BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC NGUYỄN TẤT THÀNH
KHOA KỸ THUẬT – CÔNG NGHỆ

TIỂU LUẬN

MÔN: THỊ GIÁC MÁY TÍNH

Tp. HCM - Tháng 09/2023

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC NGUYỄN TẤT THÀNH
KHOA KỸ THUẬT – CÔNG NGHỆ

ĐỀ TÀI TIỂU LUẬN:
Ứng dụng thị giác máy tính để tụ động nhận diện tình trạng buồn ngủ khi lái xe của tài xế .

Nhóm 9
Thành viên:
- Phan Thượng Khánh - 2000003912
- Phạm Hoàng Vũ – 2000000830
- Nguyễn Vi Khang -2000005536
- Lê Trần Quang Phúc-2000006517

Giảng viên hướng dẫn:
Nguyễn Hoàng Nam

Tp. HCM - Tháng 09/2023

LỜI CAM ĐOAN

Em/ chúng em xin cam đoan đề tài: '**Ứng dụng thị giác máy tính để tụ động nhận diện tình trạng buồn ngủ khi lái xe của tài xế '** do nhóm 5 nghiên cứu và thực hiện.

Chúng em đã kiểm tra dữ liệu theo quy định hiện hành. Kết quả bài làm của đề tài '**Ứng dụng thị giác máy tính để tụ động nhận diện tình trạng buồn ngủ khi lái xe của tài xế'** là trung thực và không sao chép từ bất kỳ bài tập của nhóm khác .

Các tài liệu được sử dụng trong tiểu luận có nguồn gốc, xuất xứ rõ ràng

Ngày 04 tháng 04 năm 2023

Sinh viên thực hiện

(ký và ghi họ tên)

Phan Thượng Khánh

Phạm hoàng vũ

Nguyễn Vi Khang

Lê Trần Quang Phúc

LỜI CẢM ƠN

Lời đầu tiên, xin trân trọng cảm ơn Thầy đã hướng dẫn, Thầy đã tận tình hướng dẫn nhóm trong quá trình học tập cũng như trong việc hoàn thành tiểu luận.

Xin cảm ơn các Thầy đã đọc tiểu luận và cho nhóm những nhận xét quý báu, chỉnh sửa những sai sót của nhóm trong bài tiểu luận.

Do giới hạn kiến thức và khả năng lý luận còn nhiều thiếu sót và hạn chế, kính mong sự chỉ dẫn và đóng góp của các thầy để bài tiểu luận của nhóm được hoàn thiện hơn.

Xin chân thành cảm ơn!

Ngày 04 tháng 04 năm 2023

Sinh viên thực hiện

(ký và ghi họ tên)

Phan Thượng Khánh

Phạm hoàng vũ

Nguyễn Vi Khang

Lê Trần Quang Phúc

PHỤ LỤC

PHẦN MỞ ĐẦU

1. Lý do chọn đề tài:

Hiện nay, trên toàn thế giới và đặc biệt là tại Việt Nam, Trí tuệ nhân tạo (AI) được xem là một trong những công nghệ cốt lõi của Cuộc cách mạng Công nghiệp 4.0. Nhiều quốc gia đã bắt đầu nhận thức sâu rộng về xu hướng phát triển của AI và tác động biến đổi mạnh mẽ của nó đối với mọi khía cạnh của cuộc sống xã hội, từ việc thay đổi cán cân quyền lực trong lĩnh vực kinh tế, quân sự cho đến chính trị. Để đảm bảo sự phát triển kinh tế - xã hội bền vững và mang lại thịnh vượng cho đất nước, nhiều quốc gia đã đưa ra nhiều giải pháp phù hợp với tốc độ biến đổi nhanh chóng của Cuộc cách mạng Công nghiệp hiện tại.

Trong ngữ cảnh này, công nghệ Sinh trắc học đóng một vai trò quan trọng. Công nghệ Sinh trắc học sử dụng các đặc điểm vật lý và sinh học độc đáo của từng cá nhân, như cảm xúc, võng mạc, khuôn mặt, vân tay, để thực hiện các nhiệm vụ như nhận dạng và xác thực người dùng. Ngoài việc sử dụng công nghệ Sinh trắc học trong các hệ thống bảo mật, các nhà nghiên cứu cũng đã bắt đầu tìm hiểu về cách sử dụng nó để đánh giá tình trạng của tài xế khi họ đang lái xe, nhằm đảm bảo an toàn giao thông.

Cảm xúc và biểu hiện của chúng ta điều hòa cùng cuộc sống hàng ngày và đóng vai trò quan trọng trong giao tiếp không lời. Cách chúng ta biểu hiện cảm xúc có thể thể hiện qua văn bản, đối thoại, cử chỉ cơ thể và biểu hiện khuôn mặt. Trong số đó, nhận dạng cảm xúc qua biểu hiện khuôn mặt và ánh mắt được coi là một phương pháp phổ biến, hiệu quả và dễ thực hiện. Chính vì vậy, tôi đã chọn đề tài "**Ứng dụng thị giác máy tính để tự động nhận diện tình trạng buồn ngủ của tài xế khi lái xe**" để nghiên cứu. Đề tài này nhằm mục đích áp dụng những kiến thức đã học để nhận biết trực tiếp tình trạng buồn ngủ của tài xế khi họ đang lái xe, nhằm tăng cường an toàn giao thông và giảm nguy cơ tai nạn giao thông liên quan đến tình trạng mất tập trung hoặc buồn ngủ của tài xế.

2. Mục tiêu đề tài:

Mô hình "**Ứng dụng thị giác máy tính để tụ động nhận diện tình trạng buồn ngủ khi lái xe của tài xế"** ,chúng em tạo ra nhằm mục tiêu giúp cho việc tham gia giao thông của người Việt Nam an toàn hơn trên đường. Tránh tình trạng gây ra tai nạn giao thông khi tham gia giao thông khi buồn ngủ.

3. Đối tượng nghiên cứu:

Cảm xúc và biểu hiện khuôn mặt đề cập đến trạng thái trải nghiệm tích cực hoặc tiêu cực liên quan đến các tác động lên hệ thống vận hành cả thể lý và tâm lý, gây ra sự thay đổi về mặt sinh lý, hành vi và suy nghĩ của một người hoặc người tham gia giao thông. Vì vậy, đối tượng nghiên cứu trong đề tài nhằm mục đích áp dụng các khía cạnh của cảm xúc và biểu hiện khuôn mặt đối với người tham gia giao thông, đặc biệt là việc nhận diện tình trạng buồn ngủ khi lái xe.

4. Phương pháp nghiên cứu:

Phương pháp tài liệu:

- Tìm hiểu lập trình Python và cơ sở lý thuyết về thuật toán CNN.
- Tìm hiểu các phương pháp nhận dạng.
- Tìm hiểu và lựa chọn công cụ hỗ trợ
- Pyimagesearch.com
- Kaggle.com

Phương pháp thực nghiệm:

- Xây dựng chương trình thực nghiệm.

