ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KHOA HỌC MÁY TÍNH

ĐỒ ÁN MÔN NHẬP MÔN THỊ GIÁC MÁY TÍNH

PHÂN LOẠI CẢM XÚC KHUÔN MẶT

Nguyễn Duy Thịnh - 22521414

Lê Cường Thịnh - 22521409

Thành phố Hồ Chí Minh, 06/2024

PHÂN CÔNG CÔNG VIỆC

STT	MSSV	Họ và tên	Mức độ hoàn thành
1	22521414	Nguyễn Duy Thịnh	100%
2	22521409	Lê Cường Thịnh	100%

MỤC LỤC

CHƯƠNG 1 TỔNG QUAN VỀ BÀI TOÁN

1.1 Lí do lựa chọn đề tài

Trong thời đại công nghệ số hiện nay, việc nhận diện và phân loại cảm xúc khuôn mặt đã trở thành một lĩnh vực nghiên cứu quan trọng và hấp dẫn trong Computer Vision. Đề tài này không chỉ mang tính thách thức kỹ thuật mà còn mở ra nhiều cơ hội ứng dụng thực tiễn trong đời sống hàng ngày và nhiều lĩnh vực công nghiệp. Đây là những lý do cụ thể và chi tiết cho thấy tại sao "Phân Loại Cảm Xúc Khuôn Mặt" lại là một lựa chọn hấp dẫn và có ý nghĩa thực tiễn cao:

Tăng cường giao tiếp và hiểu nhau hơn:

- **Giao tiếp không lời:** Công nghệ này giúp máy tính hiểu được cảm xúc của người dùng thông qua các biểu cảm không lời, cải thiện hiệu quả giao tiếp giữa con người và máy móc. Điều này đặc biệt hữu ích trong các ứng dụng giáo dục trực tuyến và hội thoại video.
- **Phản hồi cảm xúc tức thời:** Nhận diện cảm xúc trong thời gian thực giúp tạo ra môi trường giao tiếp đồng cảm và hiệu quả hơn, cải thiện quan hệ công việc và cá nhân.

Hỗ trợ các công tác dịch vụ chăm sóc khách hàng: y tế, du lịch, mua bán:

- **Y tế:** Giúp theo dõi và phát hiện sớm các vấn đề tâm lý như trầm cảm hoặc lo âu, hỗ trợ quá trình chẩn đoán và điều trị.
- **Du lịch:** Nâng cao chất lượng dịch vụ thông qua việc nhận diện cảm xúc của khách hàng, nhanh chóng giải quyết các vấn đề phát sinh để cải thiện trải nghiệm.
- **Mua bán:** Tạo ra trải nghiệm mua sắm cá nhân hóa hơn bằng cách nhận diện cảm xúc của khách hàng khi họ tương tác với sản phẩm.

Ứng dụng trong công nghệ và trí tuệ nhân tạo:

- **Trí tuệ nhân tạo cảm xúc (Affective Computing):** Phát triển trí tuệ nhân tạo có khả năng hiểu và phản hồi cảm xúc của con người, mở ra các ứng dụng mới trong giáo dục, chăm sóc sức khỏe tâm lý, và robot tương tác xã hội.
- **Giao diện người-máy thông minh:** Phát triển các giao diện người-máy (HMI) thông minh hơn, điều chỉnh dựa trên cảm xúc của người dùng, như hệ thống giải trí trên xe hơi điều chỉnh theo cảm xúc của tài xế.

Nâng cao trải nghiệm người dùng trong các sản phẩm công nghệ:

- **Thiết bị cá nhân:** Cải thiện trải nghiệm người dùng trên các thiết bị cá nhân như điện thoại thông minh bằng cách điều chỉnh giao diện và nội dung hiển thị dựa trên cảm xúc.
- **Ứng dụng game:** Tăng cường trải nghiệm chơi game bằng cách phản hồi theo thời gian thực dựa trên cảm xúc của người chơi.

Hỗ trợ đào tạo và giáo dục:

- **Đào tạo nhân viên:** Đánh giá và cung cấp phản hồi về khả năng xử lý cảm xúc của nhân viên trong các lĩnh vực dịch vụ khách hàng và bán hàng, cải thiện kỹ năng giao tiếp và dịch vụ.
- **Giáo dục thông minh:** Theo dõi và phân tích cảm xúc của học sinh để điều chỉnh phương pháp giảng dạy và tài liệu học tập, tạo ra môi trường học tập hiệu quả hơn.

Cải thiện an ninh và giám sát:

- **Giám sát an ninh:** Phát hiện các hành vi đáng ngờ hoặc có nguy cơ cao trong các hệ thống giám sát an ninh, đảm bảo an toàn tại các khu vực công cộng.

- **An toàn nơi làm việc:** Giúp phát hiện dấu hiệu mệt mỏi hoặc căng thẳng ở nhân viên trong môi trường làm việc nguy hiểm, đảm bảo an toàn lao động.

