PHÁT HIỆN NGỦ GẬT KHI LÁI XE SỬ DỤNG MẠNG TÍCH CHẬP

Hồng Tiến Hào – 19133022
Instructor Nguyễn Xuân Sâm
Khoa công nghệ thông tin
Đại học sư phạm kỹ thuật thành phố Hồ Chí Minh
Chủ Nhật, 12 tháng 12, năm 2022

TỔNG QUAN

Tài xế ngủ gật đã trở thành một trong những nguyên nhân hàng đầu gây ra tai nạn xe hơi trong những năm gần đây dẫn đến thương tích nghiêm trọng, tử vong và tổn thất đáng kể về tài chính. Với thống kê trên, hệ thống phát hiện người lái xe có dấu hiệu buồn ngủ và cảnh báo trước khi xảy ra va chạm là vô cùng cần thiết. Biểu hiện hành vi, sinh lý và các phương tiện được các nhà nghiên cứu đánh giá và xem xét để đưa ra biện pháp. Từ những nghiên cứu trên, ta sẽ mô hình hóa hệ thống hiện tại và khắc phục các vấn đề gặp phải để để tạo ra một hệ thống đáng tin cậy.

Từ Khóa: Ngủ gật, Mạng tích chập, Xử lý ảnh

1. GIỚI THIỆU

Theo thống kê hiện nay, mỗi năm người chết khi tham gia giao thông là 1.3 triệu và 50 triệu người bị thương tậc. Trước vô lăng nếu một người ngủ gật, chiếc xe sẽ mất kiểm soát và đâm vào phương tiện khác hoặc một vật thể đứng yên. Mức độ ngủ gật của người lái xe nên được giám để ngăn chặn những tai nạn. Các công cụ sau đây đã được dùng để đo lường sự buồn ngủ.

(1) Các biện pháp dựa trên phương tiện: Độ lệch so với làn đường, xoay vô lăng, áp suất gia tốc bàn đạp và các yếu tố khác được theo dõi liên tục, và bất kỳ thay đổi nào vượt quá mức tín hiệu được xác định trước đối với người lái xe bình thường.

(2) Dựa trên hành vi: một máy ảnh theo dõi hành động của tài xế, như là chớp mắt, co giật mắt, tư thế đầu và ngáp, và nếu có bất kỳ dấu hiệu buồn ngủ nào được quan sát thì tài xế sẽ được cảnh báo.

(3) Dựa trên sinh lý: nhiều nghiên cứu đã xem xét mối quan hệ giữa các tín hiệu sinh lý (ECC, EMG, EOG và EEG) và sự buồn ngủ của người lái xe. Một trong những đặc điểm quan trọng để phát hiện buồn ngủ.

Mục đích: của nghiên cứu này là phát hiện và cảnh báo người đó khi mắt không được mở trong một thời gian cụ thể. Khi phát hiện ngủ gật, hệ thống này sẽ thông báo cho trình điều khiển biết.

2. PHƯƠNG PHÁP

Người lái xe kiệt sức và ngủ gật là những nguyên nhân gây ra các loại tai nạn ô tô khác nhau. Trong lĩnh vực hệ thống phòng ngừa tai nạn, để thiết kế và duy trì công nghệ mà có thể phát hiện hiệu quả hoặc tránh ngủ gật ở vô lăng và cảnh báo cho tài xế trước va chạm là một thử thách lớn. Ta sẽ sử dụng OpenCV để chụp ảnh từ một webcam và những hình ảnh này được cung cấp liên tục cho một thuật toán học sâu có thể cho biết liệu mắt của ai đó đang nhắm hoặc mở. Trong trường hợp này, ta sẽ đang tìm kiếm khuôn mặt của người đó và đôi mắt của người lái.

Bước 1: Ảnh sẽ lấy như là một đầu vào từ máy ảnh.

Ta sẽ sử dụng máy ảnh để quay gương mặt tài xế để đảm bảo tính liên tục. Với mỗi khung hình (frame) thu được sẽ được xem như là một ảnh chụp làm đầu

vào cho mô hình xử lý. Chúng ta sử dụng cv2 phương pháp được đưa ra bởi OpenCV.

