# Driver Drowsiness Detection System

**Pratham Mehta (pm3483@nyu.edu), Aniket Sinha (aas10120@nyu.edu), Priyanshi Singh(ps4609@nyu.edu)**
Department of Electrical and Computer Engineering, New York University

Our CodeBase: Link

## Abstract

Driver drowsiness is a major cause of road accidents globally, often caused by fatigue and lack of sleep. To prevent accidents, early detection of drowsiness is important. There are various methods for detecting drowsiness, and this paper proposes a deep learning approach using convolutional neural networks to detect drowsiness in drivers by analyzing their face and eye regions. The study utilized two datasets: Closed Eye in the Wild (CEW) and Yawning Detection (YawDD). Results showed an accuracy of 97.05%.

## Introduction

The roads are experiencing an increase in the number of vehicles, resulting in a rise in road accidents that have become the leading cause of death in many parts of the world. As the responsible party, the driver holds the safety of themselves and their passengers in their hands. Unfortunately, drowsiness, which is often overlooked as a safety concern, can lead to accidents and fatalities if not addressed. To improve road safety, it is critical to address the issue of driver drowsiness and implement effective detection systems. With too many accidents occurring globally due to driver drowsiness, detecting and addressing this issue has become an essential component of modern driver monitoring systems.

## Literature Survey

Driver drowsiness detection is a crucial area of research in the field of transportation safety. To address this issue, many researchers have developed driver drowsiness detection systems using various techniques. Gwak et al. [1] evaluated the features of drowsiness detection using cameras. They divided the features into handcrafted features and features learned automatically using Convolutional Neural Networks (CNNs). They achieved an accuracy of 65.2% on pretended data. Kepesiova et al. [2] used a combination of Convolutional Neural Network (CNN), Convolutional Control Gate-based Recurrent Neural Network (Conv GRNN), and a voting layer to detect driver drowsiness. They achieved an average accuracy of 84.41%. You et al. [3] proposed a deep cascaded convolutional neural network and achieved an average accuracy of 94.80%. Mehta et al. [4] used the eye aspect ratio and eye closure ratio along with a Random Forest classifier to achieve an accuracy of 84%. Finally, Sathasivam et al. [5] proposed the use of the eye aspect ratio (EAR) and Support Vector Machine (SVM) classifier, which achieved an accuracy close to 94%. Overall, these studies showed that using CNNs and handcrafted features, along with classifiers like Random Forest and SVM, can achieve good accuracy in detecting driver drowsiness.

## Data Sets

There exist several standardized datasets that researchers have used for drowsiness detection. This paper uses two of these datasets. The first dataset used is the YawDD Video dataset [7], which contains videos recorded by a camera mounted on a car dashboard. The dataset consists of male and female drivers with some wearing glasses and others without. Fig. 1 shows some examples of this dataset. The second dataset used is the Closed Eyes in the Wild (CEW) dataset [6]. This dataset contains 2423 subjects, with 1192 people having closed eyes and 1231 people with open eyes.

Fig 1. Sample Images from YawDD DataSet

## Model Description

The model consists of a series of convolutional and max pooling layers, followed by some fully connected layers. The input shape of the CNN is determined by the shape of the training data, which in this case is a 4D tensor representing images with dimensions (height, width, channels) and a batch size. The first convolutional layer has 512 filters, with a kernel size of 3x3 and a ReLU activation function. This is followed by a max pooling layer with a pool size of 2x2, which reduces the spatial dimensions of the feature maps. The subsequent layers follow the same pattern, gradually decreasing the number of filters and feature map dimensions. The Flatten layer reshapes the output of the last convolutional layer into a 1D tensor, which is then passed through a Dropout layer to reduce overfitting. Finally, the fully connected layers consist of two hidden layers with ReLU activation and an output layer with a softmax activation function, which produces a probability distribution over the four possible classes. The model is compiled using the categorical_crossentropy loss function, the accuracy metric, and the Adam optimizer. This CNN model can be trained and evaluated on an image classification dataset.

