TOWARDS EFFICIENT AND SCALABLE MACHINE LEARNING FOR FUTURE NEURAL INTERFACES

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Bingzhao Zhu

August 2023

TOWARDS EFFICIENT AND SCALABLE MACHINE LEARNING FOR FUTURE NEURAL INTERFACES

Bingzhao Zhu, Ph.D.

Cornell University 2023

Closed-loop approaches in systems neuroscience and therapeutic stimulation have the potential to revolutionize our understanding of the brain and develop novel neuromodulation therapies for restoring lost functions. Neural interfaces with capabilities such as multi-channel neural recording, on-site signal processing, rapid symptom detection, and closed-loop stimulation are crucial for enabling these innovative treatments. However, current closed-loop neural interfaces are limited by their simplicity and lack of sufficient on-chip processing and intelligence.

This dissertation focuses on the development of next-generation neural decoders for closed-loop neural interfaces, utilizing on-chip machine learning to detect and suppress symptoms of neurological disorders. These neural decoders offer high versatility, low power consumption, minimal on-chip area, and robustness against neural signal fluctuations. Chapter 2 explores migraine state classification using somatosensory evoked potentials, an emerging application for neural interfaces. In Chapter 3, we introduce a resource-efficient oblique tree model that enables low-power, memory-efficient classifiers for real-time neurological disease detection and motor decoding. Chapter 4 presents a novel Tree in Tree decision graph model with applicability beyond neural data, demonstrating success in general tabular prediction tasks. In Chapter 5, we propose an adaptive machine learning-based decoder to compensate for fluc-

tuations in neural signals during test time. The dissertation concludes with a discussion of future research directions for on-chip neural decoders.

BIOGRAPHICAL SKETCH

Bingzhao Zhu received the B.Sc. degree in Opto-Electronics Science and Engineering from Zhejiang University, Hangzhou, China, in 2017. He is currently a Ph.D. candidate in Applied and Engineering Physics and a minor in Computer Science, at Cornell University, Ithaca, New York, USA. From 2020 to 2022, he was a visiting PhD student at Swiss Federal Institute of Technology (EPFL), Geneva, Switzerland. He completed an Applied Scientist internship at Amazon Web Service in 2022 Fall.

During Bingzhao Zhu's doctoral curriculum, he is fortunate to be advised by Prof. Mahsa Shoaran. His research interests include brain-computer interfaces (BCI), low-power machine learning, neural signal processing, and computational imaging.

To my family and inspiring mentors.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisor, Prof. Mahsa Shoaran, for her guidance, encouragement, and support throughout my PhD journey. Her expertise, patience, and constructive feedback have been invaluable to this thesis. I would like to extend my sincere thanks to the chair of my thesis committee, Chris Xu, and the minor member, Chris De Sa, for their support of my research.

I owe a special debt of gratitude to my family, particularly my parents and girlfriend, for their unwavering love, support, and belief in my abilities. Due to the COVID pandemic and visa restrictions, I was unable to be physically present with them during my academic journey. Despite these challenges, their sacrifices and encouragement have consistently provided me with the strength and resilience needed to persevere. This thesis would not have been possible without their constant encouragement and patience.

In conclusion, I am grateful to everyone who has contributed directly or indirectly to my academic journey and the completion of this thesis. Please know that your support has not gone unnoticed and is deeply appreciated.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Developing novel non-pharmacological treatments such as neurostimulation is becoming increasingly important to treat some of the most prevalent and intractable neurological disorders. Brain stimulation is currently the most common surgical treatment for movement disorders and has shown promise in epilepsy, neuropsychiatric disorders, memory, chronic pain, and traumatic brain injury, with new applications rapidly emerging. Despite promising proof-of-concept results, current clinical neurostimulators are limited in many aspects. For example, while deep-brain stimulation (DBS) can effectively control motor symptoms in most patients suffering from Parkinson's disease (PD), it causes persistent side effects (e.g., speech impairment and cognitive symptoms) [1, 2]. It is now widely known that this is due to the conventional "open-loop" approach, which involves delivering constant high-frequency (~130Hz) stimulation regardless of the patient's clinical state. In addition, open-loop stimulation increases the power consumption and the need for surgical battery replacement. This simplistic open-loop approach is also a key limiting factor in designing clinically effective stimulation for more complex disorders such as depression [3], Alzheimer's disease [4], and stroke [5, 6], among others [5, 7, 8].

To further leverage the benefits of stimulation and address the aforementioned limitations, closed-loop neuromodulation techniques have been recently explored, such as the responsive neurostimulator for epilepsy [9] and PD [10], with promising results. In this approach, stimulation is dynamically controlled according to a patient's clinical state, either with a continuous (i.e., adaptive) or an on-off (i.e., on-demand) strategy. Through feedback from relevant biomark-

ers of a neurological symptom (e.g., a seizure event, tremor episode, or mood change), closed-loop stimulation can titrate charge delivery to the brain, thus reducing the side effects and the amount of stimulation delivered, enhancing the therapeutic efficacy and battery life compared to its open-loop counterparts [2]. However, several critical challenges remain to be addressed in order to fully exploit the potential of closed-loop therapies for neurological disorders. The existing closed-loop devices mainly rely on simple comparison of a pre-selected biomarker (typically from 1 out of 4 channels) against a fixed threshold. Such simplistic approaches are known to be suboptimal in terms of predictive accuracy, resulting in low sensitivity and high false alarm rates, while exacerbating other symptoms [8]. Multiple biomarkers and control loops may be necessary to reliably improve symptoms, leading to design complexity.

A promising solution to address this challenge is to implement a machine learning (ML) algorithm directly on the implant or wearable to predict the onset or severity of neurological symptoms, an approach that has gained significant interest in recent years [11–15]. Real-time symptom control can be achieved through on-chip biomarker extraction and ML-based disease state detection, followed by a closed-loop intervention (e.g., electrical, magnetic or optical stimulation, drug delivery) to suppress the abnormal activity. This approach offers significant advantages over the conventional wireless transmission and external processing methods [16, 17] that suffer from feedback loop latency, high power consumption due to continuous telemetry, security and privacy concerns [18, 19]. A number of clinical trials have recently shown the advantage of machine learning-based control for closed-loop stimulation in movement disorders, epilepsy, and memory [4, 20]. In addition, machine learning systems have been developed to forecast the onset of neurological symptoms during preic-

tal phase, allowing sufficient time prior to seizure manifestation (e.g., in the order of several minutes) to provide early warnings to the patients and caregivers [21–23]. In closed-loop neural prostheses, however, both the ML decoder and neurostimulator are integrated on the implant, eliminating the need for excessively long symptom prediction horizons [24]. Therefore, most closed-loop devices train the classifier to differentiate ictal epochs from interictal period, several seconds prior to symptom onset [25]. Such systems detect the onset and termination (i.e., offset) of neurological symptoms to precisely control the delivery of stimulation [26].

Despite the benefits of using machine learning for closed-loop intervention, strict power and area requirements on an implantable or wearable device pose critical challenges for hardware realization of ML algorithms, particularly in the form of a miniaturized ASIC. The choice of learning algorithm and neural biomarkers affects the prediction accuracy and latency. Moreover, the prediction accuracy depends on the spatial resolution of the recording system and the number of input channels. Thus, there is a crucial need to develop high-performance, energy- and area-efficient biomarker extraction and ML solutions that are scalable to high channel counts and satisfy the implantable/wearable power budget and form factor.

This dissertation addresses various challenges in the development of next-generation ML-embedded neural interfaces. In Chapter 2, I explore the application of neural decoders for migraine attack prediction. I propose a machine learning approach to utilize somatosensory evoked potential (SSEP)-based biomarkers for migraine state classification in a noninvasive setting. Using a set of relevant features, we successfully separated migraine patients from

healthy controls with an accuracy of 89.7%. The proposed approach suggests the potential use of SSEP as a prominent and reliable signal in migraine state classification.

In Chapter 3, I introduce a machine learning model based on oblique decision trees to enable resource-efficient classification on a neural implant. By integrating model compression with probabilistic routing and implementing cost-aware learning, our proposed model could significantly reduce the memory and hardware cost compared to state-of-the-art models, while maintaining classification accuracy. We trained the resource-efficient oblique tree with power-efficient regularization (ResOT-PE) on three neural classification tasks to evaluate the performance, memory, and hardware requirements. On seizure detection task, we were able to reduce the model size by 3.4× and the feature extraction cost by 14.6× compared to the ensemble of boosted trees, using the intracranial EEG from 10 epilepsy patients. The proposed model can enable a low-power and memory-efficient implementation of classifiers for real-time neurological disease detection and motor decoding.

In Chapter 4, I present Tree in Tree decision graph (TnT), a framework that extends the conventional decision tree to a more generic and powerful directed acyclic graph. TnT constructs decision graphs by recursively growing decision trees inside the internal or leaf nodes instead of greedy training. The time complexity of TnT is linear to the number of nodes in the graph, and it can construct decision graphs on large datasets. Compared to decision trees, we show that TnT achieves better classification performance with reduced model size, both as a stand-alone classifier and as a base estimator in bagging/AdaBoost ensembles.

In Chapter 5, I develop an adaptive machine learning-based decoder capable of unsupervised adaptation to unseen target distributions. The adaptive decoder uses contextual information to compensate for distribution shifts in neural signals during test time and as a result, it automatically adapts to both abrupt and continual changes in neural signals. We evaluate our model on cross-session gait decoding tasks to reliably predict movement during natural walking, using neural recordings from a chronically implanted device in Parkinson patients. The adaptive model outperforms conventional decoders with fixed parameters and achieved robust gait decoding in the presence of therapy-induced domain shift. The proposed approach enables reliable neural decoding under common signal instabilities and could potentially advance adaptive stimulation control in more complex, home-based therapeutic settings.

Chapter 6 provides a summary of the dissertation while also briefly discussing potential future directions for on-chip neural decoding.

Appendix A further presents the results of my work on unsupervised domain adaptation to enable few-shot, cross-subject epileptic seizure detection. Using adversarial learning, features from multiple patients are encoded into a subject-invariant space and a discriminative model is trained on subject-invariant features to make predictions. We evaluate this approach on the intracranial EEG (iEEG) recordings from 9 patients with epilepsy. Our approach enables cross-subject seizure detection with a 9.4% improvement in 1-shot classification accuracy compared to the conventional subject-specific scheme. Appendix B introduces XTab, a framework for cross-table pretraining of tabular transformers on datasets from various domains. We address the challenge of inconsistent column types and quantities among tables by utilizing indepen-

dent featurizers and using federated learning to pretrain the shared component. Tested on 84 tabular prediction tasks from the OpenML-AutoML Benchmark (AMLB), we show that (1) XTab consistently boosts the generalizability, learning speed, and performance of multiple tabular transformers, (2) by pretraining FT-Transformer via XTab, we achieve superior performance than other state-of-the-art tabular deep learning models on various tasks such as regression, binary, and multiclass classification.

CHAPTER 2

MIGRAINE CLASSIFICATION USING SOMATOSENSORY EVOKED POTENTIALS

2.1 Introduction

Migraine is a disabling neurological disorder, characterized by recurrent headache attacks. Despite its high prevalence, migraine diagnosis is still mainly based on clinical interviews, patient diaries, and physical examinations. More advanced diagnostic methods are therefore desired for both clinical and research purposes and can potentially aid in early diagnosis and assessment of disease progression. Given the lack of consistent structural abnormalities, clinical neurophysiology methods are particularly suited to study the pathophysiology of migraine [27]. The neurophysiological techniques have been recently used in assessing the clinical fluctuations in migraine [28], and the effectiveness of emerging neuromodulation therapies such as repetitive transcranial magnetic stimulation [29]. Considering that early treatment of migraine headache is shown to be significantly more effective [30], it is important to predict migraine early in the course of attack, potentially with automated monitoring techniques. While wearable sensors and medical devices are increasingly being applied to the early diagnosis and treatment of neurological disorders, the field is relatively unexplored in migraine treatment. Such devices can detect the neurological abnormalities in real time and prompt patients to take preventive medications, monitor the progression of disease and effectiveness of medications, or trigger a therapy such as neuromodulation [15, 31]. Given the potential of noninvasive electrophysiological techniques recently shown in clinical migraine

studies [28, 32, 33], particularly the somatosensory evoked cortical potentials, we aim at assessing their power in classifying various states of migraine and separating migraine patients from healthy controls. This approach could potentially be used for early diagnosis of migraine attacks in a wearable setting in future.

According to most electrophysiological studies, migraine patients are characterized by hyper-responsivity in both somatosensory and visual cortices [34], which can be measured by standard electrophysiological techniques such as somatosensory evoked potential (SSEP) and visual evoked potential (VEP). Given the ease of evoking SSEPs (e.g. using a wristband-type device), our approach is based on the former. The SSEP correlates of the migraine brain were first reported in 2005 [32]. In particular, a lack of habituation in response to repetitive stimuli was found in migraine patients in the interictal state, both with (MA) or without aura (MO). The high-frequency oscillations (HFOs) superimposed on the median nerve SSEPs are widely reported as indicators of thalamo-cortical activation [35]. More specifically, the early and late HFO bursts are thought to be generated by the thalamo-cortical afferents and inhibitory neurons in the parietal cortex, respectively. A reduction of early HFO between attacks and increase of late HFO during attacks have been previously reported in migraine patients [33, 34]. In addition to HFOs, some low-frequency components are also critical in characterizing various states of migraine. For instance, the increased N20-P25 amplitude of low-frequency SSEP has been associated with the migraine ictal group [33]. Based on the altered SSEP signals in migraine patients, the associated HFO and low-frequency components could be used as potential features for our classification task. In other words, despite many unknowns in pathogenesis and underlying causes of migraine, the associated changes of

SSEPs allow us to reliably differentiate between the ictal or interictal phases of migraine, as well as healthy controls.

Although the electroencephalogram (EEG)-based biomarkers have been widely used in other neurological fields of research, such as epilepsy [36], studies on SSEP features are very limited. The resting-state EEG complexity was recently shown to be higher in the migraine preictal compared to the interictal group. The variations of EEG complexity were subsequently used to classify the preictal and interictal states of migraine [37]. In addition to complexity, other relevant biomarkers such as spectral power in different frequency bands and time-domain variance of EEG were extracted and tested using various machine learning models, such as support vector machines and neural networks [38, 39]. However, the EEG-based classification systems generally require multiple channels, which may add to the complexity of data acquisition and processing. Functional magnetic resonance imaging (fMRI) is another effective tool for migraine analysis, which was further used for classification of migraine from healthy controls [40]. However, fMRI and other neuroimaging techniques are costly and cumbersome for routine use and are currently impossible to integrate into portable devices. The limited temporal resolution of fMRI and computational overhead of image classification are the other drawbacks of this approach.

Alternatively, we propose to employ SSEP as a practical diagnostic tool and a sensitive predictor for migraine classification. Our goal is to distinguish migraine patients in different phases (interictal or ictal) from healthy controls, using state-of-the-art machine learning models and SSEP biomarkers. The proposed system was trained and validated on a dataset of 57 subjects. We further analyzed the most discriminating features for each classification task. This ap-

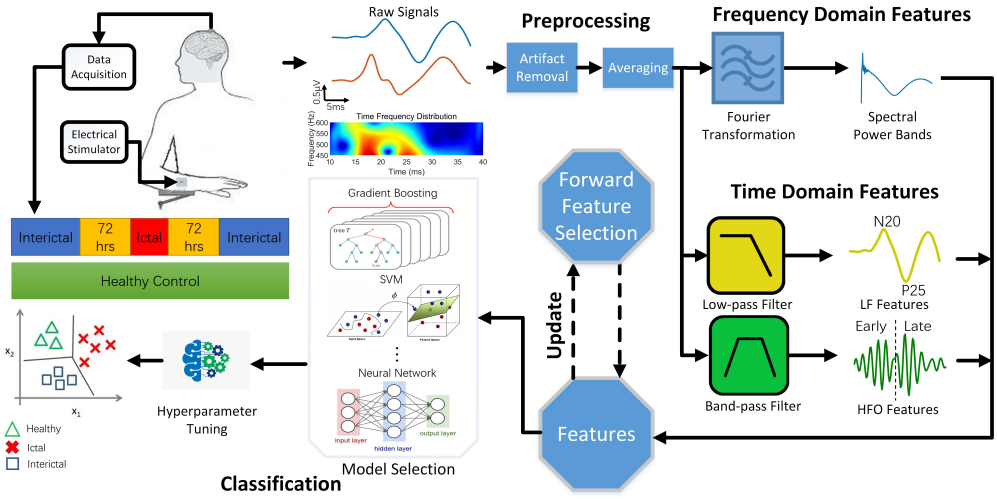

Figure 2.1: Block diagram of the proposed migraine state classification system. The SSEP signals were recorded using a CEDTM 1401 device with electrical stimulation at the wrist. After artifact removal and averaging, migraine features were extracted in both time and frequency domains. A feature selection approach was employed to find the optimal feature set for each task. Various models and classifiers were further tested and optimized to achieve the best classification results.

proach is the first step toward early detection of migraine attacks, which could be used in a personalized headache monitoring device.

2.2 Materials and methods

The block diagram of the proposed migraine classification system is shown in Fig. 2.1. The SSEP signals were recorded from healthy subjects and from migraineurs in the two phases of ictal and interictal. The raw signals are plotted as time series and their time-frequency distributions are further analyzed to find the dominant high-frequency components. From the time-frequency graph, we were able to identify two peaks in the high frequency range, which were later defined as early and late HFOs. We performed preprocessing on the raw data by removing artifacts and averaging over multiple SSEP trials. Several biomarkers

were then extracted using Fourier transform and digital filtering. The evaluated feature set is composed of spectral power in different frequency bands, low-frequency (LF) features, and HFO-related biomarkers. A wrapper-based feature selection method was used to select the most discriminative features. Several machine learning models were then tested on the resulting feature set and their parameters were carefully optimized. Finally, we compared the classifier outputs with the actual labels to calculate the prediction accuracy.

2.2.1 Participants

We initially enrolled 50 consecutive migraine patients who attended our headache clinic. Of the patients initially recorded, eight patients had an attack between 12 and 72 hours before or after the recording session, and their electrophysiological data were not included in the subsequent analysis. The final data set comprises a total number of 42 migraine patients, including 29 interictal (15 MA, 14 MO, 17 females, 12 males) and 13 ictal (four MA, nine MO, 11 females, two males). The subjects were recruited among the patients attending the headache clinic of Sapienza University of Rome. We excluded those patients who had taken medication regularly, except for the contraceptive pill. For comparison, we enrolled a group of 15 age-matched healthy volunteers recruited among medical school students and healthcare professionals. The patients and controls were examined at the same time of day, in the same laboratory and by the same investigators. Patients in the interictal state were recorded at least 72 hours before or after a migraine attack. The ictal group consisted of patients who had an interval of less than 12 hours between the recording time and a migraine attack. The study was approved by the Ethics Committee of the Faculty of Medicine, University of Rome. Informed consent was collected from all

Table 2.1: Extracted features from SSEP recordings.

Feature	Description
Early, Late HFO Amplitude	HFO maximum peak-to-peak value in the early and late bursts [32, 34, 41, 42]
Early, Late HFO RMS	HFO root-mean-square value in the early and late bursts [43, 44]
Early, Late HFO Latency	Latency of maximum peak in the early and late HFO bursts
Early, Late HFO Peak Number	Number of peaks in the early and late HFO bursts
LF N20-P25 Amplitude	Low-frequency peak-to-peak amplitude between N20 and P25 [29, 42]
N20 Latency	Latency of N20 peak [41, 45]
LF Hjo Act, Mob, Com	Hjorth activity, mobility, and complexity of the low-frequency SSEP [46, 47]
HFO Hjo Act, Mob, Com	Hjorth activity, mobility, and complexity of HFO [46, 47]
Spectral Power Features	Spectral power in ([1–30], [30–80], [80–200], [200–450], [450–750] Hz) bands [48–50]

participants in this study.

Data acquisition The SSEP signals were elicited by electrical stimulation of the median nerve and recorded using a CED 1401 device (Cambridge Electronic Design Ltd, Cambridge, UK). Subjects were asked to sit comfortably on a chair in an illuminated room and keep their eyes open, with their attention focused on the wrist movement. Electrical stimulation was applied to the right median nerve at the wrist with a constant-current square-wave pulse (0.2ms width, cathode proximal). The stimulus intensity was set at twice the motor threshold with a repetition rate of 4.4 Hz. The SSEP signals were recorded over the contralateral parietal area. The ground electrode was placed on the right arm. A CED 1902 was used to amplify the evoked potentials. The sampling frequency was set to 5000 Hz and the post-stimulus signals were recorded for a duration of 40 ms. The high sampling rate allowed us to analyze a wide range of frequency bands in the subsequent feature extraction stage.

Data analysis and feature extraction The SSEP signals were low-pass filtered at 450 Hz to extract the low-frequency (LF) components. The LF-SSEPs were then used to identify the latency of various SSEP components (N20, P25) and the associated peak-to-peak amplitude.

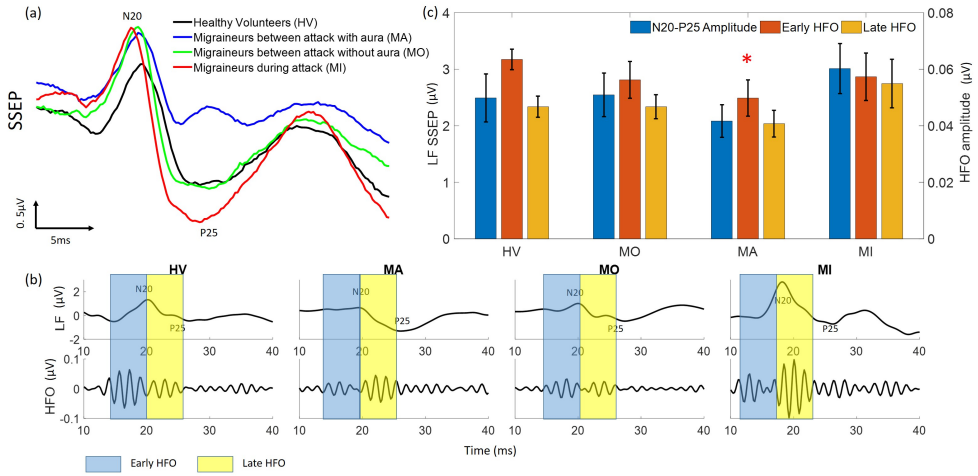

Figure 2.2: (a) Grand averaged SSEP recordings from migraine patients as well as healthy controls; (b) The amplitude changes in the N20-P25 peak-to-peak value of averaged SSEPs and the extracted HFOs from all groups; (c) Statistical analysis of N20-P25 and early/late HFO amplitude. Early HFO burst of MA group is significantly lower than healthy controls (p=0.0386). In the MI group, the low-frequency N20-P25 amplitude and late HFO bursts are higher than HV group, but it does not reach the significance level. Compared with healthy controls, the interictal MA and MO groups have reduced early HFOs and normal late HFOs, likely due to the reduced activity of sensory cortices. The early HFOs tend to normalize in the MI group, while the higher late HFOs differentiate them from healthy controls.

Prior studies have shown that high-frequency oscillations (HFOs) embedded on the parietal N20 component of SSEPs are significantly different in migraineurs between attacks, compared to healthy volunteers [32, 42, 43]. This indicates that HFO could be an important biomarker to distinguish migraineurs from healthy subjects. Here, we extracted HFOs using an FIR bandpass filter (450–750 Hz). Both low-pass and band-pass filters were designed using a Bartlett-Hanning window with a filter order of 50. Between the early and late HFO bursts, a clear change in both frequency and amplitude was observed in the majority of recordings. More specifically, the early burst was in the ascending slope of the N20 component, while the late burst occurred in the descending slope of N20, sometimes extending toward the ascending slope of N33 [33]. The

N20 peak latency was used as a cut-off point to separate early and late HFO bursts [51], while both HFOs had an approximate duration of 5 ms [44].

Figure 2.2(a) illustrates the average SSEP recordings of individuals from each group, while Fig. 2.2(b) shows the low-frequency SSEPs and the embedded high-frequency oscillations. The N20-P25 amplitudes of LF-SSEPs were found to be larger in the ictal group. The maximum peak-to-peak amplitude of the late HFO burst was different between the HV and MI groups, while MO and MA tended to be lower in the early HFOs. Considering the lack of significant difference in the HFO or LF components between the MO and MA groups, we combined them into one interictal class for the subsequent feature extraction and classification process.

The measured SSEPs were composed of 425 to 2000 independently collected trials. For the purpose of feature extraction, we averaged every 40 consecutive sweeps and labelled the resulting waveform as healthy volunteer (HV), ictal (MI), or interictal (MII), depending on the subject under study. Averaging over different numbers of trials was tested to find the value that maximized the classification accuracy. This resulted in a total of 325 MI, 534 MII, and 323 HV samples that were fed to our classifiers, following feature extraction. In the feature extraction phase, each sample was independently processed to avoid data leakage between the train and test sets.

In addition to the features of HFO and LF-SSEP described above, a number of electrophysiological measures that are widely used in EEG analysis were included, such as spectral power in various frequency bands [36, 50], and Hjorth parameters [46, 47], as listed in Table 2.1. In order to reduce the effect of power fluctuations among individuals, spectral power features were normalized to

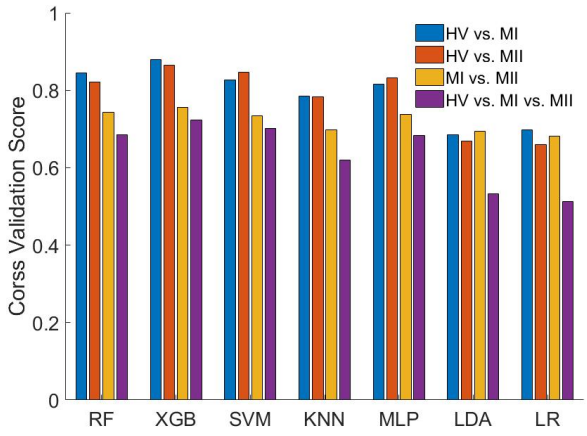

Figure 2.3: The accuracy measured by 10-fold cross validation score for different tasks and various machine learning models. The algorithms are trained and tested on the entire feature set, and the classifier settings are optimized to each task.

each sample's total power. The Hjorth parameters, known as activity, mobility, and complexity, indicate various signal properties in the time domain. Specifically, Hjorth activity is an indicator of signal variance, while mobility represents the mean frequency. Moreover, the frequency changes over a given time period can be represented by complexity.

It should be noted that compared with low-frequency SSEP, the amplitude of HFO is relatively small, making measurement noise a significant factor in HFO extraction and analysis. This is a challenge for classification tasks that employ HFOs as key characteristic features. In this study, we reduced the effect of random noise by averaging over a higher number of SSEP sweeps. However, averaging over many sweeps can reduce the number of training samples and make it impossible to reliably evaluate the classification performance. Therefore, considering the trade-off between noise and the number of samples, a 40-sweep averaging was found to be optimal, which resulted in an accuracy of 36% for HFO features and a total of 1182 training samples.

Table 2.2: The overall performance of machine learning models for different tasks measured by accuracy, sensitivity, specificity, and F_1 score. All models are trained and tested on the complete feature set. For the SVM classifier, support vectors with linear, sigmoid, and RBF kernels were tested, and SVM-RBF was chosen as it outperformed other kernels in all classification tasks.

Model	Task	Accuracy	Sensitivity	Specificity	F_1 Score	Model Parameters
XGB	HV vs. MI vs. MII	0.724	0.790*	0.916*	0.715	
Best model	HV vs. MI	0.880	0.893	0.903	0.872	Tree count: 105
	HV vs. MII	0.865	0.869	0.878	0.859	Maximum depth: 13
	MI vs. MII	0.756	0.781	0.715	0.731	
RF	HV vs. MI vs. MII	0.686	0.814*	0.875*	0.667	
p=0.0335	HV vs. MI	0.844	0.841	0.840	0.841	Tree count: 276
	HV vs. MII	0.822	0.845	0.806	0.798	Maximum depth: 10
	MI vs. MII	0.744	0.739	0.711	0.698	
SVM	HV vs. MI vs. MII	0.702	0.778*	0.884*	0.681	
p=0.1176	HV vs. MI	0.826	0.838	0.814	0.825	RBF kernel
	HV vs. MII	0.846	0.857	0.858	0.830	
	MI vs. MII	0.735	0.742	0.752	0.689	
KNN	HV vs. MI vs. MII	0.619	0.595*	0.859*	0.601	
p=1e-5	HV vs. MI	0.785	0.780	0.782	0.785	Number of
	HV vs. MII	0.783	0.767	0.793	0.759	neighbors: 5
	MI vs. MII	0.697	0.727	0.612	0.662	
MLP	HV vs. MI vs. MII	0.684	0.734*	0.888*	0.661	
p=0.0003	HV vs. MI	0.815	0.799	0.796	0.822	Hidden layer size:
	HV vs. MII	0.833	0.826	0.852	0.809	200
	MI vs. MII	0.738	0.761	0.662	0.723	
LDA	HV vs. MI vs. MII	0.532	0.528*	0.772*	0.472	
p=4e-5	HV vs. MI	0.685	0.692	0.702	0.684	Least squares solution with auto shrinkage
	HV vs. MII	0.668	0.609	0.685	0.597	
	MI vs. MII	0.695	0.693	0.671	0.630	
LR	HV vs. MI vs. MII	0.512	0.510*	0.774*	0.461	
p=4e-6	HV vs. MI	0.697	0.689	0.706	0.696	Default settings
	HV vs. MII	0.660	0.584	0.696	0.604	
	MI vs. MII	0.681	0.696	0.639	0.622	

2.3 Classification

In order to classify the SSEP recordings, we explored both handcrafted feature methods, where the domain knowledge is used to extract application-specific features (i.e. migraine biomarkers), and deep learning models such as convolutional neural network (CNN) that do not require prior knowledge and domain expertise. The classification performance was measured by a 10-fold cross-validation method for the prediction metrics of accuracy, F1 score, sensitivity and specificity, and was compared to prior works. For the handcrafted feature approach, the most critical features in the classification process were determined using a forward feature selection method, and compared with previous clinical findings.

2.3.1 Handcrafted feature methods

In machine learning based on feature engineering, extracting informative, independent, and domain-specific characteristics from data is critical. In this work, we selected a number of discriminating features for migraine based on previous SSEP studies, as described in the previous section. These features (listed in Table 2.1) form a numerical representation of the disease under study, called a feature space. Then, machine learning models were built on these features to classify the data. The effectiveness of features in migraine detection is represented by the performance of the classification algorithms that utilize them.

Our target classification problem was to distinguish between three groups of subjects using SSEP features: Migraine interictal (MII), migraine ictal (MI), and healthy (HV). We tested several state-of-the-art classifiers to achieve an optimal accuracy. These classifiers were optimized and parameter tuned in scikit-learn (scikit-learn.org) and include: Random forest (RF), extreme gradient-boosting trees (XGB), support vector machines (SVMs) with various kernels, K-nearest neighbors (KNN), mutilayer perceptron (MLP), linear discriminant analysis (LDA), and logistic regression (LR). We trained these supervised learning models on a set of labelled SSEP features and the performance of trained models was then evaluated on a test feature set with unknown labels. We compared the outputs of classifiers with the actual labels to assess the prediction accuracy. The performance of different classifiers was compared to find the optimal model. We further evaluated the three-class performance (i.e. MI-MII-HV) and compared it with the binary classification among every two groups of subjects (i.e. MI-HV, MII-HV, and MI-MII). As expected, the case of binary classification results in a higher accuracy. For the actual implementation of a migraine attack detection device, the transition from healthy to interictal and from inter-

ictal to ictal are particularly important. This could potentially be monitored in real time, by training the classifier on the previously recorded SSEP data from a patient.

2.3.2 Convolutional neural network

A challenge in neural data classification is to identify relevant features (i.e. biomarkers) that are significantly different among various states of a disease, or a cognitive or motor task. A patient-specific selection of features may be necessary to accommodate the variability of neural data among subjects. For example, in epileptic seizure detection, patient-specific features are widely used [52]. Alternatively, the convolutional neural network is a typical deep learning model that does not rely on handcrafted features. For neural signals, the one-dimensional CNN (1D CNN) has been tested in brain-computer interface

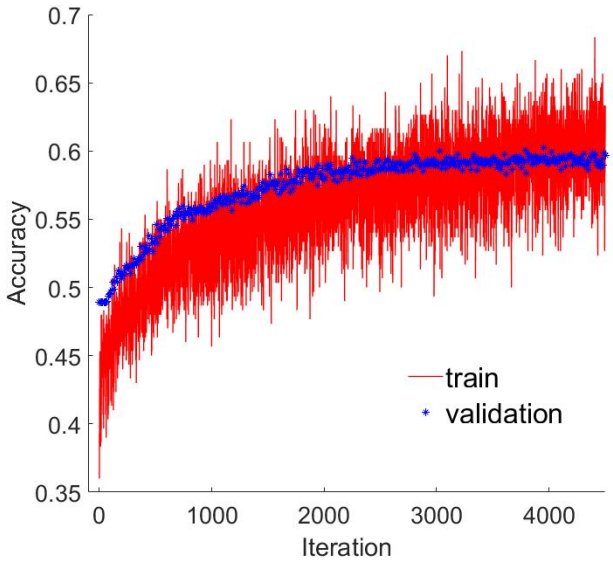

Figure 2.4: The train and test accuracies of the 1D-CNN model for the HV-MI-MII classification task. The cross validation score for test data converges to 56.3%, which is lower than most studied feature-based models, yet better than linear models.

(BCI) and seizure detection applications [53–55]. Unlike handcrafted feature models, a CNN can be used to automatically learn the critical features from raw data and subsequently classify it. The independence from prior disease-based knowledge and human effort for feature design is a major advantage. Therefore, these models may have the potential to succeed in cases where sufficient discriminating features are not available.

We implemented a CNN based on the AlexNet architecture in TensorFlow [56, 57], which is composed of five convolution layers, five max pooling layers, and a fully connected layer. We preprocessed the data by subtracting the mean and dividing it by the variance. The inputs to the model are the standardized SSEP signals after stimulation onset, and during the time segment of $10\,\text{ms} \leq t \leq 40\,\text{ms}$. In each iteration, a batch of 100 samples is fed to the network and the associated weights are updated via backpropagation. We ran thousands of iterations until the cross-validation score converged to a stable value. This score is used as an estimate of accuracy and is compared with the handcrafted feature methods for model selection.

2.4 Results

2.4.1 Model performance

In order to evaluate the classification performance, we measured the 10-fold cros-validation scores (averaged through repeated iterations) for all the studied models (Fig. 2.3), in which varying levels of accuracy were obtained for different machine learning models and on different tasks. Among the handcrafted feature methods, XGB achieves the best three-class discrimination accuracy of 72.4%. It also achieves an accuracy of 88.0% for the binary classification task of

HV-MI, 86.5% for HV-MII, and 75.6% for MI-MII, respectively. To better assess the model performance, we separately looked into false positives (the number of healthy samples classified as migraine) and false negatives (the number of migraine samples classified as healthy), by reporting the specificity (true negative rate) and sensitivity (true positive rate) of classifiers. These criteria imply different clinical priorities in practice; for example, in a migraine or seizure detection system, we may prefer to raise a false alarm rather than missing an imminent seizure or migraine attack [58]. Table 2.2 presents a comparison of performance for various machine learning models and each classification task. XGB outperforms other classifiers not only in terms of accuracy, but also sensitivity, specificity, and F1 score.

A hyperparameter tuning of classifier parameters was performed to achieve the optimal settings that lead to the highest accuracy. In order to fairly compare the performance of classifiers and select the best model, we applied one-way analysis of variance (ANOVA) between XGB and any other model. The significance level was set to $p < 0.05$, indicating that the two models perform significantly differently for a given classification task. From the statistical analysis presented in Table 2.2, we can conclude that XGB performs significantly better than RF, KNN, MLP, LDA, and LR. Compared with SVM-RBF, XGB achieves higher scores in all the metrics shown in Table 2.2, even though it does not reach the significance level.

We further examined the CNN to classify migraine states, but the performance was not satisfactory. This is likely due to the limited amount of training data (Fig. 2.4). In general, while deep neural networks achieve state-of-the-art accuracy in most learning tasks that involve large datasets of unstructured

data, the application of such techniques may not be beneficial in problems with limited training sets or under domain-specific test time constraints [59]. With a cross-validation score of 56.3%, CNN is inferior to most handcrafted feature methods in this study, yet outperforms the linear models such as LR and LDA. The XGB classifier is the winning model in our study and is developed based on the gradient boosting technique [60, 61]. Ensembles of decision trees such as gradient boosting and random forests have been among the most competitive methods in machine learning recently [59], particularly in a regime of limited training data and little need for parameter tuning. In particular, the XGB implementation has been a winning solution in many machine learning competitions, such as the intracranial EEG-based seizure prediction contest on Kaggle, and has been included in our analysis. Gradient boosting exploits gradient-based optimization and boosting by adaptively combining many simple models (in this case, binary split decision trees) to get improved predictive performance. To the best of our knowledge, this is the first study to explore XGB in migraine signal analysis.

2.4.2 Feature selection

In machine learning, classification with fewer attributes has generally been favored, as it reduces the number of redundant (i.e. highly correlated) or marginally relevant features, in addition to reducing the computation time and hardware complexity. Feature selection and dimensionality reduction are the two widely used methods for reducing the number of attributes. Dimensionality reduction is commonly achieved by obtaining a set of principal components of features, whereas feature selection methods choose the most discriminating attributes without modifying them. Principal component analysis (PCA) is a

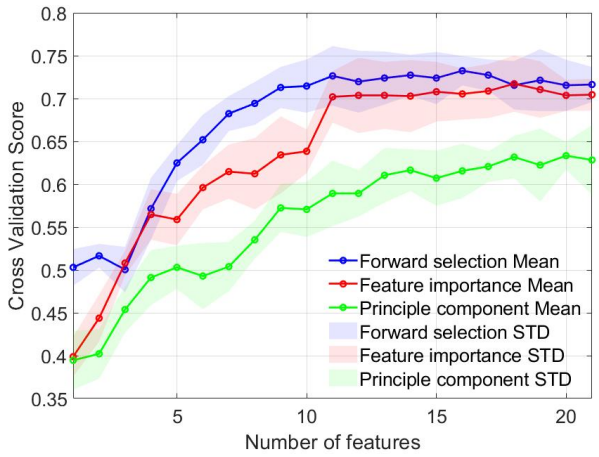

Figure 2.5: The classification performance for HV-MI-MII versus number of features for three different feature selection methods. The mean accuracy (Mean) and standard deviation (STD) are shown as solid line and shaded areas, respectively. The forward feature selection (FFS) approach outperforms the feature importance and PCA methods.

common approach for dimensionality reduction and has been widely used in EEG preprocessing and classification [62]. However, due to its unsupervised nature, the performance of PCA in supervised learning tasks may be suboptimal. In contrast, feature selection methods select a subset of relevant features

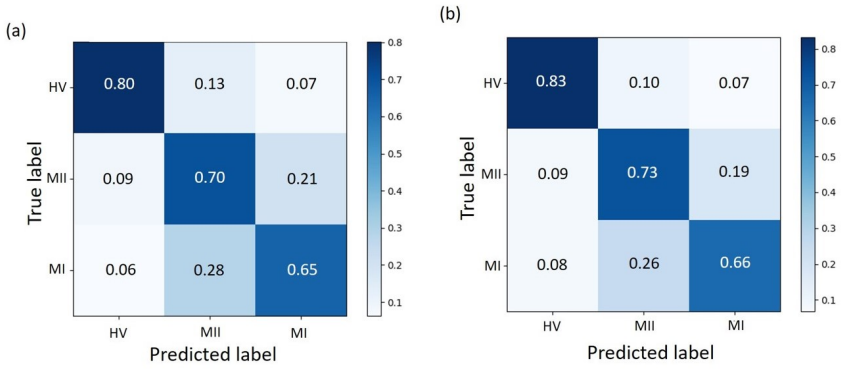

Figure 2.6: (a) The original confusion matrix; (b) Confusion matrix after feature selection. Following forward feature selection, the diagonal elements of the matrix grow larger, indicating that more samples are correctly classified. This implies that FFS effectively boosts the system performance in migraine states classification tasks, while reducing the number of redundant features.

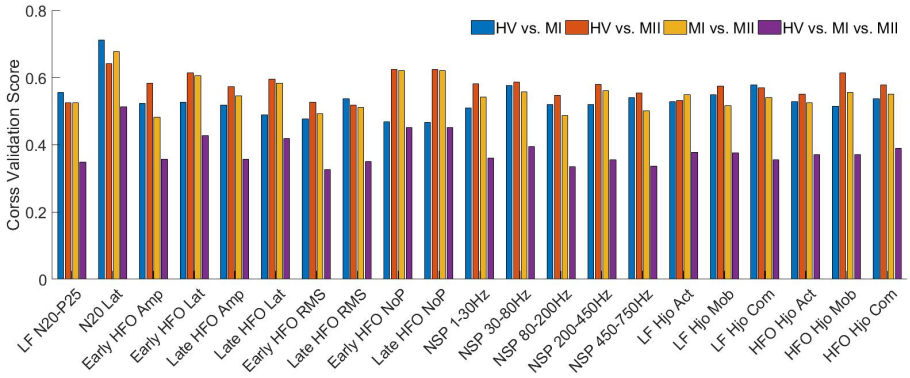

Figure 2.7: The single-feature accuracies of the entire feature set. These scores represent a measure of effectiveness of individual features in the classification task. Lat: latency; Amp: amplitude; NoP: number of peaks; NSP: normalized spectral power.

for model construction, which aids in building an accurate predictive model while reducing the number of features.

Given the relatively large number of examined features in the current study, we used a forward feature selection (FFS) method to further enhance the classification accuracy and remove the redundant features. The algorithm starts by evaluating all feature subsets which consist of only one input attribute. In each iteration, the combination of prior subset and a new feature from the pool of remaining features is exhaustively explored and evaluated with a cross-validation score. The algorithm then continues to add the best feature in each iteration and update the combinational subset until the entire feature set is analyzed [63].

