8-2010

A Computational Framework to Support the Automated Analysis of Routine Electroencephalographic Data

William Pressly, jr.
Clemson University, wbspresslyjr@gmail.com

A Computational Framework to Support the Automated Analysis of Routine Electroencephalographic Data

---

A Dissertation
Presented to
the Graduate School of
Clemson University

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
Computer Science

---

by
William Buck Sparkman Pressly, Jr.
August 2010

Accepted by:
Brian C. Dean, Ph.D., Committee Chair
James M. Westall, Ph.D.
James J. Martin, Ph.D.
Feng Luo, Ph.D.
Jonathan J. Halford, M.D.

Abstract

Epilepsy is a condition in which a patient has multiple unprovoked seizures which are not precipitated by another medical condition. It is a common neurological disorder that afflicts 1% of the population of the US, and is sometimes hard to diagnose if seizures are infrequent. Routine Electroencephalography (rEEG), where the electrical potentials of the brain are recorded on the scalp of a patient, is one of the main tools for diagnosing because rEEG can reveal indicators of epilepsy when patients are in a non-seizure state. Interpretation of rEEG is difficult and studies have shown that 20-30% of patients at specialized epilepsy centers are misdiagnosed [18, 73]. An improved ability to interpret rEEG could decrease the misdiagnosis rate of epilepsy.

The difficulty in diagnosing epilepsy from rEEG stems from the large quantity, low signal to noise ratio (SNR), and variability of the data. A usual point of error for a clinician interpreting rEEG data is the misinterpretation of PEEs (paroxysmal EEG events) – short bursts of electrical activity of high amplitude relative to the surrounding signals that have a duration of approximately .1 to 2 seconds [4]. Clinical interpretation of PEEs could be improved with the development of an automated system to detect and classify PEE activity in an rEEG dataset. Systems that have attempted to automatically classify PEEs in the past have had varying degrees of success [47]. These efforts have been hampered to a large extent by the absence of a "gold standard" data set that EEG researchers could use.

In this work we present a distributed, web-based collaborative system for collecting and creating a "gold standard" dataset for the purpose of evaluating spike detection software. We hope to advance spike detection research by creating a performance standard that facilitates comparisons between approaches of disparate research groups. Further, this work endeavors to create a new, high performance parallel implementation of ICA (independent component analysis), a potential preprocessing step for PEE classification. We also demonstrate tools for visualization and analysis

to support the initial phases of spike detection research.

These tools will first help to develop a standardized rEEG dataset of expert EEG interpreter opinion with which automated analysis can be trained and tested. Secondly, it will attempt to create a new framework for interdisciplinary research that will help improve our understanding of PEEs in rEEG. These improvements could ultimately advance the nuanced art of rEEG interpretation and decrease the misdiagnosis rate that leads to patients suffering inappropriate treatment.

Dedication

This work is dedicated to my wife, Amber. Her love, support, and patience enabled me to undertake and complete this work.

Acknowledgements

I would like to acknowledge the people listed below. This research effort would not be possible without their efforts:

1. Brian C. Dean, my advisor. He is a tremendously brilliant and patient individual who I am deeply indebted to for such a rewarding time in graduate school.

2. Jon Halford, the "brains" behind our research project, for giving me the opportunity to work on such an interesting and profound project (and paying my bills).

3. My committee, for their guidance and insight in this research project.

4. Zach Jones and Jay Steele, for all of the assistance and companionship they provided.

5. The technical and administrative staff at the School of Computing, for all of the work they did on my behalf.

6. My Mom and Dad, for always believing in me despite the times that they maybe shouldn't have.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Epilepsy is a condition in which a patient has multiple unprovoked seizures that are not precipitated by another medical condition. It is a common neurological disorder that afflicts 1-2% of the population of the US, and is sometimes hard to diagnose if seizures are infrequent. Routine Electroencephalography (rEEG), in which the electrical potentials of the brain are recorded on the scalp of a patient, is one of the main tools for diagnosing epilepsy because rEEG can reveal indicators of epilepsy when patients are in a non-seizure state. Approximately 20-30% of patients at specialized epilepsy centers are misdiagnosed [18, 73]. This is because the subtle indicators of epilepsy are not always present on EEG and it may take several EEGs to detect the evidence of epilepsy. This is also because EEG is difficult to interpret and sometime EEGs are interpreted as indicating epilepsy when they do not. An improved ability to interpret rEEG could decrease the misdiagnosis rate of epilepsy.

The difficulty in diagnosing epilepsy from rEEG stems from the large quantity, low signal to noise ratio (SNR), and high variability of the data. A usual point of error for a clinician interpreting rEEG data is the misinterpretation of PEEs (paroxysmal EEG events) – short bursts of electrical activity of higher amplitude relative to the surrounding baseline signals that have a duration of approximately .1 to 2 seconds [4]. PEEs which are due to normal brain activity are misinterpreted as indicating that the patient has epilepsy. To illustrate the challenging nature of PEE classification, consider the screen capture from clinical rEEG software in Figure 1.1 as it contains an *epileptiform PEE* (indicator of epilepsy). To the layperson looking at this data, his or her first inclination might be that the waveforms of large amplitude on the extreme right of the image in the channels labeled

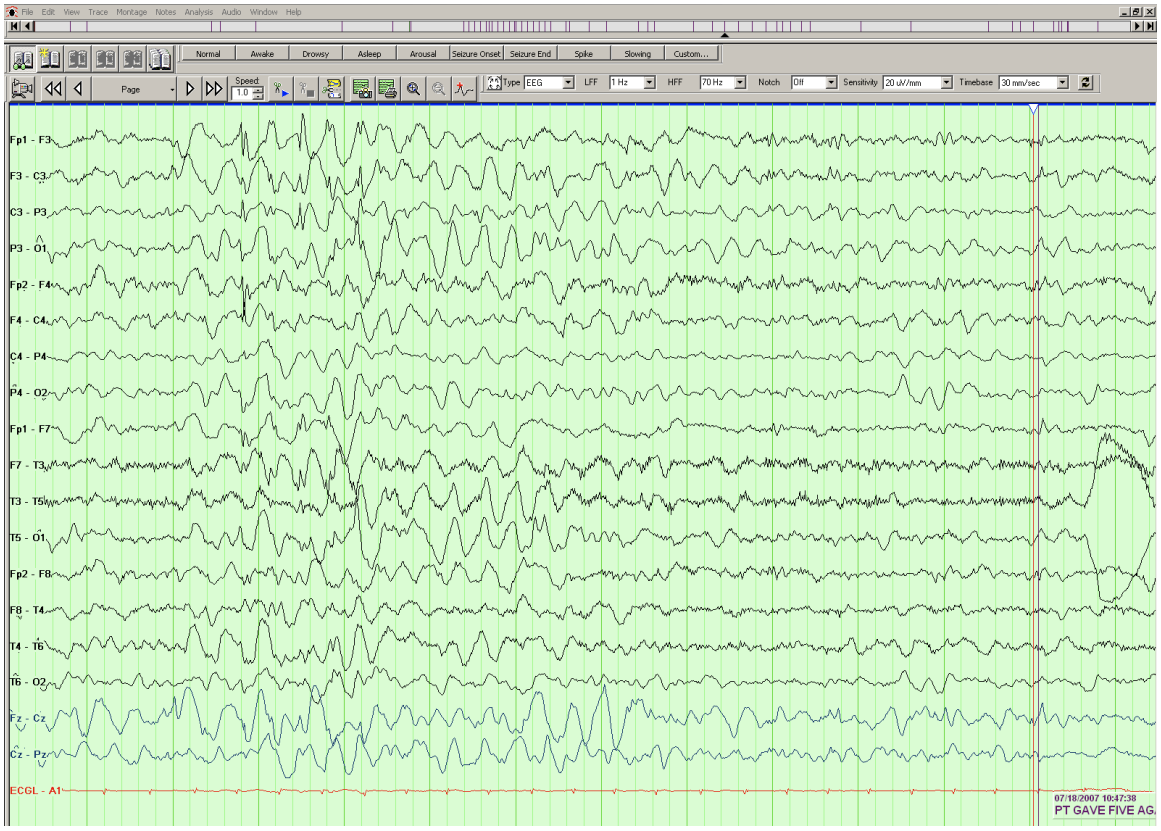


Figure 1.1: This is a screen capture from a clinical rEEG system called XLTek Neuroworks. This data contains a transient waveform that is positively indicative of epilepsy.

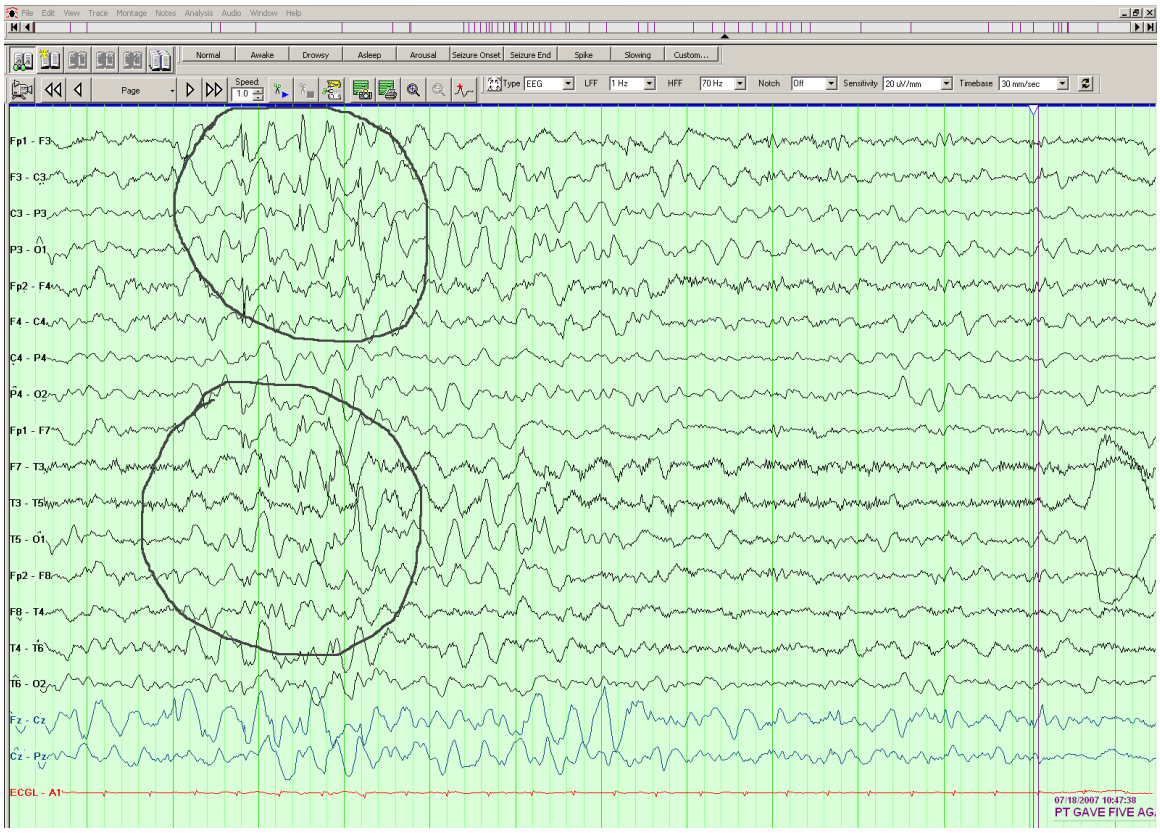


Figure 1.2: This is the same screen capture as Figure 1.1 with circles around the epileptiform PEEs.

T3-T5 and T5-O1 might be the abnormal epileptiform PEE. It turns out that this PEE is likely the first one that an experienced clinician would dismiss as an *artifact*, likely an eye blink or some voluntary muscle movement. The actual epileptiform PEE is circled in Figure 1.2. Here is the reasoning, from Dr. Jonathan Halford, our collaborator on this project from the Medical University of South Carolina: "There is a burst of spike-wave activity lasting for 2-3 seconds. Each spike (most visible on F3-C3, C3-P3, and F7-T3 channels) is followed by a slow-wave. There are some slow waves that do not have spikes associated with them and this is confusing. Also, due to the scalp distribution of the burst, the same burst is seen in two different locations on the page even though it is just one location on the scalp."

This exercise well illustrates the nuanced and subtle characteristics of interpretation of rEEG and why it is perhaps not surprising that there is significant disagreement between diagnoses of even experienced rEEG interpreters [84, 79]. There are many factors that make rEEG data challenging

to interpret: the electrical signals of the brain (1) are on the order of microvolts, and (2) are subject to many noise sources in the clinical setting since they are recorded on the scalp, and (3) are subject to changes in morphology due to variation in electrode placement and grounding. In addition, the number of original source signals of the brain are very complex and contain a wide variation of waveform morphologies. The nuanced art of interpreting the subtle clues in rEEG that could be indicative of epilepsy makes the problem extremely challenging not only for humans, but also machines. Systems that have attempted to automatically classify PEEs in the past have had varying degrees of success [47]. These efforts have been hampered to a large extent by the absence of a "gold standard" dataset that EEG researchers could use to train and test automated rEEG interpretation algorithms. Effective machine-assisted interpretation of rEEG remains an elusive, yet highly desirable goal. **It is the objective of this research to develop a framework of computational tools that support this long-term goal of machine analysis of rEEG, ultimately leading to a decrease in the misdiagnosis rate of epilepsy and other neurological disorders due to misreading of rEEG studies.**

In this work we present a distributed, web-based collaborative system for collecting and creating a correctly annotated dataset of expert opinion for the purpose of evaluating spike detection software. We hope that this will further spike detection research by creating a standard for correctness of results and improve the ability to compare the algorithms of disparate research groups. Further, this work endeavors to create a new, high performance parallel implementation of ICA (independent component analysis), a technique that shows promise as a useful preprocessing step for PEE classification, but that currently involves excessive computational demands. We have also developed tools for visualization and analysis to support the initial phases of spike detection research. These tools will first help to develop a standardized rEEG dataset of expert EEG interpreter opinion with which automated spike detection algorithms can be trained and tested. Secondly, it will attempt to create a new framework for interdisciplinary research that will help improve our understanding of PEEs in rEEG.

In the remainder of this Chapter, we will explore methods for rEEG clinical data acquisition, use in practice, channel display and nomenclature, and methods of visualization. Also, as a major focus of this work will be the creation of gold standard dataset of rEEG for spike detection research, we will frame this work in the context of a larger preceding project that focused on ECG (Electrocardiography – monitoring the electrical signals of the heart) which set precedent in estab-

lishing a standardized clinical biomedical dataset which has been used successfully for ECG research for almost 30 years [85].

1.1 Routine EEG (rEEG): An Introduction

rEEG is a clinical diagnostic tool that is commonly employed to help diagnose neurological disorders. It involves the recording of electrical potential over time from a number of electrodes (typically 22-25, but there can be many more in higher resolution systems) placed on the scalp in a specific topographic arrangement. The electrodes are attached to differential amplifiers and are amplified to about 60-100 dB of gain (about 1,000-100,000 times). As is typical, both signal and noise are amplified, so some basic filtering is used such as a notch filter (to filter out the electrical power line artifact) and both high and low pass filters for decreasing the amplitude of movement and muscle artifacts. This amplified signal is acquired at a sampling rate of 256-512Hz, for a typical recording length of 20-30 minutes. The number and placement of the electrodes and the resulting nomenclature of the electrodes is important for understanding of the rEEG signal as it relates to the spatial positioning of the electrical source signals in the brain. The standard placement for clinical rEEG both nationally and internationally is the International 10-20 System [54] which is illustrated in Figure 1.3. In this system, electrodes are placed along arcs radiating from the center of the scalp out to the temple in intervals of 20% and 10% of the length of certain scalp measurements, defining a grid over the entire scalp. Each electrode name is composed of a letter (or upper-case and lower-case letter combination) and a number, for instance: Fp2, O2, etc. The letter combination represents a part of the brain: Frontral pole (Fp), Frontal (F), Central (C), Parietal (P), Occipital (O), and Auricular (A, the ear electrode). The number that follows indicates two properties of the channel: the parity indicates which side of the brain the electrode is over (even numbers are on the right and odd numbers are on the left), and the magnitude of the number indicates distance from the centerline [44].

Since the rEEG signal is always acquired as a potential difference between two electrodes, there are many potential formats for viewing rEEG electrode data which are called *montages*. A montage is a collection of channels of a specific spatial pattern. The two most common types of montages are bipolar and referential. In bipolar montages adjacent channels are compared. An example of a bipolar montage is the "Bipolar AP Typical" montage (see upper left of Figure 1.4),

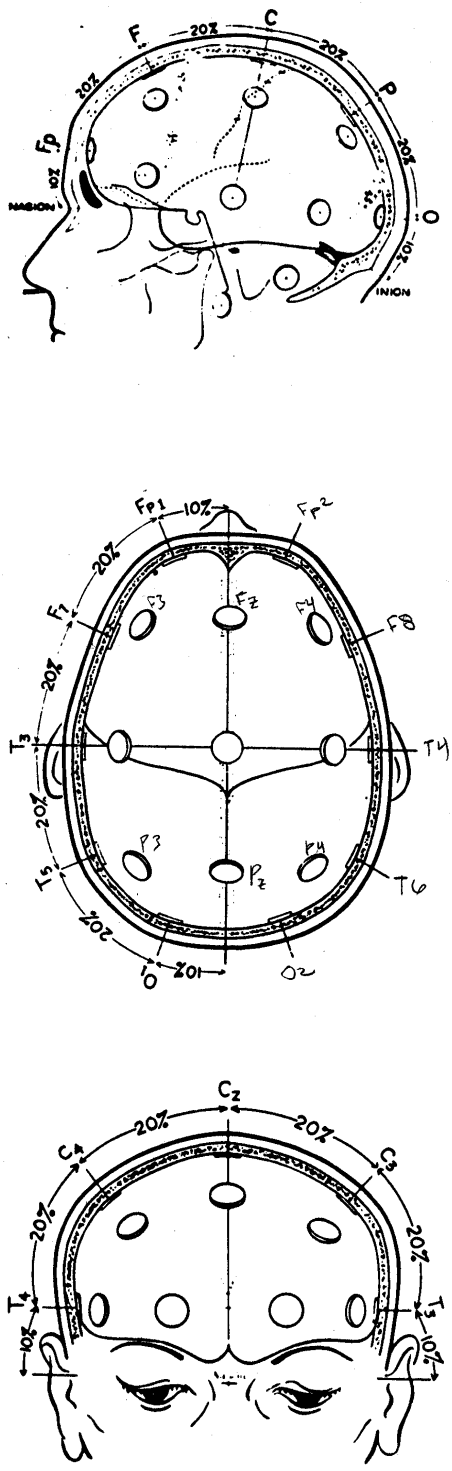


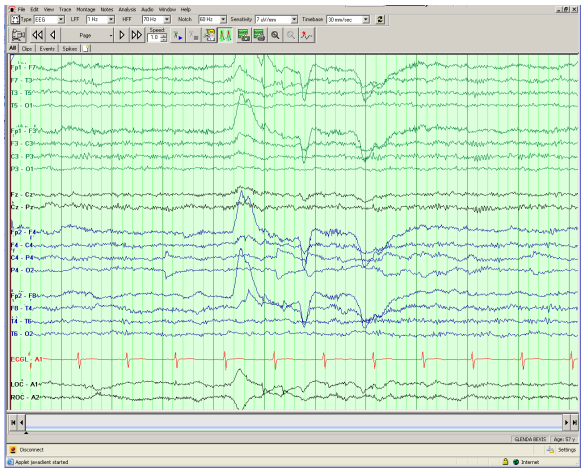
Figure 1.3: Positions of rEEG electrodes according to the Note that the first letter of the channel name indicates the region of the brain it is over, the even or oddness of the number of the channel indicates if the channel is on the right or left of the scalp respectively, and the magnitude of the number indicates distance from the center line. These images are reproduced from [44].

typically referred to as the "double banana" for the two banana shapes the montage traces over each half of the brain. For the sake of comparison, the "Bipolar Transverse" montage (upper right in Figure 1.4) is much like the "double banana" rotated 90 degrees from top view. A referential montage is one where all of the channels are differentially compared against a single common channel. Examples include the "Referential Cz" montage (all electrodes compared to the electrode on the center of the scalp, lower right in Figure 1.4) and the "Referential Ipsi-Ear" montage (lower left in Figure 1.4).

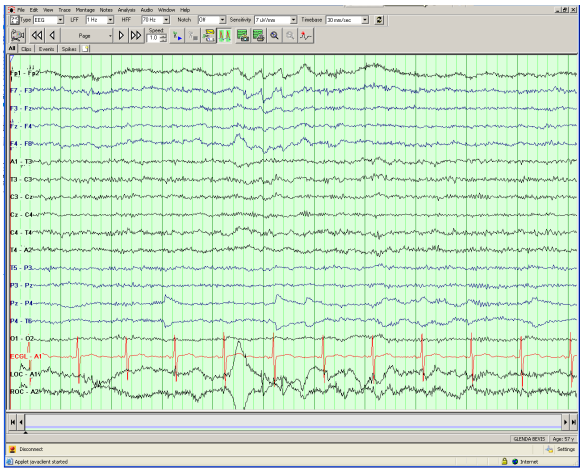
1.2 The Precedent of ECG

Electrocardiography (ECG) is the recording of the electrical activity of the heart through sensors placed on the skin. By contrast, the electrical signals recorded by ECG are approximately 1000 times stronger than that of EEG and less easily obscured by artifact [53]. ECG has a rich history as it was the first signal from the human body to be processed by a machine, around the beginning of the 20th century. In 1924, Einthoven received the Nobel Prize in Medicine for his work interpreting ECG features of cardiovascular disorders. Over time, ECG features became well understood and were decomposed into a feature vector (see Figure 1.5). Automated ECG interpretation software was developed and has been shown to be as good at interpreting the signal as human interpreters in many cases [39].

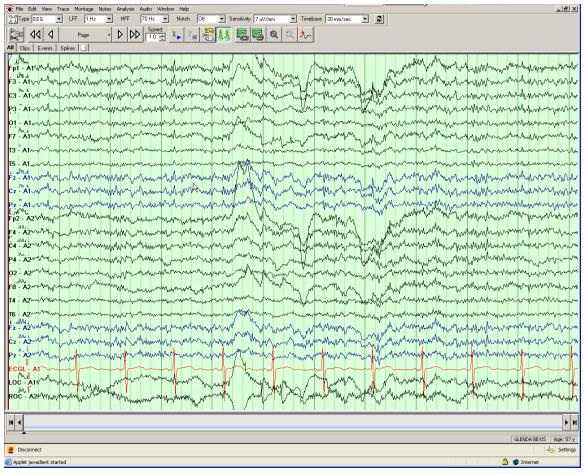
Automated ECG interpretation software was developed in Europe in the late 20th century. In 1976, the European Community Committee for Monitoring the Seriously Ill recommended that researchers in the field of ECG combine their efforts and create a standardized clinical ECG dataset. This project was called "Common Standards for Quantitative Electrocardiography" (CSE) and was launched in 1978 [85]. It progressed in two phases over a the better part of a decade. In the first phase, which lasted 3 years, a database of 500 ECGs was collected and reviewed by a panel of five cardiologists in an iterative four round process. This database was divided into training and testing sets [82] and 19 computer algorithms were compared for their ability to measure ECG waves [81]. The object of the second phase, which began in 1985, was to study the performance of ECG interpretation. Another round of data collection resulted in a database of 1220 ECGs from adult patients in 5 different European centers. This data was reviewed by nine cardiologists who correctly classified 96% of the studies compared to fifteen computer programs which correctly



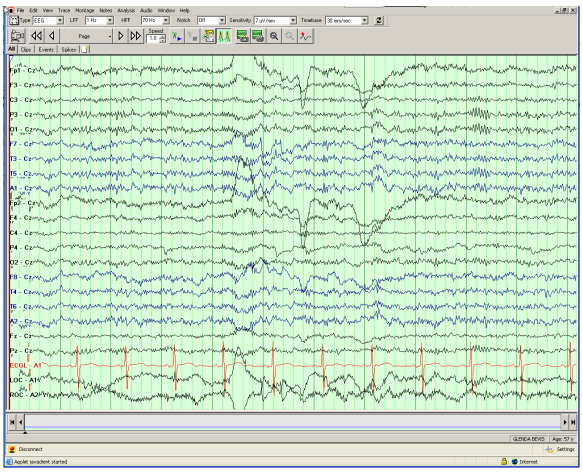
Bipolar AP Typical



Bipolar Transverse



Referential Ipsi-Ear



Referential Cz

Figure 1.4: Montage examples. These are screen captures of different montages applied to the same location in a dataset.

