8-19-2019 10:30 AM

# Implementation of User-Independent Hand Gesture Recognition Classification Models Using IMU and EMG-based Sensor Fusion Techniques

José Guillermo Collí Alfaro
*The University of Western Ontario*

Supervisor
Trejos, Ana Luisa
*The University of Western Ontario*

Graduate Program in Biomedical Engineering
A thesis submitted in partial fulfillment of the requirements for the degree in Master of Engineering Science

## Recommended Citation

Collí Alfaro, José Guillermo, "Implementation of User-Independent Hand Gesture Recognition Classification Models Using IMU and EMG-based Sensor Fusion Techniques" (2019). *Electronic Thesis and Dissertation Repository*. 6347.
https://ir.lib.uwo.ca/etd/6347

# Implementation of User-Independent Hand Gesture Recognition Classification Models Using IMU and EMG-based Sensor Fusion Techniques

José Guillermo Collí Alfaro

M.E.Sc. Thesis, 2019

School of Biomedical Engineering
The University of Western Ontario

## Abstract

According to the World Health Organization, stroke is the third leading cause of disability. A common consequence of stroke is hemiparesis, which leads to the impairment of one side of the body and affects the performance of activities of daily living. It has been proven that targeting the motor impairments as early as possible while using wearable mechatronic devices as a robot assisted therapy, and letting the patient be in control of the robotic system can improve the rehabilitation outcomes. However, despite the increased progress on control methods for wearable mechatronic devices, the need for a more natural interface that allows for better control remains.

This work presents, a user-independent gesture classification method based on a sensor fusion technique that combines surface electromyography (EMG) and an inertial measurement unit (IMU). The Myo Armband was used to measure muscle activity and motion data from healthy subjects. Participants were asked to perform 10 types of gestures in 4 different arm positions while using the Myo on their dominant limb. Data obtained from 22 participants were used to classify the gestures using 4 different classification methods. Finally, for each classification method, a 5-fold cross-validation method was used to test the efficacy of the classification algorithms. Overall classification accuracies in the range of 33.11%–72.1% were obtained. However, following the optimization of the gesture datasets, the overall classification accuracies increased to the range of 45.5%–84.5%. These results suggest that by using the proposed sensor fusion approach, it is possible to achieve a more natural human machine interface that allows better control of wearable mechatronic devices during robot assisted therapies.

***Index terms***— Body-machine interfaces, wearable robotic systems.

# Lay Summary

According to the World Health Organization, stroke is the third leading cause of disability. A common consequence of stroke is the paralysis on one side of the body, which affects the performance of activities of daily living. It has been proven that treating this paralysis as early as possible using devices that combines both electrical and mechanical components, can improve the rehabilitation outcomes. However, despite the increase progress on control methods for these devices, a need for a more natural interface that allows for an intuitive interaction remains.

This work presents, a comparison of multiple interfaces based on gesture recognition that allow a natural interaction with a wearable robotic device. Muscle electrical activity of the forearm, and motion data were collected from 22 healthy participants while they performed 10 types of gestures in 4 different arm positions. These data were used to train four interfaces to recognize these 10 gestures.

Each interface was evaluated on its ability to differentiate between gestures after being trained using only the data obtained from the muscles' electrical activity, and after being trained using both, the muscle electrical activity and motion data. The results obtained suggest that it is possible to achieve a more natural interaction with wearable devices during robot assisted therapies.

# Acknowledgements

First and foremost, I would like to express my appreciation for the guidance provided by my supervisor, Dr. Ana Luisa Trejos. It is because of her that I was able to do a Masters degree. I feel extremely lucky to have had such a wonderful person providing guidance throughout this journey. I am extremely thankful to have been a part of the Wearable Biomechatronics Laboratory. I would like to express my gratitude to each member that is, and has been, part of this awesome group. Special thanks goes to Anas Ibrahim for his help on with the neural network models. A big thanks goes to Jacob Tryon for helping me with the statistical analyses. You are the best, around! I would also like to thank Yue Zhou, for his constant support and guidance during these two years (and also for introducing me to rock climbing).

A big thanks goes to my parents José Luis Collí Solis and Militza Alfaro Gamboa for always being there when I needed them the most, and for believing in me. I love you. A thanks goes to my brother and sister as well, it was weird not having you around. Thanks to the group #325:NameOfADayEvent, and the rest of my friends in México, guys you are breathtaking! Lastly, I want to express my most sincere gratitude towards my fiancée Maremy Castillo Ocampo, for all these years of constant love and support. Thanks for being by my side on every decision I take. I love you so much.

This work was supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada under grant RGPIN-2014-03815; by the Canadian Foundation for Innovation (CFI), by the Ontario Research Fund (ORF); and by the Ontario Ministry of Economic Development, Trade and Employment and the Ontario Ministry of Research and Innovation through the Early Researcher Award. Thanks to the Mexican National Council of Science and Technology (CONACYT) for funding my studies.

# Contents

# List of Figures

# List of Tables

# Nomenclature and Acronyms

## Latin Letters

| | |
|---|---|
| $b$ | Machine learning models bias term |
| $d(\cdot)$ | Kernel distance function |
| $\mathcal{C}$ | SVM regularization term |
| $\mathcal{I}$ | Identity matrix |
| $\ell(\cdot)$ | Convex multiclass loss function |
| $\mathcal{K}(\cdot)$ | Kernel function |
| $\mathcal{L}(\cdot)$ | Lagrangian |
| $\mathcal{S}_{M_A}$ | Schur complement of matrix $M_A$ |
| $w$ | Weights vector |
| $\hat{w}$ | Pretrained LS-SVM model |
| $X_{cal}$ | Calibration dataset obtained during the PAC and Adaptive LS-SVM classification methods |
| $X_{test}$ | Test dataset obtained during the PAC and Adaptive LS-SVM classification methods |
| $X_{train}$ | Dataset used for training during the PAC and Adaptive LS-SVM classification methods |
| $Y_i$ | Predicted label of the $i^{th}$ sample |
| $\hat{Y}_i$ | Prediction of the $i^{th}$ sample obtained by using a pretrained LS-SVM model |

| | |
|---|---|
| $\tilde{Y}_i$ | Closed form solution of the leave-one-out prediction of the $i^{th}$ sample |

## Greek Letters

| | |
|---|---|
| $\alpha_i$ | Vector of support vectors |
| $\beta$ | Scaling factor that weighs a pretrained LS-SVM model |
| $\gamma$ | RBF kernel parameter |
| $\lambda$ | Factor that controls the age of the representative particles in the PAC algorithm |
| $\xi_i$ | SVM slack variables |
| $\phi(\cdot)$ | Non-linear function that maps samples to a high dimensional space |
| $\Psi(\cdot)$ | Teager-Kaiser Energy Operator function |
| $\Psi^{SC}$ | Bilinear Model of the EMG signal |
| $\Psi_{avg}$ | Mean of the conditioned EMG signal |
| $\Psi_{rms}$ | RMS of conditioned EMG signal |

## Acronyms

| | |
|---|---|
| ADC | Analog to Digital Converter |
| ANN | Artificial Neural Networks |
| ANOVA | Analysis of Variance |
| AR | Auto-Regressive |
| BM | Bilinear Models |
| Cap | Capitate bone |
| CIMT | Constraint-Induced Movement Therapy |
| CVF1 | Cross-Validation Fold 1 |
| CVF2 | Cross-Validation Fold 2 |
| CVF3 | Cross-Validation Fold 3 |

| | |
|---|---|
| CVF4 | Cross-Validation Fold 4 |
| CVF5 | Cross-Validation Fold 5 |
| DOF | Degrees of Freedom |
| EEG | Electroencephalography |
| EIT | Electrical Impedance Tomography |
| EMG | Electromyography |
| FMG | Force Myography |
| FPGA | Field Programmable Gate Arrays |
| GUI | Graphical User Interface |
| Ham | Hamate bone |
| HC | Hand Closed |
| HMM | Hidden Markov Models |
| HO | Hand Open |
| IMU | Inertial Measurement Unit |
| Itr1 | Iteration 1 |
| Itr2 | Iteration 2 |
| Itr3 | Iteration 3 |
| Itr4 | Iteration 4 |
| Itr5 | Iteration 5 |
| kNN | *k* Nearest Neighbours |
| KP | Key Pinch |
| LDA | Linear Discriminant Analysis |
| LS-SVM | Least Squares Support Vector Machines |
| Lun | Lunate bone |
| MAV | Mean Absolute Value |
| MAVS | Mean Absolute Value Slope |

| | |
|---|---|
| MC1 | Metacarpal bone 1 |
| MC2 | Metacarpal bone 2 |
| MC3 | Metacarpal bone 3 |
| MC4 | Metacarpal bone 4 |
| MC5 | Metacarpal bone 5 |
| MLP | Multilayer Perceptron |
| MMG | Mechanomyography |
| MNF | Mean Frequency |
| MUAP | Motor Unit Action Potential |
| NN | Neural Network |
| PAC | Particle Adaptive Classifier |
| PCA | Principal Component Analysis |
| Pis | Pisiform bone |
| PKF | Peak Frequency |
| PM | Predictive Model |
| PP | Precision Pinch |
| PSD | Power Spectrum Density |
| PSR | Power Spectrum Ratio |
| Rad | Radius bone |
| ReLU | Rectified Linear Unit |
| ROM | Range of Motion |
| RBF | Radial-Basis Function |
| RP | Representative Particles |
| RSNNS | R Stuttgart Neural Network Simulator |
| RSS | Representative Sample Set |
| Sca | Scaphoid bone |

| | |
|---|---|
| sEMG | surface Electromyography |
| SGD | Stochastic Gradient Descent |
| SNR | Signal to Noise Ratio |
| SPSS | Statistical Package for Social Sciences |
| SVD | Singular Value Decomposition |
| SVM | Support Vector Machines |
| TKEO | Teager-Kaiser Energy Operator |
| Tpd | Trapezoid bone |
| Tpm | Trapezium bone |
| Trq | Triquetrum bone |
| UE | Upper Extremity |
| Uln | Ulna bone |
| WAb | Wrist Abduction |
| WAd | Wrist Adduction |
| WE | Wrist Extension |
| WF | Wrist Flexion |
| WL | Waveform Length |
| WP | Wrist Pronation |
| WS | Wrist Supination |
| ZC | Zero Crossing |

## Units

| | |
|---|---|
| cm | Centimetres |
| Hz | Hertz |
| kg | Kilograms |

| | |
|---|---|
| mm | Millimetres |
| ms | Milliseconds |
| s | Seconds |
| yrs | Years |
| ° | Degrees |

# Chapter 1

# Introduction

Cerebrovascular accidents, commonly known as stroke, are the third leading cause of disability and the second leading cause of death in the world [1]. Up to 80% of stroke survivors often present upper extremity (UE) hemiparesis [2] thus, requiring extensive rehabilitation sessions to regain some UE functions. The main inconvenience for hemiparetic stroke patients is that they may not be eligible for common rehabilitation techniques, which are usually aimed towards a population with mild impairments [3]. Therefore, alternative therapies are necessary for hemiparetic stroke patients to improve neuroplasticity, which is the ability of the brain to "rewire" functions associated with damaged tissue to healthy parts of the brain.

One alternative therapy is bilateral training, which consists of the activation of motor synergies between limbs, and as explained by Stewart *et al.*[4]: "voluntary movements of the intact limb may facilitate voluntary movements in the paretic limb." Rehabilitation is enhanced because when symmetrical movements are executed, the motor cortex governing the actions of the healthy limb is activated, thus increasing the voluntary muscle contractions in the impaired limb. Hence, by performing bilateral movements, it is possible to promote neuroplasticity [4]. Moreover, bilateral training has been tackled from different approaches, with robot-assisted therapy being the one with greatest potential. Even though this innovative field has proven to be effective, much work needs to be done regarding the interaction with robotic devices during robot-assisted therapies. In this sense, by improving the patient's level of engagement during robot-mediated rehabilitation, such rehabilitation can provide an advantage over traditional rehabilitation therapies [5].

## 1.1 Motivation

Recently, it was proven that by using robot-assisted therapy as a complementary method to traditional rehabilitation techniques, it is possible to achieve significant improvements in the rehabilitation outcomes [6]. In the context of upper-limb robot-assisted therapy, wearable mechatronic devices allow for the rehabilitation of specific groups of muscles by applying different torques at certain joints of the upper limb [7]. An important aspect of robot-assisted therapy is the need to promote patient mobility instead of just letting the robot perform the complete rehabilitation task. Furthermore, it has been found that when the user's movements are governed by those of the robot, the amount of effort that the patient exerts while performing voluntary actions is reduced, leading to negative effects during the recovery process [8]. Consequently, the ideal therapy is one in which the patient is part of the control loop, which can be achieved by enabling different ways of communication and interaction between the robotic device and the user [9]. However, even when research in the field of robot-assisted therapy has increased [6, 7, 10–12], the need for a natural human machine interface that allows an intuitive control and long-term adoption of this technology remains [13].

To address this issue, gesture recognition has been studied as a possible solution for human machine interface applications [14, 15], with gesture recognition based on electromyography (EMG) being the most commonly used for interactions with robotic devices [16]. In previous years, the Myo Armband [17], was introduced as a commercially available EMG based gesture recognition device, opening a lot of possibilities for gesture recognition applications. However, the built-in proprietary system of the Myo Armband is limited to the recognition of 5 gestures. This limitation may be because the accuracy of the device is inversely proportional to the amount of gestures it can detect [18, 19]. Moreover, when the Myo Armband is used in a user-independent scenario, which means that it can be used by new users without prior training, its recognition accuracy drops from 83.1% (user-dependent) to 53.7% (user-independent) [20]. This is important because a user-independent scenario allows for a practical gesture recognition system. Such system can help new users become proficient in using wearable mechatronic devices during robot-mediated therapies after a short period of time. By using the system's previous training data, long training sessions that would otherwise be required to adjust the interface to a specific subject, would not be necessary anymore [21]. Hence, the time to complete a rehabilitation session for hemiparetic stroke patients would be reduced. By overcoming these limitations, the Myo Armband, or a similar EMG-based device, can be used to build a low-cost, reliable interface that can interact with wearable robotic devices.

## 1.2 General Problem Statement

Commonly, gesture recognition can be achieved by using different pattern recognition algorithms such as support vector machines (SVM) and linear discriminant analysis (LDA). Furthermore, the ideal gesture recognition interface should be one that requires training a classifier only once without the need of retraining every time a new subject wears the device. This is known as user-independent classification. However, because EMG signals are affected by different factors such as muscle fatigue and level of health, among others, a user-independent classification is not always possible without sacrificing classifier accuracy. To overcome this issue, one proposed solution consists of using sensor fusion techniques, for example combining EMG data with kinematic data coming from an inertial measurement unit (IMU), to improve the classification accuracy [21, 22]. Another solution is the use of incremental learning, which is a strategy to update the classifiers by retraining them using new data samples [23]. However, in the specific case of SVM, using incremental learning may lead to a concept drift, which is the change in the data distribution over time [24]. Nonetheless, recent studies [23, 25] have proposed solutions for concept drift during incremental learning by using SVM with particle adaptive classifier (PAC) [23] or SVM in combination with another classification algorithm, such as $k$ nearest neighbours (kNN) [25].

The purpose of this work is to develop a user-independent hand gesture recognition interface using sensor fusion techniques and an adaptive incremental learning classifier. This work proposes that by combining EMG and IMU data from the commercially available Myo Armband, and by using incremental learning using different classification methods, it is possible to increase the number of gestures that the Myo Armband can recognize while also improving its detection accuracy.

## 1.3 Research Objectives and Scope

This thesis specifically focuses on identifying and classifying wrist and finger gestures based on EMG and IMU data collected from the forearm muscles. A database of EMG and kinematic data was collected from healthy subjects while they performed 10 different wrist and finger gestures.

The primary objectives of this thesis are as follows:

1. To acquire and analyze EMG and IMU data from healthy subjects while they perform different hand gestures.

2. To train different classifiers using EMG data, and EMG and IMU data collected from the Myo Armband, and then evaluate their classification performance.

## 1.4 Overview of the Thesis

The structure of this thesis is summarized in the outline below:

**Chapter 1** Introduction: This introductory chapter.

**Chapter 2** Literature Review: Presents a review of wrist anatomy, wrist rehabilitation, robot-assisted therapy, human machine interfaces used in robot-assisted therapies, gesture recognition and motion intention detection, EMG and IMU signal acquisition, processing and analysis, and user-independent classification methods.

**Chapter 3** Data Collection and Processing: Presents the methods used for collecting EMG and IMU data using the Myo Armband including the data collection protocol and methods of data processing and analysis.

**Chapter 4** Classification Methods: Presents the implementation of four classification methods aimed towards an user-independent gesture recognition using EMG, and EMG and IMU data.

**Chapter 5** Results and Discussion: Presents the results of the data analysis and explains their significance.

**Chapter 6** Conclusion and Future Work: Emphasizes the contributions of this work and provides recommendations for future work.

**Appendix A** Permissions and Approvals: Includes ethics permission and approval, consent form and trial form.

**Appendix B** MATLAB Code: Describes the MATLAB code used for EMG and IMU analysis.

**Appendix C** Mathematical Formulations: Describes the mathematical equations used to solve some of the classification algorithms presented in this work.

**Appendix D** Python Code: Describes the Python code used for real-time EMG data streaming, and for classification using the bilinear EMG models.

**Appendix E** R Code: Describes the R code used for training and testing the MLP networks.

## Chapter 2

# Literature Review

## 2.1 Introduction

To provide a knowledge base for the remainder of this thesis, this chapter presents a review of the literature in the areas of wrist anatomy (Section 2.2), wrist rehabilitation (Section 2.3), robot-assisted therapy (Section 2.4), motion intention and gesture recognition interfaces (Section 2.5), user-independent gesture recognition (Section 2.6), EMG pattern recognition (Section 2.7), and classification methods aimed towards a user-independent classification (Section 2.8). A literature search was conducted using Google Scholar from September 2017 to July 2019. The keywords used in the search included combinations of the following: upper limb rehabilitation, EMG features, EMG+IMU sensor fusion, user-independent classification, and EMG gesture recognition. A total of 119 references, which include papers and books, were incorporated into the literature review.

## 2.2 Wrist Anatomy

The wrist is a complex human joint localized between the hand and the forearm. It is comprised of a collection of bones, which consist of the distal ends of the radius (Rad) and ulna (Uln) bones, 8 carpal bones, which include the scaphoid (Sca), the lunate (Lun), the triquetrum (Trq), the pisiform (Pis), the trapezoid (Tpd), the trapezium (Tpm), the capitate (Cap), and the hamate (Ham); and the proximal segments of the 5 metacarpal bones (MC1 to MC5) of the hand (Figure 2.1) [26, 27].

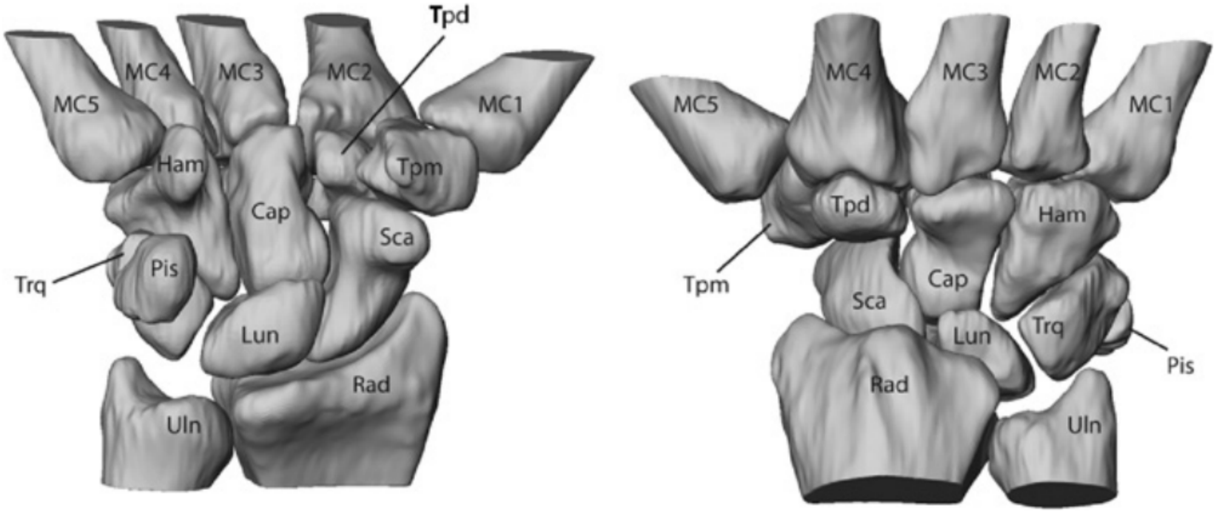


Figure 2.1: Anterior (left) and posterior (right) views of the bones in the wrist. Reprinted, with permission [27].

This complex structure allows the hand to interact with the external environment by adopting different poses depending on the situation. Furthermore, by involving the radio–ulnar complex of the forearm, complex movements, such as the rotation of the hand, can be performed [28]. These sets of motions allow the wrist joint to be represented, in mechanical terms, as a 3 degree-of-freedom (DOF) system [29].

## 2.3 Wrist Rehabilitation

Although it is possible for the the hand to achieve different poses thanks to the wrist joint, these poses are limited to a certain range of motion (ROM) (Table 2.1). However, when people suffer from a neurological injury such as a stroke, they are prone to experiencing some sort of impairment that hinders the motor abilities of the wrist joint. This impairment can come in the form of hemiparesis, which is the partial paralysis of the limbs due to muscle weakness, and can be a limiting factor during activities of daily living [30]. Therefore, to regain lost motor functions, hemiparetic stroke patients must undergo a series of rehabilitation treatments in order to promote neuroplasticity, *i.e.*, the ability of the brain to form new neural connections associated with damaged brain tissue in healthy parts of the brain [4, 31]. In the following sections, some of the popular therapeutic techniques used to promote neuroplasticity of the brain are described.

Table 2.1: Average range of motion of the wrist joint for each type of movement [32].

| **Motion** | **Average ROM** |
|---|---|
| Wrist Flexion | 73° |
| Wrist Extension | 71° |
| Wrist Radial Deviation | 19° |
| Wrist Ulnar Deviation | 33° |
| Wrist Pronation | 71° |
| Wrist Suppination | 84° |

### 2.3.1 Constraint-Induced Movement Therapy

Constraint-induced movement therapy (CIMT), also known as unilateral training therapy, is a common rehabilitation technique used for hemiparetic stroke patients. This therapeutic approach consists of improving the involvement of the affected limb during activities of daily living by forcing the movement of the paretic limb while the motions coming from the healthy limb are reduced or constrained [33, 34]. Although promising, CIMT has the drawback of being aimed only at a population whose hemiparetic symptoms are not that severe [3]. Also, the constant practice of CIMT may result in a reduced need for the brain to retain some information related to the motions being performed, which results in less improvement of neuroplasticity [35].

### 2.3.2 Bilateral Movement Training

An alternative to CIMT is known as bilateral movement training. This technique consists of activating motor synergies of the limbs by promoting the coordination of movement between the paretic and non-paretic limb, thus facilitating voluntary motions on the affected limb [4, 36]. Three main categories of bilateral training exist, divided according to the type of rehabilitation task they perform. These categories are the following:

**Repetitive reaching practice with the hand fixed:** This type of training consists of training a reaching motion by attaching the distal ends of both hands to a mechanical device. Then, a reaching movement is trained by performing symmetrical motions (both hands push the device in the same direction) or assymetrical motions (one hand pushes while the other pulls the device) [37].

**Isolated muscle repetitive task practice:** Isolated repetitive muscle training consists of isolating a group of muscles on both arms by restraining all types of motions but one. For example, patients may be asked to perform repetitive movements of wrist flexion, wrist extension, among others [37].

**Whole arm functional task training:** While the previous two tasks required the performing of a single motion or activity, this training involves a set of motions that include the grasp, reach, and release of an object. This training can be done by simultaneously using both arms to perform these three actions or by using the non-paretic limb to guide the impaired limb [37].

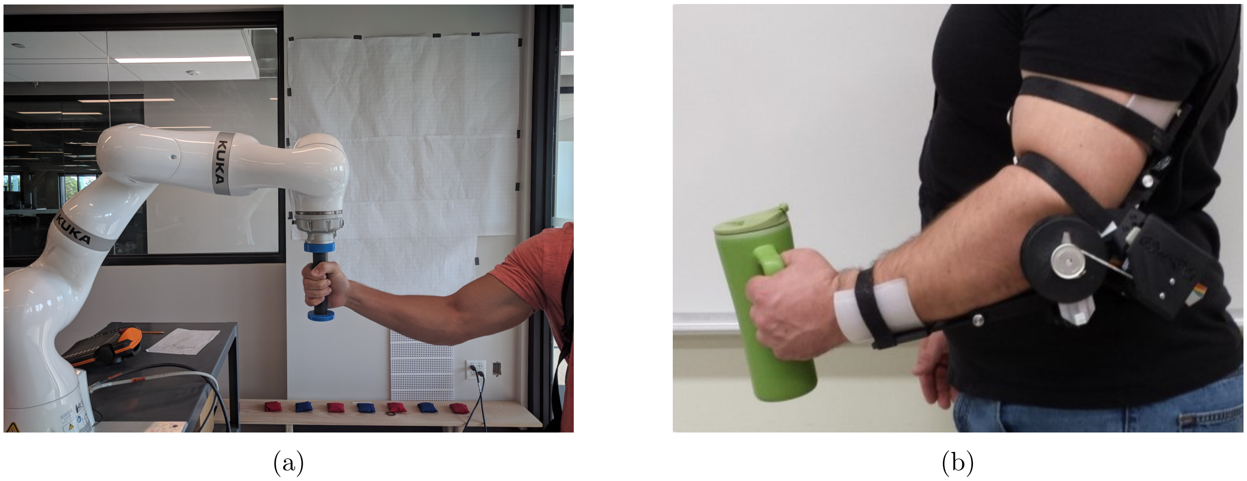
## 2.4 Robot-Assisted Therapy

The previous section described traditional upper limb rehabilitation methods aimed towards improving neuroplasticity. Even though the described therapeutic methods are designed to provide high intensity training by being repetitive, task-oriented and challenging to the patient, a need to further enhance the effects of the rehabilitation treatments exists [38]. Therefore, research groups have come with the solution of using smart robotic devices to address the need of improving the rehabilitation effects on patients with impaired arm functions after stroke. Robot-assisted therapy can provide the tools to asses the improvement of motor control of the affected limbs in a much faster, efficient, and objective way [39].

Usually, upper limb robot-assisted therapy is based on two types of robotic devices: serial robotic manipulators and wearable mechatronic devices (Figure 2.2). The former is based on the use of a robotic manipulator, which its end effector is attached to the hand of the patient, and then it assists the patient by generating forces in order to complete the rehabilitation task.

The second type of therapy robots are known as wearable mechatronic devices. Contrary to end effector robots, these types of mechatronic devices are designed so that their joints match those of the user. Moreover, the main advantage of this design is that wearable mechatronic devices allow for the rehabilitation of specific group of muscles by applying different torques at certain joints of the upper limb [7].

In order for the wearable mechatronic devices to work properly with the upper limb, it is



(a) (b)

Figure 2.2: Example of robotic devices that can be used during upper limb rehabilitation. The KUKA robot, a serial robotic manipulator (a), and the WearME Brace [40], a wearable mechatronic device (b).

necessary to apply effective control strategies, which will dictate not only the mechanical behavior of the system, but also the human-robot interactions [10]. Table 2.2 shows a summary of the different control strategies used during robot-assisted therapies.

Among all of the control strategies described in Table 2.2, partially assistive control stands as the most important because it promotes patient mobility instead of just letting the robot perform the complete rehabilitation task. Furthermore, it has been discussed that when the user's movements are led by those of the robot, the amount of effort the patient puts while performing voluntary actions is reduced, leading to negative effects during the recovery process [41]. Therefore, it is important to adopt the concept of user in the loop, in which the patient can interact with the robotic device in a more natural and active way. Consequently, by adopting a more active role that promotes self-improvement, hemiparetic stroke patients will feel more comfortable during robot-assisted therapy sessions [42].

## 2.5 Motion Intention Detection vs. Gesture Recognition

Different approaches have targeted the "user in the loop" paradigm. Of these, motion intention detection is the approach that has the potential to give almost full control to the wearer of the mechatronic device. Motion intention detection works by identifying motion patterns and classi-

Table 2.2: Control modalities used during robot-assisted therapies [10, 41].

| Modality | Sub-Modality | Further Sub-Modality |
|---|---|---|
| **Assistive Mode:** The robot produces forces that are applied to the impaired limb to complete the task | **Passive Control:** The robot moves the impaired limb without the need for the patient to start the action | **Passive Trajectory Tracking:** The robot follows a predefined path |
| | | **Passive Mirroring:** The robot mimics the behavior of the healthy limb to synchronously drag the impaired limb in a master–slave configuration |
| | | **Passive Stretching:** The robot passively stretches the joints to identify the angle-resistance relationships |
| | **Triggered Passive Control:** After the patients triggers the action, the robot passively follows a predefined trajectory | |
| | **Partially Assistive Control:** The patient controls the motion and the robot supports the patient in order to complete the task only when it senses assistance is needed | **Impedance/Admittance Control:** The robot acts with two model-based approaches: a force controller with position feedback (impedance) and a position controller with force feedback (admittance). Depending on the design, the robot may use one or the other, or a combination of both |
| | | **Attractive Force Field:** An attractive force field around the destination target pulls the robot's end effector based on its position. Assistive forces act on the other joints of the robot based on reference trajectories |
| | | **Model-Based Assistance:** Forces required to maintain a specific pose are estimated. Then a proportional control system drives the actuators of the robot with only the necessary force to maintain the pose |
| | | **Offline Adaptive Control:** This type of control strategy is an adaptation of a trial-by-trial solution. The idea consists on training a feedforward term using this approach and then, adding it to a PID or PD controller at the joint level |
| **Corrective Mode:** The robot is only active when the patient is not performing the intended motion in a correct manner | **Tunneling:** This corrective mode consists on creating virtual channels in which the patients move. Whenever the patient moves away from these channels, the robot drives him back into them | |
| | **Coordination Control:** This approach tries to solve the reference time-dependence problem by implementing a control law to regulate the position and velocities of the joints relative to the others (coordination of the joints) | |
| **Resistive Mode:** The robot applies forces in the opposite direction of the limb movement | | |

fying them into different categories. Whenever the user intends to move a joint, different muscle fibers corresponding to the muscles of that joint, produce different patterns of contraction and relaxation. These patterns generate different biological signals that can be detected using different techniques, such as electroencephalography (EEG), electromyography (EMG), mechanomyography (MMG) and force myography (FMG). These signals can be used then to predict the user movement intention and then generate control commands for the wearable mechatronic devices.

Several studies have managed to utilize motion intention detection as a sophisticated control method. Ryser *et al.* [43] developed a wearable robotic hand orthosis controlled using motion intention detection based on EMG signals. The device detected patterns produced by the activation of different muscles while performing specific hand gestures and then, these patterns were utilized to control a wrist wearable mechatronic device. Zhang and Harrison [44] created a device that was able to measure the cross-sectional impedance distribution among the muscles of the wrist of the subject using the principle of the electrical impedance tomography (EIT). Because of this, they were able to detect hand gestures with high accuracies. In studies like [45], the detection of vibrations produced by the muscles also known as mechanomyography, was used to classify the patterns; whereas in [46] force myography was utilized, which is the detection of the force produced by the inner exerted pressure of the limb.

However, adopting motion intention detection as a control method narrows the population of stroke survivors that can benefit from robot-assisted therapy to just a few. This is due to the presence of involuntary muscular activation during voluntary movements that compromise the signal classification in the training process of stroke patients [47]. Therefore, it is necessary to shift to a more reliable approach that can provide an intuitive and straightforward control method. Gesture recognition has been used as a reliable human machine interface in mobile devices, as well as with robotic manipulators [14, 15]. Yet, even when gesture recognition can be a valuable interface for the control of wearable mechatronic devices during robot-assisted therapies, its potential has not been fully explored.

Nonetheless, recent studies have started to explore the use of different technologies for the detection of upper limb gestures. For example, Jung *et al.* [48] designed a wearable device that was able to detect six gestures of the hand. The device consisted of a bracelet containing several air bladders and air pressure sensors that detected the small changes in the shape of the muscles due to swelling during voluntary actions. The problem with this device was that the presence of involuntary movements affected the sensor readings, thus decreasing its reliability for the control of wearable mechatronic devices. Noronha *et al.* [49] explored the use of eye tracking technology to control a soft robotic glove, however the problem with this approach was that they were only able to detect one single gesture, making the overall control system less versatile for activities of daily living. Another approach for detecting gestures was explored by Zheng *et al.* [50], who developed an armband with capacitive sensors. However, their study did not explore environmental disturbances that occur when the sensing band was used multiple times. Finally, Haroon and Malik [51] were able to detect gestures by obtaining EMG signals from the forearm of a subject. This information was then used for the control of a robotic gripper.

## 2.6 User-Indpendent Gesture Recognition

To effectively use gesture recognition-based interfaces during rehabilitation sessions of hemiparetic stroke patients, it is necessary to develop strategies aimed towards a user-independent scenario. In doing so, deployment of robot-assisted therapies would be facilitated by adopting a system that does not require any type of offline training for each new patient [21, 52].

Although EMG-based gesture recognition shows the potential to be the ideal interface for human machine interactions due to EMG signals being rich in information about muscle electrical activity, EMG has a low signal to noise ratio (SNR), and the fact that hemiparetic stroke patients have limited motor abilities, this type of signals are difficult to use during robot-assisted therapies. However, when used in combination with other types of sensors, its possible to compensate for this disadvantage, as shown in several studies [15, 53]. Despite the capabilities of simultaneously employing information coming from multiple sources, different studies have opted to employ more sophisticated pattern recognition techniques to enhance the capabilities of the EMG signals, in order to achieve a user-indepenendent gesture recognition interface. In the following sections, these EMG pattern recognition techniques will be presented.

## 2.7 EMG Pattern Recognition

Before explaining the user-independent pattern recognition methods, it is important to understand how pattern recognition of EMG signals works. In general, development of an EMG-based pattern recognition system follows the procedure summarized in Figure 2.3 [54, 55].

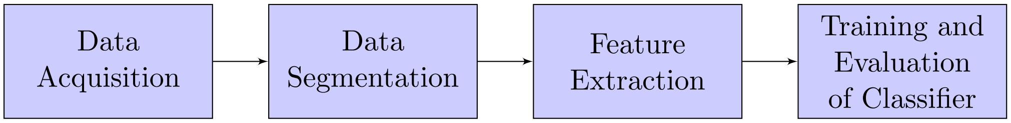


Figure 2.3: General procedure of an EMG pattern recognition system.

### 2.7.1 EMG Data Collection

Electromyography (EMG) is a way to measure the electrical activity produced by the muscles when they contract. In general, this electrical activity can be acquired using invasive and non-invasive methods [56]. The first method consists of inserting needle electrodes through the skin and directly into the muscle. On the other hand, the non-invasive technique consists of using electrodes made of conductive materials that range from stainless steel to gold or silver metals [57], placed on the surface of the skin over the muscle of interest. This process of acquiring the EMG signals using the non-invasive method is also known as surface electromyograpy (sEMG).

Whenever a muscle contracts, each of its muscle fibers produces an action potential, which when summed together, produce something known as motor unit action potential (MUAP). The MUAP is responsible for producing currents that flow from the muscle cells to the surface of the skin. The sum of all MUAPs produce an EMG signal that is read by electrodes placed on the skin [58]. Once obtained, this EMG signal needs to be preprocessed before proceeding towards the next steps of the pattern recognition. Given that the EMG signal is in the order of milivolts, the first preprocessing step consists of amplifying with a gain in the range of 1000 to 10000 [59]. After being amplified, the EMG signal is then filtered using a bandpass filter, which is usually composed by a high-pass filter and a low-pass filter. These two filters are designed with cut off frequencies around 10 to 20 Hz for the high-pass filter, and 500 Hz for the low pass filter [60]. Furthermore, a notch filter with a cutoff frequency of 60 Hz is also applied to remove the power line interference that can corrupt the EMG signal.

### 2.7.2 EMG Data Segmentation

Following the amplification and filtering of the raw EMG signal, the second step of the pattern recognition process consists of segmenting the preprocessed signal so that it can be analyzed for real-time applications. However, it is necessary to first detect the moment when the muscle goes from an idle or relaxed state to the contracted state. This process of detecting the change of state is known as EMG onset detection and is important because it can be used as the trigger to start motion analysis.

Typical EMG onset detection methods use threshold-based algorithms. These algorithms include single-threshold approaches [61], and double-threshold approaches [62–64]. While single-threshold based approaches rely on detecting the instant when the amplitude of the signal surpasses a predefined value, double-threshold approaches take this concept even further by ignoring false-alarm triggers. This is achieved by counting the number of consecutive samples in which the amplitude of the EMG signal is above a predefined threshold, after the first motion trigger event happens.

#### 2.7.2.1 Teager-Kaiser Energy Operator

Solnik *et al.* [65] showed that regardless of the motion onset detection method used, the detection accuracy could be improved by using the Teager-Kaiser energy operator (TKEO), which measures the instantaneous energy change of the signal [66], and is defined as follows:

$$\Psi\left(x_i\right) = x_i^2 - \left(x_{i+1} \times x_{i-1}\right), \tag{2.1}$$

where $x_i$ represents the $i^{th}$ EMG sample value.

#### 2.7.2.2 Data Windowing

After preprocessing the EMG signal with TKEO, it can be properly segmented. This is particularly useful because segmenting the EMG signal allows for the extraction of information from the active segments of the signal, *i.e.*, segments where the motion is being performed. However, for this information to be used in real-time applications, segments must be divided into windows, which may be either continuous or with overlaps. From these windows, features used on the latest stages of the EMG pattern recognition are extracted. If the system were to work in real time, the length of the windows should account for the maximum tolerated delay (300 ms) between processing the information and controlling a myoelectric device [67]. Furthermore, depending on the application, a trade-off between classification accuracy and delay exists, which can affect the choice of the window length. In this sense, continuous windows with lengths of 200 ms provide better classification accuracies, while overlapped windows with lengths above 200 ms, and 150 ms of overlap, provide a faster response with a noticeable increase in the classification error [68].

### 2.7.3 Feature Extraction

Following data segmentation, the next pattern recognition step consists of feature extraction. For the control of EMG-based wearable mechatronic devices, different studies have explored the use of time domain, frequency domain, and time-frequency domain features for motion classification [55, 69, 70]. Time domain features are the predominant features used in applications involving wearable mechatronic devices and most myoelectrical devices. Their popularity comes from their relatively fast computation due to not requiring any type of transformation [70]. On the other hand, frequency domain features are mostly used in applications that study muscle fatigue, and are based on the signal's estimated power spectrum density (PSD) [54, 70]. Finally, time-frequency domain features are used to extract the signal's energy information in time and frequency simultaneously. However, both frequency and time-frequency domain features require transformations that can be computationally expensive [54].

#### 2.7.3.1 Time Domain Features

Some of the most commonly used time domain features used in the literature are listed below [70, 71]:

**Mean Absolute Value (MAV)** The MAV feature represents the mean absolute value of the signal amplitude from a segment of window of size $m$, as follows:

$$MAV(n) = \frac{1}{m}\sum_{i=1}^{m} |x_n(i)|\,, \tag{2.2}$$

where $x_n(i)$ represents the measure of sample $i$ of channel $n$.

**Waveform Length (WL)** This feature represents the cumulative length of the signal over a time segment [70].

$$WL = \sum_{i=1}^{N-1} |x(i+1) - x(i)|\,, \tag{2.3}$$

where $N$ is the length of the signal, and $x(i)$ is the $i^{th}$ sample of the signal.

**Mean Absolute Value Slope (MAVS)** The MAVS feature of each segment $n$ is defined as the difference between the MAV of the current segment and the next segment for all $N$ segments [70], as follows:

$$MAVS(n) = MAV_{n+1} - MAV_n \qquad n = 1,\ \ldots,\ N. \tag{2.4}$$

**Auto-regressive Coefficients (AR)** An AR model represents each sample $x_i$ of the EMG signal as the linear combination of each previous $x_{i-p}$ samples and white noise $w_i$ [70]. The AR model is defined as follows:

$$x_i = \sum_{p=1}^{P} a_p x_{i-p} + w_i, \tag{2.5}$$

where $P$ is the AR order and the coefficients $a_p$ are used as the EMG features.

**Zero Crossing (ZC)** This feature represents the number of times the signal crosses the zero value. To avoid any background noise, a crossing is considered only when the signal exceeds certain threshold $th$ [70]. The ZC value is computed as follows:

$$ZC = \sum_{i=1}^{N-1} f\left[x\left(i\right), x\left(i+1\right)\right], \tag{2.6}$$

where the function $f(x, y)$ is defined as:

$$f(x,y) = \begin{cases} 1 & \text{if}(x \times y) < 0 \cap |x - y| \geq th \\ 0 & \text{otherwise.} \end{cases} \tag{2.7}$$

#### 2.7.3.2 Frequency Domain Features

The other type of features extracted during motion classification are the frequency domain features. Phinyomark *et al.* [70] found that frequency domain features are not well suited for EMG signal classification due to some of the features having the same discrimination as most time domain features. However, they also found that two features in the frequency domain have the ability to provide some useful information for EMG signal classification. These features are the following:

**Mean Frequency (MNF)** The mean frequency of the signal is the average frequency of the EMG power spectrum [70]. It is calculated as follows:

$$MNF = \sum_{i=1}^{M} f_i P_i \bigg/ \sum_{i=1}^{M} P_i, \tag{2.8}$$

where $M$ is the length of the frequency bin, $f_i$ is the frequency of the power spectrum at bin $i$, and $P_i$ is the EMG power spectrum at frequency bin $i$.

**Power Spectrum Ratio (PSR)** The power spectrum ratio represents the ratio between the maximum value of the EMG power spectrum and the whole energy of the EMG power spectrum [70]. The PSR is calculated as follows:

$$PSR = \frac{P_0}{P} = \sum_{i=f_0-n}^{f_0+n} P_i \bigg/ \sum_{i=E_1}^{E_2} P_i, \tag{2.9}$$

where $P$ is the energy of the EMG power spectrum, which can lie within the range of $E_1 = 20$ Hz and $E_2 = 500$ Hz [72]. On the other hand, $P_0$ is the energy near the maximum value of the

EMG power spectrum, $n$ is the integral limit, and $f_0$ is the frequency with the maximum power spectrum in a frequency bin of length $M$ [70].

**Peak Frequency (PSR)** The peak frequency, is the frequency at which the maximum power occurs [70]. The PKF is given by:

$$PKF = \max(P_i) \qquad i = 1,\ \ldots,\ M. \tag{2.10}$$

### 2.7.4 Classification Methods

After features have been extracted, they need to be used as inputs to a classifier in order to be mapped to new known gestures. This section reviews some of the common pattern recognition classification methods used for the control of wearable devices.

#### 2.7.4.1 Support Vector Machines

Support Vector Machines (SVM) are a powerful classification method for solving non-linear problems regarding pattern recognition. SVM uses separating hyperplanes to distinctly classify between data points corresponding to different classes. To properly classify these data points while also increasing the probability of correctly classifying new data, SVM tries to find the ideal separating hyperplane that maximizes the distance between samples of different classes. This distance is also known as the classification margin, and it is defined by the support vectors, *i.e.*, data that are closest to the hyperplane. However, because the support vectors are the most difficult data to classify, the classification margin is tuned to allow for some violations.

Although SVM can be used to solve linear classification problems, most of the applications involve non-linear classification problems. In other words, the vast majority of pattern recognition applications involves data samples that are not linearly separable. Therefore, SVM solve this issue by using kernel functions. These functions allow the data samples to be projected into a high dimensional space, where data that have some sort of similarity between each other are grouped together. This allows for a non-linear classification problem to be treated as a linear classification problem. Commonly used kernel functions include the linear kernel, the polynomial kernel, and the Gaussian or radial basis function kernel [73].

SVM have been successfully used for many EMG applications including motion classification for the control of wearable devices [68, 74]

#### 2.7.4.2 Least Squares Support Vector Machines

The Least Squares Support Vector Machines (LS-SVM) classification algorithm proposed by Suykens and Vandewalle [75] is a variation of the SVM classifier. It was introduced to solve one of the major drawbacks of SVM, which is the high computational burden of its optimization problem. LS-SVM solve this issue by approaching the optimization problem using equality instead of inequality constraints, and a sum of squared errors [76]. This reformulation allows the solutions of the SVM classification problem to be obtained using a system of linear equations. Because of this, LS-SVM constitute the basis algorithm for most user-independent classification problems.

#### 2.7.4.3 Multilayer Perceptron Neural Networks

Previous research in the field of EMG pattern classification [77] has successfully implemented Artificial Neural Networks (ANN) as a classification algorithm due to its high generalization abilities over large data sets that are not linearly separable. Multilayer perceptron (MLP) networks are a type of feedforward network that consists of an input layer, one or multiple hidden layers, and an output layer (Figure 2.4).

Training of a MLP network occurs through an iterative process that consists of two steps: the forward propagation phase and the back propagation phase. During the forward phase, information used as the input signal travels through the network layer by layer in a forward direction. The output of each layer is given by an activation function, whose choice depends on the application. Different activation functions exist, however the commonly used one for bilinear problems is the sigmoid function [78], whereas for multiclass classification problems, the softmax function [79] is employed. On the other hand, during the back propagation phase, the output of the network travels from the output layer to the input layer, with the purpose of computing a set of weights that minimizes the classification error. This is achieved by using different algorithms, with the backpropagation algorithm being the most commonly used [80].

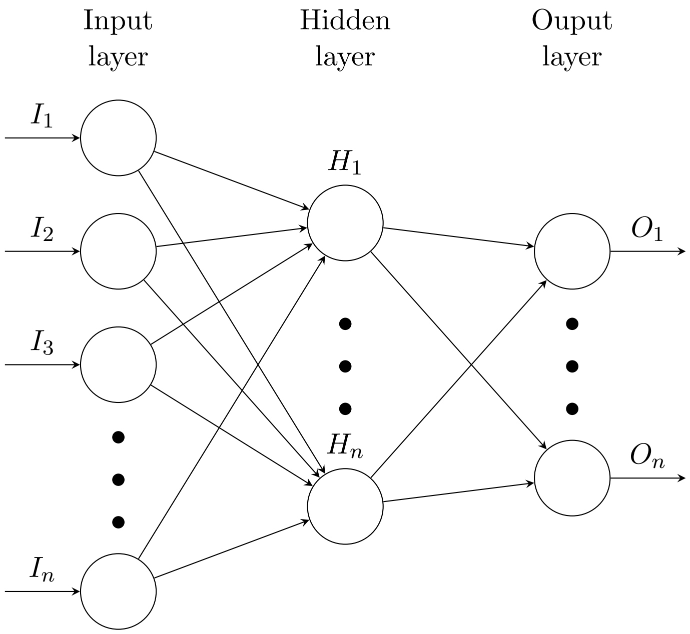


Figure 2.4: MLP network model for multiple inputs and outputs.

## 2.8 User-Independent Classification Methods

In the previous section, the process of EMG pattern recognition was described. While the same process is true for a user-independent EMG pattern recognition, the classification methods used differ from the traditional ones. In the following sections some of these classification methods are presented.

### 2.8.1 Particle Adaptive Classifier

The particle adaptive classifier (PAC) is an adaptive learning classification method based on LS-SVM, and incremental learning. The concept of incremental learning consists of taking the support vectors of a pretrained model and combining them with a new batch of incoming data. Then, a new classification model is trained using these new data. However, this method will eventually lead to a higher risk of misclassification due to the concept drift, which is caused by the change in the data distribution every time a new model is trained [24].

Therefore, Huang *et al.* [23] proposed an alternative to this method that consisted of using a new type of incremental learning that they called universal incremental learning. This new method consists of extracting a representative sample set from a pretrained LS-SVM model using clustering algorithms such as the $k$ nearest neighbours (kNN) [25]. Then, this representative sample set, also known as the particle set, is used as the new predictive model used to train a new classifier. When a new data sample is ready to be classified, the universal incremental learning compares how close this data sample is to any of the samples in the particle set. These distances are measured in the kernel space. If the distance between the sample and a particle is below a predefined threshold, the particle set is updated by substituting the sample particle with the new data sample. Finally, if the particle set was updated, a new predictive model is trained.

The advantage of the PAC method is that it is possible to avoid not only the concept drift, but also the computational cost of training new predictive models. This is because the size of the particle set used is small, which yields a reduced number of support vectors.

### 2.8.2 Adaptive LS-SVM

Another adaptive approach for a user-independent classification scenario is based on the work proposed by Tomasi *et al.* [81]. This approach assumes that a database of pretrained LS-SVM models exists, which can be used as the starting point to train a new predictive model for a new user. To properly work, the adaptive LS-SVM requires the use of a calibration set, which is formed whenever a new user performs a defined set of motions. Then, each of the pretrained models in the database are used to classify the data obtained from the calibration phase. Because the user performs the same motions used to train the pretrained models, it is assumed that at least the new distribution of the data is close to at least one of the existing models.

The Adaptive LS-SVM proved to improve the classification performance of new data when used in a user-independent scenario [81]. However, the main drawback of this method is that it relies on the assumption that at least one previous model matches the new data distribution, which as shown in [81], is not always the case.

### 2.8.3 Bilinear Models

Matsubara *et al.* [82] proposed a different user-independent classification method that consists of using bilinear models (BM) to represent the EMG signal. They based their work on a similar

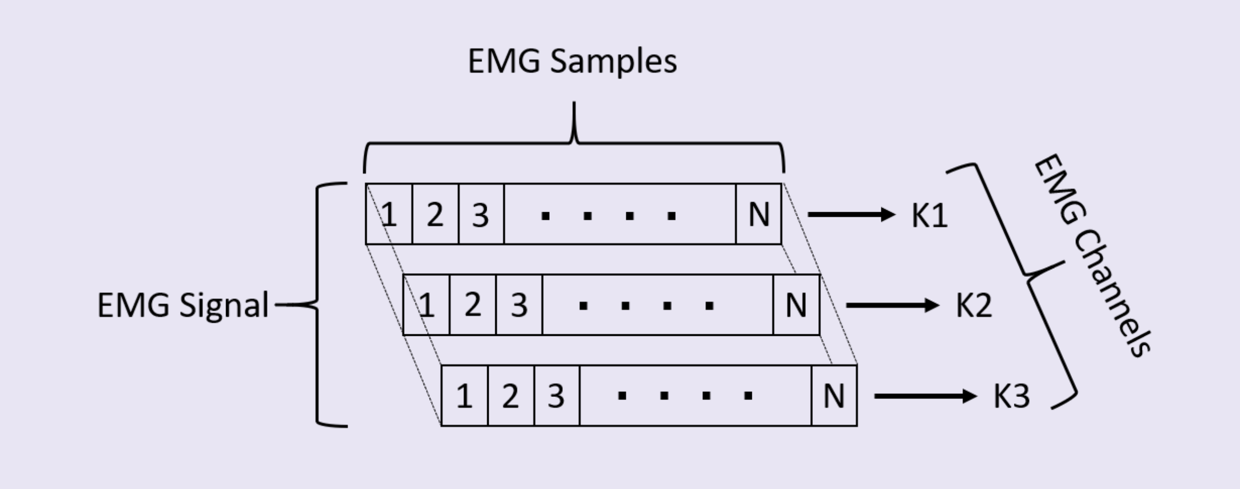


Figure 2.5: Representation of the EMG symmetric model. The EMG signal can be represented as a multidimensional array where each row K, which represents one EMG channel, contains the same number of samples.

approach used for computer vision applications [83].

To implement their technique, they represented a multichannel EMG signal (Figure 2.5) using a symmetric bilinear model, as follows:

$$\Psi^{SC} = \sum_{i=1}^{I} \sum_{j=1}^{J} \mathrm{w}_{ijk} \cdot z_i^S \cdot x_j^C, \qquad (2.11)$$

where $\Psi_k^{SC}$ represents the EMG signal of channel $k$, $z^S \in \mathbb{R}^I$ and $x^C \in \mathbb{R}^J$ indicate the parameters, respectively, of the style (user-dependent factors) and content (motion-dependent factors) vectors. Also, by denoting $W \in \mathbb{R}^{I \times J}$ as the parameter matrix of the bilinear model with entries $\mathrm{w}_{ijk}$, each channel $k$ of $\Psi^{SC}$ can be represented in a vectorized form, as follows:

$$\Psi_k^{SC} = z^{S^T} \cdot \mathrm{W}_k \cdot x^C. \qquad (2.12)$$

By using the bilinear model representation, Matsubara *et al.* [82] were able to effectively classify motion data using the content variables as a type of new feature inputs for a SVM classification model. However, they found that this approach is heavily dependent on the setup of the EMG collecting device. In this sense, if the EMG electrodes were not placed in the exact same location, the classification performance would drop.

## 2.9 Conclusion

This chapter reviewed the anatomy of the wrist, and some of the techniques used in its rehabilitation following a stroke episode. The different types of robot assistive rehabilitation devices as well as their control methods were described. Finally, EMG pattern recognition and some of the user-independent classification algorithms were reviewed. In the following chapter, an IMU and EMG-based sensor fusion technique used in combination with these user-independent classification methods will be explored.

# Chapter 3

# Data Collection and Processing

This chapter describes the procedure for EMG and IMU data collection and processing. The following sections present a description of the equipment used, the experimental protocol, which includes participant recruitment and data collection procedures, and signal processing methods implemented to move forward to the development of a user-independent interface based on different classification models. The code used in this section is shown in Appendices B.1 and B.2.

## 3.1 Equipment

### 3.1.1 The Myo Armband

The Myo Armband (Fig. 3.1), which is a gesture recognition band comprised of eight dry stainless steel medical grade EMG sensors and one 9 degree-of-freedom (DOF) IMU, was used to collect data during the trials. Each EMG sensor was sampled at a frequency of 200 Hz, and output an eight-bit unitless integer value that ranges from -128 to 127 representing the level of activation of the muscle being sensed. The 9 DOF IMU contains a three-axis accelerometer, a three-axis gyroscope, and a three-axis magnetometer, each one sampled at a frequency of 50 Hz. Furthermore, six sixteen-bit analog-to-digital converter (ADC) are used to digitize each axis of the accelerometer and gyroscope elements of the IMU, while three thirteen-bit ADC are used for the magnetometer outputs [84].

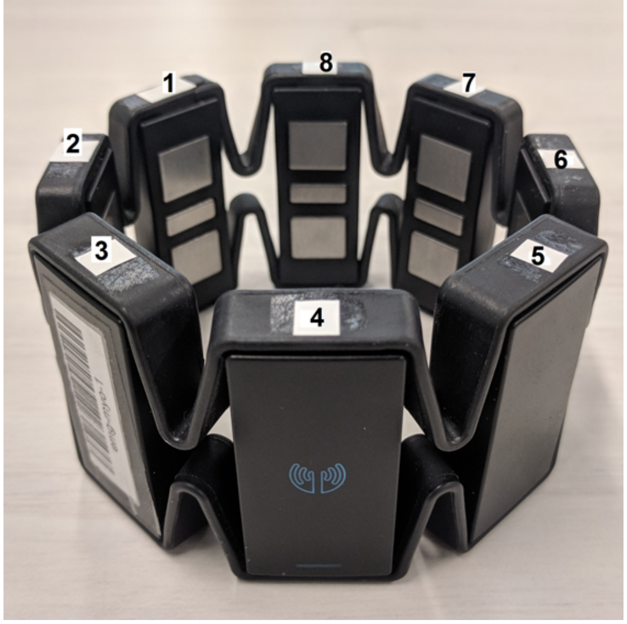
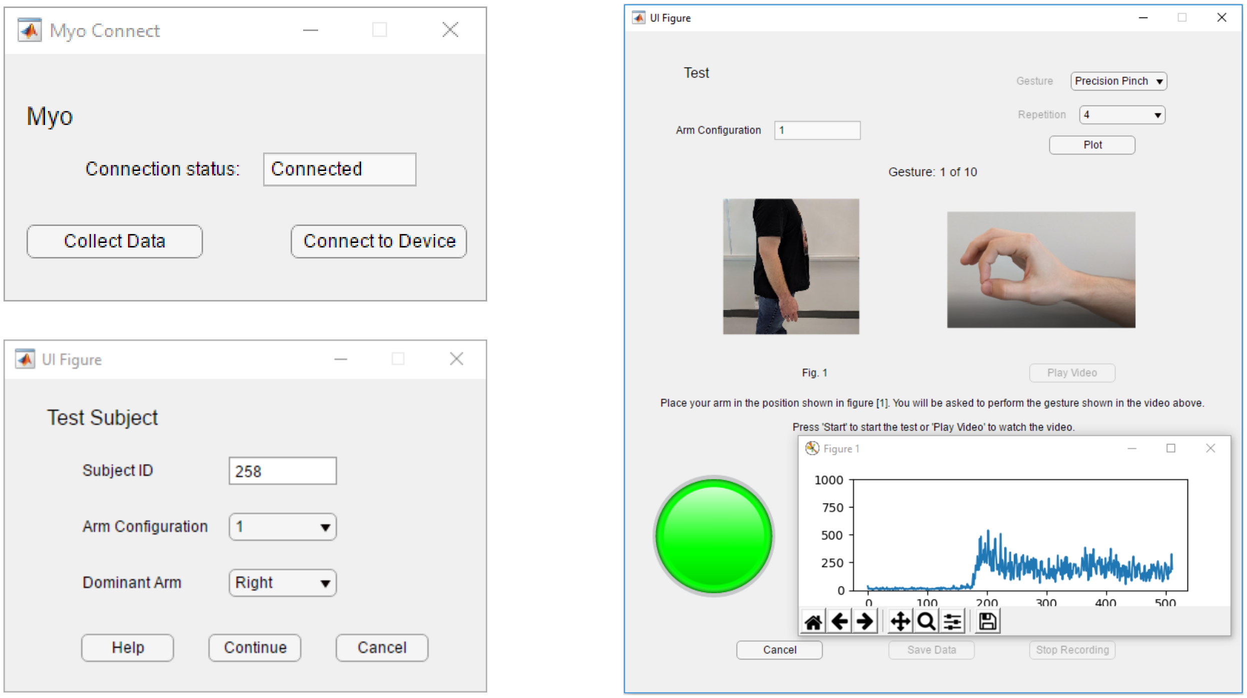


Figure 3.1: The Myo Armband showing the labels for each sensor. The IMU is located within Sensor 4.



