applied sciences

MDPI

Article

LSTM Recurrent Neural Network for Hand Gesture Recognition Using EMG Signals

Alejandro Toro-Ossaba [1,†], Juan Jaramillo-Tigreros [1,†], Juan C. Tejada [1,2,*,†], Alejandro Peña [3,†], Alexandro López-González [2,†] and Rui Alexandre Castanho [4,5,†]

[1] Grupo de Investigación en Inteligencia Computacional y Automática (GIICA), Universidad EIA, Envigado 055428, Colombia
[2] Departamento de Estudios en Ingeniería para la Innovación, Universidad Iberoamericana, Ciudad de México 01219, Mexico
[3] Grupo de Investigación en Información y Gestión, Escuela de Administración, Universidad EAFIT, Medellín 050021, Colombia
[4] Faculty of Applied Sciences, WSB University, 03-204 Dabrowa Gornicza, Poland
[5] College of Business and Economics, University of Johannesburg, Auckland Park, P.O. Box 524, Johannesburg 2006, South Africa
* Correspondence: juan.tejada@eia.edu.co
† These authors contributed equally to this work.

Abstract: Currently, research on gesture recognition systems has been on the rise due to the capabilities these systems provide to the field of human–machine interaction, however, gesture recognition in prosthesis and orthesis has been carried out through the use of an extensive amount of channels and electrodes to acquire the EMG (Electromyography) signals, increasing the cost and complexity of these systems. The scientific literature shows different approaches related to gesture recognition based on the analysis of EMG signals using deep learning models, highlighting the recurrent neural networks with deep learning structures. This paper presents the implementation of a Recurrent Neural Network (RNN) model using Long-short Term Memory (LSTM) units and dense layers to develop a gesture classifier for hand prosthesis control, aiming to decrease the number of EMG channels and the overall model complexity, in order to increase its scalability for embedded systems. The proposed model requires the use of only four EMG channels to recognize five hand gestures, greatly reducing the number of electrodes compared to other approaches found in the literature. The proposed model was trained using a dataset for each gesture EMG signals, which were recorded for 20 s using a custom EMG armband. The model reached an accuracy of to 99% for the training and validation stages, and an accuracy of 87 ± 7% during real-time testing. The results obtained by the proposed model establish a general methodology for the reduction of complexity in the recognition of gestures intended for human.machine interaction for different computational devices.

Keywords: gesture recognition; recurrent neural networks (RNN); long short-term memory (LSTM)

Citation: Toro-Ossaba, A.; Jaramillo-Tigreros, J.; Tejada, J.C.; Peña, A.; López-González, A.; Castanho, R.A. LSTM Recurrent Neural Network for Hand Gesture Recognition Using EMG Signals. *Appl. Sci.* **2022**, *12*, 9700. https://doi.org/10.3390/app12199700

Academic Editor: DaeEun Kim

Received: 26 July 2022
Accepted: 22 September 2022
Published: 27 September 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

1. Introduction

Hand gesture recognition systems have been improving greatly within the last two decades, mainly because of the need for more natural and accurate human–computer interaction systems [1]. Furthermore, Human Computer Interfaces (HCI) are becoming very important in one field in particular: upper limb prosthesis [2,3]; hands are known for being one of the most important and functional parts of our body, hence loss of them can result in a subject notable deterioration of life quality. This is mainly why the literature is rich in studies aiming for optimum ways to control upper limb active prosthesis [4]. Even with this amount of literature there are some major challenges regarding actual HCI techniques for upper limb prosthesis control [5], coming first, three of these main challenges are listed and explained:

1. The first challenge involves hand gesture recognition in real-time [6]. Most studies regarding this topic focus primarily in finding novel algorithm-based methods in order to achieve high performance rates in gesture recognition. Nevertheless these scores are obtained in non-real-time tests where factors such as computational efficiency, window length and system overall performance are not taken in to account [7,8]. Furthermore, this kind of hand gesture recognition methods may be suitable for non-mobile applications such as sign language recognition [9], but not for active prosthesis which are mobile and independent systems.
2. The second challenge is to find such a good performance classifier that it can be used in different types of amputees and still has a good accuracy rate in hand gesture recognition [10], moreover, the challenge is to find an appropriate classifier that can deliver good recognition rates in almost every amputee morphology, while keeping the prediction delay low and the overall system computational needs moderate.
3. The third challenge is to narrow the gap between academic and industrial achievements. Both of them, at this point, are heading in totally different ways. While an academic display can be set in a known environment, the industrial one needs to be as robust as possible in order to increase the overall performance of the system when is used by a real world user [11].

Taking these issues, in the scientific literature three well-defined development trends can be seen. The first development trend focuses on the use of images to cluster hand gestures, highlighting the cameras as a common sensor that can be found in a great number of electronic devices nowadays [12], allowing to identify hand gestures both in static and dynamic environments, as well as in real time [13]. The main source of information in this kind of classification is images, which is why large data sets of pictures are taken, most of them containing multiple hands from different subjects, waving different signs and gestures, regardless of whether the model sorts static or dynamic movements [14]. The main data that it needs are images, either if it is a sequence of them (Dynamic movement classification) or just one of them (Static movement classification) [15], but there is an even lower level approach to this particular data set: Images are large matrices composed of RGB (Red, Green, Blue) pixel data. Then these pixels are the basic structural and functional units of an image. That is why some studies dig even deeper in to this kind of data and add depth to the image composition. Such grids are formed with RGB-D pixels [16,17]. This can be accomplished by using commercially available cameras such as the ones that can be found in a Kinect [18] or in the commercially available leap motion controller [19]. The next step in this kind of classification is to process those large grids of pixels and to obtain a good fitting model, capable of sorting different types of gestures. Some of those are based on Convolutional Neural Networks (CNN) [16,20], K-Nearest Neighbor (KNN) and Bayesian Neural Network [21]. In certain cases the model descriptors are changed in order to lower the computational demands [22,23] for real-time image processing.

The second development focus is the use of Electromyography (EMG) for hand gesture recognition. It is important to mention that the EMG shows the electrical muscular activity recorded by positioning electrodes on top of the muscular group of interest. In order to obtain valuable information for the characterization of the hand gesture, filtering, processing and clustering methods of EMG signal have emerged as highly relevant techniques [24]. Another batch of papers shows a lot of methods and variants, used in the literature, to increase the rate of accuracy of the process mentioned above, starting by using a large amount of electrodes, followed by the use of huge time resolution ADCs (Analog to digital converters), however the real variety in literature comes when it is time to choose the descriptors [24–26] and what algorithm to use for sorting gestures, from recurrent neural networks (RNN) [27,28], to Hidden Markov Models [29], all the way to gesture recognition based on Motor Unit Spike Trains (MUST) [30], nearly every algorithm has been used. Nevertheless deep learning models have been the most accurate and reliable, RNN have shown good results along with Long-short Term Memory (LSTM) layer in them [17,27,28].

The third development trend, which is starting to take off, is force myography (FMG). It is basically an alternative for EMG sensing, and it is based on the volumetric changes of muscle fibers while they contract or distract [31]. Some researchers claim that it is even better than EMG because it only needs force or pressure sensors attached to an armband, and the requirements for such signals processing do not demand as many filters or amplifiers as EMG does [29]. Here, the simple nature of both signals is different and that is why the approaches are distinct. Whilst the EMG is an electrical signal caused by small impulses created in the motor neurons when contracting the muscles [24], FMG is a volume change that can be sensed by using special transducers in order to know its magnitude. If a big picture of the process of this kind of hand gesture recognition is taken, much would be similar to the EMG signal acquiring and clustering process. Initially the signal is acquired by using sensors suitable for this approach. It is important to take into consideration the force or pressure range that the transducer is able to sense [32]; then these data are processed and lastly the gestures are classified using a various palette of algorithms such as Linear Discriminant Analysis (LDA) [5,27] and Recurrent Neural Networks[17].

Following the three challenges and the second trend mentioned before, Recurrent Neural Networks (RNN) emerge as promising models for gesture recognition classifiers based on EMG signals due to their ability to identify patterns in dynamic time series; RNNs have been widely used in different fields to forecast time series data and model energy system behavior [30,33,34]. In the field of Human–Machine Interfaces and electromyography, multiple studies have explored the application of RNNs in gesture recognition systems in order to determine whether is or not suitable for gesture prediction and prosthesis control. For example, Jabbari et al. has explored the use of stacked LSTM Recurrent Neural Networks for gesture recognition on amputees [35]; Samadani performed a comparative study evaluating unidirectional and bidirectional RNNs, using both Long-short Term Memory (LSTM) units and Gated Recurrent Units (GRU), also analyzing the effects of the attention mechanism, Samadani found that bidirectional LSTM units with attention yielded the best results [36]; Hu et al. proposed an attention-based hybrid CNN-RNN architecture for gesture recognition, this research compared the models accuracy on different popular gesture recognition datasets [37]; other studies like the ones presented by Zhang et al., Jiang et al. and Simão, proposed different RNNs architectures composed of multiple stacked recurrent layers and their comparison with other architectures such as CNN-RNN models [13,38,39].

As can be found in the literature, most of the studies use static data and online datasets in which the data was obtained in controlled environments, however, scientific literature has shown that models that often perform well on static data do not perform well in systems operating in real-time [40], making the results obtained only in static data misleading. Moreover, most of the research found uses a high number of EMG channels (eight or more) [32,35–38,41,42], which increases the computational needs and the prediction delay, making it hard to implement those systems in embedded devices and commercial applications [5]. This research aims to propose a full hand gesture recognition system capable of predicting different gestures in real-time operation using only four EMG channels. Concepts and design of the acquisition system and the time domain optimum descriptor will be discussed during the overall development of this paper, to highlight their importance in the development of the classifier.

