POLITECNICO DI TORINO

DIPARTIMENTO DI INGEGNERIA MECCANICA E AEREOSPAZIALE (DIMEAS)

Master's Degree Course in Biomedical Engineering

Towards an Electromyographic Armband: an Embedded Machine Learning Algorithms Comparison

Supervisors
Prof. Danilo Demarchi
Ph.D. Paolo Motto Ros
M.Sc. Fabio Rossi
M.Sc. Andrea Mongardi

Candidate
Matteo Tolomei

TORINO, December 2020

Abstract

Gesture recognition is a trending topic in modern technology, being used to control mobile apps, robotics and also videogames.
Many approaches are in use to detect gestures and make them suitable for digital processing and machine learning classifier. For this specific application, data are collected starting from the surface ElectroMyoGraphic (sEMG) signals, obtained applying non-invasive electrodes on a selected skin area.
The acquisition setup makes gesture recognition also suitable for Human-Machine Interface (HMI), like prosthesis and robotic limb control.

This thesis work offers a wide overview of the machine learning algorithms used for hand gesture recognition. Considering low-energy consumption as key feature, the system is based on an event-driven approach focused on the Average Threshold Crossing (ATC) information. This feature is obtained averaging, in a fixed time window, the number of voltage threshold crossings by the sEMG signal, which can also be seen as an index of muscle activation.

The first analyzed dataset, acquired with custom made boards, has involved 25 healthy people, each one performing five gesture over five sessions. Algorithms like Neural Network (NN), K-means, Support Vector Machines (SVM), Random Forest (RF) and Gaussian Mixture Model (GMM) Naïve Bayes were tested and implemented on a microcontroller (i.e., Ambiq Apollo 3 Blue with a Cortex M4-F processor) for real-time applications.
During the assessment, requirements like power consumption and a system latency below 300ms were taken into consideration.

For all the algorithms the system latency was way below the 300ms; in particular: 2.56ms for neural network, 185.49 µs for random forest, 140.46 µs for GMM Naïve Bayes, 61.92 µs for K-means and 54.84 ms for support vector machines.
Power consumption analysis have been performed on the MCU obtaining: 0.54 mW for NN, 0.5131 mW for K-means, 0.9324 mW for SVM, 0.5126 mW for RF and 0.3758 mW for Naïve Bayes.

Further investigations were made towards the design of an armband. Electrodes (i.e. two for signal acquisition and one for reference) placement on the forearm has been deeply analyzed and an optimal setup was reached.
Bringing the acquisition channels up to seven, it was possible to increase the number of recognized gestures for a total of seven active poses plus the resting position. For these new data, a preliminary machine learning study has been conducted reaching an accuracy of 88%.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Muscular System

The muscular system is structured to allow the movement of the body and maintaining the posture to safeguard the underlying organs. There are three different types of muscle tissue, each one has is own characteristics [1]:

- **Skeletal muscle** is responsible for the movements. It is controlled by the peripheral portion of the Central Nervous System (CNS). Thus, these muscles are under conscious, or voluntary, control;
- **Smooth muscle** is an involuntary tissue directly controlled by the autonomic nervous system, meaning that they are incapable of being moved by conscious thoughts;
- **Cardiac muscle** is only found in the heart, where it performs coordinated contractions allowing the blood to move through the circulatory system. Similar to the smooth muscle, is movement is involuntary.

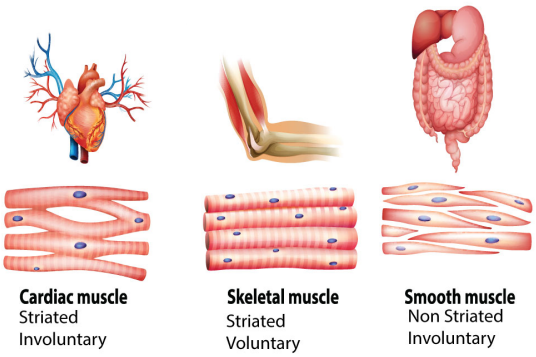


Figure 1.1: Different types of skeletal muscle [2].

1.1.1 The Skeletal Muscle

A good knowledge of the main characteristics of skeletal muscles helps to deeply understand the anatomy behind the movement and most important the related EMG signal.
Each skeletal muscle is the result of integration between the muscle fibers, blood vessels, nerve fibers, and connective tissue. Each skeletal muscle is wrapped in three connective tissue layer, providing structure and subdividing it in muscle fibers (Figure 1.2).
A dense and irregular tissue, called epimysium, encloses each muscles preserving the structural integrity while allowing indipendency from the surrounding [3]. It is also responsible for the contraction and the movement of the muscles.

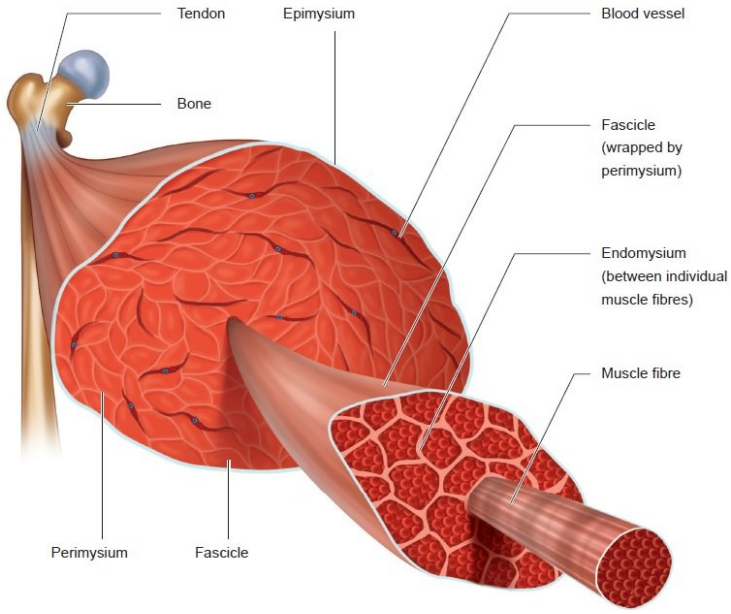


Figure 1.2: Structure of a skeletal muscle [4].

Bundles of muscle fibers, covered by a connective tissue called perimysium, constitute the fascicle. The nervous system could trigger a single fascicle, allowing many different types of movements. The last connective tissue layer is the endomysium which surrounds each muscle fiber and plays an important role for transfer the strength produced by the single muscle fiber up to the tendons.
Each muscle fiber is elongated and in most of the cases thinned out at the ends. Inside each muscle fiber is located the cytoplasm, containing glycosomes, myoglobin and sarcoplasmic reticulum. This multinuclear cell is also called sarcolemma.
Inside the cytoplasm, mitochondria have the main role in the production of the necessary energy: starting from the substances coming from the circulatory system, the ATP (*Adenosine TriPhosphate*) is obtained through chemical reactions. Based

of the fiber's function, mitochondria can increase their presence up to 20% of total volume. A myofibril, known also as muscle fiber, is the functional unit of a muscle cell which in turn can be split in sarcomeres.

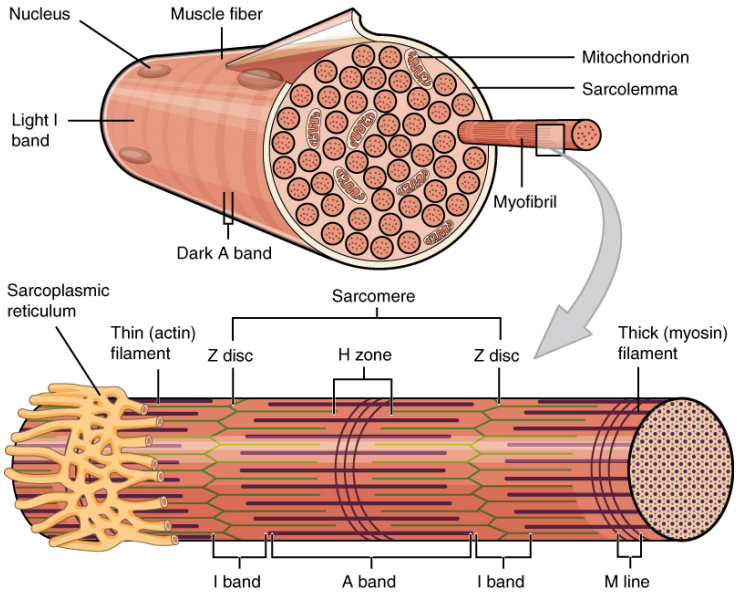


Figure 1.3: Inner view of a sarcomere [5].

A sarcomere is made of light and dark bands reoccurring in series, giving the cell a striated appearance. It is defined as the portion between two very dark-colored bands called Z-lines which also act as an anchoring point for the actin filaments. Around the Z-lines there are Isotropic bands (I-bands), which are the lighter ones and made of thin actin filaments [6].
Following the I-band are the dark Anisotropic bands (A-bands) containing myosin. The sarcomere central area is occupied by a brighter area (H-bands, from the German *heller*) due to the absence of actin filaments. Within the H-zone is a thin line (M-line, from the German *Mittelscheibe*) that helps to connect myosin filaments of the two side. During a contraction, the I-band and the H-band increase their thickness, while the A-band keeps its shape.

The focus of this work is related to analyze the acquired sEMG signals of the forearm muscles while performing gestures. Having a good knowledge of the forearm muscles helps to better understand the results and the electrode placement during the recording sessions. In the following pages, muscles' forearm are listed and briefly described.

1.1.2 Forearm muscles

The forearm is divided in posterior and anterior compartment, each one further divided into layers [7].

- **Anterior compartment**
  The anterior compartment runs along the inside of the forearm. The muscles in this area are mostly involved with wrist flexion and fingers mobility, as well as for the movement of pronation and supination of the forearm. There are eight muscles subdivided in three layers: *superficial, intermediate and deep layer* (Figure 1.4).

  The **superficial layer** contains four muscles:

  – *flexor carpi ulnaris* is the most medial muscle in the superficial layer, and it has a center role for wrist movements (*wrist flexion and adduction*);
  – *palmaris longus* is missing in almost 15% of the population. When it is present, it fits between the *flexor carpi ulnaris* and *flexor carpi radialis* muscles. It is the most superficial forearm muscle and has a secondary role like tendon transfers;
  – *flexor carpi radialis* is found close to the palmaris longus, and whenever it is present is found in a lateral position. It is responsible for the wrist flexion and abduction;
  – *pronator teres* is a functionally important muscle that contains two heads, separating the ulnar artery from the ulnar head of pronator teres. It acts for the pronation of the forearm.

  The **intermediate layer** contains only the *flexor digitorum superficialis* which function is the flexion at the proximal interphalangeal, metacarpophalangeal and wrist joints.

  The **deep layer** of the anterior compartment is made of three muscles:

  – *flexor pollicis longus* is a long muscle of the forearm. His main contribuition is during the gripping being responsible for the flexion of the thumb at the interphalangeal joint;
  – *flexor digitorum profundus* is the most voluminous muscle of the forearm. It has a central role for a wrist flexor and for powerful finger, one of the most crucial elements responsiblr for strength of the grip and the pincer grip;
  – *pronator quadratus* is a small, flat, quadrilateral muscle. As the name suggests, the main function of this muscle is to assist and stabilize the forearm pronation.

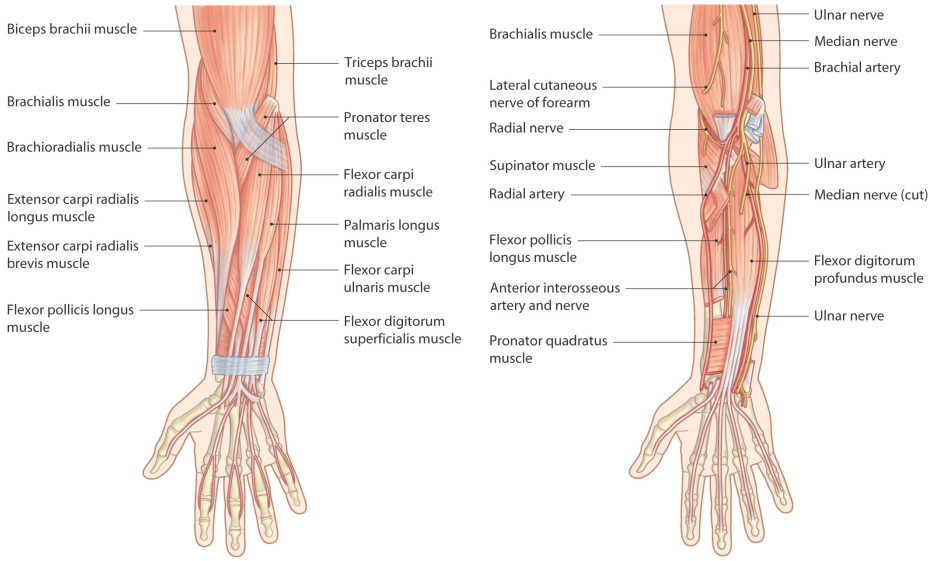


Figure 1.4: Muscles of the anterior compartment of the right forearm [8].

- **Posterior compartment**
  This compartment mainly produces wrist and/or finger extension, and thumb abduction. It is divided in *superfical* and *deep layer* for a total of twelve muscles (Figure 1.5).

  The **superficial layer** is made of seven muscles:

  - *brachioradialis* produces minimal flexion at the elbow, but has a primary role as elbow fixator
  - *extensor carpi radialis longus* extends between the humerus and the second metacarpal bone. It is responsible for the extension and abduction of the hand, vital movements for the hand gripping
  - *extensor carpi radialis brevis* has a main role for the extension and abduction of the wrist
  - *extensor digitorum (communis)* starts from a very short common muscle belly that splits into four individual muscle bellies, each giving rise to a single tendon. It activates the extension of digits II-V
  - *extensor digiti minimi (extensor digiti quinti proprius)* is responsible for the extension of digit V
  - *extensor carpi ulnaris* with the *extensor carpi radialis brevis* coordinate the extension and abduction of the wrist
  - *anconeus* provides support to the other muscles during the extension

The **deep layer** of the posterior forearm contains five muscles, all of them starting by the posterior interosseous nerve:

- *supinator* is a spiral muscle that curls around the proximal part of radius. As the name suggests, it helps to supinate the forearm
- *abductor pollicus longus* contributes to thumb abduction and has a role in abduction at the wrist
- *extensor pollicus longus* passes around the dorsal radial tubercle before reaching the thumb.
- *extensor pollicus brevis* along with the *extensor pollicus brevis* controls the extension of the carpometacarpal and metacarpophalangeal joints of digit I
- *extensor indicis* allows the extension of digit II

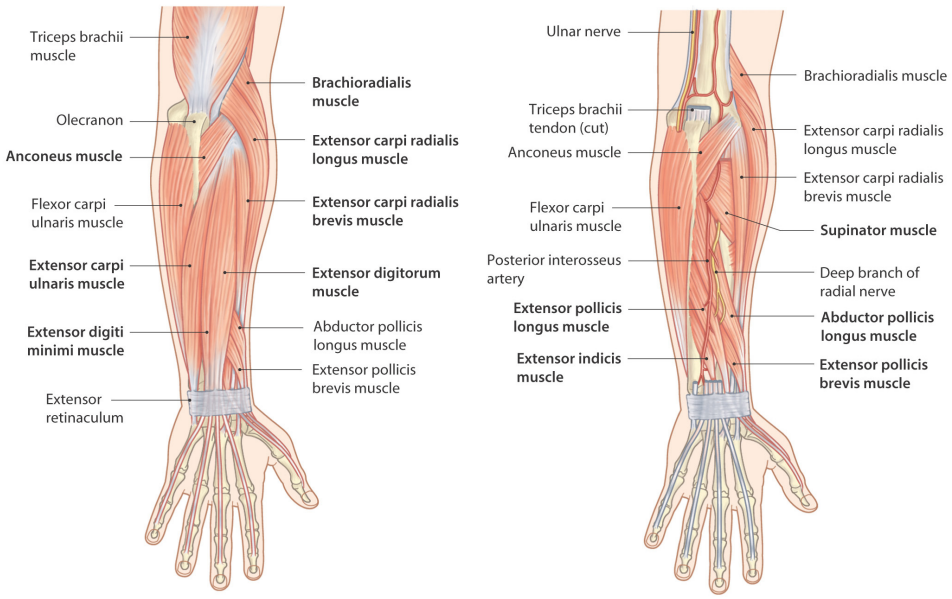


Figure 1.5: Muscles of the posterior compartment of the right forearm [8].

1.2 Electromyography (EMG)

During a muscolar contraction, the depolarization and repolarization of the external membrane of muscle fibers generate an electrical signal that can be recorded with the *electromyography (EMG)* technique providing information on the muscular activity.
The signal amplitude can be affected not only by the anatomical features but it

depends also on the type of electrodes used and their position, as it will be further discussed in 1.2.1. The measured bioeletrical signal, called *Motor Unit Action Potential* (MUAP) is the spatial-temporal summation of the individual action potentials generated by the depolarizations of the muscle fibers of a motor unit.
A sequence of activation by the same motor unit generates a train of action potential, called MUAPT. By looking at the graph representing the behaviour of a single *Intracellular Action Potential* (IAP) (Figure 1.6) it is possible to identify three phases. It starts with an initial depolarization phase, then a repolarization phase with almost the same slope. Before returning to the resting state, a long iperpolarization phase occurs. During a state of muscular fatigue, the shape can change significantly making difficult to identify the three stages: in particular, the repolarization phase become slower increasing the peak width.
The action potential of the motor unit, triggered by an external stimulus or by the central nervous system (CNS), generates some variations in the electromagnetic field around the muscle fiber. With a technology called surface electromyography (sEMG), it is possible to record these fluctuations with an invasive or non-invasive approach.

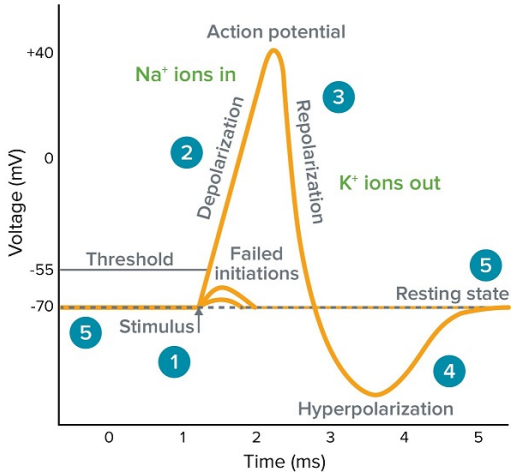


Figure 1.6: Intracellular Action Potential [9].

1.2.1 Surface - Electromyography (sEMG)

With a non-invasive methodology, called *surface electromyography*, EMG signal recording is possible by simply placing the electrode on the surface of the skin.
The acquired bio-signal is the sum of all the motor units, below the electrode, recruited in the movement. The recording is performed using a single or a electrodes matrix placed above the muscles to be tested.
The most common applicstions for sEMG technique are:

- temporal characterization of a movement, studying the muscular activation;

- feedback for the patient on how well he is performing the assigned movement during a rehabilitation session;
- monitoring a group of muscle or just a single one.

From a more technical point of view, the amplitude of the sEMG signal can span between 0mV and 10mV. The frequency range spectrum is 6–500 Hz, with major contribution in the 20Hz - 150Hz slot.

