**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**Cơ sở TP. HỒ CHÍ MINH**

**A logo of a company

Description automatically generated**

**BÁO CÁO BÀI TẬP LỚN**

**Môn học: Nhập môn trí tuệ nhân tạo**

**Giáo viên hướng dẫn: Nguyễn Thị Tuyết Hải**

**Đề tài: Ứng dụng học máy, học sâu để xây dựng chương trình nhận diện các đặc trưng dựa trên khuôn mặt con người**

**Tên thành viên MSSV**

**Phan Văn Thọ N22DCCN083**

**Lê Võ N22DCCN**

**Ngô Lê Hoài Trung N22DCCN091**

**Trần Nhật Nguyên N22DCCN**

**I. Bài toán đặt ra**

**1. Giới thiệu đề tài**

Trong thời đại công nghệ số hiện nay, việc tích hợp trí tuệ nhân tạo (AI) vào các hệ thống giám sát và hỗ trợ con người ngày càng trở nên phổ biến. Một trong những ứng dụng thiết thực và tiềm năng là xây dựng hệ thống nhận diện trạng thái khuôn mặt nhằm phát hiện trạng thái mệt mỏi, cảm xúc, độ tuổi và giới tính của người sử dụng. Đề tài này tập trung vào việc phát triển một hệ thống thông minh có khả năng nhận diện trạng thái mắt (mở/nhắm), cảm xúc khuôn mặt, tuổi và giới tính của người dùng trong thời gian thực thông qua webcam.

Hệ thống sử dụng thư viện Mediapipe để trích xuất các điểm đặc trưng trên khuôn mặt, kết hợp với các mô hình học sâu đã được huấn luyện trước để dự đoán:

* Trạng thái mắt: Phát hiện người dùng có đang nhắm hay mở mắt, hỗ trợ các ứng dụng như giám sát sự tỉnh táo của người lái xe.
* Cảm xúc: Phân tích biểu cảm khuôn mặt để suy luận trạng thái cảm xúc (vui, buồn, giận, v.v.).
* Tuổi và giới tính: Ước lượng độ tuổi và xác định giới tính người dùng, hỗ trợ phân tích nhân khẩu học.

Dữ liệu đầu vào được thu nhận qua camera và xử lý trong thời gian thực với các luồng xử lý song song (multithreading) nhằm đảm bảo hiệu suất. Ngoài ra, hệ thống còn tích hợp chức năng phát âm bằng tiếng Anh để thông báo trạng thái mắt, giúp tăng tính tương tác và hỗ trợ người dùng không cần nhìn vào màn hình.

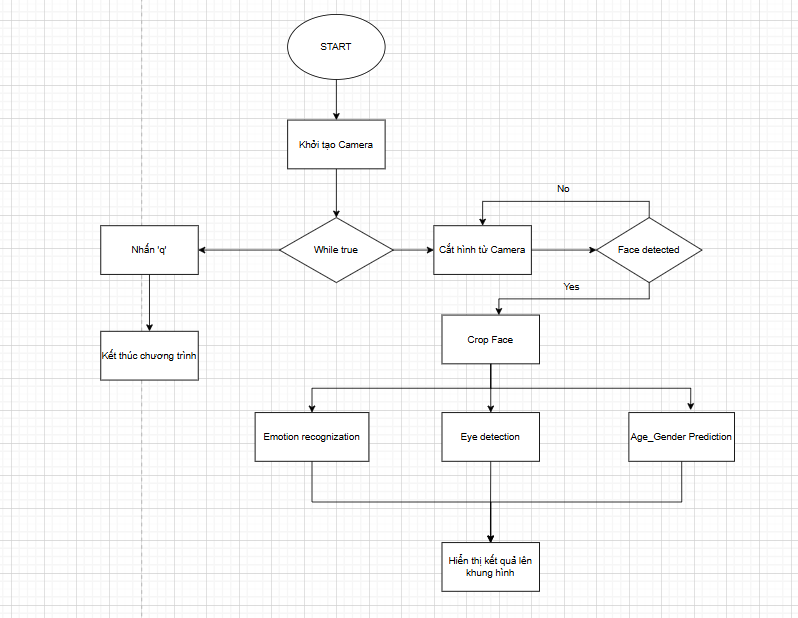
Đề tài mang lại giá trị thực tiễn cao trong các lĩnh vực như: an toàn giao thông, giám sát hành vi, phân tích dữ liệu người dùng, và có thể mở rộng thêm các chức năng nâng cao trong tương lai như nhận diện danh tính hoặc phân tích hành vi phức tạp hơn.