This paper presents a novel hybrid RNN model composed of one LSTM (Long Short-Term Memory) layer and dense layers (fully connected layers) to classify in real-time different hand gestures using only four EMG channels. The model achieved an accuracy 99% in the training and validation stages in the recognition of five hand gestures performed with static data. On the other hand, the proposed model was implemented on a personal computer that received the EMG data from a custom made EMG acquisition armband. The model achieved an accuracy of $87 \pm 7\%$ during real-time testing, proving to be suitable and scalable for real-time prosthesis control. The real-time tests revealed challenges that only occur during real-time operation, giving insights about possible solutions to this challenges to further optimize the system.

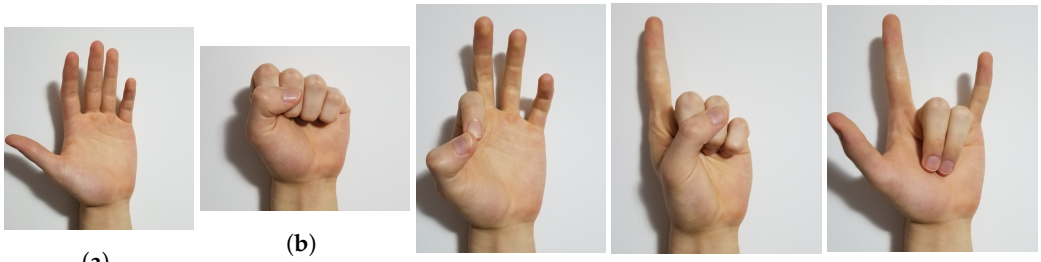
This article is structured as follows, Section 2 presents the methodology used to define the experimental set up of EMG system, the algorithm choosing and theoretic support are brought up during this section. Section 3 shows the main results obtained form the tests carried out by the system previously described; and a final section shows the conclusions (Section 4), setting ground for future work.

2. Methodology

Due to the complexity that frames the recognition of hand gesture in real time, this section presents a novel methodology and the steps followed to develop an integrated system for gesture recognition, showing key concepts that allow the implementation of a Recurrent Neural Network with LSTM and dense layers.

2.1. Experimental Set-Up and Data Acquisition

In order to set up a hand gesture classifier model, a recording of EMG signals from five different gestures was performed, which can be seen in Figure 1. These gestures were selected based on basic daily hand patterns, including two of the fundamental grasping motions, the power grip (Figure 1b) and the pinch grip (Figure 1c), which is a precision handling maneuver [43].



(**a**) (**b**) (**c**) (**d**) (**e**)

Figure 1. Gestures: (**a**) Open hand, (**b**) Power grip, (**c**) Pinch grip, (**d**) Point to, (**e**) Rock you.

The EMG data was acquired using a custom four-channel EMG armband [44]. The EMG armband acquired the EMG signal in each channel with dry electrodes, then it conditioned the signal so a microcontrollers ADC could read it and finally it sended packages containing the four channels information via serial communication. The data of the four EMG channels were received by a personal computer (PC) in which the preprocessing and development of the model were performed.

Each gesture was recorded for 20 s with a sampling frequency of 1 KHz, allowing to obtain 20,000 samples per gesture. The sampling frequency was selected following the Nyquist sampling theorem, considering a maximum frequency for the EMG signal of 500 Hz [45,46]; it was also taken into consideration the computational cost of acquiring the four EMG channels and the maximum serial transmission speed of the microcontroller. An adequate selection of the sampling frequency is key to acquire all the signal components and perform a better mapping of the signal with the proposed classifier.

The acquired samples were divided into 100 windows of 200 samples; collecting a total of 500 examples to build the dataset. The 200 samples window was selected based on a literature review on hand prosthesis requirements and EMG window length selection; where a prosthesis requires an acting speed of 500 ms or less, in order to work smoothly [47–50]. Taking into consideration that the sample time is 1 ms, 200 sample windows allow an actuation speed of around 200–300 ms considering a serial communication with the computational model, which is within the required speed; moreover, researchers have found that window lengths of 200 ms or more allow to have reduced classification errors when working with EMG signals [45,51].

The dataset was randomly shuffled and then split into two parts: 80% of the shuffled dataset was used for training, and the remaining 20% was used as a validation set to perform hyper-parameter tuning and validation of the training.

It is important to note that the acquisition of the EMG signals was done following all the ethical guidelines set by law and the institution. The data was taken from eight test subjects with an age range between 20 and 30 years. All participants were healthy people. The experiments were approved by EIA University Ethical Committee (Code P201912-04), and also an informed consent was obtained from the participants.

2.2. Data Preprocessing

Once the EMG data were acquired, the four EMG signals were preprocessed so they could be used as inputs to the proposed model. The preprocessing is a fundamental step when working with EMG signals, since raw EMG signals are random and difficult to interpreter, the preprocessing step allows to extract valuable information to the raw EMG signal This information then can be used as input data to a classifier. For this step an EMG feature extraction process is generally conducted. According to scientific literature there are many known features of EMG signals [3]. However, research on EMG signal processing showed that there is not a single best feature extraction process or formula, and that it largely depends on the data and required characteristics of the system [52].

With this in mind, the preprocessing step conducted in this research was mainly focused on smoothing the signal and removing the unuseful noise of it, leaving the feature extraction for the proposed neural model [53,54]. The preprocessing method applied to the signals can be seen in Figure 2. In this process, initially the mean of the signal is subtracted in order to center it at zero mean; then, a rectification step is performed in which the absolute value is applied to the zero mean signal, this is a useful step during the preprocessing of the EMG signal [46]; then, a moving average filter was applied in order to reduce the noise of the signal [13] and to obtain its envelope, which is an useful characteristic when working with neural network models; finally the signal was normalized between 0 and 1 as recommended in the Deep Learning literature [55].

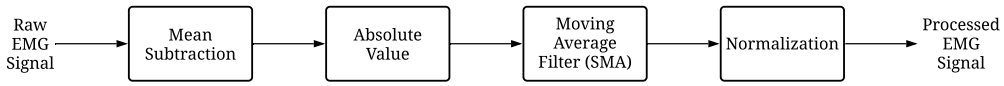


Figure 2. Data processing block diagram.

In general, the moving average is the most common filter in digital signal processing (DSP) due to its ease of implementation. In spite of its simplicity, the moving average filter is optimal for reducing random noise while retaining the signal, making it an outstanding filter for time domain encoded signals [56], thus allowing to obtain useful information about the EMG signal in the time domain, while keeping the computational cost low. Its formula is given by Equation (1).

$$y(i) = \frac{1}{M} \sum_{j=0}^{M-1} x(i-j) \tag{1}$$

For the moving average filter, a total of 20 delays for the filter period N was selected. This period allowed to smooth the signal reducing the noise, making it easier for the proposed LSTM model to identify the signal patterns that represent a hand gesture. Figure 3 shows the steps of the signal pre-processing for a window of 200 samples.

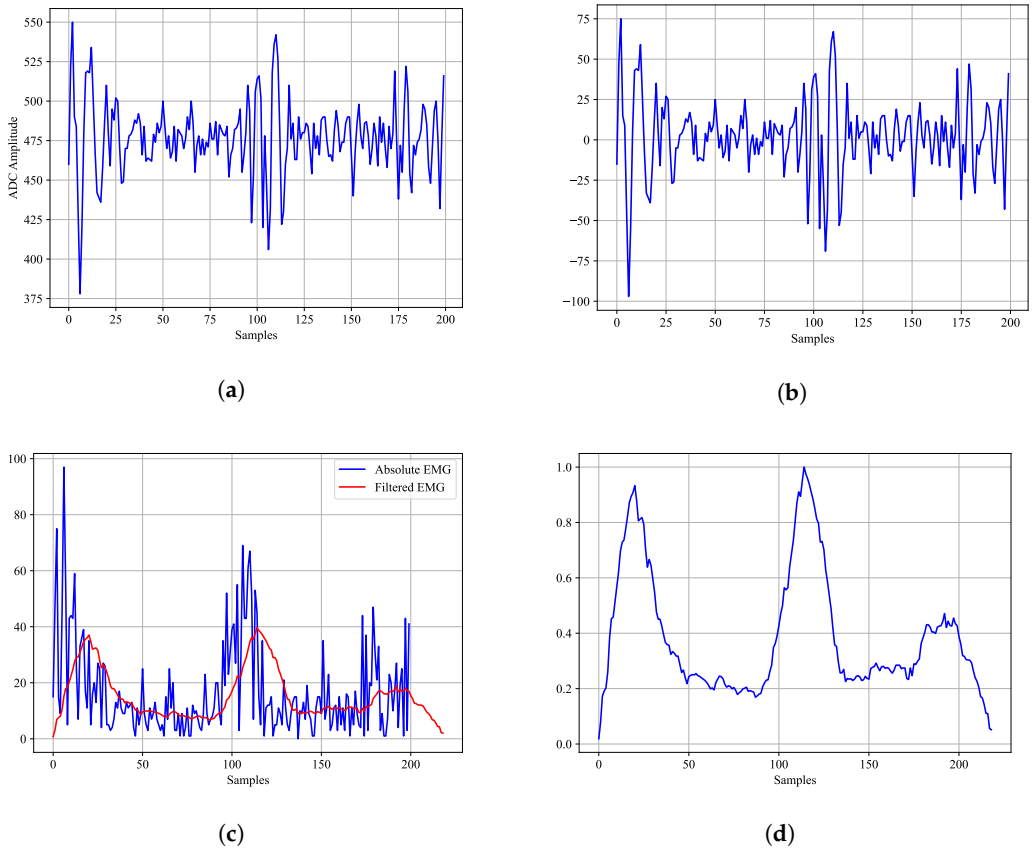


Figure 3. EMG 200 samples window processing. (**a**) Raw EMG data in 200 samples window; (**b**) EMG window after mean subtraction; (**c**) EMG window after absolute value and moving average filter; (**d**) Filtered and normalized EMG window.

