Computer Science and Information Systems 19(3):1199–1212 https://doi.org/10.2298/CSIS220228025M

A novel Approach for sEMG Gesture Recognition using Resource-constrained Hardware Platforms

Matías J. Micheletto[1], Carlos I. Chesñevar[2], and Rodrigo M. Santos[2]

[1] *Golfo San Jorge* Research and Transfer Center
Ruta 1 KM 4 (9000) Chubut, Argentina
matias.micheletto@uns.edu.ar
[2] Institute for Computer Science and Engineering (CONICET-UNS)
Campus Palihue, Bahía Blanca (8000) Buenos Aires, Argentina
cic@cs.uns.edu.ar
ierms@criba.edu.ar

Abstract. Classifying human gestures using surface electromyografic sensors (sEMG) is a challenging task. Wearable sensors have proven to be extremely useful in this context, but their performance is limited by several factors (signal noise, computing resources, battery consumption, etc.). In particular, computing resources impose a limitation in many application scenarios, in which lightweight classification approaches are desirable. Recent research has shown that machine learning techniques are useful for human gesture classification once their salient features have been determined. This paper presents a novel approach for human gesture classification in which two different strategies are combined: a) a technique based on autoencoders is used to perform feature extraction; b) two alternative machine learning algorithms (namely J48 and K*) are then used for the classification stage. Empirical results are provided, showing that for limited computing power platforms our approach outperforms other alternative methodologies.

Keywords: sEMG, gesture recognition, autoencoder, decision trees, nearest neighboors.

1. Introduction

Wearable sensors and mobile devices have introduced a wide range of new applications to support daily life. These applications involve from simple pulse and oxygen sensors that measure the physical activity during training (e.g. [31, 28]) up to more complex e-health applications (e.g. [15, 32]). Gesture recognition from electromyografic (EMG) sensors is a relevant area in which wearable sensors play a major role, particularly surface EMG sensors (sEMG for short). Gesture recognition is intended to be used in different application settings to provide human-like communication through gestures in cases where speech recognition is not possible or appropriate (e.g. because of having a noisy environment or dealing with a user who is not able to type or interact with a touch screen).

In the last years there have been many different approaches for gesture recognition using EMG and sEMG signals [3]. However, most of these approaches focus on obtaining a high overall accuracy of the proposed model as the main goal to be attained. Even though accuracy is a very relevant issue in classification, the assessment of computational costs associated with the resulting models plays also a significant role in many situations (e.g.

a neural network model can have a high accuracy for classifying gestures but only at the expense of a high computational cost for building the model).

In this respect, our research is in line with the concept of *frugal innovation*, which refers to the development of technology for solving modern problems by adapting the requirements to the constrained accessibility of the local market [1, 11, 26, 12]. This aspect is crucial for many developing and emerging countries, where the deployment of new technologies (such as sEMG gesture recognition) is constrained by availability and prices of hardware resources. In this context, lightweight classification techniques are particularly relevant. To the best of our knowledge, the trade-off between accuracy vs. computational resources involved remains as an important issue to be discussed, particularly when considering the real-time performance of gesture recognition algorithms implemented in low cost platforms.

In this article we present a lightweight classification approach for gesture recognition from sEMG signals based on a combined strategy: first, *autoencoders* –a particular subclass of artificial neural networks (ANN)– are used to perform feature extraction. On the basis of the features that have been identified, two well-known classification algorithms (namely J48 and K*) are then used for gesture recognition. From our experiments we can conclude that the proposed approach enables to correctly identify nine different gestures with a high accuracy (87% for a generic user, and up to 96% when training the algorithms with data from a single user).

The rest of this article is structured as follows. Section 2 discusses related work for gesture recognition with an emphasis on sEMG usage, summarizing some relevant advances in the state of the art on the topic. Section 3 presents our approach for gesture recognition, integrating commercial low-cost sEMG sensors and a lightweight model for gesture classification. This model is based on autoencoders for selecting salient features along with the algorithms J48 and K* (corresponding to decision tree learning and instance-based learning using an entropic distance measure, respectively). Sections 5 and 6 show how the lightweight model can be effectively built and how to assess the resulting computer hardware cost. Section 7 discusses the obtained results, contrasting them with other more expensive alternatives found in the literature. Finally, Section 8 concludes.

2. Related work

The process of gesture recognition from acquired signals using sEMG sensors involves usually three stages, namely *filtering*, *feature extraction* and *classification*. First, in the filtering step, signals must be conditioned to eliminate possible interference or noise and to extract the time window that contains the dataset necessary to perform gesture classification. In some cases, the signal envelope is extracted directly from the raw data acquired through the sensors. Next, as a second stage, a feature extraction method is usually employed, providing the basis for performing the gesture recognition process by means of an appropriate classification algorithm, which will take place in the third and final stage.

