Project Title:- Face Recognition and Drowsiness Detection

- **Lalit Ahirrao**
- **AniketGajmal**
- **Rushikesh Pawar**
- **Prasad Ghegade**
- **Samarth Gangurde**

INTRODUCTION:-

One of the main reasons for untimely deaths today is road accidents. Most of the time the drivers would lose their alertness and meet with unfortunate accidents. This loss of the state of alertness is due to fatigue and drowsiness of the driver. This situation becomes very dangerous when the driver is alone. The ultimate reason for the loss of the state of alertness is accidental micro-sleeps (i.e. temporary lapse of consciousness which occurs when a driver is drowsy and is fatigued). Drowsiness or fatigue is one of the main reasons for low road safety and some severe injuries, economic loss, and even deaths. Collectively, these situations increase the risk of road accidents. Using a computer for automatic fatigue detection, several misfortunes can be avoided. The drowsiness detection systems continuously analyze the drivers condition and warn before any unfortunate situation arises.

Due to the accidents being caused due to the fatigue state of the drivers, several methods have been developed to detect the drivers drowsiness state and warn accordingly. Each method has its advantages as well as disadvantages. There have been some great works in this field but we can have some space for future improvements. Our project aim is to identify driver drowsiness while addressing the issue of late warning due to analysis in discrete time periods by existing solutions. This project shows the design and implementation of a Deep Learning based model for eye state (open/close) classification, and the integration of the above ideas into a driver drowsiness detection system.

FACTS REGARDING DROWSINESS ACCIDENTS:-

Drowsy driving is a major contributor to motor vehicle collisions. According to the National Highway Traffic Safety Administration (NHTSA), in 2017 drowsy driving led to at least 91,000 crashes, resulting in roughly 50,000 injuries and 800 deaths3.This data likely underestimates the impact of drowsy driving because it's often impossible to definitively determine whether drowsy driving caused an accident, especially after fatal crashes.In light of this, other studies calculate that drowsy driving causes up to 6,000 deadly crashes every year. Researchers estimate that around 21% of fatal car crashes involve a person driving while drowsy

MRL-DATASET:-

MRL Eye Dataset is the large-scale dataset of human eye images. This dataset contains infrared images in low and high resolution, all captured in various lightning conditions and by different devices. The dataset is suitable for testing several features or trainable classifiers.

Every Image in dataset is numbered as S0012_03054_0_1_0_2_1_01 where as

- in the dataset, we collected the data of 37 different persons (33 men and 4 women)
- image ID; the dataset consists of 84,898 images
- gender [0 - man, 1 - woman]; the dataset contains the information about gender for each image (man, woman)
- glasses [0 - no, 1 - yes]; the information if the eye image contains glasses is also provided for each image (with and without the glasses)
- eye state [0 - closed, 1 - open]; this property contains the information about two eye states (open, close)
- eflections [0 - none, 1 - small, 2 - big]; we annotated three reflection
- states based on the size of reflections (none, small, and big reflections)
- lighting conditions [0 - bad, 1 - good]; each image has two states (bad, good) based on the amount of light during capturing the videos

PREPROCESSING:-

- Resizing image size
- Random shuffling
- Normalizing

RESIZING IMAGE SIZE:-

To choose a good image size it is important to note that a bigger image size will give you better accuracy normally. However all the filters take longer and the memory requirements rise with the image size. Additionally larger sizes yield diminishing improvements. we normally used 224x224

