DRIVER DROWSINESS DETECTION SYSTEM USING CNN

SURIYAVARMAN S

M.E(CSE WITH SPLN IN BIG DATA ANALYTICS),

COLLEGE OF ENGINEERING,

GUINDY

***Abstract*—an enormous number of people drive on the highway around the clock. Taxi drivers, bus drivers, truck drivers and people who travel long-distance at a stretch suffer from lack of sleep. This creates a very dangerous situation for the drivers when feeling sleepy. Drowsiness detection is a safety system that helps to prevent accidents that are caused by drivers who fell asleep while driving. Most of the accidents happen due to the drowsiness of the driver. This system is built using Python, OpenCV, and Keras to prevent the accidents and alerts the drivers when they feel sleepy. A convolutional neural network model was developed using various internal layers and trained on a dataset. The results produced around 90% accuracy.**

Keywords— ***driver, sleepy, Drowsiness, OpenCV, Keras, convolutional neural network***

I. INTRODUCTION

The dependency on vehicles for mostly transportation is on the rise which causes expanding numbers of vehicles in the developing and developed countries. On account of this, road accidents on highways and roads are over the threshold. Human related flaws are accused of a total 88 percent of all accidents in recent years. About 60 percent of adult drivers or about 168 million people have driven a vehicle while feeling drowsy in the past year.

Although the automobile has changed people's lifestyle and improved the convenience of conducting daily tasks, it is also associated with numerous pernicious effects, such as traffic accidents. The National Highway Traffic Safety Administration approximates that every year about 100,000 police-reported, drowsy-driving crashes result in nearly 800 fatalities and about 50,000 injuries.

The term "drowsy" is defined as the sleepiness or an inclination to fall asleep at any point of time. There are three stages of sleep which are awake, non-rapid eye movement sleep (NREM) and rapid eye movement sleep (REM). Again, the second stage NREM is classified into three more categories and the first category is the transition from awake to sleep known as drowsy and this is where people enter into micro sleep or fall into sleep unintentionally.

Another essential part of road accident is the extensive driving without any interval that means the drivers feel drowsy or sleepy because of driving for a long duration. Therefore, to ensure safety, it is essential to analyse the key factors of drowsiness and to find possible solutions for lowering the accident rates. Additionally, these occurrences are likely to happen between the 12 AM to 7 AM or during mid-afternoon when the driver may run off the road, drive at a rapid pace, driving alone in the car or show no indication of braking which leads to a conclusion that drowsiness is also a factor which can lead into a road accident. After examining, it is found out that a feasible model can be accomplished. The main features of the detection system are stated as follows:

a) A drowsiness detection system is developed. It will detect that a person's eyes are closed for a few seconds.

b) When drowsiness is detected, the driver is alerted by the system using an alarm.

c) CNN model is used to train our prototype for predicting drowsiness.

d) Closed, open, yawn, not yawn categories are extracted.

The challenge of human face detection is simpler for available static images, but it is even more complex when used in combination with computer vision. Section II discusses the previous studies on drowsiness detection using CNN techniques. The dataset and setup are presented in section III and the model development is also described in there. Section IV describes the proposed system and implementation and how those are needed in our proposed model. Section V displays the results and outputs of the developed system.

II. LITERATURE REVIEW

Driver Drowsiness Detection by Applying Deep Learning Techniques to Sequences of Images:

The paper was proposed by Magán, E.; Sesmero, M.P.; Alonso-Weber, J.M.; Sanchis, (2022) on the development of an ADAS (advanced driving assistance system) focused on driver drowsiness detection, whose objective is to alert drivers of their drowsy state to avoid road traffic accidents. The proposed technique uses a recurrent and convolutional neural network and deep learning techniques to extract numeric features from images, which are introduced into a fuzzy logic-based system afterwards. The accuracy obtained by system is around 65% accuracy over training data, and 60% accuracy on test data. The proposal presented in this work is promising and can be considered a solid baseline for future works.

Real-Time Driver-Drowsiness Detection System Using Facial Features:

The paper was proposed by Wanghua Dengi And Ruoxue Wu. Owing to the shortcomings of previous drowsiness detection algorithms, they introduced a new face-tracking algorithm to improve the tracking accuracy. Additionally, they designed a new detection system for facial regions based on 68 key points. Then they used these facial regions to evaluate the drivers' state. By combining the features of the eyes and mouth, DriCare can alert the drivers using a fatigue warning. The experimental results showed that DriCare achieved around 92% accuracy.

