ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phòng Đào tạo Sau đại học & Khoa học công nghệ

TỐNG DUY TÂN

LỚP: KHOA HỌC MÁY TÍNH KHÓA 12 (ĐỢT 1)

BÁO CÁO

ĐỒ ÁN CUỐI KÌ MÔN HỌC

CHUYÊN ĐỀ THỊ GIÁC MÁY TÍNH

Giảng viên: TS. LÊ ĐÌNH DUY

TP. Hồ Chí Minh – Tháng 06/2019

MỤC LỤC

MỞ ĐẦU

Kính chào thầy,

Lời mở đầu em xin gửi lời cảm ơn chân thành đến thầy đã luôn tận tâm dạy dỗ và truyền đạt những kinh nghiệm vô cùng quý báu cho chúng em trong suốt thời gian học vừa qua. "Thị giác máy tính" có thể nói là một lĩnh khá mới mẻ đối với cá nhân em, ban đầu em đã có một chút bối rối và cảm thấy thật khó để có thể hiểu được nó một cách thấu đáo. Nhưng sau khi được thầy giải thích cụ thể từng bước một trong qui trình qua lần lượt các bài giảng, bên cạnh đó là các ví dụ minh hoạ cụ thể đã giúp những bạn từ chưa có kiến thức căn bản gì về Machine Learning như em cũng có thể dễ dàng tiếp cận với nó và nắm bắt một cách nhanh chóng những vấn đề cốt lõi như: quá trình thu thập dữ liệu, phân tích rút trích đặc trưng, xây dựng model, … bằng cách kết hợp giữa lý thuyết và các ví dụ thực tế thầy đã cho chúng em thấy được một bức tranh toàn diện về "Thị giác máy tính", nhưng đồng thời thầy cũng đi sâu vào giải thích một cách cụ thể những khía cạnh chính để tạo nên một hệ thống. Đó thật sự là những kiến thức nền tảng vô cùng bổ ích giúp em có thể tiếp tục đào sâu nghiên cứu hơn trong lĩnh vực này. Đây thật sự là một cơ hội rất tốt để nước ta có thể bắt kịp với sự phát triển nhanh chóng của khoa học, công nghệ từ các nước khác trên thế giới, các ứng dụng thông minh xuất hiện ngày càng nhiều đó cũng là cơ hội để em có thể phát triển bản thân, chính vì vậy việc nắm bắt và vận dụng tốt những kiến thức nền tảng mà thầy đã truyền đạt sẽ giúp em có thể dễ dàng tiếp cận với hướng đi mới này hơn. Em sẽ tiếp tục cố gắng nghiên cứu, tìm hiểu để có thể xây dựng nên một ứng dụng thật sự thông minh và hữu ích trong cuộc sống. Trong khuôn khổ đồ án môn học này em đã cố gắng vận dụng hết những kiến thức mà thầy đã truyền đạt, cũng như những gì mà em đã tìm hiểu được thêm trong thời gian vừa qua để xây dựng nên ứng dụng "Nhận diện tên của từng người trong một bức ảnh", đây có thể xem như là sản phẩm đầu tiên của em gửi đến thầy để thay lời cảm ơn vì tất cả những gì mà em đã được học từ thầy.

1. CÁC THÔNG TIN CƠ BẢN

- Địa chỉ github lưu source code và báo cáo: https://github.com/ozuit/computer-vision

- Địa chỉ YouTube link đến video minh hoạ cho việc cài đặt, chạy chương trình, và hiển thị kết quả: https://youtu.be/yHFRgkuWtFk

2. TỔNG QUAN VỀ ĐỒ ÁN

2.1. Mục tiêu

- Hiểu và vận dụng được những kiến thức đã học về thu thập dữ liệu, rút trích đặc trưng, xây dựng model và sử dụng các mã nguồn nền tảng sẵn có để xây dựng một ứng dụng nhận dạng khuôn mặt người đơn giản.
- Bên cạnh đó là tìm hiểu, nghiên cứu thêm các hàm thư viện khác của Python để mở rộng chương trình demo.
- Nâng cao kỹ năng lập trình bằng ngôn ngữ Python.