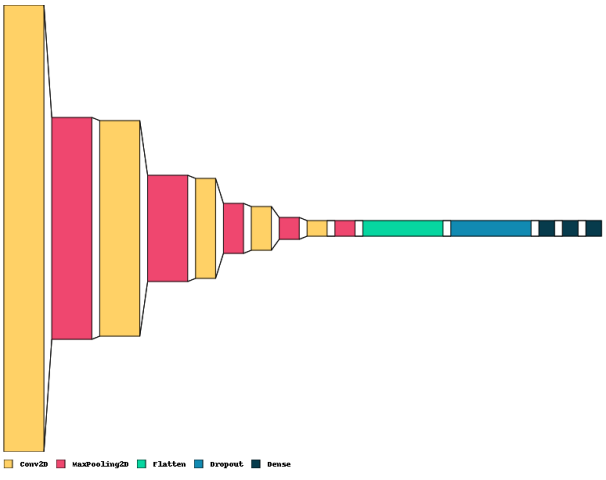
| | |
|---|---|
| Number of Convolutional Layers | 5 |
| Number of Filters in Convolutional Layers | 512, 512, 256, 256, 256 |
| Kernel Size in Convolutional Layers | (3, 3) |
| Number of Max Pooling Layers | 5 |
| Pooling Size in Max Pooling Layers | (2, 2) |
| Number of Dense Layers | 3 |
| Number of Units in Dense Layers | 128, 64, 4 |
| Loss Function | Categorical Crossentropy, also known as SoftMax |
| Optimizer | Adam |
| Learning Rate | 0.001(default of Adam) |
| Number of Parameters | 4,873,924 |

## Methodology

The code involves building a deep learning model to classify images as open or closed eyes, and yawning or not yawning. The dataset used is divided into two folders, 'Closed' and 'Open', for eye classification, and 'yawn' and 'no_yawn' for yawning classification. We then define a function, 'face_for_yawn', which uses a Haar cascade classifier to detect faces in images and crops them to obtain only the region of interest, which is then resized to 145 x 145 pixels. Another function, 'get_data', is defined to obtain the eye images and resize them to the same dimensions as the yawning images. The 'append_data' function combines the two datasets obtained from the previous two functions, and the resulting data is then split into training and testing sets using 'train_test_split' from scikit-learn. The training data is further augmented using the 'ImageDataGenerator' from Keras, which performs various transformations on the images, such as rotation, zooming, and horizontal flipping.

A Convolutional Neural Network (CNN) is built using the Keras Sequential model, with multiple layers of convolution, max pooling, and dense layers. The model is compiled with 'categorical_crossentropy' as the loss function, 'accuracy' as the metric to measure the performance, and 'adam' as the optimizer. The model is trained on the augmented training set for 50 epochs, and the validation accuracy is monitored using the test set. The training and validation loss and accuracy are also plotted using Matplotlib to assess the performance of the model.

## Network Architecture Diagram



## Drowsiness Detection

Our deep learning model is trained on a dataset of images, which are categorized as either "yawn" or "no_yawn" and "Closed" or "Open" eyes. The training set is augmented using techniques such as zooming, horizontal flipping, and rotation. The images are preprocessed by resizing them to a fixed size of 145x145 pixels and normalizing the pixel values. A Haar cascade classifier is used to detect faces in the images, and a rectangular region of interest (ROI) is selected around the detected face. The ROI is then resized to the same fixed size as the original image and fed into the CNN as input.

The CNN consists of several convolutional layers with increasing filter sizes, followed by max-pooling layers that reduce the spatial dimensions of the feature maps. The output of the last max-pooling layer is flattened and passed through a fully connected (dense) layer, which has a dropout regularization to reduce overfitting. Finally, the output layer has four nodes, representing the four classes (yawn, no_yawn, Closed eyes, and Open eyes), and uses a softmax activation function to generate class probabilities.