An Algorithmic Approach to Driver Drowsiness Detection for Ensuring Safety in an Autonomous Car:

The Algorithm was developed by Md. Motaharul Islam(2020) et al., For ensuring safety in an Autonomous Car. They have introduced an algorithmic approach in which the proposed system can locate a safe parking space after the determination of drowsiness and can also send a distress message to the authority briefing about the situation while reaching at the safe parking space to assure safety from the incompetent, drowsy driver. They have implemented a prototype model by using raspberry pi to detect drowsiness and trained image data using CNN to build an autonomous vehicle prototype which is used for finding the path to the nearest SPS. In the experimental prototype approach, we have gained 91% accuracy to find paths to SPS. In the future, the model can be improved by using night vision cameras to solve the problem of detecting drowsiness in low light and improve the accuracy at night.

CNN Based Driver Drowsiness Detection System Using Emotion Analysis:

The system was developed by H. Varun Chand and J. Karthikeyan. This paper proposes a novel model of detecting the driver drowsiness using the Convolution Neural Networks (CNN) implemented by the emotion analysis. The emotion analysis, in this model, analyses the driver's frame of mind which identifies the motivating factors for different driving patterns. These driving patterns were analysed based on the acceleration system, speed of the vehicle, Revolutions per Minute (RPM), facial recognition of the driver. The facial pattern of the driver is treated with 2D Convolution Neural Network (CNN) to detect the behavior and driver's emotion. Distraction detection duration can be added in the future works to increase the level of accuracy.

Driver drowsiness detection system based on feature representation learning using various deep networks:

Sanghyuk Park, Fei Pan, Sunghun Kang and Chang D. Yoo developed the detection system using various deep networks. This paper proposes a deep architecture referred to as deep drowsiness detection (DDD) network for learning effective facial points and detecting drowsiness given a RGB input video of a driver. The DDD network consists of three different deep networks for attaining global robustness to background and environmental variations and learning local facial movements and head gestures important for reliable detection. 73.06% detection accuracy was achieved on NTHU-drowsy driver detection benchmark dataset. Model ensemble strategies can be fused to improve the performance.

A comparative analysis on driver drowsiness detection using CNN:

The analysis was performed by V. Naren Thiruvalar and E. Vimal (2021). Two different algorithms based on Convolution Neural Network (CNN) were applied and the results were compared respectively. "Highway Hypnosis" is a serious issue to be addressed while driving especially on highways. CNN is used since it is very effective in analyzing images and videos. In this project, a live video feed is used to detect drowsiness by suitable algorithms. the drowsiness detection based on Viola-Jones and PERCLOS algorithms using CNN were tested and satisfactory results were obtained. Based on accuracy aspect, Viola-Jones algorithm outnumbered the other. But, both algorithms have their own significance in different aspects. Ultimately these algorithms are used to alert the drivers in case of emergency situations so that accidents could be anticipated and avoided.

Driver Drowsiness Detection using Deep Learning:

Yeresime Suresh developed a drowsiness detection system using deep learning. The goal of this article is to create a Sleepiness Detection System that detects whether a driver's eyes are closed for a few seconds, and then alerts the driver via an alarm. In this paper, deep learning is used to suggest a new frame that classifies the driver's eye condition, i.e. open or closed. When a driver is determined to be sleepy, the proposed system generates a beep on reaching a certain saturation point of the drowsiness measure. The proposed detection system is evaluated on a large part of MRL eye dataset consisting of 48000 images and it shows an accuracy of 86.05% using CNN model. The buzzer sound producing function is built, and if sleepiness is detected, the driver will be alerted. The system can make use of transfer learning to increase the system's performance in the future; also optimization algorithms like genetic algorithm can be used for enhancing the accuracy.

Driver Drowsiness Detection Model Using Convolutional Neural Networks Techniques for Android Application:

The model was proposed by Rateb Jabbar, Mohammed Shinoy, Mohamed Kharbeche, Khalifa Al-Khalifa, Moez Krichen, and Kamel Barkaoui for Android Application. In this paper, accuracy was attained by utilizing facial regional points which are detected by the camera and given as input to Convolutional Neural Network (CNN) to classify drowsiness. The achievement with this system is the potentiality to provide a lightweight alternative to heavier classification models with more than 88% for the category without glasses, more than 85% for the category night without glasses. On standard average, more than 83% of accuracy was achieved in all attributes. Furthermore, as for model size, complexity and storage, there is a marked reduction in the new proposed model in comparison to the benchmark model where the maximum size is 75 KB. There is still room for performance enhancement and better facial feature detection even in bad lighting conditions.

III. DATASET AND SETUP

DATASET:

The dataset was used from Kaggle repository with import function. The dataset consists of four categories. Each category holds numerous driver images captured while driving. The images are provided as inputs to the system for training and prediction.

- Closed_eyes - having 726 pictures
- Open_eyes - having 726 pictures
- Yawn - having 725 pictures
- no_yawn - having 723 pictures

These categories are considered as the target class while detecting the drowsiness of the driver. These target classes are the testing images used later in the system.

