University of Memphis
University of Memphis Digital Commons

Electronic Theses and Dissertations

1-1-2018

AUTOMATED ARTIFACT REMOVAL AND DETECTION OF MILD COGNITIVE IMPAIRMENT FROM SINGLE CHANNEL ELECTROENCEPHALOGRAPHY SIGNALS FOR REAL-TIME IMPLEMENTATIONS ON WEARABLES

Saleha Khatun

Follow this and additional works at: https://digitalcommons.memphis.edu/etd

Recommended Citation
Khatun, Saleha, "AUTOMATED ARTIFACT REMOVAL AND DETECTION OF MILD COGNITIVE IMPAIRMENT FROM SINGLE CHANNEL ELECTROENCEPHALOGRAPHY SIGNALS FOR REAL-TIME IMPLEMENTATIONS ON WEARABLES" (2018). *Electronic Theses and Dissertations*. 2918.
https://digitalcommons.memphis.edu/etd/2918

This Dissertation is brought to you for free and open access by University of Memphis Digital Commons. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of University of Memphis Digital Commons. For more information, please contact khggerty@memphis.edu.

AUTOMATED ARTIFACT REMOVAL AND DETECTION OF MILD COGNITIVE IMPAIRMENT FROM SINGLE CHANNEL ELECTROENCEPHALOGRAPHY SIGNALS FOR REAL-TIME IMPLEMENTATIONS ON WEARABLES

By

Saleha Khatun

A Dissertation

Submitted in Partial Fulfillment of the

Requirement for the Degree of

Doctor of Philosophy

Major: Electrical and Computer Engineering

The University of Memphis

December 2018

DEDICATION

This doctoral thesis is dedicated to my husband (Nurmohammed Patwary), to my Mom (Sayama Khatun), and to my Dad (Abdul Kuddus). The continuous support and inspiration from my husband have made my journey easier. My mom's love for education and my dad's teaching for higher aspiration have shaped my life.

ACKNOWLEDGEMENT

I would like to thank my advisor, Dr. Bashir I. Morshed, for his constant support, enthusiasm, and precious mentoring. He always gave me guided freedom to explore and always taught me to work systematically and thoroughly. I am extremely grateful to get the opportunity to work under his supervision. I also like to thank Dr. Gavin M. Bidelman for his valuable support, and feedbacks which helped me in different stages of the research. I want to extend thanks to Dr. Eddie Jacobs, Dr. Aaron Robinson, and Dr. Amy Curry for being my committee members.

I would also like to thank my former and current lab-mates at the Embedded Systems Advanced Research and Prototyping (ESARP) Lab for being good colleagues during my research: Dr. Ruhi Mahajan, Dr. Ankita Mohapatra, Charvi A Majmudar, Babak Noorozi, Md. Sabbir Zaman, Md. Juber Rahman, Tasnuba Siddiqui, Sharmin Afroz, Dr. Mohammad AbuSaude, and Sergi Consul-Pacareu.

I want to acknowledge the FedEx Institute of Technology, Herff College of Engineering, Institute of Intelligent Systems Student Organization (IISSO), and Student Graduate Association (SGA) for providing various funding during my study. I want to remember the love and good wishes from my family (my siblings: Md. Shahabuddin, A B M Jamil, Shamsun Nahar) and friends (Amena Majeed, Mehnaz Rahman, and others).

ABSTRACT

Khatun, Saleha. Ph.D. The University of Memphis. November 2018. Automated Artifact Removal and Detection of Mild Cognitive Impairment from Single Channel Electroencephalography Signals for Real-Time Implementations on Wearables. Major Professor: Dr. Bashir I. Morshed.

Electroencephalogram (EEG) is a technique for recording asynchronous activation of neuronal firing inside the brain with non-invasive scalp electrodes. EEG signal is well studied to evaluate the cognitive state, detect brain diseases such as epilepsy, dementia, coma, autism spectral disorder (ASD), etc. In this dissertation, the EEG signal is studied for the early detection of the Mild Cognitive Impairment (MCI). MCI is the preliminary stage of Dementia that may ultimately lead to Alzheimer's disease (AD) in the elderly people. Our goal is to develop a minimalistic MCI detection system that could be integrated to the wearable sensors. This contribution has three major aspects: 1) cleaning the EEG signal, 2) detecting MCI, and 3) predicting the severity of the MCI using the data obtained from a single-channel EEG electrode.

Artifacts such as eye blink activities can corrupt the EEG signals. We investigate unsupervised and effective removal of ocular artifact (OA) from single-channel streaming raw EEG data. Wavelet transform (WT) decomposition technique was systematically evaluated for effectiveness of OA removal for a single-channel EEG system. Discrete Wavelet Transform (DWT) and Stationary Wavelet Transform (SWT), is studied with four WT basis functions: haar, coif3, sym3, and bior4.4. The performance of the artifact removal algorithm was evaluated by the correlation coefficients (CC), mutual information (MI), signal to artifact ratio (SAR), normalized mean square error (NMSE), and time-frequency analysis. It is demonstrated that WT can be an effective tool for unsupervised OA removal from single channel EEG data for real-time applications.

For the MCI detection from the clean EEG data, we collected the scalp EEG data, while the subjects were stimulated with five auditory speech signals. We extracted 590 features from the Event-Related Potential (ERP) of the collected EEG signals, which included time and spectral domain characteristics of the response. The top 25 features, ranked by the random forest method, were used for classification models to identify subjects with MCI. Robustness of our model was tested using leave-one-out cross-validation while training the classifiers. Best results (leave-one-out cross-validation accuracy 87.9%, sensitivity 84.8%, specificity 95%, and F score 85%) were obtained using support vector machine (SVM) method with Radial Basis Kernel (RBF) (sigma = 10, cost = 102). Similar performances were also observed with logistic regression (LR), further validating the results. Our results suggest that single-channel EEG could provide a robust biomarker for early detection of MCI.

We also developed a single channel Electro-encephalography (EEG) based MCI severity monitoring algorithm by generating the Montreal Cognitive Assessment (MoCA) scores from the features extracted from EEG. We performed multi-trial and single-trail analysis for the algorithm development of the MCI severity monitoring. We studied Multivariate Regression (MR), Ensemble Regression (ER), Support Vector Regression (SVR), and Ridge Regression (RR) for multi-trial and deep neural regression for the single-trial analysis. In the case of multi-trial, the best result was obtained from the ER. In our single-trial analysis, we constructed the time-frequency image from each trial and feed it to the convolutional deep neural network (CNN). Performance of the regression models was evaluated by the RMSE and the residual analysis. We obtained the best accuracy with the deep neural regression method.

PREFACE

This dissertation is composed in 3 (three) journal format. I am the first author of these three journals. Chapter 2 is based on the published paper, *S. Khatun, R. Mahajan, and B. I. Morshed, "Comparative Study of Wavelet Based Unsupervised Ocular Artifact Removal Techniques for Single Channel EEG data," IEEE Journal of Translational Engineering in Health and Medicine (JTEHM), vol. 4, pp. 1-8, 2016.*

Chapter 3 is based on the submitted paper, *S. Khatun, B. I. Morshed and G. M. Bidelman, "A Single- Channel EEG- Based Approach to Detect Mild Cognitive Impairment via Speech-Evoked Brain Responses," IEEE Transactions on Neural Systems and Rehabilitation Engineering, (submitted minor review response), Oct. 2018.*

Chapter 4 is based on the ready to submit manuscript, *S. Khatun, B. I. Morshed and G. M. Bidelman, "Regression Based Approaches to Monitor the Severity of Mild Cognitive Impairment from Single-Channel EEG Data," IEEE Transactions on Neural Systems and Rehabilitation Engineering, (submitted),* 2018.

This work is supported by the University of Memphis through the "Herff Engineering Fellowship" awarded to Saleha Khatun.

Table of Contents

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATION

AD: Alzheimer's Disease

ASD: Autism Spectrum Disorder

BADL: Basic Activities of Daily Living

BCI: Brain Computer Interface

CC: Correlation Coefficient

CFV: Candidate Feature Vector

CNN: Convolutional Neural Network

CRQA: Cross Recurrence Quantification Analysis

CSF: Cerebrospinal Fluid

CV: Coefficient of Variation

DTI: Diffusion Tensor Imaging

DWT: Discrete Wavelet Transform

EEG: Electroencephalogram

EI: Expected Improvement

EOG: Electrooculography

ER: Ensemble Regression

ERP: Event Related Potential

ERP: Event Related Potential

FDG-PET: Fluorodeoxyglucose positron emission tomography

GPCOG: General Practitioner Assessment of Cognition

HC: Healthy Control

HPF: High Pass Filter

IADL: Instrumental Activities of Daily Living

ICA: Independent Component Analysis

IQCODE: Informant Questionnaire on Cognitive Decline in the Elderly

LPF: Low Pass Filter

LR: Logistic Regression

MAE: Mean Absolute Error

MCI: Mild Cognitive Impairment

MEG: Magnetoencephalography

MI: Mutual Information

MIS: Memory Impairment Screen

MMN: Mismatch Negativity

MMSE: Mini-Mental State Examination

MoCA: Montreal Cognitive Assessment

MR: Multivariate Regression

MRI: Magnetic Resonance Imaging

NMSE: Normalized Mean Square Error

NN: Neural Network

OA: Ocular Artifact

PCA: Principal Component Analysis

PD: Parkinson's Disease

PET: Positron Emission Tomography

QQplot: Quantile-Quantile Plot

RBF: Radial Basis Function

RF: Random Forest

RMSE: Root Mean Square Error

RQA: Recurrence Quantification Analysis

RR: Ridge Regression

Rs-fMRI: resting-state functional magnetic resonance imaging

SAR: Signal to Artifact Ratio

SCC: Smart and Connected Community

SLUMS: St. Louis University Mental Status Exam

sMRI: structural magnetic resonance imaging

SSE: Sum Square Error

ST: Statistical Threshold

SVM: Support Vector Machine

SVR: Support Vector Ratio

SVR: Support Vector Regression

SWT: Stationary Wavelet Transform

TFA: Time Frequency Analysis

UT: Universal Threshold

WT: Wavelet Transform

1. INTRODUCTION

1.1 Problem Statement

Memory impairment due to aging is natural. However, if the level of impairment goes beyond the expected level, that condition is stated as mild cognitive impairment (MCI) [1]. MCI is a transforming state in cognitive function between changes deriving from natural/normal aging and dementia [2][3]. Alzheimer's disease (AD) is the most prevalent dementia [4][5][6][7] for elderly people in many countries, and it is accompanied by progressively worsening memory, reasoning, and other aspects of cognition [7][8]. MCI can also be related with Parkinson's disease (PD) [9], cardiovascular disease, type 2 diabetes, obesity, sedentary activity, metabolic syndrome, excess alcohol, and smoking [10]. As MCI is the preliminary stage of cognitive impairment, most often if it is not treated properly, there is a 10-54% and 10-15% chance that MCI may lead to dementia, or Alzheimer's, respectively [8]. The cost of providing care for the Alzheimer patients in the US was $200 billion in 2012, and it is estimated to grow to $1.1 trillion per year by 2050 [11]. Therefore, detecting and managing this disease is of great importance for better healthcare as well as for the national financial interest. Another aspect of Alzheimer's disease is that the cause of this neurodegenerative disease is still unclear. So, earlier detection of MCI is critical in order to develop care-plans and prognosis of Alzheimer or dementia. MCI detection is possible through neurological signal monitoring such as EEG.

1.2 Motivation

Wearable sensing of elderly people's health is a growing field with the augmentation of real-time detection and monitoring. The algorithm to detect mild cognitive impairment

should be developed in such a way that it can be implementable in a simplistic single channel EEG system [12].

Neuro-physiological signals often get affected by artifacts which hinder the valuable information processing from them, and thus reduce the accuracy and performance of algorithms by increasing inaccuracies such as false positives. So, it is crucial to have artifact removal algorithm applicable in single channel EEG based mechanism for robust detection of MCI.

MCI can be the biomarker of Dementia/Alzheimer disease. Low cost, fast, and easy solution is achievable by extracting features and processing them to identify biomarkers from neuro-physiological signals such as EEG. MCI detection from EEG is thus crucial to prevent Alzheimer's disease at the onset of it.

Severity of MCI detection would be an important part of remote patient monitoring in outside of hospital settings. Single channel EEG based system can be a portable, cost effective, and fast solution to this problem.

1.3 Goal

Early detection and monitoring the progression of MCI through unsupervised algorithms that uses only single channel EEG data.

1.4 Objectives

1. Algorithm development for unsupervised removal of major artifacts from single channel EEG signals.

2. Algorithm development to detect MCI from single channel EEG data using speech-evoked brain responses.

3. Algorithm development for MCI severity detection from single trials of single-channel EEG data towards early detection and monitoring of MCI.

1.5 Technical Challenges

- EEG signals can be noisy and can easily get contaminated by ocular artifacts, motion artifacts, power line interferences, etc. To automatically remove artifacts from EEG signals is a challenge in a minimalistic system, but is needed for real-time implementation.

- Automatic classification and detection of MCI can be implemented in wearables and can allow users to continually monitor their brain states at the comfort of their home.

- For single trial-based MCI severity detection, there can be trial-trial variabilities, artifacts, etc. which add to difficulties. Features, epoch length, algorithms are crucial to resolve those issues.

1.6 Research Outcomes:

1. Algorithm to remove ocular artifact from a single-channel EEG system*.

2. Algorithm to detect MCI from a single-channel EEG system using speech-evoked brain responses.

3. Algorithm to determine MCI severity ranking from EEG signals to facilitate MCI severity monitoring.

*Made publicly available (http://www.memphis.edu/esarp/repository/index.php)

1.7 Publications and awards:

Journals:

1. S. Khatun, R. Mahajan, and B. I. Morshed, "Comparative Study of Wavelet-Based Unsupervised Ocular Artifact Removal Techniques for Single-Channel EEG Data," IEEE J. Translational Engineering in Health and Medicine (JTEHM), vol. 4, pp. 1-8, 2016.
2. S. Khatun, B. I. Morshed, and G. M. Bidelman, "A Single-Channel EEG-Based Approach to Detect Mild Cognitive Impairment via Speech-Evoked Brain Responses", IEEE Transactions on Neural Systems and Rehabilitation Engineering, (submitted minor review responses), Oct. 2018.
3. S. Khatun, B. I. Morshed, and G. M. Bidelman, "Regression Based Approaches to Monitor the Severity of Mild Cognitive Impairment from Single Channel EEG Data", IEEE Transactions on Neural Systems and Rehabilitation Engineering (draft ready for submission), 2018.

Conferences:

1. S. Khatun, B. I. Morshed, and G. M. Bidelman, "Single Channel EEG Based Score Generation to Monitor the Severity and Progression of Mild Cognitive Impairment", in IEEE Region 4 Electro-Information Technology Conference, Rochester, MI, USA, 2018.
2. S. Khatun, and B. I. Morshed, "Human Activity Recognition with transition Awareness", in IEEE Region 4 Electro-Information Technology Conference, Rochester, MI, USA, 2018.

3. S. Khatun, and B. I. Morshed, “Detection of Myocardial Infarction and Arrhythmia from Single- Lead ECG Data using Bagging Trees Classifier”, in IEEE International Conference on Electro Information Technology (EIT), 2017.
4. S. Khatun, B. I. Morshed, and G. M. Bidelman, “Single channel EEG time-frequency features to detect Mild Cognitive Impairment,” in 2017 IEEE International Symposium on Medical Measurements and Applications, MeMeA 2017 - Proceedings, 2017.
5. S. Khatun, and B. I. Morshed, “An Efficient Classification of Four Cognitive Tasks using Single Channel EEG Data from Neuromonitor device”, accepted in IEEE International Women in Engineering (WIE) Conference on Electrical and Computer Engineering, 2015.
6. S. Khatun, R. Mahajan, and B. I. Morshed, “Comparative Analysis of Wavelet Based Approaches for Reliable Removal of Ocular Artifacts from Single Channel EEG,” in IEEE Region 4 Electro-Information Technology Conference, DeKalb, IL, USA, 2015.
7. R. Mahajan, C. A. Majmudar, S. Khatun, B. I. Morshed and G. M. Bidelman., “NeuroMonitor Ambulatory EEG Device: Comparative Analysis and Its Application for Cognitive Load Assessment”, in IEEE EMBS Healthcare Innovation & Point-of-Care Technologies Conference, Seattle, WA, USA, 2014.

Poster Presentations:

1. S. Khatun, B. I. Morshed, and G. M. Bidelman, “Regression based automated scoring technique of Mild Cognitive Impairment (MCI) Severity Using Single Channel EEG Measures with Auditory Stimulus”, in 2018 IEEE 40th Annual

International Conference of Engineering in Medicine and Biology Society (EMBC), 2018

2. S. Khatun, B. I. Morshed, and G. M. Bidelman, "Automated Scoring of Mild Cognitive Impairment (MCI) Severity Using Single Channel EEG Measures with Auditory Stimulus", in IISSO Student Research Fair, 2018.
3. S. Khatun, B. I. Morshed, and G.M. Bidelman, "Early Detection of Mild Cognitive Impairment using an ERP Based Approach", in IISSO Student Research Fair, 2016.
4. S. Khatun, and B. I. Morshed, "Human Activity Recognition with Transition Awareness", in Department of EECE Poster Competition, 2016.
5. S. Khatun, and B. I. Morshed, "An Efficient Classification of Four Cognitive Tasks using Single Channel EEG Data from Neuromonitor device", in IISSO Student Research Fair, 2015.
6. S. Khatun, Ruhi Mahajan, and B. I. Morshed, "Comparative Analysis of Wavelet Based Approaches for Reliable Removal of Ocular Artifacts from Single Channel EEG", in Department of EECE Poster Competition, 2015.

Awards:

1. "Herff Graduate Fellowship" awarded by the Herff College of Engineering, University of Memphis (Fall 2015 – Fall 2017).
2. Graduate Research Assistantship awarded by the ESARP lab of the University of Memphis (Spring 2014 – Spring 2015).

3. Student Travel Award for the paper, "Single Channel EEG Time-Frequency Features to Detect Mild Cognitive Impairment," at MeMeA 2017 in Rochester, MN, USA from IEEE.
4. Student Travel Award for conference paper in 2017, 2018 from Student graduate association (SGA), university of Memphis, TN 38152.
5. Student Travel Award for conference papers in 2015, 2017, 2018 from Institute of Intelligent Systems Student Organization (IISSO), university of Memphis, TN 38152.
6. Student Travel Award for conference papers in 2017 from Department of electrical and computer engineering (EECE), university of Memphis, TN 38152.

1.8 References:

[1] R. C. Petersen, G. E. Smith, S. C. Waring, R. J. Ivnik, E. G. Tangalos, and E. Kokmen, “Mild cognitive impairment - Clinical characterization and outcome,” *Arch. Neurol.*, vol. 56, no. 3, pp. 303–308, 1999.

[2] A. Tsolaki, V. Kosmidou, I. Y. Kompatsiaris, and C. Papadaniil, “Brain source localization of MMN and P300 ERPs in Mild Cognitive Impairment and Alzheimer ’ s Disease : A High Density EEG approach,” *Neurobiol. Aging*, vol. 55, pp. 190–201, 2017.

[3] H. V. Vinters, “Emerging Concepts in Alzheimer’s Disease,” *Annu. Rev. Pathol. Mech. Dis.*, 2015.

[4] H. Ni, J. Qin, L. Zhou, Z. Zhao, J. Wang, and F. Hou, “Network analysis in detection of early-stage mild cognitive impairment,” *Phys. A Stat. Mech. its Appl.*, vol. 478, pp. 113–119, 2017.

[5] O. Ben Ahmed, J. Benois-Pineau, M. Allard, G. Catheline, and C. Ben Amar, “Recognition of Alzheimer’s disease and Mild Cognitive Impairment with multimodal image-derived biomarkers and Multiple Kernel Learning,” *Neurocomputing*, vol. 220, pp. 98–110, 2017.

[6] R. B. Knowles *et al.*, “Plaque-induced neurite abnormalities: Implications for disruption of neural networks in Alzheimer’s disease,” *Proc. Natl. Acad. Sci.*, 1999.

[7] L. R. Trambaiolli, N. Spolaôr, A. C. Lorena, R. Anghinah, and J. R. Sato, “Feature selection before EEG classification supports the diagnosis of Alzheimer’s disease,” *Clin. Neurophysiol.*, vol. 128, pp. 2058–2067, 2017.

[8] C.-L. Tsai, M.-C. Pai, J. Ukropec, and B. Ukropcová, “The Role of Physical Fitness in the Neurocognitive Performance of Task Switching in Older Persons with Mild Cognitive Impairment,” *J. Alzheimer’s Dis.*, 2016.

[9] Baschi, R., Nicoletti, A., Restivo, V., Recca, D., Zappia, M. and Monastero, R., “Frequency and Correlates of Subjective Memory Complaints in Parkinson’s Disease with and without Mild Cognitive Impairment: Data from the Parkinson’s Disease Cognitive Impairment Study,” *Journal of Alzheimer's Disease, (Preprint)*, pp.1-10, 2018.

[10] Maskill, Linda, "Mild cognitive impairment: A quiet epidemic with occupation at its heart," *British Journal of Occupational Therapy*, pp. 485-486, 2018.

[11] D. De Venuto, V. F. Annese, and G. Mezzina, “Remote Neuro-Cognitive Impairment Sensing Based on P300 Spatio-Temporal Monitoring,” *IEEE Sens. J.*, vol. 16, no. 23, pp. 8348–8356, 2016.

[12] S. K. Islam, A. Fathy, Y. Wang, M. Kuhn, and M. Mahfouz, “Hassle-free vitals,” *IEEE Microwave Magazine*. 2014.

2. COMPARATIVE STUDY OF WAVELET BASED UNSUPERVISED OCULAR ARTIFACT REMOVAL TECHNIQUES FOR SINGLE CHANNEL EEG DATA

Abstract—Electroencephalogram (EEG) is a technique for recording asynchronous activation of neuronal firing inside the brain with non-invasive scalp electrodes. Artifacts such as eye blink activities can corrupt these neuronal signals. While ocular artifact (OA) removal is well investigated for multiple channel EEG systems, in alignment with the recent momentum towards minimalistic EEG systems for use in natural environments, we investigate unsupervised and effective removal of OA from single-channel streaming raw EEG data. In this study, unsupervised wavelet transform (WT) decomposition technique was systematically evaluated for effectiveness of OA removal for a single-channel EEG system. A set of seven raw EEG dataset was analyzed. Two commonly used WT methods, Discrete Wavelet Transform (DWT) and Stationary Wavelet Transform (SWT), were applied. Four WT basis functions, haar, coif3, sym3, and bior4.4, were considered for OA removal with Universal Threshold (UT) and Statistical Threshold (ST). To quantify OA removal efficacy from single channel EEG, five performance metrics were utilized: correlation coefficients (CC), mutual information (MI), signal to artifact ratio (SAR), normalized mean square error (NMSE), and time-frequency analysis. The temporal and spectral analysis shows that the optimal combination could be DWT with ST with coif3 or bior4.4 to remove OA among sixteen combinations. This work demonstrates that WT can be an effective tool for unsupervised OA removal from single channel EEG data for real-time applications.

Index Terms—Artifact Removal, Electroencephalogram (EEG), Ocular Artifact, Wavelet Transform, Single Channel EEG.

2.1 Background

Electroencephalogram (EEG) is the recording of the brain's spontaneous electrical activity captured non-invasively by placing electrodes on the scalp [1]. EEG has been utilized in many medical diagnosis, prognosis and therapies including epilepsy, sleep disorder, coma, encephalopathy and brain deaths [2]. EEG signals are often corrupted by two sources of artifacts: physiologic such as eye, muscle, and cardiac activities, and extraphysiologic such as line interference and electrode noise. Extraphysiologic artifact can often be removed using appropriate filtering techniques as there is spectral separation. however physiologic artifact removal requires careful attention as they can be within the same frequency range of the EEG signal and are aperiodic. Ocular artifacts (OA) due to eye movement and eye blinks are dominant over other contaminating physiologic artifacts [3]. As EEG signal can be used for analyzing different diseases [4–6], monitoring brain engagement [7, 8], different techniques have been proposed for the removal of OA from EEG to make it more reliable for different purposes. The widely used methods for removing OAs are based on regression in time domain [9] and frequency domain [10]. But these methods need the recording of Electrooculogram (EOG),and can also result in the elimination of neural activities [11]. Statistical techniques like Principal Component Analysis (PCA) [12], Kurtosis [13], Independent Component Analysis (ICA) [14] and Multiscale sample entropy [15] are also shown to be

effective to remove OA, but they rely on multiple channel data. One of the robust and promising ocular artifact removal techniques for single channel EEG data is Wavelet Transform (WT) [11, 18].

T. Zikov *et al.* discussed applying stationary wavelet transform with *coif3* wavelet filters to denoise the EEG signal [8]. In their proposed study, 60-second baseline EEG was recorded to calculate threshold required for denoising. Use of *haar* wavelet is explored in detecting changes in the state of the eye (eye-blinks and eyeball movements) [17]. Stationary wavelet transform with *coif3* as a basis function with various non-adaptive thresholding methods have also been demonstrated [18]. Another wavelet-based approach of removing ocular artifact was to use stationary wavelet transform with *sym3* as a basis function and to use the coefficient of variation to detect and denoise the artifact [19].

With the advent of ambulatory and miniaturist body-worn EEG systems with few channels for routine monitoring [20, 21, 22], there is a growing need to develop effective OA removal technique that can operate on few channel EEG data. For real-time applications like mental state classification, comfort sensing, emotion sensing, movement prediction etc., algorithms should perform reasonably with short epoch of streaming EEG data. Many brain-computer interfacing (BCI) systems are utilized for routine and continuous monitoring of brain activities for epilepsy [23], autistic spectrum disorder (ASD) [24], and Alzheimer's patients [25]. Hence, removal of artifacts in real-time with the access of few channel (especially single channel) is of research interest. Though there is a vast amount of literature available on using wavelet transform for ocular artifact removal from EEG data [20–22], a little have investigated the effects of using various

possible forms of WT for single channel OA removal. The evaluation of existing WT techniques is critical in order to find the efficient, reliable and unsupervised way of denoising OA for real time BCI applications.

This paper compares different combinations of wavelet decomposition techniques, thresholds and mother wavelets. Specifically, we present a comparative study of discrete and stationary wavelet transform using four basis functions: *haar*, *coif3*, *sym3*, and *bior4.4* with Universal and Statistical Thresholding for OA removal from single channel EEG data. These combinations are carefully selected as they are commonly used for OA removal [19–22, 25]. Furthermore, we present objective performance metrics using multiple statistical measures in time domain and frequency domain.

2.2 Wavelet Transform and Performance Metrics

2.2.1 Wavelet Transform (WT) Decomposition

WT can be applied to any single channel EEG data to remove OAs without information from any other EEG or EOG channels. WT decomposes a time-varying signal into its set of basis functions known as wavelets. These basis functions known as wavelets are obtained by performing dilations and shifting of the mother wavelet:

$$\Psi_{a,b}(t) = \Psi(\frac{t-b}{a}) \tag{1}$$

where a is the scaling parameter and b is the shifting parameter [13]. In this study, we have implemented multi-level wavelet decomposition in order to get precise information

about the wavelet coefficients. In addition to ocular artifact removal, WT is proven to be a robust tool in several applications like machine condition monitoring [27], hologram analysis [28], pitch detection of speech signals [29], multi -modality medical image fusion [30], fault detection in a spur gear [31], power quality analysis [32], signal processing in white-light interferometry [33].

2.2.2 Discrete Wavelet Transform (DWT)

DWT is considered non-redundant and highly efficient wavelet transform to obtain discrete wavelet representation of signals [34, 35]. In DWT, the input signal is passed through a low pass and high pass filter to get approximate coefficients (*ak*) and detail coefficients (*dk*), respectively, where k represents the level of decomposition (Fig. 1). This process is repeated until the desired frequency range is obtained. At each stage, the filter output is down-sampled by 2, later up-sampled to reconstruct the signal. In this study, we have used in-built *wavedec* function in Matlab (MathWorks, Natick, MA) to implement DWT in the denoising algorithm.

