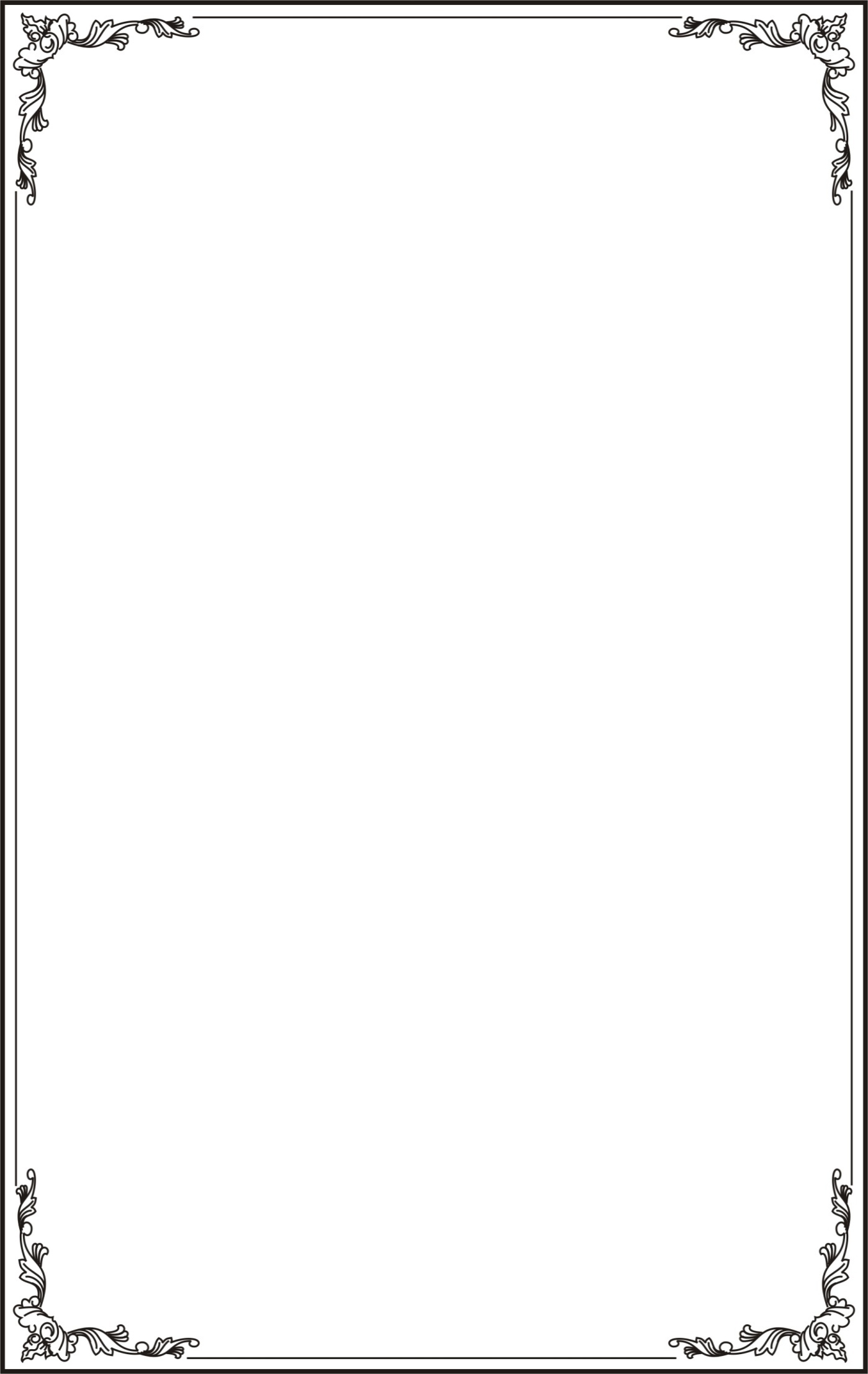
** TRƯỜNG ĐẠI HỌC KIÊN GIANG**

**KHOA THÔNG TIN & TRUYỀN THÔNG**

--------------------------------



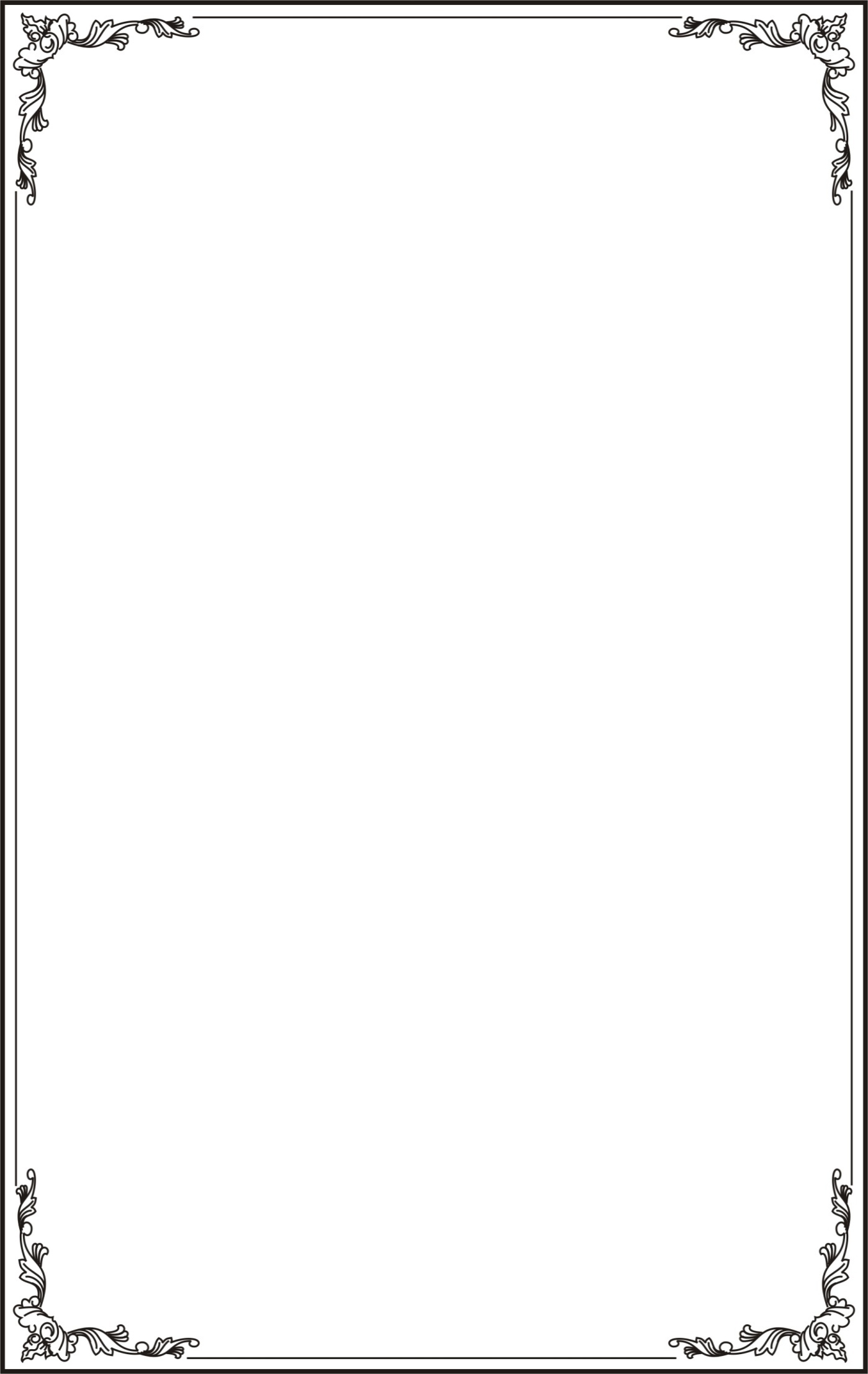
**NHẬN DIỆN KHUÔN MẶT**

**HỌC PHẦN: MÁY HỌC**

**NGÀNH: CÔNG NGHỆ THÔNG TIN  
MÃ SỐ NGÀNH: 7480201**

|  |
| --- |
| **Nhóm 9** |
| **Trần Đào Quốc Đạt - 2006206098**  **Võ Tuấn Kiệt – 2006206101** |
| **Lâm Thiện Tính – 2006206072** |
| **Quách Trùng Vương – 2006206091** |

**Kiên giang - 2023**

** TRƯỜNG ĐẠI HỌC KIÊN GIANG**

**KHOA THÔNG TIN & TRUYỀN THÔNG**

--------------------------------



**NHẬN DIỆN KHUÔN MẶT**

**HỌC PHẦN: MÁY HỌC**

**NGÀNH: CÔNG NGHỆ THÔNG TIN  
MÃ SỐ NGÀNH: 7480201**

**GIẢNG VIÊN HƯỚNG DẪN:**

**ThS. NHAN THANH NHÃ**

|  |
| --- |
| **Nhóm 9** |
| **Trần Đào Quốc Đạt - 2006206098**  **Võ Tuấn Kiệt - 2006206101** |
| **Lâm Thiện Tính - 2006206072** |
| **Quách Trùng Vương - 2006206091** |

**Kiên giang - 2023**

**LỜI CẢM ƠN**

**------------------------------------------------------------------**

Trước hết, chúng em muốn gửi lời cảm ơn chân thành đến Ban giám hiệu và Khoa Thông tin Truyền thông của trường Đại học Kiên Giang đã tạo điều kiện thuận lợi cho chúng em học tập và thực hiện đề tài này.

Cùng lúc đó, chúng em xin bày tỏ lòng biết ơn sâu sắc đến Ths.Nhan Thanh Nhã – người hướng dẫn đã tận tình chỉ bảo, truyền cảm hứng và ủng hộ chúng em trong quá trình thực hiện đề tài.

Chúng em cũng xin chân thành cảm ơn quý thầy cô trong khoa Thông tin Truyền thông đã giảng dạy, trang bị cho em nhưng kiến thức quý báu trong năm vừa qua.

Cuối cùng là xin cảm ơn sự quan tâm giúp đỡ và ủng hộ của các anh chị, bạn bè trong thời gian quá trình thực hiện khóa đề tài. Mặc dù đã cố gắng hoàn thành đề tài trong phạm vi nhưng sẽ không tránh khỏi những thiếu sót. Chúng em mong nhận được sự thống cảm và góp ý của quý thầy cô và các bạn.

**LỜI CAM KẾT**

**-----------------**

Chúng em đã đọc và hiểu về các hành vi vi phạm sự trung thực trong học thuật. Chúng em cam kết bằng danh dự cá nhân rằng nội dung tiểu luận/báo cáo này do chúng em tự thực hiện và không vi phạm yêu cầu về sự trung thực trong học thuật.

