DRIVER AWARENESS DETECTION IN AUTONOMOUS CARS

Dhanashri Palodkar
Mansi Agrawal

TABLE OF CONTENTS

01

INTRODUCTION

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.

ABSTRACT

- Drivers Awareness is one of the major cause behind road accidents 1.25 million world-wide accidents being specific

- Tesla's new model X was our inspiration behind this project

- All the current models available uses the same dataset with 80-90% accuracy

- This project aims to increase the accuracy and model reliability

EARLIER METHODS OF DROWSINESS DETECTION

1

VEHICLE BASED MEASURES

2

BEHAVIORAL MEASURES

3

PHYSIOLOGICAL MEASURES

GOALS

- Designing a hybrid drowsiness detection system that combines non-intrusive psychological measures with other measures one would accurately determine the drowsiness of the driver
- Compare and increase the accuracy and reliability of drowsiness detection and emotion detection using new dataset
- Using deep-learning and neural nets improving road safety in automatic vehicles
- Compare and state the difference between EARSVM Model and present models

02

METHODOLOGY

PROBLEM ANALYSIS

- 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.

OUR PLAN

- 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

Used Cases

Front line workers in the factory

Security officers

Students monitoring in class

Reverse Used Cases

Monitor babies

ALGORITHM

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.

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
```

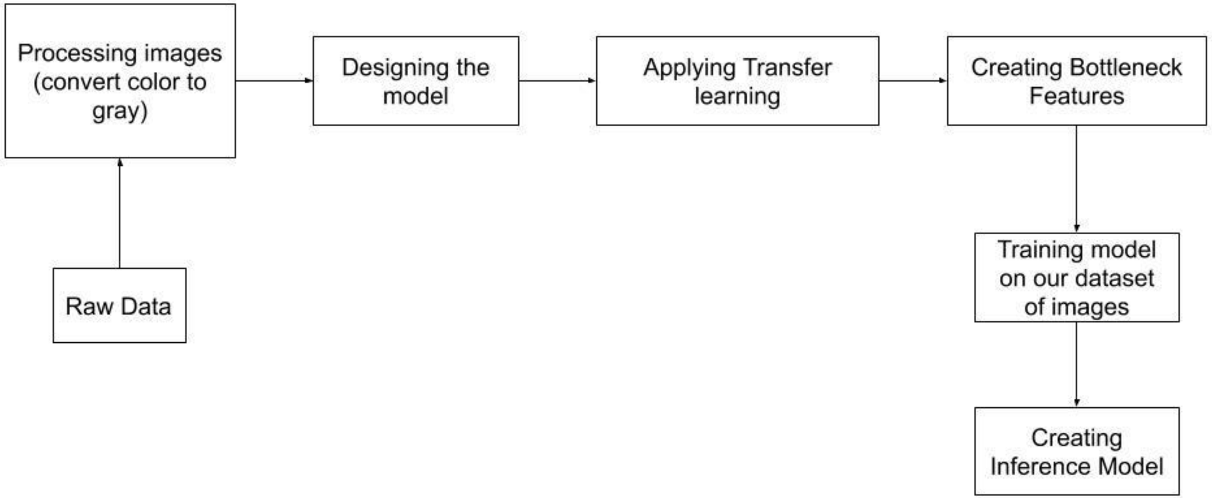
```
_________________________________________________________________
average_pooling2d_29 (Averag (None, 24, 24, 32)       0
_________________________________________________________________
dropout_25 (Dropout)         (None, 24, 24, 32)       0
_________________________________________________________________
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
```

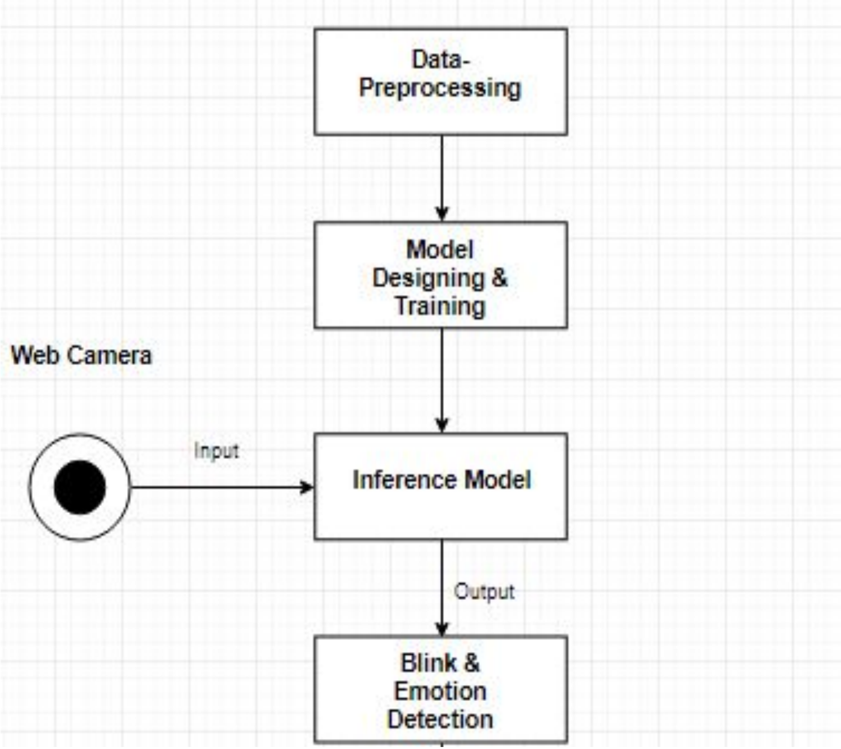
```
_________________________________________________________________
dropout_27 (Dropout)        (None, 6, 6, 128)        0
_________________________________________________________________
conv2d_74 (Conv2D)          (None, 6, 6, 128)        147584
_________________________________________________________________
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

