A
REPORT
ON

Machine Learning Based Intoxication Detection

By

Rishika Kalra- 2021A3PS2651P

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI (Rajasthan)

(November, 2024)

Table of Contents

1. Introduction

Using deep learning technology, a non-invasive drunk detection mechanism based on a camera is developed. Traditionally, alcohol intoxication detection methods comprise blood tests and breath analyzers. These two methods are effective but suffer from some visible shortcomings. Blood tests are accurate, time-consuming, expensive, and invasive, while breath analyzers, being portable and cheap, may have some issues with user resistance and delays in measurement. Thus, in this project, it is proposed that infrared cameras combined with deep learning models acquire thermal images from the face and determine the level of alcohol. It would capture tiny physiological changes, such as changes in skin temperature and even activities within the blood vessels, thus providing an ultra-fast, accurate, and non-invasive method of identification. The main aim is to couple an infrared camera with a mobile application in such a manner that ensures the live detection of alcohol intoxicated people. Giving road safety a very high priority, the project in question could help to ensure watching and identifying drunk drivers without physical contact. By using deep learning to perform an accurate classification, the system has the potential to revolutionize the current methods for detecting alcohol intoxication and ultimately lessen accidents and make the public safer.

Background

Traditionally, alcohol intoxication detection relied on invasive and cumbersome methods such as blood tests and breath analyzers. Although blood tests give accurate results, they become impractical on account of their invasiveness and the time they take for processing. Breath analyzers are non-invasive and portable, but they are often subject to delayed results, user resistance, and inaccuracies. An up-and-coming area of study theorizes that drinking alcohol causes physiological changes - skin temperature or blood vessel patterns - that may be identified non-invasively. Infrared (IR) cameras have captured these subtle changes, and this data combined with deep learning models presents a feasible alternative for accurate real-time alcohol detection.

Motivation

The main reason for commissioning this project is the need to get over the limits of existing alcohol detection methods with faster, more effective, and non-invasive alternatives. Risks posed to the general public by impaired driving continue to be a vital area of public safety interest, with numerous detection schemes being either too slow or too invasive to be applicable in real-life situations. A system using infrared cameras with deep learning to ascertain degrees of intoxication underpinned by the will to reduce road accidents caused by alcohol abuse is very much a motivating premise. Another motivating factor driving the need for this project is to develop a non-contact method that would run from different environments, including darkened ones, and give precise and real-time results without causing distress to or resistance from subjects.

Objective

The objective of this project is to establish a non-invasive camera-based system for the detection of alcohol intoxication by analysing thermal images of the face. The system is going to comprise an infrared camera conjoined by an appropriate mobile application that allows for facial thermal imaging and also allows for its real-time processing via deep learning models, for example, convolutional neural networks (CNNs). The outcome will classify these people into categories of intoxication and will be a quick, reliable, contactless method unlike the older methods. The project basically will contribute to road safety with the practical solution of identifying driving under intoxication in order to bring about a reduction in alcohol-related accidents.

2. Literature review

Different approaches have been investigated in recent years in order to render mechanisms for detecting alcohol non-intrusive; these chiefly rest upon more advanced imaging techniques and machine learning models. These methods seek to indirectly provide solutions to the various shortcomings found in traditional breathalyzers and blood testing kits.

Thermal Imaging with Infrared Cameras

One of the major ways to detect drunk driving involves thermal imaging, employing infrared (IR) cameras to exhibit facial temperature changes and vascular changes with alcohol consumption. Studies have suggested that increased skin temperature and higher rates of perspiration associated with alcohol intake could be visualised with thermal cameras. For intoxication analysis, for example, the **FLIR ONE infrared camera** pairs with mobile applications to permit capture of real-time thermal imagery. The method would allow a non-invasive mode of testing across a range of light conditions, which render it conducive to real-time applications, allowing roadside checks, for instance.

Deep Learning for Image Processing

Extensive research has been done in context to the use of deep learning models, among which Convolutional Neural network based are the most temperamental type, for thermal image processing and classification. CNNs are celebrated for their ability to analyze visual data and extract relevant features for classification. To facilitate processing, down-sample spatial dimensions, and integrate features for final classification using softmax regression, key components of CNNs, such as ReLU non-linearities, max pooling, and fully connected layers, have been employed.Numerous studies have demonstrated that CNN-based models can recognize alcohol intoxication. For example, the **NasnetMobile model** reached high accuracy of **85.10%** for classifying subjects into the **four categories of alcohol consumption (calm, 1 bifocal, 2 bifocals, and 3 bifocals)**; the classification of sober from buzzed using the MobileNet model achieved an accuracy of **74.07%**. The models were specifically trained with thermal images of the volunteers taken before and after controlled alcohol consumption. These results demonstrate that CNNs can be a great tool for processing and classifying thermal images to indicate varying levels of intoxication.

Data Preprocessing and Augmentation Techniques

Various data preprocessing and augmentation techniques are employed to enhance the model's performance. For data augmentation, such techniques applied include the use of rectangular patches.This improves the training dataset through diversity, resulting in better generalization. Other estimation processes during preprocessing include adding **Gaussian noise** and **blur filters** that simulate real world conditions to better help models cope with noisy data. Such processes help to improve classification performance as evidenced by some reports presented.

Local Difference Patterns (LDP)

Another idea that stands independently of CNNs involves the **Local Difference Patterns (LDP)**, which analyzes pixel variations in certain regions of thermal images in connection with intoxication detection. In one study, this approach used LDPs for analysis of the forehead region in order to monitor changes in blood vessel activity due to alcohol consumption. In that study, **85% of intoxication detection was accomplished in a sample of 34 out of 41 participants**, demonstrating that this method was effective even if a sober reference image was not taken. The simplicity and robustness of LDPs make them particularly suited for on-the-spot detection of intoxication, which is crucial in applications such as law enforcement.

Hybrid Embedded-Systems-Based Approach

In some instances, a hybrid manner involving image signal processing techniques and sensor network systems is employed for drunk-driving status detection. It involves choosing features relevant to certain aspects, such as **facial foundation temperature**. Classifications are then performed using machine-learning algorithms, such as the **k-Nearest Neighbor and Neural Networks**. This approach outperforms others with **98% classifiers** which are tested on real-world settings for drunk-driving detection. In addition, since the system can run on embedded devices, it opens up more possibilities for real-life vehicular applications.