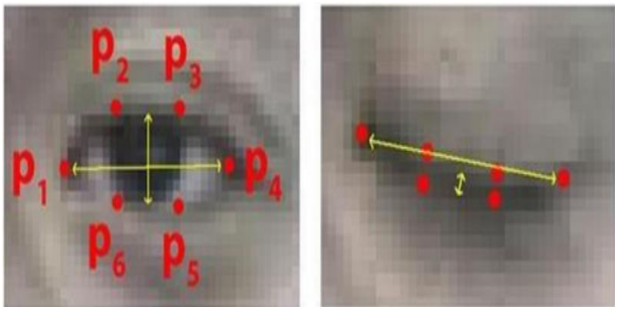
Bước 2: Tạo khoanh vùng (Region of Interest hay ROI) bằng cách xác định khuôn mặt trong ảnh.

Để phân khuôn mặt trong ảnh đã chụp thành nhiều phần, trước tiên ta chuyển đổi nó thành thang độ xám vì thuật toán phát hiện đối tượng OpenCV chỉ chấp nhận hình ảnh theo thang độ xám làm đầu vào. Để phát hiện các đối tượng, chúng ta không cần chi tiết màu sắc. Chúng ta sử dụng bộ phân loại theo tầng Haar để phát hiện khuôn mặt.

Bước 3: Sử dụng ROI để tìm mắt và đưa chúng vào bộ phân loại.

Trình phân loại Cascade được sử dụng ở mắt trái và mắt phải. Ta chỉ trích xuất các chi tiết của đôi mắt từ hình ảnh đã chụp. Điều này có thể được thực hiện bằng cách trước tiên xóa khoanh vùng của mắt và sau đó xóa hình ảnh con mắt khỏi ảnh.

Thông tin này được đưa cho CNN, nơi quyết định xem có nhắm mắt hay không.



Ảnh 1: Tính toán kích thước mắt đóng và mở

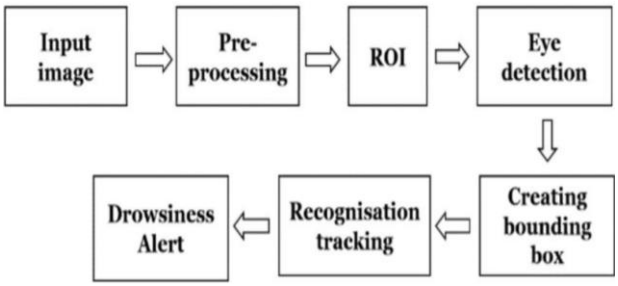
Bước 4: Bộ phân loại sẽ xác định mắt có mở hay không

Trạng thái mắt được dự đoán bằng cách sử dụng bộ phân loại CNN để đưa hình ảnh vào mô hình do mô hình yêu cầu các phép đo phù hợp để bắt đầu. Chúng ta sẽ bắt đầu bằng cách chuyển đổi hình ảnh màu thành thang độ xám. Sau đó, vì mô hình được huấn luyên trên hình ảnh có độ phân giải 24 x 24 pixel, Chúng ta phải thay đổi kích thước hình ảnh thành 24 x 24 pixel.

Bước 5: Tính điểm

Về cơ bản, điểm số là một con số mà chúng ta sẽ sử dụng để tính xem cá nhân đó đã nhắm mắt trong bao lâu. Do đó, nếu nhắm cả hai mắt, chúng ta sẽ bắt đầu tăng điểm, nhưng nếu mở cả hai mắt, chúng ta sẽ giảm điểm.

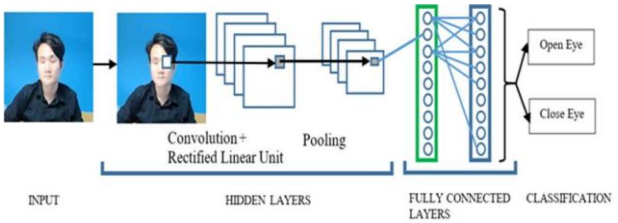
Ta sẽ xác định một tiêu chí, nếu điểm vượt quá 15, điều đó cho thấy rằng mắt của người đó đã nhắm trong một khoảng thời gian dài. Sau đó, báo động bật lên. Ta có sơ đồ của các bước



Ảnh 2: Sơ đồ khối để phát hiện ngủ gật bằng CNN

3. KIẾN TRÚC MÔ HÌNH

Mô hình chúng ta sử dụng được xây dựng với Keras bằng cách sử dụng các mạng thần kinh tích chập (Convolutional Neural Networks hay CNN. Một mạng lưới thần kinh tích chập là một loại mạng thần kinh sâu đặc biệt hoạt động rất tốt cho mục đích phân loại hình ảnh. CNN về cơ bản bao gồm một lớp đầu vào, một lớp đầu ra và một lớp ẩn có thể có nhiều lớp. Một hoạt động tích chập được thực hiện trên các lớp này bằng cách sử dụng bộ lọc thực hiện phép nhân ma trận 2D trên lớp và bộ lọc.



