big data and cognitive computing

MDPI

Article

Hand Gesture Recognition Using Automatic Feature Extraction and Deep Learning Algorithms with Memory

Rubén E. Nogales * and Marco E. Benalcázar

Artificial Intelligence and Computer Vision Research Lab, Escuela Politécnica Nacional, Quito 170517, Ecuador; marco.benalcazar@epn.edu.ec

* Correspondence: ruben.nogales@epn.edu.ec; Tel.: +593-991977675

Abstract: Gesture recognition is widely used to express emotions or to communicate with other people or machines. Hand gesture recognition is a problem of great interest to researchers because it is a high-dimensional pattern recognition problem. The high dimensionality of the problem is directly related to the performance of machine learning models. The dimensionality problem can be addressed through feature selection and feature extraction. In this sense, the evaluation of a model with manual feature extraction and automatic feature extraction was proposed. The manual feature extraction was performed using the statistical functions of central tendency, while the automatic extraction was performed by means of a CNN and BiLSTM. These features were also evaluated in classifiers such as Softmax, ANN, and SVM. The best-performing model was the combination of BiLSTM and ANN (BiLSTM-ANN), with an accuracy of 99.9912%.

Keywords: hand gesture recognition; feature selection; leap motion controller; feature extraction; recurrent neural network

Citation: Nogales, R.E.; Benalcázar, M.E. Hand Gesture Recognition Using Automatic Feature Extraction and Deep Learning Algorithms with Memory. *Big Data Cogn. Comput.* **2023**, *7*, 102. https://doi.org/10.3390/bdcc7020102

Academic Editors: Robail Yasrab and Md Mostafa Kamal Sarker

Received: 23 March 2023
Revised: 5 May 2023
Accepted: 10 May 2023
Published: 23 May 2023

1. Introduction

THE quality and quantity of data are closely related to the performance and generalization of machine learning models. The quantity of data depends on the nature of the problem, the technology used to acquire the data, and the availability of the data. In general, machine learning models tend to perform better when trained on larger amounts of data. With more data, the model has a greater opportunity to learn the underlying patterns and relationships in the data, which can lead to better predictions and generalization with new, unseen data, while data quality means that the data are free of errors, noise, and bias, which can improve the accuracy and reliability of machine learning models. Furthermore, data quality is related to how well the features describe the problem [1–3].

In this sense, machine learning models are closely related to the problem of dimensionality. This problem arises when many features are included in machine learning models. The machine learning models can work in a scenario that is close to falling into an overfitting problem. Overfitting occurs when there is a small amount of data and a large number of features relative to the amount of training data. Furthermore, the including of too many features can also increase the computational complexity of the model, making it more difficult to train and use in practice [4–6].

In this case, it is necessary to use dimensionality reduction techniques. The techniques associated with dimensionality reduction are called feature selection and feature extraction [7]. Feature selection is the selection of the best functions that can represent the problem. While feature extraction is the process of selecting and transforming the most relevant and informative features from a dataset. Feature extraction is a critical step in machine learning models [8]. This is because the quality and relevance of the features can greatly affect the performance and accuracy of the model. In many cases, the original data may contain a large number of features that are redundant, noisy, or irrelevant to the task at hand, which can lead to overfitting and poor generalization performance. It is important

to carefully select and preprocess the most relevant and informative features for a given task to avoid these problems. This can involve domain knowledge, statistical analysis, and data visualization techniques to identify the most important features and discard the redundant or irrelevant ones. There are two ways to approach the process of obtaining the best features: firstly, manual feature extraction and, secondly, automatic feature extraction.

Automatic feature extraction is a machine learning technique that involves deep learning [9]. Automatic feature extraction for hand gesture recognition that involves the Leap Motion Controller signal could use the CNN or recurrent neural networks (RNN) [5,10]. In addition, automatic feature extraction for hand gesture recognition can save time and effort in feature engineering and can potentially discover more informative and complex features than manual methods. However, it requires large amounts of training data and computational resources and may be more difficult to interpret and understand than manual methods.

Manual feature extraction for hand gesture recognition requires domain knowledge and expertise in signal processing and feature engineering. It can be time-consuming and labor-intensive and may require iterative experimentation and refinement to identify the most informative features for a given application. However, it can also be more interpretable and understandable than automatic feature extraction techniques because the features are explicitly defined and selected based on human insight and intuition.

Related Works

In this subsection a search in the scientific literature for published work on hand gesture recognition will be performed. The input data should be spatial positions as time series retrieved from the Leap Motion Controller. Furthermore, the data should be evaluated on classifiers such as CNN-ANN, CNN-SVM, BiLSTM-ANN, and BiLSTM-SVM.

In [10] they propose a sequence of images as a time series. They also propose to use CNN in combination with LSTM because CNN extracts features and LSTM works with time series as a recurrent neural network.

In [11] the authors use a pre-trained network, such as AlexNet, to extract features and then use an SVM as a classifier. They also perform dimensionality reduction using PCA. They developed this work for American Sign Language recognition. In the same sense, in [12], they use a faster R-CNN, the aim of this study being to segment the video bond and the static gesture.

In [13], the authors proposed dynamic hand gesture recognition (DHGR) based on the convolutional neural network (CNN) and long short-term memory (LSTM). The data were collected using the Leap Motion Controller. They use a dataset that consists of data extracted from the Leap Motion Controller. The dataset consists of eight gestures: these are expanded, grab, pinch, tap, clockwise, anticlockwise, swipe left and swipe right, and named LMDI. Each gesture is performed 10 times and data are collected from 10 users. The sampling rate is 50 frames per second. The authors mention in the paper that they have a relative accuracy of 98%. However, there is no evidence that the authors are proposing an increase in data. This is since deep algorithms are classified as data hungry. In addition, data from only 10 users with 8 gestures is considered too little data to generalize.

In [14], the authors propose a hand gesture recognition system using data from the Leap Motion Controller. However, in this work, the authors propose using traditional machine learning algorithms such as KNN with k = 3. In this sense, the feature extraction is manual. The features were a set of five normalized distances between each fingertip and the palm center.

In [15], they present gesture recognition using data from a webcam for gesture control in a game and use CNN. In [16], they present motion recognition for virtual reality, they propose a BiLSTM with a fully convolutional network (FCN). In Springer, you can also find studies using Leap Motion Controller signals for medical cases such as Alzheimer's detection using a CNN.