MỤC LỤC

[CHƯƠNG 1: TỔNG QUAN 1](#_Toc147436320)

[1.1 Lý do chọn đề tài 1](#_Toc147436321)

[1.2 Đối tượng hướng đến 1](#_Toc147436322)

[1.3 Phạm vi đề tài 2](#_Toc147436323)

[1.4 Mục tiêu cần đạt được 2](#_Toc147436324)

[1.4.1 Phi chức năng 2](#_Toc147436325)

[1.4.2 Chức năng 2](#_Toc147436326)

[1.5 Hướng giải quyết và kế hoạch thực hiện 3](#_Toc147436327)

[1.5.1 Hướng giải quyết 3](#_Toc147436328)

[1.5.2 Kế hoạch thực hiện 3](#_Toc147436329)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 4](#_Toc147436330)

[2.1 Khái niệm cơ bản về AI, ML và CV 4](#_Toc147436332)

[2.1.1 Trí tuệ nhân tạo (AI) 4](#_Toc147436333)

[2.1.2 Học máy (ML) 5](#_Toc147436334)

[2.1.3 Thị giác máy tính (CV) 6](#_Toc147436335)

[2.2 Khái niệm cơ bản về hình ảnh kỹ thuật số và nhận diện khuôn mặt 7](#_Toc147436336)

[2.2.1 Hình ảnh kỹ thuật số 7](#_Toc147436337)

[2.2.2 Nhận diện khuôn mặt 9](#_Toc147436338)

[2.3 Công cụ sử dụng 10](#_Toc147436339)

[2.3.1 Ngôn ngữ lập trình Python 10](#_Toc147436340)

[2.3.2 Thư viện OpenCV 11](#_Toc147436341)

[CHƯƠNG 3: XÂY DỰNG MÔ HÌNH 12](#_Toc147436342)

[3.1 Lý thuyết về các thuật toán 12](#_Toc147436344)

[3.1.1 Thuật toán Mẫu nhị phân cục bộ (Local Binary Patterns - LBP) 12](#_Toc147436345)

[3.1.2 Thuật toán K – láng giềng gần nhất (K-Nearest Neighbors - KNN) 14](#_Toc147436346)

[3.2 Triển khai thuật toán và quy trình nhận diện khuôn mặt 18](#_Toc147436347)

[3.2.1 Cấu trúc của dự án 18](#_Toc147436348)

[3.2.2 Triển khai mô hình với Python 19](#_Toc147436349)

[CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 27](#_Toc147436350)

[4.1 Kết quả đạt được 27](#_Toc147436352)

[4.2 Những thu hoạch về mặt chuyên môn 28](#_Toc147436353)

[4.3 Những khó khăn, thách thức khi phát triển mô hình 28](#_Toc147436354)

[4.4 Hướng phát triển 29](#_Toc147436355)

[Tài liệu tham khảo 30](#_Toc147436356)

**DANH MỤC HÌNH**

[Hình 2. 1Minh họa trí tuệ nhân tạo 4](#_Toc147438571)

[Hình 2. 2 Phân nhánh của AI 5](#_Toc147438572)

[Hình 2. 3 Kiểu học máy 6](#_Toc147438573)

[Hình 2. 4 Minh họa thị giá máy tính 7](#_Toc147438574)

[Hình 2. 5 Ảnh màu 8](#_Toc147438575)

[Hình 2. 6 Các mức xám 8](#_Toc147438576)

[Hình 2. 7 Ảnh xám 9](#_Toc147438577)

[Hình 2. 8 Nhận diện khuôn mặt 10](#_Toc147438578)

[Hình 2. 9 Biểu tượng và mã ví dụ của Python 11](#_Toc147438579)

[Hình 2. 10 Biểu tượng và ứng dụng của OpenCV 11](#_Toc147438580)

[Hình 3. 1 Ảnh trước và sau khi thực hiện LBP 12](#_Toc147438581)

[Hình 3. 2 Chọn điểm ảnh trung tâm 13](#_Toc147438582)

[Hình 3. 3 Phân ngưỡng các điểm ảnh lân cận 13](#_Toc147438583)

[Hình 3. 4 Thu thập mẫu nhị phân theo chiều kim đồng hồ 13](#_Toc147438584)

[Hình 3. 5 Minh họa KNN 15](#_Toc147438585)

[Hình 3. 6 Chuyển đổi ma trận thành vector 1 chiều 16](#_Toc147438586)

[Hình 3. 7 Công thức khoảng cách Euclidean 16](#_Toc147438587)

[Hình 3. 8 Cấu trúc của dự án 18](#_Toc147438588)

[Hình 3. 9 Thư mục con của dataset và các ảnh trong thư mục con 19](#_Toc147438589)

[Hình 3. 10 Tệp lưu trữ các vector đặc trưng và nhãn 19](#_Toc147438590)

[Hình 3. 11 Mã nguồn contants.py 20](#_Toc147438591)

[Hình 3. 12 Một phần mã nguồn local\_binary\_pattern.py 21](#_Toc147438592)

[Hình 3. 13 Một phần mã nguồn local\_binary\_pattern.py 22](#_Toc147438593)

[Hình 3. 14 Một phần mã nguồn local\_binary\_pattern.py 23](#_Toc147438594)

[Hình 3. 15 Một phần mã nguồn tệp preprocessing.py 23](#_Toc147438595)

[Hình 3. 16 Một phần mã nguồn tệp preprocessing.py 24](#_Toc147438596)

[Hình 3. 17 Mã nguồn tệp image\_classifier.py 25](#_Toc147438597)

[Hình 3. 18 Các vectors đặc trưng và nhãn tương ứng 26](#_Toc147438598)

[Hình 3. 19 Mã nguồn tệp face\_recognition.py 27](#_Toc147438599)

[Hình 4. 1 Giao diện sau khi chạy preprocessing.py 28](#_Toc147438600)

[Hình 4. 2 Hình ảnh trong processed\_dataset 28](#_Toc147438601)

[Hình 4. 3 Giao diện sau khi chạy image\_classifier,py 28](#_Toc147438602)

[Hình 4. 4 Giao diện khi chạy face\_recognition.py 29](#_Toc147438603)

**DANH MỤC TỪ VIẾT TẮC**

**-----------------**

|  |  |
| --- | --- |
| **Viết tắc** | **Nội dung đầy đủ** |
| **Tiếng anh** |  |
| AI | Artificial intelligence |
| ML | Machine learning |
| CV | Computer vison |
| LBP | Local binary pattern |
| KNN | K – Nearest neighbors |
|  |  |
|  |  |

# CHƯƠNG 1: TỔNG QUAN

## **Lý do chọn đề tài**

Trong thời gian gần đây, Trí tuệ nhân tạo (Artificial Intelligence - AI) đã trở thành một chủ đề nóng hổi và đầy thú vị trong các lĩnh vực công nghệ và xã hội. AI đang thúc đẩy sự tiến bộ và thay đổi cách chúng ta sống và làm việc. AI đã trải qua một sự gia tăng rõ rệt trong thập kỷ qua. Điều này được hỗ trợ bởi sự phát triển của phần cứng mạnh mẽ, khả năng tính toán đám mây và sự tiến bộ trong thuật toán học máy và học sâu. Các phân nhánh của công nghệ AI như: máy học (Machine Learning - ML), thị giác máy tính (Computer Vision – CV), xử lý ngôn ngữ tự nhiên (Natural Language Processing – NLP) và robot học (Robotics) đang trở nên ngày càng phổ biến và ảnh hưởng mạnh mẽ đến nhiều lĩnh vực khác nhau.

Trong đó, tiêu biểu nhất có thể kể đến đó là ML – một phân nhánh quan trọng của AI. ML là một phương pháp phân tích dữ liệu dựa trên khái niệm rằng máy tính có khả năng “học” và cải thiện hiệu suất từ dữ liệu mà nó nhận được. Thay vì lập trình cụ thể, ML cho phép máy tính học từ mẫu dữ liệu và tạo ra các mô hình dự đoán hoặc phân loại dựa trên những gì nó đã học.

Nhận diện khuôn mặt (Face Recognition) được xem là một triển khai cụ thể của ML và AI. Công nghệ nhận diện khuôn mặt đóng vai trò vô cùng quan trọng trong việc xác định và xác minh danh tính của con người dựa trên đặc điểm khuôn mặt. Công nghệ này là một phương pháp mạnh mẽ giúp ngăn chặn truy cập trái phép vào các hệ thống và khu vực nhạy cảm như ngân hàng, cơ quan chính phủ. Ngoài ra, còn giúp người dùng truy cập một cách tiện lợi và dễ dàng, thay vì phải nhớ các mật khẩu phức tạp thì chỉ cần nhìn vào camera là có thể mở khóa thiết bị và truy cập hệ thống, từ đó giúp tiết kiệm thời gian và tăng trải nghiệm người dùng.

Từ những lợi ích to lớn trên, nhóm em đã quyết định chọn đề tài “Xây dựng mô hình nhận diện khuôn mặt” để nghiên cứu và hoàn thành đề tài học phần với triển vọng thực tiễn cao.

## **Đối tượng hướng đến**

Đối tượng hướng đến của “Mô hình nhận diện khuôn mặt” là:

* Đối với cá nhân: việc nhận diện khuôn mặt giúp xác thực danh tính, bảo mật thông tin cá nhân trong các giao dịch trực tuyến hoặc thanh toán điện tử.
* Với doanh nghiệp, mô hình này đem lại hiệu quả khi kiểm soát truy cập vào môi trường làm việc, theo dõi giờ vào ra của nhân viên. Bên cạnh đó, nhận diện khuôn mặt còn giúp cá nhân hóa chiến dịch quảng cáo phù hợp với từng đối tượng.
* Đối với các ngân hàng, nhận diện khuôn mặt góp phần nâng cao an ninh, ngăn chặn gian lận trong giao dịch trực tuyến. Mô hình cũng hữu ích cho các cảng hàng không, bến tàu trong việc kiểm tra danh tính hành khách.
* Bên cạnh đó, nhận diện khuôn mặt còn được áp dụng trong lĩnh vực giáo dục, y tế, an ninh quốc phòng để giải quyết những bài toán thực tiễn.

## **Phạm vi đề tài**

Đề tài ngày tập trung vào việc phát triển một mô hình nhận diện khuôn mặt đơn giản với ngôn ngữ python và các thư viện mạnh mẽ về ML và CV. Mô hình này sử dụng phương pháp dựa trên đặc trưng (Feature-Based) trong học máy truyền thống (Traditional Machine Learning). Qua đó, phương pháp này kết hợp giữa ML và CV để nhận diện và dự đoán nhãn của ảnh dựa trên việc xác định, trích xuất đặc trưng và so sánh các đặc trưng của hình ảnh để đưa ra dự đoán.

## **Mục tiêu cần đạt được**

### **Phi chức năng**

* Hiệu suất của mô hình này phải được tối ưu, bởi vì tập dữ liệu (Dataset) hình ảnh là rất lớn.
* Mô hình này phải cung cấp được một giao diện trừu tượng để có thể dễ dàng tích hợp vào các hệ thống khác nhau.
* Mã nguồn của mô hình cũng phải được tổ chức hiệu quả, giúp cho việc bảo trì và mở rộng mô hình sau này.

