UNIVERSITY OF CALIFORNIA

Los Angeles

Deep Learning Applications in Motor Imagery EEG Data Classification

A thesis submitted in partial satisfaction

of the requirements for the degree Master of Science

in Bioinformatics

by

Sarah Joy Madsen

2021

ABSTRACT OF THE THESIS

Deep Learning Applications in Motor Imagery EEG Data Classification

by

Sarah Joy Madsen

Master of Science in Bioinformatics

University of California, Los Angeles, 2021

Professor Corey Arnold, Chair

Brain-computer interface (BCI) technology can return the ability to communicate to those suffering from neurological disease or paralysis. Common BCI systems require their user to constantly attend an external stimulus, which can be exhausting and may not be physiologically possible for some users. BCIs designed around sensorimotor rhythms, like those produced during the imagined movement of a body part, remove the need for a constant external stimulus. A BCI designed around motor imagery data would need to be able to determine the user's intent in the shortest amount of time possible. This study looked at the effect of shortening the training sequence of data on a proven deep learning model's (EEGNet) performance. We show that cutting the amount of data passed to the model by half did not negatively affect the model's performance and suggest this is due to a decay in feature fidelity in the latter half of the data. Additionally, we designed a hybrid convolutional-recurrent neural network (CRNN) optimized to classify short sequences of data. This new model is able to achieve similar results as EEGNet across subjects at 44.98 percent and 49.37 percent accuracies, respectively. Our results suggest a motor imagery-based BCI could determine the user's intent with as little as two seconds of data and pioneer a novel deep learning model that could perform even faster.

The thesis of Sarah Joy Madsen is approved.

Jonathan Kao

William Hsu

Corey Arnold, Committee Chair

University of California, Los Angeles

2021

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Brain-computer interface technology translates neural activity into keyboard, bypassing the need for muscle control. These systems can return the ability to communicate to those affected by neurodegenerative disease or paralysis. BCI systems often record brain activity through electroencephalography (EEG) electrodes placed on the scalp. While the signal fidelity from these electrodes is not as high as from more-invasive measures like electrocorticography (ECoG) or intracortical microelectrodes, they are much more flexible in their use. EEG systems do not require implantation surgery, cover larger surface areas, and are available at a lower cost than invasive methods. To overcome low signal-to-noise ratio (SNR) in EEG data, BCI systems are built around neural components that can be reliably recorded. Additionally, much research has gone into developing feature extraction and signal classification algorithms that can produce accurate results despite noisy signals.

There are two classes of BCI systems, those based on brain signals evoked by external stimuli and those based on spontaneous signals generated by conscious and sub-conscious brain activity [27]. BCIs designed around evoked potentials are popular because they are intuitive to use and do not require any subject pre-training before they can be deployed [25]. However, they often required consistent concentration on a fixed stimulus which can be exhausting for the user [27]. Additionally, the cue and latent brain response inherently set a limit to how fast the user can type. An optimal BCI paradigm allows the user to set the pace of communication and does not require constant attention to a cue.

The second class of BCI systems are those based on spontaneous signals, which can be generated consciously and independent of any external cues [20]. This type of BCI solves the aforementioned issues of cue-based systems. However, spontaneous signals are weak in amplitude and are thus difficult to record with EEG electrodes, resulting in poor feature extraction and low signal classification accuracy [33]. These challenges can be ameliorated by focusing on types of spontaneous signals that can be localized to specific spatial and temporal regions, like the sensorimotor rhythm (SMR) signals such as motor imagery (MI). Still, BCIs designed around SMR signals require advanced feature extraction and classification algorithms.

Designing a signal processing algorithm around the known features of sensorimotor signals can improve its classification accuracy. Traditional EEG processing methods like the Common Spatial Patterns (CSP) and Filter Bank CSP (FBCSP) algorithms require hand-selecting features used in frequency classification [2], [11]. To contrast, deep learning models have been able to match the accuracies achieved by traditional algorithms without the need for hand selecting features [36], [19]. However, deep learning models are notorious for requiring copious amount of training data, and thus a long training period. Ideally, a BCI system could accurately classify neural data with a very short training period. Thus, deep learning models for BCI systems need to be designed with this constraint in mind.

Relevant research in the EEG classification via deep learning methods field has largely been theory-based algorithm development with little focus on how the algorithms could be deployed in real-life machines. To be practical for use in MI-based BCI technology, neural signal processing algorithms need to accurately classify MI tasks from temporally short segments of data. An algorithm that requires long sequences of data to classify the user's intent inherently limits the speed of such a communication device. Thus, a speedy algorithm is essential for quick communication.

1.2 Objectives

The present study investigates the length of data necessary for accurate task classification of motor-imagery data using deep learning algorithms. To analyze the effect of temporally shorter datasets on an algorithm's accuracy, we perform an offline analysis of the BCI Competition Dataset 2a with Lawhern et. al's proposed convolutional neural network (CNN), EEGNet [3], [19]. In analyzing the amount of data required to train a CNN, we also investigate how the neural signal's properties change over time. Finally, we design and test a different deep learning algorithm by splicing together a CNN and a recurrent neural network (RNN), hypothesizing this type of network may be better suited to classifying shorter sequences of data.

Chapter 2

Background

2.1 Evoked response BCI

Evoked potential BCI systems require their user to constantly attend an external stimulus in order to make a selection from a screen. Popular stimuli include auditory tones, visual cues, or somatosensory stimuli [27]. The brain signal evoked by the stimulus is identified by the BCI system and used to determine the keyboard selection of the user.

A popular type of evoked potential BCI is the P300 speller, designed around the P300 brain signal. To elicit the P300 signal the subject is given an expected or familiar stimulus interlaced with a number of unexpected stimuli at random, called the “oddball” paradigm [8], [7], [32]. The P300 signal occurs when the subject is presented the familiar stimulus. The defining features of the P300 signal are four-fold: positive polarity, greater than 275-millisecond latency, distribution around the parietal lobe, and amplitude response that is dependent on the experimental paradigm [6]. The strength of the P300 is proportional to the rarity of the stimulus and inversely proportional to the relevance of the stimulus. Furthermore, the P300 is elicited even in the absence of the subject consciously acknowledging the stimulus through verbal or nonverbal communication. Together, these properties make the P300 a reliable signal for BCI design [8].

P300 spellers capitalize on these reliable properties of the P300 signal. Generally, the alphabet and numbers zero through nine are arranged on a six row by six column grid. The subject is instructed to choose a character from the grid and recall it as each row and column is flashed in a random pattern [8]. Here, the subject's chosen character is the familiar stimulus and the P300 signal is elicited when it is flashed (and only when it is flashed). With proper feature extraction and signal classification techniques the BCI can accurately identify and type the subject's chosen character. Current P300 speller paradigms are able to achieve higher than 90 percent accuracy [18]. However, the P300 speller is slow for communication. Each row and column must be flashed at least twice before the correct character can be identified and there is often additional classification latency. Thus, while accurate, the P300 speller is not practical for real-time communication purposes.

There are other evoked potential BCIs in addition to the P300 speller. The steady-state visually-evoked potential (SSVEP) speller requires its user to stare at a flickering visual stimulus that evoke brain potentials at the same frequency as stimulus [39]. The SSVEP speller achieves a higher communication rate than the P300 speller, but users report the flashing stimuli is annoying and exhausting [27]. Additionally, both the SSVEP speller and P300 speller rely on the subject's eye gaze. In able-bodied subjects, classification accuracy of the P300 speller substantially decreased when the subject did not focus their gaze directly on the target character [4]. Thus, neither of these spellers are good BCIs for subjects with debilitating neuromuscular disorders that affect the oculomotor muscles, like amyotrophic lateral sclerosis (ALS) [15]. The rapid serial visual presentation

(RSVP) speller does not require control of eye gaze as all the stimuli are presented in the center of the screen [1]. The trade-off is a slower typing speed than either the P300 of SSVEP spellers. A more optimized BCI communication system would not require the user's eye gaze and still maintain a high rate of information flow.

2.2 Motor imagery signals

Motor imagery (MI) is a type of sensorimotor rhythm caused by the movement of a body part and falls under the broader category of motor representation: brain signals involved in intending and preparing body movements [14]. While the process of motor representation is subconscious, it is possible to envision a motor task (MI) and thus consciously elicit MI brain signals. Furthermore, MI signals occur in the absence of actual movement and remain intact in those with compromised motor neurons [13], citelotze. The fact that MI signals can be generated consciously and independent of any external stimuli as well as in users with limited mobility make them a powerful EEG signal to base BCIs around.

MI signals have the same spatiotemporal properties as those stemming from action planning, making them easy to localize both spatially and temporally [13]. They are characterized by oscillatory brain activity in the mu (7-13 Hz) and beta (13-30 Hz) bands [25]. MI causes two types of amplitude modulation in these frequency bands: event-related desynchronization (ERD) in which oscillatory amplitudes are suppressed and event-related synchronization (ERS) in which amplitudes are enhanced. These oscillations predominantly arise from the supplementary motor cortex (SMA) and premotor cortex, with partial activation of the anterior primary motor (M1) cortex as well citelotze, [30]. MI signal detection and classification algorithms aim to quantify the percent change in oscillation amplitude from baseline or rest. By focusing on the specific frequency bands and spatial locations affected by ERD or ERS these algorithms can limit noise in the signal and achieve more accurate results. Much of the difficulty in classifying MI signals stems from their intra- and inter-subject variability. As a person practices MI, the amount of ERS and ERD increases as the sensorimotor networks strengthen [27], [34]. Thus, individuals well-trained in MI outperform those who are untrained and a single individual's MI signals will change over time. Additionally, the amount of ERD and ERS experienced during MI tasks varies between subjects so greatly that classification models can only reliably be trained on a single subject at a time [19]. Part of the inter-subject variability can be ascribed to a person's aptitude for performing MI. High quality MI signals are produced when the user imagines the kinesthetic experience of moving their own body part, rather than imaging movement from a third-party point-of-view [20], [13]. The latter does not activate the same sensorimotor rhythms. Some individuals may struggle with this type of imagery and it is impossible to ensure every subject achieves the same level of imagined kinetics.

2.3 Traditional EEG classification algorithms

2.3.1 Common spatial pattern algorithm

Many early classification algorithms for MI focused on isolating the band-power changes that occur in ERD and ERS during MI tasks [43]. Signal decomposition techniques such as Principal Component Analysis (PCA) were used to extract features from the spatial and temporal components in the EEG signal based on their contribution to the signal variance and thus band-power [16], [17]. The Common Spatial Pattern (CSP) algorithm proved to be better than PCA at decomposing EEG signals due to the orthogonality constraints placed on the components in PCA [17].

The CSP algorithm is a feature extraction technique designed to maximize the discrimination between two classes. It does so by constructing spatial filters that maximize the variance of the projected signal for one class and minimize the variance for the other class [43].

For MI task classification, the algorithm treats each EEG dataset as a set of spatial patterns recorded from N electrodes across T consecutive time points [23]. The time axis is used to index different patterns as they occur temporally, thus there is a maximum of T spatial patterns in each EEG trial. CSP finds a decomposition of the two classes of MI tasks that contains components that are common to both groups but optimally suited to discriminate between them. This is done via diagonalization into matrices that closely mirror the covariance matrices [10]. The output is a set of filters called Common Spatial Patterns constituting an ordered list of characteristics that define the best directions for discriminating between MI classes. The first and last directions in the filter bank are the best for distinguishing between the first and second class, respectively. For an example, a MI task that belongs to the first class will score maximally along the first direction and minimally along the second direction if it belongs to the first class and vice versa if it belongs to the opposite class. The same is true for the second and second-to-last directions in the list of spatial patterns which are the second-best directions for discrimination, and so on through the list. Only the best spatial patterns are useful for task classification and many different feature selection techniques exist to choose the patterns that result in the highest classification accuracies.

