TRƯỜNG ĐẠI HỌC HỌC VĂN LANG

**KHOA CÔNG NGHỆ THÔNG TIN**

**A red and white logo

AI-generated content may be incorrect.**

**BÁO CÁO ĐỒ ÁN MÔN HỌC**

**NHẬP MÔN XỬ LÍ ẢNH SỐ**

NGÀNH: CÔNG NGHỆ THÔNG TIN

***Tên Đồ án:***

**Nhận diện khuôn mặt bằng Haar Cascade**

**và KNN**

**NHÓM 03**

**HÀ ĐẠI VĨ – 2374802010551**

**PHẠM THANH TÙNG – 2374802010853**

**LÊ HỮU KHANG – 2374802011980**

**ĐOÀN NHẬT KHÔI – 2374802010242**

**MỤC LỤC**

[**CHƯƠNG I: CƠ SỞ LÝ THUYẾT** 3](#_Toc205142100)

[**1. Mục tiêu nghiên cứu** 3](#_Toc205142101)

[**2. Phạm vi và giới hạn đề tài** 3](#_Toc205142102)

[**3. Giới thiệu về HaarCascade** 4](#_Toc205142103)

[**3.1. Mở đầu** 4](#_Toc205142104)

[**3.2. Khái niệm về Haar Cascade** 5](#_Toc205142105)

[**3.3. Cơ chế hoạt động của Haar Cascade** 5](#_Toc205142105)

[**3.4. Ưu và nhược điểm của Haar Cascade** 5](#_Toc205142105)

[**4. Ứng dụng và ví dụ về HaarCascade** 6](#_Toc205142106)

[**5. Giới thiệu về thuật toán KNN** 7](#_Toc205142107)

[**6.Ứng dụng và ví dụ về thuật toán KNN** 7](#_Toc205142108)

[**CHƯƠNG II: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG** 8](#_Toc205142109)

[**1.** **Lên ý tưởng và chuẩn bị mô hình** 8](#_Toc205142110)

[**2.** **Thiết kế các bước xử lý ảnh cụ thể** 8](#_Toc205142111)

[**2.1. Thu thập dữ liệu khuôn mặt** 8](#_Toc205142112)

[**2.2. Mô hình nhận diện khuôn mặt** 9](#_Toc205142113)

[**2.3. Nhận diện khuôn mặt trực tiếp** 9](#_Toc205142114)

[**3.** **Mô tả dữ liệu đầu vào/đầu ra** 9](#_Toc205142115)

[**CHƯƠNG III: CÀI ĐẶT VÀ THỰC NGHIỆM** 11](#_Toc205142116)

[**1.** **Môi trường và công cụ phát triển** 11](#_Toc205142117)

[**2.** **Cài đặt các thành phần** 11](#_Toc205142118)

[**3.** **Kết quả thực nghiệm** 12](#_Toc205142119)

[**4.** **Đánh giá kết quả** 12](#_Toc205142120)

[**CHƯƠNG III: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN** 14](#_Toc205142121)

[**1.** **Tổng kết kết quả** 14](#_Toc205142122)

[**2.** **Hạn chế của mô hình** 14](#_Toc205142123)

[**3.** **Định hướng phát triển** 14](#_Toc205142124)

[**4.** **Tài liệu tham khảo** 14](#_Toc205142125)

**BẢNG ĐÁNH GIÁ**

|  |  |  |
| --- | --- | --- |
| **HỌ VÀ TÊN** | **NHIỆM VỤ** | **ĐÁNH GIÁ** |
| **Phạm Thanh Tùng** | **Nội dung, báo cáo,code,ppt** | **100%** |
| **Đoàn Nhật Khôi** | **Nội dung, báo cáo, code, ppt** | **100%** |
| **Hà Đại Vĩ** | **Nội dung, báo cáo, code, check ppt** | **100%** |
| **Lê Hữu Khang** | **Nội dung, báo cáo, code** | **100%** |

**LỜI CẢM ƠN**

Kính gửi thầy

Trước hết, em xin gửi lời cảm ơn chân thành đến thầy Đỗ Hữu Quân, người đã tận tình hướng dẫn và hỗ trợ em trong suốt quá trình thực hiện đề tài "Nhận diện khuôn mặt bằng Haar Cascade và KNN". Những kiến thức và kinh nghiệm quý báu mà thầy chia sẻ đã giúp em rất nhiều trong việc nghiên cứu và hoàn thành bài báo cáo này.