2.2.3 Stationary Wavelet Transform (SWT)

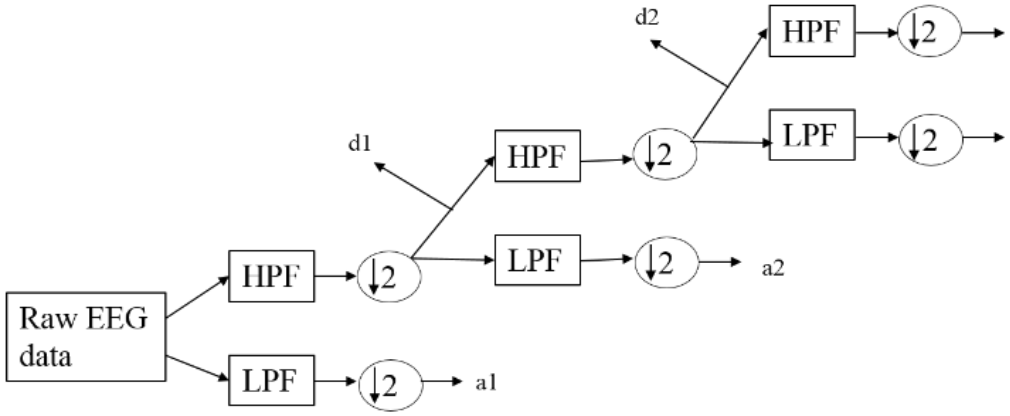


Fig. 1: Graphical representation of DWT decomposition. (HPF: High pass filter, LPF: Low pass filter.)

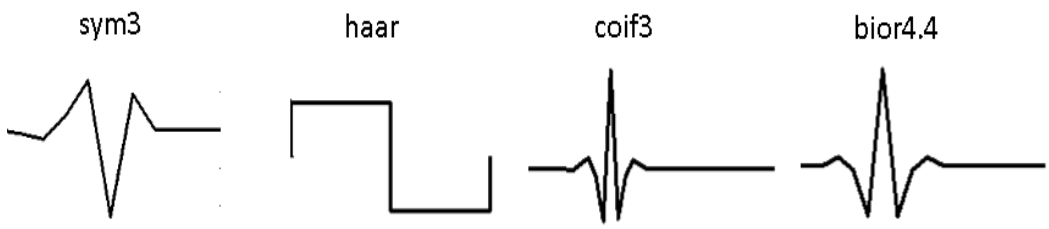


Fig. 2: Examples of common wavelet basis functions that can be applied for artifact removal from EEG data. (Plotted using Matlab built-in functions.)

The major drawback of DWT is its time-variance, which is particularly important in statistical signal processing applications such as EEG [36]. SWT overcomes this translation-invariance drawback of DWT but has redundant information and is relatively slow [37]. The design difference between DWT and SWT is the filter at each stage [38]. The approximate and detail sequences at each level of decomposition are of the same length as the original sequence. After obtaining the coefficients at *jth* level, the algorithm up samples the filter coefficients by a factor of $2j-1$. It has been implemented by the *swt* function of MATLAB in this study.

2.2.4 Wavelet Basis Functions

WT of the EEG signals yields the wavelet coefficients which represents the correlation between EEG signal and the wavelet basis functions. Fig. 2 represents some commonly used WT basis functions utilized in the literature for OA removal. For eye-blink removal, these wavelets perform well as they resemble the characteristics of these eye blinks [16, 18, and 19]. In this paper, we have compared the performance of widely used symlet (sym3), haar (haar), coiflet (coif3), and biorthogonal (bior4.4) wavelets.

2.2.5 Wavelet Thresholding for Denoising

The approximate and detail coefficients of the decomposed EEG needs to be denoised to separate the artifactual coefficients from neural signal coefficients. For thresholding the wavelet coefficients, two commonly used metrics: Universal Threshold (UT) and Statistical Threshold (ST) are evaluated in this paper.

UT is implemented as:

$$K = \sqrt{2\log N}\sigma \qquad (2)$$

$$\sigma^2 = \text{median}\left(\frac{|C_a|}{0.6745}\right) \qquad (3)$$

where, K is the estimation of neuronal wide band signal magnitude using UT, N is the length of data to be processed, C_a is the wavelet coefficients at a^{th} level of decomposition that undergoes thresholding, and 0.6745 is the constant value for Gaussian noise [15].

We also implemented ST (proposed by Krishnaveni *et al.* in [18]) in our study which is based on the statistics of the signal. As discussed in [19], the proposed threshold produces better de-noised results than the other conventional thresholds. Mathematically, the proposed ST is formulated as:.

$$T = 1.5 * \text{std}(H_k) \qquad (4)$$

where T is the estimation of neuronal wideband signal magnitude using ST, $\text{std}(H_k)$ employs standard deviation of wavelet coefficients at k^{th} level. In both cases,

hard thresholding is applied, where wavelet coefficient is removed if the absolute value of wavelet coefficient is greater than the threshold.

2.2.6 Performance Metrics

Different statistical performance metrics have been used to objectively compare various combinations of OA removal in time and frequency domain. For time domain comparison, correlation coefficient, mutual information, signal to artifact ratio and normalized mean square error have been evaluated. For frequency domain comparison, time frequency analysis has been utilized.

Correlation Coefficient (CC) is a statistical method that shows the degree of association between two variables. Suppose C(t1,t2) is the auto-covariance of a process x(t), or in another way, *C (t1, t2)* is the covariance of the random variables *x (t1)* and *x (t2)*, then correlation coefficient of the process *x (t)* is [39]:

$$\mathrm{r}\ (\mathrm{t1}, \mathrm{t2}) = \frac{\mathrm{C}\ (\mathrm{t1}, \mathrm{t2})}{\sqrt{\mathrm{C}\ (\mathrm{t1}, \mathrm{t1})\mathrm{C}\ (\mathrm{t2}, \mathrm{t2})}} \tag{5}$$

Mutual Information (MI) is used statistically to measure how much information one random variable contains about the other random variable. If U and V are two partitions of sample space S, then information about U contained in V or information about V contained in U is:

$$\mathrm{I}\ (\mathrm{U}, \mathrm{V}) = \mathrm{H}(\mathrm{U}) + \mathrm{H}(\mathrm{V}) - \mathrm{H}\ (\mathrm{U.V}) \tag{6}$$

I (U, V) represents mutual information [39].

Signal to Artifact Ratio (SAR) is a quantification method to measure the amount of artifact removal in a specific signal after processing with an algorithm [40]. If z is the EEG signal containing artifact and $\hat{z}$ is the signal obtained after running an artifact free algorithm. Hence,,

$$SAR = 10\log(\frac{std(z)}{\mathrm{std}(z - \hat{z})}) \tag{7}$$

Normalized Mean Square Error (NMSE) approximates the difference between the ideal and actual data [11]. NMSE is computed in dB using the equation:

$$NMSE = 20logE\{\frac{\sum[x1(i) - x2(i)]^2}{\sum[x1(i)]^2}\} \tag{8}$$

Time and frequency components can be analyzed simultaneously using the wavelet decomposition tool of EEGLAB toolbox (Matlab, CA, US). This allows qualitative comparison of the signals before and after artifact denoising.

2.3 Experimental Method

For EEG acquisition, a 14-channel referential montage EPOC headset (Emotiv, Eveleigh, NSW, Australia) at a sampling rate of 128 sps was used in the lab setting. Before data acquisition, the skin of the subject was cleaned using Nuprep skin preparing gel (Weaver and Company, Aurora, CO) and mild abrasive strips to remove the dead skin and thereby moisten the skin. Data were collected from four subjects (two males, two females) at AF3, AF4, F3, F4, F7, F8, FC5, FC6, P7, P8, T7, T8, O1, and O2 channel locations in a closed room for 1 min 45 s. During the recording, subjects were instructed to blink 9 times with a 5 s hiatus. As OAs are prominent in the frontal lobe, only AF3 channel data was used for analysis in this study. Out of two sessions/subject data

recording, 7 datasets were used for this study using an approved Institutional Review Board protocol (University of Memphis IRB# 2289).

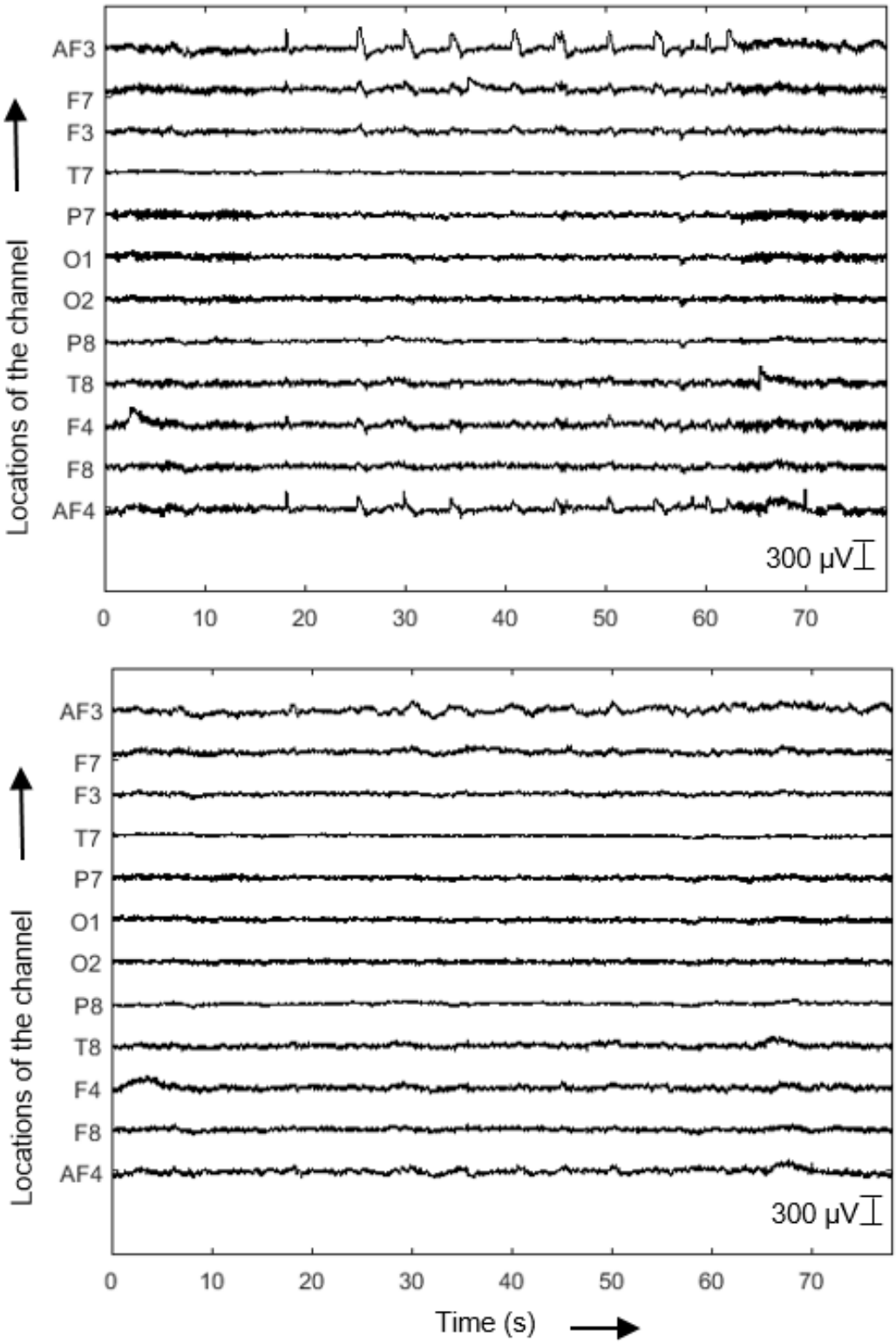


Fig. 3: 12-channel EEG data from the Dataset 4 (Top) Raw (Below) OA artifact-free. X-axis is time in seconds and Y-axis is amplitude in microvolts.

As OAs are prominent in the frontal lobe, most of the comparison plots in this study are from AF3 channel. However, as discussed in the literature, WT can be applied to denoise any channel location, as depicted in Fig. 3. OAs occur due to eye movement and eye-blinks and have frequency ranges of 0-7 Hz and 8-13 Hz, respectively [33]. To accurately identify artifact related wavelet coefficients, we have implemented multi-level wavelet decomposition using SWT or DWT. The decomposition was done up to level 8 (level 8: 0.25-0.5 Hz, level 7: 0.5-1 Hz, level 6: 1-2 Hz, level 5: 2-4 Hz, level 4: 4-8 Hz, level 3: 8-16 Hz, level 2: 16-32 Hz and level 1: 32-64 Hz) to obtain the frequency range of interest. For denoising the wavelet coefficients, thresholding has been done over the detail coefficients from level 8 up to level 3. As the sampling rate of the dataset is 128 sps, decomposing up to level 3 gives us the required ocular related wavelet coefficients for denoising. Either UT or ST is implemented for thresholding and sym3, haar, coif3 or bior4.4 have been used as mother wavelet. With two wavelet decomposition techniques (DWT/SWT), two thresholds (UT/ST) and four mother wavelets (sym3/haar/coif3/bior4.4), 16 combinations or methods are possible to remove OA from EEG. The outputs of these combinations are quantified and compared using different performance metrics.

2.4 Results and Analysis

Fig. 4 compares raw and OA-artifact free EEG data using coif3 wavelet basis function with statistical threshold to denoise single channel EEG data using SWT and DWT techniques. Careful observation indicates that SWT produced cleaner processed signals, however, DWT is a faster method, which is an important aspect of real time data processing (e.g. streaming data). Most OA removal algorithms affected neuronal signals

where there is no blink artifact (non-blink regions). To the distinguish performance of an algorithm to remove OA, while preserving neuronal signals, we have segregated the raw EEG data into “Blink regions” and “Non-blink regions”.

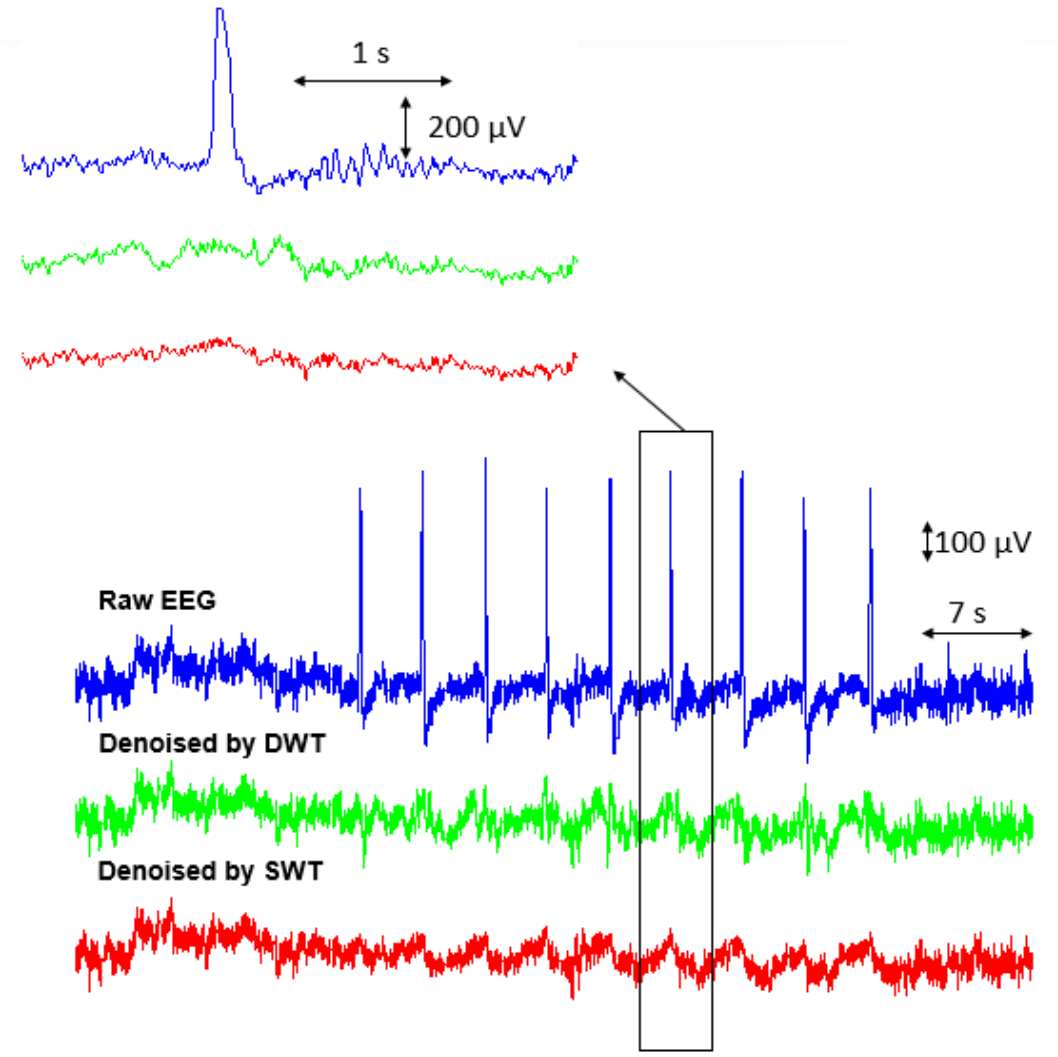


Fig. 4: Comparison of a section of EEG data from AF3 channel (Dataset 1) before and after denoising using SWT and DWT decomposition techniques with coif3 wavelet.

CC and MI have been calculated between raw and re- constructed signals for blink and non-blink regions of the entire EEG data. The ideal eye blink removal algorithm would produce high CC and MI values in the non-blink region, while low CC and MI values in the blink region [41-42]. Fig. 5 and 6 show the statistical analysis of the CC and

MI metrics (averaged over 7 datasets) for both blink and non-blink regions of AF3 channel EEG data.

In both cases, DWT with UT performs better in an eye-blink region than all other WT threshold combinations. Among them, *DWT+UT+sym3* gives the lowest value of CC, while *DWT+UT+haar* gives the lowest value of MI. However, the efficacy of DWT with UT to preserve neuronal information in a non-blink region is poor. This might be due to spectral shift introduced by DWT technique. Similarly, among all cases, the SWT with ST generates higher values of CC and MI than all other WT threshold combinations for non-blink regions. Among these, *SWT+ST+haar* provides the highest values of CC and MI in non-blink regions. Based on CC and MI metric, SWT with UT reveals that neuronal signals retention is poor, but removes or changes the blink zone in a greater amount. SWT with ST retains neuronal signals based on both CC and MI metrics, even though OA removal performance based on CC and MI is not as good as other methods. Applying DWT with UT doesn't show the power of preserving neural information based on CC and MI metric. DWT with ST is excellent to retain neuronal information based on CC metric for all wavelets, but performance is dependent on wavelet types based on MI metric, where *haar* wavelet outperforms other basis functions.

According to the definitions, the technique that produces the higher value in the SAR and the lower value in the NMSE is considered to be more effective. Table I depicts the values of SAR for different methods. Based on SAR, SWT+ST combination performs superior to others. It is noted from Table I that, in general, SAR values are higher when ST is used with any types of wavelet, indicating ST is aggressive in eliminating probable

artifacts, while UT is conservative. DWT+ST performs better after SWT+ST combination.

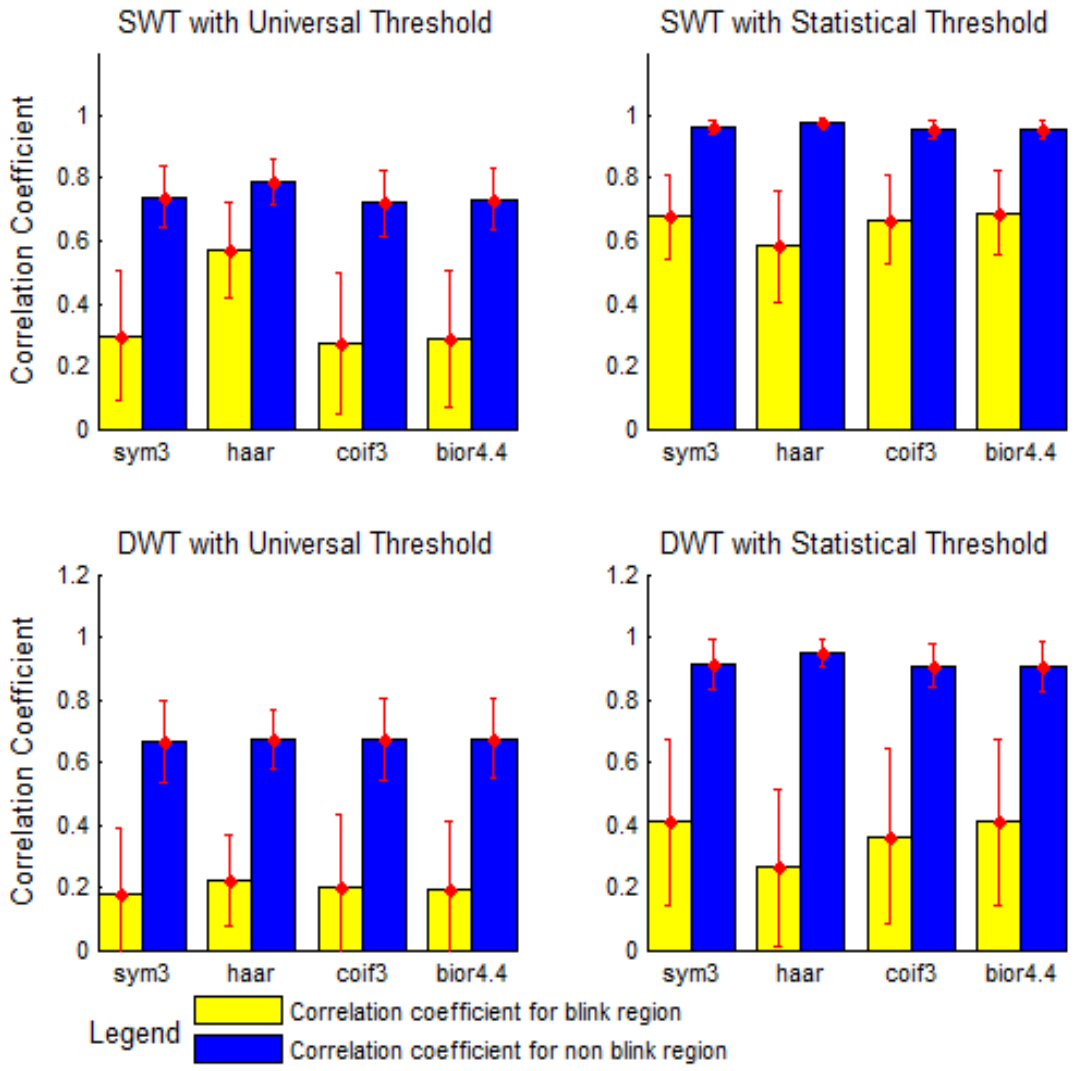


Fig. 5: Correlation coefficient comparison (N=7) for blink and non-blink EEG data using UT and ST thresholds with SWT and DWT methods.

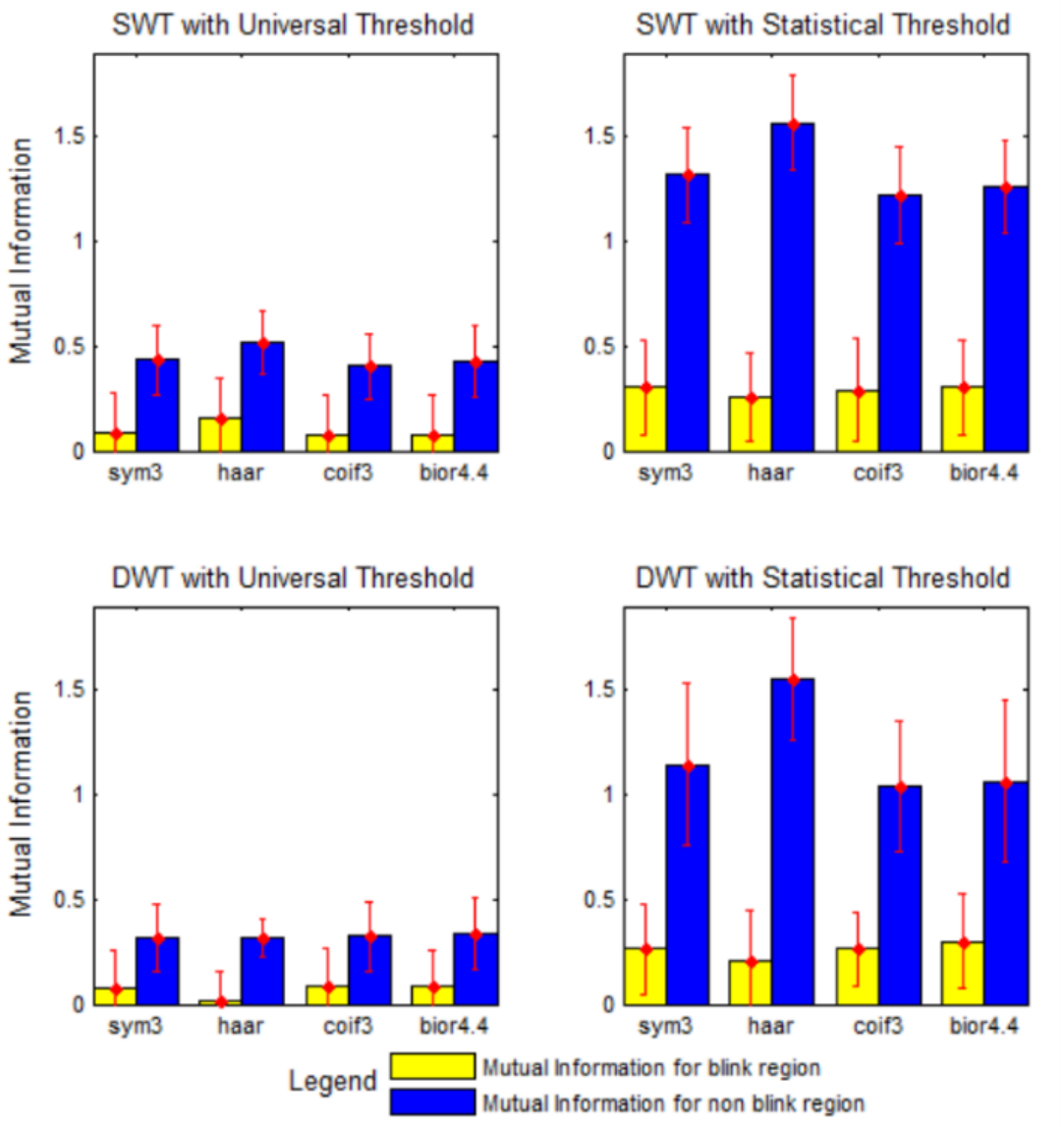


Fig. 6: Mutual Information comparison (N=7) for blink and non-blink EEG data using UT and ST thresholds with SWT and DWT methods.

Table II depicts the values of NMSE for different methods. Based on Table II, SWT+ST again outperforms other methods, while DWT+ST is the second best, compared to other WT threshold combinations. As the lower NMSE indicates better technique, both SWT and DWT show superior performance with ST for any type of wavelet basis functions. NMSE values are lower when ST is applied with both SWT and DWT using any type of wavelet basis functions.

Time-frequency analysis results are shown along with the raw signal (Dataset 1, AF3 channel location) in Fig. 7. Few DWT methods are observed to have introduced new artifactual noise in the processed data during the OA removal throughout the spectrum (e.g. DWT+UT+haar, DWT+ST+haar). DWT with UT is also noted to decrease the overall magnitudes of neuronal signals. Within DWT results, ST with coif3 and bior4.4 seems to retain neuronal signals effectively while minimizing OA. Further, in order to analyze over frequency domain, magnitude squared coherence measure is calculated between the raw and OA-artifact free EEG data and is plotted for these two combinations in Fig. 8. Magnitude squared spectral coherence estimate between 0 and 1 corresponds to how well the two-signals, a and b, relates at various frequencies and is mathematically calculated as:

$$C_{ab}(\mathrm{f}) = \frac{P_{ab}(f)^2}{P_{aa}(f)P_{bb}(f)} \tag{9}$$

where $P_{ab}(f)$ is the cross power spectral densities of the signal, $P_{aa}(f)$ is power spectral density of raw EEG signal and $P_{bb}(f)$ is power spectral density of OA-artifact free EEG signal. It has been implemented in this study using mscohere function of Matlab. It has been observed that all of the SWT combinations show efficacy in preserving neuronal signals and do not introduce new artifacts like some of the DWT combinations.

