Developing a Mobile Application and Cloud Computing Framework for Gesture Recognition Using Electromyography and Inertial Measurement Unit Data

Kattia Chang-Kam[1], Karina Abad[1], Ricardo Colin[1], Cameron Malloy[1], Charles Tolentino[1]
Xiarong Zhang[2] , Alexander David[2]
[1]Cañada College
[2] School of Engineering, San Francisco State University

Abstract

Electromyography (EMG) signals are a measurement of electrical activity in muscles. These signals contain important neural information representing movement intentions. EMG pattern recognition has been studied and used for myoelectric controlled applications such as neural-controlled prostheses, assistive robots, and virtual input devices. EMG signals are usually processed and analyzed using feature extraction and pattern recognition methods to interpret human movement. However, to apply myoelectric controlled systems in practice, some challenges still remain. The system needs to be portable, real-time, and robust. Moreover, a large amount of data needs to be obtained and stored in order to improve the system. The goal of our research was to develop an Android mobile application that is able to process incoming EMG data and output a gesture classification. Additionally, a cloud computing framework, using Amazon Web Services, was created in order to store and process that information. The application was written in Java and receives data through Bluetooth low-energy (BLE) communication from a Myo Armband (Thalmic Labs). This device is low cost and easy to use, which aligns with our project's purpose. The armband streams raw EMG data at 200 Hz from 8 different EMG sensors and 9-axis inertial measurement unit (IMU) data at 50 Hz. The incoming data is subjected to classification algorithms and from this we are able to output a gesture decision. This decision is then given to a client application; such as, a prosthetic limb or video game. Experiments were done on human subjects to evaluate the accuracy, response time, and usability of the developed system.

I. Introduction

Gesture recognition provides a convenient and natural way for humans to interact with computers [1]. These interactions include: sign language recognition and gesture-based controls. Gesture recognition technology can be classified into three different methods: data-gloved recognition, computer-vision techniques, and electromyography signals (EMG). Data-gloves use bending sensors and accelerometers. The main disadvantage of this method is that it isn't natural and convenient because the user needs to wear a cumbersome data glove in order to obtain the necessary data input for the system. In contrast, computer-vision doesn't require the user's interference. However, this method is heavily dependent of the camera that is being used. The device is susceptible to the light settings in the environment and that limits how well this technique recognizes gestures. Moreover, it needs a complicated setup that uses several cameras.

An alternative to the two previous methods is the use EMG signals. EMG signals is the electrical activity of our muscles known. This method can be used to control prosthetic arms. There are important considerations that must be taken into account in order for the user to achieve high acceptability, when using a prosthetic arm with this control scheme [2]. First, the system needs to be accurate meaning that the gesture performed needs to be correctly done or represented by the system. Second, the controls need to be intuitive to the user so that he or she can easily and seamlessly learn how to use the controls. Finally, the system should not introduce a considerable amount of delay in order to be considered as real-time gesture classification. An acceptable threshold for the delay is 300ms.

Taking all the above into consideration, the goal of our research was to develop an android application that is able to obtain EMG data, process it, output a gesture classification, and store the processed data in a cloud storage service. The Myo Armband, developed by Thalmic labs, is the device used to stream raw EMG data from muscles to an android device.

Figure 1. Myo Armband developed by Thalmic Labs.

The Myo Armband, shown in Figure 1, was chosen because it is low-cost with a value of $200. Additionally, it is user friendly because the user can wear the armband without any previous preparation. It has 8 EMG sensors that stream data at 200 Hz and 9-axis inertial measurement unit (IMU) that streams data at 50 Hz. The IMU sensors include a three axis gyroscope, magnetometer, and accelerometer.

A basic overview of the structure of our android application is shown in Figure 2 below. The application was written in java using Android studio. Additionally, the Statistical Machine Intelligence and Learning Engine (SMILE) library was used to implement different classification and model validation algorithms. Finally, our cloud application utilizes Amazon Web Services for cloud based storage and computing.

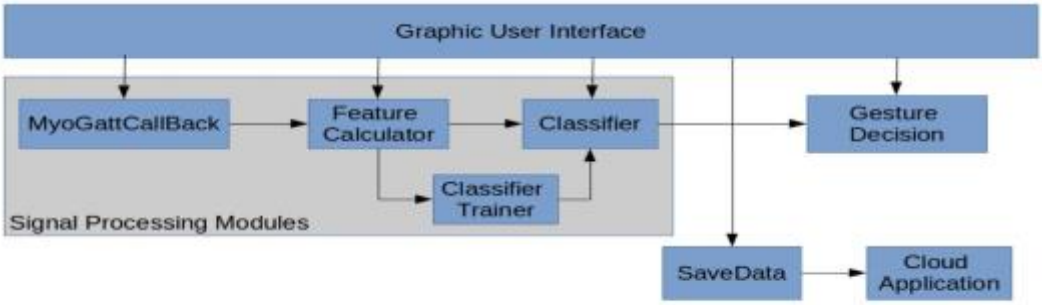

Figure 2. Structure of the android application.