Những lý do trên không chỉ nhấn mạnh tính thách thức và hấp dẫn của đề tài "Phân Loại Cảm Xúc Khuôn Mặt" mà còn cho thấy tiềm năng ứng dụng rộng rãi và sâu sắc trong nhiều lĩnh vực khác nhau của đời sống. Việc nghiên cứu và phát triển trong lĩnh vực này sẽ mang lại những đóng góp quan trọng, cải thiện chất lượng cuộc sống và thúc đẩy sự phát triển của xã hội.

1.2 Phát biểu bài toán

1.2.1 Mô tả bài toán

Trong lĩnh vực Computer Vision, việc nhận diện và phân loại cảm xúc từ biểu cảm khuôn mặt của con người là một trong những bài toán thách thức và có ý nghĩa thực tiễn cao. Bài toán này nhằm xây dựng một hệ thống có khả năng tự động phân loại các biểu cảm khuôn mặt thành các cảm xúc cụ thể, dựa trên một tập dữ liệu ảnh khuôn mặt người.

1.2.2 Input

Tập ảnh chứa khuôn mặt người: Đây là tập hợp các hình ảnh chứa khuôn mặt người mà hệ thống cần phân tích. Các hình ảnh này có thể đến từ nhiều nguồn khác nhau, như ảnh chụp, video hoặc camera giám sát.

Danh sách các nhãn cảm xúc: Đây là danh sách các cảm xúc mà hệ thống cần phân loại. Trong bài toán này, các nhãn cảm xúc bao gồm:

- 'disgust' (ghê tởm)
- 'fear' (sợ hãi)
- 'happy' (hạnh phúc)
- 'surprise' (ngạc nhiên)

1.2.3 Output

Tập ảnh chứa khuôn mặt người đã được gán nhãn cảm xúc: Đây là tập hợp các hình ảnh ban đầu nhưng mỗi ảnh đã được gán nhãn cảm xúc tương ứng. Hệ thống sẽ phân tích biểu cảm khuôn mặt trong mỗi ảnh và gán nhãn cảm xúc phù hợp từ danh sách các nhãn đã cho.

1.2.4 Mục tiêu

Mục tiêu của bài toán là xây dựng một mô hình phân loại cảm xúc khuôn mặt chính xác và hiệu quả, có khả năng:

- Nhận diện các biểu cảm khuôn mặt từ hình ảnh đầu vào.
- Phân loại chính xác biểu cảm đó thành một trong các cảm xúc đã cho.
- Cung cấp kết quả dưới dạng tập ảnh đầu vào kèm theo các nhãn cảm xúc tương ứng.

1.2.5 Thách Thức

Độ chính xác: Đảm bảo hệ thống có thể phân loại chính xác các biểu cảm khác nhau, ngay cả khi biểu cảm đó không rõ ràng hoặc bị che khuất một phần.

Đa dạng dữ liệu: Xử lý được các hình ảnh từ nhiều nguồn khác nhau với chất lượng và điều kiện ánh sáng khác nhau.

Thời gian xử lý: Đảm bảo hệ thống có thể hoạt động trong thời gian thực cho các ứng dụng yêu cầu phản hồi nhanh.

CHƯƠNG 2 PHƯƠNG PHÁP GIẢI QUYẾT BÀI TOÁN

2.1 Tiền xử lí dữ liệu

2.1.1 Phát hiện khuôn mặt

Chúng tôi sử dụng **CascadeClassifier** từ **OpenCV** để tải mô hình phát hiện khuôn mặt. Ngoài ra, hàm **detectMultiScale** được sử dụng để phát hiện các khuôn mặt trong ảnh **image**. Tham số **scaleFactor**, **minNeighbors**, **minSize** được sử dụng để điều chỉnh quá trình phát hiện khuôn mặt.

```
# Hàm để phát hiện khuôn mặt
def detect_faces(image):
    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
    faces = face_cascade.detectMultiScale(image, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))
    return faces
```

2.1.2 Trích xuất đặc trưng bằng phương pháp HOG

Để trích xuất đặc trưng từ khuôn mặt đã được phát hiện, chúng tôi sử dụng HOG (Histogram of Oriented Gradients). Các tham số của HOG như **orientations**, **pixels_per_cell**, **cells_per_block**, **block_norm**, **transform_sqrt** được thiết lập để đảm bảo rằng đặc trưng HOG được tính toán một cách hiệu quả và chính xác.

```
# Hàm để trích xuất đặc trưng HOG từ khuôn mặt
def extract_hog_features(image):
    fd = hog(image, orientations=7,
             pixels_per_cell=(8, 8),
             cells_per_block=(4, 4),
             block_norm='L2-Hys',
             transform_sqrt=False)
    return fd
```

