Imperial College London
Department of Electrical and Electronic Engineering

Low-complexity algorithms for automatic detection of sleep stages and events for use in wearable EEG systems

Syed Anas Imtiaz

December 2015

Supervised by Prof. Esther Rodriguez-Villegas

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Electrical and Electronic Engineering of Imperial College
London and the Diploma of Imperial College London

Abstract

Objective: Diagnosis of sleep disorders is an expensive procedure that requires performing a sleep study, known as polysomnography (PSG), in a controlled environment. This study monitors the neural, eye and muscle activity of a patient using electroencephalogram (EEG), electrooculogram (EOG) and electromyogram (EMG) signals which are then scored in to different sleep stages. Home PSG is often cited as an alternative of clinical PSG to make it more accessible, however it still requires patients to use a cumbersome system with multiple recording channels that need to be precisely placed. This thesis proposes a wearable sleep staging system using a single channel of EEG. For realisation of such a system, this thesis presents novel features for REM sleep detection from EEG (normally detected using EMG/EOG), a low-complexity automatic sleep staging algorithm using a single EEG channel and its complete integrated circuit implementation.

Methods: The difference between Spectral Edge Frequencies (SEF) at 95% and 50% in the 8-16 Hz frequency band is shown to have high discriminatory ability for detecting REM sleep stages. This feature, together with other spectral features from single-channel EEG are used with a set of decision trees controlled by a state machine for classification. The hardware for the complete algorithm is designed using low-power techniques and implemented on chip using 0.18μm process node technology.

Results: The use of SEF features from one channel of EEG resulted in 83% of REM sleep epochs being correctly detected. The automatic sleep staging algorithm, based on contextually aware decision trees, resulted in an accuracy of up to 79% on a large dataset. Its hardware implementation, which is also the very first complete circuit-level implementation of any sleep staging algorithm, resulted in an accuracy of 98.7% with great potential for use in fully wearable sleep systems.

Contents

List of Figures

List of Tables

List of Publications

The following papers have been published as part of this work.

Journal papers

- S. A. Imtiaz and E. Rodriguez-Villegas, "A Low Computational Cost Algorithm for REM Sleep Detection Using Single Channel EEG," Annals of Biomedical Engineering, vol. 42, no. 11, pp. 2344–2359, 2014.

- S. A. Imtiaz, A. J. Casson and E. Rodriguez-Villegas, "Compression in Wearable Sensor Nodes: Impacts of Node Topology," IEEE Transactions on Biomedical Engineering, vol. 61, no. 4, pp. 1080–1090, 2014.

- S. A. Imtiaz, L. Logesparan and E. Rodriguez-Villegas, "Performance-Power Consumption Tradeoff in Wearable Epilepsy Monitoring Systems," IEEE Journal of Biomedical and Health Informatics, vol. 19, no. 3, pp. 1019–1028, 2014.

- G. Chen, S. A. Imtiaz, E. AguilarPelaez and E. Rodriguez-Villegas, "Algorithm for heart rate extraction in a novel wearable acoustic sensor," IET Healthcare Technology Letters, vol. 2, no. 1, pp. 28–33, 2015.

Peer reviewed conference papers

- S. A. Imtiaz and E. Rodriguez-Villegas, "Automatic sleep staging using state machine–controlled decision trees," in proceedings of the 37^{th} international conference of the IEEE Engineering in Medicine and Biology Society, Milan, August 2015.

- S. A. Imtiaz and E. Rodriguez-Villegas, "An open–source toolbox for standardized use of PhysioNet Sleep EDF Expanded database," in proceedings of the 37^{th} international conference of the IEEE Engineering in Medicine and Biology Society, Milan, August 2015.

- S. A. Imtiaz and E. Rodriguez-Villegas, "Recommendations for performance assessment of automatic sleep staging algorithms," in proceedings of the 36^{th} international conference of the IEEE Engineering in Medicine and Biology Society, Chicago, August 2014, pp. 5044–5047.

- S. A. Imtiaz and E. Rodriguez-Villegas, "Evaluating the use of line length for automatic sleep spindle detection," in proceedings of the 36^{th} international conference of the IEEE Engineering in Medicine and Biology Society, Chicago, August 2014, pp. 5024–5027.

- S. A. Imtiaz, S. Saremi-Yarahmadi and E. Rodriguez-Villegas, "Automatic detection of sleep spindles using Teager energy and spectral edge frequency," in proceedings of the IEEE Biomedical Circuits and Systems Conference (BioCAS), Rotterdam, October 2013, pp. 262–265.

- L. Logesparan, A. J. Casson, S. A. Imtiaz and E. Rodriguez-Villegas, "Discriminating between best performing features for seizure detection and data selection," in proceedings of the 35^{th} international conference of the IEEE Engineering in Medicine and Biology Society, Osaka, July 2013, pp. 1692–1695.

- L. Logesparan, S. A. Imtiaz, A. J. Casson and E. Rodriguez-Villegas, "A 1.8mW 12-channel wireless seizure detector for miniaturized portable EEG systems," in proceedings of the 9^{th} international conference on ubiquitous healthcare (u-Healthcare 2012), GyeongJu, October 2012.

Acknowledgements

I would like to extend my sincerest gratitude to my supervisor, Prof. Esther Rodriguez-Villegas, for giving me the opportunity and motivating me to pursue this research. Esther has not only been an amazing supervisor but also a great mentor and friend. I am thankful to her for encouraging my research, giving me the freedom and space to pursue my ideas and allowing me to grow as a researcher. I also appreciate all her contributions of time, ideas, and funding to make my PhD experience productive and stimulating.

I consider myself lucky to be working with a team of very talented individuals in the lab. I am thankful to everyone including Sorsby, Saam, Zhou, George, Majd, Stuart, Ruchir, Mohammad and Valentin for helping me with everything I ever needed, for the countless hours we have spent together near deadlines and for all the quality entertainment during lunch and coffee breaks. It has been a great pleasure working with all of you guys.

I owe special thanks to Alex Casson for being super helpful during the first two years of my PhD assisting me in almost all aspects of my work. I would also like to acknowledge the support of all the former members of the lab including Lojini, Eduardo, Siavash and James who have helped me throughout the course of my research in multiple ways. I am also very grateful to my two best friends at Imperial, Asif and Saad, for all the great times outside the lab. And, of course, I am very thankful to Wiesia for all the administrative support she has provided.

Above all, I am forever indebted to my parents for everything they have done for me. Words can not express how grateful I am to my father and mother for all of the sacrifices that they have made for me. This PhD would never have started without their encouragement. Their unwavering support and trust means the world to me. A mere mention of thanks can never do justice to the level of support they have provided. I would also like to thank my sister and brother-in-law for their support as well as my dearest nieces for cheering me up. I am also thankful to my grandmother, extended family and in-laws for their encouragement, support and prayers.

A very special thanks to my loving, supportive, encouraging and patient wife, Quratulain, who has been by my side throughout this PhD, living every single minute of it and providing unequivocal support.

Finally, and most importantly, I am thankful to God for letting me through all the difficulties and making things happen the way they are today.

Declaration of Originality

I hereby confirm that the work presented here is my own, and that appropriate references have been used to denote the work of others.

Syed Anas Imtiaz

December, 2015.

Copyright Declaration

Terms and Abbreviations

Term	Meaning
AASM	Americal Academy of Sleep Medicine
ADC	Analogue to Digital Converter
AFE	Analogue Front End
alpha	8-13 Hz frequency band
ANN	Artifical Neural Network
AP	Absolute Power
AR	Autoregressive
ASIC	Application Specific Integrated Circuit
AUC	Area Under the Curve
Automatic sleep staging algorithm	An algorithm for detecting sleep stages from PSG recordings
beta	16-30 Hz frequency band
BIS	Bispectral Index
BLE	Bluetooth Low Energy
CNS	Central Nervous System
CR	Compression Ratio
CRC	Cyclic Redundancy Check
CWD	Choi-Williams Distribution
CWT	Continuous Wavelet Transform
DBN	Deep Belief Network
delta	0.5-4 Hz frequency band
DFT	Discrete Fourier Transform
DIT	Decimation In Time
ECG	Electrocardiogram / Electrocardiography
EDF	European Data Format
EEG	Electroencephalogram / Electroencephalography
EMD	Empirical Mode Decomposition
EMG	Electromyogram / Eletromyography
EOG	Electrooculogram / Electrooculography
ExG	Refers to either of ECG/EEG/EMG/EOG combination
FFT	Fast Fourier Transform

Term	Meaning
FN	False Negatives
FNN	Feedforward Neural Network
FP	False Positives
FSM	Finite State Machine
GMM	Gaussian Mixture Model
GP	General Physician
HHT	Hilbert Huang Transform
HMM	Hidden Markov Model
HPSG	Home Polysomnography
hypnogram	Result of polysomnography showing different stages of sleep
IC	Integrated Circuit
K-complex	An isolated sharp negative wave followed by a postive component
KLR	Kernel Logistic Regression
kNN	k-Nearest Neighbour
LAN	Local Area Network
LDA	Linear Discriminant Analysis
LOOCV	Leave-one-out Cross Validation
MATLAB	A numerical software for signal analysis
MDD	Major Depressive Disorder
MODWT	Maximum Overlap Discrete Wavelet Transform
mRMR	Minimum Redundancy Maximum Relevance
MSE	Multi-scale Entropy
MSP430	An ultra-low power microcontroller
MT	Movement Time (sleep stage based on R&K rules)
N1	NREM 1 stage of sleep according to AASM rules
N2	NREM 2 stage of sleep according to AASM rules
N3	NREM 3 stage of sleep according to AASM rules
non-REM	Any sleep stage apart from REM
NREM	Non-REM stages of sleep (excluding Wake and Movement)
OSA	Obstructive Sleep Apnoea
PC	Personal Computer
PCA	Principle Component Analysis
PLMS	Periodic Limb Movements in Sleep
PNN	Probabilistic Neural Network
PPV	Positive Predictive Value
PRD	Percentage RMS Difference
PSG	Polysomnography

Term	Meaning
R&K	Rechtschaffen and Kales
RAM	Random Access Memory
RBD	REM Behaviour Disorder
REM / R	Rapid Eye Movement (a stage of sleep)
RMS	Root Mean Square
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
ROM	Read Only Memory
RP	Relative Power
RTL	Register-transfer Level
RVM	Relevance Vector Machine
S1	Stage 1 of sleep according to R&K rules
S2	Stage 2 of sleep according to R&K rules
S3	Stage 3 of sleep according to R&K rules
S4	Stage 4 of sleep according to R&K rules
SaaS	Software as a Service
SAHS	Sleep Apnoea Hypopnoea Syndrome
SEF	Spectral Edge Frequency
SEF50	Spectral Edge Frequency at 50%
SEF95	Spectral Edge Frequency at 95%
SEFd	Difference between SEF95 and SEF50
SEFxx	Spectral Edge Frequency at XX%
sigma	11-16 Hz frequency band
sleep spindle	A burst of 11-16 Hz wave during S2/N2 stage of sleep
STFT	Short-time Fourier Transform
SVM	Support Vector Machine
SVM-RFE	Support Vector Mahine - Recursive Feature Extraction
SWS	Slow Wave Sleep
TEO	Teager Energy Operator
theta	4-8 Hz frequency band
TN	True Negatives
TP	True Positives
VHDL	Hardware description language for digital design
W	Wake stage of sleep
Wearable System	A small, portable, easy-to-use system with low power consumption
WPT	Wavelet Packet Transform

1 Introduction

1.1 Overview

Diagnosis and treatment of various medical conditions has traditionally been performed in a controlled clinical setting under strict supervision of physicians. Typically, this requires multiple visits to a clinic as well as possible admissions for prolonged periods in cases where continuous monitoring of the patient is desired. The soaring costs of healthcare together with rising human population is putting a lot of strain on this healthcare model resulting in longer waiting times to be seen by a specialist.

The recent advances in biomedical and healthcare technology have shown great potential in shifting some of the clinical monitoring and diagnosis to the patient's home. This would not only relieve some burden from the hospital resources but also provide medical care to a larger cohort of the population. Realisation of this idea, however, requires medical equipment and devices that are small in size, lower in cost than their traditional counterparts, safe, easy and comfortable to use as well as provide reliable results. Consequently, there has been a great deal of research and commercial focus on miniaturisation of devices to design wearable health systems. An ideal wearable device would not only monitor the patient's physiological signals but also be intelligent enough to assist the physician in decision-making by looking for common patterns and interpreting the results for speedy diagnosis. It cannot be stressed enough that a wearable medical device is not a substitute for physicians but rather a tool to complement them by saving their, as well as, patients' time.

Wearable devices have the potential to revolutionise healthcare services at the hospitals as well as homes. For hospital use, the presence of trained clinicians does not require patients to handle medical devices. However, for home usage, the design of devices should cater for the absence of clinicians and make it extremely easy for patients to handle the device with confidence. A typical such device would entail a wireless non-invasive sensor that transmits data to a small hand-held unit. This unit, which would incorporate some sort of processing, can then analyse the data and then store or transmit it to a clinic. Furthermore, such a device should be able to integrate seamlessly in the normal lifestyle of patients without causing any hindrance or requiring any significant change. The ubiquity of smartphones with wireless connectivity and powerful processors provides a useful platform for integration with wearable sensors reducing development costs of wearable medical devices by lending some of their powerful features to these devices.

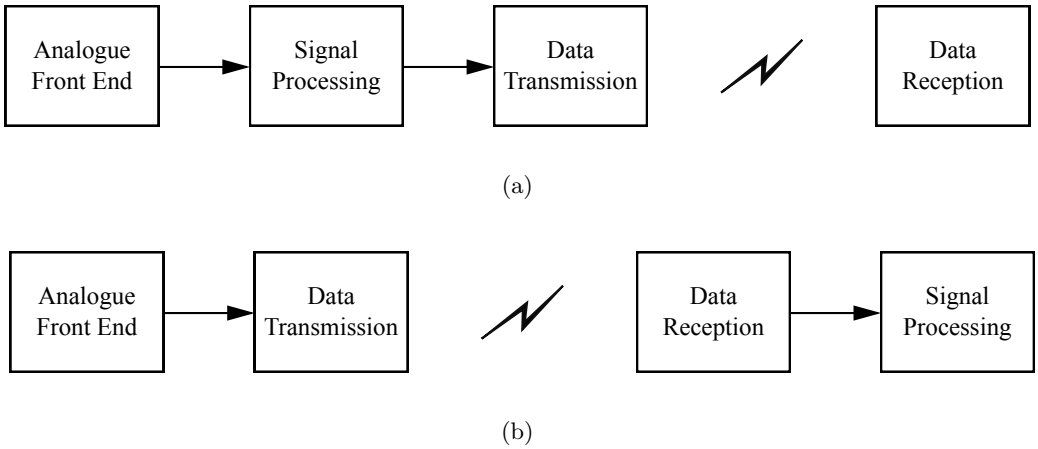


Figure 1.1: *The two system design approaches for a wireless wearable device: (a) with signal processing at the sensor node; (b) with signal processing at the receiver end.*

The inherent nature of truly wearable devices that include tiny sensors, wireless transmission and long hours of operation makes them very challenging to design. Their development involves work in a resource-constrained environment with extreme limitations on available power budget, device size and computational resources depending on their potential target application. On a system level, the various areas of development include sensor design, human factors, data transmission protocol, signal conditioning and processing, data integrity and security.

Figure 1.1 shows two possible architectures of a wireless wearable device at a simplistic level. It includes an analogue front end for data acquisition and signal conditioning, a signal processing block, wireless data transmitter and a receiver. The only difference between the two architectures is the placement of the signal processing block in the data pipeline. Signal processing is a key part of the design that adds intelligence to the sensor or device to aid in decision-making thereby reducing a physician's time spent per case, consequently reducing costs and making healthcare accessible to more people.

The first approach in Figure 1.1(a) allows for signals to be processed on the sensor prior to transmission. However, with device size being small and having limited power budget, signal processing techniques ought to be of low-complexity without compromising on end results. These restrictions make the sensor design more challenging if this approach is used. However, this architecture has the advantage of reduced data transmission bandwidth. With the second architecture shown in Figure 1.1(b), most or all of signal processing is performed at the receiver end. In this case all the raw data is transmitted to the receiver (e.g. a tablet or smartphone) where complex signal processing techniques can be applied with the availability of more power and computing resources. However, the transmission stage consumes more power with this approach due to a higher data rate.

This thesis focuses on the development of low power signal processing algorithms that identify diagnostically useful patterns observed on electroencephalogram (EEG) signals recorded from the brain during sleep. It primarily focuses on algorithms for the first architecture but also explores the design considerations required for the transmission of raw or compressed EEG signals in the second approach. The design of algorithms, their performances and hardware implementation are discussed in detail.

1.2 Thesis structure

This thesis is organised into five major chapters with four additional appendices. The main contributions of each are summarised below.

Chapter 2 - Sleep: What is it, why is it important and how is it monitored?

In this chapter, the different stages of human sleep are introduced. Several health conditions and disorders that can affect sleep are briefly discussed followed by their financial impact on the economy. The current clinical practice in the diagnosis of sleep disorders are discussed and the limitations and drawbacks of this are explained. The current practice involves analysis of a patient's sleep by monitoring his neural, eye movement and muscle activities. The signals acquired from these activities are visually analysed and classified into one of the different stages of sleep. It is argued that this process is costly and time consuming for both doctors and patients and consequently puts a strain on the healthcare system resulting in fewer patients getting the required treatment. A comprehensive literature review is presented in this chapter to explain how different researchers have proposed several methods to automate the classification of sleep signals. Further, a review of commercial systems and their performances is also covered. Finally, a wearable system for sleep monitoring is proposed that can overcome the limitations of existing systems. The design and implementation challenges associated with this proposed system are then discussed in detail.

Chapter 3 - Performance assessment of automatic sleep staging algorithms

A number of automatic sleep staging algorithms have been proposed in the last four decades. However, comparison between them is challenging because of varying metrics used to characterise their performances. The use of different databases or sections of data from same databases complicates this further since the performance of certain algorithms are dependent on the source of signals used. The prevalence of two different rules of sleep classification also adds to this problem.

In this chapter, a set of recommendations is proposed for uniform performance assessment of sleep staging algorithms. It is argued that public databases should be used to characterise the performance of an algorithm. Several public databases are briefly discussed and a set of guidelines is presented to extract data from these databases so that the same sections of data are used across the board. This is followed by the minimum set of metrics that should be calculated to highlight the detection accuracy of an algorithm during the different stages of sleep. A sleep staging algorithm is then presented as an example to show how these recommendations are to be used. Finally, with this algorithm three different cases are used to illustrate how seemingly similar results using different databases can be totally misleading. These cases confirm the need to use standardised performance metrics for sleep staging algorithms.

Chapter 4 - REM sleep detection using single channel EEG

The push towards low-power and wearable sleep systems requires using the minimum number of recording channels to enhance battery life, keep processing load small and be more comfortable for the user. Since most sleep stages can be identified using EEG traces, enormous power savings could be achieved by using a single channel of EEG. However, detection of Rapid Eye Movement (REM) sleep - an important sleep stage - from one channel EEG is challenging due to its electroencephalographic similarities with N1 and Wake stages. This limitation is one of the bottlenecks in the realisation of the system proposed in this thesis.

This chapter investigates a novel feature in sleep EEG that demonstrates high discriminatory ability for detecting REM phases. The discriminatory ability of this feature is quantified and studied in different EEG channels. This feature, which is based on spectral edge frequency (SEF) in the 8–16 Hz frequency band, is then used together with the absolute power and the relative power of the signal, to develop a simple REM detection algorithm. The performance of the proposed algorithm is evaluated with overnight single channel EEG recordings from different sources. Finally, the results are compared against the performance of other algorithms that have been evaluated on the same database.

Chapter 5 - Automatic sleep staging using state machine-controlled decision trees

In this chapter a novel sleep staging algorithm is presented to work within the constraints of a wearable system discussed in Chapter 2. This algorithm uses several frequency domain features extracted from one frontal EEG channel including the features investigated in Chapter 4. These features are then classified using a set of state machine-controlled decision trees.

In a conventional decision tree there is only one entry point in the tree resulting in several redundant nodes of analysis during different sleep stages. This results in a long

path from the entry point to the final classified result. The classifier in this algorithm has been specifically designed to use simpler features and reduce the number of analysis nodes required, making it suitable for use in a resource-constrained wearable sleep staging system. It is essentially a set of contextually aware decision trees that are activated based on the current stage of sleep. The design of each decision tree and the features required for them are discussed in detail. Finally, the performance of this algorithm is demonstrated using two public databases.

Chapter 6 - Integrated circuit design and implementation of an automatic sleep staging algorithm

This chapter presents the first complete hardware implementation of a sleep staging algorithm. The algorithm, proposed in Chapter 5, is implemented as a digital application specific integrated circuit (ASIC) in 0.18μm technology. It has four major blocks: Input controller, Fourier transform, Feature calculation and Classifier. The design of each block is covered in complete detail including its constituent modules. Several design techniques have been used to ensure there is less logic required for arithmetic computations. Further, the choices made to optimise area, meet timing and reduce power consumption of the system are also explained.

Later in this chapter, the algorithm performance after its hardware implementation is compared against the reference algorithm by simulating the hardware. It is then synthesised to produce a gate-level netlist which is formally verified against the hardware description to ensure its equivalence and simulated again for functional equivalence. Finally, the netlist of the design is placed and routed to generate the complete layout of the sleep staging system, which is also verified for logic and functional equivalence.

Appendix A - Databases

It is proposed in Chapter 3 that public databases should be used to evaluate the performance of a sleep staging algorithm. Details about the different freely available databases are presented in this chapter.

Appendix B - An open-source toolbox for standardised use of PhysioNet sleep EDF expanded database

PhysioNet Sleep EDF database has been the most popular source of data used for developing and testing many automatic sleep staging algorithms. However, the recordings from this database have been used in an inconsistent fashion. This includes, for example, arbitrary selection of start and end times from long term recordings, data-hypnogram mismatches and different performance metrics. All these differences result in different data sections and performance metrics being used by researchers thereby making any direct

comparison between algorithms very difficult. Recently, a superset of this database has been made available on PhysioNet, known as the Sleep EDF Expanded Database which includes 61 recordings. This provides an opportunity to standardise the way in which signals from this database should be used. With this goal in mind, this chapter presents a toolbox for automatically downloading and extracting recordings from the Sleep EDF Expanded database and converting them to a suitable format for use in MATLAB. This toolbox contains functions for selecting appropriate data for sleep analysis (based on the recommendations in Chapter 3), hypnogram conversion and computation of performance metrics. Its use makes it simpler to start using the new sleep database and also provides a foundation for much needed standardisation in this research field.

Appendix C - Automatic detection of sleep spindles

A sleep spindle is an important transient observed on the EEG during a certain stage of sleep. It is highly useful for the identification of this particular sleep stage and is also a subject of research to understand its importance. This chapter presents a literature review of various automatic sleep spindle detection algorithms and then proposes two algorithms for automatic identification of spindles from a single channel of EEG. The first algorithm, designed to maximise detection accuracy, uses Teager Energy Operator (TEO) and Spectral Edge Frequency (SEF) in the frequency band of interest. The second algorithm is designed to be of low complexity such that it is suitable for use in wearable devices. For this, line length of the signal is used as the characteristic feature for spindle detection. It is also implemented on a MSP430 microcontroller to demonstrate its low power consumption. Finally, the performances of the two algorithms are compared against the methods reviewed earlier.

Appendix D - Compression in wearable sensor nodes for data transmission and storage

An alternative approach for designing a wearable sleep scoring system involves performing signal analysis at the receiver end and either transmitting raw EEG data from the sensor or storing it on the sensor node itself using flash memory. However, to save transmission bandwidth and lower the power consumption it is essential to compress this data.

This chapter presents a low power MSP430-based compressive sensing implementation for providing such compression, focusing particularly on the impact of the sensor node architecture on the compression performance. Compression power performance is compared for four different sensor nodes incorporating different strategies for wireless transmission/on-sensor-node local storage of data.

2 Sleep: What is it, why is it important and how is it monitored?

2.1 Introduction

Sleep is a state of unconsciousness from which a person can be aroused [1]. It is considered a necessity of life for humans and animals alike and is essential to their physical and emotional wellbeing. It is a natural state of reduced alertness during which the response of human body to external stimuli decreases. The complexities of sleep are not well known but it is understood to be an active state during which there is an increase in the rate of anabolism. The importance of healthy sleep can be characterised by the fact that its deprivation leads to reduced physical performance, mental awareness and body temperature as well as a decrease in immune system function and an increase in heart rate variability [2]. Sleep accounts for approximately one-third of our lifetime and poor sleep leads to an overall decrease in the quality of life.

Stages of sleep

Human sleep is broadly classified into two distinct oscillatory phases based on the eye movements during sleep. These are known as Rapid Eye Movement (REM) and Non-Rapid Eye Movement (NREM). The NREM phase is further divided into different stages. According to Rechtschaffen and Kales (R&K) rules for classification of sleep stages [3], published in 1968, NREM is further classified into Stages 1, 2, 3 and 4 known as S1, S2, S3 and S4 respectively. In 2007 the American Academy of Sleep Medicine (AASM) published a set of revised guidelines [4] based on which NREM is subdivided into N1, N2 and N3 stages. Both R&K and AASM classifications include Wake (W) and REM (R) stages while the former also includes an additional Movement Time (MT) stage. The complete cycle from the start of N1 to the end of REM stage takes about 90 minutes. The cycle then repeats depending on the duration of sleep, typically 4 to 5 times during a full night's sleep. The transition from one sleep state to another and the duration of each state gives useful insight when performing sleep analysis to diagnose sleep disorders.

2.1.1 Sleep disorders

Overview

It is estimated that more than 3.5 million people in the United Kingdom [5] and more than 70 million people in United States suffer from some sort of sleep disorder [6], having a huge financial impact on the economy stemming from expensive treatments, reduced productivity, road accidents and many other areas that involve alertness and quick judgement. Common sleep disorders include obstructive sleep apnea (OSA) causing disruption of sleep by interruption of breathing; REM Sleep Behaviour Disorder (RBD) involving abnormal behaviour during REM phase of sleep and Circadian Rhythm Disorders related to jet lag, shift work, etc. These may manifest in the form of sleep deprivation, disruptive sleep, excessive sleepiness and other sleep-related abnormalities and can be fatal if left untreated.

Diagnosis

Diagnosis of sleep disorders is an expensive procedure that requires performing a sleep study in a controlled environment, usually at specialised sleep clinics, to monitor multiple parameters and physiological signals during sleep. These may include neural activity (EEG), eye movements (EOG), muscle activity (EMG), heart rhythms (ECG) and breathing functions. Together these signals, or a combination of them, are used to perform sleep scoring and identify, or rule out, the presence of multiple sleep disorders.

The sleep studies performed at clinics invariably involve a long waiting time before a patient can be seen. This is because of the limited spaces at such specialised clinics where resources need to be freed up before new patients can be taken in. In the United Kingdom, the waiting period from a GP referral to an actual sleep study can be as long as three years [7]. For this reason, some studies are increasingly being performed at the home of patients with the help of portable devices to do early screening and, in some cases, full testing. The different types of sleep studies are discussed in the next section.

2.1.2 Types of sleep studies

The AASM has classified sleep studies into four types based on where the study is performed and which signals are monitored. The different types and the signals recorded for each are shown in Figure 2.1 and further explained below.

Type 1

This is the gold standard of sleep studies that is performed at sleep clinics in the presence of trained personnel at all times. The signals monitored in this type of study include EEG, EOG, EMG, ECG, airflow, oxygen saturation and more if required.

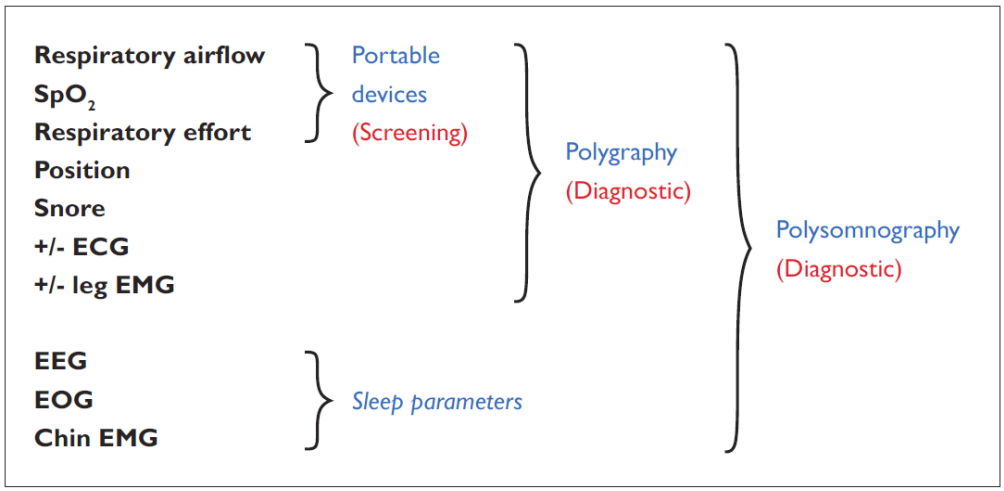


Figure 2.1: *Types of PSG and their application [8]*

Type 2

This type involves recording of all signals as in Type 1 but is performed out of the clinic at a patient's home. It does not require the presence of trained personnel but needs the patient to put on all the electrodes and probes appropriately.

Type 3

In this type, a subset of only four physiological signals are recorded. These include two respiratory signals together with the ECG and oxygen saturation using a portable device at home without the need for trained personnel.

Type 4

This is the simplest level of sleep study that is useful for early screening that may lead to a complete sleep study later. In this type, a maximum of two signals are recorded (airflow and/or oxygen saturation) using a portable device at home without the need for the presence of trained personnel.

Discussion

Type 1 sleep study, which involves recording maximum information and is the clinical gold standard is known as Polysomnography (PSG). Recently, several commercial portable devices have been available that make it possible to perform PSG at home (Type 2). This type of study is commonly referred to as Home PSG (HPSG). Of the four types of sleep studies only the first two types (1 and 2) record the EEG, EOG and EMG signals that are needed to determine the different stages and parameters of sleep accurately.

This thesis focuses on scoring of sleep stages for both Type 1 and Type 2 studies. The next section discusses PSG in detail and looks at the characteristics of signals required for sleep scoring.

2.2 Polysomnography

Polysomnography is usually performed at sleep clinics in the presence of trained technicians, typically using four to six EEG electrodes, two EOG electrodes, four EMG electrodes, two ECG electrodes and additional sensors such as pulse oximeter, sound probes, etc. All these sensors monitor various body functions and are connected to a central processing device where the signals are recorded and stored for future analysis. The types and quantities of sensors used may differ depending on the sleep disorder that the physician is attempting to diagnose. Figure 2.2 shows an illustration of a patient with all sensors attached for PSG.

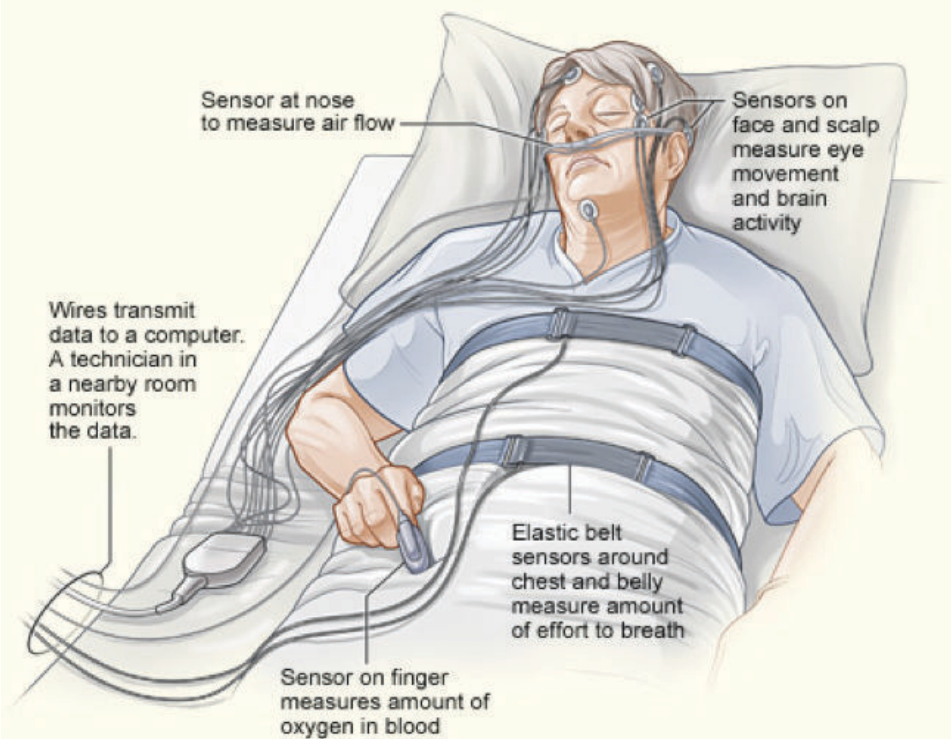


Figure 2.2: *Illustration of a patient with sensors attached for PSG [9]*

Recordings from all sensors are grouped together and analysed by trained technicians in blocks of *30-second epochs*. An example of PSG analysis waveforms is shown in Figure 2.3. The technicians use visual analysis of key waveform patterns and signal characteristics such as frequency content information and peak-to-peak voltage to mark the 30-second epochs. The sleep stages are determined by analysing the EEG, EOG and EMG signals. These are briefly discussed in the remainder of this section.

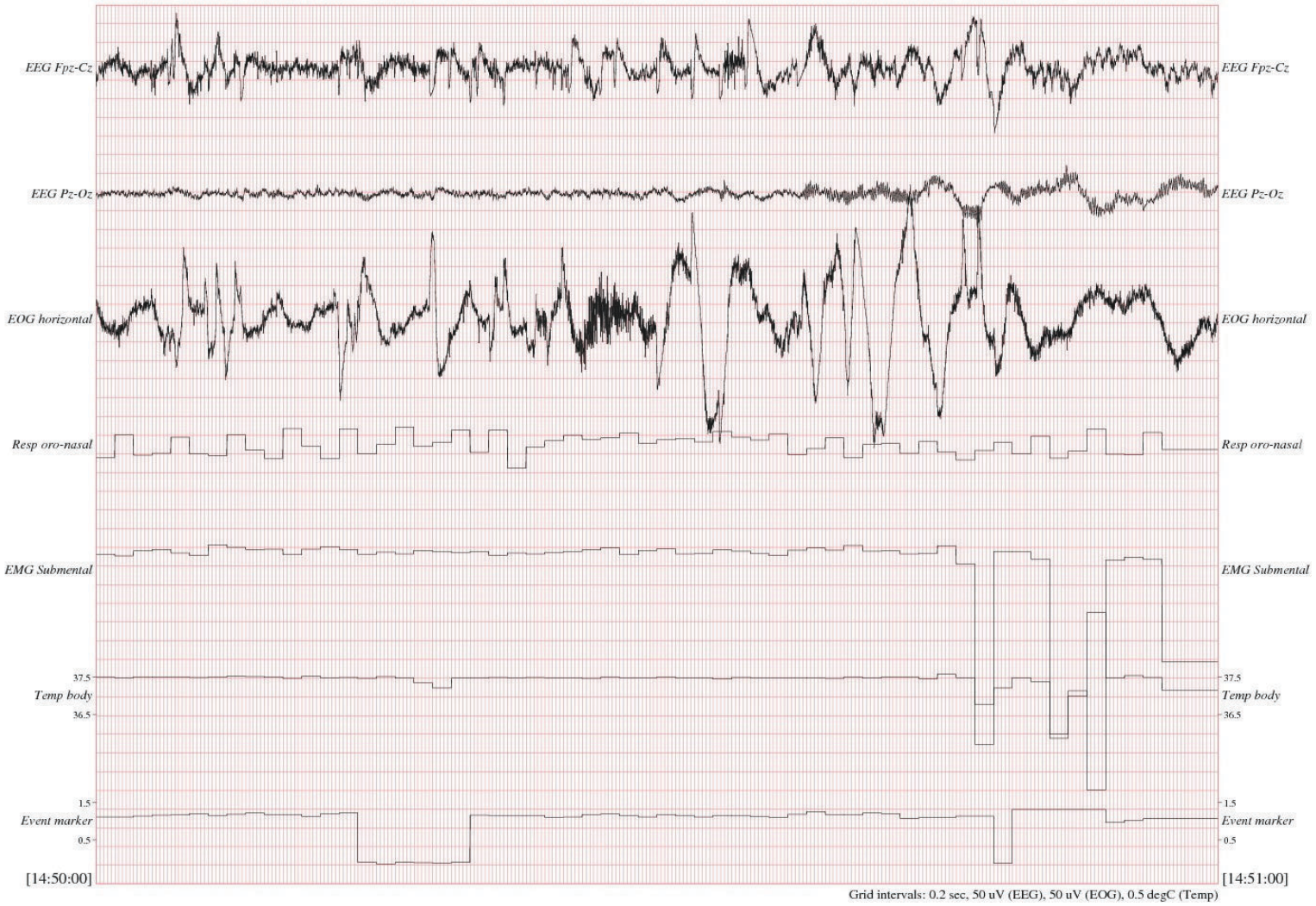


Figure 2.3: *A typical set of PSG analysis waveforms [10]*

2.2.1 EEG

Electroencephalography is a method of recording the neural activity of the brain. The electrical signal recorded is known as an electroencephalogram or EEG. To record EEG, electrodes are placed on the scalp with a front end system involving differential amplifiers and filters. EEG provides vital information for diagnosis of numerous disorders and the overall health of brain.

Configuration

Electrodes to record EEG signals are placed on the scalp according to the standard International 10-20 system as shown in Figure 2.4. For sleep staging, four to six electrodes are used to acquire signals from different locations on the scalp.

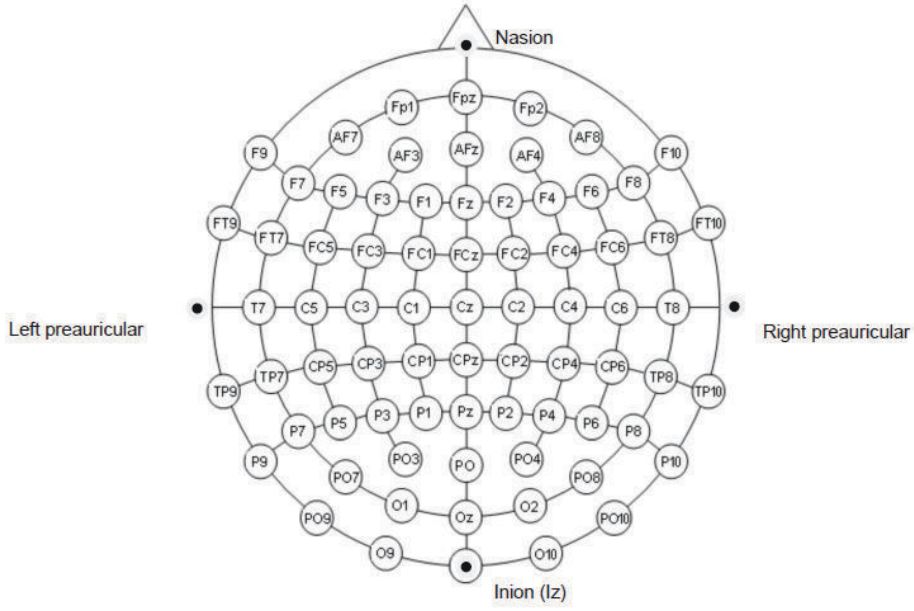


Figure 2.4: *EEG electrode placement according to 10-20 system [4]*

2.2.2 EOG

An electrooculogram (EOG) is a signal that provides information about the eye movement activity. The technique used to produce this signal is called electrooculography. It involves placement of electrodes near the eyes and recording the electrical activity produced due to their movement. Electrode placement configurations to record EOG signals from left and right eye are shown in Figure 2.5. During the REM stage of sleep these signals become very important since they indicate major eye movement activity which is mostly absent during the NREM stages. The eye movements are also observed during Wake stages and tend to slow down with the onset of sleep.

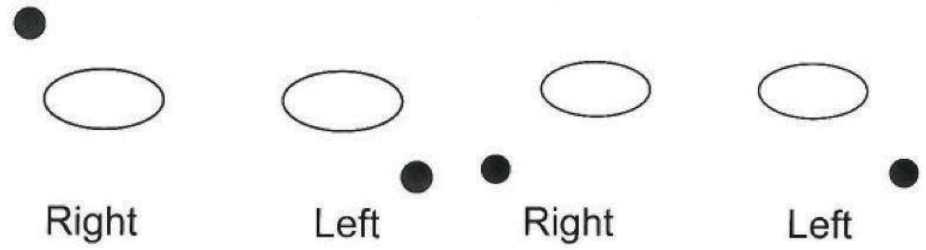


Figure 2.5: *Electrode placement configurations for EOG recording [4]*

2.2.3 EMG

An electromyogram (EMG) is a signal that provides information about the muscle movement activity. A full PSG study involves recording the leg movement and chin EMG. The former is useful for diagnosis of periodic limb movement disorder while the latter helps in identifying Wake and REM stages and differentiating them from other sleep stages with similar EEG characteristics.

2.3 Sleep staging rules

The EEG, EOG and EMG signals are segmented into epochs of 30 seconds. These epochs are visually analysed based on a set of rules and consequently assigned a sleep stage. The first set of rules for scoring of sleep were published in 1968 by Rechtschaffen and Kales, and are commonly known as R&K rules [3]. These rules were revised in 2007 and some changes were recommended to address and overcome some of their inherent limitations. The updated set is known as the AASM (American Academy of Sleep Medicine) rules [4]. However, despite the revised rules, R&K rules are still widely used while the AASM rules are gradually being adopted. The two sets of rules of briefly explained below.

2.3.1 R&K rules of sleep staging

Wake

An epoch is classified as Wake (W) if the EEG derivations show the presence of alpha (8-13 Hz) activity together with low voltage and mixed frequency activity.

S1

Stage 1 is characterised by the presence of low voltage and mixed frequency activity on the EEG but without the presence of eye movements on the EOG.

S2

Stage 2 is identified by the presence of sleep transients known as *sleep spindles* and *k-complexes* observed on the EEG signals. Spindles are defined as a burst of 12-14 Hz waves with a duration of at least 0.5 seconds. A K-complex is defined as an isolated sharp negative wave followed by a positive component and duration of more than 0.5 seconds. The presence of either of these two transients and a lack of slow wave activity is used to classify an epoch as Stage 2 of sleep.

S3

An epoch is marked as Stage 3 when its EEG content consists of between 20-50% of high amplitude delta (0.5-2 Hz) waves. The amplitude of these slow waves are more than 75 μV.

S4

Stage 4 is similar to S3 except that the EEG contains delta waves for more than 50% of the epoch duration. Together S3 and S4 stages are are also known as slow wave sleep (SWS).

REM

REM sleep is identified with the presence of low voltage and mixed frequency EEG similar to what is seen during Stage 1. It is differentiated with the presence of eye movements on the EOG.

Movement Time

An epoch is classified as Movement Time (MT) when the EEG and EOG signals are obscured for more than half of the epoch duration.

2.3.2 AASM rules of sleep staging

Wake

An epoch is classified as W when more than 50% of it visibly consists of alpha (8-13 Hz) activity. If there is a large amount of artefact in an epoch (suggesting significant movement of body) it will be marked as W if the alpha rhythms are visible, otherwise it is scored as the stage that follows this epoch.

N1

If an epoch consists of less than 50% of alpha (8-13 Hz) rhythms and more than 50% of theta (4-7 Hz) rhythms, it will marked as stage 1 of NREM sleep, known as N1.

N2

This stage of sleep is identified by the presence of either or both of two distinct features of sleep: *sleep spindles* and *k-complexes*. K-complex is defined as *"a well delineated negative sharp wave immediately followed by a positive component standing out from the background EEG with total duration ≥ 0.5 seconds"* [4]. Sleep spindle is defined as *"a train of distinct waves with frequency 11-16 Hz (most commonly 12-14 Hz) with a duration ≥ 0.5 seconds"* [4]. N2 stage can be marked with the appearance of sleep spindles and/or K-complexes.

N3

This stage is scored when more than 20% of an epoch consists of waves in the range of 0.5-2 Hz, with peak-to-peak amplitude of 75 μV or more as an optionally additional criterion. It is also known as slow wave sleep (SWS).

REM

An epoch is scored as REM when eye movement activity can be observed on data sourced from EOG electrodes while the EEG activity is similar to that observed during N1 stage. Additionally, chin EMG tone falls during this stage, which can also be used for classification of REM.

2.3.3 Limitations and drawbacks of PSG

A single night's sleep study is usually not enough for the purpose of diagnosis. This is because the patients undergoing sleep studies may not be able to have sound sleep in a new environment with many sensors attached to them. Further, more monitoring may be required owing to insufficient information obtained from one night of sleep analysis. Therefore, sleep clinics are increasingly giving patients more time to acclimatise in the alien conditions to get comfortable with the cumbersome load of sensors attached to their body. Since the PSG analysis performed at specialised sleep centres require the patients to come out of their habitat, the consequential altered sleeping environment can also affect the resultant sleeping pattern and hence add bias to the diagnosis. The whole process of a patient having to come out of his home environment, spend hours at a specialised centre, requiring an attendant, recording large amounts of data and the time taken to analyse and mark this huge chunk of sleep recordings is not only exhaustive for the patient but also an economically expensive diagnosis method.

The costs associated with PSG coupled with the necessity of clinical admission and long waiting lists [7] limits its usage despite the high prevalence of sleep disorders [5]. Home Polysomnography (HPSG), classified as a type 2 portable monitoring device by AASM [11], offers full unattended PSG at patients' homes. It has recently been shown

to be useful to rule in or out Obstructive Sleep Apnea (OSA), results in better sleep quality of patients and reduces overall costs [12]. However, even if the PSG recording is performed at the patient's home, sleeping with the hardware and electrodes connected is not a very pleasant experience. A typical PSG recording involves the patient being wrapped in a bundle of wires originating from the electrodes on his head, face, chest and legs, and terminating on to the recording/monitoring unit that may be placed in close proximity of the bed. Moreover, this large number of electrodes on the scalp adds physical mass, and therefore causes discomfort during sleep. The mesh of wires and electrodes limits the movement of patients during sleep while any interruption of sleep that requires getting out of bed (e.g. use of toilet) means removal and readjustment of electrodes. Since HPSG requires at least seven channels (including multiple EEG, EOG and EMG channels), the complexity imposed by the requirement of the patient precisely placing these multiple electrodes limits its adoption despite the benefits over clinical PSG.

Following a complete night's recording, the signals are then analysed in blocks of 30-second epochs. Manual analysis and scoring of sleep from PSG traces (acquired in clinic or at home) is a tedious task that can take 2 to 4 hours for scoring data from an entire night sleep [13]. An eight-hour sleep will, therefore, consist of 960 epochs that need to be looked at and classified into different stages of sleep resulting in a *hypnogram*. Although not a factor in modern times, but earlier paper tracing of overnight signals could be as long as 400 meters making the whole process of scoring a challenge [14]. The marking of epochs is performed by trained eyes but adds an element of human error and perception that reduces agreement rate among different scorers. The inter-scorer agreement rate among eight different scorers was found to be between 77-82% [15], [16].

The core issues and limitations associated with both clinical and home PSGs can be summarised as follows:

- PSG at sleep clinics is expensive.
- HPSG, while saving cost and allowing patient the comfort of home, still requires the patient to meticulously attach a number of electrodes in different positions.
- The sheer number of sensors makes sleep uncomfortable during PSG.
- It is laborious and time consuming to manually score the epochs.
- Manual scoring is error-prone and can lead to disagreement rate of up to 23% between different scorers.

In order to address some of the issues highlighted above, active research is ongoing in the field of sleep analysis primarily to reduce the amount of time (and data) required for analysis; and lessen the human error introduced due to inter-scorer variability. In the various proposed systems and solutions, the idea is to mimic the visual analysis

of epochs using a machine that can identify different features and patterns of signals (EEG/EOG/EMG) and classify them automatically. This idea of automating the marking of epochs is commonly referred to as *automatic sleep staging* or *automatic sleep scoring*. An automatic sleep staging method would help alleviate both inter-rater and intra-rater disagreements, reduce analysis time and lower the cost of PSG tests.

Additionally, PSG systems would greatly benefit from reduction in the number of channels, simplification of user experience and incorporation of automated sleep scoring methods without affecting clinical outcomes. Traditionally, three EEG channels are required in PSG systems together with EOG and EMG channels. Ruehland *et al.* [17] reported no significant differences in sleep scoring reliability when using a single EEG channel, so this number can potentially be reduced to one. It can also be noted from the definition and marking criteria of each stage of sleep that most of the useful information is contained within the EEG subset of the overall PSG test. However, the EOG and EMG channels are still required since identifying REM stage epochs involves observing the chin muscle and eye activity [4]. Several researchers have attempted extraction of features in different domains to see whether sleep classification could be performed using only a single channel of data.

The next section looks at the various algorithms and systems that have been published in the academic literature and others that are commercially available to address the current limitations and bottlenecks in PSG analysis. The focus will be on signal processing methods for automated scoring of sleep stages and the number of channels required.

2.4 Literature review of automatic sleep staging algorithms

Efforts to automate the scoring of sleep stages started almost 50 years ago soon after the first set of rules for scoring sleep were published by Rechtschaffen and Kales in 1968 [3]. These were broadly based on the level of spectral content present in each sleep epoch being analysed. One of the natural approaches taken by researchers is to mimic the sleep staging rules defined by R&K and, later, AASM. To gauge the performance of an automatic sleep scoring method, the results from the automatic classification are compared against a reference hypnogram that is usually scored by a human expert. The percentage of correctly detected epochs by the algorithm in all the stages of sleep is referred to as the overall *accuracy*. The percentage of correctly detected epochs in each stage is generally referred to as the *sensitivity*, *true detection rate* or the *accuracy* of detection in that stage. The rest of this section briefly discusses the various automatic sleep staging methods proposed in academic literature over the years.

The simplest and most natural approach to tackle this problem was to design matched filters for the required frequency bands and use logic circuits to make decisions based on the output of these filters. This approach was taken by Smith *et al.* [18] in 1969. They used matched filters to detect alpha, delta, spindle and k-complex features on small sec-

tions of EEG signals and assigned a sleep score based on logic circuits that were designed to emulate the sleep scoring rules. Their method used data from four EEG channels of a single night recording. This was split into five parts with each part of two hours and resulted in an overall classification accuracy of 86.2%. Another early sleep classification method, proposed by Itil *et al.* [19] around the same time, used baseline cross, first derivative and amplitude of EEG signals to extract 20 features for the classification of each epoch. Larsen and Walter [20] compared two preliminary sleep staging methods based on multiple regression and discriminant analysis. Their main conclusion was that the spectral power features extracted from the EEG signals were enough for classification of sleep stages except for Stage 1 and REM.

Frost [21] presented a rule-based real-time sleep scoring system using EEG and EOG signals. This was implemented using level detectors to indicate different amplitude thresholds and trigger pulses when these threshold conditions were met. This was also one of the earliest circuit level implementation of a sleep scoring algorithm using discrete components. Shortly afterwards, Smith and Karacan [22] also presented a circuit for real-time sleep scoring. This consisted of spindle, alpha, delta and artefact detectors designed using analogue circuits followed by a block of logic circuits to assign a sleep stage based on the output of preceding detectors. Their system required the use of EEG and EOG channels and achieved an accuracy of about 84%.

While these simple approaches showed promising results, researchers also started to explore other probabilistic techniques for classifying sleep signals to obtain better performance. Kumar [23] presented a method using clustering in amplitude subspace and achieved an agreement of 85% in all stages except REM (which was not scored) for two nights of PSG recording. Gath and Bar-On [24] also developed a clustering method for real-time sleep scoring using fuzzy partitioning. Their method was intended for use in sleep laboratories with the help of two EOG, one EEG and one EMG channels, however no results indicating the accuracy of the method were given. Kemp *et al.* [25] then proposed a method where EEG, EOG and EMG signals were modelled using statistical processes and sleep stages were assigned based on their maximum likelihood resulting in 71% agreement with manual scoring. Stanus *et al.* [26] presented two online sleep staging algorithms based on a deterministic and a stochastic classifier and compared the performances of the two classifiers. They used EEG, EOG and EMG signals and showed 75% reliability between the two methods when tested on normal subjects and 70% when those with certain pathological conditions were used. Kuwahara *et al.* [27] proposed a method of identifying characteristic patterns on EEG, EOG and EMG signals using an interval histogram method. The results from this pattern recognition stage was then used with a rule-based classifier resulting in an accuracy of 89% between the proposed method and consensus scoring of three experts. Principe *et al.* [28] used the theory of evidence to automate sleep staging in real-time with EEG, EOG and EMG signals. The

spectral information extracted from different frequency bands were fuzzified and used as partial belief based on which a further set of rules were devised. They used a finite state machine with each state corresponding to a sleep stage and the machine would change state based on the outcome of the heuristic rules. They achieved an average agreement rate of almost 85% after testing with five different records.

Almost fifteen years after Smith *et al.* [22] presented the earliest circuit implementation of their algorithm, Principe and Smith [29] presented the first fully digital implementation of a sleep analysis system to perform identification of alpha, sigma, theta and delta waveforms, sleep spindles as well as eye movement activities. They used digital filters and zero crossing detectors with thresholding - all of which were implemented on a microprocessor (which was a novelty at that time). Jansen and Dawant [30] then presented a feasibility study of using an object-oriented knowledge-based approach for the identification of sleep spindles and k-complexes which could later be extended for use in sleep stage classification.

The early 90s saw the availability of computing facilities in different research labs. This was the start of the digital revolution which meant more complex techniques could be used for signal analysis with the help of computers. The algorithms published from this time onwards saw a rise in the number of features being used. Further features other than those extracted from time and spectral domains saw mainstream adoption while sophisticated rule-based classifiers, decision trees, support vector machines, artificial neural networks, unsupervised clustering and many other classification techniques were also explored.

Decision Trees and Rule-based Classifiers

Prinz *et al.* [31] presented an algorithm called *C STAGE* that used spectral energies with a set of rules for human-assisted classification. Validated on 45 elder subjects, they reported high correlation between the algorithm and manual scores in SWS and Wake stages while a lower correlation was observed in the other sleep stages. A similar approach was proposed by Pacheco *et al.* [14] using a set of bandpass filters to detect different frequency bands of interest (alpha, beta, etc.) using the amplitude and period of EEG signals. The thresholds based on which any given epoch would be classified were predefined. However, no results for the accuracy of the algorithm were reported in their publication.

Hanaoka *et al.* [32] used a method involving waveform recognition followed by the determination of the amplitude and frequency of the recognised wave. The latter parameters were compared with pre-defined thresholds to determine if the recognised waveform was one of the characteristic waves. The features were extracted using two EEG, two EOG and a single EMG channel while a decision tree was used to classify the features and assign one of the sleep stages. The algorithm achieved an accuracy of about 81% but was tested using only one recording of eight hour duration. The highest accuracy

was achieved in S2 stage (92%). S3 and REM were classified with accuracies of 73% and 76% while S1, Wake and S4 achieved 60%, 53% and 46% respectively.

Virkkala *et al.* [33] used two facial electrodes (one channel) for sleep stage classification. In this algorithm, the power in beta frequency band and peak-to-peak amplitude in 0.5-6 Hz band were calculated using DFT and used as the two primary features. A decision tree was used to classify the segments into either Wake, REM, SWS or S1/S2 (combined). Using a large database of 132 subjects for training and 131 subjects for testing, the algorithm resulted in an overall accuracy of 73.8% for the four-class classification. The sensitivity in S1/S2 stages was about 81%. This was probably due to S2 stage masking the number of S1 epochs being correctly detected. In Wake, REM and SWS stages, the sensitivities were between 60-70%. This method was further improved to lower its complexity and enhance the accuracy [34]. Using recursive IIR filter instead of DFT to calculate the features resulted in a slightly improved overall accuracy of about 77%. Garcia-Molina *et al.* [35] also used signals from two EOG channels to classify NREM stages of sleep. Their algorithm used the spectral energy features together with eye blink and movement detection. With a rule-based classifier and a small dataset consisting of 673 epochs they achieved an overall accuracy of 84%.

McPherson *et al.* [36] showed that there exists a high correlation between the bispectral index (BIS) values during sleep and the different sleep stages. These values can potentially be mapped on to the different stages of sleep. However, they also noted great deal of inter-subject variability making this non-trivial. Swarnkar *et al.* [37] later used bispectrum estimation from a single channel of EEG in a sleep staging algorithm. A set of thresholds were used to classify the EEG segments into either Wake, NREM or REM stages. Their method was tested using overnight recordings from 23 subjects and managed to achieve overall sensitivity of 75% with 66%, 78% and 73% in Wake, NREM and REM stages respectively.

Liang *et al.* [38] used a classical rule based method to classify several different features extracted from a single EEG, EOG and EMG channels. A total of 12 features were extracted that were fed to a decision tree. Although the total number of nodes in this tree were only 13 the actual number of comparisons were higher. This is because several features were being compared against their thresholds at each node. The output of the tree was further smoothed out using 11 contextual rules. The method resulted in an overall accuracy of about 87% using a database with 13 subjects. Apart from N1, which had a low sensitivity of only 35% all other sleep stages showed detection rates between 87-91%. However despite having a sensitivity of 88% in Wake stage, the selectivity value was low (44%) indicating a very high number of false positives. In other words, every other epoch classified as Wake was a misclassification. Further, the overall accuracy dropped to 78% when sleep data from PhysioNet Sleep EDF database (see Appendix A for a discussion on public databases) was used with sensitivities in Wake and N2 stages

also being reduced to 57% and 76% respectively. The same research group published another sleep stage classification algorithm where 21 features were extracted using multi-scale entropy (MSE) and autoregressive (AR) modelling [39]. With a linear discriminant analysis (LDA) classifier and further contextual smoothing the best case overall accuracy achieved was about 88%. The sensitivities in Wake, N2 and N3 stages were over 85%. The sensitivity for REM stage was very high (97.6%) but that for N1 was only 28%. The accuracy dropped again to 83.6% when signals from the PhysioNet Sleep EDF database were used for testing.

Figueroa Helland *et al.* [40] proposed a sleep staging algorithm that used spectral power ratios in different frequency bands of one EEG channel as feature input into a LDA classifier. They tested the method with 10 different subjects and reported an accuracy of 90% for epochs in which there was agreement among three scorers. For ambiguous epochs where there were disagreements between scorers, the accuracy was found to be 61%. Sanders *et al.* [41] compared the sleep scoring performance of their algorithm using three different feature extraction methods. They used average spectral power, peak power and cross frequency coupling features with a LDA classifier. On its own, the average spectral power features were shown to have the highest classification accuracy amongst the three methods while a combination of features resulted in the best case overall accuracy of about 75%. They used a single channel of EEG data of ten subjects from the PhysioNet Sleep EDF Expanded database (see Appendix A).

Fraiwan *et al.* [42] used wavelet packet transform (WPT) coefficients extracted from one EEG channel that were classified using regression trees. They extracted 54 features initially that were later reduced down to 20 by removing the redundant ones. Testing with 32 PSG recordings, their method resulted in an overall accuracy of 74%. The same group later presented another method [43] using a random forest classifier to score sleep stages from a single channel of EEG data. This consisted of ten individual decision trees each having the same weight. The final output was considered to be the mode of the result from all decision trees. Spectral energy features in the conventional frequency bands were used as input to this classifier. Three different methods for feature extraction were used, namely Choi-Williams Distribution (CWD), Continuous Wavelet Transform (CWT) and Hilbert-Huang Transform (HHT). A total of 13387 30-second epochs from 16 subjects were used to test the method which resulted in overall accuracies of 83%, 78% and 75% using CWT, CWD and HHT based feature extraction methods respectively.

Radha *et al.* [44] compared a large number of temporal, spectral, linear and non-linear features extracted from various EEG channels. They used Random Forest and Support Vector Machine (SVM) classifiers to identify the best combination for a single-channel online sleep staging algorithm. They determined that the relative spectral powers in conventional EEG frequency bands extracted from the frontal channels were the highest performing features. Using a test database of overnight sleep recordings of ten subjects

they achieved 80% accuracy with a random forest classifier, 77% with a one-versus-one SVM classifier and 69% with a one-versus-all SVM classifier.

Support Vector Machines

Although Radha *et al.* [44] achieved better results using a random forest classifier as compared to SVM, other researchers have shown high classification accuracies using SVM using different feature sets.

Khaligi *et al.* [45] used six EEG and two EOG channels to extract a large number of features including skewness, kurtosis, Hjorth parameters, Autoregressive coefficients and others. They also extracted wavelet based features using maximum overlap discrete wavelet transform (MODWT) [46]. A total of 570 features were initially extracted across all eight channels of which 176 were selected using minimum Redundancy Maximum Relevance (mRMR) feature selection method. These features were then classified using a support vector machine. With a database comprising of 14 subjects and using leave-one-out cross validation (LOOCV) they determined the overall accuracy of the algorithm to be 78% with highest sensitivity in Wake stage (93%). The sensitivities in REM and N1 stages were 71% and 53% respectively while that in N2 and N3 stages was 86%. An adaptive method to account for inter-subject differences was presented by the same authors [47] that used six EEG, two EOG and one EMG channels. In this case, a total of 176 features were used that included spectral power, skewness, percentiles and MODWT time-frequency features. Three different classifiers were used to demonstrate the performance and compare the results of kernel logistic regression (KLR) classifier with that of SVM using data from eight subjects. Although the SVM was found to be the best classifier, the KLR classifiers came very close to matching its performance in all stages except Wake where SVM was by far superior. In N1, all classifiers resulted in sensitivity close to 40% while in N2 and REM they were close to 75%. N3 had the highest sensitivity in all classifiers which was around 88%.

Koley and Dey [48] developed a sleep scoring algorithm using a single EEG channel. They extracted a total of 39 time, frequency and non-linear features which were reduced to 21 using SVM-RFE method to remove redundant features. A separate one-versus-all SVM classifier was then designed for each sleep stage. The classifier with the highest discriminant function value output represents the winning class and the corresponding sleep stage is then assigned to the epoch under analysis. They reported classification accuracy close to 90% on two test sets involving seven non-apnoea and five apnoea subjects respectively.

Kempfner *et al.* [49] proposed a three-class sleep staging algorithm to distinguish between Wake, REM and NREM stages. They used three EEG and one EOG channels to extract 23 features for each subject using an epoch size of three seconds. These features were then used as input for three one-versus-all SVM classifiers (one for each class) that

resulted in sensitivity of 95%, 88% and 77% for NREM, REM and Wake stages respectively. Note that these results are for small epoch sizes of three seconds only. This may appear to show a somewhat inflated accuracy. Gudmundsson *et al.* [50] developed an algorithm for sleep staging in children under the age of 5 years. They combined N1 and N2 stages to form light sleep stage and used one EEG channel to perform 4-state classification using SVM and k-Nearest Neighbour (kNN) classifiers. They extracted Hjorth parameters, spectral ratios and histogram features from the signal and achieved best case accuracy of 81% using histogram features with SVM.

Sousa *et al.* [51] used a combination approach with six EEG and two EOG channels to develop a sleep staging algorithm. Although their classifier was a decision tree, each node of the tree itself was an SVM classifier. They devised a further set of rules for classifying dubious epochs in order to improve the algorithm's accuracy. The algorithm used 40 features extracted from each channel including MODWT, relative power and harmonic parameters. With a test database including 14 subjects they achieved sensitivity of over 90% in Wake, REM and N3 stages, 84% in N2 and 71% N1 stage. Lajnef *et al.* [52] also proposed a method using SVM classification based on a decision tree approach. They initially extracted 102 features across five EEG, EOG and EMG channels which were later reduced to 32 to include only those with high discriminatory power. These features mainly included relative spectral powers and their ratios. For each SVM classifier, they selected a subset of features from this reduced pool. They used recordings from ten subjects for classifier training and five for testing and obtained high sensitivities of 90% and 97% in Wake and REM stages. In N1, N2 and N3 the sensitivities were 41%, 70% and 76% respectively.

Bajaj and Pachori [53] represented EEG signals as a time frequency image and performed segmentation based on histogram and frequency bands. These were classified using SVM and resulted in 88% accuracy when tested on PhysioNet EDF database. They achieved more than 80% sensitivity in Wake, S2, S4 and REM stages while those in S1 and S3 stages were 62% and 76% respectively. Wu *et al.* [54] used unique signal processing methods namely synchrosqueezing transform and empirical intrinsic geometry to extract features from four EEG channels together with one respiratory channel. They also used an SVM classifier and achieved an overall accuracy of 82%. Huang *et al.* [55] used power spectral density features extracted from the two frontal EEG channels with a relevance vector machine (RVM), which is a learning algorithm based on SVM. They used data from 10 test subjects and reported an overall accuracy of 77%. They achieved over 80% sensitivity for all sleep stages except N1, where it was only 22%.

Artificial Neural Networks

Frank Rosenblatt published the theory behind an artificial neural network (ANN) in 1958 [56] after which several researchers used such networks for classification in various

applications. However, it was not until the early 90's when computers became mainstream that artificial neural networks started being used popularly and consistently. The computational power available allowed researchers to construct multilayer neural networks with a large number of features. Unsurprisingly, artificial neural networks were also explored for use in automated classification of sleep stages.

Roberts and Tarassenko [57] were amongst the earliest to study the applicability of neural networks in sleep classification. They used Kalman filter coefficients extracted from one EEG and EMG channels that were fed into an unsupervised multilayer neural network. The self organising map revealed the presence of eight clusters and three possible trajectories between them. In other words, the unsupervised classifier showed that there were distinct stages of sleep and the transitions between them corresponded to Wake, REM and NREM sleep.

Schaltenbrand *et al.* [58] then presented a complete algorithm for sleep staging using an ANN classifier. They used one EEG, EOG and EMG channels from which ten, four and three spectral features were extracted respectively giving a total of seventeen input features. The classifier was supervised but an additional unsupervised neural network was also used at its output to improve the classification accuracy. On a dataset consisting of 20 normal subjects, the algorithm achieved an overall accuracy of 84.5%. S1 and S3 stages had the lowest classification accuracies of 22% and 55%, Wake and S2 had 84% and 88% while REM and S4 had an accuracy of over 90%. The algorithm was also tested using data from 20 depressed and 20 insomniac subjects. In these cases the classification accuracy was reduced to 81.5% and 81% respectively.

Pacheco and Vaz [59] used three EEG with one EOG and EMG channels to extract spectral and Hjorth features which were classified with an ANN. They used a dataset consisting of only two hours recording from eight subjects and reported the classification accuracy to be 89%.