Myo Connect

Myo

Connection status: Connected

UI Figure

Test Subject

Subject ID: 258

Arm Configuration: 1

Dominant Arm: Right

UI Figure

Test

Gesture: Precision Pinch

Repetition: 4

Arm Configuration: 1

Gesture: 1 of 10

Fig. 1

Place your arm in the position shown in figure [1]. You will be asked to perform the gesture shown in the video above.

Press 'Start' to start the test or 'Play Video' to watch the video.

Figure 3.2: Custom data acquisition software developed. Figures on the left represent the panel used to establish communication with the Myo Armband (top), and the experimental setup panel (bottom). The figure on the right shows the data acquisition panel.

### 3.1.2 Data Analysis Software

Data from the Myo Armband were streamed via Bluetooth 4.0 to a 3.40 GHz Intel Core$^{\text{TM}}$ i7 PC running Windows 10 with 8 GB memory RAM. To complete the collection of the data, a custom data acquisition GUI (Fig. 3.2) was developed in MATLAB R2017b using the App Designer toolbox and the Myo SDK MATLAB Mex Wrapper toolbox [85].

## 3.2 Participant Recruitment

Trials began following approval from the Human Research Ethics Board at Western University (Appendix A.1). Participants were recruited via email advertisement over a period of 7 months from October 2018 to April 2019. Only healthy subjects over the age of 18 years old, with no previous injuries of the shoulder, elbow or wrist, nor neurological disorders, were considered for the trials. These exclusion criteria were implemented because musculoskeletal or neurological disorders in these joints could hinder the ability to perform the gestures at their full range of motion.

## 3.3 Experimental Protocol

Following participant consent, data collection began at the Wearable Biomechatronics Laboratory at Western University. Each participant provided information about their age, dominant hand, sex, weight, height, waist circumference, wrist circumference, forearm circumference, arm length, and any information about prior upper-limb injuries. Such information was collected as a standard procedure because it may account for differences in biological signals that may be useful for future studies. However, only a portion of this information was employed in this study. Table 3.1 shows a summary of the participants' information.

Table 3.1: Summary of participant information.

| Sex | Dominant Hand | Age (yrs) | Weight (kg) | Height (cm) | Wrist Circumference (cm) | Forearm Circumference (cm) |
|---|---|---|---|---|---|---|
| 18 Male | 22 Right | 23.70 ± 3.92 | 71.30 ± 12.13 | 173.67 ± 10.51 | 16.42 ± 1.20 | 26.41 ± 2.81 |
| 6 Female | 2 Left | | | | | |

### 3.3.1 Myo Armband Placement

During the trials, each participant wore the Myo Armband on their dominant arm just bellow the elbow joint (Fig. 3.3). To ensure consistency and avoid further variability in the EMG readings during the data collection phase, the fourth sensor of the Myo Armband (Fig. 3.1) was positioned above the Extensor Carpi Ulnaris muscle, which was located using palpation techniques. Therefore and depending on the dominant arm, each remaining sensor of the Myo Armband was positioned over the muscles described in Table 3.2, according to [86].

Table 3.2: Myo Armband sensor placement with respect to the forearm muscles depending on hand dominance [86].

| **Sensor #** | **Right Arm Muscles** | **Left Arm Muscles** |
|---|---|---|
| 1 | Flexor Carpi Ulnaris | Brachioradialis |
| | | Flexor Digitorum Superficialis |
| 2 | Anconeus | Extensor Carpi Radialis Longus |
| | Flexor Carpi Ulnaris | Extensor Carpi Radialis Brevis |
| | | Brachioradialis |
| 3 | Extensor Digiti Minimi | Extensor Digitorum |
| | | Extensor Pollicis Longus |
| 4 | Extensor Digitorum | Extensor Digitorum |
| | Extensor Carpi Ulnaris | Extensor Carpi ulnaris |
| 5 | Extensor Digitorum | Extensor Digiti Minimi |
| | Extensor Pollicis Longus | |
| 6 | Extensor Carpi Radialis Longus | Anconeus |
| | Extensor Carpi Radialis Brevis | Flexor Carpi Ulnaris |
| | Brachioradialis | |
| 7 | Brachioradialis | Flexor Carpi Ulnaris |
| | Flexor Digitorum Superficialis | |
| 8 | Flexor Carpi Radialis | Flexor Carpi Radialis |

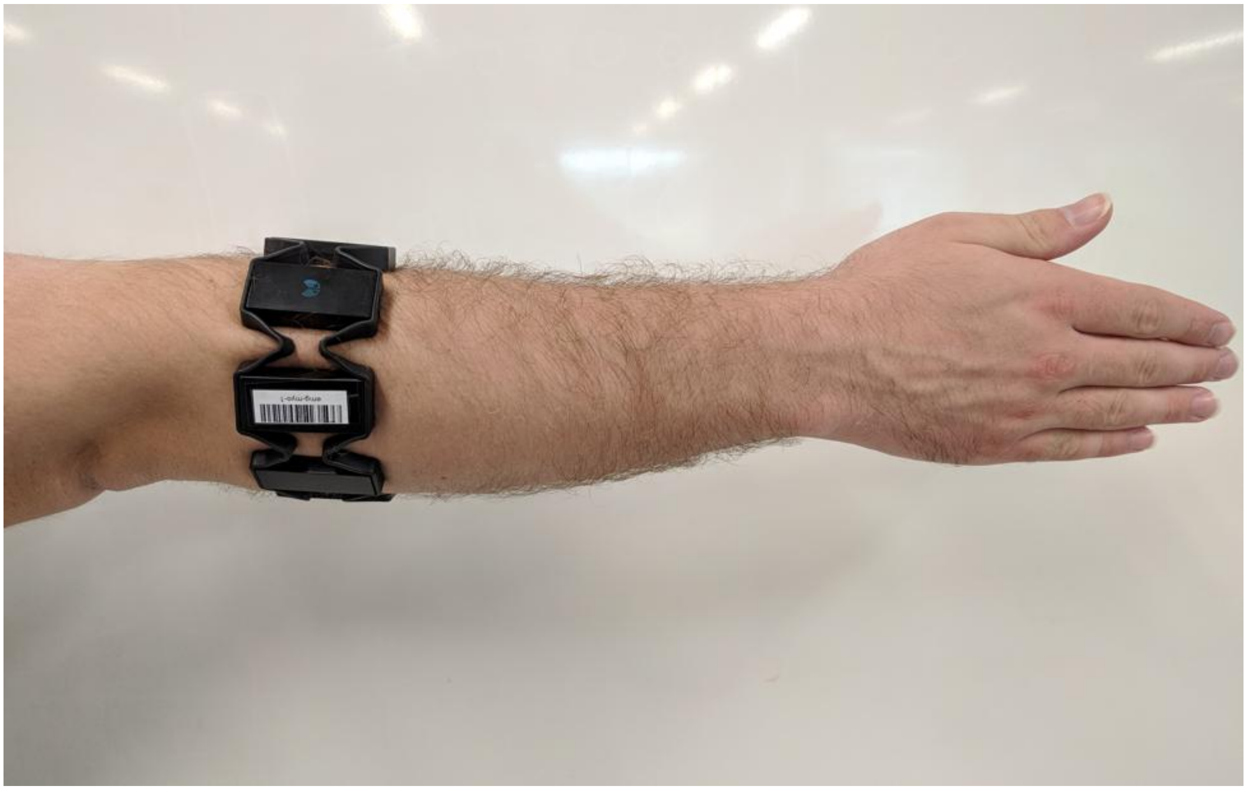


Figure 3.3: Placement of the Myo Armband on the user's dominant arm.

### 3.3.2 Gestures

After the initial setup, participants performed ten hand gestures, which included a set of six wrist motions and four finger motions (Fig. 3.4). The order of each motion was randomized for each participant. Prior to the data collection, participants were instructed to perform each gesture at a moderate and repeatable force level, *i.e.*, there was no restriction on the amount of force exerted by each participant. For each gesture, ten consecutive repetitions were performed, and each repetition was held for five seconds with three seconds of resting time between repetitions. The completion of the ten repetitions of the ten gestures was defined as a trial. Each trial was performed in four different arm positions (Fig. 3.5) to prevent degradation of the classification algorithm during the pattern recognition step, since it has been proven that changing the arm position affects the classification performance [87]. Finally, each trial was video recorded to review the motions performed by the participant in case any abnormalities in the data were found during the data analysis.

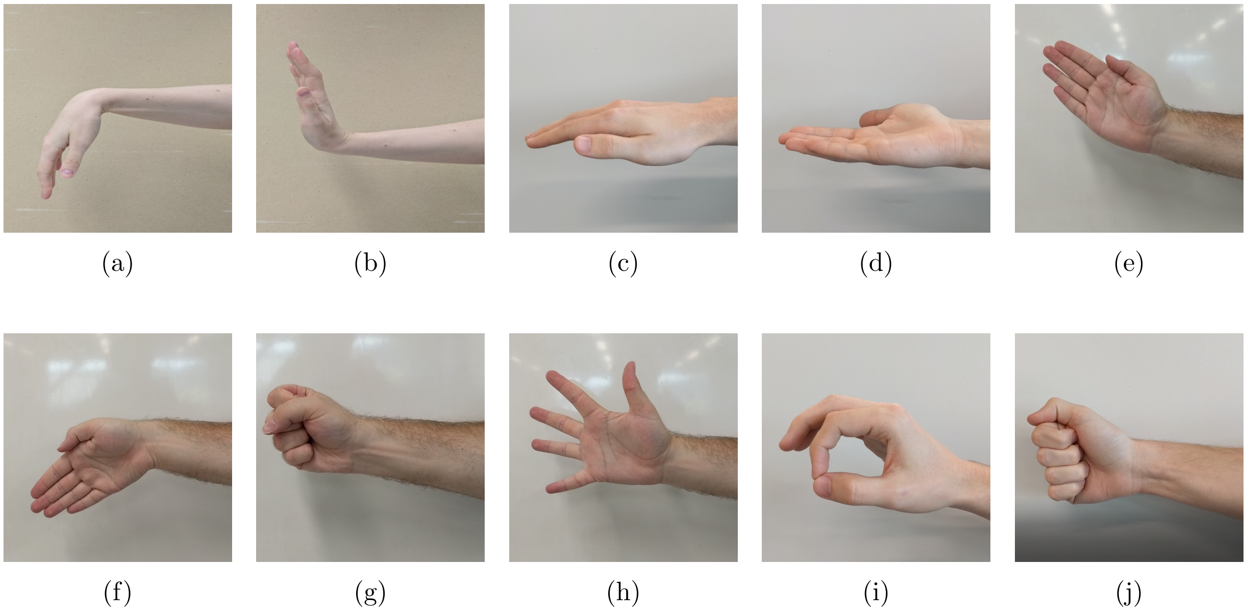


Figure 3.4: Wrist and finger motions that were recorded; a) Wrist Flexion (WF), b) Wrist Extension (WE), c) Wrist Pronation (WP), d) Wrist Supination (WS), e) Wrist Adduction (WAd), f) Wrist Abduction (WAb), g) Hand Closed (HC), h) Hand Open (HO), i) Precision Pinch (PP), j) Key Pinch (KP).

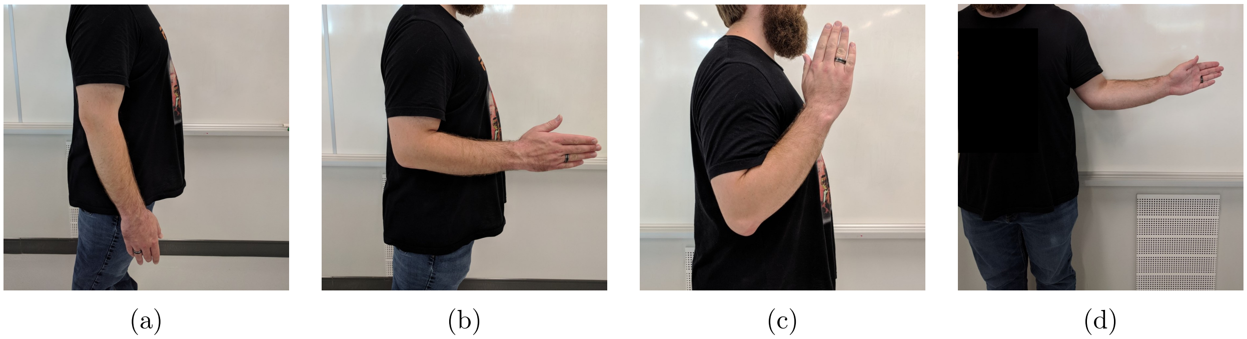


Figure 3.5: Arm positions used during data acquisition; a) forearm at full extension (0°), b) forearm flexed at 90°, c) forearm flexed at 135°, d) forearm at 90° flexion while externally rotating the shoulder at 50°.

## 3.4 Data Processing

Collected EMG data were filtered using a 60 Hz notch filter to remove power line interference, and a $4^{th}$ order high-pass Butterworth filter with a cut-off frequency of 20 Hz. Accelerometer and gyroscope data were filtered using a $4^{th}$ order Butterworth band-pass filter with cut-off frequencies of 0.2 Hz and 15 Hz to remove the gross orientation effects [88]. A summary of the data collection and processing is shown in Figure 3.6.

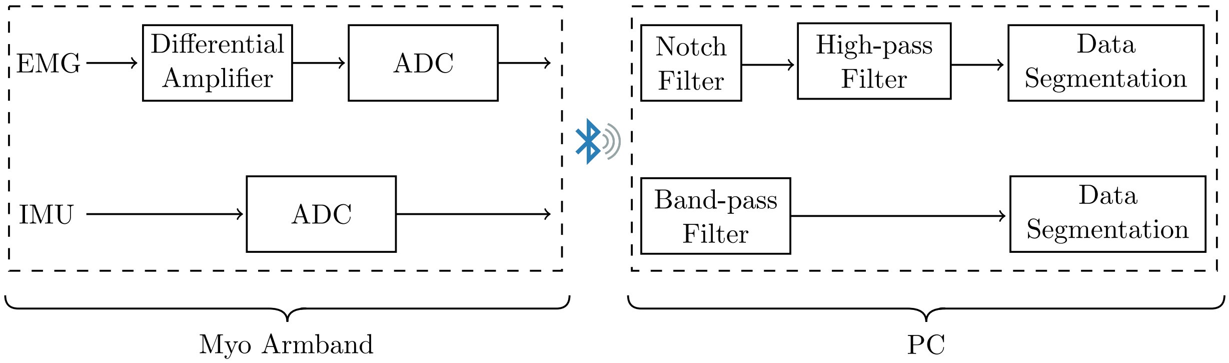


Figure 3.6: Block diagram of the EMG and IMU data collection and processing. EMG and IMU data collected from the Myo Armband were sent via Bluetooth to a PC running MATLAB. Then, the data were preprocessed before proceeding to the data segmentation step.

### 3.4.1 Signal Segmentation

The active region of the gesture, *i.e.*, sections of the EMG and IMU data where the gesture motion was being performed, was divided into segments using the following procedure. First, the active area was computed by conditioning each EMG channel with the TKEO using Equation (2.1), and then passed through a $4^{th}$ order Butterworth low-pass filter with a 50 Hz cut-off frequency, before it was finally rectified (Fig. 3.7), to improve the accuracy detection of the motion onset [65].

Then, the average absolute value of all the channels for each sample was computed using Equation (3.1), resulting in the signal shown in Fig. 3.8a.

$$\Psi_{avg} = \frac{1}{n} \sum_{i=1}^{n} \left| \Psi_i \left( t \right) \right| . \tag{3.1}$$

In this equation, $n$ is the number of channels and $\Psi_i(t)$ represents the measure of sample $t$ of the sensor $i$. Finally, $\Psi_{avg}$ was smoothed using Equation (3.2), with a window size $W = 60$, to obtain its root mean square $\Psi_{rms}$ (Fig. 3.8b).

$$\Psi_{rms} = \sqrt{\frac{1}{W} \sum_{j=t}^{t+W-1} \Psi_{avg}^2 \left( j \right)}. \tag{3.2}$$

The motion onset and offset were obtained from $\Psi_{rms}$ using a variation of the double threshold technique proposed in [89]. The onset threshold was set to 20% of the average of all the peaks (local

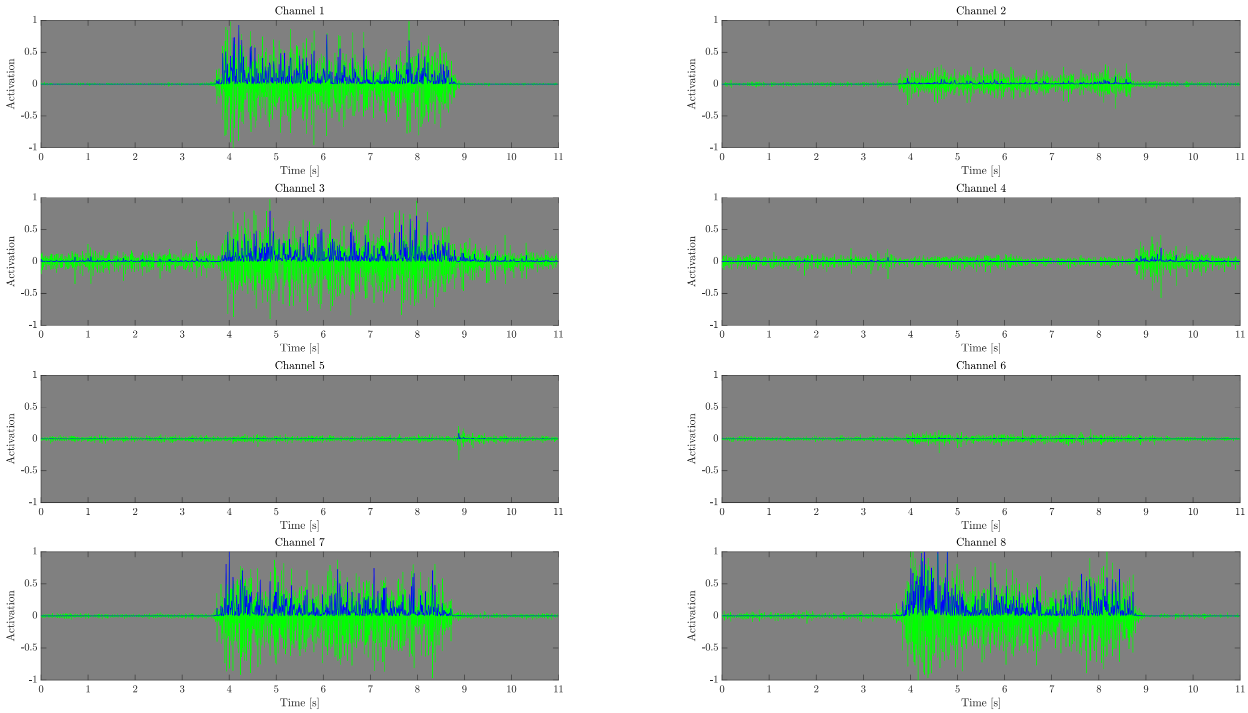
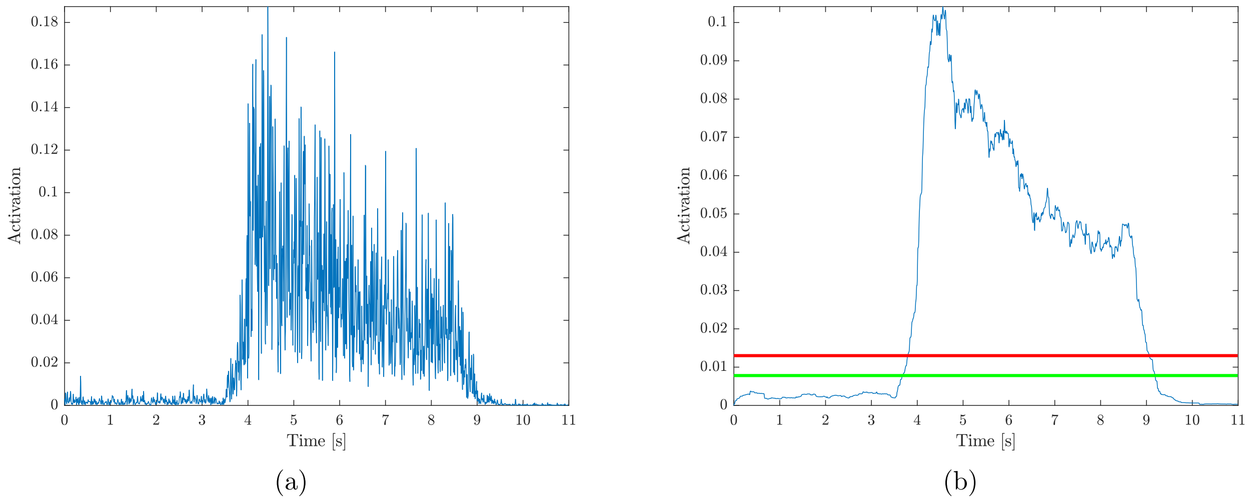


Figure 3.7: Sample data set showing a participant's EMG signals after performing the WF gesture. The first column contains the EMG signals of channels 1, 3, 5, and 7; the second column contains the EMG signals of channels 2, 4, 6 and, 8. The EMG signal is represented in green, and the conditioned EMG signal with TKEO is represented in blue.



(a) (b)

Figure 3.8: Averaged EMG signal (a) and smoothed EMG signal (b) of a WF gesture sample data set. Motion onset and offset threshold are represented by the red and green lines, respectively, in (b).

maximas), whose values were above 10% of $\Psi_{rms}$ global maxima. Then, the offset threshold was set to 60% of the onset threshold. These percentages were determined experimentally. Afterwards, the indices of $\Psi_{rms}$ where the onset and offset occurred were taken and matched to every channel of the EMG (Figure 3.9).

To avoid any false positives due to involuntary motions, data from the active region that was less than 750 samples (3.75 seconds) was discarded. Furthermore, data longer than 1200 samples (6 seconds) was truncated to 1000 samples (5 seconds as determined in the experimental protocol) in case that the smoothed EMG signal failed to cross the offset threshold. In the event that the double threshold technique failed to detect an active region, it was defined as the area between samples 800 and 1650, which corresponds to the time frame when the participant was prompted to perform the gesture. Using this technique, the active region of 72.51% of the data sets was successfully detected.

To determine the onset and offset of the motion in the IMU data, the accelerometer and gyroscopic data were upsampled from 50 Hz to 200 Hz using a cubic spline, so that the number of

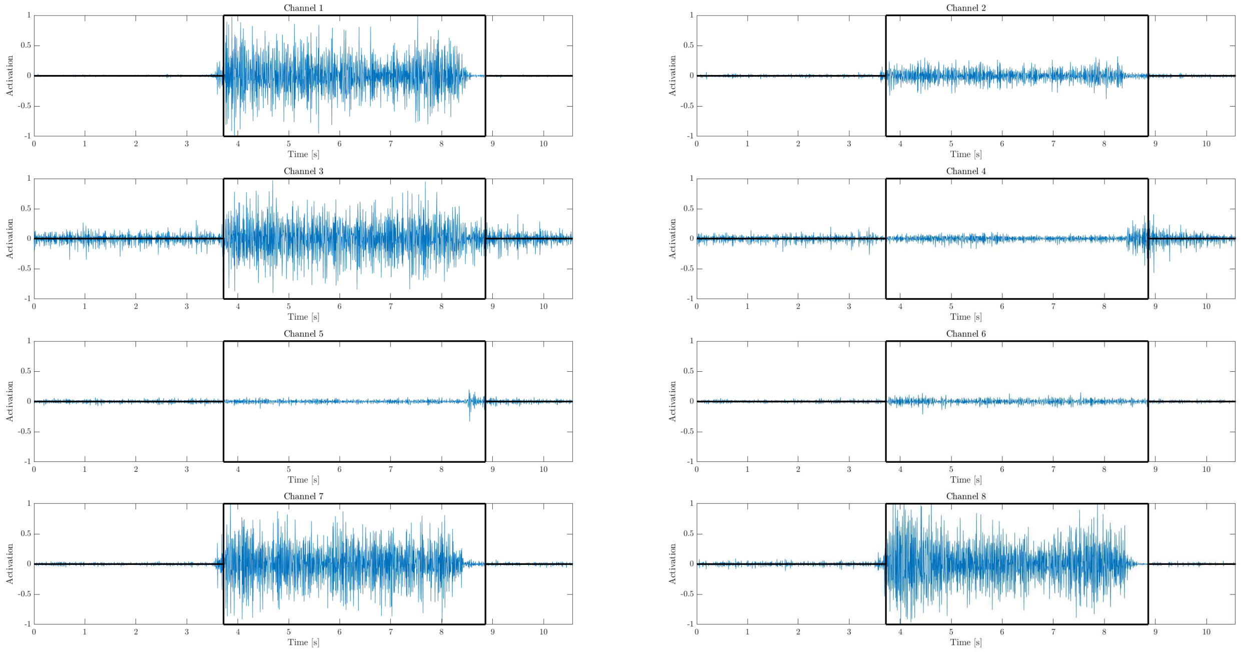


Figure 3.9: Sample data set showing the active region of each EMG channel after performing the WF gesture. The first column contains the EMG signal of channels 1, 3, 5, and 7; the second column contains the EMG signal of channels 2, 4, 6 and, 8. The active region of each EMG signal is within its corresponding black box.

samples of these data were the same as the EMG data. Then, the previously obtained onset and offset indices of the EMG were matched to the upsampled data.

Finally, each active segment was divided into overlapping windows of 250 ms with 50% overlap. This was done following the recommendations of Englehart and Hudgins [71], who stated that the maximum acceptable controller delay of upper-limb myoelectric devices should be 300 ms.

### 3.4.2 Feature Extraction

For both sensor modalities, time domain features were extracted from each window due to their low complexity and the fact that they do not require any transformation into the frequency domain [70], which reduces any extra computational resources. The following time domain features were extracted:

- From each EMG channel: MAV, MAVS, WL, $4^{th}$ order AR and ZC.
- From each axis of the IMU's accelerometer and gyroscope: MAV and WL.

The result was a vector of 64 features ([4 features + 4 AR coefficients] × 8 channels) for each window of the EMG data, and a vector of 12 features (2 features × 3 accelerometer axes + 2 features × 3 gyroscope axes) for each window of the IMU data. From these feature vectors, two data sets were developed. The first one was formed by the 64 features extracted from each window of the EMG data, and the second data set contained all 64 EMG features plus, the 12 extracted from the IMU data. In the case of the second data set, a feature-level fusion approach was employed by concatenating each feature vector to form a single vector of 76 features. By using this fusion level, correlated features can be detected better during the feature reduction phase [90]. A summary of the whole data collection process is shown in Figure 3.10.

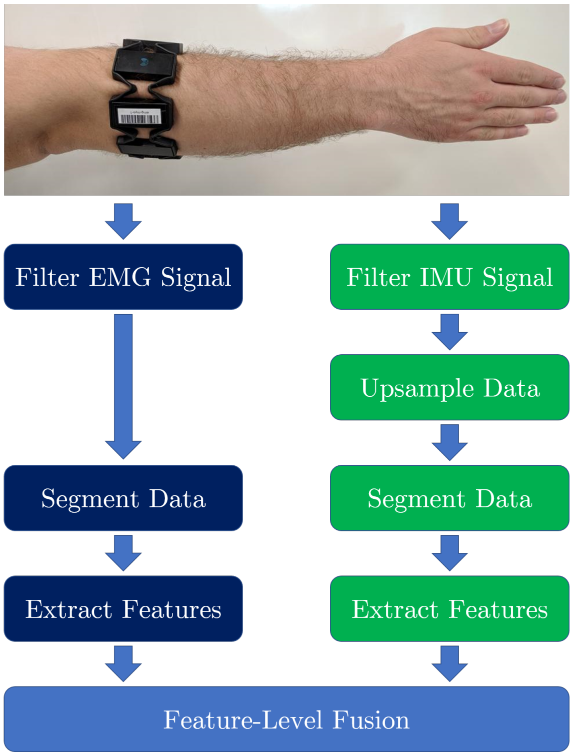


Figure 3.10: Summary of the data collection process.

### 3.4.3 Cross-Validation Sets

Before implementing the classification methods, five cross-validation folds were created from 22 participants using a 5-fold cross-validation method. Although data from a total of 24 participants were collected, data from two participants (Subject 8 and Subject 21) had to be removed due to an improper execution of the gestures during the data collection phase. This was identified from the data and confirmed when watching the videos of the trials. Therefore, each cross-validation fold was randomly formed, as follows:

- Cross-validation Fold 1 (CVF1): Subject 10 (S10), Subject 16 (S16), Subject 17 (S17), and Subject 18 (S18).
- Cross-validation Fold 2 (CVF2): Subject 13 (S13), Subject 14 (S14), Subject 19 (S19), Subject 22 (S22), and Subject 23 (S23).
- Cross-validation Fold 3 (CVF3): Subject 4 (S4), Subject 6 (S6), Subject 12 (S12), Subject 20 (S20), and Subject 24 (S24).

- Cross-validation Fold 4 (CVF4): Subject 3 (S3), Subject 5 (S5), Subject 7 (S7), and Subject 25 (S25).
- Cross-validation Fold 5 (CVF5): Subject 2 (S2), Subject 9 (S9), Subject 11 (S11), and Subject 15 (S15).

Then, training and testing of the classification methods occurred, as follows:

- Iteration 1 (Itr1): Trained on CVF2, CVF3, CVF4, and CVF5. Tested on CVF1.
- Iteration 2 (Itr2): Trained on CVF1, CVF3, CVF4, and CVF5. Tested on CVF2.
- Iteration 3 (Itr3): Trained on CVF1, CVF2, CVF4, and CVF5. Tested on CVF3.
- Iteration 4 (Itr4): Trained on CVF1, CVF2, CVF3, and CVF5. Tested on CVF4.
- Iteration 5 (Itr5): Trained on CVF1, CVF2, CVF3, and CVF4. Tested on CVF5.

## 3.5 Summary of Data Collection and Processing

This chapter described the methods of data collection and processing. An overview of the developed data collection software was given as well as the methods used for detecting the active region of the EMG and IMU signals. The methods employed for feature extraction on both sensor modalities, following signal segmentation, were described. Finally, the cross-validation models created from the 22 participants were explained. The next chapter will describe the implementation of four classification methods used to classify the EMG, and EMG and IMU datasets described in this chapter.

# Chapter 4

# Classification Methods

This chapter describes the implementation of four classification methods used on the EMG, and EMG and IMU datasets. The methods used to implement the particle adaptive classifier (PAC), the Adaptive LS-SVM, the Bilinear Model-based classifier, and the MLP networks classification methods are described. Furthermore, a description of how each model was evaluated is provided before moving towards the results and discussion chapter.

## 4.1 Classification Method: PAC

The first classification method that was explored was the Particle Adaptive Classifier (PAC), which was proposed by Huang *et al.* [23], and described in Section 2.8.1. To classify the EMG, and EMG and IMU datasets obtained in Chapter 3 using this classification method, code was developed in MATLAB R2017b. This programming code is shown in Appendix B. In the following sections, the procedure used to implement the PAC classification method during each of the cross-validation iterations presented in Section 3.4.3, is described.

### 4.1.1 Feature Normalization and Feature Reduction

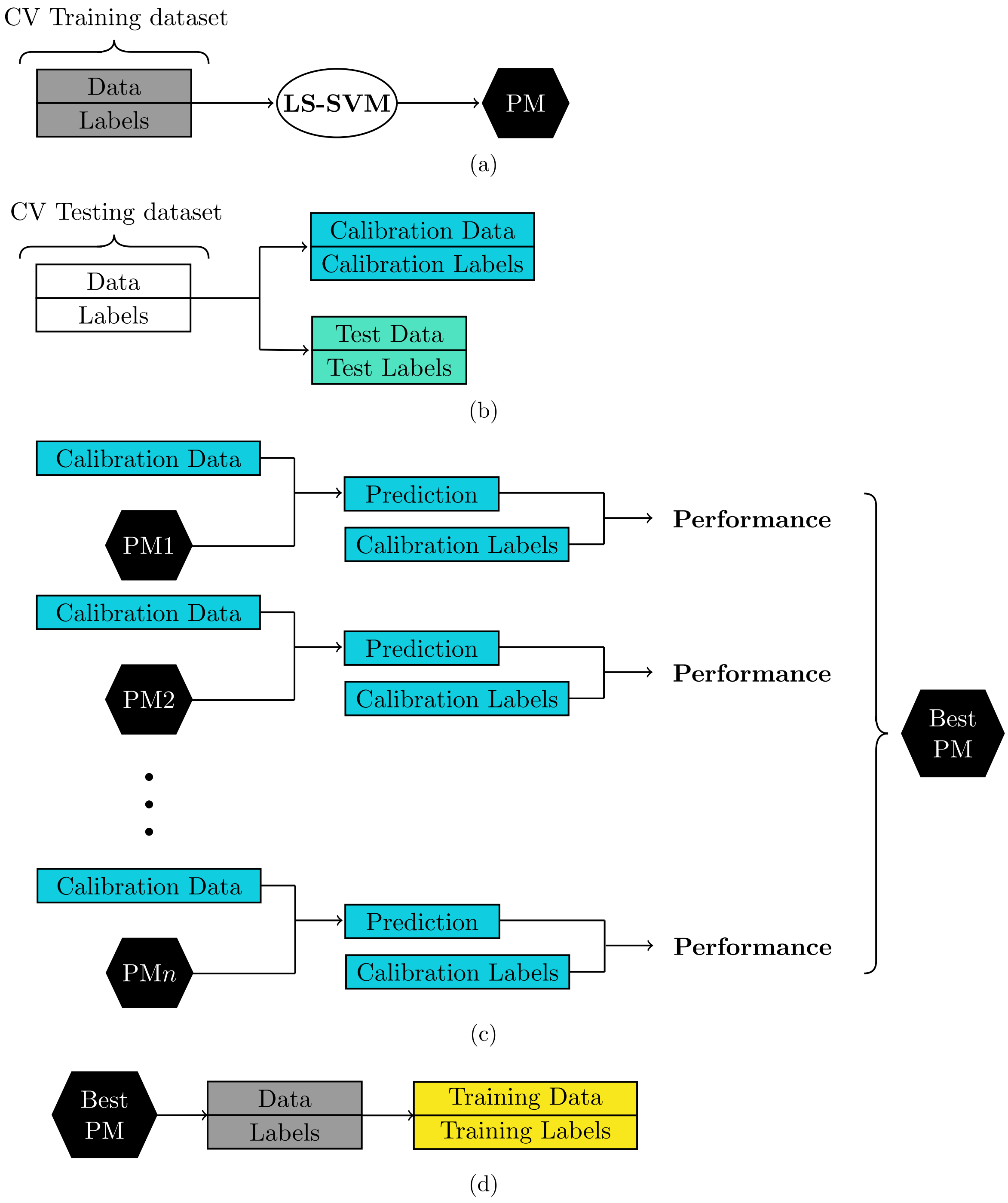
Following the feature extraction step described in Section 3.4.2, the features from the EMG, and EMG and IMU datasets were standardized using the $Z$ normalization, *i.e.*, after normalization, data had a mean equal to zero and a standard deviation equal to one. Then, these normalized features were scaled to the range of ±1 so that they had comparable range of values. Finally, these scaled features were reduced using the principal component analysis (PCA) procedure to speed up the learning process of the PAC classification method. A total of 17 principal components were used so that at least 95% of the data's variance was retained.

### 4.1.2 Calibration Phase

Once the data of the EMG, and EMG and IMU datasets were reduced using PCA, a calibration phase was performed. The purpose of this calibration phase was to find a subject in the cross-validation training set, whose data distribution was similar to that of a subject in the cross-validation testing set. This calibration phase was implemented as described below.

First, a LS-SVM predictive model (PM) was created for each each subject in the cross-validation training set (Figure 4.1a). Then, the data of each subject in the cross-validation testing set were split in two subsets of data: the calibration data $X_{cal}$, and the test data $X_{test}$ (Figure 4.1b). The calibration data were formed by two repetitions from each gesture in a random arm position (twenty motions in total), whereas the test data were formed by the rest of the motions. The reason for using two repetitions was because adaptive methods, like the PAC, require to be trained with a small portion of data that includes all gestures [82]. The amount of time required to perform these repetitions was equivalent to 2.6 minutes, which is less than the 2 hours used to collect all of the data for each subject.

Furthermore, the test data were set aside so it could be used later during the evaluation of the PAC classification method, as it was used to represent unseen data. On the other hand, the calibration data were classified using each of the previously created PM (Figure 4.1c). After finding the PM that performed best, the data from the subject in the cross-validation training set that was used to create this PM was selected as the training dataset $X_{train}$ (Figure 4.1d). This training dataset was then used to build the PM of the subject in the cross-validation testing set, whose data distribution was similar to that of the subject in the cross-validation training set. The process of building this PM is described in the next section.



(a)

(b)

(c)

(d)

Figure 4.1: Summary of the PAC calibration phase. The cross-validation training dataset, and the cross-validation testing dataset represent the data from a single subject. a) Creating a PM using the cross-validation training dataset. b) Splitting the cross-validation testing dataset. c) Finding the best PM using the calibration data. d) Selecting the best PM dataset as the training data.

### 4.1.3 The Representative Sample Set

To create a new PM for a new subject in the cross-validation testing set, a subset of the data from $X_{train}$ obtained in the previous section, was extracted. This subset of data, known as the representative sample set (RSS), served two purposes. The first one was to allow for a high efficiency training of the new PM by reducing its training time, while also keeping a similar classification performance [23]. The second purpose was to allow the update of the PM during the adaptive phase of the PAC classification method, in a simple manner.

To build the RSS, each training sample in the training set was divided into groups according to its labeled class. Then within each group, samples were split into clusters based on the kernel distance and the K-Medoids clustering algorithm developed in [91]. The kernel distance ([92, 93]) between two samples, $x_i$ and $x_j$, is defined as follows:

$$d(\phi(x_i), \phi(x_j)) = \|\phi(x_i) - \phi(x_j)\| = \sqrt{\mathcal{K}(x_i, x_i) + \mathcal{K}(x_j, x_j) - 2\mathcal{K}(x_i, x_j)}, \tag{4.1}$$

where $\phi(\cdot)$ represents the non-linear function that maps the samples to a high dimensional Hilbert space[1], and $\mathcal{K}(\cdot)$ is the kernel function. To compute the kernel distance, the radial basis function (RBF) kernel was employed.

$$\mathcal{K}(x_i, x_j) = \exp(-\gamma\|x_i - x_j\|^2), \tag{4.2}$$

where $\gamma$ is the kernel parameter. For this application, a value of $\gamma = 0.5$ was used, which was determined by using the grid search method during the hyperparameter optimization step [94]. This occurred when the PM from the best subject in the cross-validation training set was created during the calibration phase (Figure 4.1a). From Equation (4.2), Equation (4.1) was simplified as:

$$d(\phi(x_i), \phi(x_j)) = \sqrt{2 - 2\mathcal{K}(x_i, x_j)}. \tag{4.3}$$

After dividing the training set, $p$ percent of the samples from each cluster, which were obtained after using the previously mentioned K-Medoids algorithm, were selected as the representative

[1] The Hilbert space is an infinite dimensional inner product space, in which kernels can be represented as norm-based distances [92].

particles (RP) to form the RSS. For this application, a value of 10 and 23 were chosen for the number of clusters and the percentage $p$, respectively, following the recommendations in [23]. Finally, a new LS-SVM PM was created using the RSS as the training data, a regularization term[2] $\mathcal{C} = 2$ and a RBF with the kernel parameter $\gamma = 0.5$. These parameters were obtained after performing a grid search during the training step of the PM.

### 4.1.4 The Representative Particles Attractive Zone

Once the new PM and the RSS were created, an attractive zone around the RP had to be defined. The attractive zone was necessary to decide which RP had to be replaced. This was done to avoid the SVM concept drift, which is the change in the data distribution over time, that happens on every adaptive learning classifier. Therefore, the attractive zone of each RP in the RSS was defined following three principles, as follow:

a) For a new sample $x_N$, the closest RP had the same class label.

b) The distance between the new sample $x_N$ and the closest RP was small.

c) If the closest RP to the new sample $x_N$ had not been replaced in a certain amount of time, then the RP was more likely to be replaced.

The above conditions were implemented using the following procedure: First, each sample $x_N$ in $X_{cal}$, which was created in Section 4.1.2 (Figure 4.1a), was classified using the PM obtained in the previous section. Then, the nearest RP was found using the kernel distance function shown in Equation (4.3), as follows:

$$idx = argmin\{\exp(t/\lambda) \cdot KD_{r,c}\}, \tag{4.4}$$

where $idx$ is the index of the closest RP to $x_N$, $\lambda$ is the factor controlling the age of the RP, $t$ is a vector that contains the age of each RP, and $KD_{r,c}$ is a matrix that contains the distances of each sample $x_N$ (represented by the rows $r$) to each RP (represented by the columns $c$) in the RSS. Finally, if the distance of the sample $x_N$ was above a certain threshold $d_{Th}$, the RP

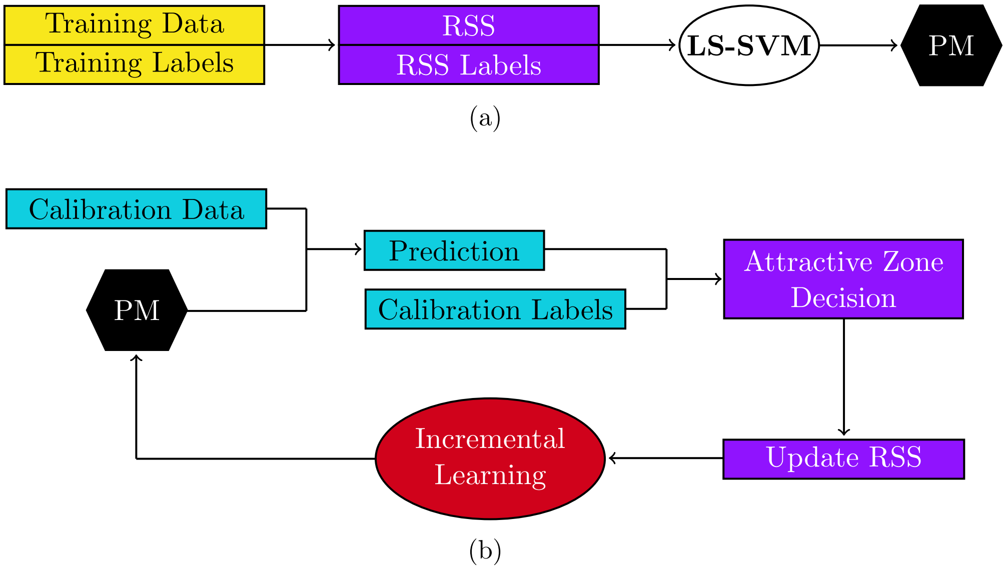
[2] The definition of the LS-SVM regularization term $\mathcal{C}$, and its derivation is explained in Appendix C.1.

in RSS was replaced by $x_N$ using the universal incremental learning algorithm developed in [23]. Equation (4.5) was applied to determine if the new sample $x_N$ was within the attractive zone of the nearest RP, using the values of 0.99 and $10^5$ for $d_{Th}$ and $\lambda$, respectively, as following the recommendations in [23].

$$D = d_{Th} - [\exp(t_{idx}/\lambda) \cdot KD_{r,idx}] \begin{cases} > 0 & \text{replace } RP_{idx} \text{ with } x_N \\ \leq 0 & \text{ignore } x_N. \end{cases} \tag{4.5}$$

The subscript *idx* in Equation (4.5) indicates the index of the variable. Finally, depending on the results of Equation (4.5), the vector $t$ in the index position *idx* was increased if the new sample $x_N$ was ignored, or reset to zero if the RP in the index position *idx* was replaced.

A summary of the general learning process of the PAC classification method presented in this section, and in the previous one, is shown in Figure 4.2.



(a)

(b)

Figure 4.2: Learning process of the PAC classification method. a) Creating the RSS, and training a PM. b) Updating the RSS and the PM using the attractive zone and incremental learning, respectively. Samples from the calibration data are predicted one at a time.

### 4.1.5 PAC Evaluation

After creating a PM for a user in the cross-validation testing set using the PAC classification method, the classification performance of the PM, obtained after iterating through each sample in $X_cal$ (Figure 4.2b), was evaluated using the testing data $X_{test}$ (Figure 4.3). The PAC classification method was repeated for each user in the cross-validation testing set, and for each cross-validation iteration described in Section 3.4.3. Furthermore, a new set of 7 gestures, which was created by removing the WAd, WAb and the PP gestures EMG, and EMG and IMU data from the datasets, were also analyzed using the PAC algorithm. This was done to further optimize the classification algorithm [95] by removing the gestures whose motions were controlled by the same muscles, *e.g.*, the Extensor Carpi Radialis Longus and the Extensor Carpi Radialis Brevis controlling both the Wrist Extension and the Wrist Adduction motions [96]. The results from the analysis of both, the 10 and 7 gesture sets using the EMG, and EMG and IMU sensor modalities are discussed in Chapter 5.

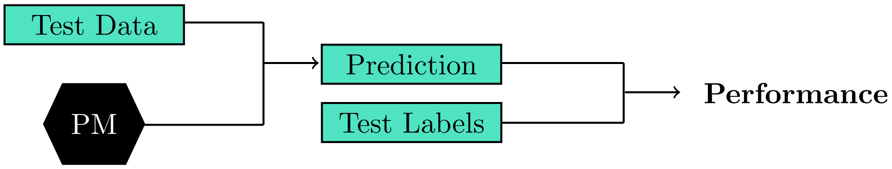


Figure 4.3: Evaluation of the classification performance of the PAC classification method.

## 4.2 Classification Method: Adaptive LS-SVM

Following the classification of the EMG, and EMG and IMU datasets using the PAC classification algorithm, the next classification method explored was the Adaptive LS-SVM. This method, introduced in Section 2.8.2, was first proposed by Tommasi *et al.* [81] as an approach towards an EMG user-independent classification scenario. In this work, the Adaptive LS-SVM classification method was implemented to classify the EMG dataset, and then adapted to classify the EMG and IMU dataset. To implement the Adaptive LS-SVM, code was developed in MATLAB R2017b.

This code can be found in Appendix B. The following sections[3] describe the procedure used to implement the Adaptive LS-SVM classification method.

### 4.2.1 Calibration Phase

Similarly to the PAC classification method, features from the EMG, and the EMG and IMU datasets were standardized, scaled, and reduced using the same procedure explained in Section 4.1.1, before implementing the Adaptive LS-SVM classification model. Then, a calibration phase was included in which a LS-SVM PM was created for each subject in the cross-validation training set (Figure 4.4a). Furthermore, the data from each subject in the cross-validation testing set were split in two subsets of data: the calibration data $X_{cal}$, and the test data $X_{test}$. As with the PAC calibration phase, two repetitions from each gesture in a random arm position were used to form $X_{cal}$. In the same way, all of the remaining gestures, *i.e.*, gestures that were not part of the calibration set, formed the testing set $X_{test}$ (Figure 4.4b). Finally, $X_{cal}$ was used to find a new PM model using the Adaptive LS-SVM classification method. On the other hand, $X_{test}$ was set aside so it could be used to asses the classification performance of the Adaptive LS-SVM PM. The process of implementing the Adaptive LS-SVM to build a PM is explained in the following sections.

### 4.2.2 The Leave-One-Out Prediction

To effectively apply the Adaptive LS-SVM classification method, a closed form solution for the leave-one-out prediction $\tilde{Y}_N$, which is the prediction of a sample $N$ when removed from a training set, was implemented. To find the close form solution, the system of linear equations shown in Equation (4.6), which represents the Adaptive LS-SVM formulation[4], was used a starting point, as follows:

[3] In the following sections, unless otherwise stated, a matrix is denoted with a capital letter. Further, when only one subscript is present, it indicates a specific column of the matrix, *e.g.*, $A_{ij}$ represent a matrix $A$ with the subscripts $(i, j)$ indicating its row and column, respectively. On the other hand $A_i$ is the same matrix $A$ with the subscript $i$ indicating the $i^{th}$ column.

[4] A thorough explanation of the mathematical derivations of the Adaptive LS-SVM formulation is given in Appendix C.2.

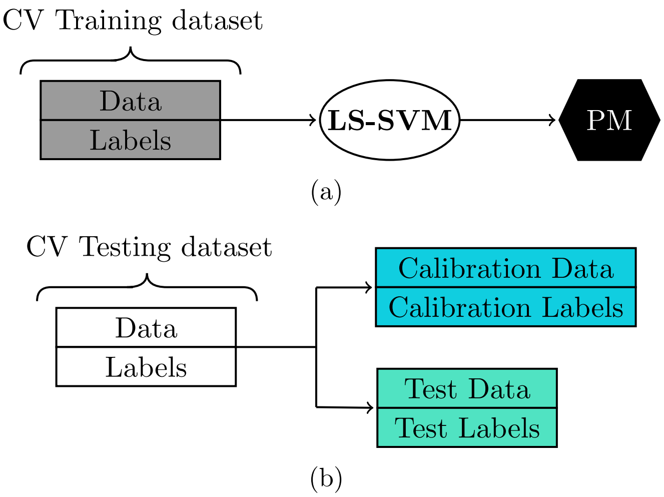


Figure 4.4: Summary of the Adaptive LS-SVM calibration phase. The cross-validation training dataset, and the cross-validation testing dataset represent the data from a single subject. a) Creating a PM using the cross-validation training dataset. b) Splitting the cross-validation testing dataset into $X_{cal}$ and $X_{test}$.

$$\begin{bmatrix} \mathcal{K} + \frac{\mathcal{I}}{\mathcal{C}} & \vec{1} \\ \vec{1}^T & 0 \end{bmatrix} \cdot \begin{bmatrix} A^T \\ \vec{b}^T \end{bmatrix} = \begin{bmatrix} Y^T - \beta \hat{Y}^T \\ 0 \end{bmatrix}, \tag{4.6}$$

where $\vec{1}$ represents a vector[5] of 1's, $\mathcal{I}$ is the identity matrix, $\mathcal{C}$ is the LS-SVM regularization parameter, and $\mathcal{K}$ is the kernel matrix with entries $\mathcal{K}_{i,j} = \mathcal{K}(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$, being $\mathcal{K}(x_i, x_j)$ the kernel function. Moreover, the coefficient $\beta$ represents the scaling factor that weighs a pretrained PM. Furthermore, the matrices $Y$, $\hat{Y}$, and $A \in \mathbb{R}^{G \times I}$, and $\vec{b} \in \mathbb{R}^{G \times 1}$. Here, $I$ is the number of training samples, and $G$ represents the number of motion labels (classes). Each row of the matrix $A$, and each element in $\vec{b}$ holds the $\alpha$'s (the LS-SVM support vectors), and the corresponding bias term of each $G$ class, respectively. Similarly, $Y$ and $\hat{Y}$ are the matrices containing the labels of each sample, and the prediction of each sample using a previous model, respectively, of $X_{cal}$. Each column of $Y$ and $\hat{Y}$ represents a vector with all of the elements equal to -1 except for the $g^{th}$ element, which is equal to 1 and indicates the corresponding class and the predicted class label, respectively, of sample $i$ in $X_{cal}$.

Denoting the first matrix to the left in Equation (4.6) as $M$, $A$ and $\vec{b}$ were obtained as follows:

[5] In this work, unless otherwise stated, a vector is defined as a column vector.

$$\begin{bmatrix} A^T \\ \vec{b}^T \end{bmatrix} = P \cdot \begin{bmatrix} Y^T - \beta \hat{Y}^T \\ 0 \end{bmatrix}. \tag{4.7}$$

Afterward, the matrix $P$, which is the inverse of the matrix $M$, was efficiently computed using the Cholesky factorization [97] and the inverse matrix block lemma using the procedure described bellow.

First, the matrix $M$ was divided into four submatrices, as follows:

$$M = \begin{bmatrix} M_A & M_B \\ M_C & M_D \end{bmatrix}, \tag{4.8}$$

where the matrices $M_A$, $M_B$, $M_C$, and $M_D$ are defined as follows:

$$M_A = \left[ \mathcal{K} + \frac{\mathcal{I}}{\mathcal{C}} \right] \tag{4.9}$$

$$M_B = \left[ \vec{1} \right] \tag{4.10}$$

$$M_C = \left[ \vec{1}^T \right] \tag{4.11}$$

$$M_D = 0. \tag{4.12}$$

Then, matrix $M_A$ was computed using $X_{cal}$ with a regularization term $\mathcal{C} = 2$, and a RBF kernel with parameter $\gamma = 0.5$, being used to form the kernel matrix $\mathcal{K}$. These parameters were the same ones used to train the PMs during the calibration phase described in section Section 4.2.1. Furthermore, given that the matrix $M_A$ was positive definite, *i.e.*, it was a symmetric matrix with all its eigenvalues being positive, the Cholesky method was used to factorize the matrix into a lower triangular matrix $L$ that satisfied the following equation:

$$M_A = L \cdot L^T. \tag{4.13}$$

Matrix $L$ was then used to efficiently compute the inverse of matrix $M_A$ using Equation (4.14):

$$M_A^{-1} = L^{-T} \cdot L^{-1}, \tag{4.14}$$

where $L^{-T}$ is the transpose of the inverse of matrix $L$. Using $M_A^{-1}$, matrix $P$ was then given by the following equation:

$$P = \begin{bmatrix} M_A^{-1} + M_A^{-1} \cdot M_B \cdot \mathcal{S}_{M_A}^{-1} \cdot M_C \cdot M_A^{-1} & -M_A^{-1} \cdot M_B \cdot \mathcal{S}_{M_A}^{-1} \\ \mathcal{S}_{M_A}^{-1} \cdot M_C \cdot M_A^{-1} & \mathcal{S}_{M_A}^{-1} \end{bmatrix}, \tag{4.15}$$

where $\mathcal{S}_{M_A} = -M_C \cdot M_A^{-1} \cdot M_B$ is the Schur complement of $M_A$. Finally, following the same procedure as in [98], and noting that the order in which the training samples are presented in Equation (4.6), does not affect the prediction outcomes, the closed form solution for the leave-one-out prediction $\tilde{Y}_i$ on sample $i$ when removed from the training set was given by the following equation:

$$\tilde{Y}_i = Y_i - \frac{A_i}{P_{ii}}. \tag{4.16}$$

Having $A = A' - \beta A''$, the leave-one-out prediction was then represented as follows:

$$\tilde{Y}_i = Y_i - \frac{A_i'}{P_{ii}} + \beta \frac{A_i''}{P_{ii}}. \tag{4.17}$$

### 4.2.3 Adaptive LS-SVM From Multiple Subjects

Having found the closed form of the leave-one-out prediction shown in Equation (4.17), it was then modified to discriminate between $G$ classes while also including information from previous $K$ models, as follows:

$$\tilde{Y}_i = Y_i - \frac{A_i'}{P_{ii}} + \sum_{k=1}^{K} \vec{\beta}^{(k)} \frac{A_i''^{(k)}}{P_{ii}} \qquad \forall k = 1, \ldots, K, \tag{4.18}$$

where $\vec{\beta}$ was the vector containing all parameters $\beta$ from previous $K$ models, and $A'$ and $A''^{(k)}$ were given by the equations:

$$[A', b'] = [Y, 0] \cdot P^T \tag{4.19}$$

$$[A''^{(k)}, b''^{(k)}] = [\hat{Y}^{(k)}, 0] \cdot P^T. \tag{4.20}$$

To find the optimal parameters of $\vec{\beta}$, Equation (4.21) [81] was used, as follows:

$$\begin{aligned} \min_{\vec{\beta}} \quad & \sum_{i=1}^{N} \ell(Y_i, \tilde{Y}_i) \\ \text{s.t.} \quad & \|\vec{\beta}\| \leq 1, \\ & \vec{\beta}^{(k)} \geq 0 \qquad k = 1, \ldots, K, \end{aligned} \tag{4.21}$$

where $\ell(\cdot)$ is the convex multiclass loss function [81] defined as the following:

$$\ell(Y_i, \tilde{Y}_i) = \max\{1 - \tilde{Y}_{g,i} + \max_{g^* \neq g}\{\tilde{Y}_{g^*,i}\}, 0\} \qquad \forall i = 1, \ldots, N, \tag{4.22}$$

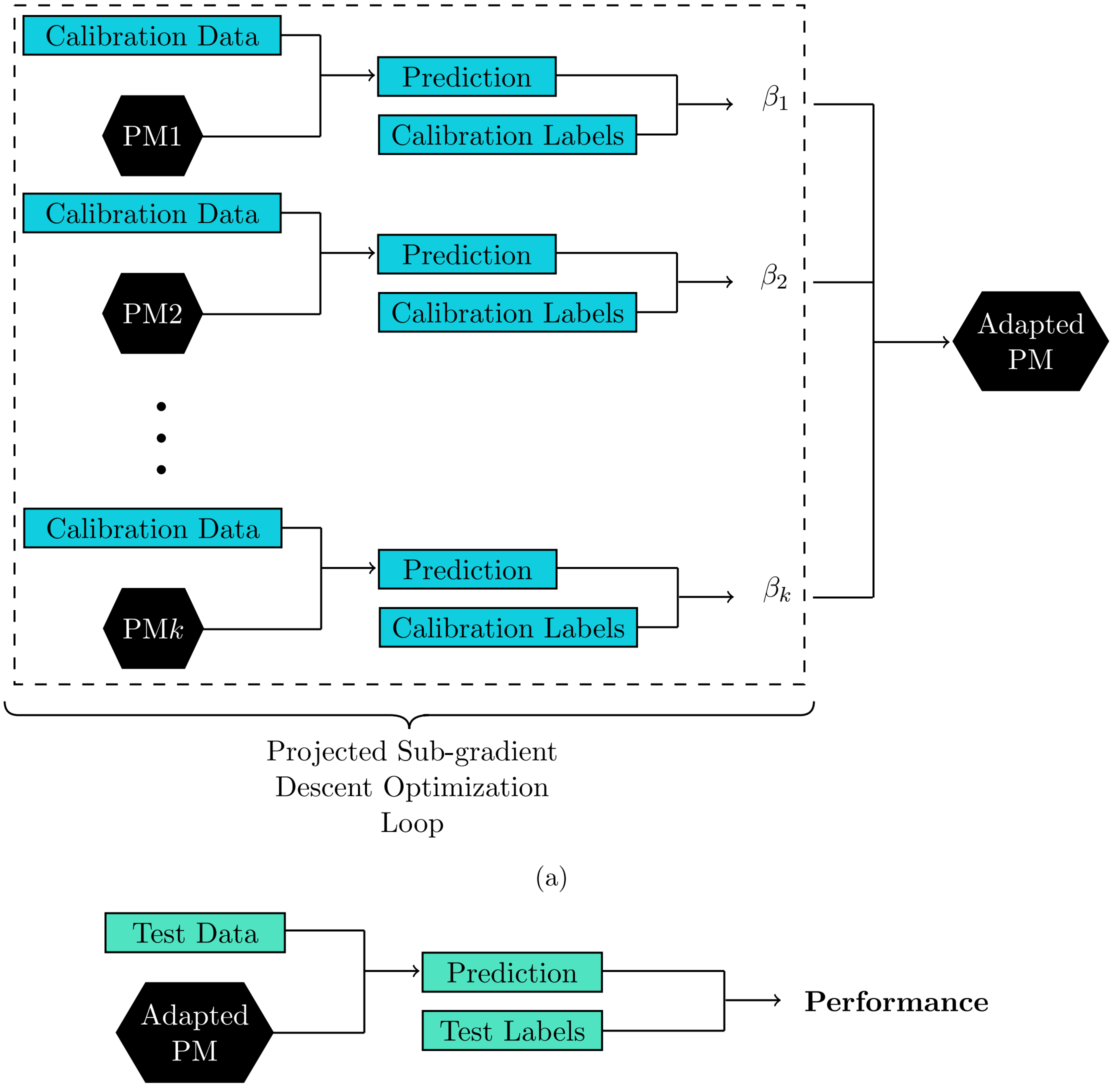
where $\tilde{Y}$ was computed using Equation (4.18), and $Y$ is the matrix that contains all of the label samples of the calibration set. Moreover, the subscript $g$ indicates the row where the matrix $Y$ is equal to 1. The loss of Equation (4.22) is equal to zero when the confidence value of the predicted class of sample $i$ is greater than at least 1 over the confidence value assigned to the rest of the classes for the same sample.

The optimization problem in Equation (4.21) was solved using a projected sub-gradient descent algorithm as in [81]. To do so, the matrices $A'$ and $A''^{(k)}$ were computed using Equations (4.19) and (4.20) respectively. Convergence was achieved when the norm of Equation (4.22) in the current update iteration was less than at least 0.05 over its norm in the previous update iteration.

Having found the optimal values of $\vec{\beta}$, the parameter $A$ in Equation (4.6) was found using matrices $A'$ and $A''$ in the following equation:

$$A = A' - \sum_{k=1}^{K} \vec{\beta}^{(k)} \cdot A''^{(k)}. \tag{4.23}$$

Finally, using the matrix $A$, and the bias term $b'$ from Equation (4.19), the prediction $Y_{test}$ on



(a)

(b)

Figure 4.5: Learning process of the Adaptive LS-SVM classification method. a) Optimization of the parameters $\beta$ during each iteration of the projected sub-gradient descent loop. b) Evaluation of the classification performance of the Adaptive LS-SVM classification method on the test data.

a new sample $i$ from the testing set was computed using a one-vs.-all approach as follows:

$$Y_{test} = argmax\{w_{test} + b'\}, \tag{4.24}$$

where $w_{test}$ was given by the following equation:

$$w_{test} = A \cdot [\mathcal{K}(X_{cal}, X_{test})]^T, \tag{4.25}$$

being $\mathcal{K}(\cdot)$ the RBF kernel function between the calibration set and the new sample from the testing set. Here, a value for parameter $\gamma$ of 0.5 was used. A summary of the training and evaluation process of the Adaptive LS-SVM classification method is given in Figure 4.5.

### 4.2.4 Adaptive LS-SVM Evaluation

Similar to the PAC classification method, the classification performance of the Adaptive LS-SVM was evaluated for each user in the cross-validation testing set during each of the cross-validation iterations described in Section 3.4.3. Furthermore, the same subset of 7 gestures from Section 4.1.5 was used to evaluate this method. The results of these analyses are discussed in Chapter 5.

## 4.3 Classification Method: Bilinear Model-based Classification

After evaluation the Adaptive LS-SVM, the third classification method explored was the bilinear model-based classification method. This method, which was described in Section 2.8.3, was proposed by Matsubara *et al.* [82] to represent EMG signals in the form of bilinear models. Then, the bilinear EMG model was classified using conventional classification algorithms. In this work, this concept was further expanded to be applied to a dataset composed of EMG and IMU signals. The implementation of this approach is presented in the following sections.

### 4.3.1 Bilinear Model-based Classification: Datasets

To properly build a user-independent classification model based on bilinear models, data acquired from both sensor modalities (EMG and IMU) had to be extracted from a fixed active region. This was done so that each motion class $G$ had the same number of samples $N$ to comply with the symmetric model in Equation (2.11). To do so, the indices of the onset and offset of the motion were manually selected for the EMG signal. Then, following a similar procedure as in Section 3.4.1 the onset and offset of the motion for the IMU's accelerometer and gyroscope were determined.

Finally, features were calculated from these active regions using the same procedure described in Chapter 3.

### 4.3.2 Bilinear Model Learning

Using the datasets obtained in the previous section, a bilinear model of each EMG signal of each subject in the cross-validation training sets (Section 3.4.3) was created using the procedure described bellow.

First, the EMG data from each subject in the cross-validation training set were divided into style[6] ($S$) and content ($C$) variables [82], which represented the user-dependent factors and the motion dependent factors respectively. To do so, the symmetric $K$-dimensional EMG signal model in Equation (2.12) was stacked into a single $FK \times C$ matrix, so that standard matrix operations could be applied, as in Equation (4.26):

$$\bar{Y}^S = \begin{bmatrix} F_1^1 \\ \vdots \\ F_1^K \\ F_2^1 \\ \vdots \\ F_x^K \end{bmatrix}, \tag{4.26}$$

where $\bar{Y}^S$ represents the symmetric model of the EMG signal of a subject (style variable), and $F_x^K$ indicates the feature $x$ computed for channel $K$. Further, $F_x^K$ was given by the following equation:

$$F_x^K = \begin{bmatrix} C_1^1 & \dots & C_N^1 & C_1^2 & \dots & C_N^G \end{bmatrix}, \tag{4.27}$$

where $C \in \mathbb{R}^{1 \times NG}$, with $N$ being the number of samples in motion $G$. By repeating this procedure for each subject in the cross-validation training set, a matrix $\bar{Y}$ was constructed by stacking each matrix $\bar{Y}^S$ on top of each other, as follows:

[6] In this work, the decision of using the word style instead of the word subject, was to follow the naming convention of the bilinear models presented in [83].

$$\bar{Y} = \begin{bmatrix} \bar{Y}^1 \\ \bar{Y}^2 \\ \vdots \\ \bar{Y}^S \end{bmatrix}. \tag{4.28}$$

Here, $\bar{Y} \in \mathbb{R}^{FS \times C}$. Moreover, the weight matrix $W$ in Equation (2.12) was defined as the stacked $FI \times J$ matrix (Equation (4.29)) consisting on the $K$-dimensional weights $\mathrm{w}_{ij}$ from Equation (2.11), with each dimension $K$ containing $F$ number of features.

$$W = \begin{bmatrix} \mathrm{w}_{1,1} & \cdots & \mathrm{w}_{1,J} \\ \vdots & \ddots & \vdots \\ \mathrm{w}_{I,1} & \cdots & \mathrm{w}_{I,J} \end{bmatrix}. \tag{4.29}$$

With these definitions, Equation (2.12) was rewritten into two equivalent matrix forms [83]:

$$\bar{Y} = \left[W^{VT} \cdot Z\right]^{VT} \cdot X, \tag{4.30}$$

$$\bar{Y}^{VT} = [W \cdot X]^{VT} \cdot Z, \tag{4.31}$$

where $Z \in \mathbb{R}^{I \times S}$ and $X \in \mathbb{R}^{J \times C}$ are the matrices containing the style, and content parameter vectors respectively. Furthermore, $\{\cdot\}^{VT}$ indicates the vector transpose [83], defined as:

$$\begin{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} & \begin{bmatrix} g \\ h \end{bmatrix} \\ \begin{bmatrix} c \\ d \end{bmatrix} & \begin{bmatrix} i \\ j \end{bmatrix} \\ \begin{bmatrix} e \\ f \end{bmatrix} & \begin{bmatrix} k \\ l \end{bmatrix} \end{bmatrix}^{VT} = \begin{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} & \begin{bmatrix} c \\ d \end{bmatrix} & \begin{bmatrix} e \\ f \end{bmatrix} \\ \begin{bmatrix} g \\ h \end{bmatrix} & \begin{bmatrix} i \\ j \end{bmatrix} & \begin{bmatrix} k \\ l \end{bmatrix} \end{bmatrix}. \tag{4.32}$$

Finally, using Equations (4.30) and (4.31), $Z$ and $X$ were iteratively derived using the steps described in [83], using $I = 2$ and $J = 3$ as recommended in [82]. Having found the optimal values for the content matrix $X$, a pairwise dataset $\{X^{(c)}, l^{(c)}\}$ was created, where $l^{(c)}$ indicates the motion label $g = 1 \ldots G$ of the corresponding column $X^{(c)}$. This pairwise dataset was then used to train a NN model, which was created by using the TensorFlow [99] library for Python [100].

The NN architecture consisted of the input layer, two hidden layers, and the output layer. Moreover, the two hidden layers consisted of 50, and 20 nodes respectively. After each hidden layer, a dropout regularization layer, with a dropout rate of 20%, was included to prevent overfitting [101]. Furthermore, a batch normalization layer was added after the first dropout layer to reduce the covariance shift, *i.e.*, change in distribution of the layer's input data during training [102]. To compute the outputs of each hidden layer, a rectified linear unit (ReLU) activation function was used. Similarly, a "softmax" activation function was used to compute the output of the output layer.

Before training the NN, each row of the content matrix $X$ was standardized to have a mean of 0 and a standard deviation equal to 1. Then, the NN model was finally trained over 300 iterations with an Adam optimizer. This optimizer was configured to have a learning rate of 0.001, and a decay value of 1e−6 to speed the learning process. These parameters were determined experimentally. A summary of the NN structure is shown in Figure 4.6.

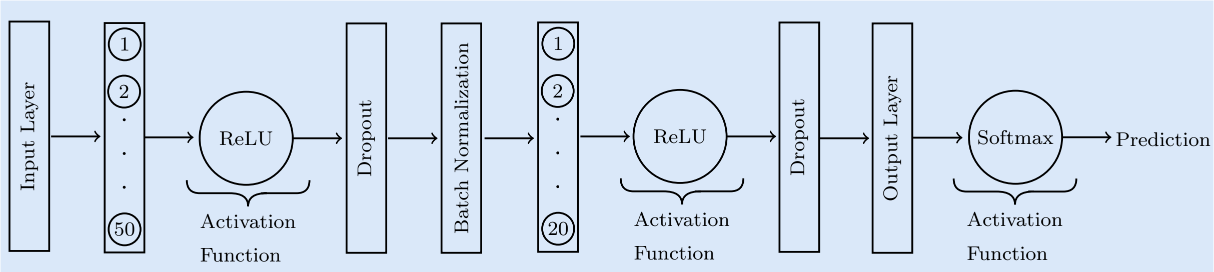
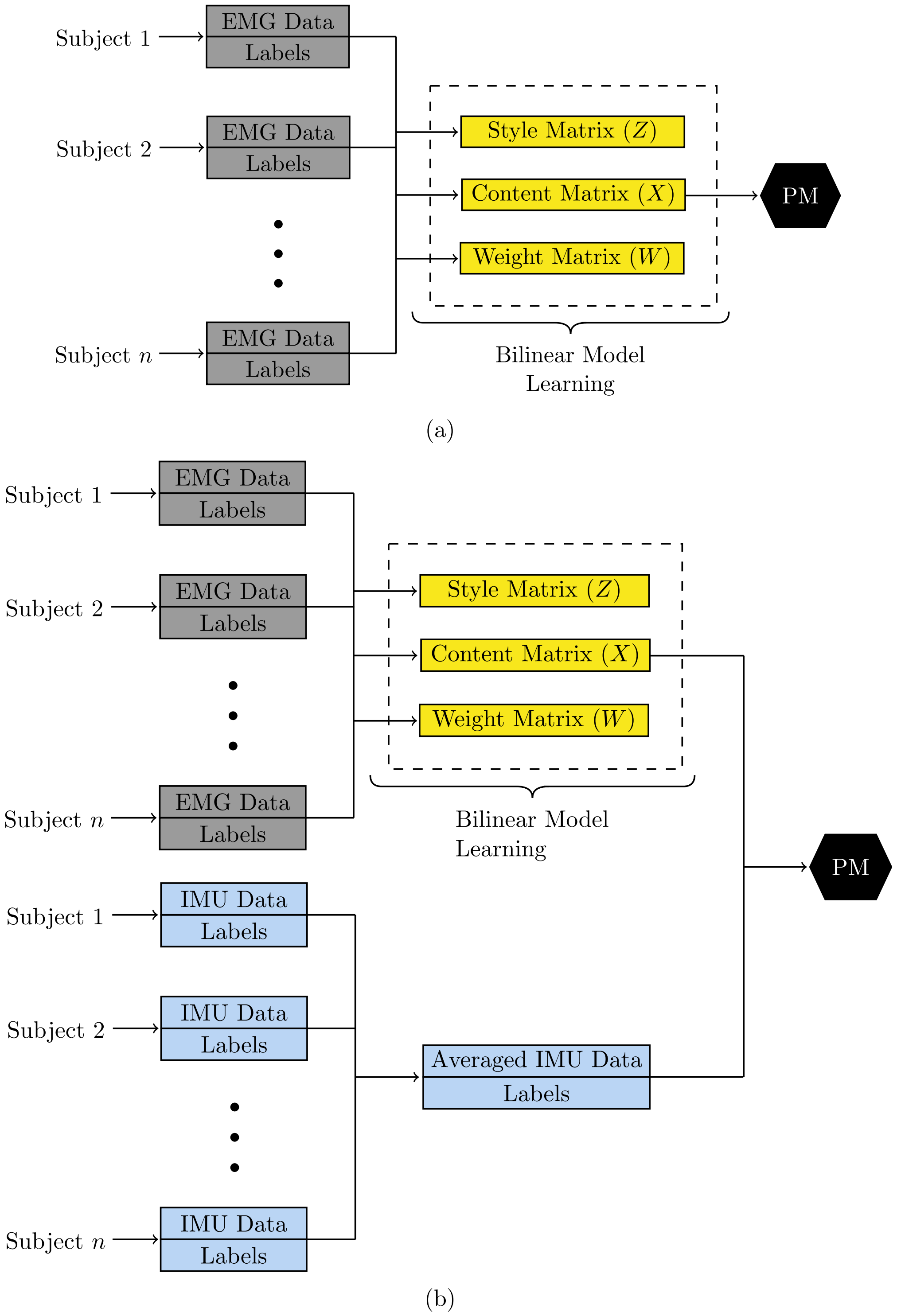


Figure 4.6: Architecture of the NN used to train the bilinear model-based classifier. The hidden layer is represented by the bar with numbered circles inside. Each numbered circle represents a node of the hidden layer.



(a)

(b)

Figure 4.7: Bilinear model learning process. The PM represents a NN model trained using the content matrix X. a) Learning a bilinear EMG model. b) Fusing the content matrix X with the averaged IMU features.

In order to explore the effects of the IMU for classifying gestures using the bilinear models, the content matrix $X$, was fused with the IMU features obtained in Section 3.4.2. However, because these features belonged to multiple users, the average of these features across all subjects in the cross-validation training set was employed (*e.g.*, the WL feature computed for the acceleration data in the $x$ direction was averaged across all subjects). Then, a different NN model was trained using this information and the same parameters used for the EMG NN model. A summary of the process described in this section, and in Section 4.3.1 is shown in Figure 4.7.

### 4.3.3 Calibration Phase

Similarly to the previously evaluated classification models, the bilinear model-based classification method required the use of a calibration phase. However, the main difference was that this calibration phase required a calibration set formed only by one repetition of only one motion. The reason behind this was because only one motion is required to estimate the style matrix $Z$ of a new user. Therefore, for each user in the cross-validation testing set, one random repetition of the wrist flexion gesture in a random arm position was used to calibrate the bilinear model. Then, using the content variables, which were computed in Section 4.3.2, for the wrist flexion gesture, the subject user-dependent factors $Z_n$ were computed using Equation (4.33):

$$Z_n = \left[ \left[ W \cdot X_g \right]^{VT} \right]^{+} \cdot \bar{Y}_g^{VT}, \tag{4.33}$$

where $\{\cdot\}^{+}$ indicates the Moore–Penrose pseudoinverse matrix, and $X_g$ and $\bar{Y}_g$ represent, respectively, the content variables and the new input data of motion $g$ used to calibrate the model.

Finally, using the new style content variable $Z_n$, the content variables $X_n$ for new observations $\bar{Y}_n$ of motion $n$ were obtained using Equation (4.34):

$$X_n = \left[ \left[ W^{VT} \cdot Z_n \right]^{VT} \right]^{+} \cdot \bar{Y}_n. \tag{4.34}$$

### 4.3.4 Bilinear Model Evaluation

As with the previous classification methods, the bilinear model-based classification method using NN was applied to the 10 gesture dataset from Section 3.3.2 to observe its performance. To do so, programming code was implemented in Python 3.6 (Appendix D). Moreover, this same classification method was equally applied to the 7 gesture dataset created in Section 4.1.5. The steps described in this section were used to condition the data from the EMG, and the EMG and IMU datasets from the 7 gestures. The results of this analysis are described in Chapter 5.

## 4.4 Classification Method: MLP Neural Networks

The final classification method implemented was based on MLP networks, which were described in Section 2.7.4.3. Here, the EMG, and EMG and IMU datasets obtained from the Myo Armband in Chapter 3 were used to train the MLP network. Further, to observe the effects in the classification outcomes, the results from the classifier using EMG data only were compared against the results obtained from the classifier using a combination of both EMG and IMU data.

### 4.4.1 MLP Dataset

The MLP networks datasets consisted of both the EMG, and EMG and IMU datasets obtained in Chapter 3. For each cross-validation iteration in Section 3.4.3, each subject's EMG data in the training set were combined into a single dataset in order to be trained using the MLP networks, as shown in Figure 4.8. This procedure was repeated for the combined EMG and IMU dataset for further comparison.

### 4.4.2 MLP Learning

For this application, a stochastic gradient descent (SGD) learning algorithm was employed during the backpropagation step. Using this learning method, which is also known as online learning [103], the weights of the network were updated for each training sample after each epoch, *i.e.*, for each individual training sample, its weight was updated after a full forward–backward propagation

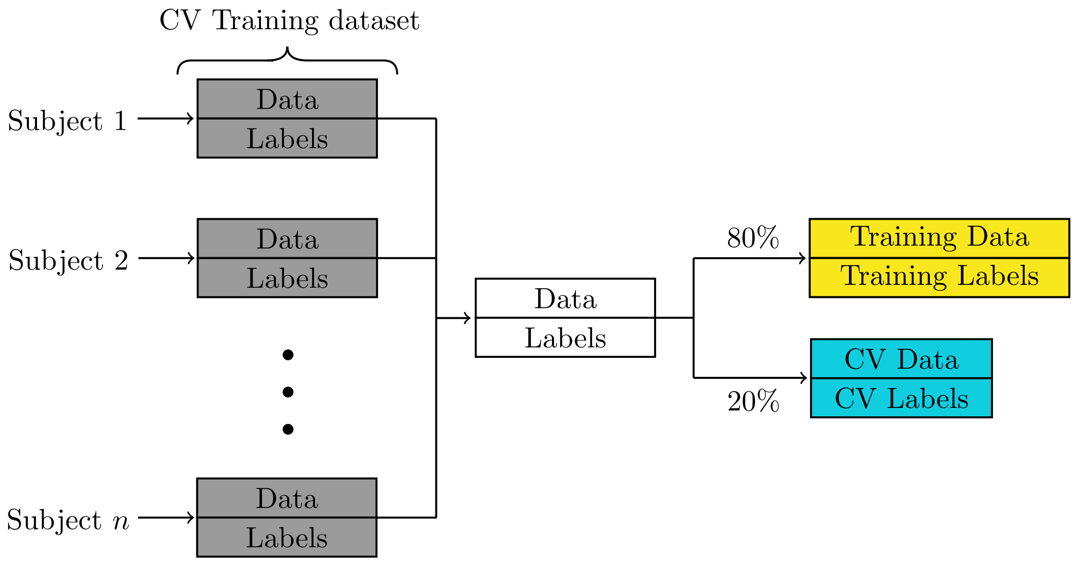


Figure 4.8: Procedure used to create the MLP dataset. Data from each subject in the cross-validation training set were combined, and then split in two data subsets: the training data, and the cross-validation data.

phase. Further, the MLP network was trained in RStudio [104] with the RSNNS package [105] software (Appendix E) following the procedure described below.

First, the data, and the data labels were standardized to have a mean of 0 and a standard deviation equal to 1. Moreover, the mean and standard deviation parameters were saved to reconstruct the original labels after training the network. Then, the original data were split into two sub-datasets, the training and cross-validation set, each one formed of 80% and 20% of the original data, respectively Figure 4.8. To determine the architecture of the MLP network, as well as the learning parameters, the cross-validation set was used to observe the efficiency of the model for classifying 10 gestures. The choice of implementing a cross-validation set was to evaluate the neural network model on its ability to generalize to an unseen dataset. By doing so, it was possible to keep a low bias, which is the model's ability to obtain an output closer to the ground truth label with small errors, and a reasonable high variance. Moreover, the cross-validation set allows for an optimal bias-variance tradeoff, thus preventing an overfitted model [106]. Figure 4.9 shows a summary of this procedure.

Having found the PM that performed best in the cross-validation set, the MLP network was created using an architecture that consisted of the input layer, three hidden layers, and the output

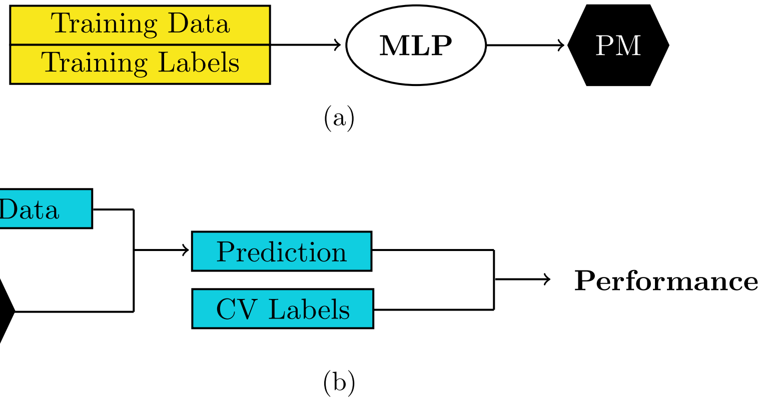


Figure 4.9: MLP network learning. a) Training a PM using MLP networks. b) Testing the PM on the cross-validation data. The steps in a), and b) were repeated until the PM achieved a high classification performance on the cross-validation dataset.

layer. The three hidden layers contained 300, 200, and 100 nodes, respectively. Training of the network took an average of 4 hours. This architecture is shown in Figure 4.10.

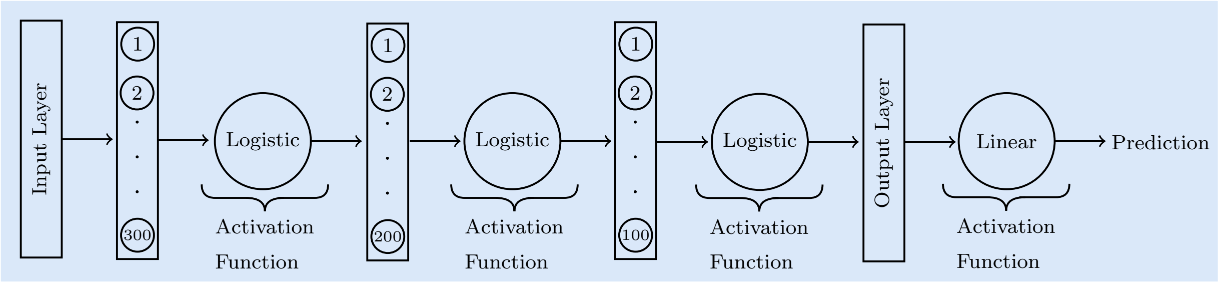


Figure 4.10: Architecture of the MLP network. The hidden layer is represented by the bar with numbered circles inside. Each numbered circle represent a node of the hidden layer.

Table 4.1 show the parameters used as the input for the RSNNS MLP function. Here the "initFunction", is the initialization function that randomized the networks weights in a range of values between $-3$ and 3. This was done to break symmetry of the network, *i.e.*, each neuron in the network was updated with a different value, thus allowing for a better accuracy of the training model. Furthermore, the "learnFunc" parameter, which stands for learning function, allowed the network to be trained using the SGD learning algorithm as explained before. Here, the learning rate of the gradient algorithm was set to 0.2. Finally, the update function "Topological_Order" computed the outputs of the units during the forward propagation phase in a topological order, *i.e.*, starting with the input layer, the network computed the output of each unit before moving on to the next layer.

Table 4.1: RSNNS MLP network parameters.

| **Name** | **Value** | **Parameters** |
|---|---|---|
| initFunc | “Randomize_Weights” | -0.3, 0.3 |
| learnFunc | “BackpropMomentum” | 0.2, 0 |
| updateFunc | “Topological_Order” | 0 |

For this application, a logistic activation function, was employed for the input and the three hidden layers, whereas for the output layer, the identity or linear function was utilized, *i.e.*, the output value of the output layer was equal to the output of the previous layer (Figure 4.10). The choice of a linear function in the output layer was due to the label values not being integer numbers. Furthermore, the use of this linear activation function allowed for the output layer to have 1 node only. Finally, the output of the network was unscaled using the previously obtained mean and standard deviation parameters from the training set. However, because the values of the unscaled labels were not integer numbers, a function was employed to round them to the nearest integer so that they lay within the range $1 \leq$ output label $\leq 10$.

### 4.4.3 MLP Network Evaluation

Once the MLP network finished training using the best network parameters, it was tested in the 10 gesture dataset from Section 3.3.2 to observe its performance. Moreover, this same classification method was equally applied to the 7 gesture dataset created in Section 4.1.5. To do so, the steps described in this section were used to condition the data from the EMG, and the EMG and IMU datasets from the 7 gestures. The results of this analysis are described in Chapter 5.

# Chapter 5

# Results and Discussion

This chapter describes the results of the four classification methods described in Sections 4.1 to 4.4 when applied in a user-independent scenario. First, each classification method was utilized to classify data from the 10 gestures described in Section 3.3.2. These data were obtained using two different sensor modalities: EMG, and EMG and IMU. Furthermore, this methodology was applied to an optimized gesture set that was created by removing the sensor data from gestures for which their motion was controlled by the same group of muscles, *i.e.*, gestures showing the same motion patterns because their EMG signals were generated from the same muscles, as indicated in Section 4.1.5.

Since one of the preliminary goals indicated in Section 1.3 was to observe the efficacy of the Myo Armband when classifying gestures using the information of the IMU, a statistical analysis was performed on the classified data to compare the performance of the two sensor modalities in each of the classification methods. Finally, the results from the best sensor modality from each classification method were also compared against each other to determine which of the classification method works best for the Myo Armband during a user-independent gesture classification scenario.

## 5.1 PAC Classification

This section describes the classification results of the datasets of 10 and 7 gestures using the PAC classification algorithm described in Section 4.1. For each gesture dataset, features extracted from the EMG were classified first using this method. Then, the IMU features were combined with the EMG feature sets for further classification.

### 5.1.1 PAC: 10 Gesture Classification

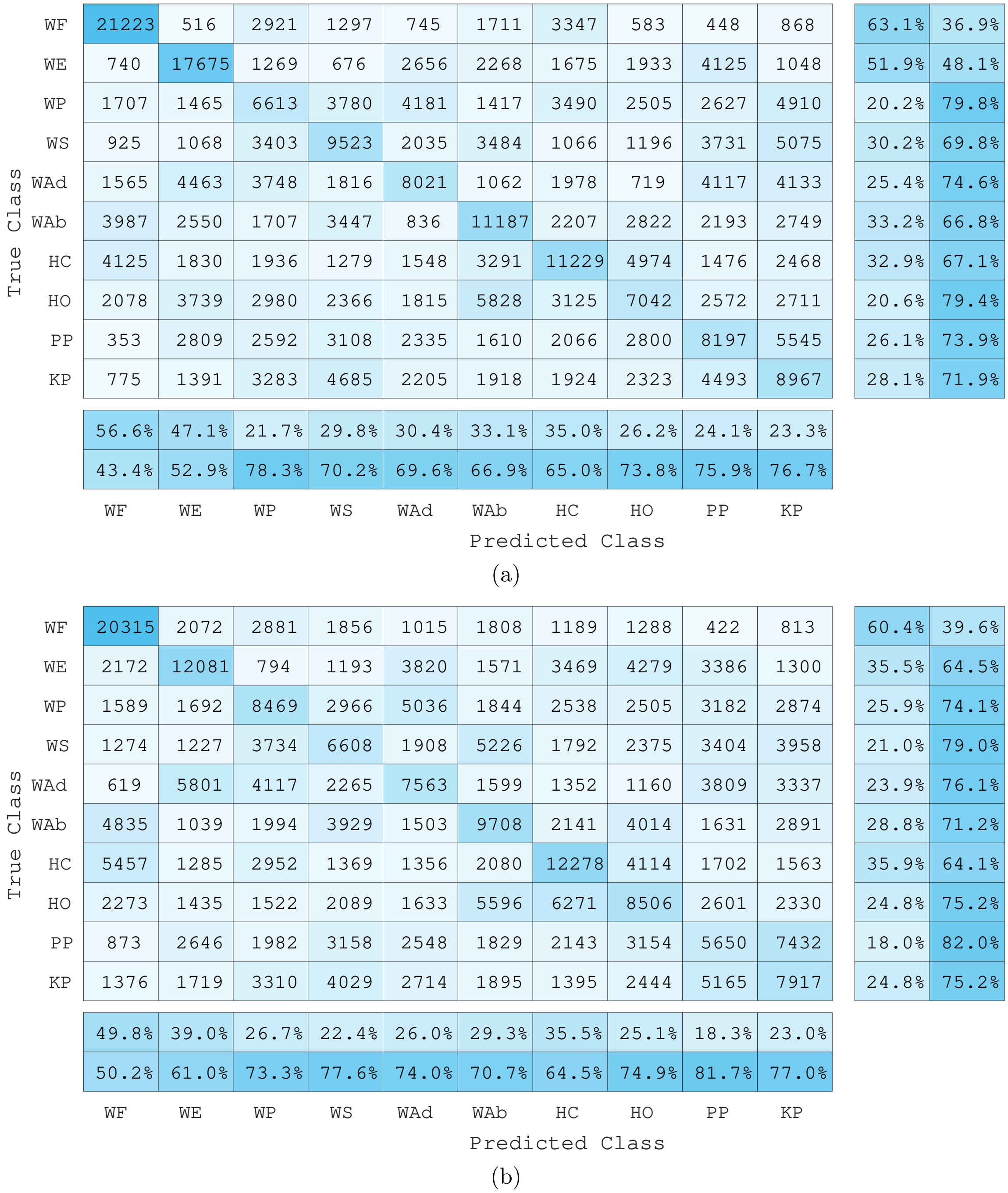
As mentioned in Chapter 4, for each cross-validation iteration described in Section 3.4.3, a LS-SVM model was created for each subject in the cross-validation training set. Then, each subject in the cross-validation testing set was tested using the PAC classification method. This procedure was repeated for both the EMG, and the EMG and IMU datasets.

Classification results from each cross-validation iteration defined in Section 3.4.3 using the PAC classification method are presented in Table 5.1. Further, Figure 5.1 shows the confusion matrices for the twenty-two testing subjects when classifying the 10 gestures in a user-independent scenario using EMG data only (Figure 5.1a), and a combination of EMG and IMU data (Figure 5.1b). Each column of each confusion matrix represents the instances in a predicted class, while each row represents the instances in an actual class. For EMG data only, classification accuracies using the PAC classification algorithm ranged from 18.37–50.41% with a mean classification accuracy of 33.11%(±8.88%), whereas for the combined EMG and IMU sensor modality, accuracies ranged from 12.49–44.29% with a mean accuracy of 29.98%(±8.81).

For each gesture, the precision and recall scores were also calculated. The former evaluates the performance of the model on the positive class or, in other words, it highlights the ability of the classification model to return only relevant data, *i.e.*, it shows which classified gestures actually belong to a specific class. On the other hand, the recall value represents the rate of the true positives (the correctly classified samples) of a specific class when compared against the false negatives, which are the samples that were incorrectly classified as a different class. Therefore, the precision and recall scores for each classified gesture using EMG data only, and EMG and IMU data combined, were calculated from the confusion matrix in Figure 5.1a, and in Figure 5.1b, respectively.

Table 5.1: Classification accuracies of 10 gestures for each cross-validation iteration set using the PAC classification method. The best classification result for each subject within each cross-validation iteration testing set is shown in bold.

| Cross-Validation Iteration | | Classification Accuracy (%) | |
|---|---|---|---|
| Training Set | Testing Set | EMG | EMG + IMU |
| CVF2, CVF3, CVF4, CVF5 | S10 | 36.40 | **39.50** |
| | S16 | 18.41 | **21.76** |
| | S17 | **50.41** | 39.32 |
| | S18 | **44.15** | 24.49 |
| CVF1, CVF3, CVF4, CVF5 | S13 | 31.03 | **33.25** |
| | S14 | **42.23** | 27.90 |
| | S19 | **35.86** | 29.66 |
| | S22 | **31.59** | 30.31 |
| | S23 | **45.48** | 41.51 |
| CVF1, CVF2, CVF4, CVF5 | S4 | 43.07 | **44.29** |
| | S6 | 32.87 | **37.85** |
| | S12 | **28.20** | 21.70 |
| | S20 | **36.97** | 29.11 |
| | S24 | **18.37** | 12.49 |
| CVF1, CVF2, CVF3, CVF5 | S3 | **36.53** | 29.86 |
| | S5 | 34.30 | **39.35** |
| | S7 | 23.67 | **30.21** |
| | S25 | **33.26** | 21.93 |
| CVF1, CVF2, CVF3, CVF4 | S2 | 35.02 | **41.44** |
| | S9 | **26.81** | 22.35 |
| | S11 | **23.83** | 15.65 |
| | S15 | 20.02 | **25.76** |


True Class
Predicted Class

| | WF | WE | WP | WS | WAd | WAb | HC | HO | PP | KP | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WF | 21223 | 516 | 2921 | 1297 | 745 | 1711 | 3347 | 583 | 448 | 868 | 63.1% | 36.9% |
| WE | 740 | 17675 | 1269 | 676 | 2656 | 2268 | 1675 | 1933 | 4125 | 1048 | 51.9% | 48.1% |
| WP | 1707 | 1465 | 6613 | 3780 | 4181 | 1417 | 3490 | 2505 | 2627 | 4910 | 20.2% | 79.8% |
| WS | 925 | 1068 | 3403 | 9523 | 2035 | 3484 | 1066 | 1196 | 3731 | 5075 | 30.2% | 69.8% |
| WAd | 1565 | 4463 | 3748 | 1816 | 8021 | 1062 | 1978 | 719 | 4117 | 4133 | 25.4% | 74.6% |
| WAb | 3987 | 2550 | 1707 | 3447 | 836 | 11187 | 2207 | 2822 | 2193 | 2749 | 33.2% | 66.8% |
| HC | 4125 | 1830 | 1936 | 1279 | 1548 | 3291 | 11229 | 4974 | 1476 | 2468 | 32.9% | 67.1% |
| HO | 2078 | 3739 | 2980 | 2366 | 1815 | 5828 | 3125 | 7042 | 2572 | 2711 | 20.6% | 79.4% |
| PP | 353 | 2809 | 2592 | 3108 | 2335 | 1610 | 2066 | 2800 | 8197 | 5545 | 26.1% | 73.9% |
| KP | 775 | 1391 | 3283 | 4685 | 2205 | 1918 | 1924 | 2323 | 4493 | 8967 | 28.1% | 71.9% |
| | 56.6% | 47.1% | 21.7% | 29.8% | 30.4% | 33.1% | 35.0% | 26.2% | 24.1% | 23.3% | | |
| | 43.4% | 52.9% | 78.3% | 70.2% | 69.6% | 66.9% | 65.0% | 73.8% | 75.9% | 76.7% | | |

(a)

| | WF | WE | WP | WS | WAd | WAb | HC | HO | PP | KP | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WF | 20315 | 2072 | 2881 | 1856 | 1015 | 1808 | 1189 | 1288 | 422 | 813 | 60.4% | 39.6% |
| WE | 2172 | 12081 | 794 | 1193 | 3820 | 1571 | 3469 | 4279 | 3386 | 1300 | 35.5% | 64.5% |
| WP | 1589 | 1692 | 8469 | 2966 | 5036 | 1844 | 2538 | 2505 | 3182 | 2874 | 25.9% | 74.1% |
| WS | 1274 | 1227 | 3734 | 6608 | 1908 | 5226 | 1792 | 2375 | 3404 | 3958 | 21.0% | 79.0% |
| WAd | 619 | 5801 | 4117 | 2265 | 7563 | 1599 | 1352 | 1160 | 3809 | 3337 | 23.9% | 76.1% |
| WAb | 4835 | 1039 | 1994 | 3929 | 1503 | 9708 | 2141 | 4014 | 1631 | 2891 | 28.8% | 71.2% |
| HC | 5457 | 1285 | 2952 | 1369 | 1356 | 2080 | 12278 | 4114 | 1702 | 1563 | 35.9% | 64.1% |
| HO | 2273 | 1435 | 1522 | 2089 | 1633 | 5596 | 6271 | 8506 | 2601 | 2330 | 24.8% | 75.2% |
| PP | 873 | 2646 | 1982 | 3158 | 2548 | 1829 | 2143 | 3154 | 5650 | 7432 | 18.0% | 82.0% |
| KP | 1376 | 1719 | 3310 | 4029 | 2714 | 1895 | 1395 | 2444 | 5165 | 7917 | 24.8% | 75.2% |
| | 49.8% | 39.0% | 26.7% | 22.4% | 26.0% | 29.3% | 35.5% | 25.1% | 18.3% | 23.0% | | |
| | 50.2% | 61.0% | 73.3% | 77.6% | 74.0% | 70.7% | 64.5% | 74.9% | 81.7% | 77.0% | | |

(b)

Figure 5.1: PAC confusion matrix of the 10 gestures collected using EMG data only (a), and EMG and IMU data (b) from all testing subjects. The last two rows of each confusion matrix represent the precision score percentages and the false positive rate percentages (the cumulative number of false positives) of each class, in the top and bottom row, respectively. Similarly, the last two columns represent the recall score percentages and the false negative rate percentages (the cumulative number of false negatives) of each class, in the left and right column, respectively.

For all of the gestures, the lowest precision score among the twenty-two subjects when using EMG data only, was for the Wrist Pronation gesture with an overall precision of 21.7%, and the highest precision was for the Wrist Flexion gesture with an overall precision of 56.6%. On the other hand, the lowest recall scores were for the Wrist Pronation and the Hand Open gestures with overall scores of 20.2% and 20.6%, respectively. The highest recall score was for the Wrist Flexion gesture with an overall score of 63.1%. Furthermore, the lowest precision score among the twenty-two subjects when using both modalities was for the Precision Pinch gesture with an overall precision of 18.3%, and the highest precision score was for the Wrist Flexion gesture with an overall precision of 49.8%. At the same time, the lowest recall score was for the Precision Pinch gesture with a value of 18%, whereas the highest score was for the Wrist Flexion gesture with a score of 60.4%.

Even though the overall classification accuracies from both sensor modalities were above chance (10%), these accuracies were lower than the lowest accuracy reported in the literature for user-independent pattern recognition applications (73%) [82]. Therefore, the classification algorithm was further optimized by removing some gestures from the gesture dataset. The next section describes the classification performance using a 7 gesture dataset in an attempt to improve the outcomes of the PAC classification method.

### 5.1.2 PAC: 7 Gesture Classification

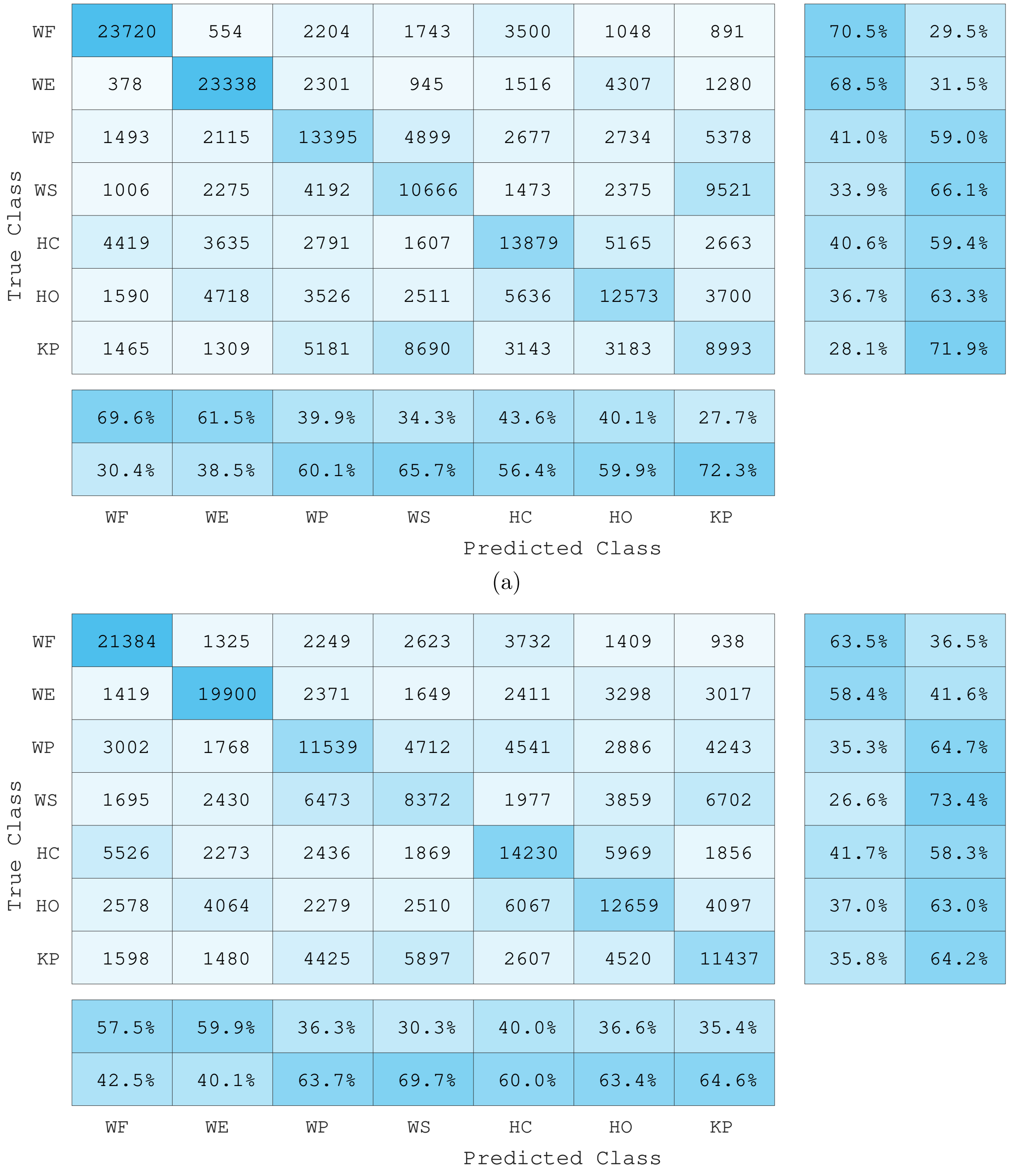
After reducing the number of gestures in the gesture dataset by removing the EMG and the IMU information from the removed gestures, a new LS-SVM classifier was created. The PAC classification method was then applied to the cross-validation sets following the same procedure described in the previous section.

The new classification results from each cross-validation iteration using the PAC classification method in the 7 gesture dataset are presented in Table 5.2. Figure 5.2 shows the confusion matrices for the twenty-two testing subjects when classifying the new 7 gestures in a user-independent scenario using EMG data only (Figure 5.2a), and a combination of EMG and IMU data (Figure 5.2b). For EMG data only, classification accuracies using the PAC classification algorithm ranged from 31.72–60.38% with a mean classification accuracy of 45.59%(±7.05%), whereas for the combined

EMG and IMU sensor modality, accuracies ranged from 23.88–57.50% with a mean accuracy of 42.58%(±10.35).

Table 5.2: Classification accuracies of 7 gestures for each cross-validation iteration set using PAC classification method. The best classification result for each subject within each cross-validation iteration testing set is shown in bold.

| Cross-Validation Iteration | | Classification Accuracy (%) | |
|---|---|---|---|
| Training Set | Testing Set | EMG | EMG + IMU |
| CVF2, CVF3, CVF4, CVF5 | S10 | 48.34 | **55.97** |
| | S16 | **31.72** | 30.92 |
| | S17 | 48.43 | **54.15** |
| | S18 | **51.49** | 34.50 |
| CVF1, CVF3, CVF4, CVF5 | S13 | **53.22** | 52.68 |
| | S14 | **45.67** | 33.67 |
| | S19 | 35.88 | **57.50** |
| | S22 | **49.19** | 39.62 |
| | S23 | **60.38** | 48.06 |
| CVF1, CVF2, CVF4, CVF5 | S4 | 51.56 | **55.53** |
| | S6 | **44.17** | 36.39 |
| | S12 | **54.58** | 46.39 |
| | S20 | **46.90** | 26.24 |
| | S24 | **37.66** | 31.70 |
| CVF1, CVF2, CVF3, CVF5 | S3 | 35.21 | **40.47** |
| | S5 | 48.29 | **54.52** |
| | S7 | **40.12** | 23.88 |
| | S25 | **48.08** | 37.31 |
| CVF1, CVF2, CVF3, CVF4 | S2 | **45.09** | 42.74 |
| | S9 | **47.14** | 40.40 |
| | S11 | 43.02 | **54.73** |
| | S15 | 36.94 | **39.30** |



| True Class | WF | WE | WP | WS | HC | HO | KP | | |
|---|---|---|---|---|---|---|---|---|---|
| WF | 23720 | 554 | 2204 | 1743 | 3500 | 1048 | 891 | 70.5% | 29.5% |
| WE | 378 | 23338 | 2301 | 945 | 1516 | 4307 | 1280 | 68.5% | 31.5% |
| WP | 1493 | 2115 | 13395 | 4899 | 2677 | 2734 | 5378 | 41.0% | 59.0% |
| WS | 1006 | 2275 | 4192 | 10666 | 1473 | 2375 | 9521 | 33.9% | 66.1% |
| HC | 4419 | 3635 | 2791 | 1607 | 13879 | 5165 | 2663 | 40.6% | 59.4% |
| HO | 1590 | 4718 | 3526 | 2511 | 5636 | 12573 | 3700 | 36.7% | 63.3% |
| KP | 1465 | 1309 | 5181 | 8690 | 3143 | 3183 | 8993 | 28.1% | 71.9% |
| | 69.6% | 61.5% | 39.9% | 34.3% | 43.6% | 40.1% | 27.7% | | |
| | 30.4% | 38.5% | 60.1% | 65.7% | 56.4% | 59.9% | 72.3% | | |

Predicted Class

(a)

| True Class | WF | WE | WP | WS | HC | HO | KP | | |
|---|---|---|---|---|---|---|---|---|---|
| WF | 21384 | 1325 | 2249 | 2623 | 3732 | 1409 | 938 | 63.5% | 36.5% |
| WE | 1419 | 19900 | 2371 | 1649 | 2411 | 3298 | 3017 | 58.4% | 41.6% |
| WP | 3002 | 1768 | 11539 | 4712 | 4541 | 2886 | 4243 | 35.3% | 64.7% |
| WS | 1695 | 2430 | 6473 | 8372 | 1977 | 3859 | 6702 | 26.6% | 73.4% |
| HC | 5526 | 2273 | 2436 | 1869 | 14230 | 5969 | 1856 | 41.7% | 58.3% |
| HO | 2578 | 4064 | 2279 | 2510 | 6067 | 12659 | 4097 | 37.0% | 63.0% |
| KP | 1598 | 1480 | 4425 | 5897 | 2607 | 4520 | 11437 | 35.8% | 64.2% |
| | 57.5% | 59.9% | 36.3% | 30.3% | 40.0% | 36.6% | 35.4% | | |
| | 42.5% | 40.1% | 63.7% | 69.7% | 60.0% | 63.4% | 64.6% | | |

Predicted Class

(b)

Figure 5.2: PAC confusion matrix of the 7 gestures collected using EMG data only (a), and EMG and IMU data (b) from all testing subjects. The last two rows from each confusion matrix represent the precision score percentages and the false positive rate percentages (the cumulative number of false positives) of each class, in the top and bottom row, respectively. Similarly, the last two columns represent the recall score percentages and the false negatives rate percentages (the cumulative number of false negatives) of each class, in the left and right column, respectively.

Similar to the 10 gesture dataset, for each gesture, the precision and recall scores were also calculated. For each classified gesture the precision and recall scores using EMG data only, and EMG and IMU data combined, were calculated from the confusion matrix in Figure 5.2a, and in Figure 5.2b, respectively.

For all of the gestures, the lowest precision score among the twenty-two subjects when using EMG data only was for the Key Pinch gesture with an overall precision of 27.7%, and the highest precision was for the Wrist Flexion gesture with an overall precision of 69.6%. Furthermore, the lowest recall scores were for the Key Pinch with an overall score of 28.1%. The highest recall score was for the Wrist Flexion gesture with an overall score of 70.5%. On the other hand, the lowest precision score among the twenty-two subjects when using both modalities was for the Wrist Supination gesture with an overall precision of 30.3%, and the highest precision score was for the Wrist Extension gesture with an overall precision of 59.9%. Similarly, the lowest recall score was for the Wrist Supination gesture with an overall score of 26.6%, whereas the highest score was for the Wrist Flexion gesture with a score of 63.5%.

### 5.1.3 PAC: Discussion

The results obtained for the PAC classification method show no clear improvement in the classification accuracy using a combination of the EMG and IMU sensor data for any of the 10 and 7 gesture datasets. To further validate this assumption, a statistical analysis was done on both gesture datasets. A paired sample $t$ test comparison of means was performed between the accuracy results from the EMG, and the EMG and IMU sensor modalities. For the 10 gesture dataset, the results show that the mean recognition accuracy of the combined EMG and IMU sensor modality (29.98%) was lower than the mean recognition accuracy of the EMG sensor data (33.11%), however, this difference was not statistically significant ($p = 0.06$). On the other hand, the results for the 7 gesture dataset show that the mean recognition accuracy of the combined EMG and IMU sensor modality (42.58%) was lower than the mean recognition accuracy using the EMG sensor data alone (45.59%). Again, these results were not statistically significant ($p = 0.189$). It can also be seen from the mean accuracies that the PAC classification algorithm had a similar performance when the IMU data were added to the EMG data on both gesture datasets. Also, from the precision and recall scores, it can be observed that there was no consistency on the classification performance as most of the misclassified samples varied significantly between sensor modalities on the gesture datasets.

The poor performance of the PAC classification method can be explained by looking at how it works and how it was implemented. Given that the algorithm relies on updating the support vectors, *i.e.*, samples located closely to the separating hyperplane, it is a requirement for most of the data to have the same probability distribution. It can be assumed that some of the data distribution was at least close between subjects since the data were coming from the same gestures across all subjects. However, other factors not considered in this study may have potentially affected the classification performance. For example, because the sensors of the Myo Armband are restricted to a specific position in the band, they cannot be placed on specific muscles, as their placement is greatly influenced by the circumference of the user's forearm, which in our study ranged from 21.59–31.75 cm. This sensor placement greatly affects the reading of the EMG signal by introducing some crosstalk between muscles, *i.e.*, undesired EMG signals from surrounding muscles [107].

Another potential factor that may have affected the distribution of the data, was the donning of the Myo Armband by the different users, as this can introduce some electrode shift that affects the classification accuracy [108]. Even though the PAC classification algorithm was meant to solve this issue, the data distribution of a gesture from a specific subject may have overlapped with a different gesture from other subjects. Furthermore, the introduction of the IMU features into the classification method, may have caused some class imbalance issues, by potentially substituting support vectors that belonged to a different class.

Even though the overall classification accuracies for the 10 and 7 gesture datasets where above chance (10% and 14.28%, respectively), these results suggest that further improvement needs to be done. In this sense, in order to potentially use the PAC classification method in a user-independent scenario, it is important to account for the change in the data distribution introduced by multiple subjects, as this has a considerable impact in the classification performance.

## 5.2 Adaptive LS-SVM Classification

After classifying the EMG and IMU data using the PAC classification method, the classification performance using a different classification method was explored. This section describes the classification results of the 10 and 7 gesture datasets using the Adaptive LS-SVM classification method described in Section 4.2. For each gesture dataset, features extracted from the EMG were classified first using this method. Then, the IMU features were combined with the EMG feature sets for further classification.

### 5.2.1 Adaptive LS-SVM: 10 Gesture Classification

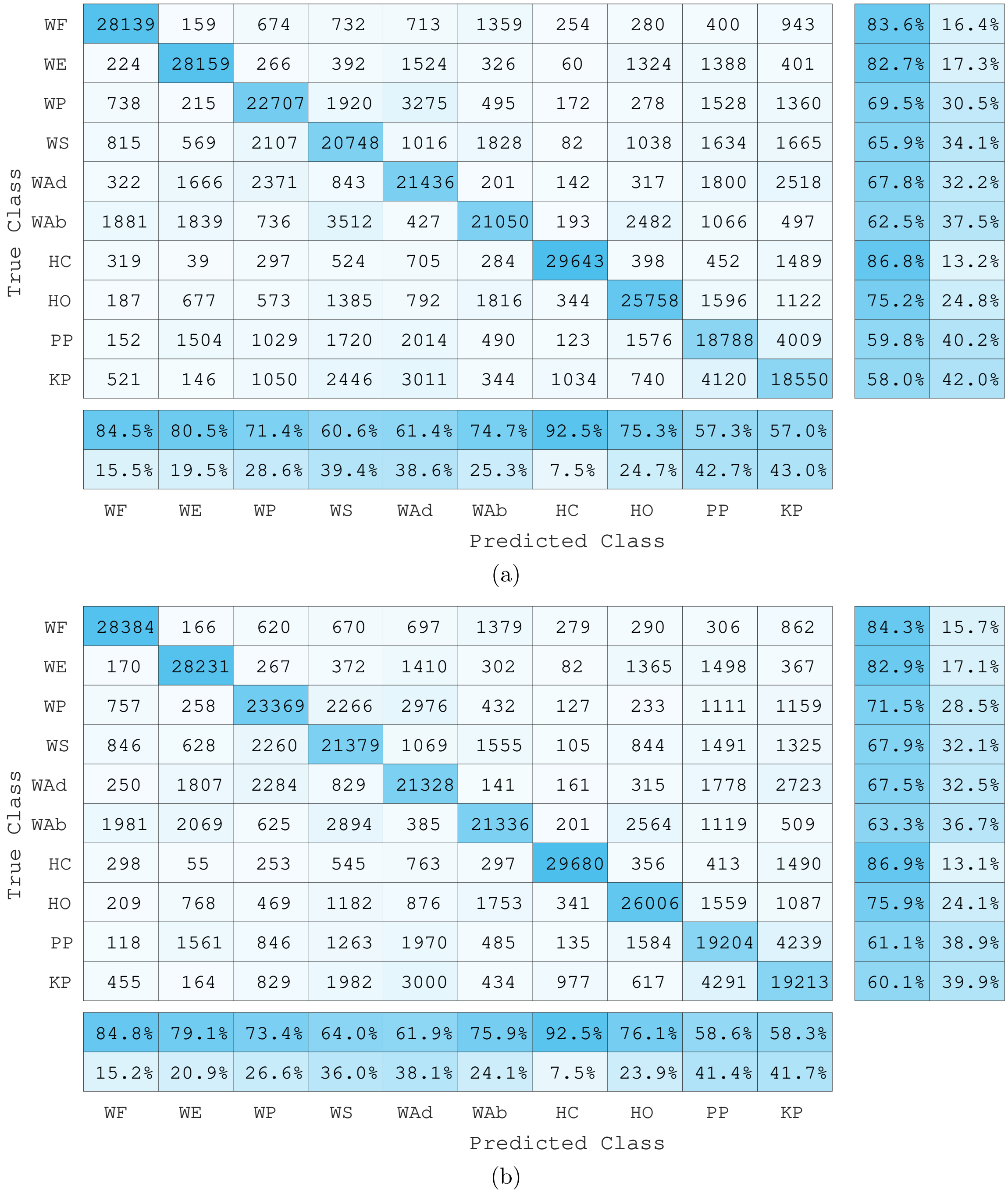
Similarly to the PAC classification method, for each cross-validation iteration, a LS-SVM model was created for each subject in the cross-validation iteration training set. Then, each subject in the cross-validation testing set was tested using the Adaptive LS-SVM classification algorithm. This procedure was repeated for both the EMG, and the EMG and IMU datasets.

Classification results from each cross-validation iteration using the Adaptive LS-SVM classification method are presented in Table 5.3. Further, Figure 5.3 shows the confusion matrices for the twenty-two testing subjects when classifying the 10 gestures in a user-independent scenario using EMG data only (Figure 5.3a), and a combination of EMG and IMU data (Figure 5.3b). Each column of each confusion matrix represents the instances in a predicted class, while each row represents the instances in an actual class. For EMG data only, classification accuracies using the Adaptive LS-SVM classification algorithm ranged from 55.64–81.14% with a mean classification accuracy of 71.14%(±7.37%), whereas for the combined EMG and IMU sensor modality, accuracies ranged from 56.39–82.40% with a mean accuracy of 72.10%(±7.24).

Following a similar procedure to the PAC classification method, for each gesture, the precision and recall scores were also calculated. The precision and recall scores for each classified gesture using EMG data only, and EMG and IMU data combined, were calculated from the confusion matrix in Figure 5.3a, and in Figure 5.3b, respectively.

Table 5.3: Classification accuracies of 10 gestures for each cross-validation iteration set using the Adaptive LS-SVM classification method. The best classification result for each subject within each cross-validation iteration testing set is shown in bold.

| Cross-Validation Iteration | | Classification Accuracy (%) | |
|---|---|---|---|
| Training Set | Testing Set | EMG | EMG + IMU |
| CVF2, CVF3, CVF4, CVF5 | S10 | 75.43 | **76.13** |
| | S16 | 60.06 | **62.08** |
| | S17 | 79.74 | **79.79** |
| | S18 | 79.74 | **80.43** |
| CVF1, CVF3, CVF4, CVF5 | S13 | 75.44 | **75.80** |
| | S14 | 73.48 | **75.87** |
| | S19 | 77.11 | **77.28** |
| | S22 | **78.98** | 78.54 |
| | S23 | **66.36** | 65.91 |
| CVF1, CVF2, CVF4, CVF5 | S4 | 77.11 | **79.89** |
| | S6 | 69.96 | **71.14** |
| | S12 | 67.29 | **68.04** |
| | S20 | 66.97 | **67.71** |
| | S24 | 73.86 | **74.84** |
| CVF1, CVF2, CVF3, CVF5 | S3 | 81.14 | **82.40** |
| | S5 | 74.17 | **74.29** |
| | S7 | 70.20 | **70.83** |
| | S25 | **60.63** | 60.62 |
| CVF1, CVF2, CVF3, CVF4 | S2 | 58.34 | **61.10** |
| | S9 | 73.84 | **74.63** |
| | S11 | 69.69 | **72.48** |
| | S15 | 55.64 | **56.39** |



True Class

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WF | 28139 | 159 | 674 | 732 | 713 | 1359 | 254 | 280 | 400 | 943 | 83.6% | 16.4% |
| WE | 224 | 28159 | 266 | 392 | 1524 | 326 | 60 | 1324 | 1388 | 401 | 82.7% | 17.3% |
| WP | 738 | 215 | 22707 | 1920 | 3275 | 495 | 172 | 278 | 1528 | 1360 | 69.5% | 30.5% |
| WS | 815 | 569 | 2107 | 20748 | 1016 | 1828 | 82 | 1038 | 1634 | 1665 | 65.9% | 34.1% |
| WAd | 322 | 1666 | 2371 | 843 | 21436 | 201 | 142 | 317 | 1800 | 2518 | 67.8% | 32.2% |
| WAb | 1881 | 1839 | 736 | 3512 | 427 | 21050 | 193 | 2482 | 1066 | 497 | 62.5% | 37.5% |
| HC | 319 | 39 | 297 | 524 | 705 | 284 | 29643 | 398 | 452 | 1489 | 86.8% | 13.2% |
| HO | 187 | 677 | 573 | 1385 | 792 | 1816 | 344 | 25758 | 1596 | 1122 | 75.2% | 24.8% |
| PP | 152 | 1504 | 1029 | 1720 | 2014 | 490 | 123 | 1576 | 18788 | 4009 | 59.8% | 40.2% |
| KP | 521 | 146 | 1050 | 2446 | 3011 | 344 | 1034 | 740 | 4120 | 18550 | 58.0% | 42.0% |
| | 84.5% | 80.5% | 71.4% | 60.6% | 61.4% | 74.7% | 92.5% | 75.3% | 57.3% | 57.0% | | |
| | 15.5% | 19.5% | 28.6% | 39.4% | 38.6% | 25.3% | 7.5% | 24.7% | 42.7% | 43.0% | | |
| | WF | WE | WP | WS | WAd | WAb | HC | HO | PP | KP | | |

Predicted Class

(a)

True Class

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WF | 28384 | 166 | 620 | 670 | 697 | 1379 | 279 | 290 | 306 | 862 | 84.3% | 15.7% |
| WE | 170 | 28231 | 267 | 372 | 1410 | 302 | 82 | 1365 | 1498 | 367 | 82.9% | 17.1% |
| WP | 757 | 258 | 23369 | 2266 | 2976 | 432 | 127 | 233 | 1111 | 1159 | 71.5% | 28.5% |
| WS | 846 | 628 | 2260 | 21379 | 1069 | 1555 | 105 | 844 | 1491 | 1325 | 67.9% | 32.1% |
| WAd | 250 | 1807 | 2284 | 829 | 21328 | 141 | 161 | 315 | 1778 | 2723 | 67.5% | 32.5% |
| WAb | 1981 | 2069 | 625 | 2894 | 385 | 21336 | 201 | 2564 | 1119 | 509 | 63.3% | 36.7% |
| HC | 298 | 55 | 253 | 545 | 763 | 297 | 29680 | 356 | 413 | 1490 | 86.9% | 13.1% |
| HO | 209 | 768 | 469 | 1182 | 876 | 1753 | 341 | 26006 | 1559 | 1087 | 75.9% | 24.1% |
| PP | 118 | 1561 | 846 | 1263 | 1970 | 485 | 135 | 1584 | 19204 | 4239 | 61.1% | 38.9% |
| KP | 455 | 164 | 829 | 1982 | 3000 | 434 | 977 | 617 | 4291 | 19213 | 60.1% | 39.9% |
| | 84.8% | 79.1% | 73.4% | 64.0% | 61.9% | 75.9% | 92.5% | 76.1% | 58.6% | 58.3% | | |
| | 15.2% | 20.9% | 26.6% | 36.0% | 38.1% | 24.1% | 7.5% | 23.9% | 41.4% | 41.7% | | |
| | WF | WE | WP | WS | WAd | WAb | HC | HO | PP | KP | | |

Predicted Class

(b)

Figure 5.3: Adaptive LS-SVM confusion matrix of the 10 gestures collected using EMG data only (a), and EMG and IMU data (b) from all testing subjects. The last two rows from each confusion matrix represent the precision score percentages and the false positive rate percentages (the cumulative number of false positives) of each class, in the top and bottom row, respectively. Similarly, the last two columns represent the recall score percentages and the false negatives rate percentages (the cumulative number of false negatives) of each class, in the left and right column, respectively.

For all of the gestures, the lowest precision scores among the twenty-two subjects when using EMG data only was for the the Precision Pinch and the Key Pinch gestures with overall precision scores of 57.3% and 57.0%, respectively. The highest precision was again for the Hand Close gesture, which achieved an overall precision of 92.5%. At the same time, the lowest recall score belonged to the Key Pinch gesture with a value of 58%, whereas the highest recall score was again for the Hand Close gesture, with an overall score of 86.8%. Likewise, the lowest precision scores among the twenty-two subjects when using both modalities were for the Precision Pinch and the Key Pinch gestures with overall scores of 58.6% and 58.3%, respectively. Further, the highest precision score was for the Hand Close gesture with an overall precision of 92.5%. Similarly, the lowest recall score belonged to the Key Pinch gesture with a value of 60.1%, whereas the highest recall score was for the Hand Close gesture, with an overall score of 86.9%.

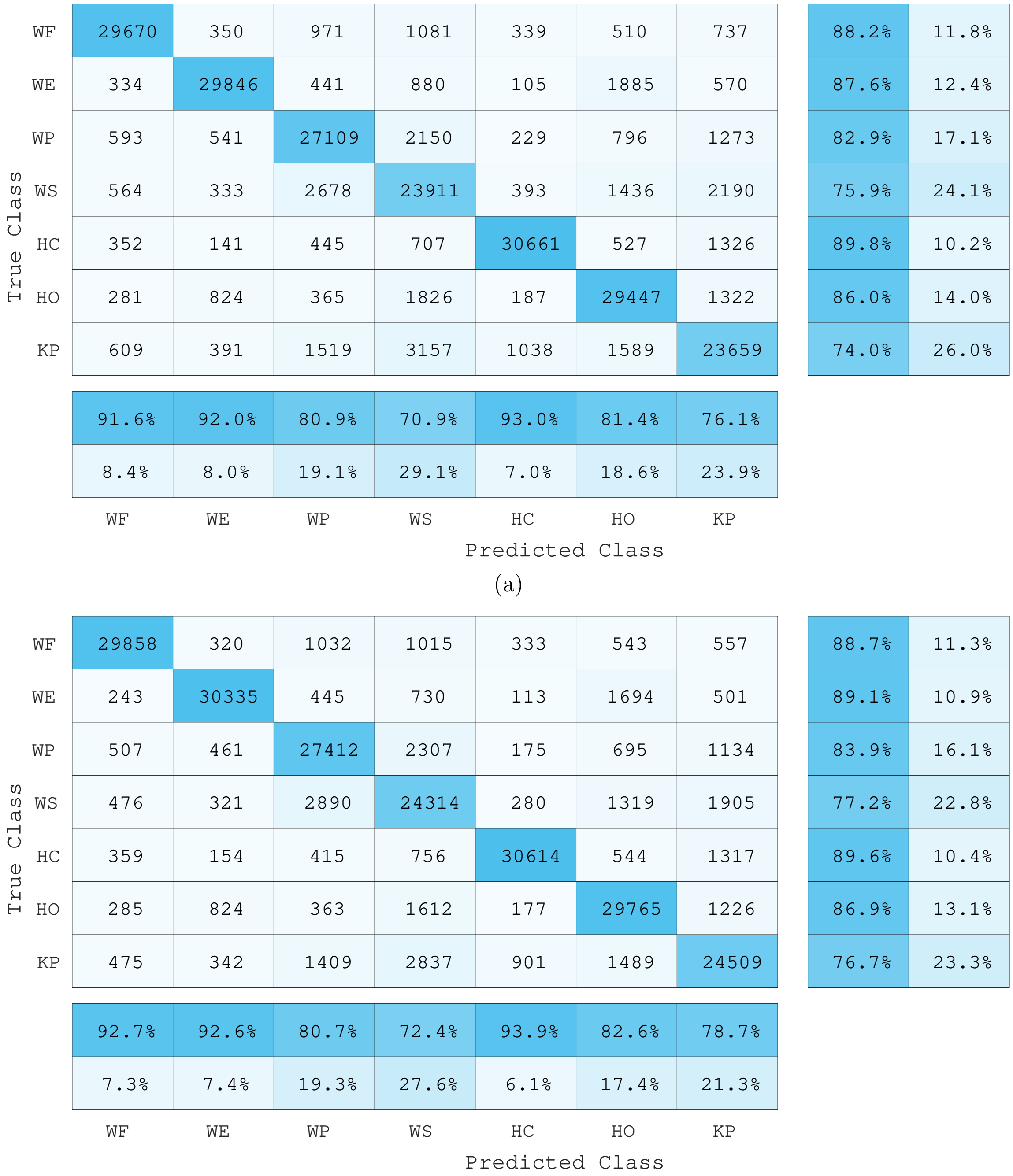
Although most of the classification accuracies of this method were above 70%, the classification algorithm was further optimized following a similar procedure as the PAC classification method, *i.e.*, by removing the data corresponding to some of the gestures. The next section will describe the classification performance using this 7 gesture dataset.

### 5.2.2 Adaptive LS-SVM: 7 Gesture Classification

After reducing the number of gestures in the gesture dataset, a new LS-SVM classifier was created for each subject so that the performance of the Adaptive LS-SVM classification method could be assessed using the cross-validation testing sets during each cross-validation iteration. The new classification results from each cross-validation iteration using the 7 gesture dataset are presented in Table 5.4. Figure 5.4 shows the confusion matrices for the twenty-two testing subjects when classifying the new 7 gestures in a user-independent scenario using EMG data only (Figure 5.4a), and a combination of EMG and IMU data (Figure 5.4b). For EMG data only, classification accuracies ranged from 61.69–92.54% with a mean classification accuracy of 83.45%(±7.46%), whereas for the combined EMG and IMU sensor modality, accuracies ranged from 62.50–92.88% with a mean accuracy of 84.55%(±7.32). Similar to the 10 gesture dataset, the precision and recall scores for each classified gesture using EMG data only, and EMG and IMU data combined, were calculated from the confusion matrix in Figure 5.4a, and in Figure 5.4b, respectively.

Table 5.4: Classification accuracies of 7 gestures for each cross-validation iteration set using the Adaptive LS-SVM classification method. The best classification result for each subject within each cross-validation iteration testing set is shown in bold.

| Cross-Validation Iteration | | Classification Accuracy (%) | |
|---|---|---|---|
| Training Set | Testing Set | EMG | EMG + IMU |
| CVF2, CVF3, CVF4, CVF5 | S10 | 83.25 | **84.57** |
| | S16 | 66.29 | **67.32** |
| | S17 | **88.77** | 88.33 |
| | S18 | 84.00 | **85.60** |
| CVF1, CVF3, CVF4, CVF5 | S13 | 87.84 | **88.20** |
| | S14 | 87.85 | **89.19** |
| | S19 | 92.54 | **92.88** |
| | S22 | 89.53 | **89.59** |
| | S23 | 82.91 | **85.64** |
| CVF1, CVF2, CVF4, CVF5 | S4 | 87.35 | **88.33** |
| | S6 | 84.75 | **85.37** |
| | S12 | 87.48 | **87.95** |
| | S20 | 74.86 | **77.48** |
| | S24 | 83.13 | **83.93** |
| CVF1, CVF2, CVF3, CVF5 | S3 | 89.66 | **90.47** |
| | S5 | 89.82 | **90.71** |
| | S7 | 85.59 | **86.69** |
| | S25 | 80.26 | **80.44** |
| CVF1, CVF2, CVF3, CVF4 | S2 | 83.46 | **86.00** |
| | S9 | 80.00 | **80.93** |
| | S11 | 84.97 | **87.85** |
| | S15 | 61.69 | **62.50** |



| True Class \ Predicted Class | WF | WE | WP | WS | HC | HO | KP | | |
|---|---|---|---|---|---|---|---|---|---|
| WF | 29670 | 350 | 971 | 1081 | 339 | 510 | 737 | 88.2% | 11.8% |
| WE | 334 | 29846 | 441 | 880 | 105 | 1885 | 570 | 87.6% | 12.4% |
| WP | 593 | 541 | 27109 | 2150 | 229 | 796 | 1273 | 82.9% | 17.1% |
| WS | 564 | 333 | 2678 | 23911 | 393 | 1436 | 2190 | 75.9% | 24.1% |
| HC | 352 | 141 | 445 | 707 | 30661 | 527 | 1326 | 89.8% | 10.2% |
| HO | 281 | 824 | 365 | 1826 | 187 | 29447 | 1322 | 86.0% | 14.0% |
| KP | 609 | 391 | 1519 | 3157 | 1038 | 1589 | 23659 | 74.0% | 26.0% |
| | 91.6% | 92.0% | 80.9% | 70.9% | 93.0% | 81.4% | 76.1% | | |
| | 8.4% | 8.0% | 19.1% | 29.1% | 7.0% | 18.6% | 23.9% | | |

(a)

| True Class \ Predicted Class | WF | WE | WP | WS | HC | HO | KP | | |
|---|---|---|---|---|---|---|---|---|---|
| WF | 29858 | 320 | 1032 | 1015 | 333 | 543 | 557 | 88.7% | 11.3% |
| WE | 243 | 30335 | 445 | 730 | 113 | 1694 | 501 | 89.1% | 10.9% |
| WP | 507 | 461 | 27412 | 2307 | 175 | 695 | 1134 | 83.9% | 16.1% |
| WS | 476 | 321 | 2890 | 24314 | 280 | 1319 | 1905 | 77.2% | 22.8% |
| HC | 359 | 154 | 415 | 756 | 30614 | 544 | 1317 | 89.6% | 10.4% |
| HO | 285 | 824 | 363 | 1612 | 177 | 29765 | 1226 | 86.9% | 13.1% |
| KP | 475 | 342 | 1409 | 2837 | 901 | 1489 | 24509 | 76.7% | 23.3% |
| | 92.7% | 92.6% | 80.7% | 72.4% | 93.9% | 82.6% | 78.7% | | |
| | 7.3% | 7.4% | 19.3% | 27.6% | 6.1% | 17.4% | 21.3% | | |

(b)

Figure 5.4: Adaptive LS-SVM confusion matrix of the 7 gestures collected using EMG data only (a), and EMG and IMU data (b) from all testing subjects. The last two rows from each confusion matrix represent the precision score percentages and the false positive rate percentages (the cumulative number of false positives) of each class, in the top and bottom row, respectively. Similarly, the last two columns represent the recall score percentages and the false negatives rate percentages (the cumulative number of false negatives) of each class, in the left and right column, respectively.

For all of the gestures, the lowest precision score among the twenty-two subjects when using EMG data only was for the Wrist Supination gesture with an overall precision of 70.9%, and the highest precision was again for the Hand Close gesture with an overall precision of 93.0%. Furthermore, the lowest recall score belonged to the Key Pinch gesture with a value of 74%, whereas the highest recall score was again for the Hand Close gesture, with an overall score of 89.8%. Likewise, the lowest precision scores among the twenty-two subjects when using both modalities was for the Wrist Supination gesture with an overall precision of 72.4%, and the highest precision score was for the Hand Close gesture with an overall precision of 93.9%. Similarly, the lowest recall score belonged to the Key Pinch gesture with a value of 76.7%, whereas the highest recall score was for the Hand Close gesture, with an overall score of 89.6%.

### 5.2.3 Adaptive LS-SVM: Discussion

The results obtained from the Adaptive LS-SVM classification method improved when using a combination of the EMG and IMU sensor data when classifying 10 gestures. In this sense, the mean recognition accuracy of this sensor modality (72.1%) was slightly greater than the mean recognition accuracy obtained when using the EMG data only (71.14%). Even though there was no clear difference of the means, the paired sample $t$ test showed that this small difference was statistically significant ($p < 0.001$). Regarding the 7 gesture dataset, a significant increase ($p < 0.001$) was also observed between the combined EMG and IMU data mean recognition accuracy (84.55%) and the EMG data only mean recognition accuracy (83.45%). Furthermore, from the confusion matrices, it can be observed that both, the precision and recall scores, scaled linearly during the classification using the EMG data only, and the combination of the EMG and IMU data. This behaviour was observed in both the 10 and the 7 gesture datasets. This indicates that by combining the EMG and IMU features, not only did the classification performance improve, but also the ability of the model to recognize each individual gesture better.

Furthermore, the addition of the IMU features did not affect the behaviour of the Adaptive LS-SVM classification method. In this sense, the distribution of the data among subjects was not affected, which is why the difference between the mean classified accuracies was of 1.091% and 0.956% for the 10 and 7 gesture datasets, respectively. However, similar to what Tommasi *et al.*[81]

found, there were subjects whose recognition accuracy performed the worst. This was because their data distribution was not able to match those of the pretrained models, thus making the norm of $\vec{\beta}$ in Equation (4.21) small enough that there was no transfer of prior knowledge. This indicates that information from more subjects needs to be recorded in an attempt to compensate for this issue. On another note, by using a linear combination from multiple prior models, *i.e.*, by selecting the best information from multiple pretrained models, the Adaptive LS-SVM classification method was able to cope with most of the limitations that the PAC classification algorithm was not able to handle. For example, by not substituting the support vectors of previous models but instead using them as a starting point for training a new predictive model, it was possible to avoid class imbalances by preserving the support vectors for each class.

These results suggest that the Adaptive LS-SVM classification based on the combination of EMG and IMU sensor data can be effectively used in a user-independent scenario. However, as explained before, data from more subjects need to be recorded to have a bigger database of pretrained models. By doing so, it will be easier to match the data distribution of new users to one of these pretrained models.

## 5.3 Bilinear Model-Based Classification

Having classified the EMG and IMU data using the PAC, and the LS-SVM classification models, the bilinear model-based classification method was explored. The classification results of the 10 and 7 gesture datasets using this method, which was described in Section 4.3, are presented. For each cross-validation iteration, a symmetric bilinear EMG model was created using the EMG data of all of the subjects in the training set. Then, a new feature matrix was formed using the motion-dependent factors of this bilinear EMG model. This feature matrix was then used to train a NN model, which was finally tested using the cross-validation iteration testing set. To test the performance of the bilinear model-based classification method when also using IMU data, IMU signals from each subject in the corresponding cross-validation iteration training set were averaged to obtain a new feature matrix that consisted of 12 features. This feature matrix was then combined with the motion-dependent factors feature matrix, and then used to train a NN model. The classification performance of this NN model was tested on the cross-validation iteration testing set.

### 5.3.1 Bilinear Models-Based Classification: 10 Gesture Classification

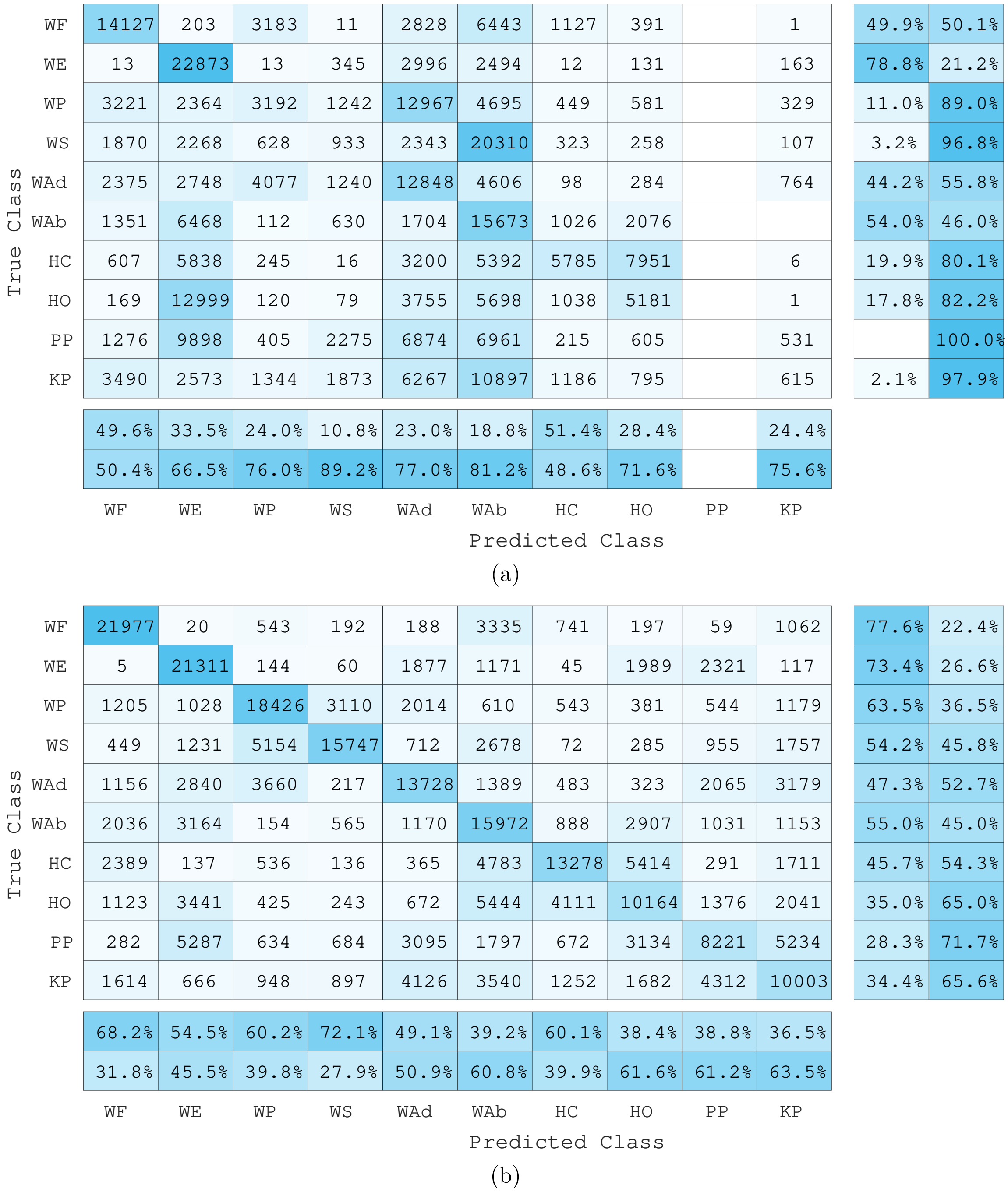
Classification results from each cross-validation iteration after using the bilinear model-based classification method are presented in Table 5.5. Figure 5.5 shows the confusion matrices for the twenty-two testing subjects when classifying the 10 gestures in a user-independent scenario using the bilinear EMG model data only (Figure 5.5a), and a combination of the bilinear EMG model and IMU data (Figure 5.5b). Each column of each confusion matrix represents the instances in a predicted class, while each row represents the instances in an actual class. For the bilinear EMG model data only, classification accuracies using this method ranged from 10.99–40.61% with a mean classification accuracy of 28.09%(±6.85%), whereas for the combined bilinear EMG model and IMU data, accuracies ranged from 22.79–75.42% with a mean accuracy of 51.44%(±10.54).

Further, for each gesture, the precision and recall scores were also calculated. The precision score for each classified gesture using the EMG bilinear model data only, and the EMG bilinear model and IMU data combined, were calculated from the confusion matrix in Figure 5.5a, and in Figure 5.5b, respectively. The same procedure was followed for calculating the recall scores.

For all of the gestures, the lowest precision score among the twenty-two subjects when using the bilinear EMG model data only was for the Precision Pinch gesture, which the NN failed to classify, followed by the Wrist Supination gesture with an overall precision score of 10.8%. Further, the highest precision was for the Hand Close gesture with an overall precision of 51.4%. At the same time, the lowest recall score belonged to the Precision Pinch gesture with a value of 0%, due to the NN failing to classify this gesture. The highest recall score was for the Wrist Extension gesture, with an overall score of 78.8%. On the other hand, the lowest precision scores among the twenty-two subjects when using both modalities was for the Key Pinch gesture with an overall precision of 36.5%, and the highest precision score was for the Wrist Flexion gesture with an overall precision of 72.1%. Similarly, the lowest recall score belonged to the Precision Pinch gesture with a value of 28.3%, whereas the highest recall score was for the Wrist Flexion gesture, with an overall score of 77.6%.

Table 5.5: Classification accuracies of 10 gestures for each cross-validation iteration set using bilinear models classification method. The best classification result for each subject within each cross-validation iteration testing set is shown in bold.

| Cross-Validation Iteration | | Classification Accuracy (%) | |
|---|---|---|---|
| Training Set | Testing Set | EMG | EMG + IMU |
| CVF2, CVF3, CVF4, CVF5 | S10 | 35.75 | **49.02** |
| | S16 | 33.62 | **53.35** |
| | S17 | 25.42 | **56.94** |
| | S18 | 38.90 | **57.29** |
| CVF1, CVF3, CVF4, CVF5 | S13 | 20.19 | **52.65** |
| | S14 | 10.99 | **22.79** |
| | S19 | 27.74 | **54.99** |
| | S22 | 24.32 | **53.37** |
| | S23 | 27.52 | **66.56** |
| CVF1, CVF2, CVF4, CVF5 | S4 | 38.63 | **75.41** |
| | S6 | 26.27 | **58.55** |
| | S12 | 20.10 | **35.43** |
| | S20 | 24.72 | **50.56** |
| | S24 | 25.32 | **48.99** |
| CVF1, CVF2, CVF3, CVF5 | S3 | 26.04 | **40.26** |
| | S5 | 40.61 | **51.32** |
| | S7 | 26.46 | **57.43** |
| | S25 | 28.68 | **58.97** |
| CVF1, CVF2, CVF3, CVF4 | S2 | 29.87 | **46.07** |
| | S9 | 26.08 | **49.19** |
| | S11 | 26.76 | **46.43** |
| | S15 | 34.00 | **46.09** |



|  | WF | WE | WP | WS | WAd | WAb | HC | HO | PP | KP |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WF | 14127 | 203 | 3183 | 11 | 2828 | 6443 | 1127 | 391 |  | 1 | 49.9% | 50.1% |
| WE | 13 | 22873 | 13 | 345 | 2996 | 2494 | 12 | 131 |  | 163 | 78.8% | 21.2% |
| WP | 3221 | 2364 | 3192 | 1242 | 12967 | 4695 | 449 | 581 |  | 329 | 11.0% | 89.0% |
| WS | 1870 | 2268 | 628 | 933 | 2343 | 20310 | 323 | 258 |  | 107 | 3.2% | 96.8% |
| WAd | 2375 | 2748 | 4077 | 1240 | 12848 | 4606 | 98 | 284 |  | 764 | 44.2% | 55.8% |
| WAb | 1351 | 6468 | 112 | 630 | 1704 | 15673 | 1026 | 2076 |  |  | 54.0% | 46.0% |
| HC | 607 | 5838 | 245 | 16 | 3200 | 5392 | 5785 | 7951 |  | 6 | 19.9% | 80.1% |
| HO | 169 | 12999 | 120 | 79 | 3755 | 5698 | 1038 | 5181 |  | 1 | 17.8% | 82.2% |
| PP | 1276 | 9898 | 405 | 2275 | 6874 | 6961 | 215 | 605 |  | 531 |  | 100.0% |
| KP | 3490 | 2573 | 1344 | 1873 | 6267 | 10897 | 1186 | 795 |  | 615 | 2.1% | 97.9% |
|  | 49.6% | 33.5% | 24.0% | 10.8% | 23.0% | 18.8% | 51.4% | 28.4% |  | 24.4% |  |  |
|  | 50.4% | 66.5% | 76.0% | 89.2% | 77.0% | 81.2% | 48.6% | 71.6% |  | 75.6% |  |  |

(a)

|  | WF | WE | WP | WS | WAd | WAb | HC | HO | PP | KP |  |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WF | 21977 | 20 | 543 | 192 | 188 | 3335 | 741 | 197 | 59 | 1062 | 77.6% | 22.4% |
| WE | 5 | 21311 | 144 | 60 | 1877 | 1171 | 45 | 1989 | 2321 | 117 | 73.4% | 26.6% |
| WP | 1205 | 1028 | 18426 | 3110 | 2014 | 610 | 543 | 381 | 544 | 1179 | 63.5% | 36.5% |
| WS | 449 | 1231 | 5154 | 15747 | 712 | 2678 | 72 | 285 | 955 | 1757 | 54.2% | 45.8% |
| WAd | 1156 | 2840 | 3660 | 217 | 13728 | 1389 | 483 | 323 | 2065 | 3179 | 47.3% | 52.7% |
| WAb | 2036 | 3164 | 154 | 565 | 1170 | 15972 | 888 | 2907 | 1031 | 1153 | 55.0% | 45.0% |
| HC | 2389 | 137 | 536 | 136 | 365 | 4783 | 13278 | 5414 | 291 | 1711 | 45.7% | 54.3% |
| HO | 1123 | 3441 | 425 | 243 | 672 | 5444 | 4111 | 10164 | 1376 | 2041 | 35.0% | 65.0% |
| PP | 282 | 5287 | 634 | 684 | 3095 | 1797 | 672 | 3134 | 8221 | 5234 | 28.3% | 71.7% |
| KP | 1614 | 666 | 948 | 897 | 4126 | 3540 | 1252 | 1682 | 4312 | 10003 | 34.4% | 65.6% |
|  | 68.2% | 54.5% | 60.2% | 72.1% | 49.1% | 39.2% | 60.1% | 38.4% | 38.8% | 36.5% |  |  |
|  | 31.8% | 45.5% | 39.8% | 27.9% | 50.9% | 60.8% | 39.9% | 61.6% | 61.2% | 63.5% |  |  |

(b)

Figure 5.5: bilinear models-based confusion matrix of the 10 gestures collected using EMG data only (a), and EMG and IMU data (b) from all testing subjects. The last two rows from each confusion matrix represent the precision score percentages and the false positive rate percentages (the cumulative number of false positives) of each class, in the top and bottom row, respectively. Similarly, the last two columns represent the recall score percentages and the false negatives rate percentages (the cumulative number of false negatives) of each class, in the left and right column, respectively.

Following a similar approach to the previous classification methods explained in this chapter, the gesture datasets where further optimized in an attempt to improve the classification accuracy. The next section will discuss the results of said optimization.

### 5.3.2 Bilinear Models-Based Classification: 7 Gesture Classification

After reducing the number of gestures in the gesture dataset, a new bilinear model had to be created from the EMG signals of each subject within each cross-validation iteration training set. Then, a similar procedure described in the previous section was followed to classify the gestures of each subject in the cross-validation testing set using a NN classifier.

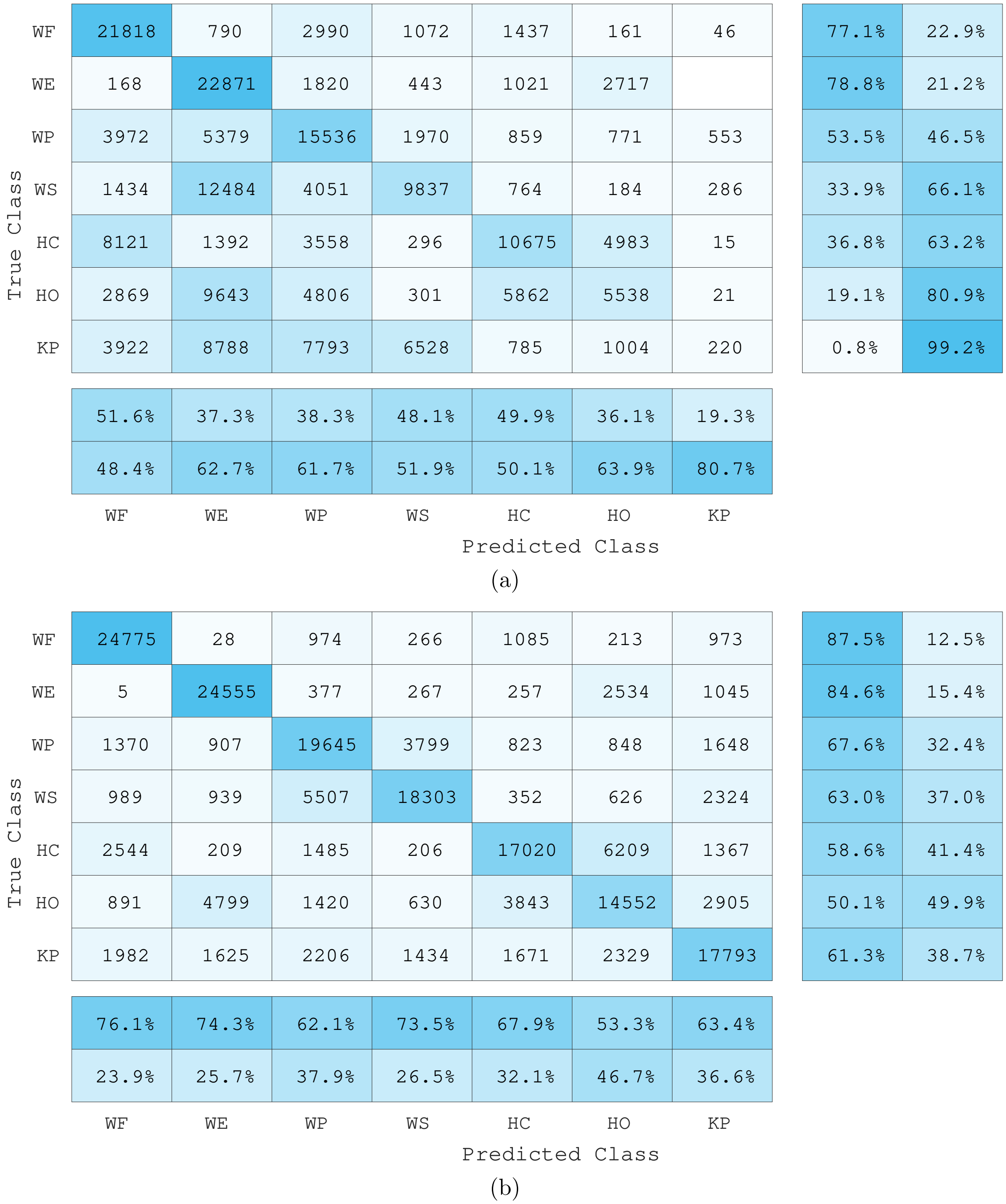
The new classification results from each cross-validation iteration using the 7 gesture dataset are presented in Table 5.6. Moreover, Figure 5.6 shows the confusion matrices for the twenty-two testing subjects when classifying the new 7 gestures in a user-independent scenario using the EMG bilinear model data only (Figure 5.6a), and a combination of the EMG bilinear model and IMU data (Figure 5.6b). For the EMG bilinear model data only, classification accuracies ranged from 21.23–67.31% with a mean classification accuracy of 42.82%(±11.53%), whereas for the combined EMG bilinear model and IMU sensor data, accuracies ranged from 42.96–84.86% with a mean accuracy of 67.53%(±11.64).

Similarly to the 10 gesture dataset, the precision and recall scores for each classified gesture using the EMG bilinear model data only, and the EMG bilinear model and IMU data combined, were calculated from the confusion matrix in Figure 5.6a, and in Figure 5.6b, respectively.

For all of the gestures, the lowest precision score among the twenty-two subjects when using the EMG bilinear model data only was for the Key Pinch gesture with an overall precision of 19.3%, and the highest precision corresponded to the Wrist Flexion gesture with an overall precision of 51.6%. At the same time, the lowest recall score belonged to the Key Pinch gesture with a value of 0.8%, whereas the highest recall score was for the Wrist Extension gesture, with an overall score of 78.8%. On the other hand, the lowest precision score among the twenty-two subjects when using both modalities was for the Hand Open gesture with an overall precision of 53.3%, and the highest precision score was for the Wrist Flexion gesture with an overall precision of 76.1%. Similarly, the lowest recall score belonged to the Hand Open gesture with a value of 50.1%, whereas the highest recall score was for the Wrist Flexion gesture, with an overall score of 87.5%.

Table 5.6: Classification accuracies of 7 gestures for each cross-validation iteration set using bilinear models-based classification method. The best classification result for each subject within each cross-validation iteration testing set is shown in bold.

| Cross-Validation Iteration | | Classification Accuracy (%) | |
|---|---|---|---|
| Training Set | Testing Set | EMG | EMG + IMU |
| CVF2, CVF3, CVF4, CVF5 | S10 | 30.20 | **62.80** |
| | S16 | 32.10 | **59.96** |
| | S17 | 36.34 | **74.11** |
| | S18 | 32.38 | **60.77** |
| CVF1, CVF3, CVF4, CVF5 | S13 | 51.52 | **77.34** |
| | S14 | 36.22 | **55.73** |
| | S19 | 67.31 | **80.46** |
| | S22 | 57.74 | **70.36** |
| | S23 | 60.14 | **80.87** |
| CVF1, CVF2, CVF4, CVF5 | S4 | 56.36 | **84.86** |
| | S6 | 43.75 | **77.51** |
| | S12 | 30.77 | **48.27** |
| | S20 | 42.05 | **66.38** |
| | S24 | 37.27 | **50.15** |
| CVF1, CVF2, CVF3, CVF5 | S3 | 21.23 | **42.96** |
| | S5 | 40.19 | **78.57** |
| | S7 | 48.90 | **78.99** |
| | S25 | 40.11 | **69.00** |
| CVF1, CVF2, CVF3, CVF4 | S2 | 55.31 | **75.09** |
| | S9 | 47.81 | **68.56** |
| | S11 | 38.95 | **57.78** |
| | S15 | 35.36 | **65.10** |



| True Class | WF | WE | WP | WS | HC | HO | KP | | |
|---|---|---|---|---|---|---|---|---|---|
| WF | 21818 | 790 | 2990 | 1072 | 1437 | 161 | 46 | 77.1% | 22.9% |
| WE | 168 | 22871 | 1820 | 443 | 1021 | 2717 | | 78.8% | 21.2% |
| WP | 3972 | 5379 | 15536 | 1970 | 859 | 771 | 553 | 53.5% | 46.5% |
| WS | 1434 | 12484 | 4051 | 9837 | 764 | 184 | 286 | 33.9% | 66.1% |
| HC | 8121 | 1392 | 3558 | 296 | 10675 | 4983 | 15 | 36.8% | 63.2% |
| HO | 2869 | 9643 | 4806 | 301 | 5862 | 5538 | 21 | 19.1% | 80.9% |
| KP | 3922 | 8788 | 7793 | 6528 | 785 | 1004 | 220 | 0.8% | 99.2% |
| | 51.6% | 37.3% | 38.3% | 48.1% | 49.9% | 36.1% | 19.3% | | |
| | 48.4% | 62.7% | 61.7% | 51.9% | 50.1% | 63.9% | 80.7% | | |

Predicted Class

(a)

| True Class | WF | WE | WP | WS | HC | HO | KP | | |
|---|---|---|---|---|---|---|---|---|---|
| WF | 24775 | 28 | 974 | 266 | 1085 | 213 | 973 | 87.5% | 12.5% |
| WE | 5 | 24555 | 377 | 267 | 257 | 2534 | 1045 | 84.6% | 15.4% |
| WP | 1370 | 907 | 19645 | 3799 | 823 | 848 | 1648 | 67.6% | 32.4% |
| WS | 989 | 939 | 5507 | 18303 | 352 | 626 | 2324 | 63.0% | 37.0% |
| HC | 2544 | 209 | 1485 | 206 | 17020 | 6209 | 1367 | 58.6% | 41.4% |
| HO | 891 | 4799 | 1420 | 630 | 3843 | 14552 | 2905 | 50.1% | 49.9% |
| KP | 1982 | 1625 | 2206 | 1434 | 1671 | 2329 | 17793 | 61.3% | 38.7% |
| | 76.1% | 74.3% | 62.1% | 73.5% | 67.9% | 53.3% | 63.4% | | |
| | 23.9% | 25.7% | 37.9% | 26.5% | 32.1% | 46.7% | 36.6% | | |

Predicted Class

(b)

Figure 5.6: bilinear models-based confusion matrix of the 7 gestures collected using EMG data only (a), and EMG and IMU data (b) from all testing subjects. The last two rows from each confusion matrix represent the precision score percentages and the false positive rate percentages (the cumulative number of false positives) of each class, in the top and bottom row respectively, whereas the last two columns represent the recall score percentages and the false negatives rate percentages (the cumulative number of false negatives) of each class, in the left and right column, respectively.

### 5.3.3 Bilinear Models-Based Classification: Discussion

The results obtained by using EMG based bilinear models in combination with the IMU sensor data when classifying 10 gestures, show a significant increase ($p < 0.001$) in the overall gesture recognition accuracy (51.44%) when compared to the recognition accuracy obtained by using the EMG bilinear model only (28.09%). Similar results were obtained when using the 7 gesture dataset. In this sense, when comparing the overall recognition accuracies obtained between the EMG bilinear models combined with the IMU sensor data (67.53%), and the bilinear models EMG data only (42.82%), a significant increase ($p < 0.001$) was also observed.

Furthermore, from the confusion matrices, it can be observed that, when using the data coming from the IMU, the classification model was able to increase the recall scores. For example, from the confusion matrix in Figure 5.5a, it can be seen that the classification algorithm failed to classify the Precision Pinch gesture 100% of the time. A similar case occurred with the Key Pinch gesture in the 7 gesture dataset, which was misclassified most of the time as being a Wrist Extension or a Wrist Supination gesture (Figure 5.6a). Therefore, by adding the IMU data, the NN algorithm employed was able to cope with this issue by using the information from the extra features to make more general assumptions about the corresponding class of unseen data.

Although the classification performance improved, the low recognition accuracies impose a major drawback on this classification method. This poor performance can be explained by three important factors with the first one being the classification under confounding factors, *i.e.*, the gestures being classified under different arm positions. To deal with this issue, Ishii *et al.* [109] proposed the use of a two-stage bilinear model in which during the first stage, user-dependent and postural-dependent factors were separated. Then, during the second stage, the motion-dependent factors were separated from the postural-dependent factors. However, even though the results showed an improvement during classification, the validation of their methods was obscure. In this sense, they failed to report the procedure followed to build the bilinear models, *e.g.*, the number of subjects assigned to the training and testing set or the classification algorithm parameters used were not reported. Further, no statistical analysis was performed to validate their approach.

The second important factor that affected the classification performance of the proposed method, was the donning of the Myo Armband by novel users. Similarly to the PAC classification algorithm, and explained by Matsubara *et al.* [82], the use of bilinear models required the electrodes to be placed on the exact same location for all the users. However, the difference in the dimensions of the forearm between subjects imposes a great obstacle for achieving this requirement. This suggests that the classification method can be further optimized by adopting a similar approach as in [110], were the displacement of the sensors was estimated in order to improve the classification accuracies.

Finally, the third factor was related to how the style and content variables (Equations (4.30) and (4.31)) were computed. As explained in Section 4.3.2, the dimension of these parameters are given by the values $I$ and $J$, which for this application were selected as 2 and 3 respectively. This resulted in a reduced number of features used for classification, which could have increased the bias of the NN algorithm. To cope with this issue, a similar approach used to obtain the principal components during PCA could be applied. Given that the iterative process to find the style and content variables is based on using the singular value decomposition (SVD) factorization, we can initialize the values $I$ and $J$ as the $n^{th}$ row of the diagonal matrix sigma computed for each variable, with $n$ being small enough, so that a specific percentage of the variance of the initial stacked matrix $\bar{Y}$ (Equation (4.28)) is retained.

Overall, the inclusion of the IMU features as an extra input for the classification of gestures using bilinear models, showed that this classification method has the potential to be used in a user-independent scenario. However, further improvement needs to be done to increase the efficacy of the proposed method in order to improve the recognition accuracies.

## 5.4 MLP Networks Classification

After exploring the classification performance of the first 3 user-independent classification methods, the MLP networks classification method was evaluated. This section describes the classification results of classifying the 10 and 7 gesture datasets using MLP networks, which were described in Section 4.4. For each gesture dataset, features extracted from the EMG were classified first using this method. Then, the IMU features were combined with the EMG feature sets for further classification.

### 5.4.1 MLP Networks: 10 Gesture Classification

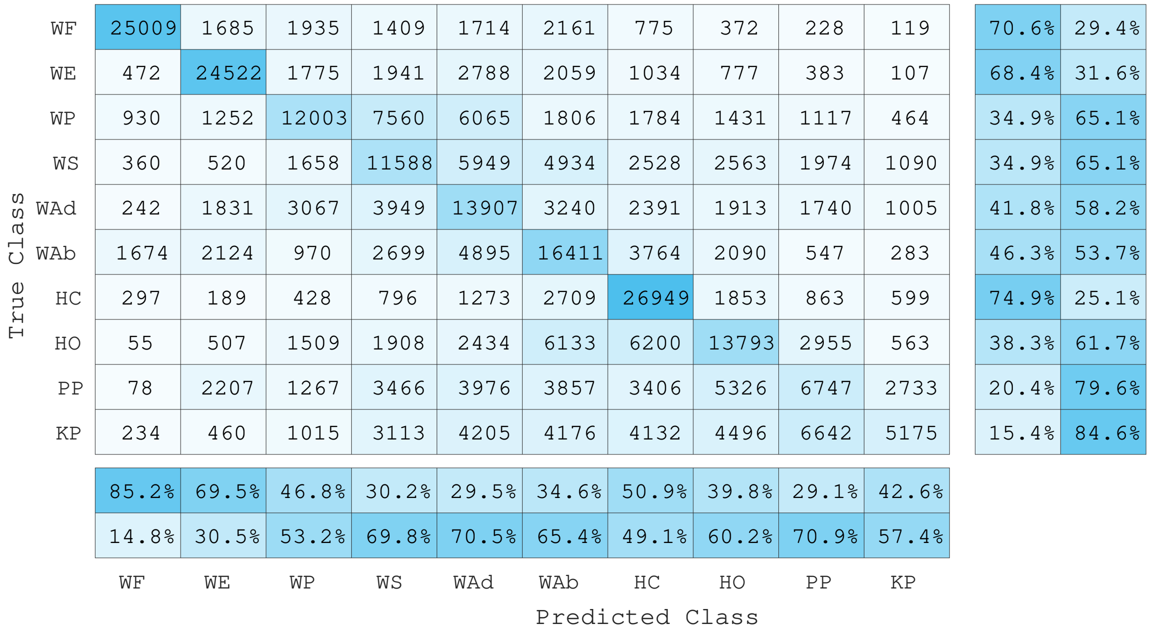
For each cross-validation iteration, a MLP network was first trained using sixty-four features extracted from the EMG data corresponding to the cross-validation training set, and then trained using twelve additional features from the IMU data. This was done to further analyze the contributions of the IMU data to the classification accuracy of the 10 gestures in a user-independent scenario. To train the network using both sensor modalities, a feature fusion approach was employed by combining features extracted from the IMU with features extracted from the EMG into a single matrix that was used as the input to the network. It is worth mentioning that, during training of the MLP network, features were reduced using PCA. However, after testing the performance of the network in the cross-validation set (Figure 4.8), it was found that the classification accuracy dropped. Therefore, no feature reduction was used for the MLP network classification. Finally, to observe the classification performance, data from each subject in the cross-validation iteration testing set were classified using the MLP network.

Classification results from each cross-validation iteration are presented in Table 5.7. Further, Figure 5.7 shows the confusion matrices for the twenty-two testing subjects when classifying the 10 gestures in a user-independent scenario using EMG data only (Figure 5.7a), and a combination of EMG and IMU data (Figure 5.7b). Each column of each confusion matrix represents the instances in a predicted class, while each row represents the instances in an actual class. For EMG data only, classification accuracies using the MLP networks ranged from 21.31–57.10% with a mean classification accuracy of 44.53%(±11.27%), whereas for the combined EMG and IMU sensor modality, accuracies ranged from 28.60–67.94% with a mean accuracy of 53.44%(±12.15).

For each gesture, the precision and recall scores were also calculated. The precision and recall scores for each classified gesture using EMG data only, and EMG and IMU data combined, were calculated from the confusion matrix in Figure 5.7a, and in Figure 5.7b, respectively.

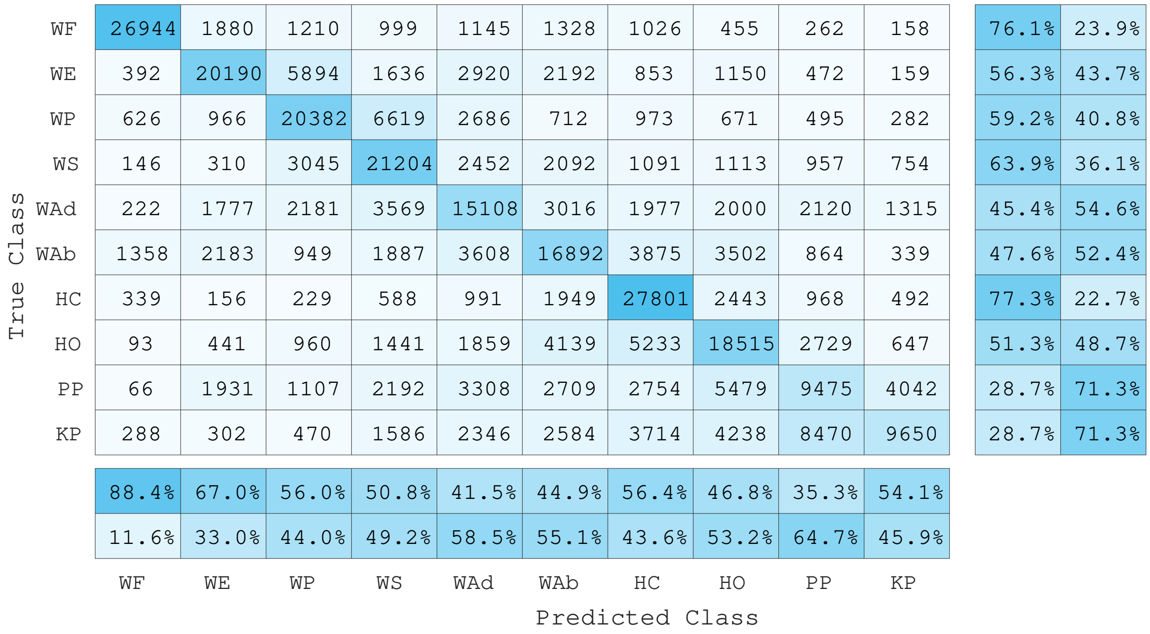
Table 5.7: Classification accuracies of 10 gestures for each cross-validation iteration set using MLP networks classification method. The best classification result for each subject within each cross-validation iteration testing set is shown in bold.

| Cross-Validation Iteration | | Classification Accuracy (%) | |
|---|---|---|---|
| Training Set | Testing Set | EMG | EMG + IMU |
| CVF2, CVF3, CVF4, CVF5 | S10 | 52.99 | **58.06** |
| | S16 | 42.46 | **56.24** |
| | S17 | 53.45 | **65.55** |
| | S18 | 53.70 | **66.14** |
| CVF1, CVF3, CVF4, CVF5 | S13 | 47.93 | **59.84** |
| | S14 | 24.09 | **33.52** |
| | S19 | 44.61 | **57.99** |
| | S22 | 42.37 | **55.39** |
| | S23 | 57.00 | **67.94** |
| CVF1, CVF2, CVF4, CVF5 | S4 | 54.94 | **67.01** |
| | S6 | 30.06 | **36.67** |
| | S12 | 21.31 | **28.60** |
| | S20 | 57.10 | **64.42** |
| | S24 | 39.01 | **46.58** |
| CVF1, CVF2, CVF3, CVF5 | S3 | 23.00 | **30.53** |
| | S5 | 39.48 | **47.96** |
| | S7 | 50.86 | **60.22** |
| | S25 | 53.46 | **64.01** |
| CVF1, CVF2, CVF3, CVF4 | S2 | 50.20 | **54.35** |
| | S9 | 51.31 | **55.09** |
| | S11 | 52.49 | **56.79** |
| | S15 | 37.86 | **42.81** |



| | WF | WE | WP | WS | WAd | WAb | HC | HO | PP | KP | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WF | 25009 | 1685 | 1935 | 1409 | 1714 | 2161 | 775 | 372 | 228 | 119 | 70.6% | 29.4% |
| WE | 472 | 24522 | 1775 | 1941 | 2788 | 2059 | 1034 | 777 | 383 | 107 | 68.4% | 31.6% |
| WP | 930 | 1252 | 12003 | 7560 | 6065 | 1806 | 1784 | 1431 | 1117 | 464 | 34.9% | 65.1% |
| WS | 360 | 520 | 1658 | 11588 | 5949 | 4934 | 2528 | 2563 | 1974 | 1090 | 34.9% | 65.1% |
| WAd | 242 | 1831 | 3067 | 3949 | 13907 | 3240 | 2391 | 1913 | 1740 | 1005 | 41.8% | 58.2% |
| WAb | 1674 | 2124 | 970 | 2699 | 4895 | 16411 | 3764 | 2090 | 547 | 283 | 46.3% | 53.7% |
| HC | 297 | 189 | 428 | 796 | 1273 | 2709 | 26949 | 1853 | 863 | 599 | 74.9% | 25.1% |
| HO | 55 | 507 | 1509 | 1908 | 2434 | 6133 | 6200 | 13793 | 2955 | 563 | 38.3% | 61.7% |
| PP | 78 | 2207 | 1267 | 3466 | 3976 | 3857 | 3406 | 5326 | 6747 | 2733 | 20.4% | 79.6% |
| KP | 234 | 460 | 1015 | 3113 | 4205 | 4176 | 4132 | 4496 | 6642 | 5175 | 15.4% | 84.6% |
| | 85.2% | 69.5% | 46.8% | 30.2% | 29.5% | 34.6% | 50.9% | 39.8% | 29.1% | 42.6% | | |
| | 14.8% | 30.5% | 53.2% | 69.8% | 70.5% | 65.4% | 49.1% | 60.2% | 70.9% | 57.4% | | |

(a)



| | WF | WE | WP | WS | WAd | WAb | HC | HO | PP | KP | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| WF | 26944 | 1880 | 1210 | 999 | 1145 | 1328 | 1026 | 455 | 262 | 158 | 76.1% | 23.9% |
| WE | 392 | 20190 | 5894 | 1636 | 2920 | 2192 | 853 | 1150 | 472 | 159 | 56.3% | 43.7% |
| WP | 626 | 966 | 20382 | 6619 | 2686 | 712 | 973 | 671 | 495 | 282 | 59.2% | 40.8% |
| WS | 146 | 310 | 3045 | 21204 | 2452 | 2092 | 1091 | 1113 | 957 | 754 | 63.9% | 36.1% |
| WAd | 222 | 1777 | 2181 | 3569 | 15108 | 3016 | 1977 | 2000 | 2120 | 1315 | 45.4% | 54.6% |
| WAb | 1358 | 2183 | 949 | 1887 | 3608 | 16892 | 3875 | 3502 | 864 | 339 | 47.6% | 52.4% |
| HC | 339 | 156 | 229 | 588 | 991 | 1949 | 27801 | 2443 | 968 | 492 | 77.3% | 22.7% |
| HO | 93 | 441 | 960 | 1441 | 1859 | 4139 | 5233 | 18515 | 2729 | 647 | 51.3% | 48.7% |
| PP | 66 | 1931 | 1107 | 2192 | 3308 | 2709 | 2754 | 5479 | 9475 | 4042 | 28.7% | 71.3% |
| KP | 288 | 302 | 470 | 1586 | 2346 | 2584 | 3714 | 4238 | 8470 | 9650 | 28.7% | 71.3% |
| | 88.4% | 67.0% | 56.0% | 50.8% | 41.5% | 44.9% | 56.4% | 46.8% | 35.3% | 54.1% | | |
| | 11.6% | 33.0% | 44.0% | 49.2% | 58.5% | 55.1% | 43.6% | 53.2% | 64.7% | 45.9% | | |

(b)

Figure 5.7: MLP network confusion matrix of the 10 gestures collected using EMG data only (a), and EMG and IMU data (b) from all testing subjects. The last two rows from each confusion matrix represent the precision score percentages and the false positive rate percentages (the cumulative number of false positives) of each class, in the top and bottom row, respectively. Similarly, the last two columns represent the recall score percentages and the false negatives rate percentages (the cumulative number of false negatives) of each class, in the left and right column, respectively.

For all of the gestures, the lowest precision score among the twenty-two subjects when using EMG data only was for the Precision Pinch gesture with an overall precision of 29.1%, and the highest precision was for the Wrist Flexion gesture with an overall precision of 85.2%. Similarly, the lowest recall score belonged to the Key Pinch gesture with a value of 15.4%, whereas the highest recall score was again for the Hand Close gesture, with an overall score of 74.9%. On the other hand, the lowest precision score among the twenty-two subjects when using both modalities was for the Precision Pinch gesture with an overall precision of 35.3%, and the highest precision score was for the Wrist Flexion gesture with an overall precision of 88.4%. Further, the lowest recall scores belonged to the Precision Pinch and Key Pinch gestures with a score of 28.7% for both gestures, whereas the highest recall score was for the Hand Close gesture, with an overall score of 77.3%.

Following a similar approach to the previous classification methods, the gesture datasets where further optimized by removing redundant gestures in an attempt to improve the classification accuracy. The next section will discuss the results of said optimization.

### 5.4.2 MLP Networks: 7 Gesture Classification

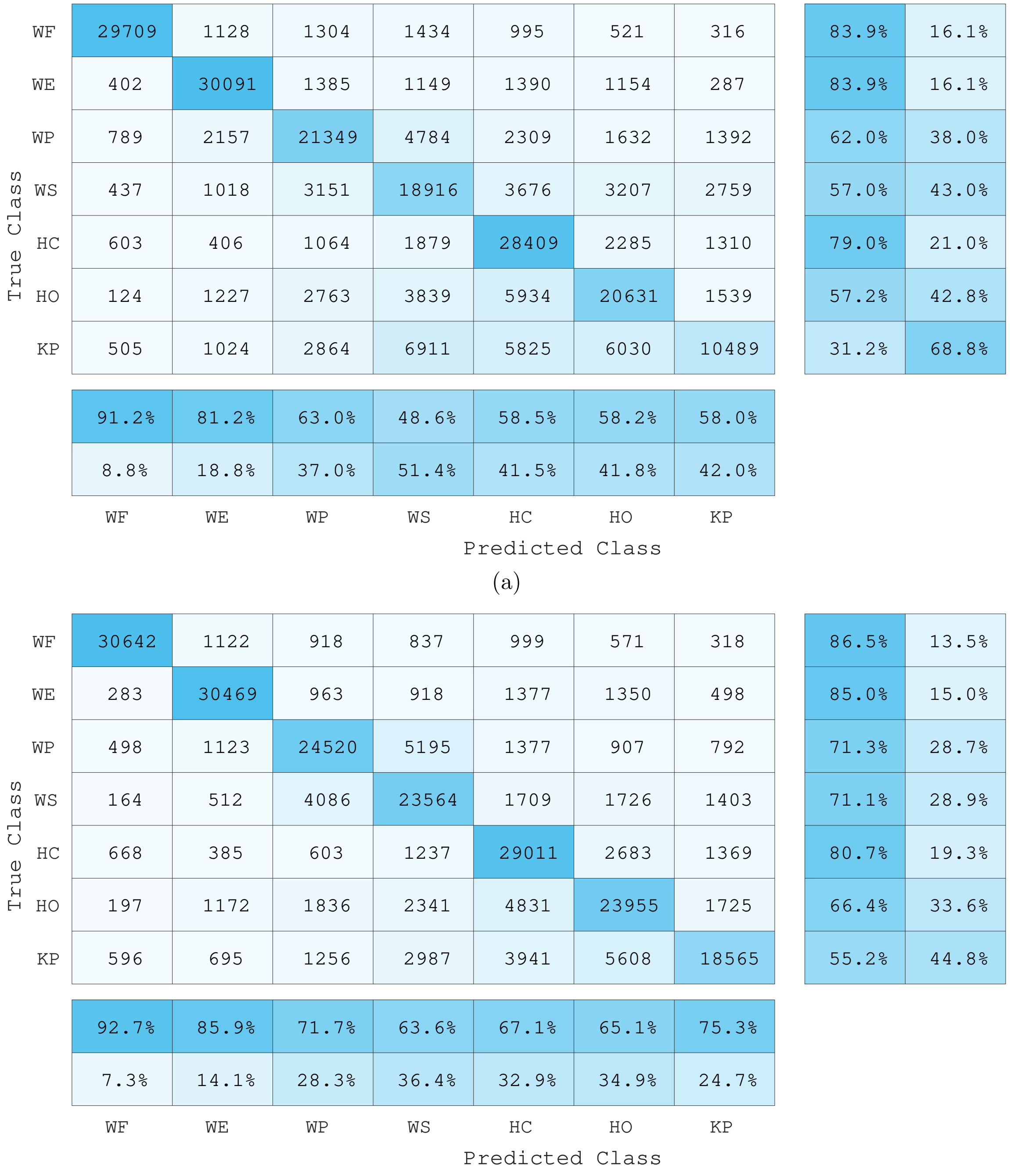
After reducing the number of gestures in the gesture dataset, a new classifier based on MLP networks was created for each subject so that the performance of the classification method could be assessed using the cross-validation iterations testing sets.

The new classification results from each cross-validation iteration using the 7 gesture dataset are presented in Table 5.8. Moreover, Figure 5.8 shows the confusion matrices for the twenty-two testing subjects when classifying the new 7 gestures in a user-independent scenario using EMG data only (Figure 5.8a), and a combination of EMG and IMU data (Figure 5.8b). For EMG data only, classification accuracies ranged from 36.53–78.91% with a mean classification accuracy of 64.82%(±12.83%), whereas for the combined EMG and IMU sensor modality, accuracies ranged from 36.10–88.27% with a mean accuracy of 73.68%(±14.05).

Again, for each gesture, the precision and recall scores were also calculated. The precision and recall scores for each classified gesture using EMG data only, and EMG and IMU data combined, were calculated from the confusion matrix in Figure 5.8a, and in Figure 5.8b, respectively.

Table 5.8: Classification accuracies of 7 gestures for each cross-validation iteration set using MLP networks classification method. The best classification result for each subject within each cross-validation iteration testing set is shown in bold.

| Cross-Validation Iteration | | Classification Accuracy (%) | |
|---|---|---|---|
| Training Set | Testing Set | EMG | EMG + IMU |
| CVF2, CVF3, CVF4, CVF5 | S10 | 68.54 | **73.34** |
| | S16 | 65.63 | **73.14** |
| | S17 | 78.91 | **87.32** |
| | S18 | 74.80 | **82.19** |
| CVF1, CVF3, CVF4, CVF5 | S13 | 70.88 | **80.56** |
| | S14 | 41.93 | **51.99** |
| | S19 | 73.92 | **86.55** |
| | S22 | 64.58 | **77.52** |
| | S23 | 74.67 | **83.88** |
| CVF1, CVF2, CVF4, CVF5 | S4 | 73.68 | **85.31** |
| | S6 | 53.90 | **60.29** |
| | S12 | 36.94 | **46.41** |
| | S20 | 76.46 | **82.11** |
| | S24 | 59.51 | **65.01** |
| CVF1, CVF2, CVF3, CVF5 | S3 | **36.53** | 36.10 |
| | S5 | 66.03 | **80.61** |
| | S7 | 78.00 | **88.27** |
| | S25 | 71.76 | **79.65** |
| CVF1, CVF2, CVF3, CVF4 | S2 | 72.88 | **79.34** |
| | S9 | 66.46 | **77.39** |
| | S11 | 68.05 | **79.72** |
| | S15 | 51.99 | **64.20** |



|  | WF | WE | WP | WS | HC | HO | KP |  |  |
|---|---|---|---|---|---|---|---|---|---|
| WF | 29709 | 1128 | 1304 | 1434 | 995 | 521 | 316 | 83.9% | 16.1% |
| WE | 402 | 30091 | 1385 | 1149 | 1390 | 1154 | 287 | 83.9% | 16.1% |
| WP | 789 | 2157 | 21349 | 4784 | 2309 | 1632 | 1392 | 62.0% | 38.0% |
| WS | 437 | 1018 | 3151 | 18916 | 3676 | 3207 | 2759 | 57.0% | 43.0% |
| HC | 603 | 406 | 1064 | 1879 | 28409 | 2285 | 1310 | 79.0% | 21.0% |
| HO | 124 | 1227 | 2763 | 3839 | 5934 | 20631 | 1539 | 57.2% | 42.8% |
| KP | 505 | 1024 | 2864 | 6911 | 5825 | 6030 | 10489 | 31.2% | 68.8% |
|  | 91.2% | 81.2% | 63.0% | 48.6% | 58.5% | 58.2% | 58.0% |  |  |
|  | 8.8% | 18.8% | 37.0% | 51.4% | 41.5% | 41.8% | 42.0% |  |  |

(a)

|  | WF | WE | WP | WS | HC | HO | KP |  |  |
|---|---|---|---|---|---|---|---|---|---|
| WF | 30642 | 1122 | 918 | 837 | 999 | 571 | 318 | 86.5% | 13.5% |
| WE | 283 | 30469 | 963 | 918 | 1377 | 1350 | 498 | 85.0% | 15.0% |
| WP | 498 | 1123 | 24520 | 5195 | 1377 | 907 | 792 | 71.3% | 28.7% |
| WS | 164 | 512 | 4086 | 23564 | 1709 | 1726 | 1403 | 71.1% | 28.9% |
| HC | 668 | 385 | 603 | 1237 | 29011 | 2683 | 1369 | 80.7% | 19.3% |
| HO | 197 | 1172 | 1836 | 2341 | 4831 | 23955 | 1725 | 66.4% | 33.6% |
| KP | 596 | 695 | 1256 | 2987 | 3941 | 5608 | 18565 | 55.2% | 44.8% |
|  | 92.7% | 85.9% | 71.7% | 63.6% | 67.1% | 65.1% | 75.3% |  |  |
|  | 7.3% | 14.1% | 28.3% | 36.4% | 32.9% | 34.9% | 24.7% |  |  |

(b)

Figure 5.8: MLP networks confusion matrix of the 7 gestures collected using EMG data only (a), and EMG and IMU data (b) from all testing subjects. The last two rows from each confusion matrix represent the precision score percentages and the false positive rate percentages (the cumulative number of false positives) of each class, in the top and bottom row, respectively. Similarly, the last two columns represent the recall score percentages and the false negatives rate percentages (the cumulative number of false negatives) of each class, in the left and right column, respectively.

For all of the gestures, the lowest precision score among the twenty-two subjects when using EMG data only was for the Wrist Supination gesture with an overall precision of 48.6%, and the highest precision was again for the Wrist Flexion gesture with an overall precision of 91.2%. At the same time, the lowest recall score belonged to the Key Pinch gesture with a value of 31.2%, whereas the highest recall scores corresponded to the Wrist Flexion and Wrist Extension gestures, with an overall score of 83.9% for both gestures. On the other hand, the lowest precision score among the twenty-two subjects when using both modalities was for the Wrist Supination gesture with an overall precision of 63.6%, and the highest precision score was for the Wrist Flexion gesture with an overall precision of 92.7%. Further, the lowest recall score belonged to the Key Pinch gesture with a score of 55.2%, whereas the highest recall score was for the Wrist Flexion gesture, with an overall score of 86.5%.

### 5.4.3 MLP Networks: Discussion

The classification performance of the MLP networks improved when using a combination of the EMG and IMU sensor data. In this sense, the mean recognition accuracy when using the two-sensor modality (53.44%) was considerably greater than the mean recognition accuracy obtained when using the EMG data only (44.53%) when classifying 10 gestures. This assumption was upheld by the paired sample $t$ test, which showed that the results were statistically significant ($p < 0.001$). Regarding the 7 gesture dataset, a significant increase ($p < 0.001$) was also observed between the mean recognition accuracy of the combined EMG and IMU data (73.68%) and the mean recognition accuracy of the EMG data only (64.82%).

Regarding the 10 gesture dataset, even though the classification performance improved when using the IMU data, there was little improvement in the precision and recall scores of the Wrist Adduction, Wrist Abduction, and Precision Pinch gestures (Figure 5.7), which indicates that the MLP network failed to generalize properly the unseen data that corresponded to these three classes. This is caused by the redundant muscles governing multiple wrist motions as in the case of the Wrist Extension and the Wrist Adduction gestures [96]. Therefore, by using a reduced number of gestures, an increase of the precision and recall scores can be observed from Figures 5.8a and 5.8b. This indicates that the MLP network was able to overcome the intrinsic variability of the EMG signals among different subjects.

Despite being able to increase the recognition accuracy and keep a consistent classification accuracy trend with new subjects, this approach has several limitations. It is well known that the topology of an ANN is usually determined empirically, so by adopting an approach similar to [77], the performance of the proposed model can be improved. Another limitation is the improper placement of the Myo Armband, which can affect the performance of the classification algorithm, specially in a user-independent scenario. To deal with this issue, a similar approach followed in [111], in which a classifier was trained using the data obtained from the Myo Armband while it was placed in different forearm locations, can be implemented. By doing so, it may be possible to avoid the change of distribution of the data due to displacements of the EMG electrodes. One final limitation of the proposed classification method is that MLP networks are sensible to feature noise, which is attributed to the inability of the users to match the same level of contraction when performing the trained gestures [67]. This will inevitably degrade the ability of the trained network to properly classify the gestures thus, requiring the MLP network to be retrained at some point. Therefore, a self-recalibration algorithm that can track said degradation (*e.g.*, [112]) can be implemented to improve the robustness of the classifier.

Overall, the results showed that it is possible to achieve recognition accuracies of up to 67.94% with an average recognition accuracy of 53.44% among five subjects when classifying 10 gestures using a combination of EMG and IMU features. Furthermore, when classifying the 7 gesture dataset, recognition accuracies increased up to 88.27% with a mean accuracy of 73.68%. This indicates that by using the proposed classification approach, which consists of classifying EMG and IMU data coming from the Myo Armband using MLP networks, it is possible to achieve a user-independent classification.

## 5.5 Comparison of Classification Methods

Following the analysis of the performance of each individual classification-method on the 10 and 7 gesture datasets, a statistical analysis was performed using the Statistical Package for Social Sciences v.25 (SPSS) software in order to identify the best classification method. First, a four-by-two repeated measures ANOVA with a Boneferroni correction Post Hoc test, was performed to identify differences between each classification method. This was done to observe the effects that the classification method (factor), and the sensor modality (level), *i.e.*, EMG data only, and EMG and IMU data, had on the gesture recognition accuracy. This procedure was performed on both the 10 and 7 gesture datasets. Finally, accuracies from the best classification method with the best sensor modality were compared against each other using a four-factor repeated measures ANOVA with a Boneferroni correction Post Hoc test. This section will discuss the significant differences found in the results.

### 5.5.1 Pairwise Comparisons

Tables 5.9 and 5.10 show the statistical analysis results of comparing the four classification models using the EMG, and EMG and IMU data on the 10 and 7 gesture datasets, respectively. A statistical significance was found between all four models ($p \leq 0.003$), which means that the classification performance is not only affected by the sensor modality as shown above, but also by the type of classification approach employed. Furthermore, a significant interaction was observed in the Bilinear Models based classification and the PAC classification method cases ($p < 0.001$), and the effects of this interaction can be seen in Figures 5.9a and 5.9b. In this sense, the Bilinear Models based classification had a lower classification performance than the PAC classification method when using EMG data only. However, when classifying a combination of EMG and IMU data, the Bilinear Model based classification algorithm outperformed the PAC classification method. A similar behaviour was observed in the 7 gesture dataset.

Furthermore, from Tables 5.9 and 5.10, it can be noted that the classification method that performed worst was the PAC, whereas the best classification method was the Adaptive LS-SVM. Interestingly, the classification method that performed second best was the one based on MLP networks, which suggests that the use of the calibration set employed in the PAC, Adaptive LS-SVM, and Bilinear Models classification algorithms was not a critical factor that affected the classification outcomes. Without the calibration set, these methods would not be able to accurately classify data. However, this is not the case of the MLP network classification model, as it was able to generalize to unseen data without transfer of prior knowledge.

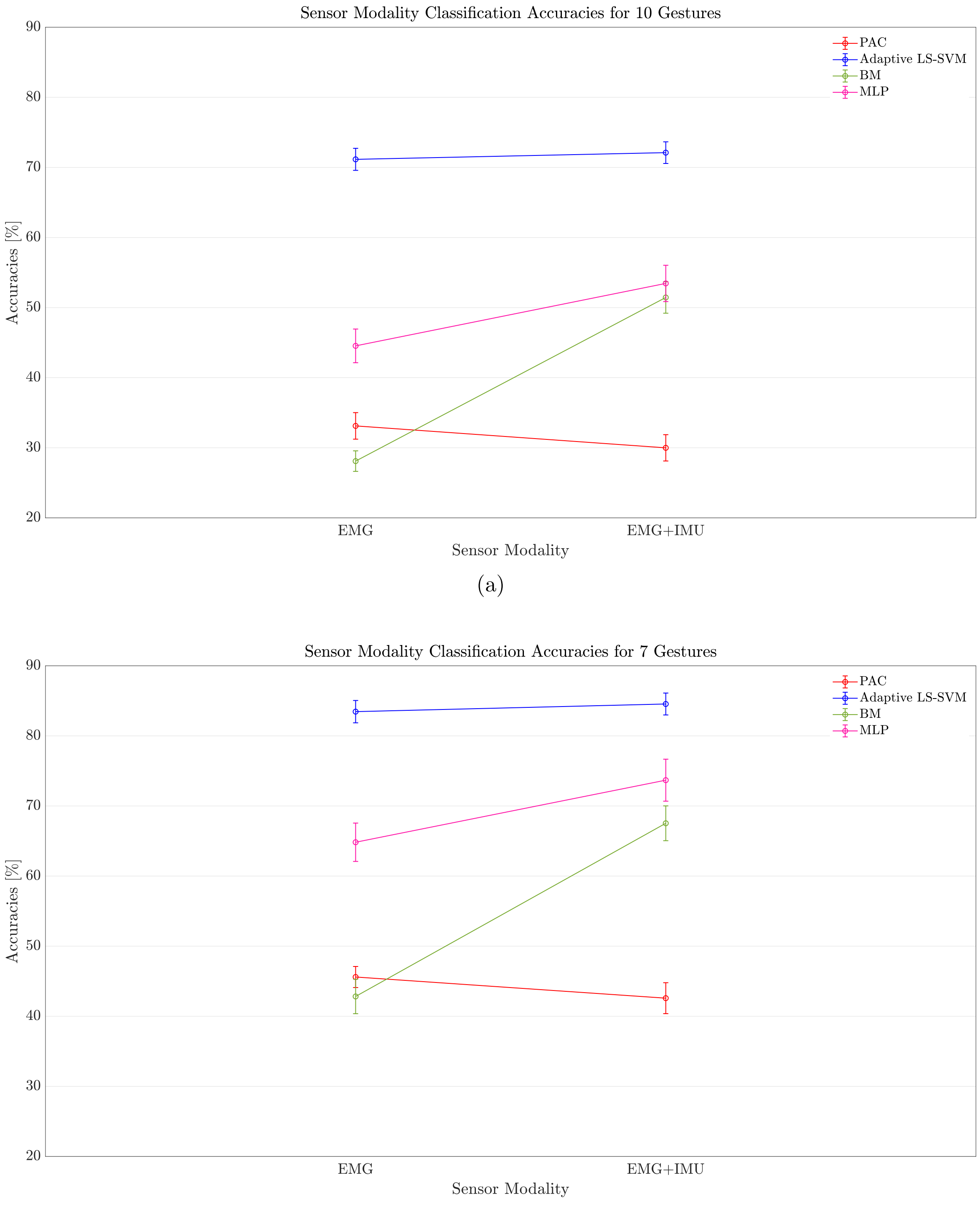
Table 5.9: Pairwise comparison of the different classification methods used with the 10 gesture dataset.

| Classification Method | | Mean Difference (%) | Std. Error (%) | Significance |
|---|---|---|---|---|
| PAC | Adaptive LS-SVM | -40.074 | 1.896 | < 0.001 |
| | Bilinear Models | -8.216 | 1.992 | 0.003 |
| | MLP Networks | -17.437 | 2.598 | < 0.001 |
| Adaptive LS-SVM | PAC | 40.074 | 1.896 | < 0.001 |
| | Bilinear Models | 31.858 | 2.258 | < 0.001 |
| | MLP Networks | 22.636 | 2.941 | < 0.001 |
| Bilinear Models | PAC | 8.216 | 1.992 | 0.003 |
| | Adaptive LS-SVM | -31.858 | 2.258 | < 0.001 |
| | MLP Networks | -9.222 | 1.857 | < 0.001 |
| MLP Networks | PAC | 17.437 | 2.598 | < 0.001 |
| | Adaptive LS-SVM | -22.636 | 2.941 | < 0.001 |
| | Bilinear Models | 9.222 | 1.857 | < 0.001 |

Figures 5.10a and 5.10b show the classification performance on the 10 and 7 gesture datasets when classifying both the EMG, and EMG and IMU data using each classification method. It can be seen that changing the number of gestures did not have an effect on the performance of a specific classification method. More specifically, the trend of each model regarding its classification accuracy did not change, *i.e.*, the classification methods improved their prediction accuracy but the order of the best performing model remained the same. It should also be noted that, while the performance of the MLP networks was similar to the Bilinear Model classification when using data from the EMG and IMU combined for the 10 gesture dataset (mean difference = 2.0029), the difference in performance increased for the 7 gesture dataset when using data from the EMG and IMU combined (mean difference = 6.15).

Table 5.10: Pairwise comparison of the different classification methods used with the 7 gesture dataset.

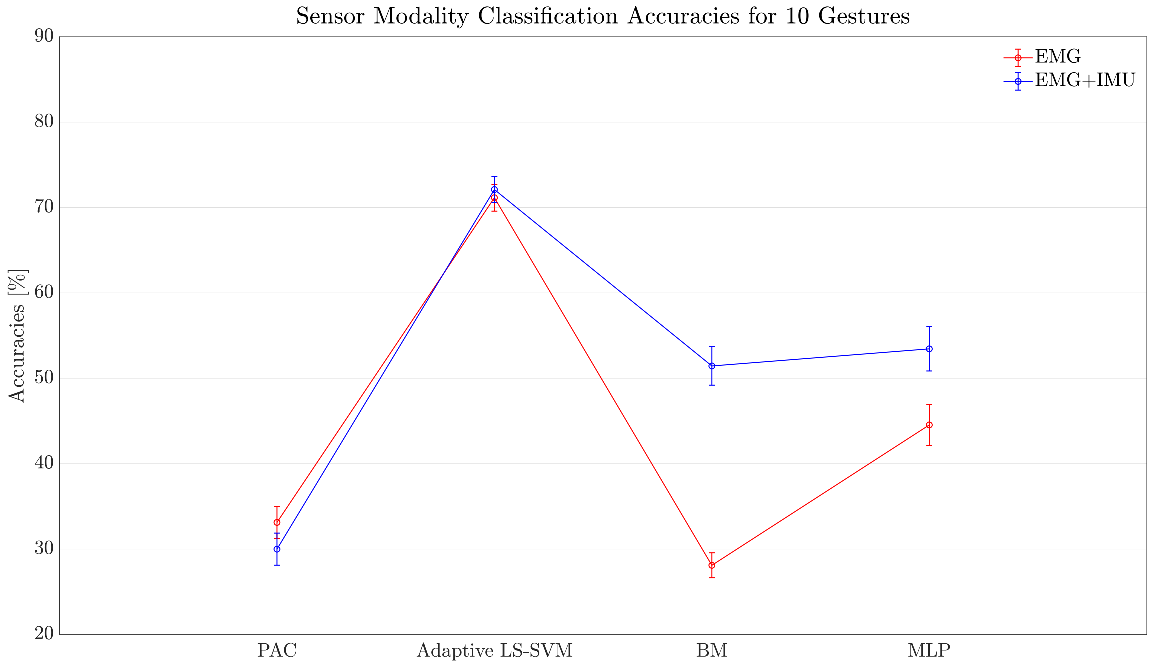
| **Classification Method** | | **Mean Difference (%)** | **Std. Error (%)** | **Significance** |
|---|---|---|---|---|
| PAC | Adaptive LS-SVM | -39.915 | 1.568 | < 0.001 |
| | Bilinear Models | -11.088 | 2.288 | 0.001 |
| | MLP Networks | -25.164 | 2.883 | < 0.001 |
| Adaptive LS-SVM | PAC | 39.915 | 1.568 | < 0.001 |
| | Bilinear Models | 28.827 | 2.508 | < 0.001 |
| | MLP Networks | 14.751 | 3.230 | 0.001 |
| Bilinear Models | PAC | 11.088 | 2.288 | 0.001 |
| | Adaptive LS-SVM | -28.827 | 2.508 | < 0.001 |
| | MLP Networks | -14.076 | 2.077 | < 0.001 |
| MLP Networks | PAC | 25.164 | 2.883 | < 0.001 |
| | Adaptive LS-SVM | -14.751 | 3.230 | 0.001 |
| | Bilinear Models | 14.076 | 2.077 | < 0.001 |



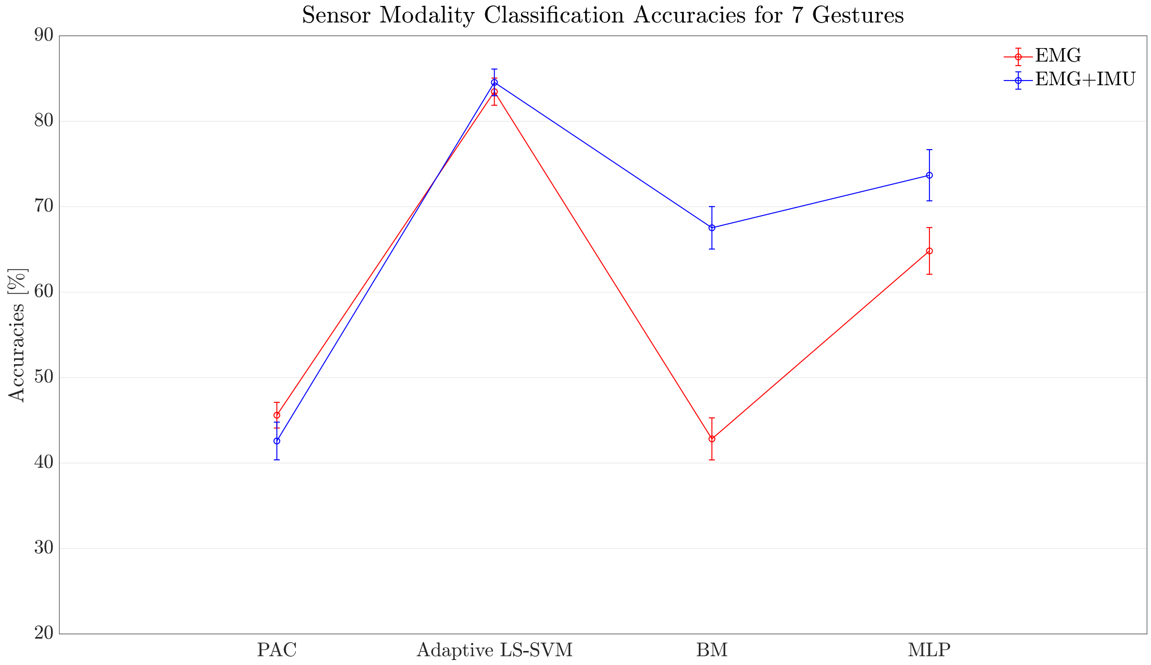
(a)

(b)

Figure 5.9: Interaction between the classification methods and the sensor modality for 10 gestures (a), and 7 gestures (b). Error bars represent the standard error of the mean.



(a)



(b)

Figure 5.10: Overall accuracies of the classification methods using EMG data, and EMG and IMU data for 10 gestures (a), and 7 gestures (b). Error bars represent the standard error of the mean.

### 5.5.2 Best Sensor Modality Pairwise Comparisons

After identifying the sensor modality that improved the performance of each classification method the most, a statistical analysis was performed to compare the ability of each method to classify the 10 and 7 gesture datasets. For the 10 gesture dataset the majority of the pairwise comparisons showed a significant difference ($p < 0.001$), the exception being the MLP networks and the Bilinear Model classification method, which showed no significant differences (Table 5.11). This can be observed better in Figure 5.11, in which the length of the MLP networks and the Bilinear Models bars are almost identical for the 10 gesture dataset. However, after reducing the number of gestures to the 7 gesture dataset, all models showed statistical differences (Table 5.12).

Table 5.11: Pairwise comparison of the different classification methods using the best sensor modality with the 10 gesture dataset.

| **Classification Method** | | **Mean Difference (%)** | **Std. Error (%)** | **Significance** |
|---|---|---|---|---|
| PAC | Adaptive LS-SVM | -38.988 | 1.845 | $<$ 0.001 |
| | Bilinear Models | -18.325 | 2.605 | $<$ 0.001 |
| | MLP Networks | -20.328 | 2.716 | $<$ 0.001 |
| Adaptive LS-SVM | PAC | 38.988 | 1.845 | $<$ 0.001 |
| | Bilinear Models | 20.662 | 2.677 | $<$ 0.001 |
| | MLP Networks | 18.659 | 2.999 | $<$ 0.001 |
| Bilinear Models | PAC | 18.325 | 2.605 | $<$ 0.001 |
| | Adaptive LS-SVM | -20.662 | 2.677 | $<$ 0.001 |
| | MLP Networks | -2.003 | 1.796 | 1.000 |
| MLP Networks | PAC | 20.328 | 2.716 | $<$ 0.001 |
| | Adaptive LS-SVM | -18.659 | 2.999 | $<$ 0.001 |
| | Bilinear Models | 2.003 | 1.796 | 1.000 |

Table 5.12: Pairwise comparison of the different classification methods using the best sensor modality with the 7 gesture dataset.

| **Classification Method** | | **Mean Difference (%)** | **Std. Error (%)** | **Significance** |
|---|---|---|---|---|
| PAC | Adaptive LS-SVM | -38.951 | 1.731 | $< 0.001$ |
| | Bilinear Models | -21.932 | 2.461 | $< 0.001$ |
| | MLP Networks | -28.082 | 2.996 | $< 0.001$ |
| Adaptive LS-SVM | PAC | 38.951 | 1.731 | $< 0.001$ |
| | Bilinear Models | 17.019 | 2.734 | $< 0.001$ |
| | MLP Networks | 10.869 | 3.335 | 0.022 |
| Bilinear Models | PAC | 21.932 | 2.461 | $< 0.001$ |
| | Adaptive LS-SVM | -17.019 | 2.734 | $< 0.001$ |
| | MLP Networks | -6.150 | 1.977 | 0.032 |
| MLP Networks | PAC | 28.082 | 2.996 | $< 0.001$ |
| | Adaptive LS-SVM | -10.869 | 3.335 | 0.022 |
| | Bilinear Models | 6.150 | 1.977 | 0.032 |

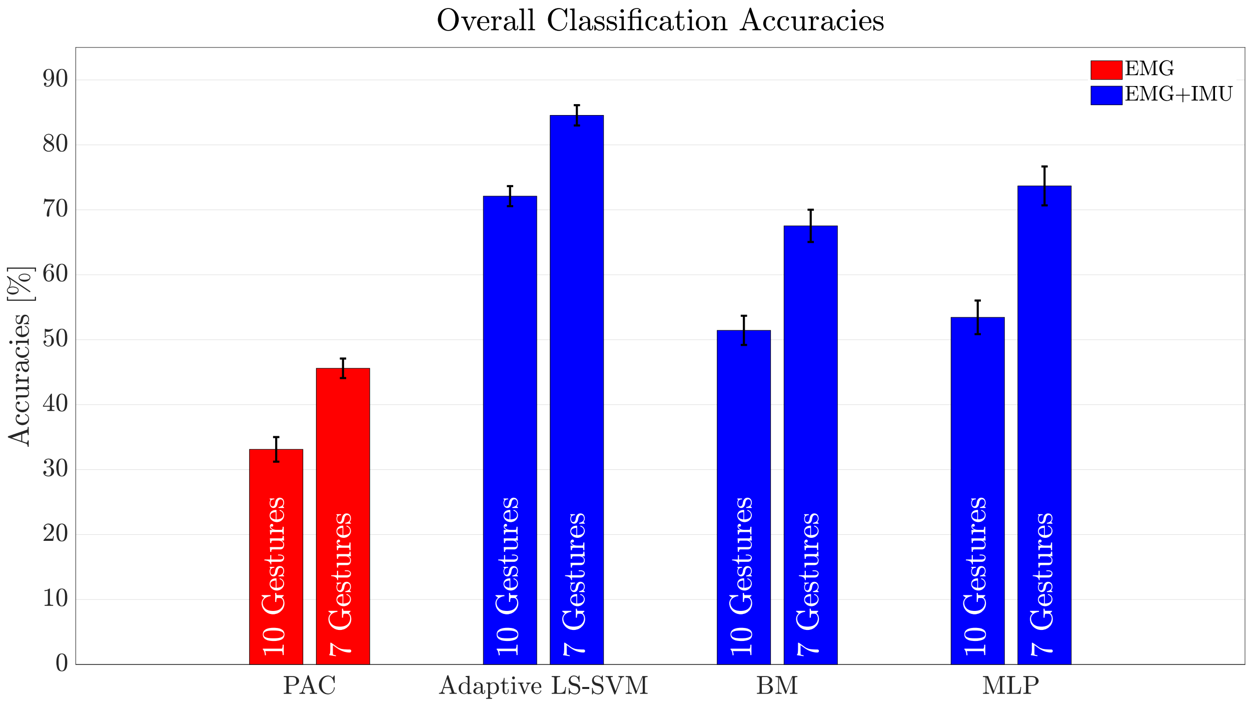


Figure 5.11: Overall accuracies of the classification methods from the best sensor modality. Error bars represent the standard error of the mean.

## 5.6 Conclusion

The results obtained show an improvement on the overall recognition accuracy when using a classifier trained with a combination of EMG and IMU data, instead of EMG data only. The exception was the PAC classification method whose performance degraded, albeit, this degradation was not statistically significant. Previous studies based on the classification of hand gestures using a combination of EMG and IMU data were able to achieve similar accuracies to the proposed model in this thesis. For example, a recognition accuracy of 74.3% was reported in [21] using Hidden Markov Models (HMM) to classify twelve gestures. Likewise, Zhang *et al.* [113] reported a classification accuracy of 90.2% when classifying eighteen gestures to solve a virtual Rubik's cube using a classifier based on HMM and decision trees. However, these studies have two main limitations, the first one is the low number of subjects recruited, which prevents the classification algorithm from generalizing better to a larger population. The second limitation is that the gesture sets presented in those studies were conformed mostly of non-static gestures that can be easily classified with IMU only, as shown in [21]. This study, to the best of the author's knowledge, is the first one to use IMU data to classify a set of static gestures in a user-independent scenario. When classifying static gestures, like the ones in this study, one can expect some crosstalk interference, given the anatomy of the muscles in the forearm as discussed in this chapter. However, by adding the IMU data, this issue can be avoided to a certain extent. Finally, by using a reduced number of gestures to classify, it can be observed that the classification performance of all of the classification methods greatly improved.

# Chapter 6

# Concluding Remarks

The work presented in this thesis was aimed towards developing a user-independent hand gesture recognition classification model using IMU and EMG-based sensor fusion techniques. The purpose of the model was to improve existing user-independent classification models that relied solely on classifying EMG data. To achieve this goal, each existing user-independent classification model was compared against each other to find the best model before and after applying the EMG and IMU sensor fusion techniques. Furthermore, a literature review was performed to identify how hemiparetic stroke patients can benefit from robot-assisted therapy, and also to determine how difficult it is to implement EMG wearable devices in this specific population.

User-independent classification methods were created in an attempt to accelerate the pattern recognition training times for end-users. By doing so, the user's learning process of the control of wearable mechatronic devices would be reduced, thus promoting long term adoption of this technology. However, even though these methods have shown promising results, they are still at an early stage [114]. This study attempted to enhance some popular user-independent classification methods, which included some adaptive learning frameworks, by employing data gathered from all the sensors embedded in the Myo Armband, which is an accessible commercially available device. The standard methods followed by other pattern recognition algorithms were applied. In this sense, EMG and IMU data from healthy subjects, who performed 10 hand and finger gestures under 4 arm positions, were collected and processed, and a set of features was extracted from these data. Then, existing EMG-based user-independent classification models were improved by adding information from the IMU. As a result, accuracies of up to 82.4% were achieved for the best performing model (Adaptive LS-SVM). Moreover, the classification models were further improved by reducing the number of gestures to classify. This caused an overall accuracy increase of 12.48%, 12.44%, 16.08%, and 20.23% for the PAC, Adaptive LS-SVM, Bilinear Models based classification method, and the MLP networks model, respectively.

Additionally, a statistical analysis was performed to compare the effects of adding the IMU data to each of the user-independent classification models. In general, the majority of the tested models improved their classification accuracy significantly, the exception being the PAC model that did not show any signs of improvement. On another note, a second statistical analysis was performed to compare each user-independent classification model against each other after classifying the combined EMG and IMU data. The results showed that the Adaptive LS-SVM classification method outperformed the rest of the tested classification models.

Although this work showed that the classification performance improved on the user-independent classification methods after adding the information collected from another sensor, there is room for further improvement.

## 6.1 Contributions

The contributions of the work presented in this thesis are as follows:

1. A software for collecting data from the Myo Armband was developed. This software allowed the information from different trials to be collected and stored in a database. A friendly and intuitive GUI was developed as part of this software. This GUI gives the user complete control over which gestures to collect, the easiness of collecting data during different arm positions, and the possibility of re-recording motions without needing to start the trials all over again. These features give the developed software an advantage over similar software developed for the Myo Armband.

2. A database of EMG and IMU signals collected from the Myo Armband was developed. These data are composed of signals collected from 22 able-bodied participants performing 10 gestures in 4 different arm positions. Furthermore, because the gestures used to form

the database were recorded in intervals of 1 repetition, the total number of motions in the database is equal to 400 per subject. These data are particularly useful because they allow the flexibility to observe the effects of having more or fewer information from a specific motion while developing user-independent classification algorithms. Furthermore, this database can be used towards the development of more user-independent algorithms based on sensor fusion techniques. It is possible to make this database publicly available online. This would be a major contribution given the fact that similar online databases are composed of EMG signals collected from only 6 gestures [111].

3. A major contribution was the development of a user-independent hand gesture recognition model using MLP networks and sensor fusion techniques. This model has the advantage of not requiring a calibration dataset for new users, which is an improvement over the other methods presented in this work. Such advantage is a step forward towards a zero calibration phase for the ideal user-independent scenario. Moreover, this classification method was the second best in terms of classification performance as shown in Section 5.5.2, Figure 5.11. With a little bit of improvement, this classification method has the potential to enhance the embodiment of wearable devices during robot-assisted therapies.

4. Another contribution is the statistically significant improvement of the classification performance of the adaptive-based user-independent algorithms (the Adaptive LS-SVM and the Bilinear Models based classification method), and the MLP network classification algorithm after combining data collected from the EMG and IMU sensors. This opens up the possibility of exploring the effects of different combinations of sensors on the performance of user-independent classification algorithms.

## 6.2 Limitations and Future Work

While sensor fusion techniques were effectively applied to develop and improve user-independent classification methods, the insight gained from this work indicates that future work can be done to effectively implement these methods during robot-assisted therapies. Some of the future research avenues to explore are highlighted below:

1. *Implement sophisticated feature fusion techniques to improve the classification methods.* In general, three types of fusion levels exist: data-level fusion, decision-level fusion, and feature-level fusion [90]. During data-level fusion, data coming from all sensor modalities are treated as a single dataset from which features are extracted for future classification. On the other hand, decision-level fusion refers to the process of separately classifying features from each sensor modality and then, fusing the outputs of these individual classifiers, using statistical methods such as Bayesian inference, to make a final class decision [90]. In this work, feature-level fusion was implemented. Features extracted from each channel of the EMG were combined with features extracted from the IMU by joining the feature matrices from each modality. Although effective, the main drawback of this technique is that, in most of the cases, it requires the use of feature reduction techniques to find an optimal feature subset. Further research should be directed towards exploring the effects and benefits of these fusion levels on the classification performance of user-independent classification methods.

2. *Real-time performance evaluation.* The user-independent classification methods presented in this work were analyzed offline. This allowed for a performance evaluation using machine learning metrics such as accuracy, precision and recall values. However, future work should focus on online analyses and evaluations. Novak and Riener [114] suggested that offline parameters, such as the signal's optimal window length, may affect the classification outcomes when tested online. Therefore, implementation of new types of metrics, outside the scope of the ones used in traditional machine learning applications, should be considered to asses the efficacy of different sensors used in a user-independent, multisensor modality scenario. It should be noted that the outcomes of this work were never tested on an actual wearable mechatronic device. Therefore, future work should be aimed towards the development of a system that encompass both hardware and software elements. Such system may be able to provide new evaluation metrics when tested under real-time conditions.

3. *Evaluation on a stroke population.* The population used to test the user-independent classification models presented in this work was composed of healthy subjects. However, future work should be aimed towards testing these models with a stroke-patient population. It is

known that stroke patients can present a combination of muscle spasticity and muscle weakness on their paretic limbs [115]. This condition can produce involuntary muscle contractions that can affect the collected EMG and IMU signals, as well as the motion onset and offset of each gesture. Therefore, new parameters, such as entropy measurements [116], may be required to account for involuntary movements that are not unusual on stroke survivors.

4. *Improve the data collection protocol.* The current data collection protocol can be modified to collect data from more gestures and add them to the database generated in this work, and also to collect data from a population of hemiparetic stroke patients. Moreover, the time of the experimental trials could be reduced. This is especially important when collecting data from hemiparetic stroke patients as some of them may be at an early rehabilitation stage. Patients at this stage may present high levels of discomfort when performing motions for prolonged times. On another note, the GUI developed in this work, which was used during the data collecting phase, can benefit from a better means for presenting visual data as feedback. Currently, the EMG signal collected from the Myo Armband is summed across each channel, and then this signal is rectified, before finally streamed in real time to a computer to be used as a means for showing the participant that they are performing an isometric contraction. However, as observed during the experiments, pinch gestures generated signals with small amplitudes that were hard to visualize for some participants.

5. *Train the classifiers using different positions of the Myo Armband.* As discussed in Sections 5.1.3 and 5.3.3, the placement of the sensors of the Myo Armband is greatly influenced by the circumference of the user's forearm. In this sense, the sensors of the Myo Armband cannot be placed on specific muscles because they are restricted to a specific position in the band. This can introduce some muscle crosstalk in the EMG signal, which affects the classification performance of the classification models. Therefore, by training the classifiers using different positions of the Myo Armband, the effect of the crosstalk between muscles could be reduced. Another potential benefit of varying the position of the band during training, is that it could be possible to account for a poor optimal sensor placement that might result from end-users placing the device themselves.

6. *Explore the effects of simultaneously using different user-independent classification methods.* In this work, we used the precision and recall scores to remove gestures from the dataset in order to boost the classification performance. Further, from the results, it was observed that different user-independent classification methods achieved different precision and recall scores for different gestures. This opens up the possibility of having multiple classification methods running in parallel, each one classifying new incoming data simultaneously. Then, by using majority voting techniques, the recognition accuracy of the gestures could potentially increase, and so, the removal of gestures from the dataset would not be necessary to improve the classification performance. Furthermore, because using multiple classification methods in parallel will be computationally expensive, there is room for improvement by implementing sophisticated embedded systems, such as field programmable gate arrays (FPGA), that are capable of handling this computational load. Nowadays, embedded systems have evolved to become powerful portable devices, which is why some studies have focused on implementing them in pattern recognition applications [117, 118].

The purpose of this thesis was to implement user-independent hand gesture recognition classification models using IMU and EMG-based sensor fusion techniques. Existing adaptive classification methods that were aimed towards a user-independent classification model were improved. This was accomplished by combining all of the information collected from the Myo Armband. The main objectives of this work, which were to increase the number of gestures that the Myo Armband can recognize while also improving its detection accuracy in a user-independent scenario, were achieved. In this sense, overall accuracies of up to 84.55% were achieved for a set of 7 gestures. Continued work developing more strategies for using sensor fusion techniques to achieve better user-independent classifications will be able to promote long term adoption of wearable mechatronic devices during robot-assisted therapies.

# References

[1] W. Johnson, O. Onuma, M. Owolabi, and S. Sachdev, “Stroke: a global response is needed,” *Bulletin of the World Health Organization*, vol. 94, no. 9, p. 634, 2016.

[2] S. N. Housley, D. Wu, K. Richards, S. Belagaje, M. Ghovanloo, and A. J. Butler, “Improving upper extremity function and quality of life with a tongue driven exoskeleton: a pilot study quantifying stroke rehabilitation,” *Stroke Research and Treatment*, 2017.

[3] M. E. Stoykov, G. N. Lewis, and D. M. Corcos, “Comparison of bilateral and unilateral training for upper extremity hemiparesis in stroke,” *Neurorehabilitation and Neural Repair*, vol. 23, no. 9, pp. 945–953, 2009.

[4] K. C. Stewart, J. H. Cauraugh, and J. J. Summers, “Bilateral movement training and stroke rehabilitation: a systematic review and meta-analysis,” *Journal of the Neurological Sciences*, vol. 244, no. 1-2, pp. 89–95, 2006.

[5] C. Duret, O. Courtial, A.-G. Grosmaire, and E. Hutin, “Use of a robotic device for the rehabilitation of severe upper limb paresis in subacute stroke: exploration of patient/robot interactions and the motor recovery process,” *BioMed Research International*, vol. 2015, 2015.

[6] M. H. Rahman, M. Saad, J. P. Kenné, and P. S. Archambault, “Nonlinear sliding mode control implementation of an upper limb exoskeleton robot to provide passive rehabilitation therapy,” in *International Conference on Intelligent Robotics and Applications*, (Montreal, QC, Canada), pp. 52–62, Springer, October 3-5, 2012.

[7] H. S. Lo and S. Q. Xie, “Exoskeleton robots for upper-limb rehabilitation: state of the art and future prospects,” *Medical Engineering & Physics*, vol. 34, no. 3, pp. 261–268, 2012.

[8] V. Squeri, A. Basteris, and V. Sanguineti, “Adaptive regulation of assistance ‘as needed’ in robot-assisted motor skill learning and neuro-rehabilitation,” in *IEEE International Conference on Rehabilitation Robotics*, (Zurich, Switzerland), pp. 1–6, June 29-July 1, 2011.

[9] M. Hillman and J. Jepson, “Evaluation of a robotic workstation for the disabled,” *Journal of Biomedical Engineering*, vol. 14, no. 3, pp. 187–192, 1992.

[10] T. Proietti, V. Crocher, A. Roby-Brami, and N. Jarrasse, “Upper-limb robotic exoskeletons for neurorehabilitation: a review on control strategies,” *IEEE Reviews in Biomedical Engineering*, vol. 9, pp. 4–14, 2016.

[11] L. Marchal-Crespo and D. J. Reinkensmeyer, "Review of control strategies for robotic movement training after neurologic injury," *Journal of Neuroengineering and Rehabilitation*, vol. 6, no. 1, p. 20, 2009.

[12] M. Baklouti, J. AbouSaleh, E. Monacelli, and S. Couvet, "Human machine interface in assistive robotics: application to a force controlled upper-limb powered exoskeleton," in *Robotics 2010 Current and Future Challenges*, IntechOpen, 2010.

[13] T. R. Makin, F. de Vignemont, and A. A. Faisal, "Neurocognitive barriers to the embodiment of technology," *Nature Biomedical Engineering*, vol. 1, p. 0014, 2017.

[14] A. Asokan, A. J. Pothen, and R. K. Vijayaraj, "Armatron—a wearable gesture recognition glove: for control of robotic devices in disaster management and human rehabilitation," in *IEEE International Conference on Robotics and Automation for Humanitarian Applications (RAHA)*, (Kollam, India), pp. 1–5, December 18-20, 2016.

[15] M. T. Wolf, C. Assad, M. T. Vernacchia, J. Fromm, and H. L. Jethani, "Gesture-based robot control with variable autonomy from the jpl biosleeve," in *IEEE International Conference on Robotics and Automation*, (Karlsruhe, Germany), pp. 1160–1165, May 6-10, 2013.

[16] Q. Meng, Q. Meng, H. Yu, and X. Wei, "A survey on sEMG control strategies of wearable hand exoskeleton for rehabilitation," in *IEEE 2nd Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*, (Wuhan, China), pp. 165–169, June 16-18, 2017.

[17] "Myo gesture control armband—wearable technology by thalmic labs." Retrieved from: `https://www.myo.com/`.

[18] A.-A. Samadani and D. Kulic, "Hand gesture recognition based on surface electromyography," in *36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, (Chicago, IL, USA), pp. 4196–4199, August 26-30, 2014.

[19] F. Kerber, M. Puhl, and A. Krüger, "User-independent real-time hand gesture recognition based on surface electromyography," in *Proceedings of the 19th International Conference on Human-Computer Interaction with Mobile Devices and Services*, (Vienna, Austria), p. 36, ACM, September 04-07, 2017.

[20] M. E. Benalcázar, C. Motoche, J. A. Zea, A. G. Jaramillo, C. E. Anchundia, P. Zambrano, M. Segura, F. B. Palacios, and M. Pérez, "Real-time hand gesture recognition using the myo armband and muscle activity detection," in *IEEE Second Ecuador Technical Chapters Meeting (ETCM)*, pp. 1–6, 2017.

[21] M. Georgi, C. Amma, and T. Schultz, "Fusion and comparison of IMU and EMG signals for wearable gesture recognition," in *International Joint Conference on Biomedical Engineering Systems and Technologies*, (Lisbon, Portugal), pp. 308–323, Springer, January 12-15, 2015.

[22] S. Jiang, B. Lv, W. Guo, C. Zhang, H. Wang, X. Sheng, and P. B. Shull, "Feasibility of wrist-worn, real-time hand, and surface gesture recognition via sEMG and IMU sensing," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3376–3385, 2017.

[23] Q. Huang, D. Yang, L. Jiang, H. Zhang, H. Liu, and K. Kotani, "A novel unsupervised adaptive learning method for long-term electromyography (EMG) pattern recognition," *Sensors*, vol. 17, no. 6, p. 1370, 2017.

[24] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, p. 44, 2014.

[25] D. Ducharme, L. Costa, L. DiPippo, and L. Hamel, "SVM constraint discovery using KNN applied to the identification of cyberbullying," in *Proceedings of the International Conference on Data Mining (DMIN)*, (Las Vegas, Nevada, USA), pp. 111–117, The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), July 17-20, 2017.

[26] Y. Kijima and S. F. Viegas, "Wrist anatomy and biomechanics," *The Journal of Hand Surgery*, vol. 34, no. 8, pp. 1555–1563, 2009.

[27] D. C. Moore, J. J. Crisco, T. G. Trafton, and E. L. Leventhal, "A digital database of wrist bone anatomy and carpal kinematics," *Journal of Biomechanics*, vol. 40, no. 11, pp. 2537–2542, 2007.

[28] V. Squeri, L. Masia, P. Giannoni, G. Sandini, and P. Morasso, "Wrist rehabilitation in chronic stroke patients by means of adaptive, progressive robot-aided therapy," *Transactions on Neural Systems and Rehabilitation Engineering*, vol. 22, no. 2, pp. 312–325, 2013.

[29] R. L. Linscheid, "Kinematic considerations of the wrist," *Clinical Orthopaedics and Related Research*, no. 202, pp. 27–39, 1986.

[30] D. Bourbonnais and S. V. Noven, "Weakness in patients with hemiparesis," *American Journal of Occupational Therapy*, vol. 43, no. 5, pp. 313–319, 1989.

[31] M. Pekna, M. Pekny, and M. Nilsson, "Modulation of neural plasticity as a basis for stroke rehabilitation," *Stroke*, vol. 43, no. 10, pp. 2819–2828, 2012.

[32] D. C. Boone and S. P. Azen, "Normal range of motion of joints in male subjects," *The Journal of Bone and Joint Surgery*, vol. 61, no. 5, pp. 756–759, 1979.

[33] J. Van Der Lee, H. Beckerman, D. Knol, H. De Vet, and L. Bouter, "Clinimetric properties of the motor activity log for the assessment of arm use in hemiparetic patients," *Stroke*, vol. 35, no. 6, pp. 1410–1414, 2004.

[34] E. Taub and D. M. Morris, "Constraint-induced movement therapy to enhance recovery after stroke," *Current Atherosclerosis Reports*, vol. 3, no. 4, pp. 279–286, 2001.

[35] B. H. Dobkin, "Strategies for stroke rehabilitation," *The Lancet Neurology*, vol. 3, no. 9, pp. 528–536, 2004.

[36] J. Whitall, S. M. Waller, K. H. Silver, and R. F. Macko, "Repetitive bilateral arm training with rhythmic auditory cueing improves motor function in chronic hemiparetic stroke," *Stroke*, vol. 31, no. 10, pp. 2390–2395, 2000.

[37] S. M. Waller and J. Whitall, “Bilateral arm training: why and who benefits?,” *NeuroRehabilitation*, vol. 23, no. 1, pp. 29–41, 2008.

[38] G. Kwakkel, B. J. Kollen, and H. I. Krebs, “Effects of robot-assisted therapy on upper limb recovery after stroke: a systematic review,” *Neurorehabilitation and Neural Repair*, vol. 22, no. 2, pp. 111–121, 2008.

[39] M. J. Johnson, “Recent trends in robot-assisted therapy environments to improve real-life functional performance after stroke,” *Journal of NeuroEngineering and Rehabilitation*, vol. 3, no. 1, p. 29, 2006.

[40] A. Kyrylova, “Development of a wearable mechatronic elbow brace for postoperative motion rehabilitation,” Master’s thesis, The University of Western Ontario, 2015.

[41] A. Basteris, S. M. Nijenhuis, A. H. Stienen, J. H. Buurke, G. B. Prange, and F. Amirabdollahian, “Training modalities in robot-mediated upper limb rehabilitation in stroke: a framework for classification based on a systematic review,” *Journal of Neuroengineering and Rehabilitation*, vol. 11, no. 1, p. 111, 2014.

[42] G. Turchetti, N. Vitiello, S. Romiti, E. Geisler, and S. Micera, “Why effectiveness of robot-mediated neurorehabilitation does not necessarily influence its adoption,” *IEEE Reviews in Biomedical Engineering*, vol. 7, pp. 143–153, 2014.

[43] F. Ryser, T. Bützer, J. P. Held, O. Lambercy, and R. Gassert, “Fully embedded myoelectric control for a wearable robotic hand orthosis,” in *IEEE International Conference on Rehabilitation Robotics*, (London, UK), pp. 615–621, July 17-20, 2017.

[44] Y. Zhang and C. Harrison, “Tomo: wearable, low-cost electrical impedance tomography for hand gesture recognition,” in *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, pp. 167–173, ACM, 2015.

[45] S. Wilson and R. Vaidyanathan, “Upper-limb prosthetic control using wearable multichannel mechanomyography,” in *IEEE International Conference on Rehabilitation Robotics*, (London, UK), pp. 1293–1298, July 17-20, 2017.

[46] G. P. Sadarangani, X. Jiang, L. A. Simpson, J. J. Eng, and C. Menon, “Force myography for monitoring grasping in individuals with stroke with mild to moderate upper-extremity impairments: a preliminary investigation in a controlled environment,” *Frontiers in bioengineering and biotechnology*, vol. 5, p. 42, 2017.

[47] X. Zhang, Y. Li, X. Chen, G. Li, W. Z. Rymer, and P. Zhou, “The effect of involuntary motor activity on myoelectric pattern recognition: a case study with chronic stroke patients,” *Journal of Neural Engineering*, vol. 10, no. 4, p. 046015, 2013.

[48] P.-G. Jung, G. Lim, S. Kim, and K. Kong, “A wearable gesture recognition device for detecting muscular activities based on air-pressure sensors,” *IEEE Transactions on Industrial Informatics*, vol. 11, no. 2, pp. 485–494, 2015.

[49] B. Noronha, S. Dziemian, G. A. Zito, C. Konnaris, and A. A. Faisal, “‘Wink to grasp’—comparing eye, voice & EMG gesture control of grasp with soft-robotic gloves,” in

*IEEE International Conference on Rehabilitation Robotics*, (London, UK), pp. 1043–1048, July 17-20, 2017.

[50] E. Zheng, Q. Wang, and H. Qiao, "A preliminary study of upper-limb motion recognition with noncontact capacitive sensing," in *International Conference on Intelligent Robotics and Applications*, (Wuhan, China), pp. 251–261, Springer, August 16-18, 2017.

[51] N. Haroon and A. N. Malik, "Multiple hand gesture recognition using surface EMG signals," *Journal of Biomedical Engineering and Medical Imaging*, vol. 3, no. 1, p. 1, 2016.

[52] M. F. Wahid, R. Tafreshi, M. Al-Sowaidi, and R. Langari, "Subject-independent hand gesture recognition using normalization and machine learning algorithms," *Journal of Computational Science*, vol. 27, pp. 69–76, 2018.

[53] A. S. Kundu, O. Mazumder, P. K. Lenka, and S. Bhaumik, "Hand gesture recognition based omnidirectional wheelchair control using IMU and EMG sensors," *Journal of Intelligent & Robotic Systems*, vol. 91, no. 3-4, pp. 529–541, 2018.

[54] E. J. Rechy-Ramirez and H. Hu, "Bio-signal based control in assistive robots: a survey," *Digital Communications and Networks*, vol. 1, no. 2, pp. 85–101, 2015.

[55] M. A. Oskoei and H. Hu, "Myoelectric control systems—a survey," *Biomedical Signal Processing and Control*, vol. 2, no. 4, pp. 275–294, 2007.

[56] L. Sörnmo and P. Laguna, *Bioelectrical Signal Processing in Cardiac and Neurological Applications*, vol. 8. Elsevier Academic Press, 2005.

[57] D. Gabriel and G. Kamen, *Essentials of Electromyography.* Champaign, IL: Human Kinetics, 2010.

[58] M. B. I. Reaz, M. Hussain, and F. Mohd-Yasin, "Techniques of EMG signal analysis: detection, processing, classification and applications," *Biological Procedures Online*, vol. 8, no. 1, p. 11, 2006.

[59] J. Wang, L. Tang, and J. E. Bronlund, "Surface EMG signal amplification and filtering," *International Journal of Computer Applications*, vol. 82, no. 1, 2013.

[60] J. Potvin and S. Brown, "Less is more: high pass filtering, to remove up to 99% of the surface EMG signal power, improves EMG-based biceps brachii muscle force estimates," *Journal of Electromyography and Kinesiology*, vol. 14, no. 3, pp. 389–399, 2004.

[61] P. W. Hodges and B. H. Bui, "A comparison of computer-based methods for the determination of onset of muscle contraction using electromyography," *Electroencephalography and Clinical Neurophysiology/Electromyography and Motor Control*, vol. 101, no. 6, pp. 511–519, 1996.

[62] J. Abbnik, A. Van Der Bilt, and H. Van Der Glas, "Detection of onset and termination of muscle activity in surface electromyograms," *Journal of Oral Rehabilitation*, vol. 25, no. 5, pp. 365–369, 1998.

[63] P. Bonato, T. D'Alessio, and M. Knaflitz, "A statistical method for the measurement of muscle activation intervals from surface myoelectric signal during gait," *IEEE Transactions on Biomedical Engineering*, vol. 45, no. 3, pp. 287–299, 1998.

[64] M. Lidierth, "A computer based method for automated measurement of the periods of muscular activity from an EMG and its application to locomotor EMGs," *Electroencephalography and Clinical Neurophysiology*, vol. 64, no. 4, pp. 378–380, 1986.

[65] S. Solnik, P. Rider, K. Steinweg, P. DeVita, and T. Hortobágyi, "Teager–kaiser energy operator signal conditioning improves EMG onset detection," *European Journal of Applied Physiology*, vol. 110, no. 3, pp. 489–498, 2010.

[66] M. Reis, C. Almeida, and R. M. Rocha, "On the performance of surface electromyography-based onset detection methods with real data in assistive technologies," *Multimedia Tools and Applications*, vol. 77, no. 9, pp. 11491–11520, 2018.

[67] B. Hudgins, P. Parker, and R. N. Scott, "A new strategy for multifunction myoelectric control," *IEEE Transactions on Biomedical Engineering*, vol. 40, no. 1, pp. 82–94, 1993.

[68] M. A. Oskoei and H. Hu, "Support vector machine-based classification scheme for myoelectric control applied to upper limb," *IEEE Transactions on Biomedical Engineering*, vol. 55, no. 8, pp. 1956–1965, 2008.

[69] K. Englehart, B. Hudgins, P. A. Parker, and M. Stevenson, "Classification of the myoelectric signal using time-frequency based representations," *Medical Engineering & Physics*, vol. 21, no. 6-7, pp. 431–438, 1999.

[70] A. Phinyomark, P. Phukpattaranont, and C. Limsakul, "Feature reduction and selection for EMG signal classification," *Expert Systems with Applications*, vol. 39, no. 8, pp. 7420–7431, 2012.

[71] K. Englehart, B. Hudgins, *et al.*, "A robust, real-time control scheme for multifunction myoelectric control," *IEEE Transactions on Biomedical Engineering*, vol. 50, no. 7, pp. 848–854, 2003.

[72] Z. Qingju and L. Zhizeng, "Wavelet de-noising of electromyography," in *IEEE International Conference on Mechatronics and Automation*, (Luoyang, Henan, China), pp. 1553–1558, June 25-28, 2006.

[73] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and Their Applications*, vol. 13, no. 4, pp. 18–28, 1998.

[74] F. Amirabdollahian and M. L. Walters, "Application of support vector machines in detecting hand grasp gestures using a commercially off the shelf wireless myoelectric armband," in *IEEE International Conference on Rehabilitation Robotics*, (London, UK), pp. 111–115, July 17-20, 2017.

[75] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Processing Letters*, vol. 9, no. 3, pp. 293–300, 1999.

[76] H. Wang and D. Hu, “Comparison of SVM and LS-SVM for regression,” in *IEEE International Conference on Neural Networks and Brain*, vol. 1, (Beijing, China), pp. 279–283, IEEE, October 13-15, 2005.

[77] A. A. M. Lima, R. M. Araujo, F. A. G. dos Santos, V. H. Yoshizumi, F. K. de Barros, D. H. Spatti, L. H. Liboni, and M. E. Dajer, “Classification of hand movements from EMG signals using optimized MLP,” in *IEEE International Joint Conference on Neural Networks (IJCNN)*, (Rio de Janeiro, Brazil), pp. 1–7, July 8-13, 2018.

[78] C. M. Bishop, *Pattern recognition and machine learning*. Springer, 2006.

[79] B. Khanna, S. Moses, and M. Nirmala, “Softmax based user attitude detection algorithm for sentimental analysis,” *Procedia Computer Science*, vol. 125, pp. 313–320, 2018.

[80] S. S. Haykin, *Neural networks: a comprehensive foundation*. Upper Saddle River, N.J: Prentice Hall, 2nd ed., 1999.

[81] T. Tommasi, F. Orabona, C. Castellini, and B. Caputo, “Improving control of dexterous hand prostheses using adaptive learning,” *IEEE Transactions on Robotics*, vol. 29, no. 1, pp. 207–219, 2013.

[82] T. Matsubara and J. Morimoto, “Bilinear modeling of EMG signals to extract user-independent features for multiuser myoelectric interface,” *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 8, pp. 2205–2213, 2013.

[83] J. B. Tenenbaum and W. T. Freeman, “Separating style and content with bilinear models,” *Neural Computation*, vol. 12, no. 6, pp. 1247–1283, 2000.

[84] P. Visconti, F. Gaetani, G. Zappatore, and P. Primiceri, “Technical features and functionalities of myo armband: an overview on related literature and advanced applications of myoelectric armbands mainly focused on arm prostheses,” *International Journal on Smart Sensing and Intelligent Systems*, vol. 11, no. 1, pp. 1–25, 2018.

[85] M. Tomaszewski, “Myo SDK matlab MEX wrapper,” 2016. Retrieved from: `https://github.com/mark-toma/MyoMex`.

[86] V. Montoya-Leal, A. Orozco-Duque, J. Ugarte, M. Portela, J. Franco, and V. Perez, “Assessment protocol of wrist flexion and extension to support processes in occupational health using myo armband,” in *VII Latin American Congress on Biomedical Engineering CLAIB*, pp. 585–588, Springer, 2017.

[87] A. Fougner, E. Scheme, A. D. Chan, K. Englehart, and Ø. Stavdahl, “Resolving the limb position effect in myoelectric pattern recognition,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 19, no. 6, pp. 644–651, 2011.

[88] B. Knorr, R. Hughes, D. Sherrill, J. Stein, M. Akay, and P. Bonato, “Quantitative measures of functional upper limb movement in persons after stroke,” in *IEEE 2nd International EMBS Conference on Neural Engineering*, (Arlington, VA, USA), pp. 252–255, March 16-19, 2005.

[89] J. Drapała, K. Brzostowski, A. Szpala, and A. Rutkowska-Kucharska, “Two stage EMG onset detection method,” *Archives of Control Sciences*, vol. 22, no. 4, pp. 427–440, 2012.

[90] R. Gravina, P. Alinia, H. Ghasemzadeh, and G. Fortino, "Multi-sensor fusion in body sensor networks: State-of-the-art and research challenges," *Information Fusion*, vol. 35, pp. 68–80, 2017.

[91] H.-S. Park and C.-H. Jun, "A simple and fast algorithm for k-medoids clustering," *Expert Systems with Applications*, vol. 36, no. 2, pp. 3336–3341, 2009.

[92] B. Schölkopf, "The kernel trick for distances," in *Advances in Neural Information Processing Systems*, pp. 301–307, 2001.

[93] J. M. Phillips and S. Venkatasubramanian, "A gentle introduction to the kernel distance," *arXiv preprint arXiv:1103.1625*, 2011.

[94] S. Raschka, "Model evaluation, model selection, and algorithm selection in machine learning," *arXiv preprint arXiv:1811.12808*, 2018.

[95] J. G. Colli-Alfaro, A. Ibrahim, and A. L. Trejos, "Design of user-independent hand gesture recognition using multilayer perceptron networks and sensor fusion techniques," in *IEEE 16th International Conference on Rehabilitation Robotics*, (Toronto, Ontario, Canada), pp. 1103–1108, June 24-28, 2019.

[96] B. P. Livingston, R. L. Segal, A. Song, K. Hopkins, A. W. English, and C. C. Manning, "Functional activation of the extensor carpi radialis muscles in humans," *Archives of Physical Medicine and Rehabilitation*, vol. 82, no. 9, pp. 1164–1170, 2001.

[97] G. H. Golub and C. F. Van Loan, *Matrix computations*. The Jhons Hopkins University Press, 4th ed., 2013.

[98] G. C. Cawley, "Leave-one-out cross-validation based model selection criteria for weighted LS-SVMs," in *IEEE International Joint Conference on Neural Network Proceedings*, (Vancouver, BC, Canada), pp. 1661–1668, July 16-21, 2006.

[99] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from `https://www.tensorflow.org/`.

[100] Python Software Foundation, "Python language reference." Version 3.6. Available at `http://www.python.org`.

[101] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[102] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[103] A. Zell, G. Mamier, M. Vogt, N. Mache, R. Hübner, S. Döring, K.-U. Herrmann, T. Soyez, M. Schmalzl, T. Sommer, *et al.*, "SNNS: Stuttgart neural network simulator," *Institute for Parallel and Distributed High Performance Systems, Technical Report*, vol. 95, no. 6, 1998. User Manual, Version 4.2.

[104] RStudio Team, "RStudio: integrated development environment for R," 2015. Version 1.1.463 Available at `http://www.rstudio.com/`.

[105] C. Bergmeir and J. M. Benítez, "Neural networks in R using the stuttgart neural network simulator: RSNNS," 2018. Retrieved from: `https://github.com/cbergmeir/RSNNS`.

[106] P. Mehta, M. Bukov, C.-H. Wang, A. G. Day, C. Richardson, C. K. Fisher, and D. J. Schwab, "A high-bias, low-variance introduction to machine learning for physicists," *Physics Reports*, 2019.

[107] R. Chowdhury, M. Reaz, M. Ali, A. Bakar, K. Chellappan, and T. Chang, "Surface electromyography signal processing and classification techniques," *Sensors*, vol. 13, no. 9, pp. 12431–12466, 2013.

[108] A. J. Young, L. J. Hargrove, and T. A. Kuiken, "The effects of electrode size and orientation on the sensitivity of myoelectric pattern recognition systems to electrode shift," *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 9, pp. 2537–2544, 2011.

[109] A. Ishii, T. Kondo, and S. Yano, "Improvement of EMG pattern recognition by eliminating posture-dependent components," in *International Conference on Intelligent Autonomous Systems*, (Shanghai, China), pp. 19–30, Springer, July 3-7, 2016.

[110] Y. Zhang, Y. Chen, H. Yu, X. Yang, W. Lu, and H. Liu, "Wearing-independent hand gesture recognition method based on EMG armband," *Personal and Ubiquitous Computing*, vol. 22, no. 3, pp. 511–524, 2018.

[111] U. C. Allard, F. Nougarou, C. L. Fall, P. Giguère, C. Gosselin, F. Laviolette, and B. Gosselin, "A convolutional neural network for robotic arm guidance using sEMG based frequency-features," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, (Daejeon, South Korea), pp. 2464–2470, October 9-14, 2016.

[112] X. Zhai, B. Jelfs, R. H. Chan, and C. Tin, "Self-recalibrating surface EMG pattern recognition for neuroprosthesis control based on convolutional neural network," *Frontiers in Neuroscience*, vol. 11, p. 379, 2017.

[113] X. Zhang, X. Chen, Y. Li, V. Lantz, K. Wang, and J. Yang, "A framework for hand gesture recognition based on accelerometer and EMG sensors," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 41, no. 6, pp. 1064–1076, 2011.

[114] D. Novak and R. Riener, "A survey of sensor fusion methods in wearable robotics," *Robotics and Autonomous Systems*, vol. 73, pp. 155–170, 2015.

[115] X. Hu, K. Tong, X. Wei, W. Rong, E. Susanto, and S. Ho, "The effects of post-stroke upper-limb training with an electromyography (EMG)-driven hand robot," *Journal of Electromyography and Kinesiology*, vol. 23, no. 5, pp. 1065–1074, 2013.

[116] X. Zhang and P. Zhou, “High-density myoelectric pattern recognition toward improved stroke rehabilitation,” *IEEE Transactions on Biomedical Engineering*, vol. 59, no. 6, pp. 1649–1657, 2012.

[117] H. Wöhrle, M. Tabie, S. Kim, F. Kirchner, and E. Kirchner, “A hybrid FPGA-based system for EEG-and EMG-based online movement prediction,” *Sensors*, vol. 17, no. 7, p. 1552, 2017.

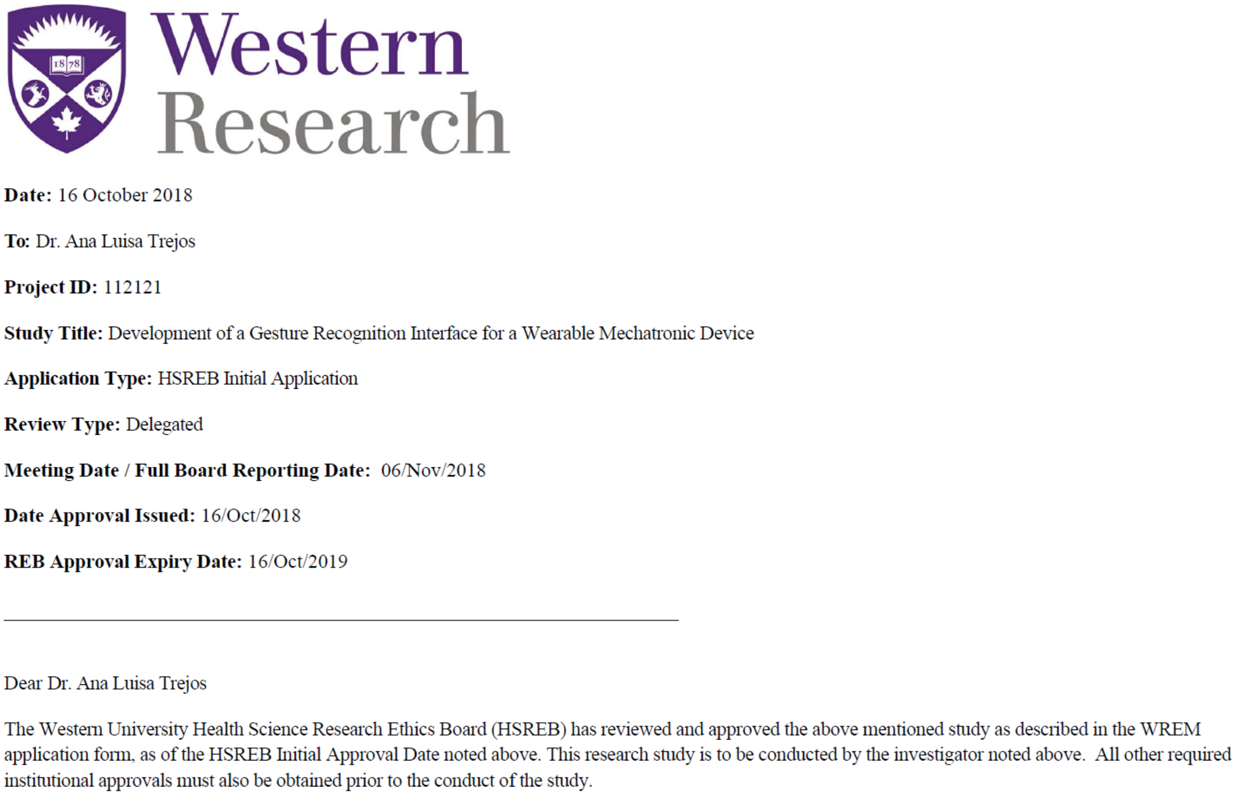
[118] D. De Venuto and G. Mezzina, “Neuromuscular disorders assessment by FPGA-based SVM classification of synchronized EEG/EMG,” in *International Conference on Applications in Electronics Pervading Industry, Environment and Society*, (Pisa, Italy), pp. 37–44, Springer, September 26-27, 2018.

[119] A. D. Chan and G. C. Green, “Myoelectric control development toolbox,” *CMBES Proceedings*, vol. 30, 2007.

# Appendix A

# Permissions and Approvals

## A.1 Ethics Approval



Western Research

**Date:** 16 October 2018

**To:** Dr. Ana Luisa Trejos

**Project ID:** 112121

**Study Title:** Development of a Gesture Recognition Interface for a Wearable Mechatronic Device

**Application Type:** HSREB Initial Application

**Review Type:** Delegated

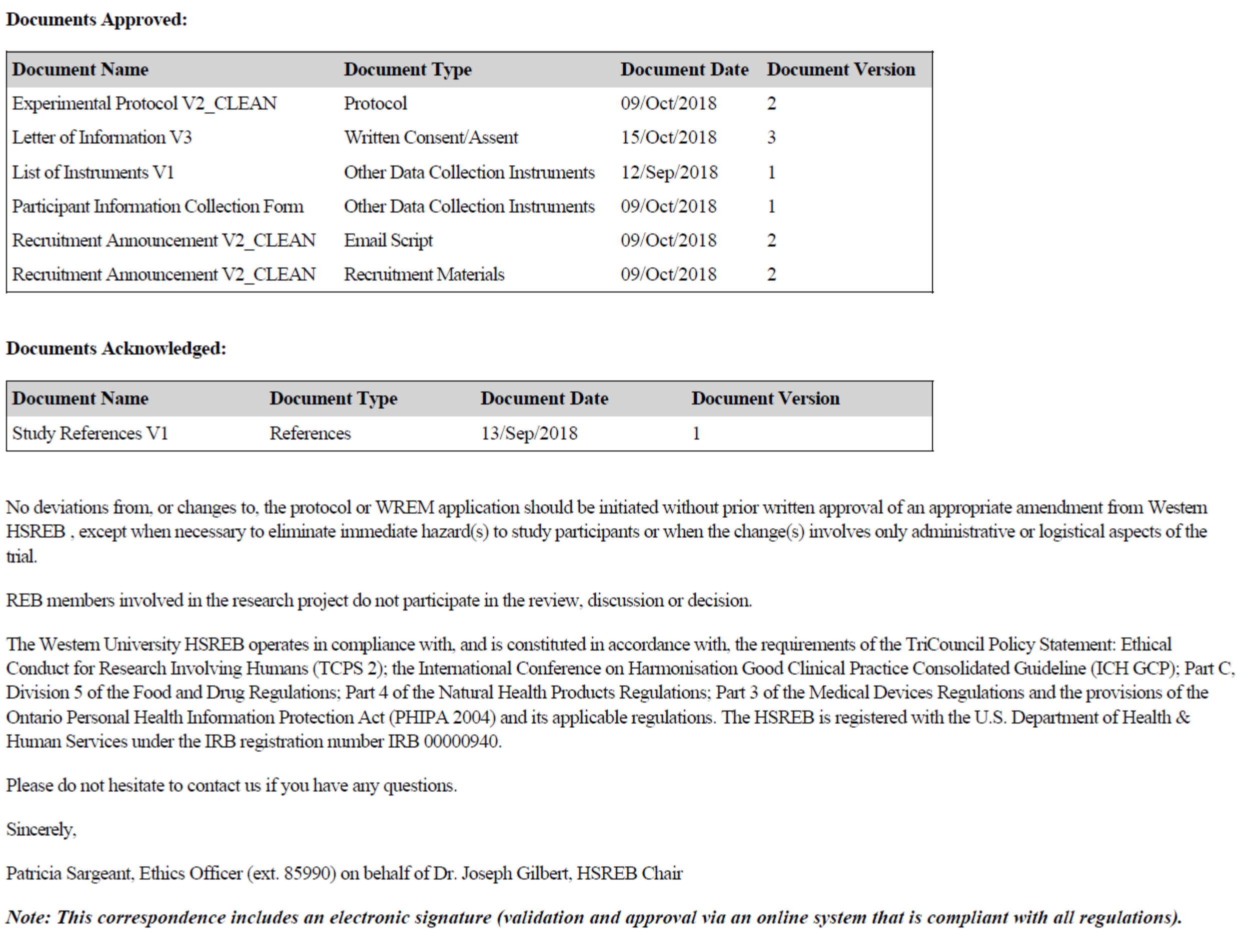
**Meeting Date / Full Board Reporting Date:** 06/Nov/2018

**Date Approval Issued:** 16/Oct/2018

**REB Approval Expiry Date:** 16/Oct/2019

Dear Dr. Ana Luisa Trejos

The Western University Health Science Research Ethics Board (HSREB) has reviewed and approved the above mentioned study as described in the WREM application form, as of the HSREB Initial Approval Date noted above. This research study is to be conducted by the investigator noted above. All other required institutional approvals must also be obtained prior to the conduct of the study.



**Documents Approved:**

| Document Name | Document Type | Document Date | Document Version |
|---|---|---|---|
| Experimental Protocol V2_CLEAN | Protocol | 09/Oct/2018 | 2 |
| Letter of Information V3 | Written Consent/Assent | 15/Oct/2018 | 3 |
| List of Instruments V1 | Other Data Collection Instruments | 12/Sep/2018 | 1 |
| Participant Information Collection Form | Other Data Collection Instruments | 09/Oct/2018 | 1 |
| Recruitment Announcement V2_CLEAN | Email Script | 09/Oct/2018 | 2 |
| Recruitment Announcement V2_CLEAN | Recruitment Materials | 09/Oct/2018 | 2 |

**Documents Acknowledged:**

| Document Name | Document Type | Document Date | Document Version |
|---|---|---|---|
| Study References V1 | References | 13/Sep/2018 | 1 |

No deviations from, or changes to, the protocol or WREM application should be initiated without prior written approval of an appropriate amendment from Western HSREB , except when necessary to eliminate immediate hazard(s) to study participants or when the change(s) involves only administrative or logistical aspects of the trial.

REB members involved in the research project do not participate in the review, discussion or decision.

The Western University HSREB operates in compliance with, and is constituted in accordance with, the requirements of the TriCouncil Policy Statement: Ethical Conduct for Research Involving Humans (TCPS 2); the International Conference on Harmonisation Good Clinical Practice Consolidated Guideline (ICH GCP); Part C, Division 5 of the Food and Drug Regulations; Part 4 of the Natural Health Products Regulations; Part 3 of the Medical Devices Regulations and the provisions of the Ontario Personal Health Information Protection Act (PHIPA 2004) and its applicable regulations. The HSREB is registered with the U.S. Department of Health & Human Services under the IRB registration number IRB 00000940.

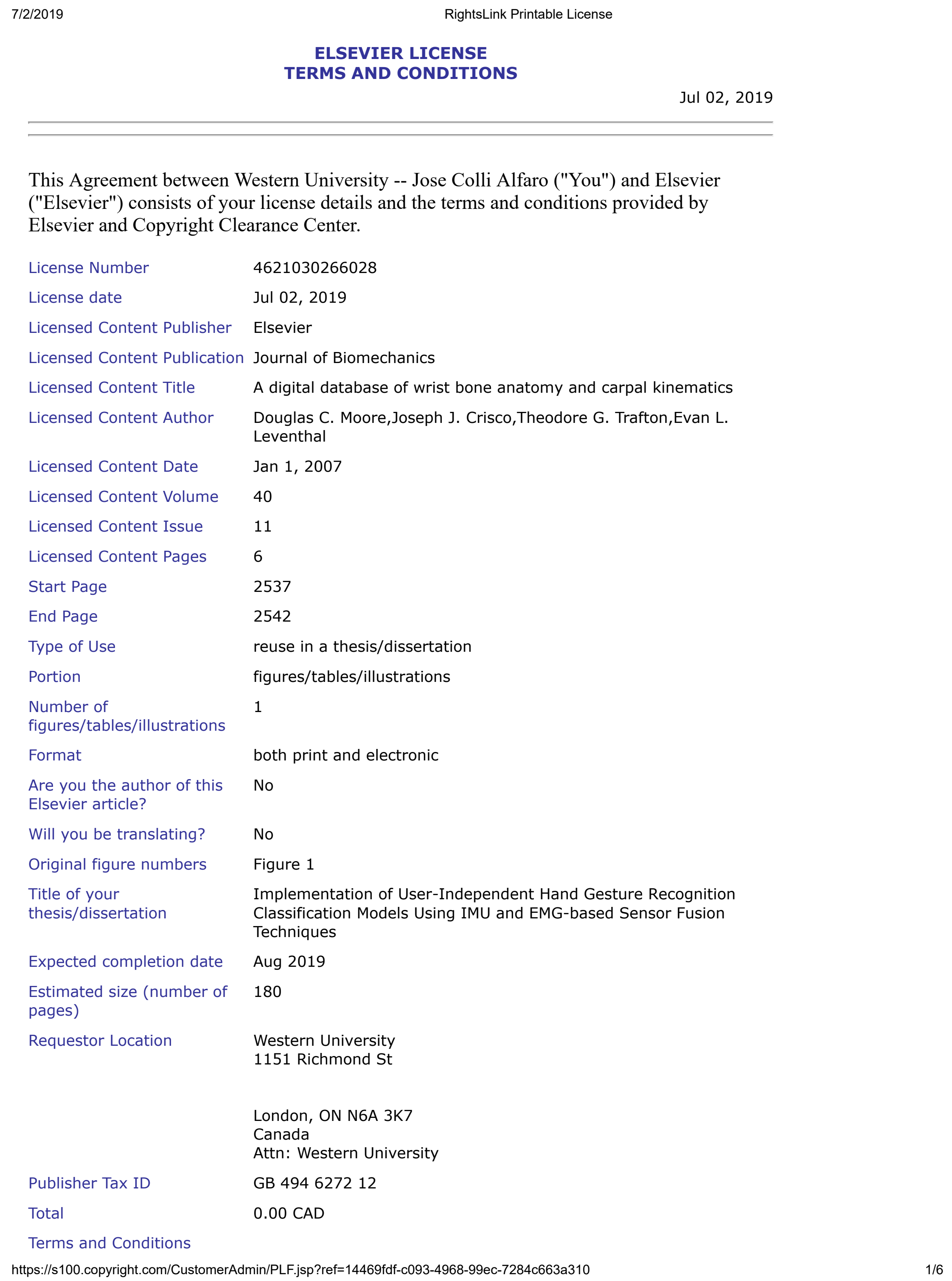
Please do not hesitate to contact us if you have any questions.

Sincerely,

Patricia Sargeant, Ethics Officer (ext. 85990) on behalf of Dr. Joseph Gilbert, HSREB Chair

***Note: This correspondence includes an electronic signature (validation and approval via an online system that is compliant with all regulations).***

## A.2 Permission for Figure 2.1



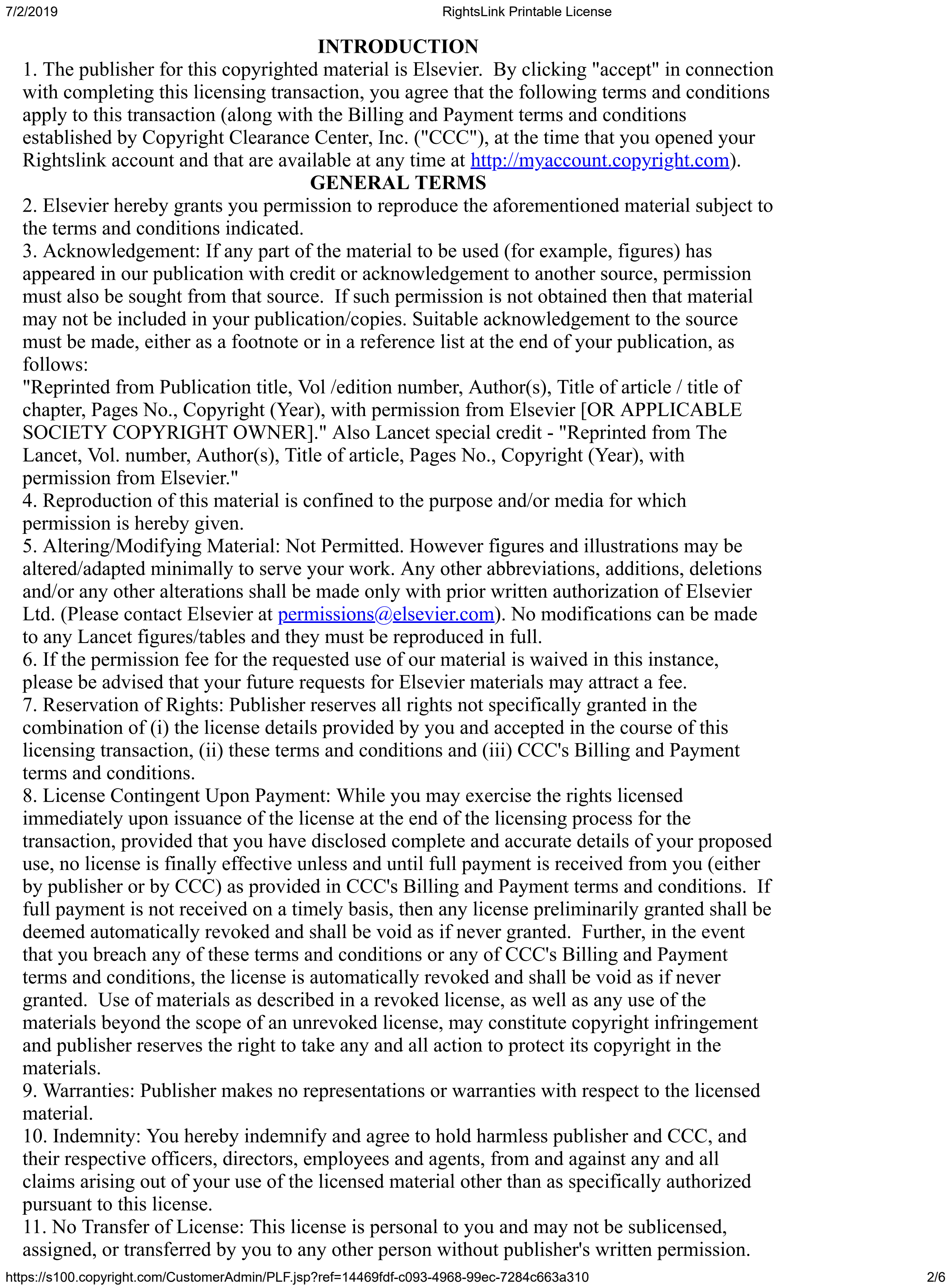
### ELSEVIER LICENSE
### TERMS AND CONDITIONS

Jul 02, 2019

This Agreement between Western University -- Jose Colli Alfaro ("You") and Elsevier ("Elsevier") consists of your license details and the terms and conditions provided by Elsevier and Copyright Clearance Center.

| | |
|---|---|
| License Number | 4621030266028 |
| License date | Jul 02, 2019 |
| Licensed Content Publisher | Elsevier |
| Licensed Content Publication | Journal of Biomechanics |
| Licensed Content Title | A digital database of wrist bone anatomy and carpal kinematics |
| Licensed Content Author | Douglas C. Moore,Joseph J. Crisco,Theodore G. Trafton,Evan L. Leventhal |
| Licensed Content Date | Jan 1, 2007 |
| Licensed Content Volume | 40 |
| Licensed Content Issue | 11 |
| Licensed Content Pages | 6 |
| Start Page | 2537 |
| End Page | 2542 |
| Type of Use | reuse in a thesis/dissertation |
| Portion | figures/tables/illustrations |
| Number of figures/tables/illustrations | 1 |
| Format | both print and electronic |
| Are you the author of this Elsevier article? | No |
| Will you be translating? | No |
| Original figure numbers | Figure 1 |
| Title of your thesis/dissertation | Implementation of User-Independent Hand Gesture Recognition Classification Models Using IMU and EMG-based Sensor Fusion Techniques |
| Expected completion date | Aug 2019 |
| Estimated size (number of pages) | 180 |
| Requestor Location | Western University<br>1151 Richmond St<br><br>London, ON N6A 3K7<br>Canada<br>Attn: Western University |
| Publisher Tax ID | GB 494 6272 12 |
| Total | 0.00 CAD |

Terms and Conditions

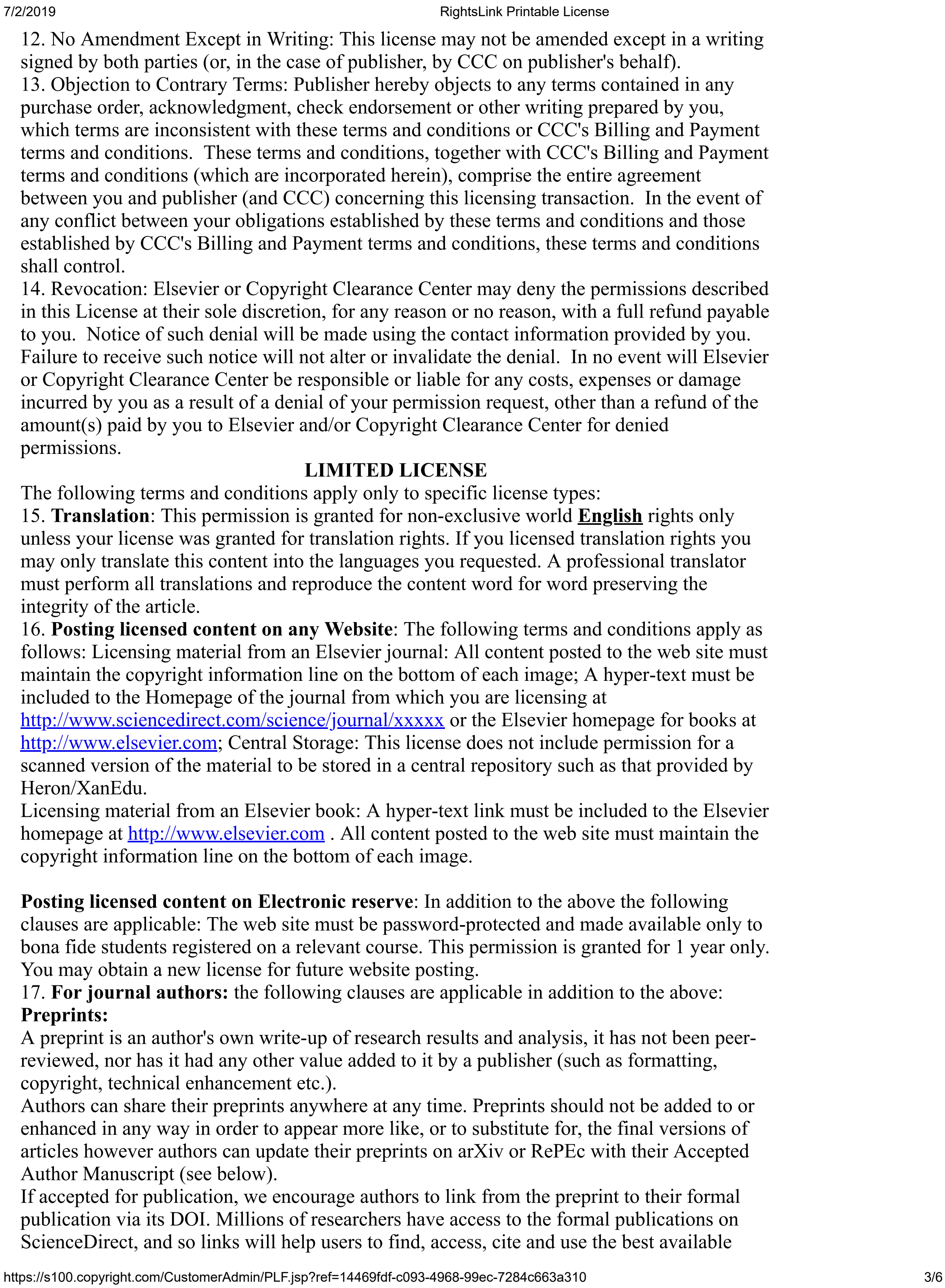


**INTRODUCTION**

1. The publisher for this copyrighted material is Elsevier. By clicking "accept" in connection with completing this licensing transaction, you agree that the following terms and conditions apply to this transaction (along with the Billing and Payment terms and conditions established by Copyright Clearance Center, Inc. ("CCC"), at the time that you opened your Rightslink account and that are available at any time at http://myaccount.copyright.com).

**GENERAL TERMS**

2. Elsevier hereby grants you permission to reproduce the aforementioned material subject to the terms and conditions indicated.
3. Acknowledgement: If any part of the material to be used (for example, figures) has appeared in our publication with credit or acknowledgement to another source, permission must also be sought from that source. If such permission is not obtained then that material may not be included in your publication/copies. Suitable acknowledgement to the source must be made, either as a footnote or in a reference list at the end of your publication, as follows:
"Reprinted from Publication title, Vol /edition number, Author(s), Title of article / title of chapter, Pages No., Copyright (Year), with permission from Elsevier [OR APPLICABLE SOCIETY COPYRIGHT OWNER]." Also Lancet special credit - "Reprinted from The Lancet, Vol. number, Author(s), Title of article, Pages No., Copyright (Year), with permission from Elsevier."
4. Reproduction of this material is confined to the purpose and/or media for which permission is hereby given.
5. Altering/Modifying Material: Not Permitted. However figures and illustrations may be altered/adapted minimally to serve your work. Any other abbreviations, additions, deletions and/or any other alterations shall be made only with prior written authorization of Elsevier Ltd. (Please contact Elsevier at permissions@elsevier.com). No modifications can be made to any Lancet figures/tables and they must be reproduced in full.
6. If the permission fee for the requested use of our material is waived in this instance, please be advised that your future requests for Elsevier materials may attract a fee.
7. Reservation of Rights: Publisher reserves all rights not specifically granted in the combination of (i) the license details provided by you and accepted in the course of this licensing transaction, (ii) these terms and conditions and (iii) CCC's Billing and Payment terms and conditions.
8. License Contingent Upon Payment: While you may exercise the rights licensed immediately upon issuance of the license at the end of the licensing process for the transaction, provided that you have disclosed complete and accurate details of your proposed use, no license is finally effective unless and until full payment is received from you (either by publisher or by CCC) as provided in CCC's Billing and Payment terms and conditions. If full payment is not received on a timely basis, then any license preliminarily granted shall be deemed automatically revoked and shall be void as if never granted. Further, in the event that you breach any of these terms and conditions or any of CCC's Billing and Payment terms and conditions, the license is automatically revoked and shall be void as if never granted. Use of materials as described in a revoked license, as well as any use of the materials beyond the scope of an unrevoked license, may constitute copyright infringement and publisher reserves the right to take any and all action to protect its copyright in the materials.
9. Warranties: Publisher makes no representations or warranties with respect to the licensed material.
10. Indemnity: You hereby indemnify and agree to hold harmless publisher and CCC, and their respective officers, directors, employees and agents, from and against any and all claims arising out of your use of the licensed material other than as specifically authorized pursuant to this license.
11. No Transfer of License: This license is personal to you and may not be sublicensed, assigned, or transferred by you to any other person without publisher's written permission.

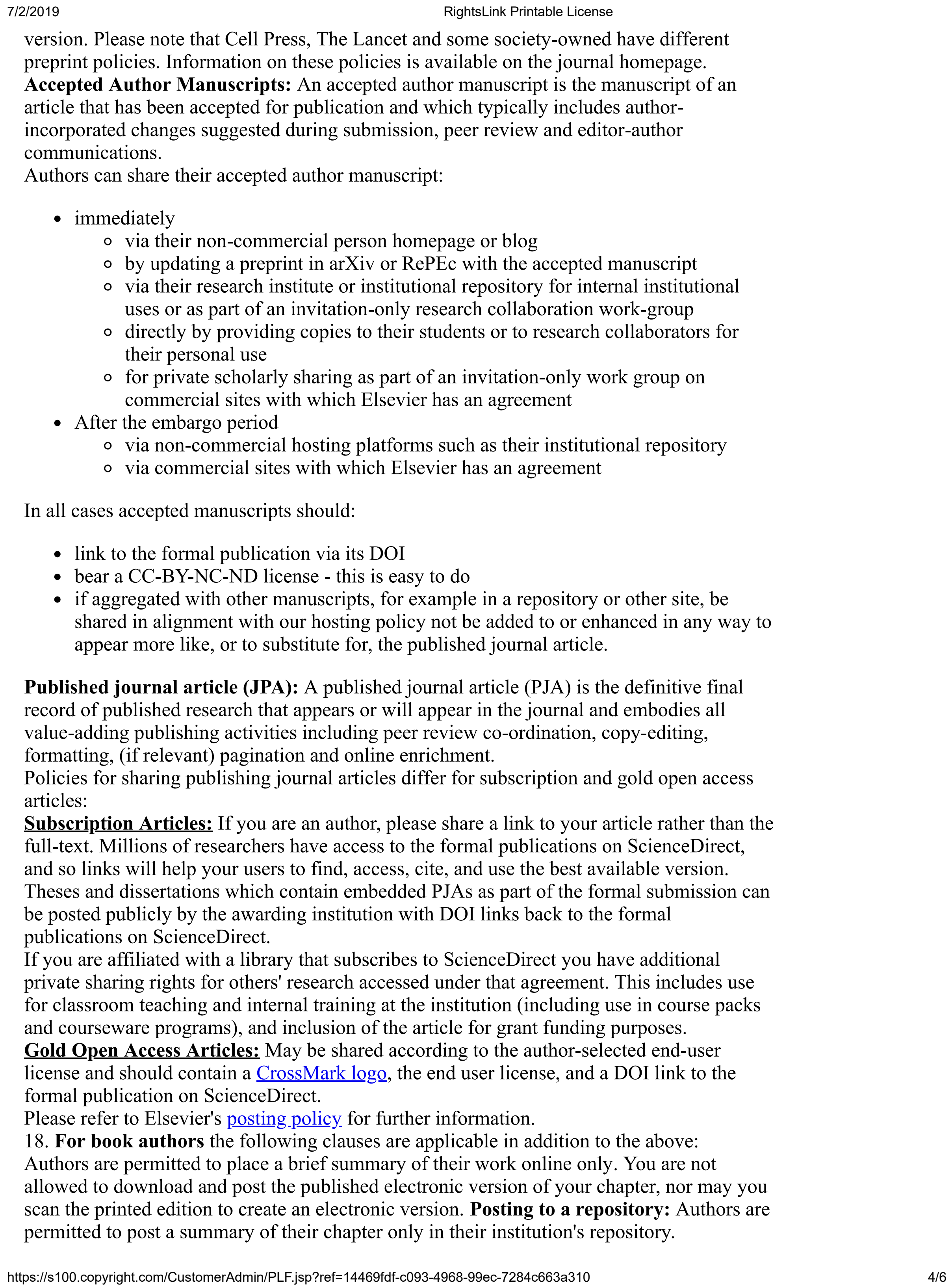


12. No Amendment Except in Writing: This license may not be amended except in a writing signed by both parties (or, in the case of publisher, by CCC on publisher's behalf).
13. Objection to Contrary Terms: Publisher hereby objects to any terms contained in any purchase order, acknowledgment, check endorsement or other writing prepared by you, which terms are inconsistent with these terms and conditions or CCC's Billing and Payment terms and conditions. These terms and conditions, together with CCC's Billing and Payment terms and conditions (which are incorporated herein), comprise the entire agreement between you and publisher (and CCC) concerning this licensing transaction. In the event of any conflict between your obligations established by these terms and conditions and those established by CCC's Billing and Payment terms and conditions, these terms and conditions shall control.
14. Revocation: Elsevier or Copyright Clearance Center may deny the permissions described in this License at their sole discretion, for any reason or no reason, with a full refund payable to you. Notice of such denial will be made using the contact information provided by you. Failure to receive such notice will not alter or invalidate the denial. In no event will Elsevier or Copyright Clearance Center be responsible or liable for any costs, expenses or damage incurred by you as a result of a denial of your permission request, other than a refund of the amount(s) paid by you to Elsevier and/or Copyright Clearance Center for denied permissions.

**LIMITED LICENSE**

The following terms and conditions apply only to specific license types:
15. **Translation**: This permission is granted for non-exclusive world **English** rights only unless your license was granted for translation rights. If you licensed translation rights you may only translate this content into the languages you requested. A professional translator must perform all translations and reproduce the content word for word preserving the integrity of the article.
16. **Posting licensed content on any Website**: The following terms and conditions apply as follows: Licensing material from an Elsevier journal: All content posted to the web site must maintain the copyright information line on the bottom of each image; A hyper-text must be included to the Homepage of the journal from which you are licensing at http://www.sciencedirect.com/science/journal/xxxxx or the Elsevier homepage for books at http://www.elsevier.com; Central Storage: This license does not include permission for a scanned version of the material to be stored in a central repository such as that provided by Heron/XanEdu.
Licensing material from an Elsevier book: A hyper-text link must be included to the Elsevier homepage at http://www.elsevier.com . All content posted to the web site must maintain the copyright information line on the bottom of each image.

**Posting licensed content on Electronic reserve**: In addition to the above the following clauses are applicable: The web site must be password-protected and made available only to bona fide students registered on a relevant course. This permission is granted for 1 year only. You may obtain a new license for future website posting.
17. **For journal authors:** the following clauses are applicable in addition to the above:
**Preprints:**
A preprint is an author's own write-up of research results and analysis, it has not been peer-reviewed, nor has it had any other value added to it by a publisher (such as formatting, copyright, technical enhancement etc.).
Authors can share their preprints anywhere at any time. Preprints should not be added to or enhanced in any way in order to appear more like, or to substitute for, the final versions of articles however authors can update their preprints on arXiv or RePEc with their Accepted Author Manuscript (see below).
If accepted for publication, we encourage authors to link from the preprint to their formal publication via its DOI. Millions of researchers have access to the formal publications on ScienceDirect, and so links will help users to find, access, cite and use the best available



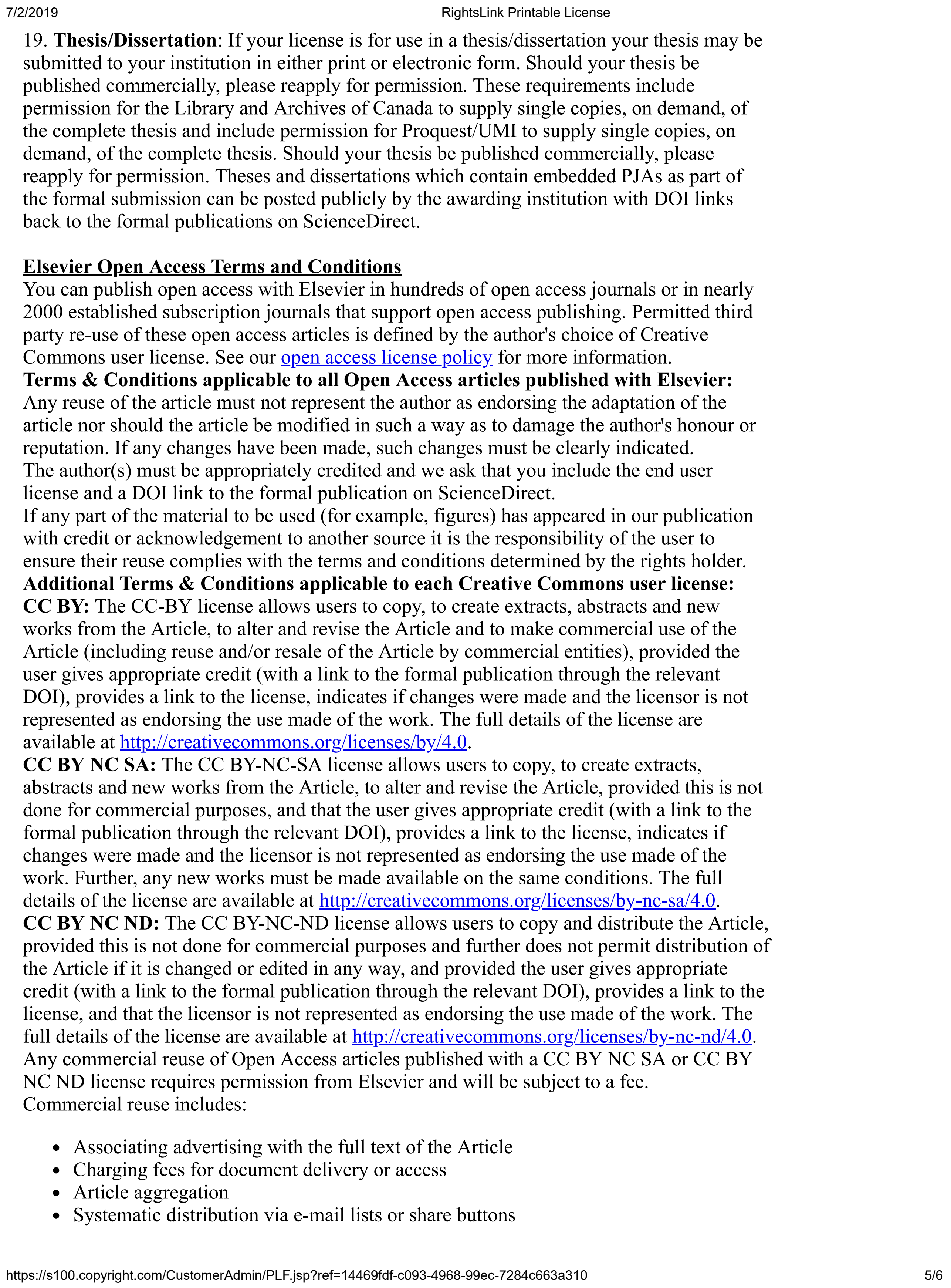
version. Please note that Cell Press, The Lancet and some society-owned have different preprint policies. Information on these policies is available on the journal homepage.
**Accepted Author Manuscripts:** An accepted author manuscript is the manuscript of an article that has been accepted for publication and which typically includes author-incorporated changes suggested during submission, peer review and editor-author communications.
Authors can share their accepted author manuscript:

- immediately
    - via their non-commercial person homepage or blog
    - by updating a preprint in arXiv or RePEc with the accepted manuscript
    - via their research institute or institutional repository for internal institutional uses or as part of an invitation-only research collaboration work-group
    - directly by providing copies to their students or to research collaborators for their personal use
    - for private scholarly sharing as part of an invitation-only work group on commercial sites with which Elsevier has an agreement
- After the embargo period
    - via non-commercial hosting platforms such as their institutional repository
    - via commercial sites with which Elsevier has an agreement

In all cases accepted manuscripts should:

- link to the formal publication via its DOI
- bear a CC-BY-NC-ND license - this is easy to do
- if aggregated with other manuscripts, for example in a repository or other site, be shared in alignment with our hosting policy not be added to or enhanced in any way to appear more like, or to substitute for, the published journal article.

**Published journal article (JPA):** A published journal article (PJA) is the definitive final record of published research that appears or will appear in the journal and embodies all value-adding publishing activities including peer review co-ordination, copy-editing, formatting, (if relevant) pagination and online enrichment.
Policies for sharing publishing journal articles differ for subscription and gold open access articles:
**Subscription Articles:** If you are an author, please share a link to your article rather than the full-text. Millions of researchers have access to the formal publications on ScienceDirect, and so links will help your users to find, access, cite, and use the best available version.
Theses and dissertations which contain embedded PJAs as part of the formal submission can be posted publicly by the awarding institution with DOI links back to the formal publications on ScienceDirect.
If you are affiliated with a library that subscribes to ScienceDirect you have additional private sharing rights for others' research accessed under that agreement. This includes use for classroom teaching and internal training at the institution (including use in course packs and courseware programs), and inclusion of the article for grant funding purposes.
**Gold Open Access Articles:** May be shared according to the author-selected end-user license and should contain a CrossMark logo, the end user license, and a DOI link to the formal publication on ScienceDirect.
Please refer to Elsevier's posting policy for further information.
18. **For book authors** the following clauses are applicable in addition to the above:
Authors are permitted to place a brief summary of their work online only. You are not allowed to download and post the published electronic version of your chapter, nor may you scan the printed edition to create an electronic version. **Posting to a repository:** Authors are permitted to post a summary of their chapter only in their institution's repository.



19. **Thesis/Dissertation**: If your license is for use in a thesis/dissertation your thesis may be submitted to your institution in either print or electronic form. Should your thesis be published commercially, please reapply for permission. These requirements include permission for the Library and Archives of Canada to supply single copies, on demand, of the complete thesis and include permission for Proquest/UMI to supply single copies, on demand, of the complete thesis. Should your thesis be published commercially, please reapply for permission. Theses and dissertations which contain embedded PJAs as part of the formal submission can be posted publicly by the awarding institution with DOI links back to the formal publications on ScienceDirect.

**Elsevier Open Access Terms and Conditions**

You can publish open access with Elsevier in hundreds of open access journals or in nearly 2000 established subscription journals that support open access publishing. Permitted third party re-use of these open access articles is defined by the author's choice of Creative Commons user license. See our open access license policy for more information.

**Terms & Conditions applicable to all Open Access articles published with Elsevier:**

Any reuse of the article must not represent the author as endorsing the adaptation of the article nor should the article be modified in such a way as to damage the author's honour or reputation. If any changes have been made, such changes must be clearly indicated.

The author(s) must be appropriately credited and we ask that you include the end user license and a DOI link to the formal publication on ScienceDirect.

If any part of the material to be used (for example, figures) has appeared in our publication with credit or acknowledgement to another source it is the responsibility of the user to ensure their reuse complies with the terms and conditions determined by the rights holder.

**Additional Terms & Conditions applicable to each Creative Commons user license:**

**CC BY:** The CC-BY license allows users to copy, to create extracts, abstracts and new works from the Article, to alter and revise the Article and to make commercial use of the Article (including reuse and/or resale of the Article by commercial entities), provided the user gives appropriate credit (with a link to the formal publication through the relevant DOI), provides a link to the license, indicates if changes were made and the licensor is not represented as endorsing the use made of the work. The full details of the license are available at http://creativecommons.org/licenses/by/4.0.

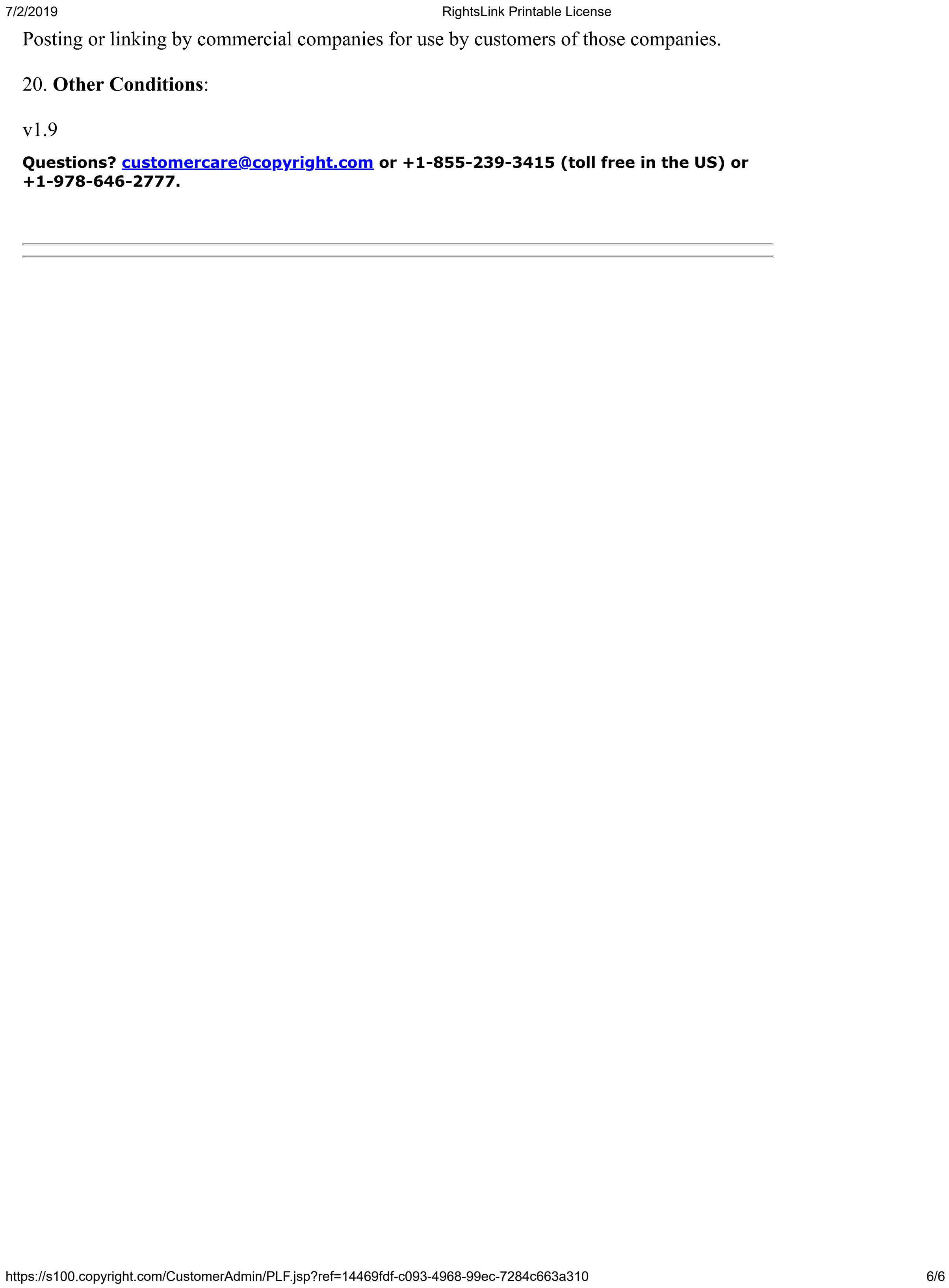
**CC BY NC SA:** The CC BY-NC-SA license allows users to copy, to create extracts, abstracts and new works from the Article, to alter and revise the Article, provided this is not done for commercial purposes, and that the user gives appropriate credit (with a link to the formal publication through the relevant DOI), provides a link to the license, indicates if changes were made and the licensor is not represented as endorsing the use made of the work. Further, any new works must be made available on the same conditions. The full details of the license are available at http://creativecommons.org/licenses/by-nc-sa/4.0.

**CC BY NC ND:** The CC BY-NC-ND license allows users to copy and distribute the Article, provided this is not done for commercial purposes and further does not permit distribution of the Article if it is changed or edited in any way, and provided the user gives appropriate credit (with a link to the formal publication through the relevant DOI), provides a link to the license, and that the licensor is not represented as endorsing the use made of the work. The full details of the license are available at http://creativecommons.org/licenses/by-nc-nd/4.0.

Any commercial reuse of Open Access articles published with a CC BY NC SA or CC BY NC ND license requires permission from Elsevier and will be subject to a fee.

Commercial reuse includes:

- Associating advertising with the full text of the Article
- Charging fees for document delivery or access
- Article aggregation
- Systematic distribution via e-mail lists or share buttons



Posting or linking by commercial companies for use by customers of those companies.

20. **Other Conditions**:

v1.9

**Questions? customercare@copyright.com or +1-855-239-3415 (toll free in the US) or +1-978-646-2777.**

# Appendix B

# MATLAB Code

## B.1 Data Processing Codes

### B.1.1 Extract Subject Data Code

```
%% Setup
clc; clear ; close all;

winSize = .250; %segment's windows size in s
winOverlap = .125; %window overlap (about 50% of windows size) in s
deadzone = 0.02;

% featsEMG = [];
% featsIMU = [];
% featsIMU1 = [];
% featsIMU2 = [];
% gest_labels = [];


aa = [];
aaimu = [];

%% Read Data
subjectID = [2:7 9:16]; % UNCOMENT TO GENERATE FEATURE MATRICES
for gg = subjectID % UNCOMENT TO GENERATE FEATURE MATRICES

% gg = 25; % COMENT TO GENERATE FEATURE MATRICES
```

```
dataFile = fullfile('Z:\Memo\Data\',sprintf('S_%d',gg),sprintf('expData_S%d.mat',gg));


workspaceVariable = sprintf('gestFeatures_S%d',gg);
dataFullFileName = fullfile('Z:\Memo\Feature Matrices',workspaceVariable);

% CLEAR VALUES HERE
featsEMG_Train = [];
featsIMU_Train = [];
featsIMU1_Train = [];
featsIMU2_Train = [];
gest_labels_Train = [];

featsEMG_Test = [];
featsIMU_Test = [];
featsIMU1_Test = [];
featsIMU2_Test = [];
gest_labels_Test = [];

% if exist([dataFullFileName '.mat'],'file')
%     load([dataFullFileName '.mat']);
% else
    emgFeats_all_Tr = [];
    imuFeats2_all_Tr = [];
    fusedFeats_all_Tr = [];
    emgFeats_all_Tst = [];
    imuFeats2_all_Tst = [];
    fusedFeats_all_Tst = [];
% end

load(dataFile);
for gestName = ["Wrist Flexion","Wrist Extension","Wrist Pronation","Wrist Supination","Wrist Aduction","Wrist Abduction","Hand Fist","Hand Open","Precision Pinch","Key Pinch"]

    switch gestName
    case "Wrist Flexion"
        gesture = 1;
    case "Wrist Extension"
        gesture = 2;
    case "Wrist Pronation"
```

```
        gesture = 3;
    case "Wrist Supination"
        gesture = 4;
    case "Wrist Aduction"
        gesture = 5; % Radial Deviation
    case "Wrist Abduction"
        gesture = 6; % Ulnar Deviation
    case "Hand Fist"
        gesture = 7;
    case "Hand Open"
        gesture = 8;
    case "Precision Pinch"
        gesture = 9;
    case "Key Pinch"
        gesture = 10;
    end
%     gesture = 4; % debug line

    for kk = 1:4 % Arm Position
%         kk = 3; % Debug line
%         rng(1); % comment after debuging
        order1 = randperm(10); % Randomize order of repetitions
        cont = 1;

        for ii = order1 % Repetition
%             ii = 7;%debug line
            if ~isempty(dataExp{ii,gesture,kk}) %gesture
                timeEMG_log = dataExp{ii,gesture,kk}{1,1};
                emg_log = dataExp{ii,gesture,kk}{1,2};
                timeIMU_log = dataExp{ii,gesture,kk}{1,3};
                imu_log = dataExp{ii,gesture,kk}{1,4};
            end
        % ============== CODE HERE ======================
        % tic
        [feats,~,~,feats_imu2,labels,actarea,actareaimu] = main_loop(gesture,timeEMG_log,
            emg_log,timeIMU_log,imu_log,winSize,winOverlap,deadzone);
%         [feats,~,~,feats_imu2,labels,actarea] = main_loop(gesture,timeEMG_log,emg_log,
    timeIMU_log,imu_log);
%         featsEMG = [featsEMG;feats];
%         %featsIMU = [featsIMU;feats_imu];
%         %featsIMU1 = [featsIMU1;feats_imu1];
%          featsIMU2 = [featsIMU2;feats_imu2];
%          gest_labels = [gest_labels;labels];


        if cont > 8
        featsEMG_Test = [featsEMG_Test;feats];
        %featsIMU_Test = [featsIMU_Test;feats_imu];
        %featsIMU1_Test = [featsIMU1_Test;feats_imu1];
        featsIMU2_Test = [featsIMU2_Test;feats_imu2];
        gest_labels_Test = [gest_labels_Test;labels];
        else
        featsEMG_Train = [featsEMG_Train;feats];
        %featsIMU_Train = [featsIMU_Train;feats_imu];
        %featsIMU1_Train = [featsIMU1_Train;feats_imu1];
        featsIMU2_Train = [featsIMU2_Train;feats_imu2];
        gest_labels_Train = [gest_labels_Train;labels];
        end
        cont = cont+1;

        aa = [aa;actarea];
        aaimu = [aaimu;actareaimu];

        % toc
        % =================================================
        end
%          plotStudyData2(dataExp,gesture,kk,aa,aaimu)
        aa = [];
        aaimu = [];
    end
end

% emgFeats = [featsEMG,gest_labels];
% imuFeats2 = [featsIMU2,gest_labels];
% fusedFeats = [featsEMG,featsIMU2,gest_labels];
%
% emgFeats_all = [emgFeats_all;emgFeats];
% imuFeats2_all = [imuFeats2_all;imuFeats2];
% fusedFeats_all = [fusedFeats_all;fusedFeats];

emgFeatsTr = [featsEMG_Train,gest_labels_Train];
imuFeats2Tr = [featsIMU2_Train,gest_labels_Train];
```

```
fusedFeatsTr = [featsEMG_Train,featsIMU2_Train,gest_labels_Train];

emgFeatsTst = [featsEMG_Test,gest_labels_Test];
imuFeats2Tst = [featsIMU2_Test,gest_labels_Test];
fusedFeatsTst = [featsEMG_Test,featsIMU2_Test,gest_labels_Test];

emgFeats_all_Tr = [emgFeats_all_Tr;emgFeatsTr];
imuFeats2_all_Tr = [imuFeats2_all_Tr;imuFeats2Tr];
fusedFeats_all_Tr = [fusedFeats_all_Tr;fusedFeatsTr];

emgFeats_all_Tst = [emgFeats_all_Tst;emgFeatsTst];
imuFeats2_all_Tst = [imuFeats2_all_Tst;imuFeats2Tst];
fusedFeats_all_Tst = [fusedFeats_all_Tst;fusedFeatsTst];

% save([dataFullFileName '.mat'],'emgFeats_all_Tr','imuFeats2_all_Tr','fusedFeats_all_Tr','emgFeats_all_Tst','imuFeats2_all_Tst','fusedFeats_all_Tst');


end % UNCOMENT TO GENERATE FEATURE MATRICES
```

### B.1.2 Main Routine Code

```
function [features, featuresIMU, featuresIMU1, featuresIMU2, labels, active_area, active_areaIMU] = main_loop(gesture,timeEMG,channels,timeIMU,imu_log,winSize,winOverlap,deadzone)
noSegmentation = 0;

if(nargin < 8)
    deadzone = 0.02;
    if (nargin < 7)
        winOverlap = .125;
        if (nargin < 6)
            noSegmentation = 1;
        end
    end
end


test_time = ceil(length(timeEMG)/200); %time duration of data acquisition in seconds
fs = length(timeEMG)/test_time; %real sampling frequency UNCOMMENT LINE
```

```
test_time_IMU = ceil(length(timeIMU)/50); %time duration of data acquisition in seconds
fs_IMU = length(timeIMU)/test_time_IMU; %real sampling frequency UNCOMMENT LINE


%% Conditioning

gyro = imu_log(:,5:7);
acc = imu_log(:,8:10);

signal = [gyro,acc];

% Because the timestamp on both timeEMG and timeIMU are different, we have
% to make them similar by substracting the first value from timeEMG and
% timeIMU from their respective vectors
timeEMG = timeEMG - timeEMG(1);
timeIMU = timeIMU - timeIMU(1);

%% Preprocessing
emgsignal = emgFilter(channels, fs); % EMG filter
IMU_signal_filt = imuFilter(signal,fs_IMU); % IMU filter (Remove 'gravity' (DC component) from the acceleration data)

% ===================================================================
% We will have to upsample the IMU signal so we can later fuse the
% feature vectors

% Upsample method one: repeat sample values
imuUp1 = (IMU_signal_filt(:)).';
imuUp1 = reshape(repmat(imuUp1,[4 1]),size(IMU_signal_filt,1)*4,size(IMU_signal_filt,2));

% Upsample method two: interpolate using a cubic spline
imuUp2 = interp1(timeIMU,IMU_signal_filt,timeEMG,'spline');

% ===================================================================
[ema, emga] = emgEnergy_test(emgsignal,fs); % Compute the Energy Moving Average
[onEMG,active_area] = onsetDetection_test(ema);

if length(onEMG)> 1200 || isempty(onEMG) % length bigger than 6 seconds UNCOMMENT LINE
onEMG = 800:1650;
active_area = [onEMG(1);onEMG(end)];
end % UNCOMMENT LINE
```

```

emgsignal_active = emgsignal(onEMG,:);

% ==================================================================
idx = timeEMG(active_area);
active_areaIMU = zeros(size(active_area));

for ii = 1:length(idx)
    temp = find(timeIMU >= idx(ii));
    active_areaIMU(ii) = temp(1);
end


cont = (active_areaIMU(2:2:end)-active_areaIMU(1:2:end-1))+1; % Se suma 1 para obtener el rango completo. Eg: [1:22]. El rango de valores es de 22 sin embargo, si restamos 22-1, obtendremos 21 por lo tanto para obtener el rango hay que sumarle 1 al resultado
check1 = reshape(active_areaIMU,2,[]);

rows_size = sum(cont);
idx1 = cumsum(cont);

onIMU = zeros(rows_size,1);

temp1 = 1;

for ii = 1:size(check1,2)
    onIMU(temp1:idx1(ii)) = check1(1,ii):check1(2,ii);
    temp1 = idx1(ii)+1;
end

IMU_active = IMU_signal_filt(onIMU,:);  % Non upsampled signal
IMU_active1 = imuUp1(onEMG,:);          % Upsampled signal method 1. We use same active area as EMG b/c both signals have the same sampling rate now
IMU_active2 = imuUp2(onEMG,:);          % Upsampled signal method 2. We use same active area as EMG b/c both signals have the same sampling rate now

%% Feature Extraction
if isempty(emgsignal_active)
    features = [];
    labels = [];
    active_area = [0;0];
    active_areaIMU = [0;0];
    return
end
if ~noSegmentation
    % ====================================================================
    winIncrement = winSize-winOverlap;
    win_size = ceil(winSize*fs);
    win_inc = ceil(winIncrement*fs);
    win_sizeIMU = ceil(winSize*fs_IMU);
    win_incIMU = ceil(winIncrement*fs_IMU);

    features = extract_feature(emgsignal_active,'winSize',win_size,'winInc',win_inc,'Features',["MAV" "MAVS" "WL" "ZC" "AR"]);
    featuresIMU = extract_feature(IMU_active,'winSize',win_sizeIMU,'winInc',win_incIMU,'Features',["MAV" "WL"]);
    featuresIMU1 = extract_feature(IMU_active1,'winSize',win_size,'winInc',win_inc,'Features',["MAV" "WL"]); % We use the same parameters as with the EMG signal b/c both signals have the same sampling rate now
    featuresIMU2 = extract_feature(IMU_active2,'winSize',win_size,'winInc',win_inc,'Features',["MAV" "WL"]); % We use the same parameters as with the EMG signal b/c both signals have the same sampling rate now
else
    features = extract_feature(emgsignal_active,'Features',["MAV" "WL" "ZC" "AR"]);
    featuresIMU = extract_feature(IMU_active,'Features',["MAV" "WL"]);
    featuresIMU1 = extract_feature(IMU_active1,'Features',["MAV" "WL"]); % We use the same parameters as with the EMG signal b/c both signals have the same sampling rate now
    featuresIMU2 = extract_feature(IMU_active2,'Features',["MAV" "WL"]); % We use the same parameters as with the EMG signal b/c both signals have the same sampling rate now
end

labels = ones(size(features,1),1).*gesture;

end
```

### B.1.3 Signal Filtering Codes

```
function [processed_signal] = emgFilter(signal, fs)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here

%% DC-Offset Removal
DCfilterEMG = signal-mean(signal);
```

```

%% Stop-Band Filter (Notch Filter)
fo = 60;
qf = 30; % Quality factor set to 30 to have a bandwith of 2 Hz
wo = fo/(fs/2);
bw = wo/qf;
[b,a] = iirnotch(wo,bw);

processed_signal = filtfilt(b,a,DCfilterEMG);

%% High-Pass Filter
fo = 20;
wo = fo/(fs/2);
[b,a] = butter(4,wo,'high');
processed_signal = filtfilt(b,a,processed_signal);
end
```

```
function [processed_signal] = imuFilter(signal, fs)
%% Band-Pass Filter (4th Order Butterworth Filter)

fcl = 0.2; %0.2 Hz cutoff frequency
fch = 15;

[b,a] = butter(4,[fcl,fch]/(fs/2),'bandpass'); %4th order butterworth

processed_signal = filtfilt(b,a,signal);


end
```

### B.1.4 Signal Conditioning Codes

```
function [emg_energy, emg_average] = emgEnergy_test(x,fs)
useTKEO = 1;
if nargin < 2
    useTKEO = 0;
end

[data, Nsignals] = size(x);

% Use TKEO before averaging
if useTKEO == 1
```

```
    emg_average = tkeo(x,fs);
else
    emg_average = x;
end


emg_average = mean(emg_average,2);

emg_average2 = emg_average.^2;

emg_average = [0; emg_average; 0] ;

windowSize = 60;
b = (1/windowSize)*ones(1,windowSize);
a = 1;

emg_energy = filter(b,a,emg_average2);

emg_energy = sqrt(emg_energy);
emg_energy = [0; emg_energy; 0];
end
```

```
function [emg_conditioned] = tkeo(x,fs)

TKEO = x(2:end-1,:).^2 - x(3:end,:) .* x(1:end-2,:);

%%
% Butterworth Filter
fc = 50; %5 Hz cutoff frequency

[b,a] = butter(4,fc/(fs/2)); %4th order butterworth

emg_conditioned = filtfilt(b,a,TKEO);

emg_conditioned = abs(emg_conditioned);


end
```

### B.1.5 Onset Detection Code

```
function [onEMG,check] = onsetDetection_test(emg_energy)
```

```
l = size(emg_energy,1);
check = zeros(l,1);


    function [pkthOn, pkthOff,P1] = peakDetection(emg_energy)
        ematst = emg_energy; %ema(5000:end);

        th =  max(ematst)*0.1; % 0.1
        [pks,locs_Swave] = findpeaks(ematst,'MinPeakHeight',th,'MinPeakDistance',100);


        pkthOn = mean(pks) * 0.2; % 0.1
        pkthOff = pkthOn *0.6;

        Fs = 200;            % Sampling frequency
        T = 1/Fs;             % Sampling period
        L = length(ematst);             % Length of signal
        Y = fft(ematst);
        P2 = abs(Y/L);
        P1 = P2(1:floor(L/2+1));
        P1(2:end-1) = 2*P1(2:end-1);
    end

[pkthOn, pkthOff, P1] = peakDetection(emg_energy);
aux = 0;
cont = 0;


% Data with length less than 100 ms is considered spurious due to the
% electromechanical delay of the muscles
st = 1;
en = length(emg_energy);
for ii = st: en
    if (emg_energy(ii) >= pkthOn)
        if (aux == 0)
            ta = ii;
            aux = 1;
        end
    end
    if (aux == 1)
        if (emg_energy(ii) <= pkthOff || ii == length(emg_energy))
```

```
            if (cont >= 750) % 20 samples = 100 ms at 200 Hz %Test with 750 samples: (3/4) of 5 secs
                check(ta) = 1;
                check(ii-1) = 1;
            end
            aux = 0;
            cont = 0;
        else
            cont = cont + 1;
        end
    end
end


% ========================================================================
% % % Code used to find the active area between '1':
% % % For example: [0 0 0 1 0 0 0 0 1 0 0 0 0] -> [0 0 0 1 1 1 1 1 1 0 0 0 0]
% ========================================================================

check = find(check == 1);
length_check = length(check);
if mod(length_check,2) ~= 0
    check = [check,length(emg_energy)];
end

cont = (check(2:2:end)-check(1:2:end-1))+1; % We add 1 to obtaine the complete range.
check1 = reshape(check,2,[]);

rows_size = sum(cont);
idx1 = cumsum(cont);

onEMG = zeros(rows_size,1);

temp1 = 1;

for ii = 1:size(check1,2)
    onEMG(temp1:idx1(ii)) = check1(1,ii):check1(2,ii);
    temp1 = idx1(ii)+1;
end

end
```

## B.2 Feature Extraction Codes

The code used for feature extraction from the EMG and IMU signal is presented in this section. To optimize the computational speed, feature matrices where prepared for vectorization. Some of the utilized code is based on a previous work presented in [119].

### B.2.1 Vectorization Code

```
%
% EXTRACT_FEATURE Extracts features from signal.
%
% [feature,seg_raw_data] = extract_feature(data,win_size,win_inc, deadzone)
%
% Author Jose Guillermo Colli Alfaro
%
% This function extracts features from the signal contained in data,
% which are stored in columns.
%
% The signals in data are divided into multiple windows of size
% win_size and the windows are space win_inc apart.
%
% A variable numwin is created to reshape the matrix data into a new
% matrix VECTDATA of size [win_size size(data,2)*numwin]. This new
% matrix is used as an input for the functions that compute the features
% of the EMG signal.
%
% Features are computed using Vectorization.
%
% 2018/04/23 Memo: First created
% 2018/04/24 Memo: Added code for reshaping vectdata
% 2018/11/22 Memo: Modified function to compute any feature. Added RMS
%                  feature calculation
% 2019/05/28 Memo: Added option to return segmented raw data

% function feature = extract_feature(data,win_size,win_inc,varargin)
function [feature, seg_raw_data] = extract_feature(data,varargin)
defaultFeatures = ["MAV" "MAVS" "WL" "ZC" "AR"];
defaultDeadzone = 0.02;
defaultAROrder = 4;
defaultWinSize = size(data,1);
```

```
defaultWinInc = 1;

p = inputParser;
addRequired(p,'data');
% addRequired(p,'win_size');
% addRequired(p,'win_inc');
addParameter(p,'winSize',defaultWinSize);
addParameter(p,'winInc',defaultWinInc);
addParameter(p,'Features',defaultFeatures);

%% Add parameters here:
% addParameter(p,'Parameter_Name',Default_parameter_value)
addParameter(p,'Deadzone',defaultDeadzone);
addParameter(p,'arOrder',defaultAROrder);

%% DO NOT MODIFY
% parse(p,data,win_size,win_inc,varargin{:});
parse(p,data,varargin{:});

%% Assign parameters to variables here:
% var_name = p.Results.Parameter_Name
feats = p.Results.Features;
deadzone = p.Results.Deadzone;
ar_order = p.Results.arOrder;
win_size = p.Results.winSize; % Test
win_inc = p.Results.winInc; % Test

%% DO NOT MODIFY
% Manipulate Data matrix for vectorization
[datasize, Nsignals] = size(data);
temp = 0:win_inc:(datasize-win_size); % Create temporal vector that will allow us to get the index of matrix Data
temp1 = 1:win_size; % Vector that will be sumed to temp vector to obtain index of matrix Data

idx = temp1+temp.'; % index matrix
idx = idx'; %transpose matrix and then...
idx = idx(:); %...unroll parameters      % 2018/04/24 Memo: probably this step is not necessary

vectdata = data(idx,:); %Create vectorized matrix
```

```
seg_raw_data = vectdata;  % Backup vectada, will be used later
numwin = size(vectdata,1)/win_size; %obtain #of windows in the vectorized matrix, there may be data loss if module(winsize/wininc) ~= 0

% This section is used to create a matrix which contains the index of the
% values in vectdata. This index matrix will be used to reshape the matrix
% vectdata so it can be further analyzed used vectorization. reshape
% function was not used because the way it sorted the columns

idxvector = 1:numel(vectdata); % Create a vector which values range from 1 to the # of elements in vectdata
idxvector = reshape(idxvector,size(vectdata)); % Reshape the vector into a matrix of the same size as vectdata
out = idxvector(1:win_size:end,:); % take every 'numwin' rows of idx vector
temp = reshape(out',[Nsignals*numwin 1]).'; % row-wise reshaping into a vector
temp1 = (0:(win_size-1))'; % Vector that will be sumed to temp vector to obtain index of matrix vectdata
idxmat = temp+temp1; % index matrix
vectdata = vectdata(idxmat); % vectadata now reshaped and ready to be used for vectorization

% ------------------------------------------------------------------------
% Initialize variable to Preallocate Memory
temp = size(feats,2);

%% Add any constraints here:
% E.g., AR features will return 1*ar_order
% features, this will affect the size of the final feature array.
% Therefore, we increase the size of our final array by ar_order-1.
% If ar_order = 4, then the columns of our final array should look
% something like this:
%           [AR AR1 AR2 AR3]; where AR1,AR2,AR3 are the 3 extra columns
%           added to the original vector size

% Blank sample function:
% if (any(feats == "Feature_Name"))
%     temp = temp + (feature_constraint - 1);
% end

if (any(feats == "AR"))
    temp = temp + (ar_order - 1);
```

```
end

%% DO NOT MODIFY
% Preallocate Memory and initialize index variable
feature = zeros(numwin,Nsignals*temp);
idx = 1;

%% Add functions for calculating new features here:
for ii = feats
  switch ii
% Blank sample function:
%   case 'Feature_Name'
%       feat = your_feature_function
%       feature(:,idx:(idx+size(feat,2)-1)) = feat; % DO NOT DELETE
%       idx = idx+size(feat,2);                     % DO NOT DELETE
  case 'MAV'
      [feat, ~] = getmavfeat_test(vectdata,Nsignals,numwin);
      feature(:,idx:(idx+size(feat,2)-1)) = feat;
      idx = idx+size(feat,2);
  case 'MAVS'
      [~, feat] = getmavfeat_test(vectdata,Nsignals,numwin);
      feature(:,idx:(idx+size(feat,2)-1)) = feat;
      idx = idx+size(feat,2);
  case 'WL'
      feat = getwlfeat_test(vectdata,Nsignals,numwin);
      feature(:,idx:(idx+size(feat,2)-1)) = feat;
      idx = idx+size(feat,2);
  case 'ZC'
      [feat] = getzcfeat_test(vectdata,deadzone,win_size,Nsignals,numwin);
      feature(:,idx:(idx+size(feat,2)-1)) = feat;
      idx = idx+size(feat,2);
  case 'AR'
      feat = getarfeat_test(vectdata,ar_order,Nsignals,numwin);
      feature(:,idx:(idx+size(feat,2)-1)) = feat;
      idx = idx+size(feat,2);
  case 'RMS'
      feat = getrmsfeat_test(vectdata,Nsignals,numwin);
      feature(:,idx:(idx+size(feat,2)-1)) = feat;
      idx = idx+size(feat,2);
  case 'MEAN'
      feat = getmeanfeat_test(vectdata,Nsignals,numwin);
      feature(:,idx:(idx+size(feat,2)-1)) = feat; % DO NOT DELETE
      idx = idx+size(feat,2);                     % DO NOT DELETE
  case 'STD'
      feat = getstdfeat_test(vectdata,Nsignals,numwin);
      feature(:,idx:(idx+size(feat,2)-1)) = feat; % DO NOT DELETE
      idx = idx+size(feat,2);                     % DO NOT DELETE
  end
end

% Test Matrix... YOU CAN IGNORE THIS
% Add the new computed feature to this matrix
% feature = [MAV MAVS WL ZC AR1 AR2 AR3 AR4 RMS]
% feature = [feature1 feature2 feature3 feature4 feature5 feature6];

end
```

### B.2.2 MAV and MAVS Features Code

```
function [feat, feat2] = getmavfeat_test(x,nsignals,nwindows)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here

% allocate memory
% feat = zeros(nwindows,nsignals);

feat = mean(abs(x));
feat = (reshape(feat, [nsignals,nwindows])).';

% feat2 = [diff(feat); zeros(1,nsignals)];
feat2 = diff([feat;feat(end,:)]);
end
```

### B.2.3 WL Feature Code

```
function feat = getwlfeat_test(x,nsignals,nwindows)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here

feat = sum(abs(diff(x)));
feat = (reshape(feat, [nsignals,nwindows])).';

end
```

### B.2.4 ZC Feature Code

```
function [feat] = getzcfeat_test(x,deadzone,winsize,nsignals,nwindows)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here


y = (x > deadzone) - (x < -deadzone);

%    % forces the zeros towards either the positive or negative
%    % the filter is chosen so that the most recent +1 or -1 has
%    % the most influence on the state of the zero.
   a=1;
   b=exp(-(1:winsize/2))';
   z = filter(b,a,y);


   z = (z > 0) - (z < -0);
   dz = diff(z);

   feat = (sum(abs(dz)==2));
   feat = (reshape(feat, [nsignals,nwindows])).';

% C = sign(x(1:end-1,:) .* x(2:end,:));
% C(C~=-1)=0;
% C = abs(C);
%
% A = abs(x(1:end-1,:) - x(2:end,:));
%
% y = sign(C);
% y(y~=-1)=0;
% z(z<=deadzone)=0;
% z(z>deadzone)=1;
% dz = (-y)&z;
% feat2 = sum(dz);
% feat2 = (reshape(feat2, [nsignals,nwindows])).';


end
```

### B.2.5 AR Coefficients Feature Code

```
function feat = getarfeat_test(x,order,nsignals,nwindows)
%UNTITLED3 Summary of this function goes here
%   Detailed explanation goes here

% While aryule might be more accurate, lpc is faster for computing the
% autoregressive coefficients. Theorethically, the computed values are the
% same. The might be different by a couple of decimals

% ar = aryule(x,order)'; % transpose b/c the parameters along the nth row of 'ar' model the nth column of x
% feat = reshape(ar(2:end,:),nsignals*order*nwindows,1)';

cur_xlpc = real(lpc(x,order)');
feat = reshape(cur_xlpc(2:end,:),nsignals*order*nwindows,1)';

feat = (reshape(feat, [nsignals*order,nwindows])).';

end
```

### B.2.6 Mean Feature Code

```
function [feat] = getmeanfeat_test(x,nsignals,nwindows)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here
feat = mean(x);
feat = (reshape(feat, [nsignals,nwindows])).'; % DO NOT DELETE

end
```

### B.2.7 Std Feature Code

```
function [feat] = getstdfeat_test(x,nsignals,nwindows)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here

feat = std(x);
feat = (reshape(feat, [nsignals,nwindows])).';
end
```

### B.2.8 RMS Feature Code

```
function feat = getrmsfeat_test(x,nsignals,nwindows)
```

```
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here

% feat = sqrt(mean(x.^2));
feat = rms(x);
feat = (reshape(feat, [nsignals,nwindows])).';

end
```

## B.3 Feature Normalization Code

```
function [x_norm, x_norm2, mu, sigma] = featureNormalize(x, mu, sigma)
%FEATURENORMALIZE Normalizes the features in X
%   FEATURENORMALIZE(X) returns a normalized version of X where
%   the mean value of each feature is 0 and the standard deviation
%   is 1. This is often a good preprocessing step to do when
%   working with learning algorithms.

if nargin < 2
    mu = mean(x);
    x_norm = x - mu;

    sigma = std(x_norm);
    x_norm = x_norm./sigma;
else
    x_norm = (x - mu)./sigma;
end


% Change the range between -1 and 1
x_norm2 = 2.*((x_norm - min(x_norm,[],1))./(max(x_norm,[],1)-min(x_norm,[],1)))-1;
end
```

## B.4 Feature Reduction Code

```
function [Z, K, U, maxVar] = projectData(X, K, U)
%PROJECTDATA Computes the reduced data representation when projecting only
%on to the top k eigenvectors
%   Z = projectData(X, K, U) computes the projection of
%   the normalized inputs X into the reduced dimensional space spanned by
%   the first K columns of U. It returns the projected examples in Z.
```

```
%

findK = 0;
computeU = 0;
if nargin < 3
    computeU = 1;
    if nargin < 2
        K = size(X,2);
        findK = 1;
    end
end

% Useful values
[m, n] = size(X);

if computeU
    % Compute covariance matrix Sigma
    Sigma = (X' * X) / (m-1);

    % Compute eigenvectors of matrix Sigma
    [U, S, V] = svd(Sigma);

    % Obtain variances
    idx = find(S);
    den = sum(S(idx));
    temp = cumsum(S(idx))./den;

    % =======================================================================
    % ========= COMMENT THIS CODE IF ALREADY KNOW THE VALUE OF K ============
    % =======================================================================

    if findK
        % Use singular value matrix S to compute the # of principal components K.

        % Matrix S is a diagonal matrix that contains the square roots of the
        % non-zero eigenvalues of both Sigma'*Sigma and Sigma*Simga', where
        % Sigma' is the conjugate transpose of Sigma.

        %     idx = find(S);
        %     den = sum(S(idx));
        %     temp = cumsum(S(idx))./den;
        K = find(temp>=0.95); % Find K so that reduced matrix retains 99% variance

        % If K = 0, then reduce varaince tolerance up to 90% MAX. If K is still
        % zero, then it is not possible to reduce the matrix

        if isempty(K)
            K = find(temp>=0.95);
            if isempty(K)
                K = find(temp>=0.90);
                if isempty(K)
                    K = n;
                end
            end
        end

        K = min(K);

    end

    % =======================================================================

    maxVar = temp(K);
end

Z = X * U(:, 1:K);


end
```

## B.5 Classification Codes

### B.5.1 LS-SVM Code

```
% featMatID = 2;
reduce = 1; %'1' or '0'
znorm = 0;
remove_gests = 1;

for featMatID = 1:2
    for subID = [2:7,9:20,22:25] %[2:7,9:16]
        workspaceVariable = sprintf('gestFeatures_S%d',subID);
        dataFullFileName = fullfile('Z:\Memo\Feature Matrices',workspaceVariable);
        load(dataFullFileName);
        switch featMatID
            case 1
                featMat = [fusedFeats_all_Tr; fusedFeats_all_Tst];
                if remove_gests == 1
                   featMat = reduce_data(featMat);
                end

                if znorm == 1
                    [feature_norm3,~] = featureNormalize(featMat(:,1:end-1));
                else
                [~,feature_norm3] = featureNormalize(featMat(:,1:end-1));
                end
                fileName = 'fusedFeats';
                if exist('bestcAll', 'var')
                    C = bestcAll;%2
                    gamma = bestgAll; %0.5
                else
                    C = 2;
                    gamma = 0.5;
                end
            case 2
                featMat = [emgFeats_all_Tr; emgFeats_all_Tst];
                if remove_gests == 1
                   featMat = reduce_data(featMat);
                end

                if znorm == 1
                    [feature_norm3,~,mu,sigma] = featureNormalize(featMat(:,1:end-1));
                else
                [~,feature_norm3,mu,sigma] = featureNormalize(featMat(:,1:end-1));
                end

                fileName = 'emgFeats';
                if exist('bestc', 'var')
                    C = bestc;%2
                    gamma = bestg; %0.5
                else
                    C = 2;
                    gamma = 0.5;
                end
            case 3
                featMat = [imuFeats2_all_Tr; imuFeats2_all_Tst];
                if remove_gests == 1
                   featMat = reduce_data(featMat);
                end
                [~,feature_norm3] = featureNormalize(featMat(:,1:end-1));
                fileName = 'imu2Feats';
        end

        if reduce == 1
            featPCA2 = projectData(feature_norm3,17);
        else
            featPCA2 = feature_norm3;
        end

        Y = featMat(:,end);

        nLab = numel(unique(Y));
        Yi = Y;

        % Kernel Matrix
        K = rbfKernel(featPCA2,featPCA2,gamma);
        H = K; H(1:size(K,1)+1:end) = H(1:size(K,1)+1:end)+(1/C);

        alphas = zeros(size(H,1),nLab);
        betas = zeros(nLab,1);

        Ev = ones(size(H,1),1);



        % Cholesky factorization
        R = chol(H,'lower');

        labels = unique(Y);
        for ii = 1:nLab
            Yi(Y==ii) = 1;
            Yi(Y~=ii) = -1;

            % Solve for eta (n) in H*eta = 1. Use the Cholesky factor of H
            eta = R.'\(R\Ev);

            % Solve for nu (v) in H*nu = y. Use the Cholesky factor of H
            nu = R.'\(R\Yi);

            betas(ii) = (Ev.'*nu)/(Ev.'*eta);
            alphas(:,ii) = nu - (eta*betas(ii));
        end

        switch featMatID
            case 1
                if reduce == 1
                    if znorm == 1
                        LSSVM_Mdl_EMGIMU_znorm.eta = eta;
                        LSSVM_Mdl_EMGIMU_znorm.nu = nu;
                        LSSVM_Mdl_EMGIMU_znorm.betas = betas;
                        LSSVM_Mdl_EMGIMU_znorm.alphas = alphas;
                        LSSVM_Mdl_EMGIMU_znorm.datTr = featPCA2; %featPCA2
                        LSSVM_Mdl_EMGIMU_znorm.C = C;
                        LSSVM_Mdl_EMGIMU_znorm.gamma = gamma;

                        save(dataFullFileName,'LSSVM_Mdl_EMGIMU_znorm','-append');
                    else
                        LSSVM_Mdl_EMGIMU_7_gest.eta = eta;
                        LSSVM_Mdl_EMGIMU_7_gest.nu = nu;
                        LSSVM_Mdl_EMGIMU_7_gest.betas = betas;
                        LSSVM_Mdl_EMGIMU_7_gest.alphas = alphas;
                        LSSVM_Mdl_EMGIMU_7_gest.datTr = featPCA2; %featPCA2
                        LSSVM_Mdl_EMGIMU_7_gest.C = C;
                        LSSVM_Mdl_EMGIMU_7_gest.gamma = gamma;

                        save(dataFullFileName,'LSSVM_Mdl_EMGIMU_7_gest','-append');
                    end

                else
                    LSSVM_Mdl_EMGIMU_NoPCA.eta = eta;
                    LSSVM_Mdl_EMGIMU_NoPCA.nu = nu;
                    LSSVM_Mdl_EMGIMU_NoPCA.betas = betas;
                    LSSVM_Mdl_EMGIMU_NoPCA.alphas = alphas;
                    LSSVM_Mdl_EMGIMU_NoPCA.datTr = featPCA2; %featPCA2
                    LSSVM_Mdl_EMGIMU_NoPCA.C = C;
                    LSSVM_Mdl_EMGIMU_NoPCA.gamma = gamma;
                    save(dataFullFileName,'LSSVM_Mdl_EMGIMU_NoPCA','-append');
                end
            case 2
                if reduce == 1
                    if znorm == 1
                        LSSVM_Mdl_EMG_znorm.eta = eta;
                        LSSVM_Mdl_EMG_znorm.nu = nu;
                        LSSVM_Mdl_EMG_znorm.betas = betas;
                        LSSVM_Mdl_EMG_znorm.alphas = alphas;
                        LSSVM_Mdl_EMG_znorm.datTr = featPCA2; %featPCA2
                        LSSVM_Mdl_EMG_znorm.C = C;
                        LSSVM_Mdl_EMG_znorm.gamma = gamma;

                        save(dataFullFileName,'LSSVM_Mdl_EMG_znorm','-append');
                    else
                        LSSVM_Mdl_EMG_7_gest.eta = eta;
                        LSSVM_Mdl_EMG_7_gest.nu = nu;
                        LSSVM_Mdl_EMG_7_gest.betas = betas;
                        LSSVM_Mdl_EMG_7_gest.alphas = alphas;
                        LSSVM_Mdl_EMG_7_gest.datTr = featPCA2; %featPCA2
                        LSSVM_Mdl_EMG_7_gest.C = C;
                        LSSVM_Mdl_EMG_7_gest.gamma = gamma;

                        save(dataFullFileName,'LSSVM_Mdl_EMG_7_gest','-append');
                    end
                else
                    LSSVM_Mdl_EMG_NoPCA.eta = eta;
                    LSSVM_Mdl_EMG_NoPCA.nu = nu;
                    LSSVM_Mdl_EMG_NoPCA.betas = betas;
                    LSSVM_Mdl_EMG_NoPCA.alphas = alphas;
                    LSSVM_Mdl_EMG_NoPCA.datTr = featPCA2; %featPCA2
                    LSSVM_Mdl_EMG_NoPCA.C = C;
                    LSSVM_Mdl_EMG_NoPCA.gamma = gamma;
                    save(dataFullFileName,'LSSVM_Mdl_EMG_NoPCA','-append');
                end
            case 3
                LSSVM_Mdl_IMU.eta = eta;
                LSSVM_Mdl_IMU.nu = nu;
                LSSVM_Mdl_IMU.betas = betas;
                LSSVM_Mdl_IMU.alphas = alphas;
                LSSVM_Mdl_IMU.datTr = featPCA2; %featPCA2
                LSSVM_Mdl_IMU.C = C;
                LSSVM_Mdl_IMU.gamma = gamma;
                save(dataFullFileName,'LSSVM_Mdl_IMU','-append');
        end
    end
end
```

#### B.5.1.1 Predict LS-SVM Code

```
function [Ys,pred,KK] = predLSSVM(datTst,varargin)
%UNTITLED5 Summary of this function goes here
%   Detailed explanation goes here
defaultModel = [];
defaultdatTr = [];
defaultAlpha = [];
defaultBeta = [];
defaultGamma = [];

validStruct = @(x) isstruct(x);

p = inputParser;
addRequired(p,'datTst');
addOptional(p,'dataTr',defaultdatTr);
addOptional(p,'alpha',defaultAlpha);
addOptional(p,'beta',defaultBeta);
addOptional(p,'gam',defaultGamma);
addParameter(p,'Model',defaultModel,validStruct);

parse(p,datTst,varargin{:});
Mdl = p.Results.Model;
datTr = p.Results.dataTr;
alphas = p.Results.alpha;
betas = p.Results.beta;
gamma = p.Results.gam;

%%

if ~isempty(Mdl)
    KK = rbfKernel(datTst,Mdl.datTr,Mdl.gamma);
    Y = (Mdl.alphas).'*KK.';
    % Ys = sign(Ys+Mdl.betas);
    Y = Y+Mdl.betas;
```

```
else
    if isempty(datTr) || isempty(alphas) || isempty(betas) || isempty(gamma)
        error('Not enough input arguments')
    end
    KK = rbfKernel(datTst,datTr,gamma);
    Y = (alphas).'*KK.';
    Y = Y+betas;
end



[~,p2]=max(Y);
pred = p2.';
lab = (1:size(Y,1)).';
G = length(lab);
N = length(p2);

Ys = repmat(lab, [1 N]);
Ys = Ys==repmat(p2,[G 1]);
Ys = double(Ys);
Ys(Ys==0) = -1;

end
```

### B.5.1.2 Get LS-SVM Parameters Code

```
function [R,alphas,betas] = findSVMParam(H,Y,varargin)
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here
defaultChol = 'upper';

p = inputParser;
addRequired(p,'H');
addRequired(p,'Y');
% addRequired(p,'win_inc');
addParameter(p,'cholFac',defaultChol);

parse(p,H,Y,varargin{:});

cholFactor = p.Results.cholFac;

%%
```

```
nLab = numel(unique(Y));
Yi = Y;
alphas = zeros(size(H,1),nLab);
betas = zeros(nLab,1);

Ev = ones(size(H,1),1);

if cholFactor == "lower"
    % Cholesky factorization
    R = chol(H,'lower');

    labels = unique(Y);
    for ii = 1:nLab
        Yi(Y==ii) = 1;
        Yi(Y~=ii) = -1;

        % Solve for eta (n) in H*eta = 1. Use the Cholesky factor of H
        eta = R.'\(R\Ev);

        % Solve for nu (v) in H*nu = y. Use the Cholesky factor of H
        nu = R.'\(R\Yi);

        betas(ii) = (Ev.'*nu)/(Ev.'*eta);
        alphas(:,ii) = nu - (eta*betas(ii));
    end
else
    % Cholesky factorization
    R = chol(H);

    labels = unique(Y);
    for ii = 1:nLab
        Yi(Y==ii) = 1;
        Yi(Y~=ii) = -1;

        % Solve for eta (n) in H*eta = 1. Use the Cholesky factor of H
        eta = R\(R.'\Ev);

        % Solve for nu (v) in H*nu = y. Use the Cholesky factor of H
        nu = R\(R.'\Yi);

        betas(ii) = (Ev.'*nu)/(Ev.'*eta);
        alphas(:,ii) = nu - (eta*betas(ii));
    end
end
end
```

### B.5.1.3 Compute RBF Kernel Code

```
function sim = rbfKernel(X, Y, gamma)
%RBFKERNEL returns a radial basis function kernel between X and Y
%   sim = gaussianKernel(X, Y) returns a gaussian kernel between X and Y
%   and returns the value in sim. RBF is computed using vectorization

sim = exp(-gamma .* pdist2(X,Y,'euclidean').^2);

end
```

## B.5.2 PAC Code

```
function [accEMG,accEMGIMU,preds] = particleAdaptiveClass(subjects,mdlID,cont,reduce)
%UNTITLED3 Summary of this function goes here
%   Detailed explanation goes here
accEMGIMU = zeros(size(subjects));
accEMG = zeros(size(subjects));
preds = cell(2,numel(subjects));


for jj = 1:length(subjects)
    subID = subjects(jj);
    for featMatID = 1:2
        switch featMatID
            case 1
%                 load(fullfile('Z:\Memo\Feature Matrices',sprintf('gestFeatures_S%d',subID)),...
%                     'fusedFeats_all_Tr','fusedFeats_all_Tst','LSSVM_Mdl_EMGIMU','LSSVM_Mdl_EMGIMU_NoPCA');
                load(fullfile('Z:\Memo\Feature Matrices',sprintf('gestFeatures_S%d',subID)),...
                    'fusedFeats_all_Tr','fusedFeats_all_Tst','LSSVM_Mdl_EMGIMU_7_gest','LSSVM_Mdl_EMGIMU_NoPCA');

                featMat = [fusedFeats_all_Tr; fusedFeats_all_Tst];
                featMat = reduce_data(featMat); % Uncomment if testing for 7 gestures

                [~,feature_norm3] = featureNormalize(featMat(:,1:end-1));
                if reduce == 1
%                     mdlName = 'LSSVM_Mdl_EMGIMU';
                    mdlName = 'LSSVM_Mdl_EMGIMU_7_gest';
                else
                    mdlName = 'LSSVM_Mdl_EMGIMU_NoPCA';
                end
            case 2
%                 load(fullfile('Z:\Memo\Feature Matrices',sprintf('gestFeatures_S%d',subID)),...
%                     'emgFeats_all_Tr','emgFeats_all_Tst','LSSVM_Mdl_EMG','LSSVM_Mdl_EMG_NoPCA');
                load(fullfile('Z:\Memo\Feature Matrices',sprintf('gestFeatures_S%d',subID)),...
                    'emgFeats_all_Tr','emgFeats_all_Tst','LSSVM_Mdl_EMG_7_gest','LSSVM_Mdl_EMG_NoPCA');

                featMat = [emgFeats_all_Tr; emgFeats_all_Tst];
                featMat = reduce_data(featMat); % Uncomment if testing for 7 gestures

                [~,feature_norm3] = featureNormalize(featMat(:,1:end-1));
                if reduce == 1
%                     mdlName = 'LSSVM_Mdl_EMG';
                    mdlName = 'LSSVM_Mdl_EMG_7_gest';
                else
                    mdlName = 'LSSVM_Mdl_EMG_NoPCA';
                end
            case 3
                feature_norm = featureNormalize(imuFeats2_all_Tr(:,1:end-1));
                feature_norm2 = featureNormalize(imuFeats2_all_Tst(:,1:end-1));

                featMat = [imuFeats2_all_Tr; imuFeats2_all_Tst];
                [~,feature_norm3] = featureNormalize(featMat(:,1:end-1));
                mdlName = 'LSSVM_Mdl_IMU';
        end
        % Data reduction
        if reduce == 1
            featPCA2 = projectData(feature_norm3,17);
        else
            featPCA2 = feature_norm3;
        end

        % Separate data
        if featMatID == 1
            rng('shuffle')
            order = randperm(40);
            labels = featMat(:,end);
            reps = cell(1,40);
            idx1 = 1;
            idx2 = 1;
            idxTr = [];
            idxTst = [];
            %          cont = 1;   % Change this value to determine the number of reps used for the training set
            for ii = 1:10
                aux1 = zeros(1,40);
                aux2 = ones(1,40);

                idxAux= find(labels == ii);
                temp = floor(length(idxAux)/40);
                aux2 = aux2*temp;
                aux1(1:length(idxAux)-(temp*40)) = 1;
                aux2 = aux2+aux1;

                sizeReps = cumsum(aux2).';
                sizeReps2 = sizeReps+1;
                sizeReps2 = [1; sizeReps2(1:end-1)];

                idx_temp = [sizeReps2 sizeReps];
                for kk = 1:40
                    reps{kk} = idxAux(idx_temp(kk,1):idx_temp(kk,2));
                end
                reps_temp = reps(order);
                for kk = 1:40
                    if kk <= cont
                        idxTr = [idxTr; reps_temp{kk}];
                    else
                        idxTst = [idxTst; reps_temp{kk}];
                    end
                end
            end
        end

        % Assign testing and training set
        featMatTr = featPCA2(idxTr,:);
        featMatTst = featPCA2(idxTst,:);
        labelsTr = labels(idxTr);
        labelsTst = labels(idxTst);

        %% Train data using different models
        k = length(mdlID);

        % Find the best model that fit the training set
        acc = 0;
        for ii = 1:k
            load(fullfile('Z:\Memo\Feature Matrices',sprintf('gestFeatures_S%d',mdlID(ii)))
                ,mdlName);
            aux = eval(mdlName);
            [~,pred] = predLSSVM(featMatTr,'Model',aux);
            CC = confusionmat(labelsTr,pred);
            C1 = CC.*eye(length(CC));
            C2 = sum(CC,2);
            a = (sum(sum(C1,2)./C2)/(length(unique(labelsTr))))*100;
            if a > acc
                acc = a;
                bestS = mdlID(ii);
                datTr = aux.datTr;
                % Change SVM variables
                gamma = aux.gamma;
                C = aux.C;
                bestMdl = aux;
                bestPred = pred;
            end
        end

        switch featMatID
            case 1
                aux = load(fullfile('Z:\Memo\Feature Matrices',sprintf('gestFeatures_S%d',
                    bestS)),'fusedFeats_all_Tr', 'fusedFeats_all_Tst');
                labTr = [aux.fusedFeats_all_Tr; aux.fusedFeats_all_Tst];
                labTr = reduce_data(labTr); % Uncomment if testing for 7 gestures
            case 2
```

```
                aux = load(fullfile('Z:\Memo\Feature Matrices',sprintf('gestFeatures_S%d',
                    bestS)),'emgFeats_all_Tr', 'emgFeats_all_Tst');
                labTr = [aux.emgFeats_all_Tr; aux.emgFeats_all_Tst];
                labTr = reduce_data(labTr); % Uncomment if testing for 7 gestures
        end

        labTr = labTr(:,end);

        %% PAC algorithm
        % Vectorized RBF Kernel
        % This is equivalent to computing the kernel on every pair of examples
        K = rbfKernel(datTr,datTr,gamma);     % All training data kernel
        H = K; H(1:size(K,1)+1:end) = H(1:size(K,1)+1:end)+(1/C);

        % ================================================================
        % GPU stuff
        reqMemory = numel(K)*8;

        % Reset GPU Memory
        gpuDev = gpuDevice(1);
        if gpuDev.AvailableMemory > reqMemory
            kD = gpuArray(abs((2-(2*K)).^(1/2)));
            kernelDist = gather(kD);

            % Reset GPU Memory
            gpuDev = gpuDevice(1);
        else
            % Compute kernel distance matrix
            kernelDist = abs((2-(2*K)).^(1/2));
        end

        % ================================================================

        % Split training samples within each class into 'm' clusters and extract
        % representative particles with percentage 'p' into dataset RSS
        m = 10;     % Recomended value between 9 and 28
        p = 0.23;   % Recomended value between 10% and 20%
        labs = unique(labTr);
        rpMat = zeros(size(labTr));
        idx_rpMat = 1;
        % For each class
        for ii = 1:length(labs)
            idx = find(labTr==labs(ii));
            dist_Class = kernelDist(idx,idx);
            aux = kmedoids1(m,dist_Class);

            % Preallocate Memory and initialize index variable
            repPart = ceil(histcounts(aux).*p);
            idxMat = zeros(sum(repPart),1);
            idxAux = 1;

            % Randomly select samples from each cluster
            for kk = 1:length(repPart)
                aux_idx = find(aux == kk);
                aux_idx = aux_idx(randperm(numel(aux_idx)));
                idxMat(idxAux:(idxAux+repPart(kk)-1)) = idx(aux_idx(1:repPart(kk)));
                idxAux = idxAux+repPart(kk);
            end

            % Save selected samples in rpMat and move its index value to the next
            % available space
            rpMat(idx_rpMat:idx_rpMat+sum(repPart)-1) = idxMat;
            idx_rpMat = idx_rpMat+sum(repPart);
        end

        % Remove zero values from rpMat
        rpMat = sort(rpMat(rpMat~=0));

        % Create new training dataset RSS
        RSS = datTr(rpMat,:);
        RSS_H = H(rpMat,rpMat);

        % Obtain labels of RSS
        labelsTrRSS = labTr(rpMat);

        [R,alphas,betas] = findSVMParam(RSS_H,labelsTrRSS);

        lambda = 10e5;
        dTh = 0.99;
        ti = zeros(1,size(RSS,1));
        for ii = 1:size(featMatTr,1)
            % Predict new sample
            [~,Yn,KK] = predLSSVM(featMatTr(ii,:),RSS,alphas,betas,gamma);
            kDist = abs((2-(2*KK)).^(1/2));
            ti = ti+1; % update unchanging time

            [RP,I] = min(((exp(ti/lambda)).*kDist));

            D = dTh - RP;
            if(Yn == labelsTr(ii))
                if (D > 0) && (Yn == labTr(I))
                    ti(I) = 0;
                    RSS(I,:) = featMatTr(ii,:);
                    [R,alphas,betas] = unincLSSVM_test(KK,labelsTrRSS,featMatTr(ii,:),R,I,C
                        ,gamma); % Require upper Cholesky Factor
                end
            end
        end
        [~,prd] = predLSSVM(featMatTst,RSS,alphas,betas,gamma);
        CC_mdl = confusionmat(labelsTst,prd);

        C1 = CC_mdl.*eye(length(CC_mdl));
        C2 = sum(CC_mdl,2);
        acc3 = (sum(sum(C1,2)./C2)/(length(unique(labelsTr))))*100;
        if featMatID == 1
            accEMGIMU(jj) = acc3;    %Fused feats
            preds{2,jj} = [labelsTst,prd];
        else
            accEMG(jj) = acc3;   %EMG only feats
            preds{1,jj} = [labelsTst,prd];
        end
    end
end
end
```

