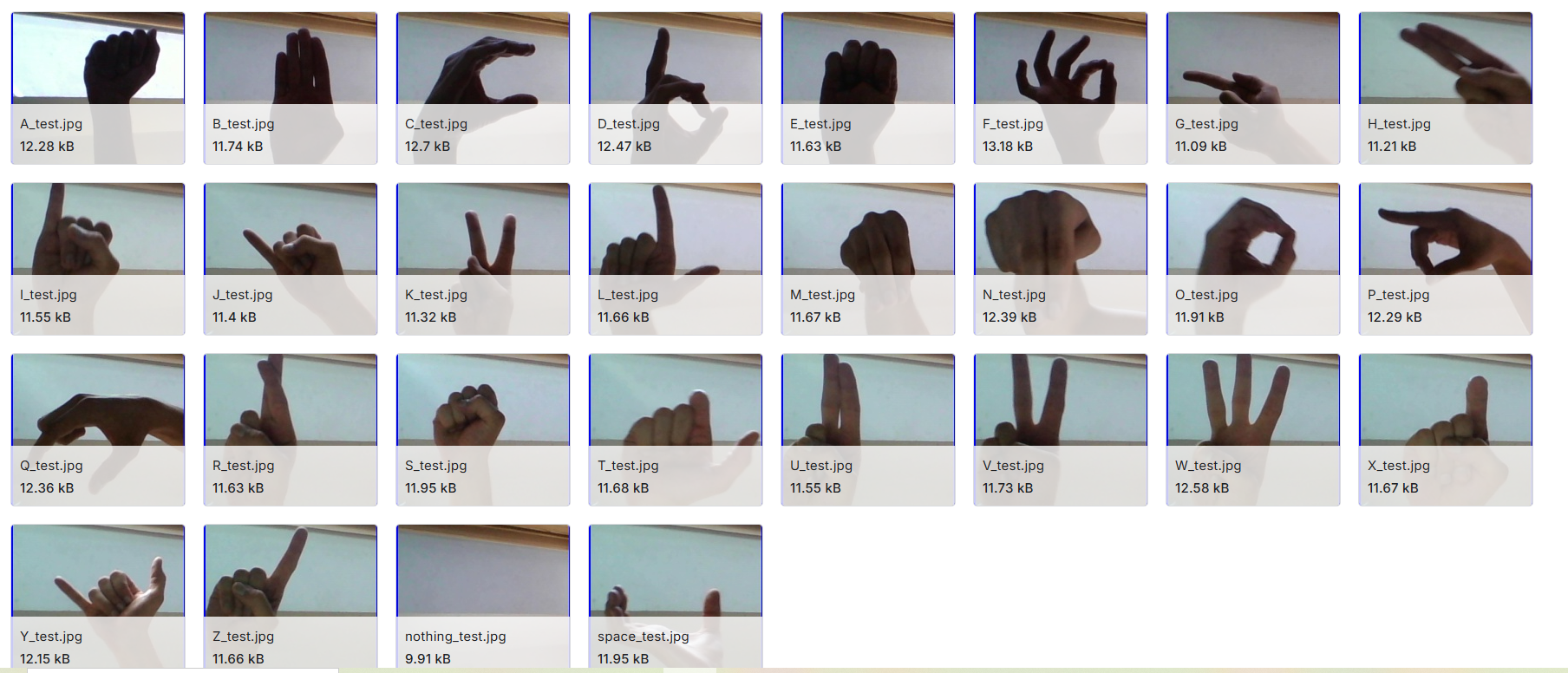
### **2.4.2 Ưu và nhược điểm**

* **Ưu điểm**
  + Giao diện đồng nhất, dễ học.
  + Đa dạng thuật toán cho hầu hết các bài toán ML cơ bản.
  + Tài liệu phong phú, cộng đồng lớn.
* **Nhược điểm**
  + Không tối ưu cho deep learning (phải dùng TensorFlow/PyTorch).
  + Không hỗ trợ tốt dữ liệu rất lớn (cần Spark hoặc Dask để scale).
  + Một số thuật toán thế hệ cũ, chưa có các cải tiến mới nhất.

# **CHƯƠNG 3: XỬ LÝ DỮ LIỆU ĐẦU VÀO**

## **3.1 Dữ liệu đầu vào của hệ thống**

Dữ liệu đầu vào của hệ thống là hình ảnh bàn tay thể hiện các ký hiệu ngôn ngữ tay tương ứng với các chữ cái và số. Nhằm huấn luyện mô hình nhận diện ngôn ngữ ký hiệu tay, nhóm đã tiến hành thu thập dữ liệu bằng cách sử dụng webcam tích hợp trên máy tính + dữ liệu tìm kiếm có sẵn và thư viện Mediapipe Hands để trích xuất các điểm đặc trưng (landmark) của bàn tay.