OPENCV:

OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common architecture for computer vision applications and to accelerate the use of machine perception in the commercial products. The library consists of more than 2500 optimized algorithms, which comprises a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms.

KERAS:

Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code. It also allows use of distributed training of deep-learning models on clusters of Graphics processing units (GPU) and tensor processing units (TPU).

CNN:

Convolutional Neural Network or CNN is a type of artificial neural network, which is widely used for image/object recognition and classification. Deep Learning thus recognizes objects in an image by using a CNN.

CNNs are playing a major role in diverse tasks/functions like image processing problems, computer vision tasks like localization and segmentation, video analysis, to recognize obstacles in self-driving cars, as well as speech recognition in natural language processing. As CNNs are playing a significant role in these fast-growing and emerging areas, they are very popular in Deep Learning. A CNN basically consists of an input layer, an output layer and a hidden layer which can have multiple layers. A convolution operation is performed on these layers using a filter that performs 2D matrix multiplication on the layer and filter.

MODEL:

The convolutional neural network is developed. A CNN can be expressed as a Sequential model because each layer has exactly one input and output and is aggregated together to form the entire neural network. CNN makes use of filters which are also known as kernels, to detect what features, such as edges, are present throughout an image. A convolution layer converts all the pixels in the image receptive field into a single field value. The most common kind of convolution is the 2D convolution layer and is often called as conv2D.

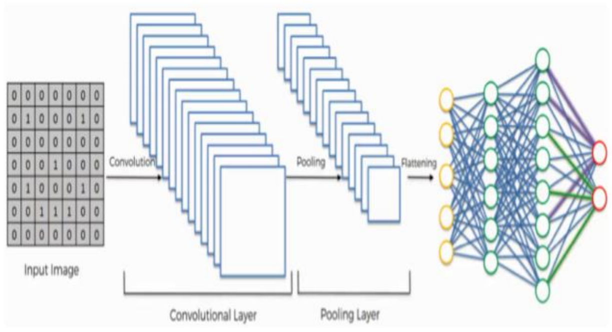


Fig. 1: CNN Architecture

a) CONV2D:

Keras Conv2D is a 2D Convolution Layer, this layer creates a convolution kernel that is added with each layer input which helps produce a tensor of outputs.

b) KERNEL**:**

In image processing kernel is a convolution matrix which can be used for blurring, sharpening, embossing, edge detection, and more by doing a convolution between a kernel and an image.

c) KERNEL_SIZE:

This parameter is used to determine the dimensions of the kernel. The common dimensions are 1×1, 3×3, 5×5, and 7×7 which can be passed as (1, 1), (3, 3), (5, 5), or (7, 7) tuples. It is either an integer or tuple/list of 2 integers, specifying the height and width of the 2D convolution window. This parameter must always be an odd integer.

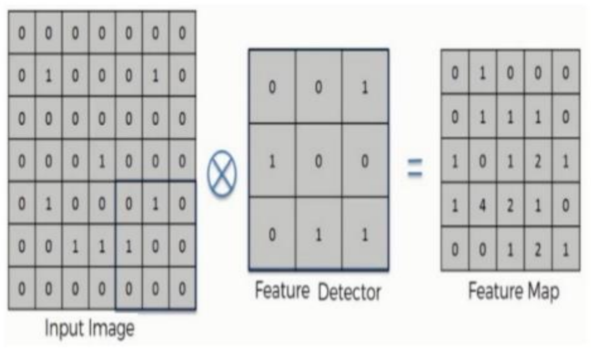


Fig. 2: Convolution operation

d) ACTIVATION FUNCTION:

The usage of ReLU activation function helps to prevent the exponential growth in the computation needed to operate the neural network. If the CNN scales in size, the computational cost of aggregating extra ReLUs increases linearly. The softmax activation function is implemented to transform the raw outputs generated by the neural network into a vector of probabilities, essentially a probability distribution over the input variables (classes).

e) MAXPOOLING:

Max pooling is one of the pooling operations that selects the maximum element from the region of the feature map enclosed by the filter function. A feature map holding the most prominent features of the previous feature map will be produced as the output after max-pooling layer.

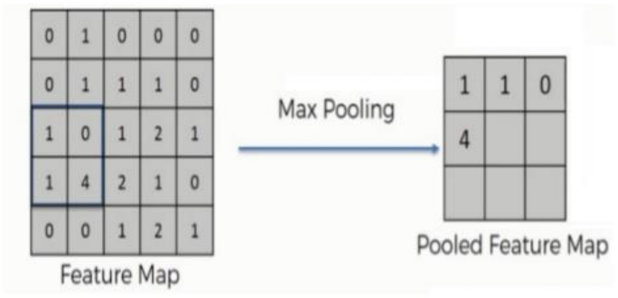


Fig. 3: Max Pooling

f) FLATTEN:

Flatten is a particular method of reshaping operation by means of which all of the axes are smoothened or squashed with each other. To flatten a tensor, at least two axes are needed.

