Electroencephalogram Controlled Rehabilitative Exoskeleton

A Technical Report submitted to the Department of Mechanical Engineering

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Taha Iqbal Shamsie

Spring, 2023

Technical Project Team Members

Jeyi Lee

William LaRow

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Dr. Sarah Sun, Department of Mechanical Engineering

Electroencephalogram Controlled Rehabilitative Exoskeleton

William LaRow
Department of Mechanical Engineering
University of Virginia
Charlottesville, USA
wal7rn@virginia.edu

Jeyi Lee
Department of Mechanical Engineering
University of Virginia
Charlottesville, USA
jl3usu@virginia.edu

Taha Iqbal Shamsie
Department of Mechanical Engineering
University of Virginia
Charlottesville, USA
tis5yz@virginia.edu

***Abstract*—This research project seeks to use convolutional neural network algorithms in order to generate control signals from electroencephalogram signals for an upper limb rehabilitative exoskeleton.**

Keywords—Electroencephalogram, Convolutional Neural Network, Support Vector Machine, Feature Extraction, Machine Learning, Rehabilitative Exoskeleton, Gaussian Kernel, Sobel Kernel

I. Introduction

A. Overview, Motivation, Background

For the scope of this assignment, we hope to develop a simple, EEG controlled, single degree of freedom forearm end-effector exoskeleton with three modes of motion to assist those suffering from neuromuscular and paralytic injuries and disorders. The use of an EEG is unique in that it provides neuro-based controls, allowing not only for a greater range of human control scenarios but also allows users to control systems with only their mind. The focus of control will be based on different movements within the human elbow. To begin, we will be focusing on a singular human arm, aiming to develop a system to control the actuation for holding and rotating the forearm left and right about the elbow with the help of EEG controlled pneumatic actuators. This form of control system is unique and exciting in terms of applications. Being that this form of rehabilitation is oriented towards patients struggling with neuromuscular injuries and paralysis, it is important to develop a system of control that is comfortable for the given individual. Not only this, but we hope to allow for real-time analysis and actuation as the data is being processed. Through the utilization of the EEG, it allows for us to achieve these novel goals that other actuation methods lack, allowing for real-time data analysis and processing for efficient, low-latency movement control. This use of the EEG will also allow for no muscular stress to be exerted by the patient in need. These controls from the EEG are to be determined through the use of a machine learning algorithm called Convolutional Neural Network (CNN), allowing for the classification and carrying-out of human interfaced movements.

B. Literature Review

As a part of the literature review, several published prior works were analyzed. One research paper that particularly stood out was “EEG-Based Exoskeleton for Rehabilitation Therapy" by Khan et al. The paper described various design choices when describing how the exoskeleton works, which we can take into consideration when determining our design. For example, the team arranged two PVC rods parallel to each other, one on each side of the forearm as elbow supports. The team claimed that such a feature will reduce weight and assist the rotation of the servo motors. In terms of data collection, the team had 109 participants perform various tasks, which yielded 14 EEG recordings by 64 channel EEG sensors. The procedure was when a target appeared on the left or right side of the screen, the participant had to move hand or feet or imagine moving hand or feet depending on the location of the target appearance. Given that we agreed to come up with a consistent set of instructions that all participants must follow during data collection, the paper provided some options to consider when creating our list of instructions. The team also explained the process for the EEG signal extraction. First step was windowing the signal per second knowing that the sample rate is 160 samples per second. Then, statistical parameters were used to extract features from each window, which were then stored in a 9 X 375 matrix. Lastly, the extracted features were classified into action and imagination with the help of LDA-based classifiers (Khan et al., 2021).

Another source we look at is a paper titled “Convolutional Neural Networks in Medical Image Understanding: a Survey” by Sarvamangala and Kulkarni. This paper outlines how imaging techniques are utilized in the medical field, particularly when identifying anomalies in a body. Throughout the paper, the authors mention several machine learning techniques: decision tree learning, clustering, support vector machines (SVMs), restricted Boltzmann machines (RBMs), and random forests (RFs). The authors argue that the aforementioned machine learning techniques are extremely difficult and unreliable in terms of image processing. Then, a convolutional neural network (CNN) model is offered as a solution. The authors state that since a CNN model is composed of filters that learn and extract useful features for efficient image processing, utilizing a CNN model is the best machine learning technique in medical image processing. To support this argument, the authors provide historical involvements of CNN in the medical field, one being the detection of COVID-19 through X-rays/CT scans. After that, the authors explain the fundamentals of medical image understanding. There are four important topics: classification, segmentation, image localization, and detection. Classification is the feature extraction and label assignment process. Segmentation is when an image is divided into regions with strong correlations. Localization is the prediction of an object in an image by drawing a box around and labeling the object. Detection is the process of classifying and localizing interested regions. In summary, this research paper was useful for answering why and how to utilize CNN in image processing.