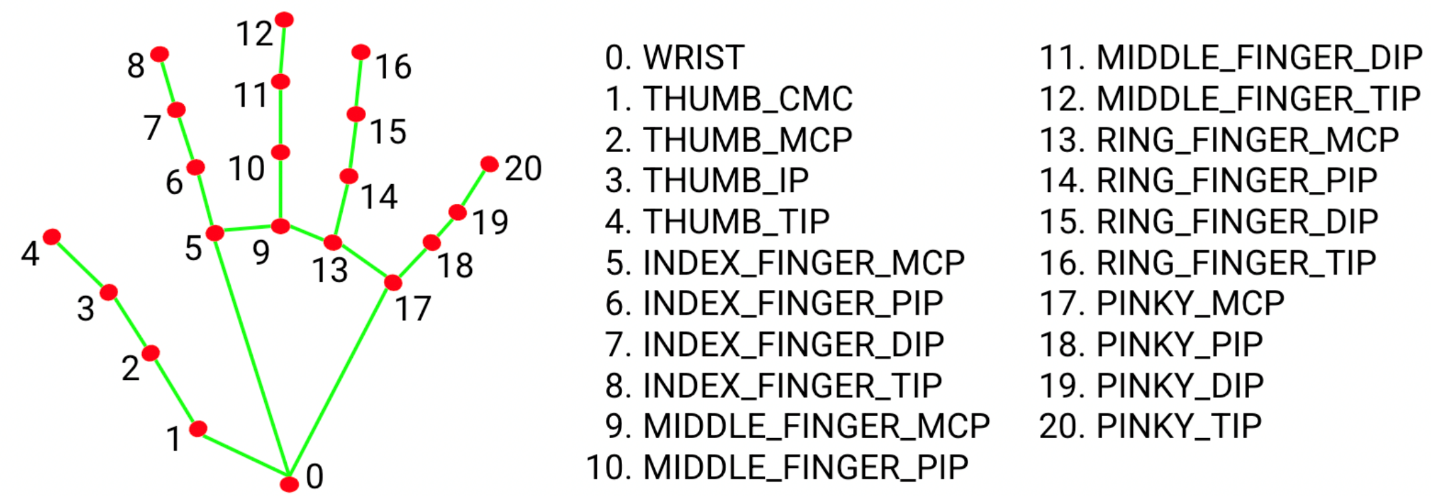


Hình 4 Dữ liệu đầu vào

Cụ thể, với mỗi ký hiệu ngôn ngữ tay (bao gồm các chữ cái và số được chọn), nhóm đã tiến hành thu thập 1.500 ảnh biểu diễn cho mỗi lớp (class). Mỗi ảnh được chụp theo nhiều góc độ và trạng thái khác nhau nhằm tăng độ đa dạng và độ chính xác cho mô hình sau này. Tổng số lượng ảnh thu thập được như sau:

* Số lượng lớp (chữ cái/số): 27 ký hiệu
* Số lượng ảnh mỗi lớp: 1.500 ảnh
* Tổng số ảnh dữ liệu: 27 × 1.500 = 40.500 ảnh

Mỗi ảnh đầu vào sau khi thu thập sẽ được xử lý bằng Mediapipe để trích xuất ra 21 điểm landmark của bàn tay, tương ứng với 21 khớp tay quan trọng.



Hình 5 Vị trí các landmark

## **3.2 Xử lý dữ liệu**

Mỗi landmark là một đối tượng có 3 giá trị tọa độ :

x – vị trí theo chiều ngang

y – vị trí theo chiều dọc

z - độ sâu (trong hệ thống này nhóm không dùng giá trị tọa độ z)

Quy trình thực hiện :

* Đọc ảnh từ thư mục
* Trích xuất landmark bằng Mediapipe
* Tạo vector đặc trưng
* Lưu trữ dữ liệu vào bộ nhớ tạm
* Lưu toàn bộ dữ liệu vào file pickle

### **3.2.1 Dữ liệu đầu vào**

* Mỗi ảnh chứa một bàn tay biểu diễn ký hiệu ngôn ngữ tay.
* Mediapipe Hands sẽ phát hiện 21 điểm landmark trên bàn tay.
* Mỗi điểm landmark được biểu diễn bằng 2 tọa độ chuẩn hóa (x, y) trong khoảng [0, 1], tương ứng với vị trí của điểm đó trên ảnh.
* Như vậy, một ảnh sẽ được biểu diễn dưới dạng vector đặc trưng gồm 42 giá trị (21 điểm × 2 tọa độ).

