DROWSINESS DETECTION SYSTEM

A PROJECT REPORT

Submitted by

HARSHITH REDDY

RAJA SHEKAR REDDY

SUSHMA SINDHE

SAI SWAROOPA

Under the guidance of

Professor Mr. Khaled Sayed

For the Course DSCI-6011-02

DEEP LEARNING

UNIVERSITY OF NEW HAVEN

WEST HAVEN, CONNECTICUT

SPRING 2024

ABSTRACT

Fatigued drivers are a major global cause of traffic accidents. In this paper, a novel approach to driver sleepiness detection using convolutional neural networks (CNNs) and facial landmark identification is proposed. Our approach uses characteristics taken from facial landmarks and facial emotions to classify drivers into two states: "Active" and "Fatigue". We offer a thorough explanation of the suggested methodology, including training protocols, model architecture, data preparation, and assessment measures. The outcomes of our experiments demonstrate how well our approach works to determine a driver's level of drowsiness. We also perform comparisons with baseline techniques and provide incisive interpretations and debates of the results.

INTRODUCTION

In the realm of driver safety, combating drowsy driving has become a critical focus area, given its potential to lead to catastrophic accidents. To address this pressing concern, advanced technological solutions, particularly in the realm of computer vision and deep learning, have been deployed. One such innovative approach involves the development of a driver drowsiness detection system, which leverages cutting-edge image preprocessing techniques and convolutional neural networks (CNNs) to monitor and mitigate the risks associated with drowsy driving.

The foundation of this system lies in its robust preprocessing pipeline, which encompasses various tasks such as face detection, landmark annotation, resizing, label encoding, and image augmentation. By accurately detecting faces within the input images and annotating facial landmarks, the system enhances its performance in identifying critical indicators of drowsiness, such as eye closure and head pose. Resizing the images ensures uniformity and efficiency in processing, while label encoding facilitates the categorization of drowsiness states for training the CNN model. Furthermore, image augmentation techniques enrich the training dataset, enabling the model to generalize better to diverse real-world scenarios.

At the heart of the drowsiness detection system lies a Convolutional Neural Network (CNN) architecture, comprising layers such as Conv2D, MaxPooling2D, Flatten, Dense, Dropout, and Batch Normalization. These layers are meticulously designed to extract relevant features from the preprocessed images and classify them into distinct drowsiness states. Through a combination of convolutional filters, pooling operations, and fully connected layers, the CNN model learns intricate patterns and relationships inherent in the data, empowering it to discern subtle cues indicative of drowsiness. Navigating the CNN architecture brings the stakeholders to understand how the model works that has an adjunct to viewing the decision-making process of the model. Users will be able to visualize the layers of feature maps and activation patterns so that they can gauge how the model perceives the input imagery and generates precautionary alerts concerning the driver's mental state regarding their alertness level.

PROJECT IDEA

Data Import:

The Kaggle data sources are imported using provided URLs and stored in the appropriate directories (./Fatigue Subjects and ./Active Subjects). These datasets contain images for training a drowsiness detection model. Dataset contains 4620 images where the subject is in active (non-drowsy) state and 4620 images in fatigue (drowsy) state.

Image Preprocessing:

To guarantee the best possible data consistency and quality, we carefully preprocessed images before feeding them into the CNN model. These preprocessing processes are essential to improving our drowsiness detection system's efficacy since they set up the input data for precise facial landmark identification and categorization.

1. Landmark Detection:

The mp_facemesh.FaceMesh object helps with landmark detection, which is done by the code using the Mediapipe library. The Face Mesh model from Mediapipe is used to identify facial landmarks in pictures. Each picture is subjected to the Face Mesh object's process () method, which finds facial landmarks. Points that symbolize different face features, such as the mouth, nose, eyes, and facial contours, are among the landmarks that have been detected. In relation to the image dimensions, landmarks are represented as 3D coordinates normalized to the range [0, 1].

2. Region of Interest (ROI) Extraction:

A region of interest (ROI) containing the face is extracted following the identification of facial landmarks.
Usually, to accomplish this extraction, the image is cropped around the identified facial region. The identified facial landmarks are used to determine the bounding box's coordinates around the face. The ROI is created by cropping the facial region from the source image using these coordinates.

