Automatic Driver Fatigue Monitoring Using Hidden Markov Models and Bayesian Networks

by

Abdullah Rashwan

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2013

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The automotive industry is growing bigger each year. The central concern for any automotive company is driver and passenger safety. Many automotive companies have developed driver assistance systems, to help the driver and to ensure driver safety. These systems include adaptive cruise control, lane departure warning, lane change assistance, collision avoidance, night vision, automatic parking, traffic sign recognition, and driver fatigue detection.

In this thesis, we aim to build a driver fatigue detection system that advances the research in this area. Using vision in detecting driver fatigue is commonly the key part for driver fatigue detection systems. We have decided to investigate different direction. We examine the driver's voice, heart rate, and driving performance to assess fatigue level. The system consists of three main modules: the audio module, the heart rate and other signals module, and the Bayesian network module.

The audio module analyzes an audio recording of a driver and tries to estimate the level of fatigue for the driver. A Voice Activity Detection (VAD) module is used to extract driver speech from the audio recording. Mel-Frequency Cepstrum Coefficients, (MFCC) features are extracted from the speech signal, and then Support Vector Machines (SVM) and Hidden Markov Models (HMM) classifiers are used to detect driver fatigue. Both classifiers are tuned for best performance, and the performance of both classifiers is reported and compared.

The heart rate and other signals module uses heart rate, steering wheel position, and the positions of the accelerator, brake, and clutch pedals to detect the level of fatigue. These signals' sample rates are then adjusted to match, allowing simple features to be extracted from the signals, and SVM and HMM classifiers are used to detect fatigue level. The performance of both classifiers is reported and compared.

Bayesian networks' abilities to capture dependencies and uncertainty make them a sound choice to perform the data fusion. Prior information (Day/Night driving and previous decision) is also incorporated into the network to improve the final decision. The accuracies of the audio and heart rate and other signals modules are used to calculate certain CPTs for the Bayesian network, while the rest of the CPTs are calculated subjectively. The inference queries are calculated using the variable elimination algorithm. For those time steps where the audio module decision is absent, a window is defined and the last decision within this window is used as a current decision. The performance of the system is assessed based on the average accuracy per second.

A dataset was built to train and test the system. The experimental results show that the system is very promising. The performance of the system was assessed based on the average accuracy per second; the total accuracy of the system is 90.5%. The system design can be easily improved by easily integrating more modules into the Bayesian network.

Acknowledgments

First, I must thank ALLAH for His great mercy supporting me all the way till the end. If it were not for His help, I would not have reached this point.

With a deep sense of gratitude, I wish to express my sincere appreciation to my supervisor, Professor Mohamed Kamel, for his immense help throughout the work. Without his help, I could not have completed this work.

Many thanks to my colleagues in the Pattern Recognition and Machine Intelligence (PAMI) group for providing a great research environment. Special thanks to Celine Craye for her great help and useful discussions.

Finally, I will never find words enough to express the gratitude that I owe to my wife Nour, and my parents for their love, encouragement, and constant support. I also cannot forget my children, Yahia and Omar, who have brought much joy to my heart during the hard times.

Dedication

I dedicate this thesis to my children Yahia and Omar, my spouse Nour, and my parents.

Contents

List of Tables **ix**

List of Figures **xi**

1 Introduction **1**
1.1 Motivations 1
1.2 Objectives 2
1.3 Challenges 3
1.4 Thesis Overview 4

2 Literature Review and Background **7**
2.1 Driver Assistance Systems 7
2.1.1 Adaptive Cruise Control 7
2.1.2 Lane Departure Warning and Lane Change Assistance Systems 8
2.1.3 Traffic Sign Recognition 8
2.1.4 Driver Fatigue Detection 9
2.2 Driver Fatigue Monitoring 9
2.2.1 Vision-Based Systems 10
2.2.2 Audio-Based Systems 11
2.3 Classification 12
2.3.1 Support Vector Machines 12
2.3.1.1 Dual Problem and Kernel Trick 14
2.3.1.2 Soft Margin Classifier 15
2.3.2 HMM: Theory and Implementation 16
2.3.2.1 Modeling Speech Using HMMs 16
2.3.2.2 Theory and Elements of HMM 17
2.3.2.3 HMM Topologies 18
2.3.2.4 The Basic Problems of HMM's 19
2.4 Data Fusion 20
2.4.1 Types of Data Fusion 20
2.4.2 Bayesian Networks 23
2.4.2.1 Example for Bayesian Networks 23

3 Proposed Approach **25**
3.1 System Architecture 25

3.1.1 Audio Module . . . 26
3.1.2 Heart Rate and Other Signals Module . . . 26
3.1.3 Decision Fusion Using Bayesian Networks . . . 27
3.2 Dataset Description and Collection . . . 27
3.2.1 Dataset Description . . . 28
3.2.2 Data Collection . . . 28
3.2.2.1 Experiment General Information . . . 30
3.2.2.2 Driving Session . . . 31
3.3 Driver Fatigue Detection Using Speech . . . 32
3.3.1 Voice Activity Detection . . . 34
3.3.2 Speech Analysis For Driver Fatigue Using SVM . . . 38
3.3.3 Speech Analysis For Driver Fatigue Using HMM . . . 39
3.4 Driver Fatigue Detection Using Heart Rate and Other Signals . . . 41
3.4.1 Using SVM for Driver Fatigue Detection . . . 42
3.4.2 Using HMM for Driver Fatigue Detection . . . 44
3.5 Fatigue Inference Using Bayesian Networks . . . 44
3.5.1 Pre-processing . . . 45
3.5.2 Bayesian Network Design . . . 46
3.5.3 Inference . . . 47

4 Experimental Results **49**
4.1 Voice Activity Detection . . . 49
4.1.1 Tuning Window Size For The Features . . . 50
4.1.2 Tuning The Penalty Constant For The SVM Classifier . . . 51
4.2 Audio Module . . . 52
4.2.1 SVM Classifier . . . 52
4.2.2 HMM Classifier . . . 53
4.3 Heart Rate and Other Signals Module . . . 55
4.3.1 SVM Classifier . . . 55
4.3.2 HMM Classifier . . . 56
4.4 Bayesian Network . . . 57

5 Conclusions and Future Work **60**
5.1 Conclusions . . . 60
5.1.1 Dataset . . . 60
5.1.2 Fatigue Monitoring System . . . 61
5.1.2.1 The Audio Module . . . 61
5.1.2.2 The Heart Rate and Other Signals Module . . . 62
5.1.2.3 Decision Fusion Using Bayesian Networks . . . 62
5.2 Future Work . . . 63
5.2.1 Dataset . . . 63
5.2.2 System . . . 64
5.3 Publications . . . 66

A Detailed Tasks Description **67**

A.1 First Phone Call 67
A.2 Second Phone Call 68
A.3 First Text Message 69
A.4 Second Text Message 69
A.5 Third Text Message 69
A.6 Map research 69

List of Tables

3.1 Sample pre-processed output from second 160 to second 190 for one of the drivers. 45

4.1 Feature extraction example. F(t), F(t-1), and F(t-2) represented the raw features of three consecutive frames. The example shows how to calculate Δ and $\Delta\Delta$ features. The final feature frame for time t is the concatination of the three colums F(t), $\Delta(t)$, and $\Delta\Delta(t)$. . . 55

List of Figures

2.1 Driver fatigue monitoring systems. 9
2.2 Two-class linearly separable classification problem. 12
2.3 Large-margin decision boundary. 13
2.4 Missclassified data points and their corresponding ξ distances drawn in black lines. 15
2.5 Noisy channel communication model where "W" is the intended message composed by the producer, and "O" is the message as observed by the receiver. 16
2.6 Ergodic Hidden Markov Model 18
2.7 Left to right Hidden Markov Model 19
2.8 Types of fusion systems. 21
2.9 Bayesian networks example. 24

3.1 System Architecture 26
3.2 Car Driving Simulator 29
3.3 The types of sensors installed on the car driving simulator 29
3.4 Timeline for the morning session. The gray areas indicate "normal driving", while black areas indicate "loading map and normal driving". 32
3.5 Timeline for the evening session. The gray areas indicate "normal driving", while black areas indicate "loading map and normal driving". 32
3.6 Example of different experiment tasks 33
3.7 Audio Module 34
3.8 Voice Activity Detection based on 20 milliseconds frames. 35
3.9 First two principle Components of the 20 milliseconds based features 36
3.10 Voice Activity Detection based on 1 second frames 37
3.11 First two principle Components of the 20 milliseconds based features 37
3.12 Speech analysis for driver fatigue detection. 38
3.13 First two principle Components of the features used to train the SVM classifier. 39
3.14 Speech analysis for driver fatigue detection using HMM system. . . 40
3.15 Analyzing heart rate and other signals for fatigue detection using SVM system. 41
3.16 Steering wheel, gas, brake, clutch positions and heart rate for a complete driving sessions for one of the drivers. 42
3.17 First two principle Components of the MFCC based features. 43

3.18 Analyzing heart rate and other signals for fatigue detection using HMM system. 44
3.19 Decision fusion of the voice and other signals modules using Bayesian Network 46

4.1 The accuracy of the VAD module plotted against the window size in seconds. 50
4.2 The accuracy of the VAD module for different window sizes plotted against the penalty constant (C) of the SVM classifier. 51
4.3 The accuracy of the audio module plotted against the penalty constant for the SVM classifier. 53
4.4 Total average accuracy when varying the number of HMM states for the audio module. 54
4.5 The accuracy of the heart rate and other signals module plotted against the window size. 56
4.6 Total average accuracy when varying the number of HMM states for the heart rate and other signals. 57
4.7 The Bayesian network used to fuse different decisions and the corresponding CPTs. 58
4.8 Figure shows the improvement in the total average accuracy for the final decision when taking the prior information into account. The window size is in seconds. 59

5.1 A proposed Bayesian network that fuses decisions from more modules. 65
5.2 Monitoring driver fatigue using driver adapted models. 66

Chapter 1

Introduction

1.1 Motivations

The automotive industry is growing bigger each year. According to the Organisation Internationale des Constructeurs d'Automobiles (OICA), 84,141,209 cars and commercial vehicles have been produced in 2012 alone [1]. The average increase in automotive production over the past decade has been 4.1%. Such a vast market drives the companies to regularly improve their products by spending more on research and development (R&D). According to a Booz & Co. report in 2013, Volkswagen automotive company spent more on R&D than any other company worldwide [2]. The R&D spending by Volkswagen alone was 11.4 billion dollars which is 4.6% of it total revenue.

Driver and passenger safety are among the primary concerns of any automotive company. According to an article published in 2009, up to 75% of all roadway accidents are caused by driver error [3]. Many automotive companies have developed driver assistance systems to help the driver in different driving tasks, and to ensure driver safety. These systems include adaptive cruise control, lane departure warning, lane change assistance, collision avoidance, night vision, automatic parking, traffic sign recognition, and driver drowsiness detection.

According to the U.S. National Highway Traffic Safety Administration in 2011, 2.5% of the vehicle drivers involved in fatal accidents were either drowsy, asleep, fatigued, ill, or blacked out [4]. Another research article in 1999 showed that 15-20% of the accidents are sleep-related [5]. These statistics and more have focused the automotive companies' attention to building driver drowsiness detection systems.

Most of the previous work used visual approaches to detect fatigue [6, 7, 8]. Percentage eye closure (PERCLOS), yawn frequency, head movement, gaze, and facial

expressions were the main features used to determine fatigue level. Little attention has been paid to monitoring fatigue using speech and other signals. In [9], a speech recognition system was used to detect the level of fatigue, finding that some phones experimentally show a predictable dependence on fatigue. In [10], different features are extracted and used to train eight different classifiers. An ensemble classifier is used to fuse the decisions from the eight classifiers. In [11], Mel-Frequency Cepstrum Coefficients (MFCC) features are used along with Gaussian Mixture Models to detect fatigue level.

Although the performance of the speech-based systems is acceptable, there are some problems that need to be tackled before they can be employed in driver fatigue detection. A Voice Activity Detector (VAD) must be used to extract the speech signal from the audio signal. Voice activity detection is still an open problem, and its performance depends on the nature of the problem. Also we need to think of a way to decide the level of the fatigue in the absence of a speech signal. Finally, using other sensors, and using the decision from different sub-systems, would allow for improvement. In this thesis, we aim at developing a driver fatigue detection system using speech, heart rate, steering wheel, and pedal positions. Decision fusion can be used to fuse decisions from the different modules, and to improve the performance of the overall system.

1.2 Objectives

Below are the main objectives of this thesis:

1. Building a practical and realistic dataset that can model both driver fatigue and inattention. The drivers involved in the experiments need to be selected such that they represent the real spectrum of drivers. We strove to design an experiment such that the drivers would exhibit realistic fatigue and inattention behaviours, while ensuring the safety of the drivers during the experiment. Many sensors, including Kinect, infrared camera, microphone, heart rate monitor, and steering wheel and pedal positions will be used to capture the driver behaviour from different perspectives. This data needs to be carefully labeled and stored in a standard format. Finally, we aim to make the dataset publicly available for other research groups.

2. Building an audio module that can analyze the driver'sa voice for fatigue. To build such a module, we need also to build a voice activity detection (VAD) module, to extract the driver's voice from the audio signal. Voice Activity Dection is still an open problem, and the performance of the VAD systems varies according to the nature of the problem. In our dataset, the audio is recorded in a quiet room, with the steering wheel and pedals as the only

source of noise. We need to take advantage of such a setup to customize a VAD module with high performance.
On the other hand, we aim to build a fatigue classification module that takes its input from the VAD module and analyzes the voice signals for the presence of the fatigue. Our goal is to achieve an accuracy above 80%. In order to reach this goal, different classifiers will be used and tuned for maximum accuracy.

3. Building a module that can analyze the heart rate, steering wheel, and pedal positions for driver fatigue. The signals from different sensors are from different values, ranges, and sampling rates. We need to pre-process the data and extract useful features. The goal is to try different classifiers to obtain high accuracy.

4. The goal of using different sensors is to be able to improve the performance of the whole system. Our goal is to fuse the decision from the audio module and the heart rate and other signals module such that the performance of the whole system improves.

5. We aim to build a system that can be extended and integrated with other systems. Fusing different modules on the decision level allows integration with the other systems while dealing with each system as a black box.

1.3 Challenges

Most of the datasets used by other research groups are not publicly available. The only dataset that we successfully obtained for use is the dataset built by the Robesafe research group [12, 6]. This dataset is very limited. Only three human drivers were involved in the experiments, and the only sensor used to record the driver behaviour was an infrared camera. According to our objectives, such a dataset is not a good fit to our project. Hence, we decided to build our own dataset.

Building our own dataset is the first challenge for this project. This dataset should meet our objectives, and not be a duplicate of other datasets. Different driver behaviours indicating fatigue or inattention need to be captured during the data collection. Various sensors need to be used to capture driver behaviour. It is essential to carefully set up the data collection experiments such that the safety of the drivers is ensured, and the number of drivers, the duration of the experiment, and the actions performed by the drivers are defined.

Noise is another challenge we are going to face. For the sound recording, the types of noise includes microphone noise, steering wheel noise, pedal noise, and the noise

of driver body movements while performing driving tasks. For the pedal position noise, the source of noise could be the position of the clutch when the driving simulator is set to be automatic. Such noise needs to be dealt with either in the pre-processing, feature extraction, or classification stage.

Feature extraction is another important challenge, one which can be an art as much as a scientific task. Trial and error is often useful in determining the best set of features to be used. The performance of the system strongly depends on the type of features used to represent the signals. These features need to be representative, robust, and reliable. Signal noise can often be filtered using the right features. On the other hand, the features need to be organized to suit the classifier. Some classifiers like SVM need fixed length feature vectors, while others like HMM need frame-based features.

The classifier is the heart of the system. When the classifier is chosen, all other modules need to be designed to suit the classifier, which means choosing the right classifier is critical. Trial and error is not the only way to determine which classifier is best for the system. Being aware of the nature of the problem and the data, and whether it is sequential or not, helps to choose the right classifier. Visualizing the data helps in such a decision. It gives information about the data and whether it is separable or not, linearly separable or not, whether the classes overlap or not, etc. Such information helps the designer to choose an appropriate classifier.

Fusion is another challenge for this project. The fusion can be done on the features level or the decisions level, each of which has its advantages and disadvantages. Fusion on the features level can produce more accuracy, but the system becomes very complex. Fusing the decisions opens a room for integrating more decisions without modifying the system submodules. We have to choose the right method of fusion to improve accuracy, and to open the doors for this project to be extended in the future.

1.4 Thesis Overview

In chapter 2, the driver assistance systems are reviewed. The previous work is presented and criticized, with focus on the systems related to driver fatigue detection. The drawbacks of each system are listed. The main concerns about each systems are 1) whether the dataset is practical or not, 2) how does the system behave in different conditions, and when does it fail to perform its function?. After revisiting the previous work, we propose a methodology to build a car driver fatigue monitoring system, while trying to avoid the drawbacks of other systems.

Data collection is an important part of our system. We briefly describe the dataset we want to build. We also present various sensors we are going to use to collect the data, and how we are going to collect the data. There are classifiers that are used throughout the thesis: SVM and HMM classifiers. Each classifier is visited, and the ideas and mathematical models are presented.

In Chapter 3, the framework for our system is described in full detail. The chapter starts with the proposed approach and the system architecture, followed by the data collection. The third section presents the audio module including voice activity detection (VAD). The fourth section presents the heart rate and other signals module. Finally, decision fusion using Bayesian networks is described in the last section.

In the experimental results chapter (Chapter 4), the framework setup for each experiment is presented. This includes programming language, toolkits, model parameters settings, train and test datasets, and performance evaluation. For the VAD module, feature extraction using MFCC features is described, and SVM model training and testing are also presented. The performance of the classifier is reported using different graphs.

For the audio module, data acquisition and feature extraction are described, and SVM model training and tuning are presented. The performance of the SVM classifier is reported to be low and unsatisfying. The reasons for such performance are described, and as a result, an HMM classifier is used. HMM model training and testing are presented. The training and testing are done using HTK toolkit. The tuning of the HMM model for maximum accuracy is described and shown using some plots.

Similar to the audio module, the framework for the heart rate and other signals module is described. To adjust for different sampling rates for the signals, signals are resampled. Features are extracted for both SVM and HMM classifiers, and The tuning of the SVM classifier is described. The best performance for the SVM-based module is reported to be very low. Such performance is analyzed and the HMM classifier is used to improve such low performance. The performance of the HMM-based module is shown in plots. The best accuracy for the HMM-based module is reported.

The framework for the decision fusing using Bayesian network is described. This framework is set up using MATLAB, and the inference is done using variable elimination algorithm. Prior information is incorporated into the network to improve the final decision. The total accuracy of the system is reported both with and without the prior information. Finally, the best total accuracy of the system is reported.

In the last chapter, conclusions and future work are presented. The contributions of this work are presented for each module. In the second section, proposed ways to extend this work are presented, including improving the framework on the dataset and methodology levels. Improving the dataset could be accomplished by involving more human subjects, more fatigue levels, by using electroencephalography signals for ground truth fatigue labeling, and using real vehicle along with the driving simulator. The ways for improving the system includes using more features and classifiers, fusing more decisions, and driver adaptation. The publications based on this work are also listed.

Chapter 2

Literature Review and Background

2.1 Driver Assistance Systems

Driver and passenger safety are of prime concern to automotive companies. According to an article published in 2009, up to 75% of all roadway accidents are caused by driver error [3]. Many automotive companies have developed driver assistance systems to help the driver and to ensure driver safety. These systems include adaptive cruise control, lane departure warning, lane change assistance, night vision, automatic parking, traffic sign recognition, and driver drowsiness detection. In the following subsections, some of assistance systems are reviewed.

2.1.1 Adaptive Cruise Control

Adaptive cruise control (ACC) [13] tracks the speeds of the vehicle ahead using a sensor (e.g. a radar) installed on the front of the vehicle. The system uses this information to adjust speed so as to maintain a safe distance. Mitsubishi was the first automotive company to bring an ACC system to the market in 1995. Now, many automotive companies have models that support adaptive cruise control.

Hoedemaeker [14] has conducted a research on the usability of the ACC systems. It was found that the ACC system is more useful for motorways than for rural roads; low-speed drivers appreciate the ACC system more than high-speed drivers. The author states that the car driver needs to know the ACC system behaviour and limitations in order to engage and disengage the system appropriately.