The third source is titled “EEG-based Control for Upper and Lower limb Exoskeletons and Prostheses: A Systematic

Review" by Al-Quraishi et al. The biggest takeaway from this source is which EEG signal types are the most useful. According to the result of research, the authors argue that exogenous and endogenous EEG signals are the most appropriate signals to control an exoskeleton with. Exogenous EEG signals are generated by providing external stimuli through auditory and visual factors. Although using exogenous EEG signals requires light training, one effect of constant external stimulation is exhaustion. One example of exogenous EEG signal is SSVEP that can be observed by staring at a light flashing in a constant frequency. Another exogenous signal is P300 that appears for 0.3 second after a person notices an external stimuli. In contrast, endogenous EEG signals are controlled by a person. For this reason, endogenous EEG signals are more recommended as the source of exoskeleton control, but endogenous EEG signals require more training. The authors further emphasize the usefulness of endogenous EEG signals by providing a statistic that 57% of studied prior works utilized event-related desynchronization (ERD) and event-related synchronization (ERS), which are examples of endogenous EEG signals, to control exoskeletons.

The fourth source we referenced is called "EEG Classification via Convolutional Neural Network-Based Interictal Epileptiform Event Detection" by Thomas et al. The source mentions that the identification of epilepsy using interictal spikes, a frequent symptom experienced by epilepsy patients, is easily done by examining the EEG plot of a patient. In reality, the epilepsy identification is unnecessarily lengthy, especially because every neurologist has a different interpretation. For this reason, the authors claim that applying machine learning algorithms to an EEG plot will resolve the issue. The authors propose an EEG classification system composed of three components: pre-processing, waveform-level classification, and EEG-level classification. The most useful component is the waveform-level classification step, which utilizes convolutional neural networks (CNN). Knowing that the typical ratio between the interictal spike (epilepsy) and the background (normal) is around 1:1250, Tensorflow 1.2.1 with a K40 Tesla GPU on Ubuntu 16.04 is utilized to build a CNN system under the purpose of determining whether each waveform is a spike or background. This source is valuable because once we learn how to differentiate flexion and extension on an EEG plot, we can take the same approach to develop our CNN system.
Electroencephalogram controlled robotics is an unexplored field as most systems that exist now are used for medical screening for strokes and research on psychology and neuroscience. EEG is advanced for robot control as most current designs for wearable robotics rely on other sensors to determine motion such as inertial measurement units and electromyography, but these sensors are limited in that they require motion or some form of physical signal from the limb in order to move the exoskeleton. EEG based robotics, however, would not have this issue as it relies on signals from the brain rather than downstream signals from the muscular system, allowing for not only rehabilitation, but recovery of motor function when completely lost from conditions such as paralysis. Due to this, it is our objective to add further research to the field of electroencephalogram-controlled robotics as it is an undeveloped field which may have significant implications on human robotic interfaces.

University of Virginia Department of Mechanical Engineering

C. Goal and Contribution

There are two primary goals to this project. The first is to successfully generate appropriate control signals for electromechanical devices based on reading taken from the electroencephalogram using convolutional neural network (CNN) machine learning algorithms. These signals must be taken without external stimuli to correctly function for the desired system application. The second goal is to then apply these control signals to a wearable exoskeleton device. This goal attempts to address the limitations of using IMU and EMG sensors to control exoskeleton devices. Together these two goals form the overall objective to be achieved by this research project. Currently EEG controlled robotics is a field without wide literature coverage, and processing EEG signals with CNN algorithms is rarely performed in favor of more common methods involving feature extraction. Furthermore, differentiating signals taken without external stimuli is rarely performed. As such this project explores a novel method for processing EEG signals and contributes to literature on EEG signals taken without external stimuli.

II. Methods and Materials

A. Methods

In order to utilize the EEG as a form of efficient human interfaced control for robotic rehabilitation exoskeletons, we must first determine specific algorithmic methods that are to be employed to harness this control system. To determine what specific algorithms to use, we must first understand how the data from the EEG is recorded and determine what specific goals we hope to achieve from this form of control. To begin with, an EEG in conjunction with an OpenBCI board records data in a text file consisting of the recorded time elapsed and the recorded voltages across each electrode channel which can be formatted into a matrix. Since the data being collected is complex and is of high dimensionality, it is important to reduce the dimensionality of the generated data matrix in order to ensure that the data being utilized in the classification is meaningful. This reduced data can then be processed and reorganized into various formats through the use of simple transformations, allowing for a wide array of algorithms to be implemented.

To fully utilize various machine learning algorithms, we must differentiate the data we are using as a training set and the data that we collect and test for real-time classification. Using the csv files generated from collecting EEG data corresponding to known movements and reducing its dimensionality, we are able to develop a large training set to be passed into the given algorithm. These training sets consist of a large number of relatively small low-resolution matrices of known, movement labeled EEG data that were implemented and analyzed through the chosen algorithm, allowing for accurate classifications to be made. When developing a test set in real-time, we need to ensure that the program developed is able to create these small matrices of data as it is being recorded, allowing for low-latency classifications to be made.

Due to the fact that we hope to develop a control system that allows for low-latency efficient movements in real-time, it is imperative that the algorithm is able to process these data inputs fairly quickly while retaining its accuracy. Given the specifications necessary for the algorithm choice, we can

narrow down the classification algorithms to three: a Convolutional Neural Network (CNN), a Support Vector Machine (SVM), and a feature extraction algorithm. These three are under consideration due to their overall efficiencies and applicability on the given reduced data set. However, due to its superior processing time and ease of use, we have decided to implement a CNN algorithm to assist in classifying these EEG controlled movements.