2.1.3 Đọc dữ liệu từ thư mục

Đầu tiên, chúng tôi tiến hành đọc từng thư mục con của train_dir và test_dir, mỗi thư mục chứa các ảnh của một cảm xúc cụ thể. Trong bài toán này, chúng tôi sử dụng cv2.imread để đọc ảnh và chuyển đổi sang ảnh xám (gray). Tiếp theo, dùng hàm detect_faces để phát hiện các khuôn mặt trong ảnh xám. Đối với mỗi khuôn mặt được phát hiện, thu nhỏ kích thước về 48x48 pixel và trích xuất đặc trưng HOG của khuôn mặt này. Cuối cùng, thêm đặc trưng HOG và nhãn (emotion) tương ứng vào data và labels.

```
# Hàm để đọc dữ liệu từ thư mục
def load_data(directory):
    data = []
    labels = []
    for emotion in emotions:
        emotion_dir = os.path.join(directory, emotion)
        imagePaths = list(paths.list_images(emotion_dir))
        for imagePath in imagePaths:
            image = cv2.imread(imagePath)
            gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
            faces = detect_faces(gray)
            for (x, y, w, h) in faces:
                face = gray[y:y+h, x:x+w]
                face_resized = cv2.resize(face, (48, 48))
                hog_features = extract_hog_features(face_resized)
                data.append(hog_features)
                labels.append(emotion)
    return np.array(data), np.array(labels)
```

2.1.4 Chia dữ liệu

Sau khi đã tải và tiền xử lý dữ liệu, chúng tôi chia tập train sẵn có trong bộ dữ liệu thành tập train và tập validation để có thể dễ dàng đánh giá và sử dụng mô hình một cách hiệu quả hơn.

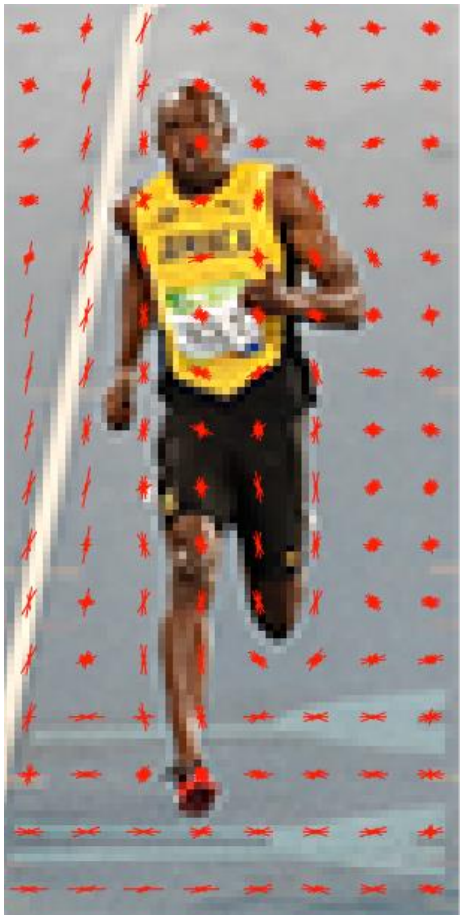
```
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size = 0.2, random_state = 42)
```

test_size = 0.2 chỉ định tỷ lệ dữ liệu được chia cho tập validation là 20%, và **random_state = 42** để đảm bảo kết quả có thể được tái lặp lại.

2.2 Trích xuất đặc trưng bằng phương pháp HOG

2.2.1 Giới thiệu

HOG là viết tắt của Histogram of Oriented Gradient - một loại "feature descriptor". Mục đích của "feature descriptor" là trừu tượng hóa đối tượng bằng cách trích xuất ra những đặc trưng của đối tượng đó và bỏ đi những thông tin không hữu ích. Vì vậy, HOG được sử dụng chủ yếu để mô tả hình dạng và sự xuất hiện của một đối tượng trong ảnh.



Bản chất của phương pháp HOG là sử dụng thông tin về sự phân bố của các cường độ gradient (intensity gradient) hoặc của hướng biên (edge directins) để mô tả các đối tượng cục bộ trong ảnh. Các toán tử HOG được cài đặt bằng cách chia nhỏ một bức ảnh thành các vùng con, được gọi là "tế bào" (cells) và với mỗi cell, ta sẽ tính toán một histogram về các hướng của gradients cho các điểm nằm trong cell. Ghép các histogram lại với nhau ta sẽ có một biểu diễn cho bức ảnh ban đầu. Để tăng cường hiệu năng nhận dạng, các histogram cục bộ có thể được chuẩn hóa về độ tương phản bằng cách tính một ngưỡng cường độ trong một vùng lớn hơn cell, gọi là các khối (blocks) và sử dụng giá trị ngưỡng đó để chuẩn hóa tất cả các cell trong khối. Kết quả sau bước chuẩn hóa sẽ là một vector đặc trưng có tính bất biến cao hơn đối với các thay đổi về điều kiện ánh sáng.

