**University of Victoria**

**Faculty of Engineering**

**Fall 2019 ENGR 446 Final Report**

# Gesture Classification of sEMG Signals for Prosthesis Control

**University of Victoria**

**Faculty of Engineering**

**Victoria, British Columbia**

**Thomas Mastromonaco**

**V00820992**

**Department of Electrical and Computer Engineering**

**tmastrom@uvic.ca**

**November 29th 2019**

**In partial fulfillment of the requirements of the B.Eng. Degree**

**Letter of Transmittal**

University of Victoria

Faculty of Engineering

Re: ENGR 446 Report

Monday, November 25, 2019

Dear Ash Senini:

Please accept the following report titled "Gesture Classification of sEMG Signals for Prosthesis Control" in partial fulfilment of the requirements of the ENGR 446 course at the University of Victoria. This report outlines the engineering problem of classifying gestures based on signals collected from the surface of muscles for the control of a prosthesis. The proposed solution is to use a supervised machine learning model called a Support Vector Classifier. This report discusses the processing and classification results of the model based on previously recorded data from the Myo armband but does not include the data collection task. The original scope of the report was to discuss the feasibility of implementing this model in an embedded system, but further work is required.

Thank you to Ash Senini and Fatima Talebzadeh for their support and feedback throughout this project.

Regards,

*Thomas Mastromonaco*

Term 4B Computer Engineering

## Table of Contents

## Table of Figures

## Glossary

ANN – Artificial Neural Network

CPU – Central Processing Unit

DWT – Discrete Wavelet Transform

GB – Gigabyte

KB – Kilobyte

KNN – K Nearest Neighbours

LDA – Linear Discriminant Analysis

MHz – Megahertz

RAM – Random Access Memory

RBF – Radial Basis Function

RF – Random Forest

sEMG – Surface Electromyography

SVC – Support Vector Classifier

## Summary

A major hurdle for building effective prostheses is the control interface. Patients will not find the device useful unless they are able to accurately and intuitively control the device. The current industry standard is to capture the muscle signals in the residual limb using surface electromyography (sEMG) electrodes. These signals must be accurately deciphered by the device for the prosthesis to perform the intended action.

This report attempts to define a supervised machine learning model called a Support Vector Classifier to accurately classify gestures based on sEMG signals. The classifier was trained on a labelled dataset consisting of four classes. Each class represents a hand gesture (rock, paper, scissors, ok).

The time series sEMG data was preprocessed using the Daubechies discrete wavelet transform to characterize the waveform before fitting the data to the classifier. After fitting the model and optimizing the parameters, 92% accurate classification was achieved on the test dataset. This is on par with current prosthetic device classification accuracy. Further testing is required to determine if the program can be executed in real-time (less than 75ms) on a typical embedded processor that would be used in a prosthesis today.

## Introduction

One of the largest problems with current bionic prostheses is quick and intuitive control. The current solutions use electrical sensors in the prosthesis to read surface electromyography (sEMG) signals from the muscles in the residual limb. The focus of this report will be the control of trans-radial (below the elbow) upper limb prostheses as shown in Figure 1 below. After the sensors have collected the sEMG from the forearm of the subject, these signals are classified and mapped to movements of the prosthesis. Accurate classification of gestures, based on the collection of sEMG data using sensors in the prosthesis, is very important for patients to find the device useful. If the patient cannot reliably make the device work as they intend or the device responds too slowly, it will not be useful.

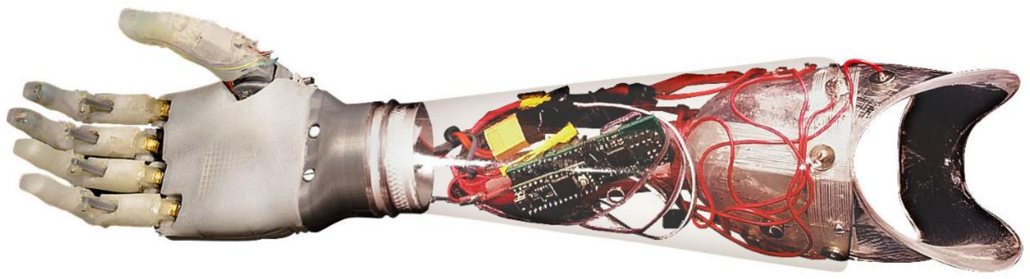

*Figure 1 Psyonic Upper Limb Prosthesis [1]*

The use of sEMG signals for the control of prosthesis is one of the earliest methods for myoelectric control due to sensors being non-invasive and relatively low cost. Early iterations of myoelectric prostheses mapped the impulse from a muscle at a site mapped to a function of the prosthesis which lead to unintuitive control.

