Real-Time Drowsiness Detector using a CNN Based Model

1st Rifah Nanzeeba
Dept. of EEE
BUET
Dhaka, Bangladesh
1906139@eee,buet.ac.bd

2nd Kazi Mubashir Tazwar
Dept. of EEE
BUET
Dhaka, Bangladesh
1906142@eee,buet.ac.bd

3rd Prantik Paul
Dept. of EEE
BUET
Dhaka, Bangladesh
1906145@eee,buet.ac.bd

4th Sadad Hasan
Dept. of EEE
BUET
Dhaka, Bangladesh
1906163@eee,buet.ac.bd

***Abstract*—The prevalence of drowsy driving poses a significant risk to road safety, leading to numerous accidents and fatalities worldwide. In this paper, we propose a novel real-time drowsiness detection system utilizing a Convolutional Neural Network (CNN) based transfer learning approach. This methodology focuses on leveraging the power of deep learning to accurately identify signs of driver drowsiness from facial images captured by an in-vehicle camera. The proposed CNN architecture is designed to extract discriminative features from facial images, enabling the detection of subtle cues indicative of drowsiness such as eye closure, head movement, and facial expressions. A comprehensive dataset was employed consisting of diverse facial images collected under varying lighting conditions and driver states to train and evaluate our model. Experimental results demonstrate the effectiveness of our approach in accurately detecting drowsiness in real-time scenarios, achieving superior performance compared to existing methods. The proposed system holds promise for integration into existing driver assistance systems, contributing to enhanced road safety by providing timely alerts to drivers, thereby mitigating the risk of accidents caused by drowsy driving.**

***Index Terms*—Convolutional Neural Network(CNN),InceptionV3, drowsiness-detection, ImageNet.**

I. Introduction

The global average of road accident deaths of a nation stand at 8,000 annually [1]. However, in Bangladesh, the number is around 24,000 in a calendar year which is around 3 times the global average [1].In recent years, advancements in computer vision and deep learning have opened up new opportunities for developing robust drowsiness detection systems. These systems leverage the power of machine learning algorithms to analyze visual cues from the driver's face, such as eye movements, facial expressions, and head posture, to infer their level of alertness. Among various machine learning techniques, Convolutional Neural Networks (CNNs) have demonstrated remarkable performance in image recognition tasks, making them well-suited for drowsiness detection applications. In this paper, a comprehensive investigation was presented into the development of a real-time drowsiness detection system using a CNN-based approach. Our objective is to design a system capable of accurately identifying signs of drowsiness from facial images captured by an in-vehicle camera, thereby providing timely alerts to the driver. By leveraging deep learning techniques, it was aimed to overcome the limitations of traditional rule-based approaches, which often struggle to generalize across diverse driving conditions and individual characteristics.

Identify applicable funding agency here. If none, delete this.

II. Methodology

A. Choosing a Model

For this project,the Inception V3 Model was selected as our base model after testing our transfer learning model for other famous architectures like VGG16, MobileNetV2 and ResNet50V2.The model benchmarking will be later shown with performance metrics and results. Among them, Inception V3 performed best and so we chose it. InceptionV3 is a convolutional neural network architecture that was developed by Google researchers as part of the Inception family of models. It is specifically designed for image recognition and classification tasks. The "V3" in its name indicates that it is the third version in the Inception series.InceptionV3 is a deep convolutional neural network featuring multiple layers, including convolutional, pooling, and fully connected layers. Its innovative design incorporates inception modules, enabling efficient feature capture across various spatial scales by executing multiple convolutions with different filter sizes in parallel and concatenating the results. This approach facilitates the extraction of both fine-grained and global features from input images. Additionally, InceptionV3 integrates reduction blocks to decrease spatial dimensions while increasing depth, thereby reducing computational complexity and enhancing efficiency. Notably, the model includes auxiliary classifiers, extra softmax layers inserted at intermediate points, which aid in training by supplying additional gradients and regularization to mitigate the vanishing gradient problem. Moreover, InceptionV3 serves as a popular choice for transfer learning, as pre-trained weights on large-scale image datasets, such as ImageNet, are readily available. This feature allows researchers and practitioners to fine-tune the model for specific tasks with relatively small datasets, leveraging the model's robust capabilities.

