# Electromyography (EMG) Based Finger Movement Detection

Ghazaleh Jowkar

A Thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science

University of Washington

2020

Reading Committee:

Wei Cheng

Juhua Hu

Eyhab Al-Masri

Program Authorized to Offer Degree:
Computer Science and System

University of Washington

## Abstract

Electromyography (EMG) Based Finger Movement Detection

Ghazaleh Jowkar

Chair of the Supervisory Committee:
Assistant Professor Wei Cheng
School of Engineering and Technology

One fundamental component of much modern human-machine interaction (HCI) devices is Myoelectric control systems which is a system that receives the Electromyography(EMG) signal originated from muscle movement. Much research has focused on determining the best general structure of the control system for a given application where the same element choices are used for all subjects. However, due to the nature of the signal and the human body, the best structure may be subject-specific. The primary aim of this research can be categorized into two major areas.

- Movement extraction and movement duration detection from a recorded set of moves.
- Subject-specific selection of classification system elements (i.e., feature set, classifier, window characteristics, dimensionality reduction method) for individual finger movement detection.

In this study, we focus on individual finger movements, therefore, two movement sets for each finger were tested: one where each finger is closed for half a second and open for half a second and the other with the duration of one second closed finger one-second open finger. Myoelectric data were collected from the forearm muscles of 27 years old female subject using a single channel Epidermal Electric System (EES) designed in [41]. We performed three sets

of tests on our subject; giving us three data sets to study and develop a model for. These data were first got prepossessed for movement extraction then used to train and test a series of classification systems, each consist of a different combination of system element choices. We introduce a novel model for movement extraction from an unfiltered EMG signal and achieve an average accuracy of 88% for our five class finger movement classification. Additionally, we show the effect of Principal Component Analysis on Classification, as well as multi-layer classification of EMG signals.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# ACKNOWLEDGMENTS

I would like to express sincere appreciation to my adviser Dr. Wei Cheng who has been very supportive and leading through out my graduate studies at University of Washington. I would like to thank Dr. Juhua Hu for her lead and support during this work, I've learned a lot from her.

# DEDICATION

To my dear Milad, who gives me courage.

# Chapter 1

# INTRODUCTION

Electromyography (EMG) signals are playing an important role in developing human-machine interaction devices. Naturally, the EMG signal recorded from the muscle contraction contains rich muscle information, which is beneficial in describing muscle behavior and condition, as well as the hand movement[16][40]. Classification of finger movements using EMG signals has not received much attention, most studies are either focused on gross hand movements or a combination of finger movements. Finger movement classification is a necessary precursor to control individual finger movements of an EMG-controlled prosthetic hand. It could also find use in areas such as application-specific(e.g., typing, playing music) prostheses and human-computer interaction for purposes such as Security and Authentication. Additionally, it can be used for HCI purposes such as device control, or muscle monitoring systems. The majority of Myoelectric classification research seeks a single classification system for a given movement set that performs best overall subjects a "one-size-fits-all" system. The intuitiveness of control is addressed solely by the classifier, which is trained separately for each subject, while the classifier type and the other elements of the classification system remain identical across all subjects. These types of studies do not consider the effect of gathered signals, type of the sensor and individual body differences, and in most cases, the study is performed based on an existing database of EMG signals from a pool of subjects. We believe that it may be possible to increase classification performance with greater user customization, tailoring all elements of the system to the subject, thereby exploiting potentially advantageous relationships among the system elements. We propose a unique approach to the classification of finger movements, based on the automatic extraction of finger movement signal with a variety of duration and generation of a customized classifi-

cation system for individual subjects. For this study we acquired our data using the sensor proposed in [41] and prototype in Georgia Institute of Technology. The sensor has a natural interface that is capable of accommodating the motions of the skin without any mechanical constraints, thereby establishing a robust skin/electrode contact to gather reliable and individual-based signals. The performance of various choices for each system elements such as the use of PCA in feature selection, validating different models of classifiers, and comparing the results of two-layer classifier versus single layer classifier for detecting individual finger movement classification was also investigated for our approach. While most studies claim that the type of classifier does not significantly affect the classification performance [37], we've shown that designing the right classifier has a significant impact on the accuracy of our detection. Additionally, we've shown that the quality of extracted features has a great impact on EMG signal classification [15]. Without loss of generality, the quality of the signal and the extracted movements have a high effect on the extracted features which has shown a great impact in EMG signal classification. Feature extraction is a technique to extract valuable information from the signal itself, which should contain much information as possible [37]. Feature extraction can be categorized into time-domain (TD), frequency-domain (FD), time-frequency-domain (TFD). We compared the quality of the features and the classifier's results using TD, FD, TFD features. For each domain, we used the whole vector of data in each domain as a vector of features to keep the whole signal's information. Additionally, we extracted 17 features from the signal in each domain and compared the results using the vector of 17 features. Comparing the result of the classifiers as well as analyzing the quality of the features showed us that using a vector of TD features gives us higher accuracy versus using the whole signal as a vector of features. These comparisons gave us a good point of start for designing a classifier. Since TD features gave us a promising initial result, we started our analysis and design using TD extracted features. This approach has both advantages of less overhead as well as higher accuracy compared to every other tested feature. Six popular machine learning algorithms that we used for classification include Convolutional Neural Network (CNN) [31], Deep Neural Network (DNN)[31], k-Nearest Neighbor (KNN)[3], Logistic

regression (LR)[26], Quadratic Discriminant Analysis (QDA)[33], and XGBoost [7].

### *1.1 Background and related works*

There has been a vast majority of studies on the EMG signal and its usage for movement detection. Some studies focused on variables that affect the signal recording such as type of the sensor or number of channels for recording. While others mainly studied feature extraction and best features to extract with most information from EMG signals [36][32][29] [35][22]. Feature extraction can be categorized into time-domain (TD), frequency-domain (FD) and time-frequency-domain (TFD) where TD features are the most commonly used. A past study, [18] introduced five EMG features for pattern recognition. The authors indicated that the proposed features are good in discriminating the EMG patterns. Later [21] developed a subset of features based on time-dependent spectral moments to classify the multiple hand movements at different limb positions. Moreover,[30] proposed three new EMG features for arm movements classification. The author showed that proposed features outperformed other conventional features in EMG pattern recognition. FD features only contains spectral information, in which the time information is limited. Our study shows that for our recorded EMG signals spectral information is not very useful. Therefore we focused on TD features. The majority of studies have been focused on extracting accurate movement decisions from EMG signals. Classification accuracies above 90% are commonplace in the literature for upper arm, wrist and gross hand movements [15][2] [8][13][10][12][11][14][17][20][25]. However, classification of finger movements using an EMG pattern recognition system has not received similar levels of attention. Finger movement classification is a key factor to detect movements for a variety of applications such as application-specific prostheses and human-computer interaction. Classification accuracy of 86% for flexion of digits one through three, flexion of all fingers, and relaxation of the hand has been reported for a single subject [39]. For flexion of digits, one through three and hand closure, 98% classification accuracy was reported for a single subject[38]. More recently, average accuracies of 94.3% for five normally-limbed subjects and 87.8% for a single trans-radial amputee have been achieved [34] for a

movement set comprising flexion and extension of each finger on one hand, and flexion and extension of the middle, ring, and little finger as a group. In [4], two different movement sets on a keyboard for typing were tested, one involving digits two through five (3 class), and the other involving digits one and two (2 class) with an accuracy of 93% on average. The majority of Myoelectric classification research seeks a single classification system for a given movement set that performs best overall subjects—a "one-size-fits-all" system[37]. In this study, we propose a user-specific model for detecting all individual finger movements (5 classes of move) with an average accuracy of 88%.

# Chapter 2

# EMG DATA RECORDING

The surface electromyographic (sEMG) signal is a combination of EMG signal that generated from the muscle movements and unavoidable noise components which affect the signal and may cause distortion. Especially, when the signal is obtained during dynamic movements (similar to our experiment) and when it is used for gathering information about the physiology and anatomy of muscles. Devices that are designed based on fully integrated electronics that have soft, stretchable forms to match the physical properties of the skin itself which [41] refers it as epidermal electronic systems (EES), laminate and adhere directly on the skin, in a conformal manner [41]. The result is a natural interface that is capable of accommodating the motions of the skin without any mechanical constraints, thereby establishing not only a robust, non-irritating skin/electrode contact but also the basis for intimate integration of diverse classes of electronic and sensor technologies directly with the body. For this study we acquired our data using the sensor proposed in [41] and prototype in Georgia Institute of Technology. In this chapter, we'll go over the design and material of the EMG Sensor used for our experiment, we'll explain the experiment protocols and our test subject.

## *2.1 Data acquisition protocol*

There are invasive and non-invasive methods for gathering EMG signals. The invasive method involves surgery and is difficult, where a non-invasive method is excellent, cost-effective and more convenient to use [5]. Thus, surface EMG is used for motion classification in this work. We used significantly thin variants of EES proposed in [41]. The construction of these multifunctional EES consists of an interconnected collection of thin, Filamentary Serpentine (FS) conductive traces and integrated devices, all in an open mesh layout [41]

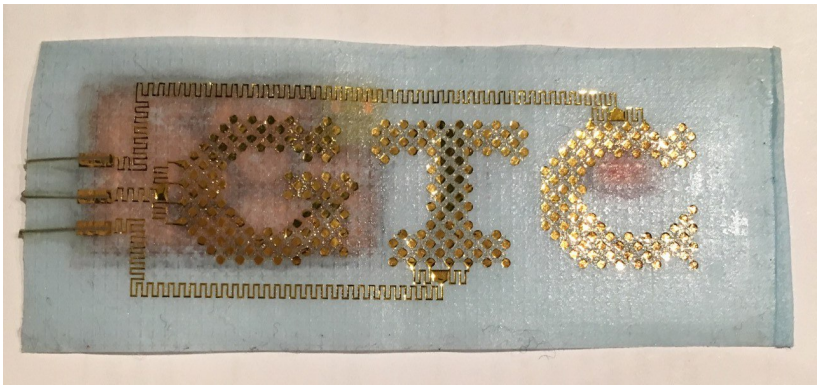
that provide very low and effective elastic moduli and large deform-ability, at the level of the overall system, even when the mesh incorporates brittle, high modulus materials such as silicon[5]. These properties allow the EES to follow the contoured surfaces and time-dynamic motions of the skin, in a natural way. The EP sensor includes three electrodes, each in the form of an FS mesh with exposed metal (Au) that contacts the skin directly, for measurement (MEA), ground (GND) and reference (REF). Figure 2.1 is showing the FS mesh that lays on the skin and the circuit that makes it possible to perform the EMG recording and Bluetooth interface with the android device.

### *2.1.1 Test Subject*

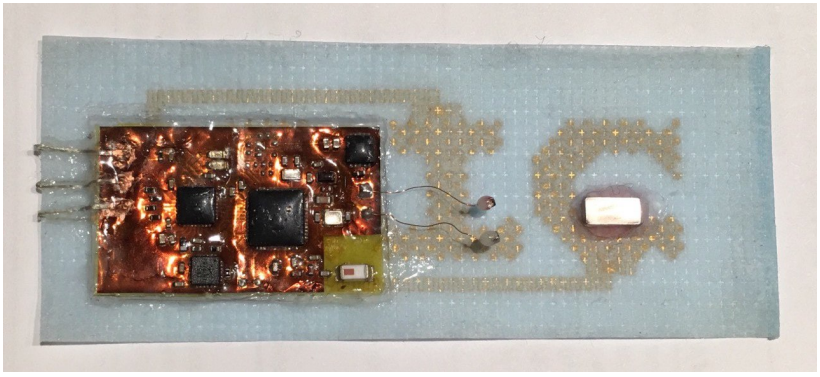
Our test subject was a 27 years old female, the sensor was located on the forearm of the right hand of the subject. Figure 2.2 shows where we attached the sensor to record the EMG signals generated as a result of finger movements of our test subject as well as the real-time interface of the sensor with the android mobile app. As figure 2.2a is displaying the sensor needs to be parallel with the targeted muscles, we found that locating the sensor with this attachment will give us the most visibility to the 3 major muscles (Palamaris Longus, Flexer Carpi Radialis, and Pronator Teres) which play important role in finger movements.