Park *et al.* [60] proposed a hybrid system using a rule-based classifier followed by an additional neural network for cases where a reasonable result was not achieved with the former classifier. They extracted 58 features from single EEG, EOG and EMG channels and used recordings from two subjects for training and two subjects for testing. Using only the rule-based classifier, they achieved an accuracy of 83%. This increased to 86% when neural network classifier was added to the output stage. A different hybrid method was proposed by Tian and Liu [61] that used a neural network classifier for broad classification followed by fuzzy rule-based reasoning for refinement. Several spectral features were extracted from EEG and EMG channels that were roughly classified using a supervised self organising feature map into Wake, Light or Deep sleep. This was then refined using a rule-based approach resulting in an overall accuracy of 85% using test data from eight subjects. The sensitivity in each stage of classification ranged between 70-87%. Pinero *et al.* [62] presented a method based on fuzzy rules for classification of

sleep stages followed by rule-based inference corrector to improve its accuracy. They used data from two EOG, one EEG and two EMG channels of four subjects. From this, 30% epochs were used to test the algorithm achieving an overall accuracy of 82% on average.

Shimada *et al.* [63] explored the classification of characteristic sleep waves rather than sleep stages directly. These could subsequently be used for sleep staging. They used several time-frequency features extracted from a single EEG channel using data only from Wake, N1 and N2 stages. Their neural network classifier was shown to be capable in discriminating between spindles, humps and slow waves when tested with a small recording of ten minutes duration. Schwaibold *et al.* [64] followed a similar approach of identifying characteristic patterns from two EEG, two EMG and one EOG channels. They used a neural network for pattern recognition to identify known waves and then used a set of rules to assign one of the sleep stage based on the detected patterns. They used partial data from four subjects and reported the classification accuracy to be around 80%. Held *et al.* [65] performed sleep staging of infants with data extracted from four EEG, one EMG and one EOG channels. They also performed pattern recognition to identify sleep spindles, delta waves, theta activity, rapid eye movements and muscle activity. These were then classified using a neurofuzzy classifier which resulted in an overall accuracy of 84% on daytime nap recordings from five infants.

Oropesa *et al.* [66] used WPT coefficients from EEG signals as feature input to a neural network classifier achieving an accuracy of 78%. Their data set was small, consisting of 590 epochs, and they considered classification in S1, S2, Wake and REM stages while ignoring the SWS stage. For these stages, the sensitivities were 75%, 91%, 74% and 65% respectively. A sleep staging algorithm by Zoubek *et al.* [67] involved extracting relative energy features and wavelet coefficients together with some time domain features including entropy, standard deviation and skewness from one EEG, EOG and EMG channels. These features were then used with two different Bayes rule-based and a neural network classifier. Using 10,000 epochs selected from 47 recordings, the best classification accuracy was obtained with the neural network classifier. N3 stage had the highest sensitivity of 93%, N1 had the lowest of 65% while others were between 70-85%. Ebrahimi *et al.* [68] also extracted WPT coefficients from one channel EEG of seven subjects from PhysioNet EDF database and used them as the input feature set to a multilayer neural network. They achieved sensitivities between 80-88% for four sleep stages where the results of REM and N1 stages were pooled together. Using 64 spectral and wavelet features, Sinha [69] managed an accuracy of 95% for classifying spindles, REM and Wake epochs. However, their dataset used to obtain the test results was very small.

Kim and Park [70] extracted 120 features (of which 110 were spectral) of an epoch from single EEG channel for its classification. They used a feature selection method to reduce the number of features down to 32 which were used with a genetic algorithm together

with a neural network. However, the performance of their algorithm was not reported. Jo *et al.* [71] used relative powers from one EEG channel with a genetic algorithm and fuzzy classifier. They reported an accuracy of 84% with four test subjects for 4-state classification i.e. Wake, REM, Light and Deep sleep.

Charbonnier *et al.* [72] used a two stage algorithm where the first stage consisted of artefact rejection method while the second stage included four different neural network classifiers. Each classifier used a different combination of EEG, EOG and EMG signals from 46 recordings that were obtained from 13 adults. A total of 33 spectral and time domain features were extracted across the three channels and the overall accuracy of their algorithm was found to be 85.5%. N3 stage had the highest sensitivity of 95% while N1 had the lowest at 65%. The Wake, REM and N2 stages had sensitivities of 78%, 80% and 87% respectively. Ronzhina *et al.* [13] used the PSG recordings from PhysioNet EDF database to demonstrate the performance of their algorithm which used relative spectral power features extracted from one channel of EEG. Using different ANN configurations they achieved the highest overall accuracy of 77% with 30 input nodes, 11 hidden nodes and 6 output nodes. The sensitivities in Wake, S2, S4 and REM stages were between 77-90% while S1 and S3 were 31% and 29% respectively. Liu *et al.* [73] extracted spectral energy features using Hilbert-Huang Transform from single channel EEG recordings in the PhysioNet EDF database. Their neural network classifier achieved accuracy of 95% in Wake, 82% in N2, 93% in N3 and 87% in N1 and REM stages combined together.

Ma *et al.* [74] designed an artificial neural network-based decision tree where each node of the tree consisted of a neural network. They extracted 19 features including relative powers, central frequency, sample entropy and correlation dimension from one EEG channel. Their test results involving 14 subjects showed an overall accuracy of 74% with 100% sensitivity in REM and N3 stages, between 70-80% in N2 and poor detection in Wake and N1 stages.

Hsu *et al.* [75] proposed a method that also used energy features from one EEG channel but used a recurrent neural network (RNN) classifier and compared that with probabilistic and feedforward neural networks (PNN and FNN). They used recordings from the PhysioNet Sleep EDF database and reported superior accuracy of 87% with RNN as compared to FNN and PNN classifiers where the accuracies were 81% and 82% respectively. For N2, N3 and REM stages they achieved sensitivities of over 90% while for Wake and N1 the true positive rate was 71% and 37% respectively.

While most of the ANN based methods use either use spectral, entropy or wavelet features, some researchers also explored the use of less common features. McGrogan *et al.* [76] used a single channel EEG data from which reflection coefficients were extracted to be used as the input feature set for an ANN. Their algorithm was designed to classify an epoch into one of the three following stages: Wake, REM/Light sleep or Deep sleep. They used overnight recordings from nine subjects to train the algorithm and achieved

an accuracy of 72%. Sun and Cheng [77] used complexity measure features as input to a neural network and reported an average accuracy of over 90%. However, they used a test set consisting of 20 epochs of each sleep stage from six subjects which is very small to be of any statistical significance. Nevertheless, their method is still an example of using a different set of features with an ANN classifier. Tagluk *et al.* [78] used a neural network with 120 features derived from one EEG, two EOG and one EMG channels. They selected a small sample of epochs from 21 subjects and achieved an accuracy between 70-80% for all sleep stages.

It is quite clear that a properly designed neural network with a relatively large number of features results in a reasonably high classification accuracy. Ronzhina *et al.* [13] published an extensive review of neural network-based sleep staging algorithms and discussed the practical issues involved in their development. Becq *et al.* [79] compared five different classifiers for sleep staging and concluded that artificial neural networks are best suited for this application. In yet another paper, Gabran *et al.* [80] also reported neural networks to be the best classifier for sleep staging when used with wavelet features from a single EEG channel. However, neural networks are computationally expensive and may not be suitable for use in all situations.

Clustering, Statistical and Other Classifiers

A semi-automatic approach for sleep staging was proposed by Agarwal and Gotman [81] where initial clustering was performed automatically by an algorithm. Each cluster was then assigned a sleep stage by a reviewer. They used two EEG, two EOG and one EMG channels from twelve overnight recordings and extracted 13 features from the signals that included information on spindles, eye movements and spectral powers. These features were then used to generate unsupervised clusters. A reviewer then assigned a sleep stage to each of the automatically generated clusters and additional post-processing to the hypnogram could be optionally applied. This method yielded an overall accuracy of 80%, highest of 93% in S4 and lowest of 39% in S1. The REM stage had an accuracy of 73% while others were between 80-90%. Van Hese *et al.* [82] also followed a similar approach where Hjorth parameters, harmonic parameters and relative energy features were extracted from one EEG channel. These features were then automatically grouped into 20 clusters by k-means clustering. From this, the sleep stages are identified by visual analysis of clusters.

Flexer *et al.* [83] used hidden Markov models (HMM) to develop a continuous sleep staging algorithm with data from two EEG and one EMG channels. They achieved over 80% sensitivity in Wake and Deep sleep and less than 30% in all other stages. However, their main focus was identifying Wake, SWS and REM only. Improving their method, Flexer *et al.* later used data from one EEG channel with their classifier [84]. This was tested with 68 recordings across two sleep labs. For recordings from one lab they achieved

sensitivities of 79%, 82% and 68% for Wake, Deep sleep and REM respectively while for the other lab the sensitivities were 25%, 87% and 61% for these stages. For S1, S2 and S3, the detection accuracy was less than 40%. Based on these results, they concluded that recordings from different sleep labs are not directly comparable.

Doroshenkov *et al.* [85] used spectral features from two EEG channels with HMM classification. They tested their method using data from PhysioNet Sleep EDF database and reported sensitivities of 86% and 92% in REM and S4 stages, 60-70% in S2 and S3 stages, 51% in Wake and only 5% in S1 stage. HMM was also used by Pan *et al.* [86] for sleep classification. They extracted 13 temporal and spectral features from multiple EEG, EOG and EMG channels. Their test dataset consisted of recordings from 10 subjects and resulted in an overall accuracy of 85%. All sleep stages had sensitivities over 80% except S1 which had a lower accuracy of 34%. Yaghouby *et al.* [87] used 22 different subjects from the PhysioNet Sleep EDF Expanded database to perform unsupervised sleep staging using HMM. They used a single EEG channel for feature extraction and reported sensitivity in N1 stage to be 42%, about 85% in N2 and REM stages and about 70% in Wake and N3 stages. They also demonstrated the HMM method to be superior to other unsupervised methods such as k-means clustering and Gaussian Mixture Modelling (GMM). Garcia-Molina *et al.* [88] also used GMM to estimate the probability density function for spectral features extracted from a central EEG channel. They reported moderate agreement with manual scoring with the maximum kappa value of 0.63.

Malinowska *et al.* [89] used matching pursuit algorithm for identification of k-complexes and spindles as well as the amplitude and relative duration of alpha, theta and delta waves. This was followed up with a rule-based classifier to assign sleep stages. They used EEG, EOG and EMG recordings from 20 subjects and obtained about 80% sensitivity in REM and S2 stages, 71% in S4, 63% in S3 and under 40% in Wake and S1 stages. Gunes *et al.* [90] used k-means clustering to identify groups which would later be classified using k-Nearest Neighbour and decision tree classifiers. They extracted 129 features from one channel of EEG initially which were later reduced down to only the four most relevant ones. Their method resulted in an overall accuracy of 82%. The accuracies in Wake, REM and N2 stages were close to the overall value at 80%, 81% and 89% respectively while those in N3 and N1 were lower at 65% and 17%. Vivaldi *et al.* [91], [92] showed that frequency domain features can be used with principal component analysis (PCA) to determine clusters that can be mapped on to sleep stages. Langkvist *et al.* [93] addressed the issue of selecting relevant features by using deep belief network (DBN) for unsupervised feature selection. Their method used features from one EEG, EOG and EMG channels and used HMM for classification. Using leave-one-out cross validation with 25 subjects, they reported the sensitivities of 68%, 33%, 77%, 89% and 60% in Wake, N1, N2, N3 and REM stages respectively.

An unsupervised clustering approach was recently proposed by Rodriguez-Sotelo *et*

al. [94] using entropy features extracted from two channels of EEG data. They used 39 recordings from PhysioNet Sleep EDF Expanded database and computed a total of ten features including Fractal Dimension, Detrended Fluctuation Analysis, Shannon Entropy, Approximate Entropy, Sample Entropy and Multiscale Entropy. Using an unsupervised classifier they reported mean sensitivity in N2 and Wake stages to be 91% and 84% respectively. However, the sensitivities in N3, REM and N1 stages were quite low at 59%, 38% and 15% respectively. They also compared their algorithm's performance using the same features with a supervised neural network classifier which resulted in slightly better performance in REM, Wake and N1 stages but worse in N2 and N3 stages.

2.4.1 Validation of commercial sleep staging softwares and systems

While new algorithms are continually being developed by researchers, a number of softwares and systems have already been introduced commercially to perform either assisted or fully automated sleep staging. Some of the automated tools are available as part of the complete PSG signal acquisition systems while others are available as a standalone program or software as a service (SaaS). These tools have been used in some studies to validate their performance and test their effectiveness in clinical environments.

Kubicki *et al.* [95] evaluated one of the earliest commercial sleep staging system. This is known as the Medilog Sleep Stager by Oxford Medical Systems. For this study, they used PSG data from ten subjects and found the agreement between automatic and manual scoring to be 73%. They also noted that the software had difficulty identifying REM, Wake and S2 stages resulting in higher classification of S1, S3 and S4 sleep stages. Ferri *et al.* [96] also evaluated the same system comparing it with manual scoring from nine different sleep labs. They came to the same conclusion about REM and Wake stages as Kubicki *et al.* but also noted some difficulty in automatic scoring of S1 stage rather than S2.

White and Gibb [97] evaluated the performance of Healthdyne ALICE 3 system for automatic sleep scoring. Their study consisted of 5 patients for epoch-by-epoch scoring analysis using four EEG, two EOG and one EMG channels. The automatically generated hypnogram was edited by a technician and then compared against manual paper-based scoring which resulted in an accuracy of 76%. The conclusion from this study was that automatically scored hypnograms still requires some technician editing to have a good enough accuracy for clinical usage.

Anderer *et al.* [98] discussed an internet-based sleep analysis software called Somnolyzer 24×7 that requires submission of the complete PSG recording through a secure connection. The software, described earlier in detail [99], used more than 20 different methods for sleep analysis. It requires one EEG, two EOG and one EMG channels from which a large number of features are extracted. It uses a decision tree structure for classification where each node of the tree is an LDA classifier and uses features which

are extracted earlier. It was validated for both R&K [99] and AASM rules of classification [100]. For R&K, they achieved an overall accuracy of about 80% using 286 PSG recordings. For AASM, 72 PSG recordings were used and the accuracy was increased slightly to 82%.

Pittman *et al.* [101] evaluated the performance of Morpheus I Sleep Scoring System (WideMed Ltd.) in people with sleep-disordered breathing. They used four EEG with single EOG and EMG channels recorded from 31 subjects. The automatic scoring from the software was compared against manual scoring performed by two different scorers. The results showed agreement of 78% and 73% between the automatic system and the two scorers. Svetnik *et al.* [102] looked at the scoring performances of both the Morpheus and Somnolyzer 24x7 systems. They used 164 PSG recordings from 82 subjects and determined that the agreement between fully automated scoring of the two systems and manual scoring was between 70-73%.

Caffarel *et al.* [103] evaluated the performance of BioSleep, an ANN-based automatic sleep scoring system, and found it to have poor accuracy when compared to manual scoring based on R&K rules. They concluded that the software, although good for micro-arousal scoring, is not suitable for classifying sleep according to R&K rules.

Berthomier *et al.* [104] published the performance of a commercial sleep staging software known as ASEEGA. They used single channel overnight EEG recordings from 15 adults that were scored with this software. This software performs artefact rejection at the initial stage and then uses a set of individually tailored filter banks for feature extraction. A fuzzy classifier followed by contextual rule-based smoothing then assigns a sleep stage to the EEG segment under analysis. The software achieves classification accuracy of more than 80% in all stages except N1 where the sensitivity is 36%.

Kaplan *et al.* [105] evaluated the performance of a sleep-wake algorithm called Z-ALG, used in Zmachine (a commercial portable single channel EEG acquisition system). The algorithm uses data obtained from the mastoids, A1-A2 channel, and extracts six time and frequency domain features for each epoch. The frequency features are separated using hyperplane partioning. Two detectors are then used in parallel with different sets of extracted features. An epoch is classified as sleep if either one (or both) of the detectors classify it as sleep. This study used 99 subjects and the algorithm was reported to have a 95.5% sensitivity in detecting the sleep state. The Z-ALG algorithm only discriminates between Wake and Sleep states and does not classify the different stages of sleep.

Shambroom *et al.* [106] presented an evaluation of a commercial sleep monitoring device (ZEO), that uses electrodes in wireless headband for data acquisition from the forehead. The algorithm classified the epochs as either Wake, REM, Light or Deep sleep and achieved sensitivities of 64%, 86%, 86% and 71% in these stages respectively. The overall accuracy of the system was found to be 81% in this study and used recordings from 26 subjects.

Malhotra *et al.* [107] evaluated the performance of an automatic sleep scoring system developed by YST Limited. The proprietary software used two EEG, two EOG and one EMG channels for scoring and was tested using 70 PSG recordings across different sleep labs. The results showed 56% accuracy in N1, 84% in N2, 47% in N3 and 64% in REM stage.

2.5 Review of commercial sleep scoring systems

2.5.1 Introduction

The research efforts of over four decades have resulted in a number of commercially available portable systems for recording and analysing sleep signals automatically. Using a very simplistic model, the functionalities of such a system can be divided into two main parts.

1. Data acquisition and storage
2. Signal analysis and scoring

Some commercial systems provide a full suite of tools including both hardware for acquiring signals and an integrated software for the analysis and scoring of these signals. Other systems exist that perform the task of data acquisition and storage and allows for the possibility of manual analysis or automated analysis of these signals using third-party softwares. To serve this latter market as well as scoring of signals acquired from clinical EEG systems, a number of companies have come up with softwares to automatically analyse and score sleep stages from recorded data. These software packages usually detect different stages of sleep and other micro-events (e.g. sleep spindles and k-complexes), and generate a report for the physician. The software packages are generally marketed in one of the following two ways.

1. A package to be installed on a computer making use of the processing power of the latest multi-core processors.
2. A web-based system where clinicians can upload their recorded data in standard EDF format [108]. The data is then marked automatically (or manually in some cases) using powerful computers, and a report generated within hours. The time delay in getting the report can be anywhere between 2-48 hours depending on the service used. This approach requires caution to ensure all patient data privacy procedures are followed.

In both these cases a strict guideline is usually provided with the software about how data should be recorded and which channels may be required for analysis by each software.

All the PSG systems in the market aim to perform the same task of getting and classifying the sleep signals. However, there are factors and functions that set one apart from the other. The following is a non-exhaustive list of important features that are helpful in differentiating the PSG systems and comparing them.

Number of channels: The minimum set of channels required for sleep scoring is 3 for EEG, 2 for EOG and 1 for EMG. However, a number PSG devices come with the ability to record from many more channels varying from 2 to 256.

Power source and operating lifetime: The clinical PSG recording units are powered by the mains while the ambulatory PSG units are generally battery powered (with the ability to power them from mains). Depending on the size of the battery and the number of channels recorded (and other factors such as wireless transmission, etc.), a system can last several hours on a single charge. In most cases it is acceptable to have a system that can last an entire night on a single charge. That, however, is the minimum requirement and a longer lifetime would make it more convenient for the end user.

Communications and Data storage: The recording unit should be able to store recorded data or transmit it somehow to another system for storage and analysis. This may be achieved by storage on flash memory devices, such as an SD-card, or transmission via local area network (LAN) or a wireless transmitter.

Sampling frequency: This has an effect on the quality of signal acquired and the commercial devices available may have a range of 200-2000 Hz in the sampling frequency depending on the environment in which they are to be used.

Weight and size: Ambulatory/portable devices must be light and small whether used in a clinical or an in-home setting. The device is expected to be worn by the patient and carried around at night without the need to unplug it. So the size and weight are crucial in determining how portable a system is.

Analysis software: Some kind of analysis software is always bundled with the PSG systems. It may simply be a viewer to see the recorded signal or have advanced features such as automatic event and stage detection.

2.5.2 PSG systems

With the above points serving as the important distinguishing characteristics, this section presents a review of the commercially available portable/ambulatory PSG systems. Although most PSG systems also have the ability to record other physiological signals such as ECG, pulse rate, air flow, etc., those will not be discussed. Instead, the focus will be on the three signals that are needed for sleep staging. Therefore, only systems that record these signals will be considered in this review.

Somté PSG by Compumedics [109]

An ambulatory PSG device that has wired electrodes connected to a patient input box that transmits wirelessly over Bluetooth to the recorder placed up to 10 metres away. The system weighs 205g without batteries. It uses 2 AA batteries for operation and lasts for up to 24 hours of recording. It allows for the recording of six EEG, two EOG and two EMG channels. It is not clear whether this is the combined weight of the two units or just that of the patient input box. The recorder size is 113mm $\times$ 65mm $\times$ 30mm while that of patient input box is 53mm $\times$ 133mm $\times$ 25mm. The system comes with an analysis software that helps to automate scoring of sleep stages.

Siesta by Compumedics [109]

This system consists of a small data recorder with a 32-channel input. It weighs 300g with battery and has dimensions of 142mm $\times$ 80mm $\times$ 40mm. It can record data on a memory card or transmit it wirelessly to a computer for real-time analysis.

Sapphire PSG by Clevemed [110]

A type I and II PSG device capable of recording six EEG, two EOG and two chin EMG channels along with other signals. The recording device is powered using 4 AA batteries, weighs 538g and provides 12 hours of continuous recording. It transfers data wirelessly to a base station, up to 100 feet away, attached to a computer. Its size is 216mm $\times$ 97mm $\times$ 36mm and has an option of recording data to a memory card. The data can be scored manually and analysed with the Crystal PSG software that comes with this recorder.

DREAM by Medatec [111]

This polysomnography system allows 33 channels to be recorded and has dimensions of 120mm $\times$ 60mm $\times$ 30mm. It does mention wireless transmission but it is not clear where that feature is integrated. It comes with an analysis software called Brainnet that performs automatic sleep analysis.

Morpheus by Micromed [112]

This PSG recorder can be used as a wired/wireless head-box for on-line analysis as well as an ambulatory recorder while storing data on a memory card. It has 12 EEG channel inputs, weighs 250g and has the dimensions of 110mm × 80mm × 30mm. In ambulatory recording mode the batteries last for over 24 hours. Its analysis software allows for viewing data on a computer.

Embletta Gold by EMBLA [113]

A portable PSG system that allows a single EEG and an EOG channel to be recorded for up to 24 hours. It weighs 218g and has dimensions of 23mm × 71mm × 140mm. It is bundled with an analysis software that allows sleep staging with custom defined rules and has advanced features to allow for networking and remote access of data.

Xltek PSG - Home Sleep by Natus [114]

An ambulatory PSG device operating on two AA batteries with internal memory to store up to 96 hours of data (battery lifetime is not provided). Recorded data is extracted by connecting the device with a computer via a USB cable. It weighs less than 300g and has 24 EEG channels input. Its analysis software, Sleepworks, can be used to review data, however automatic scoring is not provided.

AURA PSG Ambulatory Systems by Grass Technologies [115]

This PSG system has three EEG, one EOG and two EMG channel inputs and can connect to an additional base station wirelessly via Bluetooth for real-time data acquisition. It can also be used in ambulatory mode to store data on a memory card that can be read via TWin software that comes with this system. Its dimensions are 89mm × 149mm × 25mm and its weight is 280g. It comes with a rechargeable 3.6 V battery that lasts for up to 10 and 12 hours in wireless and ambulatory modes respectively.

NOX-T2 Portable Sleep Monitor by CareFusion [116]

This is a very compact device weighing only 88g. It has dimensions of 79mm × 63mm × 21mm and is powered using a single AA battery. It can record any two of the bipolar EEG and EMG channels that can be downloaded on a computer via USB connectivity. It can record data for up to 24 hours on a single charge of battery.

SOMNOwatch plus EEG6 by SOMNOmedics [117]

A miniaturised unit that can record four EEG, two EOG and one EMG channel for up to 46 hours. Powered by a rechargeable battery, it consists of two small units. The

SOMNOwatch (30g in weight) is the central unit which is attached to a head-box (60g). The electrodes are then connected to the body from the head-box which is very compact and has the dimensions of 61mm × 56mm × 13mm. Recorded data can be transferred to a computer via USB port and can be analysed using the bundled software that supports limited detection of sleep events. SOMNOmedics also makes another PSG device for ambulatory and online data acquisition called SOMNOscreen plus that weighs 220g with the battery and can provide 36 hours of continuous recording using the same head-box add-on.

Easy III PSG by Cadwell [118]

This comes as an ambulatory amplifier which weighs about 140g with a recorder weighing 770g including batteries. It can record 32 channels of data that can be scored manually in real-time while data is being acquired. This is more suitable for clinical use than for in-home testing.

Vitaport 4 PSG-lite by Temec [119]

Another PSG system with a single EEG, EOG and EMG channel inputs and a software viewer to manually score sleep stages. The dimensions and weight information is not provided. It can record up to 16 hours of data on a single charge of its battery.

eXea PSG 3 by Bitmed (Sibel Group) [120]

This ambulatory PSG device allows for 12 ExG differential channels to be recorded for up to 29 hours using a rechargeable battery.

Sleep&Go by Bitmed (Sibel Group) [120]

This is a type III PSG device that has an additional ExG module to record 3 channels for up to 24 hours of data on a memory card using 2 AA batteries. It weighs 140g without battery and has compact dimensions of 86mm × 86mm × 24mm.

BW3 PSG by Sleepvirtual [121]

Another type I PSG device that comes as a separate head-box and amplifier to record up to 50 channels and has ethernet connectivity to analyse data using the accompanying BWAnalysis PSG software.

Harmonie-S PSG System by Stellate (Natus) [114]

This is an advanced PSG system with 44 channels of input and has networking capabilities to analyse the data in real-time on a computer with its software. The software can detect

some sleep events such as spindles and rapid eye movements and can adapt to the scoring style of the technologist to make manual sleep staging efficient.

TREA Ambulatory EEG System [122]

An ambulatory EEG system that can record 25 EEG channels for 72 hours at a sampling rate of 200 Hz. It is powered by 3 Lithium AA batteries and the unit weights about 140g without the batteries.

g.Nautilus by g.tec [123]

A wireless EEG acquisition system capable of up to 10 hours continuous recording with a range of 10 metres. It can be used with either 8, 16 or 32 channel configurations. The recording unit is attached to a cap so using the device is simply a matter of putting on the cap with the attached unit.

ENOBIO32 by Neuroelectronics [124]

This is also a wearable wireless EEG acquisition device attached to a cap which has the sensors. It can record 32 channels at a resolution of 24 bits and has the option of storing the data on SD card or transmit via Bluetooth for up to 14 hours. The weight of this device is 65g.

Mobita Wireless EEG by BIOPAC [125]

This is a 32 channel ExG acquisition system with the ability to store data locally or transmit using WiFi. It can be attached to the electrode cap and has a battery lifetime between 8-10 hours on a single charge.

72-Channel Dry EEG Headset System by Cognionics [126]

This is a wireless headset for high density recordings and weighs 350g with the batteries. It can operate for 6 hours when using the Bluetooth transmission mode or 10 hours when data is stored on a SD card.

Stat X-Series Mobile EEG by Advanced Brain Monitoring [127]

These are wireless EEG acquisition systems available 4, 10 and 24 channel configurations. Their transmission range is up to 20 metres however the battery lifetime of these devices is not known.

Sleep Profiler by Advanced Brain Monitoring [128]

This device has the ability to record three frontal EEG channels using a band that is worn on the forehead. The recorded data is analysed and scored via a web-based interface.

Zeo Sleep Manager [129]

Featuring a wireless headband collecting data from the forehead, Zeo provides information about different stages of sleep classified as Wake, Light sleep, Deep sleep and REM. It is not a medical device and has been designed for consumer use for sleep tracking.

EPOC+ by Emotiv [130]

A 14-channel EEG acquisition headset with wireless transmission of either 6 or 12 hours using Bluetooth or a proprietary transmitter respectively.

2.5.3 Sleep scoring softwares

In this section, offline and web-based software services for automated sleep scoring are briefly reviewed.

Michele Sleep Scoring [131]

A web-based sleep scoring service that allows sleep technologists to upload recorded data. It returns the scored data in 15 minutes which can be edited or reviewed manually before generating the final report.

Aseega [132]

Aseega Online is a also web-based service targeted towards researchers to get a detailed report on uploaded sleep data. Once data is uploaded, Aseega takes 5 minutes to perform the analysis and generate reports. Aseega can score sleep stages using one or more channels of EEG.

Somnolyzer 24×7 [133]

Another web-based service that adds expert review process on top of automated scoring to minimise errors and includes full sleep staging with micro-events detection.

N2 Sleep [134]

This is also a web-based service which returns the scored data with reports in 48 hours. However, data is scored manually at the back-end and therefore the turnaround time is high.

FASS [115]

FASS (Fully Automated Sleep Stager) is a software by Grass Technologies that performs sleep staging using EEG, EOG and chin EMG data and allows the thresholds to be manually changed for each subject. It, however, works with the PSG systems made by Grass Technologies only.

Morpheus - Automated Sleep Testing Management [135]

Morpheus by WideMed provides a completed web-based solution to manage and automatically score sleep data that is accessible from anywhere using internet.

SleepView Web Portal [110]

This is a web-based solution by CleveMed for manual scoring of sleep stages by certified sleep physicians.

2.5.4 Conclusions

All the PSG systems discussed in this section have a similar design in that they have a portable recording unit to which wired electrodes are attached. The unit itself is mostly powered by a rechargeable battery that can last for 7-48 hours. A number of factors affect the battery lifetime including:

- Number of channels recorded.
- Wireless transmission of data or local storage.
- Size (capacity) of battery

The recording units available weigh in the range of 250g and also have small sizes. In most of the systems, the recorders are either strapped around the waist or chest region or placed next to the bed. They require some form of communication method from the device to a computer for further analysis of data and the options include wireless transmission, ethernet connection or local memory storage.

The systems reviewed in this section provide few different features to improve the PSG tests. However, there are still certain limitations including usability, wearability, power consumption, reduction in number of channels and more. The next section looks at the challenges associated with improving these systems in detail and proposes a solution that can help alleviate the limitations of existing systems.

2.6 Proposed solution and challenges

PSG in its existing form suffers from a number of drawbacks including high cost, difficulty in usage and requirement of a lot of time for recording and analysis. HPSG is the obvious potential alternative of clinical PSG in cases where possible. Even if it is not an alternative it can serve as a potential screening step in many cases. However, asking a patient to correctly place a large number of recording electrodes is not practical. Hence, researchers have explored a number of algorithms to potentially extract data from a reduced number of channels. Such methods are often very complex and require implementation on processors that are not suitable for resource-constrained wearable systems. A variety of commercial systems have been introduced that try to mitigate for some of the shortcomings of the existing PSG, however it is clear that there is a need to do more.

This thesis proposes the development of a completely wireless single-channel sleep scoring system that is easy to put on, comfortable to sleep with, gives instant results and can be used anywhere. Such a system can be implemented in one of the two following ways.

Signal processing and sleep staging on the sensor node (Approach 1)

Using this approach, data is continuously processed at the sensor end and only the sleep stage information is required to be transmitted. The primary advantage in this case is the hugely reduced data rate. If the epoch size is 30 seconds then there are only few bits of payload that must be transmitted after that time. This scenario is ideal for the ultra-low power transmitters such as Bluetooth Low Energy (BLE) [136]. Since BLE is now widely available in most smartphones, there is no need for additional hardware to receive data and act as an intermediary between the sensor and a different display device. It also does not drain the battery of a smartphone and can lead to a superior user experience.

In systems using this architecture, algorithms for scoring sleep are constrained by the processor capability. This is because complex algorithms running on these processors directly result in higher power consumption and a reduced battery life of the system. The two key stages in all sleep staging methods are feature extraction and classification. The number and types of features extracted and the choice of classifier used depend on the target application of an algorithm. For example, it may be acceptable to use 200 features with a multistage neural network in an analysis software running on a computer but the limitations of a wearable battery-powered system prohibits the use of complex features and classifiers consequently leading to a reduction in performance. Therefore a trade-off between acceptable levels of performance and algorithm complexity must be made to meet system specifications.

Raw data transmission or storage with signal processing at the receiver end (Approach 2)

In this approach, data is continuously acquired and transmitted to a base station nearby where they are received and processed to extract relevant sleep staging information. There are a number of advantages of this system architecture. The primary advantage is that all the computational load of an automated classification algorithm can be pushed to the receiver end since no data is to be processed at the sensor. The receiver has to be within the range at all times which is not an issue while performing sleep recordings. However, continuous data transmission comes with a whole host of design issues such as the data rate, security, reliability and transmission power. Further, the computational power available at the receiver depends on the type being used. For example, a desktop computer may be able to run much more complex algorithms in comparison to a smartphone.

Rather than transmitting data continuously, it is also possible to save them on local memory and retrieve later to perform sleep analysis. Further, a lightweight compression scheme can also be incorporated prior to transmission. However, regardless of whether data is being transmitted or stored, compressed or uncompressed, it does not make a difference to the design of sleep algorithms since they will still be running on machines that can cope with high computational requirements. This system architecture, with its feasibility and other system considerations, is discussed in great detail in Appendix D.

Which approach is better?

There are pros and cons of implementing the system in either of the two architectures discussed and the best choice can be made depending on the application and setting in which such a system is to be used. Hence, it is impossible to recommend one over the other. However, keeping in mind the growing trend of mobile device usage it is safe to say that, at least from the perspective of user experience, it makes sense to have a system that can communicate with a smartphone where the latter can act as the receiver and processor for the second approach and receiver in case of the first approach.

While the first approach definitely requires a low complexity algorithm for the aforementioned reasons, limiting the second approach to cases where the receiver is a smartphone also puts some constraints on the algorithm development. This is because, compared to a desktop computer, a smartphone will have lower processing power and bandwidth available. Therefore, regardless of the chosen approach there is a definite need for low complexity sleep staging algorithms that can be run on a sensor node (Approach 1) or a smartphone (Approach 2). Thus, this thesis focuses on exploring novel features and developing algorithms for sleep stage classification that can be useful in both these approaches.

2.6.1 Challenges of the proposed approach

The design of a wearable system with automatic sleep staging poses a number of challenges both in hardware and software development. These inter-related challenges are explained in detail below.

Number of channels

Based on the AASM recommendations, three EEG channels are required in PSG systems together with EOG and EMG channels. Ruehland *et al.* [17] reported no significant differences in sleep scoring reliability when using a single EEG channel, so this number can potentially be reduced to one. However, the EOG and EMG channels are still required since identifying REM stage epochs involves observing the chin muscle and eye activity [4]. A number of researchers have attempted to use only the EEG component for sleep staging in order to further reduce the total number of channels down to one. This reduction has many advantages including lower power consumption, smaller data rate, easier to put on and more comfortable to use. Having a single channel results in a few downsides as well. The first and most important one being the reliability since any loss in electrode contact means there are no backup electrodes to fall upon. Another downside is perhaps a reduction in accuracy since EOG and EMG channels are not available to help in ambiguous cases where EEG may not be enough to classify an epoch. Nevertheless, the advantages of single EEG channel, if implemented properly outweigh these downsides and can result in a system that has the potential to benefit a much wider group of population.

Power consumption

The power budget of a wearable system is extremely small which in turn limits the computational load that can be put on to the processing unit. A reduction in the number of channels is directly beneficial for saving power since there is less data to acquire and process. Even with that, the algorithm still has to be of sufficiently low complexity to run on the low power system. The other system component that often requires more power is the transmitter. This could be tackled by following one of the two approaches above depending on the application. Overall, at the very least, the system should be able to run for about eight to ten hours (average duration of a night sleep) without the need for changing or recharging the battery. The size and capacity of the battery will then set the limit on the maximum power budget available for the system and will dictate the design of algorithm and transmission protocols.

Performance metrics

Although a large number of sleep staging algorithms already exist in the literature and achieve varying degrees of accuracy it is very difficult to compare them because of the

inconsistent metrics used to report their performance. The use of data from different sources while selecting different sections as well also affect these reported accuracies. It is entirely possible that an algorithm that uses data from source A and achieves an accuracy of 90% obtains a much lower accuracy when using data from source B. In this thesis, prior to any algorithm design, a set of recommendations is therefore proposed for consistent and standardised performance assessment of automatic sleep staging algorithms (see Chapter 3).

Detection of REM sleep

One of the greatest bottlenecks in reducing the number of channels is the need for EOG and/or EMG channels to discriminate between REM and N1/Wake stages that appear very similar on EEG. Therefore, in order to realise a single-channel EEG based solution, novel features that are able to detect REM sleep epochs from EEG are required. This involves analysis using multiple features and using them in a way that gives them the highest discriminatory ability.

Detection of sleep microevents

Sleep spindles are important microevents that occur mostly during N2 and N3 stages of sleep. They are not only helpful for identifying the aforementioned sleep stages but are also an important research subject. It is therefore helpful to have an algorithm for their detection that can either be integrated with the larger sleep classification algorithm or be used on its own for research on spindles. This algorithm will also have to be constrained by the same limitations as the low complexity sleep staging algorithm (see Appendix C).

Data storage and transmission

As discussed earlier in Section 2.6, it is possible to either store the data locally or transmit wirelessly to a nearby receiver. In the case of raw data transmission, it is helpful to compress this data to reduce the transmission bandwidth and save some power. The amount of data, transmission rate, payload size, power consumption, data integrity and security are all factors that must be taken care of when designing this component of the system.

Algorithm accuracy

A small power budget limits the complexity of an algorithm and dictates the kind of signal processing methods that can be used for its design. The lower complexity algorithm will then be able to work within the constraints but has the downside of lower accuracy compared to an algorithm that has no such restrictions. There is a tradeoff between the

algorithm complexity (and hence its power consumption) and the maximum accuracy that can be achieved with it and this needs to be looked at in detail.

User experience

The need for wearable sleep staging systems stem from the desire to make them comfortable for the patients to use and easy to handle. The systems should allow patients to sleep better during the test and provide an improved overall user experience. All of the challenges stated above directly or indirectly affect the user experience since they are related to the design of the system.

2.7 Conclusion

This chapter introduced the different characteristics and stages of human sleep and discussed how the lack of sleep can lead to certain fatal disorders. The PSG test used to diagnose sleep orders was discussed in detail together with its limitations and drawbacks in its existing form. A wealth of academic literature and latest commercial PSG systems were then reviewed and discussed to see what different research groups are doing to tackle these limitations. The concept of Home PSG was discussed in detail highlighting various system aspects that are needed to make this system work. This led to the proposal of a single channel wearable sleep scoring system. The design challenges of this proposed system were then discussed which forms the basis of the work presented in the next chapters.

References

[1] S. Sanei and J. A. Chambers, *EEG Signal Processing*. Wiley, 2008.

[2] S. Kubicki, W. Scheuler, and H. Wittenbecher, "Short-term sleep EEG recordings after partial sleep deprivation as a routine procedure in order to uncover epileptic phenomena: an evaluation of 719 EEG recordings," *Epilepsy Res. Suppl.*, vol. 2, no. 1, pp. 217–30, 1991.

[3] A. Rechtschaffen and A. Kales, *A manual of standardized terminology, techniques and scoring system for sleep stages of human subjects*. Washington D.C.: Public Health Service, U.S. Government Printing Office, 1968.

[4] C. Iber, S. Ancoli-Israel, A. Chesson, and S. Quan, *The AASM manual for the scoring of sleep and associated events: rules, terminology and technical specifications*. Westchester, IL: American Academy of Sleep Medicine, 2007.

[5] "Sleep SOS report: The impact of sleep on society," *The Sleep Alliance*, 2007.

[6] A. J. Casson, S. Smith, J. S. Duncan, and E. Rodriguez-Villegas, "Wearable EEG: what is it, why is it needed and what does it entail?" in *IEEE EMBC*, Vancouver, August 2008.

[7] W. W. Flemons, N. J. Douglas, S. T. Kuna, D. O. Rodenstein, and J. Wheatley, "Access to diagnosis and treatment of patients with suspected sleep apnea," *Am. J. Respir. Crit. Care Med.*, vol. 169, no. 6, pp. 668–672, 2004.

[8] Philips Respironics. (2013) The Sleep Technician Guide. [Online]. Available: http://www.respironics.com/.

[9] Medic On Web. (2012) Medical Knowledge for the Non Medical People. [Online]. Available: http://www.mediconweb.com/.

[10] PhysioNet. (2013) Sleep-EDF Database. [Online]. Available: http://www.physionet.org/physiobank/database/sleep-edf/.

[11] A. L. Chesson, R. B. Berry, and A. Pack, "Practice parameters for the use of portable monitoring devices in the investigation of suspected obstructive sleep apnea in adults," *Sleep*, vol. 26, no. 7, pp. 907–913, 2003.

[12] M. Bruyneel and V. Ninane, "Unattended home-based polysomnography for sleep disordered breathing: Current concepts and perspectives," *Sleep Med. Rev.*, vol. 18, no. 4, pp. 341–47, 2014.

[13] M. Ronzhina, O. Janousek, J. Kolarova, M. Novakova, P. Honzik, and I. Provaznik, "Sleep scoring using artificial neural networks," *Sleep Med. Rev.*, vol. 16, no. 3, pp. 251–63, 2012.

[14] O. Pacheco, A. Tomé, and F. Vaz, "Automated analysis of sleep EEG: a new approach," in *IEEE EMBC*, San Diego, October 1993.

[15] H. Danker-Hopfe, D. Kunz, G. Gruber, G. Klösch, J. Lorenzo, S. Himanen, T. Kemp, B. amd Penzel, J. Röschke, H. Dorn, A. Schlögl, E. Trenker, and G. Dorffner, "Interrater reliability between scorers from eight european sleep laboratories in subjects with different sleep disorders," *J. Sleep Res.*, vol. 13, no. 1, pp. 63–9, 2004.

[16] H. Danker-Hopfe, P. Anderer, J. Zeitlhofer, M. Boeck, H. Dorn, G. Gruber, E. Heller, E. Loretz, D. Moser, S. Parapatics, B. Saletu, A. Schmidt, and G. Dorffner, "Interrater reliability for sleep scoring according to the Rechtschaffen & Kales and the new AASM standard," *J. Sleep Res.*, vol. 18, no. 1, pp. 74–84, 2009.

[17] W. R. Ruehland, F. J. O'Donoghue, R. J. Pierce, A. T. Thornton, P. Singh, J. M. Copland, B. Stevens, and P. D. Rochford, "The 2007 AASM recommendations for EEG electrode placement in polysomnography: impact on sleep and cortical arousal scoring," *Sleep*, vol. 34, no. 1, pp. 73–81, 2011.

[18] J. R. Smith, M. Negin, and A. Nevis, "Automatic analysis of sleep electroencephalograms by hybrid computation," *IEEE Trans. Syst., Man, Cybern.*, vol. 5, no. 4, pp. 278–284, 1969.

[19] T. M. Itil, , D. M. Shapiro, M. Fink, and D. Kassebaum, "Digital computer classifications of EEG sleep stages," *Electroencephalogr. and Clin. Neurophysiol.*, vol. 27, no. 1, pp. 76–83, 1969.

[20] L. E. Larsen and D. O. Walter, "On automatic methods of sleep staging by spectra of electroencephalograms," *Electroencephalogr. and Clin. Neurophysiol.*, vol. 28, no. 1, pp. 459–467, 1970.

[21] J. D. Frost, "An automatic sleep analyzer," *Electroencephalogr. and Clin. Neurophysiol.*, vol. 29, no. 1, pp. 88–92, 1970.

[22] J. R. Smith, M. J. Cronin, and I. Karacan, "A multichannel hybrid system for rapid eye movement detection (REM detection)," *Comput. Biomed. Res.*, vol. 4, no. 3, pp. 275–290, 1971.

[23] A. Kumar, "A real-time system for pattern recognition of human sleep stages by fuzzy system analysis," *Pattern Recognition*, vol. 9, no. 1, pp. 43–46, 1977.

[24] I. Gath and E. Bar-on, "Computerized method for scoring of polygraphic sleep recordings," *Comput. Programs Biomed.*, vol. 11, no. 3, pp. 217–223, 1980.

[25] B. Kemp, E. W. Gröneveld, A. J. Janssen, and J. M. Franzen, "A model-based monitor of human sleep stages," *Biological Cybernetics*, vol. 57, no. 6, pp. 365–378, 1987.

[26] E. Stanus, B. Lacroix, M. Kerkhofs, and J. Mendlewicz, "Automated sleep scoring: a comparative reliability study of two algorithms," *Electroencephalogr. and Clin. Neurophysiol.*, vol. 66, no. 4, pp. 448–456, 1987.

[27] H. Kuwahara, H. Higashi, Y. Mizuki, S. Matsunari, M. Tanaka, and K. Inanaga, "Automatic real-time analysis of human sleep stages by an interval histogram method," *Electroencephalogr. and Clin. Neurophysiol.*, vol. 70, no. 3, pp. 220–229, 1988.

[28] J. C. Principe, S. K. Gala, and T. G. Chang, "Sleep staging automaton based on the theory of evidence," *IEEE Trans. Biomed. Eng.*, vol. 36, no. 4, pp. 503–509, 1989.

[29] J. C. Principe and J. R. Smith, "SAMICOS–a sleep analyzing microcomputer system," *IEEE Trans. Biomed. Eng.*, vol. 33, no. 10, pp. 935–941, 1986.

[30] B. H. Jansen and B. M. Dawant, "Knowledge-based approach to sleep EEG analysis–a feasibility study," *IEEE Trans. Biomed. Eng.*, vol. 36, no. 5, pp. 510–518, 1989.

[31] P. N. Prinz, L. H. Larsen, K. E. Moe, E. M. Dulberg, and M. V. Vitiello, "C STAGE, automated sleep scoring: development and comparison with human sleep scoring for healthy older men and women," *Sleep*, vol. 17, no. 8, pp. 711–717, 1994.

[32] M. Hanaoka, M. Kobayashi, and H. Yamazaki, "Automated sleep stage scoring by decision tree learning," in *IEEE EMBC*, Chicago, July 2000.

[33] J. Virkkala, R. Velin, S. Himanen, A. Varri, K. Muller, and J. Hasan, "Automatic sleep stage classification using two facial electrodes," in *IEEE EMBC*, Vancouver, August 2008.

[34] J. Virkkala, A. Värri, J. Hasan, S.-L. Himanen, and K. Müller, "Sleep stage classification with low complexity and low bit rate," in *IEEE EMBC*, Minnesota, September 2009.

[35] G. Garcia-Molina, F. Abtahi, and M. Lagares-Lemos, "Automated NREM sleep staging using the electro-oculogram: A pilot study," in *IEEE EMBC*, San Diego, August 2012.

[36] C. McPherson, K. Behbehani, J. Burk, and E. Lucas, "Characterization of sleep using bispectral analysis," in *IEEE EMBC*, Istanbul, October 2001.

[37] V. Swarnkar, U. R. Abeyratne, and C. Hukins, "Automatic estimation of macro-sleep-architecture using a single channel of EEG," in *ICIIS*, Sri Lanka, December 2009.

[38] S.-F. Liang, C.-E. Kuo, Y.-H. Hu, and Y.-S. Cheng, "A rule-based automatic sleep staging method," *J. Neurosci. Methods*, vol. 205, no. 1, pp. 169–76, 2012.

[39] S.-F. Liang, C.-E. Kuo, Y.-H. Hu, Y.-H. Pan, and Y.-H. Wang, "Automatic stage scoring of single-channel sleep EEG by using multiscale entropy and autoregressive models," *IEEE Trans. Instrum. Meas.*, vol. 61, no. 6, pp. 1649–1657, 2012.

[40] V. C. Figueroa Helland, A. Gapelyuk, A. Suhrbier, M. Riedl, T. Penzel, J. Kurths, and N. Wessel, "Investigation of an automatic sleep stage classification by means of multiscorer hypnogram," *Method Inform. Med.*, vol. 49, no. 5, pp. 467–472, 2010.

[41] T. Sanders, M. McCurry, and M. Clements, "Sleep stage classification with cross frequency coupling," in *IEEE EMBC*, Chicago, August 2014.

[42] L. A. Fraiwan, N. Y. Khaswaneh, and K. Y. Lweesy, "Automatic sleep stage scoring with wavelet packets based on single EEG recording," *World Acad. Sci. Eng. Technol.*, vol. 54, no. 1, pp. 485–488, 2009.

[43] L. Fraiwan, K. Lweesy, N. Khasawneh, H. Wenz, and H. Dickhaus, "Automated sleep stage identification system based on time-frequency analysis of a single EEG channel and random forest classifier," *Comput. Methods Programs Biomed.*, vol. 108, no. 1, pp. 10–19, 2012.

[44] M. Radha, G. Garcia-Molina, M. Poel, and G. Tononi, "Comparison of feature and classifier algorithms for online automatic sleep staging based on a single EEG signal," in *IEEE EMBC*, Chicago, August 2014.

[45] S. Khalighi, T. Sousa, D. Oliveira, G. Pires, and U. Nunes, "Efficient feature selection for sleep staging based on maximal overlap discrete wavelet transform and SVM," in *IEEE EMBC*, Boston, September 2011.

[46] D. B. Percival and A. T. Walden, *Wavelet Methods for Time Series Analysis*. Cambridge University Press, 2006.

[47] S. Khalighi, T. Sousa, and U. Nunes, "Adaptive automatic sleep stage classification under covariate shift," in *IEEE EMBC*, San Diego, September 2012.

[48] B. Koley and D. Dey, "An ensemble system for automatic sleep stage classification using single channel EEG signal," *Comput. Biol. Med.*, vol. 42, no. 12, p. 11861195, 2012.

[49] J. Kempfner, P. Jennum, H. B. D. Sorensen, J. A. E. Christensen, and M. Nikolic, "Automatic sleep staging: From young adults to elderly patients using multi-class support vector machine," in *IEEE EMBC*, Osaka, July 2013.

[50] S. Gudmundsson, T. P. Runarsson, and S. Sigurdsson, "Automatic sleep staging using support vector machines with posterior probability estimates," in *CIMCA*, Vienna, November 2005.

[51] T. Sousa, A. Cruz, S. Khalighi, G. Pires, and U. Nunes, "A two-step automatic sleep stage classification method with dubious range detection," *Comput. Biol. Med.*, vol. 59, no. 1, pp. 42–53, 2015.

[52] T. Lajnef, S. Chaibi, P. Ruby, P.-E. Aguera, J.-B. Eichenlaub, M. Samet, A. Kachouri, and K. Jerbi, "Learning machines and sleeping brains: Automatic sleep stage classification using decision-tree multi-class support vector machines," *J. Neurosci. Methods*, no. 0, 2015.

[53] V. Bajaj and R. B. Pachori, "Automatic classification of sleep stages based on the time–frequency image of EEG signals," *Comput. Methods Programs Biomed.*, vol. 112, no. 3, pp. 320–328, 2013.

[54] H.-T. Wu, R. Talmon, and Y.-L. Lo, "Assess sleep stage by modern signal processing techniques," *IEEE Trans. Biomed. Eng.*, vol. 62, no. 4, pp. 1159–1168, 2015.

[55] C.-S. Huang, C.-L. Lin, L.-W. Ko, S.-Y. Liu, T.-P. Su, and C.-T. Lin, "Knowledge–based identification of sleep stages based on two forehead electroencephalogram channels," *Front. Neurosci.*, vol. 8, no. 9, pp. 1–12, 2014.

[56] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain," *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[57] S. Roberts and L. Tarassenko, "Analysis of the sleep EEG using a multilayer network with spatial organisation," *IEE Proc-F*, vol. 139, no. 6, pp. 420–425, 1992.

[58] N. Schaltenbrand, R. Lengelle, M. Toussaint, R. Luthringer, G. Carelli, A. Jacqmin, E. Lainey, A. Muzet, and J. P. Macher, "Sleep stage scoring using the neural network model: comparison between visual and automatic analysis in normal subjects and patients," *Sleep*, vol. 19, no. 1, pp. 26–35, 1996.

[59] O. Pacheco and F. Vaz, "Integrated system for analysis and automatic classification of sleep EEG," in *IEEE EMBC*, Hong Kong, October 1998.

[60] H. Park, K. Pa, and D.-u. Jmn, "Hybrid neural-network and rule-based expert system for automatic sleep stage scoring," in *IEEE EMBC*, Chicago, July 2000.

[61] J. Y. Tian and J. Q. Liu, "Automated sleep staging by a hybrid system comprising neural network and fuzzy rule-based reasoning," in *IEEE EMBC*, Shanghai, September 2005.

[62] P. Piñero, P. Garcia, L. Arco, A. Álvarez, M. M. García, and R. Bonal, "Sleep stage classification using fuzzy sets and machine learning techniques," *Neurocomputing*, vol. 58, no. 1, pp. 1137–1143, 2004.

[63] T. Shimada, T. Shiina, and Y. Saito, "Detection of characteristic waves of sleep EEG by neural network analysis," *IEEE Trans. Biomed. Eng.*, vol. 47, no. 3, pp. 369–79, 2000.

[64] M. Schwaibold, T. Penzel, J. Schochlin, and A. Bolz, "Combination of AI components for biosignal processing application to sleep stage recognition," in *IEEE EMBC*, Istanbul, October 2001.

[65] C. M. Held, J. E. Heiss, P. A. Estévez, C. A. Perez, M. Garrido, C. Algarín, and P. Peirano, "Extracting fuzzy rules from polysomnographic recordings for infant sleep classification," *IEEE Trans. Biomed. Eng.*, vol. 53, no. 10, pp. 1954–62, 2006.

[66] E. Oropesa, H. L. Cycon, and M. Jobert, "Sleep stage classification using wavelet transform and neural network," *International computer science institute*, 1999.

[67] L. Zoubek, S. Charbonnier, S. Lesecq, A. Buguet, and F. Chapotot, "Feature selection for sleep/wake stages classification using data driven methods," *Biomed. Signal Process. Control*, vol. 2, no. 3, pp. 171–179, 2007.

[68] F. Ebrahimi, M. Mikaeili, E. Estrada, and H. Nazeran, "Automatic sleep stage classification based on EEG signals by using neural networks and wavelet packet coefficients," in *IEEE EMBC*, Vancouver, August 2008.

[69] R. K. Sinha, "Artificial neural network and wavelet based automated detection of sleep spindles, REM sleep and wake states," *J. Med. Sys.*, vol. 32, no. 4, pp. 291–299, 2008.

[70] B. Y. Kim and K. S. Park, "Automatic sleep stage scoring system using genetic algorithms and neural network," in *IEEE EMBC*, Chicago, July 2000.

[71] H. G. Jo, J. Y. Park, C. K. Lee, S. K. An, and S. K. Yoo, "Genetic fuzzy classifier for sleep stage identification," *Comput. Biol. Med.*, vol. 40, no. 7, pp. 629–634, 2010.

[72] S. Charbonnier, L. Zoubek, S. Lesecq, and F. Chapotot, "Self-evaluated automatic classifier as a decision-support tool for sleep/wake staging," *Comput. Biol. Med.*, vol. 41, no. 6, pp. 380–9, 2011.

[73] Y. Liu, L. Yan, B. Zeng, and W. Wang, "Automatic sleep stage scoring using Hilbert-Huang transform with BP neural network," in *ICBBE*, Chengdu, June 2010.

[74] H. Ma, B. Hu, M. Jackson, J. Yan, and W. Zhao, "A hybrid classification method using artificial neural network based decision tree for automatic sleep scoring," *World Acad. Sci. Eng. Technol.*, vol. 5, no. 7, pp. 279–284, 2011.

[75] Y. L. Hsu, Y. T. Yang, J. S. Wang, and C. Y. Hsu, "Automatic sleep stage recurrent neural classifier using energy features of EEG signals," *Neurocomputing*, vol. 104, no. 1, pp. 105–114, 2013.

[76] N. McGrogan, E. Braithwaite, and L. Tarassenko, "Biosleep: a comprehensive sleep analysis system," in *IEEE EMBC*, Istanbul, October 2001.

[77] Q. Sun and J. Cheng, "Novel method of fast automated discrimination of sleep stages," in *IEEE EMBC*, Cancun, September 2003.

[78] M. E. Tagluk, N. Sezgin, and M. Akin, "Estimation of sleep stages by an artificial neural network employing EEG, EMG and EOG," *J. Med. Sys.*, vol. 34, no. 4, pp. 717–725, 2010.

[79] G. Becq, S. Charbonnier, F. Chapotot, A. Buguet, L. Bourdon, and P. Baconnier, "Comparison between five classifiers for automatic scoring of human sleep recordings," *Classification and Clustering for Knowledge Discovery*, pp. 113–127, 2005.

[80] S. Gabran, S. Zhang, M. Salama, R. Mansour, and C. George, "Real-time automated neural-network sleep classifier using single channel EEG recording for detection of narcolepsy episodes," in *IEEE EMBC*, Vancouver, August 2008.

[81] R. Agarwal and J. Gotman, "Long-term EEG compression for intensive-care settings," *IEEE Eng. Med. Biol. Mag.*, vol. 20, no. 5, pp. 23–29, 2001.

[82] P. Van Hese, W. Philips, J. De Koninck, R. Van de Walle, and I. Lemahieu, "Automatic detection of sleep stages using the EEG," in *IEEE EMBC*, Istanbul, October 2001.

[83] A. Flexer, P. Sykacek, I. Rezek, and G. Dorffner, "An automatic, continuous and probabilistic sleep stager based on a hidden markov model," *Appl. Artif. Intell.*, vol. 16, no. 3, pp. 199–207, 2002.

[84] A. Flexer, G. Gruber, and G. Dorffner, "A reliable probabilistic sleep stager based on a single EEG signal," *Artif. Intell. Med.*, vol. 33, no. 3, pp. 199–207, 2005.

[85] L. G. Doroshenkov, V. A. Konyshev, and S. V. Selishchev, "Classification of human sleep stages based on EEG processing using hidden markov models," *Biomed. Eng.*, vol. 41, no. 1, pp. 24–28, 2007.

[86] S.-T. Pan, C.-E. Kuo, J.-H. Zeng, and S.-F. Liang, "A transition-constrained discrete hidden markov model for automatic sleep staging," *Biomed. Eng. Online*, vol. 11, no. 1, pp. 1–19, 2012.

[87] F. Yaghouby, P. Modur, and S. Sunderam, "Naive scoring of human sleep based on a hidden markov model of the electroencephalogram," in *IEEE EMBC*, Chicago, August 2014.

[88] G. Garcia-Molina, M. Bellesi, S. Pastoor, S. Pfundtner, B. Riedner, and G. Tononi, "Online single EEG channel based automatic sleep staging," *Engineering Psychology and Cognitive Ergonomics. Applications and Services*, vol. 8020, pp. 333–342, 2013.

[89] U. Malinowska, H. Klekowicz, A. Wakarow, S. Niemcewicz, and P. J. Durka, "Fully parametric sleep staging compatible with the classical criteria," *Neuroinformatics*, vol. 7, no. 4, pp. 245–253, 2009.

[90] S. Günes, K. Polat, and S. Yosunkaya, "Efficient sleep stage recognition system based on EEG signal using k-means clustering based feature weighting," *Expert Sys. Appl.*, vol. 37, no. 12, pp. 7922–7928, 2010.

[91] E. A. Vivaldi and A. Bassi, "Frequency domain analysis of sleep EEG for visualization and automated state detection," in *IEEE EMBC*, New York, September 2006.

[92] E. A. Vivaldi, A. Bassi, J. Diaz, and N. Duque, "Visualization and clustering of sleep states in a frequency domain feature space," in *IEEE EMBC*, Buenos Aires, September 2010.

[93] M. Längkvist, L. Karlsson, and A. Loutfi, "Sleep stage classification using unsupervised feature learning," *Adv. Artif. Neyral Sys.*, pp. 1–9, 2012.

[94] J. L. Rodriguez-Sotelo, A. Osorio-Forero, A. Jimenez-Rodriguez, D. Cuesta-Frau, E. Cirugeda-Roldan, and D. Peluffo, "Automatic sleep stages classification using EEG entropy features and unsupervised pattern analysis techniques," *Entropy*, vol. 16, no. 12, pp. 6573–6589, 2014.

[95] S. Kubicki, L. Höller, I. Berg, C. Pastelak-Price, and R. Dorow, "Sleep EEG evaluation: a comparison of results obtained by visual scoring and automatic analysis with the Oxford sleep stager," *Sleep*, vol. 12, no. 2, pp. 140–149, 1989.

[96] R. Ferri, P. Ferri, R. M. Colognola, M. A. Petrella, S. A. Musumeci, and P. Bergonzi, "Comparison between the results of an automatic and a visual scoring of sleep EEG recordings," *Sleep*, vol. 12, no. 4, pp. 354–362, 1989.

[97] D. P. White and T. J. Gibb, "Evaluation of a computerized polysomnographic system," *Sleep*, vol. 21, no. 2, pp. 188–196, 1998.

[98] P. Anderer, G. Gruber, S. Parapatics, and G. Dorffner, "Automatic sleep classification according to Rechtschaffen and Kales," in *IEEE EMBC*, Lyon, August 2007.

[99] M. Andrle and L. Rebollo-Neira, "Cardinal B-spline dictionaries on a compact interval," *Appl. Comput. Harmon. Anal.*, vol. 18, no. 3, pp. 336–346, 2005.

[100] P. Anderer, A. Moreau, M. Woertz, M. Ross, G. Gruber, S. Parapatics, E. Loretz, E. Heller, A. Schmidt, M. Boeck, D. Moser, G. Kloesch, B. Saletu, G. M. Saletu-Zyhlarz, H. Danker-Hopfe, J. Zeitlhofer, and G. Dorffner, "Computer-assisted sleep

classification according to the standard of the American Academy of sleep medicine: Validation study of the AASM version of the Somnolyzer 24 x 7," *Neuropsychobiology*, vol. 62, no. 4, pp. 250–264, 2010.

[101] S. D. Pittman, M. M. MacDonald, R. B. Fogel, A. Malhotra, K. Todros, B. Levy, A. B. Geva, and D. P. White, "Assessment of automated scoring of polysomnographic recordings in a population with suspected sleep-disordered breathing," *Sleep*, vol. 27, no. 7, pp. 1394–1403, 2004.

[102] V. Svetnik, J. Ma, K. A. Soper, S. Doran, J. J. Renger, S. Deacon, and K. S. Koblan, "Evaluation of automated and semi-automated scoring of polysomnographic recordings from a clinical trial using zolpidem in the treatment of insomnia," *Sleep*, vol. 30, no. 11, pp. 1562–1574, 2007.

[103] J. Caffarel, G. J. Gibson, J. P. Harrison, C. J. Griffiths, and M. J. Drinnan, "Comparison of manual sleep staging with automated neural network-based analysis in clinical practice," *Med. Biol. Eng. Comput.*, vol. 44, no. 1, pp. 105–110, 2006.

[104] C. Berthomier, X. Drouot, M. Herman-Stoïca, P. Berthomier, J. Prado, D. Bokar-Thire, O. Benoit, J. Mattout, and M.-P. D'Ortho, "Automatic analysis of single-channel sleep EEG: validation in healthy individuals," *Sleep*, vol. 30, no. 11, pp. 1587–95, 2007.

[105] R. Kaplan, Y. Wang, K. Loparo, M. Kelly, and R. Bootzin, "Performance evaluation of an automated single–channel sleep–wake detection algorithm," *Nat. Sci. Sleep*, vol. 6, no. 1, pp. 113–122, 2014.

[106] J. R. Shambroom, S. E. Fábregas, and J. Johnstone, "Validation of an automated wireless system to monitor sleep in healthy adults," *J. Sleep Res.*, vol. 21, no. 2, pp. 221–30, 2012.

[107] A. Malhotra, M. Younes, S. T. Kuna, R. Benca, C. A. Kushida, J. Walsh, A. Hanlon, B. Staley, A. I. Pack, and G. W. Pien, "Performance of an automated polysomnography scoring system versus computer-assisted manual scoring," *Sleep*, vol. 36, no. 4, pp. 573–82, 2013.

[108] European Data Format. (2013) Home page. [Online]. Available: http://www.edfplus.info/.

[109] Compumedics Limited. (2013) Compumedics limited: Home page. [Online]. Available: http://www.compumedics.com/.

[110] Cleveland Medical Devices Inc. (2013) Home sleep testing, psg systems, sleep apnea monitoring, wireless polysomnography. [Online]. Available: http://www.clevemed.com/.

[111] MEDATEC. (2013) Medatec: Home page. [Online]. Available: http://www.medatec.be/.

[112] Micromed. (2013) Home page. [Online]. Available: http://www.micromed.eu/.

[113] Embla. (2013) Embla :: Home. [Online]. Available: http://www.embla.com/.

[114] Natus Medical Incorporated. (2013) Natus medical incorporated home page. [Online]. Available: http://www.natus.com/.

[115] Grass Technologies (Natus). (2013) Home page. [Online]. Available: http://www.grasstechnologies.com/.

[116] CareFusion. (2013) Home page. [Online]. Available: http://www.carefusion.co.uk/.

[117] SOMNOmedics. (2013) Home: Somnomedics. [Online]. Available: http://www.somnomedics.eu/.

[118] Cadwell Laboratories Inc. (2013) Home page. [Online]. Available: http://www.cadwell.com/.

[119] TEMEC Instruments B.V. (2013) Home page. [Online]. Available: http://www.temec.com/.

[120] Bitmed. (2013) Sibel group. [Online]. Available: http://www.sibelgroup.com/.

[121] Sleep Virtual. (2013) Home. [Online]. Available: http://sleepvirtual.com/index-s.html.

[122] Grass Technologies. (2015) TREA ambulatory EEG system. [Online]. Available: http://www.grasstechnologies.com/products/clinsystems/trea2.html

[123] g.tec. (2015) g.nautilus wireless biosignal acquisition. [Online]. Available: http://www.gtec.at/Products/Hardware-and-Accessories/g.Nautilus-Specs-Features

[124] Neuroelectronics. (2015) Enobio32. [Online]. Available: http://www.neuroelectrics.com/products/enobio/enobio-32/

[125] BIOPAC Systems Inc. (2015) Mobita wireless EEG system. [Online]. Available: http://www.biopac.com/Mobita-EEG-System

[126] Cognionics. (2015) 72-channel dry EEG headset system. [Online]. Available: http://www.cognionics.com/index.php/products/hd-eeg-systems/64-channel-system

[127] Advanced Brain Monitoring. (2015) Stat x-series mobile EEG. [Online]. Available: http://www.advancedbrainmonitoring.com/neurotechnology/medical-eeg/

[128] ——. (2013) Home page. [Online]. Available: http://www.advancedbrainmonitoring.com/.

[129] Zeo Sleep Manager. (2013) Home page. [Online]. Available: http://www.myzeo.com/sleep/.

[130] Emotiv. (2015) EPOC+. [Online]. Available: http://www.emotiv.com

[131] Michele Sleep Scoring. (2013) Home page. [Online]. Available: http://www.michelesleepscoring.com/.

[132] Aseega. (2013) Aseega online - sleep scoring-aid service. [Online]. Available: https://www.aseegaonline.com/pub/index.html.

[133] Philips. (2013) Somnolyzer 24x7. [Online]. Available: http://www.healthcare.philips.com/main/homehealth/sleep/somnolyzer/.