Ảnh 3: CNN xử lý ảnh đầu vào từ máy ảnh

Kiến trúc mô hình CNN bao gồm các lớp sau:

(1) Lớp tích chập: 32 nodes, kích thước Kernel 3.

(2) Lớp tích chập: 64 nodes, kích thước Kernel 3.

(3) Lớp tích chập: 128 nodes, kích thước Kernel 3.

(4) Lớp tích chập: 64 nodes, kích thước Kernel 3.

Lớp cuối cùng cũng là một lớp được kết nối đầy đủ với 2 node. Hàm kích hoạt Relu được sử dụng trong tất cả các lớp ngoại trừ lớp đầu ra (sử dụng SoftMax).

4. TẬP DỮ LIỆU

Bộ dữ liệu được sử dụng cho mô hình này được tạo bởi Serena Raju, được chia sẻ công khai trên Kaggle. Để tạo bộ dữ liệu, Serena Raju đã viết một tập lệnh ghi lại mắt từ máy ảnh và lưu trữ trong đĩa cục bộ. Họ đã tách chúng vào các nhãn tương ứng mắt của họ 'mở' hoặc 'đóng'. Dữ liệu được làm sạch thủ công bằng cách loại bỏ các hình ảnh không mong muốn không cần thiết để xây dựng mô hình. Dữ liệu bao gồm khoảng 7000 hình ảnh của mắt mọi người trong các điều kiện ánh sáng khác nhau.

Ảnh 4: Tập dữ liệu huấn luyện cho mô hình

Chúng ta sẽ sử dụng bộ dữ liệu này để huấn luyện với kiến trúc trên.

4. YÊU CẦU

Yêu cầu cho thực nghiệm này, ta sẽ sử dụng ngôn ngữ Python. Bên cạnh đó là một webcam thông qua đó chúng ta sẽ quay trạng thái người lái xe. Ta cần cài đặt Python (phiên bản 3.9), và thư viện cần cần thiết sau:

1. *OpenCV*: phát hiện mặt và mắt.

2. *TensorFlow*: Keras sử dụng TensorFlow làm backend để xây dựng mô hình phân loại.

3. *Keras*: để xây dựng mô hình phân loại.

4. *Pygame*: để phát âm thanh cảnh báo.

5. *flask*: một framework xây dưng web, ta dùng để áp dụng thử nghiệm mô hình.

6. *flask_cors*: module tích hợp của flask

7. *numpy*: xử lý ảnh dưới dạng dữ liệu số.

8. *matplotlib*: thư viện cần cho trực quan hóa biểu đồ, để kiểm tra hiệu năng của mô hình trong quá trình huấn luyện.

9. *pyrebase*: Thư viện cần để kết nối với firebase của Google Cloud dùng lưu dữ liệu realtime trong quá trình thử nghiệm.

5. CẤU TRÚC DỰ ÁN

Ta sẽ xây dựng một trang web để thử nghiệm mô hình, dự án ta sẽ có cấu trúc như sau: Một mô hình phân loại, trang web sử dụng mô hình để phát hiện ngủ gật, một cơ sở dữ liệu để phân tích dữ liệu theot thời gian thực.

5.1. Mô Hình Mạng Tích Chập (CNN)

Huấn luyện mô hình:

(1) Thư mục data: chứa tập dữ liệu huấn luyện mô hình và tập kiểm tra đánh giá hiệu năng mô hình.
(2) Xây dựng mô hình: ta sẽ xây dựng mô hình mạng tích chập để nhận diện (trạng mắt, ngáp) dựa trên dữ liệu và kiến trúc mô hình kể trên.
(3) Sau khi huấn luyện xong, mô hình đã được huấn luyện trên các mạng thần kinh tích chập sẽ được lưu lại để dùng cho mục đích phân tích.