#### B.5.2.1 *K* Medoids Code

```
function [idx] = kmedoids1(m,distMatrix)
% Code adapted from:
% H.-S. Park and C.-H. Jun, A simple and fast algorithm for k-medoids
% clustering," Expert Systems With Applications, vol. 36, no. 2,
% pp. 3336{3341, 2009.

% m = 4;
max_it = 10000;
didx = 1:size(distMatrix,1);

den = sum(distMatrix,2);
den = repmat(den,[1,size(distMatrix,2)]);
v = sum(distMatrix./den);

% Sort v in ascending order
[~,idx_medoid] = sort(v);

idx_medoid = sort(idx_medoid(1:m));
idx_medoid = idx_medoid(1:m);

d_medoid = distMatrix(:,idx_medoid);
[dist,idx] = min(d_medoid,[],2);

sum_dist = zeros(1,m);
for ii = 1:m
    sum_dist(:,ii) = sum(dist(idx==ii));
end

prev_sum_dist = sum_dist;

cont = 1;

% Step2
for jj = 1:max_it
    for ii = 1:m
        idx_aux = idx==ii;
        aux_mat = distMatrix(idx_aux,idx_aux);
        aux_didx = didx(:,idx_aux);
        [~,aux2] = min(sum(aux_mat));
        idx_medoid(ii) = aux_didx(aux2);
    end

    idx_medoid = sort(idx_medoid);
    d_medoid = distMatrix(:,idx_medoid);
    [dist,idx] = min(d_medoid,[],2);

    for ii = 1:m
        sum_dist(:,ii) = sum(dist(idx==ii));
    end

    if prev_sum_dist == sum_dist
        break;
    end

    cont = cont + 1;
    prev_sum_dist = sum_dist;

end

end
```

### B.5.2.2 Universal Incremental Learning Code

```
function [U,alphas,betas] = unincLSSVM_test(K2,S_labels,newDat,R,p,C,gamma)

% Code adapted from:
% Q. Huang, D. Yang, L. Jiang, H. Zhang, H. Liu, and K. Kotani,
% "A novel unsupervised adaptive learning method for long-term
% electromyography (EMG) pattern recognition", Sensors (Switzerland),
% vol. 17, no. 6, 2017.

%This function assumes we are using a RBF Kernel, code can be
%easily modified to use any type of kernel
%   S_labels = Representative sample set S labels.
%   K2 = Kernel Matrix of new testing sample newDat against representative
%        sample set S -> kernelFunction(newDat,S)
%   H = LS_SVM Kernel Matrix of RS (This kernel matrix already has the
%       regularization parameter C)
%   C = SVM regularization term (the same used for S)
%   p = index of particle in S to be replaced
%   newDat = Sample to replace the pth sample in S

l = size(S_labels,1);
W = zeros(l-1,l-1);
U = zeros(l,l);

h1 = (K2(1:p-1)).';
h2 = (K2(p+1:end)).';
```

```
hpp = (rbfKernel(newDat,newDat,gamma))+(1/C);

if p ~= 1
    W1 = R(1:p-1,1:p-1);
    W(1:size(W1,1),1:size(W1,2)) = W1;
    U(1:p-1,1:p-1) = W1;

    W2 = R(1:p-1,p+1:l);
    W(1:size(W2,1),size(W,2)-size(W2,2)+1:end) = W2;

    u1 = (W1.')\h1;
    upp = sqrt(hpp-((u1.')*u1));

    U(1:p-1,p) = u1;

    if ~isempty(W2)
        u2 = (h2 - (W2.')*u1)/upp;
        U(1:size(W2,1),size(U,2)-size(W2,2)+1:end) = W2;
    end
else
    upp = sqrt(hpp);
    u2 = h2/upp;
end


if p ~= l

    U(p,p+1:l) = u2.';

    W3 = R(p+1:l,p+1:l);
    r2 = (R(p,p+1:l)).';
    W3 = cholupdate(W3,r2);
    W(size(W,1)-size(W3,1)+1:end,size(W,2)-size(W3,2)+1:end) = W3;

    U3 = cholupdate(W3,u2,'-');

    U(size(U,1)-size(W3,1)+1:end,size(U,2)-size(W3,2)+1:end) = U3;

end
U(p,p) = upp;
```

```
% FIND ALPHAS AND BETAS USING MULTICLASS CLASSIFICATION
nLab = numel(unique(S_labels));
Yi = S_labels;
alphas = zeros(size(U,1),nLab);
betas = zeros(nLab,1);

Ev = ones(size(U,1),1);


for ii = 1:nLab
    Yi(S_labels==ii) = 1;
    Yi(S_labels~=ii) = -1;

    % Solve for eta (n) in H*eta = 1. Use the Cholesky factor of H
    eta = U\(U.'\Ev);

    % Solve for nu (v) in H*nu = y. Use the Cholesky factor of H
    nu = U\(U.'\Yi);

    betas(ii) = (Ev.'*nu)/(Ev.'*eta);
    alphas(:,ii) = nu - (eta*betas(ii));
end

end
```

### B.5.3 Adaptive LS-SVM Code

```
function [accEMG,accEMGIMU,preds] = lssvmAdapt(subjects,mdlID,cont,reduce)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here

%%
accEMGIMU = zeros(size(subjects));
accEMG = zeros(size(subjects));
preds = cell(2,numel(subjects));


for jj = 1:length(subjects)
    subID = subjects(jj);
    for featMatID = 1:2
        switch featMatID
            case 1
```

```
%        load(fullfile('Z:\Memo\Feature Matrices',sprintf('gestFeatures_S%d',subID)),...
%          'fusedFeats_all_Tr','fusedFeats_all_Tst','LSSVM_Mdl_EMGIMU_7_gest','LSSVM_Mdl_EMGIMU_NoPCA');
        load(fullfile('Z:\Memo\Feature Matrices',sprintf('gestFeatures_S%d',subID)),...
          'fusedFeats_all_Tr','fusedFeats_all_Tst','LSSVM_Mdl_EMGIMU','LSSVM_Mdl_EMGIMU_NoPCA');

        featMat = [fusedFeats_all_Tr; fusedFeats_all_Tst];

%        featMat = reduce_data(featMat);

        [~,feature_norm3] = featureNormalize(featMat(:,1:end-1));
%        [feature_norm3,~] = featureNormalize(featMat(:,1:end-1));
        if reduce == 1
%          mdlName = 'LSSVM_Mdl_EMGIMU_7_gest';
          mdlName = 'LSSVM_Mdl_EMGIMU';
        else
          mdlName = 'LSSVM_Mdl_EMGIMU_NoPCA';
        end
      case 2
%        load(fullfile('Z:\Memo\Feature Matrices',sprintf('gestFeatures_S%d',subID)),...
%          'emgFeats_all_Tr','emgFeats_all_Tst','LSSVM_Mdl_EMG_7_gest','LSSVM_Mdl_EMG_NoPCA');

        load(fullfile('Z:\Memo\Feature Matrices',sprintf('gestFeatures_S%d',subID)),...
          'emgFeats_all_Tr','emgFeats_all_Tst','LSSVM_Mdl_EMG','LSSVM_Mdl_EMG_NoPCA');

        featMat = [emgFeats_all_Tr; emgFeats_all_Tst];
%        featMat = reduce_data(featMat);

        [~,feature_norm3] = featureNormalize(featMat(:,1:end-1));
%        [feature_norm3,~] = featureNormalize(featMat(:,1:end-1));
        if reduce == 1
%          mdlName = 'LSSVM_Mdl_EMG_7_gest';
          mdlName = 'LSSVM_Mdl_EMG';
        else
```

```
                    mdlName = 'LSSVM_Mdl_EMG_NoPCA';
                end
            case 3
                feature_norm = featureNormalize(imuFeats2_all_Tr(:,1:end-1));
                feature_norm2 = featureNormalize(imuFeats2_all_Tst(:,1:end-1));

                featMat = [imuFeats2_all_Tr; imuFeats2_all_Tst];
                [~,feature_norm3] = featureNormalize(featMat(:,1:end-1));
                mdlName = 'LSSVM_Mdl_IMU';
        end

        % Data reduction
        if reduce == 1
            featPCA2 = projectData(feature_norm3,17);
        else
            featPCA2 = feature_norm3;
        end


        %--------------------------
        %test 3
        if featMatID == 1
            rng('shuffle')
            order = randperm(40);
            labels = featMat(:,end);
            reps = cell(1,40);

            idx1 = 1;
            idx2 = 1;
            idxTr = [];
            idxTst = [];

            for ii = 1:10
                aux1 = zeros(1,40);
                aux2 = ones(1,40);

                idxAux= find(labels == ii);
                temp = floor(length(idxAux)/40);
                aux2 = aux2*temp;
                aux1(1:length(idxAux)-(temp*40)) = 1;
                aux2 = aux2+aux1;

                sizeReps = cumsum(aux2).';
                sizeReps2 = sizeReps+1;
                sizeReps2 = [1; sizeReps2(1:end-1)];

                idx_temp = [sizeReps2 sizeReps];

                for kk = 1:40
                    reps{kk} = idxAux(idx_temp(kk,1):idx_temp(kk,2));
                end

                reps_temp = reps(order);
                for kk = 1:40
                    if kk <= cont % cont controls the number of repetitions from each gesture extracted for training
                        idxTr = [idxTr; reps_temp{kk}];
                    else
                        idxTst = [idxTst; reps_temp{kk}];
                    end
                end
            end
        end

        % Separate data
        featMatTr = featPCA2(idxTr,:);
        featMatTst = featPCA2(idxTst,:);
        labelsTr = labels(idxTr);
        labelsTst = labels(idxTst);
        %-------------------------

        gamma = 0.5;
        C = 2;

        %% Train data using different models
        k = length(mdlID);
        yHatCell = cell(k,2);

        % Kernel Matrix
        K = rbfKernel(featMatTr,featMatTr,gamma);
        H = K; H(1:size(K,1)+1:end) = H(1:size(K,1)+1:end)+(1/C);

        % Create vector of ones
        Ev = ones(size(H,1),1);

        % Compute M matrix
        M = [H, Ev;Ev.' 0];
        P = invBlockMat(M);

        % Compute Yhat vector for each model
        for ii = 1:k
            load(fullfile('Z:\Memo\Feature Matrices',sprintf('gestFeatures_S%d',mdlID(ii)))
                ,mdlName);
            [yHatCell{ii,1},yHatCell{ii,2}] = predLSSVM(featMatTr,'Model',eval(mdlName));
        end

        [~, ~, alphas, bs] = pSubGradDes(labelsTr,yHatCell,P,k);

        KK = rbfKernel(featMatTst,featMatTr,0.5);

        w = (alphas)*KK.';
        YY = w+bs;
        [~,p3]=max(YY);
        pred = p3.';
        CC_mdl = confusionmat(labelsTst,pred);
        C1 = CC_mdl.*eye(length(CC_mdl));
        C2 = sum(CC_mdl,2);
        acc3 = (sum(sum(C1,2)./C2)/(length(unique(labelsTr))))*100;

        if featMatID == 1
            accEMGIMU(jj) = acc3;    %Fused feats
            preds{2,jj} = [labelsTst,pred];
        else
            accEMG(jj) = acc3;   %EMG only feats
            preds{1,jj} = [labelsTst,pred];
        end
    end
end
end
```

### B.5.3.1 Inverse Block Matrix Code

```
function [P,R,S] = invBlockMat(M,varargin)
defaultDiagValue = 'False';
```

```
p = inputParser;
addRequired(p,'M');
addParameter(p,'Diagonal',defaultDiagValue);

%% DO NOT MODIFY
% parse(p,data,win_size,win_inc,varargin{:});
parse(p,M,varargin{:});

%% Assign parameters to variables here:
% var_name = p.Results.Parameter_Name
compDiagonal = p.Results.Diagonal;

%% Operations
A = M(1:end-1,1:end-1);
B = M(1:end-1,end);
C = M(end,1:end-1);
D = M(end,end);

% Cholesky factorization
R = chol(A,'lower');
S = inv(R);

Ap = S.'*S;
eta = R.'\(R\B);
SM = (-B).'*eta;

if (compDiagonal == "True")
    Sdiag = sum(S.^2);
    P = Sdiag.'+((eta.^2)./SM);
else
    Ainv = Ap+Ap*B/SM*C*Ap;
    Binv = -Ap*B/SM;
    Cinv = -SM\C*Ap;
    Dinv = inv(SM);
    P = [Ainv,Binv;Cinv,Dinv];
end

end
```

#### B.5.3.2 Projected Sub-Gradient Descent Code

```
function [bestb,Ytilde,alphas2,b] = pSubGradDes(y,Yhat,P,k)
```

```
%Projected Sub-gradient Descent Algorithm
%   Adapted from:
%   T. Tommasi, F. Orabona, C. Castellini, and B. Caputo, "Improving
%   control of dexterous hand prostheses using adaptive learning,"
%   IEEE Trans. Robot., vol. 29, no. 1, pp. 207-219, 2013.

% M is given by [H, Ev;Ev.' 0];

Pdiag = diag(P);

Adp = cell(k,1);
Adpcross = Adp;

betas = zeros(1,k);
prevBetas = betas;
t = 1;

y = y(:);
lab = unique(y);
G = length(lab);
N = length(y);

% Ensure Y is a [G, N] matrix, where G is the number of classes and N is
% the number of samples
Y = repmat(lab, [1 N]);
Y = Y==repmat(y.',[G 1]);
Y = double(Y);
Y(Y==0) = -1;

Zv = zeros(G,1);


Ap = ([Y, Zv])*(P.');
b = Ap(:,end);

% cellfun by itself uses a for loop to apply a function to each cell of the
% cell array
yHat = cellfun(@horzcat,Yhat(:,1),repmat(mat2cell(Zv,[length(Zv)],[1]),[size(Yhat,1),1]),'
    UniformOutput',false);

for ii = 1:k
```

```
    Adp{ii} = yHat{ii}*(P.');
    Adpcross{ii} = Adp{ii}./(Pdiag.');
    Adpcross{ii} = Adpcross{ii}(:,1:end-1); %remove bias term
end

Apcross = Ap(:,1:end-1)./(Pdiag(1:end-1).');

% ========================================================================
% Repeat until convergence
% ========================================================================
cont = 0;
bestd = realmax;
for jj = 1:10000
    Adpaux = zeros(size(Y));
    for ii = 1:k
        Adpaux = Adpaux + Adpcross{ii}.*betas(ii);
    end

    Ytilde = Y - Apcross + Adpaux;

    idxMat = 1:numel(Ytilde);
    idxMat = reshape(idxMat,size(Ytilde));
    idxMat2 = reshape(idxMat(Y~=1),G-1,N);

    % Compute gStar
    Ytildeaux = Ytilde(idxMat2);
    [~,argmax] = max(Ytildeaux);
    argmax2 = (0:size(Ytildeaux,1):numel(idxMat2));
    argmax2 = argmax+argmax2(1:end-1);
    gStar = idxMat2(argmax2);

    % Compute yi
    yi = idxMat(y.'+(0:size(Ytilde,1):numel(idxMat)-G));

    % Compute d
    d = sign(1-Ytilde(yi)+Ytilde(gStar)); %
    d(d<=0) = 0;
    normd = norm(d); % test

    if normd < bestd
        bestb=betas;
        if (bestd - normd) <= 0.05
            break
        end
        bestd = normd;
%         bestb = betas;
    end

    % obtain betas vector

    for ii = 1:k
        betas(ii) = betas(ii) - (1/sqrt(t))*sum(d.*(Adpcross{ii}(gStar)-Adpcross{ii}(yi)));
    end

    if norm(betas)>1
        betas = betas/norm(betas);
    end

    betas = max(betas,0);



    prevBetas = betas;
    t = t+1;
    cont = cont+1;
end

al = zeros(size(Ap,1),size(Ap,2)-1);

for ii = 1:k
    al = al+(bestb(ii).*Adp{ii}(:,1:end-1));
end

alphas2 = Ap(:,1:end-1) - al;
end
```

### B.5.4 Bilinear Model Codes

```
%% Setup
clc; clear ; close all;

winSize = .250; %segment's windows size in s
winOverlap = .125; %window overlap (about 50% of windows size) in s
```

```
deadzone = 0.02;

%% Read Data

subjectID = [2:7 9:20 22:25];

for gg = subjectID
    dataFile = fullfile('Z:\Memo\Data\',sprintf('S_%d',gg),sprintf('expData_S%d.mat',gg));

    workspaceVariable = sprintf('gestFeatures_S%d_bm',gg);
    dataFullFileName = fullfile('Z:\Memo\Feature Matrices\Bilinear Models Features2',workspaceVariable);

    % CLEAR VALUES HERE
    featsEMG = [];
    featsIMU = [];
    gest_labels = [];

    load(dataFile);
    for gestName = ["Wrist Flexion","Wrist Extension","Wrist Pronation","Wrist Supination","Wrist Aduction","Wrist Abduction","Hand Fist","Hand Open","Precision Pinch","Key Pinch"]

        switch gestName
            case "Wrist Flexion"
                gesture = 1;
            case "Wrist Extension"
                gesture = 2;
            case "Wrist Pronation"
                gesture = 3;
            case "Wrist Supination"
                gesture = 4;
            case "Wrist Aduction"
                gesture = 5; % Radial Deviation
            case "Wrist Abduction"
                gesture = 6; % Ulnar Deviation
            case "Hand Fist"
                gesture = 7;
            case "Hand Open"
                gesture = 8;
            case "Precision Pinch"
```

```
                gesture = 9;
            case "Key Pinch"
                gesture = 10;
        end

        for kk = 1:4 % Arm Position
            order1 = randperm(10); % Randomize order of repetitions

            for ii = order1 % Repetition
                if ~isempty(dataExp{ii,gesture,kk}) %gesture
                    timeEMG_log = dataExp{ii,gesture,kk}{1,1};
                    emg_log = dataExp{ii,gesture,kk}{1,2};
                    timeIMU_log = dataExp{ii,gesture,kk}{1,3};
                    imu_log = dataExp{ii,gesture,kk}{1,4};
                end
                [feats,~,~,feats_imu2,labels,actarea,actareaimu] = main_loop_bm(gesture,timeEMG_log,emg_log,timeIMU_log,imu_log,winSize,winOverlap,deadzone);

                featsEMG = [featsEMG;feats];
                featsIMU = [featsIMU;feats_imu2];
                gest_labels = [gest_labels;labels];
            end
        end
    end

    emgFeats = [featsEMG,gest_labels];
    imuFeats = [featsIMU,gest_labels];
    fusedFeats = [featsEMG,featsIMU,gest_labels];

    size(emgFeats)

    bm_Mdl.emgFeats = emgFeats;
    bm_Mdl.imuFeats = imuFeats;
    bm_Mdl.fusedFeats = fusedFeats;

    save([dataFullFileName '.mat'],'bm_Mdl');
end
```

```
function [features, featuresIMU, featuresIMU1, featuresIMU2, labels, active_area, active_areaIMU] = main_loop_bm(gesture,timeEMG,channels,timeIMU,imu_log,winSize,winOverlap,deadzone)
```

```
%UNTITLED4 Summary of this function goes here
%   Detailed explanation goes here

noSegmentation = 0;

if(nargin < 8)
    deadzone = 0.02;
    if (nargin < 7)
        winOverlap = .125;
        if (nargin < 6)
            noSegmentation = 1;
        end
    end
end


test_time = ceil(length(timeEMG)/200); %time duration of data acquisition in seconds
fs = length(timeEMG)/test_time; %real sampling frequency UNCOMMENT LINE

test_time_IMU = ceil(length(timeIMU)/50); %time duration of data acquisition in seconds
fs_IMU = length(timeIMU)/test_time_IMU; %real sampling frequency UNCOMMENT LINE

%% Conditioning

gyro = imu_log(:,5:7);
acc = imu_log(:,8:10);

signal = [gyro,acc];

% Because the timestamp on both timeEMG and timeIMU are different, we have
% to make them similar by substracting the first value from timeEMG and
% timeIMU from their respective vectors
timeEMG = timeEMG - timeEMG(1);
timeIMU = timeIMU - timeIMU(1);


%% Preprocessing
emgsignal = emgFilter(channels, fs); % EMG filter
IMU_signal_filt = imuFilter(signal,fs_IMU); % IMU filter (Remove 'gravity' (DC component) from the acceleration data)
```

```
% =====================================================================
% We will have to upsample the IMU signal so we can later fuse the
% feature vectors

% Upsample method one: repeat sample values
imuUp1 = (IMU_signal_filt(:)).';
imuUp1 = reshape(repmat(imuUp1,[4 1]),size(IMU_signal_filt,1)*4,size(IMU_signal_filt,2));

% Upsample method two: interpolate using a cubic spline
imuUp2 = interp1(timeIMU,IMU_signal_filt,timeEMG,'spline');


onEMG = 800:1650;
active_area = [onEMG(1);onEMG(end)];


emgsignal_active = emgsignal(onEMG,:);

idx = timeEMG(active_area);
active_areaIMU = zeros(size(active_area));

for ii = 1:length(idx)
    temp = find(timeIMU >= idx(ii));
    active_areaIMU(ii) = temp(1);
end


cont = (active_areaIMU(2:2:end)-active_areaIMU(1:2:end-1))+1;
check1 = reshape(active_areaIMU,2,[]);

rows_size = sum(cont);
idx1 = cumsum(cont);

onIMU = zeros(rows_size,1);

temp1 = 1;

for ii = 1:size(check1,2)
    onIMU(temp1:idx1(ii)) = check1(1,ii):check1(2,ii);
    temp1 = idx1(ii)+1;
end
```

```

IMU_active = IMU_signal_filt(onIMU,:);  % Non upsampled signal
IMU_active1 = imuUp1(onEMG,:);          % Upsampled signal method 1. We use same active area as EMG b/c both signals have the same sampling rate now
IMU_active2 = imuUp2(onEMG,:);          % Upsampled signal method 2. We use same active area as EMG b/c both signals have the same sampling rate now


%% Feature Extraction
if isempty(emgsignal_active)
    features = [];
    labels = [];
    active_area = [0;0];
    active_areaIMU = [0;0];
    return
end
if ~noSegmentation
    % ====================================================================
    winIncrement = winSize-winOverlap;
    win_size = ceil(winSize*200); % Myo EMG FS
    win_inc = ceil(winIncrement*200); % Myo EMG FS
    win_sizeIMU = ceil(winSize*50); % Myo IMU FS
    win_incIMU = ceil(winIncrement*50); % Myo IMU FS

    features = extract_feature(emgsignal_active,'winSize',win_size,'winInc',win_inc,'Features',["MAV" "MAVS" "WL" "ZC" "AR"]);%["RMS" "AR"]
    featuresIMU = extract_feature(IMU_active,'winSize',win_sizeIMU,'winInc',win_incIMU,'Features',["MAV" "WL"]);
    featuresIMU1 = extract_feature(IMU_active1,'winSize',win_size,'winInc',win_inc,'Features',["MAV" "WL"]); % We use the same parameters as with the EMG signal b/c both signals have the same sampling rate now
    featuresIMU2 = extract_feature(IMU_active2,'winSize',win_size,'winInc',win_inc,'Features',["MAV" "WL"]); % We use the same parameters as with the EMG signal b/c both signals have the same sampling rate now
else
    features = extract_feature(emgsignal_active,'Features',["MAV" "WL" "ZC" "AR"]);
    featuresIMU = extract_feature(IMU_active,'Features',["MAV" "WL"]);
    featuresIMU1 = extract_feature(IMU_active1,'Features',["MAV" "WL"]); % We use the same parameters as with the EMG signal b/c both signals have the same sampling rate now
    featuresIMU2 = extract_feature(IMU_active2,'Features',["MAV" "WL"]); % We use the same parameters as with the EMG signal b/c both signals have the same sampling rate now
```

```
end

labels = ones(size(features,1),1).*gesture;

end
```

#### B.5.4.1 Learn Bilinear Models Code

```
function [W,WVT,WVTZ,X,Z,cont] = bmLearn(Y,U,K,M,N,I,J)
%Bilinear Models Function
%   Y = Stacked matrix of Feature Matrices
%   U = Number of users
%   K = Number of features
%   M = Number of motions/gestures
%   N = Number of samples per gesture
%   I = Desired first dimension of style and content variables
%   J = Desired second dimension of style and content variables

% % U = 6;  % Number of users in our training data (we know this value); 10
% K = size(Y,1)/U;  % Number of channels being used
% M = 10;  % Number of motions (we know this value)
% N = size(Y,2)/M; % Number of samples of each motion

YVT = vectTrans2(Y,K);

MN = size(YVT,1)/K;

[~,~,V] = svd(Y);
vt = V.';
X = vt(1:J,:);
Xb = X;
Zb = zeros(I,U);

cont = 0;
for ii = 1:100000000

%----------------------
% Find Z
mat1 = Y*X.';
% mat1VT = vectTrans(mat1,U,J,K);
mat1VT = vectTrans2(mat1,K);

```

```
[~,~,V] = svd(mat1VT);
vt = V.';
Z = vt(1:I,:);
%-----------------------
% Find X
mat2 = YVT*Z.';


%===================== GPU code =====================
existGPU = gpuDeviceCount;
if existGPU > 0
    dev = gpuDevice(1);
    if dev.AvailableMemory > (numel(mat2)*8)
        mat2VT = gpuArray(vectTrans2(mat2,K));
        [~,~,vv] = svd(mat2VT);
        V = gather(vv);
        dev = gpuDevice(1); % Clear GPU memory
    else
        mat2VT = vectTrans2(mat2,K);
        [~,~,V] = svd(mat2VT);
    end
else
    mat2VT = vectTrans2(mat2,K);
    [~,~,V] = svd(mat2VT);
end
%=================== end GPU code ===================

vt = V.';
X = vt(1:J,:);

cont = cont+1;

if (norm(X,'fro')-norm(Xb,'fro') == 0) && (norm(Z,'fro')-norm(Zb,'fro') == 0)
    break;
end

Xb = X;
Zb = Z;
end

mat1 = Y*X.';
```

```
YXtVT = vectTrans2(mat1,K);


WVT = (YXtVT*Z.');
W = vectTrans2(WVT,K);

WVTZ = vectTrans2(WVT*Z,K);

end
```

#### B.5.4.2 Style and Content Separation Code

```
clear ; clc;
load('cvAccWithPCA.mat')
testDat = cell(5,1);
cvModels2 = cvModels;
reduce = 1; % 0 or 1
tic
for gg = 1:5
    ids = cvModels{gg}{1};
    stackedY_emg = [];
    stackedY_imu = [];
    for subID = ids
        load(fullfile('Z:\Memo\Feature Matrices\Bilinear Models Features',sprintf('gestFeatures_S%d_bm',subID)),'bm_Mdl');

        featMatEMG = bm_Mdl.emgFeats;
        featMatIMU = bm_Mdl.imuFeats;

        if reduce
            featMatEMG = reduce_data(featMatEMG);
            featMatIMU = reduce_data(featMatIMU);
        end

        featPCA2 = featMatEMG(:,1:end-1); % TEST LINE
        featPCA2_imu = featMatIMU(:,1:end-1); % TEST LINE

        % Transpose featPCA2 so that the features are in the rows and the samples are in the columns
        stackedY_emg = [stackedY_emg;featPCA2.'];
        stackedY_imu = [stackedY_imu;featPCA2_imu.'];
    end

    gest = numel(unique(featMatEMG(:,end)));
    [W,WVT,WVTZ,X,Z,cont] = bmLearn(stackedY_emg,numel(ids),size(featPCA2,2),gest,(size(featPCA2,1)/gest),2,3);

    yy = stackedY_imu.';
    YY = zeros(size(yy,1),12);
    for ii=1:12
        YY(:,ii) = mean(yy(:,ii:12:end-(12-ii)),2);
    end

    Mdl_bm.X = X; %featPCA2
    Mdl_bm.W = W;
    Mdl_bm.Z = Z;
    Mdl_bm.IMU = YY;
    Mdl_bm.Convergence = cont;
    Mdl_bm.labels = featMatEMG(:,end);

    testDat{gg} = Mdl_bm;
end
toc
fullfile_name = fullfile('Z:\Memo\Feature Matrices\Bilinear Models Features',sprintf('bm_Content_Variables_%d_gest.mat',gest));
save(fullfile_name,'testDat');
```

### B.5.4.3 Vector Transpose Code

```
function [vectdataVT] = vectTrans2(vectdata,K)
%Returns a matrix in the form of: [K*J,I]
%   I = Desired columns
%   J = Desired rows
%   K = Row multiplier

U = size(vectdata,1)/K;
C = size(vectdata,2);

idxvector = 1:numel(vectdata); % Create a vector which values range from 1 to the # of elements in vectdata
idxvector = reshape(idxvector,size(vectdata)); % Reshape the vector into a matrix of the same size as vectdata

auxMat = mat2cell(vectdata,ones(1,U).*K,ones(1,C));
```

```
vectdataVT = cell2mat(auxMat');

end
```

## B.6 General Purpose Codes

### B.6.1 Reduce Number of Gestures Code

```
function newData = reduce_data(data,varargin)
% Remove desired gestures from data. Gesture inputs must be in a string
% format and inside a vector. If no gestures are used as an input, the
% function will remove the WAD, WAB, and PP gestures
% Example:
%   newData = reduce_data(data,["WAD","WAB","PP"])
%
% Valid inputs:
%   WF      Wrist Flexion
%   WE      Wrist Extension
%   WP      Wrist Pronation
%   WS      Wrist Supination
%   WAD     Wrist Aduction (Default gesture to be removed)
%   WAB     Wrist Abduction (Default gesture to be removed)
%   HF      Hand Fist
%   HO      Hand Open
%   PP      Precision Pinch (Default gesture to be removed)
%   KP      Key Pinch

defaultGestures = ["WAD" "WAB" "PP"];

p = inputParser;
addRequired(p,'data');
addParameter(p,'Gestures',defaultGestures);

%% DO NOT MODIFY
% parse(p,data,win_size,win_inc,varargin{:});
parse(p,data,varargin{:});

%% Assign parameters to variables here:
% var_name = p.Results.Parameter_Name
gests_to_remove = p.Results.Gestures;

```

```
%% Main

% Substitute labels of gestures to be removed with zeros
gests = 1:10;
labels = data(:,end);
for ii = gests_to_remove
    switch ii
        case "WF"
            labels(labels == gests(1)) = 0;
        case "WE"
            labels(labels == gests(2)) = 0;
        case "WP"
            labels(labels == gests(3)) = 0;
        case "WS"
            labels(labels == gests(4)) = 0;
        case "WAD"
            labels(labels == gests(5)) = 0;
        case "WAB"
            labels(labels == gests(6)) = 0;
        case "HF"
            labels(labels == gests(7)) = 0;
        case "HO"
            labels(labels == gests(8)) = 0;
        case "PP"
            labels(labels == gests(9)) = 0;
        case "KP"
            labels(labels == gests(10)) = 0;
    end
end

% Remove data whose label is zero from the dataset
datEMGIMU_17 = data(labels~=0,:);

% Make the labels of the dataset sequential
aux1 = datEMGIMU_17(:,end);
gestures = unique(aux1);
temp = diff(gestures);
aux = find(temp>1)+1;

for ii = aux(1):length(gestures)
    aux1(aux1==gestures(ii))=ii;
```

```
end

datEMGIMU_17(:,end) = aux1;
newData = datEMGIMU_17;

end
```

# Appendix C

# Mathematical Formulations

## C.1 Least Squares Support Vector Machines

The optimization problem of the LS-SVM is given by the following equation:

$$\begin{aligned} &\min_{w,b} \frac{1}{2}\|w\|^2 + \frac{\mathcal{C}}{2}\sum_{i=1}^{N}\xi_i^2 \\ &\text{subject to } y_i = w \cdot \phi(x_i) + b + \xi_i \qquad i = 1, ..., N, \end{aligned} \tag{C.1}$$

where $w$ is a weight vector, $x_i$ is the feature vector of the $i^{th}$ training sample, $\phi(\cdot)$ is the non-linear function that maps the samples of $x_i$ to a high dimensional space, $b$ is the bias term, $\xi$ is the LS-SVM slack variable that relaxes the optimization constraints, and $y_i$ is the $i^{th}$ training sample label. After solving the Lagrangian for Equation (C.1), the following LS-SVM optimal conditions are obtained:

$$\frac{\partial}{\partial w}\mathcal{L}(w, b, \xi) = 0 \quad \Rightarrow \quad w = \sum_{i=1}^{N}\alpha_i \cdot \phi(x_i) \tag{C.2}$$

$$\frac{\partial}{\partial b}\mathcal{L}(w, b, \xi) = 0 \quad \Rightarrow \quad \sum_{i=1}^{N}\alpha_i = 0 \tag{C.3}$$

$$\frac{\partial}{\partial \xi_i}\mathcal{L}(w, b, \xi) = 0 \quad \Rightarrow \quad \mathcal{C}\xi_i = \alpha_i \tag{C.4}$$

$$\frac{\partial}{\partial \alpha_i}\mathcal{L}(w,b,\xi)=0 \quad \Rightarrow \quad \sum_{i=1}^{N} w\cdot\phi(x_i)+b+\xi_i-y_i=0. \tag{C.5}$$

Substituting Equations (C.2) to (C.4) into Equation (C.5) to eliminate $w$ and $\xi$, the following equation is found:

$$\sum_{i,j=1}^{N} \alpha_j\langle\phi(x_i),\phi(x_j)\rangle+\frac{\alpha_i}{\mathcal{C}}+b-y_i=0, \tag{C.6}$$

where $\langle\cdot\rangle$ represent the dot product between $\phi(x_i)$ and $\phi(x_j)$. Equation (C.6) can then be represented in the form of linear equations, as follows:

$$\begin{bmatrix} \mathcal{K}+\frac{I}{\mathcal{C}} & \vec{1} \\ \vec{1}^T & 0 \end{bmatrix} \cdot \begin{bmatrix} \alpha \\ b \end{bmatrix} = \begin{bmatrix} y \\ 0 \end{bmatrix}, \tag{C.7}$$

where $\vec{1}$ represents a vector of 1's, $I$ is the identity matrix, $\alpha$, $b$ and $\mathcal{C}$ are the LS-SVM parameters, and $\mathcal{K}$ is the kernel matrix with entries $\mathcal{K}_{i,j}=\mathcal{K}(x_i,x_j)=\langle\phi(x_i),\phi(x_j)\rangle$, being $\mathcal{K}(x_i,x_j)$ the kernel function.

## C.2 Adaptive LS-SVM

This section describes the mathematical formulations and derivations of the Adaptive LS-SVM classification method proposed by Tommasi *et al.* [81]. This classification method aims to solve the following optimization problem:

$$\begin{gathered} \min_{w,b}\frac{1}{2}\left\|w-\beta\hat{w}\right\|^2+\frac{\mathcal{C}}{2}\sum_{i=1}^{N}\xi_i^2 \\ \text{subject to } y_i=w\cdot\phi(x_i)+b+\xi_i \qquad i=1,...,N, \end{gathered} \tag{C.8}$$

where $\beta$ is a scaling factor that weighs a pretrained model $\hat{w}$. Solving the Lagrangian for Equation (C.8) yields:

$$\sum_{i,j=1}^{N} \alpha_j \langle \phi(x_i), \phi(x_j) \rangle + \frac{\alpha_i}{\mathcal{C}} + b = y_i - \beta \hat{w} \cdot \phi(x_i). \tag{C.9}$$

From Equation (C.2), the dot product $[\hat{w} \cdot \phi(x_i)]$ in Equation (C.9) can be rewritten as $\hat{y}$, which is the prediction on a new sample using a previous trained LS-SVM model. Thus, the system of linear equations in Equation (C.7) changes to:

$$M \cdot \begin{bmatrix} \alpha \\ b \end{bmatrix} = \begin{bmatrix} y - \beta \hat{y} \\ 0 \end{bmatrix}, \tag{C.10}$$

where $M$ is the first matrix to the left in Equation (C.7), and $y$ is the vector that contains the label samples. From Equation (C.10), the parameters $\alpha$ and $b$ are obtained, as follows:

$$\begin{bmatrix} \alpha \\ b \end{bmatrix} = P \cdot \begin{bmatrix} y - \beta \hat{y} \\ 0 \end{bmatrix}, \tag{C.11}$$

where $P = M^{-1}$. Finally, following the same procedure in [98], and using Equation (C.11) a closed form solution for the leave-one-out prediction $\tilde{y}_i$ on sample $i$ when removed from the training set is given by:

$$\tilde{y}_i = y_i - \frac{\alpha_i}{P_{ii}}. \tag{C.12}$$

Having $\alpha = \alpha' - \beta \alpha''$, the leave-one-out prediction is then given by:

$$\tilde{y}_i = y_i - \frac{\alpha'_i}{P_{ii}} + \beta \frac{\alpha''_i}{P_{ii}}, \tag{C.13}$$

where $\alpha'$ and $\alpha''$ are given by:

$$[\alpha'^T, b']^T = P \cdot [y^T, 0]^T \tag{C.14}$$

$$[\alpha''^T, b'']^T = P \cdot [\hat{y}^T, 0]^T. \tag{C.15}$$

### C.2.1 Adaptive LS-SVM From Multiple Subjects

A variation of the Adaptive LS-SVM consists of using more than one parameter from previous pretrained models to construct a new learning problem. For this case, Equation (C.8) changes so that the new optimization problem becomes:

$$\begin{aligned} &\min_{w,b} \frac{1}{2} \left\| w - \sum_{k=1}^{K} \vec{\beta}^{(k)} \vec{w}^{(k)} \right\|^2 + \frac{\mathcal{C}}{2} \sum_{i=1}^{N} \xi_i^2 \\ &\text{subject to } y_i = w \cdot \phi(x_i) + b + \xi_i \qquad i = 1, ..., N. \end{aligned} \tag{C.16}$$

The main difference between Equation (C.8) and Equation (C.16) is that the coefficient $\beta$ and the parameter $\hat{w}$ in Equation (C.8) are replaced with new vectors $\vec{\beta}$, whose number of elements are equal to $K$ pretrained models, and $\vec{w}$, which is a vector of matrices containing the pretrained parameters $w$ of previous $K$ models, respectively. The superscript $(k)$ in Equation (C.16) indicates the $k^{th}$ element of $\vec{\beta}$ and $\vec{w}$. Furthermore, for a new classification model, the parameter $w$ is given by the weighted sum of the pretrained parameters $\vec{w}^{(k)}$ of previous models scaled by its respective coefficient $\vec{\beta}^{(k)}$, plus the new model built on incoming new training data, as follows:

$$w = \sum_{k=1}^{K} \vec{\beta}^{(k)} \vec{w}^{(k)} + \sum_{i=1}^{N} \alpha_i \phi(x_i). \tag{C.17}$$

As before, it can be seen that by removing the first summation term in Equation (C.17), *i.e.*, by making all elements in $\vec{\beta}$ equal to 0, the original LS-SVM formulation is recovered.

# Appendix D

# Python Code

## D.1 Real-Time EMG Data Streaming

```
# The MIT License (MIT)
#
# Copyright (c) 2017 Niklas Rosenstein
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to
# deal in the Software without restriction, including without limitation the
# rights to use, copy, modify, merge, publish, distribute, sublicense, and/or
# sell copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in
# all copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
# FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
# IN THE SOFTWARE.
#
# Author: Jose Guillermo Colli Alfaro
# Date: October 2, 2018
# Version: 1.0

from matplotlib import pyplot as plt
from collections import deque
from threading import Lock, Thread

import myo
import numpy as np


class EmgCollector(myo.DeviceListener):
  """
  Collects EMG data in a queue with *n* maximum number of elements.
  """

  def __init__(self, n):
    self.n = n
    self.lock = Lock()
    self.emg_data_queue = deque(maxlen=n)

  def get_emg_data(self):
    with self.lock:
      return list(self.emg_data_queue)

  # myo.DeviceListener

  def on_connected(self, event):
    event.device.stream_emg(True)

  def on_emg(self, event):
    with self.lock:
      self.emg_data_queue.append((event.timestamp, event.emg))


class Plot(object):

  def __init__(self, listener):
    self.n = listener.n
    self.listener = listener
    self.fig = plt.figure()
    self.axes = self.fig.add_subplot(’111’)
    self.axes.set_ylim([0, 1000])
    self.graph = self.axes.plot(np.arange(self.n), np.zeros(self.n))[0]
    #Insert code for transparent background in plot
    plt.ion()

  def update_plot(self):
    emg_data = self.listener.get_emg_data()
    emg_data = np.array([x[1] for x in emg_data]).T
    emg_data2 = np.sum(np.abs(emg_data),axis = 0)
    #for data in emg_data2:
        #print(len(data))

    #print(emg_data2.size)
    if emg_data2.size < self.n:
     # Fill the left side with zeroes.
        data = np.concatenate([np.zeros(self.n - emg_data2.size), emg_data2], axis=None)
    else:
        data = emg_data2
    self.graph.set_ydata(data)
    plt.draw()

  def main(self):
    while True:
      self.update_plot()
      plt.pause(1.0 / 30)


def main():
  myo.init(’C:/Users/jcollial/Google Drive/UWO/Masters Project/Myo/’
           ’Real Time Plot/PythonApplication2/myo-sdk-win-0.9.0/bin/myo32.dll’)
  hub = myo.Hub()
  listener = EmgCollector(512)
  with hub.run_in_background(listener.on_event):
    Plot(listener).main()


if __name__ == ’__main__’:
  main()
```

## D.2 Classification Using Bilinear EMG Models

```python
import numpy as np
from numpy import array
import time
import pandas as pd        # For loading and processing the dataset
import tensorflow as tf    # Of course, we need TensorFlow.
from sklearn.model_selection import train_test_split
import os

from collections import deque
import random
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, LSTM, CuDNNLSTM, \
    BatchNormalization, Bidirectional
from tensorflow.keras.callbacks import TensorBoard
from tensorflow.keras.callbacks import ModelCheckpoint, ModelCheckpoint
from sklearn import preprocessing
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler,StandardScaler
from tensorflow.keras.callbacks import TensorBoard
from tensorflow.keras.callbacks import ModelCheckpoint, ModelCheckpoint


# NAME = "Classification-{}".format(int(time.time()))

data = pd.read_csv('Z:/Memo/BM Python Code/Data/CV5/DatEMGIMU5_7_gest.csv')  # Load EMG Data

x_trainEMG = data.drop(['f4','f5','f6','f7','f8','f9','f10','f11',
                        'f12','f13','f14','f15','label'], axis=1).values
x_trainEMGIMU = data.drop('label', axis=1).values
y_train = data['label'].values
y_train = y_train - 1  # Subtract 1 from the labels so that they start at 0

# Select optimizer
opt = tf.keras.optimizers.Adam(lr=0.001, decay=1e-6)

# =============================== Custom Activation Functions ===============================


def my_leaky_relu(features):
    ’’’This function allows to modify the default alpha value of
    the tf.nn.leaky_relu activation function’’’
    return tf.nn.leaky_relu(features, alpha=0.01)


# ======================================== EMG Model ========================================
sc = StandardScaler()
x_trainEMG_scaled=sc.fit_transform(x_trainEMG)
modelEMG = Sequential()
modelEMG.add(Dense(50, activation=tf.nn.relu, input_shape=(x_trainEMG_scaled.shape[1:])))
modelEMG.add(Dropout(0.2))
modelEMG.add(BatchNormalization())

# modelEMG = Sequential()
# modelEMG.add(Dense(50, activation=tf.nn.relu, input_shape=(x_trainEMG.shape[1:])))
# modelEMG.add(Dropout(0.2))
# modelEMG.add(BatchNormalization())

modelEMG.add(Dense(20, activation=tf.nn.relu))
modelEMG.add(Dropout(0.2))

modelEMG.add(Dense(7, activation=’softmax’))

# Compile EMG model
modelEMG.compile(
    loss=’sparse_categorical_crossentropy’,
    optimizer=opt,
    metrics=[’accuracy’]
)

# Train EMG model
historyEMG = modelEMG.fit(
    x_trainEMG, y_train,
    batch_size=256,
    epochs=300,
    # validation_data=(x_test_scaled, y_test),
    # validation_split = 0.2,

)

# plt.plot(historyEMG.history[’acc’])
# plt.title('model accuracy')
# plt.ylabel('accuracy')
# plt.xlabel('epoch')
# plt.legend(['train', 'test'], loc='upper left')
# plt.show()
# # summarize history for loss
# plt.plot(historyEMG.history['loss'])
# plt.title('model loss')
# plt.ylabel('loss')
# plt.xlabel('epoch')
# plt.legend(['train', 'test'], loc='upper left')
# plt.show()

# ====================================== EMG+IMU Model ======================================
sc = StandardScaler()

x_trainEMGIMU_scaled=sc.fit_transform(x_trainEMGIMU)

modelEMGIMU = Sequential()
modelEMGIMU.add(Dense(50, activation= tf.nn.relu,
                      input_shape=(x_trainEMGIMU_scaled.shape[1:])))  # 50
modelEMGIMU.add(Dropout(0.2))
modelEMGIMU.add(BatchNormalization())

modelEMGIMU.add(Dense(20, activation= tf.nn.relu))  # 20
modelEMGIMU.add(Dropout(0.2))

# modelEMGIMU.add(Dense(15, activation= tf.nn.leaky_relu))  # 20
# modelEMGIMU.add(Dropout(0.2))

modelEMGIMU.add(Dense(7, activation='softmax'))

# Compile EMG+IMU model
modelEMGIMU.compile(
    loss='sparse_categorical_crossentropy',
    optimizer=opt,
    metrics=['accuracy']
)

# Train EMG+IMU model
historyEMGIMU = modelEMGIMU.fit(
    x_trainEMGIMU_scaled, y_train,
    batch_size=256,
    epochs=300,
    # validation_data=(x_test_scaled, y_test),
    # validation_split = 0.2,

)

# plt.plot(historyEMGIMU.history['acc'])
# plt.title('model accuracy')
# plt.ylabel('accuracy')
# plt.xlabel('epoch')
# plt.legend(['train', 'test'], loc='upper left')
# plt.show()
# # summarize history for loss
# plt.plot(historyEMGIMU.history['loss'])
# plt.title('model loss')
# plt.ylabel('loss')
# plt.xlabel('epoch')
# plt.legend(['train', 'test'], loc='upper left')
# plt.show()

# ===========================================================================================
# ========================================  PREDICT  ========================================
# ===========================================================================================
datasetEMG_test = pd.read_csv('Z:/Memo/BM Python Code/Data/CV5/'
                              'CV5_bm_testdataEMG4_7_gests.csv')
datasetEMGIMU_test = pd.read_csv('Z:/Memo/BM Python Code/Data/CV5/'
                                 'CV5_bm_testdataEMGIMU4_7_gests.csv')

x_testEMG = datasetEMG_test.drop('label', axis=1).values
x_testEMGIMU = datasetEMGIMU_test.drop('label', axis=1).values

y_test = datasetEMG_test['label'].values
y_test = y_test - 1  # Subtract 1 from the labels so that they start at 0

# ======================================== EMG Model ========================================
sc = StandardScaler()
x_testEMG_scaled=sc.fit_transform(x_testEMG)
predictionsEMG = modelEMG.predict([x_testEMG_scaled])

```

```
# predictionsEMG = modelEMG.predict([x_testEMG])

y_maxPre_EMG= np.argmax(predictionsEMG, axis=1)


from sklearn.metrics import confusion_matrix, recall_score, precision_score, f1_score, \
    classification_report
confusion_matrix(y_test, y_maxPre_EMG)
print(classification_report(y_test, y_maxPre_EMG))
c_Mat_EMG = confusion_matrix(y_test, y_maxPre_EMG)

# ====================================== EMG+IMU Model ======================================
sc = StandardScaler()

x_testEMGIMU_scaled=sc.fit_transform(x_testEMGIMU)
# x_testEMGIMU_scaled=sc.transform(x_testEMGIMU)

predictionsEMGIMU = modelEMGIMU.predict([x_testEMGIMU_scaled])

y_maxPre_EMGIMU= np.argmax(predictionsEMGIMU, axis=1)


from sklearn.metrics import confusion_matrix, recall_score, precision_score, f1_score, \
    classification_report
confusion_matrix(y_test, y_maxPre_EMGIMU)
print(classification_report(y_test, y_maxPre_EMGIMU))
c_Mat_EMGIMU = confusion_matrix(y_test, y_maxPre_EMGIMU)

# =================================== Save Models Results ===================================
df1 = pd.DataFrame(c_Mat_EMG)
df2 = pd.DataFrame(c_Mat_EMGIMU)

writer = pd.ExcelWriter(’Z:/Memo/BM Python Code/Data/CV5/’
                        ’CV5_bm_7_gests_Results_S15.xlsx’, engine = ’xlsxwriter’)

df1.to_excel(writer, sheet_name = ’EMG’)
df2.to_excel(writer, sheet_name = ’EMGIMU’)

writer.save()
```

# Appendix E

# R Code

## E.1 Train MLP

```
################
#   Library
################

library(RSNNS)
library(DMwR)
#####################
#    load data
#####################

data <- read.csv("D:/Memo/R stuff/Model EMG+IMu/data7gest.csv")

##################
#   Split Data
##################

sample <- sample.int(n=nrow(data), size = floor(0.2*nrow(data)))
train <- data[-sample,]
test <- data[sample,]

################################################################
#                   Scaling
################################################################

```

```
colMeans = colMeans(train)#column means
col_stdev <- apply(train, 2, sd) #standard deviation

trainNorm <- scale(train, center=colMeans, scale=col_stdev )
trainNorm[is.nan(trainNorm)] = 0

testNorm <- scale(test,  center=colMeans, scale=col_stdev)
testNorm[is.nan(testNorm)] = 0

forUnscaleOutput <- scale(train[, ncol(train)])


####################################################################################
####################################################################################

###MLP - NN

start.time<- Sys.time() ##### start time

model7Gest<-RSNNS::mlp(trainNorm[,1:76], trainNorm[,77],
                       size = c(300,200,100), maxit = 100,
                       learnFunc="BackpropMomentum",linOut = TRUE)


end.time<-Sys.time() ##### end time
time.taken <- round(end.time - start.time,2)
time.taken

predicted <-predict(model7Gest, testNorm[,1:76], testNorm[,77])

predicted <- DMwR::unscale(predicted,forUnscaleOutput)


##############################################
####      Confusion Matrix
##############################################

result.nnet <- data.frame(actual = test$label, prediction = predicted)
roundedresults.nnet<-sapply(result.nnet,round,digits=0)
roundedresultsdf.nnet = data.frame(roundedresults.nnet)

```

```
for (i in 1:length(roundedresultsdf.nnet$prediction))
{
  if (roundedresultsdf.nnet$prediction[i]<1)
  {
    roundedresultsdf.nnet$prediction[i] = 1
  }
  else if(roundedresultsdf.nnet$prediction[i]>7){
    roundedresultsdf.nnet$prediction[i] = 7
  }
}

attach(roundedresultsdf.nnet)
table(actual,prediction)

table7gest <- table(actual, prediction)
write.csv(table7gest, file = "D:/Memo/R stuff/table7gest.csv")
```

## E.2 Predict MLP

```
################
#   Library
################

library(RSNNS)
library(DMwR)

#####################
#    load data
#####################

clist <- c("EMG", "EMGIMU")
glist <- c(10,7)
cvlist <- c(1:5)
for (cc in cvlist) {
  if(cvlist[cc]==1){
    nlist <- c(10,16,17,18)
  } else if(cvlist[cc]==2){
    nlist <- c(13,14,19,22,23)
  } else if(cvlist[cc]==3){
    nlist <- c(4,6,12,20,24)
  } else if(cvlist[cc]==4){
    nlist <- c(3,5,7,25)
  } else{
    nlist <- c(2,9,11,15)
  }
  for (gg in glist)
  {
    for (ii in clist)
    {

      if (gg == 10) {
        load(sprintf("Z:/Memo/R stuff/MLP/CV_Models/Training files/CV%d%s.RData",cc,ii))
        modelName <- sprintf("modelCV%d_%s",cc,ii)
        uppLimit = 10
      } else {
        load(sprintf("Z:/Memo/R stuff/MLP/CV_Models/Training files/CV%d%s_%dgest.RData",
                     cc,ii,gg))
        modelName <- sprintf("modelCV%d_%s_%dGest",cc,ii,gg)
        uppLimit = 7
      }

      for (jj in nlist)
      {
        data_tst <- read.csv(sprintf("Z:/Memo/R stuff/MLP/CV_Models/
                                     Data/dataS%d_%s_%d_gest.csv",jj,ii,gg))
        colMeans_tst = colMeans(data_tst)#column means
        col_stdev_tst <- apply(data_tst, 2, sd) #standard deviation

        trainNorm_tst <- scale(data_tst, center=colMeans_tst, scale=col_stdev_tst )
        trainNorm_tst[is.nan(trainNorm_tst)] = 0

        forUnscaleOutput <- scale(data_tst[, ncol(data_tst)])
        if(ii=="EMG"){
          # parse turns a string or a file into an expression,
          # and eval evaluates the expression
          predictedVal <-predict(eval(parse(text = modelName)), trainNorm_tst[,1:64],
                                 trainNorm_tst[,65])
        }else{
          predictedVal <-predict(eval(parse(text = modelName)), trainNorm_tst[,1:76],
                                 trainNorm_tst[,77])
        }


        predicted <- DMwR::unscale(predictedVal,forUnscaleOutput)
        result.nnet <- data.frame(actual = data_tst[,ncol(data_tst)], prediction = predicted)
        roundedresults.nnet<-sapply(result.nnet,round,digits=0)
        roundedresultsdf.nnet = data.frame(roundedresults.nnet)

        for (i in 1:length(roundedresultsdf.nnet$prediction))
        {
          if (roundedresultsdf.nnet$prediction[i]<1)
          {
            roundedresultsdf.nnet$prediction[i] = 1
          }
          else if(roundedresultsdf.nnet$prediction[i]>uppLimit){
            roundedresultsdf.nnet$prediction[i] = uppLimit
          }
        }

        attach(roundedresultsdf.nnet)
        tableData <- table(actual, prediction)
        print(tableData)
        tt <- data.frame(data_tst[,ncol(data_tst)],roundedresultsdf.nnet$prediction)
        write.csv(tableData, file =  sprintf("Z:/Memo/R stuff/MLP/CV_Models/Predictions/
                                             S%d_%s_%d_gest_CM.csv",jj,ii,gg))

        write.csv(tt, file =  sprintf("Z:/Memo/R stuff/MLP/CV_Models/Predictions/
                                      S%d_%s_%d_gest_predictions.csv",jj,ii,gg))
      }
    }
  }
}
```

## VITA

| | |
|---|---|
| **Name:** | José Guillermo Collí Alfaro |
| **Post-secondary Education and Degrees:** | Universidad Modelo<br>Mérida, Yucatán, México<br>2009–2013 B.E.Sc.,<br>Biomedical Engineering |
| **Honours and Awards:** | CONACYT Scholarship-Master's Program |
| **Related Work Experience:** | Teaching Assistant<br>*MSE 3302 – Sensors and Actuators*<br>*ECE 9053 – Robotic Manipulators*<br>The University of Western Ontario<br>2018–2019 |
| | Research Assistant<br>The University of Western Ontario<br>2017–2019 |
| | Summer Research Assistant<br>Texas A&M University<br>2016 |
| **Publications** | J. G. Colli-Alfaro, A. Ibrahim, and A. L. Trejos, "Design of user-independent hand gesture recognition using multilayer perceptron networks and sensor fusion techniques," in *IEEE 16th International Conference on Rehabilitation Robotics*, (Toronto, Ontario), pp. 1103–1108, June 24-28, 2019. |