2.1.2 Lane Departure Warning and Lane Change Assistance Systems

The lane departure warning system [15] is used to warn the vehicle whenever the driver unintentionally crosses the lane lines. The system attempts to alert the driver either by vibration, audio, or displaying a message on the dashboard. Sensors, including cameras and lasers, are normally installed behind the windshield of the vehicle. Information obtained from the sensors are used to decide whether the vehicle crossed the line or not. Nissan was the first company to introduce this system to the market in 2001; now it is common to find this system in luxury cars.

Many researchers have been tackling this problem for a long time. In 1995, Chen et. al. [16] developed the AURORA system, which is a vision-based roadway departure warning system. The system uses a downward-looking color video camera with a wide angle lens, a digitizer, and portable Sun workstation. The system uses template correlation to accurately detect lane markers. At each video frame, an algorithm is run to detect the car position. Based on car position and velocity, an alarm is triggered in case of lane departure. The system has been tested under different weather, road, and lighting conditions, and the results were excellent.

Another group, at University of California at Berkeley, has developed Video-based Lane Estimation and Tracking" (VioLET) system [17]. This system uses steerable filters [18] for lane marker tracking. Steerable filters offer great robustness to different shadow and lighting conditions, as well as computational simplicity which makes the system implementation very fast. The system has been tested on different road and weather conditions and the performance was robust and excellent.

Lane change assistance systems [15] help the driver to change lanes safely. The system uses cameras installed at the rear bumper to scan the blind spots for the driver. The system alerts the driver when attempting to change lane while another vehicle is approaching in the blind spots. Many automotive companies provide this system in some of their cars, including Audi, Volkswagen, BMW, Porsche, and Mazda.

2.1.3 Traffic Sign Recognition

Traffic sign recognition is an intelligent system that can read traffic signs such as speed limit, turn ahead, and others. Audi, BMW, Mercedes-Benz, Volvo, Volkswagen, and Opel all use this system in certain car models. In 2007, Escalera et. al. [19] developed a traffic sign detection and recognition system. The traffic sign

detection is done using the color and corners of the traffic signs, guided by a neural network. Different types of signs are recognized, such as speed limit, railway crossing, left turn ahead, no left turn among other signs.

Marcin L. Eichner and Toby P. Breckon have developed a vision system to recognize speed limit signs, cancellation signs, and even turn signs from a moving vehicle [20]. The system is customized for UK signs, but the authors state that it can be extended to work for other countaries. The sign detection is done using a RANdom SAmple Consensus (RANSAC) algorithm [21], while neural networks are used for signs recognition. The system has been successfully tested under different daylight and weather conditions.

2.1.4 Driver Fatigue Detection

Driver fatigue detection is an intelligent system that can monitor driver fatigue. Various sensors, such as a camera, microphone, and heart rate monitor, can be used to assess the driver fatigue. The system advises the driver to take some rest once drowsiness is detected. In the next section, previous work in driver fatigue detection systems is reviewed.

2.2 Driver Fatigue Monitoring

The increasing number of traffic accidents caused by driver fatigue has brought the attention of researchers and automotive companies to developing driver fatigue monitoring systems. The U.S. National Highway Traffic Safety Administration has listed driver fatigue as one of the primary causes of traffic accidents [4]. According to their study in 2011, 2.5% of vehicle drivers involved in fatal accidents were either drowsy, asleep, fatigued, ill, or blacked out [4]. Other reports confirm that 10-20% of accidents are fatigue-related [22]. Another research article in 1999 found that 15-20% of accidents are sleep related [5]. Such statistics , among others, showed the need for developing the fatigue monitoring systems.

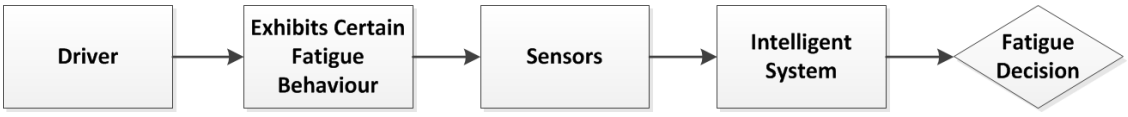


Figure 2.1: Driver fatigue monitoring systems.

The general fatigue detection system is shown in Fig. 2.1. The car driver exhibits certain fatigue behaviour/s, detected by sensors which could be intrusive or nonintrusive. The system uses sensors to record driver behaviour. An intelligent system

is used to analyze the data obtained from the sensors, and pattern recognition and machine intelligence techniques are used for the fatigue analysis. Finally, the fatigue decision is reached and an action is taken, if the driver is fatigued, to alert the driver.

There are different techniques to detect driver fatigue. The techniques based on physiological phenomena like brain waves, heart rate, pulse rate, and respiration are the most accurate [23]. These techniques are intrusive, needing electrodes to be attached to the driver's body. Such techniques are not practical for everyday driving use.

Nonintrusive techniques are more practical: nothing needs to be attached to the driver's body. Most of the previous papers in detecting fatigue level have used vision-based systems to detect the level of fatigue. Some attempts have been made to detect fatigue through the driver's voice. Heart rate and pedal positions have also been used. In the following sub-sections, the main papers addressing the driver fatigue detection are presented.

2.2.1 Vision-Based Systems

Vision-based fatigue detection systems use vision-based sensors (cameras) to record apparent driver behaviour. Different types of cameras can be used, whether that be an ordinary camera, an infrared camera, or even a Kinect. Visual features are extracted and vision-based techniques are used to estimate driver fatigue level.

Smith et al. [24] in 2003 developed an intelligent system to detect driver fatigue. A normal color camera is used for collecting the data. Three different visual features were extracted: eye blinking, eye rotation, and gaze tracking. A finite state machine is used to estimate the fatigue level. The main drawback of this system is using a color camera, making the system unsuitable for night driving the time fatigue is most likely to be a problem.

Bergasa et al. [6] in 2006 developed a driver fatigue monitoring system, in which an infrared camera was used to record driver behaviour. Various visual features were used including eye percentage closure (PERCLOS), eye closure duration, blink frequency, nodding, face position, and fixed gaze. Fuzzy logic is used in order to fuse these features and produce the fatigue decision. The main drawback of the system is the limited dataset used in the training and testing. Only three drivers were recorded in the dataset. The fatigue was also completely simulated and not genuine. Also, the performance of the system degrades when the driver wears sunglasses.

Ji et al. [25, 26] developed a probabilistic framework to monitor human fatigue. Two infrared cameras were used to record the dataset. Different visual features were extracted, including yawning frequency, PERCLOS, gaze distribution, and head tilt frequency. A Bayesian network framework was used to estimate the final fatigue decision.

D'Orazio et al. [8] in 2007 developed a fatigue detection system. A normal color camera was used for collecting the dataset. Two visual features were extracted: eye blinking, and PERCLOS. A Gaussian mixture model was used to classify the features and estimate driver fatigue. The main drawback of the system was their use of a color camera, which would fail in night driving conditions.

2.2.2 Audio-Based Systems

Audio-based fatigue detection systems use a microphone to record the driver's voice, and the speech signal is analyzed using different techniques to estimate the level of fatigue.

In [9], a speech recognition system was used to detect the level of fatigue, finding that some phones experimentally showed a predictable dependence on fatigue. The system showed that the sound 'p' and 't' are highly correlated to the Sleep Onset Latency (SOL), which is a standard measure for sleepiness level. The paper didn't mention the performance of the speech recognition system they used in their system, and how it would affect the performance of their system.

In [10], a system for fatigue detection using speech signals was presented. Different features were extracted from the speech signal, including fundamental frequency, fundamental frequency peak process, intensity, Harmonics-to-Noise Ratio (HNR), formant position and bandwidth , Linear Predictive Coding (LPC), 12 MFCCs, 12 Linear Frequency Cepstral Coefficients (LFCC), duration of voiced and unvoiced speech segments, and long term average spectrum (LTAS). Eight different classifiers were trained to estimate the level of fatigue. An ensemble classifier was used to fuse the decisions from the eight classifiers. The recognition rate for this system was 83.8%. The main drawback of the system is the dataset, only two participants were involved in the data recording.

In [11], Mel-Frequency Cepstrum Coefficients (MFCC) features were used along with Gaussian Mixture Models (GMM) to detect the fatigue level. Participants were asked to repeat certain sentences at different times during the day. Electroencephalography (EEG)-based measurements were used as the ground truth. GMMs were used to extract certain features: voiced, unvoiced, and silent parts of

the speech signals. The features were then correlated to EEG measurements. The paper didn't address how the speech signal can be extracted.

Although the performance of the speech-based systems is acceptable, there are some problems that need to be tackled before they can be employed in driver fatigue detection. A Voice Activity Detector (VAD) needs to be used to extract the speech signal from the audio signal; Voice activity detection is still an open problem, and its performance depends on the nature of the problem. Also, a way needs to be found to decide the level of the fatigue in the absence of a speech signal. Finally, using other sensors and using decisions from different modules could lead to significant improvement.

2.3 Classification

2.3.1 Support Vector Machines

Supervised classification is a central task in machine learning. The problem is to observe a number of labeled data points, and then learn a classifier that can accurately assign labels to new data points. The main challenge for each classifier is to be able to generalize to unseen conditions. Some classifiers tend to generalize well, while others don't. Support Vector Machines (SVMs) are well-known for their ability to generalize, and they are very robust against overfitting[27]. SVMs have been used extensively over the past decade because of their performance and robustness. In this section, the basic theoretical fundamentals of Support Vector Machines are introduced.

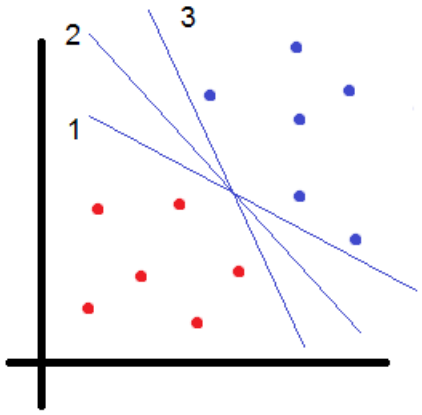


Figure 2.2: Two-class linearly separable classification problem.

Consider a linearly separable two-class classification problem as shown in Fig. 2.2. There are an infinite number of decision boundaries that can separate the two classes, the best decision boundary is that which minimizes the generalization error. The SVM classifier tries to minimizes the generalization error using a large-margin decision boundary. This large-margin decision boundary is a linear separator that is as far as possible from both class boundaries.

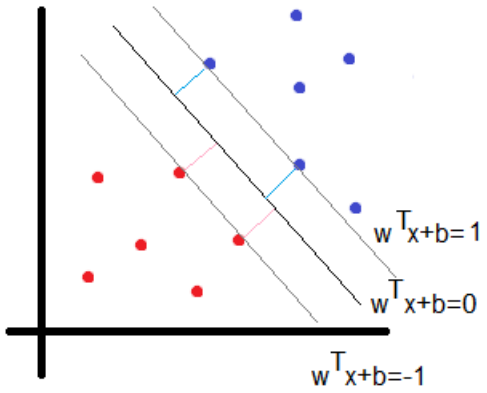


Figure 2.3: Large-margin decision boundary.

Let our data points be $\{x_1, x_2, ..., x_n\}$ and let the class label for data point x_i be $y_i \in \{-1, 1\}$. As shown in Fig. 2.3, the linear separator equation is $w^T x + b = 0$. The distance from any point to the linear separator is:

$$\frac{y(w^T x + b)}{||w||} \; where \; y \in \{-1, 1\} \tag{2.1}$$

The optimization function for the maximum margin classifier can be written as follows:

$$max_{w,b} \frac{1}{||w||} \{min_i y_i (w^T x_i + b)\} \; \forall i \tag{2.2}$$

Such a minimization function can be simplified by fixing the minimum distance to 1 and minimizing $||w||$ such that:

$$\begin{array}{c} min_{w,b} \frac{1}{2} ||w||^2 \\ s.t. \; y_i (w^T x_i + b) \geq 1 \;\; \forall i \end{array} \tag{2.3}$$

Such an optimization problem is convex and quadratic [28]. It can be solved using various optimization tools [29]. Only the points located at the boundaries $y_i(w^T x_i + b) = 1$ are necessary; these points are called support vectors.

2.3.1.1 Dual Problem and Kernel Trick

The kernel trick is one of the important properties of the SVM classifiers. The feature space can be mapped to a higher (and even an infinite) dimension feature space without adding much computations. The idea is to reform the optimization function such that all $x's$ can be replaced by a kernel function. Using a Lagrangian multiplier [30], the optimization problem in Eq. 2.3 can be reformulated as follows:

$$\begin{gathered} max_a min_{w,b} L(w,b,a) \; s.t. \; a \geq 0 \\ where \; L(w,b,a) = \frac{1}{2}||w||^2 - \sum_i a_i[y_i(w^T x_i + b) - 1] \end{gathered} \tag{2.4}$$

The inner minimization is $min_{w,b} L(w,b,a)$:

$$min_{w,b} \; \frac{1}{2}||w||^2 - \sum_i a_i[y_i(w^T x_i + b) - 1] \tag{2.5}$$

Solving the above minimization by setting the derivatives to 0, we get:

$$\begin{gathered} w = \sum_i a_i y_i x_i \\ 0 = \sum_i a_i y_i \end{gathered} \tag{2.6}$$

Substituting back in Eq. 2.4 using the above equations and replacing each $x_i^T x_j$ by $k(x_i, x_j)$ we get:

$$L(a) = \sum_i a_i - \frac{1}{2} \sum_n \sum_m a_n a_m y_n y_m k(x_n, x_m) \tag{2.7}$$

Using Eq. 2.4 and Eq. 2.7, we get the dual problem in which many $a_i's$ are 0. The data points at which $a_i \neq 0$ are called the support vectors.

$$\begin{gathered} max_a \, L(a) \\ s.t. \; \sum_i a_i y_i = 0 \\ a_i \geq 0 \end{gathered} \tag{2.8}$$

The classifier decision becomes:

$$y = sign\left(\sum_i a_i y_i k(x_i, x) + b\right) \tag{2.9}$$

2.3.1.2 Soft Margin Classifier

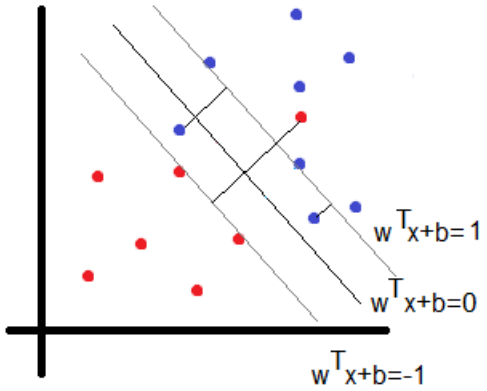


Figure 2.4: Missclassified data points and their corresponding ξ distances drawn in black lines.

In the previous derivations for the SVM classifier, the classes are considered linearly separable. To allow for misclassified samples, a slack variable ξ is defined as shown in Fig. 2.4. This slack variable reflects the order of the misclassified samples. The optimization problem is formulated as follows:

$$\begin{gathered} min_{w,b,\xi}\, C\sum_i \xi_i + \tfrac{1}{2}||w||^2 \\ s.t.\, y_i(w^T x_i + b) \geq 1 - \xi_i \;\; \forall i \\ \xi_i \geq 0 \;\forall i \end{gathered} \tag{2.10}$$

In the above formula, constant C is the penalty constant. This constant controls the tradeoff between the large margin and the slack variable penalty. Setting this variable to a very large value, the problem becomes similar to the hard margin classifier. The dual problem is formulated using Lagrangian as follows:

$$\begin{gathered} max_{a,\mu} min_{w,b,\xi} L(w,b,a,\mu,\xi) \;\; s.t.\;\; a,\mu \geq 0 \\ where\; L(w,b,a,\mu,\xi) = C\sum_i \xi_i + \tfrac{1}{2}||w||^2 - \sum_i a_i[y_i(w^T x_i + b) - 1 + \xi_i] - \sum_i \mu_i \xi_i \end{gathered} \tag{2.11}$$

Solving the above optimization problem similarly to how in Eq. 2.4 was solved, the dual problem becomes:

$$L(a) = \sum_i a_i - \frac{1}{2} \sum_n \sum_m a_n a_m y_n y_m k(x_n, x_m) \tag{2.12}$$

$$\begin{gathered} max_a \, L(a) \\ s.t. \, \textstyle\sum_i a_i y_i = 0 \\ C \geq a_i \geq 0 \end{gathered} \tag{2.13}$$

The support vectors are the samples at which $a_i > 0$. The classifier decision for the soft margin is still the same as shown in Eq. 2.9.

2.3.2 HMM: Theory and Implementation

2.3.2.1 Modeling Speech Using HMMs

The speech process starts in the mind of some human when a message is composed to be delivered to the intended listener(s). The ultimate goal of this process is to recognize the message as intended by analyzing the message as observed. This is illustrated by the famous noisy channel communication model shown in Fig.2.5.

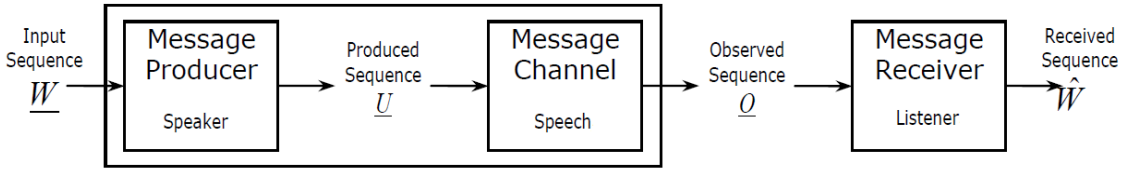


Figure 2.5: Noisy channel communication model where "W" is the intended message composed by the producer, and "O" is the message as observed by the receiver.

The distortion caused by the noisy channel typically leaves us with the ambiguous problem of having multiple possible perceived sequences, from which we have to recognize the intended message sequence. While building a fully rule-based recognizer is almost impossible due to the randomized nature of the channel distortion, the goal of a recognizer finding the intended sequence can practically be achieved stochastically.

The maximum *a posteriori* probability (MAP) approach is one of the most widely-used and mathematically well-founded methodologies in this regard. According to this methodology, the elected message sequence $\hat{W}$ is selected to maximize the *a*

posteriori probability $P(W|O)$ over all the possible (producible) messages, which is formally expressed as:

$$\hat{W} = \underset{\forall W}{argmax}\ \{P(W|O)\} = \underset{\forall W}{argmax}\ \left\{\frac{P(O|W)P(W)}{P(O)}\right\} = \underset{\forall W}{argmax}\ \{P(O|W)P(W)\} \tag{2.14}$$

Where $P(O|W)$ is called the likelihood probability which models the forward conditional stochastic relation between intended/input classes/labels and their consequent observations, and $P(W)$ is called the language model that gives the *a priori* marginal probability of any possible sequence of classes. The *a priori* probability of the observations $P(O)$ can obviously be omitted from the maximization formula, as it is independent of W.

2.3.2.2 Theory and Elements of HMM

HMM is a stochastic process with an underlying finite-state structure; each one of these states is associated with a random function. Within a state, the signal possesses some measurable, distinctive properties. Within a discrete period of time, the process is assumed to be in some state, and an observation is generated by the random function of that state. The underlying Markov chain changes to another state based on the transition probability of the current state. The sequence of states is hidden, only the sequence of observations produced by the random function of each state can be seen.

Consider a recognizer system that at any instant in time may only be in one of the state belonging to its state set. The system undergoes a change from one state to another according to a set of probabilities associated with each state. These transitions take place in regularly-spaced discrete periods of time. In other words, a system transits from state S_i at time t to a state S_j at time $t+1$, $t = 1, 2, 3, \cdots N$ (no. of observations) and $i, j = 1, 2, \cdots, L$ (no. of states/word).

Another important element to be decided is the type of probability density function (PDF) for the observations per state. In the case of a discrete PDF, we have to decide the number of observation symbols per state. In the case of a continuous PDF, we have to decide the number of Gaussians per state. The observation symbols correspond to the physical output of the system being modeled. Individual symbols are denoted as $\hat{z}_1, \hat{z}_2, \cdots \hat{z}_K$.