Em cũng xin cảm ơn các bạn trong lớp đã cùng em thảo luận và chia sẻ ý tưởng, góp phần làm cho quá trình học tập trở nên thú vị và bổ ích hơn. Hy vọng rằng những kiến thức thu được từ đề tài này sẽ là nền tảng vững chắc cho sự nghiệp học tập và nghiên cứu của em trong tương lai.

Xin chân thành cảm ơn!

# **CHƯƠNG I: CƠ SỞ LÝ THUYẾT**

## **1. Mục tiêu nghiên cứu**

* Hiểu được cơ sở toán học và trực quan của **đặc trưng Haar (Haar-like features)** và cách chúng được sử dụng để phát hiện các mẫu hình ảnh.
* Biết cách tính toán nhanh các đặc trưng bằng **Integral Image** – một công cụ cốt lõi giúp tăng tốc xử lý.
* Biết cách sử dụng mô hình Haar Cascade sẵn có trong thư viện **OpenCV**, áp dụng vào các bài toán như:
  + Nhận diện khuôn mặt
  + Phát hiện mắt, nụ cười
  + Phát hiện người, vật thể,....

## **2. Phạm vi và giới hạn đề tài**

* Đề tài tập trung vào việc tìm hiểu, xây dựng và cài đặt để nhận diện hình ảnh bằng 1 cách hiện đại và tối ưu nhất.
* Phạm vi của đề tài bao gồm:
  + **Tìm hiểu lý thuyết** về Haar-like features, integral image, Ada-boost và cascade of classifiers – các thành phần cốt lõi tạo nên thuật toán Haar Cascade.
  + **Sử dụng thư viện OpenCV** trong Python để thực hiện quá trình phát hiện khuôn mặt bằng cách áp dụng mô hình Haar Cascade được huấn luyện sẵn.
  + **Triển khai chương trình phát hiện khuôn mặt trong ảnh tĩnh và video trực tiếp**

face\_data = []

labels = []

names = {}

class\_id = 0

for fx in os.listdir(dataset\_path):

    if fx.endswith('.npy'):

        names[class\_id] = fx[:-4]

        data\_item = np.load(os.path.join(dataset\_path, fx))

        face\_data.append(data\_item)

        target = class\_id \* np.ones((data\_item.shape[0],))

        labels.append(target)

        class\_id += 1

face\_dataset = np.concatenate(face\_data, axis=0)

face\_labels = np.concatenate(labels, axis=0).reshape((-1, 1))

print("[INFO] Loaded training data:")

print("  Faces:", face\_dataset.shape)

print("  Labels:", face\_labels.shape)

Giới hạn của đề tài là vùng cắt không vượt khung hình

x1 = max(x - offset, 0)

y1 = max(y - offset, 0)

x2 = min(x + w + offset, w\_frame)

y2 = min(y + h + offset, h\_frame)

face\_offset = frame[y1:y2, x1:x2]

if face\_offset.size == 0:

    continue

face\_section = cv2.resize(face\_offset, (100, 100)).flatten().reshape(1, -1)

## **3. Giới thiệu về HaarCascade**

### **3.1. Mở đầu**

Trong lĩnh vực thị giác máy tính (Computer Vision), một trong những nhiệm vụ cơ bản và quan trọng nhất là phát hiện đối tượng (Object Detection) – quá trình xác định vị trí và kích thước của các đối tượng cụ thể trong ảnh hoặc video. Một trong những phương pháp kinh điển và phổ biến nhất để phát hiện đối tượng, đặc biệt là phát hiện khuôn mặt, chính là Haar Cascade Classifier, thường được gọi tắt là Haar Cascade. Phương pháp này đã được giới thiệu bởi Paul Viola và Michael Jones vào năm 2001 trong công trình nghiên cứu có tên "Rapid Object Detection using a Boosted Cascade of Simple Features" và đã trở thành nền tảng cho nhiều ứng dụng trong đời sống như nhận diện khuôn mặt trong máy ảnh kỹ thuật số, hệ thống giám sát, ứng dụng mở khóa khuôn mặt, v.v.