In [17], the authors propose a hybrid model for hand gesture recognition using the Leap Motion Controller. In this model, they first use an LSTM classifier, then a BiLSTM classifier. Finally, they propose the HBU-LSTM hybrid model. They present an average recognition rate of 90%. The model is tested on two public datasets, the LeapGestureDB dataset and the RIT dataset. The LeapGestureDB dataset presents 11 gestures: thumb up, index left or right, release up or down, grip in or out, index left or right, and hand swipe left or right. Meanwhile, the RIT dataset presents 12 different gestures: grab, release, one finger tap, two finger tap, swipe, wipe, capital 'E', capital 'F', pinch, check mark, number '8′, and lowercase 'e', with 100 subjects participating.

In [18], the authors propose a new approach to extract features based on chronological pattern indexing. As a classifier, they use KNN based on Hamming distance. To evaluate the proposal, they use a RIT dataset and a LeapGestureDB dataset.

In this context, this paper proposes the performing of an evaluation of manual and automatic feature extraction for the hand gesture recognition problem using data extracted from the Leap Motion Controller. The manual extraction is performed using statistical features such as: pulse percentage rate (MYOP), detector log (LD), wavelength (WL), enhanced wavelength (EWL), difference absolute standard deviation value (DASDV), and standard deviation (SD), while the automatic features are abstracted by CNN and BiLSTM. In addition, the extracted features are evaluated on Softmax, ANN, and SVM classifiers.

It is also necessary to explain the difference between classification and recognition since, in this paper, we propose the evaluating of features in a hand gesture recognition system. The classification problem is given an observation, the classification algorithm returns a label, and this label is mapped to a set of predefined labels, finally classifying the observation to the corresponding class. Meanwhile, the recognition problem is that, in addition to classifying an observation into a predicted class, the algorithm must be able to return the time at which the gesture is performed.

The rest of the paper is organized as follows: the methodology section presents the overview of work, the dataset building, machine learning algorithms, CNN, and BiLSTM; the experimentation and result section; and finally, the paper presents conclusions and discussion.

2. Materials and Methods

This section describes the general work overview, building dataset, and machine learning and deep learning algorithms used.

2.1. General Work Overview

This work uses the spatial positions and directions retrieved by the Leap Motion Controller (LMC). The LMC represents the position of the fingertips at time t using the matrix $\mathbf{P}_t = \left[p^{(x)}_{(1,t)}, p^{(y)}_{(1,t)}, p^{(z)}_{(1,t)}; \ldots; p^{(x)}_{(5,t)}, p^{(y)}_{(5,t)}, p^{(z)}_{(5,t)}\right]^{(leap)}_t$, $\left[p^{(x)}_{(i,t)}, p^{(y)}_{(i,t)}, p^{(z)}_{(i,t)}\right]$ being the vector with the spatial positions of the i-th finger with respect to the sensor coordinate axes.

The directions of the fingertips at time t are represented using the matrix: $\mathbf{D}_t = \left[d^{(x)}_{(1,t)}, d^{(y)}_{(1,t)}, d^{(z)}_{(1,t)}; \ldots; d^{(x)}_{(5,t)}, d^{(y)}_{(5,t)}, d^{(z)}_{(5,t)}\right]^{(leap)}_t$, $\left[d^{(x)}_{(i,t)}, d^{(y)}_{(i,t)}, d^{(z)}_{(i,t)}\right]$ being the vector with the directions of the i-th finger with respect to the sensor coordinate axes.

As can be seen in the previous paragraphs, each spatial and directional position is formed by values (**X**, **Y**, **Z**). For this work, we use the data of the tips of each finger. Then, at each time, t, we obtain two matrices $\mathbf{P}_t$ and $\mathbf{D}_t$. The change in the values of the matrices at the times $t_1, t_{1+i}, \ldots t_n$ characterize the hand gesture. These data, organized in matrix form, are processed. In essence, they are processed by statistical functions that manually extract features. They are also processed by deep learning algorithms to extract features automatically. In both cases, the features obtained are evaluated in classifiers such as ANN and SVM, as shown in Figure 1.

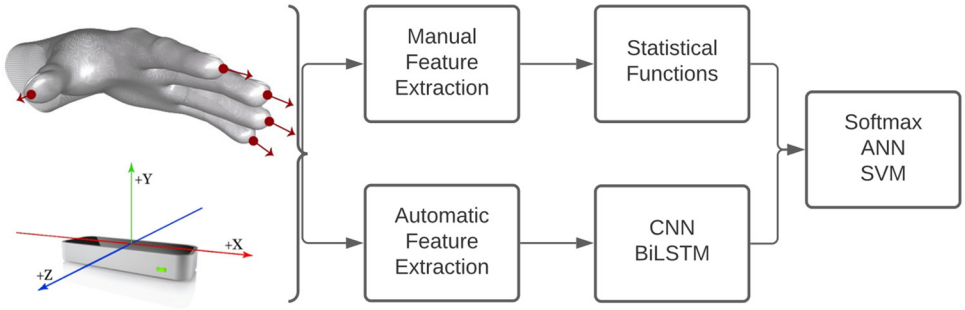

Figure 1. Overview schema about the work.

2.2. Dataset Building

The dataset was built using 56 volunteers from the Universidad Técnica de Ambato, including students and teachers, women, and men, aged 18 to 46 years old. None of the volunteers had injuries to the right upper extremity. The device used to acquire the data was a LMC. We used this device because it is small and cheaper. It is specialized to track the hand. The LMC has three LED sensors and two depth cameras. Furthermore, this device retrieves the spatial positions, directions, and velocity of hands and fingers according to the coordinate axis, whose center is the center of the device [19] Figure 2.

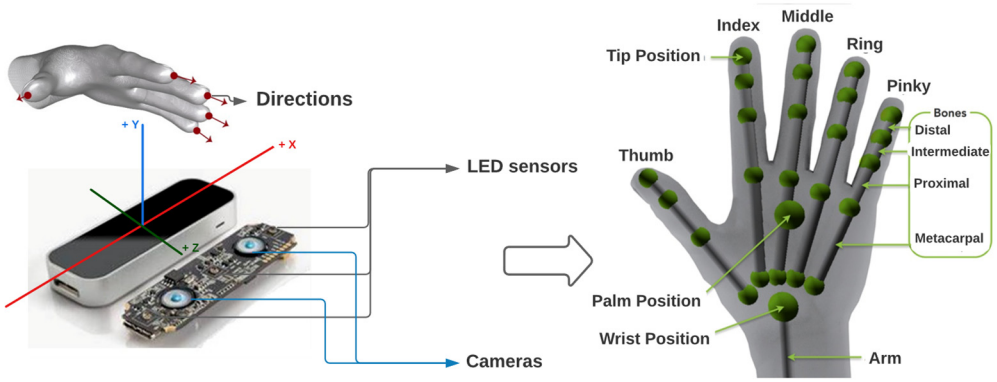

Figure 2. Leap Motion Controller description.

