

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**  
**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**  
**KHOA KHOA HỌC MÁY TÍNH**



**UIT**

**BÁO CÁO ĐỒ ÁN**

**MÔN HỌC: MÁY HỌC**

**ĐỀ TÀI: NHẬN DIỆN NGÔN NGỮ KÍ HIỆU CẦN THIẾT  
ĐỐI VỚI NGƯỜI KHIẾM THÍNH CẦN GIÚP ĐỠ TRÊN  
ĐƯỜNG**

**Lớp:** CS114.O11.KHCL

**Giảng viên hướng dẫn:**

1. Lê Đình Duy
2. Phạm Nguyễn Trường An

**Sinh viên thực hiện:**

- |                    |          |
|--------------------|----------|
| 1. Phan Trọng Nhân | 21522408 |
| 2. Lê Văn Hiến     | 21522060 |

Thành phố Hồ Chí Minh, ngày 15 tháng 01 năm 2024

## Tóm tắt đề án

- Đây là đề án xây dựng mô hình nhận diện cử chỉ bàn tay bằng ngôn ngữ ký hiệu (Đầu vào của bài toán là những hình ảnh được chụp liên tục từ camera, Đầu ra là kết quả dự đoán từ tương ứng với kết quả đó ). Mô hình sử dụng thuật toán Random Forest Classifier và dùng MediaPipe để trích xuất 21 landmark từ hình ảnh RGB của bàn tay. Có thể dự đoán được 20 từ ngôn ngữ ký hiệu (được liệt kê bên dưới).
- Link github (bao gồm đầy đủ code giải thích từng đoạn code và dữ liệu): <https://github.com/LeHienNSB?tab=repositories>

# MỤC LỤC

0. Update sau khi vấn đáp.....	0
I. Giới thiệu đề tài.....	1
1. Tổng quan về đề tài .....	1
2. Lý do chọn đề tài/đối tượng sử dụng .....	1
3. Mô tả bài toán .....	3
II. Mô tả bộ dữ liệu .....	4
1. Cách thu thập dữ liệu .....	4
2. Cách test và sà n lọc dữ liệu.....	5
3. Phân chia dữ liệu .....	6
III. Xử lí dữ liệu .....	7
1. Sơ đồ mô tả.....	7
2. Xử lý giá trị đầu vào cho modle .....	8
3. Normalization / dimensionally reduction .....	9
IV. Mô hình.....	10
V. Demo chương trình.....	11
VI. Đánh giá kết quả .....	11
1. Tỷ lệ chính xác theo accuracy_score .....	11
2. Tỷ lệ chính xác theo confusion_matrix .....	12
3. Độ chính xác thực tế .....	13
VII. Kết Luận .....	14
1. Kết luận khái quát .....	14
2. Hướng cải thiện và phát triển.....	14
a. Cải thiện .....	14
b. Phát triển.....	15
VIII. Tài liệu tham khảo.....	16

## 0. Update sau khi vấn đáp

- Chỉnh sửa file báo cáo thêm hoàn thiện và đầy đủ (Bổ sung những lời giải thích, hoàn thiện những lời giải thích trong file code ở github,..).
- Train lại những từ có xác suất dự đoán đúng thấp như: Medicine, Help, Police, Lost,...

# I. Giới thiệu đề tài

## 1. Tổng quan về đề tài

- Mục Tiêu Chính: Phát triển hệ thống máy học nhận diện ngôn ngữ kí hiệu để hỗ trợ giao tiếp cho người khiếm thính và cải thiện tương tác giữa người và máy tính.
- Phương Pháp Nhận Diện: Sử dụng máy học (Machine Learning) và trí tuệ nhân tạo (AI) để xây dựng mô hình nhận diện và hiểu ngôn ngữ kí hiệu.
- Dữ Liệu: Xây dựng tập dữ liệu đa dạng với nhiều biểu hiện ngôn ngữ kí hiệu khác nhau, và tiền xử lý dữ liệu để tăng độ chính xác của mô hình.
- Ứng Dụng Thực Tế: Tích hợp vào các thiết bị di động và máy tính để tạo ra các ứng dụng hỗ trợ giao tiếp cho người khiếm thính, và sử dụng trong giáo dục để hỗ trợ học sinh khiếm thính.