Flatten layers are used when there is a multidimensional output produced and to make it linear. This converted linear output is passed onto a Dense layer.

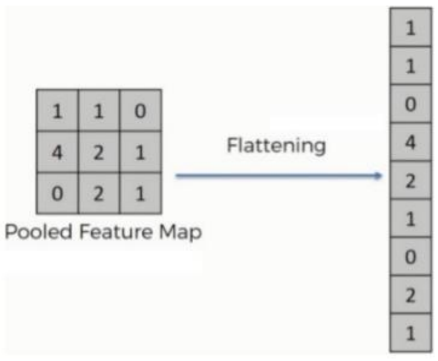


Fig. 4: Flattening

g) DENSE:

A dense network is in which the total number of links of each node almost equal to the maximal number of nodes. Each node is linked to almost all other nodes. The total connected scenario where exactly each node is linked to each other node is called a completely connected network.

h) LOSS FUNCTION:

The cross-entropy function, through its logarithm function, allows the neural network to check for such small errors and works to eliminate them. Cross-entropy loss is defined as the difference between our predicted and target output. The Adam optimizer has the best accuracy of 99.2% in increasing the CNN capability in both classification and segmentation.

The required libraries are installed using pip install command. The python packages imported are numpy, pandas, os, and cv2. Using keras load_model and img_to_array were also imported. The dataset is uploaded onto the Google drive. The drive is mounted and connected using basic connections. The categories in the dataset are considered as labels.

IV. PROPOSED SYSTEM

The proposed "Driver's Drowsiness Detection System" can prevent accidents that are caused by drivers who fell asleep while driving. In this project, OpenCV technology is used for gathering the images from webcam and feed them into a Deep Learning model which will classify whether the person's eyes are 'Open' or 'Closed' or 'Yawn' or 'Not-yawn'. The code we will be written in Python.

Labels = ['Closed', 'no_yawn', 'Open', 'yawn']

A random image is visualized from the dataset using matplotlib.pyplot function. The "imread" retrieves the required image when the correct path is specified.

```
<matplotlib.image.AxesImage at 0x245f12e0310>
```

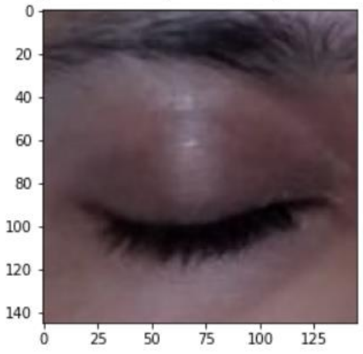


Fig. 5: Closed eye image

The images are converted into arrays and Background in the image is unnecessary. Only the face is processed into an image array. The "xml cascade files" folder is created and required xml files that are needed to detect objects from the image are added to the folder. In our system, face and eyes of the person are detected as objects.

Functions are written for yawn and not yawn as one category and closed and closed eye as the other. The image size is provided a default value - '145'. The images are extracted from the folders and given into the functions as individual inputs. These inputs are processed and resized according to the default specified value. Image arrays are generated and each label is assigned an integer value. Each array consists of the target integer value. The arrays are considered as the target class.

[Yawn – 0, Not yawn -1, Opened eye – 2, Closed eye – 3]

The image arrays are converted into numpy arrays. Numpy is a python package used for scientific computing purposes. A numpy array is a structure with different grid of values, all of the same type, and these values are indexed by a tuple of nonnegative integers.

The features and the labels are partitioned from each other and stored in discrete variables. The array is reshaped using the default image size values. The LabelBinarizer package is imported from the sklearn library. Scikit-learn's LabelBinarizer converts input image labels into binary labels. The train_test_split is used to Split arrays or matrices into random train and test subsets. The test size is specified as 0.30. The length of test images is 578.

train_generator = ImageDataGenerator(rescale=1/255, zoom _range=0.2, horizontal_flip=True, rotation_range=30)

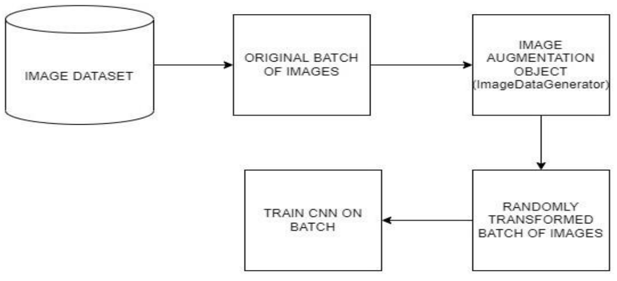


IMAGE DATASET
ORIGINAL BATCH OF IMAGES
IMAGE AUGMENTATION OBJECT (ImageDataGenerator)
RANDOMLY TRANSFORMED BATCH OF IMAGES
TRAIN CNN ON BATCH

