Learning Deep Representations of EEG Signals in Mental-Task Brain-Computer Interfaces using Convolutional and Recurrent Networks

Preliminary Exam
March 27, 2017

Elliott Forney
Colorado State University
Department of Computer Science
elliott.forney@gmail.com

ABSTRACT

Classification of spontaneously produced Electroencephalography (EEG) signals is a challenging problem with important applications in Brain-Computer Interfaces (BCIs). A successful classifier must be able to capture spatiotemporal patterns while remaining time invariant and robust to noise and artifacts. The classifier should also be able to model the transient, nonstationary and nonlinear patterns found in EEG signals, despite the fact that these patterns have not yet been fully characterized.

Current approaches have not yet reached a level of performance that is acceptable for use in many practical applications and often involve extensive manual tuning and feature selection. These approaches also tend to rely heavily on prior assumptions and discard information that may be useful for classification. The recent trend in machine learning has been to move away from manually engineered solutions in favor of multilayer (deep) networks that rely on few prior assumptions.

Along these lines, we propose several variants of the Convolutional Neural Network (CNN) architecture that we have designed for analyzing and classifying EEG signals. The convolutional layers in these networks are designed to capture nonlinear spatiotemporal patterns at multiple time scales while maintaining time invariance. These convolutional layers can be viewed as nonlinear finite impulse response filters that automatically learn to process the signals. Class labels can then be assigned using a fully connected network or by accumulating evidence over a brief period of time.

Alternately, infinite impulse response can be attained by replacing the convolutional layers with recurrent layers. Since recurrent layers contain feedback connections, they are able to capture temporal information without explicitly embedding a window of time. This may lead to better filtering characteristics and increased memory capacity while requiring fewer free parameters.

This preliminary work highlights the limitations of current approaches and demonstrates the feasibility of using deep convolutional and recurrent networks to model EEG signals. Using an offline EEG dataset, we show that a minimally tuned CNN using raw EEG signals performs comparably to a highly tuned baseline classifier with extensive preprocessing. We also outline a number of methods and experiments for interpreting the models learned by our networks.

Since these networks rely on few prior assumptions, we believe that they will learn to utilize patterns in EEG signals that current approaches are unable to identify. This will lead to insights into the types of patterns contained in EEG signals and how these patterns change during various mental tasks. In turn, this may lead to a better understanding of human cognition and improved methods for use in the next generation of noninvasive BCI systems.

Contents

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 data-to-day tasks. For people with severe impairments, even BCIs with slow communication rates may be an invaluable tool. For those who have lost all or nearly all voluntary motor function, a condition known as locked-in syndrome, BCI systems 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 BCI, ranging from nuclear magnetic resonance 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. Figure 1 shows an individual wearing an eight-channel EEG cap in the Colorado State University BCI Laboratory.

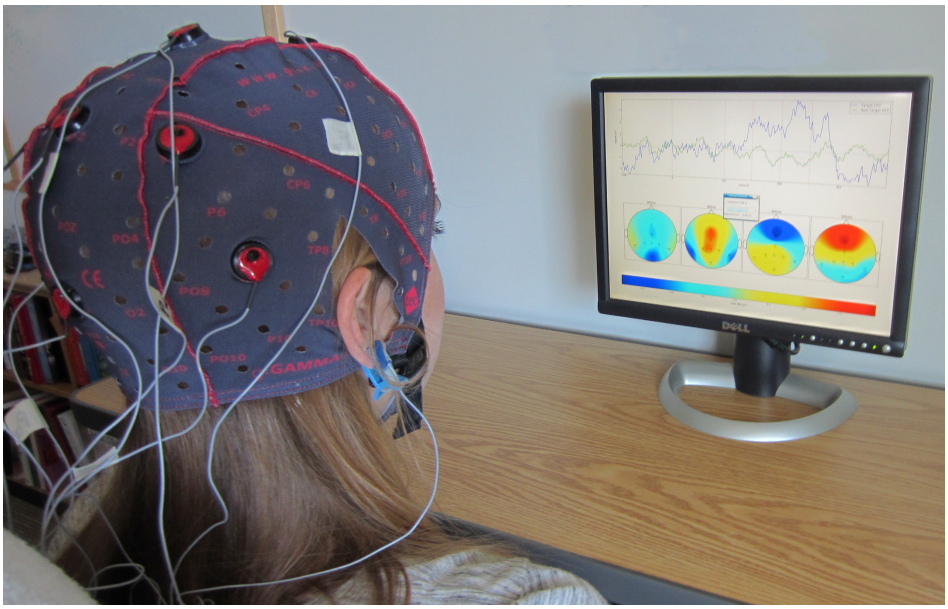


Figure 1: A participant wearing an eight-channel EEG cap while performing experiments in the Colorado State University BCI Laboratory. A small amount of electrically conductive gel is placed under each electrode in order to reduce the impedance between the sensors and the scalp.

There are also, however, a number of challenges that must be overcome when developing EEG-based BCI systems. Since EEG signals travel through the various layers of meninges, skull and scalp, the signals tend to be superficial and spatially blurred. This yields an imprecise picture of the underlying neural activity. EEG potentials also have small magnitudes, on the order of microvolts, which results in a low signal-to-noise ratio. It is typical for EEG signals to contain artifacts from biological sources, such as ocular movements, muscle contractions and sinus rhythms, as well as noise originating from

external sources, such as computer peripherals, household appliances and power mains. EEG signals appear to be particularly affected by noise and artifacts in real-world environments [13]. Furthermore, the complexity of the human brain suggests that EEG signals 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 day-to-day 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 the ERP is 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 paradigms operate in a synchronous, time-locked fashion that is not well suited for some types of control tasks. Moving a mouse cursor, steering an electric wheelchair and operating a prosthetic limb are all examples of tasks that are difficult to achieve in a fluid manner using ERP-based BCI paradigms.

Asynchronous paradigms that are not time-locked to external stimuli have also been proposed. For instance, 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 typically leverage a characteristic pattern in the EEG signals known as Event-Related (De)Synchronization (ERD/S). During motor movements, whether real or imagined, ERD/S generally appears as a decrease or increase in amplitude between the frequency ranges of 8–14Hz or 16–32Hz, known as μ and β rhythms respectively. These changes typically occur over the centro-parietal motor 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 a low communication rate and 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, it is hoped that perform-

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

For these reasons, we believe that EEG-based BCIs that follow the MT communication paradigm show considerable potential. Consequently, these types of BCIs will be the focus of the present work. Developing these types of BCIs is a difficult task, however, because they lack the type of single, clearly defined control signal found in other BCI systems. In fact, the types of changes in EEG signals that occur during various mental tasks appear to vary considerably across subjects and even across sessions for the same subject. As a result, robust machine learning algorithms are required in order to identify the relevant patterns in a user's EEG signals and discriminate between their mental states. Although several research groups have demonstrated methods for classifying EEG signals in MT-based BCIs, discussed further in Section 2, these approaches have not yet reached the levels of performance that are necessary for use in most practical applications. We assert that improved machine learning methods are required in order to achieve better classification performance and to further elucidate the types of patterns found in EEG signals recorded during imagined mental tasks. These improved methods may lead to better BCI systems as well as a better understanding of the nature of EEG signals and the human brain.

1.1 Challenges

Designing machine learning methods for use in MT-based BCIs has proven to be a unique and challenging problem. In order to clarify this, we have identified six specific challenges that must be addressed in order to successfully classify EEG signals in this setting.

Challenge 1: Time invariance.

Since asynchronous BCI paradigms are self-paced, i.e., they are not time-locked to a stimulus, the classifier must be able to identify the relevant patterns in the signal regardless of starting time. This tolerance to arbitrary shifts in time is known as time invariance.

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 are not yet well understood, EEG signals are known to have characteristics that are transient, nonstationary and nonlinear [38, 39].

Challenge 3: Patterns at multiple time scales.

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

Challenge 4: Noise and artifacts.

As discussed in Section 1, 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 controlled laboratory environments.

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 relatively high dimensionality and while using only about 10–15 minutes of sample EEG recordings.

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 in general, as well as for individuals. 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

Current approaches, which will be discussed in detail in Section 2, are inadequate for analysis and classification of asynchronous EEG signals because they fail to fully address one or more of the challenges outline in Section 1.1. Many of these approaches focus heavily on being easily interpreted and capturing patterns that are known 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

performed relatively well in the past, we will show that they omit several types of patterns, including spatial and nonlinear information, that may be important for classifying EEG signals.

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

Many current approaches also perform best when combined with linear classification algorithms. We concede that it is possible that purely linear models may be a good approximation of the underlying processes, especially in the face of high dimensionality and undersampling. We suspect, however, that the success of linear models may partially result from assumptions of linearity and from information loss during preprocessing. In any case, the problem of characterizing the, potentially nonlinear, dynamics of EEG signals recorded during imagined mental tasks has proven to be challenging and remains the topic of ongoing research [39, 40].

Given the complexity of the human brain, we believe that EEG signals likely contain a number different types of sophisticated patterns that have not yet been identified. Approaches that discard information and rely on prior assumptions about the patterns contained in the data, may limit the ability of algorithms 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 analysis and classification 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.

One of the principal advantages of deep networks is their ability to identify patterns and learn effective representations without requiring hand-crafted solutions for preprocessing, filtering, feature selection and dimensionality reduction, which can all lead to information loss. Following these principals, we propose several network architectures that are designed to address all of the challenges described in Section 1.1 while relying on few prior assumptions about the data and avoiding procedures that discard potentially useful information.

We propose several variants of the CNN architecture, discussed in detail in Section 3, that appear to be well suited for analysis and classification of EEG signals. These networks leverage convolution across time in order to achieve time invariance without explicitly discarding information. Since weights are shared across time, they may also have fewer parameters to tune than a fully connected network, which may help address problems with noise and undersampling. Nonlinear spatiotemporal patterns

can be learned by performing convolutions combined across all channels and through the use of nonlinear transfer functions. The layers of the network may also 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. Each artificial neuron in our CNNs can be interpreted as a nonlinear, multivariate filter. Since the parameters of these filters are found automatically, they rely on few prior assumptions and do not require manual tuning. The characteristics of these filters can be interpreted using techniques from time and frequency domain analysis and by examination of the learned parameters.

We also propose an approach that we call Deep Recurrent Networks (DRNs) that utilizes recurrent layers instead of convolutional layers. Since recurrent networks utilize feedback connections to capture temporal information, as opposed to convolution, we believe that they may require fewer free parameters than CNNs. Recurrent connections may also allow the network to learn longer-term patterns and achieve better filtering characteristics.

In the standard approach, a series of convolutional or, in our case, recurrent layers layers is followed by a fully connected network that assigns the final class label. Although this approach will be investigated, we also propose the use of sliding linear softmax layers combined with an evidence accumulation strategy. This may reduce the number of parameters to optimize while loosening the constraints on the network to learn a high-level ordering of events. We believe that this may allow the network to better handle transient and nonstationary patterns.

The principal hypothesis of our research, for which we will offer supporting evidence in this document, is that our proposed network architectures are well suited for learning effective representations of asynchronous EEG signals while relying on few prior assumptions. These networks are able to automatically learn to identify which types of patterns are important without discarding information that is typically lost in current approaches. As a result, we believe that our networks will lead to new insights into the types of patterns that are contained in EEG signals and how these patterns vary across different mental tasks. In addition to improving our understanding of EEG signals, we also believe that our networks have the potential to improve the performance of MT-based BCIs while requiring little manual intervention, which is an essential element for automated BCI systems.

In Section 2 of this manuscript, we begin by offering a brief review of current methods for classifying EEG signals in MT-based BCIs. This review will focus on the limitations of state-of-the-art approaches and the types of patterns that they are unable to capture. In Section 3, we give a more detailed discussion of our proposed methods, including our exact research questions, a detailed description of the network architectures that we will explore and the methods that we will use to analyze the results. In Section 4, we will describe the experimental procedures used for data collection and performance evaluation, including baseline methods that we will use for comparison. In Section 5, we present preliminary results. These results will first demonstrate that CNNs and DRNs are capable of identifying several important types of patterns in artificially generated signals. We will then use an offline EEG dataset to show that minimally tuned CNNs using raw EEG data can achieve a level of performance that is comparable to a highly tuned baseline classifier that requires extensive preprocessing. Finally, in Section 6, we will offer some concluding thoughts and a timeline for completing this research.

2 Background

In general, a BCI system consists of six components, depicted in Figure 2. First, the user alters their mental state in a way that conveys their intent according to an established communication protocol. In this case, the communication protocol follows the MT paradigm. Next, the EEG acquisition system monitors changes in the user's brain activity. This step typically includes analog-to-digital conversion

along with some preprocessing, e.g., referencing and bandpass filtering. In Figure 3a we see an example of a trace plot showing how the voltages of an EEG signal vary over time. For reference, Figure 3b shows the layout and naming convention for EEG channels in the 10/20 standard. 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. Some or all of the representation is then passed along as features to a classification algorithm. The classification algorithm identifies the mental state of the user so that the corresponding instruction can be sent to the device that the user wishes to control. Finally, the loop is closed as the user receives feedback from the controlled device. The focus of the present work is largely on the representation and classification components of BCIs; although, other preprocessing steps will also be an important consideration.

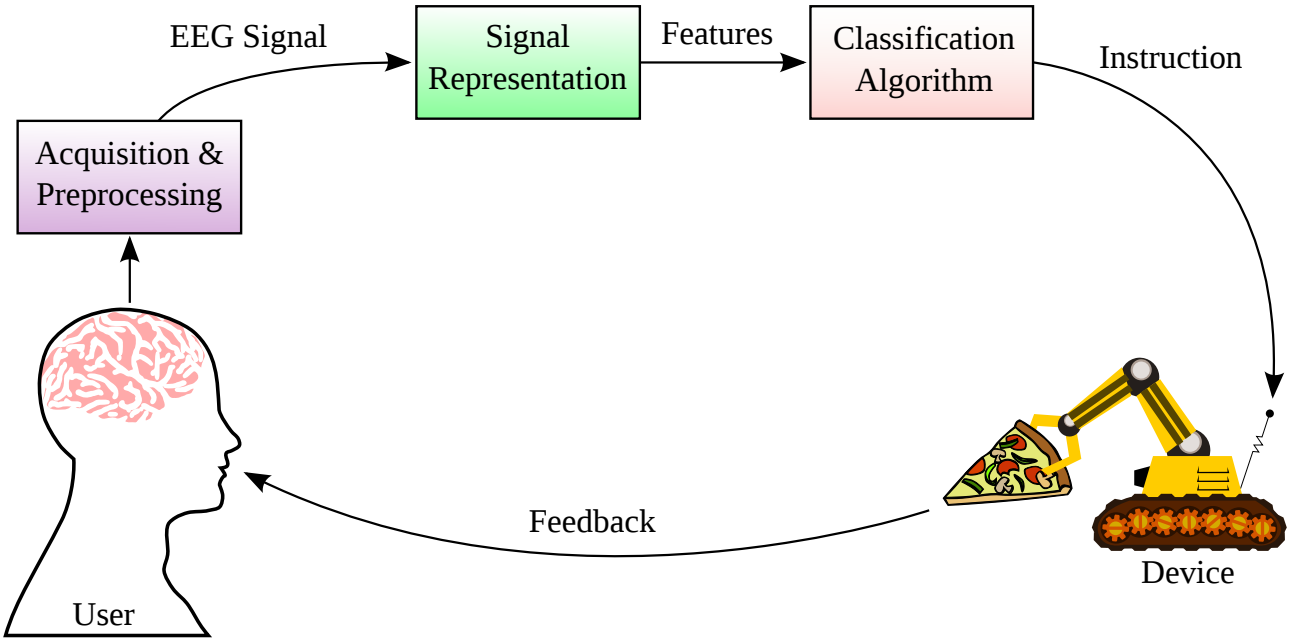
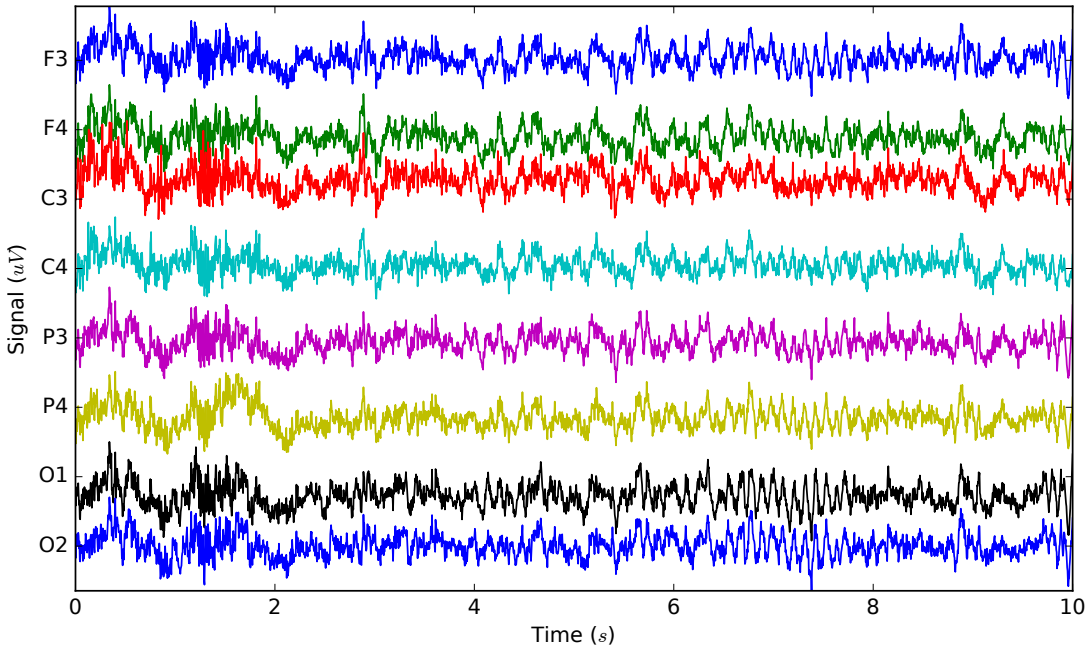
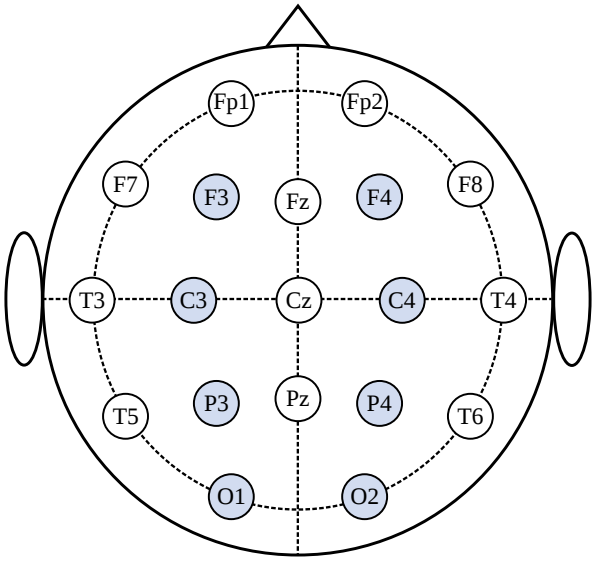


Figure 2: The basic components and flow of information in a general BCI system. The signal is first acquired from the user, converted to a digital signal and some preprocessing is typically performed. The signal is then converted to a representation that is useful for capturing the desired types of patterns. Features are then extracted from the representation and passed to a classification algorithms that attempts to identify the user's mental state. The label from the classifier can be used to send an appropriate command to the device being controlled. Finally, the user receives feedback about the action that the system has performed. *Portions of this image were taken from* `http://www.openclipart.com`

A number of representations and classification algorithms have been proposed for analyzing asynchronous EEG signals. Common representations include Fourier transforms, wavelet transforms, common spatial patterns, autoregressive models and time embedding [33, 34, 36, 44–56] For each of these representations, many classification algorithms have also been explored; however, simple linear classifiers often yield the best performance [57, 58]. Although the success of linear classifiers may be due to noise, undersampling and high dimensionality or because the relevant patterns are primarily linear, we suspect that assumptions of linearity and information loss during the representation and feature extraction stages may also play and important role. Despite the fact aggressive preprocessing and dimensionality reduction often appear to help performance, these procedures may also lead to a limited ability to capture some types of patterns.



(a) A trace plot of a 10-second EEG segment.



(b) Layout of EEG electrodes in the 10/20 standard.

Figure 3: A sample EEG segment. (a) time-domain trace plot of a 10-second EEG segment recorded while the subject relaxes. The vertical axis shows signal voltage spread across the channels and the horizontal axis shows time. High-frequency artifacts are visible between 0–2 seconds and eye movement artifacts around 3, 5 and 9 seconds. (b) The layout and naming convention for EEG channels in the 10/20 standard. The eight highlighted channels are used in our preliminary experiments.

In the remainder of this section, we will first briefly describe several state-of-the-art approaches for representing and classifying asynchronous EEG signals. For each of these approachs, we will emphasize its limitations and identify the types of patterns that cannot be captured. Where appropriate, we will highlight challenges from Section 1.1 that are not satisfied by a given approach. Our discussion will begin with the frequency domain and conclude with time domain approaches. Although the time domain is often more intuitive for those who are not familiar with signal classification, we have chosen to begin with frequency-domain approaches because they suffer from a number of limitations that can more easily be addressed in the time domain. Our discussion of time-domain approaches will also transition naturally into our discussion of CNNs and DRNs. Note that our CNNs and DRNs utilize time-domain signals.

2.1 Frequency-Domain Representations

Some of the most common methods for classifying asynchronous EEG signals utilize frequency-domain representations. In the frequency domain, a signal is represented in terms of periodic components, which makes it straightforward to describe amplitude, power, energy or phase across a spectrum of frequencies. Time invariance can be attained by omitting phase information; although, as we will show, this may limit the ability of these representations to capture some types of patterns.

Frequency-domain representations also have the advantage of being easy to interpret. Since oscillatory patterns appear to be important and commonplace in EEG signals, analysis in the frequency domain can yield powerful insights. In fact, many of the seminal works on asynchronous BCI systems cite inspiration in papers that examine how frequency-domain representations vary as a user performs different mental tasks [33, 34].

2.1.1 Power Spectral Densities

Power Spectral Densities (PSDs) are a popular choice for representing signals in MT-based BCIs [33, 34, 44, 59, 60]. PSDs represent a signal as power density ($\mu V^2/Hz$)[1] across a spectrum of frequencies. PSDs are estimated over a segment of an EEG signal by first using the Discrete Fourier Transform (DFT) to decompose the signal into a sum of sine waves in the complex domain. By taking the complex modulus of the DFT, phase information is discarded and the representation becomes time invariant. The result can then be scaled to obtain the PSD [61]. It is also common to use Welch's method to split the segment into windowed sub-segments, generate a DFT for each segment and then average the results. Welch's method can be thought of as a technique for reducing noise and dimensionality that results in fewer frequency bins. In Figure 4, we see an example of a PSD of an eight-channel EEG segment recorded while a subject rests for three minutes.

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 either cannot be captured using PSDs or for which the assumptions of the representation break down [62]. For instance, PSDs are not able to capture differences in phase that occur across EEG channels. This is illustrated in Figure 5 using two sine waves with identical frequencies but with phase offsets that differ by π radians. The fact that the resulting PSDs are identical demonstrates that PSDs cannot capture differences in relative phase in a multivariate signal. Although the inclusion of phase synchronization measures has been explored for use in MT-based BCI [63], estimating these metrics across various channels and frequency bands leads to a combinatorial explosion of features.

[1] The Hz in the denominator is the sampling frequency.

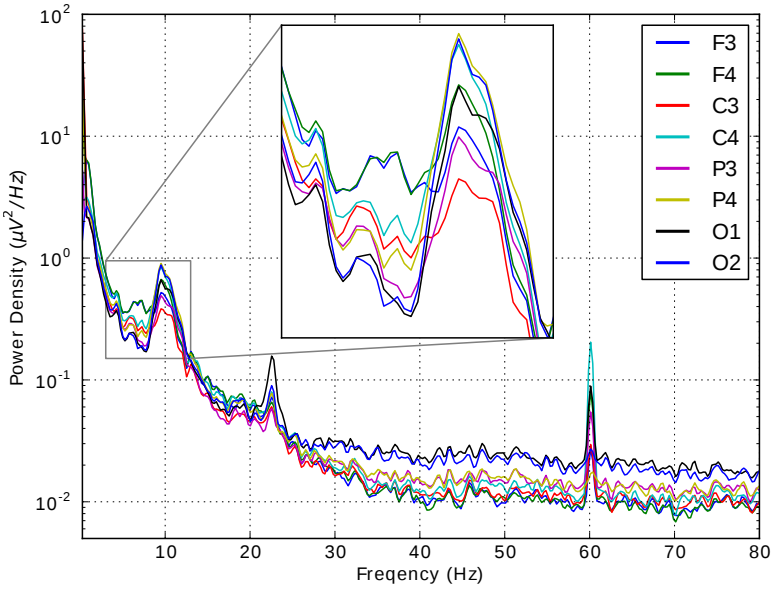
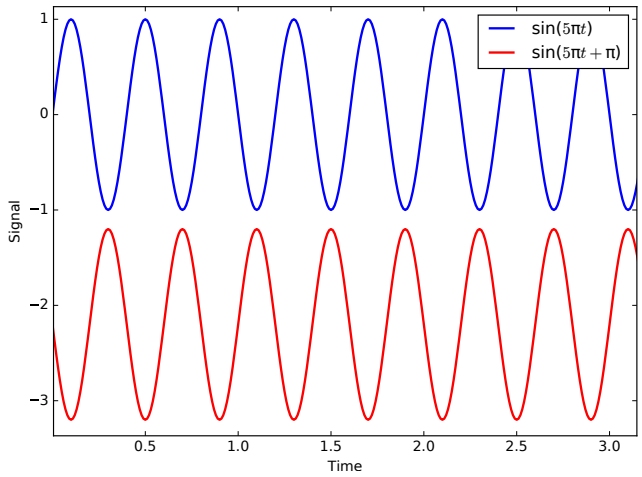
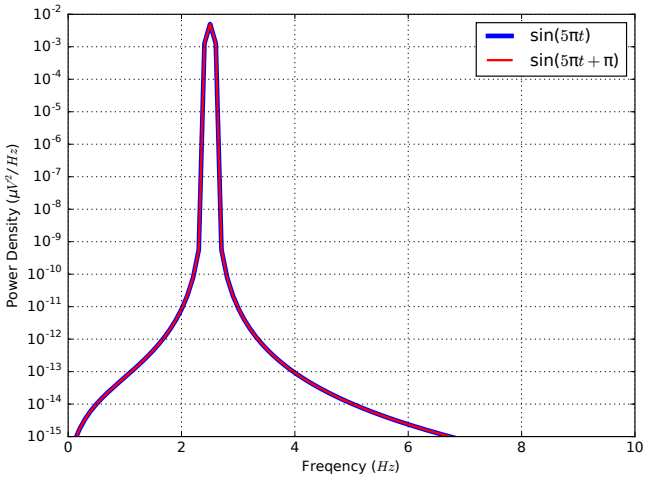


Figure 4: An example PSD of an EEG segment recorded while the subject is in a resting state. The inset axis highlights the differences in power across channels in the 8–16Hz α-band, which often increases in amplitude when a user relaxes.