In this study, we compared the forward feature selection method with the commonly used PCA and feature importance-based selection methods. The latter approach relies on how useful each feature is in the construction of the classification model (generally decision trees). As shown in Fig. 2.5, the FFS outperforms other methods in terms of both accuracy and convergence rate. Using FFS, we selected an optimal subset consisting of 16 features and achieved an

accuracy of 73.3% for the three-class detection task. For the binary classification tasks, an accuracy of 89.7% was achieved for HV-MI (by selecting 11 features), 88.7% for HV-MII (14 features), and 80.2% for MI-MII (12 features), respectively. To better illustrate this, Fig. 2.6 depicts the confusion matrices prior to and following forward feature selection. The confusion matrix reports the percentage of successfully classified or misclassified samples. We can see that following feature selection, the sensitivity and specificity scores have significantly improved. Overall, this method results in an average improvement of 2.4% in classification accuracy, and a 37.0% reduction in the number of extracted features.

2.5 Discussion

2.5.1 Migraine biomarkers

To better assess the importance of various SSEP correlates of migraine in the proposed classification system, the top three discriminating features based on the FFS method are listed in Table 2.3. For the HV-MI detection task, the LF N20-P25 amplitude and RMS value of late HFO stand out, which is consistent with prior findings [33, 44]. Our study confirms that migraineurs in the ictal state have higher N20-P25 amplitude and late HFO burst [33]. The HFO mobility characterizes healthy versus migraine interictal, indicating that the mean frequency of the power spectrum in the HFO band differs between the two groups. For the case of MI-MII, both early and late HFO bursts play a key role. The other impor-

Table 2.3: The top performing features based on FFS method.

Classification Task	Top Features
HV vs. MI vs. MII	N20 Latency, Late HFO RMS, Spectral Power in [450–750] Hz.
HV vs. MI	LF N20-P25 Amplitude, N20 Latency, Late HFO RMS.
HV vs. MII	N20 Latency, Spectral Power in [80–200] Hz, HFO Hjo Mob.
MI vs. MII	N20 Latency, Late HFO Latency, Early HFO Peak Numbers.

tant feature is the latency of the N20 peak. While the N20 latency is not widely covered in migraine literature, some studies report that it changes during migraine treatment [45]. This motivates the use of N20 latency in our migraine classification system. Besides, the N20 latency is largely stable over time, even when averaged over a relatively small number of SSEP recordings. Based on our results, the N20 latency might be a potentially prominent feature for migraine classification. We further observe that the broadband gamma (80–200 Hz) as well as HFO (450–750 Hz) frequency bands are important for HV-MII and HV-MI-MII classification tasks, respectively.

The single-feature classification accuracy for the studied features is illustrated in Fig. 2.7, using a 10-fold cross validation score and XGB classifier. The single-feature accuracy is obtained in the first step of the feature selection routine and could be used to estimate the feature importance. However, combining the top performing features based on single-feature accuracy does not necessarily lead to the most optimal feature set. In contrast, the employed feature selection method accounts for the mutual information and correlation among features to find the optimal subset. For instance, there is a high correlation among features that pertain to similar aspects of a signal, such as HFO amplitude and RMS. Therefore, the forward selection method is preferred to the single-feature importance method to build a subset of discriminating features.

For migraineurs during attack, prior studies report that the N20-P25 amplitude of LF-SSEP is higher than healthy controls [33, 64]. This is confirmed by the superior performance of the N20-P25 feature in MI-HV classification. Moreover, the primary cortical activation represented by late HFOs is shown to increase in the ictal state [33]. We also observe that the late HFO RMS is a powerful feature

Table 2.4: Performance summary and comparison with published works on migraine classification.

	Ko *et al.*[66]	Cao *et al.*[37]	Chong *et al.*[40]	This work
Subjects	29 patients, 9 HV	40 patients, 40 HV	58 patients, 50 HV	42 patients, 15 HV
Input Signal	Steady-State VEP	Resting EEG	Resting fMRI	SSEP
Channel Count	19	4	–	Single
Classifier	KNN	SVM-RBF	–	XGB
Classification Task	Preictal-MI-Postictal	MII-Preictal-HV	HV-Patients	HV-MI, HV-MII, MI-MII, HV-MI-MII
Accuracy	0.73	0.76	0.861	0.897, 0.887, 0.802, 0.733

in the HV-MI classification task. On the other hand, the reduced amplitude of the early HFO burst reflects a decrease in somatosensory thalamo-cortical activity [65]. It is likely that the reduction of pre-activation level in sensory cortices leads to habituation deficit in the migraine interictal state [32]. This could possibly explain the high single-feature accuracy of early HFO features in the HV-MII classification task. Another finding in Table 2.3 is that while the interictal group exhibits reduced early HFOs, the amplitude of early HFO is not a key indicator of the migraine ictal group. This is further verified by the single-feature performance shown in Fig. 2.7. In other words, the features of early HFO are more effective in separating the interictal group from healthy controls. This is due to the fact that the thalamo-cortical activation reflected by early HFOs starts to normalize during the ictal state. Yet, the MI group has relatively larger late HFOs compared to the HV and MII groups.

2.5.2 Comparison to prior works

In Table 2.4, we summarize the performance of the proposed method and compare it to prior works on classification of migraine phases. To the best of our knowledge, this is the first SSEP-based migraine state classification system. As shown in this table, various other signal modalities have been previously explored for detection of migraine states. The resting fMRI in ictal migraineurs

and healthy controls was recently studied [40]. Compared with fMRI, SSEPs are easier to collect and analyze in real time. Alternatively, resting-state EEGs and visual evoked potentials (VEPs) have been previously used for migraine detection [37, 66]. However, these studies rely on multiple channels of input data, which can hinder their integration into a low-cost and portable monitoring device. In this work, the classification tasks are performed using a single-channel SSEP signal, making the proposed system more effective and convenient for patient use. We attain a relatively high accuracy of above 88% in migraine ictal or interictal versus healthy discrimination (HV-MI, HV-MII), and above 80% in classification of two migraine states (MI-MII). Therefore, the proposed algorithm promises the potential to predict migraine attacks in advance, making early intervention possible for more effective migraine treatment.

2.5.3 Limitations

Despite promising results, our current study has several limitations. While some of the analyzed biomarkers are supported by prior studies [32, 33, 43], the reliability of classifiers and statistical significance of the achieved results need to be verified on more patients and long-term recordings. Limited by the availability of data, we were not able to separately study the migraine preictal and postictal states, which could be important in migraine attack prediction. Therefore, the proposed classification system should be tested on more patients and various migraine phases in future.

In addition, the individual differences among subjects might affect our analysis and similar studies that are performed across subjects. For example, N20 latency was reported to correlate with height and brain size [67], which are not included in our study. A single-subject analysis of different phases of migraine

is preferred to design a personalized migraine classification system. Here, we tried to minimize the impact of individual differences by using normalized spectral power features and standardized SSEPs. Given that the top-performing features are well consistent with published works, we may conclude that individual differences do not play a major role in our analysis. Most importantly, while this study and previous works in Table 2.4 allow us to assess the separability of various migraine states among individuals, the practical design of a headache monitoring system requires the use of continuous data from a single subject. In our future work, we will test a patient-specific classification approach on the long-term and multistate data from individual patients in order to predict the transition from one state of migraine to another. This would be similar to the patient-specific seizure monitoring systems for medication-resistant epilepsy [68].

The biomarkers discovered by this work have been implemented on hardware for multiclass migraine detection. Please refer to [69, 70] for more details.

2.6 Conclusion

In this study, we proposed a new SSEP-based system for migraine classification. Machine learning approaches were combined with an efficient feature selection method, not only to achieve a decent classification performance, but also to provide a systematic solution for feature importance assessment. Overall, we were able to achieve over 88% accuracy in migraine ictal or interictal versus healthy detection. We tested a recently developed and highly competitive class of boosting algorithms to classify migraineurs in ictal and interictal states. This XGB framework outperforms most widely used models in our analysis. Fur-

thermore, the proper selection of features can reduce the computational complexity by 37% on average, while improving the classification accuracy by 2.4%. From a clinical perspective, the study of discriminating SSEP biomarkers and their correlation with migraine phases may further reveal the potential mechanisms underlying migraine attacks. The proposed system could be used as a noninvasive headache monitoring system for early diagnosis and treatment of underlying migraine headaches.

CHAPTER 3

RESOT: RESOURCE-EFFICIENT OBLIQUE TREES FOR NEURAL SIGNAL CLASSIFICATION

3.1 Introduction

Recently, the use of machine learning (ML) techniques has been extended to emerging challenges in neural data processing, such as early symptom detection [2, 11, 12, 71–73], brain image classification [74], and motor decoding [75, 76]. With the help of domain-specific biomarkers, ML models have been used to classify neurophysiological signals with limited training sets [2, 11, 12, 71–73, 75–78] —typically recorded through invasive or non-invasive electrodes— while outperforming other conventional methods. However, although modern machine learning tools have shown promise in neural signal classification, their deployment on high-channel-count neural interfaces remains a challenge, given the tight power budget and stringent area and memory constraints for such implants. The alternative approach that consists in transmitting the extracted features from neural channels for off-the-body classification [16, 17], has several drawbacks. First, similar to raw data transmission, this approach suffers from security and privacy concerns due to transmitting patients' private data to external servers for processing. Second, while power demands for telemetry may be relaxed due to lower-dimension feature transmission, the high loop latency could be problematic for real-time and closed-loop operation of implantable devices (e.g., for closed-loop activation of a therapeutic or sensory stimulation in the brain). Making local predictions, on the other hand, could enable the device to work everywhere irrespective of connectivity to external units, and decisions could be made more quickly [12, 79, 80], with no need for raw data or feature

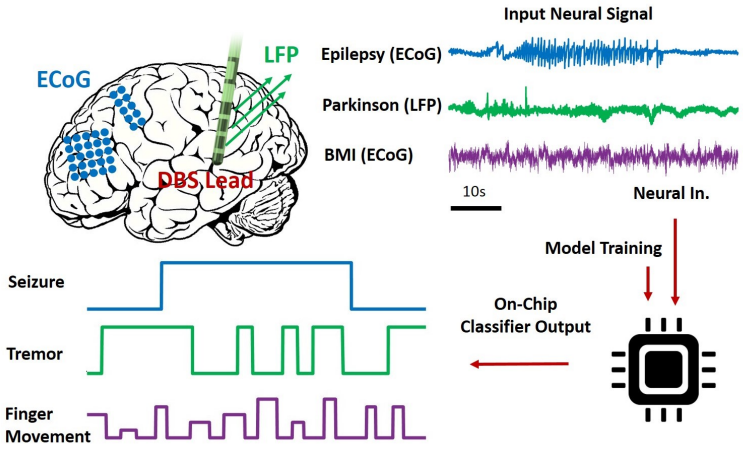

Figure 3.1: On-chip classification of neural signals for epileptic seizure detection, Parkinsonian tremor detection, or finger movement decoding. Neural signals are recorded from different regions of the brain. The machine learning model is trained offline and the parameters are stored on-chip. Decisions are made in real-time by the on-chip classifier, to predict a disease or classify a movement.
streaming.

Among the widely-used ML algorithms for neural signal classification (e.g., logistic regression, support vector machines, k-nearest neighbours, neural networks, and decision trees), the latter is compatible with a lightweight, 'on-demand' feature extraction framework, recently explored in [12]. A test sample travels through a single root-to-leaf path during inference, thus visiting only a small proportion of the entire model [12, 81]. The lightweight feature extraction capability of decision tree (DT) is crucial, considering the large number of channels and predictive features in applications such as implantable seizure detection. In addition, according to recent studies [79, 82], the model size of DTs can be largely compressed to operate under extreme memory constraints. Given their lightweight inference and small model size, tree-based models can be integrated on chip with state-of-the-art energy and area efficiency [12], and are therefore favorable for neural and brain-machine interface applications. Fur-

thermore, through techniques such as gradient boosting [83], decision tree ensembles have achieved a high accuracy in classifying time-series neurophysiological data [2, 12, 21, 76, 77], typically outperforming the neural network-based models [84]. We previously introduced a cost-efficient classification approach to further reduce the inference overhead of DTs, by adding the feature cost (e.g., power dissipation) as a regularization term to the objective function [85]. In this cost-efficient learning scheme, each feature is associated with a hardware cost and the model is trained to prioritize the lower-cost features. This led to a reduction of power dissipation by more than half for both seizure and tremor detection tasks, with only a marginal loss in performance (0.9%) [85].

However, building optimal binary DTs is essentially an NP-hard problem [86] and many approaches were recently proposed to optimize the tree structure [87, 88]. Unlike conventional axis-parallel trees that hold deterministic decision functions, probabilistic (i.e., soft) trees with oblique boundaries hold a probability decision leading to the left or right child. Inspired by back-propagation neural networks, such probabilistic trees with stochastic routing are compatible with gradient-based optimization [89]. As a result, one may effectively employ various model compression techniques such as fixed-point quantization [90], weight pruning, and sharing [91] that are widely used in hardware implementation of deep neural networks (DNNs). Moreover, these soft DTs still enable a lightweight inference, by following the most probable path along the tree.

Here, we propose a framework based on soft oblique trees, by coupling neural networks with decision trees. Therefore, we can exploit the benefits of both models and compress an oblique tree (OT) with similar techniques as employed in DNN architectures. In addition, we extend our cost-efficient ap-

proach [85] to soft oblique trees, as a promising alternative to conventional axis-aligned DTs. With these techniques, we demonstrate the performance of our single cost-aware oblique tree on several neural classification problems including seizure, tremor, and finger movement detection tasks (Fig. 3.1), and benchmark it against state-of-the-art models.

Table 3.1: Information on Patients and Neural Recordings

Epilepsy iEEG Portal ID	**# of Channels/ Sample Rate (Hz)**	**# of Seizures**	**Recording Duration**
I001_P034_D01	47/5000	16	1d8h
Study 004-2	56/500	3	7d18h
Study 022	56/500	7	3d23h
Study 024	88/500	19	8d10h
Study 026	96/500	22	3d3h
Study 029	64/500	3	5d1h
Study 030	64/500	8	5d23h
Study 033	128/500	17	6d17h
Study 037	80/500	8	8d23h
Study 038	88/500	10	3d0h
Parkinson Recording Index	**# of Channels/ Sample Rate (Hz)**	**Recording Side**	**Duration (min)/ Tremor Prevalence (%)**
1	4/2048	R	5.7/81.0
2	4/2048	L	8.3/48.9
3	4/2048	R	6.6/40.9
4	4/2048	L	6.1/95.7
5	4/2048	L	4.9/45.3
6	4/2048	L	5.5/83.4
7	4/2048	R	9.6/52.3
8	4/2048	L	10.0/89.5
9	4/2048	R	10.0/94.0
10	4/2048	L	1.5/53.0
11	4/2048	R	4.5/79.4
12	4/2048	R	6.6/49.3
13	4/2048	L	4.8/96.5
14	4/2048	L	5.9/94.8
15	4/2048	R	5.0/78.0
16	4/2048	L	4.7/83.2
Finger Movement Subject Index	**# of Channels/ Sample Rate (Hz)**	**Recording Side**	**Array Location**
1	46/1000	L	Fronto-Parietal
2	63/1000	R	Fronto-Temporal
3	61/1000	L	Fronto-Temporal-Parietal
4	58/1000	L	Fronto-Temporal
5	64/1000	L	Parietal-Temporal-Occipital
6	43/1000	L	Fronto-Temporal
7	64/1000	L	Fronto-Temporal-Parietal
8	38/1000	R	Fronto-Parietal
9	47/1000	L	Frontal

3.2 Neural classification tasks & data description

In this work, our focus is on algorithm development and hardware-algorithm co-optimization of resource-efficient oblique trees (ResOT) as a promising approach for neural signal classification. To show the broad application and effectiveness of the proposed model, we evaluate this approach on three implantable neural applications described below, including epileptic seizure detection, Parkinsonian tremor detection, and finger movement classification, as depicted in the general block diagram of Fig. 3.1.

3.2.1 Seizure detection

Our first target application is seizure detection for medically refractory epilepsy, using continuous neural recordings from human subjects. Seizure detection is a binary supervised classification problem with the aim of classifying between seizure and non-seizure states of a patient. We applied our model to the intracranial EEG (iEEG) recordings (shown as ECoG on Fig. 3.1) from patients with epilepsy, publicly available on the iEEG portal [92]. The dataset includes the iEEG recordings from 10 patients with at least 1 day of uninterrupted recording and 3 seizure events. A total number of 113 seizure events were annotated by expert neurologists, as detailed in Table. 3.1. The original recordings were segmented into 1-second windows labeled as *seizure* or *non-seizure*.

3.2.2 Tremor detection

In a second study, we analyzed 16 local field potential (LFP) recordings from 12 patients with Parkinson's disease (PD) who were implanted with 4-channel deep-brain stimulation (DBS) lead(s), as described in [2, 93]. The statistics of this

dataset are summarized in Table. 3.1. The LFP signals were recorded from the subthalamic nucleus (STN) region at a 2048 Hz sampling rate. We labeled the LFP recordings as *tremor* or *non-tremor* based on the simultaneous acceleration measurements. Our ML model was trained to differentiate between *tremor* and *non-tremor* states, using 3 bipolar LFP channels. The patients were recruited from the University of Oxford and gave informed consent to participate in the study that was approved by the local research ethics committee [2].

3.2.3 Finger movement classification

Our third study was focused on a finger movement classification task for brain-machine interface (BMI) application, using the electrocorticography (ECoG) data from 9 subjects sampled at 1kHz (Table. 3.1) [94]. During experiments, subjects were asked to move one of their fingers for 2s, as instructed on a monitor. Overall, each subject performed 30 trials per finger. The finger movement was captured by a data-glove at 25 Hz. All patients participated in a purely voluntary manner after providing informed consent, under experimental protocols approved by the Institutional Review Board of the University of Washington (#12193) [94]. Unlike seizure and tremor detection tasks, the finger movement detection is a 6-class problem. The labels for this study were defined as *thumb, index, middle, ring, and little finger movement* plus a *rest state*.

3.3 Model description and related work

The majority of current on-chip classifiers for neural signal analysis is based on support vector machines (SVMs). The EEG-based embedded seizure detectors in [11, 26, 95] achieved energy efficiencies of 2.03, 1.85, and 273 μJ/class, respectively, using SVM classifiers. An incremental-precision algorithm was proposed

to reduce the energy consumption for on-chip seizure detection [96], replacing the complex SVM with logistic regression. Compared to SVM, DTs also offer a lightweight inference and can improve the energy efficiency for implantable applications, where resource constraints are more critical than EEG-based wearable systems. We recently improved the energy efficiency for on-chip seizure detection to 41.2 nJ/class [12], using an ensemble of eight gradient-boosted trees that required 1kB of memory to store model parameters.

While previous SoCs integrate a small number of trees (e.g., <10 for seizure detection [12], one for voice activity detection [97]), larger ensembles may be necessary for reliable detection of more complex symptoms (e.g., 30 trees for tremor detection [2, 93], 100 for finger movement classification [76], and 105 for migraine ictal vs. interictal detection [77]). However, as shown in [12], the power consumption and area of the classifier could linearly scale with the number of trees [12].

To address this challenge and reduce the number of correlated trees, we propose to employ oblique decision trees in our model, as depicted in Fig. 3.2. This enables us to further improve the memory, power efficiency, and scalability of our classifier. Oblique trees require fewer splits and learn through powerful nodes that employ more than one attribute to conduct a split, resulting in reduced number of trees. Such oblique nodes are effective in separating the highly-correlated features in our neural processing tasks and impose a modest cost in hardware, as later discussed in this chapter. In addition, through gradient-based training as in neural networks, we are able to employ weight pruning and sharing techniques to compress the model and generate resource-efficient oblique trees. In this chapter, we benchmark the ResOT ap-

proach against boosted tree ensembles [12, 85], as well as recent models based on sparse oblique trees, such as TAO [82] and Bonsai [79].

3.3.1 Neural biomarker extraction

Unlike end-to-end learning approaches such as DNN, feature-engineered models rely on hand-crafted features and generally obtain a superior performance on small datasets [2, 77]. Here, we extract a set of predictive biomarkers for each task, followed by ResOT training and classification in a supervised manner.

Based on prior studies on the informative biomarkers of EEG/iEEG to predict seizures, we extracted the following features for our epilepsy task [12]: line-length (LLN), total power (Pow), variance (Var), and band power over delta (δ: 1–4 Hz), theta (θ: 4–8 Hz), alpha (α: 8–13 Hz), beta (β: 13–30 Hz), low-gamma (γ_1: 30–50 Hz), gamma (γ_2: 50–80 Hz), high-gamma (γ_3: 80–150 Hz), ripple (R: 150–250 Hz) and fast ripple (FR: 250–600 Hz) bands, where fast ripples are only extracted from iEEG with 5 kHz sampling rate.

For the second task, we used a set of predictive biomarkers of tremor in LFP, based on our recent study on Parkinson's disease [2, 93]: the power of beta, gamma, and high-frequency oscillation (HFO) in several sub-bands (β_1: 13–20 Hz, β_2: 20–30 Hz), (γ_1: 30–45 Hz, γ_2: 60–90 Hz, γ_3: 100–200 Hz), (HFO_1: 200–300 Hz, HFO_2: 300–400 Hz), the power ratio between low and high HFO (HFO_R), tremor power (TPow), and Hjorth parameters [46]. The Hjorth activity represents the signal variance, Hjorth mobility indicates the mean frequency, and the rate of frequency changes is measured by Hjorth complexity [46].

For finger movement classification task, we computed the power of ECoG

Table 3.2: Computed Features & Normalized Power Cost

Epilepsy	**Description**	**Power**
Line-Length (LLN)	$\frac{1}{d}\sum_d \lvert x[n] - x[n-1]\rvert$, d =window size	1
Power (Pow)	$\frac{1}{d}\sum_d x[n]^2$	1.87
Variance (Var)	$\frac{1}{d}\sum_d (x[n]-\mu)^2$, $\mu = \frac{1}{d}\sum_d x[n]$	2.93
Delta (δ)	Band power in 1–4 Hz	34.07
Theta (θ)	Band power in 4–8 Hz	34.07
Alpha (α)	Band power in 8–13 Hz	34.07
Beta (β)	Band power in 13–30 Hz	34.07
Low-Gamma (γ_1)	Band power in 30–50 Hz	34.07
Gamma (γ_2)	Band power in 50–80 Hz	34.07
High-Gamma (γ_3)	Band power in 80–150 Hz	34.07
Ripple (R)	Band power in 150–250 Hz	34.07
Fast Ripple (FR)	Band power in 250–600 Hz (at 5 kHz)	34.07
Parkinson	**Description**	**Power**
Low-Beta (β_1)	Band power in 13–20 Hz	34.07
High-Beta (β_2)	Band power in 20–30 Hz	34.07
Low-Gamma (γ_1)	Band power in 30–45 Hz	34.07
Gamma (γ_2)	Band power in 60–90 Hz	34.07
High-Gamma (γ_3)	Band power in 100–200 Hz	34.07
Low-HFO (HFO_1)	Band power in 200–300 Hz	34.07
High-HFO (HFO_2)	Band power in 300–400 Hz	34.07
HFO Ratio (HFO_R)	Low-HFO to High-HFO ratio	68.15
Tremor Power (TPow)	Band power in 3–7 Hz	34.07
Hjorth Activity (Act)	$\frac{1}{d}\sum_d (x[n]-\mu)^2$, $\mu = \frac{1}{d}\sum_d x[n]$	2.93
Hjorth Mobility (Mob)	$\sqrt{\frac{\mathrm{Var}(x[n]-x[n-1])}{\mathrm{Var}(x[n])}}$	6.26
Hjorth Complexity (Com)	$\frac{\mathrm{Mob}(x[n]-x[n-1])}{\mathrm{Mob}(x[n])}$	9.62
Finger Movement	**Description**	**Power**
Alpha (α)	Band power in 8–13 Hz	34.07
Beta (β)	Band power in 13–30 Hz	34.07
Low-Gamma (γ_1)	Band power in 30–60 Hz	34.07
Gamma (γ_2)	Band power in 60–100 Hz	34.07
High-Gamma (γ_3)	Band power in 100–200 Hz	34.07
Local Motor Potential (LMP)	$\frac{1}{d}\sum_d x[n]$	0.50
Hjorth Activity (Act)	$\frac{1}{d}\sum_d (x[n]-\mu)^2$, $\mu = \frac{1}{d}\sum_d x[n]$	2.93
Hjorth Mobility (Mob)	$\sqrt{\frac{\mathrm{Var}(x[n]-x[n-1])}{\mathrm{Var}(x[n])}}$	6.26
Hjorth Complexity (Com)	$\frac{\mathrm{Mob}(x[n]-x[n-1])}{\mathrm{Mob}(x[n])}$	9.62

over alpha (α: 8–13 Hz), beta (β: 13–30 Hz), low-gamma (γ_1: 30–60 Hz), gamma (γ_2: 60–100 Hz) and high-gamma (γ_3, 100–200 Hz) bands, local motor potential (LMP) as the moving average of raw ECoG signal, and the Hjorth parameters (Act, Mob, Com) [76]. A brief mathematical description of these features is given in Table. 3.2.

3.3.2 Feature cost estimation

To implement a cost-aware ML model, we first analyzed the hardware cost associated with different features, by simulating the power consumption for each individual feature. A standard digital implementation with 1.2V supply was used for circuit simulations. To extract band power features, FIR filters with 30 taps, 8-bit coefficients, and a parallel architecture were implemented. We have previously shown that 30 taps is a reasonable choice for FIR filters, considering the trade-off between hardware complexity and classification accuracy [98]. The design was synthesized in a 65nm TSMC LP process and the power consumption (post place and route) is reported in Table 3.2, after normalizing to the power of line-length, as the lowest complexity feature for the first task.

3.3.3 Oblique trees with probabilistic splits

DTs are among the most powerful ML models that are widely used in practice. A tree is composed of basic computational units, called internal nodes and leaf nodes. Characterized by the hierarchical scheme of the nodes, DTs fit into complex nonlinear distributions. We previously implemented a gradient boosting ensemble of eight axis-aligned trees to detect epileptic seizures [12]. The chip was implemented in a 65nm TSMC process and achieved state-of-the-art performance in terms of energy-area-latency product [12]. Here, we consider a classification task with an input space of $\mathcal{X} \subset \mathbb{R}^D$ and output space of $\mathcal{Y} = \{1, ..., K\}$. Alternatively, the goal of this work is to learn a probabilistic tree model that can map from the feature space ($\mathcal{X}$) to the label space ($\mathcal{Y}$).

As opposed to axis-aligned decision trees, oblique trees use multiple features to make splits. As shown in Fig. 3.2, two-layer neural networks are used as split

function to combine multiple features in each internal node. As a result, the hyperplane is oblique rather than axis-aligned and can better fit to data with correlated features [99]. Training oblique trees is not a trivial task, as the weights are not differentiable. In addition, without compressing the tree structure, oblique trees may grow overly complex and use many features to make splits, increasing both the model size and node complexity. To tackle these issues, we introduce the oblique trees with probabilistic splits. Rather than deterministically routing a decision tree, we send a sample to the left or right subtree based on a probability value. With this probabilistic routing, we can derive the objective function and train oblique trees with gradient-based optimization algorithms, and use various compression techniques applied to neural networks.

Data We consider a dataset with N instances $\{(\boldsymbol{x_1}, y_1), ..., (\boldsymbol{x_N}, y_N)\} \subset (\mathcal{X}, \mathcal{Y})$ where $\boldsymbol{x_n}$ is a feature vector of length D and y_n indicates the corresponding label.

Internal node In a binary decision tree, each internal node $i \subset \{1, ..., I\}$ has two child nodes. Generally, the internal nodes compute a binary function leading to the left or right child. On the other hand, in a probabilistic tree, the internal nodes make a soft decision which generates the probability of that split going to the left or right. Since the tree is oblique, we model the decision function as:

$$\sigma(d_i(\boldsymbol{x_n})) = \frac{1}{1 + e^{-\boldsymbol{x_n}^\top \boldsymbol{\theta_i}}} \tag{3.1}$$

where $\boldsymbol{\theta_i}$ is the weight vector for the i-th internal node, $\sigma(x)$ denotes the *sigmoid* function, $d_i(\boldsymbol{x_n})$ is the output of internal node i, and $\sigma(d_i(\boldsymbol{x_n}))$ indicates the probability of leading the feature vector $\boldsymbol{x_n}$ to the left child at node i, Fig. 3.2.

Leaf node Leaf nodes are the terminal nodes of a decision tree. Each leaf node can be reached through a unique path which follows a set of decisions made by the internal nodes. Here, we use $\mathcal{R}_\ell \subset \{1, \ldots, I\}$ to represent the internal nodes

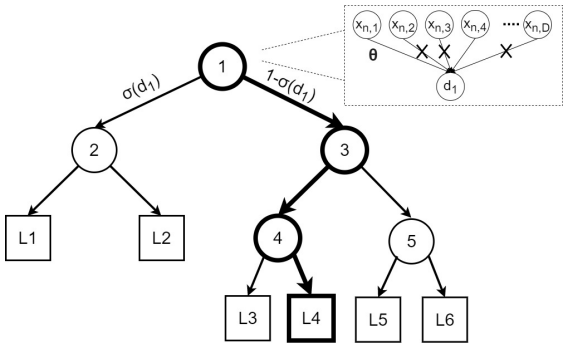

Figure 3.2: Proposed oblique tree, trained with soft decisions. In the inference phase, the test samples follow the most probable path along the tree. Inside internal nodes, the decision functions can be represented by a two-layer neural network, for which we use weight pruning and sharing techniques to create sparse connections.

which contain leaf node l in the right subtree, and $\mathcal{L}_\ell \subset \{1, \ldots, I\}$ to denote the internal nodes which contain leaf node l in the left subtree. Hence, the probability of the feature vector $\boldsymbol{x_n}$ reaching leaf node l can be expressed as:

$$p\left(l|\boldsymbol{x_n};\boldsymbol{\theta}\right) = \prod_{i\in\mathcal{L}_\ell} \sigma\left(d_i(\boldsymbol{x_n})\right) \prod_{i\in\mathcal{R}_\ell} \left(1-\sigma\left(d_i(\boldsymbol{x_n})\right)\right). \tag{3.2}$$

3.3.4 Learning procedure

Let us consider a supervised learning problem with N pairs of samples $(\boldsymbol{x_n}, y_n)$. Our goal is to maximize the empirical log-likelihood of the training data:

$$\max_{\boldsymbol{\theta},\boldsymbol{\omega}} \sum_{n=1}^{N} \log p(y_n|\boldsymbol{x_n};\boldsymbol{\theta},\boldsymbol{\omega}). \tag{3.3}$$

where $\omega_{l,k}$ indicates the probability of leaf l having the class label k. The $\omega_{l,k}$ is normalized so that the sum of probabilities in a leaf is 1, i.e., $\sum_{k=1}^{K} \omega_{l,k} = 1$. In the probabilistic routing scheme, $p(y_n|\boldsymbol{x_n};\boldsymbol{\theta},\boldsymbol{\omega}) = \sum_{l=1}^{L} p(l|\boldsymbol{x_n};\boldsymbol{\theta})\omega_{l,y_n}$. Combining the empirical loss with the regularization term $\lambda\Omega$ (explained in the following

Section), the optimization objective of our model can be expressed as follows:

$$O(\boldsymbol{\theta}, \boldsymbol{\omega}; \mathcal{X}, \mathcal{Y}) = \sum_{n=1}^{N} \log \sum_{l=1}^{L} p(l|\boldsymbol{x_n}; \boldsymbol{\theta}) \omega_{l,y_n} + \lambda \Omega. \tag{3.4}$$

The training process for probabilistic trees is to maximize the above objective function. A detailed discussion on estimating the optimal parameters ($\boldsymbol{\theta}^*$, $\boldsymbol{\omega}^*$) can be found in [89]. Intuitively, by maximizing Eq. 5.8, the samples ($\boldsymbol{x_n}, y_n$) are encouraged to reach a leaf node l where the probability of class label ω_{l,y_n} is maximized. Here, we use the gradient-based Adam optimizer [100] for learning soft oblique trees. The maximization of $O(\boldsymbol{\theta}, \boldsymbol{\omega}; \mathcal{X}, \mathcal{Y})$ is based on subsets of training samples for which $\boldsymbol{\theta}$ and $\boldsymbol{\omega}$ are updated with mini-batches until $O(\boldsymbol{\theta}, \boldsymbol{\omega}; \mathcal{X}, \mathcal{Y})$ converges.

3.4 Weight pruning, sharing & cost-aware inference

Weight pruning and sharing are model compression techniques that were initially introduced to reduce the model size for DNNs [91, 101], while a number of recent studies have focused on hardware-efficient implementation of compression techniques [102].

Considering that in the training phase of oblique trees we apply a gradient-based algorithm, each internal node can be viewed as a two-layer fully connected network. Therefore, one may compress the oblique trees through similar techniques as those used in neural network compression. Moreover, trees are based on a hierarchical structure which is favored for lightweight inference, as predictions can be made following a single root-to-leaf path without visiting the complete model. By employing these techniques, we can significantly reduce the inference complexity of oblique trees and enable their resource-efficient integration on chip.

3.4.1 Regularization

In a general neural network pruning scheme, small weights are considered trivial and set to zero. Regularization essentially penalizes the weights and encourages the values to be small. We explore two types of regularizations in this work: 1. The conventional ℓ_2 regularization; 2. A new power-efficient regularization that attempts to minimize the feature computation cost (i.e., power consumption) during inference [85].

ℓ_2 Regularization: While ℓ_1 regularization can generate sparse representations and has been used to optimize OTs [82], it has been shown that ℓ_2 regularization performs better for weight pruning [91]. Inspired by the efforts on DNN compression, here we combine the ℓ_2 regularization and weight pruning to generate sparse oblique trees. In the following, $\boldsymbol{\theta}_i$ represents the weight vector at the i-th internal node and the ℓ_2 regularization term is expressed as:

$$\Omega_{\ell_2} = \sum_{i=1}^{I} |\boldsymbol{\theta}_i|^2. \tag{3.5}$$

Power-Efficient Regularization In order to design a cost-aware classifier [81], we propose to include a regularization term that incorporates the hardware cost for feature extraction. In our previous work [85], we proposed a cost-aware model based on gradient-boosted tree ensemble to reduce the power consumption, while maintaining the classification accuracy. Here, we extend the cost-efficient study to oblique decision trees. In the following, $\boldsymbol{\beta}$ represents the feature cost vector of length D and $p_{n,i}$ is the probability of instance n going through internal node i. Therefore, the feature extraction cost for instance n at node i can be written as:

$$\Omega_{n,i} = p_{n,i}\boldsymbol{\beta}^\top ||\boldsymbol{\theta}_i||_0. \tag{3.6}$$

where $\|\theta_i\|_0$ is the ℓ_0 normalization indicating which features are used to make decisions. However, the ℓ_0 normalization is not differentiable. Thus, it is not compatible with gradient-based training. Following an approach similar to [103], we instead approximate it with ℓ_1 normalization. The new $\Omega_{n,i}$ is expressed as:

$$\Omega_{n,i} \approx p_{n,i} \sum_{j=1}^{D} \beta_j |\theta_{i,j}| \tag{3.7}$$

where β_j and $\theta_{i,j}$ indicate the j-th entry of the vectors $\boldsymbol{\beta}$ and $\boldsymbol{\theta_i}$, respectively. Here, $\Omega_{n,i}$ denotes the expected feature extraction cost for a test sample n reaching the internal node i. By evaluating the cost of mini-batch on the entire tree, the power-efficient regularization term can be expressed as:

$$\Omega_{power-efficient} = \frac{1}{N} \sum_{n=1}^{N} \sum_{i=1}^{I} \Omega_{n,i} \tag{3.8}$$

$$\approx \frac{1}{N} \sum_{n=1}^{N} \sum_{i=1}^{I} p_{n,i} \sum_{j=1}^{D} beta_j |\theta_{i,j}| \tag{3.9}$$

where N and I represent the number of training samples and internal nodes, respectively. This equation essentially introduces a power-dependent regularization term for the objective function in Eq. 5.8, by estimating the power consumption for feature extraction based on the probability of visiting each node during training. The feature costs are averaged over samples to take into account the frequency of visiting a node during training (e.g., root nodes are visited more often than those in the deeper layers of a tree).

3.4.2 Weight pruning

The number of weights associated with each internal node equals the number of attributes. Therefore, the entire weight matrix $\boldsymbol{\theta}$ has a size of $I \times D$. Each

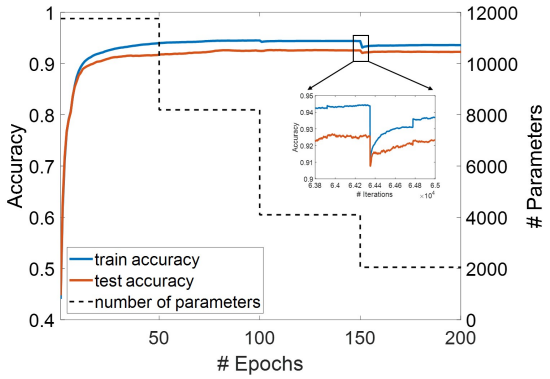

Figure 3.3: The prune-retrain process to generate sparse oblique trees. Weight pruning happens every 50 epochs. Testing on the MNIST dataset, the number of parameters is reduced by a factor of 5.7, using weight pruning.

internal node contains a decision function which can be expressed as a fully connected network with only input and output layers. Here, we follow a train-prune-retrain process to generate an oblique tree with sparse connections in its internal nodes. In the pruning phase, small weights below a threshold are set to zero. Then, we retrain the tree to optimize the remaining weights. We iteratively repeat this prune-retrain process until the final performance converges, as shown in Fig. 3.3.

Following weight pruning, we store the sparse weight matrix including the values and relative indices of the survived weights. The sparse weight matrix is stored in a column-first order and delta encoding is used to index the relative position of non-zero weights [91]. To benchmark our approach against state-of-the-art oblique tree-based models, we first tested the oblique tree with pruning on the MNIST dataset. As depicted in Fig. 3.3, we were able to reduce the parameter count from 11.8k to 2.0k with <0.1% loss in accuracy.

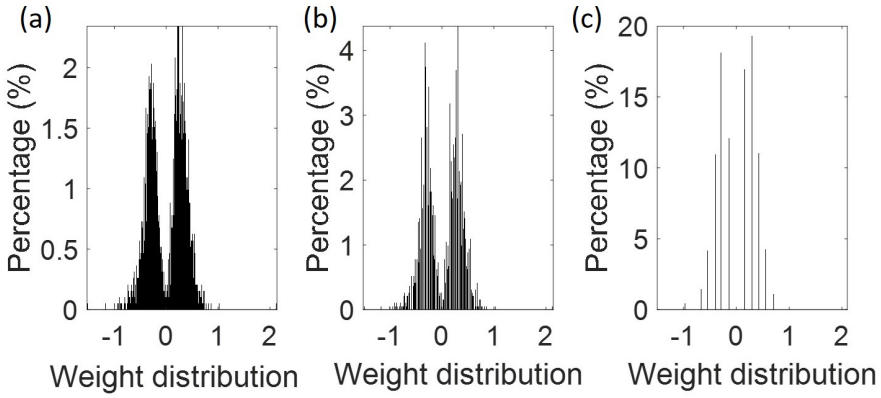

Figure 3.4: Weight distribution of an oblique decision tree: (a) without weight sharing, (b) after 8-bit weight sharing, and (c) after 4-bit weight sharing.

Algorithm 1: Learning Resource-Efficient Oblique Trees

Data: T: training set, $\boldsymbol{\beta}$: feature cost

$\boldsymbol{\theta}, \boldsymbol{\omega} \leftarrow$ pretrained by maximizing $Eq.(3.4)$;

for $i \in \{1, \ldots, nPrune\}$ **do**

  $\boldsymbol{\theta} \leftarrow WeightPruning(\boldsymbol{\theta})$;

  for $k \in \{1, \ldots, nEpochs\}$ **do**

    update $\boldsymbol{\theta}, \boldsymbol{\omega}$ by maximizing $Eq.(3.4)$

$\boldsymbol{\theta} \leftarrow WeightSharing(\boldsymbol{\theta})$

3.4.3 Weight sharing

The weight sharing process implemented here is similar to the weight quantization approach proposed in [91]. We first determine the range of the original weights. The weights are then uniformly separated into k clusters across the entire range and the shared weights are initialized by the average weight of each cluster. Thus, all weights abandon their original values and take the value of the shared weight. Following this step, the possible weights are within k shared values and only $\lceil log_2 k \rceil$ bits are required to index the weights, which are stored

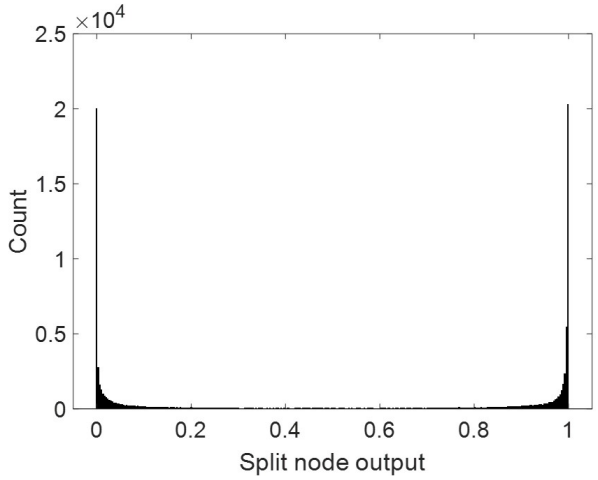

Figure 3.5: Histogram of split node outputs. The x-axis indicates the probability of leading a sample to the left sub-tree. Decisions at the split nodes are mostly certain (0 or 1). The experiment is conducted on the MNIST dataset.

in memory as floating point numbers ($k \times 32$ bits required for shared weights). Here, $\lceil x \rceil$ indicates rounding up x to the nearest integer. Lastly, we fine-tune the shared weights through gradient-based optimization and correct the potential bias induced by direct weight sharing. Figure 3.4 shows the weight distribution before and after weight sharing.

3.4.4 Single-path inference

In the common inference scheme for probabilistic trees, a test sample can travel through multiple paths, while the probability of that sample reaching each leaf node is calculated. The final prediction is made by averaging the leaf values based on their probabilities. This is referred to as a multi-path approach as shown in Eq. 3.2.

