EMG-Based Automatic Gesture Recognition Using Lipschitz-Regularized Neural Networks

Jean-Christophe Pesquet, Corneliu Burileanu, Ana-Antonia Neacșu

EMG-Based Automatic Gesture Recognition Using Robust Neural Networks (Extended version)

Ana Neacșu[a,b], Jean-Christophe Pesquet[a], Corneliu Burileanu[b]

[a] *Universite Paris-Saclay, CentraleSupelec, Inria, Centre de Vision Numerique, 3 Rue Joliot Curie, Gif-sur-Yvette, 91190, France*
[b] *Speech and Dialogue Laboratory, University Politehnica of Bucharest, 313 Splaiul Independentei, Bucharest, 060042, Romania*

Abstract

This paper introduces a novel approach for building a robust Automatic Gesture Recognition system based on Surface Electromyographic (sEMG) signals, acquired at the forearm level. Our main contribution is to propose new constrained learning strategies that ensure robustness against adversarial perturbations by controlling the Lipschitz constant of the classifier. We focus on positive neural networks for which accurate Lipschitz bounds can be derived, and we propose different spectral norm constraints offering robustness guarantees from a theoretical viewpoint. Experimental results on two distinct datasets highlight that a good trade-off in terms of accuracy and performance is achieved. We then demonstrate the robustness of our models, compared to standard trained classifiers in three scenarios, considering both white-box and black-box attacks.

Keywords: Recognition, Machine Learning, Perturbations, Stability, Lipschitz regularity, Optimization, EMG

1. Introduction

In recent years, the concept of human-computer interaction (HCI) has been at the core of many scientific and sociological developments. Combined with the power of machine learning algorithms, it has led to some of the most outstanding achievements in nowadays technology which are used successfully in an ever-increasing number of areas impacting our lives, e.g. medicine [1],

autonomous driving [2], natural language processing [3], etc. Researchers all around the world focus on providing new intuitive and accurate ways of interacting with devices around, based on gesture, voice, or vision analysis [4]. Gestures constitute a universal and intuitive way of communication, with the potential of bringing the Internet of Things (IoT) experience to a different, more organic level [5]. Automatic gesture recognition (AGR) algorithms can be successfully used in various applications, from sign language recognition (SLR) [6] to VR games [7].

Various solutions for AGRs based on image or video stream analysis, leveraging on computer vision algorithms have been proposed; see for example [8, 9, 10]. A multi-stream solution for dynamic hand-gesture recognition is described in [11]. Multi-modal approaches for gesture classification have been also studied [12]. A novel method showing a fully neurmorphic implementation [13] achieves good results (96% accuracy while reducing the inference time by 30%). Although a good performance is achieved on synthetic data, in real-life scenarios these systems may be sensitive to environmental conditions, e.g. light conditions, background, etc. Additionally, these systems are often computationally demanding and consequently not always suited for real-time applications. Accelerometers and electromyography (EMG) sensors provide an alternative low-cost technology for gesture sensing [14]. *sEMG* stands for surface electromyography and represents the electrical manifestation of the neuromuscular activation related to the contraction of the muscles [15]. In [16] the authors propose a method combining feature selection with ensamble learning, achieving around 78% classification accuracy for 52 gestures. The applications of sEMG-based classification systems are focused on, but

not limited to, assertive devices and rehabilitation or postural control therapy for physically impaired persons [17]. With the continuous development of more versatile signal processing techniques, the applications of EMG signal classification expanded to a wide range of domains including augmented reality, gaming industry, military applications, etc.[18, 19].

Two critical issues need to be addressed when developing AGR algorithms: fast enough inference to ensure real-time feeling for the end-user, and accurate and robust classification to guarantee that the gesture is correctly identified no matter the environmental conditions. Machine learning methods have become ubiquitous tools in a wide range of tasks including AGR, on account of their ability to solve a great variety of problems, from simple regressions to complex multi-modal classification.

However, deep neural networks, which are probably the most powerful methods, may appear as black boxes whose robustness is not always well-controlled. For real-life applications, it is mandatory to guarantee the reliability of such techniques. Nowadays, the main difficulty to overcome consists in developing high-performance systems that are also trustable and safe. An additional challenge is to avoid implementation heaviness during the learning phase.

In [20], the authors showed that slightly altering data inputs that were correctly classified by the network can lead to a wrong classification [21, 22, 23]. This finding was at the origin of the concept of adversarial inputs, which constitute malicious input data that can deceive machine learning models. For example, [22] shows how voice interfaces can be fooled by creating carefully crafted artificial audio inputs of unintelligible voice that are

miss-classified as specific vocal commands by the system. Also, [24] introduces several methods for generating adversarial examples on ImageNet that are so close to the original data that differences are indistinguishable for the human eye.

It must be emphasized that adversarial inputs are not necessarily artificially created with the intention to sabotage the system. As other physiological signals, e.g. EEG or EKG, EMG signals have low frequency components (usually between $10 - 150$Hz), and low amplitudes (≤ 10 mV Peak to Peak). This makes them very sensitive to noise and outside perturbations that can occur innately, under the form of noise stemming from acquisition devices, imperfect sensor contact, etc. Those can seriously flaw the performance of real-life applications based on pre-trained models [25]. An empirical way of training more robust AGR systems is detailed in [26], where a strategy of training using noisy labels is proposed.

As highlighted in [24], the Lipschitz behaviour of the network is tightly correlated with its robustness against adversarial attacks. The Lipschitz constant allows to upper bound the output perturbation knowing the magnitude of the input one, for a given metric [27]. Controlling this constant thus represents a feasible solution to limit the effect of adversarial attacks. Computing the exact Lipschitz constant of a neural network is however a very complex problem, so the main challenge is to find clever ways to approximate this constant effectively.

Recently, several techniques to ensure the Lipschitz stability of neural networks have been explored. For example, [23] proposes a novel weight spectral normalization technique applied to stabilize the training of the discriminator

in Generative Adversarial Networks (GANs). The Lipschitz constant of the network is viewed as a hyper-parameter that can be tuned in the training process of the image generation task. Doing so leads to a model with improved generalization capabilities. In [28] norm-constraint GroupSort based architectures are proposed and it is shown that they can be used as universal Lipschitz function approximators. The authors apply gradient norm preservation to create Lipschitzian networks that offer adversarial robustness guarantees. In [29] the authors introduce Parseval networks, another approach for designing networks which are intrinsically robust to adversarial noise, by imposing the Lipschitz constant of each layer of the system to be less than 1. In [30] a convex optimization framework is introduced to compute tight upper bounds for the Lipschitz constant of Deep Neural Networks (DNNs). They make use of the observation that commonly used activation operators are gradients of convex functions. Semi-definite programming approaches to ensure robustness are also explored in [31].

The main contributions of this paper are:

- To propose a robust real-time Automatic Gesture Recognition system based on sEMG signals. The robustness is ensured by using a novel learning algorithm for training feedforward neural networks.
- To show that a good accuracy-robustness balance can be reached. To do so, we train the system under carefully crafted spectral norm constraints, allowing us to finely control its Lipschitz constant. A tight Lipschitz constant is efficiently estimated by focusing on neural networks with positive weights as in [32].
- To demonstrate the performance of the final architecture in real-life ex-

periments where we show that the proposed robust model outperforms those trained conventionally.

- To analyze how our system behaves when the input is affected by different noise levels, simulating perturbations that may occur in real scenarios.
- To show the validity of our solution by experimenting on two distinct publicly available sEMG gestures datasets.

The rest of the paper is structured as follows. The theoretical background of our work is detailed in Section 2. In Section 3, we present the proposed optimization algorithm and we investigate the way of dealing with the constraints. The application and the results are discussed in Section 4, while Section 5 deals with how our model behaves when facing adversarial data. Finally, Section 6 contains some concluding remarks.

2. Robustness solutions in the context of nonnegative neural networks

2.1. Problem formulation

Any feedforward neural network is obtained by cascading m layers associated with operators $(T_i)_{1\leq i\leq m}$. The neural network can thus be expressed as the following composition of operators:

$$T = T_m \circ \cdots \circ T_1. \tag{1}$$

Each layer $i \in \{1, \ldots, m\}$ has a real-valued vector input x_i of dimension N_{i-1} which is mapped to

$$T_i(x_i) = R_i(W_i x_i + b_i), \tag{2}$$

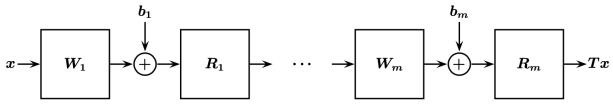


Figure 1: Representation of a NN as a composition of operators

where $W_i \in \mathbb{R}^{N_i \times N_{i-1}}$, $b_i \in \mathbb{R}^{N_i}$ are the weight matrix and bias parameter respectively. $R_i \colon \mathbb{R}^{N_i} \to \mathbb{R}^{N_i}$ constitutes a non-linear activation operator which is applied component-wise (e.g., ReLU or Sigmoid) or globally (e.g, Softmax). Figure 1 shows a graphical representation of this concept.

Even though the choice of the activation R_i may differ depending on the task at hand, it has been shown in [33, 34] that most of them are actually α_i-averaged operators with $\alpha_i \in]0, 1]$. Recall that R_i is an α_i-averaged operator if, for every pair $(x_i, y_i) \in (\mathbb{R}^{N_i})^2$, the following inequality holds:

$$\|R_i(x_i) - R_i(y_i) - (1 - \alpha_i)(x_i - y_i)\| \leq \alpha_i \|x_i - y_i\|. \tag{3}$$

When $\alpha_i = 1/2$, R_i is said to be firmly nonexpansive. For standard choices of activation operators, R_i is firmly nonexpansive since it is the proximity operator of a proper, lower-semicontinous function (see [34] for more details). Note that, in [30], it is assumed that R_i operates component-wise and is slope-bounded. The authors emphasize that the most common case corresponds to lower and upper slope values equal to 0 and 1, respectively. It follows from [35, Proposition 2.4] that a function satisfies this property if and only if it is the proximity operator of some proper lower-semicontinuous convex function, so that similar assumptions to those made in [34] are recovered.

