I. DROWSINESS DETECTION SYSTEM WITH EMOTION ANALYSIS FOR IMPROVED VEHICLE EXPERIENCE

Akshay Singh
Illinois Institute of Technology
asingh149@hawk.iit.edu

Amogh A Kori
Illinois Instiute of Technology
akori@hawk.iit.edu

Rahul Bharadwaj M
Illinois Institute of technology
rmachiraju@hawk.iit.edu

II. ABSTRACT

The goal of this project is to create a Facial recognition system, that could be implemented in a motor vehicle, which will enhance its security. We will also use the driver's facial expressions to predict whether the driver is feeling drowsy, or if they are under the influence. In that case, we will send them an alert, warning them about their condition, so they can choose to not drive further or take a break. We will also implement emotion detection system along with it, which will scan the user's face and try to predict how the user is feeling at the time based on their expressions. This could be utilized for creating a personalized in cabin personalized experience which could include a specific lighting or a specific kind of music or a combination of both for each mood of the user. This could also be paired with voice analysis to figure out if a user if distressed beyond their natural levels and an alert could be sent to their physicians or family members. We will also compare different available approaches available to get this done, compare their relative performances and navigate through our reasoning for selecting a particular approach.

III. IMPORTANCE

Drowsy drivers are one of the major causes of car accidents negatively impacting millions of lives. The National Highway Traffic Safety Administration reported an estimate of about 150 individual deaths in the US alone each year which can be attributed to drivers being tired. 71,000 injured and $12.5 billion in losses, all because of lapse of alertness due to drowsiness [1]. Another report [2] showed that the U.S government and other organizations spend an estimated amount of $60.4 billion every year on drowsiness related accidents. Because of drowsiness, it costs buyers about $16.4 billion in property harm, well-being cases, time, and efficiency misfortunes. In 2010, the National Sleep Foundation (NSF) reported that about 54% of grown-up drivers felt sluggish while driving a vehicle, and 28% were sleeping, In reality.

Massive setbacks, injuries, damage to property and life is brought upon by sleepy drivers each year not just in the USA but across the globe. Automobile giants like Toyota, Tesla, Ford spend millions of dollars in drowsiness recognition strategies each year to find the most appropriate strategy for a high level of. Toyota, Ford, Mercedes-Benz, and other automakers utilize vehicle well-being innovation to anticipate mishaps when drivers are sluggish. A system that detects a drowsy driver and encourages them to take a break will make vehicles much more secure and reduce casualties significantly.

Second part of this project deals with enhancing the in-vehicle experience by scanning the face to detect user emotions and customizing in cabin experience. This could be a great selling point for an automobile marketeer especially for automakers targeting the luxury car segment.

IV. APPROACH

We have divided this project in 3 parts, first part is concerned with detecting drowsiness of a user and giving them an alert message if they appear to be sleepy. For this, we have used libraries like SciPy, Imutils, Dlib, and OpenCV. We also share what other approaches were available to us for this and why we selected the approach we selected. Second part of this project deals with scanning the user's face and identify emotions based on their expressions. For this, we have used the Convolutional Neural Network from the well-known Deepface library which performs a facial attribute analysis to predict - age, gender, emotion and race of the user. For our purpose only emotions are required, namely - angry, fear, neutral, sad, disgust, happy and surprise. Third part of the project would be to combine parts 1 & 2 to create the final code which will give user an alert when they are feeling sleepy as well as detect and print out the emotion a driver is experiencing.

Transfer Learning and Pre-trained Models

Neural networks models are very different as compared to other supervised machine learning algorithms. There are multiple reasons for that, but the main differentiating factor is the cost of running algorithms on the hardware.