Alternatively, we apply a single-path approach to enable lightweight inference. In the proposed scheme, the test samples choose the most probable path at each internal node, as shown in Fig. 3.2. This enables us to make predictions

by only visiting a small portion of the model and extracting fewer features. The single-path routing scheme is summarized in Eq. 3.10, where $\lfloor p \rceil$ denotes rounding the probability p to either 0 or 1.

$$p_{\text{single-path}}(l|\boldsymbol{x_n};\boldsymbol{\theta}) = \prod_{i\in\mathcal{L}_\ell} \lfloor \sigma\left(d_i(\boldsymbol{x_n})\right) \rceil \prod_{i\in\mathcal{R}_\ell} \lfloor 1-\sigma\left(d_i(\boldsymbol{x_n})\right) \rceil. \tag{3.10}$$

However, the single-path inference is an approximation method and may result in information loss. To carefully analyze this, the histogram of all split node outputs for the proposed soft decision tree is plotted in Fig. 3.5. This distribution shows that samples are routed to the left or right sub-trees with very low uncertainty. In other words, most available paths will never be visited. Therefore, only the most probable path needs to be evaluated at the test time. Similar results were reported in [89, 104] and our experimental results support this single-path inference scheme.

The algorithmic pseudocode to train ResOT is presented as Algorithm. 1. We performed the train-prune-retrain process for *nPrune* rounds, and updated the parameters for *nEpochs* epochs using gradient-based learning. By employing model compression techniques and gradient-based maximization, the algorithm returns a sparse weight matrix $\boldsymbol{\theta}$ for an oblique decision tree.

3.5 Results

In this chapter, we propose a framework to generate resource-efficient oblique trees (ResOT) with model compression, power-efficient learning [85], and single-path inference. As a result, our approach offers a small model size and lightweight inference, both requisite for low-power brain implants. In this Section, we implemented the ResOT model for several tasks, including a toy digit recognition task [105] on MNIST dataset and three neural signal classification

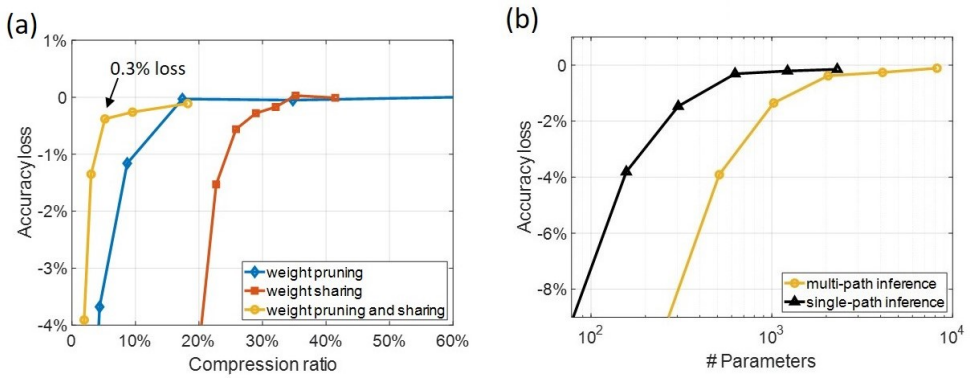

Figure 3.6: (a) The compression ratio of an oblique tree with weight pruning and sharing. We are able to compress the model by 20× with a marginal accuracy loss (0.3%); (b) Comparison between the single- and multi-path inference schemes in terms of average number of parameters used during inference. The single-path inference requires fewer parameters and causes no performance loss. The experiment is conducted on the MNIST dataset.

problems. The purpose of first task is to benchmark our model against state-of-the-art algorithms that are all tested on the MNIST dataset, in terms of accuracy and model compression.

Among these tasks, seizure and tremor detection require binary labels, whereas digit recognition and finger movement classification involve multiple classes in their label space. A single ResOT was built for both binary and multi-class tasks. In addition, we compared the ResOT approach with other DT-based models in terms of classification performance, model size, and power consumption, particularly for the three neural classification tasks that require a low-power implementation.

3.5.1 Performance on MNIST dataset

We first tested the oblique tree compression on a toy dataset, MNIST [105], for a 10-class digit recognition task. We used the ℓ_2 regularization (Eq. 3.5) for weight

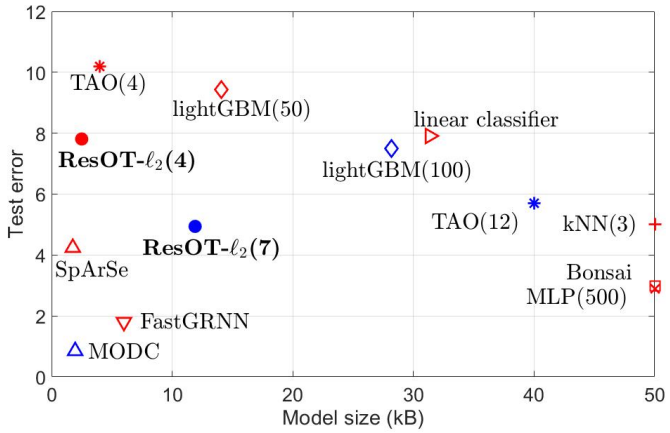

Figure 3.7: The performance of ResOT on the MNIST dataset compared to other algorithms based on sparse oblique trees (TAO [82] and Bonsai [79]). Other machine learning models such as FastGRNN [106], SpArSe [107], MODC [108], lightGBM [109], logistic regression (linear model), k-nearest neighbors (kNN) and multilayer perceptron (MLP) were also included. Various model types are shown with different marker styles, while different colors represent model configurations. Values outside the range are plotted on the boundary.

pruning, since the feature extraction cost was not a concern for this task. Testing on a computer with a 6-core i7-8700K CPU, 178.4 seconds were required to build a ResOT of depth 4 (pruning rounds: 4, batch size: 128, total epochs: 400). In order to reduce the model size, the tree was pruned to 2048 parameters, indexed by 4 bits following weight sharing. Compared with the original OT model, weight pruning and sharing significantly reduced the model size. As illustrated in Fig. 3.6(a), weight pruning and sharing could reduce the model size by 20× with only a marginal accuracy loss (0.3%). Moreover, although ResOT is trained with a probabilistic routing scheme, it is still compatible with lightweight single-path inference. In Fig. 3.6(b), we show that the single-path inference uses 3.8× fewer parameters while maintaining the classification accuracy.

We compared our model with state-of-the-art sparse oblique trees such as

TAO [82] and Bonsai [79], as well as other ML models including logistic regression, kNN, multilayer perceptron (MLP), lightGBM [109], FastGRNN [106], SpArSe [107] and MODC [108]. The settings of different models are as follows (also marked on Fig. 3.7): (1) **ResOT-ℓ_2(n)**: the proposed resource-efficient oblique tree with ℓ_2 regularization and a maximum depth of **n**. (2) **lightGBM(n)**: a boosted ensemble with **n** axis-aligned decision trees, where a single tree is a one-vs-all (ova) binary classifier. (3) **Bonsai**: the Bonsai model that is based on sparse-projected trees [79]. (4) **TAO(n)**: the alternating optimized oblique tree (TAO) with a maximum depth of **n**; TAO iteratively optimizes the decision functions of the internal nodes and uses the ℓ_1 regularization to generate sparse connections. (5) **linear classifier**: one-vs-all logistic regression classifier. (6) **kNN(n)**: **n**-nearest neighbor classifier. (7) **MLP(n)**: multilayer perceptron with single hidden layer and **n** units. (8) **SpArSe** and **MODC**: memory-optimized convolutional neural networks. (9) **FastGRNN**: gated recurrent neural network with optimized model size. In all experiments, the weights were quantized to 4 bits (16 shared weights).

With a shallow ResOT (maximum depth of 4), we achieved a 7.81% test error with only 2.5kB of memory, which is better than TAO [82] (10.2%) at similar settings (maximum depth of 4). The test error can be further reduced by increasing the depth. We achieved a 4.94% error with a depth of 7, which is better than TAO (5.69%) with a deeper tree (depth of 12). We used the single-path inference scheme to evaluate the test error of ResOT in these experiments. It should be noted that although deep learning methods based on CNN/RNN (e.g., SpArSe, MODC, FastGRNN) achieve a good performance with a small model size, they are likely to suffer from a higher computational complexity compared to DTs, since single-path inference is not possible in such networks and they typically

require many multiplications and additions in their multi-layer architectures. On the contrary, the proposed ResOT inherits the lightweight single-path inference thanks to its hierarchical structure, and outperforms state-of-the-art oblique tree-based models with low complexity such as Bonsai and TAO.

The decision process of ResOT is interpretable and can be easily visualized. Figure 3.8 shows the visualization of our oblique tree (max depth of 4) trained on the MNIST dataset. Both internal and leaf nodes are represented by pie charts, showing the label distribution of the test data going through those nodes. The labels are from 0 to 9, indicating which digit the image is belonging to. By moving deeper in the tree, different digits tend to appear at different leaves. The leaf nodes are labeled by the dominant class passing through that node. Therefore, a single ResOT is capable of accurate multi-class separation with a shallow depth of 4.

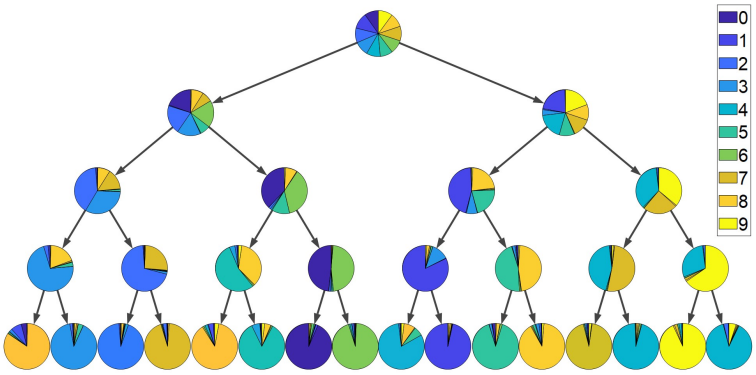

Figure 3.8: The visualization of ResOT trained on the MNIST dataset. Both internal and leaf nodes are represented by pie charts, indicating the class distribution of test samples. We can see that samples with different class labels are mixed in the root node. Following classification, however, each leaf node has a dominant class label.

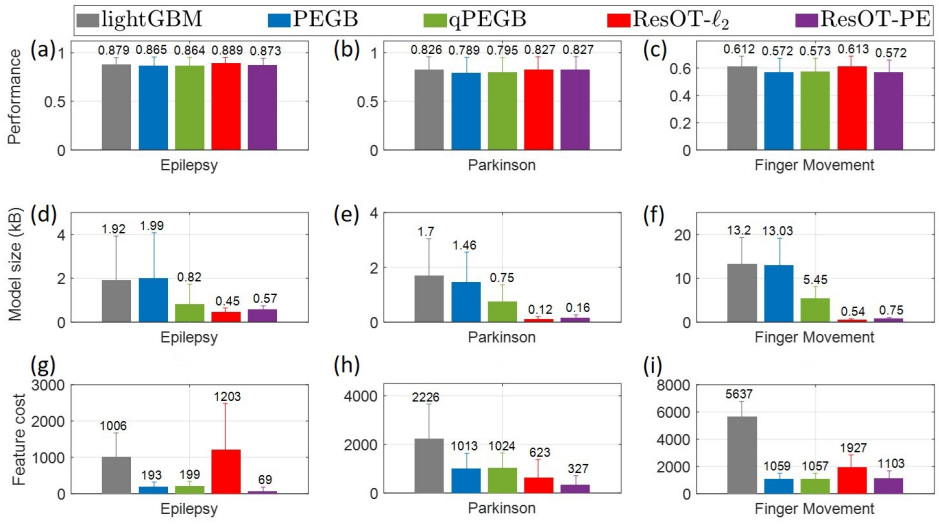

Figure 3.9: The classification performance on neural tasks. We implemented 5 different DT-based models including lightGBM [109], power-efficient gradient boosting (PEGB) [85], quantized PEGB (qPEGB) [85], and the proposed resource-efficient oblique tree with ℓ_2 regularization (ResOT-ℓ_2, this work) and with power-efficient regularization (ResOT-PE, this work). The models are compared in classification performance, model size, and feature cost (power), for each task. The error bars indicate the standard deviation across subjects and the average values are shown on the bars. The size of memory and feature extraction cost are reduced with the proposed oblique tree-based models.

3.5.2 Performance on neural tasks

In order to assess the performance of the proposed ResOT model in neural signal classification, we performed three experiments including epileptic seizure detection, Parkinsonian tremor detection, and finger movement classification. We used a block-wise approach to split the data into train and test sets. For seizure detection, the continuous iEEG recordings were divided into multiple blocks, where each block consisted of one seizure segment and the preceding non-seizure segment. For tremor and finger movement detection, the continuous LFP and ECoG recordings for each patient were divided into 5 equal-sized blocks, without shuffling. A 5-fold cross-validation method was used to report the classification performance for each task, where 80% of blocks (rounded to

the nearest integer) were used for training the model in each round and 20% for testing, and the results of 5 rounds were averaged. For epilepsy patients with less than 5 seizures, the performance was evaluated based on a leave-one-out cross-validation (i.e., 2 blocks for training and 1 block for testing in a patient with 3 seizures). The average training time of ResOT-ℓ_2/ResOT-PE for seizure detection, tremor detection, and finger movement classification was 86.1/144.2, 4.2/5.2 and 36.4/41.6 seconds, respectively.

Figure 3.9 compares different tree-based machine learning models in terms of classification performance, model size, and feature extraction cost on three neural tasks. We have previously shown that the ensemble of gradient boosted trees achieves a higher classification performance on these tasks (i.e., seizure detection [12], tremor detection [2], and finger movement classification [76]), compared to other ML models. Therefore, here we compare our approach against the lightGBM classifier [109], which is a fast and high-performance implementation of gradient-boosted trees. The hyperparameters (maximum depth and tree count) for lightGBM were optimized in a subject-specific basis. Overall, we trained 4–20 trees for seizure detection, 5–40 for tremor detection, and 30–60 for finger movement classification, with maximum depths ranging from 2 to 6. The second model, Power-Efficient Gradient Boosting (PEGB), is a modified version of lightGBM that incorporates the feature power dissipation in the objective function [85]. In our previous work, we showed that this approach can reduce the feature computational cost while maintaining the classification performance for both seizure and tremor detection tasks [85]. However, even though PEGB can greatly reduce the hardware cost of a conventional gradient boosting model, its circuit implementation suffers from the large model size of the ensemble and limited scalability. To partially tackle this issue, quantized

PEGB (qPEGB) with fixed-point arithmetic was proposed [85] that improved the model size compared to PEGB. We quantized the thresholds and leaf weights with 12 bits for tremor detection, while 3/12 bits were used to quantize the leaf weights/thresholds for seizure detection and movement classification [98]. Similarly, the maximum depth and number of trees for PEGB and qPEGB were optimized for each subject and both models were included in this study.

Furthermore, we implemented the proposed ResOT model as described in Section 3.3.3, with two different settings: (1) oblique tree with ℓ_2 regularization (ResOT-ℓ_2), and (2) oblique tree with a power-efficient regularization (ResOT-PE). Specifically, we regularized the oblique trees by tuning the weight pruning threshold and regularization coefficients (λ). We also limited the maximum tree-depth, while the final depth was determined by the algorithm under several constraints (pruning, regularization, maximum depth). Both ResOT-ℓ_2 and ResOT-PE consist of a single oblique tree with a maximum depth of 4, while 4 bits were used to represent the shared weights. The regularization coefficients (λ) and number of parameters (following weight pruning) were optimized for each patient. Overall, the size of the weight matrix $\boldsymbol{\theta}$ (i.e., $I \times D$) varies from 15×565 to 15×1409 for seizure detection, is 15×37 for tremor detection, and varies from 15×343 to 15×577 for finger movement classification. After weight pruning, the weight matrix becomes sparse, with an average of 108.8/121.6, 57/89, and 270.2/455.1 non-zero elements for ResOT-ℓ_2/ResOT-PE on three neural tasks.

We used the accuracy measure to evaluate the classification performance on finger movement task, Fig. 3.9(c), and F1 score for seizure, Fig. 3.9(a), and tremor detection tasks, Fig. 3.9(b), due to their highly unbalanced datasets.

In our experiments, lightGBM achieved an average F1 score of 0.879(±0.069) on seizure detection, 0.826(±0.131) on tremor detection, and an average accuracy of 0.612(±0.077) on finger movement classification task. However, rather than using an ensemble of boosted trees, we show that a comparable performance can be achieved using a single oblique tree with ℓ_2 regularization (F1 scores of 0.889(±0.059) for epilepsy and 0.827(±0.131) for PD, and an accuracy of 0.613(±0.076) for finger movement). The sensitivity and specificity of these models in seizure and tremor detection are reported in Table. 3.3. With power-efficient approach (PEGB, qPEGB, ResOT-PE), we made a trade-off between classification performance and hardware cost. Overall, the proposed ResOT-PE model achieves a comparable classification performance as lightGBM on all tasks, while significantly reducing the model size and power consumption, as later discussed in this Section.

Figures 3.9(d-f) compare the model size for different ML algorithms. As shown in these figures, ResOT-ℓ_2 is the most memory-efficient model, with 0.45(±0.19), 0.12(±0.08), and 0.54(±0.19) kB model sizes for the three performed tasks, respectively. Compared with lightGBM, ResOT-ℓ_2 reduces the model size by 4.3× for seizure detection, 14.2× for tremor detection, and 24.4× for finger movement classification. The ResOT-PE has a slightly larger model size compared to ResOT-ℓ_2 (memory saving of 3.4×, 10.6×, and 17.6× for seizure detection, tremor detection, and finger movement classification, compared to lightGBM), since the power consumption is optimized along with memory. The model size of PEGB is comparable to lightGBM, while a reduction factor of 2.3× can be achieved by quantizing the parameters (qPEGB).

As discussed in Section 3.3.2, the power consumption for feature extraction

Table 3.3: Sensitivity and Specificity for Seizure and Tremor Detection

Models	Seizure Detection		Tremor Detection	
	Sensitivity	Specificity	Sensitivity	Specificity
lightGBM	0.881±0.069	0.984±0.016	0.890±0.103	0.281±0.263
PEGB	0.881±0.077	0.974±0.036	0.838±0.171	0.274±0.259
qPEGB	0.881±0.077	0.974±0.036	0.850±0.157	0.262±0.252
ResOT-ℓ_2	0.877±0.076	0.986±0.012	0.897±0.097	0.274±0.252
ResOT-PE	0.860±0.084	0.989±0.012	0.901±0.091	0.247±0.260

was minimized through our power-efficient regularization framework. Figures 3.9(g-i) depict the inference power for various models on the three neural tasks. The power during inference was calculated based on a single-path scheme for all models, by summing up the power consumption of features visited along the decision path. For example, for the case of ResOT, the power cost for a test sample is calculated by summing up the cost of extracted features: $\sum_{i=1}^{I}\sum_{j=1}^{D}\beta_j C_{i,j}$, where $C_{i,j} = 0$ or 1 indicates whether the internal node i is on the decision path and feature j is extracted in that node. Compared to the benchmark model (lightGBM), PEGB achieves an average power reduction of 4.2× on these tasks. Compared to lightGBM, the ResOT-PE reduces the estimated power by 14.6× for seizure detection, 6.8× for tremor detection, and 5.1× for finger movement classification. The ResOT-PE model obtains the lowest feature extraction cost on average, while offering a considerably smaller model size compared to qPEGB.

3.5.3 Power-efficient inference

To better demonstrate the superior power efficiency of ResOT-PE over ResOT-ℓ_2, we took a closer look into the structure of these models and the extracted features in the oblique nodes. Figure 3.10 shows the average number of extractions for each feature, using the ℓ_2 and power-efficient regularization schemes. By adopting the power-efficient regularization term (Eq. 3.8) with the weight pruning framework, the power-hungry features are assigned smaller weights,

and as a result, they are less likely to survive during the weight pruning process. As depicted in Table. 3.2, each feature is associated with a normalized power cost, including its static and dynamic power. Taking the seizure detection task as an example, power, variance, and line-length have a low complexity, while band power features consume the highest energy due to multiplications and additions required for FIR filtering. In the power-efficient learning scheme, hardware-friendly features are favored over costly ones, and the model tends to use a higher number of these features compared to band powers. Similarly, we see a smaller number of band power extraction for tremor detection and finger movement classification, while a higher number of Hjorth activity, mobility, and LMP features are extracted in these tasks. Compared with ResOT-ℓ_2, the ResOT-PE model reduced the power cost by 17.4×, 1.9×, and 1.7× for these neural tasks, respectively, as shown in Fig. 3.10.

It should be noted that in this work, we implemented a specialized digital hardware to measure the power of each feature as a stand-alone block, including its dynamic and static power. However, in a full system implementation, various resource optimization and power reduction techniques (e.g., mixed-signal design, re-using the common blocks between different features, power and clock gating) can be employed to further reduce the total energy of the system.

3.6 Discussion

The proposed ResOT model could be considered as a combination of decision trees and neural networks, that is further optimized for low-power implementation. With the mini-batch gradient descent training, ResOT is also compatible

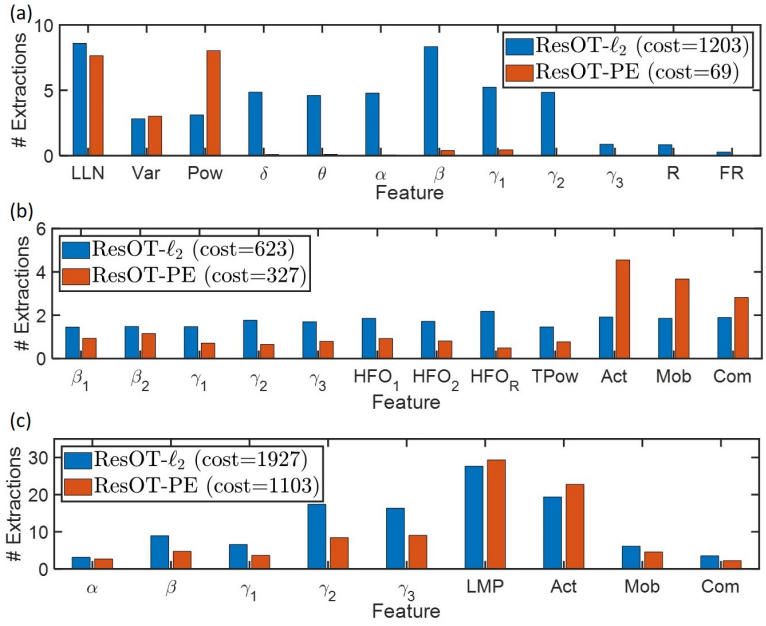

Figure 3.10: Illustration of the proposed power-efficient framework in terms of feature extraction count, by comparing the ℓ_2 and power-efficient regularization schemes. The bars indicate the total number of extractions for each feature along the decision path, that is averaged over subjects, for (a) seizure detection, (b) tremor detection, and (c) finger movement classification tasks.

with an online learning framework where the model could dynamically adapt to the non-stationary nature of neural signal, which is the focus of our future work. In this Section, we discuss the contributions of this chapter and the benefits of ResOT in terms of memory and power efficiency, as well as the hardware overhead for implementing oblique splits, with a focus on the above neural classification tasks.

3.6.1 Hardware improvements

The benefits of the proposed ResOT model for hardware implementation are as follows: **1. Memory Efficiency;** Compared with conventional oblique trees, the proposed ResOT model is compressed via weight pruning and sharing. The resulting parameter matrix is sparse and can be efficiently stored in an on-

chip memory such as SRAM. Compared to our earlier work that integrated 8 gradient-boosted trees with 1kB of memory for epilepsy [12], this work requires 2.2× less memory on average (ResOT-ℓ_2), for storing the parameters of a single oblique tree. **2. Lightweight Inference;** Compared to a neural network (that could be compressed with similar techniques), our model benefits from the hierarchical structure of trees. During inference, we only need to follow a single root-to-leaf path to make predictions, without visiting the rest of the model. This lightweight inference can potentially improve the energy efficiency for implantable and edge applications. **3. Cost-Aware Learning;** With cost-aware learning, ResOT learns to prioritize the lower-cost features during inference. We used the power consumption as a measure of cost in this work, and ResOT-PE achieved one of the lowest power consumptions among the analyzed models on all target tasks. The pros and cons of different tree-based models are summarized in Table. 3.4.

3.6.2 Benefits of oblique splits

Conventional DT-based models such as random forest and lightGBM are composed of axis-aligned trees with one feature-threshold pair in each internal node. The decision boundaries of axis-aligned trees are parallel to the feature space axes. As a result, such topologies ignore the correlations between fea-

Table 3.4: Comparison of Different Tree-based Models

Model	Cost-aware	Model Compression	# of Trees	Split Type
lightGBM	✗	✗	> 1	Axis-aligned
PEGB	✓	✗	> 1	Axis-aligned
qPEGB	✓	✓[†]	> 1	Axis-aligned
ResOT-ℓ_2	✗	✓[‡]	1	Oblique
ResOT-PE	✓	✓[‡]	1	Oblique

[†] weight and threshold quantization
[‡] weight pruning and sharing

Table 3.5: The Overhead Cost of Oblique Nodes in ResOT

Task	ResOT-ℓ_2		ResOT-PE	
	# Mult.	# Add.	# Mult.	# Add.
Epilepsy	108.8	102.6	121.6	107.1
Parkinson	57	46.8	89	75.5
Finger Movement	270.2	260	455.1	440.1

tures and might be suboptimal for classifying highly correlated data. To tackle this issue, oblique trees use a linear model as the split function. Thus, multiple features are combined at the internal node and the decision boundary is an oblique hyperplane that can better adapt to the various distributions of input data. Therefore, oblique trees perform better on signals with strongly correlated features [99]. This is typically the case for neural signal classification tasks. For example, line-length and variance both describe the signal variations and show a positive correlation during a seizure, although their mathematical definitions are different. Moreover, the correlation between neural signals recorded by adjacent channels of an electrode array (depending on the spacing of electrodes), leads to a correlation between corresponding features. Thus, oblique trees proposed here are favored over axis-aligned trees for neural signal classification tasks (e.g., ResOT-ℓ_2/ResOT-PE perform slightly better than lightGBM/PEGB in terms of F1 score, as shown in Fig. 3.9(a-c)).

Indeed, axis-aligned trees can be considered as a subset of oblique trees and have the advantage of fast training and easy interpretation. In this chapter, we show that ResOT is a memory- and power-efficient alternative for large axis-aligned ensembles, and is particularly useful for neural classification tasks that may require many trees.

3.6.3 Hardware complexity of oblique nodes

Axis-aligned decision trees use simple comparators at their internal nodes to compare a feature with a threshold. Alternatively in this work, we used oblique trees trained with a probabilistic routing. While a *sigmoid* function needs to be calculated in a probabilistic training phase, it is simplified to a comparison during single-path inference (i.e., $\boldsymbol{x}_n^\top \boldsymbol{\theta}_i > 0$ or ≤ 0), since the test samples only travel through the most probable path. Thus, similar to gradient-boosted trees [12], the hardware complexity is dominated by feature extraction process and comparators can be ignored in total power estimation.

However, oblique trees still require a weighted sum of features as input to the comparator. These features are associated with the non-zero elements of the sparse weight matrix $\boldsymbol{\theta}$ following weight pruning. In addition to the hardware cost for features (Fig. 3.9), the overhead cost of implementing the weighted sum should be considered for an oblique tree. Table. 3.5 summarizes the average overhead cost of ResOT-ℓ_2 and ResOT-PE on three neural tasks, by calculating the total number of multiplications and additions required to generate the weighted sum of features in an oblique tree (i.e., calculated for 15 nodes in a tree of depth 4). Since each non-zero element is associated with a multiplication, the number of additional multiplications is inversely proportional to the sparsity of the weight matrix $\boldsymbol{\theta}$. Interestingly, it can be shown that the hardware cost for oblique node implementation is negligible compared to a single feature extraction. Here, we consider the case of neural data classification using oblique trees, where the input signal is sampled at f Hz with a window size of W used for feature extraction. To implement a band power feature (as a comparison), the signal is first passed through a digital FIR filter. A total number of $t \times N$ multipliers and $(t-1) \times N$ adders are required to filter the neural signal, where

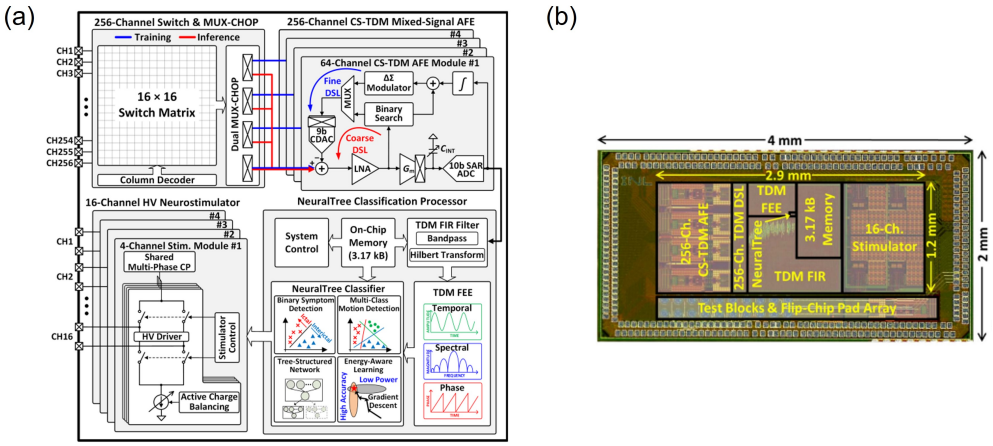

Figure 3.11: (a) The architecture of a 256-channel scalable and versatile closed-loop SoC. The proposed ResOT is implemented on-chip to decode neural activities. (b) The chip micrograph.

$t = 30$ represents the number of FIR taps and $N = f \times W$ indicates the number of samples in a window of signal. We then calculate the power of the filtered signal, which requires an additional N multipliers and $N - 1$ adders. In total, the extraction of a band power feature requires $(t + 1) \times N$ multiplications and $t \times N - 1$ additions. Assuming the worst case overhead cost for ResOT-PE applied to finger movement classification (Table. 3.5) with $f = 1$kHz and $W = 0.2s$, the extraction of a single band power feature requires 6200 multiplications and 5999 additions, whereas the ResOT-PE evaluation only requires 455.1 multiplications and 440.1 additions on average (over subjects) to linearly combine the features in the oblique nodes. Thus, the overhead cost is lower than feature cost of a single band power feature by over an order of magnitude. Moreover, in the single-path inference scheme, a maximum of 4 nodes are sequentially processed per tree, which could further reduce the complexity of the classifier during inference. Therefore, the overhead cost of oblique nodes in this single-tree scheme is marginal and will not burden the resource-efficient implementation of proposed OT model.

3.6.4 Hardware implementation

We note that ResOT was implemented on-chip, as presented in [110, 111]. The System-on-Chip (SoC) was produced using the TSMC 65-nm CMOS process, resulting in a chip dimension of 4 × 2 mm2. Figure 3.11 displays the chip architecture and micrograph. The SoC, encompassing 256 channels, only takes up an active area of 3.48 mm^2 (equivalent to 0.014 mm^2 per channel) and, when in inference mode, consumes 453 μW at a supply voltage of 1.2 V.

3.7 Conclusion

In this chapter, we proposed the ResOT model, a hardware- and memory-efficient approach for neural signal classification. Weight pruning and sharing were applied together with power-efficient regularization to compress the tree and enable cost-aware learning. Being trained with a probabilistic routing, ResOT benefits from a single-path inference scheme, enabling its lightweight implementation. Testing on three neural signal classification tasks with 31 patients, our model outperformed the state-of-the-art ensemble of boosted trees in both model size and power consumption. Resource-constrained applications such as neural implants and IoT devices could benefit from the proposed hardware-friendly classifier.

CHAPTER 4

TREE IN TREE: FROM DECISION TREES TO DECISION GRAPHS

4.1 Introduction

Decision trees (DTs) and tree ensembles are widely used in practice, particularly for applications that require few parameters [79, 112–114], fast inference [82, 115], and good interpretability [116, 117]. In a DT, the internal and leaf nodes are organized in a binary structure, with internal nodes defining the routing function and leaf nodes predicting the class label. Although DTs are easy to train by recursively splitting leaf nodes, the tree structure can be suboptimal for the following reasons: (1) DTs can grow exponentially large as the depth of the tree increases. Yet, the root-leaf path can be short even for large DTs, limiting the predictive power. (2) In a DT, the nodes are not shared across different paths, reducing the efficiency of the model.

Decision trees are similar to neural networks (NNs) in that both models are composed of basic units. A possible way to enhance the performance of DTs or NNs is to replace the basic units with more powerful models. For instance, "Network in Network" builds micro NNs with complex structures within local receptive fields to achieve state-of-the-art performances on image recognition tasks [118]. As for DTs, previous work replaced the axis-aligned splits with logistic regression or linear support vector machines to construct oblique trees [82, 87, 112, 113, 119, 120]. The work in [114] further incorporates convolution operations into DTs for improved performance on image recognition tasks, while [112] replaces the leaf predictors with linear regression to improve the regression performance. Unlike the greedy training algorithms used for axis-aligned trees (e.g., Classification and Regression Trees or CART [121]),

oblique trees are generally trained by gradient-based [113, 119, 120] or alternating [82, 112] optimization algorithms.

Inspired by the concepts of Network in Network [118] and oblique trees [82, 87], we propose a novel model, Tree in Tree (TnT), to recursively replace the internal and leaf nodes with micro decision trees. In contrast to a conventional tree structure, the nodes in a TnT form a Directed Acyclic Graph (DAG) to address the aforementioned limitations and construct a more efficient model. Unlike previous oblique trees that were optimized on a predefined tree structure [112, 114], TnT can learn graph connections from scratch. **The major contributions of this work are as follows: (1) We extend decision trees to decision graphs and propose a scalable algorithm to construct large decision graphs. (2) We show that the proposed algorithm outperforms existing decision trees/graphs, either as a stand-alone classifier or base estimator in an ensemble, under the same model complexity constraints. (3) Rather than relying on a predefined graph/tree structure, the proposed algorithm is capable of learning graph connections from scratch (i.e., starting from a single leaf node) and offers a fully interpretable decision process.**

Algorithm 2: Naive decision graph (NDG) [122]

```
G ← initialize graph with a leaf node;
for i ← 1 to N do
    for each leaf node (l_i) ∈ G do
        Find the maximum gain (g_i) if we split l_i ;
    for each pair of leaf nodes (l_i, l_j) ∈ G do
        Record gain (g_{i,j}) if we merge l_i and l_j;
    Split/merge nodes to maximize gain;
Note: The split operation has a model complexity penalty (C) for creating an internal node.
```

4.2 Related work

Decision graph (DG) is a generalization of the conventional decision tree algorithm, extending the tree structure to a directed acyclic graph [122, 123]. Despite similarity in using a sequential inference scheme, training and optimizing DGs is more challenging due to the large search space for the graph structure. The work in [122] proposed a greedy algorithm to train DGs by tentatively joining pairs of leaf nodes at each training step (NDG, Algorithm 2). Alternatively, in this work, we revisit the concept of decision graphs by exploiting recent advances in non-greedy tree optimization algorithms [82, 87, 120, 124]. Our proposed Tree in Tree algorithm can construct DGs as a more accurate and efficient alternative to the widely-used decision trees, both as stand-alone classifiers and as weak learners in the ensembles.

Conventional decision tree learning algorithms such as CART [121] and its variations follow a greedy top-down growing scheme. Recent work has focused on optimizing the structure of the tree [88, 124, 125]. However, constructing an optimal binary DT is NP-hard [86] and optimal trees are not scalable to large datasets with many samples and features [88, 124, 125]. Recent studies have further developed scalable algorithms for non-greedy decision tree optimization, with no guarantee on tree optimality [82, 87, 112, 113, 119, 120]. Such scalable approaches can be categorized into two groups: tree alternating optimization (TAO) [82, 112] and gradient-based optimization [87, 113, 119, 120].

TAO decomposes the tree optimization problem into a set of reduced problems imposed at the node levels. The work in [82] applied the alternating optimization to both axis-aligned trees and sparse oblique trees. Later, [112] extended TAO to regression tasks and ensemble methods. Unlike TAO, gradient-

based optimization requires a differentiable objective function, which can be obtained by different methods. For example, [87] derived a convex-concave upper bound of the empirical loss. [119] and [113] considered a soft (i.e., probabilistic) split at the internal nodes and formulated a global objective function. The activation function for soft splits was refined in [120] to enable conditional inference and parameter update. Both TAO and gradient-based optimization operate on a predefined tree structure and optimize the parameters of the internal nodes.

The proposed Tree in Tree algorithm aims to optimize the graph/tree structure by growing micro decision trees inside current nodes. Compared to the greedy top-down tree induction [121], Tree in Tree solves a reduced optimization problem at each node, which is enabled via non-greedy tree alternating optimization techniques [82]. Compared to NDG, TnT employs a non-greedy process to construct decision graphs, which leads to an improved classification performance (discussed in later sections). Compared to axis-aligned decision trees (e.g., TAO [82, 112], CART [121]), TnT extends the tree structure to a more accurate and compact directed acyclic graph, in which nodes are shared across multiple paths.

4.3 Methods

In this work, we consider a classification task with input and output spaces denoted by $\mathcal{X} \subset \mathbb{R}^D$ and $\mathcal{Y} = \{1, ..., K\}$, respectively. Similar to conventional decision trees, a decision graph classifier G consists of internal nodes and leaf nodes. Each internal node is assigned a binary split function $s(\cdot;\theta) : \mathcal{X} \rightarrow [left\_child, right\_child]$ parametrized by θ, which defines the routing function of a graph. For axis-aligned splits, θ indicates a feature index and a threshold. The terminal nodes (with no children) are named leaf nodes and indicate the class

labels.

4.3.1 Decision graph

As an extension to the tree structure, decision graphs organize the nodes into a more generic directed acyclic graph. In this work, we limit our discussion to axis-aligned binary DTs/DGs in which each internal node compares a feature value to a threshold to select one of the two child nodes. Similar to the sequential inference process in DTs, the test samples in a DG start from the root and successively select a path at the internal nodes until a leaf node is reached. The main differences between binary DTs and DGs are the following: (1) In DTs, each child node has one parent node. However, DGs allow multiple parent nodes to share the same child node. Therefore, DG can combine the nodes with similar behaviors (e.g., similar split functions) to reduce model complexity. (2) In binary DTs, the number of leaf nodes is always greater than the internal nodes by one. In DGs, however, $\#Leaves \leq \#Internals + 1$, since multiple internal nodes can share the same leaf node. Furthermore, there exists a unique path to reach each leaf node in a tree structure, which does not hold within DGs. (3) The model complexity of a DT is often quantified by the number of internal or leaf nodes. However, we can post-process a DG by merging the leaf nodes with the same class label. As a result, DGs have a minimum leaf node count equal to the number of classes. Therefore, we use the number of splits (i.e., internal nodes) to quantify the model complexity of a DG.

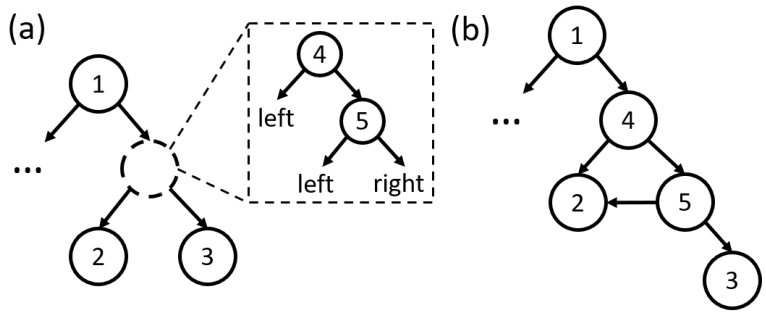

Figure 4.1: (a) The growing phase of TnT. The micro decision tree (in dashed box) replaces an internal node (dashed circle). Compared to a single node, the substitute micro tree can provide a more powerful split function. (b) The merging phase of TnT. We merge the fitted micro tree into the current structure to create a directed acyclic graph.

4.3.2 Tree in Tree

We propose a novel algorithm named Tree in Tree as a scalable method to construct large decision graphs. Conventional DT training algorithms (e.g., CART) are greedy and recursively split the leaf nodes to grow a deep structure, without optimizing the previously learned split functions. **The key difference between the proposed TnT model and conventional approaches lies in the optimization of the internal nodes. TnT fits new decision trees in place of the internal/leaf nodes and employs such micro DTs to construct a directed acyclic graph.** Overall, the proposed TnT model is a novel extension to the conventional decision trees and generates accurate predictions by routing samples through a directed acyclic graph.

Figure 4.1 shows the high-level procedure for training a decision graph with the proposed TnT algorithm. Assuming a starting decision graph (e.g., a decision tree or a single leaf node), our goal is to grow a larger model with improved predictive power. In the growing phase of TnT (Fig. 4.1(a)), we replace a node (dashed circle) with a micro decision tree with multiple splits to enable more

accurate decision boundaries. In the merging phase (Fig. 4.1(b)), the micro decision tree is merged into the starting model to construct a TnT decision graph, in which a child node (node 2) may have multiple parent nodes (node 4 and 5).

Growing the graph from internal nodes We consider the training of a decision graph as an optimization problem with the aim of minimizing the loss function on the training data:

$$\min \sum_{x,y \in \mathbb{X},\mathbb{Y}} L\left(y, G(x;\Theta)\right). \tag{4.1}$$

TnT grows the decision graph $G(\cdot;\Theta)$ from an arbitrary internal node $n_i \in G$ with the split function $s(\cdot;\theta_i)$. θ_i denotes the trainable parameters of n_i including a feature index and a threshold for axis-aligned splits. The overall goal is to replace n_i with a decision tree t_i and minimize the loss function as indicated in (4.1). All other nodes remain unchanged as we train t_i.