**2. Mục tiêu đề tài**

Mục tiêu chính của đề tài là xây dựng một hệ thống giám sát khuôn mặt trong thời gian thực, có khả năng nhận diện đồng thời nhiều đặc điểm quan trọng từ khuôn mặt người dùng, bao gồm:

1. Nhận diện trạng thái mắt (mở hoặc nhắm)
   * Phát hiện trạng thái nhắm mắt liên tục nhằm cảnh báo nguy cơ buồn ngủ hoặc mất tập trung, đặc biệt hữu ích trong các ứng dụng giám sát người lái xe hoặc nhân viên vận hành máy móc.
2. Nhận diện cảm xúc khuôn mặt
   * Phân tích biểu cảm để xác định cảm xúc hiện tại như vui, buồn, giận, sợ, ngạc nhiên… từ đó có thể ứng dụng trong các hệ thống tương tác người – máy hoặc chăm sóc sức khỏe tinh thần.
3. Dự đoán tuổi và giới tính
   * Ước lượng khoảng tuổi và giới tính của người dùng nhằm phục vụ cho mục đích phân tích nhân khẩu học hoặc cá nhân hóa dịch vụ.
4. Mở rộng thêm
   * Đảm bảo hiệu suất hệ thống trong quá trình xử lý đồng thời nhiều tác vụ (như dự đoán trạng thái mắt, cảm xúc, tuổi, giới tính và phát âm cảnh báo).
   * Tăng khả năng phản hồi và cảnh báo trực quan bằng âm thanh khi phát hiện người dùng đang nhắm mắt hoặc có biểu hiện mệt mỏi.

