KLE Technological University

Creating Value
Leveraging Knowledge

School
of
Electronics and Communication

VIII Semester Capstone Project on

EEG BASED DRIVER STATE ANALYSIS USING DEEP NEURAL NETWORK

By:

1. **Chandrika C.Kulkarni** USN: 01FE15BEC056
2. **Chirag T.Lodaya** USN: 01FE15BEC059
3. **Geetanjali S.Yadav** USN: 01FE15BEC067
4. **Girish Betadur** USN: 01FE15BEC068

Semester: VIII

Under the Guidance of

Prof. Prabhavathi C.Nissimgoudar

K.L.E SOCIETY'S
KLE Technological University,
HUBBALLI-580031
2018-2019

SCHOOL OF ELECTRONICS AND COMMUNICATION

CERTIFICATE

This is to certify that project entitled **EEG based driver state analysis using deep neural network** is a bonafide work carried out by the student team of **Chandrika C.Kulkarni (01FE15BEC056), Chirag T.Lodaya (01FE15BEC059), Geetanjali S.Yadav(01FE15BEC067), Girish Betadur (01FE15BEC068)**. The project report has been approved, as it satisfies the requirements with respect to the minor project work prescribed by the university curriculum for BE (VIII semester) in School of Electronics and Communications of KLE Technological University for the academic year 2018-2019.

Prof. Prabhavathi C.Nissimgoudar	**Dr. Nalini C. Iyer**	**Prof.B.L.Desai**
Guide	**Head of School**	**Registrar**

External Viva:

Name of Examiners	**Signature with date**
1.	
2.	

ACKNOWLEDGMENT

The sense of contentment and relation that accompanies the successful completion of this project would be incomplete without mentioning the names of the people who have helped us in accomplishing this project. And people whose constant guidance, support and encouragement resulted in its realization.

We express our sincere gratitude to our Vice Chancellor of KLE Technological University, Dr. Ashok Shettar to have given us the opportunity in exploring ourselves in Mini project.

We take this opportunity to thank our respected Principal of KLE Technological University, Dr.P.G.Tewari for providing a healthy environment in the college which helped in concentrating on the task.

We express deep sense of gratitude to our H.O.D of School of Electronics and Communication Engineering, Dr. Nalini Iyer for providing us an oppotunity to undertake this unique project.

We sincerely thank our faculty in charge Prof.Prabhavathi C.Nissimgoudar for her constant support and suggestions.

Finally, we sincerely thank all the teaching and non-teaching staff of School Of Electronics and Communications and also those who have helped us directly or indirectly in completing this project. We also offer deep gratitude to our parents who have appreciated, encouraged and assisted in our endeavour.

-The project team

ABSTRACT

Driver fatigue is a serious problem resulting in many thousands of road accidents each year. It is not possible to calculate the exact number of sleep related accidents but research shows that driver fatigue may be a contributory factor in up to 20 percent of road accidents, and up to one quarter of fatal and serious accidents.These types of crashes are about 50 percent more likely to result in death or serious injury as they tend to be high speed impacts because a driver who has fallen asleep cannot brake or swerve to avoid or reduce the impact.Sleepiness reduces reaction time (a critical element of safe driving). It also reduces vigilance, alertness and concentration so that the ability to perform attention-based activities (such as driving) is impaired. Nearly 1.3 million people die in road crashes each year, on average 3,287 deaths a day.In order to reduce road accidents occurring due to drowsiness of the driver, it is necessary to continuously monitor driver alertness and alarm when necessary. In this project we propose a strategy to analyse the alertness of the driver and notify accordingly.To achieve this goal, Deep Learning(DL) is used for characterizing and detecting the state of the driver based on EEG data.

Contents

List of Figures

Chapter 1

Introduction

Nearly 1.3 million people die in road crashes each year, on average 3,287 deaths a day.The National Highway Traffic Safety Administration estimates that drowsy driving was responsible for 72,000 crashes, 44,000 injuries, and 800 deaths in 2013. However, these numbers are underestimated and up to 6,000 fatal crashes each year may be caused by drowsy drivers. The causes of road accidents are many. Improper road structures, vehicle malfunctioning,carefree driving and driver drowsiness. Driver fatigue is a serious problem resulting in many thousands of road accidents each year. It is not possible to calculate the exact number of sleep related accidents but research shows that driver fatigue may be a contributory factor in up to 20 percent of road accidents, and up to one quarter of fatal and serious accidents.These types of crashes are about 50 percent more likely to result in death or serious injury as they tend to be high speed impacts because a driver who has fallen asleep cannot brake or swerve to avoid or reduce the impact.Sleepiness reduces reaction time (a critical element of safe driving). It also reduces vigilance, alertness and concentration so that the ability to perform attention-based activities (such as driving) is impaired. The speed at which information is processed is also reduced by sleepiness. The quality of decision-making may also be affected.It is clear that drivers are aware when they are feeling sleepy, and so make a conscious decision about whether to continue driving or to stop for a rest. It may be that those who persist in driving underestimate the risk of actually falling asleep while driving. Or it may be that some drivers choose to ignore the risks (in the way that drink drivers do).

In order to reduce road accidents occurring due to drowsiness of the driver, it is nec-

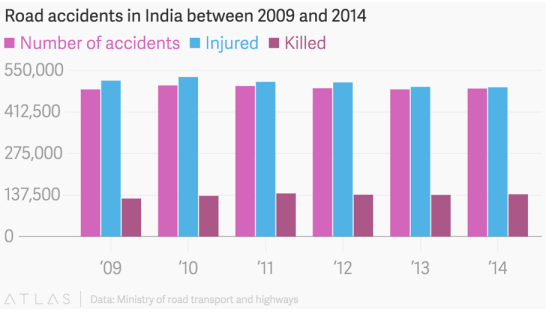


Figure 1.1: A graph depicting road accidents in India between 2009 and 2004 showing high number of accidents and no considerable change in the death rate in the consequent years.

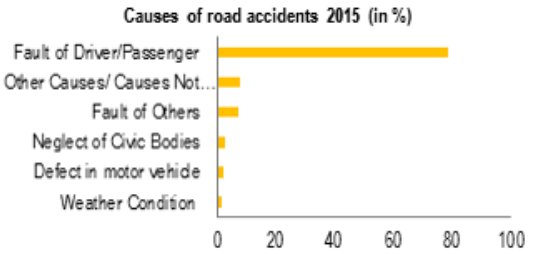


Figure 1.2: A graph depicting causes of road accidents and their contribution to the same showing most common reason being fault of driver/passenger

essary to continuously monitor driver alertness and alarm when necessary. In this project we propose a strategy to analyze the alertness of the driver and notify accordingly.To achieve this goal, Deep Learning(DL) is used for characterizing and detecting the state of the driver based on EEG data.