Fine-Grained Gait Classification

Beyond the use of facial thermal images, research explores the gait pattern for intoxication detection. In this approach, gait signals collected via mobile phones are first converted into **GAF** images for classification via the **BiCNN method**. It formulates gait intoxication detection as a **fine-grained image classification problem** and exhibits better performance than existing methods on a dataset collected from 121 participants under a controlled drinking protocol. The ability of this method to classify gait patterns without additional hardware and unaffected by illumination variation is a very promising avenue for non-invasive intoxication detection.

3. Datasets used

The first dataset represents an extract from a very big video file(kxr lab dataset), a raw collection of videos. There are 15 individual video clips featuring non-intoxicated subjects or participants. The selected clips are of each particular video for the solo focused analysis of a person's behavior and physiological characteristics, which do not include or require the presence of alcohol.
The video input is from a single larger video and segmented according to each participant. A total of 20 videos with each clip of one participant. All the videos have been of subjects in non-intoxicated states as a control group to be compared eventually against intoxicated sets.
The raw videos have no preprocessing or changes whatsoever.
The second dataset consists of a dukan dataset of 65 videos of different participants (not the same as the last dataset), each filmed under (A-F) different levels of intoxication, ranging from mild to severe alcohol consumption.
The videos showcase different participants exposed to different levels of alcohol consumption (a total of 65 videos) in different stages of intoxication(A-F).
This dataset incorporates new participants different from the non-intoxicated dataset. The videos are provided in raw form without preprocessing.
The third dataset consists of a wide variety of larger files recorded when the Oasis 2024, which were cut into 459 different files using the ffmpeg module.
The videos take place because of different participants in mostly a non- intoxicated way in the case of Oasis 2024.
459 videos.The new entrants to the experiment differentiate from the kxr lab dataset and the dukan dataset participants regarding behaviours and physiological response diversity.The raw videos are provided without preprocessing.

4. Experiments and Results

In this project, we have implemented a video processing application in OpenCV, which is capable of face detection as well as face cropping in a video file.

The detailed steps include but are not limited to:

1)Video Input: In our case, the input and the output video paths were specified. The input video would be loaded in, using frame-by-frame processing, with the help of cv2.VideoCapture.

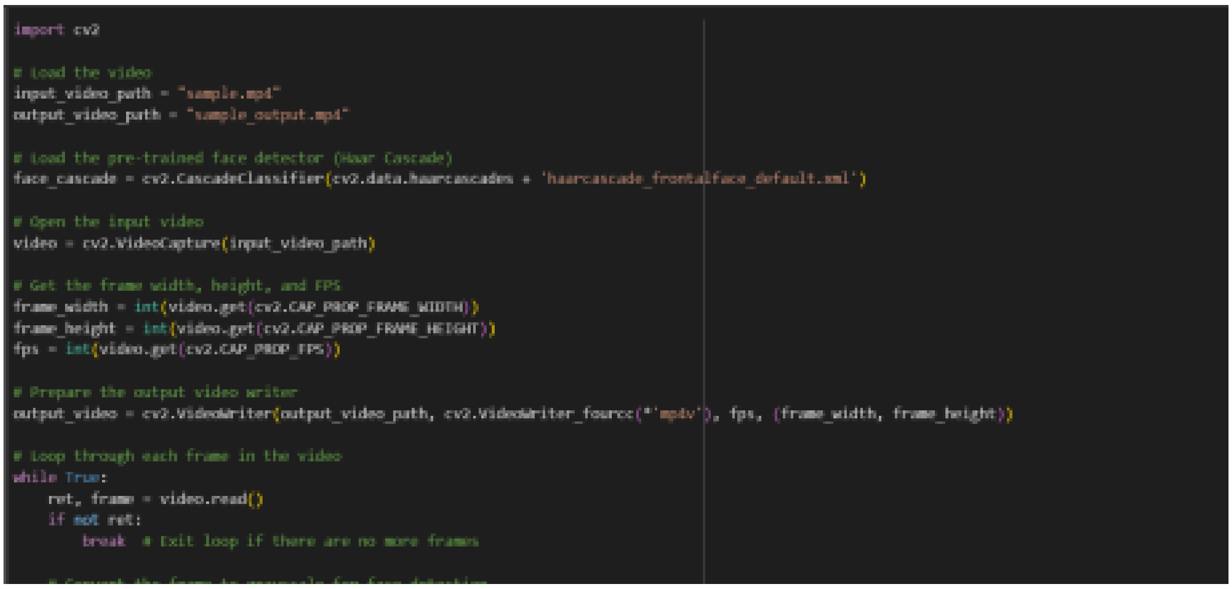
2)Face Detection Model: To accomplish this task, the Haar Cascade face detection model was loaded into the system using CascadeClassifier which is required to detect the face in the images.

3)Video Properties: The video properties such as frame width, frame height, and frames per second were also taken into consideration in order to almost retain the formats of the output with those of the input.

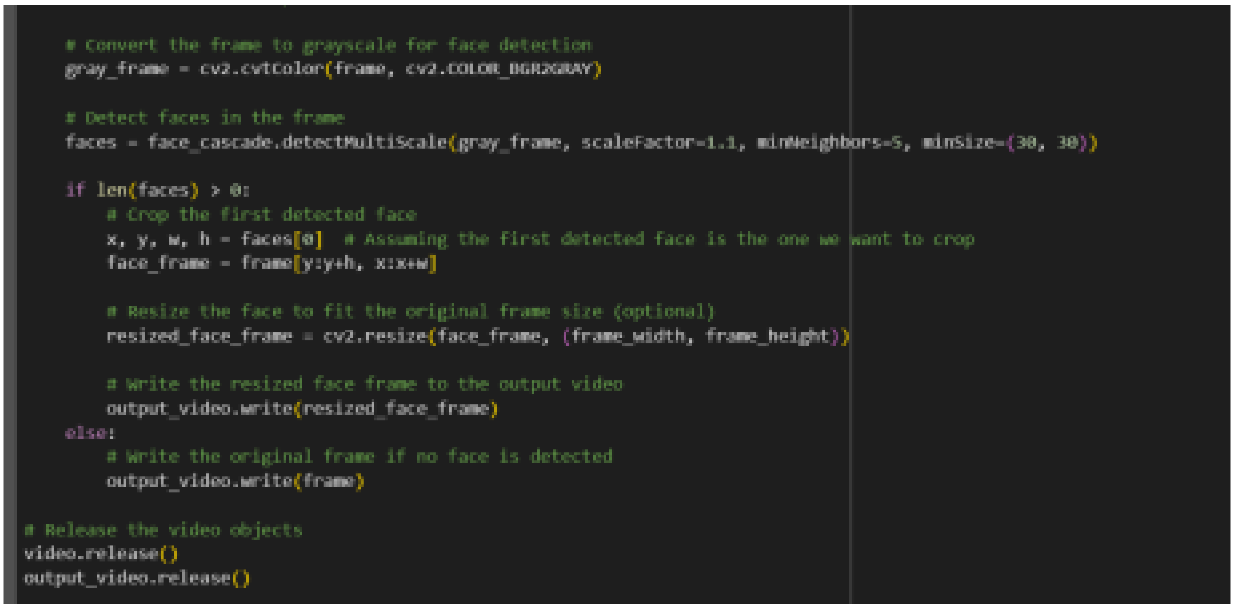
4)Output Video Preparation: The processed frames were stored in files with the help of cv2.VideoWriter which was used to configure the output video writer.