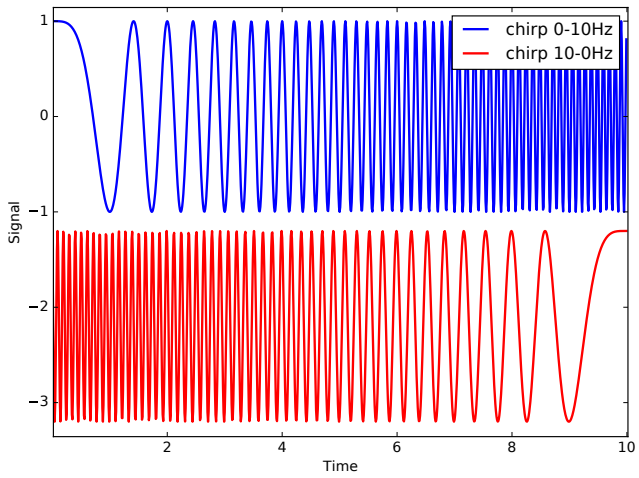


(a) Trace plots of two sinusoides with different phase offsets.

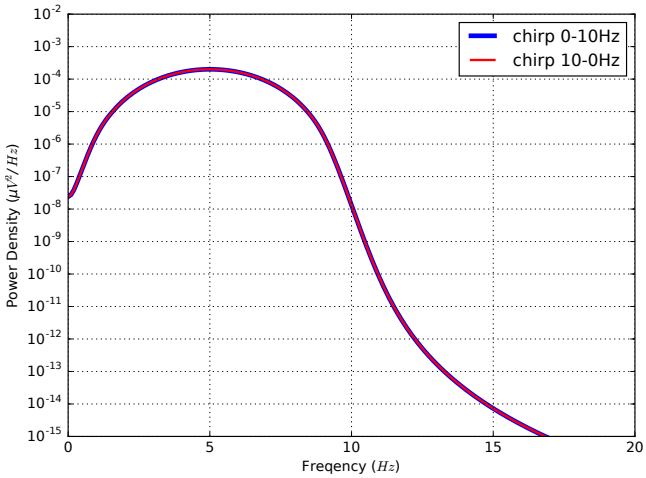


(b) PSD plots of the same two signals.

Figure 5: A demonstration of the inability of PSDs to capture differences in phase across channels. (a) Time-domain trace plots of two sinusoids with identical frequencies but different phase offsets. (b) Frequency-domain PSDs plots of the same two signals. Although the signals have different phase offsets, their PSDs are identical.

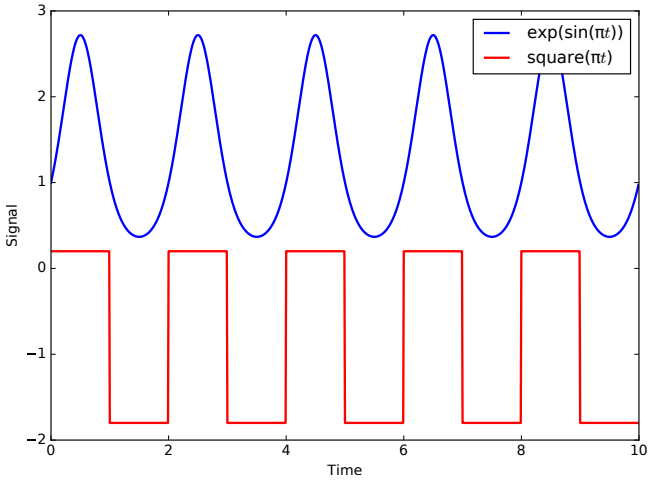


(a) Trace plots of two chirps sweeping from low-to-high and high-to-low frequencies.

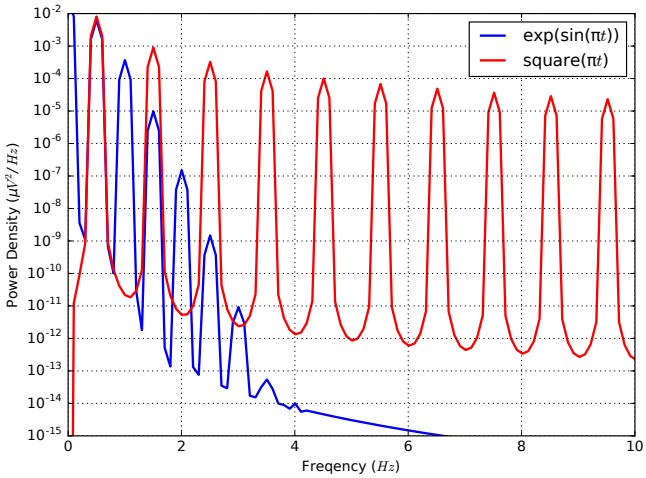


(b) PSD plots of the same two signals.

Figure 6: A demonstration of the limited ability of PSDs to capture the ordering of events. (a) Time-domain trace plots of two chirps sweeping from 0–10Hz and 10–0Hz respectively. (b) Frequency-domain PSD plots of the same two signals. Despite the fact that these signals contain different types of nonstationary patterns, the PSDs of both signals are identical.



(a) Trace plots of two nonlinear signals.



(b) PSD plots of the same two signals.

Figure 7: A demonstration of the limited ability of PSDs to model nonlinear patterns. (a) Time-domain trace plots of two nonlinear signals, $e^{\sin(\pi t)}$ and square(πt). (b) Frequency-domain PSD plots of the same two signals. Although the DFT is able to fit the nonlinear signals, it does so using a linear combination of many sine waves. This results in a number of peaks in the PSD that may lead to misleading results or violate our assumptions about the frequency ranges involved.

It can also be difficult for PSDs to represent patterns that are nonstationary, i.e., transient, drifting or aperiodic. Despite the fact that the DFT can be used to form a precise and invertible representation of any discrete signal segment, these representations are not well suited for modeling nonstationary patterns. This is due to the fact that the DFT represents the signal as in terms of sinusoids, which oscillate periodically and indefinitely. In other words, any patterns that are represented within a given signal segment are assumed to repeat indefinitely beyond the end of the segment. When this observation is combined with the loss of phase information, PSDs have a clearly limited ability to capture the ordering of short-term patterns. In Figure 6 we see a simple example that illustrates this limitation using two sinusoidal chirps that sweep linearly from low to high and high to low frequencies respectively. Although these two signal segments contain different types of nonstationary patterns, the corresponding PSDs appear identical.

In addition to assumptions about stationarity, PSDs also rely on assumptions about linearity. The DFT assumes that a signal can be appropriately described as a linear combination of sine waves. Furthermore, discrete sine waves can themselves be described as linear autoregressive processes. If these assumptions of linearity do not hold, then our interpretation of a PSD may be invalid. In Figure 7, we illustrate this point using two nonlinear signals, $e^{\sin(\pi t)}$ and a square wave. Despite the fact that both of these signals are periodic with a frequency of $\frac{1}{2}$Hz, the resulting PSDs show a number of peaks at various frequencies. This is because the DFT fits these nonlinear signals using a linear combination of many sine waves with different frequencies and phases[2]. Although this sum of sine waves does fit these signals, conclusions drawn solely from the PSD may be misleading.

BCI systems that utilize PSDs have enjoyed notable success. It is clear, however, that PSDs are not appropriate for modeling phase differences, nonstationary patterns and nonlinear patterns. If these types of patterns are present in EEG signals, then they may be overlooked by models that rely on PSDs.

2.1.2 Continuous Wavelet Transforms

Continuous Wavelet Transforms (CWTs) are a technique for generating a hybrid representation in both the time and frequency domains that may be useful for EEG classification [45, 64]. A CWT processes a signal using a function with a localized response, known as a wavelet. This wavelet is rescaled and then convolved with the time-domain signal in order to construct a filter with well-defined characteristics. The energy that is allowed to pass through this filter can then be determined at each time step for a given frequency. By repeating this process for a range of scales and frequencies, the power or energy spectrum of the signal can be estimated across both time and the frequency spectrum, known as a time-frequency spectrogram. Wavelets with a larger scale tend to achieve better frequency resolution while wavelets with a smaller scale tend achieve better time resolution. This trade-off can be exploited to achieve higher time resolution for the high end of the frequency spectrum, where the signal changes rapidly, and higher frequency resolution for low end of the frequency spectrum, where a large scale is is required to identify slow changes. As a result, CWTs can often achieve better time-frequency resolution than can be achieved with DFTs. In Figure 8 we see an example of a spectrogram of a resting-state EEG segment generated using a CWT with the morlet wavelet. This figure shows how various frequencies come and go over the course of time.

The high time-frequency resolution that can be achieved using CWTs makes them better suited than PSDs for capturing nonstationary patterns. In Figure 9 we see spectrograms generated using CWTs of the same two chirp signals shown in Figure 6. In this case, the CWT is able to identify how the signals change over the course of time. It is important to note, however, that as the scale of the wavelets is decreased, allowing more nonstationary patterns to be identified, the representation be-

[2]For a continuous signal, infinitely many sine waves are required to fit a nonlinear signal.

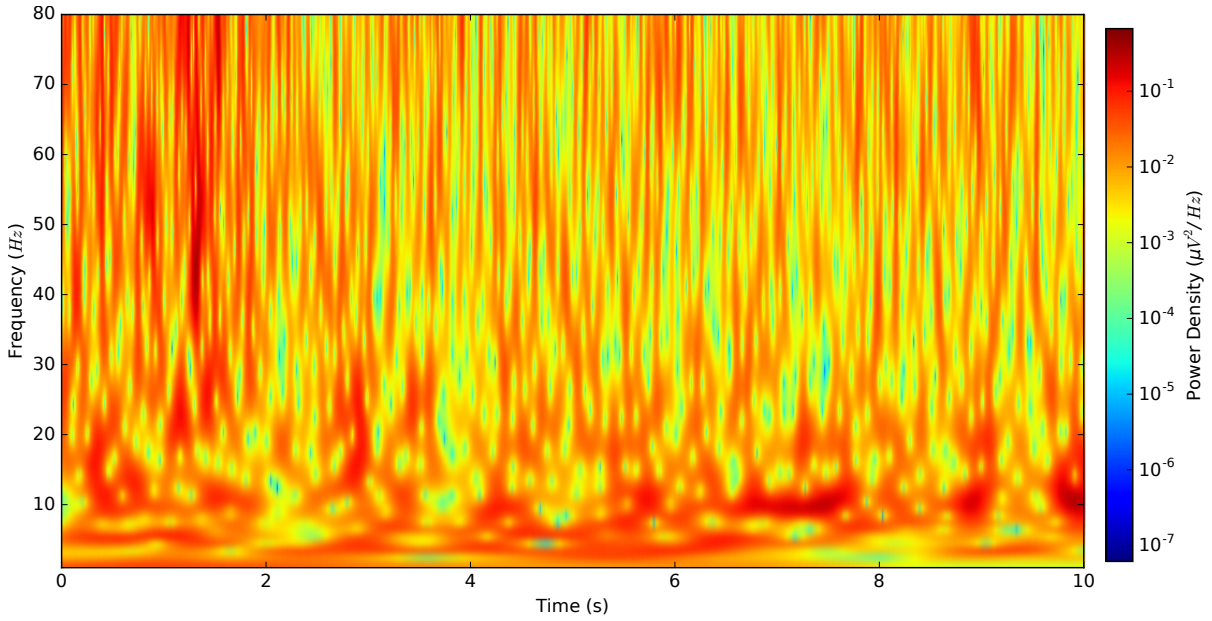
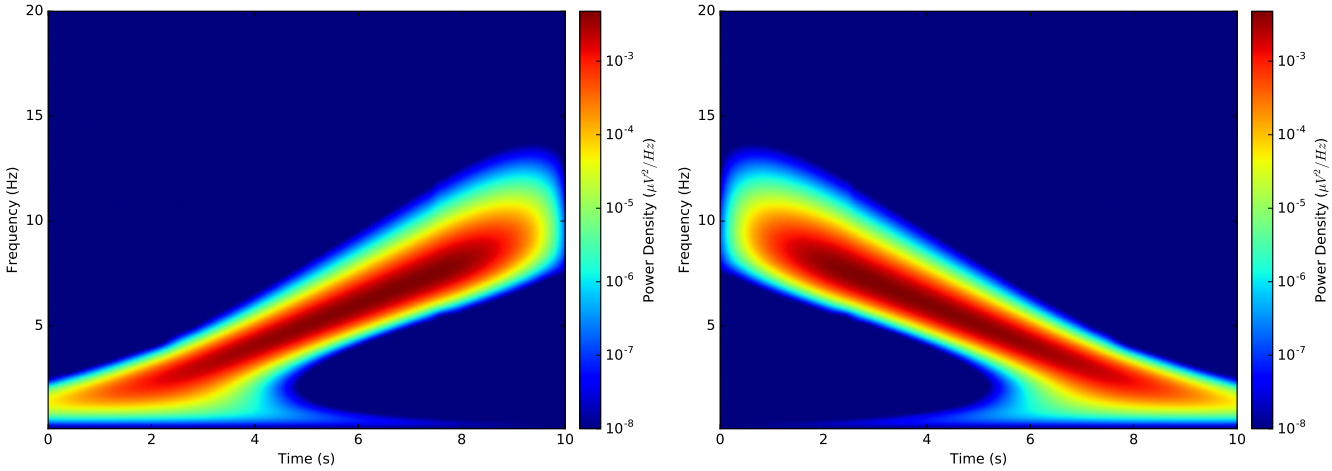


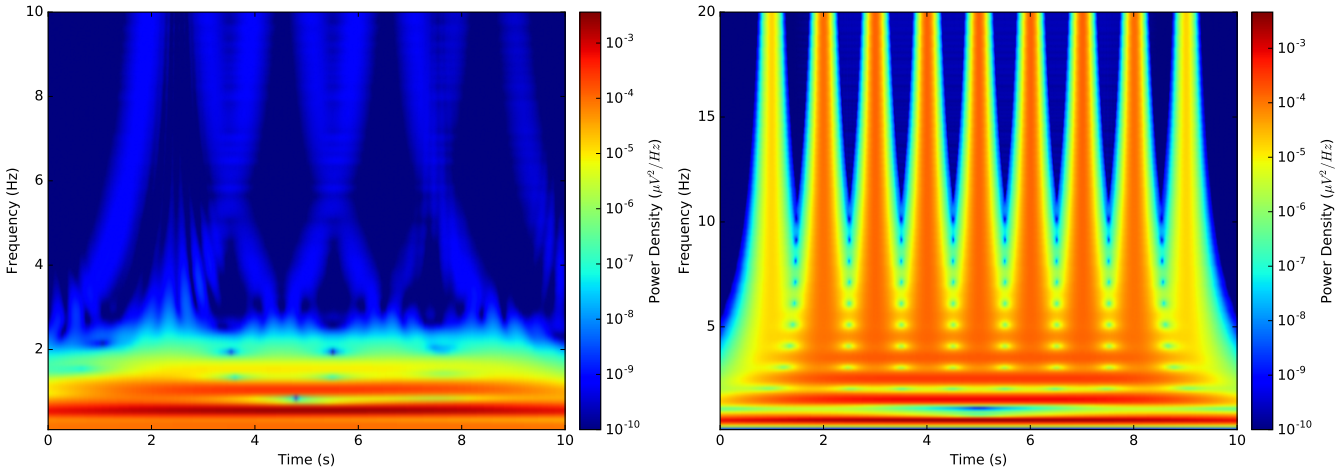
Figure 8: A sample spectrogram of a resting-state EEG segment constructed using a CWT. The bright areas appear at the times and frequencies where the power of the signal is high while darker areas indicate regions of low power content.



(a) CWT of a chirp from 0–10Hz.

(b) CWT of a chirp from 10–0Hz.

Figure 9: Two time-frequency spectrograms, generated using a CWT, of the same chirp signals found in Figure 6. These spectrograms are able to capture the differences between the low-to-high and high-to-low chirps. They have, however, also lost time invariance by creating a separate feature for each time step.



(a) CWT of $e^{\sin(\pi t)}$.

(b) CWT of square(πt).

Figure 10: Two time-frequency spectrograms, generated using a CWT, of the same nonlinear signals found in Figure 7. These spectrograms show a range of high frequency components and are unable to isolate the nonlinear frequency of $\frac{1}{2}$Hz.

gins to lose time invariance. In other words, spectrograms with higher time resolution are sensitive to shifts in time while spectrograms with lower time resolution fail to identify nonstationarities and the short-term ordering of events, as is the case with PSDs.

CWTs also have a limited ability to represent nonlinear patterns. In Figure 10, we see spectrograms of the same nonlinear signals shown in Figure 7. As was the case with PSDs, these spectrograms indicate a range of high frequency components rather than identifying the true nonlinear frequency of $\frac{1}{2}$Hz. Again, this may result in confusion between various linear and nonlinear signal components and may lead to ambiguous or inaccurate interpretations.

In general, CWTs have are fast to compute, have a straightforward interpretation and are capable of achieving a better trade-off between time and frequency resolution than PSDs. Nevertheless, tuning a CWT to have a high time resolution leads to the loss of time invariance, which is one of the principal advantages offered by frequency-domain representations. Higher time resolution also leads to lower frequency resolution, which may lead to a poor estimate of power content, especially for slowly changing patterns. On the other hand, tuning a CWT to have a higher frequency resolution leads to a representation that fails to capture nonstationary and rapidly changing patterns. Since CWTs describe the signal both across time and frequency, they also tend to produce a relatively high-dimensional representation. Although various regions of the representation can easily be omitted, such a procedure would require prior knowledge about important time or frequency ranges. As we have shown, CWTs also suffer from the limited ability to capture nonlinear patterns that we identified in PSDs. Overall, this analysis suggests that CWTs may be advantageous over PSDs with respect to interpretation and capturing nonstationary patterns; however, harnessing the ability of CWTs to capture nonstationary patterns may lead to higher-dimensional representations and an increased sensitivity to shifts in time.

2.2 Time-Domain Representations

EEG signals can also be represented in the time domain, i.e., as a function of voltage versus time. If we let N be the number of channels in our EEG acquisition system and T be the number of time steps in a segment of an EEG signal, then we can concisely represent the segment as the matrix

$$\mathbf{S} = \begin{pmatrix} s_{0,0} & s_{0,1} & s_{0,2} & \dots & s_{0,N} \\ s_{1,0} & s_{1,1} & s_{1,2} & \dots & s_{1,N} \\ s_{2,0} & s_{2,1} & s_{2,2} & \dots & s_{2,N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ s_{T,0} & s_{T,1} & s_{T,2} & \dots & s_{T,N} \end{pmatrix} \tag{1}$$

where element $s_{t,n}$ is the signal voltage at time t for channel n. The previously shown Figure 3a is simply a plot along the rows of such a matrix.

Although this representation is a natural and intuitive way to describe an EEG signal, time-domain representations can be difficult to interpret. Even a trained expert cannot reliably identify changes in EEG signals across mental tasks solely through visual inspection of EEG traces. It can also be challenging to derive features for a classifier that are time invariant while also capturing spatiotemporal information. For example, if all of $\mathbf{S}$ is passed a classifier as a single feature vector, then a small shift in time radically changes the representation. On the other hand, if each row of $\mathbf{S}$ is treated as feature vector, then no temporal information is captured.

Despite the challenges of working with EEG signals in the time domain, several promising methods have been proposed. These representations can be very general and do not necessarily assume that a signal is linear, stationary or periodic. They also do not require that phase information be discarded in order to achieve time invariance. Although these representations overcome many of the limitations we found with frequency-domain approaches, we will again see that each approach does not meet all of the criterion outlined in Section 1.1. We will use these observations as motivating arguments for exploring new time-domain approaches, i.e., CNNs and DRNs.

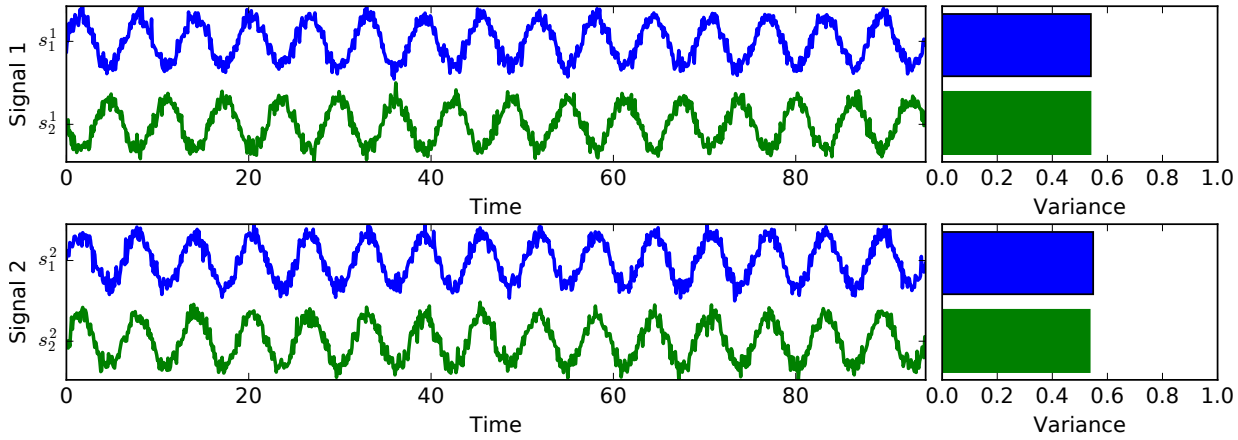
2.2.1 Common Spatial Patterns

Common Spatial Patterns (CSP), is a popular method for classifying asynchronous EEG signals in both MT [30] and MI [36, 46] communication paradigms. CSP applies a linear transform, P, to our time-domain signal representation, yielding a new signal

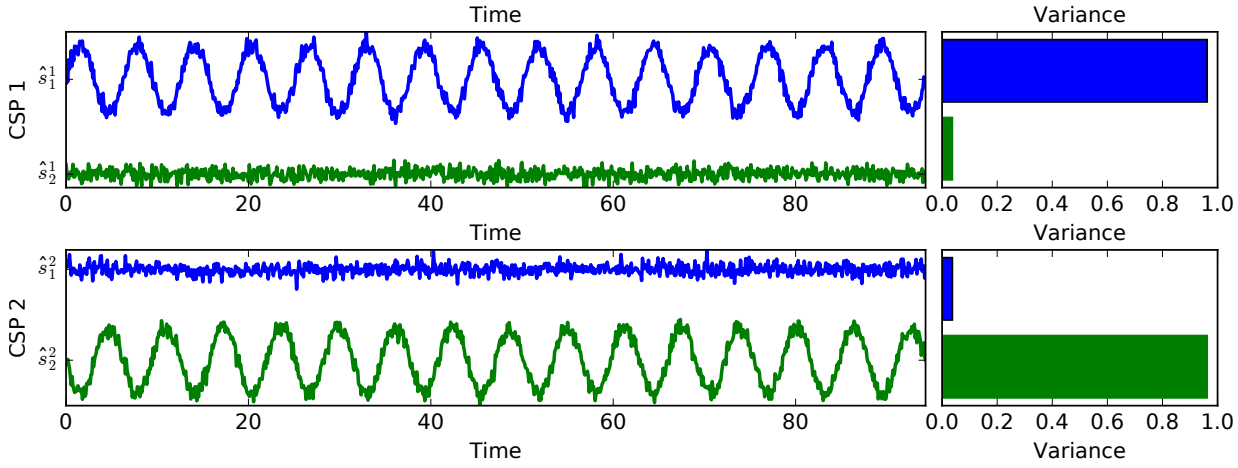
$$\hat{\mathbf{S}} = \mathbf{SP}. \tag{2}$$

The transform P is found analytically by solving a generalized singular value decomposition problem that maximizes the difference in variance among the components, i.e., columns of $\hat{\mathbf{S}}$, across two classes subject to the constraint that the columns of P are orthogonal. Computing P requires sample EEG from two classes and applying the transform to a novel EEG segment ideally leads to high variance in early components for the first class and high variance in later components for the second class.

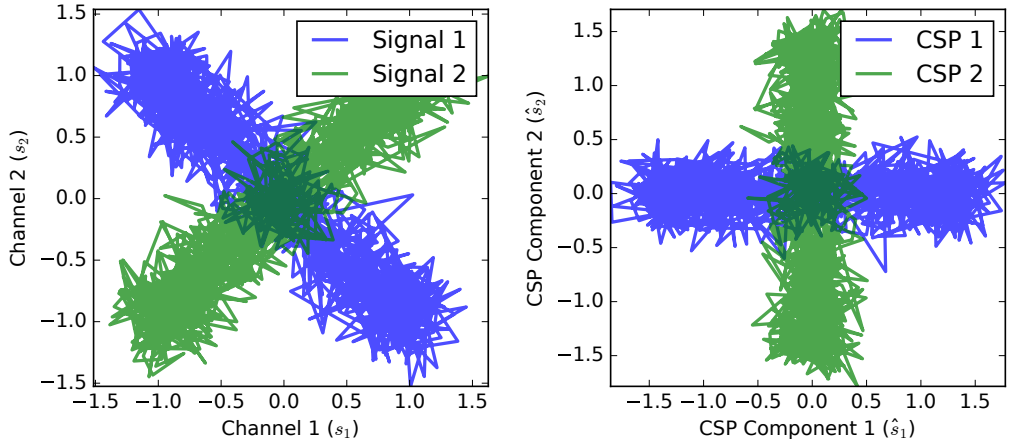
Figure 11 shows an example application of CSP to two classes of signals that each have two channels. Figure 11a shows trace plots of both signals as well as the variance across each channel. The first class, Signal-1, consists of two noisy sine waves with identical frequency and a phase difference of π radians. The second class, Signal-2, consists of two noisy sine waves with identical frequencies and identical phase offsets. Figure 11b 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 P to Signal-1 while CSP-2 is the result of applying P to Signal-2. Note that CSP-1 has higher variance in the first component while



(a) Trace plots and channel variances of bandlimited signals.