A successful Common Spatial Patterns filter bank for MI classification requires proper band-pass filtering before applying the CSP algorithm [2]. An accurate frequency band is often subject-specific and needs to be manually selected for the best results [43]. Setting a broader frequency range saves time and computational expense but produces less accurate results [2]. Later iterations of CSP attempt to address this issue [2], [43].

2.3.2 Filter bank common spatial pattern algorithm

The Filter Bank Common Spatial Pattern (FBCSP) algorithm was developed by Kai Keng Ang et. al. and designed specifically to address the subject-specific frequency range issue found in CSP [2]. There are four progressive steps to the FBCSP algorithm: bandpass filtering, spatial filtering with the CSP algorithm, feature selection and classification.

FBCSP employs frequency filtering before the feature extraction step to improve the quality of the CSP features. The first step of the FBCSP algorithm the EEG data is bandpass filtered into multiple frequency bands with a zero-phase Chebyshev Type II Infinite Impulse Response (IIR) filter bank. Each of the filters spans 4 Hz and the filter bank ranges from 4 to 40 Hz, mirroring the range of frequencies found in MI data. The CSP algorithm is applied to each of these 4 Hz bands, yielding CSP features that are unique to the frequency range of the filter. Thus, instead of producing a set of CSP features for the entire EEG dataset, FBSCP tailors the CSP features to individual frequency ranges. The features with the greatest discriminatory power are selected out of the combined CSP filter banks from all four-Hertz bands.

To select the best of the CSP features Kai Keng Ang et. al. performed a sweep of five Mutual Information-based feature selection algorithms and six classification algorithms for each subject [2]. Tailoring the feature selection process to each individual accounts for the across-subject variability in MI data frequency ranges despite the feature extraction process being constant across subjects. In this way, FBCSP provides both a feature extraction process that can be uniformly applied to all subjects and a feature extraction and classification pipeline that chooses the best features tailored to each individual.

While FBCSP is a powerful algorithm for discriminating between two MI classes, selecting features is computationally expensive. In Kai Keng Ang et. al.'s paradigm, evaluating five feature selection and six classification algorithms results in over 30 different training pipelines that must be compared per subject. In a BCI system, a training paradigm this long is impractical.

2.4 Deep learning and EEG classification

Traditional EEG processing methods like the CSP and FBCSP algorithms require separate feature selection and classification steps [2], [23], [11]. Not only is this process time consuming, but it also injects a priori assumptions about the data into the classification algorithm [9]. Deep learning algorithms have an advantage over traditional signal processing methods because they require few constraints on the data and perform both feature extraction and classification in a single step. The model learns features with the highest discriminatory power for classification on its own. There is little data preprocessing required and no restrictions are placed on what the optimal features might be.

There are many types of deep learning models, but the two most used in EEG classification are convolutional neural networks (CNNs) and recurrent neural networks (RNNs). The basic building blocks of CNNs are convolutional layers, pooling layers, and fully-connected layers. Each convolutional layer learns a kernel matrix of weights by convolving the input layer with the kernel [9]. The weights of the kernel are updated through gradient descent to optimally detect spatial features in the input [26]. The output of the convolution serves as input to the next layer. Pooling layers are inserted between convolutional layers to downsample the input, increasing the model's resilience to overfitting and reducing training complexity [35]. A fully connected layer is often the last layer of a CNN and reduces the dimensionality of the output to equal the number of classes.

While CNNs are optimized to process spatial information, RNNs are designed to model sequential information such as text and temporal data. RNNs are built with recurrent layers that sequentially take in single time steps of input. Neurons in the recurrent layer combine output from preceding time steps with the input data at the next time step, keeping an internal memory across multiple time steps [22]. While vanilla RNNs are difficult to train, updated version like the gated recurrent unit (GRU) and long-short term memory cells (LSTMs) solve these training difficulties.

2.4.1 EEGNet

Convolutional neural networks (CNNs) employ convolutional layers that are optimized to learn spatial information contained in a signal. The current champion of CNNs for EEG data classification is a model called EEGNet due to its ability to perform well across multiple different EEG classification tasks, including MI [19]. The first two layers of EEGNet are designed to parallel the first two steps of the FBCSP algorithm and are depicted in Figure 2.1. The later layers act to optimally combine and down-sample the information before classification.

The input data to EEGNet is first transformed from 22 channels of one-dimensional data to one channel of two-dimensional data, where the two dimensions are number of electrodes and the number time points, respectively. This data transformation allows the use of 2D convolutions in instead of 1D convolutions. 2D convolutions are advantageous

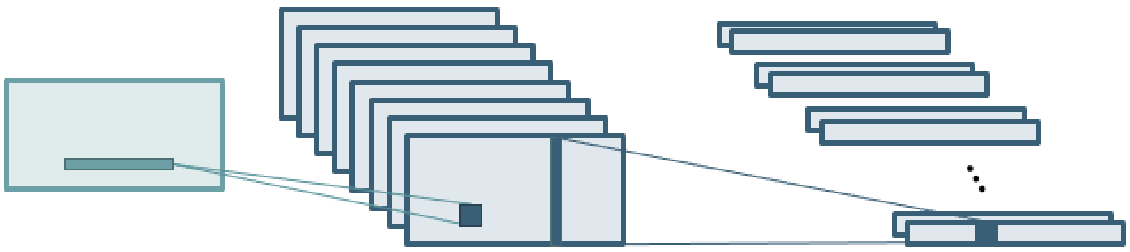


Figure 2.1: A visual schematic of the first two layers of EEGNet, that parallel the first two steps of the FBCSP algorithm. The first convolution combines temporal data for a single channel at a time. The second convolution combines spatial data for a single time step.

because they can capture the spatial information of the EEG signal; that is, how the signal changes across different electrodes.

The first layer of EEGNet is a 2D convolutional layer. The kernel is of size $(1, N)$ where N is chosen to be one-quarter of the sampling rate. By keeping the dimension of the kernel at one along the axis containing the various electrodes, the convolution kernels will only learn temporal information for a single electrode at a time. Eight kernels are used at this layer with each kernel learning temporal information at different frequencies. The convolutions are performed with the mode set to "same", so the output feature map has the same dimensions as the input feature map.

The second layer of EEGNet is a separable 2D convolution. The kernel size is $(22, 1)$ at this layer, where 22 is the number of electrodes in the training data. To contrast with the previous layer, instead of learning purely temporal information, these kernels are learning purely spatial information for each time point. Separable convolutions learn a separate set of filters for each channel of the input feature map. The output of a separable convolution is the concatenation of all the separately learned filters. In this way, separable convolutions do not mix information from the input feature maps in the learned kernels. For EEGNet's purposes, each of the set of spatial filters is uniquely learned to each temporal filters, mimicking the FBCSP algorithm. EEGNet sets the depth parameter to be two, so two spatial filters are learned for each of the eight temporal filters. Thus, there are a total of 16 channels in the output of the separable convolution layer. The mode of the separable convolution is not "same", so the output feature map has different dimensions than the input feature map. Since the depth-wise kernel spans all electrodes, the output feature map has size $(1, N)$, where N is the number of time steps in the training data.

The output feature maps of these two convolutional layers are average pooled with a kernel size of $(1, 4)$. This pooling acts as a temporal downsampler for the information gleaned in these first two layers.

To prove the network was learning what the authors designed it to, the authors of EEGNet examined the learned kernels after training was complete. They found that the temporal filters were indeed learning frequency information found in the signal. The majority of the temporal kernels were learning frequencies in the 8 to 12 Hz range. Given that MI data is known to reside in the 8 to 12 Hz range, this suggests the model is learning reasonable temporal features from the input data. Additionally, the authors proved that the information learned in the spatial filters was aggregated around the temporal and parietal lobes, just as we would expect for MI data residing in the M1 and SMA cortices. Thus, EEGNet's learned features are reasonable given the training data.

2.4.2 RNNs

While CNNs remain the favored deep learning technique for EEG classification, Ping Wang and Aimin Jiang proved RNNs just as effective [42]. However, there is more preprocessing required before passing EEG data into an RNN due to the model's inability to extract spatial information from the signal.

Wang and Jiang perform dimensionality reduction and channel weighting to raw EEG data before passing it to an RNN [42]. The 1d-SAX algorithm performs dimensionality reduction by splitting the time series of each channel into fragments of equal length and computing the mean of each fragment. Wang and Jiang added the slope of each fragment as an additional feature in case any of the fragments had the same mean. Here, the 1d-SAX algorithm reduces the dimension of the data. Channel weighting via gradient descent selected the best mean and slope across the fragments for each time series that results in the best classification accuracy. This step acts as a spatial filter, selecting the channels with the best spatial information for classification. A total of 16 channels were selected and these mean and slope matrices were analyzed in parallel by separate LSTM networks.

To find the optimal LSTM network, Wang and Jian experimented with both increasing the number of units in a single LSTM layer and increasing the number of LSTM layers. They first varied the number of units in a single LSTM layer from two to 24 and found that the model's performance increased with the number of LSTM units. Then, they set the number of LSTM units in the first layer constant at eight and varied the number of units in the second layer from eight to 16. Increasing the number of units in the second layer did not improve the model's performance. Next, the first and second LSTM layers were set to a constant eight units and the third LSTM layer was varied from eight to 16 units. As with the previous experiment, increasing the number of units in the third layer did not improve model accuracy. The results of this set of experiments indicates the number of units in the first layer is crucial to the model's performance. Additionally, it seems adding subsequent LSTM layers do not lead to a measurable increase in accuracy despite adding more training parameters. This result suggests a single LSTM layer is optimal for EEG classification.

Ping Wang and Aimin Jiang's methodology is able to match the performance of FBCSP on two-class classification. However, their methodology of least-squares averaging leads to loss of information in the signal. The averages and slopes do not hold as much information as the CSP filter banks of FBCSP, suggesting the accuracy of an RNN model could be increased if the feature extraction method were more sophisticated.

2.4.3 Hybrid CNN-RNN networks

EEG signals are unique in that they are a spatial signal that vary across time; there are both spatial and temporal components of the signal. Research has shown hybrid CNN-RNN networks are effective models for classifying spatiotemporal signals like video and EEG signals [40], [38]. The idea is the CNN performs dimensionality reduction and captures the important spatial features of the input data in a compressed form. These compressed vectors are the input into the RNN for modeling how these spatial features change over time [38]. Thus, hybrid networks have the advantages of both CNNs and RNNs. Additionally, there is no need for signal preprocessing; the model optimally extracts the best features without the need for manual feature selection, as feature extraction and classification are combined into a single pipeline.

Chapter 3

Methods

3.1 Resources and Tools

This project utilizes MATLAB and Python 2.7. In MATLAB, the open-source toolbox BioSig is used to load in the BCI Competition dataset [37]. The Python library numpy is extensively used throughout the project for its efficient data storage and mathematical data manipulation [12]. The SciPy and Scikit-learn libraries are used for signal preprocessing and data cleaning [41], [28]. The neural network models are built with Keras, a high-level deep learning API, and TensorFlow back-end [5], [21]. Due to the high level of memory required for running deep-learning models, Nvidia GPUs are used to increase the speed of calculations.

3.2 Datasets

3.2.1 BCI Competition Dataset

The BCI Competition 2008 Graz Data set A is used for data exploration and for testing deep learning model architectures [3]. The dataset contains EEG data collected from nine separate subjects. The experimental paradigm consists of four different visual cue-based motor imagery tasks: left hand, right hand, both feet, and tongue. The subjects each performed 288 total trials, with an even ratio between the four tasks.