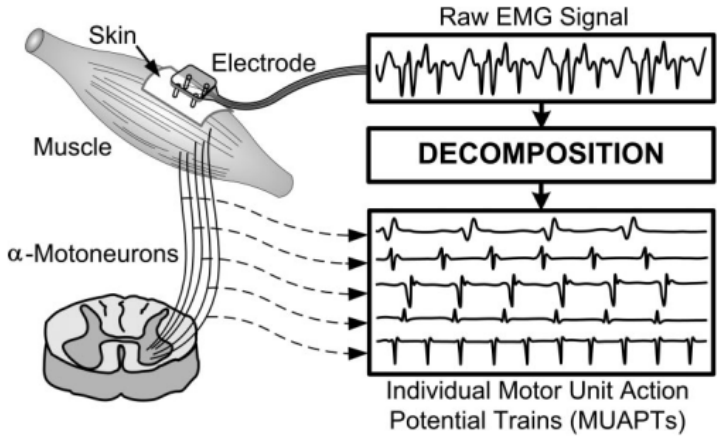


Figure 1.7: sEMG technique [10].

A signal with this low amplitude and a wide frequency range, during the recording session can be easily affected by many source of noise:

- a noise, called **movement artifact**, usually lower than 10 Hz is generated by the friction between the skin, the adjacent under skin substrates and the electrode. In the preprocessing phase, this noise component can be moved out from the sEMG spectrum;
- **cross-talk** due to the presence of multiple muscles in the area, can affect the recording of a single muscle activity. This noise can only be lowered with an accurate placement of the electrodes;
- **muscular fatigue** is another contribution to take in account. In fact, as written earlier, in a fatigue state the muscle activity will decrease, causing the signal to have a lower frequency and amplitude;
- the **environment noise** can affect the recording making more difficult to have a clean signal. One of the most dangerous could be the noise at 50 Hz introduced by the voltage supply. It can be removed but it is always a trade-off between having a clear signal and the loss of informations in that frequency range.

On the market, there are many different types of electrodes based on the type of exam and their technology. The electrodes can be divided in two main species: *wet* and *dry*. The most common are made of silver/silver chloride (Ag/AgCl), silver chloride (AgCl), silver (Ag) or gold (Au).
The Ag/AgCl type is the most used thanks to its property of not be polarizable. The use of a conductive gel skin helps to prevent and reduce the noise due to the friction between the electrode and the underneath skin layer.
The gel layer increases also the mechanical stability at the electrode-skin interface. There are disposable electrodes with a built-in gel layer for reducing the application time. They are available in all sizes and shapes: from a few millimeters to centimeters, where the dimension influences the field of application and the spatial resolution.

Figure 1.8: Example of pre-gelled EMG electrode [11].

Dry electrodes represent a good alternative and in the last years have been widely used for the acquisition of biopotentials. Using them avoids the use of conductive gel, which is ideal for long-term analysis. The absence of the gel prevents also skin irritation and the need to reapply the gel every few hours in a long procedure.
The drawback of using dry electrodes is the absence of any electrolyte between the metal and the skin, requiring a more sophisticated signal conditioning circuit to control the skin-electrode impedance.

Figure 1.9: Example of dry electrode [12].

1.3 The Average Threshold Crossing Technique

The most common chain structure for a commercial data acquisition device is composed by:

- electrodes arrays for monopolar or bipolar configuration for acquiring the sEMG simply by putting them on the skin surface;
- first stage circuit, made of amplifiers and filters, responsible for the preprocessing of the signal. A right components selection let pass only the meaningful frequency, obstructing the undesired one;
- a digital section where an Analog-to-Digital-Converter (ADC), controlled by a microcontroller, samples the conditioned signal. The data could be stored locally or transmitted to a computer.

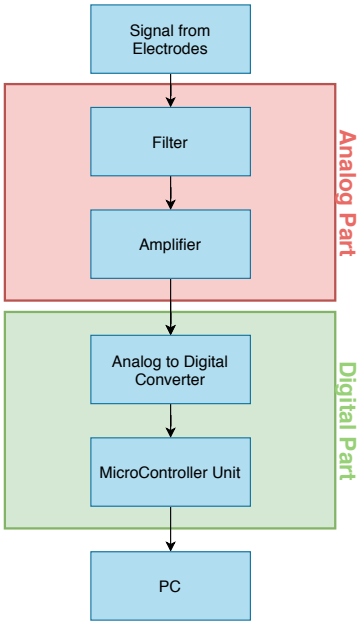


Figure 1.10: Standard structure of sEMG acquisition chain.

The particular shape and the systematically electrical variations of the IAP opened the way to different approaches of analysis. Some researchers have formalized a way to study it based on a threshold.
Among the others, the Average Threshold Crossing (ATC) technique will be described in the following lines. It is an event-driven technique where the Threshold Crossing (TC) counter increases whenever the myoelectrical signal crosses a certain set threshold. This new approach, described in detail by [13, 14, 15] is, in

some ways, a game-changer compared to the other, especially in terms of power consumption. It could easily be implemented directly in hardware with a circuit made of voltage comparators.
The output is determined based on a two signals comparison: between the set threshold and the incoming sEMG signal. This step can be found after the analog part (filter and amplifier). The output signal will have information based on the time domain and not on the analog value, leaving at the end a quasi-digital signal. The output state could be high when the threshold is crossed and low in all the other circumstances.

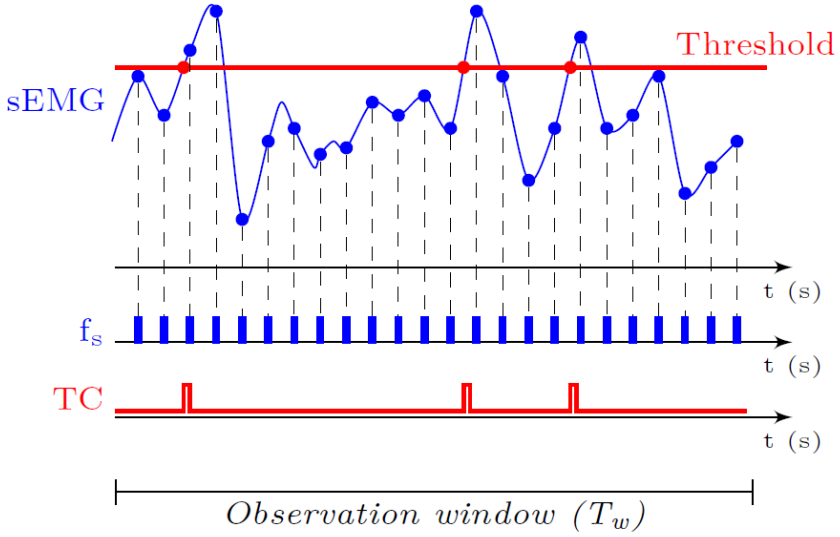


Figure 1.11: Transmission cost comparison between sampling the signal and using the ATC technique [16].

The ATC parameter is defined as how many times the sEMG signal crosses the set threshold, in a fixed time window, divided by the duration of the window itself. The threshold crossing can be defined as an event, easily detectable in hardware employing a comparator and a threshold.
In modern application, to avoid problems due to cables obstructing movements, data are sent dynamically through wireless. An event-driven approach drastically reduces the number of packets sent, while increasing the battery life.
In Figure 1.11 is shown, given a fixed time window, the different number of packets sent by the two approaches: standard sampling and the event-driven approach. Using the TC allows the transmission of a single ATC value every time window, avoiding the need of a constant transmission. This approach, like it has already been demonstrated [15], could also work with a dynamic threshold, being adaptable to many environment.

1.4 State of the art

Gesture recognition is a trending topic in many technological fields: videogames, medical, and entertainment. The possibility to perform action simply by moving the hand freely is pushing scientists and researchers to the boundaries of technology. Different technologies are used for this purpose: starting from sEMG analysis to the infrared cameras and even using an eye-tracking system.
All these approaches have in common several steps before the real prediction of the gesture begins: data preprocessing and features extraction.
In the following pages the most interesting literature works based on sEMG signals are listed, focusing on the accuracy and the computational time.

Momen et al. [17] group realized a classifier that allows the user to make all the desired movements, without any predefined set of gesture. Only two sEMG features were used during the implementation: the RMS and its logarithm extracted over a 200 ms window.
During the testing phase was reached a final accuracy of 87% with a standard deviation of 13% and no constraints about the type of movements. The classifier was based on a fuzzy clustering C-means method.

Another great study was published by *Shenoy* et al. [18] in which they reduced the number of used features, exploiting only the RMS. This approach gave good results with a predefined set of movements. The RMS value has been extracted over 128 samples windows.
With a sampling frequency of 2048 Hz, a 60 ms window is obtained allowing to control of a robotic arm in real-time. In this work, a Support Vector Machine (SVM) classifier is presented and during the cross-validation process accuracy of 90% was reached.

SVM classifier offered other good results in a work proposed by *Lucas* et al. [19]. This time the Discrete Wavelet Transform (DWT) was used as a unique feature, which increased the general performances while reducing the system flexibility and having longer computational time.
With a misclassification rate of 5% for the six classes it was necessary to use 512 samples, bringing the total system latency up to 250 ms still suitable for real-time applications. The overall performances of the proposed system were around 95%, a better result compared with the previous ones.

Coté-Allard et al. [20] proposed a hand gesture recognition system based on Convolutional Neural Network (CNN). During this study, subjects were asked to perform 6 active gestures and the sEMG signals were acquired with the Myo armband and a time window of 260 ms.

Due to the low sampling frequency of the Myo and the decision of prefering accuracy over time, the total system latency was set at 300 ms. This complex algorithm brought the accuracy of the system up to 97%.

A different approach was used by *Zhang* et al. [21] based on Artificial Neural Network (ANN). A 256 ms time window is implemented and five features were extracted: MAV, slope sign change (SSC), waveform length (WL), root mean square (RMS), and Hjorth parameter (HP).
The system was tested on 5 active gestures with a global accuracy of 98% and with a response time of 227.76 ms, which is lower than the maximum permissible latency of 300 ms for real-time classification.

Qi et al. [22] proposed a model for hand gesture recognition based on sEMG acquired from 16 electrodes.
RMS, wavelength (WL), sample entropy (SampEn), and median amplitude spectrum (MAS) were used as features, leaving the total information with 64 dimensions (16 channels x 4 features).
Using principal component analysis (using 3 principal components) and General Regression Neural Network (GRNN), obtained a system that recognizes 9 gestures with an accuracy of 95% and a recognition time of 190ms.

Chapter 2

Classification Algorithms

Several supervised and unsupervised machine learning algorithms were used for the system validation. In the following pages a briefly theorical introduction of them is given.

2.1 Neural Network

A neural network can be define as a series of algorithms that attempt to recognize underlying relationships between data through a process that takes inspiration from how the human brain works.
In the algorithm a neuron is define as a mathematical function that receives and classifies information following a particular architecture [23]. The network has a strong similarity to statistical methods such as curve fitting and regression analysis.

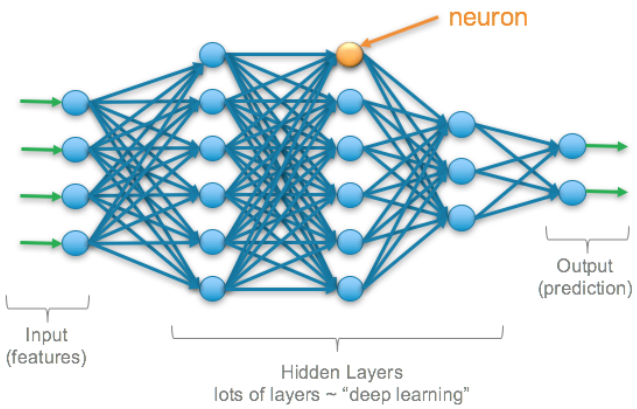


Figure 2.1: Example of a neural network [24].

A neural network is made of layers of interconnected nodes. Each node is a perceptron and is similar to a multiple linear regression. The signal produced by a multiple linear regression is feed, by the perceptron, into an activation function, which may be nonlinear [25]. If perceptrons are arranged in interconnected layers the architecure is called multi-layered perceptron (MLP). The input layer takes the features as input. The output layer has classifications for the input features based on their patterns.

2.2 Support Vector Machine (SVM)

A Support Vector Machine (SVM) is a supervised machine learning algorithm [26] that can be used either for classification or regression problems.
The key idea of SVMs is to find a hyperplane that best divides a dataset into two or more classes and maximize the distance between the two. This distance, which is also called *margin*, is the one between the closet point of each class defined as *support vectors* [27]. An example can be seen in Figure 2.2.

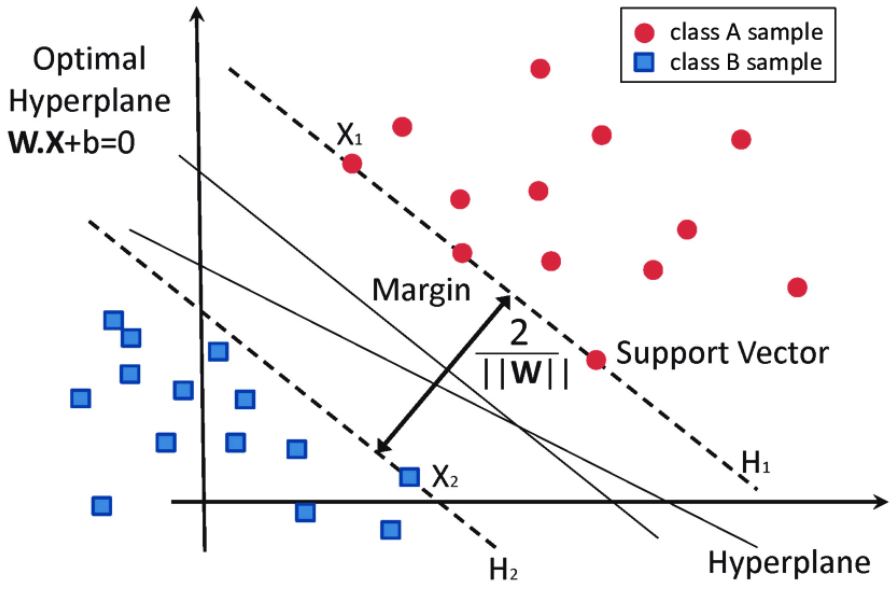


Figure 2.2: Example of a hyperplane [28].

A hyperplane is a *(N-1)*-dimensional subspace in a N-dimensional space (2.1). In a *2*-dimension space, the hyperplane will be *1*-dimension, which is simple just a line.

$$\beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + ... + \beta_n * x_n = 0 \tag{2.1}$$

In case of *non-linear dataset*, the data are randomly distributed making hard to linerly separate the classes.

A possible solution can be to increase the dimensional space for finding a hyperplane that clearly separates the classes (shown in Figure 2.3).

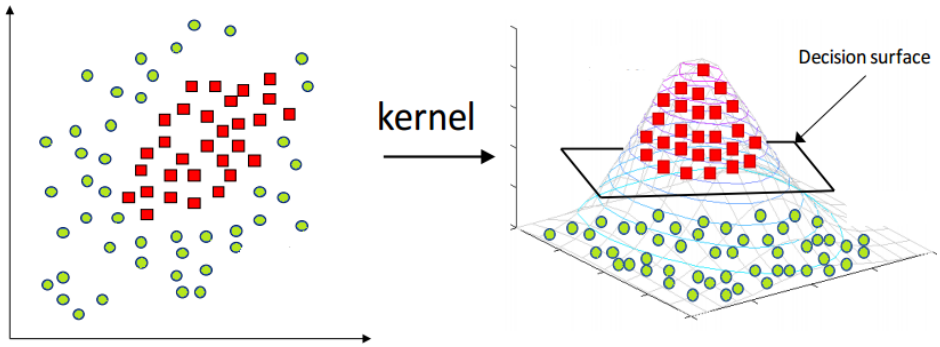


Figure 2.3: Increase the dimensional space to find an optimal decision surface [29].

However, in the case of more dimensions, the computation cost increase exponentially. In this situations it is more convenient to use a kernel. A kernel is a mathematical function that takes data as input and transform it into the required form, allowing to operate in the original feature space without increasing the dimensional space [30].
There are many kernels, the most popular ones are: *linear, polynomial, radial basis function (RBF) and sigmoid.*

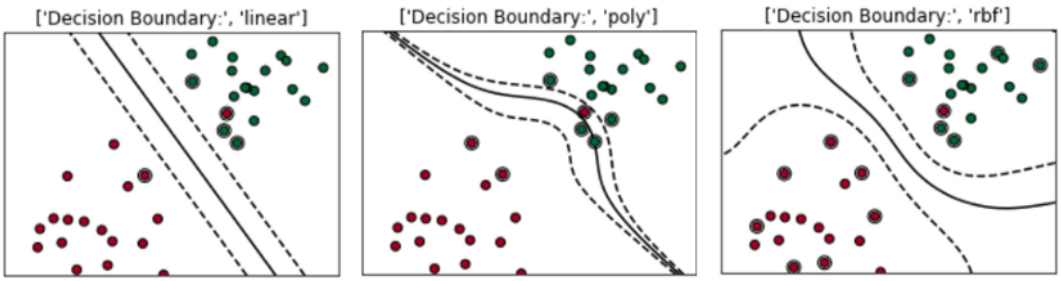


Figure 2.4: Example of different kernels [31].

While using SVM algorithms is crucial to tune the hyperparameters otherwise negative overall performances results can be achieved. The most important ones are:

- *Regularization parameter* (called C in many ML languages) sets how much to avoid miss classifying. Higher C means lower miss classification rate and smaller margin;

- *Gamma* defines how far the influence of a single training example reaches. An high gamma value will only consider points close to the hyperplane and a low value will aso take in consideration points at greater distance.

2.3 K-Means

K-means is one of the most common unsupervised machine learning algorithms used for clustering problems [32]. Clustering is used when the data are not pre-labeled, leaving the doubt to what type of groups to create. Data are grouped together trying to reach:

- *High intra-class similarity*;
- *Low inter-class similarity.*

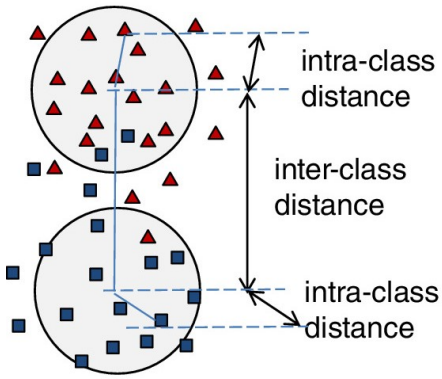


Figure 2.5: Intra-class and inter-class distance [33].

It is an iteratively algorithm assigning each point to one of the K groups. The points are assigned to clusters based on feature similarity, like the distance between them. At the end we will have:

- The centroids of the K clusters;
- Labelled data according to their cluster.

The algorithm, assuming to have input data points *x1,x2,x3,. . . ,xn* and value of K(the number of clusters needed), can be summarized as follows:

1. Select K points as the initial centroid choosing it either randomly or the first K;

2. Calculate the euclidean distance of each point in the data set with the centroids;
3. Assign each entry, according to the distance found in step *2, to the closest centroid*;
4. Compute the new centroid by averaging the points in each cluster group;
5. Repeat *2* to *4* for a fixed number of iteration or untill the centroids don't change significantly.

2.4 Decision Trees

Decision Trees (DTs) are a non-parametric supervised machine learning algorithm that learn from data to estimate a path with a set of *if-then-else* decision rules [34]. It starts by breaking down data into smaller groups while simultaneously develop the correlate tree. The final result is a tree with decision nodes and leaf nodes, where the decision node has two or more branches and the leaf node is the decision or the classification. The construction of a tree is divided in some steps:

- ***splitting***, based on particular variable, the data into subset;
- ***pruning*** is the process to reduce the tree size by switching some branch nodes into leaf nodes, and removing the leaf nodes under the original branch;
- ***tree selection*** by looking at the smallest tree that fits the data. Commonly is the one that relents the lowest cross-validated error.