As explained in [33], examples of activation operators R_i which are α_i-averaged with $\alpha_i > 1/2$ can be encountered. They basically correspond

to over-relaxations of firmly nonexpansive operator. An example of such operators is the Swish activation function [36]. Another famous example is the group-sort operator:

$$\left(\forall x_i = \begin{bmatrix} x_{i,1} \\ \vdots \\ x_{i,M} \end{bmatrix} \in \mathbb{R}^{N_i}\right) \qquad R_i(x_i) = \begin{bmatrix} x_{i,1}^{\uparrow} \\ \vdots \\ x_{i,M}^{\uparrow} \end{bmatrix}, \tag{4}$$

where the vector x_i has been decomposed in M subvectors $x_{i,j}$ with $j \in \{1, \ldots, M\}$, of dimension B ($N_i = BM$) and $x_{i,j}^{\uparrow}$ designates the vector of components of $x_{i,j}$ sorted in ascending order. R_i is then purely nonexpansive, i.e. $\alpha_i = 1$. Note that max-pooling can be achieved by composing this group sort operation with a linear operator. Indeed, if $i < m$, $M = N_{i+1}$, and W_{i+1} is the matrix extracted from the $N_i \times N_i$ identity matrix Id_{N_i} by selecting the matrix rows with indices multiple of B, then $W_{i+1} \circ R_i$ corresponds to a max-pooling.

2.2. Lipschitz robustness certificate

Consider a neural network T as described in Fig. 1. let $x \in \mathbb{R}^{N_0}$ be the input of the network and let $T(x) \in \mathbb{R}^{N_m}$ be its associated output. By adding some small perturbation $z \in \mathbb{R}^0$ to the input, the perturbed input is $\tilde{x} = x + z$. The effect of the perturbation on the output of the system can be quantified by the following inequality:

$$\|T(\tilde{x}) - T(x)\| \leq \theta_m \|z\|, \tag{5}$$

where $\theta_m \geq 0$ denotes a Lipschitz constant of the network. θ_m represents thus an important parameter that allows us to assess and control the sensitivity of a neural network to various perturbations. It needs however to be accurately

estimated to provide valuable information. A standard approximation to the Lipschitz constant [24] is given by

$$\theta_m = \prod_{i=1}^{m} \|W_i\|_{\mathrm{S}}, \tag{6}$$

where $\|\cdot\|_{\mathrm{S}}$ denotes the *spectral norm* of a matrix. Although simple to compute, this approximate bound is over-pessimistic. Different methods for obtaining tighter estimates of the Lipschitz constant have been presented in the recent literature; see for example [27, 33, 30, 37, 38]. Local estimates of the Lipschitz constant can also be performed which may appear more relevant. But they are more complex to compute and, as we will see, controlling the global Lipschitz constant is usually sufficient to get a good performance. Estimating the global Lispchitz constant of the network is an NP (non-deterministic polynomial-time)-hard problem [27]. Although there exist efficient approaches to approximate an accurate bound [30, 37, 38], computing these estimates may be expensive for wide or deep networks. In addition, using these bounds within a training procedure is a difficult task [31]. In this work, we will make the following assumption.

Assumption 2.1 Let a neural network be given by (1) where the i-th layer with $i \in \{1, \dots, m\}$ is given by (2). We assume that

(i) all the activation layers, except possibly the last one, consist of separable averaged operators, that is, for every $i \in \{1, \dots, m-1\}$, there exist averaged functions $(\rho_{i,k})_{1 \leq k \leq N_i}$ from $\mathbb{R}$ to $\mathbb{R}$ such that $R_i \colon (\xi_{i,k})_{1 \leq k \leq N_i} \mapsto \big(\rho_{i,k}(\xi_{i,k})\big)_{1 \leq i \leq k}$;

(ii) at the last activation layer, R_m is an averaged operator.

Our approach will be grounded on the following result.

Proposition 2.2 [33] *Suppose that Assumption 2.1 holds. For every* $i \in \{1, \ldots, m\}$*, let* A_i *be the matrix whose elements are the absolute values of those of* W_i*. Then,*

$$\vartheta_m = \|A_m \cdots A_1\|_{\mathrm{S}} \tag{7}$$

is a Lipschitz constant of T*. In addition*

$$\|W_m \cdots W_1\| \leq \vartheta_m. \tag{8}$$

In particular if, for every $i \in \{1, \ldots, m\}$*,* $W_i \in [0, +\infty[^{N_i \times N_{i-1}}$*,* ϑ_m *is equal to the lower bound in* (8)*.*

Based on this proposition, the best estimate for the Lipschitz constant of a given feedforward neural network having nonnegative weights simplifies to the spectral norm of the product of all the weight matrices composing the network. More precisely, the obtained Lipschitz constant

$$\vartheta_m = \|W_m \cdots W_1\|_{\mathrm{S}}$$

is the Lipschitz constant of a purely linear network, where all the non-linear activation operators have been replaced with the identity operator.

The above result is guaranteed to be valid only in the case when all the weights are nonnegative. In the general case of networks with weights having arbitrary signs, it can be proved that $\|W_m \cdots W_1\|_{\mathrm{S}}$ represents only a lower bound of the Lipschitz constant established in [33]. It is also worth mentioning that the proposed results hold for any algebraic structure of the weight matrices $(W_i)_{1 \leq i \leq m}$.

3. Optimization methods for training robust feedforward networks

3.1. Stochastic gradient descent – projected variant

Standard training in neural networks consists in the minimization of a nonconvex cost function with respect to the model parameters by means of an iterative strategy. Let $\mathcal{L}$ be the cost function defined as follows:

$$\mathcal{L}(\eta) = \sum_{k=1}^{K} \ell(z_k, \eta), \tag{9}$$

where $\eta = (\eta_i)_{1 \leq i \leq m}$ is a vector encompassing all the model parameters. For each layer $i \in \{1, \ldots, m\}$, η_i denotes a vector of dimension $N_i(N_{i-1}+1)$ that contains the scalar variables associated with the weight matrices W_i and the corresponding bias components b_i. The data information is represented by $(z_k)_{1 \leq k \leq K}$. For every $k \in \{1, \ldots, K\}$, z_k is a pair consisting of an input of the system and the associated desired output (ground truth). Also, ℓ represents the loss function assumed to be differentiable (almost everywhere) with respect to η.

To ensure robustness, we shall impose spectral norm constraints on the weight matrices. In other words, the vector of parameters η is constrained to belong to a closed set $\mathcal{S}$ that will be described in the next section. We propose to use an extension of a standard optimization techniques for training neural networks [39]. More specifically, we will implement a *projected stochastic gradient* algorithm. A momentum parameter is introduced in this algorithm to accelerate the convergence process.

Algorithm 1 describes the iterations performed at each epoch $n > 0$. We see that there are two nested loops: the outer loop operates on the batch index q and the second one on the layer index i. In this algorithm,

$\gamma_n \in]0, +\infty[$ is the learning rate, while $\zeta_n \in [0, +\infty[$ denotes the inertia parameter for momentum. The algorithm is very similar to block-iterative techniques used in convex optimization [39]. The parameters of each layer are indeed updated successively by performing a gradient step on the data in the current mini-batch (which can be epoch-dependent). ∇_i represents the gradient, computed by standard backpropagation mechanism, with respect to η_i for each $i \in \{1, \dots, m\}$. This stochastic gradient step is followed by a projection $\mathsf{P}_{\mathcal{S}_{i,n}}$ onto the constraint set $\mathcal{S}_{i,n}$. The definition of this set as well as the way of handling this projection are detailed in the following.

Algorithm 1: Projected SGD Algorithm

Partition $\{1, \dots, K\}$ *into minibatches* $(\mathbb{L}_{q,n})_{1 \leq q \leq Q}$

foreach $q \in \{1, \dots, Q\}$ **do**

  foreach $i \in \{1, \dots, m\}$ **do**

$$\Delta_{i,n} = (1 + \zeta_n)\eta_{i,n} - \zeta_n \eta_{i,n-1} \; \widetilde{\eta}_{i,n} = [(\eta_{j,n+1}^\top)_{j<i} \;\; \Delta_{i,n}^\top \;\; (\eta_{j,n}^\top)_{j>i}]^\top$$

$$\eta_{i,n+1} = \mathsf{P}_{\mathcal{S}_{i,n}}\Big(\Delta_{i,n} - \gamma_n \sum_{k \in \mathbb{L}_{q,n}} \nabla_i \ell(z_k, \widetilde{\eta}_{i,n})\Big)$$

where $\mathcal{S}_{i,n} = \big\{\eta_i \mid [(\eta_{j,n+1}^\top)_{j<i} \;\; \eta_i^\top \;\; (\eta_{j,n}^\top)_{j>i}]^\top \in \mathcal{S}\big\}$.

3.2. Constraint sets

As mentioned before, this work revolves around feed-forward networks with positive weights. Thus, the first condition that we impose is nonnegativity for each layer $i \in \{1, \dots, m\}$, which is modeled by the constraint set

$$\mathcal{D}_i = \{W_i \in \mathbb{R}^{N_i \times N_{i-1}} \mid W_i \geq 0\} \tag{10}$$

Moreover, based on our standing assumptions and Proposition 2.2, we must impose a spectral norm constraint on the weight matrices to control the

robustness of the system. This translates mathematically as the following upper bound constraint:

$$\|W_m \cdots W_1\|_{\mathrm{S}} \leq \overline{\vartheta}, \tag{11}$$

where $\overline{\vartheta}$ represents the target maximum Lipschitz constant of the network. This bound constitutes a direct measure of the system level of robustness against adversarial inputs. We need to handle these two constraints simultaneously during the training process. Imposing nonnegativity is fairly easy since (10) defines a simple convex constraint. By contrast, constraint (11) does not satisfy the convexity property. Since (11) corresponds to a closed set in the underlying space of weight matrices and this set has a nonempty intersection with $\mathcal{D}_1 \times \cdots \times \mathcal{D}_m$, the projection onto the intersection of the two sets can be defined but it is not guaranteed to be unique. To cirmcumvent this difficulty, it can be noticed that (11) actually defines a multi-convex constraint in the sense that if, for every $i \in \{1, \ldots, m\}$, $(W_j)_{1 \leq j \leq m, j \neq i}$ are given, then (11) imposes a convex constraint on W_i. This suggests to introduce the following closed and convex set:

$$\mathcal{C}_{i,n} = \{W_i \in \mathbb{R}^{N_i \times N_{i-1}} \mid \|A_{i,n} W_i B_{i,n}\|_{\mathrm{S}} \leq \overline{\vartheta}\} \tag{12}$$

in order to control the Lipschitz constant. Hereabove, the matrices $A_{i,n}$ and $B_{i,n}$ represent the product of the weight matrices for the previous and the posterior layers, respectively. By adopting the convention that $A_{i,n} = \mathrm{Id}$ if $i = m$ and $B_{i,n} = \mathrm{Id}$ if $i = 1$, we define these matrix products as

$$A_{i,n} = W_{m,n} \cdots W_{i+1,n}, \qquad B_{i,n} = W_{i-1,n+1} \cdots W_{1,n+1}, \tag{13}$$

where $(W_{j,n})_{1 \leq j \leq m}$ denote the estimates of the weight matrices at each iteration n, as it appears in Algorithm 1.