2.3. Recurrent Neural Network (RNN) Model

To perform a hand gesture classification in real-time based on the pre-processed EMG signals, a novel Recurrent Neural Network (RNN) model of LSTM type was proposed. The RNN structure with LSTM configuration has been found to be a good architecture for the identification of temporal patterns in time series [57]. In general many researchers found that the RNNs have the ability to retain information making it an excellent approach for time series applications [58]. In this context, the RNN proposed in this research has a hybrid architecture composed of dense layers wrapping a recurrent layer; it has been found that the approach where dense layers support the recurrent layer allows for an increased performance over models that only have the recurrent layer [59,60]. The overall architecture of the RNN can be seen in Figure 4. Each one of its layers will be explained in greater detail in further Sections 2.3.1 and 2.3.2.

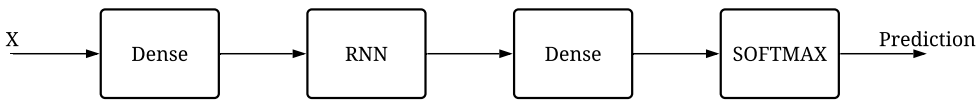


Figure 4. Recurrent neural network general architecture block diagram.

Figure 5 shows the basic structure of the recurrent layer for the proposed model. Here, each input X is composed of one sample of each EMG channel in an specific time t; where T_x is the length of the window that represent a hand gesture to classify ($T_x = 200$); and a represents the activation values passed in each time instant from one recurrent unit to the next one. These activation values a are propagated thru the network capturing

dependencies between the different time steps and then the last activation $a^{\langle T_x \rangle}$ is used to calculate the output of the network.

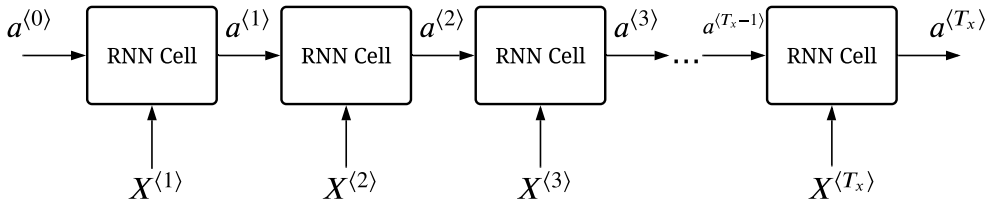


Figure 5. Basic RNN network.

The implementation of the proposed model and its training was done in Python using the Tensorflow library due to the fast and efficient way that this library gives in the implementation of deep learning models. However, in order to have a clear understanding of the models architecture, the next section presents the mathematical definitions for each component of the network.

2.3.1. Dense Layers

The dense layer represents the processing units that integrates a classic deep learning model. In the scientific literature it is common to find models that use linear functions to map an input to an specific output, however, in the case of signal processing where a specialized treatment of noise must be performed, such as when working with EMG signals as is done in this research, the literature recommends the use of activation functions of the hyperbolic tangent type. The vectorized implementation of the dense layer can be seen in Equation (2).

$$a^{\langle l \rangle} = tanh(W^{\langle l \rangle} a^{\langle l-1 \rangle} + b^{\langle l \rangle}); \qquad a^{\langle 0 \rangle} = X \tag{2}$$

where:
$a^{\langle t-1 \rangle}$ represents the previous layer activations. In the case of the first layer, they are equal to the input X which contains the value of each EMG channel.
$W^{\langle l \rangle}$ represents the current layer weights.
$b^{\langle l \rangle}$ represents the current layer bias.

2.3.2. Long Short-Term Memory (LSTM) Layer

The recurrent layer is made of Long Short-term Memory (LSTM) units, which is one of the state of the art units used in RNNs due to its capacity to memorize long term dependencies. Studies have shown that LSTM units outperform other state of the art recurrent units in complex dataset [61–64], making them a reliable choice for the recurrent layer architecture. The structure of a single LSTM unit can be seen in Figure 6.

Equations (3)–(8) show the vectorized implementation of each one of the items that make up the LSTM structure.

$$\tilde{c}^{\langle t \rangle} = tanh(W_c[a^{\langle t-1 \rangle}, X^{\langle t \rangle}] + b_c) \tag{3}$$

$$\Gamma_i^{\langle t \rangle} = tanh(W_i[a^{\langle t-1 \rangle}, X^{\langle t \rangle}] + b_i) \tag{4}$$

$$\Gamma_f^{\langle t \rangle} = tanh(W_f[a^{\langle t-1 \rangle}, X^{\langle t \rangle}] + b_f) \tag{5}$$

$$\Gamma_o^{\langle t \rangle} = tanh(W_o[a^{\langle t-1 \rangle}, X^{\langle t \rangle}] + b_o) \tag{6}$$

$$c^{\langle t \rangle} = \Gamma_i^{\langle t \rangle} * \tilde{c}^{(t)} + \Gamma_f^{\langle t \rangle} * c^{\langle t-1 \rangle} \tag{7}$$

$$a^{\langle t \rangle} = \Gamma_o^{\langle t \rangle} * tanh(c^{\langle t \rangle}) \tag{8}$$

where:
$X^{\langle t \rangle}$ is an input vector containing the value of each EMG channel or the activation values of the previous dense layer in time t.
$a^{\langle t-1 \rangle}$ represents the activations of the LSTM units in the previous time $t-1$.
$c^{\langle t-1 \rangle}$ represents the memory values in the previous time $t-1$; $a^{\langle t \rangle}$ are the activations in the current time t.
$c^{\langle t \rangle}$ are the new memory values for time t.
W_x represents the LSTM unit weights for each gate and b_x represents the LSTM unit bias for each gate.

It is important to note that $[a^{\langle t-1 \rangle}, X^{\langle t \rangle}]$ is a matrix form by the concatenation of the previous time activations $a^{\langle t-1 \rangle}$ and the current inputs $X^{\langle t \rangle}$.

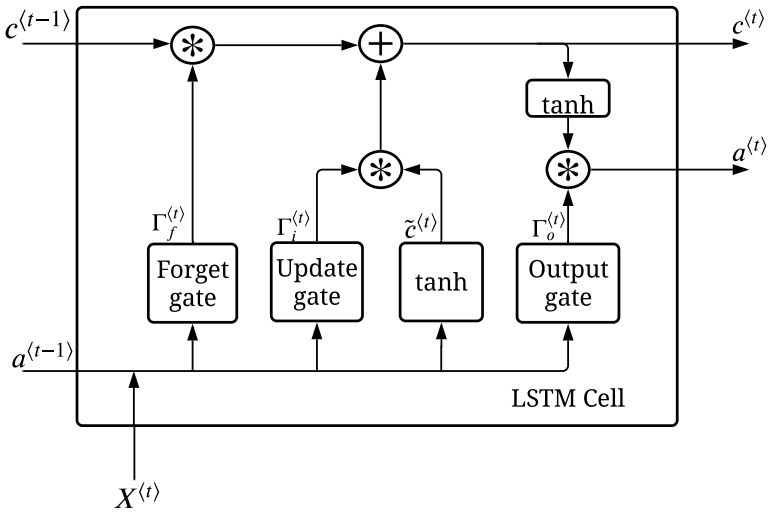


Figure 6. LSTM Cell structure.

The output layer integrates a SOFTMAX activation function to classify which gesture is being performed. The SOFTMAX activation function is one of the most widely used functions for multi-class classification problems because of its performance in predicting the probability of each class [55,65,66]. The SOFTMAX function predicts the probability of a class C_j given an input X as shown in Equation (9). Its mathematical expression can be seen in Equation (10).

$$\hat{y}_j = P(C_j|X) \tag{9}$$

$$\hat{y}_j = \frac{t_j}{\sum_{j=1}^{C} t_j}; \quad t_j = e^{a_j} \tag{10}$$

where:
$\hat{y}_j$ is the output probability for the class j.
C is the total number of classes that are being predicted.
a_j represents the linear combination of the weights and the previous layer activations, and can be defined as $a_j = Wa^{\langle l-1 \rangle}$.

Its important to note that the SOFTMAX output layer has the the same units or nodes as the number of classes that are being predicted.

2.4. Model Optimization

Once the model is defined, the next step was to select a cost function in order to optimize it through training and obtain the desired results in the classifier. In general, when

using an output layer with a SOFTMAX activation function, a categorical cross entropy loss function $\mathcal{L}(\hat{y}, y)$ is selected, which allows optimizing the cost function $\mathcal{J}(W, b)$ that sums the loss over the training examples. The categorical cross entropy loss allows a correct penalization of incorrect predictions and improve the performance in multi-class classification problems compared with other loss functions like the mean squared error (MSE). The categorical cross entropy loss and the cost function are shown in Equations (11) and (12).

$$\mathcal{L}(\hat{y}, y) = -\sum_{j=1}^{C} y_j log(\hat{y}_j) \tag{11}$$

$$\mathcal{J}(W, b) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}, y) \tag{12}$$

where:

$\hat{y}_j$ correspond to the models predicted probability for class j.
y_j represents the correct class (label) of the training example.
C is the total number of classes and m is the number of training examples in the mini batch.

It is important to note that the model was optimized with a mini batch gradient descent strategy. The batch size was set during the hyper-parameter tuning process along with the number of training epochs (Training iterations). The optimization algorithm will be explained in greater detail in the next paragraph.

In order to minimize the cost function $\mathcal{J}(W, b)$, it is necessary to update the weights W and biases b of each layer. This must be done by an optimization algorithm such as gradient descent. In this research, the Adaptive Moment Estimation ADAM [67] algorithm was selected; that is, a combination of Gradient Descent With Momentum and Root Mean Square Propagation (RMSProp). The ADAM optimization algorithm is considered to be one of the best optimization algorithms to train neural networks [68–70].

2.4.1. Hyperparameter Tuning

One of the key steps during the model training or model optimization, is the process known as hyper-parameter tuning. The hyper-parameter are values used to control the learning of the model, for example the model architecture, the number of training epochs (Training iterations), the learning rate α, among others. This process is performed with the objective of obtaining an optimal performance of the model [71–73], and correct issues like high bias and high variance.