Ví dụ:

Trạng thái ảnh được biểu diễn như sau:  
[x₀, y₀, x₁, y₁, ..., x₂₀, y₂₀]  
Trong đó xᵢ, yᵢ là tọa độ của điểm landmark thứ i.

### **3.2.2 Dữ liệu đầu ra :**

* Mỗi vector đặc trưng đầu vào sẽ gắn với một nhãn (label) tương ứng là ký hiệu ngôn ngữ tay được biểu diễn (ví dụ: 'A', 'B', ..., 'Z').
* Nhãn này sẽ là giá trị đầu ra cần dự đoán của mô hình học máy.

Ví dụ:

* Input: [0.71, 0.54, 0.68, 0.49, ..., 0.45, 0.52]
* Output: 'A'

## **3.3 Lưu trữ dữ liệu**

Sau khi hoàn tất quá trình trích xuất đặc trưng từ các ảnh đầu vào, dữ liệu được lưu trữ dưới dạng nhị phân bằng thư viện pickle của Python. Việc lưu trữ dưới định dạng nhị phân giúp đảm bảo tính toàn vẹn dữ liệu và thuận tiện cho việc tái sử dụng trong giai đoạn huấn luyện mô hình.

Cụ thể, dữ liệu được lưu dưới dạng một cấu trúc từ điển (dictionary) với hai thành phần chính:

* data: danh sách các vector đặc trưng được trích xuất từ hình ảnh. Mỗi vector chứa 42 giá trị tương ứng với tọa độ (x, y) của 21 điểm khớp trên bàn tay do Mediapipe cung cấp.
* labels: danh sách các nhãn (label) tương ứng với mỗi ảnh, được lấy theo tên thư mục chứa ảnh đó. Mỗi nhãn đại diện cho một ký hiệu ngôn ngữ tay cụ thể.

Cuối cùng, toàn bộ dictionary này được ghi vào file data27Sign2.pickle bằng chế độ ghi nhị phân ('wb'). File .pickle đóng vai trò như một kho dữ liệu đầu vào phục vụ cho việc huấn luyện mô hình học máy ở các bước tiếp theo.



Hình 6 file lưu trữ dữ liệu đầu vào đã xử lý

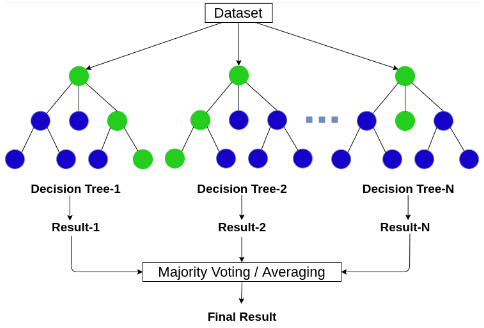
## **3.4 Code xử lý dữ liệu đầu vào**

import os  
  
import pickle  
import mediapipe as mp  
import cv2  
import matplotlib.pyplot as plt  
  
mp\_hands = mp.solutions.hands  
  
mp\_drawing = mp.solutions.drawing\_utils  
  
mp\_drawing\_styles = mp.solutions.drawing\_styles  
  
hands = mp\_hands.Hands(static\_image\_mode=True, min\_detection\_confidence=0.3)  
  
DATA\_DIR = './data27Sign2'  
  
data = []  
labels = []  
  
for dir\_ in os.listdir(DATA\_DIR):  
  
 for img\_path in os.listdir(os.path.join(DATA\_DIR, dir\_)):  
 data\_aux = []  
  
 img = cv2.imread(os.path.join(DATA\_DIR, dir\_, img\_path))  
 img\_rgb = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)  
  
 results = hands.process(img\_rgb)  
  
 if results.multi\_hand\_landmarks:  
 for hand\_landmarks in results.multi\_hand\_landmarks:  
  
 print(hand\_landmarks)  
  
 for i in range(len(hand\_landmarks.landmark)):  
 x = hand\_landmarks.landmark[i].x  
 y = hand\_landmarks.landmark[i].y  
 data\_aux.append(x)  
 data\_aux.append(y)  
  
 if len(data\_aux) == 42:  
 data.append(data\_aux)  
 labels.append(dir\_)  
 else:  
 print(f"Bỏ ảnh {img\_path} trong {dir\_} vì chỉ có {len(data\_aux)} điểm")  
  
print("dang xu ly")  
f = open('data27Sign2.pickle', 'wb')  
print("da luu")  
  
pickle.dump({'data': data, 'labels': labels}, f)  
  
f.close()