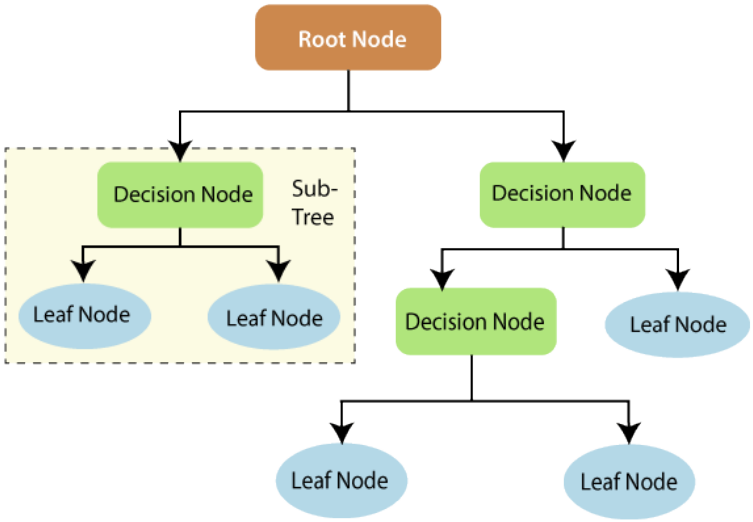


Figure 2.6: Example of a decision tree [35].

2.5 Random Forest

A Random Forest is a data construct applied to machine learning that develops large numbers of random *decision trees* analyzing sets of variables.
It merges the results from the single decision tree together to get a more accurate and stable prediction [36]. The final output will be: the most voted class if it is a label problem or the average if the trees produce a numerical value.
Random forest while computing the trees adds additional randomness to the system. While splitting a node, it does not look for the most important feature but for the best one between a random subset of features. This ends in a wide diversity that usualy results in a better outcome [37].

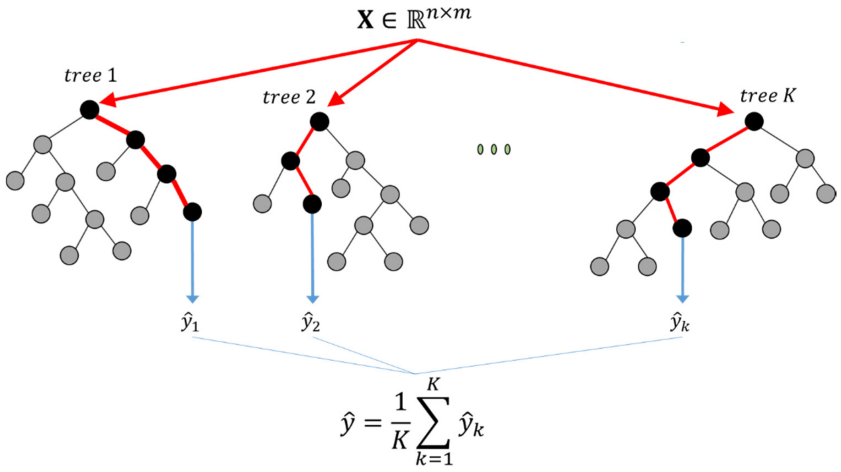


Figure 2.7: Example of a decision tree which gives numerical values as output [38].

2.6 Naive Bayes

Naive Bayes algorithm can be defined as a supervised classification algorithm, based on the *Bayes Theorem* created by *Thomas Bayes* [39].
This theorem is centered around the *conditional probability*, defined as the probability that something will happen, based on the fact that something else has already occurred. Using this probability it is possible to calculate the probability of an event knowing the history of the previous event.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.2}$$

where:

$P(A|B)$ = **posterior probability**, probability of A given value of B;
$P(B|A)$ = **likelihood** of B given A is *true*;
$P(A)$ = **prior probability**, probability of hypothesis A being true;
$P(B)$ = **marginal probability**, probability of event B.

The algorithm [40] works also based on the assumptions that each feature in the data may not be correlated or mutually independent and the likelihood P (A | B) must follow one of the statistical distributions: *Gaussian, Multinomial or Bernoulli.*

$$\begin{aligned} P(x|c) &= P(x_1|c_j) * P(x_2|c_j) * ... * P(x_d|c_j) = \\ &= \prod_{k=1}^{d} P(x_k|c_j) \end{aligned} \tag{2.3}$$

The equation (2.3) calculates the *prior probability* for the class x taking in consideration all the features c individually. Taking the highest probability we can label the data points.

2.7 Gaussian Mixture Modelling (GMM)

Gaussian mixture models is a popular unsupervised machine learning algorithm. It is similar to K-Means but it is more robust thanks to some improvements.
The first drawback of K-means is related to the use of the euclidean distance function to assign data to clusters. This methods works fine as long as the data follows a circular distribution with respect to centroids.
An other important drawback related to K-means is the use of hard clustering method, which assign each point to one and only one cluster, leaving no uncertainty measure or probability of how much a data point is related with his cluster. On the other hand, GMM captures the uncertainty of data points belonging to different clusters by using soft-assignments and it works well with non-linear dataset not having a bias for circular clusters [41].
In each cluster, for 1-dimension problem, the probability will have a Gaussian distribution, where μ is the the Gaussian's mean and σ^2 is the variance. For a d-dimensional gaussian the distribution will have a form like:

$$N(x|\mu, \Sigma) = \frac{1}{(2\pi)^{d/2}\sqrt{|\Sigma|}} \exp(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)) \tag{2.4}$$

- x refers to the random observation over which this distribution is placed;
- the mean μ, controls the Gaussian's "center position" and the standard deviation σ, controls its shape;

- Σ refers to the determinant of the covariance matrix.

Also in d-dimensional case, a Gaussian is fully specified by a mean vector µ and a d-by-d covariance matrix, Σ. For example in two dimension, the Gaussian's parameters might look like this:

$$N\left[\begin{pmatrix}\mu_1\\ \mu_2\end{pmatrix}, \begin{pmatrix}\sigma_1^2 & \sigma_{12}\\ \sigma_{21} & \sigma_2^2\end{pmatrix}\right] \tag{2.5}$$

The equation is made of two components: the first one is the mean vector, containing elements μ_1 and μ_2. This element centers the distribution of the Gaussian along every dimension. The second component, called covariance matrix, specifies the spread and orientation of the distribution.
Along covariance matrix diagonal, there are the variance terms σ_1^2 and σ_2^2 representing the shape (spread) along each of the dimensions. The off-diagonal terms, σ_{12} and σ_{21}, are equal due to the simmetry of the matrix and they specify the correlation structure of the distribution.
GMM can allow different type of covariance matrix:

- **spherical**: the probability distribution has spherical symmetry which means equal variance along the diagonal and zero off-diagonal value. Having a diagonal covariance matrix allows the distribution to spread exactly the same along the dimensions;
- **diagonal**: the covariance matrix is also a diagonal one but with different variances along it, spreading differently in each dimension in a ellipses shape;
- **full**: allows non-zero off-diagonal values, ending up with a non-axis aligned ellipses.

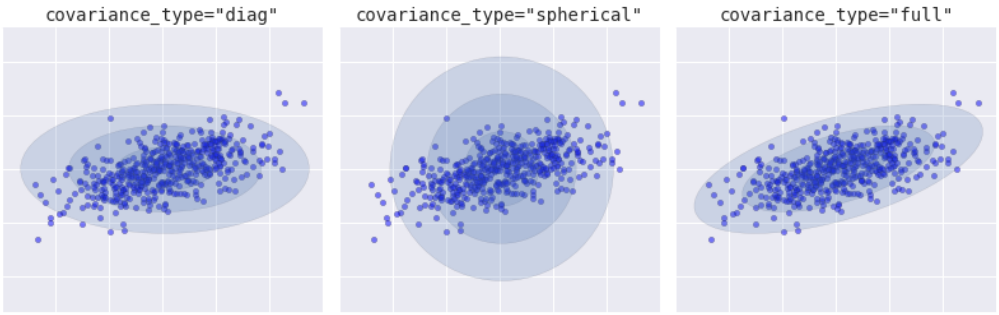


Figure 2.8: Different type of covariance.

2.7.1 GMM Naive Bayes

This algorithm combines the Naive Bayes basic principles with the statistical distribution follow the GMM instead of the Gaussian one.
The Gaussian mixture distribution helps to better describe a complex model having more Gaussian peaks and valleys.

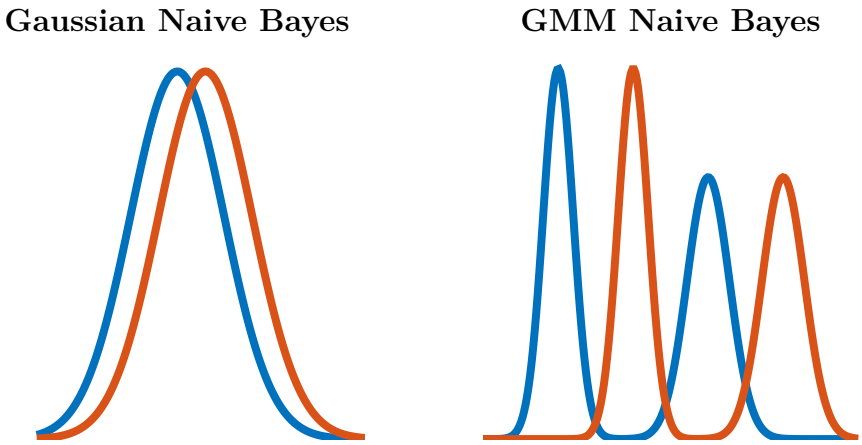


Figure 2.9: Comparison between Gaussian naive bayes and GMM naive bayes distribution for two classes.

Chapter 3

Evolution Of Hand Gesture Recognition Based On ATC

In the past few years, our research group has started to use ATC as main feature for hand gesture recognition using machine learning algorithms.
The first considerable study was done by Sapienza et al. [42], where it was demonstrated the possibility of using the ATC technique for hand gesture recognition.
In this work a system based on three differential sEMG signals is proposed, allowing the prediction of four movements and idle state:

- *wrist flexion*: the hand palm goes towards the inner arm. The muscles involved are *flexor carpi radialis, palmaris longus* and *flexor carpi ulnaris*, as well as *flexor digitorum superficialis* and *profundus*;

Figure 3.1: Wrist flexion gesture.

- *wrist extension*: the back of the hand is moved towards the distal forearm. The *extensor carpi radialis longus*, the *extensor carpi radialis brevis* and the *extensor carpi ulnaris*, together with some deep muscles, are the most used muscles while performing this action;

Figure 3.2: Wrist extension gesture.

- *hand grasp*: the finger are all closed in the direction of the hand palm. *Flexor digitorum* and *palmaris longus* are the most used, together with many intrinsic muscles of the hand;

Figure 3.3: Hand grasp gesture.

- *wrist radial deviation*: the hand is moved up along the thumb direction. *Abductor pollicis longus, flexor carpi radialis, extensor carpi radialis longus* and *brevis* are the muscles used;

Figure 3.4: Wrist radial deviation gesture.

- *idle state*: all the muscles are relaxed, keeping the hand in a steady position, without contrasting gravity.

Figure 3.5: Idle state gesture from two views.

A SVM classifier was trained using Matlab™ Statistics and Machine Learning Toolbox and after the cross-validation an average accuracy of 92.8% was reached, as fully described in Table 3.1 below.

Table 3.1: Statistical results obtained by Sapienza et al. using SVM.

	Accuracy(%)	**Sensitivity(%)**	**Specificity(%)**	**Precision(%)**
Extension	88.25	73.00	93.33	78.49
Grasp	93.25	91.00	94.00	83.49
Abduction	92.00	84.00	94.67	84.00
Flexion	98.00	95.00	99.00	96.94
Avg.	**92.87**	**85.75**	**95.25**	**85.73**

Following this direction, Mongardi et al. [43] brougth the number of recognized gesture up to five: *wrist extension, wrist flexion, wrist radial deviation, wrist ulnar deviation and hand grasp.*
Wrist ulnar deviation was added as new recognized movements, which is the action of moving the hand down, along the little finger direction. In this gesture the *extensor carpi ulnaris* and the *flexor carpi ulnaris* are the muscles involved.

Figure 3.6: Wrist ulnar deviation gesture.

An other key factor was to move all the classification process into a MCU, while using custom board for acquiring the sEMG signals and extracting the ATC values. Initially the selected MCU was Apollo 2 by Ambiq, a ultra-low power device. In this occasion a new dataset, called from now on 3D Dataset, was acquired using the configuration described above.

The sEMG signal was acquired by placing 3 couples of 24 mm electrode on the forearm (Figure 3.7), arranged in the following locations:

- the first pair was placed on the lower area of the *palmaris longus*, near to the *flexor carpi ulnaris*;
- the second couple was placed, near the wrist, on the superficial area of the *abductor pollicis longus*;
- the last pair was placed instead on the *extensor carpi ulnaris*.

(a) Forearm back (b) Forearm front

Figure 3.7: Electrode placement for the 3D Dataset acquisition.

The reference electrode was placed, looking for a neutral electrical area, on the back of the hand. This placement also did not affect the ability to perform the assigned movements.
All the process followed a very detailed protocol for both the training and testing process. In the following pages these steps are explained in detail and the classification results are presented.

3.1 Acquisition protocol for 3D Dataset

After an initial test phase, to evaluate the best electrode placement and list of gestures, an in vivo experimentation has been launched. This signal acquisition campaign aimed to acquire enough data to validate the algorithm and the system. The 3D dataset is made of data from 25 people: 16 males and 9 females, in the age range 23 to 37 years old. After being informed of the protocol and the possible risks, they all signed the informed consent for the study, following the guidelines and the regulations of the local bio-ethical committee.
Volunteers have been split into two groups: 20 people used their signals to train the classifier, while the remaining 5 have been enrolled in the online testing phase. The sessions took place on various days without trying to replicate in any case the environment conditions, to prevent a correlation between the training data and the test set. The two groups followed a slightly different procedure, but the initial calibration phase was the same for everyone.
After making the subject feel comfortable, with the arm on a table and free to move, the session began. After a brief introduction to the study, the electrodes were placed on the subject's forearm.
This placement is really critical and a bad positioning could lead to low quality of the signal, bringing the accuracy down up to 30 points. Trying to avoid this problem, a calibration protocol is follow to find the best electrode placement specific for each subject.

- starting from the rest position, one movement at the time is executed for 6.5 s. In this period, 50 values of ATC are acquired having a 130 ms time window;
- the hand returns in the rest position while relaxing for 5s to avoid muscular fatigue;
- acquiring for 6.5 s a new movement. The routine is reiterate until all the five active movements are performed;
- obtained data are then saved to a file and loaded to Matlab™ environment, where they are visually observed;
- if problems in the recorded signal are found, some electrodes could be slightly moved to enhance classifier performances.

3.1.1 Training protocol

In the 3D Dataset, subjects were asked to performed sequentially five active movements. In order to have clean signals and trying not to rush things, data were acquired in a window long twice the one used for the calibration part.
The ATC values were acquired with the Apollo2 MCU, using the debug mode of the board to control, when necessary, the flow and to reset and stop the system when needed. Trying to replicate a not optimal condition of the forearm, no skin treatment was performed before the electrode placement, to ensure the robustness of the system. In the following lines the protocol used during the training session:

1. few seconds before each movement, the supervisors reminds which one to perform;
2. a start command begins the recording session and the movement performing;
3. performing the movement for 13 s and at the end return to rest position;
4. at the end of 13 s, 100 windows should have been acquired and a stop command puts on hold the recording;
5. a rest of 5 s is observed. If there are still movements to be execute, the flow goes back to point 1;
6. at the end of the execution of all the movements, a rest session of 1 min is observed. During this time, the person can lay the arm on the table;
7. data are stored on the computer. Session restarts from 1, unless five session have already been done.

During the various sessions, the volunteers had to perform the gesture at their best capabilities since the classifier was not already trained. At the end of the procotol, the electrodes were removed from the forearm.

3.1.2 Testing protocol

After the training session, data were analyzed and the classifier was trained according to them. When the classifier was ready, the remaining five volunteers were called in to test the algorithm. During the testing protocol, the acquisitions last for 5.2 s (40 windows of 130 ms).
The acquisition time was selected of 5.2 s for keeping the execution low and to avoid muscle fatigue. Like in the training phase, data flow was controlled with the debug mode. Classifier results were not displayed to the volunteers in order to not conditioning their gesture in case of misclassification. All six movements, including Idle, were performed for complete validation of the classifier.

During the testing phase the following steps were followed:

1. few seconds before each movement, the supervisors reminds which one to perform;

2. after a start command is given and the geasture is reached, trough debug mode the firmware execution continues;

3. recording session lasts for 5.2 s;

4. at the of the 5.2 s, 40 windows are acquired and the execution is put on hold;

5. a rest of 5 s is observed. If there are still movements for the current session, the flow goes back to point 1;

6. at the end of the execution of all the movements follows a rest session of 1 min. During this time, the person can lay the arm on the table;

7. data are stored on the computer. Session restarts from 1, unless five session have already been done.

3.2 ML Analysis on the 3D Dataset

With the acquired dataset, machine learning analysis has been performed. Mongardi et al. [43] focused his attention on NN model, finding the best architecture for the model with 2 hidden layers with 26 neurons each, reaching an accuracy of more than 92%. In Table 3.2 the main characteristics of the selected model and the required training time are listed.

Table 3.2: NN perfomances on 3D Dataset by Mongardi et al.

Layers	**Nodes**	**Regularize(λ)**	**Val. Error**	**Acc(%)**	**Training Time(s)**
2	26	0.010	0.630	92.31	2110

The following step was the model implementation on the MCU to obtain also measurements related to system latency and power consumption.
In Table 3.3 the performance obtained with the MCU are listed, in particular the results were specified for each gesture and the obtained results for the accuracy were all over the 94%.

Table 3.3: Statistical results obtained by Morgardi with NN during online classification.

	Accuracy(%)	**Precision(%)**	**Recall(%)**	**F1-Score(%)**
EX	96.87	84.64	99.20	91.34
FL	96.47	90.29	88.30	89.28
RD	97.18	91.76	91.30	91.53
UD	94.97	93.63	74.90	83.22
GR	94.92	88.06	88.40	84.06
ID	97.63	87.57	100.00	93.37
Avg.	**96.34**	**89.32**	**89.02**	**88.80**

Using the same dataset, Barresi [44], using Ambiq Apollo 3 as MCU, tested two new machine learning algorithms: SVM and K-Means.
In the offline training phase, the performances reached for the SVM and K-Means were 95.3% and 83.35% respectively. With these good offline results, the two new algorithms were deployed on the MCU obtainig the results presented in Table 3.4 and 3.5.

Table 3.4: Statistical results obtained by Barresi with SVM for the online prediction.

	Accuracy(%)	**Precision(%)**	**Recall(%)**	**F1-Score(%)**
EX	96.79	78.84	79.34	79.13
FL	97.50	78.92	88.16	83.24
RD	98.86	99.38	86.09	92.26
UD	96.83	66.66	68.97	67.80
GR	98.08	80.00	81.36	80.67
ID	99.83	99.94	99.82	99.87
Avg.	**97.98**	**83.95**	**83.05**	**83.07**

Table 3.5: Statistical results obtained by Barresi with K-Means during online classification.

	Accuracy(%)	**Precision(%)**	**Recall(%)**	**F1-Score(%)**
EX	95.00	97.06	35.87	79.13
FL	93.71	52.65	100.00	83.24
RD	99.62	98.90	96.25	92.26
UD	92.79	40.00	98.28	67.80
GR	93.46	62.14	35.16	80.67
ID	96.25	99.74	94.72	97.16
Avg.	**95.14**	**75.08**	**76.71**	**69.64**

An other major focus was making sure to have a total system latency (ATC window length and computational time) below 300 ms and having a low power consumption for increasing battery life.
In Table 3.6 these two important aspects are analyzed, in particular it can be seen that it was necessary to bring the MCU clock up to 48 MHz to have a total system latency below 300 ms, which also increased the power consumption of 0.5 mW.

