DISSERTATION

CONVOLUTIONAL NEURAL NETWORKS FOR EEG SIGNAL CLASSIFICATION IN ASYNCHRONOUS BRAIN-COMPUTER INTERFACES

Submitted by

Elliott M. Forney

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Fall 2019

Doctoral Committee:

Advisor: Charles Anderson

Asa Ben-Hur
Michael Kirby
Donald Rojas

ABSTRACT

CONVOLUTIONAL NEURAL NETWORKS FOR EEG SIGNAL CLASSIFICATION IN ASYNCHRONOUS BRAIN-COMPUTER INTERFACES

Brain-Computer Interfaces (BCIs) are emerging technologies that enable users to interact with computerized devices using only voluntary changes in their mental state. BCIs have a number of important applications, especially in the development of assistive technologies for people with motor impairments. Asynchronous BCIs are systems that aim to establish smooth, continuous control of devices like mouse cursors, electric wheelchairs and robotic prostheses without requiring the user to interact with time-locked external stimuli.

Scalp-recorded Electroencephalography (EEG) is a noninvasive approach for measuring brain activity that shows considerable potential for use in BCIs. Inferring a user's intent from spontaneously produced EEG signals remains a challenging problem, however, and generally requires specialized machine learning and signal processing methods. Current approaches typically involve guided preprocessing and feature generation procedures used in combination with with carefully regularized, often linear, classification algorithms. The current trend in machine learning, however, is to move away from approaches that rely on feature engineering in favor of multilayer (deep) artificial neural networks that rely on few prior assumptions and are capable of automatically learning hierarchical, multiscale representations.

Along these lines, we propose several variants of the Convolutional Neural Network (CNN) architecture that are specifically designed for classifying EEG signals in asynchronous BCIs. These networks perform convolutions across time with dense connectivity across channels, which allows them to capture spatiotemporal patterns while achieving time invariance. Class labels are assigned using linear readout layers with label aggregation in order to reduce susceptibility to overfitting and to allow for continuous control. We also utilize transfer learning in order to reduce overfitting and leverage patterns that are common across individuals. We show

that these networks are multilayer generalizations of Time-Delay Neural Networks (TDNNs) and that the convolutional units in these networks can be interpreted as learned, multivariate, nonlinear, finite impulse-response filters.

We perform a series of offline experiments using EEG data recorded during four imagined mental tasks: silently count backward from 100 by 3's, imagine making a left-handed fist, visualize a rotating cube and silently sing a favorite song. Data were collected using a portable, eight-channel EEG system from 10 participants with no impairments in a laboratory setting and four participants with motor impairments in their home environments. Experimental results demonstrate that our proposed CNNs consistently outperform baseline classifiers that utilize power-spectral densities. Transfer learning yields an additional performance improvement, but only when used in combination with multilayer networks. Our final test results achieve a mean classification accuracy of 57.86%, which is 8.57% higher than the 49.29% achieved by our baseline classifiers. In terms of information transfer rates, our proposed methods achieve a mean of 15.82 bits-per-minute while our baseline methods achieve 9.35 bits-per-minute. For two individuals, our CNNs achieve a classification accuracy of 90.00%, which is 10–20% higher than our baseline methods. A comparison with external studies suggests that these results are on par with the state-of-the-art, despite our relatively rigorous experimental design.

We also perform a number of experiments that analyze the types of patterns our classifiers learn to utilize. This includes a detailed analysis of aggregate power-spectral densities, examining the layer-wise activations produced by our CNNs, extracting the frequency responses of convolutional layers using Fourier analysis and finding optimized input sequences for trained networks. These analyses highlight several ways that the patterns our methods learn to utilize are related to known patterns that occur in EEG signals while also creating new questions about some types of patterns, including high-frequency information. Examining the behavior of our CNNs also provides insights into the inner workings of these networks and demonstrates that they are, in fact, learning to form hierarchical, multiscale representations of EEG signals.

ACKNOWLEDGEMENTS

I would like to thank Charles Anderson for the teaching, mentorship and guidance that he has provided me over the years and for his feedback, comments and proofreading of this document. Much of the code that I have used in my implementations also originated during his courses in machine learning and from his contributions to the $CEBL_3$ project. I have been hugely impacted by all of the wonderful and inspirational teachers at CSU and I would like to thank you all. I would also like to thank my dissertation committee and all of the members of the BCI laboratory. Bill Gavin, Patti Davies and Marla Roll, in particular, have taught me much of what I know about EEG and how to work with participants in a kind and professional way. Brittany Taylor, Jewel Crasta, Stephanie Scott, Katie Bruegger, Kim Teh and Stephanie Teh have all been especially instrumental in the success of my research and the BCI project and have all been great sources of support and friendship. Tomojit Ghosh and Fereydoon Vafaei have been great friends and colleagues and were extremely helpful throughout the process of formulating my ideas and designing my experiments. I would also like to extend a special thank you to everyone that participated in our studies and, especially, to those who have graciously allowed us to enter their homes. I truly hope that this research helps to develop next-generation assistive technologies for everyone who can benefit from them. This work was supported in part by the National Science Foundation through grant number 1065513 and through generous support from the Department of Computer Science, Department of Occupational Therapy and Department of Human Development and Family Studies at Colorado State University. I have also received generous financial support from the Forney Family Scholarship and through the Computer Science Department's Artificial Intelligence and Evolutionary Computation Fellowship. I would like to thank my father, Glen Forney, for igniting my passion for science and my mother, Nancy Forney, for passing on her passion for reading, writing and critical thinking. Most of all, I would like to thank my wife, Maggie, and my two children, Parker and Alexa, for their phenomenal and enduring support of my academic endeavors.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

Brain-computer Interfaces (BCIs) are systems for establishing a direct channel of communication between the human brain and a computerized device [1–3]. Although there are many potential applications for BCI systems, the development of assistive technologies for people with motor impairments is a goal that appears to be attainable in the near future and one that could offer significant benefits to individuals and society. For people with disabilities, assistive devices that are controlled using BCIs may have the potential to restore their ability to communicate or perform daily tasks. For people with severe impairments, even BCIs with slow communication rates may be invaluable. For those who have lost all or nearly all voluntary motor function, a condition known as locked-in syndrome, BCIs may be their *only* means of communication with the outside world [4–8].

A number of methods for observing changes in brain activity have been explored for use in BCIs, ranging from functional nuclear Magnetic Resonance Imaging (fMRI) to electrode microarrays implanted directly into brain tissue [8–11]. Among these approaches, scalp-recorded Electroencephalography (EEG) is a popular choice. EEG measures changes in electrical potentials at the surface of the scalp caused by the synchronized firing of action potentials in neurons near the cortical surface of the brain [12]. Since EEG is noninvasive, i.e., does not require surgical intervention, it can be safely and easily tested in people with or without motor impairments. Modern EEG equipment is also small, portable and relatively inexpensive, making it suitable for use in a variety of assistive devices. EEG also has a high temporal resolution, on the order of microseconds, which makes it well suited for use in real-time applications.

In Figure 1.1, we see an image of one of the participants in our study, who has quadriplegia, wearing an eight-channel EEG cap while performing BCI experiments in his work environment. Note that a small amount of electrically conductive gel is placed under each electrode in order to reduce the impedance between the sensors and the scalp. In Figure 1.2, we see the layout and

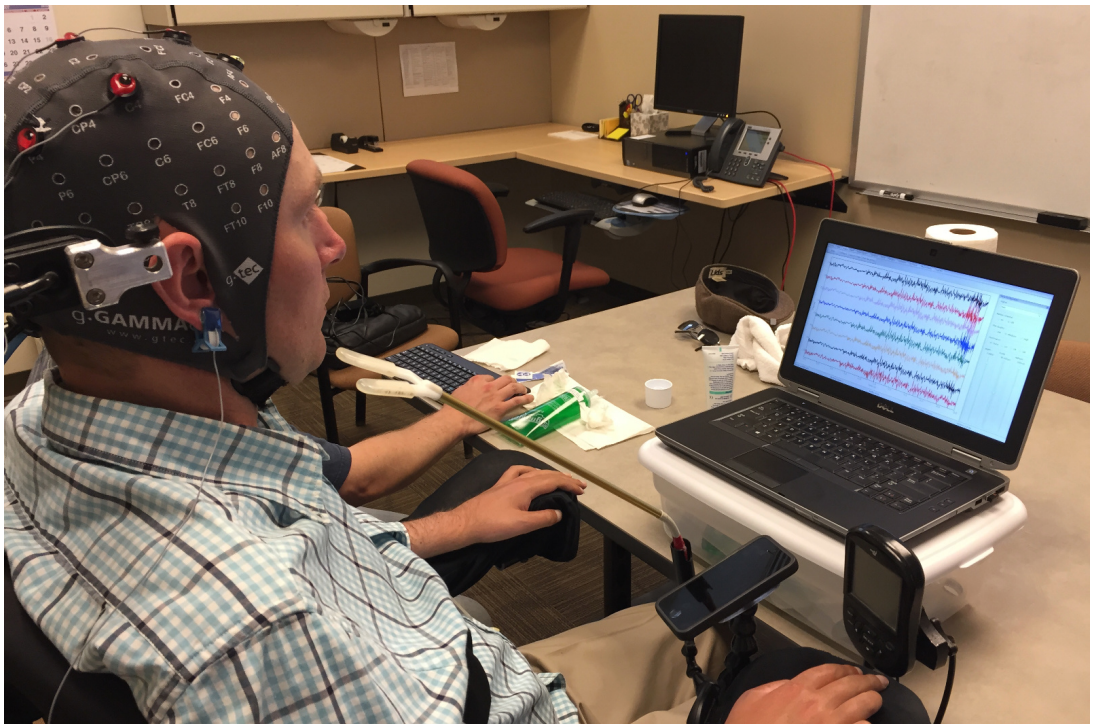


Figure 1.1: A participant with quadriplegia wearing an eight-channel EEG cap while performing BCI experiments in their office.

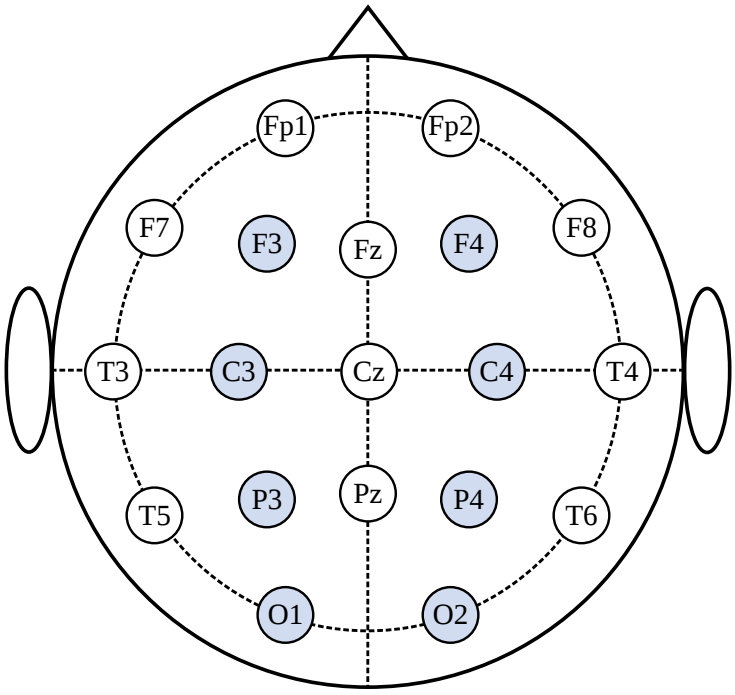


Figure 1.2: The layout and naming convention for EEG electrode placement in the 10/20 standard. Our portable EEG system is configured to use the eight highlighted channels.

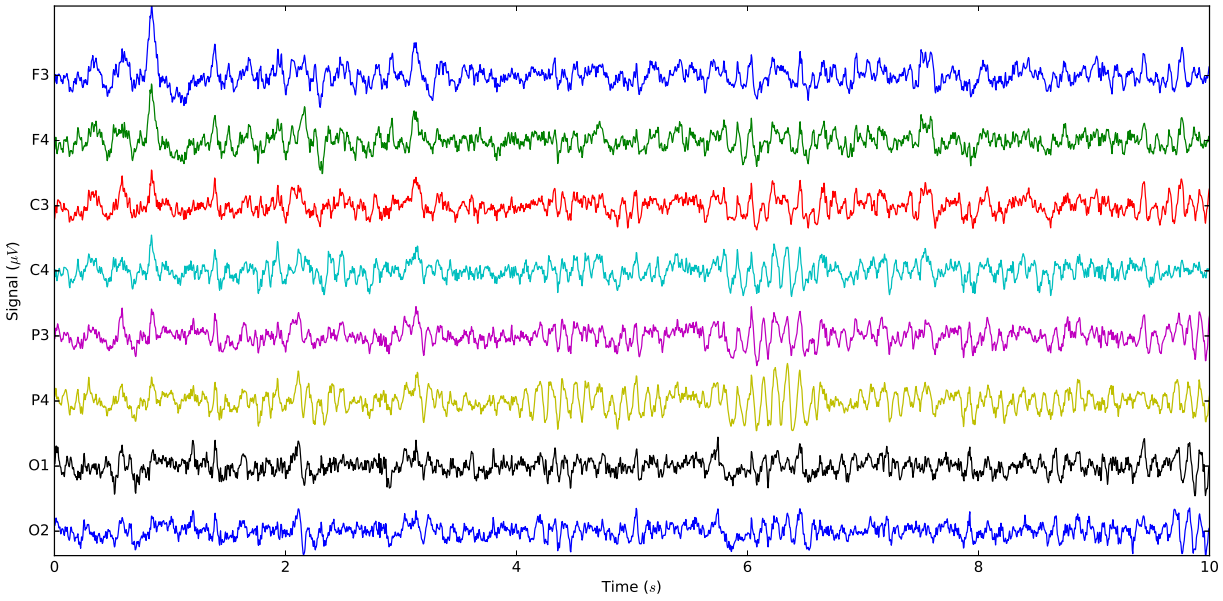


Figure 1.3: A sample trace plot of a 10-second EEG segment. The vertical axis shows signal voltage in μV spread across the eight channels and the horizontal axis shows time.

naming convention for EEG electrode placement according to the 10/20 standard. The first letter of each site name corresponds to a region of the brain: **F**rontal, **C**entral, **P**arietal, **T**emporal and **O**ccipital. The numbers further differentiate the sites and odd numbers are over the left hemisphere of the brain while even numbers are over the right hemisphere. As we will describe in Section 3.5.2, our EEG system is a portable eight-channel system configured to use the eight sites highlighted in Figure 1.2. In Figure 1.3, we see a sample trace plot of a 10-second EEG segment. The vertical axis shows the signal voltage, separated by channel, and the horizontal axis shows time.

Although EEG has a number of properties that make it well suited for use in BCIs, it also involves a number of significant challenges that should be considered. Since EEG signals travel through the various layers of meninges, skull and scalp, they tend to be spatially blurred and are primarily representative of activity near the cortical surface of the brain. This results in a partial and imprecise representation of the underlying neural activity. EEG signals also have small magnitudes, on the order of microvolts, and are frequently contaminated with noise from external sources, such as computer peripherals, household appliances and power mains. It is

also common for EEG signals to contain artifacts from biological sources, such as ocular movements, muscle contractions and cardiac rhythms. Noise and artifacts appear to be especially prominent in real-world environments, where BCI systems are likely to be used [13]. The sheer complexity of the human brain also suggests that EEG signals likely contain sophisticated spatiotemporal patterns, a problem that is confounded by the fact that the brain is continually performing multiple tasks in parallel.

Despite these challenges, a number of research groups have successfully demonstrated prototype EEG-based BCI systems. Several software packages are now available for performing BCI experiments [14–17] and several companies have begun to offer commercial BCI products [18, 19]. In recent years, several people who are nearly locked-in due to advanced Amyotrophic Lateral Sclerosis (ALS) have even begun using BCI systems for communication on a daily basis [7, 20, 21]. Some research groups and clinics have also begun using BCI technology for stroke rehabilitation [22] and assessment of consciousness [19, 23].

Although current BCI systems have clearly achieved a level of success, they often depend on changes in EEG signals that can be reliably evoked by external stimuli. The P300 speller paradigm is a notable example of a BCI design that leverages external stimuli [24, 25]. In a P300 speller, the BCI user attends to a letter that they wish to type among a grid of letters displayed on a computer monitor. The rows and columns of the grid are then flashed in a random order. When the letter that the user is attending to is flashed, a predictable waveform, known as an Event-Related Potential (ERP), is produced in the user's EEG signals. Although ERPs are difficult to detect using a single trial, multiple repetitions of flashing can be used to deduce the row and column of the letter that the user wishes to type.

BCI systems that leverage ERPs tend to be reliable; however, they also have a number of disadvantages that may hinder usability and performance. For instance, the use of external stimuli can limit the user's ability to attend to the task at hand and may result in fatigue after prolonged use. Additionally, some potential BCI users may have an impaired ability to attend to external stimuli [26]. The communication rate of ERP-based paradigms is also inherently limited by the

rate at which the stimuli can be identified and by the time required for the progression of the ERP. Finally, these types of BCI systems usually operate in a synchronous, time-locked fashion that is not well suited for some types of control tasks. For example, moving a mouse cursor, steering an electric wheelchair and operating a robotic prosthesis are all instances of tasks that are difficult to achieve using synchronous BCIs.

Asynchronous BCI systems that do not rely on time-locked external stimuli have also been proposed. BCIs that follow the Motor Imagery (MI) paradigm associate two or more imagined motor movements with commands that the BCI system might perform [27–30]. For example, a user might imagine moving their left hand to move a mouse cursor to the left or imagine moving their right hand to move the cursor to the right. These types of BCIs often leverage characteristic patterns that occur in EEG signals during motor movements, regardless of whether the movements are real or imagined. These patterns take the form of an increase or decrease in amplitude in μ and β rhythms, which roughly occur in frequencies between 8–14Hz and 14–30Hz respectively, over the centro-parietal regions of the hemisphere of the brain that is contralateral to the motor movement. Although it has been shown that some people with motor impairments can control MI-based BCIs [31,32], these systems typically have low communication rates and often require extensive training. Furthermore, the fact that these systems rely on lateralized changes in the EEG signals with similar frequency ranges and over relatively small regions of the cortex, suggests that it may be difficult to differentiate between more than a few imagined motor movements.

The Mental Task (MT) paradigm is an approach that generalizes MI to include various other cognitive tasks [33–35]. For instance, a user might silently sing a song to move a mouse cursor up, perform arithmetic to move the cursor down, imagine moving their left arm to move the cursor left and imagine moving their right arm to move it to the right. Over time and with practice and feedback, it is hoped that performing these tasks may become second-nature. The MT paradigm was inspired by a number of research projects in cognitive psychology that observed various changes in the power spectra of EEG signals that occur when a user performs differ-

ent mental tasks [33, 34]. Similar to the MI paradigm, the MT paradigm allows self-paced and stimulus-free control. MT-based approaches may also be more appropriate for differentiating between many distinct mental states, provided that the mental tasks are selected in a way that elicits a variety of responses from different regions of the brain [36]. In other words, the wide variety of patterns produced by different mental tasks may allow more degrees of control. Additionally, it has been suggested that performing tasks that are not related to imagined motor movements may be more appropriate for people who are paralyzed and unable to perform the corresponding physical movements [37].

For these reasons, we believe that asynchronous BCIs that follow the MT communication paradigm show considerable potential. Accordingly, these types of BCIs will be the focus of the present work. Developing MT-based BCIs is a difficult task, however, because they lack the type of clearly defined control signal that is typically found in other BCI systems. Furthermore, EEG signals appear to be highly variable across individuals and over the course of time. As a result, robust machine learning algorithms are required in order to infer the user's intent by identifying relevant patterns in their EEG signals. Improving the performance of these machine learning methods is an important step toward creating practical MT-based BCI systems.

1.1 Challenges

Classifying EEG signals in asynchronous BCIs is a challenging problem that requires specialized methods. In order to precisely characterize the limitations of current approaches and to guide us toward improved solutions, we have identified seven primary challenges that a successful classification system should address.

Challenge 1: Time invariance.

Since asynchronous BCI paradigms are self-paced, i.e., they are not time-locked to external stimuli, the classifier must be able to identify the relevant patterns in the signal regardless of

starting time. In order to account for this, an appropriate classifier should tolerate arbitrary shifts in time, i.e., it should be time invariant.

Challenge 2: Nonlinear and nonstationary spatiotemporal patterns.

A successful classifier should be capable of learning sophisticated spatiotemporal patterns. In this context, a pattern is said to be spatial if it exists across multiple EEG electrodes and temporal if it unfolds over the course of time. Although the exact types of patterns that are relevant for this classification task have not yet been fully characterized, EEG signals have been shown to have transient, nonstationary and nonlinear properties [38, 39].

Challenge 3: Patterns at multiple time scales.

Patterns in EEG signals are known to occur at multiple time scales. For example, slow cortical potentials occur below 1Hz while α-rhythms occur between 8–16Hz . Although multiscale interactions are not yet fully understood, we suspect that an effective classifier should be able to identify patterns that occur across multiple time scales.

Challenge 4: Noise and artifacts.

As we have previously discussed, EEG signals are susceptible to various types of noise and artifacts. A successful classifier should be robust to these types of interference. This is especially true if the BCI system is to be usable outside of a controlled environment.

Challenge 5: Undersampling and high dimensionality.

Only a relatively small amount of data can be reasonably collected during a BCI calibration phase. If a long calibration procedure is required, the user may become fatigued or frustrated with the usability of the BCI system. This problem is exasperated by the relatively high dimensionality of EEG data. For instance, an eight-channel EEG signal sampled at 256Hz results in 2,048 observations per second. A suitable classifier should be able to perform well given this rel-

atively high dimensionality and while using only about 10–15 minutes of subject-specific EEG data for calibrating the system. Note, however, that it may be feasible to utilize information that is not subject specific by incorporating prerecorded EEG signals.

Challenge 6: Interpretation.

Identifying the types of patterns that a classifier learns and interpreting their significance is important for several reasons. First, the ability to interpret the results of a classifier can lead to improvements in BCI configuration and design. For instance, if it is noted that two mental tasks produce similar brain activity, it may be beneficial to replace one of the tasks with another that produces a more distinct response. Similarly, interpretation may lead to improvements in electrode placement or signal preprocessing methods. The ability to analyze the patterns learned by the classifier may also lead to new insights into human cognition, neuroscience and electrophysiology. The process of engineering BCIs is inextricably linked to the science of the human brain.

Challenge 7: Real-time performance.

In order for a classifier to be practical for use in BCI systems, it must be possible to use the classifier interactively. The classifier should train in a matter of minutes and it should be possible to evaluate the classifier in less than one second. Furthermore, any tuning or hyperparameter selection that may be required on a session-to-session basis should be easily performed automatically or by a non-expert in only a few minutes.

1.2 Problem Statement

Unfortunately, current approaches for classifying EEG signals in asynchronous BCIs, which are discussed in detail in Chapter 2, have not achieved the levels of performance that are required by many practical applications and have largely been confined to controlled laboratory environments. We assert that these approaches do not fully address all of the challenges de-

scribed in the previous section and that new methods are required in order to improve the reliability and communication rates of MT-based BCIs.

Many current approaches focus on being easily interpreted and capturing patterns that are known or assumed to exist in EEG signals. For example, analyzing the amplitude or power content of EEG signals across a range of frequencies bands is relatively straightforward to interpret and many of these frequencies are known to vary across mental tasks. Although these approaches have been used with a degree of success in the past, we will show that they have a limited ability to capture several types of patterns, including spatial and nonstationary information, that may be important for classifying EEG signals.

Current approaches also rely heavily on filtering, feature selection and dimensionality reduction in order to address noise and undersampling. These procedures generally rely on strict prior assumptions about the nature of the patterns found in EEG signals and can lead to the loss of useful information. Furthermore, establishing these procedures can be time consuming, involve extensive manual intervention and it is often unclear how well these procedures generalize across subjects, sessions and environments.

The use of simple linear models is also quite prevalent in current approaches. Although linear models often appear to work well, we suspect that the success of these approaches largely results from their robustness to overfitting and that information loss during preprocessing may prevent nonlinear methods from leveraging more sophisticated patterns. Evidence of nonlinear dynamics in EEG signals recorded during various mental tasks supports the notion that nonlinear patterns may be valuable for EEG signal classification [39, 40].

Given the complexity of the human brain, we believe that EEG signals likely contain a variety of sophisticated patterns that have not yet been identified. Approaches that discard information and rely on prior assumptions about the data may have a limited ability to identify new types of patterns. As researchers, the use of algorithms that rely on our prior knowledge may also limit our insights while serving to reinforce our previous conceptions about the types of patterns contained in EEG signals.

There are many questions about the nature of EEG signals that remain to be answered and further research is clearly required. In order to develop the next generation of EEG classification and analysis algorithms, we believe that constraints and prior assumptions should be loosened. Instead, we should explore general methods that are capable of automatically filtering, identifying and exploiting a wide variety of patterns that may be found in EEG signals.

1.3 Proposed Solution

The recent trend in machine learning has been to move away from models that involve large amounts of manual engineering in favor of multilayer artificial neural networks that are capable of automatically learning hierarchical, multiscale representations. These approaches, known as deep networks, have enjoyed considerable success on a number of challenging problems [41].

Of particular interest are a class of deep networks known as Convolutional Neural Networks (CNNs) [42, 43]. CNNs have gained considerable traction in the computer vision community and are able to learn multiscale representations of images that generalize well and are robust to shifts and other types of deformations [44–49]. One of the principal advantages of CNNs is their ability to identify patterns and learn effective representations without requiring handcrafted solutions for preprocessing, filtering, feature selection and dimensionality reduction, which can all lead to information loss.

Following these principals, we will introduce several variants of the CNN architecture that are designed to address the challenges described in Section 1.1 while relying on few prior assumptions and avoiding procedures that discard potentially useful information. These networks leverage convolution across time in order to achieve time invariance. Since weights are shared across time, they generally have fewer parameters to tune than a fully connected network, which helps to address problems with noise and undersampling. Nonlinear spatiotemporal information can be captured by extending the convolutional window across channels and through the use of nonlinear transfer functions. Each artificial neuron in our CNNs can be interpreted as a nonlinear, multivariate, finite impulse-response filter. Since the parameters of

these filters are found automatically, they rely on few prior assumptions, and the only manual tuning required is the selection of their width and cardinality. The characteristics of these filters can be explored using techniques from time and frequency domain analysis and by examining the learned weights. The layers of these networks can then be stacked and the output of each layer downsampled in order to encourage the network to learn patterns at multiple time scales and different levels of abstraction.

In the standard CNN design, a series of stacked convolutional layers is followed by one or more fully connected layers that assign final class labels. In this setting, however, fully connected layers are prone to overfitting because of the large number of parameters that they require and due to the small amount of data we have available. Fully connected layers are also not well suited for achieving smooth and continuous control because they produce a single class label for fixed-length segments of the signal. As an alternative, we explore the use of sliding linear softmax layers, with a single time-step width, combined with a label aggregation strategy. This approach reduces the number of free parameters while also providing a great deal of flexibility when designing BCI applications.

In order to further mitigate problems with noise, undersampling and overfitting, utilize a transfer learning approach where the networks are first trained to classify data recorded from other participants and subsequently fine-tuned on an individual basis. This allows the networks to learn features and signal representations that are common across mental tasks or are generally useful for processing EEG signals. The types of patterns that these network learn to utilize can be explored by using optimization techniques to learn optimal input sequences and then examining the types of patterns that the network expects to perform well.

Several research groups have recently proposed methods for using CNNs to classify EEG signals in BCIs [50–53], which we will further review in Section 2.2.3. This research is in its infancy, however, and there are a number of questions that remain to be answered and problems that have yet to be solved. The primary goal of the present work is to establish an effective and generic strategy for using CNNs to classify asynchronous EEG signals while relying on few prior

assumptions. The CNNs that we will introduce incorporate a number of innovations that enable them to classify a broad set of mental tasks using only minimally preprocessed EEG signals, despite the various challenges that we have described. We will also provide a detailed analysis of how EEG signals tend to vary across individuals and mental tasks and lay a new foundation for interpreting the types of patterns that our proposed CNN learn to utilize.

1.4 Outline of Remaining Chapters

In Chapter 2, we will begin with a brief review of current methods for classifying EEG signals in MT-based BCIs and we will discuss the advantages and disadvantages of each approach. This investigation will demonstrate that current methods often rely on prior assumptions and manual feature engineering and have a limited ability to capture some important types of patterns. We will then offer a brief introduction to deep learning and CNNs and we will describe how these approaches relate to biological neural networks and how they have been successfully applied to problems in computer vision, BCIs and other domains. We will then assert that the key tenants of these approaches, including hierarchical and multiscale representations, local receptive fields and few prior assumptions, are potentially valuable for classifying EEG signals.

In Chapter 3, we will provide complete descriptions of all of our methods, including several baseline classifiers and proposed network architectures. These methods will range from implementations of current approaches to variants of multilayer CNNs with transfer learning and modified readout layers. We will present formal, mathematical descriptions for each approach as well as analytical and experimental methods for interpretation and analysis. We will also provide descriptive insights into the reasoning behind each of these methods and we will discuss how these networks are specifically designed to address the challenges described in Section 1.1. We will also describe the participants and procedures used to collect the dataset that will be used in subsequent experiments, which utilizes a portable EEG system and includes data recorded from participants with motor impairments in their home environments.

In Chapter 4, we will perform an in-depth analysis of the various design choices and hyper-parameters associated with each of our classification approaches. These analyses will show how various components of our proposed networks, such as label aggregation, multiple layers, pooling and transfer learning, yield incremental improvements in performance. We will then present our final test performance results, which will demonstrate that our proposed CNNs achieve a valuable improvement in performance over our baseline methods, especially when multiple layers and transfer learning are utilized in combination. We will also examine the performance of each individual mental task and their confusion matrices, which will lead to several insights into how mental tasks should be selected in practical BCIs. We will then perform a cursory comparison of our methods with external studies in order to demonstrate that we are competitive with the state-of-the-art and we will closely examine the tradeoffs between the responsiveness and accuracy of our proposed and baseline methods.

In Chapter 4, we will also perform a number of experiments that are designed to analyze the types of patterns found in these EEG signals and explore how our baseline methods and proposed CNNs learn to leverage these patterns. These experiments will provide a number of insights into how each of our methods functions and will demonstrate that our proposed CNNs do, in fact, learn to form multiscale hierarchical representations of EEG signals. We will also examine how some of the patterns found in our EEG signals align with our current understanding of EEG signals and the human brain and we will also discuss the importance of noise, artifacts, attention and high-frequency information.

In Chapter 5, we will provide a summary of this work along with discussion about our key results and findings. Notably, we will assert that there is strong evidence for using multilayer CNNs for classifying asynchronous EEG signals, provided that several key architecture principals are incorporated into these networks, which include small networks with careful regularization, single axis convolution, transfer learning and label aggregation readout layers. Finally, we will provide some discussion around questions that remain open and potential future directions for expanding upon this research.

Chapter 2

Background

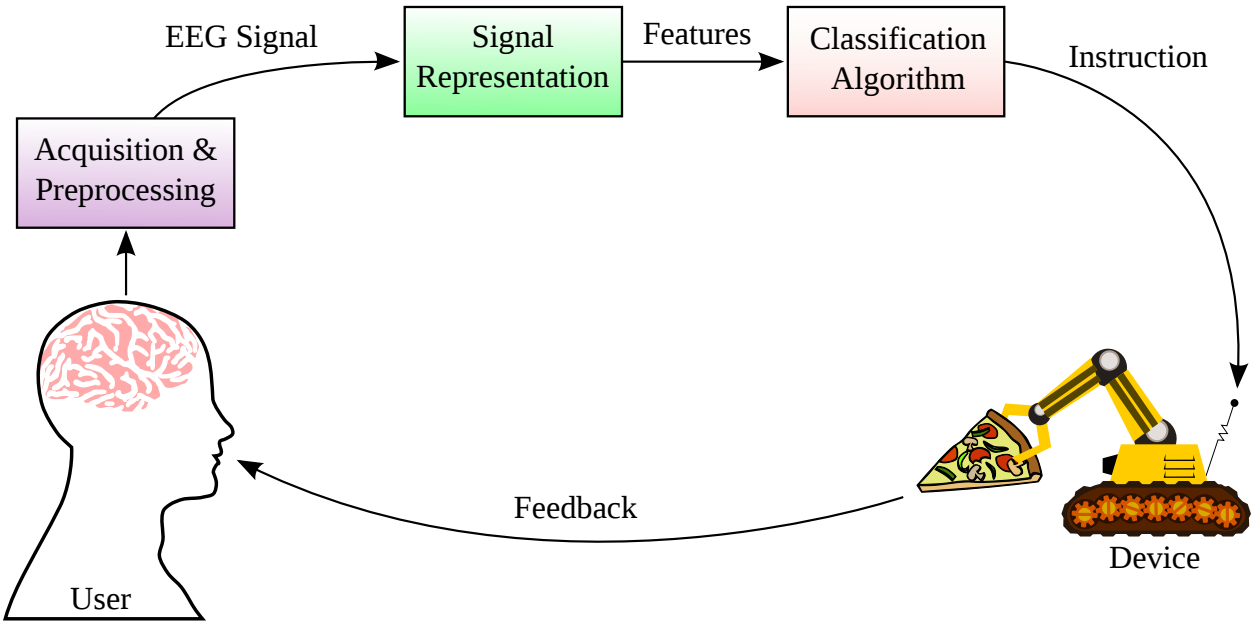


Figure 2.1: The basic components and flow of information in a generic BCI system. The signal is first acquired from the user, converted to a digital signal and some preprocessing is typically performed. The signal is then converted to a representation that is useful for capturing the desired types of patterns. Features are then extracted from the representation and passed to a classifier that attempts to identify the user's mental state. The label from the classifier can be used to send an appropriate instruction to the device being controlled. Finally, the user receives feedback about the action that the system has performed.[1]

In general, a BCI system consists of five components arranged to communicate in a feedback loop, depicted in Figure 2.1. First, the user alters their mental state in a way that conveys their intent according to an established communication protocol. In this setting, the communication protocol follows the MT paradigm, where a number of mental tasks are associated with instructions to the BCI. Next, the EEG acquisition system monitors changes in the user's brain activity and outputs a multivariate EEG signal as a function of time versus voltage for each electrode site, known as a channel. This step typically includes analog-to-digital conversion along

[1]Portions of this image were adapted from free artwork found on http://www.openclipart.org

with some preprocessing, e.g., referencing and bandpass filtering. Next, the EEG signal is transformed into a useful representation, typically as either a function of time (time domain) or else as a function of its frequency content (frequency domain) or some combination of both. This step also includes any automatic or manual feature selection procedures. The resulting features are then passed along to a classifier which, in turn, attempts to identify the mental state of the user. The label assigned by the classifier is then associated with an instruction that is sent to the device that the user wishes to control. Finally, the loop is closed as the user receives feedback from the controlled device. In a stimulus-free BCI system, as we are interested in here, this feedback is typically an observation of the action taken by the device to be controlled or an indicator on a computer console. Among these stages, we are primarily interested in preprocessing, representation and classification; although, the other components will also be important to consider in some contexts.

In this chapter, we will first review several state-of-the-art approaches for representing and classifying asynchronous EEG signals. We will also discuss the advantages and potential limitations of each approach in the context of the challenges we have defined in Section 1.1. We will then offer a brief introduction to deep learning along with some discussion about why these types of models are often successful. We will then introduce CNNs within the important context of computer vision followed by a review of previous work using CNNs to classify EEG signals.

2.1 Current Approaches

A number of approaches have been proposed for classifying EEG signals in MT-based BCIs [33, 34, 36, 54–66]. These approaches typically involve one of several types of signal representations that may be combined with a number of different classification algorithms. It is worth noting that simple linear classifiers are prevalent and often appear to work well [67, 68], despite the fact that EEG signals are known to exhibit nonlinear and nonstationary behavior. In this section, we will closely examine the most common methods for representing EEG signals in MT-based BCIs, including power spectral densities and common spatial patterns. For each of

these signal representations, we will provide concrete examples of patterns that cannot be adequately captured. This analysis will demonstrate the limitations of current approaches and will provide insights into why linear classifiers often perform best when combined with these signal representations. We will then describe the more generic time-delay embedding approach, which will provide a direct transition into our discussion of deep learning.

2.1.1 Power Spectral Densities

Power Spectral Densities (PSDs) are a popular choice for representing EEG signals in MT-based BCIs and were one of the first approaches proposed in the seminal work by Keirn and Aunon [33, 34, 54, 67, 69, 70]. PSDs represent a signal as power density ($\mu V^2/Hz$) across a spectrum of frequencies. The PSD of an EEG segment is estimated separately for each channel, which can be viewed as a univariate time series,

$$\mathbf{s} = s_1, s_2, \ldots, s_T, \tag{2.1}$$

where T is the number of timesteps in the signal segment and s_t is the signal voltage in μV at time t. The Discrete Fourier Transform (DFT) is then used to represent $\mathbf{s}$ as a sum of sine and cosine waves in the complex domain.[2] The DFT is defined as

$$\mathscr{F}_k\{\mathbf{s}\} = \sum_{t=1}^{T} s_t \cdot e^{-i2\pi kt/T} \tag{2.2}$$

$$= \sum_{t=1}^{T} s_t \cdot [\cos(2\pi kt/T) - i \cdot \sin(2\pi kt/T)], \tag{2.3}$$

for $k = 1, \ldots, T$. Note that $\mathscr{F}\{\mathbf{s}\}$ has the same length as $\mathbf{s}$ and is said to be a frequency-domain representation. For real-valued signals, as is the case here, $\mathscr{F}\{\mathbf{s}\}$ is symmetric about $\frac{T}{2}$ and the values $\mathscr{F}_f\{\mathbf{s}\}$ for $f = 1, \ldots, \frac{T}{2}$ correspond to equally spaced frequency components between

[2] In practice, the efficient Fast Fourier Transform (FFT) algorithm is used to compute the DFT.

zero Hertz, also called direct current (DC), and the Nyquist rate of the signal, which is $\frac{1}{2}$ of the sampling rate, R.

The phase offset of each frequency component is encoded as the complex argument, $\arg\left(\mathscr{F}_f\{\mathbf{s}\}\right)$, and the amplitude of each frequency component is encoded as the complex modulus, $|\mathscr{F}_f\{\mathbf{s}\}|$. The power of each component is proportional to the square of the amplitude and, in order to achieve units that account for the symmetry of the DFT and that can be compared across segments with different lengths and sampling rates [71], we scale the powers so that

$$PSD_f\{\mathbf{s}\} = \frac{2|\mathscr{F}_f\{\mathbf{s}\}|^2}{RT^2} \quad \text{for} \quad f = 1, 2, \ldots, \frac{T}{2}. \tag{2.4}$$

Since PSDs do not incorporate phase information, they are inherently time invariant. In other words, PSDs are insensitive to shifts in time because they only utilize the amplitudes of the sine waves that are used to fit the signal and not the phase offsets. In Figure 2.2, we see a PSD plot of the same 8-channel, 10-second EEG segment that we previously illustrated as a trace plot in Figure 1.3.

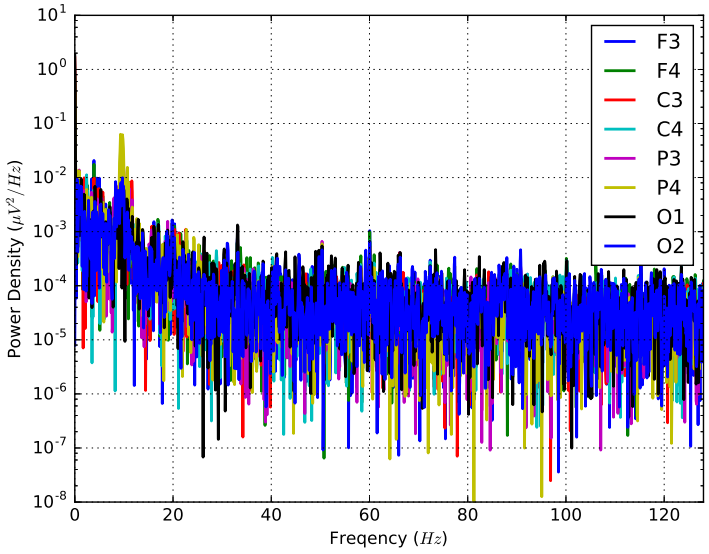


Figure 2.2: A sample PSD plot of the same 10-second EEG segment shown in Figure 1.3. The vertical axis is an estimate of the signal's power across the frequency spectrum on the horizontal axis.

In order to perform classification, the PSD is computed for each EEG channel separately and these features are then concatenated into a single vector that can then be passed to a number of generic classification algorithms. In Section 3.1, we will present several, relatively standard, algorithms for filling this role.

Since EEG signals tend to be noisy, the resulting PSDs typically have a large amount of variability between segments and frequency components. This is often true even for adjacent frequency components, as can be seen in Figure 2.2. Also, recall that the number of frequency components in a PSD is $\frac{1}{2}$ the number of timesteps in the original signal, meaning that a plain PSD provides a modest 2:1 dimensionality reduction.

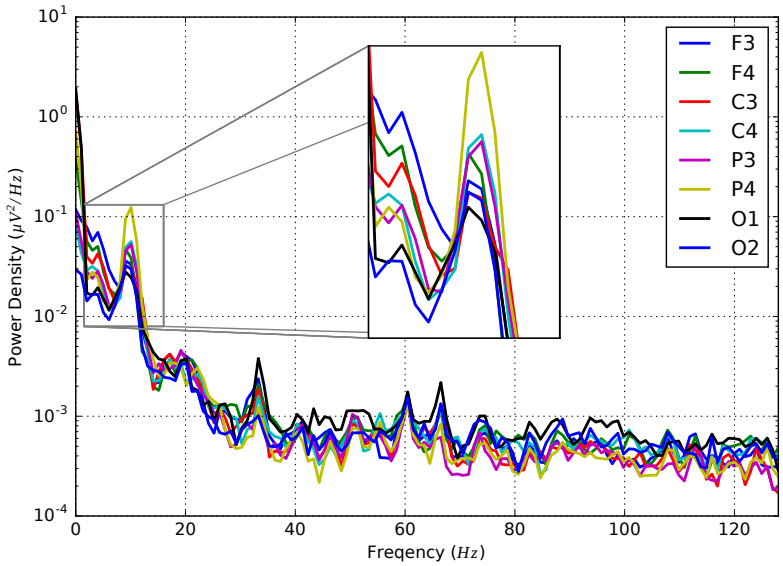


Figure 2.3: A sample Welch PSD plot of the same 10-second EEG segment shown in Figure 1.3. Note that Welch's method results in fewer frequency components with less noise. The inset axis highlights the subtle differences in power content across channels in the 2–16Hz frequency range.

Welch's method is a common technique for both reducing the noise in a PSD and also further reducing its dimensionality [71]. As depicted in Figure 2.4, Welch's method works by splitting the original signal into a number of overlapping segments that are then multiplied by a window function. In our experiments, we refer to the width of the window as the span, ω, and always use a 50% overlap and a Hann window function. Note that the window function reduces the

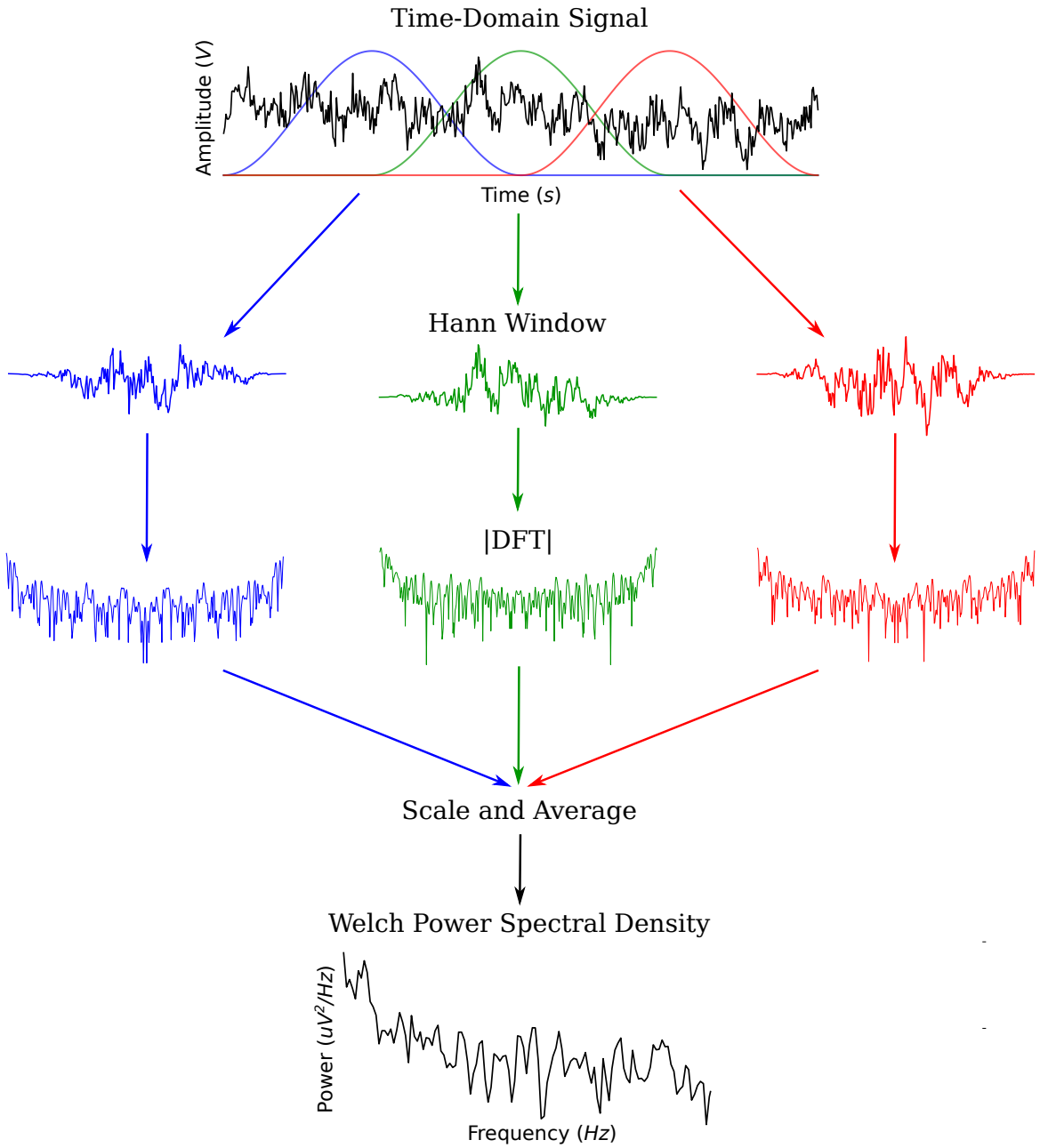
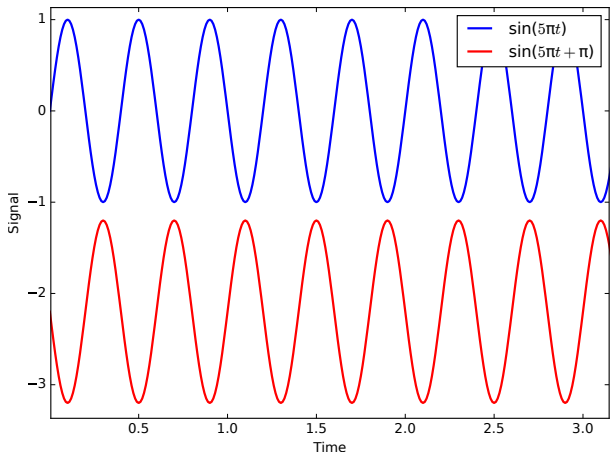


Figure 2.4: A schematic of Welch's method for estimating PSDs. First, the EEG segment is split into Hann windows with 50% overlap. The magnitude of the DFT is then computed for each window. Finally, the DFTs are scaled and averaged together. This yields an estimate of the power of the signal across the frequency spectrum. The resulting PSD has lower frequency resolution but is smoother and has reduced dimensionality.

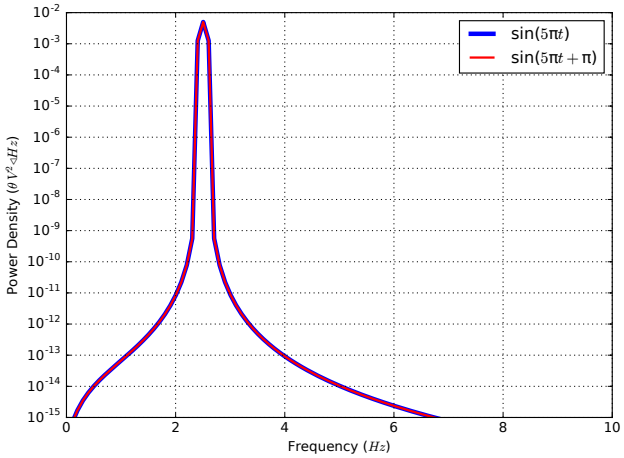
effect of overlapping so that each window has a primary focus around the center. The PSD is then computed for each individual window and, finally, the windows are averaged in order to produce the Welch PSD. Since each window has fewer timesteps than the original signal, the Welch PSD has fewer frequency components, which results in additional dimensionality reduction. Since the windows are averaged, the Welch PSD also tends to be smoother and have less variability, which is a form of noise reduction. In Figure 2.3, we again see a PSD plot of the same EEG segment, but now using Welch's method with $\omega = 1$ second. Note that the PSD is now much smoother and contains fewer frequency components.

PSDs address most of the challenges stated in Section 1.1. They are able to achieve time invariance by omitting phase information, mitigate noise and undersampling through Welch's method, they can be easily interpreted and they can be used in real time. There are, however, several types of patterns that PSDs are not capable of representing or that may lead to counter-intuitive results [72]. For instance, PSDs are not able to capture differences in phase that occur across EEG channels. This is illustrated in Figure 2.5 using two sine waves that have identical frequencies but with phase offsets that differ by π radians. The fact that the resulting PSDs are identical demonstrates how PSDs cannot capture differences in relative phase in a multivariate signal. Although the inclusion of phase synchronization measures have been explored for use in MT-based BCIs [73], estimating these metrics across various channels and frequency bands can lead to a combinatorial explosion of features.

PSDs are also not capable of modeling the ordering of events that occur within a given signal segment. In order to illustrate this, consider Figure 2.7, which shows two sinusoidal chirps that sweep linearly from low-to-high and high-to-low frequencies, respectively. Although the ordering of events within these signal segments is clearly different, the corresponding PSDs are identical. This effect can be attributed to the fact that the DFT fits the signals using only periodic sinusoids and because the phase offsets of these sinusoids are discarded when constructing PSDs. This example also highlights why PSDs are poorly suited for modeling patterns that are nonstationary, i.e., where the characteristics or distribution of the signal vary over time.

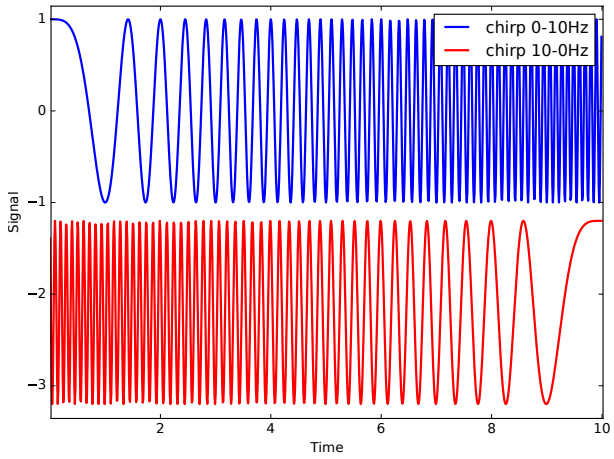


(a) Trace plots of two sinusoids with different phase offsets.

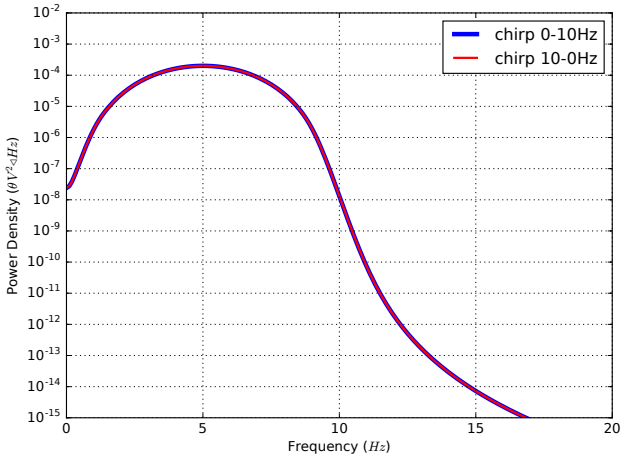


(b) PSD plots of the same two signals.

Figure 2.5: A demonstration of the inability of PSDs to capture differences in phase across channels. (a) Trace plots of two sinusoids with identical frequencies but different phase offsets. (b) PSD plots of the same two signals. Although the signals have different phase offsets, their PSDs are identical.

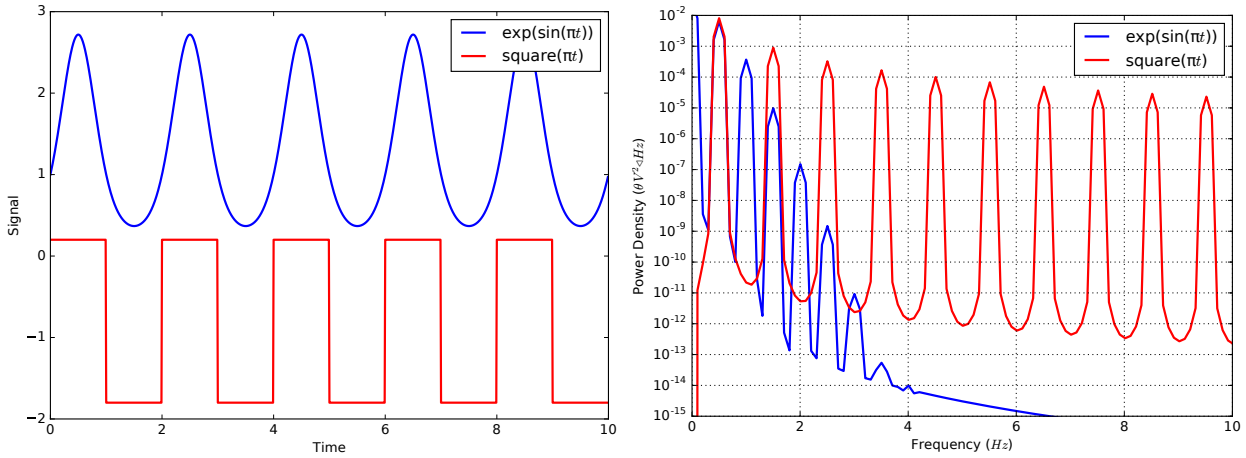


(a) Trace plots of two chirps sweeping from low-to-high and high-to-low frequencies.



(b) PSD plots of the same two signals.

Figure 2.6: A demonstration of the limited ability of PSDs to capture the ordering of events and model nonstationary patterns. (a) Trace plots of two chirps sweeping from 0–10Hz and 10–0Hz respectively. (b) PSD plots of the same two signals. Despite the fact that the ordering of events within these signal segments is different, the PSDs of both signals are the same.



(a) Trace plots of two signals that are periodic but not pure sinusoids.

(b) PSD plots of the same two signals.

Figure 2.7: A demonstration of how signals that are not pure sinusoids can yield PSDs that are difficult to interpret. (a) Trace plots of two signals that are periodic but not sinusoidal. (b) PSD plots of the same two signals. Although the DFT is able to fit these signals, it does so using a number sine waves, or harmonics, with different frequencies. This results in a number of peaks in the PSD that are not representative of the true period of the signals.

Although it is possible to partially counteract these limitations by computing a series of PSDs over relatively short segments, this approach would reduce frequency resolution and place an additional burden on the subsequent classification stage to maintain time invariance while also modeling temporal information across multiple PSDs.

Some types of signals can also lead to counter-intuitive results when interpreting PSDs. Although the DFT is capable of forming a precise and invertible representation of *any* discrete signal segment, it is possible for the resulting PSDs to yield an incomplete or misleading characterization of the signal. For instance, Figure 2.7 shows two signals that both oscillate with a period of two but are generally not considered to be sinusoids. The corresponding PSDs show a number of peeks at various frequencies that represent the sine waves used to fit these signals. Despite the fact that these PSDs are technically accurate, it may be challenging for an observing researcher or classification algorithm to establish that these peaks result from a single waveform rather than distinct sinusoidal components that are superimposed in the same channel. In other words, it can be difficult to distinguish between waveforms that are not pure sinusoids from a mixture of sinusoidal signals that are generated by different sources within the brain.

PSDs are clearly not well suited for modeling some types of patterns, including multivariate phase differences, the ordering of events within segments and patterns that are nonstationary or not pure sinusoids. If these types of patterns are present in EEG signals, then they may be overlooked or mischaracterized by models that rely on PSDs. Nevertheless, PSDs are commonly used in EEG signal analysis and are generally considered a state-of-the-art approach in asynchronous BCIs. Since PSD-based approaches have enjoyed considerable success in the past, we will closely examine these methods in Section 3.1 and use them to establish a carefully tuned baseline classifier. We will then use this baseline classifier as a benchmark for examining the performance of our proposed CNNs in Chapter 4.

2.1.2 Common Spatial Patterns

Common Spatial Patterns (CSP) is another popular method for generating features in asynchronous BCIs. Although CSP was originally proposed for use in MI-based BCIs [30, 74–76], it has also been explored for use in more general MT-based paradigms [36, 56]. CSP seeks to find a linear transformation matrix that projects EEG segments onto a new orthogonal basis that separates the spatial variance, i.e., across channels, so that early components capture more variance for one class and later components capture more variance for another class. The variances of a subset of these components can then be passed to a subsequent classification stage.

In order to describe this process more concretely,[3] assume that we have two EEG segments, $\mathbf{S}_a$ and $\mathbf{S}_b$, which are training examples from two different classes. Both of these segments can be described as matrices,

$$\mathbf{S} = \begin{pmatrix} s_{1,1} & s_{1,2} & s_{1,3} & \dots & s_{1,N} \\ s_{2,1} & s_{2,2} & s_{2,3} & \dots & s_{2,N} \\ s_{3,1} & s_{3,2} & s_{3,3} & \dots & s_{3,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s_{T,1} & s_{T,2} & s_{T,3} & \dots & s_{T,N} \end{pmatrix}, \tag{2.5}$$

[3] Although the derivation of CSP presented here is commonly found in the literature, Knight offers an alternative presentation as a generalized singular value problem [77].

where T is the number of timesteps in the segment and N is the number of channels. Assuming that our EEG segments have been mean-centered, i.e., each column has been shifted to have zero mean, then we can express the covariance matrices as

$$\mathbf{C}_a = \mathbf{S}_a^\top \mathbf{S}_a \tag{2.6}$$

$$\mathbf{C}_b = \mathbf{S}_b^\top \mathbf{S}_b \tag{2.7}$$

and we can express the composite covariance matrix for both classes as

$$\mathbf{C}_c = \mathbf{C}_a + \mathbf{C}_b. \tag{2.8}$$

We can then use an eigenvalue decomposition to factor $\mathbf{C}_c$ into

$$\mathbf{C}_c = \mathbf{U}\lambda\mathbf{U}^\top. \tag{2.9}$$

This allows us to define a whitening matrix,

$$\mathbf{P} = \lambda^{-\frac{1}{2}}\mathbf{U}^\top, \tag{2.10}$$

which equalizes the variances so that the eigenvalues of $\mathbf{P}\mathbf{C}_c\mathbf{P}$ are all equal to one. We then apply this whitening transform to $\mathbf{C}_a$ and $\mathbf{C}_b$ so that

$$\mathbf{H}_a = \mathbf{P}\mathbf{C}_a\mathbf{P}^\top \tag{2.11}$$

$$\mathbf{H}_b = \mathbf{P}\mathbf{C}_b\mathbf{P}^\top. \tag{2.12}$$

Note that $\mathbf{H}_a$ and $\mathbf{H}_b$ have the same eigenvectors and can be factored as

$$\mathbf{H}_a = \mathbf{Q}\Psi_a\mathbf{Q}^\top \tag{2.13}$$

$$\mathbf{H}_b = \mathbf{Q}\Psi_b\mathbf{Q}^\top \tag{2.14}$$

with

$$\Psi_a + \Psi_b = \mathbf{I}. \tag{2.15}$$

Since Ψ_a and Ψ_b sum to one and assuming that our eigenvalues are in decreasing order, it is clear that columns of $\mathbf{Q}$ that capture large amounts of the variance in the whitened matrix $\mathbf{H}_a$ must also capture smaller amount of the variance in $\mathbf{H}_b$.

In order to leverage this, we then define a linear transformation matrix,

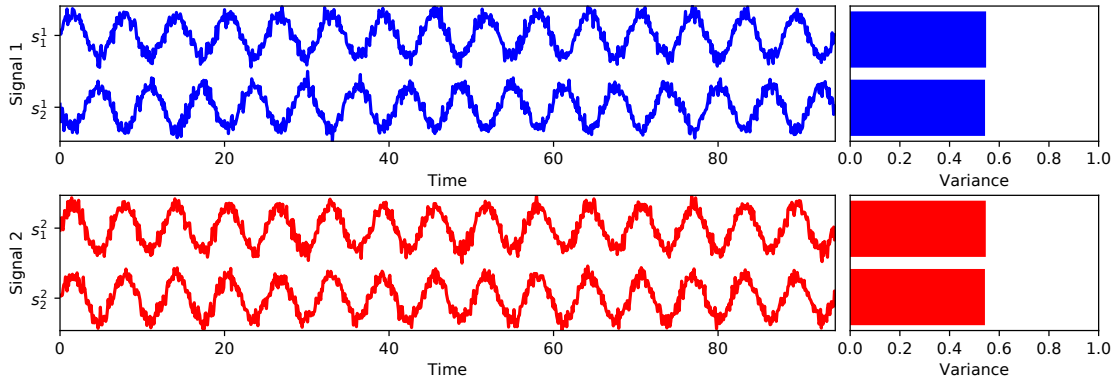
$$\mathbf{W} = \mathbf{P}^\top\mathbf{Q}, \tag{2.16}$$

which applies the whitening transform followed by a projection onto the columns of $\mathbf{Q}$. When an EEG segment is multiplied by $\mathbf{W}$, we expect that the resulting transformed segment,

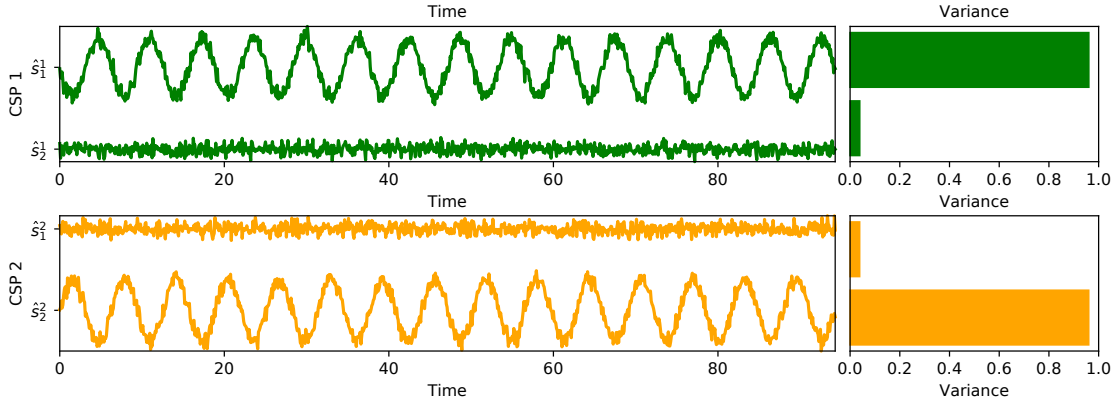
$$\hat{\mathbf{S}} = \mathbf{S}\mathbf{W}, \tag{2.17}$$

will have higher variance in the first columns of $\hat{\mathbf{S}}$ if $\mathbf{S}$ belongs to class A and higher variance in the later columns if $\mathbf{S}$ belongs to class B, at least to the extent possible given our constraints of linearity, orthogonality and a purely spatial transformation.

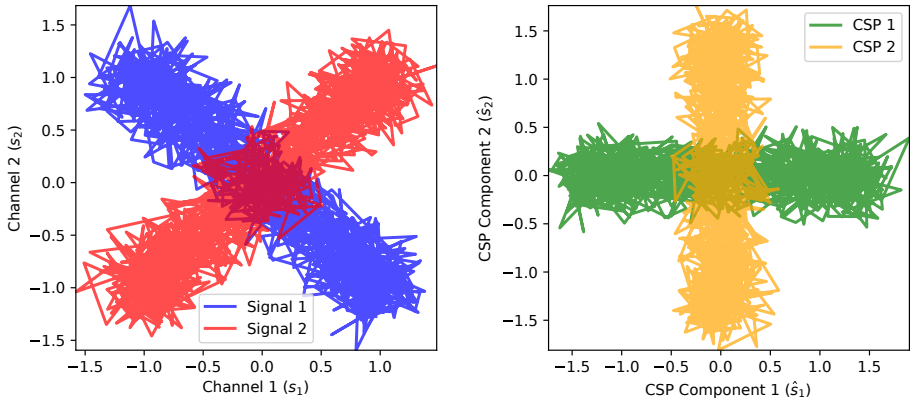
In a classification pipeline that utilizes CSP, several columns of $\hat{\mathbf{S}}$ are selected, usually several of the lowest and highest, and the variance is computed for each of these columns. These variances are then passed as features to a subsequent classification stage, which is typically a linear model. This procedure can lead to considerable dimensionality reduction. If, for example, the two highest and two lowest variances are used as features, then the dimensionality of the EEG segment is reduced to a mere four dimensions, regardless of the length of the segment.