1.1 Motivation

- The National Highway Traffic Safety Administration estimates that drowsy driving was responsible for72,000 crashes, 44,000 injuries, and 800 deaths in 2013. However, these numbers are underestimated and up to 6,000 fatal crashes each year may be caused by drowsy drivers.
- The economic loss from the road accidents, Indiastransport minister reckons, is close to 3percent of the countrys GDP. That is Rs.55,000 crore every year.

1.2 Objectives

The objective of this project are:

- To develop an optimized algorithm for the detection of driver drowsiness based on ElectroEncephaloGram.
- To develop a real time hardware system for the same.

1.3 Problem statement

To develop a deep learning algorithm to detect driver drowsiness and to develop a real time hardware system for the same.

Chapter 2

Background Work

This chapter highlights the work done in detecting the driver drowsiness. We discuss regarding the different approaches to detect driver drowsiness for example steering and lane data and feature based, vision based and E.E.G. based approaches. Section 2.1 briefs regarding the driver alertness determination based on steering and lane features by Fabian Friedrichs, Bin Yang [1] .Section 2.2 briefs regarding vision based driver alertness detection by Sayani Ghosh, Tanaya Nandy and Nilotpal Manna[2] and Anirban Dasgupta, Anjith George, S L Happy and Aurobinda Routray, Member, IEEE [3] . The last section i.e., section 2.3 highlights regarding E.E.G. based driver alertness detection by Subhrajit Roy, Isabell Kiral-Kornek, and Stefan Harrer, IEEE Senior Member[4] and SLEEPNET architecture[5].

2.1 Steering and lane data and feature based driver drowsiness detection

Fabian Friedrichs, Bin Yang [1] proposed drowsiness monitoring by steering and lane data based features under real driving conditions in which new measures (features) for detecting drowsiness are proposed in addition to promising features in literature. Most studies in literature are based on driving simulator data, whereas this article focuses on real world driving. External inuences such as road condition, road bumps and crosswind are furthermore taken into account. The presented results are based on a large selection of the Mercedes-Benz drowsiness database which covers over 1.2 million kilometers of measurements. Features are analyzed for their correlation with the subjective Karolinska Sleepiness Scale (KSS). The performance of a combination of features is assessed by sophisticated classiers and dimension reduction techniques. Even after these improvements, the classication results do not reach the results obtained in a driving simulator.

2.2 Vision based driver drowsiness detection

Sayani Ghosh, Tanaya Nandy and Nilotpal Manna[2] in their work titled Real Time Eye Detection and Tracking Method for Driver Assistance System, propose a non-intrusive prototype computer vision system for monitoring a drivers vigilance in realtime. Eye tracking is one of the key technologies for future driver assistance systems since human eyes contain much information about the drivers condition such as gaze, attention

level, and fatigue level. One problem common to many eye tracking methods proposed so far is their sensitivity to lighting condition change. This tends to signicantly limit their scope for automotive applications. This paper describes real time eye detection and tracking method that works under variable and realistic lighting conditions. It is based on a hardware system for the real-time acquisition of a drivers images using IR illuminator and the software implementation for monitoring eye that can avoid the accidents.

Anirban Dasgupta, Anjith George, S L Happy and Aurobinda Routray, Member, IEEE [3] in their work titled A Vision Based System for Monitoring the Loss of Attention in Automotive Driver, propose a solution in which the face is detected using Haar-like features and tracked using a Kalman Filter. The Eyes are detected using Principal Component Analysis (PCA) during day time and the block Local Binary Pattern (LBP) features during night. Finally the eye state is classified as open or closed using Support Vector Machines (SVM). In-plane and off-plane rotations of the drivers face have been compensated using Affine and Perspective Transformation respectively. Compensation in illumination variation is carried out using Bi-Histogram Equalization (BHE). The algorithm has been cross-validated using brain signals and finally been implemented on a Single Board Computer (SBC) having Intel Atom processor, 1 GB RAM, 1.66 GHz clock, x86 architecture, Windows Embedded XP operating system. The system is found to be robust under actual driving conditions.

2.3 E.E.G. based driver drowsiness detection

Subhrajit Roy, Isabell Kiral-Kornek, and Stefan Harrer, IEEE Senior Member[4] proposed an EEG based approach to determine driver alertness under the title, Deep Learning Enabled Automatic Abnormal EEG Identication. In this work, the focus is on one of the early decisions made in this process which is identifying whether an EEG session is normal or abnormal.There is no extraction of hand-engineered features but employ deep neural networks that automatically learn meaningful representations.They also undertook a holistic study by exploring various pre-processing techniques and machine learning algorithms for addressing this problem and compare their performance. We have used the recently released TUH Abnormal EEG Corpus dataset for evaluating the performance of these algorithms and show that modern deep gated recurrent neural networks achieve 3.47percent better performance than previously reported results.

In SLEEPNET:Automated Sleep Staging System via Deep Learning,the authors[5] propose SLEEPNET (Sleep EEG neural network), a deployed annotation tool for sleep staging.SLEEPNET uses a deep recurrent neural network trained on the largest sleep physiology database assembled to date, consisting of PSGs from over 10,000 patients from the Massachusetts General Hospital (MGH) Sleep Laboratory. SLEEPNET achieves human-level annotation performance on an independent test set of 1,000 EEGs, with an average accuracy of 85.76percent and algorithm-expert inter-rater agreement (IRA) of = 79.46percent, comparable to expert-expert IRA.The pipeline of SLEEPNET is shown in figure 2.1 This architecture was implemented for our data and the results are shown in fig 5.1 and fig 5.2