5. Phạm vi nghiên cứu

Phạm vi nghiên cứu của đề tài chỉ xem xét đến thuật toán nhận dạng biểu hiện buồn ngủ qua video trực tiếp từ webcam. Riêng việc nhận dạng cảm xúc cho luồng video trực tuyến được thực hiện trong điều kiện đầy đủ ánh sáng, gương mặt chính diện và khoảng cách nhận dạng khuôn mặt trong khoảng 1m. Dataset cho việc huấn luyện được sử dụng từ tập dữ liệu được thu nhập được từ chính các cá nhân thành viên trong nhóm.

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

1. Khuôn mặt và cảm xúc:

1.1. Đặc trưng của mặt người:

Khuôn mặt là trọng tâm chính trong mối quan hệ giao tiếp trong xã hội, đóng vai trò quan trọng trong việc truyền tải bản sắc và cảm xúc. Chúng ta có thể nhận ra khuôn mặt của rất nhiều người trong suốt cuộc đời và việc nhận dạng khuôn mặt quen thuộc chỉ trong nháy mắt thậm chí sau nhiều năm không gặp. Điều này khá là rõ nét, bất chấp những thay đổi lớn về thị giác, biểu hiện lão hóa hoặc những thay đổi về kiểu tóc, về mắt kính... Ngoài ra, ảnh khuôn mặt trong thực tế còn chứa đựng rất nhiều vấn đề như: độ sáng, độ mờ, độ nhiễu, độ phân giải và góc ảnh.

Từ khuôn mặt của một người ta có thể khai thác nhiều thông tin liên quan đến người đó không chỉ là danh tính, ví dụ như giới tính, màu da, hướng nhìn, chủng tộc, hành vi, sức khỏe, độ tuổi, cảm xúc…

1.2. Biểu hiện trên khuôn mặt người:

Khuôn mặt đóng một vai trò quan trọng trong việc nhận diện và giao tiếp trong cuộc sống hàng ngày, đặc biệt là trong ngữ cảnh của việc nhận diện tình trạng buồn ngủ khi lái xe. Khuôn mặt của tài xế có thể cung cấp nhiều thông tin quan trọng liên quan đến tình trạng lái xe, và điều này không chỉ giới hạn ở việc xác định danh tính của tài xế. Nó cũng có thể chứa các thông tin như:

- **Biểu hiện khuôn mặt**: Khuôn mặt có thể phản ánh biểu hiện tâm trạng của tài xế, bao gồm sự mệt mỏi, buồn ngủ, hoặc tình trạng sảng khoái. Sự phân biệt giữa các biểu hiện này có thể giúp xác định xem tài xế có dấn thân vào tình trạng buồn ngủ hay không.
- **Hướng nhìn**: Vị trí của đôi mắt trong khuôn mặt tài xế có thể cho thấy hướng nhìn của họ. Việc theo dõi hướng nhìn có thể giúp xác định sự tập trung của tài xế vào đường và tình hình giao thông.
- **Cảm xúc**: Khuôn mặt có thể truyền tải cảm xúc của tài xế, bao gồm cả cảm xúc tích cực và tiêu cực. Sự phát hiện của cảm xúc buồn ngủ hoặc không tập trung có thể được thực hiện thông qua những biểu hiện như sự mệt mỏi và ánh mắt mơ màng.

- **Độ sáng, độ mờ, độ nhiễu và góc ảnh**: Những yếu tố này trong ảnh khuôn mặt có thể ảnh hưởng đến khả năng nhận diện và phân tích. Điều này đặc biệt quan trọng trong việc phát triển các hệ thống nhận diện tình trạng buồn ngủ khi lái xe, bởi vì chúng phải xử lý các ảnh chất lượng thấp và có điều kiện ánh sáng khác nhau trong môi trường lái xe.

Tóm lại, khuôn mặt của tài xế không chỉ chứa thông tin về danh tính mà còn chứa các dấu hiệu cảm xúc, tình trạng tập trung và sức khỏe của họ. Việc nhận diện tình trạng buồn ngủ khi lái xe dựa trên khuôn mặt có thể giúp cải thiện an toàn giao thông và tránh các tình huống nguy hiểm.

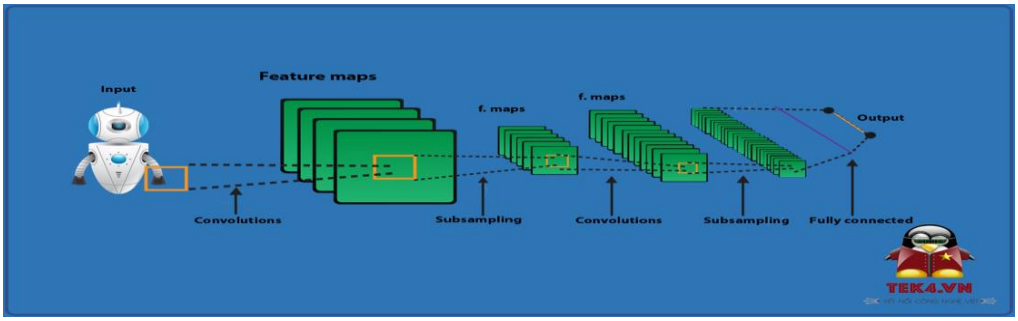
2. Mạng nơ-ron tích chập (CNN):

2.1. Giới thiệu về mạng CNN:

Mạng CNN (Convolutional Neural Network) là một trong những mô hình để nhận dạng và phân loại hình ảnh. Trong đó, xác định đối tượng và nhận dạng khuôn mặt là 1 trong số những lĩnh vực mà CNN được sử dụng rộng rãi có khả năng nhận dạng và phân loại hình ảnh với độ chính xác rất cao, thậm chí còn tốt hơn con người trong nhiều trường hợp. Mô hình này đã và đang được phát triển, ứng dụng vào các hệ thống xử lý ảnh lớn của Facebook, Google hay Amazon được dùng cho các mục đích khác nhau như tự động tìm kiếm ảnh hoặc gợi ý sản phẩm cho người tiêu dùng.

CNN phân loại hình ảnh bằng cách lấy 1 hình ảnh đầu vào, xử lý và phân loại nó theo các hạng mục nhất định. Máy tính coi hình ảnh đầu vào là 1 mảng pixel và nó phụ thuộc vào độ phân giải của hình ảnh. Dựa trên độ phân giải hình ảnh, máy tính sẽ thấy H x W x D (H: Chiều cao, W: Chiều rộng, D: Độ dày).

