Energy-Efficient and Robust Algorithms

for Biomedical Applications

Thesis by

Benyamin Haghi

In Partial Fulfillment of the Requirements for
the degree of
Doctor of Philosophy

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2024
Defense Date: March 26, 2024

Benyamin Haghi
ORCID: 0000-0002-4839-7647

To my supportive wife, Shadia;
my lovely parents, Mandana, and Behnam;
my parents in-law, Mina and Manouchehr;
my Ph.D. advisor, professor Azita Emami;
and to all my teachers, mentors, and advisors
for their continued support in my more than 20 years of study

ACKNOWLEDGEMENTS

At the beginning of my academic journey at Caltech as a Ph.D. student, I knew that my Ph.D. studies were going to be a life-changing chapter for me. Although the research projects I was working on were challenging for me and I had to figure out various scientific and application specific aspects of my projects, everything I learned in this journey helped me grow personally in ways I never imagined. With no doubt, my time at Caltech has been made special by the incredible scholars and researchers who I worked and collaborated with. Their intellectual generosity and spirit of collaboration stand out for me in the academic community.

I owe to everyone who have guided me along the way of my Ph.D. studies at Caltech. First and foremost, I want to deeply express my gratitude to my Ph.D. advisor, Prof. Azita Emami, for her support during my Ph.D. studies. Prof. Emami has influenced my academic life more than anyone else. Her intuition in framing questions and her relentless pursuit of knowledge have not only taught me a lot about research during my academic life, but they have also shaped my approach to the problem solving for the erst of my life indefinitely. On top of all that, she has provided an incredible support to me and all the other students in our lab.

I am grateful to have collaborated with Professor Richard A. Andersen, a pioneer in the field of brain-machine interfaces, and members of Andersen's lab, including Dr. Tyson Afflalo, Dr. Spencer Kellis, Dr. Jorge A. Gamez de Leon, Dr. Charles Guan, and Dr. Luke Bashford. Their combined experience contributed greatly to the success of our collaborative projects and the spirit of innovation. Special thanks to Dr. Nader Pouratian, the neurosurgeon for this project, and our colleagues at the Neurorestoration Center and Neurosurgery at USC Keck School of Medicine, such as Dr. Daniel Kramer, Dr. Brian Lee, and Dr. Charles Liu, who embody the spirit of cooperation as the driving force behind the growth of scientific knowledge.

Next, I want to thank my esteemed Ph.D. committee members: Prof. Yaser Abu-Mostafa, Prof. Richard A. Andersen, and Prof. P.P Vaidyanathan. It has been my great honor to have them as my committee members. Their insightful feedback and encouragement were absolutely crucial during my research at Caltech, and I cannot thank them enough for always pushing me to do better. Moreover, I want to thank Prof. Katie Bouman whose participation in my candidacy committee added invaluable insight to my work. While research can sometimes get narrow-minded, the members of my Ph.D. defense and candidacy committees provided fresh perspective into what I was doing, pointed out things I never would have seen otherwise.

I am really thankful for the strong sense of teamwork and shared intellectual curiosity with my colleagues at the MICS lab at Caltech. I want to specially thank my lab mates, Dr. Arian Hashemi Talkhuncheh, Dr. Saransh Sharma, Dr. Fatemeh Aghlmand, Dr. Mahsa Shoaran, Dr. Sahil Shah, Dr. Masoud Farivar, Dr. Milad Taghavi, Dr. Abhinav Agarwal, Dr. Xavier Chen, Shawn Sheng, Ting-Yu Cheng, Steven Bulfer, and Lin Ma for creating a friendly environment thanks to the cooperation and mutual support of many individuals in our lab. This environment has fostered not only academic talent, but also friendships that will last a lifetime.

I would like to acknowledge the support of several organizations such as the Heritage Medical Research Institute, the Carver Mead New Adventure Fund, and the Chen Institute of Neuroscience at Caltech who provided substantial financial support for our research projects, and the collaborative efforts that were provided by Prof. Anima Anandakumar’s and Prof. Yisong Yue’s labs in our heartbeat arrhythmia detection project. Special thanks to the undergraduate students who worked with our lab and collaborated in these projects, including Wei Foo, James Chen, Katie Chiu, Maitreyi Ashok, and Albert Yan Huang for their indispensable help in hardware and software validations, showcasing the collaborative spirit that defines our community. Their enthusiasm and contributions have been invaluable to our research’s success.

On a personal note, I want to thank my family for always being there for me, especially in the hardest times of my life by starting with my wife, Shadia and then my parents, Behnam and Mandana, also including my in-laws, Mina and Manouchehr. With no doubt, the support, endless love, and encouragement I have received from my family has been the most important part of my journey during my Ph.D. studies at Caltech. They have been my sanctuary and source of strength, enabling me to pursue my dreams with confidence.

Finally, I want to specially thank everyone who has been a part of my journey at Caltech over these years, and mention that their influence has extended far beyond the confines of my academic endeavors, profoundly shaping my personal and professional development. Their support, guidance, and friendship have been the greatest gifts of my time during my Ph.D. studies at Caltech.

ABSTRACT

Medical devices play a critical role in improving the quality of life for patients and assisting physicians by monitoring, detecting, and helping manage chronic conditions such as epilepsy and spinal cord injuries. To perform these functions effectively, these devices must extract the most relevant information from complex medical data. However, the functionality of these medical devices has been limited by the existing challenges in medical applications. Some of these challenges include the complexity in the analysis of raw medical data, adaptability, non-stationarity, noise, large data volumes, real-time processing, limited resources, and high accuracy demands. Moreover, considering factors such as individual differences, environmental influences, and genetic variations, medical data will cause numerous variations and uncertainties in analyzing and interpreting the medical conditions in different biomedical applications. Medical data analysis is already complex and is further complicated by issues like non-stationarity and noise, especially when using traditional and manual methods. When it comes to the designing, implementation, and utilization of wearable and implantable medical devices, efficiency, accuracy, and adaptability become crucial. Particularly, applications that require fast control of equipment, such as brain-machine interfaces (BMIs), make the need for fast decision-making evident. Medical data have been conventionally managed by reliance on extensive manual labor. However, such manual data management techniques are not scalable, have inefficient procedures, and are more likely to produce errors. Therefore, more advanced, automated methods are required immediately considering the existing challenges of the current medical data analysis techniques.

Such a shift in data processing and management will lead to more trustable procedures that can significantly improve the accuracy and efficiency of medical data analysis. Other than being just an improvement, such transformation signifies a noteworthy point in the development of medical devices. In this view, it is essential to introduce advanced technology and novel methods for medical data processing as well as automation. Therefore, it becomes critical that these high-performance and advanced techniques can efficiently be implemented with minimum effects on hardware for clinical applications. Currently,

artificial intelligence (AI) and its subfield machine learning (ML) has led to major transformations in designing and utilization of various medical devices. Among all these biomedical applications, three major area are addressed in this thesis: Brain Machine Interfaces (BMIs), seizure detection, and classification of arrhythmias in cardiac rhythms. We selected these three applications due to their significance and ability to improve patient treatment further. Additionally, we showed how we used machine learning algorithms for each of these applications to address their current challenges.

In our work related to Brain-Machine Interfaces (BMIs), we have been focused on improving the quality of life for individuals with spinal cord injury (SCI) through two studies. In our initial study, we have designed and implemented a deep multi-state Dynamic Recurrent Neural Network (DRNN) decoder for BMI applications. This algorithm decodes neural data recorded from the posterior parietal cortex (PPC) and the motor cortex (M1) of human participants to appropriate control signals to predict computer cursor kinematics on the computer screen. By reducing the amount of history used in predicting the movement kinematics from the recorded neural data, we have demonstrated that improved performance and robustness are preserved while memory and power consumption are reduced. We then compared the performance of DRNN with other decoding techniques to demonstrate that when operating on wavelet-based neural features, our proposed DRNN-based decoder outperforms other decoding techniques. Therefore, DRNN have the potential to be used for more efficient and effective BMIs. After developing DRNN as a decoding technique for BMI applications, we have implemented an efficient feature extraction technique, referred to as Feature Extraction Network (FENet), which has been designed by using convolutional neural networks for optimizing feature extraction and decoding to ensure consistency across electrodes when decoding the recorded neural data to the movement kinematics in BMI systems. After being tested with data recorded from the posterior parietal and motor cortices of three human participants, FENet outperformed existing feature extraction techniques such as threshold crossings and wavelet transforms, and it significantly enhanced both closed- and open-loop cursor controls. We have also evaluated the generalizability of FENet when applied to different datasets, brain regions, and participants. Therefore, the results of our research in BMI technology have the potential to promise the improvement of the quality of life for spinal cord injury (SCI) patients.

Second, we co-designed EKGNet, a convolutional network that combines analog computing and deep learning for detecting heartbeat arrhythmia. EKGNet demonstrated high accuracy while minimizing power consumption, effectively overcoming challenges related to analog circuitry and real-time processing. The experimental findings, using PhysionNet's MIT-BIH and PTB Diagnostics datasets, showed an average balanced accuracy of 95% for intra-patient arrhythmia classification and 94.25% for myocardial infarction (MI) classification.

Finally, we designed a real-time seizure detector by using XGboost as a technique relies on gradient boosted trees, which can help with the fast and accurate diagnosis of seizure for epileptic patients. With an averaged detection latency of 1.1 seconds, this design attained average F1 scores of 99.23% and 87.86% under various data splitting methods. The energy-area-latency product was 27× lower than the current state-of-the-art solutions, which allowed for adjustments that were specific to each patient and significantly reduced energy consumption.

The results presented in this dissertation demonstrate the potential of AI in addressing the existing challenges in three biomedical applications: brain-machine interfaces (BMI), seizure detection, and heartbeat arrhythmia detection. By addressing these existing challenges including complex biological data management, real-time processing constraints, and limited resources in biomedical applications, AI has the potential to improve the quality of life for patients suffering from neurological disorders and medical conditions. Moreover, the improved precision, operational efficiency, and flexibility caused by the integration of AI into the design of the future biomedical systems will potentially assist healthcare providers to offer enhanced support and treatment to patients. While we have focused on the three above-mentioned biomedical applications, the principles learned from our analysis may be relevant and can be extended to other biomedical applications.

PUBLISHED CONTENT AND CONTRIBUTIONS

B. Haghi, L. Ma, S. Lale, A. Anandkumar, A. Emami. "EKGNet: A 10.96µW Fully Analog Neural Network for Intra-Patient Arrhythmia Classification," *IEEE Biomedical Circuits and Systems (BIOCAS)*, 2023, DOI: 10.1109/BioCAS58349.2023.10389164
B.H. developed EKGNet, analyzed the results, and wrote the manuscript. B.H. also implemented EKGNet, contributed to experimental design and analysis, and contributed to the hardware/software co-design.

B. Haghi, S. Kellis, S. Shah, M. Ashok, L. Bashford, D. Kramer, B. Lee, Ch. Lie, R. A. Andersen, A. Emami, "Deep Multi-State Dynamic Recurrent Neural Networks Operating on Wavelet Based Neural Features for Robust Brain Machin Interfaces," *Advances in Neural Information Processing Systems (NeurIPS)*, 2019, DOI: 10.1101/710327
B.H. developed DRNN, analyzed the results, and wrote the manuscript. B.H. also implemented DRNN and contributed to experimental design and analysis.

S. Shah, B. Haghi, S. Kellis, L. Bashford, D. Kramer, B. Lee, Ch. Lie, R. A. Andersen, A. Emami, "Decoding Kinematics from Human Parietal Cortex Using Neural Networks," *9th International IEEE/EMBS Conference on Neural Engineering (NER)*, 2019. DOI: 10.1109/NER.2019.8717137
B.H. analyzed the performance of different decoding techniques, analyzed the results, and wrote the manuscript. B.H. also contributed to experimental design and analysis.

B. Haghi, T. Aflalo, S. Kellis, Ch. Guan, J. A. Gamez de Leon, A. Y. Huang, N. Pouratian, R. A. Andersen, A. Emami, "Convolutional Neural Network Feature Extraction Dramatically Improves Brain-Machine interface Control for Tetraplegic Participants," *Nature Biomedical Engineering, Submitted on Jan 9, 2023.* (Under Review)
B.H. developed FENet, analyzed the results, and wrote the manuscript. B.H. also implemented FENet and contributed to experimental design and analysis.

T. Aflalo, B. Haghi, R. A. Andersen, A. Emami, "Features Extraction Network for Estimating Neural Activity From Electrical Recordings," *U.S. Patent,* US20240046071A1*,* 2024.
B.H. developed FENet, analyzed the results, and wrote the manuscript. B.H. also implemented FENet and contributed to experimental design and analysis.

M. Shoaran, B. Haghi, M. Taghavi, M. Farivar, A. Emami, "Energy-Efficient Classification for Resource-constrained Biomedical Applications," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS)*, 8(4):693-707, 2018. DOI: 10.1109/JETCAS.2018.2844733

B.H. customized XGBoost for hardware implementation, analyzed the results, and wrote the manuscript. B.H. also contributed to experimental design and analysis and contributed to the hardware/software co-design.

M. Shoaran, M. Taghavi, B. Haghi, M. Farivar, A. Emami, "Energy-Efficient On-Chip Classifier for Detecting Physiological Conditions," *U.S. Patent*, 078554-8098.US01, 16/9946,151, 2020.
B.H. customized XGBoost for hardware implementation, analyzed the results, and wrote the manuscript. B.H. also contributed to experimental design and analysis and contributed to the hardware/software co-design.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Number *Page*

LIST OF TABLES

Number *Page*

Chapter 1

INTRODUCTION

Medical devices, including implantable and wearable technologies, are designed to improve the quality of treatments provided to patients and assisting physicians in monitoring, detection, and management of chronic illnesses such as epilepsy and spinal cord injuries. To fulfill these functionalities, these medical devices need to extract the most pertinent information from the complex medical data recorded from patients. However, the extraction of this information necessitates that these devices overcome challenges that are inherent in the treatment of medical conditions and disorders, such as real-time data processing, high accuracy demands, adaptability to dynamic patient conditions, and enhanced automation. Furthermore, to become clinically applicable, these medical devices need to successfully address challenges such as the limited availability of resources, the management of raw and noisy data, and the inherent non-stationarity of medical data [1], [2]. The problem of non-stationarity, which denotes the continuous variability inherent in the data, makes it more difficult to use static models in the setting of dynamic biomedical systems. Beyond the challenge of non-stationarity, the presence of noise introduces additional complexities in the acquisition of accurate and interpretable medical data. This noise stems from two main sources: shortcomings in the data recording devices and disruptions caused by external environmental factors [3], [4], [5]. The intersection of these elements greatly escalates the challenge of extracting pertinent and interpretable information from medical data, thus impeding the capacity to guarantee both clarity and accuracy in the datasets collected. Moreover, these challenges in addition to the complexity of the inherent patterns in the recorded medical data makes it difficult for the existing medical devices using conventional data analysis techniques to extract the most pertinent information from the data useful for different medical applications. Therefore, building upon the identified challenges within the domain of medical data analysis, there emerges an urgent need for the utilization of advanced signal processing and statistical techniques to address these complex issues.

Considering these challenges in extracting pertinent information from complex and dynamic medical data sets, it becomes crucial to advance our techniques to address these challenges. To effectively address these unexpected challenges, the proposed solutions need to be able to dynamically adapt to the changes in the data. Moreover, considering the significant consequences for patient care due to delays in processing and interpreting data, the ability of the proposed solutions to make decisions in a timely and accurate way becomes critical for a wide range of medical applications. When it comes to improving patients' quality of life, brain-machine interfaces [2], [6] perfectly illustrate the important of medical devices quickly interpreting the recorded neural data and responding effectively. Processing systems that are both high-performance and real-time are necessary in order to accomplish this requirement. In addition to providing users or medical professionals with fast feedback, these biomedical systems need to be able to quickly evaluate data, come to conclusions in a short amount of time, and make accurate decisions. Furthermore, the flexibility of these biomedical systems is crucial due to the wide range of different patient characteristics that may appear in the corresponding recorded data, which could make addressing the current challenges even more difficult. To overcome the adaptability challenge, medical devices should be designed to not be overly reliant on detailed patient-specific calibration. To effectively address the diverse requirements and situations of various individuals, possessing the capability for adaptation is critical. Consequently, this will ensure the widespread and effective utilization of the designed medical devices.

While medical devices running on traditional and manual data analysis techniques are still widely used in different medical applications, the traditional and manual techniques implemented on these devices may not always provide an optimal solution for medical conditions due to their technological incompatibilities, slow processing speeds, and efficiency limitations in general. Even established biomedical systems would fare considerably less favorably in the absence of automation. Moreover, analyzing and processing data manually is time-consuming, error-prone, and results in uncontrollable outcomes. These errors are more prone to arise in systems that rely heavily on human intervention, as they require significant time and resources. Manual analysis of the data demonstrates its shortcomings especially when organizing the massive data storage systems as regular practice in medical applications, which becomes a critical problem for both researchers and healthcare professionals. Furthermore, manually processing the data, especially in situations where quick decision-making is crucial, presents a significant obstacle. As a result, the existing manual approaches restrict the potential growth of biomedical systems as well as the prompt retrieval of pertinent information from the recorded medical data. Consequently, this shortcoming makes data administration and analysis less effective, which can potentially impede the medical field's progress. Therefore, the potential severity of these errors in the healthcare sector underscores the urgent need for biomedical systems to transition to automated, simplified, and reliable systems.

It is essential to emphasize the urgent need for novel approaches to bridge the gap between current techniques and their corresponding challenges in order to satisfy the constantly changing requirements of the medical industry. The use of automation, robust data processing methods, and cutting-edge technology should be the basis of these approaches. This allows us to reduce the difficult task of managing medical data, create faster solutions, and make more efficient use of computing capacity. Integration of these advanced data analysis techniques with biomedical systems minimizes the likelihood of human mistakes, thereby improves the reliability of medical devices and more importantly, the level of treatment for patients. Consequently, this shift in the medical industry necessitates a reconsideration of current concepts to develop new approaches for designing medical devices that can operate beyond current limitations. Instead of being objectives in themselves, the future development of the industry now requires automation and the replacement of human processes with more advanced data processing technologies. These techniques must be meticulously created to guarantee hardware compatibility in order to be successfully and effectively integrated into implantable or wearable devices that can operate in an actual clinical settings [7]. The data processing techniques that drive these medical devices are crucial to their efficacy such that contemporary healthcare would be inconceivable without them [7], [8], [9]. Moreover, for these methods to be useful in an actual clinical practice, they need to be generalizable to handle the noise and non-stationarity present in the recorded medical data. In order for researchers and medical professionals to have the resources they require to accurately, quickly, and appropriately handle the growing amount and complexity of medical data, incorporating these developments into the design and implementation of medical devices become very important. By improving patient treatments and changing the field of biomedical systems design, this significant advancement has the potential to notably transform healthcare.

Machine learning (ML) is one area of artificial intelligence (AI) that has recently advanced various applications [10], [11], including the design and implementation of biomedical systems and the development of their corresponding medical devices [3], [12], [13]. Previously incomprehensible healthcare difficulties are now much more understandable due to the greater insight provided by these recent, significant advances in machine learning. The adaptability and efficacy of machine learning techniques have been utilized across a wide range of healthcare sectors. More specifically, machine learning techniques have been utilized for the creation of more efficient brain-machine interface systems, the more accurate investigation of heartbeat arrhythmias, and the study of seizure occurrences for epileptic patients. With more demand to machine learning techniques in designing biomedical systems, there has been a shift in the requirements for applications used in the biomedical industry. At first, machine learning techniques have the potential to spot patterns in brief segments of intricate medical data [10], [11], [14], such as electrocardiograms (ECGs) or brain neuronal signals, which can be considered as a significant development in designing biomedical systems and medical devices for healthcare. Machine learning's ability to recognize patterns makes it a valuable tool for applications that require the identification of complex patterns in medical data. Furthermore, the fast data assessment times provided by machine learning techniques are advantageous for healthcare data management applications requiring a fast-decision-making process. To effectively synchronize human and machine thoughts, rapid decision-making is an important requirement. If designed properly, machine learning algorithms can potentially be trained as fast as is practically possible in order to handle data in real-time while using less power. In addition, the application of machine learning algorithms has benefited the implementation of medical devices by enabling the production of more accurate diagnoses [15]. The development of highly accurate machine learning models has promptly led to the emergence of new avenues for enhancing medical treatments. Furthermore, medical devices utilizing machine learning algorithms need to provide sufficient generalizability to adapt over time to various patient types and clinical environments. Recent machine learning algorithms have shown promising results in the development of effective biomedical systems when operating in adaptive clinical environments. For example, the recent machine learning techniques can now automatically adjust to user preferences and identify anomalies in medical data, which is only one example of how much progress has been made by using machine learning algorithms in healthcare. In due course, medical experts will possess the capability to objectively analyze the data to extract pertinent information and identify patterns for medical applications. Advancements in designing biomedical systems that enhance productivity, accuracy, or speed are highly valued, reflecting the healthcare industry's swift expansion and the escalating complexity of workforce requirements.

Contributing to the development of three applications with critical biomedical implications has been our goal since the beginning. Consequently, our research prioritizes three applications: Brain-Machine Interfaces (BMIs), heartbeat arrhythmia detection, and seizure detection, all have been selected according to the significance and complexity of the issues they aim to resolve. This focused approach is intentional, recognizing the indispensable role of BMIs in enhancing the spinal-cord injury (SCI) [5], [16] patient interaction with therapeutic devices, the importance of designing and implementing reliable arrhythmia identification and classification systems in the prevention and management of cardiovascular diseases, and the critical necessity of precise seizure detection for accurate epilepsy diagnoses. The growing importance of each application highlights the urgent need for focused research, driven by the significant challenges and impacts within these areas. Specifically, our goal is to see enhancements in medical device development and patient treatments stemming from our research. It is our hope that the findings of our research may be employed by the healthcare providers to address the existing challenges in the design, implementation, and utilization of medical devices in actual clinical settings. Each of these applications confronts significant obstacles that are inherently tied to the medical sector, underscoring the need for novel solutions.

Brain-machine interfaces (BMIs) as a relatively new area of study has the potential to significantly improve the lives of spinal-cord injury (SCI) patients and the patients with neurological disorders [2], [3], [6], [17], [18]. Since BMI systems provide direct brain-to-machine interaction, these systems represent a significant advance in neuroscience. This discovery is particularly important for brain circuit-related issues, for which traditional treatment approaches might not be adequate. Since BMIs are inherently novel, they offer new opportunities and inspire optimism by introducing approaches that have the potential to improve treatment for SCI patients significantly. In our research, we have been creating motor BMIs, which have been designed specifically to accommodate the needs of tetraplegic individuals. This is achieved in the framework of ongoing clinical trials by implanting microelectrode arrays into motor regions of SCI patients. With the aid of these specialized BMIs, SCI patients can mentally control robotic limbs or computer cursors through a better understanding of the complex neurological impulses related to movement intentions. [18], [19], [20], [21]. We introduce data science techniques that mainly rely on machine learning for feature extraction and decoding in BMI systems in this work. While prioritizing features like real-time functionality, generalizability, and limited end-to-end training architecture, this strategy aims to address the shortcomings of existing BMI systems while having negligible effect on the complexity of training data. Our proposed methods, DRNN [5] and FENet [22], have the potential to increase the overall efficacy and utility of implantable electrode systems in the actual clinical settings. By setting this plan into action, we show our commitment to improving the quality of life for SCI patients with brain circuit diseases and to furthering the field of neurotechnology.

Our second area of research has been the classification of arrhythmias in cardiac rhythms. In clinical practice, electrocardiograms (ECGs) are essential for monitoring heart health [23], [24]. Therefore, the precise identification and categorization of arrhythmic heartbeats are critical for the prevention and management of cardiovascular disease [23], [24], [25], [26].

The utmost importance is placed on automation and precision, given that manual ECG analysis is laborious and prone to human error [12]. In response to these obstacles, we present EKGNet, an integrated technique that merges deep learning and analog computation in order to construct a classification architecture for arrhythmias designed as a fully analog system [27]. EKGNet not only maintains low power consumption while attaining high balanced accuracies, but also takes advantage of the energy efficiency of transistors functioning in the subthreshold region. A novel analog sequential Multiply-Accumulate (MAC) circuit is integrated into the system design to reduce the impact of variations in process, supply voltage, and temperature. In this work, we introduce an additional performance enhancement to EKGNet through the incorporation of analog noise and discrepancies into its Bayesian neural network architecture [28]. By transferring knowledge from a teacher network to EKGNet via knowledge distillation [29], the efficacy of the network is enhanced. Furthermore, to optimize hardware performance, we present an algorithm that executes weight fine-tuning subsequent to quantization. Our proposed techniques in arrhythmia detection and classification are co-designed in hardware and software to potentially improve cardiovascular healthcare by addressing the difficulties linked to analog circuitry and the requirement for precise and reliable detection.

Turning to our third research emphasis, seizure detection is critical to our ongoing investigations. Epilepsy is a common neurological disorder with far-reaching consequences, thus accurate seizure detection is crucial for prompt diagnosis and treatment of epileptic patients [30]. Seizures can be identified through the utilization of low-power, implantable, or portable medical devices [7], [31]. The demand for real-time applications, stringent resource limitations, and a diverse array of potential applications beyond merely epilepsy serve as motivating factors for this work. Our primary focus has been on co-designing the algorithms with resource conservation in mind, upgrading hardware for power efficiency, devising patient-specific solutions, and seamlessly integrating them with existing medical devices. We have been working on these areas to enhance epilepsy diagnosis and treatment, hence improving the quality of life for epileptic individuals suffering from this neurological illness. Our study presents XGBoost [32], a gradient-boosted method for accurate seizure classification. According to its compatibility, our co-designed XGBoost meets the requirements of low-power design for portable or implantable medical devices. We provide a hardware solution for gradient-boosted tree creation in applications with power, area, and delay constraints. This design is energy and space efficient according to its asynchronous tree operation and consecutive feature extraction. Compared to the existing methods used in the design of the existing medical devices, our solution decreases energy-area-delay factor by 27 times. Moreover, gradient-boosting allows for adaptive patient-specific tree counts according to its flexibility. Using this technique, we could achieve a balance between detection accuracy and processing time. This classifier offers significant potential for low-power biomedical data processing beyond seizure detection. Our proposed method and the implemented device have the potential to be configured to run with varied energy requirements while still providing enhanced results for epileptic patients. Therefore, this work demonstrates our commitment to resource-efficient seizure detection.

Organization:

In the following chapters of this dissertation, the emphasis is on a detailed examination of the current challenges within three specific biomedical applications and the AI-driven solutions designed to improve their effectiveness. We will first introduce and discuss Brain-Machine Interfaces (BMIs) in Chapter 2, which is named “Brain-Machine Interfaces for Enhanced Control.” In this chapter, we delve into the use of artificial intelligence, specifically the employment of Deep Multi-State Dynamic Recurrent Neural Networks (DRNNs) [5] and the Feature Engineering Network (FENet) [22], as we strive to decode complex brain signals with the goal of gaining improved control. Throughout this chapter, we will see how artificial intelligence shows its potential flexibility by effectively handling diverse brain patterns. Therefore, our proposed designs can potentially allow for real-time, high-precision control over medical devices.

In Chapter 3, named as “Heartbeat Arrythmia Classification”, we go into the realm of cardiac arrhythmia classification to improve the accuracy and efficiency of analyzing recorded electrocardiogram (ECG) signals. In this chapter, we propose EKGNet, a fully analog convolutional architecture for on-chip heartbeat arrhythmia classification [27]. We have co-designed EKGNet with the purpose of mastering the complexities of ECG patterns and highlighting the crucial relevance of real-time processing while simultaneously retaining low energy usage.

As we go on to chapter 4, which we have titled as "Energy-Efficient Classification for Resource-Constrained Biomedical Applications," we shift our attention to the crucial field of early seizure identification for epileptic patients. The detection of seizure patterns within brain impulses is co-designed by using a machine learning technique based on gradient-boosted trees, named XGBoost, which is implemented in a 32-channel on-chip classifier and plays an important role in the application of this technology [7], [9], [33], [34]. The purpose of this study is to demonstrate how artificial intelligence can potentially meet the rigorous real-time processing requirements of seizure detection. This is accomplished by following to severe energy limits and adapting variances that are distinct to each individual patient.

Through its potential ability to solve common challenges such as data complexity, real-time processing, resource constraints, high accuracy requirements, and flexibility, machine learning algorithms can empower healthcare providers and improve the quality of patient treatment. Consequently, these studies aimed to provide an illustration of the potential influence that artificial intelligence may have in addressing existing healthcare challenges, which will eventually be of value to patients who are dealing with a variety of medical diseases and neurological defects. Chapter 1 has established the introduction for our work, and Chapter 5, which is the concluding chapter, will summarize our work and discuss about future directions by explaining the role that artificial intelligence can potentially play in determining the future of healthcare.

Chapter 2

BRAIN-MACHINE INTERFACES FOR ENHANCED CONTROL

This chapter includes three sections that explains different steps of our study to design and implement a BMI system. In section 2.1, we evaluate the performance of the existing neural-network-based decoders and compare their performance with the conventional Kalman filter for decoding computer cursor movement kinematics from the posterior parietal cortex (PPC) of a tetraplegic human subject. After evaluating the performance of the current neural network-based decoders, in section 2.2, we introduce a new decoder named deep multi-state dynamic recurrent neural network (DRNN), which is tuned for robust BMI applications and shows enhanced performance in predicting cursor movements from neural data. This follows the performance comparison of DRNN with the existing neural network-based and other learning-based decoders. In Section 2.3, we discuss about the importance of extracting pertinent and informative features for improving the decoding performance in BMI systems and we introduce FENet, a convolutional neural network-based feature extraction technique that improves cursor control for tetraplegic human participants by extracting appropriate features for BMI applications.

Overview

There are about 17,700 new cases per year of Spinal Cord Injury (SCI) in the United States [17]. SCI results in a partial or total loss of motor function. Brain machine-interfaces (BMIs), technologies that communicate directly with the brain, can improve the quality of life of millions of patients with brain circuit disorders [18]. Motor BMIs are among the most powerful examples of BMI technology: Ongoing clinical trials implant microelectrode arrays into motor regions of tetraplegic participants. Movement intentions are decoded from recorded neural signals into command signals to control a computer cursor or a robotic limb [19], [35], [36], [37], [38], [39]. There have also been efforts to use BMI to directly control paralyzed muscles [19], [35] and to decode speech signals from neural data [20], [21]. Figure 2.1.1 shows a general setup for a BMI system. BMI, in its most basic form, maps neural signals into useful movement control signals and then closes the loop to enable direct neural control of movements. Such systems have shown promise in helping SCI patients. However, these systems fail to deliver the precision, speed, degrees-of-freedom, and robustness of control enjoyed by motor-intact individuals. Even for simple movements, such as moving a computer cursor to a target on a computer screen, decoding performance can be highly variable over time. For example, electric potentials in the cortex have small amplitudes and are susceptible to noise, and electrical and mechanical properties of implanted microelectrodes change over time. Neuronal populations may also change over time. As a result, BMI decoders should be able to generalize across sources of variability to accurately infer movement commands from changing neural signals. Furthermore, most BMI systems currently run on high-power computer systems. Clinical translation of these systems will require decoders that can adapt to changing neural conditions and which operate efficiently enough to run on mobile, even implantable, platforms.

Conventionally, linear decoders have been used to find the relationship between kinematics and neural signals of the motor cortex. These linear algorithms used for such BMI systems have assumed a linear relation between inputs and outputs (e.g., Kalman filters or Wiener filters) [40]. For instance, Wu et al. [1] use a linear model to decode the neural activity of two macaque monkeys. Orsborn et al. [41] apply a Kalman filter, updating the model on batches of neural data of an adult monkey, to predict kinematics in a center-out task. Gilja et al. [36] propose a Kalman Filter to predict hand movement velocities of a monkey in a center-out task. However, all of these algorithms can only predict piecewise linear relationships between the neural data and kinematics. Moreover, because of non-stationarity and low signal-to-noise ratio (SNR) in the recorded neural data, linear decoders need to be regularly re-calibrated [1].

Recently, nonlinear machine learning algorithms have shown promise in attaining high performance and robustness in BMIs. For instance, Wessberg et al. [35] apply a fully connected neural network to neural data recorded from a monkey. Shpigelman et al. [44] show that a Gaussian kernel outperforms a linear kernel in a Kernel Auto-Regressive Moving

Average (KARMA) algorithm when decoding 3D kinematics from macaque neural activity. Sussillo et al. [5] apply a large FORCE Dynamic Recurrent Neural Network (F-DRNN) on neural data recorded from the primary motor cortex in two monkeys, and then they test the stability of the model over multiple days [4]. Zhang et al. [45] and Schwemmer et al. [14] extract wavelet-based features of motor cortex neural data of a human subject to classify intended hand movements by using a nonlinear support vector machine (SVM) and a large deep neural network, respectively. Hosman et al. [46] pass motor cortex neural firing rates to an LSTM and a Kalman filter to compare their performances for decoding intended cursor velocity of a human subject. These nonlinear learning-based decoders have shown more stability over multiple days and have improved performance compared to prior linear methods. However, they all have been applied to motor cortex data by mostly using neural firing rates as input features, which show more variability over long periods [1]. Recent work has demonstrated that neural activity in the posterior parietal cortex (PPC) can be used to support BMIs [2], [6], [7], [17], [47], [48], although the encoding of movement kinematics appears to be complex. PPC processes a rich set of high-level aspects of movement including sensory integration, planning, and execution [2] and may encode this information differently [48]. These characteristics of PPC differentiate it from other brain areas and, while providing a large amount of information to the decoder, also require new paradigms, such as those discussed here, to extract useful information. Therefore, extracting appropriate neural features and designing a robust decoder that can model this relationship in an actual BMI setting is required.

2.1 Decoding Kinematics from Human Parietal Cortex using Neural Networks

In this first section, we introduce our initial study focused on decoding kinematics from the posterior parietal cortex (PPC) of a tetraplegic human participant using brain-machine interfaces (BMIs) [16]. Our study employs advanced neural network models, including Deep Neural Network (DNN) [42], SimpleRNN (RNN) [43], and Long-Short Term Memory Recurrent Neural Network (LSTM) [44], and compare them with Kalman filter [41] as a

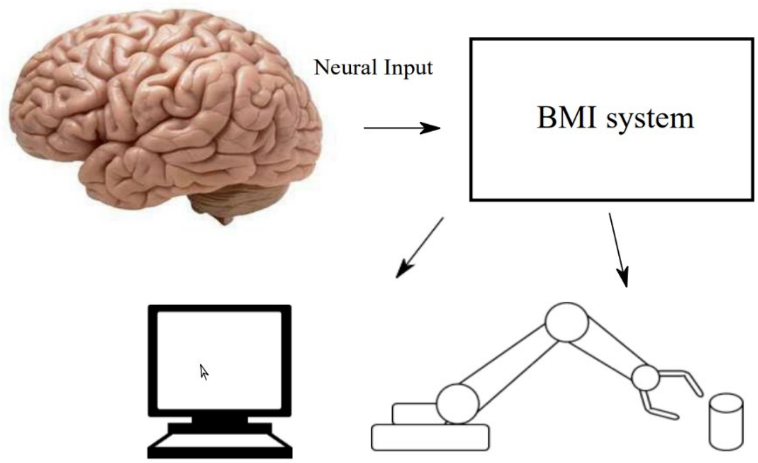


Figure 2.1.1. General setup of a Brain-Machine Interface (BMI) system. BMIs enable direct control of computers, prosthetics, and other peripheral devices by reading out and decoding brain activity. Advanced machine learning paradigms such as neural networks may be capable of learning the potentially complex relationship between recorded neural activity and control signals for these peripheral devices.

conventional decoder used for BMIs, to evaluate the performance of these neural-network-based decoders for translating neural signals into cursor movement kinematics during a 2D center-out task [16]. The subsequent discussion highlights the motivations, experimental setup, decoding algorithms, and key findings, offering a comprehensive overview of the study's objectives and outcomes.