[134] n2Sleep. (2013) Home page. [Online]. Available: http://www.n2sleep.com/.

[135] WideMed. (2013) Morpheus - automated sleep testing management. [Online]. Available: http://www.widemed.com/.

[136] Bluetooth Technology. (2015) Bluetooth Low Energy. [Online]. Available: http://www.bluetooth.com/Pages/Bluetooth-Smart.aspx

3 Performance assessment of automatic sleep staging algorithms

The research presented within this chapter is an edited version of research previously published in:

S. A. Imtiaz and E. Rodriguez-Villegas, "Recommendations for performance assessment of automatic sleep staging algorithms," in proceedings of the 36th international conference of the IEEE Engineering in Medicine and Biology Society, Chicago, August 2014, pp. 5044–5047, © *IEEE.*

3.1 Introduction

There have been a large number of automatic sleep scoring algorithms published in the last four decades. This is because visual analysis of PSG signals is a costly, tedious and error-prone task. It can take between 2-4 hours to analyse an overnight PSG recording [1] with the scoring agreement between different experts about 82% on average [2]. Therefore, automation of this analysis is desirable not only to save time and costs but also to improve uniformity between different scoring sessions and experts. Automatic sleep staging has always been an actively growing research area. However, the recent consumer focus on wearable devices for sleep tracking has further accelerated research in this area resulting in the use of other signals such as heart rate variability, body movements, etc., that are not conventionally used for the classification of sleep stages. Further, there is also a push towards using the least number of sensors for scoring sleep. For example, the use of single channel EEG or EOG and classification based on respiratory signals exclusively have received recent attention. Unsurprisingly, the number of research papers published in this area has increased steadily over the past few years. Figure 3.1 charts this rise for papers indexed in *IEEEXplore* that present features and/or methods for automatic classification of at least one stage of sleep.

Despite the existence of a large number of automatic sleep staging algorithms (reviewed in Chapter 2), a direct comparison between them is extremely difficult due to a number of reasons. This includes the use of varying performance metrics, such as defining the agreement rate of an algorithm as well as the accuracies in each sleep stage in many different ways. Further, there are instances when certain sleep stages, e.g. REM and N1 or N1 and N2, are lumped together when reporting the results. This can mask the true

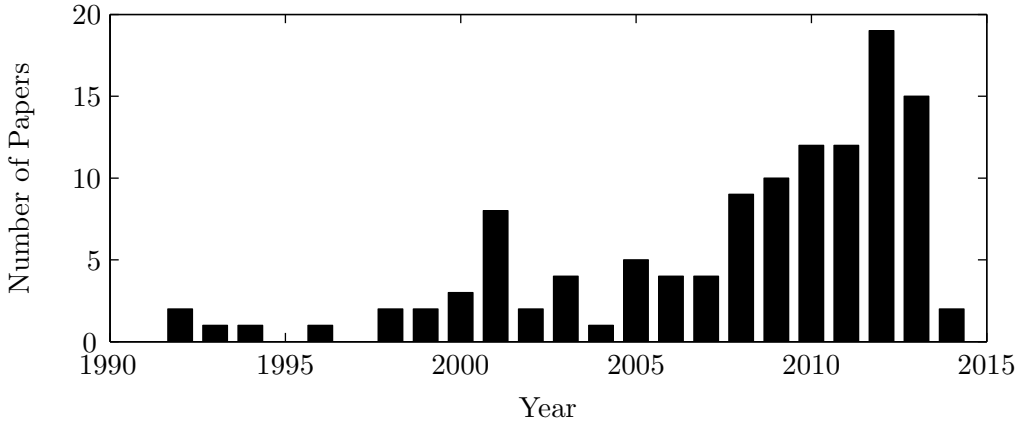


Figure 3.1: *Number of papers published in IEEEXplore over the last 25 years related to automatic sleep staging algorithms.*

accuracy of those sleep stages. The use of both R&K and AASM classification as well as different databases for evaluating the algorithm performance also makes it difficult to compare the results directly. Finally, the selection of limited or partial test signals from the same database also contribute to this problem.

It is, therefore, imperative to have a standard set of guidelines using which an algorithm is tested and its performance reported. This chapter discusses a set of recommendations and performance metrics to promote uniform testing and direct comparison of different algorithms. It describes some readily available polysomnography databases that are available at no cost and therefore accessible to all researchers. It proposes a set of guidelines for the usage of these databases so that consistent sections of data are being used across the board. It further shows how adopting these usage guidelines and uniform performance metrics would allow fair comparison of the strengths and weaknesses of sleep staging algorithms. Later in the chapter, the use of proposed recommendations and performance metrics is demonstrated with a simple sleep staging algorithm using data from two different polysomnography databases. It is illustrated how seemingly similar results using two different databases can have contrasting accuracies in different sleep stages and how selection of different training and test subjects from the same database can alter the final performance results.

3.2 Polysomnography databases

This section introduces five PSG databases that are widely available and can be used to develop and test sleep staging algorithms. Further details about these databases can be found in Appendix A.

3.2.1 PhysioNet Sleep EDF Database

The PhysioNet Sleep EDF database [3], [4] was made available online over 10 years ago and many algorithms have reported their detection performance using certain sections of this database. It consists of PSG recordings from 8 subjects, of which four were recorded overnight (cases starting with ST) while the others were recorded during a 24-hour period (starting with SC). All the recordings in this database include hypnograms scored using R&K classification.

3.2.2 PhysioNet Sleep EDF Expanded Database

This is the superset of the previously described PhysioNet Sleep EDF Database and has recently been published in full [5]. It consists of 61 subjects, including some with overnight recordings and others with up to 24 hours of recordings, scored using R&K classification.

3.2.3 DREAMS Subjects Database

This database from University of MONS — TCTS Laboratory and Université Libre de Bruxelles — CHU de Charleroi Sleep Laboratory consists of overnight PSG recordings of 20 subjects [6]. It includes hypnograms for each subject that have been scored using both R&K and AASM classification of sleep stages.

3.2.4 DREAMS Patients Database

This dataset, also from the same source as above, has 27 PSG recordings of subjects with various sleep disorders including insomnia, PLMS and others [7]. It also contains hypnograms that have been scored using both R&K and AASM classification of sleep stages.

3.2.5 Montreal Archive of Sleep Studies

This is an open-access database and currently offers 200 PSG recordings and consists of sleep scoring annotations using both R&K and AASM rules of classification [8].

3.3 Recommendations for using PSG databases

An algorithm's performance may be reported on either of the databases listed in the previous section since they are available online and free of cost. It is impossible to suggest or recommend one particular database to use for testing sleep staging algorithms. In fact, ideally a method should be tested on all of them.

It should be noted that the PhysioNet Sleep EDF database has been the most popular amongst all that have been listed. This database has now been made deprecated and its

use should be discouraged. Further, since the new PhysioNet database is a superset of the previous, the same records will already be included in it. However, if the intention is to compare against a method that used the previous database there is no reason not to use the older database.

Regardless of the chosen database, enough details about how it has been used should be provided so that the results could be reproduced by other researchers. This section lists and explains a minimum set of recommendations that is proposed to be followed in conjunction with the publicly available databases (even including those are not listed above). These would simplify the comparison and reproduction of results leading to improvement in the algorithms already published.

3.3.1 Classification: AASM and R&K

The AASM classification of sleep stages was published in 2007 and until then all sleep staging algorithms, naturally, reported their performance using the R&K classification. The adoption of R&K classification is so widespread that it is still in use in many clinics as well as some recent research publications. The major reason for publications still using the R&K instead of the AASM classification is that the PSG databases they have were scored before 2007 using the former classification. Since AASM classification is the newer standard, overcomes some of the limitations in the R&K classification [9], [10] and will eventually replace the R&K rules completely, it is recommended to use this for all future sleep staging algorithms.

From the databases listed in Section 3.2, the PhysioNet databases include hypnograms with the R&K classification while others have hypnograms scored using AASM rules. However, PhysioNet databases are the most popular ones and are used widely. To report results using these databases according to the AASM classification, care must be taken not to ignore epochs from stages which are not part of the AASM classification. In most publications MT stage (movement) of R&K is ignored when using the AASM classification to present results. This can lead to incorrect or biased results since major body movements commonly transition to wakefulness [10]. The AASM manual of classification states that an epoch with major body movements should be classified as Wake if alpha rhythms are present in the epoch or if a Wake epoch precedes or follows the epoch under analysis even if there are no alpha rhythms. If neither of the two conditions are true, then the epoch should be assigned the same sleep stage as the epoch that follows it [11].

To roughly *convert* a R&K hypnogram to AASM, S3 and S4 stages should be marked as N3 while Wake and MT together should be marked as Wake (as shown in Table 3.1). If using either of the two DREAMS or Montreal Archive of Sleep Studies (MASS) databases, the accompanying AASM hypnogram should be used without any need of conversion from R&K.

Table 3.1: *Conversion from R&K to AASM classification*

R&K	S1	S2	S3 S4	REM	Wake MT
AASM	N1	N2	N3	REM	Wake

3.3.2 Epoch size and signal duration

The standard epoch size for scoring of sleep stages according to both R&K and AASM classifications is 30 seconds. Some scorers and algorithms have also used different epoch sizes in the past with 20 seconds being a popular choice in some sleep labs. PhysioNet database includes hypnograms with standard 30s epoch size while the two DREAMS databases listed earlier have been scored at a non-standard interval of 5 s.

If the DREAMS databases are used then it is recommended to convert the hypnogram into 30s epoch scoring size using the following method. Starting from time zero, each 30s epoch will have six scores in the original hypnogram for every block of 5s. The modal value of these six scores should be determined and assigned as sleep stage of the 30 second epoch. There may be some epochs with equal number of different sleep stages assigned to the constituent subepochs. This results in a tie between the sleep stages making it impossible to assign the modal value. In such cases, the sleep score of the preceding epoch should be assigned to the current epoch. Additionally, in cases where the last or first scored epoch of the complete recording has a duration of less than 30 seconds it should be removed. In other words, partial epochs towards the end and beginning of the recording should not be analysed and the total signal duration should be a multiple of 30 (epoch size).

3.3.3 Selecting data from long term recordings

The PhysioNet databases consist of recordings from two different studies. The recordings prefixed with *ST* are overnight sleep recordings while a person is in bed. The other set, prefixed with *SC*, consist of 24-hour recordings from each subject including the day time as well as their overnight sleep. To use this latter set, most of the wake sections during the day is usually removed to select only overnight sleep data. However, this selection of data is not consistent as some research groups use data from the start of sleep removing all of the pre-sleep wake sections while others include greater periods of wake. It is possible that an algorithm using data with a lot more wake sections is suffering performance loss or reporting high overall accuracy only because of the higher number of wake epochs. Therefore, it is important to use consistent sections of data to better reflect the performance of an algorithm and also to make a fair comparison between the results of different algorithms.

For selecting the signals, it is proposed that the *lights off* time should be used as the start time for these longer recordings. In cases where this is not available, 15 minutes

of wake period prior to the first scored sleep epoch should be used. Similarly, to mark the end of a recording *lights on* time should be used, if available. Otherwise, 15 minutes of wake period after the last scored sleep epoch should be used as the end time. This selection of data is not required for DREAMS Subjects and Patients databases as they contain only the overnight recordings.

3.3.4 Training and test set

Most algorithms split the database into two sets: a training set for learning and a test set for performance validation. However, this split is often not clearly described and can have a big impact on the performance. It is, therefore, important to know how the database is split and which recordings were used for training and which ones for testing. It is difficult to reproduce the results of an algorithm without this knowledge, therefore the subjects used in each set should be clearly stated.

3.3.5 Unscored epochs

All the databases listed in Section 3.2 have some epochs that were not assigned any of the known sleep stages. These epochs are considered unscored and it is proposed to remove them from the results when reporting the performance. Further, the number of unscored epochs removed should be stated to bring it to the attention of other researchers.

3.3.6 Channels

Some sleep staging algorithms use a combination of multiple EEG, EOG and EMG channels while others use only a subset of these. It should always be ensured that the channel(s) being used by an algorithm are clearly specified.

3.4 Performance metrics

The overall performance of an algorithm is commonly represented by its *accuracy*, that is, the fraction of epochs correctly classified by the algorithm.

$$Accuracy = \frac{no.\ of\ true\ detections}{total\ no.\ of\ epochs} \tag{3.1}$$

However, not all stages of sleep occur for similar periods of time and the individual detection performances during each stage may vary considerably. Most papers present a further confusion matrix (or a contingency table) that provides details of the epochs correctly and incorrectly classified. Along with this, the fraction and rate of correctly detected epochs should also be computed for each sleep stage to give a better understanding of an algorithm's performance. For a hypothetical sleep stage X, the following terms represent the epochs that are either correctly or falsely detected/rejected.

True Positives (TP): Number of epochs correctly scored as X.

False Positives (FP): Number of epochs incorrectly scored as X.

True Negatives (TN): Number of epochs correctly rejected as not X.

False Negatives (FN): Number of epochs incorrectly rejected as not X.

The performance of an algorithm can then be characterised for each sleep stage by calculating the following metrics.

Sensitivity

It represents the fraction of correctly detected epochs in a sleep stage X, where X can be any of the five sleep stages.

$$Sensitivity = \frac{true\ positives\ in\ stage\ X}{true\ positives\ in\ stage\ X + false\ negatives\ in\ stage\ X} \tag{3.2}$$

Selectivity

It refers to the proportion of true detections of X amongst the epochs classified by the algorithm and is also known as the true positive rate.

$$Selectivity = \frac{true\ positives\ in\ stage\ X}{true\ positives\ in\ stage\ X + false\ positives\ in\ stage\ X} \tag{3.3}$$

Specificity

It is the measure of an epoch of stage other than X being correctly rejected by the algorithm.

$$Specificity = \frac{true\ negatives\ in\ stage\ X}{true\ negatives\ in\ stage\ X + false\ positives\ in\ stage\ X} \tag{3.4}$$

Discussion

Of the performance metrics described above, the *accuracy* of an algorithm is often its highlight performance number. The *sensitivity* and *selectivity* in each stage are important to determine how good the algorithm is in detecting the epochs in each sleep stage correctly. The *specificity* of a sleep stage is useful only in the context when the detection of a specific sleep stage is desired while rejecting epochs of other sleep stages. It does not give any meaningful information in the context of a complete sleep staging algorithm.

3.5 Demonstration of the performance assessment recommendations using a sleep staging algorithm

In this section an automatic sleep staging algorithm is presented and its performance is characterised using two PSG databases: PhysioNet EDF and DREAMS Subjects databases. The signals from these databases are used by following the recommendations in Section 3.3. Three cases are used to illustrate how different databases and different subjects from the same database can affect the performance results.

Case 1: Performance assessment using DREAMS Subjects Database.

Case 2: Performance assessment using PhysioNet Sleep EDF Database.

Case 3: Performance assessment using different training and test subjects from PhysioNet Sleep EDF Database.

Algorithm Overview

The algorithm uses data from one EEG (frontal) and one EOG channel which are split into epochs of 30 seconds. Each epoch is further divided into 2-second blocks and transformed to frequency domain using Fast Fourier Transform (FFT). For each block of 2s EEG, spectral power in every 2 Hz frequency bin from 0-30 Hz range is calculated i.e. 0-2 Hz, 2-4 Hz, 4-6 Hz and so on. For the corresponding EOG block, spectral power within 0-6 Hz is also calculated similarly for every 2 Hz frequency interval. Subsequently, the average of every feature is calculated within a 30s epoch. Since each feature is calculated for a 2s block, there are 15 such values within an epoch to calculate the average. This results in 18 features overall (15 EEG and 3 EOG) computed for an epoch and are classified using a Support Vector Machine (SVM). The SVM is implemented using LIBSVM package [12] in MATLAB (ver. R2010a) with a third degree radial basis kernel function.

3.5.1 Case 1: Using DREAMS Subjects Database

In this case, data from channels Fp1-A2 and EOG1 is used from the DREAMS subjects database. It is partitioned such that subjects 1-10 were used in the training set while subjects 11-20 formed the test set. The database includes hypnograms that were scored using the AASM classification with epoch size of 5 seconds. This is converted into a 30s scoring interval by using the modal value of the sleep score in every 30s epoch (as explained in Section 3.3.2). Further, it is ensured that the total duration of recording in each subject contains a whole number of 30s epochs discarding any remaining seconds at the end that formed an incomplete epoch. As a result, there are a total of 10178 epochs in the training set and 10087 epochs in the test set including 3 and 20 unscored epochs in the training and test sets respectively.

The algorithm achieved an overall accuracy of 82.7% on the training set. With the test data, the overall accuracy was found to be 77%. The confusion matrix for the algorithm performance on the test dataset is shown in Table 3.2. It shows that the sensitivity for stages W and N3 are more than 86% whereas only 17% of N1 epochs are correctly detected. This case illustrates how a high overall accuracy can easily mask the poor performance of the algorithm in one or more sleep stages.

Table 3.2: *Case 1: Results using DREAMS Subjects Database*

		REFERENCE						
ALGORITHM		**W**	**N1**	**N2**	**N3**	**R**	**Sen(%)**	**Sel(%)**
	W	1599	226	112	15	60	87.0	79.5
	N1	52	142	75	0	201	17.2	30.2
	N2	132	326	3340	249	156	82.5	79.5
	N3	9	5	334	1627	1	86.0	82.3
	R	47	126	187	0	1046	71.5	74.4

3.5.2 Case 2: Using Sleep-EDF Database

In this case data from all 8 subjects in the PhysioNet Sleep EDF database is used. It includes signals from Fpz-Cz and horizontal EOG channels. The database consists of two kinds of recordings (described in Section 3.2.1). The *ST* recordings are used *as is* while data from *SC* recordings is selected using the recommendation in Section 3.3.3. The eight subjects are partitioned to include two of each kind of recording in both the training and test dataset. The training set included SC4002, SC4102, ST7022, ST7121 and the test set included SC4012, SC4112, ST7052, ST7132. In total there are 8905 epochs (4650 in training and 4295 in test set) of which 1133 are unscored (589 in training and 544 in test set).

The overall accuracies achieved on the training and test sets are 79.5% and 73.4% respectively. This overall result appears to be very similar to that obtained in Case 1. However, the confusion matrix shown in Table 3.3 paints a different picture. The sensitivity in each sleep stage is actually quite different compared to the first case. In particular, the results show improved sensitivities in REM, N1 and N2 stages. There is a reduction in N3 sensitivity while the algorithm fails to classify any of the Wake epochs at all.

Table 3.3: *Case 2: Results using PhysioNet Sleep EDF Database*

		REFERENCE						
		W	**N1**	**N2**	**N3**	**R**	**Sen(%)**	**Sel(%)**
ALGORITHM	**W**	0	0	0	0	0	0	0
	N1	117	98	30	7	17	30.6	36.4
	N2	36	55	1562	83	25	85.1	88.7
	N3	72	9	166	392	24	80.0	59.1
	R	111	158	78	8	703	91.4	66.5

3.5.3 Case 3: Using Sleep-EDF Database with different training and test set

In this case the same database as in Case 2 is used with the difference that all four *SC* recordings are part of the training set (3929 epochs with no unscored epochs) while the other four *ST* recordings are part of the test set (5016 epochs including 1133 unscored epochs). An overall accuracy of 86.6% is achieved for the training set while the test set resulted in an overall accuracy of only 61.6%. The confusion matrix and individual sleep stage performances are shown in Table 3.4. In contrast to Case 2, the sensitivity in Wake stage is now close to 80% while in REM stage it has gone down from 91% to 19%. The overall accuracy is also less than that achieved in Case 2. This illustrates how using different groups of training and test cases from the same database can result in a vastly different performance result.

Table 3.4: *Case 3: Results using PhysioNet Sleep EDF Database with a different training and test set*

		REFERENCE						
		W	**N1**	**N2**	**N3**	**R**	**Sen(%)**	**Sel(%)**
ALGORITHM	**W**	265	178	14	4	134	79.6	44.5
	N1	1	17	2	0	61	5.4	21.0
	N2	43	102	1293	117	427	81.6	65.2
	N3	24	16	276	649	82	84.3	62.0
	R	0	5	0	0	164	18.9	97.0

3.6 Discussion

In this chapter, a set of guidelines and recommendations were proposed for using the common PSG databases that are freely available on the internet. To this end, five common databases were listed and explained. The recommendations being proposed here are not restricted to these databases only but will apply equally to other databases that may become available in future. Of all the databases listed, PhysioNet databases are the most popular. With the availability of the new PhysioNet Sleep EDF Expanded database, it is expected that this will be widely used by sleep researchers. An open source MATLAB toolbox has been developed as part of this research work to extract and use the signals from this database. The toolbox includes functions that implement the recommendations proposed in this chapter making it easy for everyone to adhere to these guidelines. This toolbox is explained in further detail in Appendix B.

To demonstrate how different databases can alter an algorithm's performance a sleep staging algorithm, based on spectral features and SVM classifier was used in this chapter. Evaluating the performance of this algorithm in three different cases, it was shown that the results can easily change when different databases are used or even if different set of training and test subjects are used from the same database. It was also shown that even if the classification accuracy using different databases is similar, the results of detection in each sleep stage can be very different. It is therefore important that the classification performance of an algorithm in each of the individual sleep stages is also reported.

In some instances, the only reported performances are for two-stage or three-stage classification where two stage is Sleep-Wake and three-stage is Sleep-REM-NREM. The overall accuracies of these algorithms cannot be directly compared with the overall accuracy of a complete sleep staging algorithm. Some algorithms classify only the sleep stages without including Wake while few also combine REM and N1 together. REM and N1 are similar in EEG but combining their detection performance does not give any meaningful information, since it is impossible to determine what fraction of REM and N1, if any, are being individually detected.

This chapter recommended using the AASM classification in all future work and explained how to roughly convert the hypnograms scored with R&K classification. It also described a method to convert non-standard epoch size hypnograms to the standard 30s scoring interval. It is hoped that the recommendations proposed in this chapter will allow researchers to fairly compare different methods subsequently leading to improvements in already existing sleep staging algorithms.

References

[1] M. Ronzhina, O. Janousek, J. Kolarova, M. Novakova, P. Honzik, and I. Provaznik, "Sleep scoring using artificial neural networks," *Sleep Med. Rev.*, vol. 16, no. 3, pp. 251–63, 2012.

[2] H. Danker-Hopfe, P. Anderer, J. Zeitlhofer, M. Boeck, H. Dorn, G. Gruber, E. Heller, E. Loretz, D. Moser, S. Parapatics, B. Saletu, A. Schmidt, and G. Dorffner, "Interrater reliability for sleep scoring according to the Rechtschaffen & Kales and the new AASM standard," *J. Sleep Res.*, vol. 18, no. 1, pp. 74–84, 2009.

[3] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, pp. e215–e220, 2000.

[4] PhysioNet. (2013) Sleep-EDF Database. [Online]. Available: http://www.physionet.org/physiobank/database/sleep-edf/.

[5] ——. (2014) Sleep-EDF Database [Expanded]. [Online]. Available: http://www.physionet.org/physiobank/database/sleep-edfx/.

[6] University of MONS - TCTS Laboratory. (2015) The DREAMS Subjects Database. [Online]. Available: http://www.tcts.fpms.ac.be/~devuyst/Databases/DatabaseSubjects/.

[7] ——. (2015) The DREAMS Patients Database. [Online]. Available: http://www.tcts.fpms.ac.be/~devuyst/Databases/DatabasePatients/.

[8] C. O'Reilly, N. Gosselin, J. Carrier, and T. Nielsen, "Montreal archive of sleep studies: an open-access resource for instrument benchmarking and exploratory research," *J. Sleep Res.*, vol. 23, no. 6, pp. 628–35, 2014.

[9] S.-L. Himanen and J. Hasan, "Limitations of Rechtschaffen and Kales," *Sleep Med. Rev.*, vol. 4, no. 2, pp. 149–167, 2000.

[10] M. Silber, S. Ancoli-Israel, M. Bonnet, S. Chokroverty, M. Grigg-Damberger, M. Hirshkowitz, S. Kapen, S. Keenan, M. Kryger, T. Penzel, M. Pressman, and C. Iber, "The visual scoring of sleep in adults," *J. Clin. Sleep Med.*, vol. 3, no. 2, pp. 121–31, 2007.

[11] C. Iber, S. Ancoli-Israel, A. Chesson, and S. Quan, *The AASM manual for the scoring of sleep and associated events: rules, terminology and technical specifications.* Westchester, IL: American Academy of Sleep Medicine, 2007.

[12] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

4 REM sleep detection using single channel EEG

The research presented within this chapter is an edited version of research previously published in:

S. A. Imtiaz and E. Rodriguez-Villegas, "A Low Computational Cost Algorithm for REM Sleep Detection Using Single Channel EEG," Annals of Biomedical Engineering, vol. 42, no. 11, pp. 2344–2359, 2014.

4.1 Introduction

Rapid Eye Movement (REM) is a distinct phase of human sleep that accounts for about 5-20% of an adult's entire night's sleep [1]. Its detection, both onset and duration, are very important for the diagnosis of certain sleep disorders including narcolepsy and REM behaviour disorder (RBD). Traditional clinical and home PSG systems are constrained by the minimum number of channels that can be used. Even if the number of EEG channels can potentially be reduced, the EOG and EMG channels are still required since identifying REM stage epochs involves observing the chin muscle and eye activity [2].

Other than its detection for being a significant fraction of sleep, analysis of REM sleep also helps to serve as an important marker for certain sleep disorders. Observing the muscle activity during REM stage is often used for the diagnosis of RBD, which is also an early marker for neurological disorders including Parkinson's disease [3]. The duration of REM sleep in the first cycle has been shown to correlate negatively with mood improvement on wake-up in patients with major depression [4]. It has also been shown that the number of REM sleep periods is higher, with a shorter average duration, in trauma-exposed people who go on to develop post-traumatic stress disorder [5]. The latency from Wake to the onset of first REM cycle and the pattern of occurrence of subsequent cycles throughout the night is commonly used in the diagnosis of narcolepsy. Vogel *et al.* [6] reported that REM sleep deprivation can be used therapeutically for the improvement of depression symptoms. Using a wearable REM sleep detection system, this could be achieved by raising an alarm to awaken the patients whenever they enter the REM sleep phase.

Apart from REM, all the stages of sleep can be identified from EEG channels only. This is because REM sleep has many electroencephalographic similarities with Wake and

N1 stages [1], [7], [8]. According to both R&K [9] and AASM [2] sleep scoring manuals, the presence of low amplitude, mixed frequency EEG is characteristic of both N1 and REM stages making its visual identification using EEG challenging. However, with most of the sleep stages identifiable with EEG it makes sense to attempt to score REM phases using the same signal to obviate the need of using extra electrodes. This could be very helpful in making HPSG systems easier and comfortable to use.

There are two main objectives of the work described in this chapter. The first is to find features and trends in sleep EEG that can distinguish REM phase from all other stages of sleep, particularly N1 and Wake. The second objective is to use these EEG features for developing a simple algorithm capable of detecting REM stage epochs. Both these objectives ultimately aid the development of a sleep staging algorithm that could be used as part of a truly wearable sleep system.

4.2 Literature review of REM detection algorithms

Several research groups have been working on automatic sleep staging using signals from PSG and EEG based systems. In this section a review of these methods is presented to show the different features and classifiers being used and their detection performance. The performances reported below are limited to the REM detection part of systems and their corresponding accuracy.

There are multiple signal processing methods for automatic identification of REM sleep alone or as part of wider sleep staging systems that already exist in literature. These algorithms tend to use a set of frequency and time-domain features extracted from one or more of EEG, EOG and EMG channels. These features are then classified using a variety of classifiers such as decision trees, LDA classifiers, fuzzy classifiers, support vector machines, artificial neural networks and more. These were discussed in detail earlier in Chapter 2, and a summary of their REM detection performance is shown in Table 4.1. Of all the methods listed in this table, less than half use EEG signals only for detection of REM (along with other sleep stages). Most methods use at least one EOG or EMG channels with the EEG channels for detection.

Estrada *et al.* [10] concluded that EMG and EOG are both important in sleep staging, particularly in REM stage. Similarly, Charbonnier *et al.* [11] reported a jump in REM detection accuracy from 63% to 83% when EMG signal was added to their analysis. It is evident from the sleep staging literature that algorithms using inputs from EEG, EOG and EMG channels are able to achieve a better REM detection performance while using just one EEG channel makes the task more challenging. Further, those that eventually achieve an impressive accuracy for REM detection using EEG only tend to use complex feature extraction and classification methods that are not suitable for a low power implementation on resource-constrained systems.

Table 4.1: *Literature review summary for automatic REM stage detection as part of sleep staging algorithms.*

Ref	Channels	Method	Result
[12]	2×EEG 2×EOG 1×EMG	Waveform recognition and rule-based classification	Sen: 76%
[13]	1×EOG	DFT features with decision tree classification	Sen: 62% Sel: 79%
[14]	1×EEG	Bispectrum estimation	Sen: 73%
[15]	1×EEG 2×EOG 1×EMG	Decision tree with power and energy features and contextual smoothing	Sen: 91% Sel: 85%
[16]	1×EEG	Multiscale entropy and autoregressive modelling	Sen: 95% Sel: 80%
[17]	6×EEG 2×EOG	MODWT features with SVM	Sen: 71% Sel: 91%
[18]	6×EEG 2×EOG	Decision tree and multiple SVMs	Sen: 93%
[19]	2×EEG 2×EOG 1×EMG	Decision trees and SVM	Sen: 97%
[20]	2×EEG	Time-frequency image representation with SVM	Sen: 85%
[21]	1×EEG 1×EOG 1×EMG	ANN and rule-based hybrid system	Sen: 85%
[22]	2×EEG 2×EOG 1×EMG	Pattern recognition with ANN	Sen: 79%
[23]	EEG EMG	ANN and fuzzy classifier with rule-based post-processing	Sen: 85% Sel: 95%
[24]	4×EEG 1×EOG 1×EMG	Neuro-fuzzy classifier with five input patterns	Sen: 72%
[25]	EEG	WPT coefficients with ANN	Sen: 65%

Table 4.1: *Literature review summary for automatic REM stage detection as part of sleep staging algorithms.*

Ref	Channels	Method	Result
[11]	1×EEG 1×EOG 1×EMG	ANN with 33 spectral, entropy and statistical features	Sen: 63-83%
[26]	1×EEG	ANN with relative power and power spectral density	Accuracy: 82%
[27]	2×EEG 2×EOG 1×EMG	k-means clustering	Sen: 73% Spe: 88%
[28]	1×EEG	Hidden Markov Model	Sen: 68% Sel: 50%
[29]	2×EEG	Hidden Markov Model	Sen: 86%
[30]	EEG EOG EMG	Hidden Markov Model	Sen: 90%
[31]	1×EEG	Hidden Markov Model	Sen: 85%
[32]	EEG EOG EMG	Matching pursuit and rule-based classification	Sen: 80%
[33]	1×EEG	Spectral features, k-means clustering and kNN classifier	Sen: 81%
[34]	1×EEG 1×EOG 1×EMG	Hidden Markov Model	Sen: 60%
[35]	2×EEG	Entropy features with unsupervised classification	Sen: 38%
[36]	1×EEG	Multiple spectral and temporal features with fuzzy classification and contextual smoothing	Sen: 83% Sel: 89%

4.3 Material

For studying different features, data from polysomnography recordings of healthy subjects in the DREAMS Subjects Database from University of MONS - TCTS Laboratory and Université Libre de Bruxelles - CHU de Charleroi Sleep Laboratory was used [37]. This database has twenty whole night recordings in EDF format including two EOG, three EEG (Fp1-A2, Cz-A1 and O1-A2) and one submental EMG channel for each subject. It is described in detail in Appendix A.

Before being used for any analysis, data from each EEG channel was first resampled to a sampling frequency of 256 Hz using the `resample` function in MATLAB. The signals were then filtered with a first order 0.16 Hz high pass filter to remove dc offset and bandlimited using a second order 50 Hz Butterworth low pass filter. The EEG data was split into 2-second long non-overlapping blocks (subepochs) and subsequently transformed to the frequency domain with a 512-point Fast Fourier Transform (FFT), hence obtaining a resolution of 0.5 Hz. The magnitude and frequency coefficients were then used to compute various features for REM detection in different frequency bands.

From the pool of twenty subjects, the first five (Subjects 01-05) were arbitrarily selected for data analysis, feature selection and training of the proposed algorithm. Subjects 06-20 were later used to test the performance of the algorithm without any parameter adjustment. The total number of epochs in Wake, REM and NREM stages for the training and test set are shown in Table 4.2.

Table 4.2: *The number of Wake, REM and NREM epochs in training and test datasets.*

		Number of Epochs		
	No. of Subjects	Wake	NREM	REM
Training	5	679	3573	798
Test	15	2880	10091	2221

Once the algorithm was finalised, its performance was evaluated using recordings from PhysioNet Sleep EDF database. This was done to compare against the performances of other algorithms in literature that have been evaluated on the same database. This database consists of eight subjects and is described in Appendix A.

4.4 Methodology

4.4.1 Frequency range of analysis

It has been shown that there exists some differences in spectral power during REM stage around a certain frequency range. Corsi-Cabrera *et al.* [7] studied the spectral power differences during REM, N1 and Wake stages. They reported similar N1 and REM spectral powers between 13 Hz and 17 Hz, higher N1 power in the 10-13 Hz band and lower N1 power between 1 Hz and 9 Hz. Uchida *et al.* [38] looked at the frequency bands potentially capable of distinguishing between REM and NREM sleep. They showed spectral power in REM to be lowest in the 12-16 Hz band when compared to NREM stages (except N1).

Since the 10-13 Hz band appears to be able to discriminate between REM and N1 while the 12-16 Hz band helps distinguishing REM from other stages, the analysis in this work was performed with more focus on these bands while also including other frequency bands. This is done to determine the best frequency range where the discriminatory ability of different features are most prominent. The frequency spectrum for REM and non-REM epochs in the 8-16 Hz range is shown in Figure 4.1 for all five training subjects. It can be seen that the spectral power during non-REM stages are higher than that during REM stages between about 9-15 Hz in all subjects while the values are similar at around 8 Hz and 16 Hz frequency. This ties in with the results in both [38] and [7]. Therefore the frequency range of 8-16 Hz is selected for analysis in this work.

The choice of the frequency band for the detection of REM stage epochs is primarily motivated by the discriminatory ability of the features within this band. Since the traditional sleep analysis is performed in the 0.5-50 Hz range any results obtained from a feature in the limited frequency band will also be compared against the performance of the same feature in the 0.5-50 Hz range.

4.4.2 Spectral Edge Frequency

Spectral Edge Frequency (*SEF*) is the frequency below which a certain fraction of the signal power is contained. It is generally written as *SEFxx* where *xx* is the fraction of signal power for which the edge frequency is calculated. An illustration of spectral edge frequency at 50% and 95% of the signal power is shown in Figure 4.2.

SEF can be useful in estimating how the spectral power is spread in a given frequency range. Spectral edge frequencies at 50% and 95% are the most common measures used in different applications. However they can be estimated at any percentage point depending on the information desired.

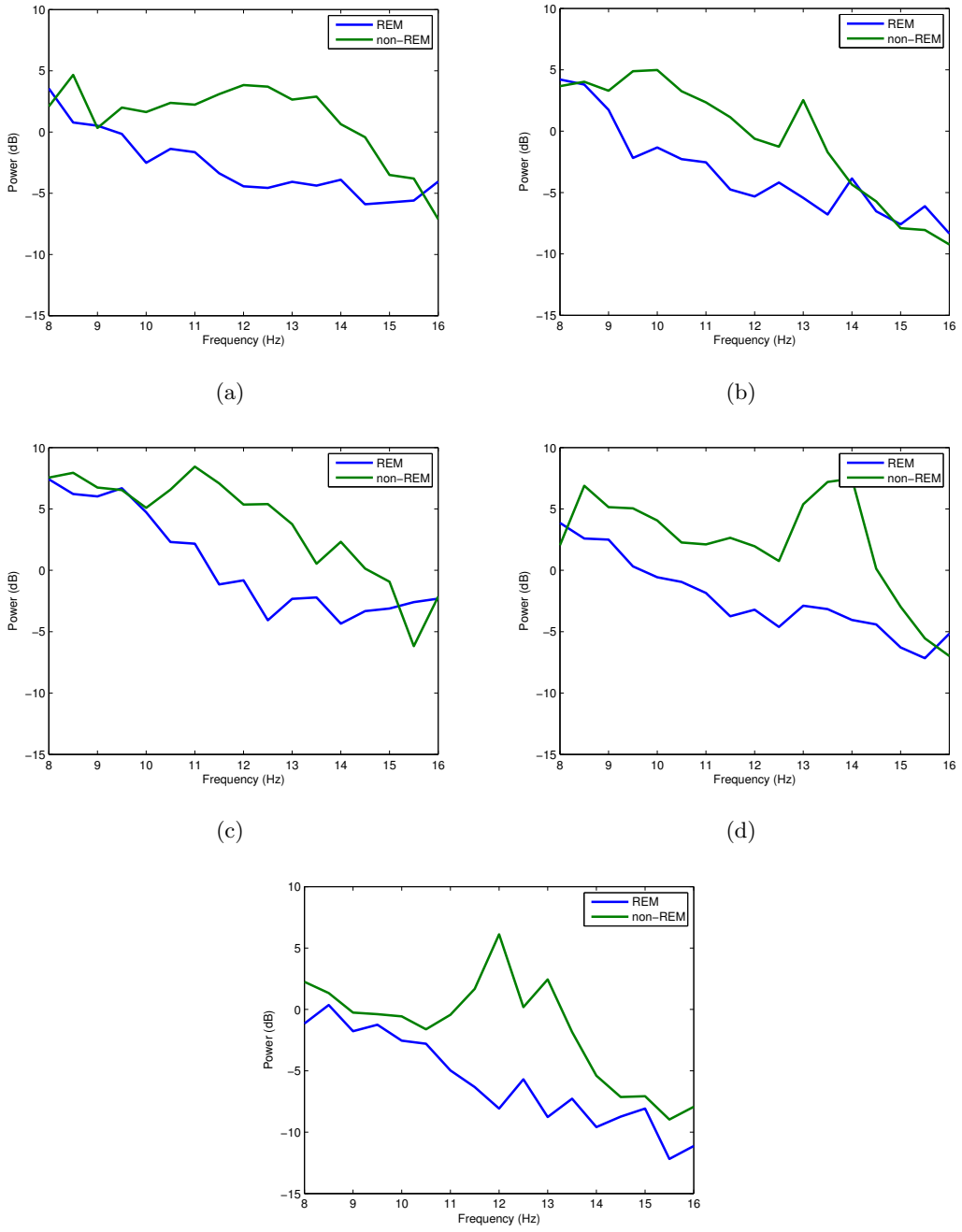


Figure 4.1: *Frequency spectrum of REM and non-REM epochs in 8-16 Hz range for different training subjects 01-05 on plots (a)-(e).*

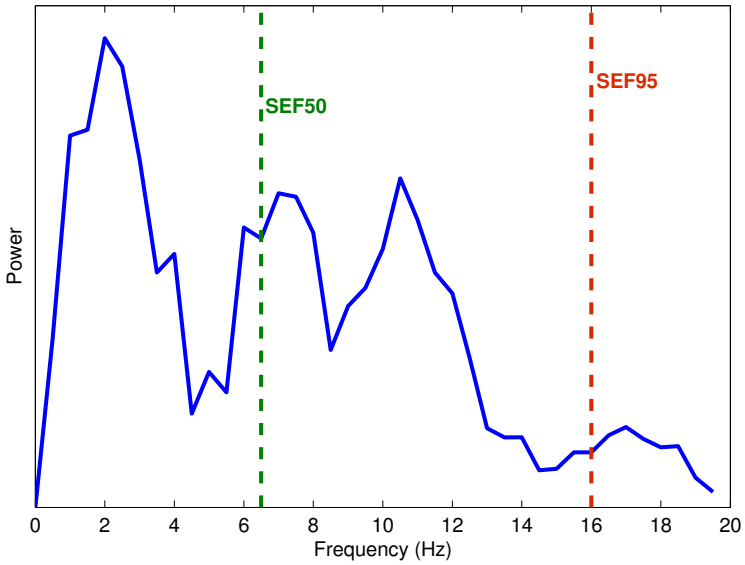


Figure 4.2: *An illustration of Spectral Edge Frequency (SEF) at 50% and 95% of the signal power in the 0-20 Hz frequency range.*

SEF50

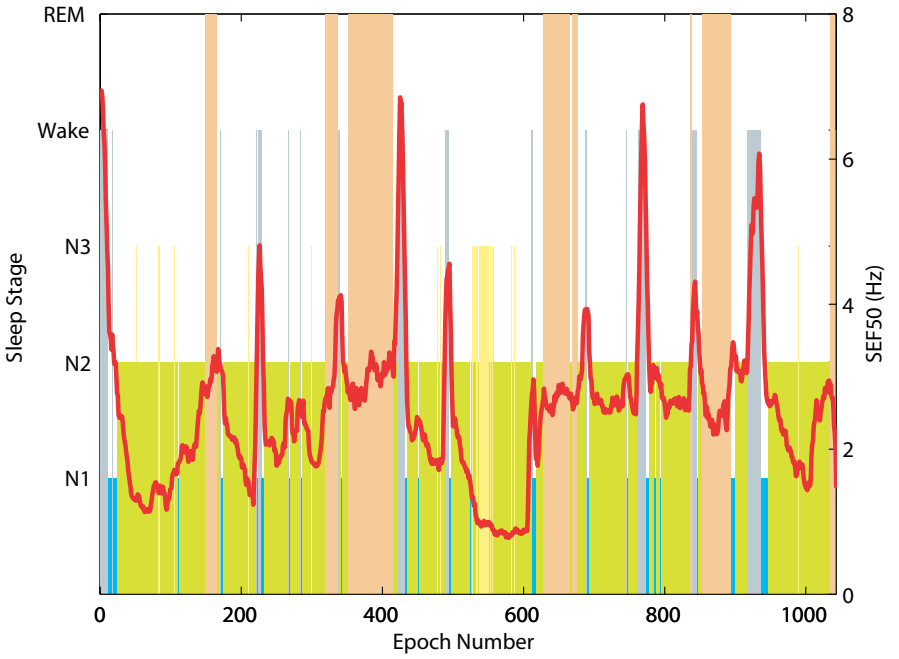
SEF at 50% (*SEF50*) is the lowest frequency below which half of the signal power is present and is equivalent to the median frequency of a signal. It is computed from the FFT coefficients as shown below, where n is the total number of FFT coefficients and x is the index to solve the equation for. The required frequency is then the x_{th} frequency from the array of FFT frequency components.

$$\sum_{i=1}^{x} |mag_i|^2 = 0.50 \times \sum_{i=1}^{n} |mag_i|^2 \tag{4.1a}$$

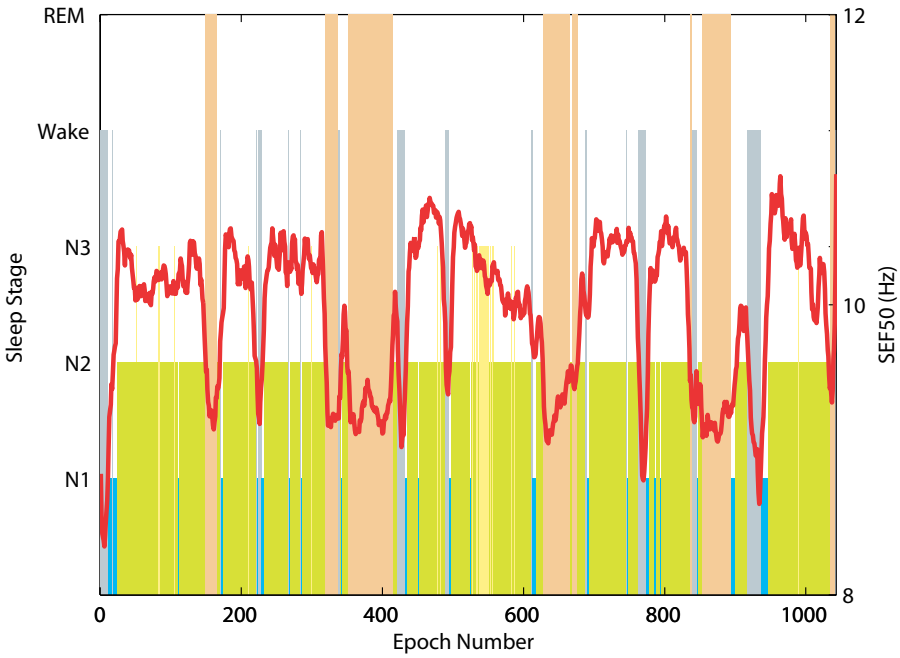
$$SEF50 = freq(x) \tag{4.1b}$$

Figure 4.3 shows the *SEF50* values for *Subject01* in 0.5-50 Hz and 8-16 Hz frequency bands for an overnight recording on top of the corresponding hypnogram. For an epoch e, its *SEF50* value is calculated by taking the mean from the fifteen 2-second subepochs that make up the epoch. A 9-point moving average filter is then applied to the final *SEF50* value which is then plotted with the hypnogram.

The graphs in Figure 4.3 show that during the REM stages, the *SEF50* values are observed to be amongst the lowest when calculated in the 8-16 Hz range in Figure 4.3(b). However, this is not the case in Figure 4.3(a) when the entire frequency range is used and the *SEF50* values during REM stages overlap with those during N2 stages.



(a)



(b)

Figure 4.3: *Hypnogram and SEF50 in the (a) 0.5-50 Hz and (b) 8-16 Hz band of the EEG signal for one training subject.*

SEF95

SEF at 95% (*SEF95*) is the lowest frequency below which 95% of the signal power is present. It is computed from the FFT coefficients, similar to the *SEF50* calculation.

$$\sum_{i=1}^{x} |mag_i|^2 = 0.95 \times \sum_{i=1}^{n} |mag_i|^2 \tag{4.2a}$$

$$SEF95 = freq(x) \tag{4.2b}$$

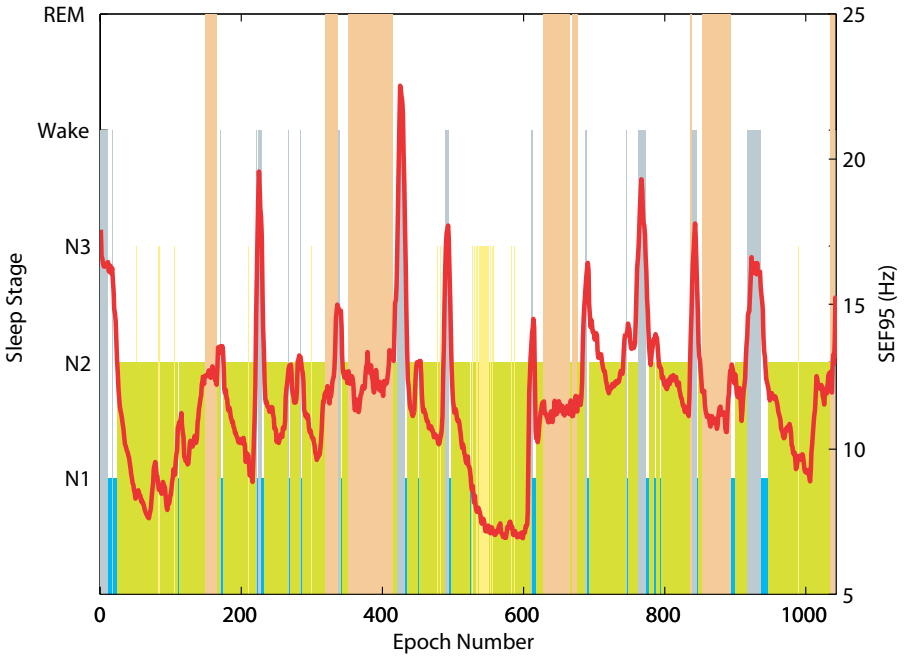
Figure 4.4 shows the *SEF95* values obtained in *Subject01* together with the hypnogram and have been calculated similar to *SEF50* by taking the mean of the values from the subepochs. The plots show how *SEF95* varies in different sleep stages for this subject in the two frequency ranges. The *SEF95* values in the 0.5-50 Hz analysis range during REM stages are neither highest nor lowest and stay close to the 12 Hz mark. In the 8-16 Hz range, however, *SEF95* values are usually highest during the REM stages.

SEFd

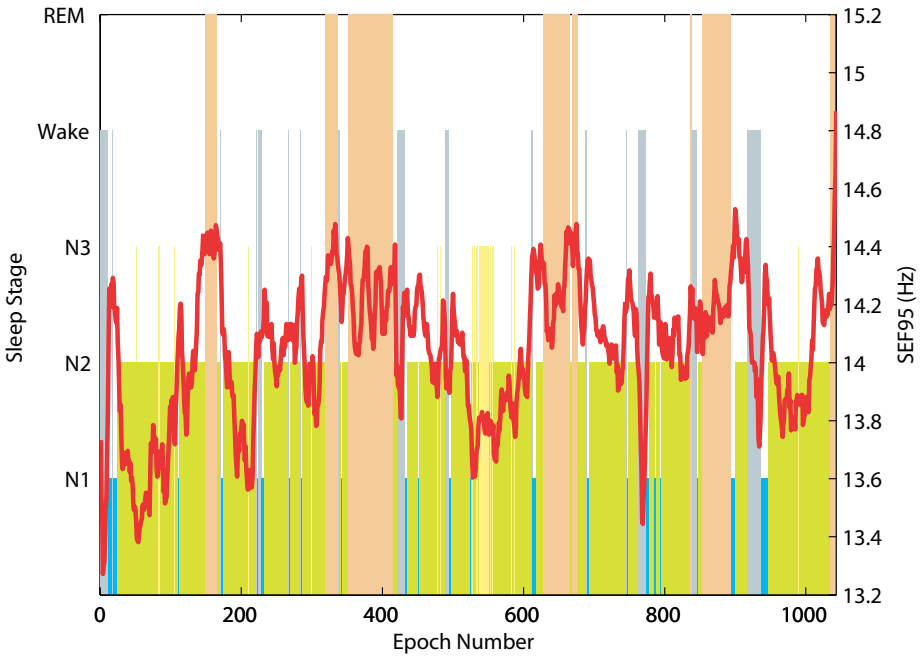
Both *SEF95* and *SEF50* features appear to have high and low values respectively during REM stages that can potentially be used for separating REM from other stages. These two features can be combined by taking their difference which would further amplify their discriminatory power during REM sleep. The difference between *SEF95* and *SEF50* is explored as a novel feature for REM stage detection in this work. This difference is hereon referred to as *SEFd*. For an epoch e, it is determined by first calculating the *SEFd* values of fifteen 2 s subepochs in the 30 s EEG epoch (i.e. the difference between *SEF95* and *SEF50* of the subepochs). The mean of these differences is taken to be the *SEFd* of the epoch being processed as shown below, where se is the subepoch and n is its index. A 9-point moving average filter is then applied to the final *SEFd* value.

$$SEFd(e) = \frac{1}{15} \times \sum_{n=1}^{15} (SEF95[se_n] - SEF50[se_n]) \tag{4.3}$$

In Figure 4.5 the *SEFd* values during different sleep stages are shown in both traditional and bandlimited frequency ranges. The figure shows clear peaks during REM stages when the analysis is restricted to the 8-16 Hz range. However no such characteristic pattern is observed when the entire frequency band is analysed.

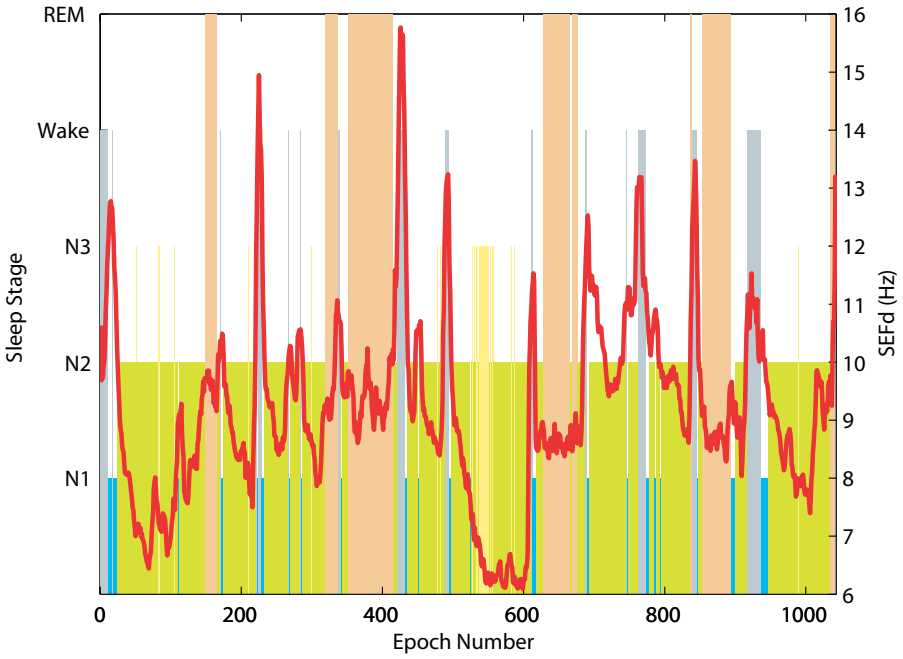


(a)

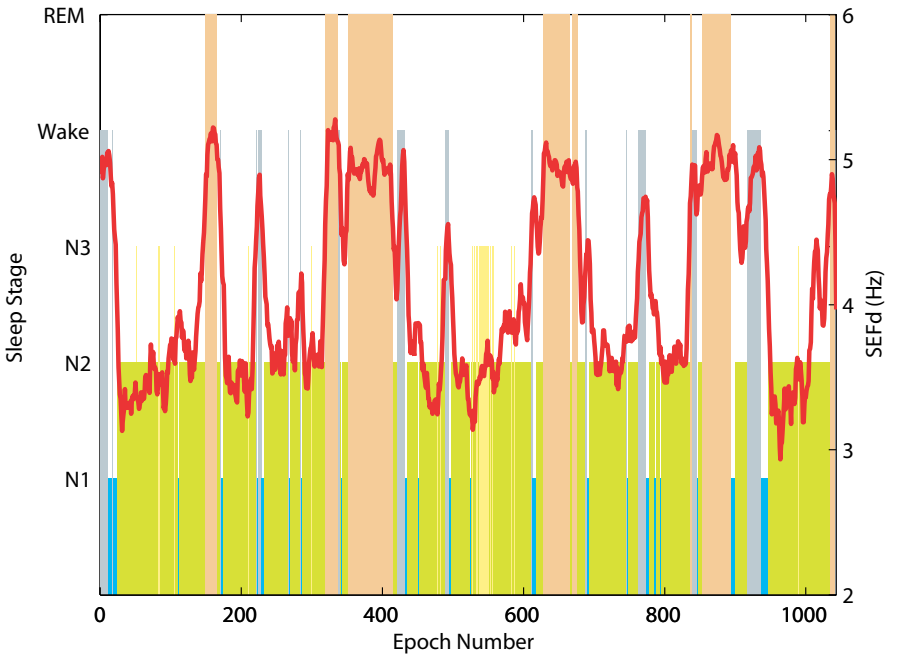


(b)

Figure 4.4: *Hypnogram and SEF95 in the (a) 0.5-50 Hz and (b) 8-16 Hz band of the EEG signal for one training subject.*



(a)



(b)

Figure 4.5: *Hypnogram and SEFd in the (a) 0.5-50 Hz and (b) 8-16 Hz band of the EEG signal for one training subject.*

Figure 4.6 shows the *SEFd* overlaid on the hypnogram of all training subjects in the 8-16 Hz range and illustrates that the values of *SEFd* are consistently high during all REM phases for the entire sleep duration of all subjects. It confirms that the results obtained earlier were not random but actually a general trend that is clearly visible from the *SEFd* and hypnogram plots of all training subjects. The plots show that, in general, all N2 and N3 phases appear to have lower *SEFd* values. N1 stages have a slightly higher value but still lower than REM stages in most cases. This pattern of high *SEFd* values during REM phase in the 8-16 Hz frequency band could be a useful feature to discriminate it from other sleep stages.

It is not a coincidence that the *SEFd* values are high during REM stages only. The reason for this is a result of lower *SEF50* and higher *SEF95* values during REM stages. The two trends in *SEF* can be explained by the observations in Figure 4.1 which shows how the power within the 8-16 Hz band changes during both REM and non-REM stages (including Wake). The signal power is similar in both REM and non-REM around 8 Hz. Following this, the power in REM is lower than non-REM from 9-15 Hz with the difference being highest around the 12 Hz mark. Uchida *et al.* [38] reported the absence of 12-16 Hz activity during REM stages which is causing the power to be lower than non-REM. Therefore the median frequency (*SEF50*) in 8-16 Hz range is expected to be lower during REM stages. The trend of higher *SEF95* values during REM suggests an increase in the higher frequency components of the 8-16 Hz band. In Figure 4.1, apart from 4.1(e), all cases demonstrate an increase in the power spectrum of REM around the 15 Hz mark. Further, the activity in the neighbouring beta frequency band is also highest during REM sleep [38]. This causes the *SEF95* values to be higher during REM within the 8-16 Hz range. *SEFd* essentially represents both these changes in *SEF50* and *SEF95*, and is observed to be greatest when the frequency band is limited between 8 Hz and 16 Hz.

4.4.3 Quantifying the discriminatory power of *SEFd*

The plots in Figure 4.6 appear to show *SEFd* having clearly discernible values that can be used to detect REM epochs. However the performance of the feature still needs to be quantified by using real data. For this, the feature will be used with a simple thresholding classifier and its classification performance examined at various thresholds. This section explains how a Receiver Operating Characteristics curve can be used to analyse the performance of a classifier. This method will then be used to quantify the discriminatory power of *SEFd* compared to *SEF50* and *SEF95* in 8-16 Hz and 0.5-50 Hz frequency bands.

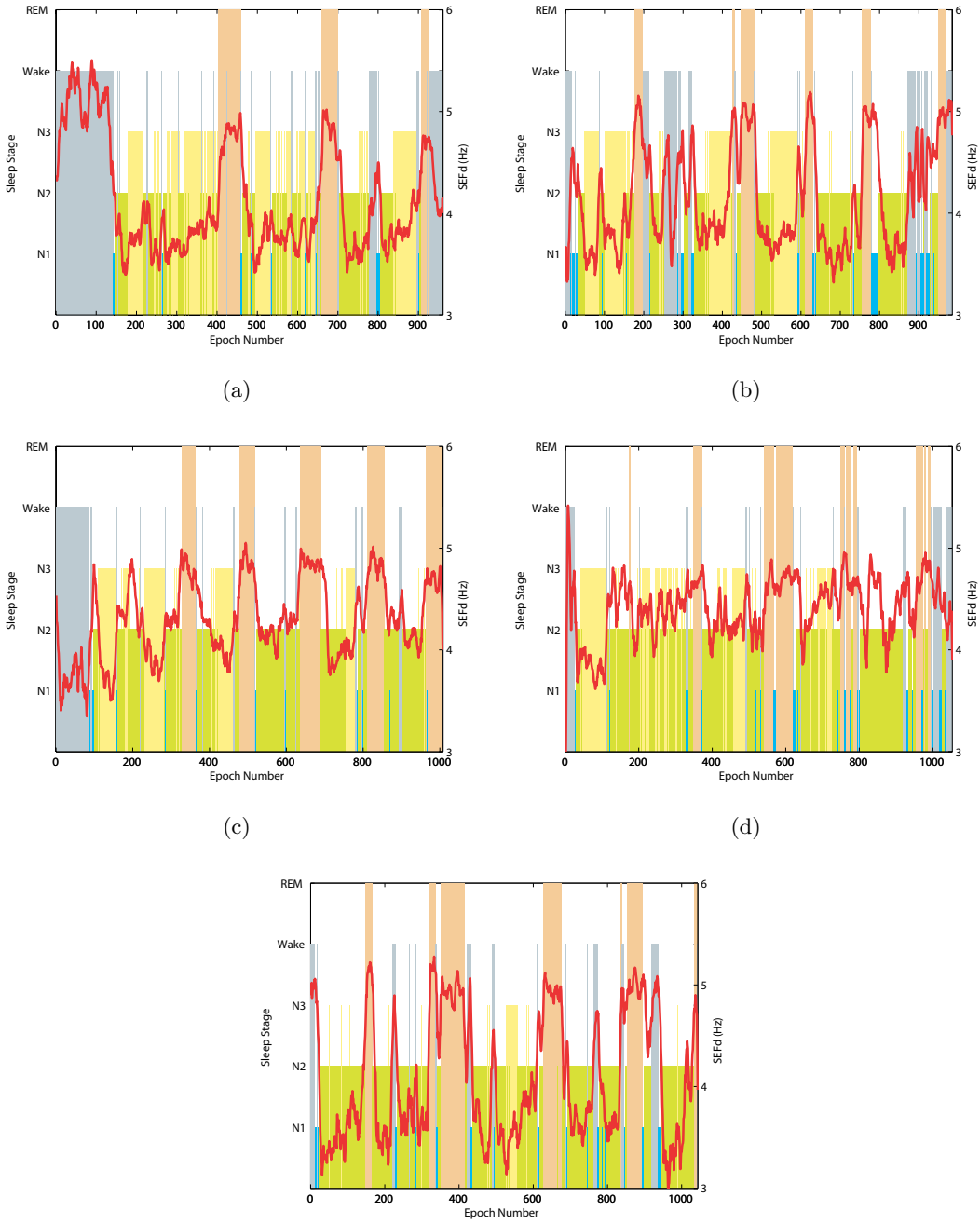


Figure 4.6: *Hypnogram and SEFd in the 8-16 Hz band of the EEG signal for training subjects 01-05 on plots (a)-(e) respectively. The plots show clear peaks during all the REM phases for every case.*

Receiver Operating Characteristics Curve

A Receiver Operating Characteristics (ROC) curve is used to show the performance of a classifier at various thresholds. It is a plot of the *sensitivity* of the algorithm against the false positive rate computed as $1 - specificity$. Each point on the curve corresponds to the performance of the classifier at a fixed threshold value.

As an example, some ROC curves are shown in Figure 4.7. The green curve shows the ideal classifier performance, such that the *sensitivity* of the classifier is always 1. The important point here is the one on top left, corresponding to a *sensitivity* of 1 and $1 - specificity$ (false positive rate) of 0. The blue curve is an example of a realistic classifier. The red line corresponds to a classifier that has an equal chance of detecting either a true positive or rejecting it [39]. Hence, any ROC curve below the red line is useless. Generally the objective is to get as close as possible to the ideal classifier and the point (0,1) on the curve [40].

A ROC curve is also used to find the optimum threshold to maximise the performance of an algorithm. This depends on what is actually considered as optimum. For example, in cases where *sensitivity* is required to be high without worrying about $specificity$, the optimum threshold is different compared to cases where $specificity$ is of prime importance. In this work, both these measures are given equal weight. The optimum threshold is then computed by finding the point on the ROC curve that is closest to $(0, 1)$ on the curve [40], [41]. The optimum point for the example classifier is also shown on Figure 4.7.

The area under the curve (AUC) gives an indication of the performance of the algorithm and is used to compare its performance with other classifiers. For the three classifiers shown in Figure 4.7, the AUC of the ideal classifier is 1 while that of the chance classifier and example classifier are 0.5 and 0.91 respectively. The closer the AUC can get to one, the better a classifier is to being ideal. Similarly, a classifier with higher AUC is better than the one with lower AUC.

SEFd vs. *SEF95* and *SEF50*

To quantify the discriminatory ability of *SEFd* as compared to both *SEF50* and *SEF95* features individually in the 8-16 Hz frequency range as well as in the 0.5-50 Hz frequency range, all the three different features were used to classify REM epochs in both frequency ranges. A simple thresholding classifier was used and the ROC curves were plotted in each case by sweeping the detection threshold. For each of the three features, assume that a given epoch $E(n)$ is classified as REM when it has an *SEFx* value greater than a certain threshold $SEFx_{th}$ (*SEFx* can be any of three features that is being analysed).

$$E(n) = \begin{cases} REM, & \text{if } SEFx(n) \geq SEFx_{th} \\ non - REM, & \text{otherwise} \end{cases} \tag{4.4}$$

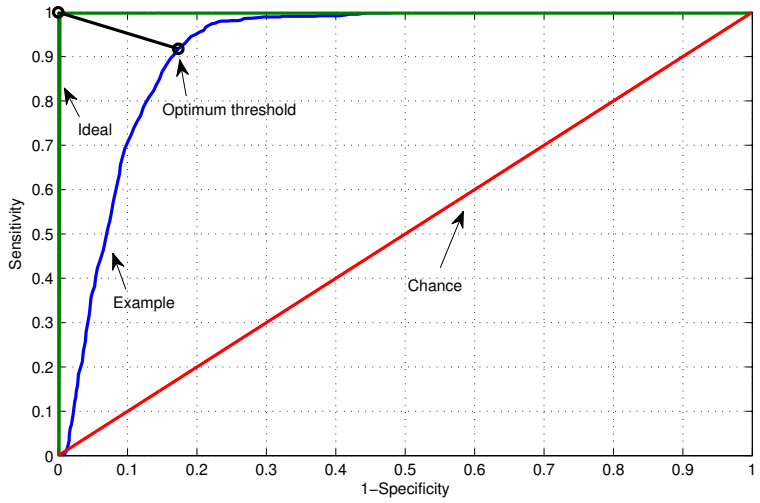


Figure 4.7: *An illustration of ideal, example and worst-case ROC curves. Also shown is the optimum threshold for the example ROC curve.*

The AUC for the three features in both frequency ranges is shown in Table 4.3. Two important points can be noted from this result. First, the AUC values confirm that all features perform better when limited to the 8-16 Hz frequency range. Second, it can be clearly seen that *SEFd*, as a feature, is far superior to both *SEF50* and *SEF95* with a much higher AUC value.

Therefore, *SEFd* in the 8-16 Hz band is used as the main feature for REM detection in this work. The step of limiting the analysis band to 8-16 Hz also has an added advantage of reducing the number of CPU cycles and resources to generate the value of the feature. When the algorithm is implemented on resource-constrained hardware, the number of cycles and registers required to compute SEF from 8-16 Hz, as compared to 0.5-50 Hz, will be at least 3 times smaller. Having said that, although the reduction in number of cycles is a desirable outcome of limiting the analysis frequency it is not the main reason of doing so. The primary reason is to obtain a more robust feature capable of discriminating REM with a simple classifier.