A. Myo Data

The application obtains raw EMG data from the Myo Armband through the use of Bluetooth low energy (BLE). In this state, the data is not useful due to noise from several sources such as the arms muscles. This data must be subjected to data processing; in order to do this, the data is separated into discrete windows for analysis.

B. Feature Calculator

Data Windowing

There are two data windowing schemes: adjacent and overlapped sliding.

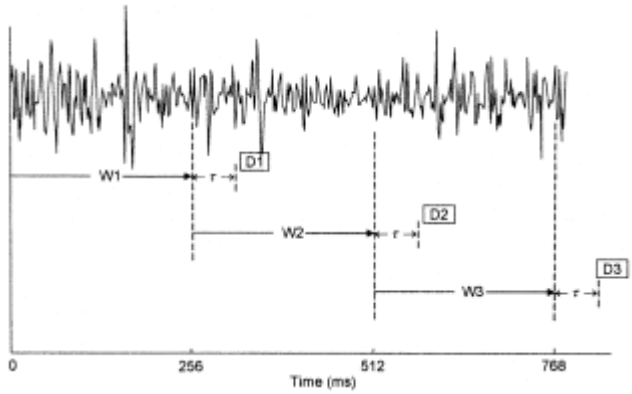

Figure 3. Adjacent window scheme [1]

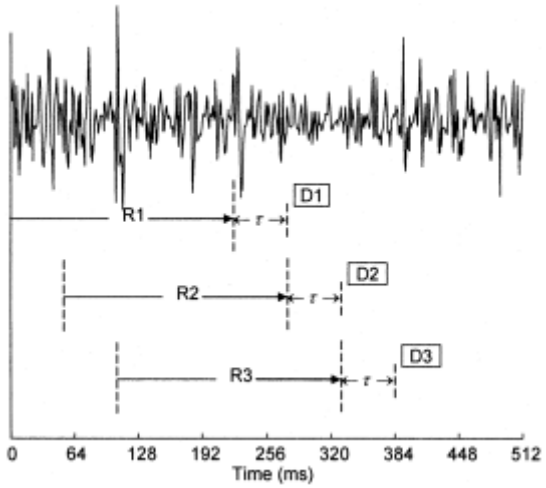

Figure 4. Overlapped sliding window scheme [1].

For adjacent windows, the window size and the window increments have the same length as shown in Figure 3. In contrast, the overlapped sliding window has a smaller window increment compared to the window length, which can be seen in Figure 4. The advantages of using this method allow the application to obtain a denser data set, meaning it can perform analysis on the next window without having to wait for the current one to finish. Ultimately, using overlapping windows takes advantage of the computing power of the device [1].

Feature Extraction

The application extracts the following time-domain features from the EMG data streamed from the Myo Armband: mean absolute value (MAV), waveform length (WAV), number of slope sign changes (Turns), and number of zero crossings (Zeros). Two additional features called scaled mean absolute value (SMAV) and adjacency uniqueness (AC) are also calculated.

Mean absolute value is a commonly used feature that detects muscle contraction levels [3].

$$MAV_i = \frac{1}{n}\sum_{k=1}^{n}|x_k| \ for\ i = 1, \dots, i \quad (1)$$

Waveform length refers to the total length of the waveform in a segment of time [3].

$$WAVE = \sum_{k-1}^{n}|x_k - x_{k-1}| \quad (2)$$

Slope sign changes indicates the amount of times there has been a change from positive to negative or negative to positive in the slope for three consecutive segments in a given threshold denoted by epsilon (ε) [3].

$$(x_k > x_{k-1}\ and\ x_k > x_{k+1})\ or\ (x_k < x_{k-1}\ and\ x_k < x_{k+1})\ and\ |x_k - x_{k+1}| \geq \varepsilon \quad (3)$$

Zero crossings calculates the amount of times the waveform has crossed zero in a given threshold epsilon (ε) [3].

$$(x_k > 0\ and < 0)\ or\ (x_k < 0\ and\ x_{k+1} > 0) and\ |x_k - x_{k-1}| \geq \varepsilon \quad (4)$$

Scaled mean absolute value indicates gesture intensity:

$$SMAV = \frac{MAV_i}{\frac{1}{i}\sum_{i=1}^{8} MAV_i} \quad (5)$$

Adjacency uniqueness detects how distinct adjacent Myo Armband channels are [4]:

$$AU_C = \frac{1}{wl}\sum_{n=1}^{wl}\left|\frac{x_c[n]}{MAV_c} - \frac{x_{c+1}[n]}{MAV_{c+1}}\right| \quad (6)$$

C. Classifier Trainer, Classification Algorithms

The features collected from the feature calculator are compiled into vectors. These vectors can be used in two different phases during the training of our classification algorithm or during the prediction of gestures.

