Data Representation for Motor Imagery Classification

by

Andrew Festa

Submitted to the
B. Thomas Golisano College of Computing and Information Sciences
Department of Computer Science
in partial fulfillment of the requirements for the
Master of Science Degree
at the Rochester Institute of Technology

Abstract

While much progress has been made to the advancement of brain-controlled interfaces (BCI), there remains an information gap between the various domains involved in progressing this area of research. Thus, this thesis seeks to address this gap through creation of a method of representing brainwave signals in a manner that is intuitive and easy to interpret for both neuroscientists and computer scientists. This method of data representation was evaluated on the ability of the model to accurately classify motor imagery events in a timely manner.

Acknowledgments

This is the acknowledgements text

This is the dedication text

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recent advances in the hardware required for small-scale and non-intrusive methods of measuring brain activity offer an unprecedented level of potential for the development of brain-controlled interfaces (BCI). Where this type of technology used to be accessible only to the professional medical community [2, 6], hobbyists are now able to approach this domain as a viable method of control. NeuroSky and Emotiv both provide cost-effective boards for recording electroencephalograms (EEG) for developers to use for experiments along with a thriving community for novices and experts alike. OpenBCI takes this a step further by open-sourcing both the software and the hardware for their boards, the Ganglion and the Cyton.

Despite the explosive growth of the field since the 90's, drawing meaning from the understanding of the brain remains a difficult challenge. The open-source community tends to focus on interpreting the signals in an effort to create control systems requiring thought alone. This is an attempt to solve the inverse problem in EEG, where we try to infer the inputs given a set of outputs, leaving open the forward problem, where we attempt to discern what types of outputs we can expect to see given a set of inputs. The forward problem is left to large research labs with extensive resources, but to create a truly effective BCI, both the forward and inverse problems in EEG must be addressed in tandem to ensure that the system operates based on theoretical truths of the functionality of the brain.

1.1 Problem Statement

The potential for BCI is often characterized by its potential to offer a unique means of control and communication in that it requires no muscle movement by a user [6]. Development of such a system inherently requires an understanding of theoretical neuroscience as well as classification and data analysis techniques [3]. This requires teams to be proficient in numerous knowledge domains running the gamut from neuroscience to computer science to information theory, leading to a necessity for communication between these various domains. Communication between different numerous individuals with differing areas of expertise inevitably leads to a breakdown in the ability to convey information across domain boundaries.

One solution teams will often take to avoid the issue of the breakdown of communication is to limit the areas of expertise required to solve a problem. That is, teams of neuroscience will progress the capabilities of BCI systems by advancing the theoretical neuroscience, while teams of data scientists will attempt to improve classification and recognition of the event using data analysis techniques [7]. This means that the domains remain disparate and sharing information is wrought with the potential for misunderstanding due to differing experiences and language use.

An arguably more robust solution is to form a team with a diverse skill-set. Unfortunately, a team can only grow so large before becoming unmanageable, and there will still inevitably exist a gaps in expertise areas due to the complex nature of such a system.

1.2 Motivation

Based on the aforementioned concern, it would prove fruitful to provide both researchers and hobbyists a common means of understanding between the distinct theoretical aspects involved in advancing the capabilities of BCI systems [1]. That is, provide a method for the subject matter experts, in this case, the neuroscientists, to communicate and provide the information and data to the computer scientists in such a manner that it is able to leverage the current state-of-the-art methods for data analysis and classification.

Neural networks are often cited as being black boxes in the sense that it is difficult to reason as to the how or why certain predictions are made, though progress is being made to this end, particularly with regards to the domain of computer vision. If the knowledge representation were able to be more closely aligned to the progress currently being made by the field of computer vision, it would offer the ability for the domain of neuroscience to leverage the research currently being undertaken by this field of computer science [5]. In this way, the inner-workings of the brain can be more directly explored and may offer insights with regards to the forward and inverse problems of EEG.

1.3 Thesis Statement

The work for this thesis will attempt to address the issue of bringing together solutions for the forward and inverse problems in EEG. By offering a novel method of representing the multivariate electrical signals captured by EEG systems, developers and researchers outside of large, heavily funded medical institutions will be able to better advance their models due to a better understanding of the theoretical operation of the brain. The proposed method of presenting this data is similar to the form taken by spectrograms and will leverage the current understanding of the mind by building upon a common method of representing EEG signals: EEG montages. To this end, neuroscientists, computer scientists, and hobbyists will all be able to more readily interpret the complex EEG signals, facilitating the advancement of research both towards understanding the function of the brain as well as how to use that understanding to develop better and more robust brain-controlled interfaces.

