MindGames

Senior Capstone Project

Brandon Knotek, Walid Muhammad, Jack Wilder

Hood College Department of Computer Science & Information Technology

Spring 2022

(Rough draft; content and formatting subject to change)

Introduction

Software is an essential part of human society. There is an application or service involved in nearly every aspect of everyday life, from the simulations for experiments used in education to the banking apps used to save for retirement. While almost all of this software can be efficiently used by the general public, where they fall short is in the vital area of accessibility. For general purpose software, only able-bodied people are typically taken into account during the development process; there are still thousands if not millions of people in the United States that require additional methods of control to navigate this vital technology.

Our project centered around this key lack of a more accessible method for controlling software. Electroencephalography (EEG) hardware and an accompanying EEG processing pipeline allow a user to simply put on a headset, train the pipeline to match their unique brain patterns, then dispatch commands to a computer using only thoughts or subtle movements. This ideally replaces the need for voice activation or extra physical peripherals that may be impossible to use for certain physically challenged individuals when using a phone or computer.

Literature Review

Over the past few decades, there has been significant research into Brain-Computer interfaces, commonly referred to as BCI. The applications for this technology have a wide array of uses in the military, accessibility, and consumer/business usage [10][2][8]. There has been significant research into classification using machine learning coupled with a BCI system, such as mental task classification, sleep-state classification, and emotional classification [9]. Within this research, the main form of classification explored is motor imagery (MI) classification.

To begin, Villegas et al. define electroencephalography (EEG) as “a brain monitoring technique, which receives signals from the cortex of the brain through electrodes placed on the scalp” [10]. EEGs are essential to the BCI process as they provide the input for the system in the form of the brain’s electrical signals, which are an integral part of the signal acquisition process. For a person to be able to control an object, specifically in this scenario a game of Pac-Man, a communication system is needed to monitor the brain’s activity and process it into what the user wants it to be; this system is referred to as the BCI system. BCIs can be classified as endogenous or exogenous, and as synchronous or asynchronous [5]. For this project, we will be focusing on exogenous (non-invasive) and synchronous (cue-based) BCI only.

The most common model that is used within a BCI to properly classify features consists of four components: signal acquisition, data preprocessing, feature extraction, and classification. First is the signal acquisition, which is concerned with getting EEG data into the BCI. The raw EEG data then needs to be preprocessed in order to reduce noise and eliminate artifacts in preparation for feature extraction. The techniques used in preprocessing or channel selection are different for each type of classification. Specifically for motor task classification, the filtering techniques are “based on EEG signal statistics” [9]. While there are many methods to select channels based on statistics, all rely on analyzing spatial patterns and searching to find an optimal combination of channels. Channel selection is quite difficult according to Alotaiby et al. as “EEG patterns in a BCI system vary from the first session to subsequent sessions on other days due to several subjects’ preconditions” [9]. This requires that the channel selection algorithm be flexible enough that it can be effectively used across several uses.

Feature extraction is “the process of distinguishing the pertinent signal characteristics from extraneous content and representing them in a compact and/or meaningful form, amenable to interpretation by a human or computer” [3], and specifically feature extraction for EEG signals can be described as “the extraction of the most discriminative features [...] followed by classification of such features into different directions” [11]. There are many techniques and methodologies for having the features of a signal extracted and they vary largely on what type of classification is being done. Often, the preprocessed signal is converted into the frequency domain with a Fast Fourier Transform or a Short-time Fourier Transform to see the relevant signal characteristics [11]. Common spatial patterns (CSP) and wavelet transforms are also commonly used in feature extractions. With the most relevant features of the signal exposed, they can be classified into different directions. There are several approaches to how the data can be classified, such as Linear Discriminant Analysis, Support Vector Machines, K-Nearest Neighbors, Deep learning, and Convolutional Neural Networks (CNN) [11]. The combination of a feature extraction technique and a classification technique together can change the accuracy and effectiveness of BCI. Specifically, CNNs are a good classification technique for MI as they are designed to handle larger variations over time and per person [10].