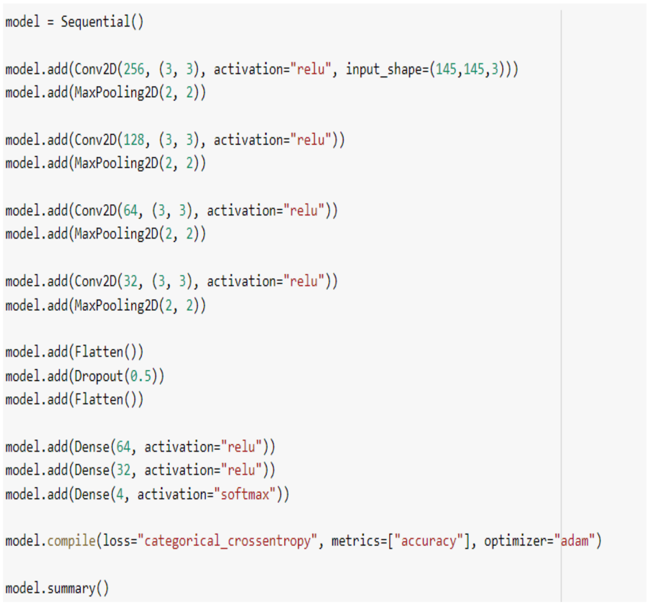
Fig. 6: Data Augmentation

Data augmentation is carried out to artificially increasing the amount of images by generating new image points from existing image dataset. Keras ImageDataGenerator is imported for retrieving the input of the original data and further, it makes the transformation of this image data on a random basis and gives the output resultant containing only the images that are newly transformed.

V. RESULTS

CNN:

The CNN model is developed using various layers. Different parameters are also generated as trainable and non-trainable types.



```
model = Sequential()

model.add(Conv2D(256, (3, 3), activation="relu", input_shape=(145,145,3)))
model.add(MaxPooling2D(2, 2))

model.add(Conv2D(128, (3, 3), activation="relu"))
model.add(MaxPooling2D(2, 2))

model.add(Conv2D(64, (3, 3), activation="relu"))
model.add(MaxPooling2D(2, 2))

model.add(Conv2D(32, (3, 3), activation="relu"))
model.add(MaxPooling2D(2, 2))

model.add(Flatten())
model.add(Dropout(0.5))
model.add(Flatten())

model.add(Dense(64, activation="relu"))
model.add(Dense(32, activation="relu"))
model.add(Dense(4, activation="softmax"))

model.compile(loss="categorical_crossentropy", metrics=["accuracy"], optimizer="adam")

model.summary()
```

Fig. 7: CNN model

Model fitting is a measure of how well a model generalizes the training data. The model is provided as the first parameter. Epochs, Validation data are the other parameters. Epoch means training the model with all the training images for one complete cycle. The epoch size is provided as 50. The accuracy and loss are calculated and plotted using matplotlib function. The accuracy is classified as: training and validation accuracy. The validation accuracy is considered to evaluate the model's gradual performance.

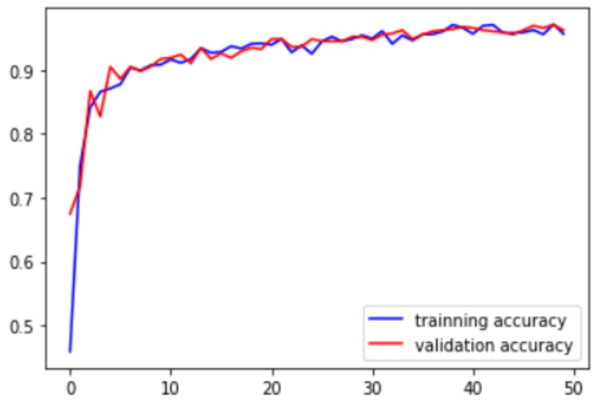


Fig. 8: Accuracy

The cross-entropy loss function is used to calculate the loss in the model. Cross-entropy loss, or log loss, measures the performance of the classification model developed whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability separates from the actual label. The loss is categorized as: Training and validation loss.