Về kỹ thuật, mô hình CNN để training và kiểm tra, mỗi hình ảnh đầu vào sẽ chuyển nó qua 1 loạt các lớp tích chập với các bộ lọc (Kernals), tổng hợp lại các lớp được kết nối đầy đủ (Full Connected) và áp dụng hàm Softmax để phân loại đối tượng có giá trị xác suất giữa 0 và 1. Hình dưới đây là toàn bộ luồng CNN để xử lý hình ảnh đầu vào và phân loại các đối tượng dựa trên giá trị.

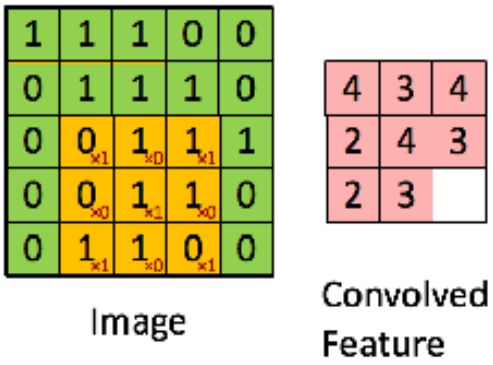


2.2. Các lớp trong mô hình:

Lớp chập – Convolutional

Convolution: bao gồm Convolufion Filter và Convolutional Layer

- Convolution Layer: Chính là các hidden layer – một tập các feature map , mỗi feature là 1 bản scan của input ban đầu nhưng được trích xuất ra các feature với đặc tính cụ thể.
- Convolution Filter(Kernel): Đây là một ma trận sẽ quét qua ma trận dữ liệu đầu vào, từ trái qua phải, trên xuống dưới, và nhân tương ứng từng giá trị của ma trận đầu vào mà ma trận kernel rồi cộng tổng lại, đưa qua activation funeiton (sigmoid, relu, elu,....), kết quả sẽ là một con số cụ thể, tập hợp các con số này lại là một ma trận nữa. chính là feature map.
- Stride: Là khoảng cách giữa 2 kernel khi quét



Hình 1: Minh họa tích chập

Lớp RELU

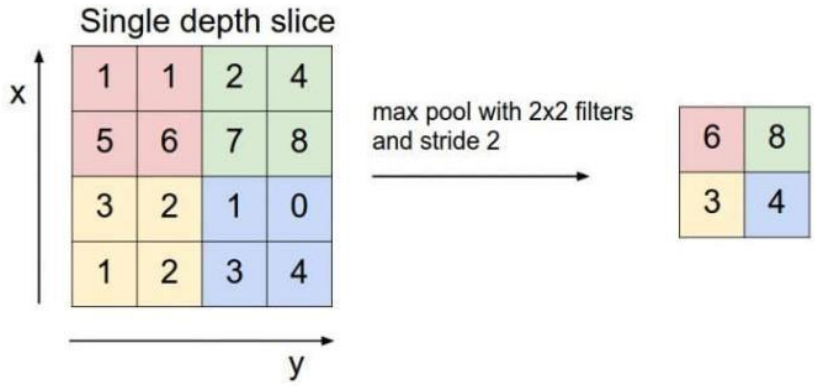
Lớp này thường được cài đặt ngay sau lớp chập. Lớp này sử dụng hàm kích hoạt *f(x) = max (0, x)* nhằm chuyển toàn bộ giá trị âm trong kết quả lấy từ lớp chập thành giá trị Ý nghĩa của cách cài đặt này chính là tạo nên tính phi tuyến cho mô hình. Tương tự như trong mạng truyền thẳng, việc xây dựng dựa trên các phép biến đổi tuyến tính sẽ khiến việc xây dựng đa tầng đa lớp trở nên vô nghĩa. Có rất nhiều cách để khiến mô hình trở nên phi tuyến như sử dụng các hàm kích hoạt sigmoid, tanh… nhưng *f (x) = max (0, x)* hàm dễ cài đặt, tính toán nhanh mà vẫn hiệu quả.

Lớp gộp (Pooling)

Lớp pooling sẽ giảm bớt số lượng tham số khi hình ảnh quá lớn. Không gian pooling còn được gọi là lấy mẫu con hoặc lấy mẫu xuống làm giảm kích thước của mỗi map nhưng vẫn giữ lại thông tin quan trọng. Các pooling có thể có nhiều loại khác nhau:

- Max Pooling
- Average Pooling
- Sum Pooling

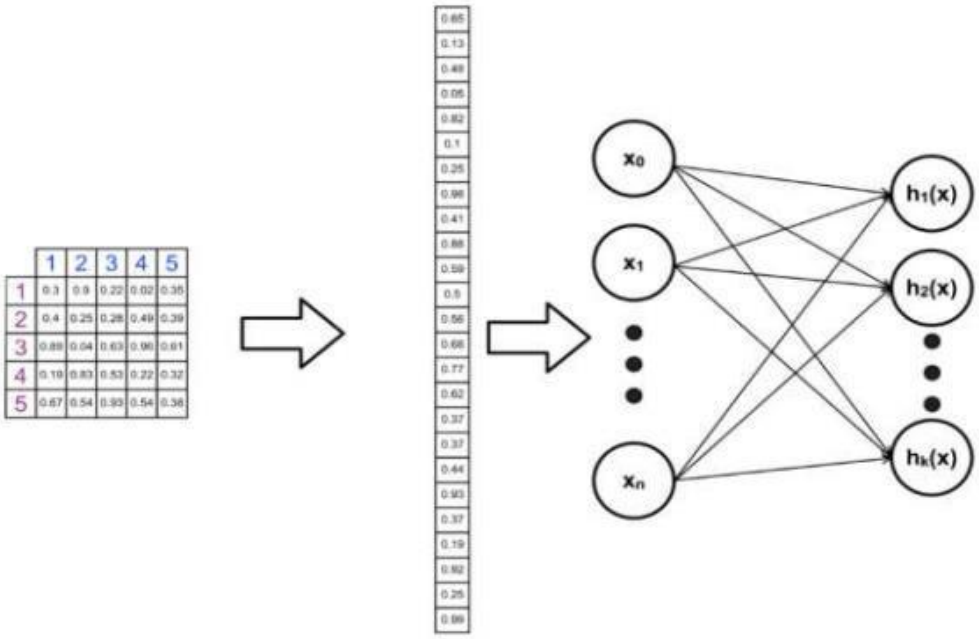
Max pooling lấy phần tử lớn nhất từ ma trận đối tượng, hoặc lấy tổng trung bình. Tổng tất cả các phần tử trong map gọi là sum pooling



Hình 2: Minh họa Maxpooling

Lớp kết nối đầy đủ (Fully-connected)

Lớp này tương tự với lớp trong mạng nơ-ron truyền thẳng, các giá trị ảnh được liên kết đầy đủ vào nút trong lớp tiếp theo. Sau khi ảnh được xử lý và rút trích đặc trưng từ các lớp trước đó, dữ liệu ảnh sẽ không còn quá lớn so với mô hình truyền thẳng nên ta có thể sử dụng mô hình truyền thẳng để tiến hành nhận dạng. Tóm lại, lớp kết nối đầy đủ đóng vai trò như một mô hình phân lớp và tiến hành dựa trên dữ liệu đã được xử lý ở các lớp trước đó.