It is only recently that the use of pattern recognition has been feasible due to advances in microprocessor and sensor technology. Collecting sEMG signals at multiple sites and using pattern recognition to distinguish between gestures allows for much more intuitive control. Pattern recognition techniques require an initial calibration in which the system learns to associate different hand gestures to different myoelectric patterns based on the phantom limb effect. This association is then adopted in the daily use of the prosthesis [2]. In January 2015, COAPT released the first commercial prosthesis control device based on pattern recognition and continues to innovate.

The dataset to be utilized in this report is based on sEMG data collected from the Myo armband which is shown in Figure 2. The use of this device is prevalent in prosthetic control research due to its 8 non-invasive sensors and the fact that it is commercially available [3]. The Myo armband sensors are dry and the device can be easily slipped on by patients. This does have performance drawbacks as dry electrodes have worse performance than gel-based electrodes [4].

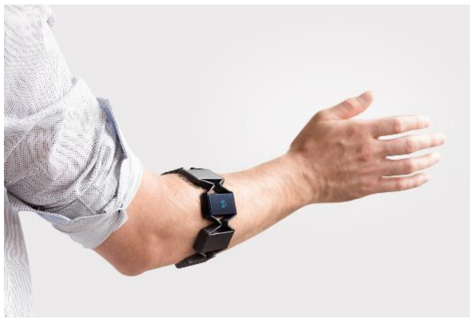

*Figure 2 Myo Armband [5]*

## Problem Definition

The classification of gestures based on sEMG signals in the residual limb for intuitive control is becoming possible due to advances in machine learning and microprocessor technology, but the problem is far from solved. The problem is computationally intensive and human variability makes the challenge even more difficult. The purpose of this report is to use machine learning to classify gestures based on forearm EMG data. The dataset was created by taking readings using the Myo armband while the patients hold gestures. The dataset consists of four classes, labelled ‘rock’, ‘paper’, ‘scissors’, ‘ok’. These gestures are shown in the Figure below.

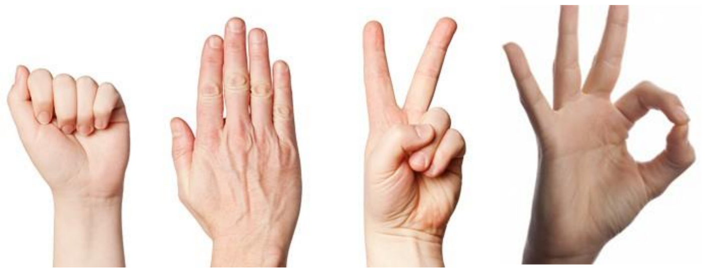

*Figure 3 Gesture Classes [6]*

The main goal is to build a classifier that satisfies the following requirements:

1. Accurately classifies gestures (>91% accuracy)
2. Classifies gestures in real-time (<75ms)
3. Can be run on embedded systems

The performance of the classifier will be evaluated by splitting the dataset into training and testing sets. Validation will not be performed on subjects with a Myo armband. It will be assumed that the data collection process was accurate. The feasibility of running the classifier on an embedded system will be evaluated theoretically using a typical microcontroller to be specified later. Speed of execution will be calculated based on the processing power of an embedded processor.

## Discussion

The next section will contain subsections of how the classifier was chosen, designed and implemented. The first subsection will give an overview of different machine learning models and how an SVC model was chosen for the task of classifying the gestures. Next the SVC design, implementation, optimization and results will be discussed.

### Technical Review

The five most common classifiers employed for sEMG pattern recognition: Support Vector Machine (SVM), Artificial Neural Networks (ANN), Random Forest (RF), K-Nearest Neighbors (KNN) and Linear Discriminant Analysis (LDA) [3]. All these classifiers fall into the category of supervised machine learning. This means that the model is trained on a labelled dataset. These classifiers have trade-offs in when it comes to accuracy, computational complexity, transparency and implementation complexity. These factors will be used to evaluate the best classifier for this application in the next section.

The simplest of the previously mentioned classifiers is Linear Discriminant Analysis (LDA). As the name suggests, it uses a linear decision boundary to group the data into classes. A linear model is relatively simple to implement with low computational complexity but can have strong biases. K-Nearest Neighbours (KNN) is also a relatively simple classifier in which a new data point is classified based which data points in the training set it is closest to in vector space. KNN is robust to noisy data but the computation cost is high because the point to be classified must be compared with all the others to find the closest ones. SVM classifiers are memory efficient and can have linear or non-linear decision boundaries. One of the drawbacks of using an SVM is the classification is binary, not probabilistic. An object is only classified into one category and none of the others are considered. A Random Forest classifier is another good option, but it is computationally intensive and can be a black box (little is known about its inner workings). An Artificial Neural Network is completely different from the other classifiers previously discussed. An ANN uses nodes connected by edges with different weights. An ANN has multiple layers of connected nodes. At minimum there is an input, hidden and an output layer, however, there can be many more hidden layers. The input to each node is transformed by some function and multiplied by some weight before travelling to another node. ANNs have the highest implementation and computational complexity.