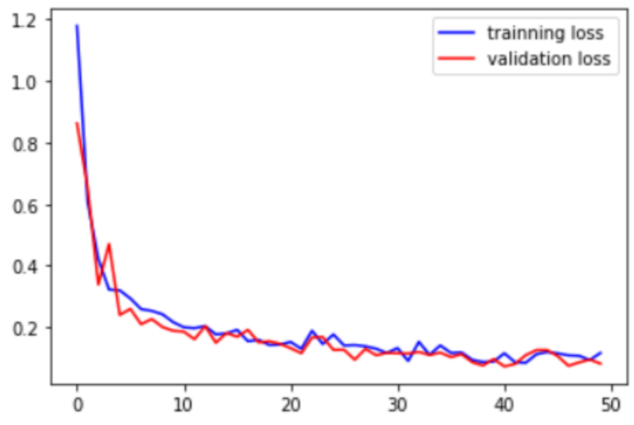
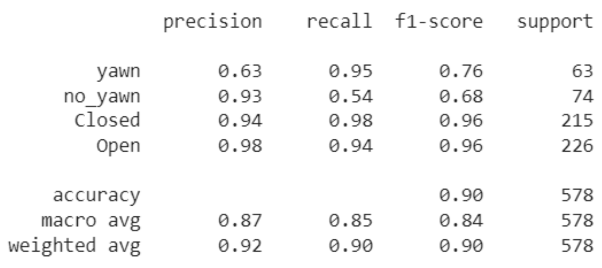


Fig. 9: Loss

The model is saved for future purposes in the same repository. An integer array is generated using model.predict function with train and test split sets. This array consists of the label values: 0,1,2,3. These predicted values are compared with actual values to generate the required classification report. A classification report is a performance evaluation metric. It is derived to display the precision, recall, F1 Score, and support of any trained classification model.



```
              precision    recall  f1-score   support

        yawn       0.63      0.95      0.76        63
     no_yawn       0.93      0.54      0.68        74
      Closed       0.94      0.98      0.96       215
        Open       0.98      0.94      0.96       226

    accuracy                           0.90       578
   macro avg       0.87      0.85      0.84       578
weighted avg       0.92      0.90      0.90       578
```

Fig. 10: Classification report

A prediction function is created for randomly predicting the class label of the images. The images are retrieved from the dataset folders and given as input to the function. The function using the above model generates the required output integer value. The models folder contains our model file "cnn.h5" which was trained on convolutional neural networks with the images. "drowsiness.py" file contains the python program through which classification model was built by training on our dataset. To start the real-time detection procedure, this file is run.

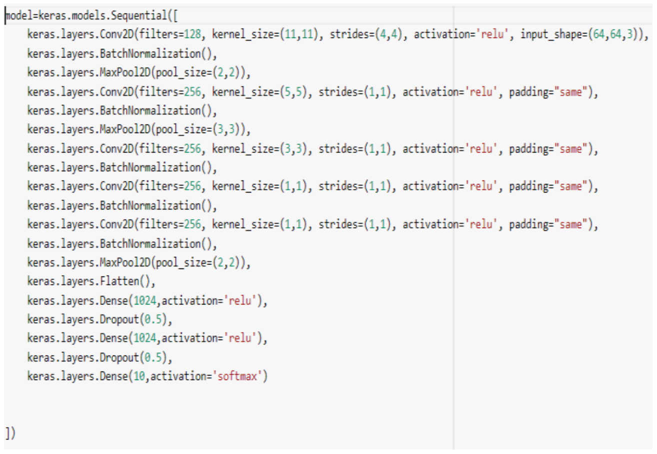
A neural network is trained for one cycle during an epoch using all of the training data. We only use each piece of information once within an epoch. In general, too many epochs might lead to an over-fitting of your model to the training set. It implies that your model memorises the data rather than learning it. To determine if it over-fits or not, you must determine the correctness of the validation data for each epoch or maybe iteration.

ALEXNET CNN:

For any object-detection task, AlexNet is a top architecture, and it may find extensive use in computer vision-related artificial intelligence issues. In the future, AlexNet might be used for picture jobs more frequently than CNNs.

A model as sophisticated as AlexNet is capable of obtaining high accuracy on very difficult datasets. However, AlexNet's performance will be significantly hampered if any of the convolutional layers are removed. For any object-detection task, AlexNet is a top architecture, and it may find extensive use in computer vision-related artificial intelligence issues. In the future, AlexNet might be used for picture jobs more frequently than CNNs.