Methods

EEG data was collected using hardware provided by Hood College, including an OpenBCI Cyton Biosensing Board (CBB) and an OpenBCI Ultracortex Mark IV (UM4) headset. Data was collected from eight spike electrodes positioned at FC5, FC1, FC2, FC6, CP5, CP1, CP2, and CP6 (Fig. X). These positions were chosen in order to capture electrical activity originating in the brain’s premotor and primary motor cortices (Fig. X).

EEG data was transmitted via Bluetooth from the CBB to an accompanying RFDuino USB dongle plugged into a personal laptop. The USB dongle was configured to appear to the computer as a standard serial data port running at a rate of 115,200 baud. Upon reaching the computer, EEG data was retransmitted through a Lab Streaming Layer (LSL) broadcast at a rate of 250 Hz using OpenBCI’s openbci_lsl Python module.

In addition to the EEG LSL broadcast, a marker LSL broadcast was required in order to allow a supervised machine learning algorithm (described below) to associate given EEG states with corresponding game commands. To broadcast the proper marker strings, a training module was produced using Python (Fig. X). The module presented a user with a random sequence of movements corresponding to game commands (up, down, left, right, neutral), and transmitted appropriate markers through an LSL broadcast while the user was executing those movements. Training session data (including both EEG and marker LSL streams) was recorded using the LabRecorder application so that it could be used multiple times.

To handle real-time signal processing of EEG data, a processing pipeline was designed using the NeuroPype Pipeline Designer application (Fig. X). The pipeline continuously received EEG and marker data from the LSL broadcasts described above, passed it through a series of processing and analysis nodes, and generated its own LSL broadcast containing the probability of correspondence between game commands and the current EEG state. The pipeline was also capable of reading files containing recordings of LSL broadcasts generated during training sessions.

The processing pipeline was divided into four sections: data acquisition, preprocessing, feature extraction, and classification. The data acquisition component either received live LSL broadcasts or read in pre-recorded broadcasts from a file. Typically, pre-recorded training data was injected into the pipeline initially, with all subsequent data being received from live LSL broadcasts. This component was also responsible for mapping marker strings to indices so that final probabilities could be returned in the form of a list rather than a dictionary.

The preprocessing component was primarily responsible for removing noise from the EEG signals. This was accomplished using a bandpass filter that removed frequencies below 6 Hz and above 32 Hz, as such frequencies are less likely to be related to brain activity. This component was also responsible for defining the start time, relative to a given marker string, and duration, in seconds, of EEG data to associate with the marker string. All pipeline variants defined this window of activity as beginning 0.5 seconds after a marker was received and lasting for 1.5 seconds.

The feature extraction component was responsible for determining which signal components (or “features”) exhibited the most notable variance between a given command type and all other command types. During this process, the raw EEG data was lost and only the identifying features were retained. This component had to be calibrated using training data before it could function correctly.

The classification component was responsible for mapping extracted features to command probabilities. This was achieved using logistic regression which, similarly to the previous component, had to be calibrated using training data before it could function correctly. During this process, all input data was lost. The result was a list of five probabilities, each corresponding to the likelihood of the current EEG state corresponding to a given game command. This list was averaged over a period of 0.5 seconds in an attempt to account for unintended probability spikes, then transmitted via an LSL broadcast at a rate of 2 Hz.

In order to provide an interactive environment in which classification accuracy could be evaluated, a simplified version of Pac-Man was developed using HTML, CSS, and JavaScript (Fig. X). The game ran in a web browser and was supported by a simple Python backend utilizing the Flask framework. The backend received the LSL broadcast originating from the processing pipeline, determined which command probability was the highest, then transmitted the corresponding command to the game via the WebSocket API. To prevent multiple unintentional commands from being sent in rapid succession, the backend was designed to transmit one command at a time, waiting for the user to reach a neutral state before sending another.

Results

Determination of the classification accuracy of a given pipeline configuration was largely subjective, for reasons discussed below. In the latest pipeline configuration, it was estimated that ‘left’ and ‘right’ commands were correctly classified in 40-50% of cases, while ‘up’ and ‘down’ commands were correctly classified in 90-100% of cases. The pipeline was reliably able to classify a ‘neutral’ state, in which no command movements were being executed. There was often a delay (of up to several seconds) between the classification of a command and the return to a ‘neutral’ state, during which seemingly random residual probability spikes were observed. This effect was most pronounced in the classification of ‘left’ and ‘right’ commands, but still occurred to a lesser degree in the classification of ‘up’ and ‘down’ commands.

