University „Politehnica" Bucharest
Faculty of Electronics, Telecommunications and Information Technology

Real-Time Gesture Recognition System

Master Thesis

Study program: *Multimedia Technologies in Biometrics and Information Security Applications*

Author:
Ana-Antonia Neacșu

Thesis advisors:
Prof. Corneliu Burileanu, PhD
Prof. Jean-Christophe Pesquet, PhD

Bucharest
June 2019

Annex 2

University "Politehnica" of Bucharest
Faculty of Electronics, Telecommunications and Information Technology
Master Program "Multimedia Technologies in Biometrics and Information Security Applications"

MASTER THESIS

of student NEACȘU M. Ana-Antonia, group 421 – BIOSINF

1. Thesis title: **Real time gesture recognition system** / SISTEM DE RECUNOAȘTERE DE GESTURI ÎN TIMP REAL

2. The student's original contribution (not including the documentation part) and design specifications:

The project aims at designing and developing a real time automatic gesture recognition system, based on EMG signals (electromyographic signals). The electromyographic signal represents the electric potential generated by muscle cells during contraction. The classification will be done with the aid of some advanced artificial intelligence algorithms.

Data acquisition is done with the Myo armband, which is equipped with 8 circularly arranged EMG sensors, that are placed on the forearm. The signal is sent to the computer via Bluetooth, where it will be processed. The processing part consists of the extraction of some relevant features in both time and frequency domains. These features represent the input for a system based on deep neural networks (DNN).

The objective is to classify 7 gestures with an accuracy of over 90%. In this regard, several DNN architectures will be implemented and a series of experiments will be performed, by varying the network parameters (number of layers, number of neurons, learning rate, etc.). Additionally, the effect of learning with constraints will be explored. This task consists in implementing spectral norm constraints on the weights of each layer in an effort to assess and control the robustness of the network against adversarial inputs. The testing will be done on a public database, which contains data from over 40 subjects, acquired with Myo armband.

3. Resources used for developing this project:
Programming language: *Python – Tensorflow*; IDE: *PyCharm*

4. The project is based on knowledge mainly from the following 3 courses:
Image Processing, Data Analysis and Machine Learning, Digital Signal Processing.

5. The Intellectual Property upon the project belongs to: *UPB and CentraleSupélec.*

6. Thesis registration date: 30.11.2018

Thesis Advisors,
Prof. Dr. Ing. Jean-Christophe PESQUET
CentraleSupélec, France

Prof. Dr. Ing. Corneliu BURILEANU

Student,
Ana-Antonia NEACȘU

Master program coordinator,
Prof. Dr. Ing. Dragoș BURILEANU

Dean,
Prof. Dr. Ing. Cristian NEGRESCU

STATEMENT OF ACADEMIC HONESTY

I hereby declare that the thesis **"Real Time Gesture Recognition System"**, submitted to the Faculty of Electronics, Telecommunications and Information Technology in partial fulfillment of the requirements for the degree of **Master Engineer** in the domain **Technology and Telecommunication Systems**, study program **Multimedia Technologies in Biometrics and Information Security Applications**, is written by myself and was never before submitted to any other faculty or higher learning institution in Romania or any other country.

I declare that all information sources I used, including the ones I found on the Internet, are properly cited in the thesis as bibliographical references. Text fragments cited "as is" or translated from other languages are written between quotes and are referenced to the source. Reformulation using different words of a certain text is also properly referenced. I understand plagiarism constitutes an offence punishable by law.

I declare that all the results I present as coming from simulations and measurements I performed, together with the procedures used to obtain them, are real and indeed come from the respective simulations and measurements. I understand that data faking is an offence punishable according to the University regulations.

Bucharest, *June 2019*

Eng. Ana-Antonia NEACȘU

(Student's signature)

Contents

List of Figures

List of Tables

List of Abbreviations

[AR]	Autoregression Coefficient
[CE]	Cross Entropy
[CNN]	Convolutional Neural Network
[CWT]	Continous Wavelet Transform
[DFB]	Dual Forward Backward
[DNN]	Deep Neural Network
[EEG]	Electroencephalography
[EMD]	Empirical Mode Decomposition
[EMG]	Electromyography
[FISTA]	Fast Iterative Shrinkage-Thresholding Algorithm
[HCI]	Human Computer Interaction
[HIST]	EMG Histogram
[IEMG]	Integrated EMG
[KL]	Kullback Libler
[MAE]	Mean Absolute Error
[MAV]	Mean Absolute Value
[MSE]	Mean Squared Error
[MSLE]	Mean Squared Logarithmic Error

[OA]	Overall Accuracy
[PA]	Per-class Accuracy
[PCA]	Principal Component Analysis
[RMS]	Root Mean Square
[RNN]	Recurent Neural Network
[ReLU]	Rectified Linear Unit
[SDK]	Software Development Kit
[SGD]	Stochastic Gradient Descent
[SLR]	Sign Language Recognition
[SSC]	Slope Sign Change
[STFT]	Short Time Fourier Transform
[SVM]	Support Vector Machine
[SampEn]	Sample Entropy
[Tanh]	Hyperbolic Tangent
[WL]	Waveform Length
[ZCR]	Zero Crossing rate
[mDWT]	margianl Discrete Wavelet Transform
[sEMG]	Surface EMG

Chapter 1

Introduction

1.1 Thesis Motivation

In recent times, a significant amount of research has been focused on human-computer interaction (HCI) based on gestures, vision and voice. Hand gesture recognition provides an intelligent, natural, and convenient way of human–computer interaction (HCI). Its main applications are sign language recognition (SLR) and gesture-based control. Sign language recognition has the goal of interpreting signs automatically using a computer, in order to help deaf people communicate easier with the society. Although it is highly structured and based on an alphabet and symbols, it also serves as a good basic for the development of general gesture-based HCI [4].

Although most common technologies use cameras and image recording devices, a lot of problems appeared because of the changing light and the color or the pattern of the background. Accelerometers and electromyography (EMG) sensors provide another two potential technologies for gesture sensing. The EMG signal is a biomedical signal that measures electrical currents generated in muscles during its contraction, representing neuromuscular activities. While accelerometers read the acceleration from vibrations and the gravity, EMG signals aim to show the activity of the muscles while performing a gesture. Though, EMG signals have the advantage of capturing fine motions such as wrist and finger movements and can be used to develop a natural and easy to use human-computer interface [5].

Correlating the EMG signals with the gesture performed by the user is a challenge that captivated researchers for a an extensive period of time. If before

mid '10s most approaches for SRL were based on HMM (Hidden Markov Models) [6], [7], [8], [9] lately researchers tried to use Artificial Intelligence techniques for the classification task.

The results improved consistently in recent years. State of the art classifiers can recognize a gesture a limited number of gestures with a precision of over 95%. Despite that, almost all these high-performance classifiers are very resource-consuming, having behind very complex architectures as DNNs (Deep Neural Networks) or CNNs (Convolutional Neural Networks). Because of this complexity integrating these classifiers in real-time or embedded applications is very expensive and requires powerful resources.

Another important issue to be considered when developing stable real-life application using neural networks is the evaluation of their robustness against adversarial inputs. Adversarial inputs represent malicious input data that can fool machine learning models. In particular, NN are highly vulnerable to attacks based on small modifications of their input at the test time [10]. Recent mathematical findings [11], [12], [13] show that there is a possibility to assess and control the robustness of a NN by imposing appropriate spectral norm constraints on the weights of each of its layers.

The main challenge nowadays consists in developing a system that has high performance but can work in real time too. This thesis proposes a Deep Neural Network (DNN) approach, based on features extracted in real time. Additionally, since the system must perform well on real applications, a scheme of training a robust network is proposed, by norm constraints on the weights.

1.2 Applicability

Surface Electromyography (sEMG) is the electrical manifestation of the neuromuscular activation associated with the contracting muscle. This technology may be used by physically impaired persons to control rehabilitation and assistive devices. EMG is also used in many types of research domains, including those involved in biomechanics, motor control, neuromuscular physiology, movement disorders, postural control, and physical therapy [14].

Classification of gesture recognition also has a vast applicability in many domains ranging from medical field to military applications. For the system

proposed in this thesis, some possible applications are the following:

- **Intelligent prosthetics** – the system can be the core of an intelligent prosthesis helping people with disabilities live a normal life as the interaction between the person and the robotic prosthesis would be as organic as possible. Persons who are missing a limb can be trained via kinetotherapy to send the nervous stimulus that would have generated the movement. The EMG signals can be acquired and translated into a gesture that can be performed by the prosthetic.

- **Sign language recognition** – as the system is specialized in hand gesture recognition this application would be a direct implementation. If the system would be connected to a sign language dictionary it could translate in real time what an impaired person is communicating. Using a speech synthesis module, the gestures could be translated directly into words, offering voice to the voiceless. This will bridge the gap between disabled persons and the rest of the world, helping them integrate easier into society.

- **Military applications** – he system could help soldiers who are in dangerous missions communicate with their peers in a discrete manner. Sign codes can be transmitted via gestures without visual contact if the person wears the acquisition system. This way, all the messages can be transmitted successfully, regardless of the environment conditions.

- **Games and virtual reality** – as the system plays the role of a link between the human gestures and the machine, it could be used to enhance the gaming experience by adding a more realistic sense to the game. This can be considered the next step in virtual reality industry, that tries to offer the user an experience as real as possible.

- **Gesture based-control applications** – the hand gesture classifier could be used to communicate with devices that are far away from the user and its applicability ranges all the way from personal computers to drones.

1.3 Thesis Objectives and Outline

This thesis aims at designing and developing a real time automatic gesture recognition system, based on sEMG signals (electromyographic signals). The classification will be done with the aid of some advanced artificial intelligence algorithms.

Data acquisition is done with the Myo armband, which is equipped with 8 circularly arranged EMG sensors, that are placed on the forearm. The signal is sent to the computer via Bluetooth, where it will be processed. The processing part consists of the extraction of some relevant features in both time and frequency domains. These features represent the input for a system based on deep neural networks (DNN).

The objective is to classify 7 gestures with an accuracy of over 90%. In this regard, several DNN architectures will be implemented and a series of experiments will be performed, by varying the network parameters (number of layers, number of neurons, learning rate, etc.). Additionally, different optimization techniques for performing the training will be explored. This task consists in imposing different norm constraints during training to ensure the robustness of the network towards adversarial inputs and increase the generalization capabilities of the network. The testing will be done on a public database, which contains data from over 40 subjects, acquired with Myo armband.

The rest of the paper is organized as follows: State of the art of the domain and some theoretical background information will be presented in Chapter 2. Chapter 3 will detail the implementation of constrained training. The optimization techniques used for improving the results will be also discussed. Chapter 4 will deal with the experimental setup, including data acquisition, feature extraction and NN architecture used for the experiments. Chapter 5 deals with experiments and results and the last section, Chapter 6, is dedicated to concluding remarks.

Chapter 2

Theoretical Background

2.1 Related Work on EMG

Surface Electromyography (sEMG) is the electrical manifestation of the neuromuscular activation associated with the contracting muscle. This technology may be used by physically impaired persons to control rehabilitation and assisting devices. EMG is also used in many types of research domains, including those involved in biomechanics, motor control, neuromuscular physiology, movement disorders, postural control, and physical therapy [15].

The first recording of EMG activity was made by Marey in 1890, who introduced the term electromyography. Clinical use of surface EMG for the treatment of different disorders began in the 1960s. Hardyck was the first practitioner who used EMG [16]. Cram and Steger introduced a clinical method for scanning a variety of muscles, in 1980, using an EMG sensing device [17]. Progress in understanding these signals has been made during the past 15 years. Still, there are some limitations in characterizing the properties of surface EMG signals (estimation of the phase, acquiring exact information) due to derivation from normality. Traditional system reconstruction algorithms have various limitations and considerable computational complexity and many show high variance [15].

The work carried by researchers focused on sEMG signals had resulted in developing better algorithms, upgrading existing methodologies, improving detection techniques to reduce noise, and acquiring accurate EMG signals. This section will further present the performance of various technologies developed with sEMG signals.

In [18], an integrated approach for the identification of daily hand movements with a view to control prosthetic members is proposed. The EMG signals were acquired using two electrodes attached on two specific muscles of the hand. Features are extracted in the frequency domain, while inserting a dimensionality reduction stage which is based either on Principal Component Analysis (PCA) or RELIEF feature selection algorithm, before the application of the classifier. Results have shown that the information extracted by EMD (Empirical Mode Decomposition) provides features in the frequency domain that can further increase the classification accuracy. Based on a collected set of raw EMG-recordings, a pre-processing stage excludes the non-contracting portions at the beginning of each movement. Following the muscle contraction detection, segmentation of the rest of the signal takes place using overlapping time-windows. From each segment several features are extracted using both the signal as well as its Intrinsic Mode Functions (IMFs) computed after the application of EMD. Due to the high dimensionality of the produced feature vector, two methods for dimensionality reduction were tested before the application of a simple linear classifier which attempts to classify each segment to one of the six basic hand movements.

In [3] two datasets are recorded using Myo Armband (Thalmic Labs), a dry electrode sEMG armband. While the device also incorporates a 9-axis inertial measurement unit (IMU), it was deactivated for the recording of the dataset. These datasets split into pre-training and evaluation dataset, are comprised of 19 and 17 able-bodied participants respectively. A convolutional network (ConvNet) is augmented with transfer learning techniques to leverage inter-user data from the first dataset, alleviating the burden imposed on a single individual to generate a vast quantity of training data for sEMG-based gesture recognition. This transfer learning scheme is shown to outperform the current state-of-the-art in gesture recognition achieving an average accuracy of 96.31% for 7 hand/wrist gestures. The dataset contains two distinct sub-datasets with the first one serving as the pre-training dataset and the second as the evaluation dataset. The first one, which is comprised of 19 able-bodied subjects, should be employed to build, validate and optimize classification techniques. The second, comprised of 17 able-bodied subjects, is utilized only for the final testing and comparisons between different methods.

Another paper that focuses on EMG signals is [19]. It proposes an EMG-based

pattern recognition algorithm for classification of joint wrist angular position during flexion extension movements from EMG signals. The algorithm uses a feature extraction stage based on a joint time-frequency representation. The pattern recognition stage uses a recurrent neural network (RNN) as classifier. Also, using an auto-encoder, a deep neural network (DNN) architecture was tested. It was carried out by relying on a set of experiment with 10 subjects. Experiments included five recorded sEMG channels from forearm executing wrist flexion and extension movements, as well as the use of a commercial electro-goniometer to acquire joint angle. Results show that shallow NN had better performance than architectures with more layers based on auto-encoders.