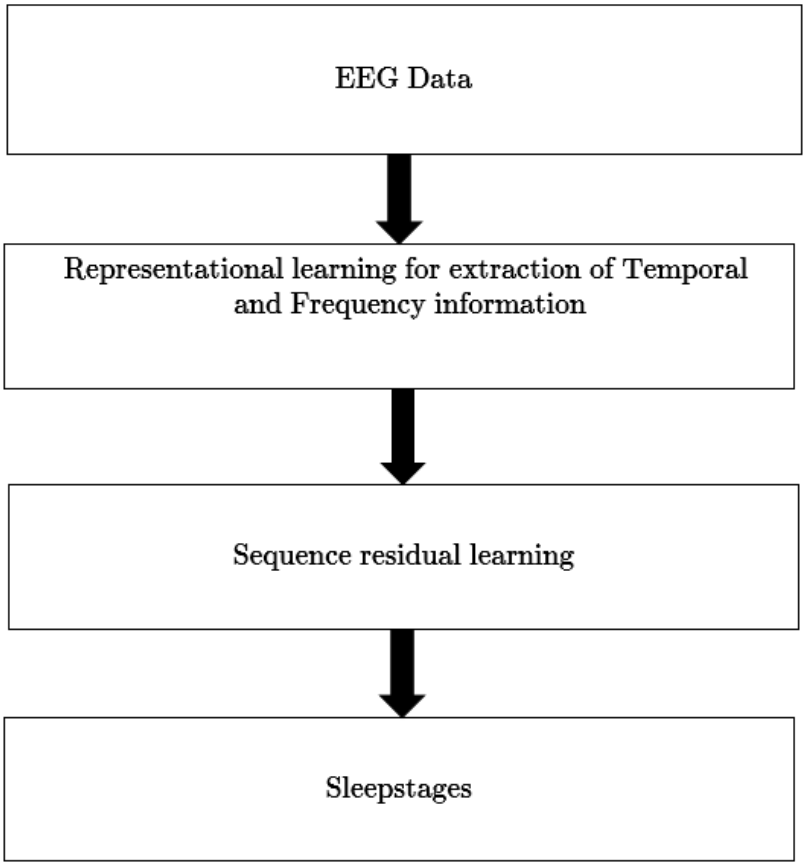


Figure 2.1: Pipeline proposed in Deep sleep net

Chapter 3

Pipeline for Detection of driver state

This chapter comprises of the entire framework for the Detection of driver state and its description. Section 3.1 contains the proposed framework and step by step explanation of the same.The description of data acqusition , data processing and output notification is given in section 3.1.1, section 3.1.2 and section 3.1.2 respectively.

This chapter contains the details regarding how we intend to design an efficient system to retrieve the EEG data and predict the driver state with appreciable results. Section 3.1 gives an outline of the design we propose for the same

3.1 Proposed Framework

The framework of the system is depicted in fig 3.1.The whole idea being classification of EEG data using Convolutional Neural network.As any other system ,the framework comprises of three main phases namely data acquisition, data processing and output notification.

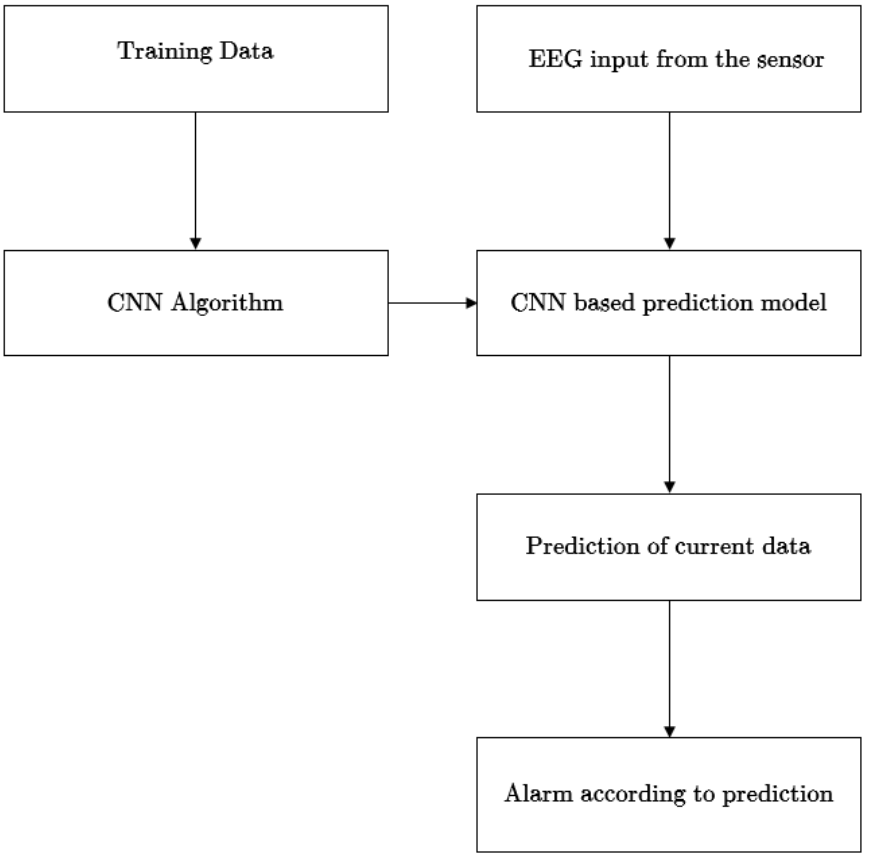


Figure 3.1: Pipeline for detection of driver state

3.1.1 Data Acquisition

As discussed earlier, the input to the proposed algorithm is EEG. There are various means of acquiring EEG values. Most of the sensors available in the market give processed EEG signal as output. A few popular sensors are mentioned below(details of which are mentioned in chapter 4).

- Think Gear AM/AT.
- Mind Wave headset.

Note that we have used a pre-recorded EEG data for the training and testing the accuracy of the CNN algorithm(during algorithm design phase) where as the EEG sensor output will be fed to the algorithm in the real time implementation(testing).

