Alma Mater Studiorum · University of Bologna

School of Engineering
Department of Computer Science and Engineering - DISI
Master Degree in Artificial Intelligence

Motor Unit Spike Trains Recontruction from sEMG Signals for Gesture Classification on a Low-Power Processing Platform

Supervisor:
Prof. Francesco Conti

Submitted by:
Mattia Orlandi

Co-supervisors:
Prof. Simone Benatti
Dr. Elisa Donati
Dr. Davide Schiavone
Marcello Zanghieri

Academic Year 2021/2022

Abstract

Hand gesture recognition based on surface electromyography (sEMG) signals is a promising approach for the development of intuitive human-machine interfaces (HMIs) in domains such as robotics and prosthetics. The sEMG signal arises from the muscles' electrical activity, and can thus be used to recognize hand gestures. The decoding from sEMG signals to actual control signals is non-trivial; typically, control systems map sEMG patterns into a set of gestures using machine learning, failing to incorporate any physiological insight.

This master thesis aims at developing a bio-inspired hand gesture recognition system based on neuromuscular spike extraction rather than on simple pattern recognition. The system relies on a decomposition algorithm based on independent component analysis (ICA) that decomposes the sEMG signal into its constituent motor unit spike trains, which are then forwarded to a machine learning classifier. Since ICA does not guarantee a consistent motor unit ordering across different sessions, 3 approaches are proposed: 2 ordering criteria based on firing rate and negative entropy, and a re-calibration approach that allows the decomposition model to retain information about previous sessions. Using a multilayer perceptron (MLP), the latter approach results in an accuracy up to 99.4% in a 1-subject, 1-degree of freedom scenario.

Afterwards, the decomposition and classification pipeline for inference is parallelized and profiled on the PULP platform, achieving a latency < 50 ms and an energy consumption < 1 mJ. Both the classification models tested (a support vector machine and a lightweight MLP) yielded an accuracy $> 92\%$ in a 1-subject, 5-classes (4 gestures and rest) scenario.

These results prove that the proposed system is suitable for real-time execution on embedded platforms and also capable of matching the accuracy of state-of-the-art approaches, while also giving some physiological insight on the neuromuscular spikes underlying the sEMG.

Sommario

Il riconoscimento di movimenti della mano basato su segnali elettromiografici di superficie (sEMG) è un approccio promettente per lo sviluppo di interfacce uomo-macchina (HMI) intuitive in settori quali la robotica e la prostetica. Il segnale sEMG deriva dall'attività elettrica dei muscoli e può quindi essere utilizzato per riconoscere i gesti della mano. La decodifica dei segnali sEMG in segnali di controllo effettivi non è banale; in genere, i sistemi di controllo mappano i pattern sEMG in una serie di gesti utilizzando l'apprendimento automatico (*machine learning*), senza assimilare alcuna comprensione fisiologica.

Questa tesi di laurea magistrale mira a sviluppare un sistema di riconoscimento dei gesti della mano bioispirato, basato sull'estrazione di impulsi neuromuscolari piuttosto che sul semplice riconoscimento di pattern. Il sistema sfrutta un algoritmo di decomposizione basato sull'analisi delle componenti indipendenti (ICA), il quale scompone il segnale sEMG nelle sequenze di impulsi delle unità motorie che lo compongono, che vengono poi inoltrati a un classificatore di apprendimento automatico. Poiché l'ICA non garantisce un ordinamento coerente delle unità motorie tra le diverse sessioni, vengono proposti tre approcci: due criteri di ordinamento basati sulla frequenza degli impulsi e sulla neghentropia (*negative entropy*) e un approccio di ricalibrazione che consente al modello di decomposizione di conservare le informazioni sulle sessioni precedenti. Utilizzando un percettrone multistrato (MLP, dall'inglese *multilayer perceptron*), quest'ultimo approccio ha permesso di ottenere un'accuratezza fino al 99,4% in uno scenario con 1 soggetto e 1 grado di libertà.

Successivamente, la pipeline di decomposizione e classificazione per l'inferenza è stata parallelizzata e profilata sulla piattaforma PULP, ottenendo una latenza < 50 ms e un consumo energetico < 1 mJ. Entrambi i modelli di classificazione testati (una macchina a vettori di supporto—SVM, dall'inglese *support vector machine*—e una variante "leggera" del MLP) hanno prodotto un'accuratezza $> 92\%$ in uno scenario con 1 soggetto e 5 classi (4 gesti e la classe "riposo").

Questi risultati dimostrano che il sistema proposto è adatto all'esecuzione in tempo reale su piattaforme embedded ed è anche in grado di eguagliare l'accuratezza degli approcci comunemente usati nella letteratura scientifica, fornendo al contempo una visione fisiologica degli impulsi neuromuscolari alla base della sEMG.

Contents

Contents iii

List of Figures v

List of Tables vii

1 Introduction 1
- 1.1 Pattern recognition . . . 1
- 1.2 Purpose of this master thesis . . . 2
- 1.3 Thesis structure . . . 2

2 Electromyography and sEMG-based Gesture Recognition 5
- 2.1 Neuromuscular system overview . . . 5
- 2.2 Electromyography . . . 6
  - 2.2.1 Data model . . . 7
- 2.3 sEMG-based gesture recognition . . . 8
  - 2.3.1 ML-based methods . . . 8
  - 2.3.2 DL-based methods . . . 8
  - 2.3.3 sEMG signal variability . . . 9

3 Independent Component Analysis 11
- 3.1 Blind source separation . . . 11
- 3.2 Independent component analysis . . . 11
  - 3.2.1 ICA model . . . 12
  - 3.2.2 Ambiguities of ICA . . . 12
  - 3.2.3 Non-gaussianity . . . 13
  - 3.2.4 Pre-processing for ICA . . . 14
- 3.3 FastICA . . . 14

4 Materials and Methods 17
- 4.1 Extension of sEMG signals . . . 17
- 4.2 ICA-based algorithm . . . 18
  - 4.2.1 Pre-processing . . . 18
  - 4.2.2 Source extraction . . . 19
  - 4.2.3 Post-processing . . . 21
  - 4.2.4 Re-calibration . . . 21
- 4.3 Classification models . . . 23
  - 4.3.1 Linear SVM . . . 23
  - 4.3.2 MLP . . . 23

4.3.3 Lightweight MLP 24

5 Implementation **25**
5.1 Offline scenario 25
5.2 Online scenario 25

6 Experimental Results **27**
6.1 Task n. 1: decomposition validation 27
6.2 Task n. 2: variability analysis 28
6.2.1 Dataset preparation 29
6.2.2 Intra-session and inter-session classification 30
6.3 Task n. 3: online decomposition in real-time 31
6.3.1 Dataset acquisition 32
6.3.2 Dataset preparation 33
6.3.3 Pipeline profiling 34

7 Conclusions and future work **37**

Acknowledgements **39**

Bibliography **41**

List of Figures

1.1 Standard pipeline for pattern recognition 1
1.2 Proposed pipeline 2

2.1 Outline of the neuromuscular system 5
2.2 Recruitment of motor units 6

3.1 Example of the cocktail party problem 12
3.2 Example of leptokurtic and platykurtic probability distributions . . . 13
3.3 Example of BSS using FastICA 15

4.1 Overview of the ICA-based algorithm 18
4.2 Example of convolutive sphering 19
4.3 Diagram of the source extraction step. 19
4.4 Classification pipeline 23
4.5 Overview of lightweight MLP. 24

6.1 Number of extracted MUs w.r.t. the number of target MUs 29
6.2 Example of extracted MUAPTs 30
6.3 Inter-session test accuracy 31
6.4 Comparison between MUs ordering strategies 31
6.5 Overview of the acquisition system 32
6.6 Speedup curve of the parallel implementation. 34

List of Tables

6.1 Decomposition of synthetic sEMG signals 28
6.2 Profiling of the decomposition and classification pipeline 34
6.3 Classification accuracy of SVM and lightweight MLP 34

1 | Introduction

Hand gestures are one of the most natural ways for humans to express intuitive intention, manipulate objects and interact with the surrounding environment; consequently, hand gesture recognition is a very active research topic for developing human-machine interfaces (HMIs) in domains such as robotics, prosthetics and augmented reality [1, 2, 3].

A very promising approach for hand gesture recognition is electromyography (EMG) signal processing [3, 4]: EMG is a technique for measuring the bio-potential generated by the ionic flow threough the membrane of muscular fibers during contraction. It can be acquired either invasively, via needle electrodes placed inside the muscle of interest, or non-invasively via surface electrodes placed on the skin above the muscle of interest. The latter procedure is referred to as surface electromyography (sEMG) [5, 6]; granting a fully non-invasive setup, sEMG signals are particularly suited for wearable myoelectric control systems.

An open challenge in sEMG-based hand gesture recognition HMIs is robustness and reliability: in fact, sEMG signals suffer from variability due to individual anatomical differences, muscle fatigue or sensor repositioning, making sEMG-based HMIs difficult to be adopted in many real-world scenarios [3, 4].

1.1 Pattern recognition

At the current state-of-the-art (SoA), sEMG-based hand gesture classification systems rely on pattern recognition: patterns in the sEMG signal are mapped to a given set of gestures. The general structure of pattern recognition approaches is illustrated in Fig. 1.1.

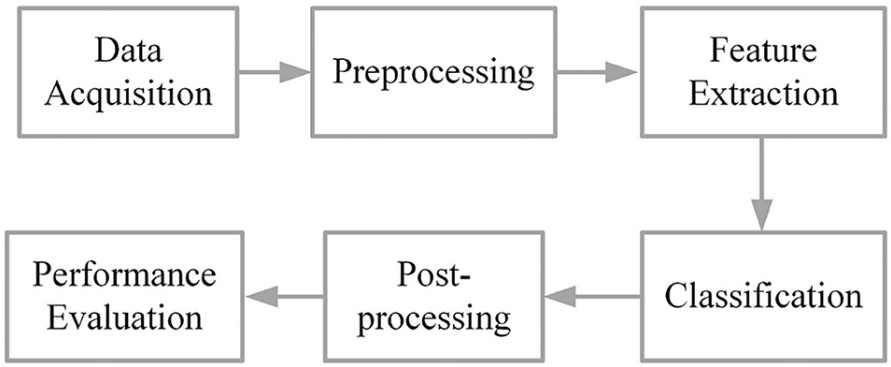

Figure 1.1: The six stages of the standard pipeline for pattern recognition. Image adapted from [4].

One of the most important stages of pattern recognition models is feature extraction: conventional machine learning (ML) models cannot learn complex features from raw sEMG data, thus they rely on hand-crafted features which however require domain-specific knowledge [4, 7]. Conversely, deep learning (DL) models can process raw sEMG data, as they have enough capacity to allow for data-driven feature learning [4, 5].

1.2 Purpose of this master thesis

Pattern recognition approaches, especially those based on DL models, act as black boxes, as they do not yield any insight on the physiology underlying the EMG. The purpose of this master thesis is two-fold: the primary goal is to develop a bio-inspired hand gesture recognition system based on neuromuscular spikes extraction, rather than on simple pattern recognition; the secondary goal is to optimize such system for real-time execution on a low-power multi-core platform.

Concerning the primary goal, the proposed system consists in a two-stage pipeline: first, the raw sEMG signal is processed by the decomposition stage, which relies on independent component analysis (ICA) for extracting the neuromuscular spike trains underlying the sEMG; then, the estimated spike trains are processed by a conventional ML classifier (see Fig. 1.2). Since ICA does not guarantee a consistent ordering of spike trains across different recording sessions, three possible approaches are tested in the inter-session scenario: two ordering criteria based on firing rate and negative entropy (a statistical property constituting the objective function of the ICA-based algorithm) and a re-calibration approach that allows the decomposition model to retain information about previous sessions.

As to the secondary goal, the decomposition and classification pipeline for inference is parallelized and profiled on the parallel ultra low power (PULP) platform, an architecture organized in a cluster of RISC-V cores targeting high energy-efficiency [8].

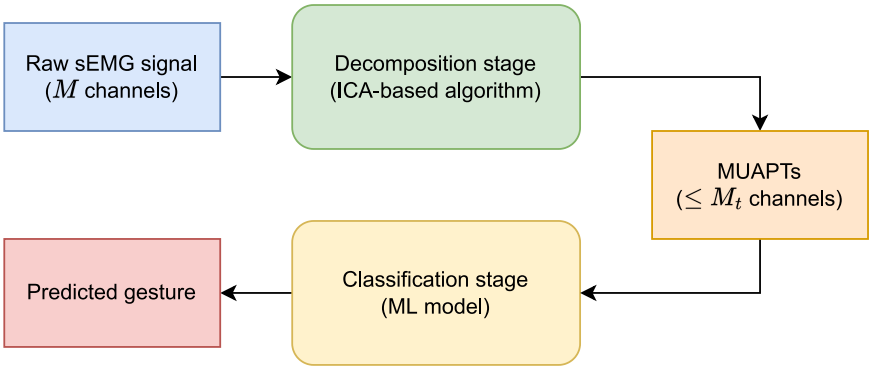

Figure 1.2: Overview of the proposed pipeline: the raw sEMG signal is first decomposed by an ICA-based algorithm into its constituent spike trains, which are then classified by a ML model that outputs the predicted gesture label.

1.3 Thesis structure

This master thesis is structured as follows:

- Chapter 2 provides an overview of the neuromuscular system, introduces the basic concepts of sEMG, and summarizes the SoA approaches based on pattern recognition, either using ML or DL models.
- Chapter 3 introduces the field of blind source separation (BSS) and describes in particular ICA, together with the FastICA algorithm.
- Chapter 4 illustrates the materials and methods of this work, namely the decomposition algorithm based on convolutive ICA, the re-calibration procedure to address the ordering of the extracted spike trains across different recording sessions, and the ML models used for classification.
- Chapter 5 provides the implementation details of the proposed system.
- Chapter 6 presents the results obtained in the tasks considered.
- Chapter 7 exposes the conclusions and outlines the future work.

