DRIVER AWARENESS DETECTION IN AUTONOMOUS CARS

INFO 7374 Neural Networks & Artificial Intelligence

Parag Bhingarkar
Rajeshree Kale
Dhanashri Palodkar

ABSTRACT

In recent years, driver awareness has been one of the major causes of road accidents and can lead to severe physical injuries, deaths and significant economic losses. Statistics indicate the need for a reliable driver awareness detection system which could alert the driver before a mishap happens. In this project,we collected different types of facial emotions images to create our own database by performing image-preprocessing. Using transfer learning, we trained our CNN models and tweaked the parameters which gave us better accuracy. Also, we created bottleneck features for training on our dataset. We created inferences for evaluating the model for testing video streams real time.The best

model gave an accuracy of 89% and the checkpoint of this file will be used for simulation. We are using Django environment for creating a Web Application using HTML & CSS where we will pass our best predictions as input and the output will be displayed in the web application as a simulation.

PROJECT GOALS

1. Compare and increase the accuracy and reliability of drowsiness detection and emotion detection using new dataset
2. Using deep-learning and neural nets improving road safety in automatic vehicles
3. Compare and state the difference between EARSVM Model and present models

DATASET

We will be using the following resources for creating the database:

1. FacesDB
2. Closed Eye Database
3. Japanese Female Facial Expression (JAFFE) Database
4. Facial Expression Recognition

The database IMPA-FACE3D was created in 2008 to assist in the research of facial animation. In particular, for analysis and synthesis of faces and expressions. We take the six universal expressions between human races proposed:

- Happiness
- Sadness
- Surprise
- Anger
- Disgust
- Fear .

This dataset includes acquisitions of 38 individuals with a neutral face sample, samples corresponding to six universal facial expressions and other expressions referring to 5 samples containing mouth and eyes open and / or closed. Also two samples were considered corresponding to the lateral profiles of individuals. Altogether, the data set is composed of 22 men and 16 women, with the majority of individuals are aged between 20 and 50 years. 14 samples were acquired for all individuals, summarizing 532 samples in total.

PROJECT DESCRIPTION:

A driver who falls asleep and when the wheel loses control of the vehicle, an action which often results in a crash with either another vehicle or stationary objects. In order to prevent these devastating accidents, the state of drowsiness of the driver should be monitored. The goals of this project are:

- Integration of multiple facial recognition datasets and CEW(closed eye in the wild)
- Data preprocessing and dataset formation (conversion of all images to a standard format and standard color scale)
- Designing and training a convolutional neural network on our dataset.
- Creating bottleneck features and training the model on the newly formed dataset
- Selecting the model with best accuracy & using that model prediction for simulation
- Creating an inference model to test on real world image
- Integrating the model with real-time video streaming with Django environment
- Simulating the best predictions obtained for creating a Web Application using HTML & CSS