The data used for training was recorded from the parietal lobe of a tetraplegic subject while the subject performed a 2D center-out task using motor imagery. We use Pearson Correlation Coefficient (ρ) as an accuracy metric. We report the accuracy of these decoders in open loop configuration, i.e., where the subject uses motor imagery while observing the task but is not in the control loop.

2.1.1. Architecture for the BMI System and Methods

In this sub-section, we will describe the architecture of the BMI system used in this study and the methods used to collect and process the neural data. The BMI system in this study consists of three main components: implanted electrodes, neural signal processing, and a decoder. The implanted electrodes record the electrical activity of neurons in the brain. The

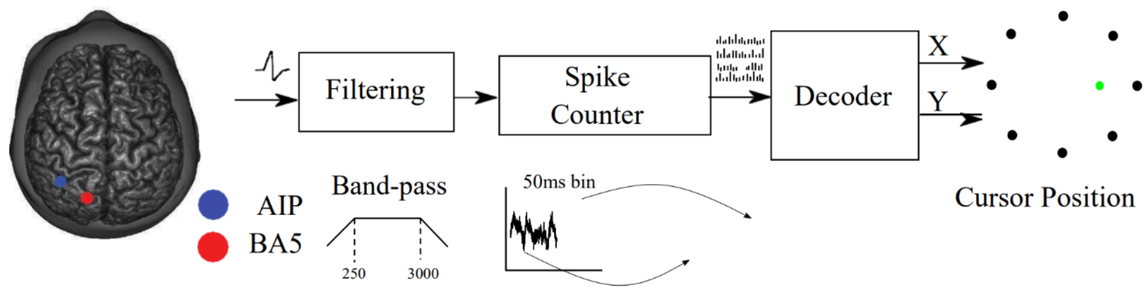


Figure 2.1.2. System architecture for decoding neural signals into relevant kinematics. Broadband recorded data were band pass filtered (250 Hz - 5 KHz) and thresholded at −4 times the noise RMS. Threshold crossing timestamps were binned in no overlapping 50ms intervals and smoothed to estimate the instantaneous threshold crossing rate. Decoding algorithms map these input features to corresponding X and Y coordinates of the cursor on screen.

neural signal processing component preprocesses the neural signals to identify neuronal action potentials and create spike train features for the decoder. The decoder uses the spike train features to predict the person's intended movement direction. In the following subsections, we will discuss each of these components in more detail.

2.1.1.1. Subject, Implanted Electrodes, and Recording

As part of our FDA- and IRB-approved study, two 96-channel Utah microelectrode arrays (Blackrock Microsystems, Inc., Salt Lake City, UT, USA) were implanted in the posterior parietal cortex (PPC) of a 32-year-old tetraplegic subject (EGS) with spinal cord lesions at C5-C6: one on the medial bank of the anterior intraparietal sulcus (AIP), and a second in Brodmann's area 5 (BA5) [2] (Figure 2.1.2). Data were recorded at 30,000 samples/sec.

2.1.1.2. Preprocessing the Neural Data

Figure 2.1.2 shows a top-level block diagram of a BMI system. Broadband data were filtered (Butterworth filter, 250 Hz - 5 KHz) and thresholded at −4 times the noise RMS of each channel to identify neuronal action potentials. These spiking events were binned at 50 ms intervals and smoothed to create spike train features for the decoding algorithms. To match the online case as closely as possible, action potential waveforms were not sorted, and spike trains were computed from the raw threshold crossings. The spikes recorded from both the electrodes were processed as described above and used as features for the decoder.

2.1.1.3. Center-Out Reaching Task

In this work, we use neural and behavioral data collected during the open-loop phase of a 2D center-out brain-control task. In this phase of the task, a cursor moves under computer control, with a minimum-jerk velocity profile, from the center of a computer screen to one of eight different target locations arranged uniformly around a unit circle, while the subject uses motor imagery to imagine controlling the cursor. Data is collected in three-minute blocks, each block consisting of 53 trials, with a pseudorandom uniform distribution of targets across trials. The dataset underlying this work consists of five such blocks recorded on the same day.

2.1.2. Decoding Algorithms

We used this recorded neural data to compare decoding performance between a Kalman filter asa conventional decoding technique used in BMI systems, with the performance of DNN, SimpleRNN, and LSTM. LSTM and SimpleRNN algorithms are used for this work since the prediction task and the input neural data are sequential.

2.1.2.1 Kalman Filter

The Kalman Filter [41] combines the idea that kinematics are function of neural firings as well as the idea that neural activity is a function of movements, or the kinematics. This can be represented by two equations:

$$\begin{cases} \hat{y}_{k+1} = A_k \hat{y}_k + w_k \\ u_k = H_k \hat{y}_k + q_k \end{cases} \qquad \text{Equation 2.1.1}$$

These represent how the system evolves over time as well as how neural activity is generated by the system's behavior. The matrices A, H, Q, and W can be found through a training process (where $q \sim \mathcal{N}(0, Q)$ and $w \sim \mathcal{N}(0, W)$. Using properties of the conditional probabilities of kinematics and neural data, we get a closed form solution for maximizing the joint probability $p(Y_M, U_M)$. Using the physical properties of the problem, we get matrix A to be of the form:

$$A = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & a & b \\ 0 & 0 & c & d \end{bmatrix} \qquad \text{Equation 2.1.2}$$

where A_v is defined as:

$$A_x = \begin{bmatrix} a & b \\ c & d \end{bmatrix} = X_2 X_1^T (X_1 X_1^T)^{-1} \qquad \text{Equation 2.1.3}$$

X_1 consists of the position kinematics points except for the last time step, X_2 consists of the position kinematics points except for the first time step, and dt is the time step size used (in our case, 50ms for our subject, EGS).

Furthermore, W is a zero matrix with the matrix $W_u = \frac{1}{N-1}(X_2 - AX_1)(X_2 - AX_1)^T$ in the bottom right corner. H and Q are given by:

$$\begin{cases} H = U^T Y (Y Y^T)^{-1} \\ Q = \frac{1}{N}(U - HY)(U - HY)^{-1} \end{cases} \qquad \text{Equation 2.1.4}$$

Then, we can use the update equations:

$$\begin{cases} \hat{y}_k^- = A\hat{y}_{k-1} \\ P_k^- = AP_{k-1}A^T + W \\ \hat{y}_k = \hat{y}_k^- + K_k(u_k - H\hat{y}_k^-) \\ P_k = (1 - K_k H)P_k^- \end{cases} \qquad \text{Equation 2.1.5}$$

Here, P is the covariance matrix of the kinematics. K_k, the Kalman filter gain is given by:

$$K_k = P_k^- H^T (H P_k^- H^T + Q)^{-1} \qquad \text{Equation 2.1.6}$$

2.1.2.2 Deep Neural Network (DNN)

In a fully connected neural network [42], there are multiple layers: an input layer, output layer, and any number of hidden layers with multiple nodes in each hidden layer. The output of each node in each layer is connected to the input of each node in the consecutive layer. Each node performs of $\sum_{i=1}^{N} W_i x_i$, where x_i is each input from the nodes in the previous layer and W_i is the weight of the connection between the node in the previous layer and this current node. The output is then converted to a normalized range using a function such as $tanh$ to get values between -1 and 1. W_i is trained through a process called back-propagation that trains the network on the inputs and finds the error, iteratively minimizing the loss function until the error stays relatively constant.

2.1.2.3 Long-Short Term Recurrent Neural Network (LSTM)

It is well-known that Simple RNN units cannot remember long term dependencies in sequential data because of the vanishing gradients problem [10]. Another version of RNNs that is widely used in the literature are RNNs with Long-Short Term Memory (LSTM) units [44]. By denoting ∘ as Hadamard product, the LSTM is defined as:

$$\begin{cases} f_k = \sigma(W_{fu} u_k + W_{fr} r_{k-1} + b_f) \\ i_k = \sigma(W_{iu} u_k + W_{ir} r_{k-1} + b_i) \\ o_k = \sigma(W_{ou} u_k + W_{or} r_{k-1} + b_i) \\ c_u = \tanh(W_{cu} u_k + W_{cr} r_{k-1} + b_c) \\ c_k = f_k \circ c_{k-1} + i_k \circ c_{k-1} \\ r_k = o_k \circ \tanh(c_k) \\ \hat{y}_k = W_{yr} r_k + b_y \end{cases} \qquad \text{Equation 2.1.7}$$

r_k is the hidden state as in Simple RNN, c_u is the output from the cell update activation function, c_k is the LSTM cell's internal state, f_k, i_k, and o_k are the output matrices from the respective forget, input, and output activation functions, which act as the LSTM's gates, W and b represent the weights and biases, and σ is the sigmoid function.

2.1.2.4 Recurrent Neural Network (SimpleRNN)

A vanilla recurrent neural network [43] with N hidden nodes for regression is defined as:

$$\begin{cases} r_k = \tanh\left(W_{rr} r_{k-1} + W_{ri} u_k + b_r\right) \\ \hat{y}_k = W_{yr} r_k + b_y \end{cases} \qquad \text{Equation 2.1.8}$$

where $r \in \mathbb{R}^N$, $\hat{y} \in \mathbb{R}^M$, and $u \in \mathbb{R}^I$ are the state, prediction, and input vectors, respectively, $W_{rr} \in \mathbb{R}^{N \times N}$, $W_{ru} \in \mathbb{R}^{N \times I}$, and $W_{yr} \in \mathbb{R}^{M \times N}$ are the weight matrices, $b_r \in \mathbb{R}^N$ and $b_y \in \mathbb{R}^M$ are the biases.

Because of the internal state r, which acts as a history unit, the RNN is capable of remembering and extracting short term temporal dependencies in sequential data. Therefore, to find the spatio-temporal relationship between the recorded neural data and kinematics as sequential data, we train an RNN with optimal parameters and compare its performance with the DRNN.

2.1.3. Training and Result Accuracy of the Decoders

The data were divided into training (80%), validation (10%) and test sets (10%). Training data was normalized to have zero mean and standard deviation of one to improve training algorithm convergence, but test and validation data were normalized using scales learned from the training data. Time bins in which the cursor did not move (zero velocity) were excluded from analysis. In the case of the neural networks, separate decoders were trained for predicting X and Y coordinates (Figure 2.1.3(a)).

The standard Kalman filter uses a model of the kinematic system, and a linear model of the relationship between the kinematics and the neural data, to form new estimates of the kinematics from noisy measurements of neural data [40]. Variants of the Kalman filter support nonlinear dynamics, but in general, Kalman filters require the researcher to establish a model of the dynamical system. In contrast, neural networks learn the model from training data.

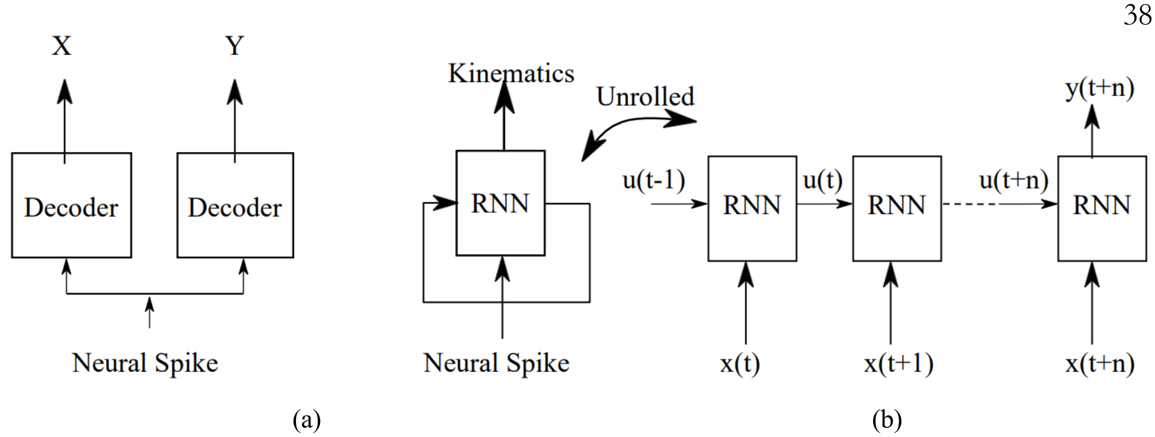


Figure 2.1.3. Output of the decoding algorithm. (a) For the neural network algorithms, two separate decoders are used to predict X and Y position of the cursor. (b) A block diagram of RNN [45] with a single dense layer for regression. Also, an unrolled block diagram of RNN with multiple time-steps. The RNN unit can be either a fully connected SimpleRNN cell or an LSTM unit cell.

We used two different neural network architectures: DNN and RNN. A DNN is a feedforward network with multiple layers and several nodes at each layer. The output of each node has a nonlinear activation function. DNNs with two layers have been shown to be a universal approximator [10]. A RNN is composed of feedforward network as well as a feedback network, meaning that all previous outputs are integrated to predict the next time-step (Figure 2.1.3(b)). RNNs also use previous time steps' input data when computing a new prediction. We tested two variants of RNN: one with LSTM unit cell [44] and one with the SimpleRNN unit cell [45].

The neural networks were trained using Keras with tensorflow backend and incorporate L1 regularization and 35% dropout for both the kernel and biases to reduce overfitting. RMSProp optimizer was used for training the network [13]. All three neural networks use the hyperbolic tangent as an activation function and incorporate a dense layer with one node and a linear activation function at the output to perform regression. Network parameters were heuristically tuned; future studies will explore optimization techniques to tune these parameters for higher accuracy. In general, optimization techniques such as Bayesian optimization, grid search, random search etc. are used to choose optimal network parameters. The number of layers and nodes used for decoding were nominal to avoid overfitting, but

Table 2.1.1. Parameters for the Neural Networks

Decoder	Nodes	Layers	Previous Neural Bins	Activation Function
LSTM	10 (X), 50 (Y)	LSTM + NN	40	tanh
RNN	25 (X), 25 (Y)	SimpleRNN + NN	20	tanh
DNN	25 (X), 25 (Y)	NN + NN	1	tanh

Table 2.1.2. Pearson Correlation Coefficient ρ For Each Decoder

	Kalman Filter	DNN	SimpleRNN	LSTM
X	0.24	0.20	0.46	0.47
Y	0.48	0.39	0.77	0.75

with a larger dataset one could increase the size of the network to predict with consistent accuracy.

Table 2.1.1 summarizes the parameters used for training these neural networks. The DNN had two layers with the first layer of the DNN composed of 25 nodes. The LSTM network for X position was set to 10 nodes with 40 time-steps of prior neural data, and the Y position was set to 50 nodes with 40 time-steps. The SimpleRNN network used 25 nodes and 20 time-steps of previous neural data for both X and Y coordinates.

Table 2.1.2 shows the accuracy of the four different decoders. The RNN algorithms, with the ability to incorporate historical data to compute new predictions, achieved the highest performance. The DNN exhibited the lowest performance, likely because it uses only a single time step of neural data to predict the current kinematics. The Kalman filter performed better than the DNN, perhaps also because its iterative nature 2.1.4(b) show the predicted X and Y

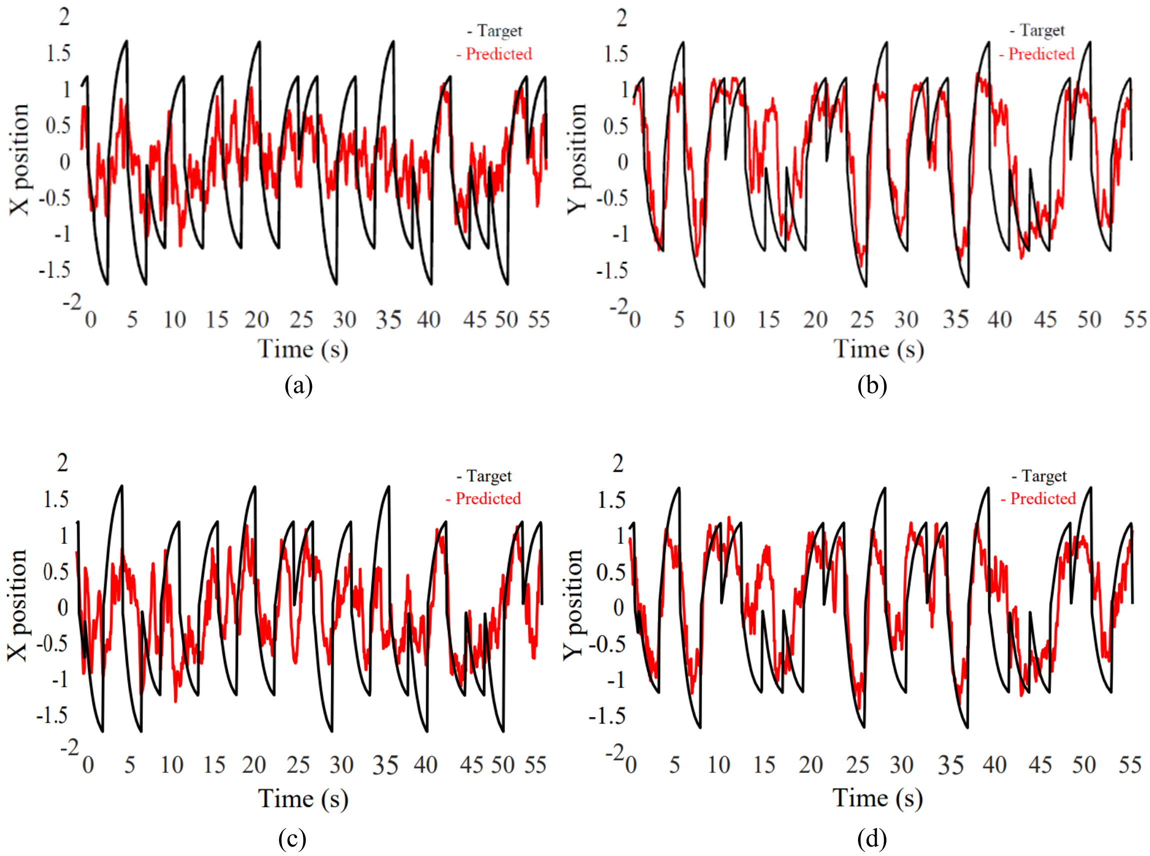


Figure 2.1.4. The predictions of the Decoders, (a) Output of a RNN with LSTM unit cell predicting the X coordinates of the cursor (ρ = 0.47). (b) Output of a RNN with LSTM unit cell predicting the Y coordinates of the cursor (ρ = 0.75). (c) Output of the decoder with SimpleRNN unit cell predicting X-coordinates of the cursor (ρ = 0.46). (d) Output of a RNN with SimpleRNN unit cell predicting the Y coordinates of the cursor (ρ = 0.77).

coordinates of the cursor for the LSTM unit cell with a ρ of 0.47 and 0.75, respectively, and figure 4(c) and figure 4(d) show the predicted X and Y coordinates of the cursor with a ρ of 0.46 and 0.77.

2.1.4. Summary

In this work we evaluated the performance of several different neural networks as the decoding techniques and compare their performance to a standard Kalman filter. Algorithms with the ability to incorporate historical data and network state demonstrated the highest performance (LSTM and SimpleRNN with the highest accuracies, and the Kalman filter with the next highest performance). LSTM also has the ability to recognize long-term dependencies in the data. Network paradigms with interconnected nodes and integration of historical data and states, such as the RNN variants tested in this work, may prove critical to first capturing the complexities of the relationship between neural activity and kinematic output, and second providing stable performance for BMI users. Our results showed a large difference in performance between X- and Y-dimension kinematics for this research participant. These differences are most likely attributable to the specific neuronal population recorded for the data used in this work, which may comprise different proportions of neurons modulated by movement in either axis. It is also possible that the research participant's cognitive strategy led to these differences. Further data must be collected to understand the source of these differences. Future work will test RNN decoders in closed loop to evaluate how well a human subject can use them for cursor control. Stability of the decoder over multiple days will also be evaluated. Also, this will determine whether the capability of the LSTM to capture long-term dependency leads to better performance over time.

While these algorithms are powerful in their capacity to capture complex relationships, they currently require power-hungry computational resources to operate. Part of making BMI systems clinically relevant is to design and develop size- and power-efficient hardware for decoding kinematics such that these systems can be implanted or worn on the body. Future directions would involve exploring such novel algorithms and energy-efficient hardware.

2.2. Deep Multi-State Dynamic Recurrent Neural Networks Operating on Wavelet Based Neural Features for Robust Brain-Machine Interfaces

After evaluating the performance of advanced machine learning algorithms on threshold crossings (TCs) as the extracted neural features, in this section, we present a new decoder, named deep multi-state Dynamic Recurrent Neural Network (DRNN) [5] architecture, which is designed for Brain Machine Interface (BMI) applications to address the challenges of performance, robustness, and potential hardware implementation. Our DRNN is used to predict Cartesian representation of a computer cursor movement kinematics from open-loop neural data recorded from the posterior parietal cortex (PPC) of a human subject in a BMI system. First, we refer to two theorems to show the stability, convergence, and potential of DRNNs for approximation of state-space trajectories. We then design an algorithm to achieve a reasonable trade-off between performance and robustness, and we constrain memory usage in favor of future hardware implementation. We feed the predictions of the network back to the input to improve the prediction performance and robustness. During the training of the model, we apply a scheduled sampling approach to the model in order to solve a statistical distribution mismatch between the ground truth and predictions during inference. Additionally, we configure a small DRNN to operate with a short history of input, reducing the required buffering of input data and number of memory accesses. This configuration lowers the expected power consumption in a neural network accelerator. By extracting different neural features, we compare the performance and robustness of the DRNN with the existing methods in the literature to predict hand movement kinematics from open-loop neural data. Our BMI data are recorded from the PPC of a human subject over 43 days. Operating on wavelet-based neural features, we show that the average performance of DRNN surpasses other state-of-the-art methods in the literature on both single- and multi-day data recorded over 43 days. Results show that multi-state DRNN has the potential to model the nonlinear relationships between the neural data and kinematics for robust BMIs.

2.2.1. Dynamic Recurrent Neural Networks

A general structure of a discrete-time DRNN is given by the following expressions:

$$\begin{cases} s_k = -as_{k-1} + f(W_{ss}, s_{k-1}, W_{su}, u_k, b_s) \\ \hat{y}_k = W_{yh^{(l)}} h_k^{(l)} + b_y \end{cases} \qquad \text{Equation 2.2.1}$$

where $s \in \mathbb{R}^N$, $\hat{y}_k \in \mathbb{R}^M$, and $u \in \mathbb{R}^I$ are the state, prediction, and the input vectors, respectively, $W_{ss} \in \mathbb{R}^{N\times N}$, $W_{su} \in \mathbb{R}^{N\times I}$, and $W_{ys} \in \mathbb{R}^{M\times N}$ are the weight matrices, $a \in [-1,1]$ is a constant controlling state decaying, $b_s \in \mathbb{R}^N$, and $b_y \in \mathbb{R}^M$ are the biases, and $f: \mathbb{R}^N \times \mathbb{R}^I \rightarrow \mathbb{R}^N$ is a vector-valued function.

2.2.1.1. Approximation of State-Space Trajectories

Theorem 2.2.1 verifies the approximation capability of DRNNs for the discrete-time and non-linear systems.

Theorem 2.2.1. Let $S \subset \mathbb{R}^M$ and $U \subset \mathbb{R}^I$ be open sets, $D_S \subset S$ and $D_U \subset U$ be compact sets, and $f: S \times U \rightarrow \mathbb{R}^M$ be a continuous vector-valued function which defines the following non-linear system

$$z_k = f(z_{k-1}, u_k), z \in \mathbb{R}^M, u \in \mathbb{R}^I \qquad \text{Equation 2.2.2}$$

with an initial value , $z_0 \in D_S$. Then for an arbitrary number $\epsilon > 0$, and an integer $0 < L < \infty$, there exist an integer N and a DRNN of the form Equation 2.2.1 with an appropriate initial state s_0 such that for any bounded input $u: \mathbb{R}^+ = [0, +\infty) \rightarrow D_U$

$$\max_{0\le k\le L} ||z_k - s_k|| < \epsilon \qquad \text{Equation 2.2.3}$$

Proof: See [46].

2.2.1.2. Local Stability and Convergence of DRNNs

Learning rate (γ) plays the main role in stability and convergence of neural networks. By using Lyapunov theorem, we define the range of the learning rate to guarantee the real-time convergence of DRNNs and the stability of the system during the whole control process.

Theorem 2.2.2. If an input series of internal dynamic neural network can be activated in the whole control process subject to $u_k \in \mathbb{R}^I$, then learning rate satisfies

$$0 < \gamma < \frac{2}{r^2} \qquad \text{Equation 2.2.4}$$

where $r = \frac{\partial e}{\partial W}$, $e = \hat{y} - y$ is the difference of prediction and ground-truth, and W is the concatenation of connection weights of each network unit. Then Equation 2.2.3 ensures the system is exponentially convergent.

Proof: see [47].

2.2.1.3 Deep multi-state dynamic recurrent neural network

A DRNN is a nonlinear dynamic system described by a set of differential or difference equations. It contains both feed-forward and feedback synaptic connections. In addition to the recurrent architecture, a nonlinear and dynamic structure enables it to capture time-varying spatiotemporal relationships in the sequential data. Moreover, because of state feedback, a small recurrent network can be equivalent to a large feed-forward network. Therefore, a recurrent network will be computationally efficient, especially for the applications that require hardware implementation [46]. We define our deep multi-state DRNN at each time step k as below:

$$\begin{cases} s_k = W_{ss}s_{k-1} + W_{sr}r_{k-1} + W_{su}u_k + W_{sf}z_{k-1} + b_s) \\ r_k = \tanh(s_k) \\ h_k^{(1)} = \tanh\left(W_{h^{(1)}h^{(1)}}h_{k-1}^{(1)} + W_{h^{(1)}r}r_k + b_{h^{(1)}}\right) \\ h_k^{(i)} = \tanh\left(W_{h^{(i)}h^{(i)}}h_{k-1}^{(i)} + W_{h^{(i)}h^{(i-1)}}h_{k-1}^{(i-1)} + b_{h^{(i)}}\right) \\ \hat{y}_k = W_{yh^{(l)}}h_k^{(l)} + b_y \\ \hat{y}_k = \tanh(\hat{y}_k)\,, |\hat{y}_k| > 1 \\ z_k \leftarrow \hat{y}_k \; or \; y_k(Scheduled\ Sampling\ during\ Training) \end{cases} \qquad \text{Equation 2.2.5}$$

$s \in \mathbb{R}^N$ is the activation variable, and $r \in \mathbb{R}^N$ is the vector of corresponding firing rates. These two internal states track the first- and zero-order differential features of the system,

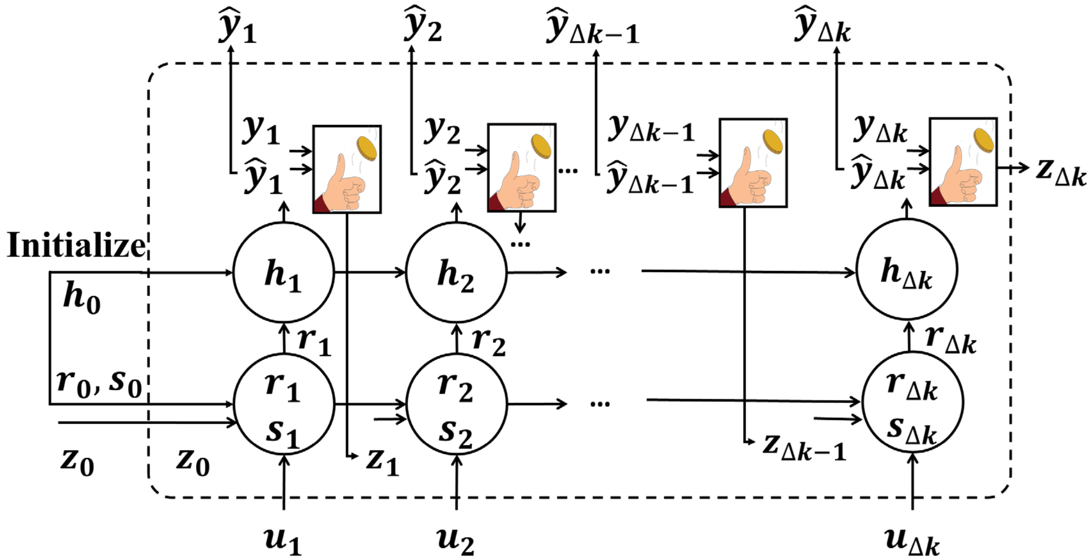


Figure 2.2.1. Training DRNN on a sample sequence of input data with length Δk.

respectively. Unlike conventional DRNNs, $W_{ss} \in \mathbb{R}^{N\times N}$ generalizes the dynamic structure of our DRNN by letting the network learn the matrix relationship between present and past values of s. $W_{sr} \in \mathbb{R}^{N\times N}$ describes the relationship between s and r. $W_{su} \in \mathbb{R}^{N\times N}$ relates s to the input vector u. $z \in \mathbb{R}^{M}$ models the added prediction feedback in our DRNN. $W_{sf} \in \mathbb{R}^{N\times M}$ tracks the effect of z on s. $i \in \{2,3,\dots,l\}$ and l is the number of layers, N_i is the number of hidden units in i^{th} layer, $h^{(i)} \in \mathbb{R}^{N_i}$ is the hidden state of the i^{th} hidden layer, $W_{h^{(1)}r} \in \mathbb{R}^{N_1\times N}$, $W_{h^{(i)}h^{(i)}} \in \mathbb{R}^{N_i\times N_i}$, $W_{h^{(i)}h^{(i-1)}} \in \mathbb{R}^{N_i\times N_{i-1}}$, $W_{yh^{(l)}} \in \mathbb{R}^{M\times N_l}$, $b_s \in \mathbb{R}^{N}$, $b_{h^{(i)}} \in \mathbb{R}^{N_i}$ are the weights and biases of the network. All the parameters are learnable in our DRNN. Although feed-forward neural networks usually require a deep structure, DRNNs generally need fewer than three layers. Algorithm 2.2.1 shows the training procedure. Inference is performed by using equation 2.2.1. Figure 2.2.1 shows the schematic of a two-layer DRNN operating on a sample sequence of input data with length Δk.

During inference, since the ground truth values are unavailable, the feedback, z_k, has to be replaced by the previous network predictions. However, the same approach cannot be applied during training since the DRNN has not been trained yet and it may cause poor performance of the DRNN. On the other hand, statistical discrepancies between ground truth and predictions mean that prior ground truth cannot be passed to the input. Because of this disparity between training and testing, the DRNN may enter unseen regions of the state-space, leading to mistakes at the beginning of the sequence prediction process. Therefore, we should find a strategy to start from the ground truth distribution and move toward the predictions' distribution slowly as the DRNN learns.

There exist several approaches to address this issue. Beam search generates several target sequences from the ground truth distribution [48]. However, for continuous state-space models like recurrent networks, the effective number of generated sequences remains small. SEARN is a batch approach that trains a new model according to the current policy at each iteration. Then, it applies the new model on the test set to generate a new policy which is a combination of the previous policy and the actual system behavior [49]. In our implementation, we apply scheduled sampling which can be implemented easily in the online case and has shown better performance than others [50].

In scheduled sampling, at the i^{th} epoch of training, the model pseudorandomly decides whether to feed ground truth (probability p_i) or a sample from the predictions' distribution (probability $(1 - p_i)$) back to the network, with probability distribution modeled by $P(y_{k-1}|r_{k-1})$. When $p_i = 1$, the algorithm selects the ground truth, and when $p_i = 0$, it works in Always-Sampling mode. Since the model is not well trained at the beginning of the training process, we adjust these probabilities during training to allow the model to learn the predictions' distribution. Among the various scheduling options for p_i [50], we select linear decay, in which p_i is ramped down linearly from p_s to p_f at each epoch e for the total number of epochs, E:

$$p_i = \frac{p_f - p_s}{E} e + p_s \qquad \text{Equation 2.2.6}$$

2.2.2 Other Methods

Since all of these methods are well-known in the literature, we only provide a brief explanation of each here. We explain the F-DRNN with details since our network is a

Algorithm 2.2.1: Training – DRNN with Feedback

Require: E, p_f, p_s
for $e = 1$ to E **do**
  $p_i = \frac{p_f - p_e}{E} e + p_s$
  for $i = 1$ to $number\ of\ batches$ **do**
    Require: u, y: Input and ground truth
    if $i = 1$ **then**
      $z = y$
    end if
    $s \leftarrow N(0, \sigma_s), r \leftarrow \tanh(s)$
    if $number\ of\ layers = 2$ **then**
      $h \leftarrow 0$
    end if
    for $k = 2$ to $batch\ length$ **do**
      $s_k = W_{ss} s_{k-1} + W_{sr} r_{k-1} + W_{si} u_k + W_{sf} z_{k-1} + b_s$
      $r_k = \tanh(s_k)$
      if $layers = 1$ **then**
        $\hat{y}_k = W_{yr} r_k + b_y$
      else if $layers = 2$ **then**
        $h_k = \tanh(W_{hh} h_{k-1} + W_{hr} r_k + b_h)$
        $\hat{y}_k = W_{yh} h_k + b_y$
      end if
      if $|\hat{y}_k| > 1$ **then**
        $\hat{y}_k = \tanh(\hat{y}_k)$
      end if
      Update weights and biases: BPTT
    end for
end for

generalization of the F-DRNN, with all the parameters to be learnable. For more information, please take a look at the main references. We used Pytorch, Keras, Scikit-learn and Python 2.7 for simulations [51], [52], [53].

2.2.2.1. Latent Factor Analysis via Dynamical Systems (LFADS)

Latent Factor Analysis via Dynamical Systems (LFADS) [54] works by modeling a dynamical system that can generate neural data. The algorithm models the nonlinear vector valued function F that can infer firing rates using neural data input. The LFADS system is a generalization of variational auto-encoders that can be used with sequences of data, to model the time-varying aspect of neural signals. We use observed spikes as the input to the encoder RNN. We bin our spikes in 50 ms bins and then separate each center-out task into a separate trial. We use the inferred firing rates that are the result of applying a nonlinearity and affine transformation on the factors output from the generator RNN. A dimensionality of 64 was chosen for the latent variables that are the controller outputs and the factors.

2.2.2.2. FORCE Dynamic Recurrent Neural Network (F-DRNN)

F-DRNN [4] is defined as below:

$$\begin{cases} \tau \frac{ds_t}{dt} = -s_{t-1} + gW_{sr}r_{t-1} + \beta W_{si}u_t + W_{sf}\hat{y}_{t-1} + b_s \\ r_t = \tanh(s_t) \\ \hat{y}_t = W_{yr}r_t + b_y \end{cases} \qquad \text{Equation 2.2.7}$$

$s \in \mathbb{R}^N$ is the activation variable, and $r \in \mathbb{R}^N$ is the vector of corresponding firing rates. These states track the first and zero order differential features of the system, respectively. $W_{sr} \in \mathbb{R}^{N \times N}$ describes the relationship between s and r. $W_{su} \in \mathbb{R}^{N \times I}$ relates s to the input vector u. $\hat{y}$ models the feedback in the network. $W_{sf} \in \mathbb{R}^{N \times M}$ tracks the effect of $\hat{y}$ on s.

2.2.2.3. Deep Neural Network (NN)

Neural Network and its architecture have been explained in section 2.1.3.2. In this work, since over-fitting is possible, which can cause issues where the trained model cannot later generalize to the separate test data, we perform early stopping during validation such that a limited number of epochs (round of training with all inputs) are used for training before the weights are finalized. The following number of epochs are considered in our work: 5, 10, 20, 30, 50, 75, 100, 125, 150, 200, 300, 400, 500, 600. In addition, we consider different network structures with up to 3 layers, where each set consists of 1, 2, or 3 hidden layers with the given number of nodes in each layer: (100), (100, 100), (100, 10), (20, 20), (20, 20, 20), (40, 40), (40, 10), (40, 40, 40), (10, 10, 10).