Table 3.6: Comparison between all the tested ML algorithms.

Work	**Algorithm**	**N. Gesture**	**Embedded**	**Global Accuracy**	**Computational Time**	**Average Power Consumption**
[42]	SVM	4		92.87%	60 ms	n.d.
[43]	NN	5	✓	92.3 %	5.15 ms	0.80 mW
[44]	K-Means	5	✓	83.3 %	124 µs	0.61 mW
	SVM (24 MHz)	5	✓	95.3 %	283.75 ms	1.28 mW
	SVM (48 MHz)	5	✓	95.3 %	141.90 ms	1.78 mW

In the wake of following this good results, it was decided to test new ML algorithms on the same 3D Dataset in order to offer a more complete overview in terms of accuracy, system latency and power consumption.

Chapter 4

Further Investigation on 3D Dataset

With the desire to further analyze the 3D Dataset and offer a complete overview of the possible ML algorithms to use, new tests were carried out.
The process started by testing offline new algorithms and then, after a carefull evaluation, deploying the best one on a MCU.

4.1 Offline training

Old and new machine learning algorithms have been tested continuing using the 3D Dataset. This time all the models were tested using ML libraries developed for Python® environment.
During the training session, to offer a complete overview and comparison were followed the same steps and settings:

- 100 initializations of the algorithm;
- k-fold validation with k=5. The fold are made by preserving the percentage of samples for each class using a stratified k-fold method.

During the training process, some parameters are learned (like node weights), some others need to be set to control the learning process. To determine the best hyper-parameter a grid search optimization was used.

4.1.1 Neural Network

The neural network model was created using a combination of *keras* and *tensorflow* libraries. In order to find the right model, many different combinations have been tried as reported in Table 4.1. The selected model was the second best in terms

of test accuracy (less than 0.5% compared with the first one) but it was much simpler, keeping in mind the constrains in terms of memory usage for the MCU. A full description of the model is given in the following lines and the complete results can be found in Table 4.2.

– **input layer**: 3 neurons as the number of features;

– **1st hidden layer**: dense layer with 26 neurons and *Rectified Linear Unit (ReLU)* as activation function. This linear function sends the result straight to the output if positive, otherwise, it will output zero;

– **2nd hidden layer**: dense layer with 26 neurons and *relu* as activation function;

– **output layer**: 6 neurons as the number of gestures to be recognized and *softmax* as activation function. Softmax creates a vector of probabilities that sum to one, representing the probability distributions of the potential outcomes.

Table 4.1: Iterations for hyparameter optimization of the NN for the 3D Dataset

Architecture	**Dropout before output layer**	**Train (%)**	**Test (%)**
36, 28, 18, 6	No	86.99	82.66
26, 18, 18, 6	No	86.38	83.66
26, 18, 12, 6	No	85.50	83.18
26, 18, 14, 6	No	85.65	82.95
22, 18, 14, 6	No	85.10	83.24
22, 18, 6	No	86.63	83.66
26, 18, 6	Yes (0.2)	86.98	83.75
26, 26, 6	Yes (0.2)	87.48	83.49
34, 26, 6	Yes (0.2)	87.56	83.69
34, 26, 6	No	87.86	83.63
26, 26, 6	**No**	**87.85**	**83.97**
32, 32, 6	No	87.89	84.32
36, 36, 6	No	88.06	84.40
36, 42, 6	No	87.86	83.55
26, 35, 6	No	87.90	83.98

Note:
[1] In the architecture column, the values are the number of neurons for each layer.
[2] The Dropout layer, each step during training time, randomly sets input units to 0 with the selected rate, trying to prevent overfitting. Non-zero inputs are scaled up by 1/(1 - rate) such that the sum over stays the same.

Adam optimizer was used during the training phase with a learning rate of 0.001. The scope is to update network weights during the epochs of the training phase. This optimizer is a stochastic gradient descent method, based on adaptive estimation of the *mean* (first moment) and the *uncentered variance* (second-order moments).

Table 4.2: NN offline training

	Accuracy(%)	**Precision(%)**	**Recall(%)**	**F1-Score(%)**
Mean	89.42	89.90	89.42	89.46
Std	0.16	0.19	0.16	0.17
Max	89.82	90.24	89.84	89.90
Min	89.08	89.42	89.08	89.12

4.1.2 Support Vector Machine (SVM)

Support vector machine was implemented using *scikit-learn* library. While using a SVM model is crucial to apply an hyperparameter optimization method (Table 4.3) to the dataset for finding the best setup.

After the optimization, the chosen hyperparameters were:

- **kernel**: radial basis function (RBF);
- **regularization parameter (C)**: '100', the regularization is inversely proportional to C. The values needs to be stricly positive, keeping in mind that the penalty during the regularization is a squared L2 penalty;
- **gamma**: 'scale' which means is equal to 1 / (n_features * variance).

Table 4.4: SVM offline training

	Accuracy(%)	**Precision(%)**	**Recall(%)**	**F1-Score(%)**
Mean	89.55	89.92	89.55	89.58
Std	0.14	0.14	0.13	0.14
Max	89.90	90.27	89.89	89.89
Min	89.17	89.64	89.17	89.24

Table 4.3: Iterations for hyparameter optimization of the SVM for the 3D Dataset

Rank	**Params**	**Test0**	**Test1**	**Test2**	**Mean**	**Std**
1	{100,'scale', 'rbf'}	0.8444	0.8664	0.8973	0.8694	0.0216
2	{1000,'auto', 'rbf'}	0.8441	0.8675	0.8960	0.8692	0.0212
3	{1000,'scale', 'rbf'}	0.8439	0.8671	0.8962	0.8691	0.0213
4	{100,'auto', 'rbf'}	0.8441	0.8659	0.8971	0.8690	0.0217
5	{10,'auto', 'rbf'}	0.8428	0.8620	0.8976	0.8674	0.0226
6	{10,'scale', 'rbf'}	0.8425	0.8614	0.8973	0.8671	0.0226
7	{1,'auto', 'rbf'}	0.8391	0.8546	0.9011	0.8650	0.0263
8	{1,'scale', 'rbf'}	0.8388	0.8543	0.9011	0.8647	0.0264
9	{100,'scale', 'sigmoid'}	0.6743	0.7147	0.6733	0.6874	0.0192
10	{100,'auto', 'sigmoid'}	0.6780	0.7071	0.6756	0.6869	0.0143
11	{1000,'scale', 'sigmoid'}	0.6715	0.7153	0.6733	0.6867	0.0202
12	{1000,'auto', 'sigmoid'}	0.6732	0.7066	0.6783	0.6860	0.0147
13	{1,'scale', 'sigmoid'}	0.6728	0.7123	0.6695	0.6849	0.0194
14	{1,'auto', 'sigmoid'}	0.6771	0.7033	0.6724	0.6843	0.0135
15	{10,'scale', 'sigmoid'}	0.6725	0.7078	0.6711	0.6838	0.0169
16	{10,'auto', 'sigmoid'}	0.6762	0.6979	0.6749	0.6830	0.0105

Note: In the params column, the first value is referred to the regularization parameter (C), the second to gamma and the last one to the type of kernel

4.1.3 Random Forest (RF)

Random forest was also implemented using *scikit-learn* library. Like the SVM algorithm, this also requires a hyperparameter optimization (Table 4.5). The selected model was the one ranked as second because, with almost the same accuracy, requires less trees. Analyzing in detail the structure and the results:

- **n_estimator**: '20', number of trees in the forest;
- **max_depth**: '20', maximum depth of the single tree;
- **min_samples_leaf**: '1', samples required to be at a leaf node;
- **min_samples_split**: '2', minimum number of samples required to split the node.

Table 4.5: Iterations for hyparameter optimization of the RF for the 3D Dataset

Rank	**Params**	**Test0**	**Test1**	**Test2**	**Mean**	**Std**
1	{200, 30, 1, 'auto', 30}	0.8902	0.8906	0.8885	0.8898	0.0008
2	{20, 2, 1, 'auto', 20}	0.8899	0.8919	0.8868	0.8895	0.0021
3	{100, 20, 5, 'sqrt', 20}	0.8897	0.8909	0.8871	0.8892	0.0015
4	{100, 15, 15, 'sqrt', 30}	0.8878	0.8909	0.8878	0.8888	0.0014
5	{100, 15, 2, 'log2', 30}	0.8883	0.8908	0.8860	0.8884	0.0019
6	{50, 5, 15, 'log2', None}	0.8878	0.8899	0.8868	0.8882	0.0012
7	{50, 30, 15, 'log2', 50}	0.8868	0.8903	0.8867	0.8879	0.0016
8	{20, 2, 10, 'sqrt', 40}	0.8895	0.8897	0.8844	0.8879	0.0024
9	{10, 5, 15, 'sqrt', None}	0.8881	0.8892	0.8863	0.8879	0.0012
10	{20, 20, 2, 'sqrt', None}	0.8880	0.8903	0.8851	0.8878	0.0021
11	{50, 5, 5, 'sqrt', None}	0.8876	0.8886	0.8843	0.8869	0.0018
12	{20, 20, 20, 'sqrt', 30}	0.8855	0.8887	0.8850	0.8864	0.0016
13	{20, 10, 20, 'sqrt', 20}	0.8844	0.8886	0.8846	0.8859	0.0019
14	{5, 20, 15, 'log2', 10}	0.8847	0.8853	0.8854	0.8851	0.0002
15	{10, 5, 20, 'auto', 10}	0.8863	0.8851	0.8834	0.8849	0.0011
16	{20, 5, 2, 'sqrt', 40}	0.8829	0.8829	0.8823	0.8827	0.0002

Note: In the params column, the values represent respectively: number of estimators, min samples split, min samples leaf, max features, max depth

Table 4.6: Random Forest offline training

	Accuracy(%)	**Precision(%)**	**Recall(%)**	**F1-Score(%)**
Mean	89.98	91.03	89.98	90.26
Std	0.12	0.54	0.12	0.21
Max	90.30	91.75	90.29	90.68
Min	89.70	90.16	89.70	89.82

4.1.4 Gaussian Mixture Modelling (GMM)

Continuing using *scikit-learn* the *GMM* algorithm was implemented. All the possible covariance settings were tested *('spherical','tied','diagonal','full')* performing each time 3 initialization for every run.

Table 4.7: GMM covariance performances comparison. All the values are in percentage

Full

	Acc	**Pre**	**Rec**	**F1-S**
Mean	19.01	55.92	19.01	17.32
Std	5.36	5.34	5.36	5.24
Max	35.04	70.70	35.04	32.30
Min	9.89	46.73	9.88	8.69

Tied

	Acc	**Pre**	**Rec**	**F1-S**
Mean	17.46	62.21	17.47	17.48
Std	5.47	8.19	5.47	6.25
Max	32.11	80.16	32.11	33.19
Min	6.42	46.12	6.43	6.54

Spherical

	Acc	**Pre**	**Rec**	**F1-S**
Mean	20.30	56.8	20.30	18.43
Std	6.34	5.98	6.35	6.05
Max	35.53	70.39	35.53	32.45
Min	8.51	44.83	8.51	6.69

Diagonal

	Acc	**Pre**	**Rec**	**F1-S**
Mean	19.32	54.72	19.32	16.69
Std	6.99	6.42	6.98	6.57
Max	37.39	71.17	37.38	33.97
Min	5.96	42.87	5.96	4.42

With any possible covariance matrix the results were way below a possible real life application.

4.1.5 Gaussian Naive Bayes

The Naive Bayes based on a Gaussian distribution was tested using *scikit-learn* library. This algorithm did not require any hyperparameter tuning since it creates the distribution starting from the labelled data. The obtained results are listed in Table 4.8.

Table 4.8: Naive Bayes offline training

	Accuracy(%)	**Precision(%)**	**Recall(%)**	**F1-Score(%)**
Mean	83.02	84.02	83.02	83.13
Std	0.22	0.19	0.22	0.21
Max	83.62	84.57	83.62	83.75
Min	82.38	83.41	82.38	82.53

4.1.6 Stacking Classifier

Two types of classifiers were combined togheter. The output of spherical GMM, which is a propability distribution, was the input for the NN or the SVM.
The NN model used in this session was:

– **input layer**: 6 neurons as the number of classes (the GMM returns the probability of belonging to each class);

- **1st hidden layer**: dense layer with 38 neurons and *ReLU* as activation function;
- **2nd hidden layer**: dense layer with 24 neurons and *relu* as activation function;
- **output layer**: 6 neurons as the number of gestures to be recognized and *softmax* as activation function.

The results in terms of accuracy, precision, recall and F1-score for the described model are presented in Table 4.9.

Table 4.9: GMM and NN offline training

	Accuracy(%)	**Precision(%)**	**Recall(%)**	**F1-Score(%)**
Mean	87.25	87.06	87.25	87.06
Std	0.22	0.29	0.22	0.26
Max	87.95	87.92	87.94	87.93
Min	86.76	86.53	86.76	86.50

An other similar configuration using stacked classifier was tested, this time giving the GMM output as input of a SVM model. The chosen plane can be described with the following paramters:

- **kernel**: 'rbf';
- **regularization parameter (C)**: '1000';
- **gamma**: 'scale'.

This second setup gave a lower performance in terms of accuracy compared to the GMM and NN as listed in Table 4.10

Table 4.10: GMM and SVM offline training

	Accuracy(%)	**Precision(%)**	**Recall(%)**	**F1-Score(%)**
Mean	84.16	83.57	84.16	83.01
Std	0.24	0.27	0.24	0.31
Max	86.00	85.66	86.01	85.49
Min	83.82	83.23	83.82	82.65

4.1.7 Offline results comparison

In Table 4.11 a comparison between all the tested ML algorithm is offered. By looking at the data several observations can be made:

- even the best results for the GMM, obtained with the spherical covariance, are way below a possible usage in real-time application;
- the overall results for the GMM & NN and GMM & SVM were acceptable but an increase in terms of accuracy did not justified a more complex system. A more complex model increases the computational time and correlate power consumption. Despite the good performances, for the reasons cited above, for both models it was decided not to go further with the implementation on MCU;
- the performaces of NN, SVM, RF and NB were at good usability level.

After a carefull evaluation of the offline training results, the attention moved on the MCU firmware design.
NN, SVM, K-Means, RF and Naive Bayes were the algorithms choosed to be deployed on the MCU. K-Means algorithm was choosen not for its performances but just for comparison, particularly in terms of computational time, being the only unsupervised algorithm tested. While writing the code, constrains in terms of memory usage and maximum latency of 300 ms had to take into account.
In the following pages the design choises, the libraries used, the system latency and power consumption are listed.

Table 4.11: Offline results comparison

	Accuracy(%)	**Precision(%)**	**Recall(%)**	**F1-Score(%)**
NN	84.16	83.57	84.16	83.01
SVM	89.55	89.92	89.55	89.58
RF	89.98	91.03	89.98	90.26
GMM (spherical)	20.30	56.8	20.30	18.43
NB	83.02	84.02	83.02	83.13
GMM & NN	87.25	87.06	87.25	87.06
GMM & SVM	84.16	83.57	84.16	83.01

4.2 Online Prediction

After the analysis on the offline training have been made, it was time to move the classification process into the MCU.
All the algorithms described in the following pages were designed specifically for the AmbiqMicro Apollo3 Blue Evaluation Board (EVB), using also some ARM custom libraries and all the online results presented were obtained using the Apollo3 Blue EVB (Figure 4.2). Due to the situation under which this work was carried out there were no possibilities to acquired real sEMG signals. For continuing testing new ML algorithms, it was decided to write a code to allow an online mode reading the 3D Dataset ATC value from a table stored on the MCU. In Figure 4.1 a possible setup for real-time applications is illustrated.

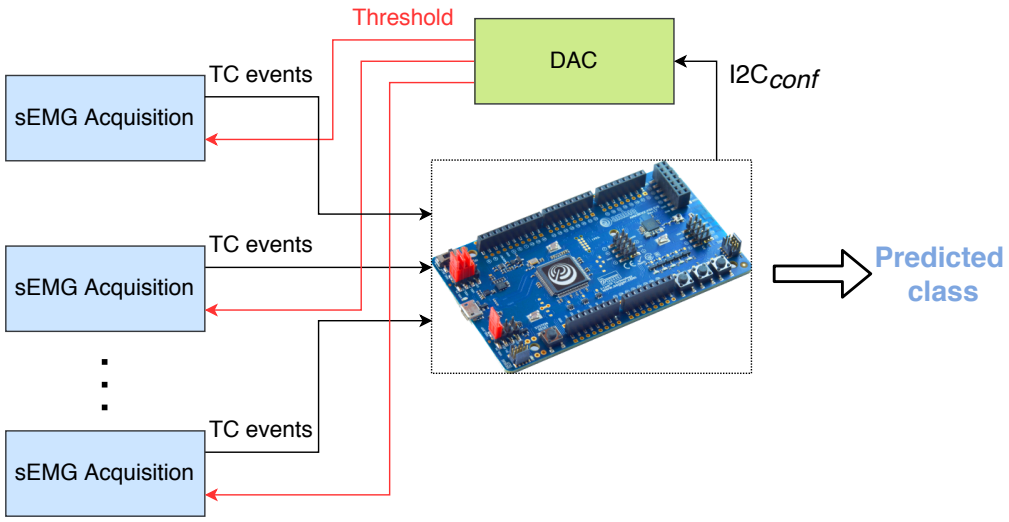


Figure 4.1: Possinle system schematic

The Apollo 3 EVB has as microprocessor (µP) the ARM Cortex-M4F DMA, a 32-bit µP designed for low power applications. In Table 4.12 the operating frequency and other main characteristics are listed. While writing the firmware for this purpose the clock frequencies was set at 48 and 24 MHz, while a low frequency crystal, designed to run at 32.768 kHz has been selected for the time window implementation needed by the ATC. The buck converters have been enabled to guarantee a very low power consumption by the µP.

Figure 4.2: The AmbiqMicro Apollo3 Blue evaluation board [45]

Table 4.12: Apollo3 Blue technical sheet

Max operating frequency	48 MHz - TurboSpot 96 MHz
MCU	32-bit ARM Cortex-M4F DMA
MCU min power	6 µA MHz^{-1}
Flash/SRAM	1 MB/384 kB
VDD	1.8 - 3.6 V
I/O	I^2C/SPI (6x) - UARTS (2x)

4.3 Firmware for Online Mode

Following AmbiqMicro guide for developers, the firmware for the MCU was written using Keil µVision IDE v5.31. The useful part about using Keil is the already pre-build software development kit (SDK) for the Arm products family, which simplifies the access to many important packages like the one used: CMSIS.
This package, and more in particular the Digital Signal Processing (DSP) library, was used for the matrices calculation. Other packages define macros to use the most common components of the board with high-level functions, simplifying the programming, especially for complex tasks.

The program is structured to have three main blocks: in the first part, all variables and constants are initialized followed by the routine implementation. The last one has the main function to execute all the commands.
In the following lines a list of the routine used during the execution is disclosed:

- system boot: functions are initialized based of the enviromental configuration. Depending of the desired operation frequency the MCU can be set to 48 Mhz or 24 Mhz and the low power mode activated;

- led initialization: using the specific library, LEDs are easily controlled with very few lines of code. During the system boot LEDs are turned on in sequence meaning that the system is ready;
- timer configuration: a timer is set to fulfill the ATC window length requirements. With the needs of having a 130 ms time window, while reducing power consumption, an external high precision clock can be used (F = 32.768 kHz). To obtain the selected 130 ms period an interrupt is raised when a value of 4260 is reached, leaving a window of 130.005 ms. While setting the timer, the REPEAT mode is activated, clearing and restarting the timer at every interrupt event. If another time measurement is needed, a second timer with a quarter of the selected hard frequency(typically 6 MHz) is initialized;
- matrices initialization: matrices are used to store intermediate value read from the ATC table that will be used as input for the classifier, and parameters of the selected ML algorithms;
- running loop: a while loop is used for class prediction, after the class evalutation has been done with the selected algorithms, the board enters in deep sleep mode untill the follow interrupt is raised.

