**I.Các biến và tham chiếu**

1. **cham**: Biến trạng thái (state) quản lý trạng thái chạm. Nếu cham là true, thì ứng dụng nhận diện là đã có hành động chạm vào video, ngược lại là không chạm.
2. **video**: Sử dụng useRef() để tạo tham chiếu đến phần tử video HTML5. Đây là nơi truyền tải luồng video từ camera để xử lý nhận diện.
3. **model**: Tham chiếu đến mô hình Mobilenet. Mobilenet là một mô hình nhẹ, giúp trích xuất các đặc trưng hình ảnh, chuẩn bị cho việc huấn luyện và phân loại.
4. **canPlay**: Biến trạng thái boolean giúp kiểm soát âm thanh khi phát hiện "chạm". Nếu canPlay là true, âm thanh sẽ phát, và ngược lại.
5. **classifier**: Tham chiếu đến bộ phân loại KNN (K-Nearest Neighbors). Bộ phân loại KNN này sẽ phân tích các mẫu hình ảnh đã gán nhãn và đưa ra dự đoán về hành động "chạm" hoặc "không chạm".

**II.Hàm init**

Hàm init thực hiện các thao tác khởi tạo và chuẩn bị:

1. **Thiết lập và khởi động TensorFlow.js**:
   * tf.setBackend('webgl'): Cài đặt backend là webgl để tăng tốc độ xử lý của TensorFlow bằng GPU.
   * tf.ready(): Chờ TensorFlow sẵn sàng.
2. **Thiết lập camera**:
   * setupCamera(): Thiết lập quyền truy cập vào camera của thiết bị để sử dụng luồng video trong ứng dụng.
3. **Tải mô hình Mobilenet và khởi tạo classifier**:
   * model.current = await mobilenet.load(): Nạp mô hình Mobilenet và lưu vào model.current.
   * classifier.current = knnClassifier.create(): Tạo bộ phân loại KNN.
4. **Khởi tạo thông báo**:
   * initNotifications({ cooldown: 3000 }): Khởi tạo hệ thống thông báo với thời gian làm mới là 3 giây.

**III.Hàm setupCamera**

Đây là hàm cài đặt camera, bao gồm:

1. **getUserMedia**: Yêu cầu quyền truy cập camera.
2. **Luồng video**:
   * Nếu quyền được cấp, gán luồng video (stream) vào video.current.srcObject.
   * Đợi loadedmetadata của phần tử video tải xong để đảm bảo video sẵn sàng, sau đó giải quyết Promise bằng resolve().
   * Nếu có lỗi trong quá trình truy cập camera, hàm sẽ ghi log và từ chối Promise bằng reject(error).

**IV.Hàm train**

Hàm train giúp huấn luyện bộ phân loại với nhãn cụ thể (label) trong time lần lặp:

1. **Huấn luyện theo nhãn**:
   * Vòng lặp for thực hiện time lần huấn luyện với nhãn label.
   * training(label): Mỗi lần gọi hàm training sẽ thêm một mẫu đặc trưng (embedding) với nhãn label vào bộ phân loại.

**V.Hàm training**

Đây là hàm lấy mẫu từ video và thêm vào bộ phân loại KNN:

1. **Trích xuất đặc trưng**:
   * embedding = model.current.infer(video.current, true): Lấy đặc trưng (embedding) từ mô hình Mobilenet dựa trên khung hình hiện tại của video.
2. **Thêm mẫu**:
   * classifier.current.addExample(embedding, label): Thêm đặc trưng embedding với nhãn label vào bộ phân loại.
3. **Tạm dừng**:
   * sleep(200): Tạm dừng 200ms giữa các mẫu để tránh việc lấy mẫu quá nhanh.

**VI.Hàm run**

Hàm run dùng để liên tục kiểm tra dự đoán hành động của video:

1. **Dự đoán hành động**:
   * result = await classifier.current.predictClass(embedding): Dự đoán nhãn của đặc trưng hiện tại embedding.
2. **Xử lý kết quả dự đoán**:
   * Nếu result.label === dacham (dự đoán là hành động "chạm") và result.confidences[result.label] > time\_hien (độ tin cậy cao hơn time\_hien), nghĩa là có hành động chạm:
     + setCham(true): Cập nhật trạng thái chạm.
     + Phát âm thanh nếu canPlay.current là true, sau đó đặt canPlay.current = false để không phát lại liên tục.
   * Nếu không chạm:
     + setCham(false): Đặt trạng thái chạm về false.
3. **Lặp lại dự đoán**:
   * await sleep(200): Tạm dừng 200ms giữa mỗi lần chạy hàm run để tránh quá tải xử lý.
   * Gọi lại run() để tiếp tục kiểm tra liên tục.

**VII.Hàm sleep**

Hàm sleep tạo độ trễ trong một khoảng thời gian ms.

**VIII.Hook useEffect**

Hook useEffect này chạy khi ứng dụng khởi động:

1. **Khởi tạo**: Gọi hàm init() để chuẩn bị các thành phần cần thiết.
2. **Lắng nghe sự kiện âm thanh kết thúc**: Sau khi âm thanh phát xong (am.on('end', ...)), đặt canPlay.current = true để cho phép phát lại âm thanh nếu phát hiện chạm lần nữa.

**IX.Giao diện App**

1. **Thẻ video**:
   * <video ref={video} className="video" autoPlay />: Thẻ video lấy luồng từ camera, sử dụng thuộc tính ref để kết nối với video trong mã JavaScript.
2. **Nút điều khiển**:
   * Các nút nhấn có các chức năng:
     + onClick={() => train(chuacham)}: Gán nhãn "chuacham" để huấn luyện mẫu khi chưa chạm.
     + onClick={() => train(dacham)}: Gán nhãn "dacham" để huấn luyện mẫu khi chạm.
     + onClick={() => run()}: Bắt đầu dự đoán liên tục hành động chạm.

**X.CSS**

1. **Điều khiển hiển thị**:
   * className={App ${cham ? 'cham' :''}} : Nếu cham là true, thêm lớp cham để thay đổi giao diện ứng dụng khi phát hiện chạm.

**XI.Tóm tắt quy trình hoạt động**

1. Ứng dụng khởi tạo mô hình và thiết lập camera.
2. Người dùng nhấn nút để huấn luyện ứng dụng với các nhãn khác nhau ("chạm" và "không chạm").
3. Sau khi huấn luyện, người dùng nhấn RUN để bắt đầu dự đoán.
4. Khi nhận diện "chạm" được phát hiện với độ tin cậy cao, ứng dụng phát âm thanh và thay đổi giao diện để biểu thị đã có chạm.