 doi:10.20944/preprints202002.0443.v1

Article

Identification of hand movements from electromyographic signals using Machine Learning

A. Mora Rubio [1,‡,*], J.A. Alzate Grisales [1,‡], R. Tabares-Soto [1], S. Orozco-Arias [2,3], C.F. Jiménez Varón [4] and J.I. Padilla Buriticá [1]

1 Universidad Autónoma de Manizales, Department of Electronics and Automation, Antigua Estación del Ferrocarril Manizales, 170001 - Caldas - Colombia
2 Universidad Autónoma de Manizales, Department of Computer Science, Antigua Estación del Ferrocarril Manizales, 170001 - Caldas - Colombia
3 Universidad de Caldas, Department of Systems and Informatics, Manizales, 170001 - Caldas - Colombia
4 Universidad Autónoma de Manizales, Department of Physics and Mathematics, Antigua Estación del Ferrocarril Manizales, 170001- Caldas-Colombia
* Correspondence: alejandro.morar@autonoma.edu.co
† These authors contributed equally to this work.

Abstract: Electromyographic (EMG) signals provide information about a person's muscle activity. For hand movements, in particular, the execution of each gesture involves the activation of different combinations of the forearm muscles, which generate distinct electrical patterns. Conversely, the analysis of these muscle activation patterns, represented by EMG signals, allows recognizing which gesture is being performed. In this study, we aimed to implement an automatic identification system of hand or wrist gestures based on supervised Machine Learning (ML) techniques. We trained different computational models and determined which of these showed the best capacity to identify six hand or wrist gestures and generalize between different subjects. We used an open access database containing recordings of EMG signals from 36 subjects. Among the results obtained, we highlight the performance of the Random Forest model, with an accuracy of 95.39%, and the performance of a convolutional neural network with an accuracy of 94.77%.

Keywords: EMG, Machine Learning, Deep Learning, Computational models, Hand and wrist gestures.

1. Introduction

EMG signals are generated by the electrical activity of skeletal muscle fibers and can be captured through two ways. First, invasive methods involve the insertion of a needle directly into the muscle; second, non-invasive methods rely on surface electrodes that are placed on the skin in areas close to the target muscle. In particular, surface EMG is the most suited for establishing human-machine interfaces of daily use [1]. These kind of signals present subtle variations between subjects, associated with anatomical differences and motor coordination [2]. The variations are random and highly non-stationary [3]; therefore, it is necessary to use different techniques of feature extraction and selection for time series, which are also useful for developing optimal models of artificial intelligence (AI) [4].

These signals are used for several purposes by health professionals, including disease diagnosis and rehabilitation. Recently, with the rise of automatic learning techniques, these signals are used in systems for the classification of movements [5–7], the recognition of motor unit action potential (MUAP) [8], and the diagnosis of neuromuscular diseases [9]. These tasks are achieved by measuring and studying the features that can be extracted from EMG signals. Feature extraction can be done through different techniques, such as autoregressive models [10], signal entropy measurements [11], and statistical measurements of amplitude in the time and frequency domains [4]. In particular,

 doi:10.20944/preprints202002.0443.v1

the Wavelet transform, which provides information in the frequency domain at both high and low frequencies, is highly used [12,13].

The feature extraction stage is followed by the implementation of classification models that can learn from the extracted features. Regarding computational models, AI is considered the main domain, which includes ML and Deep Learning (DL). AI refers to the process of displaying human intelligence features in a machine. ML is used to achieve this task, while DL is a set of models and algorithms used to implement ML [14]. In general, ML and DL technologies are powerful for extracting features and finding relationships between data; therefore, these approaches are suitable for tasks in which they can take advantage of human experience [15–17]. Machine and Deep Learning algorithms applied to electrophysiological signals constitute a rapidly growing field of research and allow researchers to train computer systems as experts that can be used later on to support the decision-making process.

In ML, the most commonly used models are Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Ensemble classifiers. A SVM was used in [18] to classify human gait phases based on bilateral EMG recordings. The KNN model was used in [8] to indicate how well different types of features could be used to decompose an EMG signal into its component motor unit potential trains (MUPTs). As for Ensemble classifiers, these include the ensemble of SVM used in [19] to detect neuromuscular disorders using statistical measurements from each sub-band of the discrete wavelet transform (DWT) as features; Random Forest (RF) used in [9] to classify disorders as neuropathies or myopathies based on EMG signals using features extracted from DWT coefficients and power spectral density; and Rotation Forest used in [20] for the diagnosis of neuromuscular disorders. Finally, in DL, different architectures are reported, including fully connected artificial neural networks (ANN) and convolutional neural networks (CNN). In particular, [21] used an architecture that includes a convolution layer with 32 filters, a ReLu activation layer, a MaxPooling layer, a fully connected layer, and, lastly, a softmax output layer with six units.