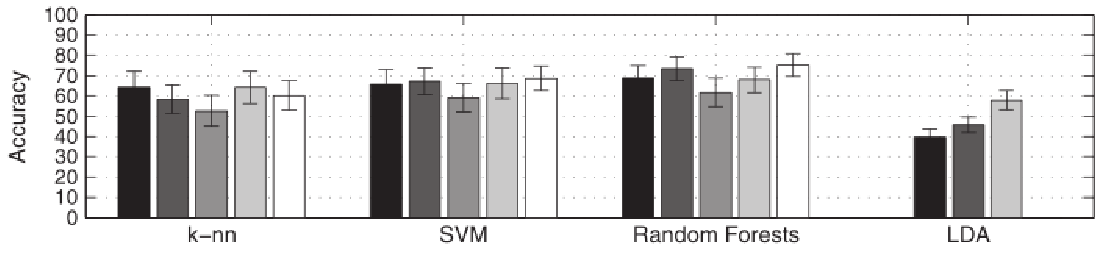

*Figure 4 Accuracy of Different Classifiers with Different Feature Sets [7]*

The accuracy of each classifier was estimated based on a comparison of these methods applied to sEMG data across different feature sets in Figure 4 above. The SVM and RF classifiers have the highest accuracy when applied to this type of problem.

*Table 1 Decision Matrix*

| | **Score:** | **21** | **20** | **15** | **17** | **17** |
|---|---|---|---|---|---|---|
| | **Classifier:** | SVM | ANN | RF | KNN | LDA |
| **Categories** | **Weight** | Suitability (0 - 3) | Suitability (0 - 3) | Suitability (0 - 3) | Suitability (0 - 3) | Suitability (0 - 3) |
| Accuracy | 3 | 2 | 3 | 2 | 1 | 0 |
| Dataset | 1 | 2 | 3 | 3 | 3 | 2 |
| Computational Complexity | 2 | 2 | 1 | 2 | 1 | 3 |
| Interpretable Results | 2 | 3 | 2 | 1 | 3 | 3 |
| Ease of Implementation | 1 | 3 | 2 | 0 | 3 | 3 |

A decision matrix was used to evaluate the five potential algorithms based on relevant factors to the classification problem outlined previously. The most important factors for the real-time classification of gestures for the control of prostheses are speed (computational complexity) and accuracy. If the subject is not able to reliably control the device and do so in a humanly fast manner, the device is not helpful. The importance of these factors is shown in the rank column of Table 1 above. Based on the results of the decision matrix, the classifier will be implemented using an SVC model.

## SVC Design

A Support Vector Machine or Support Vector Classifier (SVC) will be created to solve the problem of classifying gestures based on forearm sEMG data in real-time. This approach was chosen based on the decision matrix analysis performed in the previous section of this report. An SVC is a method of supervised machine learning which means it is trained using labelled data. The SVC finds a line or hyperplane which separates data into categories [8]. Then new data can be categorized based on where it falls relative to the separating line. Figure 5 illustrates this idea. For this project, the dataset has much higher dimensionality and the dividing line will be a hyperplane.

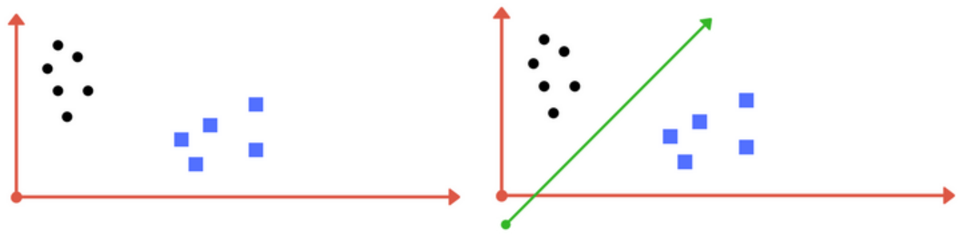

*Figure 5 Support Vector Classifier [9]*

The SVC model will be built in Python using Scikit Learn, numpy and pandas libraries. The labelled sEMG dataset [10] will be split into training and testing subsets containing a random 75% and 25% of the samples respectively. The model will be trained using the training set and the performance of the classifier will be evaluated using the testing set. The SVC model will be trained using feature vectors. The feature vectors will be created by performing the Daubechies wavelet transform to obtain coefficients which approximate the sEMG waveform data. Next, the SVC model will be tuned by experimenting with different parameters such as the kernel function.