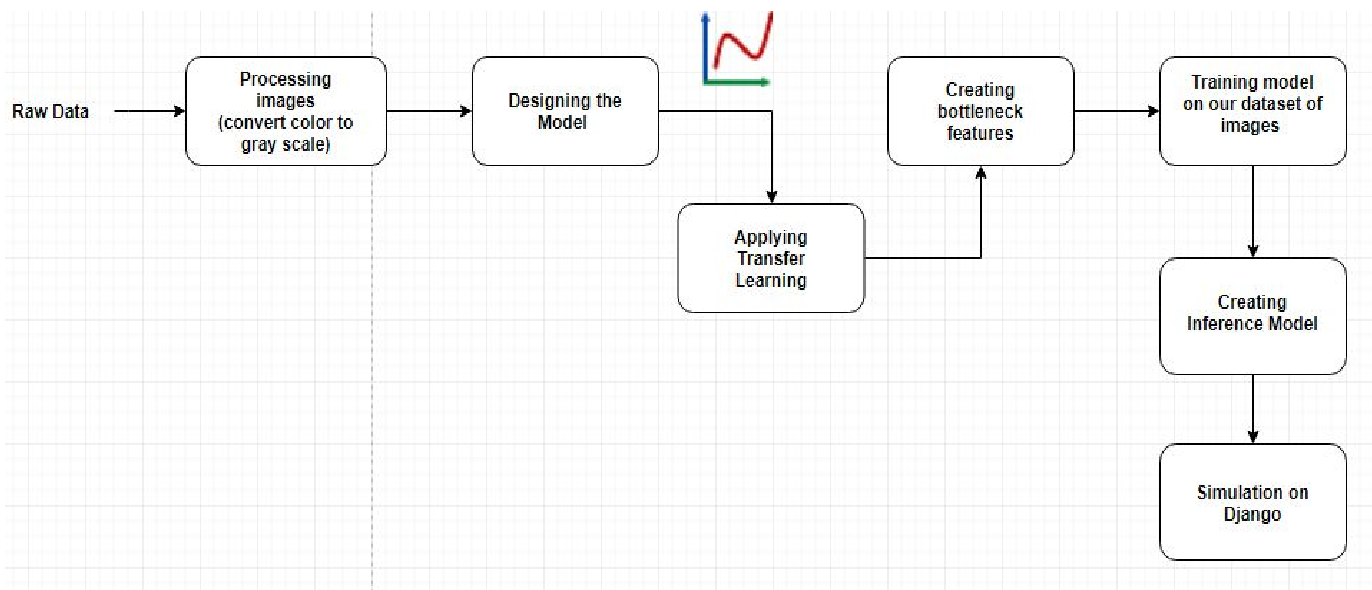
Architecture

Layer (type)	Output Shape	Param #
image_array (Conv2D)	(None, 48, 48, 16)	800
batch_normalization_61 (Batc	(None, 48, 48, 16)	64
conv2d_69 (Conv2D)	(None, 48, 48, 32)	25120
batch_normalization_62 (Batc	(None, 48, 48, 32)	128
activation_27 (Activation)	(None, 48, 48, 32)	0
average_pooling2d_29 (Averag	(None, 24, 24, 32)	0
dropout_25 (Dropout)	(None, 24, 24, 32)	0

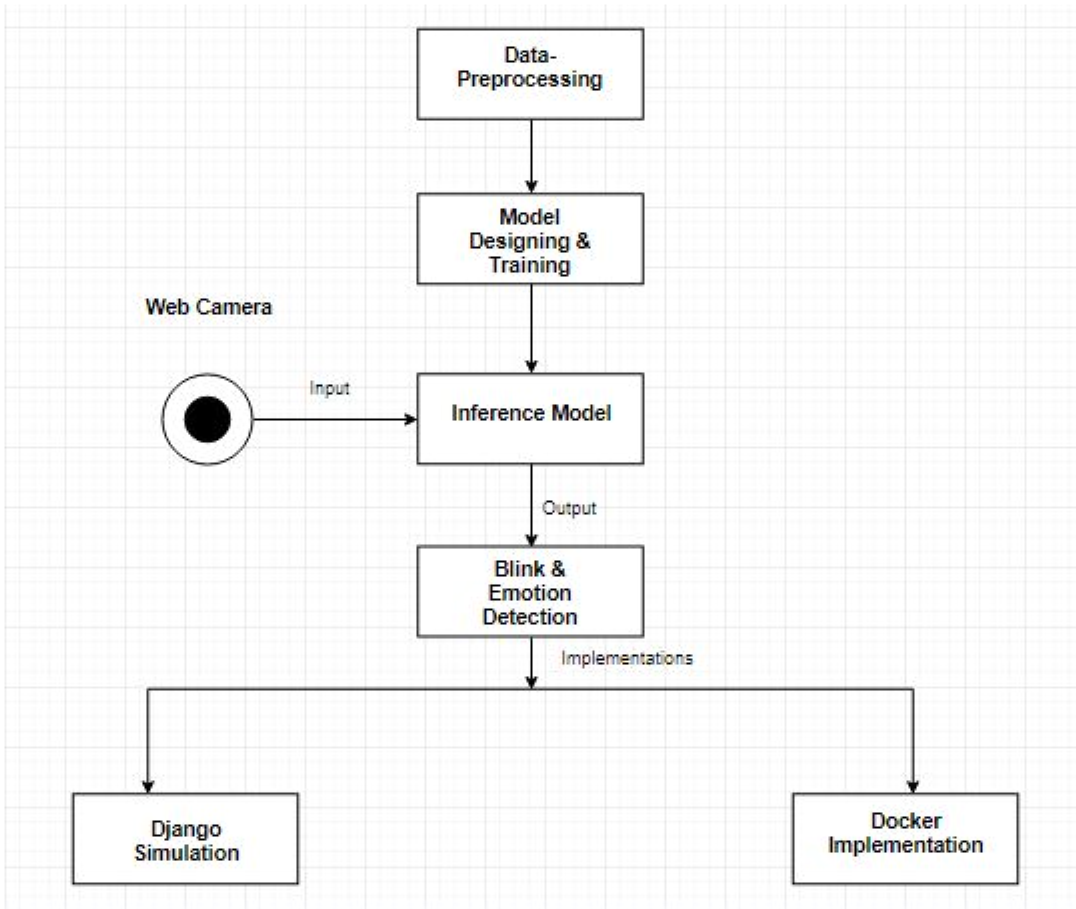
conv2d_70 (Conv2D)	(None, 24, 24, 64)	51264
batch_normalization_63 (Batc	(None, 24, 24, 64)	256
conv2d_71 (Conv2D)	(None, 24, 24, 64)	102464
batch_normalization_64 (Batc	(None, 24, 24, 64)	256
activation_28 (Activation)	(None, 24, 24, 64)	0
average_pooling2d_30 (Averag	(None, 12, 12, 64)	0
dropout_26 (Dropout)	(None, 12, 12, 64)	0
conv2d_72 (Conv2D)	(None, 12, 12, 64)	36928
batch_normalization_65 (Batc	(None, 12, 12, 64)	256
conv2d_73 (Conv2D)	(None, 12, 12, 128)	73856
batch_normalization_66 (Batc	(None, 12, 12, 128)	512
activation_29 (Activation)	(None, 12, 12, 128)	0
average_pooling2d_31 (Averag	(None, 6, 6, 128)	0
dropout_27 (Dropout)	(None, 6, 6, 128)	0
conv2d_74 (Conv2D)	(None, 6, 6, 128)	147584
batch_normalization_67 (Batc	(None, 6, 6, 128)	512
conv2d_75 (Conv2D)	(None, 6, 6, 256)	295168
batch_normalization_68 (Batc	(None, 6, 6, 256)	1024
activation_30 (Activation)	(None, 6, 6, 256)	0
average_pooling2d_32 (Averag	(None, 3, 3, 256)	0
dropout_28 (Dropout)	(None, 3, 3, 256)	0

```
conv2d_76 (Conv2D)          (None, 3, 3, 256)        590080
_________________________________________________________________
batch_normalization_69 (Batc (None, 3, 3, 256)        1024
_________________________________________________________________
conv2d_77 (Conv2D)          (None, 3, 3, 8)          18440
_________________________________________________________________
global_average_pooling2d_4 ( (None, 8)                0
_________________________________________________________________
predictions (Activation)     (None, 8)                0
=================================================================
Total params: 1,345,736
Trainable params: 1,343,720
Non-trainable params: 2,016
```