# **CHƯƠNG 4: HUẤN LUYỆN MÔ HÌNH SỬ DỤNG THUẬT TOÁN RANDOM FOREST**

## **4.1 Thuật toán huấn luyện mô hình Random Forest**

Random Forest là một thuật toán học máy thuộc nhóm học có giám sát (supervised learning) và được sử dụng phổ biến trong các bài toán phân loại (classification) và hồi quy (regression). Thuật toán này là một dạng của tập hợp học (ensemble learning), nơi mà nhiều mô hình yếu (weak learners), cụ thể là các cây quyết định (decision trees), được kết hợp lại để tạo thành một mô hình mạnh mẽ hơn.



Hình 7 Thuật toán Random Forest

### **4.1.1 Nguyên lý hoạt động**

Thuật toán Random Forest xây dựng một tập hợp các cây quyết định (decision trees) độc lập. Mỗi cây được huấn luyện trên một tập dữ liệu con (sample) được chọn ngẫu nhiên từ tập dữ liệu gốc bằng kỹ thuật bagging (Bootstrap Aggregating). Khi đưa vào một mẫu mới, mỗi cây trong rừng sẽ đưa ra một dự đoán, và kết quả cuối cùng là kết quả được biểu quyết (voting) bởi đa số (trong bài toán phân loại) hoặc trung bình cộng (trong bài toán hồi quy).

### **4.1.2 Các đặc điểm nổi bật**

* **Đa dạng hóa mô hình**: Nhờ cả bagging và chọn ngẫu nhiên đặc trưng, mỗi cây trong rừng có xu hướng học những khía cạnh khác nhau của dữ liệu, mang lại khả năng kháng nhiễu cao.
* **Chống overfitting**: Việc kết hợp kết quả từ nhiều cây giảm thiểu nguy cơ quá khớp so với một cây quyết định đơn lẻ.
* **Xử lý tốt dữ liệu phức tạp**: Random Forest có thể hoạt động hiệu quả với bộ dữ liệu đa chiều, chứa nhiều nhiễu hoặc thiếu giá trị mà không cần chuẩn hóa trước.
* **Đánh giá tầm quan trọng của đặc trưng**: Thuật toán cung cấp chỉ số “feature importance”, cho biết đặc trưng nào đóng góp nhiều nhất vào khả năng phân biệt, hỗ trợ việc giảm chiều và tối ưu hóa mô hình.

### **4.1.3 Ưu và nhược điểm**

* Ưu điểm của Random Forest
  + Độ chính xác và ổn định cao: Kết hợp kết quả từ nhiều cây giúp mô hình ít bị dao động trước dữ liệu mới.
  + Không yêu cầu nhiều giả thiết về dữ liệu: Không cần giả định tuyến tính, không yêu cầu chuẩn hóa hay chuẩn hóa biến.
  + Khả năng xử lý dữ liệu thiếu và nhiễu tốt: Do mỗi cây chỉ xem xét một phần dữ liệu và đặc trưng, nên ảnh hưởng của các giá trị bất thường sẽ được chia nhỏ.
  + Song song hóa: Các cây được huấn luyện độc lập, dễ dàng tận dụng đa lõi CPU hoặc môi trường phân tán để tăng tốc độ
* Nhược điểm của Random Forest
  + Khó diễn giải: So với một cây quyết định đơn, rừng ngẫu nhiên là “hộp đen” và không dễ dàng giải thích chi tiết quyết định của mô hình.
  + Tốc độ dự đoán và huấn luyện: Khi số lượng cây (BBB) và độ sâu cây tăng lên, thời gian huấn luyện và dự đoán có thể trở nên chậm, đòi hỏi tài nguyên tính toán lớn.
  + Kích thước mô hình lớn: Lưu trữ hàng trăm hoặc hàng ngàn cây có thể tiêu tốn bộ nhớ, không phù hợp với các thiết bị có giới hạn tài nguyên.
  + Không lý tưởng cho thời gian thực nghiêm ngặt: Trong ứng dụng yêu cầu phản hồi tức thì (real-time), độ trễ do tính toán rừng cây có thể là vấn đề.