5)Frame control: For the purpose of better face recognition each frame was converted to a grayscale palette. Employing the detectMultiScale technique for face detection, the first detected face was cut out after a quick spotting.

6)Face Cropping: The detected face was rescaled in order to maintain consistency with the size of the original frame during the output video writing process; if face detection was not executed, the original frame was kept.



```python
import cv2

# Load the video
input_video_path = "sample.mp4"
output_video_path = "sample_output.mp4"

# Load the pre-trained face detector (Haar Cascade)
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')

# Open the input video
video = cv2.VideoCapture(input_video_path)

# Get the frame width, height, and FPS
frame_width = int(video.get(cv2.CAP_PROP_FRAME_WIDTH))
frame_height = int(video.get(cv2.CAP_PROP_FRAME_HEIGHT))
fps = int(video.get(cv2.CAP_PROP_FPS))

# Prepare the output video writer
output_video = cv2.VideoWriter(output_video_path, cv2.VideoWriter_fourcc(*'mp4v'), fps, (frame_width, frame_height))

# Loop through each frame in the video
while True:
    ret, frame = video.read()
    if not ret:
        break  # Exit loop if there are no more frames
```

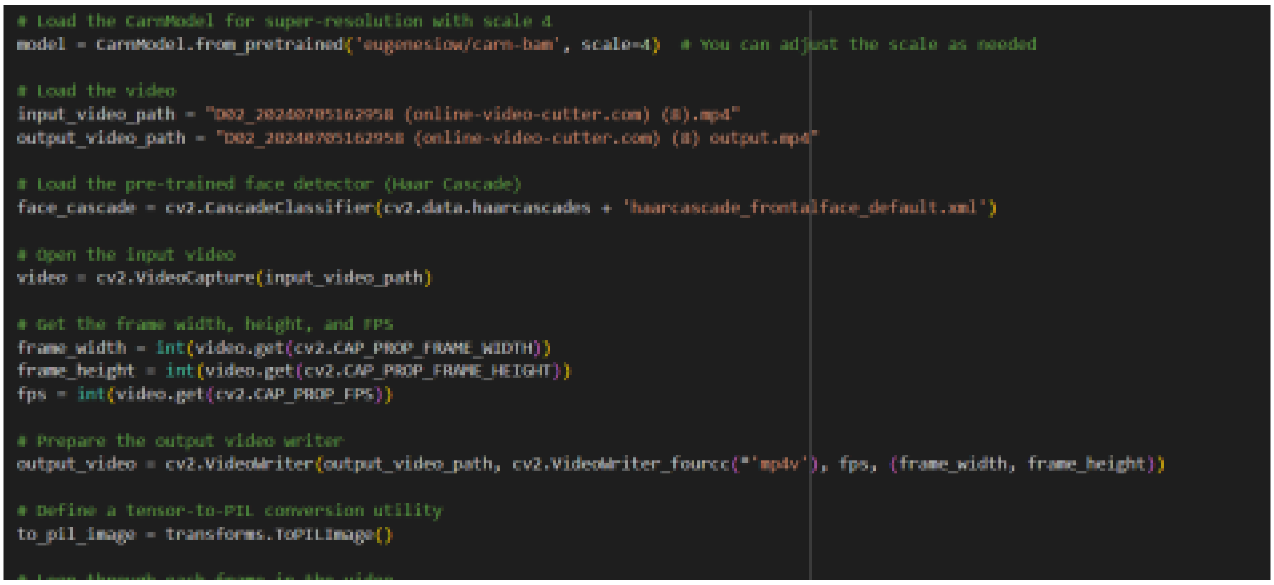


```
        # Convert the frame to grayscale for face detection
        gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        # Detect faces in the frame
        faces = face_cascade.detectMultiScale(gray_frame, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

        if len(faces) > 0:
            # Crop the first detected face
            x, y, w, h = faces[0]  # Assuming the first detected face is the one we want to crop
            face_frame = frame[y:y+h, x:x+w]

            # Resize the face to fit the original frame size (optional)
            resized_face_frame = cv2.resize(face_frame, (frame_width, frame_height))

            # Write the resized face frame to the output video
            output_video.write(resized_face_frame)
        else:
            # Write the original frame if no face is detected
            output_video.write(frame)

    # Release the video objects
    video.release()
    output_video.release()
```