### *2.1.2 Experiment Protocol*

We ran three sets of experiments on our test subject, in each experiment we located the sensor in approximately the same location on the forearm of the right hand of the subject. The sensor has an Android mobile interface via Bluetooth that provides single-channel EMG recording at 250 Hz. Figure 2.2b shows a real-time graph of the received EMG signal, we can export the sampled signal to an excel file. In each test, we recorded over 100 samples for each of the five finger movements which give us 500 samples per experiment. Since our goal is to be able to detect movements with different length of out of each 100 samples, we recorded 50 samples with the length of 1 second closed finger, and 50 samples with the length of half a second closed finger. That means the total duration of the close and open



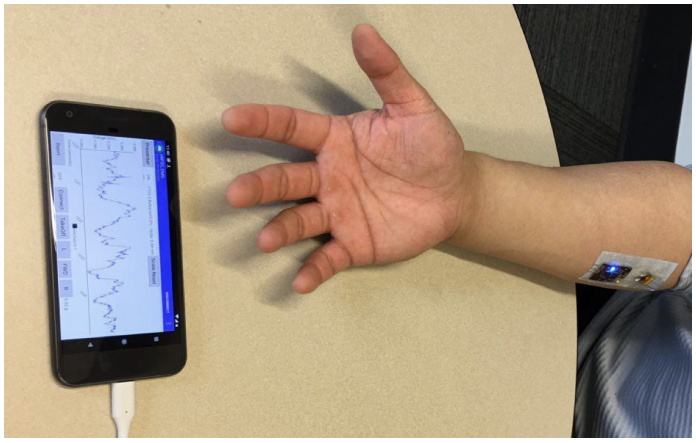
(a)



(b)

Figure 2.1: (a) Three electrodes of the Epidermal sensor, each in the form of an FS mesh with exposed metal (Au) that contacts the skin directly (b) Electrical circuits for Bluetooth connection with the android device and EMG recording

(a)



(b)

Figure 2.2: (a) Sensor attachment on the forearm of the right hand of the subject (b) Real-time interface of the sensor and the Android mobile device

finger are 2 second and 1 second respectively. After recording the data we have a database of raw signals where we will pre-process them to be suitable for feature extraction and machine learning algorithms.

# Chapter 3

# PRE-PROCESSING OF SIGNAL

The received EMG signal is noisy and disported, additionally we for each finger we recorded all of the movements in a single recording, as a result not only we have to reduce the noise from the signal but also we need to extract the movements from the received signal Figure 3.1 illustrates the flow diagram of the proposed EMG finger movement recognition system. Firstly, the EMG signals are acquired from epidermal electronic systems (EES). Afterward, signals are passed through a high pass filter to eliminate the noise caused by electrode-skin impedance, and continuous body movements. After that finger movements are extracted from the signal, which gave us a database of 100 samples per finger per experiment. We analyzed the signals in both time and frequency domain as well as the Wavelength coefficients (TFD). Since TD features provide higher accuracy and due to the low overhead and the need of being able to analyze real-time, we continued the work with analyzing the signals in TD. Next, the features are extracted from the signals to form an EMG feature set. The feature set that consists of several features are then fed into the classifiers (machine learning algorithms) for the classification of finger movements. We compared the result of different classifiers with and without using Principal Component Analysis (PCA) which is used for dimension reduction.

## *3.1 Noise and Filtering*

From the need to work with electrical biopotentials to control bio robotic mechanisms, signals are needed without noise and optimal for their use. In the case of Electromyography (EMG) signals that originate in the muscle is inevitably contaminated by various noise components and artifacts that originate at the skin-electrode interface, in the electronics that amplify

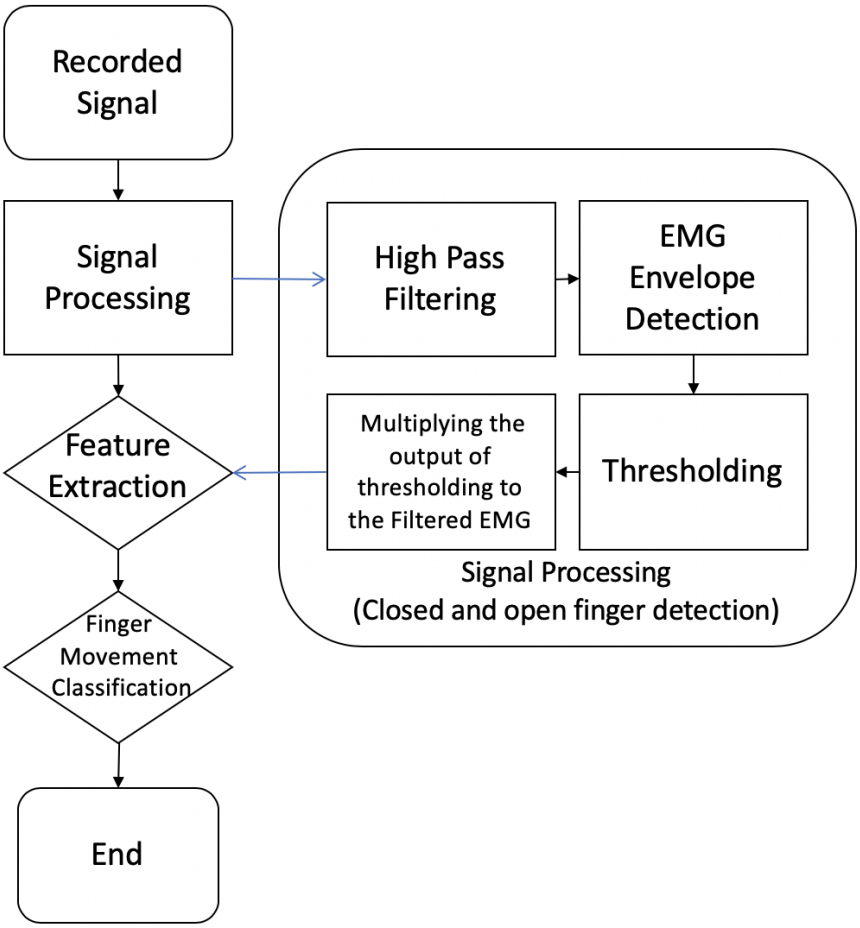



Figure 3.1: Flow Diagram of the proposed EMG Pattern Recognition System

the signals, and in external sources [2],[24]. For this reason, it's necessary to make digitally filters these signals after the acquisition. There are two major noise sources [18]:

- One originates in the electronics of the amplification system also known as thermal noise. The accuracy and intellect design of the electronic equipment can effect the noise, it also can be reduced by using high-quality materials, but it cannot be fully eliminated.

- Another is at the skin-electrode interface which is dominant and named baseline shift is detected whenever a sensor is attached to the skin [2], [24]. This baseline shift is caused by electrode-skin impedance and body movements. It is generated due to muscle movements, and the travel of force impulse through the muscle and skin causing a movement at the electrode-skin interface. However, the frequency range of this type of noise is normally between 1-10 Hz and almost completely can be canceled using a high pass filter [9].

Therefore, by removing the low-frequency noise components we render EMG signal for more practical applications, which can be done by selecting an appropriate high-pass filter and its corner frequencies. To choose a proper filtering system we need to consider many factors to remove these artifacts including the sensor configuration and specific noise source. The band-pass determination is always a trade-off between reducing noise and artifact contamination and preserving the desired information from the EMG signal. Because these noise contaminations are at low frequencies, we need to use a high pass filter to remove the noise.

In the case of EMS signals, we used Butterworth high pass filter [5]. Although the slope of this filter's response in the transition region between the pass-band and stop-band is smooth, it exhibits a nearly flat passband with no ripple [5] which can provide better performance than other filters (like Chebyshev).

The analysis established the relationship between the noise resources and the EMG signal

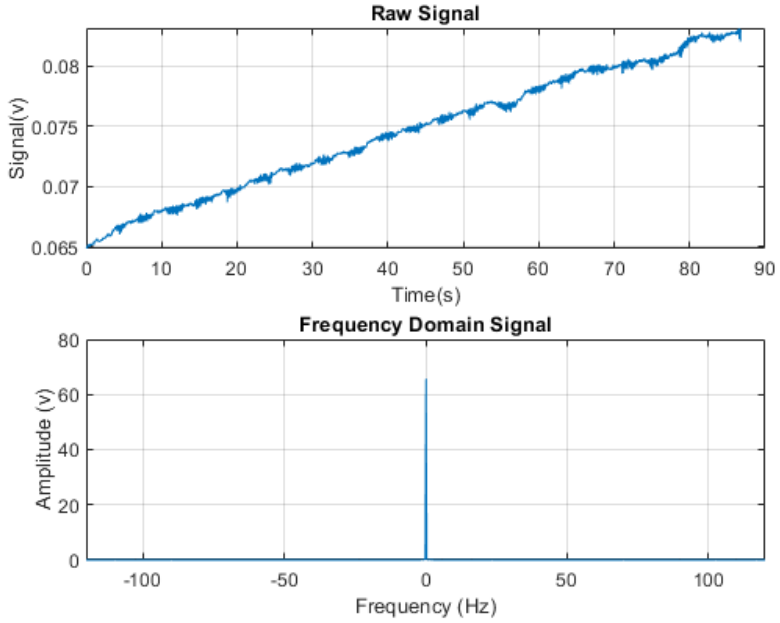



Figure 3.2: Captured signal before filtering in time and frequency domain

as a function of the filter bandpass. When this relationship is combined with other considerations related to the informational content of the signal, the signal distortion of filters, and the kinds of artifacts and noises, a Butterworth filter with a corner frequency of 20 Hz is recommended for general use [5]. The signal (EMG plus noise) captured by our device is shown in the Figure 3.2 The first graph is the received raw signal in the time domain and the second one is this signal in the frequency domain (the frequency axis is limited for a better view). Figure 3.3 shows the signal after filtering using a Butterworth filter with $F_{stop} = 14Hz$ and $F_{pass} = 16Hz$, in time and frequency domain. Compared with the signal shown in Figure 3.2, the EMG signal is completely separable with no DC level. In the next step, each finger movement in the filtered EMG signal should be separated to feed the movement duration calculator and classification blocks.

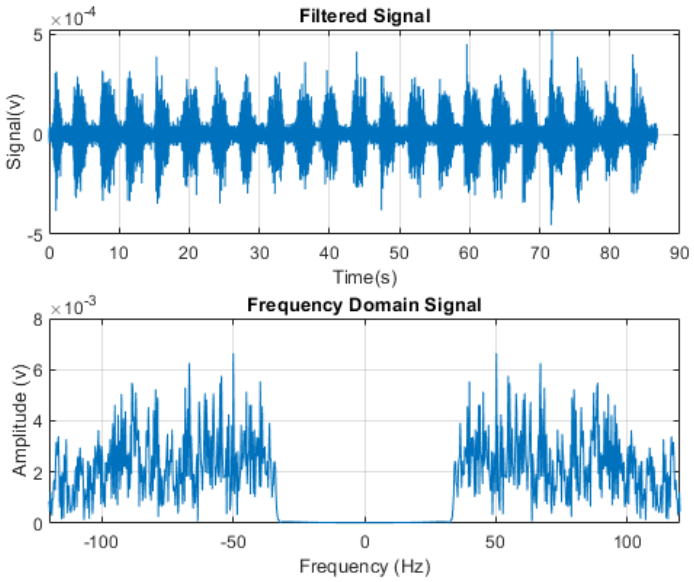



Figure 3.3: Captured signal after filtering in time and frequency domain

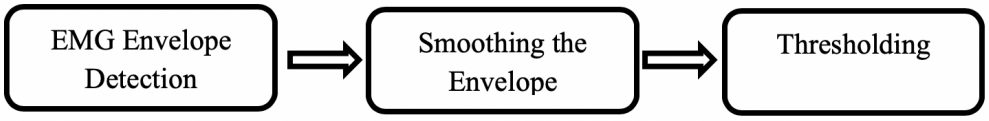



Figure 3.4: Proposed model for separating 'closed finger' and 'open hand' portion of the signal

## *3.2 Finger Movement Detection*

In this section, we discuss our proposed model for separating 'closed finger' and 'open hand' portion of the signal and extract the closed finger portion of the signal and its duration from the EMG signal based on a heuristic method. This method is presented in diagram shown in 3.2 which shows a general view of our model and each step steps of our model for movement separation. Looking at the diagram shown in 3.2, the first block returns the upper envelopes of the filtered EMG sequence, the envelope is the magnitude of the analytic signal computed