Next we summarize some of the most relevant papers related to gesture classification, grouping them according to the proposed method for each processing stage, contrasting as well the computing platforms used for validating their performance.

2.1. First stage – data acquisition and filtering

An in-depth analysis of this first stage is usually ignored in the literature. Some research work makes use of public datasets and focuses almost exclusively on the resulting algorithm performance [17, 5]. In other cases, commercial sEMG sensors are used with embedded filtering and denoising capabilities [2, 30, 7, 14, 13]. Less frequently, some articles include a specific section to detail the underlying features for acquisition and filtering [21].

2.2. Second stage – feature extraction

In the literature there are many alternatives for feature extraction for gesture recognition. The most simple consists on using statistical descriptors of the raw signal (e.g. mean average value (MAV), minimum and maximum values, standard deviation (STD), slope sign changes (SSL), zero crossings (ZC), root mean square value (RMS), waveform length (WL), autoregression coefficients (AR), among many others [2, 21, 30].

Another common technique consists on using the discrete wavelet transform (DWT) [8, 6, 24], the Fourier transform (FFT) [19] or the short time Fourier transform (STFT) [17]. These approaches imply higher computational cost, and for the case of embedded devices they are usually implemented using dedicated processing units for computing the transforms in real time.

Fractal dimension (FD) is a less explored method that takes advantage on the fact that EMG signals may show self similarity traces (as proposed in [5]). Another widely used technique applied to dimensionality reduction (detecting features which are deemed as non-relevant for classifying EMG data) is principal component analysis (PCA) and independent component analysis (ICA) [22]. The use of autoencoders (as proposed in this article) as a method for feature extraction has also proven to be useful, being its complexity as shallow neural networks relatively low [7, 18, 13, 22].

2.3. Third stage – gesture classification

For the third stage, the spectrum of possible approaches is wide, as each classifier may have many particular variations. Artificial neural networks (ANN) in their many forms are one of the most frequent methods, as they have gained relevance in the last years, growing along with the increase of available computing power. Thus, we find ANNs ranging from simple perceptron architectures up to deep and convolutional neural networks being used in the classification stage for EMG gesture recognition [13, 5, 18, 29]. In general, the main drawback of ANN approaches derives from the high computational cost required for both the training and inference stages.

The linear discriminant analysis (LDA) method basically consists on finding a linear combination of the features of the model in order to build a space where the projection of the data points corresponding to each class are as distant as possible from each other, allowing to define gesture classification rules. This technique was applied in [18]. The major drawback of this method is the requirement that data should be normally distributed, being thus prone to negative effects from outliers and requiring a carefully data cleaning in the model building phase.

Support vector machines (SVM) share some aspects with LDA and are also applied in gesture classification scenarios [2, 7]. SVM consist on computing the set of hyperplanes that best separates the different classes. However, this method allows to define a more complex decision boundary than LDA, allowing to focus on the correct classification of outliers. The computational cost and accuracy trade-off of this technique can also be manageable.

As we will detail in the next section, our proposal is based on two lightweight classification algorithms (namely decision trees using J48 and K-nearest neighbors using K*). These algorithms were used for gesture classification in previous research work (e.g. [8, 24, 21, 30, 18, 17] with two main differences: a) they were part of more complex ensemble models such as gradient boosting (GBDT) and bagging ; b) the focus of their usage was on accuracy rather than on finding a suitable approach for the trade-off between accuracy and high computational cost, as proposed in this article.

3. Data acquisition and preprocessing stage

To perform the data acquisition, commercial EMG sensors were used (in our case MyoWare Muscle Sensors, manufactured by Advancer Technologies[3]). These sensors are characterized by being small sized, low cost and having a good market availability.