1) Model Benchmarking: For our transfer learning model we have evaluated the performance metrics like accuracy and other classification metrics such as precision, recall and f1-score for all the four base models InceptionV3, VGG16, MobileNetV2 and ResNet50V2. The results are shown below:

Model	Accuracy
VGG16	0.5
InceptionV3	0.8818
ResNet50V2	0.8304
MobileNetV2	0.8186

Base Model	Class	Precision	Recall	f-1 score
VGG16	Closed	0.0	0.0	0.0
	Opened	0.5	1.0	0.67
InceptionV3	Closed	0.82	0.98	0.89
	Opened	0.98	0.78	0.87
ResNet50V2	Closed	0.75	0.99	0.85
	Opened	0.99	0.67	0.80
MobileNetV2	Closed	0.74	0.99	0.85
	Opened	0.98	0.65	0.78

2) Haarcascade classifier: The Haarcascade classifier is a popular machine learning object detection method that is used to identify objects in image or video. An integral image representation of the input image is used in this method which allows faster computations. Haar like features are extracted from input image by variations of intensity within a region of the image. Based on Haar like features, the model is trained with positive and negative samples. Multiple weak classifiers are used to speed up the detection process. Then adaptive boosting is used to combine the weak classifiers into a strong classifier. During each iteration more weights are assigned to misclassified examples to improve the model. Then a sliding window is applied to the input image to start detecting the target object based on the learned features. Weak classifiers quickly reject a region when it doesn't meet the criteria or threshold set previously. By following these simple steps, the Haarcascade classifier has been able to detect the eye region from the image of face captured by the webcam.

B. Model Architecture

InceptionV3 is a convolutional neural network architecture for image classification developed by Google Research.It was released in the year 2015 and it has a total of 42 layers which consists of five convolutional layers, two max-pooling layers, 11 inception modules, one average pooling layer, and one fully connected layer (Fig. 1).It has a lower error rate than its predecessors. It has been shown to attain greater than 78.1% accuracy on the ImageNet dataset. The initial part of the network involves a series of convolutions and pooling operations to process the input image. This includes convolutional layers with batch normalization and activation functions such as ReLU. The core of InceptionV3 consists

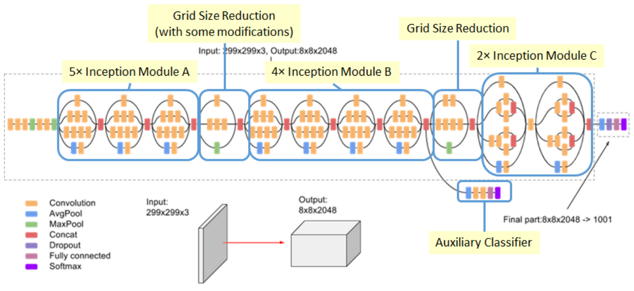

Fig. 1. InceptionV3 model architecture

of multiple inception modules stacked together. Each inception module is composed of parallel branches with different types of convolutions, pooling, and concatenation operations. InceptionV3 includes reduction modules to reduce the spatial dimensions of the feature maps. It also includes auxiliary classifiers at intermediate layers to prevent gradient loss. Fully connected layers at the end of the network have dropout regularization to help with overfitting [3].