### **Chức năng**

* Chuẩn hóa dữ liệu: với dataset đầu vào, mô hình này cần phải chuẩn hóa được các hình ảnh bằng cách chỉ lấy các khuôn mặt từ ảnh và loại bỏ đi các dữ liệu khác của ảnh.
* Trích xuất đặc trưng và gán nhãn: với dataset đã được chuẩn hóa, mô hình phải trích xuất được các đặc trưng và gán nhãn tương ứng cho ảnh. Sau đó là lưu lại các đặc trưng vào 1 tập tin để sử dụng sau này.
* Nhận dạng: sau khi có được các đặc trưng của ảnh, mô hình này cần phải dự đoán được nhãn của hình ảnh đầu vào và xác suất của dự đoán đó.

## **Hướng giải quyết và kế hoạch thực hiện**

### **Hướng giải quyết**

* Xây dựng mô hình nhận diện khuôn mặt đơn giản bằng Python với các framework về ML và CV.

### **Kế hoạch thực hiện**

* Khảo sát và tìm hiểu về các mô hình tương tự đã được tạo ra trước đó.
* Thu thập các dữ liệu hình ảnh từ internet.
* Lập ra quy trình xử lý dữ liệu cho mô hình:
  + Tiền xử lý dữ liệu: sử dụng mô hình Haar Cascade được cung cấp bởi OpenCV để phát hiện khuôn mặt và cắt ra khuôn mặt từ hình ảnh, sau đó là lưu lại vào dataset để sử dụng.
  + Trích xuất đặc trưng và gán nhãn: áp dụng thuật toán mẫu nhị phân cục bộ (Local Binary Pattern – LBP) để trích xuất được các vector đặc trưng và gán nhãn tương ứng cho các vector đặc trưng đó.
  + Nhận diện và đưa ra dự đoán: áp dụng thuật toán K láng giềng gần nhất (K-Nearest Neighbors – KNN) để phân loại và dự đoán nhãn cũng như là xác suất cho 1 hình ảnh đầu vào.
* Triển khai từng giai đoạn của mô hình bằng Python.
* Sửa đổi và cải thiện hiệu suất cho mô hình.
* Viết tài liệu hướng dẫn sử dụng.

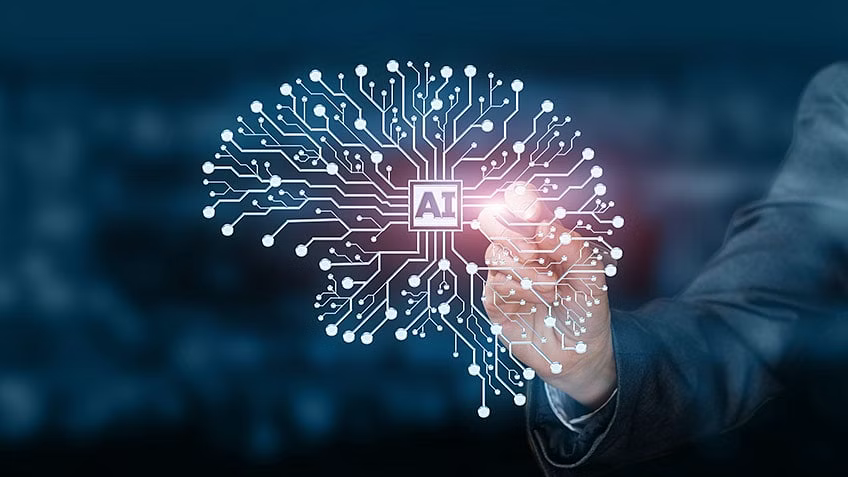
# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT



## **Khái niệm cơ bản về AI, ML và CV**

### **Trí tuệ nhân tạo (AI)**

AI là một lĩnh vực trong ngành khoa học máy tính nhằm nghiên cứu và phát triển các hệ thống và chương trình có khả năng tự động hoá các nhiệm vụ thông minh mà trước đây chỉ được thực hiện bởi con người. Mục tiêu của AI là tạo ra các hệ thống có khả năng học hỏi, suy luận, phân tích thông tin, và ra quyết định tương tự như con người.

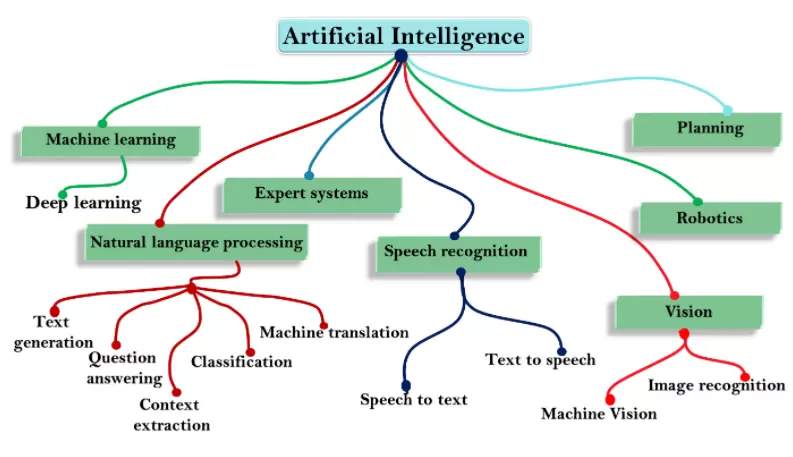


Hình 2. Minh họa trí tuệ nhân tạo

AI sử dụng các phương pháp và kỹ thuật từ nhiều lĩnh vực khác nhau như khoa học máy tính, toán học, thống kê, và sinh học. Các thuật toán và mô hình AI có thể xử lý dữ liệu lớn, tìm kiếm mẫu, phân loại, và dự đoán kết quả dựa trên dữ liệu đầu vào.

AI được chia thành nhiều phân nhánh khác nhau:

* ML: Tập trung vào xây dựng các mô hình và thuật toán để cho phép máy tính học hỏi từ dữ liệu và cải thiện hiệu suất theo thời gian.
* CV: Nghiên cứu và phát triển các thuật toán và công nghệ để máy tính có thể hiểu và xử lý hình ảnh và video.
* NLP: Tập trung vào việc xây dựng các mô hình và công nghệ để máy tính có thể hiểu, xử lý và tương tác với ngôn ngữ tự nhiên của con người.
* Robotics: tập trung vào phát triển các hệ thống robot có khả năng tự động hoá và tương tác với môi trường xung quanh
* Còn nhiều phân nhánh khác như: Hệ thống khuyến nghị (Recommendation Systems), Lý thuyết trò chơi (Game Theory)…



Hình 2. Phân nhánh của AI

### **Học máy (ML)**

ML là một phân nhánh của AI, liên quan đến việc nghiên cứu và xây dựng các kĩ thuật cho phép các hệ thống "học" tự động từ dữ liệu để giải quyết những vấn đề cụ thể. Các thuật toán học máy xây dựng một mô hình dựa trên dữ liệu mẫu, được gọi là dữ liệu huấn luyện, để đưa ra dự đoán hoặc quyết định mà không cần được lập trình chi tiết về việc đưa ra dự đoán hoặc quyết định này.

ML dựa trên ý tưởng rằng máy tính có thể học từ dữ liệu và trải nghiệm để thực hiện các tác vụ cụ thể. Quá trình “học” trong ML bao gồm ba thành phần chính: dữ liệu huấn luyện, mô hình và thuật toán.

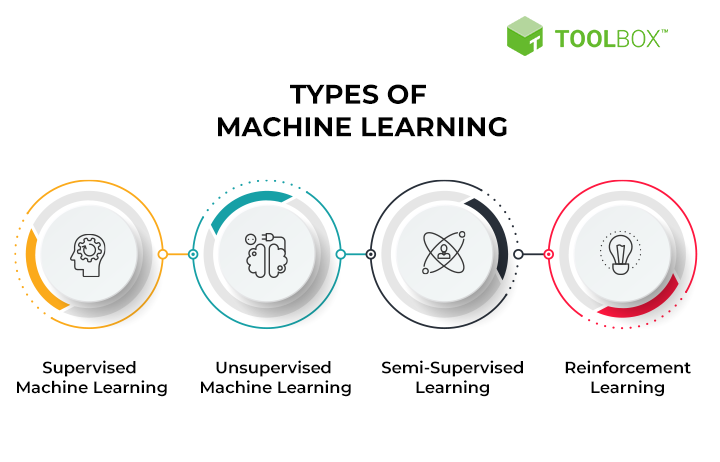
Đầu tiên, một tập dữ liệu huấn luyện được chuẩn bị, trong đó chứa các ví dụ hoặc mẫu dữ liệu có thông tin đầu vào và đầu ra tương ứng. Dữ liệu huấn luyện này được sử dụng để "huấn luyện" mô hình, tức là điều chỉnh các tham số của mô hình để tối ưu hóa hiệu suất.

Tiếp theo, một mô hình máy tính được chọn hoặc thiết kế để thực hiện tác vụ cần giải quyết. Mô hình này có thể là một hệ thống logic hoặc mạng nơ-ron nhân tạo (neural network) với các lớp và liên kết giữa chúng.

Cuối cùng, một thuật toán được áp dụng để điều chỉnh mô hình dựa trên dữ liệu huấn luyện và mục tiêu của tác vụ. Thuật toán này sẽ tối ưu hóa các tham số của mô hình dựa trên mức độ sai số giữa đầu ra dự đoán và đầu ra thực tế.

ML được chia thành 4 loại chính:

* Học có giám sát (Supervised Learning): mô hình được huấn luyện bằng cách sử dụng các cặp dữ liệu đầu vào và đầu ra tương ứng. Mục tiêu là xây dựng một mô hình có thể dự đoán đầu ra cho các dữ liệu mới dựa trên các mẫu đã biết trước.
* Học không giám sát (Unsupervised Learning): Trong học không giám sát, không có dữ liệu huấn luyện với đầu ra cho sẵn. Mục tiêu là tìm hiểu cấu trúc, mô hình hoặc mối quan hệ trong dữ liệu.
* Học bán giám sát (Semi-Supervised Learning): Kết hợp cả dữ liệu có nhãn và dữ liệu không nhãn để huấn luyện mô hình. Mục tiêu là sử dụng thông tin từ dữ liệu không nhãn để cải thiện hiệu suất.
* Học tăng cường (Reinforcement Learning): Trong học tăng cường, mô hình tương tác với một môi trường động và nhận phần thưởng hoặc hình phạt dựa trên hành động của nó. Mục tiêu là tìm hiểu cách chọn hành động để tối đa hóa tổng lợi ích lâu dài. Học tăng cường thường được áp dụng trong các lĩnh vực như robot học và trò chơi máy tính.
* Các loại khác như: Chuyển đổi, Học cách học…



Hình 2. Kiểu học máy

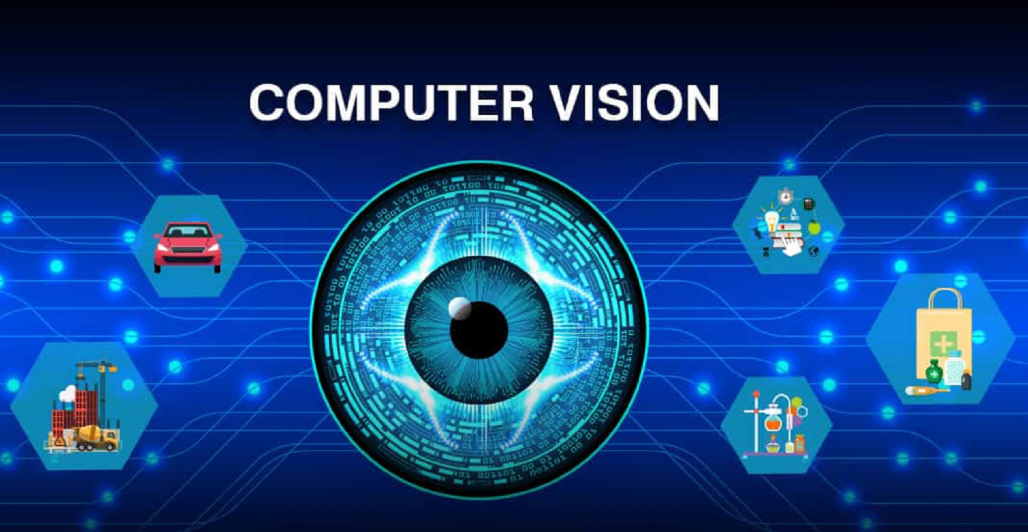
### **Thị giác máy tính (CV)**

CV là một phân nhánh của AI liên quan đến việc thu nhận, phân tích, xử lý và hiểu hình ảnh và video bằng các thuật toán và phương pháp máy tính. CV nhằm mục tiêu giúp máy tính có khả năng "nhìn" và "hiểu" thế giới xung quanh giống như con người.

