Scientific Journal of Impact Factor (SJIF): 4.72

e-ISSN (O): 2348-4470
p-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 4, Issue 4, April -2017

Autonomous Distracted Driver Detection using Machine learining Classifiers

Prof. Praveen Hore[1],Prashant Tiwari[2],Ashwani Tiwari[3], Pawan Chauhan [4],Ravish Sharma[5]

[1,2,3,4, 5]*Department of Computer Engineering, Army Institute of Technology, Pune, India*

Abstract—*In most of the cases of accidents the prime cause of the accident is the distracted state of driver With the increase of In Vehicle Information System such cases are increasing. This problem can be solved by monitoring and predicting the state of driver while driving and installing the autonomous prevention system to take preventive actions. In order to realize this strategy we have used Convolutional Neural Networks and deep learning concepts in order to classify the image of driver into 10 different classes that are which include texting,talking to passenger,talking on the phone,looking left or right,searching something at the back seat,hair and make up,operating radio and drinking. Once the class of image is predicted the autonomous preventive system can be invoked to take the preventive steps accordingly.*

Keywords—*Machine Learning, Deep Learning , Convolutional Neural Network ,Classification , Hyperparameters*

I. INTRODUCTION

We are familiar with the concepts of machine learning and know that they can be used to classify a dataset .This feature can be used here to solve this problem. We can generate a classifier that is able to classify the various states of a driver depending on the image provided .The classified state , if found distracted , can then be used to generate an alarm to alert the driver and prevent the possibility of an accident .For this purpose we have obtained the dataset of drivers driving in different set. Using this dataset as training set and machine learning algorithms we will be able to generate a classifier that would predict the distracted state of the driver .Following States of Driver are to be predicted.: texting ,talking to passenger, talking on the phone, looking left or right, searching something at the back seat, hair and makeup , operating radio and drinking.We aim at building a model using Convolutional Neutral Network which could classify the images based on different states of driver. The images that are used for training and building the model has been taken using a static driving simulator with real human subjects performing a specific secondary task.

II. GOALS AND OBJECTIVE

The main goal of this project is to build a classifier model using machine learning and deep learning concepts that can classify the image of a driver into different classes and predict the distracted state of the driver .The accuracy of such a system must be high and the response time must be very low. Also the classifier system can be integrated with a autonomous prevention system such as auto breaking system in order to prevent the occurrence of accident. The system can be extended to real time analysis of driver by reducing the response time and increasing processing power. The ideal system must look like something shown below:

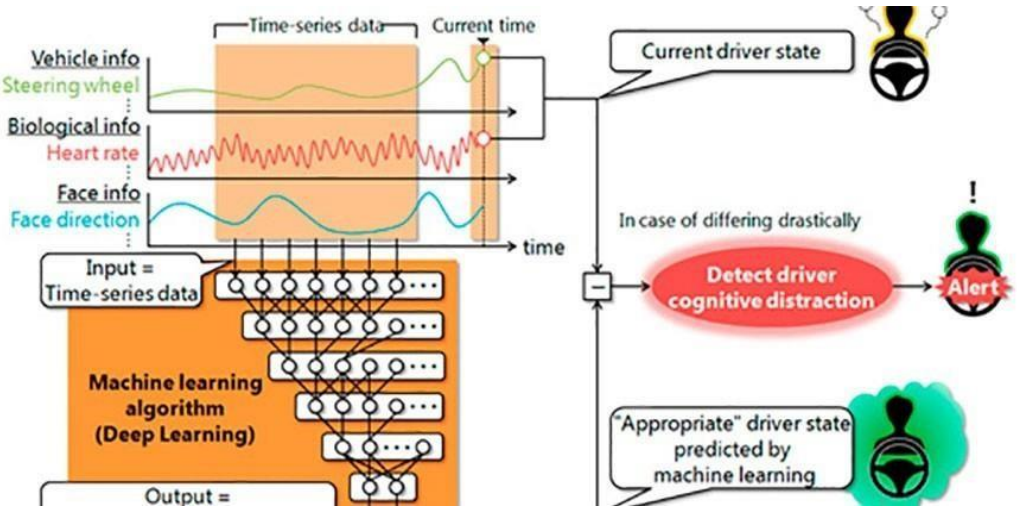


Figure 1 Overview of overall system after Completion

III. What is Distracted Driving?

According to National Highway Traffic Safety Administration (NHTSA) **, Distracted driving is any activity that diverts attention from driving, including talking or texting on your phone, eating and drinking, talking to people in your vehicle, fiddling with the stereo, entertainment or navigation system—anything that takes your attention away from the task of safe driving.** In 2015 alone, 3,477 people were killed, and 391,000 were injured in motor vehicle crashes involving distracted drivers. Although existing data are inadequate and not representative of the driving population, it is estimated that drivers engage in potentially distracting secondary tasks approximately 30time that their vehicles are in motion. (Having a conversation with passengers is the most frequent secondary task, followed by eating, smoking, manipulating controls, reaching inside the vehicle, and using cell phones).