4.3.1 Neural Network

The NN routine is based on the feed-forward propagation using the weight obtained during the offline training session. Rectified linear unit function is used to avoid divergence. No external library was used except for DSP to optimize the matrices calculation.

Table 4.13: Statistical results for NN online application

	Accuracy(%)	**Precision(%)**	**Recall(%)**	**F1-Score(%)**
EX	96.45	88.17	85.24	86.68
FL	95.60	89.09	75.38	81.67
RD	95.90	84.75	87.11	85.91
UD	94.35	74.60	68.45	71.39
GR	94.05	62.23	82.39	70.90
ID	99.95	99.88	100.00	99.94
Avg.	**96.05**	**83.12**	**83.10**	**82.75**

4.3.2 Support Vector Machine

The SVM algorithm was brought on the MCU taking advantage of the *libsvm* library [46]. It takes as input the type of kernel, the value of gamma and the file

containing the support vector obtained during the offline training.
The overall results were way above a possible usage in a real application and the best results, for an active gesture, were obtained with the wrist flexion.

Table 4.14: Performances on 3D Dataset using SVM on the MCU

	Accuracy(%)	**Precision(%)**	**Recall(%)**	**F1-Score(%)**
EX	93.86	82.58	77.63	79.68
FL	95.36	75.84	81.45	82.45
RD	94.34	82.45	83.52	81.33
UD	91.60	72.45	55.80	58.89
GR	92.67	55.84	61.44	53.71
ID	98.88	98.55	98.86	98.24
Avg.	**94.45**	**77.95**	**76.45**	**75.72**

4.3.3 Random Forest

This algorithm was deployed using *emlearn* library [47]. It allows to convert a model trained in python environment to a header file that can be used on the MCU. The implementation on the MCU was challenging due to the constraint about the memory available. To decrease the occupied space the algorithm was trained over eight subjects instead of twenty. As can be seen in Table 4.15 reducing the training set did not drastically reduced overall performance of the classifier.

Table 4.15: Statistical analysis for all the gestures of the 3D Dataset using RF

	Accuracy(%)	**Precision(%)**	**Recall(%)**	**F1-Score(%)**
EX	94.40	81.67	75.65	78.54
FL	94.50	76.79	82.69	79.63
RD	94.55	81.34	80.49	80.91
UD	92.05	64.42	50.97	56.91
GR	90.95	48.84	59.66	53.71
ID	98.95	98.27	99.13	98.69
Avg.	**94.23**	**75.22**	**74.77**	**74.73**

4.3.4 K-Means

K-Means model did not require any specific library, since it is based on the euclidean distance calculation from the centroid previously obtained in the offline training session.

In Table 4.16 all the most important statistical parameters are presented, in particular the accuracy for ulnar deviation and hand grasp was below the 90% with this type of algorithm.

Table 4.16: Online K-Means performances on 3D Dataset

	Accuracy(%)	**Precision(%)**	**Recall(%)**	**F1-Score(%)**
EX	92.40	78.47	60.52	68.33
FL	90.05	61.78	61.54	61.66
RD	93.25	72.89	84.32	78.19
UD	89.10	47.44	53.88	50.45
GR	89.60	41.58	44.89	43.17
ID	98.50	99.61	96.63	98.10
Avg.	**93.83**	**74.95**	**76.86**	**75.75**

4.3.5 Naive Bayes

This algorithm was also implemented with *emlearn*, which converts the already trained model in the python enviroment to C, allowing inference on any device with a C99 compiler.
Naive Bayes offered the low performance in terms of accuracy while performing the ulnar deviation and the best one with the wrist extension.

Table 4.17: Online Naive Bayes performances on 3D Dataset

	Accuracy(%)	**Precision(%)**	**Recall(%)**	**F1-Score(%)**
EX	94.95	83.46	78.23	80.76
FL	93.30	71.72	80.00	75.64
RD	94.25	77.92	83.62	80.67
UD	90.75	54.51	61.65	57.86
GR	93.50	62.11	67.05	64.48
ID	96.25	100.00	90.63	95.08
Avg.	**93.83**	**74.95**	**76.86**	**75.75**

4.4 System latency

In a real-time application, the system latency due to computational time is a key factor while deciding which is the best algorithm.
In this application, the latency has been measured using the Apollo3 Blue board

itself. A timer with a frequency of 6 MHz was initialized and started at the end of each ATC window and stopped after the output class is defined.
In this way, the computational time can be easily measured without interfering with any other process. With the clock frequency of the MCU set at 24 MHz, the average values obtained from the measurements were the following:

- 2.56 ms for the NN algorithm;
- 185.49 µs for the Random Forest algorithm;
- 140.46 µs for the GMM Naive Bayes;
- 61.92 µs for K-Means;
- 54.84 ms for the SVM algorithm with radial basis function as kernel.

All the tested algorithms had a total latency time (ATC window length and computation time) lower than 300 ms, possibly allowing the usage in real-time applications.

4.5 Power consumption

An important aspect to consider while choosing a machine learning algorithm is power consumption. This analysis was conducted using the DMM7510 multimeter by Tektronix.

Table 4.18: DMM7510 main specifics

DC voltage sensitivity	10 nV
Current sensitivity	1 pA
Resistance sensitivity	0.1 µΩ
Maximum Resolution	7.5 digits

Figure 4.3: DMM7510 7 1/2 digit graphical sampling multimeter

By removing the jumper placed between the board power (VDD_SP) and VDD _MCU (Figure 4.4) and adding a different header it is possible to apply the multimeter for the measure (Figure 4.5). Using the dedicated software it was possible

to control the multimeter through the computer and store the data directly on the PC. After the digitize current option has been selected, the sampling frequency was set to 650 kHz.

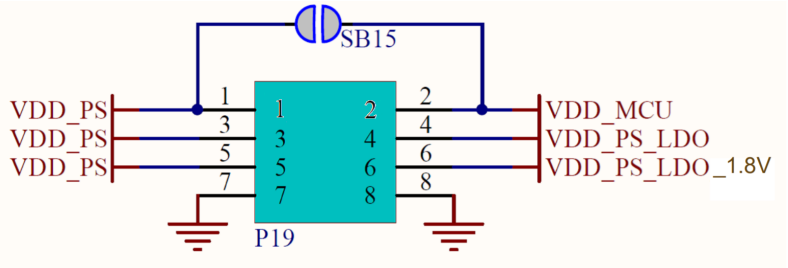


Figure 4.4: Voltage Selection on Header P19 of Apollo3 Blue

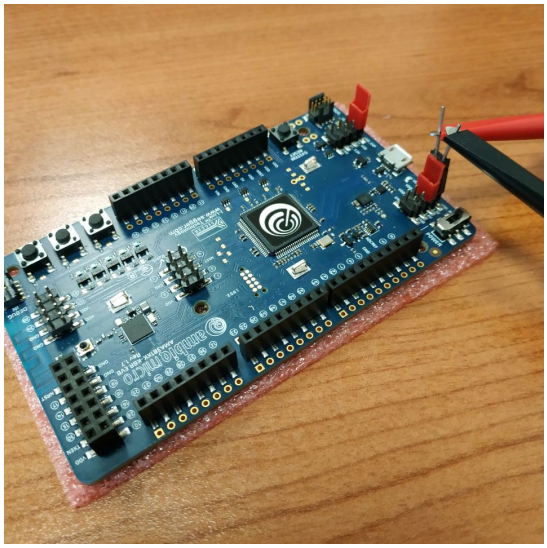


Figure 4.5: Setup used for measuring the power consumption

Figures 4.6, 4.7, 4.8, 4.9, 4.10 show the measured current for each classifier. The red rectangles highlight the prediction phase during the execution. As exepected it occurs every 130 ms and it lasts differently according to the type of algorithm.

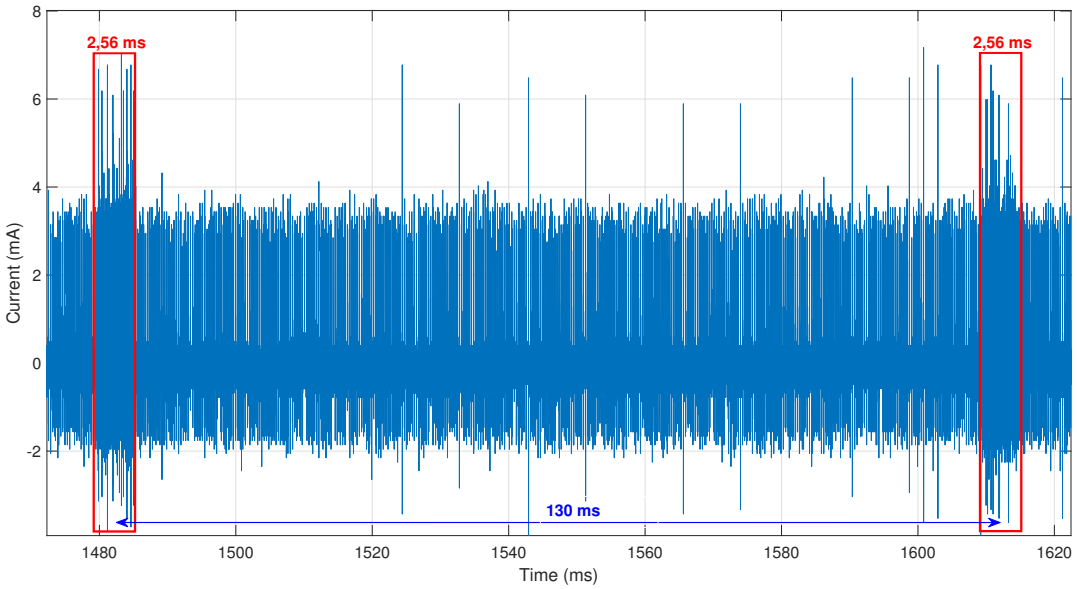


Figure 4.6: Current absorption graph for Neural Network

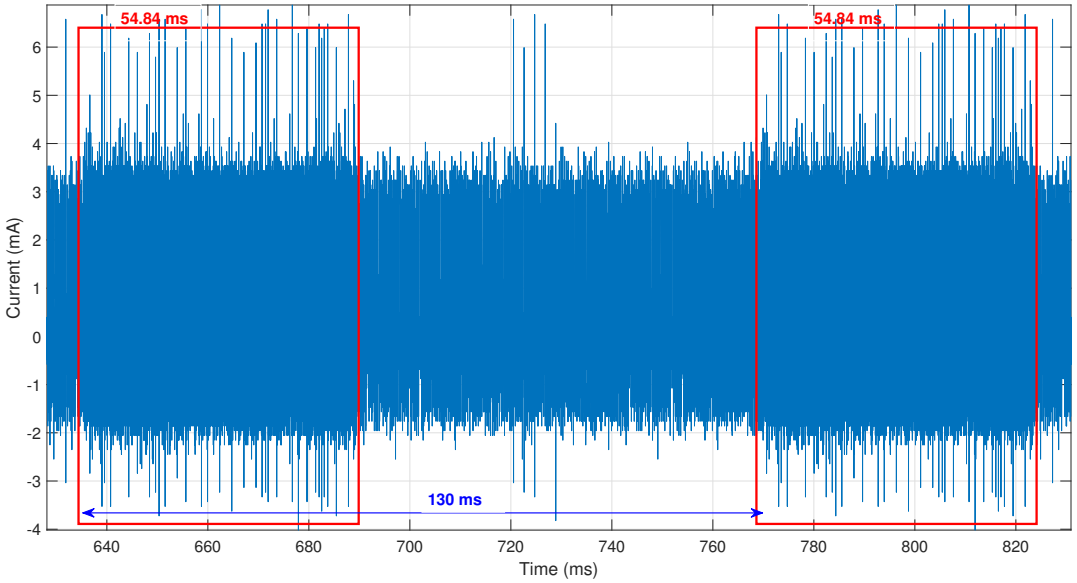


Figure 4.7: Current absorption graph for Support Vector Machine

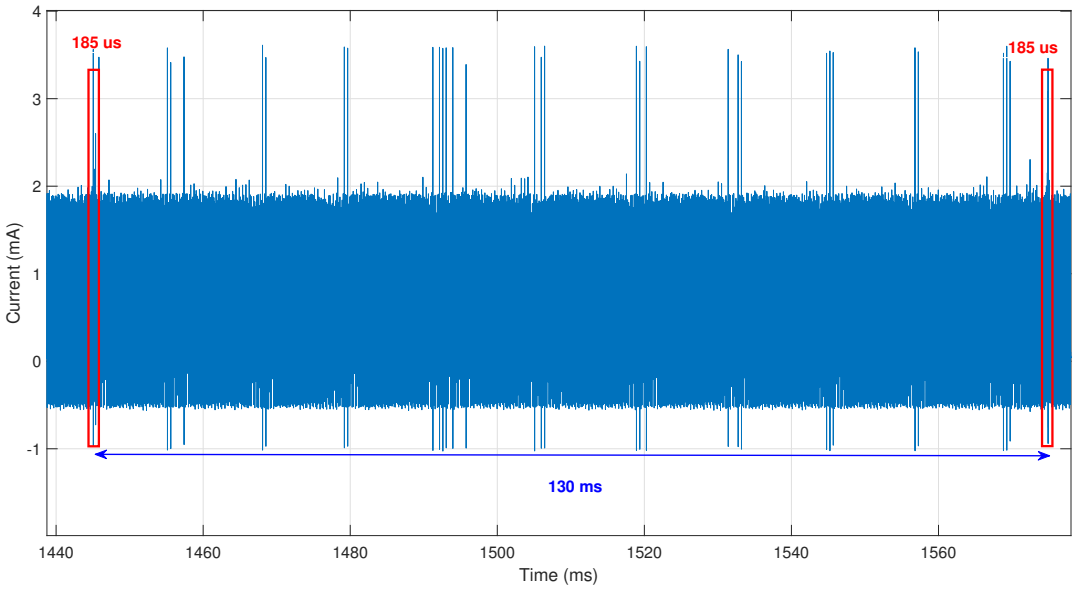


Figure 4.8: Current absorption graph for Random Forest

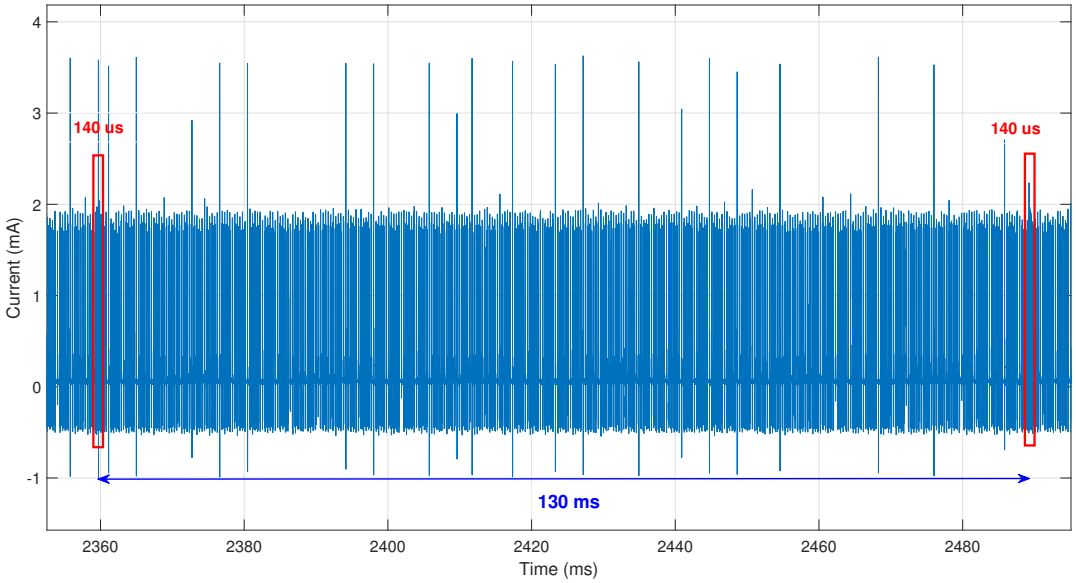


Figure 4.9: Current absorption graph for Naive Bayes

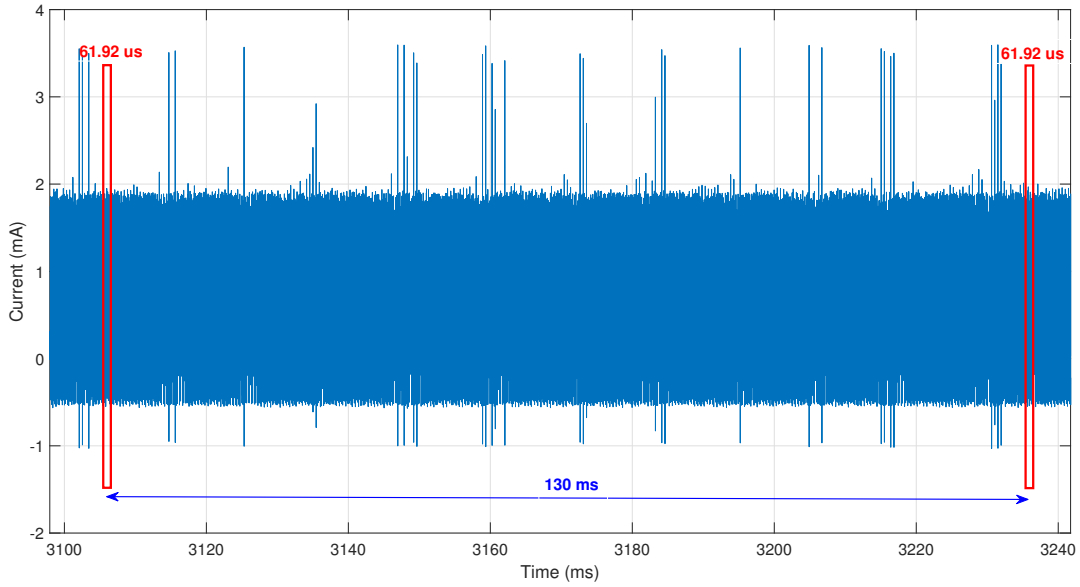


Figure 4.10: Current absorption graph for K-Means

4.6 Online Results Comparison On The 3D Dataset

In Table 4.19 results of all the classifier are listed and some observations can be made:

- K-Means offers the lowest computational time but it also has the lowest global accuracy between the five classifiers;
- NN returns the highest accuracy but with one of the highest energy consumption during computation;
- Naive Bayes and RF give good result in terms of accuracy, low time and energy for the class prediction. Between the two, with less than 2% of global accuracy, Bayes offers a better trade-off in terms also of power consumption and energy.

Table 4.19: Online Comparison

	Global Accuracy(%)	**Computational Time (µs)**	**Energy per Prediction(µJ)**	**Average Power Consumption(mW)**
NN	88.15	2560	6.20	0.5442
SVM	84.55	54840	6.90	0.9324
RF	82.70	185.49	0.79	0.5131
K-Means	76.45	61.92	0.32	0.5126
NB	81.50	140.46	0.55	0.3758

After a complete analysis on the 3D Dataset, it was time to move on trying to increase the number of recognized gestures and have an electrodes placement more suitable for an application as an armband.
Due to the situation, the study initially started analyzing datasets available online.

Chapter 5

ML With ATC On Public Dataset

To further validate machine learning algorithms based on ATC, classifications were performed on publicly available datasets of sEMG signal while performing hand gestures. ATC values were extracted by software replicating the same signal conditioning applied on our datasets, in particular hysteresis and threshold evaluation were taken into consideration for counting the TC event.
First of all the algorithm dinamically evaluates the baseline for each channel analyzing 10 s of idle state. During this evaluation, the signal is also rectified and noise mean and standard deviation are computed. Once the baseline is determined the threshold line is defined as:

$$V_{th} = baseline + mean\_noise + 3 * std\_noise \tag{5.1}$$

The signal hysteresis for triggering a TC event was set $V_{th} \pm 15\,\text{mV}$. Once all this parameters have been defined, the algorithm procedes evaluating how many times in 130 ms the sEMG signal goes first above the $V_{th} + 15\,\text{mV}$ threshold and then below the $V_{th} - 15\,\text{mV}$ one, which represents the ATC.

5.1 MeganePro

MeganePro dataset, provided by Harvard [48], is mad of data by 30 participants performing 11 gestures (10 active gesture and rest condition).
The performed movements were focused on different types of grasp: *medium wrap, lateral, parallel extension, tripod grasp, power sphere, precision disk, prismatic pinch, index finger extension, adducted thumb, prismatic four finger.*
The sEMG signal was acquired with Delsys Trigno Wireless sEMG System (Delsys Inc., USA, http://www.delsys.com/) sampled at 1926 Hz.

Gestures were classified with several algorithms:

- SVM;
- Random forest;
- Pipeline GMM and SVM;
- Pipeline GMM and NN.

For SVM and RF hyperparameters were optimized with Grid Search optimization and the results are reported below. The obtained performances did not reached acceptable values for a real application. It was probably due to the close similarity between all the gestures, all choosen between different types of grip.

Table 5.1: RF analysis on Ninapro, ranked based on accuracy.

Rank	**Params**	**Test0**	**Test1**	**Test2**	**Mean**	**Std**
1	{50, 30, 2, 'sqrt', 20}	0.5455	0.5457	0.5455	0.5456	0.00008
2	{50, 30, 1, 'auto', 20}	0.5453	0.5452	0.5451	0.5452	0.00006
3	{500, 2, 10, 'log2', 30}	0.5447	0.5449	0.5447	0.5447	0.00009
4	{50, 30, 2, 'log2', 40}	0.5450	0.5443	0.5444	0.5446	0.00029
5	{50, 2, 10, 'log2', 30}	0.5445	0.5448	0.5441	0.5445	0.00030
6	{200, 10, 15, 'auto', 40}	0.5436	0.5441	0.5436	0.5438	0.00023
7	{100, 20, 15, 'sqrt', 40}	0.5438	0.5436	0.5436	0.5437	0.00009
8	{100, 15, 15, 'auto', 20}	0.5436	0.5439	0.5434	0.5436	0.00018
9	{100, 30, 15, 'sqrt', 30}	0.5435	0.5440	0.5434	0.5436	0.00024
10	{500, 10, 20, 'sqrt', 40}	0.5427	0.5433	0.5427	0.5429	0.00026
11	{100, 2, 2, 'log2', 30}	0.5422	0.5421	0.5422	0.5421	0.00006
12	{500, 10, 1, 'sqrt', 30}	0.5425	0.5417	0.5420	0.5421	0.00034
13	{200, 20, 15, 'sqrt', 10}	0.5380	0.5380	0.5388	0.5383	0.00040
14	{200, 10, 20, 'auto', 10}	0.5364	0.5380	0.5385	0.5376	0.00091
15	{200, 2, 5, 'sqrt', 5}	0.5162	0.5173	0.5186	0.5174	0.00099
16	{50, 20, 10, 'log2', 5}	0.5147	0.5167	0.5194	0.5169	0.00193

[1] **Note**: In the params column, the values represent respectively: number of estimators, min samples split, min samples leaf, max features, max depth

Table 5.2: GMM and NN analysis on Ninapro, ranked based on accuracy.

Rank	Architecture	Test0	Test1	Test2	Mean	Std
1	{48, 36, 16}	0.1331	0.1433	0.1418	0.1394	0.0044
1	{48, 32, 18}	0.1421	0.1432	0.1328	0.1394	0.0045
1	{54, 28, 16}	0.1420	0.1432	0.1332	0.1394	0.0044
4	{46, 46, 22}	0.1422	0.1431	0.1326	0.1393	0.0047
5	{46,38,18}	0.1414	0.1430	0.1331	0.1392	0.0043
5	{46,46,24}	0.1424	0.1421	0.1332	0.1392	0.0043
7	{36,36,24}	0.1419	0.1416	0.1320	0.1386	0.0045
8	{36,36,18}	0.1418	0.1414	0.1281	0.1370	0.0065

[1] **Note**: In the architecture coloum, the values are the number of neurons for each layer

Table 5.3: SVM analysis on Ninapro, ranked based on accuracy.

Rank	Params	Test0	Test1	Test2	Mean	Std
1	{'rbf', 'scale', 1}	0.6828	0.6411	0.5964	0.6401	0.0352
2	{'rbf', 'scale', 10}	0.6738	0.6525	0.5937	0.6400	0.0338
3	{'rbf', 'auto', 1}	0.6756	0.6558	0.5742	0.6352	0.0438
4	{'rbf', 'scale', 100}	0.6531	0.6525	0.5799	0.6285	0.0343
5	{'rbf', 'scale', 1000}	0.6564	0.6456	0.5739	0.6253	0.0366
6	{'rbf', 'auto', 10}	0.6570	0.6489	0.5655	0.6238	0.0413
7	{'rbf', 'auto', 100}	0.6453	0.6369	0.5634	0.6152	0.0367
8	{'rbf', 'auto', 1000}	0.6417	0.6378	0.5646	0.6147	0.0354
9	{'sigmoid', 'scale', 10}	0.4992	0.5115	0.4734	0.4947	0.0158
10	{'sigmoid', 'scale', 1}	0.4992	0.5124	0.4722	0.4946	0.0167
11	{'sigmoid', 'scale', 100}	0.4989	0.5115	0.4644	0.4916	0.0199
11	{'sigmoid', 'scale', 1000}	0.4989	0.5115	0.4644	0.4916	0.0199
13	{'sigmoid', 'auto', 10}	0.4446	0.4803	0.4017	0.4422	0.0321
13	{'sigmoid', 'auto', 1000}	0.4446	0.4803	0.4017	0.4422	0.0321
15	{'sigmoid', 'auto', 1}	0.4440	0.4806	0.3882	0.4376	0.0379
16	{'sigmoid', 'auto', 100}	0.4446	0.4803	0.3858	0.4369	0.0389

[1] **Note**: In the params column, the first value is referred to the regularization parameter (C), the second to gamma and the last one to the type of kernel

Table 5.4: GMM and SVM analysis on Ninapro, ranked based on accuracy.

Rank	Params	Test0	Test1	Test2	Mean	Std
1	{10, 'scale', 'rbf'}	0.1418	0.1433	0.1331	0.1394	0.0044
1	{100, 'auto', 'rbf'}	0.1420	0.1433	0.1330	0.1394	0.0045
1	{1000, 'auto', 'rbf'}	0.1418	0.1433	0.1331	0.1394	0.0044
4	{1, 'auto', 'rbf'}	0.1420	0.1433	0.1326	0.1393	0.0047
5	{1, 'scale', 'rbf'}	0.1420	0.1425	0.1330	0.1392	0.0043
5	{10, 'auto', 'rbf'}	0.1420	0.1425	0.1330	0.1392	0.0043
7	{100, 'scale', 'rbf'}	0.1418	0.1418	0.1322	0.1386	0.0045
8	{1000, 'scale', 'rbf'}	0.1415	0.1418	0.1277	0.1370	0.0065

[1] **Note**: In the params column, the first value is referred to the regularization parameter (C), the second to gamma and the last one to the type of kernel

5.2 3DC Long Dataset

This is a dataset made available by Université Laval [49]. It is made of twenty-two participants performing ten active gestures plus the idle position.
The sEMG was sampled at 1000 Hz using a 10-channels custom-made armband validated making a comparison of the signal to the one acquired with Myo Armband.
Several problems were found while analyzing the data:

- unexpected spikes made of one sample (Figure 5.1), removed by substitute the value with the average of the two closest points;
- signal given not in voltage and range scale not defined.

Different settings were tried in order to reach the best results: moving the threshold line for trigger an TC event, changing the rescale range and adjusting the time window. Decent result were reached by increasing the time window up to 250ms and excluding from the classification the two pinches. Following these steps, brougth the accuracy up to 75% with a neural network and 73% with a support vector machine. The drawback of having a longer time windows is the difficulty of a real-time usage of the system.
In any circumstances the usability of these classifications were way below the standards of real hand gesture recognition device.
Below the result of different machine learning algorithms with different hyperparameter settings are presented.

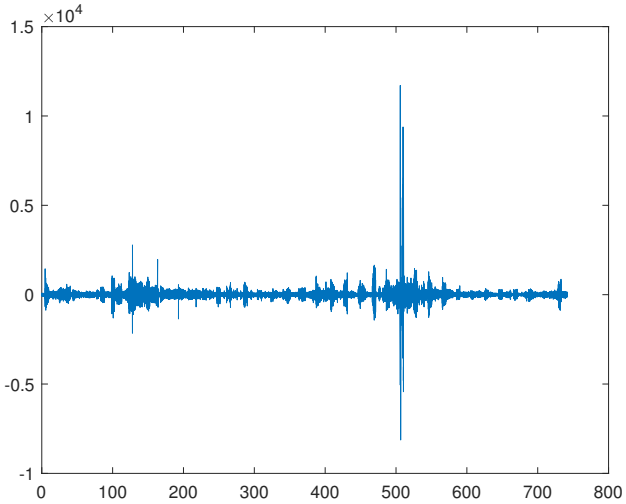


Figure 5.1: Spike problem 3DC Long Dataset.

Table 5.5: RF analysis on 3DC Long Dataset, ranked based on accuracy.

Rank	**Params**	**Test0**	**Test1**	**Test2**	**Mean**	**Std**
1	{500, 5, 2, 'sqrt', 40}	0.4083	0.4172	0.4230	0.4162	0.0060
2	{500, 15, 1, 'auto', 30}	0.4075	0.4117	0.4144	0.4112	0.0028
3	{500, 5, 10, 'log2', 30}	0.3928	0.3949	0.3972	0.3949	0.0018
4	{100, 30, 5, 'log2', 20}	0.3882	0.3908	0.3967	0.3919	0.0035
5	{100, 15, 10, 'log2', 30}	0.3854	0.3896	0.3928	0.3893	0.0030
6	{200, 15, 15, 'log2', 40}	0.3770	0.3806	0.3880	0.3819	0.0045
7	{100, 2, 15, 'log2', 30}	0.3749	0.3795	0.3838	0.3794	0.0036
8	{100, 15, 15, 'log2', 30}	0.3771	0.3801	0.3804	0.3792	0.0014
9	{100, 2, 20, 'log2', 20}	0.3635	0.3678	0.3685	0.3666	0.0021
10	{200, 30, 1, 'sqrt', 10}	0.3633	0.3590	0.3711	0.3645	0.0050
11	{500, 15, 15, 'log2', 10}	0.3545	0.3544	0.3649	0.3579	0.0049
12	{200, 20, 15, 'sqrt', 10}	0.3537	0.3546	0.3613	0.3565	0.0033
13	{50, 30, 15, 'auto', 10}	0.3493	0.3523	0.3576	0.3531	0.0034
14	{100, 2, 20, 'log2', 10}	0.3450	0.3479	0.3565	0.3498	0.0048
15	{200, 30, 2, 'sqrt', 5}	0.2569	0.2615	0.2641	0.2608	0.0029
16	{50, 2, 20, 'sqrt', 5}	0.2546	0.2578	0.2616	0.2580	0.0028

[1] **Note**: In the params column, the values represent respectively: number of estimators, min samples split, min samples leaf, max features, max depth

Table 5.6: GMM and SVM analysis on 3DC Long Dataset, ranked based on accuracy.

Rank	**Params**	**Test0**	**Test1**	**Test2**	**Mean**	**Std**
1	{1000, 'scale', 'rbf'}	0.1263	0.1394	0.1252	0.1303	0.0064
2	{100, 'scale', 'rbf'}	0.1217	0.1368	0.1225	0.1270	0.0069
3	{1, 'scale', 'rbf'}	0.1234	0.1383	0.1172	0.1263	0.0088
4	{10, 'scale', 'rbf'}	0.1219	0.1358	0.1211	0.1263	0.0067
5	{100, 'auto', 'rbf'}	0.1230	0.1365	0.1186	0.1260	0.0075
6	{1000, 'auto', 'rbf'}	0.1214	0.1312	0.1199	0.1242	0.0050
7	{1, 'auto', 'rbf'}	0.1314	0.1328	0.1055	0.1232	0.0125
8	{10, 'auto', 'rbf'}	0.1213	0.1345	0.1052	0.1203	0.0120

[1] **Note**: In the params column, the first value is referred to the regularization parameter (C), the second to gamma and the last one to the type of kernel

Table 5.7: SVM analysis on 3DC Long Dataset, ranked based on accuracy.

Rank	**Params**	**Test0**	**Test1**	**Test2**	**Mean**	**Std**
1	{10, 'scale', 'rbf'}	0.1691	0.1960	0.1500	0.1717	0.0188
2	{10, 'auto', 'rbf'}	0.1685	0.1959	0.1501	0.1715	0.0188
3	{1, 'scale', 'rbf'}	0.1581	0.1926	0.1579	0.1695	0.0163
4	{1, 'auto', 'rbf'}	0.1580	0.1930	0.1576	0.1695	0.0165
5	{100, 'scale', 'rbf'}	0.1739	0.1863	0.1469	0.1690	0.0164
6	{100, 'auto', 'rbf'}	0.1740	0.1866	0.1464	0.1690	0.0167
7	{1000, 'auto', 'rbf'}	0.1711	0.1797	0.1463	0.1657	0.0141
8	{1000, 'scale', 'rbf'}	0.1707	0.1799	0.1461	0.1656	0.0142
9	{10, 'scale', 'sigmoid'}	0.1156	0.1250	0.1272	0.1226	0.0050
10	{10, 'auto', 'sigmoid'}	0.1094	0.1163	0.1275	0.1177	0.0074
11	{1000, 'auto', 'sigmoid'}	0.1144	0.1146	0.1117	0.1136	0.0013
12	{1000, 'scale', 'sigmoid'}	0.1138	0.1138	0.1119	0.1132	0.0009
13	{100, 'scale', 'sigmoid'}	0.1139	0.1199	0.1052	0.1130	0.0060
14	{100, 'auto', 'sigmoid'}	0.1140	0.1181	0.1050	0.1124	0.0054
15	{1, 'scale', 'sigmoid'}	0.1055	0.0933	0.1052	0.1013	0.0056
16	{1, 'auto', 'sigmoid'}	0.1051	0.0954	0.1013	0.1006	0.0039

[1] **Note**: In the params column, the first value is referred to the regularization parameter (C), the second to gamma and the last one to the type of kernel

Chapter 6

Towards an Armband

The studies continued with the ultimate goal of having a user-friendly wearable device, like an armband, that can be used for gesture recognition.
With the desire to move in this direction, there is the need to gather all the electrodes from all over the forearm to a specific region, arranging them around an entire section.
The number of channels was set at seven, a good trade-off between the quantity of information and the constraints due to the dimensions of the used electrode (24 mm), especially with skinny subjects.
All the preliminary studies to reach an optimal setup were conducted using the *g.HIamp-Research* amplifier by *g.tec* (Figure 6.1) [50]. It has 144 analog input channels sampled with 24 bit of resolution and with a sampling frequency that can be set up to 38 400 Hz.
For this particular application, the sampling frequency was set to 2400 Hz with a filter that cuts out all the frequencies not in the 5-500 Hz range.

Figure 6.1: g.HIamp-Research 144-Channel Research Amplifier.

The first part of the study was dedicated to research a section of the forearm where the signal was less affected by noise and the morphological identity was maintained. The selected position of the armband was reached by placing it on the first 1/3 of the total forearm starting from the elbow, as shown in Figure 6.2.

Figure 6.2: Example of different armband placement.

(a) Different armband positioning.

(b) Selected position for the armband.

The electrode placement affects the quality of the recorded signals. In particular by looking at Figure 6.3, which is the sEMG signal referred to the positioning illustrated in Figure 6.2a, few considerations can be made.
Channel 3, which correspond to the *extensor carpi ulnaris*, in this configuration has a lot of noise due to the continuous activation of the muscle which has to support the arm weight. *Wrist radial deviation* is hardly recognized with this configuration.

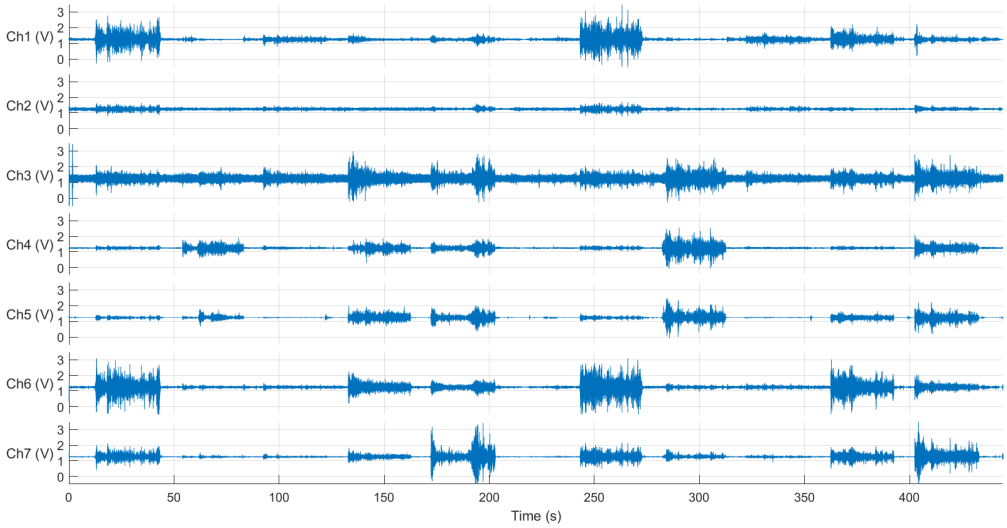


Figure 6.3: sEMG signals of the right forearm with bad electrode placement. Muscles activation investigated by each channel: **Ch1** - *Flexor carpi radialis*, **Ch2** - *Flexor carpi ulnaris*, **Ch3** - *Extensor carpi ulnaris*, **Ch4** - *Flexor digitorum profundus*, **Ch5** - *Extensor digiti minimi/Extensor digitorum*, **Ch6** - *Extensor carpi radiali brevis*, **Ch7** - *Brachioradialis*.

Figure 6.4 refers to the signals acquired with the 6.2b configuration, which allows to record a more clear signals and a better gesture recognition.