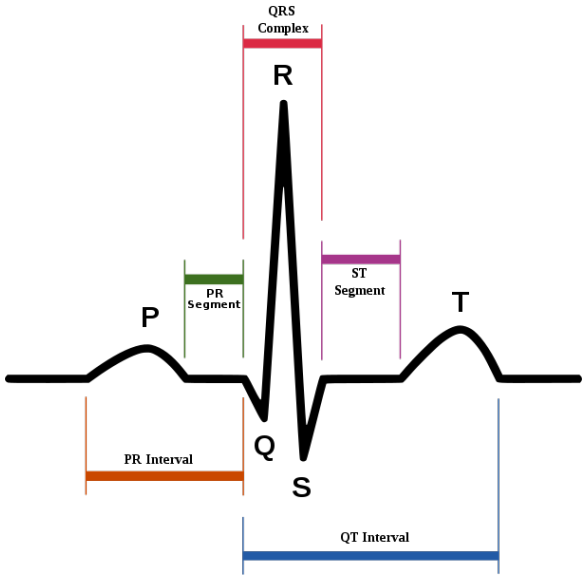


Figure 1.5: The structure to the ECG signal, allowing it to be broken down into well-defined pieces and characterized well as a low-dimensional feature vector.

classified 91.3% of the studies. There was 10%-23% variability among the cardiologists and the variability of the computer programs was even higher, although it was clear that the best computer programs performed as well as the best cardiologists [80]. The data collected and annotated from this project is still the gold standard used in automated ECG interpretation research today and can be downloaded for free from a website [86].

The interpretation of rEEG is more difficult than that of ECG, but the CSE project provides an excellent precedent and roadmap for how to proceed with developing automated analysis tools for rEEG. The CSE project gathered the best minds in the ECG research field together from 25 European, 6 North American, and 1 Japanese research institute and provided a structure for their collaboration. What turned out to be a key feature in enabling the success of this project was its "neutral" position in terms of algorithms development, in that the focus was not on developing automated ECG interpretation algorithms. The project instead focused on providing a framework of tools and standardized datasets to allow private companies to continuously improve algorithms in the field [35]. Similarly, a project that focuses on providing a framework of tools and standardized datasets for the analysis of rEEG could encourage the development of better algorithms and systems for automated rEEG interpretation. In light of the maturity of many algorithmic and signal processing techniques and the emergence of powerful new architectures for parallel development such as

cell processors and general purpose computing on graphics cards (GPGPU), a framework that can address the unique challenges of rEEG interpretation has a better chance of success now than ever.

1.3 Organization of the Research Project and this Document

As stated above, the objective of this project is to develop a computational framework for improving the automated analysis of rEEG. Our plan to accomplish this goal requires unique work from each member of our research team, which consists of Dr. Jonathan Halford of the Medical University of South Carolina, Dr. Brian C. Dean of Clemson, and the author of this dissertation. Dr. Halford's role is to gather the rEEG data from patients at the Medical University of South Carolina and recruit other academic neurologists for classification of the rEEG data. Dr. Dean's work is focused on analysis of the data from the scoring trials, and the signal processing techniques applied to the data. The author's work is dedicated to implementing systems to collect and analyze the data from the selections and classification of the neurologist scorers. Systems have also been implemented conduct this research, including systems to compare the scorers, to note their progress annotating the data, to assimilate the disparate annotations of the scorers, and to assess and visualize the accuracy of our tools.

At a high level, the way we envision this project proceeding is in two large phases. In the first phase, in which we collect data, Dr. Halford will gather a large initial rEEG dataset from clinical patients at the Medical University of South Carolina. While this data is collected, we develop software to allow neurologists around the country to help analyze this data. We will then request a group of collaborating neurologists to volunteer their time to annotate the rEEG data using the software we have developed. (We are also seeking funding from the National Institutes of Health – a portion of which we will use to compensate the reviewers for their time). At the end of this data collection cycle, we will analyze the annotations of the scorers and the performance of our tools so that we might be able to enhance our methods for subsequent data collection cycles.

In the second phase, which can be thought of as the analysis phase, we will use the data collected – which at this point will be a database of rEEG data with all of the paroxysmal activity (the PEEs) contained in it marked and classified appropriately – to develop a system to measure the variability of waveform morphology of the PEEs in the database. It is important to note that we are not attempting to classify the PEEs at this point in the project, but we are trying

to create a framework in which detecting and visualizing these PEEs is efficient and intuitive. A technique we employ to help expose these PEEs is called Independent Component Analysis (ICA), explained in Section 2.2.2. ICA is a powerful family of signal processing algorithms that are not commonly employed in clinical rEEG research because ICA is highly computationally expensive and a theoretically complex method that is overly complicated for many researchers. In this phase of the project, we will develop a highly parallel and efficient implementation of ICA.

The first phase of the project started in September of 2008. Dr. Halford collected clips of rEEG data from 40 patients. The software was developed by the author and was used by 7 certified neurologists. From this first trial, we learned how to instruct the scorers on what we were looking to classify, and we learned a great deal about the functionality and usability of the software. After reviewing those first results and receiving feedback from the scorers, we decided to re-engineer the expert opinion collection software using open web standards (including HTML5). In April of 2010, the scorers re-annotated the first 40 patients worth of data and a new, additional 60 patients worth of data. In the following months, we completed many administrative, analysis, and visualization tools to run our system and measure its correctness. Further, we completed the parallelization of a computationally expensive ICA algorithm based on Renyi's entropy.

The remainder of this document will be organized as follows:

- In Chapter 2, we review the relevant background literature including references on spike detection in rEEG, the concept of building "gold" datasets, Independent Component Analysis, some high performance computing implementations of interest, and some resources on visualization of rEEG data.

- In Chapter 3, we will discuss the methods we have used to build a dataset of expert opinion. Topics covered will include the overall process, clustering annotations from the users to produce an "intersection set", notes on our comparison techniques, and results concerning our clustering algorithm and inter-rater and intra-rater reliability from analysis of the expert opinion.

- In Chapter 4, our web-based software infrastructure is described. Systems for the scorers to annotate the datasets, administration functions for conducting the rEEG studies, and visualizations for assessing the performance of our tools and the relationships in our data are reviewed.

- In Chapter 5, we will explore in depth the problem of parallelizing a computationally intensive

Independent Component Analysis algorithm on a GPU (Graphics Processing Unit). Items covered in this chapter will be analysis of an ICA algorithm based on Renyi's entropy, the decomposition of this algorithm on the GPU, its performance against other implementations of Renyi's entropy ICA and Infomax ICA, and an interesting visualization of the results of the ICA algorithm.

- In Chapter 6, we will conclude with some future work beyond the scope of this dissertation and some final remarks.

Chapter 2

Background

This background literature review has been accumulated while preparing several documents, posters, presentations, and grant applications related to this project. Portions of this chapter are reproduced from an unpublished manuscript and an NSF grant proposal developed in collaboration with Dr. Dean and Dr. Halford.

2.1 Developing EEG Testing and Training Datasets

Developing training and testing datasets for automated rEEG interpretation software is a very expensive and time consuming affair, as the time of clinical neurologists is expensive and in-demand and the data can be challenging to interpret. In routine clinical practice of an rEEG study, an rEEG interpreter visually inspects 1-20 seconds of an EEG recording at a time, while scrolling through a typical recording length of 20-25 minutes per patient [47]. It is known that rEEG interpreters do not always mark the same events as abnormal epileptiform PEEs – an example in the literature states that with 50 rEEGs and 5 scorers, the average correlation between scorers was 0.68 [84] with similar results in [49, 16, 6, 79]. This indicates two things: (1) that many scorers be used to establish a consensus, standardized dataset, and (2) the dataset must be correlated over all of the scorers (see Chapter 3 for a description of how we approach this problem). Webber et al. [79] state that you need at least 3 scorers to draw a meaningful consensus. In his recent review paper, Halford [47] indicates that the great variability of morphology of abnormal and normal waveforms necessitates that the number EEG of datasets must be large if it is to be representative of the

population. He goes on to state that different datasets should be used for algorithm training and testing. Thus, for the reasons mentioned above, if a researcher wants to compile EEG datasets for training and testing of spike detection algorithms, he or she must: (1) make the method of acquiring these datasets simple to use so as to be considerate of the clinician's time, (2) get several clinicians to score the data so a consensus can be drawn, and (3) use many subjects in the dataset so that they will be representative of the population and encompass the variability of normal and abnormal waveforms in humans.

There are several examples of developing these datasets in the EEG literature. For an excellent summary of these see [47]; a table of the characteristics of these studies is included in Figure 2.1. These studies used between 5 and 91 subjects and between 1 and 8 scorers. One study included 521 EEGs, but no individual PEEs were marked – only if the whole EEG showed evidence of PEEs [6]. In another study by Benlamri et al. [5], the authors used a total of 137 EEG subjects, but only one human interpreter. Halford states that most studies control costs by limiting the number of EEG scorers or by not using separate training and testing datasets. In our project where we will be creating a new dataset of 100 patients reviewed by 7 human interpreters.

2.2 Signal Processing of EEG

2.2.1 Spike Detection

Given a particular source signal, we typically wish to detect whether a PEE is present in the signal, and if so, the next task is to characterize the nature of the PEE: normal or epileptiform. The term epileptiform means that presence of the PEE in the rEEG recording strongly suggests that the patient has epilepsy. Another term for an epileptiform PEE is a "spike". A significant amount of work appears in the literature for both the detection and characterization problems (also commonly called spike detection problems); many authors address both problems simultaneously. For the scope of this project, we focus on the simpler detection problem.

PEE detection (and classification) algorithms fall into two broad categories: those that attempt to perform pattern matching based on a specifically chosen model of PEE morphology, and those that automatically learn the characteristics of PEEs with no a priori specification of their features. A thorough survey of these methods appears in a recent paper by Wilson and Emerson [83], in a textbook chapter by Lopes da Silva [14], and in a survey paper by Halford [47]. The

Study	Subject number (Number of EEGs)	Spike number	EEG record Length (min)	Number of interpreters
Gotman et al. (1976)	93	NR	2	1?
Gotman et al. (1979)	24	NR	12,240	1
Fischer et al. (1980)	10	341/341	8	7
Guedes de Oliveira et al. (1983)	5/5	NR	4.2/4.2	8
Gotman and Wang (1991)	14/6	NR	1400/600	2
Gotman and Wang (1992)	20	NR	2000	2
Gabor and Seyal (1992)	5	652	63.8	1
Hostetler et al. (1992)	5	1393	100	6
Sankar and Natour (1992)	11	NR	29.5	1
Dingle et al. (1993)	11	462	180	1
Webber et al. (1993)	10	927	40	8
Pietila et al. (1994)	6	NR	360	2
Webber et al. (1994)	5/5	927	40	1
Kalayci and Ozdamar (1995)	5	1200/2414	75	2
Senhadji et al. (1995)	1?	982	10	1
Benlamri et al. (1997)	137	NR	NR	1?
Feucht et al. (1997)	4	1509	120	2
Park et al. (1998)	32	N/A	N/A	2
Ozdamar and Kalayci (1998)	5	NR	75	3
James et al. (1999)	35/8	3096/190	856/192	2–3
Sartoretto et al. (1999)	10	166	79	1
Ramabhadran et al. (1999)	6/18	NR/982	90/270	1
Wilson et al. (1999)	50	1952	143	5
Black et al. (2000)	521	NR	10,380	2–3
Goelz et al. (2000)	11	298	278	3
Ko and Chung (2000)	20	300	NR	1
Kurth et al. (2000)	4	54/NR	NR	2
Liu et al. (2002)	81	6048	~48,000	1
Sugi et al. (2002)	11	77	8	1
Castellaro et al. (2002)	NR/50	1084/NR	325/NR	4
Flanagan et al. (2003)	6/20	17302/23619	6336/15174	1
Latka and Was (2003)	4	340	NR	1
Pang et al. (2003)	3/10	NR	NR	1
Flanagan et al. (2003)	6/20	17,301/23,619	6336/16,920	1
Adjouadi et al. (2004a)	10/21	319	~800	3
Adjouadi et al. (2004b)	9/9	47/139	~450	2
Acir et al. (2005)	19/10	216/93	210/228 min total	2
Argoud et al. (2006)	7	6721	NR	3
Exarchos et al. (2006)	25	137/137	~375	2
Tzallas et al. (2006)	15/10	163/111	~225/150	2
Inan and Kuntalp (2007)	5/3	53/15	NR	2
Van Hese et al. (2008)	8	1625	130	2
Indiradevi et al. (2008)	22	684	NR	2
De Lucia et al. (2008)	7	NR	121	1

Figure 2.1: The characteristics of EEG datasets compiled in Halford 2009, [47].

findings with regard to spike detection techniques from the final paper by Halford are summarized in Figure 2.2.

The most common type of learning algorithm applied to clinical EEG research is the artificial neural network (ANN). ANNs have been used to categorize a sample of raw EEG data as normal or abnormal based on a training signal from human experts in the studies of routine clinical EEG interpretation and paroxysmal event (spike) classification [30, 66, 33, 58]. Different types of transformed EEG data have also been used as inputs to ANNs including half-wave attributes [33, 48], instantaneous power [29], and wavelet transforms [45, 25]. Various types of ANN inputs for EEG PEE detection have been compared using a limited dataset [20]. The advantage of an ANN approach is that it does not require an a priori specification of the characteristics of the PEEs to be analyzed and does not limit the data features that can be used for analysis. The algorithm learns to render an appropriate binary clinical judgment based on a training set of rEEG recordings. The disadvantage is that this method does not improve the general knowledge about the characteristics of PEE morphology or provide a quantification of PEE occurrence. Algorithms that look for PEEs matching a specific morphology model improve our general understanding of PEE signals and provide quantification of PEEs, but are highly-reliant on a priori PEE morphology specification, and their performance is easily degraded by the low SNR of the typical rEEG recording. Many methods of this flavor have been proposed in the literature. For example, the local context is used to label PEEs based on how much they stand out from the EEG waves that precede and follow them using various methods including decomposition into half-waves [33, 40, 19], the standard deviation of the local context amplitude [15] or windowed average baseline activity [31]. The overall shape of the PEE can be used to determine if it is an abnormal using half-wave attributes combined with its second derivative (i.e., spikiness) [40, 32, 28], measures of non-stationarity [72], Hilbert transform power [29] or wavelet transform output [38, 27, 60, 75]. A field criteria specifies that a spike must be detected in at least two electrodes to be classified as a clinically significant [31, 37]. Artifact rejection can be included a first stage, which prevents events from being labeled as abnormal if morphology features [40, 15, 31, 41] or dipole modeling [23] indicates that the event is artifactual. Finally, the larger context or state of EEG activity (either wake, drowsy, or sleep) during which the event occurs is taken into account in determining if the event is abnormal [42].

An ideal PEE detection algorithm would incorporate the best of both approaches above. It would not be heavily-reliant on an a priori specification of spike morphology, it would perform

Performance reported by ET detection studies.

Author	Method	Sensitivity	Specificity	Selectivity	False positive/min	Other
Gotman et al. (1976)	M				0.33	
Gotman et al. (1979)	M				0.11	
Fischer et al. (1980)	P, TM	0.73			1	
Guedes de Oliveira et al. (1983)	M	~0.65				
Gotman and Wang (1991)	SA, M				~0.25	
Gotman and Wang (1992)	M, SA			0.72	0.79	
Gabor and Seyal (1992)	NN	0.942		0.791	1.5	
Hostetler et al. (1992)	M	0.76		0.89	5.2	
Sankar and Natour (1992)	P			0.175		
Dingle et al. (1993)	M, ES	0.67	1.0		0	
Pietila et al. (1994)	FT, M	0.31	0.33			
Webber et al. (1994)	M, NN	0.736		0.736	6.1	
Kalayci and Ozdamar (1995)	WT, NN	0.908	0.932			
Senhadji et al. (1995)	P, WT	0.86			6.8	
Benlamri et al. (1997)	M					13%[a]
Feucht et al. (1997)	HT, NN	0.865–0.899		0.846	1.83	
Park et al. (1998)	WT, NN, ES	0.97		0.895		
Ozdamar and Kalayci (1998)	NN	0.93		0.94		
James et al. (1999)	M, NN	0.553		0.82	0.12	
Sartoretto et al. (1999)	WT	0.70		0.52	1.28	
Ramabhadran et al. (1999)	M, ES	0.96			0.4	
Wilson et al. (1999)	M, NN	0.47	0.9963		2.5	
Black et al. (2000)	M, ES				0.0068	
Goelz et al. (2000)	WT	0.84		0.12		
Ko and Chung (2000)	NN	N/A	N/A	N/A	N/A	
Kurth et al. (2000)	NN	0.623		0.61		
Liu et al. (2002)	P, WT, NN, ES	0.900		0.936		
Sugi et al. (2002)	FT, M	0.37			16	
Castellaro et al. (2002)	FT, NN					2%[a]
Flanagan et al. (2003)	M, DA					[e]
Latka and Was (2003)	WT	0.70		0.67		
Flanagan et al. (2003)	M, DA					[d]
Adjouadi et al. (2004a)	M, WaT	0.82	0.93	0.92		
Adjouadi et al. (2004b)	M, WaT	0.79	0.88	0.85		
Acir et al. (2005)	M, NN	0.891		0.859	0.125	
Argoud et al. (2006)	WT, NN, ES	0.709	0.9999		0.13	
Exarchos et al. (2006)	FT, M	0.861	0.917	0.831		
Tzallas et al. (2006)	M, FT, NN, ES	0.892	0.911	0.825		
Inan and Kuntalp (2007)	M, NN, O[b]	0.604	0.987	0.818		
Van Hese et al. (2008)	DA	0.92		0.77		
Indiradevi et al. (2008)	WT	0.917	0.893	0.893		
De Lucia et al. (2008)	O[c]	0.65	0.86		6	

In locations where no value is given, the performance measure was not reported.
DA, dipole analysis; ES, expert system; FT, Fourier transform; HT, Hilbert transform; M, mimetic; N/A, not applicable; NN, neural network; O, other; P, parametric; SA, state analysis; TM, template matching; WT, wavelet transform; WaT, Walsh transform.

[a] Percent of studies misinterpreted due to false positives.
[b] K-means clustering.
[c] Independent component analysis and quadratic classifier.
[d] Improvement of spike detection algorithm with 53% of artifactual detections removed while removing 4.3% of spike detections.
[e] Artifact detections reduced by 53% while only removing 4.3% of ET detection.

Figure 2.2: Spike detection research and the approach used and characteristics from [47]

well is a dataset with low SNR, and it would provide quantitative measures of the morphology and frequency-of-occurrence of PEEs.

2.2.2 Independent Component Analysis (ICA)

Part of the difficulty in analyzing EEG data is that each electrode typically records a mixture of different fundamental "source" signals from different regions in the brain. A potentially useful preprocessing step in EEG analysis is blind signal source separation (BSSS), the process of isolating individual source signals from a mixture of signals. The classic description of how BSSS works is the "cocktail party problem", in which BSSS analysis is able to reconstruct the voices of N stationary

speakers in a cocktail party based on N stationary microphones placed throughout the room [7]. More formally, let S be an $N \times L$ matrix whose N rows $s_1 \ldots s_N$ give the signals produced by the speakers, and let X be an $N \times L$ matrix whose N rows $x_1 \ldots x_N$ give the signals recorded by the microphones. Based on its location relative to the speakers, each microphone is assumed to record a linear combination of the speaker signals. We can therefore relate S and X using the matrix equation $X = AS$, where A is an $N \times N$ *mixing matrix* describing the proportion of each speaker that contributes to each microphone. Given only the microphone recordings X, the goal of the BSSS problem is to factor X into A and S to recover the original speaker signals. Since there is generally no unique solution to this problem, we typically try to find a set of s_i's that are as statistically independent as possible, a process known as independent component analysis (ICA).

BSSS algorithms such as ICA are quite useful in analyzing EEG data because the signal-to-noise ratio (SNR) of EEG is low (by contrast, the electrical signals recorded by ECG are approximately 1000 times stronger than that of EEG and less easily obscured by artifact [53]). BSSS algorithms provide a degree of separation of electrocortical from artifactual signals (eye movements, electromyographic signals, electrode signals), which can effectively boost the SNR [74]. The main application for ICA in EEG research to date has been the removal of signal artifacts. However, some clinical neurophysiology researchers have begun using BSSS as a preprocessing step in their pattern recognition algorithms [74, 61], with the hope that the improved SNR of the ICA-processed EEG data will improve the ability of computers to recognize signal patterns such as PEEs. It has been shown that the performance of pattern recognition algorithms can be improved by first applying BSSS to the neurophysiology datasets before a pattern recognition algorithm is applied [88].

There are three main obstacles to applying BSSS analysis to rEEG data, due to fundamental underlying assumptions imposed by the BSSS / ICA problem:

1. **Stationarity.** One of the assumptions of ICA is that the distances between the source generators and the recording devices do not change (are "stationary") during the time period of analysis. By analogy, in the cocktail party problem, we assume the speakers at the party are stationary, rather than walking around the room. Although the source locations for brain activity are anatomically fixed, the sequential activations of interconnected neurons which occur during brain activations frequently cause non-stationarity in measured rEEG signals [21]. Non-stationarity in the dataset can render the results of ICA analysis unreliable [21].

2. **Source Number Determination.** Conventional ICA algorithms assume that the number of source generators is equal to the number of observed source signals. This is often not the case in neurophysiologic recordings [21]. In short datasets acquired with high-density neurophysiologic recording devices such as MEG, the number of observed source signals can greatly outnumber the number of generators, causing distortion in the ICA results [50] – this is termed the "overfit" problem of ICA [53]. If there are substantially more source signals then there are recording electrodes, as in rEEG recordings, important source generators may not be resolved [21] – this is termed the "over-completeness" problem of ICA. Most researchers currently applying ICA do not attempt to determine source number before running ICA analysis [21], and a mismatch between the number of recording sources and the number of source generators can render ICA analysis unreliable.