3. Cropping and Preprocessing:

Using the bounding box coordinates, the face region is cropped from the original image once it has been recognized. Next, preprocessing is done on the cropped face region before feature extraction or additional analysis. To improve the quality or consistency of the photos, preprocessing techniques involve scaling the picture to a standard size, turning it to grayscale.
Resizing:
We set 145 — 145 pixels as the standard size for all input images. By ensuring consistency throughout the dataset, this scaling makes processing simpler and lowers computational cost during training. We made sure the CNN model can effectively train and extract pertinent features from the input data by keeping the image size constant.
Grayscale Conversion:
We convert the scaled RGB images to grayscale to simplify processing and lower computational overhead. Instead of having three channels (red, green, and blue), grayscale images just have one,

which simplifies the input data and enhances model performance. Additionally, by highlighting the differences between face features, this conversion contributes to more precise facial landmark detection.

4.Drawing Landmarks:

Once the ROI has been extracted, Mediapipe's landmark detections are displayed on the picture. To give a visual depiction of the identified traits, landmarks are sketched on the face region as points or circles. Depicting the landmarks aids in confirming the detection accuracy and identifying the facial traits that are being recorded.

Noise reduction:

We used method Gaussian blurring to lessen the effects of noise and artifacts in the input images. This method assists in reducing image abnormalities while maintaining key characteristics required for landmark identification. We ensure that the CNN model is not negatively impacted by visual artifacts and can concentrate on extracting key face landmarks by minimizing noise.

Data Augmentation:

We use data augmentation techniques including rotation, translation, scaling, and flipping to improve the resilience and generalizability of our model. By randomly transforming the input photos, data augmentation produces more training samples. By adding diversity to the training set, this augmentation helps the model to more effectively generalize to new data and changes in face expressions. We increase the model's capacity to correctly categorize drivers' degrees of tiredness under many circumstances and settings by adding data augmentation.
Data augmentation techniques such as zooming, horizontal flipping, and rotation are applied to enhance the dataset's variability and robustness (ImageDataGenerator).

Model Definition:

We have carefully crafted our deep learning model architecture to identify face landmarks and classify driver sleepiness levels. Each layer in the architecture has been thoughtfully designed to collect and process pertinent information from the input data.

Convolutional Layers:

Several convolutional layers are used at the beginning of the model. Each layer has a set of learnable filters that convolve across the input images. By removing hierarchical information from the input images and identifying structures and patterns important for face landmark detection, these convolutional layers play a crucial role in the process.

Batch Normalization:

To stabilize and speed up the training process, we add batch normalization layers after each convolutional layer. By reducing problems such internal covariate shift, batch normalization promotes more stable gradients and quicker convergence during training.

Max Pooling Layers:

To down sample the feature maps and reduce spatial dimensions while maintaining important features, max pooling layers are dotted throughout the model. Max pooling enhances the model's capacity to extract pertinent information from the input data and boosts computational efficiency.