It has been shown that neural networks and generally machine learning systems may be affected by adversial modification of their input[10] [11]. This kind of manipulation often lead to incorrect classification which will affect the performance of the system in real life applications. The Lipschitz constant of the network can be used to asses the robustness of neural network to such adversarial inputs, if accurately computed [12]. Lately, there are several methods proposed to train Lipschitz networks, which fall into two main categories. Regularization approaches include double backpropagation [20] or applying penalization on network Jacobian [21], which impose Lipschitz constant locally, but do not enforce the constraint globally on the network. Another approach consist of imposing some constraints on the architecture of the network, such as constraining the spectral norm of each layer [22] [13]. At the expense of computation complexity, these methods ensure a Lipschitzian network.

2.2 Theoretical Background

2.2.1 Deep Neural Networks – General Aspects

In recent years, the most performant systems that use AI are based on deep learning algorithms. Deep learning represents a particularization of machine learning, and it generally refers to a system based on neural networks that is characterized by a high number of parameters.

Neural Networks are not a new concept, they have been around from more than half a decade, starting in the early 1940s [23]. Only after more than 10 years, in 1958 the first attempt of classification based on machine learning is proposed

and used in the task of image recognition [24]. The neural networks have had moments of stagnation over time, due to their high computational complexity and they were not very used during the late 20th century. They started regaining popularity in the early '00s and are now considered state of the art technology in many domains including image processing, signal processing, and even speech processing, etc.

The model for NN is of biological inspiration and refers to how human nervous system perceives, processes, and transmits information: a network of cells, called neurons that are interconnected in different ways, like synapses do in the human brain.

Another analogy to humans is the learning process: naturally, us humans learn from experience and examples and we have the capability to extrapolate and apply the things we know in similar situations. Similarly, the NN goal is to give a device the power of generalization, starting from the analysis of some examples that are used to train a system.

Train examples representing the input data are usually labeled, meaning that the output for them is known. Thus, the task of the NN is to find a generally valid function that is capable to establish the correspondence between input and output. Firstly, the parameters of the function will be chosen randomly. Then, through a process called back propagation, the optimal values of the parameters will be optimally computed, by minimizing an error function and propagating the gradients backwards through the network.

The NN is organized in layers, each layer containing a variable number of neurons. The first layer is called the input layer and its purpose is to collect the input data. So, the dimension of this layer is equal to the input data vector. Each neuron represents an attribute that describes the input data. Then, all the neurons from the input layer are connected to the next layer, called hidden layer. Here, the actual processing takes place. In DNN there are several hidden layers. The number of layers and the number of neurons of each layer are called hyper-parameters and do not have a standard value. Most of the times, these hyper-parameters are determined empirically, through lots of experiments. The process of determining the optimal architecture is called hyperparameter tuning. The last layer is called the output layer and provides the data resulting from the processing. In the case of a classification task, such as the one presented in this

paper, the size of this layer is equal to the number of distinct classes. Figure 2.1 details the general architecture of a NN.

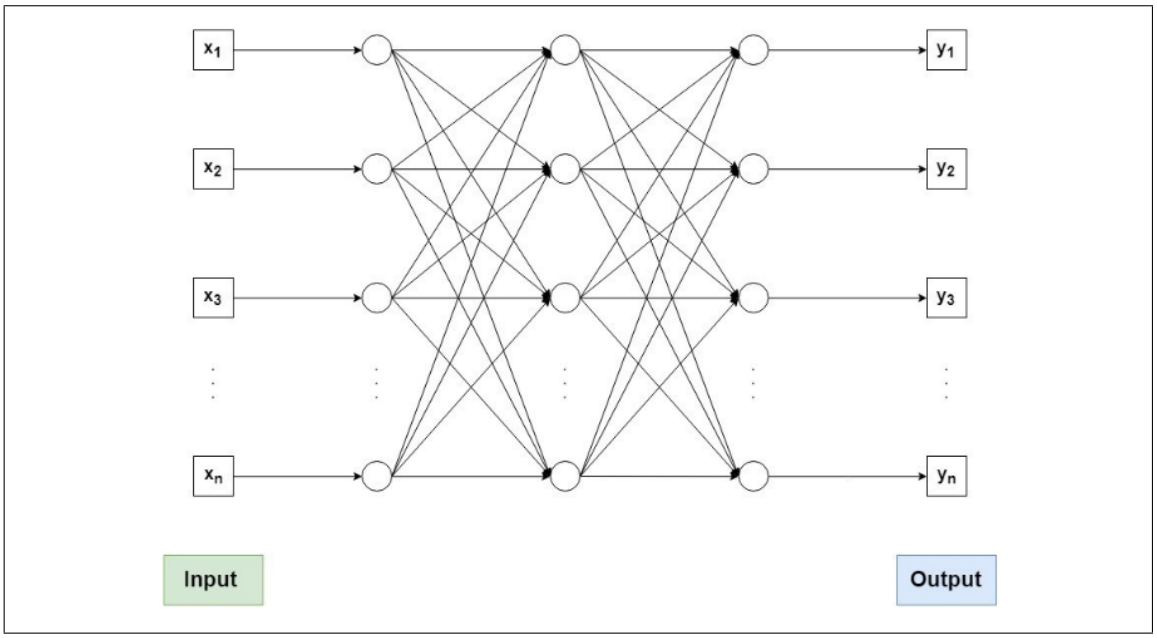


Figure 2.1: Neural Network Architecture

The output of each neuron from a layer influences with a certain weight the entry of the neurons on the next layer. The simplest way of propagating the information through the network is by using a linear function. The output of each layer will be a linear combination between the input and the weight of each neuron, as in equation 3.1. The values of z can theoretically vary up to infinity. Therefore, for the sake of numerical stability this result must be translated into the range $[0, 1]$. A linear relation is not enough to model the relation between input and output, as the correspondence function may be non-linear. Thus, all neurons are characterized by the activation function, which has the role of adding a non-linearity to the output, and possibly to keep the value of the output in the interval $[0, 1]$.

$$\xi = \sum_{i=1}^{N} x_i \cdot w_i + b, \tag{2.1}$$

where ξ is the output of the neuron, as the weighted sum of all inputs x_i, multiplied by the weights w_i, plus the bias factor b. Then, the activation function is applied

to ξ. The activation function is denoted with g.

$$y = g(\xi) \tag{2.2}$$

The activation function also decides if the value produced by the neuron further propagates and the amount of influence it will have on the next layers.

2.2.2 Activation functions

Step function

One of the earliest activation functions is the step function, which compares the input, with a threshold value. This is mostly used in binary classification tasks, because it has only two possible states: the neuron is whether active or inactive, as Equation 3.3 shows. The plot is presented in Figure 2.2.

$$g : \mathbb{R} \to \mathbb{R} : \xi \mapsto \begin{cases} 1, & \xi > th \\ 0, & \xi \leq th \end{cases} \tag{2.3}$$

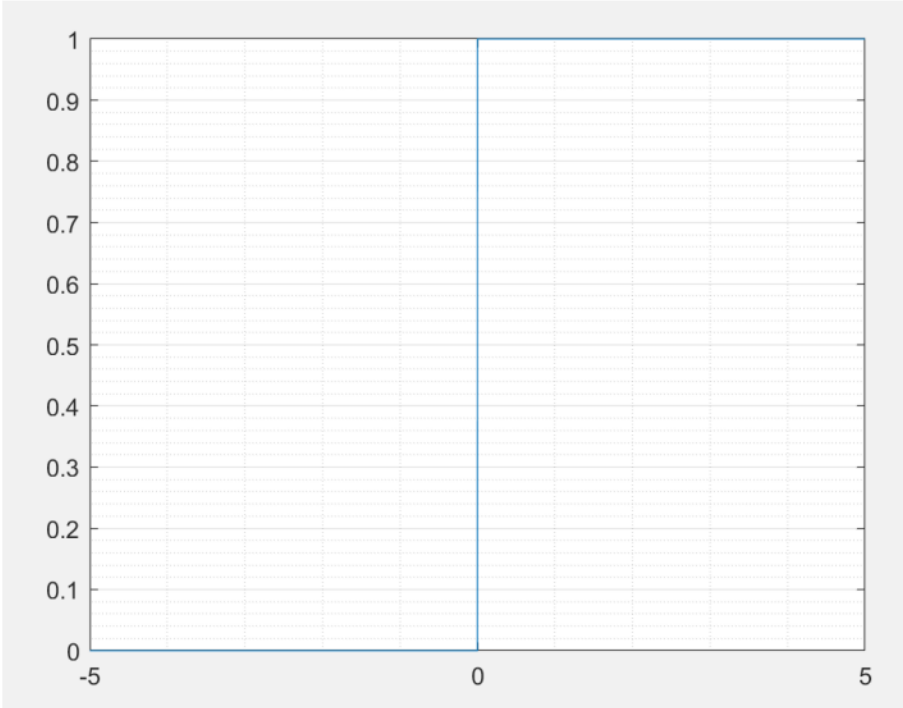


Figure 2.2: Step activation function for $th = 0$

Sigmoid function

If the classification task is not binary, then it is desirable to have different values of the output, depending on the importance of the information carried by this neuron. A very popular function used as activation is the *sigmoid*. The main advantage of this non-linear function is that is smooth, as shown in Figure 2.3 and for values of ξ around 0: even small changes of the input will affect the output significantly. Another advantage is that the derivative of this function is easily calculated, thus speeding the computation process.

$$g : \mathbb{R} \to \mathbb{R} : \xi \mapsto \frac{1}{1 + e^{-\xi}} \tag{2.4}$$

$$\frac{\partial g(\xi)}{\partial \xi} = g(\xi)\Big(g(\xi) - 1\Big) \tag{2.5}$$

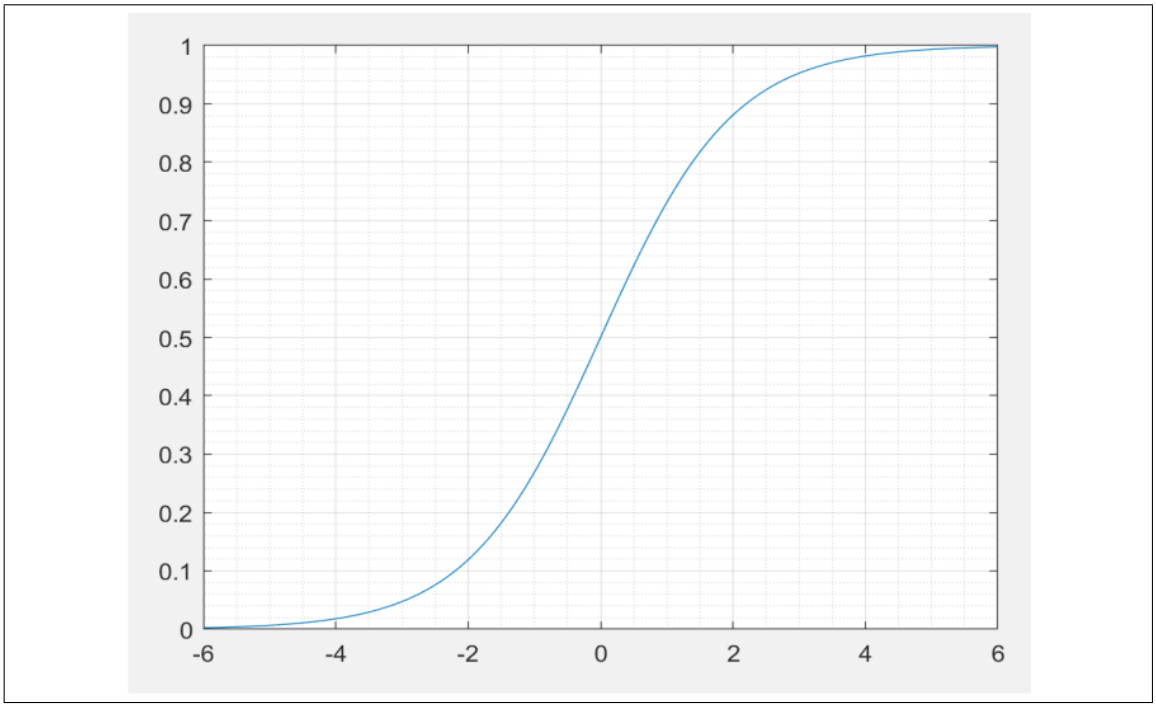


Figure 2.3: Sigmoid activation function

Hyperbolic Tangent (Tanh)

Though the sigmoid offers good results, it can have some problems finding the optimal solution. This is due to the fact that if a strongly negative input is provided, the output value will be very close to zero. Since the principle of NN

use the activation to later compute parameter gradients in the backpropagation process, the consequence of this fact is that the parameters won't be updated as they should, and the network will not converge to the optimal value. An alternative solution to sigmoid is the *Hyperbolic tangent*. This function has a shape similar to sigmoid, but instead outputs values in the interval $[-1, 1]$, as it can be observed in Figure 2.4. This solves the problem mentioned before, because the strongly negative inputs will provide negative outputs. Additionally, only for inputs very close to zero will be mapped to near-zero outputs. Also, the derivative of this function is easy to compute too.