The model will then be evaluated based on its accuracy, classification speed and computational complexity. These areas of evaluation are important for assessing the feasibility of using this classifier in a real-time prosthetic device. Accuracy is important for the usefulness of the device to the patient and computational complexity will dictate how fast the classification occurs in an embedded system with limited processing power. The goal is to create a classifier with at least 91% classification accuracy and complete the computation in under 75ms as per [11].

## Data Preprocessing

The dataset contains sEMG data from eight sensors placed on the subject's forearm as they create four distinct shapes with their hand. Each dataset line has 8 consecutive readings of all 8 sensors so 64 columns of EMG data [10]. The Myo armband records samples at 200 Hz, therefore each wave corresponds to 40 ms of time. The following Figure helps visualize what the data means. The following graphs show the mean waveform for each class.

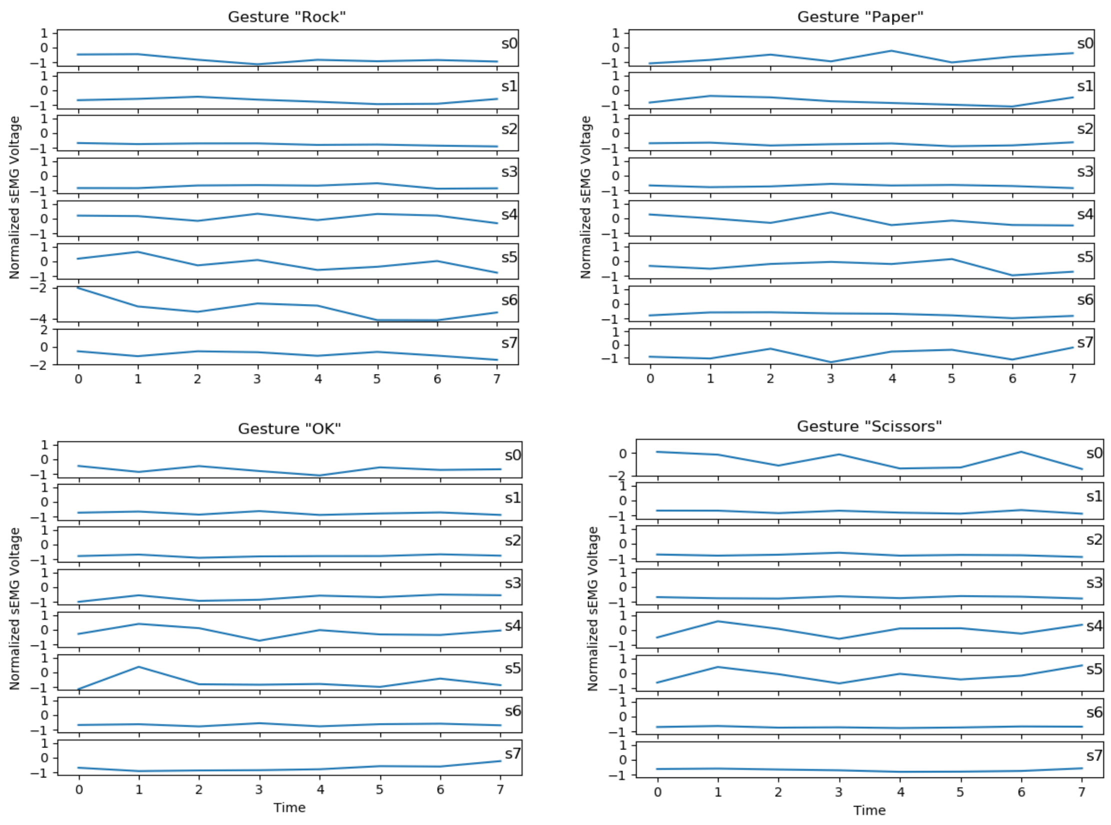

*Figure 6 Dataset Visualization*

The next step is to extract the features of the data. This preprocessing step helps to highlight the important differences between each category of data. The current best practice for feature extraction of sEMG signals is using a wavelet transform [12].  Wavelet transform methods can be discrete or continuous. The discrete wavelet transform (DWT) was selected because it is much less computationally expensive and better for real-time engineering applications. DWT is a technique that iteratively transforms an interested signal into multi-resolution subsets of coefficients as shown in Figure 7.

