ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

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. Phạn Nguyễn Trường An

Sinh viên thực hiện:

1. Phan Trọng Nhân 21522408
2. Lê Văn Hiển 21522060

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

MỤC LỤC

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. Mô tả bài toán

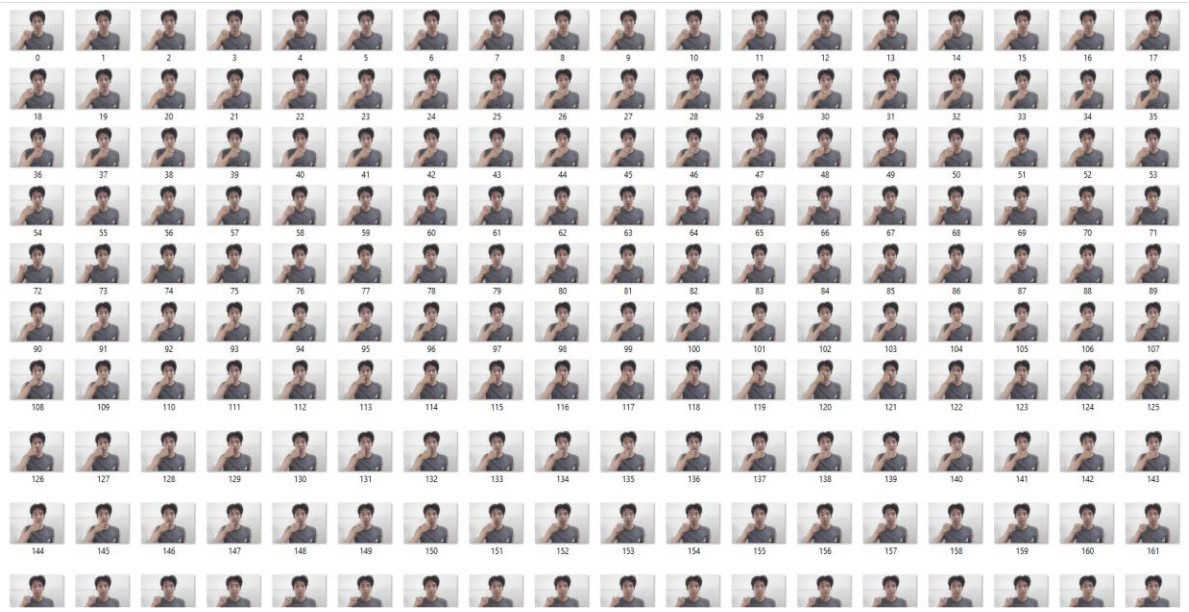
- Bài toán thuộc lớp bài toán phân loại, có 20 lớp đại diện cho 20 từ cơ bản cần thiết trong trường hợp muốn giúp đỡ người khiếm thị khi gặp trên đường (You, I/My, Where, Store/Station, Help, Eat, Drink, Home, Gas, Need, Medicine, Police, Direction, Transportation, Restroom, Call/Phone, Lost, Keys, Right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



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:

```python
import os
import cv2


DATA_DIR = './data'

number_of_classes = 0
dataset_size = 200

cap = cv2.VideoCapture(0)
name = 0


if not os.path.exists(os.path.join(DATA_DIR, str(name))):
    os.makedirs(os.path.join(DATA_DIR, str(name)))

print('Collecting data for class {}'.format(name))

done = False
while True:
    ret, frame = cap.read()
    cv2.putText(frame, 'Ready? Press "Q" ! :)', (100, 50), cv2.FONT_HERSHEY_SIMPLEX, 1.3, (0, 255, 0), 3,cv2.LINE_AA)
    cv2.imshow('frame', frame)
    if cv2.waitKey(25) == ord('q'):
        cv2.waitKey(3000)
        break

counter = 0
while counter < dataset_size:
    ret, frame = cap.read()
    cv2.imshow('frame', frame)
    cv2.waitKey(50)
    cv2.imwrite(os.path.join(DATA_DIR, str(name), '{}.jpg'.format(counter)), frame)

    counter += 1

cap.release()
cv2.destroyAllWindows()
```