The fundamental problem of hyper-parameter tuning consists in finding the sets of hyper-parameter λ that minimize the generalization error $\mathbb{E}_{(D_{train}, D_{val}) \sim D} V(\mathcal{L}, \mathcal{A}_\lambda, D_{train}, D_{val})$ of a machine learning algorithm $\mathcal{A}$ given a data set D (Divided in training D_{train} and validation D_{val}). Where the hyper-parameters λ belong to the configuration space $\wedge$ and V is the loss function of the algorithm $\mathcal{A}$ parametrized by λ ($\mathcal{A}_\lambda$). Equation (13) shows the mathematical expression of this optimization problem [71].

$$\lambda^* = \underset{\lambda \in \wedge}{argmin} \mathbb{E}_{(D_{train}, D_{val}) \sim D} V(\mathcal{L}, \mathcal{A}_\lambda, D_{train}, D_{val}) \tag{13}$$

The literature has proposed many techniques to address the problem of hyper-para meter tuning. The most widely used are: manual search, in which the hyper-parameters are tuned by hand with an iterative process; grid search, in which a finite set of values for the defined hyper-parameters are specified and then the combination of these values is evaluated; and random search [74], which follows the same principle of grid search but the set of values for the hyper-parameters is set randomly [71,73–75].

In this research, manual hyper-parameter tuning was performed due to its simplicity and low computational cost. This optimization was done by changing the following hyper-parameters: the learning rate α, the batch size, the number of training epochs and the model

architecture. With the future work in mind, in which the model might be embedded in a micro-controller to make the system portable; the architecture of the model started as simple as possible and the it was increased with more layers and hidden units in order to achieve the desired performance. The results of this process can be seen in greater detail in Section 3.1.

2.5. Metrics

Once the model is trained, it is important to define a way to determine its performance, this is when performance metrics come to play. There are many metrics to indicate the performance of a model, most of them based on the confusion matrix, which contains the values of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN), values that come in handy to calculate these metrics.

This research evaluated the performance of the gesture classifier using two metrics commonly used in the literature [76]. The first one was the accuracy, which can be mathematically defined as shown in Equation (14).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{14}$$

As Equation (14) shows, the accuracy measures how much the model is correctly predicting on the entire data-set and represents the probability that a prediction made by the model is correct. Although this is an easily interpretable measure, it comes with the issue that it does not consider the data-set class distribution and gives the same weight to each prediction, which tends to hide strong classification errors for classes with few examples, since those classes are less relevant compared to the biggest ones, thus, this metric can be easily misinterpreted and give a false sense of performance for the model. This downside in this metric, generally occurs when the model is trained or validated with imbalanced data-sets in which some classes have fewer examples than other classes.

To solve the aforementioned disadvantage in the accuracy metric, a second metric is proposed that allows to validate the performance of the model; the proposed metric is the Macro F1-Score (multi-class version of the F1-Score), which can be interpreted as a harmonic mean of the model precision and recall. It is important to note that, considering the gesture classifier has multiple classes, it is necessary to average the precision and recall across all classes in order to calculate the Macro F1-Score. Equations (15)–(17) shows the necessary calculations to find the metric.

$$\begin{aligned} Precision_j &= \frac{TP_j}{TP_j + FP_j} \\ Macro\ Precision &= \frac{\sum_{j=1}^{C} Precision_j}{C} \end{aligned} \tag{15}$$

$$\begin{aligned} Recall_j &= \frac{TP_j}{TP_j + FN_j} \\ Macro\ Recall &= \frac{\sum_{j=1}^{C} Recall_j}{C} \end{aligned} \tag{16}$$

$$Macro\ F1\text{-}Score = 2\left(\frac{Macro\ Precision * Macro\ Recall}{Macro\ Precision + Macro\ Recall}\right) \tag{17}$$

As Equations (15)–(17) show, the precision and recall are calculated for each class and then averaged across the C classes of the classifier.

The Macro F1-Score is useful to support the accuracy metric and validate the model performance even if the data-set is imbalanced. As mentioned earlier, the F1-Score requires the model precision, which is a measure of the model's quality. It indicates if the model returns more relevant predictions (TP) than irrelevant ones (FP); on the other hand, the F1-Score also requires the model recall, which is a measure of the model's quantity. It

indicates the capacity of the model to return most of the relevant predictions (TP) without considering if irrelevant predictions are also returned. With this in mind, the F1-Score, being the harmonic mean between the precision and recall, gives equal weight to both metrics. Thus if the F1-Score is high, both metrics will also be high, more relevant predictions and among all positive predictions most of them are relevant. Moreover, considering that the F1-Score is based on the average metrics for each class, it gives the same weight to all classes, taking them all into consideration independently if some classes have less examples than other classes, thus, giving a better estimate of the model performance.

2.6. Experimental Validation

In order to analyze and evaluate the proposed model two main phases were performed. An initial phase where the model was trained and tuned according to the concepts mentioned in Section 2.4; and a second phase where the best performing models of the first phase, were tested in real time gesture classification. Both experimental stages validated the model performance using the metrics seen in Section 2.5.

In the first phase, training and tuning, the training/validation data-set was constructed as mentioned in the experimental set-up (Section 2.1. Once the data sets were formed, the next step was to train and tune the model, for this, a hyper-parameter tuning process was performed (Section 2.4.1). In this process, a total of six different LSTM-RNN architectures were proposed following the general architecture presented in Section 2.3; in this part, the LSTM-RNN architecture was increased in size in each one of the models, the first three proposed model consisted in only one LSTM layer and the last three were hybrid models composed of Dense and LSTM layers; this was done in order to increase the size of the network gradually until an optimal performance architecture was found, aiming to find a good trade off between model size and performance; this trade off is a key part to embedding the model into a micro-controller in the future. During the hyper-parameter tuning, a manual search was used to set the learning rate α, the batch size and the number of training epochs. In this first stage, the trained models were evaluated with the metrics proposed in Section 2.5. For a model to be considered for the second phase (real time validation), it was required that it had a training accuracy $\geq$98% as well as a training F1-Score $\geq$98%; the model must also hve a difference $\leq$1% between training and validation for both the accuracy and F1-Score in order for it to not suffer from high variance.

In a second phase, a real time validation was carried out. This validation is a fundamental step, because it allows to identify flaws in the gesture recognition system and measure the performance of the model dynamically. To tackle this, the candidate LSTM-RNN models selected in the previous phase were evaluated in four sessions using real time EMG acquisition using the EMG armband, in this sessions each gesture was performed for 20 s. Taking into consideration the possible reduction in accuracy due to changes in the EMG armband position, thus, changes in the data distribution, the candidate models (pre-trained) were re-trained after the initial training phase using data of another two EMG recording sessions, where the EMG armband was taken off and put back on the arm between sessions. Each session the EMG armband was put approximately in the same position; this was done in order to take into account the variations in data distribution due to the small changes in the EMG armband position.

For this last phase, the re-trained models were evaluated using the metrics proposed in Section 2.5. In this phase, one of the candidate models was selected as the final LSTM-RNN gesture classifier. The conditions for the selection of this model were that it had an accuracy and F1-Score $\geq$ 80%. It is important to note that the real time target metrics are lower compared to the training/validation targets, because in this phase the sensitivity of the EMG acquisition armband and its position on the arm, which varies across different real time recording sessions, are variables that have an impact on the performance of the algorithm [40,77–79].

3. Results and Discussion

This section presents the research results based on the methodology presented in Section 2. The results are presented in three parts; Section 3.1 presents the training and hyper-parameter tuning of the gesture classifier, the performance across testing and validation sets; Section 3.2, presents the performance of the candidate models during real-time testing, and Section 3.3 presents the selected model, its architecture and detail performance during training and testing.

3.1. Recurrent Neural Network (RNN) Model Training

As mentioned in Section 2.4.1, multiple network configurations were proposed in order to find an optimal LSTM-RNN architecture for the classifier. A total of six LSTM-RNN models were trained, Figure 7 shows the learning curves of the six LSTM-RNN models, presenting their loss during the training process. It is important to note, that during the manual search in the hyper-parameter tuning process, it was found that setting the learning rate α to 0.0001, the batch size to 64 and the number of training epochs to 1000, allowed a better performance and stability during training and validation.

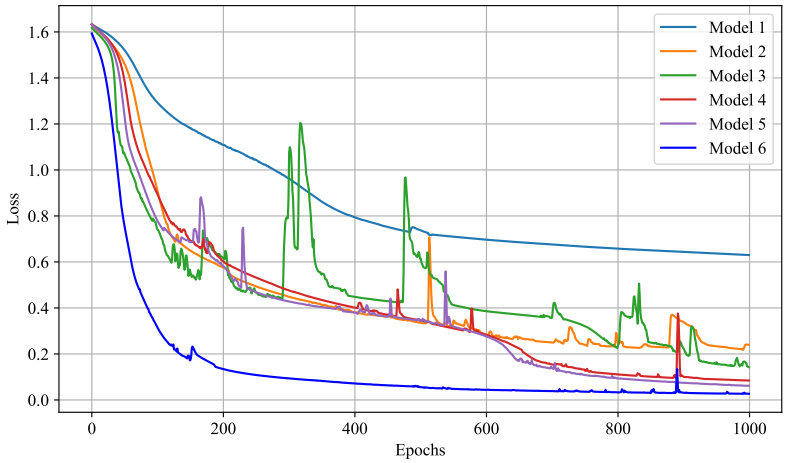


Figure 7. Evolution of different LSTM-RNN architectures losses during the training training process.

Analyzing Figure 7 it can be seen how models 1, 2 and 3 under-perform when compared to the rest of the LSTM-RNN models. It can also be noted how models 2 and 3 present high instability during training, thus making it difficult for them to converge to an optimal minimum; this instability can be due to their inability to generalize and learn the patterns in the data. In contrast, models 4, 5 and 6 presented stable convergence to a minimum in the loss function, especially model 6, which had the most stable training process and its loss value was the lowest of all models.