Có 5 bước cơ bản để xây dựng một vector HOG cho hình ảnh, bao gồm:

1. Tiền xử lý
2. Tính gradient
3. Tính vector đặc trưng cho từng ô (cells)
4. Chuẩn hóa khối (blocks)
5. Tính toán vector HOG

Kết quả là một vectơ đặc trưng mô tả sự phân bố của các đặc điểm cụ thể của hình dạng và cấu trúc trong hình ảnh. Vectơ này có thể được sử dụng để huấn luyện các mô hình máy học như máy vector hỗ trợ (SVM) để phân loại các đối tượng trong hình ảnh. HOG thường được kết hợp với các kỹ thuật khác như trích xuất vùng đặc trưng (region of interest) để tăng cường khả năng nhận dạng và phân loại.

2.2.2 Áp dụng vào bài toán

```
# Hàm để trích xuất đặc trưng HOG từ khuôn mặt
def extract_hog_features(image):
    fd = hog(image, orientations=7,
             pixels_per_cell=(8, 8),
             cells_per_block=(4, 4),
             block_norm='L2-Hys',
             transform_sqrt=False)
    return fd
```

- **orientations = 7:**

Đây là số lượng hướng (orientation) của gradient mà HOG sẽ tính toán trong mỗi ô pixel. Trong trường hợp này, HOG sẽ chia mỗi ô pixel thành 7 hướng để tính gradient.

- **pixels_per_cell = (8, 8):**

Kích thước của mỗi ô pixel (cell) trong đó HOG sẽ tính toán gradient. Ở đây, mỗi ô pixel có kích thước là 8x8 pixel.

- **cells_per_block = (4, 4):**

Kích thước của mỗi khối ô pixel (block) trong đó HOG sẽ thực hiện chuẩn hóa các đặc trưng. Mỗi khối bao gồm 4x4 ô pixel.

- **block_norm = 'L2-Hys':**

Phương pháp chuẩn hóa các khối ô pixel. 'L2-Hys' là phương pháp chuẩn hóa bằng cách lấy norm chuẩn L2 của các vectơ gradient và sau đó làm mềm hóa (soft-thresholding).

- **transform_sqrt = False:**

Không áp dụng phép biến đổi căn bậc hai (square root transform) lên đặc trưng HOG

Áp dụng vào bài toán nhận diện cảm xúc:

Trong bài toán này, đặc trưng HOG được áp dụng lên các khuôn mặt đã được phát hiện từ các bức ảnh. HOG giúp biểu diễn các khuôn mặt dưới dạng các vectơ đặc trưng dựa trên hướng của gradient tại từng điểm ảnh.

Việc sử dụng các tham số như **orientations**, **pixels_per_cell, cells_per_block** sẽ ảnh hưởng đến độ chi tiết và chất lượng của đặc trưng HOG.

Sau khi trích xuất đặc trưng HOG từ mỗi khuôn mặt, các vectơ đặc trưng này sẽ được sử dụng làm đầu vào cho mô hình máy học để nhận diện cảm xúc. Quá trình này giúp cải thiện khả năng phân loại và nhận diện các cảm xúc từ các ảnh khuôn mặt một cách hiệu quả.

Kết quả sau khi tính toán :

```
Number of features: 1008
Number of training samples: 2924
Number of validation samples: 731
Number of test samples: 902
```

Quá trình tính toán

Bước 1: Tính số lượng ô pixel theo kích thước ảnh

Kích thước của ảnh là 48x48 pixel và kích thước mỗi ô pixel là 8x8 pixel, do đó:

Số ô pixel theo chiều ngang: **48 / 8 = 6.**

Số ô pixel theo chiều dọc: **48 / 8 = 6.**

Bước 2: Tính số khối theo kích thước ô pixel và khối

Số ô pixel theo chiều ngang và chiều dọc đều là 6. Kích thước khối là 4x4 ô pixel, do đó:

Số khối theo chiều ngang: **6 – 4 + 1 = 3.**

Số khối theo chiều dọc: **6 – 4 + 1 = 3.**

Bước 3: Tính số lượng đặc trưng trong mỗi khối

Mỗi khối có kích thước 4x4 ô pixel và mỗi ô pixel có 7 hướng gradient. Do đó:

Số lượng đặc trưng trong mỗi khối: **4 x 4 x 7 = 112.**

Bước 4: Tính tổng số lượng đặc trưng cho toàn bộ ảnh

Tổng số khối là **3 × 3 = 9**, mỗi khối có 112 đặc trưng, do đó:

Tổng số đặc trưng: **9 × 112 = 1008.**

Như vậy, chiều dài của vector đặc trưng HOG là 1008.