PIPELINE DESIGN



FLOWCHART DESIGN



IMPLEMENTATION DETAILS

In this project,the techniques which we are implementing are data-resizing,transfer learning, bottleneck features, inference predictions, docker implementation,simulation on Django environment.

Libraries: Tensorflow, Keras, OpenCV, Scipy, Django, HTML, CSS, AJAX

STEP 1:Data-Resizing

- Resizing an image helps to adjust the size of the image to the desired proportions ,whether it is in pixels,inches or in a specified percentage of change. We use the rescale operation that resizes an image by a given scaling factor. The scaling factor can either be a single floating point value, or multiple values - one along each axis.
- Resize serves the same purpose, but allows to specify an output image shape instead of a scaling factor.When down-sampling an image, resize and rescale should perform Gaussian smoothing to avoid aliasing artifacts. Downscale serves the purpose of down-sampling an n-dimensional image by integer factors using the local mean on the elements of each block of the size factors given as a parameter to the function.
- We have resized the all the original images to 48x48 dimensions
- The dataset created had nearby 32000 images which belonged to 8 classes

STEP 2:Transfer learning

- Transfer learning is a machine learning technique where a model trained on one task is re-purposed on a second related task.
- In transfer learning, we first train a base network on a base dataset and task, and then we repurpose the learned features, or transfer them, to a second target network to be trained on a target dataset and task. This process will tend to work if the features are general, meaning suitable to both base and target tasks, instead of specific to the base task.