2.2. Nội dung và cách thức thực hiện

Chương trình demo được xây dựng hoàn toàn bằng ngôn ngữ lập trình Python, với chức năng chính là khi ta đưa vào một bức ảnh màu với nhiều khuôn mặt người trong đó thì chương trình sẽ xử lý và trả về cho ta kết quả là bức ảnh ban đầu cùng với tên của từng người tương ứng trong bức ảnh đó.

Chương trình được xây dựng bao gồm 3 phần chính:

- Phần thứ nhất là khi ta đưa vào bộ dữ liệu train bao gồm một tập các thư mục chứa ảnh chân dung mặt người, mỗi thư mục sẽ chứa ảnh của một người với tên thư mục chính là tên của người đó. Tất cả những hình ảnh đó sẽ đi qua một hàm xử lý để tiến hành rút trích các đặc trưng và tạo ra một model classifier.
- Phần thứ hai là khi ta đưa vào một bức ảnh có chứa nhiều khuôn mặt người khác nhau thì chương trình cũng sẽ tự động detect và crop ra khuôn mặt của từng người trong bức ảnh đó. Tiếp theo sẽ sử dụng model được tạo ra để tiên đoán tên của từng khuôn mặt đó.
- Phần cuối cùng là ráp tên của từng người tương ứng vào trong bức ảnh ban đầu và hiển thị kết quả ra.

3. CHI TIẾT QUÁ TRÌNH THỰC HIỆN

3.1. Thiết kế chương trình

3.1.1. Cách tổ chức dữ liệu

Gồm có 2 thư mục chính là train_data, test_data và 1 file thực thi chứa mã nguồn:

- Thư mục train_data: gồm có 23 thư mục con, mỗi thư mục con chứa hình ảnh chân dung của một người nào đó dùng làm ảnh huấn luyện. Nhãn của mỗi bức ảnh train sẽ chính là tên của thư mục con chứa bức ảnh đó.
- Thư mục test_data: gồm có tổng cộng 7 ảnh được dùng làm ảnh test, mỗi bức ảnh test sẽ bao gồm nhiều khuôn mặt người có trong tập ảnh train được chụp chung với nhau.
- File face_recognition_knn.py chứa mã nguồn thực thi ứng dụng.

3.1.2. Chức năng từng hàm trong file thực thi

- Hàm **train(train_dir, model_save_path, n_neighbors, knn_algo, verbose)**

  Ý nghĩa các thông số truyền vào:

  - Biến train_dir: truyền vào thư mục train_data bao gồm các thư mục con, mỗi thư mục con chứa hình ảnh của 1 người đã biết với tên của người đó.

- Biến model_save_path: truyền vào đường dẫn nơi để lưu trữ model sao khi train xong để có thể sử dụng lại cho lần sau mà không phải train lại.
- Biến n_neighbors: số lượng neighbors sẽ được sử dụng để classification. Nếu không truyền gì vào thì sẽ lấy ngẫu nhiên.
- Biến knn_algo: truyền vào thuật toán được hỗ trợ để tăng tốc độ tìm kiếm các neighbors gần , liên quan nhất. Mặc định sẽ là Ball Tree.
- Biến verbose: đây là một biến kiểu boolean, khi set nó bằng true thì chương trình sẽ có thông báo đến người dùng khi truyền vào một bức ảnh train mà không detect được khuôn mặt nào hoặc khi n_neighbors được chọn tự động.

Chức năng thực thi

Đầu tiên hàm này sẽ đọc qua từng thư mục con trong thư mục train_data, sau đó duyệt qua từng hình ảnh trong mỗi thư mục con đó, chỉ chấp nhận 1 khuôn mặt trong mỗi bức hình. Nếu số khuôn mặt trong mỗi bức hình nhiều hơn 1 hoặc không có khuôn mặt nào thì bức hình đó sẽ được bỏ qua và ngược lại sẽ tiến hành encoding khuôn mặt tìm thấy và thêm nó vào tập training.

```
X = []
y = []

for class_dir in os.listdir(train_dir):
    if not os.path.isdir(os.path.join(train_dir, class_dir)):
        continue

    for img_path in image_files_in_folder(os.path.join(train_dir, class_dir)):
        image = face_recognition.load_image_file(img_path)
        face_bounding_boxes = face_recognition.face_locations(image)

        if len(face_bounding_boxes) != 1:
            if verbose:
                print("Image {} not suitable for training: {}".format(img_path, "Didn't find a face" if len(face_bounding_boxes) < 1
        else:
            X.append(face_recognition.face_encodings(image, known_face_locations=face_bounding_boxes)[0])
            y.append(class_dir)
```