## 2. Lý do chọn đề tài/đối tượng sử dụng

- Có hai lý do chính nhóm chọn đề tài nhận diện cử chỉ bản tay giành cho người khiếm thính cần giúp đỡ khi bắt gặp trên đường:
  - Thứ nhất, đề tài có tính ứng dụng trong đời sống . Do trên thế giới có tới 70 triệu người là người khiếm thính và chiếm 7.3% số người bị khuyết tật(nhiều nhất trong tất cả các loại), dẫu vậy số người không bị khiếm thính mà biết được ngôn ngữ ký hiệu là rất ít, hầu như không bắt gặp được. Thế nên, những công cụ hỗ trợ người khiếm thính là cần thiết. Đề tài này cũng chính là một công cụ để hỗ trợ họ và nếu có thể cải thiện thành nhận diện ngôn ngữ ký hiệu nói chung thì sẽ rất hữu ích để giúp người bình thường có thể hiểu được người khiếm thính muốn nói gì, có thể giúp người khiếm thính tới gần hơn với cuộc sống như một người khỏe mạnh bình thường.

- Thứ hai, bộ dữ liệu của đề tài có thể thu thập nhiều và nhanh do thuật toán là bức ảnh của bàn tay. Nên có thể thu thập dữ liệu nhanh chóng và dễ dàng chỉnh sửa vì chỉ cần chụp ảnh của bàn tay rồi cho vào bộ dữ liệu.
- Đối tượng sử dụng :
- Do số lượng dữ liệu có thể thu thập được đối với máy laptop của nhóm giới hạn và model chưa đủ để có thể thu thập dữ liệu khuôn mặt (cho các từ liên quan tới biểu cảm), chuyển động quỹ đạo/ dự đoán chuyển động (dung cho các từ có nhiều động tác) và các thủ thuật như xử lý trật tự ngữ pháp, phát biểu những ngôn ngữ ký hiệu bị bỏ sót ,xử lý phân biệt được các ký hiệu giống nhau. Vì thế nhóm đã thu hẹp đề tài lại và thu thập 20 từ cơ bản cần thiết mà có thể một phần nào đó giúp đỡ được người qua đường bằng cách ghép từng từ lại thành 1 câu.
  - Với đề tài đó, đối tượng sử dụng của nhóm là những người đi đường bình thường đi đường không biết ngôn ngữ ký hiệu mà khi họ gặp phải người khiếm thính đang cần giúp đỡ thì họ có thể hiểu được người ta cần giúp gì và có thể giúp đỡ người ta(có thể trong cả trường hợp khẩn cấp). Tuy rằng có những cách khác để hiểu được người khiếm thính như là :viết ra giấy hay viết ra điện thoại (đặt trong trường hợp điện thoại họ hết pin hoặc mất), thì thực tế là có thể có những người khiếm thính không biết ngôn ngữ bình thường vì ngữ pháp và từ hoàn toàn khác nhau, mà nếu có biết thì việc một người đi đường bình thường mang theo bút giấy là khá ít và cho người lạ cầm điện thoại để viết ra có thể là không đảm bảo rằng họ sẽ không làm việc gì xấu. Do đó, khi đặt trong trường hợp cụ thể thì model có thể phát huy tác dụng, và nếu có thể cải tiến đa dạng từ ngữ hơn đề tài này sẽ giúp được nhiều người khiếm thính.

### 3. Mô tả bài toán

- Bài toán thuộc lớp bài toán phân loại, có 20 lớp đại diện cho 20 từ cơ bản cần thiết trong trường hợp muốn giúp đỡ người khiếm thính khi gặp trên đường (You, I/My, Where, Store/Station, Help, Eat, Drink, Home, Gas, Need, Medicine, Police, Direction, Transportation, Restroom, Call/Phone, Lost, Keys, Right now, How).