3.1.2 Data processing

This step of the framework comprises of Convolutional Neural Network that classifies the input data as driver alert or driver drowsy. In deep learning, Convolutional Neural Network is a class of deep, feed forward artificial neural networks that has been successfully applied.A typical CNN architecture is shown in fig 3.2

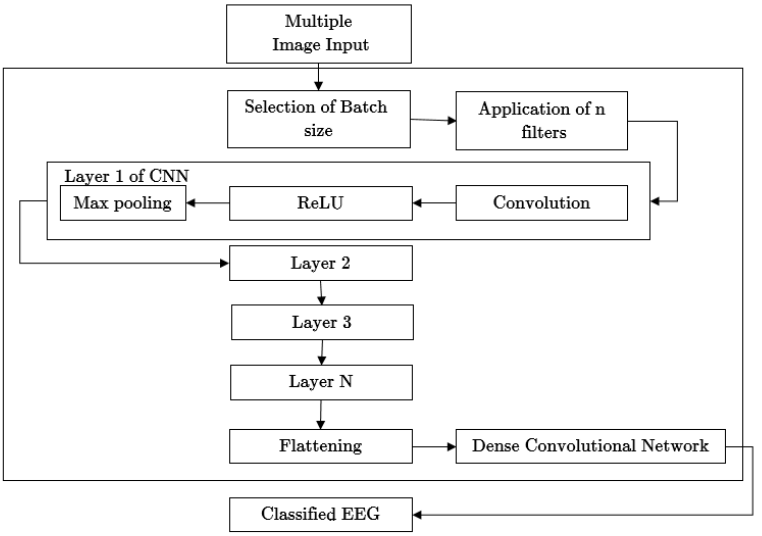


Figure 3.2: A Typical Convolutional Neural Network

- ***Convolutional layer***:

  This is the initial step of CNN. A set of filters of defined size and stride has been used. The filters help us to extract the features like color blob detectors, edges of the image etc.

  ReLU(Rectified Linear Unit):

It is a method in which, the negative values are normalized to zero. It is found that it greatly accelerates the convergence of Gradient descent compared to the sigmoid function. It is argued that it is due to its linear, non saturation form.

Pooling:

It is a technique used in CNN in which the maximum value in the patch of the defined size is extracted. The different types of pooling are max pooling, min pooling, average pooling etc.

Fully connected layer (Dense layer):

Neural networks are modeled as a collection of neurons that are connected in an acyclic graph. In other words, the output of some neurons can be fed as input to other neurons. Cycles are not allowed since that would imply an infinite loop in forward pass of a network.
This layer consists of various sub-layers such as the input,output and hidden layers.

3.1.3 Output Notification

The prediction is displayed on GUI.

Chapter 4

Implementation Details

In the previous chapter we saw the proposed design for reviving the EEG data and indicating the driver state. In this chapter we describe the implementation details of the same. Section 4.1 highlights the details of the sensor namely ThinkGear AM/AT and MindWave headset. Section 4.2 comprises of brief regarding feature reduction and feature selection techniques.Section 4.3 comprises of different architectures developed,trained and tested. Section 4.4 comprises of the details as to how the driver is alarmed or notified if detected drowsy.

4.1 EEG Sensors

- ThinkGear AM/AT: The ThinkGear AM (TGAM) EEG sensor PCB module powers over 1 million consumer EEG devices around the globe.TGAM is the core of brain-wave sensing technology. TGAM allows NeuroSky partners to bring EEG-based consumer technologies to market quickly and efficiently. The TGAM is the worlds most popular EEG solution. Together with dry-electrode, it senses the signals from the human brain, filters out extraneous noise and electrical interference and converts to digital power. This brain-power can be used in health and wellness, education and entertainment.Embedded within the TGAM, is the TGAT chip, a powerful, fully integrated single chip EEG sensor. The chip comes programmed with NeuroSky eSense, A/D, amplification off head detection, and noise filtering for EMG and 50/60Hz AC powerline interference.

- MindWave Headsdet: The family of NeuroSky MindWave headsets are designed to be used by developers to get to market quickly with complete EEG-monitoring products.The MindWave Mobile 2 headset turns your computer into a brain activity monitor. The headset safely measures brainwave signals and monitors the attention levels of individuals as they interact with a variety of different apps. This headset is useful for OEMs and developers building apps for health and wellness, education and entertainment. The MindWave family consists of MindWave and MindWave Mobile 2 headsets. The MindWave is designed for PCs and Mac, while the MindWave Mobile 2 is compatible with PCs, Mac and mobile devices like the iPhone, iPad, and Android. If you want a mobile compatible device, check out the MindWave Mobile 2. Both headsets share the following characteristics.NeuroSky ThinkGear ASIC chip is priced to power mass adoption in health and wellness, educational and entertainment devices, popular EEG technology.In this project we have used

Figure 4.1: Neurosky MindWave Sensor

mindwave headset comparison with The TGAM/TGAT, mindwave heasets are user friendly and has better performance.

The raw EEG data is obtained from the brain using NeuroSky Mind wave mobile sensor, a commercially available headset which is noninvasive type of brain computer interface. The sensor is placed on the scalp to acquire raw EEG information. Such headsets contain NeuroSky ThinkGear technology, which measures the analog electrical signals, commonly referred to as brainwaves, and processes them into digital signals. The ThinkGear Connector (TGC) runs as a background process on the computer and is responsible for directing headset data from the serial port to an open network socket. It includes the Thinkgear module and EEG sensor that touches the forehead, and the reference points located on the ear clip. Thinkgear module contains the onboard chip that processes all of the data and provides this data to software and applications in digital form. Any language or framework that contains a socket library should be able to communicate with it and here MATLAB code and its library are used to acquire the raw EEG data. The raw EEG data captured from the headset is used for further processing.

4.2 Feature Reduction and Selection

For the training of the neural network two main aspects are to be considered namely features reduction and feature selection.Not all features of data can be used for training because it may lead to overfitting. Feature reduction algorithms like Principal Component Analysis gives us the number of features to be considered and the feature selction algorithms like Random forest and Recursive Feature Elimination algorithms give us the specific features to be taken into consideration.

4.2.1 Feature Reduction

This section describes one of the popular feature reduction techniques,namely, Principal Component Analysis.

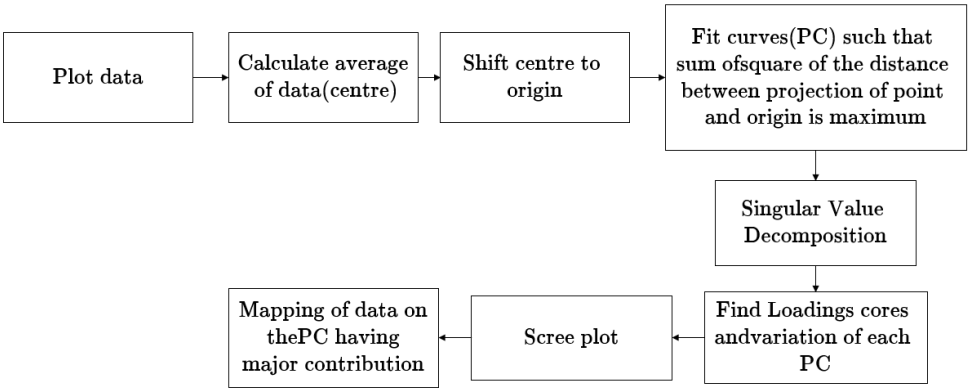


Figure 4.2: Principal Component Analysis Algorithm

- Step 1: Standardization The aim of this step is to standardize the range of the continuous initial variables so that each one of them contributes equally to the analysis.sensitive regarding the variances of the initial variables. if there are large differences between the ranges of initial variables, those variables with larger ranges will dominate over those with small ranges, which will lead to biased results. So, transforming the data to comparable scales can prevent this problem. Mathematically, this can be done by subtracting the mean and dividing by the standard deviation for each value of each variable. Once the standardization is done, all the variables will be transformed to the same range.