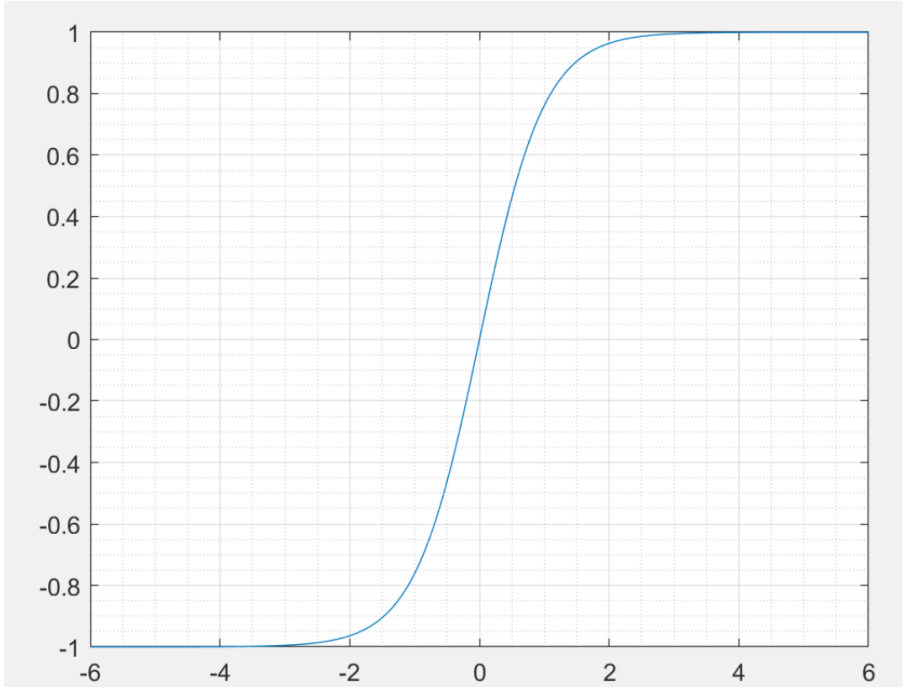


Figure 2.4: Hyperbolic Tangent activation function

$$g : \mathbb{R} \to \mathbb{R} : \xi \mapsto \frac{e^{\xi} - e^{-\xi}}{e^{\xi} + e^{-\xi}} \tag{2.6}$$

$$\frac{\partial g(\xi)}{\partial \xi} = 1 - g(\xi)^2 \tag{2.7}$$

Rectified Linear Unit (ReLu) function

ReLu is another popular activation function. It is very easy to compute because it involves very basic mathematical operations. Even if the function is not differentiable at 0, it is differentiable everywhere else. To deal with this inconvenient,

the simplest solution is to randomly assign a value for the derivative when $\xi = 0$. Most common values are: 0, 0.5 and 1. The ReLu function is displayed in Figure 2.5a.

$$g : \mathbb{R} \to \mathbb{R} : \xi \mapsto \begin{cases} 0, & \xi \leq 0 \\ \xi, & \xi > 0 \end{cases} \tag{2.8}$$

$$\frac{\partial g(\xi)}{\partial \xi} = \begin{cases} 0, & \xi < 0 \\ 1, & \xi > 0 \end{cases} \tag{2.9}$$

The main disadvantage of RelU is the so-called "dying" problem. If $\xi < 0$, then $g(\xi)$ will be 0. If for any reason the output of a *ReLU* is consistently 0, the gradient used in back propagation mechanism will also be 0. The error signal backpropagated from later layers gets multiplied by this 0, so no error signal ever passes to earlier layers. Thus, ReLu died.

To overcome this possible problem a variation of ReLu, called *leakyReLu*, has been proposed which solves the problem of *vanishing gradient*. As it can be observed in Figure 2.5b, the gradient is never 0 and this problem is avoided.

$$g : \mathbb{R} \to \mathbb{R} : \xi \mapsto \begin{cases} \alpha\xi & \xi \leq 0 \\ \xi, & \xi > 0 \end{cases} \tag{2.10}$$

$$\frac{\partial g(\xi)}{\partial \xi} = \begin{cases} \alpha, & \xi < 0 \\ \xi, & \xi > 0 \end{cases} \tag{2.11}$$

2.2.3 Proximity operators

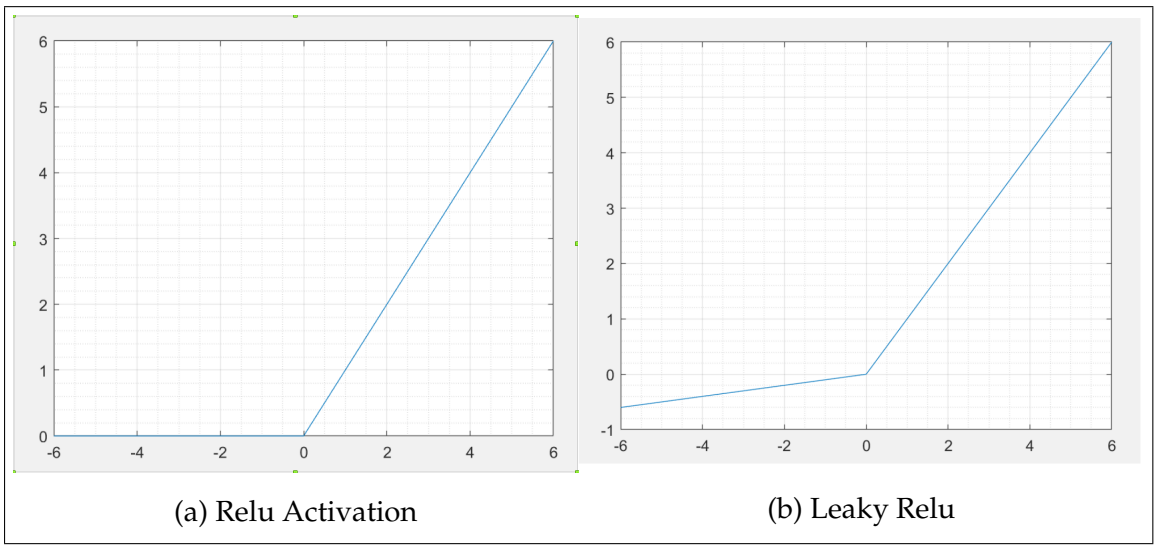
Definition

Let $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ be a closed convex function, i.e. its *epigraph*

$$epi f = \{(x, t) \in \mathbb{R}^n \times \mathbb{R} \mid f(x) \leq t\}$$

is a nonempty closed convex set. The *proximal operator* $\mathbf{prox}_f : \mathbb{R}^n \to \mathbb{R}^n$ of f is defined at $v \in \mathbb{R}^n$ as

$$\mathbf{prox}_f(v) = \underset{x}{argmin}(f(x) + \frac{1}{2} \parallel x - v \parallel_2^2) \tag{2.12}$$



(a) Relu Activation (b) Leaky Relu

Figure 2.5: ReLu and Leaky ReLu Activation Functions

where $\|\cdot\|_2$ is the Euclidean norm. The function minimized on the right hand side is strongly convex, so it has a unique minimizer for every $v \in \mathbb{R}^n$ [1].

The effect of the *prox operator* can be visualized in Figure 2.6.

The thick black line represents the domain boundary, as the thin ones are the level curves of function f. Evaluating $prox_f$ in various points (blue) results in a mapping to the corresponding red points. Note that all points will move towards the global minima of the function. The points that are inside the domain will remain inside, where the outside points will move towards the limit of the domain.

The proximal operator f can be viewed as a kind of a implicit gradient step for the function f. It is clear that there exists a close connection between proximal operators and gradient methods, which leads to the conclusion that this operators plays an important role in optimization.

Fixed points

One of the most important property of proximal operators is its *fixed point property*, which states that the point x^* minimizes f if and only if

$$x^* = \mathbf{prox}_f(x^*)$$

In other words, x^* is a minimizer of f if x^* is a fixed point of $\mathbf{prox}_f$. This fundamental property offers a connection between proximal operators and fixed point theory. Proximal algorithms for optimization can be interpreted as methods for finding fixed points of appropriate operators. We can minimize f by finding a

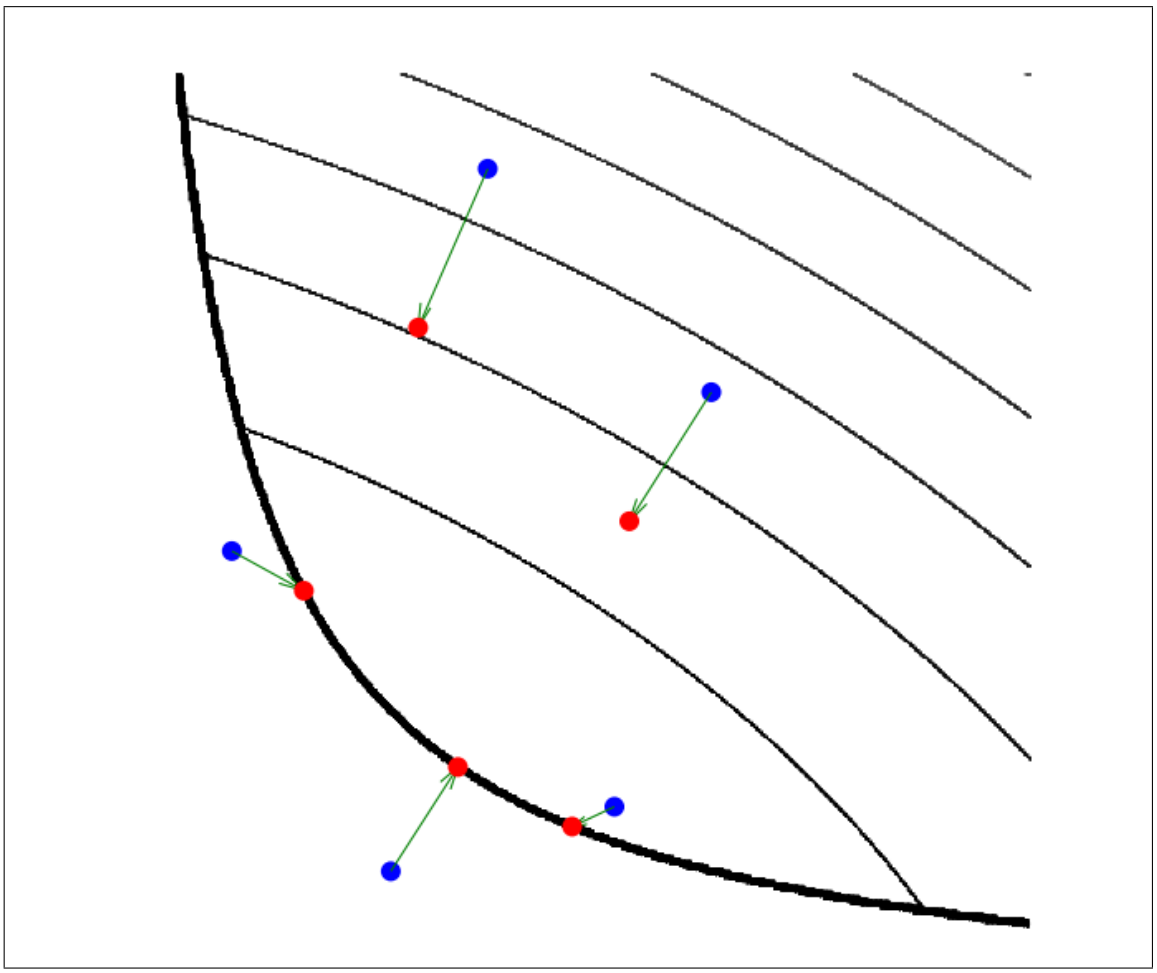


Figure 2.6: Evaluation a proximal operator at various points [1]

fixed point of its porximal operator. If $prox_f$ is continuous in Lipschitz sense with a constant less than 1, applying the operator multiple times would obviously find a fixed point. One may wonder what happens under less restrictive assumptions.

Actually, $\mathbf{prox}_f$ has another very important property: it is **firmly nonexpensive**, which is very useful for fixed point iteration:

$$\| \mathbf{prox}_f(x) - \mathbf{prox}_f(y) \|_2^2 \leq (x - y)^T (\mathbf{prox}_f(x) - \mathbf{prox}_f(y)) \tag{2.13}$$

for every $x, y \in \mathbb{R}^n$

Firmly nonexpansive operators represent instances of nonexpansive operators that have Lipschitz constant 1. Usually, iterating a nonexpansive operator is not sufficient to converge to a fixed point. However, if we consider Q - *nonexpansive*, then the operator $T = (1 - \alpha)Id + \alpha Q$ with $\alpha \in (0, 1)$ has the same fixed points as Q and an iteration of T will converge to a fixed point of T, and consequently of Q.

This kind of operators are called α - *averaged operators*. Firmly nonexpansive operators are special cases of averaged operators, for which $\alpha = \frac{1}{2}$. In this particular case, operator T has the following form:

$$T = \frac{1}{2}(Id + Q) \tag{2.14}$$

Because of this property, iterating $\mathbf{prox}_f$ is guaranteed to yield asymptotically a minimizer of f provided that it exists. Such kind of iteration is known as the *proximal point algorithm*.

Proximal activation in neural networks

In [25] it is mathematically proven that a wide range of activation operators that are currently used in neural networks applications are actually proximity operators of convex functions. In a Hilbertian setting, a neural network having n layers can be viewed as a composition of operators:

$$R_n \circ (W_n \cdot + b_n) \circ \cdots \circ R_1 \circ (W_1 \cdot + b_1),$$

where $R_i : \mathcal{H}_i \to \mathcal{H}_i$ is the non-linear activation operator, $\mathcal{H}_i$ is some Hilbert space, $W_i : \mathcal{H}_{i-1} \to \mathcal{H}_i$ is a linear operator, called *weight* and $b_i \in \mathcal{H}_i$ is the *bias* term.

1. **Sigmoid activation function**

   The unimodal *Sigmoid* activation function

   $$g : \mathbb{R} \to \mathbb{R} : \xi \mapsto \frac{1}{1 + e^{-\xi}} - \frac{1}{2} \tag{2.15}$$

   can be written as $g = \mathbf{prox}_\phi$, where

   $$\phi : \mathbb{R} \to]-\infty, +\infty]$$
   $$\xi \mapsto \begin{cases} (\xi + 1/2)\ln(\xi + 1/2) + (1/2 - \xi)\ln(1/2 - \xi) - 1/2(\xi^2 + 1/4), & |\,\xi\,| < 1/2; \\ -1, & |\,\xi\,| = 1/2; \\ +\infty, & |\,\xi\,| > 1/2. \end{cases} \tag{2.16}$$

2. **Hyperbolic tangent activation function**

   The *Hyperbolic tangent* activation function is the proximity operator of

   $$\phi : \mathbb{R} \to]-\infty, +\infty] : \xi \mapsto \begin{cases} \frac{(1+\xi)\ln(1+\xi)+(1-\xi)\ln(1-\xi)-\xi^2}{2}, & |\,\xi\,| < 1; \\ \ln(2) - 1/2, & |\,\xi\,| = 1; \\ +\infty, & |\,\xi\,| > 1. \end{cases} \tag{2.17}$$

3. **ReLu activation function**

   The *rectified linear unit* activation can be written as the *prox* of the indicator function of $[0, +\infty[$.

4. **Parametric rectified linear unit activation function**

   The *Parametric ReLu – LeakyRelu* satisfies $g = \mathbf{prox}_{\phi}$ when

$$\phi : \mathbb{R} \to]-\infty, +\infty] : \xi \mapsto \begin{cases} 0, & \xi > 0 \\ (1/\alpha - 1)\xi^2/2, & \xi \leq 0. \end{cases} \tag{2.18}$$

2.2.4 Loss functions

The actual training of the network is performed by minimizing a loss function. The loss function is the difference between the current output of the network $\hat{y}$ and the desired output (reference), denoted by y. It is an error metric, indicating the network precision when using the current output. The accuracy of the model increases as the value of the loss function decreases.