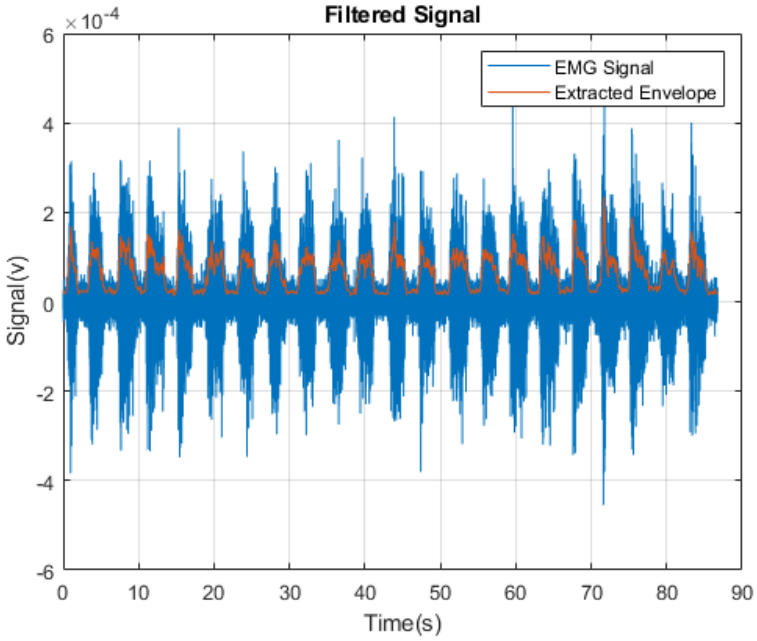



Figure 3.5: Upper envelope of the EMG signal

by Hilbert function. This way we will have the signal without any ripple and we're able to calculate the duration. The output of the first block is shown in Figure 3.5. As we can see, there are some sharp variations in the envelope of the EMG signal, and it can cause an error in duration calculations. Therefore, in the next block of the diagram shown in 3.2, the signal envelope $Y$ is smoothed ($z = smooth(Y)$). The Smooth function will smooth the data in the column vector $Y$ using a moving average filter where the results are returned in the column vector Z. The number of samples (or the span) for the moving average can be set. below we show an example of of how smooth function works, in this example, the span is 5. The first few elements of Z are defined as below. The number of samples (or the span) for the moving average can be set. Figure3.6 shows the output of the smoothing block of the

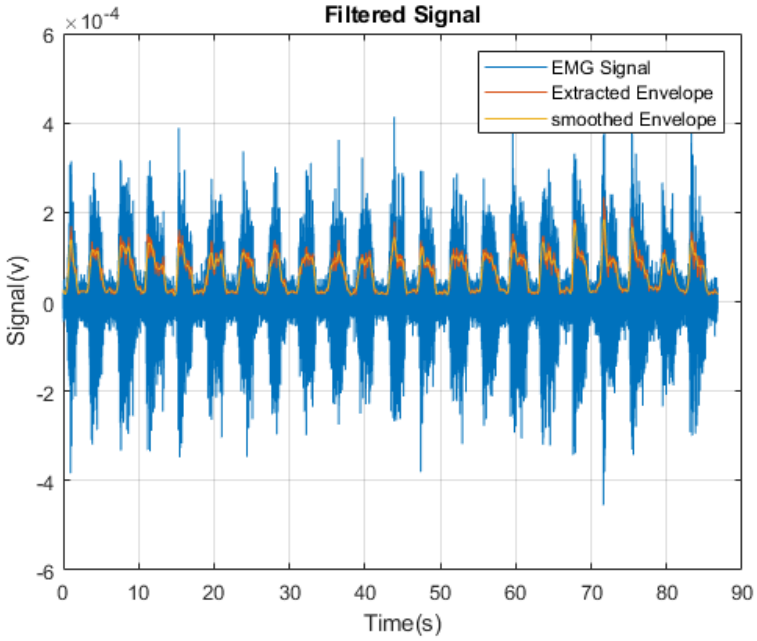



Figure 3.6: Captured smoothed signal, EMG signal and its envelope

diagram shown in 3.2, EMG signal, and its envelope.

$$
\begin{aligned}
Z(1) &= Y(1) \\
Z(2) &= (Y(1) + Y(2) + Y(3))/3 \\
Z(3) &= (Y(1) + Y(2) + Y(3) + Y(4) + Y(5))/5 \\
Z(4) &= (Y(2) + Y(3) + Y(4) + Y(5) + Y(6))/5 \\
Z(5) &= (Y(3) + Y(4) + Y(5) + Y(6) + Y(7))/5
\end{aligned}
\tag{3.1}
$$

Looking at the output of smoothing block 3.6, it is evident that sharp variations in the envelope are almost smoothed and the signal is ready for the "Thresholding Block" of the diagram shown in 3.2. At this step, we first need to define and calculate the threshold value. The threshold value is heuristically set by the average value of the EMG signal sequence as follow:

$$
Thr = \sum_{i=1}^{n} \frac{\mid EMG(i) \mid}{n} \tag{3.2}
$$

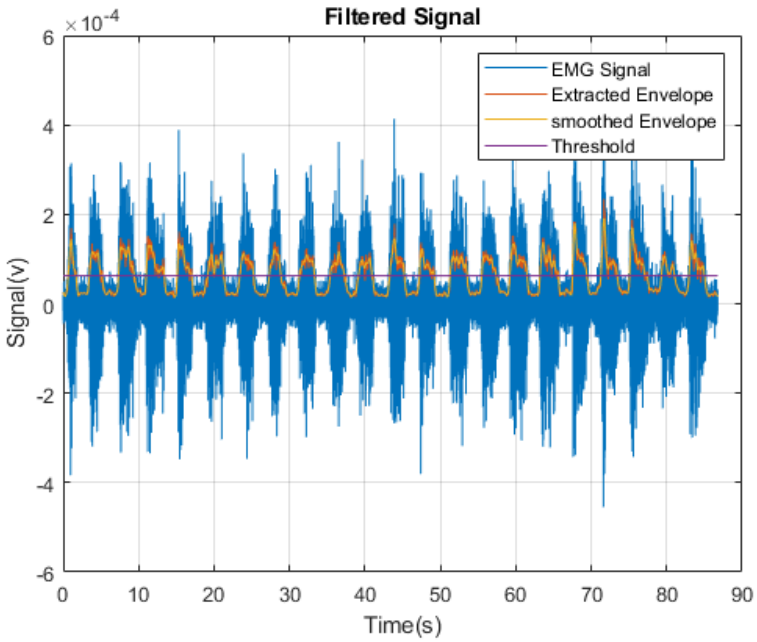



Figure 3.7: Captured smoothed signal, EMG signal, its envelope and the selected threshold

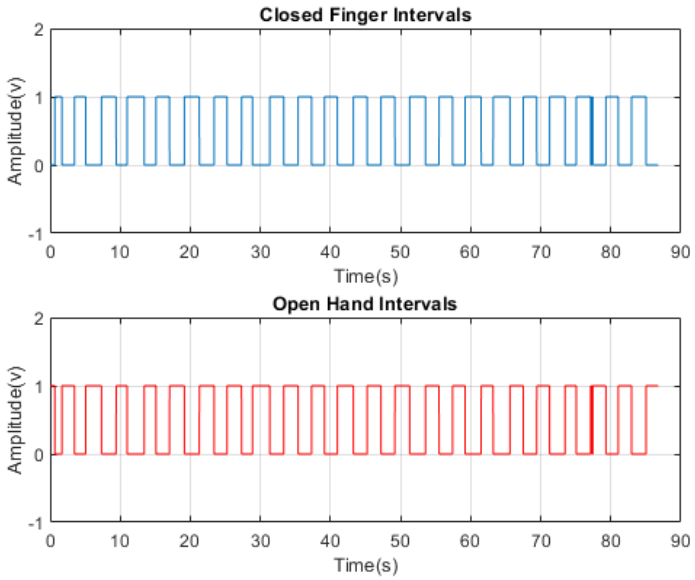



Figure 3.8: In this stage, the samples higher than threshold turn to 1 on the samples lower than threshold turn to 0

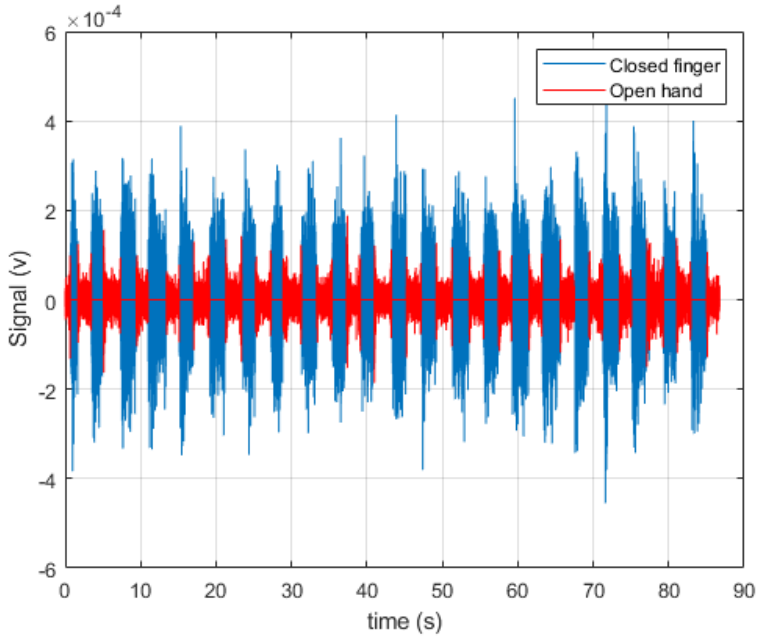



Figure 3.9: Representation of "Closed finger"/"Open hand" detection

Where n is the number of EMG samples. Figure3.7 reveals the thresholding process.

In thresholding stage, the samples higher than threshold turn to 1 on the samples lower than threshold turn to 0, as shown in Figure3.8. By multiplying the sequence shown in Figure3.8 (as a duration window), we can access to 'Closed Fingers' and 'Open Hands' samples separately. Figure3.9 and 3.10 and 3.11 demonstrate the separation of open hand and closed finger signals. After the separation process, each movement samples are stored in a row of an EMG matrix. For example, for the separated closed fingers samples (Figure 3.12), the related matrix would have 22 rows for 22 finger movements. As shown in Figure3.12, movements' duration can easily be calculated after separation by Subtracting the start time from the end time of each movement. We show that we can separate the finger movements with their duration, however, it's hard to choose a ground floor for the length of the extracted movements. However, it's hard to claim any percentage or accuracy for our detection. There are a lot of factors that make it hard to choose a ground floor for comparison of the actual