STEP 3:Bottleneck Features

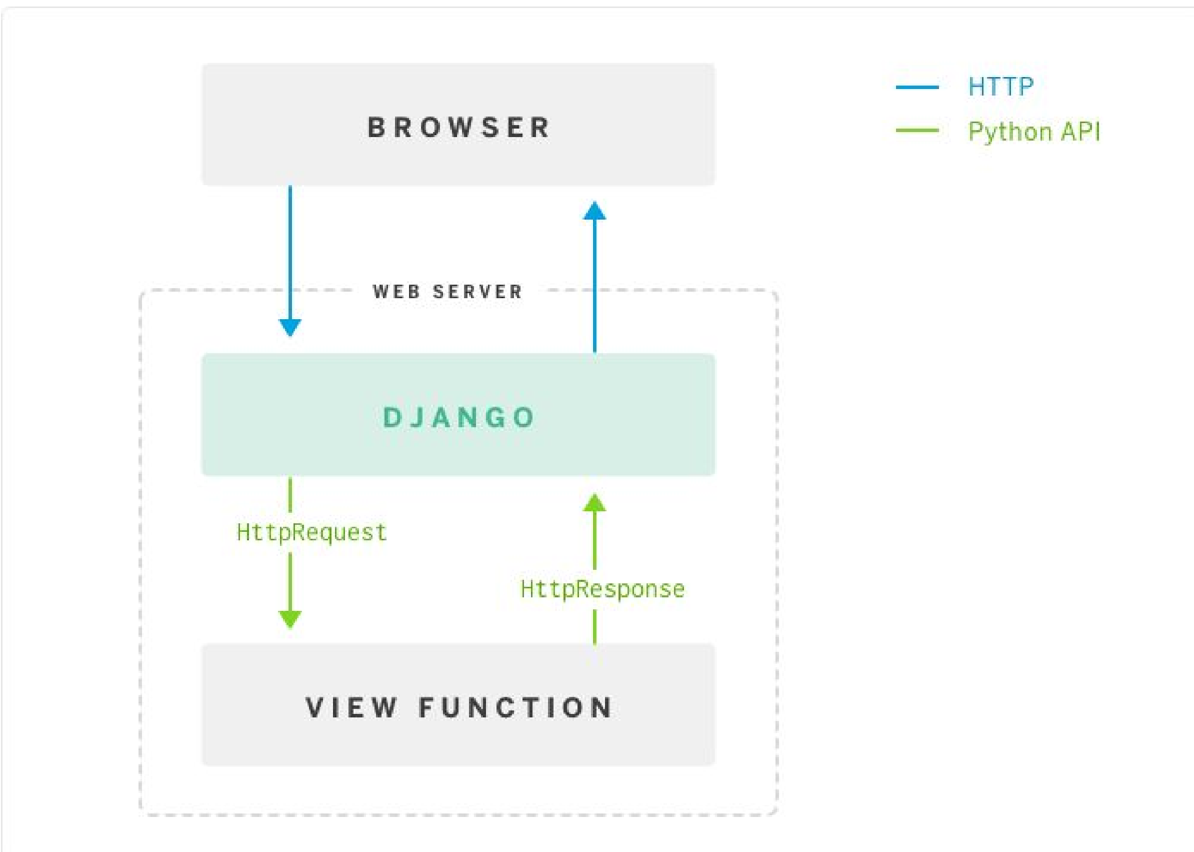
- Deep Learning supports an immensely useful feature called 'Transfer Learning'. Basically, we are able to take a pre-trained deep learning model - which is trained on a large-scale dataset and re-purpose it to handle an entirely different problem.
- The basic technique to get transfer learning working is to get a pre-trained model (with the weights loaded) and remove final fully-connected layers from that model. We then use the remaining portion of the model as a feature extractor for our smaller dataset.
- These extracted features are called "Bottleneck Features" (i.e. the last activation maps before the fully-connected layers in the original model). We then train a small fully-connected network on those extracted bottleneck features in order to get the classes we need as outputs for our problem.

STEP 4: Inference Predictions

- After building multiple models, we got the best model with an accuracy of 89% which is our best predictions and will be used as an input for simulation.

STEP 5: Simulation using Django

- Django is a Python-based free and open-source web framework, which follows the model-view-template architectural pattern. It is maintained by the Django Software Foundation, an independent organization established as a 501 non-profit. Django's primary goal is to ease the creation of complex, database-driven websites.
- In the project, we are simulating Django environment for creating a Web Application using HTML & CSS where we will pass our best predictions as input and the output will be displayed in the web application as a simulation.