Hình 3: Minh họa lớp kết nối đầy đủ

Hàm Softmax

Hàm Softmax được sử dụng trong lớp cuối cùng của thuật toán phân loại sử dụng mạng nơtron, với chức năng làm nổi bật giá trị lớn nhất và làm mờ các giá trị bé hơn đáng kể so với giá trị lớn nhật.

Công thức của hàm sofmax:

$$\text{ai} = \frac{\exp(zi)}{\sum \exp(zi)} \qquad với\ i = 0,1,2, \dots C$$

3. Xử lý ảnh với Python:

3.1. Tổng quan về ngôn ngữ Python:

- Cú pháp rất tường minh, dễ đọc.
- Các khả năng tự xét mạnh mẽ.
- Hướng đối tượng trực giác.
- Cách thể hiện tự nhiên mã thủ tục.
- Hoàn toàn mô-đun hóa, hỗ trợ các gói theo cấp bậc.
- Xử lý lỗi dựa theo ngoại lệ.
- Kiểu dữ liệu động ở mức rất cao.
- Các thư viện chuẩn và các mô-đun ngoài bao quát hầu như mọi việc.
- Phần mở rộng và mô-đun dễ dàng viết trong C, C++.
- Có thể nhúng trong ứng dụng như một giao diện kịch bản (scripting interface).
- Python mạnh mẽ và thực hiện rất nhanh.

3.2. Các thư viện được sử dụng:

Trong đề tài này sử dụng bộ phân lớp Haar-Cascade của thư viện OpenCV, Keras, Tkinter và một số thư viện cơ khác.

- **Thư viện OpenCV**

OpenCV là một thư viện mã nguồn mở hàng đầu cho thị giác máy tính, xử lý ảnh và máy học. Nó chứa hàng ngàn thuật toán tối ưu hoá, trong đó cung cấp một bộ công cụ phổ biến cho các ứng dụng về thị giác máy tính. OpenCV đang được sử dụng trong rất nhiều ứng dụng, từ khâu hình ảnh xem đường của Google tới việc chạy các chương trình nghệ thuật tương tác, nhận diện khuôn mặt, hay rô-bốt, xe hơi tự lái.

Trong đề tài này, OpenCV được sử dụng cho các thao tác đọc, xử lý ảnh cơ bản,

phần cài đặt thuật toán có thể sử dụng thư viện Haar-Cascade trong việc phát hiện vị trí khuôn mặt để cải thiện tốc độ xử lý của bài toán. Tuy nhiên, với độ chính xác tương đối kém của nó thì Haar-Cascade chỉ được áp dụng ở công đoạn đòi hỏi tốc độ xử lý cao mà không yêu cầu quá cao về độ chính xác.

- **Thư viện Keras**

Keras là một thư viện được phát triển vào năm 2015 bởi François Chollet, là một kỹ sư nghiên cứu lĩnh vực học sâu tại google. Nó là một thư viện mã nguồn mở hàng đầu cho mạng nơ-ron nhân tạo được viết bởi ngôn ngữ python. Keras là một API bậc cao có thể sử dụng chung với các thư viện học sâu nổi tiếng như tensorflow(được phát triển bởi google), Microsoft Cognitive Toolkit (được phát triển bởi microsoft), theano (người phát triển chính Yoshua Bengio). Keras có một số ưu điểm như :

- Dễ sử dụng, xây dựng model nhanh.
- Có thể run trên cả cpu và gpu
- Hỗ trợ xây dựng CNN , RNN và có thể kết hợp cả 2.

Cấu trúc của Keras có thể được chia ra thành 3 phần chính:

- Hàm chức năng để dựng bộ xương cho model.
- Hàm chức năng dùng để tiền dữ liệu.
- Hàm chức năng ở trong bộ xương của model.

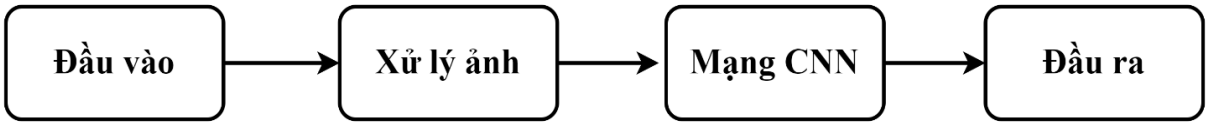
- **Thư viện TensorFlow**

Là một thư viện mã nguồn mở dùng để xây dựng và triển khai các mô hình học máy và học sâu. Ban đầu được phát triển bởi nhóm nghiên cứu Google Brain, TensorFlow đã trở thành một trong những thư viện phổ biến nhất và mạnh mẽ nhất trong lĩnh vực trí tuệ nhân tạo.

TensorFlow không chỉ hỗ trợ việc xây dựng mô hình, mà còn cung cấp các công cụ để quản lý dữ liệu, tiền xử lý, tối ưu hóa và triển khai mô hình trên nhiều nền tảng khác nhau, bao gồm cả máy tính xách tay, máy tính cá nhân, máy chủ và các thiết bị di động.