**II. Lưu đồ giải thuật**



**III. Các bước thực hiện**

1. **Khởi tạo camera, phát hiện khuôn mặt, cắt khuôn mặt, mắt(Thọ)**

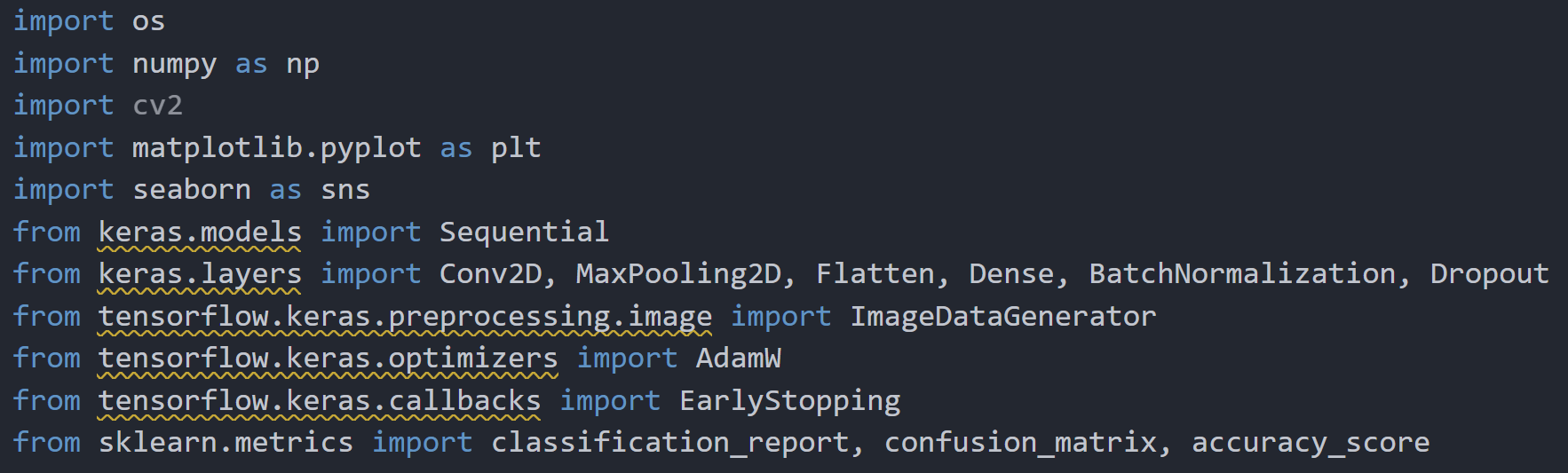
**…**

1. **Nhận diện cảm xúc (Trung)**
   1. **Huấn luyện mô hình**

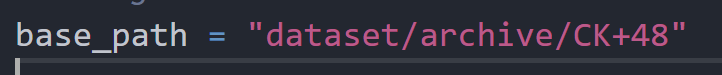
Mô hình nhận diện cảm xúc được huấn luyện để phân loại khuôn mặt thành các trạng thái cảm xúc khác nhau như: Angry (Giận dữ), Disgust (Ghê tởm), Fear (Sợ hãi), Happy (Vui vẻ), Neutral (Trung lập), Sad (Buồn bã), Surprise (Ngạc nhiên).

**Quá trình huấn luyện:**

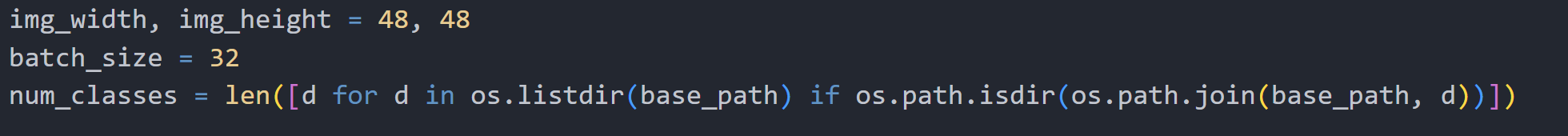
* **Chuẩn bị dữ liệu**



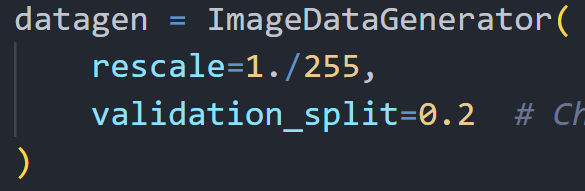
* **Import thư viện**: Các thư viện hỗ trợ xử lý ảnh (OpenCV), xây dựng mô hình (Keras, TensorFlow), tối ưu hóa (AdamW), trực quan hóa (Matplotlib, Seaborn), và đánh giá mô hình (Sklearn).
* **Đường dẫn dữ liệu**



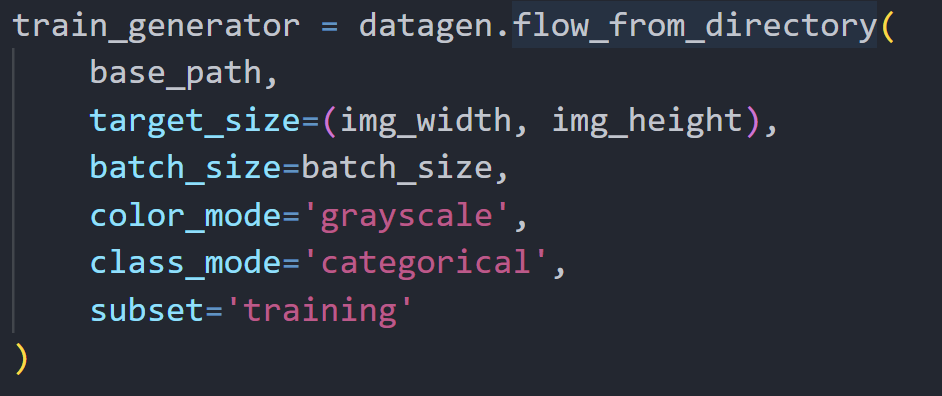
* **Xác định thư mục dữ liệu huấn luyện và kiểm tra**.