1.4 Objectives

My primary focus in conducting this research work is to present a novel method for representation of EEG data that is understandable for subject matter experts in separate domains. To this end, I will provide a library which allows for data to be quickly and easily imported for analysis and to efficiently generate the spectrograms for deeper analysis or classification.

BCI systems are intended to offer a method of communication and control

to an outside environment [4]. Any method of data representation of EEG should be able to be used towards this task. This desired characteristic will be tested by using the spectrograms of the EEG signals to train a classifier that is able to distinguish between two motor imagery event-related potentials: moving left and moving right. This will allow a user to control a character through a maze on a computer using only their mind as the control input.

Chapter 2

Methodology

There are several components involved in building out a BCI system. Namely, it is required that data is able to be accurately recorded, labeled, and stored for training and future analysis. Additionally, the data must be able to be analyzed in near-real-time when acting as a control system.

2.1 Experiment Phases

In order to address these stages of building out a BCI system, the thesis work was broken into three main phases: data acquisition, data preprocessing, and model classification. The first phase is where the actual raw data for the rest of the project was collected and stored for analysis. For manually recorded data, a low-cost board from OpenBCI was used, and the data collected was compared against a reference dataset that used the BCI2000 system for data acquisition. All the EEG data was stored as CSV files for ease of visual inspection.

The second phase transformed that data into the proposed data representation. The signals were broken into windows of different time-lengths: 0.20, 0.40, 0.60, 0.80 and 1.00 seconds. It was these windows which were used to build out the signal images.

The final phase acted on the signal images in order to build a model to classify different input events. It was these signal images which were fed to the classifier for training and prediction in different combinations in order to evaluate how the system performed when presented with variable time-lengths following the onset of an event. Additionally, this phase evaluates the system in order to determine the efficacy of the proposed data representation for the task of a control system.

This chapter covers in greater detail the steps taken from the start of recording data to the end of training a classifier and using it to perform predictions on a signal image.

2.2 Data Acquisition

There are a myriad of challenges to overcome when performing the recording sessions necessary for gathering data for BCI systems. For the non-neuroscientist, this presents a further issue that it can be difficult to verify if a recording session gathered adequate clean data and how to inspect the signals for abnormalities and artifacts.

2.2.1 Physio Dataset

In order to reduce uncertainty as to whether errors may be due to the data representation or the recorded data, an external dataset was used to first build out and evaluate the proposed method of data representation. This dataset, called the *EEG Motor Movement/Imagery Dataset*, is provided for research use by PhysioNet [**?**,8]. Not only is this dataset expertly collected and meticulously verified, but it is also used by several solutions for evaluation of BCI systems, making it a perfect candidate not only for verifying the efficacy of the proposed method for classification of motor imagery events, but also for comparison against current techniques for performing the same task.

Recording Session Protocol

This dataset was built using the BCI2000 system, which is a mid-high level system compared to the Ganglion, the OpenBCI board used for manual data collection. This system has 64-channels, compared to the Ganglion's 4 channels, and is more aimed towards neuroscientist researchers rather than the open-source and hobbyist communities. The actual data recorded is comprised of six different trial types for 109 different subjects, and each trial is either one or two minutes in length, depending on the trial type. The actual trial types, along with Physio's description of each trial, are shown in table 2.1. Each of the baseline trials are two minutes in length, while the remaining trials are 3 minutes in length.

Trial Type	Description
Baseline	Baseline recording of brain activity when the user's eyes are open
Baseline	Baseline recording of brain activity when the user's eyes are closed
Motor execution	A target appears on either the left or the right side of the screen. The subject opens and closes the corresponding fist until the target disappears. Then the subject relaxes.
Motor imagery	A target appears on either the left or the right side of the screen. The subject imagines opening and closing the corresponding fist until the target disappears. Then the subject relaxes.
Motor execution	A target appears on either the top or the bottom of the screen. The subject opens and closes either both fists (if the target is on top) or both feet (if the target is on the bottom) until the target disappears. Then the subject relaxes.
Motor imagery	A target appears on either the top or the bottom of the screen. The subject imagines opening and closing either both fists (if the target is on top) or both feet (if the target is on the bottom) until the target disappears. Then the subject relaxes.