## **4.2 Thực hiện huấn luyện mô hình**

### **4.2.1 Tìm siêu tham số tốt nhất**

Trong file OptimizeModel.py, chương trình thực hiện các bước sau một cách tự động để tìm ra bộ tham số tốt nhất cho mô hình Random Forest:

* Đầu tiên, tải dữ liệu đặc trưng và nhãn từ file data27Sign.pickle và chuyển thành mảng NumPy.
* Sau đó, chia dữ liệu thành hai phần: 80% để huấn luyện và 20% để kiểm tra, đồng thời xáo trộn và giữ nguyên tỷ lệ các lớp giống nhau ở cả hai phần.
* Nhóm các tham số cần thử — như số cây rừng (n\_estimators), độ sâu tối đa của mỗi cây (max\_depth), và số đặc trưng xét tại mỗi lần chia nhánh (max\_features) — được đặt trong một “lưới” (param\_grid).
* Dùng GridSearchCV với 6-fold cross-validation: chương trình sẽ thử từng kết hợp tham số trong lưới, mỗi lần ẩn một phần dữ liệu để kiểm tra, rồi quay lại huấn luyện trên phần còn lại. Quá trình này chạy song song trên nhiều luồng CPU để đẩy nhanh tốc độ.
* Cuối cùng, sau khi thử hết mọi kết hợp, chương trình in ra bộ tham số cho kết quả chính xác cao nhất (best\_params\_), sẵn sàng để sử dụng cho file huấn luyện mô hình

from sklearn.ensemble import RandomForestClassifier  
from sklearn.model\_selection import GridSearchCV  
  
import pickle  
  
import numpy as np  
from sklearn.model\_selection import train\_test\_split  
import os  
  
os.environ["OMP\_NUM\_THREADS"] = "8" # Dùng 8 luồng CPU  
# Danh sách siêu tham số để thử nghiệm có giải thích trong note  
param\_grid = {  
 'n\_estimators': [100, 200, 300],  
 'max\_depth': [10, 20, 30],  
 'min\_samples\_split': [2, 5, 10],  
 'min\_samples\_leaf': [1, 2, 5],  
 'max\_features': ['sqrt', 'log2']  
}  
  
data\_dictionary = pickle.load(open('./data27Sign.pickle', 'rb'))  
  
# convert từ list sang array để hỗ trợ tính toán ma trận, vector  
data = np.asarray(data\_dictionary['data'])  
label = np.asarray(data\_dictionary['labels'])  
  
x\_train, x\_test, y\_train, y\_test = train\_test\_split(data, label,  
 test\_size=0.2,  
 shuffle=True,  
 stratify=label)  
  
# Khởi tạo mô hình Random Forest  
rf = RandomForestClassifier()  
  
grid\_search = GridSearchCV(rf, param\_grid, cv=6, scoring='accuracy', n\_jobs=-1)  
grid\_search.fit(x\_train, y\_train)  
  
print("Best parameters:", grid\_search.best\_params\_)

### **4.2.2 Huấn luyện và lưu trữ mô hình**

Trong file train\_classifier.py, chương trình thực hiện các bước sau để huấn luyện và lưu trữ mô hình Random Forest:

1. Đầu tiên, chương trình tải dữ liệu đặc trưng và nhãn từ file data27Sign.pickle, sau đó chuyển chúng thành mảng NumPy để thuận tiện cho việc xử lý.
2. Tiếp theo, dữ liệu được chia thành hai phần: 80% dùng để huấn luyện và 20% dùng để kiểm tra. Việc chia được thực hiện ngẫu nhiên (shuffle=True) và đảm bảo giữ nguyên tỷ lệ phân bố các lớp (stratify=label) nhằm tránh hiện tượng mất cân bằng dữ liệu giữa hai tập.
3. Mô hình RandomForestClassifier được khởi tạo với các tham số tối ưu đã được xác định trước đó từ file OptimizeModel.py, bao gồm: 300 cây (n\_estimators), độ sâu tối đa 30, chia nhánh khi có ít nhất 5 mẫu (min\_samples\_split), mỗi nhánh lá có ít nhất 2 mẫu (min\_samples\_leaf), và chọn căn bậc hai số đặc trưng tại mỗi lần chia (max\_features='sqrt'). Tham số random\_state=42 giúp giữ kết quả nhất quán giữa các lần chạy, còn n\_jobs=-1 giúp tận dụng toàn bộ CPU để tăng tốc huấn luyện.
4. Sau đó, chương trình huấn luyện mô hình với dữ liệu huấn luyện thông qua model.fit(x\_train, y\_train) và sử dụng tập kiểm tra để đánh giá độ chính xác bằng accuracy\_score.
5. Cuối cùng, mô hình sau khi huấn luyện được lưu vào file model27SignV1.p dưới dạng nhị phân bằng pickle.dump() để có thể tái sử dụng sau này mà không cần huấn luyện lại.

