Design of a Smart American Sign Language Translator

# Introduction

Physical disabilities can prevent people from immersing themselves in everything society has to offer. Deafness, in particular, has a significant impact on a person’s life in that it prevents one from communicating through speech. Studies show that deaf persons are less likely to make use of medical services and are more prone to psychological disorders – disorders stemming from the difficulty of forming meaningful connections with hearing counterparts [1]. There have also been recorded instances of discrimination of the disabled in employment due to their physical disabilities, including deafness [2], [3]. In the case of deafness, communication with others is the overarching issue, bringing with it a surge of inevitable side-effects. This paper seeks to develop a translation tool that relies on the training of a neural network to recognize different ASL (or American Sign Language) gestures.

The following project, tentatively called the “Sign AI,” seeks to design an ASL (American Sign Language) translation tool having deaf persons as the target demographic. The translation tool will take the form of a Windows-based GUI program with which users can interact for various purposes. A deep learning model will be trained to recognize ASL through pre-processed EMG data and orientation data in the form of quaternions calculated, all acquired from the Myo armband developed by Thalmic Labs. The final product is a gesture-based translator, providing translation for each gesture performed for a total of 100 gestures. Also, the final product will have text-to-speech and speech-to-text capabilities.

There are several constraints to the project. For one, some gestures are not arm-based or hand-based, which means that they cannot be reliably measured by the EMG sensors on the Myo armband. This is due to the linguistic nuances of ASL. For example, in ASL, pronouns are expressed through pointing a finger. Even with a sensing device, there is no way to determine if a gesture meant for a pronoun is “I,” “you,” “he,” “she,” or “they.” It is up to the signers to determine this for themselves. In addition, there are no pre-made datasets involving the Myo armband that can be used to train the neural network. Therefore, the dataset must be produced by the experimenter. Finally, as a consequence of software dependencies, a wireless internet connection will be necessary to access Google’s text-to-speech and speech-to-text APIs.

# Project Approach

## Data Visualization & Acquisition

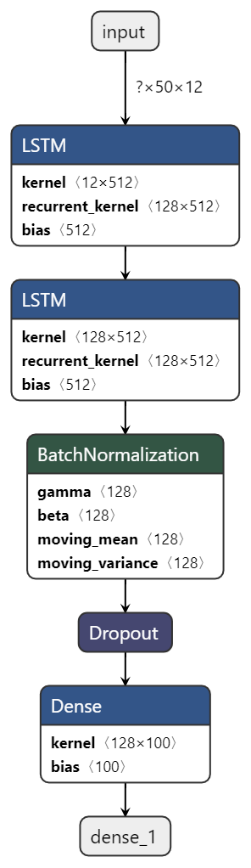
The job of data is to capture desired qualities while the job of the neural network is to make sense of them. The Myo Armband will be used here. The Myo Armband acquires EMG readings from 8 different sensor modules and transmits it to the host computer via Bluetooth. The sensor modules are also equipped with an IMU (inertial measurement unit), which the Myo Armband accounts for when calculating quaternions. The Myo Armband outputs EMG readings for gauging muscle activation and quaternions for representing orientation in 3D space. EMG readings recorded by the Myo Armband are pre-processed in that they are normalized to a range of values between -128 and 127. Since EMG can vary from person to person, the normalization of the raw EMG signals will help to center all readings to a common scale.

The Myo Python library will be used to scrape information from the Myo Armband, particularly through the hub instance. The hub instance establishes Bluetooth communication with the Myo Armband.

The key to formatting the sensor data for use in a neural network lies in stacking readings element-wise and storing them in a master file where the neural network can process them all at once. To represent them properly, all readings captured at a specific point in time must be altogether contained. The data points must correspond chronologically.

## Neural Network Development

The stacked LSTM (long-, short-term memory) model was used. Figure 1 illustrates the full neural network architecture.