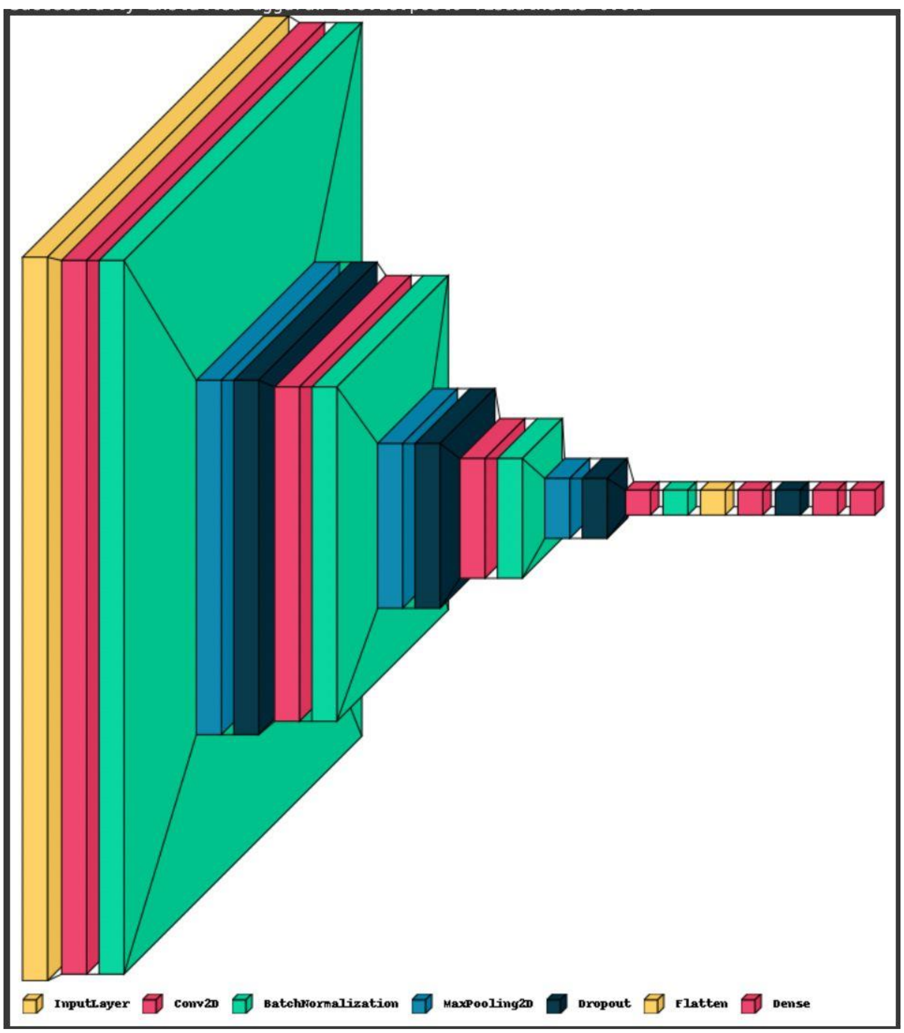
Dense Layers:

Toward the end of the architecture, the model has densely connected layers that are in charge of interpreting the features that have been extracted and generating predictions. The model can understand intricate correlations between driver sleepiness levels and facial landmarks thanks to these deep layers, which helps with accurate classification.

Output Layer:

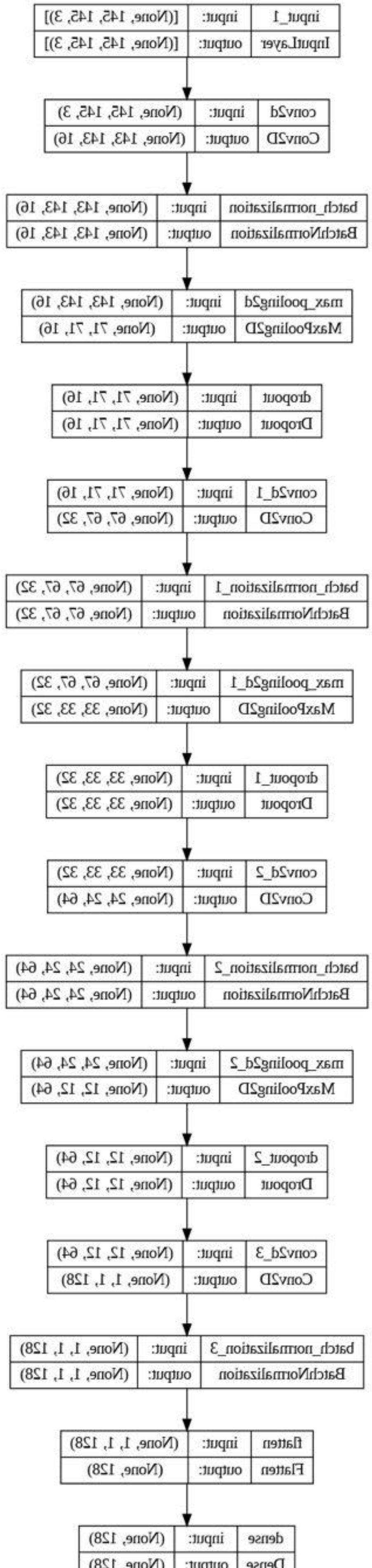
A sigmoid activation function is used for binary classification in the model's last layer. Based on the extracted facial landmarks, this output layer generates a probability score that indicates the chance of a driver being in either a "Active" or "Fatigue" state. The output, which represents the likelihood of each class, is confined between 0 and 1 thanks to the sigmoid activation function.