This article proposes a novel approach to EMG signal processing, in which not all the information of the time series is considered. Here, we use only the raw data for hand gestures (i.e. the muscular activation values provided by each channel) to identify the patterns associated with six gestures, regardless of the subject performing the action. We compared different ML algorithms to improve the precision in the identification and classification of hand gestures from the EMG signals. More important, we aimed to generalize among different subjects, which is still an issue regarding this classification task. The classification accuracies obtained using this approach are over 90% for several models, which demonstrates its feasability.

This article is organized as follows: section 2 presents the materials and methods, including the database and models used in the experimental process, which is explained in detail in section 3. Section 4 contains the experimental results and, lastly, sections 5 and 6 correspond to the discussion and conclusions, respectively, derived from the research.

2. Materials and Methods

2.1. Database

The information used to develop this research was retrieved from a free database called *EMG data for gestures Data Set* available on the website of the UCI Machine Learning Repository. This database contains surface EMG (sEMG) signal recordings in a time interval where six different static hand gestures are executed, providing a total of 72 recordings from 36 patients. The static hand gestures referred to include resting hand, grasped hand, wrist flexion, wrist extension, ulnar deviation, and radial deviation. These gestures were performed for three seconds, each with a three-second interval between gestures. The information was collected using a *MYO thalmic bracelet*; this device has eight channels equally spaced around the patient's forearm. Accordingly, each recording consists of an eight-channel time series, in which each segment is properly labeled with numbers zero to six,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 February 2020 doi:10.20944/preprints202002.0443.v1

where zero corresponds to the intervals between gestures and numbers one to six refer to the gestures mentioned above. Table 1 relates the gestures to the classes or labels.

Label	Gesture
0	Intervals between gestures
1	Resting hand
2	Grasped hand
3	Wrist flexion
4	Wrist extension
5	Ulnar deviation
6	Radial deviation

Table 1. Labels of the gestures contained in the database

This database was consolidated by [2] during their research on the factors limiting human-machine interfaces via sEMG.

2.2. Conventional models

The ML models implemented in this experiment were the following: K-Nearest Neighbors (KNN), Logistic Regression (LR), Gaussian Naive Bayes (GNB), Multilayer Perceptron (MLP) using one hidden layer, Random Forest (RF), and Decision Tree (DT).

K-Nearest Neighbors (KNN) is one of the most recognized ML algorithms for solving classification tasks. This model is quite simple and consists of storing the labeled training data. To classify an unlabeled object, a similarity metric between the object and the labeled objects is computed to identify the k closest elements to the unlabeled object. Then, the labels of these closest elements are used to perform the classification. If k=1, then, the label of the unlabeled object is determined by the label of the closest element of the training set; otherwise, if $k>$ 2, the label is determined by vote [22].

Logistic Regression (LR) is a linear classification model that computes the weighted sum of the imput features and uses a Sigmoid activation function to compute class probabilites and make a prediction [22].

Gaussian Naive Bayes (GNB) is based on a probabilistic model derived from Bayes' theorem. This classification model assumes that all features are independent from each other and determines the probability that an instance belongs to a class based on the feature probabilities. In the Naive Bayes classifier, the features are assumed to be normally distributed (Gaussian distribution) [23].

Multilayer Perceptron (MLP) model has multiple stages of processing and can be considered a generalization of linear models. Essentially, this model uses multiple linear units that compute the weighted sum of the input features and are affected by an activation function. In MLP, this process is repeated multiple times to compute an intermediate processing step called the hidden layer, which is then used to generate the final result [22].

Random Forests (RF) are in the category of ensemble learning methods, which combine multiple ML models to create a more robust one. A RF is basically a collection of Decision Trees, where each tree is different from the others. The differences between the trees can be accomplished in two ways; by randomizing the selection of the training data to build the tree or randomizing the selection of the features used in each question [22]. The overall prediction is made using the predictions of the ensemble by majority vote.

Decision Trees (DT) are widely known ML algorithms used for classification or regression tasks. The basis of this model is that it learns a hierarchy of yes/no (if/else) questions that allow making a prediction [22].

2.3. Deep Learning models

The concept of DL comes from artificial neural network research. Unlike the traditional proposals, modern DL has achieved training stability and the ability to generalize, even on big

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 February 2020 doi:10.20944/preprints202002.0443.v1

data. Nowadays, DL is the technique of choice to achieve the highest predictive accuracy since DL algorithms perform quite well in a variety of problems. DL architectures are computational models involving hierarchical feature extraction, which is typically accomplished using multiple levels of nonlinearity [14]. Particularly, artificial neural networks occupy a prominent position among the many supervised learning algorithms currently used in ML [24].

2.3.1. Fully connected artificial neural network (ANN) architecture

A fully connected artifical neural network is characterized by having all units between layers connected; therefore, the activation values computed by a unit are passed to every unit in the following layer. In general, the activation values of a layer are given by equation 1.

$$a = g(W \cdot x + b) \tag{1}$$

W represents the weights of the layer, x corresponds to the input data of the layer, which can also be the activation values of the previous layer , and b is the bias. The sign $\cdot$ is used to indicate the matrix multiplication operation and $g()$ represents the activation function used, usually nonlinear.