Obtaining objective measurements of the accuracy of a given pipeline configuration proved to be a challenge for two reasons. First, there was no way to associate movements with correct commands in real-time. This was possible during training sessions, but only because the training module prompted the user to execute movements, and was thus aware of which command to broadcast as a marker string. Additionally, the timing of these broadcasts was important because the processing pipeline defined a window of relevant EEG activity relative to a received marker. In order to objectively measure post-training classification accuracy, marker strings corresponding to the correct classification of a given movement would have to be generated live, 0.5 seconds before the movement was executed. This is virtually impossible in a game environment, where command input is unpredictable.

Second, the pipeline frequently exhibited significant classification instability during and immediately following the execution of a command movement. For example, a jaw clench would almost always correctly result in a probability spike indicating a ‘down’ command. However, this spike would often be followed by residual probability spikes corresponding to other unintentional commands. An objective measure of classification accuracy would have to account for these residual spikes, either by ignoring them and only considering the initial spike or by averaging all probability spikes across a predefined window of time and considering only the highest average probability.

Discussion

Ultimately, classification accuracy was considered to be unsatisfactory. We believe this to be the result of several factors. First, there seemed to be significant importance to the types of movements that were chosen to represent commands. A hard blink and a jaw clench were chosen to represent ‘up’ and ‘down’ commands, respectively. A left-arm raise and a right arm raise were chosen to represent ‘left’ and ‘right’ commands, respectively. We believe that the disparity between the ‘up’/’down’ classification and the ‘left’/’right’ classification was due to the nature of these movements. For example, jaw clenches engage the temporalis muscle, which spans the sides of the skull. Engagement of this muscle caused increased pressure between the scalp and the EEG electrodes. Hard blinks similarly engage muscles that directly impact the manner in which electrodes contact the scalp. It is likely that the increased classification accuracy of ‘up’ and ‘down’ commands was due to the occurrence of easily-identifiable artifacts resulting from these scalp-electrode interactions.

Conversely, the poor classification accuracy of ‘left’/’right’ commands is likely due to small whole-body movements that occur as a result of moving either arm. These whole-body movements caused slight shifts in the position of the UM4 headset. Since the EEG electrodes were fixed to the headset and not to the scalp, these positional shifts also resulted in the movement of electrodes along the scalp. These movements would generate artifacts that, while easily identifiable, are indistinguishable between the left arm and the right arm. This theory is supported by the specific ways in which the pipeline misclassified ‘left’ and ‘right’ commands. Typically, these misclassifications fell into one of two categories each time the pipeline was trained. In the first category, the pipeline would generate a strong probability spike for either the ‘left’ or ‘right’ command repeatedly, regardless of which arm was moved. In these cases, probability spikes for the complementary direction were very rarely observed. In the second category, the pipeline would generate approximately equal probability spikes for both the ‘left’ and ‘right’ commands repeatedly, regardless of which arm was moved. It is important to note that generally speaking, neither category resulted in misclassification to either ‘up’ or ‘down’ commands.

Several experiments were completed using different movements to represent ‘left’ and ‘right’ commands. These included arm raise variations, leg movements, combinations of arm and leg movements, and fist clenches. All movements involving the arms and legs resulted in the same types of misclassification discussed above. Fist clenches were the least successful, with the pipeline being generally unable to distinguish them from the ‘neutral’ state.

Experiments were also completed using variations on the training module. The initial variant stepped through each command once and prompted the user to execute the associated movement multiple times for 20 seconds each. Due to exceptionally poor classification accuracy, this model was quickly abandoned. All future variants stepped through commands multiple times in random order and prompted the user to execute the associated movement once each time. The final variant prompted the user to complete each movement 20 times, for a total of 100 movements and took about eight minutes to complete.

In addition to the movement-related problems, it is likely that a data shortage also contributed to poor classification accuracy. The CBB only supports eight electrodes by default, while more professional EEG setups would likely use anywhere from 16 to 128 electrodes. We believe that the inclusion of an OpenBCI Daisy expansion module, which allows the CBB to support 16 electrodes, would yield better results.