For each trial the subject was placed in front of a computer screen. An auditory tone accompanied a black cross on the screen to mark the beginning of each trial (t = 0s). After two seconds (t = 2s) an arrow appeared on the screen indicating which imagined movement the subject was to perform. The arrow stayed on the screen for 1.25 seconds, but the subjects continued to perform the motor imagery task until the black cross appeared on the screen again at t = 6s. The subjects were given a short break before the next trial began. The experimental paradigm is shown in Figure 3.1.

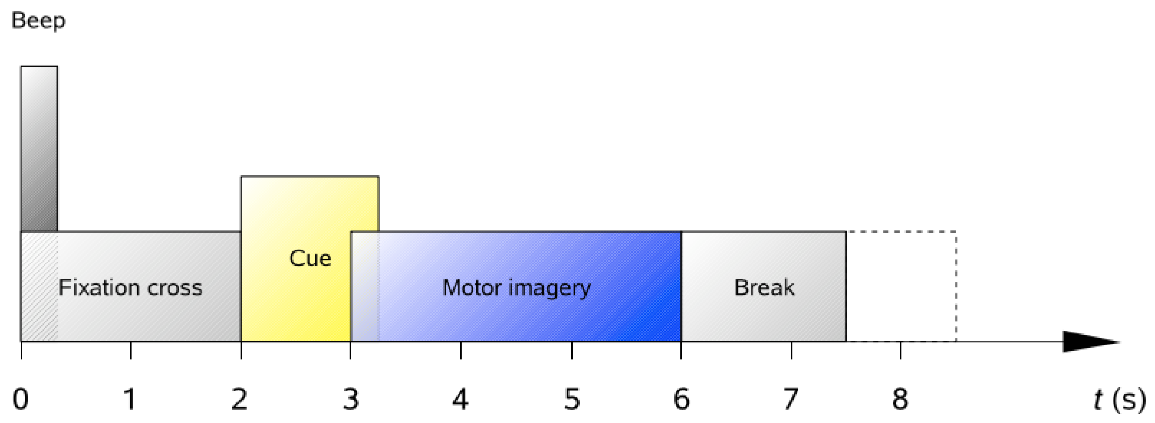


Figure 3.1: The experimental paradigm for the BCI Competition datset. Each motor imagery task was performed for four seconds at the onset of a visual cue. Figure reproduced from [3].

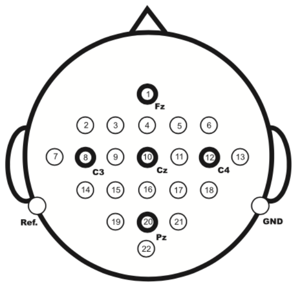


Figure 3.2: The locations of each of the 22 electrodes placed on the subjects' scalp.

Data was recorded from 22 electrodes placed on the scalp sampling at 250 Hz (Figure 3.2). Each subject's data was handled separately, as there is a high degree of between-subject variability in EEG data. Thus, the resulting data was of the form $(22, T)$, where T represents the total number of time samples. There were nine separate datasets, one for each subject. It is important to note that in this study data from separate subjects is not mixed. In addition, the subject number stays constant throughout the presented work; i.e. subject 1 is the same subject throughout all the results.

3.3 Data preprocessing

All the datasets were bandpass filtered and standardized. Third order Butterworth filters were used to bandpass filter the signal between 4 and 40 Hz to remove low-frequency eye-movement (EOG) artifacts and high-frequency noise [27]. In addition, the data were standardized with an exponential moving window to follow the standard set by Schirrmeister et al [36].

3.3.1 Epoching

For four-class classification tasks, only the data containing motor imagery tasks from the BCI Competition dataset were used. Thus, each single four-second trial of the 288 total trials from each electrode was extracted from the full dataset. For each subject, j, this resulted in an X matrix of data and a y matrix of trial labels:

$$X_j \in R^{288x22x1000}$$

$$y_j \in R^{288}$$

3.4 Data windowing

To analyze the effect of temporally shorter sequences of data effects model performance, the epoched datasets were windowed into shorter data matrices. We chose the lengths of the truncated sequences to reflect one-half, one, two, and three seconds of data collection at 250 Hz, resulting in window lengths of 125, 250, 500, and 750 samples, respectively. To create these windowed data matrices of window length L, the first L samples of every trial was copied and placed in a new data matrix. This was done for every subject, j, resulting in a pair of X and y matrices:

$$X_j \in R^{288x22xL}$$

$$y_j \in R^{288}$$

where L is the window length in number of samples. We call the set of data created in this way the single-window data matrices. The above windowing method only analyzes the samples that follow immediately after cue onset. To investigate how the data features change over time successive and non-overlapping windows were taken of the entire four-second epoched dataset (Figure 3.3). Thus, each of the original 288 trials yields $\frac{1000}{L}$

windows, where L is the length of the window in samples. We call this data the multi-window data matrices.

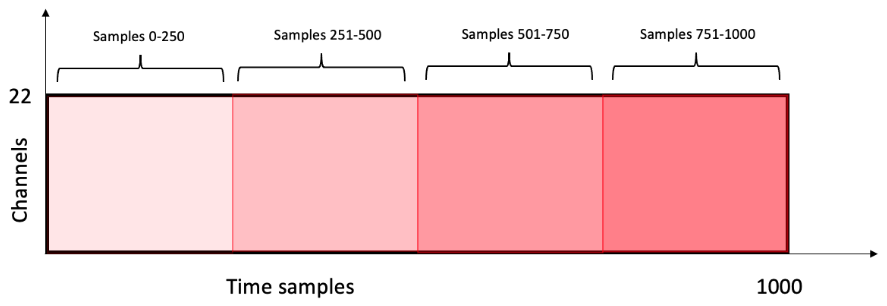


Figure 3.3: An example of the multi-window datasets created by successive and non-overlapping 250-sample window.

3.5 Data augmentation

Temporally shortening the dataset reduces the amount of data the network can learn, potentially compromising its accuracy. In the field of computer vision, dataset augmentation create new images by rotating, cropping, or altering the color of existing images. For the MI data we created new data by cropping in time rather than in space. This effectively creates "new" training data from the existing data set [29]. However, it is important to note that this "new" data is not independent from the original training data, so it is not as valuable as adding truly new data samples. Thus, there are limits to how much additional information augmented data can provide.

For each window length, L, where L was 125, 250, 500, or 750 samples, we again took successive crops of the original epoched dataset but this time with a 50 percent overlap. Instead of taking non-overlapping windows, as in the single-window data matrices, the windows here had a 50 percent overlap with both the previous and following window. As shown in Figure 3.4, starting from the first sample, L samples were copied and placed into a new data matrix as a new, truncated trial. After striding $\frac{1}{2}$ L samples, the next L samples were copied and placed in a new data matrix as another trial. This was repeated for the entire 1000-time sample original dataset. Thus, $(1000–L - 1)/0.5L$ new "trials" were created from each original 1000-time sample trial. It is important to note that in these new data matrix "trials" do not refer to unique trials, but to a collection of truncated sequences taken from a unique trial. All of these truncated sequences were combined into a single, now larger dataset. We refer to this larger set of data of overlapping truncated data matrices as the augmented dataset and refer to them by their length in time (e.g. the two-second augmented dataset).

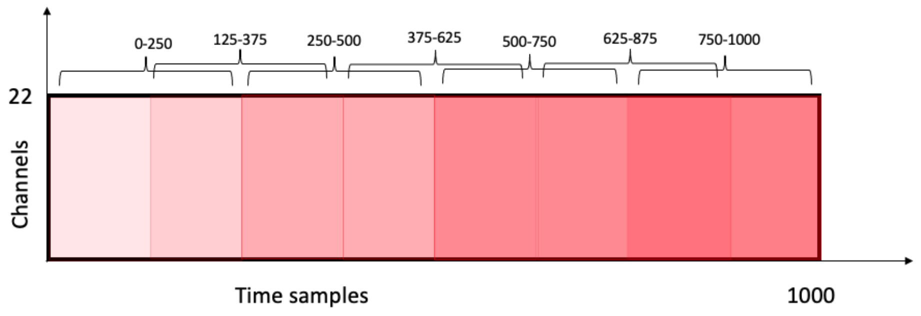


Figure 3.4: A schematic representation of the creation of the 250-sample augmented dataset. The sliding window is 250 samples in length and applied at a stride of 50 percent of window length, here 125 samples.

3.6 CNN-RNN model design

The first three layers of EEGNet – 2D convolution, separable 2D convolution, and average pooling – optimally learn compact representation of the spatial and temporal information in the original EEG signal. The output after these layers is a one-dimensional vector containing this optized information. Since RNNs take one-dimension vectors such as these as input, we hypothesize the first three layers of EEGNet would be appropriate and sufficient as the CNN layers in a hybrid CNN-RNN model.

A single layer comprised of a unidirectional RNN takes the vector output of the CNN as input and produces a classification. Only a single RNN layer was used to reduce the model complexity and thus reduce the risk of overfitting the training data.

A visual schematic of the proposed network is shown in Figure 3.5.

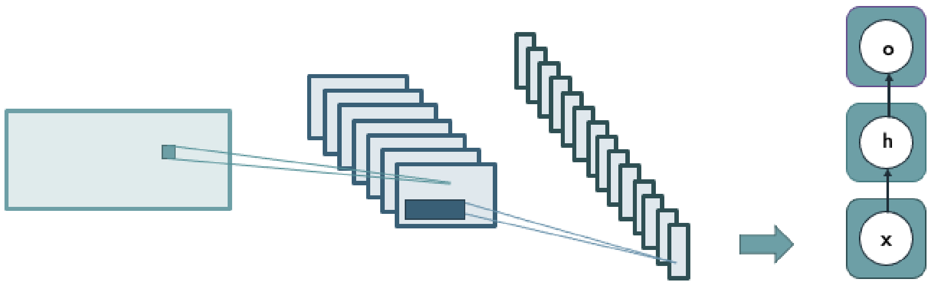


Figure 3.5: An overview of the proposed CNN-RNN model. Temporal information is learned through the first convolution and spatial information learned through the second convolution, resulting in a series of feature vectors passed as input to the RNN. The goal of the RNN is to learn feature matrices, h, that best classify, o, the input x.

3.7 CNN-RNN hyperparameter tuning

Due to its success in classifying EEG signals, the first three layers of EEGNet were used as a basis for the CNN portion of the proposed hybrid model. To better fit the CNN-RNN model, the hyperparameters of EEGNet were tuned. Specifically, the number of convolutional kernels and size of the kernels in the first layer were varied, the number of depth-wise kernels was varied, and the size of the averaging pooling kernel were all optimized (Table 3.1).

For the 2D convolutional layer, the size of the convolutional kernels was varied to find the length that resulted in the highest classification accuracy. Decreasing the length of the temporal filter forces the model to learn higher frequency information, while increasing the kernel length allows the model to learn lower frequency components in the signal. Since MI data usually resides in the 8 to 12 Hz range, a larger kernel should be better able to learn these low frequencies than a shorter kernel would. The base model of EEGNet employed a kernel of length $\frac{1}{4}$ the sampling rate, but the sampling rate of the proposed model's training set, at 250 Hz, is not divisible by four. Instead, a kernel of length 50 was used, representing $\frac{1}{5}$ the sampling rate. To asses the effects of changing the kernel length, kernels of lengths 25 and 75 were also used and their performances compared. The base model of EEGNet used a kernel of height one, meaning no spatial information was learned in the first convolutional layer. Allowing the kernel to learn temporal information from more than one electrode at a time could result in the model learning more general temporal patterns. To test this hypothesis, the performance of models with kernels of heights one, two, and four were compared. Thus, for the first convolutional layer, a sweep of nine different kernels were used: of heights one, two, and four combined with widths 25, 50, and 75.