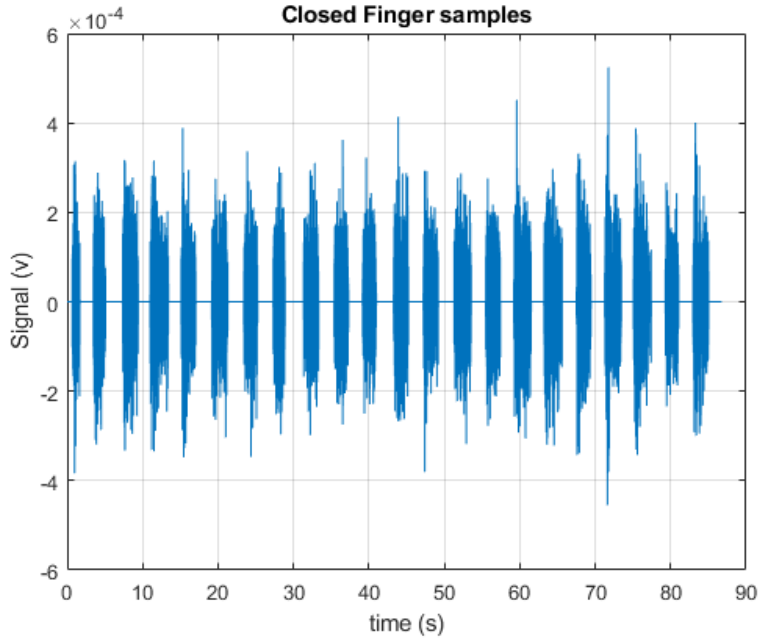



Figure 3.10: Representation of "Closed finger detected samples

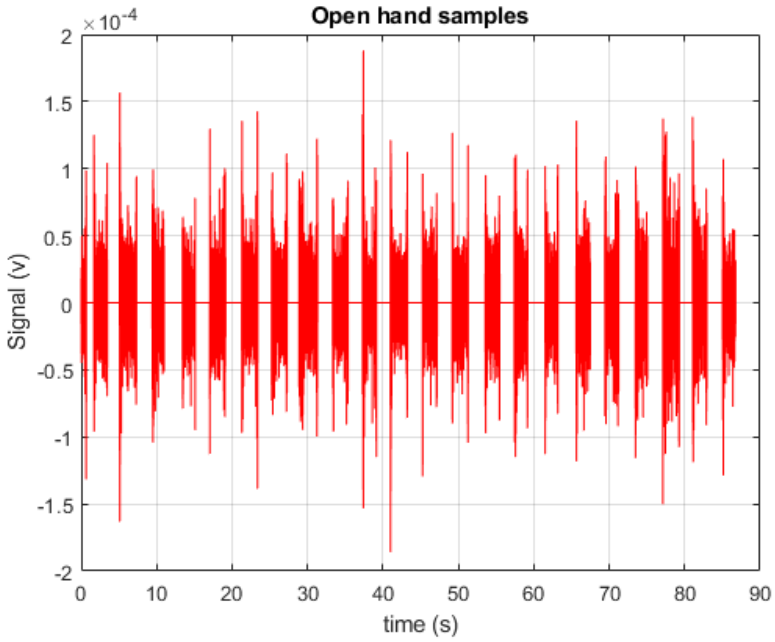



Figure 3.11: Representation of "Open hand" detected samples

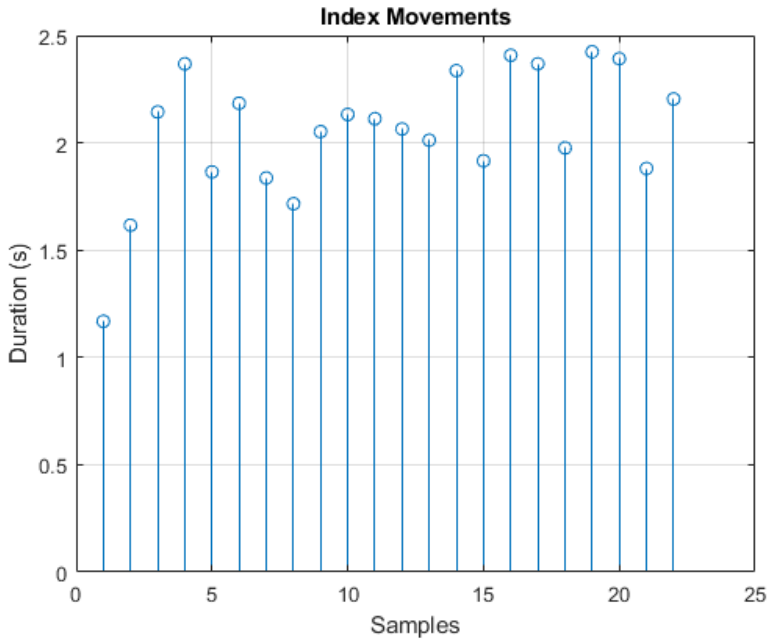



Figure 3.12: Example of the duration of 22 sampled movements extracted from the signal

duration of the movement versus the calculated duration. For instance, human error is playing a big role, let's say a test subject would move a finger for a 1 second, then depending on how strong and how week the move is, the shape of the signal and the threshold of it changes and our detection might detect it as a half a second move or a 1-second move. Due to these errors, we can't still give accurate estimates of our detection for the length of the movement.

# Chapter 4

# FEATURE EXTRACTION AND DIMENSION REDUCTION

EMG feature extraction is a process of rearranging the EMG data into a set of features. These features should contain the most descriptive information about the signal and should their size should be reduced in dimension compared to the input signal as a whole. Features should contain as much information with the highest variance. Feature extraction can be categorized into time-domain (TD), frequency domain (FD) and time-frequency domain (TFD), which is a very important step in designing any EMG pattern recognition system. In this study, we used TD features, which are the most commonly used, to feed them into the machine learning techniques for the finger movement classification process.

## *4.1 Conventional Feature Extraction*

In this study, we chose popularly and commonly used EMG features which are selected due to their simplicity and promising performances in previous works. In these relations, X can be time-domain samples, Frequency domain samples, or wavelet coefficient (as time-frequency domain) of an EMG signal and L is the length of the sequence [36].

- Mean absolute value (MAV) Mean absolute value is one of the most popular EMG features, and it is defined as the average of the summation of absolute value of signal. [18], [32][32].

$$MAV = \frac{1}{L}\sum_{i=1}^{L} \mid X_i \mid \tag{4.1}$$

- Wavelength (WL): WL is another popular EMG feature, and it can be calculated by

simplifying the cumulative length of waveform summation [18], [18].

$$WL = \frac{1}{L}\sum_{i=1}^{L} \mid X_i - X_{i-1} \mid \tag{4.2}$$

- Zero Crossing (ZC) It is an EMG feature that measures the frequency information [18].

$$ZC = \sum_{i=1}^{L-1} f(x_i) \tag{4.3}$$

where $f(x_i)$ is defined as bellow,

$$f(x_i) = \begin{cases} 1 & \{(x_i > 0 \& x_{i+1} < 0) | (x_i < 0 \& x_{i+1} > 0)\} \& \\ & \mid x_i - x_{i+1} \mid \geq T \\ 0 & otherwise \end{cases}$$

- Slope sign change (SSC) is a traditional EMG feature that determines the number of times in which the number of waveform changes sign. [18], [32].

$$SSC = \sum_{i=1}^{L-1} f(x_i) \tag{4.4}$$

where $f(x_i)$ is defined as bellow:

$$f(x_i) = \begin{cases} 1 & \{(x_i > x_{i-1} \& x_i > x_{i+1}) | \\ & (x_i < x_{i-1} \& x_i < x_{i+1})\} \& \\ & \{(\mid x_i - x_{i+1} \mid \geq T) | \\ & (\mid x_i - x_{i-1} \mid \geq T)\} \\ 0 & otherwise \end{cases}$$

- Average amplitude change (AAC), [18].

$$AAC = \sum_{i=1}^{L-1} \mid x_{i+1} - x_i \mid \tag{4.5}$$

- Log detector (LD), [35].

$$LD = exp(\frac{1}{L}\sum_{i=1}^{L-1} \mid x_i \mid) \tag{4.6}$$

- Root mean square (RMS), [22].

$$RMS = \sqrt{\frac{1}{L}\sum_{i=1}^{L-1}(x_i)^2} \tag{4.7}$$

- Difference absolute standard deviation value (DASDV), [22].

$$DASDV = \sqrt{\frac{\sum_{i=1}^{L-1}(x_{i+1} - x_i)^2}{L-1}} \tag{4.8}$$

- Myopulse percentage rate (MYOP), [18].

$$MYOP = \frac{1}{L}\sum_{i=1}^{L-1} f(x_i) \tag{4.9}$$

$$f(x_i) = \begin{cases} 1 & if x_i \geq T \\ 0 & otherwise \end{cases}$$

- Willison amplitude (WA), [35].

$$WA = \sum_{i=1}^{L-1} f(x_i) \tag{4.10}$$

$$f(x_i) = \begin{cases} 1 & if \mid x_i \mid \geq T \\ 0 & otherwise \end{cases}$$

- Simple square integral (SSI), [18].

$$SSI = \sum_{i=1}^{L-1}(x_i)^2 \tag{4.11}$$

- Variance of EMG signal (VAR) VAR is used for measuring the signal power.

$$VAR = \frac{1}{L-1}\sum_{i=1}^{L-1}(x_i)^2 \tag{4.12}$$

- Modified mean absolute value (MMAV) is an extension of MAV feature by assigning the weight window function. Mathematically, MMAV can be computed as below [29].

$$MMAV = \frac{1}{L-1}\sum_{i=1}^{L-1} w_i \mid x_i \mid \tag{4.13}$$

$$w_i = \begin{cases} 1 & if 0.25L \geq i \geq 0.75L \\ 0.5 & otherwise \end{cases}$$

- Modified mean absolute value 2 (MMAV2) is another extension of MAV feature by assigning the continuous weight window function, and it can be expressed as below [29].

$$MMAV2 = \frac{1}{L-1}\sum_{i=1}^{L-1} w_i \mid x_i \mid \tag{4.14}$$

$$w_i = \begin{cases} 1 & if 0.25L \geq i \geq 0.75L \\ 4i/L & if i \geq 0.25L \\ 4(i-L)/L & otherwise \end{cases}$$

- Maximum Fractional Length (MFL) Maximum fractal length (MFL) is a recent EMG feature used to measure the activation of low-level muscle contraction [29]. Mathematically, MFL can be defined as below,

$$MFL = log\left(\sqrt{\sum_{i=1}^{L-1}(x_{i+1} - x_i)^2}\right) \tag{4.15}$$

### *4.2 Enhanced Feature Extraction*

The following two features, Enhanced Mean Absolute Value (EMAV) and Enhanced Wavelength (EWL) provide better performance [37] for EMG signals classification. Since during each experiment the subject will have slow reaction at the early and final part of the move therefore most of the information in the signal is found within the middle region of the signal.To overcome the issues above, EMAV and EWL features are proposed. The proposed features are formulated as follows:

- Enhanced Mean Absolute Value (EMAV), [37]

$$EMAV = \frac{1}{L}\sum_{i=1}^{L} \mid (X_i)^p \mid \tag{4.16}$$

$$p = \begin{cases} 0.75 & i \geq 0.2L\& \\ & i \leq 0.8L \\ 0.5 & otherwise \end{cases}$$

- Enhanced Wavelength (EWL), [37]

$$EWL = \sum_{i=1}^{L} \mid (X_i - X_{i-1})^p \mid \tag{4.17}$$

$$p = \begin{cases} 0.75 & i \geq 0.2L\& \\ & i \leq 0.8L \\ 0.5 & otherwise \end{cases}$$

In the equations above, we use the parameter $p$ to identify the influence of the sample in the signal. In EMAV and EWL, a greater number of $p$ is set for 20% to 80% of regions. As mentioned since the center of the signal contains more information but increasing $p$ in that region we gather more information from the signal and this improves the quality of the features. Furthermore, Since EMAV and EWL are the extensions of MAV and WL with a simple modification, the computational time won't increase significantly.

## *4.3 Dimension Reduction and Classification*

After feature generation and we'll have a vector of features. The next step will be feature reduction which we'll the columns with most zeroes. Our goal here is to devise a plan to be able to distinguish each finger movement from another. Toward this goal, various range of implements can be utilized. Different effective classification methods can be used to satisfy our goal, yet as always we face different challenges such as Feature Selection, Correlation Analysis, and Classification Method.

### *4.3.1 Feature Selection Using Principal Component Analysis*

By observing results from feature extraction, we understand that the extracted features, although the best, yet are poorly separable. By this condition, no advanced classification method can distinguish different movements. However, by using Principal Component Analysis (PCA) we can combine the information in a manner to make them more separable and thus yield better results in classification [28]. PCA is used to decompose a multivariate dataset in a set of successive orthogonal components that explain a maximum amount of the variance. In scikit-learn, PCA is implemented as a transformer object that learns components in its fit method and can be used on new data to project it on these components. PCA centers but does not scale the input data for each feature before applying the SVD. The optional parameter whiten=True makes it possible to project the data onto the singular space while scaling each component to unit variance. This is often useful if the models down-stream make strong assumptions on the isotropy of the signal.