Table 4.3: *AUC values for the three features in different frequency ranges.*

Feature / Frequency Range	**0.5-50 Hz**	**8-16 Hz**
SEF50	0.7023	0.7530
SEF95	0.7082	0.7390
SEFd	0.6930	0.9247

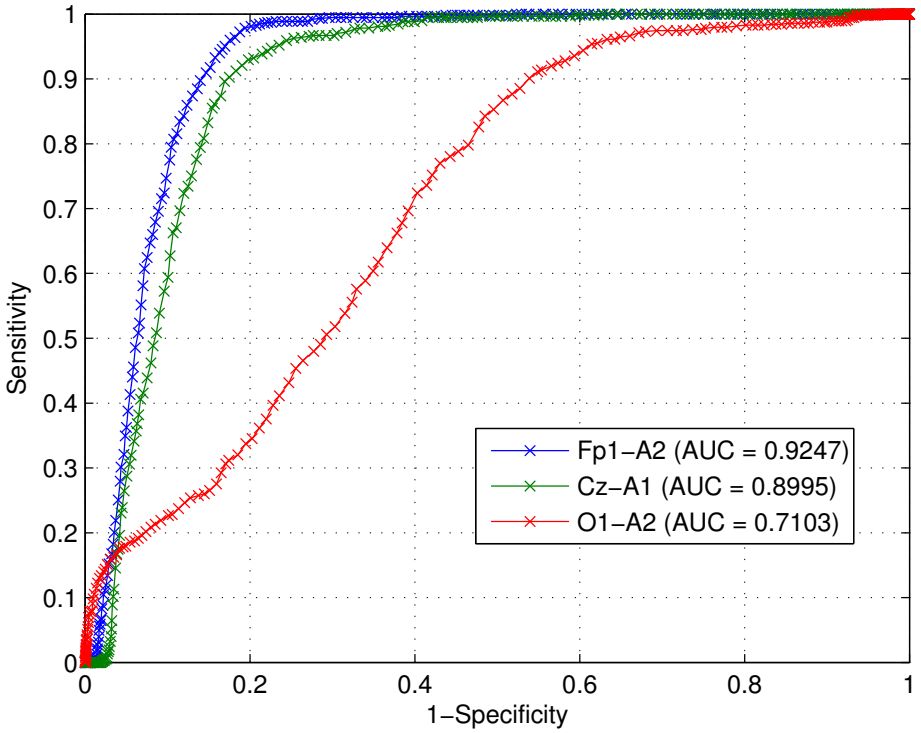


Figure 4.8: *ROC Curves and AUC for three EEG channels using SEFd feature from the training dataset.*

4.4.4 Channel selection

All the analysis so far has been performed using data from the Fp1-A2 channel. However, that selection is arbitrary and the discriminatory power of the *SEFd* feature from other channels needs to be studied. The DREAMS database used for analysis consists of data from three EEG channels. The aim of this work is to use only a single channel of EEG for sleep analysis hence it needs to be determined which channel gives the best result.

For each of the three EEG channels, a ROC curve can be plotted by sweeping the threshold *SEFd* in small steps across a wide range and then determining the sensitivity and specificity of the classifier. The AUC of each plot can then be used to compare the three channels. The ROC curves for three different EEG channels and the AUC for each are shown in Figure 4.8.

Using all five training subjects, the thresholding classifier achieved its highest detection performance using data from the frontal (Fp1-A2) channel. The performance degraded when the C3-A1 channel was used while it was worst using the O1-A2 channel. This suggests the the performance steadily reduces when moving away from the frontal region of the brain. Looking at the results from each individual subject, the AUC values are higher when using the channel Fp1-A2 for all training subjects except *Subject05* where the value is slightly lower when Fp1-A2 channel is used. Table 4.4 shows the AUC values

for all training subjects for the three different channels.

Table 4.4: *AUC values for all training subjects using the three EEG channels.*

Subject	Fp1-A2	Cz-A1	O1-A2
1	0.8491	0.8341	0.6159
2	0.9663	0.9353	0.8441
3	0.9740	0.8970	0.3025
4	0.8650	0.8413	0.6063
5	0.9574	0.9762	0.9698

There is a very clear trend in *SEFd* feature performing better when data from the frontal channel is used. This can be explained by the conclusions of Corsi-Cabrera *et al.* [42] that REM sleep exhibits uncoupled EEG activity between frontal and posterior regions of brain. Thus, features present in the frontal region during REM sleep may be completely absent in the posterior region. The close proximity of Fp1-A2 channel to the EOG could also result in some eye movement activity being picked up in the frontal EEG thus resulting in better performance.

Since the largest AUC is for channel Fp1-A2, it is selected as the one to use for further analysis.

4.4.5 Further features

It has been demonstrated so far that the frequency range of 8-16 Hz is of interest for the separation of REM sleep epochs from others. A novel feature, *SEFd*, has also been shown to have high discriminatory power for this purpose and the feature tends to perform well when frontal channels of EEG are used.

The *SEFd* shows clear peaks during REM sleep phases for all the subjects, however, occasional peaks are also observed during other phases of sleep in some cases. For example, *Subject01* in Figure 4.6(a) shows high values of *SEFd* during Wake stage (similar to those during REM) while this is not the case for *Subject03*, in Figure 4.6(c). The frequency distribution plot for the training data in Figure 4.9 also shows that while most of the REM epochs have *SEFd* values of more than 4.5 Hz, there are still some epochs from other stages overlapping in this frequency range.

As a result, two further features are investigated in the same frequency band to reduce potential false detections occurring in other sleep stages from the use of *SEFd*. These features are the absolute and relative powers in the 8-16 Hz frequency band.

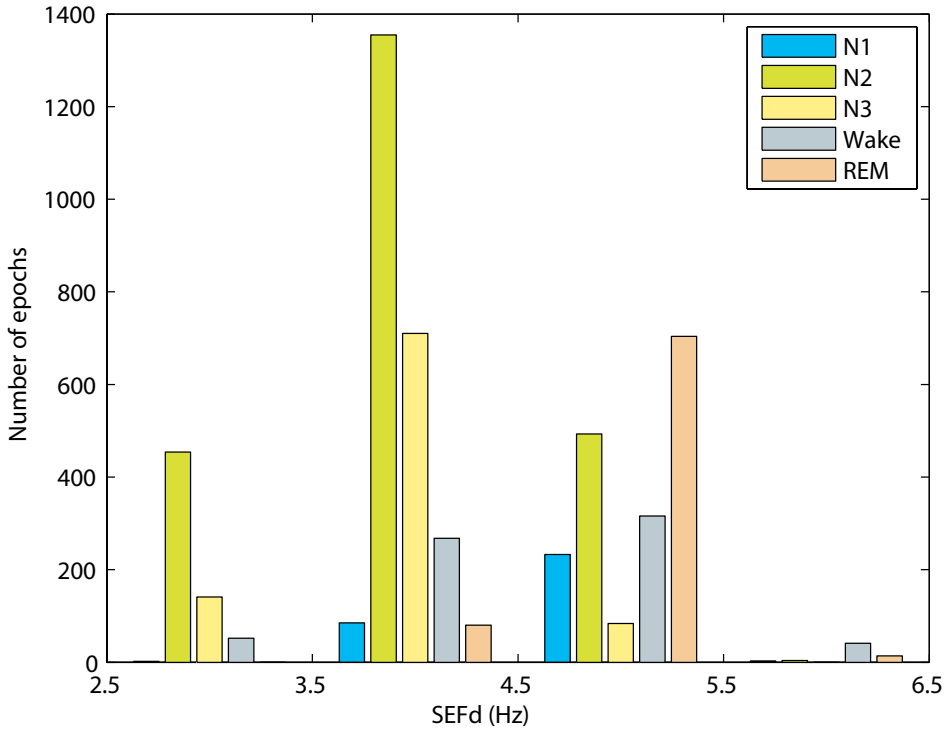


Figure 4.9: *Frequency distribution of SEFd values at different sleep stages across all training subjects.*

Absolute Power

The absolute power (AP) of a signal in a fixed frequency range, $f_1 - f_2$ Hz, is calculated by summing the magnitudes obtained from Fourier coefficients between these frequencies.

$$AP = 20 \times \log(\sum_{i=n(f_1)}^{n(f_2)} |mag_i|) \tag{4.5}$$

In the equation above, f_1 and f_2 are 8 Hz and 16 Hz respectively and $n(f_1)$ and $n(f_2)$ are the indices at these frequencies. AP is calculated for each 2 second subepoch and averaged over the standard 30 second epoch. Figure 4.10 shows the absolute power with hypnogram for *Subject01*. REM stage was observed to have the lowest AP in 8-16 Hz range. Further, AP values during Wake and N1 stages were higher than REM. These results are in line with the observations in [38] and [7]. AP hence, could be used as an extra differentiating feature for REM, Wake and N1 stages. Similar trends were also observed for the other training subjects.

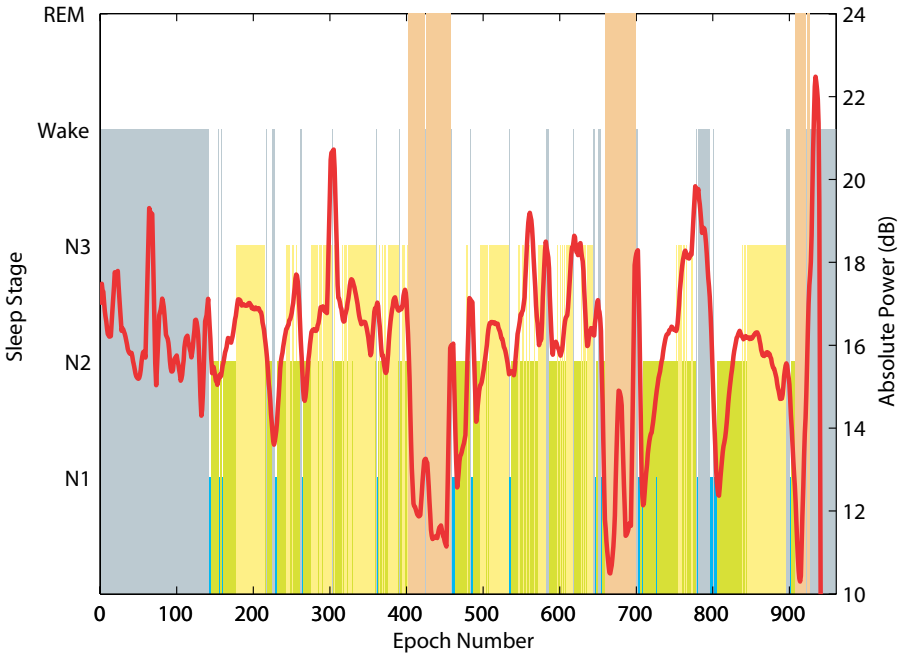


Figure 4.10: *Hypnogram and AP in the 8-16 Hz band of the EEG signal for* Subject01. *AP values can be seen to be lowest during each REM phase.*

Relative Power

The relative power (RP) of a signal in a fixed frequency range, $f_1 - f_2$ Hz (8-16 Hz) is calculated by taking the ratio of the absolute powers of the signal in the range of interest and the entire signal bandwidth.

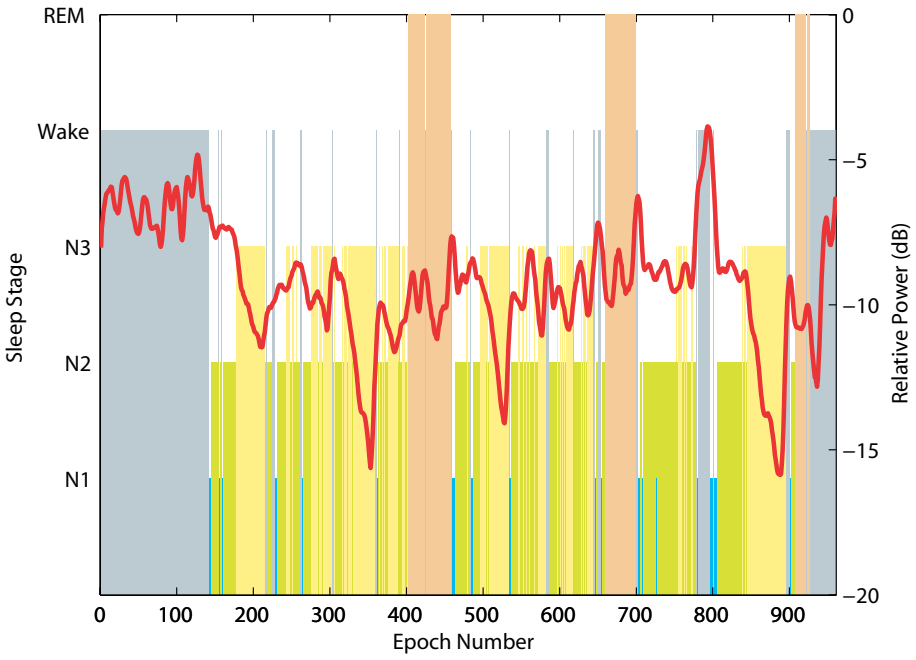


Figure 4.11: *Hypnogram and RP in the 8-16 Hz band of the EEG signal for* Subject01. *RP values can be seen to be stable around -8 dB mark.*

$$RP = 20 \times \log \left(\frac{\sum_{i=n(f_1)}^{n(f_2)} |mag_i|}{\sum_{i=1}^{n} |mag_i|} \right) \tag{4.6}$$

RP is also calculated first for 2 second subepochs and then averaged over 30 second epochs. Figure 4.11 shows the relative power in 8-16 Hz for *Subject01* together with its hypnogram. During REM stage, *RP* does not exhibit any characteristic peak or trough unlike *SEFd* or *AP* plots. However, the values stay close to -8 dB range approximately for all subjects and are also different from those during N3 and Wake stages. This makes the feature useful for reducing potential false detections.

4.5 REM detection algorithm

4.5.1 Overview

Figure 4.12 shows a complete flowchart of the proposed REM detection algorithm. A single channel EEG input is first transformed into the frequency domain using the FFT. In the first stage FFT coefficients are used to compute *SEF95* and *SEF50* within the 8-16 Hz band. The difference between these two spectral edge frequency measures, *SEFd*, is then taken for every epoch. If *SEFd* is found to be greater than a certain maximum threshold $SEFd_{th}$, the epoch under analysis is marked as a candidate REM epoch ($cREM$), and further checks are applied at the next stage. Otherwise, the epoch is rejected as non-REM and not analysed any further.

$$E(n) = \begin{cases} cREM, & \text{if } SEFd(n) \geq SEFd_{th} \\ 0, & \text{otherwise} \end{cases} \tag{4.7}$$

The second stage of the algorithm is used to reject false positives amongst the candidate REM epochs. If an epoch satisfies the condition in Equation 4.7, its AP and RP values are evaluated in the 8-16 Hz range for further analysis. Only when both AP and RP values satisfy the conditions below, a candidate REM epoch is considered a true detection. Otherwise it is rejected as non-REM.

$$AP \leq AP_{max} \tag{4.8}$$

$$RP_{min} \leq RP \leq RP_{max} \tag{4.9}$$

The algorithm works in two stages where the first stage is highly sensitive and detects candidate REM epochs. The second stage is specific and helps in reducing the number of false detections. The choice of features used at each of the two stages was determined by their discriminatory ability in detecting REM epochs. *SEFd* was found to be the most sensitive feature and was therefore used at the first stage of the algorithm (to shortlist as many REM epochs as possible) followed by AP and RP. This two-stage process also helps in keeping the computational load low since AP and RP features are calculated only when there is a candidate REM epoch identified in the first stage.

4.5.2 Establishing the threshold values

The detection thresholds (SEF_{th}, AP_{max}, RP_{max} and RP_{min}) were tuned to achieve the best average performance for REM detection in terms of both sensitivity and specificity. For this, a ROC curve was plotted of $sensitivity$ against $(1 - specificity)$ with varying thresholds.

On the ROC curve, the optimum operating point for the first stage of the algorithm

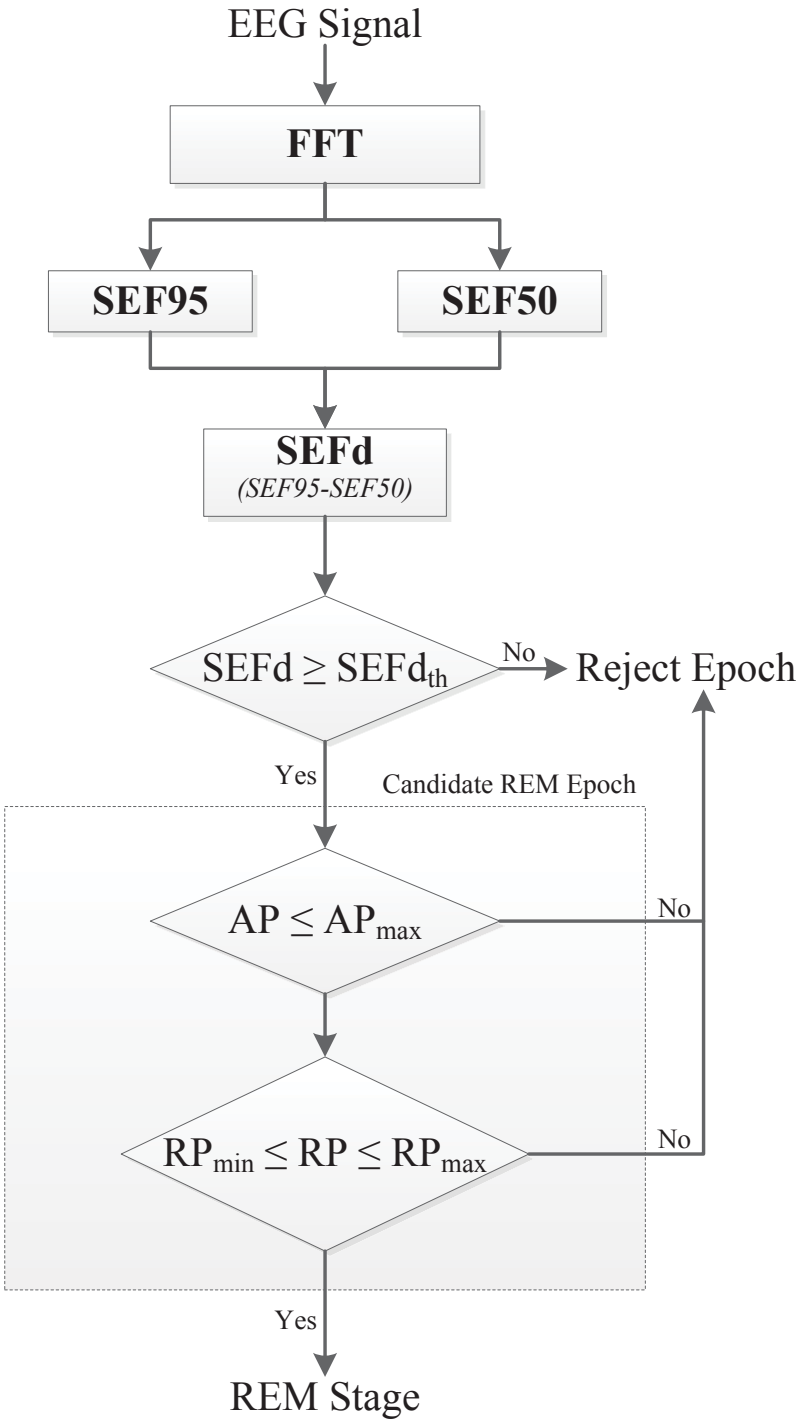


Figure 4.12: *Block diagram of the REM detection algorithm.*

(SEF_{th}) is established by giving equal weight to both sensitivity and specificity and determining the minimum distance of the curve from the $(0, 1)$ coordinate [41], [43]. This is the point on the curve closest to the $(0, 1)$ coordinate. Using this optimum threshold, the candidate REM epochs (with *SEFd* greater than this threshold) are analysed. For these epochs, a second ROC curve is plotted by sweeping the RP and AP thresholds. The optimum operating point for these features is also established by determining the shortest distance of the second curve from the $(0, 1)$ coordinate. The thresholds corresponding to the optimum points for both stages of the algorithm are shown in Table 4.5. It should be noted that a different operating point could be selected depending on whether higher sensitivity at the cost of more false positives is tolerable or if a lower false positive rate is desired at the cost of sensitivity.

Table 4.5: *Best performing thresholds for SEFd, AP and RP.*

Parameter	Value
SEF_{th}	4.54 Hz
AP_{max}	15.5 dB
RP_{max}	-6.08 dB
RP_{min}	-13.03 dB

To ascertain whether the use of fixed thresholds is a good option for this application, the standard deviation of the three features in an epoch is analysed in great detail. The standard deviation is calculated for all subjects in each sleep stage and then in all of them combined. The average values are shown in Table 4.6 and the deviations appear to be small and similar in all stages.

Table 4.6: *Standard deviation of the three features in different sleep stages.*

	Wake	**N1**	**N2**	**N3**	**REM**	**All**
SEFd	1.222	1.058	1.253	1.162	0.972	1.171
AP	3.057	2.747	2.971	2.635	1.996	2.741
RP	2.498	2.414	2.695	2.915	2.172	2.593

4.6 Results

4.6.1 Performance metrics

The performance of the algorithm is evaluated by quantifying the sensitivity and selectivity of REM detection, specificity of REM rejection and the overall accuracy of REM classification. These metrics have already been discussed in Chapter 3 of this thesis and are briefly repeated below.

1) Sensitivity, which represents the fraction of REM epochs that are correctly identified by the algorithm.

$$Sensitivity = \frac{TP}{TP + FN} \tag{4.10}$$

2) Specificity, which represents the fraction of non-REM epochs being correctly rejected.

$$Specificity = \frac{TN}{TN + FP} \tag{4.11}$$

3) Selectivity, which is the fraction of correct detections of REM with respect to the total number of automatic REM detections (also known as positive predictive value or PPV).

$$Selectivity = \frac{TP}{TP + FP} \tag{4.12}$$

4) Accuracy, which is the fraction of the total number of correct detections and rejections of REM epochs in the sleep recording.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{4.13}$$

In the equations above and the following sections, TP (True Positives) is the number of epochs correctly scored as REM, FP (False Positives) is the number of epochs incorrectly scored as REM, TN (True Negatives) is the number of epochs correctly rejected as non-REM, and FN (False Negatives) is the number of epochs incorrectly rejected as non-REM.

4.6.2 Training data results

The individual as well as average subject performance of the algorithm using the fixed optimum thresholds obtained above is shown in Table 4.7. All the five subjects showed sensitivity greater than 89% individually and around 94% on average. Only 46 out of the total 798 REM epochs are not detected by the algorithm while the number of false positives is recorded as 475 epochs from a total of 5050 epochs across all five subjects. Most of the Wake and NREM epochs are correctly rejected giving an average specificity of 89%. The overall accuracy of the system is found to be close to 90%.

Table 4.7: *Performance of algorithm on training database.*

Subject	REM_{tot}	REM_{det}	TP	Sen (%)	Spe (%)	Sel(%)	Acc(%)
1	113	158	103	91.15	93.51	65.19	93.24
2	122	242	119	97.54	85.75	49.17	87.21
3	212	224	189	89.15	95.60	84.38	94.25
4	155	324	146	94.19	80.18	45.06	82.24
5	196	279	195	99.49	90.08	69.89	91.85
Total	798	1227	752				
Average				94.31	89.03	62.74	89.76

REM_{tot} - REM epochs in the test; REM_{det} - REM epochs detected by the algorithm.

TP - true positives; Sen - sensitivity; Spe - specificity; Sel - selectivity; Acc - accuracy.

4.6.3 Test data results

The algorithm is then tested using the detection thresholds in Table 4.5 on complete night EEG recordings of the 15 test subjects. Results of the individual and average performance are shown in Table 4.8. The average sensitivity for these test subjects is reduced to 83%. Apart from *Subject12* and *Subject14*, all have a sensitivity of more than 70% and even in these cases where sensitivity is on the lower side, the accuracy is still greater than 92%. *Subject12*, with the lowest sensitivity, has a large Wake period in the middle of sleep and sporadic Wake epochs throughout the night. The exact cause of this Wake period is not known but it leads to the presence of movement artefacts, making the detection of REM difficult. The average specificity, selectivity and accuracy values of the test set are, however, similar to the training results.

The first stage of the algorithm uses *SEFd* to detect most of the REM epochs while the second stage uses *AP* and *RP* to eliminate false detections later. In order to illustrate this, the performance of the algorithm is quantified in both stages: it is first run using the *SEFd* feature only and then the *AP* and *RP* features are added to it. Results in Table 4.9 show an increase in specificity, selectivity and accuracy when the *AP* and *RP* are used together with the *SEFd*. Furthermore, the number of false positives is reduced from 2534 to 1395 when these features are added. However this performance boost comes at the cost of reduction in average sensitivity from 88.7% to 83% when the *AP* and *RP* features are added. Depending on the application, a suitable trade-off must be achieved to reduce the number of false positives up to a point where reduction in the number of true positives is acceptable. Conversely, both specificity and selectivity can be traded off to achieve higher sensitivity if a higher number of false positives can be tolerated.

Table 4.8: *Performance of algorithm on test database.*

Subject	REM_{tot}	REM_{det}	TP	Sen (%)	Spe (%)	Sel(%)	Acc(%)
6	187	297	186	99.47	86.28	62.63	88.76
7	131	285	107	81.68	79.8	37.54	80.04
8	162	284	120	74.07	79.68	42.25	78.74
9	131	184	124	94.66	93.91	67.39	94
10	146	153	113	77.4	95.48	73.86	92.92
11	212	268	203	95.75	91.83	75.75	92.66
12	87	64	52	59.77	98.63	81.25	95.11
13	89	267	88	98.88	82.42	32.96	83.74
14	163	118	105	64.42	98.45	88.98	92.93
15	123	113	92	74.8	97.06	81.42	93.79
16	147	164	105	71.43	92.8	64.02	89.56
17	162	259	137	84.57	85.16	52.9	85.06
18	166	274	166	100	87.37	60.58	89.42
19	162	303	150	92.59	82.43	49.51	84.03
20	153	225	115	75.16	88.91	51.11	87.07
Total	2221	3258	1863				
Average				82.98	89.35	61.48	88.52

REM_{tot} - REM epochs in the test; REM_{det} - REM epochs detected by the algorithm.

TP - true positives; Sen - sensitivity; Spe - specificity; Sel - selectivity; Acc - accuracy.

Table 4.9: *Algorithm performance analysis at output of first and second stages.*

Features	TP	FP	TN	FN	Sen (%)	Spe (%)	Sel(%)	Acc(%)
SEFd only	1996	2534	10437	225	88.67	80.52	48.40	81.91
SEFd, *AP* and *RP*	1863	1395	11576	358	82.98	89.35	61.48	88.52

TP - true positives; FN - false negatives; TN - true negatives; FP - false positives; (numbers are total for 15 test subjects).

Sen - sensitivity; Spe - specificity; Sel - selectivity; Acc - accuracy; (numbers are average for 15 test subjects).

A breakdown of the false detections in Table 4.10 shows the sleep stages in which these false positives occur, as well as the fraction of each stage falsely scored as REM. Across the 15 test subjects, only 18.5% of the total Wake epochs are misclassified as REM. Amongst these, almost a quarter of false positives in Wake stage come from *Subject08* alone. Similarly, about a third of total N1 epochs are misclassified as REM (424 out of a total of 1157 N1 epochs). Since N1 and REM have similarities in EEG, this is to be expected. It is however still a positive result since it does show a discriminatory ability that can be used to distinguish between REM and N1 stages using EEG. Only 431 out of the total 5936 N2 epochs are misclassified as REM (about 7%) where *Subject07* contributes almost a fifth to the false positives in N2. Finally, only 7 N3 epochs across all 15 test subjects are misclassified by the algorithm as REM and 5 of those come from *Subject07*.

Kappa agreement

The agreement rate between the algorithm and the visual scorer is also evaluated using Cohen's kappa (κ) values. These are shown in Table 4.11 for different combinations of sleep stages. For the test sleep data including all sleep stages, κ was found to be 0.61. When Wake stage was removed κ increased to 0.69. When only REM, Wake and N1 stages of data were considered κ was 0.57. According to Landis and Koch's classification [44], these κ values represent moderate to substantial agreement in all cases.

Patient-specific thresholds

The REM detection algorithm uses fixed thresholds to classify REM epochs for all test subjects. This simplifies the classification stage thus reducing the algorithm's complexity. However, the use of patient-specific thresholds is also investigated by plotting a ROC curve for one subject at a time and then finding the best thresholds for it from the curve. The individual results for each subject using patient-specific thresholds are shown in Table 4.12. The use of patient-specific thresholds results in sensitivity of over 80% in all the cases with the average sensitivity increasing to 90%, specificity 94%, selectivity 73% and accuracy of about 94%. The average results using fixed thresholds and patient-specific thresholds are compared in Table 4.13. All the performance measures using patient-specific thresholds are higher compared to the use of fixed thresholds with the greatest improvement seen in sensitivity and selectivity.

The detailed breakdown of the misclassified epochs is shown in Table 4.14. The increase in sensitivity is a consequence of using patient-specific *SEFd* threshold that resulted in 132 more REM epochs being correctly identified. Adjusting the *AP* threshold reduced the number of false detections by almost 50% (down from 1395 to 752 epochs) thereby improving the overall selectivity. The most notable reduction is in the number of misclassified epochs in Wake stage followed by N1 and N2 stages.

Table 4.10: *Breakdown of all false detections in test database.*

Subject	TP	FN	TN	FP	FP_W(W)	FP_{N1}(N1)	FP_{N2}(N2)	FP_{N3}(N3)
6	186	1	698	111	48(179)	25(51)	38(355)	0(224)
7	107	24	703	178	50(394)	31(49)	92(283)	5(155)
8	120	42	643	164	119(181)	41(95)	4(324)	0(207)
9	124	7	926	60	5(216)	33(71)	22(515)	0(184)
10	113	33	845	40	23(71)	8(66)	8(411)	1(337)
11	203	9	731	65	1(122)	6(67)	57(401)	1(206)
12	52	35	862	12	6(393)	3(90)	3(234)	0(157)
13	88	1	839	179	69(181)	73(112)	37(432)	0(293)
14	105	58	828	13	0(208)	5(46)	8(417)	0(170)
15	92	31	693	21	1(114)	20(98)	0(294)	0(208)
16	105	42	761	59	4(258)	23(88)	32(370)	0(104)
17	137	25	700	122	60(67)	20(45)	42(564)	0(146)
18	166	0	747	108	54(169)	40(87)	14(420)	0(179)
19	150	12	718	153	17(129)	70(131)	66(460)	0(151)
20	115	38	882	110	76(198)	26(61)	8(456)	0(277)
Total	1863	358	11576	1395	533(2880)	424(1157)	431(5936)	7(2998)

TP - true positives; FN - false negatives; TN - true negatives; FP - false positives.

FP_X(X) shows false positives in stage X and the total number of epochs from stage X in parentheses.

Table 4.11: *Level of agreement (Cohen's kappa values) when using different sleep stages*

Sleep stages	Cohen's kappa (κ)
REM, Wake and NREM	0.61
REM and NREM	0.69
REM, Wake and N1	0.57
REM and Wake	0.65
REM and N1	0.48

Table 4.12: *Performance of algorithm on test database using patient-specific thresholds.*

Subject	REM_{tot}	REM_{det}	TP	Sen (%)	Spe (%)	Sel(%)	Acc(%)
6	187	223	179	95.72	94.56	80.27	94.78
7	131	159	117	89.31	95.23	73.58	94.47
8	162	207	120	74.07	89.22	57.97	86.69
9	131	198	129	98.47	93	65.15	93.64
10	146	162	116	79.45	94.8	71.6	92.63
11	212	260	200	94.34	92.46	76.92	92.86
12	87	101	81	93.1	97.71	80.2	97.29
13	89	169	83	93.26	91.55	49.11	91.69
14	163	176	152	93.25	97.15	86.36	96.51
15	123	154	119	96.75	95.1	77.27	95.34
16	147	207	136	92.52	91.34	65.7	91.52
17	162	192	133	82.1	92.82	69.27	91.06
18	166	182	158	95.18	97.19	86.81	96.87
19	162	195	146	90.12	94.37	74.87	93.71
20	153	162	126	82.35	96.37	77.78	94.5
Total	2221	2747	1995				
Average				90	94.19	72.86	93.57

REM_{tot} - number of REM epochs in the test; REM_{det} - number of REM epochs detected by the algorithm.

TP - true positives; Sen - sensitivity; Spe - specificity; Sel - selectivity; Acc - accuracy.

Table 4.13: *Comparison of results when using fixed versus patient-specific thresholds.*

	Fixed Thresholds	**Patient-specific Thresholds**
Sensitivity (%)	82.98	90.00
Specificity (%)	89.35	94.19
Selectivity (%)	61.48	72.86
Accuracy (%)	88.52	93.57

Table 4.14: *Breakdown of all false detections in test database using patient-specific thresholds.*

Subject	TP	FN	TN	FP	FP_W(W)	FP_{N1}(N1)	FP_{N2}(N2)	FP_{N3}(N3)
6	179	8	765	44	18(179)	12(51)	14(355)	0(224)
7	117	14	839	42	2(394)	10(49)	30(283)	0(155)
8	120	42	720	87	49(181)	34(95)	4(324)	0(207)
9	129	2	917	69	6(216)	39(71)	24(515)	0(184)
10	116	30	839	46	28(71)	9(66)	8(411)	1(337)
11	200	12	736	60	1(122)	4(67)	54(401)	1(206)
12	81	6	854	20	13(393)	4(90)	3(234)	0(157)
13	83	6	932	86	16(181)	50(112)	20(432)	0(293)
14	152	11	817	24	2(208)	7(46)	15(417)	0(170)
15	119	4	679	35	2(114)	27(98)	6(294)	0(208)
16	136	11	749	71	4(258)	28(88)	39(370)	0(104)
17	133	29	763	59	14(67)	11(45)	34(564)	0(146)
18	158	8	831	24	8(169)	9(87)	7(420)	0(179)
19	146	16	822	49	1(129)	23(131)	25(460)	0(151)
20	126	27	956	36	9(198)	17(61)	10(456)	0(277)
Total	1995	226	12219	752	173(2880)	284(1157)	293(5936)	2(2998)

TP - true positives; FN - false negatives; TN - true negatives; FP - false positives.

FP_X(X) shows false positives in stage X and the total number of epochs from stage X in parentheses.

The mean and median averages and the standard deviation of all patient-specific thresholds are shown in Table 4.15. The mean and median values of the *SEFd* threshold are 4.54 Hz and 4.5 Hz respectively which are close to the fixed threshold being used. For AP_{max} and RP_{max}, both mean and median values are close to each other, but slightly less than the fixed threshold value used. For RP_{min} the difference between mean and median averages is the largest and both these values are lower than their fixed-threshold counterpart. The relative standard deviation is lowest for SEF_{th} at 4% while for the other three thresholds it is between 12-17%.

Table 4.15: *Mean, median and standard deviation of the patient-specific thresholds.*

Threshold	Mean	Median	Std Dev
SEF_{th} (Hz)	4.54	4.50	0.18
AP_{max} (dB)	15.07	15.30	2.12
RP_{max} (dB)	-6.19	-6.60	0.75
RP_{min} (dB)	-13.41	-14.30	2.31

A clear example of the improvement by using patient specific thresholds can be observed in *Subject14*. In this case, the best performance is achieved when *SEFd* threshold is lowered to 4.38Hz. The detection results of *Subject14* using the two *SEFd* thresholds, is contrasted in Table 4.16 and shows a drastic improvement in detection sensitivity when the threshold is lowered.

Table 4.16: *Comparison of fixed versus patient-specific SEFd thresholds in Subject14*

	$SEFd_{th}$ **= 4.54 Hz**	$SEFd_{th}$ **= 4.38 Hz**
Sensitivity (%)	64.42	93.25
Specificity (%)	98.45	97.15
Selectivity (%)	88.98	86.36
Accuracy (%)	92.93	96.51

Similarly, the use of patient-specific thresholds reduces the number of misclassified N2 epochs in *Subject07* and *Subject19* (Table 4.17). Further, patient-specific thresholds also reduce the number of misclassified Wake epochs in *Subject06*, *Subject08*, *Subject13*, *Subject07*, *Subject18* and *Subject20* as summarised in Table 4.18. In all test subjects combined, the proportion of misclassifications in Wake stage reduced from 18.5% down to 6%.

Table 4.17: *Number of misclassified N2 epochs using fixed and patient-specific thresholds.*

Subject	Number of N2 epochs	Misclassified N2 epochs with fixed threshold	Misclassified N2 epochs with patient-specific threshold
07	283	92	30
19	460	66	25

Table 4.18: *Number of misclassified Wake epochs using fixed and patient-specific thresholds.*

Subject	Number of Wake epochs	Misclassified Wake epochs with fixed threshold	Misclassified Wake epochs with patient-specific threshold
06	179	48	18
08	181	119	49
13	181	69	16
17	67	60	14
18	54	54	8
20	76	76	9

These results show that apart from movement artefacts (which is an obvious cause of lower performance) some subjects may exhibit a slightly lower/higher overall spectral edge frequency. This suggests that the use of adaptive thresholds that can adjust to individual subjects can further improve the results and should be explored in future work. However, this improvement in performance will come at the cost of additional algorithm complexity. Nevertheless, the use of fixed thresholds still achieves a performance comparable to other algorithms thus highlighting the strength of the fixed-threshold approach.

4.6.4 Performance comparison

There are very few single-channel EEG-based sleep scoring methods in literature. The review in Section 4.2 revealed only eight such methods giving their REM detection performance. It is difficult to compare the results of different algorithms due to the varying databases used to test each of them. However, since some of these methods also report their performance on the publicly available PhysioNet Sleep-EDF database [45], the algorithm in this chapter is also evaluated using the same database for a fair comparison with them.

This PhysioNet database consists of PSG recordings from 8 healthy subjects with two channels of EEG recorded for each (see Appendix A for details about the database). A single frontal EEG channel (Fpz-Cz) is used to evaluate the algorithm using leave-one-out cross validation (LOOCV) on over 8800 scored epochs across all 8 subjects consisting of about 9 hours overnight recording for each subject with the exception of movement (MT) and unscored epochs.

The performance of the algorithm in this work and those of other one-channel EEG-based methods on the same database for REM detection is shown in Table 4.19. The algorithm in this work achieved similar sensitivity and selectivity, compared to others, while using only three features. Not only are the number of features used in other algorithms much higher (at least 7 times more), the classifiers used in them are also much more complex.

If there are processing and power constraints attached with the system being designed then the algorithm presented in this chapter could prove to be useful for achieving REM detection performance that is similar to other methods using a much smaller number of features and a simple classifier. However, if there are no such limitations either of the methods listed in Table 4.19 would achieve similar results. Further, the Fpz-Cz channel was used to evaluate the algorithm's performance since the main feature used in this work ($SEFd$) exhibits strongest discriminatory ability in the frontal channels. The other algorithms listed in Table 4.19 used the channel Pz-Oz because this was closest to their algorithm requirements and gave the best results.

Table 4.19: *Performance comparison with other single-channel EEG methods that have been evaluated using PhysioNet Sleep-EDF Database.*

	Method	Features	Classifier	Sen (%)	Sel(%)
This work	Spectral power	3	Thresholding	80.6	74.8
Ref. [16]	MSE, AR model	21	LDA and contextual smoothing	85.4	78.8
Ref. [26]	Spectral power	30	Neural network	82.3	-
Ref. [36]	Spectral and temporal features	Multiple	Fuzzy classifier and contextual smoothing	63.0	91.7

4.7 Discussion

Automatic detection of REM stages in sleep is desirable to aid in the development of a fully automated sleep staging system. The bulk of sleep staging is performed using EEG signals while EOG and EMG signals are generally required to mark REM stages. During the REM phases there are characteristic bursts of eye movements observed on EOG traces that are used to score them. However these eye movements are present for only up to 27% of the total REM sleep time [46]. This suggests that EOG signals, albeit helpful, may not be able to detect all REM stage epochs. For a wearable sleep staging system, size and power are the main constraints. A reduction in the number of channels directly helps in power saving by reducing the amount of signals to process thereby minimising processor load and size and consequently improving battery life. It also leads to a physical system that is lighter in weight and easier to use. Identification of REM stage from one channel of EEG with reliable performance, therefore, could go a long way in system processing, power and size reduction.

In this chapter, the difference between spectral edge frequencies (*SEF95* and *SEF50*) in the 8-16 Hz frequency band is introduced as a novel feature that exhibits clear discriminatory abilities for scoring REM epochs. On a test database of 15 subjects, this feature alone was able to detect 88.7% of the total REM epochs. The database was used as is, without removing any movement artefacts or stages, to reflect real world recording conditions. Absolute and relative powers in the same spectral band were used as additional features to further analyse the candidate REM epochs at the first stage. This helped in reducing the number of false detections by more than 40%. The final two-stage algorithm resulted in sensitivity of 83% within a 95% confidence interval range of 81.4% to 84.5%

for a total of 2221 test REM epochs while the Cohen's kappa value showed substantial agreement between visual and automatic detection of REM. The performances were even higher when patient-specific thresholds were used resulting in a sensitivity of 90%. The algorithm also resulted in similar performance compared to other single-channel EEG-based methods when evaluated on the same database.

The REM detection algorithm presented here has several advantages. First, its performance is comparable to most of the methods in literature including those that use multiple EEG, EOG and EMG channels. Second, it uses a simple thresholding method with fixed thresholds to mark REM epochs in contrast to some other systems that use complex neural networks with a large input feature set. This low-complexity classifier is advantageous for portable and wearable systems with limited processing cycles and power budget. Third, results from automatic sleep staging systems of other research groups [47]–[49] suggest overlap of REM stage with N1 in various feature spaces. These two stages have similar EEG and are difficult to differentiate. The feature used here also successfully distinguishes between the majority of N1 and REM epochs. About 63% of the total N1 epochs were correctly distinguished from REM despite their strong EEG similarities. The misclassification proportion in Wake stage was much smaller, at 18.5%. This is, even with the inclusion of the movement epochs (which are marked as Wake according to AASM rules). This number could go down further with the use of an artefact rejection method at the front end of the algorithm as well as using adaptive thresholding at the classification stage. About 7% of N2 epochs were wrongly detected as REM while only 7 out of 2998 N3 epochs were misclassified. The total number of false positive epochs was 1395 which may seem like a large number. However, the total epochs under test were 15192 from all stages of sleep. Considering this, the fraction of false positives is actually less than 10%. Ideally, the number of false positives should be even smaller. The use of patient-specific thresholds reduces it to 752 epochs (less than 5% false positives). Finally, the REM detection algorithm in this chapter uses data from only one EEG channel and therefore keeps the data rate and processing load small.

Overall the investigations in this study illustrate that spectral edge frequency in the 8-16 Hz band of EEG can be a useful feature for the detection REM sleep phase as demonstrated with a simple algorithm. Although this algorithm showed a good performance, the main objective of this work was not to present the best performing REM detection algorithm but to introduce and evaluate a novel feature that could be used with a simple algorithm or as an added feature in a different algorithm. The heuristic classifier used in this work is very simple and may not represent the most optimal approach. Other classifiers such as decision trees, SVMs, etc., may result in an improved detection performance. Nevertheless, the results presented in this chapter will be useful for sleep EEG system designers by helping to reduce the number of channels, computational cost, device size and power consumption for future truly wearable and automated sleep staging systems.

References

[1] J. Horne, "Why REM sleep? clues beyond the laboratory in a more challenging world," *Biol. Psychol.*, vol. 92, no. 2, pp. 152–68, 2013.

[2] C. Iber, S. Ancoli-Israel, A. Chesson, and S. Quan, *The AASM manual for the scoring of sleep and associated events: rules, terminology and technical specifications.* Westchester, IL: American Academy of Sleep Medicine, 2007.

[3] A. Iranzo, J. L. Molinuevo, J. Santamaría, M. Serradell, M. J. Martí, F. Valldeoriola, and E. Tolosa, "Rapid-eye-movement sleep behaviour disorder as an early marker for a neurodegenerative disorder: a descriptive study," *Lancet Neurol.*, vol. 5, no. 7, pp. 572–577, 2006.

[4] P. Indursky and V. Rotenberg, "Change of mood during sleep and REM sleep variables," *Int. J. Psychiatry Clin. Pract.*, vol. 2, no. 1, pp. 47–51, 1998.

[5] T. A. Mellman, V. Bustamante, W. R. Pigeon, and B. Nolan, "REM sleep and the early development of posttraumatic stress disorder," *Am. J. Pschiatry*, vol. 159, no. 10, pp. 1696–1701, 2002.

[6] G. W. Vogel, F. Vogel, R. S. McAbee, and A. J. Thurmond, "Improvement of depression by REM sleep deprivation," *Arch. Gen. Pschiatry*, vol. 37, no. 3, pp. 247–253, 1980.

[7] M. Corsi-Cabrera, Z. Munoz-Torres, Y. del Rio-Portilla, and M. A. Guevara, "Power and coherent oscillations distinguish REM sleep, stage 1 and wakefulness," *Int. J. Psychophysiol.*, vol. 60, no. 1, pp. 59–66, 2006.

[8] R. Bódizs, M. Sverteczki, and E. Mészáros, "Wakefulness-sleep transition: Emerging electroencephalographic similarities with the rapid eye movement phase," *Brain Res. Bull.*, vol. 76, no. 1, pp. 85–89, 2008.

[9] A. Rechtschaffen and A. Kales, *A manual of standardized terminology, techniques and scoring system for sleep stages of human subjects.* Washington D.C.: Public Health Service, U.S. Government Printing Office, 1968.

[10] E. Estrada, H. Nazeran, J. Barragan, J. R. Burk, E. A. Lucas, and K. Behbehani, "EOG and EMG: two important switches in automatic sleep stage classification," in *IEEE EMBC*, New York, September 2006.

[11] S. Charbonnier, L. Zoubek, S. Lesecq, and F. Chapotot, "Self-evaluated automatic classifier as a decision-support tool for sleep/wake staging," *Comput. Biol. Med.*, vol. 41, no. 6, pp. 380–9, 2011.

[12] M. Hanaoka, M. Kobayashi, and H. Yamazaki, "Automated sleep stage scoring by decision tree learning," in *IEEE EMBC*, Chicago, July 2000.

[13] J. Virkkala, R. Velin, S. Himanen, A. Varri, K. Muller, and J. Hasan, "Automatic sleep stage classification using two facial electrodes," in *IEEE EMBC*, Vancouver, August 2008.

[14] V. Swarnkar, U. R. Abeyratne, and C. Hukins, "Automatic estimation of macro-sleep-architecture using a single channel of EEG," in *ICIIS*, Sri Lanka, December 2009.

[15] S.-F. Liang, C.-E. Kuo, Y.-H. Hu, and Y.-S. Cheng, "A rule-based automatic sleep staging method," *J. Neurosci. Methods*, vol. 205, no. 1, pp. 169–76, 2012.

[16] S.-F. Liang, C.-E. Kuo, Y.-H. Hu, Y.-H. Pan, and Y.-H. Wang, "Automatic stage scoring of single-channel sleep EEG by using multiscale entropy and autoregressive models," *IEEE Trans. Instrum. Meas.*, vol. 61, no. 6, pp. 1649–1657, 2012.

[17] S. Khalighi, T. Sousa, D. Oliveira, G. Pires, and U. Nunes, "Efficient feature selection for sleep staging based on maximal overlap discrete wavelet transform and SVM," in *IEEE EMBC*, Boston, September 2011.

[18] T. Sousa, A. Cruz, S. Khalighi, G. Pires, and U. Nunes, "A two-step automatic sleep stage classification method with dubious range detection," *Comput. Biol. Med.*, vol. 59, no. 1, pp. 42–53, 2015.

[19] T. Lajnef, S. Chaibi, P. Ruby, P.-E. Aguera, J.-B. Eichenlaub, M. Samet, A. Kachouri, and K. Jerbi, "Learning machines and sleeping brains: Automatic sleep stage classification using decision-tree multi-class support vector machines," *J. Neurosci. Methods*, no. 0, 2015.

[20] V. Bajaj and R. B. Pachori, "Automatic classification of sleep stages based on the time–frequency image of EEG signals," *Comput. Methods Programs Biomed.*, vol. 112, no. 3, pp. 320–328, 2013.

[21] H. Park, K. Pa, and D.-u. Jmn, "Hybrid neural-network and rule-based expert system for automatic sleep stage scoring," in *IEEE EMBC*, Chicago, July 2000.

[22] M. Schwaibold, T. Penzel, J. Schochlin, and A. Bolz, "Combination of AI components for biosignal processing application to sleep stage recognition," in *IEEE EMBC*, Istanbul, October 2001.

[23] J. Y. Tian and J. Q. Liu, "Automated sleep staging by a hybrid system comprising neural network and fuzzy rule-based reasoning," in *IEEE EMBC*, Shanghai, September 2005.

[24] C. M. Held, J. E. Heiss, P. A. Estévez, C. A. Perez, M. Garrido, C. Algarín, and P. Peirano, "Extracting fuzzy rules from polysomnographic recordings for infant sleep classification," *IEEE Trans. Biomed. Eng.*, vol. 53, no. 10, pp. 1954–62, 2006.

[25] E. Oropesa, H. L. Cycon, and M. Jobert, "Sleep stage classification using wavelet transform and neural network," *International computer science institute*, 1999.

[26] M. Ronzhina, O. Janousek, J. Kolarova, M. Novakova, P. Honzik, and I. Provaznik, "Sleep scoring using artificial neural networks," *Sleep Med. Rev.*, vol. 16, no. 3, pp. 251–63, 2012.

[27] R. Agarwal and J. Gotman, "Computer-assisted sleep staging," *IEEE Trans. Biomed. Eng.*, vol. 48, no. 12, pp. 1412–1423, 2001.

[28] A. Flexer, G. Gruber, and G. Dorffner, "Improvements on continuous unsupervised sleep staging," in *IEEE NNSP*, Martigny, September 2002.

[29] L. G. Doroshenkov, V. A. Konyshev, and S. V. Selishchev, "Classification of human sleep stages based on EEG processing using hidden markov models," *Biomed. Eng.*, vol. 41, no. 1, pp. 24–28, 2007.

[30] S.-T. Pan, C.-E. Kuo, J.-H. Zeng, and S.-F. Liang, "A transition-constrained discrete hidden markov model for automatic sleep staging," *Biomed. Eng. Online*, vol. 11, no. 1, pp. 1–19, 2012.

[31] F. Yaghouby, P. Modur, and S. Sunderam, "Naive scoring of human sleep based on a hidden markov model of the electroencephalogram," in *IEEE EMBC*, Chicago, August 2014.

[32] U. Malinowska, H. Klekowicz, A. Wakarow, S. Niemcewicz, and P. J. Durka, "Fully parametric sleep staging compatible with the classical criteria," *Neuroinformatics*, vol. 7, no. 4, pp. 245–253, 2009.

[33] S. Günes, K. Polat, and S. Yosunkaya, "Efficient sleep stage recognition system based on EEG signal using k-means clustering based feature weighting," *Expert Sys. Appl.*, vol. 37, no. 12, pp. 7922–7928, 2010.

[34] M. Längkvist, L. Karlsson, and A. Loutfi, "Sleep stage classification using unsupervised feature learning," *Adv. Artif. Neyral Sys.*, pp. 1–9, 2012.

[35] J. L. Rodriguez-Sotelo, A. Osorio-Forero, A. Jimenez-Rodriguez, D. Cuesta-Frau, E. Cirugeda-Roldan, and D. Peluffo, "Automatic sleep stages classification using EEG entropy features and unsupervised pattern analysis techniques," *Entropy*, vol. 16, no. 12, pp. 6573–6589, 2014.

[36] C. Berthomier, X. Drouot, M. Herman-Stoïca, P. Berthomier, J. Prado, D. Bokar-Thire, O. Benoit, J. Mattout, and M.-P. D'Ortho, "Automatic analysis of single-channel sleep EEG: validation in healthy individuals," *Sleep*, vol. 30, no. 11, pp. 1587–95, 2007.

[37] University of MONS - TCTS Laboratory. (2015) The DREAMS Subjects Database. [Online]. Available: http://www.tcts.fpms.ac.be/~devuyst/Databases/DatabaseSubjects/.

[38] S. Uchida, T. Maloney, and I. Feinberg, "Sigma (12-16 Hz) and beta (20–28 Hz) EEG discriminate NREM and REM sleep," *Brain Res.*, vol. 659, no. 1, pp. 243–248, 1994.

[39] R. M. McFall and T. A. Treat, "Qunatifying the information value of clinical assessments with signal detection theory," *Annu. Rev. Psychol.*, vol. 50, no. 1, pp. 215–241, 1999.

[40] A. K. Akobeng, "Understanding diagnostic tests 3: receiver operating characteristic curves," *Acta Paediatr.*, vol. 96, no. 5, pp. 644–647, 2007.

[41] N. J. Perkins and E. F. Schisterman, "The inconsistency of optimal cut-points using two ROC based criteria," *Am. J. Epidemiol*, vol. 163, no. 7, pp. 670–675, 2006.

[42] M. Corsi-Cabrera, E. Miro, Y. del Rio-Portilla, E. Perez-Garcia, Y. Villanueva, and M. A. Guevara, "Rapid eye movement sleep dreaming is characterized by uncoupled EEG activity between frontal and perceptual cortical regions," *Brain Cognition*, vol. 51, no. 3, pp. 337–345, 2003.

[43] M. Coffin and S. Sukhatme, "Receiver operating characteristic studies and measurement errors," *Biometrics*, vol. 53, no. 3, pp. 823–37, 1997.

[44] J. Landis and G. Koch, "The measurement of observer agreement for categorical data," *Biometrics*, vol. 33, no. 1, pp. 159–174, 1977.

[45] PhysioNet. (2013) Sleep-EDF Database. [Online]. Available: http://www.physionet.org/physiobank/database/sleep-edf/.

[46] L. Leclair-Visonneau, D. Oudiette, B. Gaymard, S. Leu-Semenescu, and I. Arnulf, "Do the eyes scan dream images during rapid eye movement sleep? evidence from the rapid eye movement sleep behaviour disorder model," *Brain*, vol. 133, no. 6, pp. 1737–1746, 2010.

[47] Q. Ma, X. Ning, J. Wang, and J. Li, "Sleep-stage characterization by nonlinear EEG analysis using wavelet-based multifractal formalism," in *IEEE EMBC*, Shanghai, September 2005.

[48] F. Ebrahimi, M. Mikaili, E. Estrada, and H. Nazeran, "Assessment of Itakura distance as a valuable feature for computer-aided classification of sleep stages," in *IEEE EMBC*, Lyon, August 2007.

[49] E. Estrada, H. Nazeran, F. Ebrahimi, and M. Mikaeili, "EEG signal features for computer-aided sleep stage detection," in *IEEE EMBS NER*, Antalya, April 2009.

5 Automatic sleep staging using state machine-controlled decision trees

The research presented within this chapter is an edited version of research previously published in:

S. A. Imtiaz and E. Rodriguez-Villegas, "Automatic sleep staging using state machine–controlled decision trees," in proceedings of the 37th international conference of the IEEE Engineering in Medicine and Biology Society, Milan, August 2015,

5.1 Introduction

A typical automatic sleep staging algorithm involves using some kind of signal processing technique to extract representative features followed by a classifier to assign one of the sleep stages based on the extracted features. Researchers have used support vector machines, hidden Markov models, linear discriminant analysis, artificial neural networks, decision trees and others for classification with a variety of time, frequency, entropy and wavelet based features.

A comprehensive review of various sleep staging algorithms published in literature was presented in Chapter 2. It can be concluded from this review that spectral features are by far the most popular choice when it comes to selecting the best discriminating features for sleep classification. This is not unexpected since both R&K and AASM rules of sleep analysis describe most stages of sleep in terms of the frequency content in different spectral bands. For classifiers, however, there is no *clear winner* although artificial neural networks tend to provide the highest accuracy in most cases.

In a resource-constrained wearable system, spectral features can be extracted using different filters or with Fourier transforms. This does not leave much room to implement a complex classifier such as an SVM or ANN. As a result, simpler classifiers such as decision trees are preferable to reduce design complexity.

Decision trees are commonly used for sleep stage classification. In such trees, at each node of the tree, one of the features is tested against a threshold. The result of this test determines the next node, and hence, the feature to be tested. This process continues until the end of the tree is reached and an epoch is classified. In a balanced tree with about 20-30 features, the classification of each epoch may require more than 15 comparisons.

Decision trees, in comparison to more complex classifiers, are only beneficial when the number of nodes in the tree is relatively small. This also means using smaller number of features. It is therefore important to design a decision tree that satisfies all the requirements in order for it to be useful. However, limiting the number of nodes and features directly impacts the performance that can be achieved by the classifier. An alternative approach is therefore needed that can potentially reduce the number of nodes and features of a decision tree while keeping its performance level intact.

This chapter presents a novel sleep staging classifier that uses a combination of small decision trees contextually driven by a state machine. The aim of this design is to develop a classifier that can work within the constraints discussed above. The approach used in this classifier is inspired by the combination of state machine and decision trees used in artificial intelligence for game development [1]. The idea here is to change the decision nodes that are to be executed based on the current sleep stage.

For sleep staging, when a certain stage of sleep is prevalent, it normally goes on to stay for a few epochs before transitioning to the next stage. This means that an epoch only needs to be tested to check whether it is of the same sleep stage of not. For example, if the current sleep stage is N3, the new epoch will be checked to see it is N3 or not. This can be achieved by a *one-versus-all* decision tree, which in this case is N3-versus-others. If the result indicates that the epoch is indeed N3 no further tests are needed. However, the opposite result indicates that the sleep stage has changed. Hence, four *one-versus-one* decision trees are needed to determine the new sleep stage. The order of these trees must be based on the likelihood of the next sleep stage.

This approach allows designing smaller decision trees and results in an overall shorter worst case path for individual trees in comparison to the traditional decision trees. The rest of this chapter explains the design of this novel sleep stage classifier in detail. Section 5.2 describes the database used as a source of sleep data and discusses the features extracted. Section 5.3 explains the complete algorithm showing how the trees are linked up together while Section 5.4 covers the design steps for each individual tree. The performance of this algorithm is evaluated in Section 5.5 followed by a discussion on future development and improvements in Section 5.6.

5.2 Material and methods

5.2.1 Database

Recordings from the DREAMS Subjects database [2] are used for the training and testing of the proposed sleep staging algorithm. This database consists of overnight sleep recordings of 20 subjects and has been classified according to the AASM rules. The details of this database can be found in Appendix A. The training dataset consists of recordings from the first ten subjects while the latter ten subjects will be used to test

the algorithm. Three EEG channels are available in all recordings, however, only Fp1-A2 (frontal channel) will be used in this work.

5.2.2 Features

At the initial stages of development, spectral features in several frequency bands will be extracted to study their effectiveness. The frequency bands that will be used and their corresponding ranges are shown in Table 5.1.

Table 5.1: *EEG frequency bands and their range.*

Name	Frequency Range (Hz)
alpha	8-13
alpha1	8-10
alpha2	10-13
beta	16-30
delta	0.5-4
delta1	0.5-2
delta2	2-4
gamma	30-40
sigma	11-16
theta	4-8
total	0.5-50

The ratio between the powers of all frequency bands listed in Table 5.1 are computed and added to the initial feature set. This results in having the relative powers in each frequency band (when the ratio is between a certain frequency band with respect to the power in the *total* band) as well as the power ratio between other frequency bands, for example, *sigma/beta*, *alpha/delta*, etc. Further, the three quantifications of spectral edge frequencies in Chapter 4 (*SEF50*, *SEF95* and *SEFd*) are calculated in 0.5-8 Hz, 0.5-30 Hz, 4-12 Hz and 8-16 Hz bands.

The initial set consists of a large number of features, some of which may not have any discriminatory ability while some may be duplicating the effects of other features. Because of this, the features with redundancies and low discriminatory power must be removed. This is done using sequential feature selection. As a result, 28 features remain that are shown in Table 5.2.

The reduced feature set will be used to design the decision trees in the next section as part of the sleep staging algorithm. However, it is possible that not all features from this set are required, further reducing the number of features that are needed for classification. The final list of features will be shown in Section 5.4.3 once all the decision trees have been designed.

The spectral powers for a 30-second epoch in each frequency band are calculated by taking the sum of these powers within its fifteen 2-second sub-epochs. The ratio between powers is taken at the end of this computation i.e. at the epoch level but not at the sub-epoch level. This has been done to make the feature calculation suitable for hardware implementation. As an example, the ratio between spectral powers in A and B in an epoch e is calculated as follows:

$$RelativePower(e) = \frac{\sum_{n=1}^{15}(AbsolutePower\ A)[se_n]}{\sum_{n=1}^{15}(AbsolutePower\ B)[se_n]} \tag{5.1}$$

SEF50 and *SEF95* for an epoch e are calculated by taking the mean of its fifteen sub-epoch values.

$$SEFxx(e) = \frac{1}{15} \times \sum_{n=1}^{15}(SEFxx[se_n]) \tag{5.2}$$

Finally, *SEFd* is calculated by taking the difference between the averages of *SEF50* and *SEF95*.

$$SEFd(e) = \frac{1}{15} \times \sum_{n=1}^{15}(SEF95[se_n] - SEF50[se_n]) \tag{5.3}$$

Table 5.2: *Initial list of the most relevant features for use in the sleep staging algorithm.*

Features			
rel. delta	rel. alpha1	beta/alpha	SEF50(0.5-30)
rel. delta1	rel. alpha2	SEF50(8-16)	SEF95(0.5-30)
rel. delta2	rel. theta	SEF95(8-16)	SEFd(8-16)
rel. beta	sigma/beta	SEF50(4-12)	SEFd(4-12)
rel. sigma	beta/delta	SEF95(4-12)	SEFd(0.5-8)
rel. gamma	theta/alpha	SEF50(0.5-8)	SEFd(0.5-30)
rel. alpha	delta/alpha	SEF95(0.5-8)	

rel. - relative power in a frequency band with respect to the total bandwidth (0.5-50 Hz)

5.3 Sleep staging algorithm

The sleep staging algorithm proposed in this chapter consists of several decision trees and their order of execution is controlled by a state machine. It is designed in such a way that the state machine starts with a pre-defined initial state and must satisfy two levels of checks in order to transition into another state. The first level is the *core test* which is a *one-versus-all* decision tree. It checks to determine whether the epoch being analysed is of the same sleep stage as the previous epoch or not. In other words it checks whether the current state of the machine needs to change. If the core test determines that the current epoch may potentially be of a different sleep stage then a series of *peripheral tests* are applied, otherwise the state machine remains unchanged. These peripheral tests are very small *one-versus-one* decision trees. Since there are only five possible sleep states including the current state, there can always be a maximum of four peripheral tests required. The order of these peripheral tests is important and determined during the training stage based on the likelihood of the next sleep stage. If one of these tests is passed, the sleep stage corresponding to that test is assigned to the current epoch, no further peripheral tests are executed and the state machine transitions to the new state. If, however, the peripheral tests also fail to assign a different sleep stage to the current epoch, the state of the machine remains unchanged and the last known sleep stage is assigned to the current epoch under analysis. The pseudocode of the complete algorithm is shown in Listing 5.1.

Listing 5.1: *Pseudocode of the proposed sleep staging algorithm*

Initial Condition: $current\_state$ is $Wake$

if $current\_state = Wake$ **then**
  if $CoreTest(Wake, Others) = Wake$ **then**
    $current\_state = Wake$
  else
    if $PeriTest(Wake, N2) = N2$ **then**
      $current\_state = N2$
    else if $PeriTest(Wake, N1) = N1$ **then**
      $current\_state = N1$
    else if $PeriTest(Wake, REM) = REM$ **then**
      $current\_state = REM$
    else
      $current\_state = Wake$
    end if
  end if

else if $current\_state = N1$ **then**
  if $CoreTest(N1, Others) = N1$ **then**
    $current\_state = N1$
  else
    if $PeriTest(Wake, N1) = N1$ **then**
      $current\_state = Wake$
    else if $PeriTest(N1, REM) = REM$ **then**
      if $CoreTest(REM, Others) = REM$ **then**
        $current\_state = REM$
      else
        $current\_state = N1$
      end if
    else if $PeriTest(N1, N2) = N2$ **then**
      $current\_state = N2$
    else if $PeriTest(N1, N3) = N3$ **then**
      $current\_state = N3$
    else
      $current\_state = N1$
    end if
  end if
else if $current\_state = N2$ **then**
  if $CoreTest(N2, Others) = N2$ **then**
    $current\_state = N2$
  else
    if $PeriTest(N2, REM) = REM$ *and* $CoreTest(REM, Others) = REM$ **then**
      $current\_state = REM$
    else if $PeriTest(Wake, N2) = Wake$ **then**
      if $CoreTest(Wake, Others) = Wake$ **then**
        $current\_state = Wake$
      else
        $current\_state = N2$
      end if
    else if $PeriTest(N1, N2) = N1$ **then**
      $current\_state = N1$
    else if $PeriTest(N2, N3) = N3$ **then**
      $current\_state = N3$
    else
      $current\_state = N2$
    end if

```
    end if
else if current_state = N3 then
    if CoreTest(N3, Others) = N3 then
        current_state = N3
    else
        if PeriTest(N2, N3) = N2 then
            current_state = N2
        else if PeriTest(Wake, N3) = Wake then
            current_state = Wake
        else if PeriTest(N1, N3) = N1 then
            current_state = N1
        else if PeriTest(N3, REM) = REM then
            current_state = REM
        else
            current_state = N3
        end if
    end if
else if current_state = REM then
    if CoreTest(REM, Others) = REM then
        current_state = REM
    else
        if PeriTest(N2, REM) = N2 and CoreTest(N2, Others) = N2 then
            current_state = N2
        else if PeriTest(N1, REM) = N1 then
            if PeriTest(Wake, N1) = Wake then
                current_state = Wake
            else
                current_state = N1
            end if
        else if PeriTest(Wake, REM) = Wake then
            current_state = Wake
        else if PeriTest(N3, REM) = N3 then
            current_state = N3
        else
            current_state = REM
        end if
    end if
end if
```

The algorithm starts initially with the state machine in the Wake state. For an incoming new epoch, the *CoreTest(Wake,Others)* determines whether a state change is required. If it is required, then a series of four peripheral tests are used to determine the new state of the machine. For the Wake state, the peripheral tests are: *PeriTest(Wake,N1)*, *PeriTest(Wake,N2)*, *PeriTest(Wake,N3)* and *PeriTest(Wake,REM)*. If one of these determine the epoch to be other than Wake then the epoch is assigned that sleep stage and the machine transitions to that state, otherwise the state remains unchanged. If the state of the machine is changed, then based on the newly assigned state, the next epoch will be classified by starting at a different core test following a similar pattern.