Thus, our objective will be to perform the projection onto the set $\mathcal{S}_{i,n} = \mathcal{D}_i \cap \mathcal{C}_{i,n}$, for each layer $i \in \{1, \ldots, m\}$ and at each iteration n. Several algorithms can be envisaged to solve this convex optimization problem.

Before describing our proposed algorithmic solution, let us recall the expressions of the required elementary projections. For every $W \in \mathbb{R}^{S\times T}$, the projection of W onto $[0, +\infty[^{S\times T}$ is

$$\mathsf{P}_{[0,+\infty[^{S\times T}}(W) = (\widetilde{W}_{s,t})_{1\leq s\leq S, 1\leq t\leq T}, \tag{14}$$

where, for every $s \in \{1, \ldots, S\}$ and $t \in \{1, \ldots, T\}$,

$$\widetilde{W}_{s,t} = \begin{cases} W_{s,t} & \text{if } W_{s,t} \geq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

Let $\mathcal{B}(0, \overline{\vartheta})$ be the closed spectral ball of center 0 and radius $\overline{\vartheta}$ defined as [1]

$$\mathcal{B}(0, \overline{\vartheta}) = \{W \in \mathbb{R}^{S\times T} \mid \|W\|_{\mathrm{S}} \leq \overline{\vartheta}\}. \tag{16}$$

For every $W = (W_{s,t})_{1\leq s\leq S, 1\leq t\leq T} \in \mathbb{R}^{S\times T}$, let $U\Lambda V^\top$ be the singular value decomposition of W, where $U \in \mathbb{R}^{S\times R}$ and $V \in \mathbb{R}^{T\times R}$ are matrices such that $U^\top U = \mathrm{Id}$ and $V^\top V = \mathrm{Id}$, $R = \min\{S, T\}$, and $\Lambda = \mathrm{Diag}(\lambda_1, \ldots, \lambda_R)$, $(\lambda_r)_{1\leq r\leq R} \in [0, +\infty[^R$ being the singular values of W. Then the projection of W onto $\mathcal{B}(0, \overline{\vartheta})$ is expressed as

$$\mathsf{P}_{\mathcal{B}(0,\overline{\vartheta})}(W) = U\widetilde{\Lambda}V^\top \tag{17}$$

where $\widetilde{\Lambda} = \mathrm{Diag}(\widetilde{\lambda}_1, \ldots, \widetilde{\lambda}_r)$ and

$$(\forall i \in \{1, \ldots, r\}) \quad \widetilde{\lambda}_i = \begin{cases} \lambda_i & \text{if } \lambda_i \leq \overline{\vartheta} \\ \overline{\vartheta} & \text{otherwise.} \end{cases} \tag{18}$$

[1] To simplify our notation, $\mathcal{B}(0, \overline{\vartheta})$ will designate any spectral ball of this kind whatever the dimensions of the involved matrices.

To compute the projection onto $\mathcal{S}_{i,n}$ of a matrix $\overline{W}_i \in \mathbb{R}^{N_i \times N_{i-1}}$, we propose to employ the FISTA (Fast Iterative Shrinkage-Thresholding Algorithm) version of a dual forward-backward method in Algorithm 2. This algorithm is based on a dual proximal approach [40] and constitutes an extension of the optimization method originally proposed in [41]. The rationale for this algorithm is given in the appendix.

Algorithm 2: FISTA-like accelerated version of DFB algorithm

Let $Y_0 \in \mathbb{R}^{N_m \times N_0}$

Set $\gamma = 1/(\|A_{i,n}\|_{\mathrm{S}} \|B_{i,n}\|_{\mathrm{S}})^2$

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

for $l = 0, 1, \ldots$ **do**

$\eta_l = \frac{l}{l+1+\alpha}$

$Z_l = Y_l + \eta_l (Y_l - Y_{l-1})$

$V_l = \mathsf{P}_{D_i}(\overline{W}_i - A_{i,n}^\top Z_l B_{i,n}^\top)$

$\widetilde{Y}_l = Z_l + \gamma A_{i,n} V_l B_{i,n}$

$Y_{l+1} = \widetilde{Y}_l - \gamma \mathsf{P}_{\mathcal{B}(0,\overline{\vartheta})}(\gamma^{-1} \widetilde{Y}_l)$

return V_l

3.3. Handling looser constraints

The Lipchitz constant of the network can be controlled in multiple ways. Besides the solution formulated in Section 3.2, a more standard approach to control it [20] consists in imposing

$$\prod_{i=1}^{m} \|W_i\|_{\mathrm{S}} \leq \overline{\vartheta}. \tag{19}$$

Two strategies have been implemented to enforce this constraint.

(i) The first one consists in imposing a uniform bound on the spectral norm of each weight matrix $(W_i)_{1\leq i\leq m}$, which leads to the following convex constraint sets:

$$(\forall i \in \{1,\dots,m\}) \quad \widetilde{\mathcal{C}}_i = \{W_i \in \mathbb{R}^{N_i\times N_{i-1}} \mid \|W_i\|_{\mathrm{S}} \leq \overline{\vartheta}^{1/m}\}. \quad (20)$$

(ii) The second strategy aims at introducing more flexible bounds on the spectral norms of each layer. It is based on the following choice for the individual convex constraint sets:

$$(\forall n \in \mathbb{N}\setminus\{0\})(\forall i \in \{1,\dots,m\})$$
$$\check{\mathcal{C}}_{i,n} = \Big\{W_i \in \mathbb{R}^{N_i\times N_{i-1}} \mid \|W_i\|_{\mathrm{S}} \leq \|W_{i,n}\|_{\mathrm{S}}\Big(\frac{\vartheta}{\prod_{j=1}^m \|W_{j,n}\|_{\mathrm{S}}}\Big)^{1/m}\Big\}.$$

For every $i \in \{1,\dots,m\}$, projecting onto $\widetilde{\mathcal{C}}_i$ or $\check{\mathcal{C}}_{i,n}$ is performed by truncating a singular value decomposition, similarly to the technique described at the end of Section 3.2. The projections onto $\widetilde{\mathcal{C}}_i \cap \mathcal{D}_i$ and $\check{\mathcal{C}}_{i,n} \cap \mathcal{D}_i$ can then be computed by using the same iterative method as in Algorithm 2 with $A_{i,n} = B_{i,n} = \mathrm{Id}$.

In all the proposed constrained optimization methods, the projection $\mathsf{P}_{\mathcal{B}(0,\widetilde{\vartheta})}$ onto a spectral ball with radius $\widetilde{\vartheta} > 0$ plays a prominent role. The ball radius depends on the handled constraint (11), (20), or (10). A complex operation such as a singular value decomposition may be very demanding in terms of computational resources when dealing with large size matrices. In that case, we propose to use an approximate projection [23] defined as

$$(\forall W \in \mathbb{R}^{S\times T}) \quad \mathsf{P}_{\mathcal{B}(0,\widetilde{\vartheta})}(W) \simeq \begin{cases} W & \text{if } \|W\|_{\mathrm{S}} \leq \widetilde{\vartheta} \\ \dfrac{\widetilde{\vartheta}}{\|W\|_{\mathrm{S}}} W & \text{otherwise.} \end{cases} \quad (21)$$

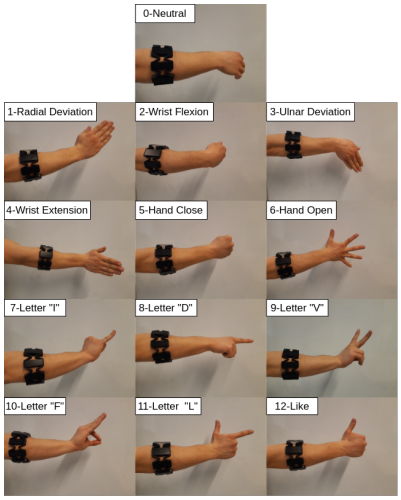


Figure 2: 13-gestures Dataset [42]

Using this approximation in Algorithm 2 yields approximate projections $(\widetilde{\mathsf{P}}_{\mathcal{C}_{i,n}\cap\mathcal{D}_i})_{1\leq i\leq m,n>0}$. Note however that we then lose the theoretical guarantees of convergence Algorithm 2, even if this issue was not observed in our implementation.

An additional advantage of Formula (21) is that it allows the nonnegativity of the elements of the input matrix to be kept. This allows us to derive cheap approximate versions of the projection onto $\widetilde{\mathcal{C}}_i \cap \mathcal{D}_i$ with $i \in \{1,\dots,m\}$ by first projecting onto $\mathcal{D}_i$ and then applying the approximate projection onto $\widetilde{\mathcal{C}}_i$. The resulting approximate projection is denoted by $(\widetilde{\mathsf{P}}_{\widetilde{\mathcal{C}}_i\cap\mathcal{D}_i})_{1\leq i\leq m}$. A similar procedure can be followed to compute approximate projections $(\widetilde{\mathsf{P}}_{\check{\mathcal{C}}_{i,n}\cap\mathcal{D}_i})_{1\leq i\leq m,n>0}$ onto $(\check{\mathcal{C}}_{i,n}\cap\mathcal{D}_i)_{1\leq i\leq m,n>0}$.

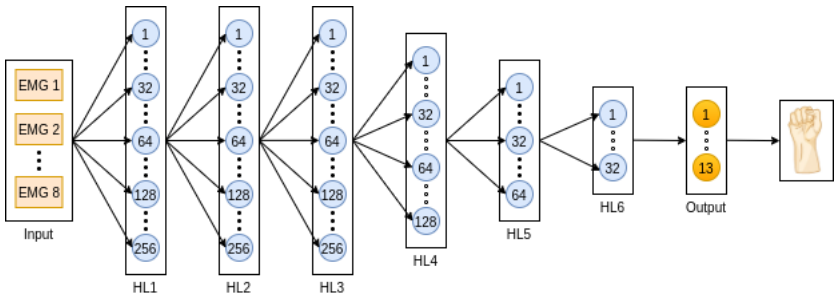


Figure 3: *Proposed neural network architecture for AGR. All the layers except the last one use ReLU activation functions; the last layer uses Softmax. The number of neurons considered for each layer is: 128, 128, 128, 64, 32, 16, in the case of 7-gestures dataset and 256, 256, 256, 128, 64, 32 in the case of 13-gesture dataset. The last layer has 7 or 13 neurons representing the gesture number being recognized. Each EMG box represents a column vector containing 8 time-descriptors.*

4. AGR Experimental Setup

4.1. sEMG datasets

We test our proposed training scheme on three online datasets containing EMG information of different hand gestures. All three were acquired using Myo armband, a device developed by Thalmic Labs, equipped with eight sEMG sensors displayed circularly.