- Step 2: Covariance Matrix computation The aim of this step is to understand how the variables of the input data set are varying from the mean with respect to each other. Because sometimes, variables are highly correlated in such a way that they contain redundant information.The covariance matrix is a(pp) symmetric matrix (where p is the number of dimensions) that has as entries the covariances associated with all possible pairs of the initial variables. Since the covariance of a variable with itself is its variance (Cov(a,a)=Var(a)) in the main diagonal (Top left to bottom right) we actually have the variances of each initial variable. since the covariance is commutative (Cov(a,b)=Cov(b,a)), the entries of the covariance matrix are symmetric with respect to the main diagonal, which means that the upper and the lower triangular portions are equal. If the sign of the covariance:
  - if positive then: the two variables increase or decrease together (correlated)
  - if negative then: One increases when the other decreases (not correlated)

- Step 3: Compute the eigenvectors and eigenvalues of the covariance matrix to identify the principal components Eigenvectors and eigenvalues are the linear algebra concepts that we need to compute from the covariance matrix in order to determine theprincipal componentsof the data. Principal components are new variables that are constructed as linear combinations or mixtures of the initial variables. These combinations are done in such a way that the new variables (i.e., principal components) are uncorrelated and most of the information within the initial variables is squeezed or compressed into the first components.Geometrically speaking, principal

components represent the directions of the data that explain amaximal amount of variance. The relationship between variance and information here, is that, the larger the variance carried by a line, the larger the dispersion of the data points along it, and the larger the dispersion along a line, the more the information it has. Principal components are constructed in such a manner that the first principal component accounts for thelargest possible variance in the data set. The second principal component is calculated in the same way, with the condition that it is uncorrelated with (i.e., perpendicular to) the first principal component and that it accounts for the next highest variance. This continues until a total of p principal components have been calculated, equal to the original number of variables. every eigenvector has an eigenvalue. And their number is equal to the number of dimensions of the data. the eigenvectors of the Covariance matrix are actuallythedirections of the axes where there is the most variance(most information) and that we call Principal Components. And eigenvalues are simply the coefficients attached to eigenvectors, which give theamount of variance carried in each Principal Component.

- Step 4: Feature vector the feature vector is simply a matrix that has as columns the eigenvectors of the components that we decide to keep.
- Step 5: Recast the data along the principal components axes the aim is to use the feature vector formed using the eigenvectors of the covariance matrix, to reorient the data from the original axes to the ones represented by the principal components. This can be done by multiplying the transpose of the original data set by the transpose of the feature vector.

4.2.2 Feature Selection

There are two popular feature selection algorithms namely Random forest and Recursive Feature Elimination described in the following sections.

Recursive Feature Elimination

A recursive neural network is a kind of deep neural network created by applying the same set of weights recursively over a structured input, to produce a structured prediction over variable-size input structures, or a scalar prediction on it, by traversing a given structure in topological order.

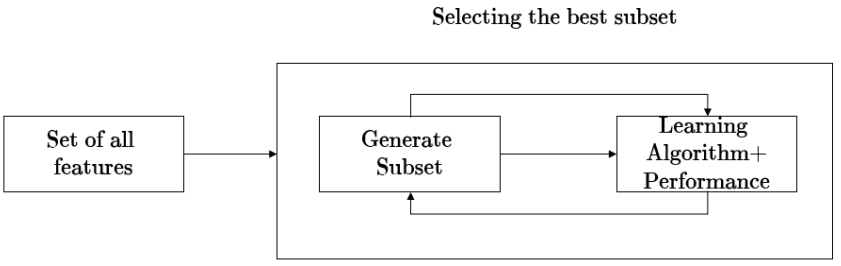


Figure 4.3: Recursive Feature Elimination algorithm

Random forest

Random Forest is a supervised learning algorithm. As the name suggests, it creates a forest and makes it somehow random. The forest it builds, is an ensemble of Decision Trees, most of the time trained with the bagging method. The general idea of the bagging method is that a combination of learning models increases the overall result.To say it in simple words: Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.One big advantage of random forest is, that it can be used for both classification and regression problems, which form the majority of current machine learning systems.

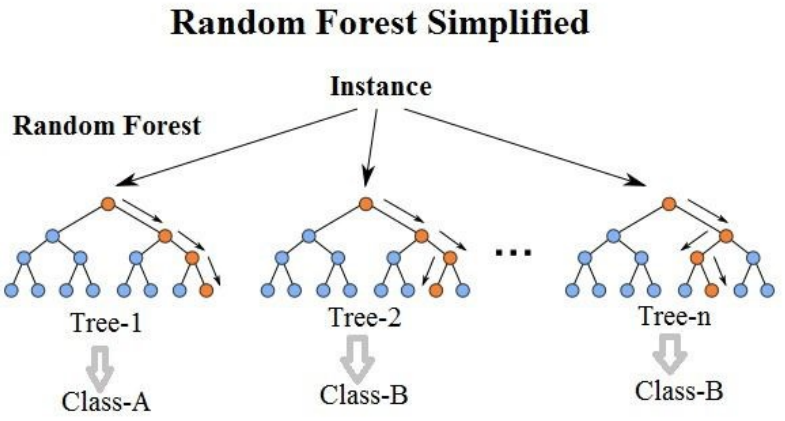


Figure 4.4: Random Forest Algorithm

4.3 Architecture of the neural network

Three different architectures for the classification of the data have been tried upon as mentioned below. Refer to the table below that comprises of description of all the three architectures that is., number of layers,filter size, number, epochs,accuracy,validation and test accuracy and the type of optimiser used.