(a) Trace plots and channel variances of narrow-band signals.



(b) Trace plots and channel variances of resulting CSP components.



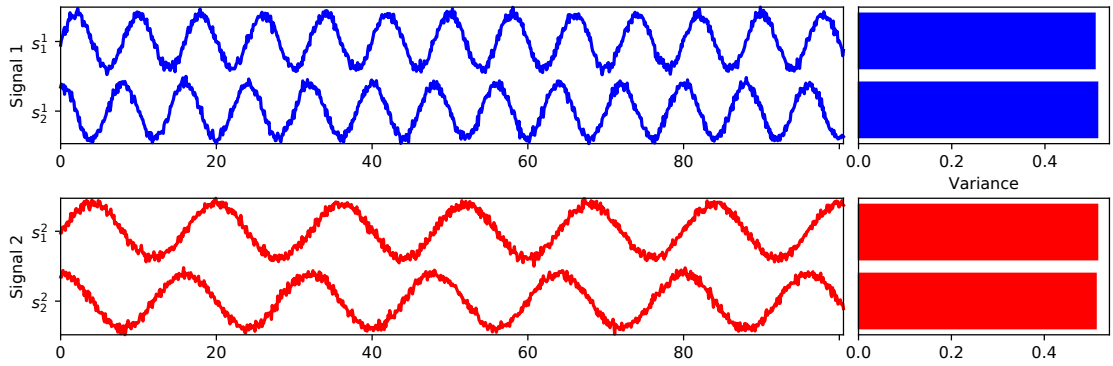
(c) Parametric plots of the signals versus CSP components on a plane.

Figure 2.8: An illustration of how CSP can separate the signal variances for two classes of signals both consisting of two-channel, noisy, narrow-band sinusoidal signals. (a) Signal-1 consists of two sine waves with a phase difference of π radians. Signal-2 consists of two sine waves with identical phase offsets. The variances are divided roughly equally among both channels in both classes. (b) CSP components for both signals. CSP-1 shows that most of the variance for Signal-1 is now in the first component. In CSP-2, most of the variance for Signal-2 is now in the second component. (c) Parametric plots of the signal channels (left) and CSP components (right). CSP performs a rotation to move most of the variance for Signal-1 to the first CSP component and most of the variance for Signal-2 to the second CSP component.

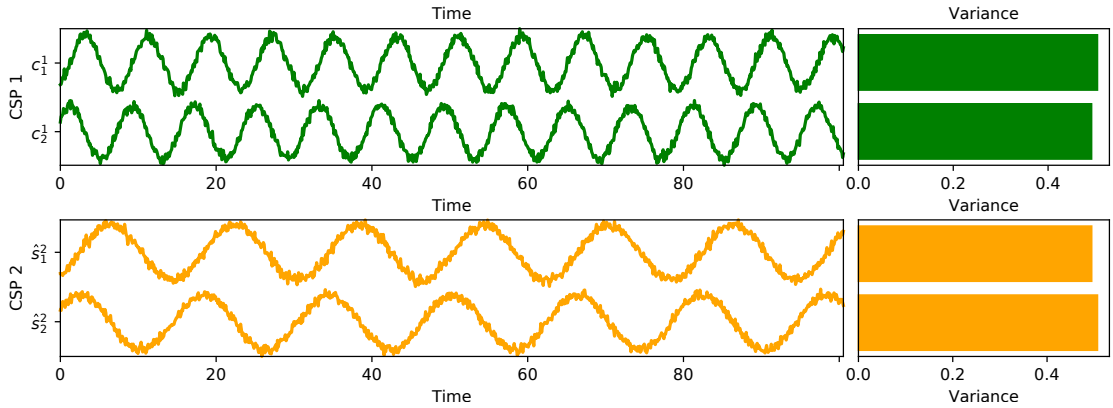
Note that while CSP is usually formulated for binary classification, multiclass generalizations have been proposed and, of course, one-to-one and one-to-many formulations can always be constructed [56, 78].

In Figure 2.8, we see a simple example of CSP applied to two classes of signals that each have two channels. Figure 2.8a shows trace plots of both signals as well as the variance across each channel. The first class, Signal-1, consists of two noisy sine waves with identical frequency and a phase difference of π radians. The second class, Signal-2, consists of two noisy sine waves with identical frequencies and identical phase offsets. Figure 2.8b shows the trace plots and variances of the CSP components, i.e., columns of $\hat{\mathbf{S}}$, for both signals. In these plots, CSP-1 is the result of applying W to Signal-1 while CSP-2 is the result of applying W to Signal-2. Note that CSP-1 has higher variance in the first component while CSP-2 has higher variance in the second component, indicating that CSP was able to achieve good separation of the class variances. Figure 2.8c illustrates how this separation is achieved by plotting the channels of the original signal and then the CSP components on a plane as a system of parametric functions of time. Before CSP is applied, the variance for both signals is spread roughly equally across both channels. Applying the CSP transform W then performs a rotation that places most of the variance for Signal-1 on the first CSP component and most of the variance for Signal-2 on the second CSP component.

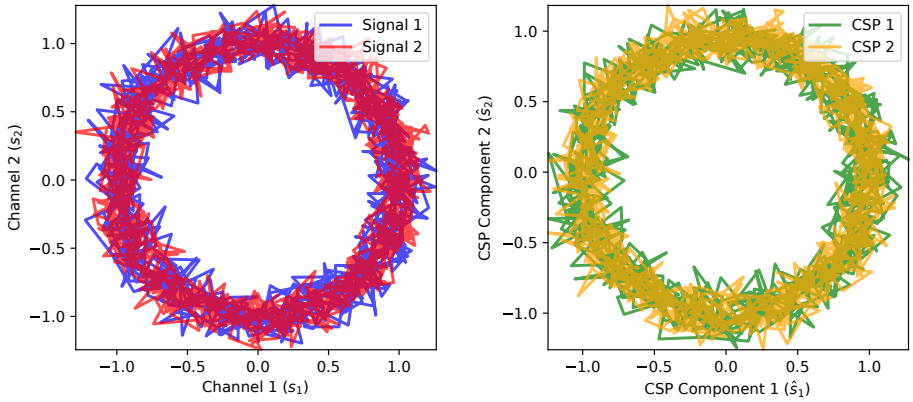
For a narrow-band signal, i.e., a signal that contains a narrow range of frequencies, the variance of a channel is equivalent to band-power, or the mean-squared amplitude, of the signal. In MI-based BCIs, we often assume that there will be reliable changes in amplitude in the relatively narrow-band μ and β ranges. In this case, CSP can be an effective method for learning the spatial distribution of the band-power in these ranges, provided that the signal is first bandpass filtered to discard other frequency ranges. In the more general MT paradigm, however, we do not necessarily know a priori which frequency ranges might be useful. Unfortunately, CSP does not generally provide a good separation of signal variance in the case of broad-band signals.



(a) Trace plots and channel variances of signals with different frequencies.



(b) Trace plots and channel variances of resulting CSP components.



(c) Parametric plots of the signals and CSP components on a plane.

Figure 2.9: An illustration of how CSP can fail to separate the signal variances for two classes of noisy, two-channel sinusoids with different frequencies. (a) Signal-1 and Signal-2 both consist of two sine waves with a phase difference of π radians. Frequencies are the same within each class and different across classes. The variances are divided roughly equally across the two channels in both classes. (b) CSP components and variances for both signals are largely unchanged from the original signals. (c) Parametric plots of the signal channels (left) and CSP components (right). Since both signals lie on the unit circle, there is no linear transform that can separate the variances.

Figure 2.9 presents a simple example of a case in which CSP is unable to differentiate between two classes of signals that contain different frequencies. In Figure 2.9a, we again see trace plots and the channel variances for two classes of signals. Signal-1 and Signal-2 consist of two sine waves with a phase difference of $\frac{\pi}{2}$ radians. Unlike our first example, however, the frequencies are now different between the two classes of signals, i.e., they are no longer narrow-band across classes. In Figure 2.9b, we see the CSP components for each signal. Note that CSP is unable to separate the variances. Figure 2.9c illustrates why this is the case using parametric plots of the signal channels and CSP components. Specifically, all of the data points for both Signal-1 and Signal-2 lie on the unit circle. As a result, there is no linear transform that can rotate this space in a way that separates the variances in the channels of the original signal.

Although CSP addresses a number of the challenges listed in Section 1.1, it also suffers from several important limitations. Time invariance is achieved by using only the variances of the CSP components over a brief window. Using only the variances may discard important information, including the ordering of events within the window. Noise and undersampling are partially handled by using bandpass filters to isolate relevant narrow-band signals and through the dimensionality reduction that is provided by selecting CSP components and by using only the signal variances. As we have previously discussed, however, these procedures are time consuming, rely on prior assumptions and can lead to information loss. CSP is computationally efficient and can be interpreted by mapping the weights and variances of each component back to the surface of the scalp; however, it is generally not possible to precisely characterize nonlinear, temporal or multiscale patterns.

Although CSP has been quite successful in MI-based BCIs, it clearly has a number of limitations that make it ill suited for use in broader EEG classification tasks. In fact, we believe that CSP is a good example of an approach that works well only under a set of strict prior assumptions, i.e., linear, narrow-band signals with spatially distributed difference in signal amplitude. CSP also requires extensive manual guidance and feature selection in order to construct appropriate frequency bands and CSP components.

2.1.3 Time-Delay Embedding

Time-Delay Embedding (TDE) is a general, time-domain method for capturing localized patterns in EEG signals that was proposed for use in MT-based BCIs by Anderson, et al. [63–66]. TDE incorporates signal values across channels and over a window of time, which allows it to capture spatiotemporal information while relying on few prior assumptions. If we again consider the matrix representation of an EEG segment, as described in (2.5), then the TDE representation can be expressed as

$$\mathscr{E}\{\mathbf{S};\, d\} = \underbrace{\begin{pmatrix} s_{1,1} & s_{2,1} & \cdots & s_{d,1} & s_{1,2} & s_{2,2} & \cdots & s_{d,2} & \cdots & s_{d,N} \\ s_{2,1} & s_{3,1} & \cdots & s_{d+1,1} & s_{2,2} & s_{3,2} & \cdots & s_{d+1,2} & \cdots & s_{d+1,N} \\ s_{3,1} & s_{4,1} & \cdots & s_{d+2,1} & s_{3,2} & s_{4,2} & \cdots & s_{d+2,2} & \cdots & s_{d+2,N} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ s_{T-d+1,1} & s_{T-d+2,1} & \cdots & s_{T,1} & s_{T-d+1,2} & s_{T-d+2,2} & \cdots & s_{T,2} & \cdots & s_{T,N} \end{pmatrix}}_{\substack{\text{Time embedding for first channel} \\ \text{Time embedding for second channel}}}, \tag{2.18}$$

where N is the number of channels, T is the number of timesteps and d is the width of the window, also known as the embedding dimension. $\mathscr{E}\{\mathbf{S};\, d\}$ contains the same signal values as $\mathbf{S}$ but rearranged with some duplication so that each row of $\mathscr{E}\{\mathbf{S};\, d\}$ contains the information from d adjacent rows of $\mathbf{S}$. In other words, each row of $\mathscr{E}\{\mathbf{S};\, d\}$ contains signal values from across all channels during a window of time that spans d adjacent timesteps.

The rows of $\mathscr{E}\{\mathbf{S};\, d\}$ can be viewed as feature vectors that contains localized spatiotemporal information that can then be passed to a subsequent classifier. Although any number of classification algorithms may be suitable for this role, previous work by Anderson, et al., has demonstrated that nonlinear feedforward neural networks appear to outperform linear and quadratic discriminant analysis [66].

Unlike the previous approaches that we have examined, TDE generally produces a class label for every timestep in the EEG segment. For applications that require smooth and continuous control, this property may be advantageous. It is important to note, however, that the

rate at which this information is delivered is generally much faster than a user can change their mental state. The noise and high sampling rate of EEG signals may also cause this raw control signal to fluctuate rapidly and lack the type of smooth and reliable behavior that BCI applications typically require.

One can imagine a number of potential approaches for smoothing and improving the reliability of these control signals. For instance, moving average, bandpass or other denoising filters could be explored. Alternately, a slower fixed-rate control signal could be provided by aggregating class labels over a brief period of time using voting or averaging schemes. This is similar to the approach used by Lawhern, et al., and to the approach that we will propose in Section 3.2.1 [52]. In our previous work, we have also proposed the use of an evidence accumulation strategy that builds confidence over the course of time in order to deliver a more reliable but variable-rate control signal [66]. This approach allows a BCI to only perform an action when it is sufficiently confident in the user's intent. Overall, the flexibility provided by TDE allows a BCI system to be tailored to a wide variety of potential requirements, ranging from applications that are fast but not necessarily reliable to mission-critical applications where reliability is more important than speed.

TDE is a general approach that addresses many of the challenges that we have outlined in Section 1.1. TDE encourages time invariance because all windows of size d are considered, i.e., the classifier is trained to label each window of the segment, regardless of its starting time. It is also capable of capturing spatiotemporal patterns, provided that the patterns are localized in time and that the subsequent classifier has sufficient expressive power. TDE does not discard information, except for the way that the embedding dimension limits the length of patterns that can be captured. TDE also does not rely on prior assumptions about linearity, stationarity, phase relationships or the bandwidth of the signals. TDE may be less susceptible to problems with undersampling because the classifier is presented with a separate training example for each timestep, as opposed to a single example per EEG segment. Noise and undersampling can

also be addressed by limiting the size of the embedding dimension, which limits the dimensionality of the representation, and also through preprocessing and classifier regularization.

Despite the favorable aspects of TDE, it also suffers from several significant limitations. First of all, there is an important tradeoff between the size of the embedding dimension, the dimensionality of the representation and the length of the temporal patterns it can capture. For small values of d, the dimensionality of the representation is small and problems with noise, undersampling and overfitting are likely to be reduced; however, only short-term patterns can be captured. For larger values of d, on the other hand, longer-term patterns can be captured but the dimensionality of the representation is large and overfitting is likely to be problematic. Similarly, TDE does not encourage multiscale patterns to be leveraged; instead, each point in time is considered equally. TDE can also be challenging to interpret. Although a number of standard methods can be used to examine the patterns learned by the classification stage, it is challenging to draw useful conclusions about the relationship between these patterns and the underlying processes in the EEG signals and in the brain. Finally, TDE is often computationally expensive. Although the process of creating the matrix $\mathscr{E}\{\mathbf{S}; d\}$ is efficient, larger sample sizes and reliance on nonlinear classifiers can lead to high computational requirements. This is especially true for large embedding dimensions and intricate regularization processes.

Although TDE has a several disadvantages, we believe that it shows considerable potential because it is very general and relies on few prior assumptions. TDE is relatively under explored in the context of BCIs, however, and we are not aware of any literature that directly compares it to other approaches. For these reasons, we will continue to explore TDE and, in Section 3.2, we will describe an approach for using TDE in combination with feedforward networks that we will examine in detail. In Section 3.2.2, we will then demonstrate that TDE can be viewed as a multivariate generalization of discrete convolution.

2.2 Deep Learning

We have now explored several common methods for representing EEG signals and we have demonstrated that each of these approaches has important limitations. The goal of this discussion has been two fold. First, we have motivated the need for investigating new methods by demonstrating that current approaches fail to meet our stated requirements. Second, we have illustrated how representations that rely on prior knowledge and manual engineering procedures can limit our ability to exploit unknown patterns that may exist in the signals. In order to reinforce this second point, we will now offer a brief introduction to deep learning, with a focus on convolutional networks, and describe how these methods have begun to play an increasingly important role in machine learning.

For many years, machine learning methods have relied heavily on manually engineered features, ranging from basic statistics to general methods for dimensionality reduction to various metrics that are highly domain specific. Although these methods for feature generation have been used successfully in many settings, they typically involve a laborious design phase and require the engineer to establish knowledge about the types of patterns in the data that capture relevant information. This has lead to specialized machine learning systems that are only able to fill narrow roles and are unable to exploit subtle but important patterns. The ability of the human brain to process and learn from very general sensory inputs has made it clear, however, that it is possible to create fast and general-purpose pattern recognition systems. This has lead researchers to seek inspiration in biological neural networks.

Artificial neural networks that are, at least partly, inspired by biological networks have been in use for many years and have been successfully applied to a variety of problems in machine learning [79, 80]. These networks consist of a number of simple, interconnected computational units, called artificial neurons. Shallow networks, which typically consist of two layers of artificial neurons with dense connectivity, have enjoyed notable success but can require manual feature engineering to solve some types of problems, e.g., general object recognition in computer vision. Deep networks, which are characterized by the use of many layers, have also been

around for quite some time and have been explored in a variety of contexts [41, 81, 82]. Similar to the way it is believed that biological network process information, deep networks are capable of learning a hierarchy of features where each layer represents an increasing level of abstraction. It has proven to be challenging, however, to utilize deep networks in practice due to the large number of parameters that they contain and because error gradients tend to dissipate rapidly through multiple layers, which makes optimizing the connection weights of deep networks difficult [83].

Several recent advances have made deep networks both more effective and more practical to use. First of all, researchers have observed that biological neural networks often have multilayer, localized, sparse and recurrent connectivity patterns. By incorporating these connectivity patterns into artificial neural networks, usually in a very simplified way, the number of free parameters can be reduced while also encouraging the network to learn robust, general representations. Second, the availability of large, high-quality datasets allows deep networks with many layers and free parameters to learn sophisticated representations that generalize well. Transfer learning approaches are often valuable for achieving these goals [82, 84]. Finally, improvements in optimization algorithms and computer hardware, especially vector processors, have made it tractable to train large networks over large datasets [85–90]. Currently, deep networks hold record performance on a number of benchmark datasets and are generally considered to be state-of-the-art for many machine learning tasks [41].

2.2.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a class of deep networks with several design elements that are highly influenced by observations about the biological neural networks found in the animal vision system [42–44, 91–93]. In the early stages of these biological networks, it has been observed that individual neurons respond almost exclusively to a small region of the retina that is associated with a corresponding region of the animal's visual field [94–96]. This localized region of response is referred to as the local receptive field of the neuron. Similarly,

deeper regions of the vision system also contain neurons that respond to localized regions of neurons in the earlier stages of the vision system. This suggests that biological neural networks in the animal vision system are, at least partly, composed of multiple layers of neurons that are connected to previous layers via localized connectivity patterns. As the eye scans a scene, the result of light moving along the receptive field of the retina can be loosely modeled as a convolution, i.e., a sliding window of responses.

CNNs follow a similar connectivity pattern by convolving the inputs at each layer with a shared window of weights, known as the convolutional kernel, followed by some nonlinear transfer function. This process resembles the localized response of a biological neuron in the animal visual system. One of the primary advantages of this architecture is that it encourages the network to learn representations that are insensitive to shifts along the convolutional axes. Since each neuron is presented with only a small window of inputs at any given time and because the same kernel is applied to all such windows, the response of the neuron is unable to depend on the global location of a given pattern. The use of localized connectivity patterns can also be viewed as a form of sparsity, which is also prevalent in biological networks. Since fully connected networks have many input connections, they can become highly sensitive to specific combinations of inputs. Since CNNs have fewer incoming connections to each neuron, they tend to have a reduced sensitivity to specific input patterns, i.e., they tend to leverage more general patterns.

The animal vision system also tends to have a layered, or columnar, architecture and consists of a number of regions that, to varying degrees, appear to model increasingly abstract visual information [96]. Stacking multiple convolutional layers in a CNN mimics this configuration and encourages the network to learn a hierarchical representation since each subsequent layer utilizes the cumulative representation of the previous layers. CNNs also typically incorporate a downsampling strategy between layers, known as pooling, in order to force a change of scale. It is also common practice to decrease the size of the convolutional kernels while simultaneously increasing the number of neurons in each layer of a CNN [42]. Together, these

procedures result in a pyramid structure where information is increasingly migrated from the axes along which the convolutions are performed onto a larger number convolutional units with more localized responses. This results in a network that is encouraged to learn a hierarchical and shift-invariant representation while potentially utilizing fewer parameters than a fully connected network [45].

2.2.2 CNNs in Computer Vision

In addition to being partially inspired by the biological networks in the animal vision system, CNNs have also achieved notable success in the field of computer vision and many advances in the CNN architecture have evolved from these lines of research [42–46, 48, 49, 91, 97]. Given the importance of CNNs in the field of computer vision, it is useful to first consider how these types of networks are can be used for image classification. In Figure 2.10, we see a schematic diagram of a CNN architectures that is commonly used to classify images. The initial layers consist of artificial neurons with small, localized windows of connectivity. Each connection in this window has a separate weight that is applied to all regions of the layer's input via a convolution across both the horizontal and vertical axes. The output of this convolution is then passed through a nonlinear transfer function, typically the hyperbolic tangent or rectified linear function. The output is then downsampled in a process called pooling. A variety of techniques for pooling have been explored including averaging, striding or selecting the maximum response. At each convolutional layer, this process produces a new feature image for each neuron. The general idea is that each neuron in a convolutional layer can learn to respond to different types of patterns, regardless of their location in the image, i.e., the response is shift invariant. The convolutional layers are then stacked in order to encourage the network to learn hierarchical and multiscale representations.

The output of the final convolutional layer is then passed through an artificial neural network with full connectivity. In other words, weighted sums of the pixels in all of the final feature images are passed to the neurons in the fully connected layer, followed by another nonlinear

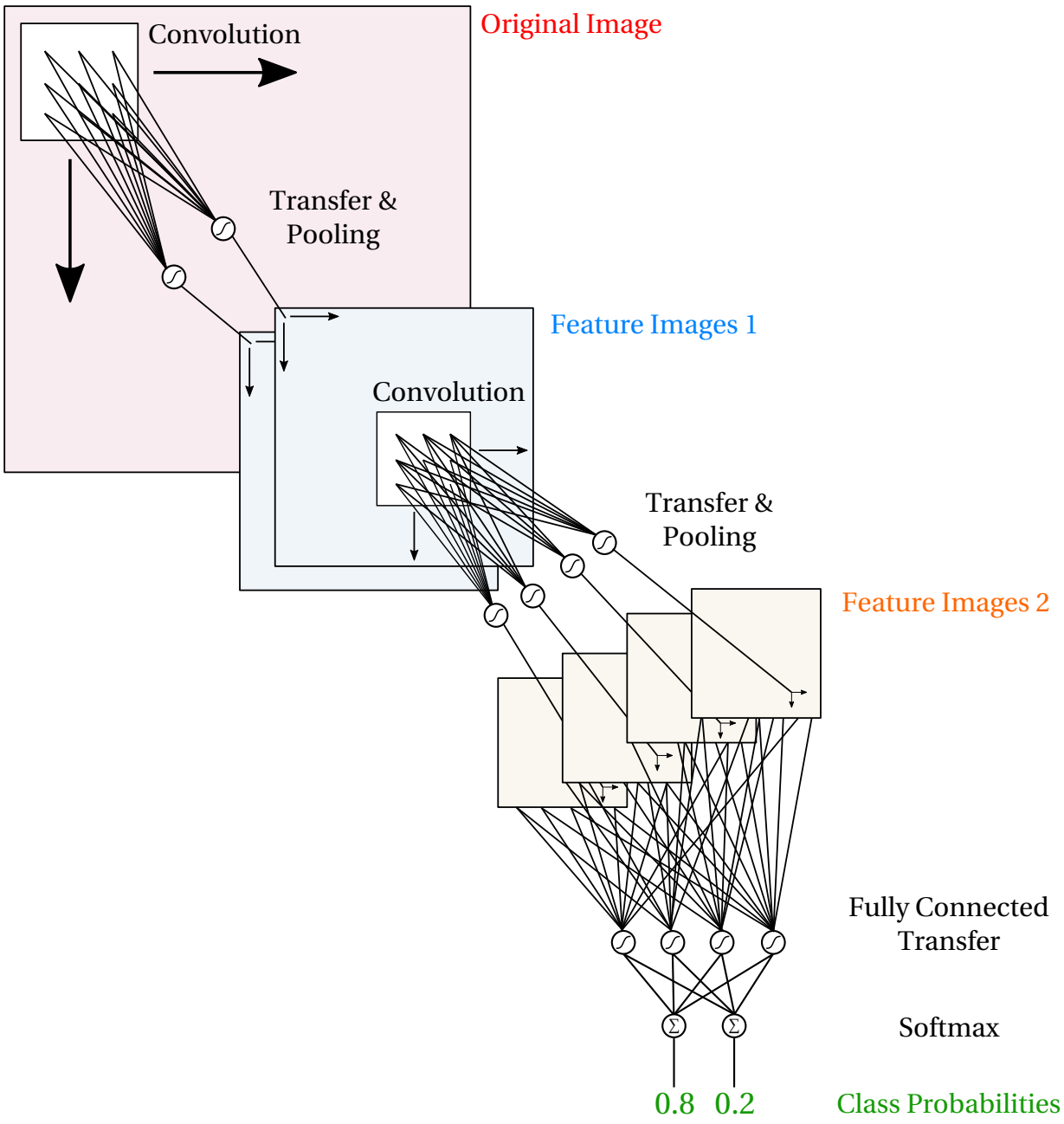


Figure 2.10: Schematic of a CNN designed for image classification. This network consists of two convolutional layers with two and four neurons followed by a fully connected layer with four neurons and, finally, a softmax readout that outputs two class probabilities.

transfer. Finally, linear softmax units can be used to combine these outputs and ensure that they sum to one, i.e., the probability that the image belongs to each class. It is important to note that this fully connected network may have many fewer parameters than a fully connected network applied to the original image because the feature images are downsampled between convolutional layers.

2.2.3 CNNs in Brain-Computer Interfaces

Recently, several research groups have also begun to explore the use of CNNs in BCI systems. For instance, Cecotti, et al., have proposed a modified CNN architecture for classifying EEG signals in synchronous P300 BCIs [51]. Cecotti, et al., have also explored a variant of the CNN architecture for classifying EEG signals in asynchronous BCIs that incorporates DFTs between the initial convolutional layers [50]. In this approach, the stimulus-driven Steady State Visual Evoked Potential (SSVEP) communication protocol was used. Since SSVEP relies on known frequency ranges in EEG signals, the use of DFTs aids in feature selection. Although this approach begins to bridge the gap between CNNs and PSDs, it may also suffer from some of the same limitations that we have identified in PSD-based approaches.

Lawhern, et al., have recently proposed a CNN architecture, which they call EEGNet, for classifying EEG signals across several different types of BCIs, including both synchronous and asynchronous paradigms [52]. This recent work was conducted concurrently with our research and independently draws a number of similar conclusions. Specifically, EEGNet utilizes relatively small CNNs with simplified label aggregation readout layers. EEGNet is subtly different than our approach, however, with respect to the design of its convolutional layers and the way that information is aggregated in its readout layers. EEGNet has also not been explored with mental tasks other than motor imagery and does not utilize transfer learning. In offline experiments, EEGNet outperforms baseline methods for P300 classification and performs comparably to a CSP-based classifier for MI communication paradigms.

Schirrmeister, et al., have explored CNNs for classifying EEG signals in multiclass MI-based BCIs [53]. These networks utilize fully connected layers but the initial layers are narrowed across time and channels in order to capture spatiotemporal patterns with few parameters. This work further supports the notion that relatively small CNN architectures often work well for classifying EEG signals. Schirrmeister, et al., also propose a method for analyzing their CNNs by correlating various frequency bands with the responses of the convolutional layers. This CNN architecture achieves classification performance that is comparable to their baseline approach that utilizes CSP.

Recent work exploring the use of CNNs in BCIs clearly suggests that this is a valuable approach that may overcome many of the limitations found in traditional EEG signal representations. These studies also consistently indicate that relatively small CNN architectures tend to perform well. Many questions remain to be answered, however, including the potential for CNNs to be used in MT-based BCIs and how various design decisions affect performance. A more in-depth analysis of these types of networks is also warranted, including a mathematical analysis of their capabilities, experimental techniques for analyzing the types of patterns that they learn to identify and direct comparisons to other approaches. In this work, we extend the current body of research by examining these topics in detail.

2.2.4 Transfer Learning

Machine learning systems used in BCIs are typically retrained, using new data, on a per-subject and per-session basis. Training individualized models often provides better performance than generic models because EEG signals are known to be highly variable over time and across individuals [98]. Larger training data sets may, however, help to address problems with noise, undersampling and overfitting. As a result, it may be valuable to consider incorporating data from other participants or training sessions in some capacity. The process of using data from one domain to aide in the development or training of a machine learning model in another domain is known as transfer learning [84, 99].

Although a handful of research projects have explored transfer learning in BCIs [98, 100], these approaches have not explored neural network based classification algorithms and primarily focus on P300 and MI communication paradigms. We are not presently aware of any research projects that have explored transfer learning for use with deep neural networks or for use in MT-based BCIs. In the broader field of machine learning, however, the use of transfer learning approaches has become increasingly popular and has proven to be invaluable for improving the performance of models in a variety of domains [47, 101–103].

The usual approach for applying transfer learning to CNNs is to first perform an initial training procedure using data from a different domain but that is expected to contain similar types of patterns. The dataset used for this initial training stage often contains many more examples than are available for the dataset that is of primary interest. Some or all of the layers of the network can then be fine-tuned by further training the network using the data for the task at hand. Since the fine-tuning stage starts with the connection weights learned during the initial training stage, potentially useful knowledge may be transferred from one domain to another. For example, a CNN might initially be trained to classify images of cars and trucks using a dataset that contains many millions of examples. The network can then be fine-tuned to classify images of motorcycles and bicycles, for example, using a dataset that consists of only a few hundred images. Since the initial model may have learned to identify important types of patterns in these images, e.g., wheels and roads, performance may be improved for the smaller dataset. In Section 3.3.3, we will propose a similar method for transfer learning that utilizes EEG data collected from across multiple subjects.

Chapter 3

Methods

Now that we have described the advantages and disadvantages of current approaches and provided motivation for exploring multilayer CNNs, we will continue by providing a thorough description of our experimental methods and proposed network architectures. First, we will describe a set of baseline classifiers that utilize PSDs. These types classifiers are commonly used in MT-based BCIs and we believe that they are generally representative of the state of the art. As such, these classifiers will serve as a benchmark for our performance comparisons in Chapter 4. We will then describe a classification approach that uses TDE combined with two-layer artificial neural networks, known as Time-Delay Neural Networks (TDNNs), along with a fixed-rate label aggregation strategy. As discussed in Section 2.1.3, we believe that TDE-based approaches show considerable potential but are currently under explored. We will then precisely describe our proposed CNN variants and demonstrate that these networks are a multilayer generalization of our proposed TDNNs. We will also describe several methods for interpreting the behavior of these networks and for analyzing the patterns that they learn to identify. We will then discuss the details of our implementation, including transfer, pooling, weight initialization, regularization, performance metrics and software. Finally, we will describe the dataset that we have collected for experimentation, which is designed to simulate an MT-based BCI application using portable hardware under both laboratory and realistic operating conditions.

3.1 Baseline Classifiers

A variety of approaches for classifying PSDs in MT-based BCIs have been demonstrated with notable success [33, 34, 54, 67, 69, 70]. There has been little comparative work before now, however, exploring how different preprocessing methods, hyperparameters and classification algorithms affect performance. In this section, we will first describe our approach for constructing PSD-based features along with several preprocessing steps that we have found to generally im-

prove performance. We will then continue by describing several algorithms for classifying these features, including variants of discriminant analysis and artificial neural networks, summarized in Table 3.1. In Section 4.1.1, we will thoroughly explore the hyperparameters for each of these approaches and in Section 4.2.1 we will examine the final performance for each of these methods in order to establish a benchmark for evaluating our proposed CNNs.

Table 3.1: Summary of PSD baseline classifiers.

Abbr.	Name	Regularization
LDA	Linear Discriminant Analysis	Covariance matrix shrinkage
QDA	Quadratic Discriminant Analysis	Covariance matrix averaging
ANN	Two-Layer Artificial Neural Network	Early-stopping

3.1.1 Preprocessing and Feature Generation

We have identified a series of preprocessing and feature generation steps that generally work well with PSD-based approaches. First, we split the EEG signals into M segments that can each be described using our matrix notation in (2.5). We then construct feature representations for each of these segments that can be passed to a classification algorithm, where they will be used either for training or to be labeled during testing or system use. The width and number of segments determines the rate at which the system responds, which leads to a tradeoff between accuracy and responsiveness that will be further explored in Section 4.2.4.

For each of these segments, we first apply a Common Average Reference (CAR), which subtracts the mean across all channels from each channel at each timestep. This procedure can be described by the operator

$$CAR\{\mathbf{S}\} = \mathbf{S} - \mathbf{u}, \tag{3.1}$$

where $\mathbf{u} \in \mathbb{R}^T$ is a row vector containing the means across the columns, i.e., the channels, of a segment $\mathbf{S}$ and where T is the number of timesteps in the segment, i.e. the number rows in $\mathbf{S}$. CAR acts as a simple spatial lowpass filter and attenuates aspects of signal that occur across

all channels, such as slow movement artifacts and, to some extent, ocular artifacts. CAR can, however, also remove legitimate EEG signals that are observed across broad regions of the scalp.

We have found that CAR generally improves classification performance for PSD-based approaches. In several pilot experiments, we have found that CAR yields roughly a 1–5% increase in classification accuracy, which is an incremental but valuable improvement. It is important to note, however, that CAR does *not* usually improve classification performance for TDNNs or CNNs. As we will later describe, it appears that TDNNs and CNNs are able to automatically learn appropriate signal filters. As a result, we only use CAR in our baseline classifiers that utilize PSDs.

We then construct frequency-domain PSD representations of the segments using the DFT and Welch's method, as previously described in Section 2.1.1. Until now, we have described the PSD only in terms of univariate time series. In order to construct the PSD of our multivariate EEG segments, we compute the PSD separately for each channel and each segment. Since the corners of our passband are from 1–80Hz, as will be described in section Section 3.5.2, we retain only the frequency bins that lie within this range. We then concatenate these PSDs across channels to form an $M \times NQ$ feature matrix, $\mathbf{X}$, where M is the number of segments, N is the number of channels and Q is the number of frequency components in each PSD. The classification stage then considers the rows of $\mathbf{X}$ to be observations, which will be labeled, and the columns of $\mathbf{X}$ to be features.

Recall that Welch's method smooths a PSD in the frequency-domain while simultaneously reducing the number of frequency components, Q. The extent of this smoothing and dimensionality reduction is controlled by the span, ω. Since ω controls the resolution of our representation as well as the complexity of our models, we consider it to be an important hyperparameter that will be carefully explored in Section 4.1.1.

We then apply a log_{10} transform to each value of $\mathbf{X}$. This procedure helps to reduce the disparity between the relatively large powers that are typically encountered at lower frequencies versus the smaller powers encountered at high frequencies. Next, we scale the columns of $\mathbf{X}$ to

have zero mean and unit variance across all EEG segments in the training partition using the same scales across all classes. This transformation further helps to ensure that each feature has a similar scale while preserving relative differences across classes. Scaling to zero mean and unit variance can improve numerical stability when computing covariance matrices in discriminant analysis and it is also useful for choosing a good weight initialization when using feedforward neural networks. Finally, our scaled feature matrix is passed to the classification stage.

3.1.2 Discriminant Analysis

Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA) are generative, Bayesian classification algorithms that are popular in the field of BCIs and have been successfully used to classify PSDs in the past [33, 67, 70]. LDA and QDA model the observations from each class using a multivariate Gaussian distribution. Novel data can then be classified by assigning the label associated with the distribution that most likely generated the data.

Assume that we have a feature vector representing a single observation, $\mathbf{x} \in \mathbb{R}^F$, where $F = NQ$ is the number of features in our PSD representation. Next, notice that we can use Bayes' Theorem to express the probability that a random variable, C, describing class membership is a specific class c given our observation matrix $\mathbf{x}$,

$$P(C = c \,|\, \mathbf{x}) = P(\mathbf{x} \,|\, C = c) \frac{P(C = c)}{P(\mathbf{x})}. \tag{3.2}$$

If we then model the distribution for each class, $P(\mathbf{x} \,|\, C = c)$, as a multivariate Gaussian distribution, then we now have

$$P(C = c \,|\, \mathbf{x}) = \frac{e^{-\frac{1}{2}(\mathbf{x}-\mu_c)\Sigma_c^{-1}(\mathbf{x}-\mu_c)^\top}}{\sqrt{(2\pi)^N |\Sigma_c|}} \frac{P(C = c)}{P(\mathbf{x})}, \tag{3.3}$$

where $|\cdot|$ denotes the matrix determinant, μ_c and Σ_c are the sample mean vector and covariance matrix for each class c and where $P(C = c)$ is the class prior, i.e., the number of training observations in class c divided by the total number of training observations. Next, it is convenient to

consider the log likelihood,

$$\log P(C=c\,|\,\mathbf{x}) = \log\left[\frac{e^{-\frac{1}{2}(\mathbf{x}-\mu_c)\Sigma_c^{-1}(\mathbf{x}-\mu_c)^\top}}{\sqrt{(2\pi)^N|\Sigma_c|}}\frac{P(C=c)}{P(\mathbf{x})}\right] \tag{3.4}$$

$$= \log e^{-\frac{1}{2}(\mathbf{x}-\mu_c)\Sigma_c^{-1}(\mathbf{x}-\mu_c)^\top} - \log\sqrt{(2\pi)^N|\Sigma_c|} + \log P(C=c) - \log P(\mathbf{x}) \tag{3.5}$$

$$= -\frac{1}{2}(\mathbf{x}-\mu_c)\Sigma_c^{-1}(\mathbf{x}-\mu_c)^\top - \frac{N}{2}\log(2\pi) + \frac{1}{2}\log|\Sigma_c| + \log P(C=c) - \log P(\mathbf{x}). \tag{3.6}$$

We now seek to find the label, L, which corresponds to the class model that most likely generated $\mathbf{x}$. This can be expressed as

$$L = \underset{c=1,\ldots,K}{\operatorname{argmax}} \log P(C=c\,|\,\mathbf{x}), \tag{3.7}$$

where K is the total number of classes. We can then simplify this expression by canceling terms that are common across all classes, which yields

$$L = \underset{c=1,\ldots,K}{\operatorname{argmax}} -\frac{1}{2}(\mathbf{x}-\mu_c)\Sigma_c^{-1}(\mathbf{x}-\mu_c)^\top - \frac{N}{2}\log(2\pi) + \frac{1}{2}\log|\Sigma_c| + \log P(C=c) - \log P(\mathbf{x}) \tag{3.8}$$

$$\equiv \underset{c=1,\ldots,K}{\operatorname{argmax}} -\frac{1}{2}(\mathbf{x}-\mu_c)\Sigma_c^{-1}(\mathbf{x}-\mu_c)^\top + \frac{1}{2}\log|\Sigma_c| + \log P(C=c) \tag{3.9}$$

$$= \underset{c=1,\ldots,K}{\operatorname{argmax}} -\frac{1}{2}\left[(\mathbf{x}-\mu_c)\Sigma_c^{-1}(\mathbf{x}-\mu_c)^\top + \log|\Sigma_c|\right] + \log P(C=c). \tag{3.10}$$

Equation (3.10), which is quadratic in $\mathbf{x}$ and produces a quadratic class boundary, is known as the discriminant function for the aptly named QDA classification algorithm.

Notice that $\Sigma_c \in \mathbb{R}^{F\times F}$ and its inverse may not exist for under-determined problems, as can be the case for large values of our span, ω. In order to account for this, we estimate Σ_c^{-1} using the Moore-Penrose pseudo inverse. Additionally, we will explore a technique for regularizing QDA by mixing the covariance matrix with the average of the covariance matrices across all classes, denoted Σ. We can then substitute Σ_c in (3.10) with

$$\gamma\Sigma + (1-\gamma)\Sigma_c, \tag{3.11}$$

where $0 \leq \gamma \leq 1$ is a tuned regularization hyperparameter. In the extreme case where $\gamma = 0$, we have ordinary QDA. When $\gamma = 1$, on the other hand, we assume that all classes have the same, averaged covariance matrix, $\Sigma_1 = \Sigma_2 = \ldots = \Sigma_K = \Sigma$.

When we substitute the average class covariance matrix, Σ, into (3.10), additional terms cancel. Our discriminant function then becomes

$$L = \underset{c=1,\ldots,K}{\operatorname{argmax}} \log P(C = c \,|\, \mathbf{x}) \tag{3.12}$$

$$= \underset{c=1,\ldots,K}{\operatorname{argmax}} -\frac{1}{2}\left[(\mathbf{x}-\mu_c)\Sigma^{-1}(\mathbf{x}-\mu_c)^\top + \log|\Sigma|\right] + \log P(C=c) - \log P(\mathbf{x}) \tag{3.13}$$

$$\equiv \underset{c=1,\ldots,K}{\operatorname{argmax}} -\frac{1}{2}(\mathbf{x}-\mu_c)\Sigma^{-1}(\mathbf{x}-\mu_c)^\top + \log P(C=c) \tag{3.14}$$

$$= \underset{c=1,\ldots,K}{\operatorname{argmax}} -\frac{1}{2}(\mathbf{x}\Sigma^{-1}\mathbf{x}^\top - 2\mathbf{x}\Sigma^{-1}\mu_c + \mu_c\Sigma^{-1}\mu_c) + \log P(C=c) \tag{3.15}$$

$$= \underset{c=1,\ldots,K}{\operatorname{argmax}} -\frac{1}{2}(-2\mathbf{x}\Sigma^{-1}\mu_c + \mu_c\Sigma^{-1}\mu_c) + \log P(C=c) \tag{3.16}$$

$$= \underset{c=1,\ldots,K}{\operatorname{argmax}} \mathbf{x}\Sigma^{-1}\mu_c + -\frac{1}{2}\mu_c\Sigma^{-1}\mu_c + \log P(C=c). \tag{3.17}$$

Equation (3.17), which is now linear in $\mathbf{x}$ and produces a linear class boundary, is known as the discriminant function for LDA. We can further regularize LDA using an approach, known as shrinkage, where the average covariance matrix is mixed with a purely diagonal matrix. In this approach, we substitute Σ in (3.17) with

$$\lambda\eta\mathbf{I} + (1-\lambda)\Sigma, \tag{3.18}$$

where $\eta = \mathit{trace}(\Sigma)/F$ is the average eigenvalue of the average class covariance matrix, $\mathbf{I}$ is the identity and $0 \leq \lambda \leq 1$ is a tuned regularization hyperparameter. In the extreme case where $\lambda = 0$, we simply have LDA. In the other extreme, where $\lambda = 1$, we assume that our class covariance matrix is diagonal and only the class means are used for classification. Note that we use $\eta\mathbf{I}$ because mixing with the average eigenvalue makes the model distribution more spherical but with a scale that approaches the mean of the principal axes of the training data.

3.1.3 Artificial Neural Networks

Artificial Neural Networks (ANNs) are learning structures that process information using a number of relatively simple computational units with weighted interconnections [79,80]. These computational units, referred to as artificial neurons or simply as units, typically perform a weighted sum followed by an optional nonlinear transfer function. In essence, these artificial neurons project their inputs onto a nonlinear basis function in a way that models a portion of the problem to be solved. In feedforward networks, information flows in a single direction, starting with the model inputs, followed by one or more layers of artificial neurons and then through a readout layer that generates the model outputs. By combining information from multiple layers of artificial neurons with appropriate weight values, ANNs can model sophisticated nonlinear problems. In fact, it has been shown that two-layer ANNs are theoretically capable of approximating any function to arbitrary precision, given that the network has enough hidden units and appropriate weight values and transfer functions [104]. In practice, however, it can be challenging to construct ANNs that generalize well on real-world problems where we have limited information about the function to approximate.

In order to classify our PSD-based features, we will explore the use of a common, two-layer ANN architecture, illustrated in Figure 3.1, that can be used to construct nonlinear, discriminative classifiers with probabilistic outputs. In the first layer of these networks, called the hidden layer, the input features are fed into a number of different artificial neurons that perform a weighted sum followed by a nonlinear transfer function. The second layer, referred to as the visible layer, consists of artificial neurons that perform a weighted sum over the outputs of the hidden layer followed by a softmax function. The softmax function results in outputs that sum to one and can be interpreted as class probabilities.

This procedure can be described succinctly in matrix notation. Assume that the input to our classifier is the feature matrix $\mathbf{X} \in \mathbb{R}^{M \times F}$, where M is the number of signal segments and F is the

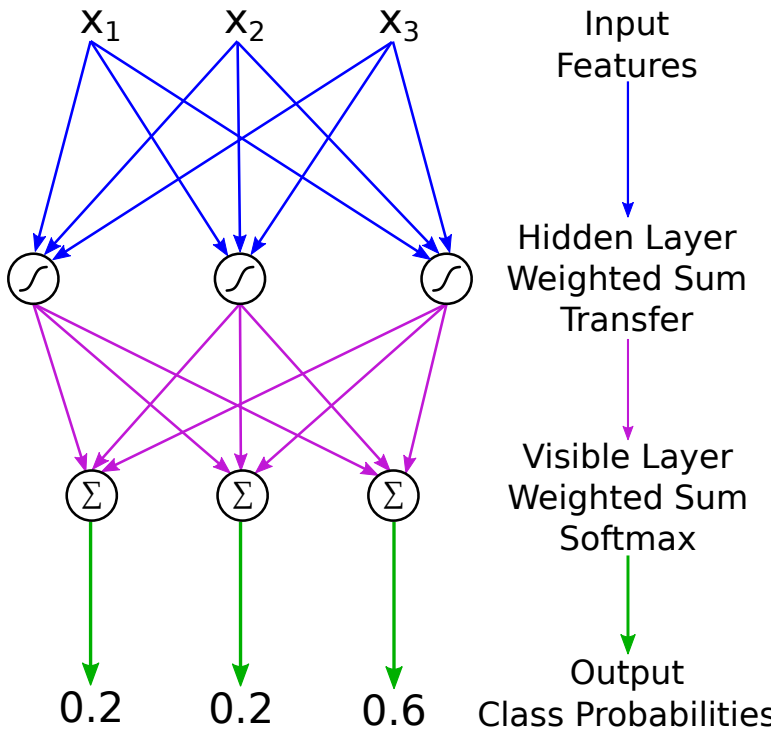


Figure 3.1: A schematic diagram of a two-layer Artificial Neural Network (ANN) used for classifying PSDs. The network consists of a nonlinear hidden layer followed by a linear visible layer with a softmax readout.

number of features. The output of the hidden layer can then be described as

$$\mathbf{H} = \phi(\mathbf{X}\mathbf{W}_1 + \mathbf{b}_1), \tag{3.19}$$

where $\mathbf{W}_1 \in \mathbb{R}^{F\times U}$ is the matrix of connection weights for a network with U hidden units, $\mathbf{b}_1 \in \mathbb{R}^U$ is a row vector of constant bias weights for each hidden unit and ϕ is our nonlinear transfer function, which will be further discussed in Section 3.4.1. We also use a subscript following a matrix, as in $\mathbf{W}_1$, to indicate that this is the weight matrix for the first layer. This notation will become useful as we begin to add layers to our network architectures.

The visible layer then computes a weighted sum of the outputs of the hidden layer,

$$\mathbf{V} = \mathbf{H}\mathbf{W}_2 + \mathbf{b}_2, \tag{3.20}$$

where $\mathbf{W}_2 \in \mathbb{R}^{U \times K}$ is the matrix of visible connection weights, $\mathbf{b}_2 \in \mathbb{R}^K$ is a row vector of bias weights and K is the number of classes. For classification problems, it is generally desirable for the network to output a matrix of estimated class membership probabilities, $\mathbf{P} \in \mathbb{R}^{M \times K}$, for each of the M segments and K classes. In order to achieve this, we apply a softmax function to the outputs of the visible layer so that

$$p_{i,c} = \frac{e^{v_{i,c}}}{\sum_{j=1}^{K} e^{v_{i,j}}}, \tag{3.21}$$

where $v_{i,c}$ and $v_{i,j}$ are elements of the matrix $\mathbf{V}$. This ensures that the network outputs sum to one for each segment, i.e., across the columns of $\mathbf{P}$, and we can then interpret each value $p_{i,c}$ to be the estimated probability that segment i belongs to class c. The final class label for segment i is then

$$L_i = \underset{c=1,\ldots,K}{\operatorname{argmax}}\, p_{i,c}. \tag{3.22}$$

In order to tune the weights of our networks, we follow a supervised learning approach that minimizes a loss function for a labeled set of training examples. We have chosen to use the relatively standard log loss function [80],

$$E(\mathbf{w}) = -\frac{1}{MK} \sum_{i=1}^{M} \sum_{c=1}^{K} g_{i,c} \log p_{i,c}, \tag{3.23}$$

where $\mathbf{w}$ is a vector containing all of the network weights and where our target outputs are

$$g_{i,c} = \begin{cases} 1, & \text{if segment } i \text{ belongs to class } c \\ 0, & \text{otherwise.} \end{cases} \tag{3.24}$$

Minimizing log loss is equivalent to maximizing the likelihood that the outputs of our classifier match the class labels of our training examples [80]. It is also convenient that log loss is smooth and differentiable, provided that our network transfer functions are as well.

In order to minimize our loss function, we utilize error backpropagation, which tracks contributions to the loss function backward through the network in order to compute first order derivatives [79, 80]. This gradient information is then used by an optimization algorithm, described in Section 3.4.3, to minimize $E(\mathbf{w})$. Although our implementation computes gradient information manually [105], many numerical software packages now compute first-order gradients automatically [106–108]. For this reason, and for the sake of brevity, we leave the derivation of error gradients as an exercise for the reader.

3.2 Time-Delay Neural Networks

As previously discussed in Section 2.1.3, TDE-based signal representations are more general than PSDs and are better suited for capturing some types of nonlinear, nonstationary and multivariate spatial patterns. In order to further explore these prospects, we propose a classification approach that combines TDE with ANNs. These networks, known as Time-Delay Neural Networks (TDNNs), closely resemble methods that we have successfully used to classify EEG signals in previous research [66]. This TDNN architecture, depicted in Figure 3.2, is similar to the ANNs that we have proposed for classifying PSDs in that it consists of two layers, i.e., a nonlinear hidden layer followed by a linear visible layer with a softmax readout. This architecture is different, however, in that its input features are a TDE of the raw EEG signal values instead of PSDs of the signal segments. As a result, our TDNNs assign a class label for each timestep of the signal, as opposed to a single class label for each signal segment. In Section 3.2.1, we will propose a label aggregation strategy that will allow us to reconcile this difference while also permitting additional flexibility for applications that require continuous control.

Assume that we have M EEG signal segments, $\mathbf{S}^i \in \mathbb{R}^{T \times N}$ for $i = 1, \ldots, M$, where T is the number of timesteps and N is the number of channels.[4] Following our previous notation for

[4] We describe the computations in a TDNN separately for each segment. In practice, however, these computations can be performed simultaneously for all segments using a three-axis tensor.

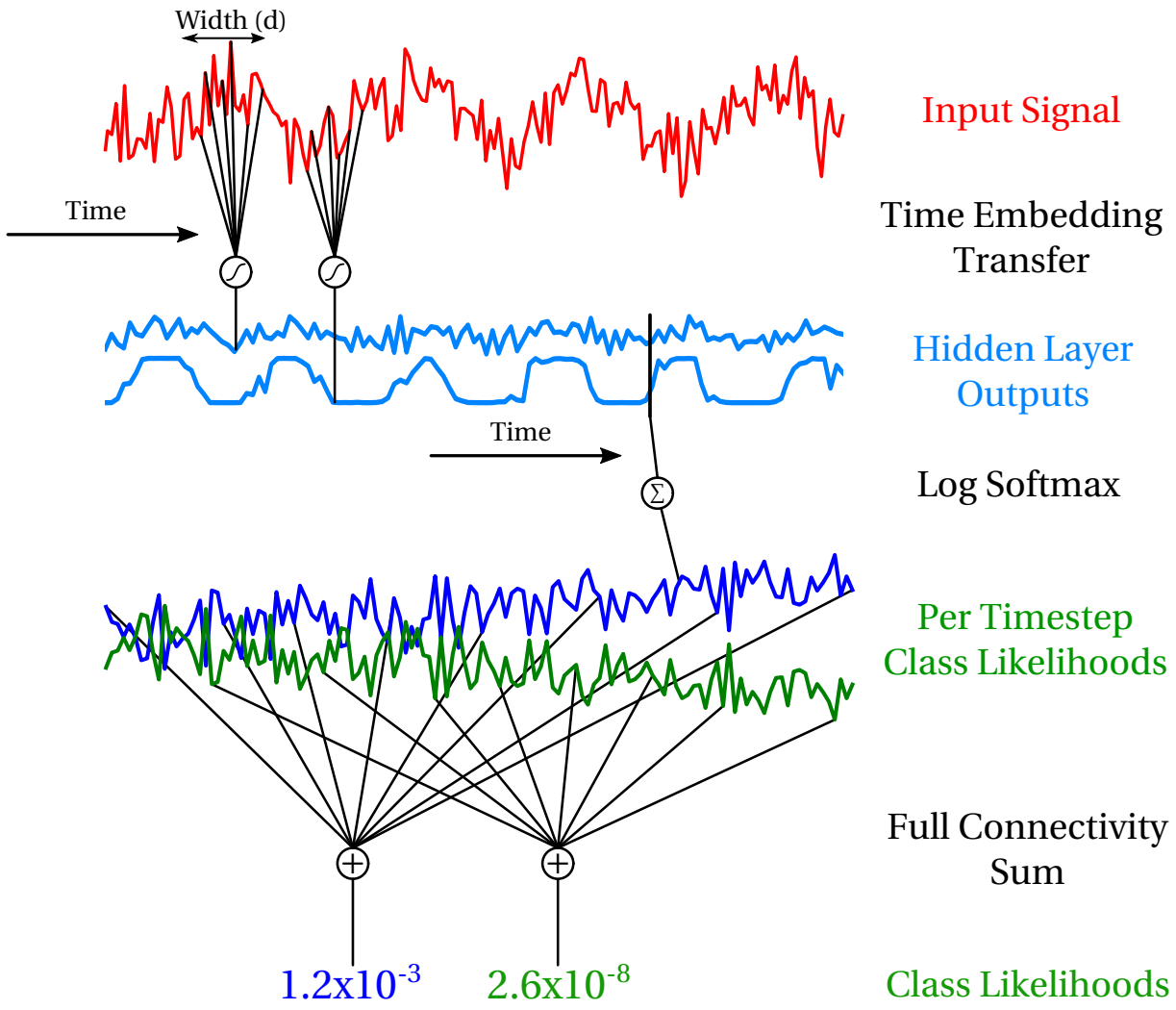


Figure 3.2: A schematic diagram of a two-layer Time-Delay Neural Network (TDNN). A window of raw signal values are first passed through a nonlinear hidden layer, which yields the hidden layer outputs. These outputs are then fed through a linear softmax layer which assigns a class label at each timestep. These labels are then aggregated over time to produce a single label for the entire segment.

ANNs, the output of the hidden layer for the i^{th} segment of a TDNN is then

$$\mathbf{H}^i = \phi(\mathscr{E}\{\mathbf{S}^i;\, d\}\,\mathbf{W}_1 + \mathbf{b}_1), \tag{3.25}$$

where $\mathscr{E}\{\mathbf{S}^i;\, d\}$ is the unary operator for time-delay embedding with embedding dimension d, as described in (2.18), applied to the i^{th} signal segment. Similarly, the output of the visible layer for segment i is then

$$\mathbf{V}^i = \mathbf{H}^i\mathbf{W}_2 + \mathbf{b}_2. \tag{3.26}$$

Next, we utilize a softmax function to estimate class membership probabilities. This process is different, however, in that we now have M probability matrices of size $(T-d+1)\times K$. Each of these probability matrices can be denoted $\mathbf{P}^i$ for $i = 1, \ldots, M$ and with values in $[0,1]$ that are computed according to the softmax function,

$$p^i_{t,c} = \frac{e^{v^i_{t,c}}}{\sum_{j=1}^{K} e^{v^i_{t,j}}}, \tag{3.27}$$

for signal segment i, timestep t and class c. The corresponding log loss function,

$$E(\mathbf{w}) = -\frac{1}{MK(T-d+1)} \sum_{i=1}^{M} \sum_{t=1}^{T-d+1} \sum_{c=1}^{K} g^i_{t,c} \log p^i_{t,c}, \tag{3.28}$$

is computed for each segment and for each timestep. We then use our gradient-based optimization routine, described in Section 3.4.3, to find values of $\mathbf{W}_1$ and $\mathbf{W}_2$ that minimize (3.28).

3.2.1 Label Aggregation

As we have previously discussed in Section 2.1.3, the fact that TDE-based approaches assign class labels for each individual timestep allows a great deal of flexibility when designing BCI applications. There is, however, a large discrepancy between the sampling frequency of an EEG signal, on the order of 256–1024Hz, and the rate at which a BCI user can reasonably be expected

to send instructions to the system, typically 0.5–1 instructions per second. As a result, TDE-based approaches generally require a method for slowing the rate at which the system makes decisions. The act of slowing the decision rate also provides an opportunity to increase the signal-to-noise ratio of the control signal, effectively increasing the accuracy of the decisions made by the system.

In order to achieve this, we propose a straightforward label aggregation strategy in which the class membership probabilities produced by our TDNNs are multiplied across each timestep in a fixed-width EEG segment. For each segment, this results in a single score for each class. The final predicted label is then the class associated with the highest score. The intuition behind multiplying the outputs of our TDNNs is to assume that these class membership probabilities are independent at each timestep and then to seek the joint probability that *all* timesteps belong to a given class. Although the assumption of independence is almost certainly violated in this setting, we have found this procedure to work quite well in practice.

When implementing this approach, multiplying probabilities can lead to a rapid loss of numerical precision as T grows large because the resulting values can become extremely small and vary widely across classes. In order to avoid this, we instead sum the log likelihoods, which is equivalent to the log of the product of probabilities. Since the log function is monotonically increasing, the class with the largest sum of log likelihoods will also be the class with the largest product of probabilities. Using this approach, the final class label for the i^{th} segment is

$$L_i = \underset{c=1,\ldots,K}{\operatorname{argmax}} \prod_{t=1}^{T-d+1} p_{t,c}^i \tag{3.29}$$

$$\equiv \underset{c=1,\ldots,K}{\operatorname{argmax}} \log \prod_{t=1}^{T-d+1} p_{t,c}^i \tag{3.30}$$

$$= \underset{c=1,\ldots,K}{\operatorname{argmax}} \sum_{t=1}^{T-d+1} \log p_{t,c}^i. \tag{3.31}$$

In our previous work, we have advocated for an evidence accumulation algorithm that builds confidence over the course of time in order to provide a stream of class labels with increased accuracy at a variable rate [66, 109]. Similar to the approach that we are proposing here, this

strategy assumes independence of the class labels assigned at each timestep and combines evidence over the course of time. In order to provide results that are comparable to our baseline methods, however, we have opted to use the above label aggregation strategy instead. When designing real-world BCI systems, a number of application-specific factors will likely be important considerations when choosing an appropriate label aggregation or evidence accumulation strategy. For instance, whether or not the application requires fixed-rate or near continuous feedback and potential tradeoffs between accuracy and responsiveness.

3.2.2 Time-Delay Embedding as a Discrete Convolution

A time-delay embedding followed by a matrix multiplication, which is a core operation in the hidden layer of our TDNNs, can be interpreted as a multivariate extension of the discrete convolution operator. In order to illustrate this, first consider the special case where our input signal consists of a single channel and a single segment and where our TDNN has only a single hidden unit. The input signal can then be expressed as the column vector

$$\mathbf{s} = [s_1,\, s_2,\, \ldots,\, s_T]^\mathsf{T} \tag{3.32}$$

where s_t is the signal value at time t and where T is the total number of timesteps. Similarly, the hidden weights of our network can be written as

$$\mathbf{w} = [w_1,\, w_2,\, \ldots,\, w_d]^\mathsf{T} \tag{3.33}$$

where w_i is the i^{th} weight and where d is the embedding dimension. Omitting the constant biases and before the application of our transfer function, ϕ, the hidden layer of our TDNN performs a time-delay embedding followed by a matrix multiplication, as described in (3.25).

These operations can be written as

$$\mathbf{z} = \mathscr{E}\{\mathbf{s};\, d\}\,\mathbf{w} = \begin{pmatrix} s_1 & s_2 & \dots & s_d \\ s_2 & s_3 & \dots & s_{d+1} \\ s_3 & s_4 & \dots & s_{d+2} \\ \vdots & \vdots & \ddots & \vdots \\ s_{T-d+1} & s_{T-d+2} & \dots & s_T \end{pmatrix} \mathbf{w}, \tag{3.34}$$

where $\mathbf{z}$ is a column vector of size T corresponding to outputs of the hidden unit at each timestep before the transfer function is applied.

If we then consider the value of $\mathbf{z}$ at each timestep t and let $t' = t + \lfloor d/2 \rfloor$, we can see that

$$z_t = [s_t,\, s_{t+1},\, \dots,\, s_{t+d-1}]\,\mathbf{w} = \sum_{i=1}^{d} s_{i+t-1}\, w_i \tag{3.35}$$

$$= \sum_{i=1}^{d} s_{i-\lfloor d/2 \rfloor + t' - 1}\, w_i = (\mathbf{s} \circ \mathbf{w})_{t'}, \tag{3.36}$$

where z_t is the t^{th} value in the column vector $\mathbf{z}$ for $t = 1, 2, \dots, T - d + 1$, $\circ$ denotes the discrete convolution operator and (3.36) is the definition of the univariate discrete convolution operator of the signal $\mathbf{s}$ with the kernel $\mathbf{w}$ at time t'. From this result, it is clear that the univariate case of TDE followed by a matrix multiply is equivalent to a discrete convolution with a constant shift of $\lfloor d/2 \rfloor$ timesteps. Note that this constant shift in time is insignificant, provided that our class labels are aligned appropriately, because it is applied across the entire signal. In the case when we apply the direct definition of discrete convolution, the labels output by our network must also be delayed by half the width of the convolutional window in order to account for the fact that we cannot look into the future, i.e., we cannot process the EEG signals until the time that they are actually recorded. It then follows that each hidden unit in a TDNN with univariate inputs is equivalent to a discrete convolution of the network weights with the input signal, followed by the application of our nonlinear transfer function.

When using CNNs for image processing, the convolution operator is typically generalized in two ways. First, the convolution slides across two axes: the horizontal and vertical axes of the image. This allows the CNN to learn shift-invariant representations across both axes of the images while utilizing relatively few weights within the local receptive fields of the convolutional windows. Second, the weighted sum is extended to include connections to all color channels. If, for example, the image has three color channels, then the number of weights is increased by three fold to incorporate a connection to each pixel in each color channel. This allows the network to incorporate information from across color channels without achieving shift invariance across color space. This makes intuitive sense because there are typically very few color channels and because patterns across color channels typically have a relatively fixed interpretation at each location in an image.

When working with EEG signals, one can also imagine approaches where convolution is applied spatially, across channels, as well as across time. Upon close examination, however, attempting to apply spatial convolutions for EEG signals is not straightforward to achieve and may not be appropriate for several reasons. First of all, EEG channels are physically laid out with unequal spacing over the surface of the scalp. In order to perform spatial convolutions, the convolution operator would be have to be further generalized to work in this setting. Although one potential approach is to apply spatial projections and interpolation to represent the signal as a series of images, this approach would introduce artificial information through interpolation and would further increase the dimensionality of our inputs. Second, EEG systems have relatively few spatial locations. While images often consist of millions of pixels, EEG systems typically have only 8–64 spatial channels. This limits the potential benefit of applying spatial convolutions, especially as the width of our convolutional kernel approaches the number of channels in the EEG system. Finally, we are primarily interested in achieving time invariance and not necessarily spatial invariance. Although we must achieve time invariance in order to account for the fact that our signals are not time locked, the sensors in our EEG system are placed in fixed locations. This is analogous to having registered images in computer

vision, where the content of each image is placed in a standard location within the field of view. Although it may be possible to learn some types of spatially invariant patterns in EEG systems with many channels, the fact that our EEG system has few channels that are placed in fixed locations means that spatial convolutions are not necessary.

With these points in mind, the generalization of the convolution operator that we will use in our networks has fixed connectivity across channels, similar to the way that color channels are typically handled in CNNs for image classification. Each of these convolutional units can then be interpreted a single-axis convolution across time with a window that also spans all of the EEG channels. In other words, the output of each artificial neuron, before the application of our transfer function, is a sum of convolutions, each with different weights, across all channels. As it turns out, this is exactly equivalent to our time-delay embedding operation followed by a matrix multiply.