The final performance of the LSTM-RNN models across the training and validation sets can be seen in Table 1. This table contains in its first column the LSTM-RNN model number; the second column presents the architecture of the LSTM-RNN model indicating its layers and number of units following the format <*layer name* (*number of units*)>; the third and fourth column presents the final accuracy and macro F1-Score on the training set, and the fifth and sixth column presents the final accuracy and macro F1-Score on the validation set.

As Table 1 shows, models 1 and 2 under-performed (have high bias or Under-fitting) when compared to the rest of the LSTM-RNN models, they also suffered high variance (Over-fitting), thus, making them not suitable for the classifier. In the case of model 3, even though it presented an accuracy superior to 95%, it was not close to the accuracy presented by models 4, 5 and 6. This can also be validated looking at the macro F1-Score. Conversely, models 4, 5 and 6 presented an accuracy superior to 98% across both training and validation

sets. It was the same case for their macro F1-Score; making them suitable for testing the classifier in real time.

Table 1. LSTM-RNN models performance in training and validation sets during the hyperparameter tuning.

Model	Architecture	Training		Validation	
		Accuracy	F1-Score	Accuracy	F1-Score
1	LSTM (8)	0.6575	0.6173	0.6000	0.5392
2	LSTM (16)	0.9075	0.9068	0.8500	0.8360
3	LSTM (32)	0.9700	0.9700	0.9600	0.9589
4	Dense (8) LSTM (16) Dense (8)	0.9850	0.9849	0.9900	0.9894
5	Dense (16) LSTM (16) Dense (16)	0.9875	0.9874	0.9800	0.9798
6	Dense (32) LSTM (16) Dense (32)	0.9925	0.9925	0.9900	0.9896

3.2. Real Time Testing

Once models 4, 5 and 6 were selected as candidates for the gesture classifier, real time testing was performed according to the proposed validation in Section 2.6. Tables 2 and 3 show the accuracy and F1-Score of the selected models obtained in different real time testing sessions along with their standard deviation (σ), Table 4 shows the summary of Tables 2 and 3.

Table 2. Accuracy of LSTM-RNN candidate models during real time testing across four different acquisition sessions.

Model	Accuracy in Session				Average	SD (σ)
	1	2	3	4		
4	0.7500	0.8443	0.8521	0.7456	0.7980	0.0581
5	0.8724	0.7455	0.7895	0.8406	0.8120	0.0559
6	0.8640	0.9660	0.7979	0.8640	0.8729	0.0694

Table 3. F1-Score of LSTM-RNN candidate models during real time testing across four different acquisition sessions.

Model	F1-Score in Session				Average	SD (σ)
	1	2	3	4		
4	0.6555	0.7548	0.8023	0.7054	0.7295	0.0632
5	0.8720	0.7465	0.7854	0.8401	0.8110	0.0559
6	0.8526	0.9660	0.7377	0.8473	0.8509	0.0932

Looking at Table 4 it can be seen that LSTM-RNN model 6 performed the best among the candidates during the real time application of the classifier. It is important to note that this model ended up being extremely sensitive. Looking at Tables 2 and 3 it can be seen that the models vary their performance across different acquisition sessions. This sensitivity was caused by the variation in the data distribution across training/validation and real time testing sessions, this change in data distribution is mainly caused by variations in

the EMG armband location; this model sensitivity to position variations of the acquisition device is one of the current challenges when working with EMG signals and real time EMG classifiers [40,77–79]. In order to try to solve this, the three candidate models were fed with more data and trained again for 150 epochs, aiming to make the models more robust and more capable of generalizing to new data; however, this approach failed and the three models presented similar accuracy to the one shown in Table 4.

Table 4. LSTM-RNN candidate models performance summary during real time classification.

Model	Accuracy	F1-Score
4	0.7980 ± 0.0581	0.7295 ± 0.0632
5	0.8120 ± 0.0559	0.8110 ± 0.0559
6	0.8729 ± 0.0694	0.8509 ± 0.0932

3.3. Selected Model Training/Validation and Real Time Testing Details

According to the results showed in Sections 3.1 and 3.2, the LSTM-RNN model that achieved the best performance was model 6. This is because of all models, it presented the best performance across training, validation and real-time testing. It also proved to be more stable during training and the more capable generalizing new data. Figure 8 shows the training process of this particular LSTM-RNN model. This figure includes both the training and validation loss across the 1000 training epochs. As can be seen in the figure, both curves converge to approximately to the same value, meaning that the model does nt suffer from high variance (over-fitting) on training and validation data. It also does not suffer from high bias (Under-fiting) because of its 99% accuracy; however, taking in consideration the results and conclusions found in Section 3.2, it is important to note that the model does suffer from high variance on training/validation and real time testing data. This high variance is caused by the difficulty in reproducing the same armband location when acquiring the EMG signals across different testing sessions.

The selected LSTM-RNN architecture can be seen in Figure 9. As proposed in the methodology the model is based on recurrent neural networks and is composed of two dense layers, one recurrent LSTM layer and an output layer with SOFTMAX activation function; the model has a total of 4005 parameters and ended up with a file size of 89 KB.

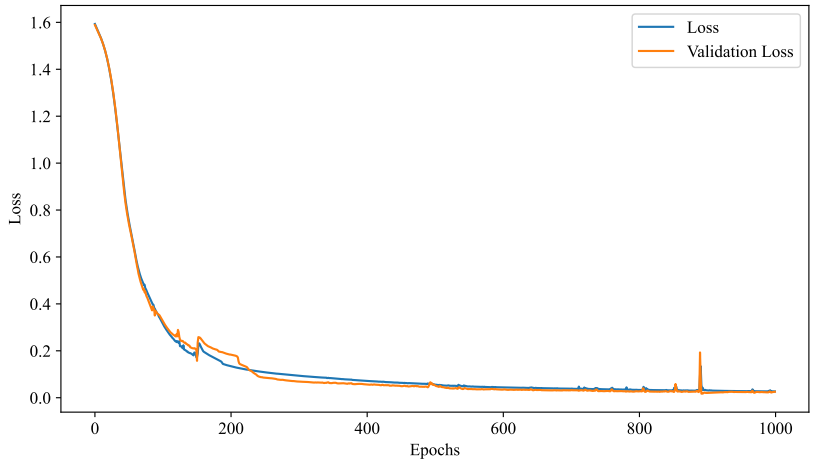


Figure 8. Loss in training and validation of selected LSTM-RNN model.

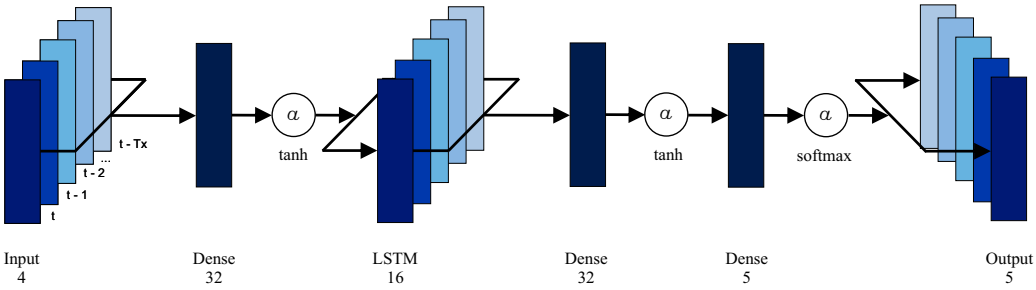
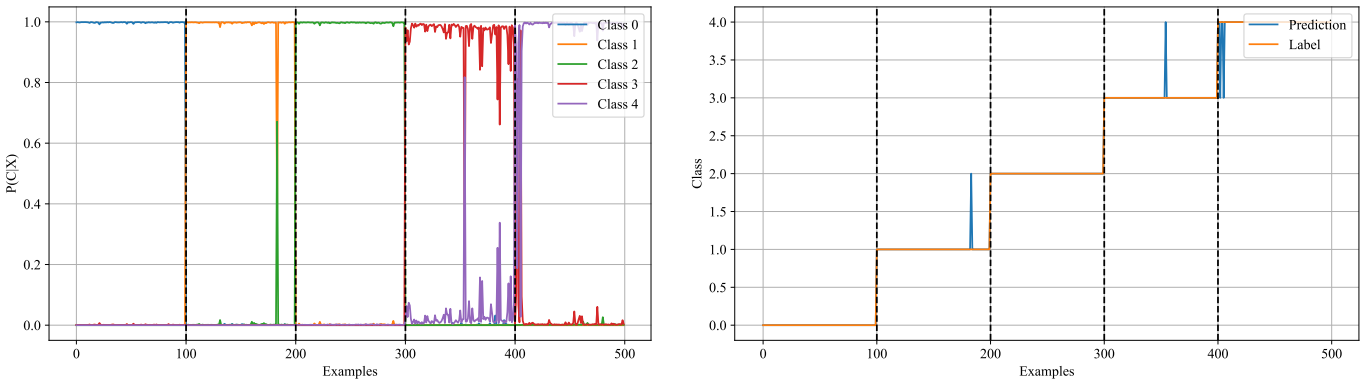


Figure 9. Selected LSTM-RNN architecture.

Figure 10 shows the prediction of the LSTM-RNN model on the training/validation recording. Figure 10a presents the network output probability of a class C_j given an input X. Figure 10b shows the actual predicted class and the actual label of the example. The recording was set to have the gestures in order.



(**a**) (**b**)

Figure 10. Predictions of the selected LSTM-RNN model in training and validation sets. (**a**) Probability of class; (**b**) Class predictions.

Figure 11 shows the predictions of the LSTM-RNN model on the four real time testing sessions. This figure presents the network output probability of a class C_j given an input X and shows the actual predicted class and the actual label of the example in the four real time testing sessions. The recording was set to have the gestures in order.