There are a few exceptions made to an otherwise symmetrical structure of the algorithm. If the *PeriTest(N1,REM)* determines an epoch to be REM (in case of current state being N1) then the *CoreTest(REM,Others)* is used additionally to confirm the epoch as REM. If the latter test returns false, the state of the machine remains N1. This is used because of the EEG similarities in N1 and REM stages. This strategy of using core tests to confirm a stage is also used in *PeriTest(N2,REM)* and *PeriTest(Wake,N2)*. In these cases, *CoreTest(REM,Others)* and *CoreTest(Wake,Others)* are used to check whether the epochs are indeed of REM and Wake stages respectively. Further when the current state is REM and the *PeriTest(N1,REM)* is executed resulting in N1, there is a likelihood of the output being Wake instead. Therefore *PeriTest(Wake,N1)* is used to determine whether the epoch should be classified as Wake or N1.

5.4 Design of decision trees

The core and peripheral tests were designed as decision trees using the `ClassificationTree` class in MATLAB. This class provides a `fitctree` function that constructs a binary classification tree based on a set of input features and their corresponding labels. At the very least, the function requires input features and their labels and to return a decision tree, however several options such as misclassification cost, pruning criterion and split criterion can be specified. MATLAB does not provide details of how this function is implemented internally only stating that it is based on the CART algorithm described in the Classification and Regression Trees book by Breiman *et al.* [3].

All the core and peripheral trees in this section are binary trees designed with equal misclassification cost for both classes of the tree. The split criterion used is Gini's diversity index (which is a measure of impurity, like entropy) and the pruning criterion is set to error as the option. Once a decision tree is returned by the function, it can be pruned down to contain a smaller number of nodes merging the leaf nodes at deeper levels. This is because the classification approach proposed here is only beneficial if the core and peripheral decision trees are small in comparison to the traditional decision trees used for sleep classification. For this reason, the core and peripheral trees are designed with a restricted number of nodes in them. Pruning can reduce the overall accuracy of a tree

since some of the nodes get merged. Therefore, it is performed only up to a level where the performance of each tree is still acceptable. The number of nodes, features and all the considerations to design these trees and their resulting performance are explained below.

5.4.1 Core tests

The core tests are the first line of tests since they are responsible to determine whether a state transition is required or not. Every new epoch will first be tested with one of these core tests hence they have to be designed with a high specificity. There will be a total of five core tests (one for each sleep stage). Each decision tree will be designed to use the minimum number of nodes possible to achieve the highest specificity and making this approach beneficial.

Wake vs. Others

This tree, shown in Figure 5.1, is designed to check whether the epoch under test is still of Wake stage or belongs to one of the four other sleep stages i.e. N1, N2, N3 and R. It has a total of 6 comparison nodes and five features: *SEF50(4-12)*, *SEFd(0.5-30)*, *beta/alpha*, *sigma/beta* and relative *theta* power. Its longest and shortest paths to decision require four and two nodes respectively. Using the training set, this tree achieves a sensitivity of 89% for detecting Wake and specificity of 94.8% for rejecting other stages.

N1 vs. Others

This decision tree is designed to discriminate N1 from N2, N3, W and R stages and is shown in Figure 5.2. It uses relative powers in *gamma*, *sigma* and *delta2* bands together with *SEF50(4-12)* and *SEFd(8-16)* as the features for classification. The shortest path in this tree requires only one comparison while the longest path has six nodes of comparisons. The results on training set shows this tree having a sensitivity of only 64.9% in detecting N1 stages and a specificity of 82% for rejecting others. The low sensitivity is expected since N1 is generally difficult to detect.

N2 vs. Others

This decision tree is used to classify an epoch as either N2 or one of the other stages (N1, N3, R and W). The shortest and longest paths in this tree require two and five comparisons respectively. The tree uses relative powers in *alpha*, *beta* and *alpha1* bands, *SEF50* and *SEFd* in 0.5-30 Hz band and *sigma/beta* power ratio at different nodes. It is shown in Figure 5.3 and results in sensitivity of 88.6% and specificity of 75.7%.

N3 vs. Others

This decision tree discriminates N3 stage from others and uses two and five comparisons in its shortest and longest paths respectively. The features needed are relative powers in *beta*, *gamma* and *delta2* bands, *sigma/beta*, *SEF50(0.5-30)* and *SEF95(4-12)*. Shown in Figure 5.4, it has an N3 detection sensitivity of 85.1% while its specificity is 91.3%.

REM vs. Others

This tree detects whether the epoch being analysed is of REM stage or of others stages. Shown in Figure 5.5, it uses *SEFd(8-16)* as the most discriminatory feature amongst others. This coincides with the results in Chapter 4 where this feature was shown to have high REM discriminatory ability. Other features in this tree include *sigma/beta*, *beta/delta* and relative powers in *delta2* and *alpha1* bands. The shortest path in this tree requires only two comparisons while the longest path requires four comparisons. The sensitivity and specificity for this tree are 86.3% and 87.9% respectively.

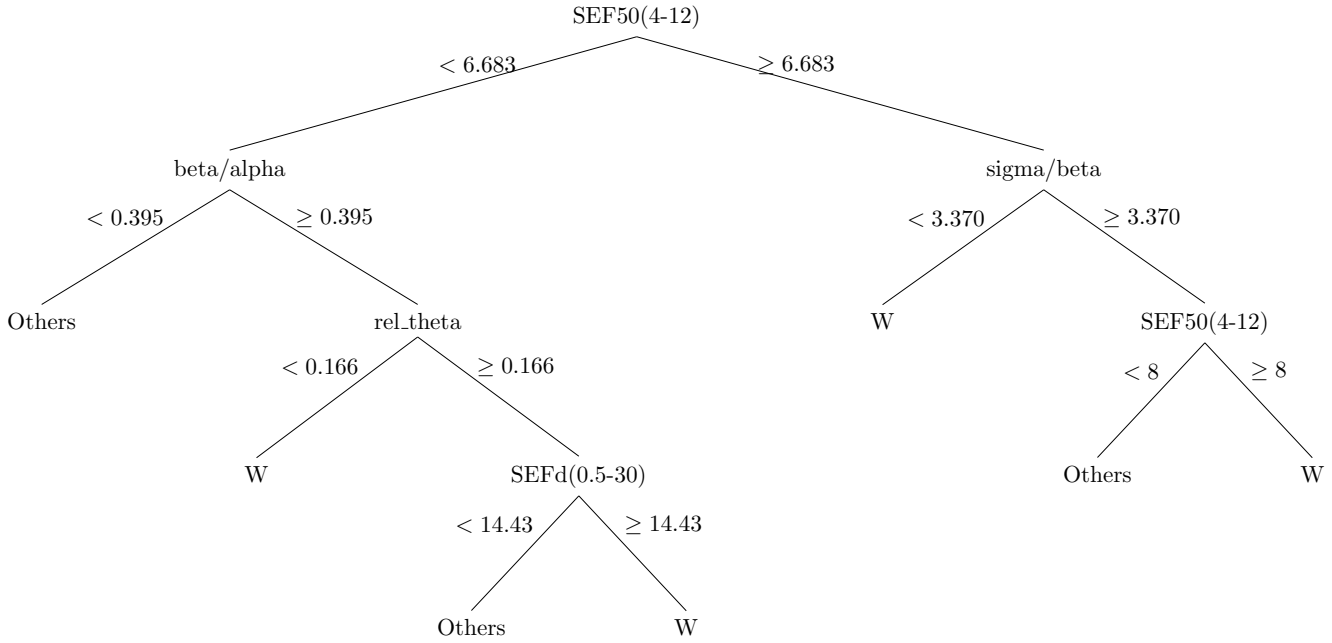


Figure 5.1: *Core decision tree to discriminate between Wake and other sleep stages.*

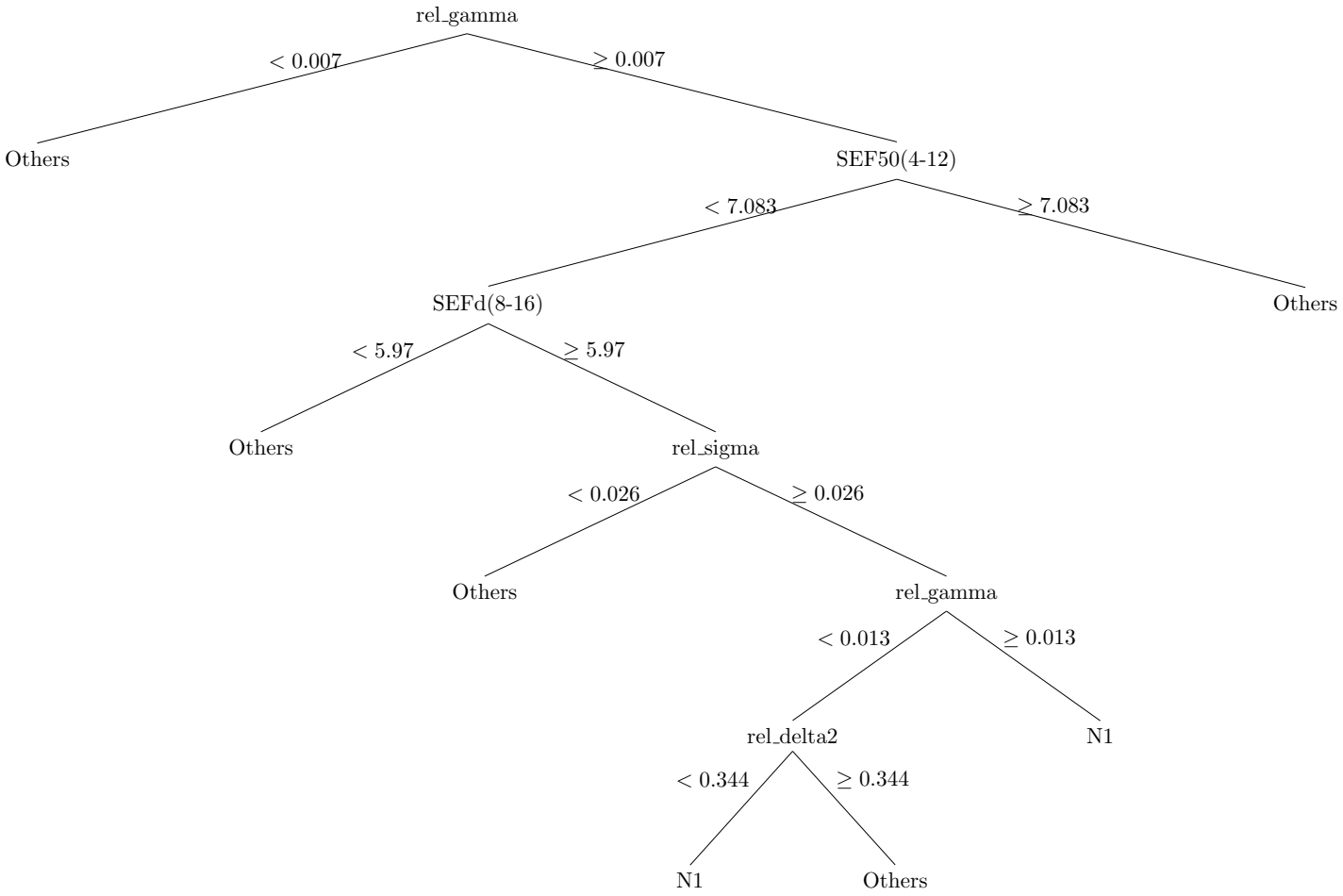


Figure 5.2: *Core decision tree to discriminate between N1 and other sleep stages.*

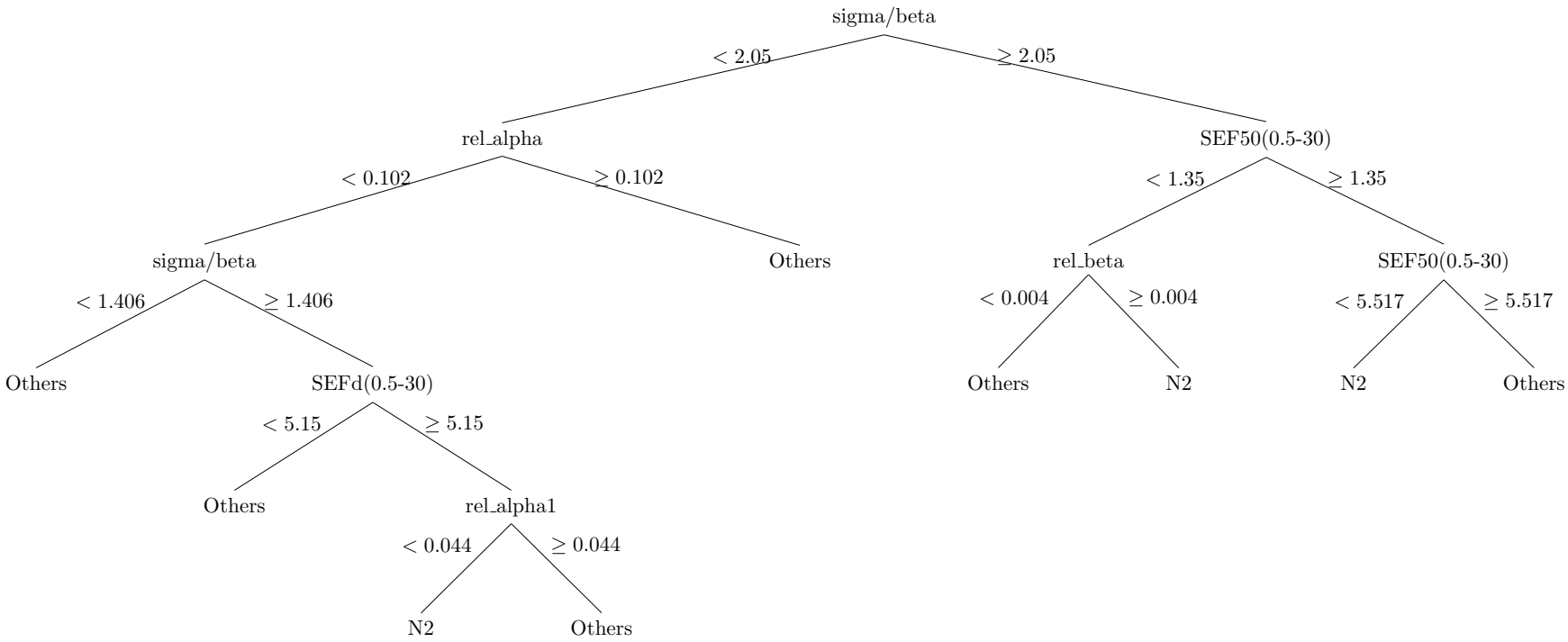


Figure 5.3: *Core decision tree to discriminate between N2 and other sleep stages.*

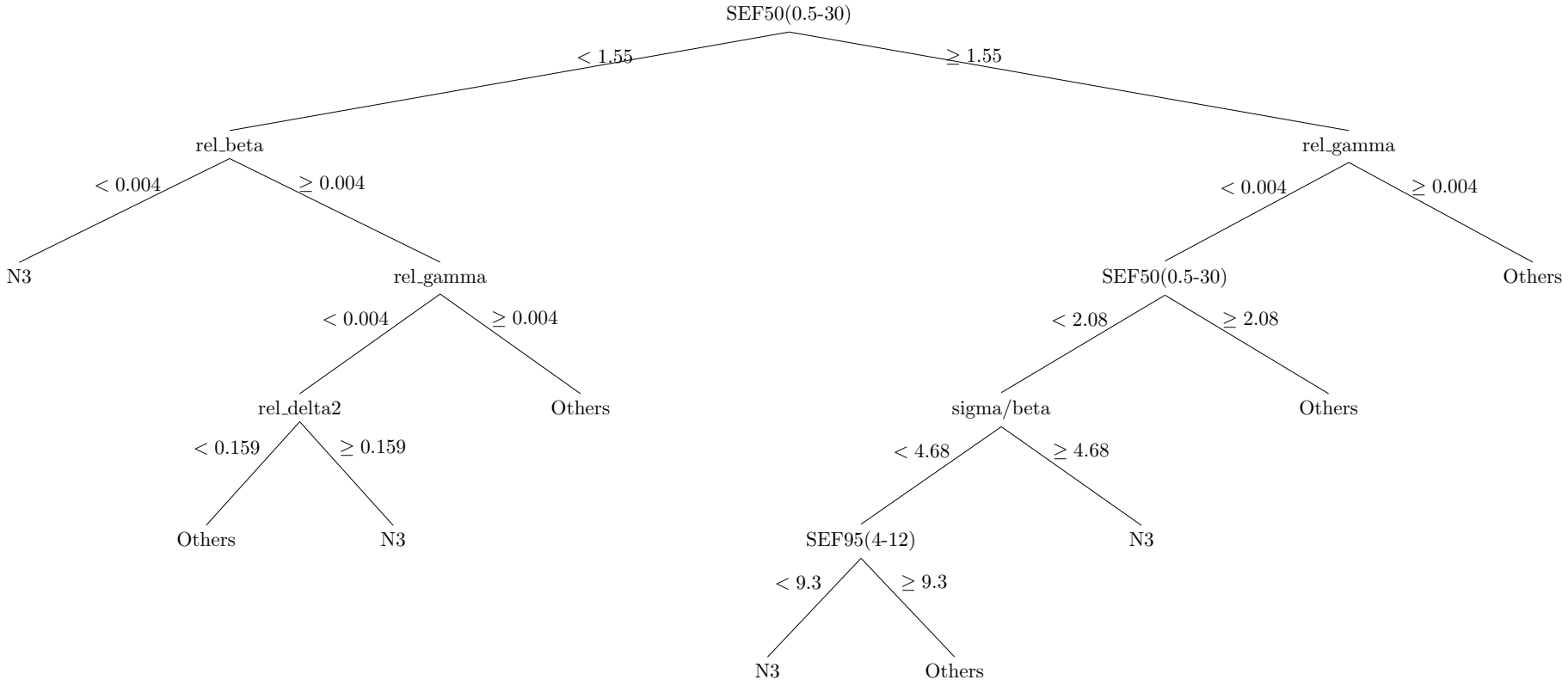


Figure 5.4: *Core decision tree to discriminate between N3 and other sleep stages.*

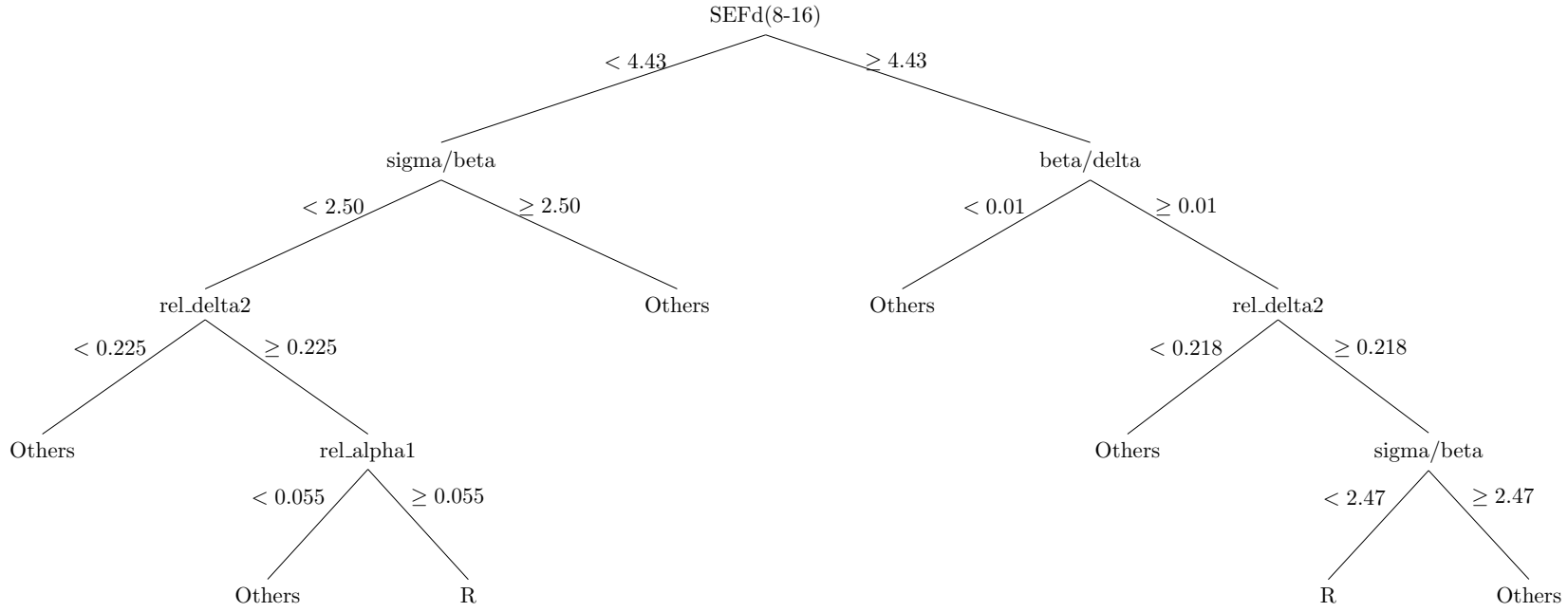


Figure 5.5: *Core decision tree to discriminate between REM and other sleep stages.*

5.4.2 Peripheral tests

The peripheral tests are *one-versus-one* decision trees that are designed to detect sleep stage transitions. They are evaluated only when a core test determines that an epoch under analysis is not of the same sleep stage as the previous one. The order of these tests is determined based on the likelihood of the occurrence of next sleep stage. Not all peripheral tests are always needed. If the first peripheral test successfully determines the next sleep stage then the others are not executed. However, if all tests fail to find the transition, then the sleep stage remains unchanged.

The peripheral decision trees are designed to be very small. They are constrained to have a maximum of two levels and three decision nodes (two nodes in the longest path). At the time of training these trees, equal weight is assigned to the two sleep stages that are being distinguished with any of the trees. The features used and training accuracy of each of these trees is explained below.

Wake vs. N1

The Wake vs. N1 peripheral test is used to determine if the sleep state is to be transitioned to N1 when current state is W or when the current state is N1 and it needs to be transitioned to W. It uses *SEF50(4-12)* and *SEF95(0.5-8)* as the two features and has a total of two nodes. The sensitivities of this tree in detecting W and N1 stages are 88% and 78.3% respectively.

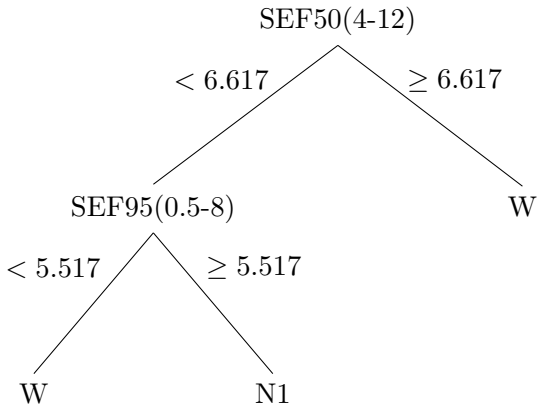


Figure 5.6: *Peripheral decision tree to discriminate between Wake and N1 sleep stages.*

Wake vs. N2

This tree is the only exception to the constraint of having a maximum of two levels in the tree. It has an extra level to achieve the required levels of accuracy. However, it still has a maximum of three nodes and uses *sigma/beta* and relative powers in *gamma* and *delta2* bands as the features. It has a sensitivity of 92% and 92.6% for detecting N2 and

Wake respectively. This test is used when the current state of the state machine is either W or N2.

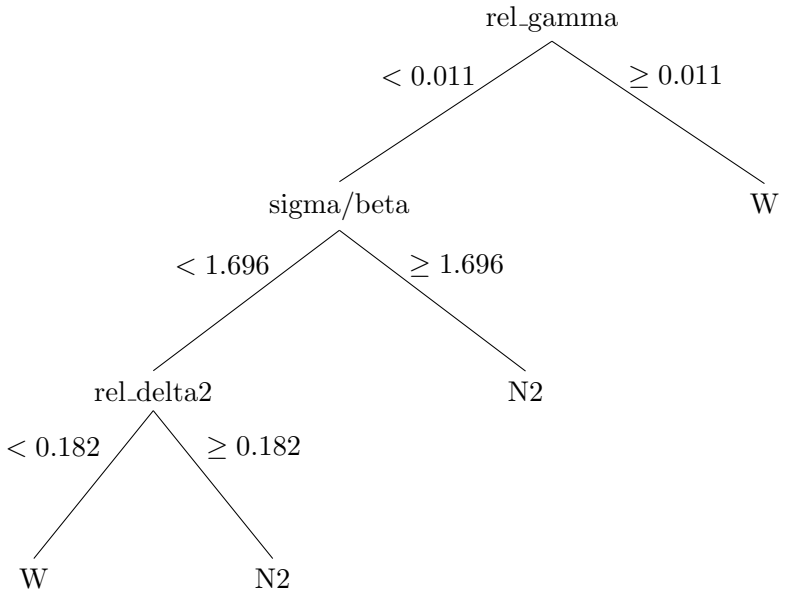


Figure 5.7: *Peripheral decision tree to discriminate between Wake and N2 sleep stages.*

Wake vs. N3

This is a simple decision tree with only one node and tests the value of relative power in *gamma* frequency band. It has a sensitivity of 97.6% for detecting N3 and 92.4% for detecting Wake.

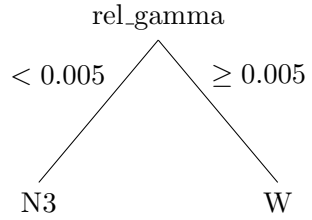


Figure 5.8: *Peripheral decision tree to discriminate between Wake and N3 sleep stages.*

Wake vs. REM

This peripheral test uses relative power in *delta2* and *SEF50* in 8-16 Hz frequency band to determine if the epoch is of Wake or REM stage. The tree has a total of two decision nodes and detects Wake and REM with sensitivity of 90.4% and 87.5% respectively.

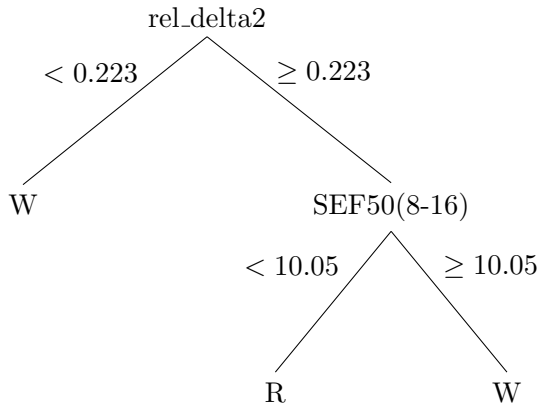


Figure 5.9: *Peripheral decision tree to discriminate between Wake and REM sleep stages.*

N1 vs. N2

This test uses relative powers in *gamma* and *alpha1* bands with two nodes to classify an epoch as either N1 or N2. It detects N2 with a sensitivity of 97.7%. However, its N1 sensitivity is low at only 44.8%.

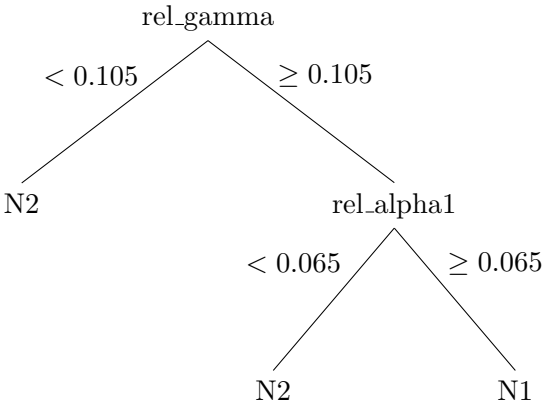


Figure 5.10: *Peripheral decision tree to discriminate between N1 and N2 sleep stages.*

N1 vs. N3

This tree uses the spectral power ratio between *beta* and *delta* bands and *SEF50* in 0.5-8 Hz range with two decision nodes to classify an epoch as N1 or N3. It detects N1 with sensitivity of 94.3% and N3 with sensitivity of 96%. It is used to test if the epoch under analysis is to be classified as N3 when the current state is N1 or if it is to be classified as N1 when the current state is N3.

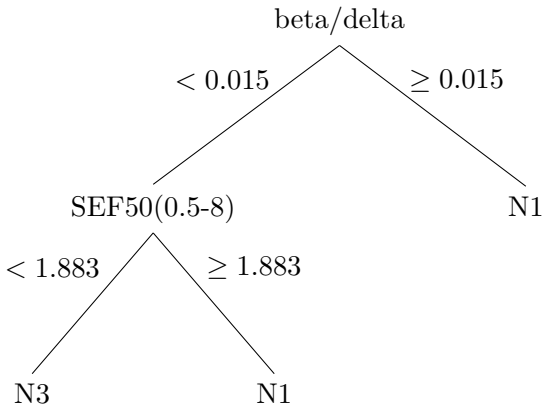


Figure 5.11: *Peripheral decision tree to discriminate between N1 and N3 sleep stages.*

N1 vs. REM

This peripheral tree detects REM epochs with a sensitivity of 91.8%. Its sensitivity for N1 is low at 43.4%. This is to be expected due to the EEG similarities between these sleep stages. It has two decision nodes and uses *SEF50* in 8-16 Hz and 0.5-30 Hz bands as the two features.

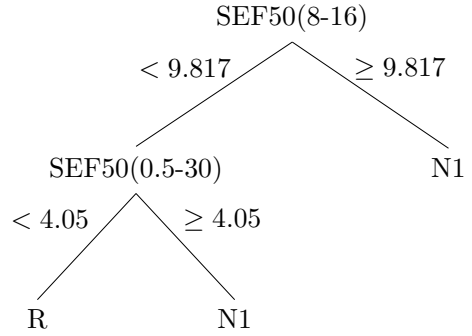


Figure 5.12: *Peripheral decision tree to discriminate between N1 and REM sleep stages.*

N2 vs. N3

This tree uses only one decision node with *SEF50(0.5-30)* as the feature. It classifies N2 and N3 epochs with sensitivity of 89.8% and 71.9% respectively.

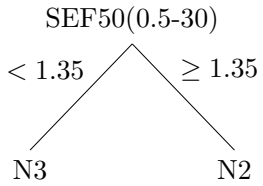


Figure 5.13: *Peripheral decision tree to discriminate between N2 and N3 sleep stages.*

N2 vs. REM

This peripheral test uses relative *theta* power and the ratio of *sigma/beta* powers with two decision nodes to classify an epoch as either N2 or REM. Its sensitivity for detecting N2 and REM are 91.1% and 80.5% respectively.

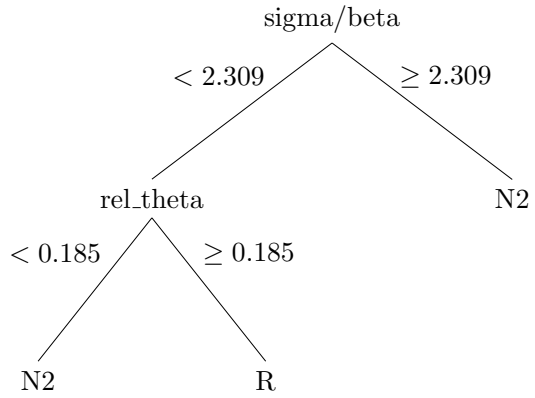


Figure 5.14: *Peripheral decision tree to discriminate between N2 and REM sleep stages.*

N3 vs. REM

This test uses relative *gamma* power with only one decision node and classifies N3 and R epochs with sensitivity of 96.7% and 92.2% respectively.

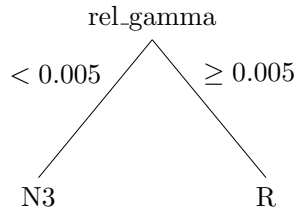


Figure 5.15: *Peripheral decision tree to discriminate between N3 and REM sleep stages.*

5.4.3 Final set of features

Of the 27 features in the reduced feature set (Table 5.2), only 18 were needed to design all the decision trees. This final set of features is listed in Table 5.3.

Table 5.3: *Final list of the most relevant features for use in the sleep staging algorithm.*

Features			
rel. delta2	rel. alpha1	SEF50(8-16)	SEF50(0.5-30)
rel. beta	rel. theta	SEF50(4-12)	SEFd(8-16)
rel. sigma	beta/delta	SEF95(4-12)	SEFd(0.5-30)
rel. gamma	sigma/beta	SEF50(0.5-8)	
rel. alpha	beta/alpha	SEF95(0.5-8)	

rel. - relative power in a frequency band with respect to the total bandwidth (0.5-50 Hz)

5.5 Results

Ordering of decision trees

In order to determine the optimum internal order of evaluation of the peripheral decision trees, all the different combinations were run in MATLAB using the training data set. The resulting sensitivity and selectivity of the algorithm were calculated for every single iteration. The harmonic mean of these two metrics gives the F-score which is used to compare the performances in different iterations.

$$F\text{–}score = 2 \times \frac{Sensitivity \times Selectivity}{Sensitivity + Selectivity} \tag{5.4}$$

The ordering or trees that results in the highest F-score is then used to evaluate the performance on the training and test data sets.

Training data results

Of the 10130 epochs in the training set, the best performance of the algorithm resulted in 7916 epochs being correctly classified giving an overall accuracy of 78.14%. The detection performance for all stages, except N1 and N3, showed a sensitivity of more than 80% and is shown in Table 5.4.

The performance of the algorithm is further validated by performing leave-one-out cross validation. In this method, the classifier is trained using a large part of the training data and tested with the remaining data from the training set. For this, the training set is divided into ten equal parts. In each iteration, nine parts (90% of data) are used to train the algorithm and the remaining part is used to validate its performance. As a result, there are a total of ten iterations. Each iteration uses a separate section of data for testing hence covering the entire test database at the end. This method resulted in an average accuracy of 80.3% with a standard deviation of only 3.3% between results of different iterations.

Table 5.4: *Algorithm performance using the training data set*

		ALGORITHM						
		W	**N1**	**N2**	**N3**	**R**	**Sen(%)**	**Sel(%)**
REFERENCE	**W**	1448	52	99	9	73	86.1	86.0
	N1	129	123	222	0	241	17.2	54.0
	N2	80	34	3763	215	260	86.5	77.8
	N3	11	0	573	1264	3	68.3	84.9
	R	16	19	177	1	1318	86.1	69.6

Testing data results

The test dataset consists of 10135 epochs in total of which 7440 were correctly classified by the algorithm with an overall accuracy of 73.41%. This is, expectedly, slightly lower than the accuracy obtained using the training set. The results for each sleep stage are shown in Table 5.5. Comparing the sensitivity in each sleep stage with that obtained using the training set, it can be seen that the accuracies for stages W, N1 and N2 are very similar. However, there is a noticeable reduction in the sensitivity for REM stage and an increase in N3 accuracy as well.

Table 5.5: *Algorithm performance using the test data set*

		ALGORITHM						
		W	**N1**	**N2**	**N3**	**R**	**Sen(%)**	**Sel(%)**
REFERENCE	**W**	1584	87	121	6	103	83.3	81.6
	N1	178	116	158	16	197	15.2	40.7
	N2	116	26	3175	399	183	81.4	71.7
	N3	30	0	514	1534	4	73.7	78.2
	R	34	56	360	7	1031	69.3	67.9

5.6 Discussion

It is generally difficult to compare the performance of algorithms that have been evaluated using different, and sometimes private, databases. It is for this reason that DREAMS Subjects database, which is freely available, was used to test the algorithm in this chapter. However, since this database is fairly recent there is no other algorithm yet that has been evaluated using this database.

Until recently, the most popular sleep database has been the PhysioNet Sleep EDF database [4]. This comprises of 8 overnight sleep recordings. A superset of this database,

the Sleep EDF Expanded database [5], is now available with 61 sleep recordings (see Appendix A). The performance of the algorithm proposed in this chapter is thus also evaluated using signals from this newer database to enable comparisons with other algorithms. The entire database is split into training and test sets of 30 and 31 subjects respectively. The overall accuracy using the training set is 82.22% while that using the test is 78.85%. The sensitivity in other sleep stages were similar with poor detection rate of N1 stage. This is not unexpected since certain spectral similarities between N1, REM and Wake stages are well documented [6], [7].

The algorithm proposed in this chapter is the first algorithm evaluated on *all* recordings from the PhysioNet Sleep EDF Expanded database. Only four other methods have used the Sleep EDF Expanded database for performance evaluation. Of these, Yaghouby et al. [8] obtained similar results to this method but used only a subset of the complete database (*ST* subjects). Sanders et al. [9] also used only the *ST* subjects and reported an overall accuracy of 75%, which is lower than that obtained in this work. Rodriguez-Sotelo et al. [10] used both *SC* and *ST* subjects. For *SC* subjects, they reported a maximum accuracy of 80% for individual test subjects separately. However, this dropped to 51% when these subjects were combined. They also used *ST* subjects for validation separately which resulted in a lower accuracy. Finally, Aboalayon et al. [11] also used a subset of this database however their method only discriminated between Wake and N1, and not all the stages of sleep.

There is potential for further improvement to the proposed algorithm. At present, all the binary peripheral decision trees are designed with equal misclassification cost for either stage. For example, the peripheral decision tree corresponding to N1 *vs.* N2 classification is the same whether the current state is N1 or N2. Initial work involving training of trees with different misclassification cost based on the current state has shown promising results with improved classification accuracy. Further, the results can also potentially be improved by adding better discriminatory features.

The core and peripheral decision trees are constrained to have a limited number of nodes in their longest path. Although, this limits the maximum accuracy that can be achieved, it is done to realise an algorithm with smaller processing requirements making it suitable for being used in a wearable environment where limited processing resources are available.

Overall the results in this chapter suggest that the approach of combining state machines and decision trees in the context of sleep staging can be highly useful for the classification of sleep. This approach allows for the use of multiple small decision trees that get activated depending on the current sleep stage. It also results in better usage of processor resources on which the algorithm may be implemented. This is because only a subset of features are computed each time depending on the current sleep stage. Further, since the starting decision trees change based on the current state, not all of them are re-

quired at all times. This saves several nodes of comparison that would have been required in an approach using conventional decision trees alone. Although the algorithm showed a good overall performance, sensitivity in N1 stage was found to be lacking. Nevertheless, the approach presented in this chapter will be highly useful for designers of automatic sleep scoring systems and can be further improved with the use of more discriminative features and better design of decision trees.

References

[1] I. Millington and J. Funge, *Artificial Intelligence for Games*. CRC Press, 2009.

[2] University of MONS - TCTS Laboratory. (2015) The DREAMS Subjects Database. [Online]. Available: http://www.tcts.fpms.ac.be/~devuyst/Databases/DatabaseSubjects/.

[3] L. Breiman, J. Friedman, C. J. Stone, and R. Olshen, *Classification and Regression Trees*. CRC Press, 1984.

[4] PhysioNet. (2013) Sleep-EDF Database. [Online]. Available: http://www.physionet.org/physiobank/database/sleep-edf/.

[5] ——. (2014) Sleep-EDF Database [Expanded]. [Online]. Available: http://www.physionet.org/physiobank/database/sleep-edfx/.

[6] M. Corsi-Cabrera, Z. Munoz-Torres, Y. del Rio-Portilla, and M. A. Guevara, "Power and coherent oscillations distinguish REM sleep, stage 1 and wakefulness," *Int. J. Psychophysiol.*, vol. 60, no. 1, pp. 59–66, 2006.

[7] R. Bódizs, M. Sverteczki, and E. Mészáros, "Wakefulness-sleep transition: Emerging electroencephalographic similarities with the rapid eye movement phase," *Brain Res. Bull.*, vol. 76, no. 1, pp. 85–89, 2008.

[8] F. Yaghouby, P. Modur, and S. Sunderam, "Naive scoring of human sleep based on a hidden markov model of the electroencephalogram," in *IEEE EMBC*, Chicago, August 2014.

[9] T. Sanders, M. McCurry, and M. Clements, "Sleep stage classification with cross frequency coupling," in *IEEE EMBC*, Chicago, August 2014.

[10] J. L. Rodriguez-Sotelo, A. Osorio-Forero, A. Jimenez-Rodriguez, D. Cuesta-Frau, E. Cirugeda-Roldan, and D. Peluffo, "Automatic sleep stages classification using EEG entropy features and unsupervised pattern analysis techniques," *Entropy*, vol. 16, no. 12, pp. 6573–6589, 2014.

[11] K. Aboalayon, H. Ocbagabir, and M. Faezipour, "Efficient sleep stage classification based on EEG signals," in *IEEE LISAT*, New York, May 2014.

6 Integrated circuit design and implementation of an automatic sleep staging algorithm

6.1 Introduction

An automatic sleep staging algorithm was presented in Chapter 5 which works by extracting spectral features from a single EEG channel and using contextually aware decision trees to classify them into one of the sleep stages. The algorithm has been specifically designed to be suitable for a low-power hardware implementation by using low complexity features and a classifier that is mostly idle with few active sections only when needed. This chapter presents the complete integrated circuit design and implementation of the algorithm using AMS 0.18μm process technology.

The architecture of the sleep staging algorithm hardware has four main blocks:

1. Input Controller
2. Fast Fourier Transform
3. Feature Calculation
4. Classifier

The *input controller* buffers up the EEG samples as they are received from external sources. This data is passed on to a block that computes the *Fast Fourier Transform* of 2-second subepochs and saves their magnitudes. These magnitudes are then used to perform the *feature calculation*. Finally, the *classifier* takes in the features for each epoch and assigns a sleep score to it which can be read out externally. The following sections explain the hardware implementation of these blocks in detail and discuss the design considerations that were taken into account during their implementation.

6.2 Input Controller

The sampling frequency of the EEG signals used by the sleep staging algorithm is 256 Hz. This means that a new sample is available at the input every $1/256th$ of a second. The

samples need to be buffered until 512 of them are received i.e. equivalent to the number of samples in a 2-second subepoch. It is for this storage and control of data samples that an *input controller* block is needed.

The *input controller* block, shown in Figure 6.1, has been designed to be able to receive data from two sources. Firstly, it gets data from an on-chip analogue front end (AFE) that includes an ADC. Secondly, it can also get digitised bits of data directly on digital input ports. This latter option is used so that it is possible to test the system directly bypassing the AFE and also make it compatible with other front ends. The source of input can be selected using a a `sel` input that controls a multiplexer. Setting this input would use data from an on-chip ADC while resetting it would use data from digital input ports.

Regardless of the mode chosen, an input data sample is valid only when the `valid_in` control signal at the input is also high. The sample is read when `valid_in` goes from high to low and stored at the appropriate place in the register bank. An internal counter generates the address of the location where the sample is to be stored in the bank. This counter is initially zero when the circuit resets and is incremented each time a valid sample is received. When 512 samples are received the counter reaches its maximum limit and is reset to zero to start over again. At the same time a `valid_out` pulse is generated from the input controller block to the FFT block which acts as an instruction to read data and start computation. This frees up the register bank so that the *input controller* can continue receiving data for the next subepoch without any interruption even while computations are taking place by other blocks.

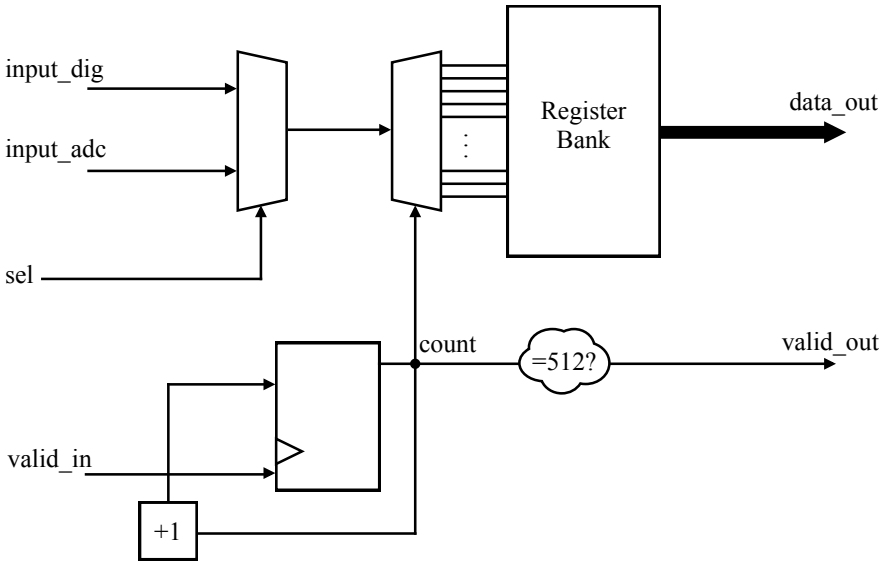


Figure 6.1: *Block diagram of the Input Controller.*

6.3 Fast Fourier Transform

6.3.1 Overview

The Discrete Fourier Transform (DFT) is used to extract the spectral properties of a signal. The calculation of N-point DFT requires N^2 multiplications which can be extremely inefficient and time-consuming. Fast Fourier Transform (FFT) is an algorithm to speed up the computation of DFT by making use of certain symmetrical properties, greatly reducing the number of multiplications required, resulting in an algorithmic complexity of *O(N log N)* compared to *O(N^2)* when using the original DFT. The FFT of an input signal x is computed as shown below, where $X(k)$ is the transformed output at index k and N is the length of the signal.

$$X(k) = \frac{1}{N}\sum_{n=0}^{N-1} x(n)e^{\frac{-j2\pi nk}{N}} = \frac{1}{N}\sum_{n=0}^{N-1} x(n)\omega_N^{nk} \tag{6.1}$$

The most popular algorithm to compute FFT is the Cooley-Tukey FFT algorithm [1]. Radix-2 Decimation in Time (DIT) is the simplest variant of the Cooley-Tukey FFT algorithm. This is a recursive algorithm that breaks down the entire calculation into a number of 2-point DFTs. Figure 6.2 shows the implementation of an 8-point FFT using the Radix-2 DIT algorithm. There are eight input samples that are rearranged in a certain order. The input values are combined in pairs to perform some mathematical operations resulting in a pair of complex values as outputs. For the eight input samples, four such pair-wise operations are needed at the first level. The next level then uses the outputs from the computations of previous level to perform similar mathematical operations using the inputs in a different order. This process continues for three levels and the results from the final stage are the required FFT coefficients.

Each unit of mathematical operation using two input values is commonly known as the *butterfly* operation. Assuming that the two input values for this operator are complex numbers A and B while ω is a *twiddle factor* that depends on the input values and level of FFT is being computed, then the two complex output values X and Y are as follows.

$$X = A + B\omega \tag{6.2a}$$

$$Y = A - B\omega \tag{6.2b}$$

For an N-point FFT, where $N = 2^n$, there are $N/2$ *butterfly* operations performed at each level and there are a total of n levels. As a result, $Nn/2$ *butterfly* operations are needed to perform the complete FFT. The *twiddle factor* for each computation can be represented as w_N^k where $0 \leq k < N$. The *twiddle factors* can either be computed on the fly or stored in memory and accessed using a lookup table.

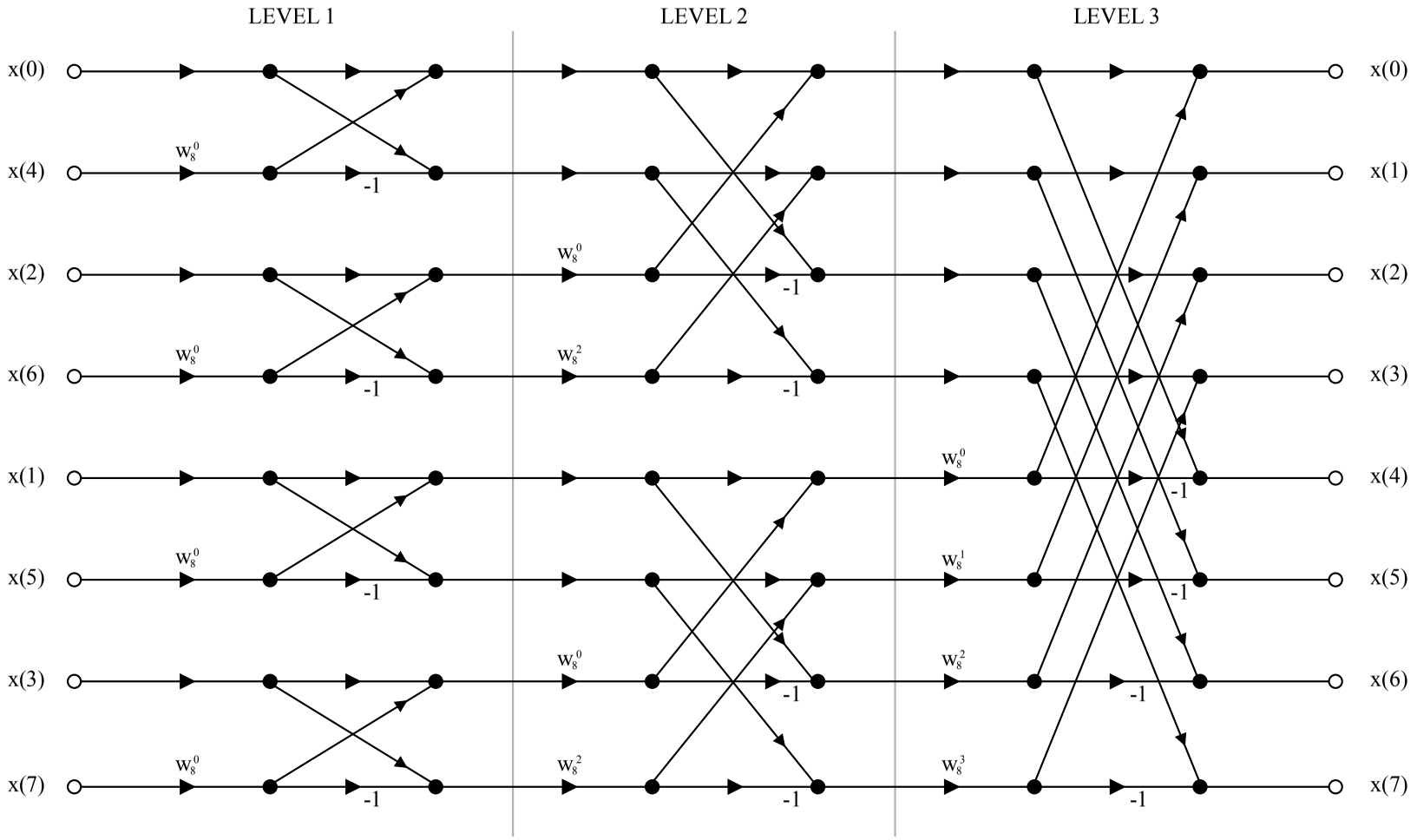


Figure 6.2: *8-point radix-2 Decimation in Time FFT algorithm.*

The FFT is a complex arithmetic transform requiring several multiplications, additions and registers to hold the results before the final result can be produced. With a large number of multiplications needed it is not feasible to use a separate multiplier for each computation in hardware since that will result in a design with huge area, complexity and redundancy. Further, since the FFT algorithm is recursive, all *butterfly* operations are independent of each other in every stage. This means that a single *butterfly* unit can be used to perform the computations for different input pairs in a multi-cycle operation. Consequently, the total number of cycles required to get the final result will be $Nn/2$ assuming that each *butterfly* operation can be performed in one clock cycle.

For the specific sleep staging algorithm being implemented in this chapter, a 512-point FFT is required. This is implemented using fixed-point representation of numbers. The integer and fraction bits of the fixed point representation are decided by performing a large number of MATLAB simulations. The maximum absolute value of a sample within the EEG database is under 500, which can be represented by 10 bits (for a signed representation). Using this as the minimum, multiple simulations were carried out to find out the number of integer bits needed that would prevent overflows resulting from a large number of multiplication and addition operations that are performed in FFT. From this, it was determined that 12 bits would be sufficient to represent the integer part without overflows.

The fractional bits in the fixed-point numbers determine the accuracy of the overall output. Using 12 bits for integer, the MATLAB simulations for FFT were performed again this time sweeping the number of fractional bits from 1 to 20 and the RMS error determined for each. The results are shown in Figure 6.3. It can be seen that increasing the number of fractional bits steadily reduces the RMS error until a point after which any increase in the number of bits results in a very small decrease in error. That point corresponds to 12 fractional bits. Together, this results in a 24-bit fixed-point number used for the FFT implementation.

Figure 6.4 shows the block diagram of the FFT implementation at the top level. The input data to be transformed is read when a `valid_in` signal indicates the availability of samples at the port. These data samples are rearranged immediately in the *bit reversal* block and stored in the *register bank*. An *address generation* module counts the number of cycles and the levels for which the calculations are being performed and uses them to compute two addresses so that correct data can be fetched from and saved to the register banks. This address is also used to calculate the *twiddle factors* for each stage of calculation. The *butterfly* module uses the fetched data and twiddle factors to compute its output, both real and imaginary parts, which are then saved in the register banks overriding the previous values. This process is repeated in each clock cycle until the final output is available after 2304 clock cycles. The design of each module in the FFT block is explained in detail in the following sections.

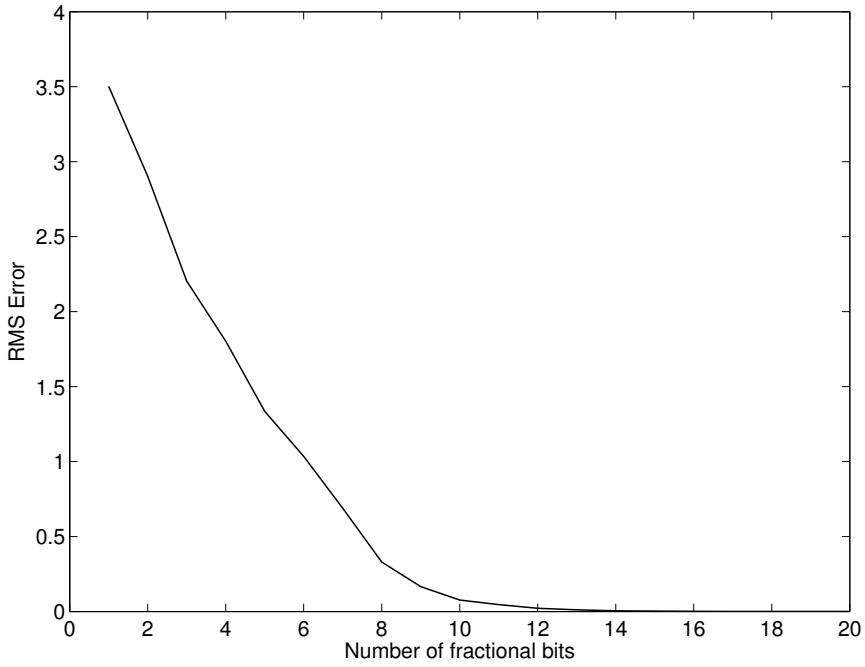


Figure 6.3: *RMS error with respect to the number of fractional bits in a fixed-point number.*

6.3.2 Data Input & Bit Reversal

In order for the FFT algorithm to follow the computations in Figure 6.2 the inputs need to be rearranged at the beginning. This requirement of the DIT algorithm ensures that the output samples are available in the correct order when the FFT computation is complete. As can be seen from Figure 6.2, the indices of the input data follow a bit reversal pattern. This means that the new position of an input data sample is determined by flipping the binary bits of its original index and the resultant binary value gives the new index. Some examples of bit reversal for a 512-point FFT are shown in Table 6.1.

Bit reversal can be performed on the hardware to determine the new index for each data sample. However, since the number of points in the FFT being implemented here is fixed, the hardware for reordering the input samples can also be fixed. In terms of hardware, this does not add any logic and translates to data bits being wired to their new locations as shown in Figure 6.5.

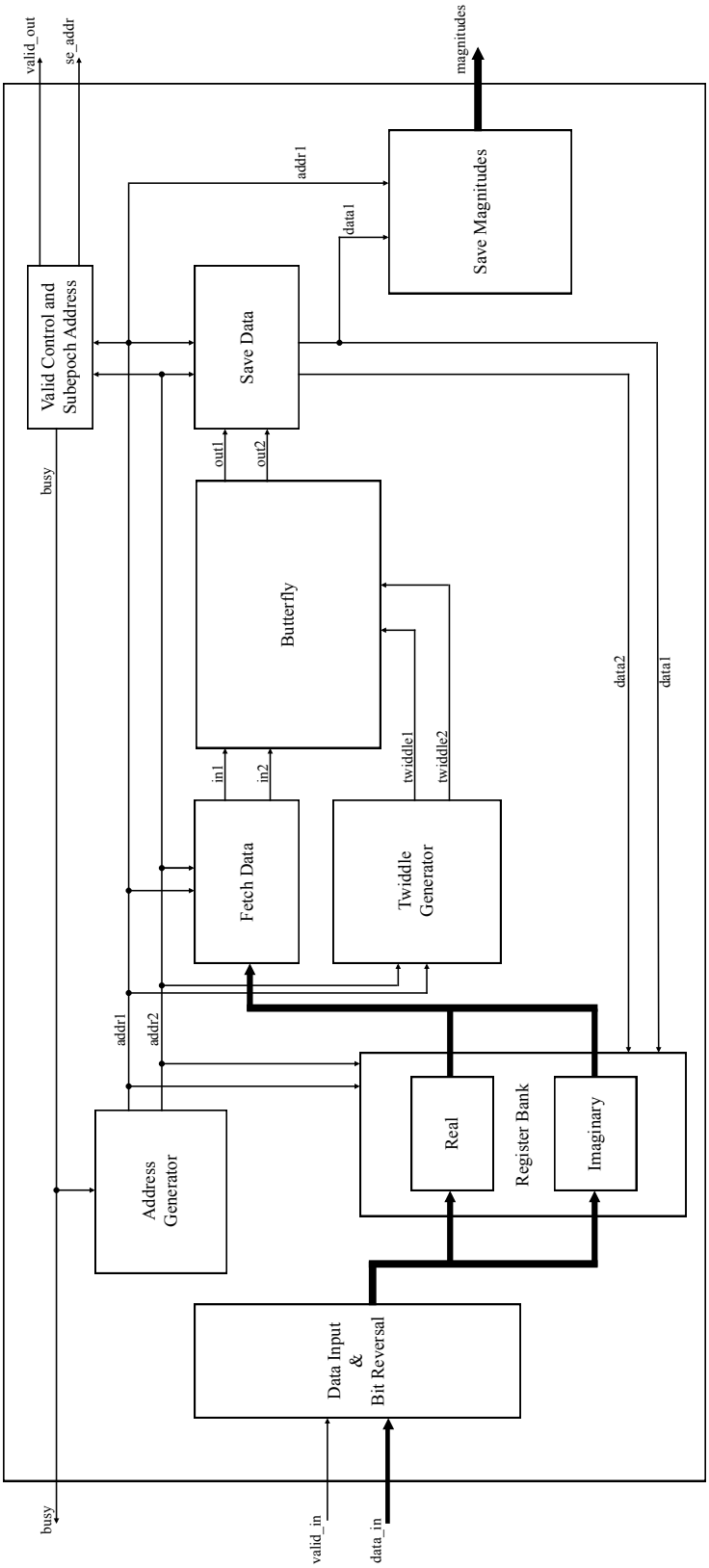


Figure 6.4: *Top level diagram of the FFT block.*

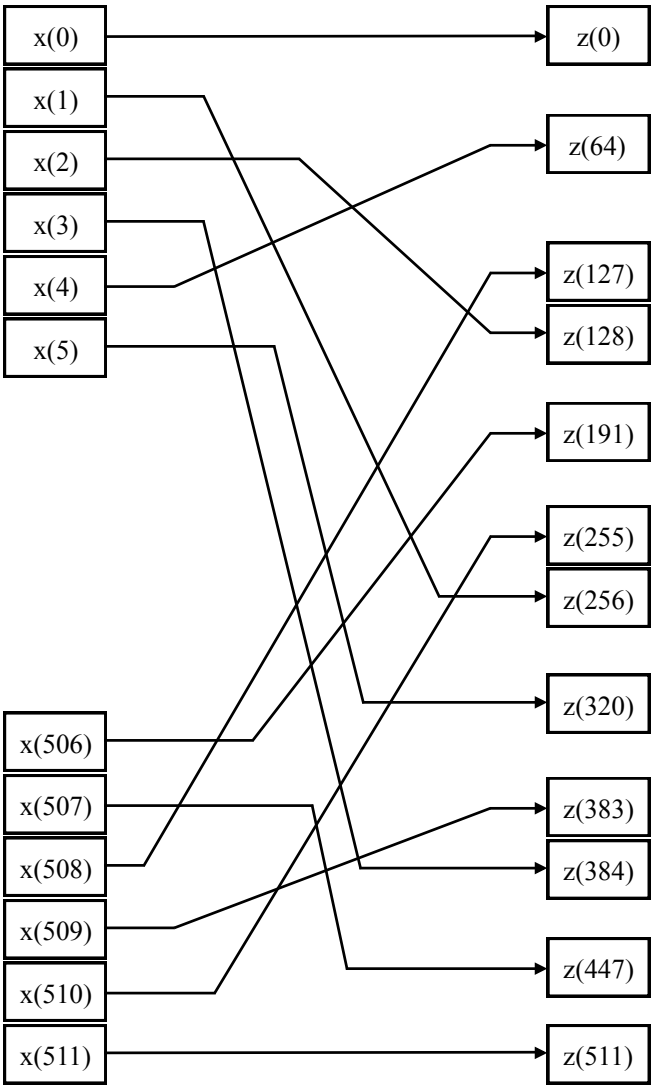


Figure 6.5: *Block diagram of the Data Input & Bit Reversal module.*

Table 6.1: *Bit reversal examples for 512-point FFT.*

Input index		Bit reversed index	
Decimal	Binary	Binary	Decimal
0	000000000	000000000	0
1	000000001	100000000	256
2	000000010	010000000	128
3	000000011	110000000	384
4	000000100	001000000	64
5	000000101	101000000	320
...	...	...	...
506	111111010	010111111	191
507	111111011	110111111	447
508	111111100	001111111	127
509	111111101	101111111	383
510	111111110	011111111	255
511	111111111	111111111	511

6.3.3 Register Banks

The FFT computation involves complex numbers having real and imaginary parts. They are both represented as *signed* fixed-point numbers. This means that each of the 512 data samples correspond to one real and one imaginary part resulting in 1024 values. These values need to be stored for intermediate processing and for final output. For this, two register banks are used; one for real and one for imaginary parts of the complex numbers. The register banks are identical and are designed to be 24-bit wide with a depth of 512.

At the start of a new FFT computation, the reordered real-valued data inputs are stored in the *real* register bank while the *imaginary* register bank is initialised to zero for all values. Each register within the bank also has an *enable* signal to control when data can be written on to it. Initially, when the input is valid, the enable signals of all registers are set to high so that all the spaces can be initialised in the bank with the input samples or zeroes. After that, an internal `busy` signal indicates that an FFT computation is ongoing and the values in only two registers will be overwritten in each bank at the end of every clock cycle.

The two register banks store the reordered input values (and zeroes) initially. Once the Transform begins, they hold the intermediate values, overwriting the previous ones. Finally, at the end of the Transform, the output of the FFT computation is also available in these register banks in the correct order and can be read out when indicated by the `valid_out` signal.

6.3.4 Address Generator

This module keeps a count of how many cycles and levels of computations have been performed. Based on the current cycle and level count, it generates addresses for the register banks to retrieve data from correct locations for each *butterfly* computation and subsequently save the result on to the same location. It also provides information to the *twiddle generator*, *save magnitudes* and *valid control* modules.

Consider the 8-point FFT operation shown earlier in Figure 6.2. In this example, N is 8 therefore the number of computation cycles required in each level is $N/2$ while the number of levels is $\log_2(8) = 3$, designated as n. The number of cycles range from 0 to $N/2 - 1$ while the number of levels range from 0 to $n - 1$.

Each cycle of *butterfly* computation uses two input values. During the first level, these are adjacent values in the register bank i.e. in the first cycle of first level $x(0)$ and $x(1)$ are used, in the second cycle $x(2)$ and $x(3)$, and so on. This relationship of inputs for each cycle is not constant and changes during every level of computation. For example, in the first cycle of second level inputs $x(0)$ and $x(2)$ are used while the same cycle in the last level uses $x(0)$ and $x(4)$ inputs. Due to this, a pair of addresses need to be generated based on the current FFT cycle and level to provide correct input values. If the current cycle and current level are assumed to be C and L, then the two addresses $ADDR1$ and $ADDR2$ can be determined as follows [2]:

$$ADDR1 = RotateLeft(2C, L) \tag{6.3a}$$

$$ADDR2 = RotateLeft(2C + 1, L) \tag{6.3b}$$

The address generator hardware, shown in Figure 6.6, consists of two incrementing counters. First is the *cycle counter* which is incremented every clock cycle (up to its maximum value after which it is reset) and the second is the *level counter* is incremented when *cycle counter* reaches its maximum value. Using the output from the two counters an *address calculation* submodule performs the circular left shift on the binary representation of $2C$ and $2C + 1$ by L bits. The resultant outputs are the two addresses which are passed to the other modules within the FFT block.

6.3.5 Fetch Data

This module is used to fetch correct data from the two register banks for the *butterfly* computation in each cycle. It gets two addresses from the *address generator* module and uses them with a set of multiplexers to output the stored values at those two addresses in both real and imaginary register banks as shown in Figure 6.7.