Tiếp theo sẽ tự động chọn số neighbors nếu như nó không được truyền vào.

```
if n_neighbors is None:
    n_neighbors = int(round(math.sqrt(len(X))))
    if verbose:
        print("Chose n_neighbors automatically:", n_neighbors)
```

Tiến hành training để tạo ra KNN classifier

```
knn_clf = neighbors.KNeighborsClassifier(n_neighbors=n_neighbors, algorithm=knn_algo, weights='distance')
knn_clf.fit(X, y)
```

Cuối cùng là lưu lại model đã tạo ra (nếu có truyền vào đường dẫn lưu trữ) và hàm này sẽ trả về model trực tiếp được sử dụng để nhận dạng khuôn mặt.

```
if model_save_path is not None:
    with open(model_save_path, 'wb') as f:
        pickle.dump(knn_clf, f)

return knn_clf
```

- Hàm **predict(X_img_path, knn_clf, model_path, distance_threshold)**

  Ý nghĩa các thông số truyền vào:

  - Biến X_img_path: truyền vào đường dẫn tới hình ảnh chứa các khuôn mặt cần nhận dạng.

  - Biến knn_clf: truyền vào một knn classifier model, nếu không bắt buộc phải có model_path.

  - Biến model_path: truyền vào đường dẫn tới một knn classifier model đã được tạo ra trước đó.

  - Biến distance_threshold: truyền vào một ngưỡng giá trị để classification khuôn mặt, nếu giá trị này càng lớn thì sẽ dễ dẫn đến việc nhận diện sai khuôn mặt bởi tên một người khác.

  Chức năng thực thi

  Đầu tiên load model đã được training lên, có thể là model được truyền trực tiếp từ hàm train hoặc được load từ ổ đĩa lưu trữ lên.

```
if not os.path.isfile(X_img_path) or os.path.splitext(X_img_path)[1][1:] not in ALLOWED_EXTENSIONS:
    raise Exception("Invalid image path: {}".format(X_img_path))

if knn_clf is None and model_path is None:
    raise Exception("Must supply knn classifier either thourgh knn_clf or model_path")

if knn_clf is None:
    with open(model_path, 'rb') as f:
        knn_clf = pickle.load(f)
```

  Tiếp theo load hình ảnh cần nhận dạng lên, tìm các khuôn mặt có trong hình ảnh và tiến hành encoding cho các khuôn mặt trong ảnh test đó.

```
X_img = face_recognition.load_image_file(X_img_path)
X_face_locations = face_recognition.face_locations(X_img)

if len(X_face_locations) == 0:
    return []

faces_encodings = face_recognition.face_encodings(X_img, known_face_locations=X_face_locations)
```

Sử dụng model KNN để tìm ra khuôn mặt phù hợp nhất, cuối cùng hàm này sẽ trả về một danh sách vị trí các khuôn mặt và tên tương ứng được nhận diện trong bức , đối với những khuôn mặt không nhận diện được sẽ được gán nhãn là "unknown".

```
closest_distances = knn_clf.kneighbors(faces_encodings, n_neighbors=1)
are_matches = [closest_distances[0][i][0] <= distance_threshold for i in range(len(X_face_locations))]

return [(pred, loc) if rec else ("unknown", loc) for pred, loc, rec in zip(knn_clf.predict(faces_encodings), X_face_locations, are_matches)]
```

- Hàm **visualize(image_file, predictions)**

Ý nghĩa các thông số truyền vào:

- Biến image_file: truyền vào đường dẫn đến hình ảnh chứa các khuôn mặt cần nhận diện.

- Biến predictions: truyền vào kết quả được tạo ra từ hàm predict.