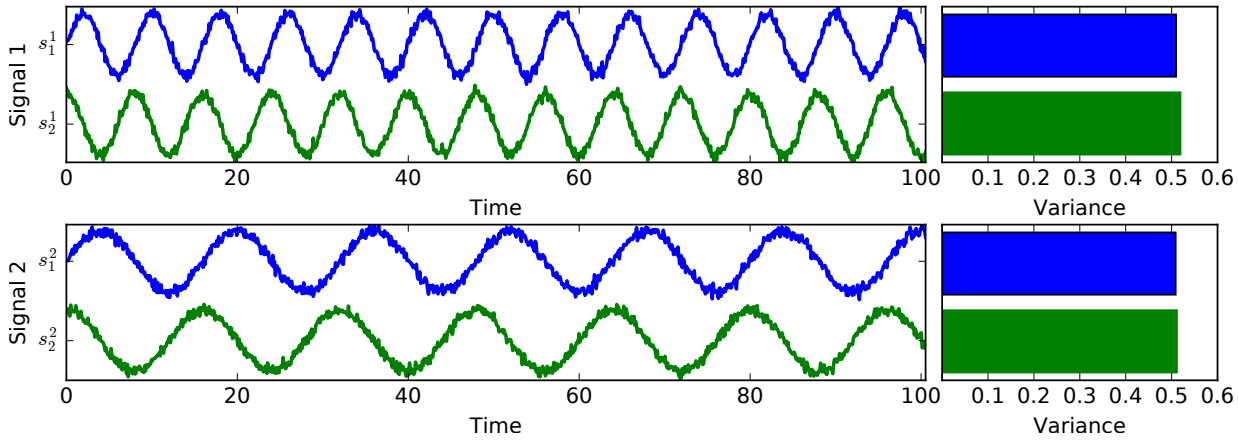


(b) Trace plots and channel variances of resulting CSP components.

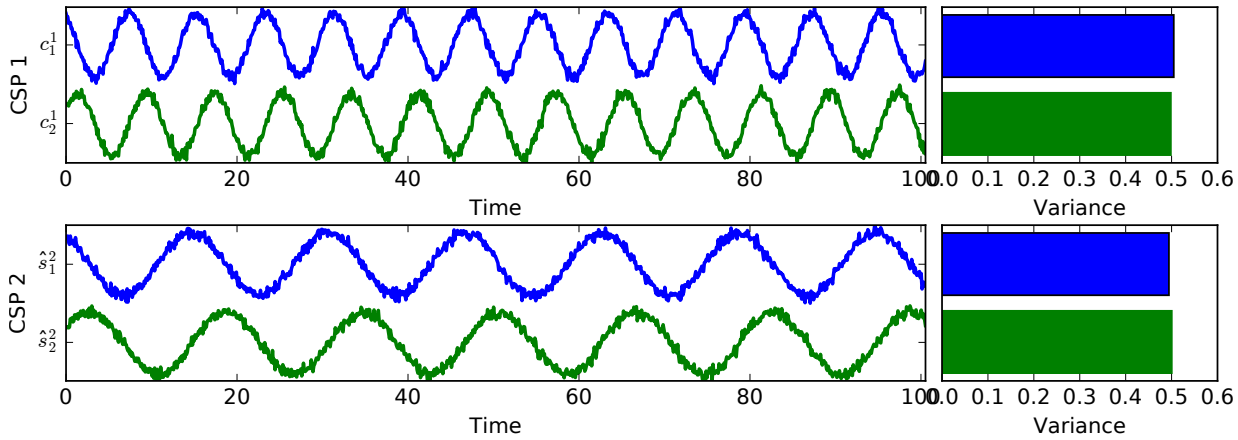


(c) Parametric plots of the signals versus CSP components on a plane.

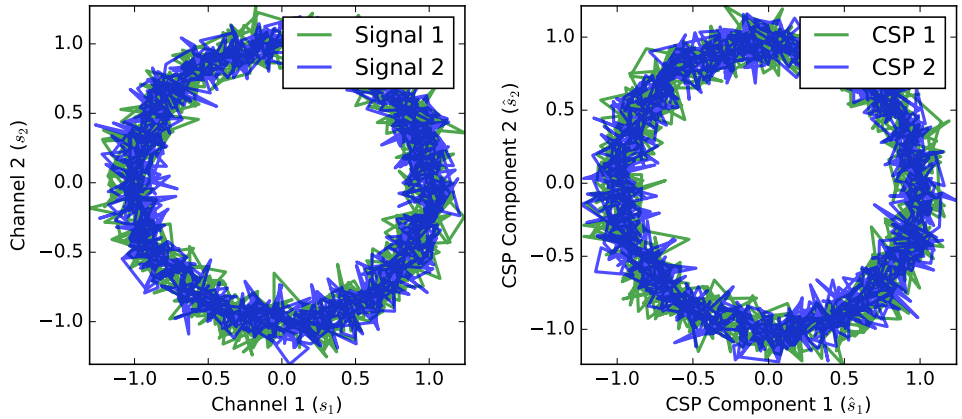
Figure 11: An illustration of how CSP can separate the signal variances for two classes of signals both consisting of two-channel, noisy, bandlimited sinusoidal signals. (a) Signal-1 consists of two sine waves with a phase difference of π radians. Signal-2 consists of two sine waves with identical phase 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.



(a) Trace plots and channel variances of non-bandlimited signals.



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



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

Figure 12: An illustration of how CSP can fail separate the signal variances for two classes of noisy, two-channel sinusoidal signals that are *not* bandlimited. (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 signal but 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 unchanges 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 and CSP yields the identity transform.

CSP-2 has higher variance in the second component, indicating that CSP was able to achieve good separation of the class variances. Figure 11c illustrates how this separation is achieved using a parametric plots of the signal channels and CSP components on the Cartesian plane. Before CSP is applied, the variance for both signals is spread roughly equally across both channels. Applying the CSP transform P 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 bandlimited signal, i.e., a signal that contains a narrow range of frequencies, measuring variance is equivalent to measuring the bandpower, or mean squared amplitude, of the signal. Following this intuition, classification typically proceeds by passing the variances of the CSP components as features to a subsequent classifier. This process reduces the dimensionality of the feature space to at most the number of channels times the number of frequency bands, or less if some components are omitted. CSP was originally designed for use in MI communication paradigms where there are changes in spatially distributed but narrow frequency bands, i.e., μ and β rhythms. Although CSP is usually formulated for binary classification, multiclass generalizations have been proposed [46, 65].

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 bandlimited signals and by dropping CSP components that are deemed unnecessary. These procedures can, however, be time consuming to establish and may vary considerably across subjects and sessions. CSP is fast to train and evaluate and the relative importance of each component and frequency band can be evaluated by mapping the weights or variances of each component back to the surface of the scalp. It can be challenging, however, to interpret temporal patterns precisely characterize spatial relationships using this approach.

Although CSP is typically able to capture differences in frequency across channels for bandlimited signals, it is not well suited for capturing these types of differences in broadband signals. Figure 12 presents an example of this limitation using a two-channel sinusoidal signal that CSP is unable to separate. Figure 12a shows trace plots and channel variances for both of these signal classes. In this case, both Signal-1 and Signal-2 consist of two sine waves with phase a phase difference of $\frac{\pi}{2}$. The frequencies of the channels are the same within each signal but different across signals, i.e., they are not bandlimited across classes. Figure 12b shows the CSP components for each signal. Note that CSP is unable to separate the variances. Figure 12c illustrates why this is the case using parametric plots of the signal channels and CSP components. All data points for both Signal-1 and Signal-2 lie on the unit circle. As a result, there is no linear transform that can separate the channel variances.

Despite the fact that CSP is fast and straightforward to interpret, it clearly does not address the challenges of capturing spatiotemporal, nonlinear and multiscale patterns. In fact, CSP is a perfect example of an approach that works well only under a set of strict prior assumptions, i.e., linear bandlimited signals and linear separability, and that requires extensive manual guidance and feature selection in order to select appropriate frequency bands and CSP components.

2.2.2 Time-Delay Embedding

Time-Delay Embedding (TDE) is a general method for capturing spatiotemporal patterns in time-domain EEG signals [53–56]. TDE incorporates a short window of adjacent time steps into each feature

vector. For a signal segment $\mathbf{S}$, as described in (1), the TDE representation is

$$\mathbf{S}^d = \begin{pmatrix} s_{0,0} & s_{1,0} & \cdots & s_{d,0} & s_{0,1} & s_{1,1} & \cdots & s_{d,1} & \cdots & s_{d,N} \\ s_{1,0} & s_{2,0} & \cdots & s_{d+1,0} & s_{1,1} & s_{2,1} & \cdots & s_{d+1,1} & \cdots & s_{d+1,N} \\ s_{2,0} & s_{3,0} & \cdots & s_{d+2,0} & s_{2,1} & s_{3,1} & \cdots & s_{d+2,1} & \cdots & s_{d+2,N} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \underbrace{s_{T-d,0} \quad s_{T-d+1,0} \quad \cdots \quad s_{T,0}}_{\text{Time embedding for first channel}} & \underbrace{s_{T-d,1} \quad s_{T-d+1,1} \quad \cdots \quad s_{T,1}}_{\text{Time embedding for second channel}} & & & \cdots & & s_{T,N} \end{pmatrix}, \tag{3}$$

where d is the number of adjacent signal values to include, known as the embedding dimension, T is the number of time steps in the segment and N is the number of channels. The rows of $\mathbf{S}^d$ are then considered feature vectors that are passed to a subsequent classification algorithm. Since each row of $\mathbf{S}^d$ contains signal values from across channels as well as from a window of time, these feature vectors are capable of capturing spatiotemporal patterns. It is important to note, however, that this approach assigns a class label at each time step. Since this is likely much faster than a BCI user can make decisions, some form of evidence accumulation, e.g., voting or averaging, is typically employed to build confidence in the user's intent before the BCI performs an action [56].

TDE representations are able to capture spatiotemporal patterns that are localized by the window of observations captured by time embedding. TDE also encourages time invariance because all windows of size d are considered. In other words, the classifier is trained to label each window regardless of its starting time. TDE requires minimal processing and does not discard information, other than the signal values beyond the embedding window. As a result, much of the burden of identifying relevant patterns is placed on the classification stage. Handling noise and artifacts, for example, must be performed by an additional filtering stage or by designing a classification algorithm that is sufficiently robust. Similarly, limiting the embedding dimension, d, is the only method for dealing with undersampling that is intrinsic to TDE. There is, however, a trade-off between the size of the embedding dimension and the length and scale of temporal patterns that can be captured. Of course, undersampling and overfitting may also be handled by the classifier using any number of regularization techniques. Since TDE-based approaches can typically be implemented using matrix operations, they are generally fast enough for real-time use; however, performance characteristics depend on the specific classification algorithm. Interpretation and visualization are, again, not intrinsic to TDE. Instead, standard tools for analyzing the relationships between the features or parameters of the classifier can be used to gain insights into the relevant patterns.

The advantages and disadvantages associated with TDE generally stem from the same source: TDE depends heavily on a classification algorithm for learning, regularization and interpretation. TDE does not rely on prior assumptions about linearity, stationarity, phase relationships or the bandwidth of the signals. These properties are compatible with the arguments we made in Section 1.2 and Section 1.3, which suggest that general methods that rely on few prior assumptions may lead to the discovery of new types of patterns in EEG signals and, subsequently, yield improved classification strategies for use in BCI systems. It has been noted that TDE often performs well with nonlinear classification algorithms [56], which may suggest that it is better suited for capturing nonlinear patterns in EEG signals.

On the other hand, TDE does little to address problems with noise and undersampling and adjusting the embedding dimension leads to a trade-off between high dimensionality and the ability to capture long-term patterns. As a result, TDE may still require filtering and dimensionality reduction techniques in combination with extensive model regularization. TDE also does not distinguish between different time scales and does not encourage the classifier to learn patterns at multiple levels of abstraction. In Section 3 we will introduce several deep neural network architectures that are, in fact, generalizations of TDE that are designed to address these limitations.

2.3 Deep and Recurrent Networks

We have now explored a number of common methods for constructing features that may be useful for classifying asynchronous EEG signals and demonstrated that each of these approaches has significant limitations. The goal of this discussion has been two fold. First, we have motivated the need for investigating new methods. Second, we have illustrated how these manually engineered representations fail to meet our stated goals and 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 networks, especially convolutional and recurrent networks, and describe how these methods have begun to play an increasingly important role in machine learning.

2.3.1 Deep Learning Revolution

For many years, machine learning methods have relied heavily on manually engineered features. In image classification, for example, intensity histograms, distance metrics, edges, segments, Fourier transforms and singular vectors have all been used extensively. Although many of these approaches have been successful, they typically involve a laborious design phase and are specialized to a specific kind of data. This has lead to machine learning systems that are able to fill only a narrow role. The ability of the human brain to learn and generalize 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 many problems in machine learning [66, 67]. There have, however, been several more recent advances have allowed artificial neural networks to achieve exciting performance improvements using only raw or minimally processed data. First of all, researchers have observed that biological neural networks often have multilayer, localized, sparse and recurrent connectivity patterns. Deep networks are artificial neural networks that mimic these configurations, usually in a very simplified way. Second, improvements in optimization algorithms and computer hardware, especially vector processors, have made it tractable to train these deep network architectures. Finally, the availability of large and high-quality datasets have provided large and diverse datasets from which deep networks can identify sophisticated patterns that generalize well. 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].

2.3.2 Convolutional Networks

Convolutional Neural Networks (CNNs) are a type of deep network that has been successfully applied to a number of machine learning problems, especially in computer vision [42, 43, 68, 69]. The CNN architecture was partly inspired by observations about the biological neural networks related to the animal vision system. In early stages of the vision system, it has been observed that individual neurons often respond almost exclusively to a small region of the retina that is associated with a corresponding region of the animal's visual field. This localized region of response is referred to as the local receptive field of the neuron. Similarly, in deeper regions of the vision system, it appears that neurons often respond to small, localized regions 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 a localized connectivity pattern. As the eye scans a scene, the result of light moving along the receptive field of the retina can be modeled as a

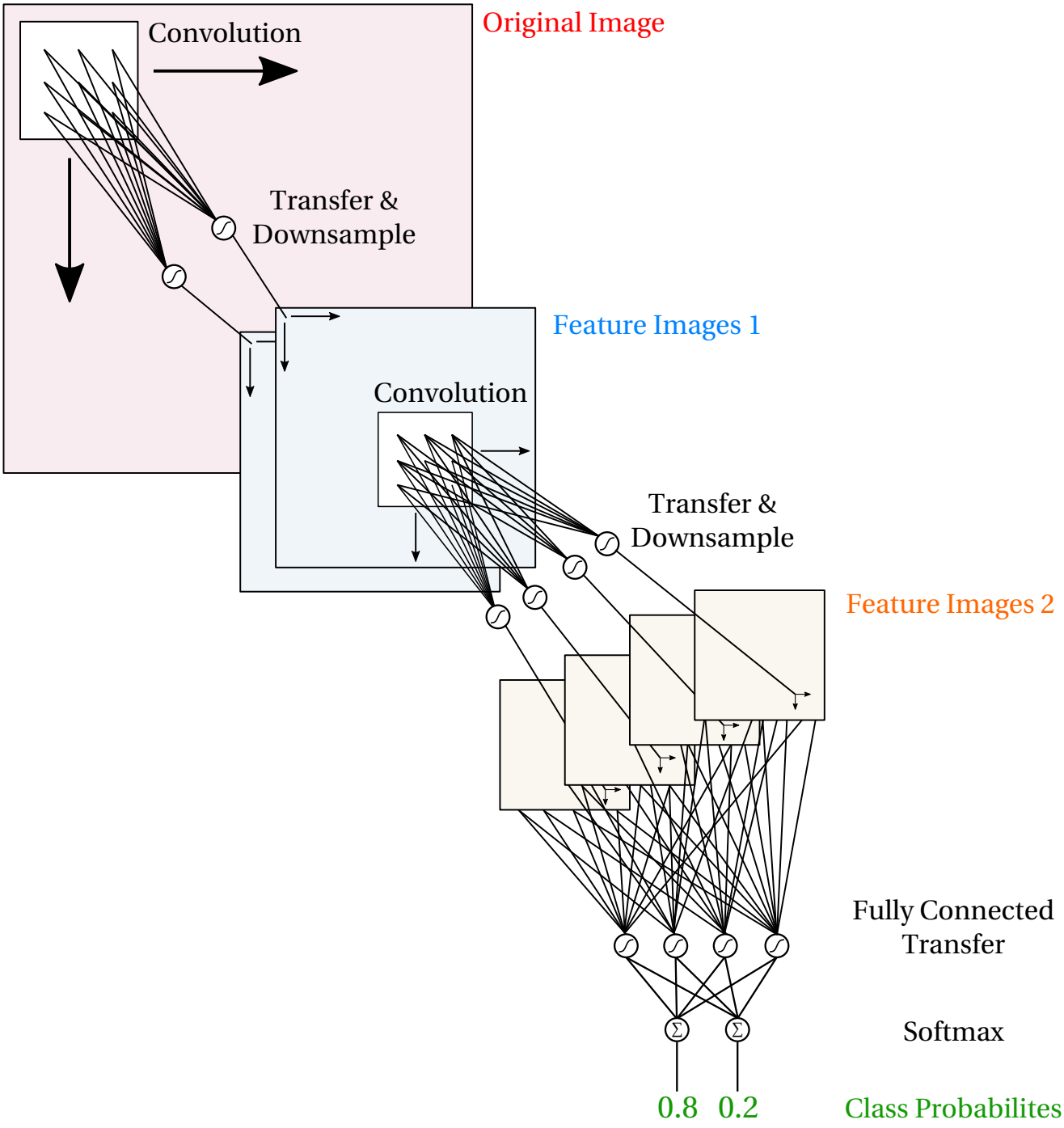


Figure 13: Schematic of Convolutional network designed for image classification. This network consists of two convolutional layers with two and four neurons followed by a fully connected layer with four neurons and, finally, a softmax readout that outputs two class probabilities.

convolution, i.e., a weighted sliding window of responses. The output of the neurons at the next layer is then convolved with the local receptive field of the previous layer, and so on.

CNNs also typically incorporate downsampling between layers, which serves a two-fold purpose. First, downsampling introduces sparsity into the network. Sparsity is also found in biological neural networks and may help generalization performance by preventing the network weights from coadapting to multiple features. Second, downsampling effectively changes the scale of the model, encouraging the network to learn a hierarchy of representations with multiple scales and levels of abstraction. Again, different regions of animal vision system are believed to model visual input at different scales.

In Figure 13, we see a schematic diagram of typical CNN used for image classification. The initial layers of the CNN, called convolutional layers, consist of several artificial neurons with small, localized windows of connectivity. Each connection in this window has a separate weight and 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 a rectified linear unit. 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 a different type of feature, regardless of its 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 multiscale representations.

The output of the final convolutional layer is then passed through an artificial neural network with full connectivity. In other words, a weighted sum of every pixel in all of the final feature images is passed to the neurons in the fully connected layer, followed by another nonlinear transfer. Finally, linear softmax units can be used to combine these outputs and ensure that the output of the network sums to one, i.e., the probability that the image belongs to each class. It is important to note that this fully connected network has many fewer parameters than a fully connected network applied to the original image because the feature images are downsampled between each convolutional layer. In fact, a CNN may have fewer parameters overall than a fully connected network. This reduction in the number of parameters can make the network more robust to noise, undersampling and overfitting. The parameters in all layers of a CNN can be optimized using gradient descent. In Section 3 we will propose several alternate CNN architectures that are suitable for classification of EEG signals.

2.3.3 Recurrent Networks

Recurrent Neural Networks (RNNs) are artificial neural networks that contain feedback connections, typically with a single time step delay. These feedback connections, also known as recurrent connections, allow the network to solve tasks that require memory and state. While networks that contain only feedforward connections can be viewed as function approximators, recurrent networks are capable of approximating finite state machines and Turing machines [70, 71]. Recurrent connections are also found in biological neural networks and contribute to its ability to perform temporal processing.

In Figure 14 we see a schematic diagram of an Elman type RNN, i.e., an RNN with full recurrent connectivity in the hidden layer [72]. In this network, a three-dimensional input signal $\mathbf{X}(t)$ is fed into the hidden layer at time t along with the previous outputs of the hidden layer from time $t-1$. The weighted sum of these outputs is passed through a sigmoidal transfer function and, finally, passed through a linear readout layer to produce the desired output $\mathbf{Y}(t)$. Since the state of the hidden layer is fed back recursively, the network is able to incorporate information from previous time steps.

RNNs can consist of a single or multiple hidden layers. In either case, they are intimately related to other deep network architectures in the way that they are trained. In order to optimize the weights

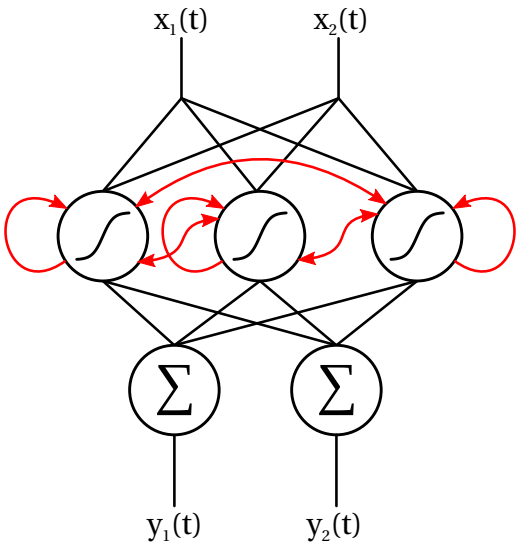


Figure 14: Schematic of Recurrent network for processing signals or time series. This network processes an input signal, $(X)(t)$, with three channels and has three recurrent neurons in the hidden layer followed by a linear readout layer that produces an output $\mathbf{Y}(t)$ with three channels.

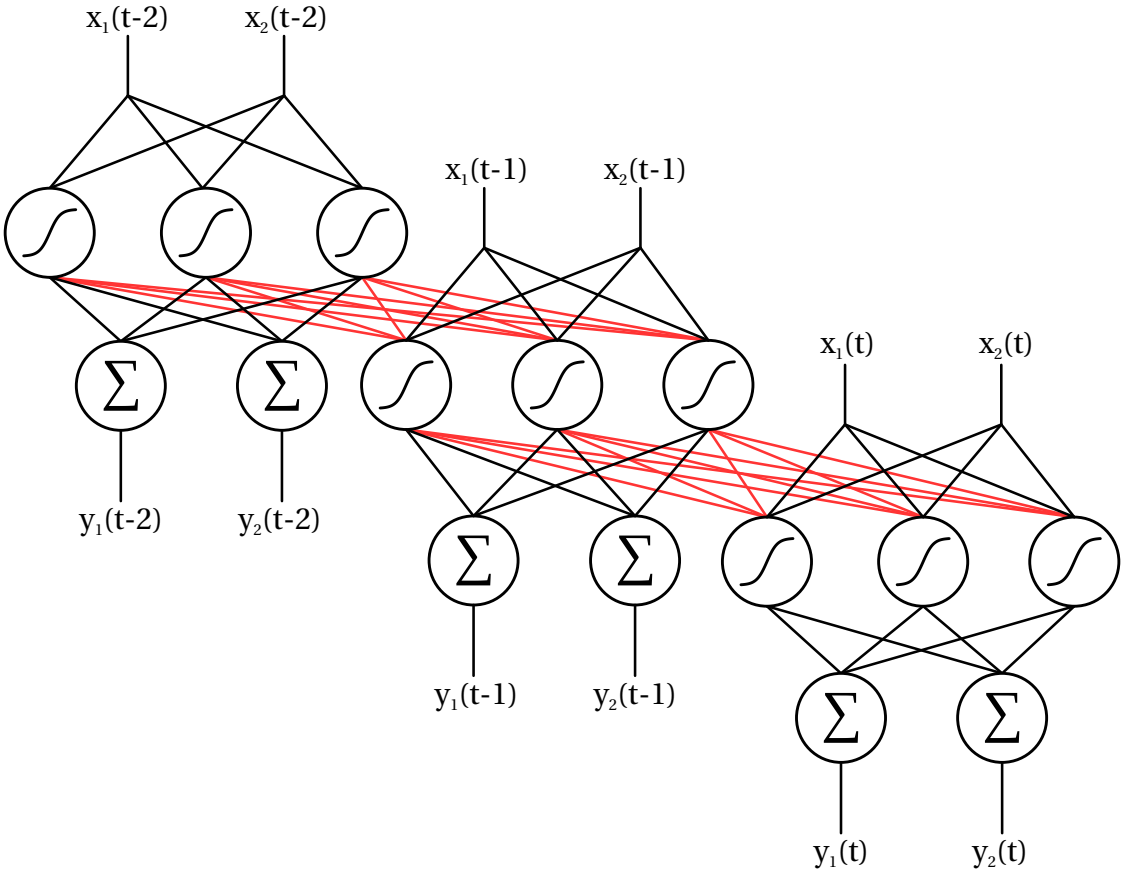


Figure 15: BPTT unrolls the network through time to form a deep cascading network. The weights at each layer are shared and the inputs consist of both the signal input at time t and the output of the previous layer. The gradients of the unrolled network are summed to find the gradient of the RNN.

of a recurrent network using backpropagation, the network must first be unrolled into a multilayer, cascading feedforward network. Depicted in Figure 15, this unrolled network shares identical weights at each layer and takes both the input of the signal at time t and the outputs of the previous layer as inputs. The gradient of the recurrent network can then be found by summing the gradients of the shared weights at each layer. In order to improve computational performance, this unrolling procedure can be truncated after a given number of time steps, yielding a short-term approximation of the gradient. This procedure is known as Backpropagation Through Time (BPTT) [66].