### **3.2. Khái niệm về Haar Cascade**

* Haar Cascade là một bộ phân loại đối tượng (object classifier) được xây dựng dựa trên các đặc trưng đơn giản gọi là Haar-like features, được huấn luyện bằng phương pháp học máy để nhận biết sự xuất hiện của đối tượng mục tiêu trong một hình ảnh. Tên gọi "Haar" xuất phát từ cơ sở toán học Haar wavelet – một loại biến đổi sóng dùng trong phân tích tín hiệu. Tuy nhiên, trong ngữ cảnh của Haar Cascade, các đặc trưng Haar chỉ là những mẫu hình chữ nhật đơn giản được **sử dụng để đo sự khác biệt về cường độ pixel giữa các vùng khác nhau của ảnh.**

### **3.3. Cơ chế hoạt động của Haar Cascade**

* Bước 1: Trích xuất đặc trưng Haar (Haar Features): Thay vì dùng toàn bộ pixel, thuật toán sử dụng các mẫu hình đơn giản (như hình chữ nhật đen-trắng) để mô tả đặc điểm của một vùng trong ảnh.
* Bước 2: Huấn luyện bằng AdaBoost: Thuật toán AdaBoost sẽ chọn ra những đặc trưng tốt nhất và kết hợp chúng thành một bộ phân loại mạnh (strong classifier).
* Bước 3: Bộ phân loại phân tầng (Cascade Classifier): Để tăng tốc, các phân loại được sắp xếp thành nhiều tầng. Tầng 1: kiểm tra rất nhanh để loại bỏ vùng không chứa khuôn mặt. Tầng 2, 3,...: chỉ kiểm tra tiếp nếu vượt qua tầng trước.
* Bước 4: Duyệt vùng ảnh (Sliding Window): Thuật toán sẽ quét toàn bộ ảnh với các cửa sổ kích thước khác nhau, thử phát hiện đối tượng ở mọi vị trí, mọi tỉ lệ.

### **3.4. Ưu và nhược điểm của Haar Cascade**

**Ưu điểm:**

* Tốc độ nhanh: Thiết kế theo dạng "cascade" giúp loại bỏ nhanh các vùng không chứa đối tượng, phù hợp với ứng dụng real-time như webcam hoặc camera giám sát.
* Dễ sử dụng với OpenCV: OpenCV cung cấp sẵn nhiều mô hình Haar cascade được huấn luyện cho các đối tượng phổ biến như: haarcascade\_frontalface\_alt.xml (khuôn mặt), haarcascade\_eye.xml (mắt), Cài đặt đơn giản, chỉ vài dòng code, v.v.
* Không cần GPU: Có thể chạy tốt trên máy tính yếu, không cần card đồ họa và phù hợp với hệ thống nhúng hoặc thiết bị di động đời cũ.
* Không yêu cầu dữ liệu lớn để huấn luyện lại: Đã có sẵn mô hình huấn luyện, người dùng không cần huấn luyện từ đầu.

**Nhược điểm:**

* Độ chính xác không cao: Dễ phát hiện sai (false positives) hoặc bỏ sót đối tượng (false negatives). Hiệu quả giảm rõ rệt khi đối tượng bị nghiêng/góc nhìn khác, ánh sáng yếu hoặc nhiều nhiễu.
* Không học sâu (deep learning): Không thể tự động học các đặc trưng phức tạp từ dữ liệu như mô hình CNN. Phụ thuộc vào đặc trưng thủ công (Haar features).
* Vì phải quét toàn ảnh với nhiều kích thước cửa sổ, sẽ tốn tài nguyên hơn nếu ảnh lớn.
* Không linh hoạt với nhiều loại đối tượng: Chỉ hoạt động tốt với những đối tượng có đặc trưng hình học đơn giản, như khuôn mặt, mắt, biển số xe,... Không phù hợp để phát hiện những đối tượng phức tạp hoặc thay đổi hình dạng nhiều (ví dụ: động vật, tư thế người đa dạng).