2.2.2.4. Support Vector Regression (SVR)

Support vector regression (SVR) [55] is the continuous form of support vector machines where the generalized error is minimized, given by the function:

$$\hat{y} = \sum_{i=1}^{N}(\alpha_i^* - \alpha_i)k(u_i, u) + b \qquad \text{Equation 2.2.8}$$

where α_i are Lagrange multipliers and k is a kernel function, where we use the radial basis function kernel in this work. The Lagrange multipliers are found by minimizing a regularized risk function:

$$\frac{1}{2}||w||^2 + C\sum_{i=1}^{l} L_{\epsilon}(y) \qquad \text{Equation 2.2.9}$$

We vary the penalty portion of the error term, C, as part of the validation process to find the optimum parameter.

2.2.2.5. Linear Model (LM)

The linear model [1] uses a standard linear regression model where we can predict kinematics ($\hat{y}$) from the neural data (u) by using:

$$\hat{y} = a + \sum_{i=1}^{N} W_i u_i \qquad \text{Equation 2.2.10}$$

We find the weights W_i and the bias term a through a least squares error optimization to minimize mean squared error between the model's predictions and true values during training. The parameters are then used to predict new kinematics data given neural data.

2.2.2.6. Kernel Auto-Regressive Moving Average (KARMA)

The Kernel Auto-Regressive Moving Average (KARMA) model [56] can also be used for prediction. ARMA (non-kernelized) uses the following model, where $\hat{y}_k^i$ is the i^{th} component of the kinematics data at time step k and u_j^s is the j^{th} component of the neural data at time step s:

$$\hat{y}_k^i = \sum_{l=1}^{r} A_l \hat{y}_{k-1}^i + \sum_{l=1}^{s} B_l u_{k-l+1}^i + e_k^i \qquad \text{Equation 2.2.11}$$

Thus, we are performing a weighted average of the past r time steps of kinematics data and the past s time steps of neural data (as well as the current one) with a residual error term, e. Then, the difference in KARMA is that we use the kernel method to translate data to the radial basis function dimension. We use a standard SVR solver for inference, by just concatenating the different histories for the kinematics and neural data. When training, we use the known kinematics values for the history. However, when predicting new kinematics data, we use old predictions for the history portion of the new predictions.

2.2.2.7. Gated Recurrent Units (GRU)

A simpler version of the RNN cells than LSTM that can extract long term dependencies in sequential data are Gated Recurrent Units (GRU) [57]. The GRU formulation is as below:

$$\begin{cases} z_k = \sigma(W_{zu}u_k + W_{zr}r_{k-1} + b_z) \\ h_k = \sigma(W_{hu}u_k + W_{hr}r_{k-1} + b_h) \\ r_u = tanh(W_{ru}u_k + W_{rr}(h_k \circ r_{k-1}) + b_r) \\ c_u = \tanh(W_{cu}u_k + W_{cr}r_{k-1} + b_c) \\ r_k = (1 - z_k) \circ r_{k-1} + z_k \circ r_u \\ \hat{y}_k = W_{yr}r_k + b_y \end{cases} \qquad \text{Equation 2.2.12}$$

Here, h is a reset gate, and z is an update gate. The reset gate determines how to combine the previous memory and the new input. The network decides how much of the previous memory should be kept by using the update gate. Vanilla RNN is the case that we set the update gate to all 0's and the reset to all 1's.

2.2.2.8. XGBoost (XGB)

XGBoost [7], [32] is one kind of boosting methods which uses ensemble of decision trees. Among 29 competitions winning solutions published at Kaggle during 2015, 17 solutions used XGBoost [32]. For a given data set with $n$examples and m features $D = \{(x_i, y_i)\}$,

$|D| = n, x_i \in \mathbb{R}^m, y_i \in \mathbb{R}$, a tree ensemble model uses K additive functions to predict the output:

$$\hat{y}_i = \rho(x_i) = \sum_{k=1}^{K} f_k(x_i), f \in F \qquad \text{Equation 2.2.13}$$

where $F = \{f(x) = w_{q(x)}\}$, $(q: \mathbb{R}^m \to T, w \in \mathbb{R}^T)$ is the space of regression trees, q represents the structure of each tree, T is the number of leaves, and each $f_k$corresponds to a tree structure q and leaf weights w.

2.2.2.9. Random Forests (RF) and Decision Trees (DT)

Random Forests [58] are one kind of bagging tree based algorithms that make the prediction by routing a feature sample through the tree to the leaf randomly. The training process will be done independently for each tree. The forest final prediction is the average of the predictions of all the trees. Decision trees [59] are a special case of random forests with one tree.

2.2.3. Pre-processing and Feature Engineering

We evaluate the performance of our DRNN on 12 neural features: High-frequency, Mid-frequency, and Low-frequency Wavelet features (HWT, MWT, LWT); High-frequency, Mid-frequency, and Low-frequency Fourier powers (HFT, MFT, LFT); Latent Factor Analysis via Dynamical Systems (LFADS) features [54]; High-Pass and Low-Pass Filtered (HPF, LPF) data; Threshold Crossings (TCs); Multi-Unit Activity (MUA); and combined MWT and TCs (MWT + TCs) (Table 2.2.1).

To extract wavelet features, we use 'db4' mother wavelet on 50ms moving windows of the voltage time series recorded from each channel. Then, the mean of absolute-valued coefficients for each scale is calculated to generate 11 time series for each channel. HWT is formed from the wavelet scales 1 and 2 (effective frequency range ≥ 3.75KHz). MWT is made from the wavelet scales 3 to 6 (234Hz - 3.75KHz). Finally, LWT shows the activity of scales 7 to 11 as the low frequency scales (≤ 234Hz). Fourier-based features are extracted by computing the Fourier transform with the sampling frequency of 30KHz on one-second moving windows for each channel. Then, the band-powers at the same 11 scales of the wavelet features are divided by the total power at the frequency band of 0Hz - 15KHz. To generate TCs, we threshold bandpass-filtered (250Hz - 5KHz) neural data at -4 times the

Table 2.2.1. Frequency Range of Features

Features	**Frequency Range**
HWT, HFT, HPF	> 3.75KHz
TCs, LFADS	250Hz – 5KHz
MWT, MFT, BPF	234 Hz – 3.75KHz
LWT, LFT, LPF	<234Hz

root-mean-square (RMS) of the noise in each channel. We do not sort the action potential waveforms [60]. Threshold crossing events were then binned at 50ms intervals.

LFADS is a generalization of variational auto-encoders that can be used to model time-varying aspect of neural signals. Pandarinath et al. [54] shows that decoding performance improves when using LFADS to infer smoothed and denoised firing rates. We use LFADS to generate LFADS features based on the trial-by-trial threshold crossings from each center-out task.

To extract HPF, MUA, and LPF features, we apply high-pass, band-pass, and low-pass filters to the broadband data, respectively, by using second-order Chebyshev filters with cut-off frequencies of 234Hz and 3.75KHz. To infer MUA features, we calculate RMS of band-pass filter output. Then, we average the output signals to generate one feature per 50ms for each channel. Table 2.2.1 shows the frequency range of features.

We smooth all features with a 1s minjerk smoothing kernel. Afterwards, the kinematics and the features are centered and normalized by the mean and standard deviation of the training data. Then, to select the most informative features for regression, we use XGBoost, which provides a score that indicates how useful each feature is in the construction of its boosted decision trees [7], [32]. In our single-day analysis, we perform Principal Component Analysis (PCA) [61]. Figure 2.2.2 shows the block diagram of our BMI system.

2.2.4. Experimental Results

We conduct our FDA- and IRB-approved study of a BMI with a 32-year-old tetraplegic (C5-C6) human research participant. This participant has Utah electrode arrays (NeuroPort, Blackrock Microsystems, Salt Lake City, UT, USA) implanted in the medial bank of Anterior Intraparietal Sulcus (AIP), and Broadman's Area 5 (BA5). In a center-out task, a cursor moves, in two dimensions on a computer screen, from the center of a computer screen outward to one of eight target points located around a unit circle. A trial is one trajectory of the cursor from the center of the screen to one of the eight targets on a unit circle (Figure 2.2.2). During open-loop training, the participant observes the cursor move under computer control for 3 minutes. We collected open-loop training data from 66 blocks over 43 days for offline analysis of the DRNN. Broadband data were sampled at 30,000 samples/sec from the two implanted electrode arrays (96 channels each). Of the 43 total days, 42 contain 1 to 2 blocks of training data and 1 day contains 6 blocks, with about 50 trials per block. Moreover, these 43 days include 32, 5, 1, and 5 days of 2015, 2016, 2017, and 2018, respectively.

As a pre-processing step before passing the neural data to the decoders, we use XGBoost feature importance score to select stable channels across the training days. The more a feature is used to make key decisions with XGBoost decision trees, the higher its relative importance. This importance is calculated explicitly for each feature in the dataset, allowing features to be ranked and compared to each other. Importance is calculated for a single decision tree by the amount that each feature split point improves the performance measure, weighted by the number of observations the node is responsible for. The importances are then averaged across all the XGBoost decision trees.

Since the predictions and the ground truth should be close in both micro and macro scales, we report root mean square error (RMSE) and R^2 as measures of average point-wise error and the strength of the linear association between the predicted and the ground truth signals, respectively. RMSE is calculated as below:

$$RMSE = \sqrt{\frac{1}{K}\sum_{i=1}^{K}(y_i - \hat{y}_i)^2} \qquad \text{Equation 2.2.14}$$

where K is the total number of data points, y_i and $\hat{y}_i$are the i^{th} ground-truth and prediction, respectively. The smaller the RMSE is, the better the performance. R^2 is also calculated as below:

$$R^2 = \left(\frac{\sum_i (y_i - \bar{y})(\hat{y}_i - \bar{\hat{y}})}{\sqrt{\sum_i (y_i - \bar{y})^2}\sqrt{\sum_i (\hat{y}_i - \bar{\hat{y}})^2}}\right)^2 \qquad \text{Equation 2.2.15}$$

The larger the R^2 is, the better the performance.

Results reported in this section are R^2 values for Y-axis position. For more analysis, we refer the reader to [5]. R^2 values for X-axis position and velocities in X and Y directions and RMSE values for all the kinematics are all presented in supplementary material. All the curves and bar plots are shown by using 95% confidence intervals and standard deviations, respectively.

The available data is split into train and validation sets for parameter tuning. Parameters are computed on the training data and applied to the validation data. We perform 10-fold cross-validation by splitting the training data to 10 sets. Every time, the decoder is trained on 9 sets for different set of parameters and validated on the last set. We find the set of optimum parameters by using random search, as it has shown better performance than grid search [62]. Finally, we test the decoder with optimized parameters on the test set. The performance on all the test sets is averaged to report the overall performance of the models in both single- and multi-day analysis.

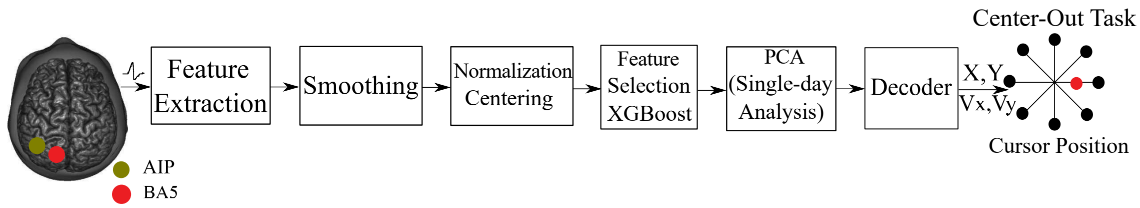


Figure 2.2.2. Architecture of our BMI system. Recorded neural activities of Anterior Intraparietal Sulcus (AIP), and Broadman's Area 5 (BA5) are passed to a feature extractor. After pre-processing and feature selection, the data is passed to the decoder to predict the kinematics in a center-out task.

We compare our DRNN with other decoders, ranging from linear and historical decoders to nonlinear and modern techniques. The linear and historical decoders with which we compare ours are the Linear Model (LM) [1] and Kalman Filter (KF) [41]. The nonlinear and modern techniques with which we also compare ours include Support Vector Regression (SVR) [55], Gaussian KARMA [56], tree based algorithms (e.g., XGBoost (XGB) [7], [32], [33], Random Forest (RF) [58], and Decision Tree (DT) [59]), and neural network based algorithms (e.g., Deep Neural Networks (NN) [42], Recurrent Neural networks with simple recurrent units (RNN) [43], Long-Short Term Memory units (LSTM) [44], Gated Recurrent Units (GRU) [57], and F-DRNN [4]).

We first present single-day performance of DRNN, which is a common practice in the field [4], [41], [63] and is applicable when the training data is limited to a single day. Moreover, there are aspects that differ between single- and multi-day decoding, which have not yet been well characterized (e.g., varying sources of signal instability) and remain challenging in neuroscience. Furthermore, single-day decoding is important before considering multi-day decoding since our implantable hardware will be developed such that the decoder parameters can be updated at any time. Table 2.2.2 summarizes the parameters of different algorithms for single- and multi-day analysis.

2.2.4.1. Single-Day Performance

We select the MWT as the input neural feature. The models are trained on the first 90% of a day and tested on the remaining 10%. Figure 2.2.3 shows the average performance of the

Table 2.2.2. Optimum Parameters for Different Algorithms (Only differences are reported for multi-day)

MODEL	SINGLE-DAY	MULTI-DAY
SVR	C : 0.1, Kernel: RBF	C : 1
KARMA	r : 0, s : 20, C : 0.1, Kernel: Gaussian	r : 0, s : 2, C : 0.1
XGB	number of trees: 15, maximum depth: 8	number of trees: 20
RF	number of trees: 20, maximum depth: 10	number of trees: 40
DT	maximum depth: 10	-
NN	Layer: 2, Optimizer: Adam, Nodes: (40, 10), Batch size: 64, dropout: 0, epoch: 118	Batch size: 128 dropout: 0.25
RNN	Optimizer: RMSprop, Nodes: 25, Batch size: 64 History: 20, dropout: 0.2, epoch: 19	Nodes: 100, Batch size: 128 History: 40, epoch: 50
LSTM	Optimizer: RMSprop, Nodes: 50, Batch size: 64 History: 40, dropout: 0.35, epoch: 17	Nodels 75, Batch size: 128 epoch: 50
GRU	Optimizer: RMSprop, Nodes: 75, Batch size: 32 History: 40, dropout: 0.3, epoch: 18	Batch size: 64
FDRNN	g : 1, β : 0.5, Nodes: 1200, Batch size: 10 σ_b: 0.025, σ_s: 0.01, τ: 250 ms, epoch: 10	g: 0.5, Nodes: 1500
DRNN	Layer: 1, Optimizer: Adam, **Nodes: 5**, p_s: 0.25, p_f: 0 Batch size: 16, **History: 10**, dropout: 0.25, **epoch: 2**	**Nodes: 10**, p_s: 0.5 Batch size: 64, **epoch: 5**

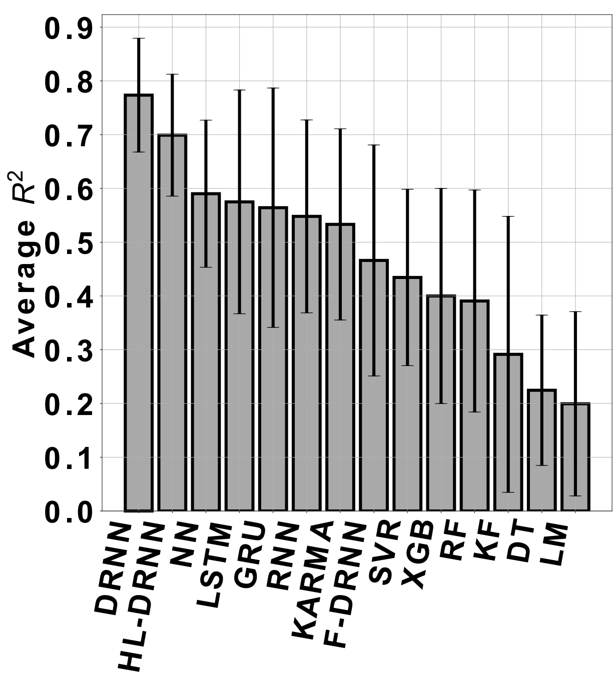


Figure 2.2.3. Average performance of decoders operating on MWT over single-day data.

decoders. History-Less DRNN (HL-DRNN) uses the neural data at time k and kinematics at time $k-1$ to make predictions at time k. As we see, DRNN and HL-DRNN are more stable and have higher average performance compared to other decoding techniques.

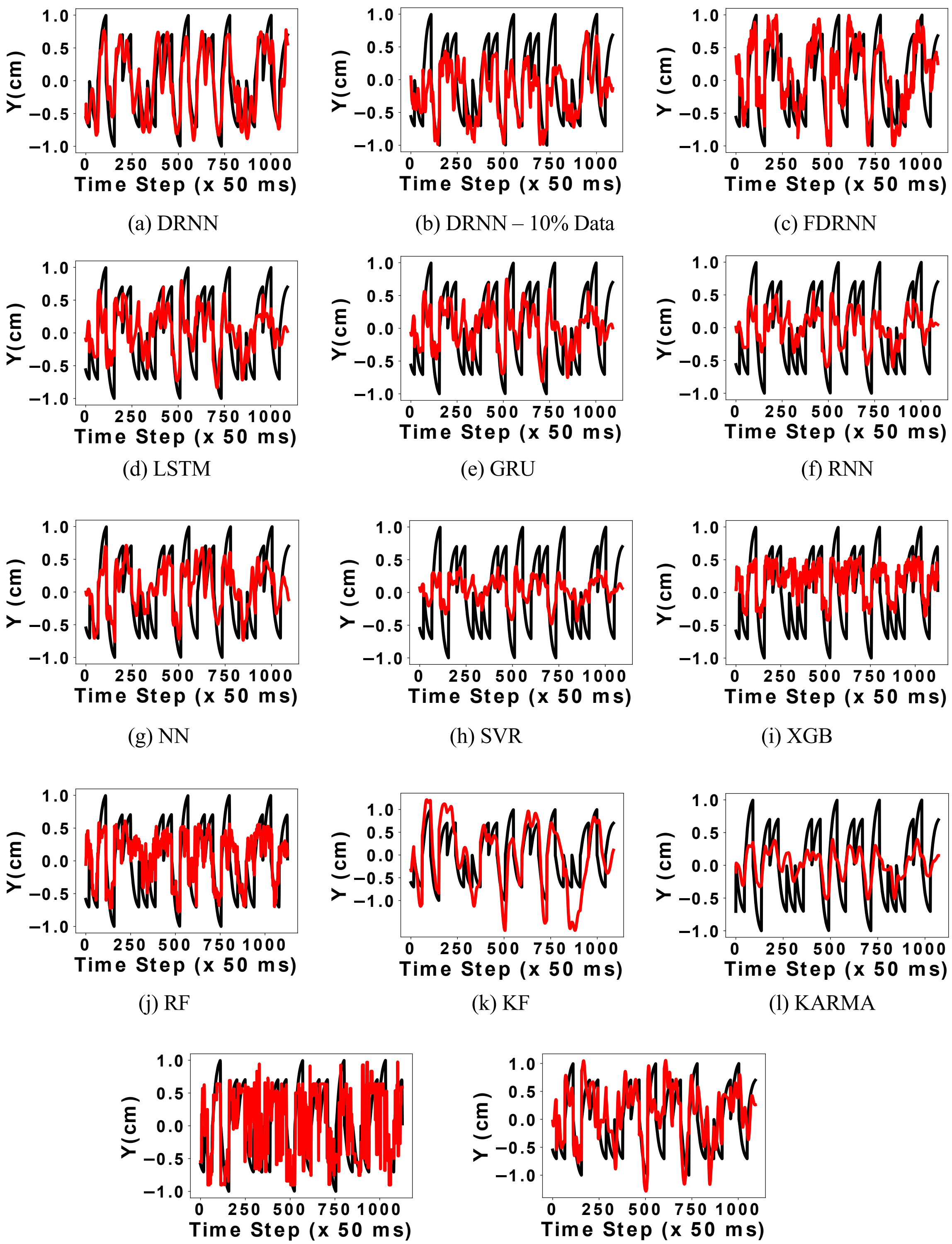
Figure 2.2.4 shows the regression of all the decoders on a sample day. We use only 10% of the single-day training data in Figure 2.2.4 (b) to show the stability of the DRNN to the limited amount of single-day training data. For cross-day analysis, we train the DRNN on a single day and test it on all the other days and repeat this scenario for all the days. Figure 2.2.5 shows the performance of the DRNN over all the days. This figure shows that MWT is a more robust feature across single days.

2.2.4.2. Multi-Day Performance

To evaluate the effect of the selected feature on the stability and performance of the DRNN, we train the DRNN on the data from the first 20 days of 2015 and test it on the consecutive days by using different features. Figure 2.2.6 shows that the DRNN operating on the MWT results in superior performance compared to the other features. Black vertical lines show the year change. We show that the MWT are also the best for a range of decoders in supplementary material.

Then, we evaluate the stability and performance of all the decoders over time. Figure 2.2.7 shows that the overall and the average performance of the DRNN exceeds other decoders. Moreover, the DRNN shows almost stable performance across 3 years. The drop in the performance of almost all the decoders is because of the future neural signal variations [2].

To assess the sensitivity of the decoders to the number of training days, we change the number of training days from 1 to 20 by starting from day 20. Figure 2.2.8 shows that the Deep-DRNN with two layers and the DRNN have higher performance compared to the other decoders, even by using a small number of training days. Moreover, figure 2.2.8 shows that the performance of the DRNN with one layer, 10 nodes, and history of 10 is comparable to the Deep-DRNN with 2 layers, 50 and 25 nodes in the first and second layers, and history of



(a) DRNN (b) DRNN – 10% Data (c) FDRNN

(d) LSTM (e) GRU (f) RNN

(g) NN (h) SVR (i) XGB

(j) RF (k) KF (l) KARMA

(m) DT (m) Linear Model

Figure 2.2.4. Regression of different algorithms on test data from the same day 2018-04-23: true target motion (black) and reconstruction (red).

20. Therefore, a small DRNN with a short history has superior performance compared to the other decoders.

To evaluate the effect of re-training the DRNN, we consider four scenarios. First, we train DRNN on the first 20 days of 2015 and test it on the subsequent days. Second, we re-train a DRNN, which has been trained on 20 days, with the first 5%, 10%, 50%, and 90% of the subsequent test days. Third, we re-train the trained DRNN annually with 5%, 10%, 50%, and 90% of the first days of 2016, 2017, and 2018. Finally, we train DRNN only on the first 5% and 90% of the single test day. Figure 2.2.9 shows a general increase in the performance of the DRNN after the network is re-trained. The differences between the performances of the first three scenarios are small, which means that the DRNN does not necessarily need to be re-trained to perform well over multiple days. However, because of inherent non-stationarity of the recorded neural data over multiple days [2], training the DRNN on the first 90% of the same test day in the last scenario results in the highest average test performance. The DRNN relies on neural data inputs–not just the kinematic feedback or target information–based on the following evidence. First, target information is not explicitly provided to the DRNN. Any target information available to the DRNN is learned from the neural data and/or feedback components. Second, DRNN outputs change substantially based on different feature engineering approaches (Figures 5, 6) and over different trials (with the same features) (Figures 2.2.4, 2.2.10a). Finally, predictions fail when the DRNN uses only feedback (Feedback-Only), feedback with noise substituted for neural data (Feedback-Noise), or feedback with the neural data provided only at the beginning of the trials (Short-Neural) (Figure 2.2.10(b)).

2.2.5. Summary and Future Work

We propose a Deep Multi-State DRNN with feedback and scheduled sampling to better model the nonlinearity between the neural data and kinematics in BMI applications. We show that feeding back the DRNN output recurrently result in better performance/more robust decodes. Feeding the output back to the input recurrently in addition to the input neural data provides more information to the DRNN to make predictions, which results in a smaller

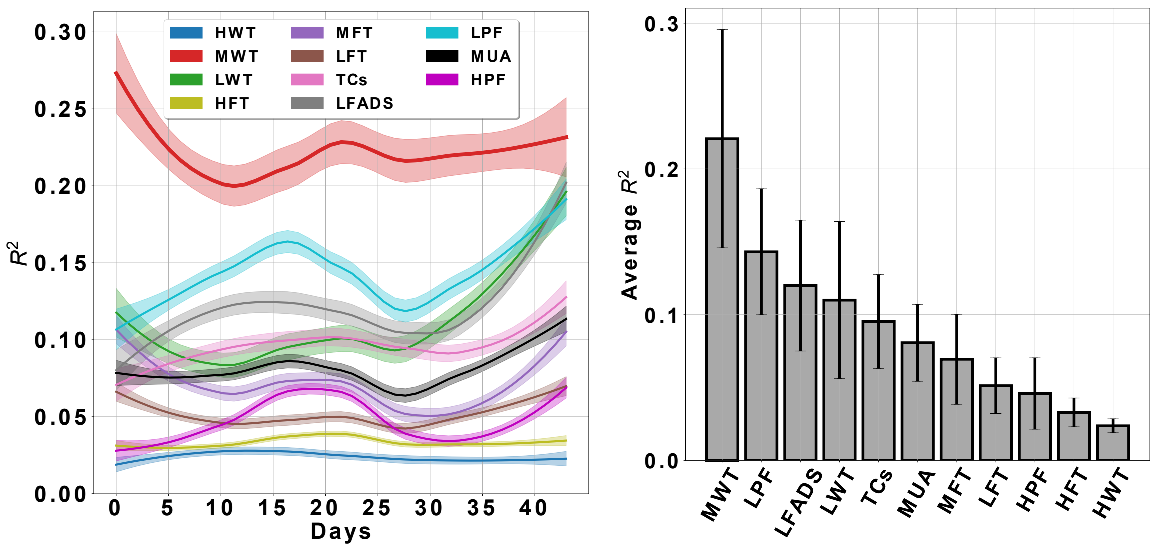


Figure 2.2.5. Cross-day analysis of the DRNN.

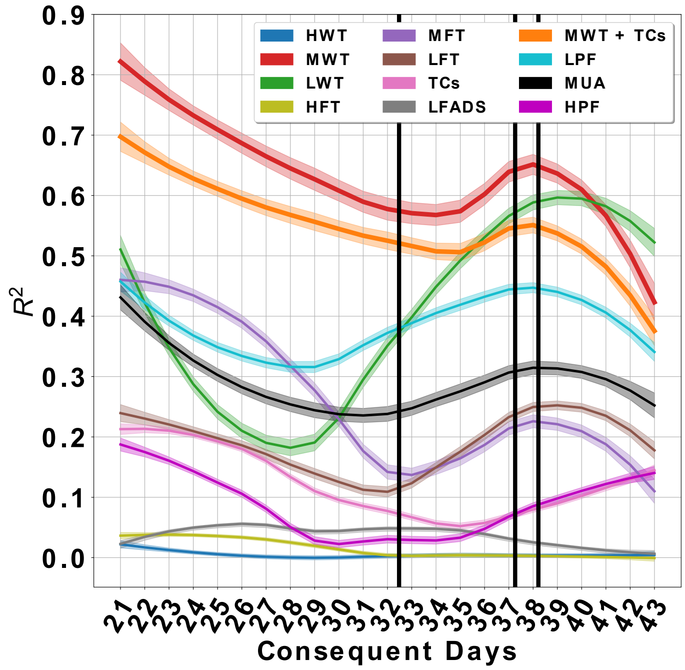


Figure 2.2.6. The DRNN operating on different features.

5

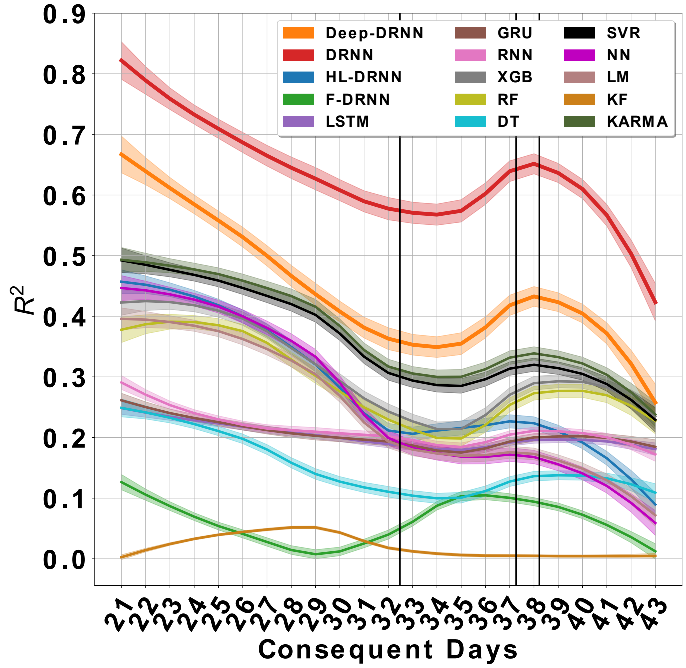


Figure 2.2.7. Multi-dayperformanceofthedecoders.

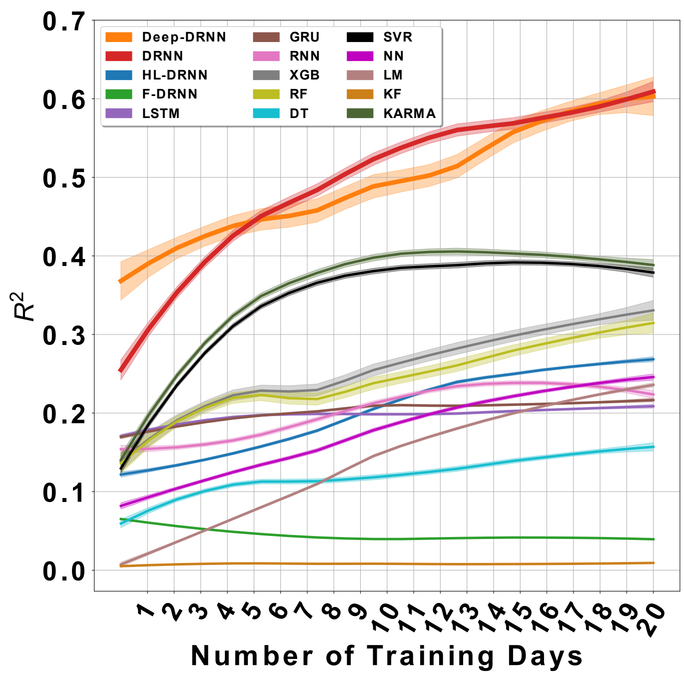


Figure 2.2.8. Effect of number of training days on the performance of the decoders.

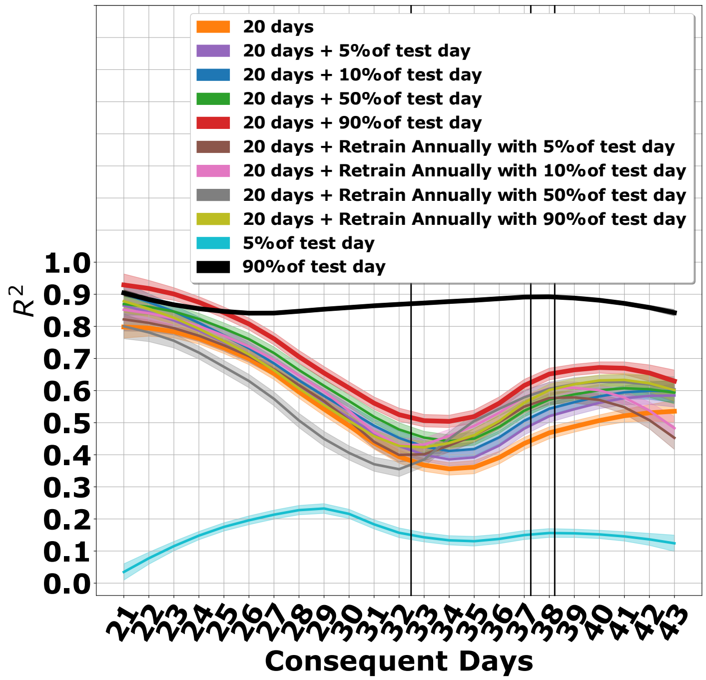
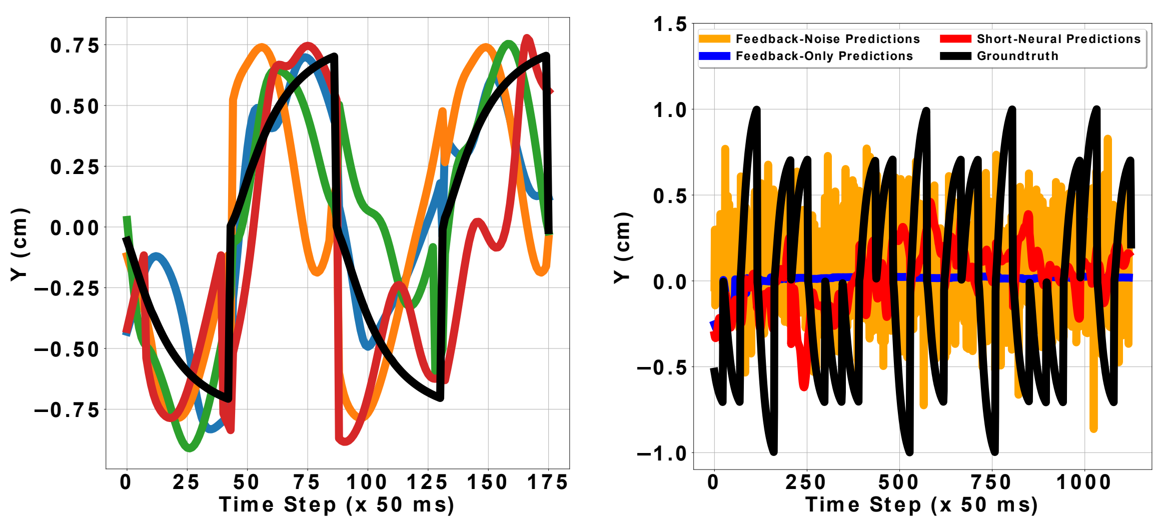


Figure 2.2.9. The DRNN operating in different training scenarios.



(a) (b)

Figure 2.2.10. The DRNN predictions in different scenarios. (a) DRNN predictions for sample targets in all four quadrants, (b) DRNN predictions no/short neural data. True target motion (black) and reconstructions (colored)

network with less history. Analogous to the gain term of the Kalman filter, the DRNN learns the relative importance of the neural data and feedback. Integrating both state and neural information in this way leads to smoother predictions (Figure 2.2.4(a)). In addition, we show that the added internal derivative state enables our DRNN to track first order and more complicated patterns in the data. Our DRNN learns a matrix that establishes a relationship between the past and present derivative states unlike the conventional DRNN. Also, our DRNN, which learns all the model parameters by using back propagation through time (BPTT), is distinct from F-DRNN as the most similar previous model in BMI, which only learns the output weight by using RLS algorithm. Moreover, its application differs from most of the existing decoders that have been applied to motor cortex data of a non-human primate. To the best of our knowledge, we present the first demonstration of applying feedback and scheduled sampling to a DRNN and comparing different learning-based decoders operating on different features to predict kinematics by using open-loop neural data recorded from the PPC area of a human subject in a real BMI setting. Our DRNN has the potential to be applied to the recorded data from other brain areas as a recurrent network.

To evaluate our DRNN, we analyzed single-day, cross-day, and multi-day behavior of the DRNN by extracting 12 different features. Moreover, we compared the performance and robustness of the DRNN with other linear and nonlinear decoders over 43 days. Results indicated that our proposed DRNN, as a nonlinear dynamical model operating on the MWT, is a powerful candidate for a robust BMI.

The focus of this work has been to first evaluate different decoders by using open-loop data since the data presented was recorded from a subject who has completed participation in the clinical trial and has had the electrodes explanted. However, the principles learned from this analysis will be relevant to the future subjects with electrodes in the same cortical area.