2.3 Các phương pháp máy học

2.3.1 SVM

2.3.1.1 Giới thiệu

SVM (Support Vector Machine) là một trong những thuật toán học máy phổ biến được sử dụng cho các bài toán phân loại và hồi quy. SVM nổi tiếng với khả năng tìm ra mặt phẳng phân cách tốt nhất giữa các lớp trong không gian đặc trưng.

SVM hoạt động bằng cách tìm kiếm một siêu phẳng (hyperplane) trong không gian đặc trưng để phân chia các mẫu dữ liệu thuộc các lớp khác nhau. Mục tiêu là chọn siêu phẳng sao cho khoảng cách (margin) từ các điểm gần nhất thuộc hai lớp đến siêu phẳng này là lớn nhất. Những điểm dữ liệu nằm sát ranh giới quyết định này được gọi là các vector hỗ trợ (support vectors). Các khái niệm chính:

Siêu phẳng (Hyperplane):

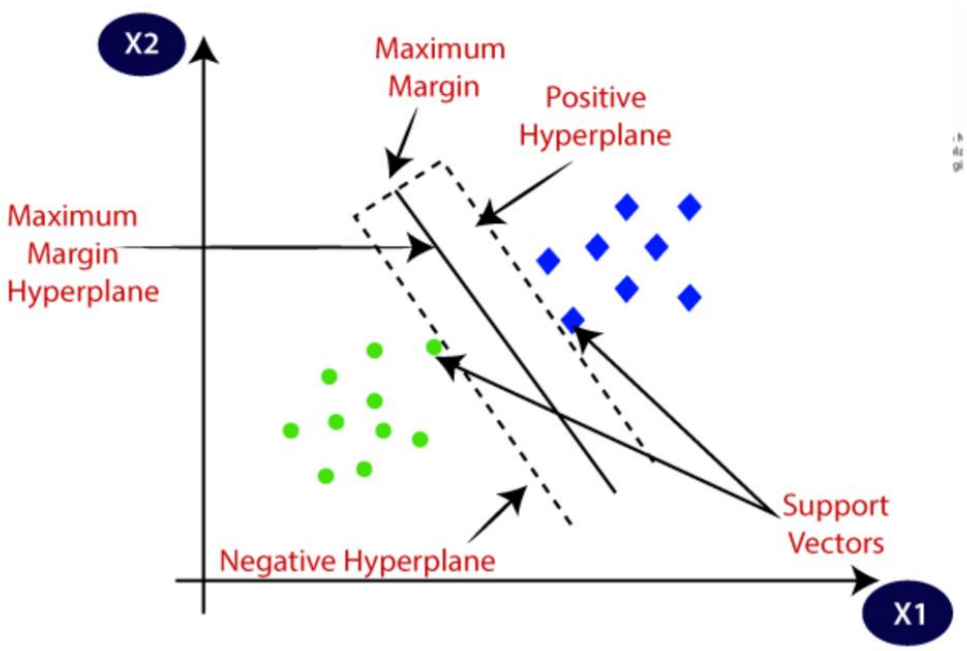
- Trong không gian 2D, siêu phẳng là một đường thẳng.
- Trong không gian 3D, siêu phẳng là một mặt phẳng.
- Trong không gian n chiều, siêu phẳng là một không gian n-1 chiều.

Margin:

- Là khoảng cách từ siêu phẳng đến các điểm dữ liệu gần nhất của mỗi lớp.
- SVM tìm cách tối ưu hóa để tối đa hóa margin này, giúp cải thiện khả năng phân loại của mô hình.

Support Vectors:

- Là các điểm dữ liệu gần nhất với siêu phẳng, đóng vai trò quyết định trong việc xác định vị trí và hướng của siêu phẳng phân loại.



Khi dữ liệu không thể phân tách tuyến tính trong không gian ban đầu, SVM sử dụng các hàm kernel để ánh xạ dữ liệu sang không gian có chiều cao hơn, nơi dữ liệu có thể trở nên phân tách tuyến tính. Một số hàm kernel phổ biến:

- **Linear Kernel:** Phù hợp khi dữ liệu có thể phân tách tuyến tính.
- **Polynomial Kernel:** Phù hợp khi dữ liệu có cấu trúc phi tuyến tính.
- **Radial Basis Function (RBF) Kernel / Gaussian Kernel:** Thường được sử dụng khi không biết trước cấu trúc của dữ liệu.
- **Sigmoid Kernel:** Thường sử dụng trong mạng nơ-ron nhân tạo.

2.3.1.2 Áp dụng vào bài toán

Để áp dụng một cách tốt nhất mô hình SVM vào bài toán, chúng tôi sử dụng Grid Search (một phương pháp tìm kiếm siêu tham số phổ biến trong máy học và học sâu).