In addition, each user developed five gestures. Figure 3 shows the types of gestures. These gestures were open hand, fist, wave in, wave out, and pinch. The time defined for the user to perform the gesture was 5 s. In this sense, the user could perform the hand movement representing the gesture at any time during the 5 s [20].

Figure 3. Types of hand gestures.

Each user repeated each gesture 30 times. In this sense, the dataset comprised 1680 observations of each gesture, with 8400 observations total. The LMC states in its manufacturer's information manual that the sampling frequency is 200 Hz/s. However, since our dataset stored images as well as spatial positions and directions, the performance of the LMC decreased. For the development of this study, it was decided to sample at 70 Hz, since the perception of motion continuity is presented at 30 frames/sg. However, we sampled at twice the required rate. These criteria eliminated redundancy in the data. In

this context, the dataset was 8400 × 70. Each dataset instance had five fingers and three channels of X, Y, and Z data [19].

2.3. Convolutional Neural Network

A CNN is a neural network that has several convolutional layers. According to [21], a convolutional layer is a small logistic regression where the convolution mask determines the weights, and the input data values determine the constants. The mask can be a matrix or a vector. It is a matrix when the input data are two-dimensional (2D) and a vector when the input data are one-dimensional (1D). The output of the convolutional layer can be the same size as the input data if an artifice called padding is used and the stride is one. Padding consists of adding values of -1 or 0 outside the size of the input data; the number of values added depends on the size of the mask. A stride is the number of jumps in which the convolution is performed. In addition, a CNN has a pooling layer. This layer reduces the dimensionality of the input data. Finally, a CNN returns a vector that is smaller than the input data with a good abstraction or representation of the input data. Figure 4 shows the generic process of a CNN.

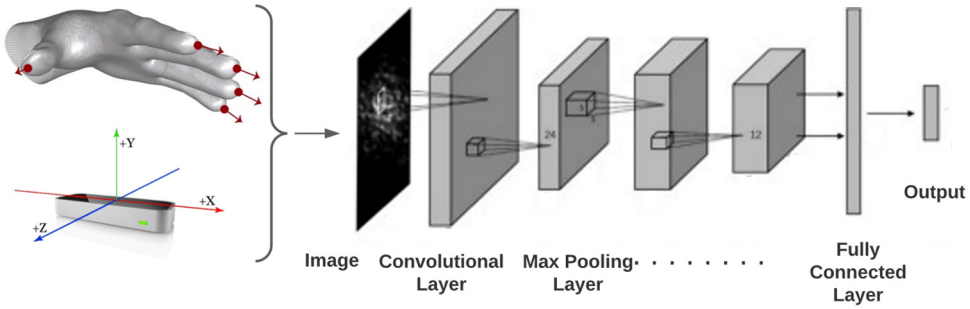

Figure 4. Convolutional neural network.

2.4. Recurrent Neural Networks

Figure 5 shows the recurrent neural network (RNN) architecture. A RNN is designed to perform prediction or classification based on a sequential dataset where certain attributes maintain a dependency. These datasets can be, for example, time series or words within a sentence. In time series, the values of time t are inferred from the values of time t_{t-1}. In the same sense, the values of time t_{t+1} are inferred from the values of time t. In this context, an RNN must be able to receive and process inputs at time t in the same order in which they are present at t_{t-1} and t_{t+1}. However, the input values at different times may be different. In this sense, a challenge for RNNs is to construct a neural network with multiple fixed parameters [22].

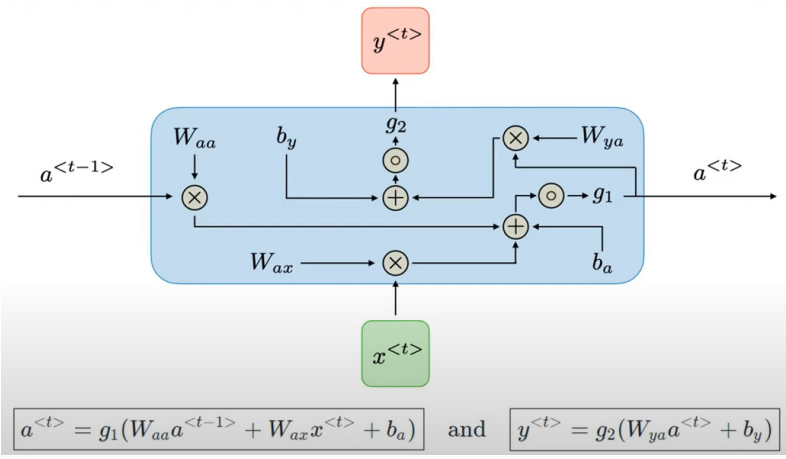

Figure 5. Internal LSTM architecture.

The types of RNNs are defined by the different architectures that these neural networks present. Between the different kinds of RNN, we have long short-term memory (LSTM) and gated recurrent unit (GRU). There is also a variant of LSTM, a bi-directional long short-term memory (BiLSTM).

RNNs can assume different types of configurations depending on the problem to be solved. These types of configurations are sequence-to-sequence, sequence-to-target, target-to-sequence, and target-to-target. The target-to-target configurations are similar to feed-forward neural networks in that this type of neural network is without memory. The target-to-sequence configurations correspond to one input in a time instant, and the model obtains many targets. The network configuration that corresponds to a sequence-to-target is one that has an input data sequence, and in which the network can predict an output target. Finally, the sequence-to-sequence configuration corresponds when the input is the sequence of data where each piece of data is a time instant, and the model forecasts a sequence of targets a label for each time instant.

3. Experiments

The experiments were carried out on an Alienware computer with six cores and twelve logical processors of the sixth generation 3.4 GHz Core i7, 32 GB of RAM, and Windows 10. The design of the experiments was based on the dataset described in the previous section. In all experiments, we measured the accuracy of testing and recognition of hand gestures. In addition, during the execution of the experiments, the processing time in the classification test was measured. Time is an important parameter in the evaluation of recognition algorithms. This is because these algorithms are used in real-time applications. Real-time, in the hand gesture recognition problem, consists in obtaining a response from the algorithm in less than 300 ms after the gesture is executed [23]. In the same context, for all cases of experimentation, training and testing was performed with cross-validation with a k = 5.