BMIs are intended to operate as wireless, implantable systems that require low-power circuits, small physical size, wireless power delivery, and low temperature deltas ($\leq 1^{\circ}$C) [7], [8], [9]. By choosing efficient algorithms that map well to CMOS technologies, Application Specific Integrated Circuit (ASIC) implementations could offer substantial power and mobility benefits. We are proposing our DRNN as a method that will not only have superior performance on single- and multi-day data compared to the other decoding techniques in this work, but can also be optimized for hardware implementation. Since it is impractical to require powerful CPUs and GPUs for everyday usage of a BMI device, we need a device that is easily portable and does not require communication of the complete signals recorded by electrodes to an external computer for computation. Doing the computation in an ASIC would reduce the latency of kinematics inference and eliminate a large power draw for the gigabytes of neural data that must be transferred otherwise. Thus, we plan to create an ASIC that can be implanted in the brain to perform inference of kinematics from neural signals. The main bottleneck in most neural network accelerators is the resources spent on fetching input history and weights from memory to the Multiplication and Accumulation (MAC) unit [64]. The DRNN can potentially help mitigate this issue since it requires fewer nodes and input history compared to the standard recurrent neural networks. This eliminates the need for large input history storage and retrieval, reducing latency and control logic. Furthermore, by using 16-bit fixed point values for the weights and inputs rather than floating point values, we can reduce the power used by the off-chip memory [64], [65].

Future studies will evaluate the DRNN performance in a closed-loop BMI, in which all the decoders use the brain's feedback. Next, since we believe that our small DRNN achieves higher efficiency and uses less memory by reducing the history of the input, number of weights, and therefore memory accesses, we are planning to implement the DRNN in a field-programmable gate array (FPGA) system where we can optimize for speed, area, and power usage. Then, we will build an ASIC of the DRNN for BMI applications. The system implemented must be optimized for real-time processing. The hardware will involve designing multiply-accumulates with localized memory to reduce the power consumption associated with memory fetch and memory store.

[Section 2.3 (pp.65-124) of Chapter 2 is temporarily redacted.]

Chapter 3

HEARTBEAT ARRYTHMIA CLASSIFICATION

Our second area of concentration is heartbeat arrhythmia detection. Electrocardiogram (ECG) plays an important role in clinical practice for monitoring heart health, making accurate detection and classification of arrhythmic heartbeats essential for cardiovascular disease management and prevention. Automation and accuracy are crucial, as manual ECG analysis is time-consuming and susceptible to human errors. To address these challenges, we propose EKGNet, an integrated approach combining analog computing and deep learning to develop a fully analog arrhythmia classification architecture. EKGNet is designed to not only maintains high balanced accuracies with low power consumption but also utilizes the energy efficiency of transistors operating in the subthreshold region. The system design incorporates a novel analog sequential Multiply-Accumulate (MAC) circuit to mitigate process, supply voltage, and temperature variations. EKGNet is modeled as a Bayesian neural network, incorporating analog noise and mismatches into the model, further enhancing the network's performance and generalizability. We employ knowledge distillation technique to transfer knowledge from a teacher network to EKGNet, improving the network's performance. Additionally, we introduce an algorithm for weight fine-tuning after quantization to enhance hardware performance. Our work in arrhythmia detection aims to enhance the accuracy and efficiency of cardiovascular healthcare while addressing the challenges associated with analog circuitry and the need for robust and accurate detection.

3.1. EKGNet: A 10.96μW Fully Analog Neural Network for Intra-Patient Arrhythmia Classification

We present an integrated approach by combining analog computing and deep learning for electrocardiogram (ECG) arrhythmia classification. We propose EKGNet, a hardware-efficient and fully analog arrhythmia classification architecture that achieves high accuracy with low power consumption. The proposed architecture leverages the energy efficiency of transistors operating in the subthreshold region, eliminating the need for analog-to-digital converters (ADC) and static random-access memory (SRAM). The system design includes a novel analog sequential Multiply-Accumulate (MAC) circuit that mitigates process, supply voltage, and temperature variations. Experimental evaluations on PhysionNet's MIT-BIH and PTB Diagnostics datasets demonstrate the effectiveness of the proposed method, achieving an average balanced accuracies of 95% and 94.25% for intra-patient arrhythmia classification and myocardial infarction (MI) classification, respectively. This approach presents a promising avenue for developing low power arrhythmia classification systems with enhanced accuracy and transferability in biomedical applications.

3.2 Overview

The electrocardiogram (ECG) is crucial for monitoring heart health in medical practice [23], [24]. However, accurately detecting and categorizing different waveforms and morphologies in ECG signals is challenging, similar to other time-series data. Moreover, manual analysis is time-consuming and prone to errors. Given the prevalence and potential lethality of irregular heartbeats, achieving accurate and cost-effective diagnosis of arrhythmic heartbeats is crucial for effectively managing and preventing cardiovascular conditions [25], [26].

Deep neural network-based algorithms [10] are commonly used for ECG arrhythmia classification (AC) due to their high accuracy [129]. However, many of the current highly accurate arrhythmia classifiers that rely on neural networks (NN) require a large number of trainable parameters, often ranging from thousands to millions, to achieve their exceptional performance [12], [129], [130], [131], [132], [133]. This poses a significant challenge when implementing these classifiers on hardware, as accommodating such a vast number of parameters becomes impractical. Consequently, existing algorithms are computationally intensive, particularly when compared to biological neural networks that operate with significantly lower energy requirements. As a result, designing low-power NN-AC systems poses significant computational challenges due to the computational demands involved.

Current approaches aim to tackle this either by (1) designing better AC algorithms, (2) better parallelism and scheduling on existing hardware such as graphics processing units (GPUs) or, (3) designing custom hardware. Previous studies [134], [135], [136], [137], [138] that concentrate on patient-specific arrhythmia classification on chip necessitate training neural networks individually for each patient, which significantly limits their potential applications. Moreover, most of the existing hardware development is with respect to digital circuits.

Analog computing in the subthreshold region offers potential energy efficiency improvements, eliminating the need for ADC and SRAM, in contrast to prior research that mainly focused on digital circuit implementations [138], [139]. This is particularly beneficial for ECG classification applications, which often face energy constraints in health monitoring devices [5], [7], [8], [9], [33], [34]. Despite the challenges associated with analog circuits, such as susceptibility to noise and device variation, they can be effectively utilized for inferring neural network algorithms. The presence of inherent system noise in analog circuits can be leveraged to enhance robustness and improve classification accuracy, aligning with the desirable properties of AI algorithms [104], [140], [141].

In this work, we propose EKGNet, a fully analog neural network with low power consumption (10.96μW) that achieves high balanced accuracies of 95% on the MIT-BIH dataset and 94.25% on the PTB dataset for intra-patient arrhythmia classification (Figure 3.1). To address the challenges of analog circuits, we design an integrated approach that combines AI algorithms and hardware design. By modeling the EKGNet as a Bayesian neural network using Bayes by Backprop [28], we incorporate analog noise and mismatches into the EKGNet model [142]. Knowledge distillation [29] is employed to further enhance the network's performance by transferring knowledge from ResNet18 [143] used as a teacher network to the EKGNet. We also propose an algorithm to conduct weight fine-tuning after quantization to improve hardware performance.

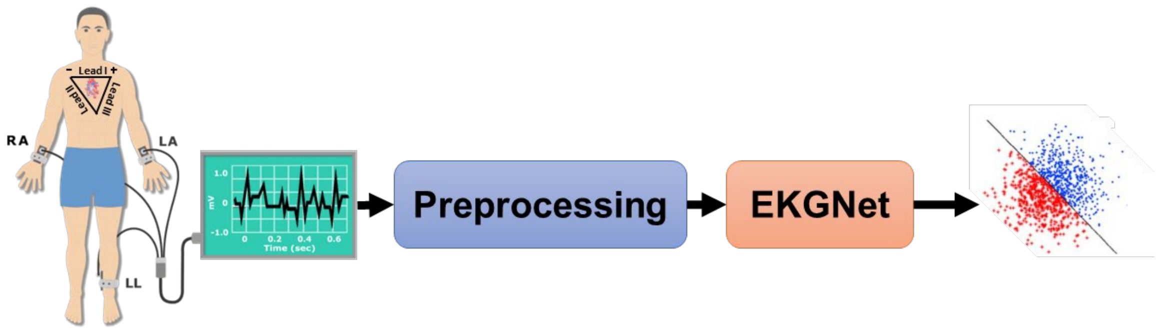


Figure 3.1. EKGNet as a low-power, fully analog neural network for intra-patient arrhythmia classification. The process involves recording the ECG waveform, extracting and preprocessing the beats, and then classifying arrhythmias using EKGNet, achieving high accuracies on the MIT-BIH and PTB datasets.

3.3. Dataset

In this work we utilize two databases; the PhysioNet MIT-BIH Arrhythmia dataset and PTB Diagnostic ECG dataset [144], [145], [146], for labeled ECG records. Specifically, we focused on ECG lead II. The MIT-BIH dataset included ECG recordings from 47 subjects, sampled at 360Hz, with beat annotations by cardiologists. Following the AAMI EC57 standard [147], beats were categorized into four categories based on annotations (Table 3.1). The PTB Diagnostics dataset contained ECG records from 290 subjects, including 148 with myocardial infarction (MI), 52 healthy controls, and other subjects with different diseases. Each record in this dataset consisted of ECG signals from 12 leads, sampled at 1000Hz. Our analysis concentrated on ECG lead II and the MI and healthy control categories.

3.4 Data Preparation

We extract beats from ECG recordings for classification by employing a straightforward and effective method [12]. Our approach avoids signal filtering or processing techniques that rely on specific signal characteristics. The extracted beats are of uniform length, ensuring compatibility with subsequent processing stages (Figure 3.2). The process involves resampling the ECG data to 125Hz, dividing it into 10-second windows, and normalizing the amplitude values between zero and one. We identify local maxima through zero-crossings of the first derivative and determine ECG R-peak candidates using a threshold of 0.9 applied

Table 3.1. AAMI EC57 CATEGORIES.

Class	Annotations
N	Normal, Left/Right bundle branch block, Atrial escape, Nodal escape
S	Atrial premature, Aberrant atrial premature, Nodal premature, Supra-ventricular premature
V	Premature ventricular contraction, Ventricular escape
Q	Paced, Fusion of paced and normal, Unclassifiable

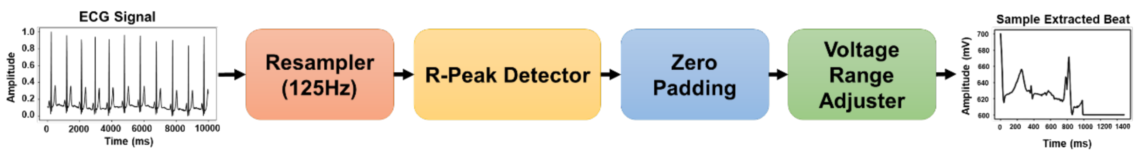


Figure 3.2. The proposed ECG beat extraction method extracts beats without relying on complex signal processing. Beats are standardized through resampling, segmentation, and normalization, with R-peaks identified for uniform analysis. To counter dataset imbalance, specific beats are set aside for testing, and the remaining data is augmented to balance class representation in both training and testing phases for MIT-BIH and PTB datasets.

to the normalized local maxima. The median of the R-R time intervals within the window provides the nominal heartbeat period (T). Each R-peak is associated with a signal segment of 1.2T length, padded with zeros to achieve a fixed length. The inputs are adjusted to fit our hardware input range of 0.6 V to 0.7 V (600 mV to 700 mV).

To address dataset imbalance, we divided the data into training and testing sets. For balanced representation, we excluded a specific number of beats for test: 3200 beats (800 beats per class) for the MIT-BIH and 2911 beats (809 healthy beats and 2102 MI beats) for the PTB dataset. The remaining beats underwent random oversampling [148], resulting in an augmented training dataset with an equal number of beats in each class. We ensured complete separation of training and testing data before augmentation to prevent overfitting. After augmentation, the training dataset consisted of 352,276 beats for the MIT-BIH (88,069 beats per class) and 16,800 beats for the PTB dataset (8,400 beats per class).

3.5. EKGNet Training

To implement the fully analog NN-AC, we optimized the software using a co-design approach. The hardware behavior was emulated in software by extracting a mathematical model of the Multiply-Accumulate (MAC) unit from circuit simulations. EKGNet, a convolutional neural network (CNN), was trained for ECG classification using the constructed ECG training set. During training, Bayes by Backprop [28] was utilized to model the standard deviation of weights (w) as derived hardware input-referred thermal noise ($\sigma = 0.0021090w^2 + 0.0002000w + 0.002355$). The weights and coefficients are expressed in Volts. Hardware leakage noise (~N(0.0005 V, 0.0001 V)) was integrated into the network's output. The training pipeline is depicted in Figure 3.4, and the high-level architecture of EKGNet is shown in Figure 3.3 and Table 3.2. EKGNet consists of two 1-D convolutional layers, two ReLU activations, a max pooling layer, two fully connected layers, and a softmax layer [10]. For optimization, we employed Adam with L_2 regularization weight decay to optimize the cross-entropy loss [105]. Learning rate of $\alpha = 0.003$ was used, which was halved every fifty epochs using a linear scheduler. This approach ensured that the trained weights remained within a small range suitable for implementation and improved linearity due to hardware noise characteristics.

By applying knowledge distillation [29] to further train EKGNet, we observed a performance improvement of 1.5% on MIT-BIH dataset (resulting in 95% test accuracy) and 1.25% on PTB dataset (resulting in 94.25% test accuracy). Knowledge distillation involves transferring knowledge from a larger teacher network (ResNet18) with high test accuracies (99.88% for MIT-BIH and 100% for PTB datasets) to the smaller student network (EKGNet). Through experimentation, we determined that a temperature parameter value of 1.5 yielded optimal results, considering EKGNet's significantly fewer trainable parameters (336) compared to ResNet18 (~11 million).

To balance power consumption and accuracy, we used a 6-bit uniform quantization for the weights. Employing a fine-tuning technique, we iteratively adjusted a single weight by shifting it up or down one quantization level and evaluating its impact on performance

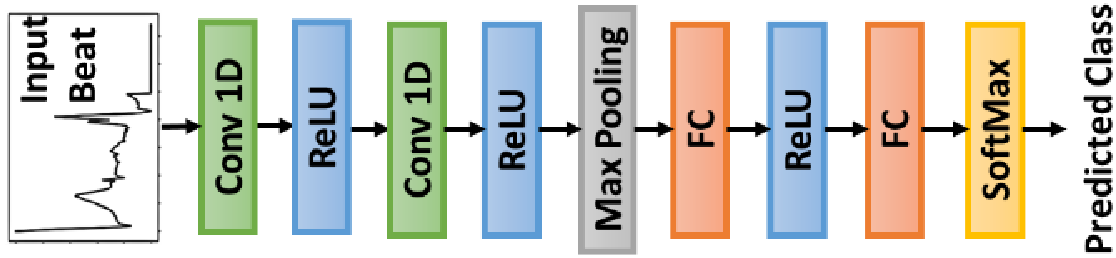


Figure 3.3. EKGNet Architecture. The network comprises two 1D convolutional layers with kernel sizes of 6 and strides of 2, transitioning from 1 to 6 output channels in the first layer and then compressing to 1 output channel in the second. A max pooling layer with a kernel size of 6 and stride of 2 follows, leading into two fully connected (FC) layers that progressively reduce the input size from 18 to 12, and finally to 4, outlining the path from ECG input to arrhythmia classification output.

Table 3.2. EKGNet Architecture

Layer	Parameters
Conv 1D	Kernel Size: 6, Input Channels: 1, Output Channels: 6, Stride: 2
Conv 1D	Kernel Size: 6, Input Channels: 6, Output Channels: 1, Stride: 2
Max Pooling	Kernel Size: 6, Stride: 2
FC	Input Size: 18, Output Size: 12
FC	Input Size: 12, Output Size: 4

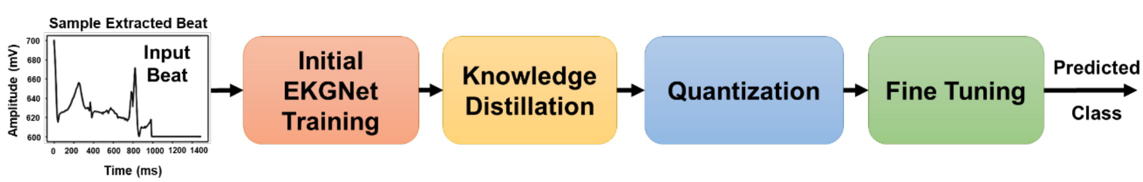


Figure 3.4. EKGNet training and optimization process. Initially, EKGNet is trained using a convolutional neural network (CNN) framework, incorporating Bayes by Backprop to model hardware noise. Following the initial training, knowledge distillation is applied with ResNet18 serving as the teacher network to enhance EKGNet's performance. Subsequently, a 6-bit uniform quantization is applied to the weights for power efficiency. Finally, fine-tuning of the quantized weights is performed (Algorithm 3.1) to further refine accuracy and performance.

Algorithm 3.1 Fine-Tuning of Weights

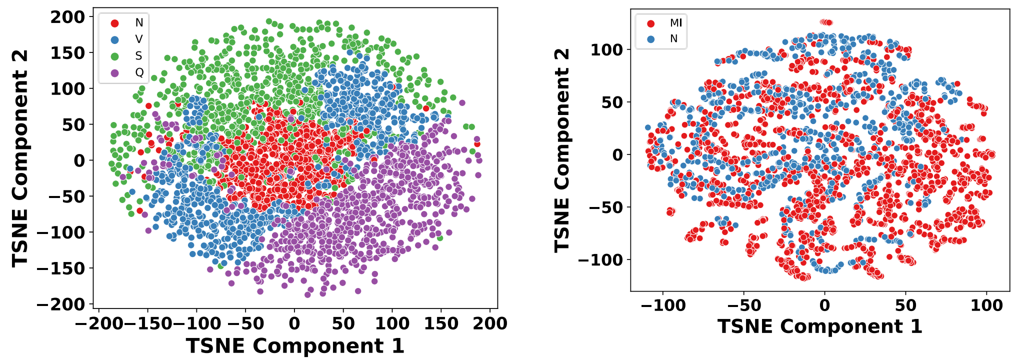
W: Weights, Q: Quantization Indices,
B: Q Levels, E: Number of Iterations

Requires W, Q, B, E
for e = 1 to E **do**
  randomly choose $w \in W$
  randomly set u to Up/Down
  if u = Up **then**
    $w_{new} = B(w_{old}, \mathrm{Q}(w_{old}) + 1)$
  else if u = down **then**
    $w_{new} = B(w_{old}, \mathrm{Q}(w_{old}) - 1)$
  if $acc_{new} < acc_{old}$ **then**
    $w_{new} = w_{old}$

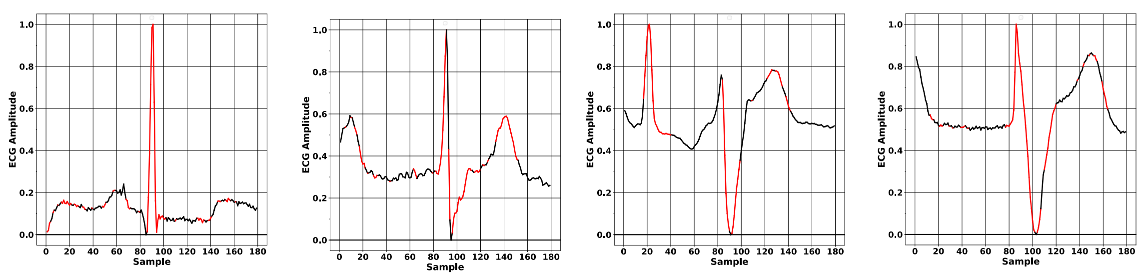
(Algorithm 3.1). With this approach, we achieved the hardware performance of 94.88% and 94.10% on the MIT-BIH and PTB datasets, respectively.

3.6. Model Interpretability

Interpreting machine learning algorithms, especially deep learning, in medical applications is a significant challenge [119]. We utilized t-SNE to visualize the learned representation by mapping high-dimensional vectors of the classified beats to a 2D space [149]. In Figure 3.5(a), we demonstrate clear separability between different classes using MIT-BIH and PTB datasets. Notably, only predicted class labels were used for colorization in the visualizations. To identify regions of input data that receive more attention from EKGNet during prediction, we selected a representative input beat from each category of the MIT-BIH dataset (Figure 3.5(b)). Color-coded visual representations were employed to highlight segments of higher importance in EKGNet's predictions. By calculating the average Shapley value [109] across the entire beat, we selectively colored samples surpassing the threshold. Figure 2(f) illustrates the most typical attribution pattern for ECG classification, aligning with established ECG abnormalities such as ST-segment elevation (STE) and pathological Q waves. However, some model attributions are less conclusive, and the highlighted areas may not perfectly align with clinical significance.



(a)



(N) (S) (V) (Q)

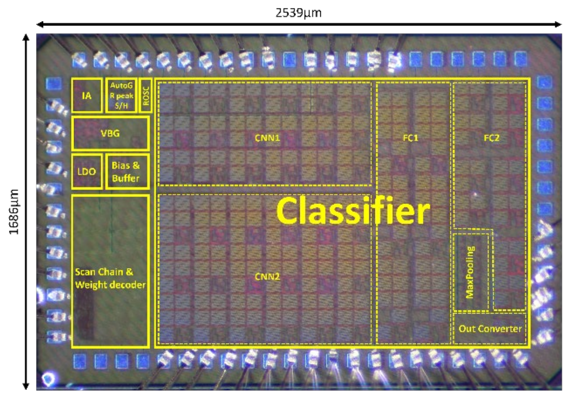
(b)

Figure 3.5. Interpretability Analysis, (a) t-SNE visualization of learned representation for MIT-BIH (left) and PTB (right) classifications. (b) Colored sections highlight important segments in EKGNet predictions.

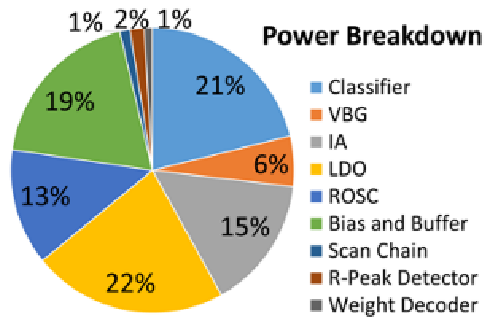
3.7 Hardware Architecture[1]

The proposed hardware architecture includes a fully analog NN-AC and System-on-Chip (SoC) implementation (Figure 3.6). The analog NN-AC, optimized for analog computing, has 336 parameters. Digitally assisted analog circuits are used for ReLU, max pooling, and max functions in the NN-AC (Figure 3.7). The SoC integrates power-on-reset, bandgap voltage reference, biasing hub, oscillator, scan chain, and low dropout regulators (LDO) (Figure 3.7). An LDO with minimal output variations enhances the analog NN-AC's robustness against supply fluctuations. All circuits operate in the subthreshold region with strict duty cycle control for reduced power consumption.

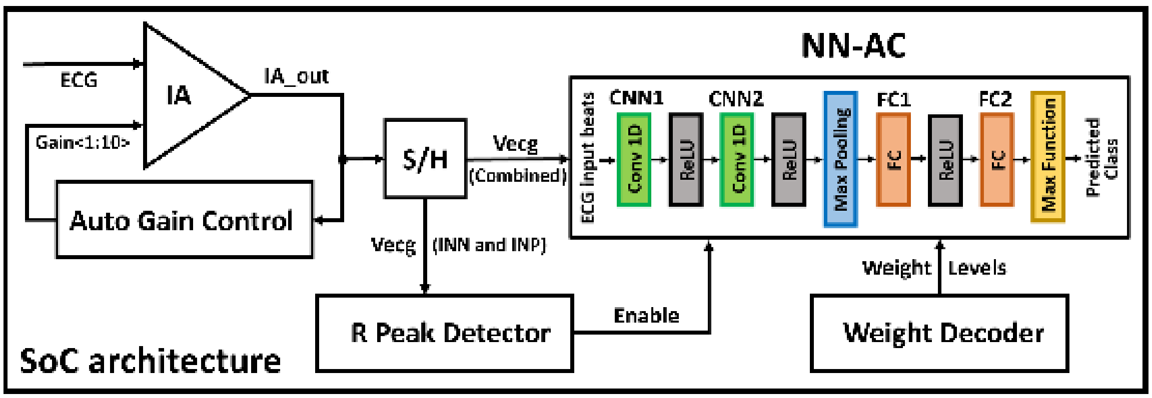
[1] Lin Ma designed and tested the hardware, while the software/hardware co-design was conducted by Benyamin Haghi and Lin Ma.



(a)



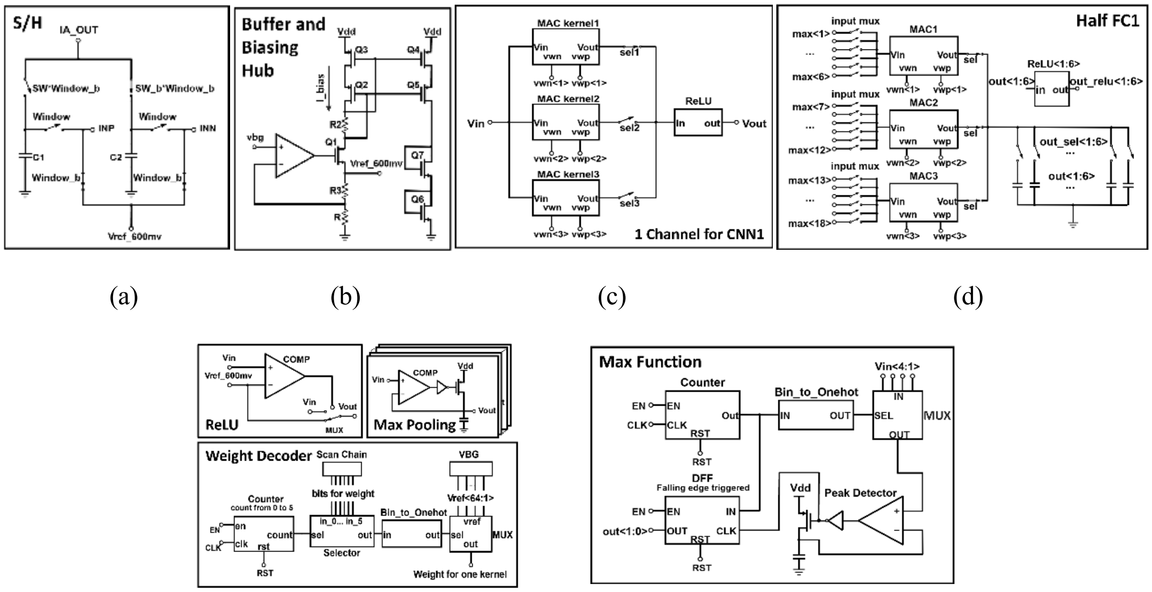
(b)



(c)

Figure 3.6. Analog NN-AC SoC Implementation and Power Efficiency Analysis, (a) Die Micrograph of Analog NN-AC SoC Implementation, showcasing a fully analog NN-AC integrated within a SoC architecture with 336 parameters optimized for analog computing. Includes digitally assisted circuits for ReLU and pooling functions. (b) Power breakdown for SoC modules, highlighting energy-efficient design across MAC units, ReLU and max pooling circuits, and the low dropout regulators ensuring system stability. (c) NN-AC and SoC Architectures detail essential components like power-on-reset, bandgap reference, enhancing operational stability and robustness. The design emphasizes subthreshold operation and utilizes three parallel MAC units for efficient CNN processing, culminating in a 2-bit digital output for arrhythmia classification.

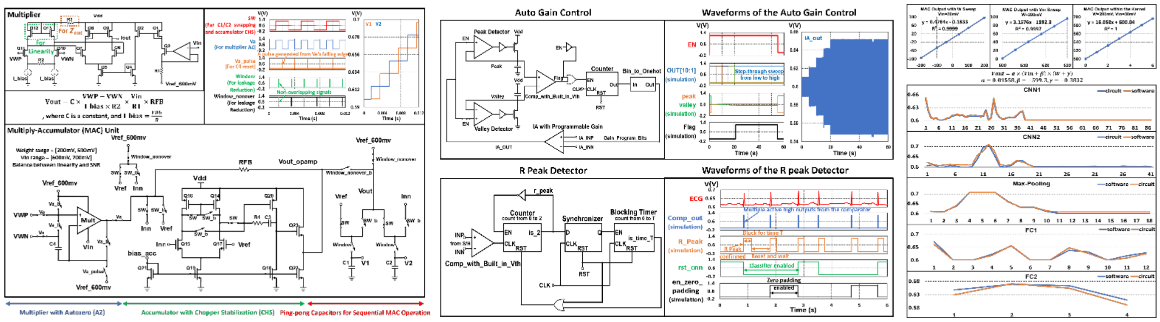
To achieve overlapping CNN operations in hardware, three parallel MAC units are used with a 2-input-sample delay. CNN1 has six channels with ReLU activation (Figure 3.7(c)). CNN2 employs charge redistribution for average pooling across all six channels, followed by ReLU activation. The first half of the fully connected layer (FC1) in Figure 3.6d consists of 18 input



(e) (f)

Figure 3.7. Comprehensive SoC Architecture for Analog NN-AC. Demonstrates the analog NN-AC and SoC implementation featuring (a) sample and hold circuit for accurate ECG signal sampling, (b) buffer and biasing hub ensuring signal integrity, (c) one CNN1 channel with ReLU activation for feature extraction, (d) half of FC1 layer performing MAC operations for data integration, (e) ReLU, Max pooling, and weight decoder modules for nonlinear activation and data summarization, and (f) the max function module for final arrhythmia class determination. This structure facilitates efficient analog computation for arrhythmia classification, balancing precision with low power requirements.

signals undergoing MAC operations in three MAC units (Figure 3.8a). The outputs are combined and sequentially output as six signals. FC2 follows the same design. The max function (Figure 3.7(f)) selects the node with the highest voltage from FC2, producing a 2-bit digital code representing the input ECG's arrhythmia class. The weight decoder synchronizes with NN-AC's control signals to convert digital codes to analog voltage levels. The fully analog NN-AC incorporates inputs from the sample and hold (S/H) (Figure 3.7(a)), enable signals from the R-peak detector (Figure 3.8(b)), and weight levels from the weight decoder (Figure 3.7(e)), generating the 2-bit digital output indicating the ECG's arrhythmia class.



(a) (b) (c)

Figure 3.8. MAC and ECG Enhancements, (a) MAC unit schematic and waveform tracking to minimize process and temperature variation sensitivity. (b) Instrumental Amplifier with automatic gain control and analog R-peak detection, enhancing ECG beat extraction accuracy. (c) Simulation results of MAC unit characterization, demonstrating optimized linearity and efficiency for neural network operations in analog computing.

Figure 3.8 depicts the analog MAC unit. It consists of a multiplier and a current (I_{out}) proportional to their product. To reduce noise and cancel offsets, the multiplier incorporates autozero functionality. Linearity enhancement is achieved through the integration of an inverse hyperbolic tangent circuit. Resistor R3 is included to optimize the multiplier's output impedance, ensuring shift-invariance of the MAC. The accumulator converts I_{out} into a voltage and stores it in the ping-pong capacitors. During each conversion, one capacitor acts as V_{ref}, while the other capacitor stores the updated voltage $V_{ref} + I_{out} \times RBF$. This sequential MAC operation scheme reduces hardware and power requirements compared to parallel operations. The accumulator utilizes chopper stabilization to mitigate offsets and noise, employing switches controlled by narrow window pulses to minimize the leakage effect. The equation in Figure 3.8 shows that the MAC output depends solely on the weight, V_{in}, and device matching.

We propose an analog R-peak detector (Figure 3.8(b)) in the analog domain for beat extraction, specifically identifying the maximum peak of the ECG R wave. Using ECG gradients, the signal is sampled at a rate of 125 samples per second (S/s) with a sample and hold (S/H) circuit employing two ping-pong capacitors to preserve consecutive samples (Figure 3.7(a)). In contrast to previous studies relying on digital R-peak detection, we introduce a digitally assisted analog R-peak detector (Figure 3.8(b)). By exploiting the higher gradient of the R wave in the ECG waveform, we accurately locate R-peaks by comparing the gradient obtained from the S/H with a predefined threshold. To address noise issues, a Schmitt trigger is integrated into the comparator, utilizing two consecutive active high outputs to confirm the presence of an R-peak. Maintaining a constant input amplitude to the NN-AC is essential for achieving an optimal balance between the linearity of the signal. We propose an automatic gain control mechanism (Figure 3.8(b)) to address challenges in the MAC unit and signal-to-noise ratio (SNR). The mechanism includes peak and valley detectors that measure the output amplitude of the instrumental amplifier (IA). A comparator compares the IA output with a target value using a predefined threshold. The IA gain is adjusted systematically from low to high until the comparator changes state, indicating the desired amplitude is achieved. To optimize performance, bias terms are eliminated, and the IA with automatic gain control ensures a consistent output amplitude.

3.8 Experimental Results

The proposed design underwent simulation and fabrication using a 65nm process. Extensive optimization and characterization of MAC linearity were performed through simulations (Figure 3.8(c)). The achieved normalized root mean square errors (NRMSE) for the weights and V_{in} were 0.0036 and 0.0062, respectively. Simulations also confirmed linearity within the kernel, resulting in an NRMSE of 0.0002. This ensures the MAC unit's linearity and shift-invariance, enabling linear operations in the CNN and FC layers. The mathematical model of the MAC, presented in Figure 3.8, along with simulated intermediate signals within the NN-AC, demonstrate waveform similarity to the software implementation with minor errors.

Our NN-AC achieved a measured accuracy of 94.88% and 94.10% on the MIT-BIH and PTB intra-patient classifications, respectively. The power consumption of the proposed NN-AC is 10.96µW at a supply voltage of 1.2V. The overall SoC consumes 67.07µW at a supply voltage of 1.55V. Power consumption breakdown for the SoC is provided in Figure 3.6(b). Additionally, Tables 3.3 and 3.4 summarize the performance of our system, demonstrating

Table 3.3. Comparison of Software-Only Algorithms

MIT-BIH Dataset	Method	Conv. Layers	FC Layers	Parameters	Accuracy (%)
This Work (EKGNet)	**Shallow CNN**	**2**	**2**	**336**	**95.00**
Acharya et al. [130]	Deep CNN	3	3	19,805	94.03
Kachuee et al. [12]	Deep Residual CNN	11	2	98,757	93.40
Yan et al. [133]	Deep CNN	5	3	196,526	92.00
Almahfuz et al. [131]	Deep CNN	13	4	4,391,685	99.90
This Work (Teacher Net)	ResNet18	17	1	11M	99.88
PTB Dataset					
This Work (EKGNet)	**Shallow CNN**	**2**	**2**	**312**	**94.25**
Acharya et al. [130]	Deep CNN	3	3	19,805	93.50
Kachuee et al. [12]	Deep Residual CNN	11	2	98,757	95.90
Kojuri et al. [132]	Deep CNN, Resnet	18, 40	0, 1	145,209; 5,001,842	95.60
This Work (Teacher Net)	ResNet18	17	1	11M	100.00

Table 3.4. Comparison of Hardware Designs

	This Work	**JSSC2014** [134]	**TBCAS2020** [135]	**TCASII2021** [136]	**ISSCC2021** [137]
Process	**65 nm**	90 nm	0.18 µm	0.18 µm	65 nm
Area	**4.28**	4.99	0.93	0.75	1.74
Complete SoC	**Yes**	Yes	No	No	No
Computing Scheme	**Analog**	Digital	Digital	Digital	Digital
Require ADC	**No**	Yes	Yes	Yes	Yes
System VDD (V)	**1.55**	0.5-1 (0.7-1 for SRAM)	N/A	N/A	N/A
Classifier VDD (V)	**1.2**	0.5-1	1.8	1.8	0.75
Test Dataset	**MIT-BIT & PTB**	In-house & MIT-BIH	MIT-BIH	MIT-BIH	MIT-BIH
Class Number	**4**	2	4	5	2/5
Intra-Patient	**Yes**	Yes	No, patient specific	No, patient specific	No, patient specific
Method	**CNN+FC**	MLC/SVM	NN (FC)	NN (FC)	CNN+FC
Accuracy	**94.88%$_{\text{(Arrythmia)}}$ 94.10%$_{\text{(MI)}}$**	95.8%$_{\text{(Arrythmia)}}$ 99%$_{\text{(MI)}}$	99.32%	98%	99.30% $_{\text{(2 class)}}$ 99.16% $_{\text{(5 class)}}$
Accuracy On	**Test Data**	Train* & Test	Train* & Test	Train* & Test	Train* & Test
System Power (µW)	**67.07**	102.2	N/A	N/A	N/A
Classifier Power (µW)	**10.96**	32.8	13.34	1.3	46.8$_{\text{@1MHz}}$ 86.7$_{\text{@2.5MHz}}$
Leakage Power (µW)	**N/A**	N/A	N/A	Not Reported	14.3

* The reported accuracy was higher than anticipated due to the incomplete exclusion of the training data.

lower parameters and power consumption compared to previous software and hardware designs while maintaining comparable accuracy utilizing the intra-patient paradigm.