Figures 10 and 11 allow to get a better intuition about the results shown in Tables 2 and 3. Figure 10 shows how the output probability of the LSTM-RNN model is stable when predicting each class, except for some wrong predictions. This was expected because the LSTM-RNN model achieved a 0.99 accuracy in the training/validation data-sets. In contrast, Figure 11 shows a more erratic behavior in the output probability of the LSTM-RNN model during real time classification, especially in classes with more wrong predictions. It can be seen how the LSTM-RNN model tends to fail in classifying one of the classes even tho it generalized and predicts correctly the other four, this figure shows how the model tends to fail either in class 1 or class 4. Figure 11 depicts the sensitivity of the model when the acquisition system is taken off and put back on, thus its varying accuracy across different acquisition sessions.

Based on the above, an ideal model should exhibit a behavior in the output probability as the one seen in Figure 10, however, when implementing the system in real-time, there are variables that change the model performance across different acquisition sessions. As mentioned earlier, small changes in the EMG acquisition device and noise in the EMG signal are the ones with grater impact and cause the model to start confusing classes as Figure 11 shows.

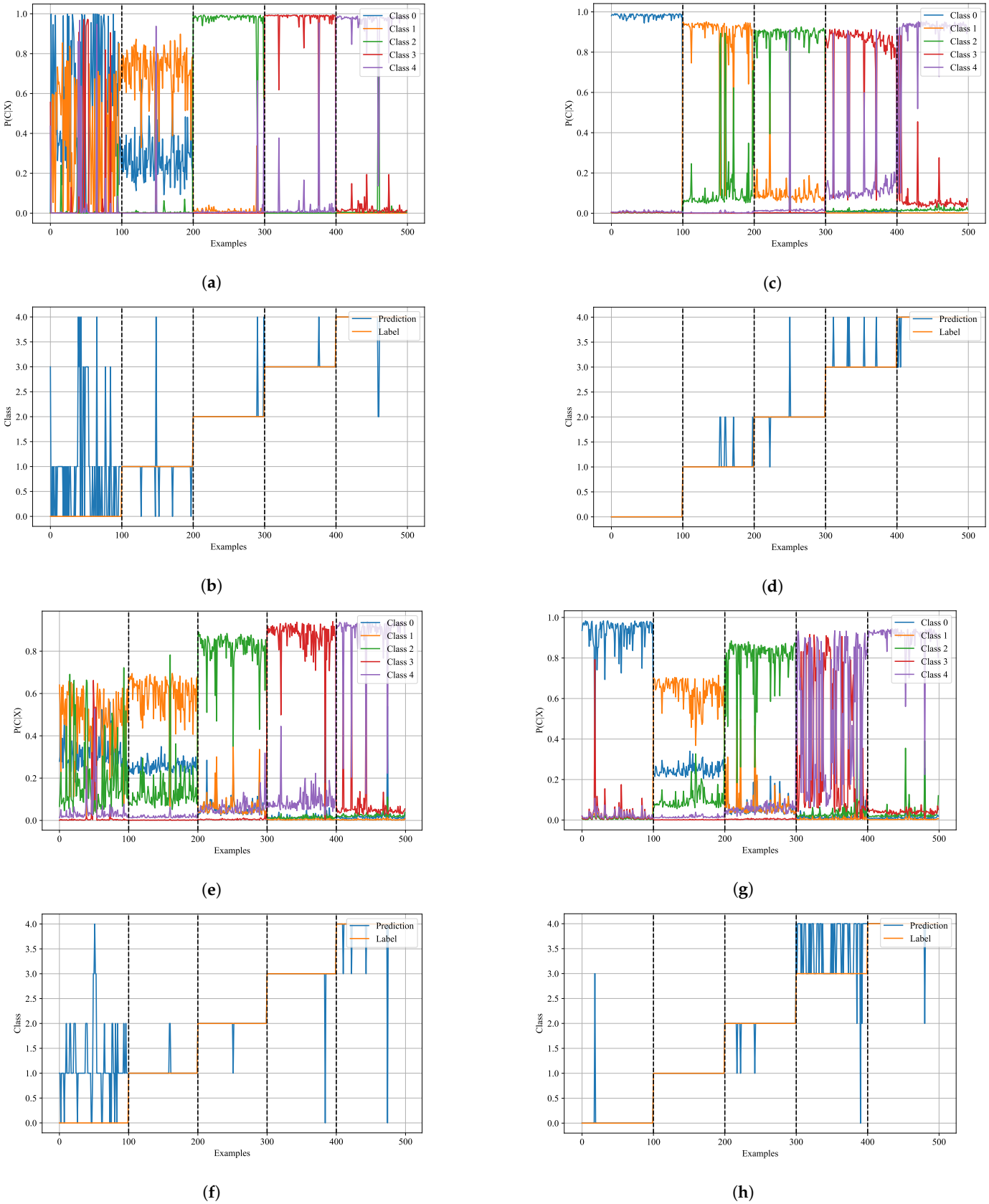


Figure 11. Predictions of the selected LSTM-RNN model in real time classification across different sessions. (**a**) $P(C_j|X)$ in session 1; (**b**) Class predictions in session 1; (**c**) $P(C_j|X)$ in session 2; (**d**) Class predictions in session 2; (**e**) $P(C_j|X)$ in session 3; (**f**) Class predictions in session 3; (**g**) $P(C_j|X)$ in session 4; (**h**) Class predictions in session 4.

3.4. Model Comparison

This work presented a LSTM-RNN model composed of an LSTM layer wrapped by dense layers, capable of performing in real-time. Table 5 shows the comparison of the proposed LSTM-RNN model with other state-of-the-art RNN models found in the literature; the comparison included the number of EMG channels used, which is an important parameter when performing real-time operation, the classifier type and the test accuracy achieved by the classifier. It is important to note that most of these studies tested the model on static data and not in real-time.

Table 5. Comparison of the proposed LSTM-RNN model with state-of-the-art RNN models found in literature.

Work	Channels	Classifier	Test Accuracy
Samadani [36]	12	Stacked LSTM	89.50%
Jabbari et al. [35]	8	BILSTM-AT	$91.40 \pm 3.00\%$
Zhang et al. [35]	8	Stacked GRU	89.60%
Jiang et al. [39]	8	Stacked LSTM	97.10%
Xie et al. [41]	16	LSTM	78.13%
He et al. [42]	12	LSTM-Dense	75.50%
Hu et al. [37]	10	CNN-RNN	87.00%
Ours	**4**	**LSTM-Dense**	$\mathbf{87.29 \pm 6.94}\%$

As can be seen in Table 5 the proposed LSTM-RNN model was capable of achieving state-of-the-art performance while using a much simpler model and less number of EMG channels (only four EMG channels); reducing both the complexity of the model and the number of EMG channels is key when working on a classifier that must be embedded in a portable device such as a hand prosthesis or a wearable Human–Machine Interface. Another key takeaway from the comparison is that most of the mentioned state-of-the-art RNN models where tested only on static data and online datasets, which does not guarantee the performance in real-time operation; in contrast, the proposed LSTM-RNN model was tested in real-time and achieved a comparable accuracy score with other state-of-the-art RNN models.

4. Conclusions and Future Work

The proposed LSTM-RNN model for gesture classification proved to be a suitable approach for gesture recognition using EMG signals. The LSTM-RNN model was able to obtain high accuracy during training and validation phase; indicating that the model was able to learn the patterns in the EMG signals and generalize on new data. Furthermore, it is important to highlight that the model is small enough to be embedded in a micro-controller, which is fundamental to implement it for a commercial or industrial application. The proposed LSTM-RNN model was able to perform at a similar level of accuracy than other state-of-the-art RNN models while reducing the model complexity and the number of EMG channels, increasing its potential to be embedded and its commercial scalability. With this in mind, the proposed model shows promising results for its industrial or commercial application, even though further improvements must be made to achieve this development state, particularly in the EMG acquisition system. This research allowed to go beyond static data classification, giving valuable insights regarding the challenges of real time classification.

It was found that the reduced performance during real-time testing was due to the high sensitivity of the model to small changes in the EMG armband position and noise present in the signal when acquiring the EMG. However, although the model accuracy varies with different postures of the acquisition system, it is still able to generalize to new data. With this in mind, having a better EMG acquisition device and a better acquisition

process, where noise and variations in position across different sessions are minimized, will probably improve the model performance considerably, without the need of changing the model.

In general, the proposed approach using deep neural models its effective for finding patterns in stochastic signals like EMG signals, making deep learning models a great tool for gesture recognition tasks based on this type of signals. It is important to note that in order to build an effective and robust gesture recognition system, not only is important to build and optimize a good model, it is also fundamental to acquire quality data and perform a good pre-processing of it. It is only when an optimal point between data and model is found that the overall system can find the best performance.

In future work the EMG acquisition armband will be upgraded with the objective of acquiring more data and improving its quality; new pre-processing and feature extraction techniques will be tried, aiming to reduce the system sensitivity to noise in the EMG signal and variations in the acquisition devise; moreover, the gesture recognition model will be optimized with the new data and it will be embedded in a microcontroller in order to control an active hand prosthesis using different hand gestures.

Author Contributions: The authors' contributions to the achievement of the research are as follows: Model design, conceptualisation, computational modelling and Analysis of Results, A.T.-O., J.J.-T.; Analysis of Results, processing and review, J.C.T., A.P., A.L.-G.; Analysis of Results and review R.A.C. All authors have read and agreed to the published version of the manuscript.

Funding: The project is funded under the program of the Minister of Education and Science titled "Regional Initiative of Excellence" in 2019–2023, project number 018/RID/2018/19, the amount of funding PLN 10 788 423,16. The research was funded by the Universidad EIA grant number CO12021002 and Universidad Iberoamericana grant number Ibero DCI-002847.

Institutional Review Board Statement: The study was conducted according to the guidelines of the Declaration of Helsinki, and approved by the Ethics Committee of EIA University (protocol code P201912-04 and date of approval)approved by EIA University Ethical Committee (Code P201912-04).