The rest of the elements which characterize the discrete observation HMM are:

1. The initial state probability. This is the probability of being in state S_i at $t = 1$.

$$\pi = \{\pi_i = P(S_i \; at \; t = 1)\} \tag{2.15}$$

2. The state transition probability. This is the probability of being in state S_i at time t, then transiting to state S_j at time $t + 1$.

$$A = \{a_{ij} = P(S_j \, at \, t + 1 | S_i \, at \, t)\} \tag{2.16}$$

3. The observation symbol probability. This is the probability of observing symbol $\hat{z_k}$ while the model is in state i at time t.

$$B = \{b_{i,k} = P(\hat{z_k} | S_i)\} \tag{2.17}$$

Considering the previous elements mentioned, a complete specification of an HMM requires specifying two model parameters, L_g and K, the observation symbols, and three sets of probability measures: A, B, and π. The compact notation to be used to refer to a class HMM is $\lambda_g(A, B, \pi)$. This notation is reduced to $\lambda_g(A, B)$ in the case of using the left to right (L-R) HMM topology.

2.3.2.3 HMM Topologies

By placing certain restrictions on the structure of the model, the model behaves differently. A fully connected model, as shown in Fig. 2.6, is known as an ergodic model.

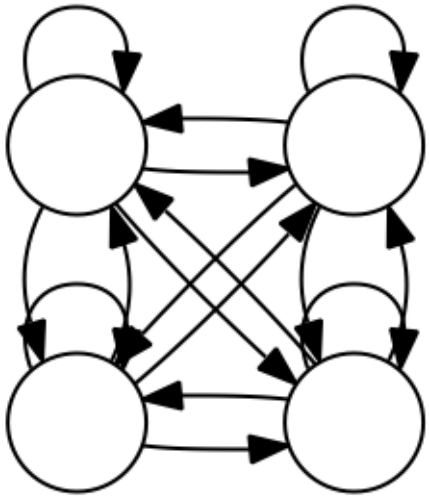


Figure 2.6: Ergodic Hidden Markov Model

For certain types of signals, other types of models have been shown to model that signal's properties better [31]. In the case of a time-variant signal, the L-R model is such a model. The state sequence of this type of model is forced to start in the leftmost state, and can only make transitions to the states to its right, or remain in the current state, as time passes. The model must also end in the rightmost state. This model has become the most popular form of HMM used in speech recognition and character recognition. An example for L-R model is shown in Fig.2.7.

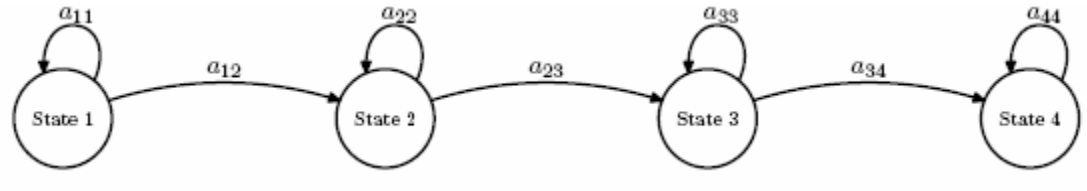


Figure 2.7: Left to right Hidden Markov Model

In the L-R HMM's,

$$\pi_i = \begin{cases} 1 & i = 1 \\ 0 & i \neq 1 \end{cases} \tag{2.18}$$

2.3.2.4 The Basic Problems of HMM's

There are three basic problems that must be solved for the model to be useful in our application. These are:

1. The evaluation problem:
   Given a sequence λ_W, concatenation of class models λ_g, and a sequence of observations $O = \{o_1, o_2, \cdots o_N\}$, what is the probability that the model generates the observations $P(O|\lambda_W)$?

2. The training problem:
   Given a sequence λ_W and a sequence of observations $O = \{o_1, o_2, \cdots o_N\}$, what should the model parameters be to maximize the probability of generating the observations?

3. The decoding problem:
   Given a set of models λ_g and a sequence of observations $O = \{o_1, o_2, \cdots o_N\}$, what is the most likely state sequence in these models that produces the observations?

The evaluation problem can be viewed as a way of scoring how well a given models matches a given observation sequence. So it is very useful in the case in which we are trying to choose among several competing models.

The decoding problem is one in which we attempt to uncover the hidden part of the model, i.e., to find the best matching state sequence given an observation sequence. In trying to find the best state sequence which is not the correct state sequence, we need an optimality criterion to solve it. The choice of criterion is a strong function of the intended use for the uncovered state sequence. The Viterbi decoder is used for decoding.

The training problem is the most important of the three problems. If we could solve this problem, we would have the means to automatically learn the parameters given an observation sequence. The Baum-Welch algorithm is used for training the HMM model [32].

2.4 Data Fusion

Any given standalone sensor may not always be able to gather all the required information about the physical environment. Using multi-sensors is very important in this regard. The data gathered by multiple sensors, if dealt with correctly, can improve the performance of the system significantly. Data fusion is needed to deal with multisensor data. Different sensors suffer from different types of noise. Such noise can not always be dealt with using filtering and denoising techniques, and thus data fusion techniques need to deal with such noise.

Hall and Llinas [33] defined data fusion: "data fusion techniques combine data from multiple sensors and related information from associated databases, to achieve improved accuracy and more specific inferences than could be achieved by the use of a single sensor alone". A more abstract definition was introduced by Federico Castanedo [34]: "we can define data fusion as a combination of multiple sources to obtain improved information". So data fusion deals with different data sources (sensors) and combines them on different levels to achieve better data quality and better system accuracy.

2.4.1 Types of Data Fusion

Data fusion can be classified into three types: data fusion, feature fusion, and decision fusion [35]. Fusion systems can be classified as shown in Fig. 2.8.

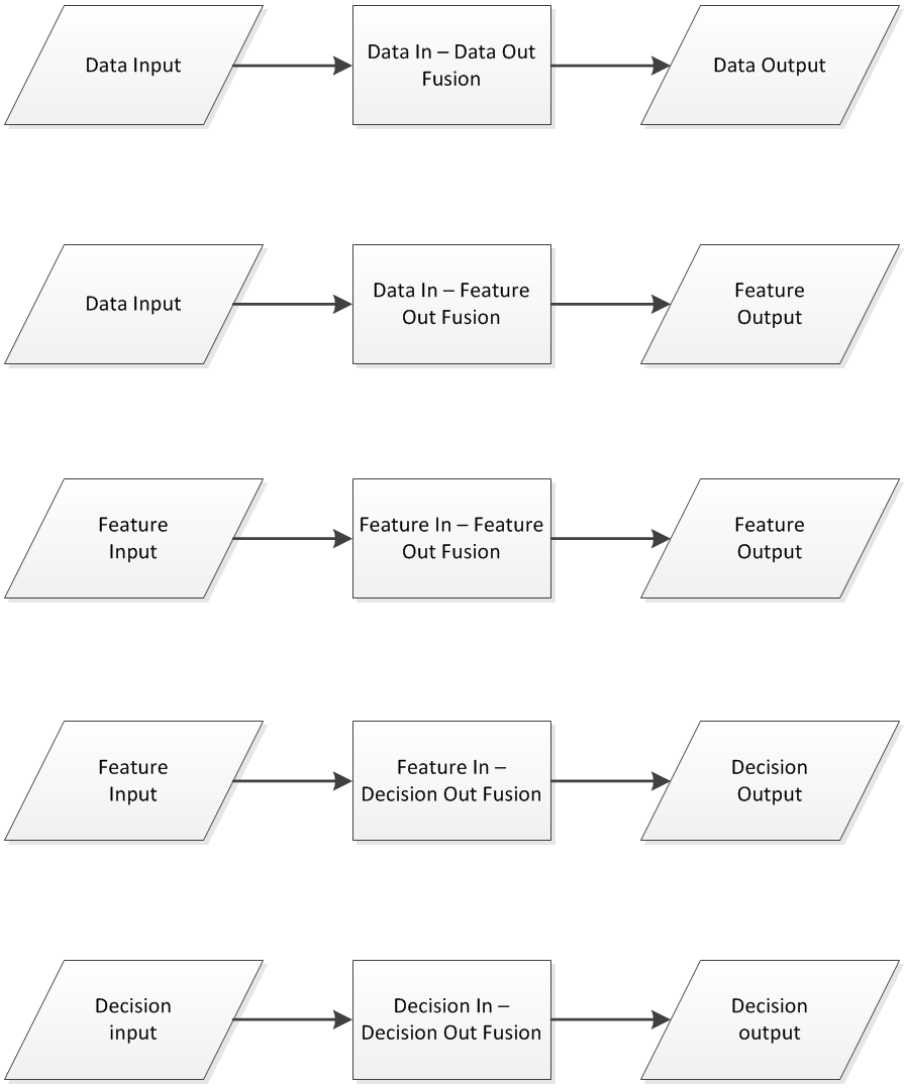


Figure 2.8: Types of fusion systems.

Data-In Data-Out Fusion This type of fusion is the lowest form of fusion. Both the input and the output of this type is data. An example of this fusion is image filtering when a filter is applied to an image for denoising, binarization, or any other reason.

Data-In Feature-Out Fusion Here, the data from different sensors are combined to extract some features. This type of fusion is well known by feature extraction. Combining data from different sensors should be done with caution. Data values, ranges, mean, variance, and missing values should be taken into consideration when extracting features from different data sources.

Feature-In Feature-Out Fusion In this type of fusion, both the input and the output are features, and thus it is known as feature fusion. Sometimes it is difficult when dealing with multi-sensor data to extract the features from the raw data directly. Features from each sensor are extracted first, and these features are then combined to extract the final features. Another example of this type is the data reduction using feature selection or feature extraction [36].

Feature-In Decision-Out Fusion Here, the inputs are the features from different sensors, and the output is the decision. It is also known as the classification step. Neural networks, SVM, Hidden Markov Models, and many other classifiers can be used for this type of fusion. In [10], eight different classifiers including SVM, Decision Trees [37], and neural networks were trained using various kinds of features.

Decision-In Decision-Out Fusion In this type of fusion, both the input and the outputs are decisions. This is the most common type of data fusion, and the most studied one [35], it is often named decision fusion. In this paradigm, different sub-systems are designed such that each system deals with subsets of the sensors. Each sub-system outputs a decision, and these decisions are then fused using decision fusion to produce the final decision. Such technques sometimes offer great flexibility in designing the whole system; each sub-system can be designed independently. The integration of the sub-systems is quite easy because each subsystem is looked at as a black-box. Adding sub-systems or taking out sub-systems can be done without much work.

There are different techniques for decision fusion. Among the most famous techniques are ensemble methods. Ensemble methods were proposed as a way for combining multiple weak classifiers (rules of thumb) to obtain a single strong (highly

accurate) classifier. Ensemble methods[38] can be categorized into two classes depending on their architecture: 1) Parallel Models and 2) Serial Models. In parallel models, each classifier is trained independently, and the votes generated by all the classifiers are fused together to obtain the final decision. In the serial models, classifiers are trained iteratively. The performance (e.g. the training error) of each classifier impacts the training phase of its subsequent classifiers.

Another important way of performing decision fusion is using expert systems. Expert systems are computer programs that can take decisions like human experts [39]. Expert systems can solve complex problems by knowledge-based reasoning. The expert systems implementation consists of two modules: the knowledge base, and the inference engine. The knowledge base is of a form IF .... THEN ..... . The IF part is called the antecedent, while the THEN part is called the consequent. An example of a knowledge base is "IF the car driver is yawning and it is 1:00 pm THEN he is fatigued". The inference engine is used to produce a reasoning on the knowledge base rule. For the previous example, it is not obvious whether the driver is fatigued or not. Yawning is an indication that the driver is fatigued. But, when it happens in the afternoon, it is not quite obvious to infer that the driver is fatigued. Different types of logic can be used to implement the inference engine, such as propositional logic, predicate logic, fuzzy logic [40], and probabilistic logic (Bayesian networks). In the next section, Bayesian networks are presented, as we are going to use a Bayesian network in our system.

2.4.2 Bayesian Networks

Bayesian networks are probabilistic models [41] that represent random variables and their conditional dependencies using a directed acyclic graph (DAG). Each node in the graph represents a random variable that can be either be discrete or continuous. An arc represents the conditional dependency between the parent node and the child node. Nodes that are not connected together are not directly dependent. The graph is parametrized by the conditional probability tables (CPTs). These tables specify the relation between each node and its parents.

2.4.2.1 Example for Bayesian Networks

Fig. 2.9 shows an example for using a Bayesian network to fuse decisions from two modules: module 1, and module 2. The example is a Bayesian network designed for driver fatigue detection, and the final decision is the driver fatigue. Modules 1 and 2 are conditionally dependent on the driver fatigue. The observed variables are called the evidence, while the variables needing to be evaluated are called the

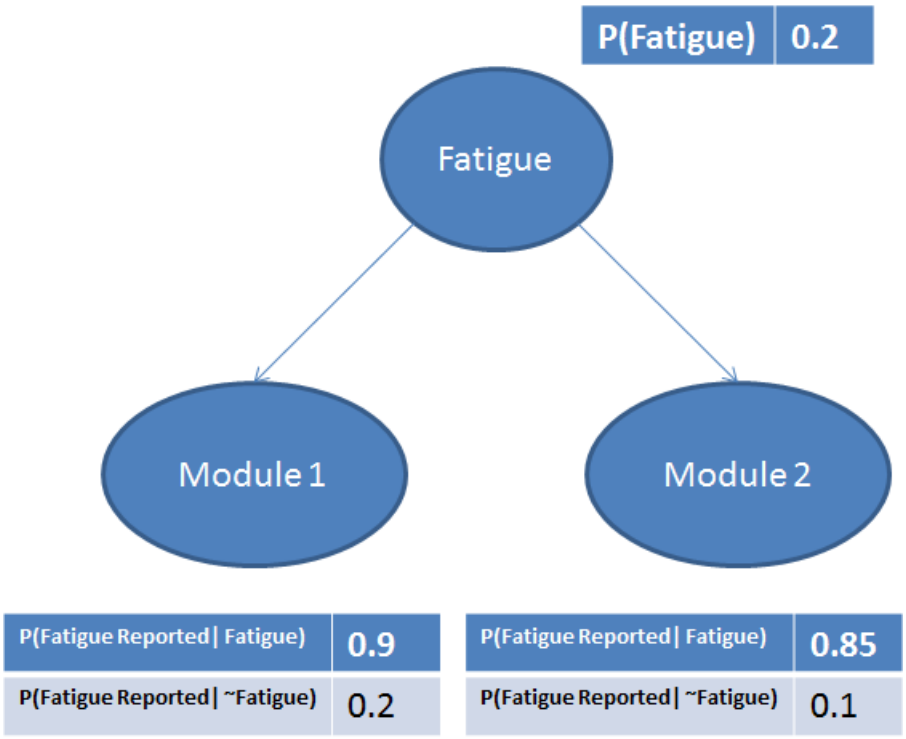


Figure 2.9: Bayesian networks example.

query variables. In this example, Module 1 and 2 are the evidence, and the fatigue variable is the query variable.

Suppose that module 1 reports that the driver is not fatigued, and module 2 reports that the driver is fatigued. We need to fuse both decisions to estimate the final fatigue decision. Using the conditional dependency, the probability of the fatigue $p1 = p(Fatigue|Module\,1 = F, Module\,2 = T)$ can be calculated as follows:

$$\begin{aligned} p(Fatigue|M1 = F, M2 = T) &= k * p(Fatigue) * p(M1 = F|Fatigue) * p(M2 = T|Fatigue) \\ &= k * 0.2 * 0.1 * 0.85 \end{aligned}$$

To calculate the value of the constant k, we need to evaluate the formula $p2 = p(\sim Fatigue|Module\,1 = F, Module\,2 = T)$. Given that the sum of both probabilities $p1$ and $p2$ is 1, the constant k can be calculated.

Despite the fact that we calculated the final decision using the probability rules, there are automatic algorithms that are used to implement the inference engine. The most common algorithm is variable elimination [42]. Variable elimination eliminates the unobserved and non-query variables using summation. After eliminating the variables, the variable elimination algorithm normalizes the result, as we did in the previous example.

Chapter 3

Proposed Approach

The goal of this thesis is to develop a car driver fatigue monitoring system that is robust and efficient. Most previous work depended on the use of vision data as their primary source for fatigue detection. Depending on one data type reduces the system robustness under different conditions. Hence we decided to use different types of data gathered by different sensors, and fuse them to automatically detect driver fatigue. In order to do so, we need to build our own dataset. The available datasets are very limited and do not provide the flexibility to implement our proposed system. In the following sections, the system architecture is described, and the data collection and the dataset description is presented. Each module in the system is then described in detail.

3.1 System Architecture

The system architecture as shown in Fig. 3.1 is divided into four main parts: 1) The first part is the vision module which takes the recorded videos and propagates the inattention and fatigue decisions to the Bayesian Network. This part, as you can see, is a black box; detailed information about this part can be found in [43]. 2) The second part is the audio part. This takes the recorded sound as an input, processes it through three modules to detect the level of fatigue, and propagates the fatigue decision to the Bayesian Network for further analysis. 3) The third part is the Heart Rate and Other Signals module. This uses the recorded heart rate signal along with steering wheel, gas, clutch, and brake pedal signals and analyzes them to decide the level of fatigue. The decision is then transferred to the Bayesian Network for further analysis. 4) The last part is the Bayesian Network Decision Fusion. This module takes the output decisions from the different parts, and fuses them together in order to decide the final fatigue level.

In the following pages, a brief description for each module in Fig. 3.1 is presented.

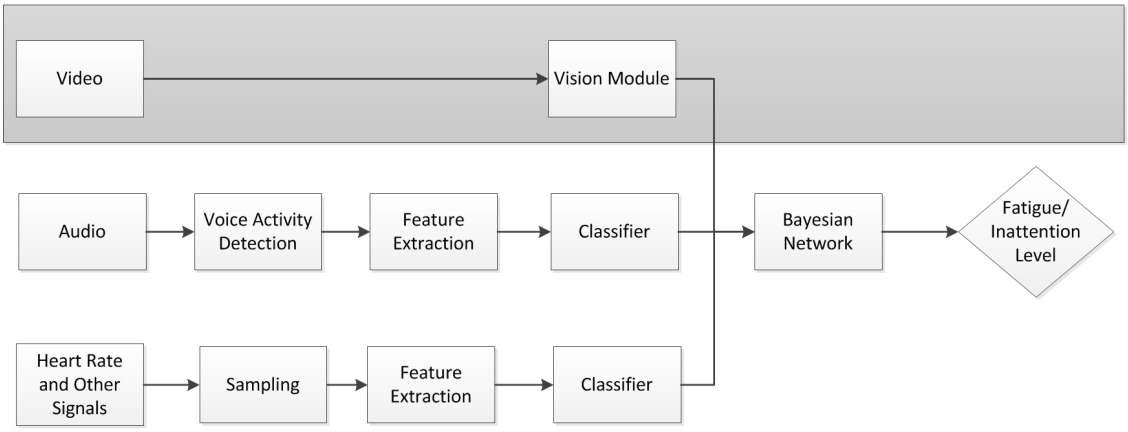


Figure 3.1: System Architecture

3.1.1 Audio Module

First, the recorded audio is processed by computer software. The audio samples, along with the sampling rate are given to the VAD module for processing. The VAD module is responsible for detecting the duration in which the driver is talking. The VAD module first divides the audio signal into small contiguous audio pieces. Feature extraction is applied to each piece to form the feature vectors,. The VAD combines a reasonable number of contiguous feature vectors into one big feature vector to form the frames. The classifier takes each frame as an input to decide the presence or the absence of the speech.

After the VAD extracts the speech parts, the feature extraction module takes them as an input to perform feature extraction. The feature extractor divides the speech signal into small contiguous audio pieces. Features are then extracted from each piece. The feature extractor module performs some statistical measures (ex. mean, average, etc) on feature vectors in order to produce one big feature vector to represent the whole speech duration. The following modules can either use the frame-based features or the one big feature vector.

Two different classifiers are used and compared: 1) Support Vector Machines (SVMs), and 2) Hidden Markov Models. SVM needs a fixed length feature vector, so using the one feature vector that represents the whole speech duration is the best option. On the other hand, Hidden Markov Model is able to model the temporal behaviour based on the frame-based feature vectors.