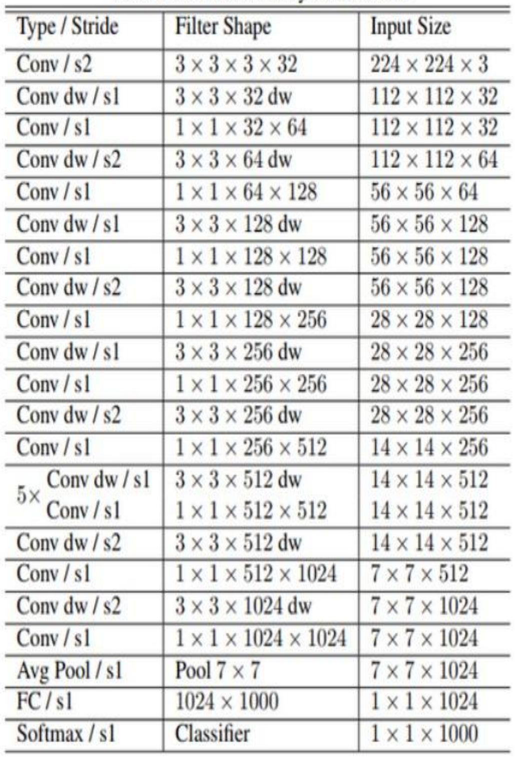
RANDOM SHUFFLING:-

Random shuffling of data is a standard procedure in all image classification problems it can't be an exception its purpose is to break possible biases during data preparation - e.g. putting all the closed_eyes images first and then the open_eyes ones in a closed_eye/opened_eyes classification dataset.

Normalization:-

pixel acts as a variable, range of this variable is expressed in terms of an integer value to describe its brightness. A value of “0” means the pixel is white in color, a value of “255” means the pixel in black in color. All possible values in between are different shades of transformation between white to dark, depending on the magnitude of the integer value. To normalize, the scale is reduced from 0–255 to 0–1. It is done by dividing each pixel valued by 255 and the resultant value will be a value between 0 to 1. This technique of normalization of input image was followed extensively in all DL frameworks.

Table 1. MobileNet Body Architecture



Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
5× Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
5× Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

MobileNet:-

As the name applied, the MobileNet model is designed to be used in mobile applications, and it is TensorFlow’s first mobile computer vision model. MobileNet uses depth wise separable convolutions. It significantly reduces the number of parameters when compared to the network with regular convolutions with the same depth in the nets. This results in lightweight deep neural networks

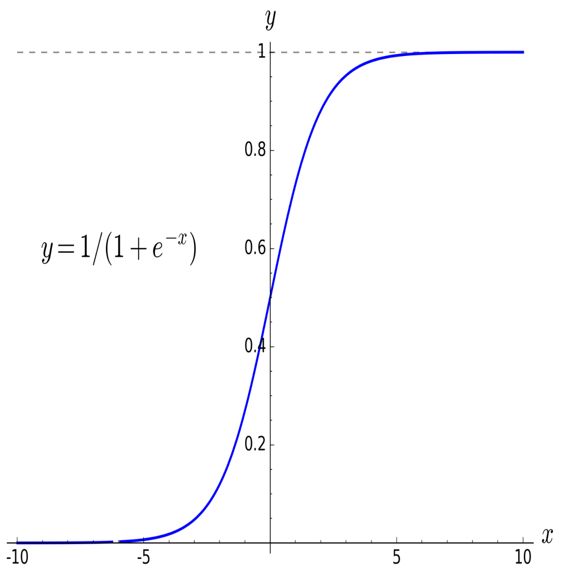
The MobileNet was proposed as a deep learning model by Andrew G. Howard et al of Google Research team in their research work entitled “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications”. This model was proposed as a family of mobile-first computer vision models for TensorFlow, designed to effectively maximize accuracy while being mindful of the restricted resources for an on-device or embedded application. MobileNets are small, low-latency, low-power models parameterized to meet the resource constraints of a variety of use cases. They can be built upon for classification, detection, embedding and segmentation similar to how other popular large scale models.

Sigmoid function:-

The sigmoid function is a mathematical function that has a characteristic that can take any real value and map it to between 0 to 1 shaped like the letter “S”. The sigmoid function is also known as a logistic function.

Y = 1 / 1+e -z

The Sigmoid function S curve



If the value of z goes up to positive infinity, then the predicted value of y will become 1. But if the value of z goes down to negative infinity, then the predicted value of y will become 0.If the outcome of the sigmoid function is greater than 0.5 then you would classify that label to be class 1 or positive class and if it is less than 0.5 then you would classify it to be a negative class or label it as class 0.The Sigmoid function performs the role of an activation function in machine learning which is used to add non-linearity in a machine learning model. Basically, the function determines which value to pass as output and what not to pass as output