*Mathematical Realization of PCA*

So much for intuition in PCA, before we can implement PCA, we must know the mathematics behind this tool. Firstly, the data is centered around the mean value of the data. Then we shift the space by that value, mainly because we want our data to be set with zero mean, this will help us in both optimization and calculation of covariance. If our data set is in

matrix $A$ columned by vectors $X_i$. We "unlabeled" the whole dataset and see every column (feature column) as a dimension, thus every sample in this matrix (which is every row) is like a point in space, i.e. every row is a vector $x_i$.

$$\mu_i = \frac{1}{N}\sum_{i=1}^{i=N} x_i \ \mu = \left\{ \begin{array}{c} \mu_1 \\ \mu_2 \\ . \\ . \\ . \\ N \end{array} \right\} \tag{4.18}$$

$$A' = A - \mu \tag{4.19}$$

Then we must understand how spread out are the data. For this, we must calculate the covariance as well [1].

$$cov(X'_i, X'_j) = E[(X'_i)^T X'_j] \tag{4.20}$$

In which $E$ stands for the expected value that here is as same as statistical mean (Since the probability for each measurement is considered as same). This will yield us a $D \times D$ covariance matrix (D is the number of features). It's important to note that we are looking for axes with the highest variance. For that, we must transform data into a subspace which provide us such feature.

We can transform a complete square matrix into a fully diagonal one with the aid of its Eigen-Vectors and this transformation is an absolute dramatic turning point for us. It is worth mentioning that after Eigen-Vectors transformation, the dimensions are not our features anymore and it is a linear independent sum of each feature and we call each set of these sums, a component. When the covariance matrix is diagonal it means components do not correlate with each other and it is a solid proof for independence between all components. Thus for this transform,

$$A'V = A'' \tag{4.21}$$

At last, $A''$is the transformed data-set which is of highest variance and independence.

# Chapter 5

# FINGER MOVEMENT CLASSIFICATION

The result of our Signal processing and feature extraction is a data-set with the highest variance, which can be fed into classifiers for classification step. Six popular machine learning algorithms that we used for classification include Convolutional Neural Network (CNN), Deep Neural Network (DNN), k-Nearest Neighbor(KNN), Logistic regression (LR), Quadratic Discriminant Analysis (QDA), and XGBoost.

## *5.1 Deep Neural Network (DNN)*

Deep Neural Networks can almost model any complex function when used as a function approximator, but as a classifier the story is different. It must output multiple probabilities, each specified to a class. To find out to which class a sample point belongs to, we take the one output with the highest probability. [27]. The architecture of the neural network is,

$$input\ dim \times hidden\ layers \times number\ of\ classes \tag{5.1}$$

And the best loss function for this case is Cross Entropy Loss function,

$$L(t,s) = \sum_{i=1}^{C} t_i log(f(s_i)) \tag{5.2}$$

$t_i$ is the ground truth and $s_i$ is the score for every output, $f(.)$ is soft-max activation function.

$$f(x) = \frac{exp(x)}{\sum_{i=1}^{N} exp(x_i)} \tag{5.3}$$

Within layers, we used the ReLu activation function that is fit for classifications in networks and easy to take its derivative.

$$Relu(x) = \begin{cases} x & x > 0 \\ 0 & c \leq 0 \end{cases} \tag{5.4}$$

Considering the whole network as an independent summation of nonlinear terms, the training is nothing but to optimize the weights and biases of the network with the above loss function. That is to minimize the loss function with a numerical optimize, since finding an explicit solution to this problem is fairly impossible! The most popular numerical optimizer in neural networks is gradient descend. Its philosophy is simple; if you are looking for a summit, then go up. This philosophy may not lead us to the highest summit, but it will drive us to a local high point. However, our case is just the opposite in neural networks, that we are minimizing the loss function thus.

$$w_{n+1} = w_n - \alpha \nabla L \tag{5.5}$$

The rest of the process summarized in backpropagation. It is noteworthy to mention that usually data is associated with noise, this will destabilize the convergence for weights and biases making the network unstable. There is a various descendant from gradient descent optimization technique that will cover this issue. One of the most successful is Adaptive Moment Estimation (ADAM) which we used in our model[23] The advantages of this solution are numerous but it best to mention this quote from Kingma and Ba the authors of the algorithm, We have introduced a simple and computationally efficient algorithm for gradient-based optimization of stochastic objective functions. Our method is aimed at machine learning problems with large datasets and/or high-dimensional parameter spaces. The method combines the advantages of two recently popular optimization methods, the ability of AdaGrad to deal with sparse gradients, and the ability of RMSProp to deal with non-stationary objectives. The method is straightforward to implement and requires little memory. The experiments confirm the analysis of the rate of convergence in convex problems. Overall, we found Adam to be robust and well-suited to a wide range of non-convex optimization problems in the field machine learning.[23]

### *5.2 Convolutional Neural Network(CNN)*

Convolutional Neural Networks are very similar to ordinary Neural Networks, they have neurons with learnable weights and biases. Each neuron performs a dot product on its

input. The differentiable score function is the same from the matrix of features feeder to the classifier all the way to the class scores at the output. Loss function (such as SVM or Softmax) are used on the last layer which is fully- connected. Unlike a regular Neural Network, the neurons of each layer in CNN are arranged in 3 dimensions known as width, height, depth. where the additional dimension (depth) is not referring to the depth of the network but refers to the third dimension of an activation volume. Neurons in a layer are not connected to all neurons of the previous layer, they only connect to a small region of the layer. Additionally, the final output layer would be a vector of classes. Because in the final layer of CNN we will compress the full matrix into a single vector of class scores, arranged along the depth dimension.

### *5.3 K-Nearest Neighbor(KNN)*

Neighbors-based classifiers are non-generalizing learning where these models won't construct a general internal model, however, they store different instances of the training data. As a result by taking majority vote from the nearest neighbors of each point they make Classification [28][1]. Scikit-learn implements two different nearest neighbors classifiers. The k-neighbors classification is mostly used classifiers, its learning and decision making is based on the majority vote of $K$ nearest neighbors of each point. Radius Neighbors Classifier makes decision-based on the majority vote of neighbors within a radius $r$ of each training point. In both models, $k$ and $r$ are specified by the user. The optimal value for $k$ is based on the type of data. choosing a value for $k$ is a trade-off between suppressing the effects of noise and how distinct the classification boundaries are. Radius-based classification is a better choice when we have un-uniformed data samples where the user specifies a fixed radius r, this way when the neighborhood is sparser, fewer neighbors will fall into the radius and fewer points are used for the classification. However, when the number of classes increases or the feature space is more complex, this method becomes less effective due to the "curse of dimensionality". On the other hand, the basic $k$ nearest neighbor classification takes majority votes with uniform weights assigned to each query point. However, it is possible to weight the neighbors based

on their distances from the learning point this way the closer neighbors contribute more to the fit. Weight keyword is defaulted to uniform value, $weights =' uniform'$, and it can be changed to be proportional to the distance from the learning point $weights =' distance'$. In other words, KNN would be like, show me your friends and I will tell who you are.

### *5.4 logistic Regression (LR))*

Logistic regression is a fundamental classification technique. It belongs to the group of linear classifiers and is somewhat similar to polynomial and linear regression. Logistic regression is fast and relatively uncomplicated, and it's convenient for you to interpret the results. Although it's essentially a method for binary classification, it can also be applied to multiclass problems. we need an understanding of the sigmoid function 5.6 and the natural logarithm function $(ln(x), ln(1-x))$to understand what logistic regression is and how it works.

$$sigmoid(x) = \frac{1}{1+e^{-x}} \tag{5.6}$$

When we're implementing the logistic regression of some dependent variable $y$ in the set of independent variables $X = (x_1, x_2, ..., x_r)$. Where $r$ is the number of predictors (or inputs), we start with the known values of the predictors $x_i$ and the corresponding actual response (or output) $y_i$ for each observation $i = 1, ..., n$.

Our goal is to find the logistic regression function $p(x)$ such that the predicted responses $p(x_i)$ are as close as possible to the actual response $y_i$ for each observation $i = 1, 2, ..., n$. The actual response can be only 0 or 1 in binary classification problems. This means that each $p(x_i)$ should be close to either 0 or 1. That's why it's convenient to use the Sigmoid function. Once you have the logistic regression function $p(x)$ you can use it to predict the outputs for new and unseen inputs, assuming that the underlying mathematical dependence is unchanged.

### *5.5 Quadratic Discriminant Analysis (QDA)*

Linear Discriminant Analysis and Quadratic Discriminant Analysis are two classic classifiers which make decision base on the data surface distribution. These classifiers has very well practical use and work well in multi-class classifiers, also they have no hyper-parameters to tune[19]. QDA is an alternate version of Logistic Regression in which we consider a model for class probabilities and then find coefficients with regression[19]. However, QDA doesn't assume that the covariance of each of the classes is identical. Also, in order to estimate parameters, QDA is more data-hungry and needs more computation than LDA. QDA is the general form of Bayesian discrimination, in order to determine which variables discriminate between two or more naturally occurring groups it uses discriminant analysis. Therefore, when we want to assess the adequacy of classification; or we wish to assign objects to one of many (known) groups of objects DA is used. As a result, both Cluster Analysis or Principal Components Analysis are using DA to make better decisions.

### *5.6 XGBoost*

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. One of the major advantages of XGBoost is that it provides a parallel tree boosting (GBDT, GBM) that results in higher accuracy and faster computation time. With a regular machine learning model, like a decision tree, we'd simply train a single model on our dataset and use that for prediction. We might play around with the parameters for a bit or augment the data, but in the end, we are still using a single model. Even if we build an ensemble, all of the models are trained and applied to our data separately. Boosting, on the other hand, takes a more iterative approach. It's still technically an ensemble technique in that many models are combined to perform the final one, but takes a more clever approach. Rather than training all of the models in isolation of one another, boosting trains models in succession, with each new model being trained to correct the errors made by the

previous ones. Models are added sequentially until no further improvements can be made. The advantage of this iterative approach is that the new models being added are focused on correcting the mistakes which were caused by other models. In a standard ensemble method where models are trained in isolation, all of the models might simply end up making the same mistakes. Before running XGBoost, we must set three types of parameters: general parameters, booster parameters, and task parameters.

- General parameters relate to which booster we are using to do boosting, commonly tree or linear model
- Booster parameters depend on which booster you have chosen
- Learning task parameters decide on the learning scenario. For example, regression tasks may use different parameters with ranking tasks.
- Command line parameters relate to the behavior of CLI version of XGBoost

### *5.7 Results and Discussion*

We performed 3 sets of tests on the right hand of a 27 year old female, in each set we gathered 100 movement samples from each finger which gives us 500 samples per experiment. After 17 features are extracted from EMG signals we analyzed the model with and without applying PCA to the data-set. PCA transforms the data-set into a new subspace, which makes it more separable. Note that there are various hyper-parameters for PCA that must be set carefully. After components are received, we feed the data to classifiers and analyzed the performance using 5 fold cross-validation. For each data-set, we analyzed different configurations and classifiers. We perform standardization to all data sets because all data must be consistent within the same numerical range. If not, classification algorithms' performance will decline as the weights for the larger components will converge faster than smaller ones. Here, we go over the design and parameters of each classifier and then we compare the result of each classifier with and without using PCA.

### *5.7.1 Design and parameters of Convolutional Neural Network (CNN)*

in CNN each neuron performs a dot product on its input. The differentiable score function is the same from the matrix of features feeder to the classifier all the way to the class scores at the output. Loss function (such as SVM or Softmax) are used on the last layer which is fully- connected. Unlike a regular Neural Network, the neurons of each layer in CNN are arranged in 3 dimensions known as width, height, depth. where the additional dimension (depth) is not referring to the depth of the network but refers to the third dimension of an activation volume. Using Keras library we generated a six-layer CNN classifier with epochs=300 and batch size=8, we use LabelEncoder to increase the dimension of input and then put them into CNN. The first layer of the classifier is a one-dimensional convolution (conv1D). The number of output filters in the convolution is at 64, the length of the one-dimensional convolution window is set at 3 (kernel size = 3), activation function to use is Rectified Linear Unit (Relu), and the input shape where the parameters are based on the shape of input data. In our cases, $n_timesteps = 17$ and $n_features = 1$. The second layer is also conv1D with filters=64 and kernel size=3. The third layer is a dropout layer with a rate of 0.5. Dropout consists of randomly setting a fraction rate of input units to 0 at each update during training time, which helps prevent overfitting. The fourth layer is a pooling layer(max pooling operation for temporal data) with pool size (max pooling window)of 2. The fifth layer is a flatten layer that returns a variable reshaped to one dimension. The last layer is the Dense layer with normalizing exponential (softmax) as the activation function.