Table 2.1: PhysioNet Trial Types

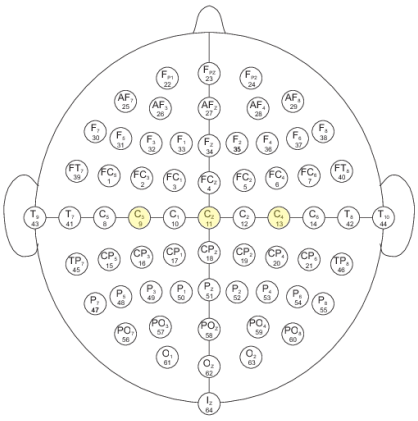


Figure 2.1: Physio Electrode Positions

Subtleties and Nuances

When dealing with this dataset, there are a few small details that bear mentioning as they influence design decisions.

1. Each subject performed each baseline trial once and each non-baseline trial 3 times, resulting in a grand total of 1526 recorded trials. However, not all of these trials were used in the development and evaluation of the proposed system. The only trials used were where the subject was instructed to imagine moving either their right or left hand. The baseline trials were purposefully ignored in order to try and limit the ability of the system to differentiate between rest and normal brain activity versus the desired events.

2. The BCI2000 system samples each electrode at 160 Hz. This value is used to correlate sample index to time as well as used in computations of window lengths, which is explained in greater detail in 2.3.

3. A FIR band-pass filter was applied to the signals to only allow frequencies between 5 to 50 Hz. This frequency range was selected as it allows for limiting the signals to the frequencies of interest and is one of the ranges provided by OpenBCI filters, which was used when manually recording data.

4. The events are annotated as the three strings shown in table 2.2:

Event	Description
T0	Rest
T1	Onset of motion of the left fist
T2	Onset of motion of the right fist

Table 2.2: PhysioNet Events

5. The data is provided as EDF files, which is a common format for providing this type of data. This is relevant as this data format is supported by the MNE library, but the relevant information is extracted using this library and then transformed as part of the preprocessing steps.

6. The electrodes were placed on the subject according to the international 10/10 system, as shown in figure 2.1. However, due to the theoretical considerations outlined previously in section **??**, only the $C3$, Cz, and $C4$ channels were used when building the system. However, the reference implementation provided by MNE uses all 64 channels for evaluating the features extracted.

2.2.2 Manually Recorded Dataset

Part of the desired outcome for this research was to explore the ability to use the proposed data format for development of a BCI system. As such, there has to be at least some capability to record and analyze EEG activity in real-time. This was done using the Ganglion board with the Ultracortex "Mark IV" EEG Headset, shown in figure 2.3. The user was connected to the headset and presented with a random stimulus on a timer. They were then expected to imagine moving either their right or left hand, based on the stimulus presented. Additionally, there was a "rest" prompt, to which they were expected to just relax and not imagine any action.

Ganglion and Headset Configuration

One of the earlier challenges to overcome was actually getting the EEG data into a coding environment in a real-time manner. OpenBCI provides a GUI

for visualization of the recorded signals, and they have a Python library that is meant to allow developers to interface with the board using Python. Unfortunately, it relies on a bluetooth library that is only compatible with linux and Mac. While not an insurmountable issue, it was desired that the system be able to work cross-platform as long as a solution was able to be found that would allow for it.

The first two approaches were to try and communicate directly with the board or with the Electron hub that the OpenBCI GUI uses to communicate with the board, thereby bypassing the GUI entirely. The first solution was rejected due to the fact that Simblee board, which is the actual breakout board used by the Ganglion, was discontinued, making it difficult to find compatible drivers for interfacing with the board directly.

The second option of communicating with the Electron hub was found to be a suitable solution and was pursued with a fair degree of success; It was able to successfully establish a connection with the board and record the desired data from the board. The reason for moving away from this solution was due to the complexity when presented with another solution that was simpler and more stable.

The final method of acquiring data in a real-time manner was by having the OpenBCI GUI establish the connection to the board and then then having the Python environment communicate with the OpenBCI GUI over UDP using the Networking widget depicted in the bottom right of figure 2.2. This provides the benefit of a simple solution that is suitable for the desired purpose as well as allowing for use of OpenBCI's filtering capabilities.

The filtering capabilities of the GUI can be seen towards the top left of figure 2.2. The first is a notch filter that is meant to filter out noise from mains lines, which operate at 60 Hz in the USA. The second filter is a band-pass filter that allows for several different ranges.

The last benefit of using the GUI was in ease of checking the impedance values for each electrode using the widget shown in the bottom left of figure 2.2. As discussed in section **??**, we want these impedance values to be as low as possible in order to help ensure that the signals of interest flow to ground through the electrodes rather than over the surface of the user's head.