IV. ECOSYSTEM SETUP

Creation an Ecosystem means installing and configuring following elements.

1. Installing Anaconda python on the system and setting up the configuration.
2. Installing the dependencies such as openCV , numpy , scikit learn and Keras.
3. Keras by default comes with Tensorflow as backend but we have to shift the backend to Theano .
4. Obtaining the dataset of driver images captured by the static simulator installed in vehicle and storing in the secondary storage of system.
5. Splitting the data into training and testing data and further splitting the training data into different classes that the model needs to be trained for. This has to be done on local storage of system by using the directory structure.

V. CONVOLUTIONAL NEURAL NETWORKS

ConvNets are designed to process data that come in the form of multiple arrays, for example a colour image composed of three 2D arrays containing pixel intensities in the three colour channels. Many data modalities are in the form of multiple arrays. Following things needs to be understood to completely understand Convolutional neural Networks :

1. **Feature (Convolution Kernels):** Just as its literal meaning implies, a feature is a distinct and useful observation or pattern obtained from the input data that aids in performing the desired image analysis. The CNN learns the features from the input images. Typically, they emerge repeatedly from the data to gain prominence. As an example, when performing Face Detection, the fact that every human face has a pair of eyes will be treated as a feature by the system , that will be detected and learned by the distinct layers. In generic object classiffication, the edge contours of the objects serve as the features.
2. **Receptive Field** : It is impractical to connect all neurons with all possible regions of the input volume. It would lead to too many weights to train, and produce too high a computational complexity. Thus, instead of connecting each neuron to all possible pixels, we specify a 2 dimensional region called the receptive field[14] (say of size 55 units) extending to the entire depth of the input (5x5x3 for a 3 colour channel input), within which the encompassed pixels are fully connected to the neural networks input layer. Its over these small regions that the network layer cross-sections (each consisting of several neurons (called depth columns)) operate and produce the activation map.
3. **Zero-Padding** : Zero-padding refers to the process of symmetrically adding zeroes to the input matrix. Its a commonly used modification that allows the size of the input to be adjusted to our requirement. It is mostly used in designing the CNN layers when the dimensions of the input volume need to be preserved in the output volume.
4. **Hyperparameters** : In CNNs, the properties pertaining to the structure of layers and neurons, such spatial arrangement and receptive field values, are called hyperparameters. Hyperparameters uniquely specify layers. The main CNN hyperparameters are receptive field (R), zero-padding (P), the input volume dimensions (Width x Height x Depth, or W x H x D) and stride length (S).
5. **Convolutional Layer** : Each neuron is connected to a certain region of the input volume called the receptive field (explained in the previous section). For example, for an input image of dimensions 28x28x3, if the receptive field is 5 x 5, then each neuron in the Conv. layer is connected to a region of 5x5x3 (the region always comprises the entire depth of the input, i.e. all the channel matrices) in the input volume. Hence each neuron will have 75 weighted inputs. For a particular value of R (receptive field), we have a cross-section of neurons entirely dedicated to taking inputs from this region. Such a cross-section is called a depth column. It extends to the entire depth of the Conv. layer.
6. **The ReLu (Rectified Linear Unit) Layer** : ReLu refers to the Rectifier Unit, the most commonly deployed activation function for the outputs of the CNN neurons.Unfortunately, the ReLu function is not differentiable at the origin, which makes it hard to use with backpropagation training.
7. **Pooling Layer** : The pooling layer is usually placed after the Convolutional layer. Its primary utility lies in reducing the spatial dimensions (Width x Height) of the Input Volume for the next Convolutional Layer. It does not affect the depth dimension of the Volume.
8. **The Fully Connected Layer** :The Fully Connected layer is configured exactly the way its name implies: it is fully connected with the output of the previous layer. Fully-connected layers are typically used in the last stages of the CNN to connect to the output layer and construct the desired number of outputs.