Mục tiêu chính của CV là giúp máy tính có khả năng nhận diện và hiểu các đối tượng, định vị vị trí, phân tích nội dung và tương tác với thế giới xung quanh thông qua hình ảnh và video. Để đạt được điều này, CV sử dụng các phương pháp và thuật toán trong xử lý hình ảnh, trích xuất đặc trưng, phân loại, nhận dạng, và phân tích dữ liệu.

Một số ứng dụng phổ biến của CV bao gồm:

* Nhận dạng đối tượng: Máy tính có thể nhận diện và phân loại các đối tượng trong hình ảnh hoặc video, chẳng hạn như nhận diện khuôn mặt, xe hơi, động vật, vật thể trong các ứng dụng an ninh, giao thông và giám sát.
* Phân tích hành vi: CV có thể xác định và phân tích hành vi của con người hoặc đối tượng trong hình ảnh hoặc video, như theo dõi chuyển động, phát hiện hành vi nghi ngờ trong các ứng dụng an ninh và quản lý.
* Tự động hóa công việc: Máy tính có thể thực hiện các nhiệm vụ tự động dựa trên thông tin hình ảnh hoặc video, chẳng hạn như phát hiện và đếm đối tượng, kiểm tra chất lượng sản phẩm trong quá trình sản xuất.



Hình 2. Minh họa thị giá máy tính

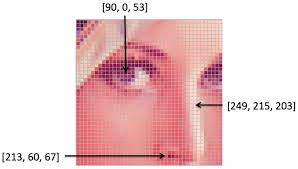
Để đạt được những ứng dụng này, CV sử dụng nhiều công cụ và kỹ thuật, bao gồm xử lý hình ảnh, trích xuất đặc trưng, mạng nơ-ron nhân tạo và ML.

## **Khái niệm cơ bản về hình ảnh kỹ thuật số và nhận diện khuôn mặt**

### **Hình ảnh kỹ thuật số**

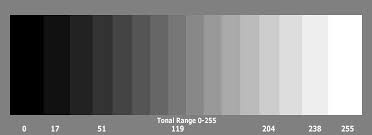
Hình ảnh kỹ thuật số là một dạng biểu diễn số học của hình ảnh, trong đó thông tin hình ảnh được biểu thị dưới dạng các giá trị số hoặc pixel. Nó là một tập hợp các điểm ảnh được sắp xếp thành một lưới hai chiều, trong đó mỗi điểm ảnh được gọi là pixel (picture element).

Mỗi pixel trong hình ảnh kỹ thuật số có thể chứa thông tin về màu sắc, gồm ba thành phần chính: đỏ (Red), xanh lá cây (Green), và xanh dương (Blue), được gọi là mô hình màu RGB. Khi kết hợp các giá trị của ba thành phần màu này, ta có thể tạo ra một loạt các màu sắc khác nhau trong hình ảnh.



Hình 2. Ảnh màu

Mỗi pixel trong hình ảnh kỹ thuật số được mô tả bằng các giá trị số, thường là các số nguyên trong khoảng từ 0 đến 255. Giá trị này thể hiện mức độ sáng tối của điểm ảnh tương ứng trong hình ảnh. Với 8 bit, ta có thể biểu diễn 256 mức xám khác nhau từ đen đến trắng.



Hình 2. Các mức xám

Ảnh xám là một loại hình ảnh kỹ thuật số trong đó mỗi pixel chỉ chứa thông tin về mức độ sáng tối của điểm ảnh tương ứng, không bao gồm thông tin về màu sắc.

Ảnh xám thường được tạo ra từ ảnh màu bằng cách chuyển đổi các giá trị màu RGB của mỗi pixel thành giá trị xám duy nhất. Có nhiều phương pháp chuyển đổi, phổ biến nhất là chuyển đổi trung bình, trong đó giá trị xám của mỗi pixel được tính bằng trung bình của các giá trị màu RGB của pixel đó.



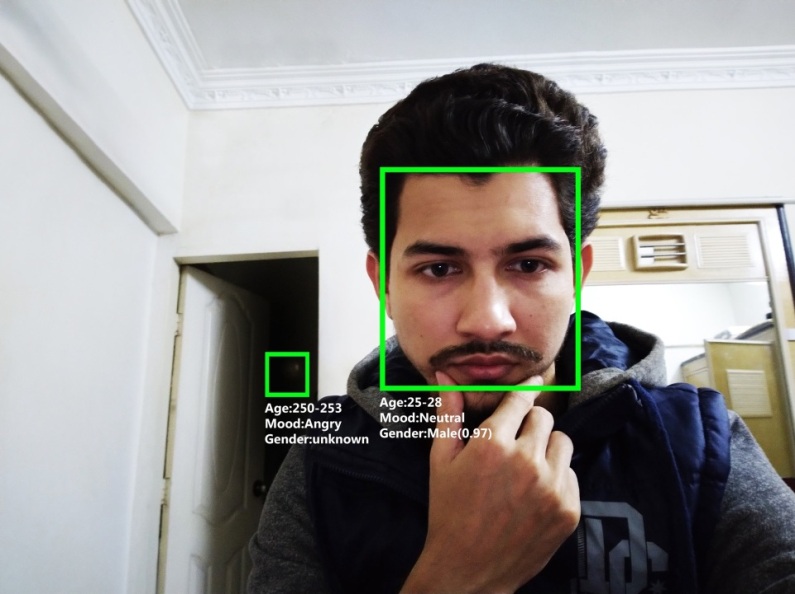
Hình 2. Ảnh xám

### **Nhận diện khuôn mặt**

Nhận diện khuôn mặt là một ứng dụng quan trọng trong lĩnh vực CV và ML, nhằm xác định và nhận biết các đặc điểm và định danh của khuôn mặt con người từ hình ảnh hoặc video. Mục tiêu chính của nhận diện khuôn mặt là xác định và phân loại các đặc trưng độc nhất của mỗi khuôn mặt để tạo ra một biểu diễn số học đại diện cho mỗi người.

Quá trình nhận diện khuôn mặt bao gồm nhiều bước:

* Phát hiện khuôn mặt: Trước khi có thể nhận diện khuôn mặt, cần phải xác định vị trí và phạm vi của khuôn mặt trong hình ảnh hoặc video. Điều này thường được thực hiện bằng cách sử dụng các thuật toán phát hiện khuôn mặt, như Haar cascades hoặc mạng nơ-ron nhân tạo.
* Trích xuất đặc trưng: Sau khi xác định được vị trí khuôn mặt, các đặc trưng độc nhất của khuôn mặt được trích xuất từ hình ảnh. Các đặc trưng này có thể bao gồm hình dạng, màu sắc, chi tiết về mắt, mũi và miệng. Các thuật toán và phương pháp phổ biến để trích xuất đặc trưng trong nhận diện khuôn mặt bao gồm Principal Component Analysis (PCA), Local Binary Patterns (LBP) và Histogram of Oriented Gradients (HOG).
* Phân loại và xác định: Sau khi trích xuất các đặc trưng của khuôn mặt, các biểu diễn số học của khuôn mặt được so khớp với các dữ liệu đã được đào tạo trước đó. Điều này thường được thực hiện bằng cách sử dụng các thuật toán phân loại như Support Vector Machines (SVM), k-Nearest Neighbors (k-NN) hoặc mạng nơ-ron nhân tạo.
* Đối sánh và nhận dạng: Khi một khuôn mặt mới được phát hiện và trích xuất đặc trưng, quá trình đối sánh và nhận dạng được thực hiện để so sánh với các biểu diễn số học đã được lưu trữ trước đó của các khuôn mặt đã biết. Kết quả của quá trình này là xác định xem khuôn mặt mới thuộc về ai trong cơ sở dữ liệu đã biết hoặc phát hiện xem đây là một khuôn mặt mới.



Hình 2. Nhận diện khuôn mặt

## **Công cụ sử dụng**

### **Ngôn ngữ lập trình Python**

Python là một ngôn ngữ lập trình được sử dụng rộng rãi trong các ứng dụng web, phát triển phần mềm, khoa học dữ liệu và máy học. Các nhà phát triển sử dụng Python vì nó hiệu quả, dễ học và có thể chạy trên nhiều nền tảng khác nhau. Phần mềm Python được tải xuống miễn phí, tích hợp tốt với tất cả các loại hệ thống và tăng tốc độ phát triển. Python là một ngôn ngữ lập trình đa mẫu hình, nó hỗ trợ hoàn toàn mẫu lập trình hướng đối tượng và lập trình cấu trúc; ngoài ra về mặt tính năng, Python cũng hỗ trợ lập trình hàm và lập trình hướng khía cạnh. Nhờ vậy mà Python có thể làm được rất nhiều thứ, sử dụng trong nhiều lĩnh vực khác nhau.

|  |  |
| --- | --- |
| D:\Desktop\Python-Symbol.png | D:\Desktop\DataScience.png |

Hình 2. Biểu tượng và mã ví dụ của Python

### **Thư viện OpenCV**

OpenCV là một thư viện mã nguồn mở chuyên về xử lý ảnh và thị giác máy tính, được phát triển để cung cấp các công cụ và thuật toán mạnh mẽ cho việc xử lý và phân tích các tác vụ liên quan đến hình ảnh và video. OpenCV cũng cung cấp các thuật toán mạnh mẽ để trích xuất đặc trưng từ hình ảnh, chẳng hạn như phát hiện biên, phát hiện góc, phát hiện đường viền, phân đoạn hình ảnh và nhận dạng vật thể. Các đặc trưng này sau đó có thể được sử dụng làm đầu vào cho các thuật toán học máy như SVM Máy vector hỗ trợ (Support Vector Machines - SVM), Random Forests hoặc mạng nơ-ron sâu, để thực hiện các tác vụ như phân loại hình ảnh, nhận dạng đối tượng hoặc phát hiện và phân loại khuôn mặt.