## **4. Ứng dụng và ví dụ về HaarCascade**

**Nhận diện khuôn mặt (Face Detection)**

Thuật toán có thể phát hiện khuôn mặt người trong ảnh hoặc video với tốc độ nhanh, kể cả trong thời gian thực. Sau đây là những tình huống được sử dụng

* **Máy ảnh kỹ thuật số, điện thoại thông minh**: Tự động nhận diện khuôn mặt để lấy nét.
* **Hệ thống điểm danh tự động**: Nhận diện khuôn mặt sinh viên hoặc nhân viên để ghi nhận thời gian làm việc/học tập.

while True:

    ret, frame = cap.read()

    if not ret:

        continue

    gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

    faces = face\_cascade.detectMultiScale(gray, 1.3, 5)

    for (x, y, w, h) in faces:

        offset = 5

        h\_frame, w\_frame = frame.shape[:2]

## **5. Giới thiệu về thuật toán KNN**

Trong lĩnh vực **Học máy (Machine Learning)**, có rất nhiều thuật toán được thiết kế nhằm giải quyết các bài toán như phân loại (classification), hồi quy (regression), cụm hóa (clustering),... Trong số đó, thuật toán **K-Nearest Neighbors (KNN)** là một trong những thuật toán **đơn giản nhất nhưng lại cực kỳ mạnh mẽ và dễ hiểu**, thường được giới thiệu đầu tiên khi người học bắt đầu làm quen với học máy.

KNN không chỉ nổi bật bởi tính trực quan mà còn bởi khả năng áp dụng hiệu quả vào nhiều bài toán thực tế như **phân loại hình ảnh**, **dự đoán hành vi người dùng**, **gợi ý sản phẩm**, hay **nhận dạng chữ viết tay**, v.v.

Khái niệm về thuật toán KNN: **K-Nearest Neighbors (KNN)** là một **thuật toán phân loại và hồi quy phi tham số (non-parametric)**. Phi tham số có nghĩa là thuật toán **không giả định trước mô hình phân phối dữ liệu**, và không có quá trình huấn luyện rõ ràng như các mô hình khác. Thay vào đó, thuật toán **ghi nhớ toàn bộ tập dữ liệu huấn luyện**, và khi cần phân loại một điểm mới, nó **so sánh điểm đó với các điểm gần nhất trong tập huấn luyện**.

## **6.Ứng dụng và ví dụ về thuật toán KNN**

Thuật toán KNN là một trong những thuật toán học máy đơn giản nhưng hiệu quả.

- Ứng dụng về thuật toán KNN.

- Phân loại hình ảnh: Nhận diện khuôn mặt.

knn\_model = KNeighborsClassifier(n\_neighbors=5, metric='euclidean')

knn\_model.fit(face\_dataset\_pca, face\_labels.ravel())

- Thuật toán có thể dự đoán được hình ảnh khuôn mặt cũng như nhận diện.

pred\_label = knn\_model.predict(face\_pca)[0]

distances, \_ = knn\_model.kneighbors(face\_pca, n\_neighbors=1)

min\_dist = distances[0][0]

# **CHƯƠNG II: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG**

## **Lên ý tưởng và chuẩn bị mô hình**

**Ý tưởng chính:** Hệ thống nhận diện khuôn mặt cá nhân được chia làm hai phần:

* **Thu thập dữ liệu:** Ghi lại hình ảnh khuôn mặt người dùng qua webcam, lưu dưới dạng mảng numpy .npy.
* **Nhận diện khuôn mặt:** Tải dữ liệu đã lưu, xử lý và huấn luyện mô hình để nhận diện khuôn mặt trực tiếp.

**Chuẩn bị mô hình:**

* **Thuật toán phát hiện khuôn mặt:** Sử dụng Haar Cascade Classifier với file haarcascade\_frontalface\_alt.xml để phát hiện vùng khuôn mặt trong ảnh.
* **Tiền xử lý dữ liệu:**
  + Chuyển đổi ảnh màu sang ảnh xám (grayscale).
  + Chuẩn hóa kích thước vùng khuôn mặt về 100x100 pixels.
  + Biến đổi dữ liệu về dạng vector 1 chiều.
* **Giảm chiều dữ liệu:** Dùng **PCA (Principal Component Analysis)** để giảm số chiều đặc trưng.
* **Huấn luyện phân loại:** Dùng **K-Nearest Neighbors (KNN)** để gán nhãn khuôn mặt theo dữ liệu đã thu thập.

## **Thiết kế các bước xử lý ảnh cụ thể**

### **2.1. Thu thập dữ liệu khuôn mặt**

* **Bước 1:** Mở webcam (cv2.VideoCapture(0)).
* **Bước 2:** Đọc từng khung hình (cap.read()).
* **Bước 3:** Chuyển ảnh sang grayscale để giảm nhiễu.
* **Bước 4:** Phát hiện khuôn mặt (detectMultiScale).
* **Bước 5:** Chọn khuôn mặt lớn nhất (gần camera nhất).
* **Bước 6:** Cắt vùng ảnh chứa khuôn mặt, resize về kích thước chuẩn.
* **Bước 7:** Sau mỗi 10 khung hình, lưu 1 ảnh vào danh sách dữ liệu.
* **Bước 8:** Khi người dùng bấm q, lưu toàn bộ dữ liệu thành file .npy.