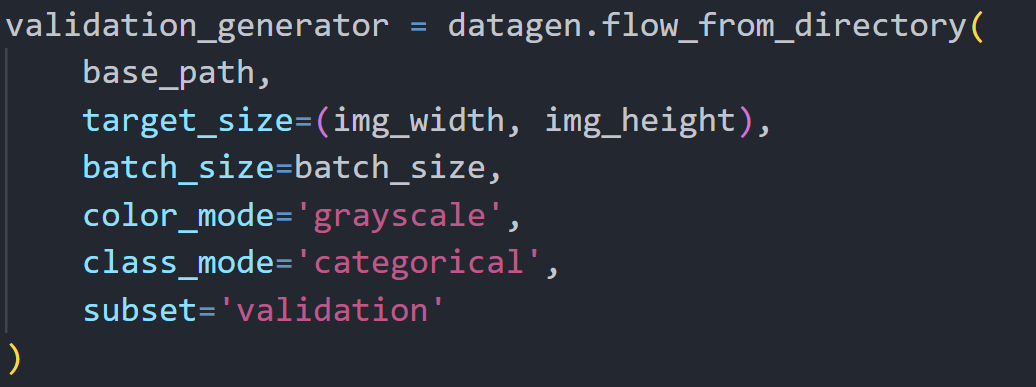
* **Thiết lập thông số ảnh** (48x48 pixel).
* **Số lượng lớp đầu ra** (num\_classes) dựa trên số thư mục (cảm xúc) trong tập huấn luyện.
* **Tiền xử lý dữ liệu**



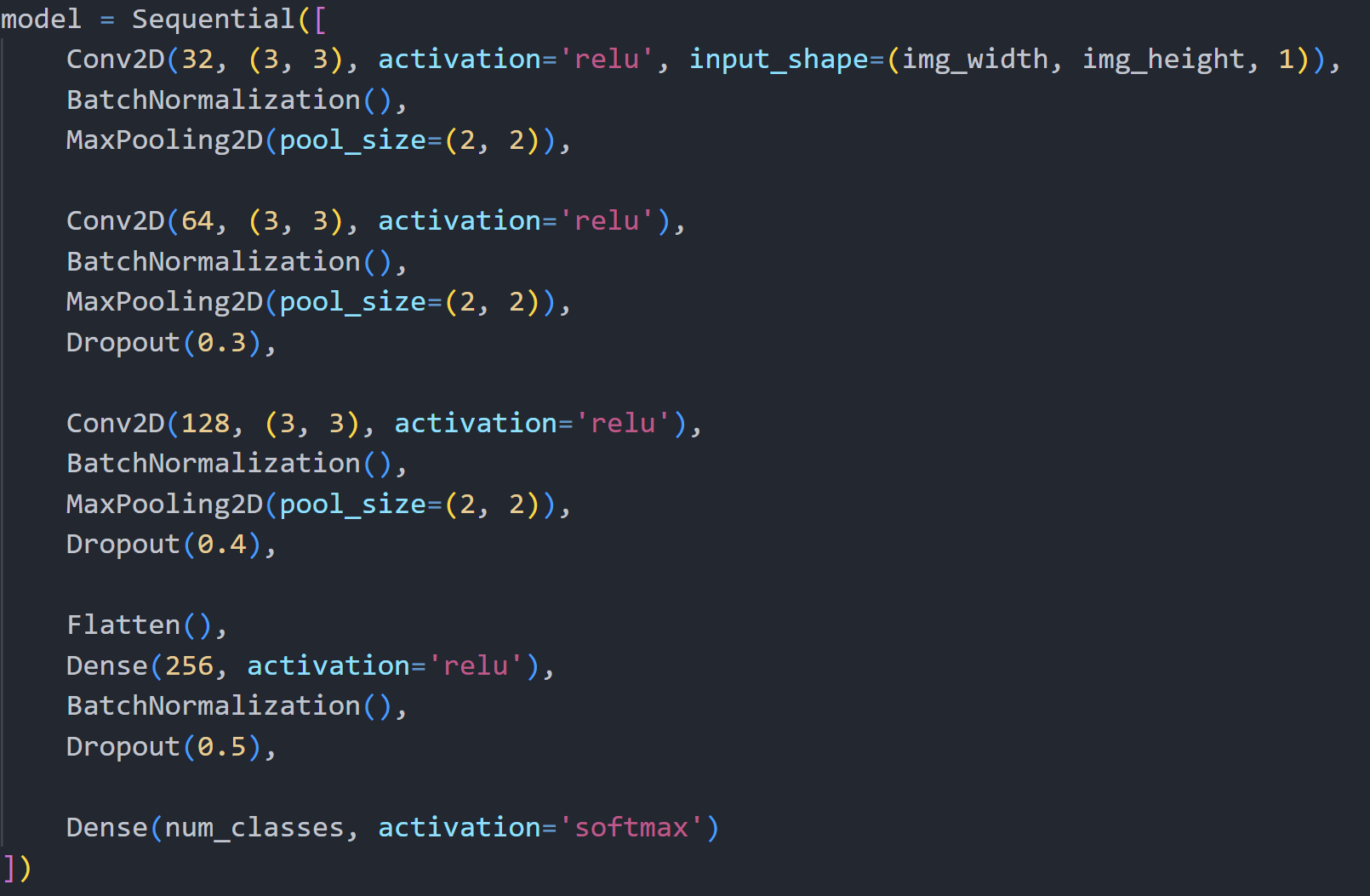
* **Chuẩn hóa ảnh** (rescale=1./255).
* **Chia dữ liệu** (validation\_split=0.2): Dữ liệu huấn luyện (80%) và kiểm tra (20%).
* **Tạo trình tải dữ liệu**