Classification Trainer

Training classifiers require two things, the feature vectors and knowledge of the corresponding gesture, or class, that is associated with that feature vector. This data is used to build a model that helps separate different classes based on similarities and differences within the feature vectors. For instance, a resting gesture that does not activate much electrical activity in the forearm would

produce a very low SMAV whereas a fist gesture would produce a much larger SMAV. When training the classifiers, the model would separate these two gestures based on SMAV.

These models are then used to predict gestures, by comparing incoming feature vectors that need to be predicted with the model built by the classification trainer.

Classification Algorithms

Classification algorithms take in feature vectors, but unlike when training do not take in corresponding class labels, or gestures. The algorithms predict the classes by comparing the feature vectors to the models generated by the classification trainers. In our project, we used four different classification algorithms, Linear Discriminant Analysis (LDA), Logistic Regression, Decision Trees, and K-Nearest Neighbor (KNN).

1. Linear Discriminant Analysis

LDA tries to find the dimension such that when all the training data is projected onto that dimension, it has the maximum difference between the means of the classes, normalized by their variances (insert equation #). LDA reduces the dimensionality of the data in the most optimal way for classification. This groups similar classes together and separates different classes. The figure below represents how LDA works, where the purple line is the most optimal subspace and the red line is the least optimal [1]:

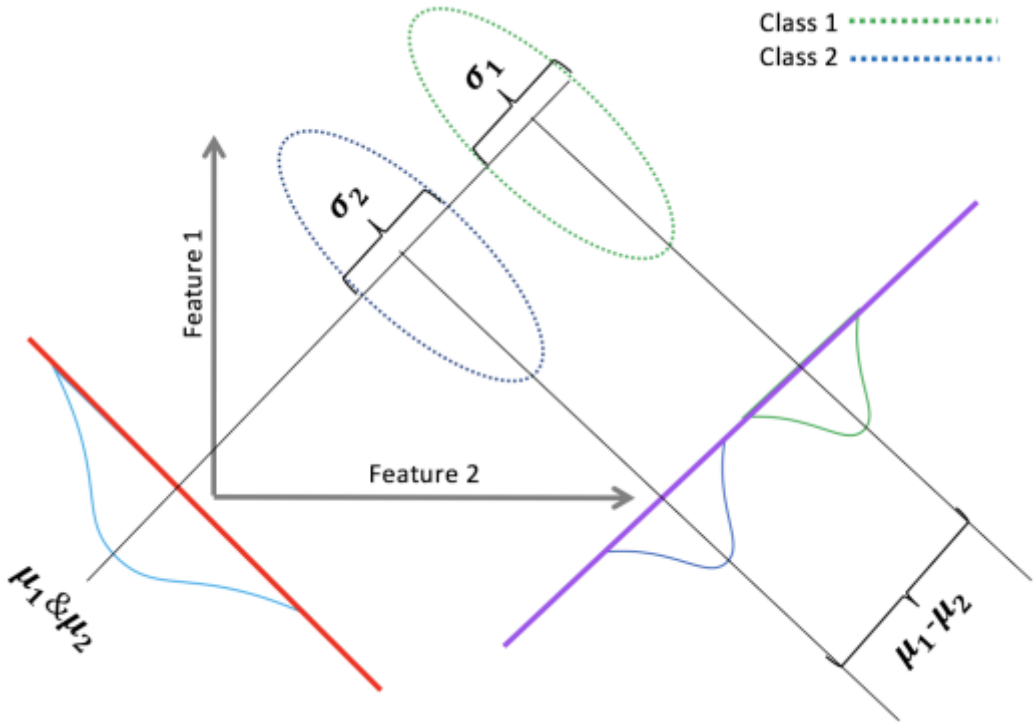

Figure 5. The figure shows two different subspaces which the data can be dimensionally reduced to. It consists of data representing two classes graphed against two different features and resembles the most and least optimal subspaces. [5]