### **2.2. Mô hình nhận diện khuôn mặt**

* **Bước 1:** Đọc tất cả file dữ liệu .npy từ thư mục lưu trữ.
* **Bước 2:** Ghép dữ liệu và nhãn thành tập huấn luyện.
* **Bước 3:** Chuẩn hóa đặc trưng (StandardScaler).
* **Bước 4:** Giảm chiều dữ liệu bằng PCA để tăng tốc tính toán.
* **Bước 5:** Huấn luyện KNN classifier với tập dữ liệu.

### **2.3. Nhận diện khuôn mặt trực tiếp**

* **Bước 1:** Mở webcam và đọc khung hình.
* **Bước 2:** Phát hiện khuôn mặt.
* **Bước 3:** Cắt, resize, chuẩn hóa và biến đổi PCA.
* **Bước 4:** Dự đoán nhãn khuôn mặt và khoảng cách tới người hàng xóm gần nhất.
* **Bước 5:** So sánh khoảng cách với ngưỡng để xác định "Unknown".
* **Bước 6:** Hiển thị kết quả nhận diện lên khung hình (tên hoặc Unknown).

## **Mô tả dữ liệu đầu vào/đầu ra**

**Dữ liệu đầu vào:**

* **Tập dữ liệu huấn luyện:**
  + Các file .npy chứa mảng ảnh khuôn mặt (vector chiều dài 100×100×3).
  + Tên file .npy dùng làm nhãn nhận diện.
* **Hình ảnh webcam thời gian thực** để phát hiện và nhận diện.

**Dữ liệu đầu ra:**

* **File kết quả lưu dữ liệu huấn luyện:**
  + ./face\_dataset/<tên người>.npy
* **Thông tin hiển thị trên màn hình:**
  + Vùng khuôn mặt được nhận diện.
  + Tên người hoặc thông báo "Unknown".
  + Hộp chữ nhật (bounding box) khoanh vùng khuôn mặt.

# **CHƯƠNG III: CÀI ĐẶT VÀ THỰC NGHIỆM**

## **Môi trường và công cụ phát triển**

Môi trường phát triển: Python 12 trên Visual Studio Code

Thư viện và Framework chính:

* Cv2: Xử lý ảnh, thao tác với webcam theo thời gian thực và giúp phát hiện khuôn mặt trong ảnh bằng Haar cascade.
* NumPy: Lưu trữ ảnh dưới dạng vector số và xử lý ma trận dữ liệu khuôn mặt và nhãn.
* Scikit-learn (sklearn): xây dựng pipeline huấn luyện – kiểm thử – đánh giá hiệu quả cho mô hình KNN nhận diện khuôn mặt.
* sklearn.preprocessing.StandardScaler: Chuẩn hóa dữ liệu ảnh và giúp thuật toán KNN hoạt động chính xác hơn, tránh ảnh hưởng bởi độ lệch biên độ giữa các pixel.
* sklearn.decomposition.PCA: Giảm chiều dữ liệu từ 10,000 pixel xuống còn ~30–50 chiều, giúp giảm nhiễu, giảm tính toán, tránh overlifting (quá khớp).
* sklearn.neighbors.KneighborsClassifier: Thuật toán phân loại KNN chính quy, tối ưu hiệu suất và chính xác.
* Os: Đọc dữ liệu từ người dùng.

Mô hình sử dụng: haarcascade\_frontalface\_alt.xml

## **Cài đặt các thành phần**

* Trước khi bắt đầu, tạo một môi trường Python độc lập để đảm bảo không bị xung đột thư viện trong tương lai
* Tải các thư viện chính đã liệt kê ở trên:
* Cv2
* NumPy
* Scikit-learn (sklearn)
* Os