.

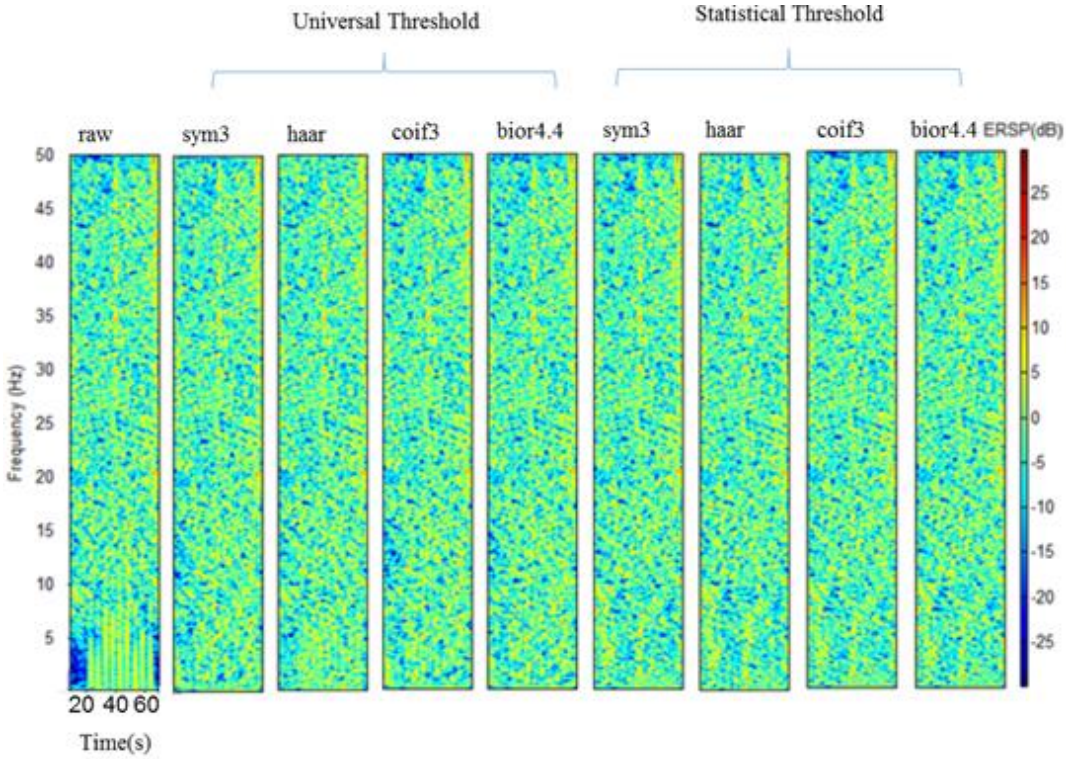
TABLE 1. SAR ON EEG DATASETS USING UT AND ST THRESHOLDS WITH SWT AND DWT METHODS

WT+Thresh.	**sym3**	**haar**	**coif3**	**bior4.4**
SWT + UT	1.42±0.61	1.48±0.58	1.40±0.61	1.42±0.61
SWT + ST	**2.33±0.86**	2.18±0.79	2.32±0.85	2.28±0.85

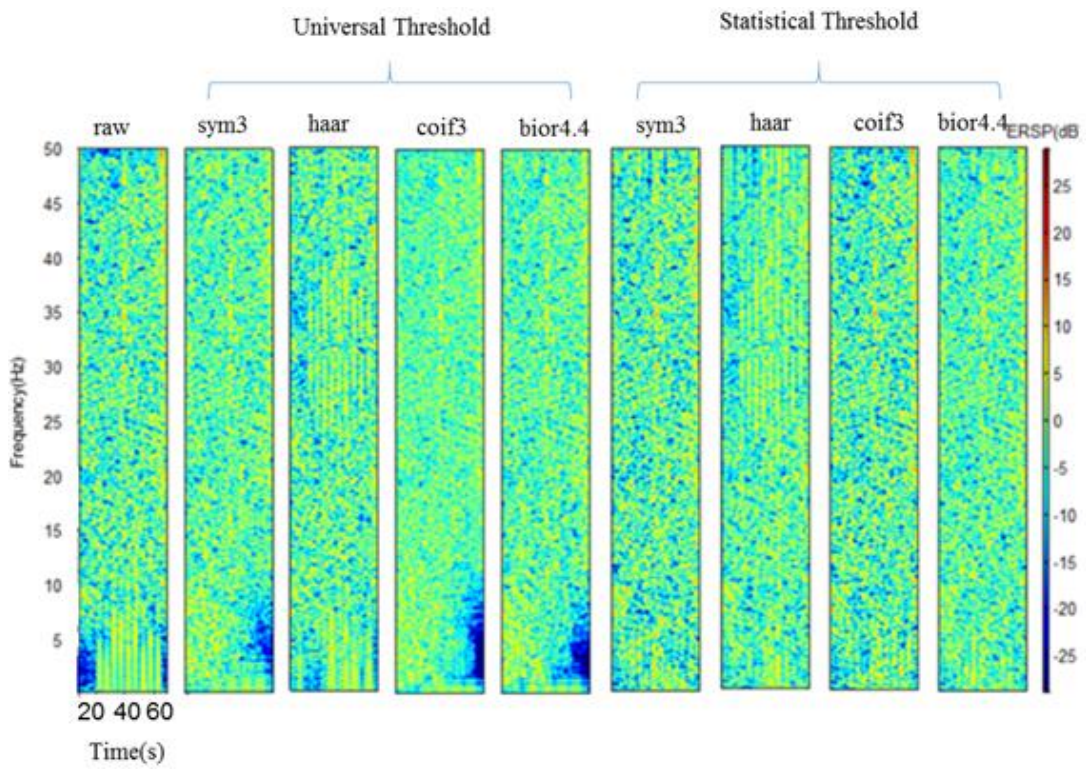
DWT + UT	1.28±0.63	1.13±0.49	1.31±0.63	1.3±0.62
DWT + ST	1.93±0.82	1.68±0.65	1.89±0.80	1.89±0.78

TABLE 2. NMSE ON EEG DATASETS USING UT AND ST THRESHOLDS WITH SWT AND DWT METHODS

WT+Thresh.	**sym3**	**haar**	**coif3**	**bior4.4**
SWT + UT	-5.88±2.37	-6.11±2.23	-5.79±2.36	-5.85±2.37
SWT + ST	**-8.92±3.25**	-8.47±2.96	-8.84±3.2	-8.77±3.21
DWT + UT	-5.31±2.5	-4.7±1.94	-5.43±2.49	-5.39±2.45
DWT + ST	-7.97±3.18	-6.9±2.49	-7.8±3.1	-7.82±3.02

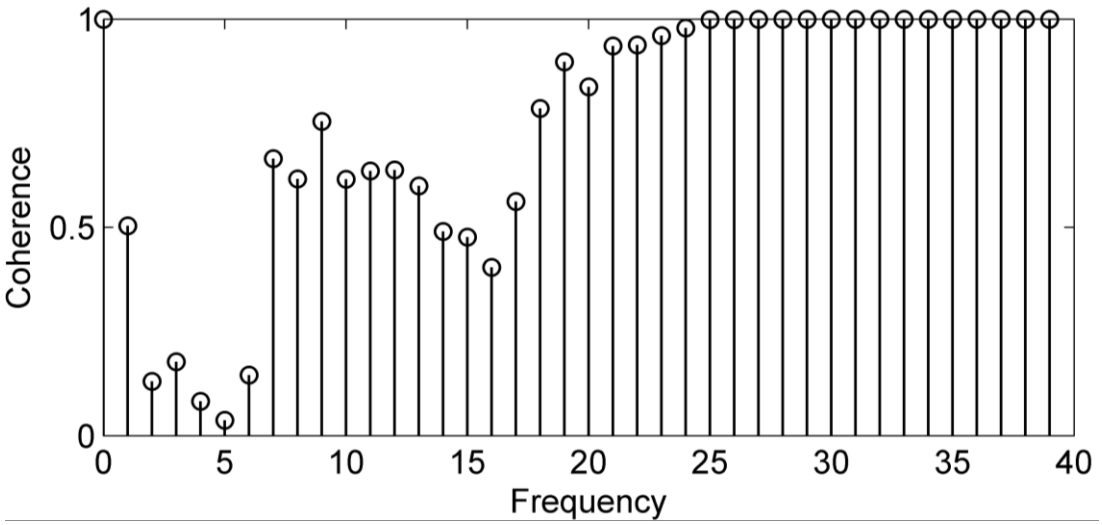


(a)

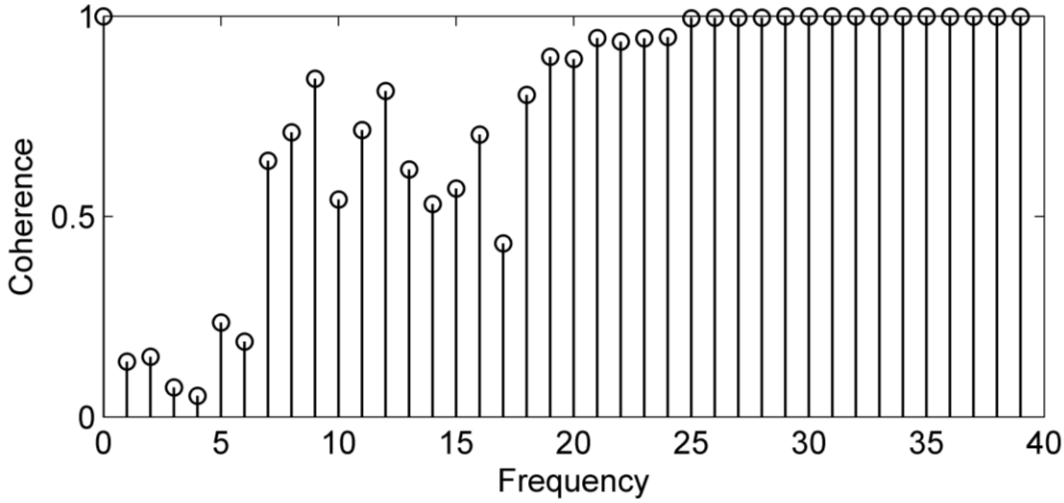


(b)

Fig. 7: Time- Frequency comparison plots of various wavelets and thresholds for OA artifact denoising technique using (a) SWT (b) DWT. Raw EEG signal is from prefrontal AF3 channel location of Dataset 1.



(a)



(b)

Fig. 8: Magnitude squared coherence measure plot for one subject from AF3 location after denoising with (a) DWT+ST+coif3 (b) DWT+ST+bior4.4 combination.

2.5 Discussion and Conclusions

Brain monitoring using a few EEG sensors in non-clinical settings has drawn a lot of attention lately. These real time BCI applications like cognitive load assessment, diagnosis of brain disorders, fatigue prediction, and cognitive biometrics, need fast and efficient pre-processing algorithms in order to process and analyze raw brain signals

reliably in real time. The most common artifact in the EEG signal is due to the ocular activity. This study, therefore, focusses on comparing the effectiveness of commonly used wavelet based techniques for ocular artifact removal in a single channel EEG system. This will allow us to determine the optimal wavelet decomposition technique and corresponding threshold to denoise EEG signal effectively.

In this paper, data from AF3 channel has been presented as a representative of EEG signal contaminated with artifacts to compare several WT based methods. However, the algorithm is not specific to the channel and is applicable to EEG recordings from any channel. Based on CC and MI metrics, DWT with ST using *haar* wavelet is found to be more effective than *DWT+ST+coif3* or *DWT+ST+bior4.4*, but time-frequency analysis shows higher distortion in the processed data using this combination. As these metrics analyze different performances of the algorithm, so none of the combinations was superior based on all metrics. Based on these results, DWT with ST using *coif3* and *bior4.4* wavelet basis functions have performed well for OA removal while preserving neuronal signals in the non-blink regions based on CC, MI, SAR, NMSE and time-frequency analysis.

As fast algorithms are required for real-time systems, a trade-off must be considered for an efficacious method with low distortion. It is known that DWT is a faster technique requires less computational resources than SWT for real-time analysis [43]. According to the results presented in this paper, *DWT+ST+coif3* or *DWT+ST+bior4.4* could be an optimum choice. For the applications, where computation time is not critical, SWT with *haar* wavelet can be used with the statistical threshold. Instead of applying OA removal algorithm over the whole dataset as outlined in this paper, another approach is to identify

blink regions and apply OA removal algorithm only to these OA regions to develop a faster OA removal technique, which we have reported previously [44]. Our future research directions include hardware implementation and optimization of efficacious OA removal technique for a single channel EEG system, real-time OA removal, feature extraction, and cognitive load classification to monitor brain engagement in natural environment within a wearable embedded system.

2.6 References

[1] W. O. Tatum, A. Husain, S. R. Benbadis, and P. W. Kaplan, "Handbook of EEG Interpretation", Demos Medical Publishing: USA, 2007.

[2] V. Sakkalis, "Review of advanced techniques for the estimation of brain connectivity measured with EEG/MEG", *Computers in Biology and Medicine*, vol. 41, pp. 1110–1117, 2011.

[3] L. Vigon, M. R. Saatchi, J. E. W. Mayhew, *et al.*, "Quantitative evaluation of techniques for ocular artefact filtering of EEG waveforms," *IEE Proceedings-Science Measurement and Technology*, vol. 147, no. 5, pp. 219-228, 2000.

[4] M. Penttila, J. V. Partanen, H. Soininen et al., "Quantitative-analysis of occipital EEG in different stages of Alzheimer's-disease," *Electroencephalography and Clinical Neurophysiology*, vol. 60, no. 1, pp. 1-6, 1985.

[5] R. P. Brenner, R. F. Ulrich, D. G. Spiker et al., "Computerized EEG spectral-analysis in elderly normal, demented and depressed subjects," *Electroencephalography and Clinical Neurophysiology*, vol. 64, no. 6, pp. 483-492, Dec, 1986.

[6] R. Soikkeli, J. Partanen, H. Soininen et al., “Slowing of EEG in Parkinson's disease,” *Electroencephalography and clinical neurophysiology*, vol. 79, no. 3, pp. 159-165, 1991.

[7] R. Mahajan, C. A. Majmudar, S. Khatun, B. I. Morshed, and G. M. Bidelman, “NeuroMonitor Ambulatory EEG Device: Comparative Analysis and Its Application for Cognitive Load Assessment”, *IEEE Healthcare Innovations and Point-of-Care Technologies Conf.*, (in press), 2014.

[8] A. Gevins, M. E. Smith, H. Leong et al., “Monitoring working memory load during computer-based tasks with EEG pattern recognition methods,” *Human Factors*, vol. 40, no. 1, pp. 79-91, Mar, 1998.

[9] P. He, G. Wilson, C. Russell et al., “Removal of ocular artifacts from the EEG: a comparison between time-domain regression method and adaptive filtering method using simulated data,” *Medical & Biological Engineering & Computing*, vol. 45, no. 5, pp. 495-503, 2007.

[10] J. C. Woestenburg, M. N. Verbaten, and J. L. Slangen, “The removal of the eye-movement artifact from the EEG by regression-analysis in the frequency-domain,” *Biological Psychology*, vol. 16, no. 1-2, pp. 127-147, 1983.

[11] M. T. Akhtar, W. Mitsuhashi, and C. J. James, “Employing spatially constrained ICA and wavelet denoising, for automatic removal of artifacts from multichannel EEG data,” *Signal Processing*, vol. 92, no. 2, pp. 401-416, 2012.

[12] T. D. Lagerlund, F. W. Sharbrough, and N. E. Busacker, “Spatial filtering of multichannel electroencephalographic recordings through principal component

analysis by singular value decomposition," *Journal of Clinical Neurophysiology*, vol. 14, no. 1, pp. 73-82, 1997.

[13] G. Inuso, F. La Foresta, N. Mammone, *et al.*, "Brain activity investigation by EEG processing: wavelet analysis, kurtosis and Renyi's entropy for artifact detection", *IEEE Intl Conf. Information Acquisition*, Seogwipo-si, pp. 195-200, 2007.

[14] N. P. Castellanos, and V. A. Makarov, "Recovering EEG brain signals: Artifact suppression with wavelet enhanced independent component analysis," *Journal of Neuroscience Methods*, vol. 158, no. 2, pp. 300-312, 2006.

[15] R. Mahajan, and B. Morshed, "Unsupervised eye blink artifact denoising of EEG data with modified multiscale sample entropy, kurtosis and wavelet-ICA," *IEEE J. Biomedical and Health Informatics*, (in press) 2015.

[16] T. Zikov, S. Bibian, G. A. Dumont, *et al.*, "A wavelet based de-noising technique for ocular artifact correction of the electroencephalogram", *IEEE Engineering in Medicine and Biology Conf.*, vol. 1, pp. 98-105, 2002.

[17] S. V. Ramanan, N. Kalpakam, and J. Sahambi, "A novel wavelet based technique for detection and de-noising of ocular artifact in normal and epileptic electroencephalogram," IEEE International Conference on Communications, Circuits and Systems, vol. 2, pp. 1027-1031, 2004.

[18] V. Krishnaveni, S. Jayaraman, N. Malmurugan, A. Kan- dasamy, and D. Ramadoss, "Non adaptive thresholding methods for correcting ocular artifacts in EEG," *Academic Open Internet Journal*, vol. 13, 2004.

[19] P. S. Kumar, R. Arumuganathan, K. Sivakumar, *et al.*, "Removal of Ocular Artifacts in the EEG through Wavelet Transform without using an EOG Reference Channel*," Int. J. Open Problems Compt. Math*, vol. 1, no. 3, pp. 188-200, 2008.

[20] "EEG Features". Available online: https://emotiv.com/epoc.php. (Accessed: Dec 2015).

[21] "Muse". Available online: http://www.choosemuse.com/ (Accessed: Dec 2015).

[22] "Ultra-Portable 4 Channels Wireless EEG Headset". Available online: http://www.advancedbrainmonitoring.com/xseries/ (Accessed: Dec 2015).

[23] H. Ocak, "Automatic detection of epileptic seizures in EEG using discrete wavelet transform and approximate entropy," *Expert Systems with Applications*, vol. 36, no. 2, pp. 2027-2036, 2009.

[24] M. Ahmadlou, H. Adeli, and A. Adeli, "Fractality and a Wavelet-Chaos-Neural Network Methodology for EEG-Based Diagnosis of Autistic Spectrum Disorder," *Journal of Clinical Neurophysiology*, vol. 27, no. 5, pp. 328-333, 2010.

[25] A. Cichocki, S. L. Shishkin, T. Musha, *et al*., "EEG filtering based on blind source separation (BSS) for early detection of Alzheimer's disease," *Clinical Neurophysiology*, vol. 116, no. 3, pp. 729-737, 2005.

[26] H. Ghandeharion and A. Erfanian, "A fully automatic ocular artifact suppression from EEG data using higher order statistics: Improved performance by wavelet analysis," Medical engineering & physics, vol. 32, no. 7, pp. 720–729, 2010.

[27] Z. K. Peng, and F. L. Chu, “Application of the wavelet transform in machine condition monitoring and fault diagnostics: a review with bibliography,” *Mechanical Systems and Signal Processing*, vol. 18, no. 2, pp. 199-221, 2004.

[28] C. Buraga-Lefebvre, S. Coetmellec, D. Lebrun, *et al.*, “Application of wavelet transform to hologram analysis: three-dimensional location of particles,” *Optics and Lasers in Engineering*, vol. 33, no. 6, pp. 409-421, 2000.

[29] S. Kadambe, and G. F. Boudreauxbartels, “Application of the wavelet transform for pitch detection of speech signals,” *IEEE Transactions on Information Theory*, vol. 38, no. 2, pp. 917-924, 1992.

[30] A. Wang, H. Sun, Y. Guan, *et al.*, "The application of wavelet transform to multi-modality medical image fusion," *IEEE Intl. Conf. on Networking, Sensing and Control*, pp. 270-274, 2006.

[31] W. Staszewski, and G.Tomlinson, “Application of the wavelet transform to fault-detection in a spur gear,” *Mech. Systems and Signal Processing*, vol. 8, no. 3, pp. 289-307, 1994.

[32] L. Angrisani, *et al.*, “Measurement method based on the wavelet transform for power quality analysis,” *IEEE Transactions on Power Delivery*, vol. 13, no. 4, pp. 990-998, 1998.

[33] P. Sandoz, “Wavelet transform as a processing tool in white-light interferometry,” *Optics Letters*, vol. 22, no. 14, pp. 1065-1067, 1997.

[34] S. G. Mallat, “A theory for multiresolution signal decomposition the wavelet representation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, no. 7, pp. 674-693, 1989.

[35] V. J. Samar, A. Bopardikar, R. Rao, *et al.*, "Wavelet analysis of neuroelectric waveforms: A conceptual tutorial," *Brain and Language*, vol. 66, no. 1, pp. 7-60, 1999.

[36] J. C. Pesquet, H. Krim, and H. Carfantan, "Time-invariant orthonormal wavelet representations," *IEEE Transactions on Signal Processing*, vol. 44, no. 8, pp. 1964-1970, 1996.

[37] P. S. Kumar, R. Arumuganathan, K. Sivakumar, *et al.*, "An adaptive method to remove ocular artifacts from EEG signals using wavelet transform," *J. Appl. Sci. Res*, vol. 5, no. 7, pp. 711-745, 2009.

[38] G. P. Nason, and B. W. Silverman, "The stationary wavelet transform and some statistical applications", *Wavelets and statistics*, pp. 281-299: Springer, 1995.

[39] A. Papoulis, and S. U. Pillai, "Probability, random variables, and stochastic processes", Tata McGraw-Hill Education, 2002.

[40] M. H. Soomro, N. Badruddin, M. Z. Yusoff, et al., "A Method for Automatic Removal of Eye Blink Artifacts from EEG Based on EMD-ICA," IEEE Intl. Colloquium on Signal Processing and Its Applications, pp. 129-134, 2013.

[41] P.T. Kavitha, et al., "Modified ocular artifact removal technique from EEG by adaptive filtering," 2007 6th International Conference on Information, Communications & Signal Processing, vol. 1-4, 2007, pp. 1653-1657.

[42] S. Valipour, et al., "Study on Performance Metrics for Consideration of Efficiency of the Ocular Artifact Removal Algorithms for EEG Signals," Indian Journal of Science and Technology, vol. 8, no. 1, 2015.

[43] S. Khatun, R. Mahajan, and B. Morshed I., "Comparative analysis of wavelet based approaches for reliable removal of ocular artifacts from single channel EEG," *IEEE Inter- national Conference on Electro/Information Technology*, 2015.

[44] C. A. Majmudar, R. Mahajan, and B. I. Morshed, "Real- time hybrid ocular artifact detection and removal for single channel EEG for consideration," *IEEE International Conference on Electro/Information Technology*, 2015.

3. A SINGLE-CHANNEL EEG-BASED APPROACH TO DETECT MILD COGNITIVE IMPAIRMENT VIA SPEECH-EVOKED BRAIN RESPONSES

Abstract— Mild Cognitive Impairment (MCI) is the preliminary stage of Dementia, which may lead to Alzheimer's disease (AD) in the elderly people. Therefore, early detection of MCI has the potential to minimize the risk of AD by ensuring the proper mental health care before it is too late. In this study, we demonstrate a single-channel EEG based MCI detection method, which is cost-effective and portable, and thus suitable for continuous patient monitoring. We collected the scalp EEG data from 23 subjects, while they were stimulated with five auditory speech signals. The cognitive state of the subjects was evaluated by the Montreal Cognitive Assessment Test (MoCA). We extracted 590 features from the Event-Related Potential (ERP) of the collected EEG signals, which included time and spectral domain characteristics of the response. The top 25 features, ranked by the random forest method, were used for classification models to identify subjects with MCI. Robustness of our model was tested using leave-one-out cross-validation while training the classifiers. Best results (leave-one-out cross-validation accuracy 87.9%, sensitivity 84.8%, specificity 95%, and F score 85%) were obtained using support vector machine (SVM) method with Radial Basis Kernel (RBF) (sigma = 10/cost = 102). Similar performances were also observed with logistic regression (LR), further validating the results. Our results suggest that single-channel EEG could provide a robust biomarker for early detection of MCI.

Index Terms— Electroencephalography, Event-related potential, Mild cognitive impairment, Speech-Evoked Brain Responses.

3.1 Introduction

Memory impairment due to aging is common. However, if the level of impairment progresses beyond what is expected in normal aging, mild cognitive impairment (MCI) can ensue [1]. MCI is a prodromal state of cognitive aging between changes deriving from natural/normal aging and dementia [2-3]. Alzheimer's disease (AD) is the most prevalent dementia [4-7] for elderly people [1, 3] in many countries, and it is accompanied by progressively worsening memory, reasoning, and other aspects of cognition [7-8]. As MCI is a preliminary stage of cognitive impairment, most often it is not treated properly despite the fact that there is up to 54% chance that MCI may lead to AD or related dementias [8]. The cost of providing care for the AD patients in the US was $200 billion in 2012 and it is estimated to grow to $1.1 trillion per year by 2050 [9]. Therefore, preventing this disease is of great importance for better healthcare as well as for the national financial interest. The root cause of this neurodegenerative disease is still unclear; Thereby, early MCI detection may play a critical role to enhance the management of the AD and dementia care.

Detection and characterization of MCI is an active field of research. Various physiological data such as resting-state functional magnetic resonance imaging (rs-fMRI) [4], structural magnetic resonance imaging (sMRI), diffusion tensor imaging (DTI) [5], positron emission tomography (PET) [10], fluorodeoxyglucose positron emission tomography (FDG-PET) [11], cerebrospinal fluid (CSF) [11,12], and magnetoencephalography (MEG) [13] are being investigated to study the effect of MCI

and its underlying biomarkers in elderly people. Although these physiological data provide multi-dimensional information about the brain, these methods are costly and also impractical in terms of portability. Relatively low-cost electroencephalography (EEG) is also being investigated to detect [2, 7, 14-19] and classify [17] MCI from other diseases that affect the cognitive state. There are different examples that multi-channel EEG data have been used to characterize MCI or AD using EEG signals in various literature such as: (i) mismatch negativity (MMN) and auditory P300 component from 256-channel EEG [2], (ii) features extracted by recurrence quantification analysis (RQA) and cross recurrence quantification analysis (CRQA) from 14-channel EEG [17], (iii) spectral features extracted from 19-channel EEG [7], (iv) auditory P2 component computed from 64-channel EEG [18], (v) the auditory mismatch negativity (MMN) from 19-channel EEG [19], (vi) ERP amplitude and latency from 256-channel EEG [20], (vii) accuracy and response time during low and high working memory conditions of memory task using 32-channel EEG [15] etc. Non-ERP based multi-channel EEG approach has also been investigated to detect MCI, e.g. using data from a 19-channel EEG system [16]. Low cost spontaneous speech data had also been used by researchers to detect MCI or AD [21 -24].

Although multi-channel EEG has been studied rigorously, we did not find any study that focuses on the single-channel EEG-based MCI detection and classification which incorporates optimal feature search. This kind of system can be integrated to the wearable headband (i.e. MUSE™ headband with a mobile application), which will allow the elderly people to assess their cognitive strength on a regular basis by their own. In this study, we have used single-channel EEG data from Fpz (near forehead) as this location is

considered optimal for analyzing auditory evoked potential [25-27] to classify between MCI and normal functioning individuals using five sound stimuli for better sensitivity in MCI detection [20, 28]. It is important to mention that EEG data may be affected by different types of artifacts such as ocular, muscle, electrode artifacts, which can be removed using either multi-channel [29] or single-channel [30-31] EEG data. Therefore, studying EEG data obtained from a single-channel is not hindered by the artifacts that might degrade the quality of the EEG signal. Initial findings of this study were reported elsewhere [32]. In this paper, we provide our complete and comprehensive study results.