This work was oriented in two approaches. The first was *manual feature extraction*, and the second was *automatic feature extraction*. The manual feature extraction was performed by combining statistical functions, while the automatic feature extraction was performed by means of a CNN and a BiLSTM. In the same context, features were evaluated in Softmax, ANN, and SVM classifiers.

With respect to manual feature extraction, in this work, we used the following feature extraction function: pulse percentage rate (MYOP), this function is adjusted by leap motion controller signal; detector log (LD) is good at estimating the exerted force; wavelength (WL) can be calculated by simplifying the cumulative length of the waveform summation; enhanced wavelength (EWL) is an extension of WL; *p*-value, a value that is used to select a region of the signal; difference absolute standard deviation value (DASDV) is the square root of the average of the difference between the squared adjacent values; and standard deviation (SD). These feature extractions were obtained after applying the sequential feature selection method. This feature selection method had as input 17 feature extraction functions. These 17 feature extraction functions were statistical functions of central tendency that the scientific literature report as the most utilized. This was investigated in greater depth in [24].

The signal consisting of the spatial positions and directions of the fingers was used for the experiments. The data of the five fingers were used in the classification and recognition algorithms. In addition, each of these fingers represented the data of the three channels X, Y, and Z. Furthermore, we used cross-validation with a k-fold equal to five. Window splitting was the technique that was used to extract the data and feed them to the classifiers. The signal was divided into windows of 20 with a step size of 15. Each data window was fed to the classifier, which returned a label. Finally, we obtained a vector of labels. By a majority vote, it returned the label that was repeated the highest number of times, while for recognition, each window was taken and associated with the instant of time in which the gesture was developed. In this sense, the signal was detected as rest and gesture. This

signal was compared with a threshold. This threshold signal was segmented beforehand. Then, the input signals and the threshold signal were compared and, if the intersection was greater than $\tau \geq 0.25$, the gesture was recognized according to $\rho = 2x\frac{|A \cap B|}{|A|+|B|}$, if $\rho \geq \tau$ [25]. Additionally, this paper will report the average time the algorithm took to return a response after the gesture was executed. This time was measured for each sample in the test dataset. The clock was started when the sample was given to the classification test algorithm, and the clock was stopped when the classifier returned a response. Additionally, before giving the data to the algorithms, the data were mixed randomly. In this context, the algorithms used were *ANN* and *SVM.*

In this work, we used a feed-forward **ANN** with two hidden layers. The first hidden layer used ReLU as an activation function with 25 neurons. The second hidden layer used logsig as an activation function and 15 neurons. The input of ANN was the number of features according to the number of combinations of feature selection functions. As optimization function used cross-entropy, a gradient descended as a technic of weights adjusted. Additionally, ANN used 2000 iterations and a regularization factor of 1.0×10^1 Furthermore, the **SVM** classifier needed to set up its hyperparameters as the kernel and scale. In this work, the kernel was gaussian, and the scale was of order ten. Table 1 presents the results of evaluating hand gesture testing and recognition using manual feature extraction on the described ANN and SVM architecture.

Table 1. Average classification testing and recognition for manual feature extraction.

	Manual Feature Extraction		
Algorithms	**Classification Testing**	**Average Time of Classification Testing**	**Recognition**
ANN	92.936	57 ms	83.227
SVM	91.370		79.863

For **automatic feature extraction**, we performed hand gesture recognition using the CNN. The input to the network was a time series of 30 features and 70 observations. The 30 features were composed of 15 features of each finger's X, Y, and Z spatial positions and 15 directions of the tips of each finger. In this sense, the initial tensor was 30×70, and this tensor was used to feed the CNN with a 1D architecture.

The first convolution block was formed by a first convolution layer consisting of 4 filters of 1×3 with a stride of 1. The second layer was a batch normalization layer with a mean decay of 0.1. The third layer was a leaky ReLU layer with a scale of 0.01. While the fourth layer was an average pooling layer with a pool size of 5, a stride of 1, and a padding of 0.

The second convolution block consisted of 8 filters of 1×3 and a stride one. The second layer was a batch normalization layer with a mean decay of 0.1. The third layer was a leaky ReLU layer with a scale of 0.01, while the fourth layer was an average pooling layer with a pool size of 5, a stride of 1, and a padding of 0.

The convolution layers were connected to a fully connected layer with 5 resulting classes. The output of this layer was normalized by a layer normalization layer with an epsilon of 1×10^{-5}. Since it is a multi-class problem, it was connected to a softmax layer and finally connected to a classification layer. Figure 6 shows the schema of the CNN classification.

The cost function optimization technique was a stochastic gradient descent with momentum (sgdm). The CNN had configurable parameters. These parameters tuned the network to achieve better average accuracy. In this paper, MATLAB was used to design the CNN architecture. We set parameters such as learn rate schedule = piecewise. This parameter allowed us to reduce the learning rate during training. This parameter was associated with learn rate drop factor = 0.2. This parameter was a multiplicative factor that allowed us to reduce the learning rate every certain number of epochs. In this sense, learn rate drop period = 1, which meant that the learning rate was updated at the end of

each epoch. Similarly, initial learn rate = 1×10^{-4} was the value of the initial learning rate. The algorithm was trained for 20 epochs, specified by max. epochs = 20. Finally, the training algorithm grouped the data into mini-batches to evaluate the gradient of the loss function and update the weights with mini batch size = 32. Furthermore, as deep learning algorithms need a great amount of data, we used the data augmentation technique. We added random noise to the original signal, making sure that the signal did not change shape. Finally, we obtained a dataset three times larger than the original dataset. Table 2 shows the results of the classification of testing and recognition of training a CNN.