$ pip install opencv-python numpy scikit-learn os-sys

* Import những thư viện và hàm cần thiết:

import numpy as np

import cv2

import os

from sklearn.decomposition import PCA

from sklearn.preprocessing import StandardScaler

from sklearn.neighbors import KNeighborsClassifier

* Tải và cài đặt mô hình haarcascade\_frontalface\_alt vào ứng dụng::
* Truy cập đường link Github của OpenCV và tìm folder opencv/data/haarcascades.
* Chọn haarcascade\_frontalface\_alt.xml và tải xuống.
* Đưa file haarcascade\_frontalface\_alt.xml vào cùng folder với chương trình.
* Dùng câu lệnh sau để tạo object face\_cascade để sử dụng mô hình

face\_cascade = cv2.CascadeClassifier("haarcascade\_frontalface\_alt.xml")

## **Kết quả thực nghiệm**

* Sau khi cài đặt môi trường, tải mô hình Haar Cascade và chạy chương trình, hệ thống đã hoạt động ổn định.
* Kết quả thực nghiệm thu được của chương trình khi ghi chạy bằng web cam, mô hình Haar Cascade đã có khả năng nhận diện các khuôn mặt đã đăng ký và các khuôn mặt lạ (unknown).

## **Đánh giá kết quả**

* Mô hình hoạt động ổn định trong các điều kiện ánh sáng và góc quay khác nhau.
* Mô hình đã nhận diện được các khuôn mặt khác nhau, bao gồm đã và chưa đăng ký.
* Một số góc máy và điều kiện ánh sáng quá đặt thù có thể ảnh hưởng đến khả năng nhận diện của mô hình.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

# **CHƯƠNG III: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN**

## **Tổng kết kết quả**

* Trãi qua thời gian nghiên cứu và phát triển, chương trình đã có thể có thể nạp trực tiếp dữ liệu của người dùng trực tiếp từ web cam.
* Nhóm đã thành công kết hợp mô hình Haar Cascade với KNN để tăng cường độ chính xác.
* Chương trình đã hoàn toàn có thể nhận diện và phân biệt nhiều khuôn mặt khác nhau trong cùng 1 lúc với độ chính xác tương đối cao.

## **Hạn chế của mô hình**

* Giao diện chưa thân thiện với người dùng, vẫn đang chạy bằng phím tắt và terminal chứ chưa có giao diện hoàn chỉnh.
* Mô hình vẫn chưa được so sánh hiệu năng thực tế với các mô hình khác để thu thập thông số chi tiết.
* Thuật toán KNN của mô hình vẫn còn đơn giản, chưa thể nhận diện được các nét đặt trưng của khuôn mặt, vì vậy nên vẫn còn trường hợp mô hình nhầm lẫn giữa những khuôn mặt giống nhau.

## **Định hướng phát triển**

* Nâng cấp thuật toán để cải thiện hiệu xuất, độ chính xác cũng như tốc độ của mô hình.
* So sánh và đánh giá mô hình hiện tại với các mô hình/ thuật toán mới và hiện đại hơn.
* Đưa ra các thông số cụ thể như tính độ chính xác, tốc độ xử lý của mô hình,... tới cho người dùng.
* Xây dựng giao diện hiện đại và thân thiện với người dùng hơn.
* Ứng dụng mô hình vào các mục đích thực tế như bảo mật, điểm danh trong tiết học và chấm công,...

## **Tài liệu tham khảo**

[1] [Cascade Classifier - OpenCV](https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html) (<https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html>)

[2] [K-Nearest Neighbor (KNN) Algorithm - GeeksforGeeks](https://www.geeksforgeeks.org/machine-learning/k-nearest-neighbours/)

(<https://www.geeksforgeeks.org/machine-learning/k-nearest-neighbours/>)

[3] [face detection - Haar Cascade XML - Stack Overflow](https://stackoverflow.com/questions/70066843/haar-cascade-xml)

(<https://stackoverflow.com/questions/70066843/haar-cascade-xml>)

[4] [austinjoyal/haar-cascade-files - GitHub](https://github.com/austinjoyal/haar-cascade-files)

(https://github.com/austinjoyal/haar-cascade-files)

[5] Haar Features for Face Detection | Face Detection - First Principles of Computer Vision (https://youtu.be/ZSqg-fZJ9tQ?si=-EEyIELAUInidsuL)

[6] [What is the K-Nearest Neighbor (KNN) Algorithm?](https://www.youtube.com/watch?v=b6uHw7QW_n4&pp=ygUDa25u) - [IBM Technology](https://www.youtube.com/@IBMTechnology)

(https://youtu.be/b6uHw7QW\_n4?si=5lSqTRUQR1KsIBKk)