3.2 Method

We designed our experiments with five contrastive speech sounds along a vowel continuum (/a/ vs. /u/) [33]. Subjects took part in an identification task where they were required to identify the vowel sounds while their EEG were collected. We obtained 590 features from the event-related potential (ERP) extracted from the EEG signal, which included time domain and spectral domain characteristics from the windowed ERP data. The top 25 features ranked by the random forest algorithm were used in several classification models that are widely used in the literature (Support Vector Machine (SVM) and Logistic Regression (LR)). We described our method of the study below in six steps: Subjects, Experimental Design, Data Collection, Event-Related Potential Processing, Features Extraction and Ranking, and Classification.

3.2.1 Subjects

The physiological data used in this study was collected from an experiment where twenty-three older adults (age ranges from 52-86 years; mean ± standard deviation: 70.2±7.2 yrs) participated. All the subjects were strongly right handed [28] with no

known history of psychiatric or neurological illness. The cognitive state of the participants was evaluated by the well-established Montreal Cognitive Assessment (MoCA) test [34]. In this assessment, among all the participants, fifteen older adults (8 male, 7 female) were normal (MoCA score ≥26 points; mean± standard deviation: 27.6±1.18; range: 26-30), and eight participants' (4 male, 4 female) were found to have MCI (MoCA score <26; mean ± standard deviation: 23.0±1.85; range: 20-25). It is worth mentioning that patients having Alzheimer's disease or severe dementia generate MoCA score within 11.4 to 21 [35]. The age of the MCI group and the control group was 74.6±3.3 years, and 67.5±8.2 years, respectively ($t_{21} = 2.34, p = 0.03$). The chi-square test for the gender of the two groups gave a p value of 0.81. The total years of formal education of the MCI, and normal group was 14.6±3.2 years, and 17.4±3.75 years, respectively ($t_{21} = -1.68, p = 0.11$). Participants in the study were compensated for their time and they gave their written consent under the protocol approved by the Baycrest Centre Ethics Committee (REB #06-31). The study was designed to observe the aging effect on the auditory system and to evaluate the cognitive state of the subjects. While recruiting the participants, exclusion criteria were based on age, musical training, handedness, and hearing loss [33].

3.2.2 Experiment design

The phonetic continuum was generated by varying the first formant (F1) frequency within 430 Hz and 730 Hz over five equal steps. Fundamental (F0), second (F2), and third formant (F3) frequencies were kept the same for all five sound tokens. The values of F0, F2 and F3 were 100 Hz, 1090 Hz, and 350 Hz, respectively. The five-stepped vowel continuum (vw1-vw5) was constructed in a way so that each sound token of 100 ms

would differ minimally acoustically, still be perceived categorically [36-37]. We were aware that hearing loss due to aging may alter cortical auditory evoked potentials [25], and may influence the response; however, audiometric testing showed that hearing thresholds did not differ between groups at octave frequencies between 250 and 4000 Hz [33], which is well beyond the bandwidth of the stimuli.

3.2.3 Data Collection

Data collection technique and response evaluation were homogeneous to the studies reported in [25-26, 32-33, 36]. During EEG recording, participants went through 200 trials for each sound token. Each participant took part in the data collection twice. The experiment was conducted in an electroacoustically shielded chamber (Industrial Acoustics, Inc.). Subjects experienced the stimuli through earphones (ER-3A, Etymotic Research) in both ears at an intensity of 83 dB SPL. To eliminate electromagnetic stimulus artifact from corrupting neurophysiological responses, extended acoustic tubing (50 cm) was used [27,33, 36]. Sound token came to subjects randomly, and they were requested to rapidly categorize them with a binary response (“u” or “a”) by pressing specific buttons on the keyboard. In this study, their response does not matter as we were only interested to see how the ERP changes with different stimulus. However, we wanted them to focus while the stimulus was applied. This was done to make sure that the resulting ERP is produced only due to the applied stimulus. Each sound token was 100 ms duration with 10 ms of rise and fall time to reduce the spectral splatter [25]. After the participants’ response, an inter-stimulus interval (ISI) followed randomly between 400 and 600 ms (20-ms steps, rectangular distribution) to avoid subjects anticipating subsequent stimuli [38]. SynAmps RT EEG amplifiers (Compumedics Neuroscan,

Charlotte, NC, USA) were used to capture EEG data. EEGs was recorded differentially between an electrode placed on the high forehead at the hairline referenced to linked mastoids. To record auditory evoked potentials from cortical origin, this montage (Fpz-A1/A2) is considered optimal [36, 39-40]. Throughout the duration of the experiment, contact impedances were maintained below 3 kΩ, and the EEG signal was captured at 20 kHz sampling rate and then filtered by a band pass filter having passband within 0.05 Hz – 3500 Hz.

3.2.4 Event Related Potential Processing

EEG data were processed using ERPLAB, an open source toolbox, which runs in the MATLAB environment [41]. An interval of 700 ms (100 ms pre-stimulus and 600 ms post-stimulus) constituted an EEG epoch as shown in Fig. 1 [32]. The left part of Fig. 1 represents five grand average ERP from a normal subject due to five auditory stimuli. The right part of Fig. 1 contains comparison between the ERPs of Normal and MCI group. The pre-stimulus region was used for baseline correction, where the subtraction method [42] was used. Trials exceeded ± 50 μV were excluded from the analysis as they were probably contaminated by different artifacts such as eye-blinks, eye-movements etc as mentioned in [30-31]. For each auditory stimulus, artifact free epochs were used to calculate the grand average ERP. Finally, the grand average ERP is bandpass filtered from within (0-30) Hz because of a priori knowledge of the ERP bandwidths and the stimuli [26, 32, 33 ,36, 40, 43]. It is to be noted that data from the different subjects do not go into the ERP calculation, rather each subject has its own ERP, which is calculated from the 200 trials of two sessions.

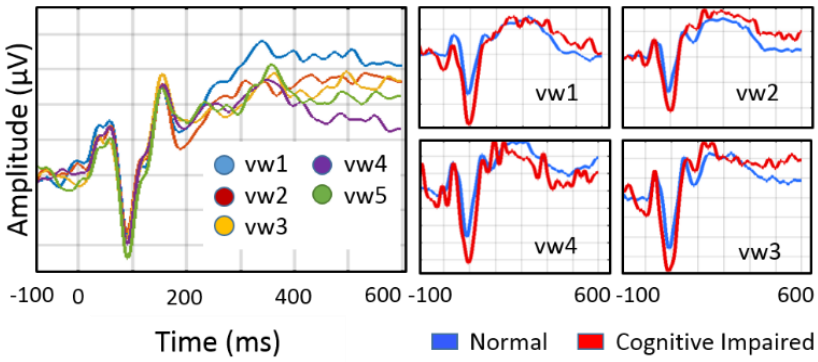


Fig. 9: Visualization of the ERP at different auditory stimuli. Left: individual response for the auditory stimulus vw1, vw2, vw3, vw4, and, vw5 of a normal subject; right: comparison between the ERPs of two subjects that belong to the Normal and the MCI group. ERP responses in the case of four auditory stimulus vw1, vw2, vw3, vw4 are visualized

3.2.5 Features extraction and ranking

We extracted total 590 candidate features from the ERP prominent points and the time and spectral domain characteristics, and used top 25 features in the classification models, which were ranked by the random forest algorithm. In the cortical auditory evoked responses, prominent ERP points seem to have discriminatory power between normal and MCI stage [18] in the older adults. For that reason, we included the ERP prominent points in the candidate feature vector (CFV). The ERP prominent points such as Pa, P1, N1, and P2 were defined as the peak points between the intervals [25 ms - 35 ms], [60 ms – 80 ms], [90 ms – 110 ms], and [150 ms - 250 ms], respectively [32] as shown in Fig. 9. The peak amplitudes and their respective latencies of these prominent points, and the mean amplitudes of the intervals containing the prominent points were also included in the CFV because of their known importance in separating groups in the experiments involving evoked responses [44]. Relative powers in the EEG bands (i.e. delta/theta/alpha/beta) were also useful in classifying normal, mild cognitive impaired,

and Alzheimer's' disease group [45]; thereby, included in the CFV. Total 16 features were calculated from the ERP prominent points from each stimulus, which resulted in total 80 feature points in the CFV.

As the ERP changes rapidly with time within the window over which the stimulus is applied, it was important to track the variation of the time-domain characteristics in high-resolution. In order to accomplish that, we applied a 25ms window, with a 50% overlap through the entire ERP signal as depicted in Fig. 10. The sliding window allowed us to observe the variation of the time domain properties, which has shown significance in classifying MCI and normal subject in earlier studies [33]. Total 107 time-domain characteristics such as signal statistics, correlation properties, entropies, etc. from each window were calculated using the opensource software package "HCTSAtool" [46-47] written in Matlab. For the details of the extracted time domain characteristics, please see Ref. [46-47]. To calculate the variation of the time-domain characteristics over time, we computed the slope and the coefficient of variation (CV), which constituted two feature points at each time-stamp for each stimulus. The feature points that had intra-class similarity and inter-class variability (judged by visual inspection) were included in the CFV. For example, Fig. 10 (a &b) show the slope and CV of one of the time domain characteristics (computed by the HCTSAtool known as "proportion of data within two-standard deviation of mean [46-47]") for the normal and the MCI group, respectively. Here we observed that this feature point was different for two classes and the shaded regions depict that there was intra-class similarity, but this feature was different between the classes (i.e. interclass variability), thereby selected as a feature point in the CFV. Similarly, the slope and CV shown in Fig. 10 (c&d) at a different timestamp also have

intra-class similarity and inter-class variability, which fulfill their requirement to be in the CFV. Among all the time-domain feature points, 510 such feature points met the condition as mentioned above and were included in the CFV.

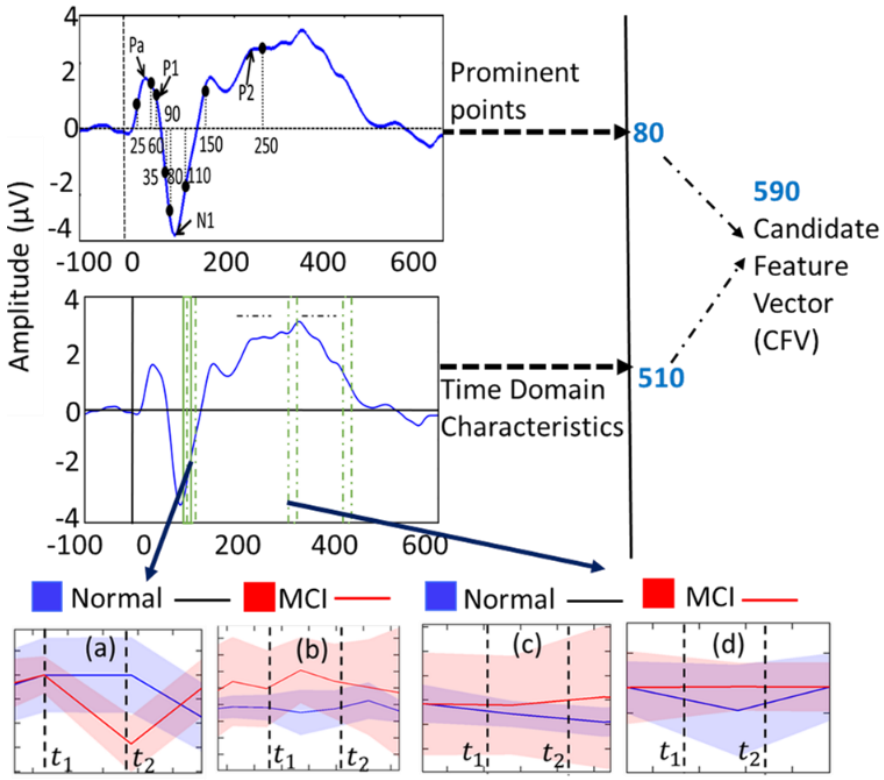


Fig. 10: Schematic of the feature extraction process: (a) and (c) are slopes and (b) and (d) are covariances for two different time windows from timestamp t1 to t2

All 590 features of the CFV (i.e. 80 from the prominent points and 510 from the time-domain characteristics) were ranked by the random forest algorithm [48], and top 25 features were used in the classification models. The random forest algorithm was implemented using the "randomForest" package of R. In the random forest algorithm, total 590 trees were constructed and at each split, 50 randomly sampled features were considered. As the feature ranking criteria, we set the "mean decrease in accuracy" to be greater than 0.002 to include only the most significant features in the final feature vector.

The fewer number of features kept the computation cheap and also reduces the probability of overfitting the data.

TABLE 3. MODELS SELECTED FOR OBSERVING AGGREGATED SOUND FEATURES' PERFORMANCE

Model Name	**Combination**	**Type**	
SVMDn C_m	SVM (degree = n, cost = m)	Polynomial kernel	degree = {2, 3, 4}, cost = {0.01, 0.1, 1, 10, 100}
SVMSigma$_n$C$_m$	SVM (sigma = n, cost = m)	Radial basis kernel	sigma = {10-5, 10-4, 10-3, 10-2, 10-1, 1, 10}, cost = {0.01, 0.1, 1, 10, 100}
LRl1C_m	LR (regularization = 1, cost = m)	l1 regularization	cost = {0.01, 0.1, 1, 10, 100}
LRl2C_m	LR (regularization = 2, cost = m)	l2 regularization	cost = {0.01, 0.1, 1, 10, 100}

3.2.6 Classification

In this study, we used support vector machine (SVM), and logistic regression (LR) as they have already been used in the study related with EEG/ERP [48] data, and at the same time inference from these algorithms were computationally efficient. In the SVM grid search, we implemented polynomial and radial basis kernel, and varied the cost, C = $\{10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2}\}$.

The degree and sigma used in the case of polynomial and RBF kernel, respectively are reported in TABLE I. Similar approach was followed while LR grid search were implemented. For the detail grid search parameters, please see TABLE I.

It was necessary to find a model that balances between bias and variance to prevent overfitting and ensure generalization. Preventing overfitting is especially challenging if the sample size is small. To overcome this challenge, we followed the approach of [22] that means we used leave-one-out cross validation set fixed for all classifiers (SVM, LR). In leave-one-out cross validation, the model is trained with N-1 samples and tested by the Nth sample. This process is repeated for all N samples. The experimental setup, EEG pre-

processing, feature exraction, and classification method are summarized through a flowchart in Fig. 11 to help the readers.

3.3 Result

To evaluate the classification models, we observed the key performance attributes such as leave-one-out cross-validation accuracy, sensitivity of the positive class (i.e. MCI group), and support vector ratio (SVR) (when applicable) in our SVM/LR grid search. In Fig. 12, we demonstrate the variation of the performance metrics in the case of the SVM for both the polynomial ($d = 2, 3,$ and 4), and the RBF ($\sigma = 10$) kernels. Top row of Fig. 12 shows the variation of the leave-one-out cross-validation accuracy with respect to the cost parameters, $C = \{10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2}\}$. The results show that with the increase of C, both the leave-one-out cross-validation accuracy initially increases and reaches at the maximum for the optimal values of C. In Fig. 12(a), Fig. 12(b) and Fig. 12(c), the cross-validation accuracy reaches at the maximum point and did not change later with the increment in C. The optimal C also provides the highest sensitivity and the lowest SVR as shown in the bottom row of Fig. 12(exception for Fig. 12(e), the sensitivity increased after the optimal C). Lower support vector ratio ensures model's stability with respect to the overfitting.

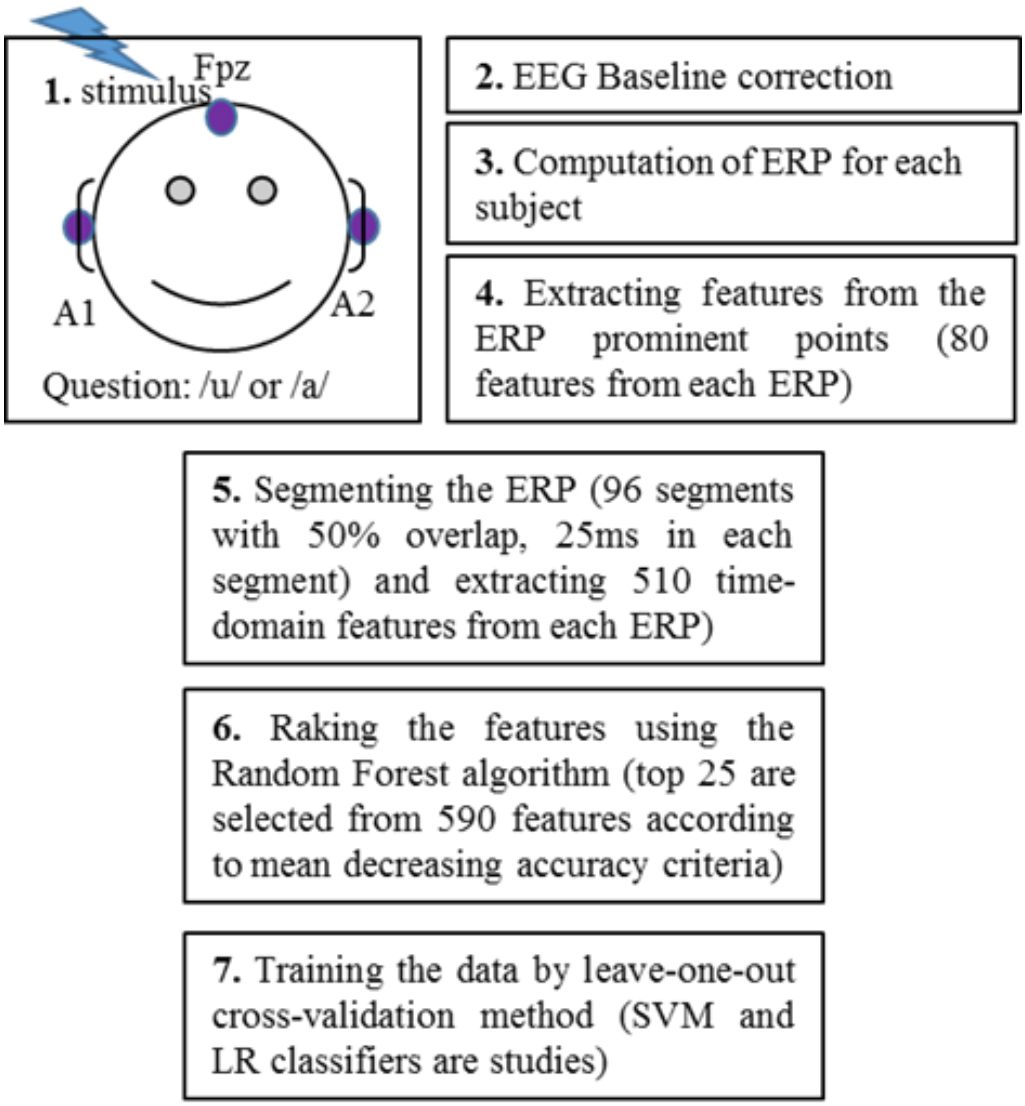


Fig. 11 Flow diagram of the study

The variation of the sensitivity, and, leave-one-out cross-validation accuracy with respect to the cost parameter, $C = \{10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2}\}$ in the case of LR for both the L1 and L2 regularization is shown in Fig. 13. Likewise, SVM, the variation of the leave-one-out cross-validation accuracy, and sensitivity follow the similar trend with *C*, and becomes maximum for the optimal value of *C*.

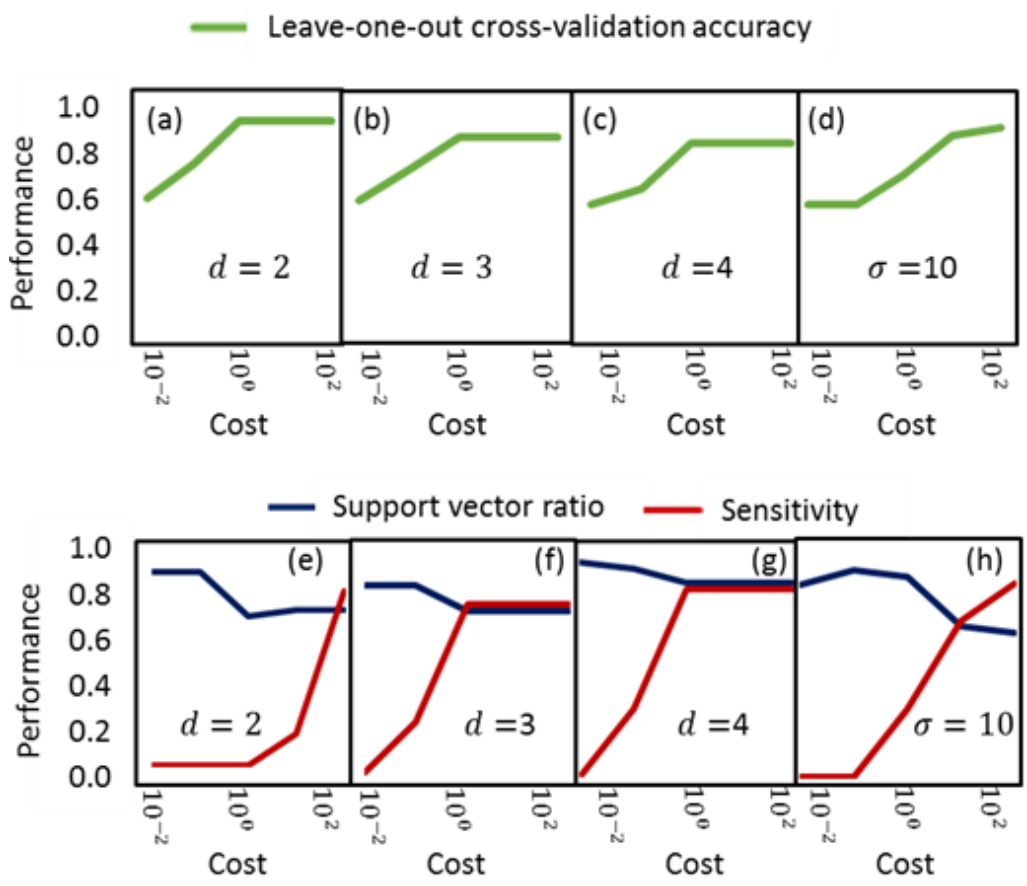


Fig. 12: Performance evaluation of SVM models: Leave-One-Out Cross Validation accuracy of Polynomial kernel (a) d=2, (b) d=3, (c) d=4, Leave-One-Out Cross Validation accuracy of RBF kernel (d) σ=10, Support vector ratio and Sensitivity of Polynomial kernel (e) d=2, (f) d=3, (g) d=4, Support vector ratio and Sensitivity of RBF kernel (h) σ=10

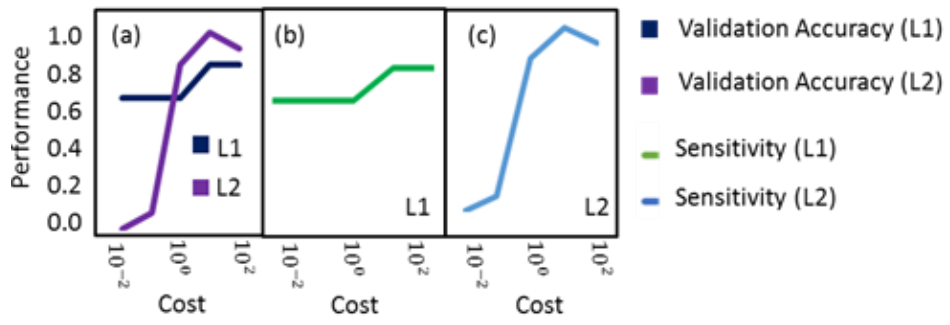


Fig. 13: Performance evaluation of LR models: (a) Sensitivity of L1 and L2 regularization, Leave-One-Out Cross-Validation accuracy of (b) L1 regularization, (c) L2 regularization

We observed that $SVMSigma_{10}C_{100}$ works best among all the models considered here.

The performance attributes for the best classification models for different cases are

summarized in TABLE II. These observations suggest that the selected feature vector is robust to multiple classification models in classifying between the normal and the MCI group.

TABLE 4. PERFORMANCE OF SVMSIGMA$_{10}$C$_{10}$ MODEL SELECTED FOR OBSERVING AGGREGATED SOUND FEATURES' PERFORMANCE

Sen.	**Spec.**	**Prec.**	**F**	**ROC**
0.85	0.95	0.92	0.88	0.91

In order to investigate the impact of different sounds used in the experiment, we used features collected from each sound in our classification models. The results obtained with the best models are summarized in Fig. 14. The sounds were ranked according to the leave-one-out cross validation accuracy, sensitivity for MCI, and F score in TABLE III. We observed that the clear sounds (vw1: */u/*, vw4, and vw5: */a/*) are perceived differently by the normal and impaired individuals and thereby are a better candidate to be used for future study related to MCI detection.

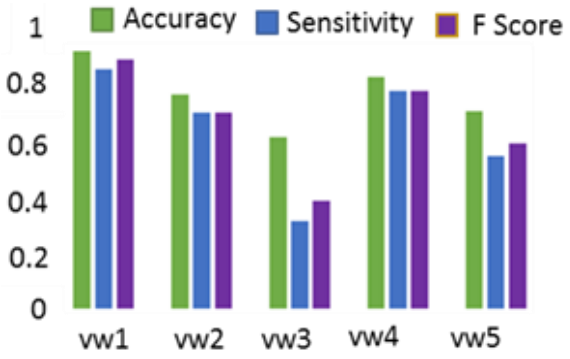


TABLE III. STIMULUS RANK

Stimulus	**Rank**
vw1	1
vw2	4
vw3	5
vw4	2
vw5	3

Fig. 14: Comparison among best performance of each stimulus

3.4 Discussion

We obtained the best performance from the classifier SVM (leave-one-out cross-validation accuracy 87.9% with 84.8 % sensitivity and 85% F score) and demonstrated the robustness of our approach using leave-one-out cross-validation. Additionally, we explored the effects of an ambiguous/non-ambiguous auditory stimulus on the MCI detection, and observed that ambiguous stimulus has less performance than non-ambiguous stimulus in the classification. Pekkonen et al, (1994) [49] reported that dementia patients have faster decay in auditory sensory memory than the age-matched controls, which gets reflected in the mismatch negativity (MMN) (a component measured from ERP). In our previous study (Bidelman et al, (2017) [33]), we observed that, prefrontal dysfunction via efferent connections or abnormalities within the ascending auditory pathways may be related to MCI. Therefore, we could assume that it is possible to have differential properties in the EPR responses between the MCI and HCs due to auditory stimulus. Our current study also supports this hypothesis.

We tabularized our result with the existing work in the literature in TABLE IV. As we discussed earlier, most of the authors used costly neurological data to classify between the normal and the MCI group. Zhang et al, (2011) [11] solved two binary classification problem AD vs. HC, and MCI vs. HC. They extracted features from sMRI, FDG-PET, and CSF data and used them in a linear svm classifier. The achieved 76.4% accuracy with 81.8% sensitivity and 66% specificity. Sui et al, (2014) [12] used the fMRI and DTI data to detect MCI vs. HC. They achieved 96.3% accuracy with 100% sensitivity and 94.1% specificity. Suk et al, (2015) [10] used a stacked auto- encoder

(SAE) with a deep learning-based latent feature representation to solve the four binary classification problems:

AD vs. health normal control (HC), MCI vs. HC, AD vs. MCI, and MCI converter (MCI-C) vs. MCI non-converter (MCI-NC). They achieved 90.7% accuracy, 95% sensitivity, and 85% specificity in classifying MCI vs. HC. They used MRI and PET data in the MCI detection. Ruzzoli et al (2016) [19] used features from multichannel EEG system to classify MCI vs. HC. The authors used LR and achieved 76.9% sensitivity and 73.3% specificity. Kashefpoor et al, (2016) [16] used features from multi-channel EEG system to detect MCI. They used neurofuzzy system and k-nearest classifier and achieved 88.9% accuracy, 100% sensitivity, and 83.3% specificity. Ahmed et al, (2017) [5] used sMRI and DTI data for solving three binary classification problem: AD vs NC, MCI vs NC, and AD vs MCI. They used multiple kernel learning as the classifier and achieved 79.4% accuracy with 71.6% sensitivity and 84.7% specificity. Toth et al, (2018) [22] used spontaneous speech features to classify MCI vs. NC and they got 71.4% cross-validation accuracy with 79.2% sensitivity, and 61.1% specificity. In our work, we used ERP features from single-channel EEG data and we achieved our best performance with SVM. Our performance achieved 87.9% accuracy, 84.8% sensitivity, and 95% specificity in the MCI detection.