3.1.2 Heart Rate and Other Signals Module

The heart rate along with the other signals (steering wheel, gas, brake, and clutch pedal position signals) are processed by computer software. Each signal has readings and corresponding time stamps, and all readings from different signals need

to be aligned. Because each signal has a different sampling rate, in the same time frame there will be a different number of samples for different signals. We can apply different feature extraction modules, and then we combine the extracted features in one feature vector. The other solution is, we could simply convert the sampling rate for each signal so that the final sampling rate for all signals is the same. We can then apply feature extraction on the resampled signals.

For feature extraction, the output from the sampling module is processed in order to obtain feature vectors for the classifier module. The feature extraction module divides the signal into small contiguous pieces. Features are then extracted from each piece. The feature extractor combines a reasonable number of contiguous feature vectors into one big feature vector to form the frames. The feature vectors are transferred to the classifier in order to decide the level of fatigue.

Two different classifiers are used and compared: 1) Support Vector Machines (SVMs), and 2) Hidden Markov Models. SVM needs a fixed length feature vector, in which case the feature extraction module is asked to combine a large number of contiguous feature vectors. On the other hand, a Hidden Markov Model can deal with the original feature vectors without the need to combine contiguous ones.

3.1.3 Decision Fusion Using Bayesian Networks

A Bayesian network is used to fuse different decisions by different modules in order to determine the final decision level for both fatigue and inattention. The decision from each module is defined as a random variable in the network, and the module accuracies are used to fill in the conditional probability tables (CPTs). Driving conditions are also defined and added to the network, and a subjective method is used to fill in the CPTs. To infer the fatigue, different decisions are used as the evidence along with the driving conditions. The variable elimination algorithm is used to infer the final decisions for fatigue and inattention.

3.2 Dataset Description and Collection

The quality of the dataset is one of the primary factors that influences the performance of the system. We could not find any available dataset online that can be used to train and test our system, so we decided to build our own dataset. In the following subsections, the dataset description and collection is presented:

3.2.1 Dataset Description

The main goal is to collect as many signals as possible to be able to accurately detect fatigue and inattention. The advantage of gathering the data using different sensors is that the collective fatigue inference will be more accurate than using only one sensor. The correlation between each signal and fatigue or inattention varies under different environments. Using different sensors stabilizes such correlation and hence keeps the system more robust.

Another aspect of the dataset is to keep it non-intrusive as much as possible. Drivers don't like attaching something to their head or their body while driving. Drivers will eventually give up using intrusive-based systems even if they are proven to be useful. Hence using intrusive systems is impractical.

Another important aspect of the dataset is to collect data from different age and gender groups. Older drivers tend to drive more safely than young drivers. Surprisingly, women drive more safely than men according to Quality Planning Company QPC study[44]. According to the study, men are cited for reckless driving 3.41 times more than women. Men are also 3.09 more times than women to get a ticket for being drunk while driving.

Finally, the dataset needs to be collected during different times and road driving conditions. The dataset needs to capture drivers' behaviour while they are awake and fatigued. The dataset also needs to capture the drivers' behaviour in different road driving conditions (e.g., city driving and highway driving). The behaviour of the driver in different lighting and driving conditions is used by intelligent systems to indicate whether the driver is fatigued or not.

3.2.2 Data Collection

Our goal is to carry out experiments on several drivers to assess their level of inattention and fatigue. The subjects will be asked to drive the simulator in situations of inattention or fatigue. Several sensors will be used to obtain as much information as possible about the driver's behaviour. Building a practical dataset is a very hard task; we tried as much as possible to build a practical dataset. The first issue was whether to use real car or a driving simulator. It is really important to collect the dataset in a real environment, however subjects are asked to simulate inattention and fatigue which makes them susceptible to having accidents. For the sake of driver safety, we decided to use a car driving simulator as shown in Fig. 3.2. The device is simply a chair sitting behind a number of screens. The main driving controllers (e.g., gear shift lever, gas, brake, and clutch pedals) are attached to the

Figure 3.2: Car Driving Simulator

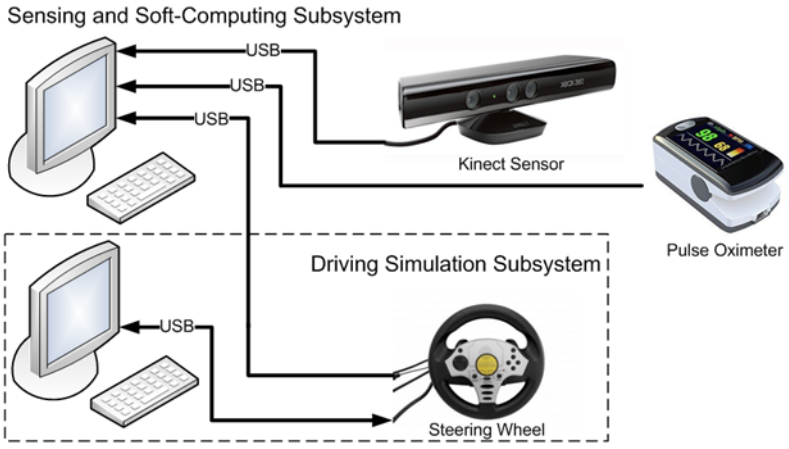


Figure 3.3: The types of sensors installed on the car driving simulator

chair. Software is used to display the driving conditions (e.g., highway/city driving, and day/night).

Various sensors are installed on the car driving simulator to record the driver's behaviour during the experiment. As shown in Fig. 3.3, a Kinect device is installed to capture different types of images (e.g., depth image, and colored image). Such images help track the driver's motions, and also in automatically analyzing his/her behaviour during the driving session. The Kinect device also records the audio during the driving session. The video and the audio signals are transferred to the computer through USB. Other software is used to store the audio and video signals in digital format.

Another sensor is used to monitor the heart rate of the driver during the driving session. There are steering wheels that are able to measure the heart rate without the headache of attaching a device to the driver's fingers; such steering wheels could be used in real life to replace the classic pulse oximeter shown in Fig. 3.3. The pulse oximeter records a new reading each second. The readings after the driving session are transferred into the computer and saved in "MS Excel" format.

Finally, a third software package is used to record readings for the steering wheel, gas, brake, and clutch pedal positions every millisecond. The position of these controllers can be used to indicate the driver's fatigue or inattention[45]. The positions recorded during the driving session are then transferred into the computer and saved in text format. This experiment has been approved by the Office of Research Ethics at the University of Waterloo. All participants have signed a consent allowing us to use the data we collected including their images and voices.

3.2.2.1 Experiment General Information

- Eight subjects (five men and three women) are selected to perform the experiment.
- Two driving sessions of thirty mins:
  - One in the morning when the driver is awake.
  - One in the late evening after a working day when the driver is fatigued.
- For each session, there are two driving conditions:
  - City driving condition with high traffic.

  - Highway driving with low traffic.

- Five minutes of free driving before the experiment to make the driver comfortable with the driving environment.

- Instructions will be provided to the driver while driving through a headset and through one of the screens.

3.2.2.2 Driving Session

For each session, the driver is asked to perform certain actions to simulate either fatigue or inattention. Each action is repeated twice for each session, and once for each driving condition. The actions that are related to fatigue are only performed during the evening session. The various tasks or actions are as listed below:

Actions that simulate inattention:

1. Phone call.

2. Text messaging.

3. Drinking.

4. Map search/ Road distraction.

5. Adjusting the radio.

6. Eye closing for a few seconds.

7. Yawning.

8. Nodding.

The first five tasks are related to inattention, while the last three tasks are related to fatigue. More detailed information about different tasks can be viewed in Appendix A.

Time (min)	0:00-3:00	3:00-5:00	5:00-6:00	6:00-7:00	7:00-8:00	8:00-10:00	10:00-12:00	12:00-13:00	13:00-14:00	14:00-15:00
Action		1		2		3		4	5	
Time (min)	15:00-18:00	18:00-20:00	20:00-21:00	21:00-22:00	22:00-23:00	23:00-25:00	25:00-27:00	27:00-28:00	28:00-29:00	29:00-30:00
Action		3		5		1		2	4	

Figure 3.4: Timeline for the morning session. The gray areas indicate "normal driving", while black areas indicate "loading map and normal driving".

Time (min)	0:00-3:00	3:00-4:00	4:00-4:30	4:30-5:30	5:30-6:00	6:00-8:00	8:00-9:00	9:00-12:00	12:00-12:30	12:30-14:30	14:30-15:00
Action		2		4		3		5		1	
Time (min)	15:00-18:00	18:00-19:00	19:00-19:30	19:30-20:30	20:30-21:00	21:00-23:00	23:00-23:30	23:30-33:30			
Action		2		3		5	1				

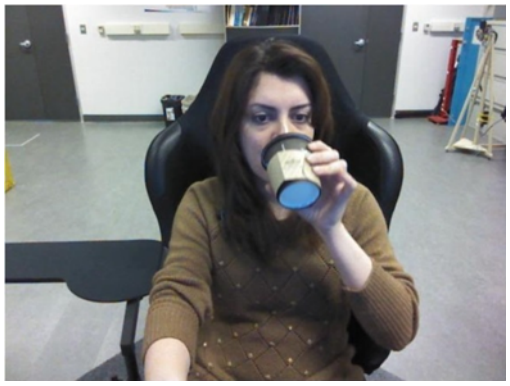
Figure 3.5: Timeline for the evening session. The gray areas indicate "normal driving", while black areas indicate "loading map and normal driving".

We can see the timeline for sessions one and two in Figs. 3.4 and 3.5. The session starts with loading the map and making the driver familiar with the environment. The gray areas are basically normal driving, where the driver just follows the traffic without performing any extra tasks. The numbers in the timeline from one to five are related to the tasks or actions listed above. The normal driving during the evening session is different than the morning session. The driver is asked to perform the tasks related to fatigue (tasks six, seven, and eight) during these times. We can see examples of the different tasks in Fig. 3.6.

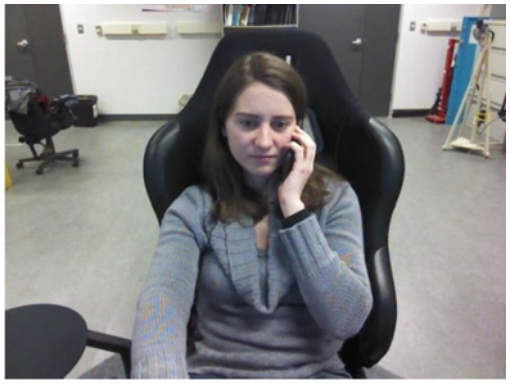
3.3 Driver Fatigue Detection Using Speech

In order to analyze the audio signal for fatigue detection, we first have to deal with the parts in which the driver is speaking. The reason behind using speech signal only is that we want to train the classifier on patterns where the driver's behaviour is involved. The parts where the driver doesn't talk are considered a type of noise, they don't indicate whether or not the driver is fatigued. Given that the speech parts are much fewer compared to the non-speech parts, it is necessary to extract the speech first. Otherwise the classifier will be trained by samples which are largely noise.

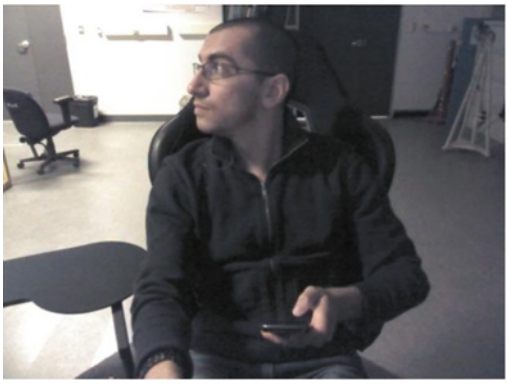
After the VAD extracts the speech parts, the feature extraction module takes the speech parts as an input to perform feature extraction. The feature extractor



(a) Drinking



(b) Phone call



(c) Road distraction

Figure 3.6: Example of different experiment tasks

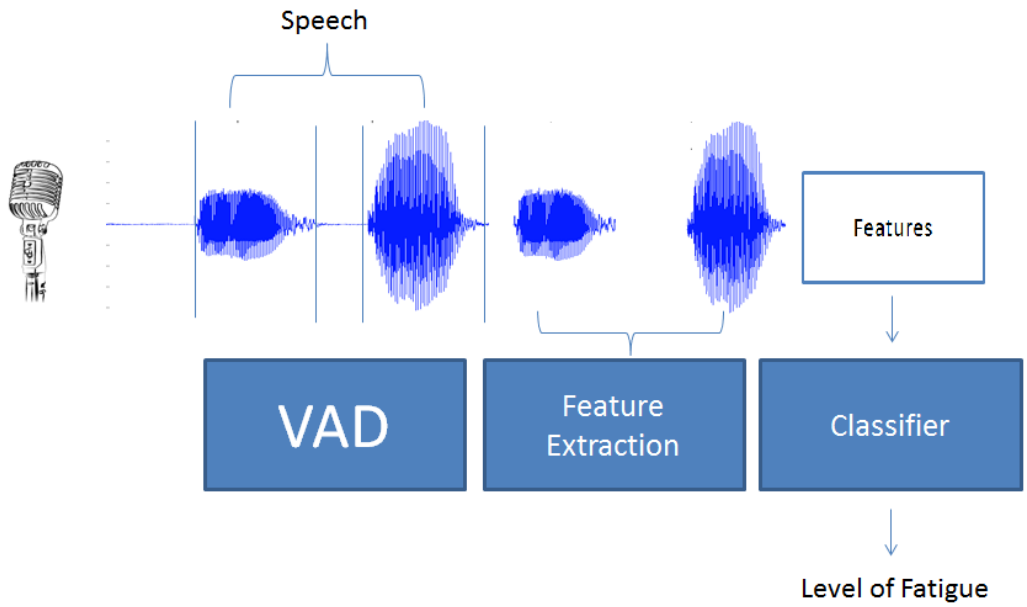


Figure 3.7: Audio Module

divides the speech signal into small contiguous audio pieces; features are then extracted from each piece. The feature extractor module performs some statistical measures (e.g., mean, average, etc) on feature vectors in order to produce one big feature vector that can represent the whole speech duration. The following modules can either use the frame-based features or the one big feature vector.

Two different classifiers are used and compared for this module: 1) Support Vector Machines (SVMs) and 2) Hidden Markov Models. SVM needs a fixed length feature vector, in which case the feature extraction module is asked to combine a large number of contiguous feature vectors. On the other hand, the Hidden Markov Model can deal with the original feature vectors without the need to combine contiguous ones.

3.3.1 Voice Activity Detection

Voice Activity Detection (VAD) is still an unsolved problem affecting many applications including robust speech recognition [46]. In general, the VAD needs to model different types of noise in order to be able to accurately detect the speech/non-speech signals in different environments. In our case, the problem is different, we only need to model types of noise that occur in the car cabin. These types of noise include steering wheel sounds, pedals sounds, air sound, etc. The limited number of noise types makes it easy to design a voice activity detector that is very accurate and robust.

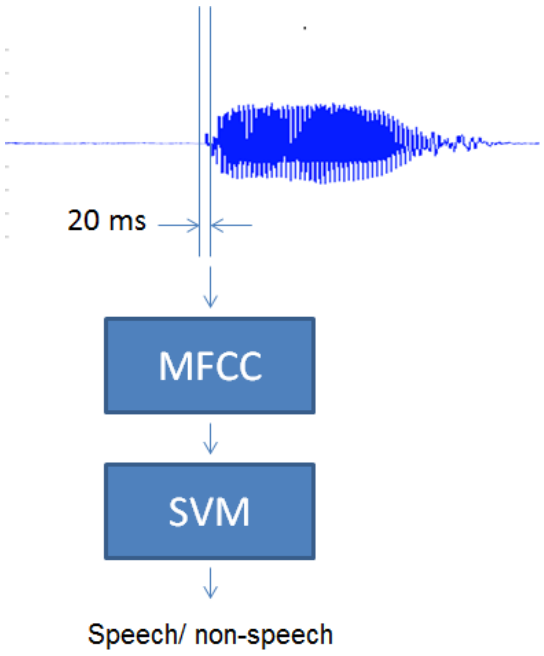


Figure 3.8: Voice Activity Detection based on 20 milliseconds frames.

As shown in Fig. 3.8, the system classifies the signal into speech/non-speech based on a small time frame (20ms). The audio signal is divided into small time frames of 20ms and a moving step of 10ms. A hamming window is applied on each time frame, then the MFCC features are extracted. An SVM classifier is used to classify the extracted features into speech or non-speech.

In the training phase, the audio signal is also divided into small time frames of 20ms and a moving step of 10ms. MFCC features are extracted after applying the hamming window. Each time frame in the dataset is manually labeled as either speech or non-speech. One-fifth of the dataset is left out for testing, while the rest is used for training. The feature vectors, along with the corresponding labels, are used to train the SVM classifier. Different Kernel functions and different penalty parameter ('C') values are used to tune the SVM classifier.

Unfortunately, the results are not as good as we want; the accuracy of the previous system is almost 75%. Detailed results regarding this system are discussed in Chapter 4. In the meantime, we want to discuss the reasons for such a low accuracy. We can understand that in a very small window (10ms), the human ear is not able to accurately classify the frame into speech or non-speech. Even if the energy within the small frame is high, one can't decide for sure whether it is a human voice or a noise. We have to look at a large window in order to be able to classify the current frame accurately, and so do the SVM.

Another explanation of the bad results, as shown in Fig. 3.9, is that the features are strongly overlapped. We can see that the 'Non-Speech' class is distributed over

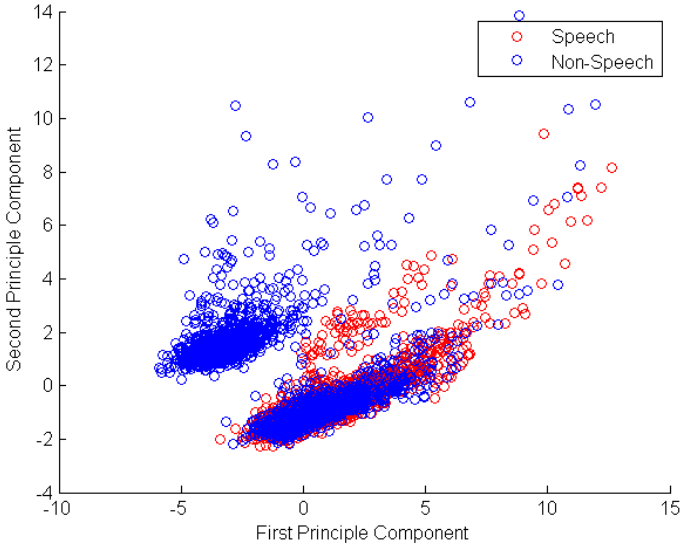


Figure 3.9: First two principle Components of the 20 milliseconds based features

two main clusters. The upper left blue cluster represents the silent frames, while the lower right blue cluster represents the noisy frames such as gas pedal sound or steering wheel sound. The 'Speech' class consists of one red cluster that is highly overlapped with the noisy 'Non-Speech' cluster. This is why this system produces ~75% accuracy using the best training setup for the SVM classifier.

As discussed, the main problem with using a small window is that it is very difficult to distinguish between noise and actual speech. A good solution to this problem is using a large window size. One way to do this is to combine contiguous frames into one big feature vector. Such a solution will give us very large feature vectors, which are not suitable to train an SVM classifier given that the number of samples are limited. The other solution, as shown in Fig. 3.10, is to take some statistical measurements for the MFCC features over a large window. For example, we could measure the average value for each MFCC feature over n contiguous frames. The statistical measurements used are the mean, variance, median, maximum, and minimum.

As shown in Fig. 3.11, the first and the second principle components of the new features are plotted. We can clearly notice that the 'Speech' class became more separable from the 'Non-Speech' class than it was in Fig. 3.9. The 'Speech' class is now distributed over different clusters representing different speech sounds. Using a non-linear classifier can be used to classify this dataset easily. An SVM classifier is used with various kernel functions, the best accuracy obtained is ~96%.