The CNN is trained using the Adam optimizer and categorical cross-entropy loss function. The model is trained for 50 epochs, with a batch size of 32. The validation data is used to evaluate the model after each epoch, and the best model is saved. Once the training is complete, the model is evaluated on the test set to determine its performance on unseen data. Thus, drowsiness is detected by using the trained CNN to classify images of individuals' faces and eyes as either alert or drowsy based on the predictions of the model.

The trained model will be integrated into a driver drowsiness detection system that uses real-time camera footage. If the model detects drowsiness, an alarm will alert the driver to take a break or stop driving, potentially reducing accidents caused by driver drowsiness.

## Data Augmentation

Data augmentation is a process that improves the generalization of a model by creating variations of the original image data. In our case, these two data augmentation techniques are implemented:

**Random cropping** is used to randomly crop a section of the image and resize it to the original size of the image. This technique helps the model to learn features from different parts of the image and hence makes it more robust to variations in the position of the eyes or the face in the image. In our project, this is achieved using the `tf.image.random_crop()` function.

**Horizontal flipping** is used to flip the image horizontally. This technique helps the model to learn features from different orientations and hence makes it more robust to variations in the orientation of the head or face in the image. In the project, this is achieved using the `tf.image.random_flip_left_right()` function.

These data augmentation techniques are applied to the training data during each epoch of training using the `tf.data.Dataset` API. This ensures that the model is trained on different variations of the images and hence improves its performance on unseen data. By using data augmentation, the model can learn to identify drowsiness under various conditions, such as different lighting conditions, different head and eye positions, and different driver appearances.

## Data Transformation

The data transformation in this project involves preprocessing the input images to be fed into the deep learning model. The `get_data()` function reads in images from the `../input/drowsiness-dataset/train/` directory, resizes them to 145x145 pixels, and converts them to arrays. Similarly, the `face_for_yawn()` function reads in images from the `../input/drowsiness-dataset/train/yawn` and `../input/drowsiness-dataset/train/no_yawn` directories, detects the faces in the images using the `haarcascade_frontalface_default.xml` file, resizes the detected faces to 145x145 pixels, and appends them to a list along with their corresponding label. The resulting preprocessed data is then combined using the `append_data()` function, which concatenates the face images with the eye images and returns them as a numpy array. Finally, the input data is split into training and testing sets using the `train_test_split()` function and transformed using the `ImageDataGenerator()` class, which performs additional image augmentation such as zooming, flipping, and rotating the images to increase the size of the training data and improve the model's performance.

## Selecting Optimizer

In deep learning, the choice of optimizer is an essential aspect of designing an effective model. Optimizers are algorithms that adjust the parameters of a model to minimize the difference between predicted and actual outputs. This process helps to improve the accuracy of the model by reducing overall loss. There are several popular optimizers available for use, each with its strengths and weaknesses. The choice of optimizer typically depends on the specific problem and data being used, as well as the architecture of the model. Ultimately, selecting an appropriate optimizer can have a significant impact on the performance of a deep learning model.

The optimization algorithm chosen in this project is **Adam**, which stands for Adaptive Moment Estimation. Adam is a popular optimization algorithm used in deep learning because of its ability to update network weights in a more adaptive way compared to other traditional gradient descent algorithms. It combines the benefits of two other popular optimization algorithms, namely, Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp). The Adam optimizer uses a combination of gradient information and past gradients to adjust the learning rate for each parameter. It has been shown to work well with large datasets and deep neural networks, which makes it an excellent choice for the image classification task being performed in our project. The Adam optimizer is widely used in many state-of-the-art deep learning models due to its ability to converge quickly and perform well on a variety of tasks.