To fully understand why we have chosen the CNN for the purpose of classification, we must first understand the architecture of a CNN algorithm for classification. Its architecture consists of four components: an "image" input layer, convolutional layers, max-pooling layers, and fully connected layers. The "image", or matrix, input of a CNN is the data set that will be passed into the algorithm where feature extraction is done automatically and classifications are able to be made. Within the convolutional layers of the CNN, the input matrix has filters, or kernels, applied to it which results in feature detection. With multiple convolutional layers, we are able to develop feature maps, or features that have specific strengths and weights. These convolutional layers are then interfaced with a max-pooling layer, where the features with the most prominent weights are retained, allowing for significant data to be returned. These prominent features filtered from the max-pooling layer are then fed into the fully connected layer where the classifications are made. Below is an illustration of how these layers are laid out in a classification CNN architecture.

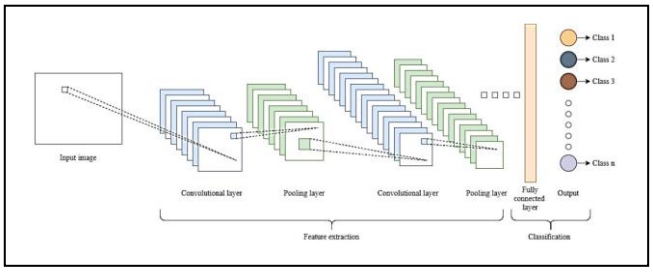

Fig. 1. Image of Simple CNN Algorithm Architecture Design [4]

Given this CNN architecture for classification, it is important to determine which kernels, or filters, should be employed within the convolutional layer, allowing for proper feature detection to become possible. There are two major kernels that are to be tested, the Sobel kernel and the Gaussian kernel. The Sobel kernel would allow for edge detection in the recorded EEG data, which could correspond to changes in the electric signal readings from the brain. This would allow for more accurate detection of features which would translate to a higher overall accuracy of the algorithm. The Gaussian kernel is a filter that is utilized for smoothing out the given EEG data. This would lead to less noise being present in the data, allowing for the EEG to better extract features from a given input. Although it is stated that we tested these two major kernels, it is important to note that we tested numerous other kernels as well through trial and error in order to determine the best filter to apply to the convolutional layer.

The equations for edge detection through Sobel kernel operations in a CNN are shown below:

$$G_x = g(x+1,y-1) + 2*g(x+1,y) + g(x+1,y+1) - g(x-1,y-1) + 2*g(x-1,y) + g(x-1,y+1) \quad (1)$$

$$G_y = g(x-1,y+1) + 2*g(x,y+1) + g(x+1,y+1) - g(x-1,y-1) + 2*g(x,y-1) + g(x+1,y-1) \quad (2)$$

Where G_x is the gradient in the horizontal direction, G_y is the gradient in the vertical direction, x is the x coordinate of the data point in the input matrix, y is the y coordinate of the data point in the input matrix, g is the input matrix, is the orientation of the edge in a given coordinate, and the magnitude of G represents the magnitude of the gradients.

The equation that dictates the Gaussian kernel is shown below:

$$\boldsymbol{G_{ND}}(\boldsymbol{x}; s) = \frac{1}{(\sigma\sqrt{2\pi})^N} e^{\frac{-|x|^2}{2\sigma^2}} \quad (3)$$

Where $\boldsymbol{G_{ND}}$ is the smoothened gaussian matrix, σ is the width of the Gaussian kernel, N is the Nth dimension of the kernel, and $\boldsymbol{x}$ is the input vector.

When comparing the CNN algorithm to the SVM algorithm, it is seen that the CNN is able to process data faster than the SVM after training. Another advantage that a CNN algorithm has over an SVM algorithm is that the CNN is able to automatically learn temporal and spatial pattern data whereas in an SVM, the user has to manually detect these features which can be time consuming. In comparison to the feature extraction algorithm, it can be seen that the CNN is able to automatically learn and extract features from data, whereas in a feature extraction algorithm, the user needs to manually select the features to be extracted. In general, manually selecting features can be time-consuming and requires prior knowledge on relevant features. Being that the data collected is complex, it is difficult to determine what prominent features should be extracted. This makes feature extraction incredibly difficult to develop in comparison to the CNN algorithm which is able to automatically detect and extract the prominent features. CNNs are also able to learn and identify complex patterns in data allowing it to make more accurate and nuanced classifications whereas feature extraction may not be able to fully capture the complex patterns in the EEG data. This ability to learn and identify complex patterns apparent in CNN algorithms is due to the fact that it is able to utilize numerous levels of abstraction whereas feature extraction is usually utilized for single levels of abstraction.

From these given specifications, needs, and comparisons, we have determined that utilizing a Convolutional Neural Network (CNN) algorithm would be the best algorithm to employ for EEG data collection and classification. We expect our algorithm to be able to successfully differentiate control signals at an accuracy above 33%. Most simple CNN algorithms for specific tasks are able to achieve an accuracy around 75%, with more rigorous models being able to reach accuracies above 95%, however due to concerns over the low resolution of data, concerns that the signals taken without

external stimulus will be too weak to record, and the large amount of noise associated with EEG data all lead to an expectation of accuracy much lower than the standard accuracies for CNN algorithms.

When implementing real time collection for the algorithm, due to a difference with the computer processing rate and data collection rate, several data packets were lost. To resolve this, the algorithm had to be modified to take into account the lost data packets over the one second recordings. With these revisions, the model is expected to be able to process the data without any issues with regards to latency.