Các ví dụ có thể ghép được từ 20 từ:

- I need help/ eat/ drink/call/police/ medicine (Right now)
  - Where is direction to resroom/ my house/ police/transportation.
  - I lost my key/phone
  - Where is the gas station/ medicine store/ police station.
  - Where direction to eat/ drink? How ?
- Đầu vào của bài toán là những hình ảnh được chụp liên tục từ camera.
  - Đầu ra là kết quả dự đoán từ tương ứng với kết quả đó.

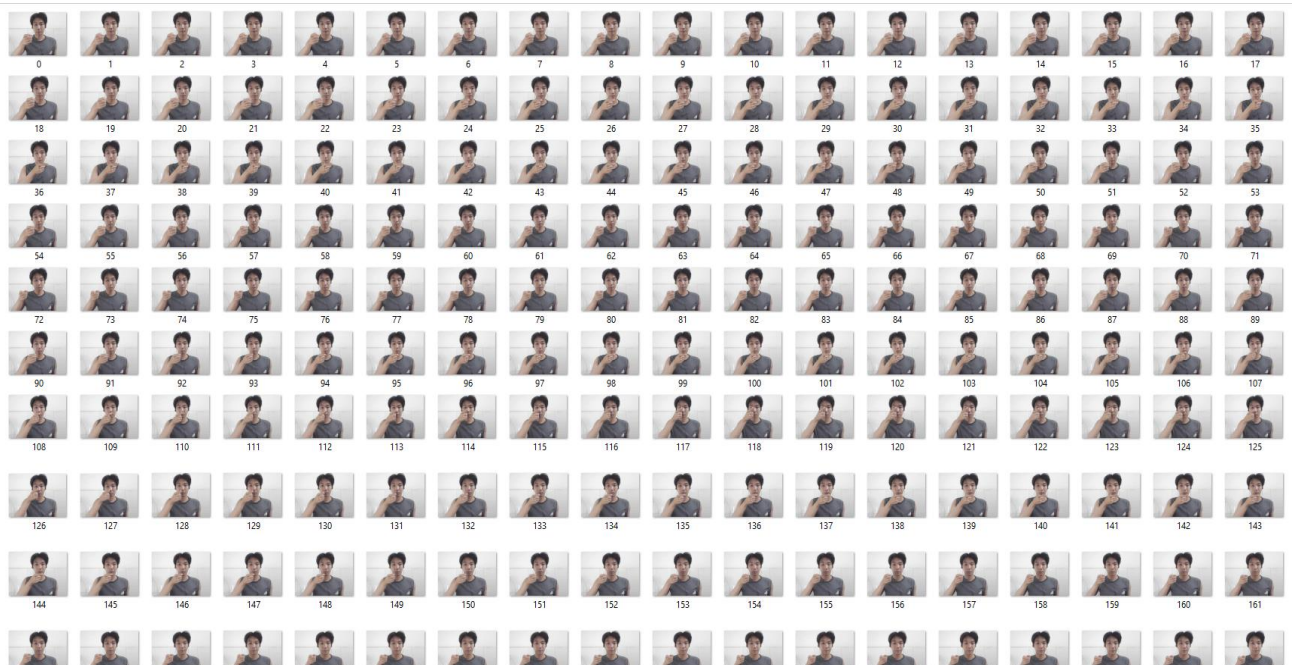


Hình 1: Ví dụ về bài toán

## II. Mô tả bộ dữ liệu

### 1. Cách thu thập dữ liệu

- Nhóm thu thập dữ liệu bằng cách tạo ngôn ngữ kí hiệu bằng tay và dùng camera máy tính để chụp lại 200 ảnh với mỗi từ ngôn ngữ kí hiệu bằng tay đó. Và 200 tấm ảnh ấy ta di chuyển tay xung quanh khung hình để camera có thể nhận được tay với nhiều góc độ, từ đó tạo nên độ đa dạng cho bộ data.