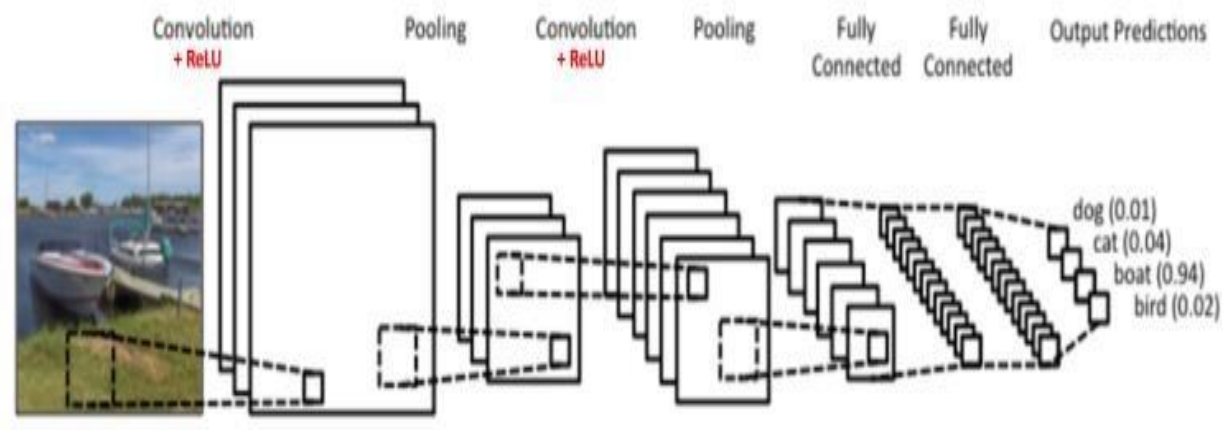


Figure 2 Layers Of Convolutional Neural Netwoks

VI. IMPLEMENTATION

Step 1 : Load the images from the training set into main memory

Step 2: Cleaning the images and converting into grey scale.

Step 3: Cropping the images within a given width and height through program.

Step 4: Defining the hyper parameters such as nb_epoch , filter size , window stride , batch_size , nb_classes.

Step 5: Stacking up of layers according to the architecture and design.

Step 6: Deciding up the validation split based on the training set.

Step 7: Training to build the CNN model combining the parameters and within training checking out the features learned by the model.

Step 8: Evaluating the loss score calculated during training and adjusting down the parameters so that error rate could be decreased.

Step 9: Changing the hyper parameters such as epoch value to compare previous models.

Step 10: After the train phase define the architecture and weight associated with neurons in the file so that model could be loaded for further testing.

Following is a code snippet how the CNN model has been constructed in python programming language using Keras Library :


```
{
   batch_size = 64
   nb_classes = 10
   nb_epoch = 2
   # input image dimensions
   img_rows, img_cols = 96, 128
   # number of convolutional filters to use
   nb_filters = 32
   # size of pooling area for max pooling
   nb_pool = 2
   # convolution kernel size
   nb_conv = 3
   model = Sequential()
   model.add(Convolution2D(nb_filters, nb_conv, nb_conv,
                    border_mode='valid',
                    input_shape=(1, img_rows, img_cols)))
   model.add(Activation('relu'))
   model.add(Convolution2D(nb_filters, nb_conv, nb_conv))
   model.add(Activation('relu'))
   model.add(MaxPooling2D(pool_size=(nb_pool, nb_pool)))
   model.add(Dropout(0.25))

   model.add(Flatten())
   model.add(Dense(128))
   model.add(Activation('relu'))
   model.add(Dropout(0.5))
   model.add(Dense(nb_classes))
   model.add(Activation('softmax'))

}
```

Figure 3 Code Snippet for Layers added to CNN model

6.1.Application Development: GUI based App for detection

The GUI of project was prepared using TKinter library of python in which the user has to select the image and output is displayed correspondingly .This is overall application using which the end-user could interact and load the image whose categories is to be predicted . It provides the description of the class to which the image belongs .The user interface of the project consists of the File Dialog Box used to select the image from any location well within your computer and provides the description of it .The components inside the user-interface must be managed in different threads to that The interface works softly and efficiently .