TABLE 5. LITERATURE SUMMARY

Paper	Source of Data	Focus	Classification Method	Accuracies (%)	Sensitivity	Specificity	Stimuli
Zhang *et al* [11].	sMRI, FDG-PET, CSF	Classification AD vs HC, MCI vs HC	Linear SVM	93.2, 76.4	93.0, 81.8	93.3, 66.0	N/A
Sui *et al*.[12]	fMRI, DTI	Classification MCI vs. HC	Multi kernel SVM	96.3	100	94.1	N/A
Suk *et al*.[10]	MRI, PET	Classification AD vs. HC, MCI vs. HC, AD vs. MCI, MCI-C vs. MCI-NC	Deep learning	98.8, 90.7, 83.7, 83.3	N/A	N/A	N/A
Ruzzoli *et al* [19]	Multi-ch. EEG	Classification MCI vs. HC	LR	N/A	76.9	73.3	Auditory
Kashefpoor *et al* [16]	Multi-ch. EEG	Classification NC vs MCI	Neurofuzzy system and KNN	88.9	100	83.3	N/A
Ahmed *et al*.[5]	sMRI, DTI	Classification AD vs. NC, MCI vs. NC, AD vs. MCI	Multiple kernel learning	90.2, 79.4, 76.6	82.9, 71.6, 65.5	94.6, 84.7, 81.3	N/A
Toth *et al.* [22]	Speech	Classification NC vs. MCI	Naïve Bayes, Linear SVM, Random Forest	71.4	79.2	61.1	Visual
This Study	Single ch. EEG	Classification NC vs MCI	SVM, LR	87.9	84.8	95.0	Auditory

Our results are comparable to those observed with multi-channel EEG, and fMRI-based techniques in terms of classification accuracy, sensitivity, and specificity. This suggests that our single- channel based method may provide an alternative way of MCI detection, which is easy-to-use, and cost-effective.

The primary goal of this work is to investigate the use of the single-channel EEG data in detecting the early cognitive impairment to support the wearable technology. As we mentioned earlier in the discussion section, multi-channel EEG and other physiological data have proven to be a reliable source for the MCI detection; however, the idea of using the single-channel EEG data in determining complex neurological phenomena is relatively new. We believe that our work will motivate further study in this area.

In MRI technique, a patient needs to lie in the MRI scanner (a very large, strong magnet) and a radio wave is used to send signals to the part of the body of interest and receive them back. A computer attached to the scanner converts the returning signal into images. In PET technique, a patient swallows, inhales, or gets injected by radioactive tracer and then lies under a big PET scanner for generating PET image for diagnosis. fMRI uses the same basic principles as MRI. However, MRI scans anatomical structure whereas fMRI scans metabolic function. DTI, and sMRI also have similarities with MRI technique. High-density multichannel EEG setup requires patient to sit in a quiet environment for a certain time determined by the study for data collection. All the techniques discussed so far here can only be employed in a hospital environment but not in wearable or portable devices. Conversely, the technique discussed in this paper can be implemented in a home environment.

3.5 Conclusion

In this study, we targeted to find a solution in the MCI detection using minimalistic, non-invasive, and low-cost approach –via ERP responses from the scalp EEG data. We selected existing algorithms such as SVM, LR and extracted features from time and frequency domain responses during speech processing responses reflected in the single-channel EEG data obtained from Fpz location. We observed that the top 25 ranked features ranked by the random forest method performed well with most of the classification models. Based on the best performances from SVM model, we can predict MCI with 87.9% leave-one-out cross-validation accuracy, 84.8% sensitivity, and 95% specificity. In future, this study can be expanded to real-time implementation of the system with the hardware-software implementation.

3.6 References:

[1] R. C. Petersen, G. E. Smith, S. C. Waring, R. J. Ivnik, E. G. Tangalos, and E. Kokmen, "Mild cognitive impairment - Clinical characterization and outcome," *Arch. Neurol.*, vol. 56, no. 3, pp. 303–308, 1999.

[2] A. Tsolaki, V. Kosmidou, I. Y. Kompatsiaris, and C. Papadaniil, "Brain source localization of MMN and P300 ERPs in Mild Cognitive Impairment and Alzheimer ' s Disease : A High Density EEG approach," *Neurobiol. Aging*, vol. 55, pp. 190–201, 2017.

[3] H. V. Vinters, "Emerging Concepts in Alzheimer's Disease," *Annu. Rev. Pathol. Mech. Dis.*, 2015.

[4] H. Ni, J. Qin, L. Zhou, Z. Zhao, J. Wang, and F. Hou, "Network analysis in detection of early-stage mild cognitive impairment," *Phys. A Stat. Mech. its Appl.*, vol. 478, pp. 113–119, 2017.

[5] O. Ben Ahmed, J. Benois-Pineau, M. Allard, G. Catheline, and C. Ben Amar, "Recognition of Alzheimer's disease and Mild Cognitive Impairment with multimodal image-derived biomarkers and Multiple Kernel Learning," *Neurocomputing*, vol. 220, pp. 98–110, 2017.

[6] R. B. Knowles *et al.*, "Plaque-induced neurite abnormalities: Implications for disruption of neural networks in Alzheimer's disease," *Proc. Natl. Acad. Sci.*, 1999.

[7] L. R. Trambaiolli, N. Spolaôr, A. C. Lorena, R. Anghinah, and J. R. Sato, "Feature selection before EEG classification supports the diagnosis of Alzheimer's disease," *Clin. Neurophysiol.*, vol. 128, pp. 2058–2067, 2017.

[8] C.-L. Tsai, M.-C. Pai, J. Ukropec, and B. Ukropcová, “The Role of Physical Fitness in the Neurocognitive Performance of Task Switching in Older Persons with Mild Cognitive Impairment,” *J. Alzheimer’s Dis.*, 2016.

[9] D. De Venuto, V. F. Annese, and G. Mezzina, “Remote Neuro-Cognitive Impairment Sensing Based on P300 Spatio-Temporal Monitoring,” *IEEE Sens. J.*, vol. 16, no. 23, pp. 8348–8356, 2016.

[10] H. Il Suk, S. W. Lee, and D. Shen, “Latent feature representation with stacked auto-encoder for AD/MCI diagnosis,” *Brain Struct. Funct.*, 2015.

[11] D. Zhang, Y. Wang, L. Zhou, H. Yuan, D. Shen, and I. Alzheimer’s Disease Neuroimaging, “Multimodal classification of Alzheimer’s disease and mild cognitive impairment,” *Neuroimage*, 2011.

[12] J. Sui, R. Huster, Q. Yu, J. M. Segall, and V. D. Calhoun, “Function-structure associations of the brain: Evidence from multimodal connectivity and covariance studies,” *NeuroImage*. 2014.

[13] M. E. López *et al.*, “MEG beamformer-based reconstructions of functional networks in mild cognitive impairment,” *Front. Aging Neurosci.*, 2017.

[14] A. K. Kaiser, M. Doppelmayr, and B. Iglseder, “EEG beta 2 power as surrogate marker for memory impairment: a pilot study,” *Int. Psychogeriatrics*, 2017.

[15] R. A. Lopez Zunini *et al.*, “Event-related potentials elicited during working memory are altered in mild cognitive impairment.,” *Int. J. Psychophysiol.*, 2016.

[16] M. Kashefpoor, H. Rabbani, and M. Barekatain, “Automatic Diagnosis of Mild Cognitive Impairment Using Electroencephalogram Spectral Features.,” *J. Med. Signals Sens.*, 2016.

[17] L. T. Timothy, B. M. Krishna, and U. Nair, “Classification of mild cognitive impairment EEG using combined recurrence and cross recurrence quantification analysis,” *Int. J. Psychophysiol.*, vol. 120, no. June, pp. 86–95, 2017.

[18] J. J. Lister, A. L. Harrison Bush, R. Andel, C. Matthews, D. Morgan, and J. D. Edwards, “Cortical auditory evoked responses of older adults with and without probable mild cognitive impairment,” *Clin. Neurophysiol.*, 2016.

[19] M. Ruzzoli, C. Pirulli, V. Mazza, C. Miniussi, and D. Brignani, “The mismatch negativity as an index of cognitive decline for the early detection of Alzheimer’s disease.,” *Sci. Rep.*, 2016.

[20] C. D. Papadaniil, V. E. Kosmidou, A. Tsolaki, M. Tsolaki, I. (Yiannis) Kompatsiaris, and L. J. Hadjileontiadis, “Cognitive MMN and P300 in mild cognitive impairment and Alzheimer’s disease: A high density EEG-3D vector field tomography approach,” *Brain Res.*, 2016.

[21] A. Konig *et al.*, “Use of Speech Analyses within a Mobile Application for the Assessment of Cognitive Impairment in Elderly People,” *Curr. Alzheimer Res.*, 2018.

[22] L. Toth *et al.*, “A Speech Recognition-based Solution for the Automatic Detection of Mild Cognitive Impairment from Spontaneous Speech,” *Curr. Alzheimer Res.*, 2018.

[23] G. Gosztolya *et al.*, “Detecting mild cognitive impairment from spontaneous speech by correlation-based phonetic feature selection,” in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2016.

[24] J. Weiner, C. Herff, and T. Schultz, “Speech-based detection of Alzheimer’s disease in conversational German,” in *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, 2016.

[25] G. M. Bidelman, J. W. Villafuerte, S. Moreno, and C. Alain, “Age-related changes in the subcortical-cortical encoding and categorical perception of speech,” *Neurobiol. Aging*, 2014.

[26] G. M. Bidelman and C. Alain, “Musical Training Orchestrates Coordinated Neuroplasticity in Auditory Brainstem and Cortex to Counteract Age-Related Declines in Categorical Vowel Perception,” *J. Neurosci.*, 2015.

[27] S. J. Aiken and T. W. Picton, “Envelope and spectral frequency-following responses to vowel sounds,” *Hear. Res.*, 2008.

[28] R. C. Oldfield, “The assessment and analysis of handedness: The Edinburgh inventory,” *Neuropsychologia*, 1971.

[29] R. Mahajan and B. Morshed, “Unsupervised Eye Blink Artifact Denoising of EEG Data with Modified Multiscale Sample Entropy, Kurtosis and Wavelet-ICA,” 2014.

[30] S. Khatun, R. Mahajan, and B. I. Morshed, “Comparative Study of Wavelet-Based Unsupervised Ocular Artifact Removal Techniques for Single-Channel EEG Data,” *IEEE J. Transl. Eng. Heal. Med.*, vol. 4, 2016.

[31] S. Khatun, R. Mahajan, and B. I. Morshed, “Comparative analysis of wavelet based approaches for reliable removal of ocular artifacts from single channel EEG,” in *IEEE International Conference on Electro Information Technology*, 2015, vol. 2015–June.

[32] S. Khatun, B. I. Morshed, and G. M. Bidelman, “Single channel EEG time-frequency features to detect Mild Cognitive Impairment,” in *2017 IEEE International Symposium on Medical Measurements and Applications, MeMeA 2017 - Proceedings*, 2017.

[33] G. M. Bidelman, J. E. Lowther, S. H. Tak, and C. Alain, “Mild Cognitive Impairment Is Characterized by Deficient Brainstem and Cortical Representations of Speech,” *J. Neurosci.*, 2017.

[34] Z. S. Nasreddine, , “The Montreal Cognitive Assessment, MoCA: A Brief Screening Tool for Mild Cognitive Impairment,” *J. Am. Geriatr. Soc.*, 2005.

[35] D.M.C. Doerflinger , “Mental status assessment in older adults: Montreal Cognitive Assessment: MoCA Version 7.1 (original version)”, *The Clinical Neuropsychologist*, *25*(1), pp.119-126, 2012.

[36] G. M. Bidelman, S. Moreno, and C. Alain, “Tracing the emergence of categorical speech perception in the human auditory system,” *Neuroimage*, 2013.

[37] D. B. Pisoni, “Auditory and phonetic memory codes in the discrimination of consonants and vowels,” *Percept. Psychophys.*, 1973.

[38] S. Luck, *An introduction to the event related potential technique*. 2005.

[39] A. Krishnan, J. T. Gandour, and G. M. Bidelman, “The effects of tone language experience on pitch processing in the brainstem,” *J. Neurolinguistics*, 2010.

[40] G. Musacchia, D. Strait, and N. Kraus, “Relationships between behavior, brainstem and cortical encoding of seen and heard speech in musicians and non-musicians,” *Hear. Res.*, 2008.

[41] J. Lopez-Calderon and S. J. Luck, “ERPLAB: an open-source toolbox for the analysis of event-related potentials,” *Front. Hum. Neurosci.*, 2014.

[42] L. Hu, P. Xiao, Z. G. Zhang, A. Mouraux, and G. D. Iannetti, “Single-trial time-frequency analysis of electrocortical signals: Baseline correction and beyond,” *Neuroimage*, 2014.

[43] G. M. Bidelman, “Towards an optimal paradigm for simultaneously recording cortical and brainstem auditory evoked potentials,” *J. Neurosci. Methods*, 2015.

[44] B. Blankertz, S. Lemm, M. Treder, S. Haufe, and K.-R. Mueller, “Single-trial analysis and classification of ERP components - A tutorial,” *Neuroimage*, vol. 56, no. 2, pp. 814–825, 2011.

[45] K. van der Hiele *et al.*, “EEG and MRI correlates of mild cognitive impairment and Alzheimer’s disease,” *Neurobiol. Aging*, 2007.

[46] B. D. Fulcher and N. S. Jones, “Automatic time-series phenotyping using massive feature extraction,” 2016.

[47] B. D. Fulcher, M. A. Little, and N. S. Jones, “Highly comparative time-series analysis: The empirical structure of time series and their methods,” R. Soc. Publ., 2013.

[48] P. Bashivan, M. Yeasin, and G. M. Bidelman, “Single trial prediction of normal and excessive cognitive load through EEG feature fusion,” in *2015 IEEE Signal Processing in Medicine and Biology Symposium - Proceedings*, 2016.

[49] E. Pekkonen, V. Jousmäki, M. Könönen, K. Reinikainen, and J. Partanen, “Auditory sensory memory impairment in Alzheimer’s disease: an event-related potential study.,” *NeuroReport*. 1994.

4. Regression Based Approaches to Monitor the Severity of Mild Cognitive Impairment from Single Channel EEG Data

Abstract—Cognitive health is one of the aspects of a human life which cannot be ignored and proper care of it ensures a healthy, happy, and balanced life. The deviation of the soundness of cognitive health is mild cognitive impairment (MCI) is important to detect early and monitor progression to reduce the risk of complicated diseases such as Dementia, Alzheimer's Disease (AD), and Parkinsons Disease (PD). In this study, we developed single channel Electro-encephalography (EEG) based MCI severity monitoring algorithm by generating Montreal Cognitive Assessment (MoCA) scores from the features extracted from EEG. We performed multi-trial and single-trail analysis for the algorithm development of the MCI severity monitoring. For multi-trial analysis, we extracted 590 features from the prominent ERP points and time domain characteristics of the ERP. We used lasso regression technique to select best feature set. 13 best features were used in the classical regression techniques: Multivariate Regression (MR), Ensemble Regression (ER), Support Vector Regression (SVR), and Ridge Regression (RR). The best results came from ER according to the RMSE (0.08) and residual analysis. In our single-trial analysis, we extracted time-frequency images from each trial, and used it as an input to the constructed convolutional deep neural network (CNN). This deep CNN model gave 0.09 RMSE. We are the first one to report this novel method to generate MCI severity from single channel EEG data for both multi-trial and single-trial data. This method will be implementable in portable setup and provide cognitive healthcare for elderly people with MCI or patients who have self-reporting concerns.

Index Terms—Electroencephalography, Event related potential, Mild cognitive impairment, Montreal Cognitive Assessment, Deep Neural Network, Single trial, Ensemble regression.

4.1 Introduction

Memory disorder in a human being brings difficulties in performing basic (BADL) [walking, eating, dressing, showering, feeding, continence etc.] and instrumental (IADL) [preparing meals, solving everyday situations, managing finances, doing laundry, taking medications etc.] activities of daily living [1]. Mild cognitive impairment (MCI) is considered to be the state between the normal cognition and dementia [2]. There are many reasons that the MCI should be detected quickly. MCI has a 54% chance to get converted in to Alzheimer's disease (AD) or related dementia [3]. MCI deteriorates the ability of the elderly people to perform daily activities and to live an independent life. In some cases, patients suffering from complicated diseases (e.g. Parkinson's Disease) may have develop MCI later [4] and progression of cognitive impairment monitoring is crucial for their caregivers, doctors and families. MCI may be associated with cardiovascular disease, metabolic syndrome, type 2 diabetes, sedentary activity, obesity, excess alcohol, and smoking [5]. According to a group from the Alzheimer's Association, persons who already have been suspected to have cognitive impairment based on clinical observations or who have self-reported concerns, can undergo cognitive impairment assessment. For the cognitive health screening, they have recommended any one of the cognitive screening tests of the lists: a) Montreal Cognitive Assessment (MoCA), b) Mini-Mental State Examination (MMSE), c) St. Louis University Mental Status Exam (SLUMS), d) General Practitioner Assessment of Cognition (GPCOG), e) Mini-Cog, f) Memory

Impairment Screen (MIS), g) AD8, or h) Informant Questionnaire on Cognitive Decline in the Elderly (short-IQCODE) [6]. Scientists compared the performances of widely used cognitive assessment tests and showed so far that when a patient's cognitive impairment goes beyond memory impairment, MoCA should be used [7] and MoCA is more sensitive than widely used MMSE [7][8]. Researchers have predicted cognitive scores based on the physiological data (Electroencephalogram (EEG)) in a motivation that it might be helpful to monitor a person's cognitive health easily and at the time, when a patient is unable to undergo a cognitive assessment. All of the research related with measuring the severity of cognitive impairment were based on features extracted from multi-channel EEG system: EEG power ratio from 16 channel system by Bennys *et al* [9], EEG power spectra and other information from 16 channel system by Kowalski *et al* [10], EEG power of prominent bands from 20 channel EEG system [11], spectral characteristics of EEG from 20 channel system [12], EEG power ratio from 19 channel system [13], grand total EEG score from 16 channel system [14] etc. To our knowledge, we didn't see anyone oher than us reported work on MCI severitiy detection based generating score from single channel EEG system.

In this work, we performed MCI severity generation based on single channel EEG data collected from Fz location. Our analysis divided in to two parts: a) Multi-trial, and b) Single-trial. We followed multi-trial approach based on classical regression techniques. We used features extracted from grand average event related potential (ERP) in this case. In single-trial analysis, we generate MCI severity based on convolutional deep neural network. We used the time-frequency image of each trial as input to the deep neural network. Our key contribution here is to propose model based on multi-trial (best for

offline system) and model based single-trial (best for real-time system) to predict the severity of cognitive impairment and thus helping to assess the cognitive health at ease.

4.2 Method

4.2.1 Participants

Twenty-three older adults were recruited under a study which was designed to observe the aging effect on the auditory system and to evaluate the cognitive performance of the subjects. Electroencephalography (EEG) data was collected in this study under an experiment where twenty-three older adults (age ranges from 52-86 years; mean ± standard deviation: 70.2±7.2 yrs) received auditory stimuli. All the participants had no known history of neurological or psychiatric illness and they all were strongly right handed [15]. During the participants' recruitment process, exclusion criteria was followed based on age, hearing loss, musical training, and, handedness [16]. Written consent under the protocol approved by the Baycrest Centre Ethics Committee (REB #06-31) was collected from each participant before data collection and participants were compensated for their time after data collection. The cognitive health status of the participants was assessed by the well-established cognitive screening tool called Montreal Cognitive Assessment (MoCA) test [17]. In this cognitive screening test, among all the participants, fifteen older participants (8 male, 7 female) were found to have normal cognition (MoCA score ≥26 points; mean± standard deviation: 27.6±1.18; range: 26-30), and eight participants' (4 male, 4 female) were assessed to have MCI (MoCA score <26; mean± standard deviation: 23.0±1.85; range: 20-25). Also, it is important mentioning that in general patients having Alzheimer's disease or more severe dementia generate MoCA score within 11.4 to 21.

4.2.2 Study Design

The study was designed in such a way that the subjects will hear five auditory stimuli. Stimuli was constructed by constructing a perceptual phonetic continuum from /u/ to /a/ by varying parametrically the first formant (F1) frequency between 430 and 730 Hz over five equal steps (For further stimulus details, see [27,28]). The synthetic five-stepped vowel continuum (denoted hereafter by "vw1-5") was built in a way such that each token of 100 ms sound would differ minimally acoustically while hearing, still be perceived categorically [28-29]. As we are aware of the fact that hearing loss due to aging may alter auditory evoked potentials from cortical region [20], and also may affect the response that the participant makes, an audiometric testing was performed. However, the audiometric testing demonstrated that hearing thresholds were not distinguishable between groups (Normal and MCI) at octave frequencies between 250 and 4000 Hz [16], which is well outside of the bandwidth of the stimuli.

4.2.3 Data Collection and Event Related Potential Processing

Data collection technique and response evaluation were homogeneous to the studies reported in [22,26,28,30,31]. During EEG recording, participants went through 200 trials for each sound token. The experiment was conducted in an electroacoustically shielded chamber (Industrial Acoustics, Inc.). Subjects experienced the stimuli through earphones (ER-3A, Etymotic Research) in both ears at an intensity of 83 dB SPL. To eliminate electromagnetic stimulus artifact from corrupting neurophysiological responses, extended acoustic tubing (50 cm) was used [26,28,32]. Sound token came to them in a randomly

ordered way and they were requested to rapidly categorize them with a binary response ("u" or "a"). Each sound token was 100 ms duration with 10 ms of rise and fall time to reduce the spectral splatter [20]. After the participants' response, an inter-stimulus interval (ISI) followed randomly between 400 and 600 ms (20-ms steps, rectangular distribution) to avoid subjects anticipating subsequent stimuli [24]. SynAmps RT EEG amplifiers (Compumedics Neuroscan, Charlotte, NC, USA) were used to capture EEG data. EEGs was recorded differentially between an electrode placed on the high forehead at the hairline referenced to linked mastoids. To record auditory evoked potentials from cortical origin, this montage (Fpz-A1/A2) is considered optimal [28, 34-35]. Throughout the duration of the experiment, contact impedances were maintained below 3 kΩ, and the EEG signal was captured at 20 kHz sampling rate and then filtered by a band pass filter having passband within 0.05 Hz – 3500 Hz.

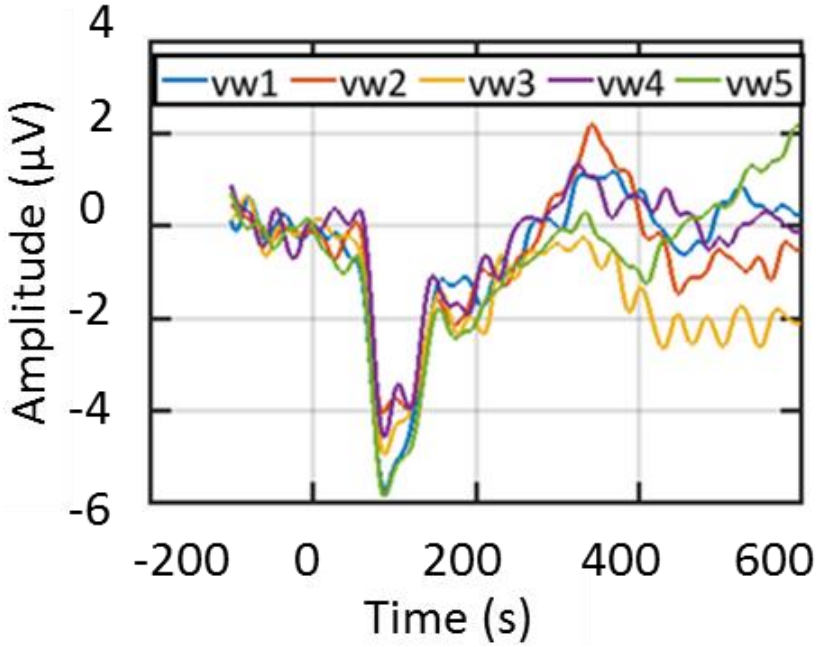


Fig. 15. A multi-trial ERP from subject ID 3610 for five auditory stimuli (vw1, vw2, vw3, vw4, vw5)

EEG data were processed using ERPLAB, an open source toolbox, which runs in the MATLAB environment [26]. An interval of 700 ms (100 ms pre-stimulus and 600 ms post-stimulus) constituted an EEG epoch as shown in Fig. 15 [3]. The pre-stimulus region was used for baseline correction, where the subtraction method [27] was used. Trials exceeded ± 50 μV were excluded from the analysis as they were probably contaminated by different artifacts such as eye-blinks, eye-movements etc as mentioned in [38-39]. For each auditory stimulus, artifact free epochs were used to calculate the grand average ERP (i.e. average of 200 trails). Finally, the grand average ERP is bandpass filtered from within (0-30) Hz because of a priori knowledge of the ERP bandwidths and the stimuli [22, 26,28, 31, 35,40]. A visual representation of the extracted ERP from the multi trials from a subject ID 3610 is presented in Fig. 15.

4.3 Multi trial analysis

4.3.1 Feature Extraction and Ranking

We extracted total 590 candidate features from the ERP prominent points and the time and spectral domain characteristics, and used top 25 features in the classification models, which were ranked by the random forest algorithm. In the cortical auditory evoked responses, prominent ERP points seem to have discriminatory power between normal and MCI stage [31] in the older adults. For that reason, we have included the ERP prominent points in the candidate feature vector (CFV). The ERP prominent points such as Pa, P1, N1, and P2 were defined as the peak between the intervals [25 ms - 35 ms], [60 ms – 80 ms], [90 ms – 110 ms], and [150 ms - 250 ms], respectively [3] as shown in Fig. 16. The peak amplitudes and their respective latencies of these prominent points, and the mean amplitudes of the intervals containing the prominent points are also included in the

CFV because of their known importance in separating groups in the experiments involving evoked responses [32]. Relative powers in the EEG bands (i.e. delta/theta/alpha/beta) are also useful in classifying normal, mild cognitive impaired, and Alzheimer's' disease group [33]; thereby, included in the CFV. Total 16 features were calculated form the ERP prominent points from each stimulus, which resulted in total 80 feature points in the CFV.

As the ERP changes rapidly with time within the window over which the stimulus is applied, it is important to track the variation of the time-domain characteristics in high-resolution. In order to accomplish that, we applied a 25ms window, with a 50% overlap through the entire ERP signal as depicted in Fig. 16. The sliding window allows us to observe the variation of the time domain properties, which has shown significance in classifying MCI and normal subject in earlier studies [16]. Total 107 time-domain characteristics such as signal statistics, correlation properties, entropies, etc. from each window were calculated using the opensource software package "HCTSAtool" [43] written in Matlab. For the details of the extracted time domain characteristics, please see Ref. [44-45]. To calculate the variation of the time-domain characteristics over time, we computed the slope and the coefficient of variation (CV), which constituted two feature points at each time-stamp for each stimulus. The feature points that had intra-class similarity and inter-class variability (judged by visual inspection) were included in the CFV. For example, Fig. 16 (a &b) show the slope and CV of one of the time domain characteristics (computed by the HCTSA tool known as "proportion of data within two-standard deviation of mean [44-45]") for the normal and the MCI group, respectively. Here we observe that this feature point is different for two classes and the shaded regions

depict that there is intra-class similarity, but this feature is different between the classes (i.e. interclass variability), thereby selected as a feature point in the CFV. Similarly, the slope and CV shown in Fig. 16 (c&d) at a different timestamp also have intra-class similarity and inter-class variability, which fulfill their requirement to be in the CFV. Among all the time-domain feature points, 510 such feature points met the condition as mentioned above and were included in the CFV.