2. Logistic Regression

Multinomial logistic regression builds upon binomial logistic regression, where the number of classes is limited to just two (for instance 0 or 1) and the number of features limited to one. In binomial logistic regression, we determine a logistic function which relates to the probability of either being one of the classes with a linear model. This probability can be summed up by the algorithm below:

$$\ln\left(\frac{\Pr(0)}{1-p(0)}\right) = \beta_0 + \beta_1 x_1 \qquad (7)$$

Where β_0 and β_1 are regression coefficients determined in the training phase that help fit a linear model to the logistic model and x1is the value of the feature, and Pr(0) is the probability that the incoming feature is in the class 0. This probability can be extracted through the following algorithm:

$$\Pr(0) = \frac{e^{\beta \cdot X}}{1 + e^{\beta \cdot X}} \qquad (8)$$

Note, the linear model represented in (7) can be represented as a dot product between vectors where the first element in the feature vector is 1. This binary classification and can be used for multiple classes. Logistic regression can find K possible classes by running K-1 independent binomial logistic models which can be represented through the following algorithm:

$$\Pr(class\ Y_i) = \frac{e^{\beta_i \cdot X}}{1 + \sum_{k=1}^{K-1} e^{\beta_k \cdot X}} \qquad (9)$$

where Y is an integer that represents one of the classes. Multinomial logistic regression calculates the probability for each class from 1 to K. The highest probable class is then the final prediction the classifier outputs. [6]

3. Decision Trees

Decision trees split the data based on attributes. For instance, in the tree below, the blue circles are attributes while the non-filled circles represent separated classes:

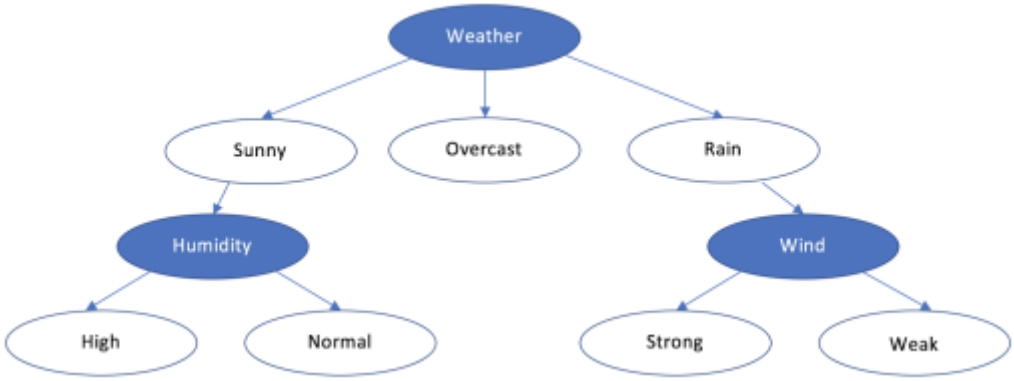

Figure 6. A graph representation of a decision tree. The solid blue ovals represent the attributes while the blue-outlined ovals represent the actual splits to create the tree. [7]

Decision trees uses the ID3 algorithm developed by Ross Quinlan in 1986. It splits the data based on the best attribute for the data. It stops splitting based on attributes once the child nodes are perfectly classified, as in, there are no other attributes that could split them.
We also use the CaRT (Classification And Regression Tree) algorithm developed in 1984. It uses a Gini impurity which measures how often a randomly selected element is identified as the incorrect class if it was randomly given a class from the subset. The Gini impurity algorithm is as follows:

$$I_G(p) = \sum_{i \neq k} p_i p_k$$

where $I_G(p)$ is the Gini Impurity and p_i and p_k is the probability that an item with the class i or k. Minimizing the Gini impurity leads to splitting the tree in the most optimal way.
Classification identifies the target by reaching the most accurate leaf node with the closest attribute. [7][8]

4. K-Nearest Neighbor

KNN is a type of instance-based learning where all computation is during classification. It uses the majority vote of its neighbors as its prediction. The object being predicted is assigned to the class most common amongst its k nearest neighbors, where k is a positive integer. The graph below resembles a 2-class, 2-feature example of KNN.

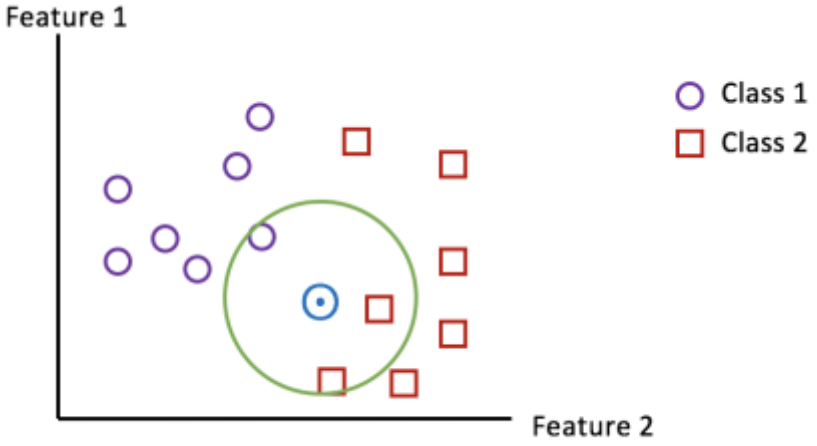

Figure 7. The image above shows a model developed by graphing two classes based on 2 different features. The blue target shows the data that wants to be predicted. The green circle represent when k is 3.