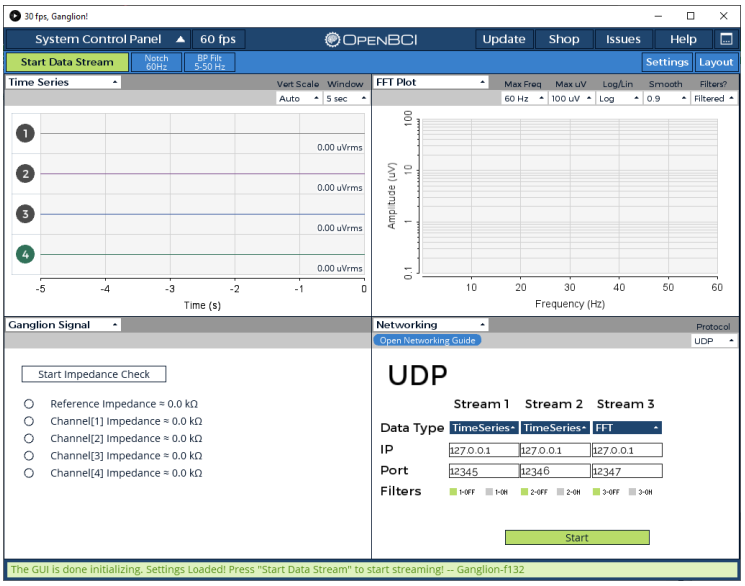


Figure 2.2: OpenBCI GUI

As previously mentioned, the headset used was the Ultracortex "Mark IV" headset from OpenBCI, shown in figure 2.3. While this headset supports sampling up to 16 channels of EEG from up to 35 different 10-20 locations, the Ganglion only supports up to 4 channels. Mainly for the symmetry, only three channels were used: $C3$, CZ, and $C4$. This leaves open the possibility of using another position as an additional reference, such as using FpZ in order to better detect and remove artifacts due to blinking. However, this was not further explored in the course of this research.

The electrodes were the dry spiky electrodes that come with the headset, and the headset was tightened to the point that it did not shift easily under slight movement, but was not uncomfortable for an extended recording session. Finally, it was found that applying a small amount of electrode gel further decreased the electrode impedance without many of the common pitfalls and troubles of wet electrodes. The final points of note were with regards to impedance checking and signal filtering. Prior to starting a recording session, the impedance was checked for each electrode. If it was over 12 $k\Omega$, the headset was adjusted in order to lower the impedance to a more acceptable level. Finally, the notch filter was also set to block noise at 60 Hz, as this is the frequency of the mains lines, and a band-pass filter was set to filter out all

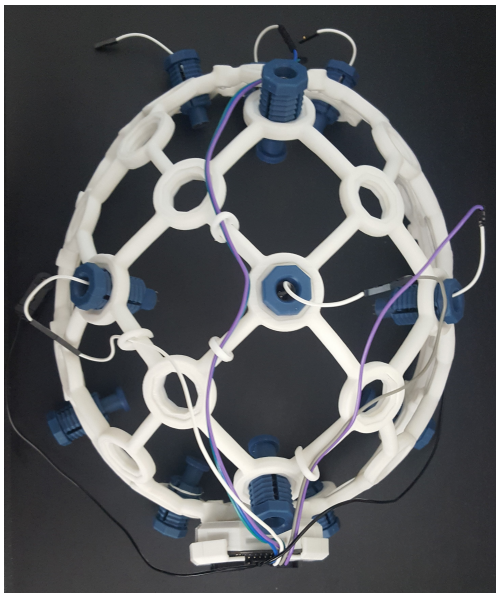


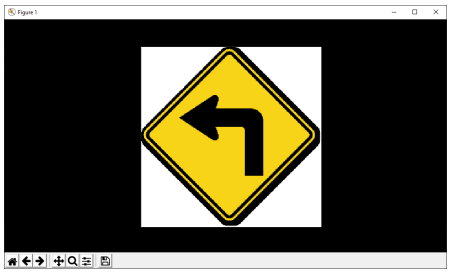
Figure 2.3: Ultracortex Headset and Ganglion Board

frequencies outside the range of 5 to 50 Hz.