All 590 features of the CFV (i.e. 80 from the prominent points and 510 from the time-domain characteristics) were ranked by the lasso regression algorithm, and top 13 features were used in the regression methods. The lasso regression algorithm was implemented using MATLAB. Lasso is a regularization technique to find the good features in a regression problem and thus to reduce the number of features in a large feature set [34]. For a given value of λ, lasso solves the below mathematical

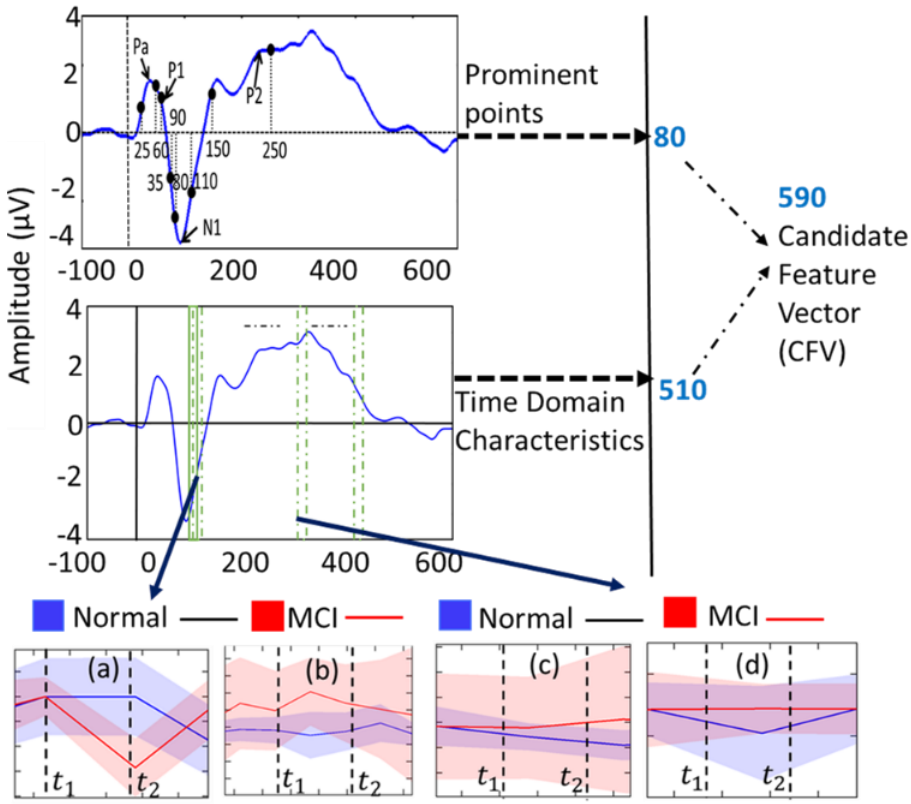


Fig. 16 Schematic of the feature extraction process: (a) and (c) are slopes and (b) and (d) are covariances for two different time windows from timestamp t1 to t2

problem:

$$\min_{\beta_0,\beta}\left(\frac{1}{2N}\sum_{i=1}^{N}(y_i - \beta_0 - x_i^T\beta)^2 + \lambda\sum_{j=1}^{p}|\beta_j|\right) \tag{10}$$

Here, N is the number of observations, y_i is the response at observation i, x_i is data, a vector of p values at observation i, λ is a positive regularization parameter corresponding to one value of Lambda, β_0, β are scalar parameters and p is vector parameters.

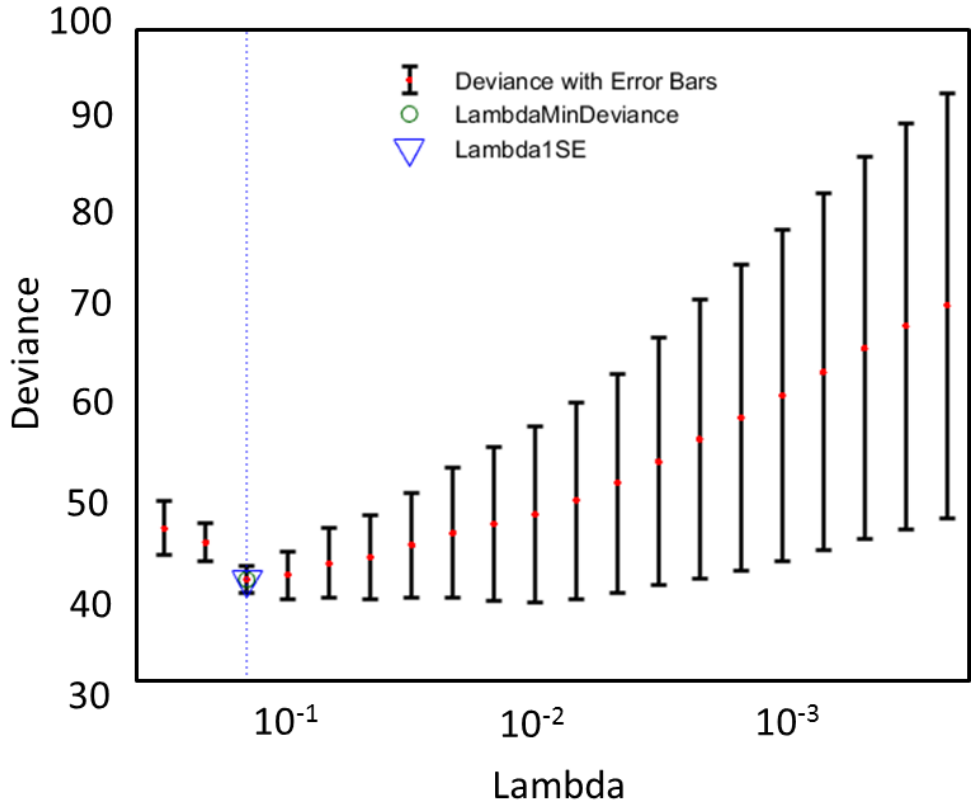


Fig. 17: Lasso regression-based feature selection: five-fold cross-validated mean squared error/deviance vs. lambda

According to the above figure (Fig. 17), when the lambda is close to 10-1, the cross-validated deviance is the lowest and the number of features that has generated the deviance is 13.

4.3.2 Regression

As the MoCA scores are bounded 0-30, so instead of straight linear regression, multivariate regression with logit link function, ensemble regression, support vector regression and ridge regression were tried. We performed five-fold cross-validation to ensure the robustness of the model and the reliability of the results.

4.4 Single trial analysis

4.4.1 Feature Extraction

For the single trial analysis, we consider each trial length from the onset of the stimulus to the response of the subjects. We excluded those responses which are less than 0.2 ms or greater than 1.5 ms [16] or where the range of the magnitude of the signal is greater than 100 µV as they might be contaminated by artifacts. [35]. We then did a time frequency analysis over each individual trial from (0-100) Hz frequency. We collected the generated image of time-frequency in png format and used it for deep neural network analysis. pspectrum.m from MATLAB 2018a was used to extract the time frequency feature. A sample image is given in Fig. 18.

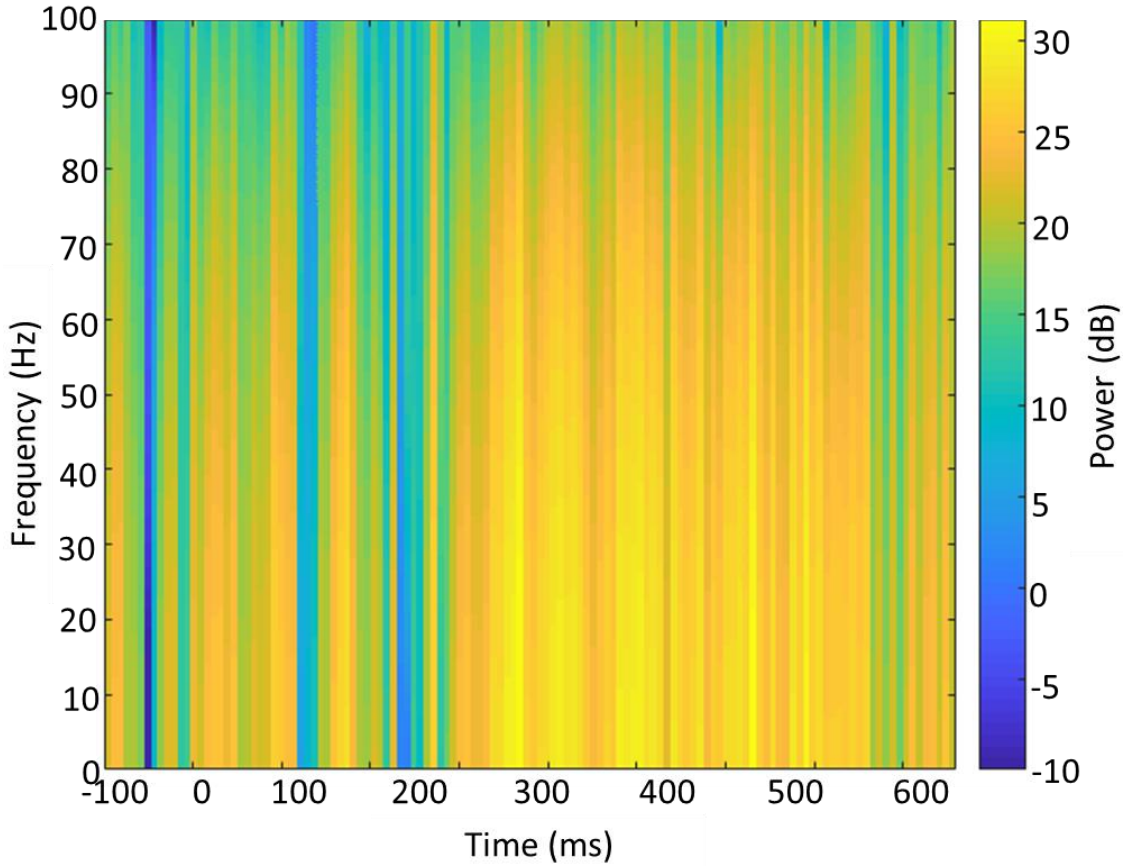


Fig. 18: Time Frequency Image Feature Sample

4.4.2 Deep Neural Regression and Bayesian Optimization

We have total 2848 samples as our input. We constructed our deep neural network with the help of keras package from python. We used sequential model. We constructed three sets of convolutional 2D layer and maxpooling layer with 'relu' activation layer. Then, we used the flatten layer, two dense layers with 'tanh' activation layer in between. To find out the optimal convolutional filter size, dropout rate, and no. of epoch, we used gaussian process-based Bayesian optimization. The whole process of the gaussian process-based Bayesian optimization can be divided into four steps. For the convenience of the explanation, we will explain the third and fourth steps together. The steps are: a) Presample, b) Kernel Comparison, and c) Exploration and Exploitation. The details of the Bayesian optimization stages are depicted in Fig. 19.

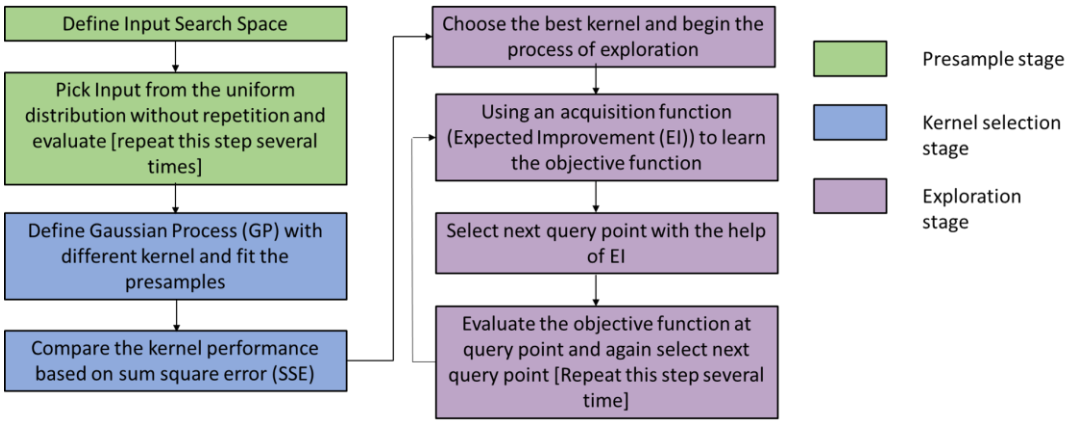


Fig. 19: Bayesian Optimization Stages

In the presample stage, we define a uniform distribution for each of our hyperparameter (convolutional filter size, dropout rate, and no. of epoch). Each time we randomly sampled from the uniform distributions without repetition and we used them in the deep neural network and obtained the cross-validated root mean square error. We repeated the process five times in the presample stage. Then we entered into the Kernel

Comparison stage. We used five kernels: i) Matern, ii) Radial Basis Function, iii) Rational Quadratic, iv) Exponential Sinsquared, and v) Dot Product to compare the performance of sample fitting in to gaussian process. The kernel performances in terms of sum square error (SSE) are given below:

TABLE 6. KERNEL PERFORMANCE OF DATA FITTING IN GAUSSIAN PROCESS

Kernel	SSE
Matern	5.45e-19
Radial Basis Function	5.45e-19
Rational Quadratic	3.59e-19
Exponential Sinsquared	2.48
Dot Product	54.14

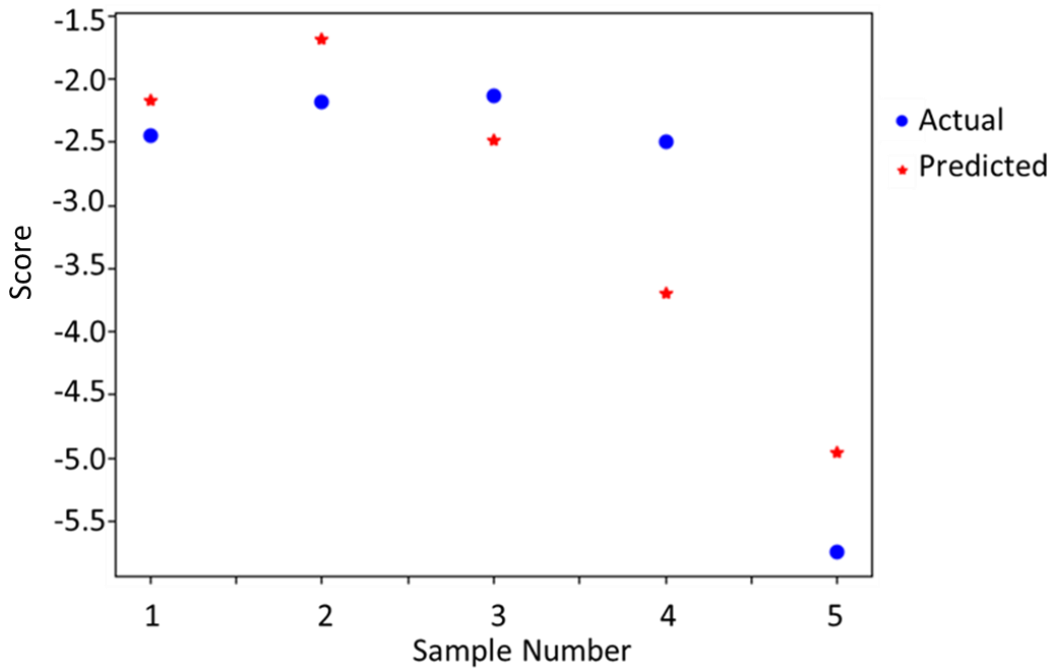


Fig. 20: Presample fit performance to gaussian kernel dot product

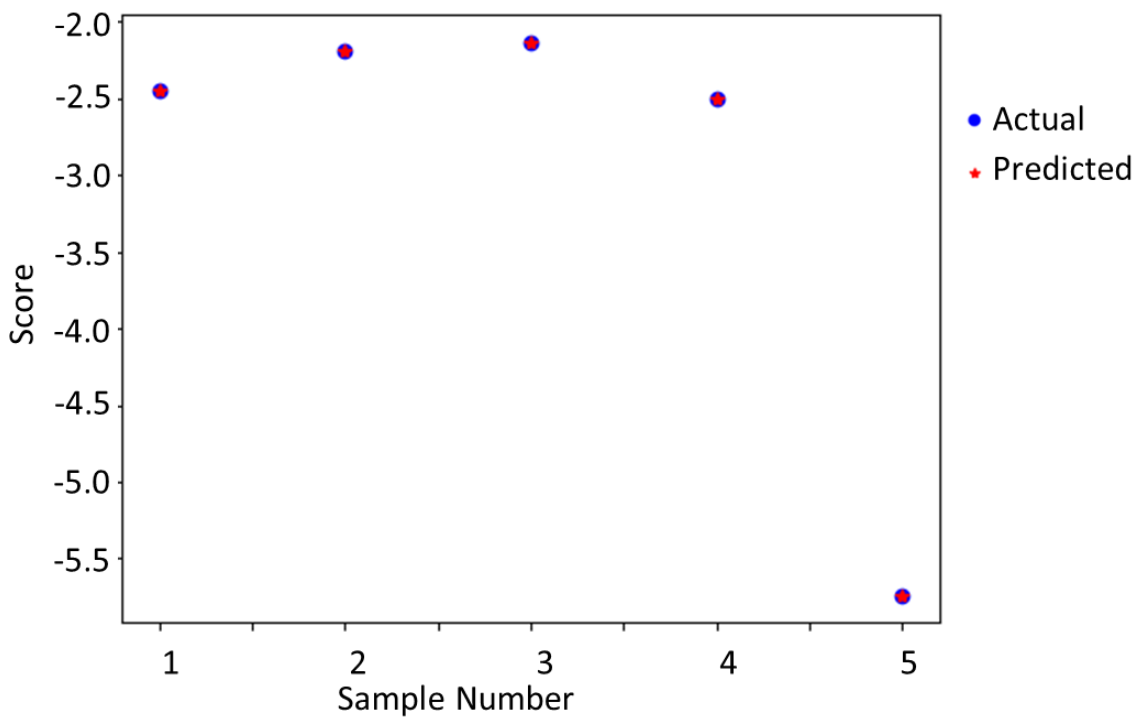


Fig. 21: Presample fit performance to gaussian kernel rational quadratic

According to Fig. 20, the gaussian model defined with kernel dot product did not fit the data well and that is why there are big differences between actual and predicted samples. Whereas Fig. 21 reflected that the gaussian model defined with the rational quadratic kernel fitted the data well. So, we used this kernel for our rest of the Bayesian optimization process. In the exploration stage, we defined the gaussian process with the y kernel. We fit it with presampled data. Then we used the following acquisition function to calculate our next sample.

$$EI(x) = \begin{cases} \left(\mu(x) - f(x^{+})\right)\Phi(Z) + \sigma(x)\phi(Z) & if\ \sigma(x) > 0 \\ 0 & if\ \ \sigma(x) = 0 \end{cases}$$

$$Z = \frac{\mu(x) - f(x^{+})}{\sigma(x)} \tag{11}$$

Where x is the array of possible samples that can be taken in the next steps, $\mu(x)$ and $\sigma(x)$ are the mean and standard deviation of the output values returned by the gaussian