3 Proposed Methods and Research Questions

Now that we have motivated the use of deep and recurrent networks for classifying asynchronous EEG signals, we proceed by proposing a series of new deep network architectures that are specifically designed for solving this type of problem. First, we will describe how convolutional layers can be constructed in a way that is appropriate for processing asynchronous EEG signals. At this point, we will also lay down a foundation for analyzing and interpreting the types of patterns that these convolutional layers learn. Next, we will explain how recurrent layers can be used instead of convolutional layers and why this might be beneficial. We will then describe several approaches for combining convolutional or recurrent layers with one or more readout layers in order to construct a deep network that is suitable for classification. Finally, we will discuss how we plan to train these networks in a way that is computationally tractable for use in real-time BCIs. In each section, we will state a number of clearly defined research questions and describe, in detail, our suppositions, motivating design decisions and proposed experiments.

3.1 Convolutional Layers

One of the most fundamental questions that we seek to answer is whether or not the convolutional layers of a CNN can be constructed in a way that is appropriate for processing asynchronous EEG signals. In image classification, convolutional layers can help a network attain a level of location invariance across the horizontal and vertical axes of the image. In an image of a face, for example, the ears may be located anywhere near the sides or center of the image. In EEG signals, however, it seems unlikely that patterns are spatially invariant; rather, activity over one region of the cortex appears to be distinct from similar activity located over a different region. Asynchronous EEG signals do, however, require temporal invariance, as previously described in Section 1.1.

Research Question 1: Can convolutional layers be designed in a way that allows them to capture spatiotemporal patterns while also achieving time invariance?

We propose a straightforward modification of the standard convolutional layer where convolution is performed only along the time axis. In CNNs that are designed for image classification, it is common practice to incorporate information from different color channels by simply passing all channels through the convolutional window. We follow a similar procedure for incorporating information across multiple EEG channels where the output of the convolution at each time step is the weighted sum of all channels and time steps within a brief window of the input signal, as depicted in Figure 16. Since the connections across channels remain fixed while the convolution slides across time, this approach encourages the network to achieve time invariance but not necessarily spatial invariance. In Section 5, we will use simple artificial problems to explore the ability of this type of layer to identify spatially fixed but temporally variable patterns.

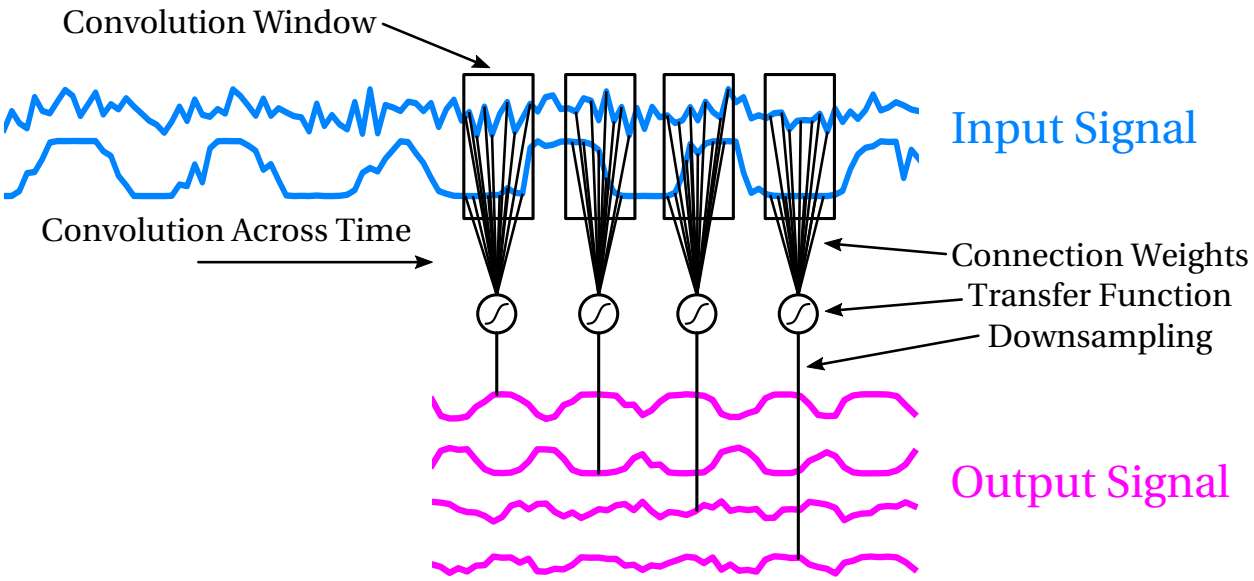


Figure 16: A single convolutional layer. Time invariance is achieved using convolution across time. Spatial information is incorporated by combining channels into the convolutional window.

When considering this type of convolutional layer, it is useful to note that the convolution of a multivariate signal along the time axis is exactly equivalent to a time-delay embedding (TDE) followed by a matrix multiplication. As a result, each convolutional layer in this type of network can be interpreted as an instance of TDE, previously discussed in Section 2.2.2. As we will see, however, the additional structure found in our CNNs may address the limitations that we identified in current TDE-based approaches.

A key limitation of TDE-based approaches is their inability to learn hierarchical representations of multiscale patterns. We believe that stacking convolutional layers will provide a structure that is better suited for forming hierarchical representations. We also believe that introducing a change in scale between layers, in the form of downsampling, will further encourage the network to identify multiscale patterns. Although these features are commonly found in CNNs, their effect on asynchronous EEG signals has, to our knowledge, not yet been studied.

Research Question 2: Does stacking convolutional layers to form a deep architecture encourage the network to learn hierarchical representations of multiscale patterns with varying levels of abstraction?

Characterizing the multiscale patterns learned by a CNN will likely be a challenging task because of the nonlinearities in the network and because of the sophisticated and only partially understood nature of EEG signals. There are, however, a number of experiments that can help us to explore the patterns that convolutional layers learn to utilize. First of all, a performance comparison between single-layer and multilayer networks may help us to determine the importance of using multiple convolutional layers. If multilayer networks outperform single-layer networks, this would suggest that hierarchical representations are important. The weights of the first layer can also be mapped directly to the surface of the scalp, helping us to determine which artificial neurons respond to different EEG sites. The first layer weights can also be viewed as matched filters, i.e., templates, highlighting the types of temporal patterns that trigger a response for each artificial neuron. Beyond the first layer, standard methods for visualizing the network weights and outputs may be revealing. For instance, trace plots may show how

the signal is transformed, spatial correlations may show relationships between channels and autocorrelations show temporal relationships.

Of course, frequency-domain approaches may also be used to analyze the outputs of each convolutional layer. This idea leads us to another important insight. In classical signal analysis, linear bandpass[3] filters can be implemented as a convolution with weights that are designed analytically according to the desired specifications of the filter [73]. This type of linear filter is known as a Finite Impulse Response (FIR) filter because the effects of a perturbation on the input signal, such as that caused by an impulse, last for a finite period of time that is determined by the width of the convolutional window. Similarly, we believe that the convolutional layers in our networks may be interpreted as learned, multivariate, nonlinear FIR filters.

Research Question 3: Can convolutional layers be interpreted as learned FIR filters? If so, do these filters separate frequencies that are thought to be useful and do they attenuate noise and artifacts?

The frequency response of a linear FIR filter, i.e., the attenuation of the signal across the frequency spectrum, can be determined using the discrete Fourier transform (DFT) of the convolutional weights. Similarly, the frequency response of a convolutional layer before the application of a nonlinear transfer function can be determined precisely by taking the DFT of the weights. The frequency response of a convolutional layer after the nonlinearity can be explored by feeding a chirp as the layer's input while examining the output using PSDs or CWTs. It is important to recognize, however, the limitations and pitfalls of using these approaches to analyze nonlinear signals, as discussed in Section 2.1.1 and 2.1.2.

Research Question 4: Are nonlinear transfer functions strictly necessary in order for convolutional layers to learn meaningful patterns? What might this suggest about the types of nonlinear patterns found in EEG signals?

Since convolutional layers can be interpreted as FIR filters and given the success of linear methods in current BCI systems, we believe that it is important to establish the need for using a nonlinear transfer function at all. Although our experience has demonstrated that at least one nonlinear layer is usually necessary in order for CNNs to achieve acceptable performance, it is not entirely clear why this is the case. An interesting compromise between strictly linear layers and the typical hyperbolic tangent is the rectified linear transfer function. Rectified linear transfer functions, which have become increasingly popular in recent years, are fast, piece-wise linear and may introduce additional sparsity into the network [68, 74]. A direct comparison of the performance and patterns learned by linear, rectified linear and hyperbolic tangent transfer functions may yield interesting insights into the types of nonlinearities found in EEG signals.

3.2 Recurrent Layers

Recurrent Neural Networks (RNNs), introduced in Section 2.3.3, have been successfully applied to a number of problems involving signals and time series. Several recent research projects, including work by our group, has demonstrated that RNNs are well suited for modeling asynchronous EEG signals and that these models can be used to construct generative classifiers [49–52, 75–78]. It has also recently been suggested that recurrent layers may be a suitable alternative to convolutional layers for some types of image classification problems [79]. We posit that recurrent layers are a viable, and perhaps

[3] We use the term "bandpass" generically in this context to include highpass, lowpass and stopband filters.

better, alternative to convolutional layers when constructing deep networks for classification of asynchronous EEG signals.

Research Question 5: Can recurrent layers be designed in a way that allows them to capture spatiotemporal patterns while also achieving time invariance? Can recurrent layers be used as an alternative to convolutional layers?

Convolutional layers, as we have seen, can utilize a brief window of adjacent time steps to incorporate temporal information. Recurrent layers, on the other hand, are designed to utilize feedback (recurrent) connections in order to capture temporal information. In Figure 17, we see an example of a recurrent layer with full feedback connectivity that is applied recursively to each time step of the signal. A nonlinear transfer function and downsampling can also be applied in a fashion similar to that used in our convolutional layers. These recurrent layers can also be stacked to form Deep Recurrent Networks (DRNs). Since information from previous states is fed back to the network at each time step, the need for explicitly embedding a window of adjacent time steps can be eliminated. Also note that this process only depends on the order in which signal was presented to the network, rather than absolute timing. This encourages the network to learn a time-invariant representation. In order to capture spatial patterns, we feed the inputs from all channels to each artificial neuron at each time step, similar to the approach taken with our convolutional layers.

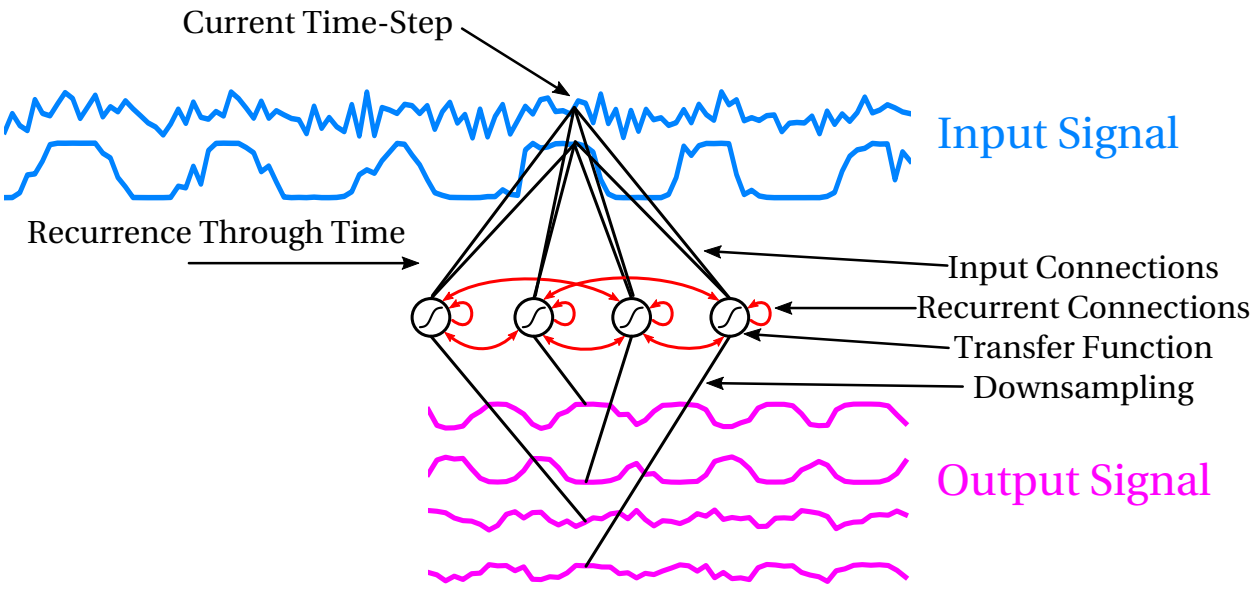


Figure 17: A single recurrent layer with full recurrent connections applied recursively across time. Spatial information is incorporated by connecting all channels at each time step.

There are several advantages and disadvantages to using recurrent layers instead of convolutional layers. First of all, recurrent layers may, possibly, require fewer free parameters than a comparable convolutional layer because recurrent layers do not require a connection to each time step within a window. Recurrent layers may, however, require more artificial neurons than a convolutional layer in order for the capacity of the network to reach a level that is suitable for retaining information that is fed back through the recurrent connections. The characteristics of this trade-off will depend on the density and duration of relevant temporal patterns and will be studied empirically. If recurrent layers require

fewer free parameters than convolutional layers, they may be more robust to noise, undersampling and overfitting, which may result in better generalization performance.

With convolutional layers, we also noted that the finite length of the convolutional window limited the duration of the network's response to a given input, yielding FIR. Recurrent layers, on the other hand, continually incorporate information from previous states in a recursive fashion. As a result, a perturbation of the input signal can, theoretically, affect a recurrent layer's response infinitely far into the future. This property is known as Infinite Impulse Response (IIR). In practice, however, the useful duration of a recurrent layer's response is limited by numerical precision and truncated gradient approximations.

Research Question 6: Does the infinite impulse response found in recurrent layers improve the filtering characteristics of the network? What impact does this have on performance and the patterns learned by the network?

In classical signal processing, IIR filters are often preferable to FIR filters because they can achieve better filtering characteristics, i.e., steeper attenuation of undesirable frequencies, using fewer parameters. On the other hand, IIR filters produce nonlinear phase distortion, may exhibit numerical instability and deriving appropriate filter parameters is considerably more involved.

Similarly, we suspect that recurrent layers may achieve better feature selection and filtering characteristics with fewer parameters than convolutional networks. A direct comparison of convolutional and recurrent layers using the analysis approaches outlined in the previous section may help to support or weaken this supposition. We also suspect, however, that recurrent layers will be more difficult to train than convolutional layers. Since recurrent layers are typically unrolled during training, as described in Section 2.3.3, these network can become very deep, especially if multiple recurrent layers are stacked. This may lead to slow or limited convergence due to truncated or vanishing gradients and local minima [80]. These problems may be mitigated through the use of carefully designed optimization algorithms and transfer functions.

3.3 Readout Layers

In order to construct deep networks that are useful for classification, one or more convolutional or recurrent layers are combined with one or more readout layers that are used to assign final class labels. While the convolutional or recurrent layers may be viewed as learned filters or feature extractors, the readout layers can be viewed as the final classification stage. In practice, however, the lines between these roles may be blurred. Next, we will describe several approaches for constructing readout layers that will be used to form deep networks that are capable of classifying EEG signals.

The first full architecture that we propose, depicted in Figure 18, closely follows the standard CNN design. These networks begin with several stacked convolutional layers operating along time with nonlinear transfer functions and downsampling between layers. The output of the final convolutional layer is then passed through a fully connected layer where each channel and time step for the entire EEG segment is passed to each artificial neuron. This first readout layer also contains nonlinear transfer functions. Finally, the output of the first readout layer is passed to a linear softmax layer that outputs class membership probabilities for the entire segment. We refer to this CNN architecture as CNN-Fully-Connected (CNN-FC) due to the fact that the readout layers contain full connectivity to the entire output of the final convolutional layer.

Although CNN-FC is designed to address many of the challenges that we have laid out, we believe that the use of fully connected readout layers leads to several significant problems. First, these networks require many free parameters because of the large number of connections from individual time

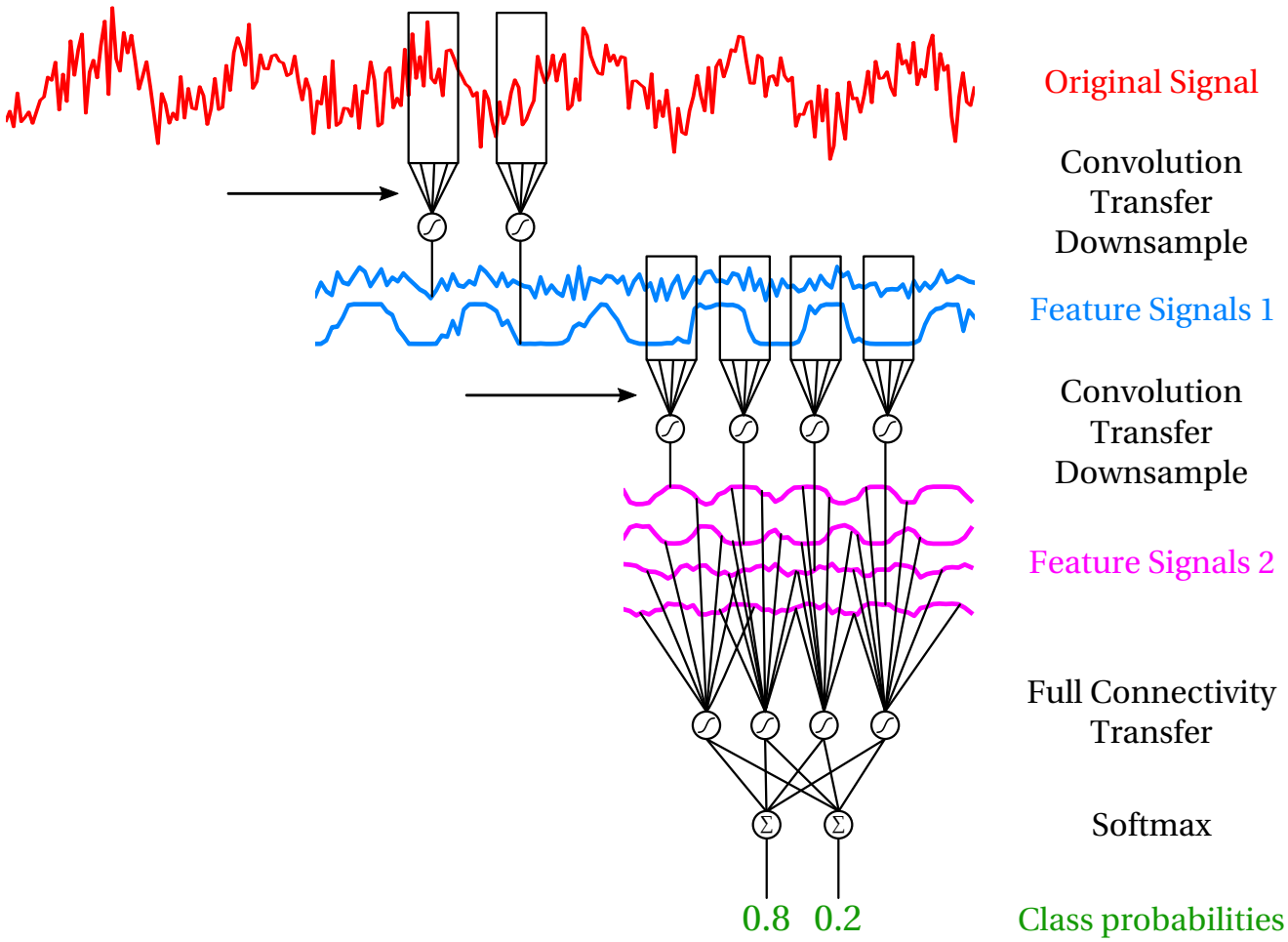


Figure 18: Schematic of convolutional network with fully connected, nonlinear readout (CNN-FC). After one or more convolutional layers, the network outputs proceed through a fully connected layer where all channels and all time steps are passed to each nonlinear unit. The outputs of the first fully connected layer are then passed to a linear softmax layer that outputs class probabilities for the entire EEG segment.

steps leading into the fully connected readout layers. For example, a CNN-FC that operates over two-second EEG segments sampled at 256Hz and that has two convolutional layers with 10 artificial neurons, 2:1 downsampling and 10 hidden units in the first fully connected layer requires approximately 12,000 parameters in the readout layers alone. Given that our data is known to be noisy and undersampled, we suspect that this large number of free parameters will lead to severe problems with overfitting and lead to poor generalization performance.

It is also important to note that fully connected layers are especially well suited for capturing patterns with large-scale relative structure. In order to illustrate this, again consider how CNNs that are designed for image classification leverage the location invariant features learned by the convolutional layers. In these problems, there is typically a high-level structure that dictates how these features fit together. In images of faces, for example, various features, such as eyes, ears and noses, may be found at differing locations. These features should, however, have a relative structure with the eyes on the sides of the nose and the ears on the sides of the eyes and, often, the entire face near the center of the frame. Fully connected readout layers are capable of capturing this type of large-scale relative structure.

The signals that we are concerned with, on the other hand, appear to lack this type of structure in several ways. First, asynchronous EEG signals are completely time invariant, i.e., the boundaries of the signal segment are completely arbitrary. Second, our previous experience with this type of problem leads us to believe that many of the patterns found in the EEG signals are transient and fleeting. This phenomenon may be explained, at least in part, by the inconsistent and repetitive-but-variable nature of the mental tasks that the subject is performing. For instance, if a subject is imagining raising and lowering the arm, the regions of the brain responsible for controlling a particular muscle may be activated repetitively but in a different order while the arm is being raised versus while the arm is being lowered. The overall timing of these activations may also vary due to inconsistencies in the way that the subject repeats the task. Given this variability and the undersampled nature of this data, we believe that capturing this type of large-scale structure may be intractable. Instead, we posit that the presence or absence of features with localized structure, e.g., imagined control of a specific muscle, may be more useful than long-term relative structure.

Research Question 7: Do EEG signals recorded during mental tasks contain discernible long-term relative structure that necessitates fully connected readout layers? Can this type of structure be captured given the large number of parameters required by fully connected layers? If not, what alternatives may be appropriate?

In our previous research, we have explored the use of TDE combined with various classifiers and evidence accumulation strategies [53–56]. In these approaches, TDE is used to capture localized patterns in the EEG signals. The classifier then uses this information to assign a class membership probability at each time step. During training, this is achieved by repeating each segment's class label for each time step. Since this decision rate is faster than a user can control a BCI and because this short-term information is often not enough to reliably assign class labels, an evidence accumulation strategy is used to build confidence in the classifiers decision over the course of time.

In Figure 19, we present a network architecture that generalizes this procedure to deep CNNs. In this case, the first readout layer consists of softmax transfer functions that output class membership probabilities at each timestep using only the outputs of the final convolutional layer at each time step. Note that this approach is somewhat similar to an architecture that has been proposed for semantic segmentation in computer vision [81]. Omitting fully connected layers places the burden of capturing temporal information solely on the convolutional layers and forces the network to identify localized patterns that occur within a narrow window. It is important to note, however, that the stacked, mul-

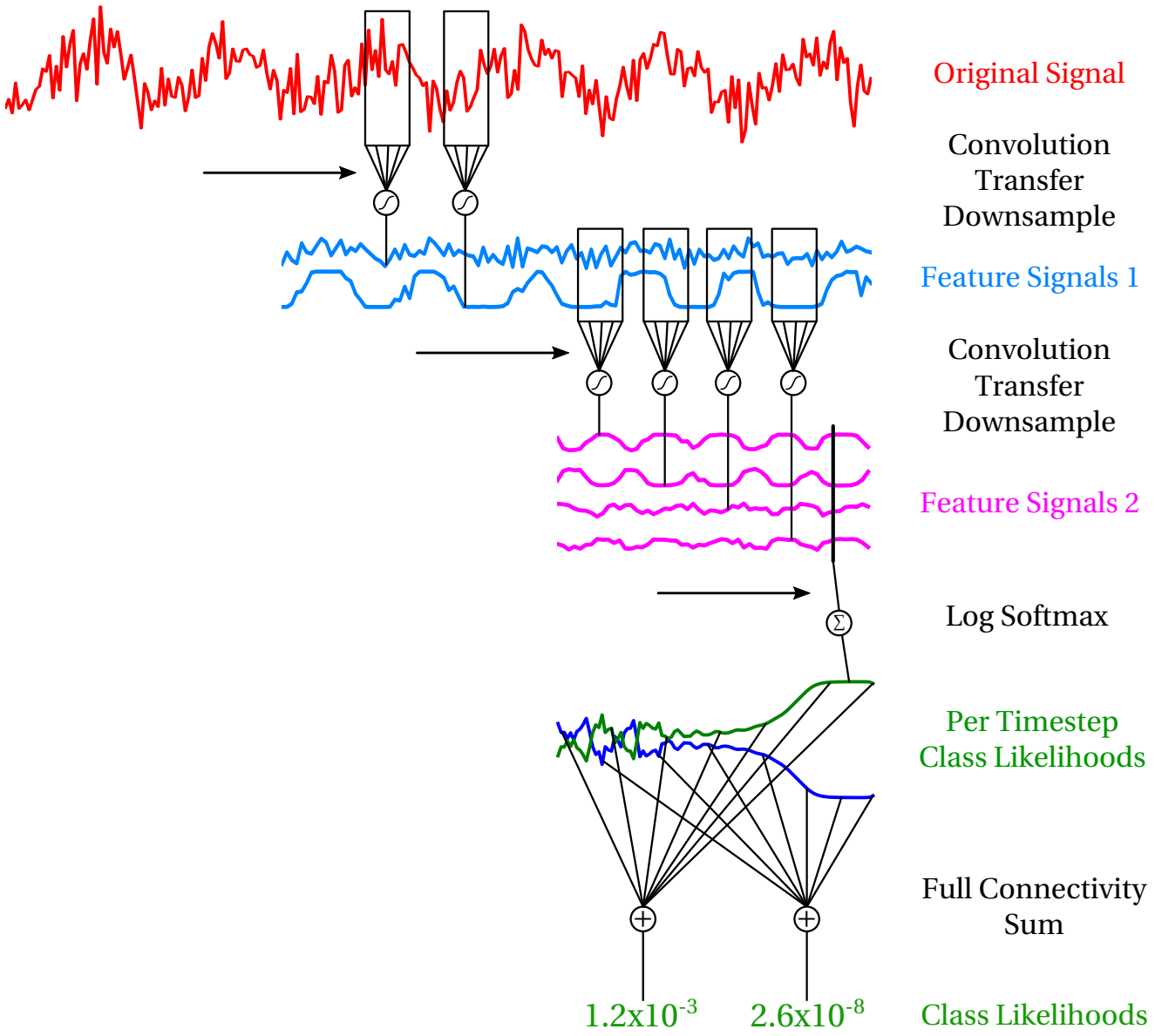


Figure 19: Schematic of convolutional network with log sum readout (CNN-EA). After one or more convolutional layers, all channels are passed to a linear log-softmax layer that assigns a class label for each individual time step. These log likelihoods are then summed, which is equivalent to multiplying the probabilities. The final label for the entire segment corresponds to the maximum of these likelihoods.