2 | Electromyography and sEMG-based Gesture Recognition

In this chapter I briefly describe the physiology underlying the neuromuscular system, and how the bio-electrical activity of muscles can be measured via EMG, and particularly sEMG. Afterwards, I show how the relationship between the EMG signal and the original bio-signal can be modelled mathematically. Lastly, I provide an overview of the SoA techniques based on both ML and DL for sEMG-based gesture recognition.

2.1 Neuromuscular system overview

The central nervous system controls muscles' contraction via sequences of electric impulses generated by alpha-motoneurons (α-MNs), neurons whose somas are located in the spinal cord and whose terminal axons innervate a group of muscle fibers. A motor unit (MU)—the basic functional unit of a muscle—consists of an α-MN and its innervated muscular fibers [6] (see Fig. 2.1 - A).

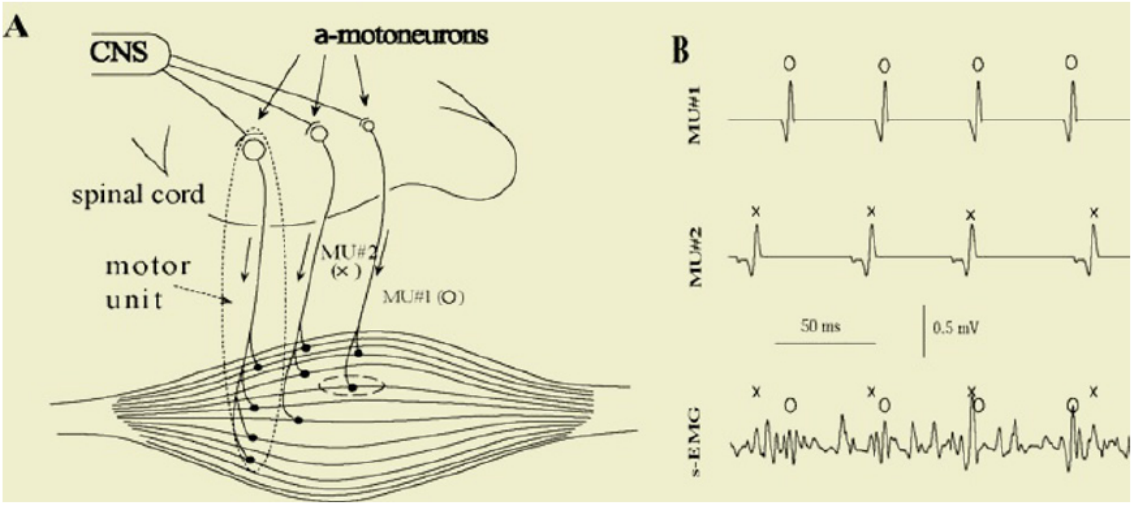

Figure 2.1: (**A**) Outline of the neuromuscular system. (**B**) Example of signal obtained from a surface electrode showing the superimposed activity of several MUs. Image adapted from [6].

When activated, MUs generate a motor unit action potential (MUAP), an electric impulse causing the contraction of the corresponding group of muscle fibers. MUAPs can be generated in sequence by repeated activations, giving rise to motor unit action potential trains (MUAPTs): in particular, as the contraction level increases,

additional MUs are recruited, and the firing rates of earlier recruited MUs increase [9] (see Fig. 2.2).

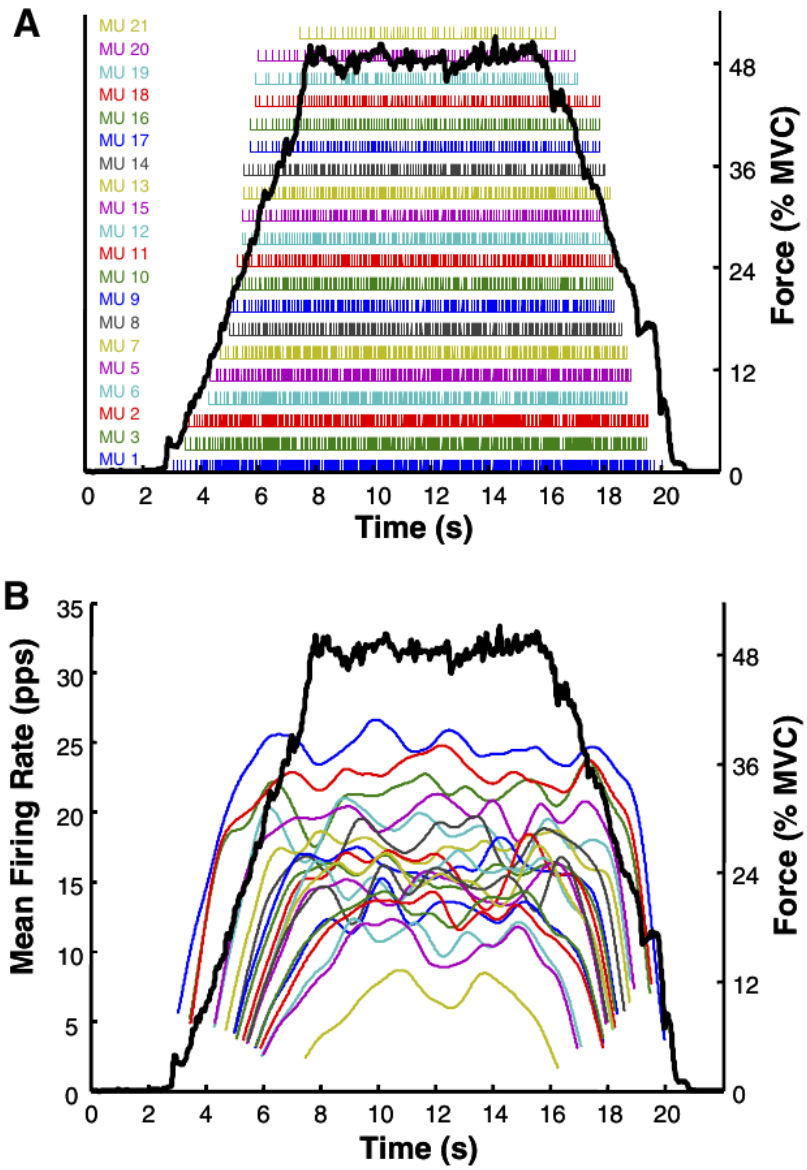

Figure 2.2: (**A**) Example of the MUAPTs generated by 21 MUs (sorted following the recruitment order) during the contraction of the first dorsal interosseous muscle; each bar represents the firing time of a MUAP, whereas the dark solid line represents the force output expressed in percentage of the maximum voluntary contraction (MVC). (**B**) Averaged time-varying firing rates for each MU derived from the timing data above; there is a hierarchical relationship between MUs, with earlier recruited MUs having greater firing rates. Image adapted from [9].

2.2 Electromyography

Electromyography (EMG) is a technique for evaluating and recording the bio-electrical activity produced by muscles during contractions. There are two kinds of EMG: intramuscular electromyography (iEMG), which relies on needle electrodes penetrating the skin to reach the muscle of interest, and surface electromyography (sEMG),

which relies on electrodes placed on the skin above the muscle of interest. Being painless and non-invasive [6, 10], the latter is preferred for developing HMIs. In particular, high-density surface electromyography (HD-sEMG) signals, obtained using grids of closely-spaced electrodes, encompass also spatial information, providing two-dimensional activation maps that describe the intensity and distribution activity of muscles [11].

2.2.1 Data model

Due to the muscle fibers being anisotropic [6], multi-channel EMG signals (either invasively or non-invasively recorded) expose a spatio-temporal superposition of many MUAPTs (see Fig. 2.1 - B), and can thus be modelled as a convolutive mixture of a series of delta functions representing the discharge timings of the MUs [10, 12]:

$$\begin{aligned} x_i(t) &= \sum_{j=1}^{N} \sum_{\tau=0}^{L-1} h_{ij}(\tau) s_j(t-\tau) + \omega_i(t) \\ i &= 1, \ldots, M, \quad t = 0, \ldots, D_{\text{rec}} \end{aligned} \tag{2.1}$$

where

- $x_i(t)$ is the i-th EMG channel;
- $h_{ij}(\tau)$ is the action potential of the j-th MU as recorded by the i-th sensor;
- $s_j(t) = \sum_k \delta\left(t - T_j^{(\text{fire})}(k)\right)$ is the MUAPT generated by the j-th MU, modelled as a sum of delta functions where $T_j^{(\text{fire})}(k)$ is the firing time of the k-th MUAP;
- $\omega_i(t)$ is the additive noise at the i-th channel;
- L is the duration of the MUAPs;
- N is the number of recruited MUs;
- M is the number of EMG channels;
- D_{rec} is the duration of the signal (measured in discrete time samples).

In matrix form, Eq. 2.1 becomes:

$$\mathbf{x}(t) = \sum_{\tau=0}^{L-1} \mathbf{H}(\tau)\mathbf{s}(t-\tau) + \boldsymbol{\omega}(t) \tag{2.2}$$

where $\mathbf{x}(t) = [x_1(t), \ldots, x_M(t)]^\intercal$ and $\mathbf{s}(t) = [s_1(t), \ldots, s_N(t)]^\intercal$ are the vectors comprising, respectively, the t-th sample of the M EMG signals and the t-th sample of the N source MUAPTs. For each τ ranging from 0 to $L-1$, the matrix of MUAPs $\mathbf{H}(\tau)$ (with shape $M \times N$) is assumed to be constant for the duration D_{rec} of the EMG signal [10].

2.3 sEMG-based gesture recognition

Thanks to its non-invasiveness and high informativeness, sEMG-based gesture recognition is a promising approach for the development of intuitive HMIs: however, the decoding from sEMG signals to actual control signals is non-trivial. At the current SoA, sEMG gesture recognition systems are mainly based on pattern recognition, achieved with two primary processing methods: machine learning (ML) and deep learning (DL) [4].

2.3.1 ML-based methods

Methods based on conventional ML are too simple to extract meaningful features from raw sEMG data; hence, they require hand-crafted feature extraction based on field-specific knowledge [4]. Typical sEMG features can be divided into three groups [4, 7]:

- Time domain: integrated EMG (IEMG), mean absolute value (MAV), variance of EMG (VAR), root mean square (RMS), waveform length (WL), zero crossings (ZC), slope sign change (SSC).
- Frequency domain: frequency median (FMD), frequency mean (FMN).
- Time-frequency domain: discrete wavelet transform (DWT).

After feature extraction, actual classification is usually performed using support vector machine (SVM), linear discriminant analysis (LDA) or random forest (RF) [7, 13, 14, 15].

For instance, Englehart and Hudgins [14] obtained an accuracy greater than 90% in a four-class problem extracting four time-domain features (ZC, WL, SSC and MAV) from a four-channel sEMG signal in combination with the LDA classifier. Atzori et al. [16] presented the Non-Invasive Advanced Prosthetics (NinaPro), a sEMG database comprising 52 hand gestures. Kuzborskij et al. [13], relying on both time- and frequency-domain features and a SVM classifier with the radial basis function kernel, obtained a 70–80% accuracy on NinaPro DB1; these results were improved by Atzori et al. [15] by considering a normalized combination of four features and using a RF classifier, which yielded an average accuracy of 75.32% and 75.27% on NinaPro DB1 and DB2, respectively.

2.3.2 DL-based methods

By contrast, DL-based methods do not need field-specific knowledge, as they allow for fully data-driven feature learning. In particular, convolutional neural networks (CNNs) can capture spatial (but also temporal) information of the signal: the two-dimensional array distribution of electrodes in HD-sEMG results in sEMG images that can be processed by CNNs [4].

Park and Lee [17] proposed a convolutional neural network (CNN)-based method to classify six hand movements from the NinaPro dataset by processing windows of sEMG signals, obtaining an accuracy up to 83%, higher than the accuracy yielded by the SVM. Geng et al. [18] used an improved CNN architecture and an eight-channel

HD-sEMG setup, obtaining an accuracy of 89.3% in the within-subject test on a single frame of sEMG image.

To capture the sequential nature of the sEMG signals, other methods rely on recurrent neural networks (RNNs) [4]: He et al. [19] proposed a model combining a long-short term memory (LSTM) network and a multilayer perceptron (MLP) and obtained an accuracy of 73.5% on NinaPro, matching the performance of Atzori et al.'s RF [15]. Zanghieri et al. [20] proposed an apporach based on temporal convolutional networks (TCNs)—a CNN-based architecture designed to process time series—yielding an inter-session accuracy of 93.7% on a 20-session dataset.

2.3.3 sEMG signal variability

One of the main challenges of sEMG-based gesture recognition is *inter-subject*, *inter-session* and *inter-posture* [21] signal variability: the statistical characteristics of sEMG signals vary with time and are affected by individual anatomical differences, sensor repositioning, skin conditions and muscle fatigue. Such variability makes it difficult to develop a robust recognition approach that can be reliably used in a real-life scenario [4, 17, 20].

3 | Independent Component Analysis

In this chapter, the concept of BSS in relation to the cocktail party problem is introduced; then, the basic concepts of ICA are defined. Lastly, the FastICA algorithm is presented.

3.1 Blind source separation

Blind source separation (BSS) is the field addressing the separation of mixed sources [22]. To better understand this concept, let us consider a situation where there are a number of signals emitted by some physical source, and that there are several receivers in different positions.

For example, N people talking simultaneously in the same room generate the speech signals (the *sources*) $s_1(t), \ldots, s_N(t)$, and M microphones produce the recored signals (the *observations*) $x_1(t), \ldots, x_M(t)$, which are a mixture of the sources with weights depending on the relative distance between people and microphones [23]. This scenario can be expressed as a linear system:

$$x_i(t) = \sum_{j=1}^{N} a_{ij} s_j(t), \quad i = 1, \ldots, M \tag{3.1}$$

or, in matrix form, as:

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t) \tag{3.2}$$

Both the mixing coefficients a_{ij} and the source signals $s_j(t)$ are unknown: the problem of recovering the sources $s_j(t)$ from the observations $x_i(t)$ alone is referred to as the *cocktail party problem* [23] (see Fig. 3.1).