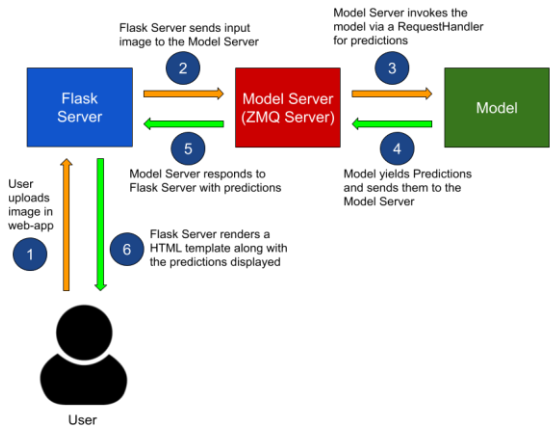
Sau khi có mô hình phân loại trạng thái mắt ta cần một chương trình nhận diện gương mặt cũng như vị trí mắt để áp dụng mô hình trên. Vì đây là một ứng dụng thời gian thực, nên ta cần công cụ phân loại nhẹ có tốc độ nhận diện cao để đáp ứng về mặt thời gian, cũng như sự chính xác để đảm bảo an toàn. Để đảm bảo các yếu tố trên ta sẽ sử dụng *Haar Cascade* để phát hiện phần mặt và mắt.

5.2. Web

Sau khi có đầy đủ chương trình cho nhận diện, ta sẽ đưa mô hình lên web để ứng dụng, thử nghiệm. Bên cạnh xây dựng một website đơn giản bằng Flask cho việc thử nghiệm, Ta sẽ cần thêm:

(1) Một chương trình thao tác với máy quay để lấy từng khung hình để đưa vào mô hình nhận diện.
(2) Một chương trình sẽ nhận ảnh liên tục từ máy quay đưa vào mô hình để dự đoán sự đóng mở của mắt.
(3) Một âm thanh sẽ được kích hoạt khi người đó ngủ gật.

Và đây là cách trang web hoạt động:



Ảnh 5: Cách trang web hoạt động

Trong dự án của chúng ta, mô hình sẽ được tải khi server được kích hoạt. Sau đó, server sẽ bắt đầu lắng nghe các yêu cầu từ client:

(1) User sẽ gửi ảnh liên tục qua web-app thông qua camera.
(2) Flask Server sẽ hành động như một máy khách và máy chủ mô hình. Nó gửi hình ảnh đầu vào (ở một số dạng được mã hóa) đến máy chủ mô hình.
(3) Máy chủ mô hình gọi mô hình thông qua RequestHandler để dự đoán.
(4) Mô hình mang lại dự đoán và gửi chúng đến máy chủ mô hình.
(5) Máy chủ mô hình phản ứng với Máy chủ Flask với dự đoán.
(6) Máy chủ Flask hiển thị một mẫu HTML cùng với các dự đoán được hiển thị.

5.3. Firebase

Trong quá trình nhận diện, các dữ liệu về điểm số được tính toán dựa trên sự đóng mở mắt sẽ được lưu trên cơ sở dữ liệu để thực hiện phân tích theo thời gian thực. Phần cơ sở dữ liệu ta sẽ chọn là Firebase của Google Cloud.

6. THỰC NGHIỆM

(1) Lấy hình ảnh làm đầu vào từ máy quay thông qua webcam:

Để truy cập webcam, ta sử dụng phương thức được cung cấp bởi OpenCV *cv2.VideoCapture(0)*. Dữ liệu hình ảnh thu được sẽ liên tục, ta dùng *vid.read()* sẽ đọc từng khung hình và ta lưu hình ảnh trong một biến để xử lý.

(2) Phát hiện khuôn mặt trong ảnh và tạo một vùng quan tâm (ROI) để phát hiện khuôn mặt trong ảnh:

Trước tiên chúng ta cần chuyển đổi ảnh thành ảnh xám, bởi vì ta không cần thông tin về màu sắc để phát hiện khuôn mặt. Như đã đề cập ta sẽ sử dụng phân loại Haar Cascade để phát hiện khuôn mặt.

Nó sẽ trả về một mảng phát hiện với tọa độ *x, y* và chiều cao *h*, chiều rộng *w* của hộp ranh giới (boundary box) của đối tượng.