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.
- Resize serves the same purpose, but allows to specify an output image shape instead of a scaling factor.
- 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.
- 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

- 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 mode
- Use the remaining portion of the model as a feature extractor for our smaller dataset.
- These extracted features are called **"Bottleneck Features"** 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.

DETAILS ON RUNNING THE MODELS

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

Load Detection model: This function deals with a 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 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 the driver's current mood. The data set used in this function is 32000.

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

03

DATASET

DATASET

01

[FacesDB]

02

[Closed Eye Database]

03

[Japanese Female Facial Expression (JAFFE) Database]

04

[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 aged between 20 and 50 years. 14 samples were acquired for all individuals, summarizing 532 samples in total.

04

RESULT & ANALYSIS

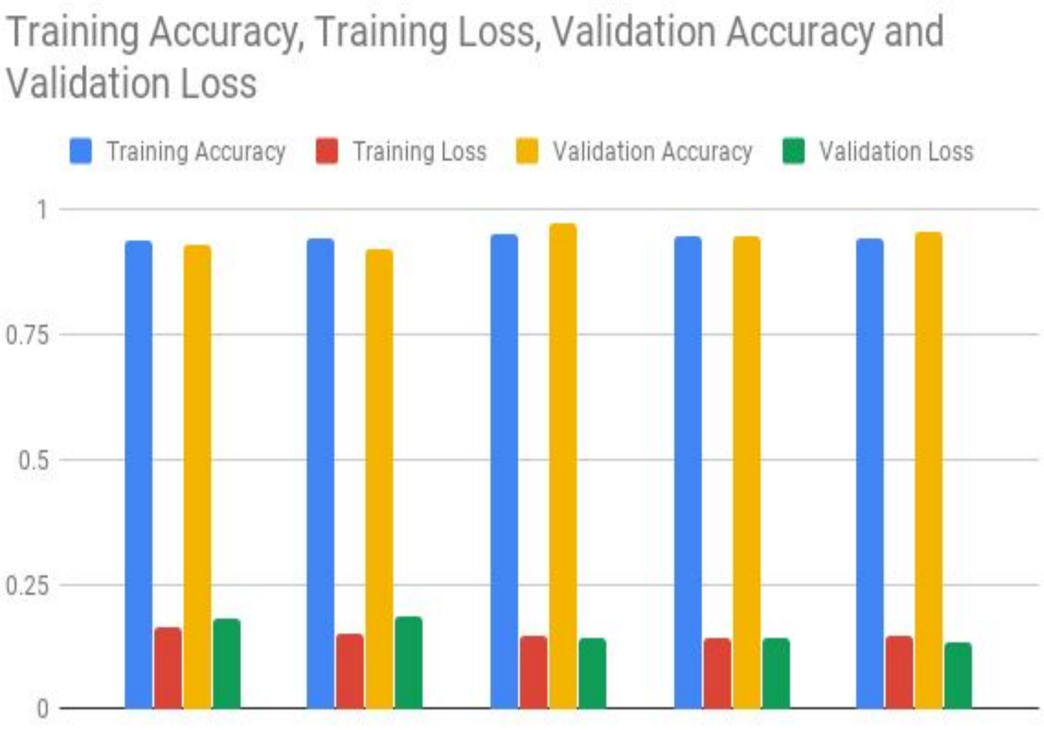
ANALYSIS OF MODELS

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

Model No	No. of Epochs	Layers	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