Such problem can be addressed using BSS techniques, and particularly independent component analysis (ICA): this technique will be described in detail in the next section.

3.2 Independent component analysis

Independent component analysis (ICA) is a BSS technique for extracting underlying components from multivariate statistical data; in particular, ICA looks for components that are *statistically independent* and *non-gaussian* [23].

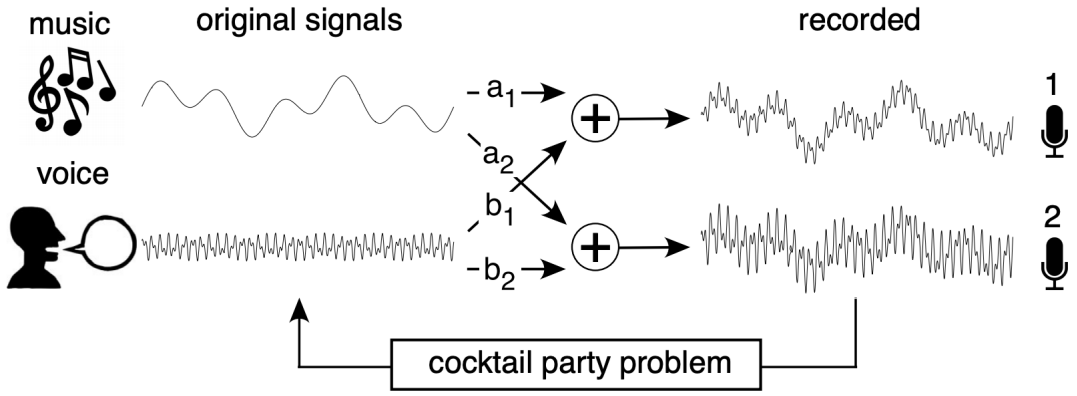

Figure 3.1: Example of the cocktail party problem: two sounds $s_1(t), s_2(t)$ are generated by music and a voice, respectively, and are recorded simultaneously by two microphones; the resulting recordings are a linear combination of the source signals, where the coefficients a_1, b_1 and a_2, b_2 reflect the proximity of each speaker to the respective microphones. Image adapted from [22].

3.2.1 ICA model

The simplest ICA model—*instantaneous* and *noiseless*—is defined as:

$$x_i = \sum_{j=1}^{N} a_{ij} s_j, \quad i = 1, \dots, M \tag{3.3}$$

or, in matrix form, as:

$$\mathbf{x} = \mathbf{A}\mathbf{s} \tag{3.4}$$

Compared to the formulation in Eq. 3.1 and 3.2, the time index t has been dropped: the reason is that the ICA model assumes that each observation x_i as well as each source s_j is a random variable; hence, the value $x_i(t)$ recorded at time t is considered as a sample of the random variable x_i [23].

For simplicity, let us assume that (i) both sources and observations have zero mean, and (ii) $N = M$, i.e., the mixing matrix $\mathbf{A}$ is square (however, this requirement can be relaxed to $N \leq M$, namely there must be more observations than sources): under this assumption, the goal of ICA is to estimate the "unmixing" matrix (i.e., the inverse of the mixing matrix $\mathbf{A}$) $\mathbf{W} = \mathbf{A}^{-1}$, such that the sources—referred to also as independent components (ICs)—can be computed as [23]:

$$\mathbf{s} = \mathbf{W}\mathbf{x} \tag{3.5}$$

3.2.2 Ambiguities of ICA

The ICA model in Eq. 3.4 presents some ambiguities [23]:

- The variances of the ICs cannot be determined, meaning that the ICs can be recovered up to a scalar multiplier; assuming that ICs have unit variance $\mathbb{E}[s_i^2] = 1$ (assuming they also have zero mean), they can be recovered up to the multiplicative sign.
- The order of the ICs cannot be determined, as the order of the terms in the sum in Eq. 3.3 can be freely changed.

3.2.3 Non-gaussianity

The key to estimating the ICA model is non-gaussianity: the Central Limit Theorem tells that the distribution of a sum of independent random variables tends towards a gaussian distribution, under certain conditions; in particular, the sum of two independent, non-gaussian random variables usually has a distribution that is closer to gaussian than any of the two original random variables [23]. In practice, to determine the j-th IC $s_j = \mathbf{w}_j^\intercal \mathbf{x}$, an algorithm must choose a vector $\mathbf{w}_j$ that *maximizes the non-gaussianity* of $\mathbf{w}_j^\intercal \mathbf{x}$ [23].

Kurtosis

A typical measure for non-gaussianity is *kurtosis* [23]; the kurtosis of a zero-mean random variable y is defined as:

$$\mathrm{kurt}(y) = \mathbb{E}[y^4] - 3\left(\mathbb{E}[y^2]\right)^2 \tag{3.6}$$

which, assuming also that y has unit variance, simplifies to:

$$\mathrm{kurt}(y) = \mathbb{E}[y^4] - 3 \tag{3.7}$$

Conveniently, kurtosis is zero for a gaussian random variable, and non-zero for (most) non-gaussian random variables: in particular, random variables that have negative kurtosis are called sub-gaussian (or *platykurtic*), while those having positive kurtosis are called super-gaussian (or *leptokurtic*). Sub-gaussian random variables have typically a "flat" probability density function (PDF), which is rather constant near zero; conversely, super-gaussian random variables have typically a "spiky" PDF with heavy tails, being relatively large at zero and for large values, while being small for intermediate values [23] (see Fig. 3.2).

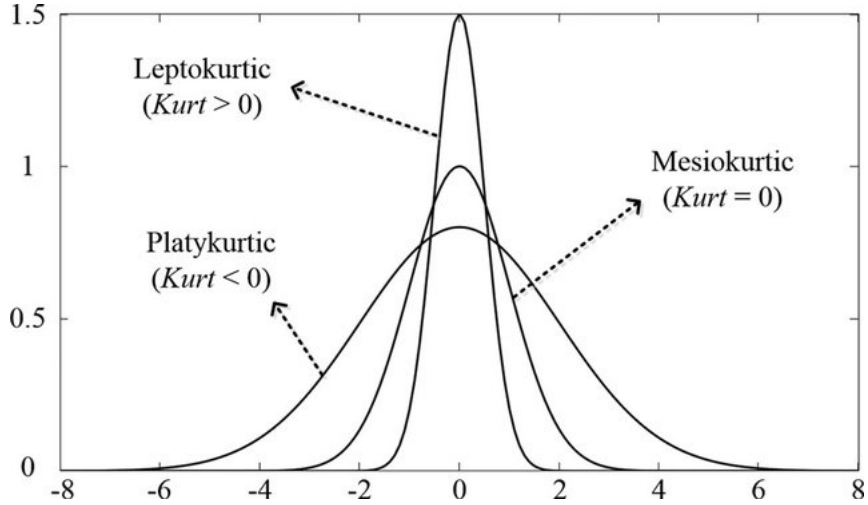

Figure 3.2: Example of leptokurtic and platykurtic probability distributions. Image adapted from [24].

Non-gaussianity could thus be measured by the absolute value of kurtosis; however, kurtosis is not used in practice, since it can be very sensitive to outliers [23].

Negative entropy

A better measure for non-gaussianity is given by *negative entropy* (or *neg-entropy*). Given a random variable y with PDF $f(y)$, its differential entropy is defined as:

$$H(y) = -\int f(y) \log f(y) dy \tag{3.8}$$

It can be shown that *a gaussian variable has the largest entropy among all random variables of equal variance*: therefore, entropy can be used as a measure of non-gaussianity [23].

To obtain a measure of non-gaussianity that is zero for a gaussian variable and always non-negative, one can employ neg-entropy, which is defined as:

$$J(y) = H(y_{\text{gauss}}) - H(y) \tag{3.9}$$

where y_{gauss} is a gaussian variable with the same covariance matrix as y.

Computing neg-entropy as per Eq. 3.9 would require a possibly non-parametric estimation of the PDF of ICs; therefore, approximations based on the maximum-entropy principle are computed in practice [23]:

$$J(y) \propto (\mathbb{E}[G(y)] - \mathbb{E}[G(\nu)])^2 \tag{3.10}$$

where ν is a normal standardized random variable, and G is some non-quadratic function referred to as *contrast function*. Some popular choices for the contrast function are the cubic function $G(y) = y^3/3$, the exponential function $G(y) = \exp(-y^2/2)$ and the log hyperbolic cosine $G(y) = \log(\cosh(y))$.

3.2.4 Pre-processing for ICA

Every ICA-based algorithm share common pre-processing steps: the most basic and necessary pre-processing is to center $\mathbf{x}$, namely to subtract its mean vector $\boldsymbol{\mu} = \mathbb{E}[\mathbf{x}]$ so as to make $\mathbf{x}$ a zero-mean variable. This step simplifies ICA-based algorithms [23].

Another useful pre-processing step is whitening: before the application of ICA-based algorithms and after centering, observations $\mathbf{x}$ are linearly transformed into a vector $\hat{\mathbf{x}}$ which is "white", namely its covariance matrix equals the identity matrix [23]. The whitened observations can be computed as:

$$\hat{\mathbf{x}} = \mathbf{E}\mathbf{D}^{-1/2}\mathbf{E}^\intercal\mathbf{x} \tag{3.11}$$

where $\mathbf{E}\mathbf{D}\mathbf{E}^\intercal$ is the eigenvalue decomposition of the covariance matrix $\mathbf{K}_{\mathbf{xx}} = \mathbb{E}[\mathbf{x}\mathbf{x}^\intercal]$.

The utility of whitening is that the new mixing matrix $\hat{\mathbf{A}} = \mathbf{E}\mathbf{D}^{-1/2}\mathbf{E}^\intercal\mathbf{A}$ is orthogonal: therefore, instead of computing the N^2 parameters of the original matrix $\mathbf{A}$, ICA-based algorithms only need to estimate the $N(N-1)/2$ parameters of the orthogonal matrix $\hat{\mathbf{A}}$ [23].

3.3 FastICA

FastICA is a very efficient algorithm to perform ICA proposed by Hyvärinen and Oja [25]. The FastICA learning rule finds for the j-th IC a direction—i.e., a unit vector $\mathbf{w}_j$—such that the projection $\mathbf{w}_j^\intercal\hat{\mathbf{x}}$ maximizes non-gaussianity, measured by the approximation of neg-entropy $J(\mathbf{w}_j^\intercal\hat{\mathbf{x}})$ in Eq. 3.10 [25]. FastICA for the j-th IC is based on the following fixed-point iteration scheme [25]:

1. Choose initial random vector $\mathbf{w}_j$.
2. Let $\mathbf{w}^+ = \mathbb{E}[\hat{\mathbf{x}}G'(\mathbf{w}_j^\intercal \hat{\mathbf{x}})] - \mathbb{E}[G''(\mathbf{w}_j^\intercal \hat{\mathbf{x}})]\mathbf{w}_j$.
3. Let $\mathbf{w}_j = \mathbf{w}_j^+ / \|\mathbf{w}_j^+\|$.
4. If not converged, repeat from step 2.

The algorithm converges when $\mathbf{w}_j$ and $\mathbf{w}_j^+$ point to the same direction, namely their dot-product is almost equal to 1 [25].

To extract N ICs, FastICA must be run with N weight vectors $\mathbf{w}_1, \ldots, \mathbf{w}_N$. To prevent the algorithm from converging multiple times to the same IC, *decorrelation* is needed after every iteration. This can be achieved via a simple deflation scheme based on Gram-Schmidt orthogonalization; assuming that FastICA extracted P ICs (i.e., P weight vectors $\mathbf{w}_1, \ldots, \mathbf{w}_P$), the weight vector $\mathbf{w}_{P+1}$ is updated after each iteration step as follows:

1. Let $\mathbf{w}_{P+1} = \mathbf{w}_{P+1} - \sum_{k=1}^{P} \mathbf{w}_{P+1}^\intercal \mathbf{w}_k \mathbf{w}_k$.
2. Let $\mathbf{w}_{P+1} = \mathbf{w}_{P+1} / \|\mathbf{w}_{P+1}\|$.

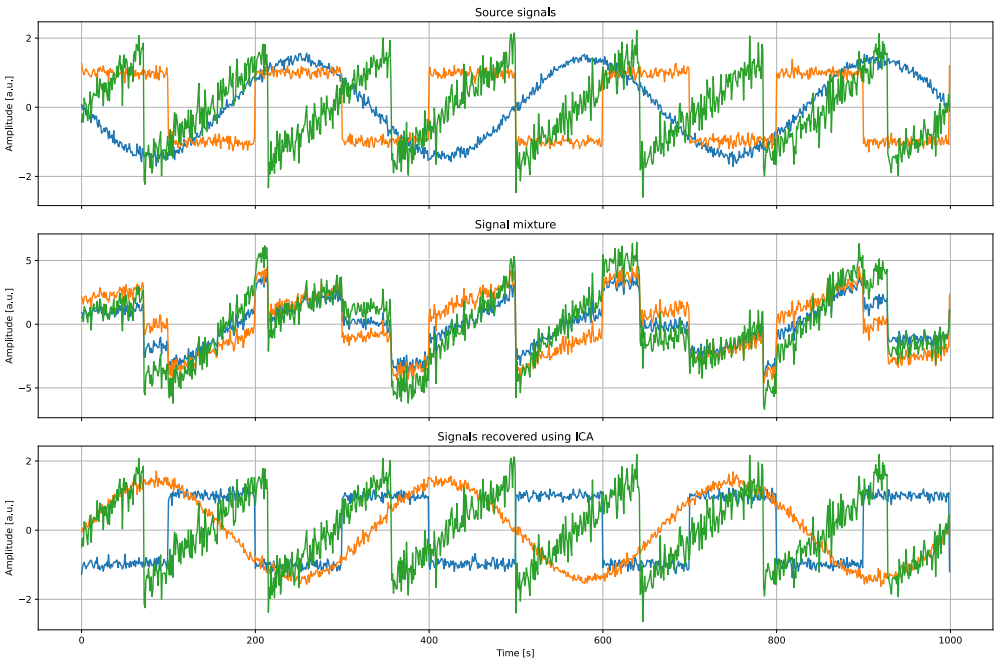

Figure 3.3: Example of the extraction of a square wave, a sine wave and a sawtooth wave from a 3-channel mixture using FastICA: as it can be seen, it manages to recover the original waveforms of the source signals even in presence of additive gaussian noise; however, it cannot recover their original signs and ordering.

4 | Materials and Methods

This chapter describes the materials and methods used in this master thesis: first, I explain how the FastICA algorithm presented in Section 3.3 can tackle the decomposition of sEMG signals, modelled by the convolutive mixture in Eq. 2.2. Afterwards, I provide an exhaustive description of the ICA-based algorithm employed in this thesis, together with a re-calibration procedure to address the problem of the order of ICs. Lastly, I provide an outline of the ML models used for actual gesture classification.

4.1 Extension of sEMG signals

As explained in Section 2.2, sEMG signals can be modelled as a mixture of source MUAPTs: in the cocktail party analogy, the MUAPTs are the speech signals whereas the sEMG channels are the signals recorded with the microphones. However, the mixture in Eq. 2.2 is convolutive rather than instantaneous: in fact, h_{ij} models the *spatial* mixing of the N sources to the M sensors, as well as the *temporal* mixing, acting as a finite impulse response (FIR) filter with length L [10, 12, 26].

The FIR-based convolutive mixture in Eq. 2.2 can equivalently be represented as a linear and instantaneous mixture of an extended vector of sources with shape $NL \times 1$, which includes the original sources and their $L-1$ *delayed replicas* [10, 12]. In order to increase the *conditionality* of the mixing process (i.e., the ratio between the number of observations and the number of sources), the vector of observations is extended by adding $f_{\text{ext}} - 1$ delayed replicas, where f_{ext} is a fixed *extension factor*. Hence, the extended model is defined as:

$$\bar{\mathbf{x}}(t) = \bar{\mathbf{H}}\bar{\mathbf{s}}(t) + \bar{\boldsymbol{\omega}}(t) \tag{4.1}$$

with

$$\bar{\mathbf{x}}(t) = [\bar{x}_1(t), \dots, \bar{x}_M(t)]^\intercal,\ \bar{x}_i(t) = [x_i(t), x_i(t-1), \dots, x_i(t-f_e+1)]$$
$$\bar{\mathbf{s}}(t) = [\bar{s}_1(t), \dots, \bar{s}_N(t)]^\intercal,\ \bar{s}_j(t) = [s_j(t), s_j(t-1), \dots, s_j(t-L-f_e+2)]$$
$$\bar{\mathbf{H}} = \begin{bmatrix} \bar{\mathbf{h}}_{11} & \dots & \bar{\mathbf{h}}_{1N} \\ \vdots & \ddots & \vdots \\ \bar{\mathbf{h}}_{M1} & \dots & \bar{\mathbf{h}}_{MN} \end{bmatrix},\ \bar{\mathbf{h}}_{ij} = \begin{bmatrix} h_{ij}(0) & \dots & h_{ij}(L-1) & 0 & \dots & 0 \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & h_{ij}(0) & \dots & h_{ij}(L-1) \end{bmatrix}$$

Assuming matrix $\bar{\mathbf{H}}$ is full rank, in order to address the inverse problem the system in Eq. 4.1 must be *overdetermined*, namely the number of extended observations must be higher than the number of sources multiplied by the length of the filters: $Mf_{\text{ext}} \geq N(L + f_{\text{ext}} - 1)$ [10, 12].

The procedure described so far is referred to as *extension*: it converts a convolutive mixture into an instantaneous mixture that can be processed by FastICA. However, the requirement of statistical independence between sources is no longer satisfied, since the delayed replicas are not independent by construction. Nonetheless, the contrast functions used in ICA measures the non-gaussianity of the estimated sources, and only indirectly their independence; in fact, non-gaussianity can be used as a proxy of sparsity and, contrary to independence, sparsity is preserved after the extension. Therefore, ICA tries to maximize the sparsity of the sources, and since MUAPTs are leptokurtic—and thus sparse—ICA can indeed be used to identify MUs [10, 27, 28, 29].

4.2 ICA-based algorithm

The proposed system is inspired by the work of Negro et al. [10], and it relies on a variation of FastICA (cf. Section 3.3). The aim is to recover as many MUs as possible from the extended observations $\bar{\mathbf{x}}(t)$; non-identified MUs will be accounted for by the additive noise term $\bar{\boldsymbol{\omega}}(t)$ [10, 12] (cf. Eq. 4.1).

The block diagram in Fig. 4.1 shows an overview of the decomposition algorithm: it takes as input the raw sEMG signal and it outputs the estimated discharge timings of the identified MUs.

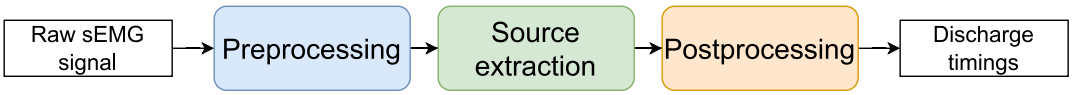

Figure 4.1: Overview of the ICA-based algorithm: the system processes raw sEMG data and outputs the estimated discharge timings of the identified MUs.

This ICA-based algorithm is designed to be *calibrated* (i.e., fit on a *calibration signal*) at the beginning of the recording session: during calibration, the decomposition model learns relevant parameters, namely the mean vector $\boldsymbol{\mu}_{\bar{\mathbf{x}}}$, the whitening matrix $\mathbf{W}$, the separation matrix $\mathbf{B}$ and the thresholds for spike/noise classification $\boldsymbol{\Gamma}$; such parameters are stored, and can be used in an online fashion to efficiently decompose raw sEMG signals during the same session [28, 27].

4.2.1 Pre-processing

The first pre-processing step is filtering: a fifth-order Butterworth band-pass filter (20 Hz–700 Hz) is applied on the sEMG signal to reduce the effect of noise, followed by several second-order infinite impulse response (IIR) notch filters to remove power line interference (30 Hz, 50 Hz, 60 Hz, 90 Hz and 150 Hz) [30]. The sEMG is then extended with the procedure described in Section 4.1.

Afterwards, the standard pre-processing steps for ICA, namely centering and whitening of the extended vector of observations $\bar{\mathbf{x}}(t)$, are applied: after subtracting its mean $\boldsymbol{\mu}_{\bar{\mathbf{x}}}$, the vector is whitened via the ZCA method, resulting in:

$$\hat{\mathbf{x}}(t) = \mathbf{W}\bar{\mathbf{x}}(t), \quad \mathbf{W} = \mathbf{U}^{-1/2}\mathbf{V}^{\intercal} \tag{4.2}$$

where $\mathbf{U}\mathbf{V}^{\intercal}$ is the singular value decomposition (SVD) of the covariance matrix $\mathbf{K}_{\bar{\mathbf{x}}\bar{\mathbf{x}}} = \mathbb{E}[\bar{\mathbf{x}}(t)\bar{\mathbf{x}}^{\intercal}(t)]$. The combination of extension and whitening is referred to as

"convolutive sphering" [10, 26]. In particular, only the eigenvalues greater than a certain threshold are considered, and such threshold is set to the average of the smallest half of the eigenvalues [10]: this regularization procedure should help reducing the effect of noise [10, 25].

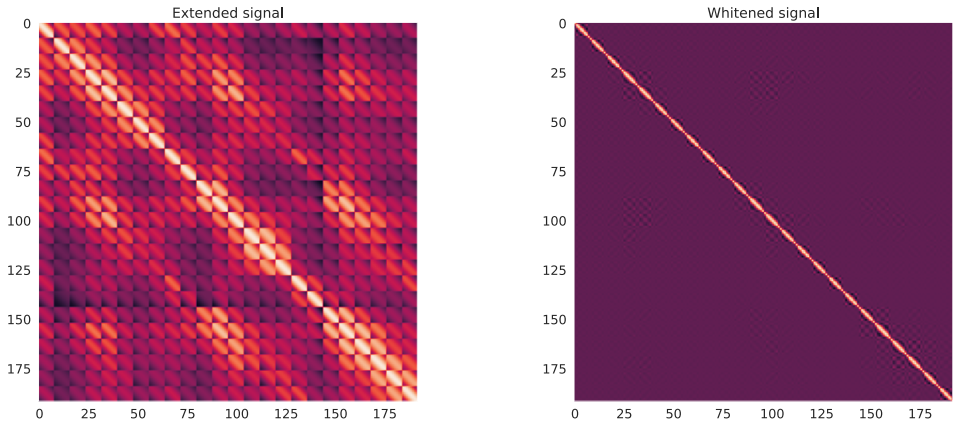

Figure 4.2: Example of convolutive sphering: on the left, the correlation matrix of a 24-channel sEMG signal, extended using $f_{\text{ext}} = 8$; on the right, the correlation matrix after the whitening process. The sEMG signal used in this example belongs to the putEMG dataset [30].

4.2.2 Source extraction

The source extraction stage is described by the block diagram in Fig. 4.3.

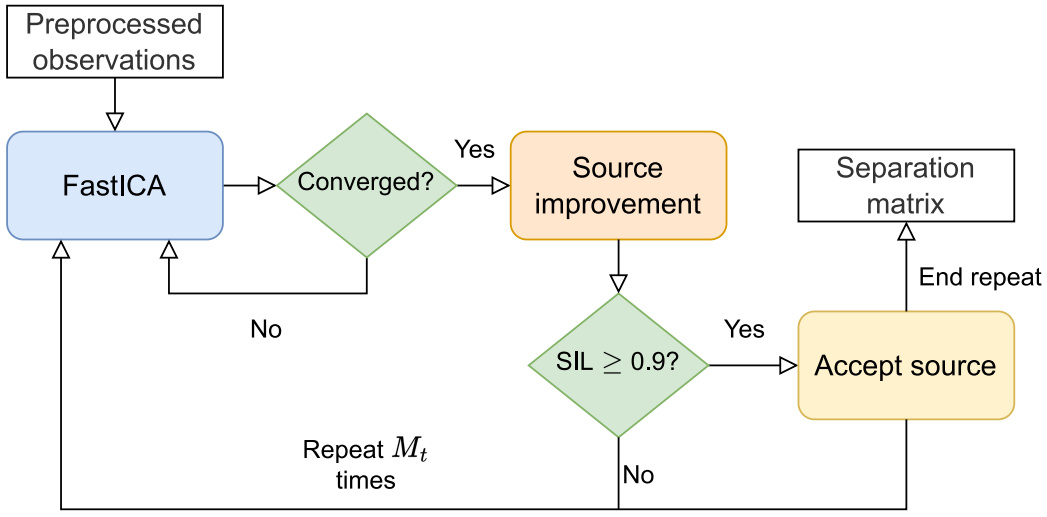

Figure 4.3: Diagram of the source extraction step.

The FastICA algorithm, described in Alg. 1, is applied to obtain an initial estimation for the MUAPT. The log hyberbolic cosine contrast function is used, since it is more robust to outliers [10]. Gram-Schmidt orthogonalization is employed to prevent the algorithm from converging to the same MUAPT multiple times [10, 25]. A fast convergence of FastICA requires a suitable initialization point of the projection vector [25], therefore, as per [10, 27], (i) the whitened observations are

squared and summed along the channels dimension, (*ii*) peak detection is performed on the resulting signal, and (*iii*) each separation vector is initialized to the value of a peak, chosen randomly among the 25% highest peaks.

Algorithm 1 FastICA iteration

Input:

$\hat{\mathbf{x}}$: pre-whitened observations with shape $Mf_e \times D$;
$\mathbf{B}$: estimated separation matrix for previous sources;
$\mathbf{w}_{\text{init}}$: initial estimate for separation vector;
`max_iter = 100`: maximum n. of iterations;
`th_conv = 1e-4`: threshold for convergence.

function FAST_ICA($\hat{\mathbf{x}}$, $\mathbf{B}$, $\mathbf{w}_{\text{init}}$, `max_iter`, `th_conv`)
  $\mathbf{w}_i \leftarrow \mathbf{w}_{\text{init}}$
  `converged` $\leftarrow \perp$
  `iter_idx` $\leftarrow 0$
  while `iter_idx` $<$ `max_iter` **do**
    $\mathbf{w}i^+ \leftarrow \mathbb{E}[\bar{\mathbf{x}} G'(\mathbf{w}i^\intercal \bar{\mathbf{x}})] - \mathbb{E}[G''(\mathbf{w}i^\intercal \bar{\mathbf{x}})]\mathbf{w}i$ ▷ compute new vector
    $\mathbf{w}i^+ \leftarrow \mathbf{w}i^+ - \mathbf{B}^\intercal \mathbf{B}\mathbf{w}i^+$ ▷ apply Gram-Schmidt orthogonalization
    $\mathbf{w}i^+ \leftarrow {\mathbf{w}i^+}/{\|\mathbf{w}i^+\|}$ ▷ normalize
    `distance` $\leftarrow 1 - |\mathbf{w}i^\intercal \mathbf{w}i^+|$
    $\mathbf{w}i \leftarrow \mathbf{w}i^+$ ▷ update vector
    if `distance` $<$ `th_conv` **then**
      `converged` $\leftarrow \top$
      `break`
    end if
    `iter_idx` $\leftarrow$ `iter_idx + 1`
  end while
  return $\mathbf{w}i$, `converged`
end function

The MUAPTs estimated by FastICA may be unreliable. Therefore, a second iterative procedure, described in Alg. 2, is employed: spike trains are estimated by performing peak detection on the squared source and by classifying the detected peaks with K-means with $K = 2$ to discriminate between actual spikes and noise; then, the estimated MUAPT is refined in order to reduce the variability of its discharge timings. This source improvement iteration is repeated until the variability reaches a minimum [10, 12, 28, 27].

A silhouette measure (SIL), defined as the normalized difference between the sum of point-to-centroid distances within cluster and between clusters—quantifying the quality of the K-means spike/noise classification—is computed on the estimated spike train, and the corresponding separation vector is kept only if SIL ≥ 0.9 [10, 28, 27, 31, 29]. The process is repeated M_t times, where M_t is a hyper-parameter indicating the number of target MUs [10, 28, 27, 31, 29] (i.e., the number of MUs the system tries to extract): eventually, the separation matrix $\mathbf{B}$ will contain at most M_t rows, namely the separation vectors for at most M_t source MUAPTs. The whole source extraction procedure is summarized in Alg. 3.

Algorithm 2 Source improvement iteration

Input:
$\hat{\mathbf{x}}$: pre-whitened observations with shape $Mf_e \times D$;
$\mathbf{B}$: separation matrix estimated for previous sources;
$\mathbf{w}i$: estimate for separation vector;
`max_iter = 100`: maximum n. of iterations;
`th_conv = 1e-4`: threshold for convergence.

function SOURCE_IMPROVEMENT($\hat{\mathbf{x}}$, $\mathbf{B}$, $\mathbf{w}i$, `max_iter`, `th_conv`)
  CoV $\leftarrow \infty$ ▷ initialize coefficient of variation (CoV)
  `iter_idx` $\leftarrow 0$
  while `iter_idx` $<$ `max_iter` **do**
    $s_i(t) \leftarrow \mathbf{w}i^\intercal \hat{\mathbf{x}}(t)$
    $[\mathrm{pk}_1, \ldots, \mathrm{pk}_U] \leftarrow$ `peak_detection`$(s_i^2(t))$ ▷ detect potential spikes
    $[\mathrm{sp}_1, \ldots, \mathrm{sp}_V] \leftarrow$ `k-means`$([\mathrm{pk}_1, \ldots, \mathrm{pk}_U], k = 2)$ ▷ classify actual spikes
    $\mathbf{w}i^+ \leftarrow \frac{1}{V} \sum_{v=1}^{V} \hat{\mathbf{x}}(\mathrm{sp}_v)$ ▷ refine vector by taking mean at spike locations
    $\mathbf{w}i^+ \leftarrow \mathbf{w}i^+ - \mathbf{B}^\intercal \mathbf{B} \mathbf{w}i^+$ ▷ apply Gram-Schmidt orthogonalization
    $\mathbf{w}i^+ \leftarrow {\mathbf{w}i^+}/{\|\mathbf{w}i^+\|}$ ▷ normalize
    isi $\leftarrow$ `diff`$([\mathrm{sp}_1, \ldots, \mathrm{sp}_V])$ ▷ compute inter-spike interval (ISI)
    $\mathrm{CoV}^+ \leftarrow {\sigma_{\mathrm{isi}}}/{\mu_{\mathrm{isi}}}$ ▷ compute new CoV
    if $|\mathrm{CoV}^+ - \mathrm{CoV}| <$ `th_conv` **then**
      $\mathbf{w}i \leftarrow \mathbf{w}i^+$ ▷ update vector
      `break`
    end if
    if $\mathrm{CoV}^+ > \mathrm{CoV}$ **then**
      `break`
    end if
    $\mathbf{w}i \leftarrow \mathbf{w}i^+$ ▷ update vector
    CoV $\leftarrow \mathrm{CoV}^+$ ▷ update CoV
    `iter_idx` $\leftarrow$ `iter_idx` $+ 1$
  end while
  return $\mathbf{w}i$
end function

4.2.3 Post-processing

Finally, a post-processing step is applied in order to remove either inactive MUs, namely MUs firing less than a given threshold (in spike/s) [27, 29, 31], and duplicates: in fact, as mentioned before, the extension step in Eq. 4.1 introduces delayed replicas of the sources, and thus the decomposition algorithm may converge to such duplicates, despite the orthogonalization step. In particular, delayed replicas are detected by checking those MUAPTs that share more than 50% firing events within a ± 1 ms tolerance window, and only the MU with the highest SIL is kept [29, 31].

4.2.4 Re-calibration

Since ICA does not guarantee a consistent ordering of the extracted sources across different sessions, the model can be re-calibrated on sEMG data from a new recording

Algorithm 3 Source extraction

Input:

$\hat{\mathbf{x}}$: pre-whitened observations with shape $Mf_e \times D$;
M_t: n. of target sources to extract;
`max_iter = 100`: maximum n. of iterations;
`th_conv = 1e-4`: threshold for convergence;
`th_sil = 0.9`: threshold for SIL.

$\mathbf{B} \leftarrow [\,]$ ▷ separation matrix initially empty
for $i \leftarrow 0, M_t$ **do**
    ▷ 1. Perform FastICA for current unit
    $\mathbf{w}i \leftarrow$ `init_w`(i) ▷ initialize separation vector
    $[\mathbf{w}i,$ `converged`$] \leftarrow$ `fast_ica`$(\hat{\mathbf{x}}, \mathbf{B}, \mathbf{w}\text{init},$ `max_iter`, `th_conv`$)$
    if ¬`converged` **then**
        `continue`
    end if
    ▷ 2. Perform source improvement
    $\mathbf{w}i \leftarrow$ `source_improvement`$(\hat{\mathbf{x}}, \mathbf{B}, \mathbf{w}i,$ `max_iter`, `th_conv`$)$
    ▷ 3. Check SIL
    SIL $\leftarrow$ `compute_sil`$(\mathbf{w}i^\intercal \hat{\mathbf{x}}(t))$
    if SIL $\geq$ `th_sil` **then**
        $\mathbf{B} \leftarrow [\mathbf{B}, \mathbf{w}i^\intercal]$ ▷ append separation vector to matrix
    end if
end for

session. First of all, the mean vector $\boldsymbol{\mu}_{\bar{\mathbf{x}}}$ and the whitening matrix $\mathbf{W}$ are updated via the following momentum-controlled rule:

$$\boldsymbol{\mu}_{\bar{\mathbf{x}}} \leftarrow (1-\beta) \cdot \boldsymbol{\mu}_{\bar{\mathbf{x}}} + \beta \cdot \boldsymbol{\mu}_{\bar{\mathbf{x}}}^{(\text{new data})}, \quad \mathbf{W} \leftarrow (1-\beta) \cdot \mathbf{W} + \beta \cdot \mathbf{W}^{(\text{new data})} \tag{4.3}$$

where β is momentum, and *new data* refers to parameters fit on the new session only. Then, assuming that in the previous session P MUs ($P < M_t$) were extracted (i.e., the separation matrix $\mathbf{B}$ has P rows), the model estimates up to $(M_t - P)$ new MUs from the current session; the corresponding separation vectors are used to update the separation matrix $\mathbf{B}$ by vertically concatenating them to $\mathbf{B}$ itself (i.e., the new rows represent the coefficients of the new extracted MUs). The orthogonalization step prevents the estimation of separation vectors already identified during the previous session (i.e., already present in $\mathbf{B}$), and for this reason the model may estimate fewer MUs than M_t. If Q MUs are extracted in total ($P \leq Q \leq M_t$), the separation matrix $\mathbf{B}$ has Q rows, and the vector of spike/noise thresholds $\boldsymbol{\Gamma}^{(\text{new data})}$ has Q entries. Finally, the first P entries of the vector of spike/noise thresholds $\boldsymbol{\Gamma}$ are updated with the same momentum-controlled rule as $\boldsymbol{\mu}_{\bar{\mathbf{x}}}$ and $\mathbf{W}$:

$$\begin{aligned} &\Gamma_i \leftarrow (1-\beta) \cdot \Gamma_i + \beta \cdot \Gamma_i^{(\text{new data})}, \quad i = 1, \ldots, P \\ &\Gamma_j \leftarrow \Gamma_j^{(\text{new data})}, \quad j = P+1, \ldots, Q \end{aligned} \tag{4.4}$$

In this master thesis, a momentum $\beta = 0.5$ was employed.

4.3 Classification models

After the calibration of the ICA-based algorithm, let us assume P MUs were identified. D-sample-long windows of sEMG signals are decomposed online, and the estimated MUAPTs are encoded as a binary matrix $\mathbf{S}$ with shape $P \times D$ where $s_{ij} = 1$ if the i-th MU fired a MUAP at the j-th time sample, or $s_{ij} = 0$ otherwise. Such binary matrix is classified by ML models to predict the actual hand gesture (see Fig. 4.4).

Three models were tested: a linear SVM and two MLPs, one standard and one lightweight. To better test the effectiveness of the decomposition stage, all the three classifiers are very simple such that complex feature learning is avoided.

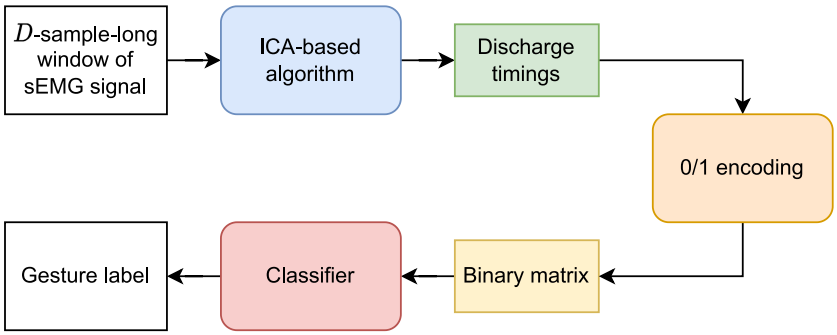

Figure 4.4: Overview of the classification pipeline: windows of a sEMG signal are decomposed into MUAPTs, which are encoded as binary matrices and fed into a classifier that predicts the gesture label.

4.3.1 Linear SVM

The linear support vector machine (SVM) [32] is a ML model that classifies data points in two classes by finding a hyperplane maximizing the distance between the nearest points of either class and the hyperplane itself. Multi-class classification is achieved by means of the "one-vs-one" approach: given C classes, $C(C-1)/2$ binary classifiers are trained on data from two classes; each binary classifier predicts one class label, and the class with most predictions is considered the predicted class.

Due to their high training complexity, it is not practicable to process high-dimensional feature vectors with SVMs: therefore, instead of processing whole (flattened) $P \cdot D$ binary matrices representing the MUAPTs, the linear SVM employed in this thesis processes a P-dimensional vector representing the number of MUAPs generated by each MU.

4.3.2 MLP

The multilayer perceptron (MLP) is a fully connected feedforward artificial neural network (ANN) consisting of at least three layers of nodes: an input layer, a hidden layer and an output layer; except for the input nodes, each node uses a non-linear activation function. MLPs are trained via a supervised learning technique called backpropagation.

The MLP employed in this thesis features a single hidden layer with 32 hidden nodes, each using the ReLU activation function, and processes whole flattened $P \cdot D$

binary matrices; for typical values of P and D, this results in a very high number of parameters.

4.3.3 Lightweight MLP

The last model tested is a variant of the MLP featuring "separable" layers to reduce the number of parameters: the whole $P \times D$ binary matrices are first processed by a *temporal aggregation* layer with N_{TA} nodes, resulting in a $P \times N_{\text{TA}}$ intermediate matrix; this matrix is then flattened and processed by a *channel aggregation* layer with N_{CA} nodes, which outputs a N_{CA}-dimensional vector. Such vector is finally processed by the classification layer. Both temporal aggregation and channel aggregation layers use the ReLU activation function. The block diagram in Fig. 4.5 provides an overview of the lightweight MLP architecture.

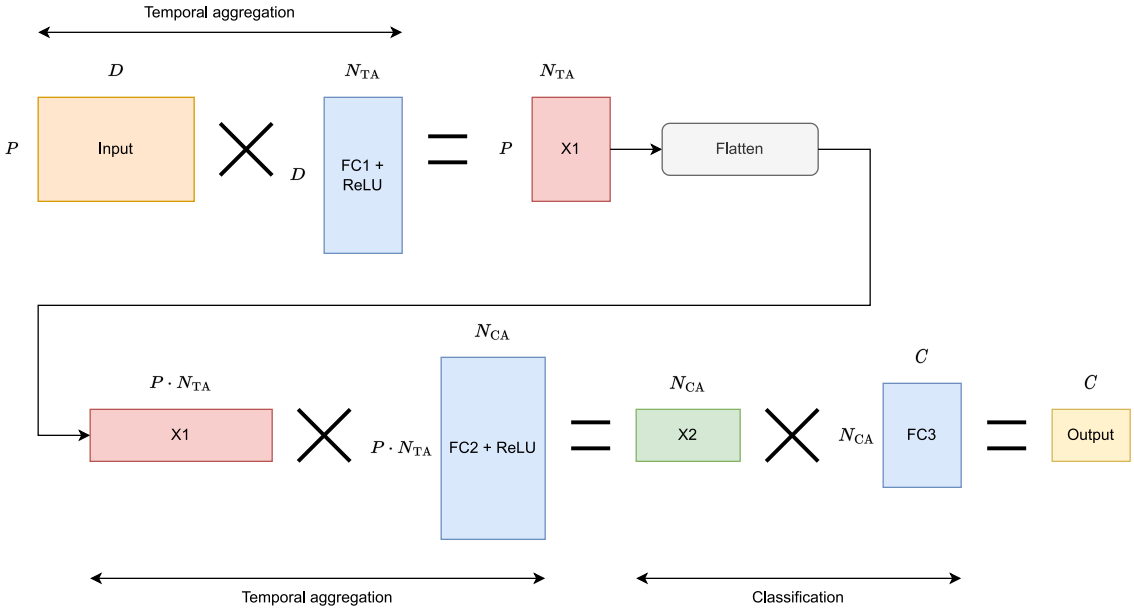

Figure 4.5: Overview of lightweight MLP.

5 | Implementation

The proposed system was tested in two scenarios: an offline scenario focused on analyzing intra-session and inter-session variability, and an online scenario focused on profiling the parallel implementation of the algorithm for real-time inference on the parallel ultra low power (PULP) platform, an architecture organized in a cluster of RISC-V cores targeting high energy-efficiency [8].

5.1 Offline scenario

The algorithm for the offline scenario was implemented using the Python language; in particular, the downstream classification stage was implemented using PyTorch [33] and scikit-learn [34] libraries. The project is organized as follows:

- all the experiments were performed in Jupyter notebooks;
- the actual logic is implemented in the `semg_bss` package:
  - the `emg_separator.py` module contains the `EMGSeparator` class, exposing the methods `calibrate`, `recalibrate` and `decompose` for initial calibration, re-calibration and decomposition using pre-computed parameters, respectively;
  - the `preprocessing.py` module contains the functions for sEMG signals pre-processing;
  - the `clf` sub-package contains the modules for the classification stage:
    - the `mlp.py` module contains the `MUAPTClassifierMLP` class, implementing the standard MLP model;
    - the `mlp_light.py` module contains the `MUAPTClassifierMLPLight` class, implementing the lightweight MLP model;
  - the `datasets` sub-package contains the modules for managing the different datasets used;
  - the `plt.py` module contains the functions for plotting and visualization.

The Python implementation is available at https://github.com/nihil21/semg-bss.

5.2 Online scenario

A parallel implementation of the decomposition and classification pipeline for the online scenario was developed using the C language; the implementation was profiled on the PULP platform [8] using the GVSoC simulator [35]:

- in the decomposition stage, each core processes a subset of the extended channels;
- in the classification stage, parallelism involves either the input features of the lightweight MLP or the binary classifiers of the SVM.

The project is organized as a Docker container [36], with the PULP-SDK pre-installed; the logic is implemented in the `semg-bss-online` directory: in particular, the decomposition stage is implemented in the `decomp` module, and the classification stage is implemented in the `clf` module. The program can be compiled and executed using `make`. The C implementation is available at https://github.com/nihil21/semg-bss-pulp.

6 | Experimental Results

The system was validated through both a synthetic sEMG dataset with known ground truth MUAPTs and real sEMG signals recorded during the execution of a set of gestures, for which a ground truth label is provided. As per [10, 28, 31, 29], I set (*i*) the maximum number of FastICA and source improvement iterations to 100, (*ii*) the threshold for convergence of FastICA and source improvement iterations to 10^{-4}, and (*iii*) the threshold for SIL to 0.9.

6.1 Task n. 1: decomposition validation

To assess the decomposition accuracy of the system, a synthetic dataset was used. The dataset was provided by Mohebian et al. [37]: it was generated using a planar volume conductor model, and consists of 15 simulated sEMG signals with 90 channels, a length of 16 s and a sampling frequency of 4096 Hz. The muscle excitation for each simulated signal was set to either 10%, 30% or 50% MVC.

As per [37], a MU was considered correctly identified when at least 30% of its firings were time-locked in a window of ±0.5 ms with a ground truth MU. For each identified MU, the accuracy of the decomposition was assessed by computing the following metrics [37]:

- rate of agreement (RoA): $\frac{\text{TP}}{\text{TP+FN+FP}}$
- precision: $\frac{\text{TP}}{\text{TP+FP}}$
- recall: $\frac{\text{TP}}{\text{TP+FN}}$

The number of target MUs M_t was set to 300, 400 and 500 for signals with MVC of 10%, 30% and 50%, respectively. As per [10], the extension factor was set to $f_{ext} = 16$.

Results are shown in Table 6.1: "#MUs GT" is the number of active ground-truth MUs, "#MUs extracted" is the actual number of MUs the system manages to extract, and "#MUs (identified)" is the number of extracted MUs correlating with a ground truth MU based on the time-locking criteria explained before. As it can be seen, our system is able to detect only a small portion of the ground-truth MUs (the ones simulated near the locations of the electrodes): this is expected, since MUs far from the electrodes contribute to the physiological noise; as a comparison, the modified gradient convolution kernel compensation (gCKC) extracted on average 16.41 ± 4.18 MUs [37]. Noteworthily, the identified MUs are very strongly correlated with the ground truth: in particular, the larger the MVC, the fewer the extracted and identified MUs are, whereas the correlation metrics remain firmly high. Differently

MVC	M_t	#MUs GT	#MUs extracted	#MUs identified	RoA	Precision	Recall
10%	300	262	90.80 ± 14.27	24.40 ± 2.87	0.94 ± 0.17	0.94 ± 0.16	0.98 ± 0.05
30%	400	388	72.40 ± 26.42	19.80 ± 2.99	0.98 ± 0.09	0.99 ± 0.08	0.99 ± 0.04
50%	500	446	51.20 ± 4.07	14.20 ± 2.86	0.97 ± 0.11	0.98 ± 0.09	0.98 ± 0.08

Table 6.1: Number of extracted and identified MUs, RoA, precision and recall, averaged over five simulated signals for each MVC value: results are reported as mean $\pm$ std.

from [37], in which the authors added gaussian noise with a given signal-to-noise ratio (SNR) to the simulated sEMG signals, these results were obtained without considering any additional noise, apart from the simulated physiological noise and the contribution of unidentified distant MUs.

6.2 Task n. 2: variability analysis

The proposed system was tested on the putEMG dataset, collected by Kaczmarek et al. [30] using a 24-electrode matrix placed on the forearm. It includes recordings sampled at 5120 Hz from 44 healthy subjects and from two sessions, separated by at least one week. Each recording contains repetitions of seven active gestures separated by an idle state. For each subject and session, the dataset provides three signals:

- `repeats_long`: seven action blocks (one per gesture), each containing eight repetitions of the same gesture;
- `repeats_short`: seven action blocks (one per gesture), each containing six repetitions of the same gesture;
- `sequential`: six action blocks, each containing all the seven gestures in sequence.

In particular, I considered a simplified scenario with only one subject (subject 03, a 37-year-old male), and two gestures (the n. 2 and 3, namely hand flexion and extension). As per [10, 28, 27, 29], all signals were pre-processed beforehand using a 20 Hz to 700 Hz band-pass filter and several notch filters at 30 Hz, 50 Hz, 60 Hz, 90 Hz and 150 Hz to attenuate power-line noise.

Three decomposition approaches were tested:

Firing Rate Ordering Two distinct decomposition models were employed, one calibrated on a signal recorded during the first session and the other on a signal recorded during the second session. In both cases the extracted MUs were ordered by their firing rate, from the highest to the lowest (the order was kept unchanged when performing inference).

Negative Entropy Ordering As before, two distinct decomposition models were employed for the two sessions, but instead of being ordered by firing rate the extracted MUs were ordered by negative entropy (or neg-entropy), from the highest to the lowest (the order was kept unchanged during inference); the rationale behind neg-entropy ordering is that FastICA uses neg-entropy as

contrast function to estimate MUAPTs, and thus the same MU in different recording sessions may have a similar neg-entropy value.

Re-calibration A single decomposition model was employed: it was calibrated on a signal from the first session, and the estimated decomposition parameters (such as the mean vector, the whitening matrix, the separation vectors and the spike thresholds) were updated by re-calibrating the model on a signal from the second session, using a momentum $\beta = 0.5$.

In all the above tests, the same calibration signal was used, namely the slice from 50 s to 150 s of the `repeats_long` signal: such slice contains the sEMG recording during the performance of both flexion and extension gestures in sequence. Multiple values for the hyper-parameter M_t were tested; moreover, I examined the effect of re-sampling the sEMG signal at a lower frequency by applying an eighth-order type I Chebyshev filter and decimating at 2560 Hz and 1280 Hz (i.e., one half and one quarter of the original sampling frequency). The extension factor was fixed to $f_{ext} = 8$ regardless of the sampling rate, since, as per [10, 28], it was observed that the decomposition performance is similar for $f_{ext} \in \{8, \dots, 31\}$). The relationship between the sampling frequency, the number of target MUs and the actual number of extracted MUs can be observed in Fig. 6.1.

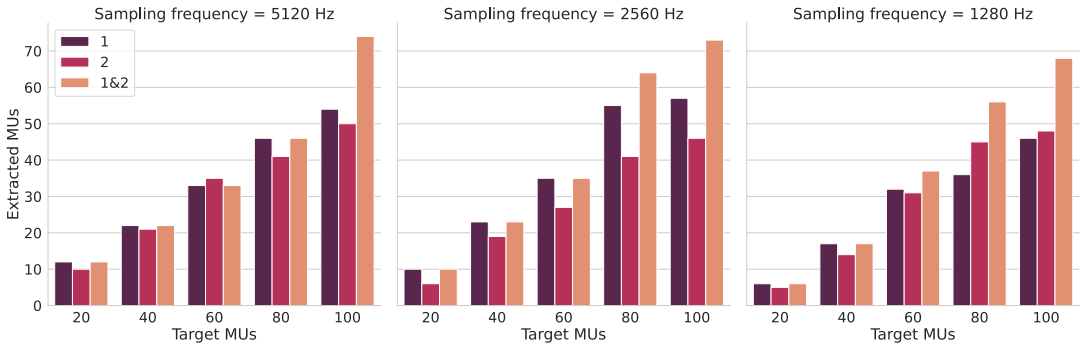

Figure 6.1: Number of extracted MUs with respect to the number of target MUs obtained by calibrating the model on the first, second and both sessions (re-calibration).

6.2.1 Dataset preparation

After calibration, the dataset for gesture classification was prepared as follows:

1. the slices corresponding to the gestures of interest from all the three sEMG signals (i.e., `repeats_long`, `repeats_short` and `sequential`) were considered;
2. since the duration of the gestures may differ, each slice was divided in several windows with a fixed length of 500 ms and with an overlap of 400 ms;
3. MUAPTs were extracted from each window using the calibrated decomposition model;
4. extracted MUAPTs were encoded as a $P \times D$ matrix of zeros and ones (the latter representing a spike), where P is the number of extracted MUs and $D = 0.5\text{s} \cdot f_s[\text{Hz}]$ is the number of samples;

5. the $P \times D$ matrix was flattened and fed into the standard MLP binary classifier described in Section 4.3.

6.2.2 Intra-session and inter-session classification

Two subtasks were considered:

- *intra-session classification* subtask, in which the MLP was trained on the flattened spikes extracted from the `repeats_long` and `repeats_short` signals recorded in the first session and was tested on the flattened spikes extracted from the `sequential` signal recorded in the first session;
- *inter-session classification* subtask, in which the MLP was trained on the flattened spikes extracted from all the signals recorded in the first session and was tested on the flattened spikes extracted from all the signals recorded in the second session.

Intra-session subtask

Regardless of the decomposition approach, the sampling frequency and the number of target MUs, the test accuracy in the intra-session subtask was 100%: this is motivated by the fact that (*i*) the dataset considered is small, and (*ii*) flexion and extension involve different MUs and the decomposition model already separates them very neatly, as it can be observed in Fig. 6.2.

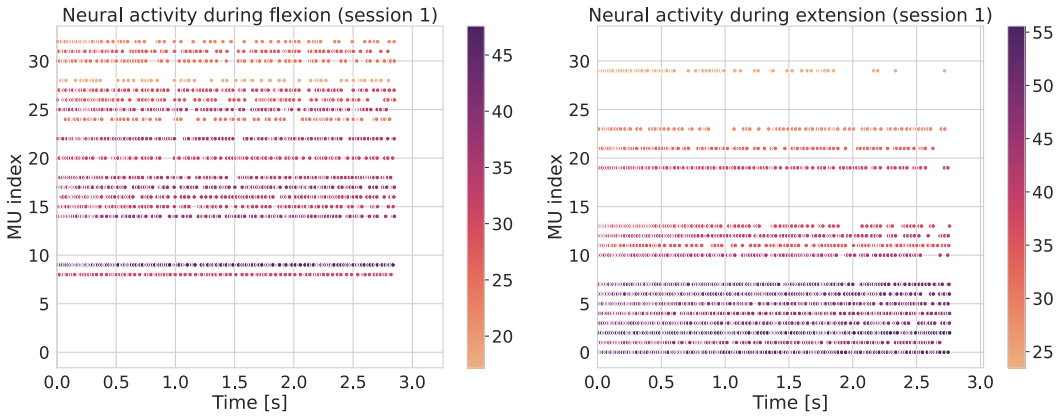

Figure 6.2: MUs extracted by the calibrated decomposition model, color coded by firing rate (i.e., ratio between the number of spikes of a MU and the duration of the contraction): each dot with coordinates (i, j) represent the MUAP generated by the j-th MU at time i. The firings characterizing a flexion gesture are shown on the left, those characterizing an extension are shown on the right: as it can be seen, the firing patterns are very different (in this example $M_t = 60$ and $f_s = 5120\,\text{Hz}$, but this difference in firing patterns was also observed for other values of M_t and f_s).

Inter-session subtask

Concerning the inter-session subtask, performance varied depending on the approach used, on the sampling frequency and on the number of target MUs, as shown in

Fig. 6.3. In general, neg-entropy ordering seems to be more reliable than firing rate ordering, especially when there are only few extracted MUs (i.e., $M_t \in 20$) and the sampling frequency is high (i.e., $f_s \in \{5120, 2560\}$): this can be confirmed also by a visual inspection of the firing pattern obtained using both ordering (Fig. 6.4). As expected, the approach yielding the best accuracy overall (almost always above 90%) is the re-calibration: since it extends the original separation matrix with new information from the new session, the ordering is guaranteed to be consistent. However, the position of the electrodes must be roughly the same between different recording session, otherwise the old separation vectors would become meaningless and may actually harm the decomposition.

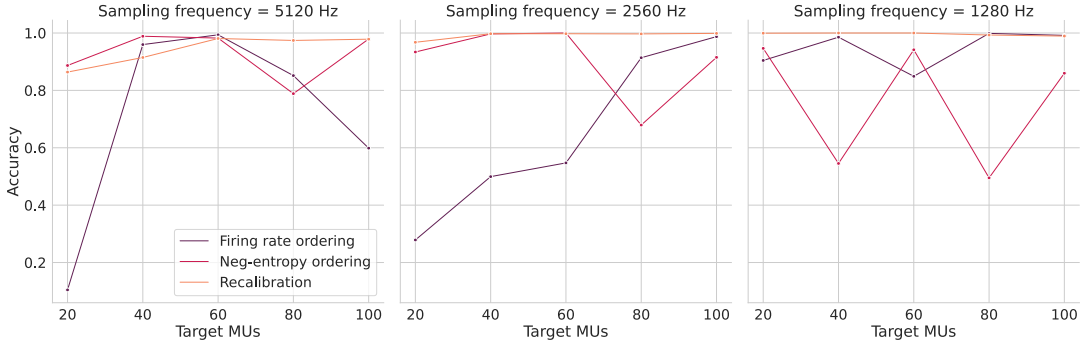

Figure 6.3: Inter-session test accuracy by varying decomposition approach and number of target MUs.