The model implemented here corresponds to a network with three hidden layers and one output layer. The output layer has six units corresponding to the number of movements that we aim to identify. The activation function for the output layer was softmax and the loss function was categorical cross-entropy. The diagram of the ANN architecture is shown in figure 1.

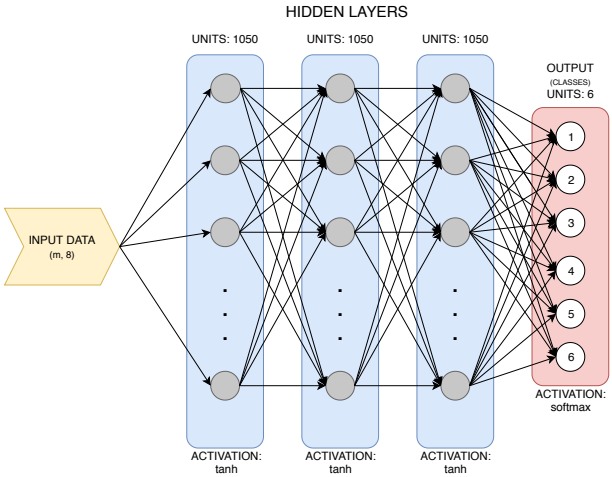


Figure 1. Fully connected artifical neural network architecture

2.3.2. Convolutional neural network (CNN) architecture

The main feature of these networks is that they use the convolution operation to extract features from the data. Each convolutional layer has several kernels, or filters, that are slid over the data while computing the number of coefficients that can be used as features. Given its nature, the output dimension of a convolutional layer depends on the size and amount of filters used. The convolution operation for one layer is presented in equation 2.

$$a = g(f_i * x + b) \tag{2}$$

f_i represents the *i-th* filter of the layer, x corresponds to the input data of the layer, which can also be the activation values of the previous layer, and b is the bias. The sign $*$ is used to indicate the convolution operation and $g()$ represents the activation function used.

The model implemented in this study corresponds to a convolutional neural network with two 1D convolutional layers, one ReLu activation layer, three fully connected layers, and one output layer.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 February 2020 doi:10.20944/preprints202002.0443.v1

The output layer has six units corresponding to the number of movements that we aim to identify. The activation function for the output layer was softmax and the loss function was categorical cross-entropy. The diagram for the CNN architecture is presented in figure 2.

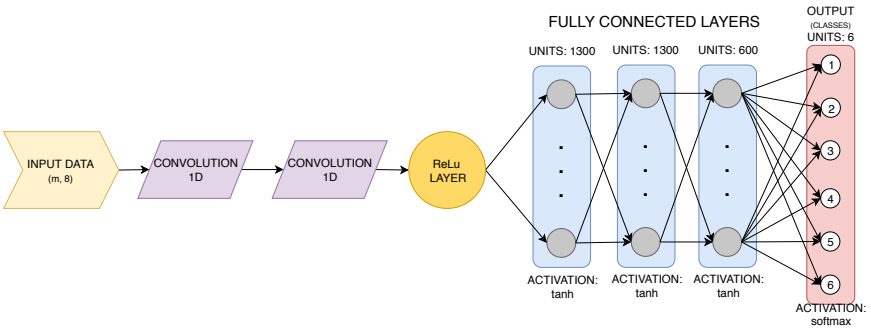


Figure 2. Convolutional neural network architecture

2.4. Tools

The complete experimental procedure was implemented in Python v. 3.7.3 programming language. We used Pandas library v. 0.24.2 for database import and preprocessing. The ML models mentioned in section 2.2, GridSearchCV, Principal Component Analysis (PCA), train-test split, and the scaling function were implemented using the Scikit-learn library v. 0.21.2 [25]. The artificial neural network architectures (section 2.3) and the corresponding training and validation processes were implemented using the Keras library v. 2.2.4 [26].

The resources used in this research are available in a public GitHub repository containing the EMG recordings files from the database, as well as a Jupyter notebook with the implemented source code. The repository is available here.

3. Detailed experimental process

Figure 3 presents the flowchart of the experimental procedure of this research.

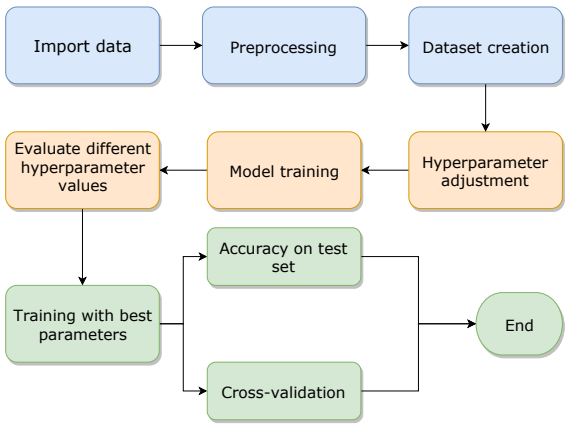


Figure 3. Flowchart of the experimental procedure

3.1. Importing the data

The data from the database is presented as a plain text file (.txt). We used the *read_csv* function from Pandas library to import these files.