|  |  |
| --- | --- |
| D:\Desktop\download.png | D:\Desktop\opencv-object-recognition.png |

Hình 2. Biểu tượng và ứng dụng của OpenCV

# CHƯƠNG 3: XÂY DỰNG MÔ HÌNH

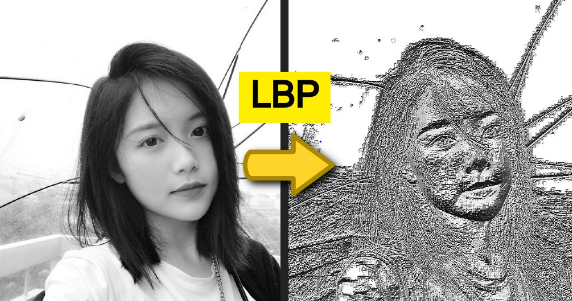


## **Lý thuyết về các thuật toán**

### **Thuật toán Mẫu nhị phân cục bộ (Local Binary Patterns - LBP)**

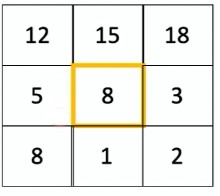
LBP là một phương pháp biểu diễn đặc trưng hình ảnh phổ biến trong lĩnh vực xử lý ảnh và nhận dạng khuôn mặt.

Ý tưởng cơ bản của LBP là mô tả đặc trưng cục bộ của mỗi pixel trong hình ảnh bằng cách so sánh giá trị của pixel đó với các pixel xung quanh nó. Quá trình này được thực hiện bằng cách so sánh giá trị của pixel với các pixel lân cận và tạo ra một chuỗi nhị phân, trong đó mỗi bit biểu diễn kết quả so sánh (1 nếu giá trị pixel lân cận lớn hơn hoặc bằng, 0 nếu nhỏ hơn). Chuỗi nhị phân này được gọi là "mẫu nhị phân cục bộ" của pixel.



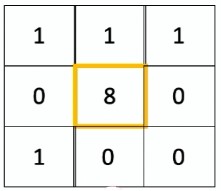
Hình 3. Ảnh trước và sau khi thực hiện LBP

Ví dụ ta có 1 hình ảnh đã được chuyển đổi sang ảnh xám. Hình này được biễu diện dưới dạng 1 ma trận pixcel, ma trận này gồm 3 hàng và 3 cột, mỗi ô vuông tương ứng với 1 pixel. Bắt đầu với việc chọn 1 pixel ngay vị trí [1, 1] có giá trị là 8, pixel được chọn sẽ được gọi là pixel trung tâm và ta áp dụng điều kiện là: Trong 8 pixel lân cận pixel trung tâm, nếu pixel nào có giá trị lớn hơn giá trị của pixel trung tâm thì được phân ngưỡng thành 1, ngược lại sẽ được phân ngưỡng thành 0.



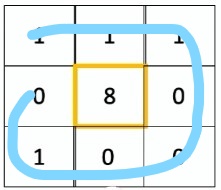
Hình 3. Chọn điểm ảnh trung tâm

Sau khi áp dụng điều kiện trên thì ma trận trông như thế này:



Hình 3. Phân ngưỡng các điểm ảnh lân cận

Lưu ý là việc phân ngưỡng thành 1 hoặc 0 không phải là gán lại giá trị cho các điểm ảnh lân cận đó, mà các giá trị phân ngưỡng đó chỉ được sử dụng tạm thời để xác định mẫu nhị phân cục bộ (tức là sẽ dùng một mảng khác để lưu các giá trị phân ngưỡng). Sau khi áp dụng điều kiện, ta có được 1 dãy 8 bit – dãy này được gọi là mẫu nhị phân cục bộ với độ dài 8 bit. Dãy này được thu thập bằng cách lấy các bit theo chiều kim đồng hồ hoặc ngược chiều kim đồng hồ. Tọa độ bắt đầu lấy có thể là 1 vị trí bất kỳ trong 8 tọa độ của 8 điểm ảnh. Như ví dụ của chúng ta là sẽ lấy theo chiều kim đồng hồ bắt đầu từ tọa độ [0, 0] -> [0, 1] -> [0, 2] -> [1, 2] …. [1, 0].



Hình 3. Thu thập mẫu nhị phân theo chiều kim đồng hồ

Sau khi có được dãy nhị phân, chúng ta tiến hành chuyển đổi dãy nhị phân này sang giá trị thập phân. Trong ví dụ của chúng ta, dãy nhị phân thu được là: 11100010, giá trị thập phân tương ứng là: 226. Có được giá trị thập phân, chúng ta sẽ gán giá trị mới này vào pixel trung tâm. Lặp lại quá trình trên cho tất cả các pixel ta sẽ thu được một ma trận mới (có thể gọi là ma trận LBP) có cùng kích thước với ma trận trước đó, ma trận này mô tả các đặc trưng cục bộ của ảnh, sự thay đổi về màu sắc, cạnh hoặc kết cấu của ảnh. Như vậy là ta đã thu được ma trận LBP của 1 hình ảnh, ta cứ tiếp tục quá trình trên cho các ảnh khác trong dataset là sẽ nhận được các ma trận LBP và chúng ta sẽ lưu lại các ma trận LBP này lại để sử dụng cho việc phân loại sau này.

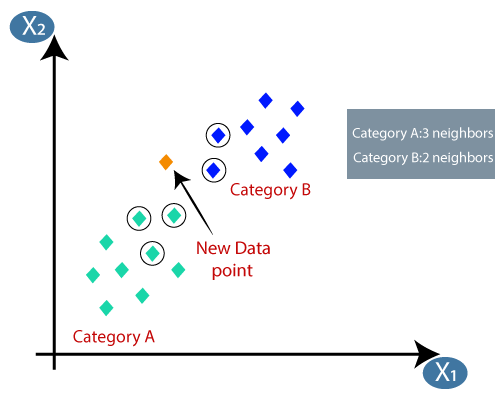
Lưu ý đối với các pixel nằm ở biên như tọa độ [0, 0], [0, 1] … thì các pixel lân cận ở ngoài biên chúng ta có thể gán giá trị cố định cho chúng ví dụ như 0 hoặc 255.

LBP có thể gặp 1 số khó khăn đối với ảnh quá sáng hoặc quá tối. Các pixel có giá trị cao gần bằng với giới hạn trên (255) hoặc thập gần bằng với giới hạn dưới (0) của mức xám, điều này dẫn đến các pixel có giá trị gần nhau khiến cho việc chuyển đổi giá trị thập phân gần giống nhau làm khó phân biệt được cấu trúc.

### **Thuật toán K – láng giềng gần nhất (K-Nearest Neighbors - KNN)**

KNN là một trong những thuật toán Supervised Learning đơn giản nhất (mà hiệu quả trong một vài trường hợp) trong ML. 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. KNN có thể áp dụng được vào cả hai loại của bài toán Supervised Learning là Phân loại (Classification) và Hồi quy (Regression).

Nguyên tắc hoạt động của thuật toán KNN là khi có một điểm dữ liệu mới cần được phân loại hoặc dự đoán, thuật toán KNN tìm K điểm gần nhất trong tập huấn luyện dựa trên khoảng cách Euclid (hoặc khoảng cách Manhattan) và sử dụng đa số lớp (trong trường hợp phân loại) hoặc trung bình (trong trường hợp dự đoán) của các nhãn của các điểm này để dự đoán nhãn của điểm mới.

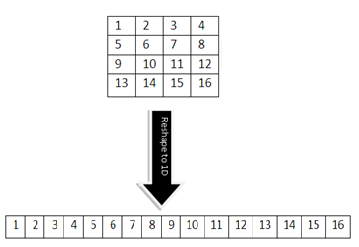


Hình 3. Minh họa KNN

Các bước thực hiện trong KNN:

* Chuẩn bị tập huấn luyện: Tập huấn luyện bao gồm các điểm dữ liệu đã được gán nhãn và các đặc trưng tương ứng với mỗi điểm. Điều này giúp thuật toán học cách phân loại các điểm dữ liệu dựa trên đặc trưng của chúng.
* Xác định giá trị K: Giá trị K là số lượng điểm gần nhất sẽ được sử dụng để đưa ra dự đoán cho điểm mới. Giá trị K có thể được chọn trước hoặc được tối ưu thông qua thử nghiệm.
* Tính toán khoảng cách: Sử dụng một phép đo khoảng cách (thường là khoảng cách Euclid hoặc khoảng cách Manhattan), tính toán khoảng cách giữa điểm mới và các điểm trong tập huấn luyện.
* Sắp xếp và chọn K điểm gần nhất: Sắp xếp các điểm trong tập huấn luyện theo khoảng cách tăng dần và chọn K điểm gần nhất.
* Đưa ra dự đoán: Dựa trên nhãn của các điểm gần nhất, đưa ra dự đoán cho điểm mới. Trong trường hợp phân loại, dự đoán sẽ là lớp xuất hiện nhiều nhất trong K điểm gần nhất. Trong trường hợp dự đoán, dự đoán sẽ là trung bình (hoặc trung vị) của giá trị của các điểm gần nhất.

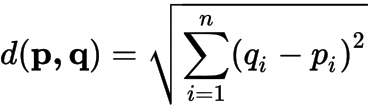
Thông thường, KNN yêu cầu tập huấn luyện là 1 danh sách các vector 1 chiều, mỗi vector 1 chiều tương ứng với mỗi mẫu dữ liệu đã được gán nhãn. Vì thế trước khi sử dụng KNN thì phải cần chuyển đổi các ma trận LBP thành vector 1 chiều (nói đơn giản chuyển mảng 2 chiều thành mảng 1 chiều) và vector 1 chiều này được gọi là vector đặc trưng.



Hình 3. Chuyển đổi ma trận thành vector 1 chiều

Khoảng cách để xác định các điểm láng giềng gần nhất với một điểm dữ liệu mới được xem là độ đo lường về sự tương đồng hoặc khác biệt giữa các điểm.

Có nhiều phương pháp để tính toán khoảng cách nhưng phổ biến nhất là Euclidean:



Hình 3. Công thức khoảng cách Euclidean

Trong đó:

* n là không gian n chiều.
* p và q là 2 điểm trong không gian n chiều.
* pi và qi là 2 vector bắt đầu từ điểm thứ i.

Cụ thể hơn đối với mối quan hệ giữa vector đặc trưng với khoảng cách Euclidean thì:

* n là số lượng điểm ảnh của 1 vector đặc trưng.
* p và q là 2 vectors đặc trưng.
* pi và qi là giá trị của pixel ở vị trí thứ i trong 2 vectors.