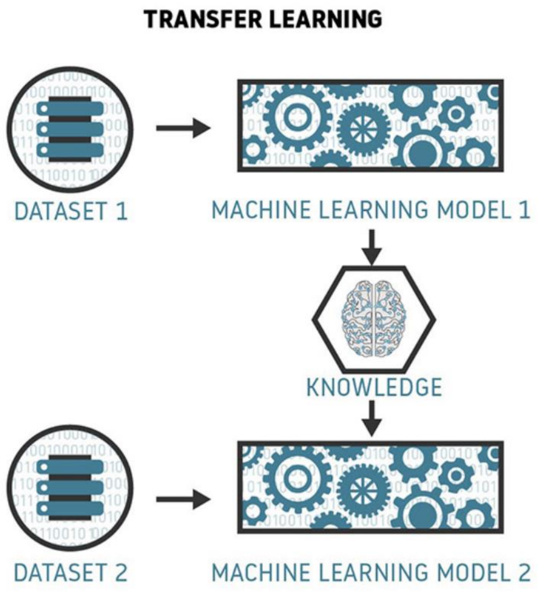
Although the price of RAM has gone down significantly, and is available for cheap, you still need access to GPUs to run a super complex machine learning problem, and access to GPU is still not so cheap. This will drive the cost of your project up. Specially in cases where we are trying to do image or voice recognition, where adding just one more layer of hidden layer would need immense resources. So, a smarter alternative of solving this problem is "Transfer Learning" [3]. This enables us to use pre-trained models from other people by implementing small changes.

What is transfer Learning?

Let's commence by developing an intuition behind what transfer learning is. Let us take the example of a simple teacher – student analogy.

A teacher has so many years of experience in the field they teach. With all this accumulated experience and information, what a student receives in a lecture is a concise and brief overview of the topic. This can be seen as a "transfer" of information from the learned to a beginner.

Keeping this example in mind, we try to compare this to the context of a neural network. A neural network is trained on a data set, consequently this network gains knowledge from this data, which is compiled in the form of "weights" of the network. These weights can then be extracted out and then used in any other neural network. Instead of training the other neural network from scratch, we "transfer" the learned features or weights.



What is Pre trained Model?

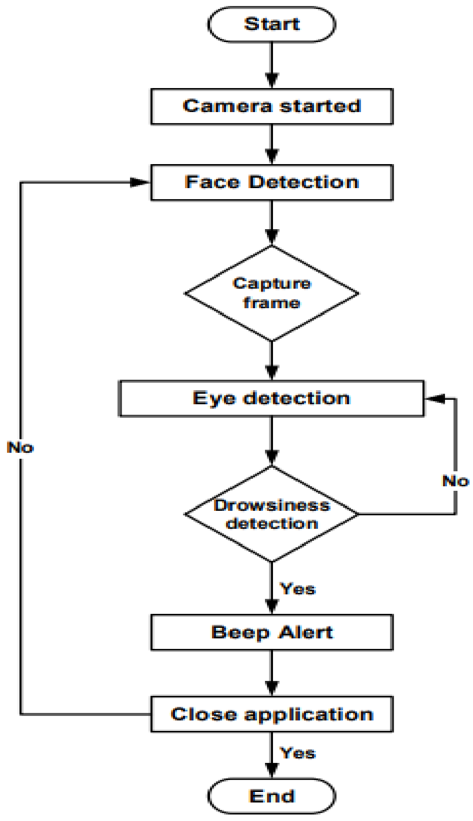
A pre-trained model[3] is a model created by someone else to solve a similar problem. Instead of building a model from ground up to solve problem with a similar scope, you use the model pre-trained on some other similar problem as the starting point.

For example, if you want to create a self-learning car. You can spend many years to build an image recognition algorithm from nothing or you can take inception model as the starting point (a pre-trained model), which was built by Google on ImageNet data to classify images in those pictures. A pre-trained model may not be as accurate as a model that you create from scratch customized to your own problem, but you save on huge efforts required to re-invent the wheel.

Although pre-trained models are effective, we need to be careful while choosing which pre-trained model we should select for our particular problem. If the problem we are trying to solve is very different from the one on which the pre-trained model was run – the prediction, we will get from it would probably be inaccurate. For example, a model trained for solving speech recognition would work poorly when we use it to identify faces.