C. Selecting a Dataset for the Model

The dataset for this project is made from two different datasets. This first is MRL eye Dataset. The dataset comprises annotations of various key properties meticulously organized in a consistent manner [2]. These properties encompass the subject ID, encapsulating data from 37 individuals, with a gender distribution of 33 men and 4 women. Each image is uniquely identified by an image ID, with a total of 84,898 images present in the dataset. Additionally, annotations include the gender of each individual (0 for men, 1 for women) and whether subjects wore glasses (0 for no, 1 for yes). Moreover, annotations detail the state of the eyes (0 for closed, 1 for open) and categorize reflections into three sizes: none (0), small (1), and big (2). Lighting conditions during image capture are annotated as bad (0) or good (1). Finally, information regarding the sensor used for image capture is provided, with three distinct sensor types identified by unique IDs: RealSense (01), IDS (02), and Aptina (03). These comprehensive annotations offer valuable insights into dataset characteristics, facilitating nuanced analysis and interpretation across various applications in computer vision and related fields. We selected our ROI(Region of Interest) which basically selected the eye portion for consideration [4] The second dataset used in our project is Frame Level Driver Drowsiness Detection (FL3D) dataset which is basically derived from NITYMED dataset (Night-Time Yawning-Microsleep-Eyeblink-driver Distraction). From the video data of NITYMED dataset the frames were extracted and FL3D was constructed. To make our project robust in detecting drowsiness, we introduced this night time data in our training dataset. With coding we extracted only the eye from the frames of FL3D dataset after hand-picking the frames manually, then filtered it. We manually balanced the data with night, day, spectacles, non-spectacle images. We also had to manipulated our dataset to include pictures with motion blur in it to make our model more robust. As there were no available dataset of blurred faces during driving, we had to prepare one on our own by writing a fucntion that takes in a portion of random images in our training dataset and add motion blur to it.

D. Training the Model

The methodology encompasses the preparation and augmentation of image data for training a convolutional neural network (CNN) model. An `ImageDataGenerator` object, denoted as `train_datagen`, is initialized with prescribed parameters to facilitate data augmentation and normalization. These parameters entail rescaling pixel values to the range $[0, 1]$, implementing random rotations (up to 20 degrees), shear transformations, zooming (up to 20%), and random shifts in width and height (up to 20%). Furthermore, the data is segregated into training and validation subsets, with a specified proportion (20%) allocated for validation. Subsequently, the `flow_from_directory` method is invoked to generate augmented image data batches from a designated directory. These batches are standardized by resizing images to 80×80 pixels and categorically labeling them. This rigorous methodology ensures the creation of diverse and augmented datasets, thereby enhancing the robustness and performance of the CNN model during training.

Then it initializes a convolutional neural network (CNN) model using the InceptionV3 architecture for transfer learning [3]. The InceptionV3 model is first loaded with pre-trained weights from the ImageNet dataset [2], excluding the fully connected layers at the top. Subsequently, additional layers are appended to the model to construct a custom classifier. These layers include a flattening layer to convert the output tensor into a one-dimensional vector, a fully connected layer with 64 neurons and ReLU activation, a dropout layer with a dropout rate of 0.5 for regularization, and a final output layer with softmax activation, yielding probabilities for two classes. Finally, the model is instantiated by specifying the input and output tensors, with all layers of the InceptionV3 model frozen to prevent weight updates during training. This strategy leverages the pre-trained weights of the InceptionV3 model while allowing fine-tuning of the custom classifier to suit the specific classification task at hand.

The `compile` method is used to configure the CNN model for training. In this line of code, the Adam optimizer is specified as the optimizer, categorical cross-entropy is selected as the loss function, and accuracy is designated as the metric to be monitored during training. The Adam optimizer is a popular choice for gradient-based optimization, while categorical cross-entropy is commonly used for multi-class classification tasks. Additionally, accuracy provides insight into the model's performance by measuring the proportion of correctly classified samples. Subsequently, the `fit` method is invoked to train the model using the provided training data (`train_data`). The input data (`x`) is set to `train_data`, which contains batches of augmented image data generated by the `ImageDataGenerator`. The `y` parameter, which represents the target labels, is set to `None`, indicating that the target labels are inferred from the directory structure of the training data. The `batch_size` specifies the number of samples per gradient update, while `epochs` defines the number of training epochs (iterations over the entire dataset). By executing these lines of code, the model undergoes training where it learns to minimize the specified loss function using the Adam optimizer, with the goal of improving its accuracy on the training data over multiple epochs.