(3) Nhận diện mắt từ ROI và đưa vào bộ phân loại:

Bây giờ, ta cần trích xuất dữ liệu mắt từ ảnh và điều này có thể thực hiện bằng cách trích xuất hộp ranh giới của mặt và sau đó ta có thể trích xuất hình ảnh mắt ra khỏi khung hình. Dữ liệu sẽ được đưa vào trình phân loại CNN của chúng ta, và sẽ dự đoán nếu mắt đang mở hoặc đóng.

(4) Sử dụng CNN để đoán trạng thái mắt:

Để đưa ảnh của vào mô hình, Ta cần thực hiện các bước nhất định vì mô hình cần các kích thước chính xác để bắt đầu. Đầu tiên, Ta chuyển đổi ảnh màu thành ảnh xám. Sau đó, Ta thay đổi kích thước ảnh thành 24x24 pixel (mô hình của ta được đào tạo trên hình ảnh 24x24 pixel). Ta cũng chuẩn hóa dữ liệu của mình để hội tụ tốt hơn (giá trị sẽ từ 0 – 1).

(5) Tính điểm để kiểm tra xem người có ngủ gật:

Về cơ bản, ta sẽ tính giá trị dùng để xác định người đó nhắm mắt hay không. Vì vậy, nếu cả hai mắt

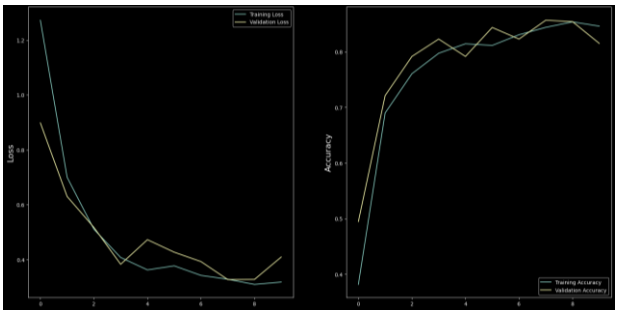
nhắm nghiền, chúng ta sẽ tiếp tục tăng điểm và khi mắt mở, Ta sẽ giảm điểm.

- 1 hoặc cả 2 mắt mở: ta sẽ giảm 1 điểm
- Cả 2 mắt nhắm: ta sẽ tăng 1 điểm

Một ngưỡng được xác định là 15 nếu vượt qua ngưỡng này tức là người này đã ngủ gật. Đây là khi chúng ta phát ra cảnh báo bằng âm thanh.

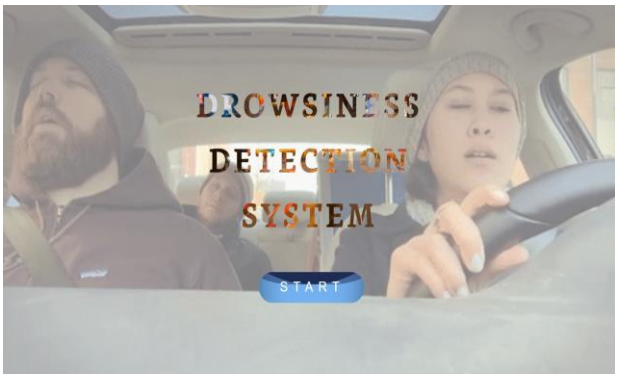
7. KẾT QUẢ

7.1. Kết Quả Huấn Luyện



Ảnh 6: độ chính xác và hàm mất trong quá trình huấn luyện

7.2. Giao Diện Flask Application



Ảnh 7: Trang chủ

Ảnh 8: Hình ảnh thu được từ camera

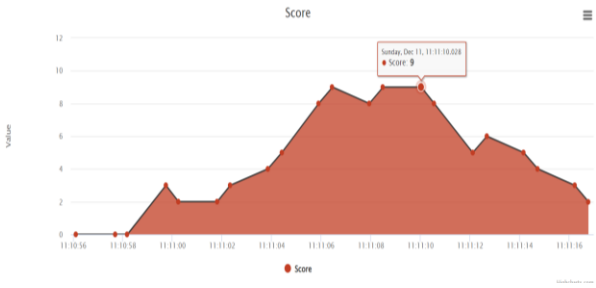
7.3. Kết Quả Áp Dụng

Đầu ra hình ảnh đầu tiên cho thấy trạng thái bình thường của người đó mà không có bất kỳ dấu hiệu ngủ gật nào nên giá trị điểm là 0. Đối với hình ảnh thứ hai, giá trị điểm vượt quá 15 nên hệ thống phát hiện sẽ cảnh báo người đó bằng báo động âm thanh cảnh báo.