Mean Squared Error (MSE) MSE, also called *quadratic loss function*, is mostly used in linear regression tasks, since by minimizing MSE, one can then find the line which minimizes the distance (as a sum) of each point to the separation line. The most popular form of MSE is defined as follows:

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^{n} (y^{(i)} - \hat{y}^{(i)})^2 \tag{2.19}$$

The difference $y^{(i)} - (\hat{y^{(i)}})$ is called *residual* and the goal is to have this value as small as possible. However, if it is used in combination with *sigmoid* as the activation function, MSE may encounter some slow convergence problem. For other activation functions, this problem is alleviated.

Mean Squared Logarithmic Error (MSLE) MSLE is loss function derived from MSE, defined as follows:

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^{n} (\log(y^{(i)} + 1) - \log(\hat{y^{(i)}} + 1))^2 \tag{2.20}$$

This loss function computes the difference beetwen the log of the predictions and the references. It is mostly used in the cases when is not desired to penalize

huge differences between the two values, in particular when working with large numbers. MLSE penalizes under-estimates more than over-estimates.

Mean Absolute Error (MAE)

MAE is defined as

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n} \mid y^{(i)} - \hat{y}^{(i)} \mid^2 \tag{2.21}$$

where $|\cdot|$ represents the absolute value. MAE is very similar to MSE but is more robust to outliers than MSE since it does not use squares. Although MAE is easily differentiable when none of the components of the residual vanishes, thus making training algorithm less resource-consuming, it is non differentiable when this condition is not satisfied.

L2

L2 loss function is computed as the square of the L2 norm of the difference between the reference and the predicted value. Mathematically is similar to MSE, the only difference is that L2 is not divided by the number of examples.

$$\mathcal{L} = \sum_{i=1}^{N}(y^{(i)} - \hat{y}^{(i)})^2 \tag{2.22}$$

L1

L1 loss function is the sum of absolute errors between the predicted and the reference value. Mathematically is similar to MAE, the only difference is that L1 is not divided by the number of examples.

$$\mathcal{L} = \sum_{i=1}^{n} \mid y^{(i)} - \hat{y}^{(i)} \mid^2 \tag{2.23}$$

Kullback Leibler (KL) Divergence

KL divergence represents a metric for the difference between one probability distribution and another probability distribution, called reference. The KL loss function is computed as follows:

$$\mathcal{L} = \underbrace{\frac{1}{n}\sum_{i=1}^{n}(y^{(i)} \log y^{(i)})}_{entropy} - \underbrace{\frac{1}{n}\sum_{i=1}^{n}(y^{(i)} \log \hat{y}^{(i)})}_{cross-entropy} \tag{2.24}$$

KL is not a distribution-wise symmetric metric. If KL is close to 0, it means that the two distributions have a quite similar behavior.

Cross Entropy (CE)

Cross Entropy is a very popular loss function that is mostly used in binary classification, but there is an extension of it, called Multi-class Cross Entropy, used for multi classification. The expression is given by

$$\mathcal{L} = -\frac{1}{n}\sum_{i=1}^{n}(y^{(i)}\log\hat{y^{(i)}} + (1-y^{(i)})\log(1-\hat{y^{(i)}})) \tag{2.25}$$

Negative Logarithmic Likelihood (NCL)

This loss function is also very commonly used, since it measures the accuracy of the classifier. It is used when a "soft" measurement is wanted, more precisely when it is desired that the model outputs the probability of each class. It can be viewed as a confidence score. Mathematically, it is expressed as

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}\log(\hat{y}^{(i)}) \tag{2.26}$$

Hinge

Hinge loss is also called max-margin objective or *multiclass SVM loss* because it is mostly used with SVMs (Support Vector Machine). The value of the loss is proportional with the error. Hinge loss will output zero if the classifier prediction $\hat{y}$ is correct. It is given by

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}max(0, m - y^{(i)}\hat{y^{(i)}}) \tag{2.27}$$

2.2.5 Backpropagation and optimization

After the loss function is computed, the next step is to find the values that minimize thiss function. This is done with the help of the backpropagation algorithm. This was originally introduced in the 1970s, but it was first use in the context of NN in the late 1980s. [26] demonstrated that using backpropagation algorithm in NN works far faster than all the earlier approaches. Nowadays backpropagation is commonly used in all NN architectures.

Backpropagation of errors is an iterative algorithm that uses the chain rule to compute the gradients (partial derivatives) $\frac{\partial \mathcal{L}}{\partial w}$, $\frac{\partial \mathcal{L}}{\partial b}$ of the loss function with respect to all the weights w (and bias b) in the network. This algorithm is used only for computing the gradients though the network. Finding the global minimum and updating the parameters to obtain the optimal model in the shortest time is a different task, done by optimization algorithms.

Optimization algorithms can be split in 2 major categories:

- **First Order Optimization Algorithms:** These methods exploit the monotonicity of the function (if it is increasing or decreasing) in a certain point, basically returning a line which is tangent to a point on the loss surface. The most commonly used algorithm is the *Gradient Descent*, which finds the minimum of the loss function based on the gradients with respect to the parameters.

- **Second Order Optimization Algorithms:** These algorithms use the second order derivative, also known as Hessian matrix to minimize the loss function. The Hessian represents a matrix containing all the second order partial derivatives. This introduces new information, about the curvature of the surface which can be a valuable information in finding the global minimum of the loss function, often accelerating the convergence of the algorithm around a local minimum.

Chapter 3

Learning with constraints

3.1 Motivation

Neural network are vulnerable to adversarial manipulation of their input that can cause incorrect classification. To formulate the problem, suppose T a learning system and x a clean input data that is correctly classified by the system: $T(x) = y_{true}$. It is possible to construct an adversarial sample $\tilde{x}$, which may be perceptually indistinguishable from x, by adding a small perturbation z: $\tilde{x} = x + z$. this new sample, though very similar to the original one, will be classified incorrectly: $T(\tilde{x}) \neq y_{true}$.

In neural networks, Lipschitz constant can be used to assess the robustness against adversarial inputs. The Lipschitz constant upper bounds the relationship between input perturbation and output variation for a given distance [27]. In other words, for the system T described above, we can majorize the effect of the perturbation z using the following inequality[3]:

$$|| T(x + z) - T(x) ||\leq \theta ||z||, \tag{3.1}$$

where θ represents the Lipschitz constant of the system.

3.2 Computation of Lipschitz constant

Consider $f : \mathbb{R}^n \to \mathbb{R}^m$. The function f is called *Lipschitz continuous* if there exists a constant θ such that

$$\| f(x) - f(y) \|_2 \leq \theta \| x - y \|_2, \quad \forall x, y \in \mathbb{R}^n \tag{3.2}$$

Any θ for which (3.2) holds is called a *Lipschitz constant* of f. The smallest Lischitz constant is called the *Lipschitz modulus* of f. If f is a continuously differentiable function, then its Lipschitz modulus is the maximum spectral norm (maximum singular value) of its Jacobian over its domain, that is

$$\theta = \sup_x \|\nabla f(x)\|_S. \tag{3.3}$$

In the context of NN, accurately computing Lipschitz constant is a NP hard problem, even for a shallow network [27]. Methods for accurately computing the Lipschitz constant of a NN has been proposed in [27], [12].

If we come back to the scenario presented in Chapter 2, in which we have non-zero Hilbert spaces $(\mathcal{H}_i)_{0 \leq i \leq m}$ with $m \geq 1$. Consider $\forall i \in \{1, \dots, m\}$ the linear weight operator $W_i : \mathcal{H}_{i-1} \to \mathcal{H}_i$, the bias parameter $b_i \in \mathcal{H}_i$ and the nonlinear activation operator $R_i : \mathcal{H}_i \to \mathcal{H}_i$, an α_i-averaged operator with $\alpha_i \in [0, 1]$. A NN T can be defined as a composition of operators defined as follows:

$$T = T_m \circ \cdots \circ T_1, \quad \textit{where} \quad T_i : \mathcal{H}_{i-1} \to \mathcal{H}_i : x \mapsto R_i(W_i x + b_i) \quad \forall i \in \{1, \dots, m\}. \tag{3.4}$$

Since all the nonlinear activation operators are nonexpansive, a rough approximation of the Lipschitz constant of the system T is given by the following formula:

$$\theta_m = \prod_{i=1}^{m} \|W_i\|_S. \tag{3.5}$$

This approximation is valid in the context of this thesis, since it was shown in Chapter 2 that most used activation functions are actually proximity operators, hence they are nonexpansive. Hence, the robustness of the network can be controlled by imposing spectral norm constraints on the weights of each layer.

Furthermore, a fundamental result shown in [12] states that if the activation functions are **firmly nonexpansive** (which is a property satisfied by proximity operators), then the Lipschitz constant of the NN is lower bounded by

$$\tilde{\theta}_m = \left\| \prod_{i=1}^{m} W_i \right\|_S. \tag{3.6}$$

In addition, if, for every $i \in \{1, \ldots, m\}$, all the weights W_i are **nonnegative**, then $\tilde{\theta}_m$ is the Lipschitz modulus of the NN. This result is remarkable since it shows that a NN with nonnegative weights has the same Lipschitz continuity properties than the equivalent fully linear network where the nonlinear activation functions are removed.

In this thesis, several methods of ensuring a 1-Lipschitzian system were implemented and later tested in the context of automatic gesture recognition. Also, it evaluates the effect that training with spectral norm constraints has on the overall performance of the system.

3.3 Proposed constraints

Several convex constraints have been considered as follows:

1. **Positivity** – constraining all weights to be positive is very important since is a compulsory condition for ensuring that we indeed compute the smallest Lipschitz constant of the network. This corresponds to the constraint set:

$$D = \{W \mid W \geq 0\}. \tag{3.7}$$

2. **Norm of layer** – the simplest way to impose a constraint on the product of the norms of the weights of all the layers in the network is to constrain each weight individually, as in Equation (3.8), leading the constraint set:

$$C_1 = \{W \mid \ \| W_i \| \leq \alpha, \ \ \forall i \in \{1, \ldots, m\}\}. \tag{3.8}$$

3. **Product of norms** – imposing the constraint on the product of all the norms is a weaker constraint, since it gives the network more freedom to learn its

parameters. This yields

$$C_2 = \{W \mid \prod_{i=1}^{m} \parallel W_i \parallel \leq \alpha\}. \tag{3.9}$$

4. **Norm of product** – constraining the norm of the product of the weights of each layer will lead to the control of the real Lipschitz modulus of the network, provided that the conditions aforementioned (firm nonexpansiveness and positivity) are respected. The associated constraint set is

$$C_3 = \left\{W \mid \left\| \prod_{i=1}^{m} W_i \right\| \leq \alpha\right\}. \tag{3.10}$$

3.3.1 Dealing with constraints

Implementing the aforementioned constraints will be done in the optimization part of the training process, where the weights are updated with respect to the gradient of a loss function. This can be done by using classical methods as *Gradient descent* or some of its variants (*Stochastic Gradient Decent (SGD)*, *Vanilla Gradient Descent*, etc.) or by using more complex optimizers like *Adam*.

All optimizers, regardless their complexity, are updating the weights after each iteration based on the following principle:

$$W_{n+1} = W_n - \gamma_n \nabla \mathcal{L}(W_n), \tag{3.11}$$

where $\nabla \mathcal{L}(W_n)$ is the gradient of the loss function $\mathcal{L}$ at the previous iteration, and γ_n is the learning rate of the system. The gradient is computed using a backprobagation algorithm [28], transferring the loss layer by layer upwards.

So, to introduce some non-empty constraint set C it it necessary to replace the previous iteration by a projected gradient, given by the following relation:

$$W_{n+1} = P_C(W_n - \gamma_n \nabla \mathcal{L}(W_n)), \tag{3.12}$$

where P_C is the projection onto C. Such projection is guaranteed to be defined if C is a closed convex set.

However, in this thesis we assume all the proposed constraints ($C_1 - C_3$) under the additional non-negativity condition (D) in order to satisfy the conditions for Equation (3.6) to be valid. Hence, the problem becomes more complex since the task now is to calculate the projection onto the intersection of two convex sets, which is usually difficult to solve. Mathematically, Equation (4.12) becomes:

$$W_{n+1} = P_{C \cap D}(W_n - \gamma_n \nabla \mathcal{L}(W_n)), \tag{3.13}$$

where $P_{C \cap D}$ represents the projection onto the intersection of the two sets. A final difficulty is that C_1 is a convex set, but C_2 and C_3 are not.

3.4 Computing the projection

Computing the projection $P_{C \cap D}$ is a complex task, but there are some iterative algorithms [29] that can compute the intersection of the two sets.

Approximate projection – The first approached that was considered in this work, was to compute an approximate projection by projecting first onto D, then onto C:

$$P_{C_1 \cap D} \approx P_{C1} \circ P_D \approx \hat{P}_{C1} \circ P_D$$

This approach, although not the true projection is easy to implement and efficient from the computational cost view point. We went one step further, and computed the true approximate projection $\hat{P}_{C1} \circ P_D$ using the *DFB* algorithm described bellow, by assuming $A = B = Id$.

True Projection – For computing the true projection, the Dual Forward-Backward algorithm formulated bellow was implemented.