CHƯƠNG 2: XÂY DỰNG MẠNG NƠTRON NHẬN DẠNG CẢM XÚC

1. Mô Hình hệ thống:

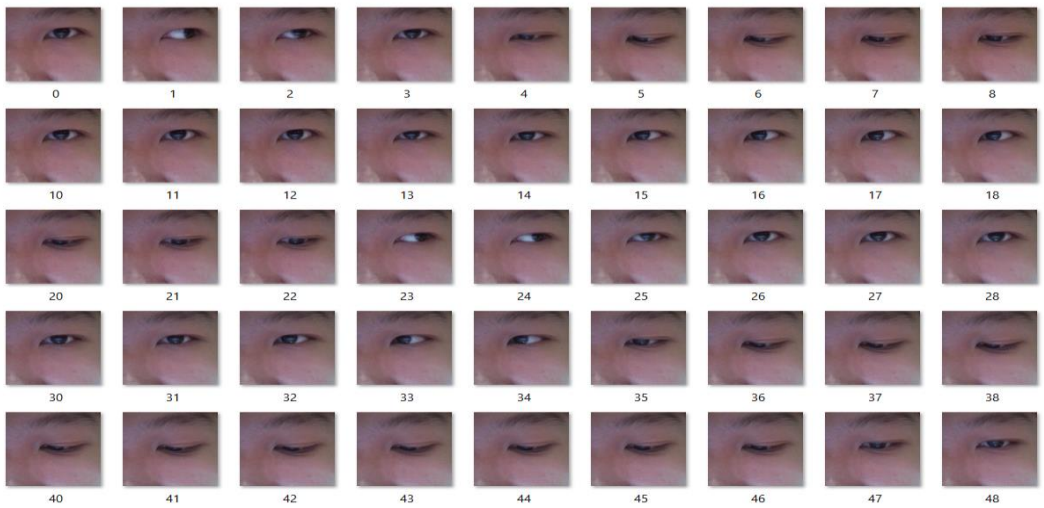


Hình 4: Lưu đồ giải thuật

Hình ảnh đầu vào được trích xuất từ ảnh, video hoặc video webcam. Quá trình xử lý ảnh sẽ được cắt thành khung ảnh tĩnh. Các ảnh này tiếp tục được xử lý đưa về ảnh xám, nhận dạng và tách khuôn mặt khỏi ảnh bằng phương pháp Haar-Cascade của OpenCV. Qua giai đoạn xử lý, hình ảnh khuôn mặt trả về ma trận có tọa độ gồm chiều dài, chiều rộng và chiều cao, sau đó giảm độ phân giải của ảnh xuống phù hợp với mô hình của mạng CNN và chuyển đổi ảnh xám sang ảnh nhị phân. Tiếp theo, sẽ được đưa vào mạng CNN để tiến hành so sánh với các dữ liệu đã được huấn luyện trước và dự đoán cảm xúc trong hình ảnh đó. Cuối cùng, mô hình xuất ra hình ảnh đã được gắn nhãn cảm xúc.

2. Tập dữ liệu – Dataset:

Tập dữ liệu được sử dụng trong đề tài được các thành viên thu thập được. Đối với mỗi hình ảnh dùng cho quá trình kiểm tra, sử dụng phân loại Cascade dựa trên tính năng Haar của OpenCV, tất cả dữ liệu được xử lý trước. Chỉ có phần hình vuông có khuôn mặt khi nhận dạng được giữ lại và phần hình vuông đôi mắt, được thay đổi kích thước và được chuyển đổi thành một mảng có giá trị đen trắng.



Hình 5: Một số hình ảnh trong tập dataset

3. Huấn luyện:

Trong tập dataset được phân loại, sắp xếp theo mắt mở và nhắm mắt. Tập dữ liệu tập hợp hai trạng thái của mắt có kích thước. Dữ liệu được xử lý được đưa vào mô hình CNN tiến hành huấn luyện.

Quá trình huấn luyện chương trình như sau:

a) **Import thư viện vào chương trình:**

```
import cv2
import os
import numpy as np
import glob
import shutil
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, Conv2D, MaxPooling2D
```

b) Tạo dữ liệu huấn luyện cho một mô hình:

Tạo đường dẫn cho tập huấn luyện (train)

```
Datadirectory = ("/kaggle/input/data-project/dataset/train")
Classes = ["Closed_Eyes", "Opened_Eyes" ]
```

Tạo danh sách dữ liệu và chức năng

```
training_Data = []


#Create a function
def create_training_Data():
  for category in Classes:
    path = os.path.join(Datadirectory,category)
    class_num = Classes.index(category) # type 0 1,
    for img in os.listdir(path):
      try:
        img_array = cv2.imread(os.path.join(path, img), cv2.IMREAD_GRAYSCALE)
        backtorgb = cv2.cvtColor(img_array, cv2.COLOR_GRAY2RGB)
        new_array= cv2.resize(backtorgb, (img_size, img_size))
        training_Data.append([new_array, class_num])
      except Exception as e:
        pass
create_training_Data()
print(len(training_Data))
```

Datadirectory: Biến này chứa đường dẫn tới thư mục chứa dữ liệu huấn luyện. Dựa vào đoạn mã, đặt đường dẫn này là "/kaggle/input/data-project/dataset/train".

Classes: Biến này là một danh sách chứa các lớp (classes) của dữ liệu. Trong trường hợp này, có hai lớp: "Closed_Eyes" và "Opened_Eyes".

img_size: Kích thước ảnh đầu vào được chọn để huấn luyện mô hình, và ở đây là 224x224 pixel.

training_Data: Đây là một danh sách (list) rỗng, sẽ được sử dụng để chứa dữ liệu huấn luyện.

Hàm create_training_Data() này sẽ được gọi để tạo dữ liệu huấn luyện bằng cách duyệt qua các lớp và hình ảnh của chúng trong thư mục dữ liệu. Kết quả là danh sách training_Data chứa cặp hình ảnh và nhãn số tương ứng để sử dụng cho việc huấn luyện mô hình.

c) Chuẩn bị dữ liệu huấn luyện cho một mô hình:

```
create_training_Data()
print(len(training_Data))
import random
random.shuffle(training_Data)
[29]
X = []
y = []
for features, label in training_Data:
    X.append(features)
    y.append(label)

# all of X , size of X(a,b) , 3 ways
X = np.array(X).reshape(-1, img_size, img_size, 3)
X.shape
X = X/255
[32]
Y=np.array(y)
Y.shape
```

create_training_Data(): Hàm này tạo dữ liệu huấn luyện từ các hình ảnh trong các thư mục tương ứng với các lớp (classes) "Closed_Eyes" và "Opened_Eyes". Mỗi hình ảnh được chuyển thành một mảng số và được gán một nhãn số tương ứng với

lớp của nó (0 cho "Closed_Eyes" và 1 cho "Opened_Eyes"). Dữ liệu này được thêm vào danh sách `training_Data`.

random.shuffle(training_Data): Dữ liệu huấn luyện sau khi được tạo được trộn ngẫu nhiên để đảm bảo tính ngẫu nhiên trong việc học của mô hình.

Chuẩn bị dữ liệu đầu vào cho mô hình:

- Dữ liệu huấn luyện (‘X’) được lấy ra từ danh sách ‘training_Data’ và chuyển đổi thành một mảng numpy có kích thước phù hợp để phù hợp với mạng nơ-ron, với mỗi mẫu có kích thước ‘img_size x img_size x 3’ (ảnh màu RGB).
- Dữ liệu huấn luyện (‘X’) được chuẩn hóa bằng cách chia cho 255 để đưa giá trị pixel về khoảng [0, 1], điều này thường làm để cải thiện hiệu suất huấn luyện của mô hình.
- Nhãn của dữ liệu (‘Y’) được chuyển đổi thành một mảng numpy.

d) Lưu trữ dữ liệu đã được chuẩn bị thành các tệp:

```
import pickle
pickle_out = open("X.pickle","wb")
pickle.dump(X, pickle_out)
pickle_out.close()
pickle_out = open("y.pickle", "wb")
pickle.dump(y, pickle_out)
pickle_out.close()
pickle_out = open("Y.pickle", "wb")
pickle.dump(Y, pickle_out)
pickle_out.close()
```

pickle_out = open("X.pickle","wb"): Đoạn mã mở một tệp có tên "X.pickle" để lưu trữ dữ liệu đầu vào X dưới dạng pickle. "wb" được sử dụng để mở tệp với quyền ghi dưới dạng binary.

pickle.dump(X, pickle_out): Hàm pickle.dump() được sử dụng để ghi dữ liệu từ biến X vào tệp "X.pickle". Dữ liệu này sẽ được lưu dưới dạng pickle, một định dạng lưu trữ dữ liệu được sử dụng trong Python.

pickle_out.close(): Sau khi dữ liệu đã được ghi vào tệp "X.pickle", tệp được đóng lại để kết thúc quá trình ghi.