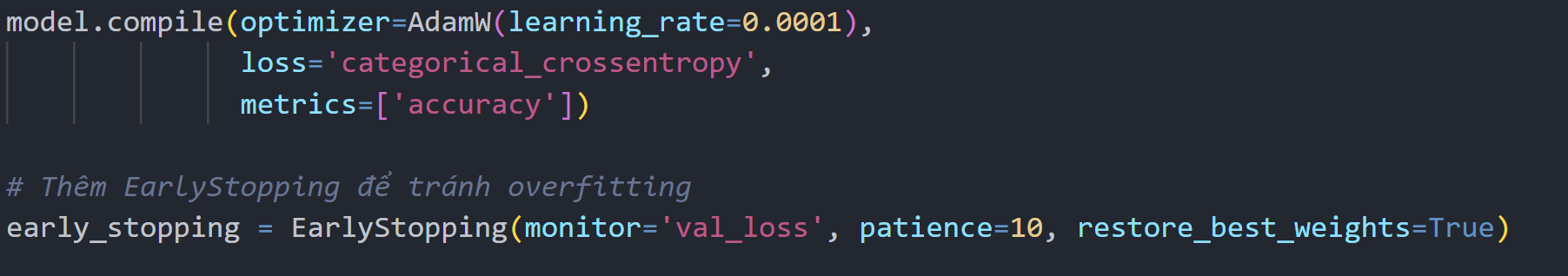
* **Tạo tập huấn luyện** với ảnh **grayscale** và nhãn dạng one-hot encoding.



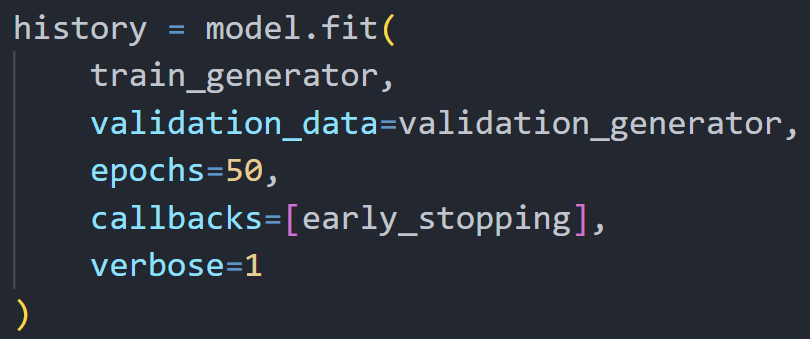
* **Tạo tập kiểm tra**, giúp đánh giá mô hình trong quá trình huấn luyện.
* **Xây dựng mô hình CNN**