*Hình 2: 200 tấm ảnh với từ “You”*



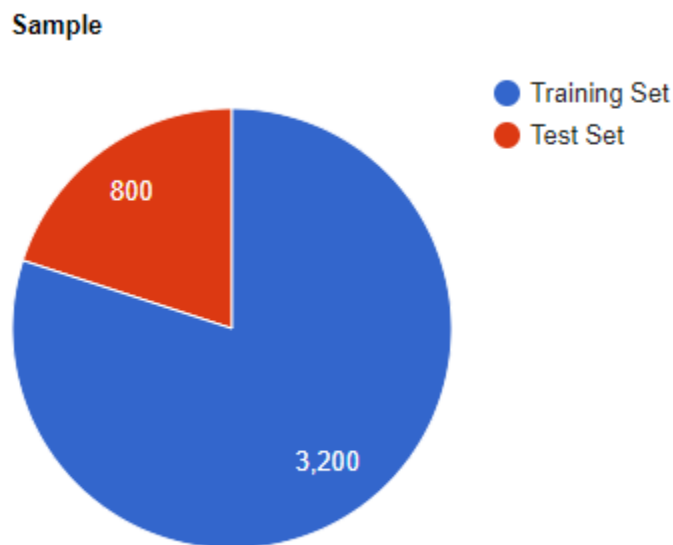
- Để lấy được những tấm ảnh một cách dễ dàng nhất thì nhóm đã code ra một file python để có thể thu thập 200 tấm ảnh trên: (hãy quan sát code trong file ***collect\_img.py*** trong link github bên dưới). Với code đó ta sử dụng thư viện OpenCV sử dụng câu lệnh `cv2.VideoCapture(0)` để thu thập dữ liệu video từ Camera của máy tính. Sau khi người dùng bấm 'Q' để bắt đầu, nó tạo thư mục cho lớp hiện tại (lớp muốn thu thập dữ liệu) và lưu các frame video dưới dạng hình ảnh (.jpg) vào thư mục đó. Số lượng frame được giới hạn bởi `dataset_size` (200). Có thêm câu lệnh `waitkey(3000)` để delay khi thu thập dữ liệu, nhóm có thể có thời gian để đặt tay vào vị trí chuẩn bị trước khi câu lệnh bắt đầu.

## 2. Cách test và sànlọc dữ liệu

- Nhóm code thêm file python với mục đích test sànlọc dữ liệu. Có yếu tố cần sànlọc đó chính là feature của data. Vì nhóm có sử dụng một thuật toán là giảm chiều của data để có thể lưu thành 1 list để dễ tương tác. Nhưng sau khi thao tác xong phải trả về dạng mảng khi cho vào train model. Bước này sẽ xảy ra sự cố nếu số lượng feature của các ảnh không bằng nhau (do có những tấm bị lỗi không nhận được đủ tất cả các điểm trên bàn tay). Và để có thể quản lý được dữ liệu đầu vào, lỗi ở tập dữ liệu nào nhóm code ra một file giành riêng cho việc trả về số lượng feature của mỗi tệp (hãy quan sát code trong file ***test\_Shape.py*** trong link github bên dưới)
- Sau khi đã biết được lỗi feature ở tệp nào nhóm code thêm một file có thể thay thế thu thập lại bộ dữ liệu ở tệp lỗi, giúp công việc thu thập dữ liệu dễ dàng và dễ quản lý hơn. Ở hình ví dụ nhóm thu thập lại dữ liệu cho tệp số một (hãy quan sát code trong file ***collect1.py*** trong link github bên dưới)
- Cách làm tương tự như thu thập dữ liệu cho 1 tệp có 20 class chỉ bỏ bớt một vòng lặp và địa chỉ dẫn sâu hơn vào bên trong tệp.