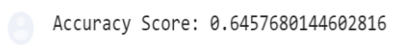
The convolutional neural network design known as AlexNet is traditional. Convolutions, max pooling, and dense layers make up its fundamental building elements. The model is fitted over two GPUs using grouped convolutions. The AlexNet CNN model is built for the same dataset. There are successive neural network layers that are stacked against one another within the model by using the Keras Sequential API. Training the custom AlexNet network is very simple with the Keras module enabled through TensorFlow.



```
model=keras.models.Sequential([
    keras.layers.Conv2D(filters=128, kernel_size=(11,11), strides=(4,4), activation='relu', input_shape=(64,64,3)),
    keras.layers.BatchNormalization(),
    keras.layers.MaxPool2D(pool_size=(2,2)),
    keras.layers.Conv2D(filters=256, kernel_size=(5,5), strides=(1,1), activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.MaxPool2D(pool_size=(3,3)),
    keras.layers.Conv2D(filters=256, kernel_size=(3,3), strides=(1,1), activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.Conv2D(filters=256, kernel_size=(1,1), strides=(1,1), activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.Conv2D(filters=256, kernel_size=(1,1), strides=(1,1), activation='relu', padding="same"),
    keras.layers.BatchNormalization(),
    keras.layers.MaxPool2D(pool_size=(2,2)),
    keras.layers.Flatten(),
    keras.layers.Dense(1024,activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(1024,activation='relu'),
    keras.layers.Dropout(0.5),
    keras.layers.Dense(10,activation='softmax')


])
```

Fig. 11: AlexNet CNN model



```
Accuracy Score: 0.6457680144602816
```

Fig. 12: Accuracy score

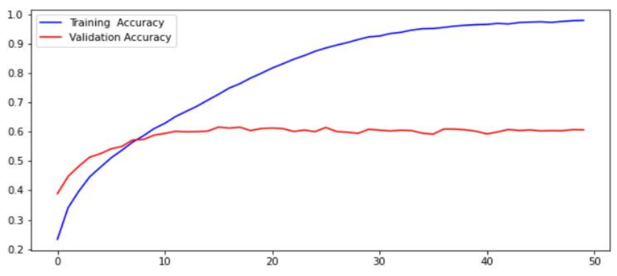


Fig. 13: Accuracy

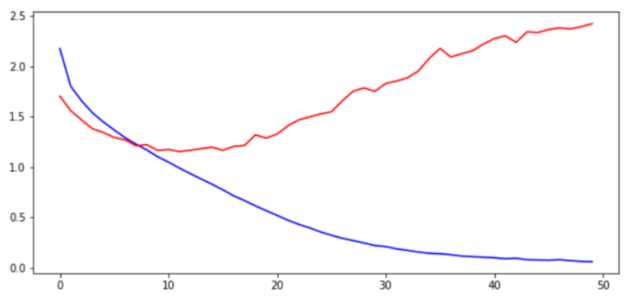


Fig. 14: Loss

By adding an additional layer that operates on the inputs from the previous layer, batch normalisation is a technique that reduces the impact of unstable gradients within a neural network. The procedures first standardise and normalise the input values before scaling and shifting operations modify the input values.

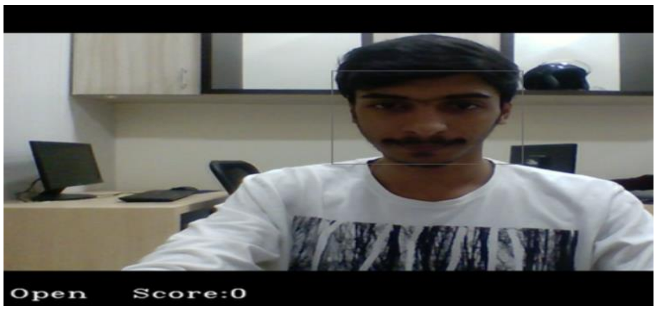
EPOCH TRAINING:

	Model	No. Of. Epochs	Accuracy
1	CNN	10	84.3
2	CNN	30	87
3	CNN	50	90
4	AlexNet CNN	50	64.57

Take Image as Input from a Camera. With a webcam, images are retrieved as input. There is an infinite loop that will capture each frame with the permission to access the webcam.

A capture function is developed and cv2.VideoCapture(0) method is used to access the camera and set the capture object (cap). cap.read() will read each frame and we store the image in a separate frame variable. The XML files such as frontface.xml, righteyesplits.xml and lefteyesplits.xml are imported. These files are used to detect the required objects.

This will be fed into the developed CNN classifier model which will predict if eyes are open or closed. The consecutive frames are checked continuously and eyes prediction is carried out at each frame. If eyes are predicted to be closed for 10 consecutive frames, alarm sound is called to start. Alarm sound is played using the playsound module.