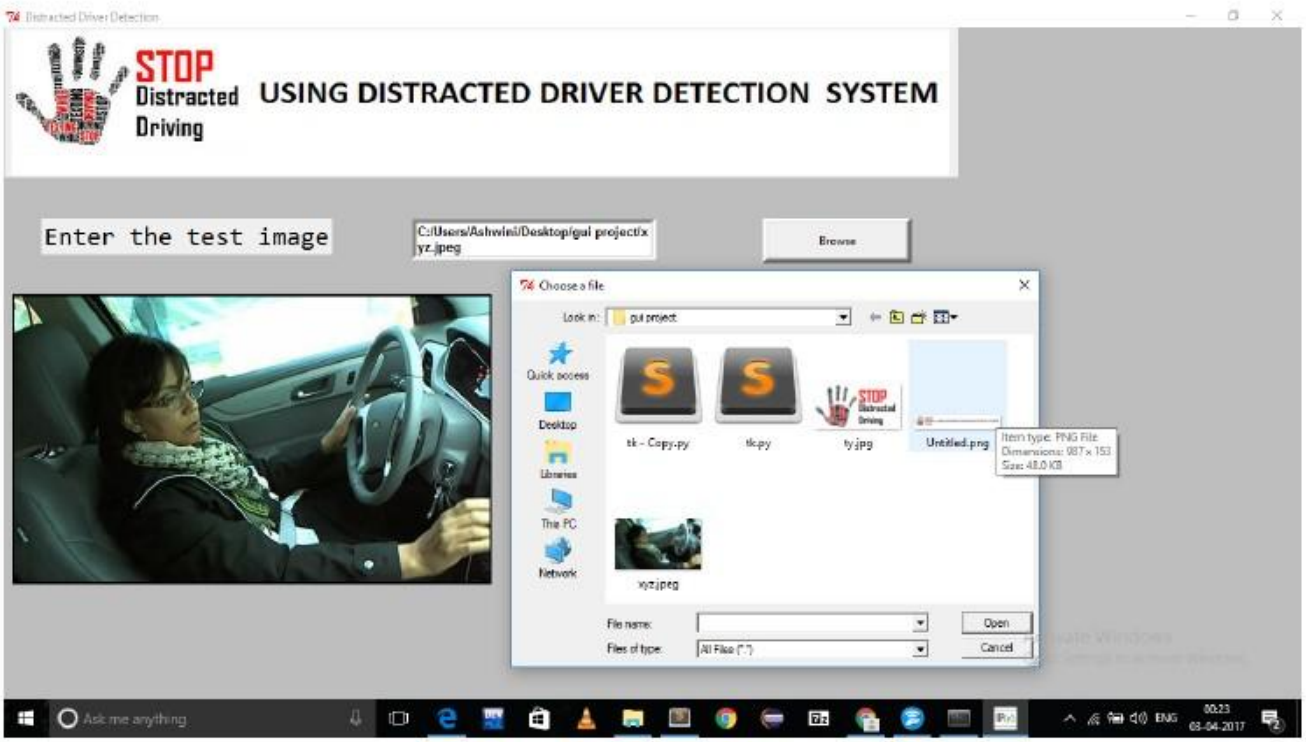


Figure 4 GUI For selecting image manually

VII. RESULTS

The model provides an accuracy score of .63 which is quite good when compared to building a model with SVM or other classifiers .That is out of 10 images our model could predict right answers for at least 6 images .The score for each of the classes for a particular image from the testing set has been stored in a csv file..
The project consisted of two phases the Training and the testing Phase . Results of each phase has been displayed below:

Training Phase:

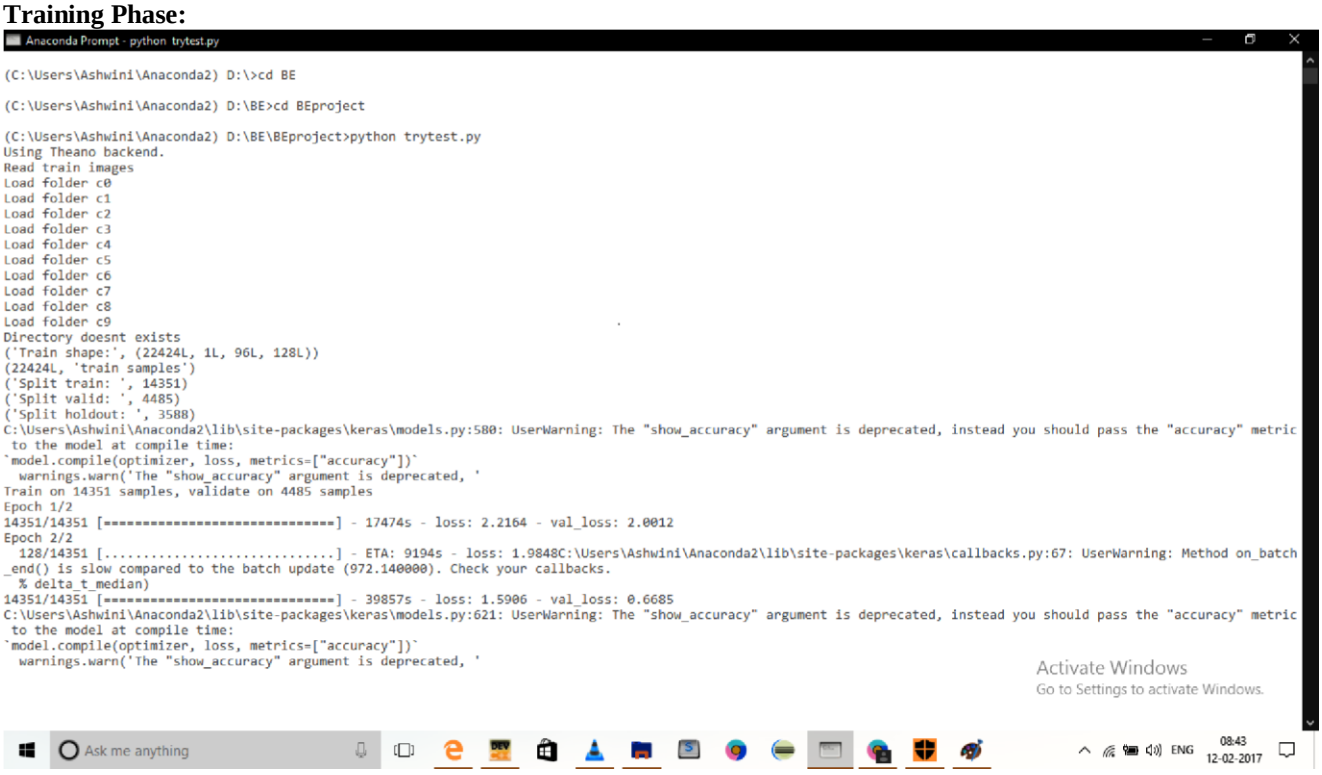
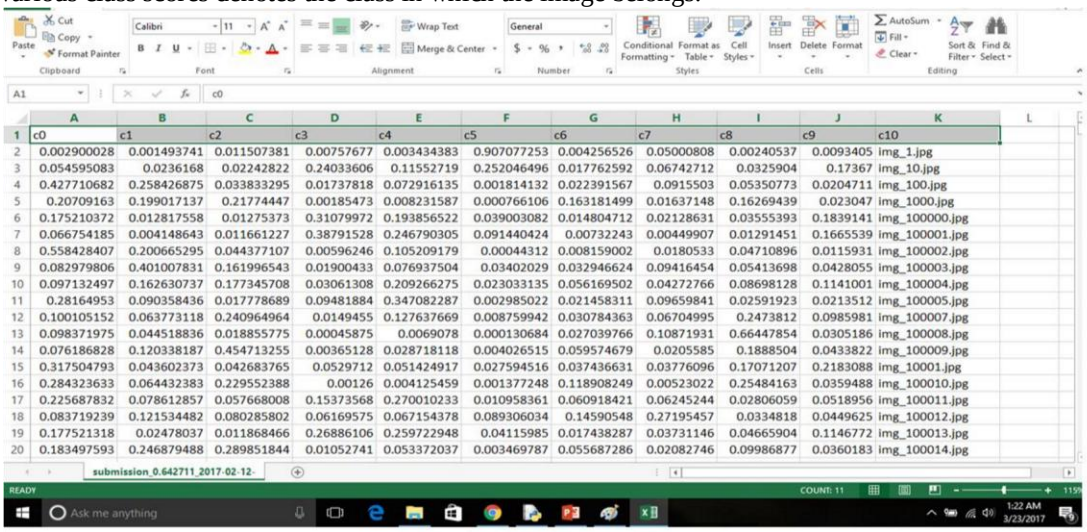


Figure5 Snapshot Of Training Phase

Testing Phase : When the images of the test set was provided as an input to the model it gave the class scores representing the pixel value which it turn denotes the probability of driver involved in that category .The maximum value along the various class scores denotes the class in which the image belongs.