3.2. Preprocessing

We removed the data that was labeled with the number zero since it corresponds to the resting intervals between gesture execution and does not contain relevant information for the solution of the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 February 2020 doi:10.20944/preprints202002.0443.v1

problem. Thus, only the data captured during the execution of the six gestures of interest were kept. We did not apply any feature extraction procedure (i.e. raw data) because we intended to use the values of the eight channels provided by the device as discriminating information between gestures.

3.3. Dataset creation

For dataset creation, we guaranteed that the patients were correctly distributed into the training, validation, and test sets. Accordingly, each patient was present in only one set to avoid bias induced by considering data from the same patient for training and testing. With this in mind, the training set was formed by concatenating the data from 30 patients; the validation set comprised three patients, and test set included the remaining three patients. The patients conforming each of the sets were randomly selected. A second group of sets was created by scaling the first one; in this way, all the features equally contributed to the training process. The sets described above were used during the hyperparameter adjustment process and to compute the *test set accuracy* metric. Finally, to conserve the correct patient distribution during the cross-validation process, each of the nine k-folds was manually generated by concatenating the data from four random patients.

3.4. Hyperparameter adjustment

Following dataset creation, we adjusted the parameters inherent to each classification method assessed. Table 2 summarizes the parameters that were adjusted in the ML models, as well as the ranges used for each one. For the parameters not listed in this table, we used the default values defined by the library.

Model	Parameters and range
KNN	n_neighbors in the range of 1 to 6, with step 1.
LR	C in the range 0.1 to 1, with step 0.1.
GNB	N/A
MLP	hidden_layer_sizes in the range of 200 to 1200, with step 200.
RF	n_estimators in the range of 1 to 100, with step 10.
DT	max_depth in the range of 1 to 110, with step 10. min_samples_split in the range of 10 to 110, with step 10. max_features in the range of 1 to 6, with step 1.

Table 2. Parameters and ranges that were adjusted for the ML models

Given the number of variable parameters for ANN and CNN architectures, we decided to adjust these one or two parameters at a time using the GridSearchCV method implemented in the Scikit-learn library. This method selects the best combination of parameters using the result of a cross-validation process. The parameters were adjusted in the following order: batch_size and epochs, optimization algorithm, network weight initialization, activation function, and, finally, neurons in hidden layers. Table 3 presents the adjusted parameters for the ANN and their corresponding ranges, and Table 4 shows the parameters set for the CNN.

Parameter	Values
batch_size	50,000 - 100,000 - 150,000
epochs	150, 170, 200
optimization algorithm	RMSprop, Adagrad, Adadelta, Adam, Adamax, Nadam
network weight initialization	uniform, lecun_uniform, normal, zero, glorot_normal, glorot_uniform, he_normal, he_uniform
activation function	softmax, softplus, softsign, relu, tanh, sigmoid, hard_sigmoid, linear
neurons on hidden layers	850 - 900 - 950 - 1,000 - 1,050 - 1,100 - 1,200

Table 3. ANN ranges and parameters that were adjusted

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 February 2020 doi:10.20944/preprints202002.0443.v1

Parameter	Values
batch_size	10,000 - 15,000 - 20,000 - 30,000 - 50,000
optimization algorithm	RMSprop, Adagrad, Adadelta, Adam, Adamax, Nadam
network weight initialization	uniform, normal, glorot_normal, he_normal
activation function	relu, tanh, linear
neurons on hidden layers	1,050 - 1,100 - 1,200 - 1,250
filters	16, 32, 64, 128
kernel_size	1, 2, 3

Table 4. CNN ranges and parameters that were adjusted

3.5. Model training and hyperparameter optimization

For parameter optimization, we used an approach similar to a grid search. Given a set of parameters and value ranges, the model was trained and tested on the validation set using every possible combination. For traditional ML models, this process was implemented manually using *for* cycles, whereas for artificial neural networks, we used the GridSearchCV function from Scikit-learn library due to the high number of parameters.

3.6. Evaluation of the model using the best parameters

After performing hyperparameter optimization, the best set of parameters for each model was used to generate the definitive evaluation metric. For this, we used cross-validation to measure model performance. This method uses the complete feature matrix and divides the dataset into k-folds parts. In this case, we used k-folds = 9 and each part contained data from four randomly selected patients. The data was iterated over these partitions, using eight for training and one for validation until they all passed through both states. This procedure was used to check the stability of the results. In addition to cross-validation, we calculated model accuracy on the test set to estimate the model's ability to generalize. The test set had not been used in any previous stage of the experiment. Finally, as class measurements, we used precision, recall, and F1-score.