In order to collect large data sets for training the algorithm on patterns relevant to the desired function we have determined a procedure for data collection in which 7 sets of 3 minutes of data are taken. The 16 channel EEG collects data at a rate of 125 Hz, creating 157,500 data points per session. The subject is asked to imagine continuous arm motion in the order of "left", "hold", "right", alternating between the three every 20 seconds. An interval of 2 minutes is given between each successive set, and data points at the changes are filtered from the set. This process is then repeated over various sessions and individuals to create a comprehensive collection of training data.

B. Current Algorithm Implementation and Architecture

In order to develop our algorithm properly, it was imperative that we carefully determined which preprocessing filters and Convolutional Neural Network (CNN) layers to employ. Through these layers and filters, we hoped to achieve proper data processing and analysis within the convolutional layers, allowing for proper patterns to be detected in the data. This would in turn lead to more accurate classifications for each tested motion.

In terms of preprocessing filters on the initial data set, we implemented a band-pass filter to filter out noise between the frequencies of 1 Hz and 55 Hz and a notch filter in order to filter out data points at a specific frequency. These filters were utilized for the purpose of reformatting the data points within the original EEG recordings in order to form a more comprehensive data set that prevents the CNN algorithm from incorporating noise as important data. Once these filters were applied to the initial data set, we then broke the EEG data into 125x16 matrices or "images", appending them into an array and applying the Gaussian kernel and Sobel kernel to each individual image. The Gaussian kernel, in this case, allows for each specific image to become normalized, allowing for the smoothening of noise within each matrix. As for the Sobel kernel, it was implemented so that specific edges within the images were detected, allowing the CNN to more easily determine relationships within the complex data. It is to be noted that the algorithm is still able to determine classifications on a dynamically sized matrix, meaning any image input would be able to receive a classification.

When determining the CNN layers, it was imperative that each layer and activation method were carefully considered and placed in order to more accurately classify the image data being processed. In order to do this, we employed the use of numerous convolutional 2D layers utilizing the rectified linear unit activation method, batch normalization layers, max pooling layers, dropout layers, flatten layers, and dense layers. These layers were all compiled into a sequential CNN model, meaning that each layer was connected and processed in a linear fashion. Once compiled, the model would then save the given weights being applied during the convolutional layers, allowing for the function weights to be called and applied on the real-time data collection algorithm. This in turn allows for real time classifications to be determined. Due to the fact that overfitting was a major concern for accuracy retention, we decided to implement dropout layers and normalization layers after various convolutional layers in order to prevent the model from depending too much on a single data set. This effectively prevented our model from allowing overfitting to majorly affect the output weights and accuracy. Although we decided to utilize dropout layers and batch normalization layers to counteract this issue, we have also been experimenting with regularization methods and plan to implement them into the model in the near future.

Once this model was confirmed to be successful, a real time data collection program was developed in order to allow for classifications. This program takes in a data stream from the Brainflow API which allows for real time, unfiltered data to be inputted into the saved model. This data was then passed through the preprocessing filters that were applied to the original CNN model, allowing for classifications to be made. These classifications were then passed through the Pyserial package to communicate with an Arduino which controlled the actuation of the end-effector. With the Arduino communication coming through the real time data collection code, it allowed for low latency classifications to be made, allowing for accurate actuation to become possible.

Given that the Raspberry Pi OS, Raspbian, was too out of date to implement the original Tensorflow model, a lot of changes were required to take place to allow for actuation to occur through the Raspberry Pi. In order to do this, we needed to install given wheel files to allow for Tensorflow to run on the device. This in tandem with the use of a virtual python environment using python version 3.7.12, we were then able to install a previous version of Tensorflow 2.5.0. Given that this version was older than the one the current model was trained on, changes in the structure of the algorithm needed to be made. In this version of Tensorflow, specific activation functions that were being utilized under the Keras package were not defined, meaning that new model structures needed to be developed. Once restructuring the CNN model with the same features as stated previously, a working model was developed.

C. Materials

For this project, we initially intended to use a flexion-based exoskeleton design focused on the flexion and extension around the elbow joint. However, to collect more distinct EEG data, we have moved from the initial design to explore an end-effector model where the wrist is moved by linear actuators. For the linear actuator we used to use a pneumatic linear double cylinder actuator. There are three modes of motion possible: move left, move right, and hold / idle. The movements occur by attaching a person's wrist to the end of the shaft of the pneumatic linear actuator. To promote safety, a 3D-printed end effector that serves as the link between the wrist and the linear pneumatic actuator was connected. The 3D-printed end effector is largely broken into two components: elbow and arm. The elbow component of the end effector is shown in Figure 2. The elbow component reflects a few design choices. First, one end has a spherical

edge for the elbow to sit on. Second, a flat plate is attached to the bottom, and rollers are attached underneath the plate to serve as wheels. Third, one hole is added at the bottom, so the linear actuator shaft can attach to the end effector and be fixed using nuts. Fourth, the end effector has holes to minimize the amount of 3D printing material usage. A layer of Styrofoam was placed above the elbow component to cover the holes and enhance comfort.