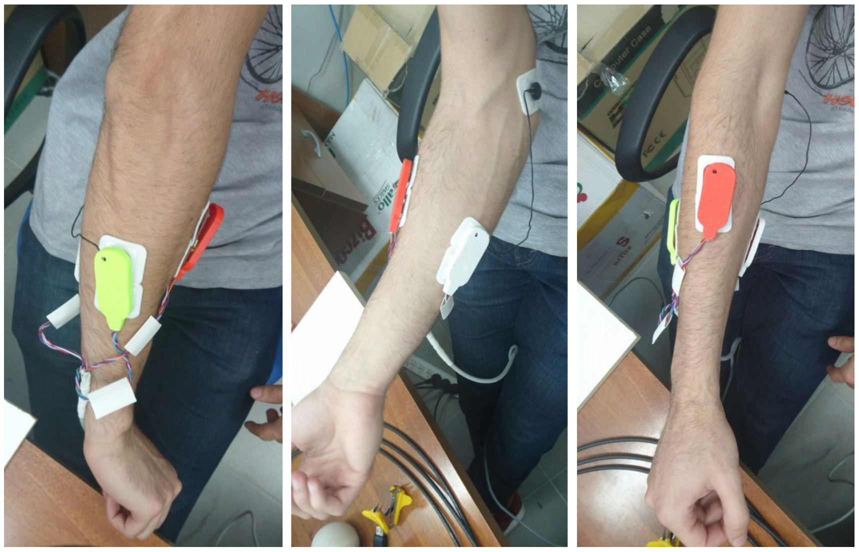

Fig. 1. Surface EMG sensors placement

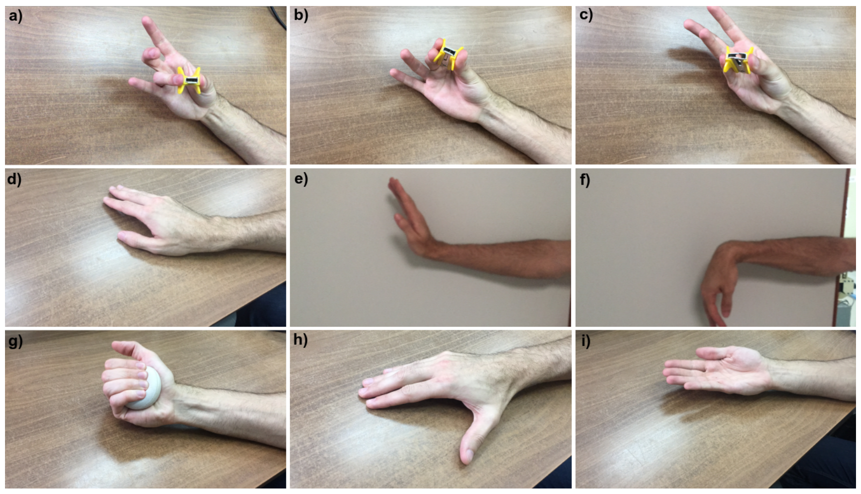

Fig. 2. Selected gestures: a) anu, b) may, c) men, d) adu, e) ext, f) fle, g) pel, h) pul, i) sup

3.1. Sensor placement

The sensor electrodes measure an electric potential difference generated by muscle activity. The resulting signal is amplified, processed through a 400Hz low-pass filter and rectified in order to obtain the signal envelope as a final output. The sensor has a third reference electrode that must be located on an area with little muscle activity to be taken as a comparison point for voltage reference. In general, skin areas close to joints or bones have little electrical activity, so they are appropriate for the placement of the reference electrode.

Figure 1 shows the adhesive electrode pads placement in the forearm of a sample subject. Following the guidelines provided by [23], after contrasting several tests the three sensor channels were located in the following positions:

Channel 1: Placed on the extensor carpi ulnaris and extensor digitorum muscles (involved in the extension of the fingers, wrist and adduction of the hand).

Channel 2: Placed on the flexor carpi ulnaris muscle, which intervenes in the flexion of the wrist.

Channel 3: Located on the brachioradial muscle, which is the main actuator of the supination movement.

Once the three sensors were placed on the subject, their output data was measured and recorded using a Hioki 8861-50 osciloscope (with 16 channels of 16-bit each). Sampling frequency was set at 200 Hz.

3.2. Gestures selection

Different gestures were selected, made with the right or left hand, so that the signals observed in the oscilloscope resulted in a good amplitude and differences observable to the

[3] http://www.advancertechnologies.com

naked eye. The nine gestures shown in Figure 2 were selected and identified with different acronyms: Ring finger (anu), middle finger (may), pinky finger (men), wrist adduction (adu), wrist extension (ext), wrist flection (fle), ball squeeze (pel), thumb extension (pul) and wrist supination (sup).

The executor was asked to perform the movements of the selected gestures repeatedly every certain interval of time, spaced enough to identify the time window with the main variations of the signals associated with the gesture execution. The acquired data was exported in plain text format to be later analyzed.

4. Feature extraction stage

An autoencoder was used as feature extraction stage. Both the input and output layers were configured using the sigmoid activation function. The three sEMG signals were sampled using a window of 200 16-bit integer values each, so that the size of the input layer of the autoencoder comprised 600 values.

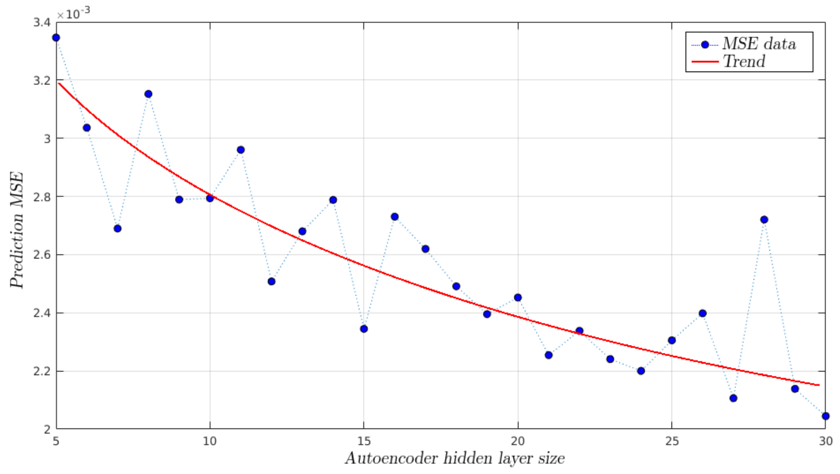

Fig. 3. Mean square error of an autoencoder: prediction MSE vs. hidden layer size

The hidden layer size of an autoencoder is the main attribute that limits the performance of the model as a lossy compression method and is also related to the classifier performance (when used together with a classification algorithm). To avoid the tight coupling between the feature extraction and classification stages, the hidden layer size of the autoencoder was determined measuring its performance when reconstructing the compressed signal and not when classifying the gestures. In order to do this, several tests were performed between 5 to 30 parameters, calculating the mean square error (MSE) in the reconstruction of the validation data (which consisted on random selected holdout subsets with a proportion of 25% of the total dataset).

The learning process was limited to 500 epochs. Fig. 3 shows the MSE values when predicting validation data as the number of neurons in the hidden layer increases. Fig.

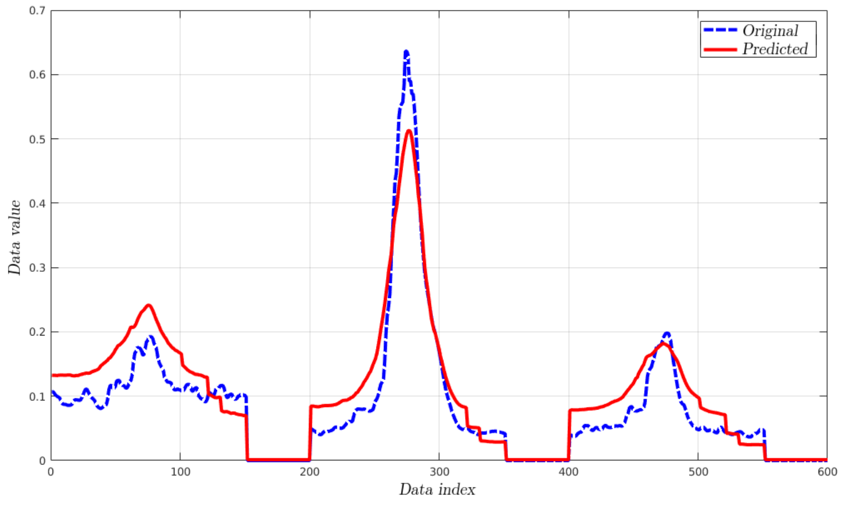

Fig. 4. Input (original) and output (predicted) signals of the trained autoencoder

4 shows the comparison between a signal from the validation data set and its respective reconstruction obtained with the trained autoencoder. It can be observed that the generated signal is not as noisy as the original one, preserving the overall associated shape.

For the classification stage, a hidden layer size of 15 neurons was chosen (as a trade-off between error rate and number of features). Since the holdout subset was randomly selected, three repetitions were performed when training the autoencoders, generating three different classification datasets (one for each executor and a fourth one that combines the gestures from all executors). This resulted in a total of 12 different datasets with 15 features and the labels for the corresponding gestures.

5. Classification stage

Two alternative classification models were built using the open-source software Weka [27, 10]. Two classification schemes were used: *decision trees*, implemented via the C.45 algorithm (available in Weka as J48 [20]) and a particular variation of *k nearest neighbors* (using entropy as a distance function via the so-called K* algorithm [4]). Two validation methods were used: a) holdout with 3/4 train/validation proportion, and b) 10 fold cross-validation. In general, very similar results were obtained between both validation techniques and between both classification methods.

The classification process was repeated for each of the three iterations in order to verify that the results remained constant in different cases. The resulting sets were named with the letter that identifies the executor and the index that indicates the repetition number (e.g. *M3* stands for the data set obtained from the 3rd iteration when measuring results from executor *M*).