3.9 Summary and Future Work

We have developed a fully analog CNN-based architecture for accurate arrhythmia classification, using the MIT-BIH and PTB datasets. Our system achieves high accuracy and reduces power consumption by utilizing analog computing, eliminating the requirement for ADC and SRAM. The integration of a novel analog sequential MAC circuit effectively handles PVT variations. Experimental outcomes validate the efficacy of our architecture, offering a low-power solution for accurate arrhythmia classification in wearable ECG sensors.

Chapter 4

ENERGY-EFFICIENT CLASSIFICATION FOR RESOURCE-CONTRAINED BIOMEDICAL APPLICATIONS

After discussing our works in brain-machine interfaces and arrhythmia detection in the previous chapters, we address the critical need for efficient seizure detection in epilepsy management in this chapter. We introduce an approach by employing gradient boosted trees, achieving improved detection performance with significantly reduced energy consumption. This method has the potential to improve seizure detection and allows for customization to meet individual patient needs, enhancing the energy-area-latency product. Highlighting the importance of real-time, resource-efficient solutions for portable or implantable medical devices, our research aims to enhance epilepsy diagnosis and treatment. By incorporating XGBoost, a gradient-boosted framework, our work seeks to contribute to advancements in low-power biomedical applications, underscoring our commitment to developing tailored, energy-efficient seizure detection technologies.

4.1 Energy-Efficient Classification for Resource-Constrained Biomedical Applications

Biomedical applications often require classifiers that are both accurate and cheap to implement. Today, deep neural networks achieve the state-of-the-art accuracy in most learning tasks that involve large data sets of unstructured data. However, the application of deep learning techniques may not be beneficial in problems with limited training sets and computational resources, or under domain-specific test time constraints. Among other algorithms, ensembles of decision trees, particularly the gradient boosted models have recently been very successful in machine learning competitions. Here, we propose an efficient software and hardware architecture to co-design and implement gradient boosted trees in applications under stringent power, area, and delay constraints, such as medical devices. Specifically, we introduce the concepts of asynchronous tree operation and sequential feature extraction to achieve the energy and area efficiency. The proposed architecture is evaluated in automated seizure detection for epilepsy, using 3074 h of intracranial EEG data (iEEG) from 26 patients with 393 seizures. Average F1 scores of 99.23% and 87.86% are achieved for random and block-wise splitting of data into train/test sets, respectively, with an average detection latency of 1.1 s. The proposed classifier is fabricated in a 65-nm TSMC process, consuming 41.2 nJ/class in a total area of 540 × 1850 μm^2. This design improves the state-of-the-art by 27× reduction in energy-area-latency product. Moreover, the proposed gradient-boosting architecture offers the flexibility to accommodate variable tree counts specific to each patient, to trade the predictive accuracy with energy. This patient-specific and energy-quality scalable classifier holds promise for low-power sensor data classification in biomedical applications.

4.2. Overview

The application of machine learning (ML) techniques has been exponentially growing over the past decade [11], with an increasing shift toward mobile, wearable, and implantable devices. ASIC implementation of machine learning models is required to ensure a sufficiently fast response in real-time applications such as deep brain stimulation and vital sign monitoring [150]. Embedded learning at the edge and near the sensors is also critical in applications with limited communication bandwidth or privacy concerns [151]. Furthermore, to meet the tight power budget in portable or implantable devices, it is necessary to embed ML into integrated circuits rather than power-hungry FPGA-based microprocessors [152].

Deep neural networks (DNNs) currently achieve state-of-the-art accuracy in most learning tasks that involve very large datasets of unstructured data (e.g., vision, audio, natural language processing). As a result, there have been significant research and development efforts to design DNN accelerators [151] and specialized ASICs, like Google's TPUs. In the context of hardware-friendly machine learning, a number of methods have been recently explored, such as reducing the bit-width precision [150], [151], sparsity-induced compression, pruning and quantization [151], and mixed-signal MAC implementation [152].

The focus of these methods is on reducing computation, data movement, and storage in neural networks.

However, application of deep learning techniques may not be practical in problems with limited computational resources, or under application-specific prediction time constraints. For instance, a common requirement of diagnostic devices is to minimize power consumption (down to microwatt-range) and battery usage, while maintaining the desired prediction accuracy and low latency. Moreover, without specialized optimization, straight-forward implementation of conventional classification techniques can be computationally intensive, requiring high processing power and large sizes of memory. Indeed, even the simple arithmetic operations performed in conventional classification methods, such as support vector machine (SVM) and k-nearest neighbor (k-NN) algorithms can become very costly with increasing number of sensors, e.g., in multichannel neural implants. Therefore, there is a need to explore alternative methods for severely resource-constrained applications.

Among other algorithms, Gradient Boosted machines, particularly the XGBoost (XGB) implementation has recently been a winning solution in multiple ML competitions (e.g., the intracranial EEG-based seizure detection contest on Kaggle [153]). Here, we propose and optimize ensembles of decision tree classifiers and related circuit level architectures for learning applications under stringent power, area, and delay constraints, such as implantable devices. In particular, we discuss a major application of embedded classifiers in the context of closed-loop neuromodulation devices: automatic seizure detection, and control in medication-resistant epilepsy. However, our techniques are broad enough to impact several other diseases and similar application domains.

With the end of Moore's Law, it is foreseeable that energy-quality (EQ) scalable systems will enable power savings that were previously provided by technology and voltage scaling [154]. EQ scaling may, in some cases, break the traditional VLSI design tradeoffs by simultaneously improving the performance, energy and area [154]. In this work, we leverage hardware-inspired techniques to implement decision tree-based classification algorithms, allowing us to employ various tree parameters as tuning knobs for accuracy, latency, and energy optimization. The resulting classifier significantly improves the power and area efficiency of conventional methods, while achieving a higher classification accuracy and sufficient latency, therefore breaking the strict energy-accuracy tradeoff. The tuning parameters include the number and depth of the trees, number of extracted features, window size, and decision update rate. By appropriate feature engineering and introducing an asynchronous learning scheme, a new class of scalable and low-complexity machine learning hardware for portable sensor-based applications is proposed. Specifically, we analyze the energy and quality scalability of our classifier in terms of hardware-related parameters and diagnostic performance.

This chapter is organized as follows. Section 4.3 presents a review of previous methods that have been used for classification in biomedical domain and describes their hardware cost and scalability challenges. Decision tree-based classifiers and existing hardware architectures are briefly discussed in Section 4.4. The hardware-friendly design of XGB classifier and performance evaluation are presented in Section 4.5 and Section 4.6, respectively. The details of SoC implementation and measurement results are presented in Section 4.7, followed by a discussion on scalability and hardware optimization in Section 4.8. Section 4.9concludes this chapter.

4.3. Embedded Classification in Biomedical Devices

Despite major advances in medicine and drug therapy over the past decade, many disorders remain largely undertreated. Where medications are poorly effective, stimulation may offer an alternative treatment. For example, neurostimulation is today a well-established therapy for essential tremor, Parkinson's diseases, and epilepsy, and has shown promise in migraine and psychiatric disorders. In particular, closed-loop neuromodulation has recently gained attention, e.g., in the form of responsive neurostimulator (RNS) for epilepsy [30], and adaptive deep brain stimulation for Parkinson's disease.

General block diagram of a closed-loop neural interface system is shown in Figure 4.1. Following signal conditioning and feature extraction, an embedded classifier detects the disease-associated abnormalities in real time and triggers a programmable stimulator to suppress symptoms of the disease, e.g., a seizure or tremor, through periodic charge delivery to neurons. A high sensitivity, sufficient specificity, and low detection latency are the key requirements for the on-chip classifier, while maintaining a small footprint and low power.

Epilepsy has been one of the primary targets of neuroengineering research, along with movement disorders, stroke, and paralysis [155]. Abrupt changes in EEG biomarkers usually precede the clinical onset of seizures. Many researchers have therefore focused on extracting epileptic biomarkers for automated seizure detection [31], [156], [157], [158], [159], [160], [161], [162], [163], [164], [165], [166], [167], and closed-loop control through neuromodulation [159], [160], [163].

4.3.1 Prior Work on Machine Learning SoCs

A number of classification algorithms have recently been explored for SoC implementation in diagnostic applications such as seizure detection. An 8-channel linear support vector

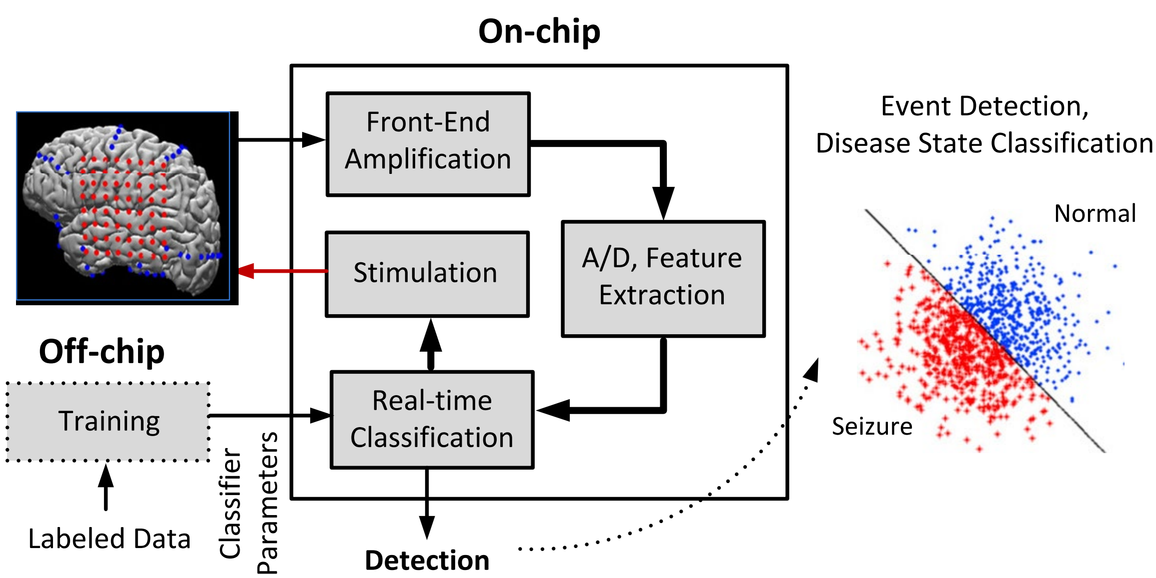


Figure 4.1. General block diagram of a closed-loop system for detection and suppression of abnormal symptoms in a neurological disease. An on-chip classifier is embedded into the implantable device.

machine EEG classifier for seizure detection is presented in [161], using the spectral energy of each EEG channel in seven frequency bins. The Gaussian basis function non-linear SVM combined with time-division multiplexing (TDM) bandpass filters in [162] achieves one of the best energy efficiencies so far (1.83 μJ/class.), a latency of 2s, and a seizure detection rate of 95.1%. Combined with front-end amplifiers and SRAM for data storage, this chip occupies an area of 25mm^2 and supports up to 8 EEG channels.

To avoid the linear growth in memory and utilized hardware with number of channels and frequency bins, a frequency-time division multiplexing approach is employed in [31] and [160], along with a dual-detector classification processor utilizing two linear SVM classifiers. This closed-loop 16-channel SoC integrates a transcranial electrical stimulator, chopping amplifiers and SRAM, occupying a die area of 25mm^2. An 8-channel wireless neural prosthetic SoC is presented in [163] for intracranial EEG-based seizure control, using time-domain entropy and frequency spectrum of individual channels and linear least-square classifier. The entire system dissipates 2.8mW in a total silicon area of 13.47mm^2. A custom processor integrating a CPU with configurable accelerators for SVM classification with various kernel functions is implemented in [164]. Two medical applications including EEG-based seizure and ECG-based arrhythmia detection are demonstrated, while consuming 273μJ and 124μJ per detection, respectively. An error-adaptive boosting classifier is proposed in [165], using decision trees as weak learners. To enable controllable injection of faults, an EEG-based seizure detection system is implemented on FPGA. Dedicated accelerators combined with RISC processors are used in the 16-channel EEG-based SoCs presented in [166] and [168], implementing the fast k-NN algorithm for seizure detection, and SVM for mental status monitoring, respectively. Performance of different classifiers such as k-NN, SVM, naïve Bayes, and Logistic Regression (LR) for EEG-based seizure detection is compared in [167], where LR provides the best F1 score, area, power, and latency. A machine learning-assisted cardiac sensor SoC integrating the maximum likelihood classification (MLC) and SVM is reported in [134] for ECG-based arrhythmia detection.

It should be noted that comparison of accuracy for classifiers that are validated on different datasets or tasks, e.g., those based on EEG vs. intracranial EEG (iEEG), is not pertinent.

While the main focus of our work is on hardware-software co-design and optimization, to evaluate the overall accuracy, we compare the proposed model to other classifiers on a large iEEG dataset [169].

In such biomedical applications, the complexity of classification algorithm, and consequently, the associated power and area, depend on the target (i.e., physician-defined) accuracy and latency for the given diagnostic task. In particular, achieving a latency of <2s and high accuracy with low energy consumption and small area is challenging [162]. To improve the strict energy-area-delay tradeoff and increase the number of channels, we employ a patient-specific prediction model in the form of an ensemble of decision trees, trained by the gradient-boosting algorithm. The main contribution of our work is a hardware-efficient approach that enables energy reduction by minimizing the number of simultaneously extracted features, therefore breaking the energy-area vs. accuracy tradeoff. We implement a low-complexity, yet accurate classification algorithm, that is inherently scalable to multichannel operation, through sharing the computational and memory resources among channels. In contrast to most other classifiers commonly used in literature (e.g., SVM and k-NN) that linearly scale in computational and memory requirements with number of channels and features, our proposed classifier extracts a limited number of features in a sequential fashion, regardless of total channel count. This approach enables significant savings in computational resources and storage on chip. Moreover, we trade accuracy for lower energy, by using the most energy-efficient tree structure for a given patient and a target diagnostic accuracy.

Given the relative complexity of classification algorithms, the commercial devices in existence today, such as the Responsive Neurostimulator (RNS, NeuroPace) [30] for epilepsy, sacrifice the detection accuracy to meet the design constraints such as low power. The battery-powered RNS device in particular, includes three types of detectors: line length (measures the total length of the signal in a given time period), area (detects changes in signal power), and bandpass detectors. Once implanted in the skull, the selected detector by the physician is applied to a maximum of four channels and simple thresholding method is used for seizure detection. However, the detector type should be selected during the programming of device (with line-length being the default detector), which highly limits the sensitivity, specificity, and latency of seizure detection task and may result in suboptimal closed-loop control. Our proposed hardware-friendly classification algorithm would potentially improve the efficacy of current closed-loop stimulation devices such as RNS, by selective computation of features from a higher number of channels. This is achieved through a nonlinear gradient-boosting ML model that can be efficiently integrated on chip with low power.

4.3.2 Hardware Cost

When integrating a classifier on-chip, excessive memory and hardware requirements for feature extraction and machine learning, and the resulting power and area may preclude the ability to process more channels. Power consumption and chip area are mainly determined by the type and number of features, the number of channels monitored, and the type of classifier. The hardware costs associated with feature computation and classification tasks are discussed below.

4.3.2.1 Feature Computation Complexity

Various characteristic features can be extracted from neural data to detect the onset of a particular disease state. A major drawback of common classification methods, with the exception of decision trees, is that they must extract all required features from every input channel to classify the data. Therefore, they require extensive computational resources. Filter banks that are commonly used for spectral power extraction in non-overlapping bands are a key to diagnose neurological disorders and many other signal classification problems, e.g., voice detection, sleep-state classification, irregular heartbeat detection. For instance, to implement the SVM classifier in [164], the band-limited components in eight different bins are extracted from EEG, using FIR filters. The energy of each component is accumulated in a 2s window, and the features from three consecutive windows are combined, resulting in a feature vector with a dimension of $8\times3\times N$, where N corresponds to number of EEG channels.

However, filters are computationally intensive due to MAC operations. Various methods have therefore been explored to reduce the number of multiplications needed or the associated overhead, such as matrix-multiplying ADCs [170], TDM [162], and frequency-time division multiplexing [160].

In contrast to low-frequency EEG-based systems [156], [157], [162], [164], at higher frequencies associated with iEEG where high-frequency oscillations (HFOs) are among relevant biomarkers [171], a larger number of bandpass filters is necessary. Moreover, depending on the application, the use of complex and non-linear features may be inevitable. Selecting a small subset of hardware-friendly features [30], [158], [167] can help to meet the power and area constraints but may sacrifice the classification accuracy. These classifiers also require combinations of serializers, MUX/DEMUX circuits, and buffers to store and process input data and features.

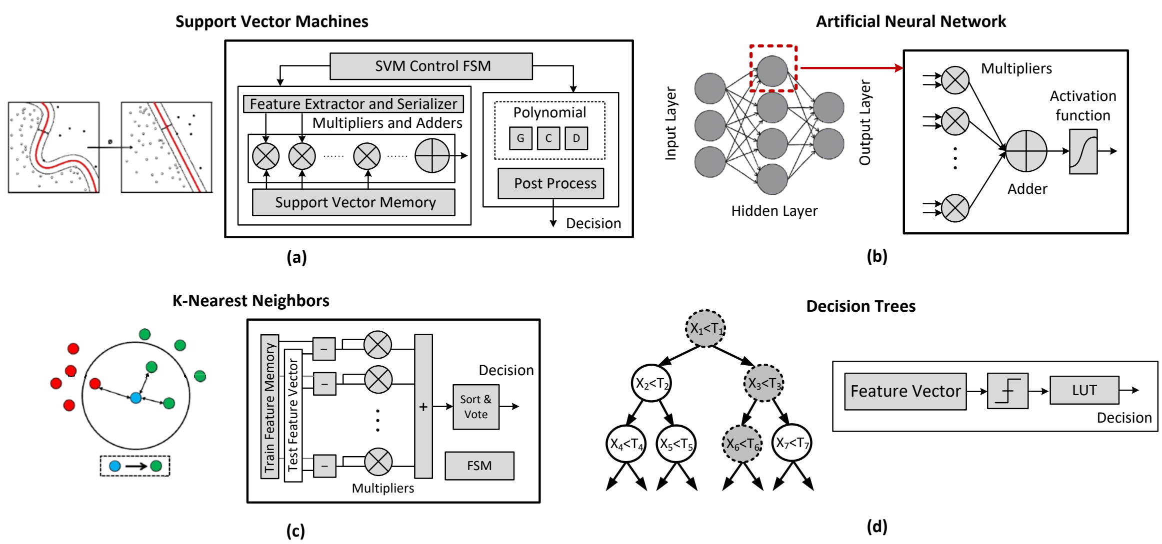


Figure 4.2. Schematic of common learning models as potential candidates for hardware implementation: (a) support vector machines, (b) artificial neural networks, (c) k-nearest neighbors [167], and (d) decision tree-based classifiers.

4.3.2.2 Classification Complexity

Simplified schematic of some of the common classifiers for sensor data classification are shown in Figure 4.2. Neural Networks (NNs) are hardware intensive and typically require high processing power to perform complex computations, as well as large amounts of memory to store many parameters on chip. Furthermore, due to limited access to training sets and patient-specific biomarkers in biomedical applications such as seizure detection (that require extensive monitoring in an invasive setup at the hospital), NN and Deep Learning classifiers would generally result in a poor classification accuracy.

SVM with its intrinsic characteristics such as easy modeling, reproducible results, and robustness through convergence to global minima, has been the most commonly used classifier for epileptic seizure detection from EEG [160]. Three SVM kernels have been applied to on-chip seizure classification: linear, second-order polynomial, and Gaussian SVM (RBF) [160]. The latter achieves better tradeoffs between classification accuracy and latency, with more complex implementation. However, both polynomial and Gaussian SVM require sufficient seizure patterns for training to achieve high accuracy, which is not the case for patients with limited seizure data available [160]. The general classification function of SVM is given by:

$$f(x) = \sum_{i=1}^{N_{sv}} a_i K(\overrightarrow{sv_i}, \overrightarrow{x_i}) + b \qquad \text{Equation 4.1}$$

where $\vec{x}$ is the feature vector, $\overrightarrow{sv_i}$ is one of the N_{sv} support vectors, K is a kernel function, a and b are the modeling parameters. Even though SVM has demonstrated impressive performance in seizure detection from EEG [156], [161], [162], [164], the computational complexity of the decision function in (1) depends on the type of kernel [172]. Generally, a large number of support vectors is required to yield high accuracy in seizure detection, and using a strong classification kernel such as RBF, the energy scales proportionally, dominating by orders of magnitude over feature extraction, front-end, and digitization [164]. While the primary computations for polynomial and linear kernels are dot-product and weighted summation over support vectors, the RBF kernel requires subtract-square accumulation, exponentiation (commonly implemented via CORDIC), and weighted summation over the support vectors [164]. Excluding the nonlinear kernel, the hardware complexity (i.e., number of multiplications and additions) is proportional to $N_{sv} \times d$, where N_{sv} is the number of support vectors and d is the dimensionality of the feature vector [172]. The number of required support vectors depends on separability of the features. A greater number of support vectors is needed for highly nonlinear separation boundary between classes. While more computational resources are available in EEG monitoring systems, the high computational complexity of the RBF kernel makes it unsuitable for implementing in an implantable device that acquires iEEG signals from within the brain (similar to RNS device [30]). The linear SVM would reduce the complexity of the seizure detection algorithm. However, the performance may be degraded if the features are not linearly separable [172].

k-NN classification requires computing the distances between the test and training features, while tracking the k smallest distances. While showing a good performance for epileptic seizure detection [166], the large size of the training set memory and the exhaustive search for nearest neighbors make the classifier power demanding [166]. Moreover, k-NN is more suitable for classification tasks with large sample sizes. In [167], the k-NN classifier achieves a higher F1 measure in seizure detection than the linear SVM, but it consumes dramatically more FPGA resources and power [167].

A simple NN has inputs being multiplied by a weight vector, added together and followed by a linear or nonlinear function to generate the output to the next stage. Logistic regression (similar to a one-layer neural network) uses a linear weighted combination of features and generates the probability of different classes. In general, such methods may not be well suited for efficient hardware implementation due to the complexity involved in feature extraction and classification.

Individual decision trees (DTs) and their ensembles, such as Random Forests and Gradient Boosting, are among the most useful and highly competitive methods in ML, particularly in the regime of limited training data, little training time and little expertise for parameter tuning. Ayinala and Parhi [1] propose a non-linear classifier using AdaBoost technique with decision stumps (trees of depth one) as base classifier, to enable a low-complexity seizure detection system. The relative hardware efficiency of DTs is evident from the fact that simple digital comparators form the main processing unit of a DT, with no need for multiplications, as illustrated in Fig. 4.2 (d). In [173], AdaBoost performs slightly better than SVM with less hardware complexity, achieving a sensitivity of 77.1% (tested on 873 hours of iEEG data) and a false alarm rate of 0.18/hour. The hardware complexity of AdaBoost depends on the required numbers of comparison operations, which is equal to the number of decision stumps (60 in [173], with average feature set size of 14.6). Given their reduced training complexity, DTs are chosen among the various classifiers that have been considered for boosting (e.g., SVMs, NNs) to implement the error-adaptive classifier proposed in [165].

A detailed discussion on hardware implementation of DTs is presented in Section 4.4. Given the variety of hardware schemes used for different arithmetic units in classification and feature extraction, we opted to use a unified metric for evaluating the overall computational complexity of our design and comparing it to prior works, by reporting the number of equivalent 2-input NAND gates. This measure is provided in the SoC comparison table in Section 4.7.

4.3.3 Scalability Challenge in Multi-Sensor Systems

Several studies show that a large number of acquisition channels are required to obtain an accurate representation of brain activity for disease diagnosis or movement decoding, and the therapeutic potential of neural devices is limited at low spatiotemporal resolution [174]. Similar concerns apply to cardiac implants and ECG electrode arrays. Therefore, it is expected that future interfaces integrate hundreds of channels, posing extreme constraints on power dissipation of the circuits. Besides, efficient realization of wearables and IoT devices requires integration of multi-sensor platforms with embedded machine learning techniques and real-time analytics.

Despite substantial research on machine learning, hardware-friendly and scalable implementation is not sufficiently addressed. Even the simple arithmetic operations performed in conventional classification methods can become very costly with increasing number of channels and feature dimensions. For instance, the size of feature vector $\vec{x}$ in equation 4.1 linearly increases with number of channels, and so does the number of multiplications and additions required in a linear SVM. Furthermore, the current method of extracting features separately from each channel requires either a dedicated ADC and feature extraction unit per channel, or power-hungry multiplexing circuits and buffers. Extensive system-level optimizations, specialized hardware techniques, and new design paradigms are needed to meet the energy and accuracy requirements, while preserving the high-channel count recording capability, that has been addressed in this chapter.

4.4 Decision Tree-Based Classifiers

Decision tree (DT) [175] is a popular non-linear ML model where the target class is determined by a sequence of queries, i.e., comparison to a threshold, on input features that start at the root node and terminate in a leaf node, as shown in Fig. 4.2 (d). Compared to NNs, tree-based classifiers are extremely fast in training and classification and require far fewer parameters for tuning. They can be easily parallelized and are robust to label noise. With simple comparators as their building blocks, DTs are naturally a viable solution to reduce complexity [176]. However, the conventional hardware for DTs may not provide optimal results.

In [177], a wearable gait monitor using DTs achieved roughly identical detection accuracy to SVMs, drawing 3× less power. While DTs are commonly implemented in software, there are a few works that implement DTs in hardware. A decision tree spike sorting classifier was reported in [178]. The feature at the active node is multiplexed from a total of four features extracted from the spikes in a neural channel. Badami et al. [179] present an acoustic front-end for speech classification using decision trees. A set of potential features (e.g., band-powers using 8 analog bandpass filters in parallel) are extracted from the input signal, and the feature at each node is multiplexed from this set. The decisions are made by logically combining the outputs of all nodes in a tree, e.g., 7 nodes in Figure 4.2 (d).

4.4.1 Conventional Hardware Architectures

Although the hardware solutions presented in [178] and [179] are suitable for applications with limited number of features and scarce activity (e.g., spike sorting/voice detection where the classifier and feature extractor are only active when a spike/voice is detected), or limited input sources (e.g., voice detection), extending this approach to multi-sensor systems with more features is challenging and can be power-hungry.

As illustrated in Figure 4.3, the direct implementation of DTs requires initial extraction of all features from the input data [178], [179], (Figure 4.3 (a), (b)), or allocation of a separate feature extraction unit to each node, Figure 4.3(c). In problems dealing with multichannel and multi-feature signals, particularly where a combination of trees is required to obtain a higher accuracy, the utilized hardware by each tree must be minimized. For example, assuming a 100-channel neural recording array and a set of 10 features per channel (typical for seizure detection), the first two architectures would require initial processing of a thousand features, the associated memory, and multiplexing circuits. Yet only a small portion of these features are employed in the classification task, that is the sum of visited nodes in all trees ($\leq$ maximum depth $\times$ number of trees). Similarly, the third method would require 7 feature extraction and multiplexing units per tree, as depicted in Figure 4.3(c). Since a maximum of one node at each level of the tree is visited, we previously proposed to utilize one feature extraction unit per level [176], to reduce the required hardware resources compared to Figure 4.3(c).

To support multichannel operation, the alternative approach of placing a tree per channel would require the allocation of a separate DT hardware to each channel. However, in case of disease detection, it is likely that only a small subset of channels capture the abnormal activity, e.g., the electrodes placed in seizure foci. Therefore, training a classifier on the entire array rather than separately classifying every single channel would avoid the unnecessary extraction of features from silent channels. In summary, while DTs offer significant advantages to other classifiers by avoiding multiplication and using fewer memory units, the existing hardware is not well-suited for high-channel-count and resource-limited applications.

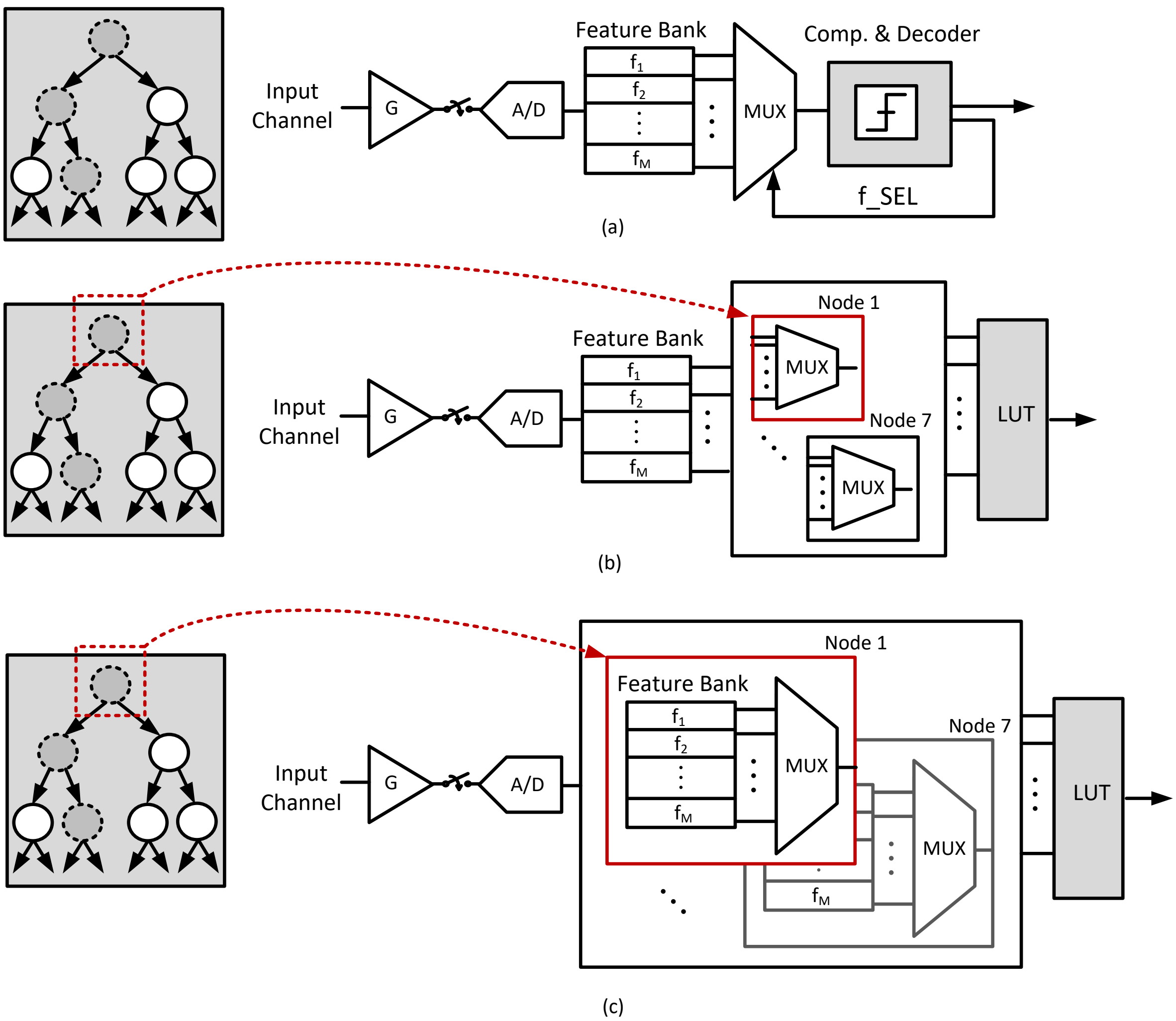


Figure 4.3. Block diagram of conventional DT architectures for a single input channel.

4.5 Hardware-Friendly XGB Classifier Design

Here, we propose a hardware-efficient online classification algorithm using an ensemble of gradient-boosted decision trees, as illustrated in Figure 4.4. Essentially during a classification task by a decision tree, only one path from the root to the leaf is visited. Therefore, unlike other classifiers, only a limited number of features are necessary in practice to make a decision. These features, however, are carefully selected by employing powerful training algorithms that produce the optimal tree structure to maximize the overall predictive accuracy. The trained prediction model, which is the output from the gradient-boosting algorithm, includes full information on tree structures in the ensemble such as thresholds, leaf values, and selected features (shown as Serial Control IN in Figure 4.4, where CH_i and FC_i represent the channel number in the array and feature number, respectively).

The intuition behind our hardware architecture is the following. Since the decision of each tree is made upon completing a series of successive comparisons, a single feature extraction module (and the preceding ADC) can be sequentially used to exclusively calculate the requested feature at the current node. The split direction and next active node are determined by comparing this feature with the corresponding threshold. Therefore, at each step, only the selected channel is used for online feature extraction, without buffering the data from other channels or extracting unnecessary features. As shown in Figure 4.4, the final answer is the sum of answers of all trees (details are discussed below).

In our proposed architecture (Figure 4.4), an ensemble of up to eight gradient-boosted decision trees, each with a fully programmable Feature Extraction Engine (FEE) including FIR filters continuously process the input channels. In a closed-loop architecture, the FEE reuses a single filter structure to execute the top-down flow of the decision tree, where FIR filter coefficients are multiplexed from a shared memory. This approach results in significant hardware saving, compared to the methods shown in Figure 4.3. A potential drawback of this serial processing approach would be the degraded latency, that is carefully studied in this Section.