process against x, $f(x^{+})$ is the lowest RMSE obtained so far, $\Phi(Z)$ and $\phi(Z)$ are the PDF and CDF of the standard normal distribution respectively [36]. We took the sample for our deep neural network implementation which got highest EI value according to equation z. We repeated the process fifteen times and we concluded the exploration stage. As we get many similar performing samples in the exploration stage, so we didn't want to exploit around some samples. Finally we selected convolutional filter size, dropout rate , and epoch .The detail parameters of the model is depicted in Fig. 22. In the bayesian optimization phase and in the final phase, after constructing the deep neural network model, we compiled it with 'adadelta' optimizer. We evaluate the model always with five-fold cross-validation.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_16 (Conv2D)           (None, 129, 173, 32)      896
_________________________________________________________________
activation_21 (Activation)   (None, 129, 173, 32)      0
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 64, 86, 32)        0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 62, 84, 32)        9248
_________________________________________________________________
activation_22 (Activation)   (None, 62, 84, 32)        0
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 31, 42, 32)        0
_________________________________________________________________
conv2d_18 (Conv2D)           (None, 29, 40, 64)        18496
_________________________________________________________________
activation_23 (Activation)   (None, 29, 40, 64)        0
_________________________________________________________________
max_pooling2d_18 (MaxPooling (None, 14, 20, 64)        0
_________________________________________________________________
flatten_6 (Flatten)          (None, 17920)             0
_________________________________________________________________
dense_11 (Dense)             (None, 64)                1146944
_________________________________________________________________
activation_24 (Activation)   (None, 64)                0
_________________________________________________________________
dense_12 (Dense)             (None, 1)                 65
=================================================================
Total params: 1,175,649
Trainable params: 1,175,649
Non-trainable params: 0
```

Fig. 22: Convolutional Deep Neural Network for Regression: Layers and Their Parameters

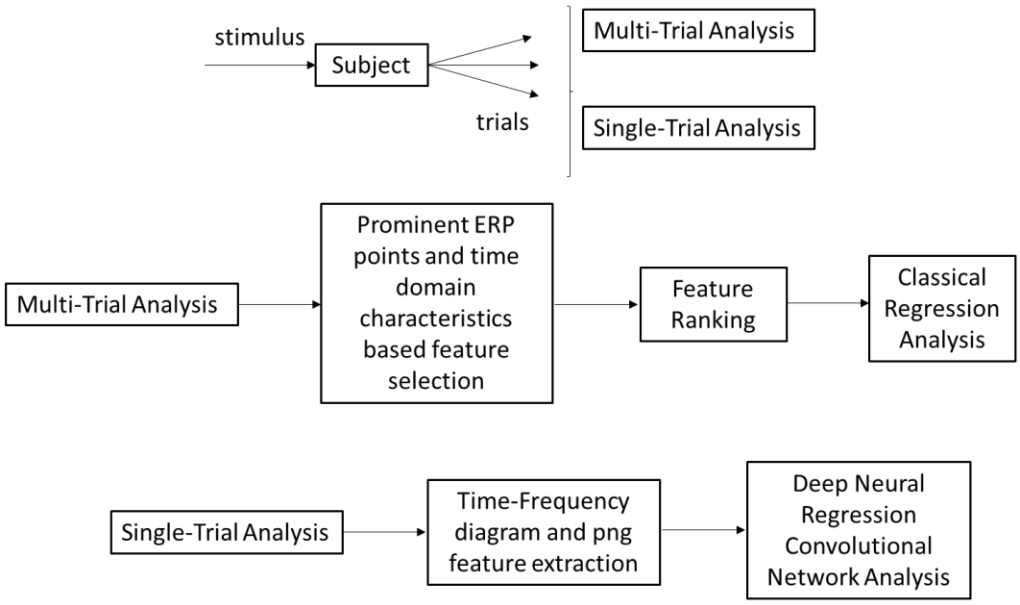


Fig. 23: Flow Diagram of the method

4.5 Results

In the multi-trial analysis, we used the top 13 features which were decided by the outcome of the lasso regression-based feature extraction techniques in all the regression techniques mentioned in the regression section to predict MoCA scores from neural measures. To compare among the regression techniques used in the multi-trial analysis, we used normalized mean square error (NMSE). The scores are reported in TABLE 7.

TABLE 7. RMSE FOR REGRESSION METHODS USED FOR MULTI-TRIAL ANALYSIS

Method	NMSE
Multivariate Regression (link function = logit) (MR)	1.55
Ensemble Regression(ER)	0.08
Support Vector Regression(SVR)	0.01

The regression methods MR, ER, and SVR generates five-fold cross-validated NMSE 1.55, 0.08, and 0.01 respectively. Based on NMSE, Support Vector Regression (SVR) performs the best among all techniques. The second-best technique is the ensemble regression (ER). The actual and the predicted MoCA scores for different samples for MR, ER, and SVR were presented in Fig. 24. In the case of SVR, the actual and predicted data match very well. MVR fails to predict the data and performance of ER is in the middle. We performed residual analysis to check the robustness of the models.

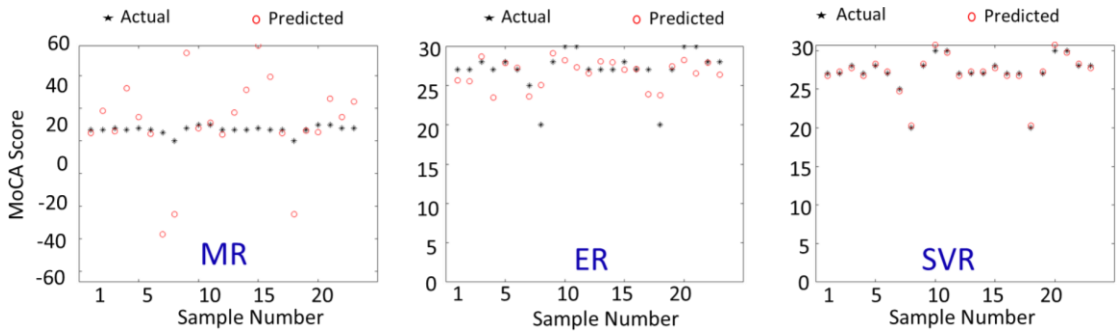


Fig. 24: Actual and Predicted MoCA score for Multivariate Regression (MVR), Ensemble Regression (ER) and Support Vector Regression (SVR)

In Fig. 25, we plotted the residual analysis results of the first two well performing regression model (SVR, ER). For each regression model, we presented two graphical analysis: a) sample quantiles vs. theoretical quantiles of the residuals or quantile-quantile plot (qqplot) and b) residuals vs fitted data by the regression model. The qqplot of the residuals is generally used to verify the assumption that the residuals are normally distributed. As both of the models show normal patterns (approximately straight line) in the qqplot, the models pass the normality test. Then we analyzed the residual vs fitted plot. A good regression model should be homoscedastic which means its residual should be uncorrelated, uniform, should be random over the fitted or predicted value. The plot of SVR shows a pattern whereas the plot for ER shows randomness of the residual. After this analysis, we recommend ER model for this data rather than SVR although SVR gave NMSE lower than the ER. It is because the ER model shows robustness in all the analysis, we state that ER the best among all the model we experimented. For the implementation of applying ER, we used MATLAB and for implementing MR, and, SVR, we used several packages such as “e1071”, “caret” etc. For residual analysis, we

used MATLAB.

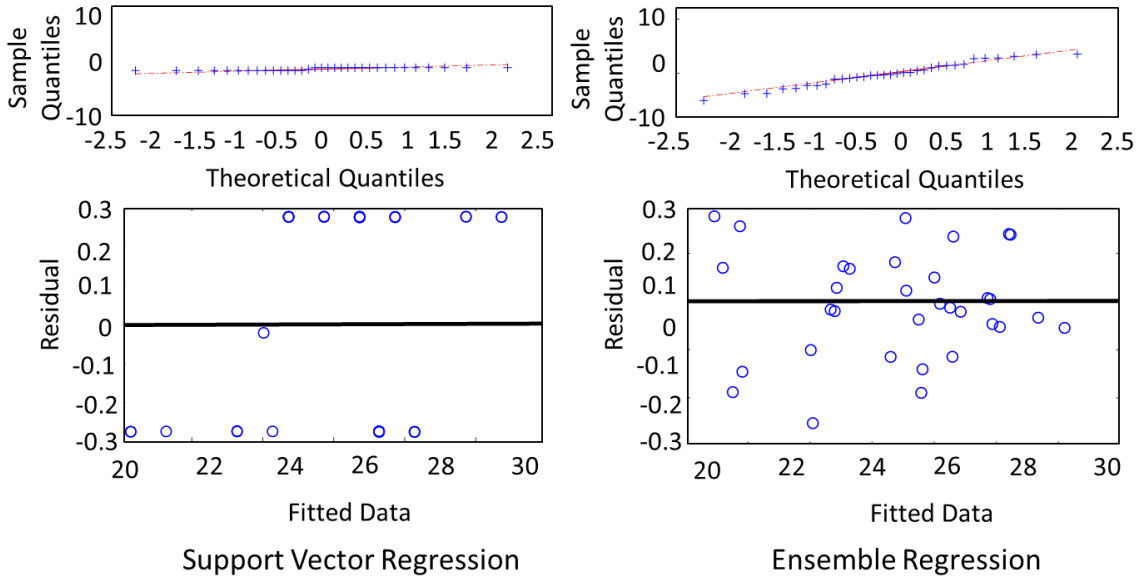


Fig. 25: Residual Analysis for Support Vector Regression (SVR) and Ensemble Regression (ER) Used in Multi Trial Analysis

TABLE 8. NMSE AND MAE FOR DEEP NEURAL NETWORK USED IN SINGLE TRIAL ANALYSIS

NMSE	MAE
0.09	1.1

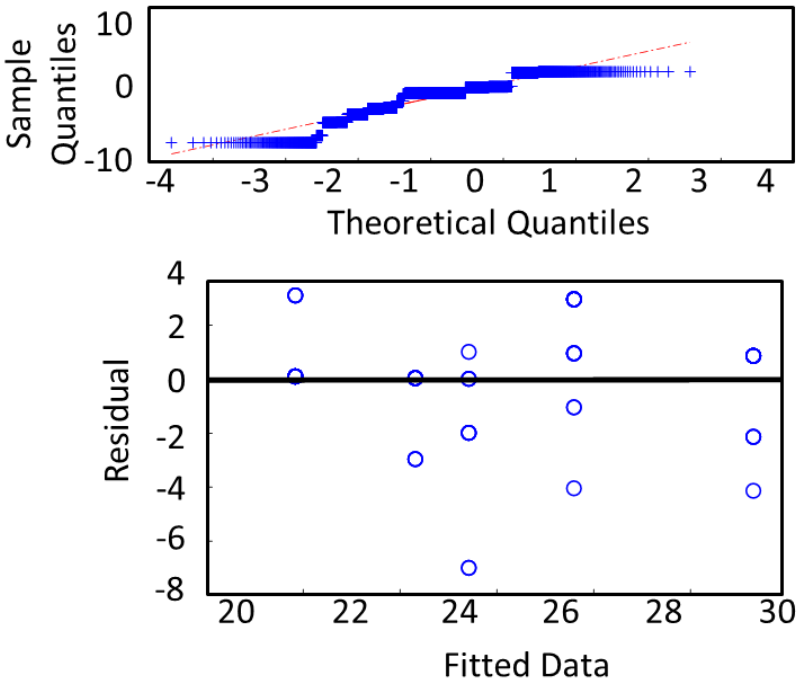


Fig. 26: Residual Analysis for Deep Neural Network Regression Model Used in Single Trial Analysis

In the single trial analysis, we calculated the cross validated NMSE and mean absolute error (MAE) for the constructed convolutional deep neural network. To check the robustness of the model, we also did a residual analysis which we presented in Fig. 26. We tried different combination of convolutional, max-pooling, and activation layer and calculated NMSE and MAE for all of them. The best result is reported here. Our deep neural network regression model able to achieve 0.09 NMSE and 1.1 MAE (TABLE 8).

In Fig. 26, We performed residual analysis of the deep neural network. We observed that the qqplot (sample quantiles vs. theoretical quantiles) of the residuals can be approximated as linear. So, the model maintains the normality assumption. Also, residuals vs. fitted plot shows that the residuals are not following pattern with the change of the fitted value.

4.6 Conclusion

Although MCI detection is being investigated by researchers in recent years, MCI severity measurement is a relatively new topic. In the current study, we generated the MoCA score automatically with good correlation from the single channel EEG data which will help to assess a person's cognitive health automatically with a low cost, portable, and less-burdensome setup. We carried out multi and single trial analysis and the results of both cases were promising. In the multi-trial analysis, we applied several classical regression models and found that the ER (Ensemble Regression) performed best with NMSE = 0.08. In the single trial analysis, our constructed convolutional deep neural network generated the MoCA scores with NMSE = 0.09. We are the first one as per our knowledge to generate MoCA score from single channel EEG data and also from both multi trial or single trial data from single channel EEG. We believe that our current study has a broad scope in cognitive health monitoring for MCI patients in mobile health (mHealth) and health status monitoring in a smart and connected community (SCC).

4.7 References

[1] M. J. Russo *et al.*, "Utility of the Spanish version of the Everyday Cognition scale in the diagnosis of mild cognitive impairment and mild dementia in an older cohort from the Argentina-ADNI," *Aging Clin. Exp. Res.*, vol. 0, no. 0, pp. 1–10, 2018.

[2] C.-L. Tsai, M.-C. Pai, J. Ukropec, and B. Ukropcová, "The Role of M. J. Russo et al., "Utility of the Spanish version of the Everyday Cognition scale in the diagnosis of mild cognitive impairment and mild dementia in an older cohort from the Argentina-ADNI," Aging Clin. Exp. Res., vol. 0, no. 0, pp. 1–10, 2018.

[3] S. Khatun, B. I. Morshed, and G. M. Bidelman, “Single channel EEG time-frequency features to detect Mild Cognitive Impairment,” in *2017 IEEE International Symposium on Medical Measurements and Applications, MeMeA 2017 - Proceedings*, 2017.

[4] D. R. Roalf *et al.*, “Quantitative assessment of finger tapping characteristics in mild cognitive impairment, Alzheimer’s disease, and Parkinson’s disease,” *J. Neurol.*, 2018.

[5] L. Maskill, “Mild cognitive impairment: A quiet epidemic with occupation at its heart,” *Br. J. Occup. Ther.*, vol. 81, no. 9, pp. 485–486, 2018.

[6] J. S. Lin *et al.*, “Screening for cognitive impairment in older adults: an evidence update for the US Preventive Services Task Force,” 2013.

[7] K. K. F. Tsoi *et al.*, “Recall Tests Are Effective to Detect Mild Cognitive Impairment: A Systematic Review and Meta-analysis of 108 Diagnostic Studies,” *J. Am. Med. Dir. Assoc.*, 2017.

[8] D. R. Roalf, P. J. Moberg, S. X. Xie, D. A. Wolk, S. T. Moelter, and S. E. Arnold, “Comparative accuracies of two common screening instruments for classification of Alzheimer’s disease, mild cognitive impairment, and healthy aging,” *Alzheimer’s Dement.*, 2013.

[9] K. Bennys, G. Rondouin, C. Vergnes, and J. Touchon, “Diagnostic value of quantitative EEG in Alzheimer’s disease,” *Neurophysiol. Clin.*, 2001.

[10] J. W. Kowalski, M. Gawel, a Pfeffer, and M. Barcikowska, “The diagnostic value of EEG in Alzheimer disease: correlation with the severity of mental impairment.,” *J. Clin. Neurophysiol.*, 2001.

[11] K. Bennys, G. Rondouin, C. Vergnes, and J. Touchon, “Quantitative EEG findings in different stages of Alzheimer’s disease.,” *Neurophysiol. Clin. Clin. Neurophysiol.*, 2001.

[12] F. J. Fraga, T. H. Falk, P. A. M. Kanda, and R. Anghinah, “Characterizing Alzheimer’s Disease Severity via Resting-Awake EEG Amplitude Modulation Analysis,” *PLoS One*, 2013.

[13] M. Brunovsky, M. Matousek, A. Edman, K. Cervena, and V. Krajca, “Objective assessment of the degree of dementia by means of EEG,” *Neuropsychobiology*, 2003.

[14] Y. Song, D. Zang, Z. Wang, J. Guo, Z. Gong, and Y. Yao, “Grand Total EEG Analyses: A Promising Method Predict The Severity of Cognitive Impairment in Alzheimer’s Disease,” *J. Neurol. Sci.*, 2014.

[15] R. C. Oldfield, “The assessment and analysis of handedness: The Edinburgh inventory,” *Neuropsychologia*, 1971.

[16] G. M. Bidelman, J. E. Lowther, S. H. Tak, and C. Alain, “Mild Cognitive Impairment Is Characterized by Deficient Brainstem and Cortical Representations of Speech,” *J. Neurosci.*, 2017.

[17] Z. S. Nasreddine *et al.*, “The Montreal Cognitive Assessment, MoCA: A Brief Screening Tool for Mild Cognitive Impairment,” *J. Am. Geriatr. Soc.*, 2005.

[18] G. M. Bidelman, S. Moreno, and C. Alain, “Tracing the emergence of categorical speech perception in the human auditory system,” *Neuroimage*, 2013.

[19] D. B. Pisoni, “Auditory and phonetic memory codes in the discrimination of consonants and vowels,” *Percept. Psychophys.*, 1973.

[20] G. M. Bidelman, J. W. Villafuerte, S. Moreno, and C. Alain, “Age-related changes in the subcortical-cortical encoding and categorical perception of speech,” *Neurobiol. Aging*, 2014.

[21] G. M. Bidelman and C. Alain, “Musical Training Orchestrates Coordinated Neuroplasticity in Auditory Brainstem and Cortex to Counteract Age-Related Declines in Categorical Vowel Perception,” *J. Neurosci.*, 2015.

[22] S. J. Aiken and T. W. Picton, “Envelope and spectral frequency-following responses to vowel sounds,” *Hear. Res.*, 2008.

[23] S. Luck, *An introduction to the event related potential technique*. 2005.

[24] A. Krishnan, J. T. Gandour, and G. M. Bidelman, “The effects of tone language experience on pitch processing in the brainstem,” *J. Neurolinguistics*, 2010.

[25] G. Musacchia, D. Strait, and N. Kraus, “Relationships between behavior, brainstem and cortical encoding of seen and heard speech in musicians and non-musicians,” *Hear. Res.*, 2008.

[26] J. Lopez-Calderon and S. J. Luck, “ERPLAB: an open-source toolbox for the analysis of event-related potentials,” *Front. Hum. Neurosci., 2014.*

[27] L. Hu, P. Xiao, Z. G. Zhang, A. Mouraux, and G. D. Iannetti, “Single-trial time-frequency analysis of electrocortical signals: Baseline correction and beyond,” Neuroimage, 2014.

[28] S. Khatun, R. Mahajan, and B. I. Morshed, “Comparative Study of Wavelet-Based Unsupervised Ocular Artifact Removal Techniques for Single-Channel EEG Data,” IEEE J. Transl. Eng. Heal. Med., vol. 4, 2016.

[29] S. Khatun, R. Mahajan, and B. I. Morshed, “Comparative analysis of wavelet based approaches for reliable removal of ocular artifacts from single channel EEG,” in IEEE International Conference on Electro Information Technology, 2015, vol. 2015–June.

[30] G. M. Bidelman, “Towards an optimal paradigm for simultaneously recording cortical and brainstem auditory evoked potentials,” J. Neurosci. Methods, 2015.

[31] J. J. Lister, A. L. Harrison Bush, R. Andel, C. Matthews, D. Morgan, and J. D. Edwards, “Cortical auditory evoked responses of older adults with and without probable mild cognitive impairment,” Clin. Neurophysiol., 2016.

[32] B. Blankertz, S. Lemm, M. Treder, S. Haufe, and K.-R. Mueller, “Single-trial analysis and classification of ERP components - A tutorial,” Neuroimage, vol. 56, no. 2, pp. 814–825, 2011.

[33] K. van der Hiele et al., “EEG and MRI correlates of mild cognitive impairment and Alzheimer’s disease,” Neurobiol. Aging, 2007.

[34] R. Tibshirani, “Regression shrinkage and selection via the lasso: A retrospective,” J. R. Stat. Soc. Ser. B Stat. Methodol., 2011.

[35] H. Aurlien et al., “EEG background activity described by a large computerized database,” Clin. Neurophysiol., 2004.

[36] E. Brochu, V. M. Cora, and N. De Freitas, “A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning,” ArXiv, 2010.

5. Conclusion and Future Directions

Brain monitoring using a small number of EEG sensors in non-clinical settings has drawn a lot of attention lately due to ease-of-use and low-cost. These real-time BCI applications like cognitive load assessment, diagnosis of brain disorders, fatigue prediction, and cognitive biometrics, need fast and efficient pre-processing algorithms to process and analyze raw brain signals reliably in real time. The most common artifact in the EEG signal is due to the ocular activity. This dissertation presents an effort to develop unsupervised ocular artifact removal algorithm and to develop algorithms to detect MCI and to measure MCI severity. The key findings of this dissertation are: (a) developing a wavelet based ocular artifact removal algorithm for single channel EEG system (Chapter 2); (b) developing a classical machine learning based algorithm to detect MCI from the single channel EEG data in response auditory stimulus (Chapter 3); (c) developing a classical regression with grand average ERP and a deep learning with single trial ERP based method to measure the severity of MCI from a single channel EEG system. This chapter summarizes the key findings from each chapter and outlines future directions of this research that can further contribute to the studies presented here. Detailed discussions and conclusions reached from each study are presented at the end of the individual chapters.

In Chapter 2, the data from the AF3 channel of a 10-20 EEG system was used as a proof of concept of a single channel EEG data, which was contaminated with ocular artifacts (OA). In this study, several WT based methods were used to find out the best technique for OA removal. Based on the different performance metric-based results, DWT with ST using *coif3* and *bior4.4* wavelet basis functions have performed well for

OA removal while preserving neuronal signals in the non-blink regions based on CC, MI, SAR, NMSE and time-frequency analysis. However, the algorithm developed here is not specific to the channel and is applicable to EEG recordings from any channel of 10-20 EEG system. It is to be noted that DWT is a faster technique which require less computational resources and thus it is suitable for real-time analysis.

In Chapter 3, an MCI detection method based on single-channel EEG data was demonstrated. In the study, scalp EEG data from Fpz location of twenty-three subjects were collected while the subjects were stimulated with five auditory stimuli. All the subjects' cognitive assessment was done by the Montreal Cognitive Assessment Test (MoCA). The 590 features from the Event-Related Potential (ERP) of the collected EEG signals, which included time and spectral domain characteristics of the response were extracted. The top 25 features, ranked by the random forest method, were used for classification models to identify subjects with MCI. Robustness of our model was tested using five-fold cross-validation while training the classifiers. Best results (leave-one-out-cross-validation accuracy 87.5%, sensitivity 85%, specificity 95%, and F score 88%) were obtained using support vector machine (SVM) method with Radial Basis Kernel (RBF) (sigma = 10/cost = 100). Similar performances were also observed with logistic regression (LR), further validating the results. These results suggest that this single-channel EEG based algorithm could provide a robust biomarker for early, cost-effective and portable, and continuous MCI patient monitoring.

In Chapter 4, a single-channel EEG based MCI severity monitoring algorithm was developed where MoCA score was generated from features extracted from single-channel EEG data. Two approaches have been taken to solve the problem: a) multi-trial analysis

and b) single-trial analysis. In the multi-trial analysis, 590 features were extracted from the prominent ERP points and time domain characteristics of the ERP. Lasso regression technique was used to select the best feature set. In this study, 13 best features were used in the classical regression techniques, which were Multivariate Regression (MR), Ensemble Regression (ER), Support Vector Regression (SVR), and Ridge Regression (RR). The best results were obtained from ER according to the RMSE 1.6 and residual analysis. In the single-trial analysis, time-frequency image from each trial was extracted and these image features used as input to the constructed convolutional deep neural network (CNN). This deep CNN model gave 1.43 RMSE. We believe that this method will be implementable in portable setup and provide cognitive healthcare for elderly people with MCI or patients who have self-reporting concerns.

Future direction regarding artifact removal algorithm development includes development of other artifact removal algorithm such as ECG, motion, and other types of artifacts and incorporating all of them in a single algorithm. Implementation of combined artifacts removal algorithm in the embedded platform and validating the process with large number of subjects would be a feasible extension of this work. In future, different thresholding, adaptive wavelet, wavelet packet transform, and single channel ICA with wavelet transform can also be investigated and compared with these findings. Future direction regarding MCI detection and measurement of severity of the MCI can include collection of data from a large number of subjects, assessing the cognitive health of the subjects by more than one cognitive assessment tool such as MoCA, MMSE, GPCOG, and Recall test, comparing the sensitivity and effectiveness of various cognitive assessment tools, compare different EEG channel performance, investigating the impact

of classical and deep neural network based machine learning technique on them, and implementation of the algorithm in a wearable platform.

Appendix

In this section, the code to produce results in different sections are provided.

A1. Matlab code for the ocular artifact removal algorithm using statistical threshold

```
function reconstructed_eeg = NonOverlap1(raw_eeg, wname)
%% Parameter Description
% raw_eeg = EEG signal contaminated with artifacts
% wname = wavelet to decompose ('sym3', 'haar', 'coif3', 'bior4.4')
 %%
   Level = 8;
   [C L] = wavedec(raw_eeg,Level,wname); % wavelet decomposition
    j=L(1)+1;
%% Statistical Thresholding
   for i=1:1:6
      x = C(j:1:(j+L(i+1)-1));
      T = 1.5*std(x);
      for k=j:1:(j+L(i+1)-1)

           if(C(k)>T)
                C(k)=0;
             elseif(C(k)<-T)
                C(k)=0;
           end

 end
   j=j+L(i+1);
   end
%%
  reconstructed_eeg = waverec(C,L,wname); % wavelet reconstruction
end
```

A2. Matlab code for ocular artifact removal algorithm using universal threshold

```
function reconstructed_eeg = NonOverlap1(raw_eeg, wname)
%% Parameter Description
% raw_eeg = EEG signal contaminated with artifacts
% wname = wavelet to decompose ('sym3', 'haar', 'coif3', 'bior4.4')
 %%
```

```
    Level = 8;
    [C L] = wavedec(raw_eeg,Level,wname); % wavelet decomposition
     j=L(1)+1;
  %% Universal Thresholding
    for i=1:1:6
       x = C(j:1:(j+L(i+1)-1));
       m1=median(abs(C(j:(j+L(i+1)-1)))/0.6745); %threshhold as per Journal article
       T=sqrt(2*log(L(i+1))*m1);
       for k=j:1:(j+L(i+1)-1)

            if(C(k)>T)
                 C(k)=0;
             elseif(C(k)<-T)
                 C(k)=0;
            end
       end
    j=j+L(i+1);
    end
  %%
   reconstructed_eeg = waverec(C,L,wname); % wavelet reconstruction
  end
```

A3.Code to produce simulated artifact

```
function bl = eyeblink(sFreq)
%% Input
%  sFreq: sampling frequency of data collection
%sFreq = 128;
freq = 2;
nSample = ceil(sFreq/freq);
t = linspace(-1,1,nSample);
bl = (sinc(t)).^2*150;
%figure, plot(t,bl)
```

A4.Code to prove the ocular artifact removal algorithm with simulated artifact

```
clc;clear all;close all;
S = {};
CC = zeros(6, 1) ; MI = zeros(6, 1) ; MSE = zeros(6, 1) ;
Mean_AE = zeros(1, 4, 6);
 %input
Fs = 128;
wname = cellstr(['sym3   ';'coif3  ';'bior4.4';'db2    ';]);
% wname = cellstr(['haar']);
```

```
% window = 128:128:1280;
window = 128:128:1280;
 for z = 1:length(wname)
   for y = 1:length(window)

        sig3=importdata('t.csv');
        sig1 = sig3(1:2927); sig2 = sig3(8159:9984);
        sig = [sig1;sig2];
        sig = sig(1:4736) - mean(sig(1:4736));

        q = length(sig)/window(y);
        t = 0:1/Fs:(q-1)/Fs;

        sig_with_artifact = sig;
        blink = eyeblink(Fs)';
        len = length(blink);
        %%
        index1 = 501:500 + len;
        index2 = 1501:1500 + len;
        index3 = 2501:2500 + len;
        index4 = 3501:3500 + len;
        %%
        sig_with_artifact(index1) = sig_with_artifact(index1) + blink;
        sig_with_artifact(index2) = sig_with_artifact(index2) + blink;
        sig_with_artifact(index3) = sig_with_artifact(index3) + blink;
        sig_with_artifact(index4) = sig_with_artifact(index4) + blink;
         %%
        dummy_sig = zeros(1,length(sig_with_artifact));
        dummy_sig(index1) = blink;
        dummy_sig(index2) = blink;
        dummy_sig(index3) = blink;
        dummy_sig(index4) = blink;
        reconstructed_signal_nonoverlapped = NonOverlap1(sig_with_artifact,
window(y), wname{z});
         % figure()
        % plot(t, sig+750), hold on, plot(t, dummy_sig+450, 'r'), hold on, plot(t,
sig_with_artifact+200), hold on, plot(t, reconstructed_signal_nonoverlapped, 'k')
        % legend('pure eeg', 'artificial blink', 'eeg with artifact added', 'after artifact
removal')
        % set(gca,'FontSize',10);
        % figure()
        % plot(sig_with_artifact)
         %performance metric
```

```
        RR = corrcoef(sig, reconstructed_signal_nonoverlapped);
        CC(1) = RR(2);

        MM = minfo(sig', reconstructed_signal_nonoverlapped');
        MI(1) = MM;

        MSE(1) = sum((sig - reconstructed_signal_nonoverlapped).^2)/length(sig);

        Mean_AE(:, :, 1) = mae(reconstructed_signal_nonoverlapped, sig);

%#########################################################################
#######
        sig3=importdata('FinalRuhi21.csv');
        sig1 = sig3(1:2596); sig2 = sig3(7374:9984);
        sig = [sig1;sig2];
        sig = sig(1:5120) - mean(sig(1:5120));

        t = 0:1/Fs:(q-1)/Fs;
         sig_with_artifact = sig;
         sig_with_artifact(index1) = sig_with_artifact(index1) + blink;
        sig_with_artifact(index2) = sig_with_artifact(index2) + blink;
        sig_with_artifact(index3) = sig_with_artifact(index3) + blink;
        sig_with_artifact(index4) = sig_with_artifact(index4) + blink;

        reconstructed_signal_nonoverlapped = NonOverlap1(sig_with_artifact,
window(y), wname{z});
         %performance metric
        RR = corrcoef(sig, reconstructed_signal_nonoverlapped);
        CC(2) = RR(2);

        MM = minfo(sig', reconstructed_signal_nonoverlapped');
        MI(2) = MM;

        MSE(2) = sum((sig - reconstructed_signal_nonoverlapped).^2)/length(sig);

        Mean_AE(:, :, 2) = mae(reconstructed_signal_nonoverlapped, sig);



%#########################################################################
#######
        sig3=importdata('Editedjack11.csv');
        sig1 = sig3(1:1140); sig2 = sig3(8447:9984);
```

```
sig = [sig1;sig2];
sig = sig(1:2560) - mean(sig(1:2560));
 t = 0:1/Fs:(q-1)/Fs;
 sig_with_artifact = sig;
 if(index1(end) < length(sig))
    sig_with_artifact(index1) = sig_with_artifact(index1) + blink;
end
if(index2(end) < length(sig))
    sig_with_artifact(index2) = sig_with_artifact(index2) + blink;
end
if(index3(end) < length(sig))
    sig_with_artifact(index3) = sig_with_artifact(index3) + blink;
end
if(index4(end) < length(sig))
    sig_with_artifact(index4) = sig_with_artifact(index4) + blink;
end

reconstructed_signal_nonoverlapped = NonOverlap1(sig_with_artifact,
window(y), wname{z});
 %performance metric
RR = corrcoef(sig, reconstructed_signal_nonoverlapped);
CC(3) = RR(2);
 MM = minfo(sig', reconstructed_signal_nonoverlapped');
MI(3) = MM;
 MSE(3) = sum((sig - reconstructed_signal_nonoverlapped).^2)/length(sig);
 Mean_AE(:, :, 3) = mae(reconstructed_signal_nonoverlapped, sig);


%#######################################################################
#######
sig3=importdata('Editedjack21.csv');
sig1 = sig3(1:2001);
sig = sig1;
sig = sig(1:1920) - mean(sig(1:1920));
 t = 0:1/Fs:(q-1)/Fs;
 sig_with_artifact = sig;
 if(index1(end) < length(sig))
    sig_with_artifact(index1) = sig_with_artifact(index1) + blink;
end
if(index2(end) < length(sig))
    sig_with_artifact(index2) = sig_with_artifact(index2) + blink;
end
if(index3(end) < length(sig))
```

```
            sig_with_artifact(index3) = sig_with_artifact(index3) + blink;
        end
        if(index4(end) < length(sig))
            sig_with_artifact(index4) = sig_with_artifact(index4) + blink;
        end

        reconstructed_signal_nonoverlapped = NonOverlap1(sig_with_artifact,
window(y), wname{z});

        %performance metric
        RR = corrcoef(sig, reconstructed_signal_nonoverlapped);
        CC(4) = RR(2);
        MM = minfo(sig', reconstructed_signal_nonoverlapped');
        MI(4) = MM;
        MSE(4) = sum((sig - reconstructed_signal_nonoverlapped).^2)/length(sig);
        Mean_AE(:, :, 4) = mae(reconstructed_signal_nonoverlapped, sig);


%#######################################################################
#######
        sig3=importdata('FinalANki11.csv');
        sig1 = sig3(1:2294);
        sig = sig1;
        sig = sig(1:2176) - mean(sig(1:2176));
        t = 0:1/Fs:(q-1)/Fs;
        sig_with_artifact = sig;

        if(index1(end) < length(sig))
            sig_with_artifact(index1) = sig_with_artifact(index1) + blink;
        end
        if(index2(end) < length(sig))
            sig_with_artifact(index2) = sig_with_artifact(index2) + blink;
        end
        if(index3(end) < length(sig))
            sig_with_artifact(index3) = sig_with_artifact(index3) + blink;
        end
        if(index4(end) < length(sig))
            sig_with_artifact(index4) = sig_with_artifact(index4) + blink;
        end

        reconstructed_signal_nonoverlapped = NonOverlap1(sig_with_artifact,
window(y), wname{z});
```

```
        %performance metric
        RR = corrcoef(sig, reconstructed_signal_nonoverlapped);
        CC(5) = RR(2);
        MM = minfo(sig', reconstructed_signal_nonoverlapped');
        MI(5) = MM;
        MSE(5) = sum((sig - reconstructed_signal_nonoverlapped).^2)/length(sig);
        Mean_AE(:, :, 5) = mae(reconstructed_signal_nonoverlapped, sig);



%#########################################################################
#######
        sig3=importdata('FinalSah11.csv');
        sig1 = sig3(7596:9984);
        sig = sig1;
        sig = sig(1:2304) - mean(sig(1:2304));

        t = 0:1/Fs:(q-1)/Fs;
        sig_with_artifact = sig;

        if(index1(end) < length(sig))
           sig_with_artifact(index1) = sig_with_artifact(index1) + blink;
        end
        if(index2(end) < length(sig))
           sig_with_artifact(index2) = sig_with_artifact(index2) + blink;
        end
        if(index3(end) < length(sig))
           sig_with_artifact(index3) = sig_with_artifact(index3) + blink;
        end
        if(index4(end) < length(sig))
           sig_with_artifact(index4) = sig_with_artifact(index4) + blink;
        end
        reconstructed_signal_nonoverlapped = NonOverlap1(sig_with_artifact,
window(y), wname{z});

        %performance metric
        RR = corrcoef(sig, reconstructed_signal_nonoverlapped);
        CC(6) = RR(2);
        MM = minfo(sig', reconstructed_signal_nonoverlapped');
        MI(6) = MM;
        MSE(6) = sum((sig - reconstructed_signal_nonoverlapped).^2)/length(sig);
        Mean_AE(:, :, 6) = mae(reconstructed_signal_nonoverlapped, sig);
        S = [S; {CC, MI, MSE, Mean_AE}];
```

```
    end
end
```

A5.R code to rank feature in Chapter 3

```
install.packages('randomForest')
library(randomForest)
data <- read.csv("feature_vector.csv", header = FALSE)
#data[, 1] <- factor(data[, 1], labels = c("Mild Cognitive Impairment", "Normal"))
feature_name <- paste0(rep('F', 590), 1:590)
names(data) <- c('Class',feature_name)
data.rf <- randomForest(Class~., data, ntree=590, keep.forest=FALSE, mtry = 50,
                        importance=TRUE)
#imp <- data.rf$importance
#write.csv(imp, "Ranking.csv")
```

A6.Matlab code for SVM parameter search

```
clc;clear all;close all;

data = importdata('Feature_Vect.csv');
train = data(:, 1:501);
k=1;
degree = 2:4;
cost = [0.01 0.1 1 10 100];
sigma = [10^-5 10^-4 10^-3 10^-2 10^-1 1 10];

result_len = length(degree)*length(cost) + length(sigma)*length(cost);

crossvalidation_accuracy = zeros(1, result_len);
support_vector = zeros(1, result_len);


trainingData = train;
inputTable = array2table(trainingData, 'VariableNames', {'column_1', 'column_2',
'column_3', 'column_4', 'column_5', 'column_6',...
'column_7', 'column_8', 'column_9', 'column_10', 'column_11', 'column_12',
'column_13', 'column_14', 'column_15', ...
'column_16', 'column_17', 'column_18', 'column_19', 'column_20', 'column_21',
'column_22', 'column_23', 'column_24',...
'column_25','column_26'});
```