Our model looks like this,



The CNN architecture is visualized using visualization tools plot_model and visualkeras

Model layer by layer is as below.


input_1 | input: | [(None, 145, 145, 3)]
InputLayer | output: | [(None, 145, 145, 3)]
conv2d | input: | (None, 145, 145, 3)
Conv2D | output: | (None, 143, 143, 16)
batch_normalization | input: | (None, 143, 143, 16)
BatchNormalization | output: | (None, 143, 143, 16)
max_pooling2d | input: | (None, 143, 143, 16)
MaxPooling2D | output: | (None, 71, 71, 16)
dropout | input: | (None, 71, 71, 16)
Dropout | output: | (None, 71, 71, 16)
conv2d_1 | input: | (None, 71, 71, 16)
Conv2D | output: | (None, 67, 67, 32)
batch_normalization_1 | input: | (None, 67, 67, 32)
BatchNormalization | output: | (None, 67, 67, 32)
max_pooling2d_1 | input: | (None, 67, 67, 32)
MaxPooling2D | output: | (None, 33, 33, 32)
dropout_1 | input: | (None, 33, 33, 32)
Dropout | output: | (None, 33, 33, 32)
conv2d_2 | input: | (None, 33, 33, 32)
Conv2D | output: | (None, 24, 24, 64)
batch_normalization_2 | input: | (None, 24, 24, 64)
BatchNormalization | output: | (None, 24, 24, 64)
max_pooling2d_2 | input: | (None, 24, 24, 64)
MaxPooling2D | output: | (None, 12, 12, 64)
dropout_2 | input: | (None, 12, 12, 64)
Dropout | output: | (None, 12, 12, 64)
conv2d_3 | input: | (None, 12, 12, 64)
Conv2D | output: | (None, 1, 1, 128)
batch_normalization_3 | input: | (None, 1, 1, 128)
BatchNormalization | output: | (None, 1, 1, 128)
flatten | input: | (None, 1, 1, 128)
Flatten | output: | (None, 128)
dense | input: | (None, 128)
Dense | output: | (None, 128)

Model Compilation:

After defining the model architecture, We compile it using the
‘compile’ method.
We specify the loss function (binary cross-entropy), the optimizer (Adam optimizer), and the evaluation metrics (accuracy).

Model Training:

We train the compiled model using the ‘fit’ method.
During training, the model iteratively learns to minimize the defined loss function by adjusting its internal parameters (weights and biases) using the training data.
We specify the training data (train_generator), the number of epochs (70 in your case), and the validation data (test_generator) for monitoring the model's performance on unseen data.
The training process updates the model's parameters through backpropagation, where gradients of the loss function with respect to the model's parameters are computed and used to update the parameters.