A comparison of hardware complexity for various DT architectures (assuming a single tree) is summarized in Table 4.1, where N, M, and l represent the channel count, maximum number of nodes, and depth of a tree, respectively. The proposed architecture enables the lowest number of FEEs and classification hardware, and therefore, the lowest complexity. The number of FEE modules (or number of computed features) linearly increase with number of channels in the first two methods. Although our proposed architecture reduces

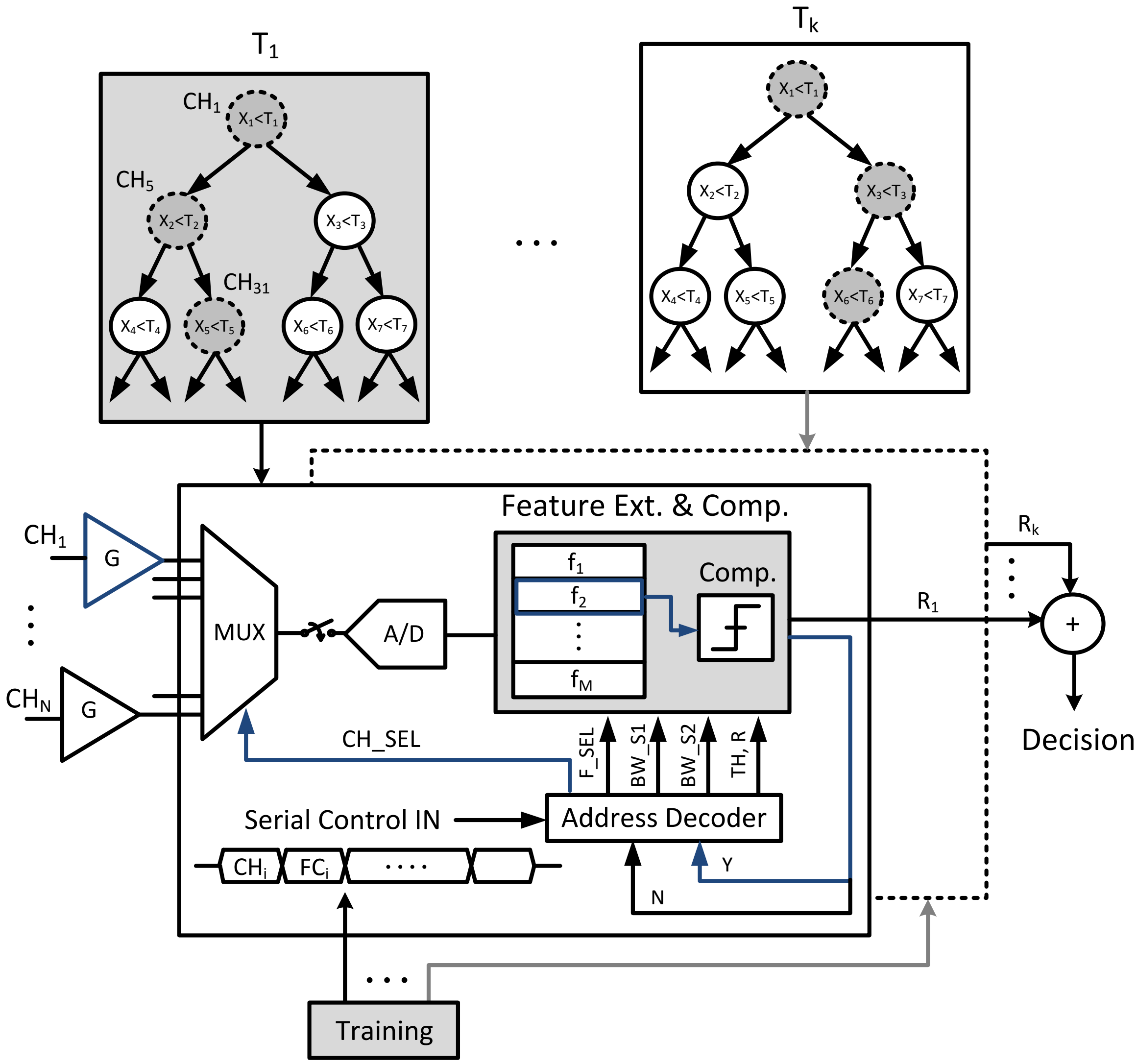


Figure 4.4. Proposed hardware architecture for an ensemble of gradient boosted decision trees.

Table 4.1. Hardware Complexity of DT Architectures

Architecture	# of FEE	# of Comparator	# of MUX
Fig. 4.3 (a)	N	N	1
Fig. 4.3 (b)	N	M^*	M
Fig. 4.3 (c)	M	M^*	M
[176]	l	l	l
This Work (XGB-HW)	1	1	1

*Additional LUT is needed to generate the final decision.

the number of feature extraction and classification (i.e., comparator and multiplexer) units, the memory needed to store the tree structure and coefficient values remains the same in all architectures in Table 4.1. The detailed memory breakdown of our proposed scheme is further discussed in this chapter.

4.5.1 Gradient Boosted Trees

Gradient-boosting [180] is one of the most successful machine learning techniques that exploits gradient-based optimization and boosting, by adaptively combining many simple models to get an improved predictive performance. Binary split DTs are commonly used as the "weak" learners. Boosted trees are at the core of state-of-the-art solutions in a variety of learning domains, given their excellent accuracy and fast operation. For example, among the 29 challenge winning solutions published on Kaggle in 2015, 17 used XGB, where DNN was the second most popular method, used in 11 solutions [181].

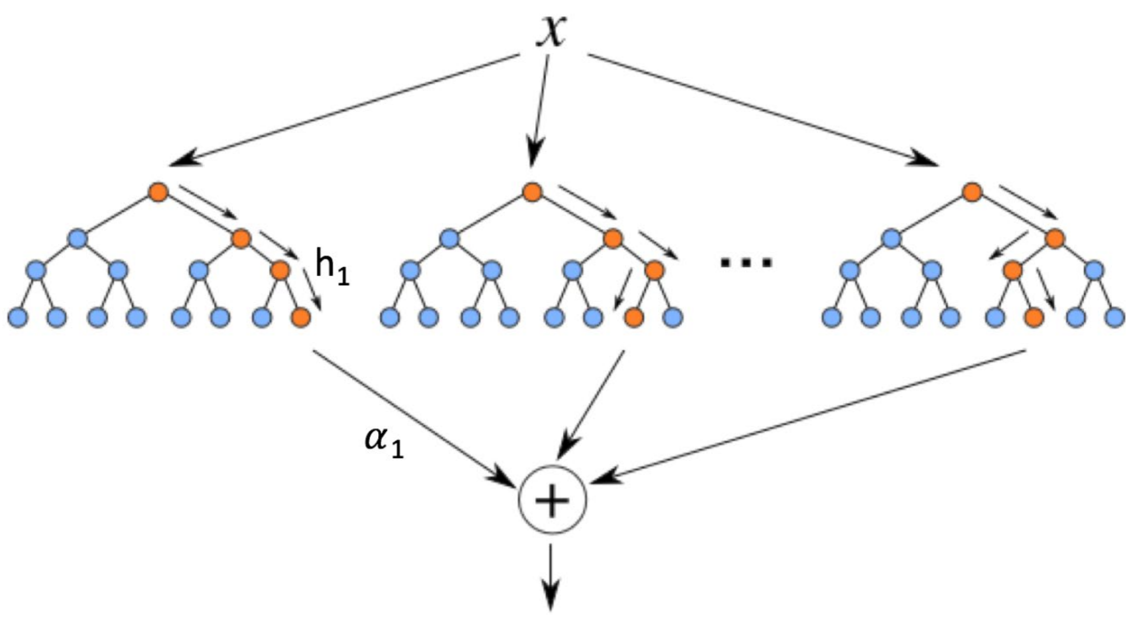


Figure 4.5. Schematic diagram of a boosted ensemble of decision trees.

Boosting involves creating a number of hypotheses $h_t(x)$ and combining them to form a more accurate composite hypothesis. The output of a boosted classifier (or regressor) with an input feature vector of x has the additive form of:

$$H(x) = \sum_t \alpha_t h_t(x) \quad \text{Equation 4.2}$$

where α_t indicates the extent of weight that should be given to $h_t(x)$. A general schematic diagram illustrating an ensemble of depth-3 trees is shown in Figure 4.5. Using gradient-boosting, the trees are built in a greedy fashion to minimize a regularized objective on the training loss [181].

In this chapter, we have employed the XGBoost package [181], a parallelized implementation of the gradient boosting algorithm. To assess the performance of proposed classifier on a relatively large dataset, epilepsy is chosen as our case study, given the availability of continuous recordings from many patients. This architecture, however, can potentially benefit many other on-chip sensor signal classification problems. Applying XGB to our iEEG dataset, we observed over 100 times improvement in training speed compared to common SVM implementations.

In the proposed hardware (Figure 4.4), given that only one channel is used at each feature computation step in a tree, the rest of input channels can be switched off to save power. For example, to classify a 100-channel neural data with 8 trees, only 8 channels are simultaneously active. In contrast to SVM and other methods that require all features from the entire array, this approach significantly reduces the memory and hardware overhead. To reduce energy, a minimum number of trees that obtain a sufficient accuracy are used, that is chosen upon training. Moreover, as a significant advantage, only one tunable bandpass filter can be used to extract as many band-power features as needed, since these features are not computed in parallel. By employing a programmable FIR (or tunable analog) filter, the corresponding coefficients (or band selection parameters) can be easily multiplexed from memory, according to the feature being processed, as shown in Figure 4.4. Besides, as shown later in this chapter, very little improvement in performance is achieved by using trees with a depth of 4 and above. Therefore, these ensembles can be made by a relatively small number of low-depth trees, resulting in significantly lower computational complexity than conventional models, as later confirmed in our comparison table in Section 4.7.

4.5.2 Delay Constraint

The proposed architecture faces a practical challenge of designing decision trees under application-specific delay constraints. Given any ensemble $T = \{T_1, T_2, \dots, T_k\}$ of decision trees obtained from our original method, we need to ensure that each tree T_i satisfies the delay constraint:

$$\sum_{i \in \pi(h)} d_i \leq \Delta T \qquad \text{Equation 4.3}$$

Algorithm 4.1 A greedy training algorithm to meet the delay constraint

Input: Original trained tree ensemble $T = \{T_1, T_2, \dots, T_k\}$
Output: Delay-constrained ensemble $T' = \left\{T_1^{'}, T_2^{'}, \dots, T_k^{'}\right\}$
Data: Training set: $S = \{(x_i, y_i)\}$
Feature set: $F = \{f_i\}$, each with delay d_i
Delay tolerance: ΔT
Set of predecessors of node h: $\pi(h)$

for $all\ trees\ T_i\ in\ T$ **do**
    for $each\ node\ h \in \{1, 2, \dots, |T_i|\}$ **do**
        if $\sum_{i \in \pi(h)} d_i > \Delta T$ **then**
            $\forall f_i \in F$ find feasible f that obtains the best $SplitCriterion(f_i, S)$
            Label node h with f
            Grow $Subtree(h)$
        end
    end
end

where d_i is the time required to compute feature f_i, ΔT is the maximum tolerable detection delay, and $\pi(h)$ is the set of all predecessors of node h. One possibility is using a "greedy" algorithm to solve this practical constraint by building trees that satisfy the delay requirement, as depicted in Algorithm 4.1. However, this algorithm may result in a suboptimal solution since the split criterion and subsequent feature selection is subject to the hard constraint on delay.

4.5.3 Asynchronous Tree Operation

To solve this issue, we introduce an asynchronous approach where trees freely run in parallel, each with features that maximize the accuracy, regardless of their computational delay. Using the averaged results of completed trees and previous results of incomplete trees, decisions are frequently updated to avoid long latencies.

4.5.3.1 Decision-Making Procedure

First, we need to select an optimum time to update the decision of the system. Suppose that we have k trees represented by T_i, $i \in \{1, 2, \dots, k\}$. Assuming that t_i is the total time associated with the longest path in T_i, we select the optimum update time as:

$$t_{opt} = \min\{t_1, t_2, \dots, t_k\} \qquad \text{Equation 4.4}$$

This guarantees that at least one tree will be completed in this interval, and a new decision is made every t_{opt}. Then, we calculate the average value of decisions for each tree:

$$D_{T_i} = \frac{1}{N_i}\sum_{j=1}^{N_i} r_j \qquad \text{Equation 4.5}$$

where N_i is the number of completed cycles over t_{opt} and r_1, r_2, …, r_{N_i} are the corresponding results (i.e., leaf values) of T_i. In a boosting classifier, the answers of all trees must be summed up to make the final decision. Positive answers are classified as seizure and negative ones as non-seizure. The final result of the system is therefore updated as below:

$$D_{final} = \sum_{i=1}^{k} D_{T_i} \qquad \text{Equation 4.6}$$

In case there is no new answer for tree T_i after t_{opt}, we simply use its previous decision. By employing this approach and assuming an initial setup time, there always happens to be at least one result produced during t_{opt} to make a decision. In the proposed asynchronous architecture, each tree continues to test the input data, without waiting for other trees to complete. Suppose that x is a test input that moves through the tree. As x enters node i, it takes time d_i to calculate the feature f_i. Based on the value of f_i, a split to either right or left branch is made, and the process continues until a leaf is reached. By effectively averaging the decisions of fast trees over multiple cycles, while allowing the longer trees to complete, we show that the overall performance of this online asynchronous approach is even superior to the conventional offline method [176], where features at different nodes are simultaneously extracted over the same window and decisions are made at the end of this window (a hardware-intensive solution). Since it is likely that more than one answer would be provided by t_{opt}, averaging can reduce the impact of noisy decisions. Moreover, features are extracted from successive parts of the decision window, rather than one feature for the entire window. Therefore, the decisions are more accurate, while the optimum selection of update time in (4.3) reduces the detection latency.

4.6 Performance Evaluation

As a benchmark, we consider a boosted ensemble of 8 trees with a maximum depth of 4 using proposed model (XGB-HW), and compare it to the linear, cubic, and RBF SVM, k-NN with 3 and 5 neighbors, Logistic Regression, offline XGB (abbreviated as XGB) [176], Random Forest and Extra Tree classifiers, both with 8 trees and a maximum depth of 4. A hyperparameter tuning of classifier parameters was performed to find optimum settings.

4.6.1 Data Description

In this work, we use the publicly available data from the intracranial EEG portal [169]. Continuous recordings from 26 patients sampled at either 500Hz or 5kHz are included in our study. The seizure events are marked by physicians, and patients have been recorded at varying channel counts (ranging from 16 to 128). The access IDs of analyzed patients and further details are provided in Table 4.2. Overall, we studied a total of 3074 hours of iEEG including 393 seizures.

4.6.2 Train/Test Split

A common problem in performance evaluation of real-time classifiers such as seizure detectors is to randomly partition the entire data into train and test samples. Shuffling provides prior information from parts of test data (that should remain unseen) during training, resulting in data leakage. We use a block-wise splitting approach to avoid this problem and fairly assess the performance of our classifier for practical test conditions such as seizure detection. In the block-wise method shown in Figure 4.6, we divide the continuous iEEG data into seizure and non-seizure segments, where each seizure is concatenated with the following non-seizure segment into a larger "block" (the first non-seizure segment is added to the beginning of first block). Thus, each block is comprised of a complete seizure attached to the following non-seizure segment. Most patients in our dataset have sufficient and long enough seizure data to allow this approach. However, cases with small number of short seizures are not good candidates for block-wise selection. Therefore, we removed two patients from our initial dataset.

For the purpose of feature extraction during training and offline testing, we divide the time series into 1s windows and extract all features from channels for each window. We compare our block-wise method with the commonly used random split, in which a 5-fold cross-validation is applied to the shuffled data, followed by a hyperparameter tuning to maximize the F1 score for all classifiers. To tune the parameters for the block-wise approach, we apply a block-wise 5-fold cross validation. In this case, 20% of blocks (rounded up to the nearest integer) are retained for testing the model, and the remaining are used as training set. The cross-validation process is then repeated for 5 times and the results are averaged to produce a single estimation. For patients with less than 5 seizures, we opted for a block-wise leave-one-out approach, where we use one block as test and the remaining blocks as train and repeat this for all blocks. To evaluate the corresponding F1 score, sensitivity, and specificity, we use the tuned parameters for each patient and average the results of cross validation tests as described above. For XGB-HW, the trained prediction model generated by the gradient-boosting algorithm includes all the information on tree structures such as leaf values,

Table 4.2. Patient Data and Signal Acquisition Info

Subj.	iEEG Portal ID	No. Elec.	No. Seiz.	Rec. Dur.	Samp. Rate
1	Study 004-2	56	3	7d 18h	500
2	Study 006	56	5	1d 14h	500
3	Study 017	16	9	7d 17h	500
4	Study 011	88	3	3d 12h	500
5	Study 022	56	7	3d 23h	500
6	Study 023	88	4	2d 5h	500
7	Study 012-1	60	6	3d 7h	500
8	Study 027	48	6	3d 21h	500
9	Study 016	64	7	5d 21h	500
10	Study 031	116	5	6d 19h	500
11	Study 030	64	8	5d 23h	500
12	Study 020	56	8	5d 0h	500
13	Study 014	104	15	6d 0h	500
14	Study 021	108	13	6d 11h	500
15	Study 026	96	22	3d 3h	500
16	Study 024	88	19	8d 10h	500
17	Study 028	96	9	1d 16h	500
18	Study 038	88	10	3d 0h	500
19	Study 005	16	151	6d 16h	500
20	I001_P034_D01	47	16	1d 8h	5k
21	Study 040	116	6	2d 23h	5k
22	Study 036	96	4	4d 14h	5k
23	Study 019	96	36	5d 16h	500
24	Study 033	128	17	6d 17h	500
25	Study 029	64	3	5d 1h	500
26	Study 037	80	8	8d 23h	500

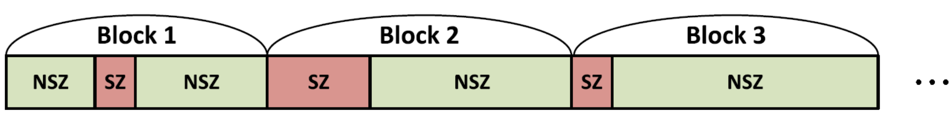


Figure 4.6. The proposed block-wise data partitioning, where SZ and NSZ represent the seizure and non-seizure segments, respectively.

thresholds and selected features. Using this trained model, the online XGB classifier is tested according to the procedure described in Section 4.6.3. To minimize the update interval and latency, features are extracted over smaller time windows than 1s.

4.6.3 Feature Extraction

Prior works [182], [183], [184], [185], [186] have extensively analyzed the optimal features for seizure onset detection. For instance, line-length achieves the best seizure detection performance among more than 65 different time and frequency-domain features in [182]. This time-domain feature is a measure of line-length between successive samples and provides an appropriate characteristic of epileptiform iEEG, since it increases at both low-amplitude fast and high-amplitude slow activities, that normally occur prior to a seizure [184]. Another frequently used feature is the energy of the signal, as a measure of signal power over time. It was firstly shown in [183] and later by several investigators [184], [185], [186] that the power and variance of EEG/iEEG signals are increased minutes prior to seizure onset. In addition, many studies on EEG signals have been focused on spectral power features in the range of below 30Hz (i.e., the Berger bands) [156], [161], [182]. However, the iEEG signals span a wider frequency range and go beyond 200Hz for seizure biomarker extraction [171]. These high-frequency oscillations (HFOs) have been previously studied by many researchers [171], [187]. The authors of [187] have concluded a significant potential of HFOs for seizure detection from iEEG.

Table 4.3. Evaluated Features

Feature	Description
Line-Length (LLN)	$\frac{1}{d}\sum_{d}\lvert x[n]-x[n-1]\rvert$, d = window length
Power (POW)	Total spectral power
Variance (VAR)	$\frac{1}{d}\sum_{d}(x[n]-\mu)^2$ where $\mu=\frac{1}{d}\sum_{d}(x[n])$
Delta (δ)	Spectral power in 1-4Hz
Theta (θ)	Spectral power in 4-8Hz
Alpha (α)	Spectral power in 8-13Hz
Beta (β)	Spectral power in 13-30Hz
Low-Gamma (γ_1)	Spectral power in 30-50Hz
Gamma (γ_2)	Spectral power in 50-80Hz
High-Gamma (γ_3)	Spectral power in 80-150Hz
Ripple	Spectral power in 150-250Hz
Fast Ripple (FR)	Spectral power in 250-600Hz (@ SR = 5kHz)

Based on our initial study on discriminative performance of several frequency and time domain features [176], and the existing literature [182], [183], [184], [185], [186], we chose the following set of features: line-length, total power, time-domain variance, and power in multiple frequency bands, as listed in Table 4.3. We previously analyzed the discriminative performance of this feature set on an extensive iEEG database [176], in which line-length was the best discriminative feature. While the optimal frequency range was patient-dependent, in majority of patients sampled at a sufficiently high rate (5k), it had a clear shift from low-frequency bands toward gamma, ripple, and fast ripples.

Rather than using the absolute value of spectral power [176], normalized features were calculated by dividing the spectral power within each frequency band by the total power. The power values (and corresponding thresholds) typically change with the daily life status of a patient, such as sleep state, physical or mental activities, and consciousness level [188]. In contrast, normalized values are more robust with respect to fluctuations in a patient's daily life and have been utilized in our study. Features are obtained from each iEEG channel using 1s windows for training and offline testing. During online testing, we assign a minimum extraction time to each feature, based on their computational delay. Using normalized band powers, we observed an improved seizure detection accuracy compared to absolute spectral power features used in [176].

It should be noted that various other features may be included to enable more accurate seizure detection. However, the focus of this work is on the classification algorithm. The literature pertaining to analysis of various features for epilepsy diagnosis is immense, and can be found in [182], [183], [184], [185], [186].

4.6.4 Depth and Number of Trees

Decision trees are very efficient, but also susceptible to overfitting in problems with high feature-space dimensionality. To address this, we limit the number of nodes in each tree, i.e., design shallow trees using small number of features [176]. Shorter trees are also more efficient in hardware and incur less detection delay. Figure 4.7 shows the area under the curve

(AUC) performance of an ensemble of gradient-boosted trees versus the number of trees for different values of depth parameter. An important observation is that the detection accuracy is not significantly improved (< 0.5%) with depth values of 4 and higher. Besides, an AUC higher than 90% is achieved using fewer than 10 trees of depth 3 or 4. Therefore, the total energy can be minimized by limiting the number of trees and depth, which are chosen as 8 and 4 in our study.

4.6.5 Performance and Comparison

The average performance of classifiers across patients are shown in Figure 4.8(a) and (b), using block-wise and random splitting methods, respectively. As mentioned before, due to correlation of iEEG waveforms, random splitting can allow the model to learn from parts of test data and statistics of unseen seizures during training. Therefore, it creates overly

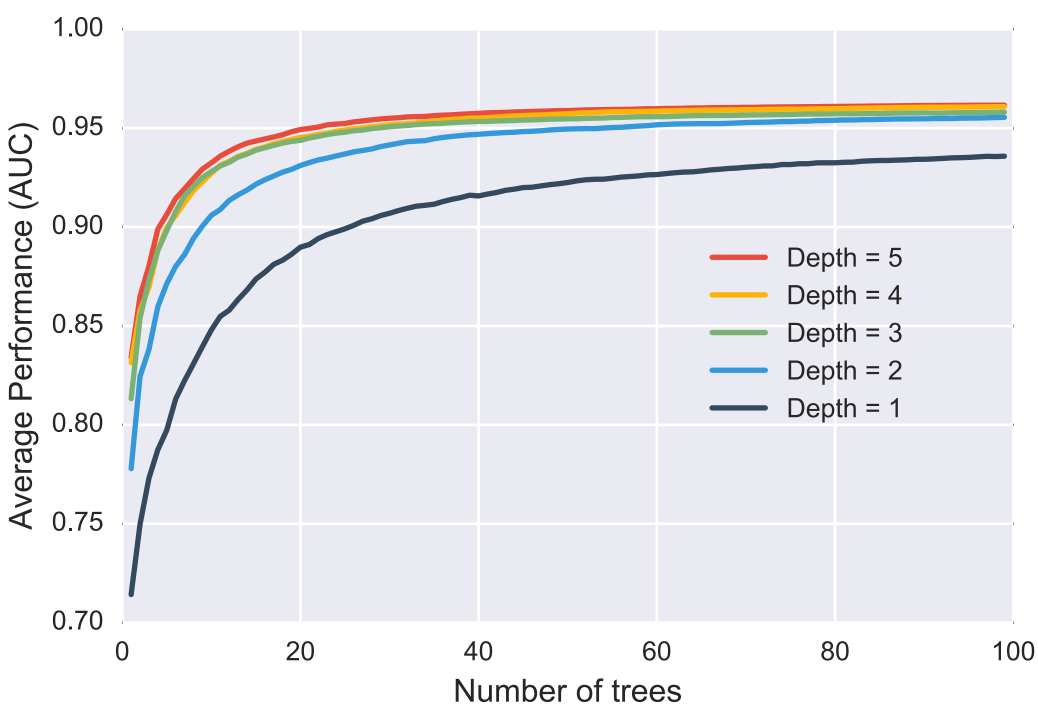
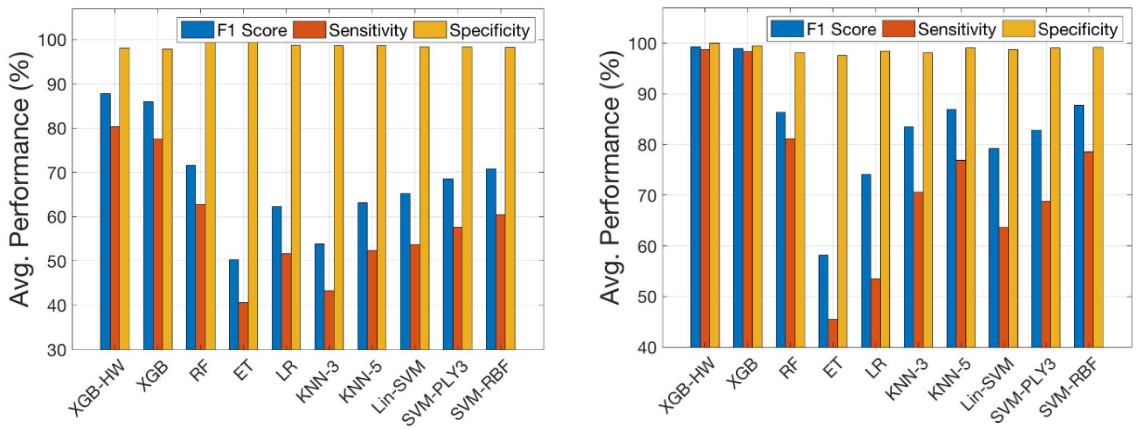


Figure 4.7. The overall classification performance at various depths versus number of trees.



(a) **(b)**

Figure 4.8. Comparison of average predictive ability (F1 score), sensitivity, and specificity of different classification methods among patients, using (a) blockwise, and (b) random splitting methods, respectively.

optimistic predictive models and invalidates the estimated performance. In this work, we consider block-wise approach to alleviate the leakage problem. The F1 score is calculated by counting the number of correctly classified windows, given by:

$$F_1 = \frac{2}{\frac{1}{Sensitivity}+\frac{1}{Specificity}} \quad \text{Equation 4.7}$$

where sensitivity and specificity represent the true positive and true negative rates, respectively. The asynchronous XGB (XGB-HW) performs best among all classifiers, reaching an average F1 score of 99.23% and 87.86%, for the random and block-wise splitting methods, respectively, with an average block-wise sensitivity of 80.33% and specificity of 98.12%.

This is achieved by efficient design of the learning algorithm in an asynchronous online fashion, while minimizing the hardware resources and energy. As expected, random split leads to higher, but unrealistic predictive accuracy. Interestingly, only tree-based methods, in particular, the XGB could classify patient 21’s seizures (87% F1 score), while all other classifiers failed for this patient. Random forests generally require a large number of trees to obtain a high performance, which is not suitable for on-chip implementation. Our results indicate that the proposed asynchronous gradient-boosting method with as low as eight trees, has a higher generalization ability on this iEEG dataset, compared to methods such as k-NN, LR, and SVM. The performance could be further boosted by artifact removal, as some datasets (e.g., patient 13) are contaminated by high-frequency artifacts that particularly overlap with FR band. To evaluate the detection latency, we count the number of correctly classified ictal windows at the beginning of a seizure, and wait for at least three consecutive seizure decisions to remove the effect of transient noises. Figure 4.9 shows the latency among patients, with an average of 1.1s.

4.6.6 Feature Importance

Figure 4.10 summarizes the overall performance of examined features across patients. Line-length stands out as the best feature, in accordance with many other studies [182]. Variance, ripple, and fast ripple are next. Interestingly, we observe a clear shift in discriminative performance of spectral power features from Berger bands toward gamma, ripple, and fast ripples (all normalized). However, as explained in [156] and [161], to distinguish between seizure and non-seizure data, both dominant and less dominant frequency components are required, as well as the spatial variation among channels, that is achieved through a multichannel analysis. In this work, we implement a programmable filter with flexible bandwidth settings to cover all seizure-related frequency components. By using a single filter architecture with programmable bandwidth, the hardware complexity of FEE is significantly reduced compared to prior works that integrate multiple parallel bandpass filters.

4.7 SoC Implementation[2]

Figure 4.11(a) shows the block diagram of the implemented SoC based on the asynchronous XGB classifier presented in Section 5 [9], [33]. This classifier supports up to 32 neural channels. One fully programmable feature extraction unit is used per tree and controlled by

---

[2] Milad Taghavi and Mahsa Shoaran designed and tested the hardware, while the software/hardware co-design was conducted by Benyamin Haghi, Milad Taghavi, and Mahsa Shoaran.

the Tree Control Unit (TCU) to extract epileptic biomarkers. A Mealy FSM implementation of the closed-loop system is chosen, that substantially reduces the power and area overhead. To extract spectral density features, a single FIR filter structure is used, and its coefficients are multiplexed according to the feature being processed, thus reducing the total area. As a result, the classifier achieves an energy efficiency of 41.2nJ/class in a small area of 1mm^2.

Features of line-length, variance, and total power are implemented with standard digital logic according to their mathematical definitions in Table 4.3 and contribute to a small portion of feature extraction area (<15%), as shown in Figure 4.11(b). The main blocks of the implemented Mealy FSM include the ensemble of 8 DTs with programmable FIR filters, a Memory Control Unit (MCU), and an Asynchronous Tree Reset Control (ATRC). The detailed functional description of these blocks is discussed as follows.

4.7.1 DT Ensemble

The ensemble includes 8 decision tree structures with a maximum depth of 4 (15 nodes). For each tree, TCU sets the next state's memory pointer according to the current state, comparator status, and internal flags. A multiplexer selects one channel from the 32-channel input data, according to the current state. This channel is then fed to FEE. At the last processing node of each tree, TCU sends out the 'tree-end' flag as well as final node info to ATRC. Epileptic features are computed in the FEE module. A decoder activates/deactivates its sub-modules according to the feature under study at the current node.

4.7.2 Programmable FIR Filters

To calculate spectral power features, a cascade of two FIR stages is implemented. The first stage decimates input samples, while the second stage provides bandpass filtering. Each stage may be bypassed according to selected feature. Since at each node of a tree only one feature is being processed, a single filter structure with programmable coefficients can be used. This

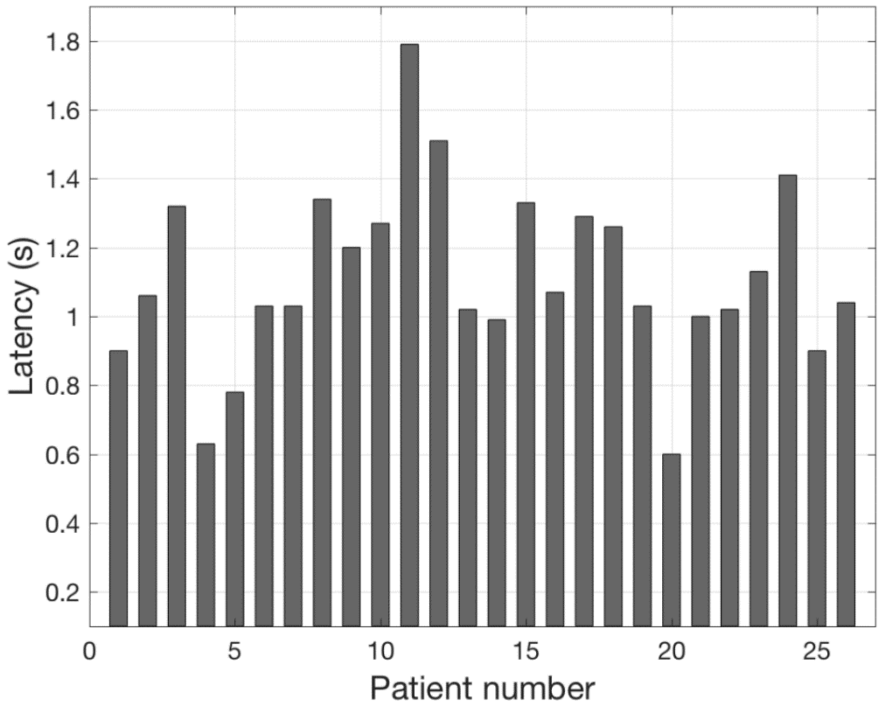


Figure 4.9. The detection latency of XGB-HW across patients.

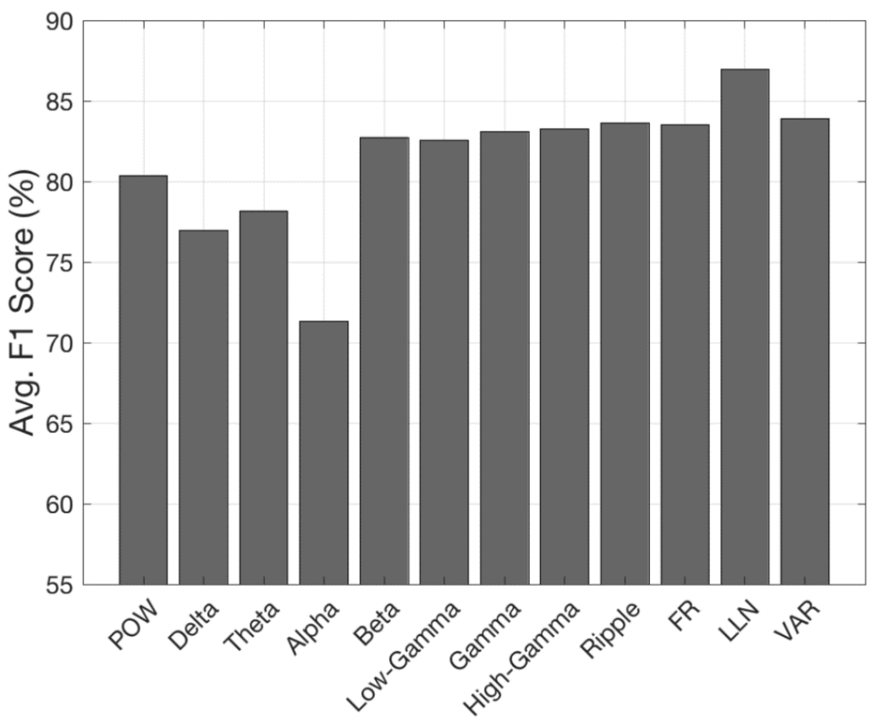
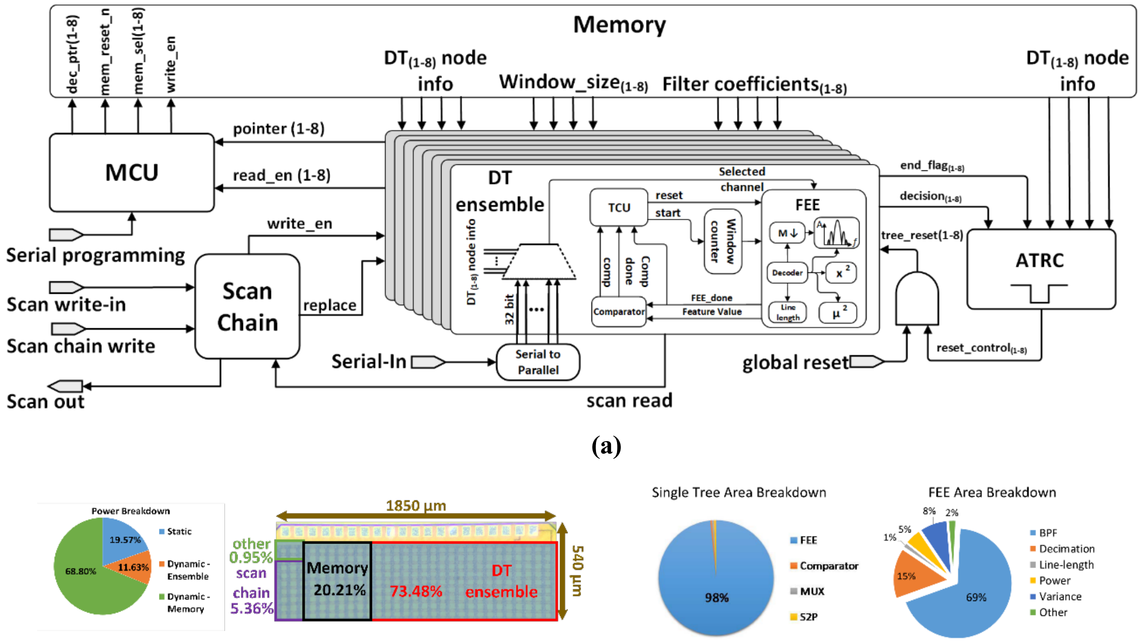


Figure 4.10. Overall feature importance for the proposed classifier.