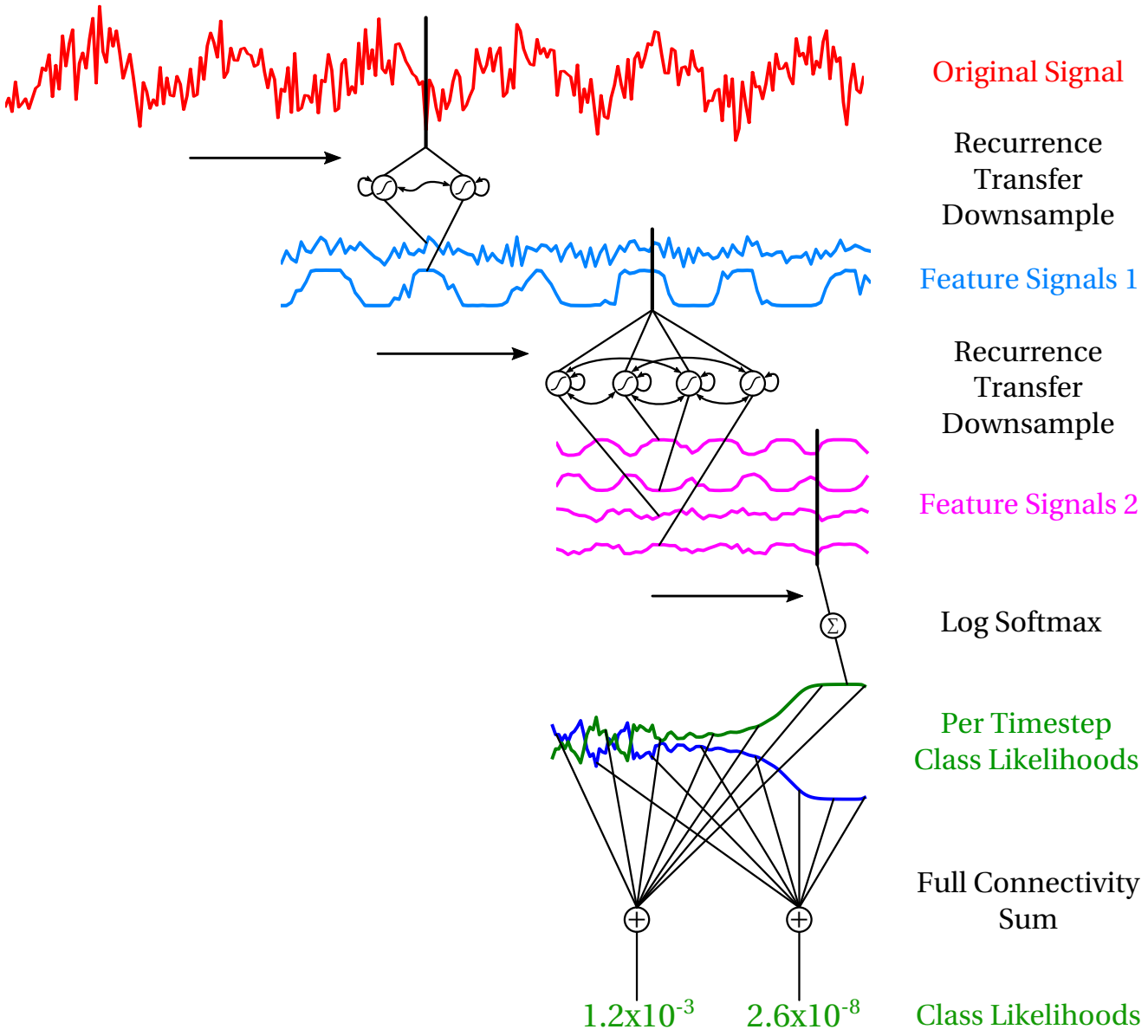


Figure 20: Schematic of deep recurrent network with log sum readout (DRN-EA). After one or more recurrent layers, all channels are passed to a linear log-softmax layer that assigns a class label for each individual time step. These log likelihoods are then summed, which is equivalent to multiplying the probabilities. The final label for the entire segment corresponds to the maximum of these likelihoods.

tiscale nature of the convolutional layers allows the network to identify patterns that are wider than the width of the convolutional window at any given layer. The second, and final, readout layer does not contain any learned weights. Instead, the final readout layer simply combines or accumulates the probabilities from the first readout layer in a way that assigns a single class membership probability for the entire EEG segment. We refer to this network architecture as CNN-Evidence-Accumulation (CNN-EA) because it combines the probabilities from each time step instead of using fully connected layers.

Research Question 8: Is evidence accumulation sufficient to capture the relevant and necessary patterns in our EEG signals? If so, what types evidence accumulation strategies are most appropriate?

There are many evidence accumulation strategies that can potentially be used in the second readout layer of CNN-EA. For instance, voting or averaging class probabilities are both straightforward choices. In our preliminary results, we will accumulate evidence by summing the log-likelihoods. This is equivalent to multiplying the probabilities and can be interpreted as the probability that all regions of the signal belong to a class given that the regions are independent, an assumption that is almost certainly violated. In our previous work related to TDE, we have also suggested the use of evidence accumulation strategies that only make decisions after sufficient evidence has been gathered [56]. A comparison of several evidence accumulation strategies may lead to insights into the overall structure of patterns found in the EEG signals.

Table 1: Summary of network architectures to be explored.

Abbr.	Convolutional/Recurrent	Readout Layer
CNN-FC	Convolutional Neural Network	Fully Connected
CNN-EA	Convolutional Neural Network	Evidence Accumulation
DRN-FC	Deep Recurrent Network	Fully Connected
DRN-EA	Deep Recurrent Network	Evidence Accumulation

For both the CNN-FC and CNN-EA architectures, recurrent layers can directly replace the convolutional layers. This results in two new architectures: DRN-Fully-Connected (DRN-FC) and DRN-Evidence-Accumulation (DRN-EA). Figure 20 depicts the DRN-EA architecture with a sum of log likelihoods readout layer. Our proposed analysis and experiments will involve an in-depth analysis of all four of these network architectures, which are summarized in Table 1.

3.4 Computational Performance

Optimizing the weights of convolutional and recurrent networks is generally a very computationally intensive process due to the size and multilayer architecture of these networks. Since our application requires that these networks can be trained while a BCI user waits and subsequently used in real time, we must pay special attention to computational performance.

Research Question 9: Is it possible to train CNNs and DRNs during a BCI calibration phase? Can they be evaluated in real time?

The standard approach for training neural networks, and the approach that we will follow, is to perform supervised learning using optimization algorithms that leverage first-order error gradients computed using backpropagation. This can be particularly challenging when using deep networks

because they often involve many parameters and because the error gradients can become extremely small as they pass through multiple layers, a phenomenon known as vanishing gradients.

These problems are typically addressed using specialized vector processors, such as general-purpose Graphics Processing Units (GPUs), combined with stochastic gradient descent optimization algorithms that utilize momentum or other heuristics. We believe, however, that we will be able to train our CNNs and DRNs in this setting without using specialized hardware. This will be achieved using several implementation strategies and algorithms that are able to exploit several properties that are specific to this type of problem. Although our implementation is written entirely in the interpreted Python programming language, we are able to access highly tuned linear algebra libraries through the use of the Numpy package. Specifically, we utilize Intel's Math Kernel Library (MKL) in order to achieve fast matrix multiplication performance on commodity CPU hardware. Since our convolutions are equivalent to TDE, as previously noted, we are able to reduce the convolution operation into a fast matrix multiply. Numpy also also permits flexible striding and manipulation of array indices, allowing us to avoid costly memory copies of our data matrices.

Since the datasets involved in BCI applications are relatively small, at least when compared to image datasets, we are also able to fit all of our data and gradients into volatile memory on commodity hardware. This allows us to use batch optimization algorithms, rather than stochastic algorithms. In particular, we have found that algorithms that approximate second-order gradient information, such as Scaled Conjugate Gradients (SCG) [82], yield fast convergence, even on deep and recurrent networks. Optimization algorithms that avoid using the magnitude of the gradient, such as Resilient Backpropagation (RProp) [83], may also be successful at avoiding problems with vanishing gradients. We are confident that a combination of these techniques will allow us to train and evaluate CNNs in a real-time setting. It may still prove challenging, however, to train DRNs and to perform automatic hyperparameter selection on a per-subject basis. A thorough analysis of the various hyperparameters involved in these networks will be required in order to determine if each parameter can be set to a general value for each potential BCI user.

4 Experimental Procedures

In order to evaluate and analyze the research questions outlined in the previous section, we will perform a number of offline experiments using prerecorded data as well as a series of online experiments that will help us to evaluate the ability of these methods to be used in an actual real-time BCI system.

4.1 Colorado EEG and BCI Laboratory

All of our algorithms and experiments will be implemented within the Colorado EEG and BCI Laboratory version 3 ($CEBL_3$) software platform, which is currently under development by our research group. $CEBL_3$ is written largely in the high-level Python programming language and is designed to support rapid prototyping and end-to-end support for all phases of BCI research and development [84]. Researchers can begin their work in $CEBL_3$ by performing a number of tasks offline using built-in routines for data manipulation, signal processing, visualization and machine learning. Successful methods can then easily be promoted to a fully functional graphical user interface that provides real-time support for a number of EEG acquisition devices as well as a number of BCI widgets and modules. Figure 21 shows a sample screenshot of the $CEBL_3$ graphical user interface.

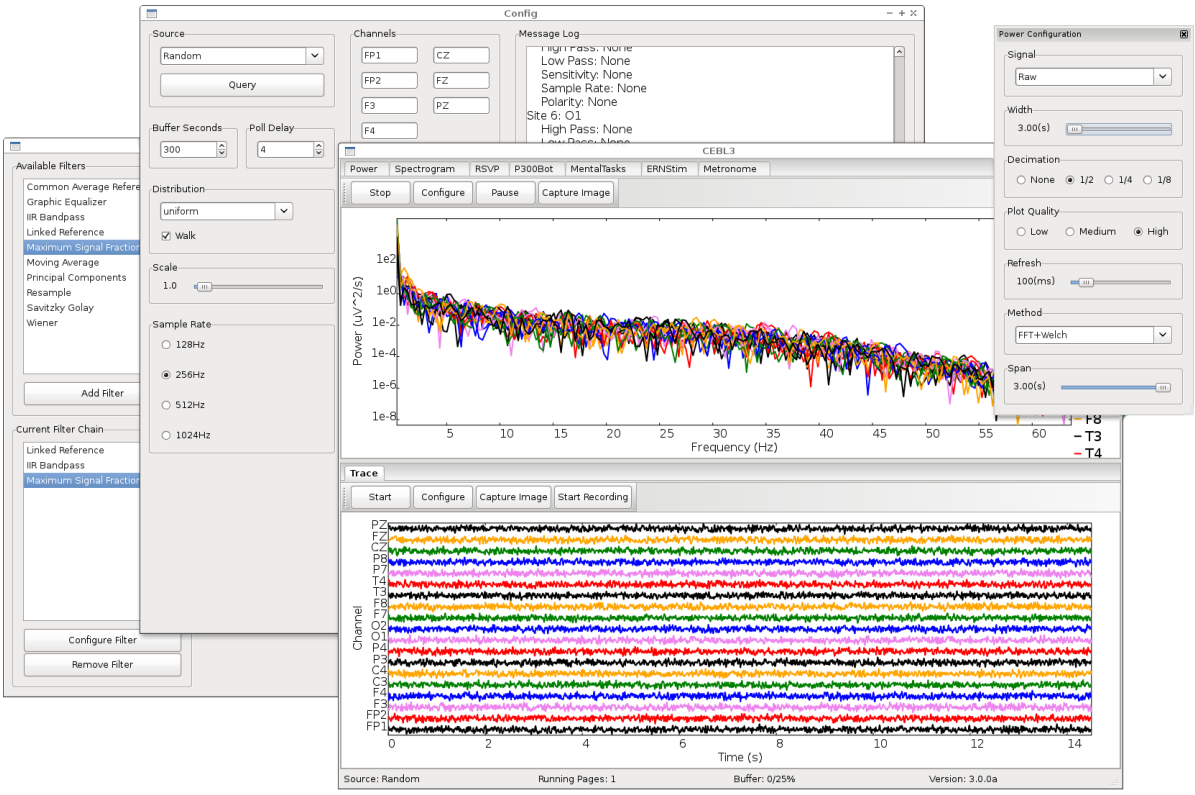


Figure 21: A screen capture of the CEBL$_3$ graphical user interface.

4.2 Offline Dataset

Our initial experiments and analysis will be performed using a publicly available[4] dataset that was recorded by our research group in 2012. This dataset includes EEG recorded from 13 participants. The first nine 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 [85, 86]. The remaining four participants all had severe motor impairments due to high-level spinal cord injury, traumatic brain injury or progressive multiple sclerosis. 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 and the other because all trials were not completed.

All data were recorded using the g.MOBILab+ EEG acquisition system with g.GAMMASys active electrodes, manufactured by Guger Technologies. 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. We chose to place the eight channels at sites F3, F4, C3, C4, P3, P4, O1 and O2, shown in Figure 3b, in order to cover a broad area of the cortex for each hemisphere of the brain.

Each participant was seated comfortably in front of 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. These tasks, summarized in Table 2, consisted of: silently singing a favorite song, imagine repeatedly making

[4] Available for download at `http://www.cs.colostate.edu/eeg/main/data/2011-12_BCI_at_CSU`

Table 2: Mental tasks used and cues shown to the participants.

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

a left-handed fist, visualizing a cube rotating in three dimensions and silently counting backward from 100 by increments of three. Although these tasks were chosen with the intent of utilizing different regions of the brain, we concede that research regarding the most effective mental tasks is ongoing and may vary considerably among individuals [36, 46]. Each participant then 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 segments with a one-second overlap, yielding 25 segments per mental task and a total of 100 EEG segments per participant.

In order to achieve a fair performance evaluation using this relatively small dataset, we use a stratified and nested cross-validation procedure. In this scheme, each trial serves as the test partition for a separate model trained over a training partition consisting of the remaining four trials. The final test results that we present will be the average test performance across this five-fold cross-validation. Each of our classifiers will use at least one hyperparameter for the purpose of regularization, i.e., to prevent overfitting. In order to tune this parameter automatically, we perform a nested cross-validation over the four trials in the training partition. This means that a cross-validation is performed for hyperparameter selection for each of the five possible test partitions. We believe that this procedure will give us a good estimate of generalization performance because each trial will be included and excluded from the training, validation and test partitions.

4.3 Real-Time Experiments

Since a BCI system is ultimately intended to be used interactively, we believe that it is important to evaluate new methods in a real-time setting. To this end, we plan to perform a series of experiments where participants will use our proposed classifiers interactively. These experiments will use a "pie-menu" interface that was previously proposed by our research group [55]. In this interface, shown in Figure 22, a circular menu is presented to the user with a separate section for each mental task. A single section of this menu will then be highlighted in order to instruct the user which mental task to perform. As the classifier assigns labels to EEG segments, a bar will grow toward the corresponding label. Once the bar reaches the pie-menu, an instruction will be selected and the menu will reset. This procedure will allow us to quantify interactive performance in the presence of real-time feedback. The presence of feedback, which is present in usable BCI systems, may allow the user to adapt their mental strategies in ways that cannot be analyzed in offline experiments using prerecorded data.

We plan to recruit 5–10 participants for our real-time experiments. Each participant will use both a baseline classifier and one of our proposed classifiers. Since human resources for performing real-time experiments are limited, we will explore only one CNN and one DRN architecture. These architectures will be determined by our offline analyses and should achieve reasonable computational performance in addition to high classification performance.

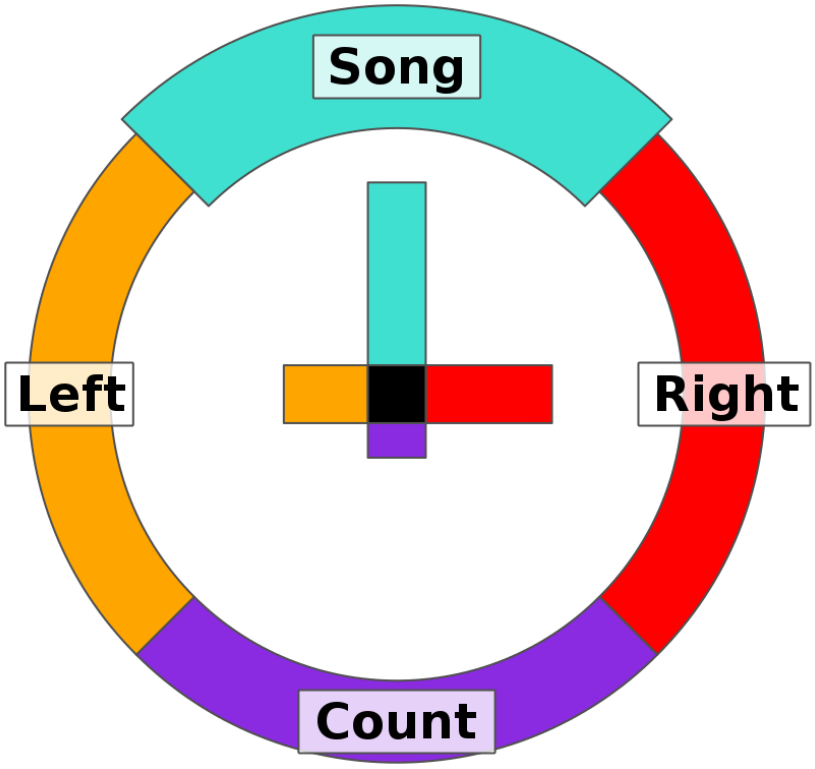


Figure 22: The pie-menu interface in $CEBL_3$ for evaluating real-time performance of mental task classification. First, a section of the menu is highlighted, instructing the user which task to perform. In this case, the user is performing the *song* task. While the user is performing the task, a bar grows toward the label assigned by the classifier, providing real-time feedback. Once the bar reaches a menu section, the menu resets and the procedure is repeated for a new task.

4.4 Baseline Classifiers

In order to compare our proposed methods to the state of the art, we will examine handful of classifiers that utilize PSD representations. Approaches that use PSDs were among the first to be proposed for classifying EEG in MT-based BCIs and, to date, have been some of the most successful [33, 34, 44, 59].

We employ Welch's method for estimating the PSD of each EEG segment [61]. In this approach, depicted in Figure 23, the segment is first subdivided into windows of width ω, called the span. As is common practice, we use Hann windows with a 50% overlap. The DFTs of the windows are then computed and the first half is discarded because the DFT of a real-valued signal is symmetric. The magnitude of the DFTs are then computed, yielding a time-invariant periodogram of each window. These periodograms are then scaled to $\mu V^2/Hz$ and averaged together. Finally, the PSD is normalized using the base 10 logarithm.

The span, ω, in Welch's method is a tunable parameter that controls smoothing, frequency resolution and dimensionality reduction. Note that the DFT of a real-valued segment of length T has $\frac{T}{2}$ values. This means that small values of ω will yield a PSD with low frequency resolution, i.e., few frequency bins, and more smoothing because a large number of small DFTs are averaged together. As ω approaches the width of the segment, on the other hand, the frequency resolution increases and the amount of smoothing due to averaging is reduced. The span parameter also controls the dimensionality of the representation because lower resolution PSDs result in fewer features. In Section 5.2.1, we will explore how varying ω affects the performance of our classifiers.

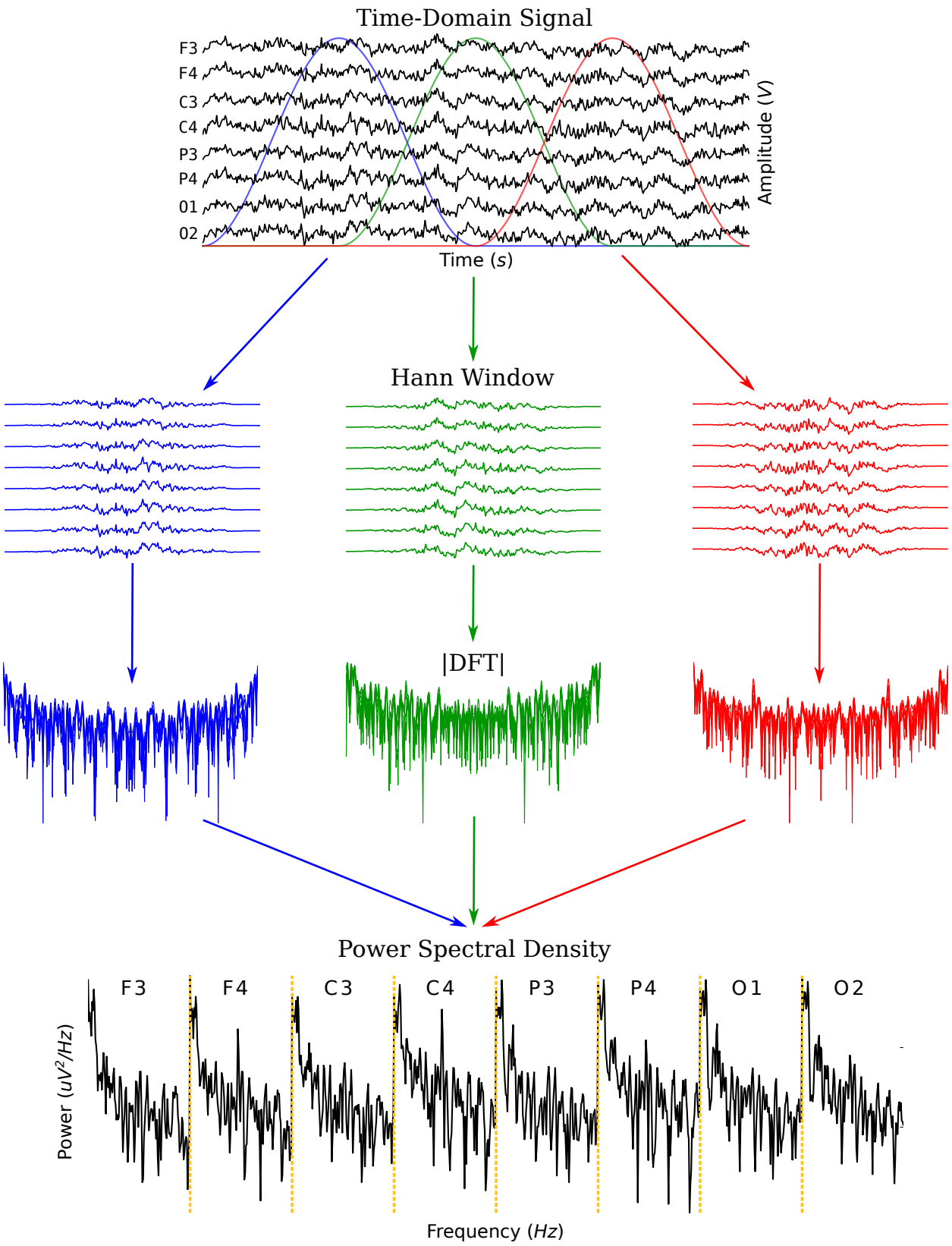


Figure 23: Welch's method for estimating a PSD. First, the EEG segment is split into Hann windows with 50% overlap. The magnitude of the DFT is then computed for window. Finally, the DFTs are scaled and averaged together. This yields an estimate of the power of the signal across the frequency spectrum. The PSDs for all channels are then concatenated together to form a single feature vector for the EEG segment.

In addition to adjusting the span of Welch's method, we have also found two preprocessing steps that often improve classification performance in this setting. Once the PSD of a segment has been computed, frequency ranges can be easily discarded. We have found that discarding frequencies below 0.5Hz and above 40Hz generally improves performance. We suspect that this is due to the artifacts found in these ranges. Artifacts related to movement of the EEG equipment are often found below 0.5Hz while artifacts related to muscle movement are very pronounced above 40Hz. It is important to note, however, that important information related to EEG activity might also be found in these frequency ranges, potentially making this a sub-optimal strategy for artifact removal. We have also found that using a Common Average Reference (CAR) improves performance for some participants. CAR subtracts the mean of all channels from each individual channel at each time step so that the mean across channels is zero. CAR can be viewed as a simple decorrelation filter or spatial high-pass filter since it attenuates activity that is common across all channels. Our experience with PSD-based classifiers has demonstrated that removing these common signal components generally increases the separability of the PSDs across mental tasks and increases classification accuracy. Further exploration is required, however, in order to fully characterize the effects of this type of filter.