Chúng tôi sẽ cho tham số C dò lần lượt qua 4 giá trị là 0.1, 1, 10, 100 để tìm ra khoảng tốt nhất của tham số C. Sau khi so sánh các kết quả thì C = 1 là kết quả tối ưu nhất.

```
# Sử dụng Grid Search để tìm tham số tốt nhất
param_grid = {'C': [0.1, 1, 10, 100], 'kernel': ['linear', 'rbf']}
grid_search = GridSearchCV(SVC(), param_grid, cv=3)
grid_search.fit(X_train, y_train)

# In ra tham số tốt nhất
print("Best parameters found: ", grid_search.best_params_)
```

```
# Sử dụng Grid Search để tìm tham số tốt nhất
param_grid = {'C': [1, 5], 'kernel': ['linear', 'rbf']}
grid_search = GridSearchCV(SVC(), param_grid, cv=3)
grid_search.fit(X_train, y_train)

# In ra tham số tốt nhất
print("Best parameters found: ", grid_search.best_params_)
```

```
# Sử dụng Grid Search để tìm tham số tốt nhất
param_grid = {'C': [0.1, 0.5], 'kernel': ['linear', 'rbf']}
grid_search = GridSearchCV(SVC(), param_grid, cv=3)
grid_search.fit(X_train, y_train)

# In ra tham số tốt nhất
print("Best parameters found: ", grid_search.best_params_)
```

```
# Sử dụng Grid Search để tìm tham số tốt nhất
param_grid = {'C': [0.1, 1], 'kernel': ['linear', 'rbf']}
grid_search = GridSearchCV(SVC(), param_grid, cv=3)
grid_search.fit(X_train, y_train)

# In ra tham số tốt nhất
print("Best parameters found: ", grid_search.best_params_)
```

Sau quá trình chạy thử và so sánh, Grid Search cho ra bộ tham số tốt nhất cho mô hình SVM dành cho bài toán này là:

Tham số	C	Kernel	gamma
	1	*rbf*	*scale*

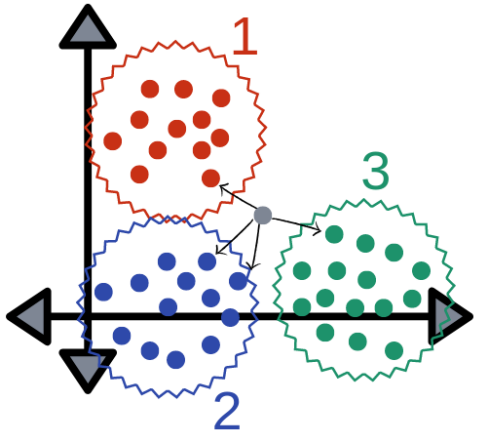
2.3.2 KNN

2.3.2.1 Giới thiệu

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.

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*.

2.3.2.2 Áp dụng vào bài toán

Tương tự như SVM ở trên, chúng tôi cũng dùng Grid Search để tìm ra bộ tham số tốt nhất cho mô hình KNN.

```
#Sử dụng Grid Search để tìm tham số tốt nhất
param_grid = {'n_neighbors': np.arange(1, 25),
              'weights': ['uniform', 'distance'],
              'metric' : ['manhattan', 'euclidean', 'cosine', 'minkowski']}
knn = KNeighborsClassifier()
knn_gscv = GridSearchCV(knn, param_grid, cv=3)
knn_gscv.fit(X_train, y_train)
```

Sau quá trình chạy thử và so sánh, Grid Search cho ra bộ tham số tốt nhất cho mô hình KNN dành cho bài toán này là:

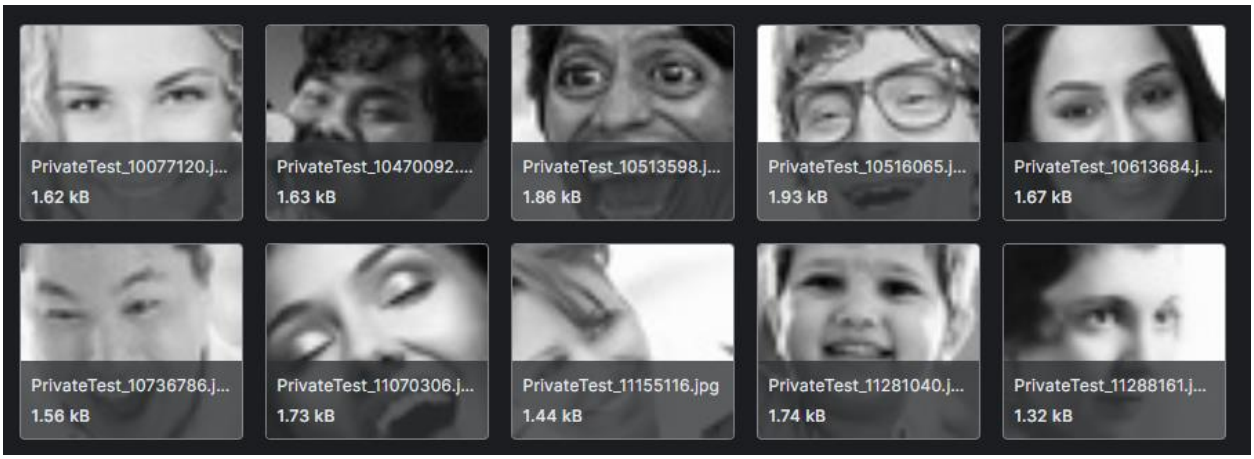
Tham số	n_neighbors	weights	metric
	3	distance	manhattan