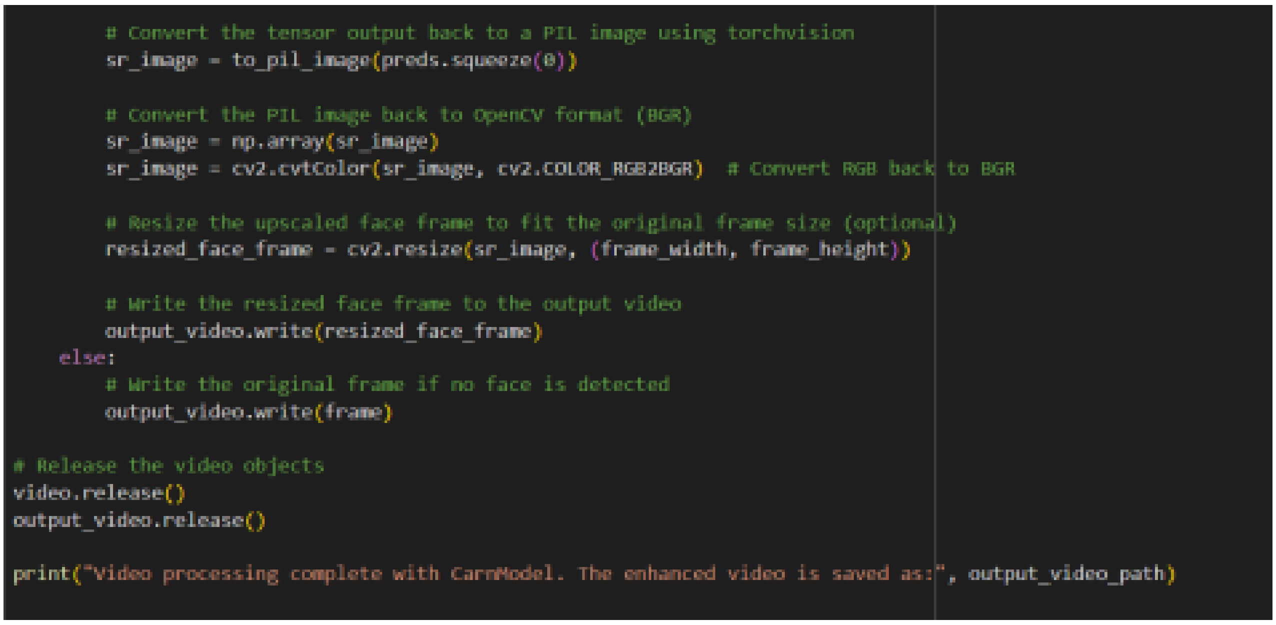
The results of this could have been improved and so we employed a **Super Resolution Model approach**.
The task involved developing a video processing application that detects faces in a video, applies super-resolution enhancement using the **CarnModel**, and saves the enhanced video.

The **Cascading Residual Network (CARN)** represents a significant advancement in single image super-resolution (SISR) technology.The model's primary strength lies in its cascading mechanism, wherein information propagates through multiple interconnected blocks, facilitating optimal feature utilisation and gradient flow. The architecture employs a sophisticated system of residual blocks, each designed to learn the residual mapping between low and high-resolution representations. These blocks comprise **dual 3×3 convolutional layers** integrated with **ReLU activation functions** and local residual connections, establishing a robust foundation for feature extraction. A distinguishing characteristic of CARN is its implementation of multi-scale feature fusion, which synthesises information from various hierarchical levels through concatenation operations, enabling comprehensive capture of both fine-grained details and broader contextual information.From an efficiency perspective, CARN exhibits notable advantages through its relatively compact model size and strategic utilisation of group convolutions, resulting in reduced computational complexity. The model demonstrates exceptional flexibility in handling **multiple upscaling factors (2×, 3×, and 4×)** through efficient pixel shuffle layers, maintaining consistent quality across various scaling requirements.
The steps requisite to performing this task are elaborated below:
1)Load the super resolution Carn Model available on Hugging Face Model Hub which acts to increase image resolution by a factor of 4.
2)Define the input and output locations of the video files so as to help in the video processing.
3)Import a pre-trained Haar Cascade face detection model with the CascadeClassifier in OpenCV to detect the faces in each video frame.
4)Capture the input video file by using cv2.VideoCapture to allow the reading of the video pictures sequentially.
5)Identify the attributes of the video which embrace a frame's width, a height and a number of frames in a single second to make sure the new video does not differ from the first one.
6)Create a video writing device with the cv2.VideoWriter so that the new video which has been altered can be stored.
7)Create a tensor to convert to image in 'PIL' format for use at a later time with the super resolution model by utilizing the application transforms.ToPILImage.

8) Create a loop that will read through each frame of the video, reading pictures in frames to the last frame that remains.
9) Convert each frame to grayscale to optimize the face detection process.
10) Detects faces in the grayscale frame using the Haar Cascade model, specifying parameters to refine the detection.
11)If faces are detected, crop the first detected face from the frame using the coordinates returned by the face detector
12)A cropped version of the face is converted to the desired PIL Image format that is compatible with the SR model input.
13) Prepare the cropped face image as input for the CarnModel and apply super-resolution without gradient tracking using `torch.no_grad()`.
14)Convert the output tensor from the super-resolution model back to a PIL image format and then transform it to an OpenCV-compatible format (BGR).
15)Resize the enhanced face frame to match the original frame dimensions and write it to the output video. If no face is detected, save the original frame instead.



```
# Load the CarnModel for super-resolution with scale 4
model = CarnModel.from_pretrained('eugenesiow/carn-bam', scale=4)  # You can adjust the scale as needed

# Load the video
input_video_path = "D02_20240705162958 (online-video-cutter.com) (8).mp4"
output_video_path = "D02_20240705162958 (online-video-cutter.com) (8) output.mp4"

# Load the pre-trained face detector (Haar Cascade)
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')

# Open the input video
video = cv2.VideoCapture(input_video_path)

# Get the frame width, height, and FPS
frame_width = int(video.get(cv2.CAP_PROP_FRAME_WIDTH))
frame_height = int(video.get(cv2.CAP_PROP_FRAME_HEIGHT))
fps = int(video.get(cv2.CAP_PROP_FPS))

# Prepare the output video writer
output_video = cv2.VideoWriter(output_video_path, cv2.VideoWriter_fourcc(*'mp4v'), fps, (frame_width, frame_height))

# Define a tensor-to-PIL conversion utility
to_pil_image = transforms.ToPILImage()
```



```
        # Convert the tensor output back to a PIL image using torchvision
        sr_image = to_pil_image(preds.squeeze(0))

        # Convert the PIL image back to OpenCV format (BGR)
        sr_image = np.array(sr_image)
        sr_image = cv2.cvtColor(sr_image, cv2.COLOR_RGB2BGR)  # Convert RGB back to BGR

        # Resize the upscaled face frame to fit the original frame size (optional)
        resized_face_frame = cv2.resize(sr_image, (frame_width, frame_height))

        # Write the resized face frame to the output video
        output_video.write(resized_face_frame)
    else:
        # Write the original frame if no face is detected
        output_video.write(frame)

# Release the video objects
video.release()
output_video.release()

print("Video processing complete with CarnModel. The enhanced video is saved as:", output_video_path)
```