Chức năng thực thi

Load ảnh test lên, sử dụng thư viện PIL để vẽ khung bao quanh các khuôn mặt được nhận diện và gán tên tương ứng vào từng khuôn mặt.

```
img_path = os.path.join("test_data", image_file)
pil_image = Image.open(img_path).convert("RGB")
draw = ImageDraw.Draw(pil_image)

for name, (top, right, bottom, left) in predictions:
    draw.rectangle(((left, top), (right, bottom)), outline=(0, 0, 255))
    name = name.encode("UTF-8")
    text_width, text_height = draw.textsize(name)
    draw.rectangle(((left, bottom - text_height - 10), (right, bottom)), fill=(0, 0, 255), outline=(0, 0, 255))
    draw.text((left + 12, bottom - text_height - 5), name, fill=(255, 255, 255, 255))
```

Hiển thị bức ảnh với các khuôn mặt đã được nhận diện lên và đồng thời lưu nó vào thư mục result_data.

```
pil_image.show()
pil_image.save(os.path.join("result_data", image_file))
```

3.2. Thuật toán KNN

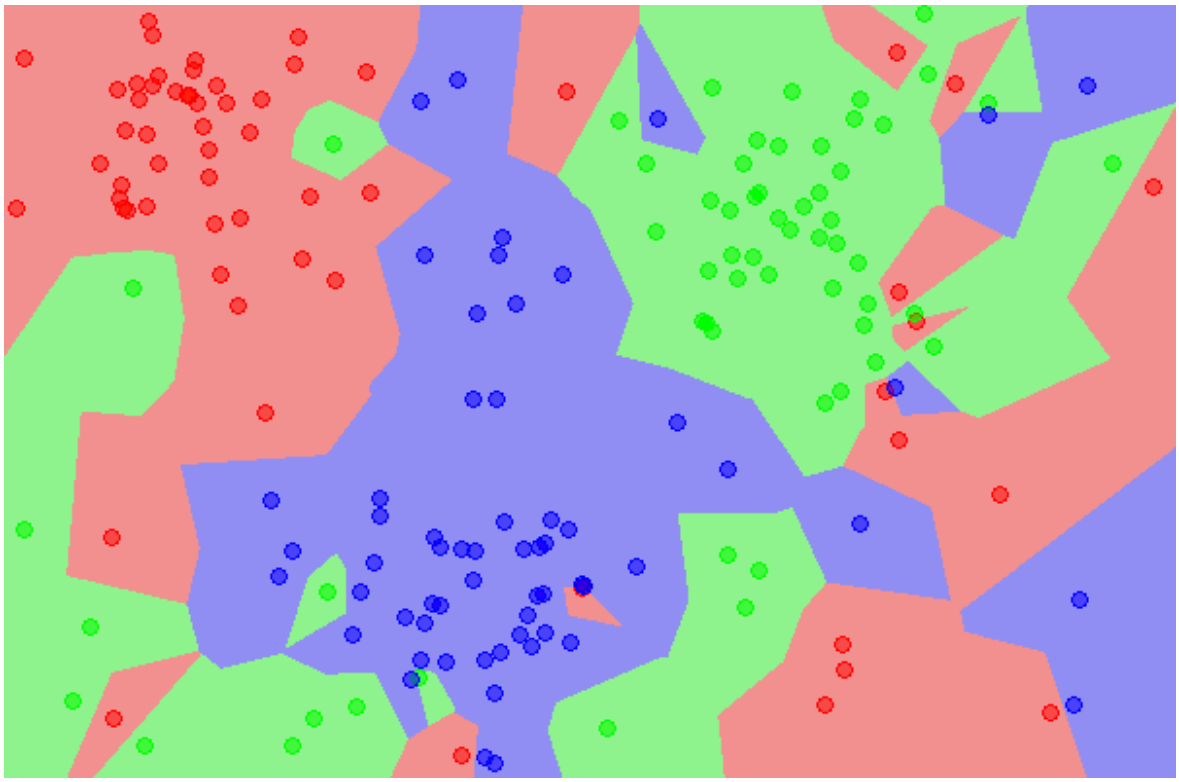
K-nearest neighbor là một trong những thuật toán supervised-learning đơn giản nhất (mà hiệu quả trong một vài trường hợp) trong Machine Learning. Khi training,

thuật toán này không học một điều gì từ dữ liệu training (đây cũng là lý do thuật toán này được xếp vào loại lazy learning), mọi tính toán được thực hiện khi nó cần dự đoán kết quả của dữ liệu mới. K-nearest neighbor có thể áp dụng được vào cả hai loại của bài toán Supervised learning là Classification và Regression. KNN còn được gọi là một thuật toán Instance-based hay Memory-based learning.