III. Results

The testing was done in two different ways. Firstly, using the dataset [2] we created a test subset which we used a static dataset. Secondly, using the webcam of a computer, we streamed a live video and tested it on a dynamic dataset.

A. Testing Results on Static Dataset

At first, we initialize an instance of `ImageDataGenerator` named `test_datagen` with rescaling specified as the only transformation, aimed at normalizing the pixel values of images within the range $[0, 1]$. Subsequently, the `flow_from_directory` method is invoked on `test_datagen` to generate batches of image data from a designated directory containing test images. This method involves setting the target size of images to be resized to 80×80 pixels and defining the batch size for image processing based on the variable `batchsize`. Additionally, the class mode is set as 'categorical', indicating that labels are provided as one-hot encoded vectors representing class labels of the images. Overall, this process prepares the test data for evaluation by generating preprocessed image batches suitable for inference using a CNN model.The accuracy of the model was found to be 88.181%. The classification report for the testing data is shown in the table:

Classification Report				
binary	Precision	Recall	f1-Score	Support
0	0.82	0.98	0.89	973
1	0.98	0.78	0.87	973
accuracy			0.88	1946
macro avg	0.90	0.88	0.88	1946
weighted avg	0.90	0.88	0.88	1946

B. Testing Results on Dynamic Dataset

The testing procedure of Dynamic Dataset is explained below.

1) Initialization and Setup:

- The sound mixer is initialized using the `mixer.init()` function, and an alarm sound file (`alarm.wav`) is loaded with the `mixer.Sound()` function from the `pygame` library.

2) Webcam Connection:

- A connection to the webcam is established using the `cv2.VideoCapture()` function, enabling real-time frame capture.

3) Face and Eye Detection:

- Within an indefinite while loop, each frame from the webcam feed is converted to grayscale for efficient face and eye detection.
- Haar cascades are utilized to detect faces and eyes within the grayscale frame.

4) Annotation of Detected Features:

- Rectangles are drawn around detected faces and eyes on the original color frame, iterating through the detected regions.

5) Preprocessing and Model Prediction:

- Detected eye regions undergo preprocessing steps, including resizing, normalization, and reshaping, to prepare them for input into the pre-trained CNN model (`model`) for drowsiness prediction.

6) Display and Prediction:

- Based on the model's prediction, the script displays whether the eyes are closed or open on the frame.
- An alarm sound is triggered if the score surpasses a predefined threshold (20), indicating prolonged eye closure.

7) Termination and Resource Release:

- The while loop continues until the 'q' key is pressed, signaling the termination of the drowsiness detection process.
- Upon termination, the webcam feed is released, and all OpenCV windows are closed to release system resources.

We tested the trained model using the live video feed of two different subjects. We intuitively checked the accuracy of our model by extracting frames from the live video cam recording of the two subjects. Since the dataset is dynamic and unlabeled, we could not quantize the accuracy of the dynamic testing.

Authors and Affiliations

R. Nanzeeba and K.M. Tazwar worked on the dataset preprocessing [2], model training and optimisation. K.M. Tazwar extracted the trainable data from FL3D dataset and merged that data as a part of the data preparation to bring nighttime factor in the training process and data was filtered by P. Paul and R. Nanzeeba. P. Paul and S.Hasan worked on the webcam live broadcast, processing of frames extracted from live broadcast and make predictions based on the trained model by R. Nanzeeba and K.M. Tazwar. From the dataset [2] some static subsets were made and also used to train the accuracy of the model by P.Paul and S.Hasan.

IV. Conclusion & Future Works

The authors added some extra layers on the InceptionV3 [3] base model. The training and testing dataset were 4000 randomly selected images from the dataset [2]. For this reason, the accuracy and versatility of the trained model is sometimes compromised. Using more images from the dataset [2] or taking a more robust dataset, the model can be made more accurate. Moreover, if more layers could be added to the model, accuracy can be improved to a great extent.