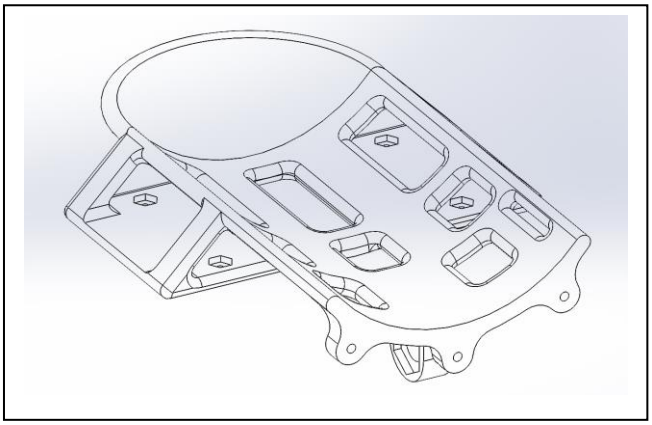

Fig. 2. Elbow Segment of the End Effector (Created by Sarah Hemler, 2023)

Figure 3 displays the arm component of the end effector. Similar to the elbow component, the arm component has a flat plate to attach the rollers as wheels and holes to save 3D printing materials and add a Velcro strap to fix the wrist. The elbow component will attach to the arm component using screw at three locations.

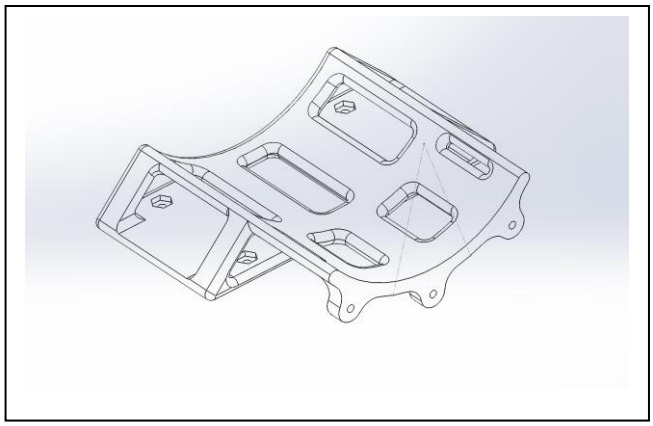

Fig. 3. Arm Segment of the End Effector (Created by Sarah Hemler, 2023)

Combined as an assembly the end effector has a length of about 16 inches, which is the approximate distance between elbow and wrist. Once the end effector is attached to the pneumatic linear actuator, the system has a 30-centimeter range of motion left and right based on the length of the pneumatic linear actuator shaft length. When the pneumatic linear actuator moves, the entire arm, strapped onto the 3D-printed end effector from elbow to wrist, linearly moves left and right as opposed to wrist rotating with the elbow as the axis of rotation.

Figure 4 below shows the engineering drawing of the end effector assembly. Some key dimensions are labeled. The assembly is about 330 millimeters long, 154 millimeters wide, and 54 millimeters tall. Additionally, there is an extra hole at the bottom that connects to the pneumatic motor shaft, which will allow the end effector and user arm to follow along with the pneumatic motor shaft motion. The top surface of the end effector is curved at a radius of 63.5 millimeters, so the user arm sits comfortably on top. Some other design choices are carving out multiple holes on the end effector to save 3D-printing cost and time. Then, a layer of Styrofoam was added on top to cover the holes and enhance comfort.

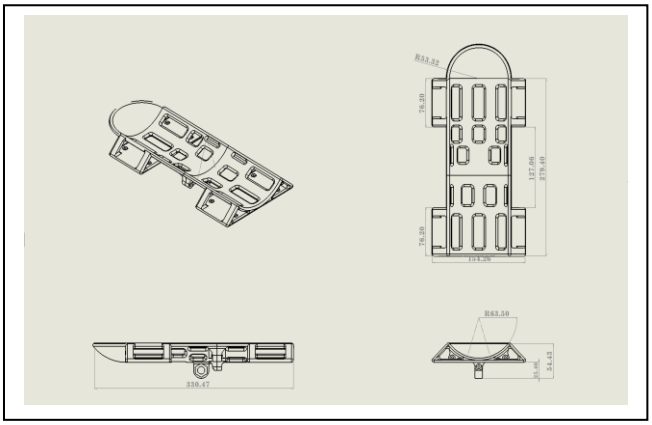

Fig. 4. Engineering Drawing of the End Effector Assembly (Created by Sarah Hemler, 2023)

Another key component of our design is the circuit design, which is shown below as Figure 5. The circuit is composed of the following equipment: wires, 220 Ohm resistor, TIP 120 transistor, Arduino Nano, breadboard, solenoid, voltage supply, and motor. The circuit design will allow us to control the pneumatic motor using an Arduino input by controlling a solenoid connected to the pneumatic actuation, which will change the pressure distribution in the pneumatic tube causing the shaft to either extend or retract.

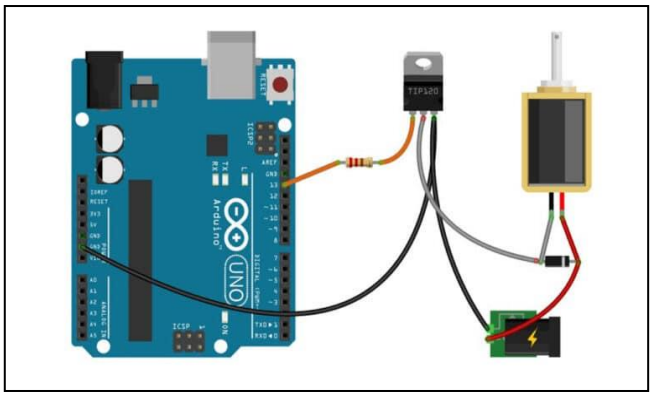

Fig. 5. Circuit Component of the Design [2]

In Figure 6, a photo of all the mechanical components connected together can be seen. The Raspberry Pi Canakit shown on the top left corner allows us to control the motor remotely. The mechanical components are shown at the bottom, and the circuit components are shown at the top half of the image.