Với KNN, trong bài toán Classification, label của một điểm dữ liệu mới (hay kết quả của câu hỏi trong bài thi) được suy ra trực tiếp từ K điểm dữ liệu gần nhất trong training set. Label của một test data có thể được quyết định bằng major voting (bầu chọn theo số phiếu) giữa các điểm gần nhất, hoặc nó có thể được suy ra bằng cách đánh trọng số khác nhau cho mỗi trong các điểm gần nhất đó rồi suy ra label. Chi tiết sẽ được nêu trong phần tiếp theo.

Trong bài toán Regresssion, đầu ra của một điểm dữ liệu sẽ bằng chính đầu ra của điểm dữ liệu đã biết gần nhất (trong trường hợp K=1), hoặc là trung bình có trọng số của đầu ra của những điểm gần nhất, hoặc bằng một mối quan hệ dựa trên khoảng cách tới các điểm gần nhất đó.

Một cách ngắn gọn, KNN là thuật toán đi tìm đầu ra của một điểm dữ liệu mới bằng cách chỉ dựa trên thông tin của K điểm dữ liệu trong training set gần nó nhất (K-lân cận), không quan tâm đến việc có một vài điểm dữ liệu trong những điểm gần nhất này là nhiễu. Hình 7 là một ví dụ về KNN trong classification với K = 1.



Hình 7. Bản đồ của 1NN

Ví dụ trên đây là bài toán Classification với 3 classes: Đỏ, Lam, Lục. Mỗi điểm dữ liệu mới (test data point) sẽ được gán label theo màu của điểm mà nó thuộc về. Trong hình này, có một vài vùng nhỏ xem lẫn vào các vùng lớn hơn khác màu. Ví dụ có một điểm màu Lục ở gần góc 11 giờ nằm giữa hai vùng lớn với nhiều dữ liệu màu Đỏ và Lam. Điểm này rất có thể là nhiễu. Dẫn đến nếu dữ liệu test rơi vào vùng này sẽ có nhiều khả năng cho kết quả không chính xác.

3.3. Kết quả đạt được

Ảnh test	Số khuôn mặt có trong ảnh	Số khuôn mặt tìm thấy	Số khuôn mặt nhận dạng đúng	Số khuôn mặt unknown
IMG_5326.jpg	11	9	7	0
IMG_5327.jpg	20	18	11	1
IMG_5328.jpg	14	11	7	1
IMG_5329.jpg	11	7	6	0
IMG_5330.jpg	12	8	6	0
IMG_5331.jpg	8	7	3	0
IMG_5332.jpg	6	5	4	0

- Tổng số người có hình ảnh được sử dụng để train: 23
- Tổng số ảnh train: 56
- Tổng số khuôn mặt nhận diện đúng: 44

3.4. Kết luận

3.4.1. Ưu điểm của KNN

- Độ phức tạp tính toán của quá trình training là bằng 0.
- Việc dự đoán kết quả của dữ liệu mới rất đơn giản.
- Không cần giả sử gì về phân phối của các class.

3.4.2. Nhược điểm của KNN

- KNN rất nhạy cảm với nhiễu khi K nhỏ.
- KNN là một thuật toán mà mọi tính toán đều nằm ở khâu test. Trong đó việc tính khoảng cách tới từng điểm dữ liệu trong training set sẽ tốn rất nhiều thời gian, đặc biệt là với các cơ sở dữ liệu có số chiều lớn và có nhiều điểm dữ liệu. Với K càng lớn thì độ phức tạp cũng sẽ tăng lên.
- Ngoài ra, việc lưu toàn bộ dữ liệu trong bộ nhớ cũng ảnh hưởng tới hiệu năng của KNN.
- Tỷ lệ nhận diện chính xác không quá cao nguyên nhân do đây là phương pháp nhận dạng cổ điển dựa trên classification, với số lượng ảnh train quá ít nên dẫn đến việc nhận diện nhầm tên một số khuôn mặt. Mặt khác một số bức hình có chứa các khuôn mặt nhìn xuống hoặc nhìn lệch hẳn qua một phía (như hình dưới) nên dẫn đến việc không thể nhận diện ra hoặc nhận diện không chính xác.