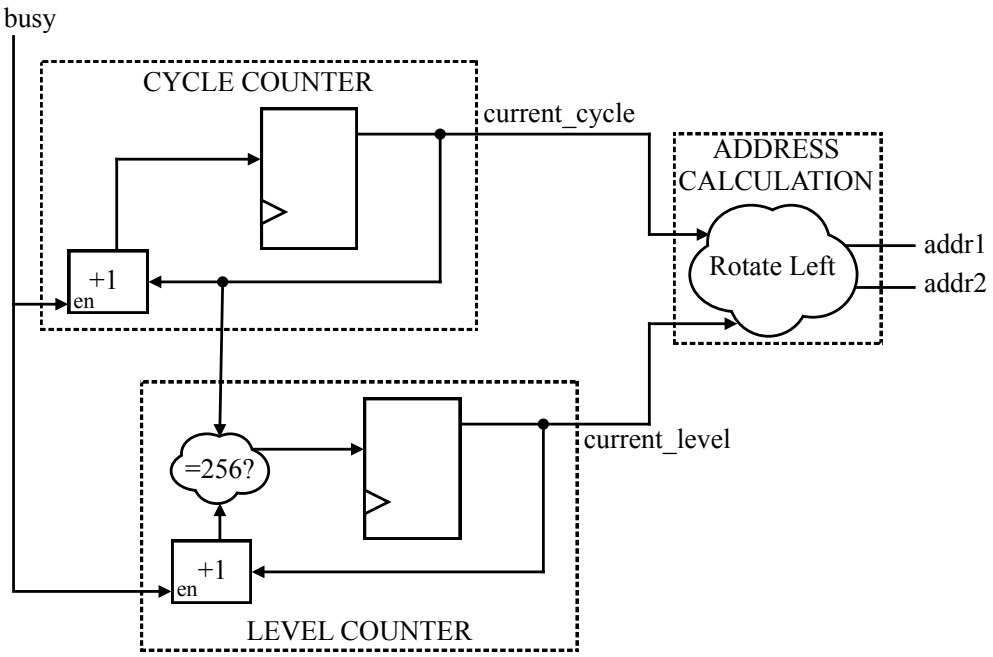


Figure 6.6: *Block diagram of the FFT Address Generator module.*

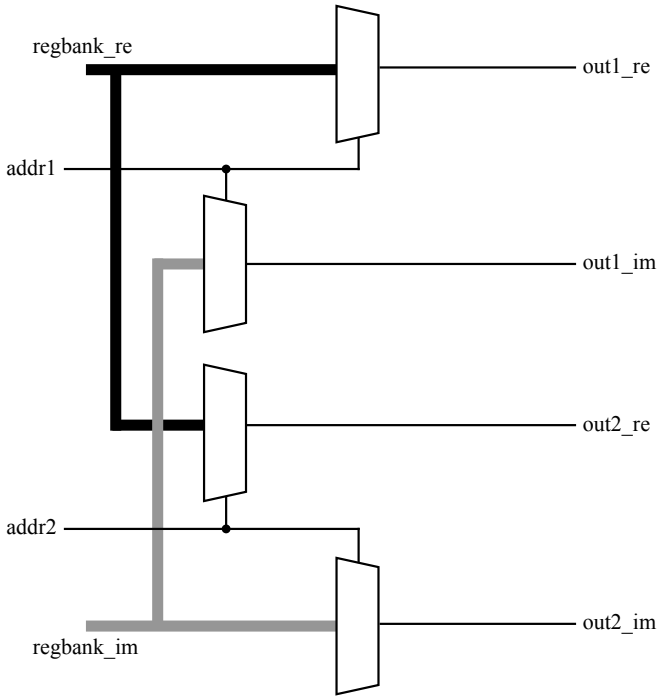


Figure 6.7: *Block diagram of the FFT Fetch Data module.*

6.3.6 Twiddle Generator

The twiddle factor needed for each *butterfly* computation depends on the cycle and level of operation. From the earlier FFT definition in Equation 6.1, the twiddle factors are defined as:

$$e^{\frac{-j2\pi nk}{N}} = \omega_N^{nk} \tag{6.4}$$

For a 16-point FFT, Table 6.2 shows the twiddle factors for all *butterfly* computations as well as the corresponding cycle and level numbers. During the first level, $N = 2$ for each *butterfly* operation since the output is only dependent on the two inputs. The next level uses four inputs to produce its output hence $N = 4$ followed by $N = 8$ in the next level and $N = 16$ in the final level.

Table 6.2: *Twiddle factors during each cycle and level of a 16-point FFT.*

Cycle Count		Level			
Decimal	Binary	0	1	2	3
0	0000	ω_2^0	ω_4^0	ω_8^0	ω_{16}^0
1	0001	ω_2^0	ω_4^0	ω_8^0	ω_{16}^1
2	0010	ω_2^0	ω_4^0	ω_8^1	ω_{16}^2
3	0011	ω_2^0	ω_4^0	ω_8^1	ω_{16}^3
4	0100	ω_2^0	ω_4^1	ω_8^2	ω_{16}^4
5	0101	ω_2^0	ω_4^1	ω_8^2	ω_{16}^5
6	0110	ω_2^0	ω_4^1	ω_8^3	ω_{16}^6
7	0111	ω_2^0	ω_4^1	ω_8^3	ω_{16}^7

All these twiddle factors can be represented as ω_{16} by simply multiplying the exponential numerator with a where $a = 16/N$ (using corresponding N for each level).

$$\omega_{16}^{ank} = \omega_N^{nk} \tag{6.5}$$

The resulting twiddle factors with the same exponential denominator of 16 are shown in Table 6.3.

Let z equal to the index (ank) of each twiddle factor in equation Equation 6.5. This index changes with the cycle and level of FFT computation and thus needs to be determined. IF C is the current cycle and L is the current level, the index z can be calculated by taking the $L + 1$ most significant bits of C and shifting it left by $n - L - 1$, where n is equal to the total number of levels.

$$z = ShiftLeft((L + 1\ bits\ of\ C), n - L - 1) \tag{6.6}$$

Table 6.3: *Twiddle factors during each cycle and level of a 16-point FFT revised to have the same denominator.*

Cycle Count		Level			
Decimal	Binary	0	1	2	3
0	0000	ω_{16}^0	ω_{16}^0	ω_{16}^0	ω_{16}^0
1	0001	ω_{16}^0	ω_{16}^0	ω_{16}^0	ω_{16}^1
2	0010	ω_{16}^0	ω_{16}^0	ω_{16}^2	ω_{16}^2
3	0011	ω_{16}^0	ω_{16}^0	ω_{16}^2	ω_{16}^3
4	0100	ω_{16}^0	ω_{16}^4	ω_{16}^4	ω_{16}^4
5	0101	ω_{16}^0	ω_{16}^4	ω_{16}^4	ω_{16}^5
6	0110	ω_{16}^0	ω_{16}^4	ω_{16}^6	ω_{16}^6
7	0111	ω_{16}^0	ω_{16}^4	ω_{16}^6	ω_{16}^7

Calculation and storage of twiddle factors

A twiddle factor ω_N^z is simply the polar representation of a complex number. It is evaluated as below:

$$\omega_N^z = \cos(\frac{2\pi z}{N}) - j\sin(\frac{2\pi z}{N}) \tag{6.7}$$

Computing the sinusoidal values on hardware is both complex and time consuming. A more efficient solution is to store the already computed values in a ROM from where they can be fetched using a look up table. For an N-point FFT, the total number of twiddle factors needed are $N/2$. Each twiddle factor itself consists of two values: a cos component and a sin component, corresponding to the real and imaginary values of the complex number respectively. As a result, the actual number of values to be stored in ROM are N.

For a 512-point FFT the number of constants to be stored is quite large. However, cos and sin values are related:

$$\cos(x) = \sin(x + \frac{\pi}{2}) \tag{6.8}$$

This means that a separate value for the sin component does not need to be stored and can be looked up by adding an offset to the cos index. This will reduce the number of constants to be stored down to $N/2$.

As an example, consider the twiddle factors needed in each cycle of the last level for a 16-point FFT in Table 6.4. In this case $N = 16$, hence $N/2$ cycles are needed. Observing the twiddle value in each cycle shows that the imaginary value at cycle C is the same as the real value at $C + N/4$. This illustrates the cos and sin relationship explained earlier. Further, the real values also repeat in reverse (with a negative sign) after $N/4$ cycles.

Hence, the second half of these values need not be stored and can be determined from the first $N/4$ values.

Table 6.4: *Twiddle factor values for a 16-point FFT.*

Cycle	Twiddle Factor	Twiddle Value
0	ω_{16}^{0}	1.0000 - 0.0000j
1	ω_{16}^{1}	0.9239 - 0.3827j
2	ω_{16}^{2}	0.7071 - 0.7071j
3	ω_{16}^{3}	0.3827 - 0.9239j
4	ω_{16}^{4}	0.0000 - 1.0000j
5	ω_{16}^{5}	-0.3827 - 0.9239j
6	ω_{16}^{6}	-0.7071 - 0.7071j
7	ω_{16}^{7}	-0.9239 - 0.3827j

As a result of these symmetry and periodicity properties, and knowing which quadrant the FFT computation is in, the number of constant values to be stored in ROM is reduced to only $N/4$. Hence, for the implementation of 512-point FFT only 128 constant values are needed to be stored.

Hardware implementation

Figure 6.8 shows the block diagram of the complete *twiddle generator*. The two inputs to this block are `current_cycle` and `current_level` from the *address generator*. These are used by the *index calculator* module to calculate the indices of the *sin* and *cos* components of the current twiddle factor. The two indices are used to fetch the values from a ROM which consists of 128 constant values. These values are the real and imaginary components of the current twiddle factor.

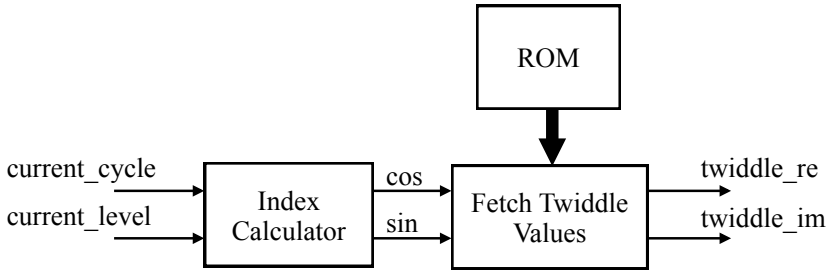


Figure 6.8: *Block diagram of the FFT Twiddle Generator module.*

6.3.7 Butterfly

The *butterfly* operation is the heart of the Cooley-Tukey FFT algorithm. It takes in two data points and performs certain arithmetic operations on them to produce two complex output values in every clock cycle. The arithmetic of this operation is shown in Figure 6.9 and can be broken down into three stages:

- The complex input B is multiplied by complex twiddle factor ω resulting in a complex output $B\omega$.
- The sum of complex numbers A and $B\omega$ results in the first output.
- The difference between complex numbers A and $B\omega$ gives the second output.

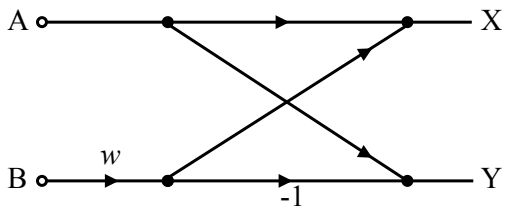


Figure 6.9: *A radix-2 butterfly operation with complex inputs A and B with twiddle factor ω.*

Complex multiplication

Multiplication of two complex numbers, $C = c + jm$ and $D = d + jn$, is performed as follows:

$$(c + id)(t + iu) = (a * c - b * d) + j(b * c + a * d) \tag{6.9}$$

In a fixed-point implementation, this requires four integer multipliers and three integer adders to multiply both real and imaginary parts of the two complex numbers. A block diagram of the implementation of this complex multiplier is shown in Figure 6.10.

Complex addition/subtraction

Addition of two complex numbers involves adding their real and imaginary parts. Similarly, complex subtraction is simply a matter of taking the difference between the real and imaginary parts of the two numbers. This arithmetic operation requires using two adders or subtractors and a block diagram of its implementation is shown in Figure 6.11.

$$C + D = (c + d) + j(m + n) \tag{6.10a}$$

$$C - D = (c - d) + j(m - n) \tag{6.10b}$$

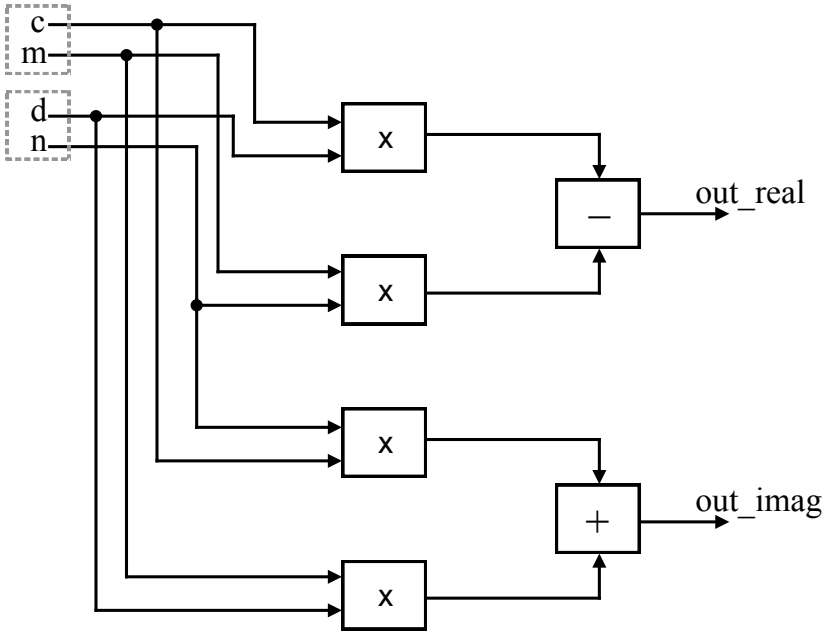


Figure 6.10: *Block diagram of the complex multiplier.*

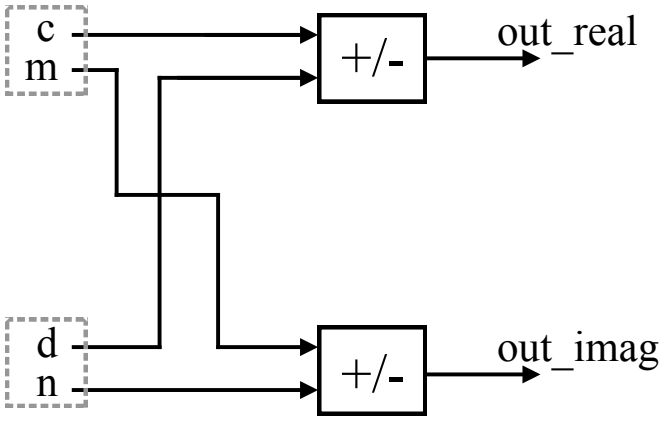


Figure 6.11: *Block diagram of the complex adder/subtractor.*

Butterfly operation

The *butterfly* module uses the three complex arithmetic components described above to produce its output. A functional diagram showing how these three components are put together in this combinatorial module is shown in Figure 6.12.

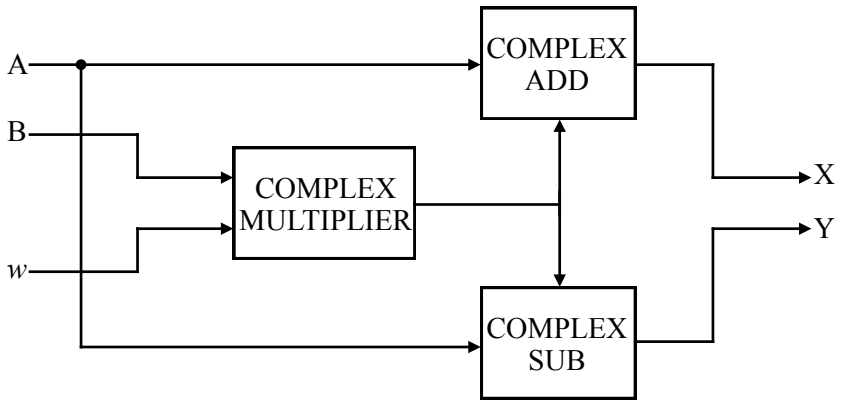


Figure 6.12: *Block diagram of the FFT Butterfly module.*

6.3.8 Save Data

Each computation in the *butterfly* module results in two outputs that need to be saved, either as intermediate values or as the final result in order to compute the magnitudes. The results are saved in the register bank at the same address from which they are read in each cycle. Consequently, the two addresses from the *address generator* are used by this module to generate an array of enable signals for the register banks. These enable signals ensure that the new values are written into the register banks only at the two generated addresses. Within the register banks, a multiplexer uses the two enable signals to direct the correct value into each register as shown in Figure 6.13.

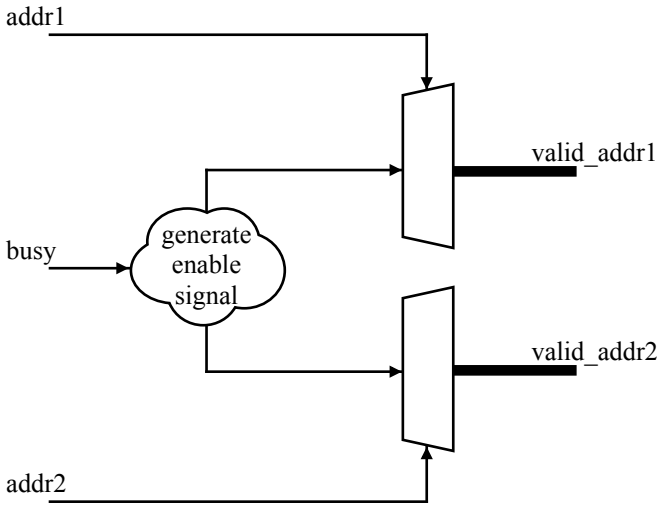


Figure 6.13: *Block diagram of the FFT Save Data module.*

6.3.9 Save Magnitudes

The output of the FFT computation is a set of complex numbers. The magnitude of these numbers correspond to the power at each frequency bin. It is the set of magnitudes that is used as feature input into the *feature calculation* block. Therefore, the magnitudes need to be computed and saved after the final results are available. One method to do this is to wait for all computations to complete, calculate the magnitudes and store them in a different set of registers. However, in the final level of computations, the magnitude at each point can be calculated immediately when the result at that point is available, saving few clock cycles. Further, the bandwidth of the EEG signal for feature calculation is up to 50 Hz and the FFT frequency resolution is 0.5 Hz. This means that only the first 100 magnitude coefficients need to be stored since others are not used.

The *save magnitudes* module consists of two submodules. The first is a *magnitude calculator* that uses the real and imaginary parts of the complex number computed by the *butterfly* operation to calculate its magnitude. The magnitude of complex number, $C = c + im$, is normally computed by summing the square of its real and imaginary parts and then taking the square root of this sum.

$$mag(C) = \sqrt{c^2 + m^2} \tag{6.11}$$

The square root operator involves complicated arithmetic hence the staging algorithm in this work was designed using features that were computed using the square of the magnitude so that this operator would not be needed at hardware level. The result is normalised by dividing with $N = 512$, which is simply a shift operation and does not require any datapath component.

$$mag(C) = \frac{c^2 + m^2}{512} \tag{6.12}$$

The second submodule within *save magnitudes* is a register bank that can store 100 values of magnitudes. The address where a value of magnitude is to be saved is provided by the *address generator*. The magnitudes are only calculated and saved when the final level of computation is being performed. During this, the enable bit of the currently addressed register is set to high when data at that address is available. Otherwise, all enable bits are zero during previous levels to prevent storing any magnitudes prior to the last level.

The complete implementation of calculating and saving the magnitudes is shown in Figure 6.14. It requires a total of two multipliers and an adder to calculate the magnitude value and some support logic to determine where this data needs to be saved in the magnitude register bank.

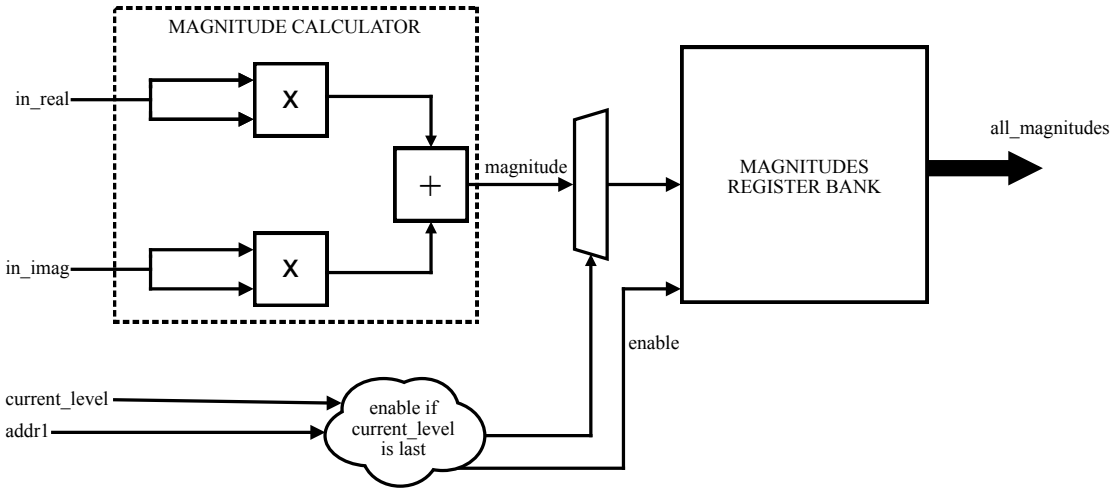


Figure 6.14: *Block diagram of the FFT Save Magnitudes module.*

6.3.10 Valid Out and Subepoch Address

When the last *butterfly* computation in the final level is complete, the register banks hold the resultant FFT coefficients and the magnitudes are available in the *magnitudes register bank*. At this time, a `valid_out` signal is raised to indicate that the FFT data is ready to be read by the *feature calculation* block. A counter within the *valid out* module keeps track of how many subepochs have been processed. It counts up to 15 and is incremented each time FFT computation is complete. This value is also passed on to the *feature calculation* block as the `subepoch_address`. Finally, the `busy` status signal is set to low which indicates to the input controller that the FFT block is free to start the next cycle of transformation if valid data is available at its input.

6.4 Feature Calculation

After the FFT block finishes its execution, the magnitude values saved at each frequency bin are used to calculate the features that are required for classification. These features were earlier listed in Chapter 5. Observing this list closely reveals that there are essentially two kinds of features that need to be calculated in different frequency bands. These are the power ratios and spectral edge frequencies. From the hardware point of view, the calculation is not dependent on the frequency band but only on the input i.e. the exact same hardware can calculate the power in two different frequency bands if the correct magnitudes are provided at the input. This means that generic blocks for the two kinds of features can be designed and instantiated with the required input corresponding to the frequency bands in which the features are desired.

6.4.1 Power calculation

A functional block diagram of the *power calculation* module is shown in Figure 6.15 and its design is explained by using alpha frequency band as an example which has the frequency range between 8-13 Hz. The FFT block produces an output with a granularity of 0.5 Hz which means that there exists a magnitude value at every 0.5 Hz interval. Consequently, between 8-13 Hz, there are 11 magnitude values that need to be added together in order to obtain the absolute power in this range. From the *magnitudes register bank*, these are the consecutive outputs from index 16 up to 26. These 11 values are wired as inputs to this module and are summed together to produce the final output. The *power calculation* module has been designed to be multi-cyclic so that only one adder is required which can be reused every cycle. This means that only one addition is performed in each cycle. Hence, it will take 11 clock cycles to produce a valid output for calculating power in alpha frequency band.

The power calculation module is normally idle and begins calculation on receiving ready signal from the FFT block. In the first clock cycle, a register is loaded with the first input from the magnitudes array. In the next cycle, the second input is added to this registered value and an internal counter gets incremented. This continues until the counter value reaches the maximum (equal to the total number of inputs) which also corresponds to the last input value being added. The value in the register is then the power value for the current subepoch and a valid signal is raised to signify that it is ready to be read. This value is valid as long as the magnitudes register bank of the FFT block is not overwritten by the results from the next subepoch. The resultant power values for all the fifteen subepochs of an epoch are also accumulated within this module. This is controlled by the subepoch address provided by the FFT block and is the feature that is eventually used by the classifier.

To make this module suitable for calculating power in various frequency ranges, a generic `INPUT_LENGTH` integer needs to be defined for each instance of the module. This integer is used to control the number of data inputs for the instance and also serves as the constant for comparison to check if all inputs have been added. An accumulator is also used at the end which sums up the power values for each subepoch when they are valid and thus holds the power value for an epoch.

6.4.2 Spectral edge frequency calculation

The calculation of spectral edge frequency, firstly, requires the sum of the total power in the desired frequency range. This is multiplied by the desired percentage (edge) and the resulting value is used as the maximum power threshold. The individual power values at each frequency range are then summed until this maximum value is reached. The corresponding frequency value at that point is the required spectral edge frequency.

In the sleep staging algorithm being implemented, spectral edge frequencies are re-

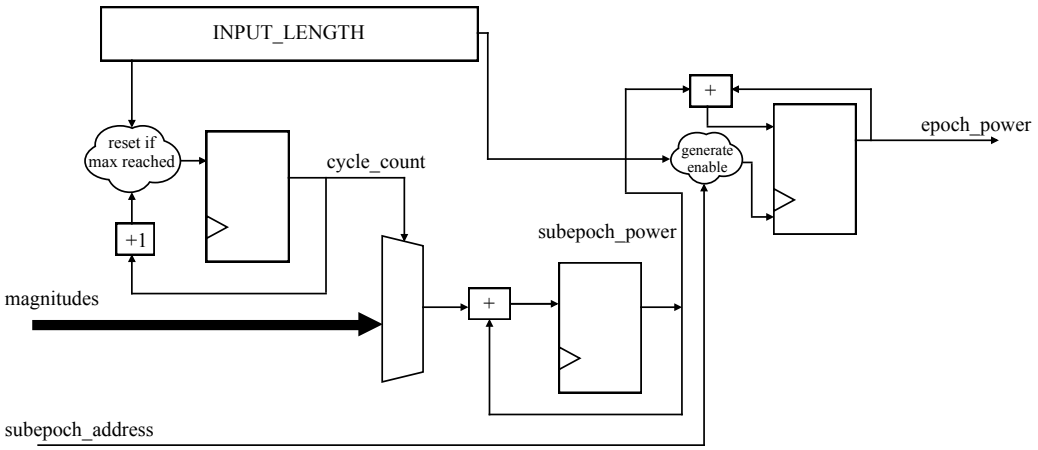


Figure 6.15: *Block diagram of the Power calculation module.*

quired at 50% and 95% (*SEF50* and *SEF95*). From the hardware point of view, the design of both *SEF50* and *SEF95* modules are exactly the same with one key difference. The total power value in *SEF50* is to be multiplied by 0.5 while in *SEF95* the multiplication factor is 0.95 to obtain the maximum threshold. In the former case, multiplication by 0.5 is essentially division by 2 which can be achieved easily by shifting one bit to the right. For multiplication by 0.95, an unsigned fixed-point multiplier is used with one input set to the constant value. This is the only difference in the design of the two modules while rest of the details that follow are exactly the same.

The calculation of *SEF95* in any frequency band requires the total power in that band. Therefore, a power calculation block (described in the previous section) is also used with the *SEF* module. The input to the module, in this case, are the FFT magnitudes in 8-16 Hz range (which is a total of 17 values). A valid signal enables the module after which the first step is the calculation of power. When the power calculation block finishes its computation it raises a flag which starts the iterative calculation of *SEF95*. The total power value is first multiplied by 0.95 to obtain the maximum threshold. This is followed by a multi-cycle operation in which the power at each frequency bin is subtracted from the total power until it goes below the threshold.

In the first cycle, the power value at 16 Hz is subtracted from the total power and stored in a register. The result is then compared against the result of the multiplier output. If it is found to be greater, the process continues and the next magnitude value (at 15.5 Hz) is subtracted from the earlier registered result and compared against the threshold. At the same time, 0.5 is subtracted every cycle from an initial frequency constant (16 in this case). This tracks the frequency up to which the comparison is performed. This way when the subtracted value falls below the threshold the spectral edge frequency will be the value stored in this register.

When the subtracted power value falls below the output of the multiplier, the compu-

tation of *SEF95* (or *SEF50*) is complete. The final value is stored in a register and a valid flag is raised. The sum of *SEF* values of all the subepochs are also accumulated in a register to be used by the classifier. Two versions of the *SEF* blocks have been designed. The first one is for the computation of *SEF95* which includes a multiplier while the second one is for *SEF50* computation and uses bit shifting to perform multiplication by 0.5. Both of these are made generic by defining `INPUT_LENGTH` and `INITIAL_FREQUENCY` integers. `INPUT_LENGTH` controls the number of data inputs while `INITIAL_FREQUENCY` sets the value from which the comparison begins and is equal to the highest frequency in any range for which the computation is taking place.

6.4.3 Data validity

Both *power calculation* and *spectral edge frequency calculation* modules take very small number of clock cycles to produce the result in comparison to the number of cycles needed for FFT computation. This means that magnitude values at their input will still be valid long after they have finished calculation. As a result they will start recalculating the same value again since inputs to them are valid. This behaviour is not desirable and the modules should stay in idle mode once they have finished calculation for a subepoch.

To tackle this, both the modules have an internal counter that determines the expected subepoch address at the input. This is compared with the current subepoch address that is provided by the FFT block. If the two addresses match, the feature is calculated. Once this is complete, the expected address counter is incremented and the valid output value is held until the subepoch address changes. The modules stay in idle mode until the next valid input is available with the address that matches the expected one. A block diagram of this operation is shown in Figure 6.16.

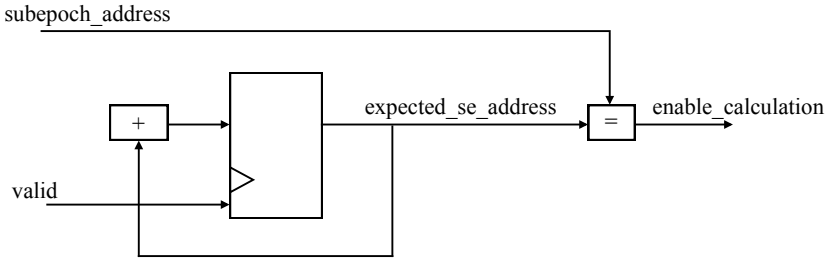


Figure 6.16: *Block diagram of the logic to check input data validity and prevent recalculation of the features for same inputs.*

6.4.4 Block level implementation

As discussed earlier in this section, there are two kinds of features that need to be calculated in multiple frequency bands: relative powers and spectral edge frequencies.

The relative power features are of two types. First, the relative power in a certain frequency band is calculated as the absolute power in that band divided by the total power in 0.5-50 Hz which is the total frequency range used. The second type is the ratio between two spectral bands, for example the ratio between alpha and beta frequency bands. In this case, the absolute power in alpha band is divided by that in the beta band. For both of these types the absolute power needs to be calculated in two frequency bands and their ratio taken. The power calculation module is used to calculate the absolute power while the ratio between the frequency bands is taken in the *classifier* block later. This is done so that the powers for an epoch are calculated only once and the classifier can derive whatever feature is needed simply by using the absolute power values.

From the list of features in Chapter 5, in order to compute the relative power features, the absolute powers need to be calculated for the following frequency bands:

- *total*
- *sigma*
- *beta*
- *theta*
- *alpha*
- *alpha1*
- *gamma*
- *delta*
- *delta2*

Further, absolute powers are also needed for the calculation of spectral edge frequencies in the following frequency bands:

- *0.5-30 Hz*
- *8-16 Hz*
- *4-12 Hz*
- *0.5-8 Hz*

As a result, there are a total of 13 instantiations of the power calculation module, one for each frequency band. For each of these instances, relevant input from the FFT magnitudes register bank must be provided and the generic constant needs to be initialised to the total number of input taps for each instance.

The spectral edge frequency features are of three types: *SEF50*, *SEF95* and *SEFd*. The latter is a derivative feature that is calculated by taking the difference between *SEF95* and *SEF50*. To avoid having to store this difference, *SEFd* is calculated in the *classifier* block when required. The *SEF50* and *SEF95* features are calculated by instantiating the two modules with the correct input frequency bands and setting the generics for input length and initial frequency. The following *SEF* features need to be calculated for the sleep algorithm:

- *SEF50(0.5-30)*
- *SEF95(0.5-30)*
- *SEF50(8-16)*
- *SEF95(8-16)*
- *SEF50(4-12)*
- *SEF95(4-12)*
- *SEF50(0.5-8)*
- *SEF95(0.5-8)*

Hence, four instances of *SEF50* and four instances of *SEF95* modules are instantiated to cover all the cases. Each instance gets its inputs from the power calculation module and the FFT magnitudes module in the required frequency range.

The complete *feature calculation* block then consists of multiple instances of *SEF50*, *SEF95* and power calculation modules in different configurations and is shown in Figure 6.17. This block is connected to the output of the FFT block and gets triggered on receiving a valid signal from it to indicate that new data is ready to be read. The features are calculated for every subepoch and accumulated for each epoch. A block-level valid pulse is generated when all the features have been calculated and the subepoch address is 15 (i.e. last subepoch of the current epoch). This pulse indicates to the *classifier* that all the features for the current epoch have been calculated and are ready to be used for classification.

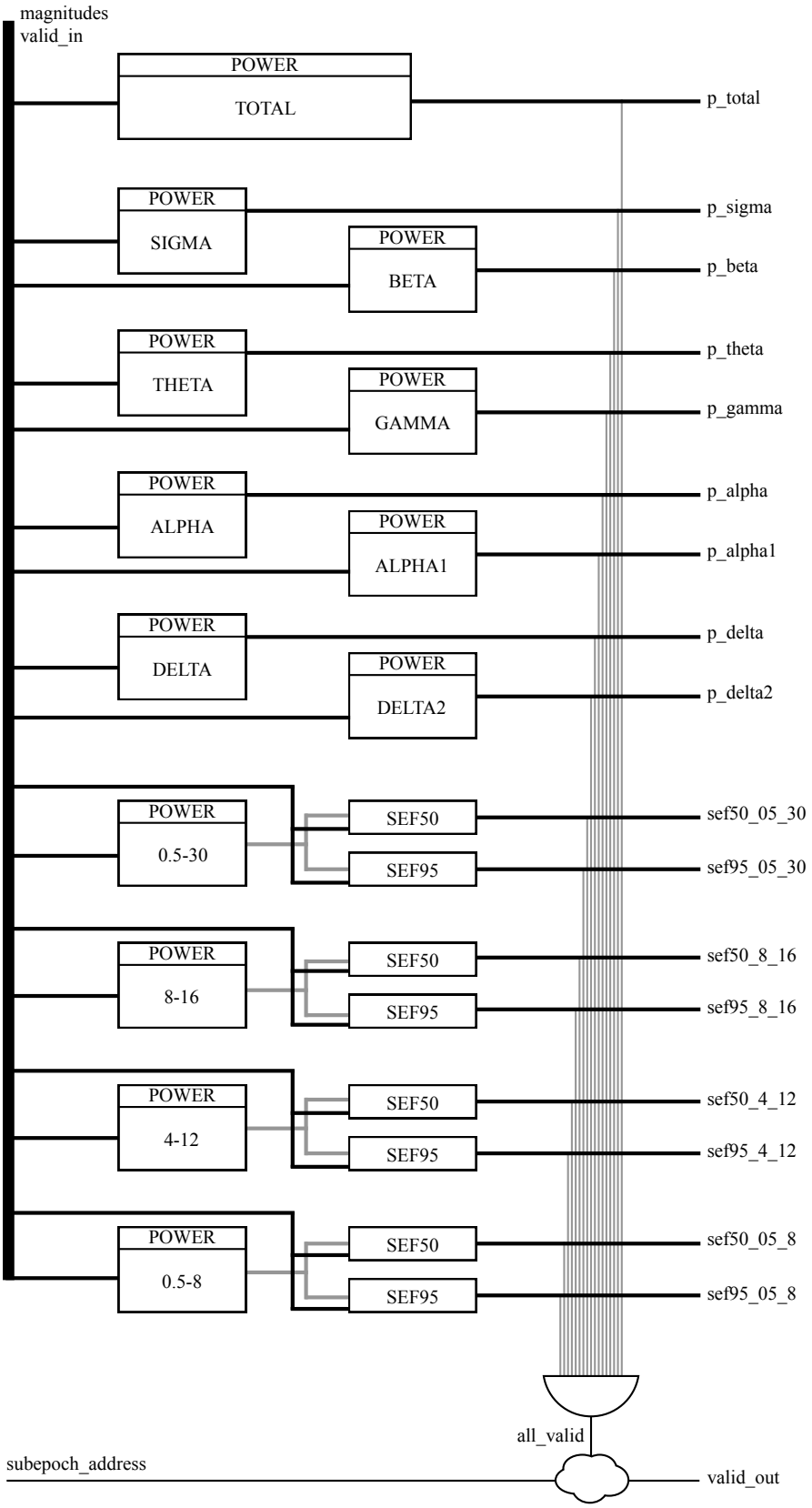


Figure 6.17: *Implementation of the complete Feature Calculation block.*

6.5 Classifier

The *classifier* block is the final stage within the algorithm pipeline. Its task is to assign a valid sleep stage to each epoch based on a set of thresholds. It sits in an idle state while all the data accumulation and feature extraction goes on for each subepoch. The classification of an epoch begins when the features for all its subepochs are calculated and a valid pulse generated by the *feature calculation* block.

The classifier is an implementation of the state machine-controlled decision trees introduces in Chapter 5. There are several different states in the finite state machine designed to implement the classifier. These states are listed below:

- `WAITING_FOR_VALID`
- `CORE_W`
- `CORE_N1`
- `CORE_N2`
- `CORE_N3`
- `CORE_R`
- `PERI_W_N1`
- `PERI_W_N2`
- `PERI_W_N3`
- `PERI_W_R`
- `PERI_N1_N2`
- `PERI_N1_N3`
- `PERI_N1_R`
- `PERI_N2_N3`
- `PERI_N2_R`
- `PERI_N3_R`

Initially the classifier is idle which corresponds to the `WAITING_FOR_VALID` state. Its next state depends on the score of the last classified epoch and can be one of `CORE_W`, `CORE_N1`, `CORE_N2`, `CORE_N3` or `CORE_R`. Each of these states correspond to a core test that determines if the epoch belongs to that particular sleep stage or one of the others. On

entering one of these core states, an enable signal for the corresponding core test is set to high which switches it on from an otherwise idle state.

The FSM remains in the same state and waits until it receives a valid signal from the core test which signifies that its output is ready to be read. If the core test classifies the epoch having the same sleep stage as the previous epoch then the FSM goes back to the initial `WAITING_FOR_VALID` state since there are no further tests required. If the sleep stage is *Others*, then the next state of the FSM is one of the peripheral tests.

The peripheral tests are implemented exactly the same way as the core tests and are enabled only when they are needed. The corresponding enable signals for the peripheral tests must be set to high until it produces a valid output. The result of each peripheral test determines the next state of the FSM until the epoch is assigned a sleep stage.

6.5.1 Design of each test

All core and peripheral tests are essentially the same in design since they have an identical structure. They all take a fixed number of features as inputs, perform one or more comparisons and produce one of the sleep stages as output. However, there are few differences between them such as the number of comparisons needed, the input features and the thresholds corresponding to each feature. This dictates how many clock cycles are required by each test to produce an output.

Each node in the decision tree performs a comparison of only one feature against its specific threshold. The feature at the node can have one of the three types:

- Relative power in a frequency band. This is the power in a certain frequency band divided by the power in the entire frequency of interest.
- Ratio of power in different frequency bands. This is the power in a certain frequency band divided by the power in a different frequency band.
- Mean spectral edge frequency. This is obtained by dividing *SEFxx* by 15 (which is the number of subepochs).

It should be noted that all the three feature types are obtained in a similar fashion by dividing two numbers and then comparing this result against a threshold. The mathematical operation at each node can be described as:

$$\frac{A}{B} < THRESHOLD \tag{6.13}$$

This can be rearranged as below, where $THRESHOLD\_INV$ is $1/THRESHOLD$.

$$A \times THRESHOLD\_INV < B \tag{6.14}$$

This rearrangement gets rid of the division operation and replaces it with multiplication. The inverted threshold can be stored on chip instead of its original value to implement this equation.

At any given point in time only one node of the tree is being used. This means that rather than having many multipliers and comparators for any core or peripheral test, only one of each component can be shared across the tree. Figure 6.18 shows a block diagram of the mathematical operation at each node. By changing the A, B and T (threshold) inputs using a multiplexer, based on the clock cycle, a single multiplier and comparator can be used to perform all the operations of a given decision tree.

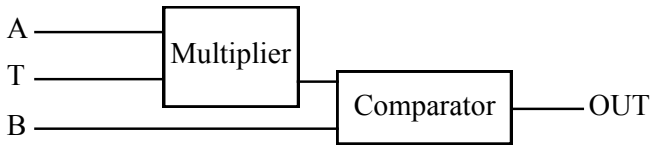


Figure 6.18: *Mathematical operation at one node of the decision tree.*

As an example, consider the decision tree shown in Figure 6.19 with fictitious thresholds. In this tree, there are three nodes using the three kinds of features for comparison. The A, B and T inputs in each cycle of execution is shown in Table 6.5. In the first cycle, the relative power in gamma band is checked against the threshold. The result of this comparison then determines the next step. The second cycle can use either of the two features depending on the output from the first cycle. When *SEF50* feature is used, it is multiplied by 1/15 (a constant) to obtain the mean value.

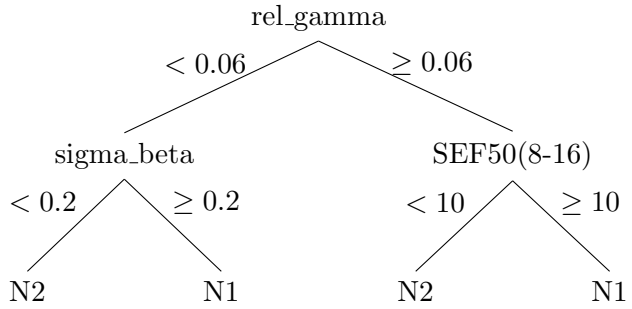


Figure 6.19: *An example decision tree to illustrate its multi-cycle operation.*

6.5.2 Further optimisations

The previous section establishes that each core and peripheral test can be implemented using one multiplier and one comparator. However, it is also observed that at any given time only one of the tests is enabled. The next test is only enabled when the current core or peripheral test finishes execution and produces a valid output. This means that

Table 6.5: *Multiplier and comparator inputs in each clock cycle.*

Cycle	A	T	B
1	gamma	1/0.06	total
2	sigma	1/0.2	beta
2	*SEF50(8-16)*	1/15	10

at any instance in time, only one multiplier and comparator is required across *all* core and peripheral tests.

All core and peripheral tests only need to provide the correct values of A, B and T in each clock cycle when they are enabled. These can then be fed to a single multiplier and comparator which is shared across all tests using a multiplexer that selects the appropriate input based on which test is currently enabled. This optimisation reduces a lot of datapath area by using only one multiplier at the cost of additional multiplexers. This takes less area in comparison to the design that uses several multipliers (one for each test).

6.5.3 Block level implementation

Figure 6.20 shows the block diagram of the *classifier* implementation at the top level. It connects directly to the output of *feature calculation* block and takes all the features and a valid signal as input. The valid signal is used to enable the *classifier* whenever the features have been calculated and are ready to be read.

Within the *classifier* block there are instances of the various core and peripheral tests. Each test is provided with relevant subset of features that are needed for decision making and an enable signal that activates the test. The output of these tests are connected to a multiplexer that controls the inputs to a single multiplier and a comparator. Figure 6.20 also shows a state machine controller that determines which test needs to be activated at any time, establishes the state of the machine and also provides the control signal for the multiplexer to allow correct inputs for the shared datapath.

When the classification of an epoch is complete, the result is available on the output port. At the same time, a valid signal is raised to signify that the data on the port is ready to read. This output is encoded as shown in Table 6.6 and remains valid until the next epoch is available at the input of the classifier block.

6.6 Top level system

The top level of the complete sleep staging system is shown in Figure 6.21. The system can receive input data samples either directly from the ports (`din_port`) or from the

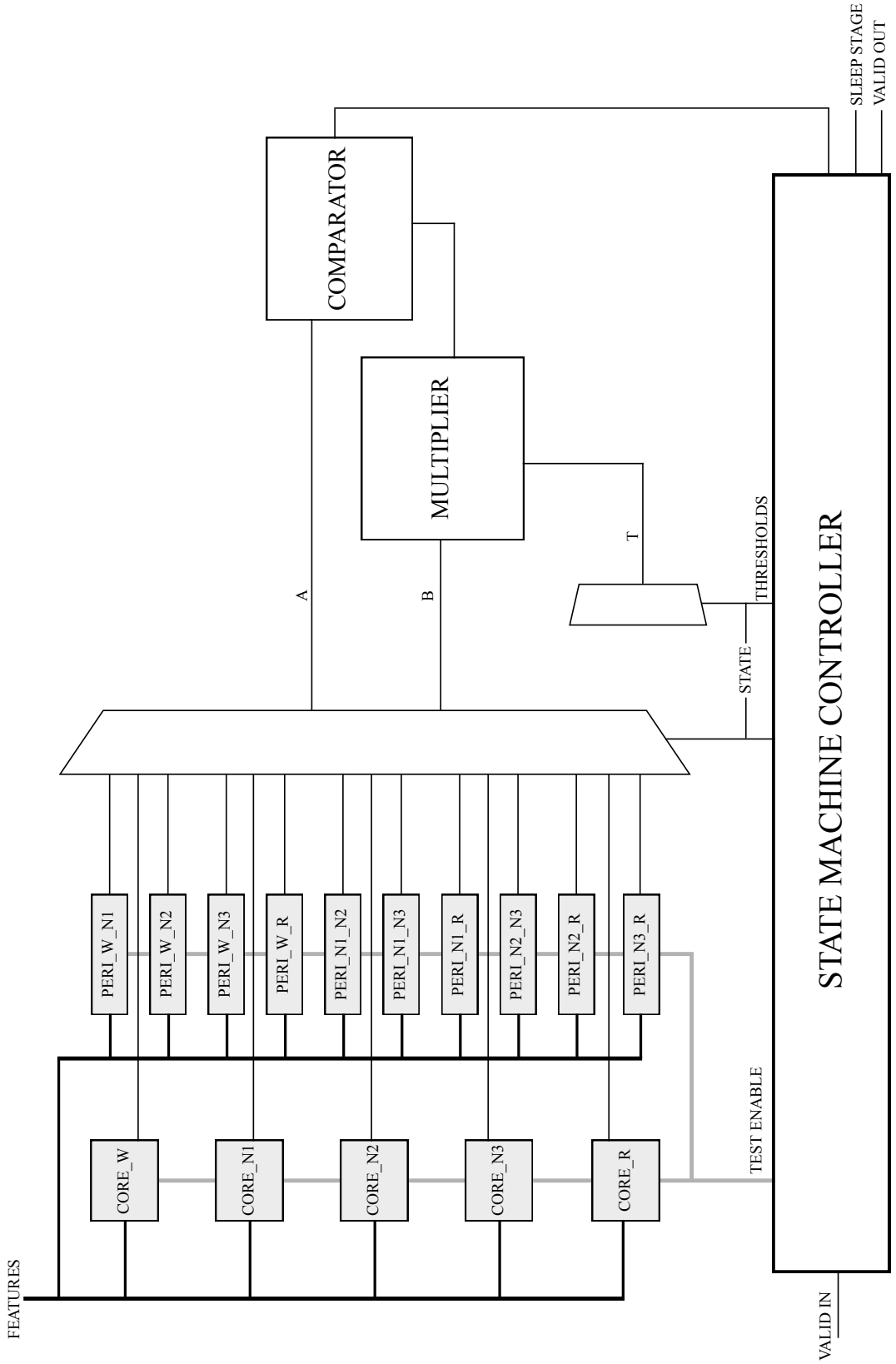


Figure 6.20: *Top level block diagram of the Classifier.*

Table 6.6: *Encoding of the output from the classifier block.*

Sleep Stage	Output Encoding
Wake	000
N1	001
N2	010
N3	011
REM	100
Others/Unknown	111

output of an on-chip ADC (`din_adc`). In either case the corresponding valid signals, `vin_port` or `vin_adc` must be high whenever data from the ports have to be read. The `sel` input is used to select the data source within the *input controller*. The outputs of this block are a set of 512 data samples `data_ic` and a `valid_ic` status signal that indicates this data is ready to be read by the *FFT* block. The *FFT* block finishes its computation and provides a set of magnitudes on the `data_fft` internal port as well as a `valid_fft` status signal. This signal is used by the *feature calculation* block to start reading magnitude values and calculate the desired features. Once all the features are calculated, a `valid_fc` signal tells the *classifier* to read them from `data_fc`. The final result of classification is available on the output port `dout` and is only valid as long as the `vout` output signal remains high.

All the modules and blocks within the complete system use a single clock source which is provided at the `clk` port. Finally, the entire system can be reinitialised by providing a low signal at the `rstn` input port. This resets all the internal registers and the system starts in normal operation again once the low signal at the `rstn` port is removed.

6.7 RTL simulation

The VHDL description of the complete sleep staging algorithm is simulated using Cadence Incisive Simulator (version 12.10) [3] for its functional verification. The testbench is designed to mimic real world conditions such that an input sample is made available to the system after every $1/256th$ of a second. EEG samples from all twenty test subjects in the DREAMS Subjects database [4] are used. For each test, EEG samples are stored in a text file as hexadecimal numbers. The testbench writes out an output file with the sleep scores. This will be compared against the reference hypnogram that was generated when the algorithm was run in MATLAB. The number of correctly classified epochs can then be used to determine the accuracy of hardware implementation.

Table 6.7 shows the number of misclassified epochs and the accuracy obtained for each test case. The overall classification accuracy of the hardware implementation compared

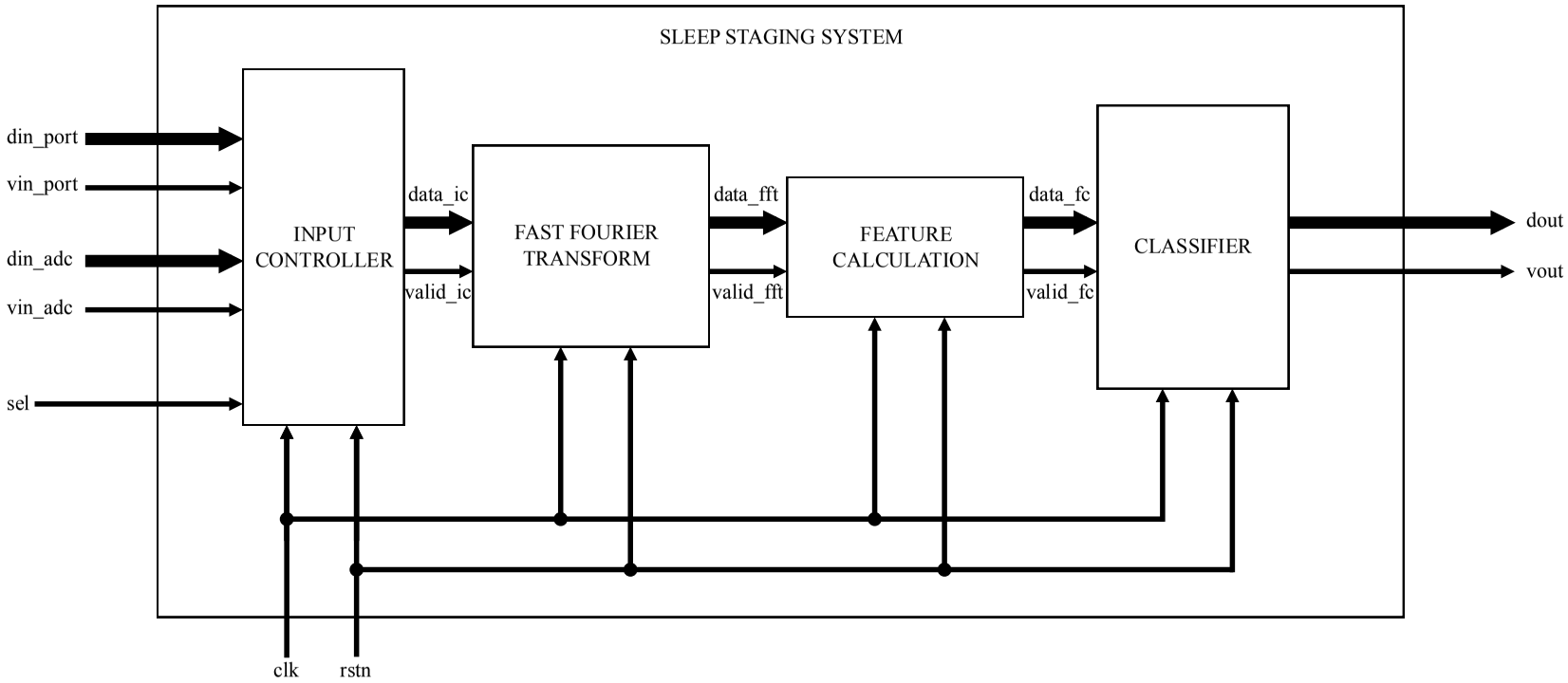


Figure 6.21: *Top level block diagram of the complete sleep staging system.*

to the reference is 98.72%. Ideally, this should be 100% but there are a number of factors that makes the hardware different from the algorithm that is implemented in a numerical software (such as MATLAB). The major difference between the two is the number representation. In MATLAB each number is represented as a 64-bit floating-point number while the hardware has been designed to represent a number as 24-bit fixed-point number. This adds certain truncation and round-off errors that can lead to misclassified epochs when any feature value is very close to a predefined threshold. It must be noted that in most cases a misclassified epoch does not cost the algorithm significantly in terms of the overall accuracy. Once an epoch is incorrectly classified the state machine controlling the decision trees goes into the wrong state. However, it is expected that the next set of core and peripheral tests will bring it back to the correct state.

From Table 6.7, it can also be seen that only subjects 10 and 15 have a relatively higher number of misclassified epochs while in all other cases this number is very low. The reason for more classifications in these two subjects is that the feature values remain close to the threshold consistently during a certain section of the recording. This is a limitation of the algorithm since the thresholds are fixed and hence can not adapt to the changing nature of the signals.

Table 6.7: *RTL simulation results of the sleep staging algorithm hardware.*

Subject	Total Epochs	Misclassified Epochs	Accuracy (%))
01	961	3	99.69
02	985	0	100
03	1008	7	99.31
04	1053	5	99.53
05	1043	10	99.04
06	997	1	99.90
07	1012	17	98.32
08	970	13	98.66
09	1117	10	99.10
10	1032	85	91.76
11	1008	9	99.11
12	961	8	99.17
13	1111	15	98.65
14	1004	2	99.80
15	840	43	94.88
16	967	4	99.59
17	997	8	99.20
18	1021	2	99.80
19	1033	10	99.03
20	1145	8	99.30
Total	20265	260	98.72

6.8 Synthesis

Synthesis is the step of mapping RTL design to technology-specific logic gates available in the library provided by the foundry. The synthesis tool first reads in the design, elaborates it (i.e. create a database) and then begins the process of mapping to the gates. This is an iterative process that is driven by a set of constraints which may be timing or area constraints. In this work, Cadence RTL Compiler (version 11.21) [5] is used to perform design synthesis. The clock frequency is set to 1 MHz i.e. a period of 1 μs. This frequency is well over the maximum at which the design is intended to run.

The synthesis process resulted in a design that met the timing requirements easily with a slack of 935592ps. The area required by sequential and combinational logic as well as datapath modules as reported by RTL Compiler is shown in Figure 6.22 and Figure 6.23.

Type	Instances	Area	Area %
sequential	41247	2900360.333	53.2
inverter	5044	42702.912	0.8
buffer	7	279.418	0.0
logic	127129	2505416.256	46.0
total	173427	5448758.918	100.0

Figure 6.22: *RTL Compiler report summary of the gates used in the netlist.*

Type	CellArea	Percentage
datapath modules	374261.53	6.87
external muxes	0.00	0.00
others	5074497.39	93.13
total	5448758.92	100.00

Figure 6.23: *RTL Compiler report summary of the datapath modules used in the netlist.*

6.9 Formal verification

A number of optimisations are applied to the design during synthesis stage to get the desired result for meeting area and timing constraints. During this process any unused logic is removed and anything that the tool considers redundant may be merged, optimised or removed. The final netlist may look nothing like the original RTL description therefore it is important to verify it against the reference RTL description to ensure it is logically and functionally identical.

6.9.1 Logic equivalence check

The synthesised netlist is checked for logical equivalence against the reference RTL using Cadence Encounter Conformal Equivalence Checker (version 11.10) [6]. The comparison is performed in a hierarchical fashion such that the inner modules are verified first before moving on to the top level. The final result of this comparison from the runfile is shown in Figure 6.24.

```
Processed 23 out of 23 module pairs     EQ: 23  NEQ: 0  ABORT: 0

================================================================================
Module Comparison Results
--------------------------------------------------------------------------------
Equivalent                      23
--------------------------------------------------------------------------------
Total                           23
--------------------------------------------------------------------------------
Hierarchical compare : Equivalent
================================================================================
// Command: usage
CPU time     : 13548.33 seconds
Memory usage : 1392.55 M bytes
```

Figure 6.24: *Conformal LEC report summary for formal verification of the synthesised netlist.*

6.9.2 Gate-level simulation

The synthesised netlist is also checked for functional equivalence to ensure that it produces the same output for an input as it did when performing RTL simulation. For this purpose, the same testbench and input set is used as in Section 6.7. The hypnogram written out for each subject is then compared against the earlier one during RTL simulation. The hypnograms were found to be completely identical for all test cases indicating functional equivalence of the netlist.

6.10 Place and route

The final layout is generated at the *place and route* stage using Cadence Encounter Digital Implementation System (version 11.12) [7]. The tool automates the process of generating the layout however a comprehensive set of timing and floorplanning constraints and instructions need to be provided to it in order to get the desired result. For this reason, several iterations may be required to get the constraints right. The steps required to perform the placement and routing of the design are summarised as follows:

- Floorplanning to set the aspect ratio of the design
- Place I/O pins appropriately
- Power planning to create a power ring around the core and other power stripes
- Routing the power connections
- Placing all the required standard cells from the library
- Clock tree synthesis
- Filling the empty spaces in the core
- Routing the design
- Verifying the design to ensure all the design rules are met

The final layout generated for the sleep staging algorithm is shown in Figure 6.25 and measures 8.35 mm^2 in size. After the design is fully placed and routed, the final netlist with additional buffers, inverters, filler cells, etc., is generated. This netlist is also verified for logical and functional equivalence using the same procedure as in Section 6.9. Once the equivalence is established and all the design rules are met, the design is ready to be taped out for fabrication.

6.11 Discussion

This chapter presented the complete IC design of the sleep staging algorithm that was proposed in Chapter 5 using the requirements and features discussed in earlier chapters. This design used several techniques to reduce the gate count, area and power consumption. For example, multi-cycle operations have been used extensively to reduce the number of datapath modules required. However, with this being the first iteration of the design there is still room for further improvement.

The register banks used in the FFT block could be replaced by RAM macros. These would not only reduce the power consumption but also reduce the area needed to store the array of real and complex numbers. The circuit is intended to run at a clock frequency of around 1 kHz but was synthesised with a target frequency of 1 MHz. Despite the higher frequency the large positive slack showed that meeting the timing is not a problem. If a different process technology with smaller node is used, the area of the chip can be almost linearly reduced while the large slack will still allow the timing to be met in the new technology.

Apart from further improvements, there are several things that need to be done in the near future once the fabricated chip is received from the foundry. The most obvious next step is to test the actual chip and verify its functionality. The debug ports added

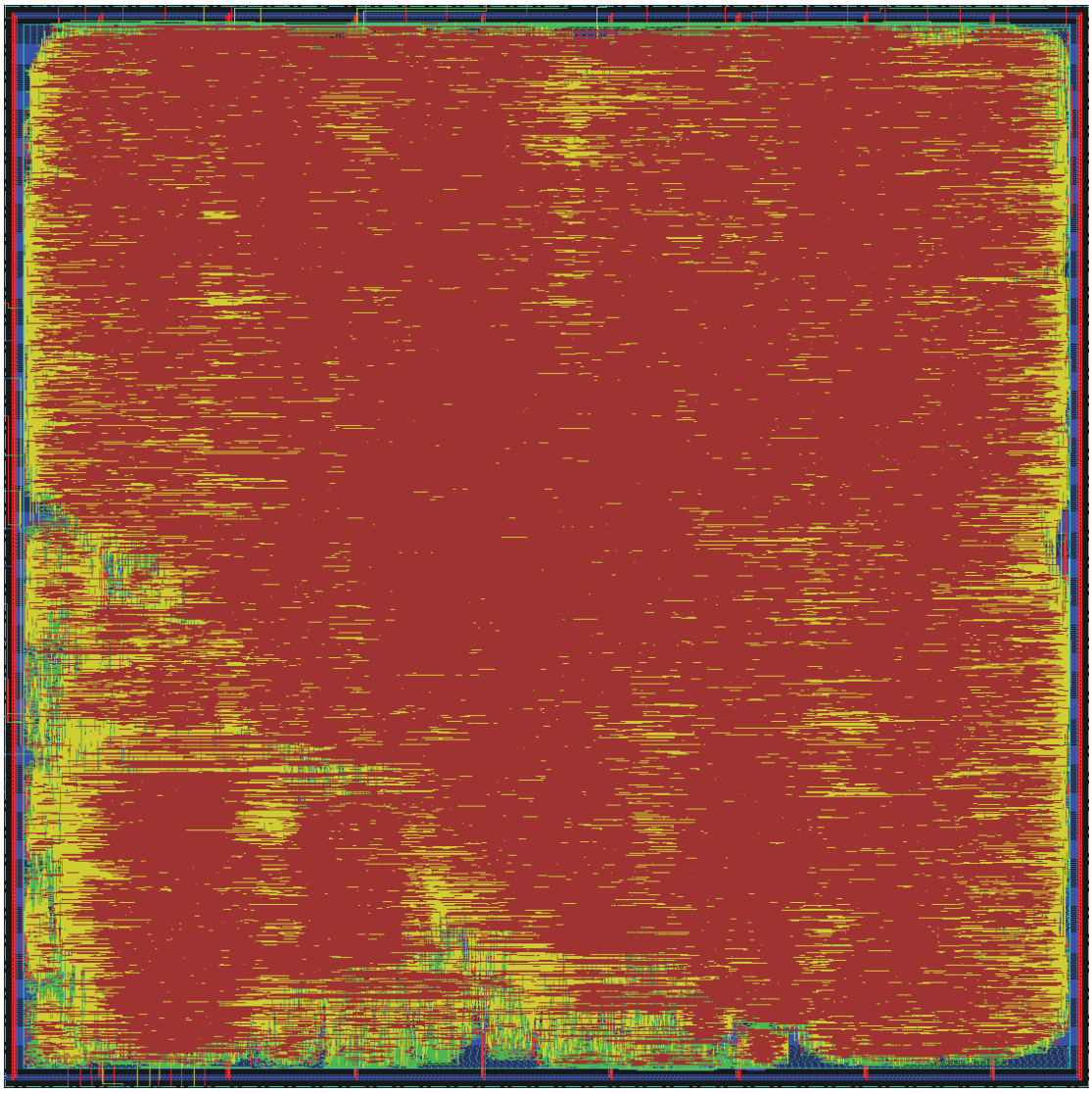


Figure 6.25: *Final layout of the automatic sleep staging algorithm integrated circuit measuring* $2890\mu m \times 2890\mu m$

to the chip will help determine if all blocks are correctly working or if there is something wrong. Finally, the power consumption of the entire circuit in real-world conditions need to be assessed. It is expected that the power consumption will be quite low which allows further complexity to be added to the algorithm in future while still remaining within the power budget.

Although an integrated circuit implementation of the sleep staging algorithm greatly reduces the power consumption requirement, it also makes the system less flexible compared to a micro-controller implementation. For example, if implemented on a micro-controller the algorithm can be easily updated allowing the thresholds and the order of decision trees to be modified if needed. It also allows an improved version of the algorithm to be easily tested in the field without incurring any additional hardware cost. Another advantage of using a micro-controller is the availability of common communication peripherals which enables the system to be integrated with other standard devices.

However, the main advantage of using a custom integrated circuit as compared to a micro-controller implementation is the difference in power requirements. Using a custom integrated circuit reduces the power by an order of magnitude freeing up space to perform more complex arithmetic operations. This saving in power also means that a small battery can be used to power up a system for a long duration of time. It also allows for the possibility of having the complete system on a single chip consisting of both the algorithm processor as well as the analogue front end. This reduces the area requirements on a circuit board making the final package small thereby suitable for wearable use. On the other hand, with a micro-controller based solution, further circuitry and additional components will be needed to create a low noise amplifier (as well as an ADC in some cases). Finally, an ASIC implementation also demonstrates the potential of this particular sleep staging algorithm to be part of a broad-based multi-purpose EEG system on chip that is not limited to sleep monitoring but also useful for other neurological conditions such as epilepsy.

References

[1] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex Fourier series," *Math. Comp.*, vol. 19, no. 1, pp. 297–301, 1965.

[2] G. Slade, "The Fast Fourier Transform in hardware: A tutorial based on an FPGA implementation," *Article*, 2013.

[3] Cadence. (2015) Incisive Enterprise Simulator. [Online]. Available: http://www.cadence.com/products/fv/enterprise_simulator/pages/default.aspx

[4] University of MONS - TCTS Laboratory. (2015) The DREAMS Subjects Database. [Online]. Available: http://www.tcts.fpms.ac.be/~devuyst/Databases/DatabaseSubjects/.

[5] Cadence. (2015) Encounter RTL Compiler. [Online]. Available: http://www.cadence.com/products/ld/rtl_compiler/pages/default.aspx

[6] ——. (2015) Encounter Conformal Equivalence Checker. [Online]. Available: http://www.cadence.com/products/ld/equivalence_checker/pages/default.aspx

[7] ——. (2015) Encounter Digital Implementation System. [Online]. Available: http://www.cadence.com/products/di/edi_system/pages/default.aspx

7 Conclusions

7.1 Contributions

This thesis has presented an automatic sleep staging algorithm for use in real-time wearable polysomnography systems. These systems are wireless, small in size and comfortable to use with a small battery thus having a very limited power budget. Hence any signal processing performed in them to extract information from sleep signals must be very limited and have low computational complexity to fit within this budget while giving meaningful results at the same time.

Chapter 2 reviewed the existing practice of clinical polysomnography in detail and discussed its importance in the diagnosis of sleep disorders. This was followed up with a discussion about the limitations and shortcomings of PSG in its existing form. It was concluded that HPSG, while being an attractive alternative or complementary solution still suffers from the requirements of using multiple channels for recording EEG, EOG and EMG data. Based on the sleep staging rules and multiple sleep staging algorithms it was concluded that the realisation of a reduced channel system for sleep staging is limited by the requirement of at least one of EOG and EMG channels to help in the scoring of REM sleep which is similar to N1 and Wake stages. A comprehensive review of various automatic sleep staging algorithms proposed in the literature was then presented together with several commercially available sleep acquisition and scoring systems that have tackled certain aspects of the limitations of PSG.

A wearable single channel EEG system was proposed as a solution to make the PSG systems easy-to-use for the patients from the comfort of their homes. Such a system would have to be very small with low power budget and will thus be limited in the computational power it contains for processing any EEG data. Its size, weight, power and complexity are cited as the core constraints, which leads to the requirement of a low-complexity sleep staging algorithm capable of running on a resource-constrained system with only a single channel of EEG data. As a result, the three most important objectives of this thesis are finding features that can detect REM sleep from a single EEG channel (without requiring EOG/EMG), developing a low-complexity sleep staging algorithm and realising the algorithm as a low power system.

The literature review of sleep staging algorithms revealed several discrepancies in the way their detection performances were reported. This means that comparison of different algorithms is non-trivial since different metrics and databases were being used. As a

result, Chapter 3 proposed a set of recommendations that would lead to standardised assessment of these algorithms. The main proposals were to use the publicly available databases, selecting consistent sections of data from them and using same selectivity and accuracy metrics. A simple sleep staging algorithm based on spectral features and an SVM was also used to demonstrate how the lack of standardisation leads to misleading results being reported.