Beyond this, we calculated the PSNR and SSIM values.
PSNR (Peak Signal-to-Noise Ratio) and **SSIM (Structural Similarity Index)** are two commonly used metrics for evaluating the quality of images and videos, especially in the context of image processing and compression.
PSNR (Peak Signal-to-Noise Ratio)
Typical Values: For effective super-resolution models, PSNR values usually range from **25 dB to 40 dB**.
Interpretation:

- **Below 20 dB**: Generally indicates poor quality, with significant differences from the original image.
- **20 dB to 25 dB**: Acceptable quality, but noticeable artifacts may be present.
- **25 dB to 30 dB**: Fair to good quality, with some visible artifacts but overall decent resemblance to the original image.
- **30 dB to 35 dB**: Good quality, often acceptable for practical applications.
- **Above 35 dB**: Excellent quality, indicating that the super-resolved image is very close to the original image

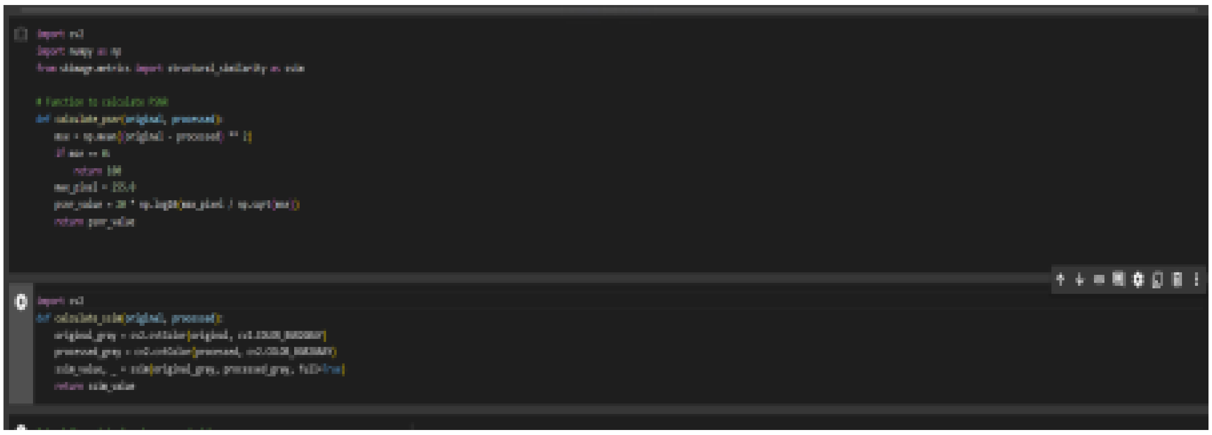
SSIM (Structural Similarity Index)

Typical Values: For effective super-resolution models, SSIM values typically range from 0.7 to 1.0.

Interpretation:

- **Below 0.5**: Indicates poor structural similarity; significant loss of details and visual integrity.
- **0.5 to 0.7**: Fair structural quality; some noticeable artifacts and differences from the original.
- **0.7 to 0.85**: Good structural similarity; generally acceptable, with some minor artifacts that may not be easily noticeable.
- **Above 0.85**: Excellent structural similarity; the super-resolved image retains most of the original structure and visual details.
- **1.0**: Perfect structural similarity (rare in practice).

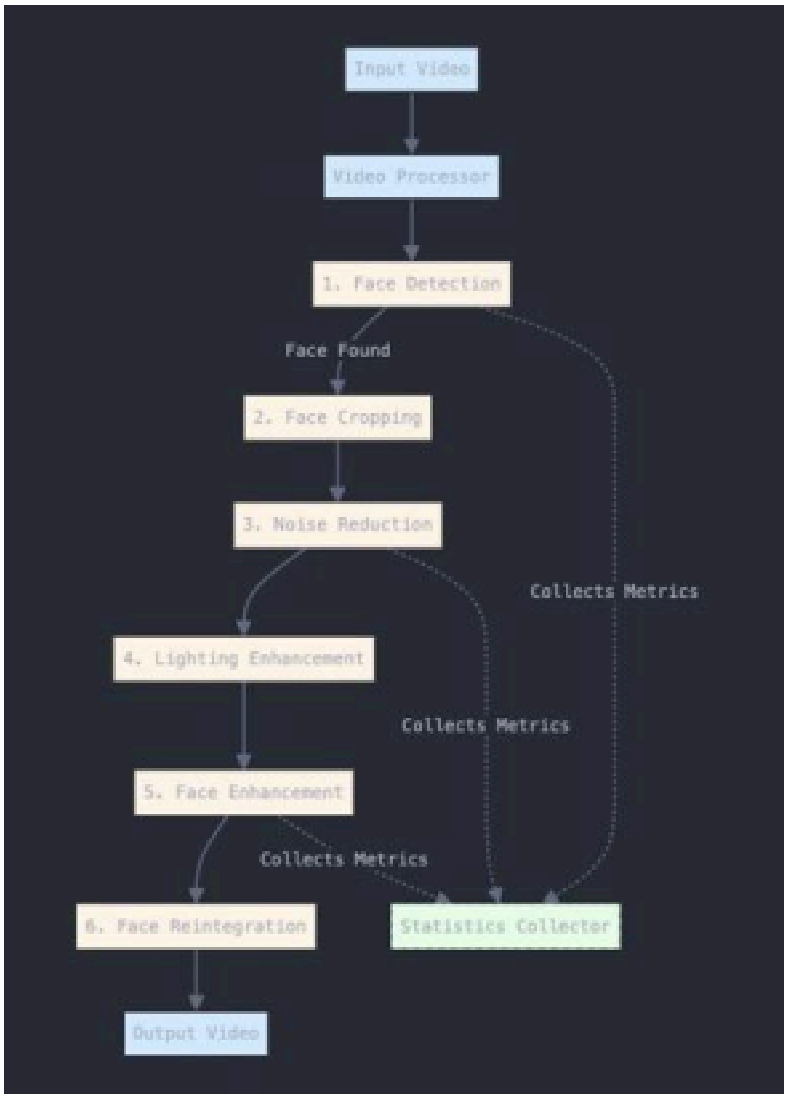
The PSNR values that we got for the respective non-intoxicated videos ranged from 28.75 to the highest being 36.12 and the SSIM values ranged from 0.4095 to 0.885.



```
import cv2
import numpy as np
from skimage.metrics import structural_similarity as ssim

# Function to calculate PSNR
def calculate_psnr(original, processed):
    mse = np.mean((original - processed) ** 2)
    if mse == 0:
        return 100
    max_pixel = 255.0
    psnr_value = 20 * np.log10(max_pixel / np.sqrt(mse))
    return psnr_value
```

```
import cv2
def calculate_ssim(original, processed):
    original_gray = cv2.cvtColor(original, cv2.COLOR_BGR2GRAY)
    processed_gray = cv2.cvtColor(processed, cv2.COLOR_BGR2GRAY)
    ssim_value, _ = ssim(original_gray, processed_gray, full=True)
    return ssim_value
```