Các bước tương tự được thực hiện cho biến y và Y để lưu trữ nhãn và nhãn đã được chuẩn bị tương ứng trong các tệp "y.pickle" và "Y.pickle".

e) Xây dựng và huấn luyện một mô hình mạng nơ-ron:

```
model = Sequential()

model.add(Conv2D(16, (3, 3), input_shape=(224, 224, 3)))
```

```
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))


model.add(Flatten())  # this converts our 3D feature maps to 1D feature vectors

model.add(Dense(512))
model.add(Activation('relu'))

model.add(Dense(1))
model.add(Activation('sigmoid'))


model.compile(loss='binary_crossentropy',
optimizer='RMSprop',
metrics='accuracy')

history = model.fit(X, Y, batch_size=10, epochs=30, verbose=1, validation_split=0.1, shuffle=True)
```

model = Sequential(): Tạo một mô hình mạng nơ-ron tuần tự (sequential model)

Mô hình này bao gồm một chuỗi các lớp:

- Conv2D: Lớp tích chập 2D, được sử dụng để trích xuất đặc trưng từ hình ảnh.
- Activation('relu'): Hàm kích hoạt ReLU (Rectified Linear Unit), thường được sử dụng sau lớp tích chập để tạo sự phi tuyến tính trong mô hình.
- MaxPooling2D: Lớp pooling 2D, được sử dụng để giảm kích thước của đặc trưng và giảm độ phức tạp của mô hình.
- Flatten(): Lớp này được sử dụng để chuyển đổi đặc trưng từ định dạng tensor 3D thành vector 1D, để tiếp tục với các lớp fully connected.

Các lớp Conv2D, Activation('relu'), và MaxPooling2D được xếp chồng lên nhau nhiều lần để tạo một kiến trúc mạng tích chập (CNN) sâu.

Dense(512) và Dense(1): Đây là các lớp fully connected với 512 và 1 đơn vị (neuron) tương ứng. Lớp cuối cùng với 1 đơn vị và hàm kích hoạt sigmoid được sử dụng để dự đoán đầu ra nhị phân (0 hoặc 1), trong đó 0 có thể tương ứng với "Closed_Eyes" và 1 với "Opened_Eyes".

odel.compile(...): Cấu hình mô hình bằng cách chọn hàm mất mát (loss function), trình tối ưu hóa (optimizer), và các metric để đo hiệu suất.

model.fit(...): Huấn luyện mô hình bằng cách đưa vào dữ liệu huấn luyện 'X' và nhãn 'Y'.

Kết quả của quá trình huấn luyện là mô hình đã được đào tạo, và lịch sử huấn luyện được lưu trong biến 'history'

f) **Đánh giá mô hình và vẽ biểu đồ mô hình:**

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = range(30)

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```

Trong đoạn mã này sẽ tiến hành đánh giá mô hình vừa được huấn luyện.

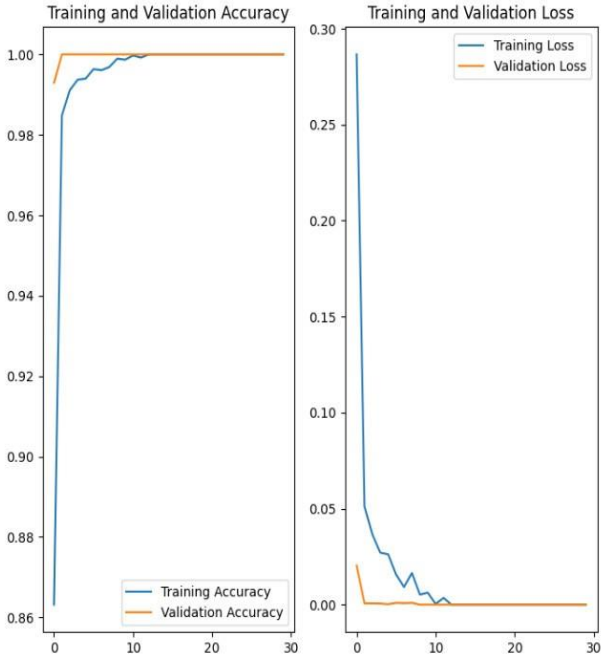
Đánh giá được thực hiện bằng cách in ra sai số 'loss' và độ chính xác 'acccuraly' của mô hình.

Đoạn mã trên sẽ biểu diễn độ chính xác của quá trình huấn luyện và độ chsinh xác trên tập dữ liệu kiểm tra qua mỗi lần eposh.

g) **Lưu mô hình:**

```
model.save('modelEYE_1_better.h5')
from keras.models import load_model
model = load_model('modelEYE_1_better.h5')
```

h) **Kết quả huấn luyện:**



Kết quả lần huấn luyện:

Epochs: 30

Accuracy: 1

Loss: 1.7364e-08

```
Epoch 1/30
383/383 [==============================] - 22s 15ms/step - loss: 0.2866 - accuracy: 0.8632 - val_loss: 0.0204 - val_accuracy: 0.9929
Epoch 2/30
383/383 [==============================] - 4s 11ms/step - loss: 0.0512 - accuracy: 0.9848 - val_loss: 7.0051e-04 - val_accuracy: 1.0000
Epoch 3/30
383/383 [==============================] - 4s 11ms/step - loss: 0.0365 - accuracy: 0.9911 - val_loss: 6.6889e-04 - val_accuracy: 1.0000
Epoch 4/30
383/383 [==============================] - 4s 11ms/step - loss: 0.0271 - accuracy: 0.9937 - val_loss: 5.7813e-04 - val_accuracy: 1.0000
Epoch 5/30
383/383 [==============================] - 4s 11ms/step - loss: 0.0262 - accuracy: 0.9940 - val_loss: 2.4313e-04 - val_accuracy: 1.0000
Epoch 6/30
383/383 [==============================] - 4s 11ms/step - loss: 0.0156 - accuracy: 0.9963 - val_loss: 9.5901e-04 - val_accuracy: 1.0000
Epoch 7/30
383/383 [==============================] - 5s 12ms/step - loss: 0.0092 - accuracy: 0.9961 - val_loss: 7.8192e-04 - val_accuracy: 1.0000
Epoch 8/30
383/383 [==============================] - 4s 11ms/step - loss: 0.0165 - accuracy: 0.9969 - val_loss: 9.5066e-04 - val_accuracy: 1.0000
Epoch 9/30
383/383 [==============================] - 4s 11ms/step - loss: 0.0053 - accuracy: 0.9990 - val_loss: 1.7267e-06 - val_accuracy: 1.0000
Epoch 10/30
383/383 [==============================] - 4s 11ms/step - loss: 0.0063 - accuracy: 0.9987 - val_loss: 1.0420e-06 - val_accuracy: 1.0000
Epoch 11/30
383/383 [==============================] - 4s 11ms/step - loss: 3.3275e-04 - accuracy: 0.9997 - val_loss: 2.4762e-07 - val_accuracy: 1.0000
Epoch 12/30
383/383 [==============================] - 4s 11ms/step - loss: 0.0035 - accuracy: 0.9992 - val_loss: 1.3509e-06 - val_accuracy: 1.0000
Epoch 13/30
383/383 [==============================] - 4s 11ms/step - loss: 6.3399e-07 - accuracy: 1.0000 - val_loss: 2.7729e-07 - val_accuracy: 1.0000
Epoch 14/30
383/383 [==============================] - 4s 11ms/step - loss: 2.2807e-07 - accuracy: 1.0000 - val_loss: 1.4856e-07 - val_accuracy: 1.0000
Epoch 15/30
383/383 [==============================] - 4s 11ms/step - loss: 1.3882e-07 - accuracy: 1.0000 - val_loss: 9.7528e-08 - val_accuracy: 1.0000
Epoch 16/30
383/383 [==============================] - 4s 11ms/step - loss: 9.8847e-08 - accuracy: 1.0000 - val_loss: 7.2043e-08 - val_accuracy: 1.0000
Epoch 17/30
383/383 [==============================] - 4s 11ms/step - loss: 7.6151e-08 - accuracy: 1.0000 - val_loss: 5.6752e-08 - val_accuracy: 1.0000
Epoch 18/30
383/383 [==============================] - 4s 11ms/step - loss: 6.1719e-08 - accuracy: 1.0000 - val_loss: 4.7020e-08 - val_accuracy: 1.0000
Epoch 19/30
383/383 [==============================] - 4s 11ms/step - loss: 5.1558e-08 - accuracy: 1.0000 - val_loss: 3.9814e-08 - val_accuracy: 1.0000
Epoch 20/30
383/383 [==============================] - 4s 11ms/step - loss: 4.4107e-08 - accuracy: 1.0000 - val_loss: 3.4162e-08 - val_accuracy: 1.0000
Epoch 21/30
383/383 [==============================] - 5s 12ms/step - loss: 3.8513e-08 - accuracy: 1.0000 - val_loss: 3.0169e-08 - val_accuracy: 1.0000
Epoch 22/30
383/383 [==============================] - 4s 11ms/step - loss: 3.4080e-08 - accuracy: 1.0000 - val_loss: 2.6688e-08 - val_accuracy: 1.0000
Epoch 23/30
383/383 [==============================] - 4s 11ms/step - loss: 3.0559e-08 - accuracy: 1.0000 - val_loss: 2.3939e-08 - val_accuracy: 1.0000
Epoch 24/30
383/383 [==============================] - 4s 11ms/step - loss: 2.7653e-08 - accuracy: 1.0000 - val_loss: 2.1688e-08 - val_accuracy: 1.0000
Epoch 25/30
383/383 [==============================] - 4s 11ms/step - loss: 2.5277e-08 - accuracy: 1.0000 - val_loss: 1.9944e-08 - val_accuracy: 1.0000
Epoch 26/30
383/383 [==============================] - 4s 11ms/step - loss: 2.3175e-08 - accuracy: 1.0000 - val_loss: 1.8318e-08 - val_accuracy: 1.0000
Epoch 27/30
383/383 [==============================] - 4s 11ms/step - loss: 2.1419e-08 - accuracy: 1.0000 - val_loss: 1.6948e-08 - val_accuracy: 1.0000
Epoch 28/30
383/383 [==============================] - 4s 11ms/step - loss: 1.9885e-08 - accuracy: 1.0000 - val_loss: 1.5745e-08 - val_accuracy: 1.0000
Epoch 29/30
383/383 [==============================] - 4s 11ms/step - loss: 1.8528e-08 - accuracy: 1.0000 - val_loss: 1.4686e-08 - val_accuracy: 1.0000
Epoch 30/30
383/383 [==============================] - 4s 11ms/step - loss: 1.7364e-08 - accuracy: 1.0000 - val_loss: 1.3762e-08 - val_accuracy: 1.0000
```

4. Tiến hành hoạt động:

Sau khi huấn luyện mô hình trí tuệ nhân tạo CNN sẽ tiến hành xây dựng một giao diện đơn giản cho phép người dùng nhận diện trình trạng buồn ngủ qua camera.

- Import thư viện và models:

```
#import các thư viện và model
from keras.models import load_model
model = load_model("modelEYE_1_better.h5")
import cv2
import numpy as np
```

  - Dòng đầu tiên nhập thư viện Keras để tải mô hình mạng nơ-ron đã đào tạo.
  - Dòng thứ hai nạp mô hình từ tệp tin "modelEYE_1_better.h5" sử dụng hàm load_model.
  - Các dòng tiếp theo import thư viện OpenCV (cv2) và numpy.
- Khởi tạo Cascade Classifier cho việc phát hiện khuôn mặt:

```
path = "haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')
```

  - Dòng đầu tiên khai báo đường dẫn đến tệp tin XML chứa thông tin về việc phát hiện khuôn mặt.
  - Dòng thứ hai khởi tạo một đối tượng CascadeClassifier để phát hiện khuôn mặt bằng cách sử dụng tệp tin XML đã được cung cấp sẵn trong OpenCV
- Mở kết nối camera:

```
cap = cv2.VideoCapture(0)
#kiểm tra webcam
if not cap.isOpened():
  cap = cv2.VideoCapture(0)
if not cap.isOpened():
  raise IOError("Cannot open webcam")
```

  - Dòng đầu tiên mở kết nối với webcam, số 0 thường tượng trưng cho webcam mặc định của máy tính.
  - Dòng thứ ba kiểm tra xem kết nối với webcam có mở được không. Nếu không, nó thử mở lại một lần nữa.
  - Dòng thứ năm là một raise exception, nếu webcam vẫn không mở được sau khi đã thử lại, nó sẽ nâng một lỗi và dừng chương trình.
- Khai báo biến và bắt đầu vòng lặp chính:

```
countDis = 0
countClose = 0
while True:
    alarmThreshold_Sleep = 5
    alarmThreshold_Distraction = 20
    ret,frame = cap.read()
    frame = cv2.flip(frame, 1)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    print(faceCascade.empty())
    faces = faceCascade.detectMultiScale(gray,1.1,4)
```

  - Các biến **countDis** và **countClose** được khởi tạo để theo dõi thời gian liên tục khi không có khuôn mặt được phát hiện và khi mắt được dự đoán là đóng.
  - **alarmThreshold_Sleep** và **alarmThreshold_Distraction** là ngưỡng để xác định khi nào nên cảnh báo giấc ngủ và mất tập trung.