The first dataset, detailed in [43] contains EMG signals characterizing 7 hand gestures correlated to the primary movements of the hand. There are four mobility gestures (i.e., wrist flexion and extension, ulnar, and radial deviation) and two gestures used for grasping and releasing objects (i.e., spread fingers and close fist). The 7^{th} gesture characterizes the neutral position, corresponding to the relaxation of the muscles.

The second dataset includes 13 gestures: the same 7 gestures described above, plus 6 additional classes. It contains gestures from 50 different subjects and two sets of trials per user. All 13 gestures are depicted in Figure 2.

More details about the dataset can be found in [42].

The third dataset is a subset of NinaPro DB5 dataset, detailed in [44]. The dataset is acquired using two Myo armbands, one positioned just below the elbow and the other one closer to the arm. For our experiments, we considered the subset C, which contains sEMG data associated to 24 gestures.

We also validate our models in a real-context scenario. For the real-life predictions, we recorded the EMG activity associated with each gesture at forearm level using Myo armband. The information collected from each channel is transmitted to a computer via Bluetooth protocol where it is processed to extract relevant time domain features that will be used by the classifier to determine which gesture has been performed.

4.2. Proposed Architecture

The raw 8 channels EMG signal is split using a 250 ms sliding window, with 50% overlap. From each window of each channel a series of 8 time descriptors are extracted. The information from all the channels is then concatenated, forming a 64-dimensional vector. The 7-gestures dataset contains around 200k vector samples, the 13-gestures dataset has around 59k vector samples, while the 24-gestures dataset has around 20k vector samples. Those are split in training, validation, and test sets at user level according to the ratio: 70%, 20%, 10%. These vectors are fed to the network in mini-batches of size 2048. The considered architectures consists of a 6-hidden layer ($m = 6$) fully connected neural networks, with different parameters depending on the considered datasets, but the same core structure, as displayed in Figure 3. Let $x = (x_k)_{0 \leq k \leq K-1}$ be the vector of EMG samples acquired on a window from one channel. For this work we considered some of the most relevant

features to describe sEMG data, as follows.

(i) **Mean Absolute Value (MAV)** – the mean of the absolute values of the signal is given by

$$\mathrm{MAV}(x) = \frac{1}{K} \sum_{k=0}^{K-1} |x_k|. \tag{22}$$

(ii) **Zero Crossing Rate (ZCR)** – this feature counts the frequency at which the signal passes through zero. A threshold $\alpha \geq 0$ is used in order to lessen the noise effect. This feature can be computed in an incremental manner and it is defined as

$$\mathrm{ZCR}(x) = \Big| \big\{ k \in \{1, \ldots, K-1\} \mid |x_k - x_{k-1}| \geq \alpha \text{ and } x_k x_{k-1} < 0 \big\} \Big|. \tag{23}$$

(iii) **Waveform Length (WL)** – this feature offers a simple characterization of the signal waveform. It corresponds to the following total variation seminorm:

$$\mathrm{WL}(x) = \sum_{k=1}^{K-1} |x_k - x_{k-1}|. \tag{24}$$

(iv) **Slope Sign Changes (SSC)** – measures the frequency at which the sign of the signal slope changes. It amounts in checking a condition on three consecutive samples x_k, x_{k-1}, x_{k+1} with $k \in \{2, \ldots, K-2\}$:

$$\mathrm{SSC}(x) = \Big| \{ k \in \{2, \ldots, K-2\} \mid (x_k - x_{k-1})(x_k - x_{k+1}) \geq \alpha \} \Big|, \tag{25}$$

where the threshold $\alpha > 0$ is employed to reduce the influence of the noise.

(v) **Root Mean Square (RMS)** – this feature, also related to the quadratic mean or local energy of the signal is given by

$$\mathrm{RMS}(x) = \sqrt{\frac{1}{K}\sum_{k=0}^{K-1} x_k^2}\,. \tag{26}$$

(vi) **Hjorth parameters** – are a set of three features originally developed for characterizing electroencephalography signals and then successfully applied to sEMG signal recognition. The most relevant Hjorth activity parameter can be thought of as the integrated power spectrum and basically corresponds to the variance of the signal calculated as follows:

$$\sigma^2(x) \;=\; \frac{1}{K}\sum_{k=0}^{K-1}(x_k - \;\mu(x))^2, \tag{27}$$

where $\mu(x)$ represents the mean value of the signal. The standard deviation and $\mathrm{RMS}(x)$ are equal when the mean of the signal is zero.

(vii) **Skewness** – measures the overall asymmetry of probability distribution of the data:

$$\mathrm{Skew}(x) = \frac{1}{K}\sum_{k=0}^{K-1}\left(\frac{x_k - \;\mu(x)}{\sigma(x)}\right)^3. \tag{28}$$

(viii) **Integrated Square-root EMG (ISEMG)** – is a feature returning the sum of the fully-rectified signal:

$$\mathrm{ISEMG}(x) = \sum_{k=0}^{K-1}\sqrt{|\;x_k\;|}. \tag{29}$$

4.3. Performance analysis in terms of accuracy and robustness

The performance of our AGR system trained conventionally achieves state-of-art performance [46, 42, 14], of over 99% accuracy for the first two

Method	Dataset	# of gestures	Accuracy [%]	Year
Fusion-Lohi [13]	DVS-EMG	5	96.04	2020
EELM [16]	NinaPro DB5	52	77.90	2022
MResLSTM [11]	NinaPro DB1	52	89.65	2021
GRU-Res [14]	sEMG-IMU	20	99.49	2022
EMG-CNN [45]	15Myo-sEMG	15	98.67	2022
7-DNN (ours)	Myo-sEMG	7	99.67	2020
13-DNN (ours)	13Myo-sEMG	13	99.31	2021
24-DNN (ours)	NinaPro DB5 (Exercise C)	24	86.20	2023

Table 1: Comparison to other State of the Art sEMG-based AGR systems

datasets and around 86% in the case of the 24-gestures dataset [16, 26]. A more detailed comparison with other new sEMG-based AGR systems is presented in Table 1. Since in this case the weights are not guaranteed to be positive, the lower bound introduced in Proposition 2.2 does not constitute a valid Lipschitz constant. Computing the exact Lipschitz constant θ_m of the system is a very difficult task [33], but we can easily bound θ_m between the estimate given by (6) and the spectral norm of the product of all the weight matrices from the network. We found that the Lipschitz constant estimate $\theta_m \in [1.56\times10^{12}, 1.59\times10^{14}]$ for all three datasets. This suggests that despite the high performance of the classifiers, their robustness is poorly controlled, leaving the systems vulnerable to adversarial perturbations.

A first step towards controlling the Lipschitz constant of the classification algorithm and implicitly its robustness is to impose the nonnegativity condition associated with constraint $\mathcal{D}$. Training under such a nonnegativity constraint is shown to improve the network operation interpretability [32] and acts as a regularization, reducing overfitting. On the other hand, it

		Accuracy	75 %	80 %	85 %	90 %	95%
Lipschitz constant 7-gestures	$\widetilde{\mathcal{C}}_i \cap \mathcal{D}_i$	$\widetilde{\mathsf{P}}_{\widetilde{\mathcal{C}}_i \cap \mathcal{D}_i}$	19.5	37.5	68.3	3.5×10^4	3.5×10^8
		$\mathsf{P}_{\widetilde{\mathcal{C}}_i \cap \mathcal{D}_i}$	0.66	13.47	74.16	1.04×10^3	1.39×10^5
	$\check{\mathcal{C}}_{i,n} \cap \mathcal{D}_i$	$\widetilde{\mathsf{P}}_{\check{\mathcal{C}}_{i,n} \cap \mathcal{D}_i}$	0.71	1.84	3.42	6.87	11.60
		$\mathsf{P}_{\check{\mathcal{C}}_i \cap \mathcal{D}_i}$	0.70	1.35	3.41	6.79	11.20
	$\mathcal{C}_{i,n} \cap \mathcal{D}_i$	$\widetilde{\mathsf{P}}_{\mathcal{C}_{i,n} \cap \mathcal{D}_i}$	0.44	1.79	2.93	4.85	5.68
		$\mathsf{P}_{\mathcal{C}_{i,n} \cap \mathcal{D}_i}$	0.35	0.46	0.65	0.82	0.95
Lipschitz constant 13 gestures	$\widetilde{\mathcal{C}}_i \cap \mathcal{D}_i$	$\widetilde{\mathsf{P}}_{\widetilde{\mathcal{C}}_i \cap \mathcal{D}_i}$	20.2	41.8	145.2	2.2×10^5	1.21×10^{11}
		$\mathsf{P}_{\widetilde{\mathcal{C}}_i \cap \mathcal{D}_i}$	0.85	20.47	112.3	1.62×10^4	2.31×10^8
	$\check{\mathcal{C}}_{i,n} \cap \mathcal{D}_i$	$\widetilde{\mathsf{P}}_{\check{\mathcal{C}}_{i,n} \cap \mathcal{D}_i}$	0.84	2.08	4.23	7.54	12.02
		$\mathsf{P}_{\check{\mathcal{C}}_i \cap \mathcal{D}_i}$	0.81	2.01	4.12	7.50	11.92
	$\mathcal{C}_{i,n} \cap \mathcal{D}_i$	$\widetilde{\mathsf{P}}_{\mathcal{C}_{i,n} \cap \mathcal{D}_i}$	0.54	1.87	3.38	4.20	5.78
		$\mathsf{P}_{\mathcal{C}_{i,n} \cap \mathcal{D}_i}$	0.49	0.53	0.75	0.92	1.25
		Accuracy	65 %	70 %	75 %	80 %	85%
Lipschitz constant 24-gestures	$\widetilde{\mathcal{C}}_i \cap \mathcal{D}_i$	$\widetilde{\mathsf{P}}_{\widetilde{\mathcal{C}}_i \cap \mathcal{D}_i}$	25.13	57.16	188.26	2.5×10^6	2.14×10^{11}
		$\mathsf{P}_{\widetilde{\mathcal{C}}_i \cap \mathcal{D}_i}$	1.85	31.12	112.3	1.82×10^4	4.63×10^8
	$\check{\mathcal{C}}_{i,n} \cap \mathcal{D}_i$	$\widetilde{\mathsf{P}}_{\check{\mathcal{C}}_{i,n} \cap \mathcal{D}_i}$	1.74	2.41	6.02	10.17	20.14
		$\mathsf{P}_{\check{\mathcal{C}}_i \cap \mathcal{D}_i}$	1.57	2.18	5.94	10.58	19.69
	$\mathcal{C}_{i,n} \cap \mathcal{D}_i$	$\widetilde{\mathsf{P}}_{\mathcal{C}_{i,n} \cap \mathcal{D}_i}$	0.88	2.05	4.28	5.74	6.84
		$\mathsf{P}_{\mathcal{C}_{i,n} \cap \mathcal{D}_i}$	0.77	0.96	1.27	1.44	1.96