3. **Independent Component Selection.** The output of ICA analysis is a set of N independent component (IC) signals of the same temporal length as the original EEG segment, each with associated scalp distribution vectors (SDVs) that describe the geometric configuration of recording electrodes from which the IC signal was recorded (these correspond to the rows of the ICA mixing matrix). Once a segment of EEG has been analyzed with ICA, there is no accepted system for interpreting the results. Selection of which IC is representative of the source signal of interest is currently based on subjective perception of relevance by the investigator and measures of IC signal variance [53].

A variety of different algorithmic approaches for ICA have been proposed in the literature, including Infomax ICA [3], FastICA [51], joint approximate diagonalization of eigenmatrices (JADE) [68], and second-order blind identification (SOBI) [10]. Although somewhat more computationally intensive, a recent algorithm of Principe et al. [87, 52] based on Renyi's entropy has been shown to exhibit somewhat stronger performance than these others; this is the approach we study in detail later in this work (Chapter 5), since it is well-suited for parallel implementation.

Recall the goal of BSSS: given a matrix X whose rows $x_1 \dots x_N$ describe mixtures of signals, we wish to compute a factorization $X = AS$, where A is a mixing matrix and the rows $s_1 \dots s_N$ correspond to source signals. Re-phrasing, we want to find an $N \times N$ de-mixing matrix W such that $Y = WX$ contains in its rows a set of unmixed signals that are as statistically independent as possible (ideally, $W = A^{-1}$, giving $Y = S$). As an optimization problem, we wish to maximize some

function $f(Y)$ reflecting the statistical independence of the rows of Y over all choices of de-mixing matrices W, so we are performing optimization over the N^2 decision variables W_{ij} for $1 \leq i, j \leq N$.

The process of *whitening* is common preprocessing step before ICA. A matrix M containing signal data in its rows is said to be "whitened" if $MM^T = I$; that is, each signal is normalized to have unit variance, and all pairs of different signals are uncorrelated with each-other. If $Q = \Lambda^{-1/2}V^T$, where Λ is a diagonal matrix containing eigenvalues of XX^T and V is an orthonormal matrix containing the eigenvectors of XX^T in its columns, then we can whiten the signal mixture matrix X by the linear transformation $\hat{X} = QX$. This is possible because the transformed signal matrix $\hat{X} = QX = QAS$ satisfies

$$\hat{X}\hat{X}^T = QXX^TQ^T = \Lambda^{-1/2}V^TXX^TV\Lambda^{-1/2} = \Lambda^{-1/2}V^TV\Lambda V^TV\Lambda^{-1/2} = I,$$

because of the standard factorization $XX^T = V\Lambda V^T$ that diagonalizes XX^T in terms of its eigenvalues and eigenvectors.

The primary benefit of whitening is that if our matrix of mixed signals X has been whitened (an assumption we make henceforth), then an ICA algorithm only needs search for a factorization $Y = WX$ where the de-mixing matrix W has *orthonormal* columns. To see this, let X denote our original matrix of mixed signals, with $\hat{X} = QX = QAS$ being the matrix of whitened mixtures. Since we assume our original sources are independent and hence uncorrelated ($SS^T = I$), the new mixing matrix QA for obtaining the whitened mixtures from the original signals satisfies

$$I = \hat{X}\hat{X}^T = (QAS)(QAS)^T = QASS^TA^TQ^T = QAA^TQ^T = (QA)(QA)^T.$$

While it generally takes N^2 variables to specify an $N \times N$ matrix, there are only $\binom{N}{2}$ degrees of freedom in an orthonormal $N \times N$ matrix, roughly halving the number of decision variables we must consider in our search for the best de-mixing matrix W. The fact that W is constrained to be orthonormal also allows us to represent W as a product of $\binom{N}{2}$ rotation matrices, one within each 2D plane formed by a pair of dimensions. This equivalent formulation gives us a different set of $\binom{N}{2}$ decision variables (the angles used in each of the rotation matrices) over which we will optimize later in our parallel implementation of the Renyi's entropy ICA algorithm of Principe et al. [87, 52]. This also gives a nice geometric interpretation of ICA: letting X be our whitened mixture matrix,

we can plot each of its L columns as a point in N-dimensional space. The orthonormal de-mixing transformation $Y = WX$ then rotates these points to express them in a new basis (with the new points being the columns of Y), so that the marginal distributions obtained by projection onto the new basis vectors (the rows of Y) correspond to distributions that are as statistically independent as possible. To solve this optimization problem,

$$\max_{W \text{ orthonormal}} f(Y = WX),$$

several reasonable objective functions f have been proposed in the literature. Most of these are based on the principle (due to the law of large numbers) that mixtures of signals tend to have Gaussian-shaped histograms, while source signals tend to have non-Gaussian histograms. Letting $y_1 \ldots y_N$ denote the rows of Y, we can therefore try to maximize an objective $f(Y) = g(y_1) + \ldots + g(y_N)$, where $g(v)$ is some function that measures the amount of "non-Gaussianity" of its vector argument v. Perhaps the simplest choice for g is the absolute value of kurtosis (this is often referred to as excess kurtosis), defined as follows assuming $||v|| = 1$ and $\sum_i v_i = 0$:

$$g(v) = |\operatorname{kurt}(v)| = \left| \sum_{j=1}^{L} v_i^4 - 3 \right|.$$

If v's entries follow a Gaussian distribution, kurtosis is zero; non-Gaussian distributions usually result in non-zero values for kurtosis. Another common choice for g is to measure the similarity to a prototypical non-Gaussian probability distribution. This is typically done by maximizing the entropy of the histogram we get when the entries of v are mapped according to the inverse cumulative density function of the target distribution (since ideally, this mapping should result in a uniform distribution if our source signal matches the target distribution exactly, and the uniform distribution has maximum possible entropy). Equivalently, one can minimize the Kullback-Leibler (KL) divergence between the source signal and target distributions (KL divergence is a common "distance measurement" between two probability distributions). This choice of objective is commonly used as part of the widely used Infomax ICA algorithm [3].

Another popular class of objective for ICA is to minimize the mutual information between $y_1 \ldots y_N$, which (assuming whitening has been applied) is shown in [9] to be mathematically equivalent to maximizing $g(y_1) + \ldots + g(y_N)$ where $g(v)$ gives the *entropy* of the probability distribution

implicitly described by the entries of v. To estimate the shape of this probability distribution, one typically uses *kernel density estimation* based on *Parzen windows* — superimposing small Gaussian "kernels" centered on each sample point from v to obtain a smooth estimate of the probability distribution these points represent. The Shannon entropy of such an estimate can be computed using the following estimator [52]:

$$g(v) = \frac{-1}{L} \sum_{i=1}^{L} \log \frac{1}{L} \sum_{j=1}^{L} G(v_i - v_j, \sigma^2),$$

where $G(x, \sigma^2)$ is the Gaussian function evaluated at point x with variance σ^2, and σ represents the standard deviation of the Gaussian kernel we are using for density estimation. At $O(L^2)$ time per evaluation, this estimator is quite computationally expensive to use. An alternative proposed in [22] gives an estimator of Renyi's entropy at $\alpha = 2$[1] that can be evaluated in only $O(L)$ time

$$-\log \frac{1}{L} \sum_{j=1}^{L} G(v_j - v_{j-1}, 2\sigma^2).$$

By biasing this estimate using the sign of the Kurtosis of v (so it is maximized by vectors v with entries distributed in a Gaussian manner), [52] shows that minimization of the objective $f(Y) = g(y_1) + \ldots + g(y_N)$ appears to outperform most other prominent ICA objectives through empirical testing, where

$$g(v) = -\operatorname{sgn}(\operatorname{kurt}(v)) \log \frac{1}{L} \sum_{j=1}^{L} G(v_j - v_{j-1}, 2\sigma^2).$$

It is this objective we use for the parallel ICA implementation we describe later in this dissertation.

There are two prominent methods used for solving the maximization problem inherent in ICA. The first, sometimes known as *projection pursuit*, identifies source signals one at a time, by finding successive one-dimensional projections of the mixed signals whose histograms are as far from Gaussian as possible. After finding a vector onto which a good projection exists, this projection is subtracted out of the dataset via Gram-Schmidt orthogonalization and we begin searching for a new direction on a reduced dataset of one lower dimension (this approach is sometimes known as "deflation" in the linear algebra literature). Other ICA algorithms optimize over the entire de-

[1] Renyi's entropy is actually a family of entropy functions parameterized on a value α; for $\alpha = 1$, it reduces to Shannon's entropy. In [52], it is shown that Renyi's entropy for $\alpha = 2$ works quite well for ICA, and moreover (as we see above) it is algorithmically well-behaved in allowing a fast estimator.

mixing matrix W at once, identifying all unmixed signals $y_1 \ldots y_N$ simultaneously rather than one at a time. In either case, optimization is typically performed either by gradient descent, or by fixed point iteration (an approach known as the FastICA algorithm [51]).

2.2.3 mBSSS

For analyzing a typical half-hour rEEG recording with ∼20 channels, an ICA computation applied to the entire input set will often encounter serious problems due to over-completeness (since we usually encounter far more than ∼20 individual source signals over the duration of the recording) and non-stationarity (since source signals often migrate over time). To mitigate these issues, a common approach is to apply ICA multiple times within a short (say, ∼10 second) sliding window that moves through the dataset. Each invocation of ICA produces a set of independent components (ICs) which are then further analyzed to detect the presence of PEEs, and finally there is a correlation phase where the PEEs produced during each invocation are compared to eliminate duplicates (since now the same PEE can be discovered multiple times if several instances of the sliding window contain the PEE). Let us call this generic approach sliding window blind signal source separation (swBSSS). One inherent difficulty with swBSSS is the choice of window size: we want windows that are sufficiently large to capture all PEEs of interest and to avoid convergence issues with ICA algorithms (FastICA in particular tends to have difficulty converging on windows of small size; in our preliminary tests of 19-electrode recordings sampled at 256 Hz, convergence improved dramatically once window sizes reached 5...6 seconds in length). However, window sizes must be small enough to avoid problems with over-completeness and non-stationarity. In order to address the difficulty of choosing a single fixed window size, Halford recently proposed a new approach called multitaper blind signal source separation (mBSSS) [46], in which ICA calculations are performed in a multitaper pattern using windows of various lengths throughout the EEG dataset, and the resultant ICs are collectively analyzed to detect PEEs just as before. This approach allows for the detection of an unlimited number of PEEs without pre-specifying length, timing, signal morphology or scalp distribution, and in addition provides a possible method for overcoming the non-stationarity and over-completeness problems.

The mBSSS algorithm generates a set of BSSS windows according to four parameters: minimum and maximum window lengths W_{min} and W_{max}, an expansion factor W_e, and an overlap factor W_0. For each window length $k \in \{W_{min}(1 + W_e)^p \mid p = 0, 1, 2, \ldots\}$ less than W_{max}, we solve

an ICA problem on windows of length k starting at positions $0, k(1W_0), 2k(1W_0)$, etc., across the entire dataset. In our present computational tests, we set W_{min} and W_{max} so that our windows range in length from 0.5 seconds to 12 seconds, we set $W_e = 0.2$ so each successive window length is 20% larger than the previous length, and we set W_0 0.50 so consecutive windows overlap by roughly 50% of their length. Note that we can describe the set of windows generated by the simpler swBSSS approach in the same framework (since swBSSS is a special case of mBSSS), by setting $W_{min} = W_{max}$.

We can think of swBSSS and mBSSS as meta-algorithms in the sense that any specific implementation for a blind signal source separation algorithm can be plugged in for use with the individual window computations. Hence, swBSSS and mBSSS provide a means of generalizing any approach for BSSS via computation across multiple windows in order to mitigate the problems of non-stationarity and over-completeness. Since mBSSS causes BSSS calculations to be performed in windows of many different sizes that include the time-of-occurrence of a given PEE, it becomes unlikely that all such windows will be over-complete with respect to the target PEE, even if the target PEE has a very unique signal morphology and scalp distribution in comparison to all other PEEs in the rEEG dataset. For the same reason, mBSSS also helps to overcome the stationarity problem. If a PEE source signal (or signals) are non-stationary, the output of the mBSSS analysis of single PEE can provide some measure of this by producing a list of successive ICs (and their associated scalp distributions), possibly of various lengths, that track the movement of the source signals during the time-of-occurrence of the PEE. Some authors have recently tried to develop more sophisticated BSSS models that can handle certain types of non-stationarity [10, 2, 34, 67, 68]; by way of comparison, we can think of the mBSSS algorithm as a simple approach for generalizing any standard BSSS algorithm (at the expense of significant extra processing time) to achieve robustness in the presence of non-stationarity.

2.3 High Performance Computing and ICA/mBSSS

A number of different computing environments are suitable for implementation of the ICA/mBSSS algorithm. We can separate these at a high level into tightly-coupled parallel environments in which we have either shared memory or a very fast interconnection between processors, versus distributed environments in which processors are spread throughout a slower or higher-latency

network. The primary challenge inherent in parallel or distributed implementation of any embarrassingly parallel algorithm is to schedule tasks across processors in a manner that ensures balanced utilization. For mBSSS, this task will be non-trivial, due to the heterogeneous task sizes arising from ICA computation on windows of different lengths. Note that the problem is even hard for swBSSS, since it is hard to predict the duration of an ICA calculation simply from the size of a window; one can experience fast convergence on large windows and also slow convergence on small windows. Another challenging problem (especially in the distributed case) is the efficient distribution of input data to parallel processors, and subsequent the accumulation of the output of each window computation back to a centralized location.

Although we can treat the windows generated by mBSSS in a completely independent fashion, gains in efficiency may be possible by saving some state between successive window computations. For example, the ICA unmixing matrix computed for one specific window may provide a good starting point that would lead to accelerated convergence for ICA computation on nearby windows. Finally, we may be able to prune certain windows from consideration in an adaptive fashion for example, it may be possible to exclude consideration of a large window if no paroxysmal activity is observed in any of its smaller subwindows. These dependencies between tasks will complicate our load balancing calculations in a distributed setting.

2.3.1 Graphics Processing Units (GPUs)

Graphics Processing Units (GPUs) found on inexpensive video cards have recently surged past traditional CPUs in terms of computing power. For roughly the same price, one can now purchase a graphics card that outperforms the fastest Intel desktop CPUs by more than an order of magnitude, in terms of raw floating point operations (flops). This extra speed is not quite as easy to harness, due to the highly specialized architecture of the GPU. Nevertheless, the potential for such a significant speedup using inexpensive commodity hardware has ignited a tremendous interest in the use of GPUs for general purpose computation of all sorts. For an extensive survey of recent results related to general-purpose GPU (GPGPU) programming, please see [26].

The GPU is a special-purpose processor that does not support many of the common operations of a traditional CPU. For instance, GPUs are not adept at executing code that makes significant use of conditional branching (e.g.,"if a < 0, then do X else do Y). Another drawback for the GPU is that its manipulation of double precision variables is very slow compared to its performance with

single precision floating points; however, we do not expect this limitation to affect our calculations in a meaningful way.

On the positive side, the GPU is extremely fast at certain types of operations. For example, it is ideal for performing stream computation, where we envision a sequence of input numbers passing through a processing unit that performs some simple operation (e.g., filtering out certain numbers, accumulating a running sum, scaling or translating the numbers by a fixed amount, etc.). A typical GPU contains dozens of processor cores, each of which can process its own stream of input elements independent of the others. As a result, we can achieve impressive speedups for many primitive mathematical operations, particularly those from linear algebra such as vector addition, vector rescaling, vector norms, inner products between vectors, matrix-vector multiplication, matrix-matrix addition and multiplication, and many more. Since the iterative update rules in the FastICA, Infomax, and Renyi's entropy ICA algorithms consist primarily of these types of operations (see previous sections), we expect the GPU to offer a significant improvement in performance. Even the evaluation of a hyperbolic tangent (one of the update steps for both FastICA and Infomax ICA) can be done quickly in parallel on a GPU. Until recently, the task of writing code for a single GPU-equipped machine was a somewhat daunting endeavor. However, new programming languages such as CUDA and OpenCL, new tools, and new debuggers, have been recently introduced to make this job somewhat simpler (although still far from trivial); see [26, 65, 8, 43] for a detailed summary of these developments.

2.3.2 Implementations of ICA

There are two prominent serial implementations of ICA in existence. FastICA is featured in the prominent IT++ signal processing library, which is free software and written in C++. And Infomax ICA, available in MATLAB and C implementations that accompany EEGLab [1], which are available for free from the Schwartz Center of Computational Neuroscience at UCSD.

Several parallel ICA implementations have been described previously in the literature. Dedicated hardware implementations using analog Complementary Metal Oxide Semiconductor (CMOS) circuits [13] as well Field Programmable Gate Arrays (FPGAs) [17]. For standard multiprocessor systems, a C++ library for ICA computation known as HiPerSAT [24] has been developed using the Message Passing Interface (MPI), a standard specification for communication between parallel processes. HiPerSAT implements both FastICA and Infomax ICA, and it presently requires a

shared-memory parallel environment (e.g., a multi-core processor), as opposed to a distributed grid environment with processors connected over a slower network, in order to achieve good performance, due to the large amount of communication required between processes. The HiPerSAT implementation of FastICA performs quite well, with nearly linear speedup in the number of processors available. On the other hand, the performance of Infomax ICA was somewhat less impressive, with speedups around 3.3x on both an 8-processor and 16-processor system; the authors attribute this to greater proportion of sequentially-executed code in the Infomax algorithm [24].

2.4 rEEG Visualizations

Visualization of rEEG is a well studied field with many results. Much of the work focuses on the localization and visualization of the raw rEEG signal in 2 or 3 dimensional space. rEEG is a ideal candidate for this kind of localization because of the spatial relationships between the electrodes of the recording system. There are many different techniques to perform this localization. In the 2 dimensional case, there are results using graph-based approaches [77], and fast implementations using a "tiled" coordinate system [76]. In the 3 dimensional case, results employ Topographic Maps [70, 55, 71] , Genetic Algorithms [62], and Bayesian networks [69]. In the case of these 3 dimensional visualizations with Topographic maps and Bayesian networks, there are some very interesting freely available, open source implementations of these called "TEMPO" [55, 71] and "BrainStorm" [69].

A large focus of our work deals with the notion of first detecting the independent components of the source signal prior to searching for PEEs. Thus, visualization of the relationships between PEEs and their independent components and original source signals are of great interest. An example in the literature of such a visualization is the 2 dimensional spatial visualization of independent components in EEGLab [1], as illustrated in Figure 2.3.

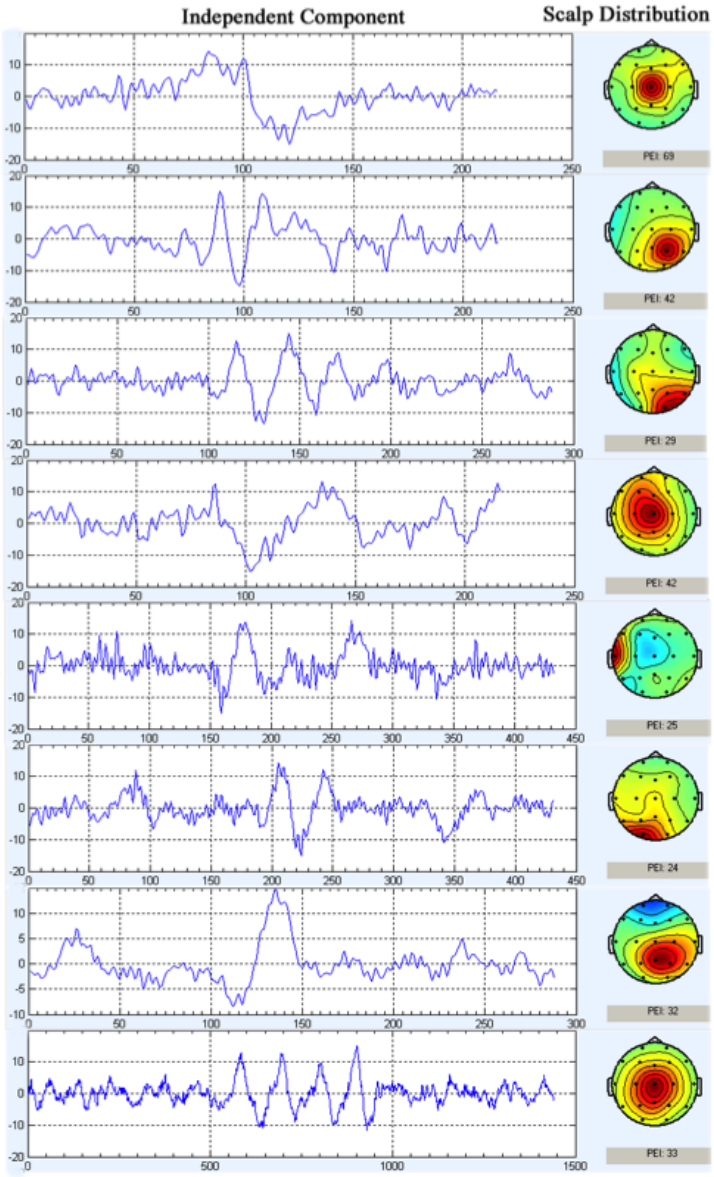


Figure 2.3: The independent components of the EEG signals and their scalp distributions. Image generated by EEGLab. [1]

Chapter 3

Collection and Analysis of Expert Opinion

3.1 The Process For Collecting Expert Opinion

As described in Section 2.1, there are certain guidelines that should be followed to successfully create an EEG training and testing dataset: (1) it should simple to do so as to be considerate of the clinician's time, (2) several clinicians should score the data so a consensus can be drawn, and (3) the dataset should contain many subjects so that it will be representative of the population and encompass the variability of normal and abnormal waveforms in humans.

The process we developed for creating the training and testing datasets was this:

1. Dr. Halford gathers the clinical data – more specifically, 30 second clips from subjects with and without epilepsy. He then de-identifies it and sends it to Clemson to upload into the database for the annotation software.

2. In a first pass, neurologist scorers log into the software[1] and select areas of the rEEG data that they think are paroxysmal (i.e. PEEs). For the sake of future discussion, we will refer to these selections as the *annotations*. Each annotation is a contiguous region of time highlighted on a single channel in any montage of the scorer's choosing. If a PEE is evident across multiple channels, scorers are instructed to highlight it once on the single channel they feel

[1] All software is described in detail in Chapter 4.

displays it best. In this pass, the annotations are not classified. Seven American Board of Clinical Neurophysiology-certified EEG interpreters (Drs. Benbadis, Edwards, Halford, Pritchard, Tatum, Turner, and Arain) participated in this study.

3. When the neurologists have completed their first pass, we process their annotations with clustering software to choose the "consensus set" obtained by "intersecting" their annotations. This is a critical step in that there are many ways in which annotations from different users can be describing the same event, such as selecting the same waveform in two different montages, or having the annotations nested or overlapping. Only events indicated by the annotations of two or more scorers will be included in the final result set.

4. In a second pass, the scorers are presented the output of the clustering algorithm for classification. This is done using a different mode of the same software used for marking the annotations. The scorers now classify the consensus annotations into 3 categories: artifactual, normal electrocortical, or abnormal electrocortical epileptiform. For the sake of future discussion, we will refer to these as *classifications*. For these first 100 subjects, the same 7 scorers participated.

5. We analyze the results of the classification pass, and seek consensus in the classifications. Much of this analysis is dependent on visualization tools we developed, mentioned in Chapter 4.

The remainder of this chapter contains detailed methods and results from analyzing the expert opinion.