It must be noted that the generated decision trees do not use all the features at the same time. This suggests that the number of features could still be reduced, which would

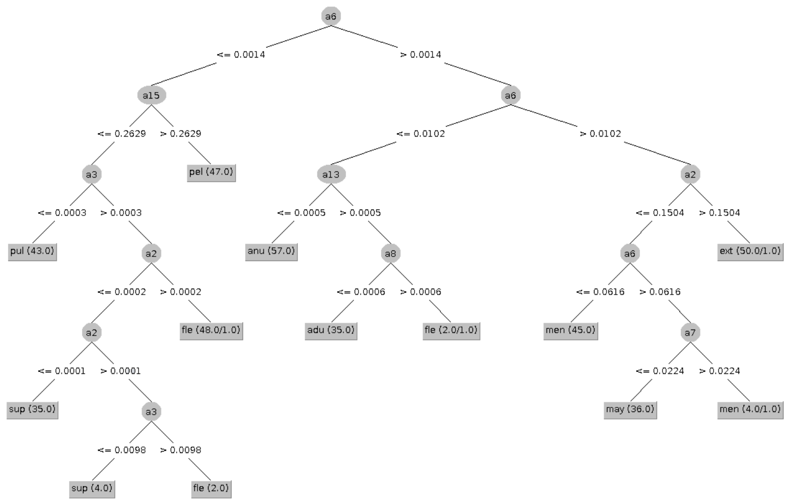

Fig. 5. Decision Tree model for the gesture recognition of the *M2* dataset

imply removing neurons from the trained autoencoder model. To give an example of the decision tree model complexity, Fig. 5 shows the generated model by means of the C4.5 method for the gesture classification trained with the *M2* data set.

6. Hardware requirements analysis

We analyzed the computational complexity and resource requirements of the proposed techniques following similar criteria as those provided by [16, 9, 25]. As an upper bound, we considered a 15-neuron fully connected encoder for 600 16-bit input values and a decision tree of 185 nodes (the latter corresponds to the most complex decision tree from the ones calculated in our experiments).

The data encoder stage requires memory space to store the weight values of 601 parameters (600 weights plus one bias) per neuron and to perform 1,201 floating point operations (FLOPs). The activation function will be evaluated one time per hidden layer neuron, which means the evaluation of a quantized sigmoid function, resulting in a total of 4 FLOPs. Thus, for an encoder with 15 neurons in its hidden layer, the total number of FLOPs required to extract the features of a 600 data values window sums up to 18,075 FLOPs.

The 185-node decision tree requires 185 FLOPs which may result negligible compared to the feature extraction stage. Consequently, using a rough estimation, the total number of FLOPs required is approx. 20,000.

To simplify the real-time analysis, we will assume that a window of 600 data values is processed at the signal sampling frequency (i.e. 200Hz). This is because the classifier was not trained with time-shift invariance considerations. Then, the processor speed should allow to perform all the computations in less than 5 ms. For the case of 16 bit architectures where a floating point operation could be performed on each clock tick, a processor of at least 4 MHz is required to perform 20,000 operations in 5 ms. For 8-bit processors, the clock frequency needs to be doubled, being thus at least 8MHz. This approximation does not take into account the time required to perform the program control, but still by doubling the minimum frequency we would obtain 16Mhz (which is a nominal frequency for low resource microcontrollers).

The memory usage on the other hand, takes into account the storage for all the model parameters and the program itself, which includes all the control instructions. The set of 16-bit 9,015 weights of the encoder itself takes up 17.61Kb of flash memory usage. Most embedded devices allow to expand their capacity by adding external memory, however this implies a non-negligible reading time and should be taken into account on the overall computing time.

The complexity of the classification stage of the instance based learner is harder to analyze as it depends on the training stage results. In order to make an estimate of the required computing resources, we will assume that an instance based classifier uses all the 15 features and 1,000 instances, from the total of 1,142 rows. Assuming two bytes of memory per feature plus a byte per class label, this represents a total memory usage of 31,000 bytes or 30.27Kb (only to store the database with examples).

Regarding the FLOPs number, we will assume that the classification stage of this algorithm compares the test instance to all the given instances to determine the probability that the test instance belong to the class of each compared example. We will also assume that the computation of this probability requires two FLOPs per instance in the database. Note that the entropy function is defined by equation 1,

$$H(x) = \sum_{i=0}^{N-1} -p_i(x) \cdot \log\left(p_i(x)\right). \tag{1}$$

Where $N = 9$ is the number of classes. The logarithm implementation for the range (0.0, 1.0) using a four-degree Taylor series would involve up to eight FLOPs, and adding another FLOP for the multiplication in equation 1 results in 9 FLOPs. Then, adding the operations used for the computation of the probability $p_i(x)$, the entropy function $H(x)$ would require 2,009 FLOPs, and should be performed nine times (i.e., one time per class), which results in a total of 18,081 FLOPs. This number falls inside the estimated range as for the case of the decision tree model.

In summary, the requirements to storage the complete classification model and compute real-time results would be 16MHz as operating frequency and around 32Kb of required memory storage (matching the specs of a low cost microcontroller, such as the AVR ATMega 328p[4]).

7. Results and discussion

Table 1 shows the accuracy of each classification method using different validation techniques. All values are expressed in percentages of classification efficiency and each cell

[4] https://en.wikipedia.org/wiki/ATmega328

corresponds to the average results between the three repetitions. The *E*, *L* and *M* sets indicate the different gesture executors and the *E+L+M* set contains the data from all three executors. Average values across columns and rows are shown in bold.

Table 1. Comparative average accuracy values by validation method

Set	**J48**			**K***		
	Holdout	**Crossvalidation**	**Average**	**Holdout**	**Crossvalidation**	**Average**
E	96	94.2	**95.1**	93.9	95.6	**94.7**
M	88.4	88.4	**88.4**	91.8	94	**92.6**
L	97.2	95.6	**96.4**	98.3	98.4	**98.4**
Average	**93.87**	**92.73**	**93.3**	**94.67**	**96**	**95.23**
E+M+L	87.1	86.9	**87**	92	92.8	**92.4**

It can be observed that the K* method obtained slightly better accuracy percentages than J48. However, the decision tree classification is computed about ten times faster than K*. The Table 2 shows the sizes and number of characterization parameters used by each decision tree (numbered columns correspond to each of the three repetitions). This result shows that when the decision tree models are trained with data corresponding to a single executor many parameters are left aside by the classifier. This allows to reduce the computational cost of the model by removing neurons from the hidden layer of the already trained encoder while maintaining the same classification accuracy level. However, when using data from multiple executors, the decision tree model complexity grows notably, and almost all 15 parameters are used.

Another experiment was performed in order to test to what extent the classification accuracy is affected by reducing the number of characterization parameters, i.e. the autoencoder's hidden layer size. In table 3, the accuracy values are shown when repeating the training process for the classifiers and using only eight parameters to characterize the signals of the three data sets combined. Here we can observe that when reducing the parameter number from 15 to 8, the accuracy of the classification stage is reduced around a 6% for the decision tree and 7.9% for the K* model. We can conclude that a substantial improvement is achieved in terms of computational cost, considering that the autoencoder size was reduced to almost the half.

Table 2. Number of used features and the resulting tree size for the classification decision tree using J48

	Number of used features						Size of tree (node count)					
Set	**Holdout**			**Crossvalidation**			**Holdout**			**Crossvalidation**		
	1	**2**	**3**	**1**	**2**	**3**	**1**	**2**	**3**	**1**	**2**	**3**
E	12	11	8	8	10	9	31	37	25	25	31	27
M	12	12	13	13	10	10	39	49	35	39	33	35
L	6	8	7	8	10	11	19	21	25	25	23	27
E+M+L	15	15	15	15	14	15	149	165	137	185	133	165

Table 3. Classification results using eight characterization parameters

Set	**J48**		**K***	
	Holdout	**Crossvalidation**	**Holdout**	**Crossvalidation**
E+M+L	87.9	86.6	86.9	87.7

Table 4. Confusion matrix for the decision tree trained with ELM3 set

Class	anu	may	men	adu	ext	fle	pel	pul	sup
anu	40	0	0	0	0	4	0	1	0
may	0	46	1	0	0	1	1	1	0
men	0	1	43	1	3	1	0	0	0
adu	0	0	0	27	4	1	0	5	0
ext	0	1	1	4	34	0	0	0	0
fle	2	0	0	0	0	32	1	0	0
pel	0	0	1	0	0	1	43	1	1
pul	0	1	2	3	0	2	0	34	1
sup	0	0	0	0	0	0	2	1	32

Table 4 shows the resulting confusion matrix for the decision tree generated with Weka's J48 algorithm applied to the complete data set of 1142 instances and validated via the holdout method. The rows correspond to the classes of the set and the columns correspond to the classification results for each instance. From 381 instances evaluated, 335 were correctly classified (87.93%).

8. Conclusion

We have presented a lightweigth classification approach to sEMG gesture recognition. The approach includes a feature extraction stage that uses autoencoders of 15-neuron hidden layer and a classification stage involving two alternatives: a decision tree (via the J48 algorithm) and an instance based learner (via the K* algorithm).

Based on previous work and by a rough estimate of minimal resource requirements, we have shown that the proposed prediction models can be implemented on low resource architectures. An example of a hardware platform that matches with the minimum requirements is the AVR ATMega328p microcontroller (a widely off-the-shelf used platform with a low price and good market availability, used for the Arduino UNO prototyping platform). The resulting decision tree based classifier was able to correctly identify nine gestures with an accuracy of 87% (for a generic user) and up to 93.3% when calibrating the algorithms using data from a single user. The instance based learning classifier reaches 92.4% accuracy when trained with multiple user data and 95.23% when calibrated by a single user data.

The results presented on this work opens the possibility of implementing sEMG based gesture classification systems on low-resource platforms with high market availability, paving the way for developing several low cost and accessible products, from commercial small wearable devices to educational prototyping tools.

References

1. Agarwal, N., Brem, A.: Frugal innovation-past, present, and future. IEEE Engineering Management Review 45(3), 37–41 (2017)
2. Akhmadeev, K., Rampone, E., Yu, T., Aoustin, Y., Carpentier, E.L.: A testing system for a real-time gesture classification using surface emg. IFAC-PapersOnLine 50(1), 11498 – 11503 (2017), 20th IFAC World Congress
3. Buongiorno, D., Cascarano, G.D., De Feudis, I., Brunetti, A., Carnimeo, L., Dimauro, G., Bevilacqua, V.: Deep learning for processing electromyographic signals: A taxonomy-based survey. Neurocomputing 452, 549–565 (2021)
4. Cleary, J.G., Trigg, L.E.: K*: An instance-based learner using an entropic distance measure. In: Prieditis, A., Russell, S. (eds.) Machine Learning Proceedings 1995, pp. 108–114. Morgan Kaufmann, San Francisco (CA) (1995)
5. Coelho, A.L., Lima, C.A.: Assessing fractal dimension methods as feature extractors for emg signal classification. Engineering Applications of Artificial Intelligence 36, 81 – 98 (2014)
6. David Orjuela-Cañón, A., Ruíz-Olaya, A.F., Forero, L.: Deep neural network for emg signal classification of wrist position: Preliminary results. In: 2017 IEEE Latin American Conference on Computational Intelligence (LA-CCI). pp. 1–5 (Nov 2017)
7. Farouk Ibrahim Ibrahim, M., Ali Al-Jumaily, A.: Auto-encoder based deep learning for surface electromyography signal processing. Advances in Science, Technology and Engineering Systems Journal 3, 94–102 (01 2018)
8. Gokgoz, E., Subasi, A.: Comparison of decision tree algorithms for emg signal classification using dwt. Biomedical Signal Processing and Control 18, 138 – 144 (2015)
9. Gordon, A., Eban, E., Nachum, O., Chen, B., Wu, H., Yang, T., Choi, E.: Morphnet: Fast and simple resource-constrained structure learning of deep networks (2018)
10. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: An update (2000)
11. Hossain, M.: Frugal innovation: Conception, development, diffusion, and outcome. Journal of Cleaner Production 262, 121456 (2020)
12. Hossain, M.: Frugal innovation and sustainable business models. Technology in Society 64, 101508 (2021)
13. Huang, Y., Chen, K., Zhang, X., Wang, K., Ota, J.: Joint torque estimation for the human arm from semg using backpropagation neural networks and autoencoders. Biomedical Signal Processing and Control 62, 102051 (2020)
14. Jiang, Y., Chen, C., Zhang, X., Chen, C., Zhou, Y., Ni, G., Muh, S., Lemos, S.: Shoulder muscle activation pattern recognition based on semg and machine learning algorithms. Computer Methods and Programs in Biomedicine 197, 105721 (2020)
15. Lowe, S., ÓLaighin, G.: Monitoring human health behaviour in one's living environment: A technological review. Medical Engineering & Physics 36(2), 147–168 (2014)
16. Mitra, S., Chattopadhyay, P.: Challenges in implementation of ann in embedded system. In: 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT). pp. 1794–1798 (2016)
17. Rabin, N., Kahlon, M., Malayev, S., Ratnovsky, A.: Classification of human hand movements based on emg signals using nonlinear dimensionality reduction and data fusion techniques. Expert Systems with Applications 149, 113281 (2020)
18. Zia ur Rehman, M., Gilani, S.O., Waris, A., Niazi, I.K., Slabaugh, G., Farina, D., Kamavuako, E.N.: Stacked sparse autoencoders for emg-based classification of hand motions: A comparative multi day analyses between surface and intramuscular emg. Applied Sciences 8(7) (2018)