Table 2: Lipschitz constant obtained with various constrained optimization strategies for different accuracies – 7-gestures dataset.

can affect its approximation capability and potentially lead to a performance decay. Training the proposed system subject to constraint $\mathcal{D}$ results in an overall accuracy of 96.92 %, 95.87%, and 84.75 % for the case of 7, 13, and 24 classes, respectively. The performance decay was balanced by an increase in the robustness, since the Lipschitz constant, computed as indicated in Proposition 2.2, equals $\theta_m = 9.69 \times 10^{10}$ for 7 classes, $\theta_m = 9.73 \times 10^{10}$ for 13 classes and $\theta_m = 1.03 \times 10^{11}$ for 24 classes. We observed that the accuracy reduction can be overcome by adding additional layers to the architecture. Indeed, we

were able to obtain a similar accuracy to the baseline by adding an extra layer to the existing architecture and retraining both systems subject to $\mathcal{D}$, i.e. 98.68%, 97.21% and 85.12% for the 7-gesture, 13-gesture, and 24-gesture datasets, respectively. Furthermore, compared to the unconstrained models, we managed to maintain a high performance while improving the robustness with respect to unconstrained training, i.e. $\theta_m = 1.02 \times 10^{11}$ for the 7-classes dataset, $\theta_m = 9.96 \times 10^{10}$ for the 13-classes dataset and $\theta_m = 4.24 \times 10^{11}$ for the 24-classes dataset. We can however conclude from these tests that imposing the nonnegativity of the weight coefficients is not sufficient to reach satisfactory robustness.

To further control the robustness of the systems, we have to manage the Lipschitz constant of the networks by training them under additional spectral norm constraints, as described by (11). Searching for the optimal accuracy robustness trade-off, we trained several models considering each of the three aforementioned constraints, namely $(\mathcal{C}_{i,n})_{1\leq i\leq m, n\in\mathbb{N}}$ in (12), $(\widetilde{\mathcal{C}}_i)_{1\leq i\leq m}$ in (20), and $(\check{\mathcal{C}}_{i,n})_{1\leq i\leq m, n\in\mathbb{N}}$ in (10).

By adjusting the upper bound $\overline{\vartheta}$, we were able to assess the effect of a robustness constraint on the overall performance of the neural network-based classifiers, and finally to achieve the optimal trade-off. All our models were trained using Algorithm 1 as the optimizer.

The obtained results are summarized in Table 2. As expected, obtaining a good robustness-accuracy trade-off requires paying attention to the way we design our constrained networks. In all the cases, we show that using tight constraints during the training phase to approximate the Lipschitz bound improves the overall performance of the classifier, proving the generaliza-

tion properties of our solution. For comparison, for each of the proposed constraints, we also evaluated the use of an inexact projection, designated by $\widetilde{\mathsf{P}}$ (see Section 3.3). It can be observed that using an exact projection yields significantly better results. By combining tight constraints and exact projection techniques, we observe that the robustness of the network can be properly ensured while keeping a good accuracy in both cases. Indeed, we succeeded in ensuring a Lipschitz constant around 1 for a 95% accuracy. The observed loss in accuracy with respect to a standard training is consistent with the "no free lunch theorem" [47].

Training neural networks subject to tight spectral norm constraints can be challenging,[2] and the cost of obtaining a good performance is the training time. We used a learning rate scheduler strategy during training, reducing the learning rate by a factor of 2 if the performance does not improve for 1000 epochs. Figure 4 shows the training curves for both validation and training sets in the context of the unconstrained baseline model (yellow and green lines), and in the case of training a constrained version (red and blue lines) using the optimal projection $\mathsf{P}_{\mathcal{C}_{i,n}\cap\mathcal{D}_i}$, with $\overline{\vartheta}_m = 0.95$. Even though it requires more iterations, the constrained model is capable of reaching an accuracy comparable with the baseline, while providing a robustness certificate.

Since the training curves may show some slight variations, we measured the accuracy variations in two ways: by computing the classical *standard deviation (*std), and by employing *median absolute deviation (*mad*)*. For a

[2]A code in TensorFlow will be made available upon the acceptance of the paper.

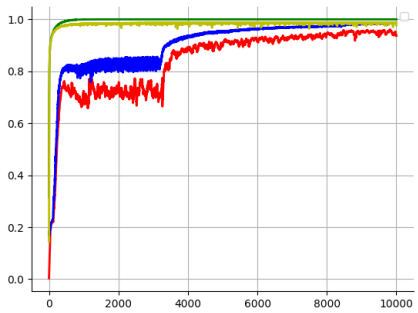


Figure 4: *Accuracy vs. Iterations – constrained and unconstrained models in the context of 7-gesture dataset. The training and validation curves are displayed in green and yellow, respectively, for the unconstrained model. The training and validation curves are displayed in blue and red, respectively, in the case of constrained training, with the bound* $\overline{\vartheta} = 0.95$.

vector $(x_i)_{1\leq i\leq I}$, it is expressed as MAD = median $\big((|x_i - \zeta(x)|)_{1\leq i\leq I}\big)$, where $\zeta(x)$ represents the median of the vector components. From this quantity, we can derive an empirical estimate of the standard deviation by multiplying MAD with a factor equal to 1.4826. The latter estimate is known to be more robust to outliers for Gaussian distributed data, especially in the case of small populations. The results are summarized in Table 3. It can be observed that the empirical standard deviation is below 1.6% and the robust estimate of it is below 1.1% for all three datasets. These deviations values are normal considering the size of the dataset and shows that the presented results are relevant and consistent.

Next, we have also evaluated how the positivity constraint impacts the overall accuracy of our system. We trained a robust network by allowing the weights to have arbitrary signs. For this purpose, we control individually the Lipschitz constant of each layer $i \in \{1, \ldots, m\}$ to be less than a given value $\overline{\vartheta}^{1/m}$. The exact projection onto $\widetilde{\mathcal{C}}_i$, $\mathsf{P}_{\widetilde{\mathcal{C}}_i}$, as well as the approximate one $\widetilde{\mathsf{P}}_{\widetilde{\mathcal{C}}_i}$

	Accuracy	75%	80%	85%	90%	95%
7-gestures dataset	empirical std	0.65	1.22	0.56	1.35	1.10
Model Variation	robust std	1.02	0.94	0.53	0.87	1.07
13-gestures	empirical std	0.65	1.05	0.75	0.75	0.72
Model Variation	robust std	0.77	0.81	0.72	0.97	0.59
	Accuracy	65%	70%	75%	80%	85%
24-gestures	empirical std	0.68	0.95	0.87	0.77	0.76
Model Variation	robust std	0.89	0.74	0.79	0.89	0.64

Table 3: Standard deviation of accuracy computed on 15 epochs, after convergence, on the test set for constrained models.

were computed as described previously. In this case $\overline{\vartheta}$ represents an upper bound on the Lipschitz constant of the system. Table 4 summarize the results for different values of $\overline{\vartheta}$, for two datasets. We compare our method for dealing with Lipschitz constraints with the approach proposed in [48]. This approach which is implemented in the *deel-lip* library allows the user to train robust networks in a convenient manner, offering a robustness certificate by performing a spectral normalization for each layer. It can be observed on these datasets that our method yields similar results when using the approximate projection, but better ones when using the exact projection. These results underline again the importance of carefully managing the projections and the effect it has on the accuracy of the system.

5. Robustness validation

In this section we investigate to what extent the theoretical concepts described in the previous sections help in improving the robustness of the classifier in different settings. To this goal, we consider the following three scenarios. In the first one, we examine the impact of adversarial attacks on

	Accuracy		75%	80%	85%	90%	95%
7-gestures dataset Lipschitz constant	$\mathcal{C}_i$	$\widetilde{\mathsf{P}}_{\widetilde{\mathcal{C}}_i}$	72.03	127.5	1296	8.75×10^4	5.43×10^9
		$\mathsf{P}_{\widetilde{\mathcal{C}}_i}$	52.06	102.49	905.45	7.23×10^4	8.14×10^8
	Deel-lip[48]		75.81	126.9	1283.6	8.70×10^4	5.43×10^9
13-gestures dataset Lipschitz constant	$\mathcal{C}_i$	$\widetilde{\mathsf{P}}_{\widetilde{\mathcal{C}}_i}$	76.59	125.20	1016	2.03×10^4	4.3×10^8
		$\mathsf{P}_{\widetilde{\mathcal{C}}_i}$	61.22	99.74	740	1.26×10^4	6.7×10^7
	Deel-lip[48]		77.21	125.63	1120	2.04×10^4	4.5×10^8

Table 4: Lipschitz constant for networks trained with arbitrary signs – 7-gestures / 13-gestures datasets.

the performance of the classifier. The second scenario takes into account the effect of noise in the acquisition process. In the case of sEMG signals, this noise may come from imperfect skin-sensor contact caused by hairs or drops of sweat. In the last scenario we perform a real-life experiment using 10 able-bodied volunteers.

5.1. Sensitivity to adversarial attacks

We evaluate our robust model on purposely designed perturbations, by studying their influence on the overall performance of the system. We lead attacks on our best robust model in terms of accuracy and robustness achieving 92.95% accuracy and a Lipschitz constant $\overline{\vartheta} = 0.87$ for the 7-gesture dataset. We compare the results with two conventionally trained models: the best one in terms of performance, which achieves 99.78% prediction accuracy on non-adversarial data, and another one trained to have similar performance as our robust model reaching 92.99% accuracy on the original test set.