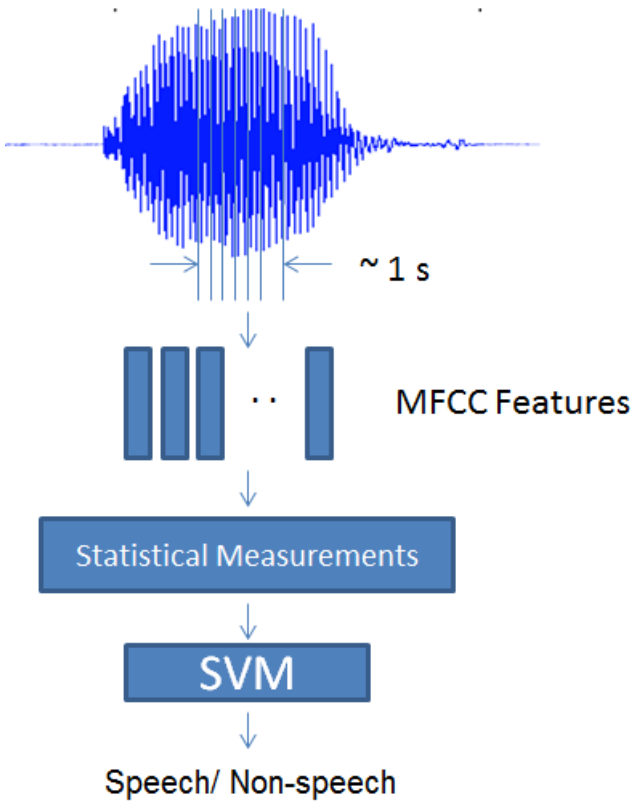


Figure 3.10: Voice Activity Detection based on 1 second frames

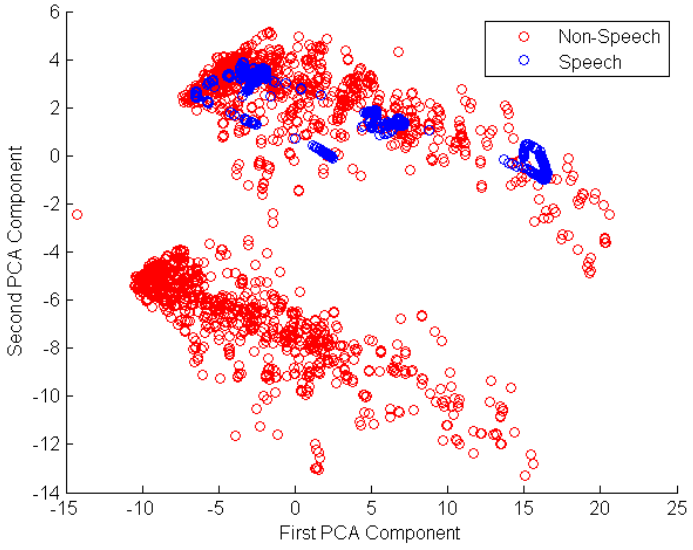


Figure 3.11: First two principle Components of the 20 milliseconds based features

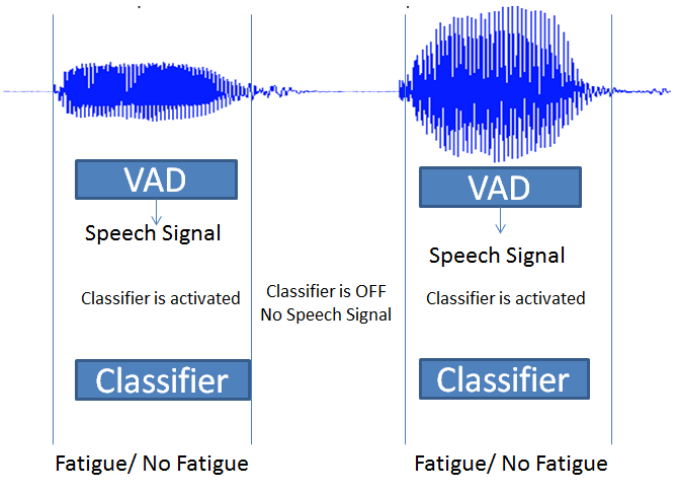


Figure 3.12: Speech analysis for driver fatigue detection.

3.3.2 Speech Analysis For Driver Fatigue Using SVM

As discussed, the classifier is only activated when a speech signal is detected. The first step is to extract all speech signals from the dataset and label each signal as either fatigue or awake. The feature extraction module takes the speech signals as an input to perform feature extraction. Finally the features, along with the corresponding labels, are used to train a classifier.

Choosing the right features is essential; the accuracy can be severely harmed when irrelevant features are chosen. The Mel-Frequency Cepstrum Coefficients (MFCC) features have been widely used for speech modeling and analysis, and they have been also used for drowsiness and fatigue detection [47, 48, 9]. The MFCC models the human auditory system through modeling the short-term power spectrum of a sound. To obtain the MFCC features, the speech is segmented into frames of 20ms and step of 10ms. A Hamming window is applied in order to reduce the abrupt transitions at the beginning and at the end of each frame. Each frame is converted to 12 MFCCs in addition to a normalized energy parameter. The first and the second derivatives (Δ's and $\Delta\Delta$'s) are computed, resulting in thirty nine coefficients representing each frame.

The feature vectors, along with the corresponding labels, are used to train an SVM classifier. Because an SVM cannot model temporal information, we have to come up with features that can represent the temporal information. The easy way to do this is by taking some statistical measurements on a number of contiguous MFCC frames. The statistical measurements that we chose are the mean, variance, median, maximum, and minimum for each coefficient. The final feature vectors have been formalized as follows:

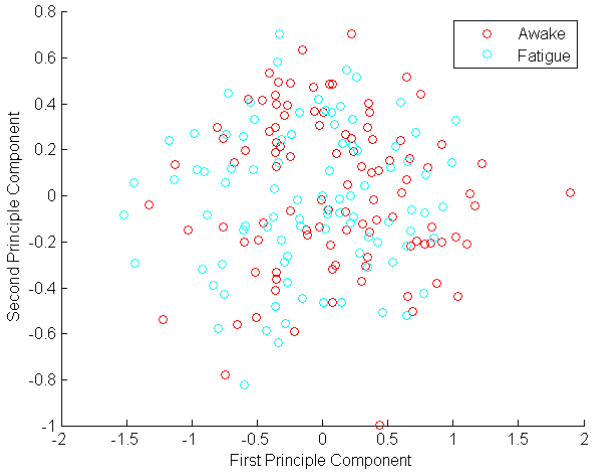


Figure 3.13: First two principle Components of the features used to train the SVM classifier.

$$F = [mean(m)\ variance(m)\ median(m)\ max(m)\ min(m)] \tag{3.1}$$

Where F refers to the final feature vector, and m refers to the MFCC frames. The m is a matrix, the number of MFCC coefficients represents the number of columns while the number of frames in the speech signal represents the number of rows. The final feature vectors are vectors of size $MFCC\,Coefficients * 5$, where 5 represents the number of different measurements that we calculate (i.e. mean, variance, ... etc).

The feature vectors F's are used to train an SVM classifier. Different kernel functions are used (e.g., linear, polynomial, and exponential). The penalty term is tuned to get the best accuracy. Although different tuning methods are used, the best accuracy is still very low. As shown in Fig. 3.13, the first two principal components are plotted. The figure shows that the two classes are strongly overlapped. Although the PCA preserves the variance, not the separability of the two classes, the figure gives us an explanation of the bad accuracy obtained by these features. Using a statistical measurement to represent the temporal variations is not a good solution to the problem. We have to look for a classifier that can capture such variations in order to obtain a better accuracy.

3.3.3 Speech Analysis For Driver Fatigue Using HMM

Hidden Markov Models (HMMs) are stochastic models which provide a high level of flexibility for modeling the structure of an observation sequence. They allow

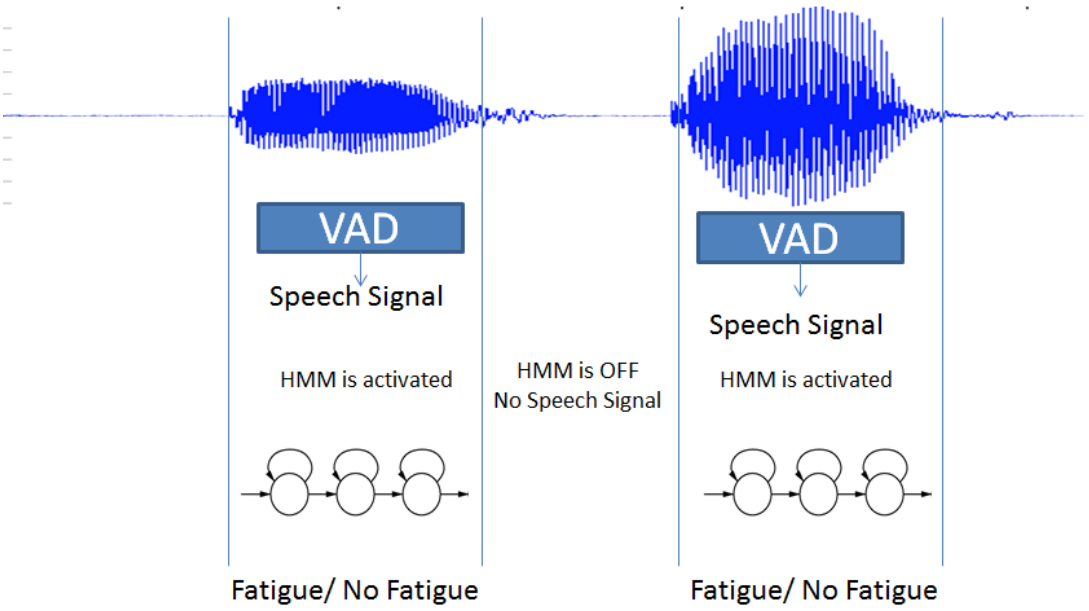


Figure 3.14: Speech analysis for driver fatigue detection using HMM system.

recognition of the hidden state sequence by the dependency of the observation sequence on the hidden states and the topology of the state transitions. It is now acknowledged that the use of HMM is fundamental in temporal pattern recognition.

We use the first order left-to-right model. A continuous HMM model is used to model the observations in which the emission probability is modeled using Gaussian Mixture Models (GMMs). The frame-based features are organized such that each feature stream is labeled as either fatigue or awake. The HTK toolkit is used for models training and recognition [49]. First, the emission probability is modeled using single Gaussian distribution. The means and the covariances of the Gaussian are initialized equally to the mean and the covariance of the data. The means, variances, and transition probabilities are re-estimated using a Baum-Welch algorithm. For better modeling of the emission probabilities, we increase the number of Gaussians during the training by splitting each Gaussian into two. Increasing the number of Gaussians is limited by the number of training data points; as a rule of thumb, each Gaussian needs at least fifty data points to train with.

For the recognition, a Viterbi decoder is used along with the trained model to classify a new stream of features as either awake or fatigue. Mean, variance, median, maximum, and minimum are used as statistical measures. The features are organized as stated in Eq. 3.1.

3.4 Driver Fatigue Detection Using Heart Rate and Other Signals

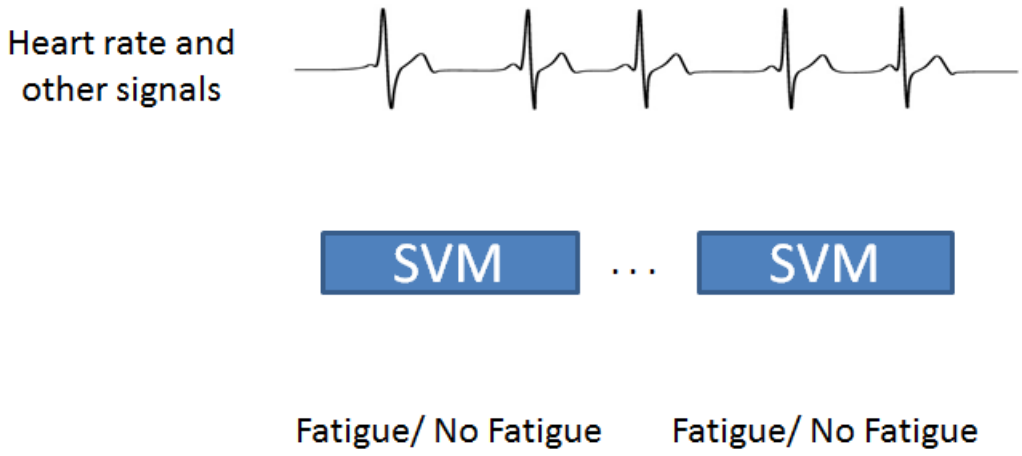


Figure 3.15: Analyzing heart rate and other signals for fatigue detection using SVM system.

In this part, the heart rate along with the steering wheel, gas, clutch, and brake pedal positions are used to analyze the level of fatigue. We designed a separate module for these signals because, unlike the speech signals, they are continuous throughout the driving sessions. In the speech module, the VAD is used to activate the HMM classifier whenever it detects a speech signal. Due to the continuity of heart rate and control position signals, as shown in Fig. 3.15, the classifier is always activated.

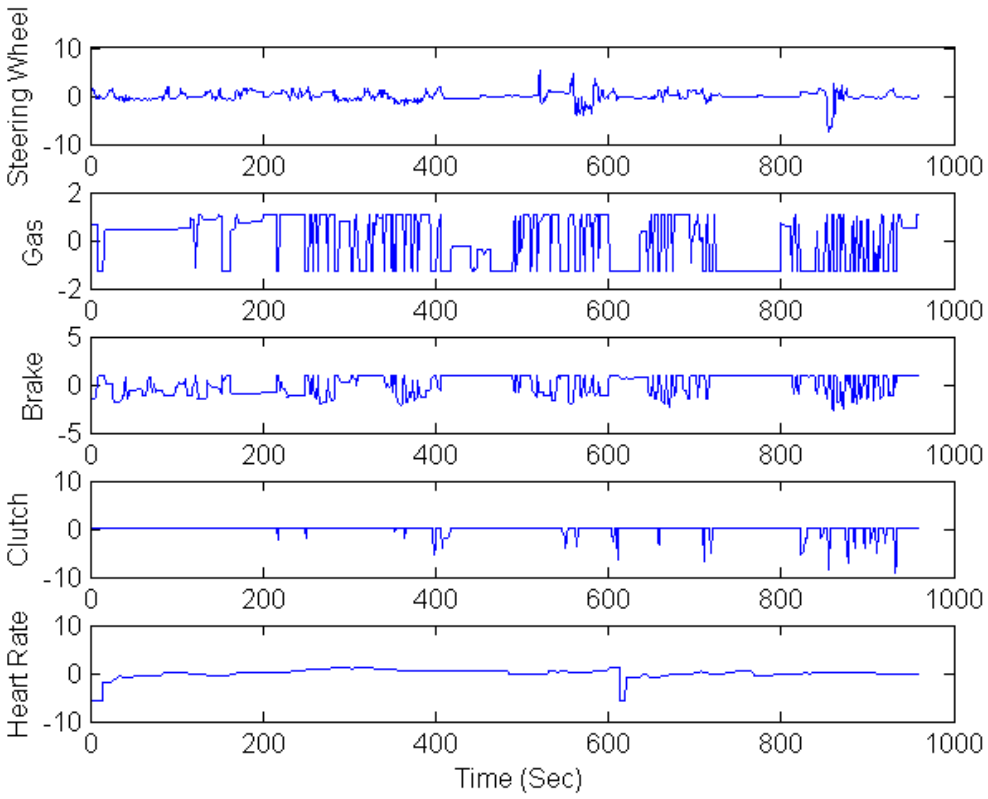


Figure 3.16: Steering wheel, gas, brake, clutch positions and heart rate for a complete driving sessions for one of the drivers.

As shown in Fig. 3.16, the values of most of the signals are discrete. This property makes it mandatory to use appropriate classifiers that can model the features correctly. The sampling rate is different for each signal. Different sampling rates need careful design of feature extraction and classification steps. We choose not to work with different sampling rates, and thus we resampled the signals down to a common sampling rate of 1/second.

3.4.1 Using SVM for Driver Fatigue Detection

For feature extraction, we use the raw features, first and the second derivatives (Δ's and $\Delta\Delta$'s). SVM and HMM classifiers are used for classification. For SVM and as we did in the audio module, some statistical measurements on a number of contiguous features frames are computed. The feature vectors and the corresponding labels are used to train an SVM classifier.

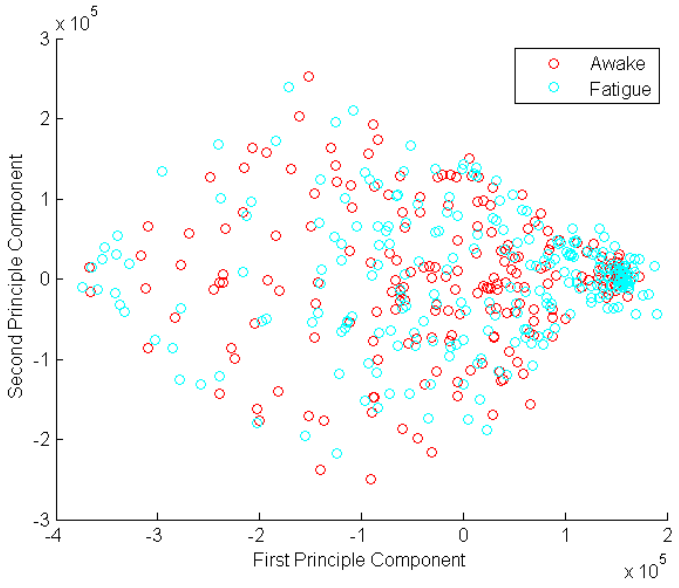


Figure 3.17: First two principle Components of the MFCC based features.

Not surprisingly, the performance of the SVM-based system is almost the same as the SVM-based system for the audio module. The reason is that the SVM classifier can't model temporal variations. Using features that can capture temporal variations is sometimes useful (e.g., VAD module), finding such features could be very difficult. The first and the second principle components for the features are shown in Fig. 3.17. This is obvious that the first two principal components for the two classes (Fatigue/No Fatigue) are strongly overlapped. It is an indication that the features are also overlapped.

3.4.2 Using HMM for Driver Fatigue Detection

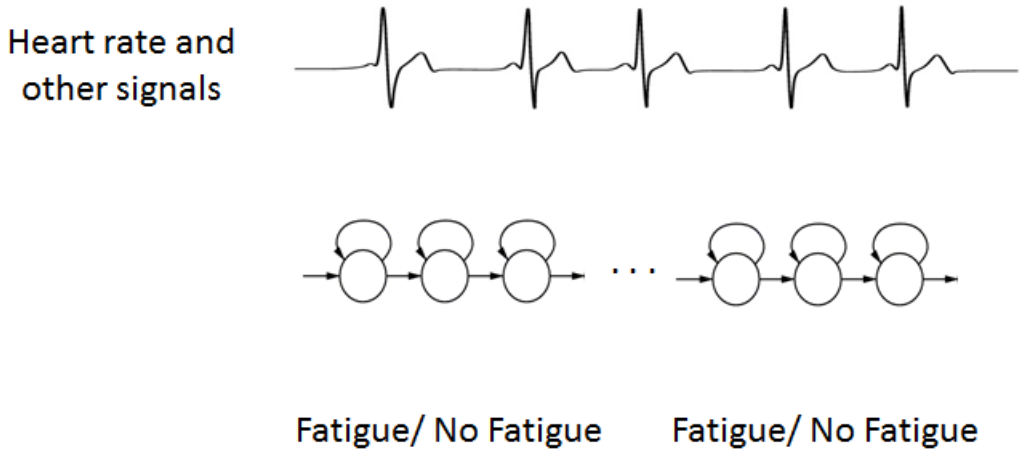


Figure 3.18: Analyzing heart rate and other signals for fatigue detection using HMM system.

Due to the discrete nature of the features, a discrete HMM classifier is used to model the discrete features. For feature extraction, we use the raw features, first and the second derivatives (Δ's and $\Delta\Delta$'s). To use a discrete HMM classifier, the feature vectors needs first to be quantized. The LBG clustering algorithm is used to cluster the features. The number of clusters k is chosen such that the clusters' centres are representative of the original features. The centres are numbered and used to generate the codebook. The codebook is used to quantize the features. Each feature is mapped to the nearest centre, and the cluster number is assigned to the feature.

The quantized features are used along with their labels to train an HMM classifier. The first order left-to-right model is used. The HTK toolkit is used for the training and the testing. The number of hidden states are varied to achieve the best possible accuracy.