Fig. 6. Photo of the All Components Combined (Created by Shamsie, 2023)

To collect data, we used an OpenBCI EEG headset connected with a Cyton+Daisy board as well as OpenBCI software to record data and stream to our CNN algorithm. We decided to host all of the scripts and code for this project on a Raspberry Pi CannaKit and use relevant OpenBCI and python packages and libraries to run the CNN algorithm in real time.

D. Evauation Plan

In order to test our prototype, we first determined acceptable thresholds for the accuracy and latency allowed in our model. The base threshold for accuracy of this design is 33%, at which any differentiation would be purely random chance. Ideally, we sought to achieve an accuracy threshold greater than 90%. Through our data collection process, we have gathered various data sets which we then train in an 80:20 split, where 80% of the data is used as a truth set for training the data, while the remaining 20% is used to test the data and determine the accuracy of the algorithm. To prevent over-constricting, a process by which the data becomes very well trained for a specific data set but fails outside of that specific data set, the algorithm must be tested against other data sets to determine true accuracy. This process continued until the model was able to determine the truth data above the accuracy threshold. With accuracy above the threshold, the EEG was then used to collect raw data, where at certain times the subject was told to think of one of the three control signals, which was then run through the algorithm. If the accuracy fell below the threshold then the algorithm was modified and larger training sets were taken. After the accuracy was maintained above the threshold for raw data, the algorithm was then adjusted to take real time data, with the same means of testing as before through the EEG. With this, the model latency was determined by taking the time difference between the expected signal and the model output. If the latency was outside of the acceptable range, the data inputs of the algorithm were changed and the model modified for faster computation. The entire process was then repeated until the latency was within the acceptable range. Once it was, the model was then loaded onto a raspberry pi and connected to the end-effector system. Live tests were then performed with the entire system, measuring the latency and accuracy. These tests were performed on the three modes of motion, left, right, and hold, where the user thought between the control signals and the accuracy and latency of the signal response were recorded.

III. Experiments and Results

A. Experiments

When discussing which aspects of the initial algorithm development went well, it can be seen that the initial idea of utilizing a CNN for motion classification works wonderfully. However, given specific layers and processes, we noticed numerous aspects that could be improved. For example, initially, when developing the CNN, we only utilized convolutional 2D layers, max pooling layers, and dense layers. This led to high accuracy outputs on the test sets, but abysmally low accuracies on recorded test sets. This is where we discovered the issue of overfitting and began accounting for this issue through the use of normalization layers and dropout layers. As a result of these changes, we found the accuracy to be more consistent between both the testing and training sets, but it still remains an issue in the current algorithm design. As we move forward, we will attempt to mitigate this issue further through the use of regularization methods and early stopping methods.

Aside from changing the general architecture to prevent overfitting, we have also been experimenting with different activation methods within the convolutional layers. These activation methods are the main determining factors for weights and function application within the CNN. To begin with, we opted to use rectified linear units (ReLU) due to their high accuracy and ease of implementation. This allowed for quick accuracy convergence and a wide array of accuracies. Due to these reasons, this activation method is still in use for our current algorithm. However, we have been experimenting with other activation methods and have discovered that leaky rectified linear units (Leaky ReLU) worked just as well, if not better than normal ReLU functions. Leaky ReLU allows for faster convergence and more complex pattern recognition than ReLU but is relatively limited in the data sets that it can be of use in. Due to the lack of full-testing that has been allocated to Leaky ReLU, we have since decided to use it in the current design as it allows for greater accuracy. Two other activation units that should be looked into further would be through the employment of the Tanh and Sigmoid activation functions. Both allow for quick convergences and generally high accuracies but would require a restructuring of the current algorithm.

Regularizers were also utilized in order to prevent overfitting while training. These in tandem with a model learning rate allowed for the data to converge to an accuracy that was adjusted for overfitting and underfitting.

Although we currently have a final design for the given CNN algorithm, we continue to investigate new models to hopefully optimize our classifications.

B. Results

The results for training our algorithm have two aspects: results for the data taken from one of the researchers, for which our training sets are relatively extensive, and results for data taken from other individuals. When training the data for the single patient only set, accuracy for the algorithm in interpreting the desired motions reached 77%. When run on

test sets in which the motion patterns were randomized, the accuracy decreased to 60%, which can be attributed to overfitting of the model in the training sets; however, this accuracy is still significant as 33% represents fully random classification and is the base threshold for the algorithm. The results for the general set were consistently lower than that of the single patient only set, with the accuracy of the training set for general subjects coming out to be just around 45% while the test set accuracy was similarly low with proper classification occurring at roughly 40% accuracy. The decrease can similarly be attributed to overfitting of the data as with the single patient only set, while the overall decrease in accuracy between the general and single patient only sets can be attributed to a lack of data from other subjects and also a more complex generalization needed to be made by the CNN algorithm as different subjects do not necessarily have consistency between the brainwaves representing the different motion patterns. The accuracy of the general set though is still above the base threshold however still requires more modification for the algorithm to be applicable to general patient sets or must simply be trained for a single user at a time.