import pickle  
  
import numpy as np  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.model\_selection import train\_test\_split  
from sklearn.metrics import accuracy\_score  
  
data\_dictionary = pickle.load(open('./data27Sign.pickle', 'rb'))  
  
# convert từ list sang array để hỗ trợ tính toán ma trận, vector  
data = np.asarray(data\_dictionary['data'])  
label = np.asarray(data\_dictionary['labels'])  
  
x\_train, x\_test, y\_train, y\_test = train\_test\_split(data, label,  
 test\_size=0.2,  
 shuffle=True,  
 stratify=label)  
  
  
model = RandomForestClassifier(  
 max\_depth=30,  
 max\_features='sqrt',  
 min\_samples\_leaf=2,  
 min\_samples\_split=5,  
 n\_estimators=300,  
 random\_state=42,  
 n\_jobs= -1  
)  
  
  
# Huấn luyện mô hình trên tập train.  
model.fit(x\_train, y\_train)  
  
y\_predict = model.predict(x\_test)  
  
# accuracy\_score(y\_predict, y\_test) → So sánh nhãn dự đoán với nhãn thực tế (y\_test).  
# score → Độ chính xác (tỉ lệ dự đoán đúng) của mô hình.  
score = accuracy\_score(y\_predict, y\_test)  
print('{}% of samples were classified correctly !'.format(score \* 100))  
  
# save model  
f = open('model27SignV1.p', 'wb')  
  
pickle.dump({'model': model}, f)