4. Results

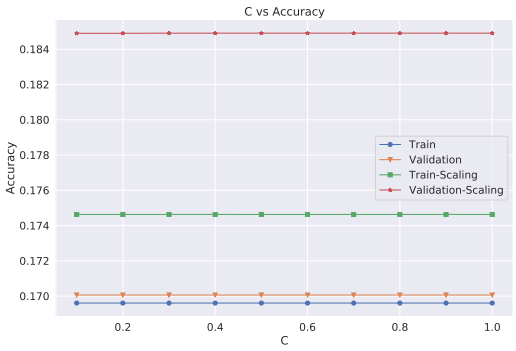
Parameter optimization of the ML models was visualized by the graphs shown in figure 4, which display the model performance behavior during parameter adjustment. According to these graphs, we selected the parameter values and dataset (Normal/Scaling) for each of the ML models, which are listed in table 5.

Model	Best parameter value
KNN	n_neighbors = 1 Dataset: Normal.
LR	C = 0,3 Dataset: Scaling.
GNB	N/A
MLP	hidden_layer_sizes = 800 Dataset: Scaling
RF	n_estimators = 71 Dataset: Scaling.
DT	Default value: max_depth = None min_samples_split = 2 max_features = None Dataset: Scaling

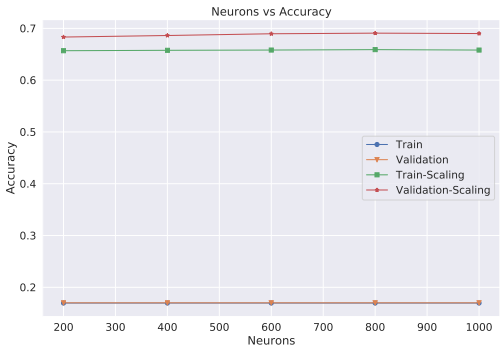
Table 5. Results of the parameter adjustment for the ML models

For the ANN architecture, the results of the parameter adjustment using GridSearchCV are presented in table 6. Using these values, we trained the ANN and generated graphs of performance accuracy and loss in each training epoch to observe the model behavior and determine the performance

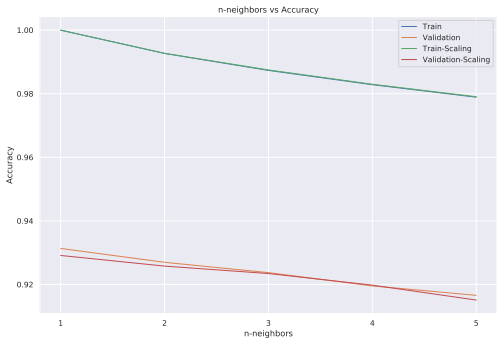
Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 February 2020 doi:10.20944/preprints202002.0443.v1



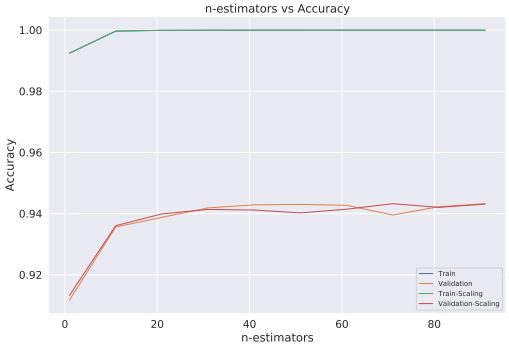
(a) LR



(b) MLP



(c) KNN



(d) RF

Figure 4. Parameter adjustment of the ML models

of the learning process. Figure 5 contains the model accuracy and loss curves during training. For the CNN architecture, the results of the parameter adjustment using GridSearchCV are shown in table 7. The number of epochs was manually adjusted according to the model training curves; as a result, the most appropriate value found was Epochs = 100. Figure 6 shows the accuracy and loss curves during training for CNN using the parameters listed in table 7.

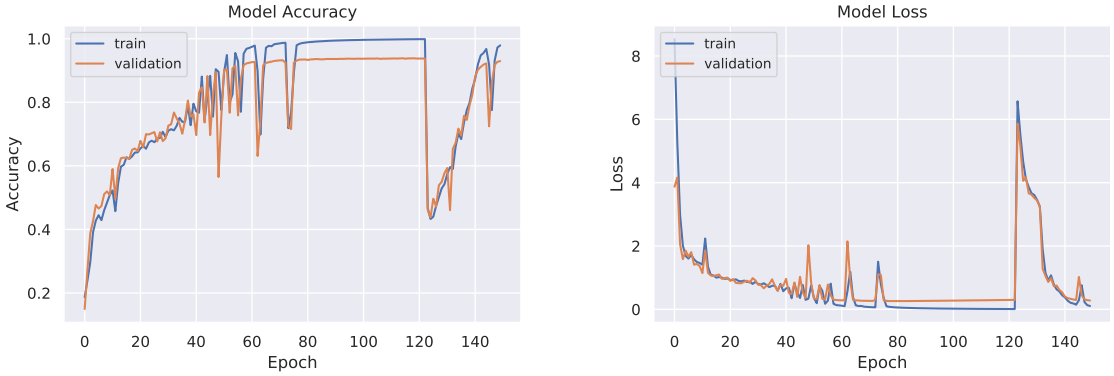
Parameter	Optimal value
batch_size	50,000
epochs	150
optimization algorithm	Nadam(lr = 0.01)
network weight initialization	normal
activation function	tanh
neurons in hidden layers	1,050