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: The data that support the findings of this study are available from the corresponding author, J.C.T. (juan.tejada@eia.edu.co), upon reasonable request.

Acknowledgments: This research was supported by the Universidad EIA (CO12021002) and Universidad Iberoamericana (Ibero DCI-002847 and Convocatoria 14 DINV). Thanks to Universidad EAFIT and WSB University.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Haria, A.; Subramanian, A.; Asokkumar, N.; Poddar, S.; Nayak, J.S. Hand Gesture Recognition for Human Computer Interaction. In *Proceedings of the Procedia Computer Science*; Elsevier: Amsterdam, The Netherlands, 2017; Volume 115, pp. 367–374. [CrossRef]
2. Lv, Z.; Xiao, F.; Wu, Z.; Liu, Z.; Wang, Y. Hand gestures recognition from surface electromyogram signal based on self-organizing mapping and radial basis function network. *Biomed. Signal Process. Control.* **2021**, *68*, 102629. [CrossRef]
3. Shi, W.T.; Lyu, Z.J.; Tang, S.T.; Chia, T.L.; Yang, h.Y. A bionic hand controlled by hand gesture recognition based on surface EMG signals: A preliminary study. *Biocybern. Biomed. Eng.* **2018**, *38*, 126–135. [CrossRef]
4. Pisharady, P.K.; Saerbeck, M. Recent methods and databases in vision-based hand gesture recognition: A review. *Comput. Vis. Image Underst.* **2015**, *141*, 152–165. [CrossRef]
5. Arunraj, M.; Srinivasan, A.; Arjunan, S.P. A Real-Time Capable Linear Time Classifier Scheme for Anticipated Hand Movements Recognition from Amputee Subjects Using Surface EMG Signals. *IRBM* **2020**. [CrossRef]
6. Boyali, A.; Hashimoto, N. Spectral Collaborative Representation based Classification for hand gestures recognition on electromyography signals. *Biomed. Signal Process. Control.* **2016**, *24*, 11–18. [CrossRef]
7. Jaber, H.A.; Rashid, M.T.; Fortuna, L. Online myoelectric pattern recognition based on hybrid spatial features. *Biomed. Signal Process. Control.* **2021**, *66*, 102482. [CrossRef]
8. Mendes Souza, G.C.; Moreno, R.L. Netlab MLP - Performance Evaluation for Pattern Recognition in Myoletric Signal. In *Proceedings of the Procedia Computer Science*; Elsevier: Amsterdam, The Netherlands, 2018; Volume 130, pp. 932–938. [CrossRef]

9. Kowdiki, M.; Khaparde, A. Automatic hand gesture recognition using hybrid meta-heuristic-based feature selection and classification with Dynamic Time Warping. *Comput. Sci. Rev.* **2021**, *39*, 100320. [CrossRef]
10. Jochumsen, M.; Waris, A.; Kamavuako, E.N. The effect of arm position on classification of hand gestures with intramuscular EMG. *Biomed. Signal Process. Control.* **2018**, *43*, 1–8. [CrossRef]
11. Cifuentes, J.; Pham, M.T.; Moreau, R.; Boulanger, P.; Prieto, F. Medical gesture recognition using dynamic arc length warping. *Biomed. Signal Process. Control.* **2019**, *52*, 162–170. [CrossRef]
12. Licsár, A.; Szirányi, T. User-adaptive hand gesture recognition system with interactive training. *Image Vis. Comput.* **2005**, *23*, 1102–1114. [CrossRef]
13. Simão, M.; Neto, P.; Gibaru, O. EMG-based online classification of gestures with recurrent neural networks. *Pattern Recognit. Lett.* **2019**, *128*, 45–51. [CrossRef]
14. Just, A.; Marcel, S. A comparative study of two state-of-the-art sequence processing techniques for hand gesture recognition. *Comput. Vis. Image Underst.* **2009**, *113*, 532–543. [CrossRef]
15. Salim, U.T.; Dawwd, S.A. Systolic hand gesture recognition/detection system based on FPGA with multi-port BRAMs. *Alex. Eng. J.* **2019**, *58*, 841–848. [CrossRef]
16. Li, Y.; Wang, X.; Liu, W.; Feng, B. Deep attention network for joint hand gesture localization and recognition using static RGB-D images. *Inf. Sci.* **2018**, *441*, 66–78. [CrossRef]
17. Ovur, S.E.; Zhou, X.; Qi, W.; Zhang, L.; Hu, Y.; Su, H.; Ferrigno, G.; De Momi, E. A novel autonomous learning framework to enhance sEMG-based hand gesture recognition using depth information. *Biomed. Signal Process. Control.* **2021**, *66*, 102444. [CrossRef]
18. Huang, Y.; Yang, J. A multi-scale descriptor for real time RGB-D hand gesture recognition. *Pattern Recognit. Lett.* **2021**, *144*, 97–104. [CrossRef]
19. Ameur, S.; Ben Khalifa, A.; Bouhlel, M.S. A novel hybrid bidirectional unidirectional LSTM network for dynamic hand gesture recognition with Leap Motion. *Entertain. Comput.* **2020**, *35*, 100373. [CrossRef]
20. Pinzón-Arenas, J.O.; Jiménez-Moreno, R.; Rubiano, A. Percentage estimation of muscular activity of the forearm by means of EMG signals based on the gesture recognized using CNN. *Sens. Bio-Sens. Res.* **2020**, *29*, 100353. [CrossRef]
21. Suk, H.I.; Sin, B.K.; Lee, S.W. Hand gesture recognition based on dynamic Bayesian network framework. *Pattern Recognit.* **2010**, *43*, 3059–3072. [CrossRef]
22. Lazarou, M.; Li, B.; Stathaki, T. A novel shape matching descriptor for real-time hand gesture recognition. *Comput. Vis. Image Underst.* **2021**, *210*, 103241. [CrossRef]
23. Escobedo Cardenas, E.J.; Chavez, G.C. Multimodal hand gesture recognition combining temporal and pose information based on CNN descriptors and histogram of cumulative magnitudes. *J. Vis. Commun. Image Represent.* **2020**, *71*, 102772. [CrossRef]
24. Sun, Y.; Xu, C.; Li, G.; Xu, W.; Kong, J.; Jiang, D.; Tao, B.; Chen, D. Intelligent human computer interaction based on non redundant EMG signal. *Alex. Eng. J.* **2020**, *59*, 1149–1157. [CrossRef]
25. Noce, E.; Dellacasa Bellingegni, A.; Ciancio, A.L.; Sacchetti, R.; Davalli, A.; Guglielmelli, E.; Zollo, L. EMG and ENG-envelope pattern recognition for prosthetic hand control. *J. Neurosci. Methods* **2019**, *311*, 38–46. [CrossRef]
26. Fatimah, B.; Singh, P.; Singhal, A.; Pachori, R.B. Hand movement recognition from sEMG signals using Fourier decomposition method. *Biocybern. Biomed. Eng.* **2021**, *41*, 690–703. [CrossRef]
27. Barron, O.; Raison, M.; Gaudet, G.; Achiche, S. Recurrent Neural Network for electromyographic gesture recognition in transhumeral amputees. *Appl. Soft Comput. J.* **2020**, *96*, 106616. [CrossRef]
28. Topalović, I.; Graovac, S.; Popović, D.B. EMG map image processing for recognition of fingers movement. *J. Electromyogr. Kinesiol.* **2019**, *49*, 102364. [CrossRef]
29. Premaratne, P.; Yang, S.; Vial, P.; Ifthikar, Z. Centroid tracking based dynamic hand gesture recognition using discrete Hidden Markov Models. *Neurocomputing* **2017**, *228*, 79–83. [CrossRef]
30. Chen, C.; Yu, Y.; Ma, S.; Sheng, X.; Lin, C.; Farina, D.; Zhu, X. Hand gesture recognition based on motor unit spike trains decoded from high-density electromyography. *Biomed. Signal Process. Control.* **2020**, *55*, 101637. [CrossRef]
31. Jiang, S.; Gao, Q.; Liu, H.; Shull, P.B. A novel, co-located EMG-FMG-sensing wearable armband for hand gesture recognition. *Sens. Actuators A Phys.* **2020**, *301*, 111738. [CrossRef]
32. Jiang, X.; Merhi, L.K.; Xiao, Z.; Menon, C. Exploration of Force Myography and surface Electromyography in hand gesture classification. *Med. Eng. Phys.* **2017**, *41*, 63–73. [CrossRef]
33. Boulila, W.; Ghandorh, H.; Khan, M.A.; Ahmed, F.; Ahmad, J. A novel CNN-LSTM-based approach to predict urban expansion. *Ecol. Inform.* **2021**, *64*, 101325. [CrossRef]
34. Saleh, N.M.; Zaini, N.; Latip, M.F.A. Recurrent neural networks-gated recurrent unit for behaviour analysis and prediction modelling to trigger high energy-consumption alert. In Proceedings of the 2019 IEEE 9th International Conference on System Engineering andvTechnology (ICSET), Shah Alam, Malaysia, 7 October 2019; pp. 493–498. [CrossRef]
35. Jabbari, M.; Khushaba, R.N.; Nazarpour, K. EMG-Based Hand Gesture Classification with Long Short-Term Memory Deep Recurrent Neural Networks. In Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS), Montreal, QC, Canada, 20–24 July 2020; pp. 3302–3305. [CrossRef]