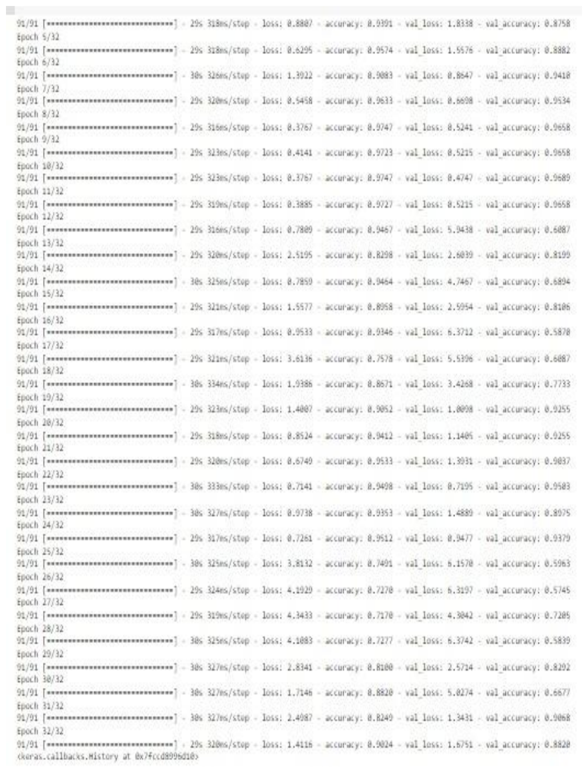
Adam Optimiser:-

It is an optimization algorithm that can be used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data.Adam was presented by Diederik Kingma from OpenAI and Jimmy Ba from the University of Toronto in their 2015 ICLR paper titled Adam: A Method for Stochastic Optimization.The algorithm is called Adam. It is not an acronym and is not written as ADAM.the name Adam is derived from adaptive moment estimation.Stochastic gradient descent maintains a single learning rate for all weight updates and the learning rate does not change during training. A learning rate is maintained for each network weight and separately adapted as learning unfolds.The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients

The authors describe Adam as combining the advantages of two other extensions of stochastic gradient descent.

- Specifically Adaptive Gradient Algorithm that maintains a per-parameter learning rate that improves performance on problems with sparse gradients.
- Root Mean Square Propagation that also maintains per-parameter learning rates that are adapted based on the average of recent magnitudes of the gradients for the weight . This means the algorithm does well on online and non-stationary problems