ALGORITHMS

HAAR cascade for face detection

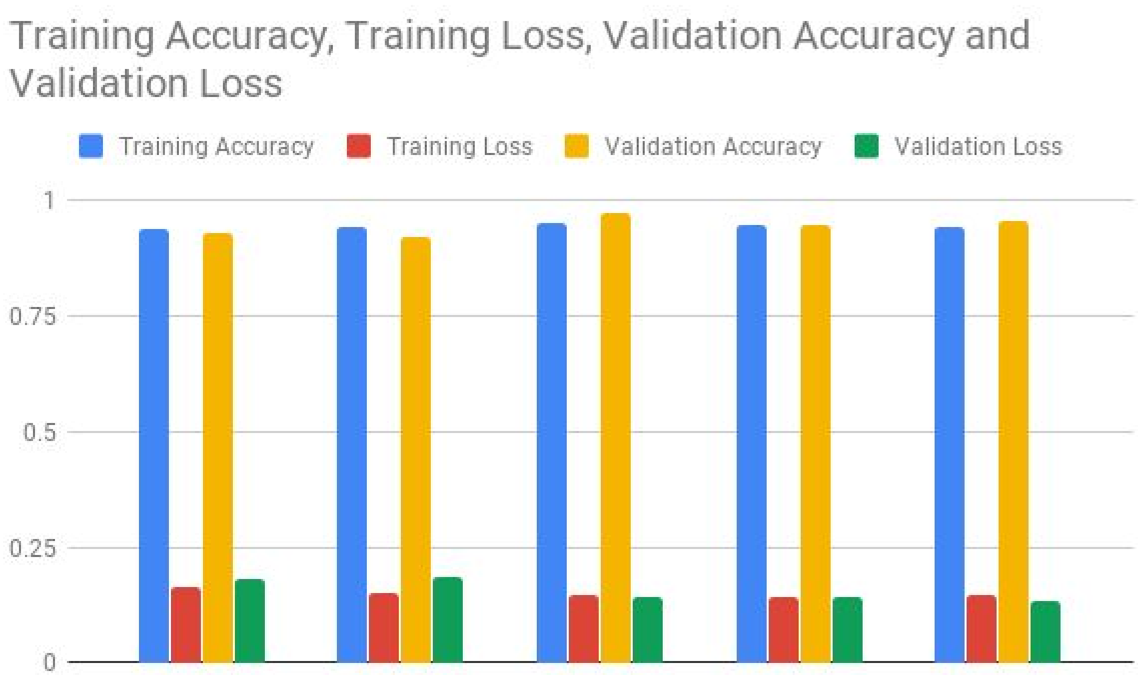
Haar Cascade is basically a classifier which is used to detect the object for which it has been trained for, from the source.The Haar Cascade is trained by superimposing the positive image over a set of negative images. The training is generally done on a server and on various stages. Better results are obtained by using high quality images and increasing the amount of stages for which the classifier is trained.

ANALYSIS OF MODELS

1) Emotion Detection Models

Model No	No. of Epochs	Layers	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
CNN Model 1	5	Convo2D :4 Dense : 2	0.3495	1.7063	0.3523	1.6901
CNN Model 2	30	Convo2D :4 Dense : 2 AvgPool: 3	0.4348	1.5039	0.4631	1.4335
CNN Model 3	41	Convo2D :4 Dense : 2 AvgPool: 4	0.8892	0.2858	0.8907	0.2800
CNN Model 4	60	Convo2D :5 Dense : 3 AvgPool: 1	0.5615	1.1753	0.5635	1.1649
CNN Model 5	56	Convo2D :4 Dense : 3 AvgPool: 3	0.5349	1.2426	0.5425	1.2060
CNN Model 6	25	Convo2D :2 Dense : 2 AvgPool: 2	0.8870	0.2933	0.8890	0.2859
CNN Model 7	72	Convo2D :4	0.8965	0.2624	0.8987	0.2554