Ví dụ: một tập huấn luyện gồm 5 vector đặc trưng và nhãn tương ứng sau:

* Vector 1 (v1): [1, 3], Nhãn: A
* Vector 2 (v2): [2, 2], Nhãn: A
* Vector 3 (v3): [4, 6], Nhãn: B
* Vector 4 (v4): [3, 5], Nhãn: B
* Vector 5 (v5): [1, 2], Nhãn: B

Giả sử muốn dự đoán nhãn của vector mới (vn): [3, 3] sử dụng KNN với K = 3

Bước 1: Tính toán khoảng cách



Bước 2: Sắp xếp và chọn K vector gần nhất (sắp xếp tăng dần)

* Vector 2: 1.41
* Vector 5: 2.24
* Vector 1: 2
* Vector 4: 2
* Vector 3: 3.61

Chọn 3 vector gần nhất: vector 2, vector 5, vector 1 (Nếu trùng khoảng cách thì có thể chọn theo thứ tự vector).

Bước 3: Đưa ra dự đoán

Vì có 2 vectors có nhãn là A và 1 vector có nhãn là B.

* Xác suất của nhãn A là: 2/3 = 0.67 tương đương với 67%.
* Xác suất của nhãn B là: 100% - 67% = 33%.

Dựa vào xác suất trên, dự đoán vector mới [3, 3] là nhãn A với xác suất là 67%.

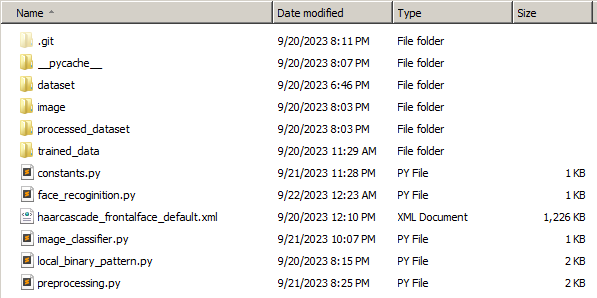
Một vấn đề quan trọng trong KNN đó là chọn K như thế nào ? Nếu chọn K không phù hợp có thể ảnh hưởng đến độ tin cậy cũng như hiệu suất của mô hình.

Có rất nhiều cách chọn K như:

* Phương pháp kiểm tra chéo (Cross-validation): Chia tập dữ liệu thành các fold, huấn luyện mô hình trên các fold con và đánh giá hiệu suất trên fold kiểm tra. Lặp lại quá trình này với các giá trị K khác nhau và chọn giá trị K tốt nhất dựa trên hiệu suất trung bình trên các fold.
* Phương pháp "sqrt(N)" hoặc "log(N)": Trong trường hợp không có thông tin cụ thể về dữ liệu, một phương pháp đơn giản là chọn giá trị K gần bằng căn bậc hai của N hoặc logarit tự nhiên của N. Trong đó N là số lượng mẫu huấn luyện.
* Hiệu suất trên tập kiểm tra: chọn giá trị K dựa trên hiệu suất trên tập kiểm tra. Huấn luyện mô hình với các giá trị K khác nhau và chọn giá trị K cho ra hiệu suất tốt nhất trên tập kiểm tra.

## **Triển khai thuật toán và quy trình nhận diện khuôn mặt**

### **Cấu trúc của dự án**



Hình 3. Cấu trúc của dự án

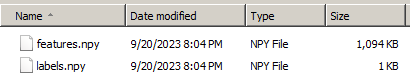
Hình trên là cấu trúc của dự án trong đó:

* Thư mục .git: một thư mục ẩn trong quản lý phiên bản của Git. Khi khởi tạo một kho lưu trữ Git thì thư mục.git sẽ được tạo ra, thư mục này chứa tất cả thông tin về lịch sử phiên bản, các nhánh (branches), các phiên bản (commits), các nhãn (tags)… Thư mục này không liên quan gì đến mô hình của chúng ta (có hay không không quan trọng).
* Thư mục \_\_pycache\_\_: được tạo ra tự động khi chạy một module Python. Nó chứa các file byte code được tạo ra từ các file nguồn Python (.py) để cải thiện hiệu suất khi thực thi chương trình. Tương tự như .git thư mục này cũng không liên quan đến mô hình của chúng ta.
* Thư mục dataset: đại diện cho tập dữ liệu chưa qua quá trình tiền xử lý, thư mục này chứa các thư mục con, mỗi thư mục con có tên là tên của một người nào đó, trong mỗi thư mục con sẽ chứa các hình ảnh của người đó.

|  |  |
| --- | --- |
|  |  |

Hình 3. Thư mục con của dataset và các ảnh trong thư mục con

* Thư mục processed\_dataset: đại diện cho tập dữ liệu đã qua quá trình tiền xử lý, cấu trúc của thư mục này tương tự như dataset.
* Thư mục image: chứa các hình ảnh cần được nhận dự đoán nhãn.
* Thư mục trained\_data: chứa 2 tập tin features.npy và labels.npy. Trong đó, features.npy là tập tin chứa các vector đặc trưng sau khi trích xuất đã được mã hóa dưới dạng nhị phân. Còn tập tin labels.npy chứa các nhãn tương ứng với các vector đặc trưng trên và các nhãn này cũng đã được mã hóa dưới dạng nhị phân.



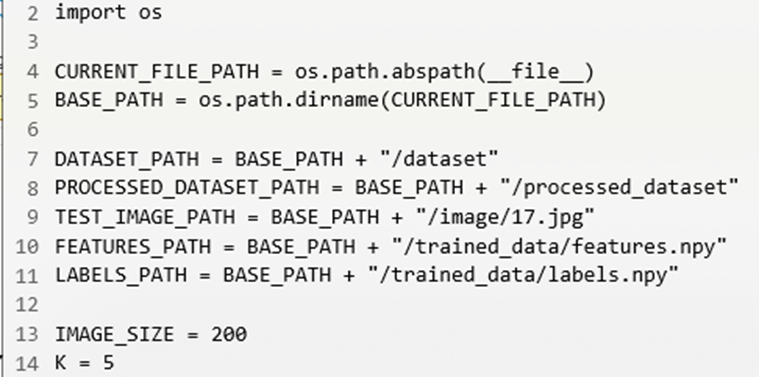
Hình 3. Tệp lưu trữ các vector đặc trưng và nhãn

* Tệp haarcascade\_frontalface\_default.xml: một mô hình phát hiện khuôn mặt được cung cấp bởi OpenCV, mô hình nãy đã được huấn luyện sẵn với hàng nghìn khuôn mặt. Chính ví thế, mô hình này rất mạnh mẽ trong việc phát hiện ra được khuôn mặt giúp cho chúng ta thực hiện quá trình tiền xử lý 1 cách dễ dàng.
* Tệp local\_binary\_pattern.py: chứa mã triển khai thuật toán LBP.
* Tệp constants.py: chứa các hằng số và đường dẫn của toàn bộ project.
* Tệp preprocessing.py: chứa mã triển khai quá trình tiền xử lý.
* Tệp image\_classifier.py: chứa mã triển khai quá trình trích xuất đặc trưng hình ảnh và gán nhãn.
* Tệp face\_recoginition.py: chứa mã triển khai nhận dạng khuôn mặt dựa trên ảnh đầu vào với KNN.

### **Triển khai mô hình với Python**

#### **Định nghĩa hằng số và đường dẫn đến các tài nguyên**

Chúng ta không nên, viết các đường dẫn hay các giá trị trực tiếp vào trong tập tin chính, điều đó làm cho việc bảo trì và mở rộng sau này cực kỳ khó khăn. Ví dụ như cấu trúc thư mục bị đổi thì phải tìm trong cả trăm dòng mã để sửa lại đường dẫn…Một cách tiếp cận tốt hơn là lưu các đường dẫn đến tài nguyên của hệ thống hoặc các hằng số xuất hiện nhiều lần trong toàn bộ chương trình thành 1 tệp riêng.



Hình 3. Mã nguồn contants.py

Hình trên là mã nguồn của constants.py. Trong đó: CURRENT\_FILE PATH là đường dẫn hiện tại của tập tin (bao gồm cả tên tập tin constants.py), khi có được đường dẫn hiện tại ta dùng hàm dirname từ module os để lấy ra đường dẫn thư mục hiện tại (tương tự như CURRENT\_FILE PATH nhưng không bao gồm tên tập tin). Khi có được BASE\_PATH ta có thể dễ dàng định nghĩa các đường dẫn đến các tài nguyên khác như: DATASET\_PATH (đường dẫn đến dataset), FEATURES\_PATH (đường dẫn đến các vector đặc trưng) … TEST\_IMAGE\_PATH (đường dẫn đến hình ảnh cần nhận dạng). Ngoài ra, còn có 2 hằng số là IMAGE\_SIZE và K, trong đó IMAGE\_SIZE là kích thước chung cho tất cả các hình ảnh (do các hình ảnh có số lượng điểm khác nhau, dẫn đến các vector đặc trưng cũng khác nhau về số lượng phần tử từ đó không thể tính khoảng cách Euclidean giữa các vector đặc trưng vì thế cần phải có 1 chuẩn chung cho số lượng điểm ảnh của các ảnh), K là giá trị K trong thuật toán KNN.

#### **Triển khai thuật toán LBP**

Đầu tiên, dựa vào thuật toán LBP ta có thể chia thành các bước như sau:

* Cần phải tìm giá trị mới cho tất cả các pixel trong ảnh.
* Để có thể tìm giá trị mới cho tất cả pixel, ta cần phải xét từng pixel trên ảnh. Với pixel đang xét, ta xem nó là pixel trung tâm và tiến hành phân ngưỡng 8 pixel lân cận pixel trung tâm.
* Để có thể phân ngưỡng, ta cần phải đảm bảo được điều kiện: ở các pixel lân cận, pixel nào có giá trị lớn hơn hoặc bằng giá trị của pixel trung tâm thì ta phân ngưỡng về 1, ngược lại phân ngưỡng về 0. Đối với pixel lân cận nằm ở ngoài biên của ảnh, ta có thể mặc định phân ngưỡng nó về 0.

Từ các bước trên ta có thể dễ dàng triển khai như sau:

A white screen with black text

Description automatically generated

Hình 3. Một phần mã nguồn local\_binary\_pattern.py

Trên đây là hàm get\_pixel() đại diện cho việc kiểm tra giá trị của pixel, xem xét pixel dựa trên điều kiện và trả về giá trị phân ngưỡng. Hàm này nhận 4 tham số:

* img: hình ảnh đang xét (ma trận pixel hiện tại).
* center: giá trị của pixel trung tâm.
* x, y: tương ứng với tọa độ x, y của pixel đang đóng vai trò là pixel lân cận.