The graph above shows a blue target as the incoming feature that needs to be predicted while the purple circles and red squares are other classes. In an instance where k equals 3, the algorithm chooses the 3 closest data points from its training data. The greatest represented class in those data points is the prediction outputted by the KNN algorithm. [8]

D. Cloud Computing

While all data could be trained, classified and validated locally on every user's device, certain processes could be migrated to a cloud server to save local CPU power, thus further signifying the importance of cloud computing. With a reliable cloud computing system such as Amazon Web Services, complex computing capabilities are made endlessly possible with a total of 1.4 million CPU's, GPU's and FPGA serves on demand [9]. Within Amazon Web Services exists Elastic Compute Cloud, which provides the ability to use virtual computers in order to run any type of application. This is important so that the same computing power is distributed among users as it is done in the cloud instead of locally on each device. With that, we are assured that results from calculations will be immediately available, therefore increasing efficiency and accuracy.

II. Design and Implementation

A. MyoHMI Modules

Figure 2 shows the overall architecture of the application. It can be divided into three different parts, which are the signal processing modules, file interaction modules, and graphical user interface (GUI). The signal processing modules communicate with the myo armband, processes the incoming data, and performs gesture classification. The file interaction modules allow us to save data needed to perform experiments and future improvements on the system. Finally, the GUI allows the user to have an intuitive interface; in which, they can access all the functions that the app has implemented. Additionally, he or she obtains feedback from the app regarding what gesture is being performed.

Signal processing modules

1. Data Collection Module – MyoGattCallback

This module allows the application to scan for Bluetooth devices in its immediate surrounding using BLE. It identifies 4 different Bluetooth characteristics and detects whenever there is a change in any of them and updates the stream. It receives two streams of data for a total of one stream of 16 bytes. However, the Myo Armband can only receive 8 bytes at a time, so the EMG data has to be segmented into two vectors of 8 bytes each.

2. Feature Calculation Module - FeatureCalculator

The module receives raw EMG data; it segments the data into overlapped sliding windows of length 40 with window increments of 8. In each window, the features: MAV, WAV, turns, zeros, SMAV, and AU are extracted from each of the 8 channels of the Myo Armband. After all the features are obtained, they are concatenated into one data (feature) vector. One-hundred samples are collected for each gesture performed. The application gives flexibility to the user as they are able to select which features they want to use.

3. Classification Trainer and Classification Module - Classifiers and SMILE Library

The classification algorithms implemented in this module utilize the SMILE (Statistical Machine Intelligence and Learning Engine) library written by Haifeng Li. SMILE has several classifiers readily available. First, the classifiers are trained. The classification trainer takes in 100 data vectors for each gesture selected. Each data vector is converted into an array of doubles, since implementation requires data vectors to be of a double data type. For each of these data vectors, the classification trainer also takes in an array of integers corresponding to gestures associated to each data vector, called class labels. Once all the data is collected, a model is built based on which classifier is selected. Once the classification models have been created, the remaining data vectors from the feature calculator are sent to the classification module. The classifiers perform predictions on incoming data vectors, this time without class labels. The classifiers currently implemented in the application are: linear discriminant analysis, logistic regression, decision tree, and k-nearest neighbor. Support vector machine, neural network, and AdaBoost (based off decision trees) are currently being implemented and optimized.

Finally, this module also performs model validation by implementing cross validation. The method generates a confusion matrix for the selected classifier. It takes in the same samples used during the training phase of the classifier and segments them into equally sized parts. It then takes 4 parts to train the classification model and performs predictions on the remaining group. This process is repeated 5 times and in each iteration the training and testing sets are changed. Once finished, the average of all the predictions is taken and the matrix is returned. This is used to show how accurate the selected classifier is.

File Interaction Module

This module allows the user to save the data vectors created during the training session into a text file. The user will then have the option to upload and store it in the cloud or store it in their SD card.

Cloud storage is one of the most important parts of the application. By implementing cloud storage, we have created a way to analyze a variety of data based on a large amount of users. Cloud storage provides accessibility of data that could be used for further improvement of the application, the research itself, as well as supply research and analysis in many different fields. This feature also allows for mobility and efficiency of computing capabilities. This feature was generated through the use of several different software applications from Amazon Web Services--a software that offers cloud storage, database storage, content delivery, and compute power.