3.2 Clustering Algorithms for Determining Scorer Agreement

3.2.1 First Generation Clustering Algorithm

The scorers selected annotations in the first phase of the dataset collection. Since they could select any region of the waveforms they found interesting. It has earlier been stated that it is common for scorers to mark different regions of EEG as indicative of the same PEEs [84]. We need to be able to determine the distinct annotations from the scorers, as their annotations might be of different lengths and in different channels or montages. Therefore we implemented software that

looked at every annotation from the first pass of the scorers and clustered the annotations temporally while assuring that the signal morphology of similar annotations could be clustered together (as in the case of annotations in different montages or different, adjacent channels). The input for the program was a clustering score threshold Δ, the annotations from all of the scorers, and the rEEG data itself. The clustering cutoff Δ is a measure of minimum similarity that determines if two annotations should belong to a common cluster. The value of this cutoff falls in the range [0,1], and for the first datasets, we used an empirically observed value of $\Delta = .75$.

To compute the correlation between annotations x and y, we first zero-pad them both so they share a common start and end time. We then regard the signal data in x and y as vectors v_x and v_y of the same length and compute their correlation

$$C(x, y) = \hat{v}_x \cdot \hat{v}_y,$$

where $\hat{v} = v/||v||$ denotes the normalized vector v. Observe that the formula for $C(x, y)$ is nothing more than the standard Pearson product-moment correlation formula applied to the vectors v_x and v_y and falls in the range [-1, 1].

Given a clustering coefficient threshold Δ, we construct a graph G whose nodes are the annotations, in which an edge connects any pair of annotations (x, y) for which $C(x, y) \geq \Delta$. The connected components of this graph are our clusters. For each cluster, we elect one of its constituent annotations as its "representative", or "center" by assigning each annotation x a centrality score

$$\text{centrality}(x) = \sum_y C(x, y),$$

and choosing the annotation of highest centrality to be the representative of a cluster. We argue that this is a sensible method for cluster center selection, due to its strong connection with cluster center selection in the prominent k-means clustering technique. In k-means clustering, if we have a cluster containing points $v_1 \ldots v_n$ (in some high-dimensional space), then we must assign a "center" point p to this cluster minimizing

$$d(p) = \sum_{i=1}^{n} ||p - v_i||^2.$$

If p is allowed to be any point in space, the optimal solution of this problem is to simply set $p = \frac{1}{n} \sum v_i$ as the average of the n points. However, we are often constrained to choose a point

from our data set as a center, in which case we simply choose whichever of the v_i's minimizes $d(v_i)$. We note that this is *exactly* the same process as our method above for choosing a representative element of a cluster, since if we let $v_1 \dots v_n$ denote vectors representing the n annotations $x_1 \dots x_n$ in a single cluster (zero padded so they all share the same start and end time, and hence have the same length), then

$$d(\hat{v}_i) = \sum_{j=1}^{n} (\hat{v}_i - \hat{v}_j) \cdot (\hat{v}_i - \hat{v}_j) = \sum_{j=1}^{n} (2 - \hat{v}_i \cdot \hat{v}_j) = 2n - 2\sum_{j=1}^{n} C(v_i, v_j) = 2n - 2\text{centrality}(x_i),$$

so selecting an annotation x_i of maximum centrality exactly corresponds to the selection of a k-means cluster center minimizing $d(\hat{v}_i)$.

The output of this program is a set of distinct annotations (cluster centers) along with pairwise correlations of all pairs of temporally-overlapping annotations. These pair-wise correlations can then be visualized in a web-based system described in the following Chapter (see Figure 3.2).

3.2.2 Second-Generation Clustering Algorithm

The clustering algorithm above presented several problems. Recall that when calculating correlation in the first generation clustering algorithm, we take the dot product of the overlapping part divided by the Euclidean length of each annotation. The rationale behind dividing by the length was to decrease the similarity between two annotations that barely overlap (we refer to this as a "length penalty"). This length penalty unfortunately caused many annotations not to be clustered together that should have. A simple illustration of this would to consider two annotations in the same channel where one annotation is very large and overlapping an annotation that is a fraction of the larger one's size. The smaller annotation could be almost fully overlapped (90% of its length, for example), but the huge difference in their lengths would cause their correlation score to fall below the cutoff.

Another problem with this clustering algorithm was the case where one annotation completely contained one or more other annotations. The scorers were instructed by Dr. Halford to mark the individual bursts of a sequence of back-to-back bursts of paroxysmal activity present in the data. Unfortunately, some of the scorers would mark the whole sequence with one large annotation. This would result in potentially all of the annotations becoming their own clusters as well as the large containing annotation, and yielding the highly undesirable result of nested cluster centers.

In considering these issues, we realized that the similarity of two annotations is really a two dimensional problem where one must concern both the morphological similarity and the temporal overlap of two annotations independently. In light of this, we devised a new algorithm to mitigate these issues. In this algorithm there is still a cluster cutoff Δ to decide when to cluster two annotations, but we introduce another parameter in the *overlap threshold*, Ω. The overlap threshold for two annotations is defined as the percentage of the smaller of the two annotations that must be overlapped by the larger for the two to be considered part of the same cluster. We use $\Delta = 0.35$ and $\Omega = 0.50$ for the values of these parameters. As a justification for $\Delta = 0.35$, it should be observed that the probability of multiplying two random vectors and obtaining a similarity score of more than 0.35 is highly unlikely, as illustrated in Figure 3.1. This would suggest that any cutoff above .35 would probably be suitable in that it would not likely be satisfied due to chance.

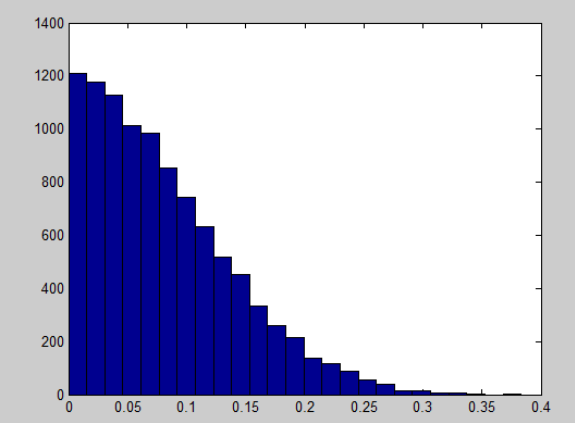


Figure 3.1: A histogram of the similarity score obtained by multiplying together two random vectors. This provides insight in where we should set the clustering cutoff parameter Δ for our clustering algorithm.

The modified correlation function we use in the second-generation algorithm computes the correlation $C'(x, y)$ between annotations x and y by truncating them to *only the region of time over which they directly overlap*, so they share a common start and end time. We then represent the truncated annotations x and y by vectors v_x and v_y, and again set their correlation to $C'(x, y) = \hat{v}_x \cdot \hat{v}_y$, using the Pearson product-moment correlation formula.

The algorithm starts with a first pass that marks all of the annotations that contain other annotations with a flag to ignore them during the clustering pass (in order to prevent creation of nested cluster centers). After this, our clustering algorithm is similar to the first-generation

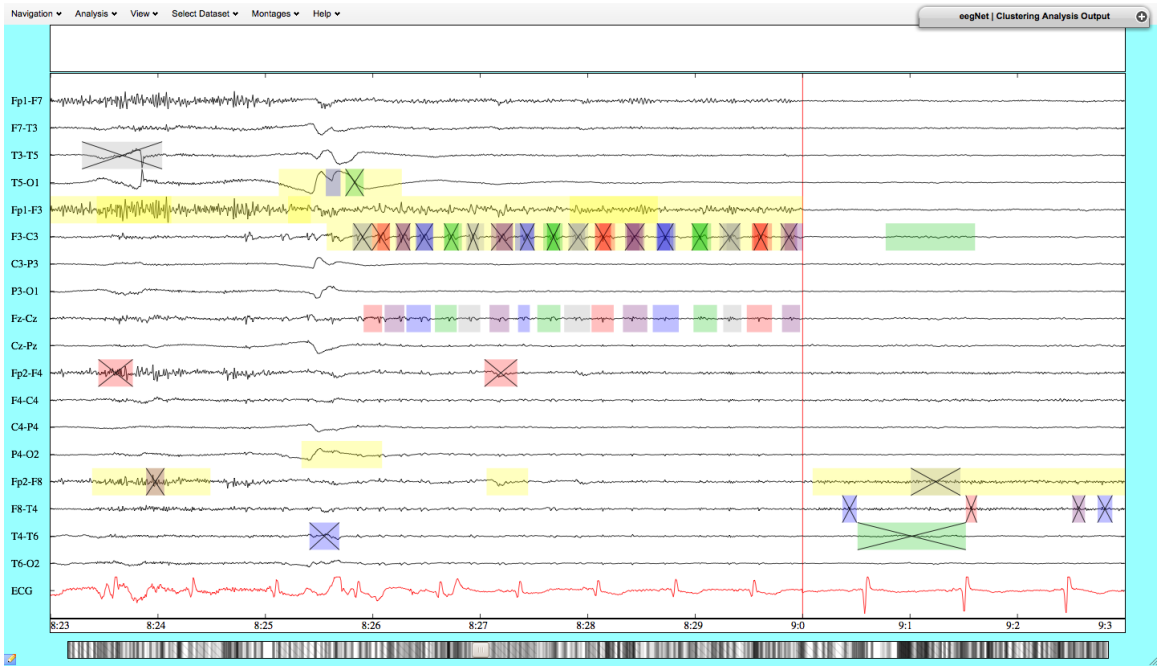


Figure 3.2: A screen capture of a clustering result. Yellow boxes are ones that are associated with more than one cluster. Boxes of matching colors (non-yellow) represent a cluster. An X across an annotation means that it is a cluster center.

algorithm. Given a clustering coefficient threshold Δ and an overlap threshold Ω, we form a graph G whose nodes are the annotations with ignore flags set to false, in which a pair of annotations (x, y) is connected by an edge if $C'(x, y) \geq \Delta$ and at least an Ω fraction of the smaller annotation is covered by the larger annotation. At this point, the connected components in G give us our initial set of clusters. Next, we re-examine each annotation with its ignore flag set to true. If this annotation, added to our graph, only contains edges to one single cluster, we add the annotation to that cluster. Any annotations that remain unclustered can now be viewed as multiply-clustered. These annotations will count towards the total agreement of the scorers, but will not be counted as a member of any particular cluster. We again choose the annotation x of maximum centrality(x) in each cluster to be the cluster center (here, we use the original formula for centrality, based on the correlation measurement C rather than C', since this makes more sense for centrality computation due to its relationship with k-means clustering).

For an example result of this clustering algorithm in a very challenging location in the data, see Figure 3.2.

3.3 Results of Expert Opinion Collection

3.3.1 Our Evaluation Techniques

In this section, we will briefly mention the techniques we use to compare the annotation and classification sets.

Precision and *recall* are two concepts that come from the field of Information Retrieval (IR), a well studied field that involves searching for data in some large repository. We believe IR is a good model for our problem if you view two annotations as analogous to two documents that you might search for in an the IR context. The problem of measuring the degree of similarity between two sets of annotations, A and B, where A and B are the annotations marked by two different scorers, is naturally formulated in an IR context as:

$$\text{precision}(A, B) = \frac{\#\textit{annotations in } A \textit{ with an edge in } G \textit{ to some annotation in } B}{\textit{total annotations in } A}$$

$$\text{recall}(A, B) = \frac{\#\textit{annotations in } B \textit{ with an edge in } G \textit{ to some annotation in } A}{\textit{total annotations in } B}$$

where the graph G is defined as in the previous section. In a document retrieval system, precision measures the fraction of documents returned in response to a query that are in fact relevant to the query, and recall measures the fraction of all relevant documents in the repository that are successfully returned by the query. In the context above, we regard the set B of annotations as the relevant documents (i.e., as if they were scored by a complete expert), and we are measuring the quality of the set A with respect to B. Due to the interpretation of B as ground truth, these measurements should be appropriate to use for expert-novice comparison as well as for eventually comparing the annotations output of a machine learning algorithm to our gold standard dataset.

Note that the definitions of precision and recall above are completely symmetric if we exchange the roles of A and B. If we want a single number that serves as an undirected measurement of similarity between two sets of annotations (so it does not matter if we compare A against B or vice versa), the option we find most is the F-score the harmonic mean of precision and recall:

$$\text{F-score}(A, B) = \frac{1}{1/\text{precision}(A, B) + 1/\text{recall}(A, B)}$$

F-score is perhaps the most common method used in the IR literature to distill precision

and recall down to a single number reflecting agreement between A and B. [64].

Kappa statistics are a widely used measure to assess the inter-rater agreement of some number of subjects assigning dichotomous, ordinal, nominal (our classifications fall under this category), or Likert-like values to data. There are different formulations of the Kappa statistics for various scenarios. In our case, when we are comparing the scorers to each other in a pairwise fashion, we use Cohen's Kappa [12]. In the case of accessing overall agreement over all of our scorers, we use the Fleiss Kappa [36]. The values of Kappa scores are generally regarded to fall into the following ranges: < 0 as indicating no agreement, $0 - .20$ as slight, $.21 - .40$ as fair, $.41 - .60$ as moderate, $.61 - .80$ as substantial, and $.81 - 1$ as almost perfect agreement [59].

3.3.2 First Trial, Subjects 1-40

In February of 2009, the scorers annotated and classified 30-second clips from 40 subjects. In this first study, the scorers marked 3810 annotations. The breakdown of how many annotations there were for each scorer is shown in Table 3.1. Note that there was both a high outlier and a low outlier. The F-score of the agreement of the scorers is show in Table 3.2.

Scorer	1	2	3	4	5	6	7	Mean
	632	786	320	693	216	729	434	544

Table 3.1: Total number of annotations per scorer and the mean.

Scorer	2	3	4	5	6	7
1	0.62	0.70	0.69	0.56	0.66	0.63
2		0.55	0.71	0.44	0.70	0.53
3			0.62	0.56	0.58	0.62
4				0.49	0.73	0.61
5					0.43	0.62
6						0.55

Table 3.2: Annotation agreement between scorers, shown as a table of F-scores.

Using the first-generation clustering algorithm, these annotations were clustered into 828 clusters. As described above in Section 3.1, the scorers were presented with the output of the clustering algorithm and asked to classify each cluster center. The overall results for how many of each class each scorer classified is shown in Table 3.3.

In this first time we conducted the classification study, we had a total of four classifications,

Classification	1	2	3	4	5	6	7	Totals
Artifactual	363	350	359	335	390	374	425	2596
Normal Electrocortical	410	383	396	273	360	396	397	2615
Abnormal Electrocortical Non-Epileptiform	30	73	42	153	54	40	1	393
Abnormal Epileptiform	25	22	31	67	24	18	5	192

Table 3.3: Classification counts for all classifications, subjects 1-40.

the fourth being "Abnormal Electrocortical Non-Epileptiform". Due to inconsistent application of this class of PEEs, we dropped it in subsequent classification studies. The scorers demonstrated moderate to substantial agreement in classifying with all 4 classes, as shown in Table 3.4.

Scorer	2	3	4	5	6	7
1	0.7695	0.7712	0.6099	0.7458	0.7532	0.7235
2		0.7339	0.6293	0.7150	0.7514	0.6448
3			0.5830	0.7061	0.6939	0.6781
4				0.5849	0.5719	0.5063
5					0.7461	0.6747
6						0.6883

Table 3.4: Pairwise Cohen of scorer classifications, Datasets 1-40.

The overall agreement using all four classes is substantial, as the Fleiss kappa score was 0.67597. An interesting "mile high" visualization of this agreement across all classifications is included in Figure 3.3. When the classifications are interpreted as only epileptiform or not, agreement drops a bit as illustrated by Table 3.5, where the pairwise Cohen scores indicate agreement ranging from poor to substantial. The Fleiss kappa score of 0.43439 (moderate agreement) upholds this as well.

Scorer	2	3	4	5	6	7
1	0.7591	0.6305	0.4088	0.6425	0.4990	0.3266
2		0.5132	0.4031	0.5081	0.5390	0.3641
3			0.3547	0.5677	0.3914	0.2702
4				0.4375	0.3300	0.1291
5					0.5117	0.3382
6						0.4294

Table 3.5: Pairwise Cohen of scorer classifications, epileptiform or non-epileptiform, only subjects 1-40.

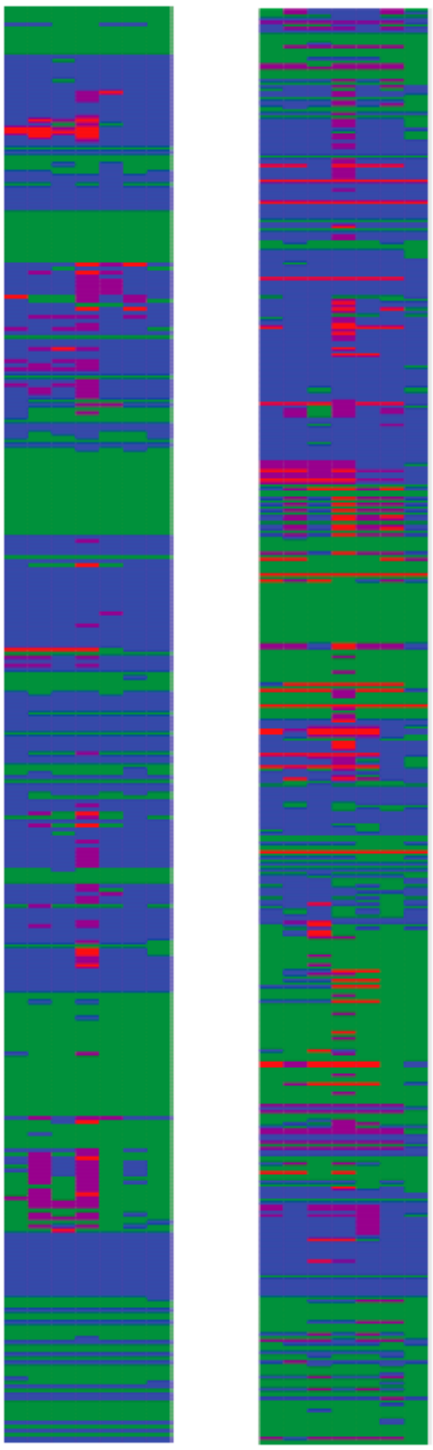


Figure 3.3: All classifications by scorers. A row in this matrix represents an annotation. A column represents a scorer. The classifications map to the following colors: green = artifactual, blue = normal electrocortical, red = abnormal electrocortical epileptiform, and purple = abnormal electrocortical non-epileptiform

3.3.3 Second Trial, Subjects 1-100

In late April of 2010, the scorers annotated and classified 100 subjects, the first 40 of which were the same as before, with 60 new subjects. In this second study, the scorers marked 9319 annotations. The breakdown of how many annotations there were for each scorer is shown in Table 3.6. Note that the low outlier is still a low outlier. Using the new algorithm, these annotations were

Scorer	1	2	3	4	5	6	7	Mean
	1191	1039	1622	1439	494	2233	1301	1331

Table 3.6: Total number of annotations per scorer and the mean.

clustered into 3452 clusters. The numbers of clusters along with how many scorers marked them is shown in Table 3.7. The F-score of the agreement of the scorers is shown in Table 3.8.

Cluster Sizes	1	2	3	4	5	6	7
	881	829	527	388	308	274	245

Table 3.7: Number of clusters of size 1...7.

Scorer	2	3	4	5	6	7
1	0.55	0.60	0.64	0.48	0.54	0.55
2		0.55	0.62	0.39	0.45	0.51
3			0.66	0.40	0.56	0.54
4				0.44	0.65	0.60
5					0.36	0.45
6						0.48

Table 3.8: Annotation correlation between scorers, subjects 1-100.

3.3.4 Datasets 1-40, Intra-Rater Reliability

In light of the fact that we had the scorers pass through the datasets twice (subjects 1-40, in both cases), we could assign a measure of intra-rater reliability. In this sense, we are seeing if the scorers are seeing the same PEEs in the data that they saw a year before. These results are shown in Table 3.9 using F-scores.

Scorer	Reliability
1	0.69
2	0.60
3	0.62
4	0.74
5	0.58
6	0.72
7	0.55

Table 3.9: Intra-rater reliability over the scorers, from their two annotation passes in subjects 1-40.

Chapter 4

eegNet, A Web-Based Software Platform

The purpose of this chapter is to review the design and function of the components of our software infrastructure for rEEG analysis. An earlier version of the software will be summarized and contrasted against the current system to communicate lessons learned from the development and evolution of the system. Following that summary, we will explore the new software platform and how it helps streamline every aspect of our rEEG research.

4.1 First Version

4.1.1 System Description

Our original application was a Windows binary that downloaded data over the internet from a database at Clemson University and allowed the scorers to annotate and classify those annotations (see Figure 4.1). It was based on previous work by the author in a TabletPC based system for collaboration on generic time series data [56]. This front-end application was written in C# for execution in the Windows .NET 3.5 environment, and hence only ran on a Windows operating system.

The software allowed for typical tasks involved with viewing rEEG, such as adjusting the gain of the EEG data and changing montages. Interpreters could annotate the location of paroxysmal

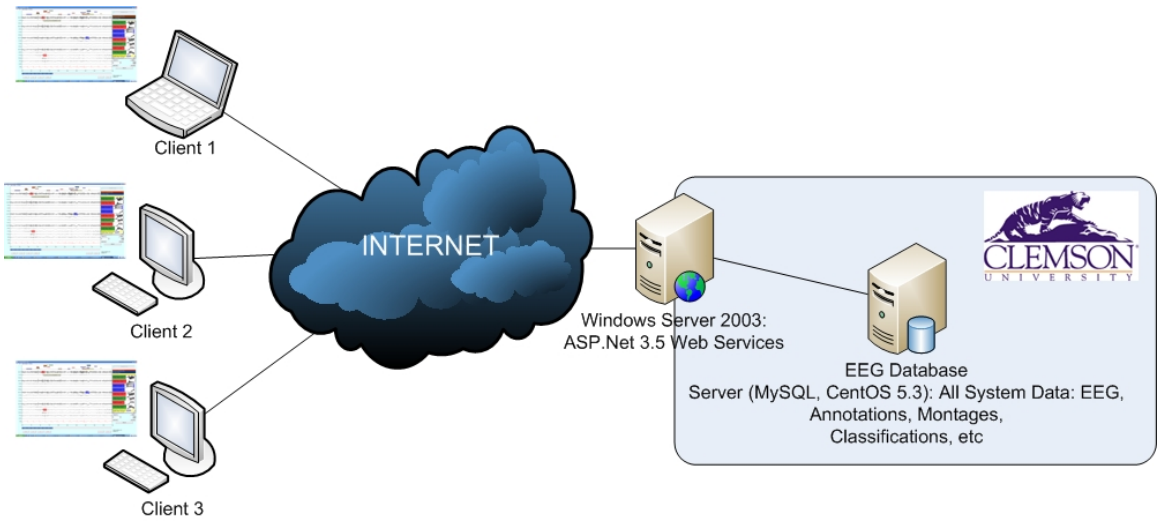


Figure 4.1: The original architecture of the backend for the scoring software system. The only change for the current system is that the Web server is now a CentOS 5.4 Linux system. This is done to prevent licensing costs.

activity in multiple montages and classify it as belonging to one of four categories: artifactual, normal electrocortical, abnormal electrocortical epileptiform, or abnormal electrocortical nonepileptiform. The system had two operating modes:

1. Selection Mode, where scorers worked independently, marking and classifying single-channel EEG signal fragments (see Figure 4.2).