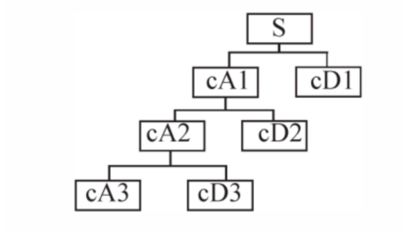

*Figure 7 Wavelet Transform [12]*

The first order Daubechies Wavelet coefficients were calculated for each sensor waveform over 40ms and used as feature vectors. More info about the Daubechies Wavelet transform [13] can be found in Appendix 1.

After creating the feature vectors, they are standardized [14] This is a very important step as most machine learning algorithms only work properly on a centered dataset.

## Classifier Parameter Tuning

Next the SVC class from the Scikit Learn library [15] was utilized to create the classifier. The SVC class has many parameters to be tuned. The main ones that will be discussed in this report are the kernel, penalty parameter, degree and gamma.

### Kernel

The SVC Kernel is the function used to draw the separating line or hyperplane. The SVC class has linear, polynomial or radial basis function (RBF) kernel options. Linear and polynomial functions can be visualized simply with the following Figures respectively. The degree of the polynomial line can also be set.

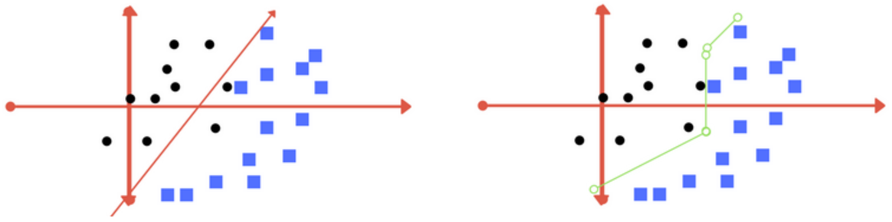

*Figure 8 SVC Polynomial Kernel [10]*

A RBF is a real-valued function whose value depends only on the distance between the input and some fixed point, either the origin, so that the RBF kernel on two samples $x^{(i)}$ and $x^{(j)}$, represented as feature vectors in some input space, is defined as

$$\begin{aligned} K\left(x^{(i)}, x^{(j)}\right) &= \phi(x^{(i)})^T \phi(x^{(j)}) \\ &= \exp\left(-\gamma \left\| x^{(i)} - x^{(j)} \right\|^2\right), \qquad \gamma > 0 \end{aligned}$$

***Eq. 1***

Since the value of the RBF kernel decreases with distance and ranges between zero (in the limit) and one (when $x^{(i)} = x^{(j)}$), it has a ready interpretation as a similarity measure [11]. The RBF function is excellent at handling data with non-separable features with high dimensions [16].

### Penalty Parameter

The penalty parameter, C, of the error term controls the trade off between smooth decision boundary and classifying the training points correctly. A low C value gives a smooth decision boundary with a high margin whereas a high C value tries to classify the training data correctly. It should also be noted that smaller values of C encourage a larger margin whereas larger values of C allow for a smaller margin [17]. C is only relevant to the RBF kernel. The effect of varying the C parameter can be seen in the following Figure.

The classifier achieves 92.36% accuracy when C = 10 and accuracy appears to plateau if C is increased beyond this value.

*Table 2 C Values*

| C | Accuracy(%) |
|---|---|
| 0.1 | 71.92 |
| 1 | 89.79 |
| 10 | 92.36 |
| 100 | 91.4 |
| 1000 | 91.37 |

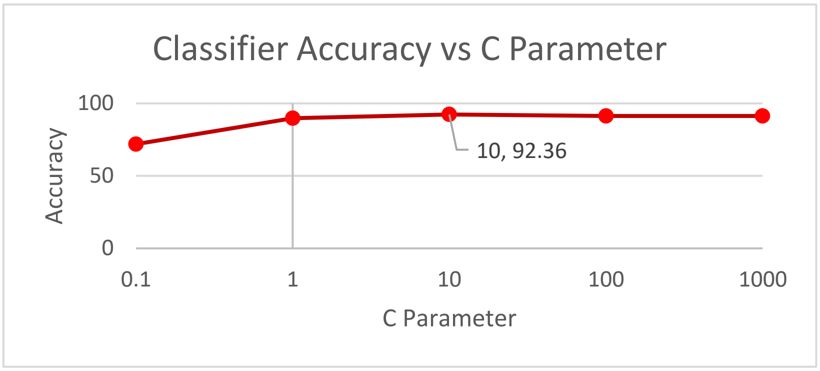

*Figure 9 Accuracy vs C*

## Gamma

The gamma parameter defines how much weight a single training example has based on its distance from the decision boundary, with low values meaning ‘far’ and high values meaning ‘close’. The gamma parameter is the inverse of the radius of influence of samples selected by the model as support vectors [17]. Gamma is only relevant to the RBF kernel. The effect of varying the gamma parameter can be seen in Figure 10.