Let us consider a subset of samples $(\mathbb{X}_{subset}, \mathbb{Y}_{subset})$ that is sensitive to the split function $s(\cdot;\theta_i)$, as defined by the following expression:

$$G_{n_i \rightarrow left}(\mathbb{X}_{subset}; \Theta \backslash \theta_i) \neq G_{n_i \rightarrow right}(\mathbb{X}_{subset}; \Theta \backslash \theta_i), \tag{4.2}$$

where $\Theta\backslash\theta_i$ denotes the parameters of all nodes in G excluding n_i. Growing the graph from n_i does not change $\Theta\backslash\theta_i$ since all other nodes are fixed as we solve the reduced optimization problem at n_i. $G_{n_i \rightarrow left}$ sends the samples to the left child at n_i (i.e., $s(\cdot;\theta_i) \rightarrow left\_child$) while $G_{n_i \rightarrow right}$ routes the samples to the right child at n_i. With $\Theta\backslash\theta_i$ being fixed, the output of decision graph only depends on θ_i (i.e., $s(\cdot;\theta_i)$)

$$G(x;\Theta) = \begin{cases} G_{n_i \rightarrow left}(x; \Theta \backslash \theta_i) & \text{if } s(x;\theta_i) \rightarrow left\_child \\ G_{n_i \rightarrow right}(x; \Theta \backslash \theta_i) & \text{if } s(x;\theta_i) \rightarrow right\_child. \end{cases} \tag{4.3}$$

Having (4.1) and Equation (4.3), the reduced optimization problem at node n_i is given by

$$\sum_{x,y \in \mathbb{X},\mathbb{Y}} \min(L(y, G_{n_i \to left}(x; \Theta \backslash \theta_i)), L(y, G_{n_i \to right}(x; \Theta \backslash \theta_i))). \tag{4.4}$$

Since $L(y, G_{n_i \to left}(x; \Theta \backslash \theta_i)) \neq L(y, G_{n_i \to right}(x; \Theta \backslash \theta_i))$ only if the inequality (4.2) holds, we train the micro decision tree $t_i(x)$ based on the subset $(\mathbb{X}_{subset}, \mathbb{Y}_{subset})$ instead of using the entire training set. The optimization problem (4.4) has a closed-form solution as follows:

$$t_i^*(x) := \begin{cases} left\_child & \text{if } L(y, G_{n_i \to left}(x; \Theta \backslash \theta_i)) < L(y, G_{n_i \to right}(x; \Theta \backslash \theta_i)) \\ right\_child & \text{if } L(y, G_{n_i \to right}(x; \Theta \backslash \theta_i)) < L(y, G_{n_i \to left}(x; \Theta \backslash \theta_i)). \end{cases} \tag{4.5}$$

Equation (4.5) defines the optimal split function at the internal node n_i which is used to fit the micro decision tree t_i. With other nodes being fixed, we show that the overall loss function of G can be minimized by pursuing the optimal spilt function at an arbitrary internal node n_i. Rather than using a simple axis-aligned split, the proposed TnT algorithm learns a complexity-constrained decision tree to better approximate the optimal split function (Equation (4.5)).

Growing the graph from leaf nodes Growing from the leaf nodes is a standard practice in greedy training algorithms, where we recursively split the leaf nodes to achieve a deeper tree with a better fit on the training data [121]. In TnT, we replace the leaf predictors with decision trees. Let $G(\cdot; \Theta)$ be a decision graph and $n_l \in G$ an arbitrary leaf node with a constant class label $l(\cdot; \theta_l) = c$. Our goal is to minimize the overall loss function $L(\mathbb{Y}, G(\mathbb{X}; \Theta))$ by replacing the leaf predictor $l(\cdot; \theta_l)$ with a micro decision tree $t_l(x)$.

Consider the subset of samples $(\mathbb{X}_{subset}, \mathbb{Y}_{subset})$ that visit the leaf node n_l. Min-

imization of the loss function (4.1) can be expressed as

$$\min \sum_{\substack{x \in \mathbb{X}_{subset} \\ y \in \mathbb{Y}_{subset}}} L\left(y, t_l(x)\right), \tag{4.6}$$

where the minimum is simply achieved at $t_l^*(x) \coloneqq y$ for $x, y \in \mathbb{X}_{subset}, \mathbb{Y}_{subset}$ (i.e., the ideal leaf predictor). We build a decision tree to approximate the ideal leaf predictor.

Following the growing phase (Fig. 4.1(a)), the micro decision trees are merged into the decision graph (Fig. 4.1(b)). The nodes of the TnT decision graph are similar to those in the decision trees, where an internal node makes a single axis-aligned split and each leaf node contains a class label. In this chapter, we construct TnT decision graphs using axis-aligned splits. However, we do not limit the form of split functions ($s(\cdot;\theta)$) or leaf predictors $l(\cdot;\theta)$ in the TnT training process. For example, we could use logistic regression as the split function of the decision graph and micro trees to construct oblique TnT. In this case, θ refers to the trainable weights in logistic regression. **Therefore, various tree-based models could potentially benefit from the proposed TnT framework.**

4.3.3 Learning procedure

Unlike the learning procedures in [82, 112] which require a predefined tree structure, our proposed TnT algorithm grows a decision graph from a single leaf node. The training of TnT decision graphs is an iterative process that follows a *grow-merge-grow- $\cdots$-merge* alternation. Algorithm 3 shows the pseudocode to train a TnT decision graph. Lines 7-15 find the subset of data samples $\mathcal{X}_{subset}, \mathcal{Y}_{subset}$ that is sensitive to the internal split functions or leaf predictors at each node, and grow micro decision trees. In the internal nodes, $\mathcal{Y}_{subset}$ represents binary labels for the left or right child (i.e., not the label of the training set).

Algorithm 3: Tree in Tree (TnT)

Data: Training set $\mathcal{X}, \mathcal{Y}$
Result: TnT decision graph G fitted on the training set
1 $G \leftarrow$ initialize graph with a leaf node;
2 $infer(n_t, \mathcal{X}_t)$ denotes the forward inference of data $\mathcal{X}_t$ starting from node n_t ;
3 **for** $i_1 \leftarrow 1$ **to** N_1 **do**
4 **for** $i_2 \leftarrow 1$ **to** N_2 **do**
5 **for** *each node* $(n_i) \in G$ **do**
6 Samples that visit n_i: $\mathcal{X}_i, \mathcal{Y}_i \subset \mathcal{X}, \mathcal{Y}$;
7 **if** n_i *is an internal node* **then**
8 $\mathcal{Y}_{i,left} \leftarrow infer(n_i.left\_child, \mathcal{X}_i)$;
9 $\mathcal{Y}_{i,right} \leftarrow infer(n_i.right\_child, \mathcal{X}_i)$;
10 $index\_left \leftarrow (\mathcal{Y}_i = \mathcal{Y}_{i,left}$ **and** $\mathcal{Y}_i \neq \mathcal{Y}_{i,right})$;
11 $index\_right \leftarrow (\mathcal{Y}_i = \mathcal{Y}_{i,right}$ **and** $\mathcal{Y}_i \neq \mathcal{Y}_{i,left})$;
12 $\mathcal{X}_{subset}, \mathcal{Y}_{subset} \leftarrow$ copy samples from $\mathcal{X}_i, \mathcal{Y}_i$ at ($index\_left$ **or** $index\_right$);
13 $\mathcal{Y}_{subset}[index\_left] \leftarrow 0, \mathcal{Y}_{subset}[index\_right] \leftarrow 1$;
14 **else if** n_i *is a leaf node* **then**
15 $\mathcal{X}_{subset} \leftarrow \mathcal{X}_i, \mathcal{Y}_{subset} \leftarrow \mathcal{Y}_i$;
16 Grow a micro tree t_i on subset $\mathcal{X}_{subset}, \mathcal{Y}_{subset}$ in place of n_i;
17 Merge t_i into the current decision graph G for all nodes $(n_i \in G)$

Line 16 grows micro decision trees according to the growing phase of the TnT. Line 17 merges the trees into the graph structure.

Regularization Regularization is critical to limit model complexity and prevent overfitting of a decision tree and it is similarly required for TnT decision graphs. In the growing phase of a TnT (either from internal or leaf nodes), the subsets of samples $\mathcal{X}_{subset}, \mathcal{Y}_{subset}$ at different nodes may have various sizes. Therefore, we need a robust regularization technique to operate across all nodes of the TnT and to train the micro decision trees without overfitting on small subsets. In this work, we propose to use the sample-weighted cost complexity pruning approach [126, 127]. We prune micro decision trees by minimizing $R(t_i) + C_i|t_i|$, where $R(t_i)$ is the misclassification measurement and $|t_i|$ denotes

the tree complexity. We calculate $R(t_i)$ using Gini impurity and measure $|t_i|$ by counting the number of splits [121]. C_i is the sample-weighted regularization coefficient calculated by

$$C_i = C\frac{\#\mathcal{X}}{\#\mathcal{X}_{subset,i}}, \tag{4.7}$$

where $\#\mathcal{X}_{subset,i}$ is the sample count of subset at node n_i. C is a hyperparameter of the TnT and is used to control the pruning strength and tune the model complexity (# splits). For a smaller subset, we need to apply a stronger cost complexity pruning to prevent overfitting.

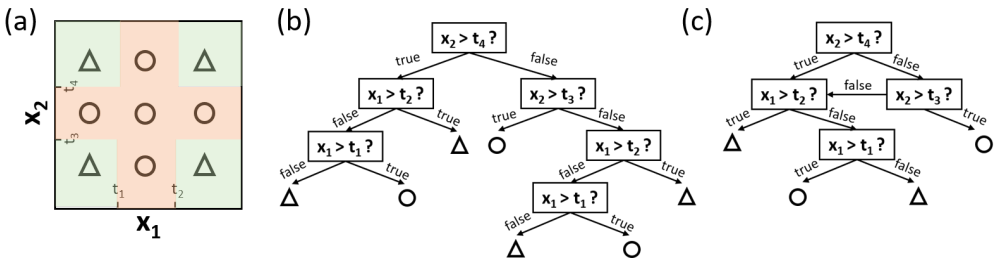

Figure 4.2: Comparison of DT and TnT decision graph on synthetic data; (a) A toy classification task with desired axis-aligned boundaries. x_1, x_2 and $t_1 - t_4$ denote two features and four thresholds, respectively. Different markers represent binary class labels. (b) A decision tree requires at least six splits to classify the data. (c) A TnT decision graph only requires four binary splits on the same task.

Fine-tune and post pruning The TnT decision graphs are compatible with Tree Alternating Optimization (TAO [82]), previously proposed to optimize decision trees. We used TAO to fine-tune the TnT decision graphs, which led to slight improvements in classification accuracy. A post pruning process is further applied to TnT decision graphs to remove the dead nodes. A node is pruned if no training samples travel through that node. Post pruning can result in a more compact decision graph and reduce the number of splits without affecting the training accuracy.

Time complexity Compared to decision trees, decision graphs offer an enriched model structure, which increases the complexity of learning the graph structure. Previous work constructed decision graphs by tentatively merging two leaf nodes at each training step, with a time complexity of $O(N_l^2)$, where N_l is the number of leaf nodes [122]. Since the proposed TnT algorithm generates new splits by growing micro decision trees inside the nodes, the dataset is initially sorted in $O(mklog(m))$ for m samples and k features. The time complexity for creating a new split depends on the dataset (i.e., $O(mk)$) and not on the size of the graph. As the graph grows larger, the TnT algorithm optimizes each node for $N_1 * N_2$ times in the worst case (Algorithm 3). Since N_1 and N_2 are hyperparameters that were fixed in this work ($N_1 = 2, N_2 = 5$, the choice of N_1 and N_2 will be discussed in the following section), TnT exhibits a linear time complexity to the number of nodes, $O(nmk + mklog(m))$ with n being the number of nodes. Testing our Python implementation on an Intel i7-9700 CPU, it took 325.3 seconds to build a TnT of 1k splits on the MNIST dataset (60k samples, 784 features, 10 classes).

Synthetic data We first construct a synthetic classification dataset to show the potential benefits of TnT over conventional decision tree algorithms (e.g., CART). Figure. 4.2(a) visualizes the two-dimensional data distribution with one class on the corners and the other class elsewhere. To achieve optimal decision boundaries, a conventional decision tree requires six splits (Fig. 4.2(b)), whereas TnT only requires four splits to generate the same decision boundaries (Fig. 4.2(c)). **By sharing nodes among different decision paths in a graph, TnT enables a more compact model with fewer splits compared to a conventional DT.**

Table 4.1: Comparison of TnT and CART at optimal split count (#S, determined by cross-validation). Mean test accuracy (±standard deviation) are calculated on 5 independent trials.

model	MNIST		Connect-4		Letter		Optical recognition	
	accuracy	#S	accuracy	#S	accuracy	#S	accuracy	#S
TnT	**90.87±0.31**	**600**	**78.85±0.46**	**864**	**86.62±0.02**	**1.2k**	**86.32±0.24**	**174**
CART	88.59±0.14	1.1k	77.23±0.01	931	86.26±0.15	1.3k	85.56±0.46	193
model	Pendigits		Protein		SenseIT		USPS	
	accuracy	#S	accuracy	#S	accuracy	#S	accuracy	#S
TnT	**92.61±0.53**	**125**	**57.26**	**69**	**80.48±0.42**	**198**	**88.76±1.36**	**31**
CART	91.74±0.13	166	55.30	76	79.40	345	87.35±0.15	109

4.4 Experiments: TnT as a stand-alone classifier

We test the TnT decision graph as a stand-alone classifier and benchmark it against several state-of-the-art decision tree/graph algorithms with axis-aligned splits, including classification and regression trees (CART [121]), tree alternating optimization (TAO [82]), and the naive decision graph (NDG [122]). We also implement the TnT algorithm in two different settings: with or without fine-tuning. We observe that the proposed TnT algorithm consistently achieves a superior performance under similar complexity constraints on multiple datasets.

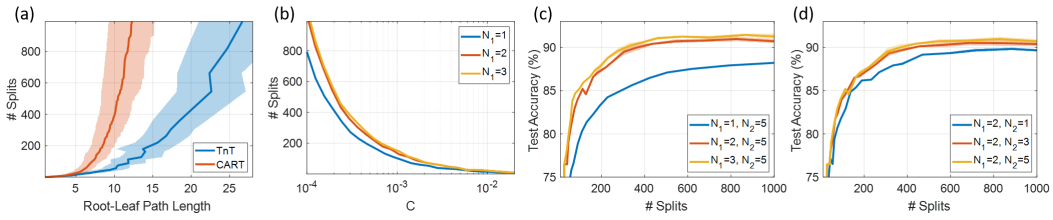

Figure 4.3: (a) The number of splits as a function of the root-leaf path length. The standard deviation across different samples is shown by shaded areas. (b) The number of splits vs. regularization coefficient C. (c, d) Test performance using different hyperparameter settings on the MNIST dataset. The default setting ($N_1 = 2, N_2 = 5$) is plotted in both figures for comparison.

In the worst-case scenario, the number of nodes increases exponentially with

the depth of a tree, which prevents DTs from growing very deep. However, this limitation does not apply to TnT decision graphs. Figure 4.3(a) illustrates the average length of the root-leaf path as a function of model complexity for TnT and CART. With 1000 splits, the average decision depth of the best-first CART is 12.3, whereas the TnT decision graph has a mean depth of 27.3. In the best-first decision tree induction, we add the best split in each step to maximize the objective [128]. **Therefore, TnT can achieve a much "deeper" model without significantly increasing the number of splits.** The regularization coefficient C is used to control the complexity of decision graphs in TnT. The number of splits decreases as we increase the pruning strength C (Fig. 4.3(b)). Figures 4.3(c, d) compare the effect of different hyperparameter settings (N_1, N_2). **We note that the proposed TnT decision graph is a superset of decision trees and that TnT can reduce to a DT learning algorithm under certain conditions.** With $N_1 = 1$, Algorithm 3 replaces a single leaf node with a decision tree, which is equivalent to training a CART with cost complexity pruning. In general, higher values of N_1 and N_2 can lead to a better classification performance. In the following experiments, we set the hyperparameters as $N_1 = 2, N_2 = 5$. A marginal improvement in classification performance can be obtained by increasing N_1 and N_2, at the cost of increased training time.

Figure 4.4 compares the proposed TnT decision graphs with axis-aligned decision trees/graphs previously reported. We include the following datasets: MNIST, Connect-4, Letter, Optical reconstruction, Pendigits, Protein, SenseIT, and USPS from the UCI machine learning repository [130] and LIBSVM Dataset [131] under Creative Commons Attribution-Share Alike 3.0 license. The statistics of datasets including the number of train/test instances, number of attributes, and number of classes are shown in Fig. 4.4. If a separate test set is

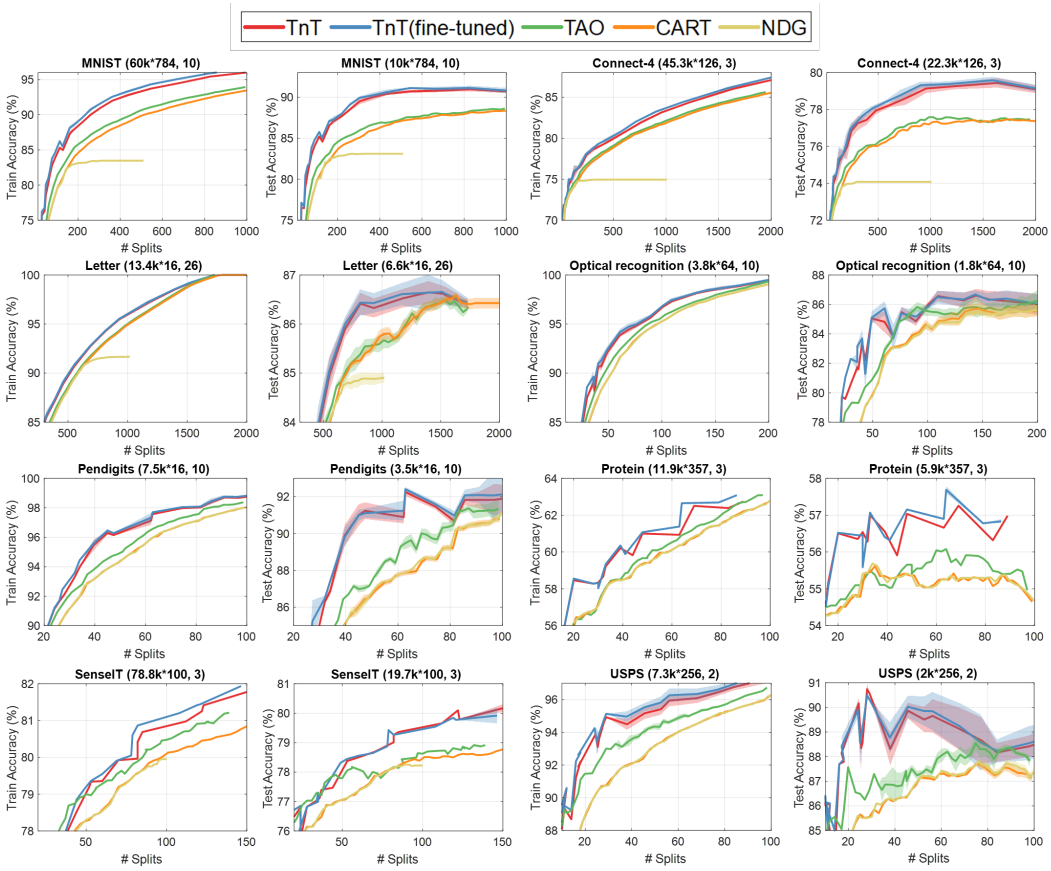

Figure 4.4: Model comparison in terms of train and test accuracy on multiple classification tasks. The following axis-aligned decision trees/graphs are included: **TnT (this work):** We implement the proposed TnT decision graph at various complexity levels. Hyperparameters are fixed at $N_1 = 2, N_2 = 5$ on all tasks. **TnT (fine-tuned):** The alternating optimization algorithm is used to fine-tune the TnT. **TAO:** The tree alternating optimization algorithm is applied to axis-aligned decision trees [129]. **CART:** Classification and regression trees trained in a best-first manner to assess the optimal tree structure under certain complexity constraint [121, 128]. **NDG:** The naive decision graph trained with Algorithm 2 [122]. The complexity penalty is fixed at $C = 3e - 4$ on all tasks. Dataset statistics are indicated on top of each figure with the following format (# Train/Test samples * # Features, # Classes).

not available for some tasks, we randomly partition 33% of the entire data as test set. For all models, we repeat the training procedure five times with different random seeds. The mean classification accuracy is plotted in Fig. 4.4 with shaded area indicating the standard deviation across trials. The proposed Tree in Tree (TnT) algorithm outperforms axis-aligned decision trees such as TAO [82, 129] and CART [121], as well as NDG which is also based on axis-aligned decision graphs [122]. We also present the results for TnT(fine-tuned), which employs alternating optimization to fine-tune the TnT and slightly improve the classification performance.

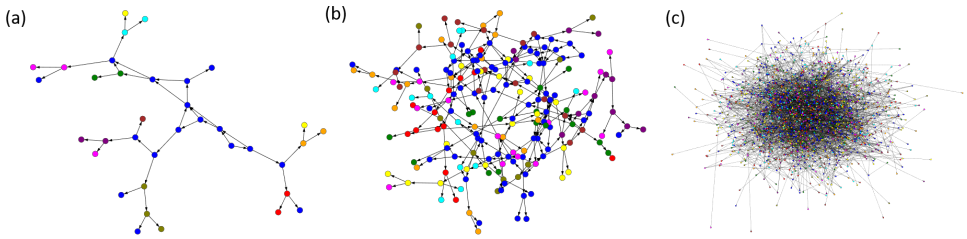

Figure 4.5: Visualization of TnT decision graphs at various complexity levels. (a) TnT with 20 internal nodes and 16 leaf nodes (train/test accuracy: 70.41%/71.75% on MNIST classification task). (b) 129 internals and 75 leaves (train/test accuracy: 85.54%/85.49%). (c) 1046 internals and 630 leaves (train/test accuracy: 96.04%/90.56%). Different node colors represent dominant class labels (more than 50% of samples belong to the same class). Nodes are shown in blue if no dominant class is found.

Visualization Similar to decision trees, TnT decision graphs enjoy a fully interpretable and visualizable decision process. Figures 4.5(a-c) visualize the TnT decision graphs with 20, 129, and 1046 splits, respectively. We use different node colors to indicate the dominant class labels. A node will have a dominant class if most samples at that node belong to the same class. We show the nodes in blue if class labels are mixed (i.e., no class label contributes to greater than 50% of the samples visiting that node). As the graph grows larger, TnT performs better on the MNIST dataset, achieving improved classification accuracy on both training

and testing sets.

4.5 Experiments: TnT in the ensemble

Decision trees are widely used as base estimators in ensemble methods such as bagging and boosting. Random Forests apply a bagging technique to decision trees to reduce variance [132], in which each base estimator is trained using a randomly drawn subset of data with replacement [133]. As opposed to bagging, boosting is used as a bias reduction technique where base estimators are incrementally added to the ensemble to correct the previously misclassified samples. Popular implementations of the boosting methods include AdaBoost [134] and gradient boosting [83, 109]. Both AdaBoost and bagging use classifiers as base estimators, whereas the gradient boosting methods require regressors [83, 109]. Although we argue that the proposed TnT algorithm can be applied to regression tasks with a slight modification in the objectives, it is beyond the scope of this chapter to demonstrate TnTs as regressors.

Here, we used the TnT decision graphs as base estimators in the bagging (TnT-bagging) and AdaBoost (TnT-AdaBoost) ensembles. **Our goal is to replace decision trees with the proposed TnT classifiers in ensemble methods and compare the performance under various model complexity constraints.** The ensemble methods are implemented using the scikit-learn library in Python (under the 3-Clause BSD license) [135]. We change the ensemble complexity by tuning the number of base estimators (# E) and the total number of splits (i.e., internal nodes, # S). Note that TnT has additional hyperparameters that do not apply to decision trees, such as N_1 and N_2. We set the hyperparameters as $N_1 = 2, N_2 = 5$ throughout the experiments so that the TnT and tree ensembles share a similar hyperparameter exploration space.

Table 4.2: Comparison of TnT-based ensembles with conventional random forest and AdaBoost. Mean train and test accuracy (± standard deviation) are calculated across 5 independent trials. We tune the ensemble size (# E, the number of base estimators) and splits count (# S) to change the complexity of the ensemble. Dataset statistics are given in the format: Dataset name (# Train/Test samples * # Features, # Classes).

	model	# E	# S	train	test		# E	# S	train	test
MNIST (60k/10k*784, 10)	TnT-bagging	5	4.8k	**97.46±0.16**	**93.65±0.24**	Connect-4 (45.3k/22.3k*126, 3)	5	4.6k	**84.42±0.19**	**80.61±0.18**
	Random Forest	5	4.8k	96.55±0.36	92.31±0.57		5	4.6k	83.60±0.12	79.21±0.19
	TnT-AdaBoost	5	640	**90.26**	88.38		5	450	**77.75±0.16**	**77.39±0.19**
	AdaBoost	5	640	89.75	**88.61**		5	450	77.28	76.74
	TnT-bagging	10	9.6k	**98.28±0.06**	**94.92±0.20**		10	9.2k	**85.11±0.05**	**81.44±0.14**
	Random Forest	10	9.6k	97.44±0.18	93.64±0.38		10	9.2k	84.21±0.12	79.85±0.20
	TnT-AdaBoost	10	1.4k	**95.09±0.09**	**92.36±0.13**		10	940	**80.10±0.23**	**78.94±0.29**
	AdaBoost	10	1.4k	94.28	91.49		10	940	79.69	78.37
	TnT-bagging	20	19.2k	**98.64±0.06**	**95.57±0.14**		20	18.3k	**85.66±0.12**	**81.93±0.13**
	Random Forest	20	19.2k	97.90±0.12	94.36±0.19		20	18.3k	84.57±0.08	80.39±0.09
	TnT-AdaBoost	20	2.9k	**98.03±0.11**	**94.49±0.21**		20	1.8k	82.46±0.41	80.53±0.50
	AdaBoost	20	2.9k	97.70	94.04		20	1.8k	**82.77**	**81.14**

Table 4.2 compares the performance of TnT ensembles with that of decision tree ensembles on two datasets. Since the bagging method can effectively reduce variance, we use large models (i.e., TnTs/decision trees with many splits) as the base estimator. On the contrary, TnTs/decision trees with few splits are used in the AdaBoost ensemble, given that boosting can decrease the bias error. According to Table 4.2, TnT-bagging is almost strictly better than Random Forest under the same model complexity constraints, indicating that TnT decision graphs outperform decision trees as base estimators. TnT-AdaBoost also outperforms AdaBoost in most cases, showing the advantage of TnT over decision trees. However, we observe a few exceptions in the TnT-AdaBoost vs. AdaBoost comparison, as weak learners with high bias (e.g., decision stumps) are also suitable for boosting ensembles. Overall, the TnT ensembles (TnT-bagging, TnT-AdaBoost) achieve a higher classification accuracy compared to decision trees when used in similar ensemble methods (Random Forest, AdaBoost).

4.6 Discussions

Broader impact Recently, the machine learning community has seen different variations of decision trees [82, 87, 112–114, 119, 120]. In this chapter, we present the TnT decision graph as a more accurate and efficient alternative to the conventional axis-aligned decision tree. However, the core idea of TnT (i.e., growing micro trees inside nodes) is generic and compatible with many existing algorithms. For example, linear-combination (oblique) splits can be easily incorporated into the proposed TnT framework. Specifically, we can grow oblique decision trees inside the nodes to construct an oblique TnT decision graph. In addition to oblique TnTs, the proposed TnT framework is also compatible with regression tasks. As suggested in [112], we may grow decision tree regressors (rather than DT classifiers) inside the leaf nodes to construct TnT regressors, which remains as our future work. Overall, our results show the benefits of extending the tree structure to directed acyclic graphs, which may inspire other novel tree-structured models in the future.

Limitations The proposed TnT decision graph is scalable to large datasets and has a linear time complexity to the number of nodes in the graph. However, the training of TnT is considerably slower than CART. The current TnT algorithm is implemented in Python. It takes about 5 minutes to construct a TnT decision graph with ~1k splits on the MNIST classification task (train/test accuracy: 95.9%/90.4%). Training a CART with the same number of splits requires 12.6 seconds (train/test accuracy: 93.6%/88.3%). TnT has a natural disadvantage in terms of training time since each node is optimized multiple times (in this work $N_1 * N_2 = 10$), similar to other non-greedy tree optimization algorithms (e.g., 1-4 minutes for TAO [82]). The Python implementation may also contribute to the

slow training, and we expect that the training time would significantly improve with a C implementation. We also observe that TnT decision graphs have longer decision paths compared to CART (Figure 4.3(a)), which may raise a concern on increased inference time.

Parallel implementation Algorithm 3 presents a sequential algorithm to construct TnT decision graphs by visiting the nodes in the breadth-first order. However, it is also possible to concurrently grow micro decision trees inside multiple nodes, which could lead to a parallel implementation of TnT. Specifically, only those nodes in the graph that are non-descendant of each other can be optimized in parallel. Parallel optimization is not applicable to the nodes on the same decision path, since the parent node optimization may alter the samples visiting the child node. The parallel optimization of non-descendant nodes follows the separability condition of TAO [82, 112]. The separability condition also holds for the proposed TnT decision graph, enabling a parallel implementation.

4.7 Conclusion

In this chapter, we propose the Tree in Tree decision graph as an effective alternative to the widely used decision trees. Starting from a single leaf node, the TnT algorithm recursively grows decision trees to construct decision graphs, extending the tree structure to a more generic directed acyclic graph. We show that the TnT decision graph outperforms the axis-aligned decision trees on a number of benchmark datasets. We also incorporate TnT decision graphs into popular ensemble methods such as bagging and AdaBoost, and show that in practice, the ensembles could also benefit from using TnTs as base estimators. Our results suggest the use of decision graphs rather than conventional deci-

sion trees to achieve superior classification performance, which may potentially inspire other novel tree-structured models in the future.

CHAPTER 5

UNSUPERVISED TEST-TIME ADAPTATION FOR ROBUST GAIT DECODING IN PATIENTS WITH PARKINSON'S DISEASE

5.1 Introduction

Neural prostheses with chronic sensing capability have drawn increasing interest for long-term monitoring of brain activity in emerging applications such as closed-loop stimulation and brain-machine interfacing [136–140]. As a key component of closed-loop prostheses, neural decoders use the brain activity to control neurostimulators for treating neurological and psychiatric disorders (e.g., Parkinson's disease [2, 110, 141, 142], epilepsy [12, 31], treatment-resistant depression [143, 144]), to control the movement of exoskeletons in spinal cord injury [145], and so on. However, while closed-loop prostheses have been well demonstrated in laboratory settings, achieving a clinically reliable solution for the daily lives of patients remains a challenge. Closing the gap requires high accessibility of neural data recorded from fully implantable devices to enable the design of more advanced neural decoders. Such chronic decoders should be robust to fluctuations in neural signals without relying on frequent recalibrations.

Deep brain stimulation (DBS) is a well-established neuromodulation therapy to alleviate cardinal motor symptoms in Parkinson's disease (PD) and essential tremor (ET) [142, 146–148]. Recent work further built machine learning (ML) models for tremor detection and adaptive DBS in PD [2, 93, 110, 149], lower-limb movement decoding in PD [141], and postural tremor detection in ET [150], all using local field potentials (LFPs) recorded from DBS leads. Thanks to the recent advance of implantable neurostimulators, LFP acquisition from patients

with PD is becoming more accessible in real-life conditions. For example, the Percept PC (Medtronic PLC, USA) is capable of wireless streaming of LFPs from up to 6 channels (2 during stimulation). Chronic LFPs differ from those recorded via externalized leads in that wirelessly-streamed LFPs are sampled at a lower frequency due to limited telemetry bandwidth (250 Hz [138] vs. 2048 Hz [2]), making a critical difference between in-lab demonstrations and practical applications. The improved access to chronic data also motivates the use of neural decoders in more complex environments, where patients may receive dopaminergic medications and/or DBS under various conditions. In Parkinson's disease, for instance, ML-based neural decoders can predict motor symptoms or STN-encoded voluntary movements using chronic LFPs recorded via devices such as Percept PC or Summit RC+S. Next-generation neural decoders must operate under DBS and typical medications that may alter the dynamics of LFP signal. Current neural decoders are highly restrictive in the aforementioned aspect, generally use simple machine learning models, and have only been verified in acute and highly stable recording conditions. There is a pressing need for more advanced decoders that could operate reliably on chronic recordings in a variety of therapeutic settings.

Conventional neural decoders with fixed model parameters are vulnerable to fluctuations in neural signal (i.e., domain shift), including the electrode movements or dysfunction [151], cross-session/subject variation [152, 153], or signal drift over time [154]. Some recent works have proposed to alleviate this issue by employing adaptive decoders capable of online parameter update [154–156]. However, current methods have made strong assumptions on the pattern of domain shift, which may not be guaranteed in complicated clinical settings applied to PD (see Methods). Adaptive neural decoders must handle

clinical domain shifts, including both abrupt changes (e.g., DBS turned on/off) and continuous shifts over time (e.g., due to intake of medications). Furthermore, such decoders should ideally cause only marginal computational overhead compared to fixed decoders. This is critical for both on-device implementation of decoders to enable low-latency closed-loop operation [157], and for reduced training complexity in software-based externally-controlled closed-loop systems. Test-time adaptation of ML models is a completely unsupervised process and requires no prior knowledge of the target domain during training [158]. Recent approaches have demonstrated successful image classification in non-stationary environments by adapting model parameters entirely during test-time [159, 160]. While test-time adaptation is an emerging technique for building domain-invariant computer vision algorithms, to the best of our knowledge it is not yet explored in neural signal processing applications.

In this chapter, we study the cross-session decoding of movement states in freely moving patients with Parkinson's disease. All patients were implanted with Percept PC neurostimulators, which enabled chronic, wireless monitoring of LFPs. We designed experimental tasks involving sitting, standing and walking conditions, and recorded LFPs under diverse clinical settings such as various DBS configurations and medication-based therapies. We show that the fluctuations in LFP biomarkers induced by PD therapies significantly degraded the performance of conventional decoders with fixed parameters. To address this issue, we proposed an adaptive machine learning-based decoder that performs unsupervised adaptation to new therapeutic settings during test time. We evaluated the proposed model on gait decoding tasks across multiple clinical sessions and found that test-time adaptation could significantly improve the decoder stability. With the proposed approach, we show the possibility to

chronically decode gait cycles with high accuracy even in the presence of major therapy-induced domain shifts.

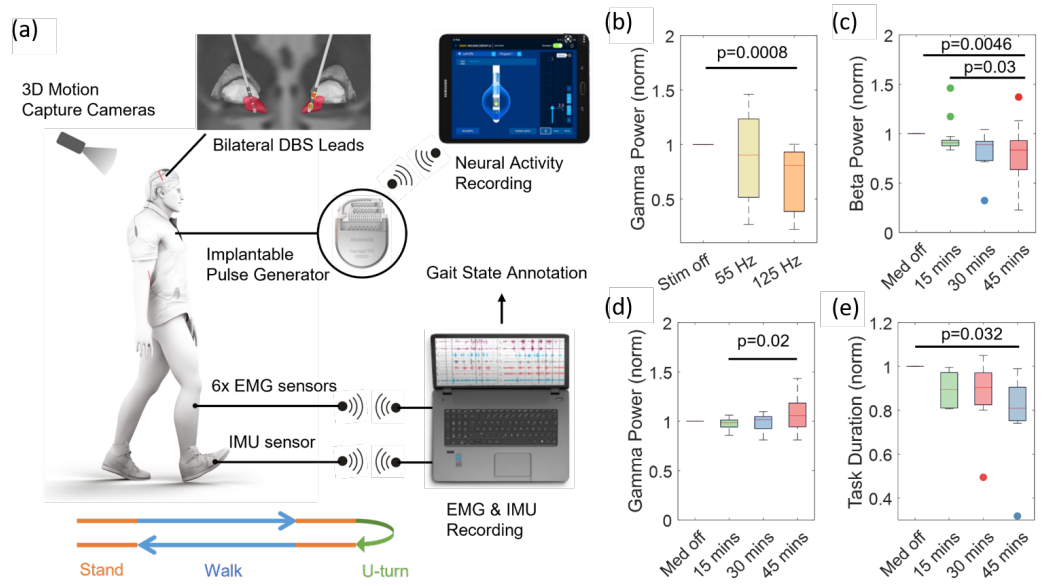

Figure 5.1: Experimental paradigm and therapy-induced domain shift in patients with PD. (a) Patients were instructed to perform lower limb movement tasks during which gait states were identified as stand, walk, and U-turn. Neural activities in the form of bipolar LFPs were recorded through bilateral DBS leads. The implantable pulse generator is capable of wireless transmission of LFP recordings. Wireless EMG and IMU sensors were placed over patients' legs (ankle: Tibialis Anterior, Medial Gastrocnemius, Lateral Gastrocnemius; knee: Vastus Lateralis, Semitendinosus; hip: Rectus Femoris) and feet to annotate the gait state. (b) Stimulation-induced domain shift during gait tasks (n = 12 patients). Gamma bands significantly deviate from Stim off after removing the stimulation artifacts. (c-e) Medication-induced domain shift during gait tasks (n = 11 patients). Dopaminergic medication significantly reduces patients' Beta-band power, whereas Gamma power increases with medication. At 45 minutes after medication, patients need significantly less time to perform the aforementioned gait task.

5.2 Methods

5.2.1 Study design

All experiments were approved by the Ethical Committee of the Canton de Vaud, Switzerland (Reference PB_2017-00064). Informed consent was obtained from each participant once the nature and possible consequences of the study were explained. Patients were predominantly in the off-medication condition (>12 hours) before the start of the experiments, although patients with high severity of disease retained some dopamine agonist medication.

5.2.2 LFP recordings

Local field potentials were recorded using the sensing capabilities of the Percept PC (sampling frequency at 250 Hz). The LFP was recorded in resting condition (sitting or standing) in the Indefinite Streaming mode, capturing all three bipolar contact pairs per hemisphere. Recordings during motor tasks were obtained in the Brainsense Streaming mode, which is restricted to one contact pair per hemisphere. We only used the recordings during motor tasks for gait decoding, whereas recordings in resting condition were used to identify artifacted channels (see artifact identification below). Prior to the first recording session for each patient, we performed a Brainsense Survey and visually inspected the power spectral density (PSD) of each pair. We selected the contact pair with the highest Beta power and retained that pair for all subsequent experiments of that patient. This selection was motivated by the fact that not all patients exhibited other frequency bands systematically (Alpha, Gamma). To ensure minimal bias in the selection, we verified that adjacent contacts did not exhibit radically different bands and modulations. We further ensured that the selected contact pair was not labeled as "artifacted" by the system. Synchronization with external de-

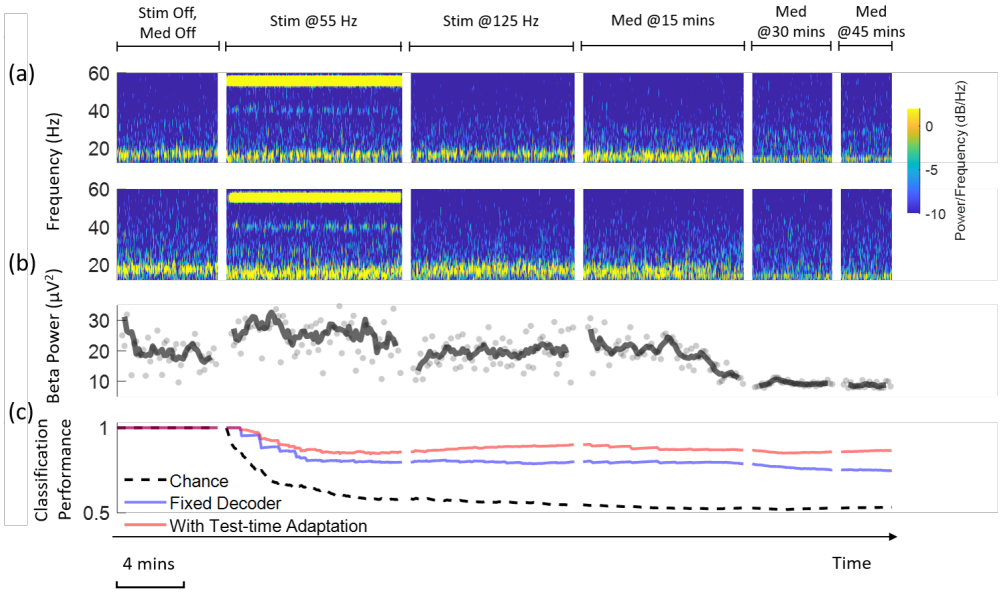

Figure 5.2: Cross-session gait decoding in a PD patient (PD4) using chronic LFP recordings. The patient received bilateral deep-brain stimulation leads (Medtronic 3389) and was recorded within five days after the surgery. We recorded bipolar LFPs from both hemispheres as the patient performed gait experiments. The patient received various interventional therapies during different recording sessions, including a baseline therapy-free session (Stim off, Med off), low-frequency DBS (Stim @55 Hz), high-frequency DBS (Stim @125 Hz), and high-frequency DBS with dopaminergic medication (Med @15/30/45 mins). Consecutive sessions were recorded at 15-minute intervals. (a) Spectrogram of LFPs from the right (top) and left (bottom) hemispheres. The task duration reduces after the intake of medication. (b) Beta-band activity under various therapeutic conditions. Each dot represents a 10-second window, while the line shows the smoothed curve. PD medication reduced Beta-band activity in LFP. (c) Cross-session decoding performance. Both fixed (baseline) and adaptive decoders are trained on the therapy-free session (Stim off, Med off) and predict gait movements in the subsequent sessions. Compared to the fixed decoder, our adaptive decoder is more robust against domain drift.

vices was performed by applying a transient DBS burst (130 Hz, 60 μs, 1 mA) at the beginning and end of each recording, which induced artifacts in an external electromyographic sensor placed on the chest of the patient in the vicinity of the IPG [138]. Recordings were all performed in the Brainsense Survey mode. Stimulation was on at either 0 mA (Stim Off condition), at the therapeutic amplitude tuned by an expert neurologist (when DBS was delivered at 125 Hz) or at the same charge delivery when stimulating at 55 Hz.

5.2.3 Artifact identification

Gait-related artifacts affect neural signal predominantly in the low frequencies but can also spread to higher frequencies, making them difficult to be removed through standard filtering techniques. Rather than aiming to identify the artifacts in the time domain, we reasoned that corrupted channels would exhibit important differences in the aperiodic (1/f) component of the power densities (PSDs) between rest and walking. The aperiodic component captures the overall baseline power across the spectrum and should not change significantly over consecutive trials (task-related modulations are expected to be captured in the activity over periodic frequency bands) [141].