When analyzing the results from applying the real time classification algorithm, it can be seen that the accuracy of the classifications can be fairly sporadic. The accuracies obtained for the real time data collection were significant for the "hold" movement with accuracies consistently around 90% while distinguishing between right and left signals achieved an accuracy of only around 50%. The algorithm was often able to distinguish "left" more accurately than "right". This can be attributed to the fact that a change in environment can affect the accuracy of the EEG's real time data collection, meaning that it may not fit the model as well as if it would have been in a controlled setting. This relative decrease in accuracy may also be attributed to problems with modifications made to address an issue of lost data packets which will be discussed in greater detail in the next section. When implemented with the mechanical system, the average accuracy of signal determination dropped further to an average of around 43%. Furthermore, due to the mechanical system, we were unable to implement a proper "hold" function due to problems related to the pneumatic design. The system also faced several issues with regards to significant latency in the actuation of the motion, though this problem lay more with the mechanical design and will be discussed in greater depth below.

We were also able to successfully develop both a real time classification program and a CNN model within the Raspberry Pi OS. This allowed us to make real time actuation given the individuals thoughts. One note about the Raspberry Pi, however, is that it is incredibly slow in training its models. This leaves longer times to train and develop a working model. Another aspect that was of note, was that the real time classifications had a small amount of latency when communicating with the Arduino Nano. This could be due to the fact that the real time collection program required the usage of numerous sleeping threads to enable the Arduino to process the given classifications into motions.

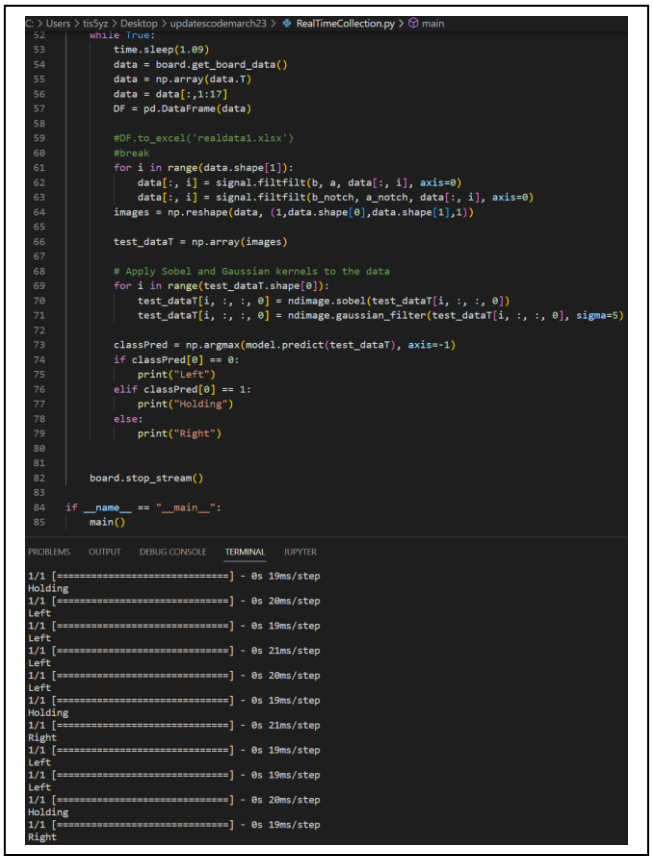

Fig. 7. Real Time Classification Example

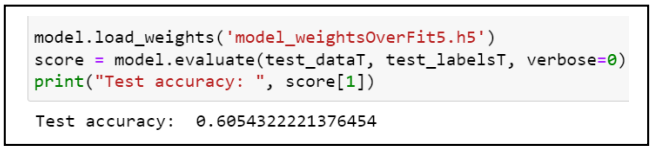

Fig. 8. Example of Accuracy Reading Using the Same Model Weights on the Single Patient Only Test Set

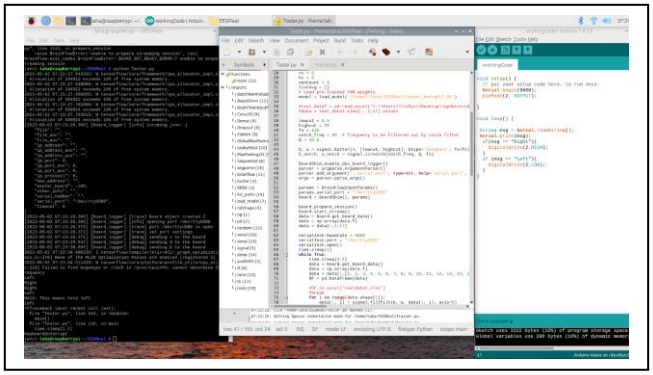

Fig. 9. Real Time Classification on Raspberry Pi

IV. Discussion and Conclusion

As mentioned briefly in the Goal and Contribution section, as this system is designed for patients suffering neurodegenerative disorders and paralysis, it is important that all signals are able to be generated based on thought alone without external stimuli as this system wouldn't be applicable if such stimuli were required. Due to this, there was no such