Recording Session Protocol

As the Physio dataset is the reference dataset, the method for manually collecting data seeks to emulate that process as much as possible. Specifically, there are several key decisions that the method copies from the creators of the Physio dataset. There are three prompts presented to the subject: 'stop', 'left', and 'right', which are shown in figure 2.4. The subject was instructed to continue to imagine moving the hand that corresponds to the given prompt in the case of the latter two prompts. In the case of the 'stop' prompt, they were to relax and not imagine moving either hand. Both the EEG samples and the events were recorded, and a single sample (which is comprised of three values – one from each electrode) is marked as the current event.

Despite the similarities in the process, there were several changes made to the method of the recording sessions. In the Physio dataset, each event is separated by about 3 seconds. When manually collecting the data, the time



(a) Left Prompt

(b) Right Prompt

(c) Stop Prompt

Figure 2.4: Stimulus Prompts

between each stimulus was set to an average of 5 seconds. Furthermore, a 20% jitter was added to the delay between generating a new prompt. That is, on average, the delay is 5 seconds. But the actual delay for any given delay is $5 \pm 20\%$ seconds, or more exactly, in the range $(4, 6)$ seconds.

The last distinction is in regards to generating the prompt. In the Physio dataset, the subject was aware of the next prompt in the sequence. When manually collecting the data, the user was not aware of the next prompt, and there was no guarantee that the same prompt might appear multiple times in succession.

2.3 Data Processing

When considering the design of the data representation, it is important to keep in mind the actual purpose and use case of a control system. It must be both reactive and accurate, and it would be further desired that it offers some level of variable sensitivity while erring on the side of inactivity. It would

not be suited for most use cases if an erroneous signal was interpreted as an input, thereby causing the system to enter a state of meta-stability as the user attempts to correct for the invalid input and further causing more erroneous inputs to the system.

With this at the forefront of our mind, we can be begin to examine the method by which signals from the differential amplifier are transformed into a signal image that is then fed into the classifier and further used as inputs to the control system.

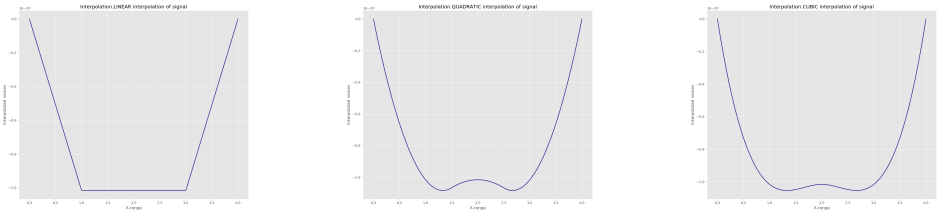
2.3.1 Signal Image Construction

Ignoring the reference channel, there are three signals of interest: $C3$, CZ, and $C4$. These three locations are adjacent to each other over the central sulcus, and at any given point of time, these three signals are 3 values at discrete locations on the head. However, brain activity is not discrete. Rather, it is characterized by magnetic and electric flows and fields. To try and recreate the field, we can apply an interpolation function between these three points. As the field is further influenced by not only these three points, but also all surrounding areas of the brain, it is reasonable to assume that this interpolation function should be of higher-order. However, this is not a strict requirement so long as the interpolation function can operate on three points.

The first step in deciding upon such a function is to define a sort of spacing between each signal point, which we can view as being placed at an x-coordinate along the surface of the head. For demonstration purposes, let's examine three example points at a single point in time and define the spacing to be 25 points between each signal location. Additionally, we will tie each end of the plot to zero as a ground since these locations are physically similar to the reference and driven-ground electrodes on the head.

The next step is to apply our interpolation function to these five points. Scikit-learn has a built in 1D interpolation function that we can use for this purpose. Furthermore, we will specify a dimensionality: linear, quadratic, or cubic. Taking these point as an array results in a 1D array of y-values over x-range, allowing us to visualize a signal as a 2D plot, as shown in figure 2.5

In preparation for the next step, it would be good practice to normalize



(a) Linear (b) Quadratic (c) Cubic

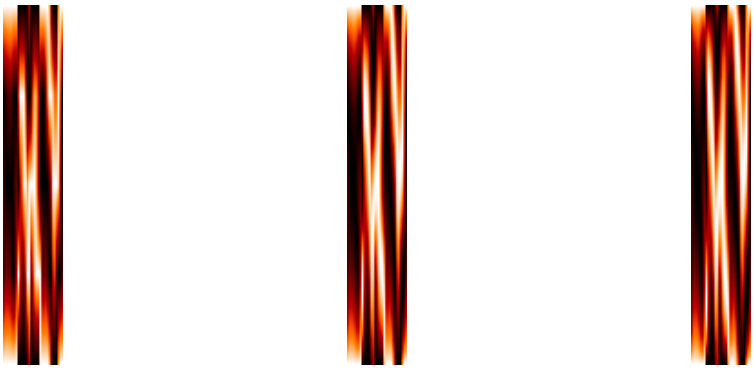
Figure 2.5: Signal plots after applying interpolation

these values to the range $[0, 1)$, and it is at this point that a column for a single time-step has been computed. Concatenating several of these columns together gives us a 2D array of single values, quite similarly to a method of representing a gray-scale image. Furthermore, while the height of the image is defined by the previously decided spacing, the width is defined by the number of time-step columns concatenated together.

While this data representation is ready to be used for the desired purpose, one more modification will be made in order to make the image more intuitively understandable. This is to apply a color-map to the grayscale image in order to better interpret not only signal frequency, but intensity as well. For this purpose, we will use Matplotlib's *gist_heat* color-map, as it is sequential and the color range feels natural with regards to variable intensity. Using the first 16 time steps of subject 'S001' of the physio dataset for trial 4 gives us the images shown in figure 2.6. Note that this is a 'rest' event.

A key point to keep in mind is that the width of an image directly correlates to a length in time. Thus, modifying the width is comparable to modifying the reactivity of the system as a longer window means it will require more time-steps following an input for the system to build the signal image. It is in this way that we can evaluate the ability of the system to react quickly or slowly to an event.

Finally, if each column of the image is effectively a time-step, then moving laterally across an image is effectively evaluating the signal in the frequency domain, as was discussed previously in section **??**. It can be expected a shorter window would prove more difficult for a model to correctly classify due to not having enough samples while a longer window may present a challenge as it



(a) Linear (b) Quadratic (c) Cubic

Figure 2.6: Signal images of start of S001 trial

may contain multiple events or parts of signals that the classifier attributes to certain types of events.

2.3.2 Signal Image Labeling

After an image has been constructed but before feeding it to the supervised classifier, it is necessary to label each image as an event: 'rest', 'left' or 'right'. To this end, it is best to take a step back and recall that a signal image is comprised of multiple signals where each sample is labeled as a singular event. This leaves open the possibility that a window could contain multiple events. While not necessarily an issue, as we could use a multi-label classifier, this is not the way we'd want to handle this case for a control system. It wouldn't make much sense for a window to be both a 'rest' and a 'left' input. Instead, we will label a window as the most common label that appears in a window. Compare this method of labeling a window with the other option of labeling a window as the same event as the last sample that appears in the window. While the latter option might feasible offer better reactivity to the onset of an event, the decided upon approach should prove more suited to accurate classification as an image is more representative of the event.

2.4 Classification

The last phase in the process of creating the BCI system is to train a classifier that is able to quickly and accurately classify a control input. That is, when fed a signal image that is labeled as either 'right' or 'left', it should be able to interpret that image as an input without a noticeable time delay.

2.4.1 Model Architecture

Many different types of models have been used for motor imagery classification. From logistic regression to recurrent neural networks, these models span the gamut in terms of understandability and interpretability. As discussed in section **??**, there has been substantial recent progress in understanding neural networks, particularly convolutional neural networks, which have proven particularly effective in the domain of computer vision. As the data representation transforms EEG signals into images, the model of choice for the classifier was a CNN. However, where typical machine learning research projects dealing with neural networks explore different model architectures, the architecture for this research was chosen to be small and simple in order to evaluate the baseline efficacy of the data representation rather than how to construct a model particularly suited to the task of motor imagery classification. The actual architecture of the model is shown in figure 2.7.

In the first layer, the CNN expects an RGB input image that is $224\,x\,224$. Recall that the width of the signal image is dependent on the number of time steps of the signal. One option to deal with this disparity is to train a different model for each image width and then create a hierarchical model where each model operates on a different time scale. This option would be building a model that is particularly suited to the task rather than a model for testing the efficacy of the data representation. The better option is to simply resize each image to a particular width and height – in this case, $224\,x\,224$. This presents an interesting change to each image, as resizing an image to a different width has two pronounced effects when applied to the information captured by a signal image. The first is that directly resizing an image effectively applies a secondary interpolation to a signal image, whereas the second is that each time step is effectively scaled in the frequency domain.