	Architecture	Layer	Filter	FilterSize	Epochs	Optimizer
Method 1	Convolution	1	32	3	5	Adam
	Dropout					
	Convolution	2	64	3		
	Maxpool	3		2		
	Flattening					
	Dense Neural Network	3	64 128 128			
Method 2	Convolution	1	16	3	5	Adam
	Convolution	2	32	3		
	Convolution	3	64	3		
	Maxpool	3		2		
	Dense Neural Network	4,5,6	32 64 128			
Method 3	Convolution	1	16	3	5	Adam
	Convolution	2	32	3		
	Convolution	3	64	3		
	Maxpool	3		2		
	Dense Neural Network	4,5,6	64 128 128			

Adam Optimiser

Adam is an optimization algorithm that can used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data. Adam is derived from adaptive moment estimation. The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients. Adaptive Gradient Algorithm (AdaGrad) that maintains a per-parameter learning rate that improves performance on problems with sparse gradients. Instead of adapting the parameter learning rates based on the average first moment (the mean) as in RMSProp, Adam also makes use of the average of the second moments of the gradients. the algorithm calculates an exponential moving average of the gradient and the squared gradient, and the parameters beta1 and beta2 control the decay rates of these moving averages. The initial value of the moving averages and beta1 and beta2 values close to 1.0 (recommended) result in a bias of moment estimates towards zero. This bias is overcome by first calculating the biased estimates before then calculating bias-corrected estimates.

4.3.1 Processing Unit

Choice of hardware to deploy a neural network requires a special attention. Training and testing both could be performed on the microcontrollers having higher RAM and a high performance processor. High performance processor is costly. Thus training on any personal computer and deploying the module file on the regular microcontroller like Raspberry Pi for testing clearly seems to be an intelligent choice. Pi is a single-board small scale computer in the size little bigger than a credit card, it packs enough power to

run games, word processor like open office, image editor like Gimp and any program of similar magnitude.

System specifications

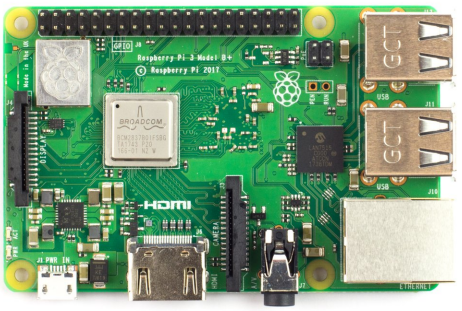


Figure 4.5: Raspberry Pi 3B

- Processor:Broadcom BCM2387 chipset, 1.2GHz Quad-Core ARM Cortex-A53 (64Bit), 802.11 b/g/n Wireless LAN and Bluetooth 4.1 (Bluetooth Classic and LE)
- GPU: Dual Core Video Core IV Multimedia Co-Processor. Provides Open GL ES 2.0,hardware-accelerated OpenVG, and 1080p30 H.264 high-profile decode. Capable of 1 G pixel/s, 1.5Gtexel/s or 24GFLOPs with texture filtering and DMA infrastructure.
- Memory: 1GB LP DDR2
- Power: Micro USB socket 5V1, 2.5A
- Ethernet: 10/100 BaseT Ethernet socket.
- Video and audio output: HDMI (rev 1.3 and 1.4) , Audio Output 3.5mm jack.
- GPIO Connector:40-pin 2.54 mm (100 mil) expansion header: 2x20 strip , Providing 27 GPIO pins as well as +3.3 V, +5 V and GND supply lines.
- Camera Connector: 15-pin MIPI Camera Serial Interface (CSI-2).
- Memory Card Slot: Push/pull Micro SDIO

Virtual Environment

Execution of the program takes place is the virtual environment.A virtual environment is a tool that helps to keep dependencies required by different projects separate by creating isolated python virtual environments for them. This is one of the most important tools that most of the Python developers use.If we are using a python process and it uses a heavy libraray (In our case keras, tensorflow) while the raspberry pi3 itself has several processes using the resources of the operating system. In such situations virtual environment can be really useful to maintain dependencies of these both situtations.By default, every process on the system will use the same directories to store and retrieve site packages (third party libraries).Now, in the above case of two processes, by creating a virtual environment for the process(Our application) we are cutting off the dependency of our application process with that of the default processes of the Raspberry Pi.Thus by using virtual environment for application processes, the load on OS is reduced i.e. the dependencies of each process is isolated from the system and each other.

4.4 Output Indication

The driver state is indicated on a GUI developed using python.The screenshot of the GUI are shown in fig 4.6 The GUI as shown has an option to upload the data to be tested and a start button to initiate the classification. If the data is classified as sleepy i.e., if the driver is detected drowsy , the red sector glows else the green sector glows indicating that the driver is alert.The GUI screenshots of the results for both sleep and wake test inputs are shown in fig 4.7 and fig 4.8