Unfortunately, due to the associated costs, we were not able to fully address any of these problems during development. Addressing the data shortage problem would have required the purchase of a Daisy expansion module, which is only sold as part of a kit that also includes the CBB. Addressing the movement problems would have required the purchase of an alternate electrode mounting system to use in place of the UM4, such as an electrode cap. Both of these purchases were outside of our budget.

In lieu of purchasing new hardware, we included two software solutions in hopes of partially mitigating the problems we encountered. First, the processing pipeline was modified to output a moving average of command probabilities at a rate of 2 Hz rather than output raw probabilities at a rate of 250 Hz. This was intended to reduce the impact of short, random probability spikes while preserving larger, more sustained spikes. Second, the application backend included logic mandating that after a single command was sent to the game, a high ‘neutral’ probability had to be received before another command could be sent. This was intended to eliminate the impact of the residual probability spikes that were observed between command classification and return to a ‘neutral’ state. These solutions yielded only minor improvements, and the latter introduced a blocking problem in which no commands were transmitted for undesirably long periods of time while the user attempted to return to a ‘neutral’ state.

Future Work

Progressing beyond this point in the work requires additional resources. For public use, it is unrealistic to believe that users would be willing to undergo minutes of training before any software navigation can begin. This is necessary for our project because of the relatively low-cost hardware specifications. As mentioned, professional groups developing brain-computer interfaces utilize more electrodes and processing power than we have available. If this were to be factored into our design, its reactive speed and accuracy would be increased.

Future development also needs to be focused on free-form inputs in software; our project does not enable the user to enter meaningful text-based responses to prompts. This may look like an adaptive cursor that reacts to similar directional thoughts from the user.

References

[1] A. A. Shoka, M. M. Dessouky, A. S. El-Sherbeny, and A. El-Sayed, “Literature Review on EEG Preprocessing, Feature Extraction, and Classifications Techniques,” *Researchgate.net*, 08-Dec-2019. [Online].

[2] Al-Nuaimi, F. A., Al-Nuaimi, R. J., Al-Dhaheri, S. S., Ouhbi, S., & Belkacem, “Mind Drone Chasing Using EEG-based Brain-Computer Interface” in *2020 16th International Conference on Intelligent Environments*

[3] D. J. Krusienski, D. J. McFarland, and J. C. Principe, “BCI Signal Processing: Feature extraction,” Brain–Computer Interfaces Principles and Practice, pp. 124–145, 2012.

[4] Intheon, “NeuroPype Suite User Guide,” Neuropype by Intheon. [Online]. Available: https://www.neuropype.io/docs/. [Accessed: 24-Apr-2022].

[5] M. A. Javaid, “Brain Computer Interface .” [Online]. Available: https://nexusacademicpublishers.com/uploads/portals/Brain-Computer_Interface.pdf. [Accessed: 24-Apr-2022].

[6] Maskeliūnas, R., Damaševičius, R., Martisius, I., & Vasiljevas, M., “Consumer-grade EEG devices: are they usable for control tasks?” in PeerJ, 4, 2016

[7] Pan, P., Tan, G., & Wai, A.A,” Evaluation of Consumer-Grade EEG Headsets for BCI Drone Control”, 2017.

[8] Shankhdhar A, Mangla A, Singh AK, Srivastava A, “Operating of a Drone Using Human Intent Recognition and Characteristics of an EEG Signal”, in 2020 Sixth International Conference on Parallel, Distributed and Grid Computing, November 2020.

[9] T. Alotaiby, F. E. A. El-Samie, S. A. Alshebeili, and I. Ahmad, “A review of channel selection algorithms for EEG Signal Processing,” SpringerLink, 01-Aug-2015. [Online]. Available: https://link.springer.com/article/10.1186/s13634-015-0251-9. [Accessed: 24-Apr-2022].

[10] Villegas, I. D., Camargo, J. R., & Perdomo,” Recognition and Characteristics EEG Signals for Flight Control of a Drone” in IFAC PapersOnLine

[11] Y. Lu, H. Jiang, and W. Liu, “Classification of EEG Signal by STFT-CNN Framework: Identification of Right-/left-hand Motor Imagination in BCI Systems,” https://pos.sissa.it/, 22-Jul-2017. [Online]. Available: https://pos.sissa.it/299/001/pdf. [Accessed: 24-Apr-2022].