The number of convolutional kernels was also optimized. More kernels allows the model to learn more specific features in the input signal, but also results in more parameters and thus the possibility of increased training time and overfitting. The base EEGNet model employed eight convolutional kernels in the first layer. In addition to eight kernels, the performance of the model with 12 and 16 kernels was compared.

For the depth-wise convolutional layer, the number of depth-wise kernels was varied. The base EEGNet model employed two depth filters at this layer. Given that the CNN portion of the hybrid network aims to learn the spatial features of the input signal, and this layer in EEGNet is optimized to learn spatial features, increasing the number of kernels at this layer should increase model performance. Because depth-wise convolutions learn unique kernels for each input feature map channel, the number of learned parameters is less than those of a traditional 2D convolutional layer. This suggests increasing the number of kernels at this layer is less likely to cause overfitting. In optimizing the number of kernels, the performances of models with one, two, four, and eight kernels were compared.

The length of the pooling layer was also optimized. Average pooling effectively acts as a downsampler, reducing the likelihood of the model overfitting to the training data. This is an especially important consideration for the CNN-RNN hybrid network since the pooling layer is the last layer of the CNN; the output of the pooling layer is passed into the RNN for classification. RNNs are difficult to train given that there is a high risk of overfitting if the input dataset is too high dimensional, thus optimizing the pooling layer kernel is essential. Increasing the pooling layer too much will result in the loss of too much information, while taking the pooling layer to be too small will increase the likelihood of the RNN model to overfit to the data. The base EEGNet model employed an average 2D pooling kernel of size $(1, 4)$. The model performance was compared to pooling kernels of $(1, 2)$ and $(1, 8)$. Since the output of the first convolutional layer is of type "same" and the depth-wise convolution does not affect the temporal axis, the input data to the pooling layer has the same sampling rate as the original EEG signal. Average pooling with a kernel of length eight downsamples the data to 31.25 Hz, indicating the maximum observable frequency is about 16 Hz. Since this is just above the 8 to 12 Hz range for MI signals, the pooling kernel of $(1, 8)$ should be optimal for this type of data.

After tuning the hyperparameters of the CNN, the number of units in the LSTM network was varied to find the best performing model. Increasing the number of LSTM units allows the network to have better "memory" but increases the likelihood of overfitting. Given the maximum number of learned kernels in the aforementioned hyperparameter experiments is 128 (eight from the 2D convolution and 16 depth-wise kernels for each of the convolution kernels), the number of units tried was eight, 16, 32, 64, and 128. Since RNNs are prone to overfitting, the RNN was partially optimized to a sub-optimal CNN first. Once the CNN was fully optimized, another hyperparameter search was performed on the RNN to fully optimize the number of LSTM units.

Layer	Number Filters	Size of Filter	Options
Conv2D	8, 12, 16	(1, 25), (1, 50), (1, 75) (2, 25), (2, 50), (2, 75), (4, 25), (4, 50), (4, 75)	mode=same
DepthwiseConv2D	1, 2, 4, 8	(22, 1)	depth constraint=1
AveragePool2D	n/a	(1, 2), (1, 4), (1, 8)	none

Table 3.1: A list of all the hyperparameters tried during hyperparameter tuning.

Chapter 4

Results

4.1 Windowed datasets

EEGNet achieves an average across-subject accuracy of 64.49 percent (standard deviation 13.45 percent) on the original four-second (1000-sample) dataset. A Wilcoxon signed-rank test proves windowing the dataset to 500 or 750 samples does not significantly affect model performance, as these two datasets achieved 66.29 and 65.54 percent accuracies, respectively (p = 0.26203, p = 0.89274, respectively). Interestingly, while reducing the data to two seconds (500 samples) dropped the across subject accuracy to 58.19, a Wilcoxon test proves this drop is not a significant for the dataset (p = 0.15478). However, further shortening the dataset to 125 samples did statistically negatively affect the models' performance, reducing the across-subject accuracy to 49.35 percent (p=0.05041). These results were partially unexpected, as we predicted all the windowed data to perform worse than the original given that more data generally leads to better results. The fact that the 125 and 500-sample datasets performed worse suggests our intuition is partially true. Perhaps increasing data length beyond the 500-sample mark causes saturation, when additional data does not provide additional useful information. However, further experimentation suggests another explanation.

4.2 Augmented datasets

Performing data augmentation does not improve the across-subject accuracies, despite the increase in training data. Performing data augmentation reduced the across-subject accuracy from 49.35 to 41.09 percent for the 125-sample window, from 58.19 to 44.63 percent for the 250-sample window, from 66.29 to 48.52 percent for the 500-sample window, and from 65.54 to 54.61 percent for the 750-sample window (Figure 4.1). It is interesting to note that the 750-sample augmented dataset outperformed the 500-sample augmented dataset despite the two corresponding single-window datasets performing the same. This trend suggests the sliding window data augmentation techniques used here are impeding model performance as the 750-sample augmented dataset contains less overlapping data compared to the 500-sample dataset.

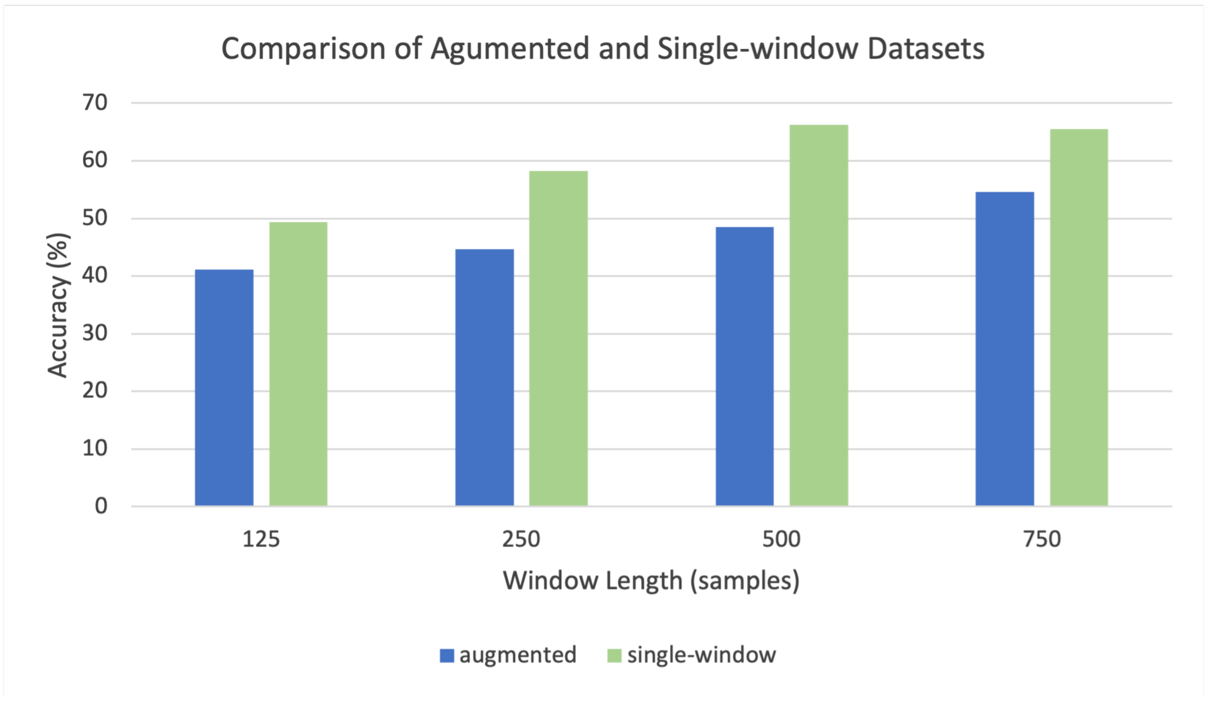


Figure 4.1: The across-subject average accuracies for the augmented datasets versus the single-window datasets. Despite having a fraction of the training data the single-window datasets outperform the augmented datsets for all window lengths.

To investigate why the data augmentation and resulting increase in training data was not leading to increased model performance, we looked at the models' performance on individual windows. The 500-sample multi-window dataset highlights nuances of the dataset that are causing these results. These analyses suggest the model performance decays in the latter half of the original four-second dataset. The first 500-sample window achieves 66.29 percent across-subject accuracy, whereas the second 500-sample window only achieve 32.20 percent accuracy (Figure 4.2). An accuracy of 32.20 percent is just above the performance level expected if the model was choosing one of the four classes at random, suggesting the model is no longer learning good features for classification in the second 500-sample window (the last two seconds of data collection). However, the model's performance on the first 500-sample window (the first two seconds of data collection) indicates there are good features for classification in the signal. These features must deteriorate or disappear after about two seconds of the subject performing a MI

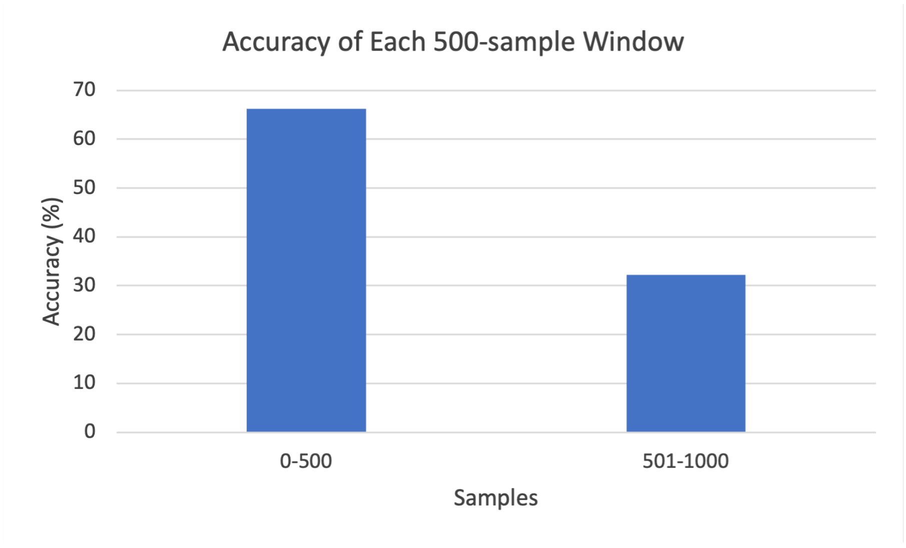


Figure 4.2: The results of each of the individual two-second windows. The first window spans samples 0-500 and the second spans samples 501-1000.

task.

The one-second multi-window dataset corroborates this hypothesis. The first 250-sample window achieves an across-subject accuracy of 58.19 percent and the second window achieves 55.37 percent. The performance drops with the latter two windows: the third achieves only 32.39 percent accuracy and the fourth achieves 28.81 percent accuracy. Figure 4.3 highlights the fact that these trends are consistent across all nine subjects, a rare phenomenon in the field of EEG data. Additionally, we notice there is a distinct performance gap of 23 percentage points between the second and third 250-sample windows. The first two and latter two 250-sample windows perform more similarly to each other than they do to their other two counterparts. However, that trend could be due to the discrete nature of the windowed data.