In order to illustrate this, let $\mathbf{S}$ now be a $T \times N$ matrix representing a multivariate signal segment, where T is the total number of timesteps in our EEG segment and N is the number of channels. Also, let $\mathbf{W}$ now be an $(N \cdot d) \times u$ matrix of weights where each row is associated with the signal values in our time-embedded matrix of signal values and each column corresponds to an artificial neuron in our network. For notational simplicity, also let $\mathbf{M}^n$ be the $d \times u$ submatrix spanning the rows of $\mathbf{W}$ from $(n-1)d+1$ to nd for $n = 1, 2, \ldots, N$. In other words, $\mathbf{M}^n$ is the submatrix of $\mathbf{W}$ that corresponds to the weights associated with the time embedding for the n^{th} channel and can be written as

$$\mathbf{W} = \begin{pmatrix} \mathbf{M}^1 \\ \mathbf{M}^2 \\ \vdots \\ \mathbf{M}^N \end{pmatrix}. \tag{3.37}$$

Our multivariate generalization of the convolution operator, denoted by the symbol $\odot$, at time t and for the j^{th} hidden unit can then be expressed as a sum of discrete convolutions and

then as a time-delay embedding followed by a matrix multiply,

$$(\mathbf{S} \odot \mathbf{W})_{t,j} = \sum_{n=1}^{N} (\mathbf{s}_{\cdot,n} \circ \mathbf{m}^n_{\cdot,j})_{t'} = \sum_{n=1}^{N} \sum_{i=1}^{d} s_{i+t-1,n} w_{(n-1)d+i,j} = (\mathscr{E}\{\mathbf{S};\, d\}\, \mathbf{W})_{t,j}, \tag{3.38}$$

for $t = 1, 2, \ldots, T-d+1$ and $j = 1, 2, \ldots, u$ and where $\mathbf{s}_{\cdot,n}$ is the n^{th} column vector in $\mathbf{S}$ and $\mathbf{m}^n_{\cdot,j}$ is the j^{th} column vector in $\mathbf{M}^n$. The full matrix form for all hidden units and timesteps can then be written as

$$(\mathbf{S} \odot \mathbf{W}) = \mathscr{E}\{\mathbf{S};\, d\}\, \mathbf{W}. \tag{3.39}$$

From (3.39) and (3.25), we can see that our TDNNs can be interpreted as single-layer convolutional networks, where the convolution operator is generalized to span channels and where our convolutional window slides only across time. In Section 3.3, we will describe several approaches for further generalizing this approach into multilayer CNNs.

3.2.3 Finite Impulse-Response Filters

In addition to our insight that the hidden layers of our TDNNs perform convolutions, it is also valuable to notice that these layers can be interpreted as Finite Impulse-Response (FIR) filters. In the field of signal processing, linear FIR filters are commonly used to attenuate given frequency ranges of a signal, i.e., FIR filters are one technique for implementing lowpass, highpass and bandpass filters [110]. FIR filters leverage the fact that a convolution in the time domain is equivalent to a pointwise multiplication in the frequency domain. If it is known a priori how the power spectrum of the signal should be transformed via a multiplication with a window function, then the corresponding time-domain filter can be designed by taking the Fourier transform of the window function and then applying the convolution operator with the time-domain signal.

In the case of our TDNNs, the hidden layer can be viewed as a nonlinear FIR filter, i.e, a convolution, followed by a nonlinear transfer transfer function and pooling. Instead of designing these filters analytically, however, the weights of our convolutional kernels are tuned

through our training procedure. Although interpreting these nonlinear filters is considerably more challenging than interpreting a purely linear FIR filter, examining the Fourier transform of our convolutional kernels can lead to valuable insights. Specifically, the Fourier transform of the convolutional kernel yields the frequency response of each artificial neuron before the transfer function is applied. If, for instance, we notice that the frequency response is high for a given frequency range and low for another, this may indicate that the network views the frequencies with a high response as more important or as containing patterns that are more sharply nonlinear than frequencies with low responses. In Section 4.3.3, we will perform this type of analysis by examining the frequency responses of the convolutional kernels learned by our TDNNs.

3.3 Deep Convolutional Networks

Since the artificial neurons in the hidden layer of a TDNN perform a multivariate generalization of a discrete convolution, it follows naturally that these layers can be stacked and pooling can be introduced between layers in order to construct a type of deep CNN. The output matrix for the first layer in such a CNN with L convolutional layers is then

$$\mathbf{H}_1^i = \psi(\phi(\mathbf{S}^i \odot \mathbf{W}_1 + \mathbf{b}_1)), \tag{3.40}$$

where ψ is our pooling function, ϕ is our transfer function, $\mathbf{S}^i$ is our matrix of signal values for the i^{th} segment, $\mathbf{W}_1$ is our matrix of network weights for the first layer and $\mathbf{b}_1$ is a row vector of constant bias weights for the first layer. Note that we use subscripts under $\mathbf{H}$, $\mathbf{W}$ and $\mathbf{b}$ to denote the layer index and superscripts over $\mathbf{H}$ and $\mathbf{S}$ to denote the segment index.[5] The $L-1$ subsequent layers then perform a convolution on the outputs of the previous layer. For layers $\ell = 2, \ldots, L$, the outputs are then

$$\mathbf{H}_\ell^i = \psi(\phi(\mathbf{H}_{\ell-1}^i \odot \mathbf{W}_\ell + \mathbf{b}_\ell)). \tag{3.41}$$

[5] In practice, all segments can be processed simultaneously by using tensors with three axes.

This multilayer convolutional architecture may have a number of advantages over single-layer TDNNs. In addition to the ability to learn time-invariant representations, stacking convolutional layers may encourage these networks to learn hierarchical representations with increasing levels of abstraction. The change of scale introduced by pooling may also encourage the network to learn patterns at different time scales across layers. Deep CNNs also have a higher ratio of nonlinearities to free parameters. Since our nonlinear transfer function is applied to each hidden unit, a network with narrow convolutional kernels and many layers tends to have more applications of the transfer function for each free parameter. This allows CNNs to learn sophisticated nonlinear patterns with a relatively small number of free parameters.

From the perspective of learned FIR filters, stacked convolutional layers can also be viewed as cascading filters that are capable of achieving increasingly more fine-grained filtering characteristics as the signals pass through each layer. The duration of the network's impulse response, and therefore the potential duration of the its memory, also increases additively when convolutional layers are stacked. Note that the impulse response of the first convolutional layer is determined by the width of its convolutional kernel because each hidden unit receives inputs from a window with a width of d_1 timesteps. In subsequent layers, however, the impulse response lasts for the width of its convolutional kernel following the end of the impulse response of the previous layer. This means that stacked convolutional networks can achieve long-term memory by stacking convolutional layers with relatively narrow convolutional kernels. Since pooling increases the time scale in subsequent convolutional layers, it also increase the impulse response of our networks. The effects of stacking convolutional layers and pooling on the impulse response and memory capacity of our networks will be further explored in Section 3.4.2.

In the following sections, we will propose several CNN architectures that will utilize these stacked convolutional layers. We will also describe a method for transfer learning that utilizes data from across multiple subjects. For future reference, Table 3.2 provides a high-level summarization of all of the network architectures that we will explore. Each of these designs will be described in detail as they are introduced.

Table 3.2: Summary of CNN architectures.

Abbr.	Name	Readout Layer	Transfer Learning
TDNN	Time-Delay Neural Network	Label Aggregation	No
TDNN-TR	Time-Delay Neural Network	Label Aggregation	Yes
CNN-FC	Convolutional Neural Network	Fully Connected	No
CNN-LA	Convolutional Neural Network	Label Aggregation	No
CNN-TR	Convolutional Neural Network	Label Aggregation	Yes

3.3.1 Fully Connected Readout Layers

Now that we have described our approach for constructing convolutional layers and stacking them in order to form deep networks, we must establish methods for constructing readout layers that are capable of transforming the outputs of our convolutional layers into class labels that can be used by a BCI system. One sensible approach for constructing readout layers utilizes fully connected layers, also called dense layers, which are similar to the readout layers commonly used in CNNs designed for image classification. We will refer to this network architecture as Convolutional Neural Networks with Fully Connected readout layers (CNN-FC).

In Figure 3.3, we see a schematic diagram illustrating the connectivity in this type of network. After the input signal for the i^{th} EEG segment is processed by one or more convolutional layers, the output of the final convolutional layer, $\mathbf{H}_L^i$, where L is the total number of convolutional layers, is flattened into a single row vector, $\mathbf{h}_L^i$, of size $T_L u_L$, where T_L is the number timesteps in the final convolutional layer, after pooling, and u_L is the number of convolutional units in the last convolutional layer. $\mathbf{h}_L^i$ is then passed to a layer of hidden units which again apply our nonlinear transfer function. Since all timesteps and the outputs of all hidden units at our final convolutional layer are passed into this hidden layer, it is said to be fully or densely connected. The outputs of this hidden layer are then passed to a linear visible layer, after which a softmax function is applied in order to generate a class membership probability for the entire EEG segment. Note that the hidden and visible readout layers in this architecture are identical to the ANN architecture described in Section 3.1.3, except in the way that the initial inputs are

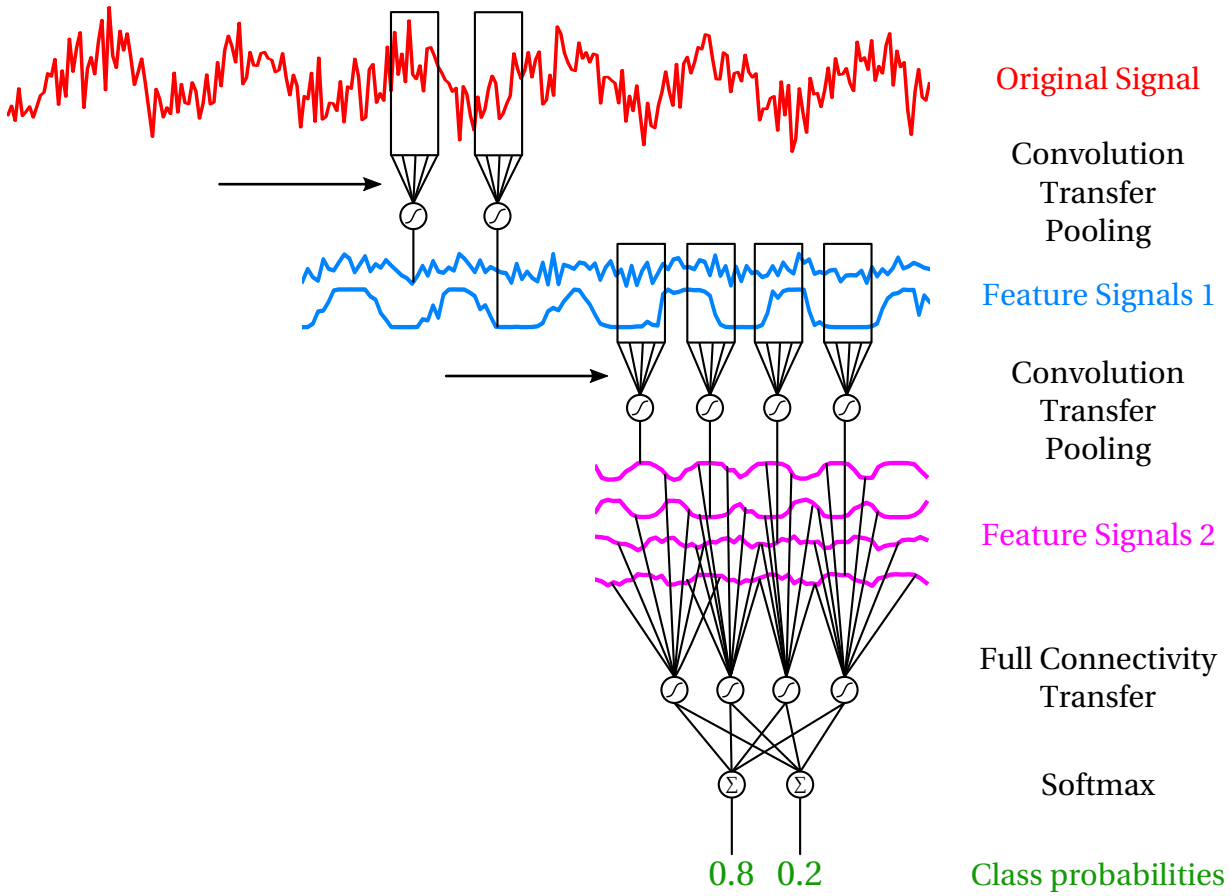


Figure 3.3: A schematic diagram showing an example of our proposed CNN architecture with fully connected readout layers (CNN-FC). After the input signal is processed by one or more convolutional layers, the network activations proceed through a fully connected layer where all channels and all timesteps are concatenated into a single row vector and passed to each nonlinear unit. The outputs of the first fully connected layer are then passed through a linear softmax function that outputs class probabilities for the entire EEG segment.

generated. This architecture is also the single-axis equivalent of the fully connected readout layers that we have described for image classification in Section 2.2.2.

Although the use of fully connected readout layers gives the network the ability to incorporate information from across the entire time span of each EEG segment, this approach also has several important limitations. First of all, fully connected readout layers may involve a large number of connections and, therefore, free parameters to tune, which may result in overfitting. This is because each timestep and hidden unit in our final convolutional layer is connected to each hidden unit in the first readout layer. Although the number of timesteps in our final convolutional layer could be reduced considerably by using many layers with pooling, this approach can also lead to a large number of free parameters due to the increased number of layers in the network. This tradeoff will be empirically explored in Section 4.1.3 and Section 4.1.4. The use of fully connected readout layers also assumes that information across the entire time span of the EEG segment is useful and, if so, that we have enough EEG segments that the network will be able to identify and utilize these longer-term patterns without overfitting the data. Given that our segments are not time locked and our data is noisy and undersampled, utilizing longer-term information across entire segments may not be achievable. Finally, the use of fully connected readout layers does not allow for continuous control or variable decision rates because class labels are assigned for each segment at fixed-width intervals.

3.3.2 Label Aggregation Readout Layers

An alternative approach for constructing our readout layers utilizes the same label aggregation strategy that we described for use with TDNNs in Section 3.2.1. Since we have already shown that the hidden layer of a TDNN is equivalent to a single convolutional layer, it follows naturally that our label aggregation strategy should also apply to multilayer CNNs. We will refer to this network architecture as Convolutional Neural Networks with Label Aggregation readout layers (CNN-LA).

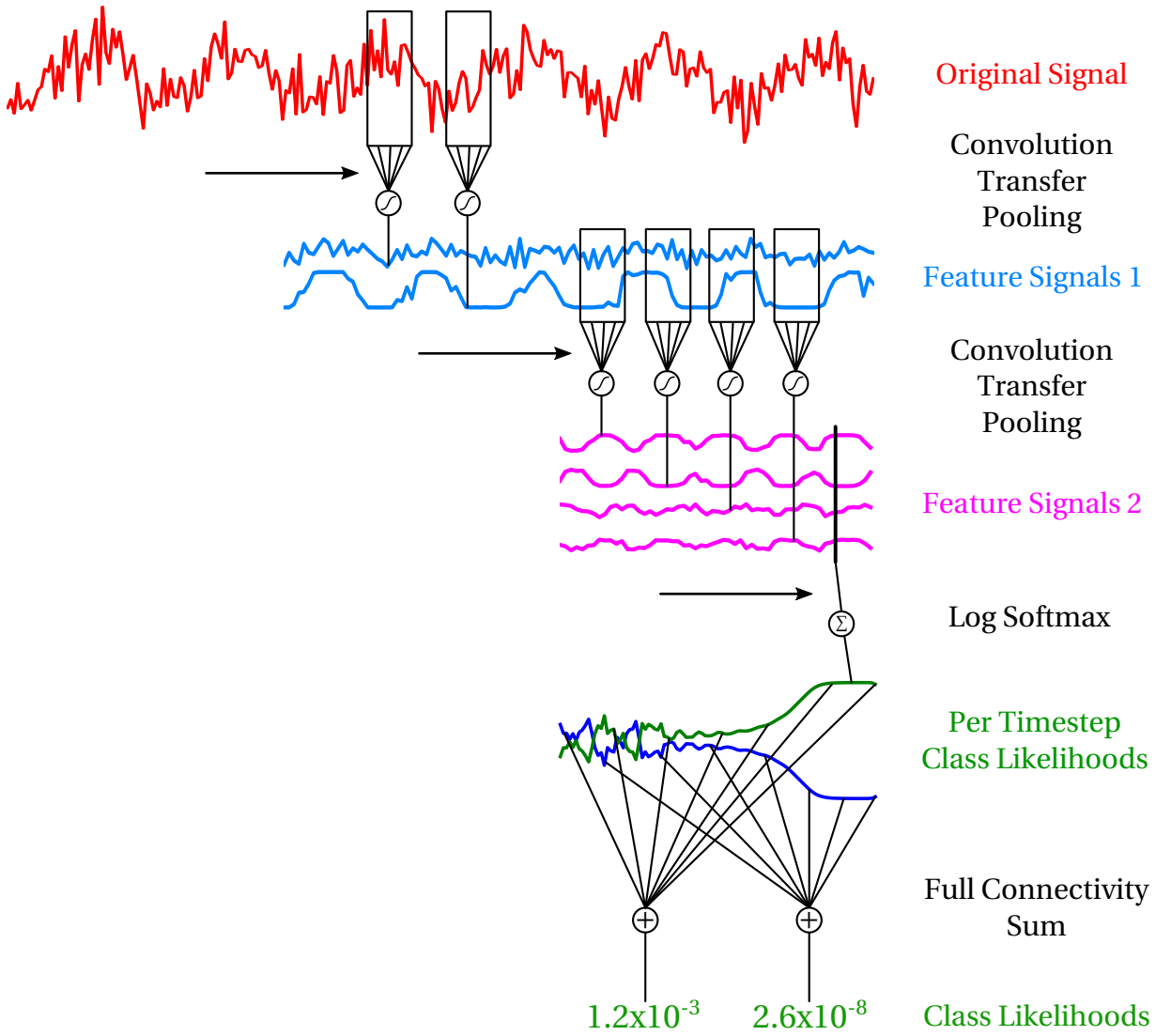


Figure 3.4: A schematic diagram showing an example of our proposed CNN architecture with label aggregation readout layers (CNN-LA). After the input signal is processed by one or more convolutional layers, the network activations are passed to a linear log-softmax layer that assigns a class label to each individual timestep. These log likelihoods are then summed, which is equivalent to multiplying the probabilities. The final label for the entire segment corresponds to the maximum of these summed likelihoods.

In Figure 3.4, we see a schematic diagram showing the connectivity in this type of network. After the input signal for a given EEG segment is processed by one or more convolutional layers, the output of the final convolutional layer is passed to a linear visible layer followed by a softmax function. Since the output of the final convolutional layer is not flattened, this layer outputs a prediction of class membership for each timestep output by the final convolutional layer. Note that if pooling is applied following some of the convolutional layers, there will be fewer timesteps after the final convolutional layer than there were in the original input signal. We then sum the $\log_{10}$ transform of these probabilities, which is equivalent to multiplying the probabilities, in order to produce class membership scores for the full EEG segment. This process is exactly identical to the readout layers that we have already described for our TDNNs in Section 3.2.1.

Unlike fully connected readout layers, label aggregation readout layers have a more limited ability to capture long-term temporal information. Although CNN-LA is capable of capturing temporal information through it's convolutional layers and by building evidence across all of the labels across the EEG segment, it does not have the ability to capture fixed or nonlinear patterns that span the entire segment. On the other hand, label aggregation readout layers may have far fewer free parameters to tune than fully connected layers because they do not have a separate connection to each timestep. As we have previously described in Section 3.2.1, CNN-LA also allows for additional flexibility when designing BCI applications because it permits continuous control, evidence accumulation and variable decision rates. Unfortunately, however, exploring the intricacies of these types of control signals is beyond the scope of the present work.

It is important to note that a number of other research groups have also explored CNN architectures that do not incorporate fully connected readout layers in several contexts, including image classification, image segmentation and BCIs [47, 52, 111–114]. This architecture is sometimes referred to as a fully convolutional network and is conceptually similar, although somewhat different in practice, to networks that utilize global average pooling layers.

3.3.3 Transfer Learning

In Section 2.2.4, we discussed how transfer learning might be useful for addressing problems with noise, undersampling and overfitting by leveraging training data from across multiple subjects. In order to explore this possibility, we will perform a series of experiments where we first train our networks using data from other subjects and then perform a fine-tuning step using the data for the subject at hand. In a real-world setting, the initial model could be provided beforehand and then the fine-tuning step could be performed using a brief calibration phase performed before using the BCI system. When incorporating transfer learning, we will only investigate label aggregation readout layers and we will refer to our network architectures as Time-Delay Neural Networks with Transfer Learning (TDNN-TR) and Convolutional Neural Networks with Transfer Learning (CNN-TR).

When exploring this transfer learning approach, we will follow a leave-one-subject-out approach where the data from all other subjects is first used to train the initial network and then the data for the current subject is used for fine-tuning the model. If J is a set containing all of our subjects and r is the current subject, then our initial model will be trained using the combined training data for $J - \{r\}$ and fine-tuning will be performed using only the training data for r. The final test performance is evaluated using the test data for r. This procedure is then repeated for all subjects.

During the initial training procedure, we first perform a fixed number of training iterations over the combined dataset where the number of iterations is a hyperparameter that will be selected empirically in Section 4.1.5. The model is then fine-tuned using the training data for the current subject by starting with the connection weights from the initial training stage and then performing additional training iterations that further update the weights in all layers of the network. Our automatic early stopping procedure for regularization, which is discussed in Section 3.4.4, is performed using cross validation during the fine-tuning process using only the training data for only the current subject. This procedure may improve the generalization performance of our networks by allowing more data to be leveraged, provided that data from

across subjects is useful for classification and that this knowledge can be retained through the fine-tuning stage.

3.3.4 Optimizing Input Sequences

In addition to interpreting the convolutions in our CNNs as FIR filters, as previously described in Section 3.2.3, another approach for analyzing the types of patterns that these networks learn to utilize is to examine input sequences that perform well. An EEG segment that produces a given class label with a high degree of confidence is likely to contain the types of patterns that the network has learned to leverage for that particular class. If we compare and contrast segments that perform well, or poorly, for each class, it may be possible to identify the types of patterns that the network is relying upon. It can be difficult, however, to visually sift through many EEG segments and, given the large amount of noise and confounding variables that influence EEG signals, it is rare to find a single segment that produces a given class label with a high degree of confidence. In order to overcome these challenges, we propose that instead of examining actual EEG segments in the dataset that perform well, we should instead use optimization techniques to search for an artificial *ideal* EEG segment that produces each class label with a high degree of confidence.

In order to see how this can be achieved, first let $\mathbf{S} \in \mathbb{R}^{N \times T}$ represent a raw, unprocessed input segment with N channels and T timesteps. Then, let

$$\mathrm{F}(\mathbf{S}) = \mathit{Filter}(\mathbf{S}) \tag{3.42}$$

denote the application of our preprocessing and filtering procedures to $\mathbf{S}$, which include a 1–80Hz bandpass filter and a 60Hz notch filter as will be described in detail in Section 3.5.2. We can then describe the forward pass of our CNN to a filtered segment as

$$\mathrm{P}(\mathbf{S}; \mathbf{w}) = \mathit{CNN}(\mathrm{F}(\mathbf{S}); \mathbf{w}), \tag{3.43}$$

which results in a $T \times K$ matrix of estimated class membership probabilities for each of the K classes and the T timesteps in $\mathbf{S}$. The network is parameterized by the weights $\mathbf{w}$ that are found during the training process and are now fixed. We can then pose a minimization problem,

$$\min_{\mathbf{S}} \; -\sum_{t=1}^{T} \log \mathrm{P}(\mathbf{S}; \mathbf{w})_{t,c}, \tag{3.44}$$

that seeks to find an artificial input segment that maximizes the likelihood of being labeled as belonging to a given class label c, as described by (3.31).

It is important that we do not allow our optimization procedure to learn a completely arbitrary input segment because realistic EEG signals in our dataset are filtered to primarily contain frequencies between 1–80Hz and with 60Hz noise removed. For this reason, we have incorporated our filtering procedure into our optimization process and we can then use $\mathrm{F}(\mathbf{S})$ to describe a optimal input sequence for the network that abides by these constraints.

In order to solve this optimization problem, we use the ALOPEX algorithm as described by Unnikrishnan and Venugopal [115]. ALOPEX is a correlative learning algorithm that does not rely on gradient information and was originally proposed by Harth and Tzanakou for mapping the visual receptive fields of neurons in the visual cortices of animal brains [116, 117]. This algorithm was then later adapted for training artificial neural networks by Venugopal, Pandya and Unnikrishnan and has been explored in-depth by Bia as well as Chen and Haykin, et. al, [115, 118–120]. Note that we run ALOPEX for 100,000 iterations with a step size of 0.015 and perform a cooling update every 100 iterations. The results of this analysis will be presented along with our results in Section 4.3.5.

We have chosen to use ALOPEX in this context for several reasons. First of all, ALOPEX does not require the computation of gradient information. Although it is possible to compute the gradients of our network with respect to the input sequence and with fixed weights, this process is complicated by the fact that we would have to compute the gradients backward through our bandpass and notch filters. Furthermore, this would be a somewhat computationally expensive operation that does not appear to be necessary in this context. Since we only need to

optimize a single input sequence with a single set of fixed weights, the forward pass of our network is computationally fast. This allows us to run ALOPEX for many iterations in a short period of time while also achieving good results. Finally, it seems appropriate that we use ALOPEX to find optimal input signals for our networks since it was originally designed to search for optimal stimuli for eliciting responses in biological neurons, which is conceptually similar to this problem except that we are now investigating the responses of artificial neural networks.

3.4 Neural Network Details

Until now, we have described only the general architectures of the artificial neural networks that we plan to explore. In this section, we will describe additional details about the way that our networks are initialized and trained, including standardization, transfer and pooling functions and the optimization algorithm we use to tune the networks' parameters. The goal of this section is to describe and justify each of these design decisions and also to ensure that our experiments are reproducible.

3.4.1 Initialization and Transfer Function

All of the neural network architectures that we will explore follow the guidelines described by Lecun, et al., for input standardization, initialization of network weights and choice of transfer function [86]. The goal of these procedures is to ensure that our networks are initialized with reasonable parameters, achieve rapid convergence during our training procedure and have a transfer function that is appropriate for modeling the patterns in our EEG signals.

First, we shift and scale our input data so that each dimension has zero mean and unit variance using the means and variances extracted from all combined classes in the training partition. For our CNN and TDNN networks, we standardize each EEG channel and for our PSD-based approaches we scale each frequency bin. Since we use all of the training data, relative differences across classes are preserved. We also use the same scaling constants for both our training and testing partitions in order to avoid extracting information from our test data.

We then initialize the connection weight matrix, say $\mathbf{W}$, for each artificial neuron by drawing from the random uniform distribution,

$$\mathbf{W} \sim U(-\sqrt{3/\kappa}, \sqrt{3/\kappa}), \tag{3.45}$$

where $U(a,b)$ denotes the random uniform distribution ranging from a to b and κ is the number of incoming connections, or fan-in, for this particular neuron, which is equivalent to the number of rows in $\mathbf{W}$. Note that the range of this distribution is chosen so that

$$\sigma = \kappa^{-1/2}, \tag{3.46}$$

where σ is the standard deviation of $\mathbf{W}$. Since the columns of our input matrix, $\mathbf{X}$, have zero mean and unit variance, $\mathbf{W}$ results in $\mathbf{H} = \mathbf{XW}$ with columns that roughly have zero mean and unit variance.[6]

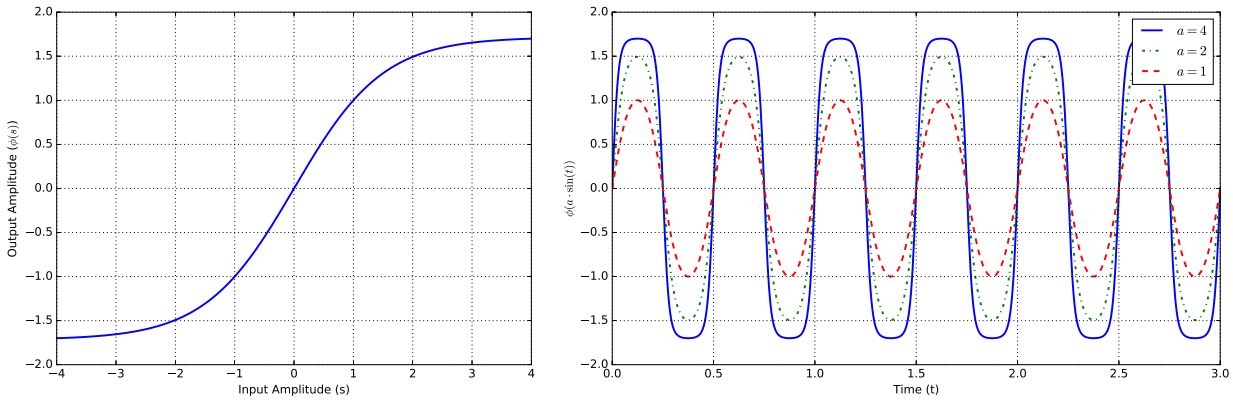
The nonlinear transfer function we use is

$$\phi(\mathbf{H}) = 1.7159 \, tanh\left(\frac{2}{3}\mathbf{H}\right), \tag{3.47}$$

where *tanh* is the element-wise hyperbolic tangent function and where $\mathbf{H}$ is the matrix of pre-transfer outputs for a given hidden layer. Note that ϕ is smooth, continuous and differentiable, which are generally desirable when using gradient-based optimization techniques.

In Figure 3.5, we explore the shape of ϕ and the effect that its application has on its inputs. In Figure 3.5a, we see the output of $\phi(s)$ for $s \in [-4,4]$. Note that ϕ has a sigmoidal shape and has a symmetric effect on positive and negative inputs. It is also nearly linear and has roughly unit gain within $[-1,1]$ but rapidly becomes nonlinear outside this range and ultimately squashes its inputs onto $[-1.7159, 1.7159]$. In Figure 3.5b we the effect of applying ϕ to sinusoids with

[6] Although we use the notation for our simple feedforward networks here, the same principals apply to our CNN architectures.



(a) $\phi(s)$. **(b)** $\phi(a \cdot \sin(t))$ for $a = 1, 2, 4$.

Figure 3.5: The application of ϕ to sinusoids with different amplitudes. (a) The sigmoidal shape of ϕ is nearly linear in $[-1, 1]$ and then increasingly nonlinear with outputs ultimately being squashed onto $[-1.7159, 1.1759]$. (b) The application of ϕ to a sine wave results in increasingly nonlinear behavior for larger amplitudes until nearly binary switching dynamics are achieved.

varying amplitudes. For sine waves with an amplitude of less than one, the signal is largely unchanged. For signals with higher amplitudes, however, the response becomes increasingly nonlinear in both the positive and negative directions until nearly binary switching dynamics are achieved. If our convolutions, which can be viewed as FIR filters, learn to amplify the signal so that it enters this nonlinear range, then the neuron will respond in a nonlinear way. On the other hand, if the outputs of these FIR filters remains near the linear range, the signal will pass unchanged. Similarly, any frequency ranges that are attenuated by these FIR filters will remain unpassed.

When our standardization and weight initialization procedures are combined with ϕ, the initial configuration of our network results in outputs at each layer that have approximately zero mean and unit variance. Keeping the scale of the outputs at each layer the same helps to ensure that the weights at each layer are tuned at similar rates during the training procedure. Preventing the outputs from growing larger as they pass through each layer of the network also helps to prevent our transfer function from becoming saturated because outputs that are far on the tails of ϕ have a nearly zero gradient, which can result in slow training convergence. Keeping our transfer function nearly linear, i.e., in $[-1, 1]$, also causes our initial configuration to have

nearly linear activation at each layer. This allows the network to learn nonlinearities during optimization rather than having arbitrary nonlinearities introduced at initialization. Note that we have explored the use of several potential transfer functions in pilot experiments, including the unscaled hyperbolic tangent and rectified linear transfer functions, and have found that this transfer function generally works best and yields small and computationally efficient networks.

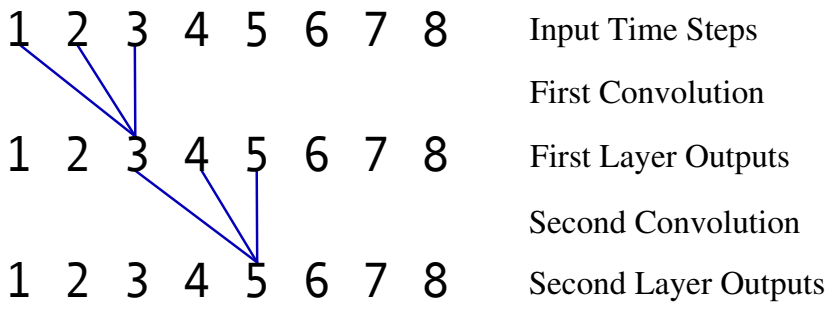
3.4.2 Pooling

In our experiments, we will examine two potential configurations for pooling. First, we will examine networks that do not incorporate pooling at all, i.e., where ψ is the identity function. Additionally, we will examine the 2:1 average pooling function. If we let $\mathbf{Z}$ be the $T-d+1 \times u$ matrix of outputs resulting from out convolution and transfer operations, then $\psi(\mathbf{Z})$ is then a $(\lfloor (T-d+1)/2 \rfloor) \times u$ matrix where the element in the i^{th} row and j^{th} column is

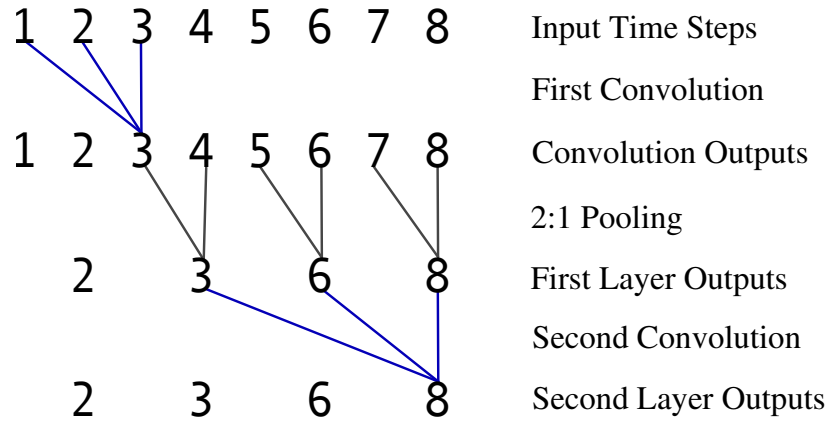
$$\psi(\mathbf{Z})_{i,j} = \frac{1}{2}(z_{2i,j} + z_{2i-1,j}). \tag{3.48}$$

Although a variety of other pooling functions have been proposed for use in CNNs [45, 121], we have found that the averaging pooling function works well while also being computationally efficient. The average pooling function also performs antialiasing because it acts as a moving average filter that attenuates frequencies above $\frac{1}{2}$ of the Nyquist rate. This reduces the potential need for our networks to learn antialiasing filters and also simplifies interpretation by attenuating aliasing artifacts.

Introducing pooling into our network architectures can potentially improve performance in two ways. First, pooling acts as a change of scale between convolutional layers. This may encourage the network to learn both shorter-term, higher-frequency patterns as well as longer-term, lower-frequency patterns. Second, pooling increases the duration of the impulse response of our networks, which permits a higher memory capacity without increasing the number of connections and free parameters in our network. In Figure 3.6 we see examine the duration of the impulse response of two convolutional layers which both have a convolutional



(a) Impulse response duration without pooling.



(b) Impulse response duration with pooling.

Figure 3.6: An illustration of how 2:1 pooling affects the duration of the impulse response of two convolutional layers that both have a convolutional kernel of width three. (a) Without pooling, the duration of the combined impulse response of both layers is $3+3-1=5$ timesteps since the second convolution can respond up to the tail of the first convolution. (b) when 2:1 pooling is introduced between layers, the duration of the impulse response increases to $3+2\cdot3-1=8$. This is because the output of the first convolutional layer is decimated two-fold, which effectively doubles the duration of its impulse response with respect to the original input signal. Note that the number of connections that are tuned during training, shown in blue, is the same for both network configurations.

width of three. In Figure 3.6a, we see that the duration of the impulse response is $3+3-1=5$ timesteps because a convolution has an impulse response duration equal to its width and because the subsequent layer can respond up to the tail of the response from the previous layer. In Figure 3.6b, we see that the duration of the impulse response after introducing 2:1 pooling between the layers increases to $3+2\cdot3-1$ timesteps of the original input signal because the output of the first convolutional layer is decimated two-fold before the second convolution is applied. This effectively doubles the length of impulse response of the second convolutional layer. There

is, however, also some information loss incurred during the pooling process that may limit the ability of the network to capture more granular shorter-term information. In Section 4.1, we will examine these trade-offs in detail.

3.4.3 Scaled Conjugate Gradients

Training a neural network to perform a given task generally involves an iterative procedure for optimizing the connection weights of the network so that it learns to map the example inputs to the desired outputs. More formally, we seek to minimize our loss function, $E(\mathbf{w})$, over our training data where $\mathbf{w}$ is a column vector containing the connection weights of our network. Due to the nonlinear and high-dimensional nature of ANNs and because there are many potential symmetries across neurons,[7] $E(\mathbf{w})$ tends to be nonlinear and nonconvex. As a result, closed-form solutions for finding the optimal connection parameters in ANNs generally do not exist and iterative optimization techniques are required. Since, ANNs are typically differentiable, through an approach known as backpropagation, gradient-based optimization techniques can be used to guide and accelerate the optimization process [79, 80].

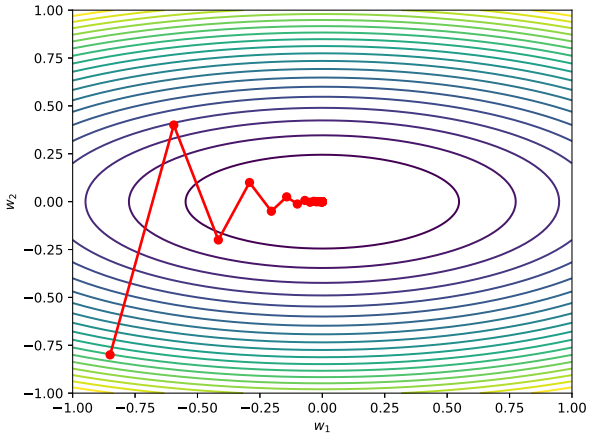
In general, the i^{th} iteration of such an optimization technique can be described as

$$\mathbf{w}_{i+1} \leftarrow \mathbf{w}_i + \alpha \mathbf{d} \tag{3.49}$$

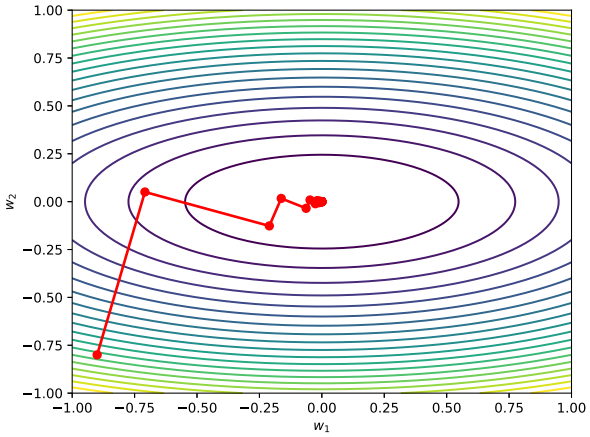
where α is a scalar step size and $\mathbf{d}$ is a vector indicating the direction of the step. The value of $\mathbf{d}$, and potentially α, can be determined using first-order gradient information in a process broadly referred to as gradient descent.

Steepest Descent (SD) is a relatively straightforward gradient descent algorithm where α is an empirically selected hyperparameter, known as the learning rate, and where $\mathbf{d}$ is the error

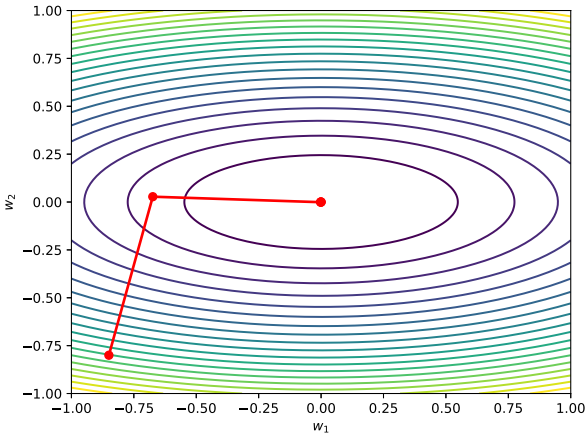
[7] Swapping all of the connection weights between two artificial neurons is a trivial demonstration that multiple configurations exist that can produce identical outputs.



(a) Steepest descent with learning rate



(b) Steepest descent with line search



(c) Scaled Conjugate Gradients

Figure 3.7: Examples of gradient descent algorithms minimizing the quadratic function $w_1^2 + 5w_2^2 = 0$. The search begins at the point $(-0.9, -0.8)$ and the global minimum is at $(0,0)$. (a) Steepest descent with a simple learning rate can have a step size that is too large in steep regions and too small in flat regions. It also tends to alternate in a zigzag pattern toward the function minimum. (b) Steepest descent with line search finds an optimal step size. It can, however, still follow a zigzag pattern toward the minimum. (c) Scaled conjugate gradients searches for appropriate step sizes and also reduces zigzagging by selecting conjugate search directions.

gradient, i.e., the vector of first-order partial derivatives,

$$\mathbf{E}'(\mathbf{w}) = \nabla E(\mathbf{w}) = \begin{bmatrix} \frac{\partial E}{\partial w_0} \\ \frac{\partial E}{\partial w_2} \\ \vdots \\ \frac{\partial E}{\partial w_N} \end{bmatrix}. \tag{3.50}$$

Although SD can work well in some cases, it often suffers from poor convergence rates and may fail to converge on a solution altogether. For instance, if the learning rate is too large then encountering a steep region of the error surface can cause SD to rapidly move far away from an appropriate solution. This problem is sometimes referred to as the exploding gradient problem. If, on the other hand, the learning rate is too small, then SD can make very slow progress over plateaus on the error surface. These types of plateaus are common in deep and recurrent networks due to the fact that the magnitudes of error gradients tend to decrease rapidly as they pass through multiple layers of a network, known as the vanishing gradient problem [83]. Since the error surfaces for ANNs may contain both steep and flat regions, ordinary SD is often insufficient. In Figure 3.7a, we see an example of ordinary SD minimizing a two-dimensional quadratic function. Note that early iterations, near a relatively steep region of the quadratic function have a step size that is too large and overshoots the optimal step size. Near the function minimum, at the point $(0,0)$, the step size then becomes too small, taking several iterations to converge on the minimum.

A variety of approaches for improving convergence can be incorporated into SD. For example, the magnitude of the step size can be capped and a momentum term can be used to ensure that the step size does not abruptly change across a few iterations. Alternately, SD can be modified to utilize meta-search in order to establish the step size, as opposed to relying on a fixed learning rate [122]. In this approach, a simple search routine, e.g., line search, can be used to find a value of α that minimizes $E(\mathbf{w})$ along the current search direction.

Although utilizing a line search in order to establish our step size can eliminate some of the problems associated with relying on a fixed learning rate, this variant of SD has a tendency to move across valleys on the error surface in an alternating zigzag pattern. In Figure 3.7b, we see an example of SD with line search minimizing a quadratic function. Although the step sizes are now optimal, which is an improvement over SD with a fixed learning rate, this approach still moves toward the function minimum in an alternating zigzag pattern with multiple iterations moving in the same direction.

Conjugate gradient (CG) methods reduce this tendency to search along the same direction in different iterations by selecting conjugate search directions. Shewchuk, et al., describe CG methods thoroughly and in detail [122]. Somewhat informally, the intuition behind CG methods is to imagine that a locally quadratic approximation of our error surface is scaled to be shaped like a round bowl. Searching in orthogonal directions in this transformed space now allows us to ensure that we do not repeatedly search in the same direction. If we neglect the limits of numerical precision, assume that optimal step sizes are accurately found and assume that our error function is purely quadratic, then CG methods converge in less than N iterations where N is the dimensionality of our parameter vector, $\mathbf{w}$.

Scaled Conjugate Gradients (SCG), originally proposed by Møller for tuning the weights of ANNs [80,85], further refines the CG method by replacing the line search with a quadratic function that can be minimized in each search direction. By utilizing a second gradient evaluation to estimate second-order gradient information, SCG fits a quadratic polynomial to the error surface in each successive search direction. The inflection point of this polynomial can then be found analytically in order to determine the appropriate step size. In Figure 3.7c, we see an example of SCG minimizing the same two-dimensional quadratic function used in our previous examples. In this case, SCG does not move in a zigzag pattern and avoids searching in the same direction in multiple iterations. Note that SCG also converges in roughly two iterations for this two-dimensional, quadratic minimization problem.

Although the error surfaces for neural networks are typically not purely quadratic, a locally quadratic approximation of the error surface generally appears to work well and, in our experience, SCG usually provides rapid convergence to good solutions for a variety of neural network architectures. These observations align with other works that have explored variants of the CG algorithm for training deep networks, including the popular Hessian-Free algorithm, which was proposed for use in neural network optimization by Martens and Sutskever and is quite similar to SCG [88, 90, 123].

It is worth noting that optimization strategies that utilize Stochastic Gradient Descent (SGD) are currently popular for optimizing the weights of deep neural networks [79, 86, 123, 124]. In SGD, the loss function during each iteration is computed using only a small, semi-random subsample of the training data, called a minibatch. Over a number of iterations, all training examples are typically used. SGD-based approaches have the advantage of being computationally convenient for large datasets that cannot be fit into volatile memory and may be less likely to converge on local minima because the introduction of stochastic behavior leads to additional local exploration of the error surface. SGD may not, however, be appropriate for small or moderately sized datasets that can easily fit in memory and where each minibatch is likely to consist of a relatively large portion of the training data.

In the current setting, computational performance, for both training and prediction, is important in order to allow interactive use of the BCI system. Furthermore, our training data is relatively undersampled and can easily fit into volatile memory. As a result of these observations and our experience using various optimization algorithms to tune the weights of our proposed network architectures, we believe that SCG is the best candidate. In all subsequent experiments, we will use SCG for weight optimization.

3.4.4 Regularization

In order to help prevent our neural networks from overfitting the training data, i.e., from fitting noise, artifacts and other patterns that do not generalize well, we utilize the early stop-

ping regularization technique [79, 80]. Early stopping limits the complexity of our networks and, therefore, helps to prevent overfitting by limiting the number of optimization iterations performed during the training procedure. Since our networks are initialized with small weight values and are designed to behave nearly linearly in their initial configurations, early termination of the optimization procedure prevents the connection weights from growing large while simultaneously limiting the nonlinear behavior of our networks.

In order to achieve this, our SCG optimization algorithm is first configured to run for either 2,500 iterations or else until the change in the loss function function falls below 1×10^{-8} between two consecutive iterations. During this first optimization pass, classification accuracy is also computed over the validation partition every 20 iterations, unless otherwise noted. This process is repeated over all of our validation folds, as described in Section 3.5.4. Finally, the network is retrained, using the same initial weight configuration, for the number of iterations that achieved the best mean validation performance. Note that this early stopping procedure is repeated for each participant and network configuration because there can be different levels of noise and nondiscriminative patterns for different subjects and because changes to network configuration can alter the model's tendency to overfit. The outcomes of our experiments with regularization are explored in detail in Section 4.1.

3.5 Data and Implementation

Now that we have thoroughly described the methods behind our classification algorithms, we will continue by describing the details of our experimental procedures. This will include our implementation, dataset and participants as well as our methods for preprocessing and evaluating the performance of our classifiers. To the extent possible, we have made these components publicly available in order to provide transparency and reproducibility.

3.5.1 The Colorado EEG and BCI Laboratory Software

All of our algorithms, experiments and data collection procedures are implemented within the framework of the Colorado EEG and BCI Laboratory version 3 ($CEBL_3$). $CEBL_3$ is an open-source and publicly available software package that is produced and provided by our research laboratory, the Colorado State University BCI Lab [14, 125].[8,9] $CEBL_3$ is written primarily in the high-level Python programming language and is heavily integrated with NumPy and SciPy for numerical methods, matplotlib for visualization and wxPython for its graphical user interface.

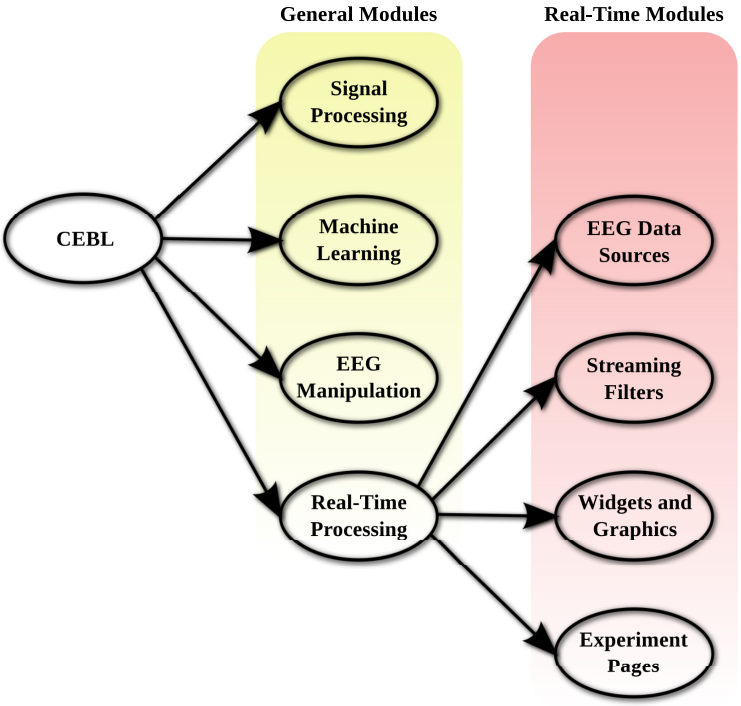


Figure 3.8: Layout of software modules in $CEBL_3$. This design allows the engineer to first develop offline prototypes using only the general modules and then promote successful prototypes to an interactive BCI using the real-time modules.

$CEBL_3$ is intended to support rapid prototyping of BCI technologies and end-to-end support for all phases of BCI research and development. In order to achieve this, $CEBL_3$ is divided into two tiers, as depicted in Figure 3.8. In a typical workflow, a researcher would begin their work in

[8]The homepage for $CEBL_3$ is located at https://www.cs.colostate.edu/eeg/main/software/cebl3

[9]We used tagged version 3.1.0 available at https://github.com/idfah/cebl.

a Python script or Jupyter notebook using only the modules from the first tier. These modules provide a broad range of functionality that is generally useful EEG analysis and developing novel BCI technologies, including data manipulation, signal processing, visualization and machine learning.

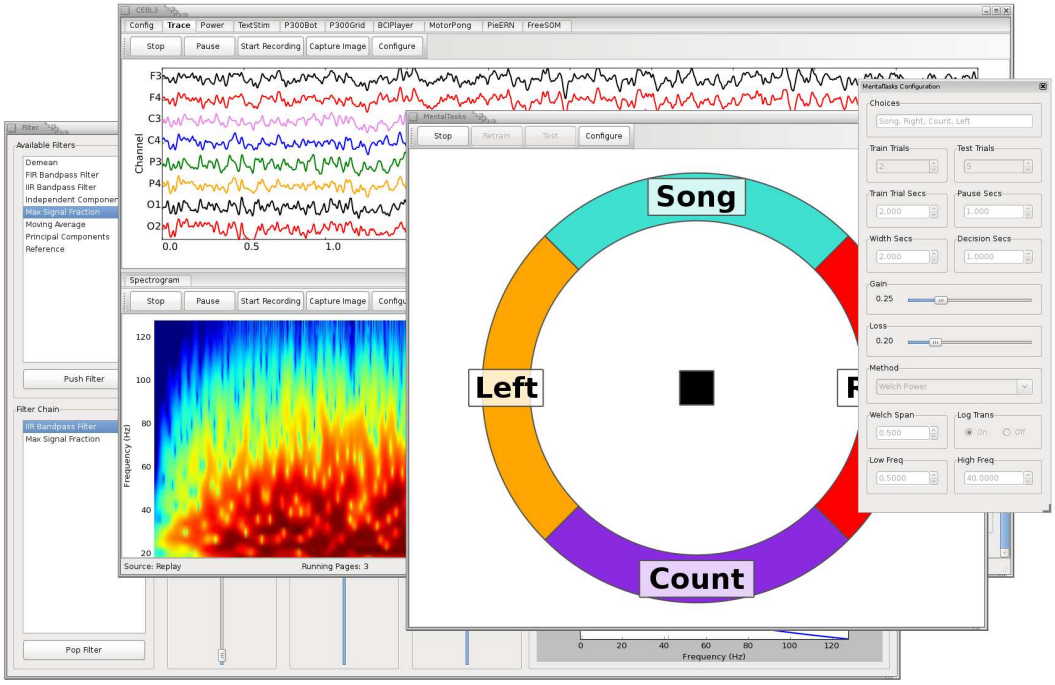


Figure 3.9: A screen capture of the CEBL$_3$ graphical user interface (GUI). The GUI has a notebook-style layout that allows for various windows to be detached and viewed across multiple displays.

Methods that prove to be successful can then be promoted to the second tier, which is designed for creating real-time, interactive BCI systems. The second tier includes a fully functional and extensible graphical interface, shown in Figure 3.9, and includes a variety of tools that are useful for creating interactive BCIs. This includes modules for real-time data acquisition from several EEG hardware devices, widgets for common BCI paradigms and a framework for configuring filtering pipelines.

All of our classifiers, including our neural networks, are available in CEBL$_3$. These implementations make heavy use of vectorization, in-place memory operations and manual gradient derivations in order to achieve good computational performance. We also use a build of

NumPy that is linked against Intel's Math Kernel Library (MKL), which we have found to greatly improve computational performance on Intel microprocessors [126]. These optimizations allow us to train each of our neural network architectures in less than ten minutes and runtime inference is easily fast enough for use in interactive BCIs, without relying on specialized neural network hardware or Graphics Processing Units (GPUs) for acceleration.

3.5.2 EEG Acquisition System

All of the data used in our experiments were recorded using the g.MOBILab+ EEG acquisition system with g.GAMMASys active electrodes, manufactured by Guger Technologies, also known as g . tec. The g.MOBILab+ EEG system is small, portable and relatively affordable, making it well suited for practical BCI applications. This system provides eight active EEG electrodes and a sampling rate of 256Hz per channel with a bandwidth of 0.5–100Hz at −3db attenuation. The specifications of this EEG system are summarized in Table 3.3. We chose to place the eight channels at sites F3, F4, C3, C4, P3, P4, O1 and O2, as previously shown in Figure 1.2, in order to cover a broad area of the cortex for each hemisphere of the brain.

Table 3.3: Specifications of the g.tec g.MOBILab+ with g.GAMMASys EEG acquisition system.

Manufacturer	Guger Technologies
Model	g.MOBILab+
Electrode System	g.GAMMASys
Channels	8
Sampling Frequency	256Hz
Passband	0.5–100Hz
Active Electrodes	yes
Reference	right earlobe
Electrode Material	Ag/AgCl
Communication	Bluetooth
Power Source	1.5V DC (AA)
Approx. Cost (USD)	11,000

Following data acquisition, we apply a 2^{nd} order bidirectional, zero phase Butterworth filter with a stopband of 59.5–60.5Hz at -3db power in order to attenuate 60Hz interference induced

by power mains and appliances. We then apply a similar 3rd order Butterworth filter with a pass-band of 1–80Hz in order to reduce both slow drift and higher-frequency components near the Nyquist rate of 128Hz. We have found that these filtering steps slightly improve performance for all of the methods explored here, especially for EEG signals that were recorded under realistic conditions. Given the large amount of noise encountered in real-world environments, we believe that these extra filtering steps are likely beneficial in BCIs used for assistive technology.

3.5.3 Participants and Mental Tasks

All of our experiments are conducted using a publicly available dataset that was collected by our laboratory in 2012 expressly for the purpose of evaluating methods for classifying EEG signals in MT-based BCIs.[10]. This dataset includes EEG recorded from 14 participants. The first 10 participants were drawn from a sample of convenience, i.e., university graduate students. These participants had no known medical conditions or impairments and data were recorded in the well-vetted Brainwaves Research Laboratory in the College of Health and Human Sciences at Colorado State University [127,128]. The remaining four participants had motor impairments causing varying degrees of paralysis: two had complete Spinal Cord Injuries (SCIs) at vertebrae C4 and two had progressive Multiple Sclerosis (MS). For these participants, EEG recording took place in their home environments in order to replicate realistic operating conditions. Note that two participants were excluded from the downloadable dataset, one because mental task data were not recorded for the g . tec system and the other because all trials were not completed.

Each participant was seated comfortably in front of a 20-inch LCD computer screen and asked to remain relaxed and move as little as possible during the experiments. They were then given instructions on how to perform four mental tasks following a cue on the monitor in the form of a single word, an example of which is shown in Figure 3.10. These tasks, which are summarized in Table 3.4, consisted of: silently singing a favorite song, imagine repeatedly making a left-handed fist, visualizing a cube rotating in three dimensions and silently counting backward

[10]Available for download at https://www.cs.colostate.edu/eeg/main/data/

Figure 3.10: A screen capture of the visual cue given to a subject prompting them to perform the Song task in CEBL$_3$.

from 100 by decrements of three. Each participant performed all four tasks in a randomized order for 10 seconds per task. A blank screen was presented for two seconds between each cue during which the subject was instructed to relax. This procedure, which we refer to as a trial, was repeated five times. This yielded 50 seconds of EEG data per mental task and a total of 200 seconds of data for each participant. The data were later split into two-second non-overlapping segments, yielding 25 segments per mental task and a total of 100 EEG segments per participant.

Table 3.4: Mental tasks used and cues shown to the participants.

Cue	Task description
Count	Silently count backward from 100 by threes.
Fist	Imagine repeatedly making a left-handed fist.
Rotate	Visualize a cube rotating in three dimensions.
Song	Silently sing a favorite song.

The mental tasks listed in Table 3.4 were selected through a process that involved reviewing previous literature as well as using our prior experience and intuitions about which mental tasks are easily performed and likely to generate different types brain signals. In the seminal work by

Keirn and Aunon and in followup studies by Anderson, et al., tasks involving rest, mathematical problem solving, visualization of rotating geometric figures, verbal letter composing and counting were used with a degree of success [33, 57, 58, 64, 65]. As is described by Keirn and Aunon, many of these tasks are known to elicit different types of brain signals, especially with respect to the specialization of the hemispheres of the brain [33]. Works by Galán, et al., and by Millán, et al., have achieved notable success by combining mental tasks, such as resting, visualization and verbal tasks, with motor imagery tasks, such as imagined left and right hand movement [34, 69]. Works by Friedrich, et al., have explored the pairwise performance of a number of mental tasks and have found that verbal word association, mathematical subtraction, visualization of rotating geometric figures and motor imagery were all among the best-performing tasks [36, 56]. Friedrich, et al., did, however, also note a large amount of inter-subject variability. Note that our Count task is similar to the mental subtraction tasks found in many of these studies, Fist is a specific type of motor imagery and Rotate is similar to geometric figure rotation. Although the Song task is rare in research by other groups, we have found it to be easy to perform and, since it involves silent singing, it may be similar to the verbal tasks found in other studies. In several previous studies, our research group has found these mental tasks generally work well [60–62, 66].

3.5.4 Validation and Performance Evaluation

We utilize several techniques to ensure that our models are trained and evaluated fairly. First, we split the data for each participant into an 80% training partition and a 20% test partition. The training partition corresponds to the first four trials while the remaining 20% corresponds to the final trial. We believe that this approach is reasonable because it parallels the general procedure that would be followed in a real-time BCI, i.e., the participant would first calibrate the system and then later use it interactively. While the training partition will be used for both training our models and for model selection, the final testing partition is completely withheld for use in our final evaluation results.

Each classification approach that we have proposed utilizes a single regularization hyper-parameter that is tuned on a per-subject basis. In order to select this parameter, we use a five-fold cross validation procedure. Since the training partition contains 20 segments for each of the four mental tasks, each fold of our cross validation then consists of $15 \cdot 4 = 60$ segments for training and $5 \cdot 4 = 20$ segments for validation. After this cross validation procedure is used to estimate the best regularization parameter, the model is trained over the entire training partition, which consists of $20 \cdot 4 = 80$ segments and, finally, the performance is recorded for the final $5 \cdot 4 = 20$ test segments.

When performing additional model selection and hyper-parameter tuning experiments, beyond the selection of our per-subject regularization parameters, we use the same cross validation procedure but for only the first five participants in the laboratory group. The mean validation performance is then averaged over all five of these participants. This procedure allows us to identify suitable model configurations while only using a small subset of the training data and participants. If we were to select our model configurations individually, it is possible that this procedure might overfit the training data, leading to poor generalization performance. If, on the other hand, we utilized the training data for all participants, our model configurations might be excessively tuned to the individuals in our study, leading to overly optimistic results.

In order to evaluate the performance of our models, we utilize three performance metrics. The first of these metrics is Classification Accuracy (CA), which conveys the percentage of segments that were correctly labeled,

$$100 \cdot \frac{\text{Number of correctly labeled segments}}{\text{Total number of segments}}. \tag{3.51}$$

CA is a valuable performance metric because it is intuitive and easy to interpret. CA also aligns closely with the user's perception of a BCI system's performance, i.e., the percentage of commands during which the system reacted correctly. Note that for our four-class problem, a CA of 25% corresponds to the expected performance of a random classifier.

CA does not, however, fully convey false positives versus false negatives and it does not describe the performance of each individual class, which are important factors to consider when gauging user experience, comparing models and choosing mental tasks. In order to examine these aspects of performance, we also present 4×4 confusion matrices where each element

$$c_{i,j} = 100 \cdot \frac{\text{Segments labeled as class } i \text{ that belong to class } j}{\text{Total number of segments in class } j}. \tag{3.52}$$

Confusion matrices allow us to examine the performance of each class separately and to identify which class-label combination are frequently confused by the classifier.

Although confusion matrices provide a detailed breakdown of performance, they do not explicitly account for the number of classes or for the rate at which class labels are assigned. Note that a BCI with fewer tasks or longer segments will likely have a higher CA than a BCI with many classes or shorter segments, even though the former provides fewer degrees of control and a slower response time. In order to account for these variables, we also present Information Transfer Rate (ITR). This performance metric, which was derived by Pierce and later adapted for use in BCI by Wolpaw, et al. [129, 130], expresses the amount of information, in bits-per-minute (bpm), that is transferred from the user to the BCI system. ITR can be written as

$$V\left(\log_2 K + A \log_2 A + (1-A) \log_2 \frac{1-A}{K-1}\right) \tag{3.53}$$

where $V = 30$ is the classification rate in decisions per minute, $K = 4$ is the number of classes and A is the fraction of correct decisions over the total number of decisions made. Although ITR can be difficult to interpret, it is valuable when comparing approaches across paradigms and it also encapsulates the rate at which a user is able to control the BCI.

Chapter 4

Results

Now that we have motivated our problem and thoroughly described our methods and data, we will continue by performing a series of concrete experiments and empirical analyses. First, we will tune each of our model hyperparameters and examine how of each our design decisions and regularization procedures impacts validation performance and generalization. These experiments will lead to a number of insights into each of our classification approaches and will yield suitable configurations for each of our models. We will then present the final test results achieved by each of our classifiers and perform a careful comparative analysis. These experiments will provide evidence that our proposed TDNNs and deep CNNs outperform PSD-based approaches and that transfer learning further boosts the performance for both of these network architectures. An examination of the tradeoff between classification accuracy and responsiveness will also support our claim that these methods are well-suited for use in real-time applications. Finally, we will perform a series of experiments that analyze the types of patterns that our classifiers learn to utilize. These experiments will examine frequency responses, layer-wise outputs and optimal input sequences. These results of these analyses will provide tangible insights into the inner workings of our networks as well as the structure of the patterns found in these EEG signals.

4.1 Model Selection

We will begin by performing a number of model selection experiments that are designed to establish effective model configurations and hyperparameters for each of our classification methods. First, we will explore a number of potential preprocessing steps and model configurations for our baseline classifiers. We then show how the performance of our CNNs progresses as we introduce various components of our network architecture, including multiple layers, pooling, label aggregation and transfer learning.

In order to ensure that our models are not excessively tuned to the dataset at hand, the experiments in this section are restricted to the validation performance for the first five participants, as described in Section 3.5.4. For experiments that involve broad hyperparameters or changes to our model architecture, we examine the average validation performance across all five of these participants. Since our regularization parameters are tuned on a per-subject basis, we will examine the performance of these parameters only for Subject-1, unless otherwise specified. We have found that the behavior of our regularization parameters for Subject-1 is typically representative of the other subjects and we will present the selected hyperparameters for all subjects along with our final test results in Section 4.2.

4.1.1 Power Spectral Densities

Recall that our baseline approaches first estimate the PSD of each signal segment, using the DFT and Welch's method, and then pass these features to a subsequent classification algorithm. The span parameter, ω, affects the smoothing, dimensionality and frequency resolution of our PSDs. Low values of ω result in a smooth and low dimensional representation while high values of ω result in increased resolution and dimensionality. In Figure 4.1, we see how our mean validation Classification Accuracy (CA) changes as we vary ω for each of our baseline classification algorithms, LDA, QDA and ANNs with 32, 64 and 96 hidden units. The regularization parameter for each method, which will be examined shortly, is individually tuned for each classifier, subject and value of ω. Also, note that $\omega = 2$ seconds is equivalent to using the raw DFT, since the length of our signal segment is two seconds, and results in $256 \cdot 8 = 2048$ features per segment. On the other hand, $\omega = \frac{1}{16}$ of a second results in only $8 \cdot 8 = 64$ features per segment.