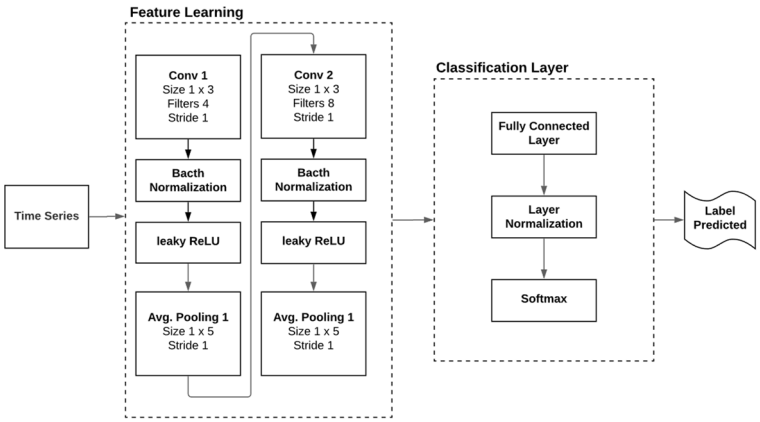

Figure 6. Schema of automatic feature extraction using CNN and softmax.

Table 2. Average classification testing and recognition for automatic feature extraction using CNN and softmax as a classifier.

Automatic Feature Extraction		
Algorithms	**Classification Testing**	**Recognition**
CNN-softmax	93.971	88.005

In addition, the layers fully connected layer, layer normalization layer, softmax, and classification layer were excluded from the model because the particular interest of the work was to obtain the features abstracted by the convolution blocks. In this sense, we invaded the CNN architecture once it abstracted the problem and obtained the features before passing them to the fully connected layer. This gave us a tensor of 8×70. This tensor was flattened, resulting in a feature vector of 560 predictors. This feature vector was fed to the SVM and ANN classification algorithms. The SVM algorithm was trained with a Gaussian kernel and a scale of 10. While the architecture of ANN consisted of 1 hidden layer, this hidden layer had 25 neurons and its activation function was a ReLU function. The output layer consisted of a softmax layer. Additionally, since the input feature vector was large, a lambda regularization factor of 2.5×10^{1} was defined. Figure 7 presents the automatic feature extraction scheme using CNN with an ANN and an SVM as classifiers.

Table 3 shows the obtained results of automatic feature extraction with an ANN and an SVM used as a classifier.

Table 3. Average classification testing for automatic feature extraction using CNN-ANN and CNN-SVM as classifiers.

Automatic Feature Extraction		
Algorithms	**Classification Testing**	**Average Processing Time**
CNN-ANN	99.795	30 ms
CNN-SVM	99.403	

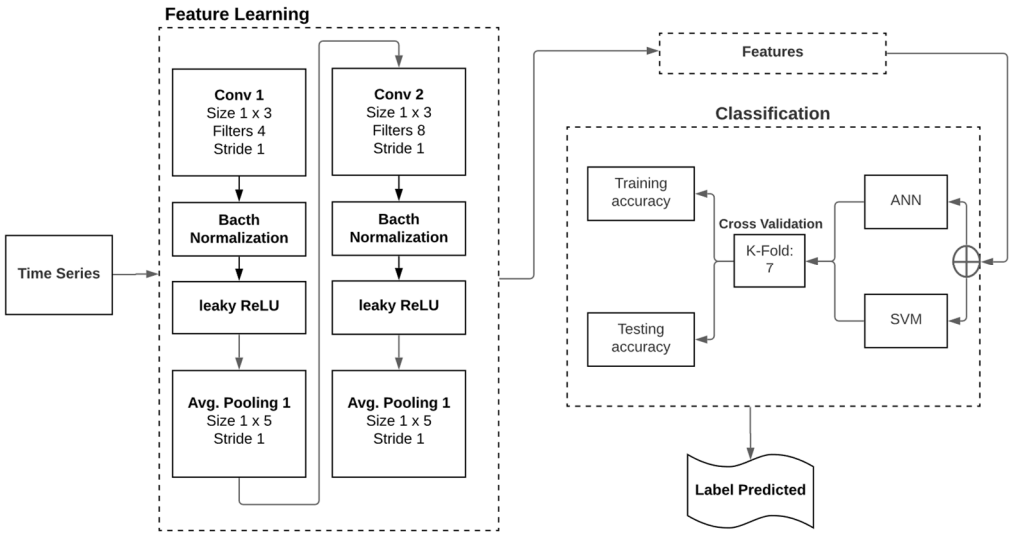

Figure 7. Schema for classification using ANN and SVM algorithms with automatic feature extraction by convolution.

In this work, we evaluated an **RNN of type BiLSTM** sequence-to-sequence. This algorithm was fed with the fingertips' X, Y, and Z spatial positions and direction values. The architecture is shown in Figure 8. This algorithm returned a vector of labels, one for each time point. This vector of labels allowed us to generate the recognition process.

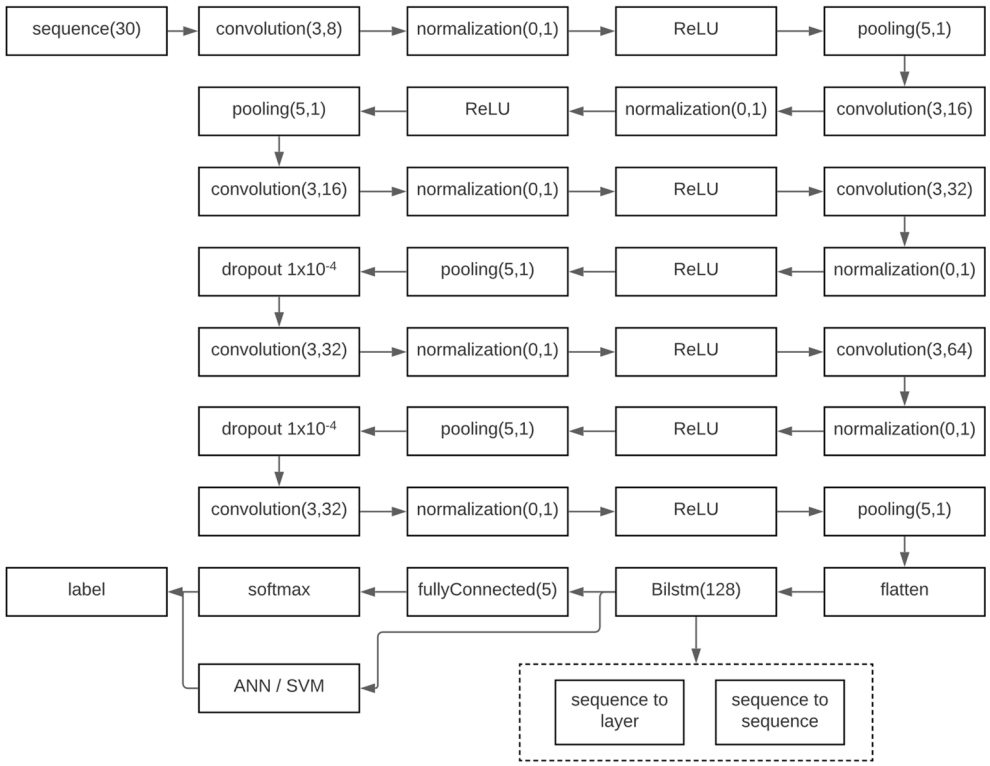

Figure 8. Schema for classification using BiLSTM with softmax, ANN, and SVM algorithms with automatic feature extraction.