For each patient, we extracted the PSDs of two separate recordings from the same session, one at rest (sitting) and the other one during walking. We further applied a fitting algorithm to split the PSDs into task-related modulations and acyclic (1/f) components that are task-invariant [161]. We then compared their aperiodic (1/f) components by computing the root mean square error (RMSE) in the range of 10 to 90 Hz (i.e., region of interest) to ensure that walking did not induce an increase above 50% in the 1/f power compared to sitting (difference of 1.76 dB). All channels exceeding this value were considered corrupted in the re-

gion of interest and patients with at least one corrupted channel were discarded from further analyses for decoding purposes. Visual inspection of the spectrogram for channels labeled as "artifacted" showed important movement-related low-frequency spikes, aligned to the timing of foot strikes, which periodically corrupted the spectrogram and spread to higher frequencies. All retained channels were also verified by visual inspection of their spectrogram. Overall, N=4 participants exhibited at least one corrupted channel.

For the remaining channels, the stimulation artifacts were further identified and removed from the LFP recordings. During stimulation therapy, DBS imposed a strong artifact at the stimulation band. We removed the stimulation artifact using a 3rd-order Butterworth filter. To remove DBS artifact at 55 Hz, we used a band-stop filter with cutoff frequencies of 52-58 Hz and 107-113 Hz. The 52-58 Hz corresponds to the stimulation band and 107-113 Hz corresponds to the second harmonic of the stimulation artifact. With DBS at 125 Hz, the filter was configured as low-pass with a cutoff frequency of 122 Hz.

5.2.4 Biomechanical recordings during gait

Kinematics: Patients were recorded in a gait lab using an optoelectronic motion capture system (Vicon, UK) that measured the 3D positions of key body joints. Kinematic data was complemented by bilateral triaxial inertial measurement unit (IMU) sensors (Delsys, MA, USA) attached to the patient's shoes, recording raw gyroscope signals from the right and left feet (sampling frequency: 148Hz).

Electromyographic signals: EMG signals were recorded using a wireless system operating at 2 kHz (Delsys, USA). Sensors were placed bilaterally according to SENIAM guidelines (Surface Electro-MyoGraphy for the Non-Invasive

Assessment of Muscles, www.seniam.org) on agonist and antagonist muscles of the ankle joint (TA Tibialis Anterior, MG Medial Gastrocnemius, LG Lateral Gastrocnemius), knee joint (VM Vastus Medialis, ST Semitendinosus) and hip joint (RF Rectus Femoris). EMG sensors were covered using protective tape (Tegaderm) to prevent them from moving during walking, and to reduce friction with the suit. For all patients, an additional EMG sensor was placed on the chest for synchronization purposes.

5.2.5 Feature engineering and epoch annotation

After data preprocessing and artifact removal, we segmented the LFP recordings into 1-second overlapping epochs with a step size of 200 ms. Following our earlier work on PD tremor [2, 93, 149] and lower limb movement decoding [141] from LFP, we extracted the following features from each epoch: Hjorth parameters (Activity, Mobility, and Complexity) and spectral power over 10-120 Hz with a bin resolution of 4 Hz. The Hjorth parameters are calculated through the following equations and represent the statistical characteristics of a signal in the time domain:

$$\text{Activity}(e) = \frac{1}{N}\sum_{n=1}^{N}(e_n - \mu)^2 \tag{5.1}$$

$$\text{Mobility}(e) = \sqrt{\frac{\text{Activity}(\Delta e)}{\text{Activity}(e)}} \tag{5.2}$$

$$\text{Complexity}(e) = \frac{\text{Mobility}(\Delta e)}{\text{Mobility}(e)} \tag{5.3}$$

where e represents an LFP epoch with N samples, μ and Δe are the mean value and first derivative of e, respectively. We applied a log transform to all extracted biomarkers and standardized the feature vectors.

We next annotated each epoch using the gyroscope signals recorded from

the right and left feet during a walking sequence (an example of a gyroscope signal is depicted in Fig 5.5). Since the raw gyroscope measurements are continuous, we used a two-level thresholding approach to discretize the gait state into binary classes of "Stand" or "Move". The 3D motion capture camera and EMG signals were used to validate the movement state. Specifically, we first normalized the gyroscope measurements from both feet by removing the mean value and scaling the signals to their unit variance. We annotated an epoch as "Stand" if the gyroscope signals from both feet were lower than the mean value (i.e., <0). Epochs were labeled as "Move" if either foot had a gyroscope signal one standard deviation above the mean (i.e., >1). Gyroscope signals between 0 and 1 were annotated as "Marginal" since they were caused by small unintentional movements. The "Marginal" epochs were considered noisy and excluded from the training process. However, "Marginal" epochs were still used for adaptive inference since excluding them requires access to ground-truth information (gyroscope signals) that are unavailable during test time.

5.2.6 Contextual Meta Adaptation

We propose Contextual Meta Adaptation (CMA) to dynamically update decoder parameters at the test time. Compared to previous adaptive neural decoders, CMA is unique from the following aspects: First, CMA makes minimal assumptions on the test domain. Different from conventional online learning or few-shot learning settings, CMA operates in a purely unsupervised manner. During inference, CMA adapts to an unseen shifted domain by only using unlabeled LFP recordings. Second, CMA is similar to unsupervised domain adaptation in that both approaches do not require labeled data from the test domain. We further design CMA to adaptively handle a stream of LFP epochs at

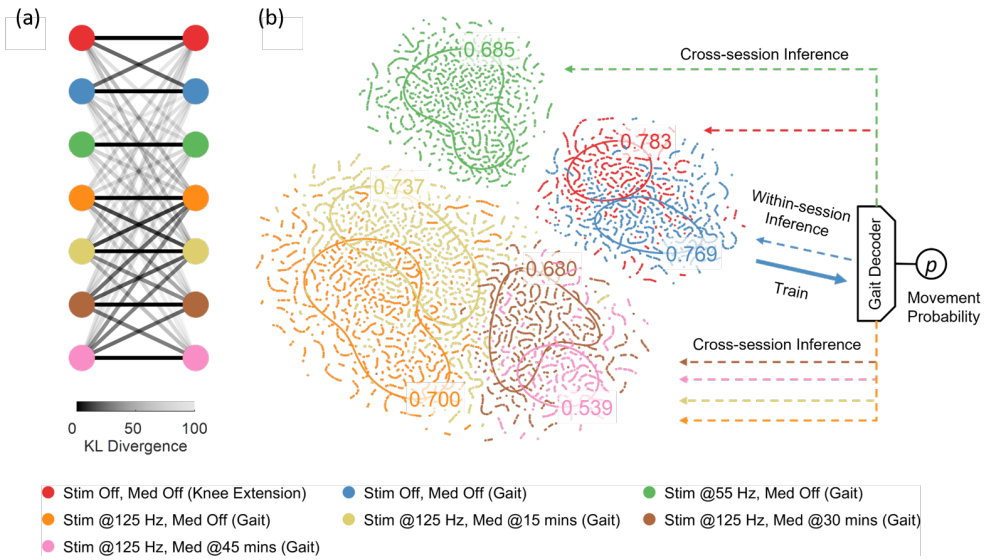

Figure 5.3: Shifted distributions and domain divergence of LFP recordings from different sessions. (a) The Kullback–Leibler divergence (KL divergence) is used to quantify the distance of LFP distributions between each pair of recording sessions. We used feature vectors as the representation of epochs and epochs from the same session were fitted into a multivariate Gaussian distribution. A larger KL divergence indicates a greater difference in data distributions. (b) t-SNE visualization of LFP recordings from various sessions. We used colors to represent different recording sessions (same as in (a)) and each dot indicates an LFP epoch. Decoders were trained on the baseline gait session with no stimulation or medication (blue). During inference, the epochs were either from the same training session (within-session inference) or from different sessions (cross-session inference). The contours represent the Gaussian fit and the numbers show the classification performance (AUC) of the fixed decoder. The performance of the fixed decoder decreases if the test distribution deviates from the training distribution.

the test time. Compared to unsupervised domain adaptation, CMA uses a more realistic training setting without making any assumptions on LFP drift due to PD therapies. Third, CMA is computationally efficient. The adaptation process does not involve computationally intensive operations such as retraining on the source data or gradient-based parameter updates. The inference phase of CMA is relatively lightweight and adds minimal computational overhead to its fixed counterparts.

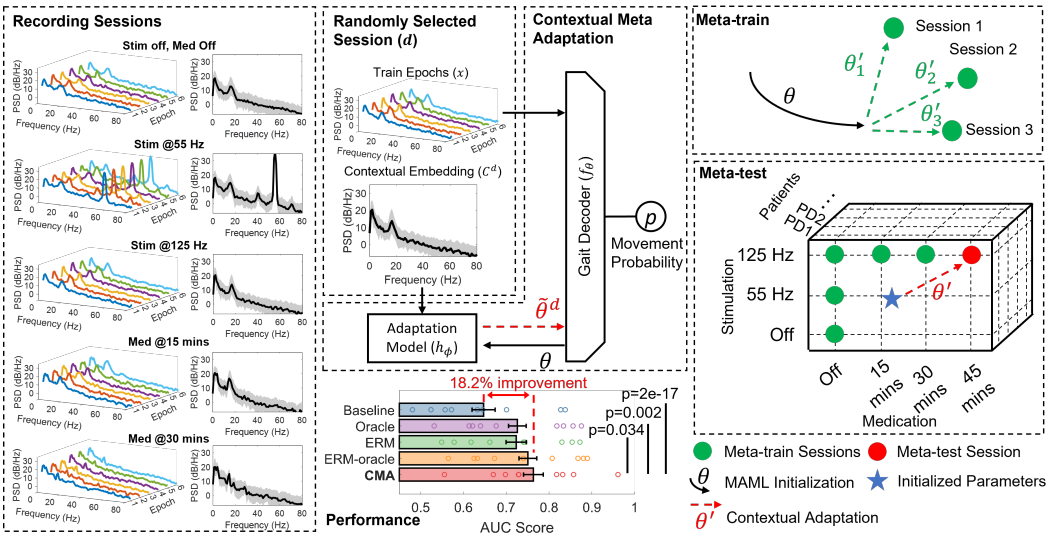

Figure 5.4: The model structure of the proposed Contextual Meta Adaptation (CMA). We used multiple sessions with different therapeutic settings for training the decoders. We calculated the contextual embedding C_i to reflect the statistics of each session and to allow us to make predictions using only the current LFP epoch (x_i). The gait decoder (f_θ), parameterized by θ, is initialized using model-agnostic meta-learning. Similar to conventional gait decoders, CMA takes LFP epochs as input. Cross-session adaptation is enabled by a separate adaptation model which adjusts the parameters of the decoders to new, unseen sessions. We benchmarked CMA with other ML models on the cross-session gait decoding task, including the current clinical practice with the fixed decoder trained on the therapy-free session (Baseline), the fixed decoder trained with labeled data from the test session (Oracle), a strong multi-task learning baseline (ERM), and ERM leveraging the test session information (ERM-oracle). CMA improves the average AUC score by 18.2% compared to the current clinical practice ($p = 2e-17$). CMA even outperforms the "oracle" decoders which are trained on labeled epochs from the test session (infeasible in practice).

Let $\mathbf{x} \in \mathcal{X}$ and $y \in \mathcal{Y}$ represent the input and output. We aim to learn an adaptive decoding model $f : \mathcal{X}^d \rightarrow \mathcal{Y}^d$ for a shifted domain d. We consider each recording session as a unique domain. CMA is composed of a decoding model f_θ parameterized by θ, and an adaptation model h_ϕ parameterized by ϕ. During inference, the decoding model takes in the LFP biomarkers and predicts the probability of gait movement. The adaptation model takes in the unlabeled contextual embeddings and adjusts the parameters of the decoding

model ($h_\phi(\cdot, \theta) \rightarrow \tilde{\theta}$).

5.2.7 Model parameter initialization

For simplicity, we first consider the case of few-shot adaptation, where model parameters are fine-tuned on a few epochs using supervised gradient updates:

$$\theta^d = \theta - \alpha_1 \nabla_\theta \frac{1}{T} \sum_{t=1}^{T} \ell(f_\theta(\mathbf{x}_t^d), y_t^d), \tag{5.4}$$

where α is the learning rate and T is the number of epochs used for fine-tuning. Here, $\mathbf{x}_t^d$ and y_t^d refer to LFP epochs and gait states sampled from the domain d: $\mathbf{x}_t^d, y_t^d \sim p(\mathbf{x}, y \mid d)$. The model parameters change from θ to θ^d as we adapt the decoding model f_θ on a new domain d. Ideally, θ should be widely suitable for a number of domains such that fine-tuning on a few epochs would be sufficient for adaptation to a new domain. On the other hand, the adapted parameters θ^d are domain-specific. After adaptation, the new model f_{θ^d} can enhance the decoding performance on new recording sessions.

Alternatively, we propose CMA as an unsupervised imitation of few-shot adaptation methods. We initialize the decoding model f_θ on multiple domains such that it can easily generalize to new domains and recording sessions. Specifically, we train f_θ using model-agnostic meta-learning (MAML) [162]. The learning process can be expressed as follows:

$$\theta \leftarrow \theta - \alpha_2 \nabla_\theta \frac{1}{D} \sum_{d=1}^{D} \frac{1}{T} \sum_{t=1}^{T} \ell(f_{\theta^d}(\mathbf{x}_t^d), y_t^d). \tag{5.5}$$

where α_2 indicates the update rate for meta-learning and θ^d is derived from

Eq. 5.4 via gradient optimization. Overall, MAML initializes decoder parameters that can be easily adapted to new domains.

5.2.8 Test-time adaptation

In this work, we replace the supervised fine-tuning steps in MAML with unsupervised test-time adaptation. In Eq. 5.4, the model parameter adaptation process ($\theta \rightarrow \theta^d$) involves supervised gradient updates. Such a process can not be performed at the test time by leveraging only unlabeled LFP epochs. In CMA, we introduce a separate adaptation model h_ϕ to infer θ^d from the initialized model parameters θ and the domain-specific contextual embedding C^d. Indeed, h_ϕ returns the *adapted* model parameter $\tilde{\theta}^d$ as an empirical estimation of θ^d. The unsupervised adaptation process holds the underlying assumption that contextual embedding C^d provides predictive knowledge on the test domain joint distribution $p(\mathbf{x}, y \mid d)$. We will discuss the choice of C^d and its update rule in the next section. In practice, the adaptation model is trained by minimizing the following objective:

$$\min_{\phi} \frac{1}{D} \sum_{d=1}^{D} \frac{1}{T} \sum_{t=1}^{T} \ell(f_{\tilde{\theta}^d}(\mathbf{x}_t^d), y_t^d), \tag{5.6}$$

where

$$\tilde{\theta}^d = h_\phi(C^d, \theta). \tag{5.7}$$

During inference, the adaptation model h_ϕ adjusts the parameters of the decoding model ($f_\theta \rightarrow f_{\tilde{\theta}^d}$), and $f_{\tilde{\theta}^d}$ predicts the movement probability from the current LFP epoch.

$$p_t^d = f_{h_\phi(C^d,\theta)}(\mathbf{x}_t^d) \tag{5.8}$$

5.2.9 Contextual embeddings

In CMA, we aim to infer the test domain knowledge from contextual embeddings. Given that the joint distribution $p(\mathbf{x}, y \mid d)$ explicitly contains the gait state y (which is not accessible at the test time), we instead use $p(\mathbf{x} \mid d)$ as a surrogate distribution to adapt the decoder parameters. During the inference phase of gait decoding, LFP epochs (i.e., $\mathbf{x}$) are collected in a streaming manner. Therefore, it is infeasible to infer $p(\mathbf{x} \mid d)$ from the entire test dataset. Rather than computing a static contextual embedding for each domain, we calculate C^d on a stream of epochs at the test time. We used a moving average filter to update $p(\mathbf{x} \mid d)$ on-the-fly. The contextual embedding contains a numerical representation of $p(\mathbf{x} \mid d)$, which includes the mean (M) and variance (Var) of preceding LFP epochs, and the difference between the mean value and current epoch (δ).

$$\delta_t = x_t - \mathrm{M}_{t-1}, \tag{5.9}$$

$$\mathrm{M}_t = \mathrm{M}_{t-1} + (1-\beta) \cdot \delta_t, \tag{5.10}$$

$$\mathrm{Var}_t = \beta \left(\mathrm{Var}_{t-1} + (1-\beta) \cdot \delta_t^2 \right). \tag{5.11}$$

where β is a hyperparameter that defines the updated momentum of the moving mean and variance. The contextual embedding at the t-th epoch C_t^d is simply the concatenation of δ_t, M_t, and Var_t.

5.2.10 Adaptive gait decoding

Given the learning process in Eq. 5.5 and 5.6, both the gait decoder f_θ and model adaptor h_ϕ are trained with gradient-based optimization. The gait decoder is a classification model which predicts the probability of gait states, whereas the model adaptor handles a regression task to adapt the decoder parameters. Both

f_θ and h_ϕ use neural networks as the backbone model and Adam as the optimizer (learning rate: 0.001).

In CMA, we train a separate model adaptor to compensate for the domain shift caused by PD therapies such as DBS and medications. As shown in Eq. 5.7, the model adaptor generates a set of decoder parameters that best suit the test condition. The output space of h_ϕ has the same dimensionality as the parameter space of f_θ. In the case of large decoding models with numerous trainable parameters, the model adaptor may suffer from high complexity. In our experiments, we limited the model adaptor to alter only the first (i.e., input) layer of f_θ. Therefore, the output space of h_ϕ has the same dimension as the LFP feature vector ($\mathbf{x}$), relaxing the complexity of the model adaptor. We note that adapting the first layer of f_θ is equivalent to adding an offset to the input feature vector. Let θ and θ^d denote the first-layer decoder parameters before and after adaptation. Rather than predicting θ^d directly, the model adaptor can learn an input offset $\Delta\mathbf{x}^d$ such that $\theta \cdot (\mathbf{x} + \Delta\mathbf{x}^d) = \theta^d \cdot \mathbf{x}$. This allows us to visualize and compare the feature distribution with and without test-time adaptation.

In Fig. 5.4, the adaptation model is separate from the gait decoder to compensate for domain shift under various therapeutic conditions. With this architecture, f_θ and h_ϕ are differentiable models capable of gradient-based optimization. An alternative would be to combine all components into one model and train it in an end-to-end fashion. The combined model is conditioned on the contextual embedding, and predicts the movement probability out of the current LFP epoch. In the framework of CMA, an end-to-end adaptive gait decoder is trained to fit $p(y \mid \mathbf{x}, d)$ whereas the conventional decoder with fixed parameters simply approximates $p(y \mid \mathbf{x})$. The training process is simple and similar to

that of a fixed decoder. Moreover, the end-to-end adaptive decoder enjoys great flexibility as it is model-agnostic, and in addition, it is compatible with non-differentiable models such as Gradient Boosted Trees and Random Forests. The end-to-end adaptive decoder achieved an improved performance compared to its fixed counterparts. However, the end-to-end training reduces the gait decoding performance by 1% compared to CMA with differentiable components.

5.2.11 Benchmark models

We benchmarked the proposed CMA against various decoder configurations including Baseline, Oracle, ERM, and ERM-oracle. We provide the implementation details as follows: **Baseline** refers to ML decoders trained on recording sessions with no therapeutic intervention. The current practice is largely based on the Baseline method where a single training session is recorded in a research laboratory with little concern on signal fluctuations over time [141]. **Oracle** further assumes that the test domain is accessible, allowing us to train the decoder on labeled data from the test domain. Specifically, the performance of Oracle was estimated using 5-fold cross-validation over unshuffled time series within each test session. We note that the Oracle approach is infeasible in practice, since the ground-truth gait states are not available during inference. By eliminating the domain shift, Oracle greatly improved the decoding performance over Baseline. However, Oracle is still sub-optimal as it does not leverage the data collected from other domains. **Empirical Risk Minimization (ERM)** is a strong baseline for meta-learning and test-time adaptation [163]. ERM takes account of various stimulation and medication settings in the training phase and strives to learn a domain-invariant representation that is robust to unexpected domain shifts. Compared to Baseline and Oracle, ERM achieved a higher per-

formance without violating the practical requirements, showing the benefits of including various training domains. However, as ERM still uses fixed model parameters, it may require an unnecessarily large training set to cover all possible changes in the neural signal. **ERM-oracle** is similar to ERM but it further assumes the accessibility of the test distribution. In addition to the training sessions used in ERM, we further leveraged a small portion of test data to train ERM-oracle so that the domain shift between training and test epochs was reduced. Similar to Oracle, implementing ERM-oracle is unrealistic in practice. **CMA** outperformed all benchmark models by effectively leveraging multiple training domains and compensating for domain shift during inference. Interestingly, CMA even surpassed the Oracle models, proving its advantage over decoder realignment methods such as [145], in addition to its simple, fast, and unsupervised adaptation process. For a fair comparison across benchmarks, we used a linear model for all the aforementioned methods.

5.2.12 Prediction uncertainty

We used Shannon entropy [156, 164] to measure the uncertainty of gait state predictions: $H = -\sum_t p_t \log p_t$. Shannon entropy is minimized ($H = 0$) when a decoder makes predictions with high certainty, either as "Move" ($p_t = 1$) or "Stand" ($p_t = 0$). On the contrary, Shannon entropy increases as decoders make uncertain predictions on an epoch (e.g., $p_t = 0.5$). Shannon entropy is a good indicator to quantify the uncertainty level, while increased entropy levels correlate with high errors in cross-domain classification [156, 158].

5.2.13 Feature importance

In Fig. 5.9(d), we measured the importance of each feature using a backward process. We first calculated the binary cross-entropy loss using the entire feature set (L_{base}). We then removed a feature (i.e., feature #i) and calculated the loss with the remaining feature subset ($L_{/i}$). The difference between L_{base} and $L_{/i}$ indicates the loss reduction by including feature i, and was used to quantify the importance of each individual feature.

5.2.14 Statistical analysis

We performed the data analysis using the statistical toolbox in MATLAB R2021b (MathWorks). We used one-way repeated measures ANOVA to make comparisons across various conditions, and the Mauchly's sphericity test for validation. Multiple comparisons with Bonferroni correction were used to report the significance level with a confidence interval of 95% ($p < 0.05$).

5.3 Results

5.3.1 Experimental setup and LFP recording with a chronic device

The objective of this study was to develop neural decoders that could predict motor states from LFP while automatically adapting to signal changes over time, particularly those that occur following the intake of medication or DBS therapies. Sixteen patients with Parkinson's disease were recruited for the study and performed gait experiments. N = 4 patients had to be removed due to the presence of heavily artifacted LFP channels (more details in Methods). Overall, 12 patients were retained for the analysis. During locomotor tasks, patients

were instructed to stand for about 3 seconds before initiating a sustained bout of walking on a straight line at their comfortable speed. When arriving at the end of the bout, patients were instructed to stop and stand for another 3 seconds, before doing a U-turn and starting again (Fig. 5.1(a)).

All participants received bilateral deep-brain stimulation leads (Medtronic 3389) and were recorded within five days following their surgery. Participants were implanted with a Percept PC stimulator (Medtronic, USA) in the right abdominal area, and recorded using the sensing capabilities provided by the device (sampling frequency of 250 Hz) [138]. We first recorded the patients in the off-medication condition, either with the DBS off, high-frequency DBS on (125 Hz), or low-frequency DBS on (55 Hz). We then recorded them in the on-medication condition (with high-frequency DBS on) at 15, 30, and 45 minutes after the medication intake. Not all patients were recorded under all conditions: N = 1 patient did not take any dopaminergic medication at the time of recording and could only be recorded under different DBS configurations.

In addition to walking, patients were instructed to perform repeated leg movements while sitting. Similar to DBS- and medication-induced variabilities, sitting versus standing or walking introduced cross-session instabilities and degraded the decoding performance. All data acquisitions during sitting were performed in the absence of medication and DBS.

5.3.2 Modulations in brain activity induced by PD therapies

Domain shift in neural signals can be caused by multiple reasons in practice. For example, recordings from intracortical microelectrode arrays have been shown to drift over time, making the recalibration sessions necessary for typi-

cal Brain–Computer Interface (BCI) tasks [151]. The impedance of an implanted electrode may vary as the scar tissue builds up at the electrode-tissue interface, altering the statistics of neural data and imposing a critical challenge on stable long-term decoding. Moreover, domain drift could be induced by the use of therapeutic interventions [165]. In this work, we primarily focused on the effects of two common therapies for Parkinson's disease: dopaminergic pharmacotherapy and deep brain stimulation.

To study the impact of dopaminergic drug treatment, we looked into patient LFPs before and after the intake of medication (Fig. 5.1(c-e)). Specifically, we compared the spectral power over Alpha (8-12 Hz), Beta (13-30 Hz), and Gamma (30-120 Hz) bands across four recording sessions: before medication, 15, 30, and 45 minutes after medication. Out of the 12 patients with Parkinson's disease, one patient did not take medication on the day of experiment. We analyzed the impact of medication on the remaining 11 patients. To reduce variability across patients, we normalized each band power to that of the baseline session (i.e., the session before the intake of medication). Following medication intake, we observed a significant reduction of Beta band activity (Fig. 5.1(c)) and a consistent increase in the spectral power over Gamma band (Fig. 5.1(d)). Furthermore, Fig. 5.2(e) compares the average time to complete the same gait task at different times after the intake of medication. At 45 minutes after medication, patients needed statistically less time to perform the instructed walking sequences (Fig. 5.1(e)). Similarly, we studied the LFP fluctuations following DBS delivery in all 12 patients. The normalized band power over the Alpha and Beta bands did not vary significantly at different stimulation settings across patients. Here, the Gamma power was heavily affected by stimulation artifact. We used an artifact removal technique to eliminate the effect of stimulation (details in

the Methods Section), at the cost of losing the original LFP over the DBS band. As a result, the Gamma power with DBS (at stimulation frequencies of 55 Hz and 125 Hz) was lower than that without DBS (Fig. 5.1(b)). The normalized task duration decreased with stimulation, but it did not reach the significance level.

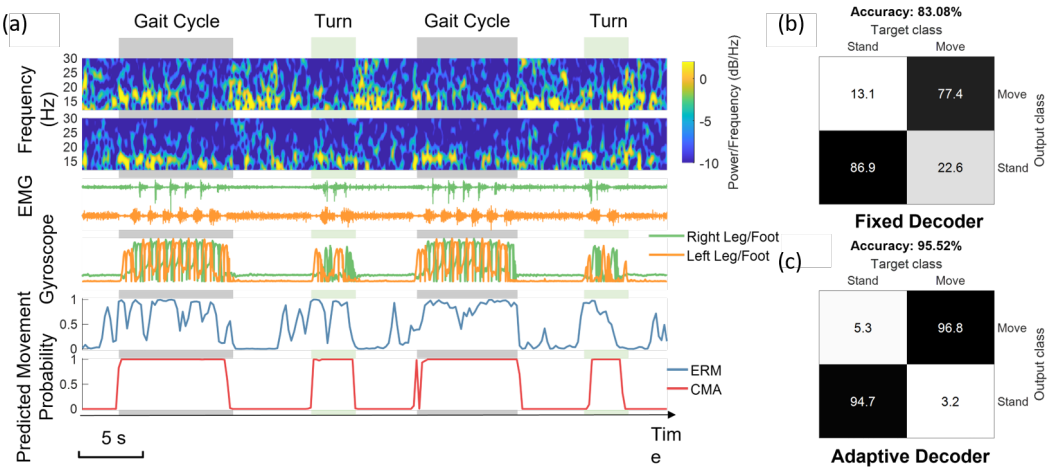

Figure 5.5: Gait state decoding in PD patients. (a) (Top) the LFP spectrogram from the right and left hemispheres in a patient with PD (PD1) under high-frequency DBS at 125 Hz and 15 minutes after medication intake; (Middle): EMG signals recorded from the hip joint (rectus femoris muscle), and Gyroscope signals recorded from both feet; The gyroscope signal is used to annotate the motion state. The"Move" state is indicated by the shaded area and includes the gait cycle and turning. (Bottom): The predicted movement probabilities by the fixed and adaptive decoders. (b, c) Confusion matrices of the fixed and adaptive decoders on the same session as (a). The adaptive decoder outperforms the fixed decoder by 12.4% on this movement classification task.

5.3.3 Therapy-induced domain shift and cross-session decoding performance

Traditional machine learning algorithms such as Decision Trees and Neural Networks assume that the training and test data are identically and independently distributed (i.i.d). However, the i.i.d assumption is often violated in practice where the training and test data are not sampled from the same static distribu-

tion. The discrepancy between training and test distributions can cause domain shift and lower the neural decoding performance in various tasks [145, 154, 166]. Here, we primarily focus on therapy-induced domain shift and aim to improve the decoding stability across consecutive sessions with different therapeutic settings.

Figure 5.2(a) visualizes the LFP spectrograms recorded from the left and right hemispheres in a patient with PD (PD4). Both DBS and medication modulated the patient's brain recordings, albeit the corresponding domain shifts showed different patterns. DBS induced instant and abrupt changes in LFP, whereas medication mediated a gradual signal drift over time. In Fig. 5.2(b), we plot the Beta band power as a function of time in this patient. Overall across 12 patients, we observed a gradual decline of Beta activity in LFP, 15 minutes after medication. The Beta activity subsequently saturated at a low level (10 μV^2), 30 minutes after medication. We trained two decoders to predict the gait state across sessions. The conventional decoder used constant model parameters upon training, whereas the adaptive decoder leveraged test-time adaptation to update parameters on-the-fly. Given the therapy-induced domain shift, we trained both decoders on the therapy-free baseline session (Stim Off, Med Off) and tested them on the subsequent sessions (Stim @55 Hz, @125 Hz, Med @15 mins, @30 mins, @45 mins). The implementation details of the fixed and adaptive decoders are presented in the following sections. Figure 5.2(c) shows the cumulative AUC scores for the conventional (i.e., fixed) and adaptive gait decoders. Compared to the conventional approach, the adaptive model is more stable in the presence of domain shifts and achieves a higher cross-session decoding performance (15.6% improvement in the AUC score).

We further quantified the discrepancy between LFPs from different recording sessions using the distance of their distributions. Epochs from each session were fitted into a multivariate gaussian distribution. We used the Kullback–Leibler (KL) divergence to measure the similarity between each pair of sessions (Fig. 5.3(a)). Two sessions were considered similar (i.e., small domain drift) if they were connected via a dark line, thus showing a low KL divergence. For example, epochs collected from the sitting sessions were close to those recorded from the therapy-free gait sessions since the leg movements were similarly performed in the absence of medication or stimulation. Furthermore, sessions with different medication settings were clustered closely, as they shared the same DBS frequency. However, as the difference in recording times increases, the corresponding distributions drift apart. For instance, compared to Med @15 mins or @30 mins, Med @45 mins showed a larger KL divergence from Med Off. We further demonstrated the exact distribution of LFP epochs using t-distributed stochastic neighbor embedding (t-SNE, Fig. 5.3(b)). We trained a fixed decoder on the therapy-free baseline session and calculated the AUC scores for all other sessions. In the case of within-session inference, the training and test data were sampled from the same session. We used 5-fold cross-validation to evaluate the within-session gait decoding performance. On the other hand, the cross-session performance was reported on the test sessions other than the baseline session. Therapy-induced domain shifts affected the cross-session inference, with the classification performance (AUC score) declining as the discrepancy between training and test distributions grows. Overall, we conclude that various therapeutic settings in PD may induce significant LFP variability, and that such therapy-induced domain drift can harm the decoding performance, urging the need for new algorithms.

5.3.4 Adaptive neural decoder and model structure

Adaptive neural decoders differ from conventional decoders in that such models are dynamically re-weighted during inference to account for signal variability over time. Therefore, adaptive decoders can learn session-specific model parameters and fit into neural data with shifted distributions. In this chapter, we propose Contextual Meta Adaptation (CMA) to compensate for therapy-induced domain shifts, by enabling an efficient test-time unsupervised adaptation. The proposed algorithm consists of a gait decoder to predict lower limb movements and an adaptation model to adjust decoder parameters (Fig. 5.4). We train our adaptive gait decoder to learn generalized representations shared across multiple training sessions using model-agnostic meta-learning (Meta-train, Fig. 5.4) [162], whereas the adaptation model aims to capture session-specific knowledge by observing the LFP statistics and adjusting the parameters of the gait decoder (Meta-test, Fig. 5.4). Specifically, we extracted contextual embeddings to represent covariate shift across sessions (details in Methods). We leveraged contextual embeddings during test time to infer session-specific knowledge since the calculation only involves an unlabeled stream of incoming epochs (details in Methods). In Fig. 5.4, contextual embedding is visualized by the mean and standard deviation for each element in the feature vector. Intuitively, contextual embedding carries the knowledge of feature distributions and measures the statistics of shifted features across sessions. The adaptation model takes contextual embedding into account to re-weight parameters associated with shifted features. We compared CMA with various non-adaptive ML models, including the current clinical practice using a fixed decoder trained on the therapy-free session (Baseline), the fixed decoder trained with labeled data from test sessions (Oracle), empirical risk minimization (ERM) which was reported as

a powerful baseline for multi-task learning [163], and ERM using information from test sessions (ERM-oracle). Implementation details of the baseline models are available in the Methods section. With the adaptation block, CMA achieved a significantly higher performance compared to various fixed decoders (Performance, Fig. 5.4).

5.3.5 Online gait decoding with the stream of epochs

We evaluated the performance of CMA in a streaming setting, where a stream of LFP epochs was processed by the gait decoder on-the-fly (Fig. 5.5). The neural decoder predicted gait states every 200 ms based on the most recent 1-second LFP epoch. Spectral power features and Hjorth parameters were extracted from LFP epochs to construct feature vectors (see Methods, similar to [2]), and were subsequently fed to the gait decoder. Estimation of LFP statistics at the beginning of test session is infeasible under the streaming setting as only a few epochs have been observed. To address this challenge, we calculated the contextual embedding using an exponential moving average filter (see Methods). The contextual embedding was updated by only using the current epoch, eliminating the need to store feature vectors from previous epochs. The continuous update of contextual embedding is particularly critical for domains that slowly drift over time. For example, we observed a gradual drift of Beta-band power 15 mins after medication (Fig. 5.2). Since domain statistics change even within a recording session, a static approach to contextual embedding fails to account for signal variations throughout a session.

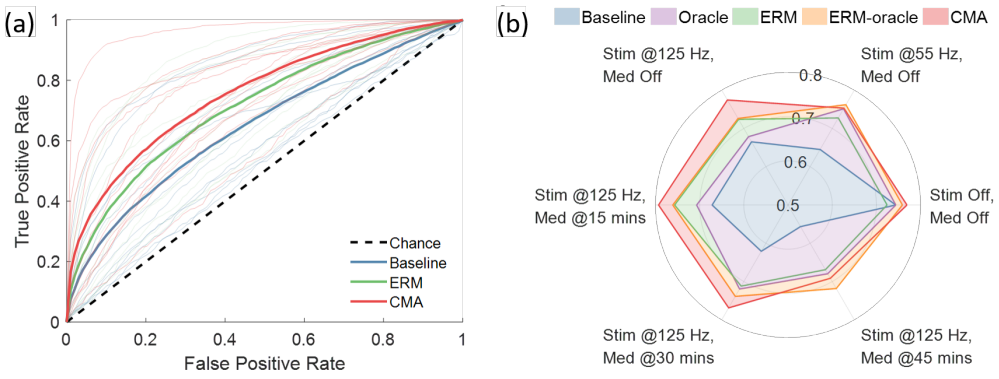

Figure 5.6: The gait decoding performance of CMA against various baseline decoders. (a) The receiver operating characteristic (ROC) curves corresponding to different approaches. The performance of each patient is shown by transparent lines, while the opaque lines indicate the average decoding performance across all patients. Baseline refers to the fixed decoder trained on the therapy-free session, whereas ERM was trained over multiple sessions excluding the test session. Overall, CMA achieved an average AUC score of 0.757 on 12 PD patients, outperforming the Baseline decoder and ERM with AUC scores of 0.644 and 0.720, respectively. (b) Comparison of gait decoding performance (AUC) under different therapeutic conditions. "Oracle" indicates that a subset of test session data is used for training. By leveraging the test session information, we observed an increase in the gait decoding performance for both Baseline and ERM. However, "Oracle" decoders are infeasible in real-world applications. The widely-used Baseline decoder worked well with the therapy-free session (Stim Off, Med Off), but failed to maintain a good performance as patients received PD therapies. The proposed adaptive decoder, CMA, could compensate for therapy-induced domain shift and maintain a good decoding performance across sessions. Overall, CMA outperformed the Baseline decoder over all 6 therapeutic settings and surpass the best "oracle" method (ERM-oracle) in 4 sessions.

5.3.6 Adaptive decoding on unseen PD therapeutic settings

We tested the performance of the proposed adaptive decoder, CMA, on the cross-session gait decoding task. We employed a leave-one-out approach to separate the data into training and test sessions. The test session underwent a unique treatment condition not seen in the training data, thus inducing a distribution shift to the acquired LFPs. Figure 5.5(a) visualizes the predictions from

the fixed and adaptive decoders. The spectrograms of LFP from the right and left hemispheres are aligned with the EMG signal that shows the activities of the rectus femoris muscle, and the gyroscope signal capturing feet movements. In the LFP spectrograms, we observed a reduction in Beta-band activity during gait movement [141, 167, 168]. We used gyroscope signals to annotate the gait state and the EMG for verification (see Methods). Overall, we observed that the adaptive decoder predicted the gait state with high certainty (i.e., movement probability equal to 0 or 1), and this was more aligned with the gait annotation. We quantitatively compared the gait decoding performance of the fixed and adaptive decoders using a confusion matrix (Fig. 5.5(b)). Here, "Move" is an aggregation of "Walk" and "Turn" states since both states showed similar behaviors in neural and EMG recordings. A threshold of 0.5 was set to discretize the movement probability into either "Stand" or "Move". The adaptive decoder improved the accuracy by 12.4% compared to the fixed decoder.

To verify the robustness of CMA against therapy-induced domain shift, we extensively compared the performance of adaptive and fixed decoders on all recording sessions. Figure 5.6(a) shows the receiver operating characteristic (ROC) curves for each individual patient (the transparent lines) as well as the mean performance (opaque lines) in cross-session gait decoding, where the adaptive decoder (CMA) achieved a superior performance. We further used the area under the ROC curve (AUC) as the evaluation metric and compared the decoding performance under each therapeutic condition (Fig. 5.6(b)). It can be seen that the adaptive decoder (CMA) outperformed its fixed counterparts (Baseline and ERM) on all recording sessions. Importantly, we observed that the cross-session decoding performance of CMA frequently outperformed the "Oracle" approaches (implementation details in Methods) which recalibrate

the decoder by using labeled data from the test session. The "Oracle" methods measure the ideal gait decoding performance without cross-session domain shift, and are infeasible for real-world applications where the test data cannot be fully accessible beforehand. This also violates the fundamental assumption of test-time adaptation.

Our proposed adaptive decoder outperformed the supervised recalibration method in 4 out of 6 PD therapeutic conditions (Fig. 5.6(b)). Thus, compared to supervised recalibration, CMA provides a more effective way for unifying the data from multiple recording sessions. ERM-oracle is trained on recordings from multiple sessions, similar to CMA. However, it lacks a mechanism to uncover the shared knowledge across sessions and unify the data recorded under different stimulation or medication settings. Such a mechanism was introduced to CMA via model-agnostic meta-learning (MAML) to initialize decoder parameters that can be efficiently adapted to new sessions [162]. Moreover, even though the "Oracle" approaches are not affected by the cross-session domain shift, the within-session domain drift may still exist [159], particularly in the absence of shuffling to ensure leakage-free training [12]. In CMA, the within-session drift is addressed by updating the contextual embedding on-the-fly. Therefore, the adaptive decoder can account for gradual changes in neural recordings. We further separated the effects of stimulation- and medication-induced domain shifts and demonstrated the decoding performance on each recording session. In Fig. 5.7(b-c), marker styles indicate the stimulation/medication settings with colors representing patients. We trained the fixed decoders (i.e., baseline) on a single stimulation-free (Stim Off, Med Off, Fig. 5.7(b)) or medication-free (Stim @125 Hz, Fig. 5.7(c)) session. We observed that most sessions lie above the diagonal lines, proving the robustness of

test-time adaptation against both stimulation- and medication-induced domain shifts.

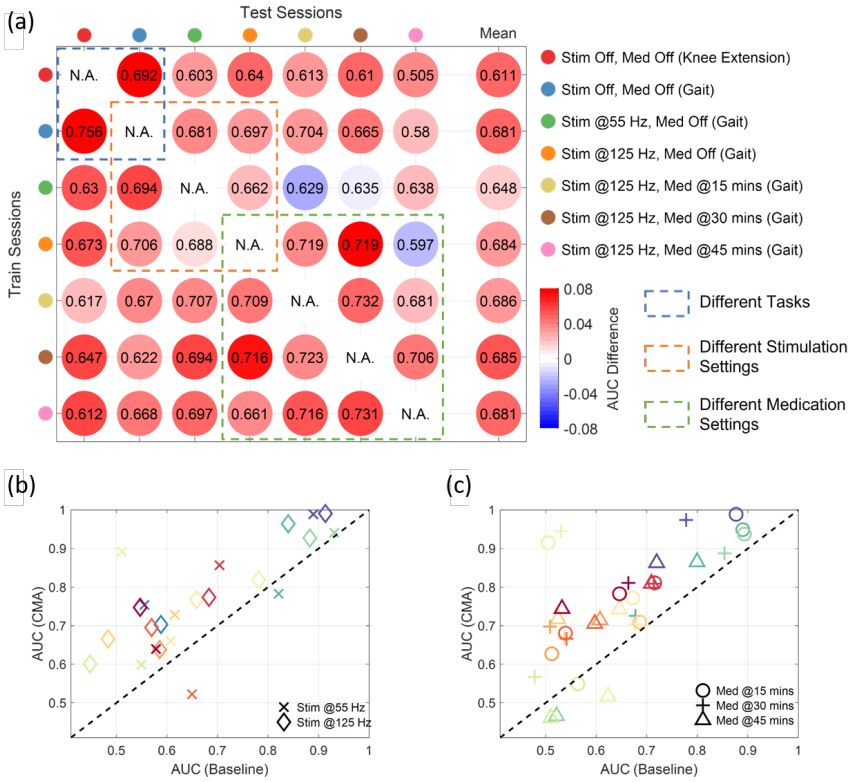

Figure 5.7: Comparison of gait decoding performance for the adaptive (CMA) and fixed (Baseline) decoders. (a) Movement classification performance on each train-test session pair. The numeric values in the circles indicate the classification performance (AUC scores) of adaptive (CMA) decoders. The colors indicate the difference between the performances of the adaptive and fixed decoders. Specifically, a red circle implies that the adaptive decoder outperforms the fixed decoder on the corresponding train-test session pair, whereas a blue circle indicates that the fixed decoder (Baseline) achieves a higher performance on that train-test pair. The dashed boxes cluster the tasks according to the type of domain shift. (b) Comparison of gait decoding performance under various medication settings (15 mins vs. 30 mins vs. 45 mins). (c) Comparison of gait decoding performance under various stimulation settings (55 Hz vs. 125 Hz). Decoders are trained on the "off" sessions for both medication and stimulation studies. Each marker represents an individual session and different PD patients are shown by various colors. Sessions above the dashed line (the majority of sessions in both cases) clearly benefit from using an adaptive decoder.

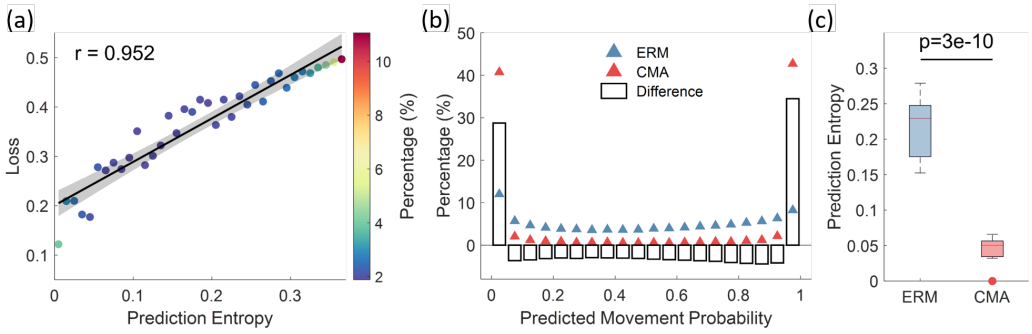

Figure 5.8: CMA reduces entropy in cross-session gait decoding. (a) Classification loss (i.e., misclassification rate) increases with prediction entropy (i.e., a measure of uncertainty). Entropy serves as a metric to quantify how well a decoder can adapt to new, shifted domains. (b) Adaptive decoder makes predictions with higher certainty. (c) Adaptive decoder leads to a significantly lower prediction entropy compared to fixed decoders.

5.3.7 Adaptive decoder with micro meta-learning

Meta-learning algorithms train on multiple domains to seek a general representation and allow fast adaptation to new domains. As the CMA parameters are initialized with meta-learning [162], its training phase requires multiple labeled recording sessions. However, recording and annotation of neural data from multiple conditions can be costly, posing a practical concern on our meta-learning-based approach. Here, we propose a micro meta-learning setting that can alleviate this issue and perform well even with a single training session.

LFP signals are multi-channel time series that vary from timestamp to timestamp in addition to cross-session domain shift. To address within-session variabilities, we considered each epoch as a separate micro-domain and applied the MAML to these micro-domains. We refer to this approach as micro meta-learning to distinguish it from the conventional meta-learning trained across multiple sessions. Therefore, CMA can be trained using the data collected under a single therapeutic condition and still generalize well to new stimulation

or medication settings. We analyzed the performance of CMA with micro meta-learning approach. Here, we present the average decoding performance (AUC score) across all patients for each pair of training and test sessions. The colors show the difference in performance of the proposed CMA and its fixed counterpart (Fig. 5.7(a)). A red circle indicates that CMA achieved a superior gait decoding performance on the corresponding train-test session pair, whereas fixed decoders outperform in blue circles. Testing on all combinations of training and test session pairs, 39 out of 42 (93%) benefited from the proposed test-time adaptation, with CMA achieving a significantly improved performance. Similar to Fig. 5.3, three types of cross-session variability exist in this case: task-induced, stimulation-induced, and medication-induced domain shift. This diversity ensures that the proposed approach is not specific to a particular type of domain shift and can be easily generalized to other tasks.

Even though CMA can work with a single training session using micro meta-learning, the performance is lower than the case where CMA is trained on multiple different domains (Fig. 5.6). This is due to the fact that within-session drift is much smaller than the cross-session domain shift (Fig. 5.3). Since CMA is trained on a single session only, it learns to generalize to within-session variability rather than the more significant shifts across tasks and therapies. Our results, however, showed that within-session decoding stability is also beneficial for cross-session gait decoding performance. CMA outperformed fixed decoders using a single training session (Fig. 5.7(a)), highlighting the advantage of CMA even in cases with limited training data.

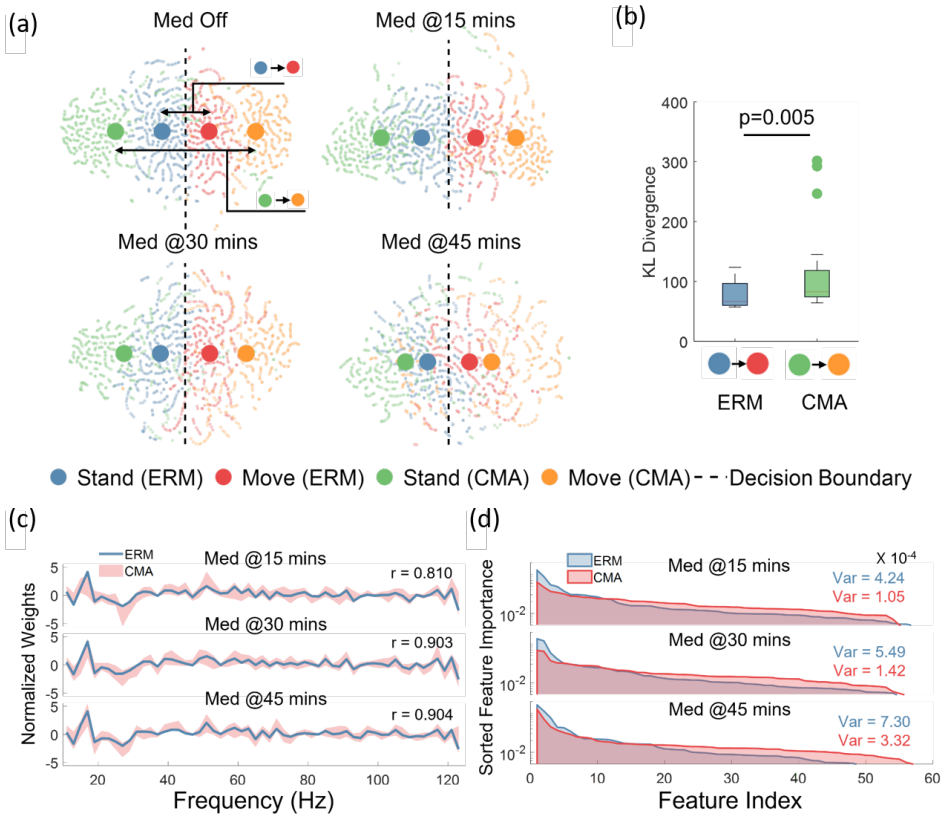

Figure 5.9: Visualization and interpretation of decoder performance (CMA and ERM) at different medication settings. (a) t-SNE visualization of epochs corresponding to stand and move states. The epochs of fixed and adaptive decoders are aligned to the decision boundary. (b) Comparison of decision margins of CMA and ERM across all patients. We used Kullback–Leibler divergence to measure the distribution difference between "Stand" and "Move" epochs. Larger KL divergence indicates that the "Stand" and "Move" states are more separable from each other, leading to a larger decision margin. The adaptive decoder achieves a larger decision margin. (c) Comparison of model weights for the fixed and adaptive decoders. We extracted spectral powers over 4 Hz frequency bands (x axis) and plotted the trained parameter values associated with each band. The parameters of ERM do not change with respect to input features, while adaptive decoders allow the weights to alter during test time. (d) Comparison of fixed and adaptive decoders in terms of feature importance distribution. We calculated the feature importance using backward feature selection and sorted the features in descending order. We show the Y-axis as a logarithmic scale to demonstrate the "long tail" effect of our proposed adaptive decoder. Compared to fixed decoders like ERM, CMA benefits from a broader set of features.

5.3.8 Interpretation of adaptive gait state decoding

By leveraging meta-learning and test-time adaptation, the proposed CMA approach is expected to make accurate predictions in an unseen, domain-shifted condition. To understand the behaviors of conventional decoders and CMA, we asked the question: why decoders with fixed parameters failed to generalize to a shifted distribution? To this end, we studied the cross-session predictions of ERM (i.e., the best-performing fixed decoder) on all PD patients. Intuitively, the predictions were more likely to be inaccurate if the gait decoder failed to recognize the test domain and as a result, decoded the gait state with low certainty. We used the entropy measure to quantify the level of uncertainty in a decoder's predictions. A strong correlation was observed between the prediction entropy and binary cross-entropy loss (r = 0.952, Fig. 5.8(a)). As shown in Fig. 5.8(b), CMA predicted the movement probability with high confidence (movement probability <0.05 for no movement or >0.95 for absolute movement), whereas ERM made more uncertain predictions (0.05 < movement probability < 0.95). Here, uncertain predictions with high entropy lead to poor decoding performance [156]. On the other hand, we observe that CMA significantly reduces the prediction entropy in cross-session gait decoding task (p = 3e-10, Fig. 5.8(c)). The reduced prediction entropy and improved decoding performance of CMA demonstrate its successful adaptation to shifted test environments.

Furthermore, CMA significantly increased the classification margin in our cross-session gait decoding experiment (Fig. 5.9(a)). The small classification margin can be considered as a cause of decision uncertainty since marginal epochs (i.e., those close to the decision boundary) are predicted with high uncertainty. However, given that uncertainty involves an unsupervised measurement method, low prediction entropy does not necessarily guarantee the superiority

of a decoder. Here, we used a more informative metric, the classification margin, which further involves class labels. Larger margins imply that the classes are more separable from each other, thereby enhancing the classification performance. As shown in Fig. 5.9(a), CMA was able to push away the epochs of different classes and maximize the distance between them. We quantitatively measured the classification margin as the KL divergence between the "Stand" and "Move" epochs. Testing on 12 patients with PD, CMA significantly increased the classification margin between different gait states ($p = 0.005$, Fig. 5.9(b)), leading to superior performance compared to the best-performing fixed decoder (ERM). With test-time adaptation, the model parameters are dependent on the contextual embedding and dynamically vary over time. Figure 5.9(c) plots the parameters associated with spectral power features when training a linear model. These adaptive parameters are highly correlated with the parameters of a conventional model without test-time adaptation ($r>0.8$), indicating that CMA only slightly adjusts the model parameters to compensate for cross-session domain shift. Another key distinction between CMA and fixed decoders lies in the distribution of feature importance scores. Machine learning models relying on only a few features are more prone to domain shift since individual features can become unstable across sessions. Alternatively, CMA made predictions by leveraging a more diverse portfolio of features rather than relying on a few critical features only (Fig. 5.9(d)). In our study, CMA led to a reduction of feature importance variance by 4.0x, 3.9x, and 2.2x for Med @15 mins, @30 mins, and @45 mins, respectively, compared to the fixed decoder. Consistent results were also observed at different stimulation settings. Our diverse feature portfolio is highly robust to the corruption of a few features and improves the decoder stability against therapy-induced domain shifts in PD.

5.4 Discussion

In this chapter, we introduced CMA, an unsupervised test-time domain adaptation approach to reliably decode gait in Parkinson's disease. Compared to conventional neural decoders with fixed model parameters, CMA can generalize to recording sessions with new stimulation or medication settings. The development of stable neural decoders is essential to the upcoming era of chronic closed-loop devices, where the embedded or external decoders can considerably suffer from changes of environment in patients' daily life settings. For the first time, we systematically verified the domain shift in the presence of common PD therapies and proposed to compensate for such variations by leveraging efficient adaptive decoders. CMA made minimal assumptions on the type of domain shift and the adaptation process to enable improved generalization over prior works [151, 152]. We showed the success of adaptation to stimulation on/off settings which causes abrupt changes in LFP, as well as the medication effect which alters LFP gradually over time. While we mainly demonstrated the performance on an LFP-based gait decoding task, the application of CMA can be extended to other tasks such as epileptic seizure detection.

Closed-loop stimulation paradigms such as adaptive DBS have been recently explored to improve the efficacy, reduce side effects, and enhance the energy efficiency of conventional open-loop stimulators [2, 31, 110, 111, 148, 157]. Accurate and stable decoding of movement state (e.g., gait cycle, tremor, voluntary movements) from chronic brain recordings is critical to enable more effective adaptive DBS systems. Due to the need for special equipment and electrodes to record neural and kinematic signals, research on motor decoding has been largely confined to laboratory settings. As adaptive DBS techniques are starting to move outside the clinic, several recent works have attempted to record

long-term brain recordings under more realistic conditions such as home environment [136, 137]. However, there exist limited efforts to develop models that can reliably decode chronic brain recordings.

Test-time domain adaptation addressed several key challenges in cross-session motor decoding including the lack of test-time neural data during training, the need for unsupervised adaptation, and lightweight parameter update. While previous works have approached cross-session neural decoding from one or more aspects, they failed to address all the aforementioned clinical concerns in practice. Recalibration, for instance, can reduce the discrepancy between training and test samples by frequently retraining the neural decoder. Despite being widely used in BCIs [145], recalibration is time-consuming and can cause major inconveniences for both patients and clinical staff. Distribution-alignment decoding (DAD) is an unsupervised motor decoding approach that is trained by matching the distributions of neural activity and movement in a low-dimensional space [169]. DAD does not need to measure both motor trajectory and neural activity simultaneously, but still requires motor statistics for calibration, making it less favorable compared to CMA.

The work in [151] introduced a stabilization algorithm for BCI applications. The decoder performed cross-session adaptation to reliably control a cursor by non-human primates. While the approach in [151] complied with unsupervised on-the-fly adaptation, it made heavy assumptions over the cause of domain shift (i.e., instability of electrodes). Artificial instabilities were intentionally generated by shifting the neural signal or disabling some electrodes. However, such BCI stabilizers can not generalize to complex therapy-induced domain shifts in PD. As a popular approach for domain adaptation, generative adversarial net-

work (GAN) was recently explored to decode hand trajectory from spike trains in monkeys [152]. Although GAN greatly accelerated cross-domain adaptation, it still required extensive training data during inference. GAN training is known to be computationally intensive and potentially unstable (it requires the iterative training of two networks), thus hindering parameter update in streaming settings [170]. Similarly, GAN augmentation requires labeled test samples for fine-tuning, which again violates the unsupervised adaptation mechanism [152]. In another recent effort [154], high-confidence predictions were used as the supervision signal to update decoder parameters on a seizure detection task. However, the pseudo-label-based approach holds the basic assumption that the underlying domain changes asymptotically such that the decoder does not make incorrect predictions with high confidence. While [154] demonstrated improved performance on seizure detection from EEG, its effectiveness has yet to be verified on neural activities undergoing abrupt changes (e.g., LFP with DBS on/off).

In general, cross-domain classification is closely related to several research topics of interest in machine learning. Transfer learning learns a decoder on a training domain with the goal of improving the model's performance on a different but related test domain. Transfer learning allows domain transfer across different tasks (e.g., lower-limb to upper-limb movement) but requires labeled test samples for fine-tuning [171]. Domain adaptation minimizes the gap between training and test sessions to boost cross-domain performance. Unlike transfer learning, domain adaptation is an unsupervised learning process. However, the test session epochs are still required to align the data with the training session [172, 173]. Test-time domain adaptation further relaxes this requirement and allows the model to update parameters using the stream of incoming test samples. Test-time domain adaptation is of particular interest

for implantable neural devices as it does not involve retraining or realignment on a large training set [156, 158]. Therefore, it achieves a good trade-off between cross-domain decoding performance and computational overhead. Meta-learning has been demonstrated to produce generalizable models across domains, but its success has been limited to few-shot learning settings [162]. As a popular variation of meta-learning, MAML learns from multiple relevant domains to rapidly adapt to future sessions. Our approach, CMA, is a combination of test-time adaptation and meta-learning, where model parameters are meta-learned offline using recordings under multiple therapeutic settings and adapted on-the-fly during test time. Therefore, our model enjoys both generalizable parameters initialized by meta-learning and a lightweight inference scheme through test-time adaptation. In addition, we proposed to update the decoder parameters using a forward pass of a stand-alone adaptation model, in contrast to backpropagation. Compared to pseudo-label-based adaptation which involves gradient descent [154], our approach circumvents the need to compute the loss derivatives and is more suitable for hardware implementations.

Despite the improved stability, generalizability, and computational efficiency of CMA, its success is still limited by a few factors. First, CMA was trained on multiple recording sessions with various therapeutic settings. Thus, the variability of training domains helped the meta-learning algorithm (i.e., MAML) to reliably capture test-time neural patterns and obtain a stable model. Recording multiple training domains, however, can increase the workload during training sessions. We relaxed this concern by introducing a novel approach, micro meta-learning, which considers each timestamp as a unique domain. While the micro setting improved performance over the baseline model, its per-

formance is still lower than the case of using multiple training domains. Second, our approach was developed under the assumption that the contextual distribution contains valuable information to correctly label epochs in the test domain (see Methods). While we empirically verified CMA on both abrupt and progressive domain shifts in neural signal, there is still a lack of theoretical guarantee on the convergence [174, 175].

Recent work suggested that the subthalamic nucleus encodes human gait via its connections to the brainstem locomotor pathway [141, 176]. Reduced Beta-band activity was reported in the STN LFP during gait movements [141, 167, 168]. Similar findings were observed in our experiments where patients demonstrated higher Beta power in the standing state compared to walking (Fig. 5.5). The separability of standing and walking states allowed us to build gait decoders based on STN LFP recordings (Fig. 5.9(a)). This could shed light on novel closed-loop DBS paradigms to treat lower-limb dysfunction in patients with PD.

In this chapter, we considered therapy-induced domain shifts that hinder the precise and stable decoding of gait state in PD. The proposed test-time adaptation could be of high interest for near-future clinical applications where an online neural decoder (potentially embedded on the implantable device) interacts with the DBS or other interventional therapies to improve gait deficits. We showed that conventional decoders with fixed parameters fail to generalize to various DBS settings, preventing the simultaneous operation of the stimulator and decoder. As closed-loop DBS aims at the precise control of stimulation delivery on a chronic basis, frequent switchings between DBS on/off or other personalized configurations are expected. In addition to stimulation, we demon-

strated that dopaminergic medication can significantly modulate neural activity in the STN. The proposed adaptive decoder is robust against PD medication effects and generalizes to different periods after the intake of dopamine agonists. The adaptive decoder, together with the chronic brain monitoring device, could provide a smart and robust solution for next-generation clinically-viable neural prostheses.

CHAPTER 6

CONCLUSION

This dissertation has explored machine learning in the realm of closed-loop neural prostheses. This concluding chapter aims to elaborate on the key insights derived, reflecting on the study's contributions, limitations, and potential directions for future research.

In conclusion, this dissertation addresses key challenges in the development of next-generation closed-loop neural prostheses. Initially, we explore a potential application of neural interfaces for migraine state classification using somatosensory evoked potentials. Recognizing the promising performance of neural interfaces in treating neurological disorders, we develop a hardware-friendly oblique tree model characterized by low power consumption and a small on-chip area. The success of tree-based models with neural data can also be extended to general tabular datasets. We introduce Tree-in-Tree decision graphs as a novel, efficient, and accurate alternative to widely-used decision trees. Lastly, we devise an adaptive decoder to compensate for fluctuations in neural signals. This adaptive neural decoder is applied to patients with Parkinson's disease in a real clinical setting, where they receive deep brain stimulation and medication, both of which alter neural activities.

6.0.1 Future directions

In future research, closed-loop neural prostheses can be further enhanced in terms of efficacy, efficiency and stability.

Firstly, it is crucial to integrate neural decoders with stimulators for controlling neural symptoms. By co-optimizing the on-chip biomarker extractor,

classifier, and stimulator, we can achieve even greater energy efficiency at a system level. Furthermore, the future may see more algorithmic ideas dedicated to optimizing stimulation parameters with reinforcement learning, an area that holds immense promise for improving treatment outcomes.

Secondly, adaptive neural decoders are anticipated to maintain stable performance in real-world scenarios over multiple years. While our adaptive decoder demonstrates stable decoding performance in the presence of therapy-induced variations, its long-term effectiveness in chronic recordings spanning years remains unvalidated due to data limitations. Investigating the drift of neural signals over extended periods and applying this technique to lifelong neural interfaces represent exciting directions for future studies.

Thirdly, the intersection of machine learning and neural prostheses presents a broad terrain of unexplored possibilities. For instance, Transformer models have notably excelled in domains such as computer vision and natural language processing. These models are exceptionally adept at processing sequential data, yet their application in the field of neural signal processing remains relatively uncharted. It is anticipated that forthcoming research endeavors will endeavor to narrow this gap, thereby further intertwining the disciplines of machine learning and neural prostheses.

In our research, we selected tree-based models in response to the dominant methodologies currently prevalent in the field. Traditional models like logistic regression and multi-layer neural networks have gained considerable traction in diverse applications. Nevertheless, when faced with neural data, they encounter certain limitations. Neural data often exhibit non-linear relationships between neurological biomarkers and disease symptoms, making linear models

less than optimal for accurate neural decoding. Conversely, while complex neural networks may capture the non-linearity in neural signals, the cost associated with on-chip implementation of these intricate models for efficient processing of numerous neural recording channels can be prohibitive.

Tree-based models, in contrast, have been recognized for their superior capacity to decipher complex neural signal patterns, thanks to their inherent flexibility and interpretability. This edge is particularly advantageous in medical settings where understanding the logic behind model decisions is paramount. The decision to use tree-based models in neural signal classification was also informed by their hierarchical structure. This structure enables dynamic inference, which is beneficial for power efficiency in neural interfaces with a multitude of recording channels.

However, as we have endeavored to underscore the merits of tree-based models in this thesis, it is equally crucial to note their limitations. Incremental training of tree-based models, for instance, remains a challenge. Exploring the deployment of these models in an online learning setting, particularly within a hardware-friendly framework, presents an intriguing avenue for future study. Further, tree-based models may occasionally fall short when dealing with exceptionally complex tasks such as managing sequential or temporal dependencies.

Among our key contributions is the development of an adaptive decoder aimed at compensating for the variability in neural signals, a vital step in achieving consistent performance in closed-loop neural prostheses. We have applied this adaptive neural decoder in a real-world clinical setting for patients with Parkinson's disease, undergoing deep brain stimulation and medication,

thereby demonstrating its effectiveness. Looking forward, the realm of unsupervised domain adaptation, which is the cornerstone of adaptive neural decoders, is rapidly evolving. We anticipate seeing a surge in the implementation of adaptive decoders in neural interfaces. However, the efficient integration of adaptive approaches on hardware remains a compelling challenge to be tackled collaboratively by both machine learning researchers and circuit designers.

One of the most significant barriers to progress in this field is the scarcity of data. Machine learning techniques, particularly those that fall within the domain of neural prostheses, require vast, high-quality datasets for both training and validation of models. The data shortage impedes model performance and limits our ability to understand the full scope of their potential capabilities. Addressing the challenges associated with collecting, processing, and interpreting large volumes of complex neural data will be a critical task. Future efforts will need to focus on developing strategies for data augmentation, such as synthetic data generation and transfer learning, to enhance the richness of data available for model training.

APPENDIX A

UNSUPERVISED DOMAIN ADAPTATION FOR CROSS-SUBJECT, FEW-SHOT NEUROLOGICAL SYMPTOM DETECTION

A.1 Introduction

Machine learning (ML) has been an increasingly useful tool in neural engineering in recent years. ML can be used to analyze and classify invasive or noninvasive electrophysiological recordings, enabling timely and accurate prediction of neurological symptoms (or events) in epilepsy [12, 113, 177], Parkinson's disease [2], migraine [77], and other emerging applications. However, despite the recent progress and potential of ML in neurological disease detection, the existing algorithms primarily use a subject-specific scheme, requiring each patient's extended neuronal recordings to train the model. Therefore, employing such algorithms on new patients with limited labeled data has been a challenge. This is particularly the case for invasive recordings, where the duration of recording is typically short due to surgical and ethical concerns (several minutes to days).

To tackle this problem, transfer learning aims to transfer the source domain knowledge to a target domain where labeled data is difficult to acquire [171]. Over the past decade, there has been an extensive literature on domain transfer learning [172, 178], with the goal of eliminating domain shift for better generative or discriminative performance. Among transfer learning approaches, the adversarial domain adaptation introduces an adversarial loss to minimize domain shift and enforce the learned representations to share a common feature space [179], and has obtained a promising performance in image-to-image translation tasks [180, 181]. Although domain adaptation techniques are widely used in computer vision tasks [180, 181], their application in neural engineer-

ing and particularly in detecting neurological symptoms is still underexplored [182].

In this work, we propose a cross-subject seizure detection algorithm based on adversarial networks [179]. We mapped the features from various subjects into a subject-invariant space via the proposed unsupervised adversarial domain adaptation. Following domain adaptation, we trained an ensemble of gradient boosted trees in the subject-invariant feature space to generate cross-subject seizure predictions. The rest of this work is organized as follows. We describe the classification task and dataset in Section A.2. The adversarial domain adaptation is introduced in Section A.3, followed by results in Section A.4. Section A.5 concludes the work.

A.2 Classification Task and Data Description

In this work, we propose a domain adaptation model for cross-subject seizure detection. This approach was evaluated on continuous iEEG recordings from 9 patients with epilepsy.

A.2.1 Seizure detection task and iEEG data

Epileptic seizure detection is a supervised classification problem to differentiate between seizure and non-seizure states of a patient. We studied a total number of 97 seizure events from 9 patients. The iEEG recordings were sampled at 500Hz and annotated as *seizure* or *non-seizure* by domain experts (publicly available at the IEEG Portal [92]). All included subjects gave written informed consent and the study was approved by the Mayo Clinic and University of Pennsylvania Institutional Review Board. We segmented the iEEG recordings of each

patient to 1s windows for the subsequent processing.

A.2.2 Feature extraction

A set of predictive biomarkers of seizure activity [12] were extracted from the segmented iEEG recordings, followed by domain adaptation and classification. The features and their definitions are as follows: line-length (LLN, $\frac{1}{d}\sum_d |x[n] - x[n-1]|$, d = window size), total power (Pow, $\frac{1}{d}\sum_d x[n]^2$), variance (Var, $\frac{1}{d}\sum_d (x[n]-\mu)^2$, $\mu = \frac{1}{d}\sum_d x[n]$), and band power over delta (δ: 1–4 Hz), theta (θ: 4–8 Hz), alpha (α: 8–13 Hz), beta (β: 13–30 Hz), low-gamma (γ_1: 30–50 Hz), gamma (γ_2: 50–80 Hz), high-gamma (γ_3: 80–150 Hz), and ripple (R: 150–250 Hz) bands.

A.2.3 Train-test split

We split the data into train and test sets using a block-wise approach, in which each block is comprised of one seizure event and the subsequent non-seizure segment. To evaluate the performance, we used the first n blocks for training and the remaining blocks for testing, referred to as 'n-shot learning' in the following sections. The block-wise approach is a fair method to evaluate the performance, as we use a number of recorded seizure events to predict a future unseen seizure [12].

A.3 Adversarial Domain Adaptation

In this section, we consider each subject (i) to be associated with a specific domain ($\mathcal{D}_i = \{\mathcal{X}_i, P_i(\mathbf{X})\}$), from which features are sampled. $P_i(\mathbf{X})$ denotes the distribution of the feature vector $\mathbf{X}$. Our goal is to learn a unique encoder for

each subject and map the features from this subject-specific domain to a subject-invariant domain.

A.3.1 Model structure

We first consider a simple case with only two patients: one patient from the source domain ($\mathcal{D}_S = \{\mathcal{X}_S, P_S(\mathbf{X})\}$) where exists abundant labeled data for a given task, and the other patient from the target domain ($\mathcal{D}_T = \{\mathcal{X}_T, P_T(\mathbf{X})\}$) where data is expensive to acquire for the same task.

As shown in Fig. A.1, the proposed adversarial domain adaptation model consists of three parts: encoder (source encoder E_S, target encoder E_T), decoder (source decoder D_S, target decoder D_T), and subject discriminator (SD). We used the handcrafted features ($\mathbf{X}$) as input to the encoders. The encoders and decoders form an autoencoder, which learns a latent representation (dimension: 2048) of the original input. The subject discriminator is a multilayer perceptron, which takes the latent representations ($E_S(\mathbf{X})$ denoted by green squares, and $E_T(\mathbf{X})$ denoted by red squares) as input. The subject discriminator has two hidden layers with 512 and 128 nodes, respectively. We trained the SD to predict whether the latent representations are from the source or target subject.

Adversarial loss The encoders and subject discriminator form a GAN model [179] for adversarial training. Here, we have encoders for both source and target domains. Let $\mathcal{L}_{\text{adv}}(\mathcal{X}_S, \mathcal{X}_T, E_S, E_T, SD)$ denote the standard supervised loss of SD. We train the subject discriminator by minimizing the loss:

$$\min_{SD} \mathcal{L}_{\text{adv}}\left(\mathcal{X}_S, \mathcal{X}_T, E_S, E_T, SD\right). \tag{A.1}$$

The goal of the encoders is to minimize the distance between the empirical source and target latent representations $E_S(\mathbf{X}_S)$ and $E_T(\mathbf{X}_T)$. Thus, we trained the encoders to fool the subject discriminator and make the source/target representations indistinguishable from each other:

$$\max_{E_S, E_T} \mathcal{L}_{\text{adv}}\left(\mathcal{X}_S, \mathcal{X}_T, E_S, E_T, SD\right). \tag{A.2}$$

Overall, the adversarial learning can be formalized as a maximin problem which can be solved using alternating optimization:

$$\max_{E_S, E_T} \min_{SD} \mathcal{L}_{\text{adv}}\left(\mathcal{X}_S, \mathcal{X}_T, E_S, E_T, SD\right).$$

Reconstruction loss and mode collapse With the adversarial training, we expect the latent space to be a subject-invariant representation of the inputs. However, the source and target encoders may simply learn to produce the same output (e.g., all zeros for latent representation), making it impossible for the subject discriminator to distinguish. In this scenario, the subject-invariant space cannot represent the inputs. This failure is referred to as mode collapse, which is a common issue with GAN training [170].

To avoid mode collapse, we reconstructed the inputs from the latent representations using a decoding stage. The decoders enforce the latent representations to preserve similar information as the inputs. We calculated the reconstruction loss $\mathcal{L}_{\text{rec}}$ using the mean squared error (MSE) and L_1 norm was used to regularize the autoencoder.

$$\mathcal{L}_{\text{rec}}(E, D) = \frac{1}{N}\sum_{n=1}^{N} \|X^{(n)} - D(E(X^{(n)}))\|^2 + \lambda(\|E\|_1 + \|D\|_1), \tag{A.3}$$

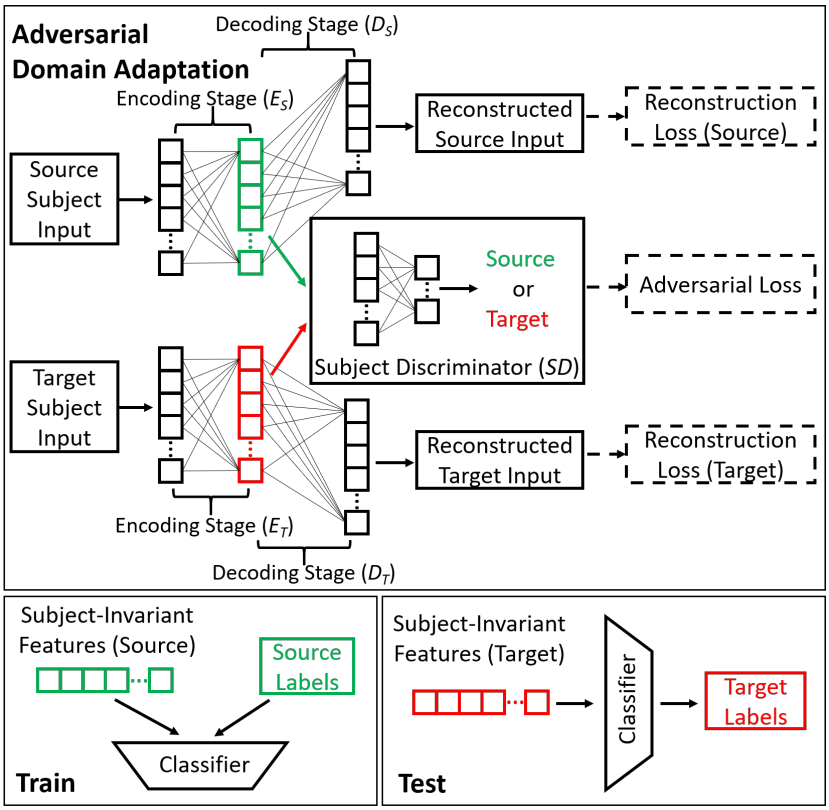

Figure A.1: Model structure of the proposed unsupervised adversarial domain adaptation (top), train and test approaches (bottom). The encoding and decoding stages form an autoencoder for each subject, which learns a latent representation of the input feature vectors. The subject discriminator takes the latent representation as input and is trained to distinguish the data from different subjects. The encoding stages are trained to fool the subject discriminator. Our goal is to learn encoders that map the input features to a subject-invariant space (denoted by green and red blocks). Following domain adaptation, a classifier is trained with the subject-invariant features to predict seizures on a target subject.

where N represents the mini-batch size, X_n denotes the n-th sample, and λ is the regularization coefficient, which was empirically set to 3e-5 in this work. In addition to maximizing $\mathcal{L}_{\text{adv}}$, we also train the encoders and decoders to minimize the reconstruction loss. Overall, the encoders and decoders are trained with the following formula:

$$\min_{E_S, E_T, D_S, DT} -\mathcal{L}_{\text{adv}}(\mathcal{X}_S, \mathcal{X}_T, E_S, E_T, SD) + \alpha(\mathcal{L}_{\text{rec}}(E_S, D_S) + \mathcal{L}_{\text{rec}}(E_T, D_T)), \tag{A.4}$$

where α controls the trade-off between the adversarial loss and reconstruction loss.

A.3.2 Multi-subject domain adaptation

Previous literature on domain adaptation has only focused on transferring from one source domain to the target [172]. However, for cross-subject seizure detection, we need to consider each patient as a unique domain and transfer the feature vectors from multiple subjects to a subject-invariant domain. Here, we extended the domain adaptation framework to enable multi-subject seizure detection. The subject discriminator predicts the patient index (rather than only a single source or target), and patients are alternately considered as target while others are considered as source. We used the cross-entropy loss for $\mathcal{L}_{\text{adv}}$:

$$\mathcal{L}_{\text{adv}} = -\sum_{i=1}^{N_s} \mathbb{E}_{\mathbf{X} \sim P_i(\mathbf{X})}[log(SD^i(E_i(\mathbf{X})))], \tag{A.5}$$

where $N_s = 9$ is the total number of patients, SD outputs a vector of size N_s, and the i-th entry of the subject discriminator output (SD^i) indicates the probability that $\mathbf{X}$ belongs to subject i.

Algorithm 4: Multi-Subject Domain Adaptation.

$E_{1,\ldots,N_s}, D_{1,\ldots,N_s}, SD \leftarrow$ random initialization ;

for *number of iterations* **do**

  Sample mini-batches of N samples from all subjects $\left\{\mathbf{X}_1^{(1)}, \ldots, \mathbf{X}_1^{(N)}, \ldots, \mathbf{X}_{N_s}^{(N)}\right\}$;

  Update the subject discriminator SD by gradient decent:

$$\nabla_{SD} \frac{1}{N} \sum_{i=1}^{N_s} \sum_{n=1}^{N} -log(SD^i(E_i(\mathbf{X_i^{(n)}})))$$

  for $i \in 1, \ldots, N_s$ **do**

    Update encoder and decoder E_i, D_i by gradient decent:

$$\nabla_{E_i, D_i} \frac{1}{N} \sum_{n=1}^{N} [log(SD^i(E_i(\mathbf{X_i^{(n)}}))) + \alpha(\|\mathbf{X}_i^{(n)} - D_i(E_i(\mathbf{X}_i^{(n)}))\|^2 + \lambda(\|E\|_i + \|D\|_i))]$$

Learning procedure Eq. A.1-A.4 show the learning objectives for the domain adaptation with two subjects: a source patient and a target patient. Here, we introduce the learning procedure for multiple patients. We alternatingly considered one patient as the target and all other patients as source. The algorithmic pseudocode is shown in Algorithm. 4. Our goal is to leverage the labeled data from source patients to make predictions for a target patient. The domain adaptation process is essentially unsupervised, mapping different subjects' data into a common feature space. A discriminative model was trained on the subject-invariant features to generate predictions for the target patients. We tested several settings. For example, 0-shot learning does not require any labeled recordings from the target patient. So we trained the discriminative model only on the labeled data from source patients. For n-shot learning, n labeled seizure blocks from the target patient were used for training, in addition to the extensive data from source patients.

Convergence Analysis The encoders map the input subject feature distribution ($P_i(\mathbf{X})$) to a latent space distribution ($Q_i(\mathbf{z})$). In the cross-subject learning scheme, we would like $Q_i(\mathbf{z})$ to be invariant across patients (i.e., $Q_i(\mathbf{z}) = Q_1(\mathbf{z})$ for all $i \in 1, \ldots, N_s$). Previous work has proven that adversarial training can reduce the shift between target and source domains [179]. In this work, the subject discriminator performs a multi-class classification task and patients are alternately considered as the target (Algorithm. 4). Following the framework in [179] , we recognize that Algorithm. 4 minimizes the Jensen-Shannon Divergence ($JSD(Q_1, \ldots, Q_{N_s})$) of latent space distributions [183]. Given that $JSD(Q_1, \ldots, Q_{N_s})$ is always non-negative and becomes zero if and only if all distributions are the same (i.e., $Q_i(\mathbf{z}) = Q_1(\mathbf{z})$ for $i \in 1, \ldots, N_s$), Algorithm. 4 will converge to a subject-invariant space given sufficient capacity.

A.4 Results

We tested the proposed algorithm for seizure detection from iEEG recordings of 9 epilepsy patients. We first mapped the input feature vectors into a subject-invariant space, using the proposed unsupervised adversarial training. A discriminative model (gradient boosted trees [109]) was trained in the subject-invariant space to make predictions for each patient.

A.4.1 t-SNE visualization of data distribution

We used t-SNE [184] to visualize the high-dimensional data distribution by mapping each data sample to a location in a 2-dimensional space. As shown in Fig. A.2, we plot the data distribution of two patients before and after domain adaptation. We used different colors and markers to show which class

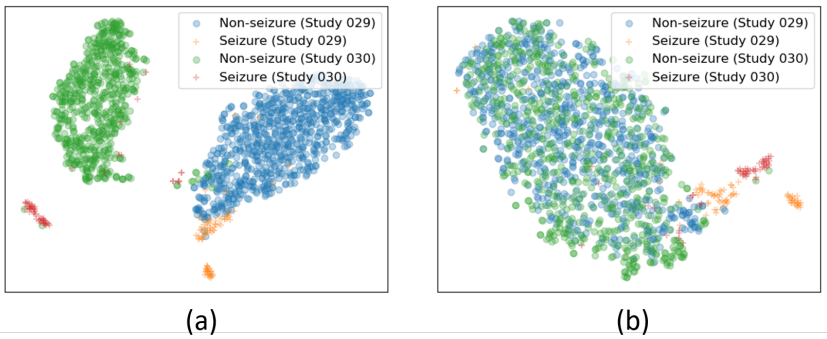

Figure A.2: t-SNE visualization of the data distribution from two patients; (a) Visualization of the data distribution before domain adaptation. (b) Visualization of the subject-invariant feature space. After domain adaptation, the data from different patients become indistinguishable.

the points are belonging to (seizure or non-seizure). In Fig. A.2(a), Study 029 has a different distribution from Study 030. The domain adaptation process successfully removed the between-subject variation and brought their distributions closer to each other (Fig. A.2(b)), enables cross-subject classification. Visualization was obtained with $\alpha = 0.01$ (see α in Eq. A.4 or Algorithm. 4), which we kept for the following experiments.

A.4.2 Cross-subject seizure detection

We first trained the encoders to map the features into a subject-invariant space, using the domain adaptation process depicted in Algorithm. 4. The size of mini-batches (N) was set to 32. We used the Adam optimizer [100] (learning rate of 1e-5) to update both encoders and subject discriminator for 100 epochs. Next, 100 gradient boosted trees with a maximum depth of 4 were trained on the subject-invariant features to predict the probability of epileptic seizures [109], using the 0-shot and n-shot learning schemes described above. In the n-shot learning scheme, we assigned different weights to the data from source and target patients. The samples from source patients received a weight of 0.01 while the

Table A.1: Performance of conventional subject-specific (SS) and cross-subject (CS) seizure detection methods.

Subject #	0-shot	1-shot		2-shot		3-shot	
	CS	SS	CS	SS	CS	SS	CS
Study 004-2	0.791 ± 0.091	0.889± 0.047	**0.935± 0.027**	**0.947± 0.033**	0.915± 0.041	N/A	N/A
Study 022	0.809 ± 0.058	0.787± 0.056	**0.962± 0.012**	0.910± 0.052	**0.960± 0.007**	0.875± 0.050	**0.915± 0.023**
Study 024	0.695 ± 0.124	0.874± 0.030	**0.949± 0.012**	0.837± 0.061	**0.944± 0.011**	0.914± 0.015	**0.938± 0.012**
Study 026	0.715± 0.099	0.658± 0.158	**0.931± 0.009**	0.931± 0.011	**0.953± 0.009**	0.928± 0.024	**0.957± 0.008**
Study 029	0.773± 0.034	**0.813± 0.059**	0.785± 0.070	0.942± 0.024	**0.944± 0.022**	N/A	N/A
Study 030	0.793± 0.044	**0.977± 0.005**	0.974± 0.010	**0.983± 0.009**	0.979± 0.003	0.976± 0.025	**0.990± 0.003**
Study 033	0.659± 0.049	0.888± 0.009	**0.901± 0.002**	0.885± 0.005	**0.887± 0.003**	0.920± 0.004	**0.923± 0.005**
Study 037	0.514± 0.145	0.631± 0.043	**0.801± 0.012**	0.597± 0.071	**0.751± 0.037**	**0.994± 0.004**	0.989± 0.004
Study 038	0.596± 0.029	**0.882± 0.019**	0.858± 0.027	**0.896± 0.005**	0.882± 0.027	0.915± 0.011	**0.929± 0.011**
Average	0.705± 0.030	0.822± 0.031	**0.899± 0.012**	0.881± 0.012	**0.913± 0.007**	0.932± 0.008	**0.949± 0.005**

data from target patients had a sample weight of 1. We applied the reweighting scheme to address the following concerns: (1) In few-shot learning, the training samples from source patients were more than the samples from target patients by an order of magnitude. (2) Compared to the data from source patients, the target patient data is more informative in predicting seizures on that patient.

Given the imbalanced nature of the seizure detection task, we evaluated the classification performance using the area under the ROC curve (AUC scores). Table. A.1 compares the classification performance with/without cross-subject knowledge. In the conventional subject-specific (SS) setting, we trained the classifiers by only using the data from a target patient (e.g., n seizure blocks in n-shot). For the cross-subject (CS) setting, we further incorporated the knowledge from source patients. We ran the proposed domain adaptation approach for 5

independent trials and reported the average performance (AUC scores) ± standard deviation. 3-shot learning on two patients (Study 004-2 and Study 029) is not applicable (N/A), since only 3 seizure events are available in both patients. As shown in this Table, 0-shot learning achieved an average AUC score of 0.705, which is much better than the chance level (0.5). For 1-, 2-, 3-shot learning, CS outperforms the SS in terms of average classification performance. However, as we used more labeled samples from the target patient (i.e., moved from 1-shot to 3-shot learning), the difference become less significant. Overall, cross-subject learning achieved a superior performance compared to the subject-specific setting, which indicates the importance of leveraging cross-subject knowledge. In addition to seizure detection, the proposed approach has the potential to help various neurological disorders and symptom detection tasks where training data is generally limited, which remains as future work.

A.5 Conclusion

In this work, we proposed a novel cross-subject seizure detection framework based on adversarial domain adaptation, by mapping the features from different subjects into a subject-invariant space and applying cross-subject learning. With unsupervised domain adaptation, we achieved a better performance compared to the conventional subject-specific approach, particularly when the training data is limited (few-shot learning). The proposed model efficiently incorporates the knowledge from previous patients to enable high-accuracy seizure detection in new patients.

APPENDIX B

XTAB: CROSS-TABLE PRETRAINING FOR TABULAR TRANSFORMERS

B.1 Introduction

With the increasing number of datasets represented as tables with rows and columns, tabular machine learning makes the foundation of many real-world applications. While deep learning has achieved tremendous success in the fields of computer vision (CV) [185, 186] and natural language processing (NLP) [187, 188], tabular deep learning models are not used as commonly as tree-based models [189, 190]. The primary challenge of tabular deep learning is the diversity of tabular tasks. Unlike text, which can be standardized as a sequence of tokens, tables are highly data-specific. Tabular data can vary in the number and types of columns. This makes it difficult for tabular deep learning models to transfer the knowledge learned from one table to another, leading to poor generalization abilities. Therefore, self-supervised learning for tabular data [185, 187], particularly one that is able to bootstrap the learning on new tables, is still an open problem.

There is an ongoing effort in migrating self-supervised pretraining techniques from CV [191] and NLP [187] to tabular tasks. With self-supervised pretraining, tabular deep models have demonstrated improved performance [192–194]. However, existing methods generally pretrain the tabular model on data from the same domain as the downstream task. As a result, the data-specific models cannot generalize to new tables.

Another direction of deep tabular learning aims to leverage Transformers, which drives the recent progress in NLP [188] and CV [195] for tabular tasks. Inspired by the success of the attention mechanism, Transformers were adapted

to tabular data [196–199] and demonstrated strong performance [189]. The core idea of tabular transformers is to consider the table columns as tokens, similar to words in a sentence. Therefore, tabular transformers can process tables with variable numbers of columns, thus making transferable learning [199] feasible.

In this work, we present *XTab*, a general framework for *cross-table pretraining of tabular transformers*. To resolve the issue that tables may vary in the number and types of columns, XTab decomposed the tabular transformers to two components: data-specific featurization and projection layers that capture the characteristics of each table, and a cross-table-shared block that stores the common knowledge. On a diverse collection of data tables, XTab trains these data-specific blocks and the shared block jointly via federated learning [200]. Once pretrained, XTab can bootstrap the learning process on a new table by initializing the shared block with pretrained weights. To verify our design, we conducted extensive experiments on AutoML Benchmark (AMLB) [190]. Our results show that transformers pretrained and initialized with XTab consistently outperform transformers with random initialization. By pretraining FT-Transformer [196] with XTab, we outperform the state-of-the-art tabular deep learning models.

The contributions of the work are summarized as follows:

- XTab offers a framework to account for cross-table variations and enable cross-table knowledge transfer.
- Given the large diversity of tabular datasets, we propose to pretrain on tabular datasets with federated learning. This allows us to perform distributed pretraining across a large collection of tables.
- To the best of our knowledge, we are the first to show that cross-table

pretraining can boost the learning speed and performance on new tables. This is different from table understanding tasks [201], the focus of which is to extract the semantical information from tables.

B.2 Related work

Tabular self-supervised learning. Inspired by the success of pretraining in CV and NLP, previous papers studied tabular self-supervised learning [192–194, 197, 199, 202, 203]. Among those works, [192, 202] proposed an auto-encoder framework with a pretext task to reconstruct the missing part of a table. [193] used contrastive learning as the pretraining objective and extended the SimCLR framework [191] to tabular tasks. [199, 203] further incorporated the label columns of tabular tasks in pretraining and proposed "target-aware" objectives leading to higher performance. As existing approaches only pretrain on one [192, 193] or a few relevant tables [199], the pretrained tabular model lacks generalizability. XTab alleviates this issue by pretraining on a large number of tables.

Tabular transformers. Transformer models are gaining popularity in the realm of deep learning for tabular data. For example, FT-Transformer has demonstrated superior performance on tabular classification/regression tasks [196]. Saint introduces the row-wise attention and captures the inter-sample interactions using transformer [197]. Fastformer proposes to use additive attention on tabular tasks, which is a lightweight attention mechanism with linear complexity to the length of input sequences [198]. TransTab features transfer learning in tabular tasks using transformers [199] and also supports the cross-table transfer. Our approach is different from TransTab in that TransTab

has limited ability in generalizing to tables from new domains, while XTab is able to generalize to new domains.

Cross-table transfer learning. Pretrained vision and text models can be adapted to a wide range of tasks [204]. One reason is that the sentences and images share general representations across various tasks. As for tabular learning, one may question if there is shared knowledge across tables as two different tables can have totally different numbers of columns and the associated semantic meanings. We argue that different tables share a similar prior given the recent success of zero-shot hyperparameter optimization (HPO) in AutoML [205], which learns a general hyperparameter configuration applicable to a wide range of tabular tasks. Unlike pretrained models in NLP [187], XTab does not attempt to learn a universal tokenizer for all tables, as the meaning and context of each table varies. Instead, we aim to learn a weight initialization that is generalizable to various downstream tasks. Concurrent to our work, tabular prior-data fitted networks (TabPFN) [206] learns a prior model on synthetic tabular data and demonstrated promising results on small numerical tabular classification tasks with ≤ 1000 samples. Different from TabPFN, the inference complexity of XTab is irrelevant to the number of training samples. Thus, XTab also works for large tables.

B.3 Methods

Previous works have proposed various pretraining methods for tabular learning [192, 193, 197, 203]. However, existing pretrained models are still domain-specific since they were pretrained on the training set of each individual tabular

prediction task. As a result, existing pretrained models lack generalizability and fail to cover downstream tasks on other types of tables. Here, we propose XTab to pretrain transformer models using the information from multiple tables. With cross-table pretraining, XTab aims to learn the shareable knowledge that can boost the performance for various downstream regression and classification tasks.

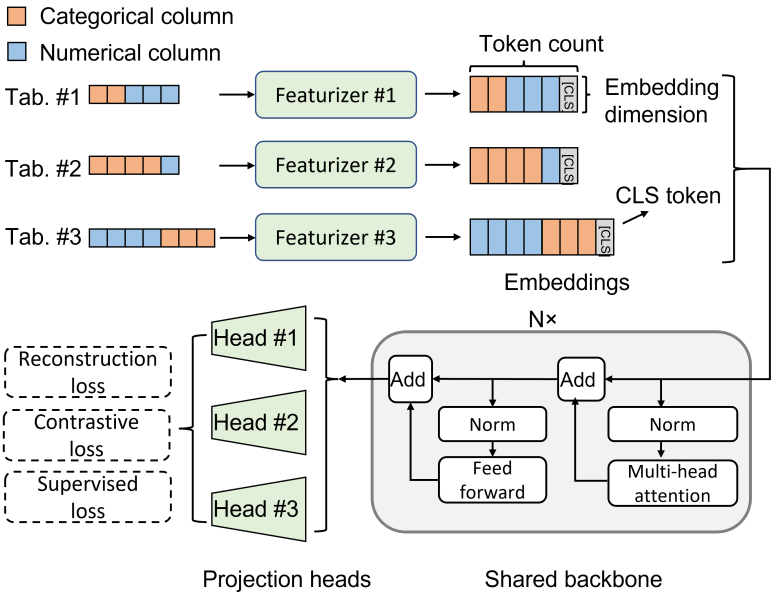

Figure B.1: The model structure of XTab. XTab is pretrained on multiple tabular tasks (Tab. #1, #2, #3). Samples from different tables are featurized and fed into a transformer model with N blocks. The output of the transformer is further processed by projection heads to derive the pretraining losses. Featurizers and projection heads are data-specific since tables may have different input/output dimensions. The transformer backbone is shared across all pretraining tables to capture the general knowledge.

B.3.1 Model structure

The model structure of XTab is described in Figure B.1. During the pretraining phase, we sample mini-batches of rows from different tables (one batch per table). The featurizers are data-specific and convert each column of the table to a token embedding. An additional [CLS] token is appended during this step

for supervised prediction or contrastive self-supervised pretraining [199]. A transformer-based backbone is shared across all tabular datasets to process token embeddings with variable sequence lengths. The output of the shared backbone is further processed by projection heads to (1) reconstruct the original table from a corrupted view; (2) identify the positive/negative pairs of samples as in contrastive learning; or (3) predict the values in the label column predefined by each table. The projection heads are not shared across tables since they are specific to each dataset and the pretraining objectives. Among all pretraining losses, reconstruction loss and contrastive loss do not require information from the label column, whereas supervised losses use the groundtruth data in the label columns of each table. Using groundtruth information during the pretraining phase is referred to as "target-aware pretraining" [199, 203] or "pre-finetuning" [207] in previous works.

A key challenge in cross-table pretraining lies in the variations of input tables. Previous works on transferable tabular learning either require tables to come from similar domains [208] or use additional information (e.g., column names) to identify the shared knowledge across tables. XTab is designed to be applicable to previously unseen tables with no assumption on the domain or column name format. To this end, XTab contains model blocks that carry the data-specific information (green blocks in Figure B.1), as well as the shared backbone that stores the common knowledge (grey blocks in Figure B.1). Once pretrained, only a shared backbone is kept for all downstream tasks. For each downstream task, featurizers and projection heads are randomly initialized and the entire model is finetuned on the downstream training data until a stopping criterion is met.

Featurizers The featurizers convert a sample to feature embeddings $E \in \mathbb{R}^{c \times d}$. Here, c denotes the number of columns and d is the embedding dimension. Each row of a table is considered as an input sample, and each column is a token. The embedding of [CLS] token is appended to the feature embedding for prediction stack$[E, \texttt{[CLS]}] \in \mathbb{R}^{c+1 \times d}$. In this work, we limit our discussion to tables with numerical and categorical columns. Text cells are treated as categorical attributes. Our tokenizer is similar to [196]. For numerical features, we multiply the numerical value x_k at the k-th column with a trainable vector $W_k \in \mathbb{R}^d$ and add a bias term b_k. For categorical columns, XTab learns an embedding matrix $\in \mathbb{R}^{N_{cat} \times d}$ as a lookup table, where N_{cat} is the total number of categories of the dataset. During the forward pass, we retrieve the categorical feature embeddings from the embedding matrix.

XTab allows tables to have different numbers of columns and arbitrary column types. Featurizers are data-specific to handle various types and numbers of columns in the input.

Backbones As the shared component across multiple pretraining datasets, transformers can handle input sequences with variable lengths. Therefore, it is possible to pretrain a tabular transformer that can be applied to all tabular datasets. Compared with other deep learning architectures like multi-layer perceptron (MLP), transformers are favorable for cross-table knowledge transfer since they can handle variable input sequences [199]. As long as the backbone can process input sequences of variable lengths, XTab is flexible on the exact implementation. In this work, we present three backbone variants:

FT-Transformer: Feature Tokenizer Transformer (FT-Transformer) is a simple yet well-performing transformer model for tabular prediction tasks [196].

The transformer module in FT-Transformer consists of a Multi-Head Self-Attention (MHSA) block and a Feed Forward block [188]. Recent work has found FT-Transformers to beat other deep learning methods on tabular data [189].

Fastfromer: Conventional Transformer-like architectures have a quadratic complexity to the length of input sequence [188], making them inefficient for tables with large numbers of columns. Fastfromer is an efficient transformer architecture which uses additive attention in place of MHSA [198]. With additive attention, Fastformer only considers the interaction between each token and the global representation, achieving a linear complexity.

Saint-v: Saint has introduced the row-wise attention in addition to the column-wise attention of FT-Transformer and Fastformer [197]. The original implementation of Saint is sensitive to the sequence length and can not handle variable-column tables [197]. We present a variation of Saint (Saint-v) to fit into our cross-table pretraining setting. Saint-v consists of both column- and row-wise attention blocks.

Projection heads and objectives There exist various pretraining objectives for tabular prediction tasks [192–194, 199, 202, 203]. Among them, table reconstruction and contrastive learning are the most popular and effective objectives for tabular tasks. In addition to the self-supervised pretraining objectives, we also tested the pre-finetuning setting using supervised loss.

Reconstruction loss: Reconstruction loss is a self-supervised training objective shown to be effective on various tabular tasks [194, 203]. The reconstruction objective aims to recover the original sample x from a corrupted view of

the sample $\tilde{x}$. The reconstruction projection head takes the representation of $\tilde{x}$ as input, and generates an estimate of the original input $\hat{x}$. The reconstruction loss is calculated by comparing x and $\hat{x}$. Specifically, we use Cross-Entropy loss to measure the reconstruction error of categorical columns and Mean Squared Error (MSE) for numerical columns.

Contrastive loss: Similar to the reconstruction objective, we also generate $\tilde{x}$ as a corrupted sample. x and its corresponding corruption $\tilde{x}$ are considered as a positive pair of samples, whereas x and other samples in the batch form negative sample pairs. In general, contrastive loss aims to minimize the distance between positive pairs of samples and maximize the distance for negative pairs. Following [191, 193], we used InfoNCE loss for contrastive cross-table pretraining. The contrastive projection heads are similar to those used in SimCLR [191], mapping the representations to the space where we apply the contrastive loss.

Supervised loss: In addition to reconstruction and contrastive losses that do not require labels in pretraining, one can directly pretrain a model using the supervised objective. With supervised losses, the projection head aims to predict the values under a certain field (or column), as predefined by each dataset. The supervised prediction tasks included regression and classification.

In XTab, the projection heads are data-specific. Different pretraining datasets do not need to share common objectives. For example, we can simultaneously pretrain XTab on both regression and classification tasks, or a mixture of reconstruction and contrastive losses. The diversity of pretraining objectives ensures that the shared backbone is widely adaptable to various downstream tables.

B.3.2 Federated pretraining

XTab introduces data-specific featurizers and projection heads (green blocks in Figure B.1) to account for the variations across table columns and pretraining objectives. During pretraining, both the time and space complexity increase linearly as we include more tabular datasets. As a result, it is challenging to quickly pretrain XTab using a single machine on a large collection of tabular tasks. To alleviate this issue, we fit XTab into the federated learning framework [209]. With the federated setting, XTab involves only marginal overhead in wall-clock time with more pretraining tasks. Federated learning makes it feasible to pretrain XTab on a cluster of commercially available GPUs (NVIDIA T4 GPUs, 16GB memory).

We use the Federated Averaging (FedAvg) algorithm to pretrain XTab [209, 210]. We have a central server and multiple clients. Each client only hosts one dataset. Therefore, we can distribute the data-specific components of XTab across clients such that each client stores one featurizer, one projection head, and the shared transformer. During pretraining, each client calculates the gradient using the local dataset:

$$w_{k,i+1} \leftarrow w_{k,i} - \alpha \nabla \ell_k, \tag{B.1}$$

where k denotes the client (or table) index and i shows the current iteration. α is the learning rate and $\ell^{(k)}$ is the loss function. w represents the trainable parameters which contains two components: $w^{(\mathrm{S})}$ for the shareable modules across all pretraining tasks, and $w^{(\mathrm{NS})}$ for the non-shareable parts ($w = \mathrm{stack}[w^{(\mathrm{NS})}, w^{(\mathrm{S})}]$). All clients operate synchronously during pretraining with the same learning rate and batch size.

The central server is responsible for aggregating the local gradients from

clients. FedAvg allows clients to make multiple local updates before an aggregation step is made on the central server. Let N denote the number of local updates per aggregation. The central server performs:

$$w_{i+N}^{(\mathrm{S})} \leftarrow w_i^{(\mathrm{S})} + \sum_{k=1}^{K} (w_{k,i+N}^{(\mathrm{S})} - w_i^{(\mathrm{S})}). \tag{B.2}$$

The aggregation is only performed on the shared weights. The term $w_{k,i+N}^{(\mathrm{S})} - w_i^{(\mathrm{S})}$ is the gradient learned by client k since the last weight aggregation. The central server simply accumulates the gradients from all clients. Such unitary scalarization was recently shown to perform well in multi-task learning [211].

After the aggregation update (i.e., Equation B.2), all clients download $w_{i+N}^{(\mathrm{S})}$ from the central server, and apply the weights to the transformer backbone $w_{k,i+N} = \mathrm{stack}[w_{k,i+N}^{(\mathrm{NS})}, w_{i+N}^{(\mathrm{S})}]$. Therefore, we force all clients to train on a shared backbone with data-specific featurizers and projection heads.

The number of local steps N is a key parameter to control communication efficiency. With $N = 1$, FedAvg corresponds to the distributed version of stochastic gradient descent (SGD). With $N > 1$, multiple local updates are performed between model aggregation steps at the server, thereby reducing the communication cost between the central server and clients. Unless otherwise specified, we choose $N = 5$ throughout the work.

Federated learning was originally proposed as a privacy-preserving approach to learning from distributed data. The collaboration of multiple clients to train a single shared model makes a good fit with our goal of cross-table pretraining. In this work, XTab leverages the distributed nature of federated learning to scale with a large number of pretraining tasks.

B.4 Experiments

We evaluate the performance of XTab on supervised tabular learning tasks, including binary and multiclass classification and regression. We tested on the following pretraining settings:

- XTab with various pretraining objectives, including reconstruction loss, contrastive loss, and supervised loss.
- XTab with various transformer backbones, including FT-Transformer, Fastformer, and Saint-v.
- XTab with the transformer backbone partially- or fully-pretrained from other tasks.
- XTab with different numbers of pretraining tasks.

During finetuning, we randomly initialize a new featurizer and projection head for each downstream task. All downstream tasks use the pretrained transformer backbone. We finetune all the model components using the training set of each downstream task. We included two different finetuning settings:

- Light finetuning: finetune XTab for a fixed number of epochs (3 epochs).
- Heavy finetuning: finetune XTab with an early stopping patience of 3 epochs. The maximum number of epochs is set to infinity in this case.

For all finetuning settings, we retrieve the best model checkpoint based on validation scores, and use it to report the performance on the test data. The baseline models share the same model architecture and finetuning configurations as XTab, but with randomly initialized parameters instead of using the pretrained backbones. We find that XTab generally outperforms the baseline models in all scenarios and beats other deep learning models on tabular tasks.

B.4.1 Datasets

We use the public OpenML-AutoML Benchmark (AMLB) [190] for pretraining and evaluation. AMLB is a recently proposed benchmark for automated machine learning, consisting of 104 tabular tasks (71 classification and 33 regression). Out of the 104 tabular datasets, we used 52 datasets for pretraining and the remaining 52 tasks for finetuning and evaluation. We split the pretraining and finetuning datasets by the alphabetical order of the task names.

Data split: For all downstream (or finetuning) tasks, AMLB reserves 10% of the tabular data for testing. Over the remaining data, we randomly partition 87.5% (7/8) into the training set and use 12.5% (1/8) for validation. We repeated 5 trials with different test folds for all tabular datasets. All methods use the same split within the same trial.

Data pre-processing: Following [193, 197, 199], we limit the discussion to tables with numerical and categorical columns. Each Category is represented by a distinct integer to index the embedding in the lookup table of the categorical featurizer (see Section B.3.1 for details). We normalized the numerical features by subtracting the mean and dividing them by the standard deviation. For regression tasks, we also apply the Standardization to the labels. The normalization parameters are calculated using the training set only to avoid information leakage. Missing entries are filled with the mean values of numerical columns, or treated as an additional category for categorical columns.

Table corruption: Self-supervised learning objectives, including both contrastive and reconstruction losses, require a corrupted view of the input sample. In this work, we follow [193, 203] to randomly resample features and construct a corrupted sample. Specifically, we randomly select a fraction of features at

each row of the table. Those features are corrupted by resampling from the empirical marginal distribution of the column. For all datasets, the corruption ratio was set to 60% as suggested in [193]. In other words, for each sample x and its corrupted view $\tilde{x}$, 60% of entries are resampled whereas 40% of features remain unchanged.

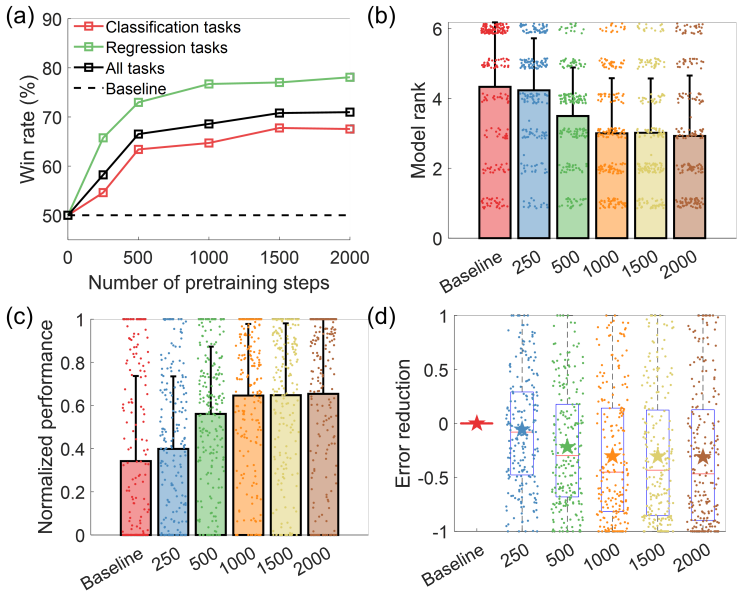

Figure B.2: Tabular prediction performance of XTab using various evaluation criteria under the light finetuning setting. **(a)** The win rate of the pretrained transformer with respect to baseline. **(b)** The average rank of the models. **(c)** The normalized prediction performance. **(d)** The average error reduction rate compared to baseline. Each dot indicates a trial of the downstream task (5 trials per dataset). The error bars show standard deviations in **(b)** and **(c)**. As the backbone is pretrained for more steps, we observe an increase in all evaluation criteria.

B.4.2 Experimental setup

We used a federated pretraining setting as detailed in Section B.3.2. Both pretraining and finetuning were performed on a cloud cluster of NVIDIA T4 GPUs (16 GB memory). We used about 30 thousand GPU hours for all experiments.

Model configuration and training: Our default model configuration of

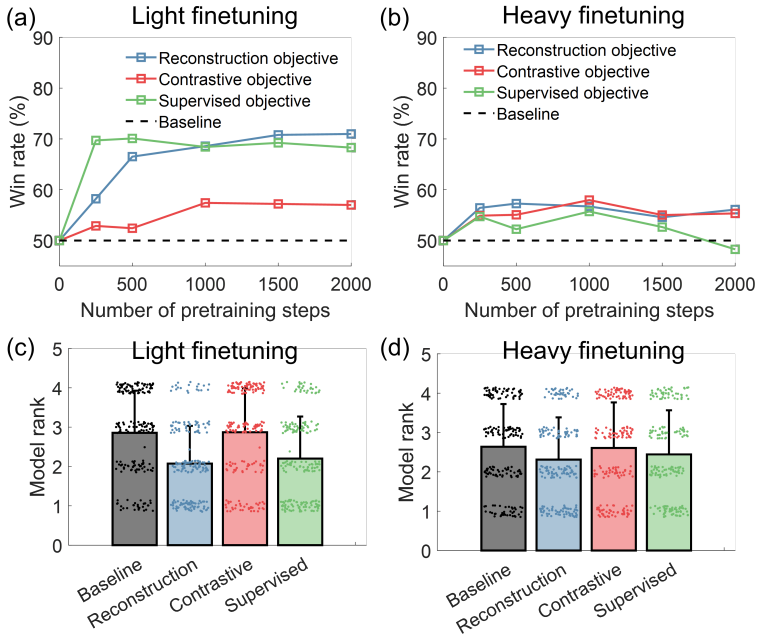

Figure B.3: Comparison of different pretraining objectives under the light **(a, c)** and heavy **(b, d)** finetuning settings. We show the win rate of XTab with different objectives with **(a)** light and **(b)** heavy finetuning settings. We also compared the performance of pretraining objectives in terms of the model rank with **(c)** light and **(d)** heavy finetuning. We observe a consistent improvement of XTab compared to baseline models with all objectives. The reconstruction pretraining objective achieves the best performance, with 71.0% win rate under light finetuning and 56.1% for heavy finetuning at 2000 pretraining steps.

transformer variants is the same as [196], with 3 transformer blocks, a feature embedding size of 192 and 8 attention heads. The feed forward networks (Figure B.1) have two layers with the same size as the embedding. We apply a dropout ratio of 20% to attention layers and 10% for feed forward networks. We use ReGLU [212] as the activation function and layer normalization [213] in the feed forward layers. The projection heads are ReLU networks with 2 layers and a hidden dimension of 192. All model components use *Kaiming* initialization [214] with the bias terms fixed at zeros.

The batch size is fixed at 128 for both pretraining and finetuning. Both stages use AdamW as the optimizer, with a learning rate of 1e-4. Following [196, 203],

we also apply a weight decay of 1e-5 to all components excluding featurizers, [CLS] tokens, layer normalization and bias terms.

Evaluation metrics: We choose the evaluation metrics as suggested by AMLB [190]. We use root mean-squared error (RMSE) for regression tasks, area under the receiver operating characteristic curve (AUC) for binary classification, and log loss for multi-class classification. The same evaluation metrics are applied to validation sets for early stopping. The efficacy of the pretrained transformer backbones is estimated by the downstream performance.

B.4.3 Comparison with baseline transformers

Cross-table pretraining improves downstream task performance. As shown in Figure B.2, we compare the downstream prediction performance of FT-Transformer before (baseline) and after cross-table pretraining. Reconstruction objective is used for pretraining and all downstream tasks are finetuned for 3 epochs (light finetuning). We checkpoint the pretrained backbone after a certain number of pretraining steps and finetune downstream tasks from various checkpoints (250/500/1000/1500/2000). In Figure B.2(a), we show the win rate of the pretrained transformer on all downstream tasks with respect to baseline. Both classification and regression tasks benefit from our proposed cross-table pretraining. As the backbone is pretrained for more steps, we observe an increase in the win rate. We also calculate the rank of the model for each downstream task (Figure B.2(b)). Model rank is an integer from 1 to 6, with a lower number indicating better performance. Equal values are assigned a rank that is the average of the ranks of those values. The rank of the model improves with XTab pretraining. To further validate the advantage of XTab over transformers

without cross-table pretraining, we further look into the normalized prediction performance and error reduction rate (Figure B.2(c, d)). We min-max normalize the prediction performance of all models, such that the worst model receives a score of 0 and the best model receives 1. Similarly, errors are also normalized to the best and worst models. Negative numbers indicate a model with lower error (1 − AUC scores for binary classification) or loss (log loss for multiclass classification and RMSE for regression) than baseline. The mean error (or loss) is indicated by the stars. FT-Transformers pretrained with XTab on average obtain higher normalized performance and reduced error compared to traditional random initialization.

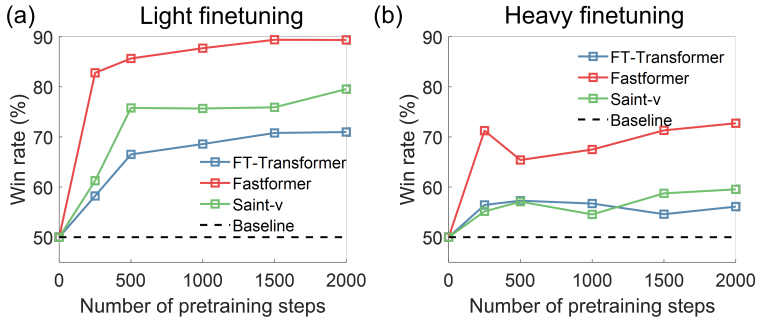

Figure B.4: XTab with transformer variants including FT-Transformer, Fastformer, and Saint-v. We use different transformer models as the shared backbone in XTab. We calculate the win rate of the pretrained backbone over randomly initialized transformers. **(a)** shows the results for light finetuning and **(b)** represents heavy finetuning. FT-Transformer, Fastformer, and Saint-v all benefit from our proposed cross-table pretraining, achieving >50% win rate in all experiments.

XTab with different pretraining objectives and finetuning settings. We extensively test XTab with various pretraining objectives and finetuning settings. Figure B.3 summarizes the downstream performance using reconstruction, contrastive and supervised objectives as described in Section B.3.1. We use FT-Transformer as the backbone. Figure B.3(a, b) plot the win rate of XTab under the light and heavy finetuning settings, respectively. We finetune on all downstream tasks for 3 epochs with light finetuning, and use an early stopping

patience of 3 for heavy finetuning. We observe a consistent improvement of XTab over the baseline with no cross-table pretraining. The advantage of XTab is more significant in the light finetuning setting compared to heavy finetuning. For example, XTab with the reconstruction objective achieves a 71.0% win rate with light finetuning, but only 56.1% with heavy finetuning. The difference is caused by catastrophic forgetting of deep models [215, 216]. As tabular transformers are relatively small (<1M parameters for the FT-Transformer backbone), they are more vulnerable to catastrophic forgetting during the finetuning phase. It is possible to alleviate this issue with additional techniques [215, 216], but this is outside the scope of the work. Figure B.3(c, d) compare different objectives by ranking the models with light and heavy finetuning. All approaches are pretrained for 2000 steps. Each dot in Figure B.3(c, d) represents a trial of downstream experiments (5 trials per dataset) and error bars indicate the standard deviations across trials. The advantage of cross-table pretraining is shown by a win rate >50% and a model rank value lower than the baseline. We conclude that XTab consistently enhances the downstream performance of tabular transformers across multiple pretraining objectives and finetuning settings. Among all pretraining objectives tested, reconstruction loss performs better than contrastive or supervised losses.

XTab is applicable to various types of transformers. XTab offers a framework to pretrain the shared model components across tabular tasks. Therefore, the choice of transformer backbone is flexible, as long as the model can process tables with variable columns. In Figure B.4, we plug three transformer variants into XTab including FT-Transformer, Fastformer, and Saint-v. The explanation of transformer backbones can be found in Section B.3.1. We pretrain all transformers using reconstruction objective, and finetune on the downstream tasks

with the light and heavy settings, Figure B.4(a, b). We show that XTab is applicable to various types of transformers and all models benefit from the proposed cross-table pretraining, achieving a higher win rate compared to the baseline.

B.4.4 Performance compared to traditional baselines

To compare the performance of XTab and various tabular models, we run experiments on the full AutoML Benchmark [190]. We split the benchmark into 2 folds, each consisting of 52 tabular datasets. We pretrain on fold #1 and evaluate the downstream performance on fold #2 and vice versa. We pretrain XTab with the FT-Transformer backbone using reconstruction loss. 20 datasets are excluded since they could not fit into the GPU memory (16 GB). We report the performance on the remaining 84 tasks. In addition to XTab, we include the following methods:

Tree-based models: Tree-based models provide strong performance on tabular tasks [189]. We include Random Forest (RF) and gradient-boosted tree variants: XGBoost [83], LightGBM [109] and CatBoost [217]. **Neural networks:** We include the AutoGluon neural networks implemented on top of PyTorch [218] and the FastAI tabular model [219]. **Transformers:** We include the FT-Transformer which is a direct counterpart of XTab without pretraining. The finetuning settings of FTT/XTab include light (FTT-l/XTab-l) and heavy (FTT-h/XTab-h) finetuning as described above. We further introduce FTT-best/XTab-best, which incorporates an early-stopping patience of 20 and model soup of the top 3 checkpoints [220] to achieve better performance. TransTab is included for comparison on classification tasks (regression not enabled yet with TransTab) under the supervised learning (TransTab-sl) and contrastive learning (TransTab-cl) settings [199].

Table B.1: Comparison of tabular prediction performance with default model configuration and hyperparameter optimization (HPO). Mean training time and model rank (± standard deviation) are calculated across 84 datasets from AutoML Benchmark. We perform 5 independent trials for each task. XTab outperforms its counterpart FTT in all scenarios thanks to cross-table pretraining, whereas CatBoost is the overall best model. The best overall method (CatBoost) and the best deep learning approach (XTab-best) are highlighted in **bold**.

	Methods	Time (s)	Rank
Default hyperparameter	RF	66.8†	7.14 ± 3.81
	XGBoost	43.1†	5.06 ± 3.08
	LightGBM	23.9†	5.23 ± 3.25
	CatBoost	**322.8**†	**2.98 ± 2.66**
	FastAI	89.6	7.24 ± 3.44
	NN	188.8	7.40 ± 3.43
	TransTab-sl*	539.7	11.04 ± 2.75
	TransTab-cl*	312.0	10.79 ± 3.00
	FTT-l	189.2	10.19 ± 2.43
	XTab-l	189.8	9.21 ± 2.57
	FTT-h	532.5	7.29 ± 2.20
	XTab-h	506.3	6.93 ± 2.09
	FTT-best	810.9	4.94 ± 2.25
	XTab-best	**755.9**	**4.39 ± 2.36**
HPO	RF	1084.4†	5.00 ± 2.40
	XGBoost	862.3†	3.69 ± 2.45
	LightGBM	285.0†	4.40 ± 1.93
	CatBoost	**1529.3**†	**3.25 ± 2.10**
	FastAI	549.7	5.24 ± 2.38
	NN	1163.5	5.32 ± 2.20
	FTT	2221.1	4.58 ± 2.08
	XTab	**2335.3**	**4.51 ± 2.00**

† CPU training time.

* Only evaluated on classification tasks.

Table B.1 shows the performance of models with the default hyperparameters and hyperparameter optimization (HPO). With the default hyperparameter, we pretrain XTab for 2000 rounds, whereas the number of pretraining rounds is

tuned under the HPO setting. We use the AutoGluon default hyperparameters for tree-based models as they outperform the official defaults to give a strong baseline [218]. CatBoost is the state-of-the-art model on tabular tasks, which agrees with the recent finding in [189]. With cross-table pretraining, XTab improves the performance over FTT under light (FTT-l/XTab-l) and heavy (FTT-h/XTab-h) finetuning. Using more finetuning time, XTab-best achieves second place in the benchmark and beats other deep learning models. The success of XTab using the default configuration ensures that the pretrained backbone is widely applicable to tabular tasks, without the need for case-by-case tuning.

With HPO, we randomly search for data-specific hyperparameters on the validation performance. We allow a maximum number of 100 HPO trials within a 1-hour time budget. Table B.1 shows that gradient-boosted trees (i.e., XGBoost, LightGBM, CatBoost) achieve higher ranking with HPO, since they are generally faster to train. The search space is also smaller for tree models as they have fewer meaningful hyperparameters and well-known highly performant search spaces. The ranks are calculated separately for default hyperparameters and HPO and are not comparable across the two settings. The advantage of XTab over FTT increases as we allocate less training time for downstream tasks (XTab-l $\leftarrow$ XTab-h $\leftarrow$ XTab-best $\leftarrow$ XTab with HPO). Therefore, one should use pretrained foundation models instead of randomly initialized weights for tabular transformers, especially with a tight training budget.

B.5 Conclusion

In this work, we present XTab to improve the performance of deep tabular models. XTab pretrains tabular transformers with a diverse collection of data tables,

and can improve the tabular prediction performance of an unseen table from arbitrary domains. XTab handles the cross-table variations by separating the models into data-specific and shared components, and encourages the shared components to learn general knowledge for tabular prediction. We also propose to combine self-supervised pretraining with federated learning to improve pretraining efficiency, where client-side nodes perform table reconstruction tasks followed by backbone averaging updates at the server. Our results suggest that finetuning from the pretrained transformer is superior to training tabular transformers from scratch. One limitation of XTab is that it still falls behind CatBoost. This motivates future works on bridging the gap between pretrained tabular deep learning models and tree models. Another interesting direction is to combine XTab with language/vision foundation models for improving multimodal learning.

BIBLIOGRAPHY

[1] Simon Little, Alex Pogosyan, Spencer Neal, Baltazar Zavala, Ludvic Zrinzo, Marwan Hariz, Thomas Foltynie, Patricia Limousin, Keyoumars Ashkan, James FitzGerald, et al. Adaptive deep brain stimulation in advanced parkinson disease. *Annals of neurology*, 74(3):449–457, 2013.

[2] Lin Yao, Peter Brown, and Mahsa Shoaran. Improved detection of Parkinsonian resting tremor with feature engineering and Kalman filtering. *Clinical Neurophysiology*, 131(1):274–284, 2020.

[3] Meng-Chen Lo and Alik S Widge. Closed-loop neuromodulation systems: next-generation treatments for psychiatric illness. *International review of psychiatry*, 29(2):191–204, 2017.

[4] Youssef Ezzyat, Paul A Wanda, Deborah F Levy, Allison Kadel, Ada Aka, Isaac Pedisich, Michael R Sperling, Ashwini D Sharan, Bradley C Lega, Alexis Burks, et al. Closed-loop stimulation of temporal cortex rescues functional networks and improves memory. *Nature Communications*, 9(1):1–8, 2018.

[5] Iñaki Iturrate, Michael Pereira, and José del R Millán. Closed-loop electrical neurostimulation: challenges and opportunities. *Current Opinion in Biomedical Engineering*, 8:28–37, 2018.

[6] Mark Huang, Richard L Harvey, Mary Ellen Stoykov, Sean Ruland, Martin Weinand, David Lowry, and Robert Levy. Cortical stimulation for upper limb recovery following ischemic stroke: a small phase ii pilot study of a fully implanted stimulator. *Topics in stroke rehabilitation*, 15(2):160–172, 2008.

[7] Andres M Lozano, Nir Lipsman, Hagai Bergman, Peter Brown, Stephan Chabardes, Jin Woo Chang, Keith Matthews, Cameron C McIntyre, Thomas E Schlaepfer, Michael Schulder, et al. Deep brain stimulation: current challenges and future directions. *Nature Reviews Neurology*, 15(3):148–160, 2019.

[8] Joachim K Krauss, Nir Lipsman, Tipu Aziz, Alexandre Boutet, Peter Brown, Jin Woo Chang, Benjamin Davidson, Warren M Grill, Marwan I Hariz, Andreas Horn, et al. Technology of deep brain stimulation: current status and future directions. *Nature Reviews Neurology*, pages 1–13, 2020.

[9] Tara L Skarpaas and Martha J Morrell. Intracranial stimulation therapy for epilepsy. *Neurotherapeutics*, 6(2):238–243, 2009.

[10] Anders Christian Meidahl, Gerd Tinkhauser, Damian Marc Herz, Hayriye Cagnan, Jean Debarros, and Peter Brown. Adaptive deep brain stimulation for movement disorders: the long road to clinical therapy. *Movement disorders*, 32(6):810–819, 2017.

[11] Jerald Yoo, Long Yan, Dina El-Damak, Muhammad Awais Bin Altaf, Ali H Shoeb, and Anantha P Chandrakasan. An 8-channel scalable EEG acquisition SoC with patient-specific seizure classification and recording processor. *IEEE journal of solid-state circuits*, 48(1):214–228, 2013.

[12] Mahsa Shoaran, Benyamin Allahgholizadeh Haghi, Milad Taghavi, Masoud Farivar, and Azita Emami-Neyestanak. Energy-efficient classification for resource-constrained biomedical applications. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 8(4):693–707, 2018.

[13] Hossein Kassiri, Sana Tonekaboni, M Tariqus Salam, Nima Soltani, Karim Abdelhalim, Jose Luis Perez Velazquez, and Roman Genov. Closed-loop neurostimulators: A survey and a seizure-predicting design example for intractable epilepsy treatment. *IEEE transactions on biomedical circuits and systems*, 11(5):1026–1040, 2017.

[14] Gerard O'Leary, David M Groppe, Taufik A Valiante, Naveen Verma, and Roman Genov. Nurip: Neural interface processor for brain-state classification and programmable-waveform neurostimulation. *IEEE Journal of Solid-State Circuits*, 53(11):3150–3162, 2018.