Chapter 4 explored the spectral power of EEG signals in 8-16 Hz frequency range. Within this range, spectral edge frequencies at 95% and 50% appeared to be distinctly different during REM sleep. The difference between these features, termed *SEFd*, was used as a novel feature with a simple thresholding classifier. This showed *SEFd* to have a high discriminatory ability for detecting REM stages. This ability was further studied in three different EEG channels and the frontal channels resulted in the best performance. The low computational complexity of the *SEFd* feature with its high REM detection ability makes it suitable for use in a single EEG channel sleep scoring system.

Using this novel feature for REM detection and the requirements set out in Chapter 2, a novel sleep staging algorithm was presented in Chapter 5. This algorithm has been designed using a set of small decision trees that are controlled using a state machine that activates them when required in a contextually driven order. The algorithm works fundamentally on the principle that a simple test should first be used to determine whether the sleep stage has changed or not. Only if it has changed, the other tests should be used to determine the next sleep stage. This way, although the algorithm used several spectral features only a handful are needed to classify an epoch, since not all trees are needed at all times. Further, an average of three to five comparisons are needed in most cases to classify an epoch. The computational complexity of this approach with respect to traditional decision trees and its suitability for implementation on a low power system is also discussed in this chapter. Finally, it was reported that the algorithm achieved an overall accuracy of 73% and 79% using two separate sleep databases.

Chapter 6 presented a full digital integrated circuit implementation of the sleep staging algorithm designed in Chapter 5. Several low power design techniques and optimisations were used to reduce the area and power consumption of the system. The algorithm was implemented using AMS 0.18μm technology and consisted of a multiplexed I/O interface allowing it to connect with an analogue front end for real-time data acquisition or a microcontroller for debugging purposes. The three major blocks designed specifically for this algorithm were *FFT*, *Feature Calculation* and *Classifier*. All of these were implemented as multi-cycle operations to share as many datapath resources as possible. The design was synthesised and laid out using industry-standard IC design tools and verified for both logical and functional equivalence with the reference design. The classification performance of the hardware implementation was also compared against that in Chapter 5 resulting in an accuracy of 98.7%.

7.2 Further work

The hardware implementation of the sleep staging algorithm has been taped out fabrication. The immediate next step, once the packaged IC arrives, is to test and verify the operation of all the circuits. This can be done by providing known input EEG values and observing the values at all intermediate (debug) and final output ports. After verifying the functionality of the complete algorithm, its power performance needs to be characterised. Although the power simulations do show low current consumption, it still needs to be performed in real-world conditions to get a more accurate figure.

The work presented in this thesis is the very first fully digital circuit-level implementation of a sleep staging algorithm. This means that the power consumption of the circuit cannot be compared against any reference. However, any hardware that can run off a small coin cell for at least eight hours (average night sleep duration) can be classed as a suitable wearable device.

Although the circuit-level implementation of a complete sleep staging algorithm is a big leap towards the realisation of a fully wearable sleep system, there is still significant work required before such a system can land in the hand of consumers or medical practitioners. First, the algorithm designed in this work is quite simplistic. The idea was to have a system capable of running in such a low-power environment. After characterising the circuits, if the power consumption is found to be very low, more complexity can be added to the algorithm to improve the classification accuracy. Second, there is room for improvement in the sleep staging algorithm itself. For example, design of peripheral tests with different weights depending on the sleep stage can lead to improvement in accuracy during stage transitions. Third, the chip area can be significantly reduced by using a different technology. Use of 65nm technology instead of 180nm will reduce the chip area by more than a half. Fourth, the FFT implementation can be improved by using dual port RAM, rather than simple register banks, further reducing area and power consumption. Fifth, more characteristic features can be explored to improve the algorithm classification accuracy in various sleep stages. Finally, in the long run, the circuit should be packaged appropriately and tested with real patients, with the help of doctors, to get proper feedback on how the system can be improved in terms of user experience and human factors.

Appendices

A Databases

The performances of different algorithms in this thesis were evaluated using freely available databases that are widely used among sleep researchers. This allows the results to be compared against the performance of other algorithms that have already been published in literature in the past and also in future. The following sections list out several databases that can be used for testing sleep algorithms.

A.1 PhysioNet Sleep EDF Database

For more than ten years the PhysioNet Sleep EDF database [1], [2] has been used to test and report the performances of various sleep staging algorithms. It consists of recordings from eight Caucasian males and females with an age range of 21–35 years. Four of these recordings, beginning with the letters *SC* were part of a study where signals were recorded for over 20 hours at subjects' homes. The other four recordings, beginning with letters *ST*, were obtained as part of a study looking at the effects of *temazepam* on sleep. In this study, the signals were recorded at a hospital while the subjects were sleeping at night typically between 8-10 hours. The EEG signals in both cases were sampled with a frequency of 100 Hz and scored using R&K rules for sleep classification. The list of subjects, their age, sex and the total recording duration for each is shown in Table A.1.

Table A.1: *Detail of the recordings in PhysioNet Sleep EDF Database.*

Name	Age	Sex	Recording Duration (HH:MM:SS)
SC4002E0	33	F	23:35:00
SC4012E0	33	F	23:45:00
SC4102E0	26	M	23:49:00
SC4112E0	26	M	23:10:00
ST7022J0	35	F	11:03:10
ST7052J0	32	F	10:45:10
ST7121J0	21	M	10:21:30
ST7132J0	22	M	09:38:40

A.2 PhysioNet Sleep EDF Expanded Database

This is a superset of the PhysioNet Sleep EDF Database and consists of recordings from the same two studies described in the previous section [3]. It consists of 61 PSG recordings with hypnograms scored using R&K rules. Of these, 22 are overnight *ST* recordings while the others are *SC* recordings of over 20 hours each. The complete list with details on each subject is shown in Table A.2.

Table A.2: *Detail of the recordings in PhysioNet Sleep EDF Expanded Database.*

Name	Age	Sex	Recording Duration (HH:MM:SS)
SC4001E0	33	F	22:05:00
SC4002E0	33	F	23:35:00
SC4011E0	33	F	23:21:00
SC4012E0	33	F	23:45:00
SC4021E0	26	F	23:22:00
SC4022E0	26	F	22:58:00
SC4031E0	26	F	23:30:00
SC4032E0	26	F	22:46:00
SC4041E0	34	F	21:25:00
SC4042E0	34	F	23:16:00
SC4051E0	28	F	22:41:00
SC4052E0	28	F	23:23:00
SC4061E0	31	F	23:05:00
SC4062E0	31	F	23:35:00
SC4071E0	30	F	23:25:00
SC4072E0	30	F	23:05:00
SC4081E0	25	F	23:18:00
SC4082E0	25	F	21:57:00
SC4091E0	25	F	22:46:00
SC4092E0	25	F	23:49:00
SC4101E0	26	M	22:40:00
SC4102E0	26	M	23:49:00
SC4111E0	26	M	22:01:00
SC4112E0	26	M	23:10:00
SC4121E0	26	M	23:13:00
SC4122E0	26	M	21:43:00
SC4131E0	27	M	23:27:00
SC4141E0	27	M	22:58:00
SC4142E0	27	M	23:07:00

Table A.2: *Detail of the recordings in PhysioNet Sleep EDF Expanded Database.*

Name	Age	Sex	Recording Duration (HH:MM:SS)
SC4151E0	31	M	21:50:00
SC4152E0	31	M	23:52:00
SC4161E0	32	M	21:53:00
SC4162E0	32	M	22:56:00
SC4171E0	31	M	22:51:00
SC4172E0	31	M	22:42:00
SC4181E0	28	M	22:58:00
SC4182E0	28	M	23:41:00
SC4191E0	28	M	23:07:00
SC4192E0	28	M	21:45:00
ST7011J0	60	M	09:58:20
ST7022J0	35	F	08:32:40
ST7041J0	18	F	08:39:20
ST7052J0	32	F	09:07:10
ST7061J0	35	F	09:02:50
ST7071J0	51	F	07:37:40
ST7082J0	66	F	07:55:30
ST7092J0	47	M	08:01:10
ST7101J0	20	F	09:09:20
ST7112J0	21	F	08:31:10
ST7121J0	21	M	08:40:00
ST7132J0	22	M	07:29:40
ST7141J0	20	M	07:40:10
ST7151J0	66	F	10:41:50
ST7162J0	79	F	08:20:40
ST7171J0	48	F	08:02:10
ST7182J0	53	F	09:25:40
ST7192J0	28	F	09:08:20
ST7201J0	24	M	08:05:20
ST7212J0	34	F	08:47:00
ST7221J0	56	M	09:09:20
ST7241J0	48	F	09:03:30

A.3 DREAMS Subjects Database

This database is part of the DREAMS project [4] which aims to reduce variability in the analysis and classification of sleep signals [5]. It consists of 20 overnight PSG recordings from healthy subjects (16 females and 4 males) between 20-65 years in age. Each recording includes three EEG (CZ-A1 or C3-A1, FP1-A1 and O1-A1), one EMG and two EOG channels sampled with a frequency of 200 Hz. Additionally, separate hypnogram scores using both R&K and AASM rules are available with each recording in this database. The details of each recording are shown in Table A.3. It should be noted that since this is a recent project there are no algorithms that have been characterised using this database. The sleep staging algorithm in this thesis uses this database to report results in Chapter 4 and Chapter 5.

Table A.3: *Detail of the recordings in DREAMS Subjects Database.*

Name	Age	Sex	Recording Duration (HH:MM:SS)
Subject1	23	F	08:00:40
Subject2	47	F	08:12:30
Subject3	24	F	08:24:20
Subject4	48	F	08:46:30
Subject5	46	F	08:51:30
Subject6	65	F	08:18:40
Subject7	45	F	08:26:00
Subject8	22	F	08:05:00
Subject9	21	F	09:18:40
Subject10	20	F	08:36:20
Subject11	30	F	08:24:10
Subject12	54	F	08:00:40
Subject13	23	F	09:15:40
Subject14	57	F	08:22:10
Subject15	20	F	07:00:00
Subject16	27	F	08:03:50
Subject17	23	M	08:18:30
Subject18	27	M	08:30:40
Subject19	27	M	08:36:50
Subject20	20	M	09:32:30

A.4 DREAMS Patients Database

This database is also part of the DREAMS project [4] but includes PSG recordings from subjects with sleep disorders (SAHS, PLMS, insomnia, dysomnia, apnoea) [6]. There are a total of 27 overnight recordings from patients that are between 19-73 years old and their details are shown in Table A.4. These also include hypnograms using both R&K and AASM rules for each recording.

Table A.4: *Detail of the recordings in DREAMS Patients Database.*

Name	Age	Sex	Recording Duration (HH:MM:SS)
Patient1	72	M	08:01:40
Patient2	50	F	08:32:00
Patient3	52	F	09:07:50
Patient4	42	M	08:41:10
Patient5	40	M	08:22:40
Patient6	74	F	08:21:20
Patient7	52	M	08:13:30
Patient8	31	M	09:30:30
Patient9	53	F	08:26:50
Patient10	61	F	09:00:30
Patient11	73	M	08:59:50
Patient12	47	M	08:38:40
Patient13	38	M	08:08:10
Patient14	48	M	08:52:10
Patient15	53	M	07:55:00
Patient16	45	F	09:32:40
Patient17	57	M	09:20:20
Patient18	52	F	10:25:20
Patient19	53	F	08:58:30
Patient20	19	M	08:13:50
Patient21	44	M	09:20:30
Patient22	38	M	09:12:40
Patient23	60	M	08:07:30
Patient24	50	M	07:43:30
Patient25	27	F	08:59:40
Patient26	71	M	08:20:30
Patient27	69	F	08:54:10

A.5 Montreal Archive of Sleep Studies

This is a fairly new database with 200 PSG recordings from 97 males and 103 females with an age range between 18 and 76 years [7]. The recordings, sampled at 256 Hz, are obtained from three different sleep labs. These are split into five sets of which two sets are scored using AASM rules while the rest are scored using R&K rules.

References

[1] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals," *Circulation*, vol. 101, no. 23, pp. e215–e220, 2000.

[2] PhysioNet. (2013) Sleep-EDF Database. [Online]. Available: http://www.physionet.org/physiobank/database/sleep-edf/.

[3] ——. (2014) Sleep-EDF Database [Expanded]. [Online]. Available: http://www.physionet.org/physiobank/database/sleep-edfx/.

[4] University of MONS - TCTS Laboratory. (2015) The DREAMS Project. [Online]. Available: http://www.tcts.fpms.ac.be/projects/dreams/.

[5] ——. (2015) The DREAMS Subjects Database. [Online]. Available: http://www.tcts.fpms.ac.be/~devuyst/Databases/DatabaseSubjects/.

[6] ——. (2015) The DREAMS Patients Database. [Online]. Available: http://www.tcts.fpms.ac.be/~devuyst/Databases/DatabasePatients/.

[7] C. O'Reilly, N. Gosselin, J. Carrier, and T. Nielsen, "Montreal archive of sleep studies: an open-access resource for instrument benchmarking and exploratory research," *J. Sleep Res.*, vol. 23, no. 6, pp. 628–35, 2014.

B An open-source toolbox for standardised use of PhysioNet sleep EDF expanded database

The research presented within this chapter is an edited version of research previously published in:

S. A. Imtiaz and E. Rodriguez-Villegas, "An open–source toolbox for standardised use of PhysioNet Sleep EDF Expanded database," in proceedings of the 37th international conference of the IEEE Engineering in Medicine and Biology Society, Milan, August 2015,

B.1 Introduction

The performance of automatic sleep staging algorithms are usually evaluated using data that has been acquired as part of their research work or by using publicly available sleep databases. The advantage of using the latter approach is that it not only helps to show a method's own detection performance but also makes it possible to compare it with other existing methods that have been tested using the same database.

PhysioNet Sleep EDF database [1], consisting of eight recordings, has been the most popular publicly available sleep database used by multiple automatic sleep staging algorithms for development. One of the problems with the use of this database has been the lack of consistency in which data is extracted. This is because there are two kind of recordings in the database: one set containing only overnight recordings while the other consisting of recordings over a period of 24 hours. Different criteria have been used by research groups to establish the sleep start and end times for extracting the sleep part from the 24 hour recordings. This results in different sections of data being used. Together with varying performance metrics, this makes the comparison between various algorithms very difficult.

More recently, the Sleep EDF database has been made deprecated and a much larger superset of this database, known as the PhysioNet Sleep EDF Expanded database [2] (also referred to as Sleep EDFx database), has been made available which includes 61 well annotated sleep recordings. Some other freely available databases have also been made available in the recent past including [3] and [4] with 200 and 20 recordings respectively.

It is still early days in the use of the new EDF Expanded database. Results of analysis using this database have slowly started coming in with only four papers published so far [5]–[8] using this database. However, with the existing popularity of PhysioNet it is likely that EDF Expanded database will remain very popular in the coming years.

In Chapter 3, a set of recommendations were proposed to minimise the aforementioned problems and the use of public databases amongst sleep researchers was encouraged. Of course, it is not possible to go back and try to standardise the practices in already published work. However, the availability of a new database, which is potentially going to be widely used, presents a unique opportunity to establish a set of rules for its usage, particularly in the context of automatic sleep scoring. This chapter builds upon the previously proposed recommendations, by providing their implementation and making it freely available as an open source toolbox for MATLAB [9]. This will make the sleep data more accessible to researchers starting in this area, result in consistent data being used and the final algorithm performances also being uniformly assessed across different research groups. This toolbox is available at `https://github.com/anas-imtiaz/sleep-edfx-toolbox/`.

B.2 Getting the data

B.2.1 Downloading PhysioNet data

The first, and often the most tedious, step in using the Sleep EDF Expanded database is to download each of the 61 recordings separately from the PhysioNet website. These data files are in EDF format [10] and require conversion to a suitable format before being used for any further signal processing. To save time and simplify this initial step, all of the EDF data files can be downloaded from the PhysioNet website by executing a single function, shown below, and optionally providing a download location. These files are downloaded in newly created directories having the same name as the recording.

```
[saved_file,status]=downloadEDFxData(download_dir)
```

B.2.2 Conversion to MATLAB format

Once the EDF data files are successfully downloaded they need to be converted to a format that is readable in MATLAB environment. This is achieved using the function below that takes the recording directory (which contains the source EDF file) as its argument.

```
convertEDFxToMat(saved_file, status)
```

The conversion of EDF files requires the use of EEGLAB, an open source toolbox for EEG analysis [11]. An error will be shown if EEGLAB is not found on the user's system. The resulting *.mat* data files are stored in a folder named *matlab* under the test directory.

```
  Time        Date           Sample #  Type  Sub Chan  Num       Aux
[16:13:00.000 24/04/1989]        0     "    0    0    0   ## time resolution: 100
[16:13:00.000 24/04/1989]        0     "    0    0    0   Sleep_stage_W duration: 30630
[00:43:30.000 25/04/1989]  3063000     "    0    0    0   Sleep_stage_1 duration: 120
[00:45:30.000 25/04/1989]  3075000     "    0    0    0   Sleep_stage_2 duration: 390
```

Figure B.1: *An example section of hypnogram annotations from the recording SC4001E0*

Additionally, the list of channels for which data are available is also saved in the same folder.

B.2.3 Downloading and processing annotations

The next step is downloading the hypnogram annotations for each recording. Multiple methods are available on PhysioNet to get the hypnogram annotations. In this case, the files as obtained from PhysioBank ATM are used due to easier processing. These files have some important information including the start time of the recording, the sample number in data to be considered as the first valid sample (with reference to the start time) and the duration (in seconds) for which each sleep stage has been classified until the end of recording. An example section of one such file is shown in Figure B.1

In this annotation, the start time is stamped as `16:13:00.000` with the sample number being `0` at that time. It also shows that data has been classified as `Sleep_stage_W` for `30630` seconds (until time `00:43:30.000`) and then as `Sleep_stage_1` for `120` seconds followed by `Sleep_stage_2` for `390` seconds. Considering an epoch size of 30 seconds, it can be seen that from the starting point 1021 epochs are classified as Wake followed by 4 epochs as Stage 1 and 13 epochs as Stage 2. This is used later to create a vector in MATLAB for the hypnogram where each value in the vector represents the sleep stage assigned to a 30-second epoch. The hypnogram annotations are downloaded using the function below.

```
downloadEDFxAnnotations()
processEDFxHypnogram(hypnogram)
```

This function downloads not only the annotation files but also extra information including the text files with *lights off* time so that they do not have to be read off the spreadsheet. Files with *end time* are also provided to follow a consistent convention for data end times where these are not available (explained in Section B.3). Once downloaded, the annotations are converted to a MATLAB-compatible file with the same code for each sleep stage as the original on PhysioNet.

The process of initial downloading of source files and conversion to compatible data files needs to be performed only once and can together be automated with the `initialSetupEDFx`

function which calls all the functions described above sequentially.

```
initialSetupEDFx(download_dir)
```

B.3 Using the data

The downloaded data files can be easily opened as a normal file in MATLAB. However, recordings have different start and end times that do not necessarily correspond to the hypnogram scoring times. Further, the *lights off* time may also have a positive or negative offset from the recording start time. This section explains how data and hypnogram files are loaded using the multiple time annotation files and by following the recommendations in Chapter 3.

Start and end times

It was recommended in Chapter 3 to use the annotated *lights on* and *lights off* times as start and end times respectively. Where these are not available then the time from 15 minutes before the first scored sleep epoch and 15 minutes after the last scored sleep epoch should be used as start and end times.

For all the subjects in the EDF Expanded database, the *lights off* times are recorded in two different spreadsheets available on the PhysioNet website. From the hypnogram annotations, the *recording start time* is also known, which may be different from the *lights off* time. The annotations file also has the sample number from which hypnograms have been scored so the *hypnogram start time* can be deduced from this. For example in case of recording *ST7212J0*, the data recording starts at `24:43:00`, hypnogram marking starts at `23:43:30` while *lights off time* is `23:44:00`. As a result there are three different times: *recording start time*, *hypnogram start time* and *lights off time*. But the question is which of these should serve as the time from which data is to be read. The *hypnogram start time* will be either the same as *recording start time* or it will be after this time (since there can be no hypnogram without data). So if the *lights off* time comes after the *hypnogram start time*, then it is used as the start time otherwise the *hypnogram starting time* is used as the start time from which data is to be read.

The end time for reading each data file would ideally be the *lights on* time but these are not provided in the EDF Expanded database and hence need to be deduced. The method will be different for *SC* and *ST* subjects since the former consists of full day recordings of nearly 24 hours while the latter includes overnight sleep recording. For *SC* subjects, end time is taken as the time 15 minutes after the last scored sleep epoch (these have already been saved in a text file are downloaded with the hypnogram). For *ST* subjects, end time is considered the same as the *recording stop time* in most cases unless the hypnogram scoring ends before the data recording end time. For the same

example of *ST7212J0*, data recording stops at `08:30:00` but the hypnogram scoring ends at `08:08:30`. For such cases, the hypnogram end time is taken as the time up to which data is to be extracted. The hypnogram end times for *ST* subjects, extracted from the main annotation files, and the time 15-minute after the last scored sleep epoch for *SC* subjects are stored in the *lights on time* text file. The time offsets are then calculated such that data and hypnogram files are both read starting from this new start time and finish at the new end time. The number of epochs between the new start and end times is also calculated.

All these steps of determining the time offsets and fully mapped data and hypnogram files are performed within the `loadEDFx` function using which any particular recording from the database can be easily loaded by providing in the path of the data directory and the classification scheme (AASM or RK) to use for hypnogram.

An example of loading data for case *SC4001E0* is shown in the code listing below.

```
loadEDFx('SC4001E0', classification_mode)
```

The result is a container with key-value pairs for each channel of data loaded, an array with a list of channels available to use, hypnogram, total number of 30-second epochs, sampling frequency of data and the start and end times between which data is read.

The hypnogram supplied with the PhysioNet EDF Expanded database uses R&K rules of sleep stage classification. In Chapter 3, a method was proposed for approximate conversion from R&K to AASM scoring (shown in Table B.1). The `classification_mode` switch can be set to `AASM` if this conversion is required.

Table B.1: *Conversion from R&K to AASM classification*

R&K	S1	S2	S3 S4	REM	Wake MT
AASM	N1	N2	N3	REM	Wake

Viewing the data with hypnogram

After the files for a particular recording have been loaded, any signal from it can be viewed as a plot together with its hypnogram as a function of clock time. This is achieved using the `viewEDFxSignals()` function by providing the signal to be plotted and the hypnogram. A subsection of a signal may also be plotted by providing the start and end times of the section provided they are between the *lights off* and *lights on* times of the recording.

```
viewEDFxSignals(signal, ref_times, start_time, end_time, f_samp, hypnogram)
```

B.3.1 Performance evaluation

The performance of any new automatic sleep staging method can be evaluated by comparing the hypnogram generated by the new method against the reference hypnogram. Often, the performance metrics computed vary between different research groups making direct comparisons difficult. To have a uniform method to evaluate performances of different algorithms, it was proposed in Chapter 3 to use the overall accuracy, sensitivity and selectivity of each sleep stage along with the confusion matrix of the epochs correctly and incorrectly classified. These metrics are shown in the equations below.

$$Accuracy = \frac{no.\ of\ true\ detections}{total\ no.\ of\ epochs} \tag{B.1}$$

$$Sensitivity = \frac{no.\ of\ true\ detections\ in\ stage\ X}{no.\ of\ reference\ epochs\ in\ stage\ X} \tag{B.2}$$

$$Selectivity = \frac{no.\ of\ true\ detections\ in\ stage\ X}{no.\ of\ all\ detections\ in\ stage\ X} \tag{B.3}$$

The toolbox includes a function to facilitate the computation of these metrics. It requires the hypnogram to be saved in the same format as the reference (i.e. as a vector of characters for each sleep stage) and is used as follows.

```
computeEDFxPerformance(test_hypnogram, ref_hypnogram, classification_mode)
```

The function takes in the new hypnogram, reference hypnogram and the classification scheme as its three arguments and prints out the overall accuracy and confusion matrix of the new algorithm as well as the sensitivity and selectivity obtained in each sleep stage separately. The use of this function can be helpful to compare different algorithms by employing a consistent method for performance analysis.

B.4 Discussion

The Sleep EDF Expanded Database is still relatively new which provides an opportunity for sleep researchers to set some consistent rules for getting, using and analysing the results from this database. The toolbox described in this chapter aims to make it easy for researchers to adhere to such rules. The functions listed here are by no means exhaustive and there are many aspects in sleep research that require standardisation [12]. However, the toolbox does provide a good starting point for, at least, using the signals from a popular database in a uniform manner. By making the code open source, and available on GitHub [13], it is hoped that other researchers could contribute to it by adding more functionality to the toolbox.

The use of this toolbox will ensure that the same segment of data is being analysed by different researchers. It also helps in standardising the way this data is sliced and

the performance metrics are reported. This makes the comparison of various algorithms fair and prevents the common problems associated with the usage of previous PhysioNet Sleep EDF database. Further, the toolbox also saves time by providing quick and easy access to the PhysioNet Sleep EDFx database. This, in turn, means more time spent on data analysis rather than getting and preparing the data for use. It is important to adopt standard guidelines while the EDFx database is still in its infancy to prevent a repeat of earlier problems where different sections of data get used by different researchers together with varying performance metrics making their comparison almost impossible. It is hoped that the use of this toolbox provides a platform for much-needed standardisation of sleep data usage, which in turn will also help advance the development of algorithms for automatic sleep staging.

References

[1] PhysioNet. (2013) Sleep-EDF Database. [Online]. Available: http://www.physionet.org/physiobank/database/sleep-edf/.

[2] ——. (2014) Sleep-EDF Database [Expanded]. [Online]. Available: http://www.physionet.org/physiobank/database/sleep-edfx/.

[3] C. O'Reilly, N. Gosselin, J. Carrier, and T. Nielsen, "Montreal archive of sleep studies: an open-access resource for instrument benchmarking and exploratory research," *J. Sleep Res.*, vol. 23, no. 6, pp. 628–35, 2014.

[4] University of MONS - TCTS Laboratory. (2015) The DREAMS Subjects Database. [Online]. Available: http://www.tcts.fpms.ac.be/~devuyst/Databases/DatabaseSubjects/.

[5] F. Yaghouby, P. Modur, and S. Sunderam, "Naive scoring of human sleep based on a hidden markov model of the electroencephalogram," in *IEEE EMBC*, Chicago, August 2014.

[6] K. Aboalayon, H. Ocbagabir, and M. Faezipour, "Efficient sleep stage classification based on EEG signals," in *IEEE LISAT*, New York, May 2014.

[7] T. Sanders, M. McCurry, and M. Clements, "Sleep stage classification with cross frequency coupling," in *IEEE EMBC*, Chicago, August 2014.

[8] J. L. Rodriguez-Sotelo, A. Osorio-Forero, A. Jimenez-Rodriguez, D. Cuesta-Frau, E. Cirugeda-Roldan, and D. Peluffo, "Automatic sleep stages classification using EEG entropy features and unsupervised pattern analysis techniques," *Entropy*, vol. 16, no. 12, pp. 6573–6589, 2014.

[9] MathWorks. (2015) Matlab and simulink for technical computing. [Online]. Available: http://www.mathworks.com/

[10] European Data Format. (2013) Home page. [Online]. Available: http://www.edfplus.info/.

[11] A. Delorme and S. Makeig, "EEGLAB: an open source toolbox for analysis of single-trial EEG dynamics," *J. Neurosci. Methods*, vol. 134, no. 1, pp. 9–21, 2004.

[12] V. Mihajlovic, B. Grundlehner, R. Vullers, and J. Penders, "Wearable, wireless EEG solutions in daily life applications: What are we missing?" *IEEE J. Biomed. Health Inform.*, vol. 19, no. 1, pp. 6–21, 2014.

[13] Anas Imtiaz. (2015) Sleep EDFx Toolbox - GitHub. [Online]. Available: https://github.com/anas-imtiaz/sleep-edfx-toolbox

C Automatic detection of sleep spindles

The research presented within this chapter is an edited version of research previously published in:

S. A. Imtiaz and E. Rodriguez-Villegas, "Evaluating the use of line length for automatic sleep spindle detection," in proceedings of the 36th international conference of the IEEE Engineering in Medicine and Biology Society, Chicago, August 2014, pp. 5024–5027, © IEEE.

S. A. Imtiaz, S. Saremi-Yarahmadi and E. Rodriguez-Villegas, "Automatic detection of sleep spindles using Teager energy and spectral edge frequency," in proceedings of the IEEE Biomedical Circuits and Systems Conference (BioCAS), Rotterdam, October 2013, pp. 262–265, © IEEE.

C.1 Introduction

Sleep spindle is a micro-event of sleep EEG and is a characteristic of NREM stages of sleep. It is a transient waveform with waxing-waning morphology and exhibits strong presence in stage 2 of NREM sleep (N2), although it may be present in N3 stage with a lower rate of occurrence. According to the American Academy of Sleep Medicine, a sleep spindle is defined as *"a train of distinct waves with frequency 11-16 Hz (most commonly 12-14 Hz) with a duration $\geq$ 0.5 seconds"* [1]. An example of typical sleep spindles in stage 2 of NREM sleep is shown in Figure C.1.

The number of sleep spindles observed during an overnight sleep is in the range of 200-1000 [2]. They are used by sleep specialists as one of the characteristic features

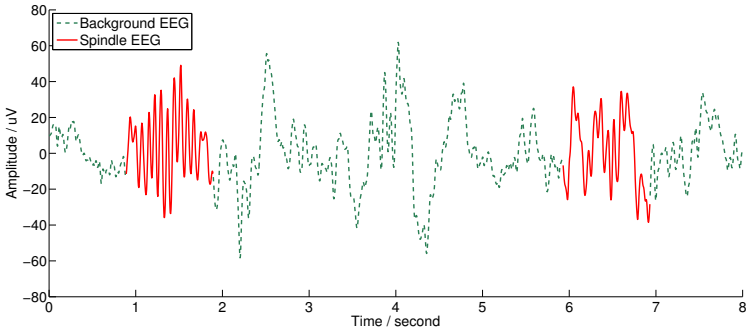


Figure C.1: *A typical sleep spindle (between dashed lines)*

when determining the appropriate stage of sleep. Identification of sleep spindles are of particular interest because of their role in classifying N2 stage of sleep. They need to be identified in order to mark the beginning and continuation of N2 phase of sleep. Scoring them, however, is a laborious task and prone to human error. There may be subjective differences in multiple spindles scored by a single scorer and between different scorers. A previous study showed the variability between different scorers to be around 20% [3]. Hence automatic detection of spindles is desirable in order to save time and reduce variability.

Although the use of spindles for sleep staging is frequent and well established, their significance as a sleep event, otherwise, is less commonly known and is an area of active research. They are known to play a fundamental role in memory consolidation during sleep [4], and are believed to be related to the secretion of melatonin, that helps in maintaining circadian rhythms in the body [5]. They are also understood to be sleep maintaining events having an active role in the progression of sleep to slow wave stages.

It has been suggested that sleep spindles may be a relevant indicator for early stage development of CNS [6]. A study exploring the link between spindle activity and major depressive disorder (MDD) found decreased spindle activity during sleep in youths with MDD and those with high risk for the disorder. It concluded that youths with reduced spindle activity show vulnerability to depression [7]. A number of studies have shown sleep spindles to be an indicator of intellectual ability in individuals. A good review of such studies appears in [4] and [6]. Another study showed direct correlation between the number of sleep spindles and IQ scores where individuals with more sleep spindles showed higher IQ scores [8].

Automatic identification of sleep spindles is, therefore, not only highly useful for sleep scoring but also for research studies looking at the general role and significance of spindles. This chapter presents the literature review of various automatic sleep spindle detection methods. Next, two different algorithms for automatic detection of sleep spindles using a single channel of EEG are presented. The first algorithm focuses on achieving high performance and uses Teager energy and spectral edge frequency to mark sleep spindles. The second algorithm evaluates the use of *line length*, an efficient and low-complexity time domain feature, for automatic detection of sleep spindles. This second algorithm is also implemented on a MSP430 microcontroller to demonstrate its low power consumption which can be suitable for use in wearable and resource-constrained systems. Finally, the performances of both these algorithms are also compared against other spindle detection methods evaluated on the same dataset.

C.2 Literature review of spindle detection algorithms

Researchers have applied various analysis methods to study sleep spindles in order to elicit suitable features for their automated detection. Techniques such as matching pursuit [9], [10], higher order statistics [11] and independent component analysis [12] have been used to show characteristic features of sleep spindles that could be useful for their identification. These methods, and many others, have been used to develop complete algorithms for automatic sleep spindle detection and will be discussed briefly in this section together with their detection performance. A summary of these methods is shown in Table C.2.

Gorur *et al.* [13] used artificial neural networks and support vector machines for detection of sleep spindles using STFT to extract input features. They used an ANN architecture with 32 input nodes and 60 nodes in the hidden layer. Their test data comprised of 1142 30-second epochs and achieved an accuracy of 88.7%. They compared the performance of ANN and SVM classifiers and concluded that SVM results in better detection rate with an average of 95.4%. Acır *et al.* [14] used autoregressive modelling coefficients with ANN and SVM independently to identify spindles and reported sensitivity of 89.1% and 94.6% respectively for the two classifiers. They used multi-channel EEG and tested on 6 subjects having a total of 264 sleep spindles.

Causa *et al.* [15] described an intensive method for sleep spindle detection in children. They used fixed power threshold in frequency domain to find candidate spindle zones and then used amplitude and duration criteria to remove false detections and mimic expert analysis. They used 40412 sleep spindles to test their method and reported an impressive sensitivity of 88.2% together with specificity and selectivity values of 89.7% and 88.1% respectively.

Devuyst *et al.* [16] used two detection schemes in parallel for spindle detection. In this method, they first filter the input signal in the spindle frequency range, use duration information and variable amplitude threshold based on signal's statistical properties to mark a candidate spindle. They also use FFT to check if the maximum signal frequency lies in a narrow spindle frequency range. Both conditions have to be true in order to mark a spindle. This method gives a sensitivity of 78.4%. The same group used bandpass filtering and level detection with autoregressive modelling and reported spindle detection sensitivity and specificity values at 70.2% and 98.6% respectively [17].

Schönwald *et al.* [18] used matching pursuit and reported both sensitivity and specificity at 81.6% for detecting 725 sleep spindles. Bodizs *et al.* [19] used an individual adjustment method for bandpass filtering and achieved a sensitivity of 92.9% with selectivity of 41.6% only when tested on 2140 sleep spindles. An individualised approach is also taken in the PRANA software package by PhiTools [20] that detects sleep spindles based on the input provided by the user for the average amplitude of spindles. The user input makes the detections more accurate and adaptable at the cost of the system

being not fully automatic. The software's performance was validated to show sensitivity and specificity of 98.96% and 88.49% respectively when tested on a subset of N2 stage segments from 10 subjects [21].

Duman *et al.* [22] used multiple approaches including Teager energy, harmonic decomposition and normalised amplitude at 12 Hz to identify spindles automatically. A detection is considered valid if all three methods have a positive output. They reported sensitivity and specificity of 96.2% and 95.5% respectively and tested on 16 subjects. The performance metrics, however, appear flawed such that they are marking a true positive when the algorithm detects the presence of spindles in a 30-sec epoch of N2 stage. Ahmed *et al.* [23] also used Teager energy as a feature for spindle detection together with Wavelet packet energy ratio and reported a sensitivity of 93.9% by testing their method on 95 sleep spindles.

Methods based on amplitude and frequency characteristics to mimic visual detection are by far the most common approach used in literature. Huuppopen *et al.* [24] described and compared the performance of four sleep spindle detection methods. The best performing method used FFT spectrum and spindle amplitude analysis and was tested on 12 subjects with 6043 spindles. They reported 70% sensitivity and 98.6% specificity for spindles in N2 stage. Wendt and Christensen [25] used bandpass filtering with level detection and power features in two EEG channels. Their method yields sensitivity and specificity values of 84.6% and 95.3% respectively. They used a test database of 882 spindles in 375 epochs but did not consider Wake segments of sleep. Nonclercq *et al.* [26] used normal modelling of spindle distribution based on amplitude and frequency features and tested the algorithm on two datasets, one for children and the other for adults. They reported sensitivity/specificity values of 78.5%/94.2% and 75.1%/96.7% respectively for the two datasets.

Table C.1: *Literature review summary for automatic sleep spindle detection.*

Ref	Data	Channels	Method	Result
[13]	689 epochs, 344.5 minutes	1×EEG	STFT features with ANN and SVM	Accuracy (ANN): 88.7% Accuracy (SVM): 95.4%
[14]	6 subjects, 264 spindles	Multiple	AR modelling coefficients with ANN and SVM	Accuracy (ANN): 89.1% Accuracy (SVM): 94.6%

Table C.1: *Literature review summary for automatic sleep spindle detection.*

Ref	Data	Channels	Method	Result
[16]	6 subjects, 575 spindles	1×EEG	Bandpass filtering with a varying amplitude threshold and maximum frequency range	Sen: 78.4% Spe: 88.6%
[6]	725 spindles	5×EEG	Matching pursuit	Sen: 81.6% Spe: 81.6%
[24]	12 subjects, 6043 spindles	4×EEG	FFT spectrum and amplitude analysis	Sen: 70% Spe: 98.6%
[19]	12 subjects, 2140 spindles	multiple	Individually adjusted bandpass filtering	Sen: 92.9% Sel: 41.6%
[22]	16 subjects	1×EEG	Max frequency, Teager energy and harmonic decomposition	Sen: 96.2% Spe: 95.4%
[23]	95 spindles	N/A	Teager energy and wavelet packet energy ratio	Sen: 93.9%
[21]	10 subjects	1×EEG	Manual amplitude threshold for PRANA software	Sen: 99% Spe: 88.5%
[15]	56 children, 40412 spindles	2×EEG	FFT, power thresholding, EMD signal decomposition, HHT	Sen: 88.2% Spe: 89.7% Sel: 88.1%
[17]	6 subjects, 537 spindles	1×EEG	Bandpass filtering with thresholding, relative power, AR modelling	Sen: 70.2% Spe: 98.6%
[25]	13 subjects, 882 spindles	2×EEG	Bandpass filtering with thresholding and power features	Sen: 84.6% Spe: 95.3%
[26]	5 children (1), 6 adults (2)	1×EEG	Amplitude-frequency normal modelling	Sen1: 78% Spe1: 94% Sen2: 75% Spe2: 97%

Most of the methods cited above achieved good performance with sensitivity of at least 70%. However direct comparison between them is difficult because of the varying dataset and performance metrics used in each.

C.3 Issues with automatic spindle detection

The previous section covered a number of automatic spindle detection methods. There appears to be a lack of standard reporting metric and that makes comparing algorithms a difficult task. Other than that, although the frequency range of spindles have been defined in the AASM scoring manual, it is rarely used in practice. Furthermore, since the amplitude range of spindles is not defined it is very subjective. Such issues of selecting the ranges and metrics, and other detection challenges will be discussed in this section.

C.3.1 Frequency range of spindles

Sleep spindles are defined to have a frequency range of 11-16 Hz (most commonly 12-14 Hz) [1]. However, different methods use a wider or narrower range for detection. Himanen *et al.* [27] showed that the frequency of spindles varies throughout the night and shows a U-turn within the first four NREM episodes. They remain within the range of 10.3-15.6 Hz during N2 stage of sleep. Nonclercq *et al.* [26] provide a comprehensive list of different frequency ranges selected by various authors. These ranges include, 12-14 Hz, 11.5-16 Hz, 10-16 Hz, 11-15 Hz, 12-16 Hz and more. This gives an idea of the variability in frequency ranges that exist in literature.

C.3.2 Amplitude

Similar to the variable frequency range, the amplitude of spindles is also a varying quantity in literature with authors having used values ranging from 8 μV to 25 μV in order to determine a suitable threshold for detection. Huupponen *et al.* [28] proposed a method to extract an optimal detection threshold for spindles.

C.3.3 Performance metrics

A quick glance at the last column of Table C.2 reveals the different metrics being used by researchers to report the performance of their algorithms. Moreover, some of those characterise the performance only in N2 stage assuming the existence of spindles in that very stage. Some choose to ignore the Wake stage while few count the presence of spindle within a standard epoch of 30 seconds good enough to be marked as a valid detection. Devuyst *et al.* [17] attempt to tackle this problem by proposing a standard assessment method for spindle detection. They also advocate the use of a common database (that they provide) for easier and more accurate comparison of algorithms.

C.3.4 Alpha rhythms

The variable frequency range of spindles was discussed in Section C.3.1. Even when the defined range of 11-16 Hz is used there may be spindle-like alpha (8-13 Hz) intrusions

resulting in false detections. These common occurrences have been discussed by several researchers [17], [28] and have to be taken care of during the development of any spindle detection algorithm.

C.3.5 Complexity

If a spindle detection needs to be incorporated in a wearable or real-time device then it goes without saying that it has to have low complexity to run within real-time constraints and keep the power requirements down.

C.4 Material

Polysomnography data from the DREAMS Sleep Spindles Database of University of MONS - TCTS Laboratory and Universite Libre de Bruxelles - CHU de Charleroi Sleep Laboratory [29] was used for the analysis and development of algorithms. The data consists of 30-minute excerpts from six subjects (3 males and 3 females, average age 45.7 years) with various sleep pathologies. The excerpts have been marked visually by two scorers for sleep spindles and the union of their markings is taken as the reference set. Data was originally recorded using two EOG, three EEG and one EMG channels. The sampling frequency is 200 Hz and the annotated channel is Cz-A1 except for subjects 1 and 3, for whom the annotated channel is C3-A1 and the sampling frequency 100 Hz and 50 Hz respectively. All signals were first resampled to have a uniform sampling frequency of 256 Hz using the MATLAB function *resample*. Of the entire 180 minutes of data available for testing, an average of 56% was part of N2 stage sleep, confirmed by the hypnogram provided with the data. No attempt was made to exclude either noisy data segments, or data from any other stages of sleep or wakefulness.

Each subject's data is a 30 minute segment from an overnight PSG recording. The percentage of all sleep stages in the test segment is shown in Table C.2. The total number of spindles for each subject and the sleep stage in which they were marked are given in Table C.3.

Table C.2: *Percentage of sleep stages in test data*

Subject / Sleep Stage %	W	N1	N2	N3	R
1	15.56	3.33	61.11	20	0
2	1.11	5.56	55.56	37.78	0
3	8.89	24.44	62.22	4.44	0
4	30	22.22	40	7.78	0
5	11.11	2.22	57.78	38.89	0
6	3.33	3.33	61.11	32.33	0

Table C.3: *Sleep spindles in each stage of the test data*

Subject / Spindles	Total	W	N1	N2	N3	R
1	134	0	0	101	33	0
2	77	0	1	68	8	0
3	44	4	0	38	2	0
4	63	31	5	25	2	0
5	103	0	0	82	21	0
6	117	0	0	90	27	0

It should be noted particularly from Table C.2 that subject 4 has a high Wake stage content (30%) in the recording. Additionally, Table C.3 shows that almost half of the spindles visually scored in this record are part of the Wake stage. All other cases have most spindles distributed among N2 and N3 stages.

C.5 Performance metrics

The spindle detection algorithms proposed in this chapter are tested on 30-minute EEG excerpt from six subjects. The union of spindles scored visually by two scorers is taken as the reference to compare the results against. The metrics proposed by Devuyst *et al.* [17] are used to characterise the performance of the two algorithms. These metrics are briefly explained in this section.

An automatically detected spindle is marked as True Positive (TP) if it overlaps at least partially with the reference spindle at that time. If no point of the detected spindle overlaps with the reference, it is considered as a false detection and marked as False Positive (FP). The number of spindles that went undetected by the algorithm are classified as False Negatives (FN). The number of True Negatives (TN) is approximated as shown below, where the average duration of the detected spindles must be computed separately for each subject.

$$TN = \frac{Total\ record\ duration}{Avg.\ detected\ spindle\ duration} - TP - FP - FN \tag{C.1}$$

The performance of the algorithms will be characterised by finding the Sensitivity and Specificity for each subject individually as well as for all of them combined.

$$Sensitivity = \frac{TP}{TP + FN} \tag{C.2}$$

$$Specificity = \frac{TN}{TN + FP} \tag{C.3}$$

C.6 Spindle detection: Algorithm I

An algorithm for automatic sleep spindle detection is presented in this section using only one channel of EEG input.

C.6.1 Methods

In this section, the various features required for the sleep spindle detection algorithm are defined.

Teager energy operator

Teager energy operator (TEO) is a non-linear operator that can estimate the energy of a signal on-the-fly [30], [31]. It is particularly useful in highlighting the abrupt transitions in a signal while suppressing the soft transitions. For a discrete-time signal, its Teager energy is computed as follows [31]:

$$\psi[x(n)] = x^2(n) - x(n+1)x(n-1) \tag{C.4}$$

Teager energy operator, ψ, when applied to EEG signals, appropriately filtered for sleep spindle detection, demonstrates a rise in energy level when a spindle appears. The sudden change in frequency and the waxing and waning amplitude of sleep spindles is well tracked by the Teager energy operator.

Spectral edge frequency - 50%

Spectral edge frequency at 50% (*SEF50*) is the frequency below which half of the signal power is present. This is equivalent to the median frequency of the signal. It can be computed from the magnitude of FFT coefficients (mag) as shown below, where n is the total number of FFT coefficients and x is the index to solve the equation for. The required frequency is then the x_{th} frequency from the array of FFT frequency components.

$$\sum_{i=1}^{x} |mag|^2 = 0.50 \times \sum_{i=1}^{n} |mag|^2 \tag{C.5a}$$

$$SEF50 = freq(x) \tag{C.5b}$$

In this work, *SEF50* is analysed in the 8-15 Hz frequency range since it covers both alpha (8-13 Hz) and spindle frequency range. It is found that spindle-like alpha rhythms have lower median frequency in this range and therefore this feature will be used to reduce the number of false detections.

C.6.2 Algorithm

In this section an algorithm for automatic detection of sleep spindles is developed using the two features described in the previous section.

A block diagram of the complete spindle detection algorithm is shown in Figure C.2. EEG input signal is first filtered using a first order high-pass filter with a cut-off frequency of 0.16 Hz, followed by a second order low-pass filter with 50 Hz as the cut-off frequency at the preprocessing stage. This bandlimited signal is used as input to the main spindle detection algorithm. The input signal is then filtered with a fourth order Butterworth band-pass filter with lower and upper cutoff frequencies 11 Hz and 16 Hz respectively, which is the spindle frequency range. This step gets rid of all the frequency content that is not of interest for spindle detection. This filtered signal is then segmented into epochs of 0.25 seconds with 50% overlap between successive ones. Teager Energy of the filtered signal is then determined in each epoch using Equation C.4. If the Teager energy values of all samples within the epoch are greater than a certain varying threshold, the epoch is marked as a candidate spindle. Because of the segmentation of epochs it is possible that a potential sleep spindle detected in one epoch may have started in the previous epoch or carried over into the next epoch. For this reason, when an epoch is marked as a candidate spindle, the epochs immediately preceding and succeeding the current epoch are also marked as part of the current spindle thus creating a candidate spindle zone.

The threshold for an epoch to be marked as a valid candidate spindle is determined by taking the mean value of the Teager energy over 60 previous epochs. The threshold is then established as 2.19 times this mean value i.e. all samples in an epoch have to be greater than the running mean of last 60 epochs by a factor of 2.19 to be considered as a valid spindle. The number of epochs and multiplication factor for threshold are determined empirically. Establishing the threshold based on the data itself obviates the need for any manual patient specific adjustments.

At this stage, minimum and maximum spindle duration constraints are also applied prior to spectral analysis. If the duration of a candidate spindle is found to be greater than 3 seconds or less than 0.5 seconds, the candidate is discarded and not subject to any further analysis.

The next stage in the algorithm is enabled only when a candidate spindle is detected at first stage and obeys the duration constraints. Frequency content of each epoch in the preprocessed signal, corresponding to the epoch in candidate spindle zone, is analysed using a 512-point FFT. *SEF50* for each epoch in the 8-15 Hz band is computed using Equation C.5 and its *average* determined for all epochs in the candidate zone. If *SEF50* is less than a fixed threshold the candidate spindle is rejected otherwise the candidate spindle is deemed to be a positive detection. This stage is highly specific and helps in removing false spindles and alpha rhythms that may have been erroneously detected at the first stage. The threshold for *SEF50* is fixed at 10.7 Hz for all test cases. This was

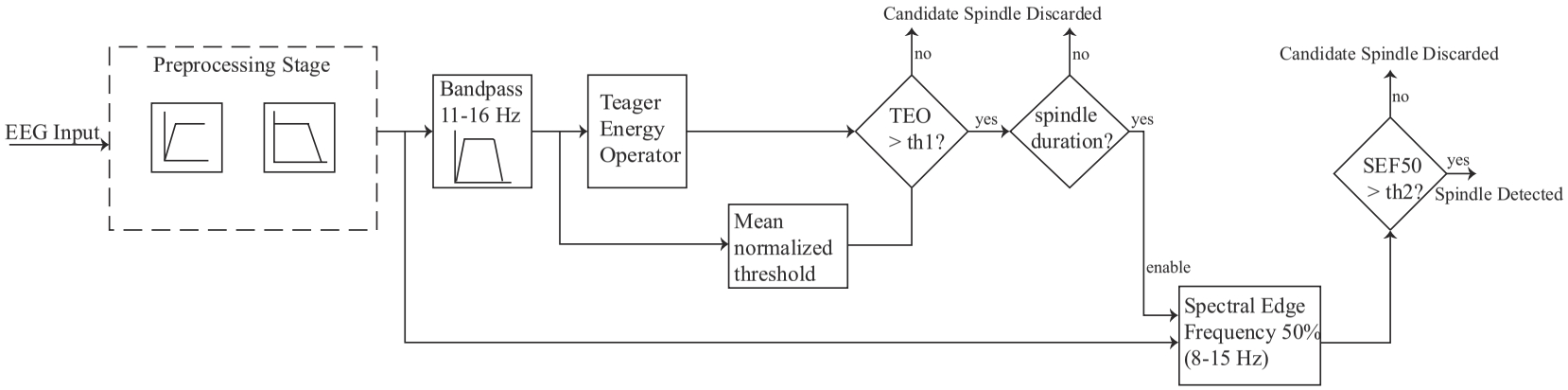


Figure C.2: *Block diagram of sleep spindle detection Algorithm I*

determined experimentally by analysing the frequency content of true spindles and false detections from the first stage.

The output from each stage of the algorithm is shown in Figure C.3 where the Teager energy block identifies four candidate spindles. One of them is a false detection which is subsequently rejected by analysing the frequency of the candidate spindles.

C.6.3 Results

The algorithm is tested on sleep data excerpts that included all sleep stages to reflect real world conditions. Its performance is evaluated at the output of each of the two stages to show the effect in performance of adding the second feature (*SEF50*) in addition to *TEO*. The detection results obtained using only *TEO* (i.e. stage one only) are shown in Table C.4. The results show a high sensitivity of TEO in detecting sleep spindles but there is also a high number of false positives. The highest contribution of false positives comes from subject 4.

Table C.4: *Performance of spindle detection using TEO (Algorithm I)*

Sub	Total	TP	FP	Sen(%)	Spe(%)
1	134	119	107	88.81	95.65
2	77	59	64	76.62	97.53
3	44	41	100	93.18	96.06
4	63	40	128	63.49	95.59
5	103	89	104	86.41	96.04
6	117	107	80	91.45	96.79
Total	538	455	583		
Avg.				83.33	96.28

The results in Table C.5 show the final performance of the algorithm with the second stage added. Of the 538 visually scored sleep spindles, the algorithm successfully detects 432 spindles yielding a sensitivity of 80.3% within $\pm$ 3.36% for a 95% confidence interval. The rejection of most of the background EEG as true negatives also leads to a high specificity of almost 98%.

The individual results are similar for all test cases except for a lower sensitivity in the case of subject 4. It was mentioned in Section C.4 that this subject has a high Wake stage content (30%) in the recording. Additionally, almost half of the spindles visually scored in this record were found to be part of the Wake stage when compared against the hypnogram provided with the database. The algorithm falsely detects only 14 of the 31 spindles in Wake stage while the other 24 true positive detections were part of NREM stages. With the Wake stage removed from analysis, sensitivity for subject 4 goes up to 75%.

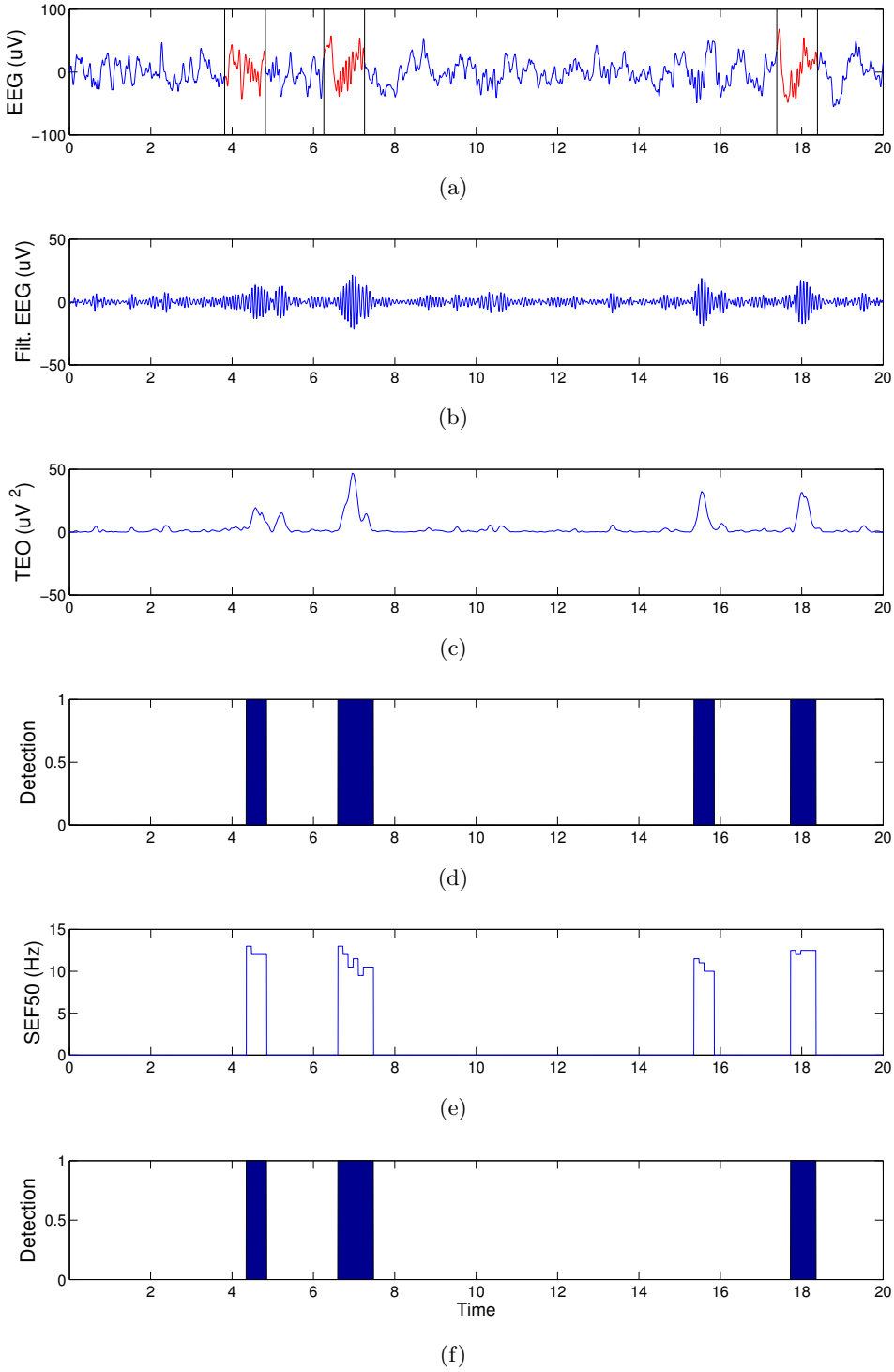


Figure C.3: *(a) EEG input with three spindles marked between vertical lines; (b) 11-16 Hz filtering output; (c) TEO output showing high activity in the spindle areas; (d) detected candidate spindles; (e)* SEF50 *for each epoch in the candidate spindle zone; (f) correctly detected spindles (removing one false candidate spindle)*

Table C.5: *Spindle detection performance of Algorithm I*

Subject	Total Spindles	True Pos.	Sens. (%)	Spec. (%)
1	134	111	82.8	96.7
2	77	58	75.3	98.3
3	44	39	88.6	97.7
4	63	38	60.3	97.8
5	103	87	84.5	97.1
6	117	99	84.6	98.1
All	538	432	80.3	97.6

The results are further analysed to determine the corresponding sleep stage in which each of the spindle is detected. Table C.6 shows the total number of sleep spindles detected by the algorithm for each subject and the number of detections in each sleep stage (classified according to AASM [1] rules). It can be seen that most of the sleep spindles detected are from N2 stage (about 69%) and more than 91% of spindles detected are from stages N2 and N3 combined. These are also the two stages where sleep spindles are most often observed.

Table C.6: *Sleep Spindles (SS) detected in each sleep stage - Algorithm I*

Sub	SS_{tot}	SS_W	SS_{N1}	SS_{N2}	SS_{N3}	SS_R
1	190	10	3	135	42	0
2	101	0	0	77	24	0
3	91	5	8	75	3	0
4	104	32	8	54	10	0
5	164	4	0	110	50	0
6	146	0	0	97	49	0
Total	796	51	19	548	178	0

The performance of this algorithm will be further discussed and compared against other methods in the literature in Section C.8.

C.7 Spindle detection: Algorithm II

In this section, the use of *line length*, an efficient and low-complexity time domain feature, for automatic detection of sleep spindles is evaluated.

C.7.1 Methods

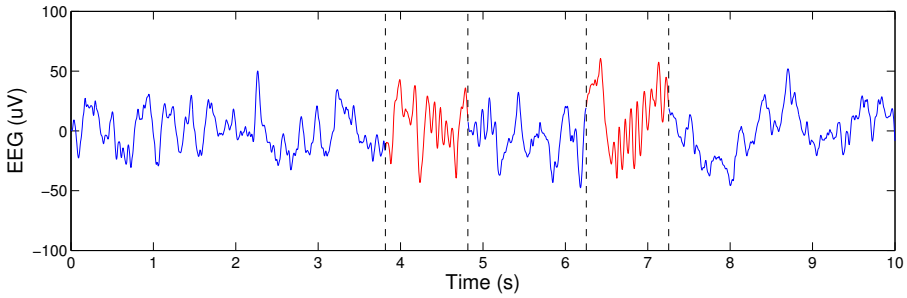
Esteller *et al.* [32] introduced *line length* as a low-complexity feature for seizure onset detection. It is the sum of absolute differences between subsequent samples and is defined by the following equation where LL is the line length, x is the input signal and N is the number of samples in the signal (or a block of signal under analysis).

$$LL = \sum_{n=1}^{N} |x(n-1) - x(n)| \tag{C.6}$$

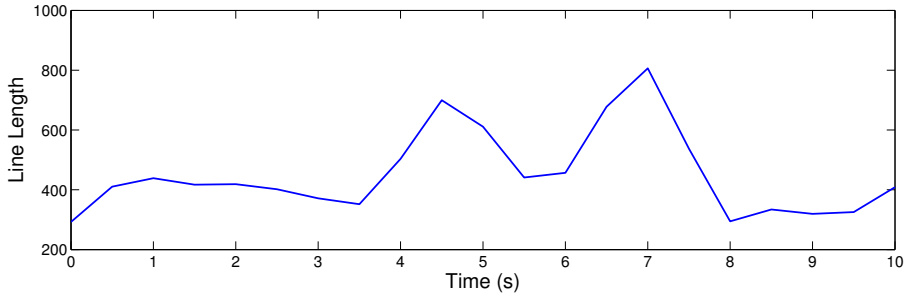
Figure C.4(a) shows an EEG signal with two sleep spindles. The line length corresponding to this signal, calculated in blocks of 1 second (with 50% overlap) is shown in Figure C.4(b). It can be seen on the figure that the occurrence of a spindle in the original signal leads to a rise in the line length. This lasts approximately until the end of spindle duration and returns to a lower level thereafter. This effect of line length having higher values during spindle occurrence can be used as a characteristic feature in an algorithm to demonstrate its utility for detecting spindles.

C.7.2 Algorithm

A block diagram of the algorithm with line length as the analysis feature is shown in Figure C.5. EEG signal for a single channel is used as input to the algorithm. At the first stage, a second order Butterworth bandpass filter is applied to the input signal to limit it in the 11-16 Hz frequency range, which is the spindle range of interest. The filtered signal is then partitioned into blocks of 1 second epochs with 50% overlap between subsequent epochs. The line length for each epoch is calculated using Equation C.6. It is normalised by a factor which is obtained by taking the median line length value of the last 80 epochs. This step obviates the need of any patient-specific adjustments. The number of epochs to use in the computation of the median was determined empirically by trying out various values. Initially when the number of epochs processed is less than 80, the median of all available previous values is taken. This normalised value is then compared against a detection threshold K. If the value is found to be greater than K, the epoch is marked as spindle. The detection threshold also controls the sensitivity of the algorithm, with higher values resulting in stricter classification criterion.



(a)



(b)

Figure C.4: *(a) EEG signal with two sleep spindles marked between vertical lines; (b) Line length of the EEG signal showing higher values during spindle occurrence*

C.7.3 Hardware implementation

The algorithm is implemented on Texas Instruments MSP430F5438A microcontroller [33] to measure its online performance and power consumption. All arithmetic operations are performed in fixed point arithmetic to make use of the hardware multiplier on the chip and the microcontroller is put to idle mode (LPM3) whenever there was no data to process. The coefficients of the bandpass filter are represented as fixed point numbers in Q15 format and the entire filtering operation requires only three 16-bit multiplication and addition operations. The line length is calculated by taking the absolute difference between each new and previous filtered sample and accumulating the result in a register which is initialised to zero at the start of an epoch. This way it is updated with each new sample without the need to store all previous samples in an epoch. The median is then computed using a linked list method described by Phil Ekstrom [34] that has a sorting complexity of N. The resulting value is used to normalise the line length. Finally, a detection flag is raised and time noted whenever the normalised line length is found to be greater than the threshold K.

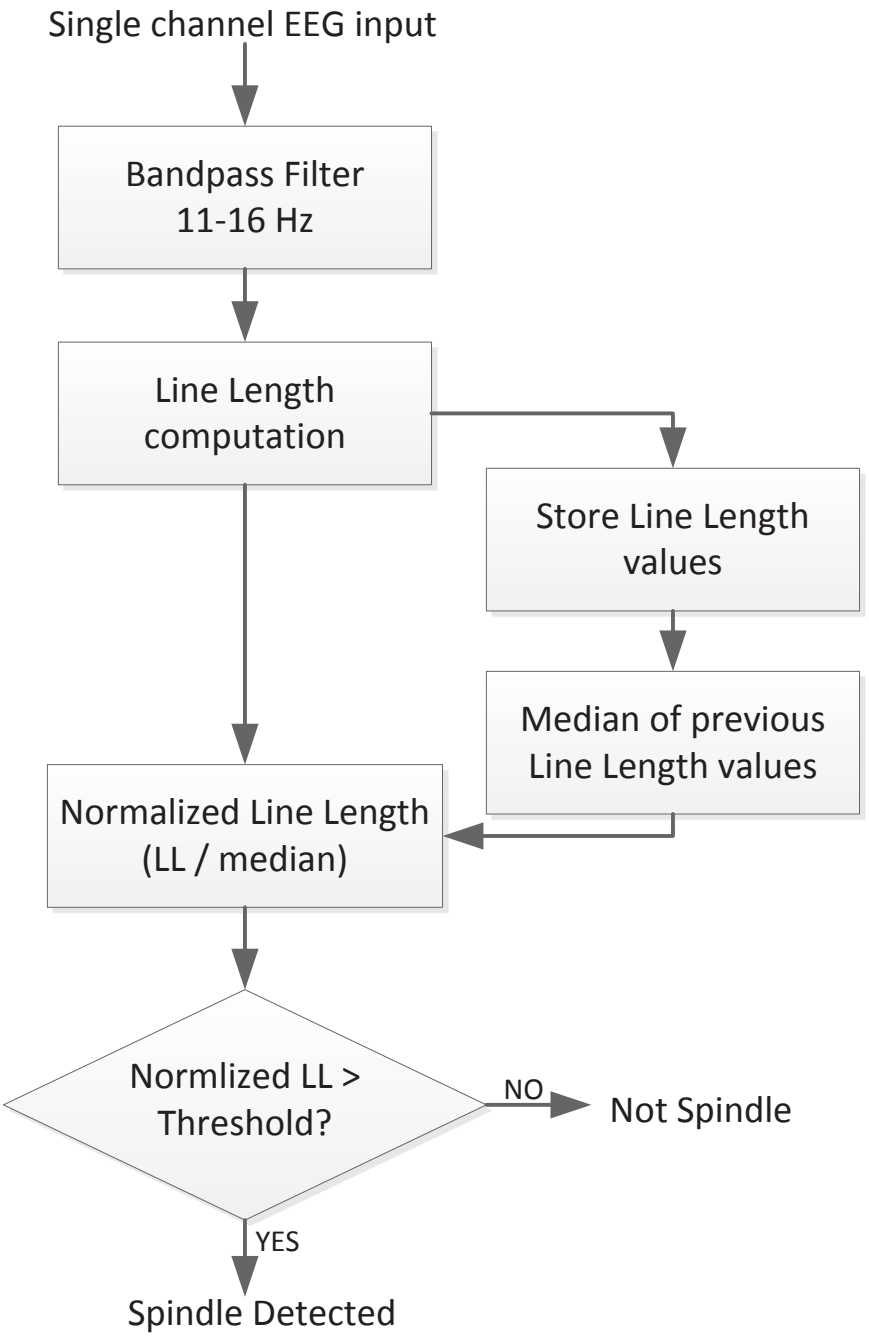


Figure C.5: *Block diagram of sleep spindle detection Algorithm II*

C.7.4 Results

Leave-one-out cross validation (LOOCV) is used to determine the detection performance of the algorithm on each subject while training it on the five remaining subjects. A receiver operating characteristic (ROC) curve is plotted for average sensitivity against specificity by varying the detection threshold K in small steps using five training subjects. The best performing threshold is determined from the curve as the point that maximised both sensitivity and specificity. This threshold is then used to test the performance of the remaining sixth subject. This procedure is carried out six times such that each subject becomes the test case once with other five being used for training.

The total number of spindles visually scored and those detected by the algorithm for each subject are shown in Table C.7. The average sensitivity of the algorithm is 83.6% and its 95% confidence interval range is between 80.5% and 86.8%. The sensitivity results are consistently greater than 80% for all cases except subject 4 where less than half of reference spindles are automatically detected (as in Algorithm I). In this case, the sensitivity is low because of the high proportion of Wake stages and artefacts present in the signal making detection difficult. The specificity in all the subjects is close to the overall average of 87.9% which shows that most of the non-spindle data has been successfully rejected.