### *5.7.2 Design and parameters of Deep Neural Network (DNN)*

We have introduced a simple and computationally efficient algorithm for gradient-based optimization of stochastic objective functions. Our method is aimed at machine learning problems with large datasets and/or high-dimensional parameter spaces. The method combines the advantages of two recently popular optimization methods, the ability of AdaGrad to deal with sparse gradients, and the ability of RMSProp to deal with non-stationary objectives.

The method is straightforward to implement and requires little memory. The experiments confirm the analysis of the rate of convergence in convex problems. Overall, we found Adam to be robust and well-suited to a wide range of non-convex optimization problems in the field machine learning.[23]. Using Tensor Flow Keras sequential model we build a simple, fully-connected six-layer network. It's important to note that we Standardize (zero mean, unit variance) the data before feeding it to the network and after each layer, we used a batch normalization technique. The first four layers of the network are densely-connected layers with 128 units and Relu activation function. The fifth layer is densely-connected layers with 128 units and Softmax activation function. The final layer is the output layer with 5 units and the Softmax activation function.

### *5.7.3 design and parameters of Logistic regression (LR)*

Although LR is essentially a method for binary classification, it can also be applied to multiclass problems. When we're implementing the logistic regression of some dependent variable $y$ in the set of independent variables $X = (x_1, x_2, ..., x_r)$. Where $r$ is the number of inputs in (which is the set of features in our case), we start with the known values of the predictors $x_i$ and the corresponding actual response (or output) $y_i$ for each observation $i = 1, ..., n$. Our goal is to find the logistic regression function $p(x)$ such that the predicted responses $p(x_i)$ are as close as possible to the actual response $y_i$ for each observation $i = 1, 2, ..., n$. The actual response can be only 0 or 1 in binary classification problems. This means that each $p(x_i)$ should be close to either 0 or 1. That's why it's convenient to use the Sigmoid function. Using the Scikit-learn library we generate a logistic regression classifier with the following parameters. The inverse of regularization strength is set to its minimal value (1.0), smaller values specify stronger regularization. Fit intercept is set to be true which is intercepting our regression model with specifying a constant (a.k.a. bias or intercept) should be added to the decision function. intercept scaling has defaulted to 1, the Maximum number of iterations taken for the solvers to converge is set to 10000. The penalty has defaulted at 12n this variable is used to specify the norm used in the penalization. If 'none'

(not supported by the Liblinear solver), no regularization is applied. the solver is lbfgs since, for multi-class problems, lbfgs handles multinomial loss. The tolerance for stopping criteria is set at 0.0001.

### *5.7.4 design and parameters of Quadratic Discriminant Analysis (QDA)*

As mentioned QDA is an alternate version of Logistic Regression in which we consider a model for class probabilities and then find coefficients with regression. For the QDA, we used Scikit-learn library to generate a QDA classifier and for that matter we used the default scikit-learn parameters for training the classifier.

### *5.7.5 design and parameters of XGBoost*

The maximum depth of a tree is set to 5. Increasing this value will make the model more complex and more likely to overfit. We have to keep in consideration that XGBoost aggressively consumes memory when training a deep tree. $normalize_type$ is defaulted to where new trees have the same weight of each of dropped trees. Weight of new trees are $1/(k + learning_rate)$. Dropped trees are scaled by a factor of $k/(k + learning_rate)$ where $learning_rate$ is set to 0.01. The objective specifies the learning task and the corresponding learning objective. The objective is set to multi: Softmax, which sets XGBoost to do multi-class classification using the softmax objective. Silent is a parameter for verbosity, verbosity of printing messages. Valid values are 0 (silent), 1 (warning), 2 (info), 3 (debug). Sometimes XGBoost tries to change configurations based on heuristics, which is displayed as a warning message. $nthred$ is set to one which is the number of parallel threads used to run XGBoost. Gamma is set to its default at zero, which is a variable for minimum loss reduction required to make a further partition on a leaf node of the tree. The larger gamma is, the more conservative the algorithm will be. $min_child_weight$ is set to zero, it's a minimum sum of instance weight (hessian) needed in a child. If the tree partition step results in a leaf node with the sum of instance weight less than $min_child_weight$, then the building process will give up further partitioning. In linear regression task, this simply corresponds to the minimum number of

instances needed to be in each node. The larger $min_child_weight$ is, the more conservative the algorithm will be. $max_delta_step$ is set to the default value of zero, maximum delta step we allow each leaf output to be. If the value is set to 0, it means there is no constraint. If it is set to a positive value, it can help to make the update step more conservative. Usually, this parameter is not needed, but it might help in logistic regression when class is extremely imbalanced. Set it to the value of 1-10 might help control the update. The subsample ratio of the training instances is set to 0.7. Setting it to 0.5 means that XGBoost would randomly sample half of the training data before growing trees and this will prevent over-fitting. Subsampling will occur once in every boosting iteration. $colsample_bytree$ is set to 0.5 and is the subsample ratio of columns when constructing each tree. $colsample_bylevel$ has defaulted to 1 which is the subsample ratio of columns for each level. Columns are subsampled from the set of columns chosen for the current tree. $reg_alpha$ is set to 0.25 and $reg_lambda$ is set to 0.6 which are regularization term on weights. Increasing this value will make the model more conservative. $scale_pos_weight$ is set to 0.2 and it controls the balance of positive and negative weights, useful for unbalanced classes. Finally, we set the seed to 1440.

### *5.7.6 Discussion of results*

*Single Layer Classifier*

Using stratifiedKfold to split the whole data set using the fixed random seed(random state = 1), we ran 5 fold cross-validation on each model and recorded the average accuracy. At first, we applied PCA to transform the data-set into a new subspace which makes it more separable. Noting that there are various hyper-parameters for PCA that must be set carefully. After components are received, we feed the data to classifiers and do cross-validation. Figure 5.1 and 5.2 and 5.3 shows the average accuracy of our classifiers to the change in the number of PCA top components when the number of components changes from 6 to 12. It's important to notice that we removed the features that are zeroed out before running PCA, for these data sets the 5 features got removed and it reduced the total number of features to 12. Also,

table 5.1 and 5.2 and 5.3 are showing a comparison between the average accuracy of the six used classifier before and after using PCA. These tables show the results related to the first second and the third data sets respectively. As you can see the result for the data set from the first and the second tests show overall lower accuracy than the third data set. Additionally, CNN has the best performance in the first and second data set, on the other hand, QDA has the best performance in the third data. Also, we can see that in the first and third data set PCA has improved our accuracy, however on the second data set we have a better result without using PCA. To explain the difference of the results between the three data sets we need to plot the data based on the features to see how separable the data is. Figures 5.4 and 5.5 and 5.5 are the data sets before using PCA, we plotted the data based on the two features with the highest variance. As you can see the data in the first data set is very combined and inseparable, on the second data set the data is very scattered where the last test is more separable than the data in the first and second tests. Figure 5.7 and 5.8 and 5.9 are the data plotted after applying PCA and based on the top two components. As you can see even after applying PCA the first and the second data are very hard to separate. For the first data, we can see some level of improvement after applying PCA but it's hard to decide whether we had improvement of the second data or not (the classifier result shows that PCa not only didn't improve the separation of classes it also reduced the accuracy). On the other hand, we have a clear separation of classes on the third data which we can also see from the results. Therefore, Discriminant Analysis models such as QDA has good performance in such data sets. Looking at the result and distribution of the data in tests one and two we can see that the improvement of classifiers might not improve the result significantly. The best accuracy at the first data is at 76.9% and the best accuracy at the second data is at 73.2%. Considering the third data set the total accuracy among the three data sets is 82.5%.

.

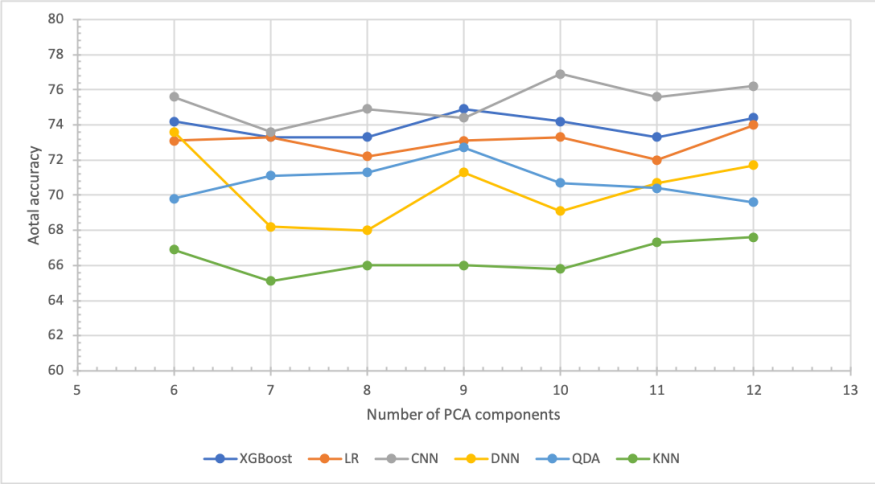



Figure 5.1: Average accuracy of 5 fold cross validation versus the number of PCA top components on the First test data

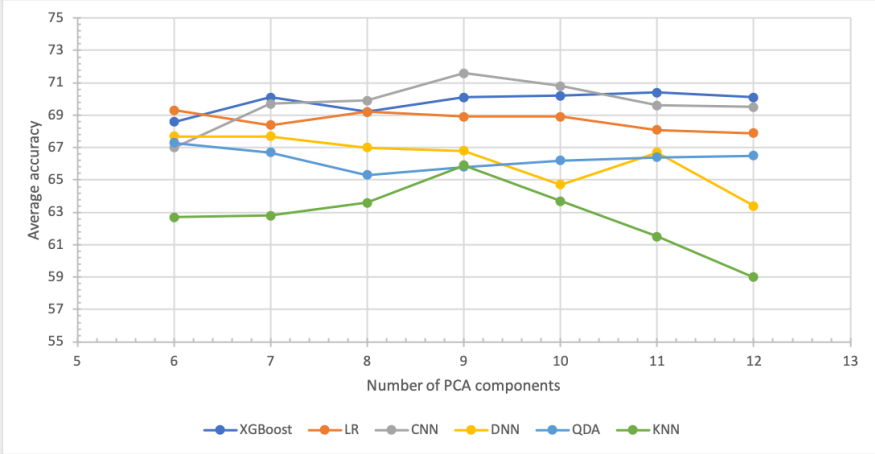



Figure 5.2: Average accuracy of 5 fold cross validation versus the number of PCA top components on the Second test data

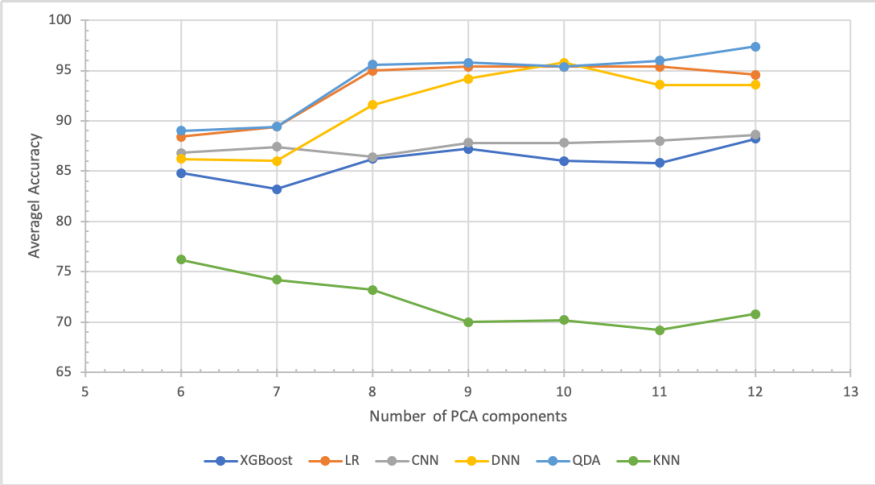



Figure 5.3: Average accuracy of 5 fold cross validation versus the number of PCA top components on the Third test data

| 5 class classifier | 5 fold cross validation total accuracy(%) | |
|---|---|---|
| Classifier | Without Using PCA | Using PCA |
| XGBoost | 74.2 | 74.9 |
| LR | 73.6 | 74 |
| CNN | 75.8 | 76.9 |
| DNN | 72.4 | 73.6 |
| QDA | 70 | 72.7 |
| KNN | 68.4 | 67.6 |