(a)

(b)

Figure 4.11. Implemented Hardware, (a) Block diagram of the implemented SoC; (b) Power breakdown, die micrograph, and area breakdown of a single tree and FEE.

would significantly relax the area-power constraints in feature extraction module. The FIR filters have Type-I direct symmetric structures with 7 and 35 taps for the first and second stages, respectively. A direct symmetric structure enables using half the multipliers needed for a standard FIR filter, as well as 50% saving in coefficient memory. A high number of taps would lead to extra power and area in FEE and memory. To select optimal number of taps, extensive analysis was made. Given the importance of higher frequency features in seizure detection as shown in Figure 4.10, we particularly focused on the required accuracy for capturing low-amplitude ripple and fast ripple features (i.e., HFOs) with short duration and rare occurrence [174], [176]. Thus, the filter architecture and length were chosen to ensure lower than 5% error in HFO extraction over the entire training set.

4.7.3 Memory Control Unit

MCU monitors the read/write access to the memory. In the write mode, a decoder activates different memory sub-modules for programming through the serial input, that is generated during patient-specific training. The filter coefficients and prediction model are stored in memory. The fully programmable memory allocation enables a patient-specific seizure detection. The total size of the register type memory is less than 1kB, with shared filter coefficients using 228B. The memory associated with filter coefficients is shared among trees. Thus, it is not scaled by increasing the number of trees. Each DT has a dedicated 690b of memory for its node information (690B for 8 trees). Four sub-memory blocks with a depth of 15 store the tree structure, including each node's feature/channel selection, decimation filter selection, threshold, and leaf values, tree structure (whether there is a child node or not), and window size for feature extraction.

In the read mode, MCU receives pointer address and commands from each DT, and sends back the requested information. It also activates/deactivates the associated filter coefficients from memory to DTs, according to the corresponding node info. Trees work independently in a parallel fashion, using an Asynchronous Tree Reset Control.

4.7.4 Asynchronous Tree Reset Control

To effectively capture all abnormalities in the data, each tree works independently and computes its trained features to maximize the accuracy, regardless of computational delay. When the 'tree-end' flag of a tree is raised, ATRC stores the tree status and resets it to the initial state. After reset is cleared, the tree starts processing of new input data. ATRC holds the tree status until the next available 'tree-end' flag. Finally, ATRC assigns each tree's respective leaf values to calculate D_{final} according to equation 4.6.

4.7.4.1 Input precision

The input bit precision should be sufficiently high to ensure the detectability of weak high-frequency features. According to [189], at least 12-bit resolution is required to extract correct FR patterns for seizure onset detection. On the other hand, lower bit resolution is preferred to reduce the chip area and power. To find the required number of bits, HFOs from various patients were calculated at 9-12 bit precisions of input data, and compared to those extracted from ideal floating point input. With some extra margin that accounts for lower effective resolution of ADC, we chose 12 bits that ensures less than 0.1% error in the amplitude of HFOs.

4.7.4.2 Experimental setup and measurement results

The chip micrograph of the proposed classification architecture fabricated in a 65nm TSMC process and its area breakdown are depicted in Figure 4.11(b), as well as the area breakdown of a single tree and the FEE. Each tree, including its dedicated and shared memory units, takes 11.25% of the die area. Figure 4.11(b) also shows the power breakdown of the proposed SoC operating at a 0.8V supply, with an energy efficiency of 41.2nJ/class. Power measurements were made at worst-case scenarios where all the internal registers are switching and FEE is saturated (i.e., electrical onset of seizure is approaching).

In order to test the seizure detection performance of the fabricated chip, iEEG recordings from epileptic patients were digitized on a local PC with 12-bit resolution. The digitized data

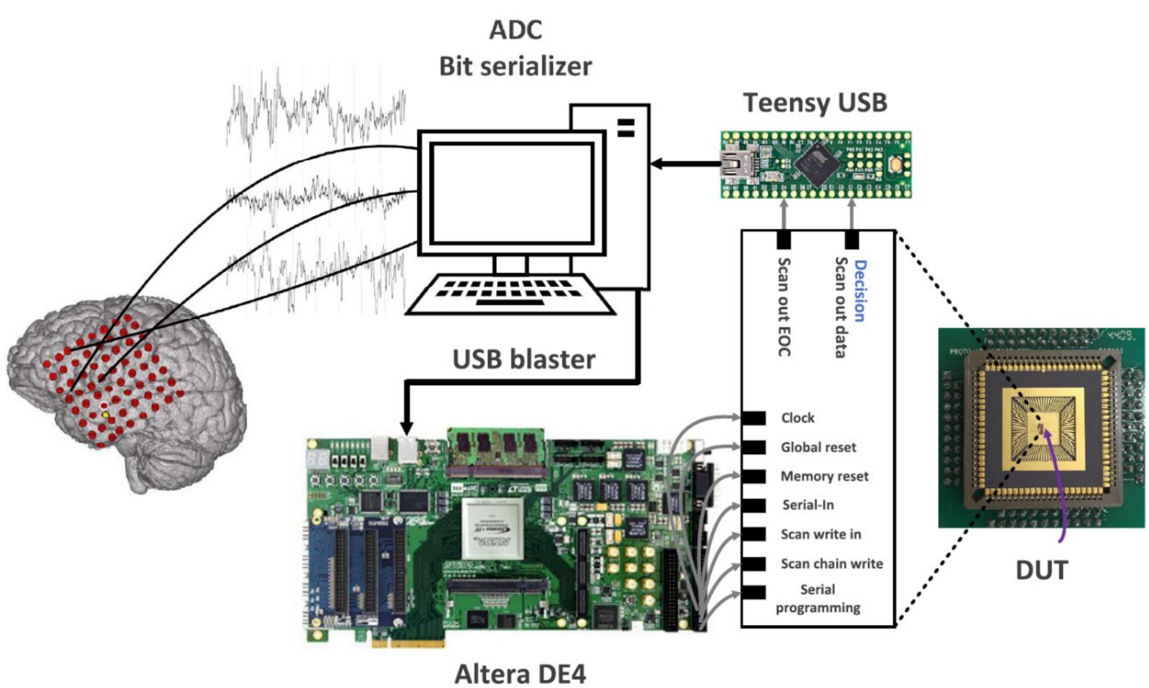


Figure 4.12. Experimental setup to measure the on-chip classifier.

of all channels were then serialized and stored on the DDR2 SDRAM of an Altera DE4 board, as shown in Figure 4.12. The information of prediction model was serially sent to the Serial Programming input of the implemented SoC (shown on the right). Once the prediction model is stored on memory, FPGA provides input clock and start command to SoC. For each patient, the chip is programmed according to the ensemble structure of his/her trained prediction model. Then, the test iEEG data of that patient is loaded to the chip for feature extraction and classification. Using the measured decisions, sensitivity and specificity are calculated. We tested the chip with 2253 hours of iEEG data from 20 patients. As the chip handles up to 32 input channels, those patients with up to 32 channels in their trained prediction model were used for the test. Given the limited data storage on FPGA, up to 10 hours of iEEG data was used for each test. The exact duration was determined based on the state of iEEG data. In the case of significant seizure-like activity in the vicinity of 10 hour, the duration of test data was reduced to 9 hours, with the last 1-hr added to the following experiment. Table IV summarizes the performance of this system compared to the state-of-the-art on-chip classifiers for seizure detection. In measurements, the classifier achieves an average sensitivity and specificity of 83.7% and 88.1%, respectively. For a fair comparison with state-of-the-art, energy and area of [162] are normalized to the 65nm technology node. The proposed architecture achieves over 27× improvement in energy-area-latency product.

4.8 Scalability and Hardware Optimization

The small number of channels in existing neural interface technology remains a barrier to the therapeutic potential. For instance, the spatial coverage and resolution of electrodes has a high impact on the detection accuracy of epilepsy implants [174]. The proposed XGB classifier in this work is inherently scalable to multi-sensor and multichannel operation, through sharing the computational and memory resources for feature extraction and classification among channels. In contrast to a majority of other classifiers that linearly scale in computational and memory requirements with number of channels and features, the proposed classifier computes a handful of features per tree, regardless of total channel count.

Table 4.4. SoC Performance and Comparison

Parameter	**This Work**	**ISSCC'13** [162]	**JSSC'13** [161]	**JSSC'14** [163]	**JSSC'13** [164]
Process	**65 nm**	180 nm	180 nm	180 nm	180 nm
Classifier	**XGB**	Non-Lin SVM	Lin-SVM	LLS	SVM‡
Signal Modality	**iEEG**	EEG	EEG	iEEG	EEG
Channel Count	**32**	8	8	Digital	18
Energy Eff.	**41.2nJ/class**	1.23*μJ/class	1.52*μJ/class	77.91μJ/class	273μJ/class
Logic Size	**330k**	2.27M	3.3M	N.A.	371k
Memory [kB]	**1**	N.A.	N.A.	N.A.	32**
Area	**1 mm^2**	7 mm^2*	8.18 mm^2*	6.5 mm^2	5.13 mm^2
Sensitivity [%]	**83.7**	95.1	N.A.	92	N.A.
Specificity [%]	**88.1**	94	N.A.	N.A.	N.A.
Latency [s]	**1.79††**	2	2	0.8	N.A.

* Area and Energy Efficiency conservatively estimated from A/P breakdown.
** 32kB SV MEM, 16kB Programming MEM, 16kB Data MEM
† Number of equivalent NAND2 gates with driving strength of one.
†† Worst case latency (patient 11)
‡Linear, Polynomial, RBF.

This approach enables significant savings in computational resources and required storage on chip.

Although we have chosen a relatively simple feature set in this study, one may use additional complex and non-linear features to boost the accuracy at a negligible cost. The total number of feature extraction units to be physically placed on chip is proportional to number of trees, while only one feature is computed in each tree at a time, saving both power and area. In other words, we can include as many features as the application requires, since they only scale up with number of trees and do not pose excessive memory and hardware requirements. Without any channel selection or feature reduction techniques (that is required in most traditional methods due to large dimension of features), the proposed classifier inherently selects an optimal set of channels and related features that form the tree structure. Thus, the main contribution of this work is a software-hardware co-design approach to enable energy reduction by minimizing the number of simultaneously extracted features, thus breaking the energy-area vs. accuracy tradeoff. Buffer-less processing of data in a closed-loop scheme is employed, and programmable bandpass filters further decrease the overall area overhead. The total power can be further reduced by dynamically controlling the channel activation and powering down the low-noise amplifiers in unused channels.

4.8.1 Energy-Quality Tradeoffs and Scaling

In our proposed gradient-boosting classifier, each tree contributes to roughly 10% of total power (static and dynamic). Based on the performance curves shown in Figure 4.7, we chose to implement an ensemble of eight trees with a maximum depth of four, to achieve an average AUC of more than 90% across a large population of patients with varying number of electrodes, seizures, and sampling rates. However, not all patients in our database need as many trees for an accurate discrimination of their seizures, as depicted in Figure 4.13 (top curves). Therefore, we enabled a programmable on/off control for each tree in the ensemble, so that upon a patient-specific training phase, one or more trees could be switched off to save power, with a minimum impact on quality. In other words, depending on the difficulty of detection task, the required number of trees can be switched on to achieve an expected classification accuracy (e.g., eight trees for patients with hardly detectable seizures, such as patient 24 in Figure 4.13). We use the AUC as our quality metric, that is widely used to evaluate the predictive accuracy of a classifier.

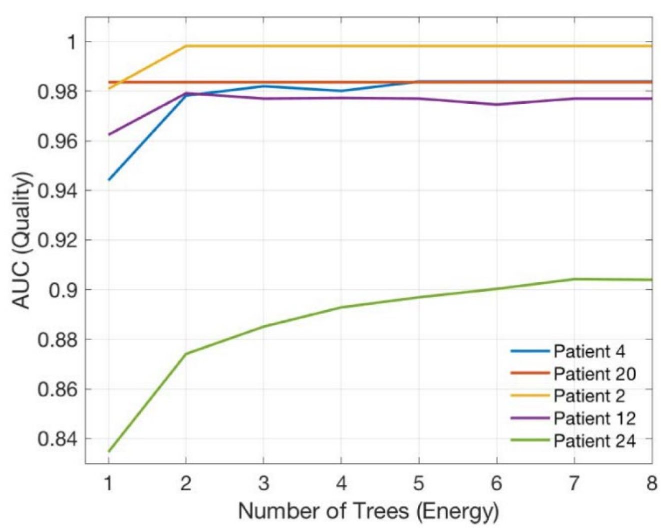


Figure 4.13. Measured AUC versus number of trees for various patients.

Boosting methods generally attain high discrimination by sequential training of weak classifiers. Here, the XGB attempts to increase the predictive accuracy by making a more accurate prediction at each iteration [181]. However, increasing the number of DTs increases the memory and power requirements of the system. The proposed XGB hardware is inherently quality-scalable through programming the number and depth of the active trees, with a maximum depth set at four. Moreover, our design offers a unique flexibility to accommodate various tree structures specific to each patient, to trade the predictive accuracy with energy (i.e., avoid unnecessary energy dissipation when accuracy is just enough for a patient). We explored the hardware parameters of tree count and depth across all patients, as potential knobs for energy-quality scaling.

As shown in Figure 4.12, we observe that in most patients, a small number of trees are sufficient for a reliable seizure detection. Indeed, the structure of successive trees are very similar in most patients, and by switching off the last few trees, we only observe a slight decrease in predictive accuracy. While chip area is limited by the required number of trees for worst case patients, the energy usage can be scaled for cases with easily detectable seizures. The other alternatives (knobs) for energy-quality scaling include pruning of trees or forcing the algorithm to use energy-aware features by modifying the cost function (i.e., adding an energy constraint similar to the delay constraint in Algorithm 4.1). However, we specifically observed that for most patients, the very last 3–4 trees in the iterative training process of XGB have a slight impact on performance and could even cause overfitting. In addition, our proposed asynchronous approach requires a single FEE in each tree that freely runs to compute one feature at a time. Thus, its energy is less sensitive to the depth parameter and is rather controlled by sampling frequency. Thus, we have focused on the hardware knob of tree count, that is easily integrated into our power-aware classification prototype.

4.8.2 Discussion on Hardware Optimization

Various opportunities to improve the energy and area efficiency of proposed classifier could be explored that remain as a future work. For instance, the input bit precision in our chip implementation has been chosen sufficiently high to allow the detectability of high-frequency features. Given the inherent error tolerance in machine learning algorithms, the energy per classification can be reduced by relaxing the quality or precision of features. For low-power and compact implementation in particular, reducing the resolution of coefficients in filter banks, feature thresholds, and leaf values is critical. New approaches to train decision trees with fixed-point and low-cost parameters can be investigated, similar to the works that reduce precision in DNNs [151], SVMs and LRs [150]. Since the training is usually performed offline, the associated cost is not critical. Such parameters could further be used as potential knobs in the proposed energy-quality scaling framework.

Furthermore, DTs can be trained to incorporate the costs of misclassification (FP or FN) and feature computation (power, area, delay) in the tree induction process. For example, it is critical to achieve a high sensitivity in seizure detection, while keeping the false alarm rate and latency below a tolerable level. This can lead to development of cost-sensitive decision trees, where the top-down tree induction algorithm may be adapted to maintain a pre-specified cost, therefore trading off the unnecessary accuracy (e.g., very high specificity or low latency) and energy. Besides, using various design parameters of DTs, the XGB classifier can be programmed to trade energy and quality in a structured and dynamic fashion.

4.9 Summary

In this work, we addressed the challenge of designing a low-power machine learning algorithm for on-chip neural data classification. By using software-hardware co-design approach, we proposed a novel hardware architecture for a gradient-boosted decision tree model, with a single feature extraction engine and programmable FIR filter per tree. The proposed asynchronous tree operation enables efficient classification of multichannel neural data, with significantly lower memory, power and area requirements compared to state-of-the-art. As a result, this on-chip classifier achieves an energy-area-latency product that is 27× lower than prior works, while processing the highest number of channels. The hardware architecture, design optimization and tradeoffs are discussed, and algorithm performance based on proposed model and SoC measurements is presented. Such classifiers could potentially allow full integration of processing circuitry with the sensor array in various resource-constrained biomedical applications.

Chapter 5

CONCLUSION

Integrating AI in the design of wearable and implantable medical devices is intended to simplify the complexities of medical data analysis, making it more useful in clinical settings to enhance patient care and assist healthcare providers in extracting relevant information. In this thesis, we explored how AI can potentially be used in the design, implementation, and utilization of biomedical systems while emphasizing its importance in advancing healthcare technology. The initial chapter establish a basis for understanding the complexity involved in processing biomedical data. This shows the potential AI's ability to manage complex medical data and variations in human physiology, leading to improvements in the accuracy of diagnostic models and personalized treatment approaches. Enhanced data analysis by AI coupled with more advanced algorithms can potentially extract valuable insights and more comprehensive interpretability from complex medical datasets, transforming patient care, and aiding healthcare providers in advancing treatment techniques.

In Chapter 2, we explored the utilization of machine learning in Brain-Machine Interfaces (BMIs) to show that the application of ML in BMIs has the potential to enhance human nervous system links with medical devices, especially those used by patients with neurological disorders. First, we demonstrated that among four different algorithms—Kalman Filter, Deep Neural Network (DNN), SimpleRNN cells, and Long-Short-Term Memory (LSTM) cells—LSTM-based decoder provides improved performance in BMI technology when measured using Pearson Correlation Coefficient (ρ). Following this, the development of deep multi-state Dynamic Recurrent Neural Network (DRNN) decoder operating on wavelet-based neural features is presented as an approach to find the complex and nonlinear relationships between neural input and movement kinematics for computer cursor control. This part emphasizes on how DRNN can potentially create improved BMI solutions, elaborating on its efficiency in terms of memory usage and power consumption for future developments of BMI systems in hardware. However, this study extends beyond the field of BMI to offer potential assistive technologies that enhance device control and interaction leading to an improved life quality for affected individuals like spinal-cord injury (SCI) patients.

In Chapter 3, we introduced EKGNet, which combines analog computation with deep learning to achieve a higher level of accuracy in the identification of heartbeat arrhythmias. The highler balanced accuracies obtained from EKGNet's design relate to intra-patient classification of arrhythmia and myocardial infarction (MI), which significantly improves the heartbeat arrythmia detection by using an efficient and accurate classifier that consume little power. By utilizing the energy efficiency of transistors operating in the sub-threshold region, EKGNet overcomes the constraints of traditional analog-to-digital conversion (ADC), enhancing its suitability for biomedical applications. This work demonstrates the potential of adaptable machine learning techniques and AI-driven technologies in the progress of early detection of heart diseases.

In chapter 4, we presented an energy efficient hardware design for seizure detection by utilizing XGBoost, a machine learning technique based on gradient-boosted trees, to achieve high performance in detecting epileptic activities and to perform accurate real-time seizure monitoring. The enhancement in performance offered by this proposed architecture, evidenced by the averaged F1 scores and the improvement in the energy-area-latency product, indicates this design's potential for integration into current medical devices. The adaptability of this architecture to different numbers of trees for personalized patient care can potentially be considered as an important advance in designing customized, power-efficient implantable and wearable medical devices. This thesis argues that this approach has the potential to be used to reduce the risks associated with undetected seizures so as to facilitate early interventions while steering clear of seizure detection technologies' conventional standards.

In summary, this thesis demonstrates that machine learning has the potential to be used in the design, the implementation, and the utilization of biomedical systems for the treatment of different medical conditions. AI and ML can potentially improve healthcare treatments through new architectures like DRNN and FENet for BMI technology, EKGNet for arrhythmia classification, and XGBoost for seizure detection, contributing to an era of personalized, efficient, and effective healthcare. An in-depth exploration of these applications indicates the capability of these techniques at present in the medical services sector, serving as a catalyst for future developments of medical devices by underlining the importance of ongoing research and development activities.

BIBLIOGRAPHY

[1] W. Wu and N. G. Hatsopoulos, “Real-Time Decoding of Nonstationary Neural Activity in Motor Cortex,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 16, no. 3, pp. 213–222, Jun. 2008, doi: 10.1109/TNSRE.2008.922679.

[2] T. Aflalo *et al.*, “Decoding Motor Imagery from the Posterior Parietal Cortex of a Tetraplegic Human,” *Science*, vol. 348, no. 6237, pp. 906–910, May 2015, doi: 10.1126/science.aaa5417.

[3] D. Sussillo, S. D. Stavisky, J. C. Kao, S. I. Ryu, and K. V. Shenoy, “Making Brain–Machine Interfaces Robust to Future Neural Variability,” *Nat. Commun.*, vol. 7, no. 1, p. 13749, Dec. 2016, doi: 10.1038/ncomms13749.

[4] D. Sussillo *et al.*, “A Recurrent Neural Network for Closed-Loop Intracortical Brain–Machine Interface Decoders,” *J. Neural Eng.*, vol. 9, no. 2, p. 026027, Mar. 2012, doi: 10.1088/1741-2560/9/2/026027.

[5] B. Allahgholizadeh Haghi *et al.*, “Deep Multi-State Dynamic Recurrent Neural Networks Operating on Wavelet Based Neural Features for Robust Brain Machine Interfaces,” *Adv. Neural Inf. Process. Syst.*, vol. 32, 2019, Accessed: Feb. 15, 2021.

[6] R. A. Andersen, S. Kellis, C. Klaes, and T. Aflalo, “Toward More Versatile and Intuitive Cortical Brain Machine Interfaces,” *Curr. Biol.*, vol. 24, no. 18, pp. 885–897, 2014, doi: 10.1016/j.cub.2014.07.068.

[7] M. Shoaran, B. A. Haghi, M. Taghavi, M. Farivar, and A. Emami-Neyestanak, “Energy-Efficient Classification for Resource-Constrained Biomedical Applications,” *IEEE J. Emerg. Sel. Top. Circuits Syst.*, vol. 8, no. 4, Art. no. 4, Dec. 2018, doi: 10.1109/JETCAS.2018.2844733.

[8] M. W. Dewhirst, B. L. Viglianti, M. Lora-Michiels, M. Hanson, and P. J. Hoopes, “Basic Principles of Thermal Dosimetry and Thermal Thresholds for Tissue Damage from Hyperthermia,” *Int. J. Hyperthermia*, vol. 19, no. 3, pp. 267–294, Jan. 2003, doi: 10.1080/0265673031000119006.

[9] M. Taghavi, B. A. Haghi, M. Farivar, M. Shoaran, and A. Emami, "A 41.2 nJ/class, 32-Channel On-Chip Classifier for Epileptic Seizure Detection," in *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Jul. 2018, pp. 3693–3696. doi: 10.1109/EMBC.2018.8513243.

[10] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, Art. no. 7553, May 2015, doi: 10.1038/nature14539.

[11] "Pattern Recognition and Machine Learning | SpringerLink." Accessed: Feb. 06, 2024. [Online]. Available: https://link.springer.com/book/9780387310732

[12] M. Kachuee, S. Fazeli, and M. Sarrafzadeh, "ECG Heartbeat Classification: A Deep Transferable Representation," in *2018 IEEE International Conference on Healthcare Informatics (ICHI)*, Jun. 2018, pp. 443–444. doi: 10.1109/ICHI.2018.00092.

[13] M. A. Schwemmer *et al.*, "Meeting Brain–Computer Interface User Performance Expectations Using A Deep Neural Network Decoding Framework," *Nat. Med.*, vol. 24, no. 11, pp. 1669–1676, Nov. 2018, doi: 10.1038/s41591-018-0171-y.

[14] "Deep Learning." Accessed: Dec. 08, 2022. [Online]. Available: https://www.deeplearningbook.org/

[15] P. Rajpurkar, A. Y. Hannun, M. Haghpanahi, C. Bourn, and A. Y. Ng, "Cardiologist-Level Arrhythmia Detection with Convolutional Neural Networks." arXiv, Jul. 06, 2017. doi: 10.48550/arXiv.1707.01836.

[16] S. Shah *et al.*, "Decoding Kinematics from Human Parietal Cortex using Neural Networks," in *2019 9th International IEEE/EMBS Conference on Neural Engineering (NER)*, Mar. 2019, pp. 1138–1141. doi: 10.1109/NER.2019.8717137.

[17] "Spinal Cord Injury Facts and Figures at a Glance." National Spinal Cord Injury Statistical Center, 2018. Accessed: Jun. 30, 2021. [Online]. Available: https://www.nscisc.uab.edu/Public/Facts%20and%20Figures%20-%202018.pdf

[18] S. Musallam, B. D. Corneil, B. Greger, H. Scherberger, and R. A. Andersen, "Cognitive Control Signals for Neural Prosthetics," *Science*, vol. 305, no. 5681, Art. no. 5681, Jul. 2004, doi: 10.1126/science.1097938.

[19] C. T. Moritz, S. I. Perlmutter, and E. E. Fetz, "Direct Control of Paralysed Muscles by Cortical Neurons," *Nature*, vol. 456, no. 7222, pp. 639–642, 2008.

[20] S. Kellis, K. Miller, K. Thomson, R. Brown, P. House, and B. Greger, “Decoding Spoken Words Using Local Field Potentials Recorded from the Cortical Surface,” *J. Neural Eng.*, vol. 7, no. 5, p. 056007, Oct. 2010, doi: 10.1088/1741-2560/7/5/056007.

[21] C. Pandarinath *et al.*, “High Performance Communication by People with Paralysis Using An Intracortical Brain-Computer Interface,” *eLife*, vol. 6, Feb. 2017, doi: 10.7554/eLife.18554.

[22] T. Aflalo, B. A. Haghi, R. A. Andersen, and A. Emami, “Features Extraction Network for Estimating Neural Activity from Electrical Recordings,” US20240046071A1, Feb. 08, 2024 Accessed: Mar. 14, 2024. [Online]. Available: https://patents.google.com/patent/US20240046071A1/en

[23] J. Schläpfer and H. J. Wellens, “Computer-Interpreted Electrocardiograms: Benefits and Limitations,” *J. Am. Coll. Cardiol.*, vol. 70, no. 9, pp. 1183–1192, Aug. 2017, doi: 10.1016/j.jacc.2017.07.723.

[24] K. Nezamabadi, N. Sardaripour, B. Haghi, and M. Forouzanfar, “Unsupervised ECG Analysis: A Review,” *IEEE Rev. Biomed. Eng.*, vol. 16, pp. 208–224, 2023, doi: 10.1109/RBME.2022.3154893.

[25] “Cardiovascular diseases (CVDs).” Accessed: Jun. 11, 2023. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds)

[26] “Heart Rhythm Disorders | UpBeat.org - powered by the Heart Rhythm Society.” Accessed: Jun. 11, 2023. [Online]. Available: https://upbeat.org/heart-rhythm-disorders

[27] B. Haghi, L. Ma, S. Lale, A. Anandkumar, and A. Emami, “EKGNet: A 10.96µW Fully Analog Neural Network for Intra-Patient Arrhythmia Classification,” in *2023 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, Oct. 2023, pp. 1–5. doi: 10.1109/BioCAS58349.2023.10389164.

[28] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, “Weight Uncertainty in Neural Networks.” arXiv, May 21, 2015. doi: 10.48550/arXiv.1505.05424.

[29] G. Hinton, O. Vinyals, and J. Dean, “Distilling the Knowledge in a Neural Network.” arXiv, Mar. 09, 2015. Accessed: Jun. 11, 2023. [Online]. Available: http://arxiv.org/abs/1503.02531

[30] T. L. Skarpaas and M. J. Morrell, “Intracranial Stimulation Therapy for Epilepsy,” *Neurotherapeutics*, vol. 6, no. 2, pp. 238–243, Apr. 2009, doi: 10.1016/j.nurt.2009.01.022.

[31] M. A. Bin Altaf, C. Zhang, and J. Yoo, “A 16-Channel Patient-Specific Seizure Onset and Termination Detection SoC With Impedance-Adaptive Transcranial Electrical Stimulator,” *IEEE J. Solid-State Circuits*, vol. 50, no. 11, pp. 2728–2740, Nov. 2015, doi: 10.1109/JSSC.2015.2482498.

[32] T. Chen and C. Guestrin, “XGBoost: A Scalable Tree Boosting System,” in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco California USA: ACM, Aug. 2016, pp. 785–794. doi: 10.1145/2939672.2939785.

[33] M. Shoaran, B. A. Haghi, M. Farivar, and A. Emami, “Efficient Feature Extraction and Classification Methods in Neural Interfaces,” in *Frontiers of Engineering: Reports on Leading-Edge Engineering from the 2017 Symposium*, National Academies Press (US), 2018. Accessed: Feb. 05, 2024. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK481634/

[34] M. Shoaran, M. Taghavi, B. Haghi, M. Farivar, and A. Emami, “Energy Efficient On-Chip Classifier for Detecting Physiological Conditions,” 16946151, 2020 [Online]. Available: https://patentimages.storage.googleapis.com/44/d3/8a/40caab42856d37/US20200388397A1.pdf

[35] A. Jackson, C. T. Moritz, J. Mavoori, T. H. Lucas, and E. E. Fetz, “The Neurochip BCI: Towards A Neural Prosthesis for Upper Limb Function,” *IEEE Trans. Neural Syst. Rehabil. Eng. Publ. IEEE Eng. Med. Biol. Soc.*, vol. 14, no. 2, pp. 187–190, Jun. 2006, doi: 10.1109/TNSRE.2006.875547.

[36] V. Gilja *et al.*, “A High-Performance Neural Prosthesis Enabled by Control Algorithm Design,” *Nat Neurosci*, vol. 15, no. 12, pp. 1752–7, Dec. 2012, doi: 10.1038/nn.3265.

[37] V. Gilja *et al.*, “Clinical Translation of a High-Performance Neural Prosthesis,” *Nat. Med.*, vol. 21, no. 10, pp. 1142–1145, Oct. 2015, doi: 10.1038/nm.3953.

[38] B. Jarosiewicz *et al.*, "Virtual Typing by People with Tetraplegia Using A Self-Calibrating Intracortical Brain-Computer Interface," *Sci. Transl. Med.*, vol. 7, no. 313, Art. no. 313, Nov. 2015, doi: 10.1126/scitranslmed.aac7328.

[39] P. Nuyujukian, J. M. Fan, J. C. Kao, S. I. Ryu, and K. V. Shenoy, "A High-Performance Keyboard Neural Prosthesis Enabled by Task Optimization," *IEEE Trans. Biomed. Eng.*, vol. 62, no. 1, Art. no. 1, Jan. 2015, doi: 10.1109/TBME.2014.2354697.

[40] W. Wu, Y. Gao, E. Bienenstock, J. P. Donoghue, and M. J. Black, "Bayesian Population Decoding of Motor Cortical Activity Using a Kalman Filter," *Neural Comput.*, vol. 18, no. 1, pp. 80–118, Jan. 2006, doi: 10.1162/089976606774841585.

[41] A. L. Orsborn, S. Dangi, H. G. Moorman, and J. M. Carmena, "Closed-Loop Decoder Adaptation on Intermediate Time-Scales Facilitates Rapid BMI Performance Improvements Independent of Decoder Initialization Conditions," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 20, no. 4, pp. 468–477, Jul. 2012, doi: 10.1109/TNSRE.2012.2185066.

[42] J. Wessberg *et al.*, "Real-Time Prediction of Hand Trajectory by Ensembles of Cortical Neurons in Primates," *Nature*, vol. 408, no. 6810, Art. no. 6810, Nov. 2000, doi: 10.1038/35042582.

[43] "Recurrent Neural Networks for Prediction: Learning Algorithms, Architectures and Stability | Wiley," Wiley.com. Accessed: May 18, 2023. [Online]. Available: https://www.wiley.com/en-us/Recurrent+Neural+Networks+for+Prediction%3A+Learning+Algorithms%2C+Architectures+and+Stability-p-9780471495178

[44] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to Forget: Continual Prediction with LSTM," *Neural Comput.*, vol. 12, no. 10, pp. 2451–2471, Oct. 2000, doi: 10.1162/089976600300015015.

[45] "Keras: Deep Learning for Humans." Accessed: Feb. 04, 2024. [Online]. Available: https://keras.io/

[46] Liang Jin, P. N. Nikiforuk, and M. M. Gupta, "Approximation of Discrete-Time State-Space Trajectories Using Dynamic Recurrent Neural Networks," *IEEE Trans. Autom. Control*, vol. 40, no. 7, pp. 1266–1270, Jul. 1995, doi: 10.1109/9.400480.

[47] L. Tian and C. Collins, “A Dynamic Recurrent Neural Network-Based Controller for A Rigid–Flexible Manipulator System,” *Mechatronics*, vol. 14, no. 5, pp. 471–490, Jun. 2004, doi: 10.1016/j.mechatronics.2003.10.002.

[48] P. S. Ow and T. E. Morton, “Filtered Beam Search in Scheduling,” *Int. J. Prod. Res.*, vol. 26, no. 1, pp. 35–62, Jan. 1988, doi: 10.1080/00207548808947840.

[49] H. Daumé, J. Langford, and D. Marcu, “Search-Based Structured Prediction,” *Mach. Learn.*, vol. 75, no. 3, pp. 297–325, Jun. 2009, doi: 10.1007/s10994-009-5106-x.

[50] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, “Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2015. Accessed: Feb. 04, 2024. [Online]. Available: https://proceedings.neurips.cc/paper/2015/hash/e995f98d56967d946471af29d7bf99f1-Abstract.html

[51] F. Chollet and others, “Keras: The Python Deep Learning library,” *Astrophys. Source Code Libr.*, p. ascl:1806.022, Jun. 2018.

[52] F. Pedregosa *et al.*, “Scikit-learn: Machine Learning in Python,” *Mach. Learn. PYTHON*.

[53] A. Paszke *et al.*, “Automatic differentiation in PyTorch”.

[54] C. Pandarinath *et al.*, “Inferring Single-Trial Neural Population Dynamics Using Sequential Auto-Encoders,” *Nat. Methods*, vol. 15, no. 10, pp. 805–815, Oct. 2018, doi: 10.1038/s41592-018-0109-9.

[55] D. Basak, S. Pal, and D. Patranabis, “Support Vector Regression,” *Neural Inf. Process. – Lett. Rev.*, vol. 11, Nov. 2007.

[56] L. Shpigelman, H. Lalazar, and E. Vaadia, “Kernel-ARMA for Hand Tracking and Brain-Machine interfacing During 3D Motor Control,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2008. Accessed: Feb. 04, 2024. [Online]. Available: https://proceedings.neurips.cc/paper/2008/hash/61b4a64be663682e8cb037d9719ad8cd-Abstract.html

[57] K. Cho *et al.*, “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation.” arXiv, Sep. 02, 2014. Accessed: Feb. 05, 2024. [Online]. Available: http://arxiv.org/abs/1406.1078

[58] L. Breiman, “Random Forests,” *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001, doi: 10.1023/A:1010933404324.

[59] J. R. Quinlan, “Induction of Decision Trees,” *Mach. Learn.*, vol. 1, no. 1, pp. 81–106, Mar. 1986, doi: 10.1007/BF00116251.

[60] B. P. Christie *et al.*, “Comparison of Spike Sorting and Thresholding of Voltage Waveforms for Intracortical Brain-Machine Interface Performance,” *J. Neural Eng.*, vol. 12, no. 1, p. 016009, Feb. 2015, doi: 10.1088/1741-2560/12/1/016009.

[61] null Hao Nan, B. A. Haghi, and A. Arbabian, “Interferogram-Based Breast Tumor Classification Using Microwave-Induced Thermoacoustic Imaging,” *Annu. Int. Conf. IEEE Eng. Med. Biol. Soc. IEEE Eng. Med. Biol. Soc. Annu. Int. Conf.*, vol. 2015, pp. 2717–2720, Aug. 2015, doi: 10.1109/EMBC.2015.7318953.