3.4.1 Formulating the problem

We want to compute the projection onto $C \cap D$ where

$$C = \left\{ X \in \mathbb{R}^{n \times m} \;\middle|\; \|AXB\|_S \leq \rho \right\} \tag{3.14}$$

where $A \in \mathbb{R}^{p \times n} \setminus \{0\}$ and $B \in \mathbb{R}^{m \times q} \setminus \{0\}$, and $D = [0, +\infty[^{n \times m}$. Here-above $\| \cdot \|_S$ designates the spectral norm, but we will also use the Frobenius norm

$$(\forall X \in \mathbb{R}^{n \times m}) \quad \|X\| = \sqrt{\mathrm{tr}(XX^\top)} \tag{3.15}$$

and the associated inner product

$$(\forall (X, Y) \in (\mathbb{R}^{n\times m})^2) \quad \langle X \mid Y \rangle = \mathrm{tr}(XY^\top). \tag{3.16}$$

More precisely, the projection will be performed in the Hilbert space of real-valued matrices of size $n \times m$ endowed with the latter norm. In this Hilbert space, C and D are closed convex sets with a nonempty intersection. For a given matrix $\overline{X} \in \mathbb{R}^{n\times}$, we will thus be interested in computing $\widehat{X} = \mathsf{P}_{C\cap D}(\overline{X})$, which is defined as:

$$\widehat{X} = \underset{X\in C\cap D}{argmin} \|X - \overline{X}\|. \tag{3.17}$$

3.4.2 Reformulation

Let us first note that

$$\begin{aligned} \mathcal{L}\colon \mathbb{R}^{n\times m} &\to \mathbb{R}^{p\times q} \\ X &\mapsto AXB \end{aligned} \tag{3.18}$$

is a linear operator. The norm of this operator is:

$$\|\mathcal{L}\| = \sup_{X\in\mathbb{R}^{n\times m}\setminus\{0\}} \frac{\|AXB\|}{\|X\|}. \tag{3.19}$$

Let vec be the column stacking operator. For every $X \in \mathbb{R}^{n\times m}$,

$$AXB = (B^\top \otimes A)\,\mathrm{vec}(X) \tag{3.20}$$

where $\otimes$ is the Kronecker product. This allows us to deduce that

$$\|\mathcal{L}\| = \|B^\top \otimes A\|_{\mathrm{S}}. \tag{3.21}$$

Since

$$\begin{aligned} \|B^\top \otimes A\|_{\mathrm{S}}^2 &= \|(B^\top \otimes A)(B^\top \otimes A)^\top\|_{\mathrm{S}} \\ &= \|(B^\top \otimes A)(B \otimes A^\top)\|_{\mathrm{S}} \\ &= \|(B^\top B) \otimes (AA^\top)\|_{\mathrm{S}} \\ &= \|AA^\top\|_{\mathrm{S}}\|B^\top B\|_{\mathrm{S}} \\ &= \|A\|_{\mathrm{S}}^2\|B\|_{\mathrm{S}}^2, \end{aligned} \tag{3.22}$$

we have

$$\|\mathcal{L}\| = \|A\|_S \|B\|_S. \tag{3.23}$$

In addition, the adjoint of this operator is $\mathcal{L}^*\colon \mathbb{R}^{p\times q} \to \mathbb{R}^{n\times m}$, which according to its definition is given by such that

$$\begin{aligned}(\forall X \in \mathbb{R}^{n\times m})(\forall Y \in \mathbb{R}^{p\times q}) \qquad & \langle \mathcal{L}(X) \mid Y\rangle = \langle X \mid \mathcal{L}^*(Y)\rangle \\ & \Leftrightarrow \operatorname{tr}(\mathcal{L}(X)Y^\top) = \operatorname{tr}(X(\mathcal{L}^*(Y))^\top) \\ & \Leftrightarrow \operatorname{tr}(XBY^\top A) = \operatorname{tr}(X(\mathcal{L}^*(Y))^\top),\end{aligned} \tag{3.24}$$

which allows us to deduce that

$$(\forall Y \in \mathbb{R}^{p\times q}) \quad \mathcal{L}^*(Y) = A^\top Y B^\top. \tag{3.25}$$

Using the operator $\mathcal{L}$, Equation 3.17 can be recast as:

$$\widehat{X} = \underset{X\in\mathbb{R}^{n\times m}}{argmin} \frac{1}{2}\|X - \overline{X}\|^2 + \iota_{\mathcal{B}(0,\rho)}(\mathcal{L}(X)) + \iota_D(X) \tag{3.26}$$

where $\mathcal{B}(0,\rho)$ is the closed ball defined as

$$\mathcal{B}(0,\rho) = \left\{X \in \mathbb{R}^{n\times m} \;\middle|\; \|X\|_S \leq \rho\right\} \tag{3.27}$$

and ι_E denotes the indicator function of a set E (equal to 0 on this set and $+\infty$ otherwise).

3.4.3 Algorithm

Problem stated by Equation 3.26 has no explicit solution and has thus to be solved by an iterative method. A possible solution is the dual Forward-Backward algorithm [29]:

Let $Y_0 \in \mathbb{R}^{p\times q}$.

Set $\epsilon \in]0, 1/\|\mathcal{L}\|^2[$.

For $n = 0, 1, \ldots$

$$\begin{cases} \text{Set } \gamma_n \in [\epsilon, 2/\|\mathcal{L}\|^2 - \epsilon] & \text{(3.28a)} \\ X_n = \mathsf{P}_D(\overline{X} - \mathcal{L}^*(Y_n)) & \text{(3.28b)} \\ \widetilde{Y}_n = Y_n + \gamma_n \mathcal{L}(X_n) & \text{(3.28c)} \\ Y_{n+1} = \widetilde{Y}_n - \gamma_n \mathsf{P}_{\mathcal{B}(0,\rho)}(\gamma_n^{-1}\widetilde{Y}_n). & \text{(3.28d)} \end{cases}$$

It can be shown that the sequence $(X_n)_{n\in\mathbb{N}}$ generated by Algorithm (3.28) converges to $\widehat{X}$. In view of the properties established in Section 3.4.2, the algorithm can be rewritten in a more tractable form:

Let $Y_0 \in \mathbb{R}^{p\times q}$.

Set $\epsilon \in]0, 1/(\|A\|_{\mathrm{S}}\|B\|_{\mathrm{S}})^2[$.

For $n = 0, 1, \ldots$

$$\begin{cases} \text{Set } \gamma_n \in [\epsilon, 2/(\|A\|_{\mathrm{S}}\|B\|_{\mathrm{S}})^2 - \epsilon] & \text{(3.29a)} \\ X_n = \mathsf{P}_D(\overline{X} - A^\top Y_n B^\top) & \text{(3.29b)} \\ \widetilde{Y}_n = Y_n + \gamma_n A X_n B & \text{(3.29c)} \\ Y_{n+1} = \widetilde{Y}_n - \gamma_n \mathsf{P}_{\mathcal{B}(0,\rho)}(\gamma_n^{-1}\widetilde{Y}_n). & \text{(3.29d)} \end{cases}$$

In addition, the expression of the required projections are provided below. For every $X = (X_{i,j})_{1\le i\le n, 1\le j\le m} \in \mathbb{R}^{n\times m}$,

$$\mathsf{P}_D(X) = (\widetilde{X}_{i,j})_{1\le i\le n, 1\le j\le m}, \tag{3.30}$$

where

$$(\forall i \in \{1,\ldots,n\})(\forall j \in \{1,\ldots,m\}) \quad \widetilde{X}_{i,j} = \begin{cases} X_{i,j} & \text{if } X_{i,j} \ge 0 \\ 0 & \text{otherwise.} \end{cases} \tag{3.31}$$

For every $Y = (Y_{i,j})_{1\le i\le p, 1\le j\le q} \in \mathbb{R}^{p\times q}$, let $Y = U\Lambda V^\top$ be the singular value decomposition of Y, where $U \in \mathbb{R}^{p\times r}$ and $V \in \mathbb{R}^{q\times r}$ are matrices such that $U^\top U = \mathrm{Id}$ and $V^\top V = \mathrm{Id}$, $r = \min\{p, q\}$, and $\Lambda = \mathrm{Diag}(\lambda_1, \ldots, \lambda_r) \in [0, +\infty[^{r\times r}$. Then $\mathsf{P}_{\mathcal{B}(0,\rho)}(Y) = U\widetilde{\Lambda}V^\top$ where $\widetilde{\Lambda} = \mathrm{Diag}(\widetilde{\lambda_1}, \ldots, \widetilde{\lambda_r})$ and

$$(\forall i \in \{1,\ldots,r\}) \quad \widetilde{\lambda_i} = \begin{cases} \lambda_i & \text{if } \lambda_i \le \rho \\ \rho & \text{otherwise.} \end{cases} \tag{3.32}$$

Note also that a FISTA-like accelerated version of the algorithm can be provided:

Let $Y_0 \in \mathbb{R}^{p\times q}$.

Set $\gamma = 1/(\|A\|_S\|B\|_S)^2$.

Set $\alpha \in]2, +\infty[$.

For $n = 0, 1, \ldots$

$$\begin{cases} \eta_n = \dfrac{n-1}{n+\alpha} & \text{(3.33a)} \\ Z_n = Y_n + \eta_n(Y_n - Y_{n-1}) & \text{(3.33b)} \\ X_n = \mathsf{P}_D(\overline{X} - A^\top Z_n B^\top) & \text{(3.33c)} \\ \widetilde{Y}_n = Z_n + \gamma A X_n B & \text{(3.33d)} \\ Y_{n+1} = \widetilde{Y}_n - \gamma \mathsf{P}_{\mathcal{B}(0,\rho)}(\gamma^{-1}\widetilde{Y}_n). & \text{(3.33e)} \end{cases}$$

In this thesis only algorithm 3.28 was implemented.

Chapter 4

Experimental setup

4.1 Data Acquisition

The hardware component is represented by Myo armband, a gesture recognition device worn on the forearm and manufactured by Thalmic Labs. Myo enables the user to control technology wireless using various hand motions. It uses a set of electromyographic (EMG) sensors that sense electrical activity in the forearm muscles, combined with a gyroscope, accelerometer and magnetometer to recognize gestures. Myo can be used to control video games, presentations, music and visual entertainment. It differs from the Leap Motion device as it is worn rather than a 3D array of cameras that sense motion in the environment [2].

Myo is made up of a set of eight segments containing muscular sensors used for signal processing, a port of loading USB, a LED which indicates the state and an extensible casing. The signal is transmitted to a central unit using Bluetooth technology. In Figure 4.1 it can be observed the outer structure and the layout of the items presented.

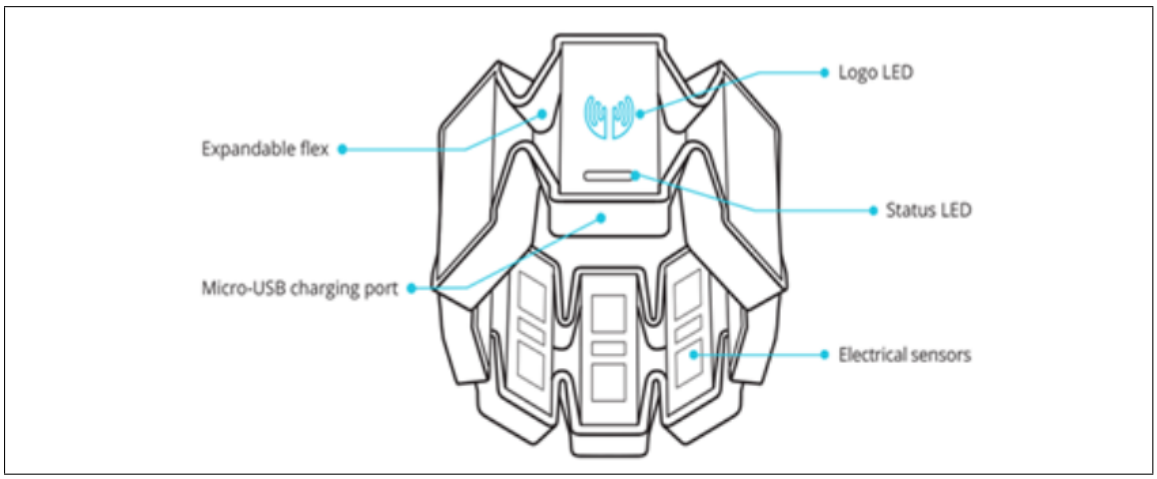


Figure 4.1: Elements of Myo armband [2]

Each segment contains electromyographic electrode units (EMG), and inside each segment is composed of an operational precision amplifier. Two of the segments are provided with batteries. On the main segment plate there is a processor MK22FN1MO – Freescale Kinetics Cortex M4 120MHz, a transmitter-receiver Bluetooth and a vibration engine. Here we can also find a motion sensor Invensense MPU-9150 9-dof with applications of gyroscope, accelerometer and magnetometer. The actual armband can be visualized in Figure 4.2.

Figure 4.2: Myo armband disassembled [2]

Myo armband is built to transmit two types of data: spatial and gestural. Spatial data informs the user about the orientation and movement of the hand and gestural data represent the actions of the user's hand. The Software Development Kit (SDK) puts at user's disposal some predefined gestures that can be recognized

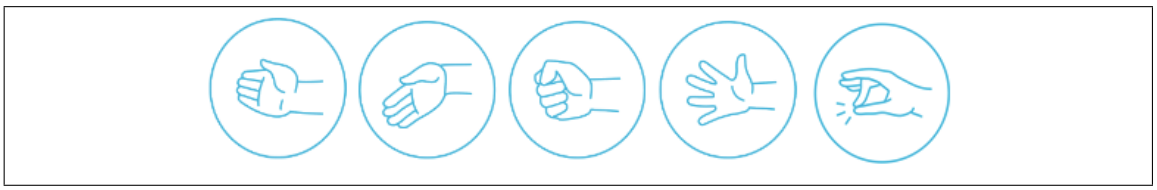


Figure 4.3: The predefined gestures of Myo

regardless of the user, after the bracelet is properly calibrated. The predefined gestures are represented by the movement of the hand to the right („Wave Right"), the movement of the hand to the left ("Wave Left"), tightening the fist ("Fist"), stretching the fingers ("Fingers Spread") and double tapping the fingers ("Double Tap"), as shown in Figure 4.3. The system works with an overall accuracy of $\approx 80\%$, given it is properly calibrated [30].

For this project, the gesture recognition system provided by the manufacturer was replaced by a trained neural network that classifies seven hand gestures to simulate all four spatial directions (right, left, up, down) along with grab and release movements. The main advantage the proposed method is that it does not need to synchronize with the bracelet to correctly recognize the movements (the original system requires to synchronize for proper functioning). The predefined hand gesture used (neutral, ulnar deviation, radial deviation, hand open, hand close, wrist extension and wrist flexion) will be described later in the thesis.

An example of a normalized 8-channels raw signal is depicted in Figure 4.4. This is a plot of the EMG activity over a period of 3s, when the user was doing a wrist flexion. The main advantage of the Myo armband is the fact that it is non-invasive and it can be used without any preparations. However, these benefits come with severe limitations since dry electrodes are not as accurate in capturing the EMG activity compared to gel-based ones.

4.2 Proposed method

This thesis proposes a real-time automatic gesture recognition system, for seven basic gestures, based on sEMG signals. The classification is performed using a fully connected Neural Network, whose training involves a free-source data set [1], also acquired with Myo armband, detailed in [3].