3.5 Fatigue Inference Using Bayesian Networks

Final fatigue decision is predicted by fusing decisions from the audio module and the heart rate and other signals module. The fusion is done on the decision level in order to enable more decisions to be integrated without modifying the system

submodules (see future work section in Chapter 5). Various methods have been used for decision fusion. These methods include Bayesian networks, fuzzy logic, and Neural Networks. Fuzzy logic and Neural Networks do not provide sufficient capabilities to capture the uncertainties and dependencies that Bayesian networks can [50]. In addition, a Bayesian network provides a way to incorporate prior information when performing the inference.

3.5.1 Pre-processing

Table 3.1: Sample pre-processed output from second 160 to second 190 for one of the drivers.

Second	Audio Decision	Heart rate and other signals decision
160	-1	0
161	-1	0
162	-1	0
163	-1	0
164	0	0
165	0	1
166	0	1
167	0	1
168	0	1
169	-1	1
170	-1	1
171	-1	1
172	-1	1
173	-1	1
174	-1	1
175	0	1
176	0	1
177	0	1
178	0	1
179	-1	1
180	-1	1

First, the outputs of the audio module and the heart rate and other signals module need to be pre-processed. The final decision is estimated every second, and the decisions from each module need to be changed accordingly. For the audio module, fatigue decisions are available only in the presence of speech. The fatigue decisions need to be organized such that every second the fatigue decision is available. The output should take values of either 0 or 1, 0 meaning 'no fatigue' while 1 means

to fatigue is present. For those seconds where fatigue decisions are not available, a '-1' label is used. The '-1' label will not be used by the Bayesian network, it is just an indication that no fatigue decision is available at this second.

For the heart rate and other signals module, the fatigue decision is always available. It is also available each second as needed by the Bayesian network. The fatigue decisions only need to be re-labeled to '0' or '1', where 0 refers to 'no fatigue' and 1 refers to 'fatigue'. Fig. 3.1 shows a sample pre-processed output from second 160 to second 190 for one of the drivers.

3.5.2 Bayesian Network Design

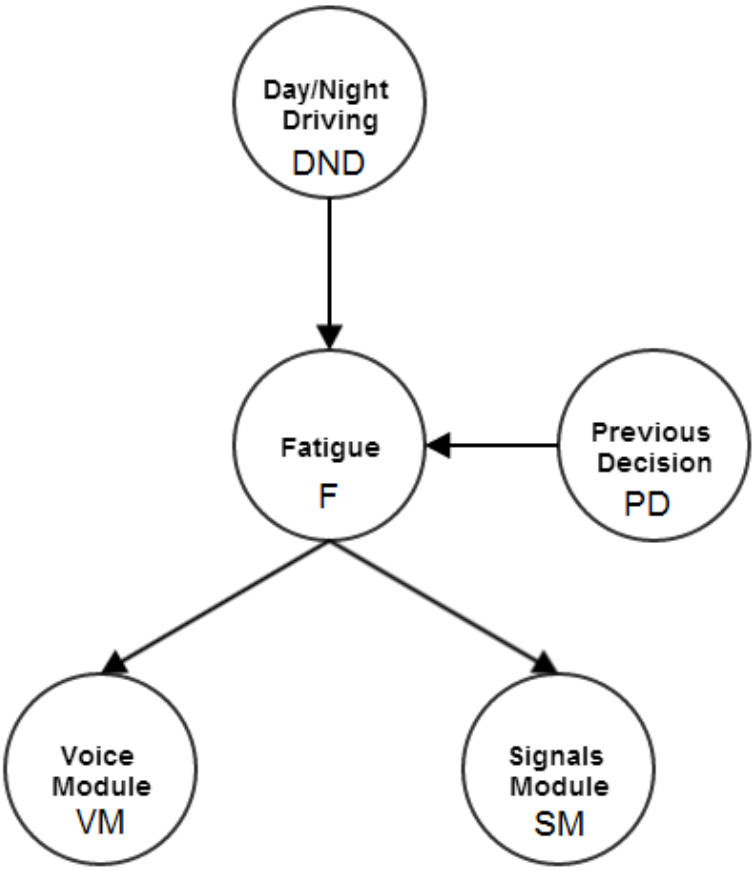


Figure 3.19: Decision fusion of the voice and other signals modules using Bayesian Network

Bayesian networks are probabilistic models that represent random variables and their conditional dependencies using directed acyclic graph (DAG). Each node in

the graph represents a random variable that can be either be discrete or continuous. An arc represents the conditional dependency between the parent node and the child node. The graph is parametrized by the conditional probability tables (CPTs). These tables specify the relation between each node and its parents.

As shown in Fig. 3.19, there are five different nodes; Day/Night driving (DND), Previous Decision (PD), Fatigue (F), Voice Module (VM), and Signals Module (SM). Each node (circle) represents a binary random variable. The values of each random variable is either 0 or 1. The arrows represent the causal dependencies between two nodes. The node at the tail of the arrow is the parent node, and the node at the head of the arrow is the child node.

The top part of network including the DND and PD nodes represents the prior information. The bottom part of the network including the VM and SM nodes represents the current decision information from different modules. The Fatigue node represents the final estimated decision. DND and PD are the parent nodes for the Fatigue node. The causal relation means that if the parent node behaves in some way, the child node will behave accordingly. For example, if it is known that the driver is driving at night, then fatigue is more probable than when driving during the day. The same applies for the previous decision. If the previous decision is found to be fatigue, the driver is more likely to be fatigued for the current decision.

Fatigue is the parent node for both VM and SM nodes. If it is found that the driver is fatigued, the voice module should capture the driver fatigue behaviour and produces its decision according to its accuracy. Also for the signals module, if the driver is fatigued, the signals module should capture the driver fatigue behaviour and the decision is then estimated based on the module accuracy. The nice property of such a setup is that conditional probability tables for VM and SM are calculated independently. Adding more modules can be easily integrated into the network by adding their conditional probability tables, which can be calculated independent of what other modules are attached to the network.

3.5.3 Inference

There are three conditional probability tables that model the upper part of the graph; $P(DND)$, $P(PD)$, and $P(F|DND, PD)$. These conditional probability tables are obtained subjectively. For the lower part of the graph, two conditional probability tables need to be estimated; $P(VM|F)$, and $P(SM)$. The confusion matrices for both modules are used to calculate the conditional probability tables.

The final fatigue level needs to be inferred from the network by the following query $P(F|DND, PD, VM, SM)$. This query is estimated each second, and the previous

decision is observed used the last estimated query. A variable elimination algorithm is used to evaluate the query [42]. Normally we would like all the random variables to be observed, however this is not always the case. There are some cases where some nodes are not observed. These cases are listed below:

1. At the first second, the previous decision node is not observed.

2. At the time where the audio decision is not available, the VM node is not observed.

At the first second, we take the previous decision from the query such that $P(F|DND, VM, SM)$. Doing the same for the times where the audio decision is not available will harm the accuracy, because the silent periods are much more prevalent than the voiced periods. By taking out the audio decision from the query, the audio decision becomes almost useless. So we define a window in which the last audio decision is taken and used as a current decision. If no audio decision is found within this window, then we infer the fatigue level by taking out the VM from the query and the query becomes $P(F|DND, PD, SM)$.

Chapter 4

Experimental Results

To train and test the system, we built our own dataset. We used a driving simulator device, sound recorder, and heart rate monitor for data collection. Eight people were involved in the experiments. They were asked to record two 30-min driving sessions; one in the early morning, and another in the late evening after a working day. For each session, the person was asked to make two phone calls. Each person's voice and heart rate were recorded during the session. The steering wheel, gas, brake, and clutch pedal positions were also recorded. The morning sessions were used to model the awakeness, while the evening sessions were used to model fatigue.

The dataset was partitioned such that 80% was used for training and 20% was used for testing. The ground truth was available each second. The system performance of the whole system was measured by the total average accuracy. The total average accuracy was calculated according to the following formula;

$$Total\ Average\ Accuracy = \frac{Number\ of\ correct\ decisions}{Total\ number\ of\ seconds} \tag{4.1}$$

.

4.1 Voice Activity Detection

The audio signal for six drivers were used to train and test the voice activity detector. The audio files were stored in wav format. One-fifth of the data was randomly selected for testing while the rest was used for training. MATLAB programming language was used to implement the module. The audio signals were read by the MATLAB program and stored in a matrix format. The data was then divided into

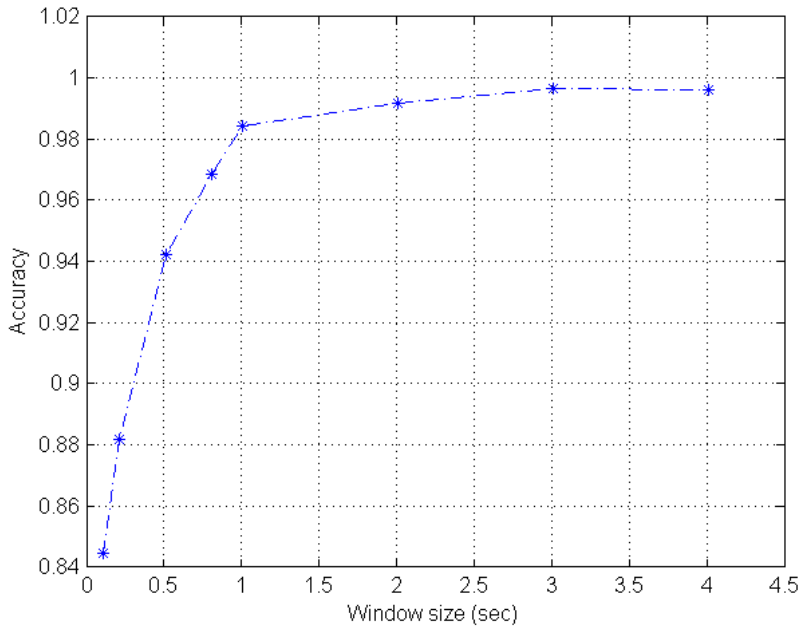


Figure 4.1: The accuracy of the VAD module plotted against the window size in seconds.

two matrices; one for training and another for testing. MFCC features were then extracted for both train and test data. The training data was used to train an SVM classifier using libsvm toolkit [51]. The trained SVM model was then used to estimate the labels for the test data. The accuracy finally was obtained by calculating the relative frequency of the correct labels to the total number of samples.

4.1.1 Tuning Window Size For The Features

As explained in chapter 3, the MFCC frames are not used directly to train the SVM classifier. Each MFCC frame represents 20 milliseconds of an audio signal. MFCC frame is calculated each 10 milliseconds. A moving window is used to extract the final features used to train the SVM. Statistical measures (e.g., mean, variance, maximum, etc) are obtained from the MFCC frames that are inside the moving window. A step of 10 milliseconds is used, so a window of x seconds contains a number of MFCC frames equal to $x * 100$.

In Fig. 4.1, the accuracy is plotted against the window size. The window size is varied from 0.1 seconds (10 MFCC frames) to 4 seconds (400 MFCC frames). The accuracy is more like a log curve; it starts to increase rapidly at the beginning and saturates at almost 1 second. The curve reaches its maximum at 3 seconds where the accuracy is 0.996, then the accuracy drops to 0.995 at 4 seconds.

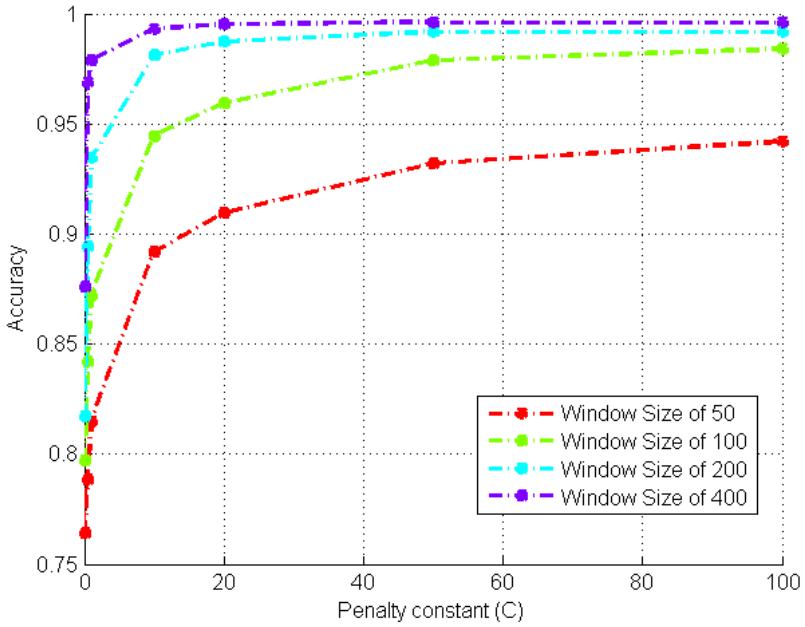


Figure 4.2: The accuracy of the VAD module for different window sizes plotted against the penalty constant (C) of the SVM classifier.

The reason for this behaviour is that when the window size is small, the information within each window is not enough to predict whether it is a human, non-human, or silence frame. As the window size increases, more information is fed into the classifier, allowing it to easily differentiate between speech and non-speech signals. The duration of the speech signal in our dataset is often less than 3 seconds. This makes it useless or sometimes even harmful to increase the window size to more than 3 seconds. We can actually see a slight decrease in accuracy after 3 seconds. In Chapter 3, Figs. 3.9 and 3.11 visualize the first two principal components of the features for window sizes of 0.01 and 1 seconds respectively. It is clear that 1 second-based features are easily separable compared to 0.01 second-based features.

4.1.2 Tuning The Penalty Constant For The SVM Classifier

Another important parameter to tune when training the SVM classifier is the penalty constant (C). Small values for C mean that the classifier is optimized to allow for misclassified samples; this choice is good when the classes are highly overlapped. Large values for the C constant mean that the classifier is optimized as a hard margin SVM. This choice is good when the data are separable in the kernel space.

In Fig. 4.2, the accuracy for different window sizes is plotted against the penalty constant (C). Here the window size is in number of MFCC frames, so we can obtain

the window size in seconds by dividing it by 100. The kernel used for the SVM is the third order polynomial, which experimentally proved to be better than other kernels for this problem. The accuracy for each window size is plotted in a different color. All the accuracy curves take the log curve. The accuracy increases rapidly at the beginning and then it saturates when the penalty constant is relatively high. The Figure shows that a high penalty value is a good choice to get a high accuracy. We can conclude that the samples in the kernel space are linearly separable.

4.2 Audio Module

The audio module takes a speech signal as an input, then decides whether or not the driver is fatigued. This module receives its input from the voice activity detector. One fifth of the speech signals are selected for testing, and the rest are kept for training. For this module, two different classifiers are trained and compared in terms of the accuracy. These classifiers are SVM and HMM. MATLAB programming language is used to implement the SVM-based module, while the HTK toolkit [49] is used to train the HMM-based module. Each classifier is tuned such that the best accuracy is obtained. In the following subsections, the accuracy for each classifier is presented.

4.2.1 SVM Classifier

The input to this module is a speech signal; the length of the speech signal is unknown. The speech signal is acquired by the MATLAB program and stored into two different matrices (training and testing). The MFCC features are extracted from each speech signal. We used a window size of 20 milliseconds and a moving step of 10 milliseconds. Because the length of the speech signal varies, the number of MFCC frames per speech signal also varies. The SVM classifier needs a fixed length feature vector. As discussed in Chapter 3, statistical measures are calculated from the MFCC frames. We used the mean, variance, median, maximum and minimum measures. After this step, the features become of fixed length. The features along with their corresponding labels are used to train the SVM classifier. Each speech signal is represented by one feature vector, and the accuracy of the SVM classifier is obtained by the following formula:

$$Accuracy = \frac{\#of\ correctly\ classified\ speech\ signals}{Total\ number\ of\ speech\ signals} \tag{4.2}$$

There were two main model parameters that needed to be tuned: the kernel and the penalty constant (C). In Fig. 4.3, the accuracy of the audio module is plotted

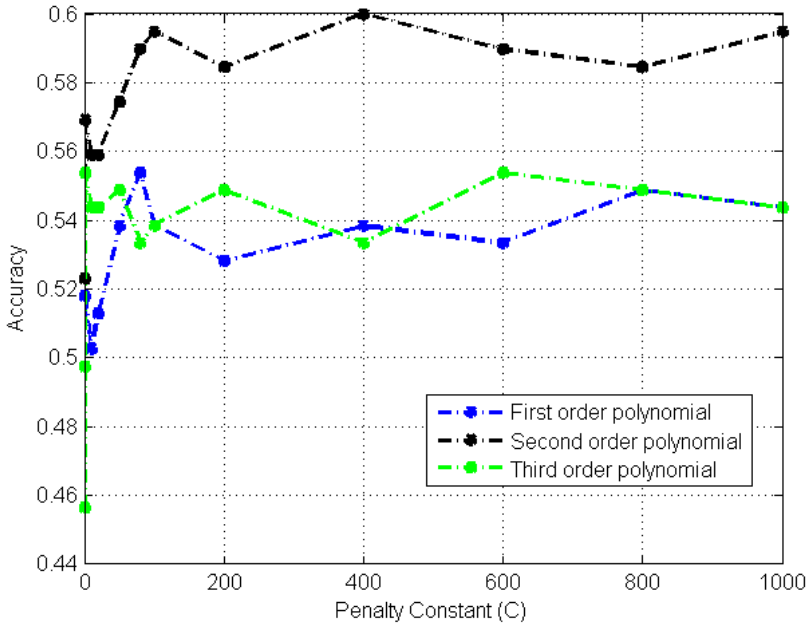


Figure 4.3: The accuracy of the audio module plotted against the penalty constant for the SVM classifier.

against the penalty constant. The accuracy for the first, second, and third order polynomial kernels are shown in the figure. We can see that the second order polynomial kernel performs better than both the first and third polynomials. Such behaviour suggests that the data is strongly overlapped such that the SVM classifier couldn't classify the two classes (Fatigue/No Fatigue) correctly. In the second order polynomial kernel space, the samples become more separable. The best accuracy obtained is 0.6, which is better than the rest of the kernels. In the third order polynomial kernel, we suffer from overfitting, in which the test results become less than the training results by a considerable amount.

The best accuracy obtained by the SVM classifier is still very low. Such results explain the reasons for such poor performance as mentioned in Chapter 3. The speech signal is a sequential signal, but the SVM classifier is not always suitable to model sequential data. We need to find the right features that can capture the temporal variations, but finding such features is not an easy task. We otherwise can use a classifier (HMM) that can model sequential data.

4.2.2 HMM Classifier

The HTK toolkit [49] was used to implement the HMM classifier. First MFCC features were extracted from each speech signals and stored in an HTK format. The

training was done over three steps: model selection, initialization, and embedded re-estimation. In model selection, the HMM paradigm was selected. In our design, the first order left-to-right model was selected. The number of hidden states was tuned for maximum accuracy. In initialization, the HTK toolkit was used to initialize the model parameters using the Viterbi algorithm. In embedded re-estimation, the Baum-Welch algorithm was used to re-estimate the parameters. As we use continuous HMM, the number of Gaussian mixtures had to be determined. We trained only one Gaussian mixture in the first few runs. We then split each Gaussian into four, and then retrained the model parameters. We did another split at the end, and ran the re-estimation algorithm for another few iterations.

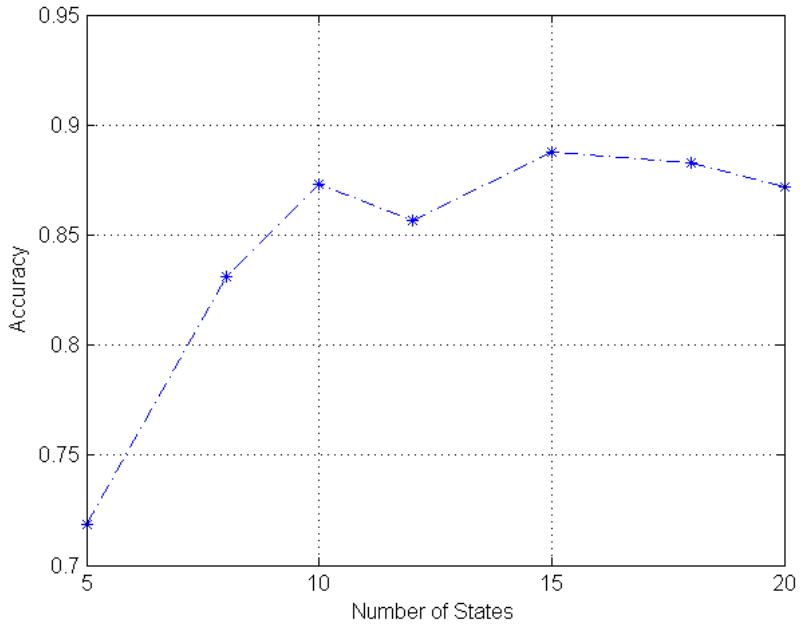


Figure 4.4: Total average accuracy when varying the number of HMM states for the audio module.