### 3. Phân chia dữ liệu

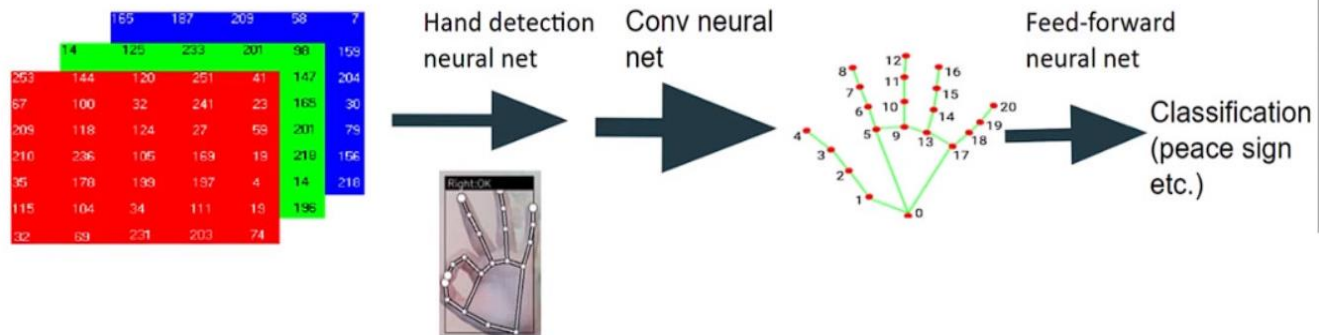
- Sau khi phân loại và gán nhãn cho dữ liệu nhóm thống kê được tổng cộng 4000 mẫu với 20 lớp, với mỗi lớp có 200 tấm ảnh.
- Khi chia dữ liệu nhóm đã chia thành 2 tập training set và test set với số lượng như sau:
  - Training set với 3200 mẫu (tức 80% của tổng cộng 4000 mẫu) .
  - Test set với 800 mẫu (tức 20% của tổng cộng 4000 mẫu) random.



*Hình 3: Sự phân bố của các tập dữ liệu*

### III. Xử lý dữ liệu

#### 1. Sơ đồ mô tả



Hình 4: Mô tả sơ đồ hoạt động

- Đầu vào dữ liệu bằng mảng hình ảnh , sau đó hình ảnh sẽ được chuyển từ BGR sang RGB . Hình ảnh sau khi chuyển màu được đưa vào model nhận diện bàn tay mediapipe của google để cắt ra hình ảnh bàn tay trong ảnh và chuyển về thành một mạng lưới neural các điểm trên một bàn tay(bao gồm 21 điểm). Các điểm trên bàn tay sau đó sẽ được đưa vào thuật toán phù hợp từ đó đưa ra model có thể nhận diện được cử chỉ bàn tay.
  - Khởi tạo mediapipe hand detection: (hãy quan sát code trong file ***data\_set.py*** (hàng 10, 11, 12) trong link github bên dưới )
  - Kết quả là landmark được đọc ra trong ảnh: (hãy quan sát code trong file ***data\_set.py*** (hàng 54) trong link github bên dưới)

## 2. Xử lý giá trị đầu vào cho modle

Hãy quan sát code trong file ***data\_set.py*** (hàng 55 đến hàng 76) trong link github bên dưới.

- Lấy tọa độ x(chiều rộng) và y(chiều dài) trên từng điểm của bàn tay trên khung hình. Nếu chỉ có một tay thì trả về 21 điểm trên bàn tay, mỗi điểm có tọa độ x và y => 42 giá trị trên một bàn tay. Nếu hai bàn tay trả về 42 điểm và có 84 tọa độ. Để hai tay và một tay có cùng kích thước mảng khi train thì cho 42 điểm mà hình ảnh chỉ có một tay thiếu, gán giá trị còn thiếu = 0. Gán từng giá trị tương ứng với lần lượt các index.