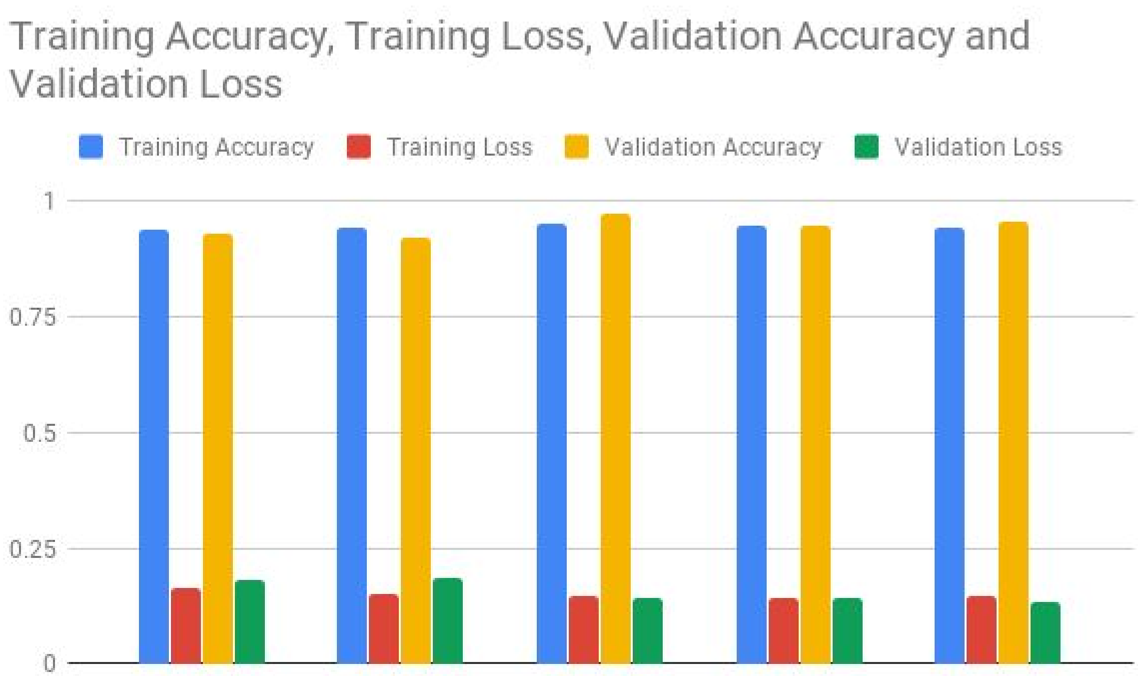
		Dense : 2				
CNN Model 8	75	Convo2D :2 Dense : 2 AvgPool : 1	0.7850	0.2783	0.7981	0.2134
CNN Model 9	85	Convo2D :3 Dense : 3	0.6970	1,1369	0.5481	1.1428
CNN Model 10	17	Convo2D :4 Dense : 3 AvgPool: 3	0.8510	0.3633	0.8528	0.3509



2) Eye-classifier Recognition Models:

Model No	No. of	Layers	Training	Training	Validation	Validation

	Epochs		Accuracy	Loss	Accuracy	Loss
1	22	Conv2D-4 Dense- 2	0.9384	0.1642	0.9317	0.1825
2	47	Conv2D-4 Dense - 2	0.9421	0.1521	0.9202	0.1873
3	17	Conv2D-4 Dense - 2	0.9517	0.1477	0.9746	0.1420
4	63	Conv2D-4 Dense -2	0.9454	0.1431	0.9462	0.1437
5	58	Conv2D-4 Dense- 2	0.9417	0.1478	0.9561	0.1354



DETAILS ON RUNNING THE MODELS:

The model for this project has various functions such as detection, sleepchecker, and emotion detection. It is explained as follows

Load Detection model: This function deals with cascaded classifier where it loads the pre-trained model which has trained modules of the face and eye detection with the listed expression. The pre-trained data for face is of 32000 and for eye-detection is 2000.

Offsets - This function deals with the coordinate mapping on face and eyelids.

Blink Detect - This function works on blink detection where it captures the time `for which eyes are closed or open. Blink detect works on the eye dataset.

Predict Emotions: We have given 6 flags to 6 different emotions which involve angry, neutral, happy, sad, surprised. 'Predict emotions function' uses face features from face data to predict the emotions of faces in real time. It helps to understand drivers current mood. The data set used in this function is of 32000.

To download pre-trained models please use the following link - Click Here

Conclusion

In the conclusion of this project, we can say that currently available technologies uses EAR technology or coordinate mapping with give accuracy till 90%. Our model uses CNN and transfer learning for blink detection which helps to increase the model accuracy to 96% with less validation and training loss.

We have compared this model with the current available pretrained models and can arguably say that the model developed by aforementioned process is better and if more data is provided.

References

1]https://towardsdatascience.com/real-time-face-liveness-detection-with-python-keras-and-opencv-c35dc70dafd3
2]https://medium.com/datadriveninvestor/real-time-facial-expression-recognition-f860dacfeb6a
3]https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html
4]https://www.pyimagesearch.com/2017/05/08/drowsiness-detection-opencv/