**Figure 1. The full neural network architecture. It includes two stacked LSTMs, followed by a batch normalization and dropout layer, respectively.**

The stacked LSTM architecture was used due to the merits discussed in [4]. The specialty of the LSTM model is memorizing long-term dependencies or patterns in data. Hermans and Schrauwen delve into the design of an RNN (recurrent neural network), particularly the logic behind it: “For RNNs, the primary function of the layers is to introduce memory, not hierarchical processing” [4]. Hierarchical processing in neural networks is the idea that new knowledge is extracted at each layer. Hermans and Schrauwen believe that hierarchical processing is not the aim of an RNN. They also believe that shallow hierarchical processing can limit an RNN’s performance. Hermans and Schrauwen make the following assertions: that the RNN does not excel in hierarchical processing, which can affect its understanding of the input, and that the RNN is not by default capable of analyzing data piece-wise with respect to time [4].

In turn, Hermans and Schrauwen proposed a fusion of the DNN (deep neural network) and RNN. Dense layers will be used to increase hierarchical processing, while RNNs will be stacked to give temporal context to the input data at different instances of time [4]. Due to the benefits of increased hierarchical processing and temporal context, the paper has adopted the Hermans and Schrauwen architecture. Therefore, several regularization methods were used to maximize the neural network’s performance: dropout and batch normalization (called “batchnorm”). An often-used form of regularization, the dropout method is a method that randomly disables nodes in a layer (or layers) of a neural network. The logic behind the dropout layer is to force layers to be more self-reliant and infer the missing information themselves [5]. For batchnorm, Ilyas et al. state that the reason why batch normalization is so beneficial is because of its reparameterization of “the underlying optimization problem” being solved by the neural network during training [6]. One might ask: what is the optimization problem that the neural network is solving? The answer lies in the gradient of the cost function. Recall that the goal of training a neural network is to reduce errors in predictions, which means developing weights that navigate the cost function towards its global minima. Ilyas et al. conclude that batch normalization makes the gradient “more predictive and well-behaved” [6].

## GUI Development

Figure 2 illustrates the final layout of the Tkinter GUI, written in the Python programming language.

Status Message Box

Speech-to-Text Listbox

All STT outputs are documented here, each having a timestamp to indicate when an output is produced.

Gesture

Listbox

“Speak”

Button

“Stop”

Button

“Listen”

Button

**Figure 2. Finalized GUI Design. The layout of the GUI, detailing positioning, and type of visual element. The list-boxes keep track of all outputs produced. The “Speak” button interfaces with the Myo Armband for data acquisition, neural network for gesture classification, and Google’s Text-to-Speech API for voicing the output. The “Listen” button executes the script for speech-to-text conversion. The “Stop” button stops all running programs. The Status Message box lets users know what the GUI is currently doing.**

In Python, making GUIs that call long-running processes through button clicks is not as straightforward as it seems. This is because Tkinter GUIs are not capable of performing asynchronous tasks without outside help. If the GUI does not regularly receive the request to update itself, it will become unresponsive. To allow my GUI to call a long-running process, I decided to use the multiprocessing library in conjunction with implementing a queue where outputs from long-running processes are polled periodically. The multiprocessing library will allow my GUI to bypass Python’s global interpreter lock (GIL) and spawn concurrent processes. Recall that the GIL prevents Python from running multiple threads at once. This is effectively dealt with by spawning sub-processes instead. The output of the subprocess is appended to the queue, where it can be fetched. This approach is successful because the GUI is not running the subprocess inside the loop where it is supposed to update UI elements.

# Results and Discussion

Table 1 splits up all 100 gestures in 3 categories: first 3 attempts, between 5 to 10 attempts, and beyond 10 attempts. Each category describes the number of attempts that were needed to get a correct translation of a gesture. Gestures belonging in the first category were gestures which the neural network was easily able to recognize. Most if not all the gestures in the category were recognized on the first attempt. Meanwhile, gestures belonging in the second category were gestures which the neural network was capable of recognizing, but with a little more difficulty. On the other hand, gestures belonging in the final category were gestures which the neural network consistently mis-predicted.

**Table 1. Assessment of Performance through Measuring Number of Translation Attempts**

|  |  |  |
| --- | --- | --- |
| **First 3 Attempts** | **Between 5 to 10 Attempts** | **Beyond 10 Attempts** |
| Sister, brother, aunt, uncle, separate, day, hot, cold, hamburger, egg, fork, hungry, socks, shoes, coat, hurt, bathroom, sleep, sad, sorry, bad, excuse, help, who, where, big, tall, full, more, blue, yellow, red, brown, orange, cost, bird, horse, sheep, pig, bug | Gold, silver, drink, spoon, angry, single, home, work, school, store, church, come, go, in, out, with, night, week, year, today, finish, hotdog, apple, cheese, cup, cereal, water, candy, cookie, cat | Mom, dad, marriage, grandma, grandpa, baby, pants, underwear, brush, nice, cry, like, good, love, please, thank you, what, when, why, how, stop, green, dollars, dog, cow, children, future, here, milk |

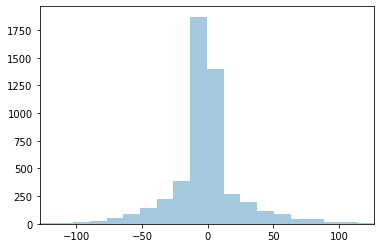
The accuracy of Sign AI in determining a gesture on its first try was also tested. The test was done a total of 5 times, with each gesture performed just once. The average number of gestures that Sign AI could guess correctly was 49. The results for each trial are recorded in Table 2. As expected, while performing the test, most of the gestures that were guessed correctly on the first try were the gestures that belonged in the first category of Table 1. In most of the trials, these gestures were guessed correctly by the neural network. On the other hand, there were times when the neural network guessed gestures in categories 2 and 3 (albeit, with lesser frequency, especially for gestures in category 3).

**Table 2. Evaluation of the Accuracy of Gesture Detection in First Attempt**

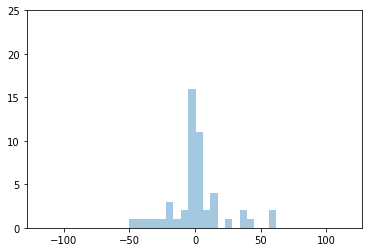
|  |  |
| --- | --- |
| Trial Number | Correctly-Guessed Gestures (Out of 100) |
| 1 | 46 |
| 2 | 55 |
| 3 | 44 |
| 4 | 53 |
| 5 | 48 |
| Average | **49** |

There are potential reasons why the real-time performance of the translator pales in comparison to the performance projected in Keras. One main reason lies in the inevitable variations in EMG readings. In theory, EMG readings can be used to measure the activation of muscle from an electrical domain. However, certain factors throw off EMG readings, especially non-invasive EMG like surface: “The amount of the tissue between contracting muscles and electrodes, along with their thickness, affects the amplitude of the EMG signal” [7]. Also, muscular diseases can lead to abnormal EMG readings, as can slight changes in skin conductance (which can be caused even by changes in one’s mood) [8], [9]. There is no catch-all when it comes to EMG. Such is likely the reason why there is a performance gap between the performance projected by Keras and real-time performance.

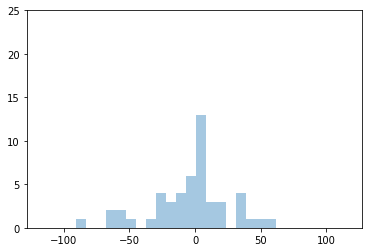
Statistical analysis was done on the distributions of EMG readings measured when certain gestures (particularly, the “marriage” gesture) were performed. All 100 training samples of the “marriage” gesture were recorded in a 20-bin histogram to find a typical distribution. The resulting histogram can be viewed in Figure 3. For comparison, Figure 4 depicts one random sample of the “marriage” gesture in the training dataset. The similarities between Figures 3 and 4 are apparent, as the distributions for both histograms are concentrated in the specific bins. This distribution makes sense as the hands are in a mostly relaxed position until it is time to squeeze one hand around the other. In contrast, Figures 5 and 6 depict EMG distributions that are unlike Figures 2 and 3.



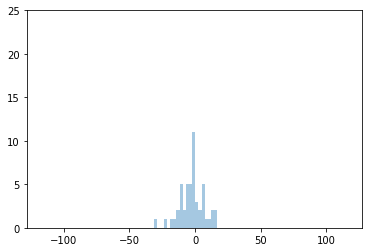
**Figure 3. Overall distribution of EMG readings of training samples for “marriage” gestures**



**Figure 4. Distribution of random training sample of “marriage” gesture (right prediction)**



**Figure 5. “Marriage” performed with increased force (wrong prediction)**



**Figure 6. “Marriage” performed after exercise (wrong prediction)**

Based on the overall distribution depicted in Figure 3, there is a “common case” sample that the neural network was trained with. This “common case” sample is akin to the distribution in Figure 4. It is apparent that most data points in the sample space were clustered within a specific region of bins. As the distribution leaves these bins, the number of instances get smaller. This could mean that outliers in EMG were not accounted for very much in the sample space, which rings true as most if not all training samples for each gesture were taken at one point in time where the specimen’s general condition was the same throughout. Figure 5 was collected from a gesture performed with more force (reflected in the wider distribution), and it was predicted incorrectly by the neural network. Meanwhile, Figure 6 was collected from a gesture performed after exercise, and it was also predicted incorrectly by the neural network. Although the circumstances behind when each gesture was performed were different, the baseline was that – in each circumstance – skin conductance was either increased or decreased. For Figure 5, skin conductance increased due to an increase in action potential. For Figure 6, sweat prevented sensors from making direct contact with the skin, thus dampening EMG readings [10]. In short, there is some evidence to the fact that variations in surface EMG – along with the lack of diversity in the sample space – could have affected the neural network’s performance.

# References

[1] J. Fileccia, “Sensitive care for the deaf: a cultural challenge,” *Creat. Nurs.*, vol. 17, no. 4, pp. 174–179, 2011.

[2] F. G. Bowe, B. T. McMahon, T. Chang, and I. Louvi, “Workplace discrimination, deafness and hearing impairment: The national EEOC ADA research project,” *Work*, vol. 25, no. 1, pp. 19–25, 2005.

[3] C. Lou Garberoglio, S. Cawthon, and M. Bond, “Deaf People and Employment in the United States,” *Washington, DC US Dep. Educ. Off. Spec. Educ. Programs, Natl. Deaf Cent. Postsecond. Outcomes*, 2016.

[4] M. Hermans and B. Schrauwen, “Training and Analysing Deep Recurrent Neural Networks,” in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 190–198.

[5] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[6] S. Santurkar, D. Tsipras, A. Ilyas, and A. Madry, “How does batch normalization help optimization?,” in *Advances in Neural Information Processing Systems*, 2018, pp. 2483–2493.

[7] R. H. Chowdhury, M. B. I. Reaz, M. A. B. M. Ali, A. A. A. Bakar, K. Chellappan, and T. G. Chang, “Surface electromyography signal processing and classification techniques,” *Sensors (Basel).*, vol. 13, no. 9, pp. 12431–12466, Sep. 2013, doi: 10.3390/s130912431.

[8] C. Larivière, D. Gagnon, and P. Loisel, “The comparison of trunk muscles EMG activation between subjects with and without chronic low back pain during flexion–extension and lateral bending tasks,” *J. Electromyogr. Kinesiol.*, vol. 10, no. 2, pp. 79–91, 2000.

[9] E. Gatti, E. Calzolari, E. Maggioni, and M. Obrist, “Emotional ratings and skin conductance response to visual, auditory and haptic stimuli,” *Sci. data*, vol. 5, p. 180120, 2018.

[10] M. Abdoli-Eramaki, C. Damecour, J. Christenson, and J. Stevenson, “The effect of perspiration on the sEMG amplitude and power spectrum.,” *J. Electromyogr. Kinesiol. Off. J. Int. Soc. Electrophysiol. Kinesiol.*, vol. 22, no. 6, pp. 908–913, Dec. 2012, doi: 10.1016/j.jelekin.2012.04.009.