From these experiments, it is clear that a moderately small span, between 0.25 – 0.5 seconds or 256–512 features, yields optimal CA across all of our classification algorithms. This suggests that Welch's method does, to some extent, help address noise, undersampling and high dimensionality. From this experiment, we can also see that models with fewer parameters often outperform larger ones. Note that with $\omega = 0.25$, LDA has 612 free parameters while an ANN with 64

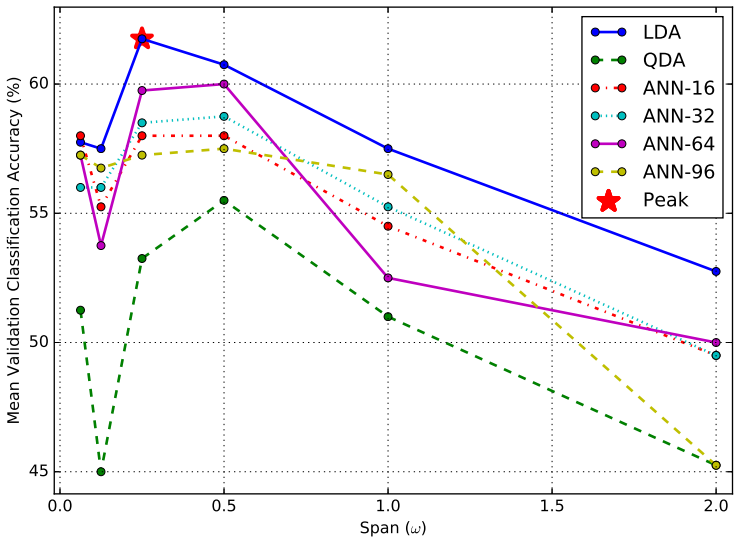
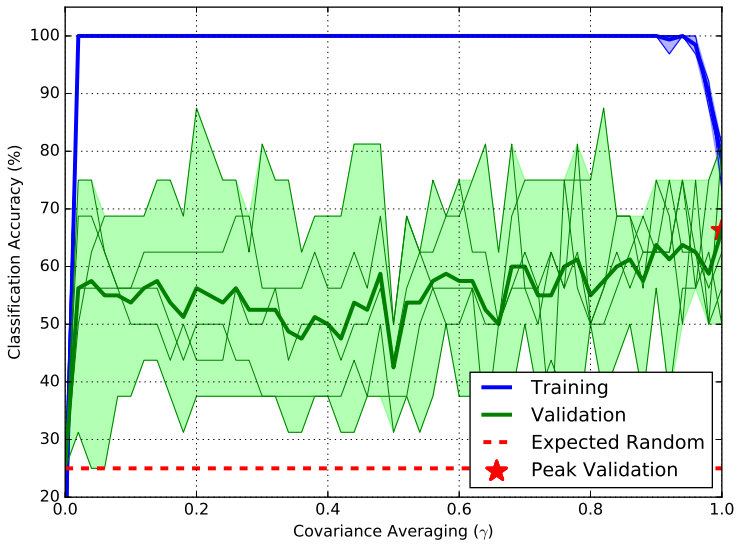


Figure 4.1: Mean validation CA versus span, ω, for each of our PSD-based classification approaches averaged across the first five participants. LDA achieves the highest mean validation performance with $\omega = 0.25$ and our ANNs with 64 hidden units outperforms smaller and larger networks with peak performance at $\omega = 0.5$.

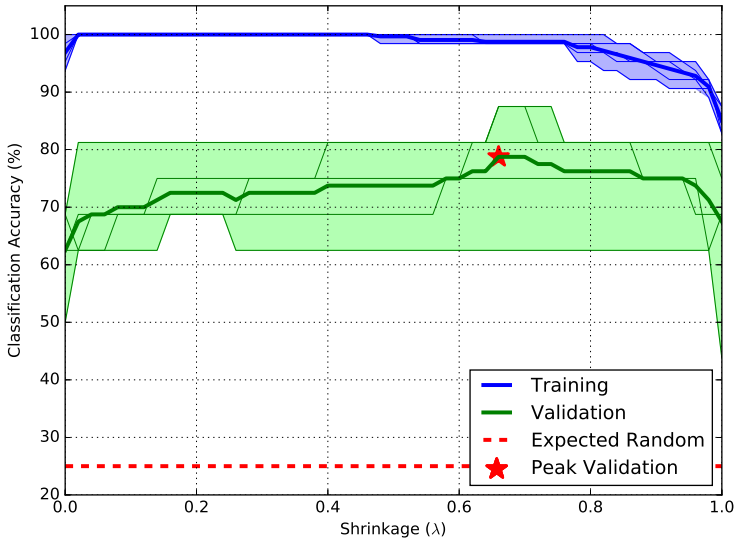
units has 10,052 and QDA has 58,145.[11] Note, however, that the peak performance for intermediate sized ANNs, with 64 hidden units, slightly outperform the smaller ANNs, with 16 and 32 hidden units. Although not shown here, we have also found similar results in experiments with two and four hidden units. This suggests that while there may be some nonlinearity present in these representations, LDA's robustness to overfitting may be more useful than an ANN's ability to form nonlinear class boundaries. Given the peak values found in Figure 4.1, future experiments will use a value of $\omega = 0.25$ for LDA and $\omega = 0.5$ for QDA and our ANNs. We will also fix the number of hidden units in our ANNs at 64.

Recall from Chapter 3 that QDA is regularized by mixing the class covariance matrices with the average covariance matrix and that LDA is regularized by mixing the average covariance matrix with a diagonal matrix. This results in a continuum ranging from QDA to LDA to a nearest-mean classifier. In Figure 4.2, we see how CA changes as these two regularization parameters are varied for Subject-1. Note that the thin lines in Figure 4.2 show the CA for each fold while

[11] QDA is parameterized by the triangular of the covariance matrices and mean vectors for all classes.



(a) QDA regularization example.



(b) LDA regularization example.

Figure 4.2: Training and validation CA versus our regularization parameters when using PSDs and discriminant analysis for Subject-1. The thin lines are the performance for each validation fold and the thick lines show the mean across folds and the shaded region is the range. (a) When varying covariance averaging for QDA, peak performance is achieved with $\gamma = 1.0$, which is equivalent to LDA. The training accuracy is near 100% for smaller values of γ, which suggests that overfitting is occurring. (b) When varying covariance shrinkage for LDA, peak performance is achieved with a moderate value of $\lambda = 0.66$. Again, the training accuracy near 100% for small values of λ, suggesting overfitting.

the shaded region represents the range and the thick line is the mean across all folds. Training accuracy is shown in blue while validation performance is shown in green.

This experiment leads to several interesting observations. First, we see that performance is near the expected random in the complete absence of regularization, i.e. QDA with $\gamma = 0.0$. Note that the inverse of the covariance matrix generally does not exist in this use of QDA and we instead rely on the Moore-Penrose pseudo inverse. As γ increases, our validation performance begins to rise while training CA remains at 100% until near $\gamma = 1.0$. This suggests that averaging the covariance matrices is not sufficient for regularizing QDA. In Figure 4.2b, we see the effect of moving the LDA covariance matrix from the class average toward a diagonal matrix via the regularization parameter λ. As λ increases, our validation performance continues to improve until our training CA drops below 100% CA at around $\lambda = 0.66$. As λ approaches one, the model appears to have insufficient complexity and both the training and validation performance begin to decrease. This suggests that, despite the use of Welch's method, our PSDs continue to be susceptible to overfitting and that our regularization process must generally move beyond averaging class covariances in order achieve effective regularization.

Recall that our ANNs are regularized using early stopping. By stopping the training procedure after a limited number of training iterations, early stopping prevents our ANNs from acquiring large weights and limits the nonlinear behavior of the model. In Figure 4.3, we see how the training and validation CA for Subject-1 varies as the number of training iterations increases. Note that for our PSDs, we measure validation performance every five iterations in order to improve computational performance. From this experiment, it is clear that our ANNs rapidly fit our PSD representations after about 30 iterations of SCG, at which point the training CA reaches 100% and validation CA levels off at about 80%. Numerical precision is also rapidly achieved during the training procedure, typically in less than 50 iterations.[12] The validation CA does not decrease on average, however, which suggests that the model does not catastrophically

[12]We consider the numerical precision of our networks to be reached when the difference in the loss function is less than 1.0×10^{-8} for two successive training iterations.

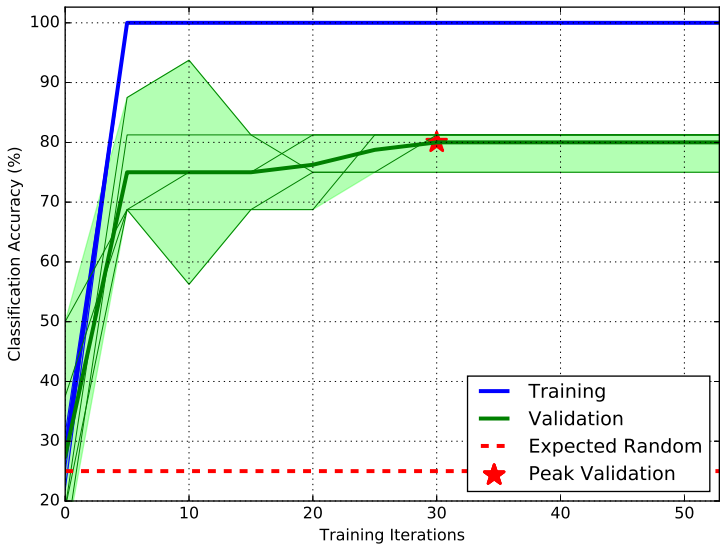


Figure 4.3: Training and validation CA versus training iterations for Subject-1 when using our PSD and ANN classifier. The thin lines are the performance for each validation fold and the thick lines show the mean across folds. Peak validation performance is achieved at around 30 training iterations. Training CA quickly rises to 100% after only five iterations while validation performance levels off after around 30 iterations. The ANN is able to fit all training examples before reaching numerical precision, suggesting that generalization may be limited by undersampling.

overfit noise in the training data. Instead, it appears that while Welch's method and our limited number of hidden units help to mitigate overfitting, the model is still able to rapidly find a class boundary that separates all of the training data perfectly. This suggests that our data continues to be undersampled for this model, which likely limits generalization performance.

4.1.2 Time-Delay Embedding Networks

There are three primary hyper-parameters to examine in our TDNNs: the number of hidden units, u, the embedding dimension, d, and the number of training iterations used in early stopping. In Figure 4.4, we see how the mean validation CA across the first five participants changes as we vary u and d. Note that early stopping is performed separately for each subject and value of u and d. From this experiment we can see that our optimal validation CA of 65.25% is achieved with $u = 52$ and $d = 7$. We can also see that our TDNNs are more sensitive to changes in d than to changes in u. Although the networks perform reasonably well for values of

u greater than about 30, performance quickly begins to drop as d grows above about 20. Note that the number of parameters in these models grows linearly with both d and u.

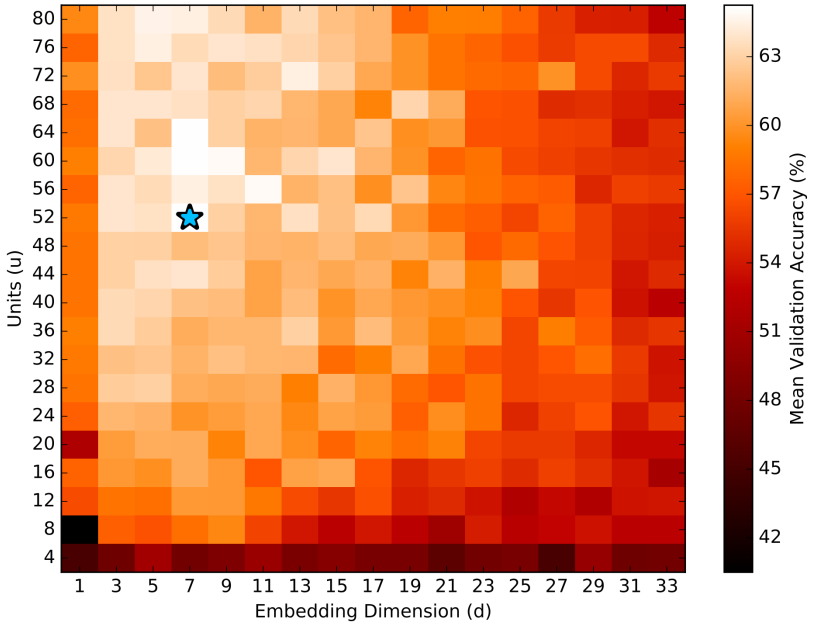


Figure 4.4: Mean validation CA versus hidden units, u, and embedding dimension, d, for TDNNs. Validation CA is averaged across the first five participants. Peak CA is achieved with $u = 52$ and $d = 7$. These networks appear to be more robust to changes in u than in d, possibly because wider FIR filters cannot be easily regularized using early stopping.

In order to explain why TDNNs may be more sensitive to changes in the embedding dimension than to the number of hidden units, consider our interpretation of each unit as an FIR filter, as described in Section 3.2.3. FIR filters with a higher order, i.e., a larger embedding dimension, are generally able to achieve sharper filtering characteristics and steeper roll-offs. In other words, larger values of d allow more selective, finer-grained response to different frequencies while smaller values of d are forced to respond to broader ranges of frequencies. We suspect that the addition of more hidden units can, to some extent, be counter-acted by stopping the training procedure before the response of the units becomes very nonlinear, i.e., early stopping is good at controlling the nonlinearity of the model. On the other hand, a large embedding dimension may lead to filters that are easily able to select overly specific frequency bands.

Since even linear FIR filters are able to achieve this, early stopping may not be very effective at preventing the model from fitting excessively narrow frequency bands.

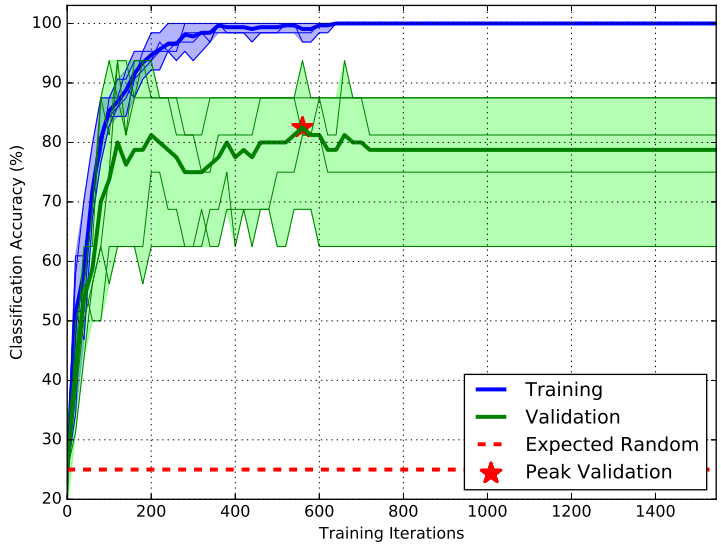


Figure 4.5: Training and validation CA versus training iterations for Subject-1 when using TDNN. The thin lines are the performance for each validation fold and the thick lines show the mean across folds. Peak validation performance is achieved at around 575 training iterations. Training CA climbs relatively slowly, achieving 100% accuracy around 400 iterations. Only a minor decline is seen in mean validation CA after it's peak, suggesting only slight overfitting.

In Figure 4.5, we see how our early stopping procedure affects training and validation CA when using a TDNN for Subject-1 with $u = 52$ and $d = 7$. Note that we measure CA every 25 iterations. The mean training accuracy climbs steadily before reaching 100% CA around 400 iterations while the mean validation accuracy reaches its peak value of 82.50% CA at 575 iterations. The validation accuracy then falls slightly before leveling off at around 700 iterations, at which point the numerical precision of our loss function is typically reached. TDNNs generally require considerably more training iterations than the ANNs we have examined for classifying PSDs. This suggests that the error surface is more sophisticated for these networks and that finding a class boundary that perfectly separates all of the examples in the training data is not as easily achieved. Also, notice that the validation accuracy continues to improve after the training accuracy nears 100% and that the validation CA drops only slightly after reaching

it's peak value. These observations suggest that TDNNs are less susceptible to undersampling and overfitting than our PSD-based approaches. This can be partially explained by the fact that our TDNNs assign class labels at each timestep using only a narrow window of the signal as the network input. This results in a larger number of smaller training examples and a higher ratio of training examples to model parameters. Since there are 512 timesteps in each our of signal segments, our TDNNs effectively have 512-fold more training examples. Our best-performing TDNN configuration also has only 3,176 model parameters, which is fewer than $\frac{1}{3}$ the number of parameters in the ANNs we examined for classifying PSDs. Provided that the narrower input windows used by our TDNNs are sufficient to capture the necessary patterns in the EEG signals, this approach may be advantageous over PSD-based methods due to its increased robustness to noise, undersampling and overfitting.

4.1.3 Convolutional Networks

When exploring our TDNNs, we have been primarily interested in the width of our convolutional windows and the number of artificial neurons within a single hidden layer. In other words, we have controlled the complexity of our models by adjusting the size of a single convolutional layer in a shallow network. A fundamental idea behind CNNs is that we should instead focus on controlling the complexity of our networks by adding multiple layers to form a *deep* network. Along these lines, our next experiment explores the effect of stacking multiple layers in our CNN-FC architecture. As described in Section 3.3.1, this architecture closely resembles the CNN architectures commonly used in image classification and consists of a number of convolutional layers followed by a hidden, fully connected readout layer containing nonlinear units and then a visible readout layer containing linear softmax units. In the following experiments, we will use 10 hidden units in our nonlinear readout layer because we have found this to be a small but effective size for this layer. In practice, we have found that our CNN-FC networks are relatively insensitive to this parameter for reasons that will become clear over the course of this section. In order to favor depth, we keep our convolutional layers simple by setting with

width of our convolutional kernel to a minimal $d = 3$ and by keeping the number of hidden units in our convolutional layers, u, the same across layers. Note that the networks in our initial experiments do not incorporate pooling, but we will explore this possibility shortly.

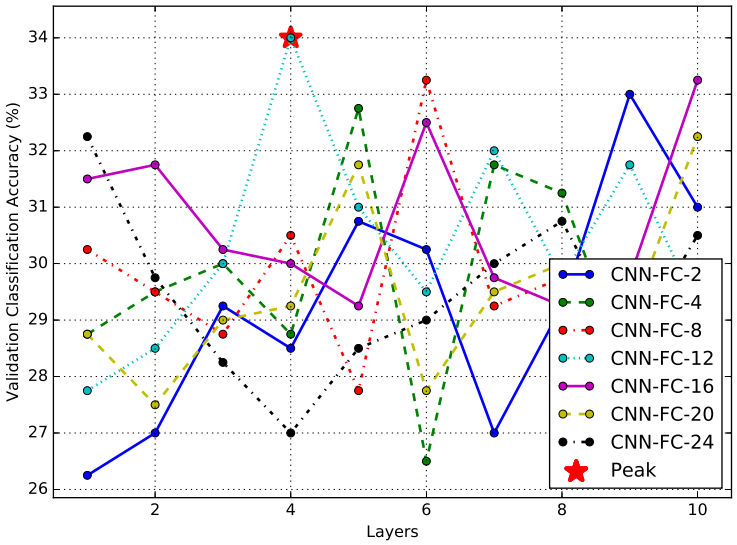


Figure 4.6: Mean validation CA, averaged across first five subjects, versus number of convolutional layers for CNN-FCs without pooling. The suffix for each model name indicates the number of units in each convolutional layer. Peak validation CA is achieved with four layers each consisting of 12 convolutional units. There does not, however, appear to be a clear trend as the number of units or layers is increased.

In Figure 4.6, we see how validation CA changes as we add layers to CNN-FCs with 2, 4, 8, 12, 16, 20 and 24 units in each convolutional layer. These network configurations perform only modestly better than the 25% CA expected from a random classifier, with results ranging from about 26–34% CA. Peak performance was achieved by a CNN-FC consisting of four layers containing 12 units each. There does not, however, appear to be a large difference in performance for networks with more or fewer convolutional units in each layer and we often see a 1–5% difference in CA when the same network configuration is retrained using a different random weight initialization.

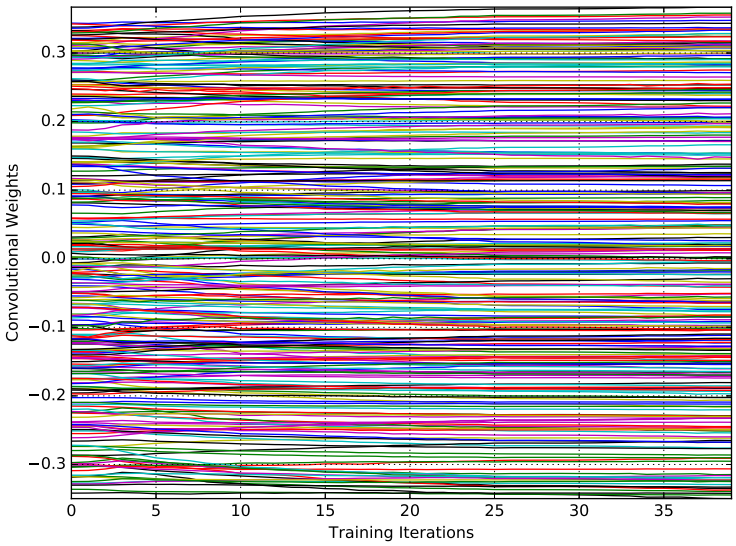
In order to see why these networks do not perform well, it is important to first consider the number of free parameters in each layer. For our CNN-FC that achieved peak performance, the total number of weights across the four convolutional layers is 1,632 while the number of

weights in the hidden readout layer is 60,490 and the number of weights in the visible readout layer is 44. The total number of free parameters in this CNN-FC is then 62,166. Recall that our best-performing LDA classifier, by contrast, has a total of only 612 free parameters.

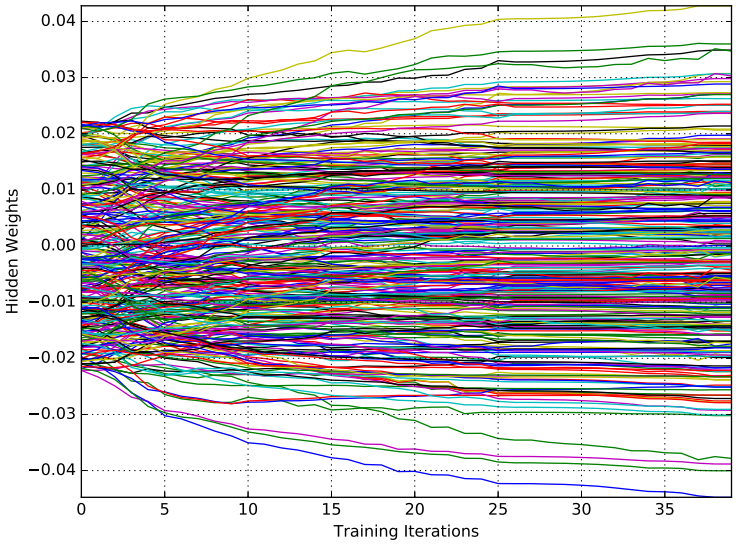
The sheer number of parameters in our CNN-FCs raises concerns about the model's propensity for overfitting, which will be confirmed when we examine the regularization characteristics of theses models shortly. Perhaps a more important concern, however, is that over 97% of the free parameters in this network are found within the fully connected, hidden readout layer. Note that the number of parameters in the hidden readout layer scales linearly with the number of hidden units in this layer. This means that even in the extreme case of a network with only a single hidden readout unit, there would still be 6,049 parameters associated with this lone hidden unit, which is still 78% of the total weights in the network.

Recall that it is primarily the responsibility of the convolutional layers to form a time-invariant representation of the signal because only these layers have a receptive field that is localized in time. A network, or layer in this case, that is fully connected across time is generally not able to learn a time-invariant representation on its own, unless it is presented with a large number of training examples with many potential time shifts.

In Figure 4.7, we see the evolution of the first 300 connection weights in both the convolutional layers, Figure 4.7a, and in the hidden readout layer, Figure 4.7b, during the training procedure for Subject-1. Notice that while there is a large amount of change and reorganization of the weights in the hidden readout layer, there is very little movement of the weights in the convolutional layers. After about 40 iterations, the loss function reaches numerical precision and the training procedure terminates. It is unsurprising that the weights in the hidden readout layer move somewhat more quickly than the weights of the convolutional layers because error gradients tend to diminish as they propagate through multiple layers during the backward pass of our optimization process. Since, however, the hidden readout layers contain many more connections than the convolutional layers, the network is able to fit the data before much weight adaptation has occurred in the convolutional layers by primarily using the hidden readout layer.



(a) CNN-FC convolutional weights during training.



(b) CNN-FC readout layer hidden weights during training.

Figure 4.7: Connection weights versus training iterations for a CNN-FC without pooling for Subject-1. (a) First 300 convolutional weights during training. There is relatively little movement in these weights indicating that only minor learning is taking place. (b) First 300 weights in the hidden readout layer. There is a relatively large amount of change and reorganization in these weights before the loss function reaches numerical precision at 40 iterations. This indicates that the hidden readout layer is primarily responsible for fitting the training data.

In other words, the poor performance of our CNN-FC architecture can be explained by the fact that the readout layers rapidly fit the training data before the convolutional layers are able to learn a time-invariant representation of the signals.

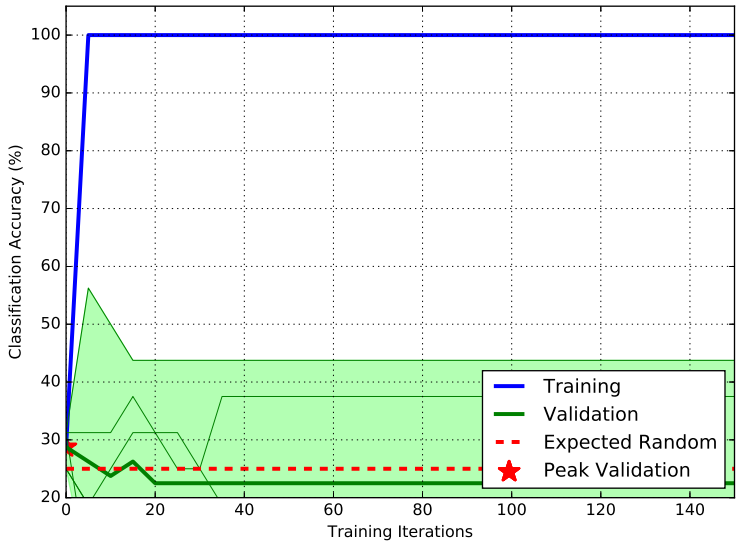


Figure 4.8: Training and validation CA versus training iterations for Subject-1 when using CNN-FC without pooling. The thin lines denote fold in the cross-validation while the thick lines show the mean across folds. Note that CA is polled at five-iteration intervals. Peak validation accuracy and 100% training accuracy are both reached in less than five iterations, after which validation accuracy drops. This indicates considerable overfitting, which can be explained by the large number of free parameters and by the large fraction of parameters in the hidden readout layer.

In Figure 4.8, we see the training and validation CA for Subject-1 during training for our best-performing CNN-FC configuration in the above experiment, which has four convolutional layers consisting of 12 units each and fully connected readout layers with 10 hidden units. Note that we now measure CA every five iterations. Notice that the training CA rapidly reaches 100% after only about five training iterations while the validation CA peaks before five iterations, after which it declines until leveling off after about 20 iterations. The loss function typically reaches numerical precision after about 40-50 iterations, at which time the training procedure is terminated. This rapid increase in training performance followed by a drop in validation performance and then convergence are clear indicators that this network architecture suffers from

catastrophic overfitting. Again, this can be explained by the large number of free parameters in these networks and by the large fraction of these parameters that are found in the hidden readout layer.

Pooling with CNN-FCs

One potential method for reducing the number of free parameters in the hidden readout layer of our our CNN-FCs is to introduce pooling between layers. Since pooling decimates the representation across time, the number of connections in the final readout layer grows smaller as the number of convolutional layers that incorporate pooling increases.

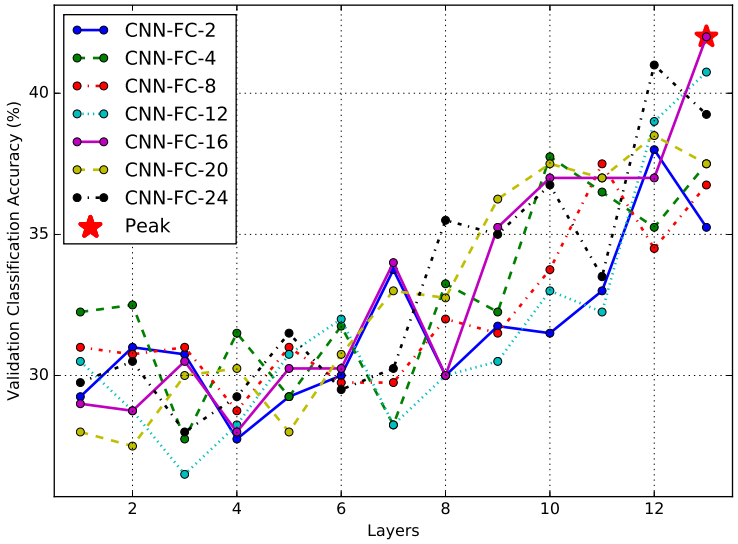


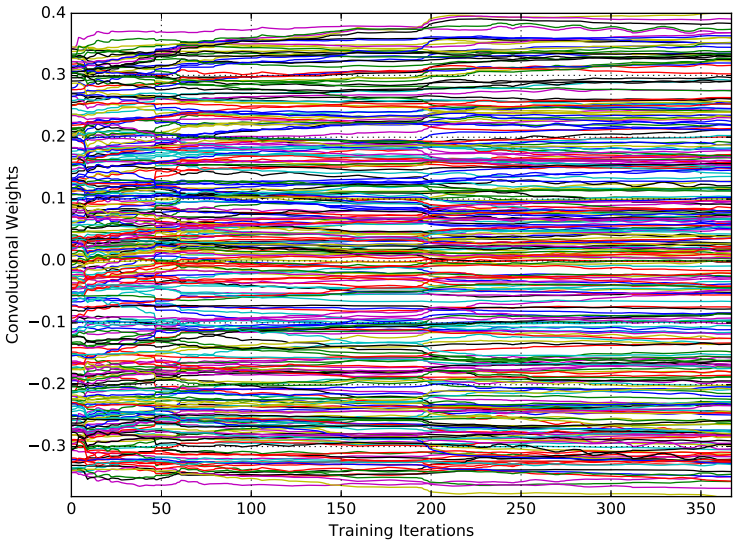
Figure 4.9: Mean validation CA, averaged across the first five subjects, versus number of convolutional layers for CNN-FCs with 2:1 pooling after alternating convolutional layers. The suffix for each model name indicates the number of units in each convolutional layer. Peak validation CA is achieved with 13 layers each consisting of 16 convolutional units. CA generally increases as the number of layers increases but cannot exceed 13 layers because the number of timesteps in the final layer reaches zero.

In Figure 4.9, we see a repetition of our previous experiment but now with 2:1 average pooling introduced after every other convolutional layer, i.e., even numbered layers. In general, we have found that pooling after alternating convolutional layers works better than pooling after every layer. Given the narrow width of the convolutional kernels used in these networks, it

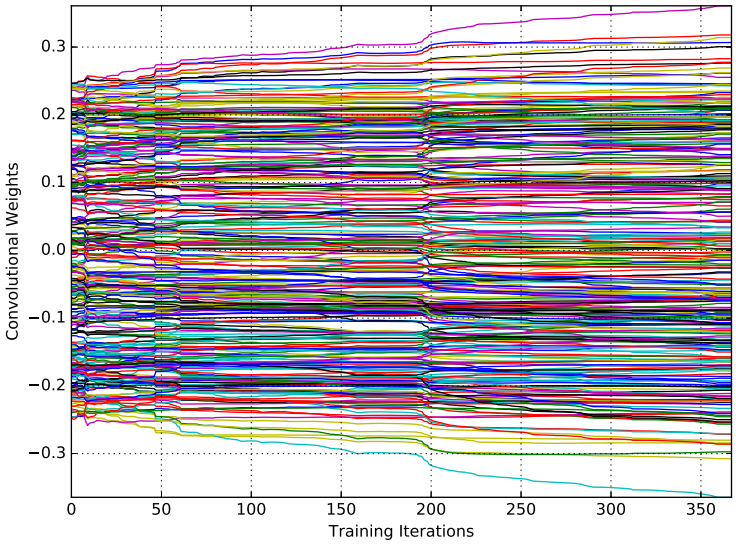
makes sense that at least two convolutional layers are required in order for the network to sufficiently capture temporal information before a change of scale is introduced. The introduction of pooling clearly improves the validation CA for our CNN-FCs and performance now continues to improve as we add layers, up to 42.00% CA with 13 layers and 16 units in each convolutional layer. We cannot, however, stack more than 13 layers because the number of timesteps in the final layer would be reduced to zero.

The optimal network configuration in this experiment has a total of 10,182 free parameters with 9,808 residing in the convolutional layers and 330 in the hidden readout layer. This is a dramatic change in the ratio of parameters found in the hidden readout layer from the previous experiments with about 97% of the weights now residing in the convolutional layers. Although the introduction of pooling in our CNN-FC architecture clearly reduces the number of weights found in the hidden readout layer, the network now has many more convolutional layers and still has a large number of free parameters overall. This may lead to difficulties optimizing the weights in the earlier convolutional layers, due vanishing gradients, and may limit generalization performance due to the model being overly complex.

In Figure 4.10, we examine how the convolutional weights in this CNN-FC architecture evolve during the training procedure for Subject-1. In Figure 4.10a, we see how the first 300 convolutional weights in the first convolutional layer, i.e., closest to the input signal, change during training. Note that when compared to our CNN-FC architecture without pooling, we now see more change and reorganization of the weights. This supports our claim that reducing the number of free parameters in the fully connected readout layers encourages more learning to take place in the convolutional layers. In Figure 4.10b, we see how the first 300 weights in the final convolutional layer, i.e., closest to the readout layers, changes over the course of the training procedure. Notice that there appears to be more change in the weights in the final layer than we observed in the first layer. This can be explained by the fact that error gradients tend to diminish as they pass through multiple layers of a neural network. Given that this architecture now has 13 convolutional layers followed by two readout layers, it is not surprising that the



(a) CNN-FC first-layer convolutional weights during training.



(b) CNN-FC final-layer convolutional weights during training.

Figure 4.10: Connection weights versus training iterations for Subject-1 when using a CNN-FC with 13 convolutional layers consisting of 16 hidden units and 2:1 pooling after every other convolutional layers. We now see more movement in the convolutional weights during training than we did without pooling. This can be attributed to the reduction in the number of parameters in the hidden readout layer. The weights in the first convolutional layer (a) show less movement than the weights in the final convolutional layer (b). This is likely due to a combination of vanishing gradients and the large number of free parameters now found in the convolutional layers.

weights in later layers tend to move more quickly. This does raise the concern, however, that the earlier layers may not be fully trained when the network converges. In other words, much of the learning may be taking place in the later convolutional layers before early stopping terminates the training procedure, which may hinder generalization performance.

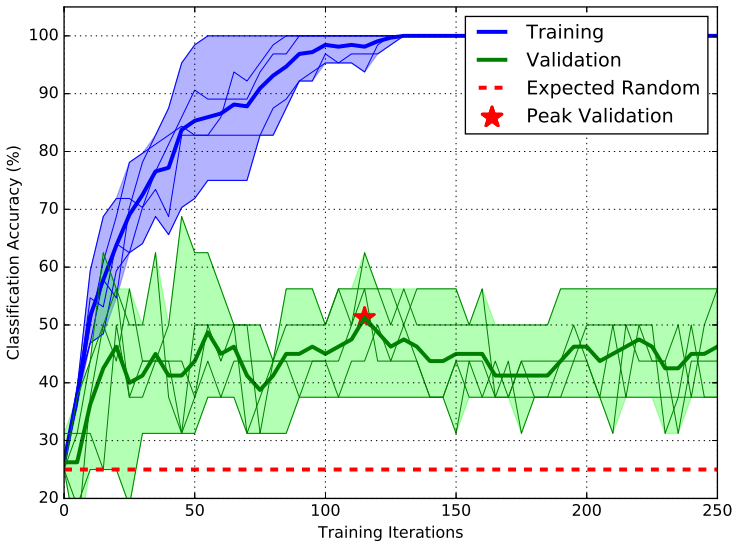


Figure 4.11: Training and validation CA versus training iterations for Subject-1 when using a CNN-FC with 13 convolutional layers and 2:1 pooling after every other convolutional layers. The thin lines denote per-fold CA while the thick lines show the mean. Note that CA is polled at five-iteration intervals. Peak validation accuracy is achieved at about 35 iterations after which it declines somewhat. Training CA climbs to 100% at about 120 iterations, after which it levels off. The training procedure is slower than without pooling and overfitting appears to be better controlled.

In Figure 4.11, we see how classification accuracy varies during our early stopping regularization procedure for Subject-1. Again, our validation CA is again evaluated every five iterations. Note that our CNN-FC architecture with pooling tends to converge more slowly than our CNN-FCs without pooling, achieving 100% training CA after about 140 iterations. Validation CA grows quickly from 0–25 iterations and then gradually grows until peaking at 51.25% CA at about 115 iterations and then levels off with some fluctuation. The fact that our CNN-FCs with pooling tend to converge more slowly than without pooling and the fact that the validation accuracy does not drop sharply, suggests that pooling helps to reduce overfitting.

The introduction of pooling to our CNN-FC architecture clearly improves performance, encourages learning in the convolutional layers and reduces overfitting. CNN-FCs with pooling do, however, still have many free parameters overall and we have shown that the movement of the weights in the early convolutional layers is relatively slow. Furthermore, we are largely unable to tune the number of convolutional layers to capture an appropriate amount of hierarchical, nonlinear and temporal information because we must always use as many layers as possible in order to increase pooling and reduce the size of the hidden readout layer. Although having a very large number of convolutional layers may be desirable when working with voluminous datasets, it seems likely that it is appropriate to use fewer convolutional layers when working with the relatively undersampled dataset at hand. These observations suggest that our CNN-FC architecture may be achieving less-than-ideal generalization performance by imposing an overly complex model with too many layers and free parameters.

4.1.4 Label Aggregation

In order to address the limitations we have identified in our CNN-FCs and to further simplify our network architecture, we can replace the fully connected readout layers with label aggregation readout layers, as previously described in Section 3.2.1 and Section 3.3.2. In Figure 4.12 we examine how classification performance varies as we stack convolutional layers for our CNN-LA architecture without pooling. The introduction of label aggregation readout layers leads to a significant improvement in validation CA relative to our CNN-FC architecture. A peak mean validation CA of 66.00% is now achieved with three convolutional layers each consisting of 24 units. For networks with fewer units in the convolutional layers, performance generally peaks with slightly more layers, e.g., six layers for a network with only four units, while networks with more units typically achieve peak performance with fewer layers, e.g., two layers for a network with 28 units per layer. This suggests that there is a benefit to limiting the number of nonlinear transfer functions, whether by using fewer layers with more hidden units or more layers with fewer hidden units. Peak performance is, however, always achieved with networks that contain

multiple convolutional layers, suggesting that there is also a benefit to stacking convolutional layers to form a deep network, at least when using our narrow convolutional window of $d = 3$.

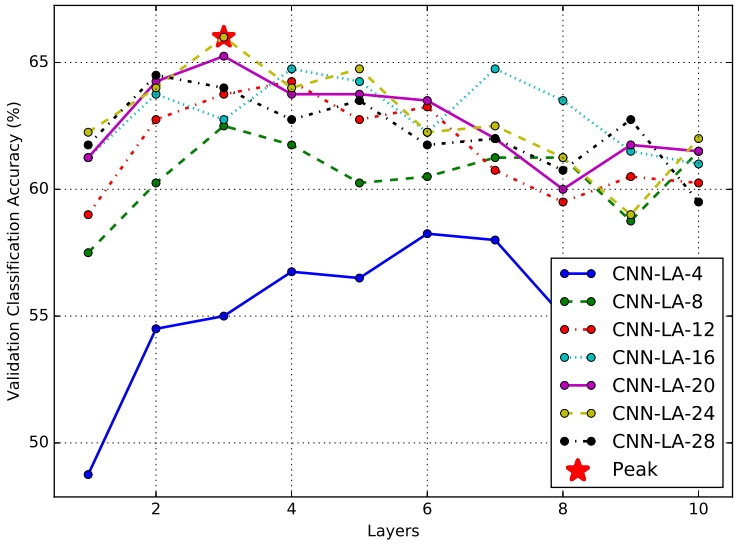


Figure 4.12: Mean validation CA, averaged across first five subjects, versus number of convolutional layers for a CNN-LA without pooling. The suffix for each model name indicates the number of units in each convolutional layer. Peak validation CA is achieved with three layers each consisting of 24 convolutional units. Networks with fewer hidden units often require more layers than networks with many hidden units. In all cases, however, multiple convolutional layers are beneficial.

The introduction of label aggregation readout layers also reduces the number of free parameters in our models considerably when compared to our CNN-FC architectures. A CNN-LA with three layers and 24 convolutional units in each layer has a total of 4,204 parameters. In addition to the reduction of free model parameters, label aggregation layers also have only a few linear model parameters in the readout layer. In this case, there are 100 parameters in the readout layer regardless of the number of convolutional layers. This causes learning to take place almost entirely in the convolutional layers, which encourages the network to learn a time invariant representation using only the small local receptive fields of our convolutional units.

Pooling with CNN-LAs

Pooling can also be introduced into our CNN-LA architecture. In Figure 4.13, we see a repetition of our previous experiment except that our networks now have 2:1 average pooling between every other convolutional layer. The peak mean validation CA for these models is now 66.50% with three convolutional layers consisting of 16 units each. Although this is only a slight improvement over our models without pooling, 0.5% validation CA, it is important to note that the introduction of pooling causes peak performance to now occur with fewer convolutional units and, in some cases, fewer convolutional layers.

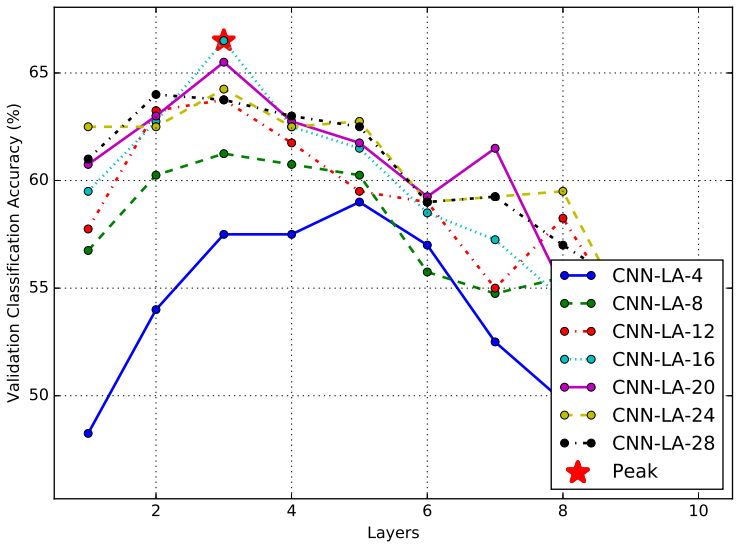


Figure 4.13: Mean validation CA, averaged across first five subjects, versus number of convolutional layers for a CNN-LA with 2:1 average pooling. The suffix for each model name indicates the number of units in each convolutional layer. Peak validation CA is achieved with three layers each consisting of 16 convolutional units. Although the CA is only slightly better than without pooling, the optimal configurations typically have fewer hidden units and convolutional layers.

Recall from Section 3.4.2 that the introduction of pooling increases the duration of the network's impulse response and its ability to incorporate longer-term temporal information. The addition of this longer-term temporal information may allow the network to achieve similar levels of performance while requiring fewer convolutional units. In other words, it may be more valuable for the network to fit slightly longer-term patterns in the signals than to thoroughly fit

shorter-term patterns. It is also possible that there may be some level of redundant information that can be captured either by using more convolutional units or by increasing the amount of temporal information through pooling. Note that a CNN-LA with pooling and three convolutional layers and 16 units in each layer has a total of 2,036 model parameters.

These experiments suggest that the change in scale and increased memory capacity provided by pooling slightly improves the performance of the network while also reducing the number of free parameters required. There is, however, only a benefit to introducing a single pooling stage, after which validation CA begins to fall. This can be explained by the fact that there is some information loss incurred during the pooling process and also suggests that there is a limit to the value of imposing changes in temporal scale and incorporating additional longer-term information.

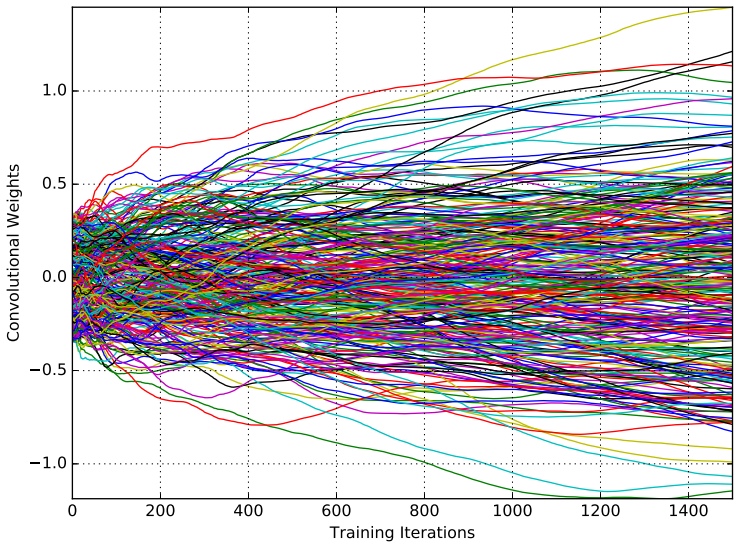


Figure 4.14: First 300 connection weights in the first convolutional layer versus training iterations for Subject-1 when using a CNN-LA with three convolutional layers consisting of 16 units with 2:1 pooling after alternating convolutional layers. CNN-LA runs for more training iterations than CNN-FC before converging and we see much more change in the convolutional weights. This aligns with our assertion that CNN-LA relies more heavily on the convolutional layers than the readout layer.

In Figure 4.14, we see how the first 300 convolutional weights in a CNN-LA with pooling change during the training procedure for Subject-1. This network has three convolutional lay-

ers with 16 hidden units each, which is the configuration that achieved peak performance in the previous experiment. Notice that the training procedure now lasts for many more iterations than we previously saw for our CNN-FCs and there there is a larger amount of change in the convolutional weights. This supports our assertion that our CNN-LA architecture places a much greater responsibility on the convolutional layers and, therefore, encourages a more time invariant representation.

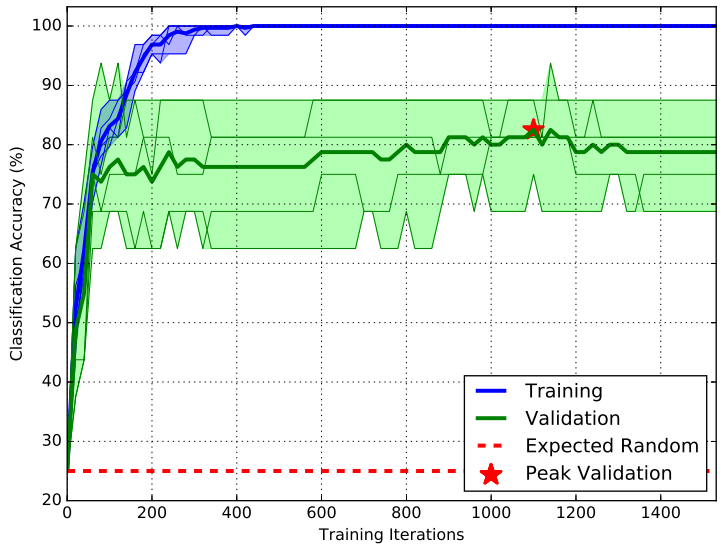


Figure 4.15: Training and validation CA versus training iterations for Subject-1 when using a CNN-LA with three convolutional layers consisting of 16 units and 2:1 pooling after alternating convolutional layers. The thin lines denote per-fold CA while the thick lines show the mean. Note that CA is polled at 20-iteration intervals. Peak validation accuracy is 82.5% at about 1,100 iterations. Training CA climbs to 100% at about 400 iterations. Note that validation CA continues to climb after the training CA reaches 100% and does not drop significantly, demonstrating that the network continues to learn after perfectly classifying the training data.

In Figure 4.15, we see how validation CA varies for Subject-1 during our early stopping procedure for the same CNN-LA. Training CA climbs to 100% around 400 iterations while the peak validation CA of 82.5% is achieved for this subject at about 1,100 training iterations, after which validation accuracy falls slightly and levels off. Numerical precision for our loss function is not typically reached until about 1,300 iterations. The observations that CNN-LAs typically require more training iterations than CNN-FCs and that the validation CA does not begin to fall rapidly

are consistent with our assertions that CNN-LAs are better equipped to learn a time invariant representation and are less prone to overfitting. Also notice that the validation CA of our CNN-LAs often continues to grow even after the training accuracy has reached 100%. This means that the model continues to learn meaningful patterns in the data even after all training segments are perfectly classified. This can be partially explained by the fact that the loss function in our CNN-LAs incorporates information from every timestep. Even after each segment is classified correctly, the network still attempts to improve the class labels for each timestep, which further improves generalization performance.

4.1.5 Transfer Learning

Another potential approach for improving the performance of our CNNs is to incorporate transfer learning by performing an initial training procedure using data from multiple participants, as described in Section 2.2.4 and Section 3.3.3. In order to explore this possibility, we first initialize the weights of our networks by performing a fixed number of training iterations using the combined training data for multiple subjects, excluding the subject at hand. We then fine-tune the network by applying an additional number of training iterations, determined by our early-stopping procedure, using only the training data for the current subject. Since we have established that our datasets are noisy and undersampled yet we seek to model sophisticated patterns in our EEG signals, it seems reasonable that this approach for increasing the amount of training data available might improve performance by providing more examples for our networks to learn from. This approach does assume, however, that there is information in the EEG signals that is generally useful across subjects and that this knowledge will not be lost during the fine-tuning procedure. It is worth noting that similar transfer learning approaches for CNNs have been successfully applied to a variety of problems in computer vision [47, 84, 101]. We are not aware, however, of any previous work exploring transfer learning in the context of mental-task BCIs.

In the following experiments with transfer learning, we utilize only the data from the first 10 participants, or the nine remaining if the subject of interest is within the first 10, during the initial training stage. Recall that the first 10 participants do not have impairments and data were recorded in a laboratory environment while the remaining four participants have severe motor impairments and data were recorded in their home environments. This setup allows us to simulate the realistic use case where the models are initially trained under controlled circumstances with minimal environmental noise and artifacts and are then fine-tuned on an individual basis for a variety of users in different environments. Although we do recognize that including transfer learning data from participants with impairments may help to improve performance for people with disabilities, the size and scope of the current dataset does not permit this type of analysis.

Table 4.1: The network configurations used in our experiments with transfer learning. Note that all of these networks use label aggregation readout layers. In addition to the best-performing configurations found in our experiments without transfer learning, we also examine somewhat larger networks because this procedure has more training data available.

	TDNN-TR			CNN-TR-A			CNN-TR-B			CNN-TR-C		
Layer	u	d	Pool	u	d	Pool	u	d	Pool	u	d	Pool
1	52	7	1:1	16	3	1:1	16	3	1:1	16	5	1:1
2				16	3	2:1	16	3	2:1	16	5	2:1
3				16	3	1:1	16	3	1:1	20	3	1:1
4							16	3	1:1	20	3	1:1
Parameters	3,176			2,036			2,820			4,236		

We will examine four variants of our proposed CNNs, summarized in Table 4.1. Since our previous experiments and analyses suggest that label accumulation readout layers allow for the most flexibility and often yield the highest validation accuracies, we will focus solely on these types of networks. Since it is important to identify any differences between single-layer and multilayer networks, the first approach that we will examine combines our transfer learning procedure with the TDNN configuration that we established in our previous model selection experiments, i.e., with 52 hidden units and an embedding dimension of 7. We refer to this type

of network as a Time-Delay Neural Network with Transfer Learning (TDNN-TR). We will also examine our transfer learning approach along with the CNN-LA configuration established in our previous experiments, i.e., with three layers with a convolutional width of three, 16 artificial neurons in each layer and 2:1 pooling following the second convolutional layer. We refer to this type of network as a Convolutional Neural Network with Transfer Learning with configuration A (CNN-TR-A).

Given that transfer learning increases the amount of training data available, it also seems appropriate to investigate network configurations that are slightly larger than those established in our previous experiments, which did not involve transfer learning. Along these lines, we will examine the performance of CNN-LAs that have four convolutional layers with a width of three, 16 artificial neurons in each layer and 2:1 pooling following the second layer. We refer to this configuration as CNN-TR-B. Note that this network is similar to CNN-TR-A except that it has one additional convolutional layer.

It is also common practice when designing CNNs to gradually increase the number of hidden units in each convolutional layer while simultaneously reducing the width of the convolutions [44, 45]. This approach adjusts the width of the convolutional kernel to align with the change of scale introduced by pooling and can also encourage the network to migrate information from the temporal axis to spatial patterns across the hidden units. This may be especially useful for our CNN-LA architectures because the final softmax readout layer has connectivity only across the hidden units and not directly along the time axis. In order to explore this possibility, we will also examine a CNN-LA that has four convolutional layers with an initial width of five in the first two layers that is then reduced to three following 2:1 pooling after the second layer. In this configuration, we also increase the number of artificial neurons from 16 in the first two layers to 20 in the final two convolutional layers. We refer to this configuration as CNN-TR-C. Note that while we have also explored this type of tapered network design in pilot experiments without transfer learning, we have found that it typically offers little or no bene-

fit over the configurations found in our previous experiments. Presumably, larger amounts of training data are required in order to take full advantage of these larger networks.

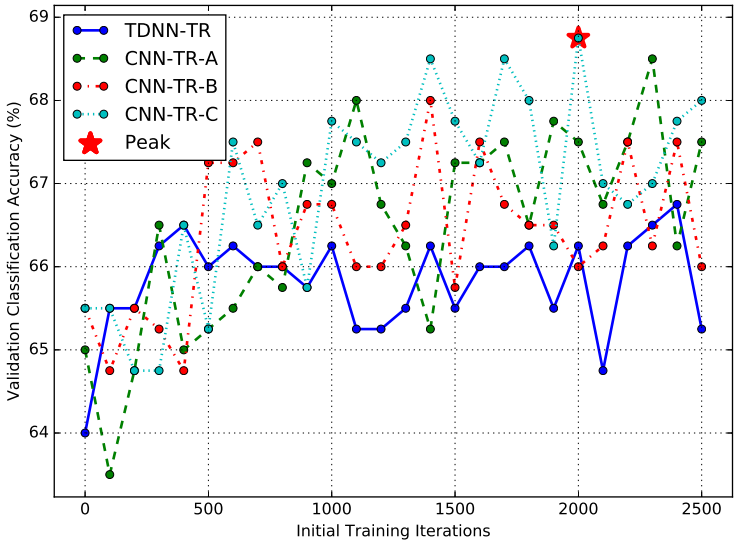


Figure 4.16: Validation CA versus transfer learning initialization iterations for TDNN and each of our CNN-TR architectures. Introducing transfer learning appears to be beneficial for all architectures, but the largest improvement is seen for larger multilayer networks. CA tends to level off after around 1,500 iterations and the highest peak validation CA is for CNN-TR-C at 2,000 iterations.

In Figure 4.16, we see how the mean validation CA over the first five participants varies as the number of initial transfer learning iterations is increased for each of the network configurations described above. At each point this figure, the horizontal axis specifies the number of initial transfer learning iterations that are performed before fine-tuning. The network is then fine-tuned for the subject at hand while early stopping is used for regularization. Mean validation CA is measured each time the number of initialization iterations increases by 100.

Although we do use a different random weight initialization for each subject and network configuration, our implementation of this experiment saves the weights from the initial transfer learning stage before branching to the fine-tuning stage and then reinstates the weights before performing additional transfer learning iterations. In this way, we avoid unnecessary computation and avoid reinitializing our networks each time the number of transfer learning iterations is increased. Nevertheless, we see considerable variability in our validation CA as the number of

initial transfer learning iterations is varied. This type of behavior is not unusual when working with neural networks because optimizing the weights of these networks is known to be a non-convex problem. In other words, our neural networks contain many locally optimal solutions and a change in the network's weights can steer the optimization and regularization processes toward a different solution. The variability in validation CA seen in this experiment is, however, notably higher than in our previous experiments. This can be explained by the fact that the weights at the start of our fine-tuning process are no longer small and drawn from a random uniform distribution, as was the case with our previous random weight initializations. Instead, our initial transfer learning stage can create relatively large changes in the network weights before the onset of our fine-tuning stage which, in turn, can steer the optimization process toward locally optimal solutions that are very different.

Despite the increased variability in this experiment, we can clearly see that the introduction of transfer learning improves our mean validation accuracy considerably by noting that zero initial training iterations describes the scenario in which no transfer learning is performed. Validation accuracy then increases as the number of initial transfer learning iterations increases before leveling off with some fluctuation after about 500 iterations for TDNN-TR and 1,500 iterations for our multilayer networks. These observations provide supporting evidence that there is information contained in these EEG signals that is relevant for classifying these mental tasks that can be transferred across subjects. It also appears that our larger multilayer architectures tend to benefit more from transfer learning than our smaller networks. Note that TDNN-TR achieves a peak mean validation CA of 66.75% at 2,400 initial iterations while CNN-TR-C achieves a peak of 68.75% at 2,000 initial iterations. Also, notice that our tapered CNN-TR-C configuration generally outperforms CNN-TR-A and CNN-TR-B. Although the difference in performance between these network configurations is modest, it is consistent enough to suggest that the additional capacity provided by the larger CNN-TR-C results in a slight performance benefit relative to the other network configurations. In subsequent experiments with transfer

learning, we will use our CNN-TR-C architecture with 2,000 initial iterations. For the sake of brevity, we will use the CNN-TR acronym to refer to CNN-TR-C, unless otherwise specified.

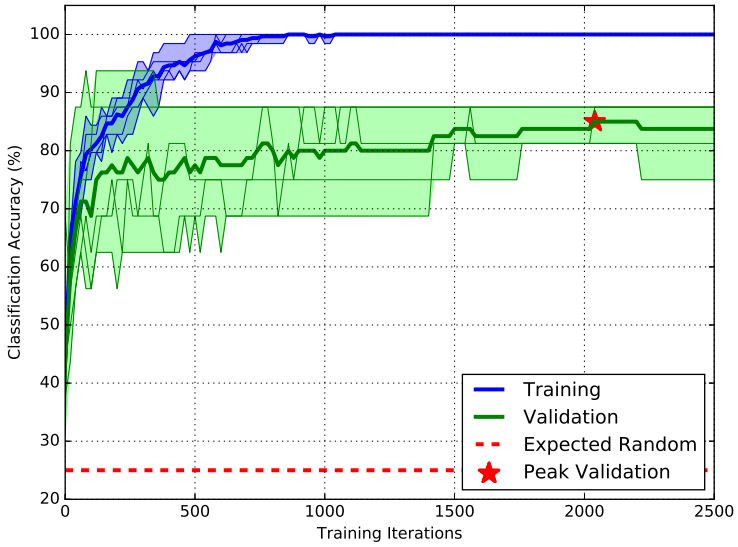


Figure 4.17: Training and validation CA versus training iterations for Subject-1 when using CNN-TR-C. The thin lines denote per-fold CA while the thick lines show the mean. Note that CA is polled at 20-iteration intervals. Peak validation accuracy is 83.75% at about 1,040 iterations. Training CA climbs to 100% at about 750 iterations. Note that validation CA again continues to climb after the training CA reaches 100% and does not drop significantly, suggesting little overfitting.

In Figure 4.17, we see how the training and validation CA changes during the fine-tuning procedure for Subject-1 when using CNN-TR. Notice that our initial training and validation CAs are now well above the random 25% at iteration zero, starting at about 35%. In other words, our transfer learning approach improves the validation CA for Subject-1 by more than 10% at the onset of our fine-tuning stage. Training CA reaches 100% CA after about 750 training iterations while validation CA continues to improve for up to about 2,000 training iterations. As we saw with CNN-LA, the validation accuracy continues to gradually improve long after training accuracy reaches 100% and there is minimal drop in validation accuracy. This suggests that there is little overfitting.

4.2 Test Performance

We have now established good model configurations for each of our classification approaches and explored the validation performance of our models across several network architectures and a variety of hyperparameters. Next, we will proceed by examining the final test performance of each of our methods over the test partitions for all subjects, which have been reserved until now. We will begin by examining the test Classification Accuracies (CAs) achieved by our baseline classifiers and each of our proposed TDNN and CNN architectures. We will perform several analyses of these results that will lead to insights into their relative performance.

In Section 4.2.3, we also present our final performance results in terms of Information Transfer Rates (ITRs), which incorporate both the number of class labels and the rate at which decisions are made. These results can be more easily compared with other experimental paradigms and are more informative about the amount of information that a user of a BCI system might expect to convey. We will also perform a brief comparison between our methods and several other studies, in terms of ITRs, that will demonstrate that our results are at least on par with the state of the art.

In Section 4.2.4, we will then examine how the performance characteristics of our methods change as their decision rates are varied. These experiments will explore the tradeoffs between accuracy and responsiveness and highlight how both CA and ITR can be influenced by decision rate. These aspects of performance are important for predicting user experience and should be considered when designing practical BCI applications. Finally, in Section 4.2.5, we will examine the performance of each of our individual mental tasks. These experiments will show clear differences in the ability of our methods to distinguish between different mental tasks and will yield insights into the types of mental tasks that are most appropriate for use in BCIs.

4.2.1 Power Spectral Densities

In Table 4.2, we present the final test Classification Accuracies (CAs) for each of our PSD-based classification approaches. These results are broken down by our two groups of partic-

ipants described in Section 3.5.3: 10 participants without impairments in a laboratory environment and four participants with impairments in their homes. Test CAs are shown for each classifier and individual participant as well as the mean CAs for each group of participants. The mean CAs across all participants are presented in the final row. The highest CAs in each row of the table, including ties, are highlighted in bold. As was established in the previous section, we use a span for Welch's method of $\omega = 0.25$ for LDA and $\omega = 0.5$ for QDA and ANNs. Our ANNs all contain 64 hidden units. The final values for our regularization parameters, covariance averaging (λ), covariance shrinkage (γ) and training iterations (Iter.) are established on a per-subject basis using our five-fold cross validation procedure over the training partition, as described in Section 3.5.4.

In Figure 4.18, we see a visualization of Table 4.2 that shows both the distribution of CAs for each method as well as the relative performance for each pair of classifiers. In the plots along the diagonal, we see histograms of the distribution of CAs across subjects for each classification approach. Note that the histograms are stacked showing the subjects with impairments on the top and participants without impairments on the bottom. In the off-diagonal scatter plots, we see the relative CAs for each subject and pair of classifiers with a single point for each participant and where the horizontal axis represents the CA achieved by the classifier listed on the bottom and the vertical axis represents the CA achieved by the classifier on the left. The diagonal is shown for reference and each point below the diagonal shows an instance where the classifier on the bottom outperformed the classifier on the left and vice versa.

The histograms in Figure 4.18 show that the distributions of our CAs tend to be skewed to the right, with the majority of participants achieving CAs that are lower than the mean. The mode of the distributions for LDA and QDA is just above the expected random of 25% CA and the distribution for our ANNs is bimodal around 25% and 50% CA. Note that three out of the four participants with impairments, subjects 11, 13 and 14, achieve relatively low CAs across all three methods, which contributes considerably to this skew and low mode values.

Table 4.2: Test classification accuracies and regularization parameters for our baseline classifiers. Results are reported for both subjects with impairments in the laboratory and subjects without impairments in their homes. The means for each group of participants are shown in the final row of each section and the overall means are shown in the final row. The highest CA for each row, including ties, is shown in bold.

	Participant	LDA	λ	QDA	γ	ANN	Iter.
No Impairment	1	**85.00**	0.66	70.00	1.00	**85.00**	30
	2	**55.00**	0.80	50.00	1.00	50.00	20
	3	**40.00**	0.44	20.00	1.00	30.00	25
	4	**55.00**	0.94	40.00	0.98	45.00	15
	5	70.00	0.70	**85.00**	1.00	75.00	5
	6	**40.00**	0.94	**40.00**	1.00	20.00	30
	7	**65.00**	0.12	40.00	0.82	60.00	10
	8	**45.00**	0.88	**45.00**	1.00	**45.00**	15
	9	50.00	0.10	**70.00**	1.00	55.00	20
	10	**25.00**	0.16	**25.00**	1.00	**25.00**	30
	Mean	**53.00**	0.57	48.50	0.98	49.00	20.00
Impairment	11	25.00	0.18	30.00	1.00	**35.00**	15
	12	**80.00**	1.00	55.00	0.02	65.00	15
	13	**25.00**	0.22	**25.00**	1.00	**25.00**	15
	14	30.00	0.90	25.00	1.00	**40.00**	30
	Mean	40.00	0.57	33.75	0.76	**41.25**	18.75
All	Mean	**49.29**	0.57	44.29	0.92	46.79	19.64

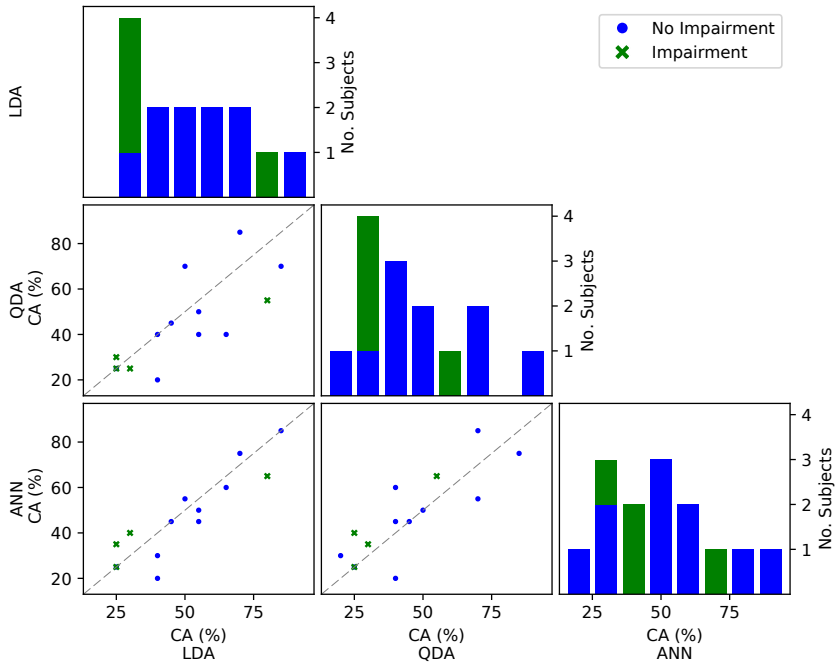


Figure 4.18: The distributions and pairwise CAs for each of our baseline classification approaches. The plots along the diagonal, show histograms of the distribution of test CAs for each participant. The off-diagonal scatter plots show the relative CAs for each subject and pair of classifiers.

Kolmogorov-Smirnov tests for normality fail to show that our CAs are normally distributed for any of these three classification approaches ($0.880 < p < 0.997$). In Section 4.2.2, we will see that the CAs for our CNN-based approaches also tend to be skewed or multimodal. As a result, we will utilize Wilcoxon rank sum tests, also known as Mann-Whitney U tests, in order to check our results for statistical significance because Wilcoxon tests are nonparametric and do not rely on assumptions of normality [131, 132]. Since Wilcoxon tests can be more conservative than t-tests, it is valuable to note that we have also performed t-tests and found that both methods lead to the similar conclusions.

When using paired, two-sided Wilcoxon tests without correcting for multiple comparisons, we fail to show that there is a statistically significant difference in the mean CAs achieved by any of these three classifiers, both within the two groups ($0.18 < p < 1.00$) and across all participants ($0.20 < p < 0.50$). A full table of p-values for each individual test, with and without corrections for multiple comparisons, can be found in Appendix A. Our inability to draw firm statistical

conclusions in this case may be partially due to our small and diverse sample of subjects or because these methods might not achieve large differences in CA. Nevertheless, the observed overall mean for LDA is higher than both QDA (5.0% difference) and ANNs (2.5% difference). The only exception to this is for the group of participants with impairments where the mean CA for our ANNs is slightly higher than for LDA (1.2% difference). We can also see in both Table 4.2 and Figure 4.18 that the peak CAs achieved by LDA for individual participants is higher or equally as good as the other approaches for 10 out of the 14 participants.

Regularization for all of these models also tends to be aggressive, which is consistent with our observations during model selection in Section 4.1.1. ANNs tend to achieve peak validation accuracy after an average of only about 20 iterations and the average value established for covariance mixing in QDA, γ, was 0.92. This suggests that the fully averaged covariance matrices used by LDA tend to outperform the separate class covariance matrices used by QDA. Despite the fact that our ANNs perform slightly better for the group of participants with impairments, which is a potentially valuable observation, we believe that the consistently higher overall performance observed with LDA and its tendency to be regularized more moderately suggest that LDA is the most appropriate of these three classification algorithms. As such, we will use LDA as the benchmark algorithm for PSD-based approaches in our remaining experiments.

Although the CAs in Table 4.2 may appear somewhat low upon first inspection, there are several important points to consider when evaluating these performance metrics. First of all, note that our classifiers have four potential output labels, one for each mental task, which means that assigning random class labels would be expected to result in only 25% CA. The rate at which class labels are assigned also affects CA. In Section 4.2.3, we will examine our final performance results in terms of Information Transfer Rates (ITRs), which incorporate both the number of class labels and the rate at which decisions are made, and in Section 4.2.4 will directly examine the effects of varying the decision rates of our classifiers.

It is also important to note that our mean CAs are averaged over individuals with widely varying levels of performance; although some subjects attain CAs as high as 85% correct, others

do not exceed 25% for any of these three methods. This is consistent with studies by other research groups that have found it to be common for some subjects to be unable to achieve good performance with BCIs [133, 134]. This is a trend that we will see across all of our methods and may be due to a variety of reasons, including individual differences in brain signals, differences in the way that the mental tasks were performed, lack of attention and environmental noise and artifacts, which will be discussed in more detail in Section 4.3.1.

EEG signal classification is also a challenging problem in general for the reasons described in Section 1.1 and it is typical for CAs to be considerably less than 100%. This may be further exasperated by the fact that we have designed our experimental setup in order to assure that we are investigating a realistic scenario for use in practical BCI systems. Specifically, we are using an inexpensive, portable EEG system with only eight channels, a relatively small amount of training data and we include participants with impairments in realistic environments. In Section 4.2.4, we will perform a rough comparison between our methods and several other studies that will demonstrate that our methods are on par with the state of the art, despite our rigorous experimental design.

4.2.2 Convolutional Networks

Now that we have closely examined our baseline methods, we will continue by examining the CAs attained by all of our proposed CNN-based approaches and compare these methods directly with LDA. In Table 4.3, we see the final test CAs for each of our proposed neural network architectures and for our baseline LDA classifier. We again present the CAs for each participant as well as the mean for each group of participants and the overall mean in the final row. The best-performing method for each participant, including ties, is highlighted in bold. We use the hyperparameters established in Section 4.1 for each of our neural networks except for the number of training iterations, which is established on an individual basis using our cross-validation procedure over the training partition and is shown alongside the CA for each participant and classifier. In Figure 4.19, we again show a visualization of Table 4.3 that is similar to Figure 4.18

Table 4.3: Test classification accuracies and regularization parameters for our baseline LDA and proposed CNN classifiers. Results are reported for both subjects with impairments in the laboratory and subjects without impairments in their homes. The means for each group of participants are shown in the final row of each section and the overall means are shown in the final row. The overall means are shown in the final row. The highest CA for each row, including ties, is shown in bold.

	Participant	LDA	TDNN	Iter.	TDNN-TR	Iter.	CNN-FC	Iter.	CNN-LA	Iter.	CNN-TR	Iter.
No Impairment	1	**85.00**	80.00	860	**85.00**	2300	40.00	115	80.00	700	**85.00**	1040
	2	55.00	60.00	860	**70.00**	160	20.00	10	**70.00**	680	55.00	1220
	3	40.00	75.00	740	60.00	1280	10.00	60	60.00	280	**80.00**	220
	4	55.00	55.00	1340	55.00	320	25.00	30	60.00	540	**65.00**	260
	5	70.00	85.00	800	80.00	200	55.00	110	80.00	220	**90.00**	540
	6	**40.00**	30.00	140	30.00	120	**40.00**	180	**40.00**	320	30.00	20
	7	65.00	**75.00**	160	**75.00**	700	30.00	140	70.00	300	**75.00**	660
	8	45.00	55.00	1500	**65.00**	1140	20.00	0	40.00	660	60.00	820
	9	50.00	50.00	540	**55.00**	1160	15.00	0	45.00	460	**55.00**	1520
	10	**25.00**	**25.00**	1520	**25.00**	1360	**25.00**	50	**25.00**	1380	**25.00**	80
	Mean	53.00	59.00	846	60.00	874	28.00	70	57.00	554	**62.00**	638
Impairment	11	25.00	45.00	180	40.00	600	35.00	345	45.00	280	**50.00**	180
	12	80.00	80.00	340	85.00	700	15.00	25	**90.00**	1240	**90.00**	440
	13	25.00	25.00	1200	25.00	620	**35.00**	245	25.00	1000	30.00	660
	14	30.00	25.00	40	20.00	240	**35.00**	20	30.00	60	20.00	480
	Mean	40.00	43.75	440	42.50	540	30.00	159	**47.50**	645	**47.50**	440
All	Mean	49.29	54.64	730	55.00	779	28.57	95	54.29	580	**57.86**	581

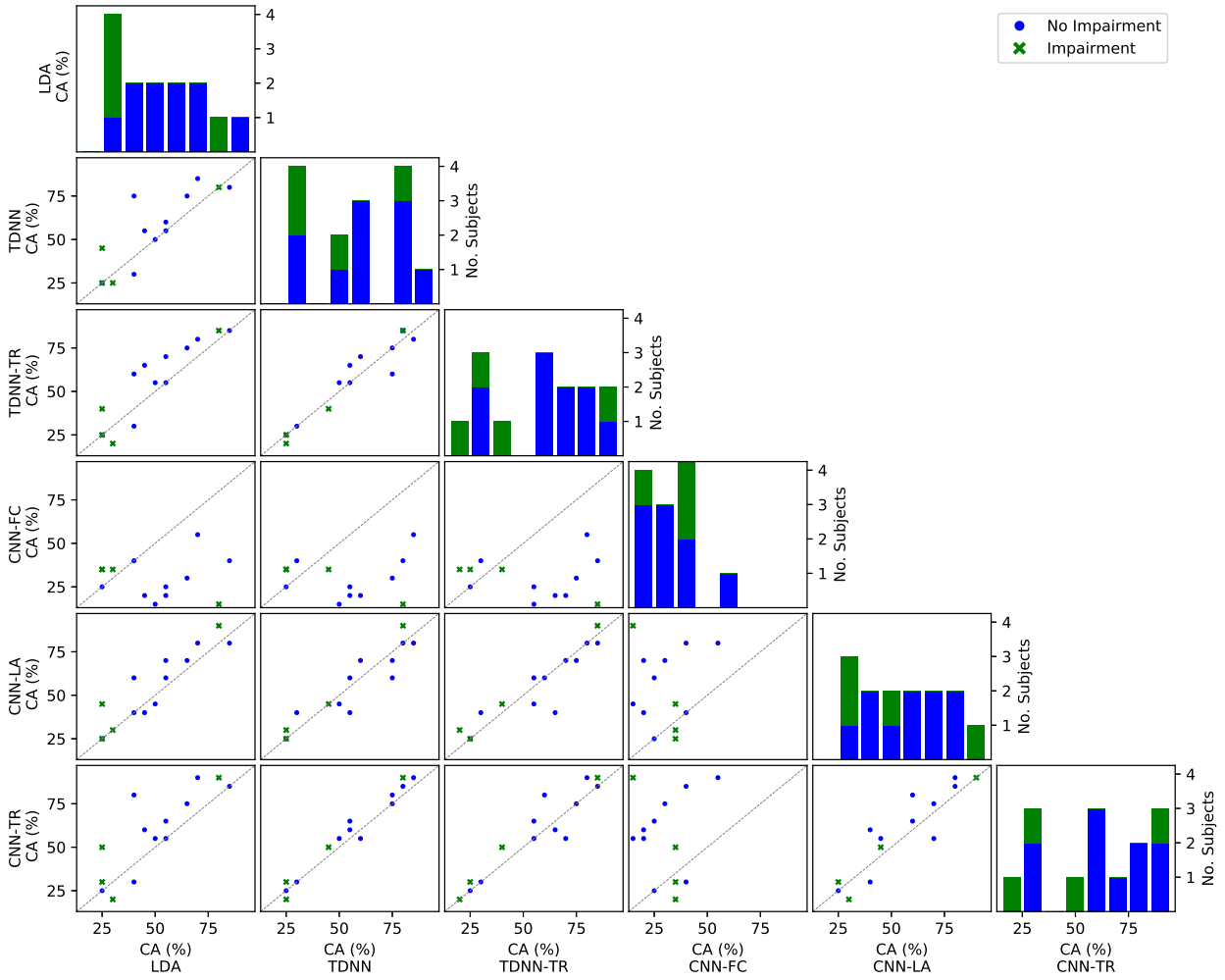


Figure 4.19: The distributions and pairwise CAs for our baseline LDA and proposed CNN classifiers. The plots along the diagonal, show histograms of the distribution of test CAs for each participant. The off-diagonal scatter plots show the relative CAs for each subject and pair of classifiers.

with histograms of the CAs in the plots along the diagonal and pairwise scatter plots of the individual CAs in the off-diagonal plots.

First, notice that CNN-FC achieves an overall mean CA of 28.57%, which is only slightly higher than our expected random CA of 25%. A one-sample, one-sided Wilcoxon test fails to conclude that the mean CA for CNN-FC is higher than 25% ($p = 0.16$). Also note that CNN-FC typically requires aggressive regularization, terminating after only 95 iterations on average. This is consistent with our assertions in Section 4.1.3 that CNN-FC contains an excessive number

of free parameters and is often subject to catastrophic overfitting. All of our approaches that utilize label aggregation (TDNN, TDNN-TR, CNN-LA, CNN-TR), on the other hand, achieve overall mean CAs that are higher than the expected random of 25% with statistical significance ($p < 0.002$). These approaches also typically require much less aggressive regularization, 580–779 training iterations on average. These observations offer strong evidence for the use of label aggregation readout layers instead of dense, fully connected readout layers.

Next, recall that TDNN can be viewed as a single-layer incarnation of CNN-LA and notice that both methods achieve comparable CAs. The overall mean CA for TDNN is slightly higher than CNN-LA (2.00% difference) for the group of participants without impairments and slightly lower (3.75% difference) for the group of participants with impairments and both methods perform comparably overall (0.35% difference). A paired, two-sided Wilcoxon test without correction for multiple comparisons fails to show a statistically significant difference between these two methods ($p = 1.00$).[13] This suggests that, in the absence of transfer learning, incorporating multiple layers to form a deep network may *not* offer improved performance relative to a single-layer network with more hidden units and a wider convolutional kernel. These observations can be explained by the fact that our data are undersampled. Without a sufficient number of examples to learn from, CNN-LA may be unable to learn the hierarchical and multiscale representations that are permitted it's multilayer architecture.

When transfer learning is incorporated into our TDNNs, we see a small but statistically insignificant increase in the overall mean CA for TDNN-TR versus TDNN (0.36% difference; $p = 0.76$). Also note that while TDNN-TR achieves a higher mean CA relative to TDNN for the group of participants without impairments (1.00% difference) it also achieves a lower mean CA (1.25% difference) for the group of participants with impairments. This suggests that our transfer learning approach offers a minimal, if any, improvement when incorporated into our single-layer TDNN architecture and especially under realistic operating conditions.

[13] The p-values for all of our pairwise Wilcoxon tests are enumerated in Appendix A.

When incorporating transfer learning into our multilayer CNN-LA architecture we see a higher overall mean CA for CNN-TR than for CNN-LA (3.57% difference). CNN-TR outperforms CNN-LA for the group of participants without impairments (5.00% difference) and matches CNN-LA for the group of participants with impairments. Although our paired Wilcoxon tests are unable to show a statistically significant difference between the overall mean CAs for CNN-LA and CNN-TR ($p = 0.28$) or between TDNN-TR and CNN-TR ($p = 0.23$), we are able to conclude a statistically significant difference between TDNN and CNN-TR ($p = 0.03$). Note, however, that this conclusion of statistical significance should be met with caution since we are not correcting for multiple comparisons. If we perform a correction for multiple comparisons using Holm's method for all of our baseline and proposed approaches, then our comparison of overall mean CAs for TDNN versus CNN-TR fails to be significant ($p = 0.67$). Note, however, that our pairwise tests with corrections for multiple comparisons have low statistical power due to our relatively low sample size and the number of methods that we are comparing. Despite this lack of statistical significance, we feel that the larger improvement in CA observed when introducing transfer learning into our multilayer CNNs relative to our our single layer TDNNs offers supporting evidence for our claim that multilayer networks are better able to take advantage of the additional information provided by transfer learning.

Also notice that the CAs for CNN-TR appear to have a more bimodal or multimodal distribution than either CNN-LA or LDA, which have more skewed distributions. Upon examination of the relative CAs found in the scatter plots of Figure 4.19, we can see that CNN-TR tends to yield the largest improvements in performance for participants that also attain high CAs for other methods while resulting in modest performance decreases for those participants that tend to attain low CAs for other methods, i.e., subjects 6, 10, 13 and 14. This suggests that introducing transfer learning into our deep CNNs offers the largest benefit for participants that tend to attain high CAs in general. This might also suggest that there tends to be more commonality in the EEG signals produced by participants for which our classifiers tend to perform well overall.

Although it is difficult to establish the exact reason why some subjects tend to perform poorly across all methods, it should be noted that the inability for some users to attain acceptable control of BCIs has been observed by other researchers [133, 134]. Possible causes include environmental noise and artifacts, the manner in which the mental tasks are performed and individual differences in EEG signals or brain organization. It is worth noting, however, that we have observed a higher incidence of high-frequency noise among participants that tend to perform poorly across methods, indicating that there may have been poor connections and signal quality for some of these participants. We have also noticed that there tends to be lower overall signal powers in the α range (8–14Hz) for participants that perform poorly across methods, which might suggest that a lack of attention could be related to poor performance for some participants. These possibilities will be further explored in Section 4.3.1.

When comparing our baseline and proposed methods, we see that all of our CNN-based classifiers achieve higher observed mean CAs than LDA, with the exception of CNN-FC, for both groups of participants and overall. CNN-TR achieves the highest overall mean CA of 57.86%, which is 8.57% higher than LDA. CNN-TR achieves an improvement of 9.00% over LDA for the group of participants without impairments and 7.50% for the group of participants with impairments. It is also worth noting that if we exclude subjects 6, 10, 13 and 14, all of which never attain higher than 40.00% CA, then LDA achieves a CA of 57.00% and CNN-TR achieves 70.50%, which is a 13.50% improvement. Our paired Wilcoxon tests, which are fully enumerated in Appendix A, indicate that CNN-TR is significantly better than LDA ($p = 0.04$), TDNN ($p = 0.03$) and CNN-FC ($p < 0.01$) but not significantly better than TDNN-TR ($p = 0.23$) or CNN-LA ($p = 0.29$).[14] When examining the performance of individual subjects, we see that CNN-TR achieves the highest CA or ties with other methods for nine out of the 14 subjects. In contrast, LDA ties for the highest CA for only three of the 14 subjects.

[14] These p values drop below the level of significance if we apply Holm's correction for multiple comparisons across all of our baseline and proposed methods.

Overall, these results demonstrate that TDNNs and CNNs with label aggregation readout layers offer a considerable improvement in CA over our baseline methods. Additionally, the consistency with which CNN-TR outperforms our other CNN-based methods offers strong evidence that combining multiple layers with transfer learning further improves the performance of these methods. These conclusions are also consistent with the incremental improvements in validation performance that we found during our model selection experiments in Section 4.1 and align with our analytical assessments in Section 3.3.

4.2.3 Comparison with External Studies

In addition to comparing our proposed methods with our baseline classifiers, it may be useful to perform indirect comparisons with relevant studies performed by other research groups. These types of comparisons are difficult, however, due to differences in participants, mental tasks, environmental conditions, EEG acquisition systems and the amount of training data utilized. Nevertheless, we believe that it is worthwhile to perform approximate comparisons with external studies in order to establish how our methods perform relative to the state of the art.

Although raw CA is a valuable performance metric because it conveys the fraction of decisions that were correctly made by a BCI, it does not directly account for the total number of possible class labels or the rate at which labels are assigned. It is difficult to compare the performance of BCIs across experimental paradigms and to get a sense of the amount of information that a BCI user might be able to communicate using only CA. For these reasons, we also present our final test results in terms of Information Transfer Rates (ITRs) in Table 4.4. Recall that ITR, which was defined in Section 3.5.4, incorporates both the number of class labels and the rate at which the classifier makes decisions in order to establish an information-based performance metric with units of bits-per-minute (bpm). It is important to note, however, that both ITR and CA can be sensitive to changes in the rate at which the classifier assigns labels. This phenomenon will be closely examined in Section 4.2.4. Nevertheless, we feel that ITR is

Table 4.4: Test information transfer rates, in bits-per-minute, and regularization parameters for our baseline and proposed CNN classifiers. Results are reported for both subjects with impairments in the laboratory and subjects without impairments in their homes. The means for each group of participants are shown in the final row of each section and the overall means are shown in the final row. The highest ITR for each row, including ties, is shown in bold.

	Participant	LDA	TDNN	TDNN-TR	CNN-FC	CNN-LA	CNN-TR
No Impairment	1	**34.57**	28.83	**34.57**	2.34	28.83	**34.57**
	2	8.82	11.85	**19.30**	0.30	**19.30**	8.82
	3	2.34	23.77	11.85	3.14	11.85	**28.83**
	4	8.82	8.82	8.82	0.00	11.85	**15.34**
	5	19.30	34.57	28.83	8.82	28.83	**41.18**
	6	**2.34**	0.28	0.28	**2.34**	**2.34**	0.28
	7	15.34	**23.77**	**23.77**	0.28	19.30	**23.77**
	8	4.06	8.82	**15.34**	0.30	2.34	11.85
	9	6.23	6.23	**8.82**	1.29	4.06	**8.82**
	10	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	Mean	10.18	14.69	15.16	1.88	12.87	**17.35**
Impairment	11	0.00	4.06	2.34	1.07	4.06	**6.23**
	12	28.83	28.83	34.57	1.29	**41.18**	**41.18**
	13	0.00	0.00	0.00	**1.07**	0.00	0.28
	14	0.28	0.00	0.30	**1.07**	0.28	0.30
	Mean	7.28	8.22	9.30	1.13	11.38	**12.00**
All	Mean	9.35	12.85	13.49	1.67	12.44	**15.82**

a more appropriate metric than CA for comparing results across studies because it does utilize both the decision rate and the number of classes used.

From Table 4.4, we can see that our baseline LDA classifier achieves a mean ITR of 10.18 bpm for the group of participants without impairments, 7.28 bpm for those with impairments and 9.35 bpm overall. CNN-TR, on the other hand, achieves an ITR of 17.35 bpm for the group of participants without impairments, 12.00 bpm for the group of participants with impairments and 15.82 bpm overall, which is an improvement over LDA of 6.47 bpm in overall mean ITR. ITR varies widely across participants with some achieving only 0.00 bpm[15] across all methods and others achieving ITRs as high as 41.18 bpm for CNN-TR. Note that there is a monotonic

[15]We assign 0.00 bpm for accuracies that are less than 25% with the assumption that a reversal of class labels is not particularly useful in a BCI system.

function mapping CA to ITR, which means that the participants and classifiers that performed best in Table 4.4 also perform best in Table 4.3.

In an early study by Keirn and Aunon, a combination of PSDs with QDA were explored for classifying EEG signals recorded during mental tasks [33]. This investigation involved five subjects in a controlled laboratory, including a sound-proof booth, a low-impedance, passive EEG system and an artifact rejection algorithm. Each subject performed five different mental tasks that were somewhat similar to our tasks: baseline rest, complex problem solving, geometric figure rotation, mental letter composing and visual counting. Only binary classification was reported for pairs of mental tasks using two-second EEG segments. These experiments achieved a mean two-class CA of 81% averaged across all participants and task pairs. Converting this value using Equation (3.53), we arrive at an ITR of 8.96 bpm. This is slightly lower than the overall mean ITR of 9.35 bpm achieved by our baseline LDA classifier, despite the fact that we utilize a portable EEG system, include participants with impairments in their home environments and do not utilize artifact rejection.

As was discussed in Section 2.1.1, PSDs alone are unable to capture spatial patterns in the form of phase differences across channels. In order to address this limitation, Gysels, et al., explored the use of Phase Locking Value (PLV), which is a measure of phase synchronization across two channels, in combination Spectral Coherence (SC) and PSDs, which are measures of signal amplitude across a frequency spectrum [73]. Since PLV and SC are used to compare pairs of signals, this approach can lead to a large number of features. In order to address this, Gysels, et al., performed an extensive and partially manual feature selection procedure in order to select a combination of PLV, SC and PSD features that work well. These features were then passed to linear support vector machines for classification. Their experiments utilized 60 minutes of EEG data recorded from five participants over the course of three days; however, the data for two participants was entirely discarded because these two participants did not achieve good performance for any of their methods. Data were recorded using a high-impedance, active EEG system and no artifact rejection was performed. One-second EEG segments were used and

a five-fold cross validation was used to evaluate performance. Note that a fully separate test set was not withheld and so it is possible that their manual feature selection procedure has lead to their results being overly optimistic for their particular dataset. For the three remaining participants, classification accuracies of 67.81%, 64.49% and 55.14% were achieved for the best combination of features. According to our calculations, the ITRs are then 21.40bpm, 17.48bpm and 8.64bpm with a mean of 15.84bpm. Although the mean ITR achieved by Gysels, et al., is roughly equivalent to the ITR achieved by our CNN-TR method, we have utilized considerably less training data and have not discarded the data for low-performing subjects. Note that if we exclude the data for participants 6, 10, 13 and 14, who achieve low performance across all of our methods and have a high incidence of noisy EEG segments, as noted in the previous section, CNN-TR achieves a mean ITR of 19.72bpm.

In another study by Millán, et al., PSD-based signal representations were combined with a variation of a mixture-of-Gaussians type classifier [34]. In this study, two participants using a portable EEG system performed three mental tasks: relax, left hand movement and right hand movement. At one-second intervals, overlapping by 50%, these two subjects achieved CAs of 52% and 63%. Accuracies were then increased to greater than 70% for each participant by using feedback during multiple training sessions over the course of several days and by including cube rotation and arithmetic subtraction mental tasks. Again using our own conversion, we arrive at ITRs of 6.37 bpm and 15.86 bpm for the two participants before using feedback and then greater than 24 bpm after training with feedback. The individual ITRs achieved in this study before utilizing feedback appear to roughly range from the mean to the peak ITRs achieved by our baseline LDA classifier. The ITRs achieved after training with feedback and additional mental tasks exceed our mean ITRs for any method but are less than the performance achieved by CNN-TR for subjects 1, 3, 5 and 12.

In a recent study by Lawhern, et al., a CNN architecture that utilizes label aggregation readout layers, similar to our approach, was investigated for classifying EEG signals recorded dur-

ing motor imagery tasks [52].[16] This network architecture, which the authors refer to as EEGNet, was briefly discussed in Section 2.2.3. These experiments utilized the widely available BCI Competition IV 2a dataset, which was collected from nine participants without impairments using a clinical EEG acquisition system in a laboratory environment. This dataset consists of four motor imagery tasks: left hand, right hand, both feet and tongue [135, 136]. The data is split into two-second segments and was preprocessed using the methods of Schirrmeister, et al., which includes rejection of EEG segments that contain high-amplitude artifacts [53]. The authors achieve a mean CA of approximately 70%, which, by our calculation, is 19.30 bpm. The authors also note that this performance is roughly equivalent to a refined CSP-based classification algorithm. Although this ITR is higher than the overall mean CA achieved by CNN-TR, it is unclear if it is able to match our peak ITRs of 41.18 bpm achieved by subjects 5 and 12. Additionally, note that CNN-TR achieves a nearly equivalent ITR of 19.72 bpm if we again exclude our low-performing subjects. Also note that the work by Lawhern, et al., utilizes only motor imagery tasks, does not use a portable EEG system and does not include participants with impairments or in their homes.

Despite the challenges associated with comparing methods across different studies and experimental paradigms, these comparative analyses allow us to draw several conclusions. First of all, our baseline classifiers are competitive with other studies that have utilized PSD-based signal representations. Our baseline LDA classifier exceeds the mean ITRs achieved by Keirn and Aunon and aligns with the range of results attained by the two participants in the study by Millán, et al. This supports our claim that our baseline LDA classifier is representative of the performance expected from EEG classifiers that rely strictly on PSDs. Second, we can see that our proposed CNN-TR approach generally outperforms these PSD-based methods, achieving a considerably higher mean ITR than Keirn and Aunon's results and a mean ITR that is higher than one participant in Millán's study and comparable to the other participant before train-

[16] The work by Lawhern, et al., on EEGNet was performed concurrently with our research and independently draws similar conclusions about combining relatively small CNNs with label aggregation readout layers.

ing with feedback. CNN-TR also achieves ITRs for some individuals that are higher than the results achieved by Millán after repeated training with interactive feedback. This further supports our claim that CNN-TR generally outperforms PSD-based methods. Third, it appears that the introduction of additional spatial information, as demonstrated by Gysels, et al, can yield a considerable improvement in performance. This supports our claim in Section 1.1 that sophisticated spatiotemporal patterns are important as well as our claim in Section 2.1.1 that PSDs alone are not well suited for capturing these types of patterns. As we have previously asserted in Section 2.2, however, manual feature engineering procedures are likely not practical for use in real-world BCIs. Finally, we see that CNN-TR performs comparably to the CNNs proposed by Lawhern, et al., achieving a similar mean ITR when we omit participants that attain low performance across all methods. Although Lawhern, et al., only explored motor imagery tasks in a laboratory environment, this work provides corroborating evidence that CNNs with few free parameters and label aggregation readout layers perform well for classifying EEG signals in asynchronous paradigms. Although these comparisons are admittedly limited by our ability to compare methods across experimental paradigms, we believe that they provide supporting evidence that our baseline methods provide an adequate benchmark and that our proposed CNN-TR architecture achieves results that are on par with or better than other state-of-the-art methods for classifying asynchronous EEG signals.

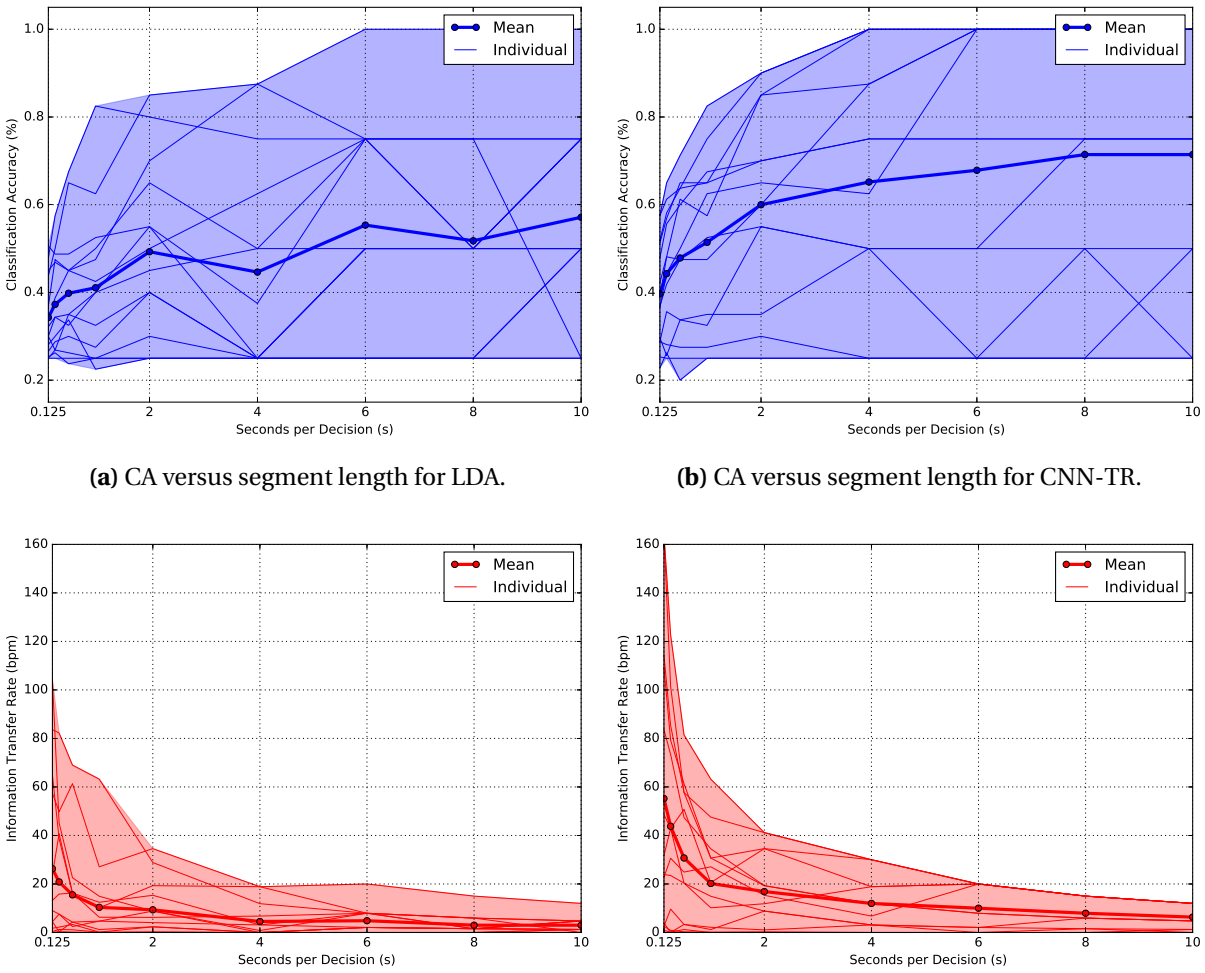
4.2.4 Varying the Decision Rate

Thus far, all of our experiments have utilized EEG segments with a length of two seconds. In other words, we have simulated a BCI system that takes an action every two seconds, making our decision rate 0.5 decisions per second or 30 decisions per minute. In our experience, this is generally the fastest pace at which a user can be reasonably expected to switch between the mental tasks that we have defined. It is certainly possible, however, for a BCI to be controlled at a slower rate and it may be possible to achieve much faster rates of control after extensive training with feedback and, potentially, when using alternate mental tasks.

As was discussed in the previous section, CA alone does not take into account the rate at which decisions are made. Although ITR does incorporate the decision rate, it is possible and often the case that ITR changes as the decision rate of the classifier is varied. It is clear from Equation (3.53) that there is a linear relationship between decision rate and ITR, i.e., doubling the decision rate also doubles the ITR, provided that the accuracy and number classes does not change. Longer EEG segments contain more information, however, and changing the length of our EEG segments is also likely to alter the accuracy of the classifier. In other words, if using a slower decision rate and longer EEG segments increases our CA by introducing additional information or allowing for better noise or dimensionality reduction, then *both* the CA and ITR will change.

Characterizing the tradeoffs between decision rate, CA and ITR is important when designing practical BCI systems because both the accuracy and responsiveness of the system influence user experience and the types of BCI applications that can be implemented. It may be desirable, for instance, to develop a typing application using a method that achieves high CA but requires a slow decision rate. For applications that aide in communication on a daily basis, users are likely to feel a high degree of frustration if they are required to correct the system often and are likely to prefer a slower but more reliable BCI. On the other hand, it may be preferable to sacrifice a degree of accuracy in exchange for faster responses when designing applications such as real-time video games. In this case, sub-second response times may be required in order to deliver an entertaining experience, even if these faster responses lead to a decrease in reliability.

Note that one of our primary justifications for using CNNs with label aggregation is that they appear to be well suited for use in applications that require continuous or near real-time control. While PSD-based methods require relatively long EEG segments in order to attain both narrow frequency bins and low levels of noise, CNNs with label aggregation utilize relatively narrow windows of time while relying on the learned filters for noise reduction and are capable of assigning class labels at arbitrary intervals.



(a) CA versus segment length for LDA.

(b) CA versus segment length for CNN-TR.

(c) ITR versus segment length for LDA.

(d) ITR versus segment length for CNN-TR.

Figure 4.20: Classifier performance versus signal segment length. The thin lines show individual performance while the thick lines show the means. The horizontal axis shows the length of the signal segment, i.e., smaller values correspond to decisions being made more quickly. The CA for (a) LDA and (b) CNN-TR increases with a diminishing return as the signal segment increases. Notice that CNN-TR achieves higher mean CAs for both short and long segments but the largest difference is for longer segments. The ITRs for (c) LDA and (d) CNN-TR decrease rapidly at first and levels off as the segment length increases. Notice that the largest difference in ITR between both methods is for short segments.

In Figure 4.20, we explore how the rate at which decisions are made affects performance by examining both CA and ITR as the length of our EEG segments is varied.[17] In Figure 4.20a and Figure 4.20b we see test CA versus seconds per decision for LDA and CNN-TR, respectively.[18] The thin lines show the performance for each individual participant while the thick lines show the means across all participants and the shaded regions show the ranges. Notice that the CA for both methods increases as the length of the EEG segments grows longer. There is a diminishing return, however, since there is an increasingly smaller improvement in CA as our segment length is increased. Also notice that CNN-TR achieves a higher mean CA for all segment lengths with the largest difference being for long segments. At $\frac{1}{8}$ seconds per decision, LDA achieves a mean CA of 34.29% while CNN-TR achieves 39.69% and at 10 seconds per decision, LDA achieves 57.41% while CNN-TR achieves 71.43%. After six seconds per decision, one individual participant achieves 100% CA for LDA and six individuals achieve 100% CA for CNN-TR. The curve is also somewhat smoother for CNN-TR than for LDA. This can be explained by the fact that PSDs are more sensitive to the length of our segments since the DFT changes and more spans are averaged. CNN-TR, on the other hand, simply aggregates more labels in its readout layer as the segment length grows longer. Note that CNN-TR does not have to be retrained in order to vary the decision rate since the input features do not change.

In Figure 4.20c and Figure 4.20d, we see how test ITR changes as we vary seconds per decision. Again, the thin lines show individual performance and the thick lines show the mean across all participants. Notice that our mean ITR decreases for both methods as the segment length is increased, which is contrary to the behavior we observed for CA. In other words, reducing the decision rate of our classifiers also reduces the amount of information that a BCI user is expected to convey, despite the fact that a slower decision rate also results in more accurate classification. This means that longer EEG segments, thus a slower decision rate, results

[17] We have found it to be more intuitive to vary the length of our EEG segments, which is seconds per decisions, rather the decision rate, which is the reciprocal decisions per second.

[18] Since the span used by Welch's method for LDA cannot exceed the length of the segment, we set our span to be the minimum of the segment length and our predetermined span of 0.25.

in a more reliable BCI system while a faster decision rate results in a more responsive but less reliable BCI system. The largest difference in ITR between LDA and CNN-TR is observed for the shortest EEG segments. At $\frac{1}{8}$ seconds per decision, LDA achieves a mean ITR of 26.27 bpm and a peak of 104.42 bpm while CNN-TR achieves a mean ITR of 55.22 bpm and a peak of 164.49 bpm. At 10 seconds per decision, LDA achieves a mean ITR of 3.00 bpm and a peak of 12.00 bpm while CNN-TR achieves a mean of 6.34 bpm and a peak of 12.00 bpm.

This analysis leads us to several conclusions. First of all, the increase in CA that results from increasing our segment lengths is overpowered by the reduction in information transfer for both LDA and CNN-TR. In other words, there is a definite tradeoff between the accuracy and responsiveness for these classification algorithms when the decision rate is varied. While long segment lengths result in a slow but accurate BCI, short segment lengths result in a fast and responsive BCI with lower accuracy. This is an important aspect of performance for any EEG signal classifier that should always be explored. Second, it appears that CNN-TR outperforms LDA for a wide range of decision rates. The largest difference in accuracy between the two methods is for long signal segments, which suggests that CNN-TR is likely to be better than LDA for constructing highly accurate BCIs. Similarly the largest difference in ITR was found to be for very short signal segments, suggesting that BCIs that require responsiveness and large amounts of information to be conveyed are also likely to achieve the best results with CNN-TR. The difference in ITR for short segments is considerable, which supports our claim that CNNs with label aggregation are well suited for use in near real-time BCIs.

4.2.5 Mental Tasks

In previous sections, we have focused on performance metrics that were computed across all classes. In order to examine the relative performance of each of our mental tasks across both our baseline and proposed classifiers, we will now present several types of class-wise performance metrics that will highlight the differences between each of our mental tasks. Recall from Section 3.5.3 that our mental tasks were chosen by reviewing previous studies and using

our intuition about which tasks are likely to elicit a variety of patterns across different regions of the brain. Although a number of studies that investigate mental-task BCIs do provide some discussion and justification of the mental tasks used [33,34], there are currently only a few studies that provide an in-depth comparison of the relative performance of different mental tasks or that attempt to identify which mental tasks perform best [36, 56]. We are not aware of any research to date that investigates which mental tasks perform best when using CNNs or other types of artificial neural networks. For these reasons, we feel that it is important to perform a thorough investigation into which of our mental tasks tend to perform better than others.

In Table 4.5, we see confusion matrices for LDA, TDNN and CNN-TR computed using the labels for all segments in the test partitions for all subjects using corresponding individualized classifiers. These three methods represent our best-performing baseline classifier, CNN without transfer learning and CNN with transfer learning, respectively. In these confusion matrices the rows indicate the label that was predicted by the classifier while the columns indicate the actual class labels. Each cell contains the percent of two-second EEG segments, across all participants, that were assigned the class label of the row over the total number of segments belonging to the actual class label of the column. This means that all columns sum to 100% and the cells along the diagonal represent the true positive rate, or sensitivity, for each mental task. In other words, each diagonal cell shows the percent of segments that belong to the task and were correctly labeled as such while the off diagonal cells show the percent of segments that belong to the task of the column and were incorrectly labeled as, or confused with, the task of the row. We also formally define confusion matrices in Equation (3.52) in Section 3.5.4.

In Table 4.5a, we see the overall test confusion matrix for LDA. Note that the highest sensitivity is 62.86% for the Count task followed by 48.57% for Rotate and Song and 37.14% for Fist. Also notice that many segments are often mislabeled as Count or Song and that Song is also often mislabeled as Rotate. It is rare, however, for Count or Song to be mislabeled as Fist. This indicates that LDA tends to label too many segments as Count and Song while often failing to identify segments that belong to the class Fist.

Table 4.5: Confusion matrices computed using the labels from all subjects and mental tasks. The true positive rate, or sensitivity, for each task is shown in bold along the diagonal. The columns indicate the actual task being performed and the rows indicate the predicted class label. The values in each cell are the percentage of actual segments in the test partition that were assigned the predicted label, i.e., 100 × the number of predicted segments / total actual segments for the task.

(a) Confusion matrix for LDA.

		Actual (%)			
		Count	Fist	Rotate	Song
Predicted (%)	Count	**62.86**	22.86	11.43	24.29
	Fist	4.29	**37.14**	12.86	5.71
	Rotate	11.43	17.14	**48.57**	21.43
	Song	21.43	22.86	27.14	**48.57**

(b) Confusion matrix for TDNN.

		Actual (%)			
		Count	Fist	Rotate	Song
Predicted (%)	Count	**78.57**	34.29	15.71	18.57
	Fist	5.71	**40.00**	18.57	8.57
	Rotate	10.00	11.43	**48.57**	21.43
	Song	5.71	14.29	17.14	**51.43**

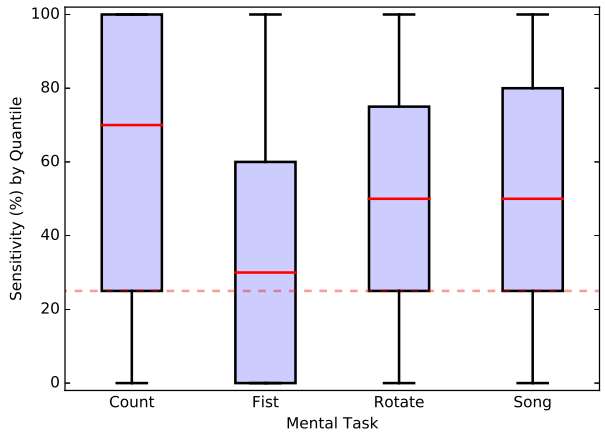
(c) Confusion matrix for CNN-TR.

		Actual (%)			
		Count	Fist	Rotate	Song
Predicted (%)	Count	**75.71**	31.43	15.71	12.86
	Fist	5.71	**41.43**	10.00	5.71
	Rotate	11.43	17.14	**62.86**	30.00
	Song	7.14	10.00	11.43	**51.43**

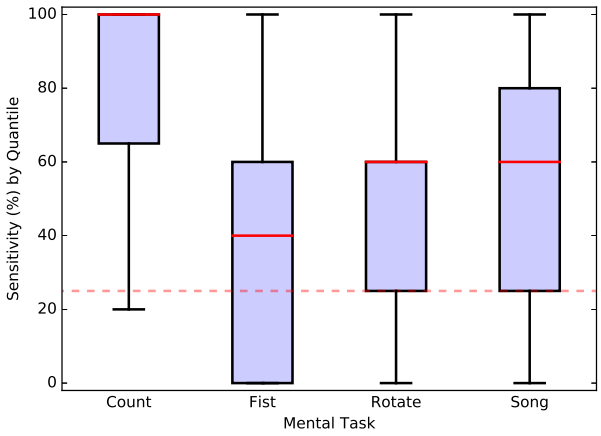
In Table 4.5b, we see the overall test confusion matrix for TDNN. The highest sensitivity is now 78.57% for Count followed by 51.43% for Song, 48.57% for Rotate and 40.00% for Fist. The confusion matrix for TDNN is similar to LDA in that Count has the highest sensitivity while Fist has the lowest. All sensitivities for TDNN are higher than LDA except for Rotate, which is tied. Fewer segments are now mislabeled as Count and Song across all classes except for Fist and Rotate, which are now mislabeled as Count more often. In general, TDNN is better than LDA at distinguishing between all classes and tends to lean less toward assigning the Count and Song labels with the notable exception of mislabeling Fist as Count.

In Table 4.5c, we see the overall test confusion matrix for CNN-TR. The highest sensitivity is now 75.71% for Count followed by 62.86% for Rotate, 51.83% for Song and 41.43% for Fist. Note that CNN-TR achieves higher sensitivities than LDA and TDNN for all classes except for Count, which is slightly lower than TDNN, and Song, which is tied with TDNN. CNN-TR has only a slightly lower rate than TDNN of mislabeling Fist as Count and frequently mislabels Song as Rotate. The largest improvement seen for CNN-TR over TDNN is for the Rotate task, with other tasks having relatively small changes in performance.

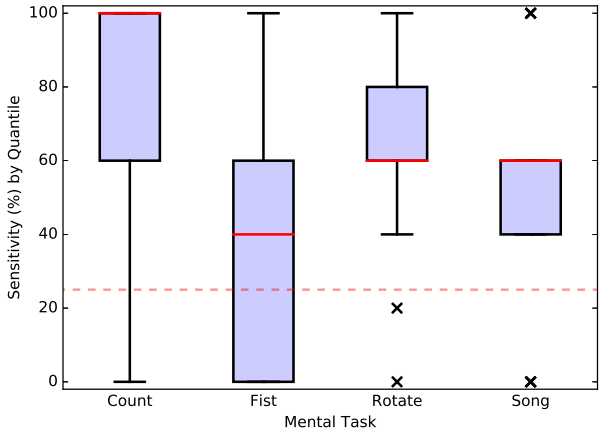
Although the above confusion matrices help to give us a sense of which tasks tend to perform well for all subjects, they do not help us to identify how much variability is seen in the accuracy of the labels for each task across subjects. In Figure 4.21, we see boxplots showing the quantiles of our test sensitivities across participants. In other words, we have computed the confusion matrices for each participant and are examining the distributions of the values along the diagonals. For the Count task, we see a significant improvement for TDNN and CNN over LDA. While 25% of participants achieve a sensitivity of 100% for Count when using LDA, about 50% of participants achieve 100% when using either TDNN or CNN-TR. For the Fist task, the sensitivities are distributed roughly the same for all three methods with roughly 25% of participants achieving a sensitivity of zero and only about 25% of participants achieving a sensitivity above 60%. For the Rotate task, a number of participants achieve a lower sensitivity for TDNN than for LDA; although the median sensitivity is slightly higher. For the Rotate task with



(a) Sensitivity quantiles for LDA.



(b) Sensitivity quantiles for TDNN.



(c) Sensitivity quantiles for CNN-TR.

Figure 4.21: Boxplots showing the quantiles of the distribution of per-task sensitivities for all subjects. The central blue boxes show the second and third quantiles while the tails show the first and fourth quantiles. The medians are indicated by the solid red lines and the expected random of 25% is highlighted by the dashed red line. (a) For LDA, the Count task tends to perform best while the Fist task often performs poorly. (b) TDNN offers an improvement for the Count task for many participants but offers little benefit for other tasks. (c) The introduction of transfer learning in CNN-TR improves the performance of the Rotate task for many participants while offering mixed results for the Song task and little benefit for Fist.

CNN-TR, however, we see a notable improvement for many participants. While 50% of participants achieved less than 60% sensitivity for TDNN, 50% of participants achieve greater than 60% for TDNN-TR. For the Song task, the distribution of sensitivities is roughly equivalent for both LDA and TDNN. For CNN-TR, on the other hand, all but two participants achieved a sensitivity between roughly 40–60% with some participants seeing an improvement relative to the other methods and some participants seeing a reduction in performance.

Our examination of confusion matrices and of our per-task sensitivities leads us to several important conclusions. First of all, EEG segments belonging to the Count task are labeled correctly more often than the other tasks across all three methods and for most participants. This suggests that the count task can often be discriminated from our other mental tasks and is likely well suited for use in BCI systems. Both TDNN and CNN-TR outperform LDA for the Count task, which suggests that spatial or nonlinear information may be valuable. There does not, however, appear to be a benefit for the Count task when adding transfer learning in CNN-TR. The Fist task, on the other hand, performs relatively poorly across all three methods. This is contrary to the prevailing trend in BCI research to focus primarily on motor-imagery tasks and supports our claim from Chapter 1 that mental tasks other than motor-imagery tasks are valuable for use in asynchronous BCIs. Note, however, that the Fist task is often confused for the Count task, which may suggest that there is some similarity between the types of patterns found in signals belonging to both of these classes. It is possible that the Fist task might perform considerably better if the Count task had not been included. For the Rotate task, both LDA and TDNN perform equivalently on average but some participants see slightly better performance when using LDA. This suggests that the additional spatial and nonlinear information that can be utilized by TDNNs are not beneficial for the Rotate task when using limited training data. We do, however, see a considerable improvement in performance for the Rotate task for most participants when we introduce multiple layers and transfer learning with CNN-TR. This suggests that there are common aspects to the patterns produced in EEG signals during the Rotate task across participants and that the hierarchical and multiscale architecture of CNN-TR well suited for capturing

these patterns, provided that transfer learning or sufficient training data are used. For the Song task, there appears to be only a modest benefit to using TDNN over LDA and for CNN-TR there appears to be a benefit for some participants and a detriment for others. This may suggest that power-based features are useful for identifying the Song task for some participants while there may also be some spatial and nonlinear patterns that are common across some, but likely not all, participants.

4.3 Analysis

Now that we have examined the performance achieved by each of our methods, we will continue by investigating the types of patterns present in these EEG signals and how our classifiers learn to leverage these patterns. This type of analysis is important for several reasons. First of all, it is essential that we understand the types of patterns that our classifiers learn to utilize and fail to identify in order to validate and further elucidate the advantages and disadvantages of each of our classification approaches. Examining how our classifiers processes information internally may also lead to new insights into how these methods function in practice, which compliments our mathematical analyses and observations about performance in previous sections. We also seek to gain insights into the nature of EEG signals and how brain activity tends to vary across mental tasks. In addition to providing information about the inner workings of the brain, establishing the differences in the types of patterns produced during each of our mental tasks is valuable for designing, calibrating and understanding BCI systems. Finally, it is important to determine why some participants and mental tasks tend to perform better than others. Understanding how our classifiers function and the types of patterns that they learn to utilize may provide new insights into the capabilities and limitations of BCIs and may lead to new methods for improving future BCI systems.

In order to achieve these goals, we will first analyze our baseline methods by examining aggregate PSDs both across subjects as well as for individuals along side the weights learned by our LDA classifiers. Next, we will examine the patterns leveraged by our single-layer TDNNs by

interpreting their convolutional layers as FIR filters and extracting the frequency responses of these filters. We will then examine the layer-wise outputs of our multilayer CNN-TR networks and we will also use the optimization techniques described in Section 3.3.4 to learn optimal input sequences for these networks. Although analyzing the patterns learned by artificial neural networks is notoriously difficult, we believe that these experiments will demystify the inner workings of our proposed classifiers and provide valuable insights into the types of patterns that they learn to utilize.

4.3.1 Aggregate PSDs

An important first step toward examining the types of patterns found in our EEG dataset and interpreting how our classifiers learn to leverage these patterns is to examine the differences in PSDs that are averaged across participants and mental tasks. This approach is similar to analyses commonly found in cognitive neuroscience research and, since PSDs are inherently time-invariant and straightforward to visualize, they naturally lend themselves to this type of aggregate interpretation. Unfortunately, we do not have enough subjects, and therefore statistical power, to draw conclusions about these PSDs with statistical significance. We can, however, make a number of observations about these averaged PSDs that offer supporting evidence for our other experiments and conjectures and that yield valuable insights into our dataset and how these EEG signals may relate to the human brain as well as environmental noise and artifacts.

Subject-wise PSDs

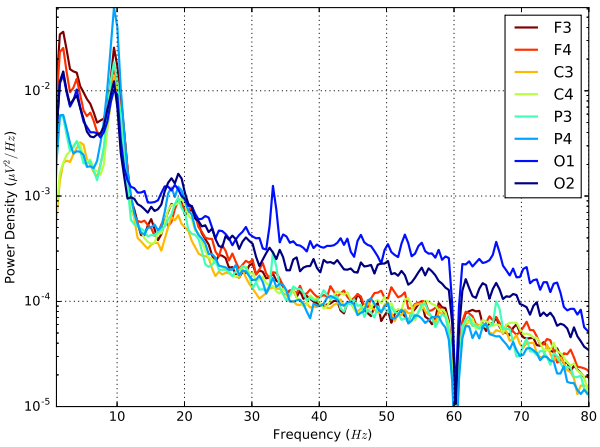
In Figure 4.22, we see PSDs without Welch's method[19] averaged across all tasks and EEG segments for each subject. These figures reinforce several of our assertions about the variability of EEG signals across individuals as well as the quality of the signals for some participants in our dataset. First, notice that there are considerable differences in the magnitudes and precise frequencies of some peaks in these PSDs, especially in the α (8 – 14Hz) and β (14 – 30Hz) bands.

[19] Since we are now averaging many EEG segments and because we are seeking a detailed visualization, rather than dimensionality reduction, we utilize raw periodograms instead of Welch's method.

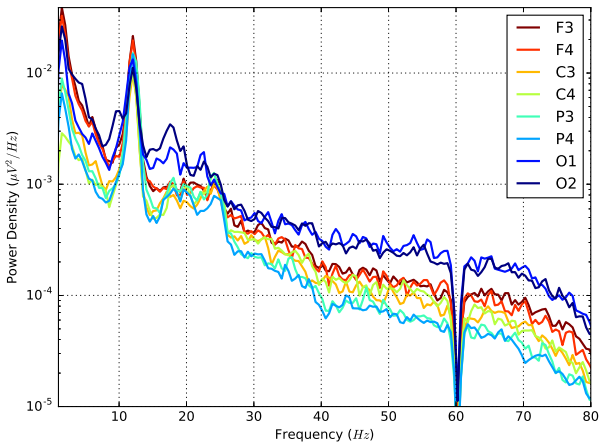
For example, the peak in α power for Subject-1 is near 10Hz and for Subject-2 is near 14Hz and for Subject-7 is closer to 8Hz. This demonstrates that while there is some commonality in the general shape of the power spectrum across subjects, there are also considerable differences among individuals.

Next, notice that there is little, if any, peak in α power relative to neighboring frequencies for subjects 4, 6, 11, 13, 14. These participants also achieve relatively low performance in our classification results presented in Section 4.2. Although the behavior of α rhythms is complicated and not yet fully understood, it is worthwhile to note that α rhythms have been linked to attention; specifically, the inhibition of cortical regions that are not involved in processing a given task and in suppressing distractions [137, 138]. We suspect that the low α powers observed in some participants might be an indicator of a lack of attention or a decreased ability or willingness to filter out distractions. It is also relevant to note that three of our four participants with impairments have notably low power in the α band. Although previous research has indicated some differences in EEG signals for people with Multiple Sclerosis (MS), which affects subjects 13 and 14, there has not been an observed difference in baseline α rhythms for people with MS [139, 140]. It currently remains unclear if the low levels of α power for these individuals is related to their disabilities or environmental factors is simply coincidental. A future study involving more participants might be valuable for investigating this topic.

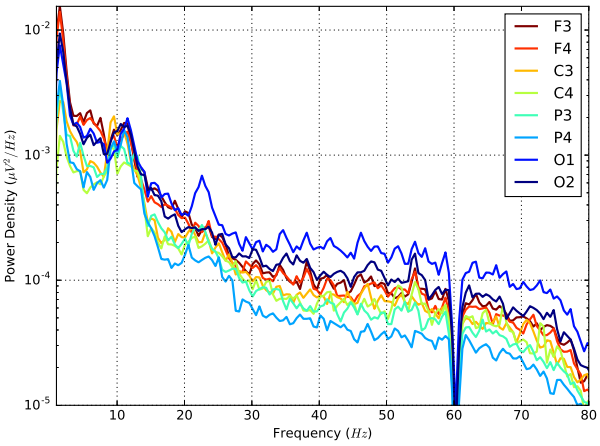
We also observe that there is very high power in some channels for some subjects either across the γ ($>$ 30Hz) band or the entire power spectrum. For example, subjects 1, 2 and 3 have relatively high γ power in in the occipital region, subjects 4 and 11 have high γ power in the frontal region, Subject 14 has high γ power in channels C3 and O1, subjects 5 and 10 have high γ power across all channels and Subject 13 has very high power across the entire frequency spectrum, especially in channel P3. As described by Fries, et. al, distinguishing between γ-band signals that are generated by the brain from artifacts caused by muscle activity can be extremely challenging and is often not entirely possible [141]. The PSDs for subjects 5, 10 and 14 do, however, show characteristic signs of high-amplitude contamination from muscle activ-



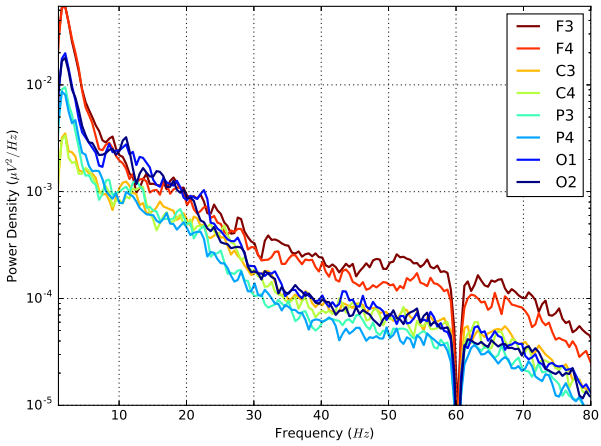
(a) Subject 01



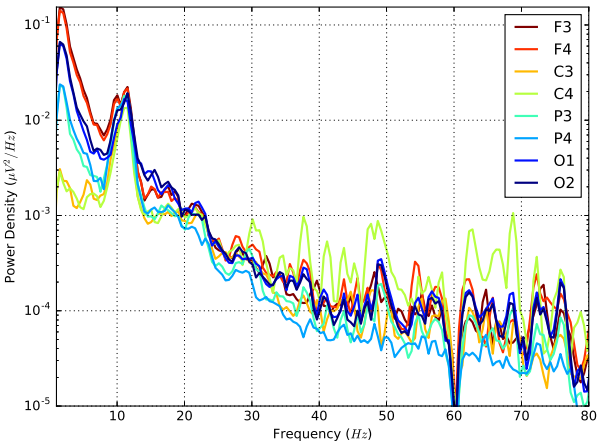
(b) Subject 02



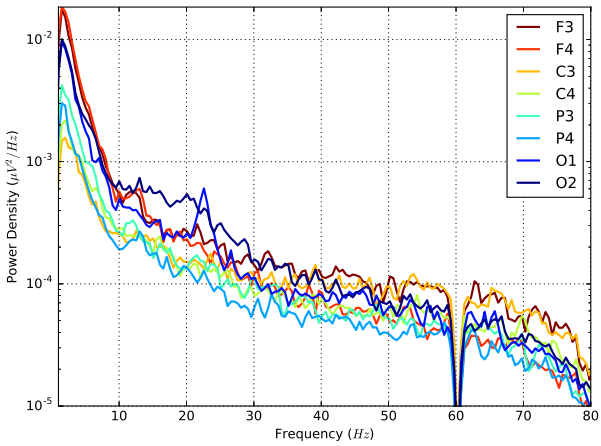
(c) Subject 03



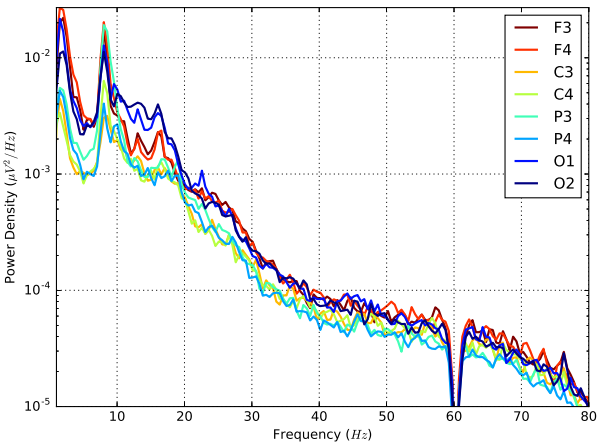
(d) Subject 04



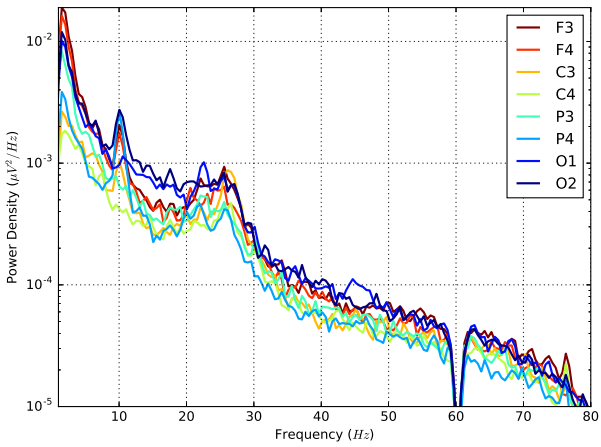
(e) Subject 05



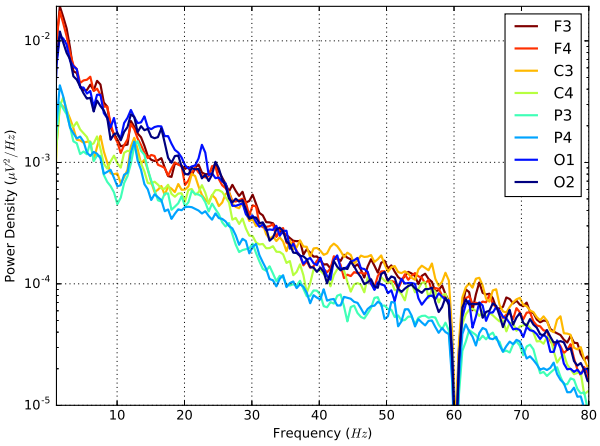
(f) Subject 06



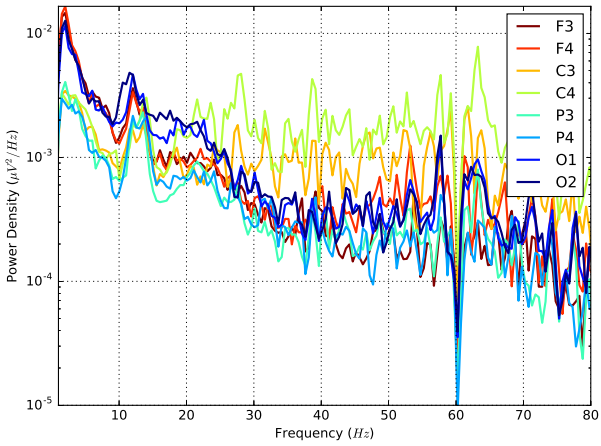
(g) Subject 07



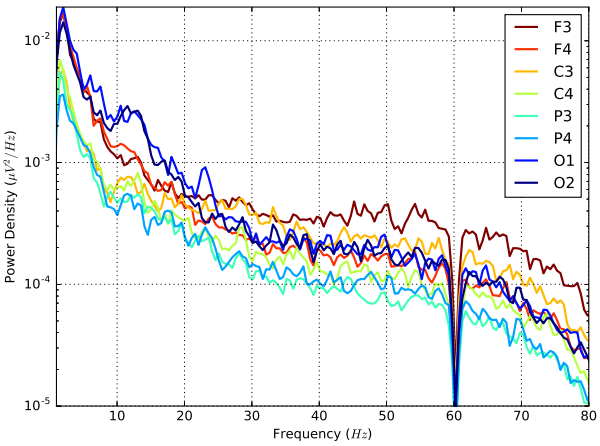
(h) Subject 08



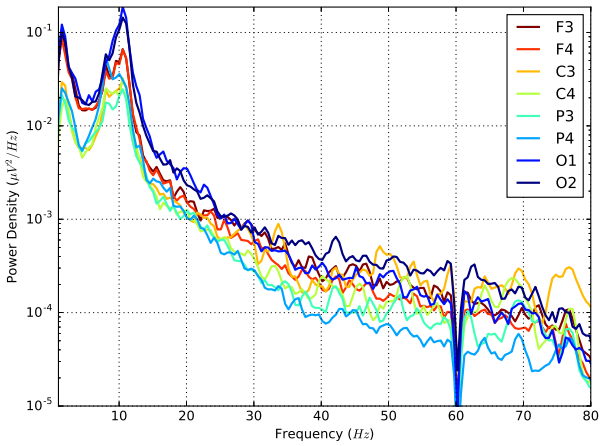
(i) Subject 09



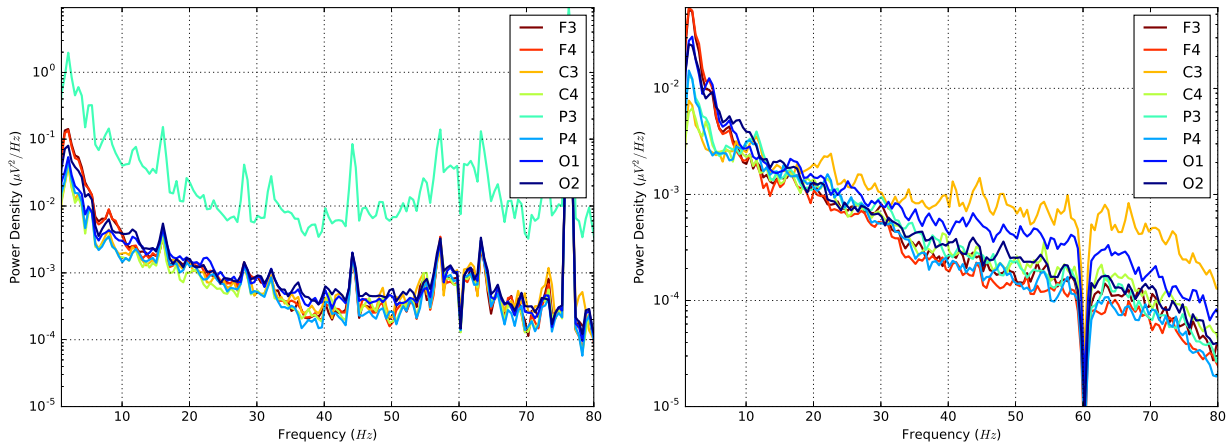
(j) Subject 10



(k) Subject 11



(l) Subject 12



(m) Subject 13

(n) Subject 14

Figure 4.22: Power Spectral Densities (PSDs) averaged across EEG segments in all classes for each individual subject. These plots highlight individual differences in the power content of the signals across participants. For example, α (8 – 14Hz) and β (14 – 30Hz) rhythms vary considerably with respect to power and peak frequency. These figures also show that some participants likely have poor signal quality. For example, Subject 13 appears to have broad-band contamination in site P3 from a poor connection and Subject 10 appears to have significant γ-band (> 30Hz) contamination from muscle artifacts.

ity, including γ power exceeding 1.0×10.0^{-3} μV^2/Hz across multiple recording sites and erratic peaks in the γ band of the PSDs. Recall that subjects 10 and 14 attained low performance in our classification experiments, ranging from 20–35% CA. Subject-5, on the other hand, attained good performance, 70% CA for LDA and 90% CA for CNN-TR, despite this apparent contamination from muscle artifacts. Although it is possible that the classifier for Subject-5 utilizes muscle artifacts to improve classification accuracy, we will see in our analysis of LDA that this model performs reasonable well while not heavily utilizing γ-band activity.

Subjects 11 and 12, who both suffer from considerable muscular spasticity due to spinal cord injuries, do not show strong indications of large-scale γ-band contamination from muscle activity, which confirms that high-quality EEG signals can be recorded from individuals who suffer from this condition. For Subject-13, the extremely high power in site P3 indicates a poor electrode connection. This poor connection appears to have also interfered with the quality of other recording sites, which explains the spikes in power at 16Hz, 28Hz, 44Hz and 76Hz that are

caused by aliasing of the harmonics of 60Hz interference from the surrounding power mains. Upon examination of time-domain plots over the course of the experiment for Subject-13, it appears that this electrode became dislodged part way through the recording session. This is likely due to contact between the EEG cap and the headrest on the participant's wheelchair. Care should be taken when using EEG caps with a headrest and an ideal EEG system for use in BCIs would be as resilient to this situation as much as possible. For the other subjects with relatively high levels of γ-band activity, e.g., subjects 1, 2, 3, 4 and 11, it is unclear from these PSDs alone whether or not these relatively high levels of γ-band activity are due to muscle activity or due to synchronized electrical activity in the brain. We will, however, continue to examine this question in later experiments.

Task-wise PSDs

Another approach for analyzing aggregate differences in our EEG signals is to average the PSDs across subjects in order to examine the mean differences in power content for each mental task. Although we have shown that there are considerable differences in EEG signals across individuals, this experiment examines patterns in the EEG signals for each mental task that are common across subjects. Since it is difficult to visualize the differences among PSDs for four mental tasks and eight channels, each of which have powers across a spectrum of frequencies, we also average these PSDs across all channels. Although the spatial distribution of our PSDs is certainly important, and will be investigated in the next section, PSDs averaged across channels are straightforward to visualize and can be viewed as an estimate of the overall power content of the signals.

In Figure 4.23, we show PSDs without Welch's method for each of our four mental tasks averaged across all segments and channels for all participants except for the low-performing subjects 6, 10, 13 and 14 that were identified in Section 4.2.2. As was noted for our previous visualization, many of these participants have either poor signal quality, high-amplitude artifacts or unusually low α power. Excluding these participants ensures that this does not adversely affect our averaged PSDs. The inset axes in Figure 4.23 highlight the differences across tasks in

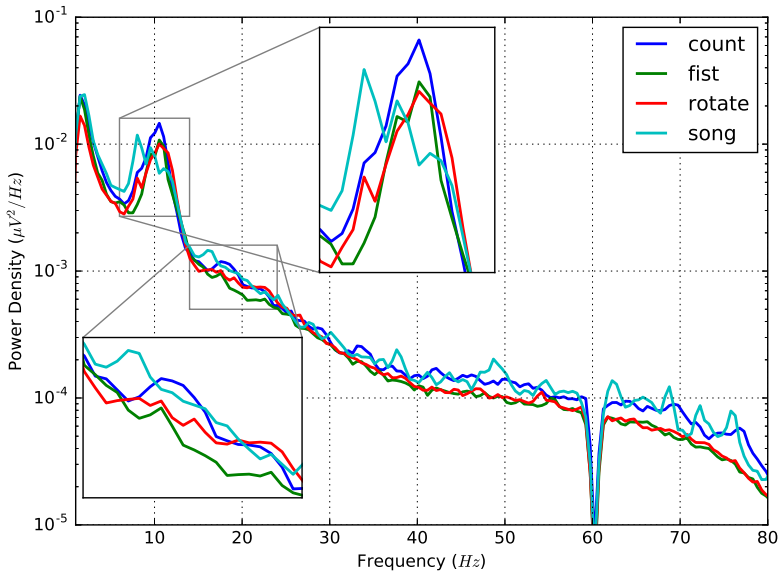


Figure 4.23: PSDs for each mental task averaged across channels and subjects, except the lowest-performing subjects, 6, 10, 13 and 14. The inset axes highlight the differences roughly in the α and low β bands. On average, power is higher in low-frequency α for the Song task and higher in high-frequency α for the Count task relative to the other tasks. Mean power tends to be higher for the Song and Count tasks in low-frequency β and lower in both low-frequency α and mid-frequency β for the Fist task. Both Count and Song have higher power in the γ band relative to the other tasks.

the 6–14Hz and 14–24Hz ranges, which roughly corresponds to the α (8–14Hz) and lower end of the β (14–30Hz) bands, respectively. Note that, on average, power is higher in low-frequency α for the Song task and higher in high-frequency α for the Count task relative to the other two tasks. We suspect that this increase in α power may be related to attention and the suppression of activity in regions of the brain that are not related to these tasks [137]. Also notice that the Count and Song tasks have a subtly higher mean power in the low range of the β band while the Fist task has a lower mean power in the low α and mid-range β bands relative to the other tasks. This decrease in power for the Fist task is consistent with the μ and β rhythm desynchronization that is often leveraged by motor-imagery BCIs [27, 29, 30]. There is also an increase in mean power across the γ (> 30Hz) band for both the Count and Song tasks relative to the Fist and Rotate tasks. Although it is difficult to establish whether or not these differences in γ power have a neuronal origin or are caused by muscle movement artifacts, as noted earlier, it is worth

noting that these differences in γ rhythms are relatively consistent for the Count and Song tasks across participants.

Despite the similarity between the γ-bands for the Count and Song tasks and also between the Fist and Rotate tasks, it is important to note that these similarities do not strictly align with the class confusions that we noted in Section 4.2.5. Although we did observe some confusion between the Count and Song tasks when using LDA, we did not see an exceptionally high rate of confusion between Count and Song for our TDNNs and CNN-TRs and we did not see a high rate of confusion between Fist and Rotate for any of our classification approaches. This means that, despite the apparent similarity between these PSDs, our classifiers do not often confuse these tasks for each other. This may suggest that our classifiers tend to utilize differences in the α and β bands rather than differences in the γ band or that they may be leveraging differences in the spatial distribution of these γ rhythms. In the following section, we will see that our LDA classifiers generally do not place a high level of importance on γ rhythms. We will also see in subsequent experiments, however, that our CNN-based classifiers tend to leverage high frequency information to a greater extent. This may also suggest that some of the differences in these γ rhythms may require nonlinear or multivariate phase information in order to be used effectively for discrimination.

Overall, this experiment demonstrates that there are some consistent average differences in the PSDs for each mental task and that these differences generally align with our prior knowledge about the types of patterns that we might expect to find in EEG signals produced during these tasks. This allows us to conclude that there are, in fact, some types of patterns that are common across individuals, despite the large differences that we have also seen in our subject-wise PSDs averaged across all tasks. This supports our claim that transfer learning may be a valuable tool for increasing the amount of training data available and for leveraging patterns that tend to be common across individuals.

4.3.2 LDA Weight Analysis

A valuable benefit of combining PSD-based signal representations with linear classifiers is the ease with which these methods can be interpreted. Since the weights of a linear classifier represent a relative measure of importance associated with a given input feature, interpretation is relatively straightforward. In Figure 4.24, we see a visualization of the inputs and weights for the LDA classifier trained for Subject-1. In the top of Figure 4.24 we see the input features averaged across all segments in the training partition for Subject-1 for each task laid out by channel. Recall from Section 3.1.1 that the input features for our baseline classifiers are the log_{10} transform of the Welch PSD which is then standardized to have zero mean and unit variance for each dimension. In the bottom of Figure 4.24, we see the magnitudes of the weights of our LDA classifier, except for the biases, aligned with each input feature along the top and the associated mental task along the left.

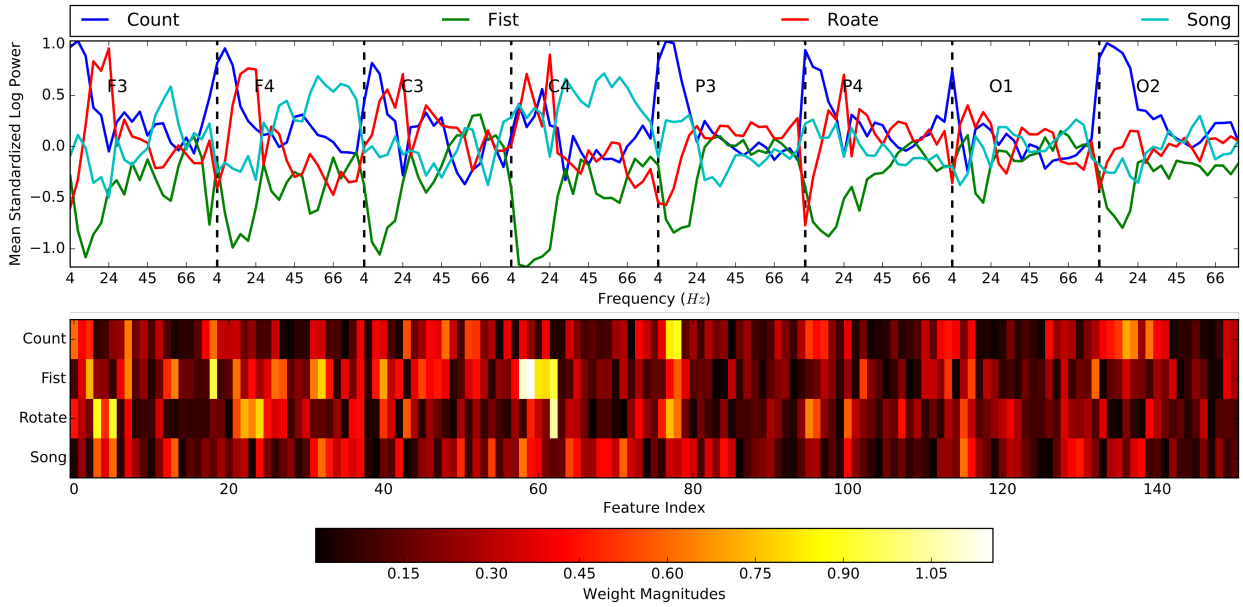


Figure 4.24: PSD features and trained LDA weights for Subject-1. (top) The log PSD with $\omega = 0.25$ averaged across all segments within each class. The features are concatenated into a single vector and passed to the subsequent classifier. (bottom) The weight magnitudes for a trained LDA classifier indicate the relative important for each feature and class.

This visualization allows us to make a number of observations about which input features are useful for identifying each mental task for Subject-1. First, notice that the Count task tends to have higher power than the other tasks in the α band for all channels except for C4 and O1 and in the low β band for P3 and O2. The classifier also learned higher weights for these bands for the Count task, especially in the parietal, frontal and right occipital regions. The classifier also has moderately large weights in the γ band for the Count task at sites C3, C4, P4 and O1 and O2; although the γ band powers for the Count task are not exceptionally higher than the other tasks for Subject-1. This aligns with our observation in the previous section that increases in power in the α, β and γ bands are associated with the Count task in our averaged PSDs.

Next, notice that there tends to be considerably less power in the α and β bands for the Fist task. Again, this is consistent with our observations about the averaged PSDs in the previous section. Although there are a number of weights associated with these frequency bands, the largest weights for the Fist task are located near the α and low β bands at site C4. This is consistent with the μ (8–14Hz) and β rhythm desynchronization over the motor cortex expected for a left-sided motor imagery task. Also note that some weights for the Fist task are moderately large in the γ band, especially in the frontal region and left central region, which may be related to executive control and planning of the motor movements.

For the Rotate task, we see a relatively high power in the α and low β bands in the frontal and central regions and low power in the α band over the parietal and occipital regions, which is consistent with a combination of increased attention along with desynchronization in the primary visual cortex in the occipital region. The largest weights for the Rotate task are associated with the high power in the α and β bands over the frontal and central regions, especially in channels F3, F4 and C4. To a lesser degree, there are also weights associated with the relatively low α power in P3 and P4. There are few weights associated with the γ band for the Rotate task, which is consistent with our averaged PSDs in the previous experiment, which do not show elevated γ power for the Rotate task on average.

The Song task generally has mid-range power in the α and β bands and has relatively high γ power in channels F4 and C4, which are over the frontal and central regions of the right hemisphere of the brain. While this relatively high level of γ power is consistent with our averaged PSDs, Subject-1 appears to have a less-than-average α power relative to the other tasks. Although the weight magnitudes for the Song task are somewhat diffuse across features, we do see moderately high weight magnitudes for the γ band in channels F4 and C4 and for α power in channel P3. Note that γ-band activity in the frontal region is has been linked with speech production and rehearsal, especially in Brocha's area [142]. Although the increase in γ-band activity observed for Subject-1 is over the right hemisphere and Brocha's area is usually in the left hemisphere for right-handed individuals, bilateral activity in the frontal regions has been observed during speech production [143]. Notice that the Count task, which also has a verbal component, shows elevated levels of γ-band power in our averaged PSDs in the previous section. These observations suggest that γ-band activity associated with speech production may be an important pattern for mental tasks that have a verbal component.

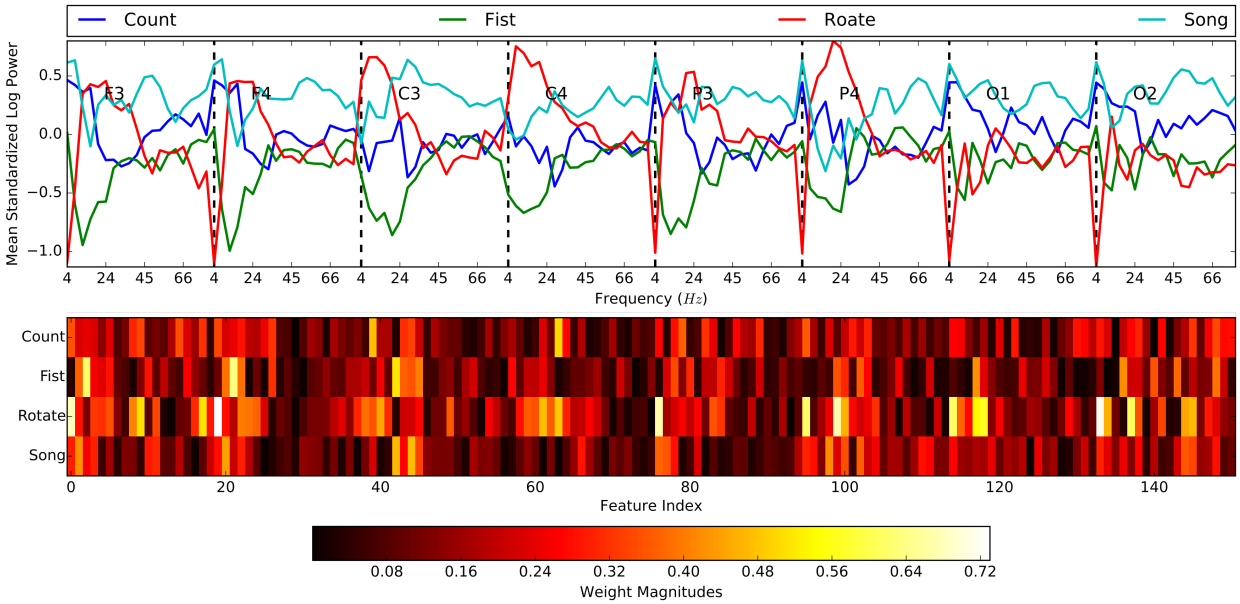


Figure 4.25: PSD features and trained LDA weights for Subject-5. (top) The log PSD with $\omega = 0.25$ averaged across all segments within each class. The features are concatenated into a single vector and passed to the subsequent classifier. (bottom) The weight magnitudes for a trained LDA classifier indicate the relative important for each feature and class.

In Figure 4.25, we see a similar visualization but now for Subject-5. Although Subject-5 is similar to Subject-1 in many ways, there are several notable differences. First of all, notice that the increase in α and β powers for the Count task is much less pronounced for Subject-5 and power in the γ band appears to be slightly more important instead, especially over the left frontal and occipital regions. For the Fist task, we again see a broad decrease in α and β power in the frontal and central regions with correspondingly high weights in the classifier, which is quite similar to Subject-1. For the Rotate task, Subject-5 has a much more pronounced increase in α and low β power over the central and parietal regions; although we now see important weights in the α band of nearly all channels, including over the occipital region, which again shows a decrease in α power. For the Song task, we again see high levels of γ power, but now across all channels. The most important weights for the Song task, however, appear to be related to α power in the frontal and left centroparietal regions and a broad range of frequency bands over the right-parietal and occipital regions.

Although the weights for the Song task that are associated with α power over the frontal lobe may again be related to speech production, it is unclear why there is elevated γ power and associated weights over the parietal and occipital regions. Recall from the previous section that Subject-5 has a notable amount of γ-band power that appears to be caused by muscle movement. We cannot rule out the possibility that Subject-5 may be inadvertently moving or tightening their muscles during the Song task. We do not, however, see excessively high weights associated with the γ band for the Song task, which suggests that even if muscle artifacts are more common for this subject during the Song task, such muscle movement does not appear to be the most important feature for classification.

Our analyses of aggregate PSDs in the previous section combined with our analyses of the individual features and classification weights for Subject-1 and Subject-5 in this section lead us to several conclusions. First of all, there does appear to be a general trend in the form of the PSDs for each mental task. For the Count and Song tasks we see increases α and γ activity over the frontal region that might be related to speech production. For the Count and Rotate tasks

we see increases in α power in the frontal region which may be related to attention and for the Rotate task we see decreases in α power in the occipital region that may be related to visual processing. For the Fist task we see decreases in μ and β rhythms over the central and parietal regions that are likely due to desynchronization in the motor cortex. There are, however, considerable individual differences in the precise frequencies and locations of these changes in the PSDs. This provides supporting evidence for the use of both transfer learning as well as individualized classifiers. Although we are unable to verify or reject the possibility that muscle movement may also correlate with some tasks for some subjects, we see little evidence that high-frequency muscle artifacts are a primary pattern utilized by our baseline classifiers and we have also identified that some subjects with contamination from muscle movement can still achieve relatively high classification accuracy.

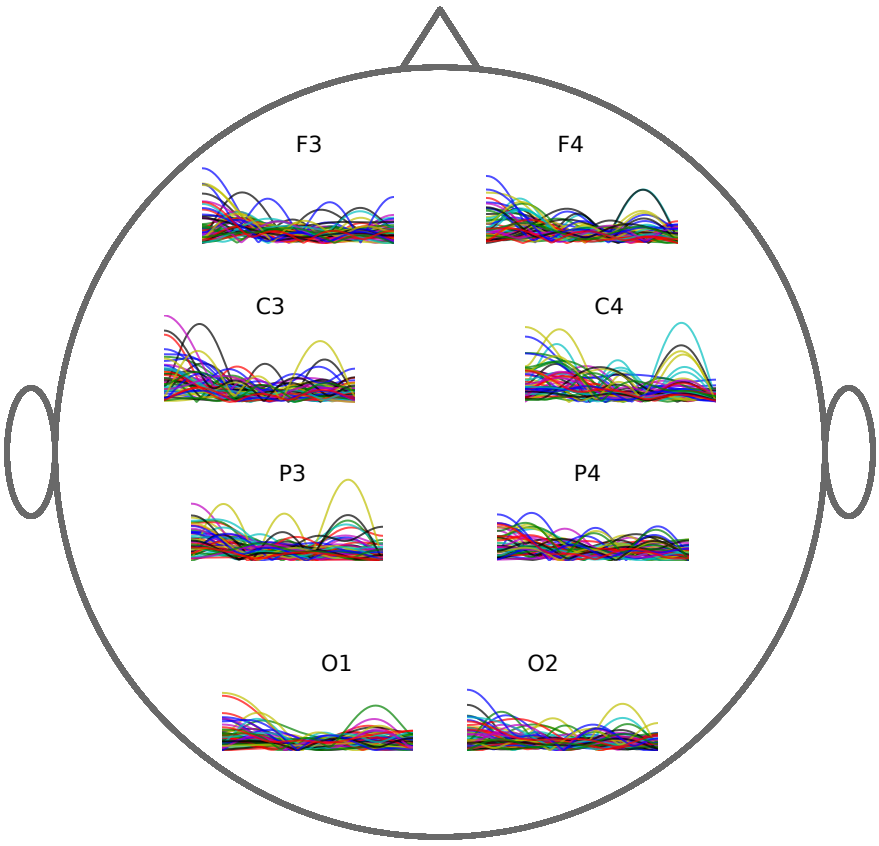
4.3.3 Learned Filters

Interpreting the patterns that are leveraged by our convolutional networks is less straightforward than analyzing our linear baseline classifiers for several reasons. First of all, there are many weights in each artificial neuron that span the width of the convolutional kernel, as opposed to a single weight associated with each channel and frequency bin in a PSD. Second, there is a nonlinear transfer function in each artificial neuron. Although a larger weight in a nonlinear model generally does indicate a greater level of importance for the associated input feature, larger weights also indicate a more sharply nonlinear response to the input. Stacked convolutional layers are also challenging to interpret because the behavior of our networks becomes increasingly nonlinear as the signals pass through multiple layers. The spatial layout of the network connections also becomes arbitrary after the first layer. Although we have a spatial mapping from the input channels to the surface of the scalp at the first convolutional layer, there is no enforced layout of the signals at the subsequent layers, which correspond only to the artificial neurons in the previous layer. Finally, pooling introduces a change in scale that must be considered during analysis.

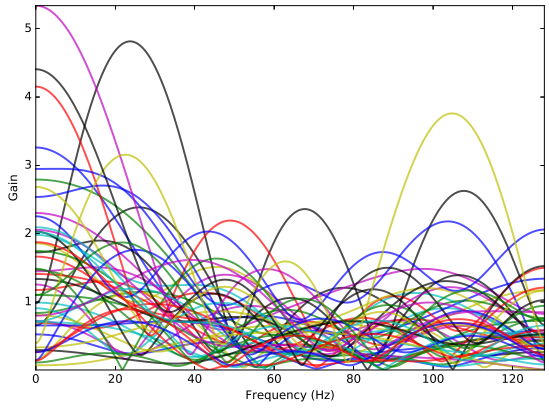
In order to overcome these challenges, we employ two tactics when analyzing the weights of our convolutional networks. First, in order to avoid the complications associated with the loss of a spatial mapping and increasingly nonlinear responses in multilayer networks, we will examine only the weights in our single-layer TDNNs. We will, however, utilize other approaches for analyzing our multilayer networks in later sections. Second, rather than attempting to analyze the weight magnitudes directly, we will utilize our interpretation of convolutional units as FIR filters and extract the frequency responses of these learned filters. Although it is certainly possible to visualize the weights of the convolutional kernels directly and interpret these weights as matched filters, i.e., attempting to match the shape of the weights to sections of the time-domain EEG signals, we have found that this approach generally does not yield valuable insights into the behavior of these networks.

Recall from Section 3.2.3 that the convolutional layers in our CNNs and TDNNs can be interpreted as learned, nonlinear FIR filters. Since a convolution in the time domain is equivalent to a pointwise multiplication in the frequency domain, the frequency response of the filter can be computed, before the application of our nonlinear transfer function, by finding the magnitudes of the discrete Fourier transform of the convolutional kernel. Note that this is a common practice in the field of signal processing for designing and analyzing linear FIR filters [110]. To our knowledge, however, we are the first to utilize this technique to analyze the weights of convolutional networks.

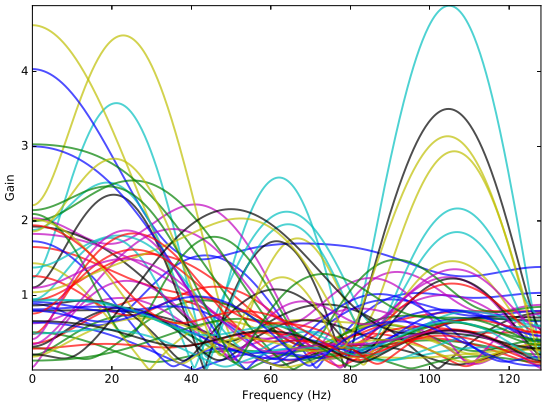
In Figure 4.26a we see the result of computing the magnitudes of the Fourier transforms of each convolutional kernel in each artificial neuron laid out over the surface of the scalp for a TDNN trained for Subject-1. Each line in this plot corresponds to a different artificial neuron in our network. Note that each artificial neuron spans all channels and computes a sum of the convolutions over all channels. Each subplot then corresponds to the frequency responses of the individual convolutions computed at each EEG recording site. The horizontal axis in each subplot corresponds to the frequency spectrum ranging from 0–128Hz, i.e., DC to the Nyquist rate, and the vertical axis corresponds to the gain of the FIR filter before the application of our



(a) Frequency responses at all recording sites.



(b) Frequency responses at site C3.



(c) Frequency responses at site C4.

Figure 4.26: Frequency responses, before the application of the nonlinear transfer function, for the convolutional kernels in all artificial neurons for a TDNN trained for Subject 1. Each line corresponds to the frequency response of a single convolutional kernel. Note that a given artificial neuron sums the convolutions across all channels. The horizontal axes correspond to the frequency spectrum while the vertical axes shows the gain of the filter at each frequency. (a) The frequency response for each convolutional kernel laid out by recording site. (b) The frequency response of each convolutional kernel associated with channel C3. (c) The frequency response of each convolutional kernel associated with channel C4. Note that many of these filters have amplify frequencies in the α, β and γ bands.

nonlinear transfer function. For example, if an FIR filter has a gain of two at a given frequency, then the signal will be amplified two-fold and if the filter has a gain of 0.5 then the amplitude of the signal will be attenuated by one half, before the application of our transfer function. In Figure 4.26b and Figure 4.26c, we see larger plots of the frequency responses at sites C3 and C4 for reference. Note that the scale of the axes is the same in all of the subplots in Figure 4.26a.

One clear limitation of this approach for analyzing the weights of our TDNNs is that we can only see the filtering characteristics of the network and not how it responds to each mental task separately. Nevertheless, we can make several observations about our networks from Figure 4.26. First of all, notice that the majority of the convolutional kernels have a high gain associated with the α, β and low γ bands. This suggests that frequencies in these ranges are highly utilized by this TDNN, which aligns with our observations about our PSD-based classifiers. Note, however, that the gain for some of these filters is as high as five. This indicates that the network is amplifying these frequencies considerably. Recall that our input signals are standardized to have zero mean and unit variance and that α frequencies are often near the highest amplitude signal components. Since our transfer function saturates at slightly above one, this suggests that some of our artificial neurons respond to α and β frequencies in a highly nonlinear way in some channels. Note, however, that we do not see the bias weights of our networks in this visualization, which can also affect nonlinear behavior by pushing the entire signal toward one a nonlinear region of our transfer function. This large amplification of α and β rhythms supports our claim that nonlinear information may be useful for classifying EEG signals.

Also notice that some convolutional kernels have a high gain for frequencies in the γ band. In particular, notice that sites F4 and C4 have several kernels with high gain in the γ band. This aligns with our previous observations in Figure 4.24 where we noted that the Song task had high amplitude γ power in sites F4 and C4 for Subject-1, which may be related to speech production. Some kernels in sites C3, P3, O1 and O2 also have relatively high gain in the γ bands which does not closely align with our observations about the LDA classifier for Subject-1. This may indicate that our TDNNs are able to extract additional information from the γ band that is not

easily identified in the PSDs, e.g., subtle short-term or nonstationary patterns in the signals. It is valuable to note, however, that signal components in the γ band typically have amplitudes that are several times smaller than α rhythms, which suggests that these signal components likely still have lower amplitudes than in the α after passing through these filters.

Note that plots similar to Figure 4.26 for other subjects generally show similar trends. This suggests that the convolutional kernels in our TDNNs tend to strongly amplify frequencies in the α and β bands and, to a lesser degree, in the γ band. Although this roughly aligns with our observations about the important frequencies leveraged by our PSD-based LDA classifier, it appears that our TDNNs may be able to leverage patterns in the γ band in some channels that LDA is not able to leverage. The importance of γ-band patterns will be further examined in later experiments.

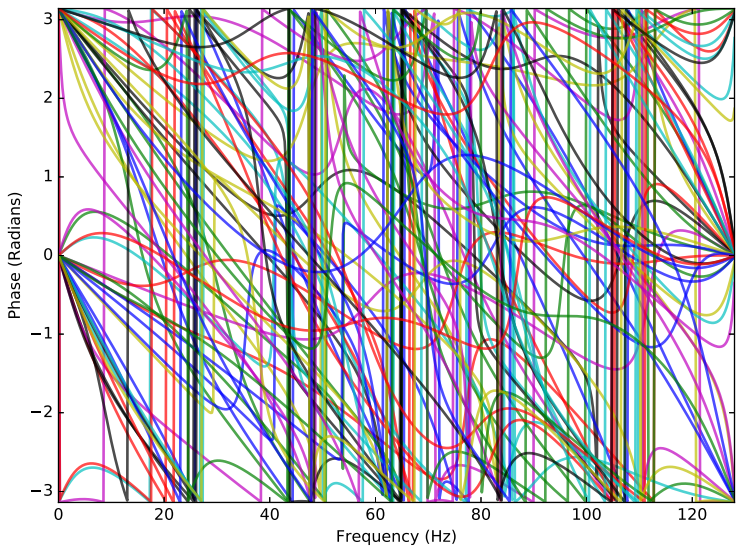


Figure 4.27: The phase responses for the convolutional kernels in a TDNN trained for Subject 1 for channel C4. Note that the phase responses are highly nonlinear across the frequency spectrum, which indicates that our convolutional kernels are not learning optimal bandpass filters. This may be due to the filters learning to incorporate phase information in addition to amplitude information and also because the filters learn to pass amplitude information across multiple frequency ranges.

In addition to examining the gain of our FIR filters, it is also possible to examine the phase responses of these filters by extracting the complex arguments of the Fourier transforms of the

convolutional kernels. In Figure 4.27, we see the phase responses of all convolutional kernels at site C4 for the same network in the previous experiment for Subject-1. Note that the horizontal axis corresponds to the frequency of the input signal, from DC to the Nyquist rate, while the vertical axis indicates the amount that the phase will be shifted in the output signal in units of radians. Although this figure is difficult to parse visually, it does lead us to an important observation: the phase responses of our convolutional units are highly nonlinear across the frequency spectrum. Note that an optimal FIR bandpass filter, in the sense that it passes only a specific band of frequencies with the steepest possible rolloff and minimal ripple, has a linear phase response where the filter shifts the phase of the signal linearly as the frequency of its input increases.[20] The fact that our learned filters have very nonlinear phase responses indicates that they are not approximating optimal bandpass filters. This may be for several reasons. First of all, it is likely that our convolutional units are learning to utilize phase information, in addition to amplitude information, by manipulating the phase offsets of the signals as they pass through each artificial neuron. As we noted in Section 2.1.1, a major limitation of PSDs is their inability to incorporate multivariate phase information, despite the fact that phase synchronization is known to be an important aspect of EEG signals. Second, it is possible that our convolutional units are learning to filter multiple frequency ranges simultaneously, rather than simply selecting for a narrow passband. This demonstrates how these filters extract information that is useful, rather than attempting to adhere to assumptions about specific frequency bands. The sophisticated nature of the phase responses of our convolutional kernels demonstrates their versatility while also highlighting the difficulties associated with interpreting the types of patterns that they learn to leverage.

4.3.4 Layerwise Outputs

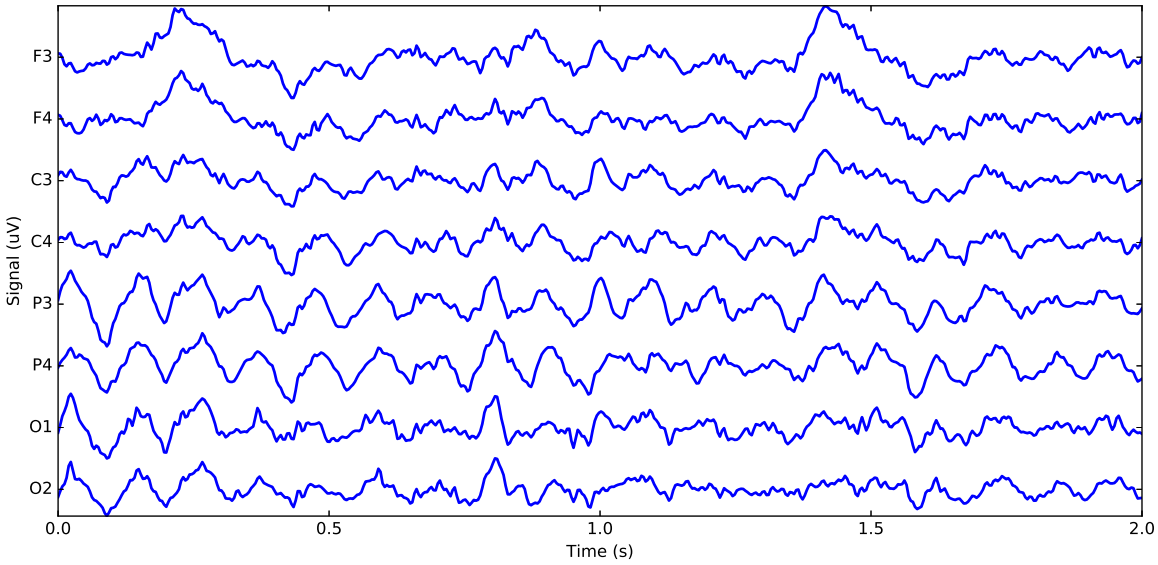
As we have discussed in the previous section, it is difficult to analyze how the weights of a convolutional layer interact with different classes and it is also difficult to analyze the weights of

[20] Optimal linear FIR bandpass filters with finite order do have nonlinear jump discontinuities associated with ripple but are linear everywhere else.

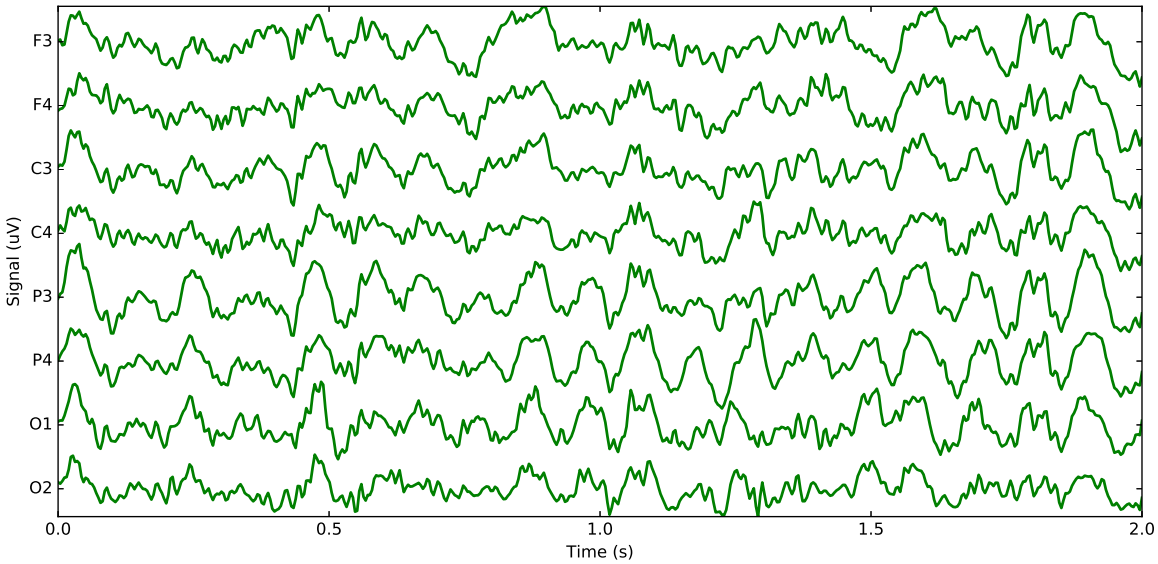
multilayer CNNs. In order to address these challenges, we will examine the outputs, also called activations, produced by each layer rather examining the weights directly. This allows us to observe how the network responds to inputs belonging to a given class, rather than attempting to make inferences about the network's behavior by examining the connection weights between its artificial neurons.

In order to ensure that we are using input segments that are representative of each class, we select the best-performing EEG segment for each class from the test partition. In other words, we run a forward pass of the network for each segment in the training partition and identify a single segment for each class that produced the highest sum of log likelihoods, as described in Section 3.2.1, for the correct class label. Note that we utilize segments from the test partition in order to ensure that we are examining the generalization behavior of the network rather than examining segments that the network was exposed to during training. Time-domain trace plots of the best-performing EEG segment for each class for Subject-1 are shown in Figure 4.28. Note that these plots are color coded with a unique color for each class for reference in future plots. Aside from a few ordinary ocular artifacts, notably around 0.25s and 1.5s centered in channels F3 and F4 for the Count task in Figure 4.28a, these EEG segments appear to have clean signals with few artifacts. These EEG segments also have clearly visible α and β and γ rhythms.

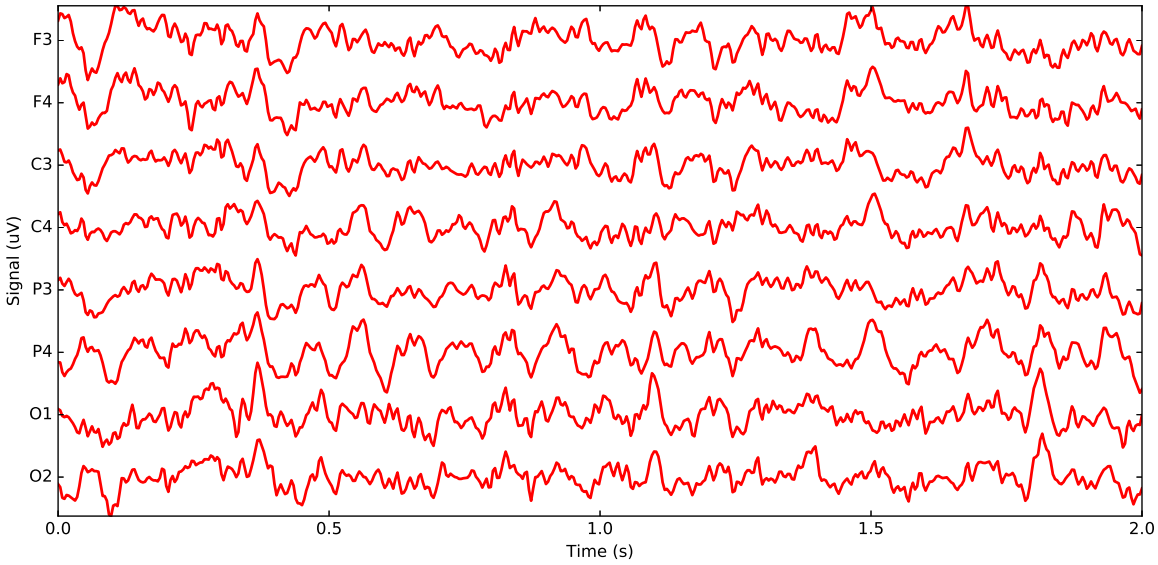
Recall that our full CNN-TR architecture, described in Table 4.1, consists of four convolutional layers with 16 hidden units in the first two layers and 20 hidden units in the next two layers. Since it is difficult to visualize the activations produced by this many hidden units, we have trained a smaller network with the same configuration except that it has only six hidden units in the first two convolutional layers and eight hidden units in the following two convolutional layers. This network is trained using our full transfer learning and early stopping procedures for Subject-1. Note that the test CA for this smaller network is 65% in this case, which is 20% lower than the test CA for our larger network configuration. Despite this difference in performance, this network does perform significantly better than a random classifier and we



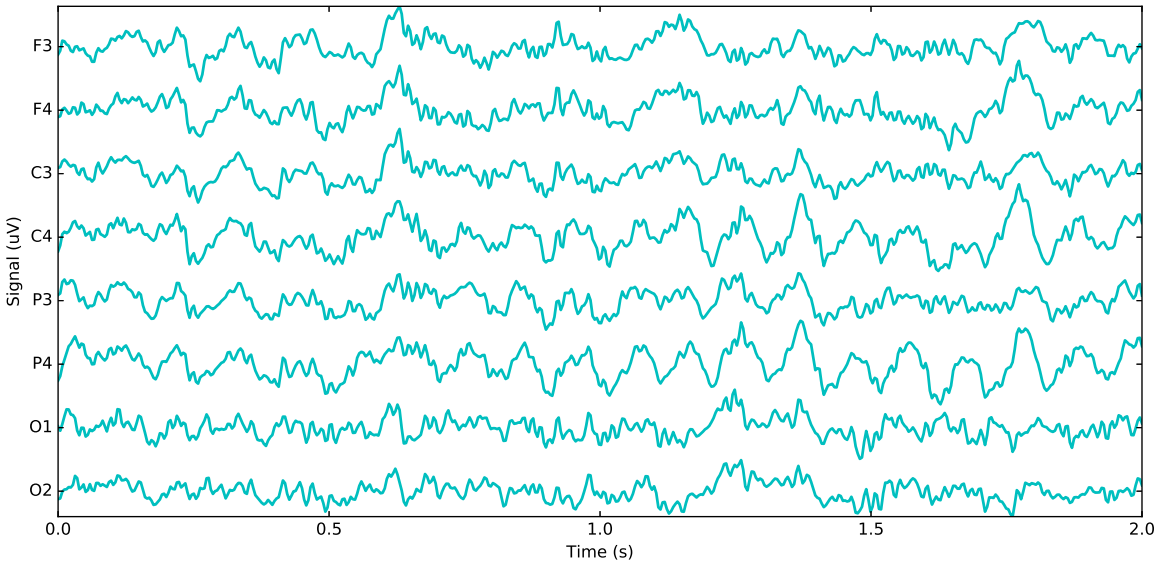
(a) Best performing segment for Count.



(b) Best performing segment for Fist.



(c) Best performing segment for Rotate.



(d) Best performing segment for Song.

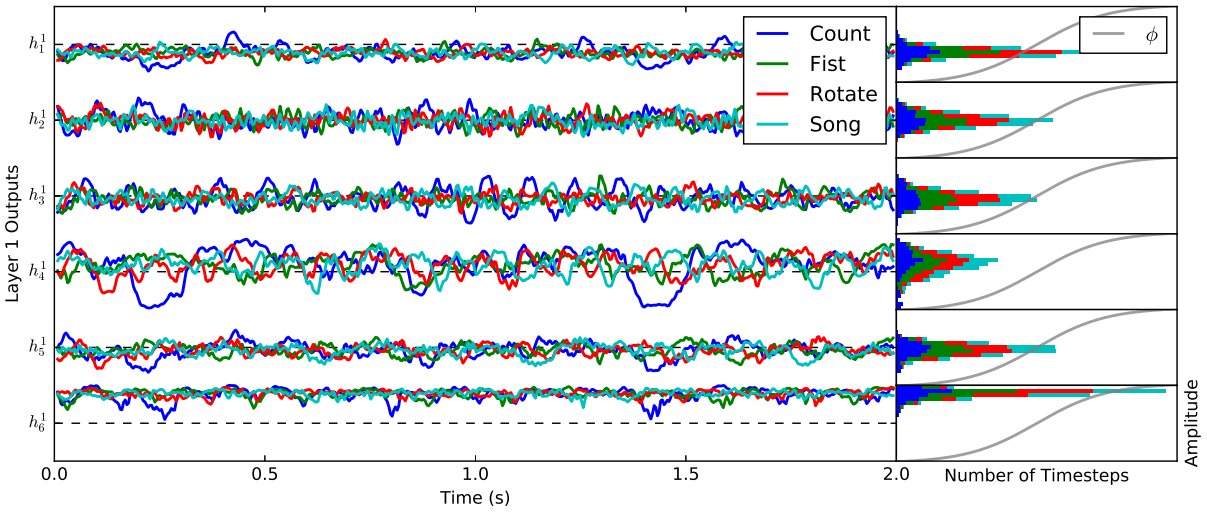
Figure 4.28: The best-performing EEG segments, i.e., highest sum of likelihood, for each class for our small CNN-TR for Subject-1. Each segment has a different color for reference in future plots. Note the ocular artifacts in (a) around 0.25s and 1.5s in F3 and F4. The other segments are largely free of artifacts and contain visible α, β and γ rhythms.

have confirmed that the activations at each layer of this network appear to behave similarly to the larger configuration.

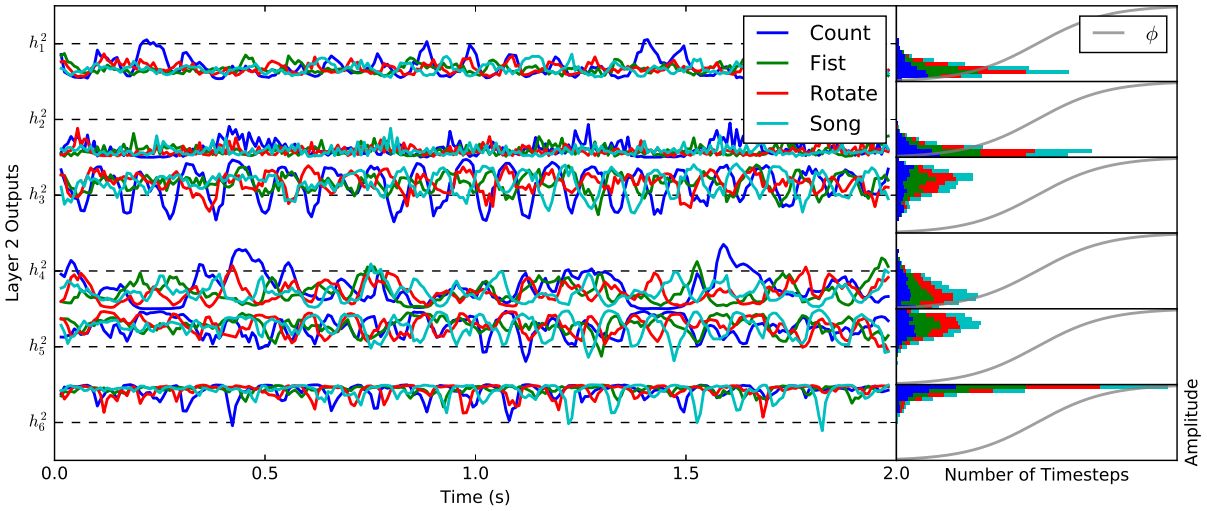
In Figure 4.29, we see the activations produced by each hidden unit for the best-performing EEG segment in each class at each of the four convolutional layers in our smaller CNN-TR network. The activations produced by each EEG segment are plotted on top of each other for comparison and are color coded using the same colors from Figure 4.28. On the left side of Figure 4.29, we see trace plots of the activations where the horizontal axis corresponds to time and the vertical axis corresponds to the network activations spread out across the hidden units. Each hidden unit is labeled h_i^ℓ where i denotes the index of the hidden unit and ℓ denotes the layer. On the right side of Figure 4.28, we see a histogram for each hidden unit that shows the distribution of the activations stacked by class. The grey line superimposed over the histograms shows the transfer function with appropriate scale so that we can see the saturation and degree of nonlinear behavior of the activations.

From the histograms in Figure 4.29a, we can see that most of our activations occur around the center of our transfer function, with the exception of h_6^1. This indicates that our first layer primarily acts a bank of linear filters that is applied to the input signals. Note that the ocular artifacts we observed in the segment for the Count Task near 0.25s and 1.5s are largely isolated by units h_4^1 and to a lesser degree h_1^1, h_5^1 and h_6^1. This may suggest that part of the role learned by these units is to filter artifacts caused by eye movement. α rhythms also appear to be most prominent, especially for the Count task, in h_3^1 and h_4^1 and slower drifts appear to be largely isolated to h_4^1 and h_5^1 while the other channels appear to contain mostly higher frequency activations. Many of the changes in our input signals from Figure 4.29 can also be identified in the activations of these units. This again suggests that the units in the first layer are primarily acting as filters that isolate different components of the signals, likely in ways that are both spatial and temporal.

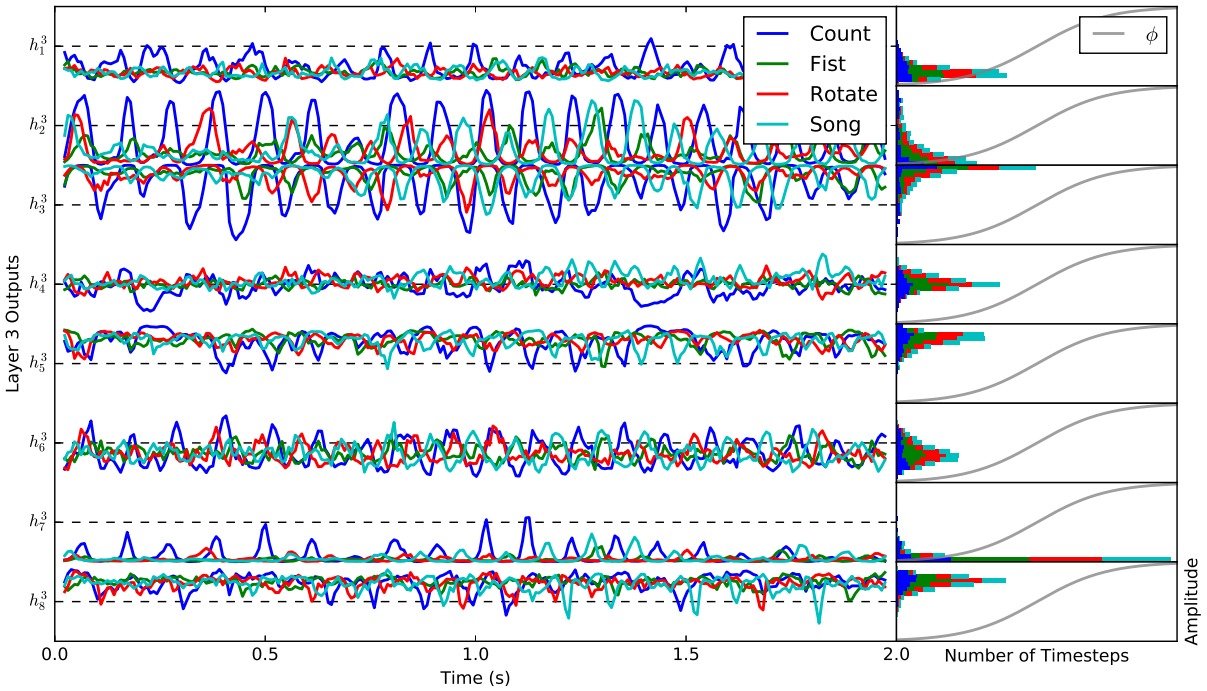
From the trace plots in Figure 4.29b, we can see that the second layer continues to filter the activations from the previous layer. It appears that most of the units now generate activations



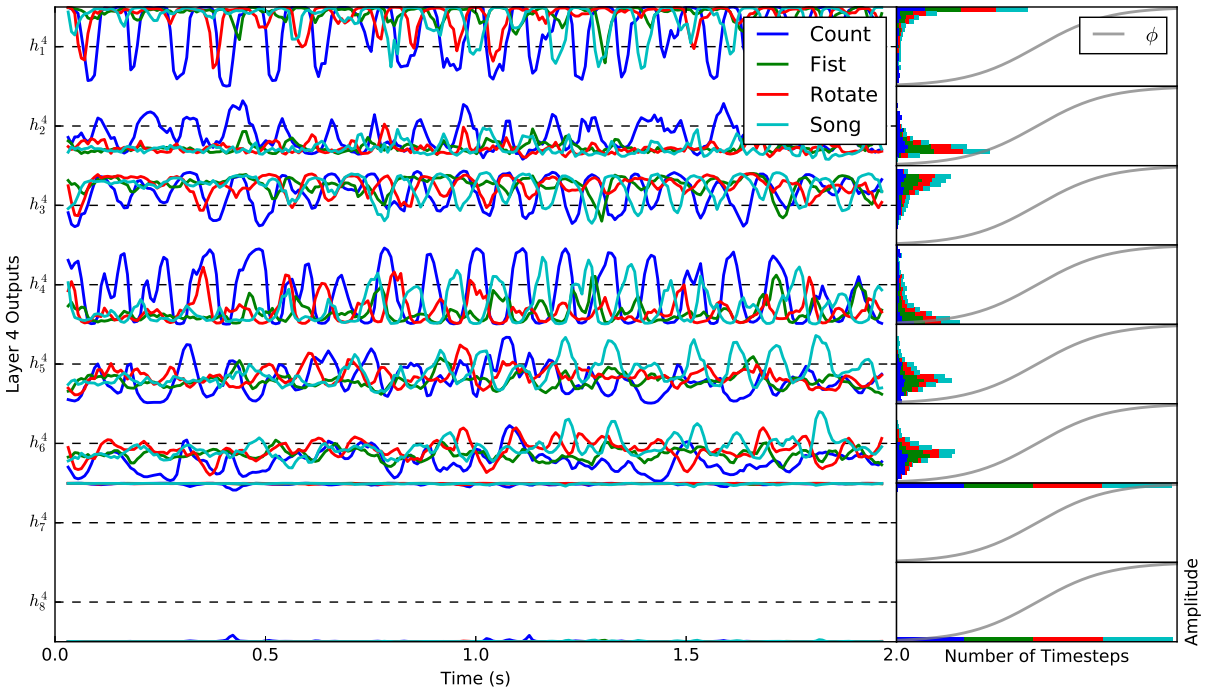
(a) Layer 1 activations.



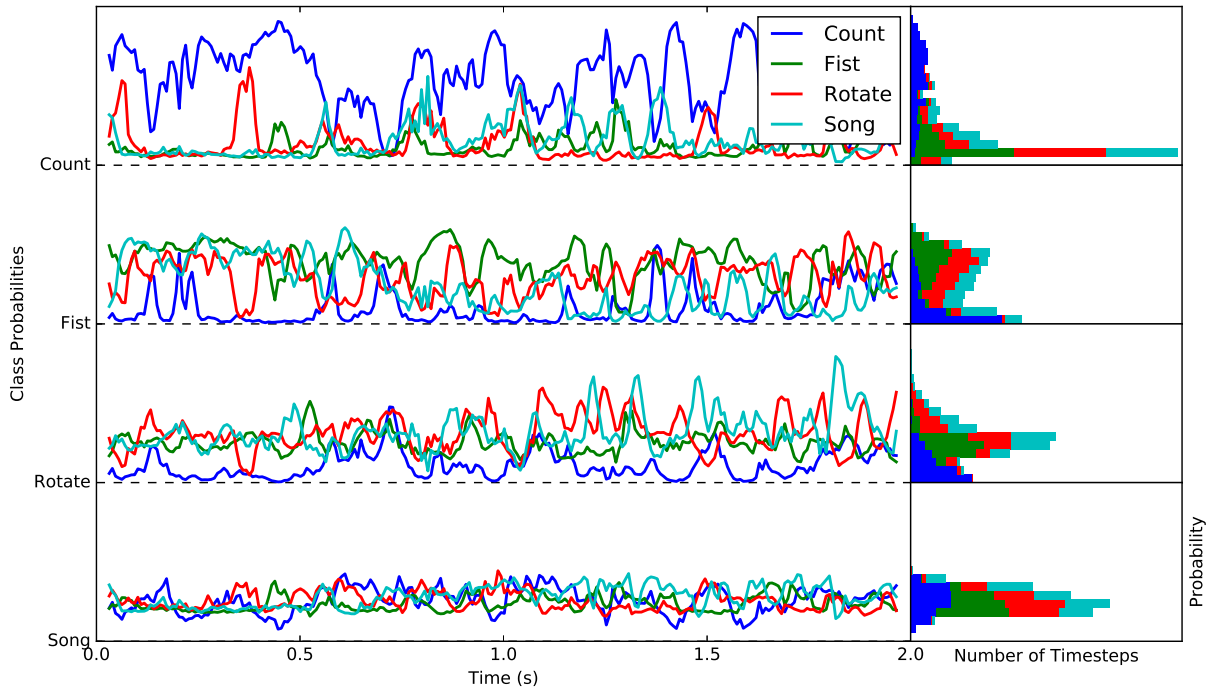
(b) Layer 2 activations.



(c) Layer 3 activations.



(d) Layer 4 activations.



(e) Layer 5 estimated class membership probabilities at each timestep.

Figure 4.29: Activations produced at each layer by our small CNN-TR for Subject-1. The trace plots on the left side show the time-domain activations produced by each hidden unit described on the left vertical axis over the course of time along the horizontal axis. These activations are color coded according to classes of the signals, which correspond to the best-performing segments in the previous figure. The histograms on the right show the distribution of the activations for each neuron and signal segment across the transfer function, shown in grey. (a) The activations produced by the first convolutional layer largely fall on the linear region of the transfer function, suggesting that this layer primarily acts as a bank of linear filters. (b) The activations produced by the second layer have increasingly nonlinear behavior and some isolation of specific frequencies and of ocular artifacts is apparent. (c) The activations produced by the third layer, which is preceded by pooling, are increasingly nonlinear and are generally slower moving. (d) The activations produced by the fourth layer are highly nonlinear and often have a one-sided threshold type response. A higher degree of specialization is now visible with some units responding more strongly to patterns in specific classes and not others. (e) The predicted class membership probabilities at each timestep produced by the linear readout layer. The Count task is clearly predicted well; although it is also apparent from the histograms that the other tasks are often predicted correctly as well.

that are primarily in the α and low β bands with the exception of h_2^2, which largely generates higher frequency activations in the γ band and both h_1^2 and h_3^2, which contain some frequencies across all three of these bands. The ocular artifact in the EEG segment for the Count task continues to be present and is now isolated primarily in h_1^2 and h_4^2 and, to a lesser degree, h_2^2.

From the histograms in Figure 4.29b, we can also see that the activations in the second layer now fall in the nonlinear region of our transfer function much more often for all units. This indicates that the second layer behaves in a more nonlinear way than the first. Also note that the activations tend to lie closer to one side or the other of our sigmoidal transfer function. This indicates that the network has learned to use the bias values to push the activations toward the nonlinear region of the transfer function. This allows the network to transform oscillating signals into a more rectified, binary response where there tends to be a pulse in one direction if a certain waveform is present while remaining largely inactivated if the desired waveform is not present. Overall, it appears that the second layer of our network is continuing to filter the signals while also beginning to form nonlinear filters that become activated in response to more specific patterns in the signals.

In the third layer, pooling has been applied and so the scale of our activations is now $\frac{1}{2}$ that of the original input signals. From the histograms in Figure 4.29c, we can see that the activations in the third layer have become increasingly nonlinear, with the exception of h_4^3 and perhaps h_6^3. From the trace plots we can see that our activations contain increasingly different responses for each class. Many of the activations also continue to have one-sided responses. For instance, notice that h_7^3 generates pulses roughly in the β range but almost exclusively for the Count task and, to a lesser degree, the Song tasks. These class-specific activations appear to be strongest for the Count task, which is the most accurately classified task for Subject-1. These observations suggest that the third layer has learned to produce increasingly nonlinear and task-specific responses.

In Figure 4.29d, we can see that the activations produced by the fourth layer are predominantly in the α and β bands. This suggests that higher frequency information has either been

largely attenuated, or that the previous layers have already consumed the higher frequency information in order to generate the activations produced up to this point. Since we have already established the importance of high frequency for some tasks in our previous experiments, we believe it is likely that high frequency information has already been utilized and has essentially been smoothed into the activations by the fourth layer. Given that we have changed the scale of our networks before the third layer and since the network is optimized to output the class label across the entire output sequence, it makes intuitive sense that the network would utilize higher frequency information in earlier layers and transfer this information into slower moving activations. We also note that there continues to be one-sided nonlinear activations, which suggests that the network has learned to essentially place thresholds on these activations. There also appears to be increased specialization of the activations, with some units responding more strongly to some classes than others. Although this specialization is somewhat subtle and can be difficult to see from the trace plots, a close examination of the histograms shows that there is now more separation in the distributions of the activations in some hidden units. For instance, many of the activations for the Fist and Rotate tasks now fall on one side of h_1^4 and h_4^4 while the activations for Count are now spread out and more often fall on near the other side of the transfer function. Also notice that h_7^4 and h_8^4 have very small amplitude but extremely specialized responses to patterns in the signals that primarily occur only for the Count task.

In Figure 4.29e, we see the output of the readout layer, which is the final layer in the network. The role of the readout layer is to combine the outputs of the fourth convolutional layer in a linear way in order to produce class membership likelihoods for each class at each predicted timestep of the network. Note that we have converted the log likelihoods produced by our network at each timestep into proper class membership probabilities. From the trace plots in this figure, we can see that the network did quite well at identifying the Count task, moderately well at identifying the Fist and Rotate tasks and somewhat poorly at identifying the Song task. Upon examination of the histograms, we can see, however, that there does tend to be a degree of separation between the predicted probabilities and that the correct class is often

predicted with a higher probability than the incorrect classes. This supports our claim that by the fourth layer of our network the activations have a high degree of task and pattern specific specialization that allows our linear readout layer to combine these activations in a way that produces appropriate class predictions. In Appendix B, we show the activations produced by a full-sized CNN-TR network at each layer and the best-performing EEG segment for each task for Subject-1. Although these larger visualizations are somewhat more difficult to parse due to the size of these networks, a careful examination does demonstrate a similar pattern of filtering and increasing specialization.

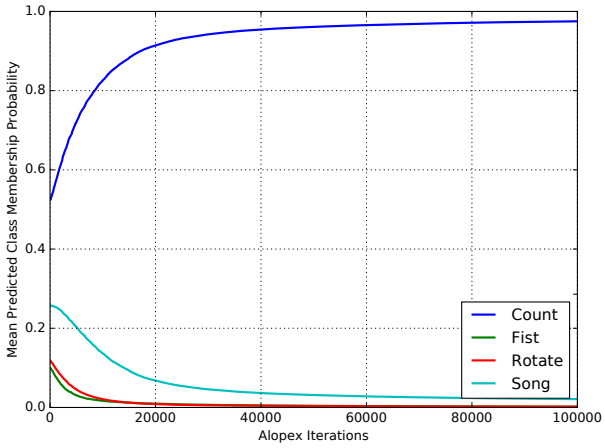
This analysis of the layer-wise activations produced by a sample CNN-TR allow us to draw several conclusions. First of all, we have shown that our networks tend to leverage increasingly nonlinear patterns at each successive layer. While the first layer acts largely as a bank of linear filters, deeper layers begin to behave in very nonlinear ways and often appear to have a one-sided threshold type response, leveraging the unit's bias, in order to produce activations only in response to very specific types of patterns. Second, we have noted the higher frequency information tends to largely be consumed by the earlier layers and is incorporated into slower moving activations in later layers, after pooling has forced a change in scale. Since the memory capacity and impulse response of our networks increases at each successive layer and since the network seeks to assign correct class labels for as many timesteps as possible following a pattern that is indicative of a given class, it makes sense that the network would attempt to transform fast patterns into slower moving responses. Along with the increasingly nonlinear and lower frequency responses produced at each layer, we have also observed increasing specialization. In other words, each artificial neurons in deeper layers primarily responds to patterns that are often associated with some classes and not others. These observations support our claims from Section 1.3, and throughout the previous chapters, that our proposed CNNs are able to learn nonlinear and hierarchical representations with increasing specialization and changes in scale that can ultimately by used to produce appropriate class membership predictions.