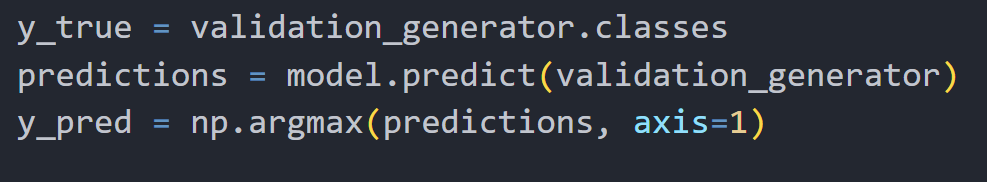
* **Lớp tích chập đầu tiên**: Conv2D(32, (3,3), activation='relu') giúp trích xuất đặc trưng từ ảnh.
* **BatchNormalization**: Giúp tăng tốc độ huấn luyện bằng cách chuẩn hóa dữ liệu.
* **MaxPooling2D** (pool\_size=(2,2)) giúp giảm kích thước ảnh đầu ra.
* **Lớp tích chập thứ hai** với **64 bộ lọc**, kết hợp **Dropout (0.3)** để giảm overfitting.
* **Tăng số bộ lọc lên** 128, giúp mô hình học đặc trưng sâu hơn.
* **Flatten**: Biến ma trận đặc trưng thành vector.
* **Dense(256)**: Lớp kết nối với 256 neuron.
* **Dropout(0.5)**: Giảm nguy cơ overfitting.
* **Lớp đầu ra** sử dụng softmax để phân loại cảm xúc.
* **Cấu hình huấn luyện**



* **AdamW Optimizer** (learning\_rate=0.0001) giúp tối ưu hóa mô hình.
* **categorical\_crossentropy**: Hàm mất mát phù hợp với bài toán phân loại đa lớp.
* **Dừng sớm nếu** val\_loss **không cải thiện sau** 10 **epoch** để tránh overfitting.
* **Huấn luyện mô hình**



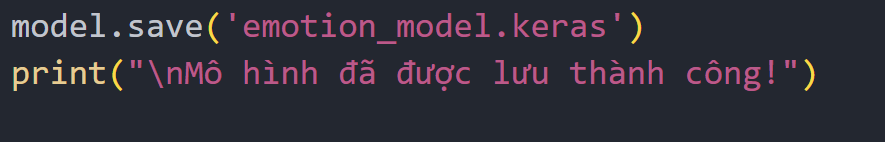
* **Huấn luyện với** 50 **epoch**, kiểm tra bằng tập validation.
* **EarlyStopping** đảm bảo mô hình không bị overfitting.
* **Đánh giá mô hình**



* **Lấy nhãn thực tế** (y\_true) và nhãn dự đoán (y\_pred).



* **Hiển thị báo cáo phân loại** (classification\_report) để xem độ chính xác từng cảm xúc.
* **Tính độ chính xác tổng thể** của mô hình.
* **Hiển thị ma trận nhầm lẫn** (confusion\_matrix) bằng Seaborn.
* **Lưu mô hình**



* **Lưu mô hình** dưới dạng .keras để sử dụng sau này.
  1. **Dự đoán**

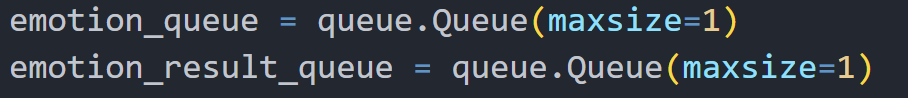
**Cách hoạt động:**

Mô hình nhận diện cảm xúc được áp dụng vào hình ảnh đầu vào và dự đoán cảm xúc của khuôn mặt. Quy trình được thực hiện như sau:

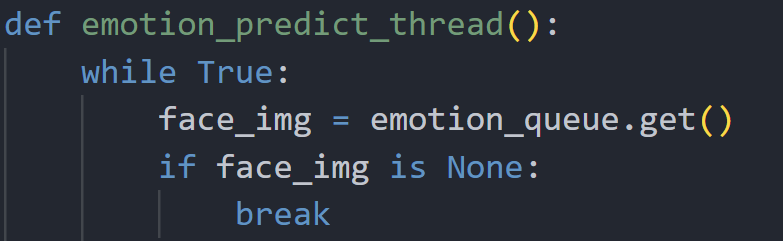
* **Bước 1: Khởi tạo mô hình**



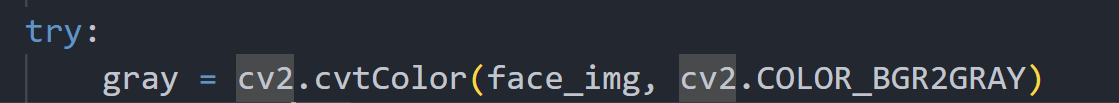
* **load\_model()**: Hàm này tải mô hình học sâu đã được huấn luyện trước đó.
* "model/emotion\_model.keras": Đường dẫn đến tệp mô hình được lưu trữ. Đây là mô hình CNN nhận diện cảm xúc từ ảnh khuôn mặt.
* **Bước 2: Hàng đợi để quản lý dự đoán cảm xúc**



* **emotion\_queue**: Hàng đợi lưu hình ảnh khuôn mặt cần dự đoán cảm xúc.
* **emotion\_result\_queue**: Hàng đợi chứa kết quả cảm xúc sau khi mô hình dự đoán.
* **Bước 3: Luồng dự đoán cảm xúc**



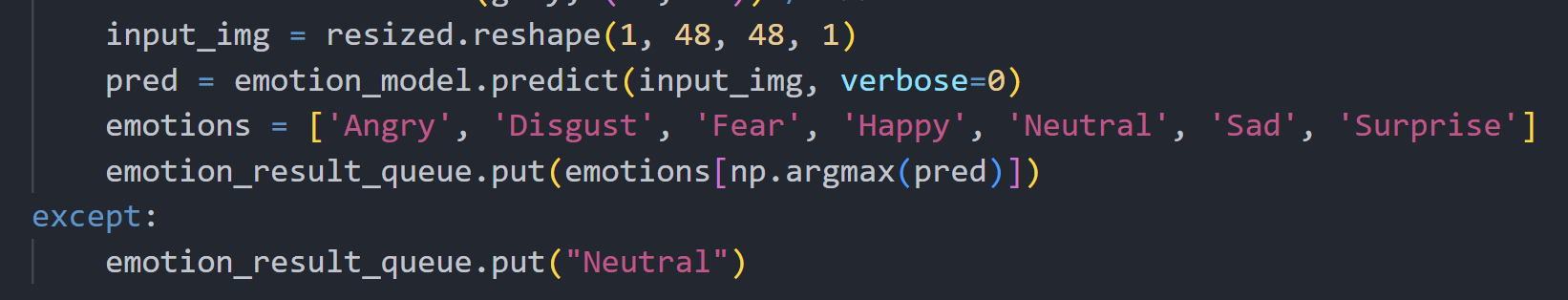
* **Hàm này chạy liên tục** để lấy ảnh từ emotion\_queue và dự đoán cảm xúc.
* **get()**: Lấy ảnh từ hàng đợi.
* **Kiểm tra** None: Nếu ảnh None, tức là không có dữ liệu, thì thoát vòng lặp.



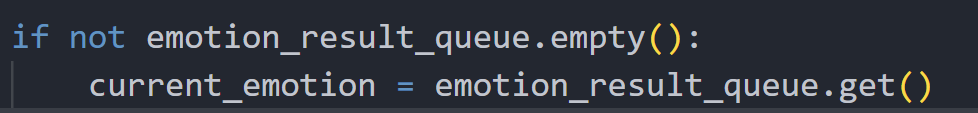
* **cv2.cvtColor()**: Chuyển đổi ảnh sang mức xám (grayscale) để giảm số lượng kênh màu (RGB → Grayscale).



* **cv2.resize()**: Thay đổi kích thước ảnh về 48x48 pixel để phù hợp với đầu vào của mô hình.
* **/255.0**: Chuẩn hóa giá trị pixel (0-255) thành khoảng (0-1).