*Table 3 Gamma Values*

| Gamma | Accuracy(%) |
|---|---|
| 0.001 | 48.53 |
| 0.01 | 87.67 |
| 0.1 | 70.72 |
| 1 | 24.18 |

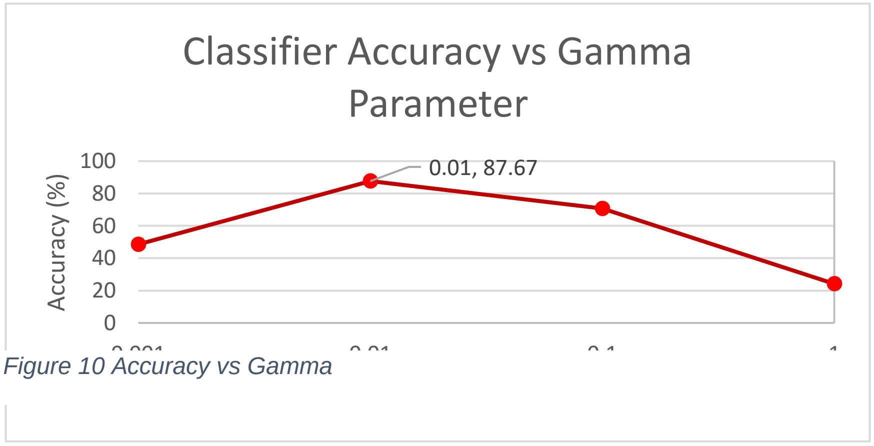

*Figure 10 Accuracy vs Gamma*

From the graph, the optimal gamma value is 0.01. At this value the classifier achieves 87.67% accuracy.

### Degree

The degree is a parameter relevant only to the polynomial kernel. A degree 1 polynomial kernel is equivalent to a linear kernel. The effect of varying the degree of the kernel is shown in the following Figure.

*Table 4 Degree Values*

| Degree | Accuracy(%) |
|---:|---:|
| 0 | 24.18 |
| 1 | 33.01 |
| 2 | 89.52 |
| 3 | 43.84 |
| 4 | 61.34 |
| 5 | 31.27 |
| 6 | 38.01 |

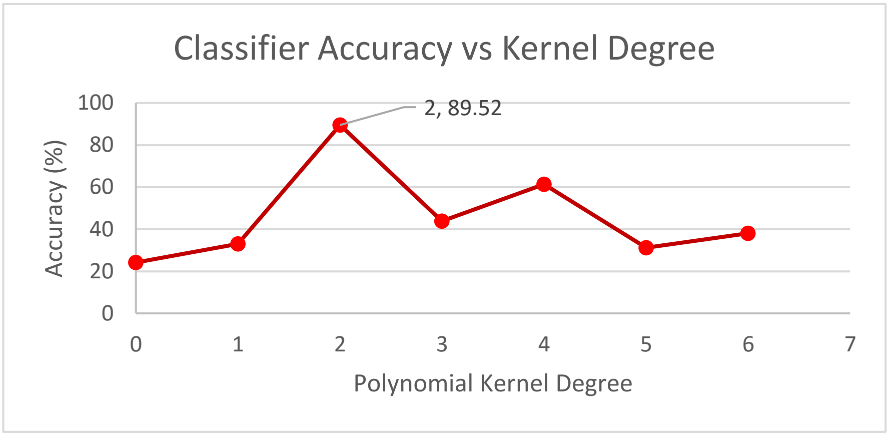

*Figure 11 Accuracy vs Kernel Degree*

The optimal degree is 2 when using a polynomial kernel. With this kernel the classifier can achieve an accuracy of 89.52%.

### Timing

Testing the execution time of the classifier was a challenging task. The most accurate way to test the timing would be to run it on an actual embedded processor. Timing is of great interest only during the classification of new samples. The timing to train the model is not important. It is only the time for the system to classify a new gesture that would impact the usability of the device by the patient. The goal is to classify gestures in under 75ms. The data utilized for this classifier is recorded over 40ms. This leaves just 35ms for the data to be processed assuming ideal conditions.

The classifier was able to classify 2920 samples in 1.998 seconds. This is an average of 0.0006848824350801232 seconds per sample. This is running on a four-core intel i5-5200 CPU @ 2.2GHz with 12GB ram. In comparison, STM32F4 microcontroller utilizes a single ARM cortex @ 240MHz with 64KB of RAM [18]. The processor speed of the PC is approximately 10x faster. It is unclear if the program is utilizing multiple cores and how much memory is used. Further investigation and testing is required to determine if a microcontroller like this would be able to classify gestures within the 35ms time requirement.