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

Model No	No. of Epochs	Layers	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
CNN Model 7	72	Convo2D :4 Dense : 2	0.8965	0.2624	0.8987	0.2554
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

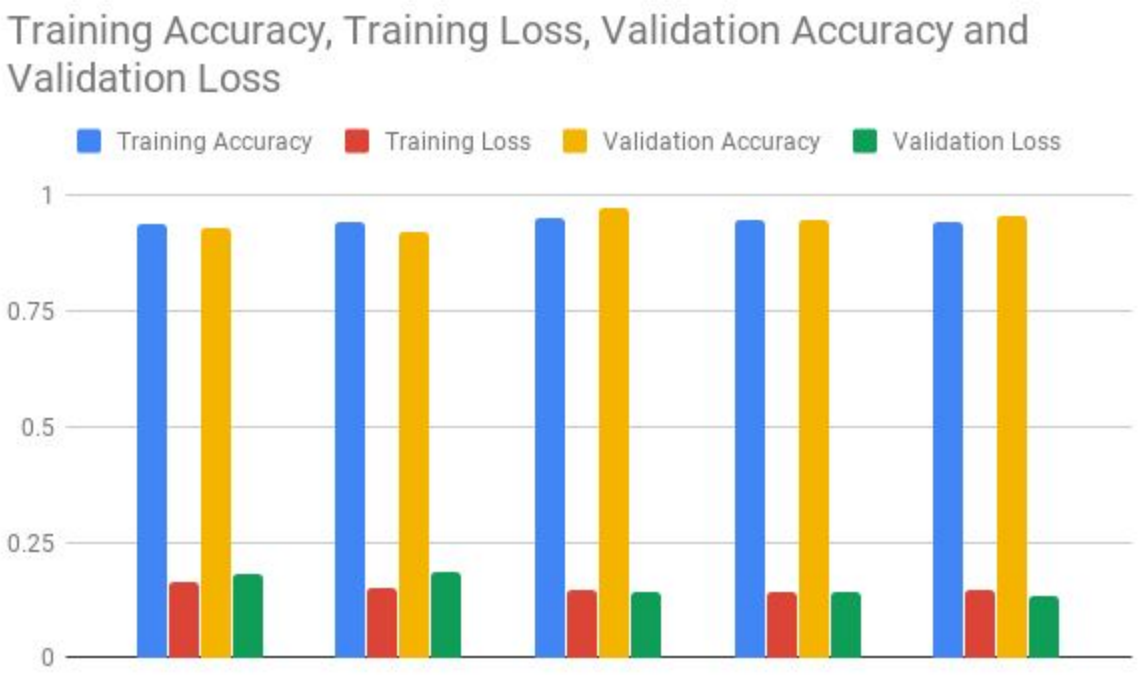

Training Accuracy, Training Loss, Validation Accuracy and Validation Loss
Training Accuracy
Training Loss
Validation Accuracy
Validation Loss
1
0.75
0.5
0.25
0

Eye-classifier Recognition Models:

Model No	No. of Epochs	Layers	Training Accuracy	Training Loss	Validation Accuracy	Validation 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

Eye-classifier Recognition Models:

Model No	No. of Epochs	Layers	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
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


Training Accuracy, Training Loss, Validation Accuracy and Validation Loss
Training Accuracy
Training Loss
Validation Accuracy
Validation Loss
1
0.75
0.5
0.25
0

05

Conclusion

Conclusion

- we can say that currently available technologies use EAR technology or coordinate mapping with 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 currently available pretrained models and can arguably say that the model developed by aforementioned process is better and if more data is provided.

06

Reference

Reference

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