In order to construct feature vectors that are appropriate for classification, we simply concatenate all channels of the PSD into a single vector, as shown in the bottom of Figure 23. Since PSDs, and therefore our feature vectors, are time invariant, any number of standard algorithms may be then be used to perform classification. We will explore the use of five different classification algorithms, summarized in Table 3. In previous studies, generative classifiers that model the PSDs using Gaussian distributions, such as Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA), have been noted to be effective [33,34]. It has also been noted, however, that more sophisticated nonlinear methods may yield good performance [34]. As a result, we will also explore the use fully connected Feedforward Neural Networks (FNN) with a single layer containing 30 hidden units and a softmax readout layer. For the sake of completeness, we will also examine two standard classifiers that should generally be tried for most machine learning problems: Linear Logistic Regression (LGR) and K-Nearest Neighbors (KNN) with Euclidean distance. For each of these algorithms, we will tune a single regularization parameter, listed in Table 3. Although there are typically a number of methods for regularizing a given classifier, the methods that we have selected are quite common. A detailed analysis of each of these algorithms remains to be performed and may provide valuable insights into the advantages and disadvantages of each method.

Table 3: Summary of PSD baseline classifiers.

Abbr.	Name	Regularization Parameter
LDA	Linear Discriminant Analysis	Covariance matrix shrinkage
QDA	Quadratic Discriminant Analysis	Covariance matrix averaging
KNN	K-Nearest Neighbors	Number of neighbors
LGR	Linear Logistic Regression	Early-stopping
FNN	Feedforward Neural Network	Early-stopping

4.5 Deep Networks

In order to evaluate each of our deep network architectures, we will use the same datasets and evaluation procedures that we have described for our baseline classifiers. In order to conform with our hypothesis that convolutional and recurrent layers can automatically learn feature extractors and filters, however, we will generally not perform any preprocessing on the data when using our CNNs or DRNs. Unless otherwise stated, we will use raw and unfiltered EEG signals when evaluating our proposed network architectures.

We will also regularize our CNNs and DRNs during our nested cross-validation using a single hyperparameter, similar to our baseline approach. Specifically, we believe that early-stopping will prevent our networks from overfitting by preventing large weights from developing and, as a result, limiting the amount of nonlinearity in our models. If this does not prove to be effective, however, we may also consider using L2 or L1-norm weight decays or dropout techniques [67, 87].

Of course, the complexity of our models will also be partially determined by various other hyperparameters involved with these networks, i.e., convolutional widths, number of layers, number of neurons in each layer, choice of transfer function and pooling method. We intend to explore the effect that each of these parameters has on classification performance and the patterns learned by the networks. In order to maintain a fair evaluation of test performance, these analyses will only use mean validation performance, reserving test performance for our final results. Since there are a relatively large number of hyperparameters to explore, we will begin by performing a broad random search in order to find suitable initial values.

5 Preliminary Results

Now that we have described our research questions and laid the foundations for our experimental methods, we will present a series of preliminary experiments that will demonstrate the feasibility of our approach. First, we will use several contrived experiments to demonstrate that our networks are, in fact, capable of capturing some of the most important types of patterns that we have discussed. Specifically, we will show that CNNs and DRNs can form hierarchical representations of spatiotemporal patterns while achieving time invariance. We will also examine the impulse response of these networks and show that they can be interpreted as learned filters, at least in some circumstances.

We will then proceed by using our proposed network architectures to classify the EEG signals found in our offline dataset. These experiments will show that minimally tuned CNNs using raw EEG signals perform comparably to highly tuned baseline classifiers using extensively preprocessed EEG signals. Overall, our preliminary results will provide a proof-of-concept and support our claims that CNNs and DRNs can automatically learn to represent and filter asynchronous EEG signals in a way that is useful for classification.

5.1 Artificial Problems

Our first set of experiments involve a series of artificially generated signal classification problems. These problems are designed to provide minimal examples that demonstrate the ability of CNNs and DRNs to classify asynchronous and multivariate signals. The simplicity of these problems will allow us to perform an in-depth analysis of the network responses at each layer and to gain a number of insights into the types of patterns that each network learns to identify.

5.1.1 The Staggered Impulse Problem with CNN-EA

In order to demonstrate that our networks are able to achieve time invariance while also capturing spatiotemporal patterns, we have devised a simple classification problem that we refer to as The Staggered Impulse Problem (SIP). This classification task involves two signal classes, Class-A and Class-B, both of which have two channels, s_1 and s_2. The data consist of 50 segments per class that each contain 60 time steps. The first $\frac{1}{2}$ of the data, 25 segments per class, are used for training while the remainder are reserved for testing and analyzing our models.

Each signal segment in Class-A is zero everywhere except for a single unit impulse in each channel. The timing of the impulses in Class-A differs across channels by 10 time steps. Each signal segment is then shifted in time by a constant that is drawn from the random uniform distribution of integers, $\bar{U}$, between 10 and 50. Precisely, the first channel for each segment from Class-A can be written as

$$s_1^A(t) = \begin{cases} 1, & \text{if } t = i \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

and the second channel can be expressed as

$$s_2^A(t) = \begin{cases} 1, & \text{if } t = i + 10 \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

for $t = 1, \ldots, 60$ and where $i \sim \bar{U}(10, 50)$.

The segments in Class-B are also zero everywhere except for a single impulse in each channel; however, the difference in timing across channels now differs by a single time step instead of ten. The first channel for each segment in Class-B can be described as

$$s_1^B(t) = \begin{cases} 1, & \text{if } t = i \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

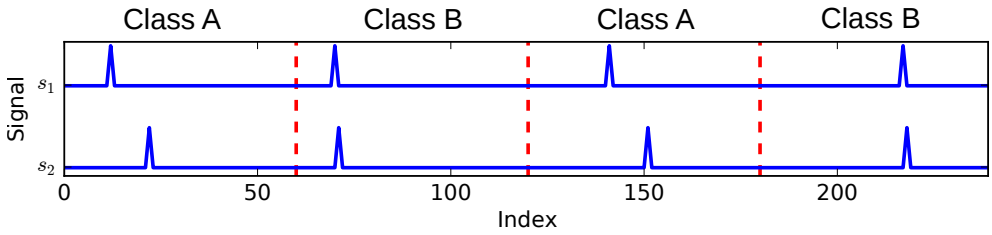
and the second channel can be expressed as

$$s_2^B(t) = \begin{cases} 1, & \text{if } t = i + 1 \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

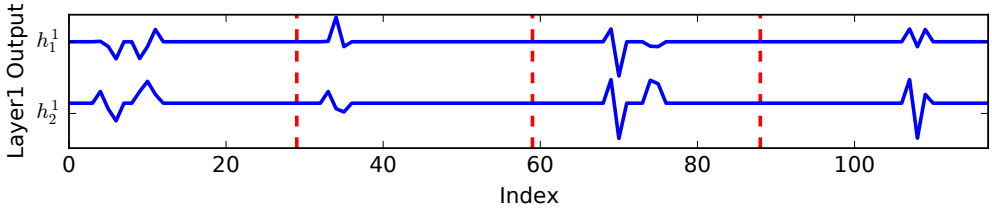
In order for a classifier to distinguish between segments belonging to these two classes, it must be able to capture spatial patterns across channels as well as temporal patterns across a localized period of time. In other words, it must be able to identify whether the impulses in the two channels differ by a single time step or by 10 time steps, regardless of when the impulses begin. Note that this problem is undersampled because there are two channels and 60 time steps per segment but only 25 training segments per class. As a result, a standard classifier cannot typically solve this problem by simply flattening each segment into a $2 \cdot 60 = 120$ dimensional feature vector. An effective classifier must instead learn to ignore absolute timing and focus on localized patterns and relative timing.

CNN-EA can consistently solve this problem with 100% test classification accuracy. Although CNN-EA is fairly robust to hyperparameter selection in this particular case, we have found that a network with two convolutional layers consisting of two neurons in the first layer, four neurons in the second layer, a convolutional width of four time steps in both layers and 2:1 downsampling after each layer can reliably solve this classification problem.

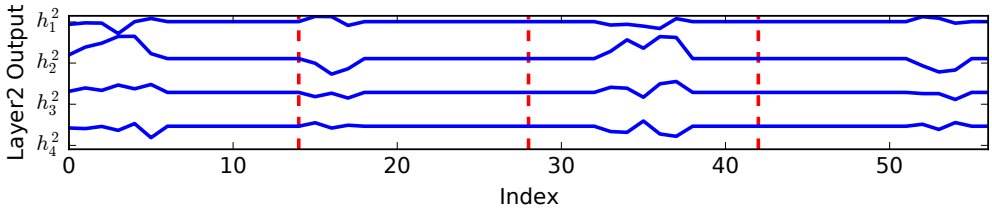
In Figure 24, we see an example of the outputs at each layer of a network that is trained to solve SIP. In Figure 24a we see a trace of signal that alternates between instances of Class-A and Class-B.



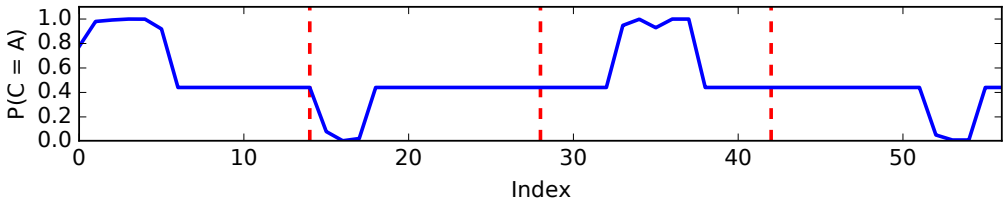
(a) Segments from both classes concatenated across time.



(b) Trace of outputs at Layer-1.



(c) Trace of outputs at Layer-2.



(d) Class Probabilities at each time step.

Figure 24: Test outputs at each layer of a CNN-EA network that has been trained to solve SIP. (a) Two segments from Class-A and two segments from Class-B concatenated together in an alternating fashion. For Class-A the timing of the impulses across channels is offset by 10 time steps. For Class-B, the timing of the impulses across channels is offset by a single time step. (b) The outputs for each neuron in the first layer. There are only subtle differences across the neurons and Class-B evokes a larger excitation following the second impulse. (c) The outputs for each neuron in the second layer. The network responses are now more smooth and vary considerably between the two classes. (d) The estimated probability that the signal belongs to Class-A at each time step. The network reliably assigns class probabilities around the time that the impulses occur.

Although our network is trained over individual segments in practice, testing our network over this alternating signal helps us to understand the differences between the network's response for each class. In Figure 24b, we see traces showing the output of each neuron in the first layer. Note that the output generated by a given neuron is labeled as h_n^l where n is the index of the neuron and l is the index of the layer. Also, notice that the scale of the horizontal axis has changed as a result of our 2:1 downsampling. In this layer, there are only subtle differences in response across the two neurons. There are notable differences, however, between instances of Class-A and Class-B with Class-A generating a larger secondary excitation following the second impulse. In the second layer, shown in Figure 24c, the responses become smoothed and drawn out while the differences across neurons becomes more pronounced. This suggests that each layer learns features that are progressively more distinct for each class. Finally, Figure 24d shows the probability estimated by the network that the class label, C, is equal to the class label A, written compactly as $P(C = A)$. Note that this network estimates a high probability that the label is A surrounding the impulses found in instances of Class-A and a probability near zero around the impulses in instances of Class-B. In regions that are outside the memory capacity of the network, which is limited by the convolutional widths and downsampling, the network assigns a %50 probability for both classes.

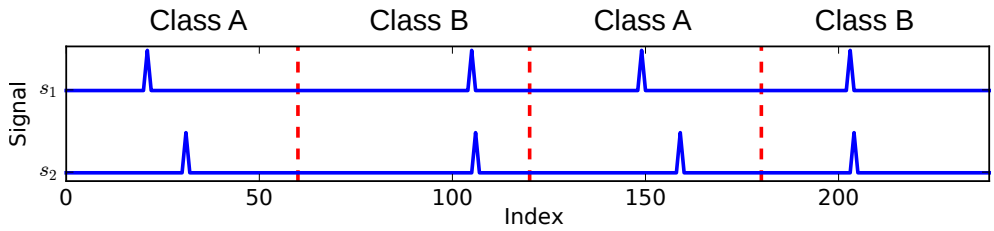
The limited duration of the responses generated by each layer, which are most clearly visible in Figure 24d, are a direct result of the Finite Impulse Response (FIR) characteristics of convolutional layers, previously discussed in Section 3.1. This property has the advantage of giving the network stability. In other words, the network responses settle to a constant value when the network receives a long enough sequence of constant inputs. On the other hand, FIR leads to an inherently limited memory capacity. Even after our network has been presented with an input sequence that definitively designates the segment as belonging to one class or another, it is only capable of outputting the corresponding responses and class labels for a limited period of time.

This experiment demonstrates that CNN-EA is capable of capturing localized spatiotemporal patterns while also achieving time invariance. This network is able to achieve this feat using an undersampled training set and despite the fact that more traditional classifiers with flat feature vectors are typically unable to solve this problem. Our observations about the responses generated at each layer demonstrate the FIR characteristics of convolutional layers and support our claim that the use of multiple layers encourages the network to learn hierarchical representations.

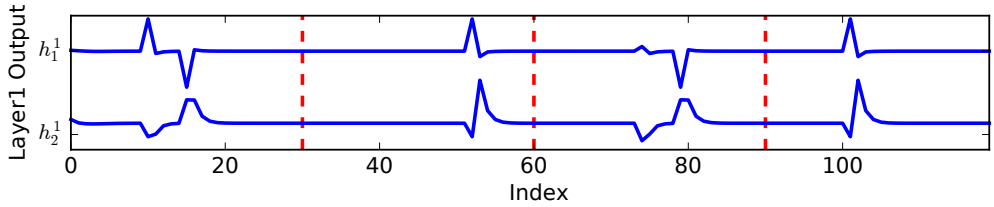
5.1.2 The Staggered Impulse Problem with DRN-EA

Next, we will explore the use of DRN-EA to solve SIP. This experiment will demonstrate that multilayer networks with recurrent layers are also capable of capturing spatiotemporal patterns and achieving time invariance. To our knowledge, this is the first demonstration of a network like DRN-EA for signal classification. This experiment will also highlight the difference in impulse response between convolutional and recurrent layers.

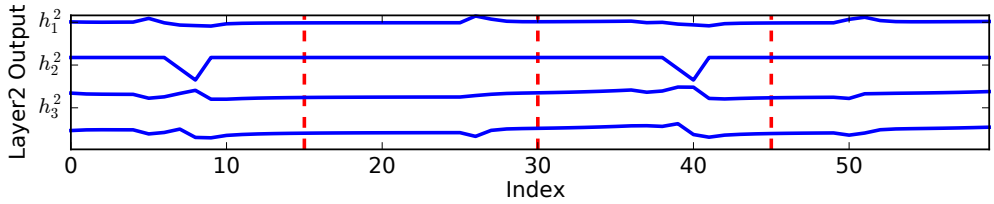
We have found that DRN-EA can consistently solve SIP with 100% test accuracy when configured with two recurrent layers where the first layer contains two artificial neurons, the second layer contains four neurons and with 2:1 downsampling between layers. This level of performance is comparable to the results that we obtained in the previous section with CNN-EA. It is important to note, however, that DRN-EA has a considerably larger computational burden due to the recursive nature of the forward pass and the gradient unrolling required by the backward pass. Nevertheless, our recurrent networks have fewer free parameters because they do not have connections to all time steps across a convolutional window. In these examples, CNN-EA has 59 free parameters while DRN-EA has only 43.



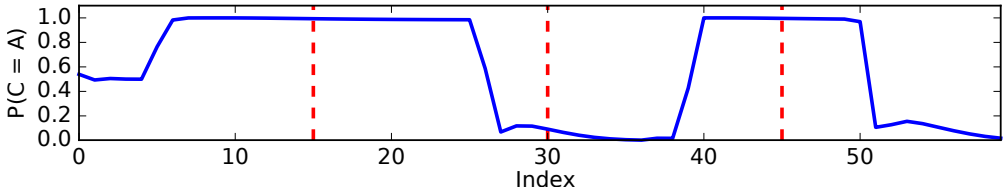
(a) Signal trace for instances of both Class-A and Class-B.



(b) Trace of outputs at Layer-1.



(c) Trace of outputs at Layer-2.



(d) Class Probabilities at each time step.

Figure 25: Test outputs at each layer of a DRN-EA network that has been trained to solve SIP. (a) Two segments from Class-A and two segments from Class-B concatenated together in an alternating fashion. For Class-A the timing of the impulses across channels is offset by 10 time steps. For Class-B, the timing of the impulses across channels is offset by a single time step. (b) The outputs for each neuron in the first layer. While this layer appears to largely mimic instances of Class-A, a large excitation with opposite signs across neurons is elicited following the second impulse in instances of Class-B. (c) The outputs for each neuron in the second layer. The network responses are now more smooth and contain regions with slow drifts or changing offsets. Individual neurons also appear to respond more strongly to instances of a single class. (d) The estimated probability that the signal belongs to Class-A at each time step. The probability alternates between the class labels as soon as the corresponding pattern of impulses is encountered.

In Figure 25, we see an example of the outputs at each layer of a DRN-EA network trained to solve SIP. In Figure 25a, we again see traces of the signal segments belonging to each class. In Figure 25b, we see the output of both artificial neurons in the first recurrent layer. Note that the responses at this layer are similar to the responses generated by our CNNs, with only subtle differences across the two neurons and the primary difference between classes being a larger secondary excitement following the second impulse in Class-A. Although the secondary excitation has opposite signs between the two neurons in this case, this property is not consistent across different random weight initializations. In Figure 25c, we see the network responses generated by the second recurrent layer. Note that the excitations following the impulses are now smoothed together and that the second neuron appears to respond only to the second impulse in segments belonging to Class-A while the other neurons have a strong response to segments from Class-A and a more slight response to segment from Class-B. Also, note that several regions of these outputs contain gradual drifts or constant offsets that change following the input of an impulse. This is in contrast to the outputs of our CNNs, which tended to settle back to a predictable constant offset. Finally, Figure 25d shows the estimated probability that the class label is *A*. The network begins with a 50% probability for each class and then jumps near 100% following the impulses from Class-A and near zero after encountering the impulses from Class-B. Unlike the probabilities generated by our CNNs, this output changes when the next pattern of impulses is encountered, rather than brief window surrounding the patterns.

These observations allow us to draw several conclusions. First of all, DRN-EA is able to reliably solve SIP by learning a time-invariant model that captures localized spatiotemporal patterns. Second, we believe that this experiment supports our assertion that multilayer recurrent networks are able to learn hierarchical representations. In this case, it appears that the first layer largely mimics instances of Class-B while producing a large and distinct excitation following the second impulse in instances of Class-A. The second layer, on the other hand, appears to hold longer-term state information while also producing increasingly distinct responses following the presentation of an instance of either class.

Perhaps the most notable difference between the outputs generated by our CNNs and DRNs is the duration of the responses following a given pattern. Specifically, the outputs of the second recurrent layer contain slow drifts and long-lasting offsets and the class probabilities produced by the final layer persist until a new input pattern is presented. These observations highlight the Infinite Impulse Response (IIR) characteristics of recurrent layers that we previously discussed in Section 3.2. From this example, it is clear that IIR may be advantageous because the network can incorporate longer-term information and indefinitely output the appropriate class label once the network has become sufficiently confident in its decision. While this effect even persists across segment boundaries in this example, it is straightforward to reinitialize the state of the recurrent layer after each segment in order to clear information encountered in previous segments. The primary disadvantages of IIR appear to involve slower learning due to vanishing gradients and potential instabilities [80].

5.1.3 The Three Frequencies Problem

The next artificial experiment that we have constructed is somewhat more realistic in that it consists of continually oscillating signals rather than a brief pattern of impulses. The primary goal of this experiment is to demonstrate the ability of CNNs to isolate relevant frequencies and to validate our assertion that convolutional layers can be interpreted as learned filters.

This experiment consists of three univariate sinusoids that are superimposed in different combinations to form two classes, Class-A and Class-B. We refer to this classification task as The Three Frequencies Problem (TFP). The first of these sinusoids,

$$f_1(t) = \sin(\pi t), \tag{8}$$

was chosen to have unit amplitude and a relatively slow frequency of $\frac{1}{2}$Hz. This signal component will be common to both classes and is, therefore, not relevant for classification. The other two signal components,

$$f_2(t) = 0.3\sin(20\pi t) \tag{9}$$

and

$$f_3(t) = 0.3\sin(10\pi t), \tag{10}$$

have a smaller amplitude of 0.3 and oscillate more rapidly, with frequencies of 10Hz and 5Hz respectively. Segments belonging to Class-A are constructed by applying a constant phase shift to both f_1 and f_2 and then summing these components together. Each segment from Class-A can be precisely expressed as

$$s^A(t) = f_1(t+\theta) + f_2(t+\theta) \tag{11}$$

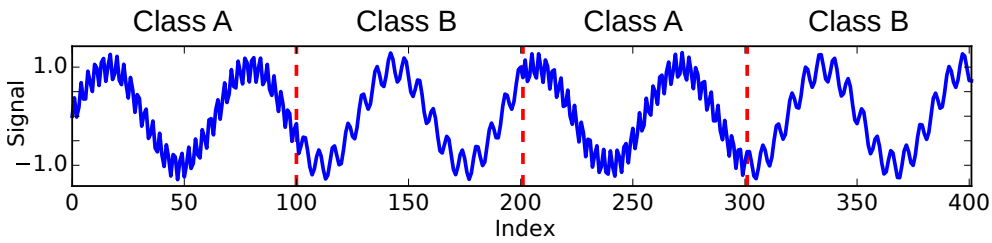
where $t = 1,\ldots,100$ and $\theta \sim U(0,2\pi)$, where U is the continuous random uniform distribution. Similarly, each segment from Class-B is constructed from f_1 and f_3 and can be written as

$$s^B(t) = f_1(t+\theta) + f_3(t+\theta). \tag{12}$$

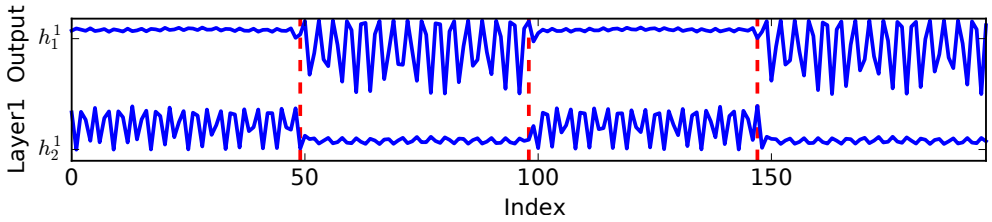
Note that the unique phase shift applied to each segment necessitates time invariance. Also, our classifiers are trained using 25 example segments from each class, again yielding a somewhat undersampled training set.

We have found that a two-layer CNN-EA network with two neurons and a convolutional width of eight in the first layer, four neurons and a convolutional width of four in the second layer and 2:1 downsampling between layers can reliably solve this problem with 100% accuracy. Although not described in detail here, our other network architectures are also able to solve this problem with similar configurations. In Figure 26, we see sample test outputs at each layer for a CNN-EA network that was trained to solve TFP. In Figure 26a, we see a trace of an example input signal. While both signals contain the slower f_1 component, only Class-A contains the faster f_2 component and only Class-B contains the f_3 component. Note that while each segment in our training set contained a random phase, here we allow the signal to seamlessly transition from Class-A to Class-B in order to allow us to characterize how the network responses change as the signal transitions between classes. In Figure 26b, we see the output responses generated by each neuron in the first layer. Notice that both neurons completely remove the f_1 component. Neuron h_1^1 appears to attenuate the f_2 component and pass to the f_3 component while neuron h_2^1 appears to attenuate the f_3 component and pass the f_2 component. Also notice that while the frequencies passed by h_1^1 appear to roughly match the f_3 component, the frequencies passed by h_2^1 are slightly slower than the f_2 component. This suggests that some frequency distortion is taking place as the signal passes through h_2^1, which is actually required because of our 2:1 downsampling. In Figure 26c, we see the network responses generated by each neuron in the second layer of our network. The responses in this layer largely remove the remaining oscillations and roughly capture either the upper or lower envelope of the responses generated by a neuron in the previous layer. It is clear to see how these responses could be combined in a linear way in order to produce a correct class label. In Figure 26d, we see the estimated probability that the class label is A at each time step. These probability estimates are almost perfectly correct.

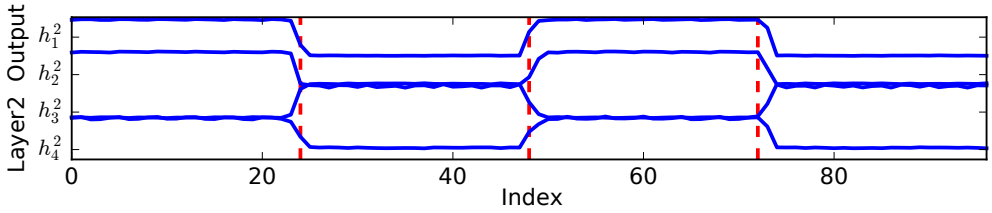
In this experiment, we have demonstrated that each artificial neuron in a convolutional layer can learn to isolate different frequencies that are relevant for classification. These neurons can be viewed as learned bandpass filters. Care must be taken when analyzing these filters, however, because they may produce and rely upon various kinds of linear and nonlinear distortion. We have also shown that subsequent layers can learn to respond to the presence or absence of these filtered signals by capturing the



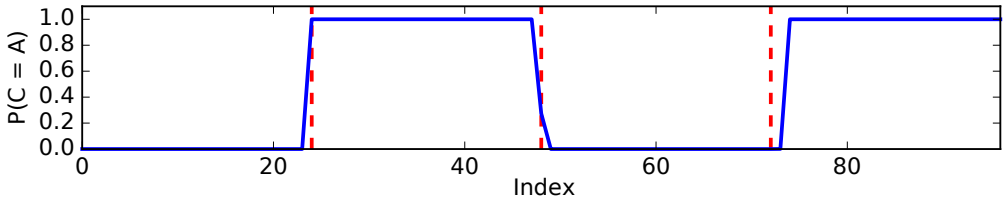
(a) Signal trace for instances of both Class-A and Class-B.



(b) Trace of outputs at Layer-1.



(c) Trace of outputs at Layer-2.



(d) Class Probabilities at each time step.