### Results

The initial classifier used the SVC class auto parameters (RBF, gamma=auto) and did not perform a wavelet transform on the data as a preprocessing step. The data points were simply standardized and used as the feature vectors. The classifier had a classification rate of 89.59%. Utilizing the DB1 coefficients as the feature vectors with the same classifier improved the classification rate to 89.79%. Performing the wavelet transform did very little to improve classification accuracy because the input wavelet was just eight samples long and the output coefficient vector was also eight samples long. The wavelet transform is much more effective with longer timeseries data because it can significantly reduce the dimensionality.

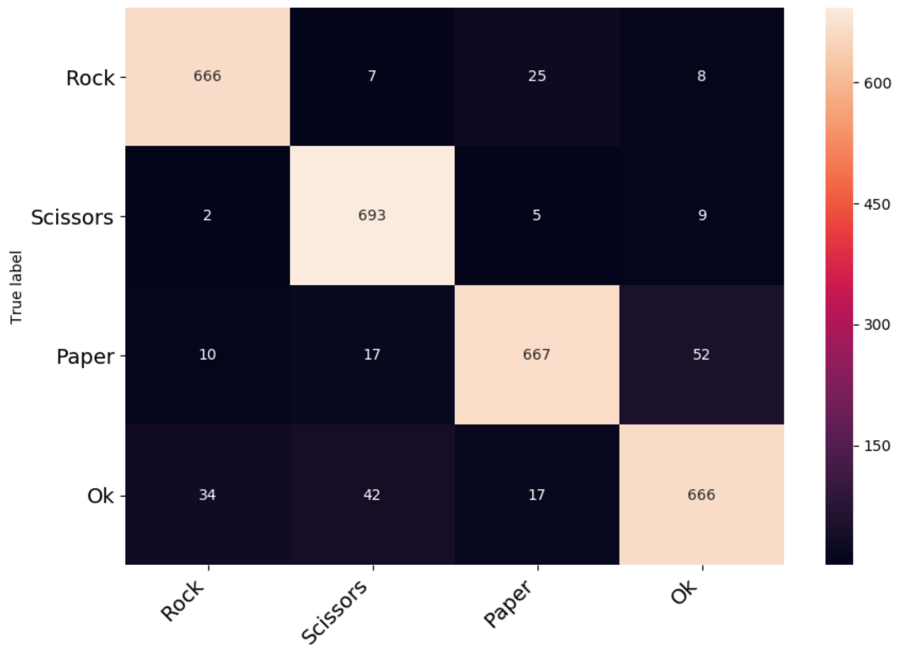

*Figure 12 Optimized Model Confusion Matrix*

A grid search was performed to find the optimal C and gamma parameters for the model. The optimal parameters were found to be C=10 and gamma=0.01. The classification rate achieved was 92.36% using the RBF kernel. This satisfies the accuracy goal of classification accuracy above 91%. The confusion matrix for the model with the optimized parameters can be seen in the Figure below. The most common mistake the classifier makes is between 'paper' and 'ok'. The classifier labelled samples as 'ok' 52 times when the correct label was 'paper'. The classifier also labelled 42 samples 'scissors' when the correct label was 'ok'

## Conclusions

The Support Vector Classifier described in this document was able to classify gestures based on forearm sEMG signals with a high degree of accuracy (92%). Optimizing the tuning parameters of the model were essential to achieving high accuracy classification.

Considerable further work is necessary to ensure this model can feasibly run on an embedded system. Timing requirements were met on a personal computer, but it is still unclear how these results will translate to a resource limited system.

## Recommendations

To verify the performance requirements of the classifier, the model must be implemented on an embedded system or run in a virtual environment which simulates a limited resource system. The preferred method of verification would be the former option, testing on a real system.

The second area to explore further would be the feature extraction step during preprocessing. It was shown that taking the first-order wavelet transform coefficients as the feature vectors did not provide significant improvements to classification accuracy. The coefficients resulted in the same length 8 vector as the time series data which could explain these results. Further work with using higher order coefficients or other feature extraction methods is a priority.

## References

[1]
“REDEFINING HUMAN.” [Online]. Available: http://www.psyonic.co/. [Accessed: 16-Oct-2019].

[2]
A. L. Ciancio *et al.*, “Control of Prosthetic Hands via the Peripheral Nervous System,” *Front Neurosci*, vol. 10, Apr. 2016.