# vì sao sử dụng cách lấy tọa độ thay vì sử dụng xử lý ảnh?

Nguyên nhân: vì tay của mỗi người là khác nhau, những khác biệt như là(màu da, độ dài ngón tay, kích thước bàn tay, hình dáng bàn tay), ngoài ra các sự khác biệt như là hậu cảnh ,ánh sáng. Các nguyên nhân đó dẫn tới việc thu thập ảnh cho dữ liệu phải rất khắt khe và đặc biệt phải nhiều vô số để có thể tạo ra được model có thể đúng. Việc sử dụng hình ảnh để train gần như là không thể đối với bài toán này.

Cách giải quyết: đó chính là sử dụng landmark của bàn tay, vì mọi người hầu hết đều có cấu tạo bàn tay giống nhau nên việc chỉ sử dụng dữ liệu này cải thiện đáng kể độ chính xác và giảm đáng kể dữ liệu cần phải thu thập, khiến cho bài toán mới thực sự khả thi.

Hãy quan sát code trong file ***data\_set.py*** (hàng 25 đến hàng 31) trong link github bên dưới.

- Chọn điểm số 0(index) làm mốc gán giá trị tọa độ x=0, y=0; các điểm còn lại là khoảng cách từ điểm đó tới điểm mốc => như vậy người dùng có thể di chuyển tay trên khắp màn hình mà không cần phải thu thập thêm quá nhiều data.

- Việc xử lý tọa độ chuyển thành khoảng cách là một cách nữa khiến lượng data cần cho việc train có thể giảm thiểu hơn nữa đặc biệt là khi tài nguyên của nhóm bị giới hạn .  
\*\* Giải thích: vì tọa độ khi tay di chuyển khắp màn hình là khác nhau dẫn đến việc bộ dữ liệu cần phải thu thập nhiều lên. Nên khi chuyển về khoảng cách của các điểm tới điểm mốc thì sẽ giải quyết được vấn đề này vì dù ở đâu trên màn hình thì khoảng cách từ các điểm tới điểm mốc là gần như không đổi.

### 3. Normalization / dimensionally reduction

Hãy quan sát code trong file *data\_set.py* (hàng 37 đến hàng 44) trong link github bên dưới.

- Normalize các giá trị trong list bằng cách chia cho phần tử lớn nhất => giá trị sẽ chỉ giao động từ -1 đến 1 => dễ dàng hơn cho model train, cải thiện được độ chính xác , chạy nhanh hơn và hiệu quả hơn với các giá trị có tính nhất quán

Hãy quan sát code trong file *data\_set.py* (hàng 42) trong link github bên dưới.

- Chuyển mảng nhiều phần tử thành mảng một chiều ( cách này gọi là dimensionally reduction) => dữ liệu dễ dàng tương tác và chỉnh sửa.

## IV. Mô hình

- Với bài toán nhận diện chữ chỉ bàn tay ta dùng thuật toán Random forest classifier. (Hãy quan sát code trong file *train\_clasifier.py* (hàng 13 đến hàng 20) trong link github bên dưới)
- Hàm `train_test_split`: hàm được import từ thư viện `sklearn` dùng chia dữ liệu ra thành 2 phần `train` và `test`

Các thông số :

- `Data`, `labels` là đầu vào của dữ liệu
- `Test_size= 0,2` lấy 20% dữ liệu để test và 80% dữ liệu còn lại là để `train`
- `Shuffle = True` lấy dữ liệu bất kì trong tập
- `Stratify= labels` đảm bảo việc phân phối nhãn lớp trong tập dữ liệu gốc được giữ nguyên
- Khái niệm cơ bản về Random Forest classifier: gồm nhiều decision tree và chọn ra kết quả có nhiều vote nhất , nhiều decision tree luôn cho ra kết quả đúng hơn một decision tree.
  - Các bước để tạo nên random forest classifier:

Bước 1: Boosttrapped dataset(bagging) :tạo nhiều mẫu data ngẫu nhiên từ một data lớn.