MODEL TRAINING:-



```
91/91 [==============================] - 29s 318ms/step - loss: 0.8807 - accuracy: 0.9391 - val_loss: 1.8338 - val_accuracy: 0.8758
Epoch 5/32
91/91 [==============================] - 29s 318ms/step - loss: 0.6295 - accuracy: 0.9574 - val_loss: 1.5576 - val_accuracy: 0.8882
Epoch 6/32
91/91 [==============================] - 30s 326ms/step - loss: 1.3922 - accuracy: 0.9083 - val_loss: 0.8647 - val_accuracy: 0.9410
Epoch 7/32
91/91 [==============================] - 29s 320ms/step - loss: 0.5458 - accuracy: 0.9633 - val_loss: 0.6698 - val_accuracy: 0.9534
Epoch 8/32
91/91 [==============================] - 29s 316ms/step - loss: 0.3767 - accuracy: 0.9747 - val_loss: 0.5241 - val_accuracy: 0.9658
Epoch 9/32
91/91 [==============================] - 29s 323ms/step - loss: 0.4141 - accuracy: 0.9723 - val_loss: 0.5215 - val_accuracy: 0.9658
Epoch 10/32
91/91 [==============================] - 29s 323ms/step - loss: 0.3767 - accuracy: 0.9747 - val_loss: 0.4747 - val_accuracy: 0.9689
Epoch 11/32
91/91 [==============================] - 29s 319ms/step - loss: 0.3885 - accuracy: 0.9727 - val_loss: 0.5215 - val_accuracy: 0.9658
Epoch 12/32
91/91 [==============================] - 29s 316ms/step - loss: 0.7809 - accuracy: 0.9467 - val_loss: 5.9438 - val_accuracy: 0.6087
Epoch 13/32
91/91 [==============================] - 29s 320ms/step - loss: 2.5195 - accuracy: 0.8298 - val_loss: 2.6039 - val_accuracy: 0.8199
Epoch 14/32
91/91 [==============================] - 30s 325ms/step - loss: 0.7859 - accuracy: 0.9464 - val_loss: 4.7467 - val_accuracy: 0.6894
Epoch 15/32
91/91 [==============================] - 29s 321ms/step - loss: 1.5577 - accuracy: 0.8958 - val_loss: 2.5954 - val_accuracy: 0.8106
Epoch 16/32
91/91 [==============================] - 29s 317ms/step - loss: 0.9533 - accuracy: 0.9346 - val_loss: 6.3712 - val_accuracy: 0.5870
Epoch 17/32
91/91 [==============================] - 29s 321ms/step - loss: 3.6136 - accuracy: 0.7578 - val_loss: 5.5396 - val_accuracy: 0.6087
Epoch 18/32
91/91 [==============================] - 30s 334ms/step - loss: 1.9386 - accuracy: 0.8671 - val_loss: 3.4268 - val_accuracy: 0.7733
Epoch 19/32
91/91 [==============================] - 29s 323ms/step - loss: 1.4007 - accuracy: 0.9052 - val_loss: 1.0098 - val_accuracy: 0.9255
Epoch 20/32
91/91 [==============================] - 29s 318ms/step - loss: 0.8524 - accuracy: 0.9412 - val_loss: 1.1405 - val_accuracy: 0.9255
Epoch 21/32
91/91 [==============================] - 29s 320ms/step - loss: 0.6749 - accuracy: 0.9533 - val_loss: 1.3931 - val_accuracy: 0.9037
Epoch 22/32
91/91 [==============================] - 30s 333ms/step - loss: 0.7141 - accuracy: 0.9498 - val_loss: 0.7195 - val_accuracy: 0.9503
Epoch 23/32
91/91 [==============================] - 30s 327ms/step - loss: 0.9738 - accuracy: 0.9353 - val_loss: 1.4889 - val_accuracy: 0.8975
Epoch 24/32
91/91 [==============================] - 29s 317ms/step - loss: 0.7261 - accuracy: 0.9512 - val_loss: 0.9477 - val_accuracy: 0.9379
Epoch 25/32
91/91 [==============================] - 30s 325ms/step - loss: 3.8132 - accuracy: 0.7491 - val_loss: 6.1570 - val_accuracy: 0.5963
Epoch 26/32
91/91 [==============================] - 29s 324ms/step - loss: 4.1929 - accuracy: 0.7270 - val_loss: 6.3197 - val_accuracy: 0.5745
Epoch 27/32
91/91 [==============================] - 29s 319ms/step - loss: 4.3433 - accuracy: 0.7170 - val_loss: 4.3042 - val_accuracy: 0.7205
Epoch 28/32
91/91 [==============================] - 30s 325ms/step - loss: 4.1083 - accuracy: 0.7277 - val_loss: 6.3742 - val_accuracy: 0.5839
Epoch 29/32
91/91 [==============================] - 30s 327ms/step - loss: 2.8341 - accuracy: 0.8100 - val_loss: 2.5714 - val_accuracy: 0.8292
Epoch 30/32
91/91 [==============================] - 30s 327ms/step - loss: 1.7146 - accuracy: 0.8820 - val_loss: 5.0274 - val_accuracy: 0.6677
Epoch 31/32
91/91 [==============================] - 30s 327ms/step - loss: 2.4987 - accuracy: 0.8249 - val_loss: 1.3431 - val_accuracy: 0.9068
Epoch 32/32
91/91 [==============================] - 29s 320ms/step - loss: 1.4116 - accuracy: 0.9024 - val_loss: 1.6751 - val_accuracy: 0.8820
<keras.callbacks.History at 0x7fccd8996d10>
```

Here we used 10 epochs (10 iteration) on our model in starting it gave us accuracy of 0.57 which was not that good but after three iterations it increased to more than 89% and in last epoch it gave us accuracy upto 0.88 was pretty good for tring actual prediction and deployment of the model

REFERENCES:-

- **Real-time Driver Drowsiness Detection using Deep Learning (thesai.org)**
- **A Deep Learning Approach To Detect Driver Drowsiness – IJERT**
- **1. OpenCV with Python — OpenCV Guide documentation**
- **MobileNet, MobileNetV2, and MobileNetV3 (keras.io)**
- **MobileNet Convolutional neural network Machine Learning Algorithms | Analytics Vidhya (medium.com)**