## Selecting Loss Function

In our project, the loss function chosen is the **categorical cross-entropy**. It is a commonly used loss function in multi-class classification tasks, which aims to minimize the difference between the predicted probabilities and the true class labels. The categorical cross-entropy loss calculates the difference between the predicted probability distribution and the true distribution, and then takes the logarithm of the difference, and finally multiplies it by -1 to get the loss value. The main advantage of using this loss function is that it penalizes the model heavily for making high-confidence incorrect predictions, which is particularly important in multi-class classification tasks where misclassification can easily occur between similar classes.

This loss function is also commonly known as **softmax loss**. This is because the function uses the softmax activation function to calculate the probabilities of each class, which are then compared to the true labels to compute the loss. The softmax function is a popular choice for multi-class classification problems because it ensures that the predicted probabilities sum to 1, making it easy to interpret the results and compare the relative likelihoods of different classes.

## Results

| | |
|---|---|
| **Maximum Testing Accuracy** | **97.06** |
| **Maximum Training Accuracy** | **97.62** |
| **Minimum Training Loss** | **0.0628** |
| **Minimum Testing Loss** | **0.0755** |

The proposed driver drowsiness detection system was trained and evaluated on two datasets: YawDD and CEW. The maximum testing accuracy achieved on the YawDD dataset was 97.06%, and the maximum training accuracy achieved was 97.62%. The minimum training loss was 0.0628, and the minimum testing loss was 0.0755. The trained model was also evaluated on test images and the accuracy was found to be 96.02% and the loss was 0.1072. The proposed driver drowsiness detection system, based on computer vision and deep learning algorithms, has demonstrated high accuracy in detecting driver drowsiness on two datasets - YawDD and CEW. The YawDD dataset, in particular, presents a challenge due to the variability in yawning and non-yawning frames. However, our system achieved excellent results on this dataset, which suggests its effectiveness in real-world scenarios.

The trained model has been integrated into a complete driver drowsiness detection system that uses a web camera to capture the driver's facial expressions and head movements in real-time. When the model predicts that the driver is drowsy, an alarm is triggered to alert the driver and prompt them to take a break or stop driving, thus further enhancing driver safety.The results demonstrate that our system can accurately predict whether the driver is drowsy or not based on their facial expressions and head movements. Therefore, it can be an effective tool for detecting driver drowsiness in real-time applications, such as in-vehicle driver monitoring systems. The proposed

system has the potential to contribute to improving road safety and reducing the number of accidents caused by driver drowsiness.

In conclusion, the proposed driver drowsiness detection system can be an effective tool for real-time driver monitoring and detection of drowsiness. The results from the evaluation of the system on two datasets demonstrate its high accuracy and effectiveness, and its integration into a complete driver drowsiness detection system further enhances its potential to improve driver safety. The system can be a valuable addition to existing driver monitoring systems and can potentially save lives by preventing accidents caused by driver drowsiness.