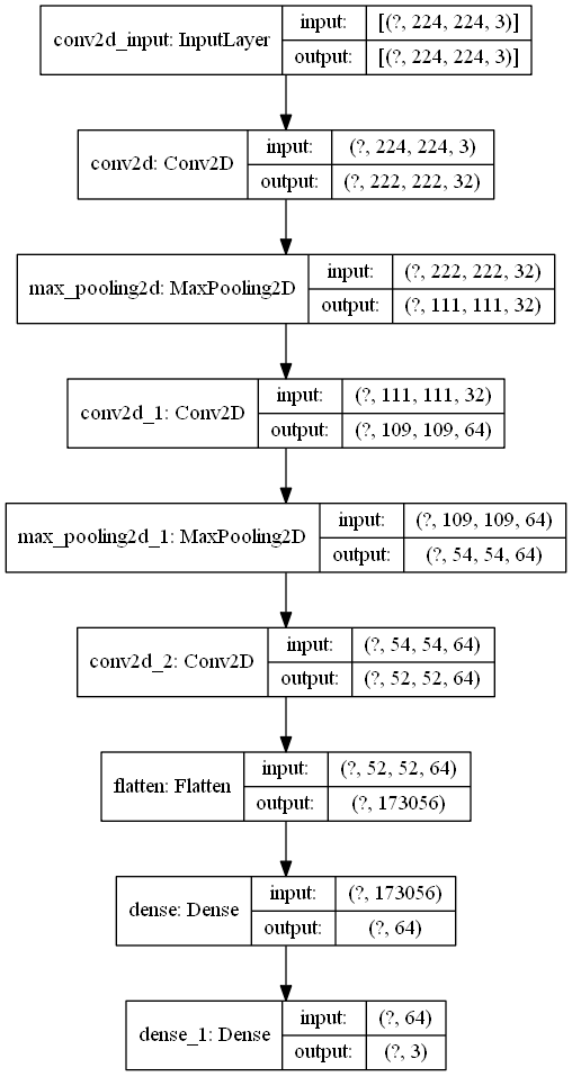
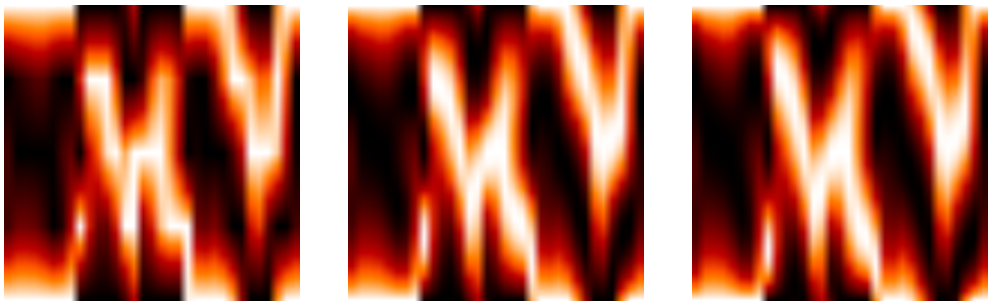


Figure 2.7: Convolutional Neural Network Architecture



(a) Linear (b) Quadratic (c) Cubic

Figure 2.8: Resized Signal Images

The effects of these changes are best seen through example. Consider the images shown in figure 2.8, which show the same signal images from figure 2.6 after having been resized. The additional interpolation can most easily be seen in the linear image, towards the middle of the image. In the original image, the black cleft is more pronounced, whereas in the resized image, this same cleft has a horizontal component added, which is not present in either the quadratic or cubic resized images, despite a similar cleft being present in the originals.

The more interesting change is the second: the effective scaling in the frequency domain. This is of particular importance due to the fact that traditional feature extraction techniques tend to operate on the frequency domain. For example, as discussed in section **??**, motor imagery can be detected based on Mu-rhythm suppression. After scaling, from a high-level perspective, a model can be expected to interpret each image as being on a comparable time length, even if the original images are of different time lengths. Thus, the frequency represented by each image, which can be visually seen as a horizontal change from black to white, is fundamentally altered after resizing the image.

2.4.2 Training

As with building the model, when training the model, the training parameters were chosen to prioritize simplicity and interpretability. There are only 5 main parameters, with regards to the model, that were configured during the training step: maximum number of epochs, learning rate, batch size, optimizer

technique, and loss function. A further four parameters were specified that dealt with the data that was fed to the model during training: interpolation types, data source, subject name, and window lengths. In all, the only one that was varied across trials was the window lengths. All others were set to the default values shown in table 2.3. These were found to work passably well. This allowed for the anaylsis of a model to focus primarily on the effect of window sizes on the ability of the classifier to accurately predict events.