4.3.5 Optimized Input Sequences

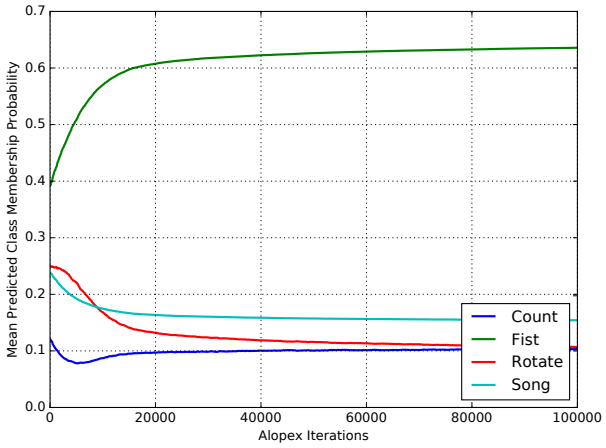
In addition to examining the behavior of our networks when processing actual EEG segments, we have also described a method in Section 3.3.4 for finding optimized, artificial input sequences that produce high class membership probabilities. These optimized input sequences will help us to further understand the types of patterns that our networks are leveraging by allowing us to examine inputs that contain patterns that the network finds highly valuable for discriminating between classes. Examining the behavior the network's activations when processing these sequences will also allow us to further understand how our networks are processing these patterns and forming useful representations of EEG signals.

In order to generate these optimized input sequences, we first seed our optimization routine with the same EEG segments that we examined in the previous section. This initializes our optimization procedure with a real EEG segment that produces a correct label for the corresponding class. We then use the ALOPEX optimization procedure, as described in Section 3.3.4, to modify these segments in a way that increases the predicted class membership probabilities, beyond that of the original input segments, while the weights in our network remain fixed.

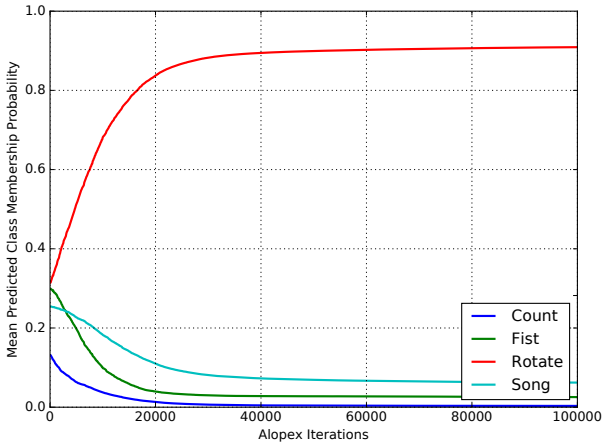
In Figure 4.30, we see how the mean predicted class membership probabilities, averaged across timesteps, change for the segment in each class as the iterations progress for our ALOPEX optimization procedure. Notice that for the Count and Fist segments, the mean predicted class membership probability rapidly rises above 0.9 after about 30,000 iterations while the mean probabilities for the other tasks drop below 0.1. Similarly, but to a lesser degree, the mean probability for the Fist task rises to about 0.65 while the predicted probabilities for the other classes falls to 0.1–0.15. Recall that we have found that the Fist task often performs relatively poorly, which likely corresponds to these lower probabilities for the optimized input sequence. For the Song segment, the mean probability predicted for the Song task gradually rises to 0.5 at about 100,000 iterations while the mean probability falls to about 0.05 for Fist and 0.25 for Count and Rotate. Although applying additional iterations of ALOPEX does continue to improve the mean predicted probability for the Song task, we have found that this amount of training is suffi-



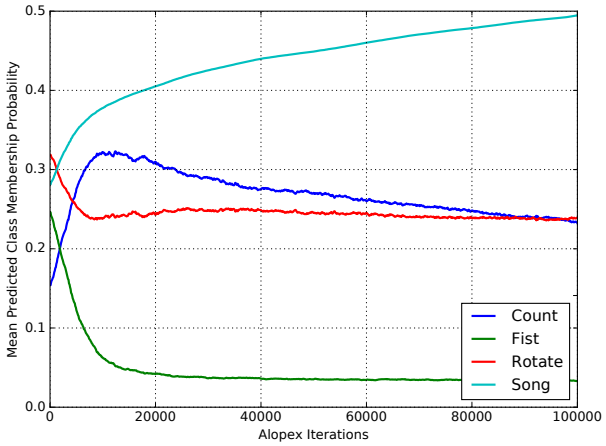
(a) Probabilities vs. iterations for Count.



(b) Probabilities vs. iterations for Fist.



(c) Probabilities vs. iterations for Rotate.



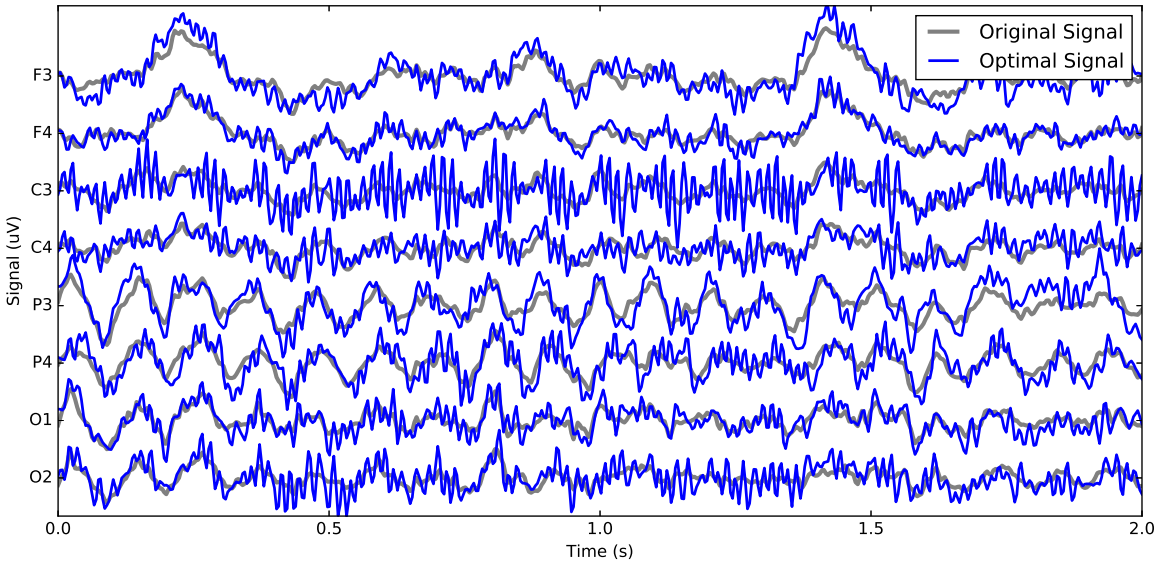
(d) Probabilities vs. iterations for Song.

Figure 4.30: Predicted class membership probabilities, averaged across timesteps, versus ALOPEX optimization iterations. (a) For the Count segment, the mean probability for Count approaches 0.98 after 30,000 iterations while the other tasks near zero. (b) For the Fist segment, the mean probability for Fist levels off at around 0.65 while the other tasks level off between 0.1–0.15. (c) For the Rotate segment, the mean probability for Rotate exceeds 0.9 after 40,000 iterations while the other tasks fall below 0.1. (d) For the Count task, the mean probability for Count approaches 0.5 and continues to approve beyond 100,000 iterations while Fist approaches about 0.05 and Count and Rotate fall to about 0.25. Although further optimization does improve the predictions for the Count task, we have found that this level of optimization allows us to draw conclusions about inputs that yield improved performance without departing too dramatically from the original input segments.

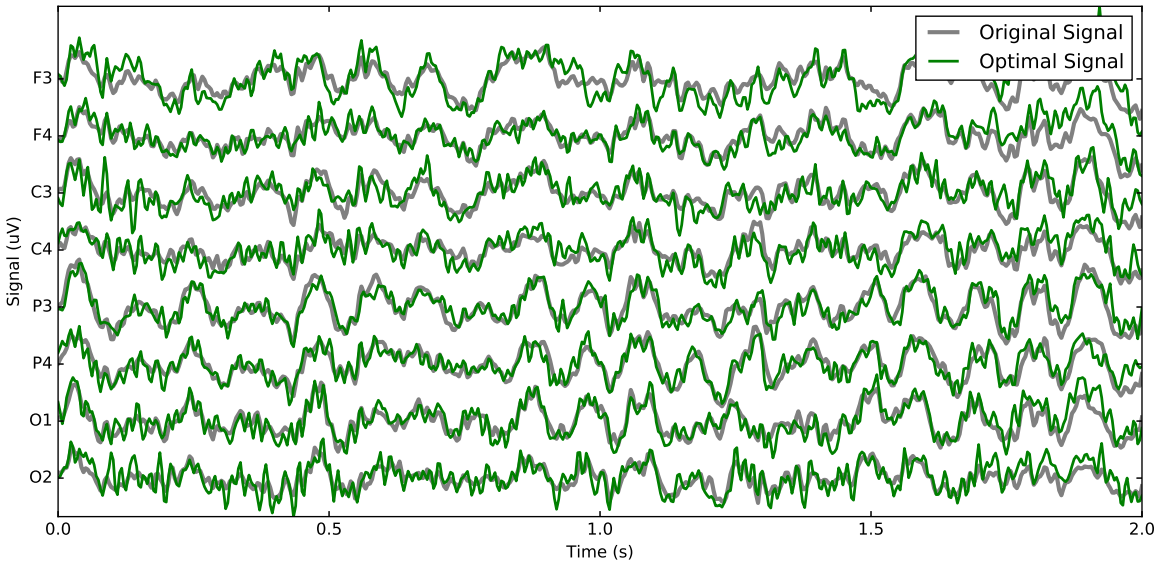
cient to allow us to draw conclusions about the optimal input sequences for our networks while also not departing dramatically from the original input segments. Note that in Section 4.2.5 we observed that the Song task is the second-lowest performing task, after Fist, and that Song is often confused with the Count and Rotate tasks. Given this confusion and also that we are using a smaller CNN-TR than when evaluating our test performance, it is unsurprising that this optimization process finds it more challenging to find an optimal input sequence that can be differentiated from Count and Rotate with a high degree of confidence.

In Figure 4.31, we see trace plots of both the original input segments, shown in gray, and our optimized input sequences, which are shown using the same color coding scheme as in the previous section. In Figure 4.32, we see the PSDs, generated using Welch's method with $\omega = 1s$, for both our original EEG segments, in the left column, as well as for our optimized input sequences, shown in the right column. By examining these figures concurrently, we are able to gain thorough insights into the changes that our optimization routine has made to the original inputs segments in both the time and frequency domains.

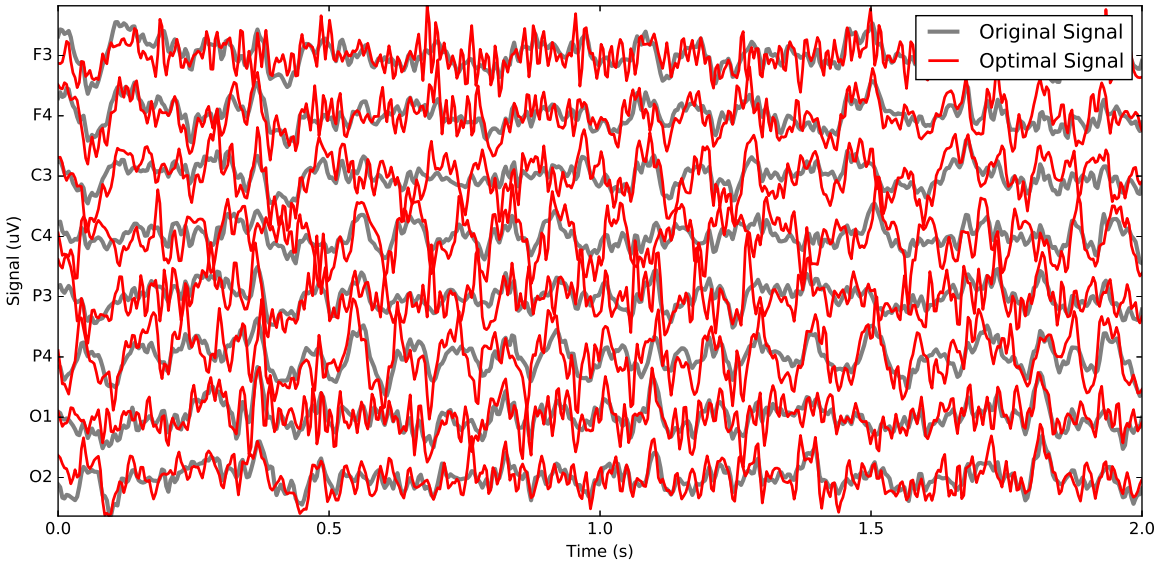
In Figure 4.31a, Figure 4.32a and Figure 4.32b, we can see that our optimization procedure for the Count task has dramatically increased the amount of γ activity across a number of channels, especially in channel C3 and the other channels over the central and occipital regions. Recall that γ activity in the frontal and central regions may be associated with speech production, which is likely involved in the Count task to some degree. Note that we have seen relatively high levels of γ activity associated with the Count task across subjects and our previous analysis of LDA for Subject-1 showed somewhat elevated power associated with mid-range γ in channel C3. It is also valuable to note that this network does utilize transfer learning and it is also possible that this particularly high level of γ activity learned for channel C3 may partially be explained by residual effects learned from other participants. There is also a notable increase in δ (0.5 – 4Hz) power in channel F3 and in α power over the parietal region, which roughly aligns with our previous observations from the PSDs and LDA weights for Subject-1.



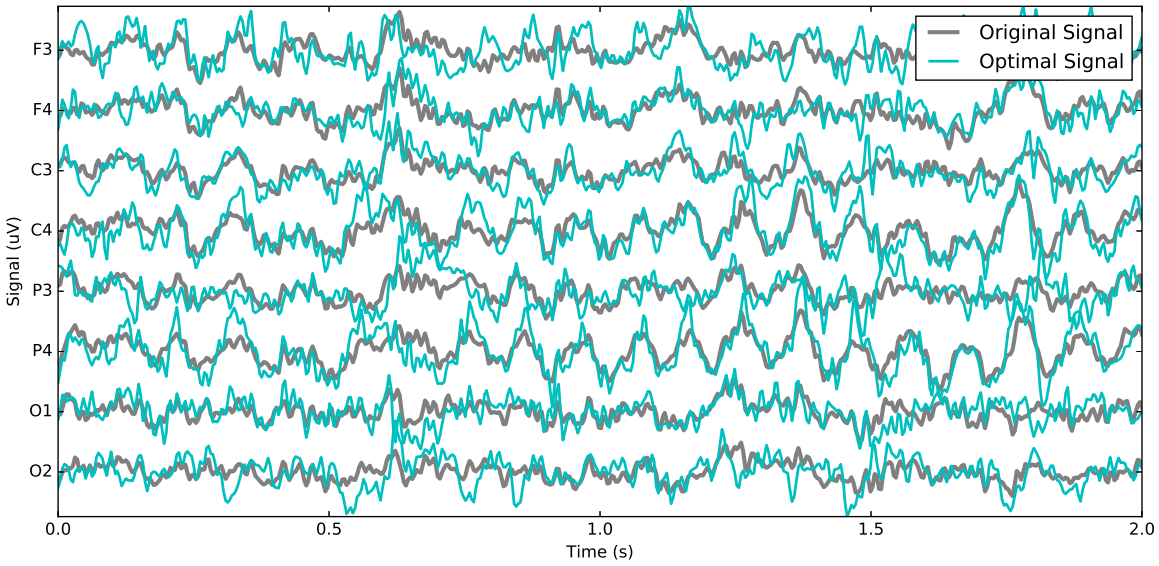
(a) Optimized input segment for Count.



(b) Optimized input segment for Fist.

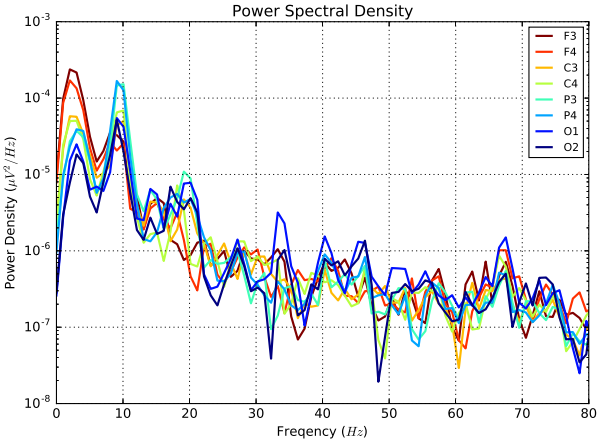


(c) Optimized input segment for Rotate.

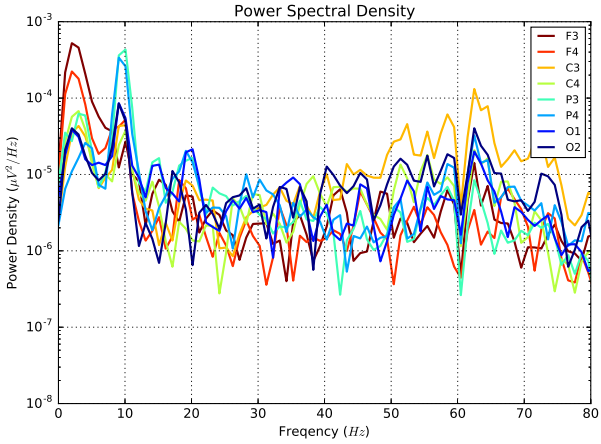


(d) Optimized input segment for Song.

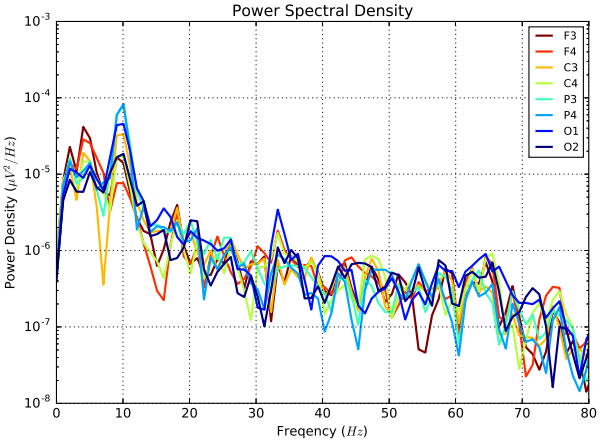
Figure 4.31: Optimized input sequences, shown in the same colors as in the previous section, superimposed over the actual EEG segments, before optimization, shown in grey. (a) For the Count task, there is a noticeable increase in γ activity across channels and especially in C3, C4 and O2. There are also noticeable increases α and slow moving frequencies in the frontal and parietal channels. (b) For the Fist task, the optimized sequence is not dramatically changed from the input sequence; although there are noticeable increases in γ across all channels. (c) For the Rotate task, there appear to be considerable increases in β and γ rhythms across all channels. (d) For the Song task, there again appear to be increases in β and γ rhythms but to a lesser degree than for Rotate. Notice the significant departure from the original input segment around 0.6s where there appears to be a small ocular artifact.



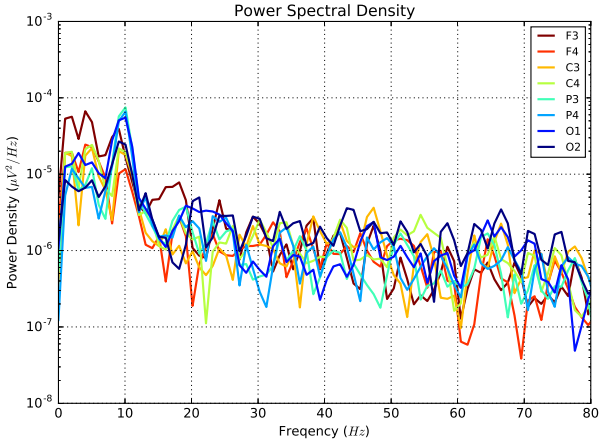
(a) PSD for actual Count segment.



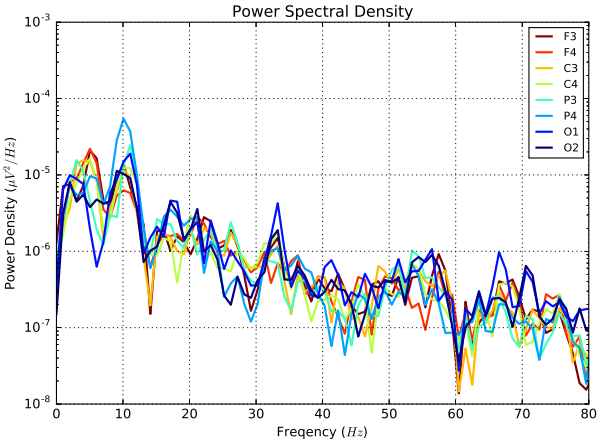
(b) PSD for optimized Count segment.



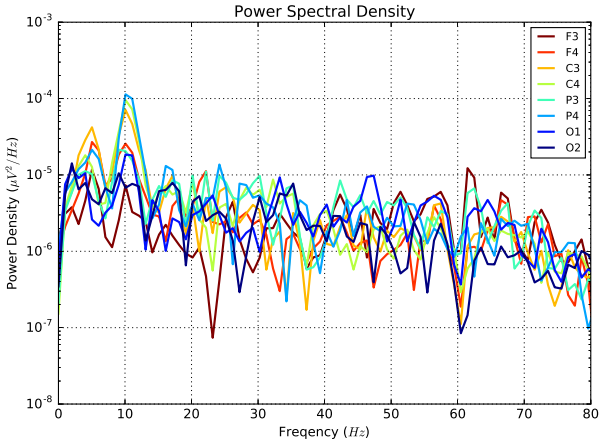
(c) PSD for actual Fist segment.



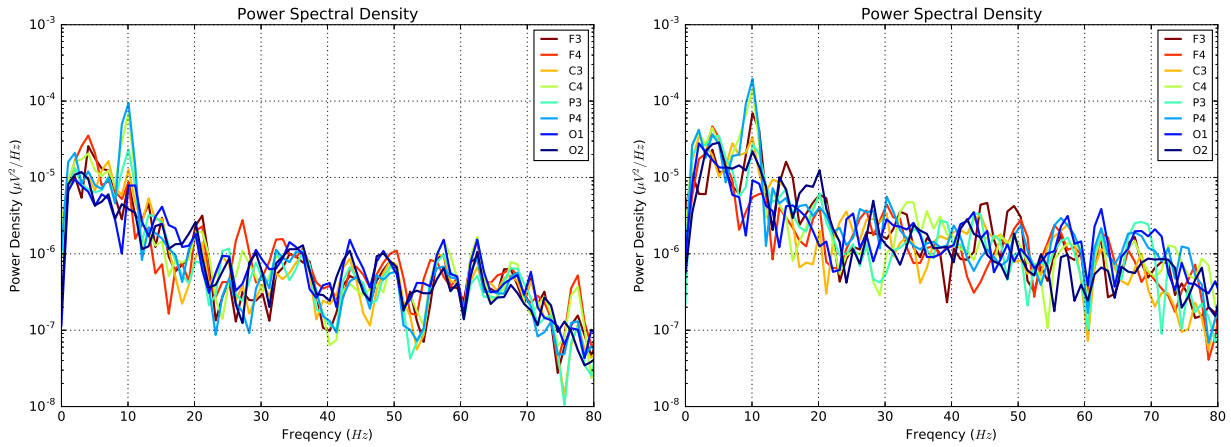
(d) PSD for optimized Fist segment.



(e) PSD for actual Rotate segment.



(f) PSD for optimized Rotate segment.



(g) PSD for actual Song segment.

(h) PSD for optimized Song segment.

Figure 4.32: PSDs of the actual input segments (left column) and optimized input sequences (right column) for each task. (a-b) For the Count task, the optimization process has increased δ power in F3 and α power in P3 and P4. β and γ power is also increased across all channels and especially in C3. (c-d) For the Fist task, δ and β power are increased in channel F3. Low α is increased in C3 and γ power is increased across all channels. (e-f) For the Rotate task, δ power is increased in C3 and α power is increased in several frontal, central and parietal channels. β power is considerably decreased in F3 and γ power is increased considerably across all channels. (g-h) α power is moderately increased across all channels except for O1 and F4 and γ is moderately increased across all channels.

In Figure 4.31b, Figure 4.32c and Figure 4.32d, we see a modest increase in β activity in channel F3 and a corresponding decrease in β in all other channels following optimization for the Fist task. Note that this may correspond to the β rhythm desynchronization that is often seen during motor imagery tasks but it is somewhat surprising that it is entirely in F3 rather than the other central and parietal channels. Note that there is also an increase in δ activity in F3 which does not directly correspond to our previous analysis of LDA for Subject-1. Again, this may be partially explained by the residual effects transfer learning. In other words, these changes in channel F3 may be reliable indicators of the Fist task across subjects, rather than for Subject-1 alone. Also notice that we again see increased γ activity across all channels following optimization, despite the fact that γ activity has generally not been associated with the Fist task in our previous analyses of PSDs and LDA. This may, however, align with our observations when analyzing the frequency responses of the convolutional units of our TDNNs, where we noted an increased reliance on γ in several sites. The fact that γ activity appears to be an indicator for

the Fist task in this case may offer further evidence that our CNNs are able to leverage patterns in the γ band that are not easily captured by PSDs, e.g., short-term nonstationary patterns or multivariate phase relationships.

In Figure 4.31c, Figure 4.32e and Figure 4.32f, we see that our optimization procedure has made considerable changes to the original signals for the Rotate task. First, notice that there is an increase in δ power in the C3 and in α power across several frontal, central and parietal channels. Note that in our previous analysis of LDA weights for Subject-1, we did see an increase in α power in the frontal region but not in the parietal region and we did not observe an increase in δ in C3. Our analysis of the weights for Subject-5 does, however, align with these observations. Again, these patterns may be general indicators for the Rotate task that are reliable across participants. Our optimization process has also introduced considerable increases in γ activity across all channels. Similar to our observation about γ for the Fist task, this change does not appear to align with our previous observations about PSDs for the Rotate task.

In Figure 4.31d, Figure 4.32g and Figure 4.32h, we see that our optimization procedure for the Song task has resulted modest increases in α power across multiple sites and, again, increases in γ activity across all channels. Note that these changes in α and, to some degree, γ do align with our observations about averaged PSDs and the LDA weights for Subject-1. Also notice that our optimization algorithm made significant changes to the original signal between 0.5–0.75 seconds, which may be an attempt to correct a small ocular artifact.

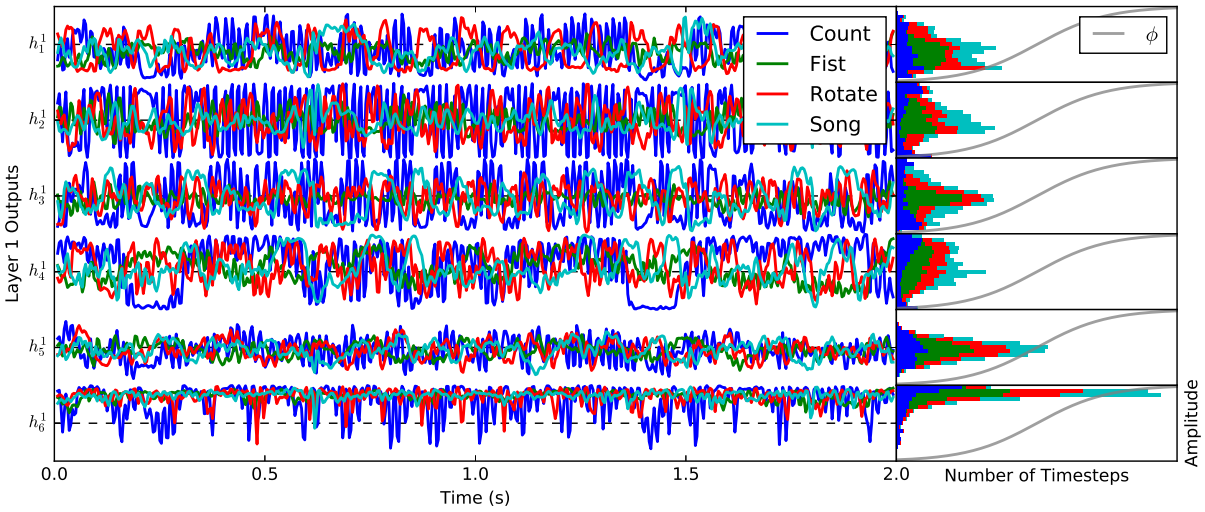
Overall, these observations about the optimized input sequences learned by our networks leads us to two primary conclusions. First of all, it appears that many of the patterns that our optimization process learns to emphasize appear to be broadly associated with each mental task across subjects instead of being focused solely on Subject-1. This suggests that our transfer learning process plays an important role in establishing the types of patterns that our networks learn to utilize. Second, our optimization algorithm tends to increase γ activity across a wide range of channels for nearly all tasks. This may provide evidence that γ activity is more important for our CNNs than for our PSD-based approaches. This may be a result of the ability of

CNNs to utilize patterns that PSDs are unable to leverage. It is important to note, however, that this may also potentially result from our optimization algorithm learning to produce unrealistic patterns. Since our networks have certainly not been exposed to all possible patterns found in EEG signals, it is possible that our optimization algorithm has learned to take advantage of shortcomings in our networks by exploiting patterns for which our networks do not generalize well. Further experiments are likely necessary in order to conclusively answer these questions, as will be discussed in Section 5.2.

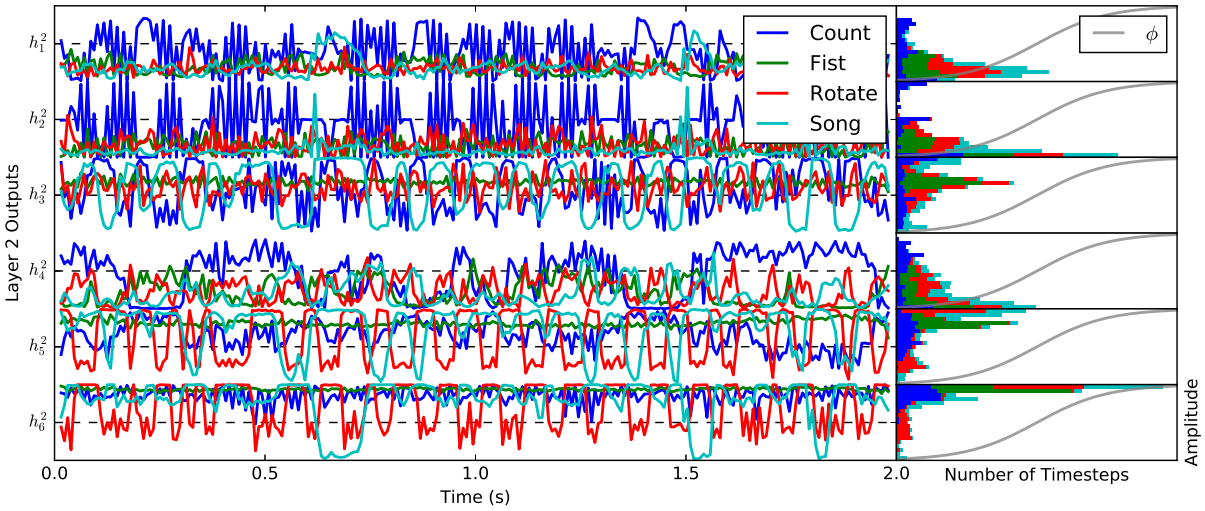
Network Activations for Optimized Input Sequences

In Figure 4.33, we see a plot similar to Figure 4.29 that shows the layer-wise activations produced by passing our optimized signals through our small CNN-TR network. In Figure 4.33a, we again see the our activations primarily falls on the linear regions of our transfer function; although to a slightly lesser degree than with our original inputs and, again, with the exception of h_6^1. Notice that the increased γ activity in our optimized Count and Rotate segments is now clearly visible across all units, especially for the Count task. α and γ rhythms for the Count and Rotate tasks appear to be associated with very nonlinear responses generated by h_6^1 while the other tasks have little influence on this unit. The ocular artifact in the Count task is again largely isolated to h_4^1 and α rhythms associated with the Rotate task appear to be highly correlated with the activations produced by h_1^1 and h_2^1. The Fist task appears to be associated with lower amplitude activations overall and the Song task appears to be associated with α rhythms in the activations produced by a number of units. These observations support our previous claim that the first layer of our CNNs largely performs linear spatiotemporal filtering that begins to isolate various signal components.

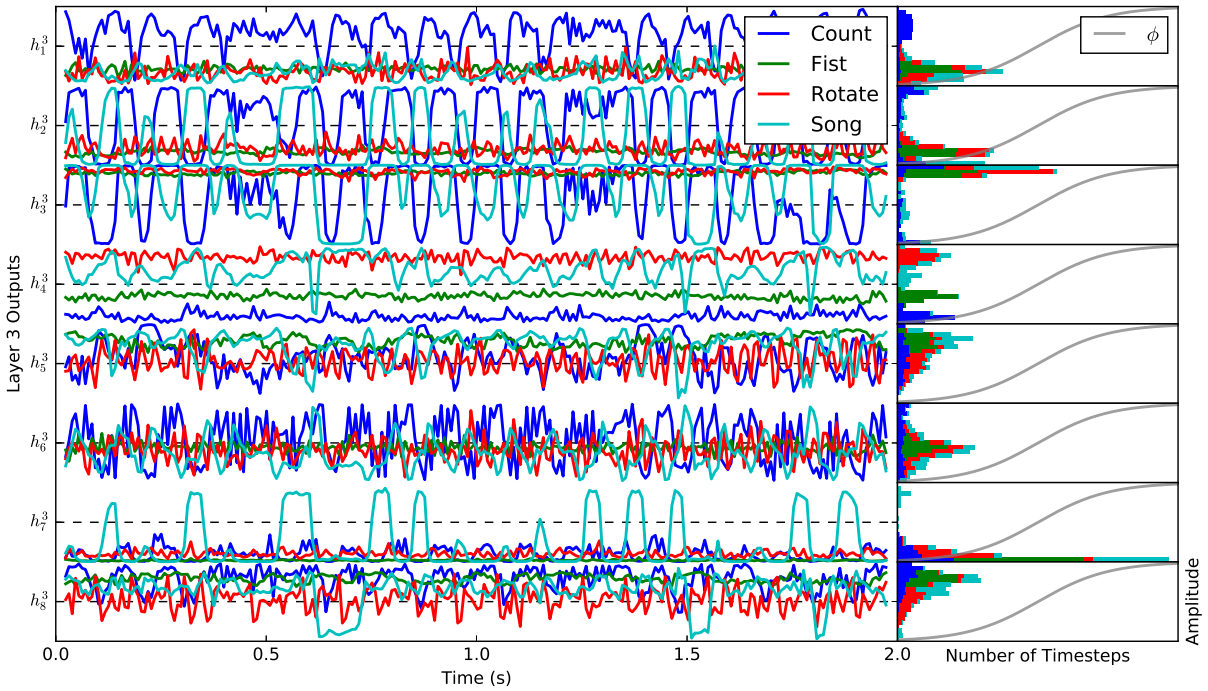
In Figure 4.33b, we see increasingly nonlinear behavior in the activations produced by all units as well as some degree of specialization for several units. For example, h_1^2 and h_2^2 generate strong responses in the γ band for the Count task while h_3^2 generates α rhythms for the Count and Song tasks. Also, h_4^2 generates relatively slow moving responses for the Count and Song tasks and h_5^2 and h_6^2 generate δ and α rhythms for the Rotate and Song tasks.



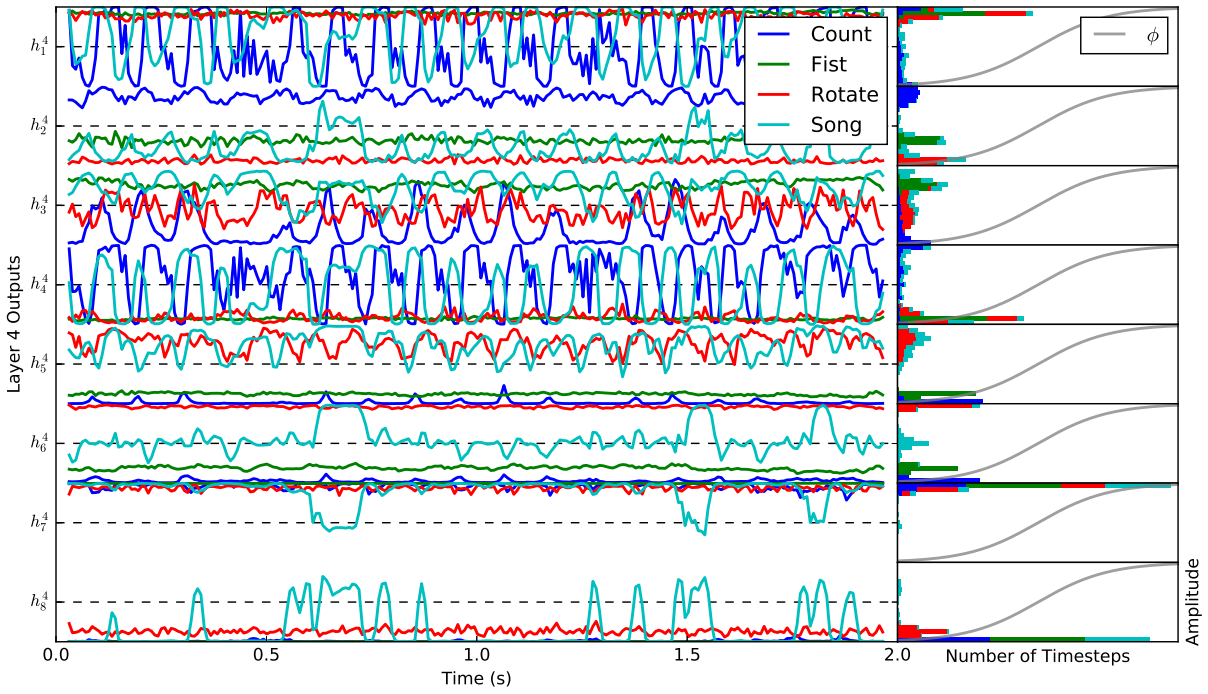
(a) Layer 1 activations.



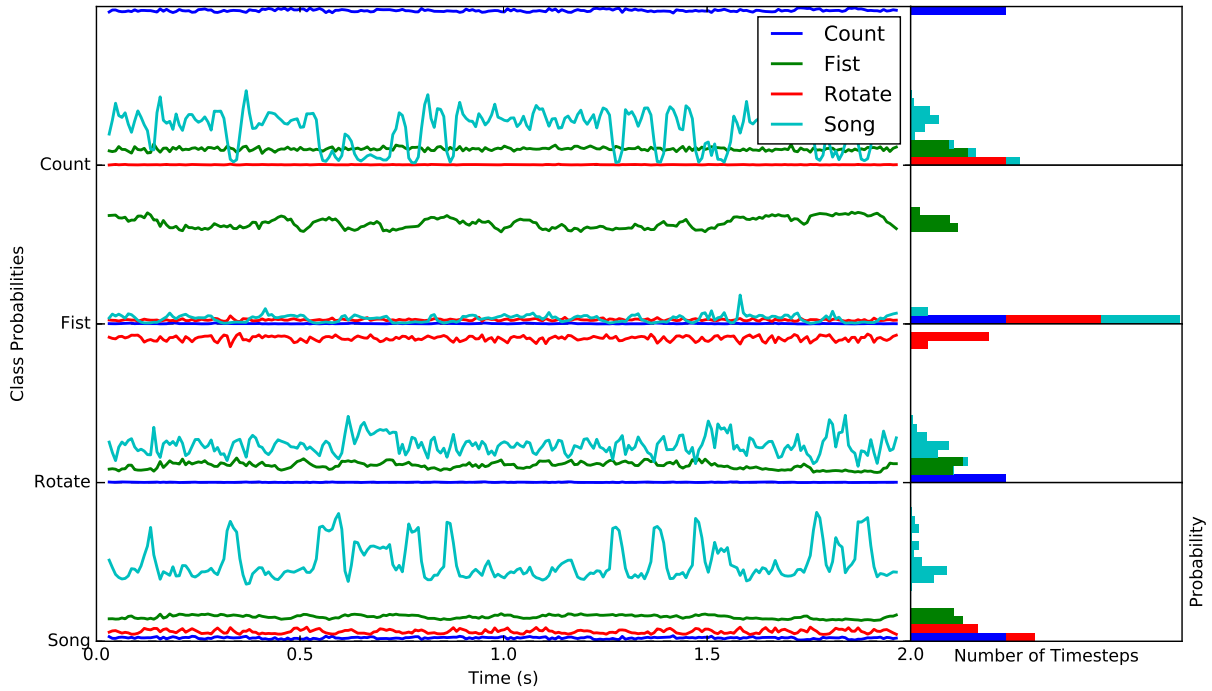
(b) Layer 2 activations.



(c) Layer 3 activations.



(d) Layer 4 activations.



(e) Layer 5 estimated class membership probabilities at each timestep.

Figure 4.33: Activations produced at each layer by our small CNN-TR for Subject-1 when processing our optimized input sequences. (a) The first layer continues to act primarily as a bank of linear filters, with only h_6^1 show significant nonlinear activity. Notice the high levels of γ activity for the Count task and α activity for the Rotate task and the isolation of the ocular artifact in h_4^1. (b) The second layer shows increasingly nonlinear behavior and specialization. Notice, for example, that h_1^2 and h_2^2 respond primarily to high frequency patterns for Count while h_6^2 responds primarily to the Rotate task. (c) In the third layer, after pooling, the activations begin to move more slowly and continue to show increasing specialization. Notice the clear separation across all tasks in h_4^3. (d) The activations in the fourth layer contain very little high frequency information and again show specialization and nonlinear behavior. (e) The output of the readout layer shows the predicted class membership probabilities at each timestep. Notice that we have perfect classification for all timesteps; although the probabilities are often considerably less than one for the Song and Fist tasks.

In Figure 4.33c, following our change of scale from pooling, we again see a transition to slower frequency activations and additional specialization of the units. For example, the activations produced by h_4^3 are almost completely separated for all four tasks. h_1^3 appears to respond primarily to the Count task, h_2^3 and h_3^3 respond primarily to the Count and Song tasks. h_7^3 responds intermittently to the Song task and h_8^3 responds strongly to the Rotate task.

In Figure 4.33d, we see that the activations produced by our final layer now contain very little high frequency information and even more specialization for each task. Notice that h_1^4

responds almost exclusively to Count and Song and h_2^4 completely isolates Count. Also, h_4^4 separates Count and Song from Fist and Rotate while h_5^4 separates Song and Rotate from Count and Fist. h_6^4 separates four tasks across the amplitude of its responses and h_7^4 and h_8^4 respond intermittently to the Song task.

In Figure 4.33e, we see the final estimated class membership probabilities for each task across all timesteps. Notice that our optimization algorithm was able to achieve the highest class membership probabilities for the correct label at all timesteps for all classes. It it worth noting, however, that while the Count and Rotate tasks achieve estimated probabilities near one across all timesteps, the probabilities for the Fist task are closer to 0.65 and for the Song task are closer to 0.5. As we have previously discussed, this appears to be related to the fact that these tasks tend to be classified with a lower degree of success in general and, as a result, our ALOPEX optimizer was unable to find input sequences that achieve predicted probabilities near one for these tasks.

This examination of the network activations produced when processing optimized input sequences has lead us to several valuable observations. First of all, this experiment aligns with our previous experiments where we have seen that the initial layers of our networks act primarily as linear spatiotemporal filters. Subsequent layers tend to have increasingly nonlinear behavior with some units appearing to produce nearly binary switching dynamics in response to some types of patterns. We have also seen that our optimized input sequences and their corresponding network activations tend to have relatively large amount of high-frequency information. This provides evidence that high frequency information, including γ rhythms, is viewed with relative importance by our networks and can be highly informative when attempting to discriminate between tasks. The precise role of high-frequency information in EEG signals remains somewhat unclear, however, and future experiments are required in order to fully understand this topic. We have also observed that high-frequency information tends to be gradually transformed into slower moving activations as the signals progress through the layers of our networks. As noted in our previous analyses, our networks appear to be migrating information

from shorter time scales in the input signals to longer time scales that that are more useful for assigning class labels across the duration of the segment. This provides evidence that our networks are learning at multiple time scales and partially explains why pooling is beneficial. We have also noticed that many artificial neurons become highly specialized and produce activations primarily for a small number of mental tasks, especially in the later layers of the network. This demonstrates that our proposed CNNs are, indeed, learning hierarchical and multiscale representations with increasing specialization and decreasing granularity across the layers of the network.

Chapter 5

Conclusions

Now that we have completed the presentation of our methods, results and analyses, we will provide a high-level summary and discussion of our key results and contributions. We will begin by reiterating our case for using CNNs in asynchronous BCIs and by summarizing our findings about how these networks should be designed in order to achieve good performance. We will then discuss our experimental results, at a high level, along with the primary conclusions that can be drawn from these experiments. We will then offer some potential directions for future research that might explore remaining open questions. Finally, we will offer some concluding remarks about the potential of this research and the future of asynchronous BCIs.

5.1 Discussion and Summary

The ability to reliably and quickly classify EEG signals that are spontaneously produced during various mental tasks may enable the development of asynchronous BCIs that can be controlled in a fluid, self-paced manner in the absence of external stimuli. These types of BCIs would have valuable applications in assistive technologies and might, eventually, be used in other types of human-computer interfaces. Current approaches for classifying these types of EEG signals have not, however, achieved the levels of performance that are required for many practical applications. We assert that a primary obstacle for current approaches is their reliance on signal representations and feature extraction procedures that require strict prior assumptions and have a limited ability to capture some types of patterns. For instance, we demonstrated in Section 2.1.1 that Power-Spectral Densities (PSDs) are unable to capture phase differences across multiple channels, they are unable to capture the ordering of short-term events and they are poorly suited for modeling patterns that are nonstationary. Similarly, common-spatial patterns may fail to differentiate between relatively simple patterns unless the signals are filtered to lie within very narrow frequency bands. Common-Spatial Patterns (CSP) is also

not able to model patterns that are nonlinear or nonstationary over time. Given that multivariate phase differences and patterns that are nonlinear, transient and nonstationary are known to exist in EEG signals, it appears that these approaches are less-than-ideal for solving this classification task. It is also important to note that there are many aspects of EEG signals that are not yet fully understood and that *any* prior assumptions about the nature of these signals should be avoided to the greatest extent possible in order to prevent us from failing to identify and leverage patterns that have not yet been discovered or fully characterized.

Deep artificial neural networks may offer solutions to these problems because they can be designed to be generic and rely on few prior assumptions about the nature of their inputs. Indeed, biologically inspired artificial neural networks that operate on raw data with minimal preprocessing and feature engineering are now among the best-performing machine learning approaches in numerous domains. In particular, multilayer convolutional neural networks are capable of automatically forming hierarchical, nonlinear, shift-invariant and multiscale representations of their inputs. Although CNNs have been successfully applied to a number of important problems, there has not, until now, been an in-depth exploration of using CNNs for classifying EEG signals in asynchronous BCIs. In order to address this gap in current research, we have set out to explore the use of CNNs for classifying EEG signals recorded during mental tasks that are not time-locked and with a specific focus on potential applications in assistive technologies.

The dataset that we have used to explore this potential application of CNNs was collected by our research group using a portable, eight-channel EEG system, as described in Section 3.5.3. Fourteen subjects participated in this study, ten of which had no motor impairments with EEG recording taking place in a laboratory while the remaining four had motor impairments and recording took place in their homes. Each of these participants performed four imagined mental tasks: silently count backward from 100 by threes, imagine making a left-handed fist, visualize a rotating cube and silently sing a favorite song. Each task was performed for five trials each lasting ten seconds, which we believe to be a reasonable calibration period for a BCI sys-

tem. This dataset was collected with the intent of accurately representing the kinds of data that might be used in real-world BCIs, i.e., minimal training data, portable hardware, users in a target demographic and realistic operating conditions. Such a dataset was not previously available for mental-task BCIs and we have now made this data publicly accessible.[21]

In order to compare our proposed methods with current approaches, we have established three baseline classifiers that closely resemble methods used in other research studies. These baseline approaches utilize Welch's method to compute smoothed PSDs followed by the application of a relatively standard classification algorithm. We have selected three algorithms for this purpose that cover a range of model complexities, varying from linear discriminant analysis to quadratic discriminant analysis to fully connected, two-layer artificial neural networks. During our analysis of the hyper-parameters involved in these approaches, in Section 4.1.1, and their final test results, in Section 4.2.1, we have found that these classifiers tend to be susceptible to overfitting. The nonlinear QDA and ANN-based approaches often overfit the data even when combined with large amounts of smoothing and regularization. Due to this propensity for overfitting, simple linear classifiers generally outperform nonlinear methods in this setting and, as a result, we use the performance of our LDA-based classifiers as our primary benchmark.

The design of our CNN architectures was guided by the seven challenges associated with classifying these types of EEG signals that we identified in Section 1.1: achieving time invariance, capturing nonlinear and multiscale spatiotemporal patterns, handling noise and undersampling, high dimensionality, interpretation of the models and near real-time computational performance. Over the course of Chapter 3, we described several variants of the CNN architecture that we have specifically designed to address these challenges. These networks utilize single-axis convolution, label aggregation readout layers, average pooling, optimization with scaled conjugate gradients and transfer learning, which will be reviewed in detail shortly. We also introduced our open-source CEBL_3 software package for performing BCI experiments,

[21] https://www.cs.colostate.edu/eeg/main/data/

which is freely available for download under an open-source license.[22] This software package includes implementations of these networks as well as the other core functionality used in our experiments.

In previous research, we have explored the use of single-layer time-delay neural networks (TDNNs) for classifying EEG signals. As we described in Section 3.2.2, it turns out that the time-delay embedding operation used in these networks is equivalent to performing a convolution over time with dense connectivity across channels, provided that our class labels are properly aligned. Performing convolutions across time allows us to achieve time invariance, which is essential because our signals are not time locked. Fixed connectivity across channels ensures that different spatial patterns have distinct representations while also keeping our networks relatively small and simple. Note that it is also not straightforward to perform convolutions over the nonplanar surface of the scalp and that EEG signals also do not typically have enough channels to justify the use of spatial convolutions. When used in combination with nonlinear transfer functions, these networks are capable of modeling nonlinear and nonstationary patterns, provided that the width of the convolutional kernel is sufficiently large. We have also demonstrated that this type of convolutional unit can be viewed as a learned, nonlinear and multivariate Finite Impulse-Response (FIR) filter. This allows us to perform some interpretation of the patterns that these networks learn to leverage by using the Fourier transform to extract the frequency response of these filters. These frequency responses provide a relative measure of importance and an indicator of nonlinear behavior across the frequency spectrum.

We have also proposed the use of label aggregation readout layers instead of fully connected readout layers. In this approach, a linear layer is used to assign class labels at each time step of the signal. A full segment of the signal can then be assigned a class label by summing the log likelihoods of these predictions over time. This is in contrast to fully connected readout layers, which have dense connectivity across all timesteps of the input segment. The use of label aggregation readout layers allows us to keep the number of free parameters in our net-

[22] https://github.com/idfah/cebl

works low, which helps to make our classifiers more robust to noise, undersampling and high dimensionality. As we have shown in Section 4.1.3, networks with fully connected readout layers have many connections across time if they do not incorporate pooling and have many connections in the convolutional layers if many layers with pooling are stacked together. Our model selection experiments in Section 4.1.3 and our final test results in Section 4.2.2, demonstrate that networks with fully connected readout layers suffer from catastrophic overfitting due to this large number of free parameters and because we have a relatively small amount of noisy and high-dimensional training data. Label aggregation readout layers, on the other hand, only have connections to the outputs of the convolutional units, and not across all timesteps, which greatly reduces the number of free parameters. Although this design limits the ability of our networks to capture information across a full EEG segment, it appears that these networks are able to learn sufficient temporal information through the local receptive fields of the convolutional units and by aggregating information over the course of time.

Label aggregation readout layers can also be used with arbitrary or variable length input segments. Evidence accumulation strategies could, potentially, be employed to assign class labels only after the system has established a high degree of confidence in the user's intent. In other words, label aggregation readout layers provide a great deal of flexibility when designing BCI applications and attempting to achieve the best possible user experience. Label aggregation readout layers also allow for easier interpretation than fully connected layers because they produce a predicted class membership probability for each timestep. This allows us to directly correlate a network's predictions with a window of time in the input signal.

Another approach that we have used for improving the performance of our CNNs is to incorporate transfer learning across multiple subjects. In this approach, a network is first initialized by performing a number of training iterations over the combined training data for all subjects, except for the subject at hand. The network is then fine-tuned by applying a number of additional training iterations using only the data for the subject of interest. This allows our networks to effectively utilize additional training data, provided that there are common pat-

terns in the EEG signals across multiple subjects. In our analysis of PSDs in Section 4.3.1 and Section 4.3.2, which will be discussed shortly, we have found that these EEG signals contain both patterns that are common across multiple subjects as well as patterns that are specific to individuals. For example, we have found that there tends to be an increase in γ activity for the Song task on average but that there are also considerable differences in the spatial layout of these rhythms between Subject-1 and Subject-5. This notion that there are both common and individual patterns in these signals supports the use of our transfer learning approach because it is able to first identify common patterns and then allow for fine-tuning toward the unique aspects of the signals for each individual and, potentially, for the same individual across multiple BCI calibration sessions.

In Section 4.1, we explored the benefit of stacking multiple layers together to form deep networks. These experiments demonstrated that when using CNNs with label aggregation and narrow, three-timestep, convolutional kernels, stacking multiple convolutional layers improves validation performance up to about three or four layers. Although the introduction of pooling did not lead to a clear improvement in performance, networks that included pooling did tend to achieve results that were at least as good while using fewer convolutional units. As described in Section 3.3 and Section 3.4.2, stacking convolutional layers and incorporating pooling increases the duration of the impulse response and, therefore, the potential memory capacity of the network. This suggests that our networks with pooling tend to rely more heavily on longer-term temporal information rather than requiring more artificial neurons to more carefully model spatial patterns. Stacking convolutional layers also permits the convolutional units in our networks to achieve increasingly fine-grained filtering characteristics and encourages our networks to learn hierarchical and multiscale representations of the input signals. Since a network with multiple convolutional layers also has a higher ratio of applications of our nonlinear transfer function to the total number of parameters in the network, it may also be able to model highly nonlinear patterns while requiring relatively few free parameters.

In Section 4.2.2 we provided an in-depth comparison of the test performance of both single-layer TDNNs and multilayer CNNs. These results demonstrated that there is actually *not* an improvement in test classification accuracy for CNNs with stacked convolutional layers that have narrow convolutional kernels relative to single-layer TDNNs with wider convolutional kernels, unless transfer learning is incorporated into our training procedure. This suggests that the additional training data provided by transfer learning is required in order for our networks to take full advantage of the benefits provided by stacking multiple convolutional layers. This can be explained by the fact that multilayer networks are more susceptible to vanishing gradients and by the increasingly complex behavior of the network's activations as they pass through multiple layers and applications of the pooling and transfer functions. Note that our multilayer CNNs with transfer learning also outperform single-layer TDNNs with transfer learning and that TDNNs achieve only a minor benefit from the introduction of transfer learning. These observations support our claim that multilayer networks are advantageous over single layer networks, provided that they are exposed to enough information during the training procedure. This also supports our claim that transfer learning is an effective method for improving performance by exposing our networks to additional training data, but only for multilayer networks.

When analyzing the final test performance for all of our methods in Section 4.2.2, we have found that our CNN-TR networks, which include stacked convolutional layers and transfer learning, achieve a mean test classification accuracy of 57.86%, which is 8.57% higher than the 49.29% achieved by our baseline classifiers that use PSDs with LDA. This difference in mean performance is statistically significant ($p = 0.04$) when we do not correct for multiple comparisons; however, it fails to be significant when we correct for multiple comparisons across all of the methods that we have explored ($p = 0.74$). It is valuable to note, however, that this correction for multiple comparisons is across all eight of our baseline and proposed methods, despite the fact that our model selection experiments, which used only the validation performance for the first five participants, clearly directed us toward using LDA and CNN-TR. Given the significance of our direct comparison between CNN-TR and LDA along with the fact that our CNNs

outperform LDA for the vast majority of participants, some of which achieved performance improvements as high as 40%, we believe that there is convincing evidence that our proposed CNNs offer a considerable performance benefit over our baseline methods.

We have also observed test classification accuracies as high as 90% for two individuals when using CNN-TR, which is 10–20% higher than with LDA, while several other individuals achieved near the expected random of 25% across all methods. This suggests that CNN-TRs may yield more benefit for some individuals or circumstances and also that some of these participants would likely not have been able to attain control of a BCI. Several other studies have similarly noted that some individuals were unable to control various types of BCIs [133]. During our examination of aggregate PSDs in Section 4.3.1, we noted that many of the low-performing individuals in our dataset have either high levels of erratic peaks in γ power, which may indicate poor signal quality or contamination from muscle movement, or else unusually low power in the α band, which might be related to a lack of internal focus or attention [137, 138]. These phenomenon were especially prevalent among the individuals with motor impairments in their homes; although it is difficult to conclude that this is typical given our small number of subjects with impairments. Further experiments with larger sample sizes would be valuable for establishing the relationships between α rhythms versus BCI performance as well as the differences in EEG signals between individuals with and without various types of motor impairments.

We also compared our test results with several external studies in terms of information transfer rates in Section 4.2.3. Information transfer rate, which is defined in Section 3.5.4, is a metric that is commonly used to compare results across different BCI studies because it incorporates the number of classes and the rate at which decisions are made in addition to classification accuracy. We have noted, however, that information transfer rate is an imperfect metric because it does not account for how classification accuracy changes as decision rates are varied. Nevertheless, information transfer rate does provide some useful insights into the relative performance of methods with different experimental designs. The mean information transfer rate achieved by our CNN-TRs is 15.82 bits-per-minute (bpm) with some individuals achiev-

ing information transfer rates as high as 41.18 bpm. These results compare favorably with our calculation of mean information transfer rates achieved by other studies, which range from 8.96–19.30 bpm, despite the fact that these studies typically use more training data and artifact rejection algorithms and do not involve participants with motor impairments in realistic environments [33, 34, 52, 73]. Note that our baseline classifiers achieve a mean information transfer rate of 9.35 bpm, which is slightly higher than the information transfer rates achieved by Keirn and Aunon when using a similar method [33]. This supports our claim that our baseline methods are representative of other approaches that use PSDs. The highest information transfer rates among the studies that we have reviewed were achieved in the recent work by Lawhern, et al., who also used a variant of the CNN architecture [52]. Although our comparisons with external studies are somewhat cursory, these comparisons do provide evidence that our methods are at least on par with the state of the art and also that CNNs are advantageous relative to other current approaches.

In Section 4.2.4, we also analyzed the tradeoff between accuracy and responsiveness for both LDA and CNN-TR by examining the performance of these methods across a range of decisions rates. We believe that it is important to examine how performance changes as the decision rate is varied for any classification approach that might be used in BCIs. This is because classification accuracy may not, and in this case does not, vary linearly as the decision rate of the classifier changes. In this setting, neither classification accuracy nor information transfer rate can be fairly compared across experimental paradigms or applications with different decision rates, unless this full performance curve is examined. The results of these experiments demonstrate that CNN-TR achieves both higher classification accuracies with slow decision rates as well as higher information transfer rates as decision rates approach continuous control. Note that with decisions made every $\frac{1}{8}$ of a second, LDA achieves a mean ITR of 26.27 bpm while CNN-TR achieves over 55.22 bpm. With decisions made every 10 seconds, on the other hand, LDA achieves a mean classification accuracy of 57.41% while CNN-TR achieves 71.43%. These experiments support our claim that CNN-TR is especially well suited for BCI applications that

require near continuous control while also demonstrating that CNN-TRs achieve better performance for applications that require slower but more reliable control.

We also carefully examined the performance attained by each of our four mental tasks when using LDA, TDNNs and CNN-TRs in Section 4.2.5. These results show that the Count task generally outperforms the other tasks while the Fist task is, on average, the worst performing task and Rotate and Song are generally in between. This observation is notable because many current asynchronous BCIs focus primarily on motor imagery, yet we have found, at least for this dataset, that the Fist task performs poorly relative to the other three tasks. This supports our claim that the more generic mental-task paradigm may be more flexible and better-suited for use in assistive technology. During our interactions with participants that have motor impairments, we have also, informally, found that many of these participants prefer non-motor tasks, partially because they have a limited ability or inability to perform the actual corresponding motor task. This observation aligns with the work by Friedrich, et al., which suggest that BCI users with motor impairments often prefer and achieve better performance with non-motor imagery tasks [37]. It is also valuable to note that the Count task appears to achieve the greatest performance benefit from using TDNNs and CNN-TRs instead of LDA. This suggests that the Count task may elicit patterns that are nonlinear, involve multivariate phase differences or are otherwise challenging for PSDs or LDA to represent. The Rotate and Song tasks achieve little, if any, benefit from using TDNNs over LDA but achieve a modest benefit from the introduction of multiple layers and transfer learning. This may suggest that these tasks share common aspects across individuals or that the patterns elicited by these tasks naturally lend themselves to hierarchical, multiscale representations. With respect to task confusion, we have seen that the Fist task is often confused with Count and that Song is often confused with Rotate. Given the confusion of these tasks along with the relatively high performance of the Count task and the low performance of the Fist task, we recommend that potential BCI users and future experiments might attempt swapping the Fist or Song tasks for an alternate task while generally using the Count and Rotate tasks unless they are clearly not working for a given individual.

In addition to examining the performance of our classifiers, we have also performed a number of experiments in Section 4.3 that aim to analyze the types of patterns contained in these EEG signals and how our baseline and proposed methods learn to leverage these patterns for classification. One of the initial methods that we have used for exploring these EEG signals is to examine PSDs that are averaged across participants and channels for each mental task. Since PSDs represent the power contained in the signals across the frequency spectrum, this approach allows us to easily visualize the relative power for the various rhythms contained in these EEG signals. Although our sample size, i.e., the number of subjects, is not large enough to draw statistically significant conclusions about how the power content of EEG signals varies across a wider population users, these experiments do provide valuable insights into how these signals tend to vary across the individuals in our dataset. These experiments also provide corroborating evidence for some of the observations that we and other research groups have made about how EEG signals tend to vary across these mental tasks.

This analysis of aggregate PSDs has shown that there tends to be considerable differences in the peaks and precise frequencies of the dominant α rhythms for each of our four mental tasks. This suggests that α rhythms are likely important for discriminating between these tasks. We have also observed a relative decrease in β and low frequency α rhythms during the Fist task, which aligns with the expected response for motor imagery [27, 29, 30]. We have also observed that there tends to be a broad increase in γ activity during both the Count and Song tasks relative to the Fist and Rotate tasks. Although it is often difficult to distinguish between neural and muscular sources of γ-band activity, the relative increase in γ power for these tasks does appear to be consistent and we have observed little evidence of contamination from muscle movement. We have also noted that our classifiers do not have a particularly high rate of confusion between the Fist-Rotate and Count-Song task pairs, despite the similarity in γ power for these tasks. This suggests that other information is being utilized for classification, which likely includes differences in α and β power and the spatial layout of these patterns. In the case of our

CNNs, it is also possible that nonlinear or multivariate phase information in the γ band might be leveraged, which is examined in our analysis of the patterns leveraged by our CNNs.

In addition to visualizing PSDs that are averaged across individuals and channels, we have also examined the standardized, log-transformed PSDs for two individuals, Subject-1 and Subject-5, along side the weights learned by our LDA classifiers. This exploration has lead us to several valuable observations. First of all, we have noted that for Subject-1 there is an increase in γ power over the frontal lobes of the brain during the Count task, which might be linked to the speech production and rehearsal component of this task. We have noted, however, that the increase in γ power is less pronounced during the Song task for Subject-1 and Subject-5 and that these high levels of γ power for the Count task occur over nearly all regions of the brain for Subject-5. In any case, our LDA classifiers have only modestly high weights associated with this γ activity, which suggests that our baseline methods do not place an especially high level of importance on power in the γ band. We have also seen for both Subject-1 and Subject-5 that there tends to be relatively high levels of α power over the frontal and parietal lobes but not over the occipital lobes during the Rotate task. We believe that this α synchronization across the frontal and central lobes of the brain indicate that activity is being broadly suppressed in these areas while α desynchronization in the in the occipital lobes may be an indicator of increased activity corresponding to the processing of visual information. We have also seen relatively low power in the α and β bands over the frontal, central and parietal regions for the Fist task, which corresponds to the typical desynchronization that occurs during motor imagery. For both the Fist and Rotate tasks, our LDA classifiers do have relatively high weights associated with α-band activity but over different regions of the brain. This explains why our classifiers have a relatively low rate of confusion for these tasks, despite their apparent similarity in PSDs averaged across subjects and channels.

Examining the weights of convolutional networks is more challenging than examining the weights of our LDA classifiers due to the fact that a convolutional unit has a local receptive field that slides across time rather than direct connectivity to input features. CNNs also exhibit non-

linear behavior and mapping the magnitudes of weight connections backward through multiple convolutional layers is extremely challenging since there is no enforced spatial layout after the first layer. As we have previously discussed, however, it is possible to view our convolutional units as FIR filters and extract the frequency and phase responses of these filters, before the application of the nonlinear transfer function, using the Fourier transform. In Section 4.3.3, we performed such an analysis for a single-layer TDNN trained over the data for Subject-1. Although this approach does not allow us to associate specific weights with different mental tasks, this experiment has provided us with several valuable insights into the frequency bands that this network has learned to view as important. In particular, we have observed that these convolutional layers tend to focus on frequencies in the α and β bands. The degree of amplification introduced by these FIR filters also suggests some degree of nonlinear behavior is often associated with these frequency bands. We have also noted that some of the convolutional units in this network focus heavily on information in the γ band. This focus on information in the γ band is somewhat contrary to our observation that our LDA classifiers rarely associated large weights with γ power. We have also noted that the phase response of these FIR filters is very nonlinear across the frequency spectrum. This is in contrast to near-ideal FIR bandpass filters, which introduce a nearly linear phase shift. This implies that our networks are likely incorporating information related to phase or that they are learning to filter multiple frequencies simultaneously. This provides some supporting evidence that phase information and sophisticated spatiotemporal relationships may be valuable for classification.