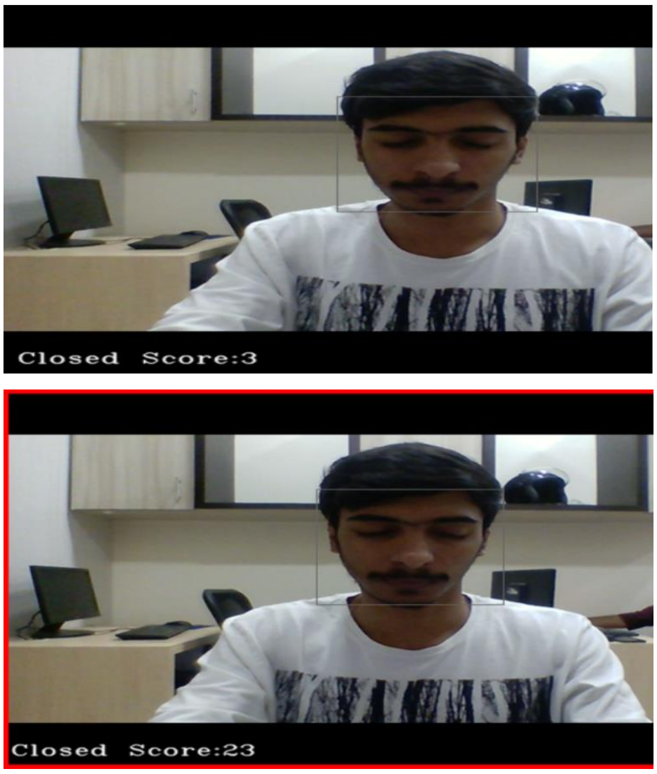


Fig. 15: Captured frames

VI. CONCLUSION AND FUTURE WORKS

An approach for detecting driver drowsiness based on eye condition and yawning is presented in this proposed system. To extract required features from the images, a stacked convolutional neural network is created and employed in the learning phase. The system preprocesses the images and predicts the drowsiness state using convolutional neural network.

The alarm buzzer sound generating function is in-built, and if sleepiness is detected, the driver will be alerted. The accuracy of the proposed system is around 90%. This detection system successfully recognizes the drowsy condition of the driver and alerts with an alarm. Transfer learning methods can be implemented to increase the system's performance in the future and also optimization algorithms like genetic algorithm can be used for enhancing the accuracy.

Better facial feature detection can be achieved even in bad lighting conditions by using advanced pre-processing techniques such as Pixel brightness transformations/ Brightness corrections.

REFERENCES

[1] Magán, E.; Sesmero, M.P.; Alonso-Weber, J.M.; Sanchis, "On Driver Drowsiness Detection by Applying Deep Learning Techniques to Sequences of Images",2022, Applied Sciences 12(3) Journals, Computer Science and Engineering Department, Universidad Carlos III de Madrid.

[2] Wanghua Dengi, Ruoxue Wu, "On Real-Time Driver-Drowsiness Detection System Using Facial Features", 2021 IEEE Gobal Conference on Consumer Electronics, 2021.

[3] Md. Motaharul Islam, "An Algorithmic Approach to Driver Drowsiness Detection for Ensuring Safety in an Autonomous Car", 2020 IEEE Region 10 Symposium (TENSYMP), 5-7 June 2020, Dhaka, Bangladesh

[4] Park,; Pan; Kang; Yoo, " On Driver drowsiness detection system based on feature representation learning using various Deep networks", ACCV 2020: Computer Vision – ACCV 2016 Workshops pp 154–164

[5] Varun Chand and Karthikeyan, "On CNN based Drowsiness Detection System using emotinal analysis", Intelligent Automation & Soft Computing 2022, 31(2), 717-728.

[6] V.Naren Thiruvalara, E. Vimala, " A comparative analysis on driver drowsiness detection using CNN", Int. J. Nonlinear Anal. Appl. Volume 12, Special Issue, Winter and Spring 2021, 1835–1843.

[7] Yeresime Suresh, "Driver drowsiness Detection using Deep learning", 2021 2nd International Conference on Smart Electronics and Communication (ICOSEC).

[8] Prabhu; Chandana; Sunny, " On An Enhanced Driver drowsiness detection System using Transfer Learning", 2021 5th International Conference on Electronics, Communication and Aerospace Technology (ICECA).

[9] Park; Pan; Yoo, "On Drowsy driver detection based on feature representation learning using various deep networks", ACCV Workshops, 2020.

[10] Ngxande; Tapamo; Burke, "On Drowsiness detection using behaviourial measures: A review of stae-of-art techniques", 2017 Pattern Recognition Association of South Africa and Robotics and Mechatronics (PRASA-RobMech).

[11] Parth; Pavesha; Sabat; More, " Deep learning based driver drowsiness detection", 2022 International Conference on Applied Artificial Intelligence and Computing (ICAAIC).

[12] Farahnakian; Leoste, " On Driver drowsiness detection system using Deep CNN", 2021 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME).

[13] Dwivedi; Biswaranjan; Sethi, "On Drowsy driver detection using representation learning", 2014 IEEE International Advance Computing Conference (IACC)