stimulus present when collecting training data, and this lack of external stimuli, or artifacts, may have contributed to difficulty for the model to differentiate control signals as signals without external stimuli are far more difficult to differentiate then those with external stimuli. This was a large concern in our initial design and may have contributed to a lack of accuracy in our initial model training. Another major concern for this design is the lack of feature extraction. The goal of this research is, in part, to use a CNN algorithm to determine the control signals so that feature extraction would not be a necessary process. However, the model may not have been able to determine the relevant data from the EEG channels as it may be too low in resolution without the feature extraction. This is especially a concern for noise. Without feature extraction, noise filtering becomes difficult. To account for this, large data sets including noise data must be taken, but the data sets we took may not have been large enough to account for this. As the system must operate in real-time, the computation must be fast with low latency. This is why a neural network was chosen; however, attempts to reduce latency inevitably reduced system accuracy and increasing accuracy similarly increased computation time and latency. As such, a tradeoff exists between the accuracy and latency which became a difficult problem to work around. The drop in accuracy from the training sets to the real time classification is attributed to data packet loss, which is a significant problem we ran into when transitioning to real time data collection. The Cyton+Daisy Board we used for collecting the electroencephalogram signals had a much higher sampling rate than the computer refresh rate, and due to this several data packets were lost in the collection of data. The lost packets were randomly distributed, and due to this not only did the size of the classification matrix become reduced, the data was no longer linear in time with random gaps. Because of this we were forced to modify the algorithm to account for this and sothe model accuracy was reduced greatly. When connected to the mechanical system, noise from the pump was a very large problem and contributed significantly to noise in the data which led to a much lower classification rate. The pneumatic design also had very high latency which greatly affected the processing of signals contributing to loss of accuracy. The use of pneumatics also restricted our initial design as the actuator could not stop and hold in the middle of motion, which led to the loss of an entire classification signal. Because of these, the pneumatic design was the most significant contribution to the loss of accuracy when connected to the mechanical design.

This project seeks to use electroencephalogram signals coupled with a convolutional neural network in order to generate real-time control signals for a rehabilitative upper limb robotic exoskeleton system without external stimuli. The convolutional neural network will undergo a continuous cycle of revision until both accuracy and latency of the model are within acceptable ranges for the described system. As our project stands, we have a CNN algorithm that is able to differentiate signals for a single subject at relatively high accuracy, however still requires further modification. The algorithm was modified to accept dynamic data with tolerance for dropped data to improve the CNN model such that we can stream real time data through the model. Attempts may be made to explore alternative CNN model types such as Residual Network (ResNet) and Visual Geometry Group (VGG). We similarly would like to make the algorithm more robust for general patient sets rather than only single patient sets, and as such this project still requires much more data collection. As this system must eventually be connected to the external world, we intend to create a simulation to visualize the motion predicted by the algorithm, so that we may collect data while the subject is receiving feedback from the algorithm. For the focus of this current research, we must still create the mechanical system to complete the full system, and to do this a better model would include the use of motors rather than pneumatics to increase the accuracy and decrease the latency of real time processing. Though not within the scope of this project, other modifications and future work could be performed. For this project, we only focus on a single degree of freedom with three modes of motion, however this could be extended for multiple degrees of freedom by collecting larger training sets with additional control signal outputs for the additional degrees of freedom. This would allow the mechanical design to include a more full range of body control. This project also focuses only on using a convolutional neural network, so in the future other algorithms may be considered for use instead, such as vector machine, random forest, and feature extraction algorithms, which may increase overall system performance. Another aspect to highlight for future improvements is the developing and executing of averaging methods on the real time data collection algorithm. By investigating these features and methods, it will allow for the algorithm to more accurately determine what movement is being implied. The scope of this project could also be changed to create an automated day-by-day training method for the CNN. This would entail having a patient record data on the EEG for training and then carry out their physical therapy with the model trained at the specific point in time. This can then be supported through a reinforcement learning algorithm, allowing for an eventual model to be developed through months of use and training. Changing the scope of this project in this direction would eliminate the issue of a patient's mood or state of mind affecting the model accuracy.

REFERENCES

[1] AL-Quraishi, M., Elamvazuthi, I., Daud, S., Parasuraman, S., & Borboni, A. (2018). EEG-based control for upper and lower limb exoskeletons and prostheses: A systematic review. Sensors, 18(10), 3342.

[2] Benne. (2022, September 6). Controlling a solenoid valve with Arduino: A complete guide. Makerguides.com. Retrieved May 7, 2023, from https://www.makerguides.com/control-a-solenoid-with-arduino/

[3] Khan, B. A., Usmani, A. R., Athar, S., Hashmi, A., Farooq, O., & Muzammil, M. (2021). EEG-based exoskeleton for rehabilitation therapy. Design Science and Innovation, 645–653.

[4] Kumar, A. (2021, September 15). *CNN Basic Architecture for Classification & Segmentation*. Data Analytics. Retrieved December 13, 2022, from https://vitalflux.com/cnn-basic-architecture-for-classification-segmentation/

[5] Sarvamangala, D. R., & Kulkarni, R. V. (2021). Convolutional neural networks in medical image understanding: A survey. Evolutionary Intelligence, 15(1), 1–22.

[6] Thomas, J., Comoretto, L., Jin, J., Dauwels, J., Cash, S. S., & Westover, M. B. (2018). EEG classification via Convolutional Neural Network-based interictal epileptiform event detection. 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC).