SCHOOL *of* BUSINESS AND TECHNOLOGY
Department of Engineering and Aviation Sciences

Design of Sign AI

A Smart ASL Translator

Josheb Dayrit

Advisor: Dr. Zhang

Spring 2020

Contents

1. Introduction

The following project seeks to develop an ASL (American Sign Language) translation tool for deaf persons. It will be hosted on a microcomputer, like the 32-bit Raspberry Pi. A deep learning model will be trained to recognize ASL through pre-processed EMG (electromyography) data and orientation data in the form of quaternions, all acquired by the Myo armband by Thalmic Labs. Each part in the development process is crucial – doubly so for the deep learning model, which is in charge of gesture classification.

As far as hardware is concerned, an LCD (liquid crystal display) will be used to print all speech outputs. After all, communication is a two-way street. Interfacing with the LCD is a speech-to-text software, which passes data for the LCD to display. When used in conjunction with the neural network, the LCD presents users with the missing part: understanding of speech. The end product, tailor-made for the deaf population, is expected to offer a "sign" and "listen" functionality, whereby deaf users can freely switch between signing and listening through a mixture of software and hardware.

For Sign AI, ease of use as well as translation accuracy are of utmost importance. Ease of use is incorporated in the design process through accessories which increase portability and convenience by virtue of being small, little, and therefore easy-to-carry. Meanwhile, translation accuracy hinges on the deep learning model and its ability to differentiate between different types of EMG data. In the end, what Sign AI aims to offer is a reliable ASL translator that creates a user-friendly experience by automating the translation process, which is gesture-by-gesture. Everything is done in the backend and little input is required from the user.

1.1 Aims

Different means of communication have existed since time immemorial. While speech stands as the most common mode of communication, people communicate through writing as well. Unique modes of communication, serving their own specialized purposes, are as plenty as they are ubiquitous. In technology, computers use a scheme of 0s and 1s to perform complicated tasks; in nature, all scores of animals are able to encode meaning in the noises they produce. For modern humans, speech has been intrinsic to communication. However, for ancient humans, it is different. They used a primitive language based around meaning-filled gestures. In a paper titled "The Origin of Human Multi-Model Communication," two researchers (Stephen Levinson and Judith Holler) from the Max Planck Institute for Psycholinguistics and Donders Centre for Brain, Cognition and Behavior stated that "the modern human communication system is, on a biological time scale, a recent innovation" [1]. Citing the Gestural Theory of Language Origin, Levinson and Holler have much reason to believe that the initial humans (the "cavemen," or Homo troglodytes) initially employed a gesture-based language.

In a society that communicates through speech, it will be difficult for deaf persons to interact with their non-deaf peers. According to the Gallaudet Research Institute, in the year 2006, "nearly 10,000,000 persons (Americans) are hard of hearing and close to 1,000,000 are functionally deaf" [2]. This is also corroborated by more recent 2012 statistics from the Center of Disease Control and Prevention (CDC), which state that "approximately 15% of American adults (37.5 million) aged 18 and over report some trouble hearing" [3]. While the CDC clarifies that "the overall annual prevalence of hearing loss (has) dropped slightly" [3] with the passing of time (from 16% in 1999 to 14% in 2012), additional findings from the National Deaf Center on Post-

Secondary Outcomes reveal an alarming truth about deafness: “In 2014, only 48% of deaf people were employed, compared to 72% of hearing people” [4].

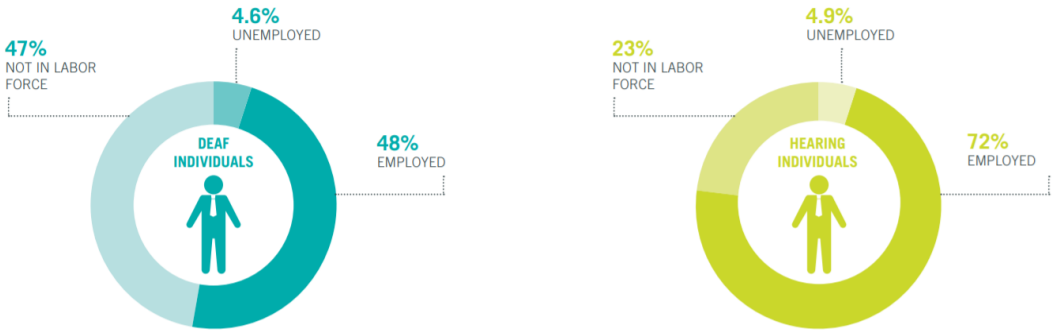


Figure 1: Rates of employment in 2014, according to the National Deaf Center on Post-Secondary Outcomes

Unfortunately, disparities in opportunities and (to some extent) outcomes are due in large part to physical disabilities, as shown by the countless statistics on deaf people. Since deaf people are not capable of speech, they choose to communicate through gesture-based languages, such as American Sign Language (ASL). Because of this, however, prospects for work are limited for deaf people. The harsh reality is that many employers are unwilling to hire deaf employees. While the ADA (American Discrimination Act) outlaws discriminatory practices based on race, gender, or disability, it does not prevent them entirely. In an American study which closely scrutinized successful discrimination lawsuits made against companies, the EEOC (Equal Employment Opportunity Commission) concluded that “young people who are deaf or hard of hearing may be advised to anticipate some resistance from employers” [5].

As outrageous as it sounds, even if a deaf candidate applies for a position while possessing qualifications that meet or exceed a company’s standards, the factor that determines their employment is whether or not they require excessive accommodation. From a corporate perspective, hiring employees who will only place undue burden on a company and its resources

is generally not worth the cost. This idea also aligns with the EEOC's secondary conclusion, which states that disabled applicants who "articulate well their qualifications" and provide long-term work-arounds or solutions to their disabilities are "more likely than others to find success in the world of work."

One solution that Sign AI brings to the table is the function of a basic, real-time, AI-based ASL translation tool. The inability to speak properly is one disadvantage of deaf persons that either prevents them from fully pursuing specific career paths or leads them to be unjustifiably discriminated against in their search for employment. This inability stems from the fact that speech is a product of both the ears and the mouth; the mouth produces the sounds, while the ears regulate them. Without coordination from both the mouth and ears, it becomes increasingly difficult to produce intelligible speech. Many non-deaf employers simply do not have the time to learn sign language for their deaf applicants. In the hiring process, most are likely to turn their attention to the non-deaf candidates, even if the deaf candidates have the necessary qualifications. Such discrimination is, by all accounts, unfair. However, there is some wisdom to be found in it, even if morally questionable. All businesses are utilitarian in that their goal is to maximize productivity while minimizing expenditure of any sort (human, financial, etc.). If a prospective employee demands more from a company than their fellow competitors, that company will likely settle for those competitors in hopes of making things easier on themselves. While Sign AI cannot shift this mindset, it can address the most prominent disadvantage encumbering deaf persons when competing with their non-deaf peers.

As stated before, while deaf people face many barriers of entry regarding employment, communication is by far the most pressing issue. In the workplace, collaboration among employees is important. Typically, company projects are completed not by a single person, but by teams of

people making collective contributions. A pre-requisite to effective collaboration is effective communication, especially in the nascent stages of a project, during which first impressions are gathered, and ideas are put forth for evaluation by others. For deaf persons, communication frequently comes in the form of sign language. Deaf persons have a hard time expressing their ideas to non-deaf persons, because sign language is fundamentally different from spoken language. To bridge the gap between sign language and spoken language, a linguistic interface is needed between the two languages to translate one to the other. Sign AI strives to fulfill this role through a trained AI, a text-to-speech feature that will speak out the meaning of various ASL gestures, and a speech-to-text feature that will convert spoken word back to text.

1.2 Design Requirements

1. The final product is a gesture-based translator, providing translation for each gesture performed.
2. The final product can translate a total of 100 gestures.
3. The final product can differentiate between gestures with an accuracy of 70-90 percent.
4. The final product will have gesture-to-speech and speech-to-text capabilities.
5. The final product will have an LCD to display speech-to-text outputs.

1.3 Design Constraints

Although design constraints are few in number due to the "hands-off" nature of an AI project, undoubtedly, they do exist. One constraint lies in the ASL language itself, while others in hardware.

1. Some gestures are not arm-based or hand-based, which means that they cannot be measured by muscle sensors. In ASL, pronouns are expressed through a singular gesture: pointing a finger. Even with a sensing device as sensitive as the Myo armband, there is no way to determine if a gesture meant for a pronoun is "I," "you," "he," "she," or "they."
2. Since the neural network will be hosted locally, it must run on a microcomputer that possesses sufficient storage as well as processing power. The libraries required for running a neural network (such as Tensorflow and Keras) are large, reaching gigabytes. Without a middleman like the Anaconda Environment to access such libraries through the cloud, it will not be possible to host a neural network on any microcomputer.

1.4 Design Method

The reliability of a neural network depends on 2 important factors: its architecture and the training data. If a neural network is underperforming, the reason will be related to one or both of the aforementioned factors. In the case of Sign AI, the training data originates from the Myo armband, which, for all intents and purposes, is a glorified sensing device. The Myo armband takes EMG readings from 8 different sensor modules and transmits to the host computer via Bluetooth. The sensor modules are also equipped with inertial measurements units, 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.

Naturally, the start of any deep learning project involves determining a means of acquiring reliable data. While the meaning of "reliable data" is different from project to project, its function remains the same for every project: to capture qualities of *something*. The job of data is to capture desired qualities while the job of the neural network is to make sense of them. Data collection is expected to be a major bottleneck in the development of Sign AI, as it is a time-consuming endeavor. The task of collecting data and the task of training the neural network cannot be pipelined because the neural network takes in the collected data as an input.

Once data is acquired, one must then design a neural network capable of processing collected data. Architectures for neural networks are plenty, and some even specialize in solving niche problems. For example, there exists the recurrent neural network (or RNN, for short), which is frequently used for pattern recognition in text and speech. In solving image-based problems, the convolutional neural network was developed to fulfill the need. The design of the neural network should be congruent with the type(s) of data it will be handling.

2. Project Description

2.1 System Description

The Sign AI has 3 main parts: the neural network, the Myo armband, and the hardware for the communication features. The neural network is the crux of the system, as it is, for all intents and purposes, the translator itself. The neural network forms a unit and works in tandem with the Myo armband, where Myo is responsible data acquisition while the neural network data classification. As stated before, Sign AI targets the needs of deaf persons. It will not only translate gestures, but also establish a method for allowing deaf users to understand spoken word.

3. Implementation Plan

3.1 Tasks

Task 1: Data Acquisition and Data Processing

Subtask 1.1: Recruit test subjects for project.

Subtask 1.2: Interface microcomputer and sensing devices; then, collect data.

Subtask 1.3: Write code to automatically document all sensor outputs.

Subtask 1.4: Write code to automatically filter undesired data.

Subtask 1.5: Write code to automatically format data for use in a neural network.

Task 2: Neural Network Design

Subtask 2.1: Propose a data visualization for sensor data.

Subtask 2.2: Determine a suitable neural network architecture for the project.

Subtask 2.3: Verify the feasibility of proposed data visualization method and neural network architecture.

Task 3: Neural Network Implementation

Subtask 3.1: Create a TensorFlow model based on the results of Task 1.

Subtask 3.2: Train TensorFlow model using acquired sensor data.

Subtask 3.3: Refine TensorFlow model to optimize accuracy.

Subtask 3.4: Export TensorFlow model for real-time use.

Subtask 3.5: Develop a script for facilitating data flow to TensorFlow model.

Subtask 3.6: Test finalized TensorFlow model in real time.

Task 4: Design of Gesture-to-Speech Feature

Subtask 4.1: Determine required hardware and software.

Subtask 4.2: Research part specifications.

Subtask 4.3: Interface the software with the hardware.

Subtask 4.4: Assemble the required hardware.

Task 5: Design of Speech-to-Text Feature

Subtask 5.1: Determine required hardware and software.

Subtask 5.2: Research part specifications.

Subtask 5.3: Interface the software with the hardware.

Subtask 5.4: Assemble the required hardware.

Task 6: System Assembly, Testing, and Refinement

3.2 Timeline/Milestones/Delivery Plan

Time	Task	Comments	Responsible Personnel
Week 1-5	Project Planning		Josheb Dayrit
Week 6-7	Subtask 2.1 & 2.2	Data visualization method and neural network architecture.	Josheb Dayrit
Week 8-11	Subtask 2.3, Subtask 4.1 & 4.2, Subtask 5.1 & 5.2	Are the proposed data visualization method and neural network architecture viable for use in project?	Josheb Dayrit
Week 12-15	Subtask 1.1, 1.2, 1.3, 1.4, & 1.5	Collection and processing of sensor data.	Josheb Dayrit
Week 16-19	Subtask 3.1, 3.2, 3.3, 3.4, 3.5, 3.6	Deployment of neural network on microcomputer.	Josheb Dayrit
Week 20-22	Subtask 4.3 & 4.4, Subtask 5.3 & 5.4, Task 6	Putting all the pieces together.	Josheb Dayrit

4. Implementation

4.1 Implementation of Task 1

4.1.1. Implementation of Subtask 1.2

4.1.1.1 *Myo Armband*

The Myo armband is the sensing device that will be used for data acquisition. It was developed by Thalmic Labs and released for consumer use on July 2014. However, it was later pulled off the shelves on October 2018, with developer Thalmic Labs citing that another project was in the works. The intent behind the Myo armband was to provide a wireless control for computers, laptops, and phones through implementation of EMG sensors and IMUs. It has also been used in academia for other purposes, such as gesture classification.

On the inside, the Myo armband contains an Invensense MPU-9150, which is a 9-axis IMU (inertial measurement unit), and 8 different EMG electrode units, which supplies 8 different EMG readings from 8 different locations in the forearm. EMG readings are pre-processed using an in-house algorithm developed by Thalmic Labs. While Thalmic Labs has made it possible to acquire raw EMG readings from Myo, their EMG algorithm still has not been disclosed to the public. In addition, Myo calculates orientation data in the form of quaternions. The 9-axis IMU processes the kinetic information necessary for these calculations. The IMU chip lies on the back of the motherboard, alongside other peripheral connections. The processor used by the Myo armband is the MK22FN1M0, which is a Freescale Kinetis Cortex M4 processor with clock speed of 120MHz. A Bluetooth module (NRF51822) can also be found close to the processor for wireless transmission and reception.

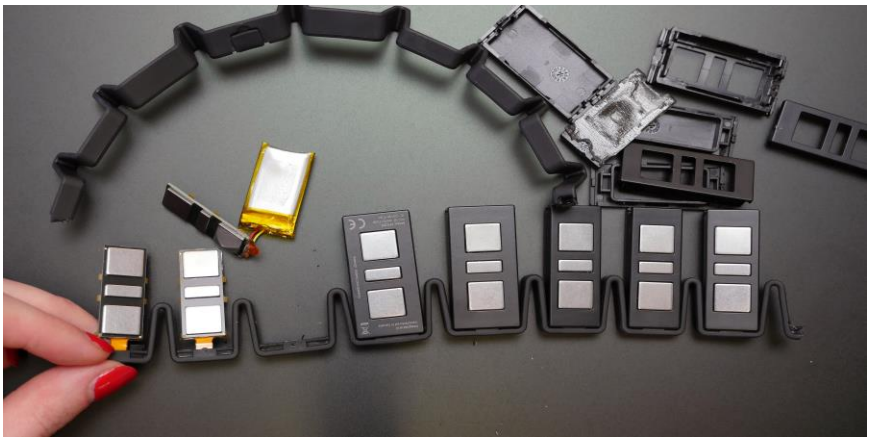


Figure: EMG sensor modules in Myo armband

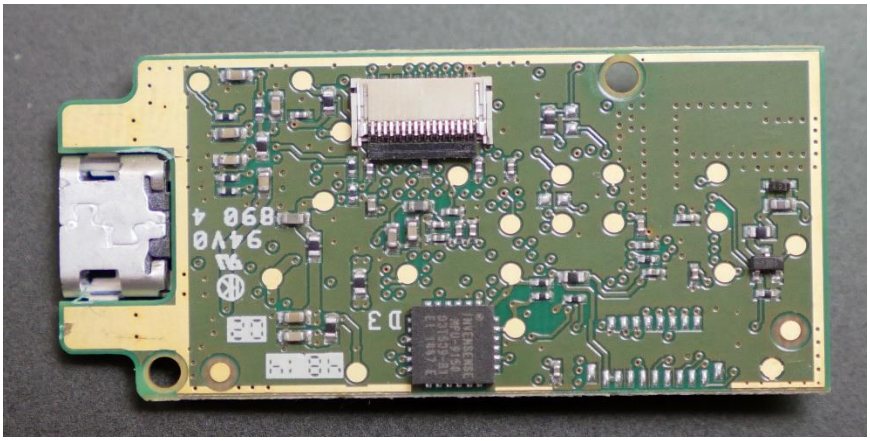


Figure: Invensense MPU-9150 on the back of the motherboard

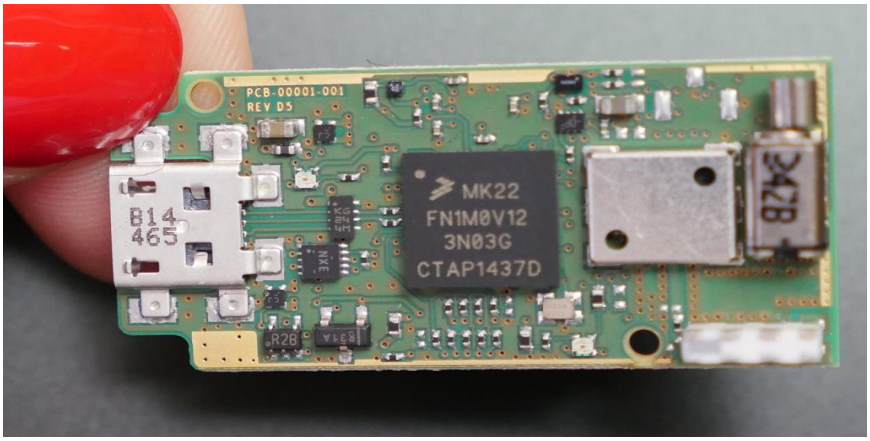


Figure: Processor for Myo armband

The Myo armband was the sensor of choice for various reasons. First, it was readily available, provided by the Department of Engineering and Aviation Sciences. Thus, it did not cost a thing. Second, it is more than capable of performing human activity recognition on the gesture scale. In a study done by the University of Western Switzerland, the Myo armband was compared side-by-size with more sophisticated EMG sensors. The sensors tested were the Otto Bock 13 E200, the Delsys Trigno Wireless, the Cometa miniWave, and the Myo armband from Thalmic Labs.

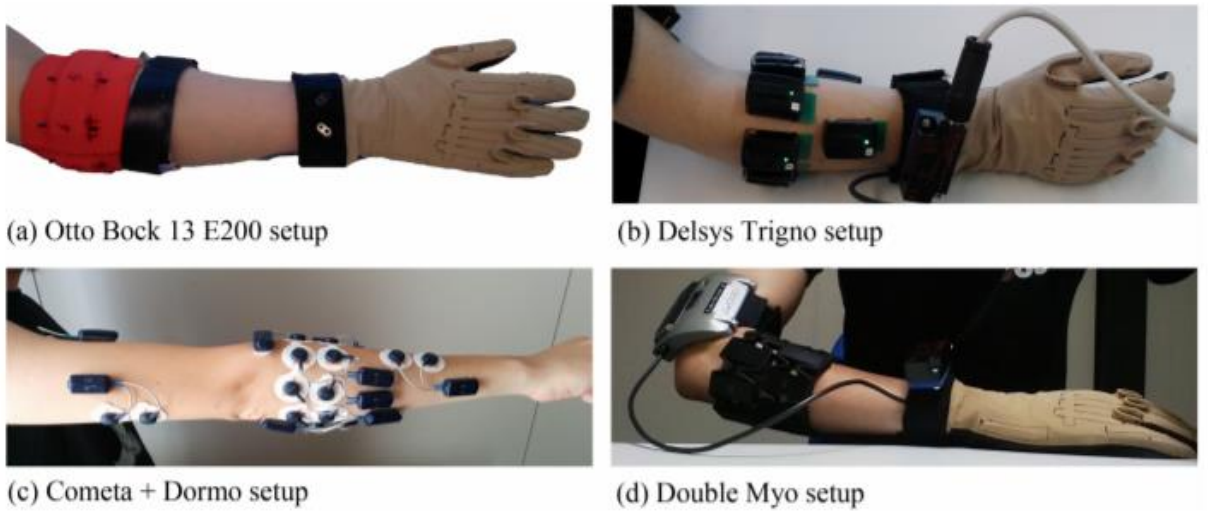


Figure: Setup for each sensor.

All neural networks regardless of architecture are data-hungry, with some hungrier than others. The neural network requires large pools of data from which to extract features in a process appropriately named “feature extraction,” which is the process where data is “interpreted” by a neural network. In the previously-mentioned study, the Switzerland-born researchers attempted to develop a neural network for recognizing 41 hand movements. They acquire data from different EMG instruments (Otto Bock 13 E200, Delsys Trigno Wireless, Cometa miniWave, and Myo armband) to train their machine learning model. The performance of the model was evaluated for each EMG device.

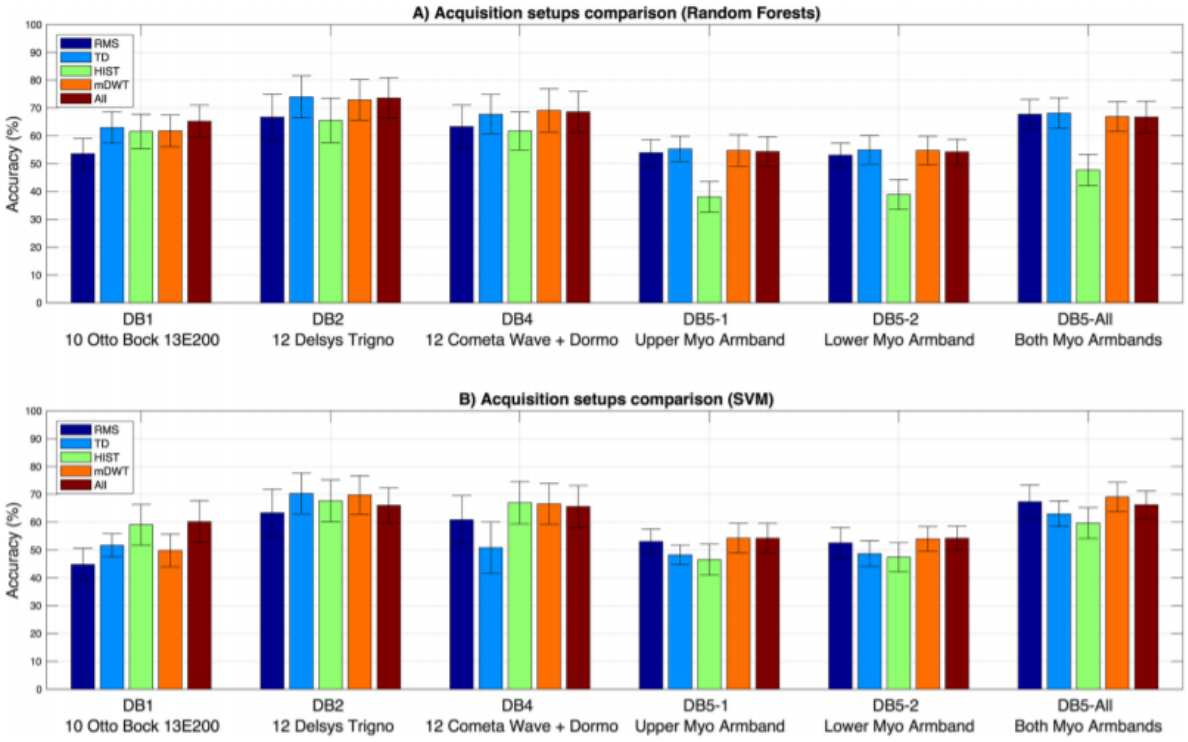


Figure: Accuracies of 2 models for each EMG device

2 machines learning models were developed and evaluated in the study. The first model used the Random Forest classifier while the second model was an SVM (support vector machine) model. The Random Forest classifier adopts the design of a traditional decision tree. A decision tree is a series of yes-or-no choices. A machine learning model that use a decision tree for classification tries to divide up all available data samples into groups. Each group is associated with its own “branch” of yes-or-no choices. These “choices” are qualities which the machine learning model deems each sample group to have. The Random Forest classifier is more complex than a decision tree because it contains multiple (and, sometimes, nested) decision trees, leading to deeper understanding of data.

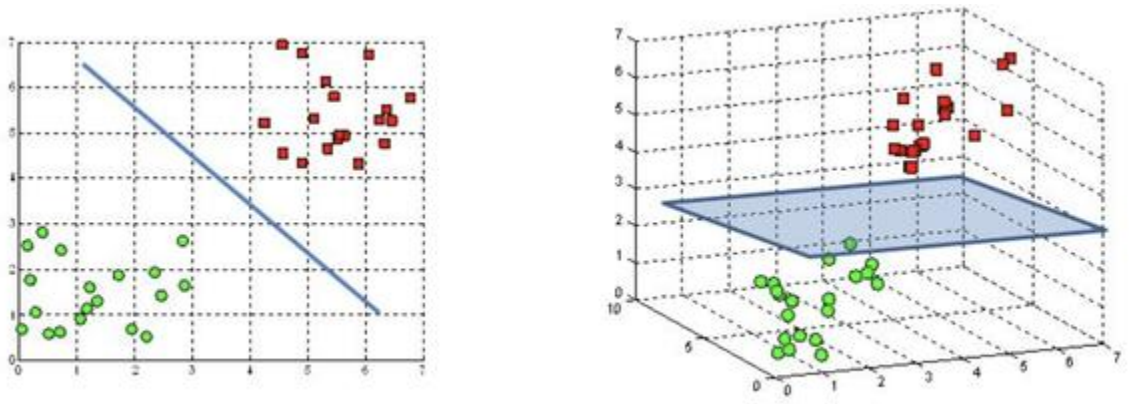


Figure: Hyperplanes in an SVM model

Meanwhile, the SVM model is a machine learning model that relies on the concept of "extrapolation" – making inferences from prior information. Specifically, it attempts to locate data points in close proximity of each other through finding a "hyperplane." In simple terms, the hyperplane is a boundary that delineates a clear difference between data regardless of its dimension (1D or 3D). The hyperplane is a line in one-dimensional data or a plane in three-dimensional data. It is asymptotic in nature, defining a range through which data points cannot traverse. By finding a hyperplane, an SVM model is able to pinpoint distinct qualities in different sets of data. This enables it to better classify said data.

The Swiss study concluded that, while the best accuracy was achieved through data gathered by the Delsys Trigno, all EMG devices exhibited near-equivalent results, with differences ranges from small to negligible. The Myo armbands were still able to reach accuracy values which rivaled the Delsys Trigno. Although higher accuracy was achieved by the Delsys Trigno through the Random Forest model, the performance difference between the 2 (Delsys and Myo) becomes downright non-existent with the SVM model.

The purpose of citing and giving an overview of the Swiss study was to justify usage of the Myo armband. Not only is the Myo armband the cheaper option than all other alternatives mentioned in the Swiss study, it managed to provide accurate enough EMG data to allow a machine learning model intended for classification to perform adequately. For reference, the cost of the Delsys Trigno is $35,000 while the cost of the Myo armband is a measly $300 in comparison. When attempting to fabricate a product meant for the general populace, the cost of production must be at a reasonable level. Even if a given sensor is not the *most* optimal for the job, all options must be considered and weighed against each other through different criteria. Is an increase of 1-3 percent in accuracy truly worth $30,000? The answer can depend from project to project. In the case of Sign AI, this increase in accuracy is too small to warrant its cost.

4.1.1.2 ***Data Acquisition via Bluetooth & Myo Python***

There are 2 methods through which a user can view data (EMG, orientation) from the Myo. The first method is through the Myo diagnostic page located at the following link: http://diagnostics.myo.com. The second is through the Myo Python library created by Nikolas Rosenstein. All documentation related to the Myo Python library can be found on Github and PyPi. As the Myo diagnostics page cannot be web-scraped, the second option was chosen instead. Thanks to the Bluetooth module and dongle, allowing Myo to communicate data to other devices is simple and hassle-free. The remaining hurdle is writing a script to continuously acquire said data.

The Myo Python library has a class known as the “Listener” class, which borrows from an already-existing class, the Device Listener from the Myo SDK (Standard Development Kit). The Myo SDK was put together by Thalmic Labs and was initially written in the C language.

4.2 Implementation of Task 2

4.2.1. Implementation of Subtask 1

4.2.1.1 Flaws of Traditional Approaches

How data is represented, in addition to the data itself, has an impact on the performance of a neural network. For projects that perform translation of ASL gestures through a neural network, data is traditionally represented as images of the ASL gestures, transformed into the array counterparts. While this method of visualizing data has produced serviceable outcomes for some ASL projects, it is not a method that is without flaws.

To start, it should be noted that static images can successfully depict certain sets of gestures. For these gestures, the final position of the hand is the distinguishing characteristic. Thus, it can be contained in a single frame. However, this is not the case for gestures where rotation or movement is a key characteristic. These gestures cannot be expressed in a single frame. The figures below show the uncanny similarities between 2 gestures. These gestures are “try” and “rock,” respectively. Although the orientations of both gestures are different, the palms are balled up for both gestures. These overlaps in final hand position will make it hard for the neural network to differentiate between gestures if given that information alone. The reason why most AI-based ASL projects do not venture beyond certain sets of gestures is because the method through which their data is expressed is narrowly focused, suitable for some circumstances but not others. This constraint prevents the neural network from doing translation work on other gestures.

Figure: ASL for “try” (final hand position)

Figure: ASL for “rock” (final hand position)

Another inherent defect of image data is the presence of image noise, which is all-too-often too hard to decouple from the desired parts of an image. In a paper on techniques for image noise reduction, Indian engineering professors Mythili and Kavitha describe 8 types of image noise in great detail: amplifier noise (gaussian noise), salt-and-pepper noise (impulse noise), shot noise, quantization noise (uniform noise), film grain, on-isotropic noise, speckle noise (multiplicative noise), and periodic noise. Although some discussion is devoted to each category of noise, discussions on impulse noise, gaussian noise, and speckle noise are particularly lengthy, as these are the most common types of image noise found in images.

The first type of noise discussed by the paper is amplifier noise (or Gaussian noise), wherein it is stated that amplifier noise is "additive". Mathematically speaking, Mythili and Kavitha disclose that, in ideal cases, amplifier noise can be quite easily modeled by a Gaussian distribution. Armed with this information, various filters have been developed and implemented by DSP (digital signal processing) researchers in order to combat it. On the fundamental level, amplifier noise is noise that adds a bias to the original red, green, and blue values of an image signal.

In another ResearchGate paper by professors from 2 European universities, Gaussian noise was applied on an image to develop techniques relating to facial recognition. A progression is shown of the image as it is degraded by noise of increasing severity. Based on the differences that can be observed from the image at each stage of the "noising" process, it becomes clear how each pixel is being affected the applied noise. Comparing the original image to its noisier counterparts, one can see how pixel intensity is changed at every location in the image, with some pixels becoming grayer and others less gray. These random fluctuations in pixel intensity can be attributed to the fact that Gaussian noise is "additive," causing parts of an image to be lighter or

darker than they originally are. Such fluctuations are the “biases” mentioned in the paper by Mythili and Kavitha.

Figure: Gaussian noise applied to already-noisy image

Another commonplace noise type that is discussed in Mythili’s and Kavitha’s paper is impulse noise or salt-and-pepper noise, which is similar in some ways to amplifier or Gaussian noise. In Gaussian noise, the image is affected as a whole. However, in impulse noise, a select number of pixels are affected, which make them stand out from the rest. The primary difference between Gaussian noise and impulse noise is that impulse noise is smaller in scale and caused on the hardware side. Hardware mishaps which lead to impulse noise include non-functional analog-to-digital converters in cameras, overheating of equipment, and dust in lenses.

Figure: Salt-and-pepper noise applied to clear image

The figure above makes the differences between impulse noise and amplifier noise rather apparent. Firstly, it is clear to see that the original image does not lose most of its integrity when the impulse noise was applied. There are fewer corrupted pixels, which means that the image is left intact apart from those pixels. By comparison, an image affected by amplifier noise essentially becomes indiscernible.

The final noise type covered in detail by Mythili and Kavitha is speckle noise, which, unlike Gaussian noise, is multiplicative rather than additive. In particular, satellite and radar technologies are said to be more acutely affected by speckle noise, as it is a direct result of interference from the surrounding area. Image-capturing hardware ranging from mundane devices like cameras to more sophisticated contraptions such as the Synthetic Aperture Radar rely on the concept of backscattering. In the case of the Synthetic Aperture Radar, an image is captured through a mixture of steps involving the radar itself and the specimen. First, the radar illuminates the specimen with a light of its own. Then, sensors on the radar pick up the light that is reflected back to it. This back-and-forth interaction between the radar and the specimen is known as backscattering. Once the reflection is received by the radar, it is processed digitally; based on this reflection, the radar reconstructs its version of the specimen.

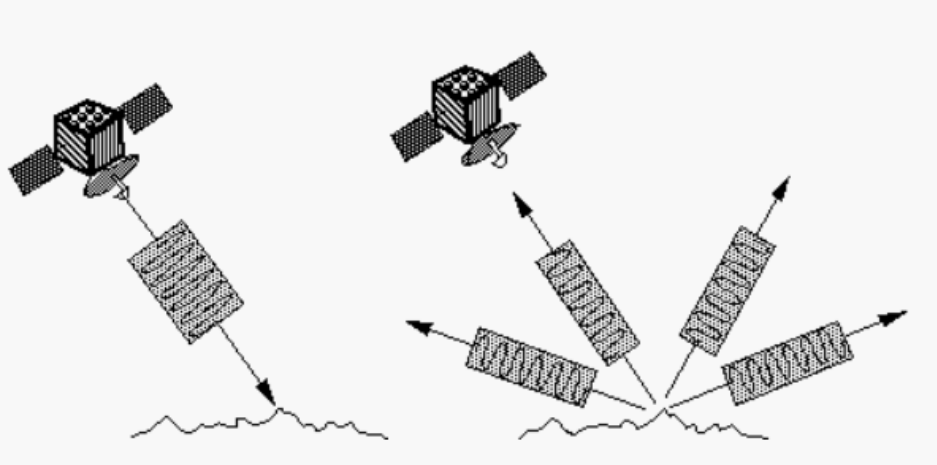


Figure: An imaging radar at work

In the concept of backscattering, the parameter that matters most is the energy that is reflected by the specimen. An article by NASA on radar technology reveals that the energy of the echoed photon signal can determine the brightness of an image that is produced. Specifically, it states that “bright features mean that a large fraction of the radar energy was reflected back to the radar, while dark features imply that very little energy was reflected.” In addition to the energy of the reflected signal, there are also environmental and material-specific factors that can influence the quality of an image. For example, in radar images, specimens which contain more water generally appear brighter than their dry counterparts. This is because water has an effect on an object’s electrical properties. When water molecules comes into contact with an object, they cohere with other molecules and form a barrier on top of the skin. The resistance of the skin is reduced because it is connected in parallel to the layer of water. Variance in a specimen’s resistance can also dictate if an object appears brighter or darker in a radar image.

Figure: Water droplets captured in an image

Radars and cameras are similar in that backscattering can affect the quality of images that both produce. For cameras, the negative impacts of backscattering can be particularly observed in images where small obstructions such as droplets of water appear as clear or white imperfections. Whenever there are small particles littered about in the frame of a camera, the chance of producing an image with undesirable affectations increases drastically. In addition to water droplets, fine substances like dust or sand can also produce similar affectations in an image. The factor that influences these affectations is distance. According to the Inverse-Square Law, light is reflected more strongly if it is closer to its source. By virtue of being closer to the camera lens, the water droplets in the figure above appear brighter and more prominent than the rest of the image.

Image noise is the prevailing reason why data from image-capturing devices proves to be unreliable for use in a neural network. There are simply too many factors – some controllable, while others not – that can affect image data from image-capturing devices. It will take a lot of work just to pre-process the data and isolate the desired parts. Even if a researcher decides to train the neural network to be able to recognize noise from the gestures, it will take a large pool of data to tackle such a complicated problem. The complication is due in large part by the various types of noise that can exist in a single image. To be able to recognize them all, a neural network would need an abundant number of samples. Otherwise, the alternative would be to develop numerous robust filters capable of suppressing noise to an acceptable but preferably ideal level.

4.3 Implementation of Task 2

4.3.1. Implementation of Subtask 2

4.3.1.1 The ConvNet (Convolutional Neural Network)

The convolutional neural network (or ConvNet, for short) is a neural network architecture whose specialty lies in the field of image processing. Many ASL projects have used the ConvNet for the translation of ASL gestures, often achieving successful results. The typical approach of such projects is to use images of the ASL language as training material for the neural network.

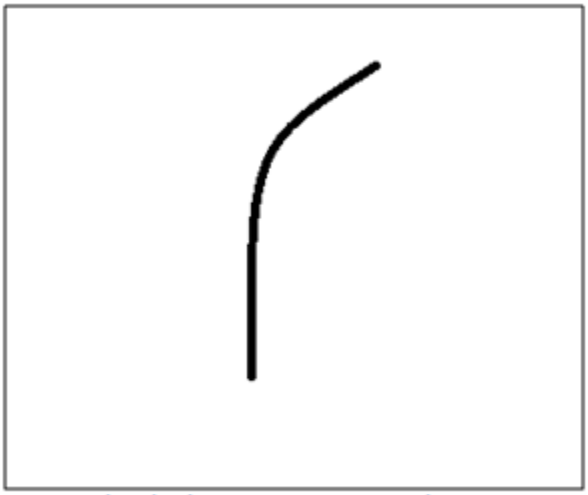
When the human eye sees an image of a cat and a dog, it is able to pick out which image is depicting the cat and which the dog. The brain receives stimulus from sensing organs and interprets said stimulus through a series of chemical reactions. Learned patterns and prior knowledge enable the brain to interpret new information. Unsurprisingly, something similar is done in machine learning models. The neural network is, after all, just a mathematical abstraction of the brain.

The early origins of the ConvNet can be traced to an experiment conducted by a pair of scientists Hubel and Wiesel in 1962. The Hubel and Wiesel experiment involved observations of the visual cortex. Using cats as a test group, Hubel and Wiesel determined that the neural response of a stimulus can vary depending on how it is presented. When the pair shined a light over an empty region of paper, the cats did not respond, and their neurons registered little to no activity. However, when the light was instead shined on a cluster of vertical lines or horizontal lines, the cats displayed significant brain activity. Specific sets of neurons, organized in rows and columns, were activated each time the light was shined on the lines. This study of the visual cortex served as an inspiration for the design of the ConvNet.

The selling point of the ConvNet is its convolutional layer, which is a layer that takes 3D arrays as inputs. When a convolutional layer is defined in Tensorflow or Keras, 3 parameters must be provided: input size, the number of nodes, and the filter size. As an analog, one can think of the convolutional layer as the visual cortex of a ConvNet. The filter is an important aspect of the convolutional layer because it is multiplied element-wise to the input array. It is an array itself, which is composed of parameters called "weights" – numbers that influence and decide the outputs of a layer. The size of a filter is specified by the user, and it cycles through the input array piece-wise like a magnifying glass. The dot product is taken of the filter array and a segment of the input array (called the "receptive field"), and the result is put into another array called the "feature map" – the ConvNet's interpretation of an image, expressed in numerical form (a one-dimensional array). Based on the feature map, the ConvNet is able to pick up on characteristics in an image, like curves or corners.

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter



Visualization of a curve detector filter

Figure: A filter tuned for detecting curves in an image

Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

*

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

Figure: Dot product of the receptive field and filter

In the two figures above, an example is given of the mathematics involved in calculating an element of the feature map from the filters and receptive fields. The dot product of the two arrays are taken, which results in an element for the feature map. The produced element is related to the filter because it determines if a desired attribute (i.e. a curve or corner) has been detected in the receptive field. If the number is high, then it is probable that the receptive field contains the desired attribute. Through the feature map, the ConvNet is able to extract characteristics from the input array, which it can use for purposes of classification or prediction.

In an ASL project conducted by a computer scientist (Vivek Bheda) and a linguist (Dianna Radpour) from the University of New York, the pair attempted to use the ConvNet for classifying the most basic ASL hand signs, which are the letter and number hand signs. In total, 36 hand signs were processed by the ConvNet, consisting of letters from A to Z and numbers from 0 to 10. Supervised learning is the name of the game for dozens of machine-learning models, including the model designed by Bheda and Radpour. It is a kind of learning where the inputs and outputs are pre-established. In supervised learning, the task of the neural network is to engineer a way of mapping inputs to the correct outputs.

In terms of design, the model developed by the team of Bheda and Radpour adheres to the conventional structure of a ConvNet, which is a conglomeration of convolutional layers, dense (or fully connected) layers, and other supplementary layers used for minimizing computational load. The layer that differentiates the ConvNet from neural network architectures similar to it is the convolutional layer, a special layer that processes an image piecewise rather than as a whole. Using the standard ConvNet architecture, Bheda and Radpour were able to achieve accuracies ranging from 80 to 90 percent, which meant that their ConvNet only failed to distinguish a gesture 10 to 20 percent of the time, depending on the gesture.

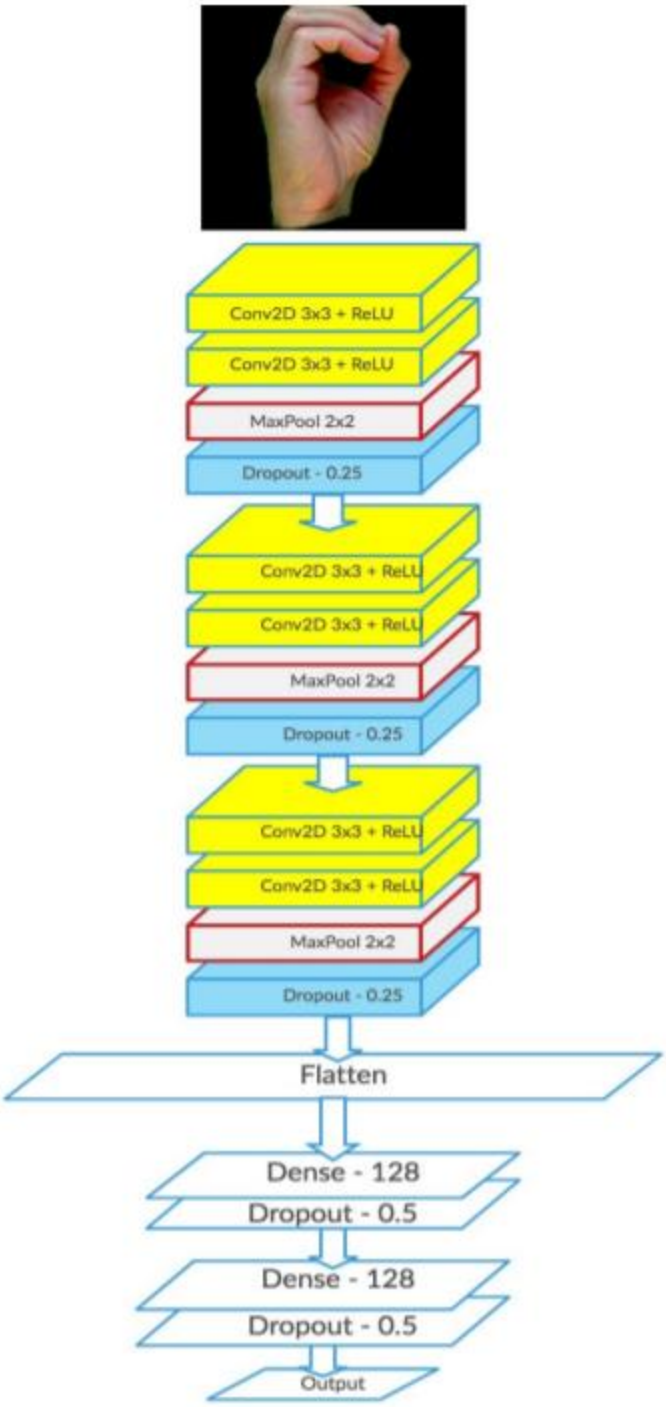


Figure: Design of Bheda and Radpour's ConvNet

4.3.1.2 *LSTM (Long-Short Term Memory) Model*

The LSTM model falls under the umbrella of the recurrent neural network (RNN). For context, the RNN is a type of neural network that excel in processing sequences of data. It is able to do this through transmitting prior information to later parts of a neural network. Most neural networks are feed-forward networks, meaning that information flows in a single direction. Every layer in a feed-forward neural network produces outputs that other layers are not able to access. The other layers are not told how these outputs are produced. This is where the RNN begins to diverge from the traditional feed-forward neural network. The RNN interconnects outputs of layers by passing said outputs from one layer to the next through a loop. Prior information is known as the "hidden state" in the realm of deep learning. The hidden state relays more than just the outputs of a previous layer. It also describes how previous outputs have changed over time while in previous layers. RNNs have layers with feedback loops, and this is where the hidden state is passed on and updated. In an RNN, the hidden state of a previous layer can affect the outputs of the following layers because it is processed in conjunction with the inputs of the following layers.

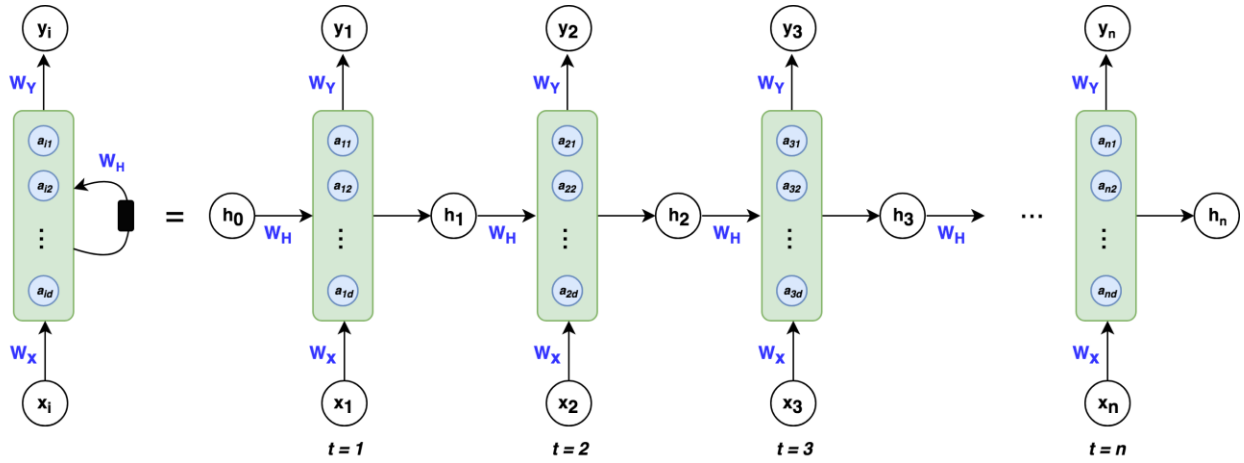


Figure: A simple RNN layer

5. Conclusion