	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10
2	0.002900028	0.001493741	0.011507381	0.00757677	0.003434383	0.907077253	0.004256526	0.05000808	0.00240537	0.0093405	img_1.jpg
3	0.054595083	0.0236168	0.02242822	0.24033606	0.11552719	0.252046496	0.017762592	0.06742712	0.0325904	0.17367	img_10.jpg
4	0.427710682	0.258426875	0.033833295	0.01737818	0.072916135	0.001814132	0.022391567	0.0915503	0.05350773	0.0204711	img_100.jpg
5	0.20709163	0.199017137	0.21774447	0.00185473	0.008231587	0.000766106	0.163181499	0.01637148	0.16269439	0.023047	img_1000.jpg
6	0.175210372	0.012817558	0.01275373	0.31079972	0.193856522	0.039003082	0.014804712	0.02128631	0.03555393	0.1839141	img_100000.jpg
7	0.066754185	0.004148643	0.011661227	0.38791528	0.246790305	0.091440424	0.00732243	0.00449907	0.01291451	0.1665539	img_100001.jpg
8	0.558428407	0.200665295	0.044377107	0.00596246	0.105209179	0.00044312	0.008159002	0.0180533	0.04710896	0.0115931	img_100002.jpg
9	0.082979806	0.401007831	0.161996543	0.01900433	0.076937504	0.03402029	0.032946624	0.09416454	0.05413698	0.0428055	img_100003.jpg
10	0.097132497	0.162630737	0.177345708	0.03061308	0.209266275	0.023033135	0.056169502	0.04272766	0.08698128	0.1141001	img_100004.jpg
11	0.28164953	0.090358436	0.017778689	0.09481884	0.347082287	0.002985022	0.021458311	0.09659841	0.02591923	0.0213512	img_100005.jpg
12	0.100105152	0.063773118	0.240964964	0.0149455	0.127637669	0.008759942	0.030784363	0.06704995	0.2473812	0.0985981	img_100007.jpg
13	0.098371975	0.044518836	0.018855775	0.00045875	0.0069078	0.000130684	0.027039766	0.10871931	0.66447854	0.0305186	img_100008.jpg
14	0.076186828	0.120338187	0.454713255	0.00365128	0.028718118	0.004026515	0.059574679	0.0205585	0.1888504	0.0433822	img_100009.jpg
15	0.317504793	0.043602373	0.042683765	0.0529712	0.051424917	0.027594516	0.037436631	0.03776096	0.17071207	0.2183088	img_10001.jpg
16	0.284323633	0.064432383	0.229552388	0.00126	0.004125459	0.001377248	0.118908249	0.00523022	0.25484163	0.0359488	img_100010.jpg
17	0.225687832	0.078612857	0.057668008	0.15373568	0.270010233	0.010958361	0.060918421	0.06245244	0.02806059	0.0518956	img_100011.jpg
18	0.083719239	0.121534482	0.080285802	0.06169575	0.067154378	0.089306034	0.14590548	0.27195457	0.0334818	0.0449625	img_100012.jpg
19	0.177521318	0.02478037	0.011868466	0.26886106	0.259722948	0.04115985	0.017438287	0.03731146	0.04665904	0.1146772	img_100013.jpg
20	0.183497593	0.246879488	0.289851844	0.01052741	0.053372037	0.003469787	0.055687286	0.02082746	0.09986877	0.0360183	img_100014.jpg

Figure 6 Snapshot Of Testing Phase

VIII. CONCLUSION

This paper has presented an overview of different driver distraction classifiers based on ML techniques . Our project will predict all sorts of distracted state of the driver and thus this application could be used in order to acquaint driver for wrong driving deeds which could finally help in accident prevention. This project can be implemented in developing systems that would be able to alert the driver if found distracted and help in preventing the accident. This project works on classifying images but it could be extended for real time monitoring application that could detect a distracted state in a video feed and generate an alarm

REFERENCES

1. S. McEvoy, M. Stevenson, and M. Woodward, *The impact of driver distraction on road safety: results from a representative survey in two Australian states*, Injury Prevention, vol. 11, no. 2, pp. 242247, 2006 .

2. M. A. Regan, C. Hallet, and C. P. Gordon, Driver distraction and driver inattention: Denition , relationship and taxonomy, Accid. Anal. Prev. J., vol. 43, no. 5, pp. 17711781, Sep. 2011.

3. Y. Zhang, Y. Owechko, and J. Zhang, "Driver cognitive workload estimation*: A data-driven perspective," in Proc. IEEE Intell. Transp. Syst. Conf*., Washington, DC, 2004, pp. 642647.