In the testing phase, the trained model was used to estimate the labels for each speech signal in the test dataset. The accuracy was calculated similarly to the accuracy for the SVM classifier in Eq. 4.2. As shown in Fig. 4.4, the average accuracy is plotted against the number of hidden HMM states. The accuracy first increases with increasing the number of hidden states, reaches the maximum at fifteen states, and then starts to fall again at twenty hidden states due to overfitting. We chose the number of hidden states to be fifteen. The best accuracy obtained from the SVM classifier was 59.1%, while the best accuracy for the HMM based system was 88.77%.

Table 4.1: Feature extraction example. F(t), F(t-1), and F(t-2) represented the raw features of three consecutive frames. The example shows how to calculate Δ and $\Delta\Delta$ features. The final feature frame for time t is the concatination of the three colums F(t), $\Delta(t)$, and $\Delta\Delta(t)$.

	F(t-2)	F(t-1)	F(t)	Δ(t-1)=F(t-1)-F(t-2)	Δ(t)=F(t)-F(t-1)	$\Delta\Delta$(t)=Δ(t)-Δ(t-1)
Steering Whell	-388	-4	817	384	821	437
Gas	14823	14823	14823	0	0	0
Brakes	20284	20284	20024	0	-260	-260
Clutch	32767	32767	32767	0	0	0
Heart Rate	98	98	98	0	0	0

4.3 Heart Rate and Other Signals Module

The heart rate and the other signals module use the heart rate, steering wheel position, and the gas, brake, and clutch pedal positions as inputs. The module tries to analyze the input signals to infer whether or not the driver is fatigued. One fifth of the signals are selected for testing and the rest are kept for training. Similar to the audio module, two different classifiers are used and compared as well. The input signals have different sampling rates. The sampling rate for the heart signals is one second, while the sampling rate for the other signals is around 50ms. The other signals are resampled to match the heart rate so the sampling rate for all signals becomes 1 second. The decision for this module is calculated each second. The accuracy is calculated according to the following formula:

$$Accuracy = \frac{Number\,of\,correct\,labels}{Total\,number\,of\,seconds} \tag{4.3}$$

4.3.1 SVM Classifier

The SVM-based module was implemented using MATLAB. Each second was represented by a frame containing raw signal values, Δ, and $\Delta\Delta$ as shown in table 4.1. Because the SVM classifier needs a fixed length feature vectors, a moving window was used to extract suitable features for the SVM classifier. The window size was tuned for maximum accuracy as shown in Fig. 4.3. Some statistical measures (mean, variance, median, maximum, and minimum) were calculated from the frames that are within the window. The features along with their corresponding labels were used to train the SVM classifier. The accuracy was calculated according to Eq. 4.3.

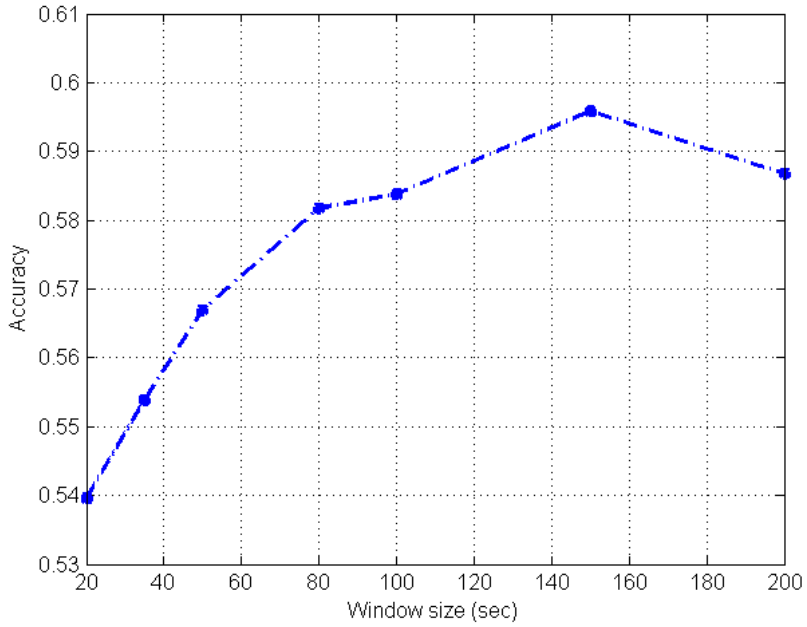


Figure 4.5: The accuracy of the heart rate and other signals module plotted against the window size.

The main parameter to model was the size of the moving window. In Fig. 4.5, the accuracy for the heart rate and other signals module is plotted against the window size. The window size is varied from 20 seconds to 200 seconds. The accuracy first increases along with the window size. It reaches its maximum at 150 seconds, where the maximum accuracy is 0.596. The accuracy decreases afterwards. The reason behind such behaviour is that the information within an appropriate window size reflects the driver's fatigue better than other window sizes. Empirically, the appropriate window size is 150 seconds.

4.3.2 HMM Classifier

Because the features take integer values, we preferred to use discrete HMM to avoid the singular covariance matrix problem. In discrete HMMs, the features are quantized using a codebook. The codebook is generated by clustering the training dataset; we used the LBG algorithm to cluster the dataset [52]. The discrete features along with the corresponding labels were used to train the discrete HMM model. The HTK toolbox was used to train and test the module. The features are the raw signal values, Δ, and $\Delta\Delta$. Each frame represents only 1 second; we left modeling the temporal variation to the HMM to capture. We chose the size of codebook to be 512. The number of hidden states was tuned for

maximum accuracy. In the test phase, the output of the HMM is post-processed to have a decision (Fatigue/No Fatigue) each second. The accuracy of the module is calculated according to Eq. 4.2.

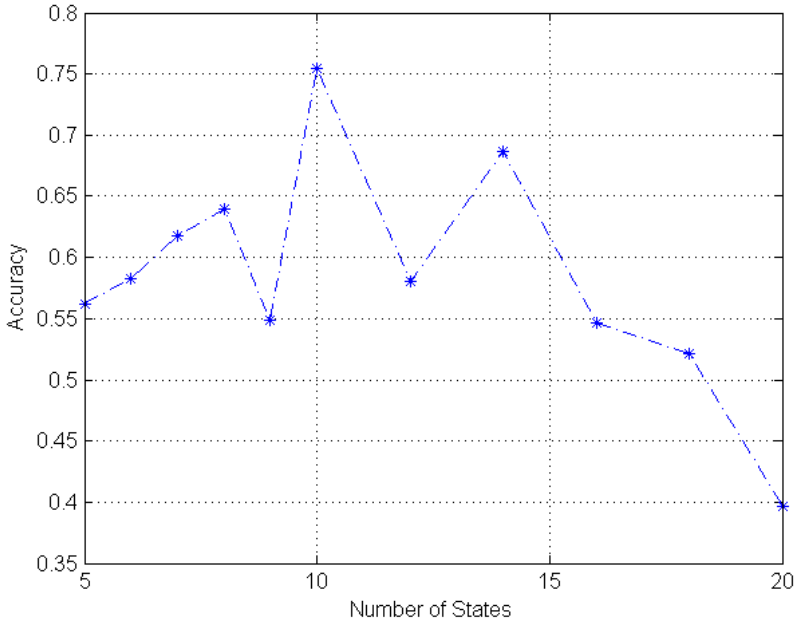


Figure 4.6: Total average accuracy when varying the number of HMM states for the heart rate and other signals.

Fig. 4.6 shows the accuracy against the number of hidden HMM states for the heart rate and other signals module. The accuracy tends to increase till it reaches its maximum at 10, then it tends to decrease afterwards. Due to the fact that the HMM problem is not a convex optimization problem, the HMM might be trapped in local maxima. This is why we see sometimes on the graph an increase in the accuracy while it should be decreasing and vice versa. The best accuracy obtained from the SVM classifier is 0.596, while the best accuracy for the HMM-based system is 0.755.

4.4 Bayesian Network

For fusing different decisions from different modules, we used a Bayesian network to infer the final decision. Fig. 4.7 shows the Bayesian network design along with the CPTs values. The CPTs for the lower part of the network were estimated using the accuracy for each module. The CPTs for the upper part of the network

were calculated subjectively. Variable elimination algorithm is used for inference calculations [42]. We used MATLAB programming to implement this module.

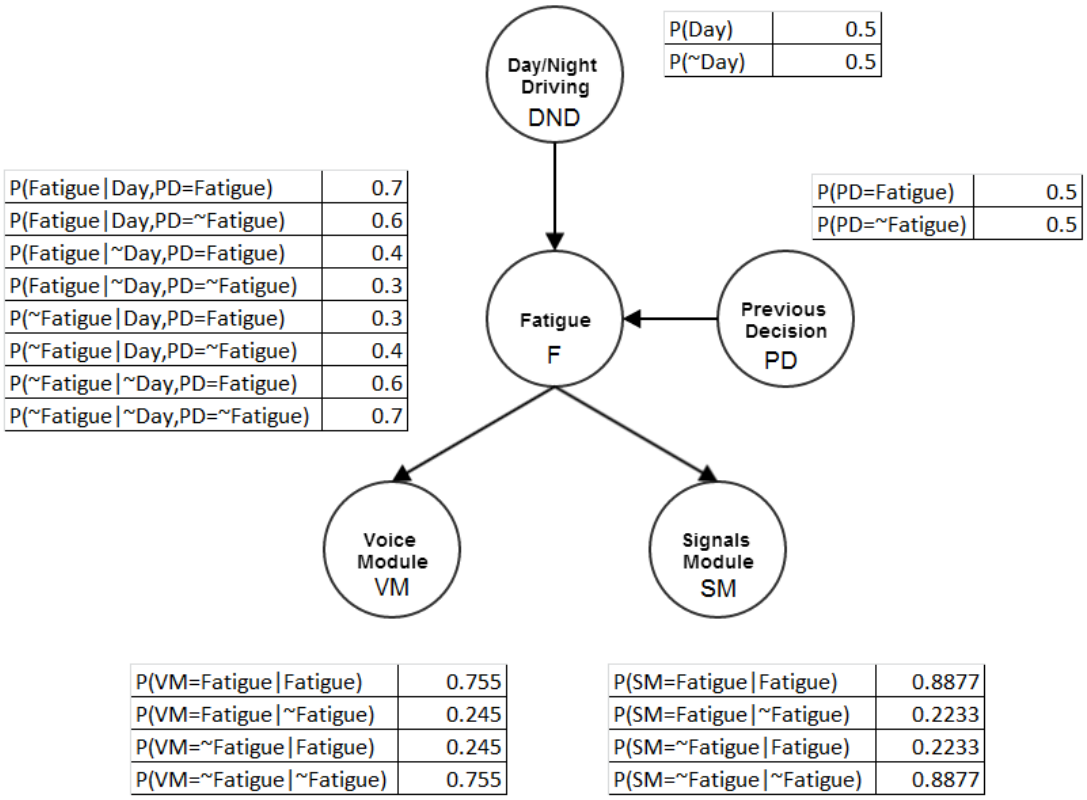


Figure 4.7: The Bayesian network used to fuse different decisions and the corresponding CPTs.

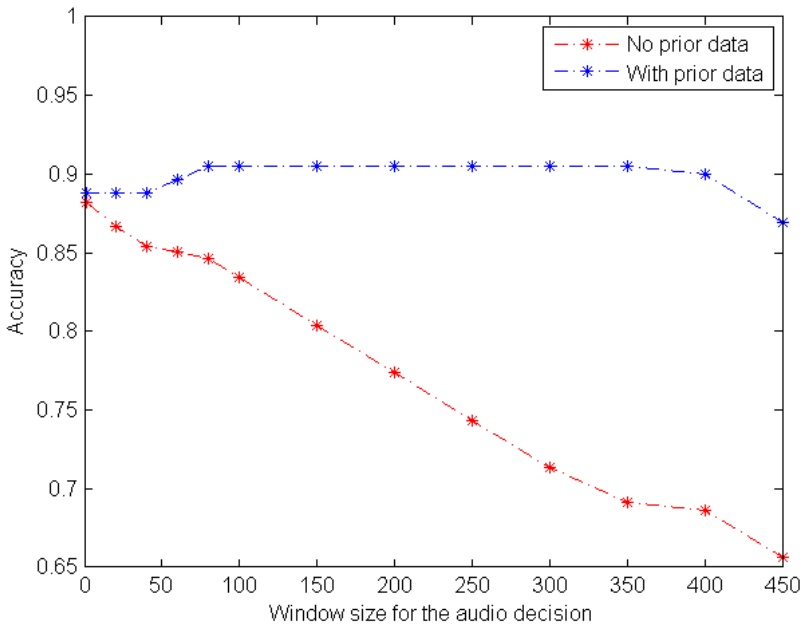


Figure 4.8: Figure shows the improvement in the total average accuracy for the final decision when taking the prior information into account. The window size is in seconds.

The Day/Night and Previous Decision Variables are considered prior data. Because the decision from the audio module is not always available, we use a window in which we track the last decision from the audio module and use it as the current decision. Because the voice signals are not distributed uniformly over the session, a wrong audio decision may seriously harm the accuracy, as we see in Fig. 4.8. Fig. 4.8 shows the average accuracy of the whole system, calculated according to Eq. 4.1, against the window size of the audio module. Without prior data, the accuracy decreases when the window size increases. This is due to wrong decisions being carried forward more often than correct decisions. By integrating prior data, the final decision is no longer dominated by the audio decision. The system becomes more stable and the best accuracy achieved is 90.5% at a window size of 100.

Chapter 5

Conclusions and Future Work

In this chapter we are going to present the conclusions while focusing on the contributions of this work. We are going to present some ideas for expanding this work in the future. Some challenges and problems will be mentioned and how to overcome them. Finally, the last section will present the publications based on this work.

5.1 Conclusions

In this thesis, a system for car driver fatigue monitoring has been presented. The system combines audio, heart rate, steering wheel, and pedal positions to decide the level of fatigue. The use of an HMM classifier has been experimentally proven to be a good solution to the problem. Combining more than one decision has improved the accuracy of the system by 3%. The next subsections briefly present the conclusions and the contribution for each module in the system.

5.1.1 Dataset

Building the dataset was a very challenging task. Most of the datasets that have been used by other research groups to build similar systems used either one sensor (camera), few human subjects, or non-realistic fatigue simulation. The motivation for building a new dataset that compensates for the gaps in other datasets was obvious. The goal was to capture the driver distractions and fatigue levels.

A driving simulator device was used to simulate the driving conditions. Many sensors were used to capture the driver's behaviour and fatigue level. These sensors

include Kinect, infrared camera, microphone array, heart rate monitor, steering wheel position, and gas, brake, and clutch pedal positions. The dataset was collected in different driving sessions where the drivers were fully awake/fatigued and attentive/distracted. Ten human subjects were involved in the experiments. Five distractions and two levels of fatigue were captured for each driver. The dataset was manually annotated and stored in a computer format.

5.1.2 Fatigue Monitoring System

The goal for the fatigue monitoring system was to acquire different signals, analyze them, and decide the driver fatigue level. The sensors used for this system were microphone array, heart rate monitor, steering wheel position, gas, brake, and clutch pedals positions. The system comprises three main modules: the audio module, the heart rate and other signals module, and the decision fusion module.

5.1.2.1 The Audio Module

Although some research was conducted to analyze the level of fatigue based on human voice, these systems weren't customized for driver fatigue monitoring. For the driver fatigue monitoring system, the audio module needs to extract the speech signals from the audio signal, and it needs also to be combined with other sensors to support the decision when the speech signal is absent. The audio module in our system was designed to acquire the audio signal, extract the speech parts, and decide the fatigue level from the speech signals. The main parts of this module were the voice activity detector (VAD), and the classifier.

The VAD problem is still an open one. However, when the environment is known and types of noise are limited, designing the VAD module is achievable. The types of noise in our dataset were limited to the steering wheel noise, pedals noise, and the human movements inside the room. The VAD module for our system was built using MFCC features and SVM classifier. At each time step, a window of 3 seconds was analyzed in order to accurately extract the speech signal. The final accuracy for the VAD module was 99.5%.

The classification of the speech signal step was the last step in the audio module. MFCC features were used in this step. At each time step a window of the MFCC features was analyzed using an SVM classifier to decide the level of fatigue. The experimental results showed that the performance of the SVM classifier was unsatisfactory. In the best case, the accuracy of the SVM classifier was 60%. Such accuracy drove us to model the speech signals using sequential modeling. Continuous HMMs were used to model the speech signals, and we trained and tested the data using the HTK toolkit. The HMM hidden states were tuned for best accuracy. The best accuracy obtained was 88.77%.

5.1.2.2 The Heart Rate and Other Signals Module

For this module, the heart rate monitor, steering wheel position, and gas, brake, and clutch pedal positions were used to monitor the level of fatigue. Due to the sampling rate for each signal being different, all signals were resampled to the same sampling rate (1 sample/second). The performance was determined by calculating the average accuracy per second. SVM was used to monitor the level of fatigue at each time step. At each time step a large window of signal values was analyzed. The SVM classifier was trained and tuned to obtain the best accuracy. The best accuracy for the SVM-based system was as low as 59.1%.

The problem of the SVM classifier for the heart rate and other signals module was similar to the problem of the SVM-based classifier for the audio module. The SVM wasn't able to capture the temporal variation in order to model the fatigue correctly. We used an HMM classifier to overcome this problem. For this module, the signal values are discrete, so the use of discrete HMMs was more appropriate. The features were clustered into 512 clusters using the LBG clustering algorithm, and were then quantized each to the nearest cluster. The HTK toolkit was used to train and test the system. The number of hidden HMM states was tuned for maximum accuracy. The accuracy was calculated used one fifth of the dataset, and the best accuracy obtained was 75.5%.

5.1.2.3 Decision Fusion Using Bayesian Networks

The main contribution of this thesis is combining more than one decision from different modules to assess the final decision. For the first time, the decision from an audio module was fused with other decisions in a car driver fatigue monitoring system. Due to the capabilities of the Bayesian network for capturing the dependencies and the uncertainty, we chose the Bayesian network to perform this fusion. We also incorporated the prior information (Day/Night driving and previous decision) to improve the final decision. The accuracies for the audio and heart rate and other signals modules were used to calculate certain CPTs for the Bayesian network. The rest of the CPTs were calculated subjectively.

The implementation of the Bayesian network was done using MATLAB. The inference was calculated using the variable elimination algorithm. For the time steps where the audio module decision was absent, a window was defined and the last decision within this window was used as a current decision. The performance of the system was assessed based on the average accuracy per second. The total accuracy of the system was 90.5%.

5.2 Future Work

Building the dataset was one of the important tasks for this thesis. This dataset made it possible to explore the possibility of using the audio signal along with the other signals in assessing the fatigue level. More experiments can be carried out to explore the use of the other signals in the dataset, such as the vision. There are some improvements that can be done for the dataset to make it more practical and realistic. In the following subsections, future work is presented for both dataset and system.

5.2.1 Dataset

When we built this dataset, we tried to avoid the drawbacks of other datasets we have came across. Despite the effort that has been spent to build this dataset, there are some improvements that need to be done to make it more practical and more realistic. Below are some improvements.

1. More Subjects: ten people were involved in the experiments. Only one was over thirty years old, while the rest were under thirty. The experience level of the drivers wasn't investigated in order to assess the fatigue level for drivers with different experience. For these reasons, it is important to involve more people in the experiments. The drivers need to be selected to represent all different car drivers.

2. More Fatigue Levels: in the dataset, there are only two fatigue levels that were captured during the driving sessions; fatigue and no fatigue. More levels need to be added to add precision to the fatigue decision.

3. Using Electroencephalography Signals For Fatigue Levels: National Highway Traffic Safety Administration (NHTSA) published a report on 2013 [53], stating that the peak fatigue time for most drivers is at night. However this might not be the case for those drivers who sleep mainly during the day. In our dataset, the drivers were asked to perform two driving sessions, one in the early morning and another after twelve hours of working day. Although drivers should be deprived of alertness after twelve hours of work, this is a subjective way of measuring fatigue. Another way of accurately labeling the level of fatigue is by using Electroencephalography signals as done by [11]. These signals are recording by attaching multiple electrodes to the brain.