[1] https://github.com/Giguelingueling/MyoArmbandDataset

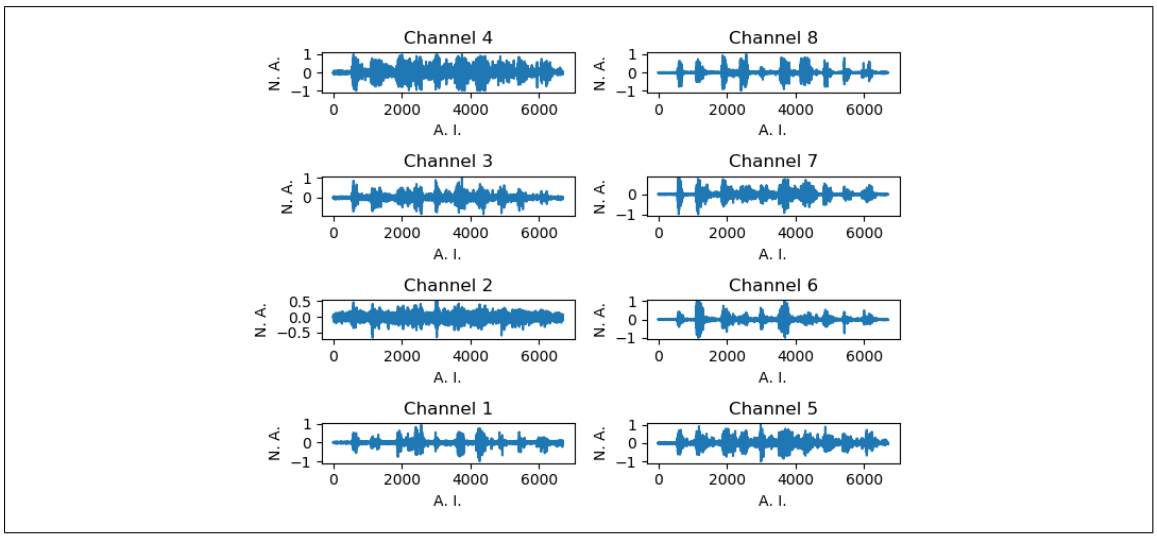


Figure 4.4: Raw EMG signal captured with Myo

4.2.1 Training Data Set

This data set consists in two sub-datasets: the first one serving as pretraining dataset,has signals acquired from 19able-bodied subjects.The second is the evaluation datasetand it is comprised of 17 subjects. For both sub-datasets, the labeled data was created by requiring the user to hold each gesture for 5 s. The recording of the full seven gestures for 5 s is referred to as a cycle, with four cycles forming a round. In the case of the pretraining dataset, a single round is available per subject. For the evaluation dataset three rounds are available with the first round utilized for training and the last two for testing. For each participant, the armband was systematically tightened to its maximum and slid up the user's forearm until the circumference of the armband matched that of the forearm.

Seven hand/wrist gestures are considered in this work. The first one referred to as neutral is the natural posture of the hand of the subject when no significant muscle activity is detected. The six other gestures are: ulnar deviation, radial deviation, hand open, hand close, wrist extension and wrist flexion. Figure 4.5 depicts all the gestures.

4.2.2 General Overview

A general overview of the proposed method is shown in Figure 4.6. The signals captured by the armband are transmitted to a computer via Bluetooth. The sampling frequency of the device is $Fs = 200Hz$, whilst for the analysis, we

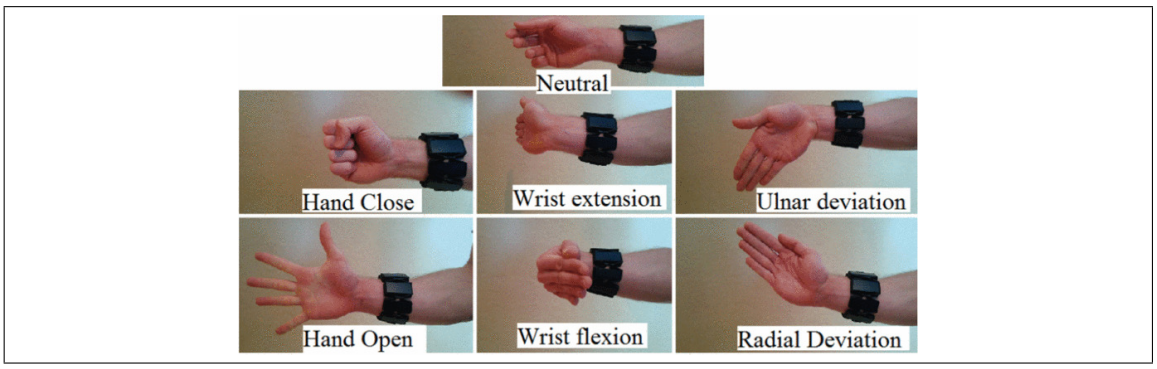


Figure 4.5: The seven hand gestures considered during the experiment [3]

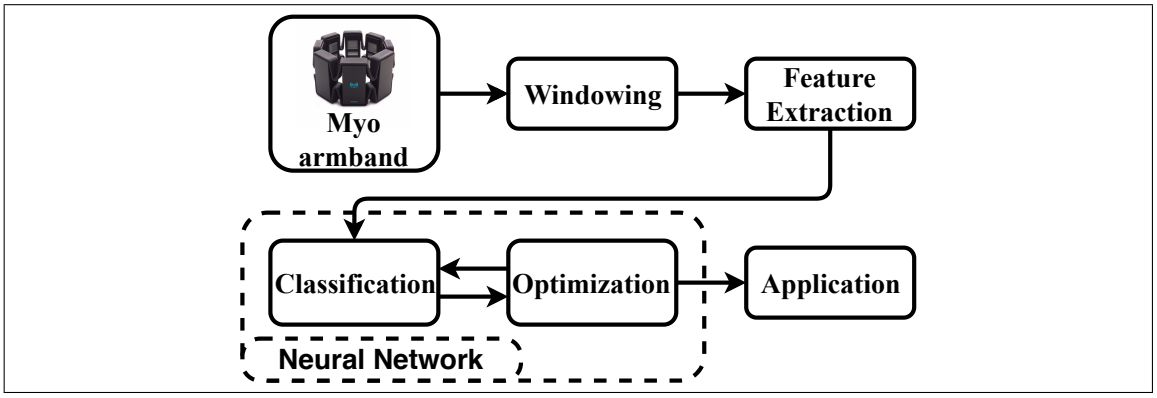


Figure 4.6: General overview of the system

considered a rectangle sliding window of 250 ms (50 samples) length, with an overlap between consecutive windows of 200 ms (40 samples). The window size is chosen to allow lower statistical variance in the feature sets and a continuous classification of the EMG signals.

4.2.3 Feature Extraction

Signal processing must be used to efficiently train the EMG data from all datasets. Recent advances in technologies of signal processing and mathematical models have made it possible to develop advanced EMG detection and analysis techniques. So far, research and extensive efforts have been made in the area, developing better algorithms, upgrading existing methodologies, improving detection techniques to reduce noise, to acquire accurate EMG signals.

Traditionally, one of the most researched aspects of sEMG based gesture recognition comes from feature engineering. In other words, manually finding a representation for sEMG signals that allows easy differentiation between gestures

while reducing the variance of examples within the same gestures. Over the years, several efficient combinations of features both in the time and frequency domain have been proposed [30], [31], [32], [33]. Features can be regrouped into different types, mainly: time, frequency and time-frequency domains. A short description of the most used features for gesture classification is presented below:

Time Descriptors

1. **Mean Absolute Value (MAV)** [31] – a feature returning the mean of a fully-rectified signal

$$MAV(x) = \frac{1}{L}\sum_{k=0}^{L-1}|x_k| \tag{4.1}$$

2. **Zero Crossing Rate (ZCR)** [15] – a feature that counts the frequency at which the signal passes through zero. A threshold α $geq0$ is used in order to lessen the impact of the noise. The value of this feature is incremented whenever the following condition is satisfied:

$$ZC(x) = |\Big\{k : (|x_k - x_{k-1}| \geq \alpha) \wedge (sgn(x_i) \neq sgn(x_{i-1}))\Big\}| \tag{4.2}$$

   where $sgn(a, b)$ return true if a and b (two real numbers) have the same sign and false otherwise. Note that depending on the slope of the signal and the selected, the zero-crossing point might not be detected.

3. **Slope Sign Changes (SSC)** [31] – a feature that measures the frequency at which the sign of the signal slope changes. Given three consecutive samples x_i, x_{i-1}, x_{i+1}, the SSC value will be incremented if:

$$SSC(x) = |k : (x_k - x_{k-1}) \cdot (x_k - x_{k+1}) \geq \alpha| \tag{4.3}$$

   where $\alpha > 0$ is employed as a threshold to reduce the mpact of the noise on this feature.

4. **Waveform Length (WL)** [15] – a feature that offers a simple characterization of the signal's waveform. It is computed as follows:

$$WL(x) = \sum_{k=1}^{L-1}|x_k - x_{k-1}| \tag{4.4}$$

5. **Skewness** – is the third central movement of a distribution that measures the overall asymmetry of a distribution. It is computed as follows:

$$Skewness(x) = \frac{1}{L}\sum_{k=0}^{L-1}\left(\frac{x_k - \overline{x}}{\sigma}\right)^3 \tag{4.5}$$

6. **Root Mean Square (RMS)** – this feature, also known as the quadratic mean, is closely related to the standard deviation as both are equal when the mean of the signal is zero.

$$RMS(x) = \sqrt{\frac{1}{L}\sum_{k=0}^{L-1} x_k^2} \tag{4.6}$$

7. **Hjorth Parameters** [31] – Hjorth parameters are a set of three features originally developed for characterizing electroencephalography signals and then successfully applied to sEMG signal recognition. Hjorth Activity Parameter can be thought of as the surface of the power spectrum in the frequency domain and corresponds to the variance of the signal calculated as follows:

$$Activity(x) = \frac{1}{L}\sum_{k=0}^{L-1}(x_k - \overline{x})^2 \tag{4.7}$$

where $\hat{x}_i$ is the mean of the i^{th} window. Hjorth Mobility Parameter is a representation of the mean frequency of the signal and it is calculated as follows:

$$Mobility(x_i) = \sqrt{\frac{Activity(x_i')}{Activity(x_i)}} \tag{4.8}$$

where x_i' is the first derivative with respect to time of the signal for the i^{th} window. Similarly, the Hjorth Complexity Parameter, which represents the change in frequency, is computed as follows:

$$Complexity(x_i) = \frac{Mobility(x_i')}{Mobility(x_i)} \tag{4.9}$$

8. **Integrated EMG (IEMG)** [34] – a feature returning the sum of the fully-rectified signal.

$$IEMG(x) = \sum_{k=0}^{L-1} \mid x_k \mid \tag{4.10}$$

9. **Autoregression Coefficient (AR)** [34] – an autoregressive model tries to predict future data, based on a weighted average of the previous data. This model characterizes each sample of the signal as a linear combination of the previous sample with an added white noise. The number of coefficients calculated is a trade-off between computational complexity and predictive power. The model is defined as follows:

$$x_{i,k} = \sum_{i=1}^{P} \rho_j x_{i,k-j} + \epsilon_t \tag{4.11}$$

where P is the model order, ρ_j is the j^{th} coefficient of the model and ϵ_t represents the residual white noise.

10. **Sample Entropy (SampEn)** – entropy measures the complexity and randomness of a system. Sample Entropy is a method which allows entropy estimation.

$$SampEn\Big(x_i, m, r\Big) = \ln\Big(\frac{A^m(r)}{B^m(r)}\Big) \tag{4.12}$$

11. **EMG Histogram (HIST)** [35] – when a muscle is in contraction, the EMG signal deviates from its baseline. The idea behind HIST is to quantify the frequency at which this deviation occurs for different amplitude levels. HIST is calculated by determining a symmetric amplitude range centered around the baseline. This range is then separated into n bins of equal length (n is a hyperparameter). The HIST is obtained by counting how often the amplitude of the signal falls within each bin boundaries.

Frequency Descriptors

1. **Marginal Discrete Wavelet Transform (mDWT)** [36] – is a feature that removes the time-information from the discrete wavelet transform to be insensitive to wavelet time instants. The feature instead calculates the cumulative energy of each level of the decomposition. The mDWT is defined as follows:

$$m_{x_k}(s) = \sum_{u=0}^{\frac{N}{2S-1}} \mid d_{x_i}(s,u) \mid, \qquad \forall s = 1, \ldots, S \tag{4.13}$$

   where the number of coefficients of the discrete wavelet transform $d_{x_i}(s,u)$ is equal to N and $S = [log_2 N_c]$ is the deepest level of decomposition.

2. **Cepstral Coefficients** [37] – the cepstrum of a signal is the inverse Fourier transform of the log power spectrum magnitude of the signal. Like the AR, the coefficients of the cepstral coefficients are employed as features. They can be directly derived from AR as follows:

$$\begin{aligned} c_1 &= -a1 \\ c_i &= -a_i - \sum_{n=1}^{i-1} \left(1 - \frac{n}{i}\right) a_n c_{i-n}, \qquad with \quad 1 < i \leq P \end{aligned} \tag{4.14}$$

Time-frequency Descriptors

1. **Continuous Wavelet Transform (CWT)** – The Gabor limit states that a high resolution both in the frequency and time domain cannot be achieved. Thus, for the STFT, choosing a wider window yields better frequency resolution to the detriment of time resolution for all frequencies and vice versa. Depending on the frequency, the relevance of the different signal's attributes changes. Low frequency signals must be precisely located in the frequency range, as signals a few Hz apart can have dramatically different origins (e.g. Theta brain waves (4 to 8 Hz) and Alpha brain waves (8 to 13 Hz)). On the other hand, for high frequency signals, the relative difference between a few or hundreds Hz is often irrelevant compared to its resolution in time for the characterization of a phenomenon. This behavior can be obtained by

employing wavelets. A wavelet is a signal with a limited duration, varying frequency and a mean of zero [38]. The mother wavelet is an arbitrarily defined wavelet that is utilized to generate different wavelets. The idea behind the wavelets transform is to analyze a signal at different scales of the mother wavelet [39]. For this, a set of wavelet functions are generated from the mother wavelet (by applying different scaling and shifting on the time-axis). The CWT is then computed by calculating the convolution between the input signal and the generated wavelets.

2. **Short Term Fourier Transform based Spectogram (Spectogram)** – The Fourier transform allows for a frequency-based analysis of the signal as opposed to a time-based analysis. However, by its nature, this technique cannot detect if a signal is non-stationary. As sEMG are non-stationary [40], an analysis of these signals employing the Fourier transform is of limited use. An intuitive technique to address this problem is the STFT, which consists of separating the signal into smaller segments by applying a sliding window where the Fourier transform is computed for each segment. In this context, a window is a function utilized to reduce frequency leakage and delimits the segment's width (i.e. zero-valued outside of the specified segment). The spectrogram is calculated by computing the squared magnitude of the STFT of the signal. In other words, given a signal s(t) and a window of width w, the spectrogram is then:

$$Spectogram(s(t), w) = \mid STFT(s(t), w) \mid^2 \tag{4.15}$$

Since the purpose of this work is to design a fast and efficient algorithm for gesture recognition, the features used for classification must be simple and cost-effective to compute. Time-frequency analysis methods involving the Short-Time Fourier Transform, the wavelet transform or the wavelet packet transform would require a larger computation time, with no improvement over the classification performance. For these reasons, only time-domain descriptors were considered. The features used in this work are: *Mean absolute value (MAV)* (5.1), *Zero Crossing Rate (ZC)* (4.2), *Slope Sign Changes (SSC)* (4.3), *Waveform Length (WL)* (4.4), *Skewness* (4.5), *Root Mean Square (RMS)* (4.6), *Hjorth Activity* (4.7), *Integrated EMG* (4.10).