Table C.7: *Spindle detection performance of Algorithm II*

Subject	Total Spindles	True Pos.	Sens. (%)	Spec. (%)
1	134	121	90.3	83.4
2	77	62	80.5	94.0
3	44	44	100	85.8
4	63	28	44.4	88.1
5	103	87	84.5	88.2
6	117	108	92.3	86.5
All	538	450	83.6	87.9

A breakdown of the spindles detected in each sleep stage is shown in Table C.8. Of the 841 spindles detected by the algorithm (SS_{det}) 61.5% true positives are in N2, 58.2% in N2 and N3 combined and 53.5% across all sleep stages. The proportion of false positives is highest in subjects 3 and 4. From the total detections made by the algorithm, 90% are in N2 and N3 combined and less than 6% are recorded in the Wake stage.

Table C.8: *Sleep Spindles (SS) detected in each sleep stage - Algorithm II*

Sub	SS_{det}	SS_{Wake}	SS_{N1}	SS_{N2}	SS_{N3}	SS_{REM}
1	202	13	1	141	47	0
2	99	0	2	84	13	0
3	126	11	11	95	9	0
4	87	16	23	42	6	0
5	154	7	0	103	44	0
6	173	0	1	114	58	0
Total	841	47	38	579	177	0

C.7.5 Power consumption

The epoch size used for processing is 1 second with a 50% overlap between subsequent epochs. This means that two epochs are required to be processed every second. The microcontroller was found to be active for only 10% of the time to perform all of the signal processing needed for the algorithm while spending the rest of the time in idle mode until a new sample arrives. Operating at a clock frequency of 1 MHz and supply voltage of 1.8 V the power consumption for the algorithm was found to be 56.7 μW with one channel of EEG input.

C.8 Discussion

For the two algorithms, all signals from the DREAMS spindles database are used *as is* in the analysis, including those with artefacts and noisy segments. The addition of any artefact removal preprocessing stage is likely to help improve the performance in such cases.

The first algorithm using Teager energy has been developed to achieve a high detection performance. Its first stage involves a highly sensitive and simple non-linear operator and with a normalised threshold it obviates the need for any patient specific adjustment externally. The second stage which is highly specific and involves computation of FFT is called upon only when there are candidate spindles thus reducing the processing load making this algorithm suitable for online implementation.

The second algorithm has been developed with power consumption as the main constraint as well as demonstrating the use of line length as a novel feature for spindle detection. Its spindle detection sensitivity was slightly higher (with more false positives) than the first algorithm. It was also implemented on a MSP430 microcontroller to measure its power consumption and determine its computational load. The overall system power consumption was 56.7 μW with only 10% of microcontroller active time

operating at clock frequency of 1 MHz. This shows that line length is not only useful for getting a good spindle detection performance but also a very efficient feature for use in resource-constrained wearable systems.

The performances of the two algorithms presented in this chapter are compared against other sleep detection methods that report their results on the same database. This comparison is shown in Table C.9. In contrast to Devuyst *et al.* [17], both algorithms detected a higher number of spindles overall. Algorithm I resulted in similar specificity highlighting its better performance both in terms of true positive detections and false positive rejections. Algorithm II, despite a better sensitivity, also resulted in a higher proportion of false positives resulting in a lower specificity. In another method, Nonclercq *et al.* [26] reported a sensitivity of 75.1% after normal modelling however they were not able to detect any spindles in subject 4 and averaging the results including this case reduces their method's sensitivity to 62.6% only. Nevertheless, they obtained specificity of more than 90% for other test subjects.

Table C.9: *Comparison of this work with other sleep spindle detection algorithms*

Method	Sens. (%)	Spec. (%)
Devuyst *et al.* [17]	70.2	98.6
Nonclercq *et al.* [26]	75.1	96.7
Algorithm I	80.3	97.6
Algorithm II	83.6	87.9

The results using Algorithm I show superior overall performance. Further, the results of the second algorithm, based on line length, show a higher sensitivity but a slightly reduced specificity. This reduction is because the average duration of detected spindles is higher in the latter case. In Algorithm I, average duration is approximately 1 second where as in Algorithm II, it is 2.7 seconds. This reduces the number of estimated true negatives in equation Equation C.1 and, consequently, the specificity. However, despite a lower specificity, using only one feature with a simple algorithm without any artefact rejection still demonstrates line length as a very useful feature for spindle detection.

References

[1] C. Iber, S. Ancoli-Israel, A. Chesson, and S. Quan, *The AASM manual for the scoring of sleep and associated events: rules, terminology and technical specifications.* Westchester, IL: American Academy of Sleep Medicine, 2007.

[2] E. Huupponen, G. Gómez-Herrero, A. Saastamoinen, J. Varri, A. Hasan, and S. Himanen, "Development and comparison of four sleep spindle detection methods," *Artif. Intell. Med.*, vol. 40, no. 3, pp. 157–170, 2007.

[3] K. Campbell, A. Kumar, and W. Hofman, "Human and automatic validation of a phase-locked loop spindle detection system," *Electroencephalogr. and Clin. Neurophysiol.*, vol. 48, no. 5, pp. 602–5, 1980.

[4] S. M. Fogel and C. T. Smith, "The function of the sleep spindle: A physiological index of intelligence and a mechanism for sleep-dependent memory consolidation," *Neurosci. Biobehav. R.*, vol. 35, no. 5, pp. 1154–1165, 2011.

[5] L. D. Gennaro and M. Ferrara, "Sleep spindles: an overview," *Sleep Med. Rev.*, vol. 7, no. 5, pp. 423–440, 2003.

[6] M. Schabus, K. Hödlmoser, G. Gruber, C. Sauter, P. Anderer, G. Klsch, S. Parapatics, B. Saletu, W. Klimesch, and J. Zeitlhofer, "Sleep spindle-related activity in the human EEG and its relation to general cognitive and learning abilities," *Eur. J. Neurosci.*, vol. 23, no. 7, pp. 1738–1746, 2006.

[7] J. Lopez, R. Hoffmann, and R. Armitage, "Reduced sleep spindle activity in early-onset and elevated risk for depression," *J. Am. Acad. Child Adolesc. Psychiatry*, vol. 49, no. 9, pp. 934–43, 2010.

[8] R. Nader and C. Smith, "The relationship between stage 2 sleep spindles and intelligence," *Sleep*, vol. 24, no. 1, p. A160, 2001.

[9] P. Durka and K. Blinowska, "Matching pursuit parametrization of sleep spindles," in *IEEE EMBC*, Amsterdam, October 1996.

[10] J. Żygierewicz, K. J. Blinowska, P. J. Durka, W. Szelenberger, S. Niemcewicz, and W. Androsiuk, "High resolution study of sleep spindles," *Clin. Neurophysiol.*, vol. 110, no. 12, pp. 2136–2147, 1999.

[11] T. Akgul, M. Sun, R. Sclahassi, and A. Cetin, "Characterization of sleep spindles using higher order statistics and spectra," *IEEE Trans. Biomed. Eng.*, vol. 47, no. 8, pp. 997–1009, 2000.

[12] E. Ventouras, I. Alevizos, P. Ktonas, H. Tsekou, T. Paparrigopoulos, I. Kalatzis, C. Soldatos, and G. Nikiforidis, "Independent components of sleep spindles," in *IEEE EMBC*, Lyon, August 2007.

[13] D. Gorur, U. Halici, H. Aydin, G. Ongun, F. Ozgen, and K. Leblebicioglu, "Sleep spindles detection using short time Fourier transform and neural networks," in *IEEE IJCNN*, Honolulu, May 2002.

[14] N. Acır and C. Güzeliş, "Automatic recognition of sleep spindles in EEG by using artificial neural networks," *Expert Sys. Appl.*, vol. 27, no. 3, pp. 451–458, 2004.

[15] L. Causa, C. M. Held, J. Causa, P. A. Estévez, C. A. Perez, R. Chamorro, M. Garrido, C. Algarín, and P. Peirano, "Automated sleep-spindle detection in healthy children polysomnograms," *IEEE Trans. Biomed. Eng.*, vol. 57, no. 9, pp. 2135–46, 2010.

[16] S. Devuyst, T. Dutoit, , J. Didier, F. Meers, E. Stanus, P. Stenuit, and M. Kerkhofs, "Automatic sleep spindle detection in patients with sleep disorders," in *IEEE EMBC*, Amsterdam, August 2006.

[17] S. Devuyst, T. Dutoit, P. Stenuit, and M. Kerkhofs, "Automatic sleep spindles detection overview and development of a standard proposal assessment method," in *IEEE EMBC*, Boston, September 2011.

[18] S. V. Schönwald, E. L. de Santa-Helena, R. Rossatto, M. L. Chaves, and G. J. Gerhardt, "Benchmarking matching pursuit to find sleep spindles," *J. Neurosci. Methods*, vol. 156, no. 1-2, pp. 314–321, 2006.

[19] R. Bodizs, J. Kormendi, P. Rigo, and A. S. Lazar, "The individual adjustment method of sleep spindle analysis: methodological improvements and roots in the fingerprint paradigm," *J. Neurosci. Methods*, vol. 178, no. 1, pp. 205–13, 2009.

[20] PhiTools. (2013) Home page. [Online]. Available: http://www.phitools.com/.

[21] L. B. Ray, S. M. Fogel, C. T. Smith, and K. R. Peters, "Validating an automated sleep spindle detection algorithm using an individualized approach," *J. Sleep Res.*, vol. 19, no. 2, pp. 374–8, 2010.

[22] F. Duman, A. Erdamar, O. Erogul, Z. Telatar, and S. Yetkin, "Efficient sleep spindle detection algorithm with decision tree," *Expert Sys. Appl.*, vol. 36, no. 6, pp. 9980–9985, 2009.

[23] B. Ahmed, A. Redissi, and R. Tafreshi, "An automatic sleep spindle detector based on wavelets and the teager energy operator," in *IEEE EMBC*, Minnesota, August 2009.

[24] E. Huupponen, G. Gomez-Herrero, A. Saastamoinen, A. Varri, J. Hasan, and S.-L. Himanen, "Development and comparison of four sleep spindle detection methods," *Artif. Intell. Med.*, vol. 40, no. 3, pp. 157–70, 2007.

[25] S. Wendt and J. Christensen, "Validation of a novel automatic sleep spindle detector with high performance during sleep in middle aged subjects," in *IEEE EMBC*, San Diego, August 2012.

[26] A. Nonclercq, C. Urbain, D. Verheulpen, C. Decaestecker, P. Van Bogaert, and P. Peigneux, "Sleep spindle detection through amplitude-frequency normal modelling," *J. Neurosci. Methods*, vol. 214, no. 2, pp. 192–203, 2013.

[27] S.-L. Himanen, J. Virkkala, H. Huhtala, and J. Hasan, "Spindle frequencies in sleep EEG show U-shape within first four NREM sleep episodes," *J. Sleep Res.*, vol. 11, no. 1, pp. 35–42, 2002.

[28] E. Huupponen, A. Varri, S.-L. Himanen, J. Hasan, M. Lehtokangas, and J. Saarinen, "Optimization of sigma amplitude threshold in sleep spindle detection," *J. Sleep Res.*, vol. 9, no. 4, pp. 327–334, 2000.

[29] University of MONS - TCTS Laboratory. (2015) The DREAMS Sleep Spindles Database. [Online]. Available: http://www.tcts.fpms.ac.be/~devuyst/Databases/DatabaseSpindles/.

[30] J. Kaiser, "On a simple algorithm to calculate the 'energy' of a signal," in *IEEE ICASSP*, Albuquerque, April 1990.

[31] ——, "Some useful properties of teager's energy operators," in *IEEE ICASSP*, Minneapolis, April 1993.

[32] R. Esteller, J. Echauz, T. Tcheng, B. Litt, and B. Pless, "Line length: an efficient feature for seizure onset detection," in *IEEE EMBC*, Istanbul, October 2001.

[33] Texas Instruments. (2013) MSP430 ultra-low power 16-bit microcontrollers. [Online]. Available: http://www.ti.com/msp430/.

[34] P. Ekstrom. (2000) Running median better than average. [Online]. Available: http://www.eetindia.co.in/STATIC/PDF/200011/EEIOL_2000NOV03_EMS_EDA_TA.pdf

D Compression in wearable sensor nodes for data transmission and storage

The research presented within this chapter is an edited version of research previously published in:

S. A. Imtiaz, A. J. Casson and E. Rodriguez-Villegas, "Compression in Wearable Sensor Nodes: Impacts of Node Topology," IEEE Transactions on Biomedical Engineering, vol. 61, no. 4, pp. 1080–1090, 2014, © IEEE.

D.1 Introduction

It is not always possible to perform all signal processing at the sensor node. In such cases raw data is either wirelessly transmitted to a nearby receiver or stored on local memory for offline usage and analysis later pushing the algorithm complexity to the receiver end. Even when this system architecture is used the data rate may be prohibitively high to be transmitted within the available power budget. It is widely accepted that the power consumption of sensor nodes can potentially be reduced by the inclusion of online, real-time, data compression embedded in the node itself [1]–[3]. The challenge lies in having compression algorithms that provide a high level of data reduction while introducing little error into the recorded signal and which require very little power to operate.

This chapter presents a Texas Instruments MSP430 based sensor architecture for providing such compression, and in particular investigates the impact of the overall sensor node topology on the compression performance. Both wireless sensor nodes, where the sensor incorporates a wireless transmitter for passing the collected data to a processing smartphone or computer in real-time, and local memory sensor nodes, where memory incorporated in the sensor is used to store the data until the node is physically connected to the processing computer, are widely used sensor node topologies. However these two different types of *back-end* have widely different requirements, constraints and impacts on the compression to be used. The reconstruction accuracy and power consumption in four different sensor nodes, two from each topology type, will be investigated in this chapter to provide a real system comparison of state-of-the-art compression for sensor nodes.

Compressive sensing is used as the compression basis as it is a relatively new technique very suitable for use in power constrained sensor nodes: the computational complexity

of the different compression stages naturally fit within the power budgets available in the different stages of wearable sensor systems [4]. The fundamental compression step is simply a random sampling of the input signal and so has a low complexity for running online in the very power constrained sensor node. This low computational complexity is traded-off against having a higher complexity when the input signal needs to be reconstructed from the compressed samples. However, this stage can be run on a smartphone or fixed computer installation with much relaxed power constraints compared to the sensor itself. Compressive sensing itself can be implemented in either analogue or digital domains. In the former case, an ADC with a sub-Nyquist frequency is used to reduce the sampling rate (and consequently reduce data) while the latter uses a standard Nyquist ADC followed by a sparse sensing matrix for data reduction. A complete comparison of these two compressive sensing implementations and their merits is discussed in [5]. In this chapter, compressive sensing in the digital domain is used for EEG signals that are generally sampled at low frequencies in the range 200-1000 Hz. Since these sampling rates are easily achievable, no further reduction in sampling frequency is desired (which is the main motivation for using analogue-domain compressive sensing) hence digital compressive sensing is preferred for data reduction.

The underlying theory of compressive sensing has been established previously [6], [7], as has the reconstruction performance in a number of applications [8]–[13]. In this work, compressive sensing will be evaluated in terms of *both* reconstruction performance and power consumption of the system including the back-end. The impact on power consumption of using different wireless transceiver and local storage configurations will be analysed. Further, the reconstruction accuracy performance by using fixed/floating point realisations and different processing frame sizes will also be studied.

Section D.2 outlines the four prototype sensor nodes used, all of which are MSP430 based, with this being connected to a wireless transmitter on the same chip, an external wireless transmitter chip, a NAND flash memory chip, and a microSD card incorporating a FAT16 file system. Section D.3 details the performance assessment of these different nodes using scalp EEG signals as an example medical application. Section D.4 presents detailed results for the performance of different compressive sensing configurations in terms of both reconstruction accuracy and power consumption, and these are discussed in Section D.5 to provide an overview of the relative performance, advantages, and disadvantages, of the different sensor node topologies.

D.2 Sensor platforms

D.2.1 Hardware set up

Due to its ultra low power consumption the Texas Instruments MSP430 family [14] is a popular hardware platform for wearable sensor nodes (for example it is used in

the TinyNode, Tmote Sky, and Shimmer systems [15]–[17]) and it is the basis of the sensor nodes used in this work. Three of these prototypes use the MSP430F5438A microcontroller connected to different back-end systems as discussed below, while the fourth uses the CC430F6147. This is a single *system-on-chip* solution combining both an MSP430 core and a wireless transceiver on the same microchip allowing significant miniaturisation and cost savings. The set up of these four systems is shown in Figure D.1.

The four different prototypes across the two wireless and local memory topologies are implemented using:

1. A Nordic nRF2401+ wireless transmitter.
2. A Texas Instruments CC430 wireless transmitter.
3. A microSD card.
4. A Hynix NAND flash memory.

Wireless node topologies (1 and 2) provide flexibility, ease of use, and allow real-time access to the collected signals. Local memory topologies (3 and 4) do not provide real-time access to the data being collected, but are still of significant interest as there is no risk of missing packets, and due to their low power operation. Generally, with no compression present, local memory based systems have a lower average current consumption than wireless systems and this is demonstrated here in Section D.4.

D.2.2 Wireless nodes

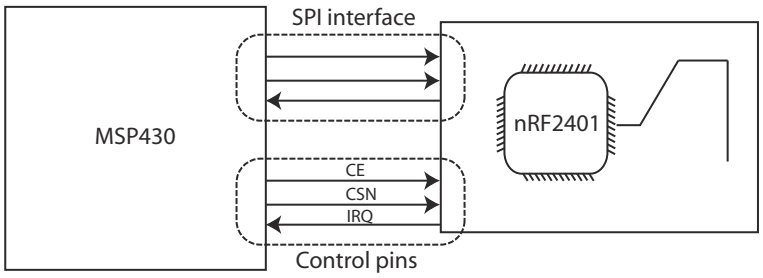
The wireless nodes are set up as:

Nordic nRF2401+

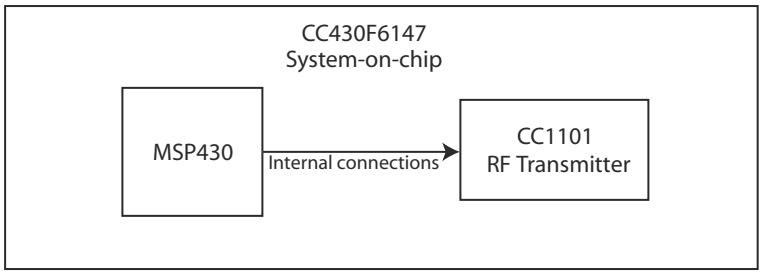
The first prototype wireless node uses a Nordic Semiconductor nRF2401+ transceiver [18] set with a transmit power of 0 dBm, 2 Mbps over-the-air data rate, and 2.4 GHz operation frequency. Each time a frame of samples is passed to the transmitter by the MSP430 the data is transmitted as quickly as possible over the 2 Mbps link with the transmitter turned off the rest of the time to save power. The Nordic Shockburst protocol is used with a 32 byte payload size and five preamble, address and CRC (Cyclic Redundancy Check) bytes. It operates from a 2 V supply.

TI CC430 wireless transmitter

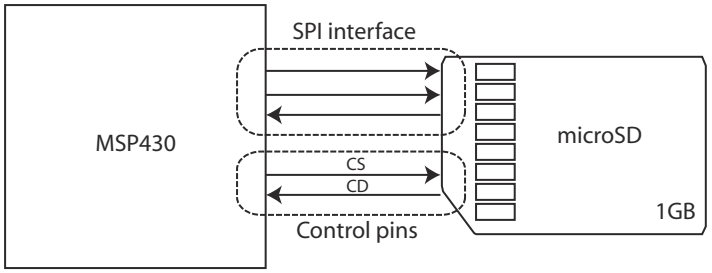
The CC430 node is based upon the CC430F6147 [19] which incorporates both an MSP430 core and a CC1101 sub-GHz wireless transceiver. It is set to operate at 868 MHz, with a transmit power of 0 dBm, 64 byte payload size and three preamble and CRC bytes. This node is powered using a 3 V supply.



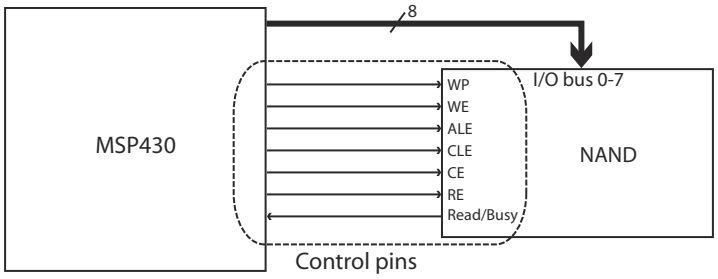
(a)



(b)



(c)



(d)

Figure D.1: *Physical set up of the prototype sensor nodes with the four different back-ends. (a) Nordic transmitter; (b) CC430 transmitter; (c) microSD card; (d) NAND flash memory chip.*

D.2.3 Local memory nodes

The local memory sensor nodes are set up as below. In these there is a choice of both the type of memory and the file system used. All microSD cards require a file system hence FAT16 is used for universal support across many commercial PCs with integrated card readers, making additional reader software unnecessary. In contrast the NAND memory node uses no standardised file system with data stored as simple binary files. The absence of a file system simplifies the data writing process and reduces the processing and memory burden on the microcontroller during the collection of data. This simplicity comes at the cost of more complexity when passing the data to an interpreting computer, but the sensor node can generally be powered by the interpreting computer during these operations.

microSD card

The microSD based sensor makes use of a class 4 microSD card and the MSP430 implements a customised version of the freely available Fatfs library [20] to access it. This allows cards up to 2 GB in size to be written to, enough space for at least a day of continuous operation. As more compression is employed in the system this duration will increase. A 3.3 V supply is required.

Hynix NAND flash memory

The NAND sensor node uses a Hynix NAND HY27UF081G2A flash chip [21]. This is a standard large block sized NAND device and so is representative of all large memory NAND units, although the precise model used at present only has a 128 MB capacity. It operates from a 2.7 V supply.

D.2.4 Compression design

The MSP430 is responsible for the set up and access of the back-end, and for performing low power digital compressive sensing to minimise the amount of data that needs to be transferred. For each input signal $\mathbf{x}$—in this case an EEG channel recorded from the human scalp—a compressively sensed representation is generated in the digital domain by carrying out the matrix multiplication

$$\mathbf{y} = \mathbf{\Phi x}. \tag{D.1}$$

Here $\mathbf{x}$ is a non-overlapping frame of N EEG samples, and $\mathbf{\Phi}$ is an $M \times N$ sensing matrix. $\mathbf{y}$ has dimensions $M \times 1$ and so if $M < N$ data compression is achieved. Provided that $\mathbf{\Phi}$ is correctly chosen, reconstruction of $\mathbf{x}$ from $\mathbf{y}$ is possible even though $\mathbf{y}$ has fewer samples than a signal sampled at the Nyquist rate. It is this vector $\mathbf{y}$ that is actually stored in the local memory or wirelessly transmitted from the sensor.

In either case, once the samples of $\mathbf{y}$ have been passed to a receiving computer the original vector $\mathbf{x}$ needs to be reconstructed so that the original EEG signal can be inspected and used. This reconstruction is carried by solving the optimisation problem

$$\min_{\mathbf{s} \in \Re^N} \|\mathbf{s}\|_{l_1} \text{ subject to } \mathbf{y} = \mathbf{\Phi \Psi s} \tag{D.2}$$

$$\mathbf{r} = \mathbf{\Psi s} \tag{D.3}$$

where $\mathbf{\Psi}$ is a basis function in which the input $\mathbf{x}$ can be represented sparsely as $\mathbf{s}$ ($\mathbf{x} = \mathbf{\Psi s}$). That is, most entries in $\mathbf{s}$ should be zero, or near zero. $\mathbf{r}$ is then the reconstructed estimate of the original input $\mathbf{x}$.

The key criteria for successful compressive sensing are that $\mathbf{\Phi}$ and $\mathbf{\Psi}$ are incoherent and that $\mathbf{s}$ is a good sparse representation of the signal [6], [7]. The first condition is satisfied by drawing entries in $\mathbf{\Phi}$ from a random distribution. $\mathbf{\Psi}$ can then be any suitable choice for obtaining a sparse $\mathbf{s}$, and it can be changed during the reconstruction stage; knowledge of $\mathbf{\Psi}$ is not required during the initial sampling process. Depending on the precise choices for $\mathbf{\Phi}$, $\mathbf{\Psi}$, and the reconstruction minimisation method, many different compressive sensing implementations are possible with differing reconstruction performances and characteristics.

To minimise the sensor node power consumption, Bernoulli distribution ($p = 0.6$) is used with values of only 0 and 1 for $\mathbf{\Phi}$. This $\mathbf{\Phi}$ reduces the matrix multiplication Equation D.1 to an accumulation, greatly reducing the processing load. The fixed point output is accumulated into the MSP430 16 bit registers avoiding any potential overflow. On-the-fly generation of random numbers is an intensive process therefore to reduce the processing load non-adaptive compressive sensing [12] is used where a fixed sampling matrix is used for each frame, with the same matrix used for all EEG channels. This

matrix is generated *a priori* in MATLAB via the `randn` function and stored in the local flash memory of the MSP430.

For the signal reconstruction, which is carried out on a standard desktop computer, a Basis Pursuit optimisation procedure is used [22], [23] with a cubic B-spline dictionary [24], [25] for $\mathbf{\Psi}$. These settings correspond to the best reconstruction case for EEG signals as reported in [11].

D.2.5 Other system aspects

All nodes are set up assuming the collection of 16 channels of 200 Hz sampled EEG data. To provide a fair comparison platform, front-end signal conditioning and analogue-to-digital conversion affects all cases uniformly and so is not included in the analysis here which measures only the current consumption of the MSP430 and back-end unit. Suitable front-end systems for EEG applications include [3], [26] and a complete review of low power ADCs is in [27]. Instead of the front-end blocks, 16 channels of pre-recorded EEG data are stored in signed 16 bit fixed-point format (Q3.12) in the internal MSP430 flash memory. This is loaded a sample at a time into the MSP430 and once N samples from all 16 channels have been loaded the compressive sensing operation is performed. This produces M compressively sensed samples per channel which are passed out of the MSP430 to the transmitter/memory unit. The MSP430 can operate with a supply voltage anywhere between 1.8 V and 3.6 V. To minimise the power consumption, and to require only one regulator in each system, this voltage is set to match requirements of the back-end being used, as given in Section D.2.1.

D.3 Analysis methods

In this section, a number of different factors affecting the performance of compressive sensing in wearable sensor nodes are investigated. Firstly, the current consumption to quantify the power benefits of the on-sensor-node compression is investigated. The sensor node current is measured using a 500 MHz oscilloscope and a 10 Ω sense resistor placed in series between the power supply and the system. This allows the current to be measured as the system moves through the stages: MSP430 active and doing compressive sensing; MSP430 and back-end active and writing/sending out data; and the MSP430 and back-end idle. The total average current is then easily calculated.

Secondly, the reconstruction accuracy is investigated which demonstrates that the input physiological signal can be accurately recorded. A section of scalp EEG signals is used as the example application for this. While a number of studies have investigated the reconstruction performance of compressive sensing applied to EEG signals previously [11], [28]–[31] these do not take account of the compressive sensing hardware platform and its constraints. In particular, [11] presented a comprehensive evaluation of the reconstruc-

tion performance of multiple compressive sensing implementations for scalp EEG signals. It is not the intention here to repeat this analysis. Instead, it is noted that [11] used a fully MATLAB based, floating point, implementation of compressive sensing.

Thirdly it is verified here that a fixed point implementation suitable for the MSP430 is not substantially worse than the previous floating point one. The input EEG signals are stored offline as 16 bit EDF files [32] and are converted for a 16 bit fixed-point compressive sensing implementation. The results for the reconstruction error using the fixed point implementation are presented and compared with the fully MATLAB based implementation used in [11]. The reconstruction and error calculation procedures are the same for both, only the compressively sensed samples change. Using the fixed point toolbox MATLAB can accurately simulate both the floating and fixed point compressive sensing implementations and the performance is found offline, not by connecting sensor nodes to subjects. Although the MSP430 based compressive sensing assumes a full 16 channel EEG system, for compactness and visualisation here the reconstruction results for only seven channels are presented. These are Fp1, Fp2, T3, Cz, T4, O1, O2 all recorded at 200 Hz with an FCz reference. Thirty minutes of data from one subject are used.

The reconstruction performance is characterised through the RMS (Root-Mean-Square) error calculated using 10 s of reconstructed data, allowing maximum, minimum and median values over time to be found. The reconstruction performance was also characterised through the PRD (Percentage RMS Difference) error and similar trends were observed. All of these performance measures are calculated for a range of compression ratios (CR) and frame sizes (N) where

$$CR = \frac{M}{N}. \tag{D.4}$$

Lower CR values represent more data compression. This allows investigation of the optimal compressive sensing set up for each sensor node topology as a function of compression ratio CR and frame size N.

D.4 System performances

D.4.1 Reconstruction accuracy

Figure D.2 shows the reconstruction performance of the fixed point compressive sensing across the different regions of the head. It can be seen that reconstruction is possible, with the general result similar across all channels. As observed in [11], better performance is achieved along the midline (Cz here) and the performance is approximately symmetric across the midline except in T3/T4 where higher maximum errors are seen in T4. This occurs due to a period of asymmetric EMG artefact in the EEG (with more artefact present on T4 than T3) most likely due to chewing/jaw action. The test EEG data

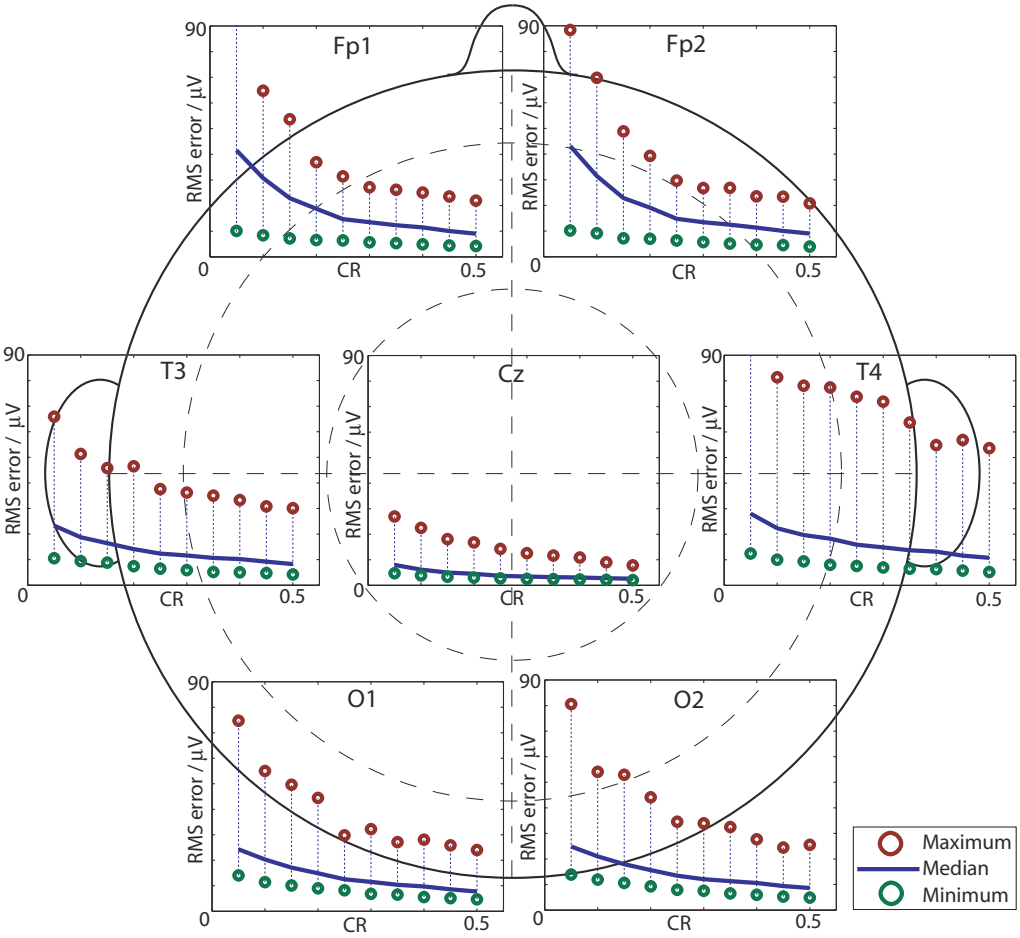
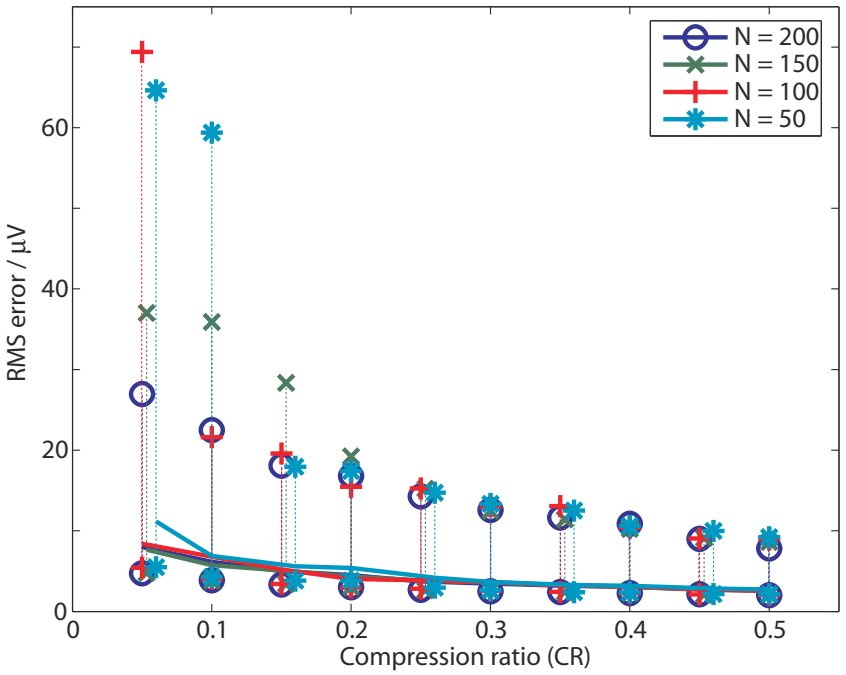


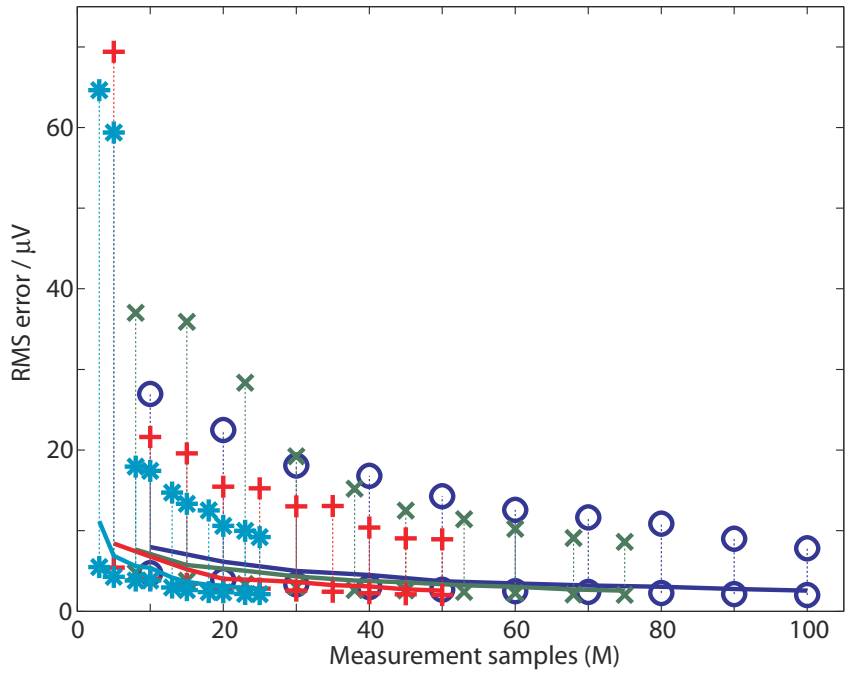
Figure D.2: *Reconstruction performance of the fixed point compressive sensing scheme as the compression ratio (CR) is varied. Values for the median, minimum and maximum RMS error values found from analysing 30 minutes of EEG data are shown. Results are arranged for each EEG channel as they are located on the head, and generated using frame size* $N = 200$.

is taken from out-patient, ambulatory EEG recordings in uncontrolled environments to reflect real-world sensor node operation. As such the results demonstrate the absolute performance bounds obtained during prolonged free-running EEG recording. In some use cases, if the entire free-running EEG is not wanted and artefact sections can be removed, these results are thus a pessimistic bound.

In all cases in Figure D.2 as the compression ratio is reduced the median, maximum and minimum of errors all get worse, and the impact of frame size N on this is shown in Figure D.3. Here, the larger $N = 200$ frame size performance is slightly better, but it is not a large effect. The reconstruction performance is principally only a function of the compression ratio.



(a)



(b)

Figure D.3: *Reconstruction performance of the fixed point compressive sensing scheme in channel Cz using four different frame sizes (N). Each vertical line plots the median, minimum and maximum RMS error found from analysing 30 minutes of EEG data. (a) against the compression ratio (CR); (b) against the number of compressed samples (M).*

D.4.2 Comparison to floating point

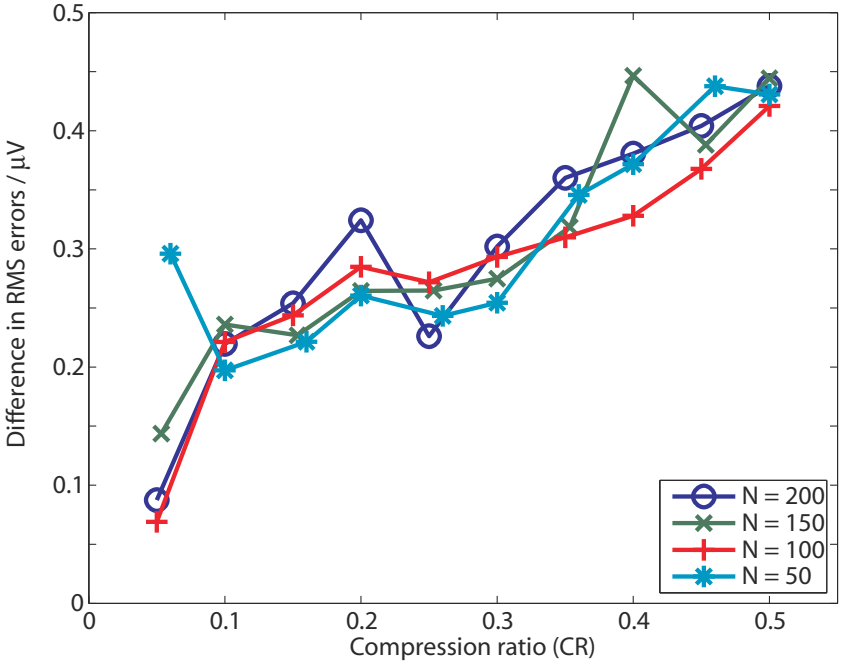
The differences in RMS error between the MATLAB simulated fixed point and floating point implementations are shown in Figure D.4 for two channels. Channel Cz, which achieved the best overall performance in Figure D.2, now achieves the worst performance of all of the analysed channels, although the differences are never large. In addition, in Cz there is a strong correlation between the compression ratio and how well the two implementations match: smaller compression ratios have less error between them. This effect is not seen in any of the other channels where the error is essentially independent of the compression ratio used. The origin of this is currently unknown but highlighted here as a systematic effect which may become significant to future researchers. It is noted that Cz is the closest electrode to the reference at FCz, but it is not clear why this would systematically impact the fixed-point vs floating-point operation of the compressive sensing.

Overall from Figure D.4 it is clear that the use of a fixed point implementation has no meaningful impact on the compressive sensing performance. As with the reconstruction accuracy results the frame size used has little consistent impact on reconstruction.

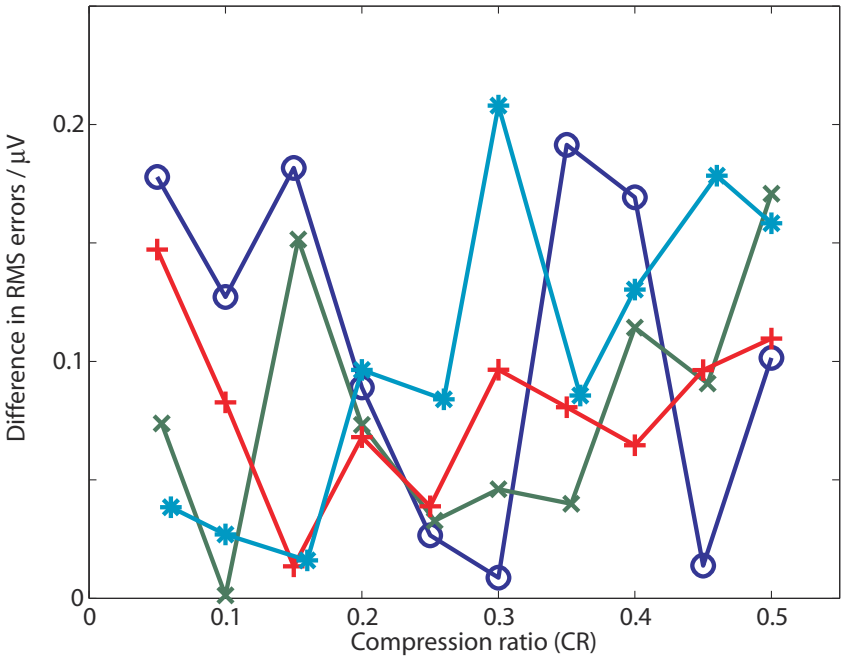
D.4.3 Peak current consumption

For classical systems with high average current draws, or which use physically large batteries, the average current consumption of the system is the only power consumption factor of interest. However many physically small batteries, in addition to having low total amounts of energy storage have low maximum peak draws [33]. For low power systems the peak current can thus be the limiting factor determining the battery size, not the energy density. Hence the measured peak currents from the entire system, and their durations, are given in Table D.1 for the four sensor nodes. The maximum current ever drawn by MSP430 alone is also listed. The maximum current drawn by the MSP430 alone (3.8 mA) is much smaller than that drawn when both the MSP430 and back-end are considered (which varies from 14–57 mA). The MSP430 can therefore be clocked at 16 MHz where it is more efficient per clock cycle to give a lower average power consumption. Ordinarily such operation might have been avoided if the increase in peak current was not acceptable. Similarly, both wireless back-ends have lower peak current requirements than the local memory units, highlighting a further potential advantage of moving to wireless sensor node topologies in low power, low size sensor nodes.

Typical timing diagrams for the four sensor nodes are shown in Figure D.5 illustrating very different current profiles. For the wireless transmitters distinct bursts are present corresponding to the transmission of individual packets. These are interspersed with the compressive sensing of the input signals and a small amount of set up time for the transmitter. In contrast for both the NAND memory and microSD card there is a broad increase in current as all of the data is written out with local peaks present during



(a)



(b)

Figure D.4: *Difference in the RMS reconstruction errors between the fixed point MSP430 suitable and fully floating point compressive sensing implementations for different frame sizes (N). (a) in channel Cz; (b) in channel O2.*

Table D.1: *Peak currents and their duration for the MSP430 compressive sensing and for the complete system with four different back-ends.*

Operation	Peak current draw / mA	Duration / μs
MSP430 compressive sensing (1 MHz clock)	0.32	–
MSP430 compressive sensing (16 MHz clock)	3.8	–
Nordic wireless transmitter	14	205
CC430 wireless transmitter	18	220
microSD card	57	250
NAND flash memory chip	19	230

the card access times. The microSD card takes considerably longer to write 50 samples, approximately 2.5 ms. The Nordic and CC430 transmitters require approximately 1.5 ms and 1.2 ms respectively while the NAND flash memory only 300 μs.

D.4.4 Average continuous current draw

Assuming that the battery used can provide the required peak current, the average continuous current consumption is the main power design factor. This current requirement is shown for the four nodes in Figure D.6 as a function of the compression ratio used. Horizontal lines show the power consumption of the system when no compression is present and all of the input data is sent to the back-end. For compressive sensing to be power beneficial the current consumption with compressive sensing turned on must be below these lines.

When no compression is present it is clearly beneficial to process the data in as large a frame size as possible, with larger frame sizes always requiring the least current. In line with the traditional motivation for the use of local memory sensor nodes, in the no compression case the average current of the Nordic system is approximately 1 mA, whereas for the NAND system it is only 0.2 mA, nearly an order of magnitude lower.

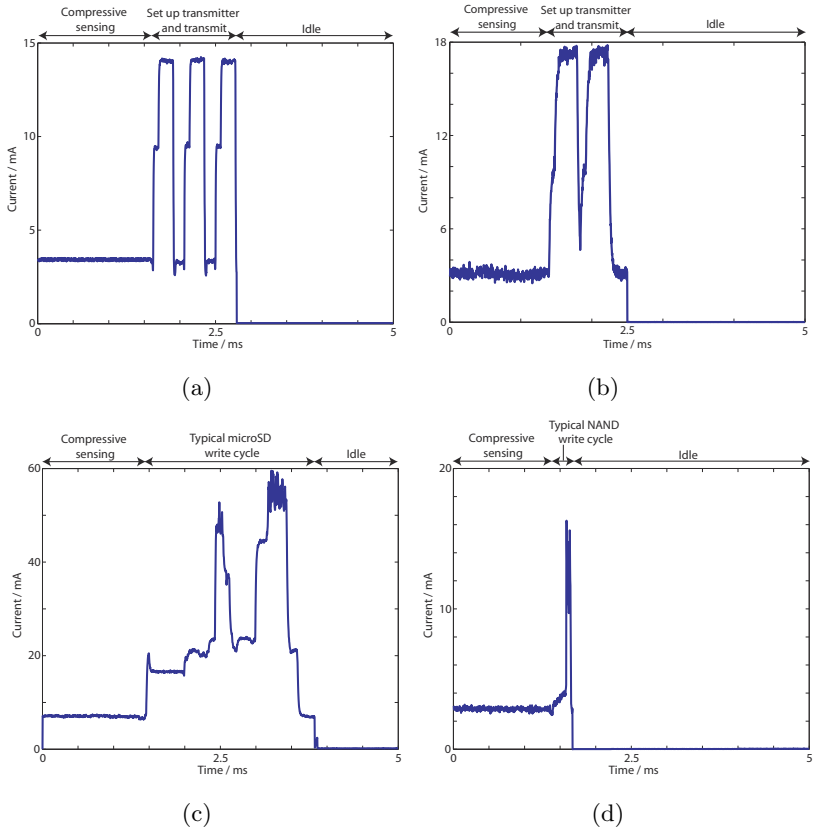


Figure D.5: *Current measured during a compress, transmit/store, idle cycle of the sensor node for $M = 50$, $N = 100$. Only a portion of the compressive sensing period is shown. Actual duration is 31 ms (for 16 channels) and the remainder of the cycle is spent in the idle state. (a) Nordic transmitter; (b) CC430 transmitter; (c) microSD card; (d) NAND flash memory chip.*

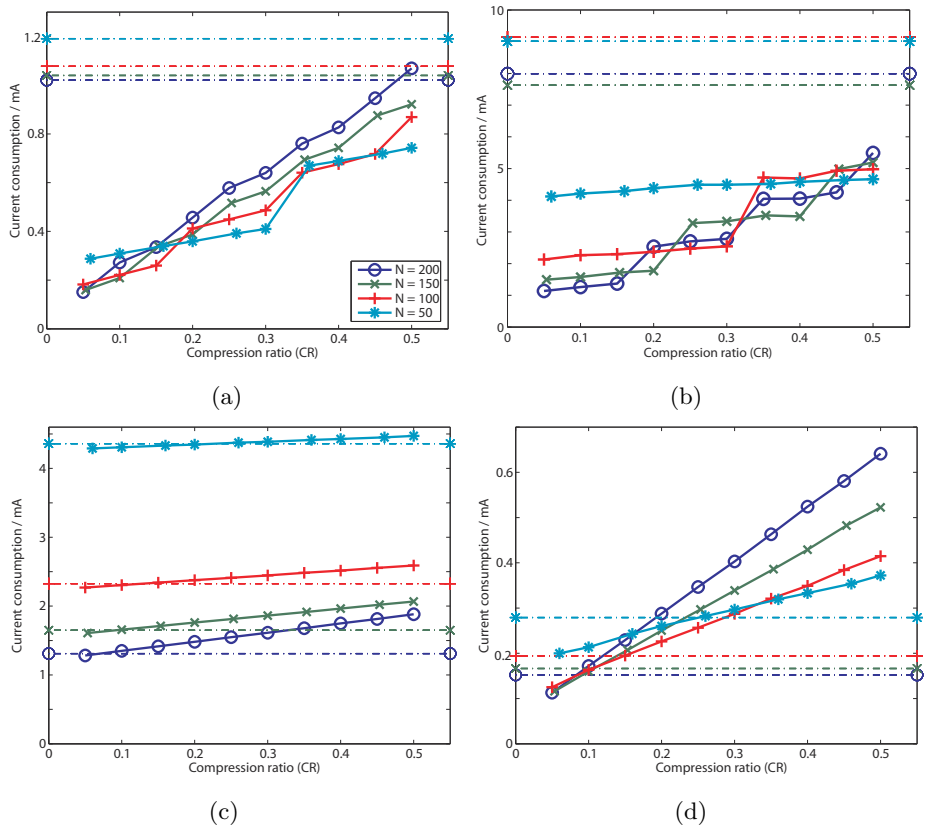


Figure D.6: *Average continuous current draw for the three sensor nodes. Horizontal lines show the current consumption in the direct sampling cases when no compressive sensing is used and all data is passed to the back-end. (a) Nordic transmitter; (b) CC430 transmitter; (c) microSD card; (d) NAND flash memory chip.*

However compressive sensing is much more beneficial for the wireless transmitter cases (Figure D.6(a) and Figure D.6(b)) than for the local memory cases. In particular for the microSD card, although there are cases where the use of compressive sensing is beneficial, the improvement in current consumption is in general small. This is because the setting up of the file system dominates the system power consumption, not the process of actually writing data. The frame size N used therefore dominates the system design: if N is large the file system is set up, addressed and accessed fewer times for writing out the same amount of data.

In both wireless systems, compressive sensing decreases the system current requirements significantly, and at the lowest compression ratios it can be brought to the same power ranges as the local memory topologies. Moreover, with the wireless transmitters compressive sensing is power beneficial for essentially all of the frame sizes and compression ratios used. In all cases the lowest absolute power consumption is achieved by using larger frame sizes. However the system needs to operate at the lowest compression ratios to achieve this and the increase in the reconstruction error (Figure D.2) must be tolerated. For more practical compression ratios in the 0.2–0.5 range, using smaller frame sizes is a more appropriate design strategy. For the wireless transmitter at these compression ratios it is the smaller frames that give the lowest current consumption and correspondingly the largest percentage improvement in current consumption.

D.4.5 Relative current consumptions

The motivation for on-sensor-node data compression is due to the presence of a high power transmitter/storage stage. This dominates the power consumption and thus expending some power on data compression to give less data to transmit can remove this dominance and lead to overall power savings [1]–[3]. To evaluate the impact of compressive sensing on this Figure D.7 breaks down the measured average current values from Figure D.6 into the three main stages of operation: idle, compressive sensing, and transmission/storage of the data; and shows the percentage of the current used during each stage. For clarity only results for two frame sizes, $N = 50$ and $N = 200$, are shown.

To minimise power consumption the sensor nodes have been designed to spend as much time as possible in the idle mode and this results in a considerable amount (up to 20%) of the total system current being used in this mode. Indeed the sensor node with the highest absolute consumption (the CC430 node) is the one which spends the minimum amount of power in idle. For the Nordic transmitter (Figure D.7(a)) at $N = 200$ the transmission and the compressive sensing both use approximately equal percentages ($\sim$50%) of the total current at all compression ratios and so the transmitter dominance has been reduced compared to the compression costs. Nevertheless lower absolute consumptions are achieved at lower compression ratios due to the substantial increases in idle time possible as less data passes through the system.

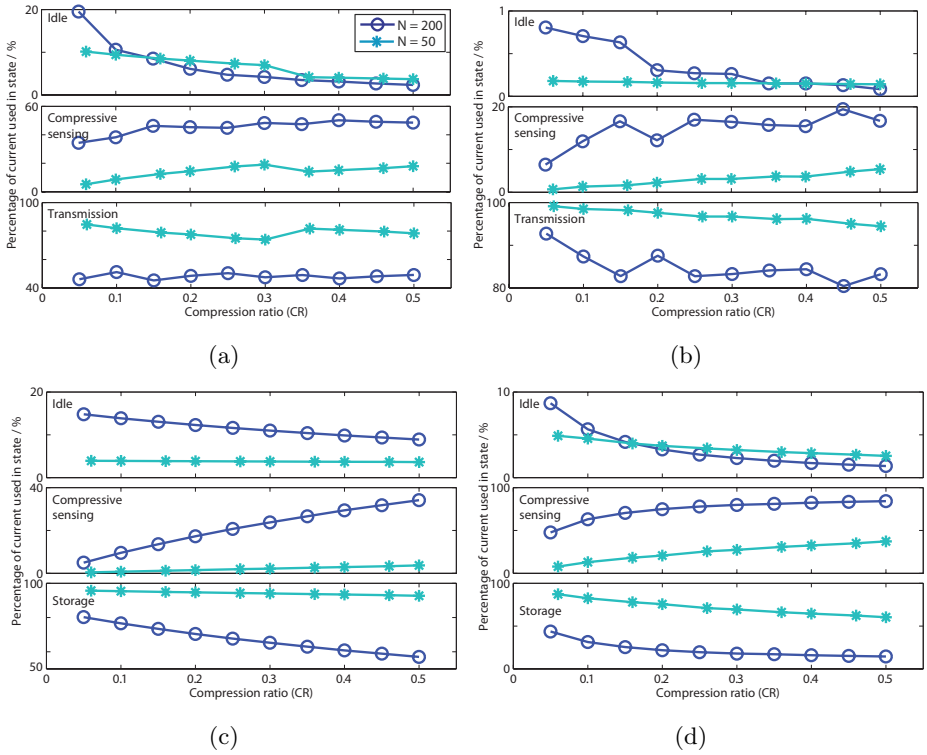


Figure D.7: *Break-down of the current consumptions from Figure D.6 showing the percentage of the total consumption used in each of the main stages of operation. (a) Nordic transmitter; (b) CC430 transmitter; (c) microSD card; (d) NAND flash memory chip.*

In all of the cases in Figure D.7 there is a direct trade-off with the frame size (N) used: large frames push towards using proportionally more current on compression, while smaller frame sizes push towards transmission/storage. Minimum power realisations of compressive sensing can thus use this to optimise the system power consumption without substantial impacts on the reconstruction performance (Figure D.2). This minimum power realisation is best achieved at low compression ratios. However for the local memory sensor nodes, and the microSD card in particular, the percentage of current used for storage actually increases as more data reduction is provided.

D.5 Discussion

Low power data compression is an essential part of wearable sensor nodes for monitoring the human body. Compressive sensing is an emerging technique for use in these and it has shown utility in a number of physiological sensing applications [8]–[13]. Recent results have also suggested that compressive sensing applied within the sensor node could relax the noise specification for the front-end amplifier [5], provide a level of encryption [30], or potentially allow the number of bits in the analogue-to-digital converter to be reduced [34].

However compressive sensing is a lossy compression technique and although the requirement for minimum power implementations is well established, its performance is often only assessed in terms of reconstruction accuracy [13]. [35] presented a compressive sensing implementation using a MSP430 microcontroller and produced a complete working wireless ECG sensor. [5] presented a fully custom implementation of the digital multiplier required for the sampling process and provided the first fully on-chip implementation of sensor node compressive sensing. These however focused on the power minimisation of the compressive sensing stage only. The motivation for on-sensor-node data compression is due to the presence of a high power transmitter/storage stage and its power changes under different compressive sensing arrangements must also be accounted for. This chapter has therefore investigated compressive sensing in terms of both reconstruction performance and power consumption taking into account different back-end technologies. Four different sensor nodes have been presented comprising of both wireless and local memory topologies.

Local memory sensor nodes are often used as, without on-node data compression, they have lower average current draws than wireless nodes. The NAND memory based system used here required nearly an order of magnitude less current than the wireless transmitter systems. However wireless data collection is highly advantageous as it allows real-time access to the data. The results showed that both wireless systems benefited substantially from the use of compressive sensing and in the Nordic transmitter case it successfully brought down the average current consumption to a comparable level to the NAND system.

Therefore, assuming that the reconstruction error of the compressive sensing can be tolerated for the given application, there is now no intrinsic reason to prefer local storage based sensor nodes from a current consumption point of view. Indeed, once the average consumption of the wireless system is comparable to the NAND, the overall system can benefit from the lower supply voltage required by the wireless transmitter and the lower peak current requirement. Both of these allow the use of physically smaller batteries to power the overall system. This is made possible because the raw EEG data rate is only 25.6 kbps and is low compared to the 2 Mbps data rate of the wireless transmitter and so it can be significantly duty cycled. Although 16 channel EEG data was used as the example application, this comparatively low data rate will also apply to many other physiological parameters such as the EOG, EMG and ECG.

The reconstruction error was found to be independent of whether a fixed or floating point compressive sensing implementation was used, and also independent of the frame size N. As a result, considering the signal processing perspective only these are free choices. When combined with the current consumptions presented here however the power consumption is strongly dependent on the frame size and the back-end technology used. Indeed the frame size controls the power balance present in the system: larger frames shifted power towards compression while smaller sizes shifted power towards transmission. Far from a free choice, this is therefore a critical parameter for optimising the compressive sensing power consumption, and compressive sensing schemes for sensor nodes must be evaluated in terms of both reconstruction and power performance. This power trade-off has been illustrated for four different types of back-end.

Inevitably there are other practical factors in addition to the current consumption to consider when selecting between the use of wireless sensor nodes and local memory sensor nodes. For example, the wireless sensor nodes use a minimum latency transmission scheme where data is sent as soon as possible to keep the system real-time. It is possible to potentially reduce power consumption further by choosing a transmission scheme where data is accumulated for some time and then sent out in larger packets. This, and other techniques for transmitter power optimisation are discussed in [36]. It is also possible to use a memory card in series with a wireless transmitter to buffer the data before transmission if latency can be tolerated in the node. Further, the regulations governing transmit power and interference are much more stringent for wireless nodes, and the impacts of bit errors and missing packets need to be accounted for. For local memory systems, the microSD node required a FAT16 file system and the set up of this dominated the power consumption, not the process of actually writing out data. The NAND memory node did not use this to save power, but as a result the memory cannot be directly read by a host PC without requiring additional software, impacting the ease of use. Finally the results do not control for the semiconductor process size used in the microSD card and NAND memories. While the general form of these results will be unchanged, this

parameter will affect the precise currents drawn by these memory units, with smaller process nodes requiring less current.

Nevertheless the results give a practical overview of the performance of state-of-the-art sensor node systems and will be highly informative for others selecting between the different sensor node topologies. For optimising the compression implementation, for practical compression ratios in the 0.2–0.5 range smaller frame sizes offer lower current consumption, and a greater percentage improvements in the current consumption, although the lowest absolute current consumptions were achieved with larger frame sizes. The reconstruction error was independent of this and so the compressive sensing design must be governed by the back-end considerations; it is not possible to consider the compression design and sensor topology design in isolation.

References

[1] B. Gyselinckx, C. Van Hoof, J. Ryckaert, R. F. Yazicioglu, P. Fiorini, and V. Leonov, "Human++: autonomous wireless sensors for body area networks," in *IEEE CICC*, San Jose, September 2005.

[2] A. J. Casson, D. C. Yates, S. J. Smith, J. S. Duncan, and E. Rodriguez-Villegas, "Wearable electroencephalography," *IEEE Eng. Med. Biol. Mag.*, vol. 29, no. 3, pp. 44–56, 2010.

[3] N. Verma, A. Shoeb, J. Bohorquez, J. Dawson, J. Guttag, and A. P. Chandrakasan, "A micro-power EEG acquisition SoC with integrated feature extraction processor for a chronic seizure detection system," *IEEE J. Solid-State Circuits*, vol. 45, no. 4, pp. 804–816, 2010.

[4] A. M. Abdulghani, A. J. Casson, and E. Rodriguez-Villegas, "Quantifying the feasibility of compressive sensing in portable electroencephalography systems," in *HCI international*, San Diego, July 2009.

[5] F. Chen, A. P. Chandrakasan, and V. M. Stojanovic, "Design and analysis of a hardware-efficient compressed sensing architecture for data compression in wireless sensors," *IEEE J. Solid-State Circuits*, vol. 47, no. 3, pp. 744–756, 2012.

[6] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inform. Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[7] E. J. Candes and M. B. Wakin, "An introduction to compressive sampling," *IEEE Signal Processing Mag.*, vol. 25, no. 2, pp. 21–30, 2008.

[8] U. Gamper, P. Boesiger, and S. Kozerke, "Compressed sensing in dynamic MRI," *Magn. Reson. Med.*, vol. 59, no. 2, pp. 365–373, 2008.

[9] T. V. Sreenivas and W. B. Kleijn, "Compressive sensing for sparsely excited speech signals," in *IEEE ICASSP*, Taipei, April 2009.

[10] Y. Zhang, S. Mei, Q. Chen, and Z. Chen, "A novel image/video coding method based on compressive sensing theory," in *IEEE ICASSP*, Las Vagas, April 2008.

[11] A. M. Abdulghani, A. J. Casson, and E. Rodriguez-Villegas, "Compressive sensing scalp EEG signals: implementations and practical performance," *Med. Biol. Eng. Comput.*, vol. 50, no. 11, pp. 1137–1145, 2012.

[12] A. M. R. Dixon, E. G. Allstot, D. Gangopadhyay, and D. J. Allstot, "Compressed sensing system considerations for ECG and EMG wireless biosensors," *IEEE Trans. Biomed. Circuits Syst.*, vol. 6, no. 2, pp. 156–166, 2012.

[13] Z. Zhang, T.-P. Jung, S. Makeig, and B. D. Rao, "Compressed sensing of EEG for wireless telemonitoring with low energy consumption and inexpensive hardware," *IEEE Trans. Biomed. Eng.*, vol. 60, no. 1, pp. 221–224, 2013.

[14] Texas Instruments. (2012) MSP430 ultra-low power 16-bit microcontrollers. [Online]. Available: http://www.msp430.com/.

[15] H. Dubois-Ferrière, L. Fabre, R. Meier, and P. Metrailler, "TinyNode: a comprehensive platform for wireless sensor network applications," in *Proc. IPSN*, Nashville, April 2006.

[16] M. Johnson, M. Healy, P. van de Ven, M. Hayes, J. Nelson, T. Newe, and E. Lewis, "A comparative review of wireless sensor network mote technologies," in *IEEE Sensors*, Christchurch, October 2009.

[17] A. Burns, B. R. Greene, M. J. McGrath, T. J. O'Shea, B. Kuris, S. M. Ayer, F. Stroiescu, and V. Cionca, "Shimmer^{TM} — a wireless sensor platform for noninvasive biomedical research," *IEEE Sensors J.*, vol. 10, no. 9, pp. 1527–1534, 2010.

[18] Nordic Semiconductor. (2012) Home page. [Online]. Available: http://www.nordicsemi.com/.

[19] Texas Instruments. (2013) CC430 RF SOC series. [Online]. Available: http://www.ti.com/lsds/ti/microcontroller/16-bit_msp430/rf_soc/overview.page.

[20] FatFs Generic FAT File System Module. (2012) Home page. [Online]. Available: http://elm-chan.org/fsw/ff/00index_e.html.

[21] Hynix. (2012) Home page. [Online]. Available: http://www.hynix.com/.

[22] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Rev.*, vol. 43, no. 1, pp. 129–159, 2001.

[23] SparseLab. (2012) Home page. [Online]. Available: http://sparselab.stanford.edu/.

[24] M. Andrle and L. Rebollo-Neira, "Cardinal B-spline dictionaries on a compact interval," *Appl. Comput. Harmon. Anal.*, vol. 18, no. 3, pp. 336–346, 2005.

[25] Highly nonlinear approximations for sparse signal representation. (2012) Home page. [Online]. Available: http://www.nonlinear-approx.info/.

[26] J. Xu, R. Yazicioglu, P. Harpe, K. Makinwa, and C. V. Hoof, "A 200μw eight-channel acquisition ASIC for ambulatory EEG systems," in *IEEE ISSCC*, San Francisco, February 2011.

[27] B. Murmann. (2013) ADC Performance Survey 1997-2013. [Online]. Available: http://www.stanford.edu/~murmann/adcsurvey.html.

[28] S. Aviyente, "Compressed sensing framework for EEG compression," in *IEEE/SP SSP*, Madison, August 2007.

[29] S. Senay, L. F. Chaparro, M. Sun, and R. J. Sclabassi, "Compressive sensing and random filtering of EEG signals using slepian basis," in *EUSIPC*, Lausanne, August 2008.

[30] A. M. Abdulghani and E. Rodriguez-Villegas, "Compressive sensing: From "compressing while sampling" to "compressing and securing while sampling"," in *IEEE EMBC*, Buenos Aires, September 2010.

[31] Q. Hao and F. Hu, "A compressive eletroencephalography EEG sensor design," in *IEEE Sensors*, Kona, September 2010.

[32] B. Kemp and J. Olivan, "European data format 'plus' (EDF+), an EDF alike standard format for the exchange of physiological data," *Clin. Neurophysiol.*, vol. 114, no. 9, pp. 1755–1761, 2003.

[33] G. Chen and E. Rodriguez-Villegas, "System-level design trade-offs for truly wearable wireless medical devices," in *IEEE EMBC*, Buenos Aires, September 2010.

[34] Igor Carron. (30th January 2010) CS: Q&A with Esther Rodriguez-Villegas on a compressive sensing EEG. [Online]. Available: http://nuit-blanche.blogspot.com/.

[35] H. Mamaghanian, N. Khaled, D. Atienza, and P. Vandergheynst, "Compressed sensing for real-time energy-efficient ECG compression on wireless body sensor nodes," *IEEE Trans. Biomed. Eng.*, vol. 58, no. 9, pp. 2456–2466, 2011.

[36] A. Y. Wang, "Low power RF transceiver modelling and design for wireless microsensor networks," PhD thesis, Massachusetts Institute of Technology, 2005.