Since the multi-window results show a decrease in model performance after two seconds of data collection, we omitted any windows starting after the two second mark for the 125 and 250-sample datasets (i.e. for the 250-sample window, the window spanning samples 375-625 was included but the window spanning samples 500-750 was dropped,

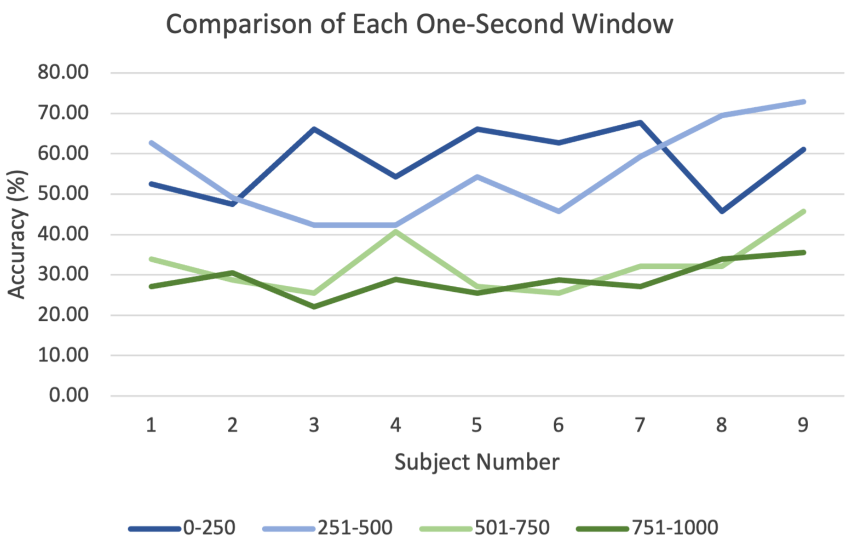


Figure 4.3: The results of each of the four one-second windows. The windows that span the first two seconds (blue) outperform the windows that span the second two seconds (green).

Figure 3.4 in Methods). Since the 750-sample and 500-sample windows span more than the first two seconds they were omitted from these analyses. Decreasing the size of the datasets in this way did result in an improved performance in across-subject accuracy: the 125-sample dataset increased to 43.37 percent and the 250-sample dataset to 53.74 percent (Figure 4.4). Still, these numbers fall short of the accuracies attained from the single-window analyses.

4.3 Individual window analysis

To better understand why the data augmentation techniques did not improve the model's performance, each overlapping 125-sample window from the augmented dataset was analyzed. That is, the model was trained and tested on each separate window. The results are shown in Figure 4.5a. The average accuracy of all the windows spanning the first two seconds is 48.18 percent and the average for the last two seconds is 27.93 percent, giving a total average across all windows of 37.38 percent. Interestingly, the accuracy

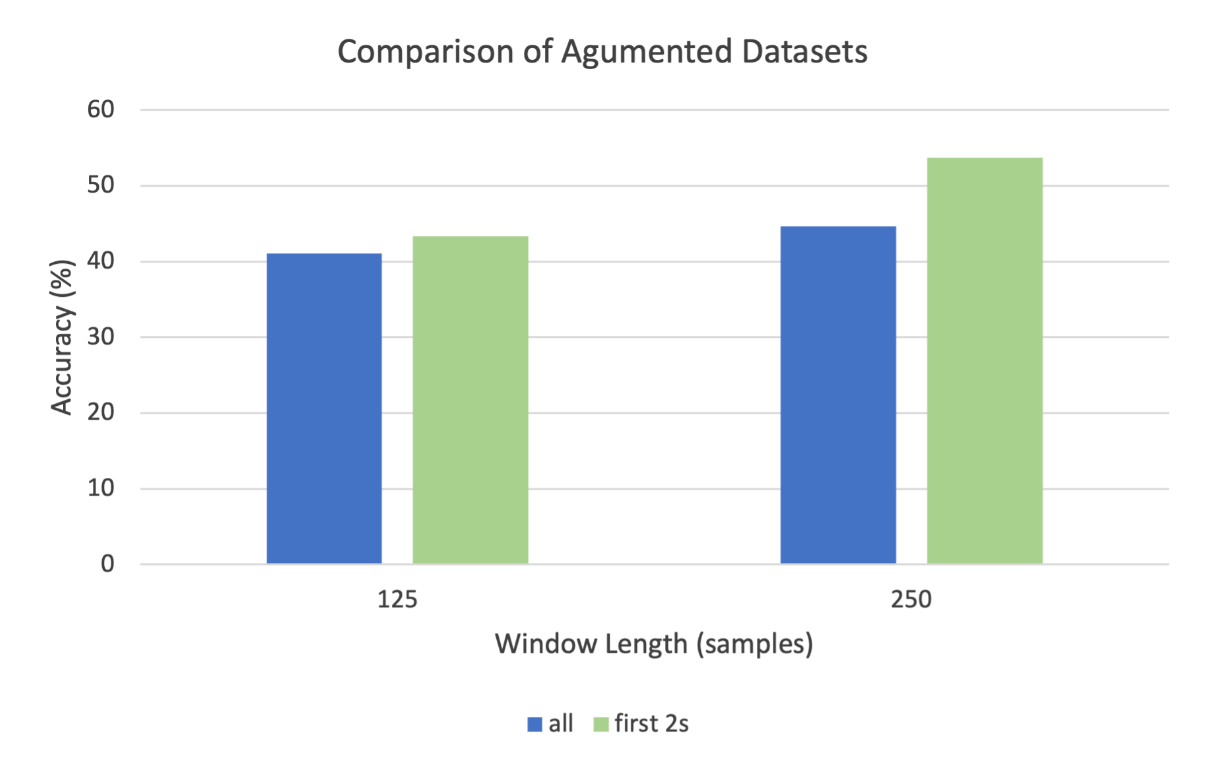
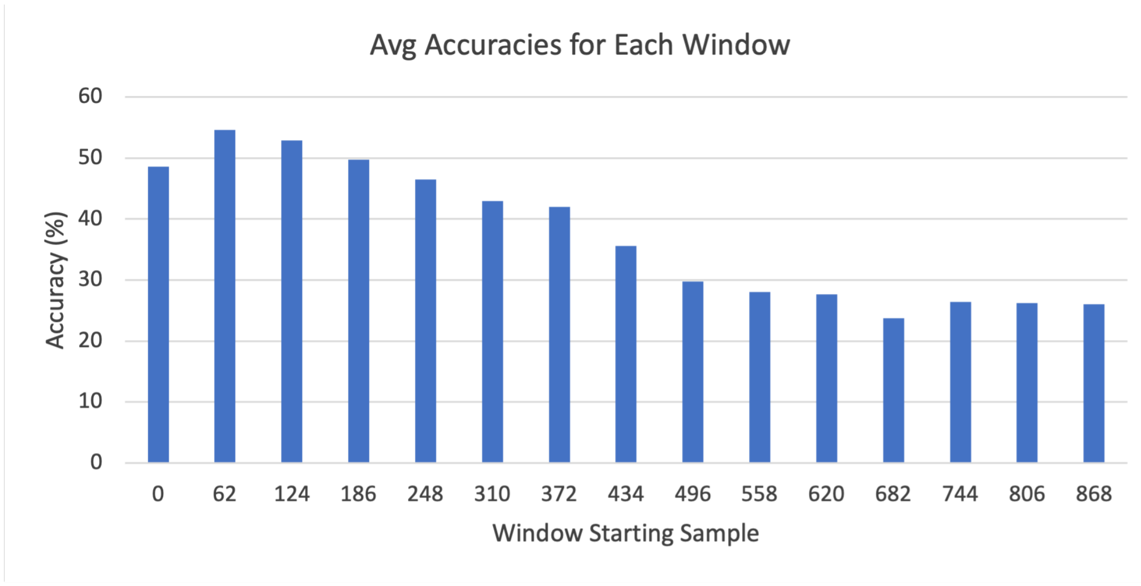


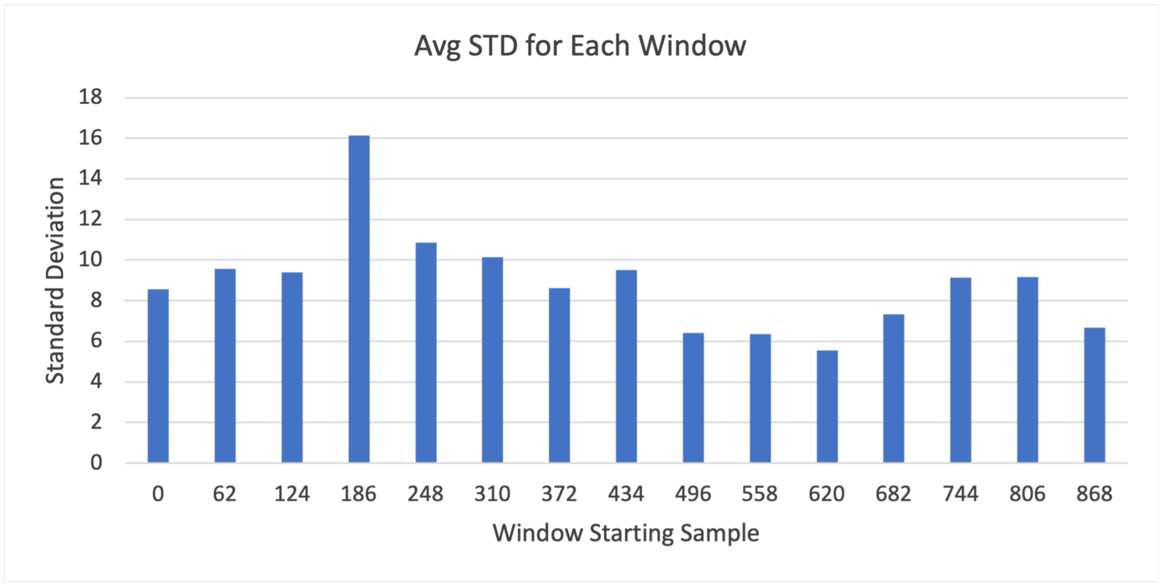
Figure 4.4: The results of the full augmented dataset versus the augmented dataset with the last two seconds of data omitted. Reducing the size of the dataset increases the models performance for both datasets.

of the augmented dataset with the last two seconds of data omitted is lower than the accuracy of the average accuracy of the individual windows spanning the first two seconds of data collection (43.37 percent versus 48.18 percent, respectively). This result is surprising given that the augmented dataset contains seven times more training data than in individual window datasets.

The across-subject standard deviations (STD) for each individual window also reveal interesting trends (Figure 4.5b). The STDs for the windows spanning the first two seconds is higher than that of the last two seconds (10.47 versus 7.52, respectively). This suggests the trends seen in the last two second of the data are more uniform across subjects than those seen in the first two seconds. Additionally, the windows with the highest STDs are those spanning samples 186 to 310 (0.74 to 1.2 seconds after cue onset). These three windows also correspond to the begin of the decrease in accuracy. Taken together, these



(a) The across-subject results of training and testing the model on each window separately.



(b) The across-subject STDs for each window. The windows in the latter half of the data have lower STDs than the windows in the first half.

two trends in the data suggest that while there is some across-subject variation in how strong the signal begins and when the signal deteriorates, it uniformly deteriorates across all subjects.

The decrease in model performance suggests the features decay over the course of the four seconds of data. In the first part of the data the model is able to learn well-generalized features in the data, but is unable to do so in the latter half. To test the types of features learned in each window the model was trained on all windows at once (the full augmented dataset) but tested individually on each window at a time. Thus, the model was trained only once (on the full dataset), but the test dataset was varied to analyze the model's performance on each window separately. In doing this, we force the model to learn features found in the entire dataset. At test time, individual windows with features similar to those the model learned will have high test accuracies. The results of this experiment are shown in Figure 4.6 and are compared to the results obtained from testing and training on each individual window. We notice this split training and testing paradigm leads to worse performance in the first few windows but better performance in the latter windows. In fact, the overall accuracy increases; the average accuracy of all windows is now 41.11 percent, up from 37.38 percent when we trained and tested the model on a single window at a time. Thus, the improved performance in the latter part of the data overcomes the decreased performance in the first few windows.