2. Classification Mode, where scorers annotated independently and then classified the same sequence of randomly selected, uniqueEEG signal fragments combined with some redundant EEG signal fragments to test intra-scorer reliability (see Figure 4.3).

In addition to the Windows client, we developed a set of interactive web-based tools for visualization, analysis and administration. These web tools were very primitive, and of limited utility – but they provided the basis for the current system. The tools included were:

1. The user permissions system, shown in Figure 4.4, this system allowed the administrator to set permissions for the scorers on the different datasets of subjects. A drawback of this initial design is that it only allowed a scorer to work in one dataset of subjects in a certain mode.

2. The first-generation clustering algorithm selections viewer, shown in Figure 4.5, was a simple view of the output of the clustering algorithm where a yellow box highlighted a cluster center

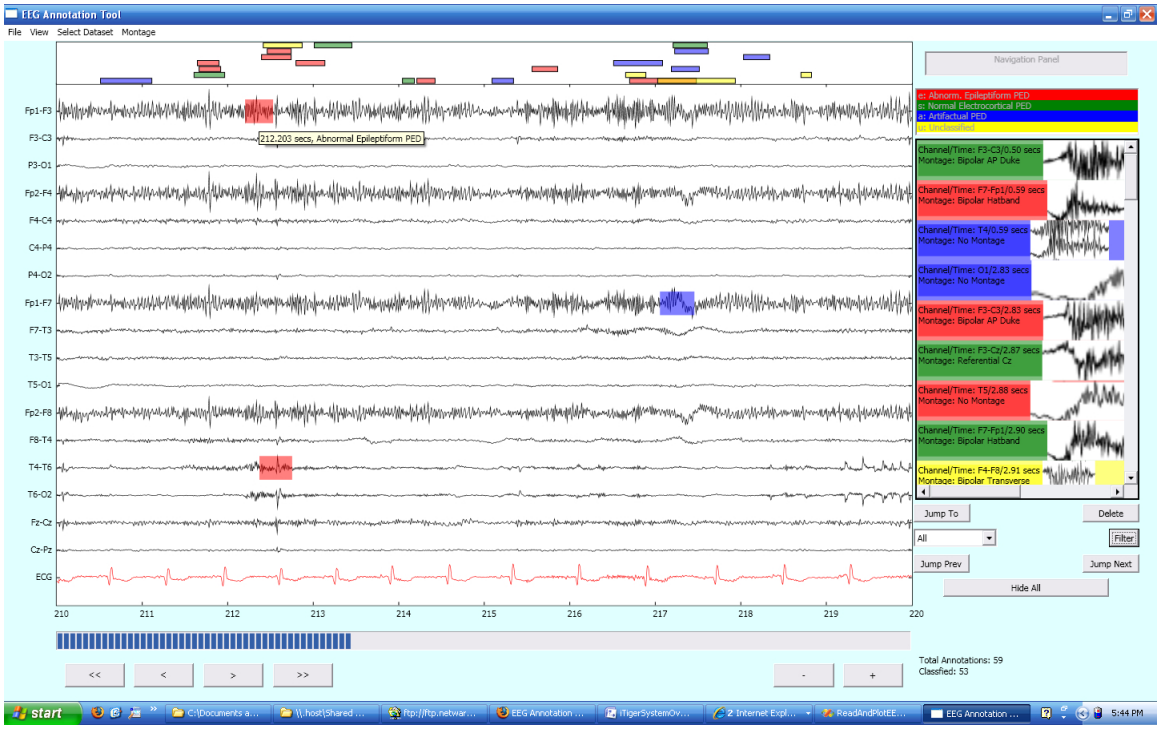


Figure 4.2: The main Windows-based client in Phase 1 (Selection) mode. Note the preview pane on the right. This pane allowed the user to see a thumbnail preview of the annotation (these previews are created at annotation time). To get to any point in the dataset, the scorer would click on one of the annotations in this panel. This panel was ultimately of little use as it took up too much "screen real estate" and was phased out by the new, Web 2.0 version.

and a blue box simply indicated non-selected. This view did not communicate why the clustering decision was made. The process for using this system was to have the administrator run the clustering algorithm and manually upload it to this visualization.

3. The cluster correlations viewer, shown in Figure 4.6, was a visualization we built to help us understand why certain clustering decisions were made, and to prove to us that the clustering algorithm was doing something reasonable. Clicking on a row in the left column would bring up all annotations overlapping the selected one in the right column. Green annotations are above the clustering threshold, Δ, while red are below it. These thumbnails were not terribly informative, as they had no context, and the user of the system could not resize them.

4. The first-generation classification comparison visualization, shown in Figure 4.7, captured a great deal of information about the first classification pass. It featured a matrix of all

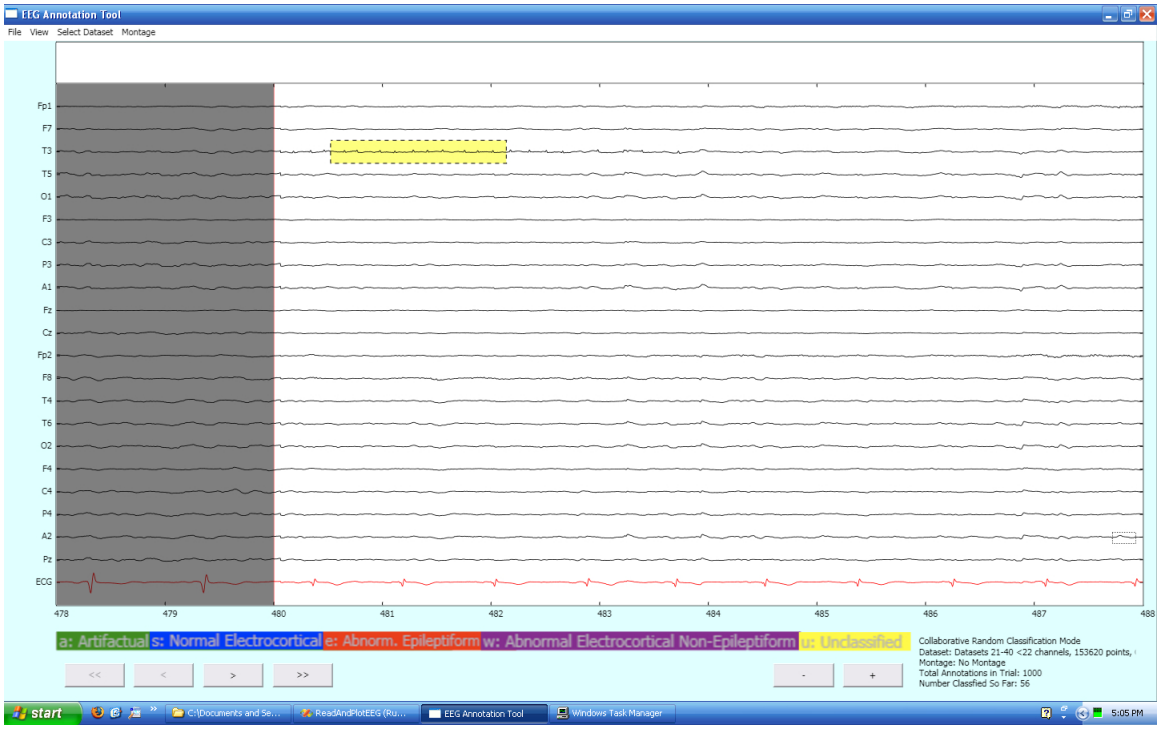


Figure 4.3: The main Windows-based client in Phase 2 (Classification) mode. The scorers were presented the classifications in a random order from the dataset of subjects.

of the classifications in which a row was a single classification and a column was a single scorer. This allowed the administrator to easily view classifications that were contentious. The summary statistics were naïve pairwise percent-agreement scores. This was a problem because the metrics were not symmetric because they assumed that one of the scorers was the "ground truth".

4.1.2 Problems with the First Version

The first large problem we observed with using the Windows-based front-end software platform was that change management and version management of the scorer's Windows executables quickly became a problem. When a bug would be detected, back end changes often could not be made until all of the users upgraded to the new code base. Also, the version of the .Net platform required the users to make a one-time, lengthy .Net 3.5 version installation. Using Windows Server in the backend also meant a significant licensing cost for anyone wishing to deploy this system in the future.

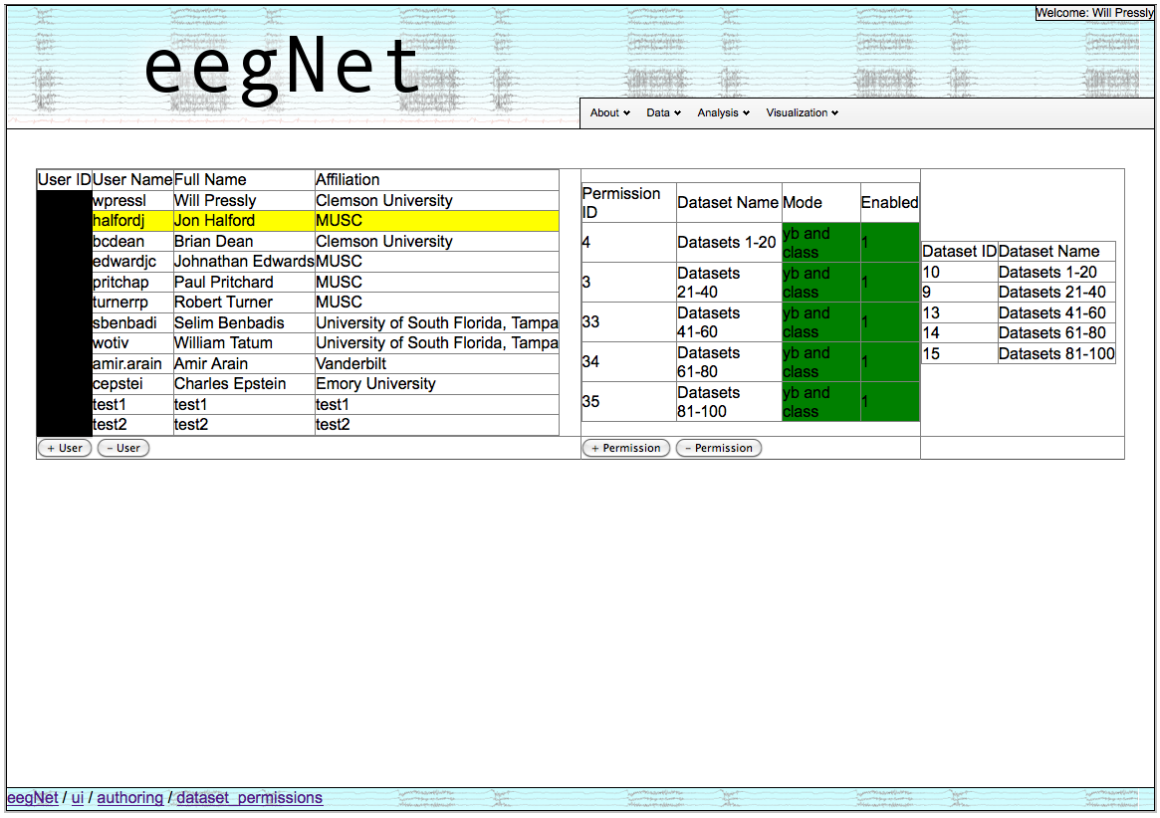


Figure 4.4: The main permissions system. This allowed a user to use a particular dataset in one particular mode only.

The analysis and visualization tools mentioned above were fairly primitive and narrowly focused. For the most part, they did not integrate with each other and all resided in different web pages across several web servers. The decentralized nature of the tools made them cumbersome to use.

Lastly, as the analytical needs of our system grew and evolved, this system was shown not to be very robust. One of the single largest problems we saw with this system occurred when we decided that the notion of intra-rater reliability (see Section 3.3.4) was interesting to explore. The database design at the time supported no method of conducting separate studies that involve the same datasets. For example, it was difficult to answer questions such as "how are the annotations from the first annotation phase in 2008 differentiated from the annotations in subsequent studies?", or "how do we perform analysis and visualization on just one study?". Further, if the administrator of the system wanted to conduct two different studies at once, it was impossible to distinguish the

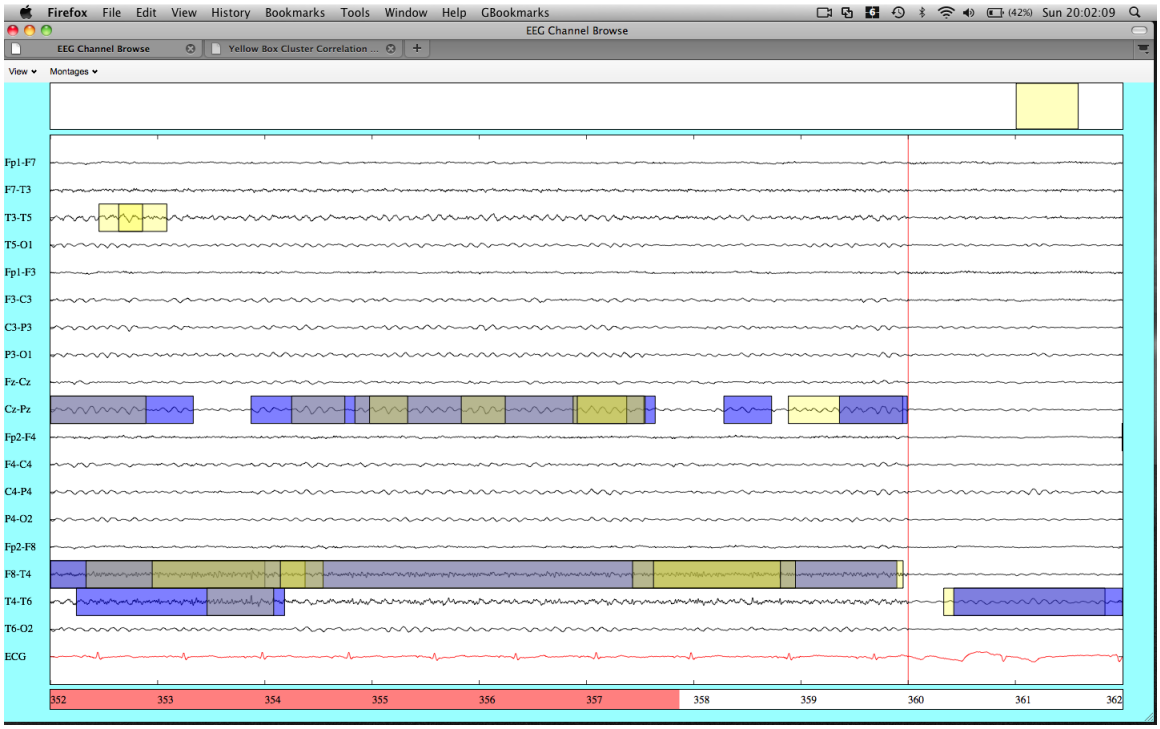


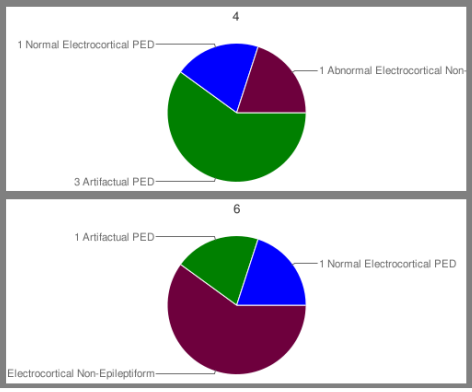
Figure 4.5: The first-generation view of the annotations selected by the first-generation clustering algorithm. The administrator of the system would run the clustering algorithm and manually upload the result into this system for visualization.

Image	Ann ID	Position	Duration	Gain	Montage Name	Channel Name	UserID	Correlation Score
	8422	354.8398	2.6953	-9	Bipolar AP Typical	Cz-Pz	8	ORIGINAL
	8464	353.8711	3.7695	-7	Bipolar AP Typical	Cz-Pz	6	1.00000
	6601	356.8750	0.5000	-25	Bipolar AP Typical	Cz-Pz	11	1.00000
	6600	355.8320	0.4102	-25	Bipolar AP Typical	Cz-Pz	11	1.00000
	6599	354.9766	0.3555	-25	Bipolar AP Typical	Cz-Pz	11	1.00000
	6015	356.9141	0.6328	-23	Bipolar AP Typical	Cz-Pz	12	0.99655
	8416	350.0469	9.8477	-9	Bipolar AP Typical	F8-T4	8	0.33805
	6602	354.1445	5.8086	-25	Bipolar AP Typical	F8-T4	11	0.33805
	8465	357.4297	1.5195	-7	Bipolar AP Typical	F8-T4	6	0.06037

Figure 4.6: The first-generation cluster viewer. It used thumbnails captured at annotation-time to preview the annotations.

Start	Duration	Montage	Channel	4 ☐	6 ☐	Only Selected / Show All	Classification Consensus
				99.62%	99.62%		
0.97	0.66	Bipolar AP Typical	T3-T5				50.00%
2.71	2.20	Bipolar AP Typical	T3-T5				50.00%
5.49	0.64	Bipolar AP Typical	Fp2-F8				100.00%
6.73	1.36	Bipolar AP Typical	T4-T6				100.00%
7.57	1.48	Bipolar AP Typical	T3-T5				100.00%
					Average:		80.00%
Breakdown by PED Type							
Average for Artifactual PED PED Type:							66.67%
Average for Normal Electrocortical PED PED Type:							100.00%
Average for Abnormal Electrocortical Non-Epileptiform PED Type:							100.00%
Breakdown by Montage							
Average for Bipolar AP Typical Montage:							80.00%
Breakdown by Channels							
Average for T3 Channel:							66.67%
Average for T5 Channel:							66.67%
Average for Fp2 Channel:							100.00%
Average for F8 Channel:							100.00%
Average for T4 Channel:							100.00%
Average for T6 Channel:							100.00%

PEDTypes by User



Legend
Artifactual PED
Abnormal Epileptiform PED
Normal Electrocortical PED
Unclassified
Abnormal Electrocortical Non-Epileptiform

User Vs User Agreements (All)

	4	6
4	5/5 (100.00 %)	3/5 (60.00 %)
6	3/5 (60.00 %)	5/5 (100.00 %)

User Vs User Agreements (Artifactual PED)

	4	6
4	3/3 (100.00 %)	1/3 (33.33 %)
6	1/1 (100.00 %)	1/1 (100.00 %)

User Vs User Agreements (Normal Electrocortical PED)

	4	6
4	1/1 (100.00 %)	1/1 (100.00 %)
6	1/1 (100.00 %)	1/1 (100.00 %)

User Vs User Agreements (Abnormal Electrocortical Non-Epileptiform)

	4	6
4	1/1 (100.00 %)	1/1 (100.00 %)
6	1/3 (33.33 %)	3/3 (100.00 %)

Figure 4.7: The first-generation scorer comparison view. The agreement numbers in this view were all naïve percent agreements.

annotations of one study from another. It became apparent that this was a large design flaw in the original system.

4.2 Current Version

We learned a great deal in the process of developing the first generation of tools. In rewriting the tools, it became clear that all of the functionality of the Windows client could be provided using new Web standards, and that this would be a compelling change from the Windows application model. Accordingly, the current version of this system is largely a Web 2.0/Rich Internet Application (RIA) and leverages the languages of open standard Web development (Javascript, HTML, CSS, PHP). There are also back-end applications that are written in C and linked to the MySQL database backend. A count of the lines of code for the current version in each language is in Table 4.1. This move to the Web vastly simplifies many of the deployment and versioning challenges that we

Language	Count
javascript	11686
php	5144
c	3187
css	270
html	137
PERL	23
Total	20447

Table 4.1: Composite lines of code of whole system, latest version. This does not include prior versions, or scrapped earlier designs.

experienced in the Windows version since the back end and the front end are deployed in the same fashion over the web. This is a vast improvement over an independent windows binary that might be incompatible with the current back end. Access to the latest code changes now require nothing more than a browser page refresh.

A design objective of this new software was that it should be a well integrated system – that the administrator or scorer did not have to go to many different places to use the system. Further, one of the most important lessons of that development cycle was that "screen real estate is king" – meaning that visually inspecting these large and noisy datasets requires that as much of the interface as possible is dedicated to rendering the rEEG data. This data-centric design quickly became an ethos that defined the rest of the tool set. The whole system is designed now to quickly provide rich

and meaningful data in an easily accessible way over the Web.

The last issue mentioned in the previous section was the problem that the first-generation software does not support the concept of different studies. In the new version of the software, we introduce the idea of a *trial* to map to the concept of a study. A trial is an abstraction that represents a set of scorers making a pass through a set of subjects in a certain mode. This abstraction can be easily implemented in the database by adding intermediate many-to-many tables that map datasets, annotations, classifications, and users to different trials, as illustrated in the latest entity relationship diagram (ERD) in Figure 4.8. This allows for many users to participate in several trials using the

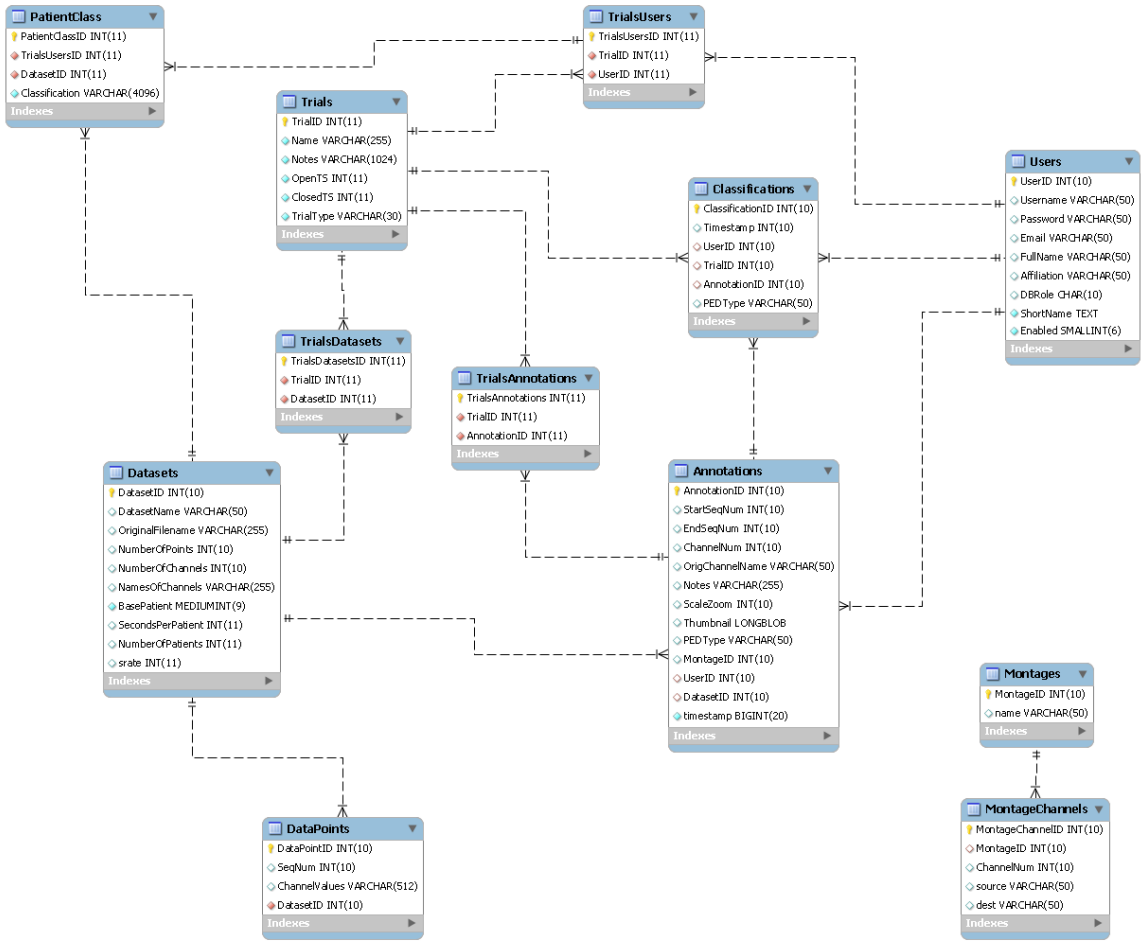


Figure 4.8: The current design of the database for the annotation software. The abstraction of trials allows for users to mark annotations in different datasets in different modes and different studies.

same or different datasets in different modes. The operations an administrator can perform on a trial are the following: create the trial; assign the trial properties (mode,users, datasets, and annotations);

open a trial; close a trial; and delete a trial. This system of trials overcomes all of the problems mentioned in the previous section and further allows for better organization and segregation of data and access to appropriate data by the analysis and visualization tools.