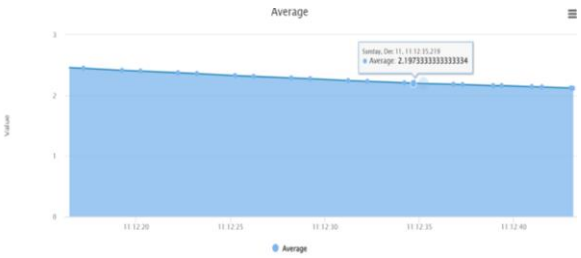
Ảnh 9: Khi nhắm mắt điểm bắt đầu tăng

Ảnh 10: Khi vượt ngưỡng hệ thống bắt đầu báo động

7.4. Dữ Liệu Theo Thời Gian



Ảnh 11: Điểm an toàn



Ảnh 12: Điểm an toàn trung bình

Bảng 1 Chi Tiết Kết Quả Huấn Luyện

Epoch	1	2	3	4	5	6	7	8	**9**	10
Loss	1.2721	0.7	0.5110	0.4080	0.3627	0.3775	0.3426	0.3287	**0.3098**	0.3185
Accuracy	0.3818	0.69	0. 7606	0.7973	0.8145	0.8115	0.8307	0.8440	**0.8538**	0.8461

8. ĐÁNH GIÁ

Mô hình phân loại dựa trên bộ dữ liệu không quá lớn nên khi đổi một số góc độ trước máy quay có thể làm giảm khá năng phát hiện xuống. Thêm một điểm nữa, bộ dữ liệu được huấn luyện là các hình ảnh của người phương tây (đặc điểm mắt to), nên đối với người có vùng mắt nhỏ (thường là người châu á) cũng sẽ làm giảm khả năng phân loại đóng mở mắt của mô hình. Vấn đề này sẽ được khắc phục với bộ dữ liệu lớn hơn, đa dạng các đặc điểm về mắt.

Về chương trình thử nghiệm còn khá chậm thường xử lý không kịp thời các khung hình làm chậm đầu ra (điểm đánh giá, trạng thái mắt). Ta có thể khắc phục bằng cách nâng cấp phần cứng và tối ưu code.

9. KẾT LUẬN

Hệ thống Phát hiện ngủ gật, dựa trên việc nhắm mắt của người lái xe, có thể phân biệt giữa chớp mắt bình thường và nhắm mắt. Đề xuất sẽ giúp ngăn ngừa thương tích do lái xe trong tình trạng buồn ngủ. Sử dụng bộ phân loại theo tầng Haar, OpenCV được sử dụng để phát hiện khuôn mặt và mắt, sau đó là mô hình CNN để dự đoán trạng thái. Một tín hiệu cảnh báo được cung cấp khi nhắm mắt trong một khoảng thời gian dài. Nhắm mắt liên tục được sử dụng để đánh giá mức độ tỉnh táo của người lái xe. Đối với các tác vụ trong tương lai, hệ thống phát hiện này có thể được tạo thành phần cứng với các tính năng nâng cao.

THAM KHẢO

[1] Ann Williamson and Tim Chamberlain. Review of on-road driver fatigue monitoring devices. NSW Injury Risk Management Research Centre, University of New South Wales, July 2013.

[2] C. Hentschel, T. P. Wiradarma, and H. Sack, Fine tuning CNNS with scarce training data-adapting imagenet to art epoch classification, in Proceedings of the IEEE International Conference on Image Processing (ICIP), Phoenix, AZ, USA, September 2016.

[3] K. He, X. Zhang, S. Ren, and J. Sun, Deep residual learning for image recognition, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, June 2016.

[4] ZuopengZhao, LanZhang, Hualin Yan, Yi Xu, and Zhongxin Zhang, Driver Fatigue Detection Based on Convolutional Neural Networks Using EM -CNN, 2020.

[5] Vagdevi K, Improving the Performance of Deep-Learning based Flask App with ZMQ, Retrieved on December 3 from https://cloudxlab.com/blog/improving-the-performance-of-deep-learning-based-flask-app-with-zmq

[6] Data-flair, Driver Drowsiness Detection System with OpenCV & Keras, Retrieved on December 3 from https://data-flair.training/blogs/python-project-driver-drowsiness-detection-system