Bước 2: Feature Randomness : huấn luyện decsion tree cho nhưng bộ data một cách độc lập (Các feature cũng được chọn một cách ngẫu nhiên, không chọn tất cả).

Bước 3: Xây dựng tree.

Bước 4: Cho dữ liệu vào từng cây và ghi chú kết quả mà tree dự đoán.

Bước 5: Aggregation: chọn kết quả mà được vote nhiều nhất.

## V. Demo chương trình

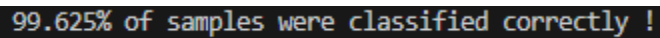
- Link test module:

[https://youtube.com/shorts/Qygo7XkFkOY?si=cFUOwQ5B\\_Lo3ycXY](https://youtube.com/shorts/Qygo7XkFkOY?si=cFUOwQ5B_Lo3ycXY)

## VI. Đánh giá kết quả

- Ở đây ta dùng `accuracy_score` và `confusion_matrix` từ thư viện `sklearn` để đánh giá mô hình, `accuracy` càng lớn và ma trận càng sát với mẫu test thì mô hình càng tốt.
- Sau quá trình training thì nhận kết quả predict trên tập train và tập test như sau:

### 1. Tỉ lệ chính xác theo `accuracy_score`



```
99.625% of samples were classified correctly !
```

*Hình 5: Độ chính xác của mô hình theo `accuracy_score`*

⇒ Do bộ data khá ít nên kết quả chưa thực sự ý nghĩa

## 2. Tỷ lệ chính xác theo confusion\_matrix

[40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0]
[ 0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0]
[ 0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0]
[ 0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0]
[ 1	0	0	0	39	0	0	0	0	0	0	0	0	0	0	0	0	0	0]
[ 0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	0]
[ 0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0]
[ 0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0]
[ 0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0]
[ 0	0	0	0	2	0	0	0	0	38	0	0	0	0	0	0	0	0	0]
[ 0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0	0]
[ 0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0	0]
[ 0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0	0]
[ 0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0	0]
[ 0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0	0]
[ 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0	0]
[ 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0	0]
[ 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40	0]
[ 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	40]

Hình 6: Độ chính xác của mô hình theo confusion\_matrix

- Mô hình có độ chính xác 99.625% và ma trận gần sát với mẫu test , mức chính xác này rất cao tuy nhiên không chắc chắn rằng mô hình sẽ đúng tất cả vì nhiều lí do khác nhau.

Phân tích:

- Ở dòng thứ nhất 40 tấm có label là class 0 và model đã predict được 40/40 tấm là ở class 0 => hoàn toàn chính xác
- tuy nhiên ở dòng thứ 5 có 39 tấm là đoán đúng class 4 và một tấm bị đoán nhầm sang class 0 , dòng thứ 10 có 28 tấm là predict đúng ở class 9 và 2 tấm nhầm sang label class 4.

⇒ Từ đó suy luận được kết quả chỉ đúng 99.625% là do sai 3 tấm



### **3. Độ chính xác thực tế**

Sau nhiều lần thử nghiệm model thực tế thì nhóm rút ra kết luận: model chạy chưa ổn định như test trên giấy. Nguyên nhân chính là do dữ liệu bị giới hạn muốn đúng hơn cần thu thập thêm nhiều dữ liệu. Trên thực tế, thời gian thực chuyển động nhanh và model media pipe một phần là không thể bắt được đúng cử chỉ tay. Hơn nữa đầu vào ảnh mỗi giây là rất nhiều nên dù test trên giấy sai là không lớn nhưng so với đầu vào nhiều tấm như vậy thì độ sai có thể trở nên lớn. Tuy vậy, model vẫn predict được đúng nhiều trường hợp và một số từ nhận diện được đúng hơn so với những từ khác.