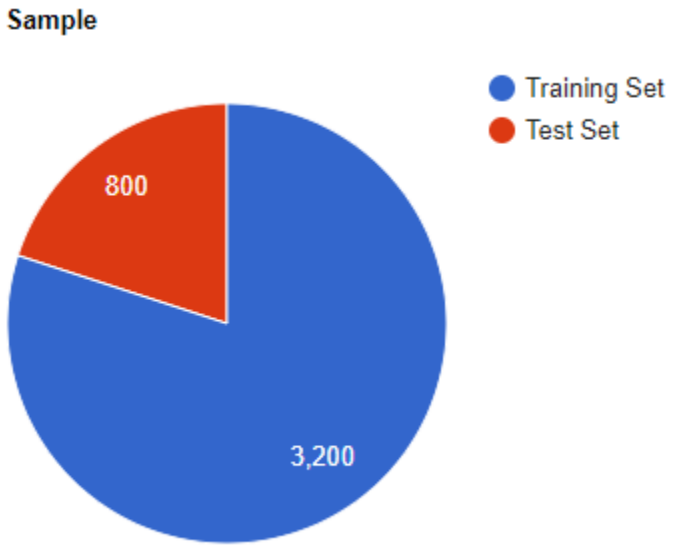
Hình 3: Code thu thập dữ liệu

- Mã trên sử dụng thư viện OpenCV để thu thập dữ liệu video từ webcam. 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).

2. 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:

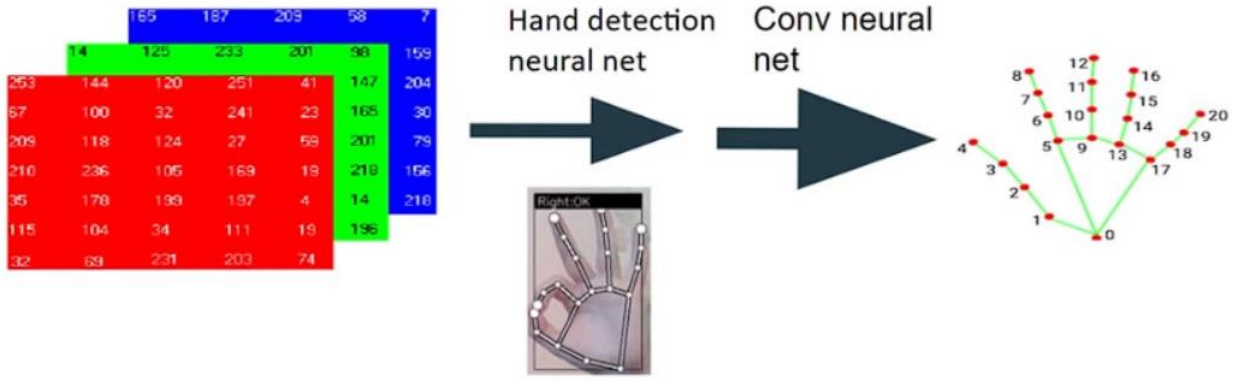
- Training set với 3200 mẫu (tức 80% của tổng cộng 4000 mẩu) random.
- Test set với 800 mẫu (tức 80% của tổng cộng 4000 mẩu) còn lại.



Hình 4: Sự phân bố của các tập dữ liệu

III. Xử lí dữ liệu

1. Nhận điện bàn tay



Hình 5: Mô tả cách nhận diện bàn tay

- 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 và chuyển về thành một mạng lưới neural các điểm trên bàn tay(bao gồm 21 điểm).
- Khởi tạo mediapipe hand detection:

```
mp_hands = mp.solutions.hands #khởi tạo lớp hand
mp_drawing = mp.solutions.drawing_utils  # thiết
mp_drawing_styles = mp.solutions.drawing_styles
```

Hình 6: Code khởi tạo mediapipe hand detection

- Kết quả là landmark được đọc ra trong ảnh:

```
results = hands.process(img_rgb)
```

Hình 7: Code kết quả là landmark được đọc ra trong ảnh

2. Xử lý giá trị đầu vào cho modle

```
results = hands.process(img_rgb) #kết quả là landmark được đọc ra trong ảnh
if len(results.multi_hand_landmarks) == 2 and results.multi_hand_landmarks: #check xem kết quả có được phát hiện hay không
    for hand_landmarks in results.multi_hand_landmarks: #vòng lặp số bàn tay
        for _, landmark in enumerate(hand_landmarks.landmark): #vòng lặp từng điểm trên bàn tay
            landmark_x = min(int(landmark.x * image_width), image_width - 1) #đề phòng nếu có ảnh mà tay nằm ngoài ảnh về chiều rộng
            landmark_y = min(int(landmark.y * image_height), image_height - 1) #đề phòng nếu có ảnh mà tay nằm ngoài ảnh về chiều dà
            data_aux = np.array((landmark_x, landmark_y)) #chuyển x và y thành dạng mảng
            aux.append(data_aux) #gộp các tọa độ lại theo nhóm mốc
        temp_landmark_list = pre_process_landmark(aux) #list sau khi được chuyển giá trị và normalize
    data.append(temp_landmark_list)
    labels.append(dir_)
elif len(results.multi_hand_landmarks) == 1 and results.multi_hand_landmarks: #check xem kết quả có được phát hiện hay không
    for hand_landmarks in results.multi_hand_landmarks: #vòng lặp số bàn tay
        for _, landmark in enumerate(hand_landmarks.landmark):#vòng lặp từng điểm trên bàn tay
            landmark_x = min(int(landmark.x * image_width), image_width - 1) #đề phòng nếu có ảnh mà tay nằm ngoài ảnh về chiều rộng
            landmark_y = min(int(landmark.y * image_height), image_height - 1) #đề phòng nếu có ảnh mà tay nằm ngoài ảnh về chiều dà
            data_aux = np.array((landmark_x, landmark_y))#chuyển x và y thành dạng mảng
            aux.append(data_aux)
        temp_landmark_list = pre_process_landmark(aux)
        for i in range (len(temp_landmark_list),84): #nếu bộ dữ liệu chưa đủ 84 phần tử thì thêm các phần tử còn thiếu là 0
            temp_landmark_list.append(0.0)
    data.append(temp_landmark_list)
    labels.append(dir_)
```