36. Samadani, A. Gated Recurrent Neural Networks for EMG-Based Hand Gesture Classification. A Comparative Study. In Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS), Honolulu, HI, USA, 18–21 July 2018; pp. 1–4. [CrossRef]
37. Geng, W.; Hu, Y.; Wong, Y.; Wei, W.; Du, Y.; Kankanhalli, M. A novel attention-based hybrid CNN-RNN architecture for sEMG-based gesture recognition. *PLoS ONE* **2018**, *13*, e0206049. [CrossRef]
38. Zhang, Z.; He, C.; Yang, K. A Novel Surface Electromyographic Signal-Based Hand Gesture Prediction Using a Recurrent Neural Network. *Sensors* **2020**, *20*, 3994. [CrossRef] [PubMed]
39. Jiang, Y.; Song, L.; Zhang, J.; Song, Y.; Yan, M. Multi-Category Gesture Recognition Modeling Based on sEMG and IMU Signals. *Sensors* **2022**, *22*, 5855. [CrossRef] [PubMed]
40. Bi, L.; Feleke, A.; Guan, C. A review on EMG-based motor intention prediction of continuous human upper limb motion for human-robot collaboration. *Biomed. Signal Process. Control.* **2019**, *51*, 113–127. [CrossRef]
41. Xie, B.; Li, B.; Harland, A. Movement and Gesture Recognition Using Deep Learning and Wearable-sensor Technology. *ACM Int. Conf. Proc. Ser.* **2018**, 26–31. [CrossRef]
42. He, Y.; Fukuda, O.; Bu, N.; Okumura, H.; Yamaguchi, N. Surface EMG Pattern Recognition Using Long Short-Term Memory Combined with Multilayer Perceptron. In Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Honolulu, HI, USA, 18–21 July 2018; pp. 5636–5639. [CrossRef]
43. Nordin, M.; Frankel, V.H. *Basic Biomechanics of the Musculoskeletal System*, 4th ed.; Lippincott Williams & Wilkins: Baltimore, MD, USA, 2012; p. 454.
44. Toro Ossaba, A.; Jaramillo Tigreros, J.J.; Tejada Orjuela, J.C. Open Source Multichannel EMG Armband design. In Proceedings of the 2020 IX International Congress of Mechatronics Engineering and Automation (CIIMA), Cartagena, Colombia, 4–6 November 2020. [CrossRef]
45. Asogbon, M.G.; Samuel, O.W.; Jiang, Y.; Wang, L.; Geng, Y.; Sangaiah, A.K.; Chen, S.; Fang, P.; Li, G. Appropriate Feature Set and Window Parameters Selection for Efficient Motion Intent Characterization towards Intelligently Smart EMG-PR System. *Symmetry* **2020**, *12*, 1710. [CrossRef]
46. Merletti, R.; Farina, D. *Surface Electromyography: Physiology, Engineering and Applications*; Wiley: Hoboken, NJ, USA, 2016; pp. 1–570. [CrossRef]
47. Cipriani, C.; Controzzi, M.; Carrozza, M.C. Objectives, criteria and methods for the design of the SmartHand transradial prosthesis. *Robotica* **2010**, *28*, 919–927. [CrossRef]
48. Ventimiglia, P.; Padir, T.; Schaufeld, J. Design of a Human Hand Prosthesis. A Major Qualifying Project Report. Bachelor's Thesis, Faculty of the Worcester Polytechnic Institute, Worcester, MA, USA, 2012.
49. Belter, J.T.; Segil, J.L.; Dollar, A.M.; Weir, R.F. Mechanical design and performance specifications of anthropomorphic prosthetic hands: A review. *J. Rehabil. Res. Dev.* **2013**, *50*, 599–618. [CrossRef]
50. Cordella, F.; Ciancio, A.L.; Sacchetti, R.; Davalli, A.; Cutti, A.G.; Guglielmelli, E.; Zollo, L. Literature review on needs of upper limb prosthesis users. *Front. Neurosci.* **2016**, *10*, 209. [CrossRef]
51. Smith, L.H.; Hargrove, L.J.; Lock, B.A.; Kuiken, T.A. Determining the optimal window length for pattern recognition-based myoelectric control: Balancing the competing effects of classification error and controller delay. *IEEE Trans. Neural Syst. Rehabil. Eng.* **2011**, *19*, 186–192. [CrossRef]
52. Phinyomark, A.; Phukpattaranont, P.; Limsakul, C. Feature reduction and selection for EMG signal classification. *Expert Syst. Appl.* **2012**, *39*, 7420–7431. [CrossRef]
53. Moin, A.; Zhou, A.; Rahimi, A.; Benatti, S.; Menon, A.; Tamakloe, S.; Ting, J.; Yamamoto, N.; Khan, Y.; Burghardt, F.; et al. An EMG Gesture Recognition System with Flexible High-Density Sensors and Brain-Inspired High-Dimensional Classifier. In Proceedings of the 2018 IEEE International Symposium on Circuits and Systems (ISCAS), Florence, Italy, 27–30 May 2018.
54. Jaramillo-Yánez, A.; Benalcázar, M.E.; Mena-Maldonado, E. Real-time hand gesture recognition using surface electromyography and machine learning: A systematic literature review. *Sensors* **2020**, *20*, 2467. [CrossRef]
55. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
56. Smith, S.W. *The Scientist and Engineer's Guide to Digital Signal Processing*, 2nd ed.; California Technical Publishing: San Diego, CA, USA, 1999; p. 640.
57. Weerakody, P.B.; Wong, K.W.; Wang, G.; Ela, W. A review of irregular time series data handling with gated recurrent neural networks. *Neurocomputing* **2021**, *441*, 161–178. [CrossRef]
58. Wang, Y.; Perry, M.; Whitlock, D.; Sutherland, J.W. Detecting anomalies in time series data from a manufacturing system using recurrent neural networks. *J. Manuf. Syst.* **2020**. [CrossRef]
59. Yu, Y.; Si, X.; Hu, C.; Zhang, J. A review of recurrent neural networks: LSTM cells and network architectures. *Neural Comput.* **2019**, *31*, 1235–1270. [CrossRef]
60. Hewamalage, H.; Bergmeir, C.; Bandara, K. Recurrent Neural Networks for Time Series Forecasting: Current status and future directions. *Int. J. Forecast.* **2021**, *37*, 388–427. [CrossRef]
61. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv* **2014**, arXiv:1412.3555.
62. Józefowicz, R.; Zaremba, W.; Sutskever, I. An Empirical Exploration of Recurrent Network Architectures. In Proceedings of the 32nd International Conference on International Conference on Machine Learning (ICML), Lille, France, 6–11 July 2015.

63. Mangal, S.; Joshi, P.; Modak, R. LSTM vs. GRU vs. Bidirectional RNN for script generation. *arXiv* **2019**, arXiv:1908.04332.
64. Yang, S.; Yu, X.; Zhou, Y. LSTM and GRU Neural Network Performance Comparison Study: Taking Yelp Review Dataset as an Example. In Proceedings of the 2020 International Workshop on Electronic Communication and Artificial Intelligence (IWECAI), Shanghai, China, 12–14 June 2020; pp. 98–101. [CrossRef]
65. Mohammed, A.A.; Umaashankar, V. Effectiveness of Hierarchical Softmax in Large Scale Classification Tasks. In Proceedings of the 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Bangalore, India, 19–22 September 2018; pp. 1090–1094. [CrossRef]
66. Sharma, S.; Sharma, S.; Athaiya, A. Activation Function in Neural Networks. *Int. J. Eng. Appl. Sci. Technol.* **2020**, *4*, 310–316. [CrossRef]
67. Kingma, D.P.; Ba, J.L. Adam: A method for stochastic optimization. In Proceedings of the 3rd International Conference on Learning Representations (ICLR), San Diego, CA, USA, 7–9 May 2015.
68. Jais, I.K.M.; Ismail, A.R.; Nisa, S.Q. Adam Optimization Algorithm for Wide and Deep Neural Network. *Knowl. Eng. Data Sci.* **2019**, *2*, 41. [CrossRef]
69. Sakinah, N.; Tahir, M.; Badriyah, T.; Syarif, I. LSTM with Adam Optimization-Powered High Accuracy Preeclampsia Classification. In Proceedings of the IES 2019—International Electronics Symposium: The Role of Techno-Intelligence in Creating an Open Energy System Towards Energy Democracy, Surabaya, Indonesia, 27–28 September 2019; pp. 314–319. [CrossRef]
70. Défossez, A.; Bottou, L.; Bach, F.; Usunier, N. A Simple Convergence Proof of Adam and Adagrad. *arXiv* **2020**, arXiv:2003.02395.
71. Feurer, M.; Hutter, F. *Automated Machine Learning*; Springer: Cham, Switzerland, 2019; pp. 3–33. [CrossRef]
72. Wu, J.; Chen, X.Y.; Zhang, H.; Xiong, L.D.; Lei, H.; Deng, S.H. Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization. *J. Electron. Sci. Technol.* **2019**, *17*, 26–40. [CrossRef]
73. Li, W.; Wing, W.W.; Wang, T.; Pelillo, M.; Kwong, S. HELP: An LSTM-based approach to hyperparameter exploration in neural network learning. *Neurocomputing* **2021**, *442*, 161–172. [CrossRef]
74. Bergstra, J.; Bengio, Y. Random Search for Hyper-Parameter Optimization. *J. Mach. Learn. Res.* **2012**, *13*, 281–305.
75. Quemy, A. Two-stage optimization for machine learning workflow. *Inf. Syst.* **2020**, *92*, 101483. [CrossRef]
76. Grandini, M.; Bagli, E.; Visani, G. Metrics for Multi-Class Classification: An Overview. *arXiv* **2020**, arXiv:2008.05756.
77. Igual, C.; Pardo, L.A.; Hahne, J.M.; Igual, J. Myoelectric Control for Upper Limb Prostheses. *Electronics* **2019**, *8*, 1244. [CrossRef]
78. Singh, R.M.; Chatterji, S. Trends and Challenges in EMG Based Control Scheme of Exoskeleton Robots—A Review. *Int. J. Sci. Eng. Res.* **2012**, *3*, 933–940.
79. Ison, M.; Artemiadis, P. The role of muscle synergies in myoelectric control: Trends and challenges for simultaneous multifunction control. *J. Neural Eng.* **2014**, *11*, 051001. [CrossRef]