Table 6. ANN parameter adjustment

Parameter	Optimal value
batch_size	10,000
optimization algorithm	Nadam (lr = 0.001)
network weight initialization	normal
activation function	convolution layers: linear fully connected: tanh
neurons in hidden layers	1,300, 1,300, 600
filters	128
kernel_size	3

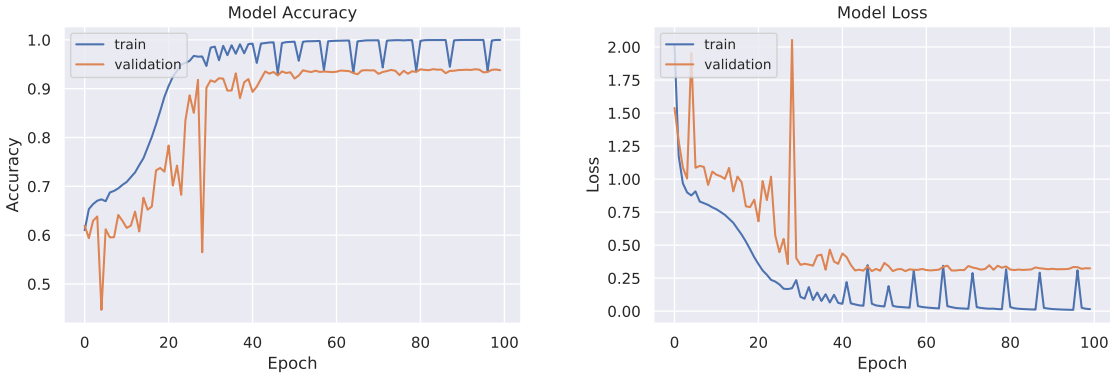
Table 7. CNN parameter adjustment

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 February 2020 doi:10.20944/preprints202002.0443.v1



(a) Accuracy **(b)** Loss

Figure 5. Accuracy and loss curves during training - ANN



(a) Accuracy **(b)** Loss

Figure 6. Accuracy and loss curves during training - CNN

The two methods mentioned in section 3 for measuring model performance, namely cross-validation and test set accuracy, were implemented using the optimized parameters for each model. The results are shown in table 8, where STD stands for standard deviation.

Model	Cross-validation Accuracy (%) ± STD (%)	Test set Accuracy (%)
K-Nearest Neighbors	95.06 ± 1.985	93.22
Logistic Regression	22.25 ± 3.349	17.87
Gaussian Naive Bayes	63.27 ± 1.099	58.56
Multilayer Perceptron	72.68 ± 1.119	69.16
Random Forest	**96.25 ± 1.638**	**95.39**
Decision Tree	94.59 ± 2.081	93.29
Artificial Neural Network	**96.09 ± 1.742**	**93.73**
Convolutional Neural Network	**95.84 ± 1.812**	**94.77**

Table 8. Results of model performance for cross-validation and test set accuracy

Summarizing the results from table 8, the LR, GNB, and MLP models displayed the lowest performance. On the other hand, the best results belong to the KNN, RF, DT, and DL models; the best three results are highlighted in yellow. For these models with the best performance, we also calculated accuracy, recall, and F1-score per class. We found that the resting hand gesture shows the best classification according to these measures. The results are found in table 9.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 28 February 2020 doi:10.20944/preprints202002.0443.v1

Model / Class	RF			ANN			CNN		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
Resting hand	0.93	1.00	0.96	0.93	0.99	0.96	0.94	1.00	0.97
Grasped hand	0.98	0.92	0.95	0.96	0.91	0.94	0.97	0.92	0.95
Wrist flexion	0.93	0.97	0.95	0.93	0.95	0.94	0.94	0.96	0.95
Wrist extension	0.97	0.96	0.96	0.94	0.94	0.94	0.95	0.95	0.95
Ulnar deviation	0.95	0.91	0.93	0.93	0.88	0.90	0.95	0.89	0.92
Radial deviation	0.96	0.96	0.96	0.94	0.94	0.94	0.94	0.95	0.95

Table 9. Performance by class

5. Discussion

In this study, we explored the performance of different ML and DL techniques in the task of classifying six hand gestures from electromyographic signals without using any feature extraction method for time series. We demonstrate the feasibility of using the information provided by each of the channels as discriminant features between the gestures. Thus, it is possible to use each record as an instance of the features matrix and obtain a more significant amount of labeled information for model training. Also, this approach allows avoiding computational costs and reduces the time required to implement the different feature extraction methods. For instance when training using Google Colaboratory platform in a GPU environment, the KNN model takes less than a minute for training and testing; RF takes approximately seven minutes, and both DL models take nearly 20 minutes. These computational time metrics are not provided by other authors, which makes it difficult to determine if our approach is faster. A novel finding was that SVMs, or at least the Python implementation by the Scikit-Learn library, require a high amount of computational resources when training over a high-dimensional feature matrix, such as the one generated here. We used ensemble classifiers (Bagging) to train up to ten models in parallel with a fraction of the dataset; however, the computational time and results were not encouraging enough to spend additional time attempting to run this model.