To create the adversarial samples we used some of the most popular white-box attackers, namely: *Fast gradient sign method (FGSM)* [24] – generates adversarial data based on the gradient of the cost function with respect to the input data; *Jacobian Saliency Map Attacker (JSMA)* [49] – computes a

Accuracy [%]								
	robust model			baseline model			adversarial trained model – PGD	
Attack	adversarial	non-adversarial	adversarial	non-adversarial	adversarial	non-adversarial	adversarial	non-adversarial
FGSM [24]	91.75	92.95	76.48	99.78	71.21	92.99	...	97.25
C&W ℓ_2 [51]	90.09		48.03		45.85		...	
PGD [50]	91.92		59.36		56.38		...	
JSMA [49]	91.10		89.37		81.27		...	

Table 5: White-box attack results. We consider out best constrained model, having a Lipschitz constant $\theta = 0.97$, two models trained conventionally: the best baseline and another one having similar performance as the constrained one. On the last columns we feature an adversarial trained model using PGD-generated perturbations.

perturbation based on ℓ_2 distance metric by iteratively selecting the input sample that will increase the chance of miss-classification; *Projected gradient descent (PGD)*[50] – uses local first order information about the network to create adversarial examples; *Carlini and Wagner (C&W)* [51] – utilizes ℓ_2 distance to compute the optimal adversarial perturbation.

We also show a comparison with another popular technique of ensuring the robustness of neural network-based models, namely *Adversarial training*. This implies training an extended version of the dataset, containing the original training data together with a perturbed version of the samples in an effort to increase the system stability against adversarial inputs. Note that this method is purely empirical and gives no theoretical robustness certificates. We implemented an adversarial training strategy detailed in [50], training the system using an augmented version of the dataset which was updated every 25 epochs. The adversarial samples were created using PGD attack and then the model was validated using data containing perturbations computed with various attacks.

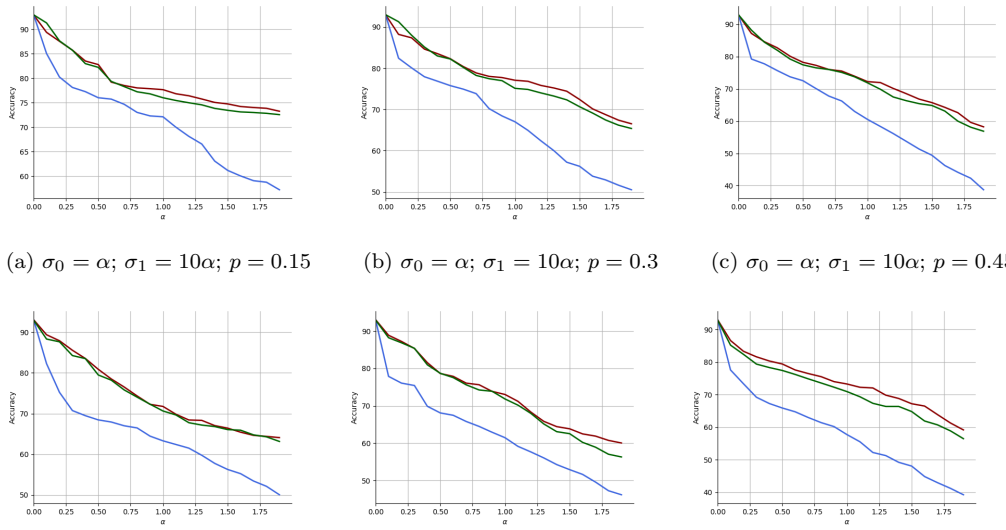
The results summarised in Table 5 show the performance obtained for

the 7-gesture test set. Note that the robust model performance is barely affected by the adversarial perturbations, whereas the baseline models show a huge drop in accuracy. It can be observed that adversarial training helps to increase the robustness, but our method of controlling the Lipschitz constant the network provides better results when facing data perturbed with other attackers than PGD. This shows that our method is more versatile, since its performance remains stable whatever the attacker.

5.2. Noisy input behaviour

To simulate the effect of underlying noise generated during the acquisition process, we added synthetic noise directly to the raw sEMG data, prior to the feature extraction step. The noise is chosen independent and identically distributed according to a Gaussian mixture law $(1-p)\mathcal{N}(0,\sigma_0^2)+p\mathcal{N}(0,\sigma_1^2)$. The mixture comprises a background component, corresponding to the intrinsic electronic noise in the armband, such as thermal or quantization noise, and an impulsive component accounting for outliers. Those may be related to imperfect wiring that can generate impulse-like artifacts. In our experiments, we consider background and impulse noises with standard deviations $\sigma_0 = \alpha$ and $\sigma_1 = 10\alpha$ with $\alpha \in [0, +\infty[$. We generate different levels of noise, by varying the parameter α. The probability of peaks $p \in [0, 1]$ is also adjusted to simulate more or less severe scenarios in terms of outliers.

From the resulting noisy signals, we extract the features described in Section 4 and pass them to the classifier, using our robust models reaching an accuracy of 92.95% ($\overline{\vartheta} = 0.87$) for the 7-gestures dataset, and 93.05% ($\overline{\vartheta} = 0.98$) in the case of the 13-gestures dataset, trained with non-altered data. We compared the results achieved with our robust training with those



(a) $\sigma_0 = \alpha$; $\sigma_1 = 10\alpha$; $p = 0.15$ (b) $\sigma_0 = \alpha$; $\sigma_1 = 10\alpha$; $p = 0.3$ (c) $\sigma_0 = \alpha$; $\sigma_1 = 10\alpha$; $p = 0.45$

(d) $\sigma_0 = \alpha$; $\sigma_1 = 10\alpha$; $p = 0.15$ (e) $\sigma_0 = \alpha$; $\sigma_1 = 10\alpha$; $p = 0.3$ (f) $\sigma_0 = \alpha$; $\sigma_1 = 10\alpha$; $p = 0.45$

Figure 5: *Accuracy vs. α in the context Noisy Inputs training. First row: 7-gesture dataset; Second row: 13-gestures dataset. Red line: robust model; Blue line: baseline model; Green line: adversarial trained model*

obtained with i) classical training and ii) adversarial training. In this case, the adversarial training was performed by generating an extended dataset, containing the original data and corrupted versions of them by additive noise following the Gaussian mixture law described above, where the parameters p and α were drawn randomly in a uniform manner on $[0.15, 0.45]$ and $[0, 2]$, respectively. In the absence of noise, a similar performance in terms of accuracy was obtained: 7-gestures dataset – 92.99%, and 92.97%, 13-gestures dataset – 93.03% and 92.98% for baseline and the adversarial training, respectively. The experimental results obtained on two datasets are depicted in Figure 5. The red, blue, and green lines correspond to the unconstrained, constrained, and adversarial models, respectively. We observe that the constrained model is significantly less affected by the presence of noise in the inputs than the one

Movement	User #1		User#2		User#3		User#4		User#5		User#6		User#7		User#8		User#9		User#10	
	C	U	C	U	C	U	C	U	C	U	C	U	C	U	C	U	C	U	C	U
up	2	2	1	3	0	0	0	1	0	0	0	2	0	2	1	2	0	2	1	3
down	1	1	0	2	2	3	0	0	2	4	1	0	2	3	1	1	0	1	0	1
right	0	4	0	0	0	1	0	1	1	1	0	2	0	0	0	0	0	1	1	2
left	3	5	1	4	0	1	0	1	2	5	0	0	0	1	2	3	1	2	0	1
fist	0	2	2	4	0	0	1	0	0	3	0	1	1	1	0	2	1	1	1	3
spread	0	3	2	5	3	4	2	4	1	0	0	0	1	2	1	0	0	1	0	3
Sum	6	17	6	18	5	9	3	7	6	13	1	5	4	9	6	7	2	8	3	3
Error rate (%)	5	14	5	15	4.1	7.5	2.5	5.7	5	10.7	0.7	4.1	3.3	7.5	5	5.7	1.6	6.6	2.5	10.7

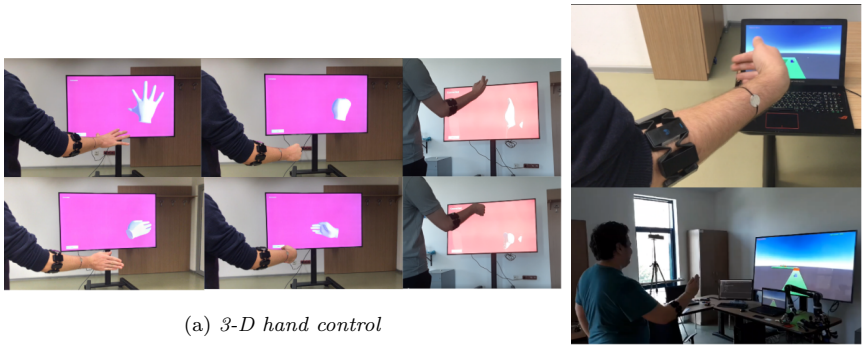
Table 6: Experiment results

trained without robustness guarantees. It is also worth noting that training with adversarial inputs also leads to satisfactory results, although usually slightly less accurate. The Lipschitz lower and upper bounds computed for the networks trained in an adversarial manner are indeed much lower than those with standard training, but they remain quite large ((1845.23, 79534.2) for 7-gestures dataset and (1754.74, 64595.8) for 13-gestures dataset).

This experiment emphasizes that controlling the Lipschitz constant of a network improves its robustness not only against targeted adversarial attacks, as shown previously, but also in the case of black-box attacks, where no prior information about the model is used.

5.3. Real-life scenario validation

To illustrate the practical applicability of our findings, we proceed to validate our model in a real-life context. For this purpose, we designed an experiment to compare a conventionally trained model with the constrained one. We integrated both models in a real-time application that controls a 3D hand on a screen, as well as a game that can be controlled by gestures,



(a) *3-D hand control*

(b) *Gesture-control game*

Figure 6: *Real-life experimental setup*

to give the user a tangible feedback. We used the `Unity`[3] platform to design and control a 3D hand and then encapsulated our models in an application which performed real-time inference and the hand was moving on the screen in accordance with the predicted gesture. We asked 10 volunteers (males and females) to test both models by performing each gesture 20 times. We emphasize that the user had no prior knowledge about what model was implemented, since it was randomly selected at the beginning of each new trial. Pictures of the experimental setup are provided in Figure 6. Table 6 details on a user level, how many (out of the 20) trials were erroneously classified. `U` and `C` denote the Unconstrained and the Constrained models, respectively. Note that, despite obtaining very good results on the test set, the unconstrained model loses a lot in terms of performance (up to 15%) when facing real-life data. We can observe that training a positive neural networks subject to Lipschitz constraints improves the overall robustness of the classifier against adversarial perturbations, not only from a theoretical viewpoint, but

[3] `https://unity.com/`

also practically by leading to more reliable systems with greater generalization power.

As for the other application, we asked the volunteers to play 2 rounds of a gesture-controlled game, one with each model. The game was inspired by the famous `Temple Run` [4], and consists of a moving cube which the user controls via gestures. The player can move his/hers hand left or right to move the character to either side of the screen to avoid obstacles. The player can also move the hand up to jump or spread its fingers to shoot and clear the obstacles ahead. The game is over when the player fails to take a turn or to jump/ clear an obstacle. We observed that 70% of the users were able to obtain higher scores when they used the constrained model, showing again that our solution is more stable when it comes to real-life applications.

5.4. Limitations

Increased training time is one of the main limitations of our proposed approach. Indeed, to compute the true projection, the proposed method uses an iterative algorithm which performs singular value decomposition at each iteration, which is a resource consuming operation, especially when performed on large matrices. We propose several lower complexity solutions, which have proved to offer a good trade-off between training time, robustness and performance.

Another limitation is related to the fact that our method for controlling the Lipschitz constant of the system is currently applicable in the context

[4] `https://play.google.com/store/apps/details?id=com.imangi.templerun&hl=en&gl=US&pli=1`

of nonnegative-weighted fully connected feed-forward neural networks. Although, the performance remains good for the considered AGR systems, the nonnegativity constraint might lead to a loss of expressivity of the neural networks in other inference tasks. In a future work, we plan to extend our method towards more general neural network architectures, including convolutional layers, skip connections, etc.

6. Conclusion

This work has shown the usefulness of designing robust feed-forward neural networks for automatic gesture recognition based on sEMG physiological signals. More precisely, we proposed to finely control the Lipschitz constant of these nonlinear systems by considering positively weighted neural architectures. To offer robustness certificates, we also developed new optimization techniques for training classifiers subject to spectral norm constraints on the weights. We studied various constrained formulations and showed that robustness can be secured without sacrificing accuracy when using a combination of tight constraints and exact projections. We also provide several lower-complexity solutions, which reduce the training time significantly.

Experiments on three distinct datasets illustrated the good performance of our approach. We further demonstrated the effectiveness of our robust classifier, compared to classically trained ones, when facing white-box and black-box attacks, as well as in real-life usage.

In future works, it would be interesting to apply such a robust training procedure to other applications in pattern recognition involving data acquired in real-time.

Appendix – Accelerated DFB algorithm

Let $n \in \mathbb{N} \setminus \{0\}$ and $i \in \{1, \dots, m\}$. Computing the projection of a matrix $\overline{W}_i \in \mathbb{R}^{N_i \times N_{i-1}}$ onto $\mathcal{D}_i \cap \mathcal{C}_{i,n}$ is equivalent to solve the following matrix optimization problem:

$$\underset{W_i \in \mathbb{R}^{N_i \times N_{i-1}}}{\text{minimize}} \quad \iota_{\mathcal{D}_i}(W_i) + \iota_{\mathcal{B}(0,\overline{\vartheta})}(A_{i,n} W_i B_{i,n}) + \frac{1}{2}\|W_i - \overline{W}_i\|_{\mathrm{F}}^2 \tag{30}$$

where $\|\cdot\|_{\mathrm{F}}$ is the Frobenius norm and $\iota_{\mathcal{S}}$ denotes the indicator of a set $\mathcal{S}$ (this function is equal to 0 on this set and $+\infty$ otherwise.) The dual optimization problem associated to this strongly convex minimization problem reads

$$\underset{Y \in \mathbb{R}^{N_m \times N_0}}{\text{minimize}} \quad f^*(-A_{i,n}^\top Y B_{i,n}^\top) + \iota^*_{\mathcal{B}(0,\overline{\vartheta})}(Y), \tag{31}$$

where for a given function g, g^* denotes its Fenchel-Legendre conjugate. In our case $f = \iota_{\mathcal{D}_i} + \frac{1}{2}\|\cdot - \overline{W}_i\|_{\mathrm{F}}^2$. From standard conjugation rules [40], f^* is equal to

$$(\forall W_i \in \mathbb{R}^{N_i \times N_{i-1}}) \quad f^*(W_i) = \widetilde{\iota_{\mathcal{D}_i}}(W_i + \overline{W}_i), \tag{32}$$

where $\widetilde{\iota_{\mathcal{D}_i}}$ is the Moreau envelope of $\iota^*_{\mathcal{D}_i}$ given by

$$\widetilde{\iota_{\mathcal{D}_i}}(W_i) = \inf_{W_i' \in \mathbb{R}^{N_i \times N_{i-1}}} \iota^*_{\mathcal{D}_i}(W_i') + \frac{1}{2}\|W_i' - W_i\|_{\mathrm{F}}^2. \tag{33}$$

The Moreau envelope of a proper lower-semincontinuous convex function is differentiable. Thus f^* is differentiable and its gradient is [52, Example 17.33]

$$\nabla f^*(W_i) = \mathsf{P}_{\mathcal{D}_i}(W_i + \overline{W}_i). \tag{34}$$

We deduce that the gradient of $Y \mapsto f^*(-A_{i,n}^\top Y B_{i,n}^\top)$ is

$$-A_{i,n} \mathsf{P}_{\mathcal{D}_i}(\overline{W}_i - A_{i,n}^\top Y B_{i,n}^\top) B_{i,n}.$$

Since $\mathsf{P}_{\mathcal{D}_i}$ is a nonexpansive operator, the latter function has a Lipschitz gradient with constant $\beta = \|A_{i,n}\|_{\mathrm{S}}^2 \|B_{i,n}\|_{\mathrm{S}}^2$. The dual problem (31) thus corresponds to the minimization of the sum of a smooth convex function and a proper lower-semicontinuous function. Consequently, it can be minimized by a proximal algorithm. Such a strategy will require to calculate the proximity operator of $\gamma \iota^*_{\mathcal{B}(0,\overline{\vartheta})}$ for some scaling parameter $\gamma \in]0, +\infty[$. By using Moreau's formula [52], this proximity operator is expressed as

$$(\forall Y \in \mathbb{R}^{N_m \times N_0}) \quad \mathrm{prox}_{\gamma \iota^*_{\mathcal{B}(0,\overline{\vartheta})}}(Y) = Y - \gamma \mathsf{P}_{\mathcal{B}(0,\overline{\vartheta})}(\gamma^{-1} Y). \tag{35}$$

A classical solution for solving the dual problem consists in using the standard forward-backward algorithm [53, 35]. This leads to Algorithm 3 [41]. Another solution consists in using the FISTA-like algorithm in [54], which leads to the accelerated version in Algorithm 2. The sequences $(Y_\ell)_{\ell \in \mathbb{N}}$ generated by these two algorithms is guaranteed to converge to a solution $\widehat{Y}$ to the dual problem. In addition, from Kuhn-Tucker conditions, the solution to the primal problem $\widehat{W}_i = \mathsf{P}_{\mathcal{S}_{i,n}}(\overline{W}_i)$ is equal to $\nabla f^*(-A_{i,n}^\top \widehat{Y} B_{i,n}^\top)$. It follows from (34) and the continuity of $\mathsf{P}_{\mathcal{D}_i}$ that the sequence $(V_\ell)_{\ell \in \mathbb{N}}$ converges to $\widehat{W}_i$.

Algorithm 3: Dual Forward-backward algorithm

Let $Y_0 \in \mathbb{R}^{N_m \times N_0}$

Set $\epsilon \in]0, 1/(\|A_{i,n}\|_{\mathrm{S}} \|B_{i,n}\|_{\mathrm{S}})^2[$

for $l = 0, 1, \ldots$ **do**

  Set $\gamma_\ell \in [\epsilon, 2/(\|A_{i,n}\|_{\mathrm{S}} \|B_{i,n}\|_{\mathrm{S}})^2 - \epsilon]$ $V_\ell = \mathsf{P}_{\mathcal{D}_i}(\overline{W}_i - A_{i,n}^\top Y_\ell B_{i,n}^\top)$

  $\widetilde{Y}_\ell = Y_\ell + \gamma_\ell A_{i,n} V_\ell B_{i,n}$ $Y_{\ell+1} = \widetilde{Y}_\ell - \gamma_\ell \mathsf{P}_{\mathcal{B}(0,\overline{\vartheta})}(\gamma_\ell^{-1} \widetilde{Y}_\ell)$

Acknowledgements

This project was supported by the BRIDGEABLE ANR Research and Teaching Chair in AI and by the Romanian National Authority for Scientific Research and Innovation, UEFISCDI, project EGMP-AI, agreement 425PED/2020, grant PN-III-P2-2.1-PED-2019-2392.

References

[1] J. He, S. L. Baxter, J. Xu, J. Xu, X. Zhou, K. Zhang, The practical implementation of artificial intelligence technologies in medicine, Nat. med. 25 (1) (2019) 30–36.

[2] P. Li, X. Chen, S. Shen, Stereo R-CNN based 3D object detection for autonomous driving, in: Proc. IEEE Conf. Comput. Vis. Pattern Recogn., Long Beach, CA, USA, 2019, pp. 7644–7652.

[3] N. Widiastuti, Convolution neural network for text mining and natural language processing, in: IOP Conf.: Mater. Sci. Eng., Vol. 662, Kazimierz Dolny, Poland, 2019.

[4] A. Kurakin, Z. Zhang, Z. Liu, A real time system for dynamic hand gesture recognition with a depth sensor, in: Proc. IEEE European Signal Processing Conf., Bucharest, Romania, 2012, pp. 1975–1979.

[5] J. Qi, G. Jiang, G. Li, Y. Sun, B. Tao, Intelligent human-computer interaction based on surface EMG gesture recognition, IEEE Access 7 (2019) 61378–61387.

[6] M. J. Cheok, Z. Omar, M. H. Jaward, A review of hand gesture and sign language recognition techniques, Int. J. Mach. Learn. Cyber. 10 (1) (2019) 131–153.

[7] F. Wen, Z. Sun, T. He, Q. Shi, M. Zhu, Z. Zhang, L. Li, T. Zhang, C. Lee, Machine learning glove using self-powered conductive superhydrophobic triboelectric textile for gesture recognition in VR/AR applications, Adv. Sci. 7 (14) (2020) 2000261.