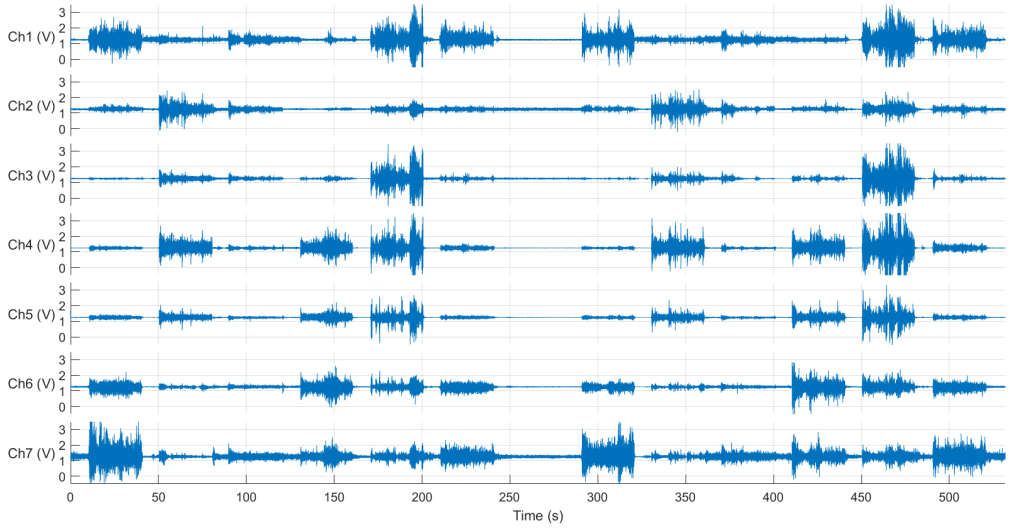
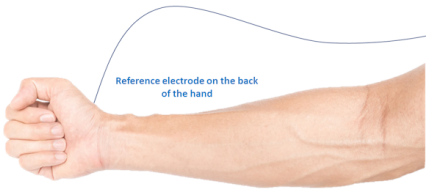


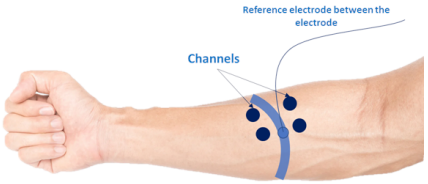
Figure 6.4: sEMG signals of the right forearm with good electrode placement. Muscles activation investigated by each channel: **Ch1** - *Flexor carpi radialis*, **Ch2** - *Flexor carpi ulnaris*, **Ch3** - *Extensor carpi ulnaris*, **Ch4** - *Flexor digitorum profundus*, **Ch5** - *Extensor digiti minimi/Extensor digitorum*, **Ch6** - *Extensor carpi radiali brevis*, **Ch7** - *Brachioradialis*.

An other challenge was trying to move the reference electrode from the back of the hand to a region closer to the active electrodes.
After some researches and experiments have been done [51], the reference electrode was moved and substituted with a fabric bracelet positioned between the first and the second row of electrodes.

Figure 6.5: Reference electrode placement.



(a) Reference electrode placed on the back of the hand.



(b) Reference electrode placed between the active electrodes.

When the armband placement was completely defined, it was time to standardize the placement of the electrodes around the entire section. The decision, according to other works [52], was to place the first electrode above the *extensor digitorum* muscle (Figure 6.6) and to equally space the other starting from the medial section.

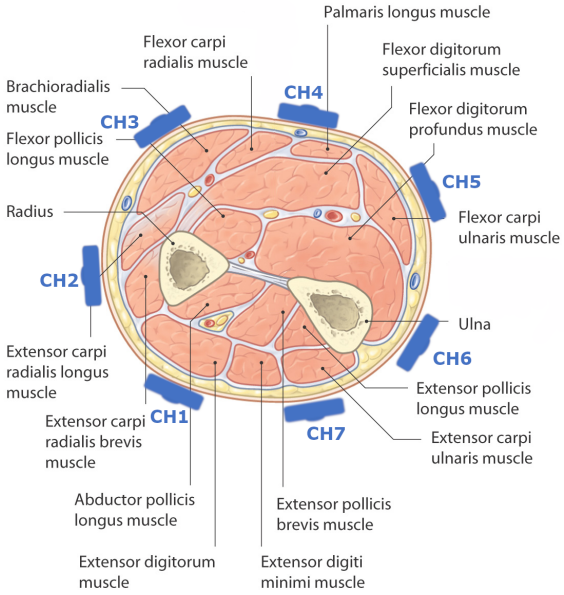


Figure 6.6: Electrode placement: section of the forearm viewed from distal to proximal.

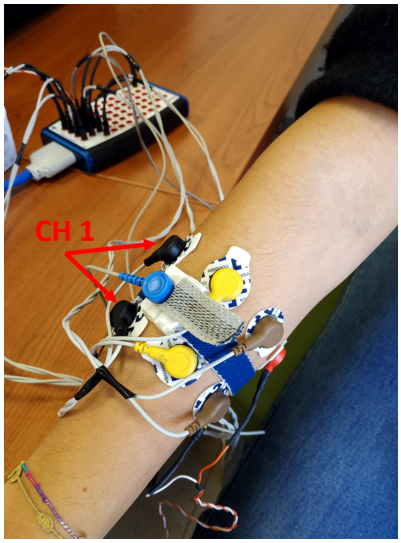


Figure 6.7: Electrode placement on the forearm of a subject.

The sEMG signals presented in Figure 6.8 were acquired with the fully defined setup, in which it was possibile to recognize the different gestures focusing on the different muscles activations.

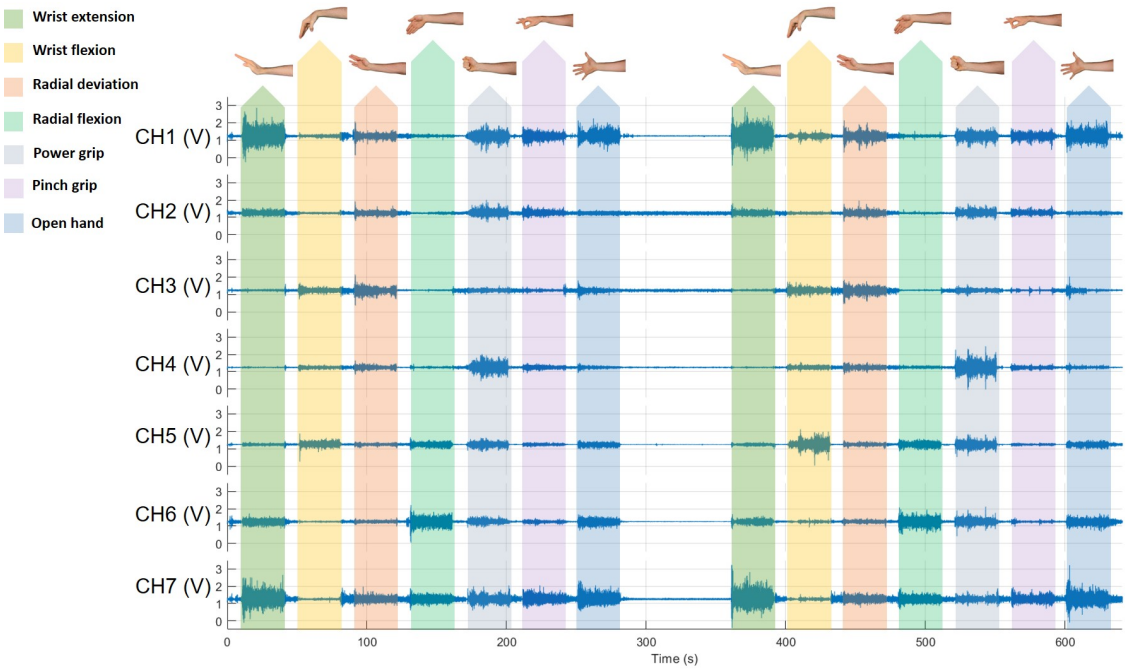


Figure 6.8: sEMG signals obtained with the described configuration.

For this acquisition, after a carefull evaluation of muscles activations, it was decided to bring the number of gestures up to seven active ones and the idle position. The two new added gestures, in addition to the ones proposed by [43], were:

- *pinch grip*: a form of precision grip where the palmar surface of the index finger touches the opposing thumb. *Flexor digitorum superficialis*, *flexor digitorum profundus*, *palmaris longus, and flexor pollicis brevis* combined with the adducting force of the *adductor pollicis* are the muscles used for this movement;

Figure 6.9: Pinch grip gesture.

- *open hand*: the hand is fully extended with the open palm. All the forearm muscles activate during this gesture, especcially the *flexor carpi radialis.*

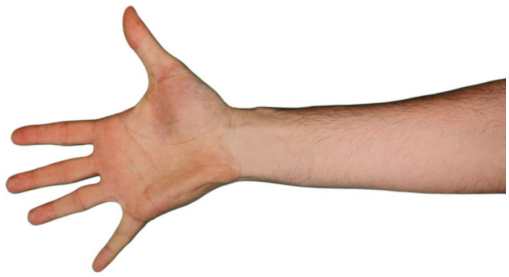


Figure 6.10: Open hand gesture.

After a first qualitative evalution of the recorded signals, a preliminary machine learning analysis has been performed to extract quantitative data to verify the quality of the sEMG signals.

6.1 Preliminary ML Analysis

A preliminary analysis was performed on a small dataset made of data from 2 people: 1 male and 1 female performing seven active gesture and the idle state. The methods and all the libraries used during the study were the same as the one described in Section 4.1.

6.1.1 Neural Network

The selected model, which gave the performances listed in Table 6.1, for this study was:

– **input layer**: 7 neurons as the number of features;

– **1st hidden layer**: dense layer with 64 neurons and ReLU activation function;

– **2nd hidden layer**: dense layer with 36 neurons and ReLU activation function;

– **3rd hidden layer**: dense layer with 22 neurons and ReLU activation function;

– **output layer**: 8 neurons as the number of gestures to be recognized and softmax as activation function.

Adam optimizer was used during the training phase with a learning rate of 0.001.

Table 6.1: Performances obtained with Neural Network during the preliminary analysis.

	Accuracy(%)	**Precision(%)**	**Recall(%)**	**F1-Score(%)**
Mean	82.78	82.83	82.38	83.16
Std	0.32	0.35	0.47	0.32
Max	83.41	83.16	83.04	83.63
Min	82.34	82.27	81.77	82.83

6.1.2 Random Forest

The results of Table 6.2 were obtained with the following hyperparameters:

- **n_estimator**: '100', number of trees in the forest;
- **max_depth**: '20', maximum depth of the single tree;
- **min_samples_leaf**: '1', samples required to be at a leaf node;
- **min_samples_split**: '2', minimum number of samples required to split the node.

Table 6.2: Performances obtained with Random Forest during the preliminary analysis.

	Accuracy(%)	**Precision(%)**	**Recall(%)**	**F1-Score(%)**
Mean	88.55	88.55	88.56	88.51
Std	0.18	0.17	0.18	0.17
Max	88.96	88.96	88.96	88.90
Min	88.22	88.26	88.22	88.18

6.1.3 SVM

The results for the SVM algorithm are presented in Table 6.3 and they were obtained with the following settings:

- **kernel**: 'RBF';
- **regularization parameter (C)**: '1000';
- **gamma**: 'scale'.

Table 6.3: Performances obtained with SVM during the preliminary analysis.

	Accuracy(%)	**Precision(%)**	**Recall(%)**	**F1-Score(%)**
Mean	83.98	83.92	83.98	83.88
Std	0.33	0.32	0.33	0.33
Max	85.01	84.92	85.01	84.92
Min	83.61	83.63	83.60	83.50

6.1.4 Naive Bayes

The best NB performances were achieved with the following parameters:

- **n_components**: '8', mixture components;
- **max_iter**: '300', number of iterations to perform;
- **n_init**: '3', number of initializations, keeping the best results.

Table 6.4: Performances obtained with Naive Bayes during the preliminary analysis.

	Accuracy(%)	**Precision(%)**	**Recall(%)**	**F1-Score(%)**
Mean	61.90	62.91	62.90	60.91
Std	0.44	0.47	0.44	0.49
Max	62.50	63.70	62.50	61.58
Min	60.90	62.40	60.90	59.85

6.1.5 Preliminary ML Analysis Summary

Table 6.5 offers a comparison between all the tested ML algorithms and gives a quantitative information about the new recording setup.
For the RF model, the performance were as good as the one obtained with the 3D Dataset but this time performed over seven active gestures rather than five.
NN and SVM performances were a little lower than the RF one, still allowing a usage in real applications. NB had the lowest accuracy, probably related to the increased number of features.

Table 6.5: Preliminary ML results comparison with the new electrodes placement.

	Accuracy(%)	**Precision(%)**	**Recall(%)**	**F1-Score(%)**
NN	82.78	82.83	82.38	83.16
SVM	83.98	83.92	83.98	83.88
RF	88.55	88.55	88.56	88.51
NB	61.90	62.91	62.90	60.91

After the results were analyzed, it was decided to further validate, over a larger dataset, the choosen electrode placement.
With the approval of the bio-ethical committee of the Università degli studi di Torino, an acquisition campaign was launched.

6.2 Protocol for the Dataset Acquisition

The acquisition campaign involved 14 subjects (11 males and 3 females) performing seven active gesture with both arms, one at a time.
After being informed about the protocol and the possible risks, each volunteer signed the informed consent for the study, following the guidelines and the regulations of the local bio-ethical committee.
After making the subject feel comfortable, the electrodes were placed on the subject's forearm starting from the electrodes to be placed above the *extensor digitorum*. A bad placement, for example too close to the elbow, could lead to low signal quality or to the loss of the gesture's morphological trend.
Trying to avoid this problem, a calibration check was followed in which the subject was asked to quickly perform all the gestures to verify muscles activations on all the channels. During the protocol the subjects were asked to sequentially perform each gesture for 30 s, with 15 s of rest between each one.
The gesture sequence was performed two times, with 30 s of rest between each session. The protocol can be summarized in the following steps:

1. the signal acquisition is started and a timer is set to mark a rhythm;
2. few seconds before each movement, the supervisors reminds which one has to be perform;
3. the timer dictates the starting point for the gesture;
4. the gesture is mantained for 30 s;
5. a rest of 5 s is observed. If there are still movements to be execute for the actual session, the flow goes back to point 2;

6. at the end of the execution of all the movements a rest session of 30 s is followed;
7. session restarts from 2, unless two session have already been done.

6.3 ML Analysis on the new dataset

At the end of the acquisition campaign, data were analyzed and ATC values were extracted over a window of 130 ms. For cleaning the data from any possible spike due to noise or involuntary movements during the recording, for each gesture only data between the 5^{th} and the 95^{th} percentile were taken into account. In the following pages, results based on this new dataset are presented.

6.3.1 Neural Network

Many architectures have been tested to find the best solution in terms of accuracy and system complexity. Different numbers of hidden layers and learning rate value have been taken into account.
The selected model had two hidden layers of 16 neurons each and combines an accuracy over 80% with a low system complexity (Table 6.6).

Table 6.6: Iterations for hyparameter optimization of the NN for the new Dataset.

Architecture[1]	**LR**[2]	**Acc(%)**	**Prec(%)**	**Rec(%)**	**F1-S(%)**
36-36-22	0.001	0.7817	0.7555	0.7817	0.7509
32-32-14	0.001	0.7961	0.7808	0.7961	0.7835
32-32	0.001	0.7918	0.7601	0.7918	0.7651
38-38	0.001	0.7846	0.7753	0.7846	0.7714
16-16	**0.001**	**0.8033**	**0.7761**	**0.8033**	**0.7823**
12-12-12	0.001	0.7721	0.7472	0.7721	0.7526
16-16-16	0.001	0.7625	0.7643	0.7625	0.7583
16-16-16	0.002	0.7841	0.7715	0.7841	0.7728
18-18	0.002	0.7583	0.7494	0.7583	0.7518
24-24	0.002	0.7865	0.7707	0.7865	0.7719
64-36-22	0.002	0.7745	0.7526	0.7745	0.7578
64-36-22	0.001	0.7769	0.7563	0.7769	0.7604
32-24-16	0.001	0.7631	0.7534	0.7631	0.7533

Note:
[1] In the architecture column, the values are the number of neurons for each hidden layer.
[2] The LR column indicates the learning rate used for the Adam optimizer

6.3.2 Random Forest

The best settings for a RF model were obtained through a random search for hyperparameter optimization. In Table 6.7 the best results are listed. The chosen model was the one ranked as fourth because it has good accuracy, only 0.3% lower compared to the first one, while maintaining low the number of trees needed.

Table 6.7: Iterations for hyparameter optimization of the new dataset, ranked based on the accuracy of the different tests.

Rank	Params	Test0	Test1	Test2	Mean	Std
1	{500, 5, 1, 'sqrt', 30}	0.7920	0.8000	0.7831	0.7917	0.0068
2	{800, 5, 2, 'auto', 40}	0.7960	0.7880	0.7831	0.7890	0.0053
2	{800, 2, 1, 'log2', 20}	0.8000	0.7840	0.7831	0.7890	0.0077
4	**{100, 2, 1, 'auto', 30}**	0.8120	**0.7800**	**0.7751**	**0.7890**	**0.0163**
5	{800, 5, 1, 'auto', 10}	0.7880	0.7880	0.7871	0.7877	0.0004
6	{500, 2, 1, 'log2', 20}	0.8000	0.7760	0.7831	0.7863	0.0100
6	{800, 5, 2, 'auto', 30}	0.7840	0.7920	0.7831	0.7863	0.0039
6	{800, 5, 1, 'sqrt', 30}	0.7920	0.7840	0.7831	0.7863	0.0039
9	{500, 10, 1, 'auto', 40}	0.7920	0.7880	0.7791	0.7863	0.0053
9	{500, 10, 1, 'sqrt', 40}	0.8000	0.7800	0.7791	0.7863	0.0096
11	{100, 5, 2, 'sqrt', 40}	0.7880	0.7720	0.7951	0.7850	0.0096
12	{800, 10, 1, 'sqrt', 10}	0.7920	0.7840	0.7791	0.7850	0.0053
13	{800, 2, 2, 'log2', 30}	0.7800	0.7840	0.7871	0.7837	0.0029
14	{800, 10, 2, 'sqrt', 40}	0.7800	0.7880	0.7831	0.7837	0.0032
14	{500, 10, 2, 'auto', 20}	0.7800	0.7880	0.7831	0.7837	0.0032
16	{100, 5, 2, 'log2', 40}	0.7880	0.7840	0.7791	0.7837	0.0036
16	{200, 2, 1, 'sqrt', 30}	0.8000	0.7720	0.7791	0.7837	0.0118
16	{200, 15, 1, 'sqrt', 20}	0.7800	0.7920	0.7791	0.7837	0.0058
19	{800, 2, 2, 'sqrt', 30}	0.7880	0.7720	0.7871	0.7823	0.0073
20	{200, 5, 1, 'log2', 20}	0.7840	0.7800	0.7831	0.7823	0.0017

Note: In the params column, the values represent respectively: number of estimators, min samples split, min samples leaf, max features, max depth

6.3.3 K-Means

K-Means algorithm was tested as comparison being an unsupervised one with no required settings. An high standard deviation is acceptable for this type of architecture since the final results is strictly related to the centroid initialization and it suffer from local minima. Even the best performance in terms of accuracy (Table 6.8) was below a possible usage for real applications.

Table 6.8: K-Means training for the new dataset.

	Accuracy(%)	**Precision(%)**	**Recall(%)**	**F1-Score(%)**
Mean	14.84	18.62	14.84	15.61
Std	7.95	11.32	7.95	8.98
Max	35.74	41.65	35.74	37.41
Min	3.42	2.30	3.42	2.74

6.3.4 SVM

The SVM algorithm required optimization, and in the end, the selected model was the one ranked as first. It combines a high accuracy (Table 6.9) and a regularization parameter equal to one, which indicates a lower computational effort.

Table 6.9: Iterations for hyparameter optimization of the SVM for the new dataset.