CHƯƠNG 3 KẾT QUẢ THỰC NGHIỆM

3.1 Dataset

Việc chọn dataset là một phần quan trọng và căn bản của quá trình huấn luyện mô hình máy học hay học sâu. Dữ liệu đóng vai trò quan trọng trong việc định hình và đánh giá hiệu suất của mô hình dành cho bài toán. Chính vì thế, chúng tôi đã lựa chọn FER – 2013 để làm dataset cho bài toán của chúng tôi.

Bộ dữ liệu FER-2013 trên Kaggle là một trong những bộ dữ liệu phổ biến được sử dụng trong lĩnh vực nhận dạng cảm xúc (Emotion Recognition). Được công bố trên Kaggle vào năm 2013, đây là một bộ dữ liệu lớn chứa các hình ảnh khuôn mặt đã được gắn nhãn với các cảm xúc khác nhau.



Một số thông tin cơ bản về bộ dữ liệu:

- **Kích thước**: Dữ liệu bao gồm các hình ảnh thang độ xám 48x48 pixel của khuôn mặt. Các khuôn mặt đã được tự động đăng ký để khuôn mặt ít nhiều được căn giữa và chiếm cùng một lượng không gian trong mỗi hình ảnh. Bộ dữ liệu chứa khoảng 35,887 hình ảnh khuôn mặt với 28.709 ảnh cho tập train và 3.589 ảnh cho tập test.

- **Loại cảm xúc**: Mỗi hình ảnh được gắn nhãn với một trong 7 loại cảm xúc cơ bản: Happy (Hạnh phúc), Sad (Buồn), Angry (Tức giận), Neutral (Bình thường), Fear (Sợ hãi), Surprise (Ngạc nhiên), và Disgust (Chán ghét).
- **Phân phối**: Phân phối của các lớp cảm xúc trong bộ dữ liệu có thể không đồng đều, với một số lớp có số lượng hình ảnh ít hơn so với các lớp khác.
- **Mục đích sử dụng**: Bộ dữ liệu FER-2013 thường được sử dụng cho các nhiệm vụ nhận dạng cảm xúc và huấn luyện các mô hình học máy hoặc học sâu để tự động phân loại cảm xúc từ hình ảnh khuôn mặt.

3.2 Độ đo

Để có thể đánh giá mức độ chính xác của hai mô hình đã áp dụng cho bài toán, chúng tôi sử dụng độ đo Accuracy để làm việc này.

Độ đo "accuracy" (độ chính xác) là một trong những phép đo quan trọng nhất để đánh giá hiệu suất của mô hình. Độ đo này đơn giản là tỷ lệ giữa số lượng các dự đoán đúng và tổng số lượng mẫu trong tập dữ liệu kiểm tra.

Bên cạnh đó, Precision, Recall và F1-Score cũng được sử dụng để đánh giá chi tiết hơn

- Precision (Độ chính xác): Precision cho biết độ chính xác của các dự đoán của mô hình đối với từng lớp.
- Recall (Độ nhớ): Recall cho biết mô hình có khả năng phát hiện đúng các mẫu của từng lớp.
- F1-Score: cho biết sự cân bằng giữa precision và recall của mô hình.