Figure 26: Test outputs at each layer of a CNN-EA network that has been trained to solve our first artificial classification problem. (a) Two segments from Class-A and two segments from Class-B. The segments are concatenated in an alternating fashion and the networks is applied to the entire signal continually in order to highlight the difference in outputs between classes. (b) The outputs for each neuron in the first layer. There are only subtle differences across the neurons and Class-B invokes a longer-lasting excitation. (c) The outputs for each neuron in the second layer. There are now considerable differences across time and channels. (d) The estimated probability that the signal belongs to Class-A at each time step. The network reliably assigns class probabilities around the time that the impulses occur.

envelope of the responses generated by the previously layer. A softmax layer can then combine these responses in a straightforward way to generate the corresponding class membership probabilities. Although not described in detail here, we have also observed similar results using our fully connected and recurrent network architectures.

5.2 Offline EEG Classification

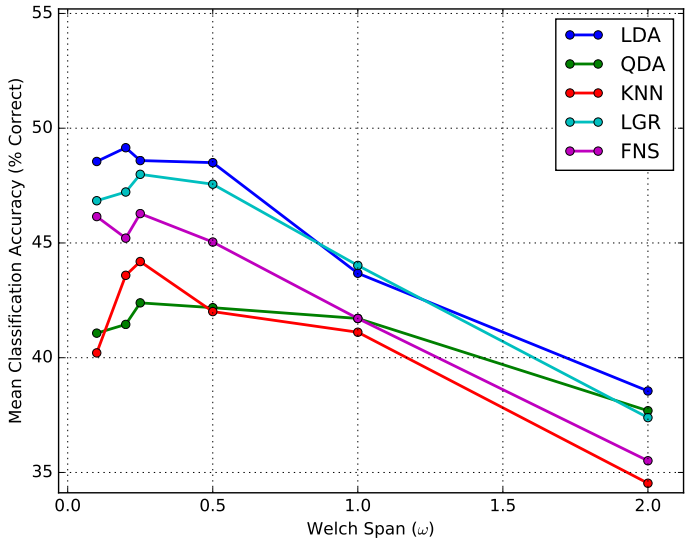
Now that we have performed several basic experiments using artificially generated signals, we will proceed by performing a series of preliminary experiments using actual EEG signals. These experiments will using our offline mental-task dataset that was previously described in Section 4.2. The goal of these experiments is three-fold. First, we will establish a solid baseline level of performance using our PSD-based classifiers. These methods are similar to approaches that have performed well in the current literature and are intended to represent the state of the art. Second, we will demonstrate a proof-of-concept by showing that a CNN with minimally tuned hyperparameters and using raw EEG signals performs comparably to our highly tuned baseline classifiers. Finally, we will show the network responses at each layer of a trained CNN and posit that these responses can be used to gain new insights into the patterns found in EEG signals recorded during imagined mental tasks.

5.2.1 Baseline Classifiers

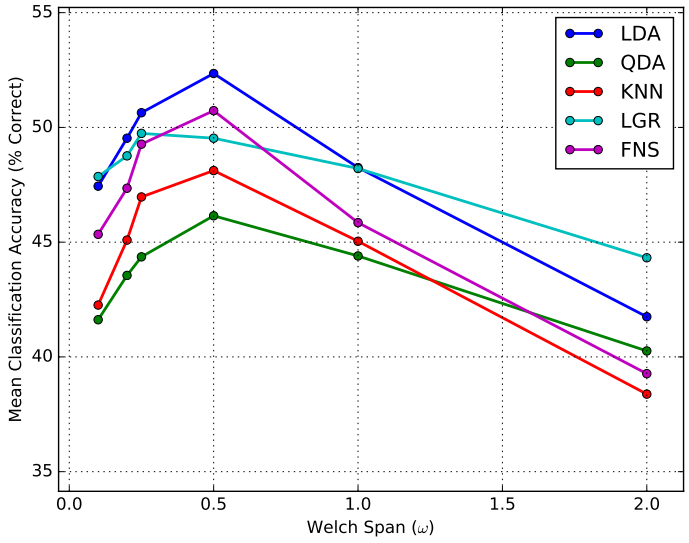
Previously, in Section 2.1.1, we described how PSDs can be used to construct time-invariant signal representations. In Section 4.4, we then described how these representations can be used to construct classifiers for EEG signals. Next, we will apply these methods to our offline dataset with the intent of establishing a benchmark level of performance. We will also explore the effects and importance of varying the different preprocessing and hyperparameters involved. Finally, we will also examine some of the patterns learned by these classifiers and discuss how they tend to vary across subjects.

Of central importance to PSD-based methods is the width of the span parameter, ω, used by Welch's method. Recall that larger values of ω result in higher dimensionality and frequency resolution while smaller values of ω result in reduced dimensionality and increased smoothing in the frequency domain. In Figure 27a, we see how our mean validation accuracy, averaged across all participants, changes for each of our baseline classifiers as ω is varied from 0.1–2 seconds. Since the segment length is two-seconds, $\omega = 2$ yields the raw periodogram produced without Welch's method. Also, note that our regularization parameters, which will be discussed shortly, are individually tuned or each data point. In this experiment, smaller values of ω, between 0.1–0.25 seconds, typically yield the best performance. This suggests that that Welch's method improves generalization performance by reducing noise and the dimensionality of the PSD. We also see that our linear models, LDA and LGR, typically achieve slightly better classification accuracies than the other methods. This may suggest that noise and undersampling continue to be problematic, even after applying Welch's method, and that the additional simplicity provided by linear models helps to prevent overfitting.

In Section 4.4, we described the use of a Common Average Reference (CAR), which may reduce noise and artifacts by attenuating signal components that are common to all channels. In Figure 27b, we see how our mean validation accuracy varies with ω when CAR is applied as a preprocessing step. Incorporating CAR increases our peak classification accuracies by approximately 4%. We also see that our best-performing value of ω has now increased to about $\frac{1}{2}$-second and that our nonlinear FNS model now slightly out-performs LGR at this value of ω; although LDA still yields the best classification accuracies overall. Together, these observations suggest that CAR is effective at reducing the noise



(a) Mean validation accuracy versus ω without CAR.



(b) Mean validation accuracy versus ω with CAR.

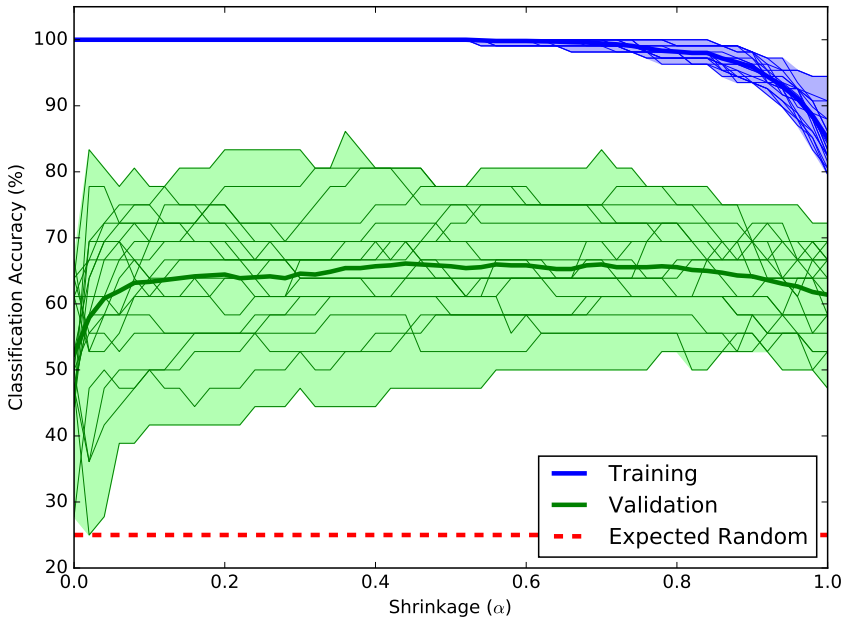
Figure 27: Mean validation accuracy averaged across all participants for all baseline classifiers as Welch's span, ω is varied. (a) Without CAR preprocessing, the peak value of ω is typically between 0.1–0.25s. Across classifiers, peak performance ranges from 40% with QDA to 49% with LDA. (b) With CAR preprocessing, the peak value of ω is increases to about 0.5s. Across classifiers, peak performance improves to a range of about 46% with QDA to 52% with LDA.

in the signal and improves the situation with respect to overfitting by reducing our reliance on Welch's method and purely linear models.

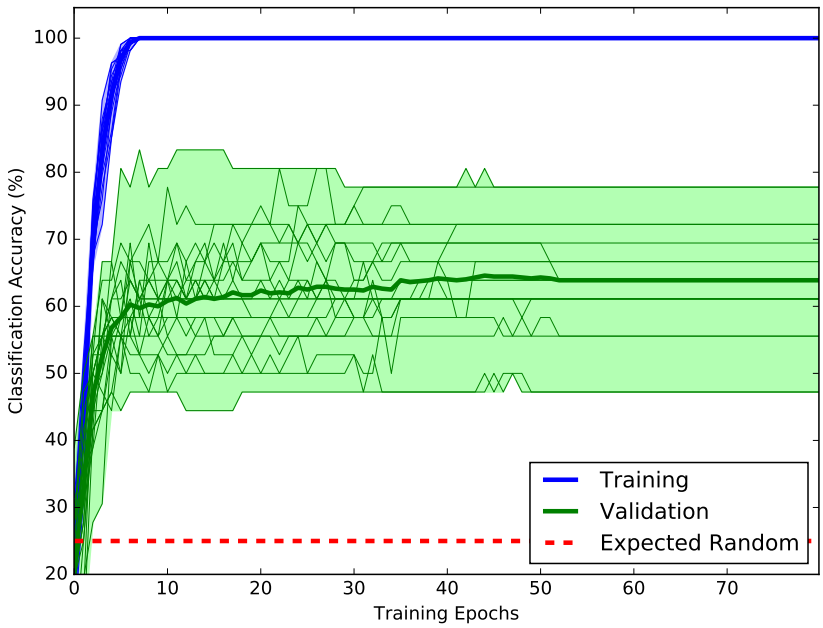
In Sections 4.2 and 4.4, we also described the use of a nested cross-validation procedure to select a single regularization hyperparameter for each of our baseline classifiers. These parameters further control the complexity of our models in a finely tuned and user-specific way and may help prevent them from fitting noise and irrelevant patterns in the training data. In Figure 28, we see two examples of how our regularization parameters affect training and validation accuracy for Subject-6 with $\omega = 0.5$. In Figure 28a, we see how shrinkage affects the generalization performance of our LDA classifier. Note that each thin line represents a single fold in our cross-validation while the thick line is the average across all folds. LDA is a linear statistical classifier that utilizes the average class covariance matrix, i.e., the variance between each pair of values in the PSD, as well as the class means for each dimension of the PSD. For shrinkage values near zero, the covariance matrix is unaltered and the variance of each dimension is taken into account. For shrinkage values near one, on the other hand, the covariance matrix becomes a scalar multiple of the identity matrix and only the class means are utilized. The introduction of a small shrinkage value, between 0–0.1, appears to increase validation accuracy sharply. This is likely because shrinkage helps condition the covariance matrix of our undersampled problem to be invertible, a necessary step for computing the LDA weight matrix. Validation accuracy then continues to improve gradually until a shrinkage value of about 0.5 is reached. Training accuracy remains near 100% for shrinkage values less than 0.5 and then gradually begins to decrease. These observations suggest that even our linear classifiers are somewhat prone to overfitting and that a moderate amount of regularization can help improve generalization performance by increasing the classifiers dependence on the means of the PSD.

In Figure 28b, we see a similar plot that shows how our training and validation accuracies vary with the number of training epochs for our nonlinear FNN classifier. Since the initial weights are chosen to be small and uniformly distributed, the network outputs tend to be relatively linear and homogeneous early in the training procedure. As training progresses, some weights may grow to increase the network's dependence on individual features and the outputs in the hidden layer may also reach the nonlinear regions of our hyperbolic tangent transfer function. By terminating the training procedure before this occurs, we can effectively limit the complexity of our models and prevent overfitting. In this instance, we see that training accuracy quickly reaches 100% in less than 10 training epochs. Meanwhile, the validation accuracy sharply increases between 0–10 epochs and then gradually increases until about 45 epochs, after which it typically converges. This suggests that early-stopping is not a particularly effective method for regularizing these models. Given that even a linear model is capable of overfitting our PSD representations, we suspect that FNNs are able to quickly learn an essentially linear class boundary that separates the training set.

Now that we have explored the various hyperparameters that are involved in our baseline classifiers, we present the final test classification accuracies for each method and each participant in our offline dataset. Table 4 summarizes these results with best performing model for each participant shown in bold and with separate sections for the participants with no impairments in the laboratory and for the participants with motor impairments in their homes. Across all subjects, the mean test accuracies for our baseline methods vary from about 46–52% correct. Individual accuracies range from about 28–69% correct. It is important to note that, for a four-task problem, the expected classification accuracy for a random classifier would be 25%, suggesting that our classifiers are correctly assigning labels to many of the EEG segments. Although this level of performance is less than ideal, we believe that many of these users would be able to successfully control a BCI system. Since our approach assigns a class label every second, a user should be able to control a slowly moving actuator or cursor with 50% classification accuracy.



(a) Classification accuracy vs. shrinkage for LDA.



(b) Classification accuracy vs. training epochs for FNN.

Figure 28: Training and validation accuracies during our nested cross-validation procedure for Subject-11. The results from individual folds are shown with thin lines while the mean across all folds in shown in bold. (a) For LDA, validation accuracy peaks at about 0.5, around which time the training accuracy also begins to fall. (b) For FNN, training and validation accuracy quickly increase during the first 10 epochs. Validation accuracy then gradually increases until around 45 epochs, after which there is little change.

Table 4: Test classification accuracies for each participant and baseline classifier. Results are averaged over all test folds in the nested cross-validation. The best performance for each participant is shown in bold.

	Participant	LDA	QDA	KNN	LGR	FNN
No Impairment	1	**66.67**	55.56	58.33	56.11	60.00
	2	43.33	**43.89**	**43.89**	40.00	43.33
	3	**42.22**	37.22	40.00	37.78	37.78
	4	63.89	59.44	56.11	61.67	**67.78**
	5	47.78	**51.67**	44.44	**51.67**	40.56
	6	**68.89**	62.22	58.89	60.00	63.33
	7	**47.78**	**47.78**	44.44	40.56	46.67
	8	**60.00**	49.44	53.33	56.67	59.44
	9	**56.11**	25.56	53.89	50.00	50.00
Impairment	10	**42.78**	40.56	41.11	37.78	39.44
	11	68.89	66.11	64.44	65.00	**69.44**
	12	28.33	25.00	32.22	**42.78**	36.11
	13	43.89	35.56	34.44	43.89	**45.56**
	Mean	**52.35**	46.15	48.12	49.53	50.73

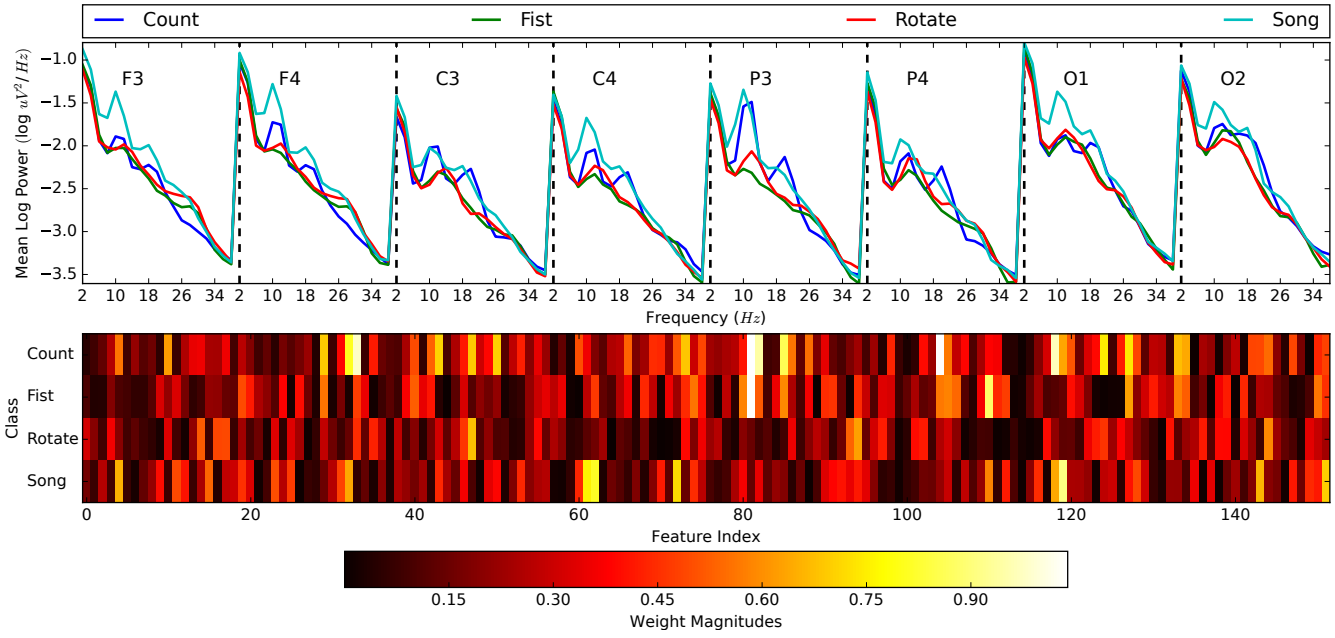


Figure 29: PSD features and trained LDA weights for Subject-6. (top) The log PSD with $\omega = 0.5$ averaged across all segments within each class. The features are concatenated into a single vector and passed to the subsequent classifier. (bottom) The weight magnitudes for a trained LDA classifier indicate the relative important for each feature and class.

Visualizing the PSDs along with the weights learned by the classifier may also lead to valuable insights. The top of Figure 29 shows the class means of our PSD features for Subject-6. This can help us to visualize the differences in power across mental tasks. For example, note that the *Song* task yields high average power in the α range (8–16Hz) at sites F3, F4, C4, P3, O1 and O2 while the *Count* task yields high average power in the α range only at sites F4, C3, C4 and P3. The bottom of Figure 29 shows the weight magnitudes of a trained LDA classifier. A large weight magnitude below a given feature indicates that the feature is important for labeling the corresponding class, show to the left. For instance, the large weight magnitudes for the *Count* and *Fist* tasks near the α range at site P3, indicate that these features are important for separating these tasks from the remaining two.

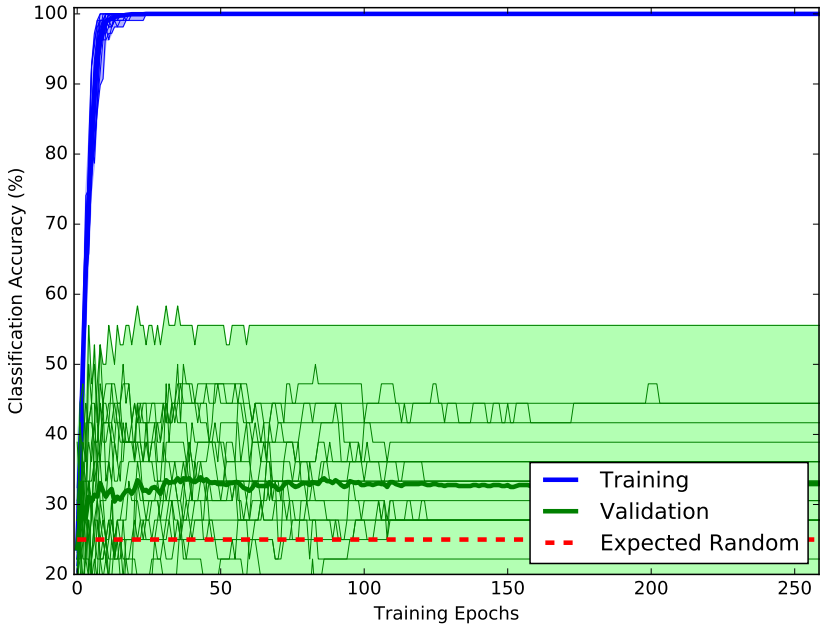
The mean PSDs and the weights learned by our classifiers vary considerably among subjects, meaning that the plot shown in Figure 29 looks quite different for each participant. Also, the differences across classes can not always be easily visualized from the averaged PSDs, even for subjects and tasks that achieve relatively high classification accuracies. These observations support the use of user-specific models and reinforce the notion that machine learning algorithms are able to identify patterns that are not easily identified by visual inspection.

A number of conclusions can be drawn from the experiments we have conducted with our baseline classifiers. First of all, there appears to be considerable variability across subjects, both with respect to the levels of performance attained as well as with respect to the patterns in the PSDs that are elicited by each mental task. There also appears to be some variability across classification algorithms, with LDA outperforming the other classifiers by 1–6% on average; however, a larger dataset and thorough statistical analysis would likely be required to draw firm conclusions about the differences in performance across all of these classifiers. In any case, it does appear that careful tuning of the various preprocessing steps and hyperparameters involved is extremely important, perhaps even more so than the choice of classifier. This observation supports our claim that these approaches require involved preprocessing and tuning procedures. Finally, we have established a benchmark level of performance that we believe is representative of the state of the art.

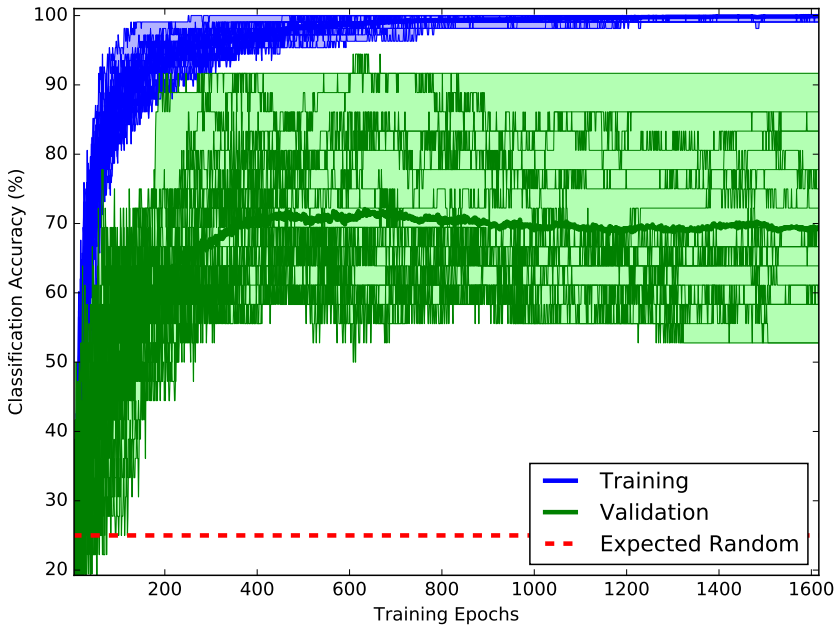
5.2.2 Convolutional Networks

Now that we have established and analyzed our baseline classifiers, we continue by exploring the application of our CNN architectures to our offline dataset. The primary goal of these experiments is to demonstrate that CNNs are capable of forming representations of raw EEG signals that are useful for classification. Although we will perform a brief analysis of the outputs of a trained network at each layer, we will reserve a more sophisticated analysis of the patterns that are learned by these networks for a later date. A thorough examination of the network hyperparameters and experiments using our DRN architectures have also not yet been completed.

In order to roughly establish network hyperparameters that are suitable for use in our preliminary experiments, we performed a series of pilot experiments that evaluated the average validation accuracy across all subjects for 100 different CNN-EA configurations. These experiments included a variety of configurations that seemed reasonable, including networks with 1–3 convolutional layers, 1:1 and 2:1 downsampling and various numbers of neurons and convolutional widths in each layer. From these experiments, we have found that networks with two convolutional layers and 2:1 downsampling between layers typically produce the best validation accuracies. In particular, we have found that networks with a first layer consisting of 16 neurons with a width of 11 and a second layer consisting of 8 neurons with a width of 9 generally perform well. For the experiments conducted here, we use CNN-EA networks with this configuration and CNN-FC networks with the same configuration except that they include an additional five hidden neurons in the fully connected readout layers.



(a) Classification accuracy vs. training epochs for CNN-FC.



(b) Classification accuracy vs. training epochs for CNN-EA.

Figure 30: Training and test Classification accuracies during CNN training for Subject-11. (a) For CNN-FC, training accuracy quickly reaches 100% around 25 epochs. Validation accuracy also rises for about 45 epochs before leveling off near 33%. In this case, overfitting appears to be problematic and early-stopping is unable to sufficiently regularize the model. (b) For CNN-EA, training accuracy rises much more slowly and does not achieve 100% until beyond 1,500 epochs. Validation accuracy rises until about 500 epochs before leveling off at about 70%. Overfitting is better controlled when using CNN-EA with early-stopping

In order to regularize our individual CNN networks, we will use an early-stopping procedure that is similar to what we previously used with our FNN baseline classifier. In addition to being a relatively standard method for regularizing neural networks, early-stopping also has a computational advantage over other methods because the network only needs to be trained once per fold. Regularization techniques that require a separate parameter to be tuned typically must be retrained for each possible value of the parameter. This computational advantage becomes important with our CNN architectures because they can take several minutes to train for each fold and, of course, interactive BCIs must maintain a reasonable training time.

In Figure 30, we see examples of how the training and validation accuracies progress during the training procedure for Subject-11. For CNN-FC, shown in Figure 30a, training accuracy rapidly increases to 100% within the first 25 training epochs. Similarly, the validation accuracy increases during the beginning of training and peaks around 45 epochs, at which point it levels off at about 33%. The fact that the network quickly reaches 100% training accuracy while the validation accuracy has risen only a small amount suggests that this network rapidly over-fits the training data. Unfortunately, early-stopping is largely unable to prevent this. As discussed in Section 3.3, we believe that this rapid overfitting is primarily the result of the large number of parameters found in the fully connected readout layer of these networks. Note that this CNN-FC configuration contains a total of 7,447 free parameters with 4,863 of these parameters in the readout layers alone. We believe that this results in the readout layers of the network learning to separate the training data using large-scale patterns in the EEG segments before the convolutional layers are able to learn localized patterns and a suitable time-invariant representation. In other words, our CNN-FC architecture roughly achieves the same result that we would expect from flattening the EEG segments into a single feature vector and passing them to an FNN classifier. It is possible that a much larger training set, such as those found in other applications for CNNs, might slow the process of overfitting and allow the convolutional layers to learn effective representations. It is not feasible, however, to collect such large training sets during a BCI calibration phase of reasonable duration.