The results of Figure 4.6 show that discriminatory features do exist in the last two seconds of data, but the model is unable to extract them when just trained to an individual window at a time. Only when the model is forced to learn features found across the entire four seconds of data collection do these features appear – in the form of better model classification accuracy – in the latter windows. It could be that the features in these windows are too weak to be extracted from a single window at a time and thus require more data for the model to learn them accurately.

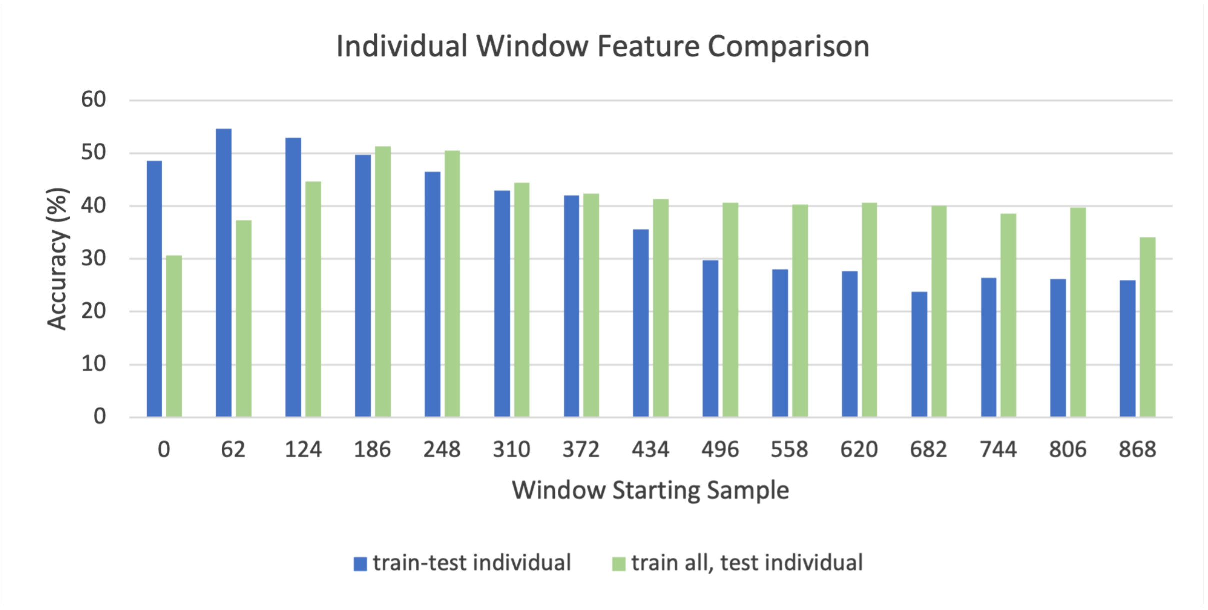


Figure 4.6: The results of training on the full augmented dataset and testing on individual windows.

4.4 CNN-RNN hyperparameter tuning

The hybrid model hyperparameters tuning was performed with the 125-sample augmented dataset. The results of Figure 4.6 suggest training to the entire dataset allows the model to learn features that are better generalized, which is why this dataset was chosen to tune the hybrid model. For the following experiments the 125-sample augmented dataset was split into training, validation, and testing sets in a 80-20-20 split.

The first hyperparameter search experiment analyzed the effect of number of LSTM units and size of the pooling kernel. For this set of experiments, the 2D convolutional kernel size was fixed at (1, 25) with eight kernels. The separable convolution layer was at kernel size (22, 1) with two depth-wise kernels. The pooling kernel size was chosen to be (1, 2), (1, 4), or (1, 8), representing varying levels of downsampling. The output feature maps from the CNN were passed into an LSTM with M units, where M was eight, 16, 32, or 64. Since the size of the pooling layer so heavily effects the potential overfit of the RNN model, which is also heavily affected by number of LSTM units, these two hyperparameters were optimized together. In total, the hyperparameter search space was 12 (three pooling kernel choices and four LSTM unit choices).

The test results of this first set of experiments are shown in Figure 4.7. The first thing to note is that increasing the size of the pooling layer generally increased performance. This trend aligns with the intuition that a $(1, 8)$ pooling kernel would be optimal given the frequency signature of MI signals. The other important note here is that increasing the number of LSTM units monotonically increased model performance. While not shown in Figure 4.7, the 64 unit model showed worse overfitting than the 32 unit model and took more training time due to the increased number of parameters. Since this was the first set of hyperparameter optimization experiments, and the number of LSTM units would

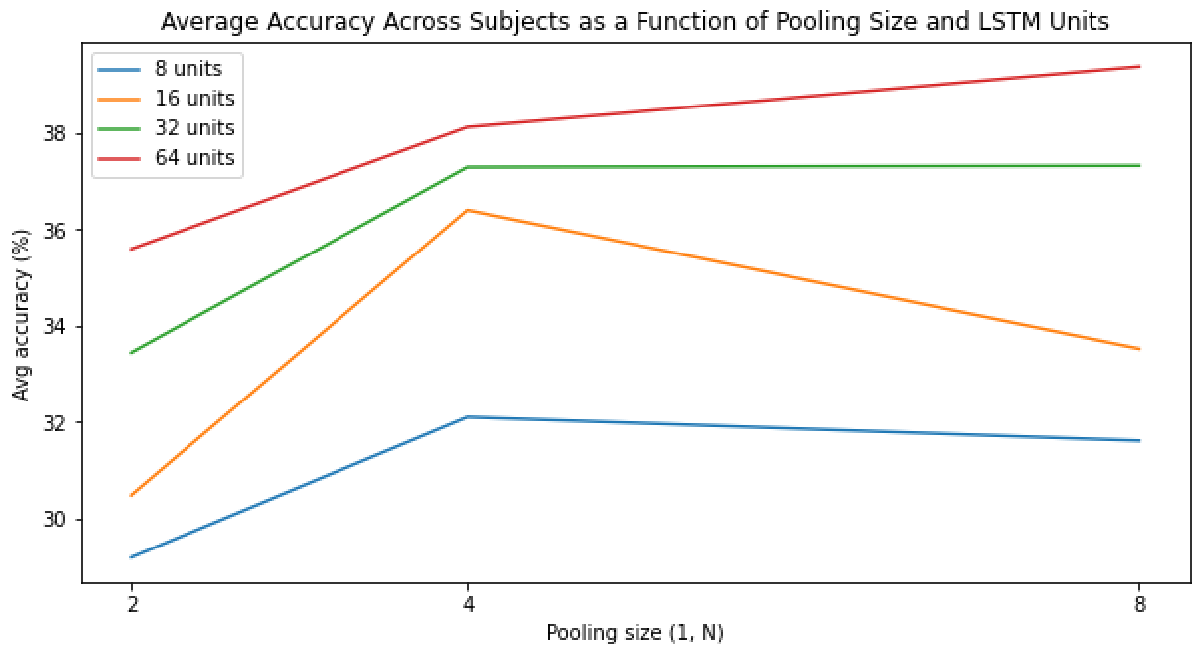


Figure 4.7: The average subject accuracy as a function of both pooling kernel size and number of LSTM units. The pooling kernel size is shown on the x-axis. The number of LSTM units is represented by the varying colors of the plot lines. Using a pooling kernel size of (1,8) with 64 LSTM units shown the best results.

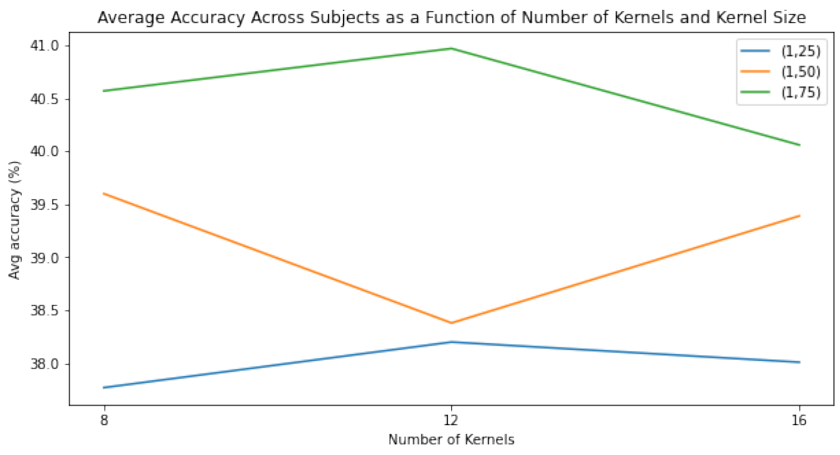
be further optimized after the CNN hyperparameter search was finished, the rest of the experiments were carried out with 32 LSTM units. Additionally, the pooling kernel of $(1, 8)$ was chosen as the best and the rest of the experiments were carried out with this pooling kernel.

The second set of experiments analyzed the effect of convolutional kernel size and number of kernels on model performance. The convolutional kernel size was chosen to be $(1, 25)$, $(1, 50)$, or $(1, 75)$. For each of these kernel sizes, eight, 12, and 16 kernels were trained. To reduce overfitting, each of the convolutional kernels was constrained to have a maximum norm of one. For this set of experiments, the separable convolution was held constant at two depth-wise kernels of size $(22, 1)$ and the pooling kernel at $(1, 8)$. The number of LSTM units employed was 32. The results of this set of experiments are shown in Figure 4.8a. The same set of experiments were run with convolutional kernels of size (2, 25), (2, 50), and (2, 75) (Figure 4.8b) as well as (4, 25), (4, 50), and (4, 75) (Figure

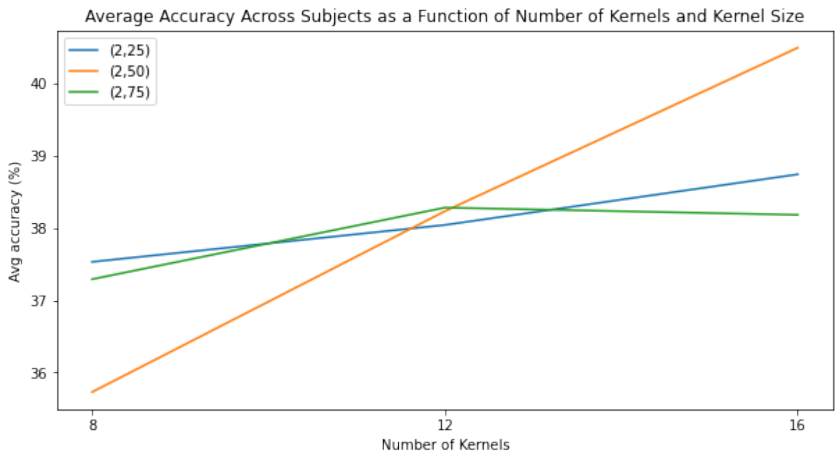
4.8c.

Taken together, Figures 4.8a - 4.8c indicate that using a larger kernel length is optimal, affirming our intuition. An interesting thing to note here is that increasing the number of learned kernels does not seem to negatively affect model performance. In fact, performance increases with increasing the number of kernels. This is surprising as we would expect adding more kernels would lead to overfitting and thus negatively impact model performance. Another interesting note is that increasing the height of the kernel worsens model performance. All of the $(2, N)$ and $(4, N)$ kernels perform similarly for a given number of kernels. Performance for the $(2, N)$ and $(4, N)$ kernels is worse than performance for the $(1, N)$ kernels across all number of kernels. In fact, the top accuracies occurred with kernel of size (1, 75) across all number of kernels. This result suggests that allowing spatial information to mix with frequency data hurts model performance, which goes against our intuition. It is possible that adjacent electrodes in the data array did not contain similar enough frequency components for the model to be able to generalize them well.

