

Voice Control of the HOPE Hand Exoskeleton for Individuals with Motor Impairments and Aphasic Speech



A Major Qualifying Project
Submitted to the Faculty of
WORCESTER POLYTECHNIC INSTITUTE
in partial fulfillment of the requirements for the
Degree of Bachelor of Science

By
Connor Gaudette
Matthew McGourty
Allison Rozear
Keenan Segenchuk

Project Advisors:
Tess Meier (Robotics Engineering)
Prof. Christopher Nycz (Robotics Engineering)
Prof. Erin Solovey (Computer Science)
Prof. Yunus Telliell (Humanities and Arts)
Prof. Haichong Zhang (Biomedical Engineering)

Date: April 23, 2024

This report represents the work of one or more WPI undergraduate students submitted to the faculty as evidence of completion of a degree requirement. WPI routinely publishes these reports on the web without editorial or peer review. For more information about the projects program at WPI, see <http://www.wpi.edu/Academics/Projects>.

Abstract

The HOPE Hand exoskeleton is a motor aid for individuals with hypertonicity and spasticity. This project develops a voice control system for the HOPE Hand which is trained for Broca's aphasia, a communication disorder which often impacts this population. A database was created using vocalizations of typical and aphasic speech collected from participants in conjunction with open source datasets. AI models were modified and trained on these data to identify users and recognize commands, then integrated with the HOPE Hand.

Contents

Figures	5
Tables	6
1 Introduction	7
1.1 Goal Statement and Objectives	7
1.2 Scope	8
2 Background	9
2.1 Traumatic Brain Injury, Stroke, Cerebral Palsy	9
2.2 Hand Impairments and Rehabilitation	9
2.3 Speech Aphasia and Rehabilitation	11
2.4 Voice Control	12
2.5 Machine Learning and Voice Detection	13
3 Ethically Aligned Design Considerations	14
3.1 Assessing Broader Impacts	14
3.2 Broader Impacts of Machine Learning and Data Collection	15
3.3 Interviewing and Data Collection	16
4 Evaluation	17
4.1 User Performance	17
4.2 Metrics for Objective Algorithm Assessment	17
5 Methodology	19
5.1 User Experience Model	19
5.2 System Design Requirements	20
5.3 Database Curation	20
5.3.1 Google Mini Speech Commands	22
5.3.2 Vocalizations Collected from The Team	22
5.3.3 Mozilla Common Voice Dataset	23
5.3.4 AphasiaBank Data	23
5.3.5 Study Data	24
5.3.6 Augmented Data	25
5.4 Voice Control Pipeline Approach	26
5.4.1 Recording	26
5.4.2 Command Detection	28
5.4.3 Speaker Verification	29
5.4.4 Command Recognition	30
5.4.4.1 Overview of Python Code	30
5.4.4.2 Integration of the MATLAB Engine	31
5.4.4.3 File Conditioning	32

5.4.4.4 Creation of the Command Recognition Model	32
5.4.4.5 Command Identification	35
5.4.4.6 Creation of Configuration File	36
5.4.5 Updated Pipeline	36
5.5 Connecting to the HOPE Hand	37
5.5.1 State Machine For Identifying Open and Close	37
5.5.2 GUI Communication	40
5.5.2.1 File Reading/Writing	41
5.5.2.2 IP Address Communication	42
6 Results	44
6.1 Database Curation	44
6.2 Second Phase of Interview	46
6.3 Pipeline Metrics	47
6.3.1 Command Detection Accuracy	47
6.3.2 Speaker Verification Accuracy	48
6.3.3 Command Recognition Accuracy	48
6.3.4 Updated Command Recognition	49
6.4 Connecting the Voice Control System to the HOPE Hand	49
7 Discussion	49
7.1 Pre-Processing	49
7.2 Command Recognition	50
7.3 Limitations	50
7.4 Future Work	51
7.4.1 Increase in data	51
7.4.2 Practical testing	51
7.4.3 Activation Keyword	52
7.4.4 MRI Compatibility	52
8 Conclusion	53
Acknowledgements	54
References	55
Appendix A: Voice Recognition Description of Files, Libraries, and Resources	61

Figures

Figure 1: Index finger range of motion with and without exoskeleton assistance	10
Figure 2: Description of Health Condition (Galletta & Barrett, 2014).	12
Figure 3: Formulas for recall, specificity, and accuracy...	17
Figure 4: State diagram of the overall system from the perspective of the user...	20
Figure 5: Hierarchy of Database	21
Figure 6: Hierarchy of Database from Model Trainer Perspective	21
Figure 7: Code to Process AphasiaBank Videos	24
Figure 8: Images used in IRB Study	25
Figure 9: Recording Timeline	27
Figure 10: Code for Recorder Proof of Concept	28
Figure 11: Recorded Audio Saved	28
Figure 12: MATLAB Engine Importation	31
Figure 13: MATLAB Code Showing checkUserCommand Function	31
Figure 14: Example of Use of eng.checkUserCommand	32
Figure 15: Keras audio_dataset_from_directory function	33
Figure 16: Converting waveform to spectrogram Tensorflow	33
Figure 17: Mapping onto dataset	33
Figure 18: Defining the Architecture	34
Figure 19: Training Model	34
Figure 20A: Saving Model	
Figure 20B: Saved Model	35
Figure 21: Confidence Value Example and Unorganized Output of Command Recognition	36
Figure 22: Updated Pipeline Flowchart	37
Figure 23: State Machine Organization Flowchart	38
Figure 24: Code Description of readopen()	39
Figure 25: Code Description of readclose()	39
Figure 26: readopen() and readclose() Called in Main	40
Figure 27: Image of GUI	40
Figure 28: Creating .txt File Based on Command	41
Figure 29: GUI Code to Extend or Flex HOPE Hand	42
Figure 30: Socket Communication from Python	43
Figure 31: Socket Communication from MATLAB (MATLAB)	43
Figure 32: Example speech data used for command recognition training.	46

Tables

Table 1: The results of a box and blocks test...	11
Table 2: Metrics and Goals for Voice Control Pipeline Elements	18
Table 3: Command Recognition Pipeline Steps	30
Table 4: Reported Metrics for Voice Control Pipeline	47

1 Introduction

Hand functionality plays a significant role in many aspects of life, and many individuals take this ability for granted. The practical ability to easily utilize both hands allows people to perform essential tasks like lifting items, dressing themselves, performing at work, engaging in social interactions, opening doors, and more. Without this functionality, it becomes more difficult for one to accomplish these common day-to-day activities. Diseases and neurological traumas such as traumatic brain injury (TBI), stroke, or cerebral palsy can result in upper limb impairments that impede hand motion in this way (Parker, 1986; Douglas, 1965). These impairments include hypertonicity, or overactive muscle contraction, spasticity, and weakness (Teasell, 2003). To assist these individuals in daily tasks, rehabilitation robotics is being used more to aid in hand movements, and has shown promise in leading to functional gains (Schabowsky et al., 2010).

Individuals with moderate to severe spasticity have shown to struggle to open their fingers using existing rigid commercial exoskeletons (Peters et al., 2017). Many of these existing exoskeletons are not specifically designed for spasticity and do not fully meet the needs of these individuals. In contrast, the HOPE Hand is specifically designed to assist in individuated finger motion and extension with high resting muscle tone or spasticity present, with the goal of increasing independence to these individuals. It was found that controlling the HOPE Hand through a button or electromyography can pose challenges for users and even counter therapeutic methods in some cases (Meier et al., 2019). A voice control system is a promising solution to fix these problems and help users successfully control the HOPE Hand to their needs.

Oftentimes, the TBI or stroke which causes hand hypertonicity and spasticity also causes a communication disorder known as aphasia. Due to differences in motor function, processing, or speech production, people affected with aphasia use different speech patterns than unaffected people. As a result of this, existing voice control systems designed for those with healthy speech may often not work well for these individuals. Therefore, the designed voice controlled system must accommodate these vocalization differences to work for the intended user population.

1.1 Goal Statement and Objectives

Our goal was to develop a usable voice control system for the HOPE Hand that accommodates the speech patterns of the intended users. Therefore, this system was designed to be compatible with individuals that have speech aphasia in addition to the aforementioned motor impairment of the hand. In order to accomplish this goal, our team set out the following objectives to accomplish.

1. Develop an algorithm able to detect the commands “open” and “close” from a speaker with aphasia with an accuracy of $90\% \mp 10\%$.
 - 1.1. Apply algorithm to detect commands and speaker with aphasic vocalization data
 - 1.2. Train and implement algorithm to recognize commands with aphasic and typical vocalization data

- 1.3. Implement speech distortion algorithms to make model more robust
2. Connect speech model (Objective 1) to HOPE Hand exoskeleton
 - 2.1. Convert microphone feed to a compatible speech signal input for model
 - 2.2. Translate algorithm output to signal the HOPE Hand to enter the desired state
3. Curate database of aphasic speech and healthy speech to train speaker and speech recognition models (Objective 1)
 - 3.1. Compile database from AphasiaBank, Mozilla Common Voice, Google dataset, and other open-sources
 - 3.2. Collect vocalization data of aphasic speech from participants with aphasia
4. Create user manual/description for future teams to work with the control system in its existing state
 - 4.1. Document workflow and design choices in a legacy document
 - 4.2. Create manual for potential users of the HOPE hand and the voice control system
 - 4.3. Uploading all of the code to a repository (GitHub) (include all software packets and system requirement)

1.2 Scope

This project was intended to help bridge the gap from the design and creation of the HOPE Hand exoskeleton and the use of the HOPE Hand exoskeleton in assisting activities of daily life. The voice control system described in this paper does not meet the requirements for use in activities of daily life but lays the foundation on which another project may improve upon to bring the voice control system to the level that is reasonable for use in activities of daily life. The project described here is the first prototype of the voice control system that is designed to be used with the HOPE Hand exoskeleton. It is the intention of the authors that with the descriptions provided below another team may be able to develop a voice control system that is ready to assist in activities of daily life.

2 Background

The goal of the HOPE Hand is to help individuals with hypertonicity and spasticity with activities of daily living. The cause of many hand impairments is traumatic brain injury, such as stroke. Rehabilitation of such hand impairments and hand exoskeletons are closely related by their goal to assist individuals with hand impairments in everyday activities. The voice control system was intended to help make this goal a reality. The voice control system described in this paper uses several machine learning algorithms to detect and recognize the commands issues. There are many forms of voice control in literature that were considered when designing the voice control system. However, many of these voice control systems were not designed for individuals with speech aphasia. This section provides relevant information on traumatic brain injuries, hand impairments and rehabilitation as it relates to hand impairments, voice control, speech aphasia and rehabilitation as it relates to speech aphasia, and machine learning and voice detection

2.1 Traumatic Brain Injury, Stroke, Cerebral Palsy

The HOPE Hand is designed for individuals who have had a traumatic brain injury, such as stroke. Over 795,000 people in the United States suffer from a stroke every year (Center for Disease Control and Prevention [CDC], 2023). The leading causes of traumatic brain injuries are falls, firearm related injuries, motor vehicle crashes, and assault, while the leading causes of stroke are high blood pressure, high cholesterol, smoking, obesity, and diabetes (CDC, 2023). Both strokes and traumatic brain injuries can often cause those affected to have serious and long-term disabilities, and it is estimated that over 5 million Americans currently live with long-term disabilities as a result of a brain injury (Coons, 2023), with a subset having hand impairments and speech aphasia.

2.2 Hand Impairments and Rehabilitation

There are several traditional methods for rehabilitation for upper limb impairments. Physical therapy, surgery, routine injections, or a combination of these methods have been used as rehabilitation strategies for disorders of the arms, wrists, and hands (du Plessis et al., 2021; Copley, 2014; Kamper, 2022). The treatment plan for these types of impairments are outlined in models such as the “Hypertonicity Intervention Planning Model” that take into account one's symptoms and clinical test results to match treatment options designed for the most effective outcome (Copley, 2014). For upper limb hypertonicity specifically, treatment can include stretching, splinting, strengthening of antagonist muscles, oral medications and focal injections (Marciniak, 2011). For muscle spasticity, common treatments include oral medications and injections, and specific stretches and physical therapy exercises designed to relax the hand muscles (“Spasticity Treatment”, 2013).

Unfortunately, current plans are not always able to return individuals to typical functionality (Kamper, 2022). Combined with the prolonged duration of these treatment plans, associated costs, and the limited effectiveness, many may lose interest in their recovery plans (du Plessis, 2021). Most medications and physical therapy leave affected individuals in need of assistance to complete many activities of daily life. Especially in these cases, assistive technology can be helpful.

Technological advancements such as assistive robotics have shown to have the ability to help with hand function and assist these individuals in achieving their daily tasks (Alhamad et al., 2023). HandSOME II, a hand exoskeleton developed for hand extension assistance, had users attempt to grip various objects, and then release them. It was shown that subjects were able to complete 13 of the 42 tasks without assistance, and 36 of the 42 tasks with assistance from the exoskeleton (Casas et al., 2021). The hand exoskeleton also allowed users to increase the range of motion in their fingers, as shown below in Figure 1. This provides evidence of the success that users can have with assistance from assistive robotic hand exoskeletons.

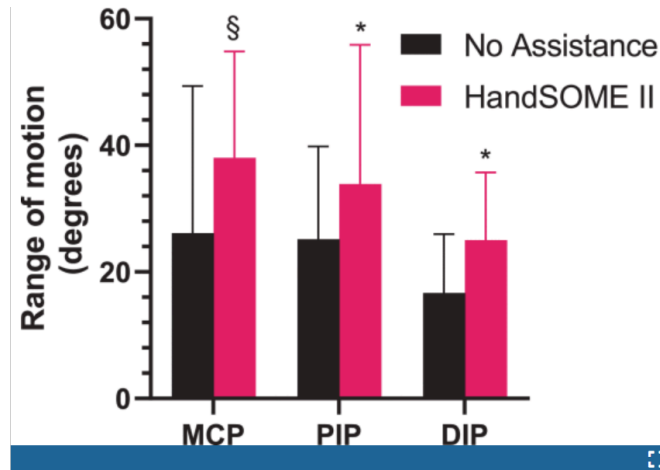


Figure 1: Index finger range of motion with and without exoskeleton assistance (Casas et al., 2021).

Test Condition	Hand	Training	Testing
Control 1	Sound	15	53
Foot button	Affected	1	2
Voice commands	Affected	1	1
EMG commands	Affected	0	0

Table 1: The results of a box and blocks test, one method of testing the effectiveness of hand assistive devices, from a previous study demonstrating the effectiveness of different control mechanisms for a hand exoskeleton (Meier et al., 2019).

While hand exoskeletons for individuals with hypertonicity and spasticity exist, they lack the proper control systems to help this population with activities of daily life (Meier et al., 2019). The HOPE Hand is one such exoskeleton designed for individuals with hypertonicity and spasticity. The voice control system described in this paper can be used in conjunction with the HOPE Hand to properly control the extremities of the users.

2.3 Speech Aphasia and Rehabilitation

Speech is an involved process that engages many parts of the body and brain. The nervous, respiratory, and muscular systems come together to produce speech through the processes of hearing, comprehension, speech formulation, and speech transmission (Owens & Farinella, n.d.). Oftentimes, traumatic brain injury, stroke, and other neurological conditions may affect these processes, therefore halting or impacting speech in a variety of ways.

Certain techniques such as magnetic resonance imaging (MRI), more specifically functional magnetic resonance imaging (fMRI), can give insight into the parts of the brain that play a role in speech. There are many specific sections of the brain involved in language, or the practice of assigning symbols or sounds meaning. These are integral parts of understanding and transmitting speech. Relevant identified areas include Broca's area and Wernicke's area. The primary auditory area and motor cortex are also important for their contributions to hearing and muscular control respectively (Binder et al., 1997).

As discussed, upon being affected by TBI or stroke, an individual may experience aphasia, which is a communication disorder. Because of the range of functions needed to communicate, there are also a range of types of aphasia caused by malfunction of these pathways, the most common being Broca's, Wernecke's, and anomic. This may include having difficulty naming things, recalling words, reading, writing, or understanding visual information (Owens & Farinella, n.d.). People with Broca's aphasia may struggle with creating "novel words" or stumble through words, causing repetitions or slowness (Owens & Farinella, n.d.). There also may be a range of concomitant deficits that one is impacted with based on their specific experience with TBI or stroke, including muscular difficulties such as hypertonicity and spasticity (Owens & Farinella, n.d.).

There are several treatment paths available that a person with aphasia may choose to take. These treatments aim to target the impairments that the person is facing, and therefore may look different based on the specifics of a person's aphasia and concomitant deficits. However, treatment is often based around the concept of brain plasticity. This describes the brain's ability to reassign brain function to changing relevance in stimulus.

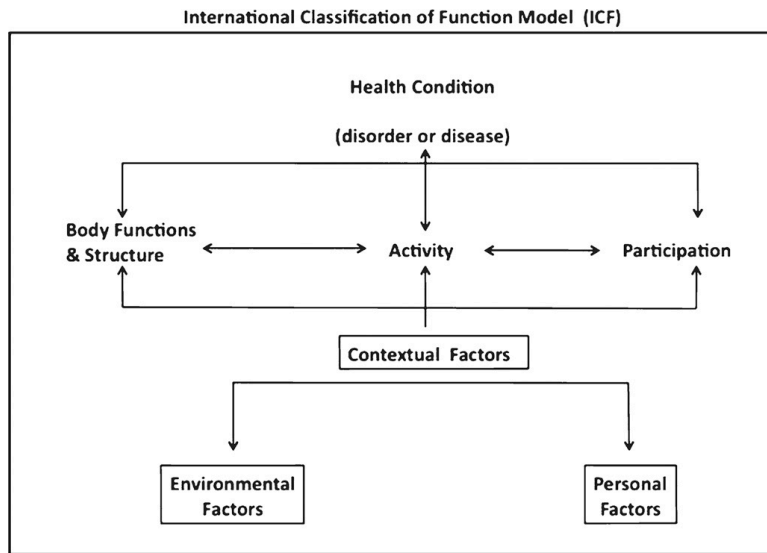


Figure 2: Description of Health Condition (Galletta & Barrett, 2014).

In Figure 2, Barrett and Galletta discuss the contributing impacts that a disease may have on a person, in general and in relation to day-to-day life. This source then discusses the types of intervention that may be successful based on this definition of function, which includes “impairment-based intervention” such as phonological practice, and “functionally oriented intervention” such as support groups and supported conversation (Galletta & Barrett, 2014).

Some other intervention techniques that capitalize on brain plasticity for aphasia include constraint-induced language treatment (CILT), non-invasive brain stimulation (NBS), and transcranial direct cranial stimulation (tDCS). CILT is designed after CIMT, and describes the practice of limiting the body’s reaction to make up for the difficulty in communication using symbols and visual cues, forcing the individual to practice speaking. NBS uses transcranial magnetic stimulation (TMS) to stimulate the brain into using language-areas more. tDCS uses electrical currents across the scalp to similarly stimulate the brain in areas that are underused. Speech production activities allow users to actively engage the speech centers of the brain. Therefore an assistive device that requires speech engagement like the voice control system we are developing may contribute to an improvement in speech production and improved functional outcomes.

2.4 Voice Control

Researchers have implemented voice control to detect user intention and actuation of a hand exoskeleton. An exoskeleton designed for individuals with spinal cord injury called the FLEXotendon Glove-II had voice control using a smartphone for onboard signal processing and a Bluetooth Lower Energy capable microcontroller for controlling the motors. Originally, they required users to press a button on the phone screen to give a voice command, but found that this

reduced user efficacy (Tran et al., 2020). They found that using an online application for the user interface that employed continuous voice control through speech recognition better assisted users, as they could now use the voice control hands free and not divide their attention between tasks and their phone (Tran et al., 2020). Using discrete, task-based voice control has been shown to work well in hand exoskeletons, but it is important to not have a large time delay for the voice control system and motor, as delays of 4 seconds between issuing the command and end of hand attenuation resulted in some voice-control exoskeletons not improving the speed of activities of daily life in users (Tran et al., 2020). Given that many individuals with hand impairments also have concomitant speech disorders such as aphasia, a voice control system tailored to their needs could be highly beneficial. Table 1 shows the results of a study run to investigate potential control mechanisms for the HOPE Hand exoskeleton (Meier et al., 2019). This study showed that a commercially available voice control system would not be sufficient to control the HOPE Hand exoskeleton for individuals with speech aphasia. It was also discussed that EMG sensing and foot pedals would not be practical for the users due to a lack of ability to move blocks and the lack of ability to move the foot pedal respectively.

Having various noise filters is another aspect that was important in hand exoskeletons that used voice commands. One system used with a rigid hand exoskeleton included a noise reduction filter, a loudness filter, and a command detector designed to detect possible activation commands with minimal computational cost (Guo et al., 2022). The noise reduction filter is used to minimize high and low-frequency noise to detect potential human voices, which then enter a loudness detector. The loudness detector detects audio with volume above a certain threshold and uses this to set the system into pre-active mode, where it begins listening to human audio. From there the model could listen to commands, and take action if it recognized the correct speaker.

2.5 Machine Learning and Voice Detection

In order to implement voice control for people with aphasia, we need a way for a computer to reliably recognize commands from their speech. Early automatic speech recognition systems recognize speech by filtering out noise, identifying specific sounds (phonemes) or letters (graphemes), and attempting to map these data points to text or acoustic signals of specific words using machine learning (Jamal et al., 2017). However, in recent applications, simply running audio through a neural network such as a time-delay neural network or a convolutional neural network has become more popular and successful as advances in that field are being made (Mahmoud et al., 2023).

People with aphasia vocalize differently from healthy people whose speech phonemes and graphemes have influenced automatic speech recognition systems. Therefore, a system that analyzes speech on a whole-word level will likely be more effective than one that recognizes individual sounds, just as has been used before to recognize dysarthric speech (Hawley et al., 2007). That being said, convolutional neural networks are conventional for recognizing healthy speech too, and have advantages like noise robustness and a small foot-print for deploying to a minicomputer like the Raspberry Pi (Huang et al., 2015).

Although a large variety of machine learning models have been used to recognize speech, researchers have found the most success using convolutional neural networks with aphasic speech, which evaluate the vocalization at the whole-word level. This differs from time-domain sensitive networks like the time-delay neural networks used to evaluate syllables for correcting aphasic speech ((1)Qin et al., 2020). Even with tone dependent languages such as Mandarin and Cantonese, convolutional neural networks yielded good results for classifying aphasic speech (Mahmoud et al., 2020)((2)Qin et al., 2020). In one study, the curated convolutional neural network had a calculated accuracy of 67.78% and a precision of 73.23% for Mandarin speakers with aphasia (Mahmoud et al., 2023).

Convolutional neural networks work by classifying spectrographic images of speech signals. Mahmoud's study found that the standard wavelet transform used to extract frequency-domain speech images failed to resolve frequency well enough to capture subtly different tones present in Mandarin and aphasic speech, and therefore elected to use the Hyperbolic T-distribution (HTD) outlined another paper (Mahmoud et al., 2006). Mahmoud's present year review of automatic speech recognition systems for aphasic speech found that their custom built convolutional neural network with HTD images outperformed Microsoft and Google's prebuilt speech recognition models, as while as a sophisticated natural language processing approach using both healthy and aphasic training data, ultimately achieving an accuracy of 67.8% and similarly high precision, recall, and F1 scores (Mahmoud et al., 2023). While those results may not be ideal for a voice control system designed for daily use, they do show the ability for speech recognition models that are tuned and trained around aphasic speech to outperform prebuilt speech recognition models like the one tested with the HOPE Hand in table 1.

3 Human-Centric Design Considerations

Every research project has the potential to impact both the general public and its stakeholders in unforeseen ways. Biomedical research that works with a physically disabled target population holds its own set of possible negative impacts that must be explored and mitigated at every stage of research. Most importantly, there needs to be collaboration and input from the stakeholders throughout the designing, data collection, and testing portions of the project to ensure that the stakeholders' point of view be appropriately considered. There also must be a thoughtful analysis of ways in which the products and ideas produced by the research team will change the day-to-day lives of the people they were designed for, as well as the general public. This will encourage responsible research and product development.

3.1 Assessing Broader Impacts

Alongside the metrics used to determine precision and accuracy of the functionality of this device as described in the Evaluation section, certain techniques to assess the potential broader impacts of the work done were also looked at. There are many tools and techniques

available to researchers that can help them consider the broader impacts relevant to their project. For example, the Center for Advancing Research in Society (ARIS) is a community that focuses on providing resources and networking opportunities to researchers for the goal of promoting ethical research. Some of their resources, such as the Broader Impacts Wizard and the Broader Impact Project Rubric, teach about and emphasize relevant topics such as: researcher identity, target population alignment and engagement, infrastructure support, and external partnership (Rutgers University and ARIS). As this program specifically references the National Science Foundation (NSF) and its' expectations on broader impact assessment, there may be certain topics with less relevance to the HOPE Hand. However, it provided a foundation from which we considered this device, and has been used by the research team to ensure thoughtful deliberation about the impacts the HOPE Hand will have on stakeholders.

In order to assess the potential impacts of the voice controlled HOPE Hand, we had a meeting with a speech therapist. When asked, the therapist said that it is possible that the user could become accustomed to issuing a command in a specific way in order to get the voice control system to recognize it, but that it would not have much effect on how the user says those words when not issuing a command. Furthermore, the speech therapist also confirmed that using a voice controlled exoskeleton is an established speech therapy method, however the example they gave used a second person to recognize commands rather than a machine learning model. Based on the information gathered about assessing the research process, it can be determined that topics of interest may include the comfort or discomfort of mobility aids, the difficulty of living with slow or ineffective mobility aids, and the importance for access to and control over one's own individual device, as explained in the following subsection. For more detail on interactions with stakeholders (specifically an interview where preferences for the device was discussed), see Section 6.2.

3.2 Broader Impacts of Machine Learning and Data Collection

There are a few drawbacks to the use of convolutional neural networks, which will make up the machine learning algorithms of the voice-control system. Technology effective in identifying speech raises concerns of privacy and data security. For many people, it is disconcerting to feel like they are being listened to or watched by the machinery in the world around them. It has been shown that mental health is negatively affected by the “ubiquitous use of cameras and voice monitoring equipment in a home environment...” and that it is “a major obstacle to the deployment of smart home systems for the care of the elderly and disabled,” (Yang et al., 2018). There is the possibility that users of the HOPE Hand may feel the same way, as the convolutional neural network will be taking inputted speech data from a microphone on the exoskeleton. Though this global fear of the misuse of technology may not be easily mitigated, it is important for the project team to ensure the least amount of mental stress on the users of this voice control system as possible. Therefore, we strive to be transparent about how we are storing data, as well as the algorithms and models that we are using. This information will be made apparent to anyone whose vocalization samples are being collected by the research

team, as well as in the user manual. These concerns were primarily explored and addressed in the interviews that we conducted over the course of this project. For more information about storing interviewee data, see Section 5.3.5.

3.3 Interviewing and Data Collection

One aspect of this project's methods that may have clear impacts on both the broader target community, and also those who are contributing to data collection, is the data collection process itself. Understanding the meaning of the data we collected in the larger scope of usability is critical in determining the success of the project. As the team previously lacked exposure to and knowledge about the user demographic group, getting to know the data collection participants was beneficial in understanding their needs in relation to this project.

In order to learn about interviewing in general, this team conducted a pilot interview, where we acted out several potential interview situations for practice and to make sure the process was smooth. This led to the reorganization of a few of the speech recording sections, such as making the interview shorter and therefore less vocally intensive on the speakers by removing some of the longer phrases. Before recording participant data, we thoroughly explained the contents of the informed consent form to the interviewees to ensure that they understood the risks associated with the recording of their vocalization data and what their data will be used for, as indicated by the accepted IRB application.

4 Evaluation

The HOPE Hand needs to be used in everyday activities. This section describes how the authors evaluated the potential for the HOPE Hand to be used in everyday activities. The goals for these evaluations were decided upon to minimize user frustration and based on commercially available voice recognition systems for individuals with typical speech. These goals are the gold standard for the field with typical speech. This section includes a description of how users might be affected by misinterpreted or unheard commands and evaluation criteria for the algorithms created

4.1 User Performance

We must ensure that the HOPE Hand does not cause harm to the user or damage their possessions. The two possible situations we have identified where the hand could affect the user in such a way are: 1) if the user is holding a fragile or heavy object in the HOPE Hand and the hand mistakenly hears the ‘open’ command, or 2) if the user commands the hand to grasp a hot or sharp object and the hand does not respond to repeated ‘open’ commands due to the pain or surprise causing inflection in the user’s speech which the HOPE Hand has not been trained on. For people with speech aphasia, the latter may be an especially prevalent issue due to their pre-existing difficulties with speech. While our ability to find out how often these types of errors in practical use is limited by our testing environment, we will measure and report false positive and false negative recognition rate for each command separately.

4.2 Metrics for Objective Algorithm Assessment

Figure 3 below shows the formulas used to calculate the metrics for each part of the voice control system. Table 2 shows the goals created for each of these metrics that our team determined would allow the voice control system to be usable for individuals with the aforementioned impairments.

$$Recall = \frac{TP}{TP+FN} \quad Specificity = \frac{TN}{TN+FP} \quad Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

Figure 3: Formulas for recall, specificity, and accuracy. Abbreviations are True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN).

System	Metrics	Goals
Command Detection	Recall, Specificity	.99, N/A

Speaker Verification	Recall, Specificity, Accuracy	.95+, .90, N/A
Command Recognition	Recall, Specificity, Accuracy	.90, .95, .90
Voice Control Pipeline	Recall, Specificity, Accuracy	.90, .95, .90

Table 2: Metrics and Goals for Voice Control Pipeline Elements

In order to prevent the HOPE Hand voice control system from being a frustrating experience for the user, our goal was to achieve a recall of 90% or greater, and a specificity of at least 95% for the entire system properly processing a command, though this was an optimistic goal. To this end, we should aim for command recognition around the same metrics, since that is the major choke point where an error cannot be made up for further down the pipeline. This would be an impactful, but reasonable improvement from a review study, from 2017, that found three available speech recognition models - one of which bolstered with data from AphasiaBank - that had erred 21-57.8% of the time (Egaji et al., 2019). Recent analysis of more contemporary machine learning models and off-the-shelf speech recognition platforms found that the most successful model was a convolutional neural network with an accuracy of $67.78 \pm 0.003\%$ (Mahmoud et al., 2023), however that was trained using Mandarin speech which is highly tone dependant, so English could yield different results. Furthermore, we have our own data to add to that from AphasiaBank, so we expect our model to out-perform that which Egaji et al. trained on just AphasiaBank data. The command recognition model is split into two convolutional neural networks, one for each command, and the accuracy measures for each model was collected separately.

Besides identifying which command is issued, we must also be able to differentiate commands from ongoing speech and verify that it is the intended user who is issuing the command. For differentiating commands from ongoing speech, since full sentences should not be recognized by the command recognition convolutional neural network, we will optimize this to favor accidentally accepting part of a sentence over accidentally not accepting a command, aiming for 99% of commands being accepted. For speaker verification, since it is more likely that the user is issuing a command than a malicious actor, we will also make speaker verification favor accepting the speaker over denying them. In addition to the functionality of the voice control system, the model must reliably send the perceived signal to the HOPE Hand. For research purposes, we will report accuracy of the HOPE Hand as a whole as well as for each of the algorithms that make it up, but for our specific use case we will focus more on the rate of false positives and false negatives.

Despite the abundance of research on voice control systems, they often lack speaker verification and only test the automatic speech recognition system as a whole, and therefore largely exclude detail on the individual functions that make up their automatic speech

recognition systems. For the HOPE Hand’s voice control system, since aphasic speech may have different effects on our algorithms’ abilities to filter out sentences, verify the speaker, and identify which command has been issued, each function was tested individually. To measure practical performance of the noise preprocessing system, we will report specificity. In order to measure the amount of false negatives caused by the preprocessing system, it was tested on single word command collected from people with aphasia for training the command recognition algorithm, then reported false negative error rate as the percent of commands that did not make it past preprocessing.

5 Methodology

5.1 User Experience Model

The design of the HOPE Hand voice control system is based on the experience of the user. Following the description of the proposed user interaction flow in Figure 4 below, the user will first turn the hand on. Once connected, they will say the word “open” or “close” in order to open or close the hand. If the hand malfunctions, the user will retrieve an error signal and will retry the dictation. If the hand succeeds in recognizing the command and opening/closing, it will move to the correct position. The user may turn the hand off at any time.

Note that the activation key is a feature that has not yet been implemented. Because it is heavily recommended to be a part of the final design of the voice control system, we considered this an integral part of the user flow interaction.

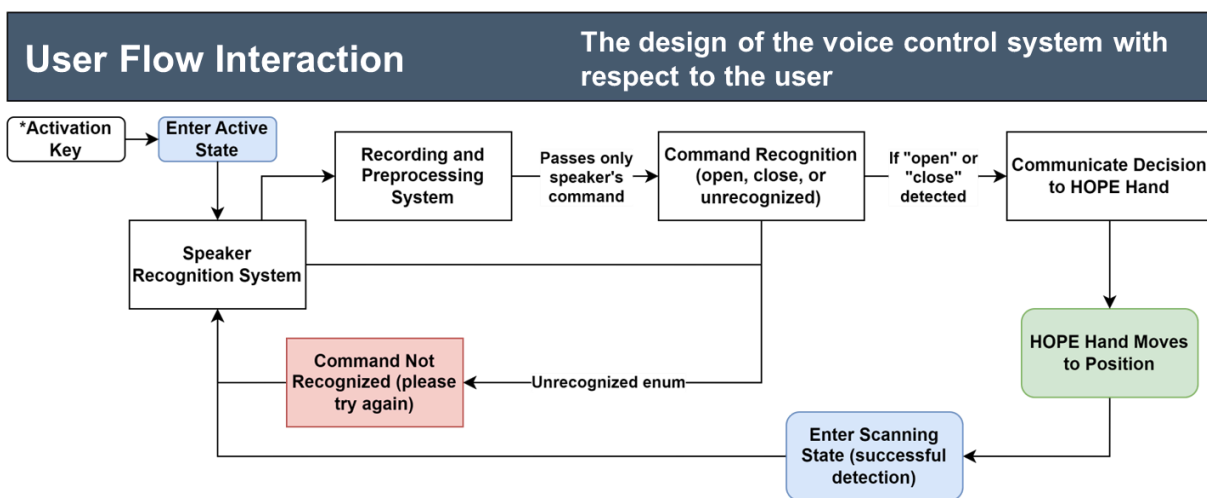


Figure 4: State diagram of the overall system from the perspective of the user. This provides insight into the UI and how it might be designed to reach the goal of optimized design for the users.

5.2 System Design Requirements

The system needed to meet certain minimum requirements to be a viable product for potential users. Currently available voice recognition systems and other potential systems were researched to determine goals that were both comparable to current solutions and useful for the potential clients. It was decided that the voice control system would need to have an accuracy of $90\% \pm 10\%$, respond to user input within 5 seconds, and ignore words that are not commands by labeling them as "notopen", "notclose".

5.3 Database Curation

In order to create the voice control system, it was necessary to curate a database that could be used to train and improve the accuracy of the algorithm. Since the voice control is designed to work for people with aphasic speech, it was required to input aphasic speech data into the database. Open source databases like AphasiaBank were found, but there was still a lack of aphasic data, particularly for words like "open" and "close" that were required for our training purposes (AphasiaBank: MacWhinney et. al). To mediate this, our team met with participants with aphasic speech via zoom to collect vocalizations and integrate them into our database. We also supplemented this aphasic data with typical speech data in our training, as this helped improve our model by increasing the data size, therefore making it more robust. The final form of our database combined aphasic and typical speech, having folders separating the word "open" from other words ("notopen"), and separating "close" from other words ("notclose").

The organization hierarchy of this database may be seen in Figure 5 below. The data is first split into two categories: vocalizations and noise. The noise folder has not yet been implemented, but is recommended for future work. This folder should be filled with sounds that may be found in certain environments in which people may use their HOPE Hand, and recorded by the microphone that the HOPE Hand uses to record. Though some noise is currently filtered out during the preprocessing step of the command recognition pipeline (see Figure 23), the implementation of a specific noise folder in the database will allow for a more robust way to identify background noise as opposed to speech.

Vocalizations are then determined to be healthy or aphasia speech patterns. Each category is then organized into location of origin (collected, AphasiaBank, Mozilla Commonvoice, Google Mini Commands, etc.). Each file is then labeled with the word that is said and given a unique name and timestamp combination.

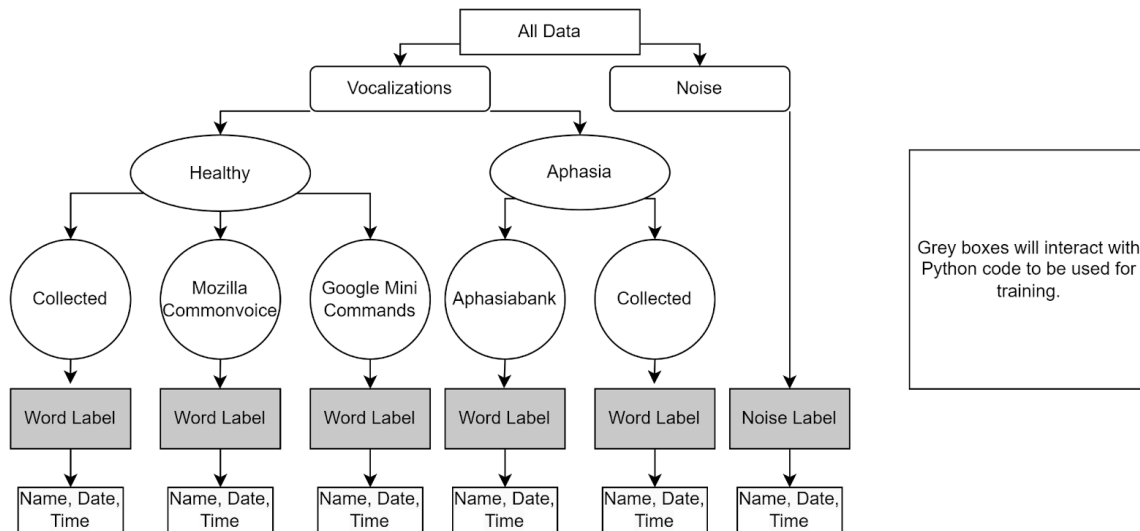


Figure 5: Hierarchy of Database

As indicated in Figure 5, the word labels will be key in training the command recognition algorithm. However, the other categorizations are necessary. The name and time stamps work to indicate each unique file’s contents, and the categories higher in the hierarchy, such as healthy vs aphasia or collected vs AphasiaBank is helpful in directing the Python file responsible for training the model in how much data to use from which categories. This allows different models to be created for different users, accuracies, or effects.

Figure 6 indicates a more streamlined view of the data available. There are ultimately two final models, both created and run separately, which will work in an alternating pattern based on which state the state machine is running in when appropriate. The model on the left describes a model with effectively two data training categories: “open” and “notopen”, which recognizes the word “open” and when “open” is not said. The model on the right has the categories “close” and “notclose” in a similar fashion. Effectively, this is the code categorization most relevant for what the main command recognition Python file, main.py, interacts with.

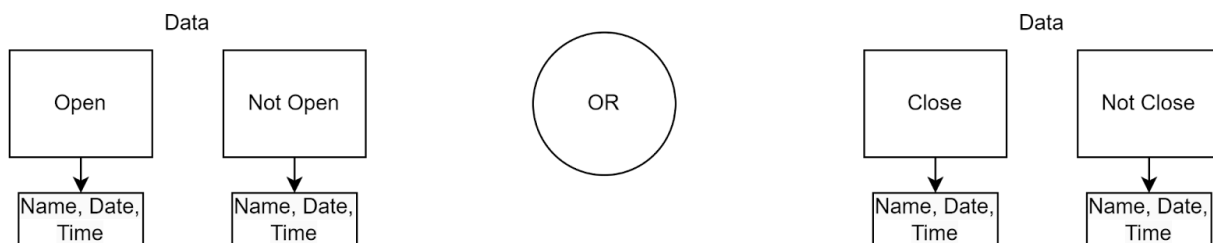


Figure 6: Hierarchy of Database from Model Trainer Perspective

For data to successfully train models using the algorithm developed, there are certain specifications for the audio files. Each audio file we put in was converted to have a sampling rate

of 16KHz and to be mono, not stereo. When audio files were used to train the models without these two requirements, the algorithm either did not work or would only be trained on the spectrograph of part of the word, not the full word.

5.3.1 Google Mini Speech Commands

The Google Speech Commands dataset is an open source dataset designed to help train and evaluate keyword spotting systems (Warden, 2018). A smaller version of this dataset was used for training and testing of our command recognition model described in section 5.44. This version contains 8 different English words, each said 1,000 times by a variety of typical speakers. It contains the words “yes”, “no”, “stop”, “go”, “left”, “right”, “up”, “down”, which were chosen as they are common words at the core of English vocabulary.

A challenging problem for command recognition is ignoring speech that does not contain trigger words that the algorithm is looking for (Warden, 2018). Although this database does not contain audio from aphasic speech nor the words “open” or “close”, it was still useful in our model to help recognize what is not “open” or “close”. This dataset provided samples to our algorithm of 8 words that differed from open and close, helping ensure that our end model is less likely to predict open or close when another word was said. This is the dataset that had been used in a Tensorflow tutorial that we based our command recognition model on (*Simple audio recognition: Recognizing keywords*), so it was very prevalent in early iterations of our models.

These files have an undescribed amount of diversity of speakers, though it can be clearly seen that there is a significant amount, and there are 1000 vocalizations in the form of 1 second clips per word. The clips are all sampled at a rate of 16 kHz.

5.3.2 Vocalizations Collected from The Team

Our team found it difficult to find enough clear audio samples of the words “open” and “close” through open source databases. Having enough data on these words is crucial for the success of our system, as these are the two words that it will be looking for. The Tensorflow tutorial *Simple audio recognition: Recognizing keywords*, which was modified to create our command recognition system, was trained on a dataset that used one-thousand samples of each word. Using this as a reference to the preferred amount of data for a command recognition system, we took measures to increase the amount of training data we had. Each team member recorded 30 samples respectively of themselves saying the words “open”, and 30 samples of “close”. This provided our team with 120 new audio samples for each word to input into our databases, increasing our amount of data with the goal of improving the accuracy of the command recognition model. These files are 1 second long with a sampling rate of 16 kHz after proper conversion.

5.3.3 Mozilla Common Voice Dataset

In order to train the speaker verification system, we needed data that was labeled by the speaker rather than what word was being said. For this we used the Mozilla Common Voice Delta Segment 15.0. The delta segment was sufficient for getting a small sample of phrases to train the i-vector system used in speaker verification, but we also found that the larger corpus contained thousands of opens and closes embedded in these phrases. Unfortunately, since we had no way to automatically separate those from the phrases that contained them, we could not use them. If there was a procedure to automatically label the audio by index, these could be extracted and used to drastically increase the amount and variety of training data. Such a labeler would also allow for much more audio from AphasiaBank to be used since their labeling was not as precise as we would need to get proper training data.

5.3.4 AphasiaBank Data

Given the large percentage of aphasic speakers among those with hypertonicity and spasticity in the arm, we need to make sure that people with aphasic speech are well-represented in the data we use for training our command recognition. We found that AphasiaBank, an online database, contained dozens of hours of precisely labeled aphasic speech.

AphasiaBank is an open source database containing vocalizations of aphasic speech from interviews or discussions with people with aphasia. Many researchers have contributed data to AphasiaBank which provides a wide variety of information. There were various types of aphasia available, but as our project focused on Broca's aphasia, we aimed to collect that type of data the most. AphasiaBank transcribed each video on the word level with indices of the labeled speech in milliseconds. Using these transcriptions and the video files, we used two procedures to isolate specific words such as "open" and "close". The isolated clips are made to be the length of the word said, so each of these files are different lengths. They have a sampling rate of 16 kHz.

The first strategy we developed was to process the .mp4 files found on AphasiaBank as quickly as possible using Python (The Python Code & DevCommunity). As the transcripts to the videos were accessible via .cha files alongside timestamps of each word said, the following code shows in Figure 7, an example of the procedure used to convert the downloaded .mp4 file into an audio (.wav) file, and clip it by converting the timestamps in the accompanying .cha file from milliseconds to individual samples. This left only the target words of "open" or "close". This code made use of the library pydub and moviepy.editor.

```

from moviepy.editor import *
from pydub import AudioSegment

video = VideoFileClip("thompson08a.mp4")
video.audio.write_audiofile("open0thompson08a.wav")
audio = AudioSegment.from_file("open0thompson08a.wav")

start_time = 158799
end_time = 163609
cut_audio = audio[start_time:end_time]
cut_audio.export("cutopen0thompson08a.wav", format="wav")

```

Figure 7: Code to Process AphasiaBank Videos

We then developed a second, more effective procedure to automatically isolate these words using a MATLAB script. In this script, we added a 100ms buffer to the indices specified in the transcript in order to account for instances where the word was cut off and in order to avoid making audio clips that were much smaller than our other data. The data in this database was not great for our use case since their interviews were conducted in small echoey interview rooms, and since AphasiaBank goes back to 1988 they did not have access to high fidelity audio recording equipment with echo cancellation either.

With both of these procedures working to process AphasiaBank files, over 400 usable .wav files became available to train the algorithms that constitute the voice control system deliverable.

5.3.5 Study Data

In order to help increase the amount of aphasic speech data in our database, we conducted an IRB approved study (IRB # 24-0082) in which we held online meetings with two different participants and collected their speech data. This effort not only increased our data size, but proved valuable by including speech data from participants with aphasia who could potentially utilize the HOPE Hand’s voice control system. Training the model using samples from a participant’s voice increases the likelihood of a higher accuracy when the same individual issues commands later on to the system.

Once the participants agreed to be in the study and signed the IRB consent form, our team met them in a Zoom call. The IRB consent form was reiterated to them in the call, and once the participants made it clear that they understood and agreed, the data collection began. Participants were asked to move out of the view of the camera or turn off their video in order to help keep their identity private, and then the audio recording began. Our team recorded the audio on the Zoom call computer, and on a phone as well to provide a backup in case one of the audio files got lost or corrupted.

A slideshow our team made prior was then screen shared to the participant. The goal of this slideshow was to give the participants specific words to say and encourage a variety of tones and inflections. The first 9 slides prompted the participants to repeat sentences from the Harvard sentences list like “Always close the barn door tight”. This gave us data on how the user says “open” or “close” while reading in a sentence, and provided other words that could be used in the

“notopen” and “notclose” folders. Once complete, our team moved on to having the participants repeat 70 single words. These words consisted of the words “open” and “close” repeatedly, and similar sounding words like “opposing” and “broken”. Our team also chose to alter the capitalization for some repetitions of the target words, for example having “oPEN” instead of “open”. This helped make our data more robust by having the participant say the words with different tones and emphasis, as it will likely change when speaking / commanding in their activities of daily life.



Figure 8: Images used in IRB Study

Finally, our slideshow depicted multiple images of an open and closed jar (see Figure 8), and an open and closed door. We instructed our participants to say the word “open” to open the closed jar or door, and vice versa with the word “close”. To open and close the jar and door after participant instruction, we used a simple animation. The goal of this was to help mimic how the participant may command something, such as the HOPE Hand, to open and close. This differed from the previous slides, as a person may speak differently when simply reading a word then they may speak when thinking of the word themselves and then commanding it.

Once these meetings were complete, our team securely stored the full audio on a drive with access restricted to project members with IRB permission. We then went through the interview audio for both phone and computer recordings, cutting out the words we needed in ~1 second clips and saving them as separate files with no identifying information about the participant in the name. With this, we now had over 25 new instances of “open” and “close” for each participant with aphasic speech, and over 20 instances of other words to help recognize what is not open or close.

5.3.6 Augmented Data

The data we have personally collected from our study participants is tailored to training our command recognition algorithm to detect open and close, the only issue is that asking someone with aphasia to repeat “open” and “close” hundreds of times is tedious and unrealistic, so we do not have enough data from our two participants alone. In order to increase the volume of audio from study participants that the command recognition model can be trained on, we

decided to augment the data using the MATLAB `audioDataAugmenter` class. From this class we applied 4 augments to each instance of “open” and “close”: speeding up or slowing down the audio by 0-20%, shifting the pitch up or down a few semitones, applying a slight positive or negative volume gain, and increasing or reducing the amount of background noise in the audio.

5.4 Voice Control Pipeline Approach

The voice control system can be broken down into various steps. Moving down the pipeline, the HOPE Hand starts by recording audio, checking if the audio contains a command or part of a sentence, then checking to make sure that the person issuing the command is a user that has been enrolled into speaker verification, before finally checking if the user is telling the hand to open or to close. At any of the checks if the audio does not pass, then we throw it out and wait for the next clip of recorded audio. The first two checks serve to weed out scenarios where the hand could erroneously activate and hopefully save battery by reducing the amount of times we have to predict “open” or “close” with our command recognition convolutional neural network. For a flow chart showing the path a speech signal takes through the pipeline, refer to Figure 22.

5.4.1 Recording

Since we want the system to pick out instances of “open” and “close” from a live mic feed, we will need to break that live feed up into clips that are big enough to include an entire “open” or “close” command said by the user. The audio recorder is set up to record 1 second long windows of audio with 500ms overlap between one window and the next, as seen in Figure 9. This allows for a second of overlap from one clip to the next in order to account for cases where a command could be cut off if it’s at the end of the clip. This current configuration allows for any command less than one half second long that is cut off at the end of one clip to be fully captured by the next. Since the command recognition model is trained on one second long audio clips, these clips will be cut down to one second later in the pipeline.

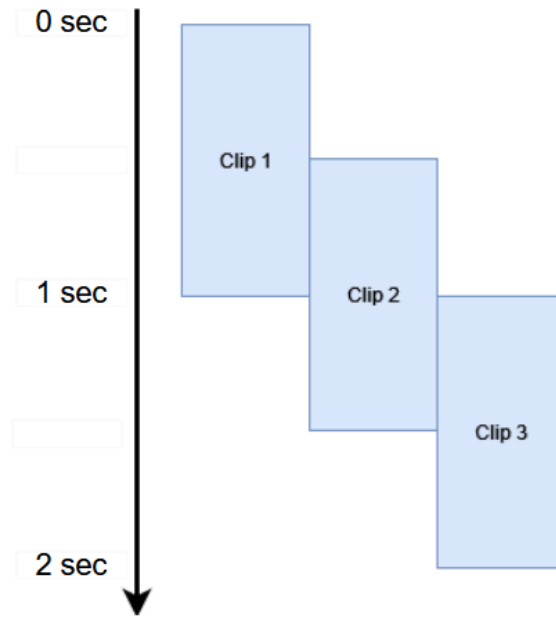


Figure 9: Recording Timeline

The main drawback of this method is that the user will have to wait until the end of the clip that their command is in to finish processing before the HOPE Hand responds. The worst case for this would be if the user issued a command at the start of the clip and had to wait until the clip finished in order for it to be processed. In order to finetune the length of the clips to be as short as possible while only cutting off commands very rarely, one could extract the mean and standard deviation of the length of “open” and “close” from AphasiaBank and our study participants to find the percent of commands that would be in a given length range.

We conducted a proof of concept to better understand the functioning of this procedure, the code for which can be seen in Figure 10. First, the libraries wave and pyaudio are imported (Python Documentation & Python). These libraries are used to save audio in .wav form and read in a livestream respectfully. Then, the recording function is run and the stream recording, ‘recorded_audio’, is saved under a uniquely generated name, ‘recorded_audio (X).wav’ where X increases after each recording is saved (SarahDev).

```

stream.stop_stream()
stream.close()

actualname = 'yes'

def unique_file(basename, ext):
    actualname = "%s.%s" % (basename, ext)
    c = itertools.count()
    while os.path.exists(actualname):
        actualname = "%s (%d).%s" % (basename, next(c), ext)
    return actualname

# https://medium.com/@sarahisdevs/create-a-voice-recorder-using-python-daadd9523e98

#WAVE_OUTPUT_FILENAME = "recorded_audio.wav"
actualname = unique_file('recorded_audio', 'wav')
wf = wave.open(actualname, 'wb')
wf.setnchannels(CHANNELS)
wf.setsampwidth(p.get_sample_size(FORMAT))
wf.setframerate(RATE)
wf.writeframes(b''.join(frames))
wf.close()

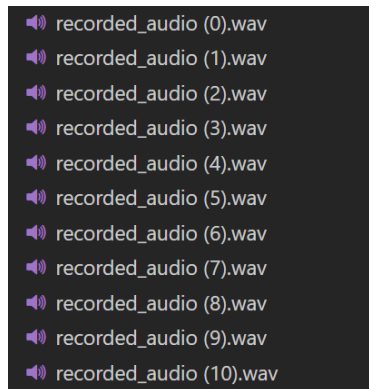
print("Recording saved as ", actualname)

return np.frombuffer(b''.join(frames), dtype=np.int16)

```

Figure 10: Code for Recorder Proof of Concept

Figure 11 shows the result of running these functions, the audio that is recorded has been saved into unique files. As these recordings are taken before the audio is processed and commands are identified, all files are saved, not just those that produce predictions. These files may be played back.



```

🔊 recorded_audio (0).wav
🔊 recorded_audio (1).wav
🔊 recorded_audio (2).wav
🔊 recorded_audio (3).wav
🔊 recorded_audio (4).wav
🔊 recorded_audio (5).wav
🔊 recorded_audio (6).wav
🔊 recorded_audio (7).wav
🔊 recorded_audio (8).wav
🔊 recorded_audio (9).wav
🔊 recorded_audio (10).wav

```

Figure 11: Recorded Audio Saved

The significance of this exercise is to confirm the way the audio recorder processes the live stream of audio that it is fed from the microphone. Because it functions in the way it does, creating these saved files, we were able to move forward with processing.

5.4.2 Command Detection

In order to reduce the possibility of the hand activating when the user says “open” or “close” in the middle of a sentence, we decided to check if a command is being issued by

checking how long the user’s speech lasted. To find the indices of where speech occurs, we use MATLAB’s `detectSpeech` function with a 45ms long Hamming windows at 35% overlap from one to the next, and a 240ms gap between words for them to be considered separate speech. We then check if any speech is shorter than 750ms to consider it a command and check it with command recognition. The `detectSpeech` function fails under noisy conditions and generally overestimates the length of speech, to mitigate this error we remove any noise that’s quieter than 5% the volume of the loudest sample in the audio signal. It is worth noting that even if `detectSpeech` works perfectly, this method could still fail if the command is at the start of a sentence but at the end of the clip it’s recorded in, or if the user pauses to think during a sentence. It is also important to note that this method will fail if the user takes longer than 750ms to dictate their command, which is a significant possibility for the population afflicted with speech aphasia, but even for people with speech aphasia, if they are regularly using the HOPE Hand they should get used to saying the commands quick enough.

5.4.3 Speaker Verification

Speaker verification is important as it helps ensure that only the voice control system only listens to commands from its specified user. Without speaker verification, someone nearby the HOPE Hand user could say “open” or “close” and the hand may listen to it, even though the actual user did not state the command. For speaker verification, we used a linear discriminant analysis (LDA) of i-vectors extracted from different speakers’ speech (Prince & Elder, 2007). This was a standard approach for speaker verification before modern neural network techniques helped it evolve into today’s x-vector systems (Wang et al., 2018)(Snyder et al., 2018), but since that involves calibrating the LDA with a neural network, we opted for an i-vector system due to memory and power concerns. We specifically use MATLAB’s i-vector system to do the math for us, then compare the results from the LDA with a threshold where the likelihood of a false positive is twice that of a false negative since the vast majority of the time it will just be the user issuing commands.

The i-vector system was trained on data from the Mozilla Common Voice dataset and some of the data we collected. Since our database layout only separates files by what word is said and where we got the data, we used a csv to map the file locations to a unique id for each different speaker. To find the threshold for converting the LDA likelihood outputs to a decision on whether the input audio signal is from an enrolled user or not, we used MATLAB’s `detectionErrorTradeoff` function (Mathworks, 2021) which graphs the rate of false positives with respect to the rate of false negatives for various possible decision thresholds. This wouldn’t always give the same error rates as live testing on different mics, so from there the threshold can be tuned heuristically.

5.4.4 Command Recognition

The command recognition part of this pipeline works to identify what is being said by the user. This process makes use of the following steps.

Command Recognition Pipeline Step	Purpose of Pipeline Step
Integration of the MATLAB engine	Integrates command detection and speaker verification into the command recognition function procedure
File conditioning	Ensures data is usable by the model trainer
Creation of the command recognition model	Establishes and saves a usable model to be used by the main command recognition file (main.py)
Command identification	Ensures a command is being said, discourages false positives
Creation of configuration file	Allows for easy access to variables important for customization
Connection to HOPE Hand GUI	Allowing the command recognized to impact the status of the HOPE Hand

Table 3: Command Recognition Pipeline Steps

5.4.4.1 Overview of Python Code

The command recognition model was trained using code adapted from the TensorFlow tutorial: *Simple audio recognition: Recognizing keywords* (TensorFlow). This TensorFlow code takes a selection of .wav files and produces waveforms and spectrograms, representing the audio. A convolutional neural network is used to train the models on these spectrograms.

Using a Jupyter Notebook, a file named `vr_w_application.ipynb` makes use of this tutorial code, with added functionality relevant to the HOPE Hand project. This added functionality includes certain preprocessing steps, such as a procedure that ensures that the files inputted for training the model are the appropriate file type and audio type. Certain variables are also changed to fit our models' needs, such as batch size, split of training vs validation data, the data path of the target data and the classes of these files, and the target number of epochs and patience threshold for the training process. For more information on the libraries and resources used by this system, as well as detailed descriptions of functions written, see Appendix A.

5.4.4.2 Integration of the MATLAB Engine

In order to implement the command detection and speaker recognition procedures into the command recognition process, it is necessary to call these functions in Python. This begins with connecting to the MATLAB Engine, which is a tool that allows for communication between the two programs, as shown in Figure 12.

```
import matlab.engine
eng = matlab.engine.start_matlab()

print("Connected to Engine")
```

Figure 12: MATLAB Engine Importation

Reference the two blocks of code below (Figures 13 and 14). Figure 13 shows the MATLAB code containing the function for checkUserCommand, which is used to determine three statements: whether or not a command is found, whether or not the speaker is recognized, and whether or not both of these things are true. These values are then stored and returned as an array in the variable “user”.

The second block of code, Figure 14, is in Python (in main.py). After initializing the MATLAB engine, the MATLAB function is then called with reference to the engine using eng.checkUserCommand. This method allows the passing of variables (here, we pass the function checkUserCommand its variables defined in MATLAB as ivs, audio, and fs. In the case of this use, we define iv (which stands for ivector) using another MATLAB function which returns the appropriate iv (see eng.loadiv()). We define audio as the audio clip received by the recorder, and we define the sampling frequency (fs) as a constant of 16000.

```
function user = checkUserCommand(ivs, audio, fs)
%CHECKSPEAKERCOMMAND Summary of this function goes here
% return true if a user enrolled in ivs has spoken for less than half a
% second
audio = double(audio)';
fs = double(fs);
speaker = verifySpeaker(audio, ivs);
disp("User recognized: ");
disp(speaker);
command = cc2(audio, fs);
disp("Command Recognized: ");
disp(command);
user = speaker & command;
user = [speaker, command, user];
end
```

Figure 13: MATLAB Code Showing checkUserCommand Function

```

iv = eng.loaddiv()

def predict_mic():
    audio = record_audio()
    tf = eng.checkUserCommand(iv, audio, 48000)
    if not tf[2]:
        print("User or Command Unrecognized")
        if not tf[1]:
            print("No Command Found")
        if not tf[0]:
            print("Speaker Unrecognized")
        return "nothing", 0
    else:
        spec = preprocess_audiobuffer(audio)

        ...

prediction = loaded_model(spec)
#softmax
print(prediction)
label_pred = np.argmax(prediction, axis=1)
command = commands[label_pred[0]]
confidence = prediction[0][label_pred[0]]
#print("Predicted label:", command)
return command, confidence

```

Figure 14: Example of Use of eng.checkUserCommand

Other relevant MATLAB function files and Python files which are used in this pipeline but not shown in this example can be found on the github address in Appendix A.

5.4.4.3 File Conditioning

A major part in ensuring that every data point can be used by the deep learning model is creating a file conditioning system that can ensure that the data, whether coming from the AphasiaBank dataset, the Google dataset, a self-created dataset, or other, are compatible. Python code was used to ensure that the files taken from the database are all .wav files. These files are then converted into mono, as opposed to stereo (*GeeksforGeeks & Vašina*). This code can be modified for future iterations of this project based on expected input and model training characteristics. Downsampling data with sampling rates inconsistent with the model training procedure is also a type of file conditioning. This is further expanded upon in Section 6.1.

5.4.4.4 Creation of the Command Recognition Model

Though there are many steps in training the model, loading a dataset is the first. Once the files are conditioned, you can use the `audio_dataset_from_directory` function from keras, shown in Figure 15.


```

train_ds, val_ds = tf.keras.utils.audio_dataset_from_directory(
    directory=data_dir,
    batch_size=64,
    validation_split=0.2,
    seed=0,
    output_sequence_length=16000,
    #label_mode="binary",
    subset='both')

```

Figure 15: Keras `audio_dataset_from_directory` function

Next, the dataset is converted from flat audio signals to spectrographic images that the convolutional neural network can process. This example used the TensorFlow short-time Fourier transform (stft) function as is shown below in Figure 16. This is then mapped onto the dataset using the map function in Figure 17.

```

def get_spectrogram(waveform):
    # Convert the waveform to a spectrogram via a STFT.
    spectrogram = tf.signal.stft(
        waveform, frame_length=255, frame_step=128)
    # Obtain the magnitude of the STFT.
    spectrogram = tf.abs(spectrogram)
    # Add a `channels` dimension, so that the spectrogram can be used
    # as image-like input data with convolution layers (which expect
    # shape (`batch_size`, `height`, `width`, `channels`)).
    spectrogram = spectrogram[..., tf.newaxis]
    return spectrogram

```

Figure 16: Converting waveform to spectrogram Tensorflow

```

def make_spec_ds(ds):
    return ds.map(
        map_func=lambda audio,label: (get_spectrogram(audio), label),
        num_parallel_calls=tf.data.AUTOTUNE)

```

Figure 17: Mapping onto dataset

Next, the model and training protocol must be defined. Figure 18 shows the architecture of the model, and Figure 19 shows the block of code that calls for the model training specifically (Tensorflow).

```

model = models.Sequential([
    layers.Input(shape=input_shape),
    # Downsample the input.
    layers.Resizing(128, 32),
    # Normalize.
    norm_layer,
    layers.Conv2D(32, 3, activation='relu'),
    layers.MaxPooling2D(),
    layers.Conv2D(64, 3, activation='relu'),
    layers.MaxPooling2D(),
    layers.Dropout(0.1),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.1),
    layers.Dense(num_labels),
])

```

Figure 18: Defining the Architecture

```

EPOCHS = 20
history = model.fit(
    train_spectrogram_ds,
    validation_data=val_spectrogram_ds,
    epochs=EPOCHS,
    callbacks=tf.keras.callbacks.EarlyStopping(verbose=1, patience=4),
)

```

Figure 19: Training Model

The following is the output of this block of code. It shows the loss (which is an indicator of how inaccurate the model currently is), training accuracy, and validation accuracy of the model at each step:

Epoch 1/20

1/101 [.....] - ETA: 3s - loss: 0.2433 - accuracy: 0.9062101/101
[=====] - 4s 41ms/step - loss: 0.2065 - accuracy: 0.9331 -
val_loss: 0.5808 - val_accuracy: 0.8142

Epoch 2/20

101/101 [=====] - 4s 41ms/step - loss: 0.1813 - accuracy:
0.9418 - val_loss: 0.5669 - val_accuracy: 0.8374

Epoch 3/20

101/101 [=====] - 4s 40ms/step - loss: 0.1585 - accuracy:
0.9504 - val_loss: 0.5985 - val_accuracy: 0.8310

Epoch 4/20

```

101/101 [=====] - 4s 44ms/step - loss: 0.1397 - accuracy:
0.9565 - val_loss: 0.6027 - val_accuracy: 0.8374
Epoch 5/20
101/101 [=====] - 5s 46ms/step - loss: 0.1428 - accuracy:
0.9529 - val_loss: 0.5916 - val_accuracy: 0.8490
Epoch 6/20
101/101 [=====] - 5s 45ms/step - loss: 0.1185 - accuracy:
0.9611 - val_loss: 0.5629 - val_accuracy: 0.8606
Epoch 7/20
101/101 [=====] - 6s 55ms/step - loss: 0.1176 - accuracy:
0.9655 - val_loss: 0.6007 - val_accuracy: 0.8542
Epoch 8/20
101/101 [=====] - 6s 58ms/step - loss: 0.1064 - accuracy:
0.9683 - val_loss: 0.6920 - val_accuracy: 0.8387
Epoch 9/20
101/101 [=====] - 6s 63ms/step - loss: 0.1221 - accuracy:
0.9655 - val_loss: 0.6516 - val_accuracy: 0.8555
Epoch 10/20
101/101 [=====] - 6s 57ms/step - loss: 0.0933 - accuracy:
0.9703 - val_loss: 0.6829 - val_accuracy: 0.8361
Epoch 10: early stopping

```

After saving the model, the procedure of which may be seen in Figure 20A, it will appear in the relevant folder as seen in Figure 20B. This saved model will be loaded into the main Python command recognition file called main.py for use in command recognition.

```

model.save("saved_model2")

INFO:tensorflow:Assets written to: saved_model2\assets
INFO:tensorflow:Assets written to: saved_model2\assets

```

Figure 20A: Saving Model

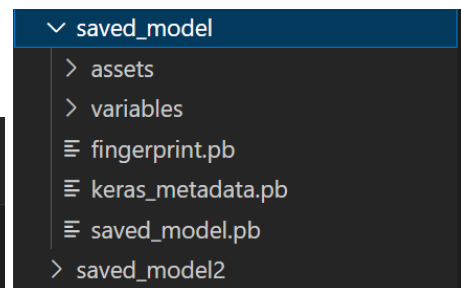


Figure 20B: Saved Model

5.4.4.5 Command Identification

Command identification was put in place to ensure that the user is saying “open” or “close” as an actual command, rather than saying either of these words in a regular sentence. In order to diminish the rate of false positives, we tuned the command identification system to work best with our model. This ensures that the system does not assume a command that it is not sure of for the sake of returning a value (AssemblyAI).

Figure 21 shows the terminal output produced when printing the command recognition “prediction”. As an example, it shows the confidence rating per word for when “open” is said two times and for when no word is said two times. It shows the results specifically when the confidence parameter is chosen to be greater than 4. This was tuned heuristically based on what resulted in the most accurate prediction, however it may be changed based on factors such as background noise, speaker, microphone quality, and any changes in data quality.

As shown in Figure 21, “open” is the second confidence value listed (4.62... or 4.73...) and, because these values are above 4 and greater than the confidence values for “not open”, is therefore reported to be the correct command. Below this, all confidence values are under 4, and therefore no command is reported.

```
tf.Tensor([[ -6.035523   4.6261563]], shape=(1, 2), dtype=float32)
Predicted label: open
tf.Tensor([[ -6.21806   4.739651]], shape=(1, 2), dtype=float32)
Predicted label: open
tf.Tensor([[ 0.2662695   0.24235417]], shape=(1, 2), dtype=float32)
tf.Tensor([[ 0.419549   0.12066908]], shape=(1, 2), dtype=float32)
```

Figure 21: Confidence Value Example and Unorganized Output of Command Recognition

5.4.4.6 Creation of Configuration File

The research team created a practice representative config file called ‘config.ini’ for the purpose of accessibility to relevant variables. Config parser is a tool that allows Python to read in config files. We used ConfigParser to open config.ini and navigate through it (Python Documentation). With this, the team is able to identify and iterate through our classes, variables, and values of these variables for use in our code.

The variables included in this code include commands that may be identified, confidence values, the sampling rate of the recording, and the time in seconds of each recording.

5.4.5 Updated Pipeline

During testing, we decided that centering the speech signals around spoken words and including non-vocal background noise in our training data was necessary for optimizing the performance of our command recognition model. These changes led us to develop an updated voice command pipeline layout and test it for the command “open”.

The first of two changes we made was to instead record 2-second audio clips and replace command detection by cutting them down to 1-second centered around the largest instance of detected speech. Since each 2 second audio clip overlaps 1 second with the next and each audio signal is clipped down to 1 second, it will take no more than 1 second from the end of the command for it to be fully recorded.

The second change was making our command recognition model more robust by augmenting the existing data and adding about 40 instances of data collected in very windy conditions as well as roughly 300 1-second clips recorded during a conversation in a noisy room

to the training set. This includes 10 instances of open said in windy conditions and 42 instances of it said in a noisy room.

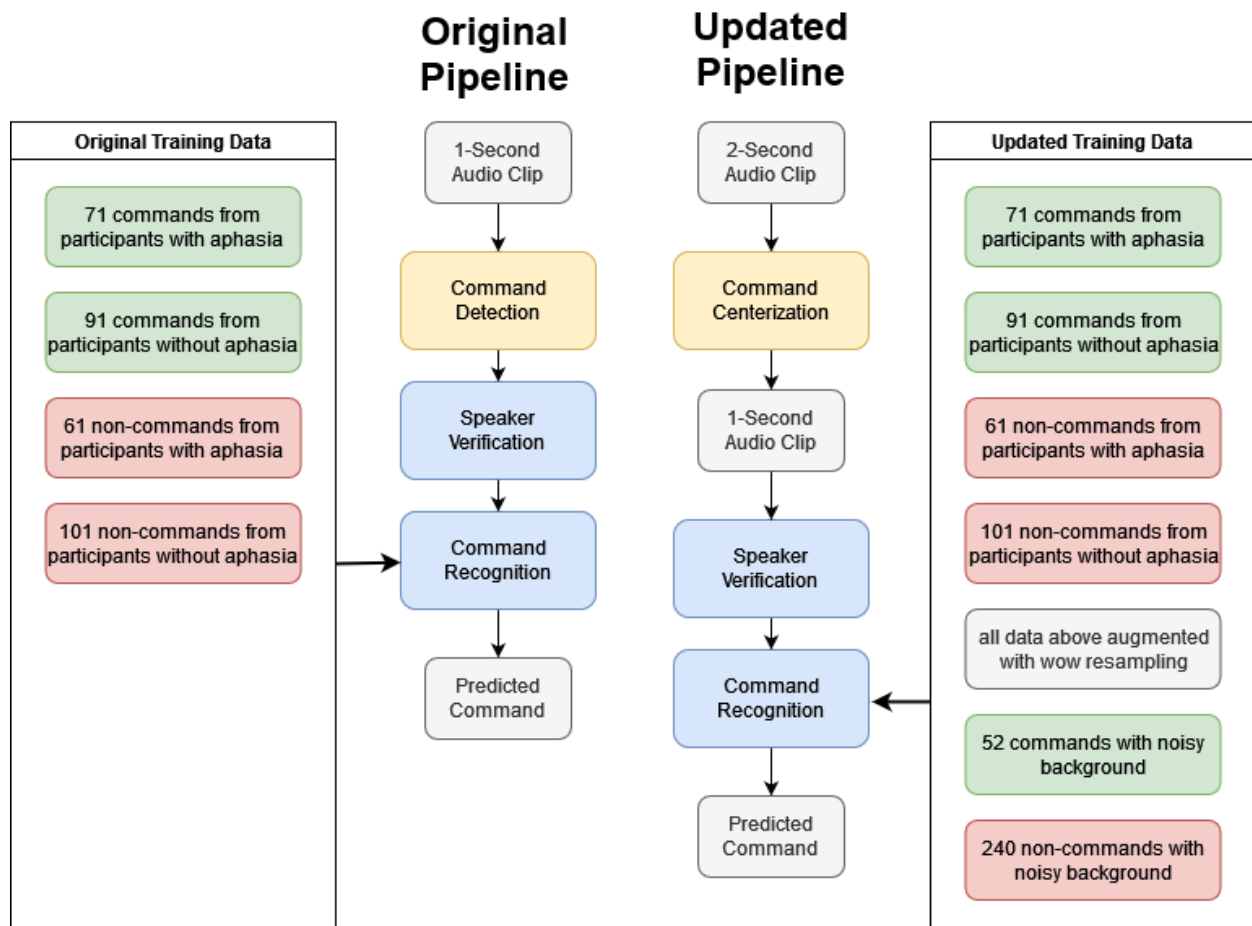


Figure 22: Updated Pipeline Flowchart

5.5 Connecting to the HOPE Hand

5.5.1 State Machine For Identifying Open and Close

The first step in the HOPE Hand GUI integration is setting up the state machine, which will identify which state the hand is in (open or closed) and follow a different procedure depending on the state. The flowchart below describes the state machine’s theoretical functionality, when the state is open, it will use one AI model which looks for the command “close” and when the state is closed, it will use a second AI model which looks for the command “open”. Each individual model will be trained with different data categories.

The “open” model, for example, will be trained with two categories of data: “open” and “not open” (which will contain the word “close” as well as other words that are not open). In future iterations, there may be a “noise” folder as well, to ensure the algorithm can identify what noise that may be found in locations the hand is used in might sound like.

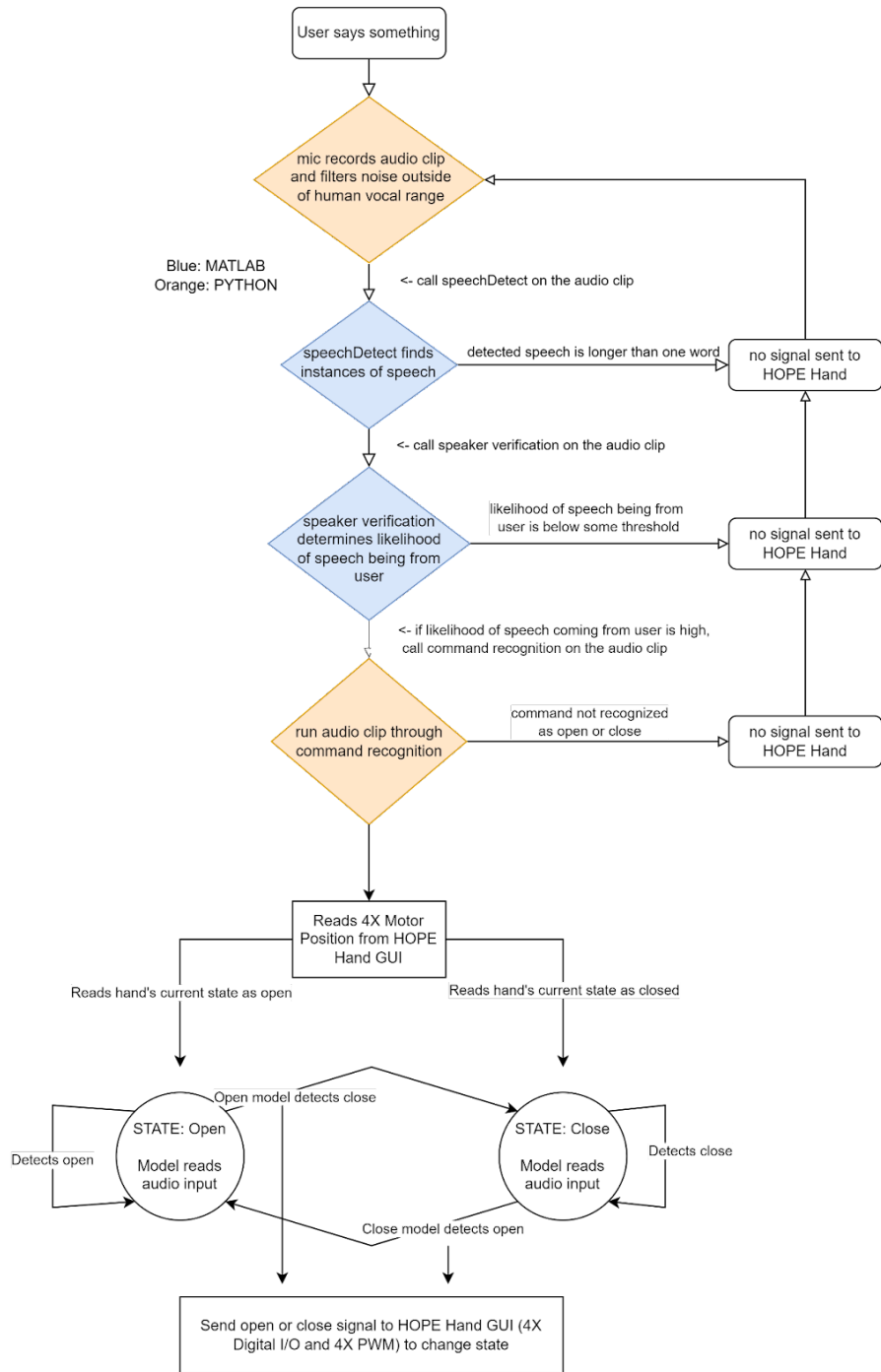


Figure 23: State Machine Organization Flowchart

The state machine is implemented in the code through two functions, each called in the Python file main.py. These functions, readopen(), and readclose() are shown in Figures 24, 25, and 26. The confidence values for each function are individually customizable, and the words that the models expect to hear ('open' vs 'notopen' or 'close' vs 'notclose') are also customizable.

```

def readopen():
    while True:
        command, confidence = predict_mic()
        if confidence > .5:
            print("Predicted label1:", command)
            print("Looking for stop")
            sys.stdout.flush()
            if command == "stop":
                createcommandfile(state)
                #if close is detected, send signal to close hand, change state
                #pause to let hand change
                #terminate()
                break

        command2, confidence2 = predict_mic2()
        if confidence2 > .5:
            print("Predicted label2:", command2)
            print("Looking for stop")
            sys.stdout.flush()
            if command2 == "stop":
                createcommandfile(state)
                #if close is detected, send signal to close hand, change state
                #pause to let hand change
                #terminate()
                break

```

Figure 24: Code Description of readopen()

```

def readclose():
    while True:
        command, confidence = predict_mic()
        if confidence > .5:
            print("Predicted label1:", command)
            print("Looking for left")
            sys.stdout.flush()
            if command == "left":
                createcommandfile(state)
                #in the open models, if open is detected, send signal to open hand, change state
                #pause to let hand change
                break

        command2, confidence2 = predict_mic2()
        if confidence2 > .5:
            print("Predicted label2:", command2)
            print("Looking for left")
            sys.stdout.flush()
            if command2 == "left":
                createcommandfile(state)
                #in the open models, if open is detected, send signal to open hand, change state
                #pause to let hand change
                break

```

Figure 25: Code Description of readclose()

These functions are called by the main function in the main file. As can be seen in Figure 20, if the state is open, the model is set in a certain way. This is customizable independently from the identified closed state model. Then, the readopen() function will be called and the state will be set to close.

```

while True:
    while state == 'open':
        # Loads model
        loaded_model = models.load_model("saved_model12")
        readopen()
        print("CHANGING STATE TO CLOSED")
        state = 'close'

    while state == 'close':
        # Loads model
        loaded_model = models.load_model("saved_model12")
        readclose()
        print("CHANGING STATE TO OPEN")
        state = 'open'

```

Figure 26: readopen() and readclose() Called in Main

5.5.2 GUI Communication

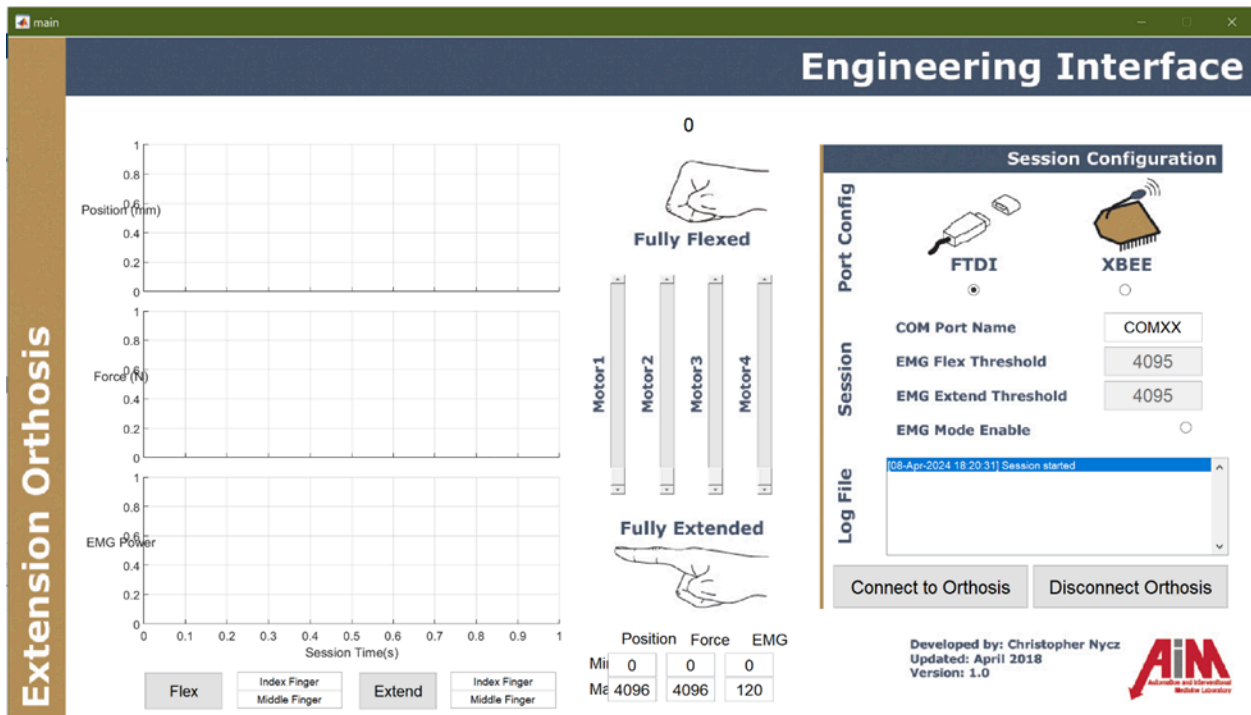


Figure 27: Image of GUI

The MATLAB GUI is an interface created by the creators of the HOPE Hand that allows for communication between code and the exoskeleton. This interface has slider settings for position indicators and controls for different joints on the exoskeletons and the option for customizing these positions for a “Flex” or “Extend” state. More information about the MATLAB GUI can be found in Appendix A.

We identified the need for the command recognition system to communicate with the HOPE Hand. The options for communication were identified as importing MATLAB functions (bypassing the GUI), serial communication (on two laptops), virtual serial ports (on one laptop), IP address communication (on one laptop), or file writing and reading or other hard-coded messaging system (on one laptop). The Python code sends the state to the MATLAB GUI, which physically opens or closes the exoskeleton.

Preferably, the established GUI would be used, and for lack of ease of access to resources, serial communication via two laptops would be difficult. Because of these reasons, we explored IP address communication and hard-coded messaging.

5.5.2.1 File Reading/Writing

The current successful procedure used is the file reading/writing system. This allows for communication on the same laptop without using the MATLAB engine. Each time a command is recognized and there is a change in state, the Python code clears all text files from this folder and then writes a .txt file to the folder that the GUI is located in. The GUI will call a function written in MATLAB intermittently which will check for the existence of said text file, and then it will set the flex/extend states appropriately.

The code in Figure 25 and 28 describes the part of this communication in Python. The `readclose()` function, shown in Figure 25, calls “`createcommandfile(state)`”. This function then looks for the correct file location (the folder that holds the GUI file) and clears text files. It then creates a text file called either “`open.txt`” or “`close.txt`” depending on the state.

```
def createcommandfile(comm):
    data_dir = Name of File Directory
    for file in os.listdir(data_dir):
        file_path = os.path.join(data_dir, file)
        if file.endswith('.txt'):
            os.remove(file_path)
            print('folder successfully cleared')
    name = comm + '.txt'
    path = os.path.join(data_dir, name)
    with open(path, 'w') as newfile:
        # newfile.write(comm)
        print('file created')
```

Figure 28: Creating .txt File Based on Command

The MATLAB function called in the Main GUI file sets Flex and Extend as global functions and then checks for a file called ‘`open.txt`’ or ‘`close.txt`’ in the relative file path. Should it find one of these files, it sets the Flex and Extend variables to true or false. True indicates that the hand should be in said state, and False indicates that the hand should not be in that state. As the Main GUI file is already set to read these variables in its functionality, the hand will continue

as normal, opening or closing based on slider differences, “Flex” or “Extend” button presses, or voice commands.

```
if isfile('open.txt')
    Flex = true;
    disp("Flex = " + Flex)
    Extend = false;
end
if isfile('close.txt')
    Flex = false;
    disp("Flex = " + Flex)
    Extend = true;
end
```

Figure 29: GUI Code to Extend or Flex HOPE Hand

In the Main GUI file, the function call shown in Figure 29 is placed in a try/catch loop preceding when the hand checks the values of the Flex and Extend variables. It consistently takes under 1 second to iterate through this process (as indicated by the GUI while running the code).

5.5.2.2 IP Address Communication

Using the local IP address on one laptop to communicate between the command recognition code and the MATLAB GUI did not end up a viable solution to solve the problem of integrating our code into HOPE Hand functionality. However, it proved helpful for conceptual understanding of the problem at hand and is promising as a possible solution moving forward.

For this approach, there are three actors: the Python command recognition code (server), which will recognize the command and send it as a string over MATLAB engine to the MATLAB function (in this case, it is called `practicefunc2()`), the MATLAB function (client), which will identify the data sent by the server and set the variables “Flex” and “Extend” to true or false, and the MATLAB GUI, which will periodically call `practicefunc2()` in order to receive information from the server. This code is based on the MATLAB documentation for TCP clients and socket functions (MATLAB).

An IP Address connection requires an IP address, which is variable to the machine used and location. In this case, a local address “127.0.0.1” is used, and a port is identified as 65432. In Python, these are used in the creation of a sender socket, which identifies the connection, binds the host to the port, and uses “`conn.sendall()`” to send a byte message over the connection. In this case, the state, identified as “open”, will be transferred into bytes and sent to the `practicefunction2` using the MATLAB engine. This can be seen in Figure 30. This code is called when a command is recognized and the state is changed.

```

state = 'open'

host = '127.0.0.1'
port = 65432

#conn = <socket.socket fd=3812, family=2, type=1, proto=0, laddr=

def socketconnectfunc():
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        print("looking for connection")
        s.bind((host, port))
        s.listen()
        conn, addr = s.accept()
        with conn:
            print(f"Connected by {addr}")
            if state == 'open':
                conn.sendall(b'open ')
                tf = eng.practicefunction2
                print(tf)
            else:
                conn.sendall(b'close')
                tf = eng.practicefunction2
                print(tf)

```

Figure 30: Socket Communication from Python

The MATLAB client uses a tcp client to receive data. The tcp client is defined as `tcpclient("127.0.0.1", 65432)`, using the same IP address and port as before. `Practicefunction2` then reads the data from the data stream and sets the “Flex” and “Extend” variables as seen in Figure 31.

```

at GUI initialization
pause(1);
if t.NumBytesAvailable > 4
    data = read(t, 5, 'string');
    % data = readline(t);
    if isempty(t)
        disp("Error")
    end
    disp(data)
    if strcmp (data, 'open ')
        Flex == true;
        disp("Flex = " + Flex)
        Extend == false;
    end
    if strcmp (data, 'close')
        Flex == false;
        disp("Flex = " + Flex)
        Extend == true;
    end
end

```

Figure 31: Socket Communication from MATLAB (MATLAB)

In the Main file for the GUI, the connection is identified and `practicefunction2` is called in the loop before the hand checks the variables “Flex” and “Extend”, similarly to the file reading/writing procedure. Unfortunately, this system takes a long time to function. The connection has to be started and stopped each time communication is to occur because IP address communication is meant to act as an interrupt or a messaging system, not a perpetually open means of communication. This means that, while a functional communication method, it was taking about 12 seconds to loop through the code each time, and the connection to the orthosis was timing out and closing.

Either way this is implemented in the final project, the overall idea is to have a messaging system available to both the GUI and the command recognition code. This system must be appropriately quick to allow for communication between the code and the exoskeleton without timing out or causing the commands to back-up.

5.6 Documentation

To ensure our work on the voice control system for the HOPE Hand can be continued, we provided a document outlining the signatures and purpose for every Python/MATLAB function our team writes. The functions themselves will also have detailed comments breaking them down to make it easier for others to understand and update them in the future. Researchers furthering work on the HOPE Hand can also refer to our “Future Work” section to get a sense of what added functionality we would add to the system if we had more time, and how we would go about making those changes.

While the documentation is intended to be for an audience of researchers working on this project, a manual will likely be used by individuals using the HOPE Hand. This manual will outline setup instructions and how to interact with the physical and virtual interfaces. The setup instructions cover connecting the virtual interface with the minicomputer on the HOPE Hand, enrolling your voice into speaker verification. The physical interface instructions cover how to enable and disable the voice control and go over the details of issuing a command. These instructions will also include an explanation on how researchers working with the voice control system can efficiently add data, retrain, and run tests in order to update the speaker verification and command recognition algorithms with more data. This manual has not yet been developed, but is necessary in the final implementation of this project.

6 Results

6.1 Database Curation

Source	Collection Type	Aphasia or Typical	Vocalizations Collected
--------	-----------------	--------------------	-------------------------

AphasiaBank	Curated	Aphasia	“Open” “Close” Sentences
Interviews	Recorded over Zoom	Aphasia	“Open” “Close” Words that sound like “open” Words that sound like “close” Sentences
Team Vocalizations	Recorded live	Typical	“Open” “Close” Words that sound like “open” Words that sound like “close”
Google Mini Speech Commands	Curated	Typical	“Yes” “No” “Go” “Stop” “Left” “Right” “Up” “Down”
Mozilla Commonvoice	Curated	Typical	Sentences with “open” and “close”

Source	Clip Length	Clip Sampling Rate	Number of Files Available	Number of Files Used in Training Final Models
AphasiaBank	Continuous speech, clipped to 1 second	16 kHz	> 400	0
Interviews	Continuous speech, clipped to 1 second	Modified to 16 kHz	235	235
Team Vocalizations	1 second	Modified to 16 kHz	270	270
Google Mini Speech Commands	1 second	16 kHz	8000	0
Mozilla Commonvoice	Sentences, clipped to 1 second for testing	32 kHz, modified to 16 kHz	>100,000	0

Our team was able to curate a database using a combination of collected data from individuals with aphasia, team collected data, aphasic speech from AphasiaBank, and typical speech from Mozilla Common Voice and Google speech commands. For testing and training of the command recognition algorithm, data was split into sections “open”, “notopen”, “close”, and “notclose”. Within the “open” and “close” folder were instances of both healthy and aphasic speakers saying the words open and close. Within the “notopen” and “notclose” folder were

instances of both healthy and aphasic speakers saying words similar to open and close, and a large variety of other random words.

The command recognition models we trained make use of 168 instances of “open”, with 80 of those being aphasic and 88 typical speech, as well as 168 instances of “close”, with 87 of those from people with aphasia and 81 from typical speech. For miscellaneous words, we got 169 different audio files, with 68 being aphasic speech and the remaining typical. Figure 32 below shows some of the data in the “notopen” and “notclose” folder, where the naming convention is the speaker (anonymized) along with the word and instance number of that word, as some were repeated many times.

- Participant1_Broken1
- Participant1_Closet1
- Participant1_Clown1
- Participant1_Go1
- Participant1_Golden1
- Participant1_Gross1
- Participant1_Grow1
- Participant1_Hiccup1
- Participant1_High1
- Participant1_Known1

Figure 32: Example speech data used for command recognition training.

For the command recognition model data, each recorded word is roughly one second long, and in the format of .wav. The sampling rate of each audio file in the set is also at 16kHz, and any audio that was originally larger than this (44.1kHz or 48kHz) was downsampled in order to be consistent and work with the algorithm created. Any future data added to to improve the model must be consistent with the above parameters.

6.2 Second Phase of Interview

We were able to conduct a second interview with one person under IRB #24-0082 focusing not on vocalization collection, but on their expected experience with the HOPE Hand. They were asked questions about their personal preferences for the device and the anticipated circumstances of its use, for example, their preferences for how to turn the device on and off, and in what instances or environments they might look forward to using the hand. Notably, they spoke about using the HOPE Hand for video games and hunting, and explained how it might

have changed how they worked at their old job. They also noted that they would prefer the microphone to be clipped to their shirt, or closer to their face, and that they would be willing to try using the exoskeleton if there was a high risk of it accidentally opening or closing. They indicated a preference for an activation key being used to power the device on or off, and would be willing to spend at least thirty minutes calibrating the HOPE Hand upon receiving the device.

During this interview, we also spoke to the interviewee’s physical therapist. They noted that, in general, their patients have shared with her that they are enthusiastic to participate in research such as the HOPE Hand. They also discussed how combining speech related tasks like issuing a command with physical feedback could help users overcome plateaus in the progression of their physical therapy.

Though our sample size is too small to draw conclusions about the preferences of the intended user population for the HOPE Hand, this conversation gave insight into one person’s experience with their hand impairment and aphasia and how this might interact with the exoskeleton. In the future, it is imperative that conversations be had with stakeholders such as this interviewee. Consideration to their lived experience will make the design requirements for the hand more effective and will help future teams working on this project avoid negative impacts.

6.3 Pipeline Metrics

The final results of each aspect of our voice control pipeline are shown in Table 4 below.

Part of Pipeline	Accuracy	Recall	Specificity	Avg. Time Taken
Command Detection	Not Measured	98%	45%	17ms
Speaker Verification	Not Measured	97%	60%	Not Measured
Command Recognition (Open State)	62%	45%	69%	26ms
Command Recognition (Closed State)	66%	42%	78%	24ms

Table 4: Reported Metrics for Voice Control Pipeline

6.3.1 Command Detection Accuracy

The goal of command detection is to sort out audio recorded during a conversation. This helped to prevent the voice control system from reacting when the user does not intend for the words spoken to actuate the hand. Success for command detection would allow all spoken commands to pass on to the next stage of the pipeline, removing any audio from a conversation. Our team tested command detection on 25 commands and 10 sentences that we collected from

ourselves and aphasic speakers in order to train the command recognition model. The sentences had to be cut into 1 second clips since that is how they would be input to command detection in our current voice control pipeline. For healthy speakers, the command detection algorithm was able to detect commands 100% of the time and weeded out 50% of sentences. When testing command detection on vocalizations from our study participants with speech aphasia, it correctly detected 96% of commands and recognized 39% of sentences as not being commands.

6.3.2 Speaker Verification Accuracy

Speaker verification is implemented to ensure that only the user of the HOPE Hand can issue commands that actuate the exoskeleton. Success for the speaker verification would allow all of the user's speech to pass onto the next stage of the pipeline but not speech from any other speaker. Speaker verification was also tested with the same commands we recorded during data collection, but not full sentences. For healthy speakers, the system recognized an enrolled user 98% of the time, but allowed unenrolled users to issue a command 20% of the time. When vocalizations from our study participants with aphasia were enrolled into speaker verification, the system recognized the user 96% of the time, but allowed unenrolled users to issue a command 60% of the time when tested on 25 commands from the enrolled speaker and 25 commands across 5 unenrolled users.

6.3.3 Command Recognition Accuracy

After creation of our command recognition model, our team did live testing of the system by testing it with our own typical speech directly into the microphone. It was found that the model was able to recognize "open" and words that are not open with an accuracy of 62% and recognized "close" and words that were not close with an accuracy of 66%. The open accuracy was found by having each of the four team members say open 25 times each (total of 100 times), different words similar to open 100 times total, and different words not similar to open 100 times, and calculating how many times it predicted each correctly. Similarly, the close accuracy was found by each member saying close 25 times for a total of 100 times, words similar to close 100 times, and words not similar to close 100 times.

This way of testing helped show how accurate the command recognition system was at predicting "open" and "close", how many times similar words could trip up the system, and how many times random words may be predicted as open or close. The recall shows the percentage of time the system predicted "open" or "close" correctly, which was 45% for open and 42% for close. These values are lower, but if the algorithm was specifically designed to be a bit more likely to predict that the command is not open or close, as false positives where the hand is opened or closed by accident could cause more problems. This is shown with the specificity, where the open state machine was able to recognize words that were not open 69% of the time, and the close state machine was able to recognize words that were not closed 78% of the time.

6.3.4 Updated Command Recognition

While conducting the original test for the command recognition model we noticed a few points for improvement. The results with the updated pipeline outlined in section 5.45 improved significantly from the original pipeline. Command verification was tested with the same speaker who added 32 of the new noisy “open” data samples. Those samples were recorded using the same command centerization from the updated pipeline, so they were the same sort of signal command recognition would be supplied during live testing. In order to achieve the same accuracy with users, they may have to record around 30 instances of them saying commands over background noise. With the updates, command recognition correctly identified commands 24 out of 25 times, or a recall of 96%. It also correctly ignored 24 out of 25 random non-open words, or a specificity of 96%. With words that sound like open, it rejected an unimpressive 13/25 or 52%. Over the ten sentences in the 1st Harvard Sentences List and 5 sentences that included the word “open”, it did not falsely activate once, even when open was said in a sentence.

6.4 Connecting the Voice Control System to the HOPE Hand

The voice control system connected to HOPE Hand as described in the methods. The communication time between the MATLAB GUI and the HOPE Hand averaged around 5000 milliseconds \mp 1000 milliseconds, however it reached as low as below 200 milliseconds. TCP IP was tried to connect the Python script and the MATLAB GUI, but the connection was not permanent and would need to be reestablished after each message. It was determined that reconnecting would take longer than the allotted time between user input and actuation so this method of communication was abandoned. The txt files that are now used to communicate between the Python script and the MATLAB GUI are blank because the system would take longer to find, open, and read a file than it would to search for two separate files. The position that the HOPE Hand opens and closes to is controlled in the MATLAB GUI for the index, middle, and thumb fingers, but is hard coded for the ring and pinky fingers.

7 Discussion

There are many existing hand exoskeletons in literature with varying control systems (Alhamad et al., 2023; Casas et al., 2021; Guo et al., 2022; Schabowsky et al., 2010). However, none of the control systems commonly used are practical for individuals with speech aphasia in addition to hypertonicity and spasticity in the hand (Meier, 2019). This project lays the groundwork for a voice control system that is practical for use with the HOPE Hand exoskeleton and users with speech aphasia in addition to hypertonicity and spasticity.

7.1 Pre-Processing

The pre-processing section of the pipeline includes the command verification and speaker recognition. These two sections of the pipeline were within the goal range for the recall statistic.

This allows most of the intended user's commands to get through. The specificity of these sections of the pipeline did not meet the goals set previously in the paper. This means that many non-commands or other speakers may be passed on into the next stage of the pipeline. The balance of a high recall and low specificity was decided upon to minimize user frustration and with the idea that other sections of the pipeline would help to reduce undesirable inputs.

7.2 Command Recognition

The command recognition system did not meet the $90\% \mp 10\%$ accuracy goal established earlier in the paper. This had a cascading effect that meant the voice control system did not meet the goals established to ensure the voice control system was assistive for the potential users of the HOPE Hand. The command recognition model did however show promise that with more training and proper preprocessing of the audio input, the voice control system may be practical for use in activities of daily life.

While conducting live tests of the command recognition model our team noticed two patterns that suggested the system had specific flaws. First we noticed that we could get the model to recognize a command reliably if we said the command at the right timing after the last command line output from the last command. This suggested that it was crucial to have the command centered properly in the 1-second mic clip that was being fed to command recognition. The other pattern we noticed was that non-vocal noise could produce outlier confidence values when input into command recognition; while normal speech would almost always return confidence values between -5 and 5, noises like a chair squeaking or wind would produce confidence values ranging from -60 to 60. This suggests the training data for our model lacks non-vocal background noise data. These patterns informed the changes which lead to the updated voice command pipeline described in section 5.45.

Limited results for command recognition in the updated pipeline suggest it is sufficient to meet our original goal of 95% recall, 90% specificity for words that don't sound too similar to commands. The system was only successful at rejecting words that sound like commands half of the time, but the user will not likely say those words very often outside of sentences, so this is not much of a concern. The updated pipeline was only formally tested on one person, so further research is necessary. The results when the updated command recognition is used on sentences suggests that command detection is not necessary in this pipeline since the updated model does not recognize open in a sentence often.

7.3 Limitations

It is important to highlight the limitations of this work to contextualize the voice control system described in this work. The command recognition system was limited by the lack of training available to the authors at the time of the work. As this work was done over the course of 8 months as a senior capstone project, there was a limit on the time the participants could be recruited and data could be collected.

The authors were not able to test the voice control system and the subsections of the pipeline with live participants with speech aphasia. It was assumed that recorded audio files, not used for training the models, played through a phone or computer next to the microphone used for testing was a sufficient substitute for the lack of live participant data.

7.4 Future Work

Further improvements to the voice control system can be made. Potential future work include an increase in data used to train models, practical testing of the system with aphasic participants, an activation word to interchange when the system should and should not listen to user commands, and tests on MRI compatibility.

7.4.1 Increase in data

Collecting more good typical and aphasic audio and using it to train the models would likely result in higher accuracies, as there would be more data for the algorithm to learn from. In particular, getting more speech data from individuals with aphasia who will use this HOPE Hand would be highly useful. This would help the algorithms be more individualized to each user, thus becoming more adept to successfully predict commands spoken. Any new data added into the database must have a sampling rate of 16KHz and be mono audio in order to work correctly with our current algorithms. If the data is not initially recorded with these specifications, they must be converted. We also suggest replaying audio after conversions to ensure it sounds correct, and using the algorithm to test how the waveforms of added audio look like if needed.

7.4.2 Practical testing

Our team has tested out the accuracy of the voice control in speaker verification and command prediction, but to get a better idea of how it will perform in activities of daily life, more practical testing would be useful. One user test that simulates activities of daily life is the box and block test. This test requires a participant to grab and move as many wooden blocks as possible from one compartment to another of equal size in 60 seconds (Figueiredo, 2011). To evaluate the effectiveness of the HOPE Hand and its voice control system, the user would first be asked to complete this test without using the exoskeleton and then repeat it using the device. The way in which a user is able to move the blocks with assistance from the voice-controlled exoskeleton relative to without assistance will give insight into how the aid may improve their ability to complete tasks in their everyday life. In other words, this kind of evaluation technique will give insight into the functionality of the voice-control system, both in relation to the user and the exoskeleton.

In addition to a user's ability to grab and drop objects in a timely fashion, a goal of the HOPE Hand is to allow the user to hold a variety of different objects. One downfall of the box and block test is the homogeneous nature of the blocks. Therefore, other pick and place tasks should be implemented and tested to ensure the exoskeleton with voice control is able to assist users in working with numerous objects in their activities of daily life. One test that has been

used in the past is having 7 different objects (1 inch block, 2 inch block, 3 inch block, baseball, marble, pen, key) and giving users 30 seconds to pick each object up and place it into a bin (Casas et al., 2021). These objects help simulate common real life objects and require individuals to use different types of grasps and finger movements (Casas et al., 2021). Similarly to the box and blocks test, to evaluate the effectiveness of the voice-controlled HOPE hand, the user will first attempt this test without assistance and then repeat with assistance from the hand exoskeleton. The difference between these two tests would provide more evidence into how effective this system is. It will also provide useful information just seeing how the user interacts with the voice control while simulating activities of daily life.

7.4.3 Activation Keyword

Our team has had various discussions on the idea of using some type of activation keyword for the voice control system in the future. An example application using an activation word in the real world is “Hey Siri” while using Apple’s Siri. Having some way for users to turn on and off the voice control could be useful in numerous scenarios. If one wanted to keep their hand closed around an object like a drink for a longer period of time, temporarily shutting off the voice control system to ensure it does not accidentally have an unintended open would be useful. When asked about this idea, a participant with aphasia stated that they think this would be helpful in the future and that “On” and “Off” would be good keywords to use for this. If this idea were to be implemented in the future, it would require more audio data on the word(s) used as activation keywords.

7.4.4 MRI Compatibility

Ensuring that the voice-controlled HOPE Hand is MRI safe would provide more benefits for its users. Magnetic resonance imaging (MRI) is a medical imaging modality that allows for three dimensional representation of the body. MRI tracks the change in polarity alignment of hydrogen atoms in the imaging space to render a black and white, density map (Nycz, lecture, 2023). MRI machines use powerful magnets to change the alignment of the hydrogen atoms magnetic field, and inductors to sense the relaxation of these fields (Kern et al., 2007). These magnets can cause serious problems when ferrous materials are present in the imaging room because the magnets can displace the objects, or the objects might cause distortions in the image by distorting the magnetic field (Kern et al., 2007). However, MRI can be a useful tool for monitoring brain function during assigned activities (Radiological, 2022). Functional MRI (fMRI) is a form of MRI that can track the blood flow density in the body, typically this method is used to map brain activity. By aligning all the hydrogen ions in the blood before the blood enters the brain and monitoring the change in alignment as the blood flows through the brain fMRI can detect where the blood is flowing in the brain (Radiological, 2022). Many exoskeleton studies use fMRI to examine the effects of the exoskeletons on brain function (Adans-Dester et al., 2022).

MRI results, in summary, hold applications to understand rehabilitation for people with neurological impairments (Radiological, 2022), which will allow for further research into providing greater rehabilitation for individuals with similar neurological disabilities. The HOPE Hand being tested to be MRI safe would allow for brain activation to be studied to further determine the therapeutic effect of the hand.

8 Conclusion

This paper describes the creation of a voice control system for the HOPE Hand exoskeleton. The authors were able to create the groundwork for a database that has instances of typical and aphasic speech saying “open”, “close”, and other miscellaneous words. This was done by using audio files from open source data sets, collecting vocalization data from aphasic participants, and collecting audio from our team. The collection of aphasic vocalizations was necessary to address the lack of aphasic “open” and “close” vocalizations available to the authors. A voice control pipeline with command detection, speaker verification, and command recognition was also created for this project. These models were trained using the audio files in our created database. From there, we were able to connect this voice control system to the HOPE Hand, allowing it to interact with the hand and cause the hand to open and close based on one's command. To aid future researchers a manual describing our code and the decisions made by the authors in creating the voice control system was made.

Although the final voice control system did not fully reach the recall, specificity, and accuracy scores specified in the design requirements, this system does still give a strong start to making the voice-controlled HOPE Hand usable for individuals with speech aphasia and hand impairments. Between the initial database created, algorithm developed with models connected to the HOPE Hand, and manuals for future researchers, the system is in a good place with lots of room to improve and become more usable if the future work is followed.

Acknowledgements

Our team would like to thank our advisors, Tess Meier, Professor Christopher Nycz, Professor Erin Solovey, Professor Yunus Telliel, and Professor Haichong Zhang, for their continued support and guidance throughout this project. Additionally, we would like to extend our gratitude to the individuals who participated in our studies along with any accompanying parties, and speech therapist Sarah Hall for her expertise. We would also like to acknowledge the grant support for AphasiaBank -- NIH-NIDCD R01-DC008524 (2022-2027).

References

- A Complete Guide to Data Augmentation*. (n.d.). Retrieved December 11, 2023, from <https://www.datacamp.com/tutorial/complete-guide-data-augmentation>
- Adans-Dester, Catherine P., et al. "Wearable Sensors for Stroke Rehabilitation." *Neurorehabilitation Technology*, edited by David J. Reinkensmeyer et al., Springer International Publishing, 2022, pp. 467–507, https://doi.org/10.1007/978-3-031-08995-4_21.
- Alhamad, R., Seth, N., & Abdullah, H. A. (2023). Initial Testing of Robotic Exoskeleton Hand Device for Stroke Rehabilitation. *Sensors*, 23(14), 6339. <https://doi.org/10.3390/s23146339>
- AphasiaBank: MacWhinney, B., Fromm, D., Forbes, M. & Holland, A. (2011). AphasiaBank: Methods for studying discourse. *Aphasiology*, 25,1286-1307.
- Ardila, R., Branson, M., Davis, K., Henretty, M., Kohler, M., Meyer, J., Morais, R., Saunders, L., Tyers, F. M. and Weber, G. (2020) "[Common Voice: A Massively-Multilingual Speech Corpus](#)". *Proceedings of the 12th Conference on Language Resources and Evaluation (LREC 2020)*. pp. 4211—4215
- AssemblyAI. *Build Your Own Real-Time Voice Command Recognition Model with TensorFlow*. 2022, <https://www.youtube.com/watch?v=m-JzldXm9bQ>.
- Binder, J. R., Frost, J. A., Hammeke, T. A., Cox, R. W., Rao, S. M., & Prieto, T. (1997). Human Brain Language Areas Identified by Functional Magnetic Resonance Imaging. *The Journal of Neuroscience*, 17(1), 353–362. <https://doi.org/10.1523/JNEUROSCI.17-01-00353.1997>
- Bruce, C., Edmundson, A., & Coleman, M. (2003). Writing with voice: An investigation of the use of a voice recognition system as a writing aid for a man with aphasia. *International Journal of Language & Communication Disorders*, 38(2), 131–148. <https://doi.org/10.1080/1368282021000048258>
- Casas, R., Sandison, M., Chen, T., & Lum, P. (2021). *Clinical Test of a Wearable, High DOF, Spring Powered Hand Exoskeleton (HandSOME II)*. <https://ieeexplore.ieee.org/document/9529163>
- Centers for Disease Control and Prevention. (2023). *Get the facts about TBI*. https://www.cdc.gov/traumaticbraininjury/get_the_facts.html
- Centers for Disease Control and Prevention. (2023). *Stroke*. <https://www.cdc.gov/stroke/facts.htm>
- CircuitPython | *Seed Studio Wiki*. (2023, January 11). https://wiki.seeedstudio.com/XIAO-BLE_CircuitPython/
- Coons, C. (2023). *Focusing on brain injury prevention during awareness month*. UC Davis Health; UC Davis. <https://health.ucdavis.edu/news/features/focusing-on-brain-injury-prevention-during-awareness-month/2023/03>
- configparser—Configuration file parser*. (n.d.). Python Documentation. Retrieved March 15, 2024, from <https://docs.python.org/3/library/configparser.html>

- Convert TensorFlow models | TensorFlow Lite.* (n.d.). TensorFlow. Retrieved November 29, 2023, from https://www.tensorflow.org/lite/models/convert/convert_models
- Cutting Audio Files in Python—The Python Code.* (n.d.). Retrieved December 11, 2023, from <https://thepythoncode.com/assistant/transformation-details/cutting-audio-files-in-python-with-pydub/>
- Douglas Carroll, Hand function in hemiplegia, *Journal of Chronic Diseases*, Volume 18, Issue 5, 1965, Pages 493-500, ISSN 0021-9681, [https://doi.org/10.1016/0021-9681\(65\)90031-7](https://doi.org/10.1016/0021-9681(65)90031-7). (<https://www.sciencedirect.com/science/article/pii/0021968165900317>)
- educ8s.tv (Director). (2020, October 13). *How to install CircuitPython on the Seeeduno Xiao Board (SAMD21)*. <https://www.youtube.com/watch?v=1GKF9u7pVgs>
- Egaji, O. A., Asghar, I., Griffiths, M., & Warren, W. (2019). Digital Speech Therapy for the Aphasia Patients: Challenges, Opportunities and Solutions. *Proceedings of the 9th International Conference on Information Communication and Management*, 85–88. <https://doi.org/10.1145/3357419.3357449>
- Figueiredo, S. (2011). *Box and block test (BBT)*. Strokengine. <https://strokengine.ca/en/assessments/box-and-block-test-bbt/>
- Galletta, E. E., & Barrett, A. M. (2014). Impairment and Functional Interventions for Aphasia: Having it All. *Current Physical Medicine and Rehabilitation Reports*, 2(2), 114–120. <https://doi.org/10.1007/s40141-014-0050-5>
- Getting Started | Seed Studio Wiki.* (2023, January 11). https://wiki.seedstudio.com/XIAO_BLE/
- Guo, Y., Wenda, X., Prahdan, S., Bravo, C., Ben-Tzvi, P. (2022) Personalized voice activation grasping system for a robotic hand exoskeleton glove. *Mechatronics*. <https://doi.org/10.1016/j.mechatronics.2022.102745>
- Hawley, M. S., Enderby, P., Green, P., Cunningham, S., Brownsell, S., Carmichael, J., Parker, M., Hatzis, A., O'Neill, P., & Palmer, R. (2007). A speech-controlled environmental control system for people with severe dysarthria. *Medical Engineering & Physics*, 29(5), 586–593. <https://doi.org/10.1016/j.medengphy.2006.06.009>
- Help for Mu 1.2.*.* (n.d.). Retrieved November 29, 2023, from <https://codewith.mu/en/help/1.2>
- How to convert mp4 to mp3 using python.* (2023, January 19). DEV Community. <https://dev.to/jimajs/how-to-convert-mp4-to-mp3-using-python-1dcf>
- H. Hu, T. Tan and Y. Qian, "Generative Adversarial Networks Based Data Augmentation for Noise Robust Speech Recognition," *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Calgary, AB, Canada, 2018, pp. 5044-5048, doi: 10.1109/ICASSP.2018.8462624.
- Huang, J.-T., Li, J., & Gong, Y. (2015). An analysis of convolutional neural networks for speech recognition. *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4989–4993. <https://doi.org/10.1109/ICASSP.2015.7178920>
- Jamal, N., Shanta, S., Mahmud, F., & Sha'abani, M. (2017). Automatic speech recognition (ASR) based approach for speech therapy of aphasic patients: A review. *AIP Conference Proceedings*, 1883(1), 020028. <https://doi.org/10.1063/1.5002046>
- Kamper, Derek et al. "Use of Cyproheptadine Hydrochloride (HCl) to Reduce Neuromuscular

- Hypertonicity in Stroke Survivors: A Randomized Trial: Reducing Hypertonicity in Stroke.” *Journal of stroke and cerebrovascular diseases* 31.10 (2022): 106724–. Web.
- Kern, Morton. “Which Devices Are Safe for Magnetic Resonance Imaging?” *Hmpgloballearningnetwork.Com*, Cardiovascular Learning Network, Dec. 2007, www.hmpgloballearningnetwork.com/site/cathlab/articles/which-devices-are-safe-magnetic-resonance-imaging.
- Launching the Speech Commands Dataset*. (2017, August 24).
<https://blog.research.google/2017/08/launching-speech-commands-dataset.html>
- MacWhinney, B. & Fromm, D. (2016). AphasiaBank as Big Data. *Seminars in speech and data* 37(1), 10-22. <https://doi.org/10.1055/s-0036-1571357>
- Maguolo, G., Paci, M., Nanni, L., & Bonan, L. (2021). Audiogmenter: A MATLAB toolbox for audio data augmentation. *Applied Computing and Informatics*, ahead-of-print(ahead-of-print). <https://doi.org/10.1108/ACI-03-2021-0064>
- Mahmoud, S. S., Hussain, Z. M., Cosic, I., & Fang, Q. (2006). Time-frequency analysis of normal and abnormal biological signals. *Biomedical Signal Processing and Control*, 1(1), 33–43. <https://doi.org/10.1016/j.bspc.2006.02.001>
- Mahmoud, S. S., Kumar, A., Tang, Y., Li, Y., Gu, X., Fu, J., & Fang, Q. (2020). An Efficient Deep Learning Based Method for Speech Assessment of Mandarin-Speaking Aphasic Patients. *IEEE Journal of Biomedical and Health Informatics*, 24(11), 3191–3202. <https://doi.org/10.1109/JBHI.2020.3011104>
- Mahmoud, S. S., Pallaud, R. F., Kumar, A., Faisal, S., Wang, Y., & Fang, Q. (2023). A Comparative Investigation of Automatic Speech Recognition Platforms for Aphasia Assessment Batteries. *Sensors (Basel, Switzerland)*, 23(2), 857. <https://doi.org/10.3390/s23020857>
- Marciniak, C. (2011). Poststroke hypertonicity: upper limb assessment and treatment. *Top Stroke Rehabil.* <https://www.tandfonline.com/doi/abs/10.1310/tsr1803-179>
- Mathworks. (2021). detectioninErrorTradeoff.
<https://www.mathworks.com/help/audio/ref/ivectorsystem.detectionerrortradeoff.html>
- Mathworks. (2021). ivectorSystem
<https://www.mathworks.com/help/audio/ref/ivectorsystem.html>
- Meier, T., Gandomi, K., Carvalho, P., Fischer, G., & Nycz, C. (2019). *Assisting Hand Movement Of TBI Patients Through Robotic Orthoses*. 10.31256/HSMR2019.43
- Meier, T. B. (2018). *Mechanical Redesign and Implementation of Intuitive User Input Methods for a Hand Exoskeleton Informed by User Studies on Individuals with Chronic Upper Limb Impairments* [Master’s Thesis, Worcester Polytechnic Institute].
- Model conversion overview | TensorFlow Lite*. (n.d.). TensorFlow. Retrieved November 29, 2023, from <https://www.tensorflow.org/lite/models/convert>
- ModuleNotFoundError: No module named “matplotlib” in Python | bobbyhadz*. (n.d.). Retrieved November 29, 2023, from <https://bobbyhadz.com/blog/python-no-module-named-matplotlib>
- MATLAB. (n.d.). *Create TCP/IP server—MATLAB*.
<https://www.mathworks.com/help/instrument/tcpserver.html>
- Nycz, Christopher "Medical Imaging." Lecture 3, Biomedical Robotics, Worcester Polytechnic Institute, Worcester MA, March 28, 2023

- Owens Jr., R. E., & Farinella, K. A. (n.d.). *Introduction to Communication Disorders, a Lifespan Evidence-Based Perspective* (6th Edition). Retrieved September 28, 2023, from <https://plus.pearson.com/products/aa4dbbde-1ffc-4e3e-a966-bcf93c1a743f/pages/af6bed7603bf0d455e77a5407a9e41a2c581f5a5?userPreferredType=read>
- Parker, V. M., Wade, D. T., & Langton Hewer, R. (1986). Loss of arm function after stroke: measurement, frequency, and recovery. *International rehabilitation medicine*, 8(2), 69–73. <https://doi.org/10.3109/03790798609166178>
- Peters, H. T., Page, S. J., & Persch, A. (2017). Giving them a hand: wearing a myoelectric elbow-wrist-hand orthosis reduces upper extremity impairment in chronic stroke. *Archives of physical medicine and rehabilitation*, 98(9), 1821-1827.
- Prince, S. J. D., & Elder, J. H. (2007). Probabilistic Linear Discriminant Analysis for Inferences About Identity. *2007 IEEE 11th International Conference on Computer Vision*, 1–8. <https://doi.org/10.1109/ICCV.2007.4409052>
- PyAudio: Cross-platform audio I/O with PortAudio* (0.2.14). (n.d.). [Python]. Retrieved March 15, 2024, from <https://people.csail.mit.edu/hubert/pyaudio/>
- PyTorch documentation—PyTorch 2.1 documentation*. (n.d.). Retrieved December 11, 2023, from <https://pytorch.org/docs/stable/index.html>
- Qin, Y., Lee, T., & Kong, A. P. H. (2020). Automatic Assessment of Speech Impairment in Cantonese-Speaking People with Aphasia. *IEEE Journal of Selected Topics in Signal Processing*, 14(2), 331–345. <https://doi.org/10.1109/JSTSP.2019.2956371>
- Qin, Y., Wu, Y., Lee, T., & Kong, A. P. H. (2020). An End-to-End Approach to Automatic Speech Assessment for Cantonese-speaking People with Aphasia. *Journal of Signal Processing Systems*, 92(8), 819–830. <https://doi.org/10.1007/s11265-019-01511-3>
- Radiological Society of North America (RSNA) and American College of Radiology (ACR). “Functional MRI (Fmri).” *Radiologyinfo.Org*, Radiology Society of North America, 6 Dec. 2022, www.radiologyinfo.org/en/info/fmribrain#top.
- Realtime Voice Command Recognition*. (2023). [Python]. AssemblyAI - Code Examples. <https://github.com/AssemblyAI-Examples/realtime-voice-command-recognition> (Original work published 2022)
- Rutgers University and ARIS. (2023). *Broader Impacts Wizard*. <https://aris.marine.rutgers.edu/wizard.php>
- Schabowsky, C., Godfrey, S., Holley, R., & Lum, P. (2010). *Development and pilot testing of HEXORR: Hand EXOskeleton Rehabilitation Robot*. Springer Link. <https://doi.org/10.1186/1743-0003-7-36>
- SarahDev. (2023, September 3). Create a Voice Recorder using Python. *Medium*. <https://medium.com/@sarahisdevs/create-a-voice-recorder-using-python-daadd9523e98>
- Shaik, J. (2020, October 24). Reducing Deep Learning Model Size Without Effecting It’s Original Performance and Accuracy With.... *Analytics Vidhya*. <https://medium.com/analytics-vidhya/reducing-deep-learning-model-size-without-effecting-its-original-performance-and-accuracy-with-a809b49cf519>
- Simple audio recognition: Recognizing keywords | TensorFlow Core*. (n.d.). TensorFlow, from https://www.tensorflow.org/tutorials/audio/simple_audio
- Simpson, R. C., & Levine, S. P. (2002). Voice control of a powered wheelchair. *IEEE*

- Transactions on Neural Systems and Rehabilitation Engineering: A Publication of the IEEE Engineering in Medicine and Biology Society*, 10(2), 122–125.
<https://doi.org/10.1109/TNSRE.2002.1031981>
- Splitting stereo audio to mono with PyDub. (2021, March 16). *GeeksforGeeks*.
<https://www.geeksforgeeks.org/splitting-stereo-audio-to-mono-with-pydub/>
- Snyder, D., Garcia-Romero, D., Sell, G., Povey, D., & Khudanpur, S. (2018). X-Vectors: Robust DNN Embeddings for Speaker Recognition. 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 5329–5333.
<https://doi.org/10.1109/ICASSP.2018.8461375>
- Spasticity Treatment - Physical Medicine & Rehabilitation*. UPMC. (2013). UPMC | Life Changing Medicine
<https://www.upmc.com/services/rehab/physical-medicine-rehab/treatments/spasticity-treatment#:~:text=Physical%20and%20occupational%20therapy%20for>
- Tan, Amelia, Jodie Copley & Jennifer Fleming (2023) Clinical utility of a decision-making aid for upper limb neurorehabilitation: applying the Hypertonicity Intervention Planning Model across cultures, *Brain Injury*, 37:7, 572-580, DOI: 10.1080/02699052.2023.2205661
- Teasell, R. W., Foley, N. C., Bhogal, S. K., & Speechley, M. R. (2003). An evidence-based review of stroke rehabilitation. *Topics in stroke rehabilitation*, 10(1), 29–58.
<https://doi.org/10.1310/8YNA-1YHK-YMHB-XTE1>
- Tóth, L., Kovács, G., & Van Compernelle, D. (2018). A perceptually inspired data augmentation method for noise robust cnn acoustic models. In *Speech and Computer: 20th International Conference, SPECOM 2018, Leipzig, Germany, September 18–22, 2018, Proceedings 20* (pp. 697-706). Springer International Publishing.
- Tran, P., Jeong, S., Wolf, S., Desai, J. (2020) Patient-Specific, Voice-Controlled, Robotic FLEXotendon Globe-II System for Spinal Cord Injury. *IEEE Robotics and Automation Letters*, vol. 5, no.2, pp. 898-905. 10.1109/LRA.2020.2965900
- Vašina, L. (2023, October 12). *Answer to “How can I convert a WAV from stereo to mono in Python?”* Stack Overflow. <https://stackoverflow.com/a/77282372>
- venv—Creation of virtual environments*. (n.d.). Python Documentation. Retrieved November 29, 2023, from <https://docs.python.org/3/library/venv.html>
- Wade, B. P., R. Cain, J. (2001). Voice recognition and aphasia: Can computers understand aphasic speech? *Disability and Rehabilitation*, 23(14), 604–613.
<https://doi.org/10.1080/09638280110044932>
- Wang, S., Huang, Z., Qian, Y., & Yu, K. (2018). Deep Discriminant Analysis for i-vector Based Robust Speaker Recognition. 2018 11th International Symposium on Chinese Spoken Language Processing (ISCSLP), 195–199. <https://doi.org/10.1109/ISCSLP.2018.8706632>
- Warden, Pete. “Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition.” *arXiv.Org*, Cornell University, 9 Apr. 2018, arxiv.org/abs/1804.03209.
- wave—Read and write WAV files*. (n.d.). Python Documentation. Retrieved March 15, 2024, from <https://docs.python.org/3/library/wave.html>

- Yakubu A. Ibrahim, Juliet C. Odiketa, & Tunji S, Ibeyimi. (2017). Preprocessing Technique in Automatic Speech Recognition for Human Computer Interaction: An Overview. *Annals. Computer Science Series*.
- Yang, G., Yang, J., Sheng, W., Junior, F. E. F., & Li, S. (2018). Convolutional Neural Network-Based Embarrassing Situation Detection under Camera for Social Robot in Smart Homes. *Sensors (Basel, Switzerland)*, 18(5), 1530.
<https://doi.org/10.3390/s18051530>

Appendix A: Voice Recognition Description of Files, Libraries, and Resources

The purpose of this Appendix is to provide an overview of the files relevant to the finalized voice recognition pipeline and the resources that impact their functioning (such as libraries and other imports).

Relevant software includes code-editors that run Python and MATLAB. Visual Studio Code and MATLAB version R2023b was used for this project.

GITHUB Link

See repository under the following link for relevant files:

<https://github.com/aprozear/Voice-Control-of-the-HOPE-Hand-Exoskeleton.git>

Library Directories

Configparser

This module allows for the live communication between a configuration file and a Python file. For more information, visit: <https://docs.python.org/3/library/configparser.html>

IPython

While this library performs several functions, it is implemented in this project through its display feature, which organizes data to be displayed. For more information, visit:

<https://ipython.readthedocs.io/en/stable/>

Itertools

This library provides looping methods for working with iterables. For more information, visit: <https://docs.python.org/3/library/itertools.html>

Matlab.engine

This API allows for the live communication between MATLAB and Python functions.

For more information, visit:

<https://www.mathworks.com/help/matlab/matlab-engine-for-python.html>

Matplotlib

This library provides functions to create graphs, plots, and other visualizations from data. For more information, visit: <https://matplotlib.org/stable/index.html>

Numpy

This library provides many mathematical functions. For more information, visit:

<https://numpy.org/doc/>

OS

This module provides access to operating system functionality. For more information, visit: <https://docs.python.org/3/library/os.html>

Pathlib

This module allows for the organization and manipulation of file paths. For more information, visit: <https://docs.python.org/3/library/pathlib.html>

Pyaudio

This library provides methods for recording and manipulating audio. For more information, visit: <https://pypi.org/project/PyAudio/>

Pydub

This module allows for audio handling. For more information, visit:

<https://pypi.org/project/pydub/>

Scipy

This library allows for the use of certain algorithms and mathematical functions. For more information, visit: <https://scipy.org/>

Seaborn

Similar to Matplotlib, this library provides functions to create graphs, plots, and other visualizations from data. For more information, visit: <https://seaborn.pydata.org/>

Sys

This module provides functionality specific to the system. For more information, visit: <https://docs.python.org/3/library/sys.html>

Tensorflow

This library provides many resources and functions relevant to the development of artificial intelligence. For more information, visit: https://www.tensorflow.org/api_docs

Wave

This library allows for the interfacing with WAV files (opening, closing, reading, writing WAV files). For more information, visit: <https://docs.python.org/3/library/wave.html>

Jupyter Notebook

The jupyter notebook file `vr_w_application.ipynb` imports libraries `tensorflow`, `os`, `pathlib`, `matplotlib.pyplot`, `numpy`, `seaborn`, `layers` from `tensorflow.keras`, `models` from `tensorflow.keras`, `display` from `IPython`, and `AudioSegment` from `pydub`.

Once the models are saved, there are several different relevant python files for command recognition.

Python Files

`Main.py` applies the convolutional neural network models to a live feed of audio and sends a message to a MATLAB HOPE Hand GUI to change the position of the hand. This file imports the libraries `numpy`, `sys`, `os`, `tensorflow`, `models` from `tensorflow.keras`, `configparser`, and `matlab.engine`, as well as certain helper functions from peripheral files such as `record_audio`, `record_audio2`, and `preprocess_audiobuffer` from `recording_helper.py`, `recording_helper2.py`, and `tf_helper.py` respectively.

In `main.py`, a `commands` array is instantiated and filled with respect to inputs from the configuration file. The state of the hand is also instantiated as “open”, as the HOPE Hand GUI always initially opens the exoskeleton upon connection. Function `createcommandfile` takes the input of an intended command and produces a text file for the purpose of communicating with the GUI.

In relation to reading audio, the Python code records for one second. However, in order to decrease the chance of the audio ending its recording while someone is speaking and missing the command, there are two recorders implemented. They begin recording in a staggered manner, so there is always a recording beginning each second. `Main.py` function `prediction` takes the input of an integer 1 or 2, indicating which of the alternating predictions the function will expect. It then applies the appropriate model based on the state of the hand and prints a prediction.

Function `readopen` and `readclose` identify the state and call the two other functions to create a prediction and send a signal to the GUI.

Helper files `recording_helper.py` and `recording_helper2.py` use libraries `pyaudio`, `numpy`, `wave`, `os`, `itertools`, `configparser`, `wavfile` from `scipy.io`, and `scipy.signal`. These files each record live audio, in alternating fashion, every second, as explained in section 5.41. They also resample these recorded audio files to 16 kHz, as to be compatible with the models. Helper file `tf_helper.py` uses the libraries `numpy` and `TensorFlow`, and sets up the creation of the spectrograms used by `vr_w_applicatio.ipynb`.

MATLAB GUI

The current GUI in use is an application run through the Guide program in MATLAB, This program communicates with the HOPE Hand exoskeleton via FTDI connection. To use this, one must connect to the hand by entering the appropriate COM port and pressing “Connect to Orthosis”. Once connected, the hand’s status will show on the log file and each encoder and ball screw positions will be graphed. There are four motors (motor 1 is the thumb and motor 4 is the pinky and ring fingers combined). There are flex and extend sliders which control these motors, as well as settings where the motor values can be numerically set. For our purposes, the flex settings for the index finger is 2300 and the middle is 2500. For extend, the index is 1000 and the middle is 800. The pinky and ring fingers also have a set value of 2300 and 1000 for flex and extend, though this value is not changeable via the GUI.