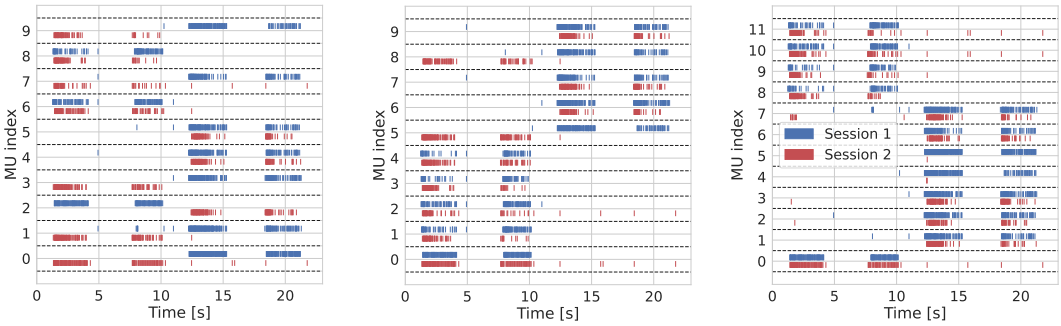

Figure 6.4: Comparison between firing rate ordering (left), negative entropy ordering (middle), and re-calibration (right) with $M_t = 20$: each bar with coordinates (i, j) represent the MUAP generated by the j-th MU at time i; the color indicates the session in which the MUAPTs were estimated. The MUAPs from 0 s to ∼12 s correspond to two repetitions of the flexion gesture, whereas the MUAPs from ∼12 s to ∼24 s correspond to two repetitions of the extension gesture: in the ideal case, the same MUs should be active during the same gestures in both session.

6.3 Task n. 3: online decomposition in real-time

The focus of this task is the profiling of the decomposition and classification pipeline for inference on the PULP platform.

6.3.1 Dataset acquisition

The complete system for sEMG signal acquisition employed in this task is presented in Fig. 6.5. The bracelet features eight pairs of dry gold-plated electrodes, interconnected through elastic bands to provide a flexible fit for different arm sizes. Each electrode uses a first amplification stage directly on the gold-plated pads to minimize the stray capacitance of the noise sensitive pads. Reference and ground electrodes are adhesive and placed at the elbow. Signals from the bracelet are sampled at 4 ksps with BioWolf16, a low-power HMI device for ExG signals [38, 39]. BioWolf16 features three main components: Mr. Wolf MCU, a Nordic SoC and two Texas Instruments analog front-ends (AFEs). Mr. Wolf is a 1 GFLOP/s energy-proportional PULP processor featuring a cluster of eight processing units and a fabric controller [40].

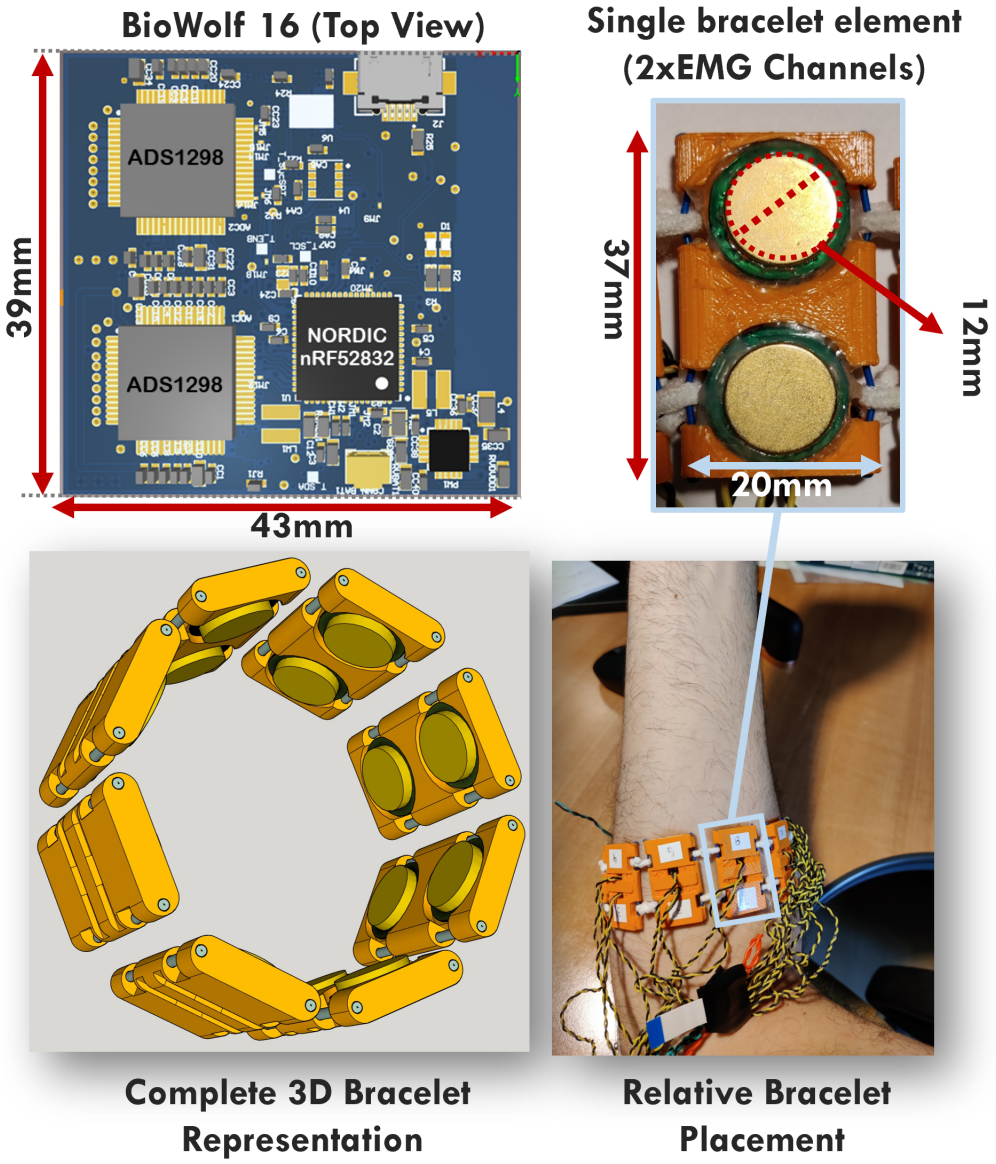

Figure 6.5: Complete acquisition system including a 3D representation of the 16-channel dry bracelet, its placement on the arm during experimentation, and the top view of the acquisition device (BioWolf16).

A single healthy subject participated in data collection, conducted in an office

environment, performing four hand gestures: `open hand`, `fist` (power grip), `pointing index`, `ok` (thumb up). The subject repeated every gesture two times during the experimental session. Gesture contractions were maintained for ∼5 s, followed by ∼5 s of rest. The bracelet was placed in the vicinity of the *flexor carpi ulnaris* muscle, as depicted in Fig. 6.5.

Gesture labelling

Contractions were labeled by computing the 200 ms-RMS of the sEMG then performing K-means (with $K = 2$ to discriminate rest and contractions) on the contractions with the lowest amplitude; this procedure finds the minimum sEMG threshold denoting the transition from rest to contraction and vice versa. The 400 ms windows centered in the rest-contraction transitions timing were considered as *transients*, namely sEMG segments whose corresponding gesture class cannot be defined exactly.

6.3.2 Dataset preparation

The decomposition described in the previous section was calibrated off-device on the raw sEMG data of one whole repetition per gesture, including transients and the adjacent rests. An extension factor $f_{\text{ext}} = 4$ and a number of target MUs $M_t = 60$ were employed: after the post-processing step, the model extracted 19 MUs.

The *training set* was built (*i*) considering all the 200 ms-slices (i.e., 800 samples with sampling at 4 ksps) corresponding to either contraction or rest (i.e., transients were discarded), from the first four repetitions of each gesture (and adjacent rests); (*ii*) performing online decomposition on these windows; and (*iii*) producing binary matrices $\mathbf{S}$ with size 19×800 where $s_{ij} = 1$ if the i-th MU fired at the j-th time sample, or $s_{ij} = 0$ otherwise. The *test set* was prepared in the same way, using the remaining one repetition of each gesture (and adjacent rests), yielding no overlap with the calibration and training data. The window length of 200 ms was chosen as a tradeoff between the algorithm's space and time complexity and sufficient information content for classification.

In the classification stage, the linear SVM and the lightweight MLP described in Section 4.3 were tested. The SVM processes 19-dimensional vectors containing the number of spikes for each MU, instead of whole 19×800 binary matrices, to reduce computation. The MLP processes full 19×800 binary matrices, and consists of three sequential fully connected layers: (*i*) a 4-neuron layer aggregating the time dimension, with parameters shared across channels; (*ii*) a 16-neuron layer aggregating the channel dimension; and (*iii*) a 5-neuron layer returning a score for each class. All hidden neurons have ReLU activation. The MLP was trained for 20 epochs with mini-batch size 32, cross-entropy loss function, Adam optimizer, initial learning rate 0.001, and weight decay 0.01 for regularization. The training of the two classifiers was performed off-device.

Since the rest class alone constitutes the 48% of the data, the class-imbalance was compensated by training both the linear SVM and the MLP using weighted random sampling based on the inverse frequency of each class, and assessing their performance based on both imbalanced and balanced accuracy. *The final accuracy assesses not only the classifier, but the decomposition and classification pipeline in a joint fashion*: a high accuracy means that the decomposition is able to extract spikes carrying valuable information about the movement.

6.3.3 Pipeline profiling

The speedup curves of the pipeline are shown in Fig. 6.6: using 8 cores, both decomposition and classification stages (either using the lightweight MLP or the SVM) achieve a speedup $> 6\times$; the sub-optimality compared to the ideal speedup is due to the frequent memory transfers between the fabric controller and the cluster (decomposition stage and MLP) and to sequential operations (SVM's one-vs-one voting).

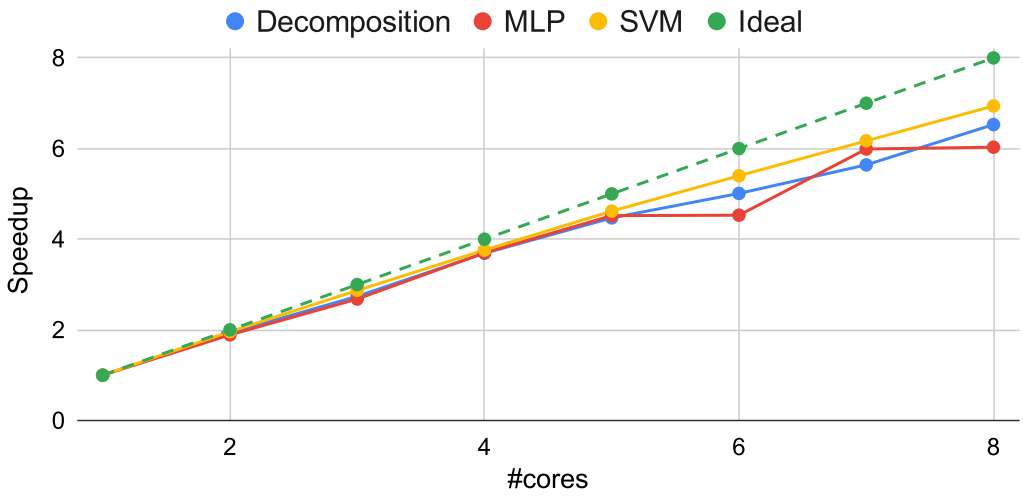

Figure 6.6: Speedup curve of the parallel implementation.

The profiling results of decomposition and classification (both for the linear SVM and the MLP option) are reported in Tabs. 6.2, 6.3.

STAGE		Memory (kB)	MAC	Latency Cycles	Latency Time (ms)	Energy (µJ)
Decomposition		73.0	4249k	2902k	29.02	536.87
Classification	option 1: MLP	18.1	62k	115k	1.15	21.27
	option 2: SVM	1.0	15k	20k	0.20	3.70

Table 6.2: Profiling of decomposition and classification on the PULP platform [8]. Latency and energy are referred to parallelization on all the cluster's 8 cores, running at 100 MHz.

STAGE		Test accuracy Unbalanced	Test accuracy Balanced
Classification	option 1: MLP	95.28%	92.48%
	option 2: SVM	95.38%	92.63%

Table 6.3: Classification accuracy of the SVM and lightweight MLP, averaged over 10 repetitions of training from scratch.

As to memory footprint, the decomposition stage is the most demanding, but the 73 kB it requires are within the typical memory budget of embedded platforms; downstream, the MLP only requires 18 kB, whereas the SVM has negligible footprint thanks to the linear kernel. A similar trend is found for computation energies, with

each decomposition employing 536.87 µJ, and the MLP and SVM consuming 21.27 µJ and 3.70 µJ per inference, respectively. Moreover, the number of multiply-accumulate (MAC) operations required involves a number of clock cycles that results in a latency below 50 ms for each decomposition and inference run. For this reason, all the 8 PULP cores running at 100 MHz—their most energy-efficient frequency [40]—are exploited. This makes the implementation suitable for working in real-time, since each execution processes a time window of 200 ms of data, which is twice the computation latency.