[3]
U. Côté-Allard *et al.*, “Deep Learning for Electromyographic Hand Gesture Signal Classification Using Transfer Learning,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 27, no. 4, pp. 760–771, Apr. 2019.

[4]
S. Dick F., K. Bert U., L. Bernd G., and V. D. Johannes P., “High-density Surface EMG: Techniques and Applications at a Motor Unit Level,” *Biocybernetics and Biomedical Engineering*, vol. 32, no. 3, pp. 3–27, Jan. 2012.

[5]
“This Futuristic Armband Lets You Control Computers Like Magic,” *Time*. [Online]. Available: https://time.com/4173507/myo-armband-review/. [Accessed: 16-Oct-2019]. https://www.orfit.com/blog/the-peripheral-nerves-a-quick-assessment-of-nerve-functioning-or-rock-paper-scissors/

[7]
M. Atzori *et al.*, “Electromyography data for non-invasive naturally-controlled robotic hand prostheses,” *Sci Data*, vol. 1, no. 1, pp. 1–13, Dec. 2014.

[8]
M. B. Fraj, “In Depth: Parameter tuning for SVC,” Medium, 05-Jan-2018. [Online]. Available: https://medium.com/all-things-ai/in-depth-parameter-tuning-for-svc-758215394769. [Accessed: 07-Nov-2019].

[9]
S. Patel, “Chapter 2 : SVM (Support Vector Machine) — Theory,” Medium, 04-May-2017. [Online]. Available: https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72. [Accessed: 27-Oct-2019].

[10]
“Classify gestures by reading muscle activity.” [Online]. Available: https://kaggle.com/kyr7plus/emg-4. [Accessed: 07-Nov-2019].

[11]
S. Pancholi and A. M. Joshi, “Electromyography-Based Hand Gesture Recognition System for Upper Limb Amputees,” IEEE Sensors Letters, vol. 3, no. 3, pp. 1–4, Mar. 2019.

[12]

A. Lolure and V. R. Thool, “Wavelet transform based EMG feature extraction and evaluation using scatter graphs,” in 2015 International Conference on Industrial Instrumentation and Control (ICIC), 2015, pp. 1273–1277.

[13]

A. C. H. Rowe and P. C. Abbott, “Daubechies wavelets and Mathematica,” Comput. Phys., vol. 9, no. 6, p. 635, 1995.

[14]

“sklearn.preprocessing.StandardScaler — scikit-learn 0.21.3 documentation.” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html. [Accessed: 07-Nov-2019].

[15]

“sklearn.svm.SVC — scikit-learn 0.21.3 documentation.” [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html. [Accessed: 07-Nov-2019].

[16]

“Machine Learning.” [Online]. Available: http://openclassroom.stanford.edu/MainFolder/DocumentPage.php?course=MachineLearning&doc=exercises/ex8/ex8.html. [Accessed: 07-Nov-2019].

[17]

“RBF SVM parameters — scikit-learn 0.21.3 documentation.” [Online]. Available: https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html. [Accessed: 04-Nov-2019].

[18]

“STM32F4DISCOVERY - Discovery kit with STM32F407VG MCU * New order code STM32F407G-DISC1 (replaces STM32F4DISCOVERY) - STMicroelectronics.” [Online]. Available: https://www.st.com/en/evaluation-tools/stm32f4discovery.html. [Accessed: 25-Nov-2019].

## Appendix 1 – Daubechies Wavelet

The Daubechies Wavelet transform defines a family of orthogonal wavelets. Wavelets *Ψ(x)* are defined in terms of the scaling function [6]. The following equation relates the scaling function Φ(x) and the mother wavelet.

$$\psi(x) = \sqrt{2} \sum_{k=0}^{N-1} (-1)^k c_{N-1-k} \phi(2x-k). \qquad \textbf{\textit{Eq. 1}}$$

Where the scaling function is normalized by

$$\int \phi(x)dx = 1. \qquad \textbf{\textit{Eq. 2}}$$

Next, the filter coefficients are found based on the order of the transform and the following equations.

$$\begin{aligned} \int \phi(x)dx &= \sqrt{2} \sum_k c_k \int \phi(2x-k)dx \qquad \textbf{\textit{Eq. 3}} \\ &= \frac{1}{\sqrt{2}} \sum_k c_k \int \phi(2x-k)d(2x-k) \\ &= \frac{1}{\sqrt{2}} \sum_k c_k \int \phi(x)dx. \end{aligned}$$

$$\sum_{k=0}^{N-1} c_k = \sqrt{2}. \qquad \textbf{\textit{Eq. 4}}$$

The first order Daubechies Wavelet (db1) coefficients were used as feature vectors.