* **reshape()**: Định dạng lại ảnh thành tensor (1, 48, 48, 1) để mô hình CNN có thể xử lý.
* **predict()**: Gọi mô hình để dự đoán cảm xúc từ ảnh đầu vào.
* **Danh sách** emotions: Danh sách các cảm xúc mà mô hình có thể dự đoán.
* **np.argmax(pred)**: Lấy chỉ số của cảm xúc có xác suất cao nhất.
* **emotion\_result\_queue.put()**: Đưa kết quả vào hàng đợi.
* Nếu có lỗi trong quá trình xử lý, mặc định kết quả là **“Neutral”** để tránh gián đoạn.
* **Bước 4: Nhận kết quả và hiển thị lên màn hình**



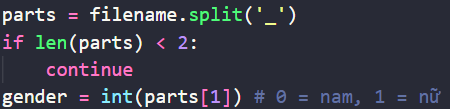
* **Kiểm tra nếu có kết quả dự đoán** (emotion\_result\_queue không rỗng).
* **Lấy cảm xúc mới nhất** từ hàng đợi.



* **Hiển thị cảm xúc** lên màn hình bằng cv2.putText().
* **Màu chữ đỏ** ((0,0,255)) để dễ nhìn.

1. **Nhận diện giới tính ( Võ)**
   1. **Xử lí dữ liệu**

source code lưu ở process\_gender\_dataset.py

Tập dữ liệu gồm các file ảnh, tên ảnh sẽ mang định dạng age\_gender\_race\_date\_time.jpg, cần tách nhãn giới tính từ tên file. 

Lưu vào tập y (nhãn nhận diện giới tính):



chuyển đổi từ BGR (blue-green-red) sang RGB (red-green-blue). OpenCV (cv2) đọc ảnh theo định dạng màu BGR (Blue-Green-Red), trong khi hầu hết các thư viện hiển thị ảnh như matplotlib, PIL hoặc TensorFlow lại sử dụng RGB (Red-Green-Blue). Nếu không chuyển đổi, màu sắc sẽ bị lệch khi hiển thị hoặc huấn luyện mô hình. 

Chuẩn hóa kích thước ảnh



Lưu các ảnh vào tập X:



Sau đó chia các tập train và test:



* 1. **Huấn luyện mô hình**

Khởi tạo mô hình:



1. Conv2D(32, (3,3), activation='relu', padding='same') + MaxPooling2D

Học được các đặc trưng cơ bản: cạnh, góc cạnh.

Padding 'same' giúp giữ nguyên kích thước đầu ra → dễ xử lý về sau.

Pooling 2x2 giảm kích thước không gian (downsampling), tránh quá tải bộ nhớ.

2. Conv2D(64, (3,3), ...) + MaxPooling2D

Tăng số lượng bộ lọc (filters) → trích xuất đặc trưng phức tạp hơn như hình dạng mắt, mũi...

3. Conv2D(128, (3,3), ...) + MaxPooling2D

Càng lên cao, mô hình học được đặc trưng sâu hơn như tỷ lệ khuôn mặt, cấu trúc xương.

4. Flatten + Dense(128, activation='relu')

Chuyển ảnh 3D thành vector 1D để đưa vào tầng fully connected.

Dense với 128 đơn vị giúp học mối liên hệ phi tuyến giữa đặc trưng và nhãn.

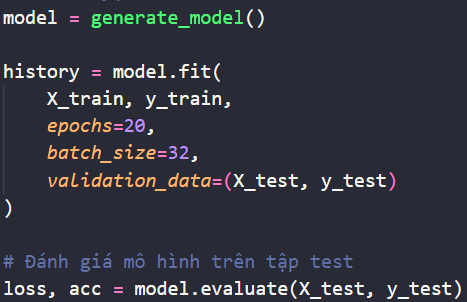
5. Dropout(0.5)

Giảm overfitting bằng cách "bỏ" ngẫu nhiên 50% neuron trong quá trình huấn luyện.

6. Dense(1, activation='sigmoid')

Đầu ra dạng xác suất → phù hợp với bài toán nhị phân.

Huấn luyện và đánh giá



1. **Đoán tuổi (Nguyên)**
2. **Nhận diện trạng thái mắt (Thọ)**
   1. **Huấn luyện mô hình**
   2. **Nhận diện**
3. **Nhận kết quả từ các mô hình và hiển thị lên màn hình (Thọ)**
4. **Kết thúc chương trình (Thọ)**