Table 5.1: Comparison of the classifiers results before and after applying PCA to the First data set

| 5 class classifier | 5 fold cross validation total accuracy(%) | |
|---|---|---|
| Classifier | Without Using PCA | Using PCA |
| XGBoost | 68 | 70.4 |
| LR | 70.5 | 69.3 |
| CNN | 73.2 | 71.6 |
| DNN | 67.4 | 67.7 |
| QDA | 65 | 67.3 |
| KNN | 64.3 | 65.9 |

Table 5.2: Comparison of the classifiers results before and after applying PCA to the Second data set

| 5 class classifier | 5 fold cross validation total accuracy(%) | |
|---|---|---|
| Classifier | Without Using PCA | Using PCA |
| XGBoost | 84.00 | 88.2 |
| LR | 90 | 95.4 |
| CNN | 89.8 | 88.6 |
| DNN | 75.95 | 95.8 |
| QDA | 76 | 97.4 |
| KNN | 78 | 76.2 |

Table 5.3: Comparison of the classifiers results before and after applying PCA to the Third data set

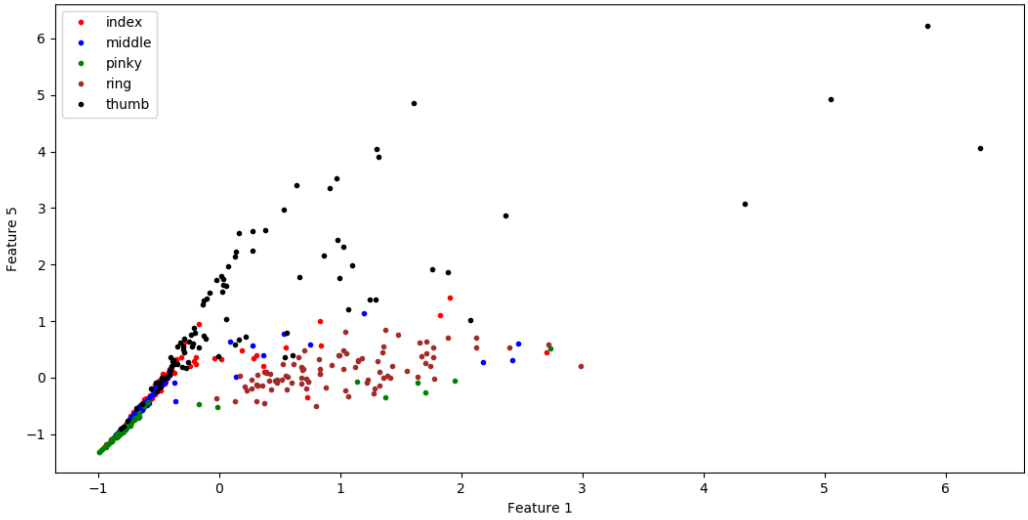



Figure 5.4: First data set color representation of different finger movements before applying PCA and based on top two high variance features

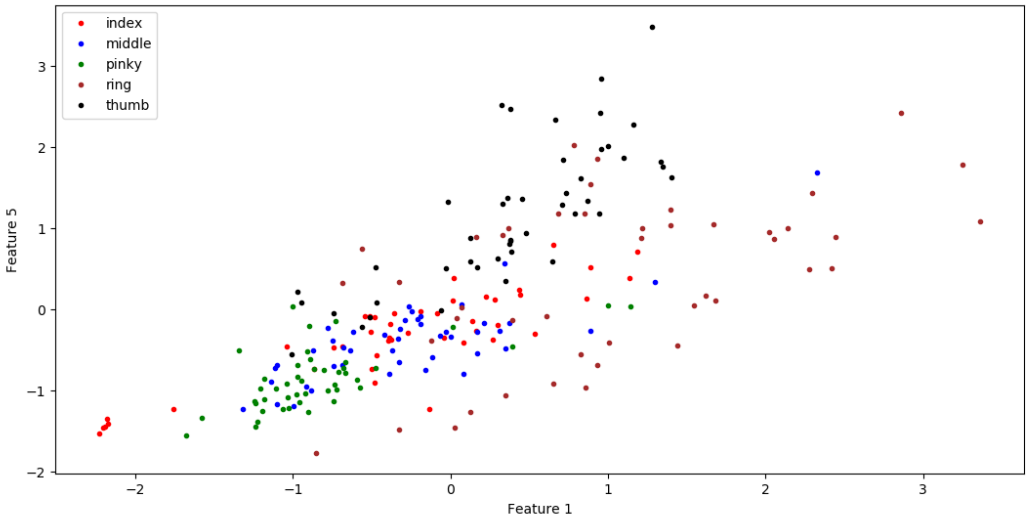



Figure 5.5: Second data set color representation of different finger movements before applying PCA and based on top two high variance features

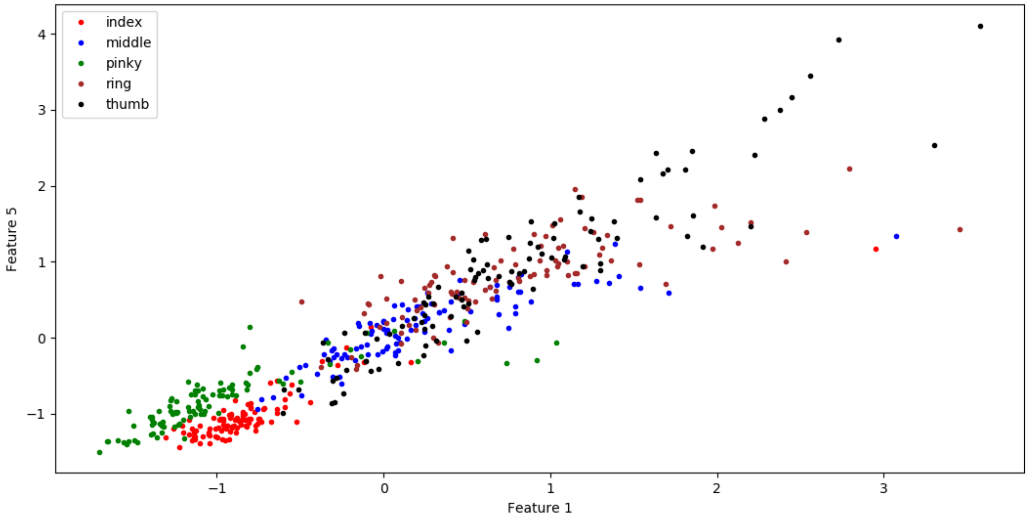



Figure 5.6: Third data set color representation of different finger movements before applying PCA and based on top two high variance features

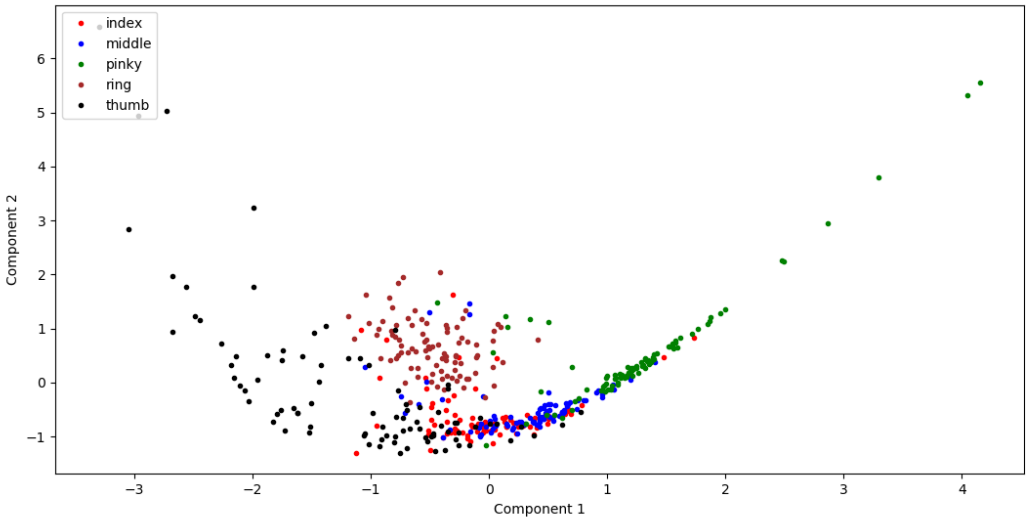



Figure 5.7: First data set color representation of different finger movements after applying PCA and based on top two PCA components

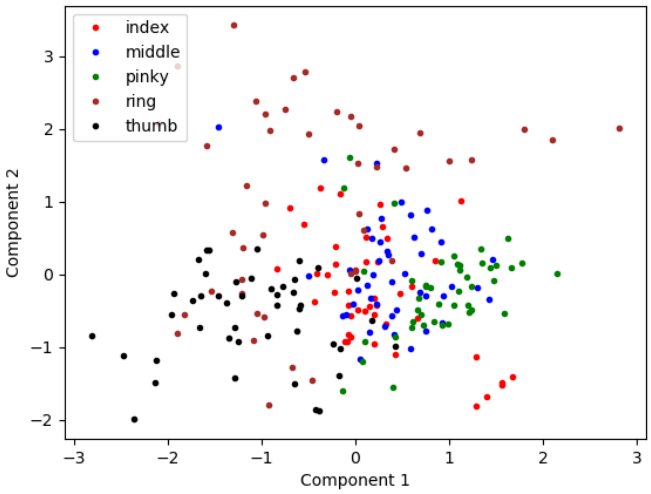



Figure 5.8: Second data set color representation of different finger movements after applying PCA and based on top two PCA components

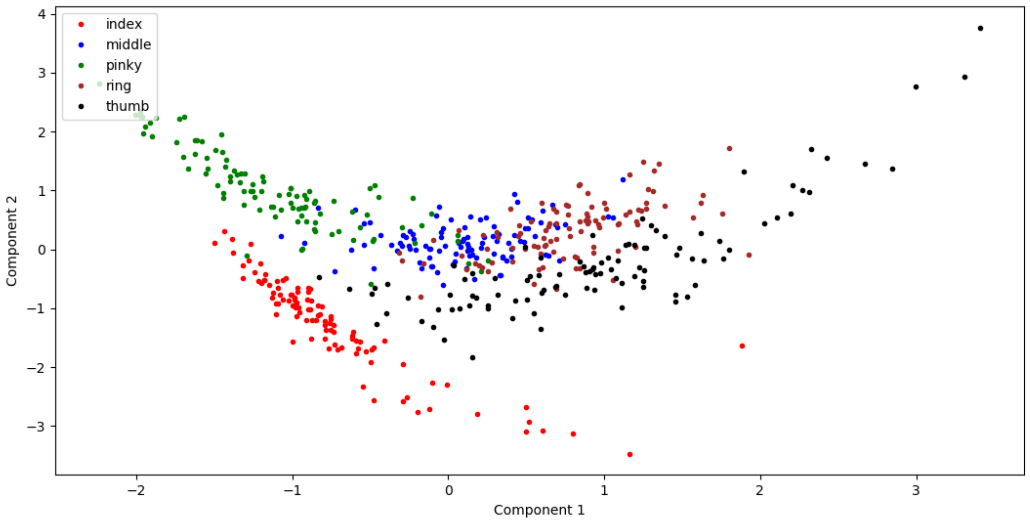



Figure 5.9: Third data set color representation of different finger movements after applying PCA and based on top two PCA components

*Two layer classifier*

Looking at the distribution of data before and after applying PCA, as well as the overall result of classifiers we look at alternative solutions for improving accuracy. It seems like if we eliminate the most misclassified class we might be able to improve the accuracy. Looking at the confusion matrix of different classifiers and all test we see that the Middle finger has the lowest accuracy and mostly misclassified. Additionally, looking at the distribution of the data we see that the middle finger has scattered in other classes. As a result, we decided to design a two-layer classifier, where at the first layer we have a binary classifier between the Middle finger and the rest of the fingers. At the second layer, we have 4 class classifiers between the rest of the fingers. Tables 5.4 and 5.5 and 5.6 shows the comparison of results for our tested classifiers on both layers of the classification system. As you can see the total accuracy for the first and second data has increased and brings the accuracy for these data sets to above 84% and the average accuracy among three data sets to 88.3%. Our next step to improve the accuracy as well as keep the system automated for any data set will be using voting systems such as modified ensembling methods which we are working on for our future studies.