Suggested Preprocessing Pipeline

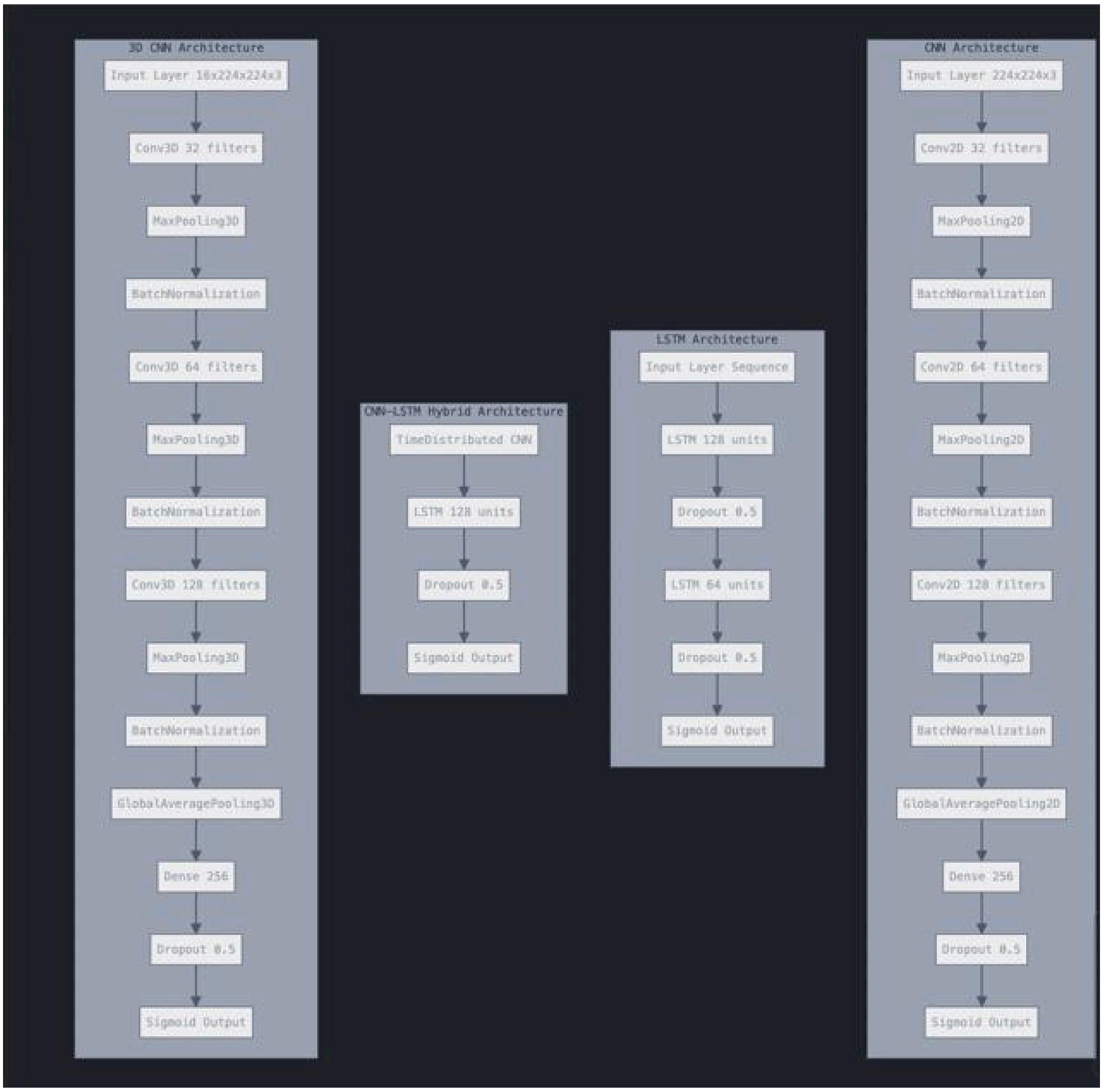


The system works in an efficient pipeline that starts with the **VideoProcessor class**, which performs basic video related tasks like extraction of frames or production of output. This processor seamlessly integrates with the powerful face detection module as implemented within the **ImprovedFaceDetector** class that employs **OpenCV's Haar Cascade classifiers** to detect and crop out facial regions from the videos' frames precisely.The enhancement procedure is done in very specific steps that are well planned. Once a face is visible, the system performs these sophisticated enhancement processes by using special purpose blocks. The **NoiseReducer makes use of bilateral and Gaussian filtering algorithms** that are embedded at adaptive levels depending on the **image noise level estimated by the NoiseEstimator**. This is followed by the **Lighting Enhancer, which efficiently applies CLAHE techniques** to enhance the offset and contrast of the frame without any visible artifacts.

Of special interest is the **AdaptiveFaceEnhancer** , which dynamically employs resolution enhancement techniques based on the quality of the evaluated facial areas. This is coupled with the **FaceReintegrator**, which guarantees a perfect match between the improved facial area and the original picture, allowing organic cuts and unions without compromising the holisticity of the image.