A. Drowsiness Detection

Workflow for Drowsiness

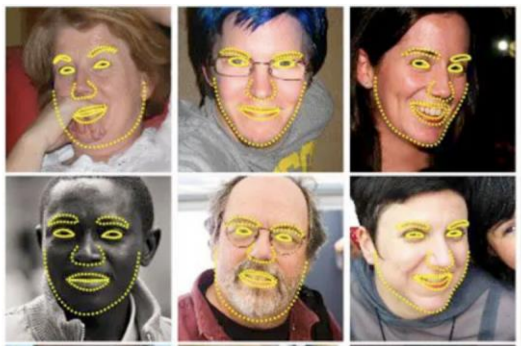


The libraries that are used:

OpenCv – This is an open-source computer vision and machine learning software library. OpenCV[4] was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.
Imutils package - Series of computer vision and image processing functions to make working with OpenCV easier.
SciPy package - we can compute the Euclidean distance between facial landmarks points in the eye aspect ratio calculation.
Dlib – It's a landmark's facial detector with pre-trained models, the dlib is used to estimate the location of 68 coordinates (x, y) that map the facial points on a person's face like image given here.

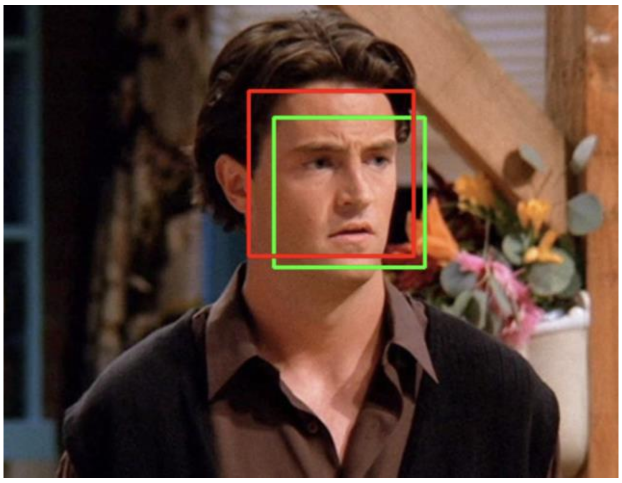
Two main functions used in the Dlib are –

shape_predictor()



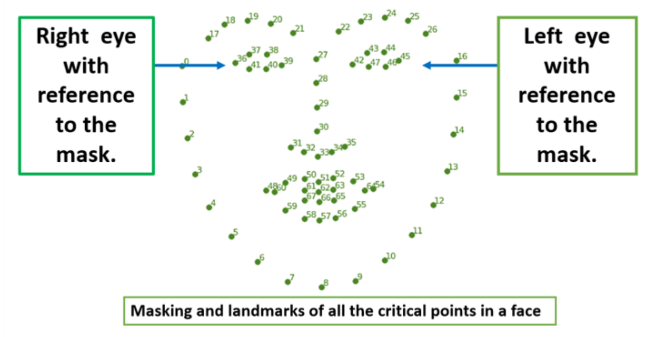
Shape predictors, also called landmark predictors, are used to predict key *(x, y)*-coordinates of a given "shape". The most common, well-known shape predictor is dlib's facial landmark predictor, which is used to localize individual facial structures, including the - Eyes, Eyebrows, Nose, Lips/mouth, Jawline.

get_frontal_face_detector()



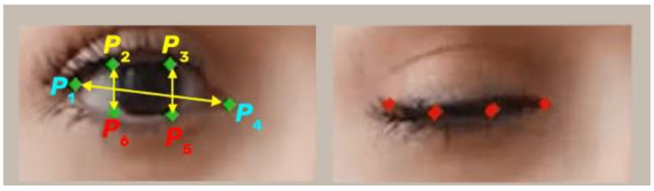
The get_frontal_face_detector function does not accept any parameters. A call to it returns the pre-trained HOG + Linear SVM face detector included in the dlib library.

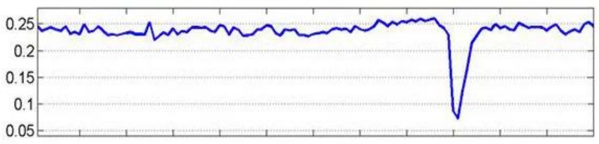
Dlib's HOG + Linear SVM face detector is fast and efficient. By nature of how the Histogram of Oriented Gradients (HOG) descriptor works, it is *not* invariant to changes in rotation and viewing angle.