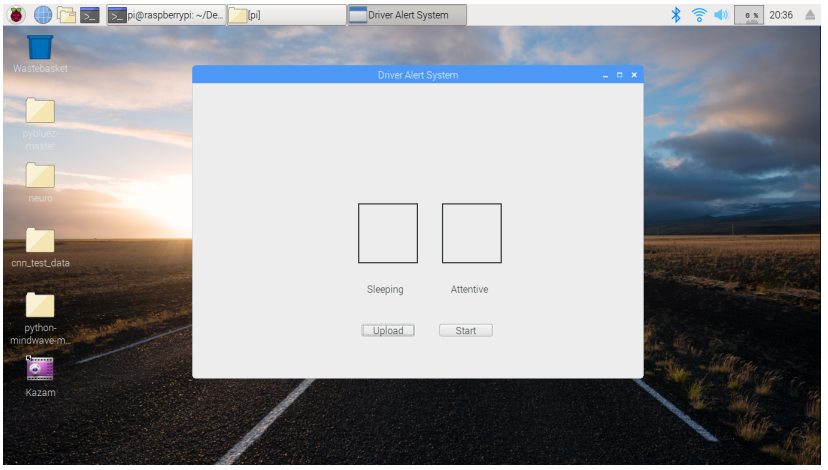


Figure 4.6: GUI

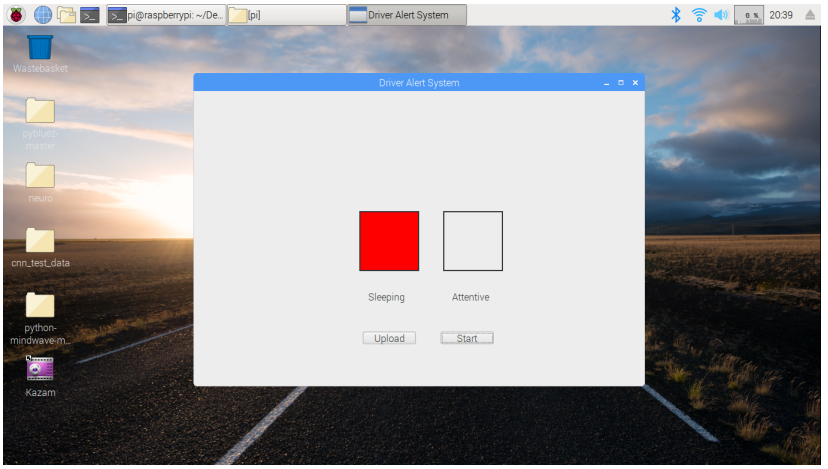


Figure 4.7: GUI showing results for Sleep data

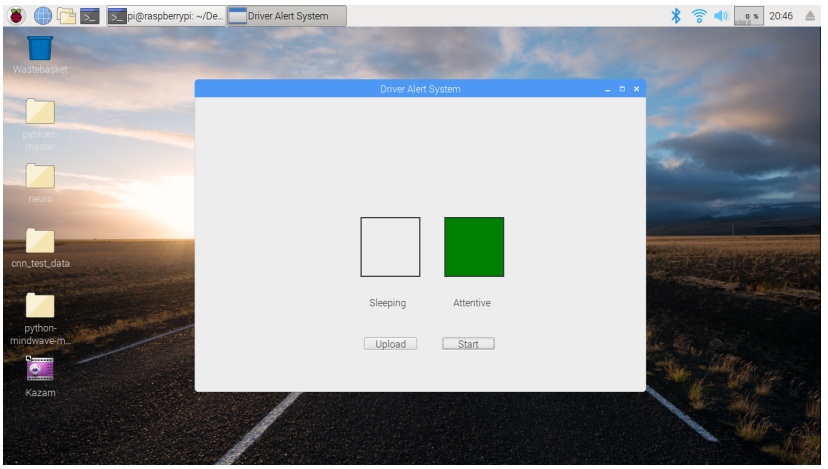


Figure 4.8: GUI showing results for Wake data

4.5 Optimisation

- Reduction of CNN layer from 3 to 2
  - The layers in a CNN algorithm are the main components that decide speed and performance
  - By trial and error we reduced the number of layers of the CNN algorithm and checked for change in accuracy
  - Thus with a basic 4-5percent change in accuracy we decided to reduce the number of layers from 3 to 2 (Although there is trade off with accuracy to increase speed and to be able to run on basic hardware)
- Reduction of filters i.e. 32,64 from 16,32,64
  - The filter is one of the main component which decided speed and accuracy in the CNN algorithm

  - By trial and error we reduced the number of components of the filter of the CNN algorithm and checked for change in accuracy
  - Thus with a basic 1-2percent change in accuracy we decided to reduce filter size from to 2 (,Although there is trade off with accuracy to increase speed and to be able to run on basic hardware)

- Using dropout technique This is a way to Prevent Neural Networks from overfitting. Dropout is a technique where randomly selected neurons are ignored during training. They are dropped-out randomly. After this, accuracy is checked to see whether important neurons were dropped and hence changes are made accordingly by trial and error

- Using virtual environment (As required packages are large in size- Keras, Tensorflow) The main advantage of this is that, virtual environment for application processes, the load on OS is reduced i.e. the dependencies of each process is isolated from the system and each other.

- Using flash drive for installing modules and packages The flash-drive acts as a secondary memory (RAM) for the controller. When modules or packages are being installed in a micro controller and in case these are very large in size and complicated, this flash drive helps reduce the load on OS and the controller. This basically prevents crash of important components of the same.

Chapter 5

Results and Discussions

This chapter comprises of the outcomes of the different architectures proposed in section 4.3 . Fig 5.1 and fig 5.2 are the implementation results of Deep Sleep Net mentioned in section 2.3. The results of feature selection and feature reduction algorithms have been shown in fig 5.3. The training results hav bn shown in fig 5.4 .The results of the proposed architechture for the sleep and wake test inputs have been shown in fig 5.5 and fig 5.6 respectively. Fig 5.7 shows the time profiling and fig 5.8 shows memory profiling.

```
_________________________________________________________________
dense_113 (Dense)            (None, 64)                4160
_________________________________________________________________
dense_114 (Dense)            (None, 128)               8320
_________________________________________________________________
dense_115 (Dense)            (None, 2)                 258
=================================================================
Total params: 195,106
Trainable params: 195,106
Non-trainable params: 0
_________________________________________________________________
Test loss: 0.2192728966474533
Test accuracy: 0.8999999761581421
```

Figure 5.1: Output of LSTM

```
_________________________________________________________________
dense_89 (Dense)             (None, 64)                12352
_________________________________________________________________
dense_90 (Dense)             (None, 128)               8320
_________________________________________________________________
dense_91 (Dense)             (None, 2)                 258
=================================================================
Total params: 30,498
Trainable params: 30,498
Non-trainable params: 0
_________________________________________________________________
Test loss: 0.3453372120857239
Test accuracy: 0.8999999761581421
```

Figure 5.2: Output of dense layer

```
In [31]: accuracy_rfe=recursive_function(X,Y)
1
Accuracy: 52.10% (3.53%)

In [32]: accuracy_pca=principle_component_analysis(X,Y)
1
Accuracy: 51.40% (5.10%)

In [33]: random_forest_accuracy=random_forest(X,Y)
[[ 94   0]
 [  3 103]]
             precision    recall  f1-score   support

          0       0.97      1.00      0.98        94
          1       1.00      0.97      0.99       106

avg / total       0.99      0.98      0.99       200

0.985
```

Figure 5.3: Output of Principle Component Analysis

	Architecture	Layer	Filter	Filter Size	Epochs	Optimizer	Accuracy	Value Accuracy	Test Accuracy
Method 1	Convolution	1	32	3	5	Adam	89.99	0.9	90
	Dropout								
	Convolution	2	64	3					
	Maxpool	3		2					
	Flattening								
	Dense Neural Network	3	64 128 128						
Method 2	Convolution	1	16	3	5	Adam	80	0.8	80
	Convolution	2	32	3					
	Convolution	3	64	3					
	Maxpool	3		2					
	Dense Neural Network	4,5,6	32 64 128						
Method 3	Convolution	1	16	3	5	Adam	80	0.8	80
	Convolution	2	32	3					
	Convolution	3	64	3					
	Maxpool	3		2					
	Dense Neural Network	4,5,6	64 128 128						