# VII. Kết Luận

## 1. Kết luận khái quát

- Mô hình có sau khi huấn luyện có độ chính xác rất cao , điểm accuracy cao nhất là 99%.
- Tuy nhiên một số từ kí hiệu bằng 2 tay có độ chính xác rất thấp (ví dụ: Medicine, Help,..) những từ ấy hay dự đoán sang các từ khác như: “Help” thành “Medicine” và ngược lại.
- Nguyên nhân chủ yếu khiến mô hình chưa tốt:
  - Dữ liệu khá ít, mỗi lớp chỉ có 200 ảnh cho máy học.
  - Chưa có kinh nghiệm trong khâu thu thập và gán dữ liệu nên dẫn đến dữ liệu nhiều khá nhiều.
  - Ngôn ngữ kí hiệu quá đa dạng nên một số từ có thể khá giống nhau dẫn đến mô hình dự đoán sai lệch.
  - Mô hình code chưa tối ưu khiến cho máy tương tác với ngôn ngữ kí hiệu giảm đi sự hiệu quả.
  - Chưa hiểu sâu về ngôn ngữ kí hiệu khiến cho một số từ có thể kí hiệu sai tay dẫn đến sự sai lệch.

## 2. Hướng cải thiện và phát triển

### a. Cải thiện

- Thu thập và Mở rộng Dữ liệu: Thu thập dữ liệu đa dạng về ngôn ngữ kí hiệu từ nhiều nguồn và cộng đồng khác nhau. Tăng cường dữ liệu bằng cách tạo thêm biến thể, đa dạng hóa ngữ cảnh, và đảm bảo rằng dữ liệu phản ánh đầy đủ sự đa dạng của ngôn ngữ kí hiệu.

- Cải thiện xử lý dữ liệu: Áp dụng các kỹ thuật tiền xử lý để giảm nhiễu và làm sạch dữ liệu. Chuẩn hóa dữ liệu để đảm bảo đồng đều giữa các đặc trưng.
- Sử dụng Kiến trúc Mô hình Nâng cao: Sử dụng các kiến trúc mô hình mạnh mẽ và phức tạp như các biến thể của mạng nơ-ron sâu (deep neural networks) để nắm bắt được các mô hình phức tạp của ngôn ngữ kí hiệu.
- Thêm nhiều từ để model đa dạng hơn và sử dụng được nhiều mục đích hơn.

## **b. Phát triển**

- Phát triển Ứng dụng Thực tế: Hướng phát triển mô hình để tích hợp vào các ứng dụng thực tế như hệ thống giao tiếp hỗ trợ cho người khiếm thính hoặc giáo dục ngôn ngữ kí hiệu.
- Áp dụng thêm nhiều model như face detectin, movement detection,... để có thể tăng độ chính xác khi nhận diện ngôn ngữ ký hiệu. Và có thể detect được nhiều ký hiệu hơn.

## VIII. Tài liệu tham khảo

1. <https://developers.google.com/mediapipe>
2. [https://youtu.be/a99p\\_fAr6e4?si=1lGJROWtlBDoGPdd](https://youtu.be/a99p_fAr6e4?si=1lGJROWtlBDoGPdd)
3. <https://www.youtube.com/watch?v=G6hVRVG74lc>
4. <https://www.youtube.com/watch?v=0FcwzMq4iWg&t=304s>
5. <https://www.youtube.com/watch?v=uVJEbDQ5a7M>
6. <https://www.youtube.com/watch?v=MJCSjXepaAM&t=1112s>
7. <https://techvidvan.com/tutorials/hand-gesture-recognition-tensorflow-opencv/>
8. [https://www.researchgate.net/publication/345142103\\_Hand\\_gesture\\_recognition\\_using\\_machine\\_learning\\_algorithms](https://www.researchgate.net/publication/345142103_Hand_gesture_recognition_using_machine_learning_algorithms)
9. <https://github.com/topics/hand-gesture-recognition>
10. <https://www.javatpoint.com/image-processing-using-machine-learning>