4.2.1 Annotation Functions

The new software is shown in Figure 4.9. The function of the annotation software still allows the user to draw a box around an interesting region of data, and to classify it into the three categories mentioned in Section 3.1. There are four different modes supported by the software:

1. **Annotation Mode**. In Annotation mode, the scorers can only create annotations in a particular trial. They cannot classify them as anything other than unclassified (yellow). The focus of this type of trial is solely to mark rEEG signals that the scorers think are PEEs – not to make a judgement about what the PEE represents.

2. **Classification Mode**. In this mode, all scorers in the trial are presented the same set of classifications (starting in the unclassified, yellow state). The scorers select an annotation by clicking on it, and then press a keystroke to classify it into one of the 3 categories mentioned in Section 3.1. The users cannot delete or create annotations.

3. **Annotation and Classification Mode**. This mode is the same as Annotation Mode, with the exception that a scorer can classify it as they create the annotations.

4. **Patient Classification Mode**. In this mode, the scorer classifies the entire 30 second clip of the patient as one of the categories mentioned in Section 3.1.

There are several new features of the annotation software, as well. A new addition to the software is that it now has better support for random access through the dataset of subjects. This function is accomplished by adding a "playback bar" in the lower portion of the screen that jumps to the appropriate position in the data. This can be seen in Figure 4.9. Note that the preview panel that was on the right in the original client is gone as it is deprecated in function by the playback bar, and to make more room for the rEEG viewer. The annotation client also has built in documentation that hovers over the rEEG viewer for a "quick reference"; again, this can be seen in Figure 4.9. Individual channel gains can be set in the rEEG viewer.

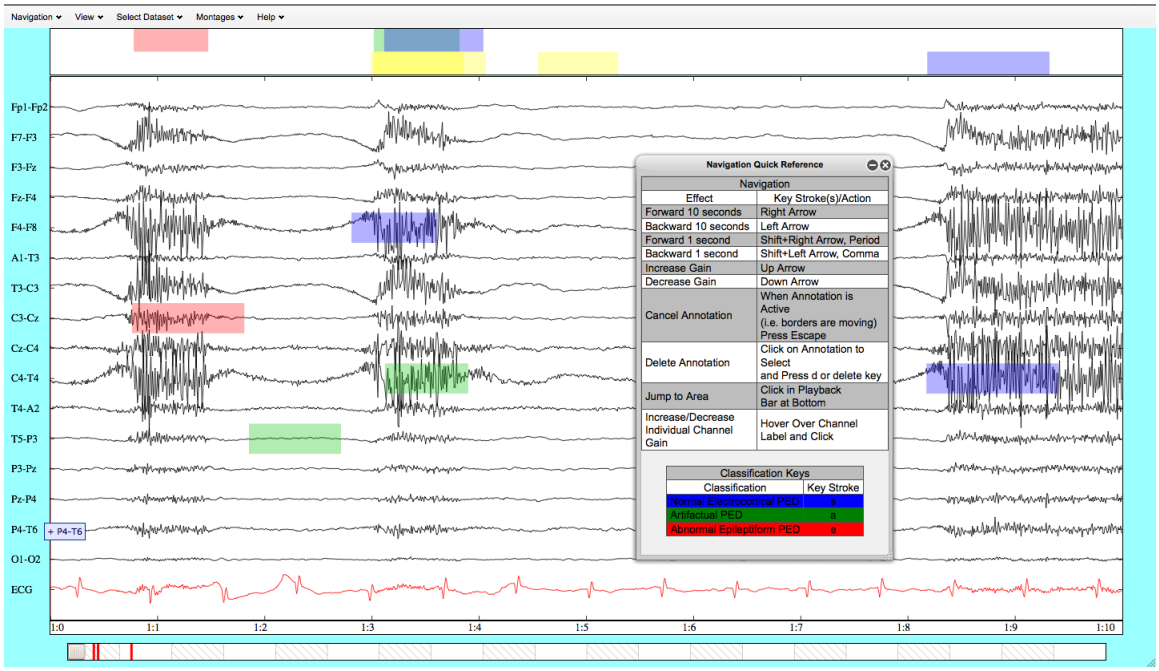


Figure 4.9: The current annotation client, based on open web standards, including HTML5, Javascript, and CSS.

4.2.2 Administration Functions

The administrative functions of the system are all now contained in one page on four tabs. The tabs for each of the administrative functions are shown in Figure 4.10 and include:

1. **The summary tab** (upper left in Figure 4.10) provides a summary of all of the annotations, datasets of subjects, users, etc. on the system. On this summary, an administrator can view a "mile-high" view of the trial that can be used to gauge the progress of the scorers through the trial. For an example, see Figure 4.11.

2. **The datasets maintenance tab** (upper right in Figure 4.10) allows the administrator to add, edit, and delete datasets.

3. **The trials maintenance tab** (lower right in Figure 4.10) allows the user to add, edit, and delete trials. Trial creation is more complicated than user or dataset creation because certain system modes (described in Section 4.2.1) take different inputs. For example, a classification mode trial takes as input a saved clustering analysis (described in the following section) to get the list of annotations included in the trial. This interface handles all of these cases in a

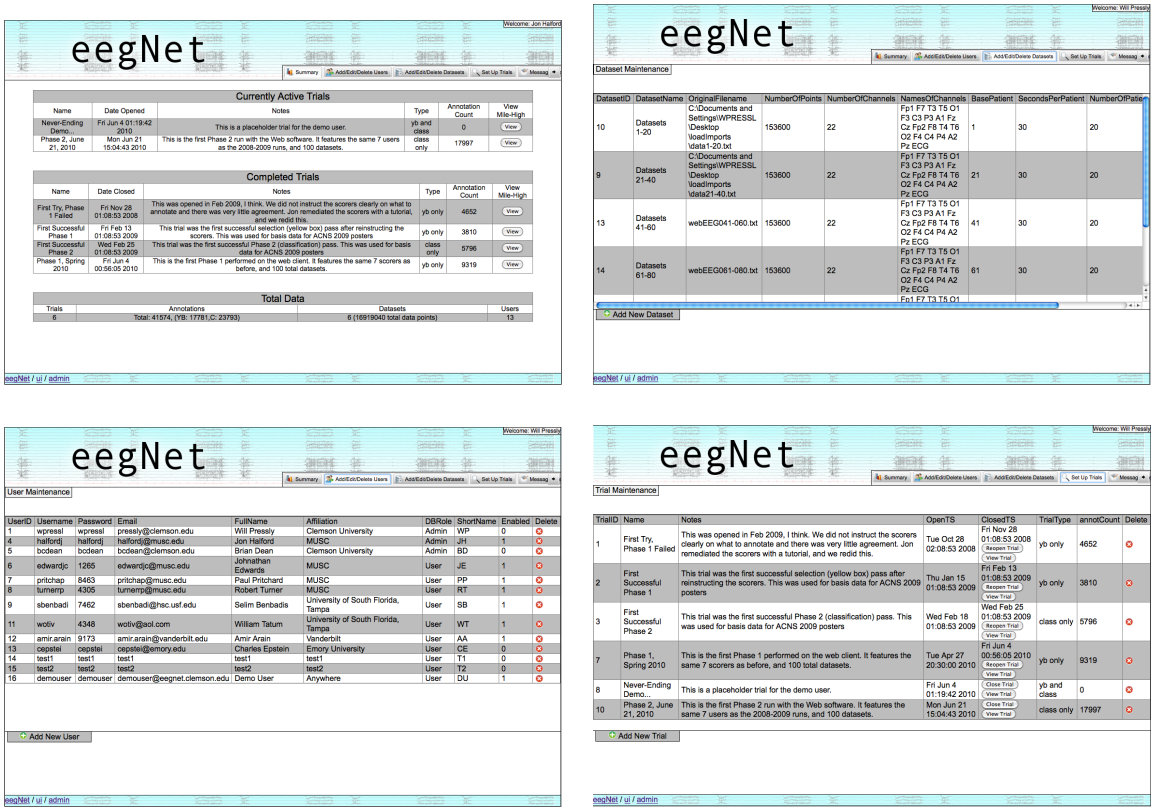


Figure 4.10: Administrative screens. Clockwise from top left: (1) Summary information on trials and datasets along with quick overviews, (2) dataset administration, (3) user administration, and (4) trial administration.

seamless fashion.

4. **The users maintenance tab** (lower left in Figure 4.10) allows the administrator to add, edit, and delete users.

4.2.3 Visualization And Analysis Functions

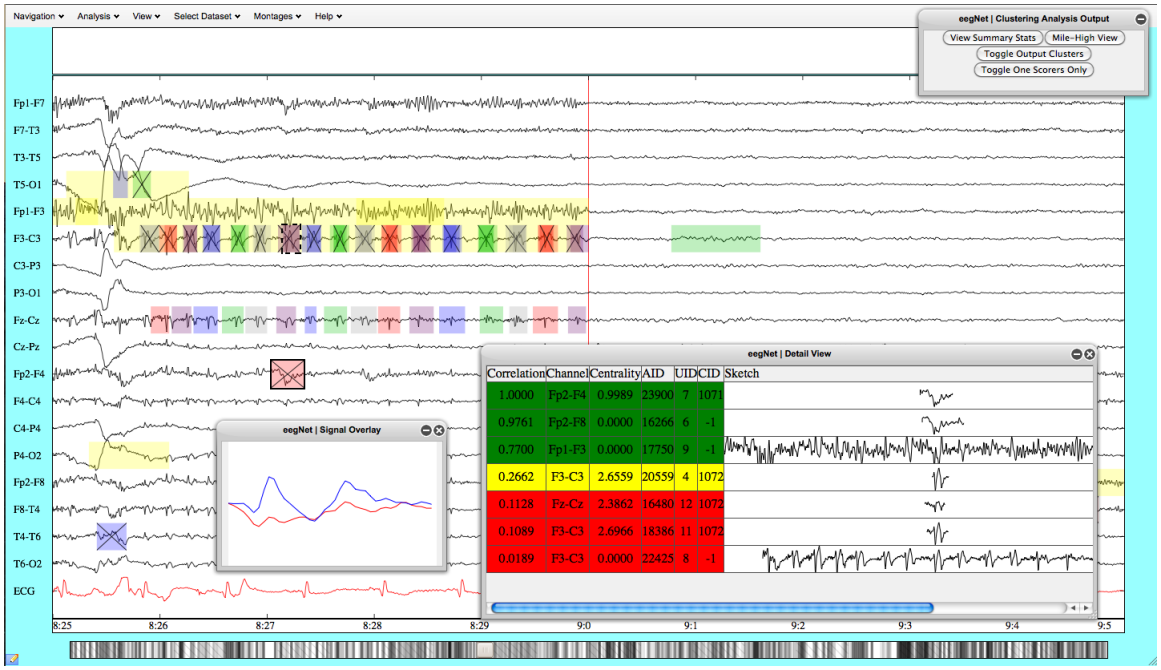
The visualizations included in eegNet are highly interactive and informative. They currently include the following.

The clustering visualization. In this visualization (top of Figure 4.12), yellow boxes are multiply-included annotations, while all other colors represent a single cluster. An X over the annotation means that it is a cluster center. Clicking on any annotation will bring up the correlation detail window, where the administrator can explore why the clustering decision was made based on

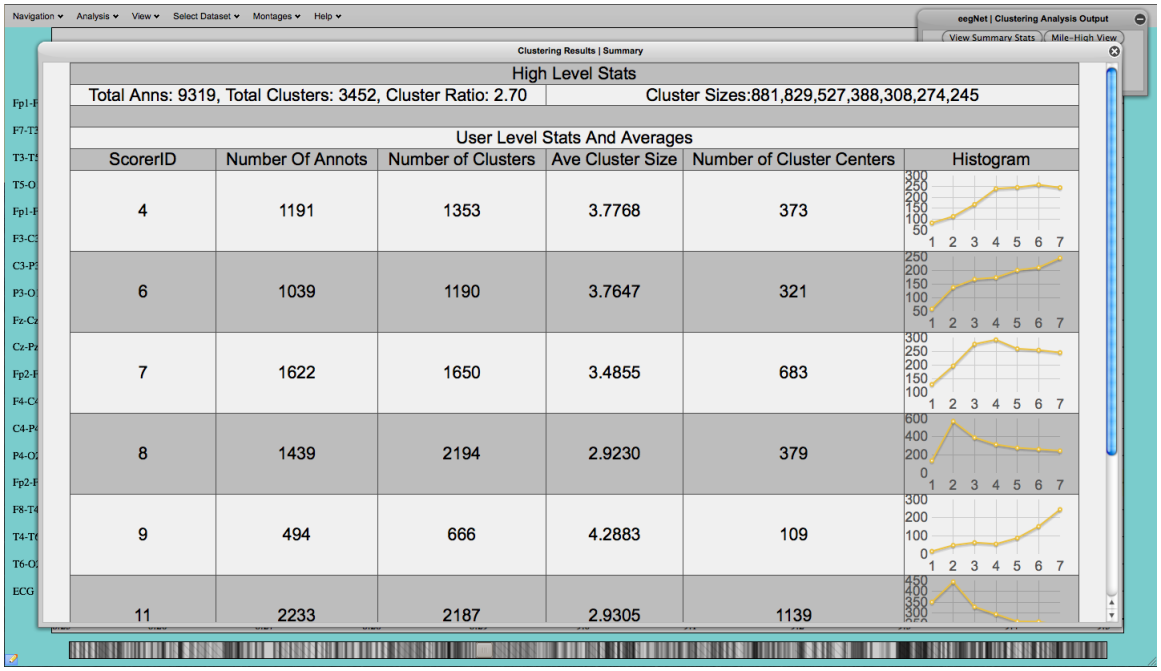
Figure 4.11: The progress viewer in the administrative tool. This is taken at the start of a new classification pass (The scorer's annotations have been clustered and presented back to them for classification).

the pairwise correlations of all of the annotations that overlap it, and their centralities. Clicking on the top row of this detail view will bring up a view where two waveforms may be overlaid (the currently selected annotation from the detail window and whatever row the mouse is over). Summary statistics on the clustering result can be show from a panel that overlays the rEEG signal. The summary dialog can be seen in the bottom of Figure 4.12. Statistics it tracks are: number of clusters (total and per scorer), number of annotations (total and per scorer), average cluster sizes of users, and scorer-cluster participation histograms. A clustering result can be created in an interface on this page by specifying a trial, the users and datasets of subjects to include from the trial, and the parameters for the clustering Δ and Ω. The clustering algorithm is then invoked on the web server and the clustering result is rendered in the browser. A clustering result can also be saved for later viewing, or loading into a classification trial.

The classification analysis tools and visualizations. As seen in the top of Figure 4.13, the classification analysis tool shows the classification Kappa scores for an input annotation trial. The visualization in Figure 4.14 overlays the classifications of all of the scorers in the trial over the



The cluster viewer.

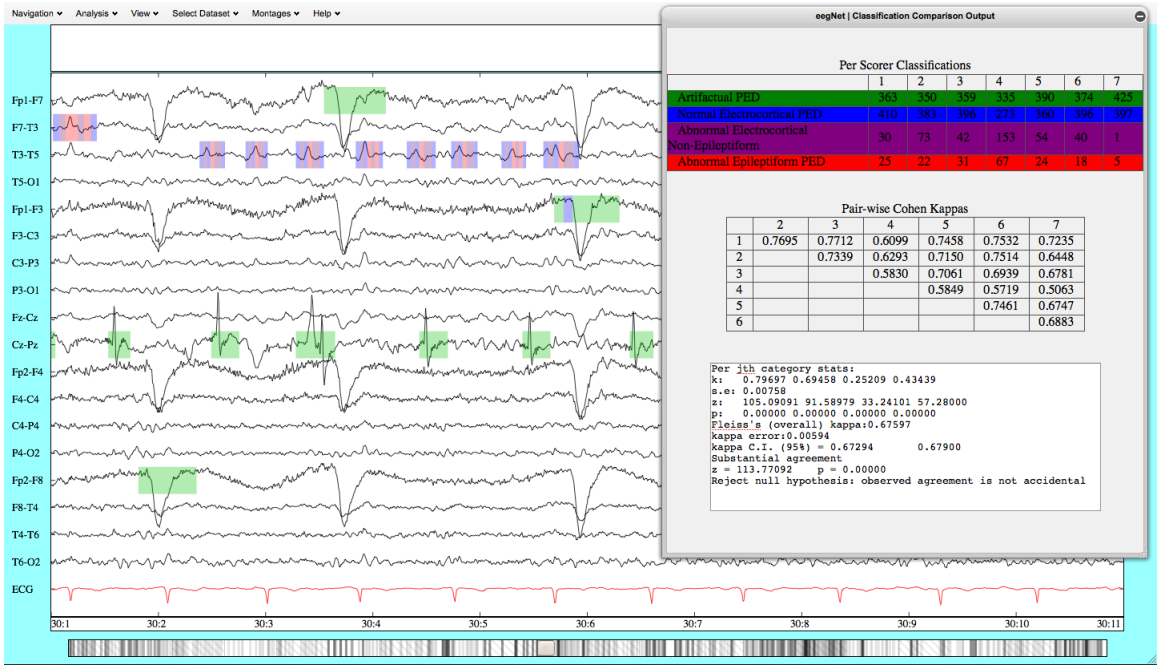


Cluster summary data.

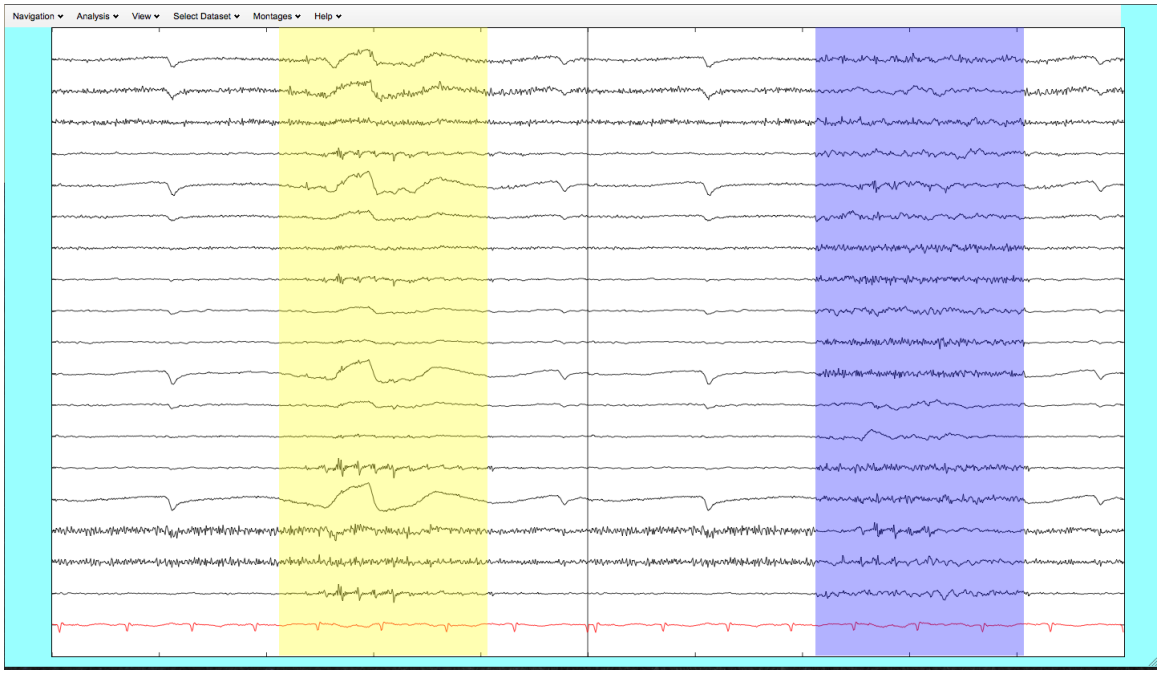
Figure 4.12: The clustering visualizations.

original annotation. This view allows for easily discovering and analyzing contentious classifications, as shown in the figure.

The split-view ICA visualization. Shown in the bottom of Figure 4.13, this view features a screen split into two halves, each showing the same region of the signals, with the exception that the blue box shows the independent components of the yellow box. When the yellow box is highlighted, the browser requests the back end to perform Renyi's entropy ICA on that region of the signal and return the result.



The classification comparison tool. It shows the how many classifications of each type the scorer classified, the pairwise Cohen's kappa scores of agreement, and the overall Fleiss kappa agreement. It also allows quick access to the most problematic classifications.



The ICA split-screen viewer. The screen is split into two halves, each showing the same region of the signals,with the exception that the blue box shows the independent components of the yellow box.