As to accuracy (cf. Tab. 6.3), both the linear SVM and the MLP yield an unbalanced accuracy $> 95\%$ and a balanced accuracy $> 92\%$ on the test data. An unbalanced accuracy higher than the balanced accuracy is due to the fact that the rest class is both the most represented in the data (48%) and the easiest to recognize, a common situation in sEMG-based ML, that motivates the use of balanced accuracy as a fairer metric. The obtained accuracy proves that this setup matches the performance of SoA implementations targeting a comparable number of classes but ignoring any physiological insight [20]. These results show that the decomposition stage reconstructs neural spikes which carry relevant information about the movement and are accurately separable by linear classifiers.

Power consumption and estimated battery life

The total power consumption and battery life of an embedded real-time implementation of the system was estimated, including the contribution of the AFEs, providing a complete power budgeting. The results show that the complete system can operate at a power envelope of ~ 17 mW (for either MLP or SVM), providing < 20 h of continuous operation with a single 100 mA h-battery charge. From this power, only 15% is due to the computation itself, thanks to Mr Wolf's parallel computing capabilities and energy efficiency. From this, $\sim 84\%$ is due to signal sampling, while the rest is due to computation. Using MLP or SVM does not change the overall power consumption, as their computational loads are negligible compared to decomposition (25× and 145× fewer clock cycles, respectively).

7 | Conclusions and future work

This master thesis tackles the problem of sEMG-based hand gesture recognition via the extraction of MUAPTs. Two scenarios were considered: one focused on off-device execution for intra-session and inter-session variability analysis, and one focused on profiling the end-to-end pipeline for inference on the PULP platform, an architecture organized in a cluster of RISC-V cores and designed targeting at high energy-efficiency.

Concerning the first scenario, a subset of the publicly available putEMG dataset was employed: in particular, only one subject and two opposite gestures was considered. For the classification stage, a standard 3-layer MLP was employed. The proposed pipeline proved to be very effective in the intra-session scenario: the MUAPTs estimated by the decomposition stage were linearly separable, thus the downstream MLP could easily classify them, resulting in an accuracy on the test set of 100%. In the inter-session scenario, the problem of ICA not guaranteeing a consistent MU ordering across different sessions was addressed with three approaches: two ordering criteria based on firing rate and negative entropy, and a re-calibration procedure that allows the decomposition model to retain information about previous recording sessions when decomposing new data. While both firing rate and neg-entropy orderings proved to be not very robust, the results obtained with the re-calibration procedure seem promising, as this approach yielded an accuracy on the test set up to 99.4%.

As to the second scenario, the sEMG dataset was acquired from one subject using BioWolf16, a low-power HMI device for ExG signals, and consisted of five classes (four hand gestures and rest). Both the calibration of the decomposition model and the training of SVM and lightweight MLP were performed off-device: the SVM and the lightweight MLP yielded an accuracy on the test set of 92.6% and 92.5%, respectively. A parallel, end-to-end pipeline for inference was then implemented and profiled on the PULP architecture, showing a latency $< 50\,\text{ms}$ and an energy consumption $< 1\,\text{mJ}$.

These results prove that the system proposed in this thesis is not only suitable for real-time execution on resource-constrained embedded platforms, but is also capable of matching the recognition accuracy of SoA approaches, while also giving some physiological insight on the neuromuscular spikes underlying the sEMG.

Future work will continue this research line, testing the generalization capabilities of the system on more subjects, more gestures and more recording sessions.

Acknowledgements

I would first like to thank my thesis supervisor, Prof. Francesco Conti, and my co-supervisor, Prof. Simone Benatti, for their assistance, their time and their valuable advices on the development of this thesis work.

I would also like to express my gratitude to my co-supervisors Marcello Zanghieri and Dr. Davide Schiavone for their constant guidance in the research work, the valuable support they gave me. I especially thank Marcello Zanghieri for the detailed and timely comments provided during the writing of the thesis paper.

An acknowledgement also goes to Dr. Elisa Donati for welcoming me at the Institute of Neuroinformatics in Zurich and for assisting me during my research period abroad.

Finally, I would like to recognize the invaluable moral support and continued encouragement received from my family, friends and all those who have been close to me.

Bibliography

[1] James Cannan and Huosheng Hu. "Human-machine interaction (HMI): A survey". In: *University of Essex* (2011).

[2] Roberto Meattini et al. "Experimental evaluation of a sEMG-based human-robot interface for human-like grasping tasks". In: *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. 2015, pp. 1030–1035. DOI: 10.1109/ROBIO.2015.7418907.

[3] Lin Guo et al. "Human-Machine Interaction Sensing Technology Based on Hand Gesture Recognition: A Review". In: *IEEE Transactions on Human-Machine Systems* 51.4 (2021), pp. 300–309. DOI: 10.1109/THMS.2021.3086003.

[4] Wei Li et al. "Gesture Recognition Using Surface Electromyography and Deep Learning for Prostheses Hand: State-of-the-Art, Challenges, and Future". In: *Frontiers in Neuroscience* 15 (2021). ISSN: 1662-453X. DOI: 10.3389/fnins.2021.621885.

[5] Panagiotis Tsinganos et al. "Deep Learning in EMG-based Gesture Recognition". In: Sept. 2018, pp. 107–114. DOI: 10.5220/0006960201070114.

[6] Gonzalo A. García et al. "Decomposition of Synthetic Multi-channel Surface-Electromyogram Using Independent Component Analysis". In: *Independent Component Analysis and Blind Signal Separation*. Ed. by Carlos G. Puntonet and Alberto Prieto. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 985–992. ISBN: 978-3-540-30110-3.

[7] Christopher Spiewak et al. "A Comprehensive Study on EMG Feature Extraction and Classifiers". In: *Open Access Journal of Biomedical Engineering and its Applications* 1 (Feb. 2018). DOI: 10.32474/OAJBEB.2018.01.000104.

[8] Francesco Conti et al. "PULP: A Ultra-Low Power Parallel Accelerator for Energy-Efficient and Flexible Embedded Vision". In: *Journal of Signal Processing Systems* 84.3 (2016), pp. 339–354. ISSN: 1939-8115. DOI: 10.1007/s11265-015-1070-9.

[9] Carlo J. De Luca and Emily Hostage. "Relationship Between Firing Rate and Recruitment Threshold of Motoneurons in Voluntary Isometric Contractions". In: *Journal of neurophysiology* 104.2 (2010), pp. 1034–46. DOI: 10.1152/jn.01018.2009.

[10] Francesco Negro et al. "Multi-channel intramuscular and surface EMG decomposition by convolutive blind source separation". In: *Journal of Neural Engineering* 13.2 (Feb. 2016), p. 026027. DOI: 10.1088/1741-2560/13/2/026027.

[11] Alessandro Del Vecchio et al. "Tutorial: Analysis of motor unit discharge characteristics from high-density surface EMG signals". In: *Journal of Electromyography and Kinesiology* 53 (2020), p. 102426. ISSN: 1050-6411. DOI: 10.1016/j.jelekin.2020.102426.

[12] Aleš Holobar and Damjan Zazula. "Multichannel Blind Source Separation Using Convolution Kernel Compensation". In: *IEEE Transactions on Signal Processing* 55.9 (2007), pp. 4487–4496. DOI: 10.1109/TSP.2007.896108.

[13] Ilja Kuzborskij et al. "On the challenge of classifying 52 hand movements from surface electromyography". In: *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. 2012, pp. 4931–4937. DOI: 10.1109/EMBC.2012.6347099.

[14] Kevin Englehart and Bernard Hudgins. "A robust, real-time control scheme for multifunction myoelectric control". In: *IEEE Transactions on Biomedical Engineering* 50.7 (2003), pp. 848–854. DOI: 10.1109/TBME.2003.813539.

[15] Manfredo Atzori et al. "Electromyography data for non-invasive naturally-controlled robotic hand prostheses". In: *Scientific Data* 1.1 (Dec. 2014), p. 140053. ISSN: 2052-4463. DOI: 10.1038/sdata.2014.53.

[16] Manfredo Atzori et al. "Characterization of a Benchmark Database for Myoelectric Movement Classification". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 23.1 (2015), pp. 73–83. DOI: 10.1109/TNSRE.2014.2328495.

[17] Ki-Hee Park and Seong-Whan Lee. "Movement intention decoding based on deep learning for multiuser myoelectric interfaces". In: *2016 4th International Winter Conference on Brain-Computer Interface (BCI)*. 2016, pp. 1–2. DOI: 10.1109/IWW-BCI.2016.7457459.

[18] Weidong Geng et al. "Gesture recognition by instantaneous surface EMG images". In: *Scientific Reports* 6.1 (Nov. 2016), p. 36571. ISSN: 2045-2322. DOI: 10.1038/srep36571.

[19] Yunan He et al. "Surface EMG Pattern Recognition Using Long Short-Term Memory Combined with Multilayer Perceptron". In: *2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. 2018, pp. 5636–5639. DOI: 10.1109/EMBC.2018.8513595.

[20] Marcello Zanghieri et al. "Robust Real-Time Embedded EMG Recognition Framework Using Temporal Convolutional Networks on a Multicore IoT Processor". In: *IEEE Transactions on Biomedical Circuits and Systems* 14.2 (2020), pp. 244–256. DOI: 10.1109/TBCAS.2019.2959160.

[21] Bojan Milosevic et al. "Exploring Arm Posture and Temporal Variability in Myoelectric Hand Gesture Recognition". In: *2018 7th IEEE International Conference on Biomedical Robotics and Biomechatronics (Biorob)*. 2018, pp. 1032–1037. DOI: 10.1109/BIOROB.2018.8487838.

[22] Jonathon Shlens. *A Tutorial on Independent Component Analysis*. 2014. DOI: 10.48550/ARXIV.1404.2986.

[23] Aapo Hyvärinen et al. "What is Independent Component Analysis?" In: *Independent Component Analysis*. John Wiley & Sons, Ltd, 2001. Chap. 7, pp. 145–164. ISBN: 9780471221319. DOI: 10.1002/0471221317.ch7.

[24] Lishan Zhong et al. “Segmentation of Individual Trees From TLS and MLS Data”. In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 10 (June 2016), pp. 1–14. DOI: 10.1109/JSTARS.2016.2565519.

[25] Aapo Hyvärinen et al. “ICA by Maximization of Nongaussianity”. In: *Independent Component Analysis*. John Wiley & Sons, Ltd, 2001. Chap. 8, pp. 165–202. ISBN: 9780471221319. DOI: 10.1002/0471221317.ch8.

[26] Johan Thomas et al. “Time-domain fast fixed-point algorithms for convolutive ICA”. In: *IEEE Signal Processing Letters* 13.4 (2006), pp. 228–231. DOI: 10.1109/LSP.2005.863638.

[27] Emanuele Formento et al. “Skilled independent control of individual motor units via a non-invasive neuromuscular-machine interface”. In: *Journal of Neural Engineering* 18.6 (Nov. 2021), p. 066019. DOI: 10.1088/1741-2552/ac35ac.

[28] Deren Y. Barsakcioglu et al. “Control of Spinal Motoneurons by Feedback From a Non-Invasive Real-Time Interface”. In: *IEEE Transactions on Biomedical Engineering* PP (June 2020), pp. 1–1. DOI: 10.1109/TBME.2020.3001942.

[29] Chenyun Dai and Xiaogang Hu. “Independent component analysis based algorithms for high-density electromyogram decomposition: Systematic evaluation through simulation”. In: *Computers in Biology and Medicine* 109 (2019), pp. 171–181. ISSN: 0010-4825. DOI: 10.1016/j.compbiomed.2019.04.033.

[30] Piotr Kaczmarek et al. “putEMG—A Surface Electromyography Hand Gesture Recognition Dataset”. In: *Sensors* 19.16 (2019). ISSN: 1424-8220. DOI: 10.3390/s19163548.

[31] Xinyu Jiang et al. “Open Access Dataset, Toolbox and Benchmark Processing Results of High-Density Surface Electromyogram Recordings”. In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 29 (2021), pp. 1035–1046. DOI: 10.1109/TNSRE.2021.3082551.

[32] Corinna Cortes and Vladimir Vapnik. “Support-vector networks”. In: *Machine Learning* 20.3 (Sept. 1995), pp. 273–297. ISSN: 1573-0565. DOI: 10.1007/BF00994018.

[33] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[34] Fabian Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[35] Nazareno Bruschi et al. “GVSoC: A Highly Configurable, Fast and Accurate Full-Platform Simulator for RISC-V based IoT Processors”. In: *2021 IEEE 39th International Conference on Computer Design (ICCD)*. 2021, pp. 409–416. DOI: 10.1109/ICCD53106.2021.00071.

[36] Dirk Merkel. “Docker: lightweight linux containers for consistent development and deployment”. In: *Linux journal* 2014.239 (2014), p. 2.

[37] Mohammad Reza Mohebian et al. “Non-invasive Decoding of the Motoneurons: A Guided Source Separation Method Based on Convolution Kernel Compensation With Clustered Initial Points”. In: *Frontiers in Computational Neuroscience* 13 (2019). ISSN: 1662-5188. DOI: 10.3389/fncom.2019.00014.

[38] Victor Kartsch et al. “Biowolf: A sub-10-mw 8-channel advanced brain–computer interface platform with a nine-core processor and ble connectivity”. In: *IEEE transactions on biomedical circuits and systems* 13.5 (2019), pp. 893–906. DOI: 10.1109/TBCAS.2019.2927551.

[39] Riccardo Donati et al. “BioWolf16: a 16-channel, 24-bit, 4kSPS Ultra-Low Power Platform for Wearable Clinical-grade Bio-potential Parallel Processing and Streaming”. In: *2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*. 2022, pp. 2518–2522. DOI: 10.1109/EMBC48229.2022.9871898.

[40] Antonio Pullini et al. “Mr. Wolf: A 1 GFLOP/s Energy-Proportional Parallel Ultra Low Power SoC for IOT Edge Processing”. In: *ESSCIRC 2018 - IEEE 44th European Solid State Circuits Conference (ESSCIRC)*. 2018, pp. 274–277. DOI: 10.1109/ESSCIRC.2018.8494247.