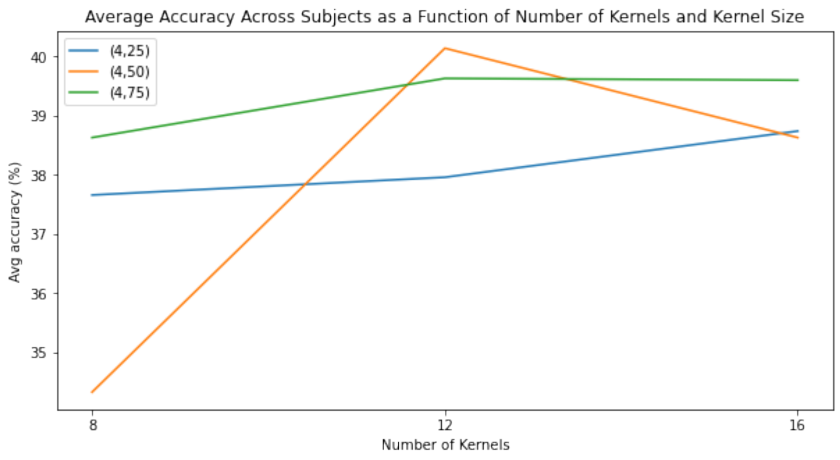
The next set of experiments investigated the influence of the number of depth-wise kernels on model performance. For these experiments, the convolutional kernel was of size (1, 75), given that kernel performed best in the previous set of experiments. The number of convolutional kernels was chosen to be either eight or 12, to further compare these top-two performing models. The depth-wise kernel was held constant at size (22, 1), but the number of kernels was varied at one, two, four, and eight. To reduce overfitting, the max-norm of the depth-wise kernels was also constrained to be one. The pooling kernel was constant at size (1, 8) and 32 LSTM units were used. The results are in Figure 4.9.



(a) The average subject accuracy as a function of both convolutional kernel size and number of convolutional kernels. The best accuracy is shown by the model with kernel size (1, 75) and with 12 kernels.



(b) The average subject accuracy as a function of both convolutional kernel size and number of convolutional kernels. The best accuracy is shown by the model with kernel size (2, 50) and with 16 kernels.



(c) The average subject accuracy as a function of both convolutional kernel size and number of convolutional kernels. The best accuracy is shown by the model with kernel size (4, 50) and with 12 kernels.

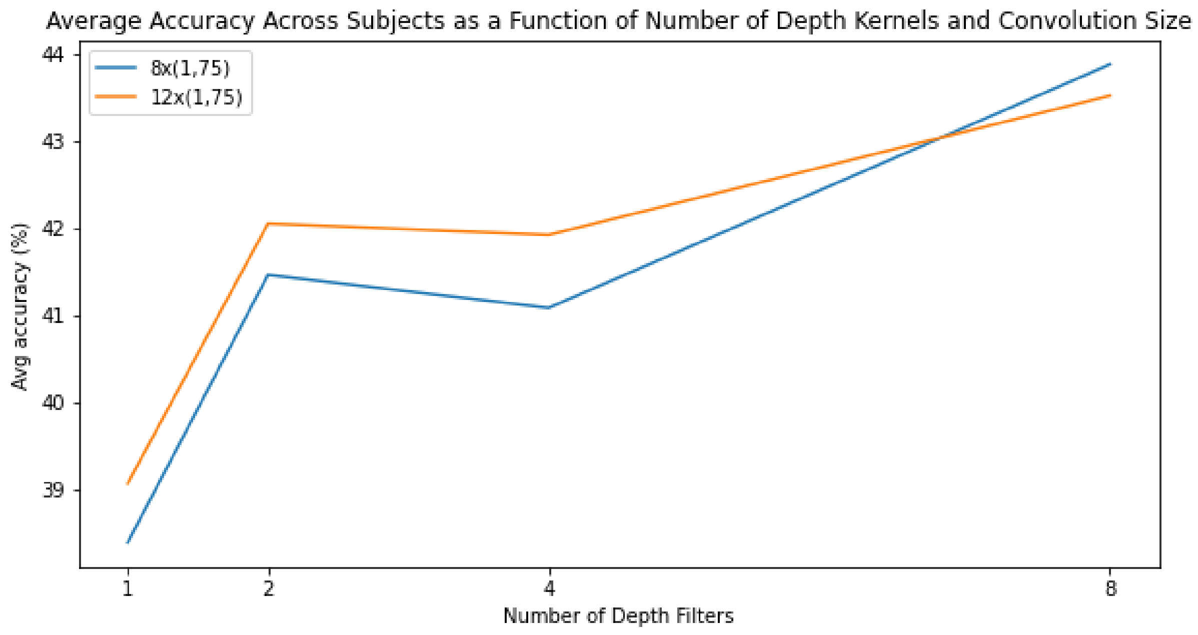


Figure 4.9: The average subject accuracy as a function of number of depth-wise filters. The performance of the 8x(1, 75) and 12x(1, 75) convolutional kernels is compared. The best performing model is the 8x(1, 75) convolutional kernel with 8 depth-wise kernels.

The results of this experiment indicate increasing the number of depth-wise kernels increases model performance. This agrees with our intuition, given that these kernels are learning spatial information and the CNN network in a hybrid CNN-RNN model is designed to extract spatial features.

At this point, all of the hyperparameters of the CNN were considered tuned. The optimal model found was a CNN with three layers: a 2D convolution, a depth-wise convolution, and an average pooling layer. Two optimal 2D convolutions were found: both with kernel size (1, 75), but one with eight kernels and one with 12 kernels. The optimal depth-wise convolution learns eight kernels learned for each input feature map. The resulting 64 feature maps are of size $(1, N)$, where N is the number of time steps in the data (here 125 for the 125-sample dataset). These feature maps are average-pooled with a kernel of size (1, 8) to downsample the data. The resulting feature maps are of size (64, 1, 16). This 3D input is reshaped to be of size (64, 16) for the eight-kernel model and (96, 16) for the 12-kernel model. Thus, the input to the RNN is a set of 16 64-dimensional or 96-dimensional spatial vectors.

The final set of experiments analyzed the effect of the number of LSTM units on the optimal CNN model. Given that the dimension of the input to the RNN is 64, the number of units was chosen to be 32, 64, and 128. The 12-kernel model saw a performance decrease as the number of LSTM units increased, suggesting the model was overfitting. The eight-kernel model showed a performance peak with 64 LSTM units, at 43.88 percent accuracy. This was chosen as the best performing model. A summary of all the hyperparameters in this optimal model are shown in Table 4.1.

Layer	Number Filters	Size of Filter	Options
Conv2D	8	(1, 75)	mode=same
DepthwiseConv2D	8	(22, 1)	depth constraint=1
AveragePool2D	n/a	(1, 8)	none

Table 4.1: A summary of the best hyperparameters found during model tuning.

4.5 Testing hybrid model

The optimized hybrid CRNN model was tested on the single-window task-only datasets and the performances compared to those of EEGNet. The across-subject accuracy of the hybrid model was best for the shorter windows at 45.98 percent for the 125-sample window and 44.43 percent for the 250-sample window. The 500-sample window achieved an across-subject accuracy of 34.10 percent and the 750-sample window achieved 25.11 percent. The fact that the performance decreases as the window size increases agrees with our intuition. The CNN was optimized to extract spatial features from a 125-

sample window. Increasing the window length increases the amount of information in the signal, some of which is lost during the dimensionality reduction performed by the CNN.

The results of the hybrid model are compared to the results of EEGNet on the same datasets and modeled in Figure 4.10. The hybrid model does not perform as well as EEGNet for the longer datasets, but is almost able to match the performance of EEGNet for the 125-sample window (45.98 percent versus 49.35 percent, respectively). Given that the goal was to produce an effective way to classify temporally shorter sequences of data, these results are encouraging.

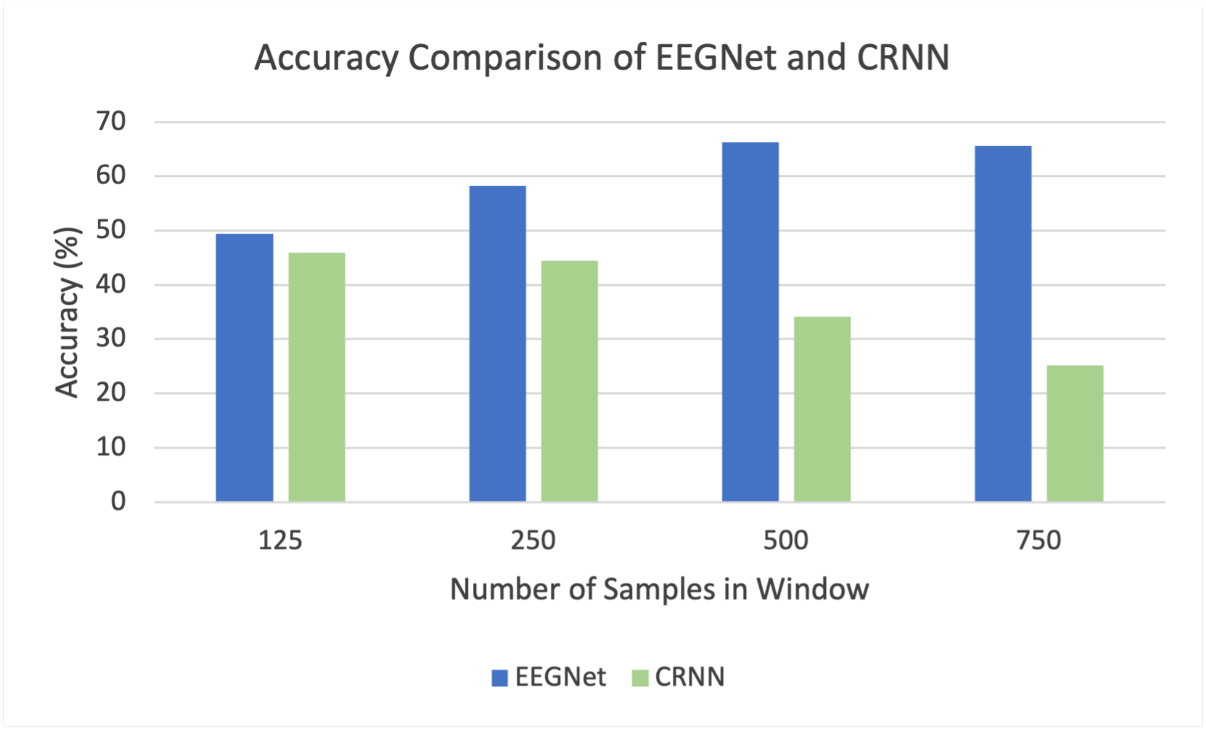


Figure 4.10: Comparing the across-subject accuracies of EEGNet and the hybrid CNN-RNN on the single-window datasets. EEGNet outperforms the hybrid model on all window lengths. The hybrid model performs best on the shorter windows.

Chapter 5

Discussion

5.1 Task-only data trends

The aggregate results of the windowed and augmented datasets indicate MI signals deteriorate in strength as the subject continues to perform the task. Our analyses show this trend occurs uniformly across subjects, despite EEG data having notoriously high across-subject variability, suggesting the cause lies within the MI signal or experimental design rather than a subject-specific phenomenon.