Right eye with reference to the mask.
Left eye with reference to the mask.
Masking and landmarks of all the critical points in a face

Here are the 69 landmarks provided to us by Dlib's HOG + Linear SVM model. For our purpose we are using just the landmarks around right left eyes. Since, our purpose is to detect drowsiness, we can choose to ignore the landmarks around nose and lips for now.

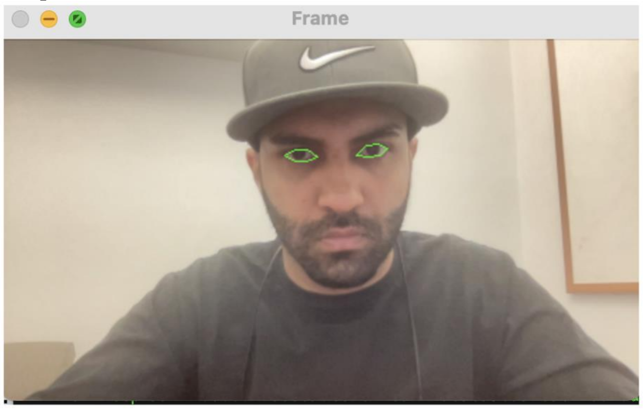
We need to define the eye_aspect_ratio function which is used to compute the ratio of distances between the vertical eye landmarks and the distances between the horizontal eye landmarks:



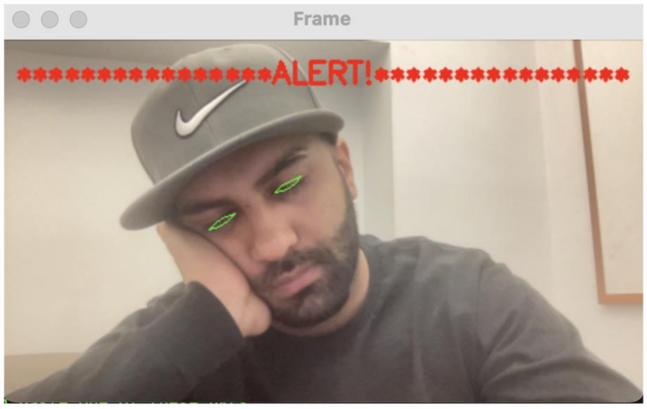


On the *top-left* we have an eye that is fully open with the eye facial landmarks plotted. Then on the *top-right* we have an eye that is closed. The *bottom* then plots the eye aspect ratio over time. As we can see, the eye aspect ratio is constant (indicating the eye is open), then rapidly drops to zero, then increases again, indicating a blink has taken place. In our drowsiness detector case, we'll be monitoring the eye aspect ratio to see if the value *falls* but *does not increase again*, thus implying that the person has closed their eyes.

Output:



When the user is active



When the user is sleepy

B. *Emotion Detection*

Deepface

Deepface[5] is a lightweight_face recognition and facial attribute analysis (age, gender, emotion and race) framework for python. It is a hybrid face recognition framework wrapping state-of-the-art models: VGG-Face, Google FaceNet, OpenFace, Facebook DeepFace, DeepID, ArcFace, Dlib and SFace. DeepFace is even used by Facebook for tagging images. It was proposed by researchers at *Facebook AI Research (FAIR)* at the *2014 IEEE Computer Vision and Pattern Recognition Conference (CVPR)*. [ref]