4. Using Real Vehicle: the data was recorded in a quiet room. The seat of the driving simulator is fixed, and doesn't simulate the motions of real driving. Such aspects prevents the datasets from being practical and realistic. For example, the audio signal was clean and the only noise that was recorded were the steering wheel and pedal sounds. In real life, the street noise and the voices of other people will be also recorded making it a challenging task to extract the driver's voice. The main reason behind recording the dataset using a driving simulator device is the safety of the drivers. The drivers are asked to perform certain tasks that might result in accidents if they were driving real vehicles. Real vehicles can be used for safe tasks. The drivers can be asked to pull over before performing certain tasks in order to ensure their safety.

5.2.2 System

Our system comprises fusing two modules; audio module, and a heart rate and other signals module. Only car driver fatigue is monitored. The system could be extended as discussed below.

1. Features: for the audio module, although the MFCC features were proven to be appropriate for speech recognition, they might not be the best features to model fatigue. To assess such a claim, MFCC features need to be compared to or combined with other features. In the literature, fundamental frequency, energy, harmonics-to-noise ratio, formant position, formant bandwidth, duration of voiced-unvoiced segments, linear frequency cepstrum coefficients, and long-term average spetrum features have been used to detect the level of fatigue [54]. Feature selection algorithm can be used to select the best set of features, where maximizing the accuracy can be its objective function.
   For the heart rate and other signals module, many features could also be tried to improve the accuracy. The use of raw values has been proven to be ineffective when combined with the SVM classifier. Frequency based features such as DFT, or wavelet transform [55] could be explored. The use of different features can capture the temporal variations, making the use of the SVM classifier very useful.

2. Classifiers: in this thesis, we explored two classifiers: SVM and HMM. SVM was found to be very powerful in the VAD module, but it failed to capture the temporal variation in both the audio and the heart rate and other signals modules. On the other hand, HMM classifier was very successful in classifying the data for both audio and other signals modules. This opens the door for trying classifiers that can capture temporal variations. Conditional Random Fields (CRF) are very powerful and have been commonly used lately in various applications [56]. Restricted Boltzmann machines (RBM) classifier is also

a potential classifier, and it has performed much better than classic HMMs in some applications[57].

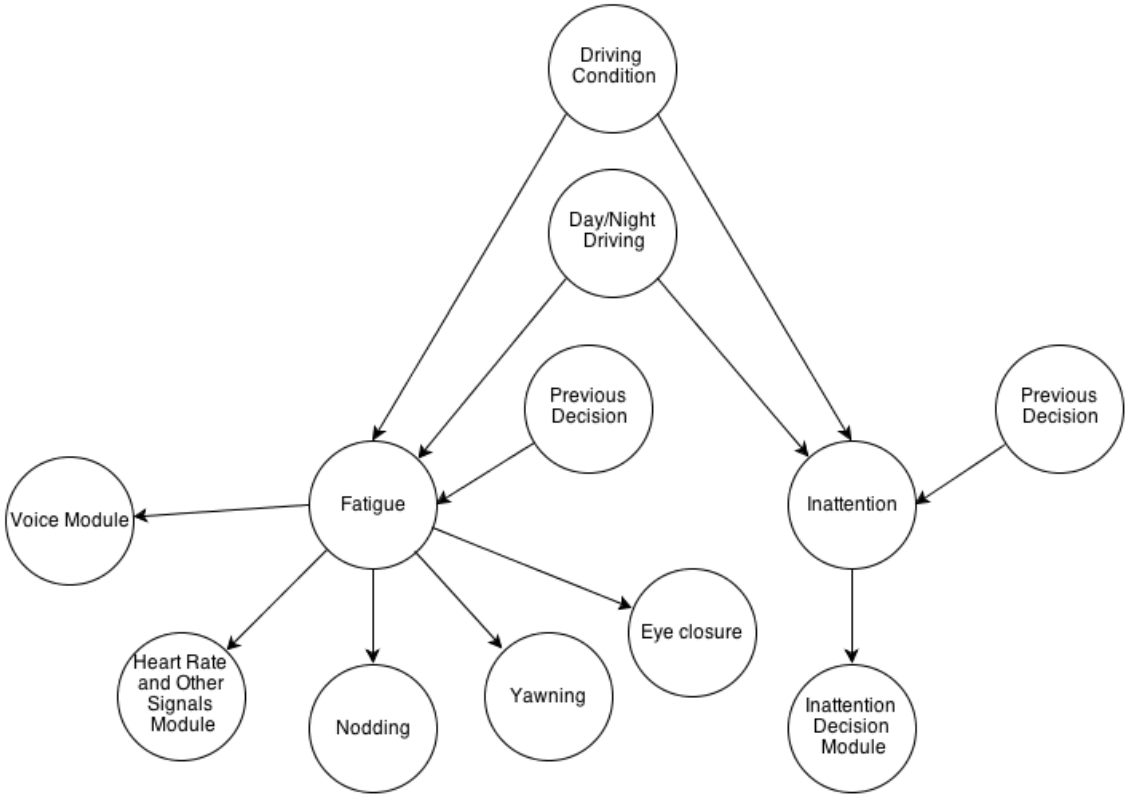


Figure 5.1: A proposed Bayesian network that fuses decisions from more modules.

3. Fusing More Decisions: the system could be extended such that more modules are added to support the final fatigue decision. The inattention level could also be integrated into the system. Fig. 5.1 shows a way to fuse decisions from different modules using a Bayesian network. The available dataset can be used without any modifications to build such system.
   In Fig. 5.1, the driving condition variable is added to the prior information. Driving condition is either city driving or highway driving. Nodding, yawning, and eye closure are three new modules that use vision information to assess the level of fatigue. The drivers were asked to perform each of these actions at the night driving sessions to simulate driver fatigue.
   On the right side of the Bayesian network, an inattention variable is added. Inattention can be monitored using different actions the drivers were asked to perform, as discussed in the dataset description and collection section in Chapter 3. The inattention decision module can be implemented using different vision techniques. One way to implement such a system is presented in [43]. The fatigue and the inattention levels can together be used to alert

the driver to potential risk.

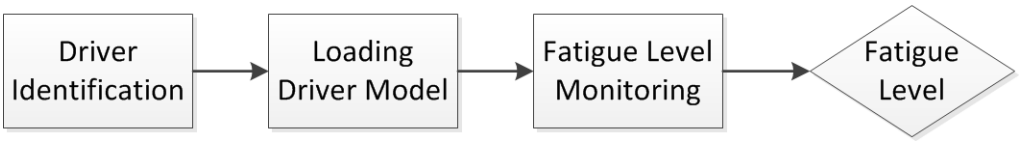


Figure 5.2: Monitoring driver fatigue using driver adapted models.

4. Adaptation: the idea, as shown in Fig. 5.2, is to identify the driver use either vision or audio information, then using the right models to assess the driver's fatigue level. Some preliminary experiments were conducted to train different models for each driver. The results were better than using just one model for all drivers. However, a driver identification system needs to be implemented in order to build such a system. For example, many researchers have developed techniques for speaker adaptation [58, 59, 60]. The HTK toolkit often supports speaker adaptation [49].

5.3 Publications

Abdullah Rashwan and Mohamed Kamel and Fakhri Karray, "Car Driver Fatigue Monitoring Using Hidden Markov Models and Bayesian Networks", accepted in 2013 International Conference on Connected Vehicles & Expo (ICCVE 2013), Las Vegas, USA: , 2013.

Appendix A

Detailed Tasks Description

A.1 First Phone Call

Hostess: Pepi's Pizza. How can I help you?

Subject: Hi. I'd like to order a pizza please.

Hostess: Okay. I'll have to transfer your call to our take-out department. One moment please.

Recorded Message: Thank you for calling Pepi's Pizza. All of our operators are busy. Please hold for the next available person.

Take-out Clerk: Thank you for waiting. Is this for take-out or delivery?

Subject: Delivery please.

Take-out Clerk: Can I have your name and address please?

Subject: My name is... My address is ....

Take-out Clerk: Thank you. Is that an apartment or a house?

Subject: It's an apartment. Number ....

Take-out Clerk: Okay. And what would you like to order today?

Subject: I'd like a large pepperoni pizza with extra cheese.

Take-out Clerk: Ok so large pizza isn't it ?

Subject: Yes

Take-out Clerk: With pepperoni and extra cheese

Subject: That's right.

Take-out Clerk: Ok fine. Is there anything else?

Subject: No, that would be all

Take-out Clerk: Anything to drink with that?

Subject: Nothing, thanks

Take-out Clerk: Alright, it's 15 $. How would you like to pay?

Subject: Do you accept credit card?

Take-out Clerk: Credit card? Sure. You pizza should arrive in about thirty minutes. Is that ok?

Subject: Absolutely, thank you very much.

Take-out Clerk: You're welcome. Thanks for calling. Bye.

A.2 Second Phone Call

Subject: Hi, How are you?

Friend: I am good, I was trying to call you all the day, where were you?

Subject: I have been with my sister at Toronto the whole day, why?

Friend: There is a movie night event at the University, do you want to come?

Subject: Sure, I have to cancel first my meeting with Adam.

Friend: Okay, we will be waiting for you.

Subject: Great! See you then.

Friend: See you, Bye.

Subject: Bye.

A.3 First Text Message

“Hey, what about eating some pizzas tonight? I just ordered one, I hope you don’t mind”

A.4 Second Text Message

“I will be there in 30 mins”

A.5 Third Text Message

“I will be there in 30 mins”

A.6 Map research

Locating an intersection on Waterloo-Kitchener map.

Bibliography

[1] World motor vehicle production by country and type. `http://www.oica.net/wp-content/uploads/2013/03/total-production-2012.pdf`, Mar, 2013. 1

[2] John Loehr Barry Jaruzelski and Richard Holman. The global innovation 1000: Navigating the digital future. Oct 2013. 1

[3] Neville A. Stanton and Paul M. Salmon. Human error taxonomies applied to driving: A generic driver error taxonomy and its implications for intelligent transport systems. *Safety Science*, 47(2):227 – 237, 2009. 1, 7

[4] The U.S. National Highway Traffic Safety Administration. Traffic safety facts 2011: A compilation of motor vehicle crash data from the fatality analysis reporting system and general estimates system. 2011. 1, 9

[5] J. Horne and L. Reyner. Vehicle accidents related to sleep: A review. *Occupational and Environmental Medicine*, 56:189–294, 1999. 1, 9

[6] L.M. Bergasa, J. Nuevo, M.A. Sotelo, R. Barea, and M.E. Lopez. Real-time system for monitoring driver vigilance. *Intelligent Transportation Systems, IEEE Transactions on*, 7(1):63–77, 2006. 1, 3, 10

[7] Qiang Ji, P. Lan, and C. Looney. A probabilistic framework for modeling and real-time monitoring human fatigue. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 36(5):862–875, 2006. 1

[8] T. D Orazio, M. Leo, C. Guaragnella, and A. Distante. A visual approach for driver inattention detection. *Pattern Recognition*, 40(8):2341 – 2355, 2007. Part Special Issue on Visual Information Processing. 1, 11

[9] H. P. Greeley, E. Friets, J. P. Wilson, S. Raghavan, J. Picone, and J. Berg. Detecting fatigue from voice using speech recognition. In *Signal Processing and Information Technology, 2006 IEEE International Symposium on*, pages 567–571, 2006. 2, 11, 38

[10] Jarek Krajewski, Udo Trutschel, Martin Golz, David Sommer, and David Edwards. Estimating fatigue from predetermined speech samples transmitted by operator communication systems. *Proceedings of the International Driving Symposium on Human Factors in Driver Assessment, Training and Vehicle Design*, 5:468–474, 2009. 2, 11, 22

[11] L.S. Dhupati, S. Kar, A. Rajaguru, and A. Routray. A novel drowsiness detection scheme based on speech analysis with validation using simultaneous eeg recordings. In *Automation Science and Engineering (CASE), 2010 IEEE Conference on*, pages 917–921, 2010. 2, 11, 63

[12] I. Garcia, S. Bronte, L.M. Bergasa, J. Almazan, and J. Yebes. Vision-based drowsiness detector for real driving conditions. In *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pages 618–623, 2012. 3

[13] A. Vahidi and A. Eskandarian. Research advances in intelligent collision avoidance and adaptive cruise control. *Intelligent Transportation Systems, IEEE Transactions on*, 4(3):143–153, 2003. 7

[14] M. Hoedemaeker. Driving behaviour with acc and the acceptance by individual drivers. In *Intelligent Transportation Systems, 2000. Proceedings. 2000 IEEE*, pages 506–509, 2000. 7

[15] C Visvikis, T L Smith, M Pitcher, and R Smith. Study on lane departure warning and lane change assistant systems. Technical Report PPR 374, Transport Research Laboratory, 2008. 8

[16] Mei Chen, T. Jochem, and D. Pomerleau. Aurora: a vision-based roadway departure warning system. In *Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 1, pages 243–248 vol.1, 1995. 8

[17] J.C. McCall and M.M. Trivedi. Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation. *Intelligent Transportation Systems, IEEE Transactions on*, 7(1):20–37, 2006. 8

[18] W.T. Freeman and E.H. Adelson. The design and use of steerable filters. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 13(9):891–906, 1991. 8

[19] A. de la Escalera, L.E. Moreno, M.A. Salichs, and J.M. Armingol. Road traffic sign detection and classification. *Industrial Electronics, IEEE Transactions on*, 44(6):848–859, 1997. 8

[20] M.L. Eichner and T.P. Breckon. Integrated speed limit detection and recognition from real-time video. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 626–631, 2008. 9

[21] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981. 9

[22] Vikas Yadav, Deepa Makhija, Shruti Savant, Nilesh Dodani, and Dhananjay K. Theckedath. Article: Driver drowsiness detection system. *IJCA Proceedings on International Conference on Intuitive Systems and Solutions 2012*, ICISS(1):11–13, August 2012. Published by Foundation of Computer Science, New York, USA. 9

[23] Qiong Wang, Jingyu Yang, Mingwu Ren, and Yujie Zheng. Driver fatigue detection: A survey. In *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on*, volume 2, pages 8587–8591, 2006. 10

[24] Paul Smith, Student Member, Mubarak Shah, and Niels Da Vitoria Lobo. Determining driver visual attention with one camera. *IEEE Trans. on Intelligent Transportation Systems*, 4:2003, 2003. 10

[25] Qiang Ji and Xiaojie Yang. Real-time eye, gaze, and face pose tracking for monitoring driver vigilance. *Real-Time Imaging*, 8(5):357–377, October 2002. 11

[26] Qiang Ji, P. Lan, and C. Looney. A probabilistic framework for modeling and real-time monitoring human fatigue. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 36(5):862–875, 2006. 11

[27] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Mach. Learn.*, 20(3):273–297, September 1995. 12

[28] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. 13

[29] Todd Munson, Jason Sarich, Stefan Wild, Steven Benson, and Lois Curfman McInnes. Tao 2.0 users manual. Technical Report ANL/MCS-TM-322, Mathematics and Computer Science Division, Argonne National Laboratory, 2012. http://www.mcs.anl.gov/tao. 13

[30] Dimitri P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods (Optimization and Neural Computation Series)*. Athena Scientific, 1 edition, 1996. 14

[31] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257 –286, Feb 1989. 19

[32] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. 20

[33] David L Hall and James Llinas. An introduction to multisensor data fusion. *Proceedings of the IEEE*, 85(1):6–23, 1997. 20

[34] Federico Castanedo. A review of data fusion techniques. *The Scientific World Journal*, 2013:19, 2013. 20

[35] B.V. Dasarathy. Sensor fusion potential exploitation-innovative architectures and illustrative applications. *Proceedings of the IEEE*, 85(1):24–38, 1997. 20, 22

[36] Shifei Ding, Hong Zhu, Weikuan Jia, and Chunyang Su. A survey on feature extraction for pattern recognition. *Artificial Intelligence Review*, 37(3):169–180, 2012. 22

[37] S.R. Safavian and D. Landgrebe. A survey of decision tree classifier methodology. *Systems, Man and Cybernetics, IEEE Transactions on*, 21(3):660–674, 1991. 22

[38] *Multiple Classifier Systems - 10th International Workshop, MCS 2011, Naples, Italy, June 15-17, 2011. Proceedings*, volume 6713 of *Lecture Notes in Computer Science*. Springer, 2011. 23

[39] Peter Jackson. *Introduction to Expert Systems*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 3rd edition, 1998. 23

[40] Irina Perfilieva and Jiří Močkoř. *Mathematical principles of fuzzy logic*. Springer, 1999. 23

[41] Stuart Jonathan Russell, Peter Norvig, John F Canny, Jitendra M Malik, and Douglas D Edwards. *Artificial intelligence: a modern approach*, volume 74. Prentice hall Englewood Cliffs, 1995. 23

[42] Nevin Zhang and David Poole. A simple approach to Bayesian network computations. In *Proceedings of the Tenth Canadian Conference on Artificial Intelligence*, pages 171–178, 1994. 24, 48, 58

[43] Celine Craye. A framework for context-aware driver status assessment systems. Master's thesis, University of Waterloo, 2013. 25, 65

[44] Men break more traffic laws, drive more dangerously than women, concludes qpc study. http://www.qualityplanning.com/news/2008-articles/men-break-more-traffic-laws,-drive-more-dangerously-than-women,-concludes-qpc-study-.aspx, Nov, 2008. 28

[45] K. Torkkola, N. Massey, and C. Wood. Driver inattention detection through intelligent analysis of readily available sensors. In *Intelligent Transportation Systems, 2004. Proceedings. The 7th International IEEE Conference on*, pages 326–331, 2004. 30

[46] J. Ramirez, J. M. Gorriz, and J. C. Segura. *Voice Activity Detection. Fundamentals and Speech Recognition System Robustness*. InTech, June 2007. 34

[47] Jarek Krajewski, Udo Trutschel, Martin Golz, David Sommer, and David Edwards. Estimating fatigue from predetermined speech samples transmitted by operator communication systems. *Proceedings of the International Driving Symposium on Human Factors in Driver Assessment, Training and Vehicle Design*, 5:468–474, 2009. 38

[48] L.S. Dhupati, S. Kar, A. Rajaguru, and A. Routray. A novel drowsiness detection scheme based on speech analysis with validation using simultaneous eeg recordings. In *Automation Science and Engineering (CASE), 2010 IEEE Conference on*, pages 917–921, 2010. 38

[49] S. J. Young, G. Evermann, M. J. F. Gales, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. C. Woodland. *The HTK Book, version 3.4*. Cambridge University Engineering Department, Cambridge, UK, 2006. 40, 52, 53, 66

[50] Qiang Ji, P. Lan, and C. Looney. A probabilistic framework for modeling and real-time monitoring human fatigue. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 36(5):862–875, 2006. 45

[51] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm. 50

[52] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *Communications, IEEE Transactions on*, 28(1):84 – 95, Jan 1980. 56

[53] Drowsy driving and automobile crashes. http://www.nhtsa.gov/people/injury/drowsy_driving1/drowsy.html, Aug, 2013. 63

[54] Jarek Krajewski, Anton Batliner, and Martin Golz. Acoustic sleepiness detection: Framework and validation of a speech-adapted pattern recognition approach. *Behavior Research Methods*, 41(3):795–804, 2009. 64

[55] C.K. Chui. *An Introduction to Wavelets*. Wavelet analysis and its applications. Academic Press, 1992. 64

[56] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. 64

[57] D. Yu G. Dahl A.Mohamed N. Jaitly A. Senior V. Vanhoucke P. Nguyen T. Sainath G. Hinton, L. Deng and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 29, 2012. 65

[58] V.V. Digalakis, D. Rtischev, and L.G. Neumeyer. Speaker adaptation using constrained estimation of gaussian mixtures. *Speech and Audio Processing, IEEE Transactions on*, 3(5):357–366, 1995. 66

[59] S. Young. A review of large-vocabulary continuous-speech. *Signal Processing Magazine, IEEE*, 13(5):45, 1996. 66

[60] K. Shikano, K.-F. Lee, and R. Reddy. Speaker adaptation through vector quantization. In *Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '86.*, volume 11, pages 2643–2646, 1986. 66