A potential cause is the effect of a visual ERP on model performance. Because the dataset we use here employs a cue to signal the start of a new MI task, there is the risk of a P300 signal occurring in the MI data. Since ERPs like the P300 cause a strong amplitude modulation in the signal, we would expect EEGNet to subsequently detect it easily. Interestingly, the single 125-sample window that achieves the highest accuracy is the one that spans samples 62 through 187 (Figure 4.5a) and corresponds to a 300-millisecond offset from the visual cue as we would expect for a P300 signal. Additionally, this same window experiences the largest accuracy decrease between training/testing on a single window and training on all (Figure 4.6, 54.61 percent compared to 37.28 percent, respectively). The fact that the performance decreases in this experiment suggests the type of features found in this window are not found across the entire four-second dataset and could also be cause by a P300 artifact. Ideally, another set of experiments would be performed without any task-start cue to control for a P300 signal. Unfortunately, designing such an experimental paradigm is nearly impossible as any cue, whether auditory, tactile, or visual, will cause a P300 signal and un-cued tasks would be difficult to label. Rather, an in-depth analysis of the frequency bands and spatial location of the learned features of EEGNet could provide more insight as to the root cause.

Another potential cause of the decrease in signal fidelity is the decay of ERD and ERS over time. MI task classification algorithms like FBCSP and EEGNet are designed to learn the ERD and ERS patterns associated with each task. Our results show that the ERD/ERS features the model learns decay over time, suggesting the amount of amplitude modulation caused by ERD/ERS in sensorimotor rhythms also decays as the subject continuously performs the imagined movement. C. S. Nam et. al. showed that lateralization of ERD is significantly affected by MI task duration [24]. The amount of contralateral dominance of ERD is weaker for prolonged MI tasks. Given that EEGNet learns spatial features, if these features were to weaken over time, it follows that the classification accuracy would also decrease. G. Pfurtscheller et. al. show that the ERS power in the alpha band decays especially for feet and tongue MI tasks [31]. These previously published results corroborate our findings and provide some physiological explanation to the results we see in the present study. Further studies on the effect of continued task performance on ERD and ERS in sensorimotor rhythms could provide more clarity to these phenomena.

Regardless of the physiological cause of the decrease in model performance, our results indicate four seconds of data collection is more than enough time for a BCI system to make a decision about the MI task its user is performing. In fact, our results suggest two seconds of data would be sufficient and potentially improve model performance. These results have profound implications for the design of online BCI system. For a four-class system, a user could make four keyboard selections in two seconds of data collection.

5.2 CNN-RNN network performance

The performance of our novel CRNN network on the BCI Competition Dataset suggest this network could be used to classify shorter sequences of data. EEGNet is a proven deep learning model for EEG data classification and the fact that our novel network was able to achieve similar accuracy levels on the 125-sample dataset on a four-class task suggests the architecture is well designed for this type of data.

While the CRNN's 45 percent accuracy on the 125-sample dataset is substantially lower than the accuracy EEGNet achieved on the 500-sample dataset, 125 samples contains only one quarter of the data as the 500-sample dataset. Reducing the complexity of the problem from four classes to three or two could potentially bring the performance back to EEGNet's levels. A BCI system that can make binary decisions every half-second is just as fast as one that can make an eight-bit decision every two seconds. More testing is needed to prove our CRNN's ability to classify two-class data, but this could make up for the performance discrepancy.

Chapter 6

Conclusions

Our goal of this study was to investigate the practicality of using a deep learning model in a motor imagery-based BCI system. In analyzing the effect of temporally shortening the training and testing datasets we prove that the CNN EEGNet performs just as well on two seconds of data collection as it does four seconds. Further breaking down the signal suggests this result is due to a decay in signal features that occur after roughly two seconds of MI task performance. Concurrently, we designed and optimized a hybrid CNN-RNN network in an attempt to develop a model better suited to temporally short sequences of data. While the model tested here was unable to achieve accuracies as high as EEGNet, it was able to discern some discriminatory features from the dataset. The fact that the model performance was best on a very short dataset suggest with further architecture design optimization a hybrid CNN-RNN could be a powerful classification tool for EEG data. Taken together, our results suggest deep learning models are appropriate classification tools for an EEG-based BCI system, where the amount of time required to determine the user's intention is crucial. Further directions of this work should include corroborating these findings in an online analysis.

Bibliography

[1] Laura Acqualagna and Benjamin Blankertz. “Gaze-independent BCI-spelling using rapid serial visual presentation (RSVP)”. In: *Clinical Neurophysiology* 124.5 (2013), pp. 901–908.

[2] Kai Keng Ang et al. “Filter bank common spatial pattern (FBCSP) in brain-computer interface”. In: *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. IEEE. 2008, pp. 2390–2397.

[3] Clemens Brunner et al. “BCI Competition 2008–Graz data set A”. In: *Institute for Knowledge Discovery (Laboratory of Brain-Computer Interfaces), Graz University of Technology* 16 (2008), pp. 1–6.

[4] Peter Brunner et al. “Does the ‘P300’speller depend on eye gaze?” In: *Journal of neural engineering* 7.5 (2010), p. 056013.

[5] Francois Chollet et al. *Keras*. 2015. URL: `https://github.com/fchollet/keras`.

[6] Emanuel Donchin, Walter Ritter, W CHEYNE McCallum, et al. “Cognitive psychophysiology: The endogenous components of the ERP”. In: *Event-related brain potentials in man* 349 (1978), p. 411.

[7] Monica Fabiani et al. "Definition, identification, and reliability of measurement of the P300 component of the event-related brain potential". In: *Advances in psychophysiology* 2.S 1 (1987), p. 78.

[8] Lawrence Ashley Farwell and Emanuel Donchin. "Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials". In: *Electroencephalography and clinical Neurophysiology* 70.6 (1988), pp. 510–523.

[9] Oliver Faust et al. "Deep learning for healthcare applications based on physiological signals: A review". In: *Computer methods and programs in biomedicine* 161 (2018), pp. 1–13.

[10] Keinosuke Fukunaga. *Introduction to statistical pattern recognition.* Elsevier, 2013.

[11] Christoph Guger, Herbert Ramoser, and Gert Pfurtscheller. "Real-time EEG analysis with subject-specific spatial patterns for a brain-computer interface (BCI)". In: *IEEE transactions on rehabilitation engineering* 8.4 (2000), pp. 447–456.

[12] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: `10.1038/s41586-020-2649-2`. URL: `https://doi.org/10.1038/s41586-020-2649-2`.

[13] Marc Jeannerod et al. "The representing brain: Neural correlates of motor intention and imagery". In: *Behavioral and Brain sciences* 17.2 (1994), pp. 187–201.

[14] Marc Jeannerod. "Mental imagery in the motor context". In: *Neuropsychologia* 33.11 (1995), pp. 1419–1432.

[15] Matthew C Kiernan et al. "Amyotrophic lateral sclerosis". In: *The lancet* 377.9769 (2011), pp. 942–955.

[16] ZJ Koles, JC Lind, and ACK Soong. “Spatio-temporal decomposition of the EEG: a general approach to the isolation and localization of sources”. In: *Electroencephalography and clinical Neurophysiology* 95.4 (1995), pp. 219–230.

[17] Zoltan J Koles and Anthony CK Soong. “EEG source localization: implementing the spatio-temporal decomposition approach”. In: *Electroencephalography and clinical Neurophysiology* 107.5 (1998), pp. 343–352.

[18] Dean J Krusienski et al. “A comparison of classification techniques for the P300 Speller”. In: *Journal of neural engineering* 3.4 (2006), p. 299.

[19] Vernon J Lawhern et al. “EEGNet: a compact convolutional neural network for EEG-based brain–computer interfaces”. In: *Journal of neural engineering* 15.5 (2018), p. 056013.

[20] Martin Lotze and Ulrike Halsband. “Motor imagery”. In: *Journal of Physiology-paris* 99.4-6 (2006), pp. 386–395.

[21] Martın Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `https://www.tensorflow.org/`.

[22] Tomáš Mikolov et al. “Recurrent neural network based language model”. In: *Eleventh annual conference of the international speech communication association*. 2010.

[23] Johannes Müller-Gerking, Gert Pfurtscheller, and Henrik Flyvbjerg. “Designing optimal spatial filters for single-trial EEG classification in a movement task”. In: *Clinical neurophysiology* 110.5 (1999), pp. 787–798.

[24] Chang S Nam et al. “Movement imagery-related lateralization of event-related (de) synchronization (ERD/ERS): motor-imagery duration effects”. In: *Clinical Neurophysiology* 122.3 (2011), pp. 567–577.

[25] Luis Fernando Nicolas-Alonso and Jaime Gomez-Gil. “Brain computer interfaces, a review”. In: *sensors* 12.2 (2012), pp. 1211–1279.

[26] Keiron O’Shea and Ryan Nash. “An introduction to convolutional neural networks”. In: *arXiv preprint arXiv:1511.08458* (2015).

[27] Natasha Padfield et al. “EEG-based brain-computer interfaces using motor-imagery: Techniques and challenges”. In: *Sensors* 19.6 (2019), p. 1423.

[28] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[29] Luis Perez and Jason Wang. “The effectiveness of data augmentation in image classification using deep learning”. In: *arXiv preprint arXiv:1712.04621* (2017).

[30] Gert Pfurtscheller and Christa Neuper. “Motor imagery and direct brain-computer communication”. In: *Proceedings of the IEEE* 89.7 (2001), pp. 1123–1134.

[31] Gert Pfurtscheller et al. “Mu rhythm (de) synchronization and EEG single-trial classification of different motor imagery tasks”. In: *NeuroImage* 31.1 (2006), pp. 153–159.

[32] Walter S Pritchard. “Psychophysiology of P300.” In: *Psychological bulletin* 89.3 (1981), p. 506.

[33] Aya Rezeika et al. “Brain–computer interface spellers: A review”. In: *Brain sciences* 8.4 (2018), p. 57.

[34] Simanto Saha and Mathias Baumert. “Intra-and inter-subject variability in EEG-based sensorimotor brain computer interface: a review”. In: *Frontiers in computational neuroscience* 13 (2020), p. 87.

[35] Dominik Scherer, Andreas Müller, and Sven Behnke. “Evaluation of pooling operations in convolutional architectures for object recognition”. In: *International conference on artificial neural networks*. Springer. 2010, pp. 92–101.

[36] Robin Tibor Schirrmeister et al. “Deep learning with convolutional neural networks for EEG decoding and visualization”. In: *Human brain mapping* 38.11 (2017), pp. 5391–5420.

[37] Hooman Sedghamiz. “BioSigKit: a matlab toolbox and interface for analysis of biosignals”. In: *Journal of Open Source Software* 3.30 (2018), p. 671.

[38] Xinjie Shi et al. “Hybrid Convolutional Recurrent Neural Networks Outperform CNN and RNN in Task-state EEG Detection for Parkinson’s Disease”. In: *2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. IEEE. 2019, pp. 939–944.

[39] Piotr Stawicki et al. “A novel hybrid mental spelling application based on eye tracking and SSVEP-based BCI”. In: *Brain sciences* 7.4 (2017), p. 35.

[40] Chuanqi Tan et al. “Multimodal classification with deep convolutional-recurrent neural networks for electroencephalography”. In: *International Conference on Neural Information Processing*. Springer. 2017, pp. 767–776.

[41] Pauli Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[42] Ping Wang et al. “LSTM-based EEG classification in motor imagery tasks”. In: *IEEE transactions on neural systems and rehabilitation engineering* 26.11 (2018), pp. 2086–2095.

[43] Yu Zhang et al. “Optimizing spatial patterns with sparse filter bands for motor-imagery based brain–computer interface”. In: *Journal of neuroscience methods* 255 (2015), pp. 85–91.