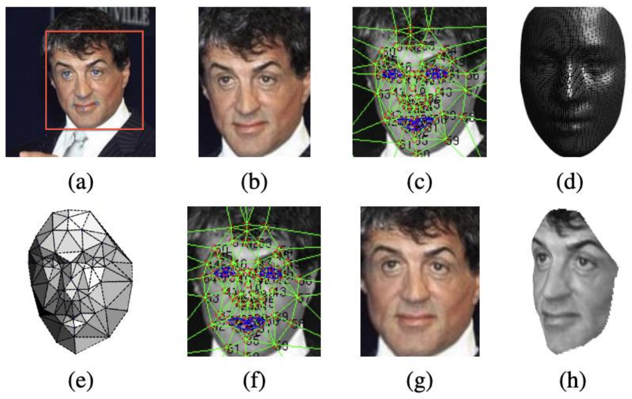
Deepface also comes with a strong facial attribute analysis module including age, gender, facial expression (including angry, fear, neutral, sad, disgust, happy and surprise) and race (including asian, white, middle eastern, indian, latino and black) predictions.

In modern face recognition there are 4 steps:

1. Detect
2. Align
3. Represent
4. Classify

Our model is only concerned with alignment[9] and representation of facial images. So, We will discuss these two parts in detail.

Alignment



The goal of this alignment process is to generate frontal face from the input image or a camera, that may contain faces from different any pose or angle. The method proposed in this paper used 3D frontalization of faces based on the fiducial (face feature points) to extract the frontal face. The whole alignment process is done in the following steps:

Given an input image, we first identify the face using *six fiducial points*. These six fiducial points are 2 eyes, tip of the nose and 3 points on the lips. These feature points are used to detect faces in the image.

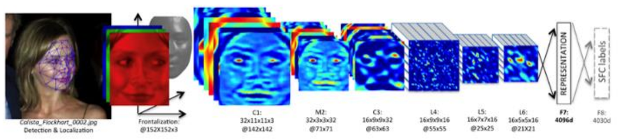
In the second step we generate the 2D-face image cropped from the original image using 6 fiducial points.

In the third step, we apply the *67 fiducial point map* with their corresponding Delauney Triangulation on the 2D-aligned cropped image. This step is done in order to align the out of plane rotations. In this step, we also generate a 3D-model using a generic 2D to 3D model generator and plot 67 fiducial points on that manually.

Then we try to establish a relation between 2D and 3D using given relation $x_{2d} = X_{3d}\overrightarrow{P}$.

The final stage is *frontalization of alignment*. But before achieving frontalization we add the residual component to x-y coordinates of 3D warp because it reduces corruption in 3D-warp. Finally, frontalization is achieved by doing piece-wise affine on Delauney triangulation that we generated on 67-fiducial points.

Classification Architecture



Outline of the DeepFace architecture. A front-end of a single convolution-pooling-convolution filtering on the rectified input, followed by three locally connected layers and two fully-connected layers. Colors illustrate feature maps produced at each layer. The net includes more than 120 million parameters, where more than 95% come from the local and fully connected layers

DeepFace is trained for multi-class face recognition i.e. to classify the images of multiple peoples based on their identities.

It takes input into a 3D-aligned RGB image of *152*152*. This image is then passed the Convolution layer with *32 filters* and size *11*11*3* and a *3*3* max-pooling layer with the stride of *2*. This is followed by another convolution layer of *16 filters* and size *9*9*16*. The purpose of these layers to extract low-level features from the image edges and textures.

The next three layers are locally connected layers, a type of fully connected layer that has different types of filters in a different feature map. This helps in improving the model because different regions of the face have different discrimination ability, so it is better to have different types of feature maps to distinguish these facial regions.

The last two layers of the model are fully connected layers. These layers help in establishing a correlation between two distant parts of the face. Example: Position and shape of eyes and position and shape of the mouth. The output of the second last fully connected layer is used as a face representation and the output of the last layer is the SoftMax layer *K* classes for the classification of the face.

The total number of parameters in this network is *120 million* approximately with most of them *(~95%)* comes from the final fully connected layers. The interesting property of this network is the feature map/vector generated during the training of this model amazingly sparse. For Example, *75%* of the values in topmost layers is 0. This may be because of this network uses ReLU activation function in every convolution network which is essentially *max (0, x)*. This network also uses *Drop-out Regularization* which also contributed to sparsity. However, Dropout is only applied to the first fully connected layer.