Rank	**Params**	**Test0**	**Test1**	**Test2**	**Mean**	**Std**
1	**{'rbf', 'scale', 1}**	**0.7920**	**0.7840**	**0.7670**	**0.7810**	**0.0103**
2	{'rbf', 'scale', 10}	0.7720	0.7800	0.7630	0.7716	0.0069
3	{'poly', 'scale', 1}	0.7600	0.7480	0.7429	0.7503	0.0071
4	{'rbf', 'scale', 100}	0.7360	0.7280	0.7389	0.7343	0.0046
5	{'poly', 'scale', 10}	0.7280	0.7040	0.7469	0.7263	0.0175
6	{'poly', 'scale', 100}	0.7200	0.7280	0.7228	0.7236	0.0033
7	{'rbf', 'scale', 1000}	0.7040	0.7120	0.7309	0.7156	0.0112
8	{'poly', 'scale', 1000}	0.7040	0.7040	0.7148	0.7076	0.0051
9	{'poly', 'auto', 1}	0.6880	0.7080	0.6987	0.6982	0.0081
10	{'rbf', 'auto', 1000}	0.6840	0.6840	0.6827	0.6835	0.0005
10	{'rbf', 'auto', 100}	0.6840	0.6840	0.6827	0.6835	0.0005
10	{'rbf', 'auto', 10}	0.6840	0.6840	0.6827	0.6835	0.0005
13	{'rbf', 'auto', 1}	0.6640	0.6800	0.6827	0.6755	0.0082
14	{'poly', 'auto', 100}	0.6560	0.6640	0.6827	0.6675	0.0112
15	{'poly', 'auto', 1000}	0.6560	0.6640	0.6787	0.6662	0.0094
16	{'sigmoid', 'scale', 1}	0.4040	0.4200	0.3453	0.3897	0.0320
17	{'sigmoid', 'auto', 1}	0.3840	0.3880	0.3855	0.3858	0.0016
17	{'sigmoid', 'auto', 100}	0.3840	0.3880	0.3855	0.3858	0.0016
19	{'sigmoid', 'scale', 100}	0.3600	0.4000	0.3253	0.3617	0.0305
20	{'sigmoid', 'scale', 10}	0.3440	0.3760	0.3293	0.3497	0.0194

Note: In the params column, the first value is referred to the regularization parameter (C), the second to gamma and the last one to the type of kernel

6.3.5 GMM Naive Bayes

For GMM Naive Bayes several iterations were made to find the number of mixtures and the type of covariance required to achieve high accuracy.
Between the models ranked as firsts, it was chosen the one with a spherical covariance for keeping the model as simple as possible in terms of complexity.

Table 6.10: Iterations for hyparameter optimization of the GMM NB for the new dataset, ranked based on the accuracy of the different tests.

Rank	Params	Test0	Test1	Test2	Mean	Std
1	{1, 'full'}	0.7120	0.6960	0.6947	0.7009	0.0078
1	**{1, 'spherical'}**	**0.7120**	**0.6960**	**0.6947**	**0.7009**	**0.0078**
1	{1, 'tied'}	0.7120	0.6960	0.6947	0.7009	0.0078
1	{1, 'diag'}	0.7120	0.6960	0.6947	0.7009	0.0078
5	{2, 'tied'}	0.6200	0.6720	0.6385	0.6435	0.0215
6	{2, 'spherical'}	0.6520	0.6720	0.5943	0.6394	0.0329
7	{3, 'tied'}	0.6240	0.6400	0.6506	0.6382	0.0109
8	{2, 'diag'}	0.6440	0.6520	0.6064	0.6341	0.0198
9	{2, 'full'}	0.6400	0.6560	0.5542	0.6167	0.0446
10	{6, 'full'}	0.6200	0.6240	0.5863	0.6101	0.0168
11	{6, 'spherical'}	0.6320	0.5760	0.6144	0.6074	0.0233
12	{5, 'tied'}	0.5480	0.6480	0.6104	0.6021	0.0412
13	{6, 'diag'}	0.5760	0.6520	0.5742	0.6007	0.0362
14	{4, 'tied'}	0.6120	0.5800	0.5943	0.5954	0.0130
15	{5, 'spherical'}	0.6000	0.5680	0.6104	0.5928	0.0180
16	{3, 'full'}	0.5440	0.6520	0.5823	0.5927	0.0447
17	{5, 'diag'}	0.5880	0.6040	0.5742	0.5887	0.0121
18	{3, 'spherical'}	0.5920	0.6040	0.5662	0.5874	0.0157
19	{3, 'diag'}	0.5680	0.5960	0.5421	0.5687	0.0219
20	{5, 'full'}	0.5800	0.5360	0.5863	0.5674	0.0223

Note: In the params column, the first value is the number of mixture components and the second setting is the type of covariance to use.

6.4 Final ML comparison

A final overview of the five ML algorithms discussed in the previous sections are compared with previous analyses performed in this thesis work as well with state of the art, as reported in Table 6.11.
Regarding the last analysis, it can be observed that K-Means unsupervised algorithm is the only one with an accuracy not suitable for real life applications.
The other four algorithms, even if with different accuracies, could all be deployed on a MCU for online usage. In fact, they have obtained a slitghly lower accuracy than state of the art works recognizing a higher number of gestures, and needing even less channels than some of them.

Table 6.11: Comparison between all the tested ML algorithms.

Work	**Features**	**# Channels**	**# Gestures**	**Algorithm**	**Embedded**	**Accuracy (%)**
[53]	Multiple	8	6	RBF		66
[54]	n.d.	64	5	HD		90-96
[20]	Features Maps	8	7	CNN		98.7
[55]	DWT	3	5	SVM		94
[42]	ATC	3	5	SVM	✓	93
[43]	ATC	3	6	NN	✓	92.3
[44]	ATC	3	6	SVM	✓	89.5
	ATC	3	6	K-Means	✓	83.3
This	RF	3	6	ATC	✓	82.7
	NB	3	6	ATC	✓	83.0
	NN	7	8	ATC		80.3
	SVM	7	8	ATC		78.1
	K-Means	7	8	ATC		35.7
	RF	7	8	ATC		78.9
	GMM NB	7	8	ATC		70.9

Chapter 7

Conclusions and Future Works

This thesis proposed an overview of machine learning algorithms for hand gesture recognition based on the Average Threshold Crossing (ATC) technique.
The goal was to deeply analyze this innovative approach and offer new solutions to move towards an armband.

The first part of the analysis, focused on the 3D Dataset, made a comparison of five algorithms: Neural Network, Gaussian Mixture Modelling, Support Vector Machine, Random Forest and Naive Bayes.
Tests were first conducted offline using ML libraries specifically designed for Python environment, aiming to recognize the five wrist movements performed: extension, flexion, radial deviation, ulnar deviation and grasp.
All the models offer the possibility of usage in real-time applications, thanks to the total system latency below the 300 ms.
Although, current measurements helped to underline Naive Bayes as a good trade-off between global accuracy, energy and power consumption.

Once the 3D Dataset was fully analyzed, the focus moved to apply the ATC technique on publicly available dataset. MeganePro and 3DC Long Dataset were investigated but due to problems, mostly related to the type and number of performed gestures and lack of information, the performances were below a possible real usage.

In the last part of the thesis, further investigations helped to determine a protocol to arrange the electrodes as they were part of an armband. With this new electrode placement it was also possible to recognize two new gestures: pinch grip and open hand, bringing the total movements up to eight.
The new setup was first validated over a 2 subjects dataset and then an acquisition

campaign was launched.

The acquired dataset, made of data from 11 males and 3 females, included sEMG signals of both arms to offer a more robust analysis. Offline ML analysis has been performed and a global accuracy of 80% was reached with two of tested algorithms, justifying a possible deployment on a MCU for real-time predictions.

A natural first step will be to deploy the already tested ML algorithms on the MCU and make a full comparison keeping in mind key factors such as power consumption and system latency.
In the case of constraints due to memory space or computational power, a possible solution could be to move to a more high-performance MCU with more flash space, for example like Ambiq Apollo 4 Blue.

An other big step forward could be the choice of a native tensorflow-lite MCU, like the SparkFun Edge board, which allows a partial fit on an already trained model.
The second fit will tailor the parameters based on the current user and his hand mobility, increasing global accuracy.

The design and development of an electrode holder will bring two major upgrades: the standardization of the electrode placement and the possible usage of dry electrodes, making the device more user-friendly and easier to place on the forearm.

Bibliography

[1] Shane W. Cummings and Christopher Tangen. *Human muscle system*. URL: https://www.britannica.com/science/human-muscle-system.

[2] URL: https://nursecepts.com/7-facts-about-the-muscular-system-every-nursing-student-should-know/.

[3] Lindsay M. Biga et al. *10.2 Skeletal Muscle*. URL: https://open.oregonstate.education/aandp/chapter/10-2-skeletal-muscle/.

[4] URL: http://www.brainkart.com/article/Structure-of-a-skeletal-muscle(Voluntary-muscle)-fibre_33243/.

[5] Openstax Content. *Muscle Tissue*. URL: https://tophat.com/marketplace/science-&-math/biology/textbooks/oer-openstax-anatomy-and-physiology-openstax-content/78/4160/.

[6] Gabriel Nasri Marzuca-Nassr et al. "Sarcomere Structure: The Importance of Desmin Protein in Muscle Atrophy". In: *International Journal of Morphology* 36.2 (2018), pp. 576–583. DOI: 10.4067/s0717-95022018000200576.

[7] Keiichi Akita and Akimoto Nimura. "Forearm Muscles". In: *Bergman's Comprehensive Encyclopedia of Human Anatomic Variation*. John Wiley and Sons, Ltd, 2016. Chap. 33, pp. 298–314. ISBN: 9781118430309. DOI: https://doi.org/10.1002/9781118430309.ch33. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781118430309.ch33.

[8] *Anterior Forearm*. June 2016. URL: https://basicmedicalkey.com/anterior-forearm/.

[9] *What is an action potential?* URL: https://www.moleculardevices.com/applications/patch-clamp-electrophysiology/what-action-potential.

[10] Carlo J. De Luca et al. "Decomposition of Surface EMG Signals". In: *Journal of Neurophysiology* 96.3 (2006), pp. 1646–1657. DOI: 10.1152/jn.00009.2006.

[11] *Adhesive Electrodes*. URL: https://shop.neurospec.com/mini-adhesive-electrodes-covidien.

[12] *Dry Electrodes*. URL: https://www.biometricsltd.com/surface-emg-sensor.htm.

[13] M. Crepaldi et al. "A quasi-digital radio system for muscle force transmission based on event-driven IR-UWB". In: *2012 IEEE Biomedical Circuits and Systems Conference (BioCAS)*. 2012, pp. 116–119. DOI: 10.1109/BioCAS.2012.6418406.

[14] S. Sapienza et al. "On Integration and Validation of a Very Low Complexity ATC UWB System for Muscle Force Transmission". In: *IEEE Transactions on Biomedical Circuits and Systems* 10.2 (2016), pp. 497–506. DOI: 10.1109/TBCAS.2015.2416918.

[15] Masoud Shahshahani et al. "An All-Digital Spike-based Ultra-Low-Power IR-UWB Dynamic Average Threshold Crossing Scheme for Muscle Force Wireless Transmission". In: Mar. 2016.

[16] Fabio Rossi et al. "Wireless Low Energy System Architecture for Event-Driven Surface Electromyography". In: May 2019, pp. 179–185. ISBN: 978-3-030-11972-0. DOI: 10.1007/978-3-030-11973-7_21.

[17] K. Momen, S. Krishnan, and T. Chau. "Real-Time Classification of Forearm Electromyographic Signals Corresponding to User-Selected Intentional Movements for Multifunction Prosthesis Control". In: *IEEE Transactions on Neural Systems and Rehabilitation Engineering* 15.4 (2007), pp. 535–542. DOI: 10.1109/TNSRE.2007.908376.

[18] Pradeep Shenoy et al. "Online Electromyographic Control of a Robotic Prosthesis". In: *IEEE transactions on bio-medical engineering* 55 (Apr. 2008), pp. 1128–35. DOI: 10.1109/TBME.2007.909536.

[19] Marie-Francoise Lucas et al. "Multi-channel surface EMG classification using support vector machines and signal-based wavelet optimization". English. In: *Biomedical Signal Processing and Control* 3.2 (2008), pp. 169–174. ISSN: 1746-8094. DOI: 10.1016/j.bspc.2007.09.002.

[20] U. Côté-Allard et al. "Transfer learning for sEMG hand gestures recognition using convolutional neural networks". In: *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2017, pp. 1663–1668. DOI: 10.1109/SMC.2017.8122854.

[21] Min Zhang, Jaemo Yang, and Qian. "Real-Time Surface EMG Pattern Recognition for Hand Gestures Based on an Artificial Neural Network". In: *Sensors* 19 (July 2019), p. 3170. DOI: 10.3390/s19143170.

[22] Jinxian Qi et al. "Surface EMG hand gesture recognition system based on PCA and GRNN". In: *Neural Computing and Applications* 32 (May 2020). DOI: 10.1007/s00521-019-04142-8.

[23] Charu Aggarwal. *Neural Networks and Deep Learning: A Textbook*. URL: https://link.springer.com/book/10.1007/978-3-319-94463-0.

[24] Stacey Ronaghan. *Deep Learning: Overview of Neurons and Activation Functions*. July 2018. URL: https://srnghn.medium.com/deep-learning-overview-of-neurons-and-activation-functions-1d98286cf1e4.

[25] Shizhao Sun et al. *On the Depth of Deep Neural Networks: A Theoretical View*. 2015. arXiv: 1506.05232 [cs.LG].

[26] John C. Platt. "Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods". In: *ADVANCES IN LARGE MARGIN CLASSIFIERS*. MIT Press, 1999, pp. 61–74.

[27] Theodoros Evgeniou and Massimiliano Pontil. "Support Vector Machines: Theory and Applications". In: vol. 2049. Jan. 2001, pp. 249–257. DOI: 10.1007/3-540-44673-7_12.

[28] Esperanza García-Gonzalo et al. "Hard-Rock Stability Analysis for Span Design in Entry-Type Excavations with Learning Classifiers". In: *Materials* 9 (June 2016), p. 531. DOI: 10.3390/ma9070531.

[29] Grace Zhang. *What is the kernel trick? Why is it important?* Nov. 2018. URL: https://medium.com/@zxr.nju/what-is-the-kernel-trick-why-is-it-important-98a98db0961d.

[30] Zhiliang Liu and Hongbing Xu. "Kernel Parameter Selection for Support Vector Machine Classification". In: *Journal of Algorithms & Computational Technology* 8.2 (2014), pp. 163–177. DOI: 10.1260/1748-3018.8.2.163.

[31] Lujing Chen. *Support Vector Machine*. Jan. 2019. URL: https://towardsdatascience.com/support-vector-machine-simply-explained-fee28eba5496.

[32] J. MacQueen. "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. Berkeley, Calif.: University of California Press, 1967, pp. 281–297. URL: https://projecteuclid.org/euclid.bsmsp/1200512992.

[33] Yuanzheng Cai et al. "A robust interclass and intraclass loss function for deep learning based tongue segmentation". In: *Concurrency and Computation: Practice and Experience* 32.22 (2020), e5849. DOI: https://doi.org/10.1002/cpe.5849. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.5849.

[34] Lior Rokach and Oded Maimon. "Decision Trees". In: *Data Mining and Knowledge Discovery Handbook*. Ed. by Oded Maimon and Lior Rokach. Boston, MA: Springer US, 2005, pp. 165–192. ISBN: 978-0-387-25465-4. DOI: 10.1007/0-387-25465-X_9. URL: https://doi.org/10.1007/0-387-25465-X_9.

[35] *Decision Trees in Machine Learning*. Oct. 2019. URL: https://www.tutorialandexample.com/decision-trees/.

[36] Leo Breiman. "Random Forests". In: *Machine Learning* 45.1 (2001), pp. 5–32. DOI: 10.1023/a:1010933404324.

[37] Misha Denil, David Matheson, and Nando De Freitas. "Narrowing the Gap: Random Forests In Theory and In Practice". In: ed. by Eric P. Xing and Tony Jebara. Vol. 32. Proceedings of Machine Learning Research 1. Bejing, China: PMLR, 22–24 Jun 2014, pp. 665–673.

[38] Chris Aldrich. "Process Variable Importance Analysis by Use of Random Forests in a Shapley Regression Framework". In: *Minerals* 10.5 (2020), p. 420. DOI: 10.3390/min10050420.

[39] Daniel Berrar. "Bayes' Theorem and Naive Bayes Classifier". In: Jan. 2018. ISBN: 9780128096338. DOI: 10.1016/B978-0-12-809633-8.20473-1.

[40] Harry Zhang. "The Optimality of Naïve Bayes". In: *In FLAIRS2004 conference*. 2004.

[41] Christopher M. Bishop. *Pattern recognition and machine learning*. Springer, 2016.

[42] S. Sapienza et al. "On-Line Event-Driven Hand Gesture Recognition Based on Surface Electromyographic Signals". In: *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2018, pp. 1–5. DOI: 10.1109/ISCAS.2018.8351065.

[43] Andrea Mongardi et al. "A Low-Power Embedded System for Real-Time sEMG based Event-Driven Gesture Recognition". In: Nov. 2019, pp. 65–68. DOI: 10.1109/ICECS46596.2019.8964944.

[44] V. Barresi. "Machine Learning Approaches for Embedded Real-Time Gesture Recognition". 2020.

[45] *Apollo3 Blue*. Nov. 2020. URL: https://ambiq.com/apollo3-blue/.

[46] Chih-Chung Chang and Chih-Jen Lin. "LIBSVM : a library for support vector machines". In: ed. by ACM Transactions on Intelligent Systems and Technology. http://www.csie.ntu.edu.tw/ cjlin/libsvm, 2011, 2:27:1–27:27.

[47] Jon Nordby. *emlearn: Machine Learning inference engine for Microcontrollers and Embedded Devices*. Mar. 2019. DOI: 10.5281/zenodo.2589394. URL: https://doi.org/10.5281/zenodo.2589394.

[48] Francesca Giordaniello et al. "Megane Pro: Myo-electricity, visual and gaze tracking data acquisitions to improve hand prosthetics". In: vol. 2017. July 2017, pp. 1148–1153. DOI: 10.1109/ICORR.2017.8009404.

[49] Ulysse Côté-Allard; Gabriel Gagnon-Turcotte; Angkoon Phinyomark; Kyrre Glette; Erik Scheme; François Laviolette; Benoit Gosselin. *Long-term 3DC Dataset*. 2019. DOI: 10.21227/f5ne-ya31. URL: https://dx.doi.org/10.21227/f5ne-ya31.

[50] *g.tec medical engineering GmbH*. URL: https://www.gtec.at/.

[51] J. Tomczyński, T. Mańkowski, and P. Kaczmarek. "Influence of sEMG electrode matrix configuration on hand gesture recognition performance". In: *2017 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*. 2017, pp. 42–47. DOI: 10.23919/SPA.2017.8166835.

[52] Zhenjin Xu et al. "Advanced Hand Gesture Prediction Robust to Electrode Shift with an Arbitrary Angle". In: *Sensors* 20 (Feb. 2020), p. 1113. DOI: 10.3390/s20041113.

[53] T. Phienthrakul. "Armband Gesture Recognition on Electromyography Signal for Virtual Control". In: *2018 10th International Conference on Knowledge and Smart Technology (KST)*. 2018, pp. 149–153. DOI: 10.1109/KST.2018.8426118.

[54] A. Moin et al. "An EMG Gesture Recognition System with Flexible High-Density Sensors and Brain-Inspired High-Dimensional Classifier". In: *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2018, pp. 1–5. DOI: 10.1109/ISCAS.2018.8351613.

[55] S. Benatti et al. "Online Learning and Classification of EMG-Based Gestures on a Parallel Ultra-Low Power Platform Using Hyperdimensional Computing". In: *IEEE Transactions on Biomedical Circuits and Systems* 13.3 (2019), pp. 516–528. DOI: 10.1109/TBCAS.2019.2914476.