## *5.8 Conclusion and future studies*

In this thesis report, we've aimed to design and implement an automated subject-specific finger movement extraction and detection system that would be able to extract finger movements from a series of movements and detect the finger movement classification. To achieve these goals not only we start by pre-processing the signal to extract each finger movement but also we focused on being able to detect the length of the moves in TD and generate a data set for each set of movements for feature extraction and classification. We introduced a novel model to extract each movements and their duration. Also, 17 EMG features are proposed which aims to enhance the prediction accuracy for the classification of hand movements. Initially, the proposed features are extracted from the extracted movements in time domain.

| Binary Classifier | 5 fold cross validation | | 4 class classifier | 5 fold cross validation | |
|---|---|---|---|---|---|
| Classifier | Without Using PCA | Using PCA | Classifier | Without Using PCA | Using PCA |
| XGBoost | 83.3 | 82.4 | XGBoost | 81.6 | 82.9 |
| LR | 79.6 | 79.6 | LR | 82.9 | 80.8 |
| CNN | 83.3 | 84.9 | CNN | 81.6 | 82.9 |
| DNN | 80.6 | 83.6 | DNN | 78.9 | 80.3 |
| QDA | 67.8 | 67.8 | QDA | 73 | 76 |
| KNN | 84.8 | 75.3 | KNN | 80.3 | 84.5 |

Table 5.4: Comparison of the two layers classifiers results before and after applying PCA to the First data set

| Binary Classifier | 5 fold cross validation | | 4 class classifier | 5 fold cross validation | |
|---|---|---|---|---|---|
| Classifier | Without Using PCA | Using PCA | Classifier | Without Using PCA | Using PCA |
| XGBoost | 80.9 | 81.2 | XGBoost | 81.7 | 83.1 |
| LR | 79.7 | 79.7 | LR | 81.7 | 82.4 |
| CNN | 84.9 | 82.8 | CNN | 84.1 | 81.3 |
| DNN | 80 | 80.4 | DNN | 80.7 | 81.9 |
| QDA | 79.7 | 62.2 | QDA | 79 | 77.8 |
| KNN | 75.4 | 79.9 | KNN | 74.8 | 74.8 |

Table 5.5: Comparison of the two layers classifiers results before and after applying PCA to the Second data set

| Binary Classifier | 5 fold cross validation | | 4 class classifier | 5 fold cross validation | |
|---|---|---|---|---|---|
| Classifier | Without Using PCA | Using PCA | Classifier | Without Using PCA | Using PCA |
| XGBoost | 84.2 | 85.6 | XGBoost | 91.2 | 92.7 |
| LR | 98 | 98.2 | LR | 95.75 | 94.25 |
| CNN | 89.8 | 88.4 | CNN | 96 | 96.5 |
| DNN | 80.6 | 98 | DNN | 81.75 | 95.75 |
| QDA | 85 | 97 | QDA | 89 | 95.75 |
| KNN | 84.9 | 83 | KNN | 88.6 | 83.5 |

Table 5.6: Comparison of the two layers classifiers results before and after applying PCA to the Third data set

We're able to detect over 97% of the moves, however, due to noise factors and human error, it's hard to choose a ground rule for analyzing the length and accuracy of the detected length of the movements. After feature extraction we performed Principal Component analysis on the extracted features and fed them into the machine learning algorithm for classification process. we've shown that the quality of the received signal has an important effect on the result of our classifier, and their behavior towards changes. Finally, We proposed a two-layer classifier where at the first layer we have a binary classifier to detect the middle finger (the most misclassified finger in single layer classifier) from the rest of the fingers. We were able to achieve an average accuracy of 88.3%.

### *5.8.1 Future studies*

There are several areas that we have room for improvement. Based on our current results we can confidently claim that this work can be very successful in application-specific HCI

needs. One of our areas of interest is the use of this model for HCI security models, for instance, we can use this method to recognize coded finger movements for security purposes, or it can be used for applications such as drone control. Additionally, our study shows the importance of the quality of the received signal, this quality can be affected by the location of the signal, the way the fingers are moved, the physical situation of the test subject and external factors of this type. As we add more standardization to the test environment, test duration and installation of the signals we realized the quality of can be improved in such circumstances. We have room for improving our protocol for testing and data recording. Therefore, for our future studies, one of our focuses will be on capturing high-quality signals. We also like to focus on performing the tests in a way that we can put a ground-rule on our duration detection and analyze our model accuracy for movement extraction. Additionally, by improving the gathered data we can have better subject specified classifications. Finally, since classifiers' behavior changes based on the received signal we'd like to continue this work by focusing on voting models such as Ensemble, this would improve automating the finger detection system.

# BIBLIOGRAPHY

[1] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[2] Yousef Al-Assaf. Surface myoelectric signal analysis: Dynamic approaches for change detection and classification. *IEEE Transactions on biomedical engineering*, 53(11):2248–2256, 2006.

[3] Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

[4] Alex Andrews, Evelyn Morin, and Linda McLean. Optimal electrode configurations for finger movement classification using emg. In *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 2987–2990. IEEE, 2009.

[5] Claudio Castellini and Patrick van der Smagt. Surface emg in advanced hand prosthetics. *Biological cybernetics*, 100(1):35–47, 2009.

[6] Adrian DC Chan and Kevin B Englehart. Continuous myoelectric control for powered prostheses using hidden markov models. *IEEE Transactions on Biomedical Engineering*, 52(1):121–124, 2004.

[7] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[8] J-U Chu, Inhyuk Moon, and M-S Mun. A real-time emg pattern recognition system based on linear-nonlinear feature projection for a multifunction myoelectric hand. *IEEE Transactions on biomedical engineering*, 53(11):2232–2239, 2006.

[9] Carlo J De Luca, L Donald Gilmore, Mikhail Kuznetsov, and Serge H Roy. Filtering the surface emg signal: Movement artifact and baseline noise contamination. *Journal of biomechanics*, 43(8):1573–1579, 2010.

[10] Kevin Englehart, B Hudgin, and Philip A Parker. A wavelet-based continuous classification scheme for multifunction myoelectric control. *IEEE Transactions on Biomedical Engineering*, 48(3):302–311, 2001.

[11] Kevin Englehart and Bernard Hudgins. A robust, real-time control scheme for multifunction myoelectric control. *IEEE transactions on biomedical engineering*, 50(7):848–854, 2003.

[12] Kevin Englehart, Bernard Hudgins, and Adrian DC Chan. Continuous multifunction myoelectric control using pattern recognition. *Technology and disability*, 15(2):95–103, 2003.

[13] Kevin Englehart, Bernard Hudgins, Philip A Parker, and Maryhelen Stevenson. Classification of the myoelectric signal using time-frequency based representations. *Medical engineering & physics*, 21(6-7):431–438, 1999.

[14] Kristin A Farry, Ian D Walker, and Richard G Baraniuk. Myoelectric teleoperation of a complex robotic hand. *IEEE Transactions on Robotics and Automation*, 12(5):775–788, 1996.

[15] Guillaume Gaudet, Maxime Raison, and Sofiane Achiche. Classification of upper limb phantom movements in transhumeral amputees using electromyographic and kinematic features. *Engineering Applications of Artificial Intelligence*, 68:153–164, 2018.

[16] Yikun Gu, Dapeng Yang, Qi Huang, Wei Yang, and Hong Liu. Robust emg pattern recognition in the presence of confounding factors: features, classifiers and adaptive learning. *Expert Systems with Applications*, 96:208–217, 2018.

[17] Yonghong Huang, Kevin B Englehart, Bernard Hudgins, and Adrian DC Chan. A gaussian mixture model based classification scheme for myoelectric control of powered upper limb prostheses. *IEEE Transactions on Biomedical Engineering*, 52(11):1801–1811, 2005.

[18] Bernard Hudgins, Philip Parker, and Robert N Scott. A new strategy for multifunction myoelectric control. *IEEE Transactions on Biomedical Engineering*, 40(1):82–94, 1993.

[19] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[20] Bekir Karlik, M Osman Tokhi, and Musa Alci. A fuzzy clustering neural network architecture for multifunction upper-limb prosthesis. *IEEE Transactions on Biomedical Engineering*, 50(11):1255–1261, 2003.

[21] Rami N Khushaba, Maen Takruri, Jaime Valls Miro, and Sarath Kodagoda. Towards limb position invariant myoelectric pattern recognition using time-dependent spectral features. *Neural Networks*, 55:42–58, 2014.

[22] Kang Soo Kim, Heung Ho Choi, Chang Soo Moon, and Chi Woong Mun. Comparison of k-nearest neighbor, quadratic discriminant and linear discriminant analysis in classification of electromyogram signals based on the wrist-motion directions. *Current applied physics*, 11(3):740–745, 2011.

[23] DP Kingma and JL Ba. Adam optimizer. *arXiv preprint arXiv:1412.6980*, pages 1–15, 2014.

[24] Vijay R Mankar. Emg signal noise removal using neural netwoks. In *Advances in Applied Electromyography*. IntechOpen, 2011.

[25] Kianoush Nazarpour, Ahmad R Sharafat, and S Mohammad P Firoozabadi. Application of higher order statistics to surface electromyogram signal classification. *IEEE Transactions on Biomedical Engineering*, 54(10):1762–1769, 2007.

[26] Andrew Y Ng and Michael I Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems*, pages 841–848, 2002.

[27] Michael A Nielsen. *Neural networks and deep learning*, volume 2018. Determination press San Francisco, CA, USA:, 2015.

[28] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[29] Angkoon Phinyomark, Pornchai Phukpattaranont, and Chusak Limsakul. Feature reduction and selection for emg signal classification. *Expert systems with applications*, 39(8):7420–7431, 2012.

[30] Oluwarotimi Williams Samuel, Hui Zhou, Xiangxin Li, Hui Wang, Haoshi Zhang, Arun Kumar Sangaiah, and Guanglin Li. Pattern recognition of electromyography signals based on novel time domain features for amputees' limb motion classification. *Computers & Electrical Engineering*, 67:646–655, 2018.

[31] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[32] Wan-Ting Shi, Zong-Jhe Lyu, Shih-Tsang Tang, Tsorng-Lin Chia, and Chia-Yen Yang. A bionic hand controlled by hand gesture recognition based on surface emg signals: A preliminary study. *Biocybernetics and Biomedical Engineering*, 38(1):126–135, 2018.

[33] A Silva and Antonie Stam. Discriminant analysis. 1995.

[34] Francesco VG Tenore, Ander Ramos, Amir Fahmy, Soumyadipta Acharya, Ralph Etienne-Cummings, and Nitish V Thakor. Decoding of individuated finger movements using surface electromyography. *IEEE transactions on biomedical engineering*, 56(5):1427–1434, 2008.

[35] Dennis Tkach, He Huang, and Todd A Kuiken. Study of stability of time-domain features for electromyographic pattern recognition. *Journal of neuroengineering and rehabilitation*, 7(1):21, 2010.

[36] Jingwei Too, Abdul Rahim Abdullah, Norhashimah Mohd Saad, and Weihown Tee. Emg feature selection and classification using a pbest-guide binary particle swarm optimization. *Computation*, 7(1):12, 2019.

[37] Jingwei Too, Abdul Rahim Abdullah, and Norhashimah Mohd Saad. Classification of hand movements based on discrete wavelet transform and enhanced feature extraction.

[38] G Tsenov, AH Zeghbib, F Palis, N Shoylev, and V Mladenov. Neural networks for online classification of hand and finger movements using surface emg signals. In *2006 8th Seminar on Neural Network Applications in Electrical Engineering*, pages 167–171. IEEE, 2006.

[39] Noriyoshi Uchida, Akira Hiraiwa, Noboru Sonehara, and Katsunori Shimohara. Emg pattern recognition by neural networks for multi fingers control. In *1992 14th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, volume 3, pages 1016–1018. IEEE, 1992.

[40] Md Ferdous Wahid, Reza Tafreshi, Mubarak Al-Sowaidi, and Reza Langari. Subject-independent hand gesture recognition using normalization and machine learning algorithms. *Journal of computational science*, 27:69–76, 2018.

[41] Woon-Hong Yeo, Yun-Soung Kim, Jongwoo Lee, Abid Ameen, Luke Shi, Ming Li, Shuodao Wang, Rui Ma, Sung Hun Jin, Zhan Kang, et al. Multifunctional epidermal electronics printed directly onto the skin. *Advanced Materials*, 25(20):2773–2778, 2013.