In the final stages of this network, we also normalize the feature to be between 0 and 1. This also reduces the effect of illumination changes across. We also perform an L2-regularization after this normalization.

Haar Cascade Classifier

A modern implementation of the Classifier Cascade face detection algorithm is provided in the OpenCV library. This is a C++ computer vision library that provides a python interface. The benefit of this implementation is that it provides pre-trained face detection models and provides an interface to train a model on your own dataset.

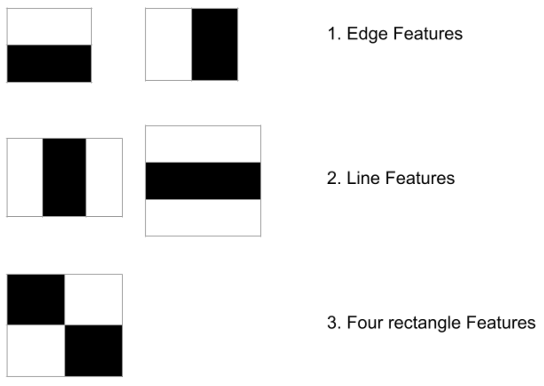
The nice thing about haar feature-based cascade classifiers is that you can make a classifier of any object you want, OpenCV already provided some classifier parameters to you, so you don’t have to collect any data to train on it.

It is superfast to work with and like the simple CNN, it extracts a lot of features from images. The best features are then selected via Adaboost. This reduces the original 160000+ features to 6000 features. But applying all these features in a sliding window will still take a lot of time. So, they introduced a Cascade of Classifiers, where the features are grouped. If a window fails at the first stage, these remaining features in that cascade are not processed. If it passes, then the next feature is tested, and the same procedure is repeated. If a window can pass all the features, then it is classified as a face region.

What are haar features like?

Haar features [6] are the relevant features for face detection. It was proposed by Alfred Haar in 1909. They are like convolutional kernels. There are various types of haar like features, but the most dominant features used are:

1. Rectangular Haar features
2. Rectangular Haar features
3. Rectangular Haar features

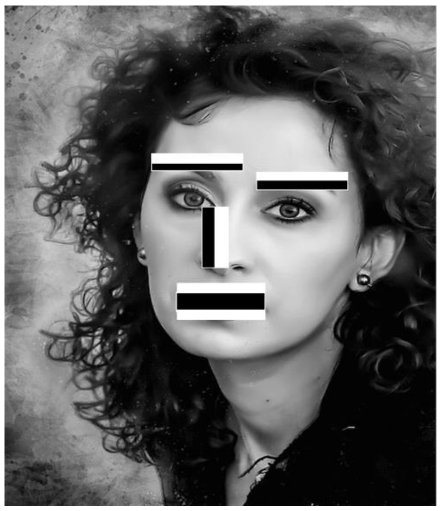


The values of a 2 rectangular feature are the difference between the sum of the pixels within 2 rectangular regions. The regions have same shape and size and are horizontally and vertically adjacent. A three rectangular feature computes the sum in a center rectangle. Finally, a four rectangular feature computes the difference between diagonal pairs of rectangles. Various variations of these regions of different sizes are convolved through the image in order to get multiple filters that will be inputs to the AdaBoost training algorithm.

Calculation of these features using the standard technique would require a high computation time. In order to reduce this time, a new approach called the integral image was suggested by the authors of the paper.

USE OF HAAR FEATURES IN FACE DETECTION

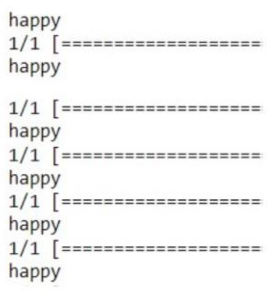
Face recognition means identifying the person and face verification means verify the person who is claimed to be, or in our case, detecting the emotion the user is experiencing. The key aspect in face recognition is detecting relevant features in human face like eyes, eyebrows, nose, lips. We detect these features in real time using Haar Wavelets or Haar Features. And the algorithm used is called as Viola-Jones Algorithm [7].