[15] Wei-Ming Chen, Herming Chiueh, Tsan-Jieh Chen, Chia-Lun Ho, Chi Jeng, Ming-Dou Ker, Chun-Yu Lin, Ya-Chun Huang, Chia-Wei Chou, Tsun-Yuan Fan, et al. A fully integrated 8-channel closed-loop neural-prosthetic cmos soc for real-time epileptic seizure control. *IEEE journal of solid-state circuits*, 49(1):232–247, 2014.

[16] Naveen Verma, Ali Shoeb, Jose Bohorquez, Joel Dawson, John Guttag, and Anantha P Chandrakasan. A micro-power EEG acquisition SoC with integrated feature extraction processor for a chronic seizure detection system. *IEEE journal of solid-state circuits*, 45(4):804–816, 2010.

[17] Fei Zhang, Mehdi Aghagolzadeh, and Karim Oweiss. An implantable neuroprocessor for multichannel compressive neural recording and on-the-fly spike sorting with wireless telemetry. In *2010 Biomedical Circuits and Systems Conference (BioCAS)*, pages 1–4. IEEE, 2010.

[18] Tianyu Zhan, Syyeda Zainab Fatmi, Sam Guraya, and Hossein Kassiri. A resource-optimized VLSI implementation of a patient-specific seizure

detection algorithm on a custom-made 2.2 cm^2 wireless device for ambulatory epilepsy diagnostics. *IEEE Transactions on Biomedical Circuits and Systems*, 13(6):1175–1185, 2019.

[19] Bingzhao Zhu, Uisub Shin, and Mahsa Shoaran. Closed-loop neural interfaces with embedded machine learning. In *2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pages 1–4. IEEE, 2020.

[20] Shenghong He, Fahd Baig, Abteen Mostofi, Alek Pogosyan, Jean Debarros, Alexander L Green, Tipu Z Aziz, Erlick Pereira, Peter Brown, and Huiling Tan. Closed-loop deep brain stimulation for essential tremor based on thalamic local field potentials. *Movement Disorders*, 36(4):863–873, 2021.

[21] Levin Kuhlmann, Philippa Karoly, Dean R Freestone, Benjamin H Brinkmann, Andriy Temko, Alexandre Barachant, Feng Li, Gilberto Titericz Jr, Brian W Lang, Daniel Lavery, et al. Epilepsyecosystem. org: crowd-sourcing reproducible seizure prediction with long-term human intracranial EEG. *Brain*, 141(9):2619–2630, 2018.

[22] Nhan Duy Truong, Luping Zhou, and Omid Kavehei. Semi-supervised seizure prediction with generative adversarial networks. In *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 2369–2372. IEEE, 2019.

[23] Daniel John DiLorenzo, Kent W Leyde, and Dmitry Kaplan. Neural state monitoring in the treatment of epilepsy: Seizure prediction—conceptualization to first-in-man study. *Brain sciences*, 9(7):156, 2019.

[24] Thomas Maiwald, Matthias Winterhalder, Richard Aschenbrenner-Scheibe, Henning U Voss, Andreas Schulze-Bonhage, and Jens Timmer. Comparison of three nonlinear seizure prediction methods by means of the seizure prediction characteristic. *Physica D: nonlinear phenomena*, 194(3-4):357–368, 2004.

[25] Christophe C Jouny, Piotr J Franaszczuk, and Gregory K Bergey. Improving early seizure detection. *Epilepsy & Behavior*, 22:S44–S48, 2011.

[26] Muhammad Awais Bin Altaf, Chen Zhang, and Jerald Yoo. A 16-channel patient-specific seizure onset and termination detection SoC with impedance-adaptive transcranial electrical stimulator. *IEEE Journal of Solid-State Circuits*, 50(11):2728–2740, 2015.

[27] Gianluca Coppola, Cherubino Di Lorenzo, Jean Schoenen, and Francesco Pierelli. Habituation and sensitization in primary headaches. *The journal of headache and pain*, 14(1):65, 2013.

[28] Domenico Restuccia, Catello Vollono, Ivana del Piero, Lucia Martucci, and Sergio Zanini. Different levels of cortical excitability reflect clinical fluctuations in migraine. *Cephalalgia*, 33(12):1035–1047, 2013.

[29] Gianluca Coppola, Victor De Pasqua, Francesco Pierelli, and Jean Schoenen. Effects of repetitive transcranial magnetic stimulation on somatosensory evoked potentials and high frequency oscillations in migraine. *Cephalalgia*, 32(9):700–709, 2012.

[30] Kathleen A Foley, Roger Cady, Vincent Martin, James Adelman, Merle Diamond, Christopher F Bell, Jeffrey M Dayno, and X Henry Hu. Treating early versus treating mild: timing of migraine prescription medications

among patients with diagnosed migraine. *Headache: The Journal of Head and Face Pain*, 45(5):538–545, 2005.

[31] Mahsa Shoaran, Masoud Shahshahani, Masoud Farivar, Joyel Almajano, Amirhossein Shahshahani, Alexandre Schmid, Anatol Bragin, Yusuf Leblebici, and Azita Emami. A 16-channel 1.1 mm 2 implantable seizure control soc with sub-μw/channel consumption and closed-loop stimulation in 0.18 μm cmos. In *VLSI Circuits (VLSI-Circuits), 2016 IEEE Symposium on*, pages 1–2. Ieee, 2016.

[32] Gianluca Coppola, Michel Vandenheede, Laura Di Clemente, Anna Ambrosini, Arnaud Fumal, Victor De Pasqua, and Jean Schoenen. Somatosensory evoked high-frequency oscillations reflecting thalamo-cortical activity are decreased in migraine patients between attacks. *Brain*, 128(1):98–103, 2005.

[33] Gianluca Coppola, Elisa Iacovelli, Martina Bracaglia, Mariano Serrao, Cherubino Di Lorenzo, and Francesco Pierelli. Electrophysiological correlates of episodic migraine chronification: evidence for thalamic involvement. *The journal of headache and pain*, 14(1):76, 2013.

[34] Marina De Tommaso, Anna Ambrosini, Filippo Brighina, Gianluca Coppola, Armando Perrotta, Francesco Pierelli, Giorgio Sandrini, Massimiliano Valeriani, Daniele Marinazzo, Sebastiano Stramaglia, et al. Altered processing of sensory stimuli in patients with migraine. *Nature Reviews Neurology*, 10(3):144, 2014.

[35] Gabriel Curio. Linking 600-hz "spikelike" eeg/meg wavelets ("ς-bursts") to cellular substrates: Concepts and caveats. *Journal of Clinical Neurophysiology*, 17(4):377–396, 2000.

[36] Yun Park, Lan Luo, Keshab K Parhi, and Theoden Netoff. Seizure prediction with spectral power of eeg using cost-sensitive support vector machines. *Epilepsia*, 52(10):1761–1770, 2011.

[37] Zehong Cao, Kuan-Lin Lai, Chin-Teng Lin, Chun-Hsiang Chuang, Chien-Chen Chou, and Shuu-Jiun Wang. Exploring resting-state eeg complexity before migraine attacks. *Cephalalgia*, 38(7):1296–1306, 2018.

[38] Ze-Hong Cao, Li-Wei Ko, Kuan-Lin Lai, Song-Bo Huang, Shuu-Jiun Wang, and Chin-Teng Lin. Classification of migraine stages based on resting-state eeg power. In *Neural Networks (IJCNN), 2015 International Joint Conference on*, pages 1–5. IEEE, 2015.

[39] Erfan Sayyari, Mohsen Farzi, Roohollah Rezaei Estakhrooeieh, Farzaneh Samiee, and Mohammad Bagher Shamsollahi. Migraine analysis through eeg signals with classification approach. In *Information Science, Signal Processing and their Applications (ISSPA), 2012 11th International Conference on*, pages 859–863. IEEE, 2012.

[40] Catherine D Chong, Nathan Gaw, Yinlin Fu, Jing Li, Teresa Wu, and Todd J Schwedt. Migraine classification using magnetic resonance imaging resting-state functional connectivity data. *Cephalalgia*, 37(9):828–844, 2017.

[41] Jean Schoenen. Neurophysiological features of the migrainous brain. *Neurological Sciences*, 27(2):s77–s81, 2006.

[42] Gianluca Coppola, Antonio Di Renzo, Emanuele Tinelli, Chiara Lepre, Cherubino Di Lorenzo, Giorgio Di Lorenzo, Marco Scapeccia, Vincenzo Parisi, Mariano Serrao, Claudio Colonnese, et al. Thalamo-cortical net-

work activity between migraine attacks: insights from mri-based microstructural and functional resting-state network correlation analysis. *The journal of headache and pain*, 17(1):100, 2016.

[43] Kenji Sakuma, Takao Takeshima, Kumiko Ishizaki, and Kenji Nakashima. Somatosensory evoked high-frequency oscillations in migraine patients. *Clinical neurophysiology*, 115(8):1857–1862, 2004.

[44] Domenico Restuccia, Catello Vollono, Ivana Del Piero, Lucia Martucci, and Sergio Zanini. Somatosensory high frequency oscillations reflect clinical fluctuations in migraine. *Clinical Neurophysiology*, 123(10):2050–2056, 2012.

[45] Sherifa A Hamed. A migraine variant with abdominal colic and alice in wonderland syndrome: a case report and review. *BMC neurology*, 10(1):2, 2010.

[46] Bo Hjorth. Eeg analysis based on time domain properties. *Electroencephalography and clinical neurophysiology*, 29(3):306–310, 1970.

[47] M Vourkas, S Micheloyannis, and G Papadourakis. Use of ann and hjorth parameters in mental-task discrimination. In *Advances in medical signal and information processing, 2000. First international conference on (IEE conf. publ. no. 476)*, pages 327–332. IET, 2000.

[48] Johannes Sarnthein, Jair Stern, Christoph Aufenberg, Valentin Rousson, and Daniel Jeanmonod. Increased eeg power and slowed dominant frequency in patients with neurogenic pain. *Brain*, 129(1):55–64, 2005.

[49] Jonathan E Walker. Qeeg-guided neurofeedback for recurrent migraine headaches. *Clinical EEG and Neuroscience*, 42(1):59–61, 2011.

[50] Richard J Staba, Matt Stead, and Gregory A Worrell. Electrophysiological biomarkers of epilepsy. *Neurotherapeutics*, 11(2):334–346, 2014.

[51] Ken Inoue, Isao Hashimoto, Takushi Shirai, Hideshi Kawakami, Takafumi Miyachi, Yasuyo Mimori, and Masayasu Matsumoto. Disinhibition of the somatosensory cortex in cervical dystonia—decreased amplitudes of high-frequency oscillations. *Clinical neurophysiology*, 115(7):1624–1630, 2004.

[52] Zisheng Zhang and Keshab K Parhi. Low-complexity seizure prediction from ieeg/seeg using spectral power and ratios of spectral power. *IEEE transactions on biomedical circuits and systems*, 10(3):693–706, 2016.

[53] Dan Li, Jianxin Zhang, Qiang Zhang, and Xiaopeng Wei. Classification of ecg signals based on 1d convolution neural network. In *e-Health Networking, Applications and Services (Healthcom), 2017 IEEE 19th International Conference on*, pages 1–6. IEEE, 2017.

[54] Hubert Cecotti and Axel Graeser. Convolutional neural network with embedded fourier transform for eeg classification. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4. IEEE, 2008.

[55] RK Maddula, J Stivers, M Mousavi, S Ravindran, and VR de Sa. Deep recurrent convolutional neural networks for classifying p300 bci signals. In *Proceedings of the 7th Graz Brain-Computer Interface Conference, Graz, Austria*, pages 18–22, 2017.

[56] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[57] Yonglong Tian, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning strong parts for pedestrian detection. In *Proceedings of the IEEE international conference on computer vision*, pages 1904–1912, 2015.

[58] Florian Mormann, Ralph G Andrzejak, Christian E Elger, and Klaus Lehnertz. Seizure prediction: the long and winding road. *Brain*, 130(2):314–333, 2006.

[59] Mahsa Shoaran, Benyamin A. Haghi, Milad Taghavi, Masoud Farivar, and Azita Emami. Energy-Efficient Classification for Resource-Constrained Biomedical Applications. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS)*, 2018.

[60] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[61] Tianqi Chen. Introduction to boosted trees. *University of Washing Computer Science. University of Washington*, 22:115, 2014.

[62] Abdulhamit Subasi and M Ismail Gursoy. Eeg signal classification using pca, ica, lda and support vector machines. *Expert systems with applications*, 37(12):8659–8666, 2010.

[63] Anil Jain and Douglas Zongker. Feature selection: Evaluation, application, and small sample performance. *IEEE transactions on pattern analysis and machine intelligence*, 19(2):153–158, 1997.

[64] Gianluca Coppola, Antonio Currà, Cherubino Di Lorenzo, Vincenzo Parisi, Manuela Gorini, Simona Liliana Sava, Jean Schoenen, and Francesco Pierelli. Abnormal cortical responses to somatosensory stimulation in medication-overuse headache. *BMC neurology*, 10(1):126, 2010.

[65] Jean Schoenen. Is chronic migraine a never-ending migraine attack? *Pain*, 152(2):239–240, 2011.

[66] Li-Wei Ko, Kuan-Lin Lai, Pei-Hua Huang, Chin-Teng Lin, and Shuu-Jiun Wang. Steady-state visual evoked potential based classification system for detecting migraine seizures. In *Neural Engineering (NER), 2013 6th International IEEE/EMBS Conference on*, pages 1299–1302. IEEE, 2013.

[67] Truett Allison, Charles C Wood, and William R Goff. Brain stem auditory, pattern-reversal visual, and short-latency somatosensory evoked potentials: latencies in relation to age, sex, and brain and body size. *Electroencephalography and Clinical Neurophysiology*, 55(6):619–636, 1983.

[68] Ali H Shoeb and John V Guttag. Application of machine learning to epileptic seizure detection. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 975–982, 2010.

[69] Zain Taufique, Bingzhao Zhu, Gianluca Coppola, Mahsa Shoaran, Wala Saadeh, and Muhammad Awais Bin Altaf. An 8.7 μj/class. fft accelerator and dnn-based configurable soc for multi-class chronic neurological disorder detection. In *2021 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, pages 1–3. IEEE, 2021.

[70] Zain Taufique, Bingzhao Zhu, Gianluca Coppola, Mahsa Shoaran, and Muhammad Awais Bin Altaf. A low power multi-class migraine detection processor based on somatosensory evoked potentials. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 68(5):1720–1724, 2021.

[71] Zisheng Zhang and Keshab K Parhi. Low-complexity seizure prediction

from iEEG/sEEG using spectral power and ratios of spectral power. *IEEE transactions on biomedical circuits and systems*, 10(3):693–706, 2015.

[72] Adam Page, Chris Sagedy, Emily Smith, Nasrin Attaran, Tim Oates, and Tinoosh Mohsenin. A flexible multichannel EEG feature extractor and classifier for seizure detection. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 62(2):109–113, 2014.

[73] Zisheng Zhang and Keshab K Parhi. Seizure detection using regression tree based feature selection and polynomial SVM classification. In *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 6578–6581. IEEE, 2015.

[74] Deepak Ranjan Nayak, Ratnakar Dash, and Banshidhar Majhi. Brain MR image classification using two-dimensional discrete wavelet transform and AdaBoost with random forests. *Neurocomputing*, 177:188–197, 2016.

[75] Yi Chen, Enyi Yao, and Arindam Basu. A 128-channel extreme learning machine-based neural decoder for brain machine interfaces. *IEEE transactions on biomedical circuits and systems*, 10(3):679–692, 2016.

[76] Lin Yao and Mahsa Shoaran. Enhanced classification of individual finger movements with ECoG. In *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pages 2063–2066. IEEE, 2019.

[77] Bingzhao Zhu, Gianluca Coppola, and Mahsa Shoaran. Migraine classification using somatosensory evoked potentials. *Cephalalgia*, 39(9):1143–1155, 2019.

[78] Zisheng Zhang and Keshab K Parhi. Seizure detection using wavelet decomposition of the prediction error signal from a single channel of intra-

cranial EEG. In *2014 36th annual international conference of the IEEE engineering in medicine and biology society*, pages 4443–4446. IEEE, 2014.

[79] Ashish Kumar, Saurabh Goyal, and Manik Varma. Resource-efficient machine learning in 2 kb ram for the internet of things. In *International Conference on Machine Learning*, pages 1935–1944. PMLR, 2017.

[80] Shoeb Shaikh, Rosa So, Tafadzwa Sibindi, Camilo Libedinsky, and Arindam Basu. Towards intelligent intracortical BMI (i^2BMI): Low-power neuromorphic decoders that outperform Kalman filters. *IEEE Transactions on Biomedical Circuits and Systems*, 13(6):1615–1624, 2019.

[81] Sven Peter, Ferran Diego, Fred A Hamprecht, and Boaz Nadler. Cost efficient gradient boosting. In *Advances in Neural Information Processing Systems*, pages 1551–1561, 2017.

[82] Miguel A Carreira-Perpinán and Pooya Tavallali. Alternating optimization of decision trees, with application to learning sparse oblique trees. *Advances in Neural Information Processing Systems*, 31:1211–1221, 2018.

[83] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[84] Milad Taghavi and Mahsa Shoaran. Hardware complexity analysis of deep neural networks and decision tree ensembles for real-time neural data classification. In *2019 9th International IEEE/EMBS Conference on Neural Engineering (NER)*, pages 407–410. IEEE, 2019.

[85] Bingzhao Zhu, Milad Taghavi, and Mahsa Shoaran. Cost-efficient classifi-

cation for neurological disease detection. In *2019 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 1–4. IEEE, 2019.

[86] Hyafil Laurent and Ronald L Rivest. Constructing optimal binary decision trees is np-complete. *Information processing letters*, 5(1):15–17, 1976.

[87] Mohammad Norouzi, Maxwell D Collins, Matthew Johnson, David J Fleet, and Pushmeet Kohli. Efficient non-greedy optimization of decision trees. *arXiv preprint arXiv:1511.04056*, 2015.

[88] Xiyang Hu, Cynthia Rudin, and Margo Seltzer. Optimal sparse decision trees. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[89] Thomas M Hehn and Fred A Hamprecht. End-to-end learning of deterministic decision trees. In *German conference on pattern recognition*, pages 612–627. Springer, 2018.

[90] Darryl Lin, Sachin Talathi, and Sreekanth Annapureddy. Fixed point quantization of deep convolutional networks. In *International Conference on Machine Learning*, pages 2849–2858, 2016.

[91] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[92] Joost B Wagenaar, Gregory A Worrell, Zachary Ives, MATTHIAS Dümpelmann, Brian Litt, and Andreas Schulze-Bonhage. Collaborating and sharing data in epilepsy research. *Journal of clinical neurophysiology: official publication of the American Electroencephalographic Society*, 32(3):235, 2015.

[93] Lin Yao, Peter Brown, and Mahsa Shoaran. Resting tremor detection in Parkinson's disease with machine learning and Kalman filtering. In *2018 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 1–4. IEEE, 2018.

[94] Kai J Miller, Dora Hermes, Christopher J Honey, Adam O Hebb, Nick F Ramsey, Robert T Knight, Jeffrey G Ojemann, and Eberhard E Fetz. Human motor cortical activity is selectively phase-entrained on underlying rhythms. *PLoS computational biology*, 8(9):e1002655, 2012.

[95] Kyong Ho Lee and Naveen Verma. A low-power processor with configurable embedded machine-learning accelerators for high-order and adaptive analysis of medical-sensor signals. *IEEE Journal of Solid-State Circuits*, 48(7):1625–1637, 2013.

[96] Sandhya Koteshwara and Keshab K Parhi. Incremental-precision based feature computation and multi-level classification for low-energy internet-of-things. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 8(4):822–835, 2018.

[97] Komail MH Badami, Steven Lauwereins, Wannes Meert, and Marian Verhelst. A 90 nm cmos, 6 μw power-proportional acoustic sensing frontend for voice activity detection. *IEEE Journal of Solid-State Circuits*, 51(1):291–302, 2015.

[98] Bingzhao Zhu and Mahsa Shoaran. Hardware-efficient seizure detection. In *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pages 2040–2043. IEEE, 2019.

[99] Bjoern H Menze, B Michael Kelm, Daniel N Splitthoff, Ullrich Koethe,

and Fred A Hamprecht. On oblique random forests. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 453–469. Springer, 2011.

[100] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[101] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in neural information processing systems*, pages 598–605, 1990.

[102] Chunhua Deng, Siyu Liao, Yi Xie, Keshab K Parhi, Xuehai Qian, and Bo Yuan. Permdnn: efficient compressed dnn architecture with permuted diagonal matrices. In *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 189–202. IEEE, 2018.

[103] Zhixiang Xu, Kilian Weinberger, and Olivier Chapelle. The greedy miser: Learning under test-time budgets. *arXiv preprint arXiv:1206.6451*, 2012.

[104] Ryutaro Tanno, Kai Arulkumaran, Daniel C Alexander, Antonio Criminisi, and Aditya Nori. Adaptive neural trees. *arXiv preprint arXiv:1807.06699*, 2018.

[105] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[106] Aditya Kusupati, Manish Singh, Kush Bhatia, Ashish Kumar, Prateek Jain, and Manik Varma. Fastgrnn: A fast, accurate, stable and tiny kilobyte sized gated recurrent neural network. In *Advances in Neural Information Processing Systems*, pages 9017–9028, 2018.

[107] Igor Fedorov, Ryan P Adams, Matthew Mattina, and Paul Whatmough. Sparse: Sparse architecture search for cnns on resource-constrained microcontrollers. In *Advances in Neural Information Processing Systems*, pages 4978–4990, 2019.

[108] Albert Gural and Boris Murmann. Memory-optimal direct convolutions for maximizing classification accuracy in embedded applications. In *International Conference on Machine Learning*, pages 2515–2524, 2019.

[109] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30:3146–3154, 2017.

[110] Uisub Shin, Cong Ding, Bingzhao Zhu, Yashwanth Vyza, Alix Trouillet, Emilie Revol, Stéphanie P Lacour, and Mahsa Shoaran. Neuraltree: A 256-channel 0.227 μj/class versatile neural activity classification and closed-loop neuromodulation soc. *IEEE Journal of Solid-State Circuits*, 57(11):3243–3257, 2022.

[111] Uisub Shin, Laxmeesha Somappa, Cong Ding, Yashwanth Vyza, Bingzhao Zhu, Alix Trouillet, Stephanie P Lacour, and Mahsa Shoaran. A 256-channel 0.227 μj/class versatile brain activity classification and closed-loop neuromodulation soc with 0.004 mm2-1.51 μw/channel fast-settling highly multiplexed mixed-signal front-end. In *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, volume 65, pages 338–340, 2022.

[112] Arman Zharmagambetov and Miguel Carreira-Perpinan. Smaller, more accurate regression forests using tree alternating optimization. In *International Conference on Machine Learning*, pages 11398–11408. PMLR, 2020.

[113] Bingzhao Zhu, Masoud Farivar, and Mahsa Shoaran. Resot: Resource-efficient oblique trees for neural signal classification. *IEEE Transactions on Biomedical Circuits and Systems*, 14(4):692–704, 2020.

[114] Ryutaro Tanno, Kai Arulkumaran, Daniel Alexander, Antonio Criminisi, and Aditya Nori. Adaptive neural trees. In *International Conference on Machine Learning*, pages 6166–6175. PMLR, 2019.

[115] Charles Mathy, Nate Derbinsky, José Bento, Jonathan Rosenthal, and Jonathan Yedidia. The boundary forest algorithm for online supervised and unsupervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[116] Andrew Silva, Matthew Gombolay, Taylor Killian, Ivan Jimenez, and Sung-Hyun Son. Optimization methods for interpretable differentiable decision trees applied to reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pages 1855–1865. PMLR, 2020.

[117] Scott Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*, 2017.

[118] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[119] Peter Kontschieder, Madalina Fiterau, Antonio Criminisi, and Samuel Rota Bulo. Deep neural decision forests. In *Proceedings of the IEEE international conference on computer vision*, pages 1467–1475, 2015.

[120] Hussein Hazimeh, Natalia Ponomareva, Petros Mol, Zhenyu Tan, and Rahul Mazumder. The tree ensemble layer: Differentiability meets condi-

tional computation. In *International Conference on Machine Learning*, pages 4138–4148. PMLR, 2020.

[121] Dan Steinberg and Phillip Colla. Cart: classification and regression trees. *The top ten algorithms in data mining*, 9:179, 2009.

[122] Jonathan Oliver. *Decision graphs: an extension of decision trees*. Citeseer, 1992.

[123] Hiroki Sudo, Koji Nuida, and Kana Shimizu. An efficient private evaluation of a decision graph. In *International Conference on Information Security and Cryptology*, pages 143–160. Springer, 2018.

[124] Jimmy Lin, Chudi Zhong, Diane Hu, Cynthia Rudin, and Margo Seltzer. Generalized and scalable optimal sparse decision trees. In *International Conference on Machine Learning*, pages 6150–6160. PMLR, 2020.

[125] Haoran Zhu, Pavankumar Murali, Dzung T Phan, Lam M Nguyen, and Jayant R Kalagnanam. A scalable mip-based method for learning optimal multivariate decision trees. *arXiv preprint arXiv:2011.03375*, 2020.

[126] Jeffrey P Bradford, Clayton Kunz, Ron Kohavi, Cliff Brunk, and Carla E Brodley. Pruning decision trees with misclassification costs. In *European Conference on Machine Learning*, pages 131–136. Springer, 1998.

[127] B Ravi Kiran and Jean Serra. Cost-complexity pruning of random forests. In *International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing*, pages 222–232. Springer, 2017.

[128] Haijian Shi. *Best-first decision tree learning*. PhD thesis, The University of Waikato, 2007.

[129] Arman Zharmagambetov, Suryabhan Singh Hada, Miguel Á Carreira-Perpiñán, and Magzhan Gabidolla. An experimental comparison of old and new decision tree algorithms. *arXiv preprint arXiv:1911.03054*, 2019.

[130] Uc irvine machine learning repository. http://archive.ics.uci.edu/ml/index.php. Accessed: 2021-05-02.

[131] Libsvm data. https://www.csie.ntu.edu.tw/ cjlin/libsvmtools. Accessed: 2021-05-02.

[132] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[133] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[134] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[135] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[136] Ro'ee Gilron, Simon Little, Randy Perrone, Robert Wilt, Coralie de Hemptinne, Maria S Yaroshinsky, Caroline A Racine, Sarah S Wang, Jill L Ostrem, Paul S Larson, et al. Long-term wireless streaming of neural recordings for circuit discovery and adaptive stimulation in individuals with parkinson's disease. *Nature biotechnology*, 39(9):1078–1085, 2021.

[137] Nicole R Provenza, Sameer A Sheth, Evan M Dastin-van Rijn, Raissa K Mathura, Yaohan Ding, Gregory S Vogt, Michelle Avendano-Ortega,

Nithya Ramakrishnan, Noam Peled, Luiz Fernando Fracassi Gelin, et al. Long-term ecological assessment of intracranial electrophysiology synchronized to behavioral markers in obsessive-compulsive disorder. *Nature Medicine*, 27(12):2154–2164, 2021.

[138] Yohann Thenaisie, Chiara Palmisano, Andrea Canessa, Bart J Keulen, Philipp Capetian, Mayte Castro Jiménez, Julien F Bally, Elena Manferlotti, Laura Beccaria, Rodi Zutt, et al. Towards adaptive deep brain stimulation: clinical and technical notes on a novel commercial device for chronic brain sensing. *Journal of Neural Engineering*, 18(4):042002, 2021.

[139] Flavio Raschellà, Stefano Scafa, Alessandro Puiatti, Eduardo Martin Moraud, and Pietro-Luca Ratti. Rbdact: Home screening of rem sleep behaviour disorder based on wrist actigraphy in parkinson's patients. *medRxiv*, 2022.

[140] Mariska J Vansteensel, Elmar GM Pels, Martin G Bleichner, Mariana P Branco, Timothy Denison, Zachary V Freudenburg, Peter Gosselaar, Sacha Leinders, Thomas H Ottens, Max A Van Den Boom, et al. Fully implanted brain–computer interface in a locked-in patient with als. *New England Journal of Medicine*, 375(21):2060–2066, 2016.

[141] Yohann Thenaisie, Kyuhwa Lee, Charlotte Moerman, Stefano Scafa, Andrea Gálvez, Elvira Pirondini, Morgane Buri, Jimmy Ravier, Alessandro Puiatti, Ettore Accolla, et al. Principles of gait encoding in the subthalamic nucleus of people with parkinson's disease. *medRxiv*, 2022.

[142] Joachim K Krauss, Nir Lipsman, Tipu Aziz, Alexandre Boutet, Peter Brown, Jin Woo Chang, Benjamin Davidson, Warren M Grill, Marwan I

Hariz, Andreas Horn, et al. Technology of deep brain stimulation: current status and future directions. *Nature Reviews Neurology*, 17(2):75–87, 2021.

[143] Omid G Sani, Yuxiao Yang, Morgan B Lee, Heather E Dawes, Edward F Chang, and Maryam M Shanechi. Mood variations decoded from multisite intracranial human brain activity. *Nature biotechnology*, 36(10):954–961, 2018.

[144] Katherine W Scangos, Ankit N Khambhati, Patrick M Daly, Ghassan S Makhoul, Leo P Sugrue, Hashem Zamanian, Tony X Liu, Vikram R Rao, Kristin K Sellers, Heather E Dawes, et al. Closed-loop neuromodulation in an individual with treatment-resistant depression. *Nature medicine*, 27(10):1696–1700, 2021.

[145] Alim Louis Benabid, Thomas Costecalde, Andrey Eliseyev, Guillaume Charvet, Alexandre Verney, Serpil Karakas, Michael Foerster, Aurélien Lambert, Boris Morinière, Neil Abroug, et al. An exoskeleton controlled by an epidural wireless brain–machine interface in a tetraplegic patient: a proof-of-concept demonstration. *The Lancet Neurology*, 18(12):1112–1122, 2019.

[146] Coralie De Hemptinne, Nicole C Swann, Jill L Ostrem, Elena S Ryapolova-Webb, Marta San Luciano, Nicholas B Galifianakis, and Philip A Starr. Therapeutic deep brain stimulation reduces cortical phase-amplitude coupling in parkinson's disease. *Nature neuroscience*, 18(5):779–786, 2015.

[147] Alfonso Fasano, Camila C Aquino, Joachim K Krauss, Christopher R Honey, and Bastiaan R Bloem. Axial disability and deep brain stimulation

in patients with parkinson disease. *Nature Reviews Neurology*, 11(2):98–110, 2015.

[148] Hayriye Cagnan, Timothy Denison, Cameron McIntyre, and Peter Brown. Emerging technologies for improved deep brain stimulation. *Nature biotechnology*, 37(9):1024–1033, 2019.

[149] Taige Wang, Mahsa Shoaran, and Azita Emami. Towards adaptive deep brain stimulation in parkinson's disease: Lfp-based feature analysis and classification. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2536–2540, 2018.

[150] Huiling Tan, Jean Debarros, Shenghong He, Alek Pogosyan, Tipu Z Aziz, Yongzhi Huang, Shouyan Wang, Lars Timmermann, Veerle Visser-Vandewalle, David J Pedrosa, et al. Decoding voluntary movements and postural tremor based on thalamic lfps as a basis for closed-loop stimulation for essential tremor. *Brain stimulation*, 12(4):858–867, 2019.

[151] Alan D Degenhart, William E Bishop, Emily R Oby, Elizabeth C Tyler-Kabara, Steven M Chase, Aaron P Batista, and Byron M Yu. Stabilization of a brain–computer interface via the alignment of low-dimensional spaces of neural activity. *Nature biomedical engineering*, 4(7):672–685, 2020.

[152] Shixian Wen, Allen Yin, Tommaso Furlanello, MG Perich, LE Miller, and Laurent Itti. Rapid adaptation of brain–computer interfaces to new neuronal ensembles or participants via generative modelling. *Nature Biomedical Engineering*, pages 1–13, 2021.

[153] Bingzhao Zhu and Mahsa Shoaran. Unsupervised domain adaptation for cross-subject few-shot neurological symptom detection. In *2021 10th In-*

ternational IEEE/EMBS Conference on Neural Engineering (NER), pages 181–184, 2021.

[154] Adelson Chua, Michael I Jordan, and Rikky Muller. Soul: An energy-efficient unsupervised online learning seizure detection classifier. *IEEE Journal of Solid-State Circuits*, 2022.

[155] Miaolin Zhang, Lian Zhang, Jeong Hoan Park, Chne-Wuen Tsai, Kian Ann Ng, Longyang Lin, Yilong Dong, Jiamin Li, Tao Tang, Han Wu, et al. A one-shot learning, online-tuning, closed-loop epilepsy management soc with 0.97 μj/classification and 97.8% vector-based sensitivity. In *2021 Symposium on VLSI Circuits*, pages 1–2, 2021.

[156] Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020.

[157] Bingzhao Zhu, Uisub Shin, and Mahsa Shoaran. Closed-loop neural prostheses with on-chip intelligence: A review and a low-latency machine learning model for brain state detection. *IEEE Transactions on Biomedical Circuits and Systems*, 2021.

[158] Yusuke Iwasawa and Yutaka Matsuo. Test-time classifier adjustment module for model-agnostic domain generalization. *Advances in Neural Information Processing Systems*, 34:2427–2440, 2021.

[159] Qin Wang, Olga Fink, Luc Van Gool, and Dengxin Dai. Continual test-time domain adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7201–7211, 2022.

[160] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, pages 9229–9248, 2020.

[161] Thomas Donoghue, Matar Haller, Erik J Peterson, Paroma Varma, Priyadarshini Sebastian, Richard Gao, Torben Noto, Antonio H Lara, Joni D Wallis, Robert T Knight, et al. Parameterizing neural power spectra into periodic and aperiodic components. *Nature neuroscience*, 23(12):1655–1665, 2020.

[162] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135, 2017.

[163] Ishaan Gulrajani and David Lopez-Paz. In search of lost domain generalization. *arXiv preprint arXiv:2007.01434*, 2020.

[164] Claude Elwood Shannon. A mathematical theory of communication. *ACM SIGMOBILE mobile computing and communications review*, 5(1):3–55, 2001.

[165] Yuxiao Yang, Shaoyu Qiao, Omid G Sani, J Isaac Sedillo, Breonna Ferrentino, Bijan Pesaran, and Maryam M Shanechi. Modelling and prediction of the dynamic responses of large-scale brain networks during direct electrical stimulation. *Nature biomedical engineering*, 5(4):324–345, 2021.

[166] Alexandre Moly, Thomas Costecalde, Félix Martel, Matthieu Martin, Christelle Larzabal, Serpil Karakas, Alexandre Verney, Guillaume Charvet, Stephan Chabardes, Alim Louis Benabid, et al. An adaptive

closed-loop ecog decoder for long-term and stable bimanual control of an exoskeleton by a tetraplegic. *Journal of Neural Engineering*, 19(2):026021, 2022.

[167] Franz Hell, Annika Plate, Jan H Mehrkens, and Kai Bötzel. Subthalamic oscillatory activity and connectivity during gait in parkinson's disease. *NeuroImage: Clinical*, 19:396–405, 2018.

[168] Petra Fischer, Chiung Chu Chen, Ya-Ju Chang, Chien-Hung Yeh, Alek Pogosyan, Damian M Herz, Binith Cheeran, Alexander L Green, Tipu Z Aziz, Jonathan Hyam, et al. Alternating modulation of subthalamic nucleus beta oscillations during stepping. *Journal of Neuroscience*, 38(22):5111–5121, 2018.

[169] Eva L Dyer, Mohammad Gheshlaghi Azar, Matthew G Perich, Hugo L Fernandes, Stephanie Naufel, Lee E Miller, and Konrad P Körding. A cryptography-based approach for movement decoding. *Nature biomedical engineering*, 1(12):967–976, 2017.

[170] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR, 2017.

[171] Fabien Lotte, Laurent Bougrain, Andrzej Cichocki, Maureen Clerc, Marco Congedo, Alain Rakotomamonjy, and Florian Yger. A review of classification algorithms for eeg-based brain–computer interfaces: a 10 year update. *Journal of neural engineering*, 15(3):031005, 2018.

[172] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial

discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7167–7176, 2017.

[173] Trung Le, Tuan Nguyen, Nhat Ho, Hung Bui, and Dinh Phung. Lamda: Label matching deep domain adaptation. In *International Conference on Machine Learning*, pages 6043–6054, 2021.

[174] Marvin Zhang, Henrik Marklund, Abhishek Gupta, Sergey Levine, and Chelsea Finn. Adaptive risk minimization: A meta-learning approach for tackling group shift. *arXiv preprint arXiv:2007.02931*, 8:9, 2020.

[175] Gilles Blanchard, Aniket Anand Deshmukh, Ürun Dogan, Gyemin Lee, and Clayton Scott. Domain generalization by marginal transfer learning. *The Journal of Machine Learning Research*, 22(1):46–100, 2021.

[176] Kaoru Takakusaki. Functional neuroanatomy for posture and gait control. *Journal of movement disorders*, 10(1):1, 2017.

[177] Muhammad Awais Bin Altaf, Judyta Tillak, Yonatan Kifle, and Jerald 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*, pages 100–101. IEEE, 2013.

[178] Jian Shen, Yanru Qu, Weinan Zhang, and Yong Yu. Wasserstein distance guided representation learning for domain adaptation. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[179] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[180] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

[181] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[182] Xiang Zhang, Lina Yao, Manqing Dong, Zhe Liu, Yu Zhang, and Yong Li. Adversarial representation learning for robust patient-independent epileptic seizure detection. *IEEE Journal of Biomedical and Health Informatics*, 2020.

[183] Jianhua Lin. Divergence measures based on the shannon entropy. *IEEE Transactions on Information theory*, 37(1):145–151, 1991.

[184] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[185] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022.

[186] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021.

[187] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[188] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[189] Leo Grinsztajn, Edouard Oyallon, and Gael Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[190] Pieter Gijsbers, Marcos LP Bueno, Stefan Coors, Erin LeDell, Sébastien Poirier, Janek Thomas, Bernd Bischl, and Joaquin Vanschoren. Amlb: an automl benchmark. *arXiv preprint arXiv:2207.12560*, 2022.

[191] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[192] Talip Ucar, Ehsan Hajiramezanali, and Lindsay Edwards. Subtab: Subsetting features of tabular data for self-supervised representation learning. *Advances in Neural Information Processing Systems*, 34:18853–18865, 2021.

[193] Dara Bahri, Heinrich Jiang, Yi Tay, and Donald Metzler. Scarf: Self-supervised contrastive learning using random feature corruption. *arXiv preprint arXiv:2106.15147*, 2021.

[194] Kushal Majmundar, Sachin Goyal, Praneeth Netrapalli, and Prateek Jain.

Met: Masked encoding for tabular data. *arXiv preprint arXiv:2206.08564*, 2022.

[195] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[196] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, 34:18932–18943, 2021.

[197] Gowthami Somepalli, Micah Goldblum, Avi Schwarzschild, C Bayan Bruss, and Tom Goldstein. Saint: Improved neural networks for tabular data via row attention and contrastive pre-training. *arXiv preprint arXiv:2106.01342*, 2021.

[198] Chuhan Wu, Fangzhao Wu, Tao Qi, Yongfeng Huang, and Xing Xie. Fastformer: Additive attention can be all you need. *arXiv preprint arXiv:2108.09084*, 2021.

[199] Zifeng Wang and Jimeng Sun. Transtab: Learning transferable tabular transformers across tables. *arXiv preprint arXiv:2205.09328*, 2022.

[200] Liam Collins, Hamed Hassani, Aryan Mokhtari, and Sanjay Shakkottai. Fedavg with fine tuning: Local updates lead to representation learning. *arXiv preprint arXiv:2205.13692*, 2022.

[201] Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel.

TaBERT: Pretraining for joint understanding of textual and tabular data. *arXiv preprint arXiv:2005.08314*, 2020.

[202] Jinsung Yoon, Yao Zhang, James Jordon, and Mihaela van der Schaar. Vime: Extending the success of self-and semi-supervised learning to tabular domain. *Advances in Neural Information Processing Systems*, 33:11033–11043, 2020.

[203] Ivan Rubachev, Artem Alekberov, Yury Gorishniy, and Artem Babenko. Revisiting pretraining objectives for tabular deep learning. *arXiv preprint arXiv:2207.03208*, 2022.

[204] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[205] Fela Winkelmolen, Nikita Ivkin, H Furkan Bozkurt, and Zohar Karnin. Practical and sample efficient zero-shot hpo. *arXiv preprint arXiv:2007.13382*, 2020.

[206] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second. *arXiv preprint arXiv:2207.01848*, 2022.

[207] Armen Aghajanyan, Anchit Gupta, Akshat Shrivastava, Xilun Chen, Luke Zettlemoyer, and Sonal Gupta. Muppet: Massive multi-task representations with pre-finetuning. *arXiv preprint arXiv:2101.11038*, 2021.

[208] Roman Levin, Valeriia Cherepanova, Avi Schwarzschild, Arpit Bansal, C Bayan Bruss, Tom Goldstein, Andrew Gordon Wilson, and Micah

Goldblum. Transfer learning with deep tabular models. *arXiv preprint arXiv:2206.15306*, 2022.

[209] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[210] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019.

[211] Vitaly Kurin, Alessandro De Palma, Ilya Kostrikov, Shimon Whiteson, and M Pawan Kumar. In defense of the unitary scalarization for deep multi-task learning. *arXiv preprint arXiv:2201.04122*, 2022.

[212] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[213] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[214] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[215] Vinay Venkatesh Ramasesh, Aitor Lewkowycz, and Ethan Dyer. Effect of scale on catastrophic forgetting in neural networks. In *International Conference on Learning Representations*, 2021.

[216] Prakhar Kaushik, Alex Gain, Adam Kortylewski, and Alan Yuille. Understanding catastrophic forgetting and remembering in continual learning with optimal relevance mapping. *arXiv preprint arXiv:2102.11343*, 2021.

[217] Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. Catboost: gradient boosting with categorical features support. *arXiv preprint arXiv:1810.11363*, 2018.

[218] Nick Erickson, Jonas Mueller, Alexander Shirkov, Hang Zhang, Pedro Larroy, Mu Li, and Alexander Smola. Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505*, 2020.

[219] Jeremy Howard and Sylvain Gugger. Fastai: a layered api for deep learning. *Information*, 11(2):108, 2020.

[220] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, pages 23965–23998. PMLR, 2022.