Figure 4.13: Analysis and Visualization tools

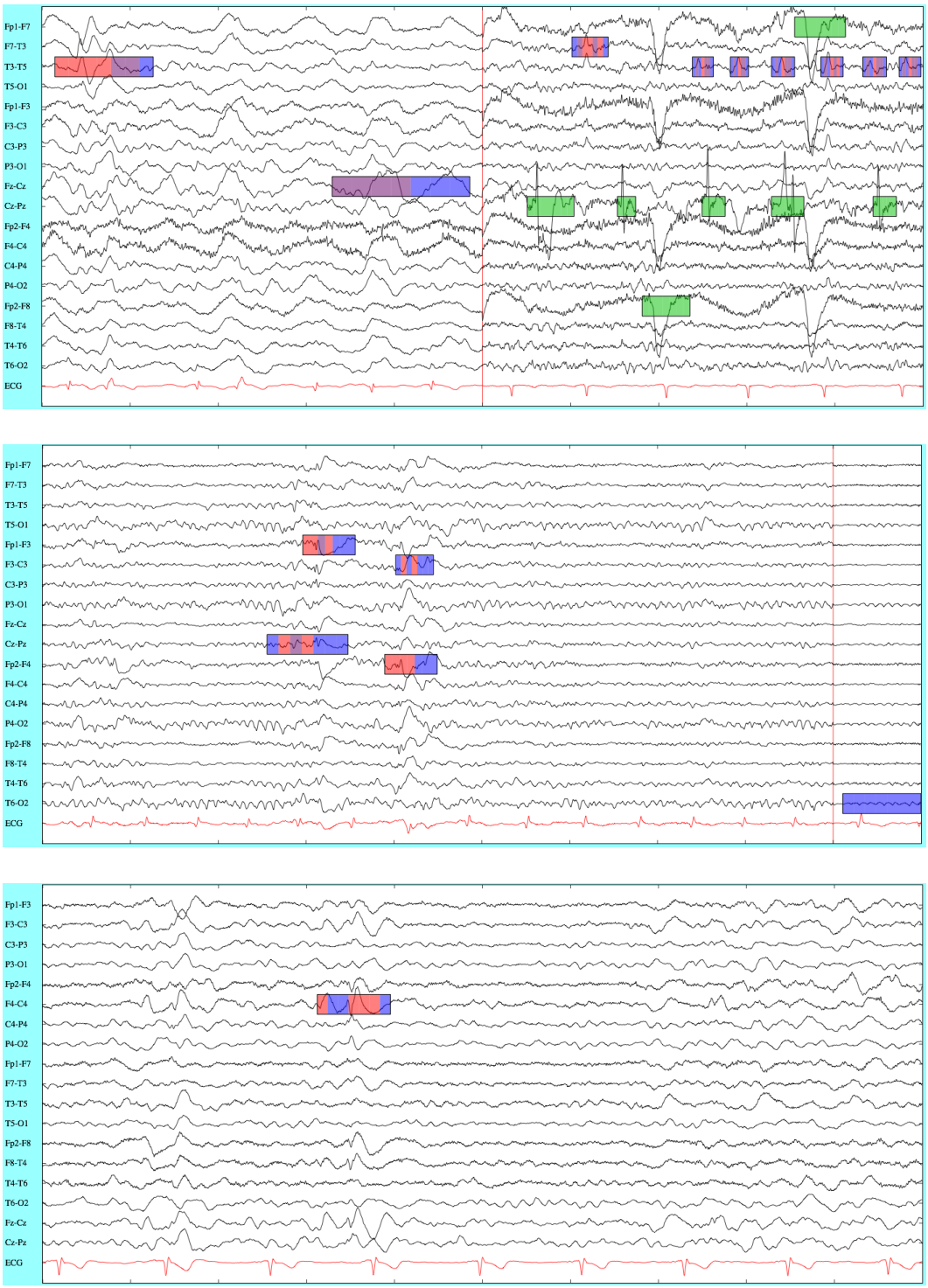


Figure 4.14: Some contentious PEEs. In this overlay, each annotation is colored in strips according to how each scorer classified it. These annotations are particularly interesting in that the scorers were evenly split on if the PEEs were epileptiform or not.

Chapter 5

A GPU implementation of ICA based on Renyi's Entropy

In this chapter we will discuss Renyi's entropy ICA, an algorithm that has been shown to produce some of the best results when compared to other ICA algorithms [52]. We will first explore how to apply Renyi's entropy to a BSSS algorithm. Next, we will introduce some novel algorithmic contributions to improve the base algorithm described in the first section. We will then provide a brief overview of the pertinent architectural and procedural details for using OpenCL to leverage GPUs. Next, we describe the GPU decomposition of the Renyi's entropy algorithm, as described in the first section of this chapter. Lastly, we will review the results of speed testing our Renyi's entropy GPU code against a non-GPU version and Infomax [3].

5.1 ICA based on Renyi's Entropy Overview

Recall the goal of the Renyi entropy ICA algorithm of [87, 52]: given a whitened $N \times L$ matrix X whose N rows contains mixtures of signals, we seek to compute an $N \times N$ orthonormal de-mixing matrix W such that the rows $y_1 \dots y_N$ of $Y = WX$ minimize $f(Y) = g(y_1) + \dots + g(y_N)$, where

$$g(v) = -\operatorname{sgn}(kurt(v)) \log \frac{1}{L} \sum_{j=1}^{L} G(v_j - v_{j-1}, 2\sigma^2).$$

Knowing that the de-mixing matrix W is orthonormal gives us a convenient alternative means of representation: we can write W is a product of $\binom{N}{2}$ rotations along each pair of coordinate axes — a rotation of angle θ_{12} in the plane spanning dimensions 1 and 2, a rotation of angle θ_{13} in the plane spanning dimensions 1 and 3, and so on. If we define the *Givens* rotation matrix R_{ij} as the $N \times N$ identity matrix with its (i, j) submatrix (that is, the 2×2 submatrix containing only the entries in the ith and jth rows and columns) set to

$$\begin{bmatrix} \cos\theta_{ij} & \sin\theta_{ij} \\ -\sin\theta_{ij} & \cos\theta_{ij} \end{bmatrix},$$

then the product $R_{ij}x$ enacts a rotation of the vector x by the angle θ_{ij} within the plane spanned by dimensions i and j. We can now write the de-mixing matrix W as

$$W = \prod_{i<j} R_{ij}$$

for an appropriate set of $\binom{N}{2}$ decision variables θ_{ij}. Our approach to ICA using Renyi's entropy will use this representation for W. Therefore, our goal will be to learn the best possible settings of the θ_{ij} decision variables so that we minimize $f(Y) = g(y_1) + \ldots + g(y_N)$ for $Y = WX$.

The following algorithmic observations will be helpful concerning the R_{ij} rotation matrices defined above.

- If A is any $N \times N$ matrix, we can compute AR_{ij} in only $O(N)$ time by simultaneously replacing the ith column A_i of and the jth column A_j of A with new columns:

  $$A'_i = A_i \cos\theta_{ij} - A_j \sin\theta_{ij} \qquad A'_j = A_i \sin\theta_{ij} + A_j \cos\theta_{ij}.$$

  Similarly, if A is any $N \times N$ matrix, we can compute $R_{ij}A$ in only $O(N)$ time by simultaneously replacing the ith row A_i of A and the jth row A_j with new rows:

  $$A'_i = A_i \cos\theta_{ij} + A_j \sin\theta_{ij} \qquad A'_j = -A_i \sin\theta_{ij} + A_j \cos\theta_{ij}.$$

- The inverse R_{ij}^{-1} is a matrix of the same form as R_{ij}, except we replace θ_{ij} with $-\theta_{ij}$. Hence, we can multiply any $N \times N$ matrix on the left or right by R_{ij}^{-1} in $O(N)$ time.

- The matrix-vector product $R_{ij}x$ takes only $O(1)$ time to compute, as it changes only the ith and jth entries of x to:

$$x'_i = x_i \cos\theta_{ij} + x_j \sin\theta_{ij} \qquad x'_j = -x_i \sin\theta_{ij} + x_j \cos\theta_{ij}.$$

- The derivative (with respect to θ_{ij}) of multiplication by R_{ij} is multiplication by a matrix R'_{ij} of the same form as R_{ij} with θ_{ij} replaced by $\theta_{ij} + \pi/2$, and with all 1's removed from the diagonal. Hence, if A is an $N \times N$ matrix, we can compute AR'_{ij} and $R'_{ij}A$ both in only $O(N)$ time, and moreover these product matrices will only contain two nonzero columns or rows, respectively. In addition, we can compute the matrix-vector multiplication $R'_{ij}x$ in only $O(1)$ time, and the output of this operation will be a vector with only two nonzero entries.

In particular, these facts allow us to compute W by multiplying together all $\binom{N}{2}$ individual R_{ij} matrices in only $O(N^3)$ total time, much faster than one would normally be able to compute a product of a quadratic number of matrices.

At a high level, we minimize $f(Y)$ using gradient descent. Starting with a vector Θ containing our $\binom{N}{2}$ decision variables θ_{ij}, we repeatedly apply the update rule

$$\Theta' = \Theta - \alpha \nabla f,$$

where α is some specified step size. It would be possible to adjust α in an adaptive fashion, however, our implementation simply uses a fixed constant step size of $\alpha = 0.1$. Each component

$$\left.\frac{\partial f}{\partial \theta_{ij}}\right|_Y$$

in the gradient vector ∇f is slightly daunting to write down as a single formula (see [87] for a page-width formula). Hence, we show how to build up this quantity in a step-by-step manner that will also highlight how we construct this quantity in parallel on a GPU. Consider a specific length-n *column* of the original X matrix – say, the pth column, which we denote $x(p)$. We also let $\Delta x(p) = x(p) - x(p-1)$ denote the difference between this column and the previous column. For just this column, we describe shortly how to compute three length-N vectors $a(p)$, $b(p)$, and $k(p)$. These vectors are then respectively summed across all columns $p = 1 \ldots L$ to obtain length-N

vectors a, b, and k, after which we have

$$\left.\frac{\partial f}{\partial \theta_{ij}}\right|_Y = \sum_{r=1}^{N} \operatorname{sgn}(k_r) a_r / b_r.$$

The components of k will represent kurtosis estimates for the signals $y_1 \ldots y_N$, and the components of a and b will represent the numerators and denominators in formula we get by differentiating the Renyi entropy estimator stated above. To compute the vectors $a(p)$, $b(p)$, and $k(p)$ based on $x(p)$ and $\Delta x(p)$, we do the following:

1. Compute $y(p) = Wx(p)$, where $W = \prod_{(u<v)} R_{uv}$ as defined above.
2. Compute $\Delta y(p) = W \Delta x(p)$.
3. Compute $\Delta y'(p) = \nabla_{ij} W \Delta x(p)$, where $\nabla_{ij} W$ denotes the derivative of the linear transformation W with respect to θ_{ij}, computed by taking the product $\prod_{u<v} R_{uv}$ but with the matrix R_{ij} replaced by R'_{ij}.
4. Set $k(p) = [y(p)]^4 - 3$ (that is, we take all entries in $y(p)$ raised to the fourth power, and subtract 3 from them).
5. Set $b(p) = G(\Delta y(p), 2\sigma^2)$ (again, doing all operations component-wise, so we compute a Gaussian function on every component of the vector $\Delta y(p)$.
6. Set $a(p) = b(p) * \Delta y(p) * \Delta y'(p) / \sigma^2$, where $*$ denotes component-wise multiplication of vectors.

This concludes the mathematical description of our implementation of the Renyi entropy ICA algorithm.

5.2 Algorithmic Enhancements

Let us now focus on computational efficiency. Before applying steps 1-6 above, we first compute the matrices W and $\nabla_{ij} W$, both of which require $O(N^3)$ time to compute using the observations above about the special structure of the R_{ij} matrices. Steps 1, 2, and 3 each require a matrix-vector multiply, running in $O(N^2)$ time, and steps 4, 5, and 6, each require only $O(N)$ time. Since these steps are performed for every value of $p = 1 \ldots L$, this gives a running time of $O(N^3 + N^2 L)$ for computing a single entry $\partial f / \partial \theta_{ij}|_Y$ in $\nabla f|_Y$. To compute all $\binom{N}{2}$ of these entries,

it would therefore seem we need to spend $O(N^5 + N^4 L)$ time. However, observe that the work we do in steps 1-6 above for different values of p is more or less independent, so parallelism will help reduce the $O(N^4 L)$ part of the running time.

To further improve the running time above, both for serial and parallel implementations, we propose the following algorithmic enhancements. We note that W only needs to be computed once over the entire process of computing ∇f, so we can afford to do this in $O(N^3)$ time. However, computing each of the $\nabla_{ij} W$ matrices currently takes $\binom{N}{2} \cdot O(N^3) = O(N^5)$ time. To decrease this, we maintain extra state in order to compute each successive $\nabla_{ij} W$ matrix more quickly. Recall that $\nabla_{ij} W$ is obtained by multiplying together the sequence of all R_{uv} matrices, with R_{ij} replaced by R'_{ij}. Let P_{ij} denote the "prefix" of this matrix product up to but not including R'_{ij}, and let S_{ij} denote the suffix of this product after R'_{ij}:

$$\nabla_{ij} W = \underbrace{\begin{bmatrix} R_{12} \end{bmatrix} \begin{bmatrix} R_{13} \end{bmatrix} \cdots}_{P_{ij}} \begin{bmatrix} R'_{ij} \end{bmatrix} \underbrace{\cdots \begin{bmatrix} R_{N-1,N} \end{bmatrix}}_{S_{ij}}$$

Our approach maintains P_{ij}, R'_{ij}, and S_{ij} as we compute each successive term $\partial f / \partial \theta_{ij}|_Y$ in $\nabla f|_Y$. We never need to explicitly multiply P_{ij}, R'_{ij}, and S_{ij} together to obtain $\nabla_{ij} W$, since all we ever do with $\nabla_{ij} W$ is a matrix-vector multiplication $\Delta y'(p) = \nabla_{ij} W \Delta x(p)$ in step 3, which can be written as

$$\Delta y'(p) = \nabla_{ij} W \Delta x(p) = P_{ij}(R'_{ij}(S_{ij} \Delta x(p))).$$

Parenthesized this way, we compute $y'(p)$ in only $O(N^2)$ time (the same as before) but using three matrix-vector multiplies instead of one. However, we can now maintain P_{ij} and S_{ij} efficiently as we loop over all entries $\partial f / \partial \theta_{ij}|_Y$ in $\nabla f|_Y$. Initially, $P_{ij} = I$ and $S_{ij} = W$. In each iteration, we update P_{ij} by multiplying on the right by R_{ij} (which takes only $O(N)$ time), and we update S_{ij} by multiplying on the left by R_{ij}^{-1} (which also takes only $O(N)$ time). Hence, the total time to maintain P_{ij} and S_{ij} is only $O(N^3)$ over the entire computation of $\nabla f|_Y$, lowering our total running time to $O(N^3 + N^4 L)$ (with the $N^4 L$ term subject to further reduction due to parallelism).

We can reduce our running time yet further by carefully examining once again the update

taking place in step 3,

$$\Delta y'(p) = P_{ij}(R'_{ij}(S_{ij}\Delta x(p))),$$

which currently takes $O(N^2)$ time. Suppose we maintain in memory the vector $Sx(p) = S_{ij}x(p)$. To update this in each iteration, we need only left-multiply by R_{ij}^{-1} in $O(N)$ time ($O(N^3L)$ over the entire gradient computation). Hence, let us consider the following modified version of step 3:

$$\Delta y'(p) = P_{ij}(R'_{ij}Sx(p)).$$

Since R'_{ij} is extremely sparse, the multiplication $R'_{ij}Sx(p)$ takes only $O(1)$ time and produces as output a vector with *only two nonzero components*. Multiplying this by P_{ij} therefore only takes $O(N)$ time, since this involves taking a linear combination of only two of P_{ij}'s columns. Hence, we can reduce the total effective running time of step 3 to $O(N)$, or $O(N^3L)$ over the entire algorithm. Steps 1 and 2 still take $O(N^2)$ time, but these only need to be applied once at the beginning of the gradient computation, rather than for every pair (i, j). As a result, we can lower the sequential running time for computing $\nabla f|_Y$ to only $O(N^3 + N^3L)$, where the $O(N^3L)$ part will decompose further due to parallelization.

5.3 OpenCL Overview

OpenCL is an emerging open standard for parallel programming over heterogenous systems [57]. It allows for developers to simultaneously leverage all computing resources on a system, such as the CPU and the GPU in one program. It accomplishes this by providing an API that abstracts away the particulars of the hardware, such as the number of cores.

At the heart of an OpenCL program is its *kernel*. The kernel is a set of instructions that the developer wishes to execute in parallel over the heterogenous hardware available in the system. This may sound fairly simple; however, developing an OpenCL application is far more nuanced than simply inserting all of your code in a kernel. For instance, there is a complex and hierarchal memory architecture between the program and the computing resources of the GPU, and there are constraints about how processes can intercommunicate. Understanding and leveraging this memory architecture is very important for achieving good performance.

Kernels are executed on *workitems* that are grouped into of *workgroups*. Workgroups have

access to fast, shared local memory and all the workitems within it are allowed to synchronize. It is important to note that this synchronization does not occur across workgroups. Further, all of the workitems (each running a kernel) in a workgroup are executed in simultaneous blocks of threads called warps. If threads in a warp all read or write to sequential locations in a contiguous block of memory at the same time, these memory accesses are "coalesced" and executed in parallel for dramatic speed increases. To make GPU-based code run as fast as possible, one must ensure that as many memory accesses are coalesced as possible and one should minimize costly host to device memory transfers. For further examples of optimization strategies and best practices for developing GPU applications, see [57].

5.4 GPU decomposition of Renyi's Entropy based ICA

The structure of the Renyi's entropy ICA algorithm mentioned in Section 5.1 lends itself especially well to development in OpenCL. As mentioned in the previous section, read/write coalescing is a very important optimization strategy for developing on the GPU. Thus, our GPU task parallelization should make every attempt to organize the memory it manipulates in way that will be easily coalesced by the thread warps. Our main code does this prior to the host to device memory transfer by storing our data in column major format. It also stores this memory aligned to 16 word boundaries to align properly with the memory reads from a warp.

There are two main kernels that we use in the code. The first kernel, the main body of the Renyi entropy ICA calculation, is task parallelized as working on each element of the input matrix, an illustration of this is in Figure 5.1. Thus, our workitems are individual locations in the signal matrix, and our workgroups are the columns. This design was chosen due to the need for coordination of data across columns (e.g. during matrix-vector multiplies), as we compute the vectors $a(p)$, $b(p)$, and $k(p)$ for the p^{th} column according to the six-step algorithm from section 5.1. The second kernel then aggregates the results of the first in a logarithmic fashion illustrated by Figure 5.2.

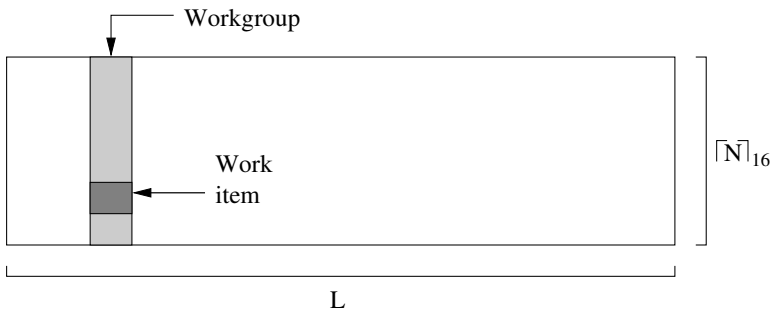


Figure 5.1: An illustration of the 2D memory layout over which the main kernel function is applied. $\lceil N \rceil_{16}$ denotes N rounded up to the next highest multiple of 16.

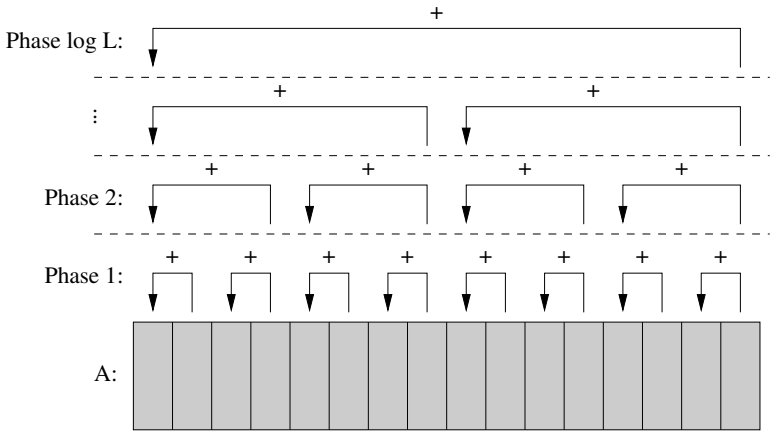


Figure 5.2: An illustration of the function of the aggregate kernel for summing the L columns of the A matrix.

5.5 Results

Our methods for empirical performance testing were as follows: We tested 3 ICA programs. These included our C implementation of Renyi's entropy ICA (non-GPU), our OpenCL implementation of Renyi's entropy ICA, and the Infomax ICA implementation included in EEGLab [1, 3]. We tested on four different machines, the relevant properties of which are listed in Table 5.1. The GPU code was only executed on the GPU capable machines, A and D. Each test consisted of 10 averaged iterations using a specific value of L. The values of L that we used in each of the tests were [100,200,500,1000,2000,5000,10000]. For the results we present here, we picked the fastest result set for each ICA program. Listed here: Infomax ran best on Machine B because MATLAB is multi-threading capable, Renyi non-GPU ran best on Machine C because it had the fastest clock speed,

and Renyi GPU ran best on Machine A because it had the best GPU.

Machine	Processor	Video Card	OS
A	Core i7 980 (6 Core) @ 3.3GHz	GeForce GTX 480	Fedora 13
B	Dual Quad Xeon @ 2.33GHz	Not Used	CentOS 5.3
C	Intel Core2 Duo @ 2.6GHz	Not Used	Mac OS X 10.6.4
D	AMD Opteron @ 1GHz	GeForce 8800 GTX	CentOS 5.5

Table 5.1: The machines we used to run the tests.

The first result that we will cover is the speedup between our Renyi's entropy ICA implementation on the CPU and our Renyi's entropy ICA implementation on the GPU. This result is summarized in Table 5.2. The GPU version of the code displays a 22x speedup in the largest value of L tested, $L = 10000$. The timing of the two versions appears to grow linearly in L as shown in Figure 5.3. This result follows from the fact that the GPU is using 480 cores to perform its calculations, where the non-GPU version is only using 1.

L	GPU Version	non-GPU Version	Speedup Factor
100	750 ms	881 ms	1.17x
200	793 ms	1737 ms	2.19x
500	910 ms	4379 ms	4.81x
1000	1063 ms	8707 ms	8.19x
2000	1431 ms	17552 ms	12.27x
5000	2388 ms	43094 ms	18.05x
10000	3843 ms	85800 ms	22.33x

Table 5.2: Timing Results between non-GPU Renyi and GPU Renyi code.

The next result is from comparing Infomax [3, 1] to our Renyi entropy GPU code. This result, shown in Table 5.3, our GPU of Renyi's Entropy ICA version beats Infomax for tested values of $L > 1000$. This result could be further improved by changing the number of iterations that our algorithm runs. There were many times during our testing of the Renyi GPU code, that we noted that the quality of the result was very high in half to a third of the total iterations. Reducing the number of iterations could easily reduce our timing results or expand them, if more iterations are deemed necessary.

For completeness, Table 5.4 shows the head-to-head performance differences of the two graphics cards we used. The Fermi card (an GeForce GTX 480) is the latest hardware version of NVIDIA's architecture and boasts 480 GPU cores. The GeForce 8800 GTX card is a much older hardware version and only has 128 cores. We observed a speedup of close to 2 times from using the

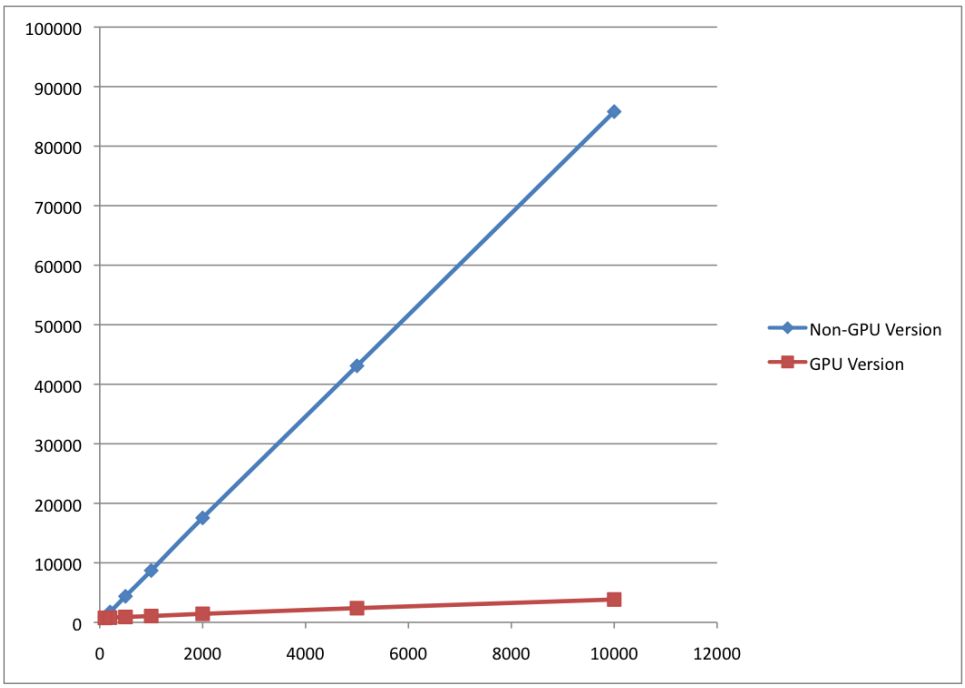


Figure 5.3: The performance curves of the non-GPU and GPU versions of the Renyi's entropy ICA code. The y-axis is in milliseconds, while the x-axis is sizes of L.

newer hardware architecture.