Parameter	Default value
Maximum number of epochs	20
Learning Rate	$1E-5$
Batch size	16
Optimizer	Adam
Loss function	Sparse categorical accuracy
Interpolation types	Linear, quadratic, and cubic

Table 2.3: Model Training Parameters

The three data parameters control the data that is fed to the classifier during training. Together, data source and subject name uniquely identify an actual set of trials performed by an individual, either from the Physio dataset or the manually recorded data. The window lengths specify a list of different sized signal images generated from those trials. This allows for training a model to be specific to an individual as it is generally expected that different people's brainwaves will appear differently even given the same set of events. While the model's ability to perform classification on groups of individuals was slightly explored during the course of the research project, it was deemed out-of-scope and left for future work.

Finally, it bears mentioning how the data was split. Prior to being fed into the classifier for training, the dataset, built by pulling the images based on the data parameters, was split into a training, validation, and test set at a ratio of $60:15:25$. The training and validation sets were used during training while the test set was held out for evaluating the model's ability to generalize to unseen data. Furthermore, the performance on the validation set was used as the early stopping criteria when training the model. If the validation loss did not improve over the previous three epochs, then the training was stopped, and the model weights were restored to the weights of the model with the lowest validation loss.

2.4.3 Evaluation

By structuring the classification phase of the experiment in this way, it allows for the system's performance to be measured both in terms of accuracy and timing. The first is done by simply comparing the expected result versus the result of the classification based on the user's input. The timing analysis was done by analyzing the model's ability to accurately predict an event based on different combinations of window lengths and looking for trends across these different combinations.

Bibliography

[1] E. Larson D. Engemann D. Strohmeier C. Brodbeck L. Parkkonen M. Hmlinen A. Gramfort, M. Luessi. Mne software for processing meg and eeg data, February 2014.

[2] Sarah N. Abdulkader, Ayman Atia, and Mostafa-Sami M. Mostafa. Brain computer interfacing: Applications and challenges. *Egyptian Informatics Journal*, 16(2):213 – 230, 2015.

[3] Haider Hussein Alwasiti, Ishak Aris, and Adznan Bin Jantan. Brain computer interface design and applications: Challenges and future. 2010.

[4] Rehab Ashari and Charles Anderson. Eeg subspace analysis and classification using principal angles for brain-computer interfaces. pages 57–63, 12 2014.

[5] Ikhtiyor Majidov and Taegkeun Whangbo. Efficient classification of motor imagery electroencephalography signals using deep learning methods. In *Sensors*, 2019.

[6] D.J. McFarland and J.R. Wolpaw. Eeg-based braincomputer interfaces. *Current Opinion in Biomedical Engineering*, 4:194 – 200, 2017. Synthetic Biology and Biomedical Engineering / Neural Engineering.

[7] S. S. R, J. Rabha, K. Y. Nagarjuna, D. Samanta, P. Mitra, and M. Sarma. Motor imagery eeg signal processing and classification using machine learning approach. In *2017 International Conference on New Trends in Computing Sciences (ICTCS)*, pages 61–66, Oct 2017.

[8] G. Schalk, D. J. McFarland, T. Hinterberger, N. Birbaumer, and J. R. Wolpaw. Bci2000: a general-purpose brain-computer interface (bci) sys-

tem. *IEEE Transactions on Biomedical Engineering*, 51(6):1034–1043, June 2004.

Appendices

Appendix A

Glossary

Table A.1: Autogenerated table from .csv file.

$Value1$ A	$Value1$ V	$Value1$ A	$Value1$ V	$Value1$ V
0	1.05	0.49	1.03	0.52
1	1.00	0.50	0.99	0.56
2	0.97	0.53	0.95	0.60
3	0.91	0.56	0.94	0.62
4	0.82	0.61	0.84	0.65
5	0.67	0.69	0.77	0.65
6	0.49	0.78	0.58	0.71
7	0.34	0.82	0.58	0.73
8	0.22	0.88	0.47	0.78
9	0.18	0.89	0.43	0.80
10	0.12	0.92	0.34	0.86
11	0.10	0.92	0.32	0.86
12	0.09	0.94	0.33	0.88
13	0.08	0.94	0.34	0.88
14	0.06	0.95	0.31	0.88
15	0.06	0.95	0.29	0.90
16	0.05	0.95	0.31	0.91
17	0.05	0.95	0.35	0.90
18	0.04	0.95	0.35	0.91