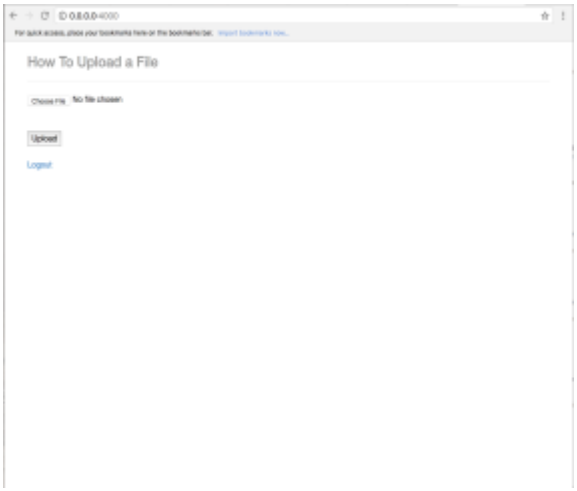

Figure 8. Upload feature in the Flask application, all deployable to Amazon's S3

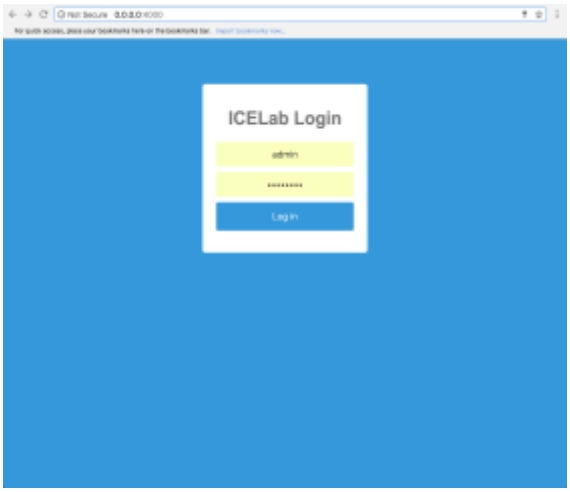

Figure 9. Log-in feature in the Flask application

Elastic Compute Cloud (EC2) provided the rental of virtual machines on the cloud which enables complex applications to be ran without running into potential hardware and software problems. With EC2, access keys and secret key ID's were provided in order to synchronize with other software systems and allows access for applications to be ran in the cloud. We then used Flask, a python based micro framework, to create an application that stores data directly to Amazon's Simple Storage Service (S3) from an upload feature shown in Figure 8. We also created a login feature (Figure 9) for added security to the application. This application is made deployable to the web through Amazon's Elastic Beanstalk, a cloud deployment and provisioning service.

While it is important to be able to save data manually through the Flask application, tested data within the mobile application must have the same storage capabilities without having to log on to a website. Within the mobile app, constructed a connection to S3 through setting up credentials that enables connection to our virtual computer in EC2. Saving data in the cloud can now be accessed the click of the cloud button, which provides options to save locally or through the cloud (see top left of Figure 13 on page 10).

Graphical User Interface (GUI)

The GUI implementation of the app is done by using Java and eXtensible Markup Language (XML). This is an appealing and user friendly GUI which contains three main tabs: EMG, Features, and Classification.

Figure 10. Bluetooth scanner screen in the MyoHMI application, displays all bluetooth devices detected.

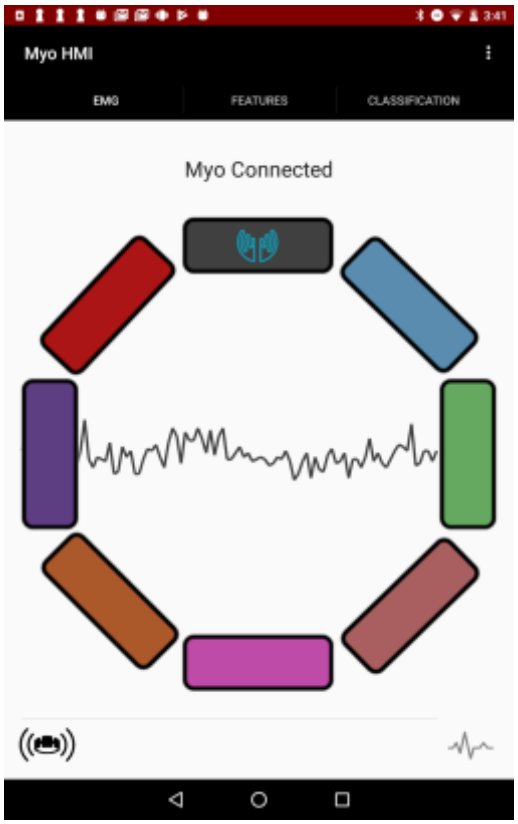

Figure 11. EMG tab in the MyoHMI application has a graph that displays raw EMG data.

In Figure 10, a button on the top right side of the app gives users the option to connect or disconnect to the armband. The bluetooth devices detected will be displayed on a list. Once the user clicks on the desired device, it will take them back to the EMG tab. As shown in Figure 11, the EMG tab includes a text message on top that displays the status of the Myo Armband, connected or disconnected. It also has a circular image with clickable colored buttons representing each sensor on the device. The graph's color and raw data output will change according to the selected sensor. The bottom right icon is used to start and/or stop streaming raw data. The icon on the bottom left corner will vibrate the armband.