L	Renyi's Entropy ICA GPU	Infomax	Speedup Factor
100	750 ms	255 ms	0.34x
200	793 ms	67 ms	0.08x
500	910 ms	433 ms	0.48x
1000	1063 ms	838 ms	0.79x
2000	1431 ms	2881 ms	2.01x
5000	2388 ms	8034 ms	3.36x
10000	3843 ms	7280 ms	1.89x

Table 5.3: Timing results between regular Infomax ICA and GPU Renyi code.

L	GeForce GTX 480	GEForce 8800GTX	Speedup Factor
100	750 ms	1066 ms	1.42x
200	793 ms	1149 ms	1.45x
500	910 ms	1325 ms	1.46x
1000	1063 ms	1612 ms	1.52x
2000	1431 ms	2174 ms	1.52x
5000	2388 ms	3848 ms	1.61x
10000	3843 ms	6533 ms	1.70x

Table 5.4: Timing results of the Renyi GPU code on the two GPUs we tested on.

Chapter 6

Conclusion

6.1 Future Directions

The work described in this dissertation constitutes the first steps in a large-scale collaborative research effort envisioned by Dr. Halford and Dr. Dean. Some of the more long term objectives that are not included in the scope of this dissertation are listed in this final chapter. We also briefly review how each of these objectives can be beneficial to the continued development of EEG research.

6.1.1 Spike Detection/Classification

Having a computer automatically classify any PEEs present in an EEG dataset is one of the main goals for computational EEG research. As stated in Chapter 1 and Section 2.2, this is a very difficult problem and the purpose of the system developed here is to build a computational framework that supports this ultimate goal. To this end, we believe that the future of the project will include a computational research portal (described in the next section) where users will be able to upload a specification of their spike detection algorithm and run that against the gold-standard datasets. Such a specification might be similar to a script for Condor, a cycle-scavenging cluster system [78], where the user would specify a binary to execute, the input data for the problem, and architectural details of both the binary and the cluster resources needed. This last point is particularly difficult and intriguing considering the vast differences in algorithmic approaches taken (see Section 2.2) and the architectures that they may be conceivably need to exploit (e.g. grids,

MPI-based systems, GPUs).

6.1.2 mBSSS

Due to the embarrassingly parallel nature of mBSSS, grid architectures seem well suited for it. We have already implemented a baseline solution for the mBSSS algorithm leveraging the OSG (Open Science Grid) interface here at Clemson University. The system, as implemented, created the datasets as needed and transferred them to the client machines with a windows binary for execution of the ICA algorithm. The clients then returned the results to the dispatching machine. This approach was not very sophisticated in its management of resources and the performance increase from this approach was minimal (28% faster). This drives home the point that the success of a coarse-grained parallelization will depend heavily on the transfer time of the binaries and data to the worker machines as well as the use of a smart, high-throughput dispatch mechanism for this data (e.g. a hierarchal approach). Also, optimization of selection of data (such that each worker can calculate several adjacent windows and return them) sent to the workers will help this system out greatly. mBSSS can also be implemented using our Renyi's entropy GPU system, described in Chapter 5, working over clusters of GPUs. When mBSSS can be performed efficiently, we can then assess its usefulness in helping alleviate the problems inherent in standard sliding window ICA approaches to EEG analysis as mentioned in Section 2.2.2.

6.1.3 EEG Computing Portal and Pedagogy

Ultimately, we would like to create a interactive public research portal where EEG researchers can collaborate. There are many Computational Neuroscience centers, such as the Swartz Center for Computational Neuroscience at University of California, San Diego (one of their chief contributions to the field is the excellent MATLAB program EEGLab [1]). However, these centers do not host collaborative portals for EEG research – currently, spike detection research is carried out in independent labs using different datasets with very little direct comparison to other researchers [47].

We would like this portal to accomplish such tasks as: uploading datasets, annotating new training sets, downloading existing training sets, and most ambitiously, running arbitrary spike detection algorithms on the datasets. There is precedent for this kind of system/portal in other

domains such as Machine Learning. For example, Percy Liang, describes the MLComp.org system he has developed as follows: "The goal of mlcomp is two-fold: (1) to objectively evaluate and compare different machine learning methods; and (2) help practitioners quickly try out many different machine learning methods. Here's how it works: (1) you upload programs and/or datasets; (2) programs are run on datasets; (3) various performance metrics are reported on these runs. An important aspect of mlcomp is that it is collaborative: each person can upload just one program or dataset, but the collective contribution yields a vast infrastructure that everyone can benefit from." [63]

Among the benefits of the system that we are describing, one of the more obvious ones is the application in the pedagogy of EEG analysis. A portion of the portal we intend to develop might be dedicated to interactive training for the purposes of educating students of EEG and remediating the faults of existing practitioners. In the future, a certification system could arise around such a system and prove to be very valuable to the field.

6.2 Concluding Thoughts

The automatic interpretation of EEG is a very difficult problem that has eluded a well-accepted solution so far. It is the purpose of this dissertation proposal to create a computational framework that helps work toward that solution. By creating software for collaborating on building new datasets against which researchers can measure correctness of their spike detection algorithms (and therefore have a basis of comparison with others) and providing a new parallel, high performance implementation of ICA (a computationally intensive preprocessing step) on a GPGPU, this project will add great value to the study of EEG. As the area of intersection between neuroscience and computer science grows, this framework may be employed to build the future algorithms that can be of great assistance in helping decrease the 20%-30% misdiagnosis rate of epilepsy [18, 73] in rEEG and help bring a better quality of life to sufferers of epilepsy.

Bibliography

[1] S Makeig A Delorme. EEGlab: an open source toolbox for analysis of single-trial EEG dynamics. *Journal of Neuroscience Methods*, 134:9–21, 2004.

[2] W. Addison and S. Roberts. Blind source separation with non-stationary mixing using wavelets. In *6th International Conference on Independent Component Analysis and Blind Source Separation.*, 2006.

[3] A.J. Bell and T.J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computing*, 7:1129–1159, 1995.

[4] S.R. Benbadis and W.O. Tatum. Overinterpretation of EEGs and misdiagnosis of epilepsy. *Journal of Clinical Neurophysiology*, 20:42–44, 2003.

[5] R. Benlamri, M. Batouche, S. Rami, and C. Bouanaka. An automated system for analysis and interpretation of epileptiform activity in the EEG. *Comput Biol Med*, 27:129–139, 1997.

[6] M.A. Black, R.D Jones, G.J Carroll, A.A. Dingle, I.M. Donaldson, and P.J. Parkin. Real-time detection of epileptiform activity in the EEG: a blinded clinical trial. *Clin Electroencephalogr*, 31:122–130, 2000.

[7] G.D. Brown, S. Yamada, and T.J. Sejnowski. Independent component analysis at the neural cocktail party. *Trends in Neurosciences*, 24:54–63, 2001.

[8] I. Buck, T. Foley, D. Horn, J. Sugerman, K. Fatahalian, M. Houston, and P. Hanrahan. Brook for gpus: Stream computing on graphics hardware. *ACM Trans. on Graphics*, 23(3):777–786, August 2004.

[9] J.F. Cardoso. Blind signal separation: statistical principles. *Proc. IEEE*, 86(1):2009–2025, 1998.

[10] S.J. Choi, A. Cichocki, and A. Beloucharni. Second order nonstationary source separation. *Journal of VLSI Signal Processing Systems for Signal Image and Video Technology*, 32:93–104, 2002.

[11] Robert Cochran and Jay Steele. Second-order illumination in real-time (student paper). In *ACM-SE 45: Proceedings of the 45th annual southeast regional conference*, pages 13–18, New York, NY, USA, 2007. ACM.

[12] J. Cohen. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37, 1960.

[13] M.H. Cohen and A.G. Andreou. Analog CMOS integration and experimentation with an autoadaptive independent component analyzer. *Ieee Transactions on Circuits and Systems Ii-Analog and Digital Signal Processing*, 42:65–77, 1995.

[14] Lopes da Silva. *Computer-Assisted EEG Diagnosis: Pattern Recognition and Brain Mapping, in Electroencephalography: Basic Principles, Clinical Applications, and Related Fields, E. Niedermeyer and F. Lopes da Silva, Editors.* Lippincott Williams and Wilkins, Philadelphia, 2005.

[15] P. Guedes de Oliveira, C. Queiroz, and F. Lopes da Silva. Spike detection based on a pattern recognition approach using a microcomputer. *Electroencephalography and Clinical Neurophysiology*, 56:97–103, 1983.

[16] C.A. Van Donselaar, R. Schimsheimer, A.T. Geerts, and A.C. Declerck. Value of the electroencephalogram in adult patients with untreated idiopathic first seizure. *Arch Neurol*, 49:231–237, 1992.

[17] H. Du and H. Qi. An FPGA implementation of parallel ICA for dimensionality reduction in hyperspectral images. In *IEEE International Geoscience and Remote Sensing Symposium (IGARSS).*, 2004.

[18] A. ZaidiA et al. Misdiagnosis of epilepsy: Many seizure-like attacks have a cardiovascular cause. *Journal of the American College of Cardiology*, 36:2000, 2000.

[19] B. L. Davey et al. Expert system approach to detection of epileptiform activity in the EEG. *Medical and Biological Engineering and Computing*, 27:365–370, 1989.

[20] C.C.C. Pang et al. A comparison of algorithms for detection of spikes in the electroencephalogram. *IEEE Transactions on Biomedical Engineering*, 50:521–525, 2003.

[21] C.P. Unsworth et al. Redundancy of independent component analysis in four common types of childhood epileptic seizure. *Journal of Clinical Neurophysiology*, 23:245–253, 2006.

[22] D. Erdogmus et al. On-line entropy manipulation: stochastic information gradient. *IEEE Signal Processing Letters*, 10(8):242–245, 2003.

[23] D. Flanagan et al. Improvement in the performance of automated spike detection using dipole source features for artifact rejection. *Clinical Neurophysiology*, 114:38–49, 2003.

[24] D.B. Keith et al. Parallel ICA methods for EEG neuroimaging. In *IEEE Parallel and Distributed Processing Symposium (IPDPS)*, 2006.

[25] H.S. Park et al. Detection of epileptiform activities in the EEG using neural network and expert system. *Medinfo*, 9 pt 2:1255–1259, 1998.

[26] J.D. Owens et al. A survey of general-purpose computation on graphics hardware. *Computer Graphics Forum*, 26:80–113, 2007.

[27] L. Senhadji et al. Wavelet analysis of EEG for three-dimensional mapping of epileptic events. *Annals of Biomedical Engineering*, 23:543–552, 1995.

[28] M. Ajouadi et al. Detection of interictal spikes and artifactual data through orthogonal transformations. *Journal of Clinical Neurophysiology*, 22:53–64, 2005.

[29] M. Feucht et al. Simultaneous spike detection and topographic classification in pediatric surface EEGs. *Neuroreport*, 8:2193–2197, 1997.

[30] N. Acir et al. Automatic detection of epileptiform events in EEG by a three-stage procedure based on artificial neural networks. *IEEE Transactions on Biomedical Engineering*, 52:30–40, 2005.

[31] S.B. Wilson et al. Spike detection II: automatic, perception-based detection and clustering. *Clinical Neurophysiology*, 110:404–411, 1999.

[32] T. Sugi et al. Adaptive EEG spike detection: determination of threshold values based on conditional probability. *Frontiers of Medicine, Biology, and Engineering*, 11:261–277, 2002.

[33] W.R. Webber et al. Practical detection of epileptiform discharges (EDs) in the EEG using an artificial neural network: a comparison of raw and parameterized EEG data. *Electroencephalography and Clinical Neurophysiology*, 91:194–204, 1994.

[34] R. Everson and S. Roberts. Blind source separation for non-stationary mixing. *Journal of VLSI Signal Processing Systems for Signal Image and Video Technology*, 26:15–23, 2000.

[35] J. Fayn, P. Rubel, and P.W. Macfarlane. Can the lessons learned from the assessment of automated electrocardiogram analysis in the common standards for quantitative electrocardiography study benefit measurement of delayed contrast-enhanced magnetic resonance images? *J Electrocardiol*, 40:246–250, 2007.

[36] J.L. Fleiss et al. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382, 1971.

[37] A.J. Gabor and M. Seyal. Automated interictal EEG spike detection using artificial neural networks. *Electroencephalography and Clinical Neurophysiology*, 83:271–280, 1992.

[38] H. Goelz, R.D. Jones, and P.J. Bones. Wavelet analysis of transient biomedical signals and its application to detection of epileptiform activity in the EEG. *Clinical Electroencephalography*, 31:181–191, 2000.

[39] S. Goodacre, A. Webster, and F. Morris. Do computer generated ECG reports improve interpretation by accident and emergency senior house officers? *Postgraduate Medical Journal*, 77:455–457, 2001.

[40] J. Gotman and P. Gloor. Automatic recognition and quantification of interictal epileptic activity in the human scalp EEG. *Electroencephalography and Clinical Neurophysiology*, 41:513–529, 1976.

[41] J. Gotman, J.R. Ives, and P. Gloor. Automatic recognition of inter-ictal epileptic activity in prolonged EEG recordings. *Electroencephalography and Clinical Neurophysiology*, 46:510–520, 1979.

[42] J. Gotman and L.Y. Wang. State-dependent spike detection: concepts and preliminary results. *Electroencephalography and Clinical Neurophysiology*, 79:11–19, 1991.

[43] N. Govindaraju, J. Gray, R. Kuman, and D. Manocha. Gputerasort: High performance graphics co-processor sorting for large database management. In *Proc. ACM SIGMOD*, pages 325–336, June 2006.

[44] Larry Greischar. Brain imaging workshop. http://psyphz.psych.wisc.edu/∼greischar/BIW12-11-02/EEGintro.htm, 2002.

[45] I. Guler and E.D. Ubeyli. Adaptive neuro-fuzzy inference system for classification of EEG signals using wavelet coefficients. *Journal of Neuroscience Methods*, 148:113–121, 2005.

[46] J.J. Halford. Detection of paroxysmal EEG discharges using multi-taper blind signal source separation. *Lecture Notes in Computer Science*, 4666:601–608, 2007.

[47] J.J. Halford. Computerized epileptiform transient detection in the scalp electroencephalogram: Obstacles to progress and the example of computerized ECG interpretation. *Clinical Neurophysiology*, 120:1909–1915, 2009.

[48] G. Hellmann. Multifold features determine linear equation for automatic spike detection applying neural non interictal ECoG. *Clinical Neurophysiology*, 110:887–894, 1999.

[49] W.E. Hostetler, H.J.Doller, and R.W. Homan. Assessment of a computer program to detect epileptiform spikes. *Electroencephalogr Clin Neurophysiol*, 83:1–11, 1992.

[50] A. Hyvarinen, J. Karhunen, and E. Oja. *Independent Component Analysis.* Wiley, New York, revised second edition, 2001.

[51] A. Hyvarinen and E. Oja. A fast fixed-point algorithm for independent component analysis. *Neural Computation*, 9:1483–1492, 1997.

[52] Kenneth E. Hild II, Deniz Erdogmus, and Jose C. Principe. An analysis of entropy estimators for blind source separation. *Signal Processing*, 86(1):182 – 194, 2006.

[53] C.J. James and C.W. Hesse. Independent component analysis for biomedical signals. *Physiological Measurement*, 26:15–39, 2005.

[54] H.H. Jasper. Appendix to report to committee on clinical examination in EEG: the ten-twenty electrode system of the international federation. *Electroencephalography Clinical Neurophysiology*, 10:371–375, 1958.

[55] E. Jovanov, D. Starcevic, V. Radivojevic, A. Samardzic, and V. Simeunovic. Perceptualization of biomedical data. an experimental environment for visualization and sonification of brain electrical activity. *Engineering in Medicine and Biology Magazine, IEEE*, 18(1):50–55, Jan.-Feb. 1999.

[56] William B. S. Pressly Jr. Tspad: A Tablet-PC Based application for annotation and collaboration on time series data. In *Proc. of the* 46^{th} *Annual ACM SE Conf.*, pages 166–171, Auburn, Alabama, March 2008.

[57] Khronos Group. Opencl introduction and overview. http://www.khronos.org/opencl, June 2010.

[58] C. Kurth, F. Gilliam, and B.J. Steinhoff. EEG spike detection with a kohonen feature map. *Annals of Biomedical Engineering*, 28:1362–1369, 2000.

[59] J. R. Landis and G. G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174, March 1977.

[60] M. Latka and Z. Was. Wavelet analysis of epileptic spikes. *Physical Review*, 67:1–4, 2003.

[61] P. LeVan, E. Urrestarazu, and J. Gotman. A system for automatic artifact removal in ictal scalp eeg based on independent component analysis and bayesian classification. *Clinical Neurophysiology*, 117:912–927, 2006.

[62] P.S. Lewis and J.C. Mosher. Genetic algorithms for neuromagnetic source reconstruction. In *Acoustics, Speech, and Signal Processing, 1994. ICASSP-94.,*, volume v, pages V/293–V/296 vol.5, Apr 1994.

[63] Percy Liang. Mlcomp. http://mlcomp.org/, 2009.

[64] J. Makhoul, F. Kubala, R. Schwartz, and R. Weischedel. Performance measures for information extraction. In *Proceedings of the DARPA Broadcast News Workshop*, 1999.

[65] NVIDIA Corp. Nvidia CUDA programming guide, version 2.1. http://www.nvidia.com/object/cuda_get.html, December 2008.

[66] O. Ozdamar and T. Kalayci. Detection of spikes with artificial neural networks using raw EEG. *Computers and Biomedical Research*, 31:122–142, 1998.

[67] W.D. Penny, R.M. Everson, and S.J. Roberts. Hidden markov independent component analysis. In M. Girolami, editor, *Advances in independent component analysis*, page 279. Springer: London, New York, 2000.

[68] D.T. Pham and J.F. Cardoso. Blind separation of instantaneous mixtures of nonstationary sources. *IEEE Transactions on Signal Processing*, 49:1837–1848, 2001.

[69] J.W. Phillips, R.M. Leahy, J.C. Mosher, and B. Timsari. Imaging neural activity using MEG and EEG. *Engineering in Medicine and Biology Magazine, IEEE*, 16(3):34–42, May/Jun 1997.

[70] A.B. Samardzic, E. Jovanov, and D.B. Starcevic. 3d visualization of brain electrical activity. In *Engineering in Medicine and Biology Society, 1996. Bridging Disciplines for Biomedicine. Proceedings of the 18th Annual International Conference of the IEEE*, volume 5, pages 2273–2274 vol.5, Oct-3 Nov 1996.

[71] Aleksandar B. Samardzic, Emil S. Jovanov, and Dusan B. Starcevic. Real-time visualization of brain electrical activity. *Real-Time Imaging*, 6(1):69 – 76, 2000.

[72] R. Sankar and J. Natour. Automatic computer analysis of transients in EEG. *Computers in Biology and Medicine*, 22:407–422, 1992.

[73] B. Scheepers, P. Clough, and C. Pickles. The misdiagnosis of epilepsy: findings of a population study. *Seizure*, 7:403–406, 1998.

[74] A. Snellings, D.J. Anderson, and J.W. Aldridge. Improved signal and reduced noise in neural recordings from close-spaced electrode arrays using independent component analysis as a preprocessor. *Journal of Neuroscience Methods*, 150:254–264, 2006.

[75] A. Subasi and E. Ercelebi. Classification of EEG signals using neural network and logistic regression. *Computer Methods and Programs in Biomedicine*, 78:87–99, 2005.

[76] Michael ten Caat, Natasha M. Maurits, and Jos B.T.M. Roerdink. Design and evaluation of tiled parallel coordinate visualization of multichannel EEG data. *IEEE Transactions on Visualization and Computer Graphics*, 13(1):70–79, 2007.

[77] Michael ten Caat, Natasha M. Maurits, and Jos. B.T.M. Roerdink. Data-driven visualization and group analysis of multichannel EEG coherence with functional units. *IEEE Transactions on Visualization and Computer Graphics*, 14(4):756–771, 2008.

[78] Douglas Thain, Todd Tannenbaum, and Miron Livny. Distributed computing in practice: the condor experience. *Concurrency - Practice and Experience*, 17(2-4):323–356, 2005.

[79] W.R.S. Webber, B. Litt, R.P. Lesser, R.S. Fisher, and Bankman. I. automatic EEG spike detection: what should the computer imitate? *Electroencephalography and Clinical Neurophysiology*, 87:364–373, 1993.

[80] J.L. Willems, C. Abreau-Lima, P. Arnaud, J.H. Van Bemmel, C. Brohet, and R. Degani et al. The diagnostic performance of computer programs for the interpretation of electrocardiograms. *New England Journal of Medicine*, 325:1767–1773, 1991.

[81] J.L. Willems, P. Arnaud, J.H. Van Bemmel, P.J. Bourdillon, C. Brohet, and S. Dalla Volta et al. Assessment of the performance of electrocardiographic computer programs with the use of a reference data base. *Circulation*, 71:523–524, 1985.

[82] J.L. Willems, P. Arnaud, J.H. Van Bemmel, R. Degani P.W. Macfarlane, and C. Zywietz. Common standards for quantitative electrocardiography: goals and main results. *Methods Inf Med*, 29:263–271, 1990.

[83] S.B. Wilson and R. Emerson. Spike detection: a review and comparison of algorithms. *Clinical Neurophysiology*, 113:1873–1881, 2002.

[84] S.B. Wilson, R.N. Harner, F.H. Duffy, B.R. Tharp, M.R. Nuwer, and M.R. Sperling. Spike detection. I. correlation and reliability of human experts. *Electroencephalography and Clinical Neurophysiology*, 98:186–198, 1996.

[85] CSE Workgroup. Common standards for quantitative electrocardiography: the CSE pilot study. In *Medical informatics Europe 81: third congress of the European federation of medical informatics*, pages 319–326, 1981.

[86] CSE Workgroup. CSE data. http://CSE.insa-lyon.fr, 1981.

[87] Dongxin Xu, Jose C. Principe, John Fisher, III, John Fisher Iii, and Hsiao-Chun Wu. A novel measure for independent component analysis (ica). pages 1161–1164.

[88] H. Zhang, M.O. Balaban, and J.C. Principe. Improving pattern recognition of electronic nose data with time-delay neural networks. *Sensors and Actuators B: Chemical*, 96:385–389, 2003.