In Figure 30b, we see how the training and validation accuracy progresses for CNN-EA. In this case, the training accuracy requires approximately 250 epochs to reach 90% accuracy and does not reliably achieve 100% accuracy until after 1,500 training epochs. The validation accuracy rises until achieving over 70% accuracy at about 500 epochs, after which the validation accuracy falls very gradually. The training procedure for CNN-EA clearly progress much more slowly than CNN-FC and the peak validation accuracy reaches a much higher peak. In this case, early-stopping also terminates the training procedure around 500 epochs, which yields a slight improvement in accuracy as opposed to allowing the network to run until convergence. These observations suggest that CNN-EA is more robust to overfitting than CNN-FC. Note that our CNN-EA networks contain a total of 2,611 free parameters with only 24 of those parameters in the readout layer. This reduction in free parameters in the readout layers increases dependence on the convolutional layers and, in turn, encourages the network to emphasize localized patterns and time-invariant representations.

Now that we have established a suitable network configuration and explored the effects of our regularization procedure, we are prepared to present our preliminary test classification accuracies. Table 5 shows our mean test classification accuracies for each participant, separated between those with and without impairments, for our LDA baseline classifier along side our CNN-FC and CNN-EA networks. The final row shows mean performance across all subjects for each method. These results demonstrate that CNN-FC performs quite poorly, achieving a mean accuracy across subjects near the 25% level that we would expect from a classifier that assigns random class labels. CNN-EA, on the other hand, performs comparably to our LDA baseline classifier, achieving a mean accuracy of 50% across subjects while LDA achieves 52% accuracy. Although our CNN-EA networks do not achieve a higher mean test

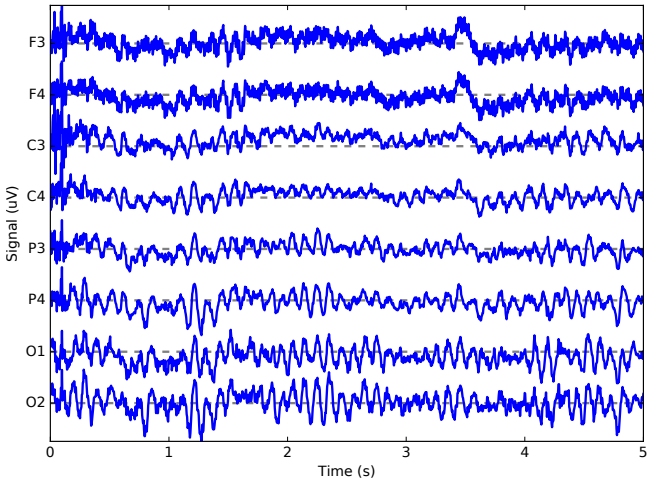
Table 5: Test classification accuracies for our CNN-FC and CNN-EA networks along with our LDA baseline classifier. Results are averaged over all test folds. The best performance for each participant is shown in bold.

	Participant	LDA	CNN-FC	CNN-EA
No Impairment	1	**66.67**	19.44	55.00
	2	43.33	24.44	**73.89**
	3	**42.22**	20.56	28.33
	4	63.89	32.22	**66.67**
	5	**47.78**	21.11	35.56
	6	**68.89**	30.56	54.44
	7	**47.78**	20.56	43.89
	8	60.00	20.00	**63.33**
	9	**56.11**	24.44	33.33
Impairment	10	42.78	24.44	**50.00**
	11	68.89	31.11	**75.56**
	12	28.33	31.11	**34.44**
	13	**43.89**	22.78	37.22
	Mean	**52.35**	24.83	50.13

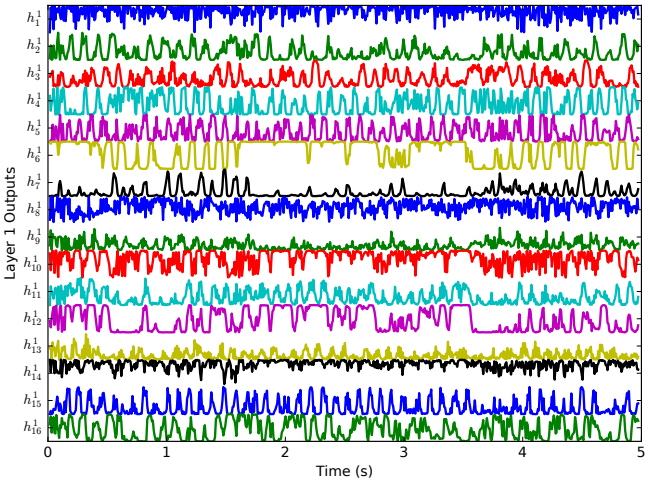
accuracy than LDA when averaged across subjects, we are unable to conclude a statistically significant difference in means when using a paired t-test ($p = 0.57$). It may be important to note, however, that some individual subjects achieve considerably higher test accuracy with CNN-EA than with LDA. For instance, Subject-2 achieves 74% accuracy with CNN-EA but only 43% with LDA, which is a 34% improvement. Furthermore, the highest two individual test accuracies, for Subject-2 and Subject-11, were both achieved with CNN-EA. These are also the only test accuracies that were above 70% for any of the classifiers that we have examined thus far. It is also important to reiterate that our CNN networks use the raw EEG data without any preprocessing or filtering. This supports our claim that CNNs are able to automatically learn effective representations without relying on manually engineered solutions.

Determining the types of patterns that our networks learn to identify is central to our hypothesis that CNNs and DRNs are able to automatically learn to filter and process EEG signals and, potentially, leverage information that may be discarded by other methods. As a first step toward this goal, Figure 31 shows trace plots of a single EEG segment along with the corresponding network responses at each layer. These plots were generated using a CNN-EA network that was trained to classify our offline EEG data for Subject-11. In Figure 31a, we see a trace plot of a five-second test EEG segment recorded during the *Count* task. Note that there is a brief artifact that is likely caused by muscle movement near zero seconds and an artifact that is likely caused by ocular movement near the 3.5-second mark. There also appears to be some 60Hz interference caused by power mains in channels F3 and F4. This segment also appears to contain a relatively large amount of activity in the α (8–16Hz) frequency range, which has especially high amplitude during the intervals from 0–2.5 and 4–5 seconds.

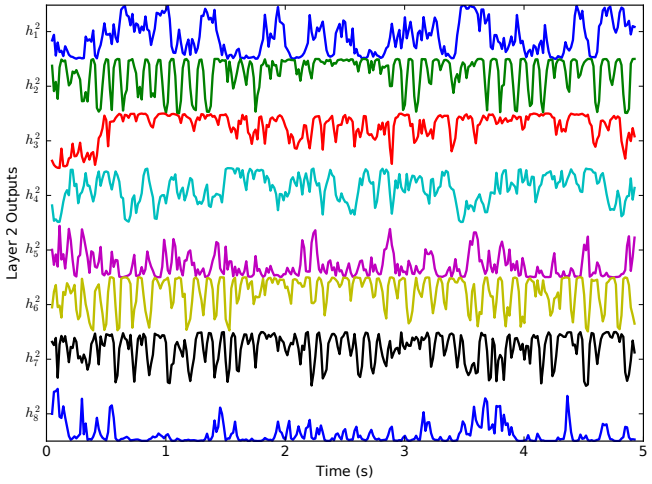
In Figure 31b, we see trace plots of the network responses generated by each artificial neuron in the first convolutional layer. Notice that neurons h_1^1 and h_8^1 appear to contain a large amount of high-frequency information relative to the other responses. This may indicate that these neurons have learned to isolate some of the 60Hz interference or other high-frequency information. Also, note that none of the responses in the first layer appear to respond strongly to the muscle or ocular artifacts in the EEG signals, suggesting that the network has learned to reject these artifacts. Neurons h_6^1 and



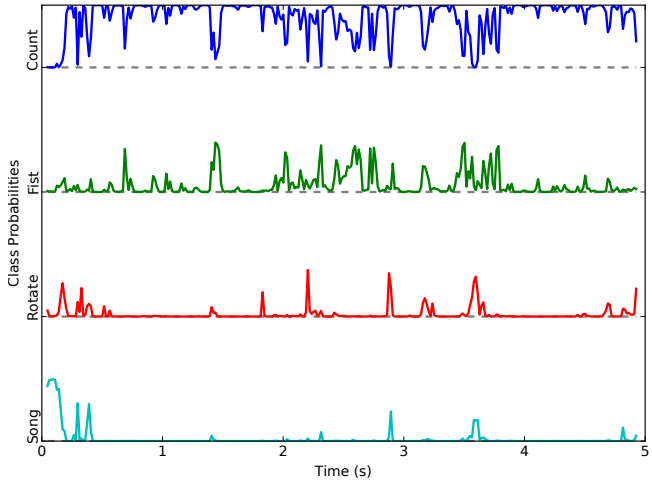
(a) Sample EEG segment recorded during the *Count* task.



(b) Network responses produced by the first layer.



(c) Network responses produced by the second layer.



(d) Class probabilities produced by the readout layer.

Figure 31: Trace plots of network outputs at each layer for a CNN-EA network trained to classify EEG data for Subject-11. (a) A five-second test EEG segment recorded during the *Count* task. Note the muscle movement artifact near zero seconds, the ocular artifact near 3.5 seconds and the strong α (8–16Hz) waves between 0–2.5 and 4–5 seconds. (b) Network responses generated by the first layer. Note that h_1^1 and h_8^1 contain large amounts of high-frequency information. Also, h_6^1 and $h_1 2^1$ appear to have very nonlinear responses. Other outputs appear to oscillate near the α frequency range. (c) Network responses generated by the second layer. Note that h_2^2 appears to have a very nonlinear response and h_6^2 and h_7^2 appear to oscillate near the α frequency range. (d) Class probabilities generated by the readout layer. Count task is generally the highest; although there is some notable confusion with the *Fist* and *Song* tasks near the muscle and ocular artifacts.

h^2_{12} tend to switch between positive and negative values in an almost binary fashion. This suggests that these neurons have very nonlinear responses that tend to fall near the saturated regions of our hyperbolic tangent transfer function. Neuron h^1_6 also appears to respond most frequently during the time when α activity is highest in the EEG segment. This might suggest that nonlinear patterns occur during these time periods. Many of the other channels also appear to oscillate near the α frequency range; although they do not appear to respond much differently during the times when the α activity is highest in the EEG segment. Combined, these observations seem to suggest that activity near the α range may be especially important for classifying segments belonging to this task.

In Figure 31c, we see trace plots of the network responses generated by each neuron in the second convolutional layer. Notice that much of the high-frequency information found in the first layer is no longer present. This may partially be caused by the change in scale following downsampling or it may also be due to the network learning to reject 60Hz noise and other high-frequency noise. Neuron h^2_2 appears to have a very nonlinear response that is strongest during the periods when α activity is high, similar to the response in the first layer for neuron h^1_6. Neurons h^2_6 and h^2_7 also appear to respond strongly to α activity.

In Figure 31d, we see the class probabilities that are produced at each time step by our readout layer. Recall that our final class labels are assigned by summing the log of these probabilities across the segment and selecting the class label associated with the highest sum of these likelihoods. Note that the network is generally correct in assigning the *Count* label for most time steps. The *Count* and *Song* tasks appear to be confused, however, during the muscle-movement artifact near zero seconds. There is also some occasional confusion between the other tasks, especially during the ocular artifact near 3.5 seconds. Also, notice that the class probabilities for the *Rotate* and *Song* tasks appear to be highly correlated with the output of neuron h^2_8 in the second layer responses. This suggests that any patterns isolated by this neuron are of particular importance for these tasks.

Examining our network outputs at each layer clearly allows us to make a number of interesting observations and conjectures; however, this process is also somewhat laborious and subjective. In Section 3, we proposed a number of other approaches for examining the types of patterns that are learned by these networks. For example, examining the network weights and attempting to map them between layers and back to the segments or the surface of the scalp may be revealing. Determining the frequency response and other filtering characteristics of our convolutional layers may also clarify the types of patterns that they learn to identify. Examining the distribution of the network responses along with various transfer functions may help to quantify the important of nonlinearities. Despite the wide variety of tools we have at hand, one of the most challenging and important tasks that remains to be accomplished is to draw firm conclusions about the types of patterns that our networks learn to identify and to generalize those conclusions across segments, subjects, tasks and network architectures.

6 Discussion

Designing classification algorithms for asynchronous EEG signals is clearly a challenging task, especially within the context of BCI systems. An effective classifier must act in real time to handle noise, artifacts, undersampling and high dimensionality while also achieving time invariance and capturing sophisticated spatiotemporal patterns. The human brain is an extremely complex and dynamic system and a collection of evidence suggests that EEG signals contain nonlinear and nonstationary patterns at multiple time scales [38,39]. We believe that an effective classifier should be as general as possible and capable of modeling a wide variety of patterns that may be present in EEG signals, even if these types of patterns have not yet been fully characterized. Furthermore, we maintain that it is extremely impor-

tant to be able to interpret the models learned by these classifiers. Engineering and neuroscience are inextricably linked in the field of BCIs and interpretation is important for furthering our understanding of EEG signals and their underlying neurological correlates as well as for guiding the construction of next-generation BCI systems.

Historically, BCIs have addressed these challenges by leveraging clearly defined control signals, e.g, P300 ERPs or μ and β rhythms associated with motor-imagery. For the more general mental-task paradigms, however, there are no known control signals that are consistent across tasks, subjects, participants and sessions. As a result, mental-task paradigms rely more heavily on signal representations and machine learning methods to identify relevant patterns on a per-subject and per-session basis.

We have demonstrated, however, that current approaches often rely on strict prior assumptions and, as a result, may have a limited ability to capture some types of patterns. For instance, signal representations based on PSDs and CWTs rely on assumptions about linearity and stationarity and achieve time invariance by discarding phase information. Similarly, the commonplace CSP-based approaches rely on assumptions about linearity, require the signal to be separated into narrowband components and assume that the component variances contain adequate information. TDE-based approaches, on the other hand, appear to be promising because they are extremely general and rely on few prior assumptions; however, TDE has a limited ability to model longer-term and multiscale patterns.

We have also noted in the literature and demonstrated with our baseline classifiers that current approaches often involve extensive manual engineering in the form of filtering, preprocessing and feature selection. Although these approaches often appear to boost performance, they also rely on prior assumptions or empirical observations and discard information that may, potentially, be useful for classification. For example, noise reduction techniques like linear bandpass filters and CAR eliminate signals that originate from within the brain along with noise and artifacts.

We assert that improved methods for analyzing and classifying EEG signals should rely on few prior assumptions and, instead, automatically learn to represent and identify appropriate patterns. Following recent successes in other areas of machine learning, we posit that multilayer artificial neural networks, known as deep networks, are well suited for this role [41, 42]. Specifically, we believe that variants of the deep convolutional and recurrent network architectures, which have biologically inspired connectivity, may be able to automatically learn hierarchical representations that can be viewed as learned filters and feature extractors.

Along these lines, we have proposed several novel deep network architectures that have convolutional and recurrent connectivity. Unlike the CNNs that are commonly used for image classification, we apply convolution (or recurrence) only across the time axis. This encourages our networks to learn time invariant representations while keeping spatial patterns fixed across the different regions of the brain. Our convolutional layers are otherwise similar to the standard approach and consist of a number convolutional neurons with learned weights followed by downsampling and a transfer function. As we have demonstrated in our analytical assessment and using artificially generated signal classification problems, these convolutional layers are capable of capturing localized spatiotemporal patterns while maintaining time invariance. We have also shown that these convolutional layers can be viewed as learned nonlinear filters and we have offered evidence that stacking multiple layers encourages these networks to learn hierarchical representations.

In CNN-FC, these layers are then followed by fully connected layers that combine information across time steps to assign a final class label for each EEG segment, which is consistent with the standard approach. In CNN-EA, on the other hand, we omit the fully connected layers and, instead, output class membership probabilities for each time step. This information can then be accumulated over a period of time in order to assign a final class label. Omitting the fully connected layers is a relatively novel idea; although it is similar to our previous work with TDE-based approaches and is related to an

architecture that has been proposed for semantic segmentation of images [56, 81]. Since CNN-EA does not contain fully connected layers, it has far fewer model parameters and is encouraged to learn localized patterns in the convolutional layers rather than focusing on any longer-term relative structure that may exist at the scale of the EEG segment. Although further analysis is called for, our preliminary results have shown strong evidence supporting the use of CNN-EA over CNN-FC and suggest that CNN-FC suffers from severe problems with overfitting.

A number of parallels can be drawn between our proposed CNN architectures and methods that are commonly used for filtering, analyzing and classifying EEG signals. FIR bandpass filters utilize convolution across time with a kernel that is determined analytically in order to attenuate a given range of frequencies. This filtering procedure has a variety of uses, including noise removal, preventing aliasing before downsampling and feature selection. Similarly, each artificial neuron in our convolutional layers utilizes convolution of its inputs with a kernel consisting of learned weights before downsampling the result and passing it to the transfer function. As a result, we believe that it is appropriate to view each convolutional layer as a bank of learned, nonlinear FIR filters. CWTs utilize convolution with wavelets at varying scales in order to estimate the energy content of the signal across time and frequency. This similarity supports the notion that convolutional layers may be well suited for isolating spatiotemporal patterns at multiple time and frequency scales. We have also noted that convolution of a signal across time is exactly equivalent to TDE followed by a matrix multiplication. In fact, our implementation utilizes this property to achieve a computational performance improvement by replacing the convolution operation with a fast matrix multiply. This means that TDE is a special case of CNN-EA where the network consists of a single layer without downsampling. Conversely, CNN-EA can be viewed as a multilayer and multiscale generalization of TDE. As such, we believe that CNN-EA is well positioned to address the shortcomings of TDE.

We have also proposed that recurrent layers may be a suitable alternative to convolution in our deep network architectures. To our knowledge, this approach is a novel use of recurrent layers for signal classification. It is, however, somewhat related to a recent work that uses recurrent layers for image classification [79]. Since recurrent layers contain feedback connections, they are able to process temporal information while only requiring direct input connections to the current time step of the signal, as opposed to a separate connection to each signal value within a window of time. Provided that these recurrent layers are able to achieve sufficient memory capacity with relatively few artificial neurons, this reduced connectivity may lead to fewer model parameters and, therefore, improved generalization performance. Recurrent layers also have infinite impulse response. When working with linear bandpass filters, it is generally accepted that IIR filters achieve better filtering characteristics, i.e., faster rolloff with fewer parameters. This can, however, come at the expense of complexity and possible challenges with stability and nonlinear phase distortion. Although achieving long-term memory with recurrent networks can be challenging in practice, we have demonstrated using our artificially generated signal classification problems that our DRNs can, in some cases, recall an input pattern for a longer period of time and with fewer parameters than a similar CNN. Regardless of any benefits that are achieved by using recurrent layers, we believe that the introduction of these networks and our analysis of their capabilities will be a significant contribution.

The central hypothesis of this research is that our proposed network architectures will be able to automatically learn to represent and classify EEG signals while relying on few prior assumptions and requiring little, if any, preprocessing or manual intervention. To this end, we have outlined a series of research questions that will be explored in order to thoroughly probe the ability of our networks to capture various types of patterns that may be found in EEG signals. In our preliminary results using offline EEG data, we have provided supporting evidence for this claim by demonstrating that CNNs using raw, unfiltered EEG signals perform comparably to highly tuned PSD-based classifiers that utilize

extensive filtering and preprocessing.

Due to the fact that these networks are very general, we are also confident that they will learn to identify patterns in EEG signals that are currently unknown or not well characterized. As a result, we believe that these network may be a valuable tool for analyzing EEG signals and gaining new insights into how the human brain performs various mental tasks. In order to explore this possibility, we have outlined a number of methods and experiments for analyzing our networks and interpreting the patterns that they learn to identify. For instance, DFTs, PSDs and CWTs can all be used to examine the filtering characteristics of each convolutional and recurrent neuron. Exploration of various types of transfer functions, e.g., linear, linear rectified and hyperbolic tangent, may lead to insights regarding the importance of nonlinear patterns in the signals. Carefully examining the weights and outputs of each artificial neuron and attempting to map these patterns back to the scalp may also lead to insights about the types of patterns identified by our networks. In our preliminary results, we have given several simple examples of how these types of analyses may be revealing; however, generalizing these approaches across tasks, subjects and sessions will be considerably more involved.

Certainly, developing improved BCI systems is an important and overarching goal of this research. In our preliminary results using offline data, we have not yet been able to show a statistically significant difference in mean classification accuracy between our CNNs and baseline classifiers when the results are averaged across subjects. We have, however, noted a considerable performance benefit with CNNs for some participants. In fact, the best individual results within both the group of participants with impairments and the group without impairments were achieved using CNN-EA. It is important to note, however, that we have not yet systematically explored our network hyperparameters and configurations. We have also not yet examined the performance of our recurrent architectures and a number of additional possibilities exist for regularizing our networks. In any case, a realistic measure of performance for a BCI system can only be determined in real-time because the users of the system tend to adapt their mental strategies based on the feedback they receive. In order to provide realistic performance comparisons, we plan to evaluate our classifiers in a real-time setting.

We are confident that this line of research will lead to significant contributions to a variety of areas, including machine learning, signal and time series analysis, neuroscience and BCIs. At the very least, we have proposed several network architectures that follow a very general approach to modeling and classifying signals and time series. Since these networks require little manual intervention, they are likely advantageous over other approaches, regardless of any performance benefits. Since these networks rely on few prior assumptions they may be able to harness new types of patterns that have not yet been discovered or fully appreciated. Through an analysis of these models, we may also learn new things about EEG signals and the human brain. In the end, we believe that this work will lead to improved BCI technologies that will, in turn, be used to build assistive technologies that will help to improve the lives of people with disabilities.

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 various snippets that he has enthusiastically shared with me. I would also like to thank my dissertation committee and all of the members of the BCI laboratory. Without your time, feedback and support, none of this work would be possible. This work is 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 my parents, Glen and Nancy Forney, and from my wife, Maggie VanDenBerg, and through the Computer Science Department's Artificial Intelligence and Evolutionary Computation Fellowship. I would also like to thank my father for igniting my passion for science and my mother for passing on her passion for reading, writing and critical thinking. Last, but definitely not least, I would like to thank my wife, children and family for their phenomenal and enduring support of my academic endeavors.

References

[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] Department of Compter Science at Colorado State University. CEBL: Colorado EEG and Brain-Computer Interface Laboratory. `http://www.cs.colostate.edu/eeg/main/software/cebl3`, March 2016.

[15] Center for Adaptive Neurotechnologies. BCI 2000. `http://www.schalklab.org/research/bci2000`, March 2016.

[16] Swartz Center for Computational Neuroscience. BCILAB. `http://sccn.ucsd.edu/wiki/BCILAB`, March 2016.

[17] French Institute for Research in Computer Science and Automation. OpenVibe. `http://www.inria.fr/`, March 2016.

[18] g.tec medical engineering GmbH. intendiX. `http://www.intendix.com/`, March 2016.

[19] g.tec medical engineering GmbH. mindBEAGLE. `http://www.mindbeagle.com/`, March 2016.

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

[45] 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.

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

[47] 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.

[48] 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.

[49] 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.

[50] 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.

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

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

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] 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.

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

[59] 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.

[60] Li Zhang, Wei He, Chuanhong He, and Ping Wang. Improving mental task classification by adding high frequency band information. *Journal of medical systems*, 34(1):51–60, 2010.

[61] 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.

[62] 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.

[63] 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.

[64] Paul S Addison. *The illustrated wavelet transform handbook: introductory theory and applications in science, engineering, medicine and finance*. Taylor & Francis, 2010.

[65] 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.

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

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

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

[69] 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.

[70] Stefan Kremer. On the computational power of elman-style recurrent networks. *IEEE Transactions on Neural Networks*, 6:1000–1004, 1995.

[71] Hava T Siegelmann and Eduardo D Sontag. On the computational power of neural nets. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 440–449. ACM, 1992.

[72] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[73] Sanjit K Mitra and James F Kaiser. *Handbook for digital signal processing*. John Wiley & Sons, Inc., 1993.

[74] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of The International Conference on Machine Learning (ICML)*, volume 30, 2013.

[75] Elliott Forney, Charles Anderson, William Gavin, and Patricia Davies. A stimulus-free brain-computer interface using mental tasks and echo state networks. In *Proceedings of the Fifth International Brain-Computer Interface Meeting: Defining the Future*. Graz University of Technology Publishing House, 2013.

[76] Damien Coyle, Thomas McGinnity, and Girijesh Prasad. Creating a nonparametric brain-computer interface with neural time-series prediction preprocessing. In *28th Annual International Conference of The IEEE Engineering in Medicine and Biology Society*, pages 2183–2186. IEEE, 2006.

[77] Damien Coyle, Girijesh Prasad, and Thomas McGinnity. Extracting features for a brain-computer interface by self-organising fuzzy neural network-based time series prediction. In *26th Annual International Conference of The IEEE Engineering in Medicine and Biology Society*, volume 2, pages 4371–4374. IEEE, 2004.

[78] Damien Coyle, Thomas McGinnity, and Girijesh Prasad. A multi-class brain-computer interface with SOFNN-based prediction preprocessing. In *The 2008 International Joint Conference on Neural Networks (IJCNN)*, pages 3696–3703. IEEE, 2008.

[79] Francesco Visin, Kyle Kastner, Kyunghyun Cho, Matteo Matteucci, Aaron Courville, and Yoshua Bengio. ReNet: A recurrent neural network based alternative to convolutional networks. *arXiv preprint arXiv:1505.00393*, 2015.

[80] 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.

[81] 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.

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

[83] Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *IEEE International Conference on Neural Networks*, pages 586–591. IEEE, 1993.

[84] 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.

[85] 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.

[86] College of Health and Human Sciences at Colorado State University. Brainwaves research laboratory. `http://brainwaves.colostate.edu`, Jan 2014.

[87] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.