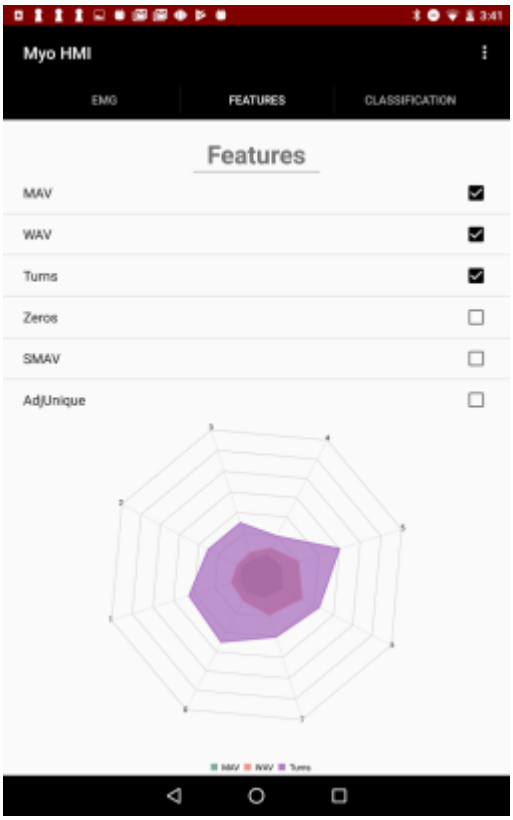

Figure 12. Features tab in the MyoHMI app, displays selected features on the radar graph

Feature 13. Classification tab in the MyoHMI app. Allows training, prediction of gestures, and data storage.

In the Features tab, shown in Figure 12, the user can select which features to implement before training the gestures. In addition, there is a radar graph that indicates the values of the features corresponding to each channel. In the classification tab, as seen in Figure 13, the user is shown a list of predetermined gestures and classifiers from which they can choose any combination to be used. Moreover, the user has the flexibility to clear and delete gestures, as well as add a custom gesture to the list. After the user has trained the desired classifier according to the particular setting they have chosen, the live box will output the predicted gesture. Additionally, the user can save the training data onto the device's local storage or upload it to the cloud storage service.

B. Experimental Protocol

Ten subjects participated in the experiment which consisted of eight male and two female. Each subject conducted three trials using the classifier K-Nearest Neighbor and the feature SMAV. We chose K-Nearest Neighbor because it was the most accurate classifier in our offline cross-validation tests as seen in Table 1 below.

Table 1. Shows the average accuracies with standard deviations and confidence intervals for the four implemented classifiers. Accuracies were found using seventeen training data sets through our cross-validation testing.

	Average	Standard Deviation	Confidence Interval (%) 95	98	99
LDA (%)	97.544	3.276	0.809	0.962	1.065
Logistic Regression (%)	98.863	2.667	0.659	0.783	0.867
Decision Tree (%)	97.685	3.348	0.827	0.983	1.088
KNN (%)	99.401	1.473	0.364	0.433	0.479

We chose SMAV because it is the best single-feature representation of EMG data. Each subject trained the following eight gestures: rest, wave in, wave out, point, fist, open hand, supination, and pronation. In the beginning and between training each gesture, there is a three-second interval that prompts the user to prepare to hold the next gesture in the queue of selected gestures. Figure 14 demonstrates a subject holding the prompted gesture. Once all the gestures were trained, the subject tested the app's ability to recognize the gestures performed as shown in Figure 15, and the data collected by that trial is then saved into the device's internal storage which concludes one trial. After all trials have been completed, the subject filled out an online survey providing ratings regarding the responsiveness, accuracy, usability, and aesthetic of the mobile application. The survey questions allow the subject to answer the questions from a 1 to 5 rating in which 1 is Poor, 2 is Fair, 3 is Satisfactory, 4 is Very Good, and 5 is Excellent.

Figure 14. Training Gestures.

Figure 15. Gesture Prediction.

III. Results and Discussion

A. Results

Based on the surveys taken by the subjects, the subjects responded fairly positively as shown in Table 2. The ratings in the survey are from a 1 to 5 rating in which 1 is Poor, 2 is Fair, 3 is Satisfactory, 4 is Very Good, and 5 is Excellent. In regard to the control scheme, on average the responsiveness of the gesture recognition was a 4.4 out of 5 and the accuracy of the accuracy of the gesture recognition was a 3.6 out of 5. In regard to the usability of the app, on average the ease of use was rated a 4.5 out of 5 and the aesthetic of the app design was rated a 4.5 out of 5.