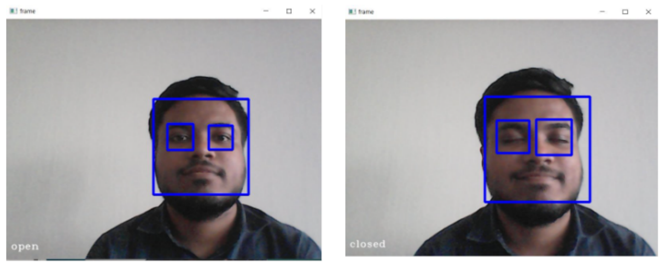

Fig. 2. Testing on Subject-1 at low-light condition

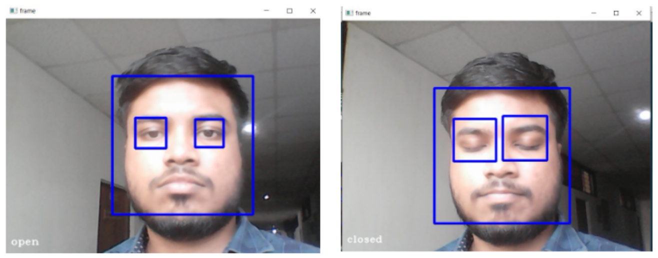

Fig. 3. Testing on Subject-1 at bright daylight condition

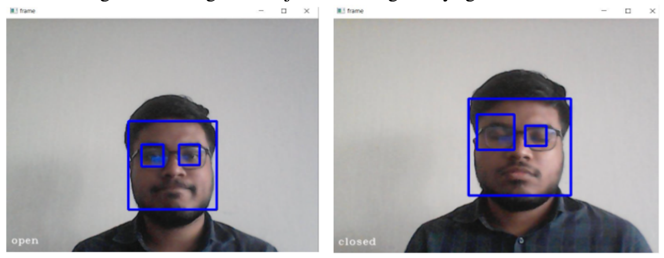

Fig. 4. Testing on Subject-1 with spectacles

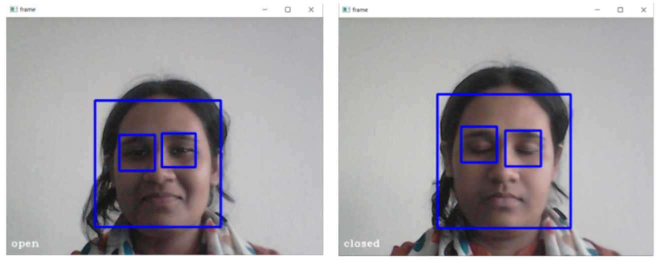

Fig. 5. Testing on Subject-2 at low-light condition

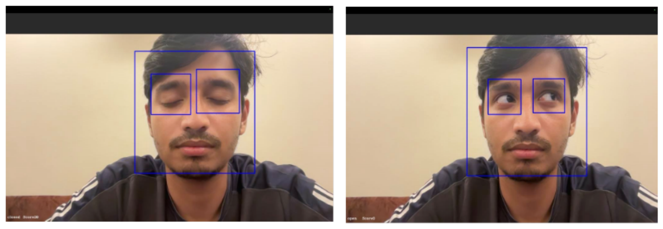
Fig. 6. Testing on Subject-3 at normal room lighting condition

References

[1] Dhaka Tribune article titled “25,000 lives lost in road accidents every year” published on 22 Oct,2023. Available online: https://www.dhakatribune.com/bangladesh/328677/25-000-lives-lost-in-road-accidents-every-year

[2] Fusek, R. (2018). Pupil localization using geodesic distance. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 11241 LNCS, 433-444. doi: 10.1007/978-3-030-03801-438.

[3] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z. (2016). Rethinking the inception architecture for computer vision. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

[4] 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