- Phát hiện sự mất tập trung (Distraction):

```
if len(faces) == 0:
    status = 'Tap trung'
    countDis +=1
    if countDis >= alarmThreshold_Distraction:
        cv2.putText(frame, "phat hien su mat tap trung!",
                    (100, 100), cv2.FONT_HERSHEY_SIMPLEX,
                    1.0,(0, 0, 255),
                    lineType=cv2.LINE_AA)
        cv2.imshow('Distraction', frame)
```

  - kiểm tra nếu không có khuôn mặt nào, biến status được gán là "**Tap trung**" và biến **countDis** tăng lên.
  - Nếu countDis vượt qua ngưỡng **alarmThreshold_Distraction**, một thông báo "**Phát hiện sự mất tập trung**" được hiển thị trên video.
- Phát hiện khuôn mặt và mắt:

```
else:
    countDis =0
    for(x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

        eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades
+'haarcascade_eye_tree_eyeglasses.xml')
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        eyes = eye_cascade.detectMultiScale(gray,1.1,4)
        for x,y,w,h in eyes:
            roi_gray = gray[y:y+h, x:x+w]
            roi_color = frame[y:y+h, x:x+w]
            cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 1)
            eyess = eye_cascade.detectMultiScale(roi_gray)
            if len(eyess) == 0:
                print("eyes are not detected")
                status = "Closed"
                countClose +=1
```

  - Nếu có khuôn mặt được phát hiện, vòng lặp for duyệt qua từng khuôn mặt và vẽ hình chữ nhật xung quanh chúng.
  - Tiếp theo, code phát hiện mắt bên trong khuôn mặt. Nếu mắt không được phát hiện (đóng), biến status được gán là "**Closed**" và biến **countClose** tăng lên.

- Dự đoán trạng thái mắt và hiển thị thông báo:

```
else:
    for (ex, ey, ew, eh) in eyess:
        eyes_roi = roi_color[ey:ey+eh, ex:ex + ew]
        final_image = cv2.resize(eyes_roi, (224,224))
        final_image = np.expand_dims (final_image, axis =0) # need
fourth dimension
        final_image = final_image/255.0
        Predictions = model.predict(final_image)

    if (Predictions>0):
        status = "Opened"
        countClose = 0

if countClose > alarmThreshold_Sleep:
    cv2.putText(frame, "da phat hien giac ngu", (100, 100),
cv2.FONT_HERSHEY_SIMPLEX, 1.0, (0, 0, 255),
```

```
    lineType=cv2.LINE_AA)
    cv2.imshow('Sleep Detection', frame)
```

- Code lấy mẫu mắt, thay đổi kích thước và chuẩn hóa chúng để chuẩn bị cho việc dự đoán.
- Sử dụng mô hình đã nạp, code dự đoán trạng thái mắt là "**Opened**" hoặc "**Closed**" dựa trên hình ảnh mắt đã chuẩn bị.
- Nếu mắt được dự đoán là "**Opened**", biến status được gán lại là "**Opened**" và biến countClose được đặt lại về 0.
- Nếu countClose vượt qua ngưỡng **alarmThreshold_Sleep**, một thông báo "**Phát hiện giấc ngủ**" được hiển thị trên video.

- Hiển thị kết quả và thoát khỏi chương trình:

```
font = cv2.FONT_HERSHEY_SIMPLEX
    # sử dụng thư viện putTest() để chèn văn bản vào video
    cv2.putText(frame,status,(100, 100),font,3,(0, 255,
0),2,cv2.LINE_AA)
    cv2.imshow('Drowsiness Detection Tutorial', frame)
    if cv2.waitKey(2) & 0xFF== ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

- Code sử dụng OpenCV để thêm văn bản trạng thái lên video.
- Cuối cùng, code kiểm tra nút nhấn phím và nếu người dùng nhấn "q", vòng lặp chính sẽ kết thúc, và kết nối với webcam sẽ được giải phóng (release) và cửa sổ hiển thị sẽ đóng lại.

CHƯƠNG III: PHẦN KẾT LUẬN

1. Về lý thuyết:

- Đã áp dụng được những gì đã học để thiết kế ra một hệ thống nhận diện sử dụng mạng CNN có thể nhận diện được trạng thái buồn ngủ của tài xế từ ảnh có sẵn hoặc nhận diện realtime.
- Xây dựng được giao diện trực quan, dễ sử dụng cho người sử dụng.

2. Ưu điểm, nhược điểm:

❖ **Ưu điểm:**

- Nhận dạng được ảnh, có sẵn hoặc sử dụng realtime bằng webcam.
- Tốc độ xử lý nhanh.
- Nhận diện được nhiều gương mặt cùng lúc.

❖ **Nhược điểm:**

- Còn hạn chế trong việc nhận diện tình trạng, xảy ra sự nhầm lẫn do vài biểu cảm,biểu hiện của khuôn mặt không rõ ràng.
- Nguồn ảnh đầu vào bị nhiễu hoặc thay đổi góc cạnh của gương mặt.

3. Hướng phát triển:

- Tăng số lượng bộ data về các biểu cảm,biểu hiện phức tạp
- Nâng cao độ chính xác khi nhận diện
- Có thể nhận diện ở các môi trường khác nhau.
- Có thể triển khai áp dụng vào thực tế nhằm đánh giá sự hài lòng của người sử dụng trong giám sát trên hệ thống giao thông, công ty giám sát nhân viên lái xe, cảnh báo tài xế khi lái xe trên đường,...

TÀI LIỆU THAM KHẢO

[1]. Docs, OpenCV. “Face Detection Using Haar Cascades.”, OpenCV: Face Detection Using Haar Cascades, 4 Aug. 2017.

[2]. Krizhevsky, Alex. "ImageNet Classification with Deep Convolutional Neural Networks". Retrieved 17 November, 2013.

[3]. Tìm hiểu về mạng tích chập CNN, https://viblo.asia/p/deep-learning-tim-hieu-ve-mang-tich-chap-cnn-maGK73bOKj2, truy cập 12/06/2022

[4]. Giới thiệu Python Tkinter, https://viettuts.vn/python-tkinter, truy cập 12/06/2022

[5]. Tổng quan về mạng CNN, https://nhdp.net/blog/2018/11/tong-quan-don-gian-ve-mang-no-ron-tich-chap-convolutional-neural-networks/, truy cập 08/06/2022