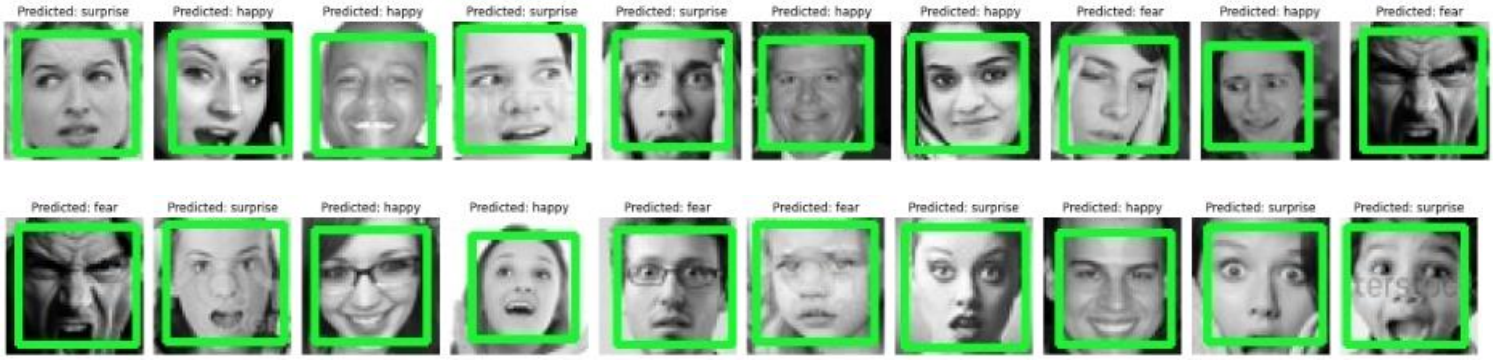
3.3 Kết quả

Chúng tôi sử dụng độ đo Accurancy để đánh giá hai mô hình SVM và KNN trên hai tập dữ liệu Validation và Test. Thu được kết quả như sau:

	Validation	Test
SVM	**80.16%**	**80.93%**
KNN	**76.88%**	**78.60%**

Kết quả accuracy khá cao trên cả tập validation và tập test cho thấy cả hai mô hình đã đạt được hiệu suất tốt trong việc phân loại.

Dưới đây là một số ảnh dự đoán đúng trong tập test của cả 2 mô hình:



Mặc dù mô hình dự đoán đúng trong nhiều trường hợp, vẫn có những ảnh mà mô hình dự đoán sai. Điều này có thể được giải thích do mô hình học cách nhận diện các đặc trưng như mắt, mũi, và đặc biệt là miệng từ bộ dữ liệu huấn luyện. Khi gặp những ảnh có khuôn miệng mở, mô hình có thể nhầm lẫn và gán nhãn cảm xúc sai, chẳng hạn như nhận định sai rằng biểu cảm là "happy" hay "surprise".

CHƯƠNG 4 ĐÁNH GIÁ NHỮNG ƯU ĐIỂM VÀ HẠN CHẾ

4.1 Ưu điểm

Cả mô hình SVM và KNN đều có khả năng hoạt động hiệu quả trên cả dữ liệu ảnh màu và ảnh xám. Việc này là rất hữu ích vì đôi khi việc sử dụng ảnh xám có thể giúp giảm chi phí tính toán và yêu cầu về dữ liệu, nhất là khi có ít tài nguyên tính toán hoặc khi dữ liệu màu không cần thiết cho bài toán cụ thể.

Thông qua kết quả đánh giá khá cao, cả hai mô hình đều đã học được các đặc trưng quan trọng từ dữ liệu để phân loại cảm xúc một cách chính xác. Thể hiện được sự linh hoạt và khả năng tổng quát hóa của mô hình, với khả năng áp dụng hiệu quả trên nhiều loại dữ liệu đầu vào khác nhau. Ngoài ra, điều này còn làm tăng tính ứng dụng của mô hình trong nhiều tình huống và môi trường khác nhau.

4.2 Hạn chế

Ảnh khuôn mặt bị vật thể che khuất: Trong một số trường hợp, các ảnh khuôn mặt có thể bị che khuất bởi các vật thể khác, như kính râm, tay, hoặc tóc. Điều này có thể dẫn đến việc mô hình không nhận diện được khuôn mặt trong ảnh, và do đó không thể phân loại được cảm xúc của người đó. Trong trường hợp này, việc tiền xử lý dữ liệu để loại bỏ hoặc giảm thiểu các vật thể che khuất có thể cần thiết để cải thiện hiệu suất của mô hình.

Ảnh khuôn mặt có kích thước quá lớn: Trong trường hợp ảnh có kích thước quá lớn, mô hình có thể không nhận diện được hoặc nhận diện sai về các đặc trưng của khuôn mặt, dẫn đến kết quả phân loại không chính xác.

Hiệu suất phân loại giữa các lớp cảm xúc có sự chênh lệch: Trong bộ dữ liệu, có sự mất cân bằng giữa các lớp cảm xúc với một hoặc một số lớp có số lượng mẫu ít hơn so với các lớp khác. Điều này có thể dẫn đến việc mô hình thiên vị hướng tới các

lớp có số lượng mẫu lớn hơn, gây ra kết quả phân loại không chính xác đối với các lớp thiểu số.

TÀI LIỆU THAM KHẢO

Sách

"Deep Learning for Computer Vision" by Rajalingappaa Shanmugamani

"Deep Learning for Computer Vision with Python" by Adrian Rosebrock

Bài báo và nghiên cứu

"Deep Residual Learning for Image Recognition" by He et al. (2016)

"Facial expression recognition using convolutional neural networks: state of the art" by Michel F. Valstar et al. (2015)

"Efficient Face and Facial Expression Recognition Algorithm Using Adaptive VSR Method" by Bhoyar et al. (2017)

IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

Dataset:

FER-2013 Dataset

CK+ (Cohn-Kanade) Dataset

Trang web và blog:

Towards Data Science

Kaggle