[8] X. Gao, Y. Jin, Q. Dou, P.-A. Heng, Automatic gesture recognition in robot-assisted surgery with reinforcement learning and tree search, in: Proc. IEEE Int. Conf. Robot. Autom., Paris, France, 2020, pp. 8440–8446.

[9] O. Kopuklu, N. Kose, G. Rigoll, Motion fused frames: Data level fusion strategy for hand gesture recognition, in: Proc. IEEE Conf. Comput. Vis. Pattern Recogn., Utah, USA, 2018, pp. 2103–2111.

[10] T. J. Darrell, I. A. Essa, A. P. Pentland, Task-specific gesture analysis in real-time using interpolated views, IEEE Trans. Pattern Anal. Mach. Intell. 18 (12) (1996) 1236–1242.

[11] Z. Yang, D. Jiang, Y. Sun, B. Tao, X. Tong, G. Jiang, M. Xu, J. Yun, Y. Liu, B. Chen, J. Kong, Dynamic gesture recognition using surface EMG signals based on multi-stream residual network, Front. in Bioengin. and Biotech. 9 (2021).

[12] N. Neverova, C. Wolf, G. Taylor, F. Nebout, Moddrop: adaptive multi-modal gesture recognition, IEEE Trans. Pattern Anal. Mach. Intell. 38 (8) (2015) 1692–1706.

[13] E. Ceolini, C. Frenkel, S. B. Shrestha, G. Taverni, L. Khacef, M. Payvand, E. Donati, Hand-gesture recognition based on EMG and event-based camera sensor fusion: A benchmark in neuromorphic computing, Front. in Neurosci. 14 (2020).

[14] Y. Jiang, L. Song, J. Zhang, Y. Song, M. Yan, Multi-category gesture recognition modeling based on sEMG and IMU signals, Sensors 22 (15) (2022).

[15] M. Atzori, A. Gijsberts, C. Castellini, B. Caputo, A.-G. M. Hager, S. Elsig, G. Giatsidis, F. Bassetto, H. Müller, Electromyography data for non-invasive naturally-controlled robotic hand prostheses, Sci. data (140053) (23 Dec 2014).

[16] F. Peng, C. Chen, D. Lv, N. Zhang, X. Wang, X. Zhang, Z. Wang, Gesture recognition by ensemble extreme learning machine based on surface electromyography signals, Front. in Human Neurosci. 16 (2022).

[17] R. N. Khushaba, S. Kodagoda, Electromyogram (EMG) feature reduction using mutual components analysis for multifunction prosthetic fingers control, in: Int. Conf. Control Autom. Robotics & Vision, Guangzhou, China, 2012, pp. 1534–1539.

[18] J. Kim, S. Mastnik, E. André, EMG-based hand gesture recognition for realtime biosignal interfacing, in: Proc. Int. Conf. Intell. User Interfac., Canaria, Spain, 2008, pp. 30–39.

[19] A. D. Orjuela-Cañón, A. F. Ruíz-Olaya, L. Forero, Deep neural network for EMG signal classification of wrist position: Preliminary results, in: IEEE Latin American Conf. Comput. Intell., Arequipa, Peru, 2017, pp. 1–5.

[20] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, R. Fergus, Intriguing properties of neural networks, in: Proc. Int. Conf. Learn. Represent., Banff, Canada, 2014.

[21] A. Kurakin, I. Goodfellow, S. Bengio, Adversarial machine learning at scale, in: Int. Conf. Learn. Represent., Toulon, France, 2017.

[22] N. Carlini, P. Mishra, T. Vaidya, Y. Zhang, M. Sherr, C. Shields, D. Wagner, W. Zhou, Hidden voice commands, in: USENIX Security Symp., Austin, TX, USA, 2016, pp. 513–530.

[23] M. Takeru, K. Toshiki, K. Masanori, Y. Yuichi, Spectral normalization for generative adversarial networks, in: Int. Conf. Learn. Represent., Vancouver, Canada, 2018.

[24] I. J. Goodfellow, J. Shlens, C. Szegedy, Explaining and harnessing adversarial examples, in: Int. Conf. Learn. Represent., San Diego, CA, USA, 2015.

[25] A. Neacşu, J.-C. Pesquet, C. Burileanu, Accuracy-robustness trade-off for positively weighted neural networks, in: Proc. IEEE Int. Conf. Acoust. Speech Signal Process., Barcelona, Spain, 2020, pp. 8389–8393.

[26] A. Fatayer, W. Gao, Y. Fu, sEMG-based gesture recognition using deep learning from noisy labels, IEEE J. of Biomed. and Health Info. 26 (9) (2022) 4462–4473. doi:10.1109/JBHI.2022.3179630.

[27] K. Scaman, A. Virmaux, Lipschitz regularity of deep neural networks: analysis and efficient estimation, in: Proc. Ann. Conf. Neur. Inform. Proc. Syst., Montreal, Canada, 2018, pp. 3839–3848.

[28] C. Anil, J. Lucas, R. Grosse, Sorting out Lipschitz function approximation, in: Proc. Int. Conf. Mach. Learn., Long Beach, California, USA, 2019, pp. 291–301.

[29] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin, N. Usunier, Parseval networks: Improving robustness to adversarial examples, in: Proc. Int. Conf. Mach. Learn., Sydney, NSW, Australia, 2017, pp. 854–863.

[30] M. Fazlyab, A. Robey, H. Hassani, M. Morari, G. Pappas, Efficient and accurate estimation of lipschitz constants for deep neural networks, in: Adv. Neural Info. Process. Syst., Vancouver, Canada, 2019, pp. 11423–11434.

[31] P. Pauli, A. Koch, J. Berberich, P. Kohler, F. Allgower, Training robust neural networks using Lipschitz bounds, IEEE Control Syst. Lett. 6 (2021) 121–126.

[32] J. Chorowski, J. M. Zurada, Learning understandable neural networks with nonnegative weight constraints, IEEE Trans. Neural Netw. Learn. Syst. 26 (1) (2015) 62–69.

[33] P. L. Combettes, J.-C. Pesquet, Lipschitz certificates for layered network structures driven by averaged activation operators, SIAM J. on Math. Data Sci. 2 (4) (2020) 529–557.

[34] P. Combettes, J.-C. Pesquet, Deep neural network structures solving variational inequalities, Set-Valued Var. Anal. (2020) 1–28.

[35] P. Combettes, J.-C. Pesquet, Proximal thresholding algorithm for minimization over orthonormal bases, SIAM J. on Optim. 18 (4) (2008) 1351–1376.

[36] P. Ramachandran, B. Zoph, Q. V. Le, Searching for activation functions, Proc. Int. Conf. Learn. Represent. (30 Apr.–03 May 2018).

[37] F. Latorre, P. Rolland, V. Cevher, Lipschitz constant estimation of neural networks via sparse polynomial optimization, in: Int. Conf. on Learning Representations, 2020.

[38] T. Chen, J.-B. Lasserre, V. Magron, E. Pauwels, Semialgebraic optimization for Lipschitz constants of ReLU networks (2020) 19189–19200.

[39] P. L. Combettes, J.-C. Pesquet, Fixed point strategies in data science, IEEE Trans. Sig. Proc. 69 (2021) 3878–3905.

[40] N. Komodakis, J.-C. Pesquet, Playing with duality: An overview of recent primal-dual approaches for solving large-scale optimization problems, IEEE Signal Process. Mag. 32 (6) (2015) 31–54.

[41] P. L. Combettes, D. Dũng, B. C. Vũ, Dualization of signal recovery problems, Set-Valued Var. Anal. 18 (3–4) (2010) 373–404.

[42] C. Andronache, M. Negru, A. Neacşu, G. Cioroiu, A. Rădoi, C. Burileanu, Towards extending real-time EMG-based gesture recognition system, in: Proc. IEEE Int. Conf. Telecomm. Signal Porcess., Milan, 2020, pp. 301–304.

[43] U. Côté-Allard, C. L. Fall, A. Drouin, A. Campeau-Lecours, C. Gosselin, K. Glette, F. Laviolette, B. Gosselin, Deep learning for electromyographic hand gesture signal classification using transfer learning, IEEE Trans. Neural Syst. Rehabilitation Eng. 27 (4) (2019) 760–771.

[44] S. Pizzolato, L. Tagliapietra, M. Cognolato, M. Reggiani, H. Müller, M. Atzori, Comparison of six electromyography acquisition setups on hand movement classification tasks, PLOS ONE 12 (10) (10 2017).

[45] C. Andronache, M. Negru, I. Bădiţoiu, G. Cioroiu, A. Neacsu, C. Burileanu, Automatic gesture

recognition framework based on forearm emg activity, in: Proc. IEEE Int. Conf. Telecomm. Signal Porcess., 2022, pp. 284–288.

[46] A. A. Neacsu, G. Cioroiu, A. Radoi, C. Burileanu, Automatic EMG-based hand gesture recognition system using time-domain descriptors and fully-connected neural networks, in: Int. Conf. Telecommunications Signal Process., Budapest, Hungary, 2019, pp. 232–235.

[47] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, A. Madry, There is no free lunch in adversarial robustness (but there are unexpected benefits), CoRR, abs/1805.12152, 2018 (2018).

[48] M. Serrurier, F. Mamalet, A. González-Sanz, T. Boissin, J.-M. Loubes, E. del Barrio, Achieving robustness in classification using optimal transport with hinge regularization, in: Proc. IEEE Conf. Comput. Vis. Pattern Recogn., Nashville, USA, 2021, pp. 505–514.

[49] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, A. Swami, The limitations of deep learning in adversarial settings, in: IEEE Symp. Security Privacy, Saarbrücken, Germany, 2016.

[50] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, A. Vladu, Towards deep learning models resistant to adversarial attacks, in: Proc. Int. Conf. Learning Representations, Vancouver, BC, Canada, 2018.

[51] N. Carlini, D. Wagner, Towards evaluating the robustness of neural networks, in: IEEE Symp. Security Privacy, San Jose, CA, USA, 2017, pp. 39–57.

[52] H. Bauschke, P. Combettes, Convex analysis and monotone operator theory in Hilbert spaces, 2019.

[53] P. L. Combettes, V. R. Wajs, Signal recovery by proximal forward-backward splitting, Multiscale Model. Simul. 4 (4) (2005) 1168–1200.

[54] A. Chambolle, C. Dossal, On the convergence of the iterates of the fast iterative shrinkage/thresholding algorithm, J. Optim. Theory Appl. 166 (3) (2015) 968–982.