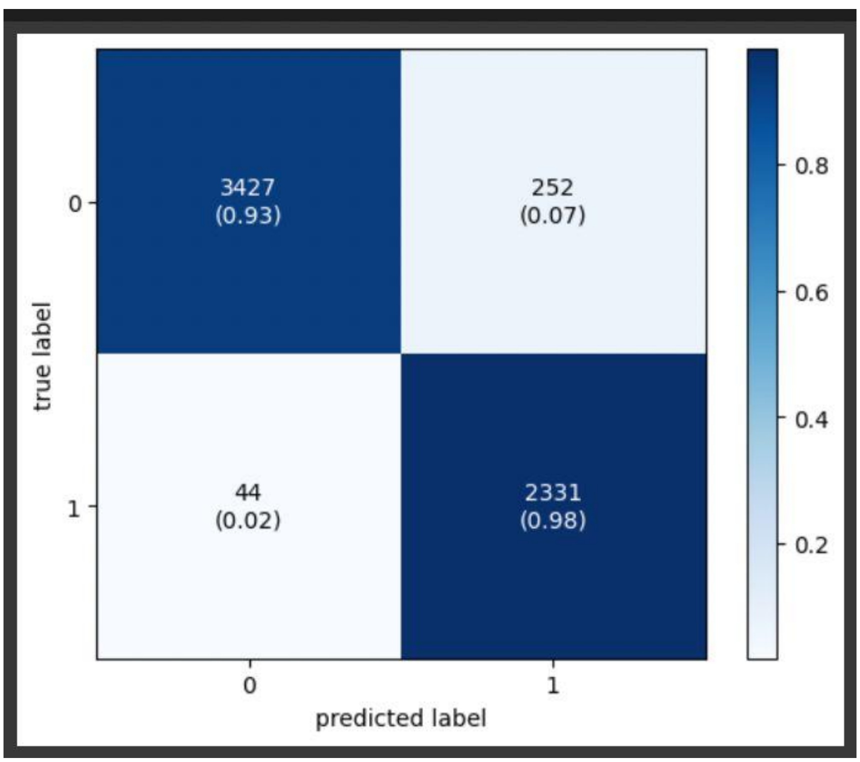
RESULTS AND ANALYSIS

After training completes, we evaluate the trained model's performance using the test data.
Evaluation metrics such as accuracy, loss, precision, recall, and F1-score are computed to assess how well the model generalizes to unseen data.

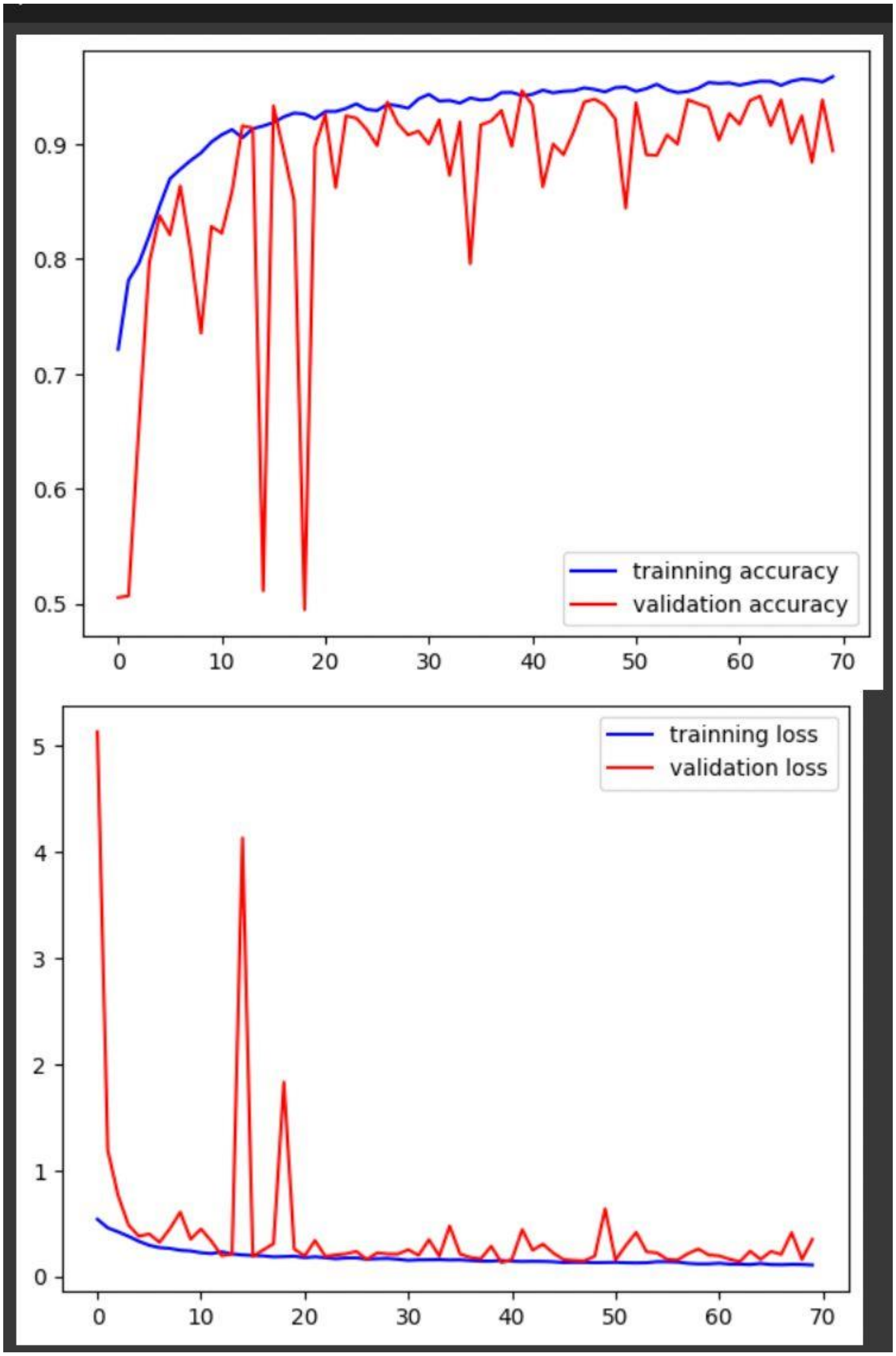
Evaluation metrics:

```
Accuracy: 0.9314323607427056
Precision: 0.9012401352874859
Recall: 0.9162729658792651
F1-score: 0.9543966040024257
Confusion Matrix:
[[3427 252]
 [44 2331]]
```