```
predictorNames = {'column_2', 'column_3', 'column_4', 'column_5', 'column_6',...
'column_7', 'column_8', 'column_9', 'column_10', 'column_11', 'column_12',
'column_13', 'column_14', 'column_15', ...
'column_16', 'column_17', 'column_18', 'column_19', 'column_20', 'column_21',
'column_22', 'column_23', 'column_24',...
'column_25','column_26'}
isCategoricalPredictor = [];

predictors = inputTable(:, predictorNames);
response = inputTable.column_1;

for i = 1:length(degree)
   for j = 1:length(cost)
      predictors = inputTable(:, predictorNames);
      classificationSVM = fitcsvm(...
            predictors, ...
            response, ...
            'KernelFunction', 'polynomial', ...
            'PolynomialOrder', degree(i), ...
             'KernelScale', 'auto', ...
               'BoxConstraint',cost(j) , ...
               'Standardize', true, ...
               'ClassNames', [0; 1]);

         % Create the result struct with predict function
   %        predictorExtractionFcn = @(x) array2table(x, 'VariableNames',
predictorNames(1:i));
         predictorExtractionFcn = @(x) array2table(x, 'VariableNames',
predictorNames(ind));
         svmPredictFcn = @(x) predict(classificationSVM, x);
         trainedClassifier.predictFcn = @(x)
svmPredictFcn(predictorExtractionFcn(x));

         % Add additional fields to the result struct
         trainedClassifier.ClassificationSVM = classificationSVM;
         trainedClassifier.About = 'This struct is a trained classifier exported from
Classification Learner R2016a.';
         trainedClassifier.HowToPredict = sprintf('To make predictions on a new
predictor column matrix, X, use: \n  yfit = c.predictFcn(X) \nreplacing "c" with the
name of the variable that is this struct, e.g. "trainedClassifier". \n \nX must contain
exactly 16 columns because this classifier was trained using 16 predictors. \nX must
contain only predictor columns in exactly the same order and format as your training
\ndata. Do not include the response column or any columns you did not import into
```

```
\nClassification Learner. \n \nFor more information, see <a
href="matlab:helpview(fullfile(docroot, "stats", "stats.map"),
"appclassification_exportmodeltoworkspace")">How to predict using an exported
model</a>.');

        % Extract predictors and response
        % This code processes the data into the right shape for training the
        % classifier.
        % Convert input to table

        isCategoricalPredictor = [isCategoricalPredictor; false];
        % Perform cross-validation
        partitionedModel = crossval(trainedClassifier.ClassificationSVM, 'KFold',
33);

        % Compute validation accuracy
        validationAccuracy1 = 1 - kfoldLoss(partitionedModel, 'LossFun',
'ClassifError');
        validationAccuracy(k) = validationAccuracy1;

        supportvector1 = sum(classificationSVM.IsSupportVector);
        supportvector(k) = supportvector1;
        k = k +1;
    end
end

for i = 1:length(sigma)
    for j = 1:length(cost)
%         predictors = inputTable(:, predictorNames(ind));
        classificationSVM = fitcsvm(...
            predictors, ...
            response, ...
            'KernelFunction', 'rbf', ...
            'KernelScale', sigma(i), ...
              'BoxConstraint',cost(j) , ...
              'Standardize', true, ...
              'ClassNames', [0; 1]);

        % Create the result struct with predict function
    %         predictorExtractionFcn = @(x) array2table(x, 'VariableNames',
predictorNames(1:i));
        predictorExtractionFcn = @(x) array2table(x, 'VariableNames',
predictorNames);
```

```
        svmPredictFcn = @(x) predict(classificationSVM, x);
        trainedClassifier.predictFcn = @(x)
svmPredictFcn(predictorExtractionFcn(x));

        % Add additional fields to the result struct
        trainedClassifier.ClassificationSVM = classificationSVM;
        trainedClassifier.About = 'This struct is a trained classifier exported from
Classification Learner R2016a.';
        trainedClassifier.HowToPredict = sprintf('To make predictions on a new
predictor column matrix, X, use: \n  yfit = c.predictFcn(X) \nreplacing "c" with the
name of the variable that is this struct, e.g. "trainedClassifier". \n \nX must contain
exactly 16 columns because this classifier was trained using 16 predictors. \nX must
contain only predictor columns in exactly the same order and format as your training
\ndata. Do not include the response column or any columns you did not import into
\nClassification Learner. \n \nFor more information, see <a
href="matlab:helpview(fullfile(docroot, "stats", "stats.map"),
"appclassification_exportmodeltoworkspace")">How to predict using an exported
model</a>.');

        % Extract predictors and response
        % This code processes the data into the right shape for training the
        % classifier.
        % Convert input to table

        isCategoricalPredictor = [isCategoricalPredictor; false];
        % Perform cross-validation
        partitionedModel = crossval(trainedClassifier.ClassificationSVM, 'KFold',
33);

        % Compute validation accuracy
        validationAccuracy1 = 1 - kfoldLoss(partitionedModel, 'LossFun',
'ClassifError');
        validationAccuracy(k) = validationAccuracy1;

        supportvector1 = sum(classificationSVM.IsSupportVector);
        supportvector(k) = supportvector1;
        k = k +1;
    end
end
```

A7.Matlab code for feature selection with lasso regression

```
clc;clear all;close all;
```

```
data = importdata('C:/Research1/DrBidelman/Feature_Vector.csv');

ind = find(data(:, 1) <= 26);
data(ind, 1) = 0;
ind = find(data(:, 1) > 26);
data(ind, 1) = 1;

%% deviance plot
rng('default') % for reproducibility
[B,FitInfo] = lassoglm(data(:, 2:end),data(:, 1),'binomial',...
    'NumLambda',25,'CV',5);
lassoPlot(B,FitInfo,'PlotType','CV');
legend('show','Location','best') % show legend

%% trace plot
lassoPlot(B,FitInfo,'PlotType','Lambda','XScale','log');

feature_no = 1:590;
% non_zero_feature_no = feature_no(B0 ~= 0);

%% Regularized model
indx = FitInfo.Index1SE;
B0 = B(:,indx);
nonzeros = sum(B0 ~= 0);
cnst = FitInfo.Intercept(indx);
B1 = [cnst;B0];

%% Residual Analysis

preds = glmval(B1,data(:, 2:end),'logit');
histogram(data(:, 1) - preds) % plot residuals
title('Residuals from lassoglm model')
```

A8.Matlab code to perform ensemble regression

```
clear all; close all;clc;

input = readtable('Feature_Vect.csv', 'ReadVariableNames', false);
selected_feature_ind = [20 35 36 149 157 180 199 299 356 374 394 409 563];
selected_feature_ind = selected_feature_ind + 1;

col_name = input.Properties.VariableNames;
```

```
for i = 1:13
   if (i == 1)
       formula = [col_name{1} '~' ];
       formula = [formula col_name{selected_feature_ind(1)}];
   elseif(i > 1 & i <= 590)
      formula = [formula '+' col_name{selected_feature_ind(end)}];
   end
end

rng(1)
Mdl = fitrensemble(input, formula,'OptimizeHyperparameters','auto',...
   'HyperparameterOptimizationOptions',struct('AcquisitionFunctionName',...
   'expected-improvement-plus'))

Mdlfinal = crossval(Mdl, 'kfold', 5);
%mse = resubLoss(Mdlfinal);
y = table2array(input(:, 1));
y1 = Mdlfinal.kfoldPredict;
RMSE= sqrt(mean((y - round(y1)).^2));  % Root Mean Squared Error
```

A9.Python code to perform Bayesian optimization

ObjectiveFunc.py:

```
import os
import dill
import numpy as np
import pandas as pd
import subprocess as sp
from colorama import Fore
from sklearn.datasets import make_classification
from sklearn.model_selection import cross_val_score
from sklearn.svm import SVC

from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Convolution2D, MaxPooling2D

from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
```

```
def objetive_func_deep_NN_regression(selected_parameter, TRAIN_DATA_1,
TRAIN_LABEL_1):
    def model_builder(n, selected_parameter):
        def mb():
            param1 = int(selected_parameter[0])
            param2 = selected_parameter[1]
            model = Sequential()
            model.add(Convolution2D(param1, 3, 3, input_shape=n))
            model.add(Activation('relu'))
            model.add(MaxPooling2D(pool_size=(2, 2)))

            model.add(Convolution2D(param1, 3, 3))
            model.add(Activation('relu'))
            model.add(MaxPooling2D(pool_size=(2, 2)))

            model.add(Convolution2D(param1*2, 3, 3))
            model.add(Activation('relu'))
            model.add(MaxPooling2D(pool_size=(2, 2)))

            model.add(Flatten())
            model.add(Dense(param1*2))
            model.add(Activation('tanh')) #-7.72 (3.61) MSE

            model.add(Dropout(param2))
            model.add(Dense(1))

            model.compile(optimizer='adadelta',  loss='mean_absolute_error',
metrics=['mae'])
            return model
        return mb
    seed = 7
    np.random.seed(seed)

    isDataValid = True
    n = TRAIN_DATA_1[0].shape
    # evaluate model with standardized dataset
    param3 = selected_parameter[2]
    estimator = KerasRegressor(build_fn=model_builder(n, selected_parameter),
nb_epoch=param3, batch_size=5, verbose=0)

    kfold = KFold(n_splits=5, random_state=seed)
    results = cross_val_score(estimator, TRAIN_DATA_1, TRAIN_LABEL_1,
cv=kfold, n_jobs=1)
```

```
    return [isDataValid, results.mean()]
```

BayesianUtility.py:

```
import numpy as np
import sklearn.gaussian_process as gp
from scipy.stats import norm
import ObjectiveFunc as ob
from colorama import Fore
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
import random


"""
These fucntion reshapes the array to feed to the
gaussian model (2d and nd)
"""
def vector_2d(array):
    return np.array(array).reshape((-1, 1))
def vector_nd(array, n):
    return np.array(array).reshape((-1, n))


"""
This function is using expected improvement acquisition function to calculate
the expected improvement at varoius points in the parameter space and returns
the expected improvement, sigma, and mu at various points in the parameter
space. Help is taken from https://thuijskens.github.io/2016/12/29/bayesian-
optimisation/
to build this function
"""
def expected_improvement(x,                    # possible samples
                    gaussian_process,       # gaussian model
                    evaluated_loss,         # score/output array
                    greater_is_better=False, # Define whether the desired score from
objective
                                        # function is better to be high or low
                    n_params=1):            # # number of hyper-parameters to be optimized

    x_to_predict = x.reshape(-1, n_params)
```

```
    mu, sigma = gaussian_process.predict(x_to_predict, return_std=True)

    if greater_is_better:
        loss_optimum = np.max(evaluated_loss)
    else:
        loss_optimum = np.min(evaluated_loss)

    scaling_factor = (-1) ** (not greater_is_better)

    # In case sigma equals zero
    with np.errstate(divide='ignore'):
        A = (mu - loss_optimum)
        Z = scaling_factor * A[0] / sigma
        expected_improvement = scaling_factor * A[0] * norm.cdf(Z) + sigma *
norm.pdf(Z)
        expected_improvement[sigma == 0.0] == 0.0
    return [-1 * expected_improvement, sigma, mu]


"""
This fucntion collects the presample (if they are not collected offline) and
saves the parameters and corresponding scores in the 'parameter_presample.csv',
and 'score_presample.csv' files. This fucntion returns void.
"""

def presample(
    objective_func,          # objective function
    TRAIN_DATA_1,
    TRAIN_LABEL_1,
    PRESAMPLE_PARAMETER_FILE, # filename of the presample parameter
    PRESAMPLE_OUTPUT_FILE,    # filename of the output at the presample
parameter
    n_sample,                # number of presamples
    num_of_hyperparameter,    # number of hyper-parameters to be optimized
    bounds,                  # bounds of each hyper-parameter
    steps):                  # Search steps of each hyper-parameters

    print('Presample collection started')
    scores = []
    parameters = []

    for iteration in range(1, n_sample + 1):
```

```
    selected_parameter = []
    print(Fore.YELLOW +'------------------------')
    print(Fore.RED +'Sample No.{} collection: '.format(iteration))
    print(Fore.YELLOW +'------------------------')
    for hyperparameter in range(0, num_of_hyperparameter):
        parameter_space = np.arange(bounds[hyperparameter, 0],
bounds[hyperparameter, 1]+steps[hyperparameter], steps[hyperparameter])
        #parameter_space = np.arange(bounds[hyperparameter,
PARAMS_LOWER_BOUND], bounds[hyperparameter,
PARAMS_UPPER_BOUND]+steps[hyperparameter], steps[hyperparameter])
        temp = parameter_space[random.randint(0, np.size(parameter_space)-1)]
        selected_parameter.append(temp)
        print('Parameter {0:2d} : {1:.2f}'.format(hyperparameter,
selected_parameter[hyperparameter]))
    #print(Fore.WHITE +
'No.ofHyperparameter#{}'.format(len(selected_parameter)))
    if iteration != 1:
        isTwoSamplesAreSame = True
        isParamsPreviouslySeen = []
        while isTwoSamplesAreSame == True:
          for i in range(0, num_of_hyperparameter):
              if selected_parameter[i] in np.matrix(parameters)[:,i]:
                  print(selected_parameter)
                  isParamsPreviouslySeen.append(True)
              else:
                  isParamsPreviouslySeen.append(False)

          isTwoSamplesAreSame  = all(isParamsPreviouslySeen)
          isParamsPreviouslySeen = []

          if isTwoSamplesAreSame == True:
            selected_parameter[0] = np.arange(bounds[0, 0], bounds[0, 1] + steps[0],
steps[0])

    isDataValid, area = objective_func(selected_parameter, TRAIN_DATA_1,
TRAIN_LABEL_1)
    if isDataValid == True:
        parameters.append(selected_parameter)
        scores.append(area)
    print('score: {0:.2f}'.format(area))
    filename1 = PRESAMPLE_PARAMETER_FILE
    filename2 = PRESAMPLE_OUTPUT_FILE
    np.savetxt(filename1, parameters, fmt = "%2.6f", delimiter = ",")
```

```
        np.savetxt(filename2, scores, fmt = "%2.6f", delimiter = ",")
"""
This fucntion implements the 'switch' operation
(used in Kernel Selection)
"""
def best_kernel(x):
    return {
            0: gp.kernels.Matern(),
            1: gp.kernels.RBF(),
            2: gp.kernels.RationalQuadratic(),
            3: gp.kernels.DotProduct(),
            4: gp.kernels.ExpSineSquared(),
            }.get(x, gp.kernels.Matern())

"""
This function finds the best Kernel used in the gaussian process (GP). This function
fits the presamples in the GP and computes the performance of the fit in terms of
the sum-squared-difference (SSD), and returns the 'best kernel' that has the smallest
SSD between the true data and model predicted data.
"""
def kernel_comparison(num_of_hyperparameter,    # number of hyper-parameters to
be optimized
                    PRESAMPLE_PARAMETER_FILE, # filename of the presample
parameter
                    PRESAMPLE_OUTPUT_FILE):   # filename of the output at the
presample parameter

    filename1 = PRESAMPLE_PARAMETER_FILE
    filename2 = PRESAMPLE_OUTPUT_FILE
    xp = pd.read_csv(filename1, header = None)
    yp = pd.read_csv(filename2, header = None)
    xp1 = xp.values
    yp1 = yp.values
    xp1 = vector_nd(xp1, num_of_hyperparameter)
    yp1 = vector_2d(yp1)
    SSE = []
    print(Fore.GREEN + "Comparing Kernel Performance:\n")
    # Specify Gaussian Process
    kernel = gp.kernels.Matern()
    gp1 = gp.GaussianProcessRegressor(kernel=kernel)
    # Fitting data into Gaussian Process
    gp1.fit(xp1, yp1)
    # Predicting data using Gaussian Process
```

```
    y_pred, sigma = gp1.predict(xp1, return_std=True )
    # Calculating and appending sum squared error
    er = np.sum((yp1-y_pred)**2)
    SSE.append(er)
    print(Fore.WHITE + "Sum square error of Matern Kernel: {}".format(er))

    # Specify Gaussian Process
    kernel = gp.kernels.RBF()
    gp1 = gp.GaussianProcessRegressor(kernel=kernel)
    # Fitting data into Gaussian Process
    gp1.fit(xp1, yp1)
    # Predicting data using Gaussian Process
    y_pred, sigma = gp1.predict(xp1, return_std=True )
    # Calculating and appending sum squared error
    er = np.sum((yp1-y_pred)**2)
    SSE.append(er)
    print(Fore.WHITE + "Sum square error of RBF Kernel: {}".format(er))

    # Specify Gaussian Process
    kernel = gp.kernels.RationalQuadratic()
    gp1 = gp.GaussianProcessRegressor(kernel=kernel)
    # Fitting data into Gaussian Process
    gp1.fit(xp1, yp1)
    # Predicting data using Gaussian Process
    y_pred, sigma = gp1.predict(xp1, return_std=True )
    # Calculating and appending sum squared error
    er = np.sum((yp1-y_pred)**2)
    SSE.append(er)
    print(Fore.WHITE + "Sum square error of Rational Quadratic Kernel:
{}".format(er))

    # Specify Gaussian Process
    kernel = gp.kernels.DotProduct()
    gp1 = gp.GaussianProcessRegressor(kernel=kernel)
    # Fitting data into Gaussian Process
    gp1.fit(xp1, yp1)
    # Predicting data using Gaussian Process
    y_pred, sigma = gp1.predict(xp1, return_std=True )
    # Calculating and appending sum squared error
    er = np.sum((yp1-y_pred)**2)
    SSE.append(er)
    print(Fore.WHITE + "Sum square error of Dot Product Kernel: {}".format(er))
```

```
    # Specify Gaussian Process
    kernel = gp.kernels.ExpSineSquared()
    alpha=1e3
    gp1 = gp.GaussianProcessRegressor(kernel=kernel, alpha = alpha)
    # Fitting data into Gaussian Process
    gp1.fit(xp1, yp1)
    # Predicting data using Gaussian Process
    y_pred, sigma = gp1.predict(xp1, return_std=True )
    # Calculating and appending sum squared error
    er = np.sum((yp1-y_pred)**2)
    SSE.append(er)
    print(Fore.WHITE + "Sum square error of ExpSineSquared Kernel: {}".format(er))
    min_error_kernel = np.argmin(SSE)
    kernel = best_kernel(min_error_kernel)
    return kernel

"""
This function explores through the search space with the help of expected
improvement function. That means it evaluates at the sample suggested by the
expected improvement function in each iteration.
"""
def explore(objective_func,            # objective function
            TRAIN_DATA_1,
            TRAIN_LABEL_1,
            PRESAMPLE_PARAMETER_FILE,  # filename of the presample parameter
            PRESAMPLE_OUTPUT_FILE,     # filename of the output at the presample
parameter
            EXPLORE_PARAMETER_FILE,    # filename of the explored parameter
            EXPLORE_OUTPUT_FILE,       # filename of the output at the explored
parameter
            EXPLORE_VAR_FILE,          # filename of the variance at the explored
parameter
            exploration_count,         # Number of points to be explored
            num_of_hyperparameter,     # number of hyper-parameters
            bounds,                    # corresponding bounds of the parameters
            model,                     # Fitted Gaussian model
            granularity_dist,          # granularity of the distribution
            greater_is_better):        # Define whether the desired score from objective
                                       # function is better to be high or low

    var_eimax = []
    mu_eimax = []
    eimax = []
```

```
    filename1 = PRESAMPLE_PARAMETER_FILE
    filename2 = PRESAMPLE_OUTPUT_FILE
    xp = pd.read_csv(filename1, header = None)
    yp = pd.read_csv(filename2, header = None)
    xp1 = xp.values
    yp1 = yp.values
    parameters = list(xp1)
    scores = list(yp1)

    print(Fore.YELLOW + 'Exploration started')
    for explore in range(1, exploration_count + 1):
        print(Fore.LIGHTRED_EX + 'Exploration {} out of {}'.format(explore,
exploration_count))
        x_train = vector_nd(parameters, num_of_hyperparameter)
        y_train = vector_2d(scores)
        model.fit(x_train, y_train)
        x_test = np.random.uniform(bounds[:, 0], bounds[:, 1], size=(granularity_dist,
num_of_hyperparameter))
        ei_var_m = expected_improvement(x_test, model, y_train, greater_is_better,
num_of_hyperparameter)
        ei = -1*ei_var_m[0]
        var = ei_var_m[1]
        mu = ei_var_m[2]
        var_eimax.append(var[np.argmax(ei)])
        mu_eimax.append(mu[np.argmax(ei)])
        eimax.append(ei[np.argmax(ei)])
        next_sample = x_test[np.argmax(ei), :]
        if np.any(np.abs(next_sample - x_train) <= 1e-7):
            next_sample = np.random.uniform(bounds[:, 0], bounds[:, 1],
bounds.shape[0])
        for i in range(0, num_of_hyperparameter):
            print('Parameter {0:2d}: {1:.2f}'.format(i, next_sample[i]))
        isDataValid, score = objective_func(next_sample, TRAIN_DATA_1,
TRAIN_LABEL_1)
        if isDataValid == True:
            parameters.append(next_sample)
            scores.append(score)
        print('score: {0:.2f}'.format(score))

    filename1 = EXPLORE_PARAMETER_FILE
    filename2 = EXPLORE_OUTPUT_FILE
    filename3 = EXPLORE_VAR_FILE
```

```
    np.savetxt(filename1, parameters, fmt = "%2.6f", delimiter = ",")
    np.savetxt(filename2, np.matrix(scores), fmt = "%2.6f", delimiter = ",")
    np.savetxt(filename3, np.matrix(var_eimax), fmt = "%2.6f", delimiter = ",")


"""
This function defines the gaussian distribution with a mean at the exploitation
centre and a standard deviation = sigma at the exploitation center*parameter step.
Then it calulates the expected improvment at various points in the already
defined gaussian distribution and selects the best point as the sample to
evaluate. It continues to fit the newly evaluated sample and it's corresponding
output in the existing gaussian model and to find new best point with the help
of expected improvement function within a specified number of iterations.
"""
def exploit(objective_func,         # objective function
            EXPLORE_PARAMETER_FILE, # filename of the explored parameter
            EXPLORE_OUTPUT_FILE,    # filename of the output at the explored
parameter
            exploit_center,         # the sample around which we exploit
            n_params,               # number of hyper-parameters
            steps,                  # Search steps of each hyper-parameters
            n_iters,                # number of times we will search for the
                                    # best sample around exploitation center
            kernel,                 # kernel of the gaussian model
            granularity = 500,      # granularity of the distribution
            sigma = 0.05):          # sigma

    x_list = []
    y_list = []
    sigma_max = []
    mu_max = []
    ei_max = []

    # Find gaussian process regressor model
    model = gp.GaussianProcessRegressor(kernel=kernel)
    filename1 = EXPLORE_PARAMETER_FILE
    filename2 = EXPLORE_OUTPUT_FILE
    explored_parameter = pd.read_csv(filename1, header = None)
    explored_score = pd.read_csv(filename2, header = None)
    xp1 = explored_parameter.values
    yp1 = explored_score.values
```

```
    xp = np.array(xp1)
    xp = vector_nd(xp, n_params)
    yp = np.array(yp1)
    yp = vector_2d(yp)

    for n in range(n_iters):
        model.fit(xp, yp)
        # Sample next hyperparameter
        x_random = np.random.randn(granularity, n_params)
        for i in range(1, n_params):
          x_random[:,i] = x_random[:,i]* sigma*steps[i] + exploit_center[i]
        ei_var_m  = expected_improvement(x_random, model, yp, True, n_params)
        ei = -1*ei_var_m[0]
        var = ei_var_m[1]
        mu = ei_var_m[2]
        selected_parameter = x_random[np.argmax(ei), :]
        sigma_max.append(var[np.argmax(ei)])
        mu_max.append(mu[np.argmax(ei)])
        ei_max.append(ei[np.argmax(ei)])
        # Sample loss for new set of parameters
        isDataValid, score = objective_func(selected_parameter)

        if isDataValid == True:
            # Update lists
            x_list.append(selected_parameter)
            y_list.append(score)
        # Update xp and yp
        xp = np.array(x_list)
        yp = np.array(y_list)
        xp = vector_nd(xp, n_params)
        yp = vector_2d(yp)
    return xp, yp
```

Main.py:

```
import tempfile
import keras.models
from keras.optimizers import SGD
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation, Flatten
from keras.layers import Convolution2D, MaxPooling2D
from keras.callbacks import EarlyStopping
```

```
from keras.wrappers.scikit_learn import KerasRegressor
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold

import dill
from colorama import Fore
import numpy as np
import pandas as pd
import sklearn.gaussian_process as gp
import BayesianUtility as bu
import ObjectiveFunc as ob



PRESAMPLE_PARAMETER_FILE = 'parameter_presample.csv'
PRESAMPLE_OUTPUT_FILE = 'score_presample.csv'
EXPLORE_PARAMETER_FILE = 'parameters_explore.csv'
EXPLORE_OUTPUT_FILE = 'scores_explore.csv'
EXPLORE_VAR_FILE = 'var_explore.csv'

#filename = 'train_data.pkl'
#dill.load_session(filename)


TRAIN_DATA_1 = np.delete(train_data, 2847, 0)
TRAIN_LABEL_1 = np.delete(train_label, 2847, 0)



#######################################################################
num_of_hyperparameter = 3 # Filtersize, Dropout, Epoch
bounds = np.array([[16, 80], [0.05, 0.9],[10, 100]])
steps = [8, 0.05, 10]


n_pre_samples = 5                          # No. of presamples to be collected
exploration_count = 10                       # How many samples will be explored
greater_is_better = True

objective_func = ob.objetive_func_deep_NN_regression

exploitation_count = 5                      # How many times to exploit around an
exploitation center
```

```
max_exploitation_enter = 8                          # Maximum number of exploitation
centres to work on/ upper bound of the number of exploitations

granularity_dist=1000                               # granularity of the distribution
#*********************************************************************
*********************************************************************
def make_keras_picklable():
    def __getstate__(self):
        model_str = ""
        with tempfile.NamedTemporaryFile(suffix='.hdf5', delete=False) as fd:
            keras.models.save_model(self, fd.name, overwrite=True)
            model_str = fd.read()
        d = { 'model_str': model_str }
        return d

    def __setstate__(self, state):
        with tempfile.NamedTemporaryFile(suffix='.hdf5', delete=False) as fd:
            fd.write(state['model_str'])
            fd.flush()
            model = keras.models.load_model(fd.name)
        self.__dict__ = model.__dict__


    cls = keras.models.Model
    cls.__getstate__ = __getstate__
    cls.__setstate__ = __setstate__

make_keras_picklable()

#*********************************************************************
*********************************************************************

### COLLECTING PRESAMPLE
##############################################

presample_already_collected = False                 #
if presample_already_collected == False:            #
    bu.presample(objective_func, TRAIN_DATA_1, TRAIN_LABEL_1,
PRESAMPLE_PARAMETER_FILE, PRESAMPLE_OUTPUT_FILE,
n_pre_samples, num_of_hyperparameter, bounds, steps)

filename1 = PRESAMPLE_PARAMETER_FILE                #
filename2 = PRESAMPLE_OUTPUT_FILE                   #
```

```
xp_1 = pd.read_csv(filename1, header = None)
yp_1 = pd.read_csv(filename2, header = None)
xp1 = xp_1.values
yp1 = yp_1.values

xp1 = bu.vector_nd(xp1, num_of_hyperparameter)
yp1 = bu.vector_2d(yp1)

### AUTOMATIC KERNEL SELECTION AND MODEL FITTING
##############################
kernel = bu.kernel_comparison(num_of_hyperparameter,
PRESAMPLE_PARAMETER_FILE, PRESAMPLE_OUTPUT_FILE)
model  = gp.GaussianProcessRegressor(kernel=kernel)

### EXPLORATION
#################################################################
bu.explore(objective_func, TRAIN_DATA_1, TRAIN_LABEL_1,
PRESAMPLE_PARAMETER_FILE, PRESAMPLE_OUTPUT_FILE,
EXPLORE_PARAMETER_FILE, EXPLORE_OUTPUT_FILE,
EXPLORE_VAR_FILE, exploration_count, num_of_hyperparameter, bounds,
model, granularity_dist, greater_is_better)
```