According to Viola-Jonas algorithm, to detect Haar like feature present in an image, below formula should give result closer to 1. The closer the value is to 1, the greater the change of detecting Haar feature in image.

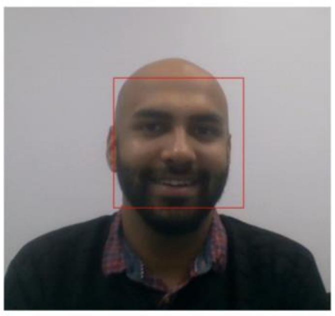
$$\Delta = \text{dark} - \text{white} = \frac{1}{n}\sum_{\text{dark}}^{n} I(x) - \frac{1}{n}\sum_{\text{white}}^{n} I(x)$$

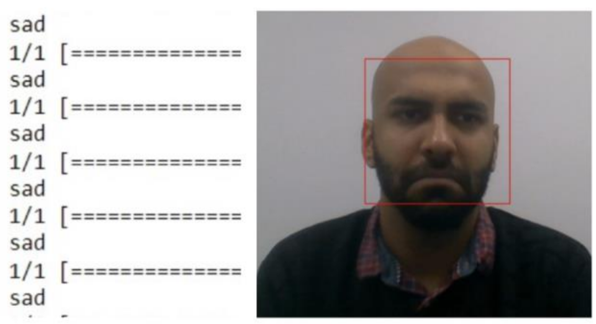
Output:



```
happy
1/1 [==================
happy

1/1 [==================
happy
1/1 [==================
happy
1/1 [==================
happy
1/1 [==================
happy
```

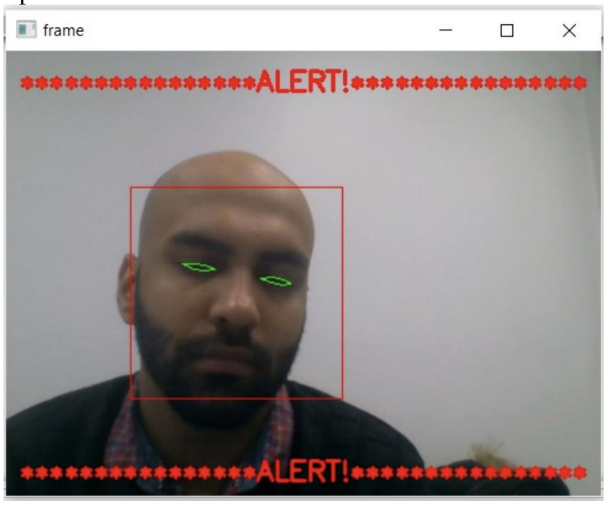


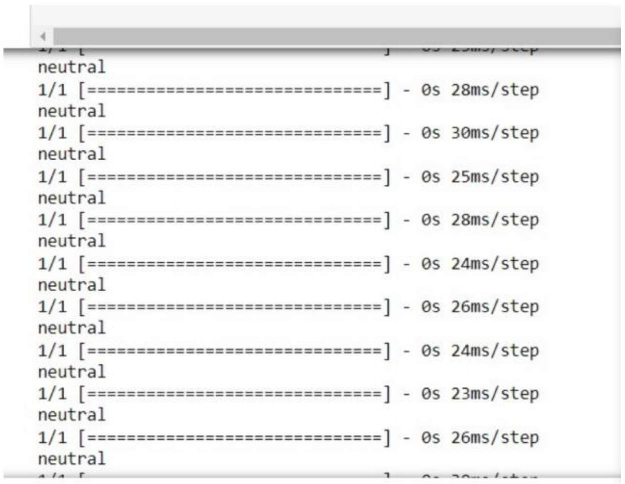


```
sad
1/1 [==============
sad
1/1 [==============
sad
1/1 [==============
sad
1/1 [==============
sad
1/1 [==============
sad
```