A screenshot of a computer program

Description automatically generated

Hình 3. Một phần mã nguồn local\_binary\_pattern.py

Trên đây là hàm lbp\_calculated\_pixel(img, x, y), hàm này đại diện cho việc tính toán giá trị mới cho pixel trung tâm. Hàm này gồm 3 tham số: img (hình ảnh hiện tại), x (tọa độ x của pixel trung tâm), y (tạo độ y của pixel trung tâm). Hàm này tiến hành lấy ra các giá trị đã được phân ngưỡng từ 8 pixel lân cận bằng cách gọi hàm get\_pixel().Vì pixel trung tâm đang có tọa độ (x, y) vì thế 8 pixel lân cận sẽ có tọa độ là:

* Góc trên trái: (x – 1, y – 1)
* Ở trên: (x – 1, y)
* Góc trên phải: (x – 1, y + 1)
* Bên phải: (x, y + 1)
* Góc phải dưới: (x + 1, y + 1)
* Bên dưới: (x + 1, y)
* Góc trái dưới: (x + 1, y – 1)
* Bên trái: (x, y – 1)

Dựa trên cá tọa độ trên của 8 pixel lân cận, ta có thể thấy được là hàm này đang thu thập các bit theo chiều kim đồng hồ và gán các bit đó vào mảng val\_ar[]. Sau đó, là dùng thuật toán chuyển đổi 8 bit nhị phân sang thập phân và trả về giá trị thập phân.

A screenshot of a computer code

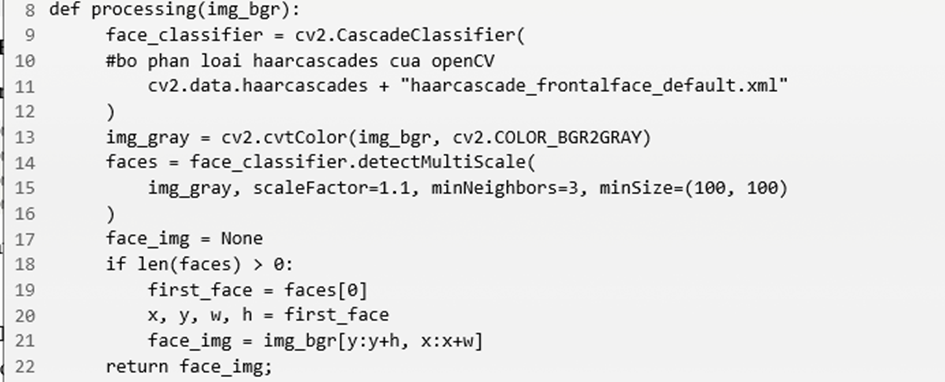
Description automatically generated

Hình 3. Một phần mã nguồn local\_binary\_pattern.py

Hàm LBP đại diện cho việc thực hiện áp dụng LBP cho tất cả các pixel của ma trận và trả về ma trận LBP. Hàm này nhận vào 2 tham số là img\_gray (hình ảnh xám), image\_shape (kích thước của img\_gray). Hàm này sẽ đi qua từng điểm ảnh và chỉ định vị trí đang đi qua là tọa độ của pixel trung tâm và đưa tọa độ đó cho hàm lbp\_calculated\_pixel() để lấy giá trị thâp phân mới. Sau khi có, giá trị thập phân mới rồi thì ta chỉ cần gán giá trị mới đó ngược lại vào pixel trung tâm.

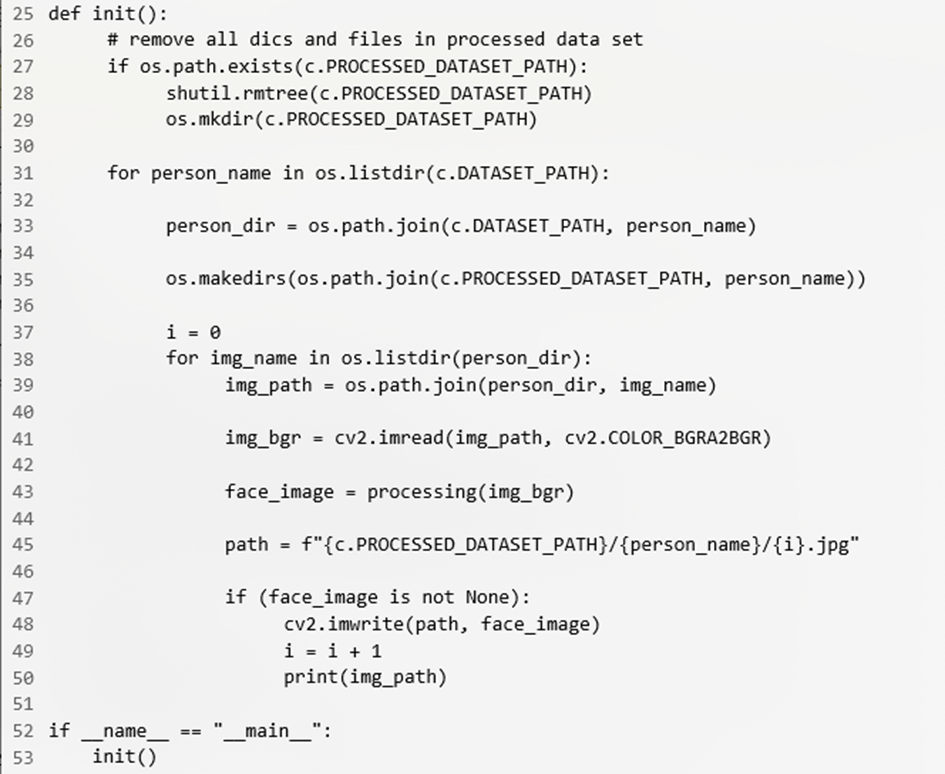
#### **Tiến trình tiền xử lý**

Tiến trình này bao gồm việc sử dụng mô hình Harcascade của OpenCV để phát hiện khuôn mặt (phát hiện khuôn mặt là tìm được khuôn mặt từ 1 ảnh đầu vào, trong khi nhận diện khuôn mặt là dự đoán được khuôn mặt đó thuộc về ai) và lấy ra khuôn mặt đó để làm dữ liệu huấn luyện.



Hình 3. Một phần mã nguồn tệp preprocessing.py

Hình trên là phần đầu của tệp preprocessing.py. Ở đây chúng ta sẽ tạo 1 hàm processing nhận 1 tham số là 1ảnh màu, hàm này có chức năng là chuyển đổi ảnh màu thành ảnh xám. Sau đó, mở mô hình Harcascade của OpenCV để tiến hành phát hiện khuôn mặt của ảnh xám, khi phát hiện được khuôn mặt thì sẽ tiến hành cắt khuôn mặt đã được phát hiện và trả về khuôn mặt đó.



Hình 3. Một phần mã nguồn tệp preprocessing.py

Hình trên là phần tiếp theo của tệp preprocessing.py. Hàm init sẽ tiến hành lặp qua các ảnh trong dataset và gọi hàm process() để thực hiện việc lấy ra khuôn mặt đã được cắt và lưu lại vào trong processed\_dataset. Do có định nghĩa \_\_name\_\_ == “\_\_main\_\_” vì thế hàm init() chỉ được gọi trong quá trình chạy trực tiếp tập tin preprocessing.py. Lý do cho việc chỉ gọi init() khi chạy trực tiếp preprocessing.py là do thao tác xử lý hình ảnh bằng cách gọi hàm process() không chỉ dành riêng để xử lý các hình ảnh trong dataset mà hàm process() còn được những tiến trình khác gọi để thực hiện xử lý cho ảnh cần nhận dạng nữa. Nếu không \_\_name\_\_ == “\_\_main\_\_” thì các tập tin khác import tập tin này sẽ tự động thực hiện process() cho các dataset 1 lần nữa.

Sau khi chạy preprocessing.py ta sẽ nhận được 1 dataset mới đã được xử lý gọi là processed\_dataset, dataset mới này có cấu trúc tương tự với dataset cũ, chỉ các biệt là ở các hình ảnh trong dataset mới chỉ chứa các khuôn mặt đã được phát hiện.

#### **Tiến trình trích xuất đặc trưng và gán nhãn**

Tiến trình này là bao gồm việc sử dụng LBP để trích xuất đặc trưng để tạo ra vector đặc trưng của tất cả hình ảnh trong dataset và gán nhãn cho các vector đặc trưng đó.

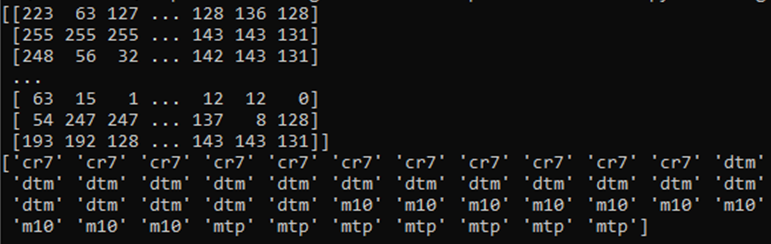
A screenshot of a computer code

Description automatically generated

Hình 3. Mã nguồn tệp image\_classifier.py

Hình trên là đoạn mã của tệp image\_classifier.py. Tệp này là triển khai cho quá trình trích xuất đặt trưng và gán nhãn. Trong đó, features[] là mảng chứa các vector đặc trưng của hình ảnh, còn labels[] là mảng chứa các nhãn tương ứng với vector đặc trưng đó. Do chúng ta đã triển khai thuật toán LBP tron tệp local\_binary\_pattern.py, và trong tệp này có chứa hàm LBP() trả về ma trận LBP từ ảnh đầu vào. Vì thế, chúng ta chỉ cần import tập tin local\_binary\_pattern.py vào tệp hiện tại để sử dụng.

Quá trình trích xuất đặc trưng này đơn giản là lặp qua các hình ảnh trong dataset đã được xử lý (processed\_dataset) lấy ra từng tấm ảnh trong thư mục con và truyền vào hàm LBP() là ta đã nhận được 1 ma trận LBP từ tấm ảnh đầu vào. Sau đó, ta tiến hành chuyển đổi ma trận LBP này thành vector 1 chiều bằng cách gọi hàm flatten(). Khi có được vector 1 chiều ta tiến hành lưu vector đặc trưng này vào features[] và gán nhãn tương ứng cho vector đó. Việc gán nhãn được thực hiện bằng cách khi lặp qua thư mục con trong processed\_dataset, thì các tệp hình ảnh trong cùng 1 thư mục con được xem như là các vector đặc trưng thì tên thư mục con được xem là nhãn của các vector đặc trưng và tên này chúng ta sẽ lưu vào labels[]. Sau khi thực hiện quá trình trên với tất cả hình ảnh trong processed\_dataset, chúng ta sẽ thu được 1 các vector đặc trưng nằm trong features[] và các nhãn tương ứng nằm trong labels[], sau đó là lưu 2 mảng này vào thư mục trained\_data. Và 2 tệp này sẽ được dùng để so sánh và phân loại đặc trưng sau này. Lưu ý, khi có 1 dữ liệu mới ở dataset, ta phải thực hiện lại quá trình tiền xử lý, sau đó là quá trình rút trích đặc trưng và gán nhãn để đảm bảo mô hình không bị thiếu sót dữ liệu.