Figure 5.4: Training results for different architechtures

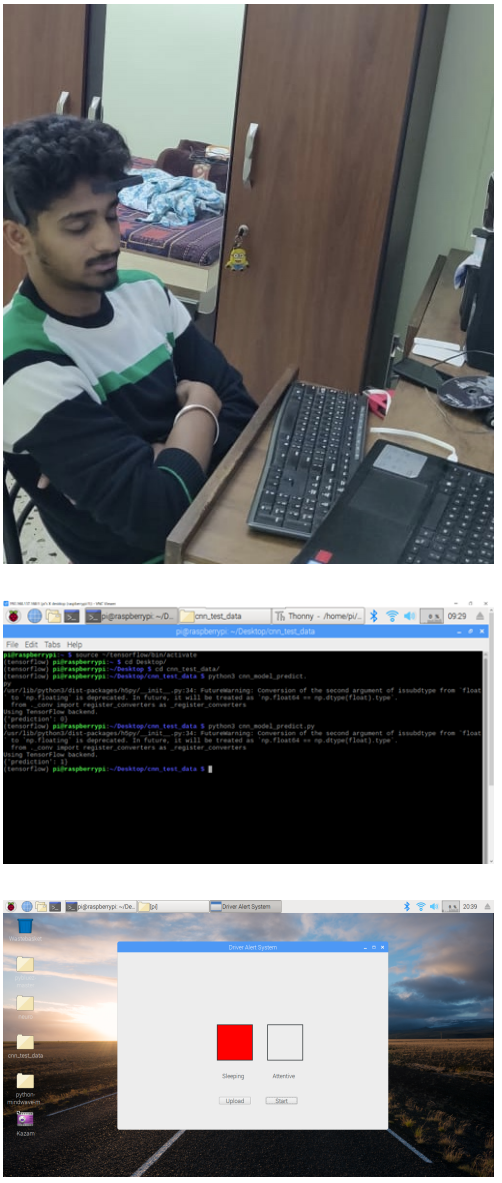
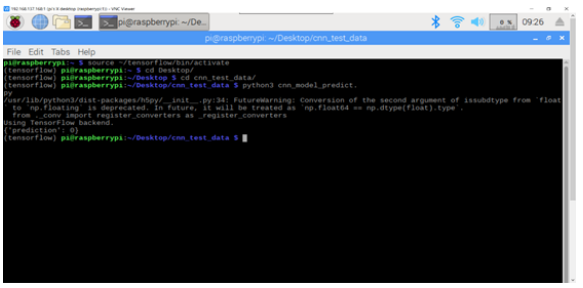


Figure 5.5: a)Drowsy driver b) Output of proposed architecture for sleep test data c)GUI for test data classified as Drowsy



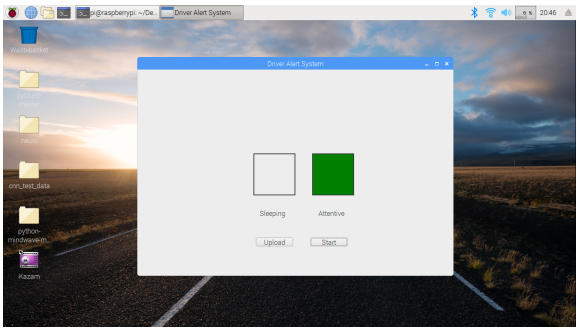


Figure 5.6: a)Alert Driver b) Output of proposed architecture for wake test data c)GUI for test data classified as Alert.

Parameter	Value
Time consumed for one Prediction	40-42 seconds
CPU time (user mode)	29-40 seconds
CPU time (system mode)	45-50 seconds
CPU time (IOwait time)	15-20 seconds
CPU time (IRQ wait)	0 seconds
CPU time (soft irq mode)	1-2 seconds

Figure 5.7: Time Profiling

Parameters	Value
Total	972MB
Available for user	530MB
Used	385-320MB
Buffered	166-200MB
Cached	200-230MB
Shared	148-200

Figure 5.8: Memory Profiling

Chapter 6

Conclusions and future scope

In this project, we propose a system which that detects the driver state ,indicate outcome and develop a real time hardware system for the same. From our results, we deduce certain conclusions and scope for future work.

6.1 Conclusion

In this project, we propose a system for driver alertness/sleep satuts in an automobile.

- We used EEG signals from the brain (using neurosky mindwave sensor) as a measure of drivers alertness.
- Using deep learning algorithms (CNN), we trained a model by pre-feeding 692 sets of drivers sleep status data and obtained a deep learning prediction model file.
- The same prediction file was dumped onto the hardware and the prediction algorithm runs successfully on the controller (Raspberry Pi 3)
- We tested the same by using 10 samples of sleepy and alert data to test our model for prediction, for which we obtained 90percent accuracy.

Thus our system implemented on hardware successfully predicts the sleep status of a driver in a an automobile.

6.2 Future Scope

- To interface controller and EEG signal sensor and to create a continuous real time prediction system.
- To improve the prediction time of the system by using high capacity controllers and advanced optimisation techniques.

Bibliography

[1] DROWSINESS MONITORING BY STEERING AND LANE DATA BASED FEATURES UNDER REAL DRIVING CONDITIONS, Chair of System Theory and Signal Processing, University of Stuttgart ,Pfaffenwaldring 47, 70550 Stuttgart, Germany, fabian.friedrichs, bin.yang@lss.uni-stuttgart.de

Author : Fabian Friedrichs, Bin Yang

[2] Real Time Eye Detection and Tracking Method for Driver Assistance System

Author : Sayani Ghosh, Tanaya Nandy and Nilotpal Manna

[3] A Vision Based System for Monitoring the Loss of Attention in Automotive Drivers

Author : Anirban Dasgupta, Anjith George, S L Happy and Aurobinda Routray, Member, IEEE

[4] Deep Learning Enabled Automatic Abnormal EEG Identication

Author : Subhrajit Roy, Isabell Kiral-Kornek, and Stefan Harrer, IEEE Senior Member

[5] SLEEPNET: Automated Sleep Staging System via Deep Learning

Author :Siddharth Biswal ,Joshua Kulas ,Haoqi Sun,Balaji Goparaju, M Brandon Westover ,Matt T Bianchi,Jimeng Sun.