C. *Combination of Drowsiness and Emotion Detection*

Output:


frame
*****************ALERT!*****************
*****************ALERT!*****************



```
neutral
1/1 [==============================] - 0s 28ms/step
neutral
1/1 [==============================] - 0s 30ms/step
neutral
1/1 [==============================] - 0s 25ms/step
neutral
1/1 [==============================] - 0s 28ms/step
neutral
1/1 [==============================] - 0s 24ms/step
neutral
1/1 [==============================] - 0s 26ms/step
neutral
1/1 [==============================] - 0s 24ms/step
neutral
1/1 [==============================] - 0s 23ms/step
neutral
1/1 [==============================] - 0s 26ms/step
neutral
```

Upon completing the two segments separately, we moved on to the third step that is, combining these two parts. We had used Dlib's SVM classifier in the first part, which was working really efficiently and producing good results while trying to calculate eye aspect ratio. On the other hand, we had used Haar cascade classifier to not only classify emotions of the user, but also detect the correct orientation of the face. Which means, even if someone is driving their car looking sideways, our model has the capability to correctly map their facial features and classify their emotions. On combining, both features are seem to be working, although, the speed of execution has suffered a bit. This might be due to the fact that we are applying multiple models to read a user's facial features.

V. Conclusion

1. Dlib's HOG + Linear SVM face detector is fast and efficient. By nature of how the Histogram of Oriented Gradients (HOG) descriptor works, it is not invariant to changes in rotation and viewing angle. We could have used CNN for solving the problem of drowsiness detection and it would have given us better face detection even in the case of tilted face, but that would have come at the expense of a slower model, but since we are implementing this model for a vehicle, we are assuming that people will not be driving their cars with a tilted face, and therefore we decided that HOG + Linear SVM would be a better option for the particular problem we are trying to solve.

2. The use of CNN from the DeepFace library for the emotion detection helped us understand the use of Pre-trained models, which is a newly developed methodology. By working on this project, we wanted to explore both traditional methods, to see how the logic came about, as well as the newly developed methods like CNN to ultimately find the right balance between efficiency and accuracy.

VI. Future SCOPE

Personalized in-cabin experience

By correctly identifying the emotion of the driver, we can use this information to change certain car settings such as music, temperature, lights etc to match the emotion. For example - if a user and is feeling ecstatic that day, the car would automatically play some pop music or if the user is feeling angry or stressed, the car could identify it using our model play some soothing jazz music or ocean sounds while making the lights dim and making the temperature cooler. These settings would be customizable, and the user could save their own preferences for different moods.

Medical Diagnosis

A distressed face coupled with voice analysis could be used to alert the user as well as close family members or a family doctor in case of an extreme emotional state.

Child Protection

Kids getting injured because of heat strokes is still a huge problem that hasn't been solved on a large scale. There are products in the market that offer solution to this problem, but they all depend on expensive censors, so they are not in widespread use. However, with a simple camera device with face recognition system, it could offer help in recognizing unattended children in vehicles, and beyond set timeframe, it would send an alert to the parents. This could also be used to adjust entertainment systems inside the car to only display family-friendly content when it identifies a minor sitting in the car.

VII. References

1. *https://ieeexplore.ieee.org/document/8744224*
2. *https://ieeexplore.ieee.org/document/7460119*
3. *https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model/*
4. *https://pyimagesearch.com/2017/05/08/drowsiness-detection-opencv/*
5. *https://www.cs.toronto.edu/~ranzato/publications/taigman_cvpr14.pdf*
6. *https://medium.com/analytics-vidhya/what-is-haar-features-used-in-face-detection-a7e531c8332b*
7. *https://towardsdatascience.com/viola-jones-algorithm-and-haar-cascade-classifier-ee3bfb19f7d8*
8. *https://medium.com/analytics-vidhya/what-is-haar-features-used-in-face-detection-a7e531c8332b*
9. *https://www.geeksforgeeks.org/deep-face-recognition/*
10. *Github repo link : https://github.com/Amoghkori/CS-584--ML-Project*

VIII. Contributuions

Rahul M Bharadwaj – Emotion Detection, Project Proposal and Documentation

Amogh Kori – Drowsiness Detection, Project Proposal and Documentation

Akshay Singh – Combined Emotion Detection and Drowsiness Detection into one, Presentation and Documentation