# **CHƯƠNG 5: XÂY DỰNG CHƯƠNG TRÌNH NHẬN DIỆN NGÔN NGỮ KÍ HIỆU THEO THỜI GIAN THỰC**

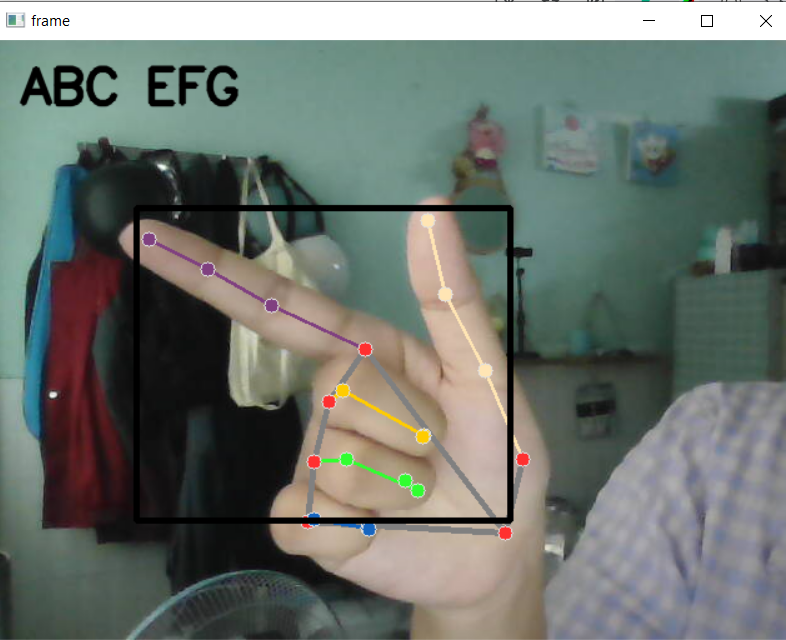
## **5.1 Các bước thực hiện**

* Mở file mô hình đã huấn luyện model27SignV1.p ở chế độ nhị phân và giải mã bằng pickle.load(), sau đó trích xuất đối tượng mô hình Random Forest từ khóa 'model'.
* Thiết lập webcam bằng cv2.VideoCapture(0) và khởi tạo các thành phần của MediaPipe Hands để phát hiện 21 điểm khớp trên bàn tay với ngưỡng tin cậy tối thiểu.
* Xác định một từ điển ánh xạ giá trị số trả về từ mô hình sang các ký tự A–Z và dấu cách, cũng như khởi tạo các biến để lưu trữ từ đang xây dựng, thời điểm nhận diện cuối cùng và cơ chế xác nhận ký tự.
* Trong vòng lặp chính, đọc từng khung hình từ webcam, chuyển đổi sang định dạng RGB rồi gọi hands.process() để phát hiện bàn tay và tọa độ landmark.
* Nếu phát hiện được bàn tay, chỉ giữ lại bộ landmark đầu tiên nếu có nhiều hơn một, rồi vẽ các kết nối và điểm khớp lên khung hình để hiển thị trực quan.
* Tách riêng tọa độ chuẩn hóa x và y của từng điểm, tính toạ độ pixel thực tế để vẽ khung chữ nhật bao quanh bàn tay, đồng thời ghép cặp x–y vào một mảng data\_aux làm đầu vào cho mô hình.
* Chuyển data\_aux thành mảng NumPy 2D và gọi model.predict() để thu về giá trị số tương ứng với ký tự dự đoán, rồi dùng từ điển nhãn để nhận dạng ký tự thực.
* So sánh ký tự hiện tại với ký tự trước đó:
  + Nếu trùng khớp, bắt đầu hoặc tiếp tục đếm thời gian; nếu khác, khởi động lại bộ đếm.
  + Khi tay giữ nguyên ký tự ít nhất 2 giây, đánh dấu ký tự đó là đã “xác nhận” và thêm vào chuỗi từ hiện tại (chỉ thêm một lần cho mỗi ký tự liên tiếp).
* Vẽ khung chữ nhật quanh bàn tay và hiển thị chuỗi từ hiện tại lên khung hình sử dụng cv2.putText().
* Theo dõi thời gian kể từ lần cuối phát hiện bàn tay; nếu vượt quá 2 giây không thấy tay, xóa nội dung từ đang xây dựng để bắt đầu lại.
* Hiển thị liên tục khung hình kết quả, cho phép nhấn phím q để thoát vòng lặp, sau đó giải phóng webcam và đóng tất cả cửa sổ OpenCV.

## **5.2 Code nhận diện ngôn ngữ kí hiệu theo thời gian thực**

import pickle  
import cv2  
import mediapipe as mp  
import numpy as np  
import time  
  
  
model\_path = r"D:\daihocnam4\sign\_language\project\model27SignV1.p"  
  
with open(model\_path, "rb") as f:   
 model\_dist = pickle.load(f)   
  
model = model\_dist['model']   
  
  
cap = cv2.VideoCapture(0)  
  
mp\_hands = mp.solutions.hands  
  
  
mp\_drawing = mp.solutions.drawing\_utils  
  
  
mp\_drawing\_styles = mp.solutions.drawing\_styles  
  
hands = mp\_hands.Hands(static\_image\_mode=True, min\_detection\_confidence=0.3)  
  
labels\_dict = {1: 'A', 2: 'B', 3: 'C', 4: 'D', 5: 'E', 6: 'F', 7: 'G', 8: 'H', 9: 'I',  
 10: 'J', 11: 'K', 12: 'L', 13: 'M', 14: 'N', 15: 'O', 16: 'P', 17: 'Q', 18: 'R', 19: 'S'  
 , 20:'T', 21:'U', 22:'V',23:'W', 24:'X',25:'Y',26:'Z',27:' '  
 }  
  
  
current\_word = ""   
last\_detected\_time = time.time()   
clear\_timeout = 2   
last\_character = None   
last\_time = time.time()   
  
confirm\_time = 2   
prediction\_start\_time = None  
confirmed\_character = None  
while True:  
 data\_aux = []  
 x\_ = []  
 y\_ = []  
 ret, frame = cap.read()  
  
 H, W, \_ = frame.shape  
  
 frame\_rgb = cv2.cvtColor(frame, cv2.COLOR\_BGR2RGB)  
  
 results = hands.process(frame\_rgb)  
  
 if results.multi\_hand\_landmarks:  
 last\_detected\_time = time.time()   
 if len(results.multi\_hand\_landmarks) > 1:  
 results.multi\_hand\_landmarks = [results.multi\_hand\_landmarks[0]]  
 # DRAW POINT  
 for hand\_landmarks in results.multi\_hand\_landmarks:  
 mp\_drawing.draw\_landmarks(  
 frame,  
  
 hand\_landmarks,  
  
 mp\_hands.HAND\_CONNECTIONS,  
  
  
 mp\_drawing\_styles.get\_default\_hand\_landmarks\_style(),  
  
 mp\_drawing\_styles.get\_default\_hand\_connections\_style()  
 )  
  
 for hand\_landmarks in results.multi\_hand\_landmarks:  
 for i in range(len(hand\_landmarks.landmark)):  
 x = hand\_landmarks.landmark[i].x  
 y = hand\_landmarks.landmark[i].y  
  
  
 x\_.append(x)  
 y\_.append(y)  
  
 for i in range(len(hand\_landmarks.landmark)):  
 x = hand\_landmarks.landmark[i].x  
 y = hand\_landmarks.landmark[i].y  
  