4.2.4 Classification

Recently, the use of machine learning algorithms has become more prominent, as they are being employed in various tasks, ranging from simple regressions up to complex multinomial classification. In the field of EMG-based gesture recognition Neural Networks can be successfully used for classification, if the data set contains sufficient examples. Deep Network-based architectures can learn very complex patterns, but they are prone to overfitting. However, such architectures may be time-consuming, hence, not adequate for real time applications. This thesis proposes a fully connected architecture with a forward pass of less than 4.5 ms, including the feature extraction stage. The model parameters were determined using the cross entropy loss. Considering l targets, the cross entropy loss for a single example is given by the sum:

$$E = -\sum_{i=1}^{l} \Big(t_i \log_2(y_i) + (1 - t_i) \log_2(1 - y_i) \Big) \tag{4.16}$$

where $t_1, t_2, ..., t_l$ are the targets and $y_1, y_2, ..., y_l$ are the outputs of the neural network architecture.

The model parameters are determined via backpropagation, using the ADAM optimizer instead of the classical Stochastic Gradient Descent (SGD). The reason for using the ADAM optimizer is its robustness to changes in hyperparameters [41].

The entire architecture diagram is displayed in Fig. 4.7. The proposed network is composed of 6 hidden layers, having 128, 128,128, 64, 32 and 16 neurons each. After each layer, a *batch normalization* step is performed to avoid overfitting. The activation used for all the layers is ReLu, except for the output layer which uses *Softmax activation*.

4.2.5 Experiments

This thesis proposes a simple architecture, consisting of only Fully Connected Layers. The experimental part carried out for this work can be divided in two: training the Neural Network without constrains and training the system under spectral norms constraints.

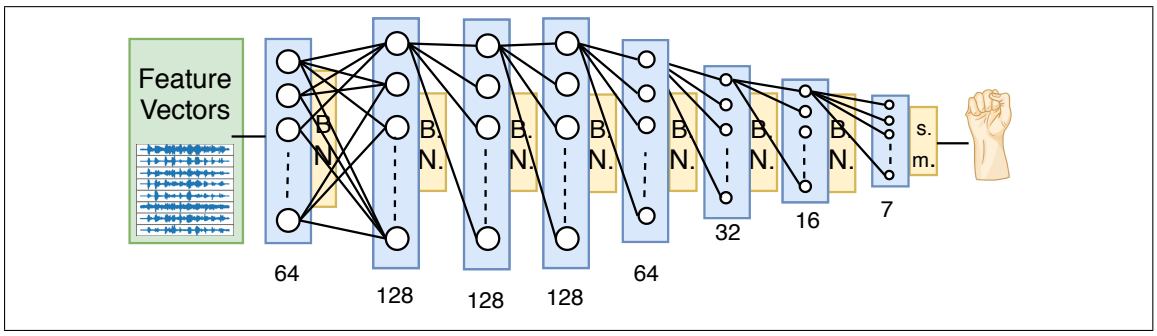


Figure 4.7: Proposed NN architecture

In the first part, the network was trained without imposing any kind of constraints, in an effort to find the best architecture for the task at hand. Several experiments were performed, varying the depth of the network from 4 to 7 hidden layers, but in all cases the number of neurons was set to be higher than the number of neurons in the next layer. The architecture consisting of 6 hidden layers achieved the highest performance, whereas the number of neurons considered for each layer are: 128, 128, 128, 64, 32, 16.

After the optimal design for the architecture was found, the next step was to train the same NN subjected to spectral norm constraints on the weight matrices of each layer, as described in Chapter 3. This was done in order to assess the effect of training under such circumstances on the overall performance of the system. First constraint imposed was $P_{C1 \cap D}$, (3.8, 3.7). The system was trained for several values of the parameter α.

Finally, the algorithm described in 3.28 was implemented in order to accurately compute the projection $P_{C_3 \cap D}$. The same Dual Forward-backward algorithm was used to compute the true approximation of the projection, $P_{\widehat{C_1 \cap D}}$, by imposing $A = B = Id$. The results obtained will be shown and discussed in the next Chapter.

Chapter 5

Results

5.1 Performance metrics

In order to measure the performance of the proposed classification system, the performance metrics used are: the overall classification accuracy (OA), the per-class accuracies (PC) and Kappa index ($\mathcal{K}$). These measures are computed starting from the confusion matrix C which has the number of predicted labels on the columns and the ground truth on the rows:

$$OA = \sum_{i=1}^{l} \frac{C_{ii}}{N} \tag{5.1}$$

$$PC_i = \frac{C_{ii}}{C_{i+}}, \quad \forall i \in \{1, ..., l\} \tag{5.2}$$

$$\mathcal{K} = \frac{\frac{1}{N}\sum_{i=1}^{l} C_{ii} - \frac{1}{N^2}\sum_{i=1}^{l} C_{i+}C_{+i}}{1 - \frac{1}{N^2}\sum_{i=1}^{l} C_{i+}C_{+i}} \tag{5.3}$$

where N represents the total number of analyzed EMG signals, l is the number of gesture categories, C_{ij} is the number of signals in ground truth class j and classified as class i and the values C_{i+} and C_{+j} are computed as:

$$C_{i+} = \sum_{j=1}^{l} C_{ij} \tag{5.4}$$

$$C_{+j} = \sum_{i=1}^{l} C_{ij} \tag{5.5}$$

5.2 Results – training without constraints

The proposed method is compared to other state-of-the-art methods, such as the transfer learning-based method presented in [3] (abbreviated DL-TL). The results are synthesized in Table 5.1.

Table 5.1: Per-class accuracy rates (PC), overall accuracy (OA) and Kappa index ($\mathcal{K}$) of proposed method for hand gesture recognition.

Class	Proposed method	DL-TL [3]
Neutral	99.96	98.89
Radial Deviation	99.75	99.46
Wrist Flexion	99.86	98.42
Ulnar Deviation	99.72	96.52
Wrist Extension	99.72	99.55
Hand Close	99.86	99.43
Hand Open	99.58	94.61
OA	**99.78**	98.12
$\mathcal{K}$	**0.99**	0.98

Table 5.2: Average running time per gesture (decomposed on feature extraction and prediction stages).

Method	Feature extraction	Prediction	Total
Proposed	**1.9 ms**	**2.5 ms**	**4.4 ms**
DL-TL [3]	50.2 ms	19.4 ms	69.9 ms

Compared to the solution proposed in [3], the overall accuracy achieved by the proposed recognition system (99.78%) is higher. Moreover, for all 7 hand gestures, the reported per-class accuracies are greater than the ones obtained by the DL-TL method [3].

An important aspect for real-time applications is the average running time, which, in the case of the proposed recognition technique, is 4.4 ms. Compared to the DL-TL method, this represents a speedup of 16 times. However, apart from

the feature extraction and prediction stages, the most time-consuming part of the DL-TL method is the transfer learning stage which requires almost 5.25 minutes. This represents a drawback for real-time applications in the context of sEMG-based gesture recognition, since the convolutional network (ConvNet) scheme presented in [3] relies upon transfer learning techniques. These techniques leverage inter-user data and increase the overall accuracy by pre-training a model on multiple subjects before training it on a new participant, but come at the cost of spending additional time for learning a specific mapping.

The average running times for both methods, decomposed on feature extraction and prediction stages, are shown in Table 5.2. All the experiments were carried on an NVIDIA Quadro M4000 GPU.

5.3 Results – training with constraints

Without imposing any constraints, the approximate Lipschitz constant of the network is

$$\theta_T = 1.5930763e09$$

. This proves that despite very good theoretical results, the robustness of the system against potential adversarial inputs is not controlled.

To overcome this, the second part of this thesis is concentrated on training the neural network subjected to spectral norm constraints on the weights. This is done by projecting the weights in the optimization step onto the non-empty space of intersection of two convex sets that will satisfy the conditions of non-negativity and impose some restrictions on the spectral norm operator of the weights of each layer.

In order to do this, first an approximate projection C_1 described in Equation (3.8) was considered: first the projection on the spectral ball $\mathcal{B}(0, \alpha)$ was computed for each weight matrix individually, and than the result was again projected with respect to the positivity constraint.

Figure 5.1 shows how the accuracy of the system, computed using Equation 5.1 varies with respect to the value of α. As it was expected, imposing strict constraints has an important effect on the overall performance. Accuracy drops from 99.78% to 71.43% is the spectral norm of each weight matrix is less than 1. The results improve as the constraints loosen (e.g. as α increases). But, as the

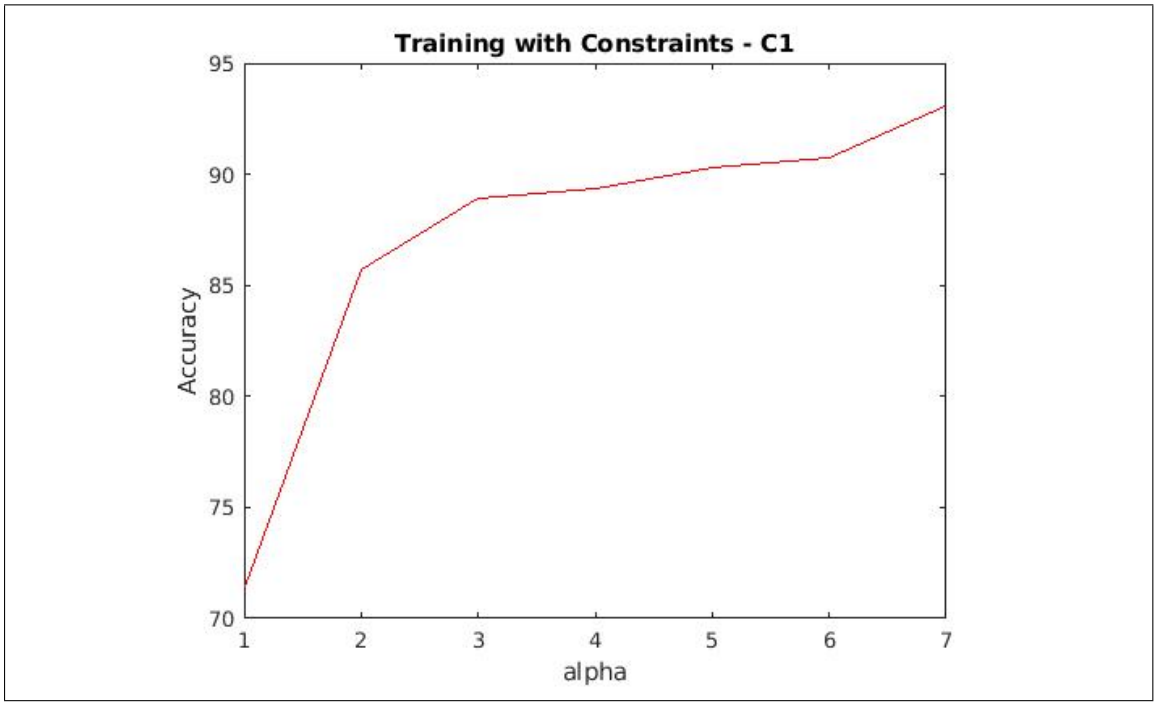


Figure 5.1: Results - $P_{C_1 \cap D}$

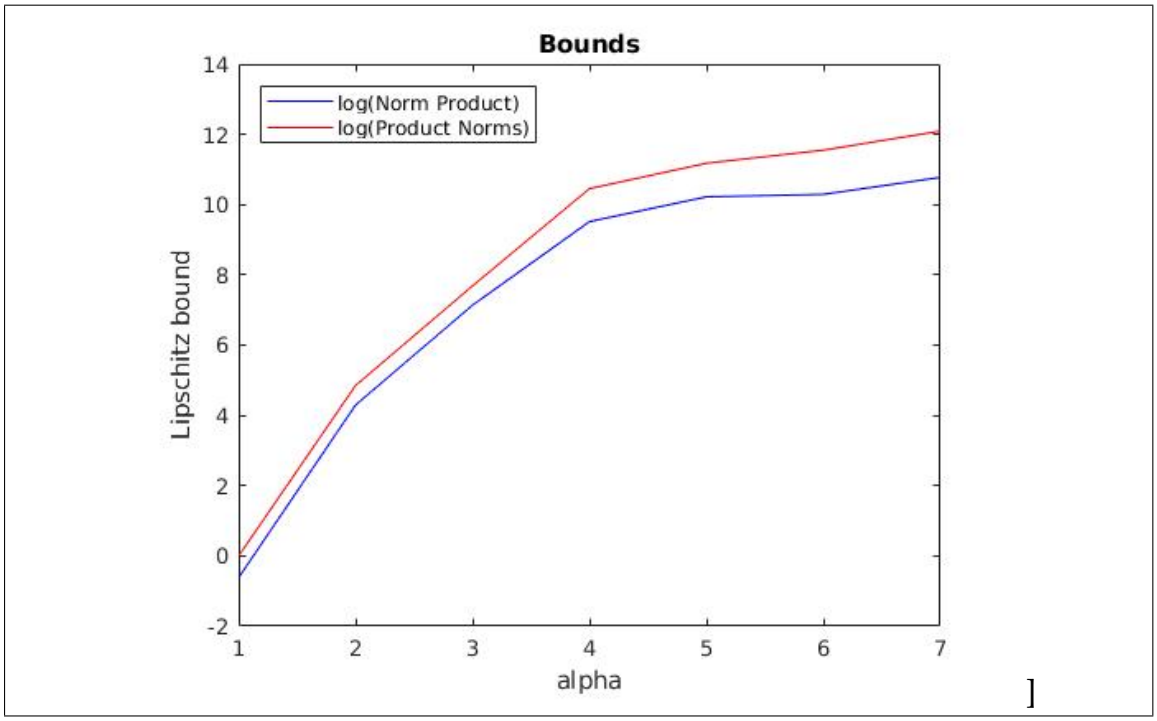


Figure 5.2: Lipschitz Constant Bounds - C_1 constraint

performance improve, the Lipschitz of the network (θ_T) increases exponentially.