Hình 8: Code lấy tọa độ các điểm

- Lấy tọa độ x(chiều rộng) và y(chiều dài) trên từng điểm trên bàn tay của khung hình. Nếu chỉ có 1 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 2 bàn tay trả về 42 điểm và có 84 tọa độ. Để 2 tay và một tay có cùng kích thước mảng khi train thì cho 42 điểm mà hình ảnh chỉ có 1 tay thiếu , gán giá trị còn thiếu = 0. gán từng giá trị tương ứng với lần lượt các index.

```
def pre_process_landmark(landmark_list):
    temp_landmark_list = copy.deepcopy(landmark_list) #tạo một list giống list cũ nhưng ở địa chỉ khác hoàn toàn

    # chuyển các tọa độ của từng mốc thành khoảng cách các mốc so với điểm gốc
    base_x, base_y = 0, 0
    for index, landmark_point in enumerate(temp_landmark_list):
        if index == 0: #nếu là điểm gốc
            base_x, base_y = landmark_point[0], landmark_point[1]

        temp_landmark_list[index][0] = temp_landmark_list[index][0] - base_x
        temp_landmark_list[index][1] = temp_landmark_list[index][1] - base_y
```

Hình 9: Code xử lí tọa độ

- Chọn điểm số 0 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 dung có thể di chuyển tay trên khắp màn hình mà không cần phải thu thập thêm quá nhiều data.

3. Normalization / dimensionally reduction

```python
# Normalization để các phần tử giá trị chỉ từ -1 đến 1
max_value = max(list(map(abs, temp_landmark_list)))
def normalize_(n):
    if max_value !=0:
        return n / max_value

temp_landmark_list = list(map(normalize_, temp_landmark_list))

return temp_landmark_list
```

Hình 10: Code hàm Normalize

- 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.

```python
# Convert to a one-dimensional list
temp_landmark_list = list(itertools.chain.from_iterable(temp_landmark_list))
```

Hình 11: Code thực hiện dimensionally reduction

- Chuyển mảng nhiều phần tử thành mảng một chiều => cách này gọi là dimensionally reduction.

IV. Mô hình

- Với bài toán nhận diện cử chỉ bàn tay ta dùng thuật toán Random forest classifier.

```
data = np.asarray(data_dict['data']) #chuyển list thành 1 array (mảng)
labels = np.asarray(data_dict['labels'])

x_train, x_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, shuffle=True, stratify=labels) #ph

model = RandomForestClassifier() #sử dụng thuật toán randomforestclassifier làm model

model.fit(x_train, y_train) #fit giá trị
```

Hình 12: Code xử lí dữ liệu trong mô hình

- 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 tets 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 decission 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: https://youtube.com/shorts/Qygo7XkFkOY?si=cFUOwQ5B_Lo3ycXY

VI. Đánh giá kết quả

- Ở đây ta dung 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 lệnh chính xác theo accuracy_score

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

Hình 13: Độ chính xác của mô hình theo accuracy_score

2. Tỉ lệnh 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]
 [ 0 40  0  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  0 40  0  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  0 40  0  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  0 40  0  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  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  0 40  0  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  0 40  0  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  0 40  0  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  0 40  0  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  0 40  0]
 [ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 40]]
```

Hình 14: 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.

VII. Kết Luận

1. Kết luận khái quát

- Mô hình có sau khi huấn luyện có đô 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 nhiễu 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 lí 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.

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.

VIII. Tài liệu tham khảo

1. https://developers.google.com/mediapipe
2. 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
9. https://github.com/topics/hand-gesture-recognition
10. https://www.javatpoint.com/image-processing-using-machine-learning