 data\_aux.append(x)  
 data\_aux.append(y)  
  
  
 x1 = int(min(x\_) \* W) - 10  
 y1 = int(min(y\_) \* H) - 10  
  
 x2 = int(max(x\_) \* W) - 10  
 y2 = int(max(y\_) \* H) - 10  
  
 prediction = model.predict([np.asarray(data\_aux)])  
 # print(prediction)  
 predicted\_character = labels\_dict[int(prediction[0])]  
  
 if predicted\_character == last\_character:  
 if prediction\_start\_time is None: prediction\_start\_time = time.time()  
 elif time.time() - prediction\_start\_time >= confirm\_time: confirmed\_character = predicted\_character prediction\_start\_time = None   
 else:  
 prediction\_start\_time = None   
  
 last\_character = predicted\_character  
  
 if confirmed\_character:  
 if len(current\_word) == 0 or confirmed\_character != current\_word[-1]:  
 current\_word += confirmed\_character  
 confirmed\_character = None   
 cv2.rectangle(frame, (x1, y1), (x2, y2), (0,0,0), 4)  
  
 cv2.putText(frame, current\_word, (20, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 1.3, (0, 0, 0), 3, cv2.LINE\_AA)  
  
  
 if time.time() - last\_detected\_time > clear\_timeout:  
 current\_word = ""  
  
 cv2.imshow('frame', frame)  
 if cv2.waitKey(1) == ord('q'):  
 break  
  
cap.release()  
cv2.destroyAllWindows()

## **5.3 Kết quả**



Hình 8 Kết quả của chương trình

# **CHƯƠNG 6: KẾT LUẬN**

Trong quá trình xây dựng hệ thống nhận diện ngôn ngữ ký hiệu sử dụng MediaPipe và mô hình học máy Random Forest, chương trình đã cho thấy hiệu quả khá tốt trong việc nhận diện các ký hiệu bàn tay theo thời gian thực. Việc kết hợp giữa khả năng trích xuất đặc trưng mạnh mẽ từ MediaPipe và mô hình Random Forest giúp hệ thống đạt độ chính xác cao, thời gian phản hồi nhanh và hoạt động ổn định trong điều kiện ánh sáng và môi trường bình thường.

## **6.1 Ưu và nhược điểm**

Ưu điểm:

* Giao diện trực quan, dễ sử dụng và không yêu cầu thiết bị phần cứng đặc biệt.
* Mô hình nhận diện có độ chính xác cao với tốc độ xử lý nhanh, hoạt động gần như thời gian thực.
* Sử dụng thư viện mã nguồn mở (OpenCV, MediaPipe, scikit-learn), tiết kiệm chi phí và dễ dàng triển khai.
* Hệ thống có khả năng học từ dữ liệu đầu vào thực tế và cải thiện hiệu suất thông qua huấn luyện lại mô hình.

Nhược điểm:

* Chưa hỗ trợ nhận diện nhiều bàn tay cùng lúc, gây hạn chế khi có nhiều người sử dụng đồng thời.
* Môi trường ánh sáng yếu hoặc phông nền phức tạp có thể ảnh hưởng đến độ chính xác.
* Chưa có khả năng nhận diện ký hiệu động (các từ hoặc câu ký hiệu yêu cầu chuyển động liên tục).
* Bộ dữ liệu sử dụng vẫn còn hạn chế về số lượng mẫu, chủng loại tay, màu da và độ tuổi người dùng.

## **6.2 Phương hướng phát triển**

* Mở rộng bộ dữ liệu huấn luyện, bổ sung thêm nhiều kiểu tay, góc nhìn, điều kiện ánh sáng và người dùng đa dạng để tăng khả năng khái quát của mô hình.
* Phát triển khả năng nhận diện ký hiệu động, tức là các chuỗi chuyển động tay để biểu diễn một từ hoặc câu hoàn chỉnh trong ngôn ngữ ký hiệu.
* Tích hợp công nghệ học sâu (Deep Learning) như CNN, RNN hoặc LSTM để cải thiện độ chính xác và khả năng học các đặc trưng phức tạp hơn.
* Tối ưu hóa giao diện người dùng, tích hợp hệ thống vào ứng dụng web hoặc di động nhằm tăng tính tiện lợi và khả năng tiếp cận.
* Hỗ trợ đa người dùng, nhận diện đồng thời nhiều bàn tay hoặc phân biệt từng người sử dụng khác nhau.
* Kết hợp với công nghệ Text-to-Speech để hỗ trợ chuyển đổi ngôn ngữ ký hiệu thành âm thanh phục vụ người nghe bình thường