Figure 5.2 depicts how the Lipschitz constant of the network (θ_T) varies with respect to α. Lipschitz constant was computed using the rough approximation described in Equation 3.5 and also the lower bound provided by Equation 3.6. This method won't affect the speed of the system, since it relatively easy from the computational point of view, but is not a very good solution to the robustness problem aforementioned.

Note that as the performance increases, θ_T grows up exponentially, with an order of 7 (6 hidden layers, and the output). Therefore, for $\alpha = 7$, even though the overall accuracy $OA = 93.039\%$, the Lipschitz constant of the network has increased to

$$\theta_{T_{C_1 \cap D}} = 4.779224e04$$

This is an important improvement from the unconstrained training, but still a good robustness is not ensured.

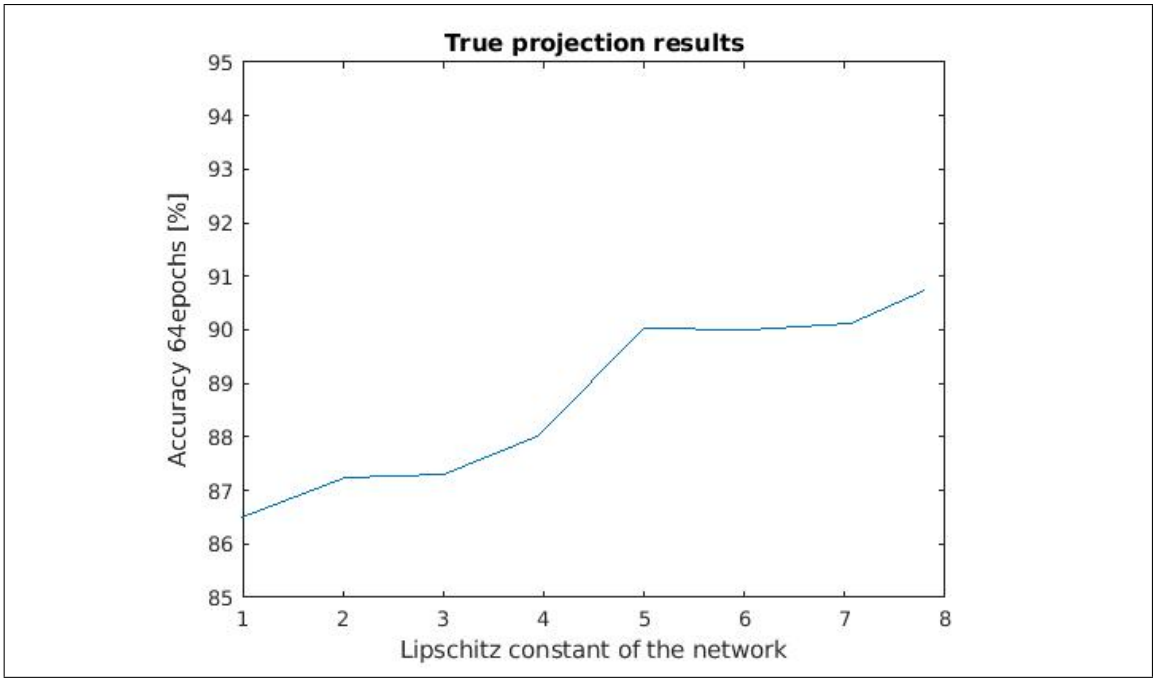


Figure 5.3: Results - $P_{C_3 \cap D}$

In an effort to ensure a small Lipschitz constant, but not affect too much the performance of the system, another constraint was considered, Equation 3.6. The projection $C_3 \cap D$ was computed accurately using the *DFB* algorithm described in 3.28. This is a far lighter constraint, since the spectral norm limitation is imposed

on the product of all weights of the network, offering the system more freedom of learning its parameters.

Additionally, note that in this context the parameter α is in fact the Lipschitz constant of the system ($\theta_T = \alpha$), so the linear variation of α is no longer producing an exponential response from θ_T. The results can be viewed in Figure 5.3, which depicts how overall accuracy OA varies with respect to θ_T. Note that the results have improved with more than 10%. This improvement comes with a great computational cost, since the iterative algorithm must be applied to all the weights of the network. For a $[128 \times 128]$ matrix up to ≈ 10.000 iterations are required for the algorithm to converge with a tolerance of 0.01. In an effort to optimize the learning process, warm restart technique was employed, which consisted on transferring the variable Y_0 of the DFB algorithm, from one epoch to another. By doing this, the computational time is reduced after after the first 3 epochs. This last implementation ensures a great trade-off between the overall performance of the system and its robustness against adversarial attacks, but comes at the expanse of a high computational cost.

Chapter 6

Conclusions

6.1 General Conclusions

In the past decades, researchers had been focused on creating human-computer interfaces. They managed to create hand gesture interfaces by sensors which measure the activities of the musculature system (sEMG). Previous studies have focused on forearm movements controlled by upper arm muscles, which is setting limitations on what kind of hand motions can be classified. Research results have demonstrated that not only finger,but also joint motions can be distinguished from each other successfully by using sEMG signals. This thesis presents a novel method for hand gesture recognition based on simple and easy to compute time-domain features. Using a relative easy to train neural network architecture, the proposed system is able to accurately recognize 7 basic hand gestures in a timely manner, being a candidate for online recognition of rapidly varying hand gestures.

The signals were acquired using an armband equipped with 8 sEMG sensors displayed circularly around the forearm. The raw signal is then process and a vector of 64 relevant features is extracted from each window of 250 ms. These vectors were used as input data to train a fully connected 6-hidden layers neural network, designed to classify 7 basic gestures: ulnar deviation, radial deviation, hand open, hand close, wrist extension, wrist flexion and neutral position. The system performs very well, obtaining an overall accuracy of 99.78%, competing with other state-of-the-art methods presented in literature. A comparison with another method is provided in Chapter 5.

The aim of this work is the development of a real-life application that can be used in a wide variety of fields, ranging from medical health domain to military applications, as mentioned in Chapter 1. To make sure that the proposed system is reliable, the stability of the NN was evaluated. Using recent mathematical findings, novel solutions were proposed to ensure a robust system.

Starting from the premises detailed in Chapter 3, that activation functions used in machine learning algorithms are proximal operators, a strategy of training a NN under spectral norm constraints was designed in order to control the robustness of the system against possible malicious inputs. Several constraints described in Chapter 3 were proposed and implemented (see Chapter 4 for details) in an effort to find the best trade-off between the robustness of the NN and the overall performance of the system. All results were detailed in Chapter 5.

6.2 Personal Contributions

The contributions of the author are the following:

- Implementing a wide list of descriptors in both time and frequency to characterize the sEMG signal.
- Finding the the most relevant and fast features that will serve as an input of the learning system.
- Designing a fast feature extraction block, with a running time of 1.9 ms.
- Designing and implementing a 6-hidden layers Fully Connected Neural Network capable of classifying with a high accuracy (99,78 %) 7 basic gestures in real time.
- Proposing several spectral norm constraints to be applied on the weights of the network to ensure robustness
- Training NN under norm constraints, finding the the best trade-off between performance, speed and robustness
- Implementing a novel DFB algorithm to accurately compute the intersection of two convex sets in order to find the true projection of the weights.

6.3 Future Work

This work has proven that learning methods can be very successful in classification tasks compared to traditional approaches in the area. However, deep NN pose problems of implementation heaviness during the learning phase. Moreover, as shown in this thesis the appear as black boxes whose robustness is poorly controlled.

To overcome these difficulties, lately new architectures were proposed inspired by iterative algorithms used in data analysis or inverse problems, generally obtained by unfolding an optimization algorithm. recent mathematical results show that it becomes easier to control the stability of these networks by introducing appropriate constraints on their weights. This nevertheless requires the management of constraints that are not necessary convex in the training phase. Hence, an important part of the future work to be done consists in implementing and testing with more constraints, to ensure a better evaluation of the performance of the system as a function of robustness.

Another objective is to propose new architectures based on this philosophy. It is necessary to develop efficient optimization methods for NN during supervised learning and to consider different structures of NN, given different classes of iterative methods existing (proximal methods, in particular).

Applications of these theoretical background can be applied in the field of biometrics, for classification of not only EMG signals but also extended towards more complex ones as EEG.

Bibliography

[1] N. Parikh, S. Boyd *et al.*, "Proximal algorithms," *Foundations and Trends® in Optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[2] T. labs, "Myo documentation."

[3] U. Côté-Allard, C. L. Fall, A. Drouin, A. Campeau-Lecours, C. Gosselin, K. Glette, F. Laviolette, and B. Gosselin, "Deep learning for electromyographic hand gesture signal classification using transfer learning," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 2019.

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

[5] T. S. Saponas, D. S. Tan, D. Morris, and R. Balakrishnan, "Demonstrating the feasibility of using forearm electromyography for muscle-computer interfaces," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2008, pp. 515–524.

[6] T. Shanableh, K. Assaleh, and M. Al-Rousan, "Spatio-temporal feature-extraction techniques for isolated gesture recognition in arabic sign language," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 3, pp. 641–650, 2007.

[7] G. Fang, W. Gao, and D. Zhao, "Large vocabulary sign language recognition based on fuzzy decision trees," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 34, no. 3, pp. 305–314, 2004.

[8] T. Starner, J. Weaver, and A. Pentland, "Real-time american sign language recognition using desk and wearable computer based video," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 20, no. 12, pp. 1371–1375, 1998.

[9] K. Assaleh, T. Shanableh, M. Fanaswala, H. Bajaj, and F. Amin, "Vision-based system for continuous arabic sign language recognition in user dependent mode," in *2008 5th International Symposium on Mechatronics and Its Applications*. IEEE, 2008, pp. 1–5.

[10] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," 2016.

[11] C. Anil, J. Lucas, and R. Grosse, "Sorting out lipschitz function approximation," 2018.

[12] P. L. Combettes and J.-C. Pesquet, "Lipschitz certificates for neural network structures driven by averaged activation operators," *arXiv preprint arXiv:1903.01014*, 2019.

[13] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, and N. Usunier, "Parseval networks: Improving robustness to adversarial examples," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 854–863.

[14] J. Kim, S. Mastnik, and E. André, "Emg-based hand gesture recognition for realtime biosignal interfacing," in *Proceedings of the 13th international conference on Intelligent user interfaces*. ACM, 2008, pp. 30–39.

[15] J. Pauk, "Different techniques for emg signal processing," *Journal of Vibroengineering*, vol. 10, pp. 571–576, 12 2008.

[16] N. Womack, N. Williams, J. Holmfield, J. Morrison, and K. Simpkins, "New method for the dynamic assessment of anorectal function in constipation," *British Journal of Surgery*, vol. 72, no. 12, pp. 994–998, 1985.

[17] J. R. Cram, *Introduction to surface electromyography*. Aspen Publishers, 1998.

[18] C. Sapsanis, G. Georgoulas, and A. Tzes, "Emg based classification of basic hand movements based on time-frequency features," in *21st Mediterranean Conference on Control and Automation*. IEEE, 2013, pp. 716–722.

[19] A. D. Orjuela-Cañón, A. F. Ruíz-Olaya, and L. Forero, "Deep neural network for emg signal classification of wrist position: Preliminary results," in *2017 IEEE Latin American Conference on Computational Intelligence (LA-CCI)*. IEEE, 2017, pp. 1–5.

[20] H. Drucker and Y. LeCun, "Improving generalization performance using double backpropagation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 3, no. 6, pp. 991–997, 11 1992.

[21] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved training of wasserstein gans," 2017.

[22] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," 2018.

[23] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[24] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[25] P. L. Combettes and J.-C. Pesquet, "Deep neural network structures solving variational inequalities," *arXiv preprint arXiv:1808.07526*, 2018.

[26] D. E. Rumelhart, G. E. Hinton, R. J. Williams *et al.*, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[27] A. Virmaux and K. Scaman, "Lipschitz regularity of deep neural networks: analysis and efficient estimation," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018, pp. 3835–3844.

[28] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural networks for perception*. Elsevier, 1992, pp. 65–93.

[29] P. L. Combettes and J.-C. Pesquet, "Proximal splitting methods in signal processing," in *Fixed-point algorithms for inverse problems in science and engineering*. Springer, 2011, pp. 185–212.

[30] A. Phinyomark, F. Quaine, S. Charbonnier, C. Serviere, F. Tarpin-Bernard, and Y. Laurillau, "Emg feature evaluation for improving myoelectric pattern recognition robustness," *Expert Systems with applications*, vol. 40, no. 12, pp. 4832–4840, 2013.

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

[32] L. Hargrove, Y. Losier, B. Lock, K. Englehart, and B. Hudgins, "A real-time pattern recognition based myoelectric control usability study implemented in a virtual environment," in *2007 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 2007, pp. 4842–4845.

[33] M. Atzori, A. Gijsberts, C. Castellini, B. Caputo, A.-G. M. Hager, S. Elsig, G. Giatsidis, F. Bassetto, and H. Müller, "Electromyography data for non-invasive naturally-controlled robotic hand prostheses," *Scientific data*, vol. 1, p. 140053, 2014.

[34] A. Phinyomark, S. Hirunviriya, C. Limsakul, and P. Phukpattaranont, "Evaluation of emg feature extraction for hand movement recognition based on euclidean distance and standard deviation," in *ECTI-CON2010: The 2010 ECTI International Confernce on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*. IEEE, 2010, pp. 856–860.

[35] M. Zardoshti-Kermani, B. C. Wheeler, K. Badie, and R. M. Hashemi, "Emg feature evaluation for movement control of upper extremity prostheses," *IEEE Transactions on Rehabilitation Engineering*, vol. 3, no. 4, pp. 324–333, 1995.

[36] M.-F. Lucas, A. Gaufriau, S. Pascual, C. Doncarli, and D. Farina, "Multi-channel surface emg classification using support vector machines and signal-based wavelet optimization," *Biomedical Signal Processing and Control*, vol. 3, no. 2, pp. 169–174, 2008.

[37] R. N. Khushaba, "Correlation analysis of electromyogram signals for multiuser myoelectric interfaces," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 22, no. 4, pp. 745–755, 2014.

[38] M. Teplan *et al.*, "Fundamentals of eeg measurement," *Measurement science review*, vol. 2, no. 2, pp. 1–11, 2002.

[39] A. Graps, "An introduction to wavelets," *IEEE computational science and engineering*, vol. 2, no. 2, pp. 50–61, 1995.

[40] R. N. Khushaba, S. Kodagoda, M. Takruri, and G. Dissanayake, "Toward improved control of prosthetic fingers using surface electrosashmyogram (emg) signals," *Expert Systems with Applications*, vol. 39, no. 12, pp. 10 731–10 738, 2012.

[41] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.