[62] J. Bergstra and Y. Bengio, “Random Search for Hyper-Parameter Optimization”.

[63] S. Gowda, A. L. Orsborn, S. A. Overduin, H. G. Moorman, and J. M. Carmena, “Designing Dynamical Properties of Brain–Machine Interfaces to Optimize Task-Specific Performance,” *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 22, no. 5, pp. 911–920, Sep. 2014, doi: 10.1109/TNSRE.2014.2309673.

[64] P. N. Whatmough, S. K. Lee, D. Brooks, and G.-Y. Wei, “DNN Engine: A 28-nm Timing-Error Tolerant Sparse Deep Neural Network Processor for IoT Applications,” *IEEE J. Solid-State Circuits*, vol. 53, no. 9, pp. 2722–2731, Sep. 2018, doi: 10.1109/JSSC.2018.2841824.

[65] M. Shah *et al.*, “A Fixed-Point Neural Network Architecture for Speech Applications on Resource Constrained Hardware,” *J. Signal Process. Syst.*, vol. 90, no. 5, pp. 727–741, May 2018, doi: 10.1007/s11265-016-1202-x.

[66] L. R. Hochberg *et al.*, “Neuronal Ensemble Control of Prosthetic Devices by A Human with Tetraplegia,” *Nature*, vol. 442, no. 7099, Art. no. 7099, Jul. 2006, doi: 10.1038/nature04970.

[67] M. D. Golub, B. M. Yu, A. B. Schwartz, and S. M. Chase, “Motor Cortical Control of Movement Speed with Implications for Brain-Machine Interface Control,” *J. Neurophysiol.*, vol. 112, no. 2, pp. 411–429, Jul. 2014, doi: 10.1152/jn.00391.2013.

[68] Y. Inoue, H. Mao, S. B. Suway, J. Orellana, and A. B. Schwartz, “Decoding Arm Speed During Reaching,” *Nat. Commun.*, vol. 9, no. 1, Art. no. 1, Dec. 2018, doi: 10.1038/s41467-018-07647-3.

[69] S. B. Hamed, M. H. Schieber, and A. Pouget, “Decoding M1 Neurons During Multiple Finger Movements,” *J. Neurophysiol.*, vol. 98, no. 1, pp. 327–333, Jul. 2007, doi: 10.1152/jn.00760.2006.

[70] M. M. Churchland and K. V. Shenoy, “Temporal Complexity and Heterogeneity of Single-Neuron Activity in Premotor and Motor Cortex,” *J. Neurophysiol.*, vol. 97, no. 6, pp. 4235–4257, Jun. 2007, doi: 10.1152/jn.00095.2007.

[71] T. N. Aflalo and M. S. A. Graziano, “Relationship between Unconstrained Arm Movements and Single-Neuron Firing in the Macaque Motor Cortex,” *J. Neurosci.*, vol. 27, no. 11, pp. 2760–2780, Mar. 2007, doi: 10.1523/JNEUROSCI.3147-06.2007.

[72] T. N. Aflalo and M. S. A. Graziano, “Partial Tuning of Motor Cortex Neurons to Final Posture in A Free-Moving Paradigm,” *Proc. Natl. Acad. Sci.*, vol. 103, no. 8, pp. 2909–2914, Feb. 2006, doi: 10.1073/pnas.0511139103.

[73] J. M. Goodman *et al.*, “Postural Representations of the Hand in the Primate Sensorimotor Cortex,” *Neuron*, vol. 104, no. 5, pp. 1000-1009.e7, Dec. 2019, doi: 10.1016/j.neuron.2019.09.004.

[74] “Stabilization of A Brain–Computer Interface via the Alignment of Low-Dimensional Spaces of Neural Activity | Nature Biomedical Engineering.” Accessed: Feb. 05, 2024. [Online]. Available: https://www.nature.com/articles/s41551-020-0542-9

[75] “Long-term Stability of Cortical Population Dynamics Underlying Consistent Behavior | Nature Neuroscience.” Accessed: Feb. 05, 2024. [Online]. Available: https://www.nature.com/articles/s41593-019-0555-4

[76] M. Zhang *et al.*, “Extracting Wavelet Based Neural Features from Human Intracortical Recordings for Neuroprosthetics Applications,” *Bioelectron. Med.*, vol. 4, p. 11, 2018, doi: 10.1186/s42234-018-0011-x.

[77] A. L. Orsborn, H. G. Moorman, S. A. Overduin, M. M. Shanechi, D. F. Dimitrov, and J. M. Carmena, "Closed-Loop Decoder Adaptation Shapes Neural Plasticity for Skillful Neuroprosthetic Control," *Neuron*, vol. 82, no. 6, Art. no. 6, Jun. 2014, doi: 10.1016/j.neuron.2014.04.048.

[78] M. S. Willsey *et al.*, "Real-Time Brain-Machine Interface In Non-Human Primates Achieves High-Velocity Prosthetic Finger Movements Using a Shallow Feedforward Neural Network Decoder," *Nat. Commun.*, vol. 13, no. 1, Art. no. 1, Nov. 2022, doi: 10.1038/s41467-022-34452-w.

[79] F. R. Willett, D. T. Avansino, L. R. Hochberg, J. M. Henderson, and K. V. Shenoy, "High-Performance Brain-to-Text Communication via Handwriting," *Nature*, vol. 593, no. 7858, Art. no. 7858, May 2021, doi: 10.1038/s41586-021-03506-2.

[80] P. T. Sadtler, S. I. Ryu, E. C. Tyler-Kabara, B. M. Yu, and A. P. Batista, "Brain-Computer Interface Control Along Instructed Paths," *J. Neural Eng.*, vol. 12, no. 1, p. 016015, Feb. 2015, doi: 10.1088/1741-2560/12/1/016015.

[81] D. Young *et al.*, "Closed-Loop Cortical Control of Virtual Reach and Posture Using Cartesian and Joint Velocity Commands," *J. Neural Eng.*, vol. 16, no. 2, Art. no. 2, Jan. 2019, doi: 10.1088/1741-2552/aaf606.

[82] J. M. Fan, P. Nuyujukian, J. C. Kao, C. A. Chestek, S. I. Ryu, and K. V. Shenoy, "Intention Estimation in Brain-Machine Interfaces," *J. Neural Eng.*, vol. 11, no. 1, p. 016004, Feb. 2014, doi: 10.1088/1741-2560/11/1/016004.

[83] J. Martinez, C. Pedreira, M. J. Ison, and R. Quian Quiroga, "Realistic Simulation of Extracellular Recordings," *J. Neurosci. Methods*, vol. 184, no. 2, pp. 285–293, Nov. 2009, doi: 10.1016/j.jneumeth.2009.08.017.

[84] "Compositional Coding of Individual Finger Movements in Human Posterior Parietal Cortex and Motor Cortex Enables Ten-Finger Decoding | medRxiv." Accessed: Jan. 03, 2023. [Online]. Available: https://www.medrxiv.org/content/10.1101/2022.12.07.22283227v1

[85] C. Klaes *et al.*, "Hand Shape Representations in the Human Posterior Parietal Cortex," *J. Neurosci.*, vol. 35, no. 46, Art. no. 46, Nov. 2015, doi: 10.1523/JNEUROSCI.2747-15.2015.

[86] T. Aflalo, C. Y. Zhang, E. R. Rosario, N. Pouratian, G. A. Orban, and R. A. Andersen, "A Shared Neural Substrate for Action Verbs and Observed Actions in Human Posterior Parietal Cortex," *Sci. Adv.*, vol. 6, no. 43, Art. no. 43, Oct. 2020, doi: 10.1126/sciadv.abb3984.

[87] C. Guan *et al.*, "Stability of Motor Representations After Paralysis," *eLife*, vol. 11, p. e74478, Sep. 2022, doi: 10.7554/eLife.74478.

[88] M. Jafari *et al.*, "The Human Primary Somatosensory Cortex Encodes Imagined Movement in the Absence of Sensory Information," *Commun. Biol.*, vol. 3, no. 1, Art. no. 1, Dec. 2020, doi: 10.1038/s42003-020-01484-1.

[89] R. A. Andersen, T. Aflalo, and S. Kellis, "From Thought to Action: The Brain–Machine Interface in Posterior Parietal Cortex," *Proc. Natl. Acad. Sci.*, vol. 116, no. 52, Art. no. 52, Dec. 2019, doi: 10.1073/pnas.1902276116.

[90] G. A. Orban and F. Caruana, "The Neural Basis of Human Tool Use," *Front. Psychol.*, vol. 5, 2014, Accessed: Dec. 08, 2022. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fpsyg.2014.00310

[91] J. P. Gallivan and J. C. Culham, "Neural Coding within Human Brain Areas Involved in Actions," *Curr. Opin. Neurobiol.*, vol. 33, pp. 141–149, Aug. 2015, doi: 10.1016/j.conb.2015.03.012.

[92] C. Guan *et al.*, "Decoding and Geometry of Ten Finger Movements in Human Posterior Parietal Cortex and Motor Cortex," *J. Neural Eng.*, 2023, doi: 10.1088/1741-2552/acd3b1.

[93] N. Ejaz, M. Hamada, and J. Diedrichsen, "Hand Use Predicts the Structure of Representations in Sensorimotor Cortex," *Nat. Neurosci.*, vol. 18, no. 7, Art. no. 7, Jul. 2015, doi: 10.1038/nn.4038.

[94] K. A. Ludwig, R. M. Miriani, N. B. Langhals, M. D. Joseph, D. J. Anderson, and D. R. Kipke, "Using a Common Average Reference to Improve Cortical Neuron Recordings From Microelectrode Arrays," *J. Neurophysiol.*, vol. 101, no. 3, pp. 1679–1689, Mar. 2009, doi: 10.1152/jn.90989.2008.

[95] E. Stark and M. Abeles, "Predicting Movement from Multiunit Activity," *J. Neurosci.*, vol. 27, no. 31, pp. 8387–8394, Aug. 2007, doi: 10.1523/JNEUROSCI.1321-07.2007.

[96] N. Y. Masse *et al.*, “Non-Causal Spike Filtering Improves Decoding of Movement Intention for Intracortical BCIs,” *J. Neurosci. Methods*, vol. 236, pp. 58–67, Oct. 2014, doi: 10.1016/j.jneumeth.2014.08.004.

[97] F. R. Willett *et al.*, “Principled BCI Decoder Design and Parameter Selection Using a Feedback Control Model,” *Sci. Rep.*, vol. 9, no. 1, Art. no. 1, Jun. 2019, doi: 10.1038/s41598-019-44166-7.

[98] P. Geladi and B. R. Kowalski, “Partial Least-Squares Regression: A Tutorial,” *Anal. Chim. Acta*, vol. 185, pp. 1–17, Jan. 1986, doi: 10.1016/0003-2670(86)80028-9.

[99] A. Paszke *et al.*, “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2019. Accessed: Dec. 09, 2022. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html

[100] S. Ryoo, C. I. Rodrigues, S. S. Baghsorkhi, S. S. Stone, D. B. Kirk, and W. W. Hwu, “Optimization Principles and Application Performance Evaluation of A Multithreaded GPU Using CUDA,” in *Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming - PPoPP ’08*, Salt Lake City, UT, USA: ACM Press, 2008, p. 73. doi: 10.1145/1345206.1345220.

[101] D. C. Montgomery, E. A. Peck, and G. G. Vining, ”Introduction to linear regression analysis,” 2nd ed. New York: John Wiley & Sons, 1992. [Online]. Available: https://scholar.google.com/scholar_lookup?title=Introduction%20to%20Linear%20Regression%20Analysis&author=D.C.%20Montgomery&publication_year=2015

[102] G. H. Mulliken, S. Musallam, and R. A. Andersen, “Decoding Trajectories from Posterior Parietal Cortex Ensembles,” *J. Neurosci.*, vol. 28, no. 48, Art. no. 48, Nov. 2008, doi: 10.1523/JNEUROSCI.1463-08.2008.

[103] D. E. Rumelhart, G. E. Hintont, and R. J. Williams, “Learning Representations by Back-Propagating Errors,” p. 4, 1986.

[104] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” p. 30.

[105] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization.” arXiv, Jan. 29, 2017. Accessed: Dec. 09, 2022. [Online]. Available: http://arxiv.org/abs/1412.6980

[106] J. Snoek, H. Larochelle, and R. P. Adams, “Practical Bayesian Optimization of Machine Learning Algorithms.” arXiv, Aug. 29, 2012. doi: 10.48550/arXiv.1206.2944.

[107] John M. Chambers and Trevor J. Hastie, ”Statistical Models in S,” 1st ed. New York: Routledge, 1992. [Online]. Available: https://doi.org/10.1201/9780203738535

[108] William S. Cleveland, Eric Grosse, and William M. Shyu, *Local Regression Models*, 1st Edition. Routledge, 1992.

[109] S. M. Lundberg and S.-I. Lee, “A Unified Approach to Interpreting Model Predictions,” in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2017. Accessed: May 26, 2023. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html

[110] M. Awad and R. Khanna, “Support Vector Regression,” in *Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers*, M. Awad and R. Khanna, Eds., Berkeley, CA: Apress, 2015, pp. 67–80. doi: 10.1007/978-1-4302-5990-9_4.

[111] A. J. Smola and B. Schölkopf, “A Tutorial on Support Vector Regression,” *Stat. Comput.*, vol. 14, no. 3, pp. 199–222, Aug. 2004, doi: 10.1023/B:STCO.0000035301.49549.88.

[112] O. G. Sani, H. Abbaspourazad, Y. T. Wong, B. Pesaran, and M. M. Shanechi, “Modeling Behaviorally Relevant Neural Dynamics Enabled by Preferential Subspace Identification,” *Nat. Neurosci.*, vol. 24, no. 1, Art. no. 1, Jan. 2021, doi: 10.1038/s41593-020-00733-0.

[113] C. R. Rao, ”Linear Statistical Inference and its Applications,” 2nd ed. Wiley-Interscience, 2001.

[114] N. Kriegeskorte, M. Mur, and P. Bandettini, “Representational SimilarityAnalysis - Connecting the Branches of Systems Neuroscience,” *Front. Syst. Neurosci.*, vol. 2,

2008, Accessed: May 19, 2023. [Online]. Available: https://www.frontiersin.org/articles/10.3389/neuro.06.004.2008

[115] "Representational Models: A Common Framework for Understanding Encoding, Pattern-Component, and Representational-Similarity Analysis | PLOS Computational Biology." Accessed: May 19, 2023. [Online]. Available: https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1005508

[116] T. C. Kietzmann, C. J. Spoerer, L. K. A. Sörensen, R. M. Cichy, O. Hauk, and N. Kriegeskorte, "Recurrence Is Required to Capture the Representational Dynamics of the Human Visual System," *Proc. Natl. Acad. Sci.*, vol. 116, no. 43, pp. 21854–21863, Oct. 2019, doi: 10.1073/pnas.1905544116.

[117] N. Kriegeskorte *et al.*, "Matching Categorical Object Representations in Inferior Temporal Cortex of Man and Monkey," *Neuron*, vol. 60, no. 6, pp. 1126–1141, Dec. 2008, doi: 10.1016/j.neuron.2008.10.043.

[118] P. Yuan, X. Gao, B. Allison, Y. Wang, G. Bin, and S. Gao, "A Study of the Existing Problems of Estimating the Information Transfer Rate in Online Brain–Computer Interfaces," *J. Neural Eng.*, vol. 10, no. 2, p. 026014, Feb. 2013, doi: 10.1088/1741-2560/10/2/026014.

[119] F. Cabitza, R. Rasoini, and G. F. Gensini, "Unintended Consequences of Machine Learning in Medicine," *JAMA*, vol. 318, no. 6, pp. 517–518, Aug. 2017, doi: 10.1001/jama.2017.7797.

[120] Y. Zhang, P. Tiňo, A. Leonardis, and K. Tang, "A Survey on Neural Network Interpretability," *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 5, no. 5, pp. 726–742, Oct. 2021, doi: 10.1109/TETCI.2021.3100641.

[121] S. F. Cogan, "Neural Stimulation and Recording Electrodes," *Annu. Rev. Biomed. Eng.*, vol. 10, no. 1, pp. 275–309, 2008, doi: 10.1146/annurev.bioeng.10.061807.160518.

[122] J. C. Barrese *et al.*, "Failure Mode Analysis of Silicon-Based Intracortical Microelectrode Arrays in Non-Human Primates," *J. Neural Eng.*, vol. 10, no. 6, Art. no. 6, Dec. 2013, doi: 10.1088/1741-2560/10/6/066014.

[123] C. A. Chestek *et al.*, “Long-Term Stability of Neural Prosthetic Control Signals from Silicon Cortical Arrays in Rhesus Macaque Motor Cortex,” *J. Neural Eng.*, vol. 8, no. 4, p. 045005, Aug. 2011, doi: 10.1088/1741-2560/8/4/045005.

[124] J. E. Downey, N. Schwed, S. M. Chase, A. B. Schwartz, and J. L. Collinger, “Intracortical Recording Stability in Human Brain-Computer Interface Users,” *J. Neural Eng.*, vol. 15, no. 4, Art. no. 4, Aug. 2018, doi: 10.1088/1741-2552/aab7a0.

[125] S. R. Nason *et al.*, “A Low-Power Band of Neuronal Spiking Activity Dominated by Local Single Units Improves the Performance of Brain–Machine Interfaces,” *Nat. Biomed. Eng.*, vol. 4, no. 10, Art. no. 10, Oct. 2020, doi: 10.1038/s41551-020-0591-0.

[126] E. M. Trautmann *et al.*, “Accurate Estimation of Neural Population Dynamics without Spike Sorting,” *Neuron*, vol. 103, no. 2, pp. 292-308.e4, Jul. 2019, doi: 10.1016/j.neuron.2019.05.003.

[127] C. Gold, D. A. Henze, C. Koch, and G. Buzsáki, “On the Origin of the Extracellular Action Potential Waveform: A Modeling Study,” *J. Neurophysiol.*, vol. 95, no. 5, pp. 3113–3128, May 2006, doi: 10.1152/jn.00979.2005.

[128] C. Stringer, M. Pachitariu, N. Steinmetz, C. B. Reddy, M. Carandini, and K. D. Harris, “Spontaneous Behaviors Drive Multidimensional, Brainwide Activity,” *Science*, vol. 364, no. 6437, p. eaav7893, Apr. 2019, doi: 10.1126/science.aav7893.

[129] A. Y. Hannun *et al.*, “Cardiologist-Level Arrhythmia Detection and Classification in Ambulatory Electrocardiograms Using a Deep Neural Network,” *Nat. Med.*, vol. 25, no. 1, Art. no. 1, Jan. 2019, doi: 10.1038/s41591-018-0268-3.

[130] U. R. Acharya *et al.*, “A Deep Convolutional Neural Network Model to Classify Heartbeats,” *Comput. Biol. Med.*, vol. 89, pp. 389–396, Oct. 2017, doi: 10.1016/j.compbiomed.2017.08.022.

[131] Md. Rashed-Al-Mahfuz *et al.*, “Deep Convolutional Neural Networks Based ECG Beats Classification to Diagnose Cardiovascular Conditions,” *Biomed. Eng. Lett.*, vol. 11, no. 2, pp. 147–162, May 2021, doi: 10.1007/s13534-021-00185-w.

[132] J. Kojuri, R. Boostani, P. Dehghani, F. Nowroozipour, and N. Saki, “Prediction of Acute Myocardial Infarction with Artificial Neural Networks in Patients with

Nondiagnostic Electrocardiogram," *J. Cardiovasc. Dis. Res.*, vol. 6, no. 2, pp. 51–59, May 2015, doi: 10.5530/jcdr.2015.2.2.

[133] Z. Yan, J. Zhou, and W.-F. Wong, "Energy Efficient ECG Classification with Spiking Neural Network," *Biomed. Signal Process. Control*, vol. 63, p. 102170, Jan. 2021, doi: 10.1016/j.bspc.2020.102170.

[134] S.-Y. Hsu, Y. Ho, P.-Y. Chang, C. Su, and C.-Y. Lee, "A 48.6-to-105.2 μW Machine Learning Assisted Cardiac Sensor SoC for Mobile Healthcare Applications," *IEEE J. Solid-State Circuits*, vol. 49, no. 4, pp. 801–811, Apr. 2014, doi: 10.1109/JSSC.2013.2297406.

[135] Y. Zhao, Z. Shang, and Y. Lian, "A 13.34 μW Event-Driven Patient-Specific ANN Cardiac Arrhythmia Classifier for Wearable ECG Sensors," *IEEE Trans. Biomed. Circuits Syst.*, vol. 14, no. 2, pp. 186–197, Apr. 2020, doi: 10.1109/TBCAS.2019.2954479.

[136] Q. Cai, X. Xu, Y. Zhao, L. Ying, Y. Li, and Y. Lian, "A 1.3 μW Event-Driven ANN Core for Cardiac Arrhythmia Classification in Wearable Sensors," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 68, no. 9, pp. 3123–3127, Sep. 2021, doi: 10.1109/TCSII.2021.3091198.

[137] J. Liu *et al.*, "4.5 BioAIP: A Reconfigurable Biomedical AI Processor with Adaptive Learning for Versatile Intelligent Health Monitoring," in *2021 IEEE International Solid- State Circuits Conference (ISSCC)*, Feb. 2021, pp. 62–64. doi: 10.1109/ISSCC42613.2021.9365996.

[138] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge, "Near-Threshold Computing: Reclaiming Moore's Law Through Energy Efficient Integrated Circuits," *Proc. IEEE*, vol. 98, no. 2, pp. 253–266, Feb. 2010, doi: 10.1109/JPROC.2009.2034764.

[139] C. Mead, "Neuromorphic Electronic Systems," *Proc. IEEE*, vol. 78, no. 10, pp. 1629–1636, Oct. 1990, doi: 10.1109/5.58356.

[140] S. Arora, R. Ge, B. Neyshabur, and Y. Zhang, "Stronger generalization bounds for deep nets via a compression approach." arXiv, Nov. 26, 2018. doi: 10.48550/arXiv.1802.05296.

[141] A. Graves, A. Mohamed, and G. Hinton, "Speech Recognition with Deep Recurrent Neural Networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2013, pp. 6645–6649. doi: 10.1109/ICASSP.2013.6638947.

[142] O. Rivasplata, V. M. Tankasali, and C. Szepesvari, "PAC-Bayes with Backprop." arXiv, Oct. 04, 2019. doi: 10.48550/arXiv.1908.07380.

[143] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition." arXiv, Dec. 10, 2015. doi: 10.48550/arXiv.1512.03385.

[144] A. L. Goldberger *et al.*, "PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals," *Circulation*, vol. 101, no. 23, pp. E215-220, Jun. 2000, doi: 10.1161/01.cir.101.23.e215.

[145] G. B. Moody and R. G. Mark, "The Impact of the MIT-BIH Arrhythmia Database," *IEEE Eng. Med. Biol. Mag.*, vol. 20, no. 3, pp. 45–50, May 2001, doi: 10.1109/51.932724.

[146] R. Bousseljot, D. Kreiseler, and A. Schnabel, "Nutzung der EKG-Signaldatenbank CARDIODAT der PTB über das Internet," vol. 40, no. s1, pp. 317–318, Jan. 1995, doi: 10.1515/bmte.1995.40.s1.317.

[147] A. for the Advancement of Medical Instrumentation et al., "Testing and Reporting Performance Results of Cardiac Rhythm and ST Segment Measurement Algorithms." ANSI/AAMI EC38, 1998.

[148] P. Kaur and A. Gosain, "Comparing the Behavior of Oversampling and Undersampling Approach of Class Imbalance Learning by Combining Class Imbalance Problem with Noise," 2018, pp. 23–30. doi: 10.1007/978-981-10-6602-3_3.

[149] L. van der Maaten and G. Hinton, "Visualizing Data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 86, pp. 2579–2605, 2008.

[150] H. Albalawi, Y. Li, and X. Li, "Training Fixed-Point Classifiers for On-Chip Low-Power Implementation," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 22, no. 4, pp. 1–18, Oct. 2017, doi: 10.1145/3057275.

[151] V. Sze, "Designing Hardware for Machine Learning: The Important Role Played by Circuit Designers," *IEEE Solid-State Circuits Mag.*, vol. 9, no. 4, pp. 46–54, 2017, doi: 10.1109/MSSC.2017.2745798.

[152] B. Murmann, D. Bankman, E. Chai, D. Miyashita, and L. Yang, "Mixed-Signal Circuits for Embedded Machine-Learning Applications," in *2015 49th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, USA: IEEE, Nov. 2015, pp. 1341–1345. doi: 10.1109/ACSSC.2015.7421361.

[153] "UPenn and Mayo Clinic's Seizure Detection Challenge." Accessed: Feb. 06, 2024. [Online]. Available: https://kaggle.com/competitions/seizure-detection

[154] M. Alioto, "Energy-Quality Scalable Adaptive VLSI Circuits and Systems Beyond Approximate Computing," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, Lausanne, Switzerland: IEEE, Mar. 2017, pp. 127–132. doi: 10.23919/DATE.2017.7926970.

[155] W. C. Stacey and B. Litt, "Technology Insight: Neuroengineering and Epilepsy—Designing Devices for Seizure Control," *Nat. Clin. Pract. Neurol.*, vol. 4, no. 4, pp. 190–201, Apr. 2008, doi: 10.1038/ncpneuro0750.

[156] A. Shoeb, H. Edwards, J. Connolly, B. Bourgeois, S. Ted Treves, and J. Guttag, "Patient-Specific Seizure Onset Detection," *Epilepsy Behav.*, vol. 5, no. 4, pp. 483–498, Aug. 2004, doi: 10.1016/j.yebeh.2004.05.005.

[157] N. Verma, A. Shoeb, J. Bohorquez, J. Dawson, J. Guttag, and A. P. Chandrakasan, "A Micro-Power EEG Acquisition SoC With Integrated Feature Extraction Processor for a Chronic Seizure Detection System," *IEEE J. Solid-State Circuits*, vol. 45, no. 4, pp. 804–816, Apr. 2010, doi: 10.1109/JSSC.2010.2042245.

[158] M. Shoaran, C. Pollo, K. Schindler, and A. Schmid, "A Fully Integrated IC With 0.85-μW/Channel Consumption for Epileptic iEEG Detection," *IEEE Trans. Circuits Syst. II Express Briefs*, vol. 62, no. 2, pp. 114–118, Feb. 2015, doi: 10.1109/TCSII.2014.2387652.

[159] M. Shoaran *et al.*, "A 16-Channel 1.1mm ^{2} Implantable Seizure Control SoC with Sub-μW/Channel Consumption and Closed-Loop Stimulation in 0.18μm CMOS," in *2016*

IEEE Symposium on VLSI Circuits (VLSI-Circuits), Honolulu, HI, USA: IEEE, Jun. 2016, pp. 1–2. doi: 10.1109/VLSIC.2016.7573557.

[160] C. Zhang, M. A. Bin Altaf, and J. Yoo, "Design and Implementation of an On-Chip Patient-Specific Closed-Loop Seizure Onset and Termination Detection System," *IEEE J. Biomed. Health Inform.*, vol. 20, no. 4, pp. 996–1007, Jul. 2016, doi: 10.1109/JBHI.2016.2553368.

[161] J. Yoo, L. Yan, D. El-Damak, M. A. B. Altaf, A. H. Shoeb, and A. P. Chandrakasan, "An 8-Channel Scalable EEG Acquisition SoC With Patient-Specific Seizure Classification and Recording Processor," *IEEE J. Solid-State Circuits*, vol. 48, no. 1, pp. 214–228, Jan. 2013, doi: 10.1109/JSSC.2012.2221220.

[162] M. A. B. Altaf, J. Tillak, Y. Kifle, and J. Yoo, "A 1.83 μJ/classification nonlinear support-vector-machine-based patient-specific seizure classification SoC," in *2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, San Francisco, CA: IEEE, Feb. 2013, pp. 100–101. doi: 10.1109/ISSCC.2013.6487654.

[163] W. M. Chen *et al.*, "A Fully Integrated 8-Channel Closed-Loop Neural-Prosthetic CMOS SoC for Real-Time Epileptic Seizure Control," *IEEE J. Solid-State Circuits*, vol. 49, no. 1, pp. 232–247, Jan. 2014, doi: 10.1109/JSSC.2013.2284346.

[164] K. H. Lee and N. Verma, "A Low-Power Processor With Configurable Embedded Machine-Learning Accelerators for High-Order and Adaptive Analysis of Medical-Sensor Signals," *IEEE J. Solid-State Circuits*, vol. 48, no. 7, pp. 1625–1637, Jul. 2013, doi: 10.1109/JSSC.2013.2253226.

[165] Z. Wang, R. E. Schapire, and N. Verma, "Error Adaptive Classifier Boosting (EACB): Leveraging Data-Driven Training Towards Hardware Resilience for Signal Inference," *IEEE Trans. Circuits Syst. Regul. Pap.*, vol. 62, no. 4, pp. 1136–1145, Apr. 2015, doi: 10.1109/TCSI.2015.2395591.

[166] T. C. Chen *et al.*, "1.4 μW/Channel 16-Channel EEG/ECoG Processor for Smart Brain Sensor SoC," in *2010 Symposium on VLSI Circuits*, Honolulu, HI, USA: IEEE, Jun. 2010, pp. 21–22. doi: 10.1109/VLSIC.2010.5560258.

[167] A. Page, C. Sagedy, E. Smith, N. Attaran, T. Oates, and T. Mohsenin, "A Flexible Multichannel EEG Feature Extractor and Classifier for Seizure Detection," *IEEE*

Trans. Circuits Syst. II Express Briefs, vol. 62, no. 2, pp. 109–113, Feb. 2015, doi: 10.1109/TCSII.2014.2385211.

[168] T. Roh, K. Song, H. Cho, D. Shin, and H.-J. Yoo, "A Wearable Neuro-Feedback System With EEG-Based Mental Status Monitoring and Transcranial Electrical Stimulation," *IEEE Trans. Biomed. Circuits Syst.*, vol. 8, no. 6, pp. 755–764, Dec. 2014, doi: 10.1109/TBCAS.2014.2384017.

[169] "International Epilepsy Electrophysiology Portal." Accessed: Feb. 06, 2024. [Online]. Available: https://main.ieeg.org/

[170] J. Zhang, L. Huang, Z. Wang, and N. Verma, "A Seizure-Detection IC Employing Machine Learning to Overcome Data Conversion and Analog-Processing Non-Idealities," in *2015 IEEE Custom Integrated Circuits Conference (CICC)*, San Jose, CA, USA: IEEE, Sep. 2015, pp. 1–4. doi: 10.1109/CICC.2015.7338456.

[171] A. Bragin, J. Engel Jr, C. L. Wilson, I. Fried, and G. Buzsáki, "High-Frequency Oscillations in Human Brain," *Hippocampus*, vol. 9, no. 2, pp. 137–142, 1999, doi: 10.1002/(SICI)1098-1063(1999)9:2<137::AID-HIPO5>3.0.CO;2-0.

[172] M. Ayinala and K. K. Parhi, "Low Complexity Algorithm for Seizure Prediction Using Adaboost," in *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, San Diego, CA: IEEE, Aug. 2012, pp. 1061–1064. doi: 10.1109/EMBC.2012.6346117.

[173] M. Bandarabadi, A. Dourado, C. A. Teixeira, T. I. Netoff, and K. K. Parhi, "Seizure Prediction with Bipolar Spectral Power Features Using Adaboost and SVM Classifiers," in *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Osaka: IEEE, Jul. 2013, pp. 6305–6308. doi: 10.1109/EMBC.2013.6610995.

[174] M. Stead *et al.*, "Microseizures and the Spatiotemporal Scales of Human Partial Epilepsy," *Brain*, vol. 133, no. 9, pp. 2789–2797, Sep. 2010, doi: 10.1093/brain/awq190.

[175] L. B. Stone Jerome Friedman, R. A. Olshen, Charles J., "Classification and Regression Trees," New York: Routledge, 2017. doi: 10.1201/9781315139470.

[176] M. Shoaran, M. Farivar, and A. Emami, “Hardware-Friendly Seizure Detection with A Boosted Ensemble of Shallow Decision Trees,” in *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, Orlando, FL, USA: IEEE, Aug. 2016, pp. 1826–1829. doi: 10.1109/EMBC.2016.7591074.

[177] A. Y. Benbasat and J. A. Paradiso, “A Framework for the Automated Generation of Power-Efficient Classifiers for Embedded Sensor Nodes,” in *Proceedings of the 5th international conference on Embedded networked sensor systems*, Sydney Australia: ACM, Nov. 2007, pp. 219–232. doi: 10.1145/1322263.1322285.

[178] Y. Yang, S. Boling, and A. J. Mason, “A Hardware-Efficient Scalable Spike Sorting Neural Signal Processor Module for Implantable High-Channel-Count Brain Machine Interfaces,” *IEEE Trans. Biomed. Circuits Syst.*, vol. 11, no. 4, pp. 743–754, Aug. 2017, doi: 10.1109/TBCAS.2017.2679032.

[179] “A 90 nm CMOS, Power-Proportional Acoustic Sensing Frontend for Voice Activity Detection,” *IEEE J. Solid-State Circuits*, vol. 51, no. 1, pp. 291–302, Jan. 2016, doi: 10.1109/JSSC.2015.2487276.

[180] J. H. Friedman, “Greedy Function Approximation: A Gradient Boosting Machine,” *Ann. Stat.*, vol. 29, no. 5, pp. 1189–1232, 2001.

[181] T. Chen and T. He, “xgboost: eXtreme Gradient Boosting”.

[182] L. Logesparan, A. J. Casson, and E. Rodriguez-Villegas, “Optimal Features for Online Seizure Detection,” *Med. Biol. Eng. Comput.*, vol. 50, no. 7, pp. 659–669, Jul. 2012, doi: 10.1007/s11517-012-0904-x.

[183] R. Esteller, J. Echauz, T. Tcheng, B. Litt, and B. Pless, “Line Length: An Efficient Feature for Seizure Onset Detection,” in *2001 Conference Proceedings of the 23rd Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Istanbul, Turkey: IEEE, 2001, pp. 1707–1710. doi: 10.1109/IEMBS.2001.1020545.

[184] K. Schindler, H. Leung, C. E. Elger, and K. Lehnertz, “Assessing Seizure Dynamics by Analysing the Correlation Structure of Multichannel Intracranial EEG,” *Brain*, vol. 130, no. 1, pp. 65–77, Nov. 2006, doi: 10.1093/brain/awl304.

[185] P. E. McSharry, L. A. Smith, and L. Tarassenko, "Comparison of Predictability of Epileptic Seizures by A Linear and A Nonlinear Method," *IEEE Trans. Biomed. Eng.*, vol. 50, no. 5, pp. 628–633, May 2003, doi: 10.1109/TBME.2003.810688.

[186] K. K. Majumdar and P. Vardhan, "Automatic Seizure Detection in ECoG by Differential Operator and Windowed Variance," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 19, no. 4, pp. 356–365, Aug. 2011, doi: 10.1109/TNSRE.2011.2157525.

[187] L. Ayoubian, H. Lacoma, and J. Gotman, "Automatic Seizure Detection in SEEG Using High Frequency Activities in Wavelet Domain," *Med. Eng. Phys.*, vol. 35, no. 3, pp. 319–328, Mar. 2013, doi: 10.1016/j.medengphy.2012.05.005.

[188] M. Bandarabadi, C. A. Teixeira, J. Rasekhi, and A. Dourado, "Epileptic Seizure Prediction Using Relative Spectral Power Features," *Clin. Neurophysiol.*, vol. 126, no. 2, pp. 237–248, Feb. 2015, doi: 10.1016/j.clinph.2014.05.022.

[189] Chengliang Qian, J. Shi, J. Parramon, and E. Sanchez-Sinencio, "A Low-Power Configurable Neural Recording System for Epileptic Seizure Detection," *IEEE Trans. Biomed. Circuits Syst.*, vol. 7, no. 4, pp. 499–512, Aug. 2013, doi: 10.1109/TBCAS.2012.2228857.