The best results described here are comparable to literature reports on similar problems and on the same classification task. Concerning the best results from previous studies, we highlight those by [20], who obtained 99.2% accuracy in MUAP classification using the statistical properties of each sub-band of the discrete Wavelet transform as features. On the other hand, [21] report a 98.12% accuracy for the identification of seven hand gestures using four time-domain features and stacked sparse autoencoders as a classifier. Working on a similar classification task, [27] achieved a 96.38% accuracy when using RF model to identify three hand gestures in a subject-indenpendent fashion with data from 10 subjects, the best model on their investigation matches the model that achieved the highest accuracy in our investigation.

The contribution of this study is to demonstrate the feasibility of identifying, by means of traditional ML and DL models, different hand gestures from raw EMG signals recorded from the surface of the subject's forearm, as well as to generalize among 36 subjects and still achieve high classification accuracy. An important detail that we want to mention, since we did not see the researchers suggest this topic, is the correct way to distribute the patients in the training, validation and test sets, this implies that the information of a patient is limited to only one of the sets; this way the results are reliable avoiding the bias induced by having information of the same patient in two or even all three sets. This is an important point when designing and implementing ML models since they can memorize the information in the training set, and if we have the same information in the validation or test set, we can obtain biased performance measures.

6. Conclusions

Our results confirm the feasibility of using Machine Learning and Deep Learning techniques to identify muscle activation patterns, specifically, hand movements. Further, we present a different

approach to this classification problem using raw data provided by the *MYO thalmic bracelet* device to train models with high performance levels, regardless of the subject to which the signals belong. This allows overcoming the different anatomical and biological factors between subjects, which translate into subtle differences in EMG signals. The best performances reported here were obtained by implementing Random Forest with 95.391% precision, Convolutional Neural Network with 94.77%, Artificial Neural Network with 93.73%, Decision Tree with 93.29%, and K-nearest neighbor with 93.22%. As future work, we intend to optimize and adapt our trained models for real-time classification tasks; for instance, in the control of movements of a prosthesis.

Author Contributions: conceptualization, A.M.R. and J.A.A.G.; methodology, A.M.R. and J.A.A.G.; software, A.M.R. and J.A.A.G.; validation, A.M.R., J.A.A.G., R.T.S. and S.O.A.; writing–original draft preparation, A.M.R.; writing–review and editing, C.F.J.V., J.I.P.B., R.T.S. and S.O.A.

Funding: This research was funded by Universidad Autónoma de Manizales grant number 645-2019 TD.

Acknowledgments: The authors acknowledge Universidad Autónoma de Manizales, Department of Electronics and Automation.

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

References

1. Phinyomark, A.; N. Khushaba, R.; Scheme, E. Feature Extraction and Selection for Myoelectric Control Based on Wearable EMG Sensors. *Sensors* **2018**, *18*, 1615. doi:10.3390/s18051615.
2. Lobov, S.; Krilova, N.; Kastalskiy, I.; Kazantsev, V.; Makarov, V. Latent Factors Limiting the Performance of sEMG-Interfaces. *Sensors* **2018**, *18*, 1122. doi:10.3390/s18041122.
3. Phinyomark, A.; Campbell, E.; Scheme, E., Surface Electromyography (EMG) Signal Processing, Classification, and Practical Considerations. In *Biomedical Signal Processing: Advances in Theory, Algorithms and Applications*; Naik, G., Ed.; Springer Singapore: Singapore, 2020; pp. 3–29. doi:10.1007/978-981-13-9097-5_1.
4. Phinyomark, A.; Phukpattaranont, P.; Limsakul, C. Feature reduction and selection for EMG signal classification. *Expert Systems with Applications* **2012**, *39*, 7420–7431. doi:10.1016/J.ESWA.2012.01.102.
5. Khushaba, R.N.; Kodagoda, S.; Takruri, M.; Dissanayake, G. Toward improved control of prosthetic fingers using surface electromyogram (EMG) signals. *Expert Systems with Applications* **2012**, *39*, 10731–10738. doi:10.1016/J.ESWA.2012.02.192.
6. Phukan, N.; Kakoty, N.M.; Shivam, P.; Gan, J.Q. Finger movements recognition using minimally redundant features of wavelet denoised EMG. *Health and Technology* **2019**. doi:10.1007/s12553-019-00338-z.
7. Ji, Y.; Sun, S.; Xie, H.B. Stationary Wavelet-based Two-directional Two-dimensional Principal Component Analysis for EMG Signal Classification. *Measurement Science Review* **2017**, *17*, 117–124. doi:10.1515/msr-2017-0015.
8. Ghofrani Jahromi, M.; Parsaei, H.; Zamani, A.; Stashuk, D.W. Cross Comparison of Motor Unit Potential Features Used in EMG Signal Decomposition. *IEEE Transactions on Neural Systems and Rehabilitation Engineering* **2018**, *26*, 1017–1025. doi:10.1109/TNSRE.2018.2817498.
9. Benazzouz, A.; Guilal, R.; Amirouche, F.; Hadj Slimane, Z.E. EMG Feature Selection for Diagnosis of Neuromuscular Disorders. 2019 International Conference on Networking and Advanced Systems (ICNAS), 2019, pp. 1–5. doi:10.1109/ICNAS.2019.8807862.
10. Koçer, S.; Tümer, A.E. Classifying neuromuscular diseases using artificial neural networks with applied Autoregressive and Cepstral analysis. *Neural Computing and Applications* **2017**, *28*, 945–952. doi:10.1007/s00521-016-2383-8.
11. Chen, W.; Wang, Z.; Xie, H.; Yu, W. Characterization of Surface EMG Signal Based on Fuzzy Entropy. *IEEE Transactions on Neural Systems and Rehabilitation Engineering* **2007**, *15*, 266–272. doi:10.1109/TNSRE.2007.897025.
12. Gokgoz, E.; Subasi, A. Comparison of decision tree algorithms for EMG signal classification using DWT. *Biomedical Signal Processing and Control* **2015**, *18*, 138–144. doi:https://doi.org/10.1016/j.bspc.2014.12.005.
13. Singh, S.P.; Urooj, S. Wavelets: Biomedical applications. *International Journal of Biomedical Engineering and Technology* **2015**, *19*, 1–25. doi:10.1504/IJBET.2015.071405.