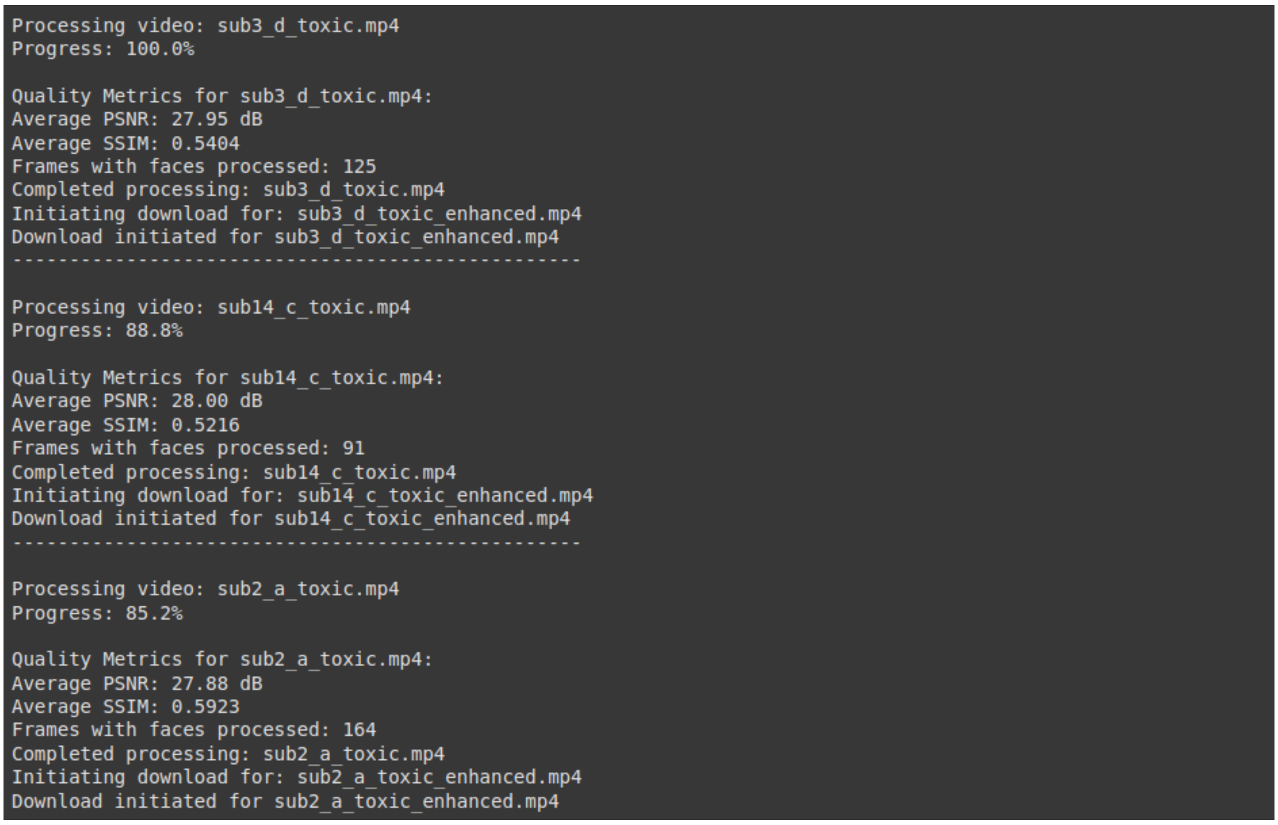
EnhancementStatsCollector, which analyzes the several parameters **including time spent in processing, quality of enhancement, confidence in face detection etc.**

Post this, we expanded our dataset to the **Oasis dataset and the dukan dataset**, wherein we applied the earlier codes. The Oasis dataset was really large with certain files being more than 60 mins long as well, and we needed to slice videos for each person using the **ffmpeg module**.
The classification of the videos into intoxicated and non intoxicated videos, could be done using a variety of different models such as **3D CNN, CNN-LSTM or even a basic LSTM or CNN model.**
A few neural network architectures that could work for the same are:

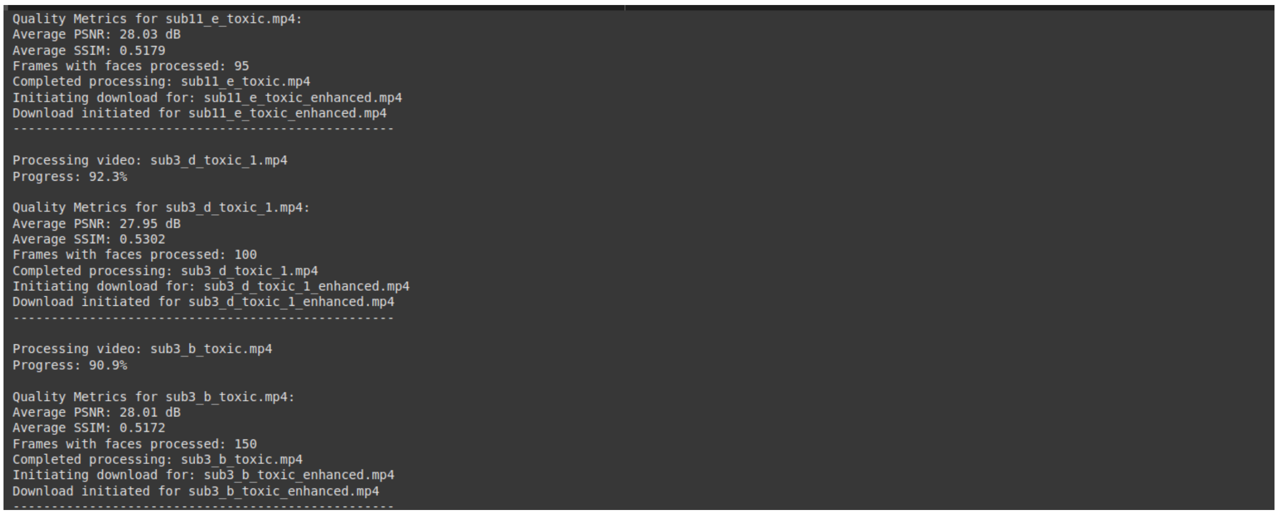


5. Conclusion and Future Work

The integration of the **CARN model with OpenCV's face detection** capabilities has yielded **promising results, as evidenced by the PSNR values ranging from 28.75 to 36.12 dB and SSIM values between 0.4095 and 0.885 for non-intoxicated subjects**. These metrics indicate generally **good to excellent quality enhancement** across the processed videos. For the **intoxicated subjects**, we got values ranging similarly, from **PSNR ranging from 27.88 to 28.04 and SSIM ranging from 0.5114 to 0.5972.**A few snapshots of the results are as follows:



```
Processing video: sub3_d_toxic.mp4
Progress: 100.0%

Quality Metrics for sub3_d_toxic.mp4:
Average PSNR: 27.95 dB
Average SSIM: 0.5404
Frames with faces processed: 125
Completed processing: sub3_d_toxic.mp4
Initiating download for: sub3_d_toxic_enhanced.mp4
Download initiated for sub3_d_toxic_enhanced.mp4
--------------------------------------------------

Processing video: sub14_c_toxic.mp4
Progress: 88.8%

Quality Metrics for sub14_c_toxic.mp4:
Average PSNR: 28.00 dB
Average SSIM: 0.5216
Frames with faces processed: 91
Completed processing: sub14_c_toxic.mp4
Initiating download for: sub14_c_toxic_enhanced.mp4
Download initiated for sub14_c_toxic_enhanced.mp4
--------------------------------------------------

Processing video: sub2_a_toxic.mp4
Progress: 85.2%

Quality Metrics for sub2_a_toxic.mp4:
Average PSNR: 27.88 dB
Average SSIM: 0.5923
Frames with faces processed: 164
Completed processing: sub2_a_toxic.mp4
Initiating download for: sub2_a_toxic_enhanced.mp4
Download initiated for sub2_a_toxic_enhanced.mp4
```



```
Quality Metrics for sub11_e_toxic.mp4:
Average PSNR: 28.03 dB
Average SSIM: 0.5179
Frames with faces processed: 95
Completed processing: sub11_e_toxic.mp4
Initiating download for: sub11_e_toxic_enhanced.mp4
Download initiated for sub11_e_toxic_enhanced.mp4
--------------------------------------------------

Processing video: sub3_d_toxic_1.mp4
Progress: 92.3%

Quality Metrics for sub3_d_toxic_1.mp4:
Average PSNR: 27.95 dB
Average SSIM: 0.5302
Frames with faces processed: 100
Completed processing: sub3_d_toxic_1.mp4
Initiating download for: sub3_d_toxic_1_enhanced.mp4
Download initiated for sub3_d_toxic_1_enhanced.mp4
--------------------------------------------------

Processing video: sub3_b_toxic.mp4
Progress: 90.9%

Quality Metrics for sub3_b_toxic.mp4:
Average PSNR: 28.01 dB
Average SSIM: 0.5172
Frames with faces processed: 150
Completed processing: sub3_b_toxic.mp4
Initiating download for: sub3_b_toxic_enhanced.mp4
Download initiated for sub3_b_toxic_enhanced.mp4
--------------------------------------------------
```

The proposed preprocessing pipeline, featuring the **VideoProcessor** and **ImprovedFaceDetector** classes, along with sophisticated components like the **NoiseReducer and LightingEnhancer**, establishes a robust foundation for video enhancement. The adaptive approach to face enhancement and reintegration ensures optimal processing while maintaining natural visual coherence.

The use of specialized algorithms such as CNNs for frame-level feature extraction, LSTMs for temporal sequence analysis, and 3D CNNs for spatiotemporal modeling exemplifies a comprehensive approach to video-based intoxication detection. This adaptive framework ensures robust feature extraction, dynamic temporal integration, and a holistic understanding of video data.

Future Work:

1. **Pipeline Improvements:**
   - ○ Enhancement of the NoiseEstimator to better handle varying lighting conditions
   - ○ Implementation of more sophisticated CLAHE parameters for the LightingEnhancer
   - ○ Integration of temporal consistency checks to ensure smooth transitions between frames
   - ○ Optimization of the AdaptiveFaceEnhancer for real-time processing capabilities
2. **Classification Model Development:**
   - ○ Given the various model pipelines developed, we can apply these models to our various datasets- the dukan dataset, kxr lab dataset and the oasis dataset, and check which provides the best accuracy.

6. References

- Sripal Thorupunoori, Siddartha Reddy Sungomula, Koushik Billakanti, Santhosh Gurram, Tavva Chinni Hemanth, Harpreet Kaur, and Manik Rakhra, "Camera Based Drunks Detection Mechanism Integrated With D-L (Deep Learning)," *2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, Amity University, Noida, India, Sep. 3-4, 2021. DOI: 10.1109/ICRITO51393.2021.9596283
- P. Iamudomchai, S. Pattanasak, and P. Seelaso, "Deep Learning Technology for Drunks Detection with Infrared Camera," *2020 IEEE International Conference on Biomedical Engineering and Technology (ICBET)*, King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand, 2020. DOI: 10.1109/ICBET.2020.997888
- G. Koukiou and V. Anastassopoulos, "Drunk Person Identification using Local Difference Patterns," in 2016 IEEE International Instrumentation and Measurement Technology Conference Proceedings, Taipei, 2016, pp. 1-6, doi: 10.1109/I2MTC.2016.7520351
- K. T. Huynh and H. P. T. Nguyen, "Drunkenness Detection Using a CNN with Adding Gaussian Noise and Blur in the Thermal Infrared Images," Int. J. Intell. Inf. Database Syst., vol. ., no. ., 2022
- Emmanuel Agu, "Fine-Grained Intoxicated Gait Classification Using a Bilinear CNN", in IEEE Sensors Journal · December Massachusetts, 2023, pp. 1-4, doi: 10.1109/JSEN.2023.3248868
- P. D. Rosero-Montalvo, V. F. López-Batista and D. H. Peluffo-Ordóñez, "Hybrid Embedded-Systems-Based Approach to in-Driver Drunk Status Detection Using Image Processing and Sensor Networks," in IEEE Sensors Journal, vol. 21, no. 14, pp. 15729-15740, 15 July 15, 2021, doi: 10.1109/JSEN.2020.3038143.
- Garrisson H, Scholey A, Ogden E, Benson S." The effects of alcohol intoxication on cognitive functions critical for driving: A systematic review.", in Accid Anal Prev. 2021 May;154:106052. doi: 10.1016/j.aap.2021.106052. Epub 2021 Mar 3. PMID: 33676142