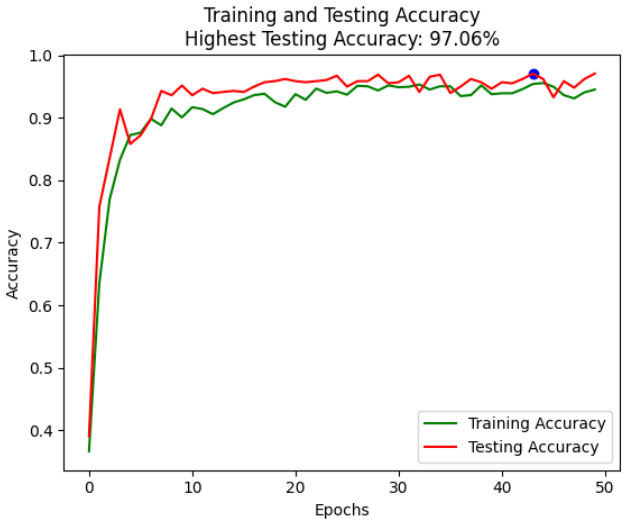



Fig. Training accuracy vs Testing accuracy

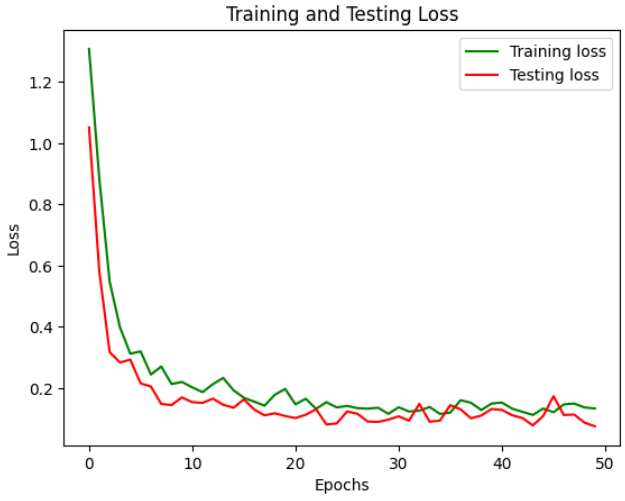



Fig: train loss vs test loss

## Conclusion

The driver drowsiness detection system was successfully developed using a Convolutional Neural Network (CNN) architecture. The system was trained on two datasets, YawDD and CEW, and achieved a maximum testing accuracy of 97.06%.

The YawDD dataset consists of images of drivers with different levels of drowsiness, while the CEW dataset includes images of drivers with different emotional expressions. The use of these datasets allowed for the development of a robust system that can accurately detect drowsiness in various situations. The results indicate that the developed system has the potential to be used in real-world applications, such as in vehicles equipped with a camera to monitor the driver's state. The system can provide an alert to the driver in case of drowsiness, which can help prevent accidents and improve road safety.

Future work in deep learning-based driver drowsiness detection systems can explore various technical aspects to enhance their performance. Firstly, the integration of multi-modal data can provide a richer understanding of drowsiness. Combining facial cues from images or videos with eye gaze tracking, head movement analysis, and physiological signals like heart rate or brain activity can offer a more comprehensive representation of drowsiness-related patterns. This integration can be achieved through advanced fusion techniques such as multi-stream networks or attention mechanisms that selectively attend to relevant modalities.

Improving the generalization of models across diverse populations and environmental conditions is another crucial aspect. Currently, many drowsiness detection models are trained and evaluated on specific datasets, limiting their performance in real-world scenarios. Future research can focus on designing models that are robust to variations in demographics, lighting conditions, camera angles, and other environmental factors. This can involve strategies such as domain adaptation, data augmentation, or leveraging adversarial training to simulate a wider range of conditions during training.

Efficient real-time implementation is essential for practical deployment of drowsiness detection systems. Techniques such as model compression, quantization, or hardware

accelerators can be explored to optimize the computational requirements of deep learning models. Additionally, designing lightweight architectures tailored specifically for drowsiness detection can ensure low-latency processing, enabling timely alerts to prevent accidents.

Lastly, transfer learning techniques can be leveraged to overcome the challenge of limited annotated datasets. By pre-training models on large-scale datasets or related tasks, and then fine-tuning them on drowsiness-specific data, models can benefit from learned features and improve performance even with limited labeled samples. This approach can alleviate the data scarcity issue and help develop robust drowsiness detection models.

By addressing these technical aspects, future research in driver drowsiness detection systems can enhance their accuracy, generalization, real-time capabilities, privacy preservation, and data efficiency. These advancements will contribute to more effective and practical systems that can mitigate the risks associated with drowsy driving, ultimately leading to improved road safety.