14. Bengio, Y.; Goodfellow, I.J.; Courville, A. Deep Learning. Technical report, 2015.
15. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Networks* **2015**, *61*, 85–117. doi:10.1016/J.NEUNET.2014.09.003.
16. Tabares-Soto, R.; Raúl, R.P.; Gustavo, I. Deep learning applied to steganalysis of digital images: A systematic review. *IEEE Access* **2019**, *7*, 68970–68990. doi:10.1109/ACCESS.2019.2918086.
17. Orozco-Arias, S.; Isaza, G.; Guyot, R.; Tabares-Soto, R. A systematic review of the application of machine learning in the detection and classification of transposable elements. *PeerJ* **2019**, *7*, e8311. doi:10.7717/peerj.8311.
18. Ziegier, J.; Gattringer, H.; Mueller, A. Classification of Gait Phases Based on Bilateral EMG Data Using Support Vector Machines. Proceedings of the IEEE RAS and EMBS International Conference on Biomedical Robotics and Biomechatronics. IEEE Computer Society, 2018, Vol. 2018-August, pp. 978–983. doi:10.1109/BIOROB.2018.8487750.
19. Subasi, A.; Yaman, E.; Somaily, Y.; Alynabawi, H.A.; Alobaidi, F.; Altheibani, S. Automated EMG Signal Classification for Diagnosis of Neuromuscular Disorders Using DWT and Bagging. *Procedia Computer Science* **2018**, *140*, 230–237. doi:10.1016/J.PROCS.2018.10.333.
20. Subasi, A.; Yaman, E. EMG Signal Classification Using Discrete Wavelet Transform and Rotation Forest. CMBEBIH 2019; Badnjevic, A.; Škrbić, R.; Gurbeta Pokvić, L., Eds.; Springer International Publishing: Cham, 2020; pp. 29–35.
21. Zia ur Rehman, M.; Waris, A.; Gilani, S.; Jochumsen, M.; Niazi, I.; Jamil, M.; Farina, D.; Kamavuako, E. Multiday EMG-Based Classification of Hand Motions with Deep Learning Techniques. *Sensors* **2018**, *18*, 2497. doi:10.3390/s18082497.
22. Guido, S.; C. Müller, A. *Introduction to machine learning with scikit-learn*; O'Reilly Media, Inc., 2016.
23. Garreta, R.; Moncecchi, G. *Learning Scikit-Learn : Machine Learning in Python*; 2013.
24. LeCun, Y.; Kavukcuoglu, K.; Farabet, C. Convolutional networks and applications in vision. ISCAS 2010 - 2010 IEEE International Symposium on Circuits and Systems: Nano-Bio Circuit Fabrics and Systems, 2010, pp. 253–256. doi:10.1109/ISCAS.2010.5537907.
25. Pedregosa, F.; Others. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* **2011**, *12*, 2825–2830.
26. Chollet, F.; Others. Keras. https://keras.io, 2015.
27. Wahid, M.F.; Tafreshi, R.; Al-Sowaidi, M.; Langari, R. Subject-independent hand gesture recognition using normalization and machine learning algorithms. *Journal of Computational Science* **2018**, *27*, 69–76. doi:10.1016/J.JOCS.2018.04.019.