19. Sadikoglu, F., Kavalcioglu, C., Dagman, B.: Electromyogram (emg) signal detection, classification of emg signals and diagnosis of neuropathy muscle disease. Procedia Computer Science 120, 422 – 429 (2017), 9th International Conference on Theory and Application of Soft Computing, Computing with Words and Perception, ICSCCW 2017, 22-23 August 2017, Budapest, Hungary
20. Salzberg, S.L.: C4.5: Programs for machine learning by j. ross quinlan. morgan kaufmann publishers, inc., 1993. Machine Learning 16, 235–240 (1994)
21. Song, W., Han, Q., Lin, Z., Yan, N., Luo, D., Liao, Y., Zhang, M., Wang, Z., Xie, X., Wang, A., Chen, Y., Bai, S.: Design of a flexible wearable smart semg recorder integrated gradient boosting decision tree based hand gesture recognition. IEEE Transactions on Biomedical Circuits and Systems 13(6), 1563–1574 (2019)
22. Spuler, M., Irastorza Landa, N., Sarasola Sanz, A., Ramos-Murguialday, A.: Extracting muscle synergy patterns from emg data using autoencoders. In: Villa, A.E., Masulli, P., Pons Rivero, A.J. (eds.) Artificial Neural Networks and Machine Learning - ICANN 2016. pp. 47–54. Springer International Publishing, Cham (2016)
23. Stegeman, D., Hermens, H.: Standards for suface electromyography: The european project surface emg for non-invasive assessment of muscles (seniam). SENIAM Project 1, 352–360 (01 2007)
24. Subasi, A., Yaman, E., Somaily, Y., A. Alynabawi, H., Alobaidi, F., Altheibani, S.: Automated emg signal classification for diagnosis of neuromuscular disorders using dwt and bagging. Procedia Computer Science 140, 230–237 (01 2018)
25. Tang, R., Adhikari, A., Lin, J.: Flops as a direct optimization objective for learning sparse neural networks. In: NIPS 2018 Workshop on Compact Deep Neural Networks with Industrial Applications (CDNNRIA). pp. 1–4 (11 2018)
26. Winkler, T., Ulz, A., Knobl, W., Lercher, H.: Frugal innovation in developed markets - adaption of a criteria-based evaluation model. Journal of Innovation and Knowledge 5(4), 251–259 (2020)
27. Witten, I.H., Frank, E., Hall, M.A., Pal, C.J.: Data Mining, Fourth Edition: Practical Machine Learning Tools and Techniques. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 4th edn. (2016)
28. Xiao, N., Yu, W., Han, X.: Wearable heart rate monitoring intelligent sports bracelet based on internet of things. Measurement 164, 108102 (2020)
29. Yu, Y., Chen, C., Sheng, X., Zhu, X.: Multi-dof continuous estimation for wrist torques using stacked autoencoder. Biomedical Signal Processing and Control 57, 101733 (2020)
30. Cedeño Z., C., Cordova-Garcia, J., Asanza A., V., Ponguillo, R., Muñoz M., L.: k-nn-based emg recognition for gestures communication with limited hardware resources. In: 2019 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI). pp. 812–817 (2019)
31. Zhang, L., Yang, L., Wang, Z., Yan, D.: Sports wearable device design and health data monitoring based on wireless internet of things. Microprocessors and Microsystems p. 103423 (2020)
32. Zhou, X.: Wearable health monitoring system based on human motion state recognition. Computer Communications 150, 62–71 (2020)

Matías Micheletto received the Electrical Engineering degree and the Doctorate (Ph.D.) in Engineering from the Universidad Nacional del Sur (UNS) Bahía Blanca, Argentina in 2016 and 2020 respectively. He is currently a posdoctorate CONICET scholar at ”Golfo San Jorge” Research and Transfer Center, in Comodoro Rivadavia, Argentina. His research interest is in the field of numerical modeling and optimization, evolutionary computing and embedded systems engineering.

Carlos Chesñevar is principal researcher and full professor at the Universidad Nacional del Sur in Bahía Blanca, Argentina. From 2015 he is the director of the Institute of Computer Science and Engineering (ICIC CONICET UNS). His research is oriented towards different applications of AI technologies. He has participated as PC member in most major AI conferences (IJCAI, AAMAS, ECSQARU, etc.) and has published more than 40 journal articles, 10 book chapters and more than 100 international conference papers. He has led international projects in Artificial Intelligence funded by DAAD (Deutscher Akademischer Austauschdienst) and Microsoft Research Latinamerica.

Rodrigo Santos received the Electrical Engineering degree and the Doctorate (Ph.D.) in Engineering from the Universidad Nacional del Sur (UNS) in Bahía Blanca, Argentina in 1997 and 2001, respectively. He is currently CONICET Adjunt Researcher and Full Proffesor at UNS. His research interests are scheduling embedded real-time systems, data ad-hoc networks and collaborative systems. He has been a visiting Professor at Universidad Nacional de San Agustin, Arequipa, Peru, Scuola Superiore Sant'Anna, Universidad Nacional de la Patagonia San Juan Bosco and Universidad Argentina de la Empresa.

Received: February 28, 2022; Accepted: June 25, 2022.