## References


1. Gwak, J., Hirao, A., Shino, M. (2020). An investigation of early detection of driver drowsiness using ensemble machine learning based on hybrid sensing. Applied Sciences, 10(8), 2890. Retrieved from https://doi.org/10.3390/app10082890
2. Kepesiova, Z., Ciganek, J., & Kozak, S. (2020). Driver drowsiness detection using convolutional neural networks. In 2020 Cybernetics & Informatics (K&I). Retrieved from https://doi.org/10.1109/ki48306.2020.9039851
3. You, F., Li, X., Gong, Y., Wang, H., & Li, H. (2019). A real-time driving drowsiness detection algorithm with individual differences consideration. IEEE Access, 7, 179396-179408. Retrieved from https://doi.org/10.1109/access.2019.2958667
4. Mehta, S., Dadhich, S., Gumber, S., & Bhatt, A. J. (2019). Real-time driver drowsiness detection system using eye aspect ratio and eye closure ratio. SSRN Electronic Journal. Retrieved from https://doi.org/10.2139/ssrn.3356401
5. Sathasivam, S., Mahamad, A. K., Saon, S., Sidek, A., Som, M. M., & Ameen, H. A. (2020). Drowsiness detection system using eye aspect ratio technique. In 2020 IEEE Student Conference on Research and Development (SCOReD). Retrieved from https://doi.org/10.1109/scored50371.2020.9251035
6. Closed Eye in the Wild dataset (CEW) - Tan, X., Lu, Y., Yang, J., & Jin, L. (2017). Closed eye in the wild (cew) database and benchmarking. arXiv preprint arXiv:1705.05679. Retrieved from http://parnec.nuaa.edu.cn/_upload/tpl/02/db/731/template731/pages/xtan/ClosedEyeDatabases.html
7. Yawning Detection Dataset (YawDD) - Abtahi, S., Omidyeganeh, M., Shirmohammadi, S., & Hariri, B. (2019). YawDD: a yawning detection dataset. IEEE Dataport. Retrieved from https://ieee-dataport.org/open-access/yawdd-yawning-detection-dataset
8. Savas, B. K., & Becerikli, Y. (2020). Real time driver fatigue detection system based on multi-task ConNN. IEEE Access, 8, 12491-12498. Retrieved from https://doi.org/10.1109/access.2020.2963960
9. Driver drowsiness detection. (2023, May 15). In Wikipedia, The Free Encyclopedia. Retrieved from https://en.wikipedia.org/wiki/Driver_drowsiness_detection
10. Brownlee, J. (2017, May 11). Adam Optimization Algorithm for Deep Learning. Retrieved from https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/
11. Keras Team. (2021). Keras Documentation. Retrieved May 15, 2023, from https://keras.io
12. TensorFlow. (n.d.). Retrieved May 15, 2023, from https://www.tensorflow.org/
13. Awan, A. (2023, May 15). A Complete Guide to Data Augmentation. Retrieved from https://www.datacamp.com/tutorial/complete-guide-data-augmentation
14. Gong, D. (2022, March 8). Data Transformation and Feature Engineering. Retrieved from https://towardsdatascience.com/data-transformation-and-feature-engineering-e3c7dfbb4899
15. Analytics Vidhya. (2022). Visualize Deep Learning Models using VisualKeras. Retrieved from https://www.analyticsvidhya.com/blog/2022/03/visualize-deep-learning-models-using-visualkeras/
16. Koech, K. E. (2022, March 8). Cross-Entropy Loss Function. Retrieved from https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e
17. Scikit-learn. (2023, May 15). Scikit-learn: Machine learning in Python. Retrieved from https://scikit-learn.org/stable/
18. OpenCV. (n.d.). Face Detection using Haar Cascades. Retrieved from https://docs.opencv.org/3.4/d2/d99/tutorial_js_face_detection.htm
19. Mandal, M. (2021). Convolutional Neural Networks (CNN). Retrieved from https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/
20. Brownlee, J. (2023). A Tour of Optimization Algorithms. Machine Learning Mastery. Retrieved from https://machinelearningmastery.com/tour-of-optimization-algorithms/
21. Deshpande, A. V. (2023). Transfer learning-based drowsiness detection system for driver assistance. Retrieved from https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4239335