Table 2. Average Rating of Mobile Application

Question Topic	Responsiveness	Accuracy	Ease of Use	Aesthetic
Average Rating	4.4	3.6	4.5	4.5

B. Discussion

Overall, the app seemed to work well for the subjects in terms of responsiveness and aesthetics. However, the accuracy of the gesture recognition was rated the lowest, averaging at a 3.6 out of 5. This was surprising since our offline tests were very accurate. We suspect this rating to be lower than expected because when predicting gestures, the user has to use the same gesture intensity as when they trained it. This is especially important since the classification algorithms only relied on the SMAV feature. We found that fatigue played a large role in users not gesture intensity output.

We also observed, through our tests and survey, a learning curve to the MyoHMI application. Some users were able to quickly produce the correct gesture based by mimicking their training while others found it harder.

When testing the subjects, we had them select the classifier K-Nearest-Neighbor since it gave the best results prior to the trials being conducted. As for selecting the features, we chose to only conduct the trials with just SMAV since it is the best individual feature. The layout, design, and overall aesthetic of the application is still in its early stages of development, but it's a solid foundation for future development.

IV. Conclusion

The work completed during this project has expanded the HMI from a desktop application to an android device application. The HMI can now be used anywhere and the user is able to save the EMG data from training gestures in the cloud storage if there is an established cellular signal or Wi-Fi connection. Additionally, with the help of the SMILE java library, we were able to implement and test several more classification algorithms. These additional algorithms dramatically increased the performance of the machine learning portion of the HMI. Our contributions have laid the foundation for the advancements that the application will be able to perform in the future such as implementation with external applications and cloud computing.

V. Future Work

Classifiers

So far, we have four classification algorithms working at a very fast and efficient rate. Implementing more classifiers, specifically neural-networks to enhance deep learning capabilities, may allow for more accurate predictions.

Use with External Applications

Since the program currently is able to dynamically recognize gestures, the next step would be to connect it to an external application such as a prosthetic hand or to a virtual reality environment.

Cloud Storage and Cloud Computing

The future also calls for cloud computing. Cloud computing gives access to superior processing power compared to local android devices. We would have the devices send data to cloud servers to utilize the cloud computing resources to relieve local workload. The data being sent to the server would allow for post processing to help better user experience. For instance, cloud computing can create user-specific, independent, classification methods that would further boost prediction accuracy.

VI. Acknowledgment

This project is supported by the US Department of Education through the Minority Science and Engineering Improvement Program (MSEIP, Award No. P120A150014); and through the Hispanic-Serving Institution Science, Technology, Engineering, and Mathematics (HSI STEM) Program, Award No. P031C110159.

We would like to thank Dr. Amelito Enriquez from Cañada College for the opportunity to participate in this internship and for guiding us through the whole program. Also, we would like to express our appreciation to Dr. Xiaorong Zhang from San Francisco State University, our faculty advisor, and our San Francisco State graduate mentor, Alexander David, for all his guidance and advice throughout the internship.

References

1. Zhang, X.., Chen, X., Li, Y., Wang, Kongqiao, and Yang, J. “A Framework for Hand Gesture Recognition Based on Accelerometer and EMG Sensors.” *IEEE Transactions on Systems, Man, and Cybernetics.* 41.6 (2011).

2. Englehart, Kevin. “A Robust, Real-Time Control Scheme for Multifunction Myoelectric Control”. *IEEE Transactions on Biomedical Engineering*. 50.7 (2003).

3. Phinyomark, Angkoon, Chusak Limsakul, and Pornchai Phukpattaranont. "A Novel Feature Extraction for Robust EMG Pattern Recognition." *Journal of Computing*, (2009).

4. Donovan, I., Puchin, J., Okada, K., Zhang, X. “Simple Space-Domain Features for Low-Resolution EMG Pattern Recognition,” in 39th Annual International Conference of the IEEE EMBS, Jeju Island, Korea, 2017.

5. Lavrenko V., Sutton C. IAML: Dimensionality Reduction. *School of Informatics*, University of Edinburgh [Lecture]. Retrieved from: http://www.inf.ed.ac.uk/teaching/courses/iaml/2011/slides/pca.pdf.

6. Bohning, D. (1990). Multinomial Logistic Regression Algorithm. *Annals of the Institute of Statistical Mathematics*, 44(1), 197-200.

7. Lavrenko V., Sutton C. IAML: Decision Trees. *School of Informatics*, University of Edinburgh [Lecture]. Retrieved from: http://www.inf.ed.ac.uk/teaching/courses/iaml/2011/slides/dt.pdf.

8. Smile - Statistical Machine Intelligence and Learning Engine. 2017. Retrieved from: https://haifengl.github.io/smile/

9. Morgon, T. “A Rare Peek Into The Massive Scale of AWS. Retrieved from: https://www.enterprisetech.com/2014/11/14/rare-peek-massive-scale-aws/, 2014.