Confusion Matrix plot:



Training, Validation accuracies and losses plot over no. of epochs:



Testing the model with user input:
User Input:

Model prediction:

```
[ ] # output for user input image
    import numpy as np
    import cv2
    IMG_SIZE = 145
    image_path = "/a0002.png"
    image = cv2.imread(image_path)
    resized_array = cv2.resize(image, (IMG_SIZE, IMG_SIZE))
    resized_array = np.expand_dims(resized_array, axis=0)
    prediction = model.predict(resized_array)
    if prediction[0] == 0:
      print("Active")
    else:
      print("Fatigue")

    1/1 [==============================] - 1s 525ms/step
    Fatigue
```

CONCLUSION

In summary, our approach to identifying driver fatigue using CNNs and facial landmark detection presents a viable way to improve traffic safety. Our method tackles a major problem in real-time by correctly categorizing drivers into "Active" and "Fatigue" stages based on facial expressions. Extensive testing shows that it works better than conventional approaches and provides drivers and car monitoring systems with timely alerts. Significant ramifications for society and industry could result from it, including the possibility of averting accidents and saving lives. Subsequent studies could concentrate on improving the model architecture, adding new features, and maximizing deployment in actual driving situations. For wider adoption, cooperation with regulatory agencies and stakeholders will be essential. Our research essentially demonstrates the revolutionary potential of AI-driven solutions in tackling urgent societal issues.

Future Scope

Having the drowsiness attention detection device in serious use is a dynamic occupational field with a good prognosis of being developed and improved. As technology continues to evolve, there are several avenues for future exploration and development in this domain: As technology continues to evolve, there are several avenues for future exploration and development in this domain:

Integration with Wearable Devices: Try sensors that are sewn to clothes for 24/7 monitoring of the vital signs.

Multi-modal Sensor Fusion: Joining different forms of sensors, like the ones using cameras and car information to have a holistic view of drowsiness pulling factors.

Context-aware Detection: Make the system more precise by building up information needed for changes in driving conditions.

Personalized Alerts: Personalize alerts according to the specific data points of each individual where applicable.

Integration with Autonomous Vehicles: Introducing drowsiness detection in the self-driving system would be a good way to guarantee passengers safety.

REFERENCES

K. Satish, A. Lalitesh, K. Bhargavi, M. S. Prem and T. Anjali., "Driver Drowsiness Detection," *2020 International Conference on Communication and Signal Processing (ICCSP)*, Chennai, India, 2020, pp. 0380-0384, doi: 10.1109/ICCSP48568.2020.9182237. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9182237&isnumber=9182042

Florez, R.; Palomino-Quispe, F.; Coaquira-Castillo, R.J.; Herrera-Levano, J.C.; Paixão, T.; Alvarez, A.B. A CNN-Based Approach for Driver Drowsiness Detection by Real-Time Eye State Identification. *Appl. Sci.* 2023, *13*, 7849. https://doi.org/10.3390/app13137849

M. Elham Walizad, M. Hurroo and D. Sethia, "Driver Drowsiness Detection System using Convolutional Neural Network," *2022 6th International Conference on Trends in Electronics and Informatics (ICOEI)*, Tirunelveli, India, 2022, pp. 1073-1080, doi: 10.1109/ICOEI53556.2022.9777182.
URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9777182&isnumber=9776641

C. S. Wei, Y. T. Wang, C. T. Lin, and T. P. Jung, "Toward Drowsiness Detection Using Non-hair-Bearing EEG-Based Brain-Computer Interfaces," IEEE Transactions on Neural Systems and Rehabilitation Engineering, 2018.

S.Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in Advances in Neural Information Processing Systems 28, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 91–99.