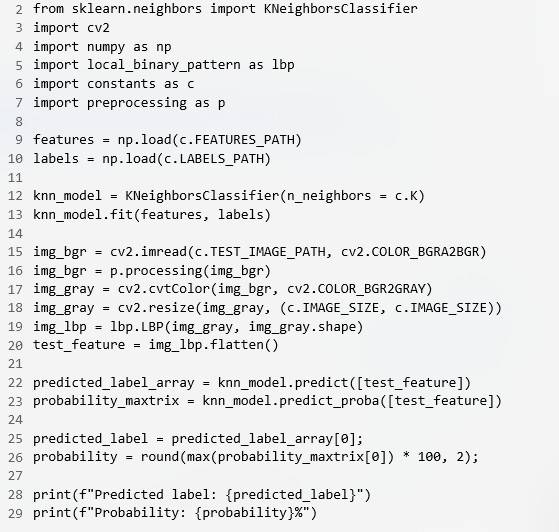


Hình 3. Các vectors đặc trưng và nhãn tương ứng

Hình ảnh trên cho thấy mối quan hệ trực quan giữa features[] và labels[], với 3 vector đặc trưng đầu tiên của mảng bên trên sẽ tương ứng với 3 nhãn đầu tiên của mảng bên dưới. Kỹ thuật này có thể xem là 1 kỹ thuật đánh dấu, dù 2 đối tượng đang nằm ở 2 mảng khác nhau, nhưng vẫn có mối liên hệ giữa vị trí xuất hiện của chúng. Ví dụ để lấy được vector đầu tiên và nhãn tương ứng của nó ta có thể truy xuất theo kiểu features[0] (vector đặc trưng) và labels[0]( nhãn của vector đặc trưng).

#### **Tiến trình nhận diện khuôn mặt**

Tiến trình này bao gồm các công việc như là đọc hình ảnh cần được nhận dạng, trích xuất đặc trưng của hình ảnh cần nhận dạng, triển khai KNN để tìm sự tương đồng giữa vector đặc trưng đầu vào và các vector đặc trưng trong dataset, cuối cùng là đưa ra dự đoán.



Hình 3. Mã nguồn tệp face\_recognition.py

Trên đây là nội dung của tệp face\_recognition.py, tập tin này đại diện cho quá trình nhận diện khuôn mặt. Đầu tiên là lấy các vector đặc trưng và nhãn được huấn luyện và gán trước đó bằng cách gọi hàm load của module numpy và truyền vào đường dẫn. Tiếp theo là đọc hình ảnh đầu vào, hình ảnh đầu vào khi được đọc là 1 ảnh màu, ta cần chuyển đổi sang ảnh xám và rút trích đặc trưng của ảnh đầu vào thành 1 vector đặc trưng bằng cách gọi hàm LBP() và flatten(). Sau đó, chúng ta sẽ sử dụng KNN đã được xây dựng sẵn, gọi hàm fit() truyền vào các vector đặc trưng và nhãn, sau đó gọi hàm predicted() truyền vào vector đặc trưng cửa ảnh đầu vào để tiến hành dự đoán nhãn, nếu muốn biết thêm xác suất dự đoán có thể gọi hàm predicted\_proba(). Hàm predicted\_proba sẽ trả về xác suất của các nhãn, chúng ta có thể lấy xác suất của nhãn đã được dự đoán bằng cách kết hợp hàm max() với predict\_proba().

# CHƯƠNG 4: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN



## **Kết quả đạt được**

Đầu tiên là chạy tệp preprocessing.py để xử lý các hình ảnh đầu vào, với mỗi hình ảnh đã được xử lý thì nó sẽ in ra đường dẫn đến hình ảnh đó.

A screen shot of a computer

Description automatically generated

Hình 4. Giao diện sau khi chạy preprocessing.py

Sau khi chạy xong tệp preprocessing.py thư mục processed\_data của chúng ta có các hình ảnh đã được xử lý

A collage of a person's face

Description automatically generated

Hình 4. Hình ảnh trong processed\_dataset

Tiếp theo là tiến hành trích xuất đặc trưng và gán nhãn bằng cách chạy tệp image\_classifier.py. Khi chạy xong tệp này, nó sẽ hiển thị ra danh sách các vector đặc trưng và các nhãn tương ứng với vector đó.

A screen shot of a computer

Description automatically generated

Hình 4. Giao diện sau khi chạy image\_classifier,py

Cuối cùng là tiến hành nhận diện khuôn mặt từ ảnh đầu vào bằng cách chạy tệp face\_recognition.py. Sau khi chạy xong, mô hình sẽ đưa ra dự đoán nhãn và xác suất đúng của dự đoán.

**D:\Desktop\328db0594fba9be4c2ab.jpg**

Hình 4. Giao diện khi chạy face\_recognition.py

## **Những thu hoạch về mặt chuyên môn**

Trong quá trình nghiên cứu về đề tài nhận diện khuôn mặt, chúng em đã tiếp cận được các kiến thức cơ bản về trí tuệ nhân tạo, học máy và các thuật toán nhận diện hình ảnh. Đầu tiên là chúng em hiểu được cách thức hoạt động của phương pháp LBP trong việc miêu tả đặc trưng khuôn mặt. LBP sẽ đánh giá các điểm ảnh xung quanh một điểm ảnh trung tâm để biểu diễn được các chi tiết nhỏ như nét mặt, nốt ruồi... Giúp phân biệt các khuôn mặt tốt hơn.

Sau khi lấy được vector đặc trưng của mỗi khuôn mặt bằng LBP, chúng em hiểu KNN sẽ đóng vai trò quan trọng trong bước nhận diện. KNN sẽ tính toán khoảng cách giữa vector của khuôn mặt cần nhận diện với những vector trong bộ dữ liệu huấn luyện và chọn nhãn của khuôn mặt gần nhất làm kết quả nhận diện.

Nhờ nghiên cứu đề tài này, em hiểu rõ hơn về quá trình nhận diện khuôn mặt tổng quan cũng như hai phương pháp LBP và KNN. Chúng em rút ra được nhiều bài học quý giá về khả năng ứng dụng trí tuệ nhân tạo trong đời sống. Đây sẽ là nền tảng giúp chúng em phát triển hơn nữa trong lĩnh vực này.

## **Những khó khăn, thách thức khi phát triển mô hình**

Khi chúng em nghiên cứu và phát triển mô hình nhận diện khuôn mặt sử dụng LBP và KNN, chúng em đã gặp một số khó khăn và thách thức nhất định.

* Hiểu và triển khai phương pháp LBP: Ban đầu, việc hiểu và triển khai phương pháp LBP là một thách thức đối với chúng em. LBP là một phương pháp trích xuất đặc trưng từ mẫu ảnh bằng cách so sánh các điểm ảnh trong vùng lân cận của mỗi điểm ảnh. Việc hiểu cách hoạt động của LBP và cách tính toán histogram của các giá trị LBP đã đòi hỏi chúng em phải đọc nhiều tài liệu và nghiên cứu sâu hơn về thuật toán này.
* Tìm hiểu và triển khai thuật toán KNN: Thuật toán KNN là một thuật toán đơn giản và dễ hiểu, nhưng chúng em đã gặp khó khăn khi triển khai nó trong mô hình nhận diện khuôn mặt. Việc tính toán khoảng cách giữa các điểm dữ liệu và tìm kiếm các láng giềng gần nhất có thể yêu cầu nhiều thời gian và tài nguyên tính toán. Chúng em đã phải tìm hiểu cách tối ưu hóa thuật toán và làm việc với các thư viện phù hợp để đảm bảo hiệu suất tính toán tốt.
* Số lượng mẫu huấn luyện và tính tổng quát hóa: Một khó khăn quan trọng khác là việc thu thập và gán nhãn cho một tập dữ liệu huấn luyện đủ lớn và đa dạng về khuôn mặt. Điều này đòi hỏi nỗ lực và thời gian đáng kể để thu thập và chuẩn bị dữ liệu. Ngoài ra, mô hình cũng phải đảm bảo tính tổng quát hóa trên các tập dữ liệu mới mà không nằm trong tập huấn luyện ban đầu. Điều này đòi hỏi chúng em phải kiểm tra và đánh giá mô hình trên các tập dữ liệu thực tế để đảm bảo hiệu suất ổn định và chính xác.

## **Hướng phát triển**

Sau khi đối mặt với các khó khăn và thách thức khi phát triển mô hình nhận diện khuôn mặt sử dụng phương pháp LBP và thuật toán KNN, chúng em đã thu được những kết quả khá tích cực. Tuy nhiên, chúng em nhận thấy còn nhiều hướng phát triển tiềm năng để cải thiện mô hình và tăng cường hiệu suất nhận diện. Dưới đây là một số hướng phát triển mà chúng em đang xem xét:

* Tăng cường độ chính xác của phương pháp LBP: Mặc dù phương pháp LBP đã cho kết quả tốt, chúng em đang nghiên cứu cách cải tiến phương pháp này để tăng cường độ chính xác trong việc trích xuất đặc trưng khuôn mặt. Điều này có thể đạt được bằng cách sử dụng các biến thể của LBP như Improved LBP hay Extended LBP…
* Áp dụng kỹ thuật học máy tiên tiến: Một hướng phát triển quan trọng là áp dụng các kỹ thuật học máy tiên tiến vào mô hình nhận diện khuôn mặt. Thay vì sử dụng thuật toán KNN, chúng em có thể xem xét sử dụng các thuật toán phân lớp khác như Support Vector Machine (SVM), Random Forest hoặc Neural Networks để tăng cường khả năng phân loại và đạt được hiệu suất tốt hơn.
* Xử lý biến đổi ánh sáng và biến đổi khác: Để cải thiện khả năng xử lý biến đổi ánh sáng và biến đổi khác trên khuôn mặt, chúng em đang nghiên cứu cách sử dụng các phương pháp tiền xử lý ảnh như đồng nhất hóa histogram, cân bằng sáng tạo (adaptive histogram equalization) hay xử lý bằng không gian màu (color space transformation). Điều này giúp tăng tính ổn định và độ chính xác của mô hình khi đối mặt với các biến đổi không mong muốn trên khuôn mặt.
* Mở rộng tập dữ liệu huấn luyện: Để đảm bảo tính tổng quát hóa của mô hình, chúng em đang lập kế hoạch mở rộng tập dữ liệu huấn luyện bằng cách thu thập thêm các mẫu khuôn mặt từ nhiều nguồn khác nhau. Điều này giúp mô hình được huấn luyện trên nhiều đặc trưng và biến thể của khuôn mặt, từ đó tăng khả năng nhận diện và giảm sai số trong quá trình phân loại.
* Tối ưu hóa và tăng tốc độ tính toán: Để đảm bảo hiệu suất tính toán tốt và thời gian phản hồi nhanh, chúng em đang nghiên cứu cách tối ưu hóa thuật toán và sử dụng các công nghệ tính toán song song như CUDA hoặc OpenCL để tận dụng sức mạnh tính toán của GPU.

# Tài liệu tham khảo

[1] Nhan Thanh Nhã. Học máy. Khoa TT-TT. Đại học Kiên Giang, 2020 - 2021

[2] https://www.analyticsvidhya.com/blog/2021/07/understanding-face-recognition-using-lbph-algorithm/

[3] https://www.geeksforgeeks.org/k-nearest-neighbours/