The Bilstm has 34 layers, an input sequence layer, 7 convolution layers, 7 normalization layers, 7 ReLU layers, 5 pooling layers, 2 dropout layers, 1 flatten layer, 1 Bilstm, 1 fully connected layer, 1 softmax layer, and a classification layer. Figure 8 shows each of the layers with their configuration parameters. The convolution layers presented a vector of weights or convolution of 1×3 with 8, 16, 32, and 64 filters, respectively. In addition, all

normalization layers had a normalization factor of 0.1. Similarly, the pooling layers worked with the max function with a pool of 5 and jumps of 1. Furthermore, to avoid overfitting, the dropout layer was configured with a regularization factor of 1×10^{-4}. Finally, the bilstm layer presented 128 gates. Table 4 shows the classification and recognition accuracy evaluated in the softmax classifier.

Table 4. Average classification testing and recognition for automatic feature extraction using BiLSTM and softmax as a classifier.

Automatic Feature Extraction		
Algorithms	**Classification Testing**	**Recognition**
BiLSTM-softmax	95.6161	91.8601

The features automatically generated by BiLSTM were also obtained. These features were obtained at the output of the BiLSTM layer. These features fed an ANN-based classifier. In the same way, an SVM-based classifier was fed. The ANN and SVM configurations were exactly the same as those used to train the algorithms described in the previous experiments. Table 5 shows the training and testing accuracy of BiLSTM-ANN and BiLSTM-SVM.

Table 5. Average classification testing for automatic feature extraction using BiLSTM-ANN and BiLSTM-SVM as classifiers.

Automatic Feature Extraction			
	Classification		
Algorithms	**Training**	**Testing**	**Average Processing Time**
BiLSTM-ANN	99.999	99.9912	27 ms
BiLSTM-SVM	99.999	99.8840	

Figure 9 shows a summary of the accuracies obtained from the experiments with manual and automatic feature extraction. It is grouped by feature extraction methods and evaluated in classification algorithms such as ANN, SVM, and softmax.

As can be seen in Figure 9, the baseline with 95% confidence is an overlap of the algorithms. This does not indicate that there was a significant difference. In this context, we generated a statistic to determine if there were significant differences.

Firstly, a Shapiro–Wilk normality test was performed, which confirmed that the *p*-value was less than 0.05. Next, a homogeneity of variances test was performed, which also confirmed that the *p*-value was less than 0.05. Finally, an ANOVA was performed to determine if there were significant differences. Table 6 presents the statistical analysis to determine if there was a significant difference.

Table 6. Pairwise test to determine the method that has a significant difference.

	diff	**lwr**	**Upr**	**P adj**
CNN-BiLSTM	−0.1964286	−0.3639679	−0.02888928	0.0205355
Manual-BiLSTM	−6.9910714	−7.1654519	−6.81669101	0.0000000
Manual-CNN	−6.7946429	−6.9690233	−6.62026243	0.0000000

In order to perform the pairwise analysis, the results of the ANN were filtered since it was the one that reported the highest accuracy. Furthermore, the hypothesis contrast statistic shows that there was a significant difference between BiLSTM-ANN and CNN-ANN, with BiLSTM-ANN being the best. It was also observed that there was a difference between the automatic feature extraction of the two methods compared to the manual feature extraction.

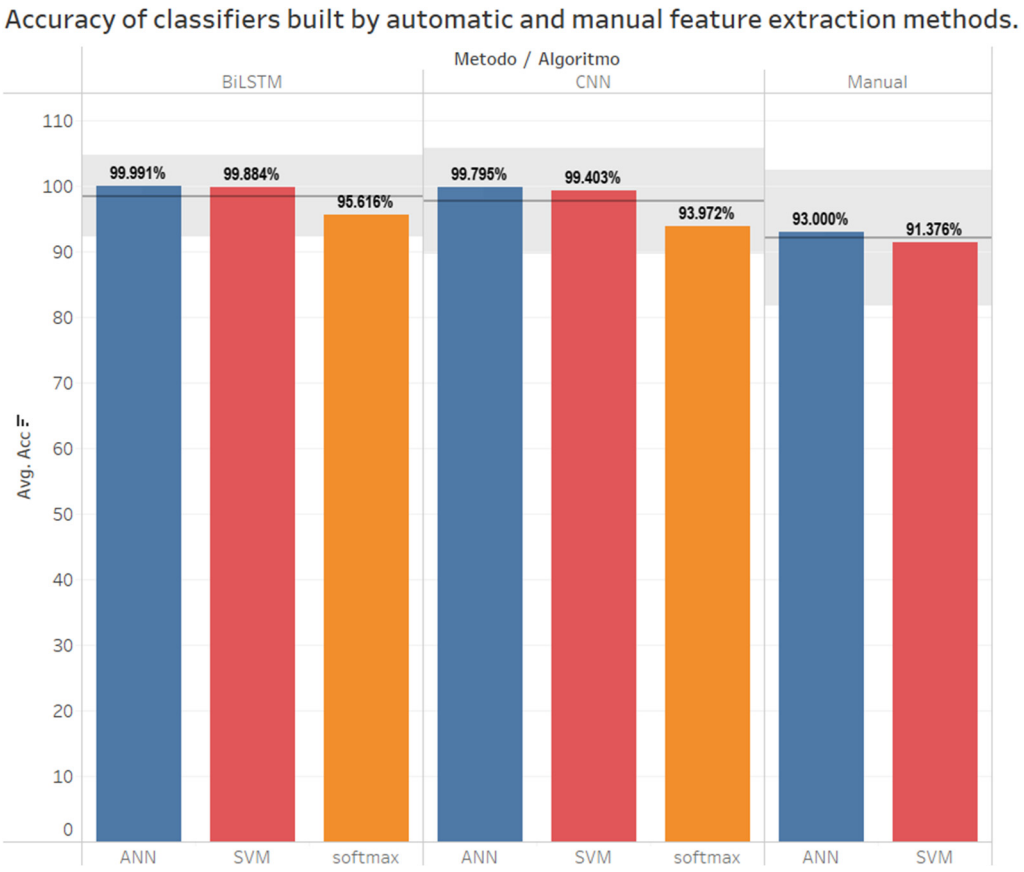

Figure 9. Accuracy of ANN, SVM, and Softmax classifiers with automatic feature extraction by CNN and BiLSTM. Accuracy of ANN and SVM classifiers with manual feature extraction.

Finally, Table 7 shows the performance of simple models with traditional algorithms and the behavior of complex models using traditional machine learning algorithms can be observed.

Table 7. Summary of the performance of simple and complex models.

		Clasification Tetsing	Recognition	Processing Time
ANN	Manual Feature extraction	92.9360%	83.227%	57 ms
SVM		91.3700%	79.863%	
CNN-softmax	Automatic Feature extraction	93.9710%	88.005%	30 ms
CNN-ANN		99.7950%		
CNN-SVM		99.4030%		
BiLSTM-Softmax		95.6161%	91.8601%	27 ms
BiLSTM-ANN		99.9912%		
BiLSTM-SVM		99.8840%		

4. Conclusions

This paper analyzes a hand gesture recognition model's classification test accuracy, recognition, and processing time. The data used to evaluate the model were acquired by the LMC. Both accuracy and processing time were measured on a model with manual feature extraction and a model with automatic feature extraction. Manual feature extraction was performed by applying statistical functions of central tendency such as MYOP, LD, WL, EWL, DASDV, and SD. These functions were chosen based on a previous study by the same authors. Furthermore, keeping the same order of feature extraction functions was necessary because the combination gives the best results. The application technique

for feature extraction was window splitting. In the same sense, two classifiers, ANN and SVM, were evaluated. The results obtained were 92.936% for ANN and 91.370% for SVM for testing while, for recognition, they were 83.227% for ANN and 79.863% for SVM. Additionally, the average classification test time was reported. The test returned 57 ms. This time was taken over the test set, from when an instance fed the classifier to when a label was returned. This evaluation showed that the model was running in real-time.

For automatic feature extraction, a CNN and a BiLSTM were used. The CNN was used to obtain an abstraction of the 1D input problem based on convolutions. The 1D input feature vector consisted of 2100 features. A total of 516 features were obtained after automatic feature extraction, which was processed by the convolutional layers of the CNN. These features fed the classifiers. The classifiers used were Softmax, ANN, and SVM. For the softmax classifier, the results obtained were 93.971% for the classification test and 88.005% for the recognition while, for the evaluation of the features extracted by the CNN and classified with an ANN (CNN-ANN), 99.795% accuracy was obtained. In the same context, the CNN-SVM evaluation reported 99.403% accuracy. The average processing time reported for the classification test with CNN-ANN and CNN-SVM was 30 ms.

A BiLSTM was used because the data obtained for gesture recognition were a time series. The configuration of BiLSTM was sequence-to-sequence. This is because the algorithm returned a label for each time point, and this was used for recognition. The BiLSTM evaluated on a softmax classifier gave a test accuracy of 95.6161% and, for recognition, it gave 91.8601%. The same features extracted by BiLSTM were evaluated on ANN; BiLSTM-ANN gives 99.9912% and BiLSTM-SVM gives 99.8840%. Likewise, the processing time in the classification test for the BiLSTM-ANN algorithm and for BiLSTM-SVM was evaluated, with an average processing time of 27 ms.

After the evaluation of the models, simple with manual feature extraction and complex with automatic feature extraction, in terms of the number of parameters and layers that needed to be adjusted, it was observed that there was no significant difference between the two complex models evaluated. It was observed that there was a significant difference between the simple models and the complex models. However, between the two complex models, CNN-ANN and BiLSTM-ANN, the difference was 0.1962% which was an almost negligible difference. It is important to understand that the CNN-ANN model is a deep model with few layers, while the BiLSTM-ANN model is a model with many layers, and it is also a model with memory. In this sense, when analyzing the CNN-ANN model, we could think about the portability of hand recognition systems with deep models. In addition, the response times of these models were less than 300 ms, which allowed us to conclude that they are models that run in real-time.

Author Contributions: All authors contributed to the study conception and design. Material preparation, data collection, and analysis were performed by R.E.N. and M.E.B. The first draft of the manuscript was written by R.E.N. and all authors commented on previous versions of the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The authors have published the hand gesture dataset, the statistical analysis, and the data obtained from the experiments at the following link. https://utaedu-my.sharepoint.com/:f:/g/personal/re_nogales_uta_edu_ec/Eq08ii7fupJCnMzRmOeZTBcB7TmA3tntuHSWX87vVk9mJw?e=qFLwrx (accessed on 22 March 2023).