Although our exploration of convolutional layers as FIR filters was informative, it did not allow us to explore the response of our networks to each mental task or to explore multilayer networks and it did not provide us with a complete understanding of how our networks are processing information. In order to explore these topics further, we have analyzed the activations produced by each layer of a reduced-size multilayer CNN-TR for Subject-1 in Section 4.3.4. These experiments allowed us to make a number of important observations. First of all, we noted that the activations produced by the first layer of our networks tend to fall near the lin-

ear region of our transfer function, which suggests that the first layer primarily acts as a linear filter. In subsequent layers, we see increasingly nonlinear behavior, with some artificial neurons producing nearly binary responses. Many of these neurons also appear to have relatively high bias values, which allows them to act similar to switches that produce an activation in response to a particular combination of inputs and to produce a constant activation otherwise. In this way, the convolutional layers of our networks are also able to respond in increasingly specialized ways to particular inputs that tend to be associated with a given mental task. We have also observed that the activations produced by the later layers of these networks tend to contain primarily low frequency oscillations while the activations produced by earlier layers contain both low and high frequency oscillations. This suggests that our networks tend to migrate information from high-frequency, short-term time scales to slower-moving, longer-term time scales. This makes intuitive sense because our networks are optimized to produce correct labels across as many timesteps as possible. In other words, if our networks identify a pattern at a short time scale that is indicative of a given class, the network can stretch this response out over time in order to produce a correct class label for as long as it's impulse response allows. Combined, these observations demonstrate that our networks are, in fact, learning to model our signals in a nonlinear and hierarchical way with increasing granularity and across multiple time scales.

In our final series of analysis experiments, we used optimization to find artificial input sequences for our reduced-size CNN-TR for Subject-1 that produce high estimated class membership probabilities for each class. In other words, we generated artificial signals that are nearly optimal for each class from the perspective of a trained CNN-TR. In order to achieve this, we first seeded our optimization procedure with the best-performing EEG segment for each class in the test partition, and then used the ALOPEX optimization algorithm to maximize the sum of the estimated class membership likelihoods across all timesteps for the corresponding class label. In addition to the fact that ALOPEX is gradient-free, which allows us to easily perform optimization through our initial bandpass step, ALOPEX was also originally designed for the

purpose of mapping the local receptive fields of biological neurons, which seems appropriately similar to this use case. In these experiments, we first analyzed the PSDs of our original seed EEG segments along side the PSDs of our optimized input sequences for each class. From this analysis, we were able to identify several frequency-domain patterns that our networks appear to identify as important. For instance, our optimized sequences contained increased δ power in over the frontal lobes and increased α power over the parietal lobes during the Count task relative to the original unoptimized input segment. We also observed a considerable increase in γ power across all tasks, but especially for the Count task. This increase in γ power in our optimized input sequences provides further evidence, in addition to our analysis of FIR filters, that our CNNs place more importance on γ-band activity than our baseline approaches. This may mean that the increased sophistication of these networks and their ability to model nonlinear spatiotemporal patterns are required in order to successfully utilize γ-band activity. Of course, these observations are certainly limited by the fact that they are only for a single subject and also by the fact that these inputs are artificially generated. Although it is possible that these artificially generated input sequences do not fully generalize to the optimal patterns in actual EEG signals, these results are generally consistent with our observations about PSDs and our other analyses experiments. Overall, it does appear that this approach for finding optimal input sequences may be a valuable tool for analyzing the patterns that our networks tend to leverage and for identifying patterns that are important for discriminating between mental tasks.

We have also examined the layerwise activations produced by this network when processing our artificially generated input sequences. High-frequency information in the γ band is extremely pronounced in these activations, especially for the Count and Rotate tasks in the first two layers. Again, this provides supporting evidence that our networks have learned to utilize high-frequency information for identifying these tasks. This analysis of layerwise activations also affirms the conclusions drawn from our examination of the activations produced by our networks when processing actual EEG segments. Namely, we see that the first layer acts primarily as a linear filter with subsequent layers having increasingly nonlinear behavior and spe-

cialization for each task and a gradual migration from shorter to longer-term time scales. These effects are, however, more pronounced for our optimized input segments, which demonstrates that these networks do, in fact, appear to be forming hierarchical, multiscale representations of these EEG signals.

In this research, we have laid out a strong case for using multilayer convolutional neural networks for classifying EEG signals in asynchronous mental-task BCIs. We have shown that single-axis convolutions across time combined with small networks, careful regularization and label aggregation readout layers yields networks that outperform PSD-based approaches while also providing a great deal of flexibility with respect to BCI application design. We have also shown that multilayer networks combined with transfer learning offer a considerable performance improvement over single-layer networks and that these networks are able to learn hierarchical, multiscale representations of EEG signals. These networks are also able to utilize some types of patterns, including high frequency information, better than PSD-based approaches. Although other research groups have, concurrently, begun to explore the use of CNNs for classifying EEG signals, we are the first to provide an in-depth exploration of using CNNs with label aggregation layers and, to our knowledge, we are the first to combine CNNs with transfer learning for classifying EEG signals in asynchronous BCIs. We have also provided several novel approaches for analyzing the patterns that these networks learn to leverage, including analyzing the weights of these networks as FIR filters and using optimization techniques to generate artificial input sequences. Clearly, deep artificial neural networks have an important place in the future of asynchronous BCIs and in the analysis of EEG signals.

5.2 Looking Forward

Despite the progress that we have made in this research, there are many remaining open questions and engineering problems that must be solved in order to construct effective mental-task BCIs and to further our understanding of EEG signals, the human brain and artificial neural networks. We believe that the first step toward these goals should be to examine the perfor-

mance of these classifiers in an interactive, real-time setting and over a prolonged period of time. All of the components necessary to perform such experiments have been implemented in CEBL$_3$ and our research can easily be extended to examine how these methods perform in an interactive setting. It is also important to determine the extent to which these methods are valuable for use in practical assistive technologies. We believe that it is essential to focus on real-world scenarios and people with disabilities and their actual needs and use cases. Along these lines, further improvements must be made to user interfaces and BCI applications so that real-world devices can be easily controlled using these technologies.

We have also noted that, at least in our dataset, the Count task tends to perform quite well while the Fist task performs somewhat poorly. Although Friedrich, et al., have experimented with a variety different mental tasks [36, 56], such research has not been conducted in combination with CNNs or in long-term interactive studies. It is possible that there may be considerable mutual adaptation, between both BCI users and machine learning models, when users are allowed to choose the tasks that they perform and adapt the way they perform those tasks over a long period of interactive use.

It is also important to improve reliability and accuracy in order for these methods to be useful in many practical BCI applications. Intelligent evidence accumulation strategies that prevent the system from taking an action until it is highly confident in the user's intent appear to have considerable potential for improving reliability, especially when combined with CNNs. Applying transfer learning to extremely large EEG datasets might also improve accuracy. The use of large datasets along with multilayer networks and transfer learning has recently played an important role in improving the accuracy of machine learning models in a number of domains [101–103]. When combined with increasingly powerful hardware vector processors, very large EEG datasets may be extremely valuable for improving the accuracy of CNN-based classifiers.

This work has also created several new questions about the importance of various types of patterns found in EEG signals. In particular, we have noted that our CNNs tend to place a higher level of importance on high frequency information than PSD-based approaches. We have also

observed that there tends to be relatively high power in the γ band during the Count and Song tasks. We have also provided evidence that nonlinear patterns and phase relationships are valuable for classification. Further experimentation should be conducted in order to establish the importance of high-frequency information, phase relationships and nonlinear patterns in EEG signals. There are many things that we still do not know about EEG signals and it is important that we continue to leverage methods from both data science and neuroscience in order to discover and investigate new patterns found in these signals.

There have also been recent advances in constructing artificial inputs that may be useful for both improving classification performance and also for analyzing the patterns leveraged by a network, similar to the approach that we have proposed using ALOPEX. Specifically, Generative Adversarial Networks (GANs) have become increasingly popular as a technique for generating artificial inputs for deep networks [144]. Future research should examine the potential uses of GANs in EEG signal classification and analysis.

Finally, it is possible that the convolutional units in these networks might be replaced with recurrent units, i.e., artificial neurons that use feedback connections instead of convolutional kernels to achieve memory [60–62]. While we have shown that convolutions can be viewed as finite impulse-response filters, recurrent networks can be viewed as infinite impulse-response filters. Such a configuration may have greatly increased memory capacity without requiring more free parameters. It is also valuable to note that in the field of signal processing, infinite impulse-response filters typically have better filtering characteristics with fewer parameters than finite impulse-response filters.

With these future research topics laid out, and some of this work already underway, it appears that there will be a number of innovations to come with respect to the application of deep artificial neural networks to BCIs and EEG signal analysis. It seems extremely promising that we will soon be able to construct asynchronous BCIs that are able to harness the power of the brain to control human-computer interfaces and practical assistive technologies.

Bibliography

[1] Luis Nicolas-Alonso and Jaime Gomez-Gil. Brain computer interfaces, a review. *Sensors*, 12(2):1211–1279, 2012.

[2] Jonathan Wolpaw and Elizabeth Winter Wolpaw. *Brain-computer interfaces: principles and practice*. Oxfort University Press, 2012.

[3] Guido Dornhege. *Toward brain-computer interfacing*. The MIT Press, 2007.

[4] A Ramos Murguialday, J Hill, M Bensch, S Martens, S Halder, F Nijboer, Bernhard Schoelkopf, N Birbaumer, and A Gharabaghi. Transition from the locked in to the completely locked-in state: a physiological analysis. *Clinical Neurophysiology*, 122(5):925–933, 2011.

[5] Fred Plum and Jerome B Posner. *The Diagnosis Of Stupor & Coma*, volume 19. Oxford University Press, USA, 1982.

[6] Jean-Dominique Bauby. *The diving bell and the butterfly: A memoir of life in death*. Vintage, 1998.

[7] Eric Sellers, Theresa Vaughan, and Jonathan Wolpaw. A brain-computer interface for long-term independent home use. *Amyotrophic lateral sclerosis*, 11(5):449–455, 2010.

[8] Adrian M Owen and Martin R Coleman. Detecting awareness in the vegetative state. *Annals of the New York Academy of Sciences*, 1129(1):130–138, 2008.

[9] Fiachra Matthews, Barak A Pearlmutter, Tomas E Ward, Christopher Soraghan, and Charles Markham. Hemodynamics for brain-computer interfaces. *IEEE Signal Processing Magazine*, 25(1):87–94, 2008.

[10] Eric C Leuthardt, Kai J Miller, Gerwin Schalk, Rajesh PN Rao, and Jeffrey G Ojemann. Electrocorticography-based brain computer interface-the seattle experience. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 14(2):194–198, 2006.

[11] Leigh R Hochberg, Daniel Bacher, Beata Jarosiewicz, Nicolas Y Masse, John D Simeral, Joern Vogel, Sami Haddadin, Jie Liu, Sydney S Cash, Patrick van der Smagt, et al. Reach and grasp by people with tetraplegia using a neurally controlled robotic arm. *Nature*, 485(7398):372–375, 2012.

[12] Paul Nunez and Ramesh Srinivasan. *Electric fields of the brain: the neurophysics of EEG*. Oxford University Press, USA, 2006.

[13] Elliott Forney, Charles Anderson, Patricia Davies, William Gavin, Brittany Taylor, and Marla Roll. A comparison of EEG systems for use in P300 spellers by users with motor impairments in real-world environments. In *Proceedings of the Fifth International Brain-Computer Interface Meeting: Defining the Future*. Graz University of Technology Publishing House, 2013.

[14] Elliott Forney, Charles Anderson, and Colorado State University Brain-Computer Interfaces Laboratory. Colorado EEG and BCI laboratory. https://www.cs.colostate.edu/eeg/main/software/cebl3, Jul 2018.

[15] Center for Adaptive Neurotechnologies. BCI 2000. http://www.schalklab.org/research/bci2000, August 2017.

[16] Swartz Center for Computational Neuroscience. BCILAB. http://sccn.ucsd.edu/wiki/BCILAB, August 2017.

[17] French Institute for Research in Computer Science and Automation. OpenVibe. http://www.inria.fr/, August 2017.

[18] g.tec medical engineering GmbH. intendiX. http://www.intendix.com/, August 2017.

[19] g.tec medical engineering GmbH. mindBEAGLE. http://www.mindbeagle.com/, August 2017.

[20] Joseph Mak, Dennis McFarland, Theresa Vaughan, Lynn McCane, Phillippa Tsui, Debra Zeitlin, Eric Sellers, and Jonathan Wolpaw. EEG correlates of P300-based brain–computer interface (BCI) performance in people with amyotrophic lateral sclerosis. *Journal of neural engineering*, 9(2):026014, 2012.

[21] Theresa M Vaughan, Dennis J McFarland, Gerwin Schalk, William A Sarnacki, Dean J Krusienski, Eric W Sellers, and Jonathan R Wolpaw. The wadsworth BCI research and development program: at home with BCI. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 14(2):229–233, 2006.

[22] F. Pichiorri, G. Morone, M. Petti, M. Molinari, L. Astolfi, F. Cincotti, and D. Mattia. BCI for stroke rehabilitation: a randomized controlled trial of efficacy. In *Proceedings of the Fifth International Brain-Computer Interface Meeting: Defining the Future*. Graz University of Technology Publishing House, 2013.

[23] Dorothée Lulé, Quentin Noirhomme, Sonja C Kleih, Camille Chatelle, Sebastian Halder, Athena Demertzi, Marie-Aurélie Bruno, Olivia Gosseries, Audrey Vanhaudenhuyse, Caroline Schnakers, et al. Probing command following in patients with disorders of consciousness using a brain–computer interface. *Clinical Neurophysiology*, 124(1):101–106, 2013.

[24] Lawrence Ashley Farwell and Emanuel Donchin. Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalography and clinical Neurophysiology*, 70(6):510–523, 1988.

[25] Benjamin Blankertz, Steven Lemm, Matthias Treder, Stefan Haufe, and Klaus-Robert Müller. Single-trial analysis and classification of erp components–a tutorial. *Neuroimage*, 56(2):814–825, 2011.

[26] P Brunner, S Joshi, S Briskin, JR Wolpaw, H Bischof, and G Schalk. Does the "P300" speller depend on eye gaze? *Journal of neural engineering*, 7(5):056013, 2010.

[27] Gert Pfurtscheller. Event-related synchronization (ers): an electrophysiological correlate of cortical areas at rest. *Electroencephalography and clinical neurophysiology*, 83(1):62–69, 1992.

[28] Jonathan R Wolpaw and Dennis J McFarland. Multichannel EEG-based brain-computer communication. *Electroencephalography and clinical Neurophysiology*, 90(6):444–449, 1994.

[29] Dennis J McFarland, Laurie A Miner, Theresa M Vaughan, and Jonathan R Wolpaw. Mu and beta rhythm topographies during motor imagery and actual movements. *Brain topography*, 12(3):177–186, 2000.

[30] Gert Pfurtscheller and Christa Neuper. Motor imagery and direct brain-computer communication. *Proceedings of the IEEE*, 89(7):1123–1134, 2001.

[31] Christa Neuper, Gernot Müller, Andrea Kübler, Niels Birbaumer, and Gert Pfurtscheller. Clinical application of an EEG-based brain–computer interface: a case study in a patient with severe motor impairment. *Clinical neurophysiology*, 114(3):399–409, 2003.

[32] Andrea Kübler, Femke Nijboer, Jürgen Mellinger, Theresa M Vaughan, Hannelore Pawelzik, Gerwin Schalk, Dennis J McFarland, Niels Birbaumer, and Jonathan R Wolpaw. Patients with ALS can use sensorimotor rhythms to operate a brain-computer interface. *Neurology*, 64(10):1775–1777, 2005.

[33] Zachary Keirn and Jorge Aunon. A new mode of communication between man and his surroundings. *IEEE Transactions on Biomedical Engineering*, 37(12):1209–1214, 1990.

[34] José Millán, Josep Mouriño, Marco Franzé, Febo Cincotti, Markus Varsta, Jukka Heikkonen, and Fabio Babiloni. A local neural classifier for the recognition of EEG patterns associated to mental tasks. *IEEE Transactions on Neural Networks*, 13(3):678–686, 2002.

[35] José Millán, Pierre Ferrez, Ferran Galán, Eileen Lew, and Ricardo Chavarriaga. Non-invasive brain-machine interaction. *International Journal of Pattern Recognition and Artificial Intelligence*, 22(05):959–972, 2008.

[36] Elisabeth Friedrich, Reinhold Scherer, and Christa Neuper. The effect of distinct mental strategies on classification performance for brain–computer interfaces. *International Journal of Psychophysiology*, 84(1):86–94, 2012.

[37] Elisabeth Friedrich, Reinhold Scherer, and Christa Neuper. Do user-related factors of motor impaired and able-bodied participants correlate with classification accuracy? In *Procceedings of the 5th International Brain-Computer Interface Conference*, pages 156–159. Graz University of Technology, 2011.

[38] Alexander Ya Kaplan, Andrew A Fingelkurts, Alexander A Fingelkurts, Sergei V Borisov, and Boris S Darkhovsky. Nonstationary nature of the brain activity as revealed by EEG/MEG: methodological, practical and conceptual challenges. *Signal processing*, 85(11):2190–2212, 2005.

[39] Cornelis J Stam. Nonlinear dynamical analysis of EEG and MEG: review of an emerging field. *Clinical Neurophysiology*, 116(10):2266–2301, 2005.

[40] CJ Stam, TCAM Van Woerkom, and WS Pritchard. Use of non-linear EEG measures to characterize EEG changes during mental activity. *Electroencephalography and clinical Neurophysiology*, 99(3):214–224, 1996.

[41] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[42] Yann LeCun and Yoshua Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[43] Yann LeCun, Koray Kavukcuoglu, Clément Farabet, et al. Convolutional networks and applications in vision. In *ISCAS*, pages 253–256, 2010.

[44] Yann LeCun, Patrick Haffner, Léon Bottou, and Yoshua Bengio. Object recognition with gradient-based learning. *Shape, contour and grouping in computer vision*, pages 823–823, 1999.

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

[46] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1915–1929, 2013.

[47] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[48] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3642–3649. IEEE, 2012.

[49] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[50] Hubert Cecotti and Axel Graeser. Convolutional neural network with embedded fourier transform for EEG classification. In *Proceedings of the International Conference on Pattern Recognition (ICPR)*, pages 1–4. IEEE, 2008.

[51] Hubert Cecotti and Axel Graser. Convolutional neural networks for P300 detection with application to brain-computer interfaces. *IEEE transactions on pattern analysis and machine intelligence*, 33(3):433–445, 2011.

[52] Vernon J Lawhern, Amelia J Solon, Nicholas R Waytowich, Stephen M Gordon, Chou P Hung, and Brent J Lance. EEGNet: a compact convolutional neural network for EEG-based brain–computer interfaces. *Journal of neural engineering*, 15(5):056013, 2018.

[53] Robin Tibor Schirrmeister, Jost Tobias Springenberg, Lukas Dominique Josef Fiederer, Martin Glasstetter, Katharina Eggensperger, Michael Tangermann, Frank Hutter, Wolfram Burgard, and Tonio Ball. Deep learning with convolutional neural networks for EEG decoding and visualization. *Human Brain Mapping*, 2017.

[54] José Millán, Frédéric Renkens, Josep Mouriño, and Wulfram Gerstner. Brain-actuated interaction. *Artificial Intelligence*, 159(1):241–259, 2004.

[55] Li Zhiwei and Shen Minfen. Classification of mental task EEG signals using wavelet packet entropy and SVM. In *8th International Conference on Electronic Measurement and Instruments*, pages 3–906. IEEE, 2007.

[56] Elisabeth Friedrich, Reinhold Scherer, and Christa Neuper. Long-term evaluation of a 4-class imagery-based brain–computer interface. *Clinical Neurophysiology*, 2013.

[57] Charles Anderson, Erik Stolz, and Sanyogita Shamsunder. Discriminating mental tasks using EEG represented by AR models. In *17th Annual Conference of The IEEE Engineering in Medicine and Biology Society*, volume 2, pages 875–876. IEEE, 1995.

[58] Charles Anderson, Erik Stolz, and Sanyogita Shamsunder. Multivariate autoregressive models for classification of spontaneous electroencephalographic signals during mental tasks. *IEEE Transactions on Biomedical Engineering*, 45(3):277–286, 1998.

[59] Damien Coyle, Girijesh Prasad, and Thomas McGinnity. A time-series prediction approach for feature extraction in a brain-computer interface. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 13(4):461–467, 2005.

[60] Elliott M. Forney, Charles W. Anderson, William J. Gavin, Patricia L. Davies, Marla C. Roll, and Brittany K. Taylor. Echo state networks for modeling and classification of EEG signals

in mental-task brain computer interfaces. Technical report, Colorado State University, Department of Computer Science, November 2015.

[61] Elliott Forney and Charles Anderson. Classification of EEG during imagined mental tasks by forecasting with elman recurrent neural networks. *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pages 2749–2755, 2011.

[62] Elliott Forney. Electroencephalogram classification by forecasting with recurrent neural networks. Master's thesis, Department of Computer Science, Colorado State University, Fort Collins, CO, 2011.

[63] Charles Anderson, James Knight, Tim O'Connor, Michael Kirby, and Artem Sokolov. Geometric subspace methods and time-delay embedding for EEG artifact removal and classification. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 14(2):142–146, 2006.

[64] Charles Anderson, James Knight, Michael Kirby, and Douglas Hundley. Classification of time-embedded EEG using short-time principal component analysis. *Toward Brain-Computer Interfacing*, pages 261–278, 2007.

[65] Charles Anderson and Jeshua Bratman. Translating thoughts into actions by finding patterns in brainwaves. In *Proceedings of the Fourteenth Yale Workshop on Adaptive and Learning Systems*, pages 1–6, 2008.

[66] Charles Anderson, Elliott Forney, Douglas Hains, and Annamalai Natarajan. Reliable identification of mental tasks using time-embedded EEG and sequential evidence accumulation. *Journal of Neural Engineering*, 8(2):025023, 2011.

[67] Deon Garrett, David A Peterson, Charles W Anderson, and Michael H Thaut. Comparison of linear, nonlinear, and feature selection methods for EEG signal classification. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 11(2):141–144, 2003.

[68] Klaus-Robert Müller, Charles W Anderson, and Gary E Birch. Linear and nonlinear methods for brain-computer interfaces. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 11(2):165–169, 2003.

[69] Ferran Galán, Marnix Nuttin, Eileen Lew, Pierre Ferrez, Gerolf Vanacker, Johan Philips, and José Millán. A brain-actuated wheelchair: Asynchronous and non-invasive brain-computer interfaces for continuous control of robots. *Clinical Neurophysiology*, 119(9):2159–2169, 2008.

[70] Jaime F Delgado Saa and Miguel Sotaquirá Gutierrez. EEG signal classification using power spectral features and linear discriminant analysis: A brain computer interface application. In *Eighth Latin American and Caribbean Conference for Engineering and Technology*, pages 1–7, 2010.

[71] G. Heinzel, A. Rüdiger, R. Schilling, and T. Hannover. Spectrum and spectral density estimation by the discrete fourier transform (dft), including a comprehensive list of window functions and some new flat-top windows. Technical report, Max Plank Institute, 2002.

[72] Norden E Huang, Zheng Shen, Steven R Long, Manli C Wu, Hsing H Shih, Quanan Zheng, Nai-Chyuan Yen, Chi Chao Tung, and Henry H Liu. The empirical mode decomposition and the hilbert spectrum for nonlinear and non-stationary time series analysis. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 454, pages 903–995. The Royal Society, 1998.

[73] Elly Gysels and Patrick Celka. Phase synchronization for the recognition of mental tasks in a brain-computer interface. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 12(4):406–415, 2004.

[74] Johannes Müller-Gerking, Gert Pfurtscheller, and Henrik Flyvbjerg. Designing optimal spatial filters for single-trial EEG classification in a movement task. *Clinical neurophysiology*, 110(5):787–798, 1999.

[75] Herbert Ramoser, Johannes Muller-Gerking, and Gert Pfurtscheller. Optimal spatial filtering of single trial EEG during imagined hand movement. *IEEE Transactions on Rehabilitation Engineering*, 8(4):441–446, 2000.

[76] Benjamin Blankertz, Ryota Tomioka, Steven Lemm, Motoaki Kawanabe, and K-R Muller. Optimizing spatial filters for robust EEG single-trial analysis. *IEEE Signal processing magazine*, 25(1):41–56, 2008.

[77] James N. Knight. Signal fraction analysis and artifact removal in EEG. Master's thesis, Department of Computer Science, Colorado State University, Fort Collins, CO, 2003.

[78] Moritz Grosse-Wentrup and Martin Buss. Multiclass common spatial patterns and information theoretic feature extraction. *IEEE transactions on Biomedical Engineering*, 55(8):1991–2000, 2008.

[79] Simon Haykin. *Neural networks and learning machines*, volume 3. Pearson Upper Saddle River, NJ, USA, 2009.

[80] Christopher M Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.

[81] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[82] Yoshua Bengio et al. Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[83] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[84] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660, 2010.

[85] Martin Fodslette Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural networks*, 6(4):525–533, 1993.

[86] Y. LeCun, L. Bottou, G. Orr, and K. Müller. Efficient backprop. In G. Orr and Muller K., editors, *Neural Networks: Tricks of the trade*, page 546. Springer, 1998.

[87] Rajat Raina, Anand Madhavan, and Andrew Y Ng. Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th annual international conference on machine learning (ICML)*, pages 873–880. ACM, 2009.

[88] James Martens. Deep learning via hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, pages 735–742, 2010.

[89] Jiquan Ngiam, Adam Coates, Ahbik Lahiri, Bobby Prochnow, Quoc V Le, and Andrew Y Ng. On optimization methods for deep learning. In *Proceedings of the 28th international conference on machine learning (ICML)*, pages 265–272, 2011.

[90] James Martens and Ilya Sutskever. Training deep and recurrent networks with hessian-free optimization. In *Neural networks: Tricks of the trade*, pages 479–535. Springer, 2012.

[91] Kunihiko Fukushima and Sei Miyake. Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern recognition*, 15(6):455–469, 1982.

[92] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[93] Michael C Mozer. *The perception of multiple objects: A connectionist approach*. The MIT Press, 1991.

[94] David H Hubel and Torsten N Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology*, 148(3):574–591, 1959.

[95] David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154, 1962.

[96] Marcello G.P. Rosa. Visual cortex. In V.S. Ramachandran, editor, *Encyclopedia of the Human Brain*, pages 753–773. Academic Press, New York, 2002.

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

[98] Vinay Jayaram, Morteza Alamgir, Yasemin Altun, Bernhard Scholkopf, and Moritz Grosse-Wentrup. Transfer learning in brain-computer interfaces. *IEEE Computational Intelligence Magazine*, 11(1):20–31, 2016.

[99] Sinno Jialin Pan, Qiang Yang, et al. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[100] Marco Congedo, Alexandre Barachant, and Anton Andreev. A new generation of brain-computer interface based on riemannian geometry. *arXiv preprint arXiv:1310.8115*, 2013.

[101] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1717–1724, 2014.

[102] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[103] 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.

[104] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[105] NumPy. NumPy. http://www.numpy.org, February 2019.

[106] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[107] Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, Alexander Belopolsky, Yoshua Bengio, Arnaud Bergeron, James Bergstra, Valentin Bisson, Josh Bleecher Snyder, Nicolas Bouchard, Nicolas Boulanger-Lewandowski, Xavier Bouthillier, Alexandre de Brébisson, Olivier Breuleux, Pierre-Luc Carrier, Kyunghyun Cho, Jan Chorowski, Paul Christiano, Tim Cooijmans, Marc-Alexandre Côté, Myriam Côté, Aaron Courville, Yann N. Dauphin, Olivier Delalleau, Julien Demouth, Guillaume Desjardins, Sander Dieleman,

Laurent Dinh, Mélanie Ducoffe, Vincent Dumoulin, Samira Ebrahimi Kahou, Dumitru Erhan, Ziye Fan, Orhan Firat, Mathieu Germain, Xavier Glorot, Ian Goodfellow, Matt Graham, Caglar Gulcehre, Philippe Hamel, Iban Harlouchet, Jean-Philippe Heng, Balázs Hidasi, Sina Honari, Arjun Jain, Sébastien Jean, Kai Jia, Mikhail Korobov, Vivek Kulkarni, Alex Lamb, Pascal Lamblin, Eric Larsen, César Laurent, Sean Lee, Simon Lefrancois, Simon Lemieux, Nicholas Léonard, Zhouhan Lin, Jesse A. Livezey, Cory Lorenz, Jeremiah Lowin, Qianli Ma, Pierre-Antoine Manzagol, Olivier Mastropietro, Robert T. McGibbon, Roland Memisevic, Bart van Merriënboer, Vincent Michalski, Mehdi Mirza, Alberto Orlandi, Christopher Pal, Razvan Pascanu, Mohammad Pezeshki, Colin Raffel, Daniel Renshaw, Matthew Rocklin, Adriana Romero, Markus Roth, Peter Sadowski, John Salvatier, François Savard, Jan Schlüter, John Schulman, Gabriel Schwartz, Iulian Vlad Serban, Dmitriy Serdyuk, Samira Shabanian, Étienne Simon, Sigurd Spieckermann, S. Ramana Subramanyam, Jakub Sygnowski, Jérémie Tanguay, Gijs van Tulder, Joseph Turian, Sebastian Urban, Pascal Vincent, Francesco Visin, Harm de Vries, David Warde-Farley, Dustin J. Webb, Matthew Willson, Kelvin Xu, Lijun Xue, Li Yao, Saizheng Zhang, and Ying Zhang. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[108] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017.

[109] Carl W Baum and Venugopal V Veeravalli. A sequential procedure for multihypothesis testing. *IEEE Transactions on Information Theory*, 40(6), 1994.

[110] Sanjit K Mitra and James F Kaiser. *Handbook for digital signal processing*. John Wiley & Sons, Inc., 1993.

[111] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems (NIPS)*, pages 379–387, 2016.

[112] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

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

[114] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 1–9, 2015.

[115] KP Unnikrishnan and Kootala P Venugopal. Alopex: A correlation-based learning algorithm for feedforward and recurrent neural networks. *Neural Computation*, 6(3):469–490, 1994.

[116] E Harth and E Tzanakou. Alopex: A stochastic method for determining visual receptive fields. *Vision Research*, 14(12):1475–1482, 1974.

[117] Evangelia Tzanakou, R Michalak, and Erich Harth. The alopex process: Visual receptive fields by response feedback. *Biological Cybernetics*, 35(3):161–174, 1979.

[118] KP Venugopal and AS Pandya. Alopex algorithm for training multilayer neural networks. In *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN)*, pages 196–201. IEEE, 1991.

[119] Alejandro Bia. Alopex-b: A new, simpler, but yet faster version of the alopex training algorithm. *International Journal of Neural Systems*, 11(06):497–507, 2001.

[120] Zhe Chen, Simon Haykin, Jos J Eggermont, and Suzanna Becker. *Correlative learning: a basis for brain and adaptive systems*, volume 49. John Wiley & Sons, 2008.

[121] Kevin Jarrett, Koray Kavukcuoglu, Yann LeCun, et al. What is the best multi-stage architecture for object recognition? In *Proceedings of the 12th International Conference on Computer Vision*, pages 2146–2153. IEEE, 2009.

[122] Jonathan Richard Shewchuk et al. An introduction to the conjugate gradient method without the agonizing pain. https://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf, 1994.

[123] Quoc V Le, Jiquan Ngiam, Adam Coates, Abhik Lahiri, Bobby Prochnow, and Andrew Y Ng. On optimization methods for deep learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 265–272. Omnipress, 2011.

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

[125] Elliott M. Forney, Charles W. Anderson, William J. Gavin, Patricia L. Davies, Marla C. Roll, Igor Ryzhkov, and Fereydoon Vafaei. CEBL3: A new software platform for EEG analysis and rapid prototyping of BCI technologies. In *Proceedings of the Sixth International Brain-Computer Interface Meeting*. Graz University of Technology Publishing House, 2016.

[126] Intel math kernel library. https://software.intel.com/en-us/mkl, 2019.

[127] William Gavin and Patricia Davies. Obtaining reliable psychophysiological data with child participants. *Developmental Psychophysiology: Theory, Systems, and Methods. Cambridge University Press, New York, NY*, pages 424–447, 2008.

[128] College of Health and Human Sciences at Colorado State University. Brainwaves research laboratory. http://brainwaves.colostate.edu, Jan 2014.

[129] John Pierce. *An introduction to information theory: symbols, signals & noise*. Dover, 1980.

[130] Jonathan Wolpaw, Herbert Ramoser, Dennis McFarland, and Gert Pfurtscheller. EEG-based communication: improved accuracy by response verification. *IEEE Transactions on Rehabilitation Engineering*, 6(3):326–333, 1998.

[131] Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics bulletin*, 1(6):80–83, 1945.

[132] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2019. ISBN 3–900051–07–0.

[133] Brendan Z Allison and Christa Neuper. Could anyone use a BCI? In *Brain-Computer Interfaces*, pages 35–54. Springer, 2010.

[134] Benjamin Blankertz, Claudia Sannelli, Sebastian Halder, Eva M Hammer, Andrea Kübler, Klaus-Robert Müller, Gabriel Curio, and Thorsten Dickhaus. Neurophysiological predictor of SMR-based BCI performance. *Neuroimage*, 51(4):1303–1309, 2010.

[135] Michael Tangermann, Klaus-Robert Müller, Ad Aertsen, Niels Birbaumer, Christoph Braun, Clemens Brunner, Robert Leeb, Carsten Mehring, Kai J Miller, Gernot Mueller-Putz, et al. Review of the bci competition iv. *Frontiers in neuroscience*, 6:55, 2012.

[136] Bci competition iv, dataset 2a. http://www.bbci.de/competition/iv/#dataset2a, 2019.

[137] John J Foxe and Adam C Snyder. The role of alpha-band brain oscillations as a sensory suppression mechanism during selective attention. *Frontiers in psychology*, 2:154, 2011.

[138] Wolfgang Klimesch. Alpha-band oscillations, attention, and controlled access to stored information. *Trends in cognitive sciences*, 16(12):606–617, 2012.

[139] Keith S Cover, Hugo Vrenken, Jeroen JG Geurts, Bob W van Oosten, Brechtje Jelles, Chris H Polman, Cornelis J Stam, and Bob W van Dijk. Multiple sclerosis patients show a highly

significant decrease in alpha band interhemispheric synchronization measured using MEG. *Neuroimage*, 29(3):783–788, 2006.

[140] Letizia Leocani, Tiziana Locatelli, Vittorio Martinelli, Marco Rovaris, Monica Falautano, Massimo Filippi, Giuseppe Magnani, and Giancarlo Comi. Electroencephalographic coherence analysis in multiple sclerosis: correlation with clinical, neuropsychological, and MRI findings. *Journal of Neurology, Neurosurgery & Psychiatry*, 69(2):192–198, 2000.

[141] Suresh Muthukumaraswamy. High-frequency brain activity and muscle artifacts in MEG/EEG: a review and recommendations. *Frontiers in human neuroscience*, 7:138, 2013.

[142] Judith M Ford, Max Gray, William O Faustman, Theda H Heinks, and Daniel H Mathalon. Reduced gamma-band coherence to distorted feedback during speech when what you say is not what you hear. *International Journal of Psychophysiology*, 57(2):143–150, 2005.

[143] Eraldo Paulesu, Christopher D Frith, and Richard SJ Frackowiak. The neural correlates of the verbal component of working memory. *Nature*, 362(6418):342, 1993.

[144] 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 (NIPS)*, pages 2672–2680, 2014.

Appendix A

Comparisons of Mean Test Classification Accuracies

Table A.1: Results of pairwise Wilcoxon tests for all classification approaches without correcting for multiple comparisons. Each cell shows the p-value resulting from performing a pairwise Wilcoxon test between the method on the left and on the top. Significant results, $p < 0.05$, and shown in bold and borderline results, $p < 0.1$, are underlined.

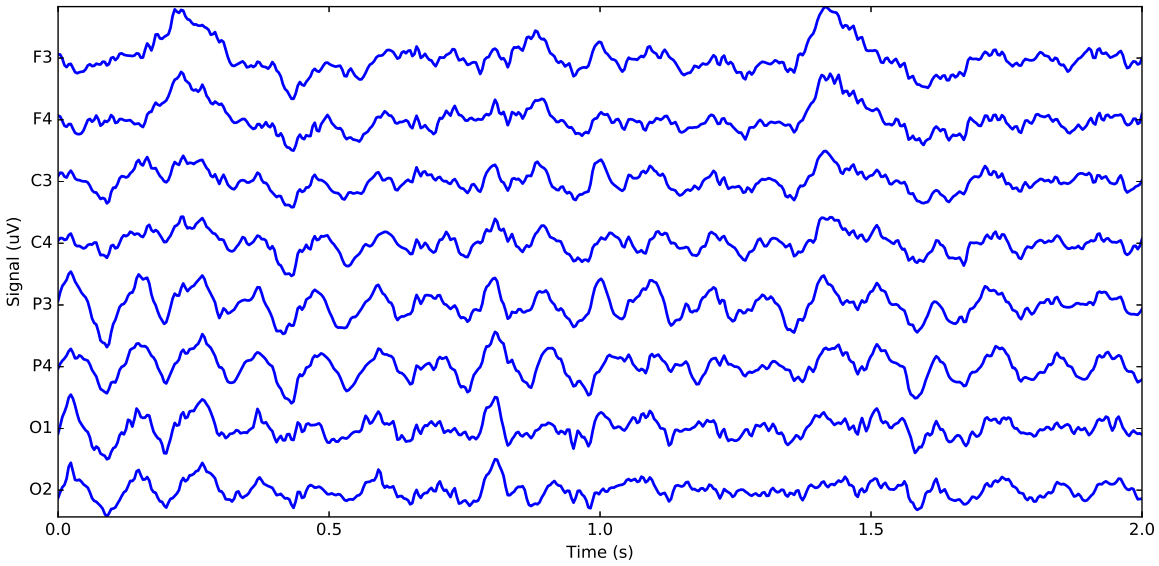
	LDA	QDA	ANN	TDNN	TDNN-TR	CNN-FC	CNN-LA
QDA	0.1997						
ANN	0.3526	0.5048					
TDNN	0.1209	0.0724	**0.0422**				
TDNN-TR	0.0643	**0.0408**	**0.0451**	0.7596			
CNN-FC	**0.0106**	**0.0161**	**0.0117**	**0.0085**	**0.0117**		
CNN-LA	0.0627	0.0677	0.0637	1.0000	1.0000	**0.0060**	
CNN-TR	**0.0432**	**0.0322**	**0.0318**	**0.0330**	0.2293	**0.0070**	0.2838

Table A.2: Results of pairwise Wilcoxon tests for all classification approaches with Holm's correction for multiple comparisons. Each cell shows the p-value resulting from performing a pairwise Wilcoxon test with Holm's correction between the method on the left and on the top. In this case, no results were found to be significant or borderline significant. Note, however, that we have low statistical power given our relatively small sample size and large number of methods to compare.

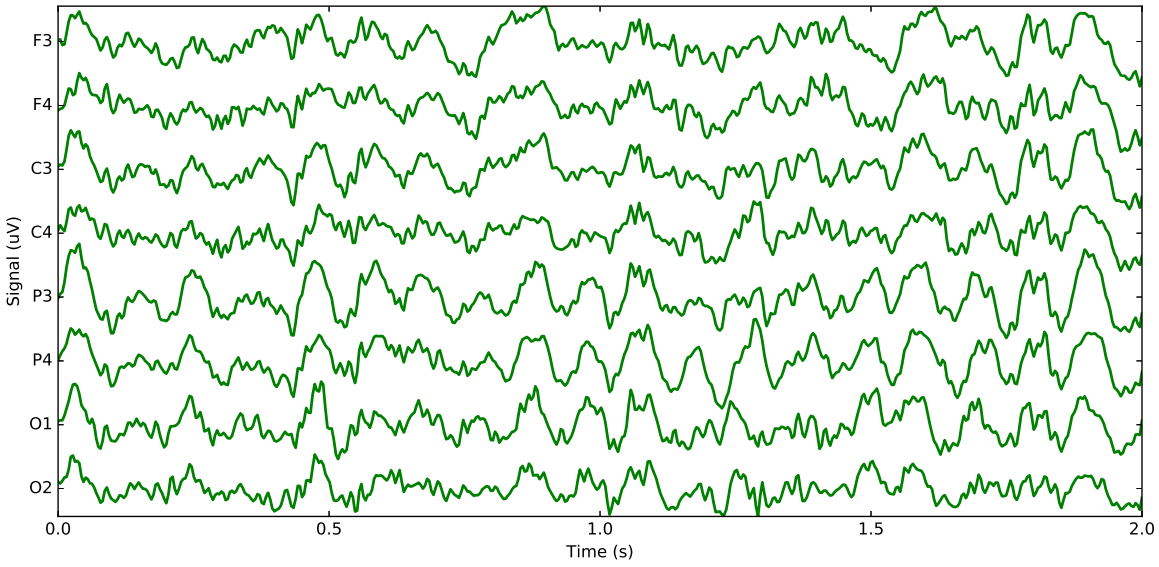
	LDA	QDA	ANN	TDNN	TDNN-TR	CNN-FC	CNN-LA
QDA	1.00						
ANN	1.00	1.00					
TDNN	1.00	0.88	0.74				
TDNN-TR	0.88	0.74	0.74	1.00			
CNN-FC	0.27	0.36	0.28	0.22	0.28		
CNN-LA	0.88	0.88	0.88	1.00	1.00	0.17	
CNN-TR	0.74	0.67	0.67	0.67	1.00	0.19	1.00

Appendix B

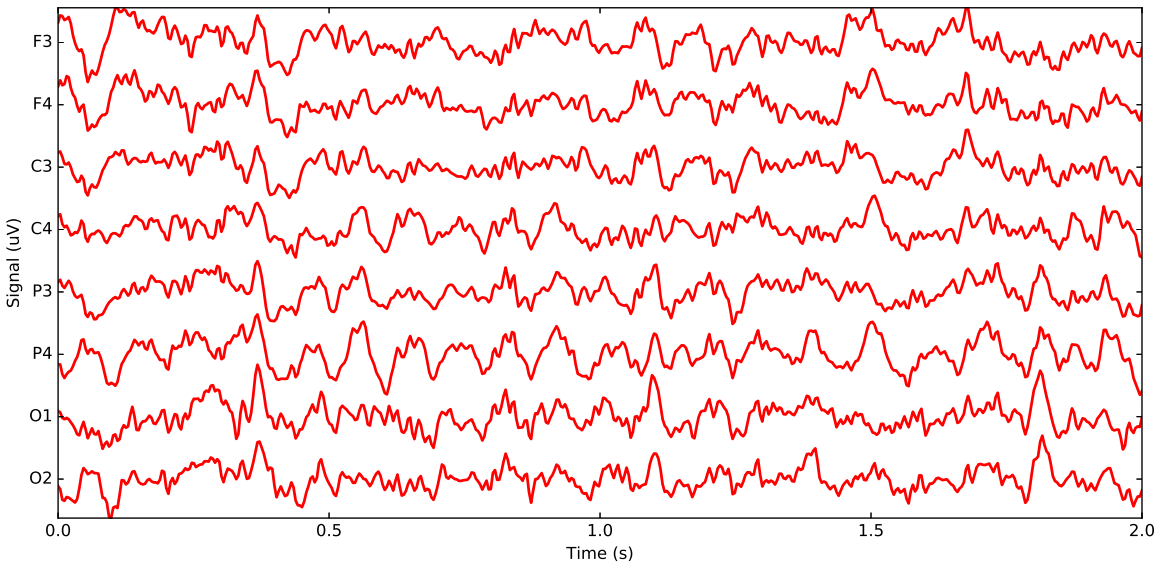
Layerwise Outputs for a Full-Sized CNN-TR.



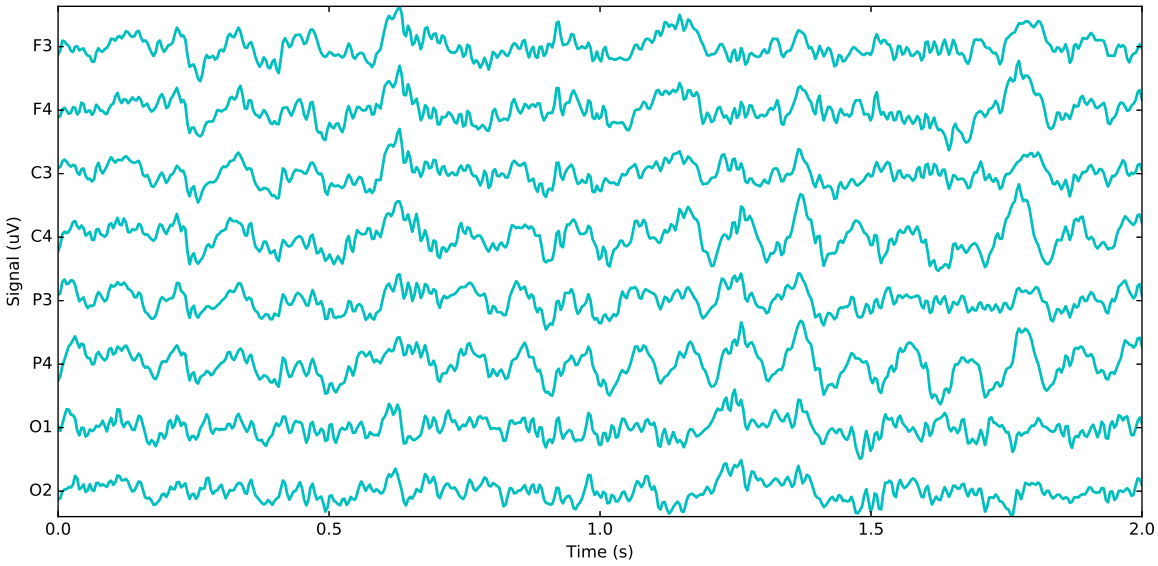
(a) Best performing segment for Count.



(b) Best performing segment for Fist.

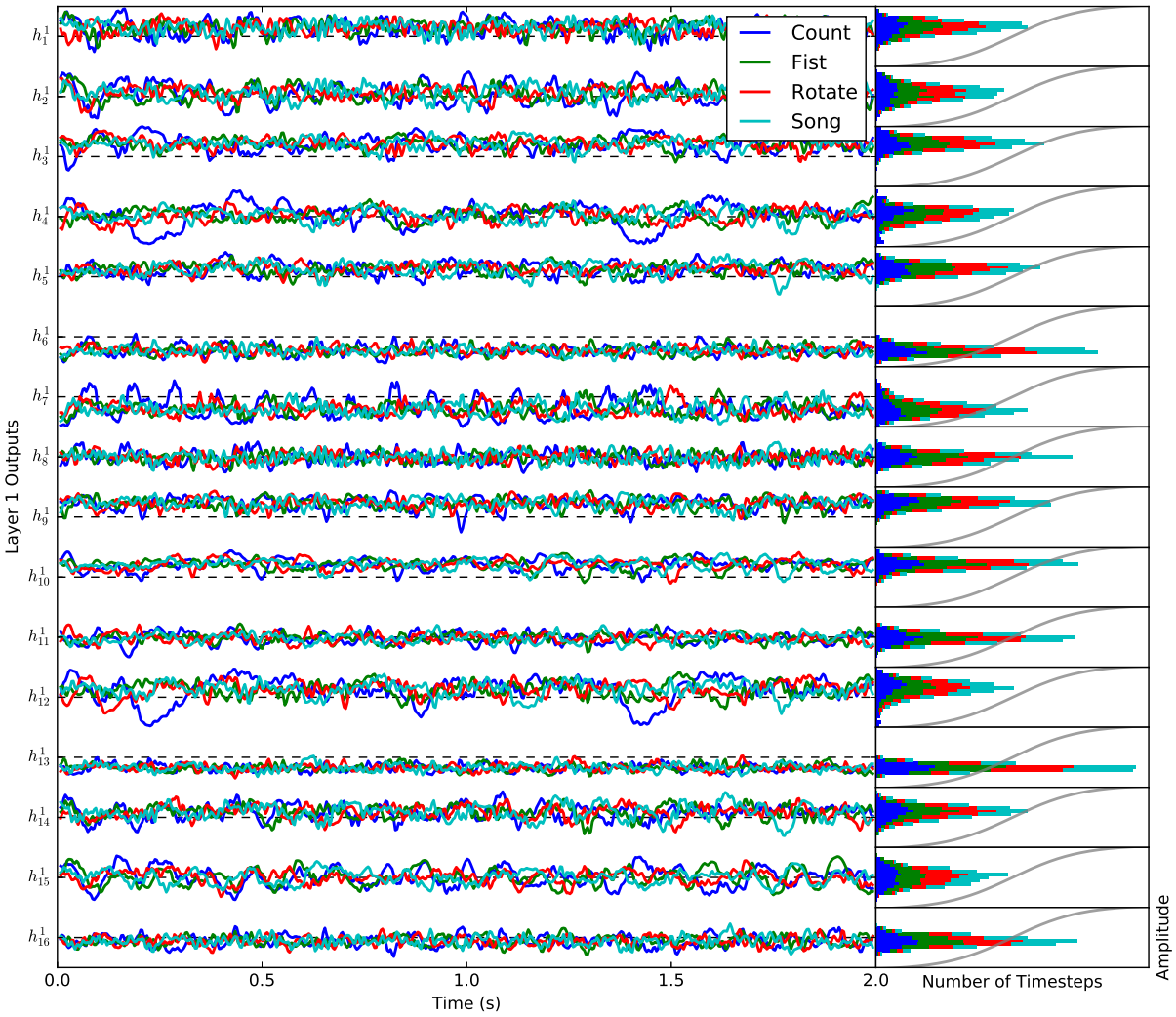


(c) Best performing segment for Rotate.

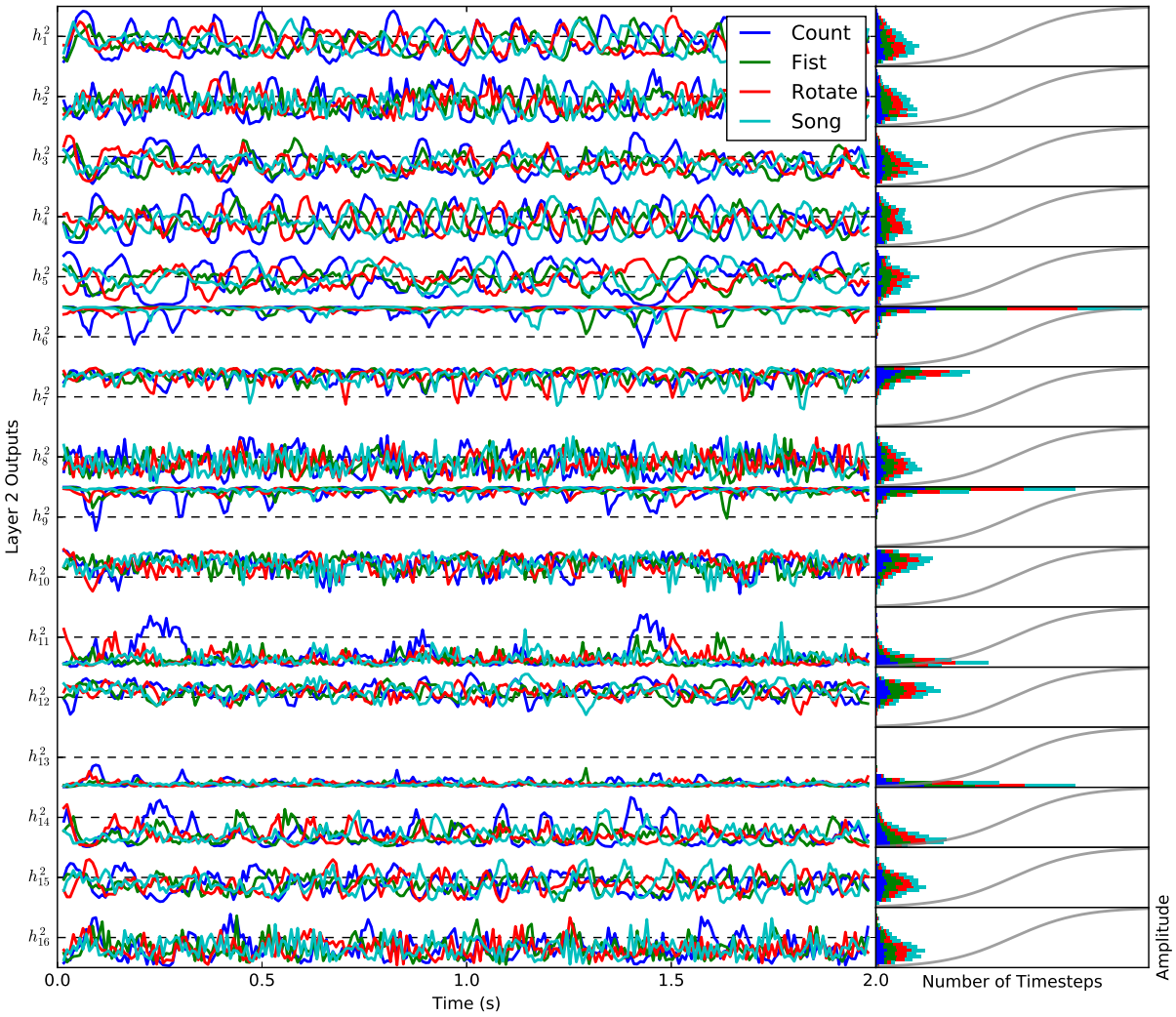


(d) Best performing segment for Song.

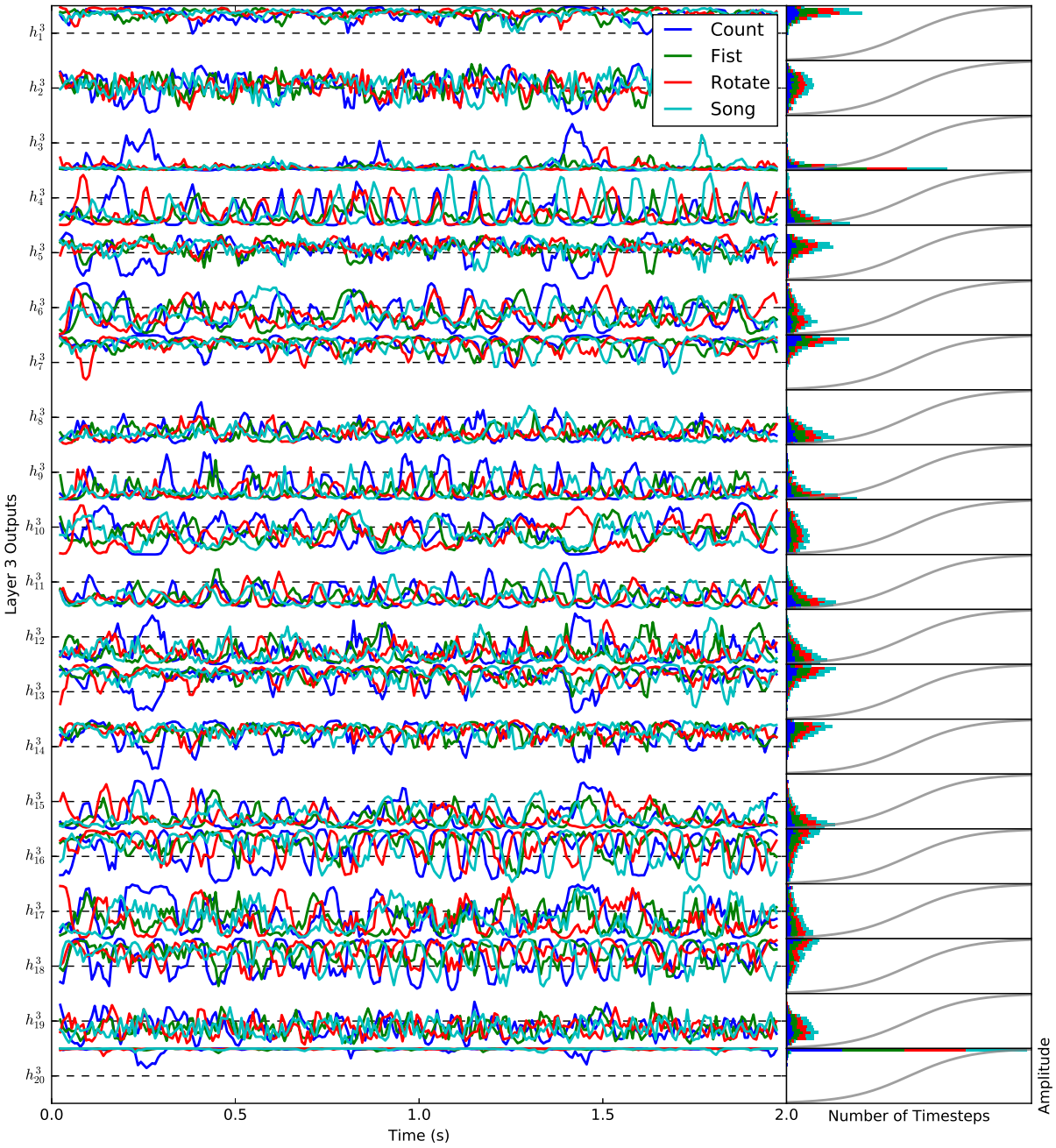
Figure B.1: The best-performing EEG segments, i.e., highest sum of likelihoods, for each class for our full-sized CNN-TR for Subject-1.



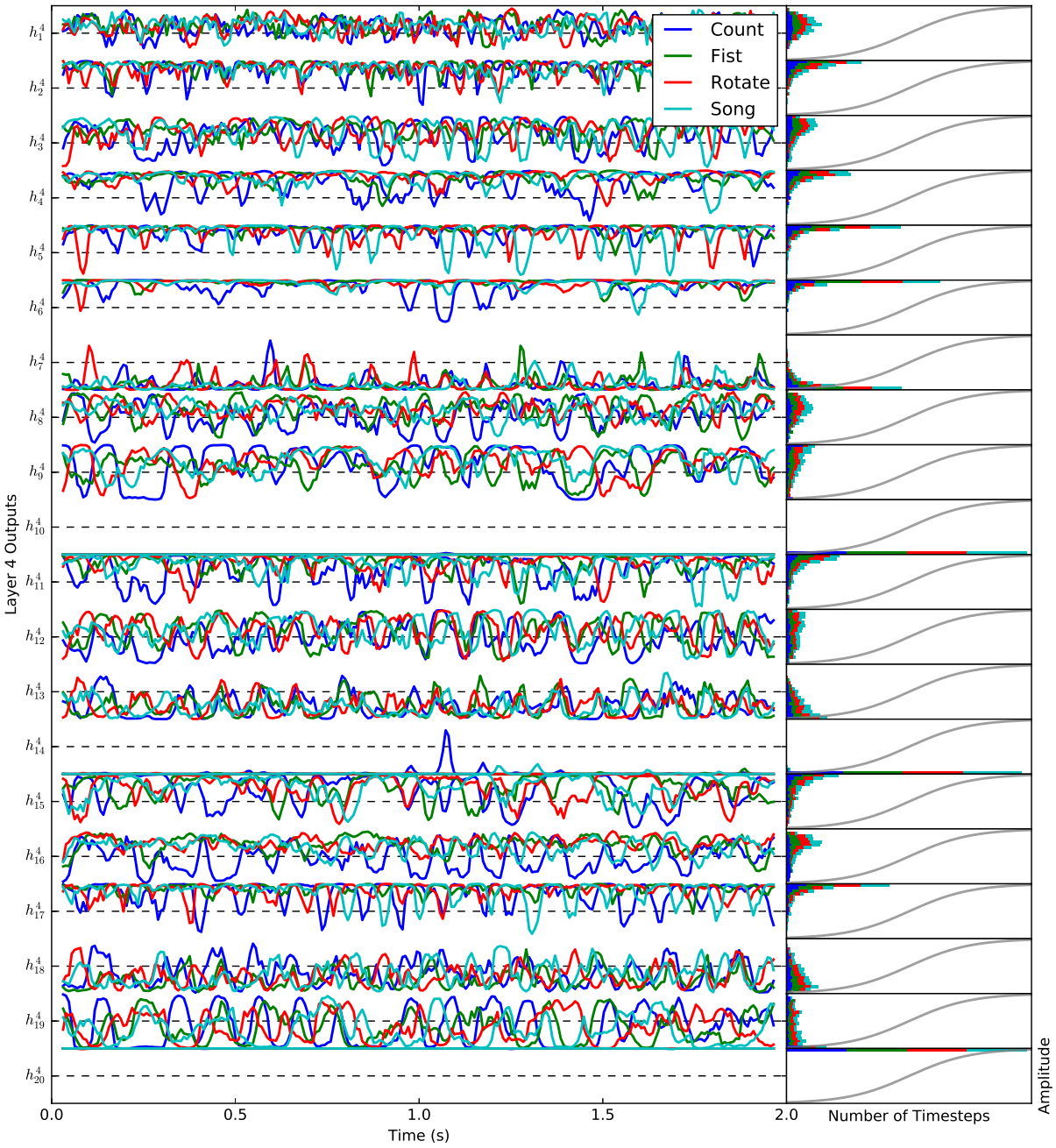
(a) Layer 1 activations.



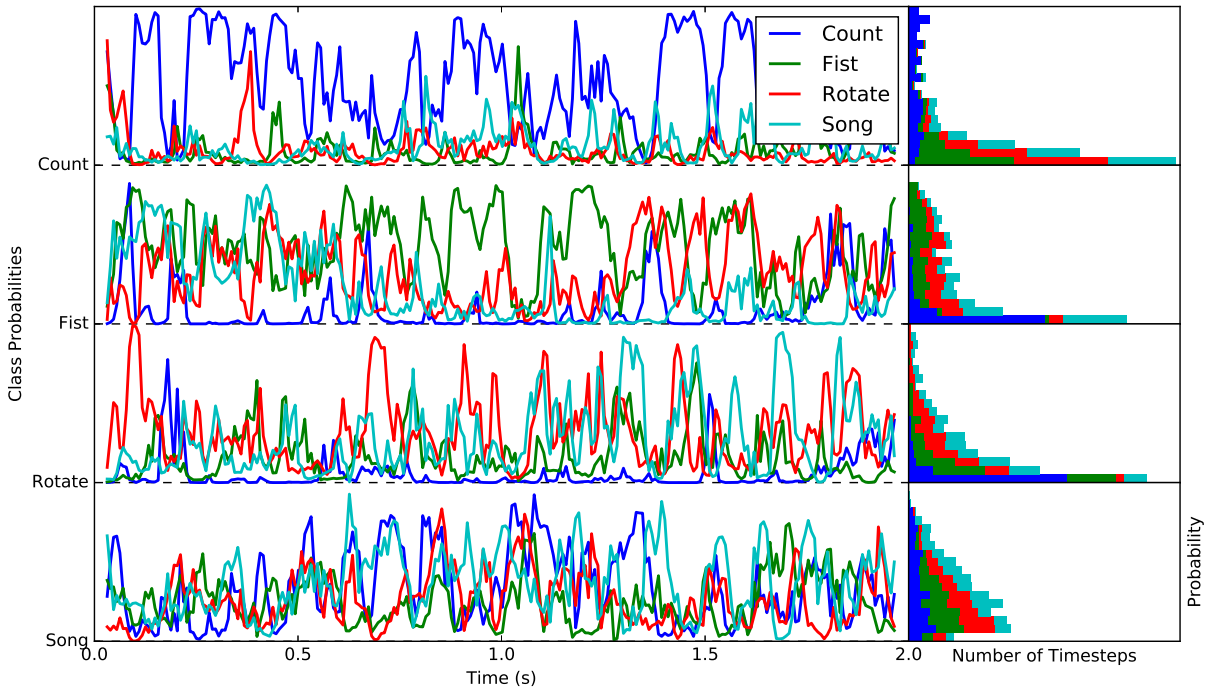
(b) Layer 2 activations.



(c) Layer 3 activations.



(d) Layer 4 activations.



(e) Layer 5 estimated class membership probabilities at each timestep.

Figure B.2: Activations produced at each layer by our full-sized CNN-TR, as described in Section 4.1.5, for Subject-1. These plots confirm that our experiments with reduced-size CNN-TRs, in Section 4.3.4 are generally representative of the behavior seen for a full-sized networks. (a) The activations produced by the first layer primarily lie in the linear region of the transfer function. (b) The activations produced by the second layer show more nonlinear behavior. There is also clear evidence that the second layer is performing filtering by frequency. For example, h^2_{16} contains primarily high-frequency information while h^2_1 contains lower frequency information. (c-d) The activations produced by the third layer, after pooling, and the fourth layer show increasing specialization. The activations also tend to oscillate more slowly, showing migration of information to slower time scales. (e) The final class membership probabilities are more accurate than with our reduced-size networks. The ordering of the predicted probabilities roughly aligns with the task confusions observed in Section 4.2.5.