Acknowledgments: The authors would like to express gratitude to the Escuela Politécnica Nacional and its doctoral program in computer science for having the best human resources for the development of its students. Thanks are also due to the Universidad Técnica de Ambato for providing the facilities for continuous improvement.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Kariluoto, A.; Kultanen, J.; Soininen, J.; Pärnänen, A.; Abrahamsson, P. Quality of Data in Machine Learning. In Proceedings of the 2021 IEEE 21st International Conference on Software Quality, Reliability and Security Companion (QRS-C), Hainan, China, 6–10 December 2021; pp. 216–221. [CrossRef]
2. Valderrama, C.E.; Marzbanrad, F.; Stroux, L.; Martinez, B.; Hall-Clifford, R.; Liu, C.; Katebi, N.; Rohloff, P.; Clifford, G.D. Improving the quality of point of care diagnostics with real-time machine learning in low literacy LMIC settings. In Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies, Menlo Park and San Jose, CA, USA, 20–22 June 2018. [CrossRef]
3. Alhazmi, K.; Alsumari, W.; Seppo, I.; Podkuiko, L.; Simon, M. Effects of annotation quality on model performance. In Proceedings of the 2021 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), Jeju Island, Republic of Korea, 13–16 April 2021; pp. 63–67. [CrossRef]
4. Kolluri, J.; Kotte, V.K.; Phridviraj, M.S.B.; Razia, S. Using Novel L1 / 4 Regularization Method. *IEEE Access*. In Proceedings of the 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184), Tirunelveli, India, 15–17 July 2020; pp. 934–938. [CrossRef]
5. Naguri, C.R.; Bunescu, R.C. Recognition of dynamic hand gestures from 3D motion data using LSTM and CNN architectures. In Proceedings of the 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), Cancun, Mexico, 18–21 December 2018; pp. 1130–1133. [CrossRef]
6. Dynamic, A.I.; Warping, T. Author's Accepted Manuscript An Image-to-Class Dynamic Time Warping Approach for both 3D Static and Trajectory Hand Gesture Recognition. *Pattern Recognit.* **2016**, *55*, 137–147. [CrossRef]
7. Normani, N.; Urru, A.; Abraham, L.; Walsh, M.; Tedesco, S.; Cenedese, A.; Susto, G.A.; O'Flynn, B. "A Machine Learning Approach for Gesture Recognition with a Lensless Smart Sensor System. In Proceedings of the 2018 IEEE 15th International Conference on Wearable and Implantable Body Sensor Networks (BSN), Las Vegas, NV, USA, 4–7 March 2018; pp. 4–7.
8. Núñez, J.C.; Cabido, R.; Pantrigo, J.J.; Montemayor, A.S.; Vélez, J.F. Convolutional Neural Networks and Long Short-Term Memory for skeleton-based human activity and hand gesture recognition. *Pattern Recognit.* **2018**, *76*, 80–94. [CrossRef]
9. Shaheen, F.; Verma, B. An ensemble of deep learning architectures for automatic feature extraction. In Proceedings of the 2016 IEEE Symposium Series on Computational Intelligence (SSCI), Athens, Greece, 6–9 December 2016. [CrossRef]
10. Gunawan, M.R.; Djamal, E.C. Spatio-Temporal Approach using CNN-RNN in Hand Gesture Recognition. In Proceedings of the 2021 4th International Conference of Computer and Informatics Engineering (IC2IE), Depok, Indonesia, 14–15 September 2021; pp. 385–389. [CrossRef]
11. Sahoo, J.P.; Ari, S.; Patra, S.K. Hand gesture recognition using PCA based deep CNN reduced features and SVM classifier. In Proceedings of the 2019 IEEE International Symposium on Smart Electronic Systems (iSES) (Formerly iNiS), Rourkela, India, 16–18 December 2019; pp. 221–224. [CrossRef]
12. Muchtar, R.A.; Yuniarti, R.; Komarudin, A. Hand Gesture Recognition for Controlling Game Objects Using Two-Stream Faster Region Convolutional Neural Networks Methods. In Proceedings of the 2022 International Conference on Information Technology Research and Innovation (ICITRI), Jakarta, Indonesia, 10 November 2022; pp. 59–64. [CrossRef]
13. Ikram, A.; Liu, Y. Skeleton based dynamic hand gesture recognition using LSTM and CNN. In Proceedings of the 2020 2nd International Conference on Image Processing and Machine Vision, Bangkok, Thailand, 5–7 August 2020; pp. 63–68. [CrossRef]
14. Clark, A.; Moodley, D. A System for a Hand Gesture-Manipulated Virtual Reality Environment. In Proceedings of the Annual Conference of the South African Institute of Computer Scientists and Information Technologists, Johannesburg, South Africa, 26–28 September 2016; pp. 1–10. [CrossRef]
15. Aggarwal, K.; Arora, A. Hand Gesture Recognition for Real-Time Game Play Using Background Elimination and Deep Convolution Neural Network. In *Virtual and Augmented Reality for Automobile Industry: Innovation Vision and Applications*; Hassanien, A.E., Gupta, D., Khanna, A., Slowik, A., Eds.; Springer International Publishing: Cham, Switzerland, 2022; pp. 145–160.
16. Thenmozhi, R.; Aslam, S.M.; Jovith, A.A.; Avudaiappan, T. Modeling of Optimal Bidirectional LSTM Based Human Motion Recognition for Virtual Reality Environment. In *Virtual and Augmented Reality for Automobile Industry: Innovation Vision and Applications*; Hassanien, A.E., Gupta, D., Khanna, A., Slowik, A., Eds.; Springer International Publishing: Cham, Switzerland, 2022; pp. 161–174.
17. Ameur, S.; Khalifa, A.B.; Bouhlel, M.S. A novel hybrid bidirectional unidirectional LSTM network for dynamic hand gesture recognition with Leap Motion. *Entertain. Comput.* **2020**, *35*, 100373. [CrossRef]
18. Ameur, S.; Khalifa, A.B.; Bouhlel, M.S. Chronological pattern indexing: An efficient feature extraction method for hand gesture recognition with Leap Motion. *J. Vis. Commun. Image Represent.* **2020**, *70*, 102842. [CrossRef]
19. Nogales, R.; Benalcazar, M.E.; Toalumbo, B.; Palate, A.; Martinez, R.; Vargas, J. Construction of a Dataset for Static and Dynamic Hand Tracking Using a Non-invasive Environment. In *Advances and Applications in Computer Science, Electronics and Industrial Engineering: Proceedings of CSEI 2020*; Springer: Singapore, 2021; Volume 1307 AISC, pp. 185–197. [CrossRef]
20. Nogales, R.; Benalcázar, M. Real-Time Hand Gesture Recognition Using the Leap Motion Controller and Machine Learning. In Proceedings of the 2019 IEEE Latin American Conference on Computational Intelligence (LA-CCI), Guayaquil, Ecuador, 11–15 November 2019; pp. 1–6.
21. Xin, R.; Zhang, J.; Shao, Y. Complex network classification with convolutional neural network. *Tsinghua Sci. Technol.* **2020**, *25*, 447–457. [CrossRef]

22. Gouhara, K.; Watanabe, T.; Uchikawa, Y. Learning Process. In Proceedings of the 1991 IEEE International Joint Conference on Neural Networks, Singapore, 8–21 November 1991; pp. 746–751. [CrossRef]
23. Benalcázar, M.; Motoche, C.; Zea, J. Real-Time Hand Gesture Recognition Using the Myo Armband and Muscle Activity Detection. In Proceedings of the 2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM), Salinas, Ecuador, 16–20 October 2017. [CrossRef]
24. Nogales, R.; Benalcázar, M. Analysis and evaluation of feature selection and feature extraction methods. under review.
25. Nogales, R.; Benalcázar, M. Real-Time Hand Gesture Recognition Using KNN-DTW and Leap Motion Controller. In Proceedings of the Information and Communication Technologies: 8th Conference, TICEC 2020, Guayaquil, Ecuador, 25–27 November 2020; Springer International Publishing: Cham, Switzerland, 2020.