






Article

Handwriting Recognition Based on 3D Accelerometer Data by Deep Learning

Pedro Lopez-Rodriguez ^{1,2} , Juan Gabriel Avina-Cervantes ¹ , Jose Luis Contreras-Hernandez ¹ ,
Rodrigo Correa ³  and Jose Ruiz-Pinales ^{1,*} 

¹ Digital Signal Processing and Telematics, Engineering Division of the Campus Irapuato-Salamanca (DICIS), Universidad de Guanajuato, Carr. Salamanca-Valle de Santiago Km 3.5 + 1.8, Palo Blanco, Salamanca 36885, Mexico; p.lopez.rodriguez@ugto.mx or prodriguez@upgto.edu.mx (P.L.-R.); avina@ugto.mx (J.G.A.-C.); jose.contreras@ugto.mx (J.L.C.-H.)

² Automotive Engineering Department, Universidad Politécnica de Guanajuato, Av. Universidad Sur 101, Cortazar 38496, Mexico

³ Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones, Universidad Industrial de Santander, Cra 27 Calle 9, Bucaramanga 680002, Colombia; crcorrea@saber.uis.edu.co

* Correspondence: pinales@ugto.mx; Tel.: +52-4646479940 (ext. 2402)

Abstract: Online handwriting recognition has been the subject of research for many years. Despite that, a limited number of practical applications are currently available. The widespread use of devices such as smartphones, smartwatches, and tablets has not been enough to convince the user to use pen-based interfaces. This implies that more research on the pen interface and recognition methods is still necessary. This paper proposes a handwritten character recognition system based on 3D accelerometer signal processing using Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM). First, a user wearing an MYO armband on the forearm writes a multi-stroke freestyle character on a touchpad by using the finger or a pen. Next, the 3D accelerometer signals generated during the writing process are fed into a CNN, LSTM, or CNN-LSTM network for recognition. The convolutional backbone obtains spatial features in order to feed an LSTM that extracts short-term temporal information. The system was evaluated on a proprietary dataset of 3D accelerometer data collected from multiple users with an armband device, corresponding to handwritten English lowercase letters (a–z) and digits (0–9) in a freestyle. The results show that the proposed system overcomes other systems from the state of the art by 0.53%.

Keywords: 3D accelerometer data; handwritten character recognition; Convolutional Neural Networks (CNN); Long Short-Term Memory (LSTM); 3D signal processing



Citation: Lopez-Rodriguez, P.; Avina-Cervantes, J.G.; Contreras-Hernandez, J.L.; Correa, R.; Ruiz-Pinales, J. Handwriting Recognition Based on 3D Accelerometer Data by Deep Learning. *Appl. Sci.* **2022**, *12*, 6707. <https://doi.org/10.3390/app12136707>

Academic Editor: José Salvador Sánchez Garreta

Received: 22 May 2022

Accepted: 30 June 2022

Published: 2 July 2022

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



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Online handwriting recognition has been the subject of research for a long time, and part of the technology has found commercial application. However, the limited number of success stories from the market suggests that more research on the pen interface and the recognition methods is still necessary [1].

Currently, it is common to find wearable devices with a wide range of embedded sensors, which can be used in human activity and behavior studies. Such devices have contributed to the development of new smart applications to use that kind of data. The number of wearables with 3D accelerometer sensors available in the market, and their low cost makes it possible to develop new friendly and non-invasive human–computer interfaces. Such technology allows users to write customarily and freely on smartphones, smartwatches, smart TVs, computers, etc. In order to attract the user's attention to the pen interface, fast and accurate handwriting recognition interfaces are highly required.

Some systems using 3D accelerometer data for human activity, handwriting and sign language recognition have been developed [2–8]. Online handwriting recognition is still

challenging due to the variability of stroke order, shape and style of handwriting [1]. Some approaches recover the trajectory of the handwritten character to convert it to image. In this case, to cope with variability, robust feature extraction methods can be employed [9,10]. Another option is to perform feature extraction directly in the time domain using a Convolutional Neural Network [11].

This paper proposes a novel deep learning approach to online handwriting character recognition. The method is focused on efficient feature extraction, classification, and evaluation modules using 3D accelerometer data. The recognition system is based on combining the CNN and LSTM architectures. CNN architecture was used for feature extraction, which allows transforming 3D raw accelerometer data into a feature vector sequence. Finally, an LSTM architecture is trained to recognize the patterns delivered by the CNN. In this way, the classical ad hoc techniques in feature extraction and classification are avoided, permitting artificial intelligence (AI) to make these tasks.

The main contributions reported in this paper comprise:

1. A proprietary dataset of 3D accelerometer data corresponding to multi-stroke freestyle handwritten lowercase letters and digits. Unlike previous approaches, we do not impose restrictions on handwriting style and number and order of the strokes.
2. Three neural network architectures (CNN, LSTM, and CNN-LSTM) were proposed. In the last architecture, a CNN was used for feature extraction to encode the global characteristics of raw 3D accelerometer data together with an LSTM for sequence processing and classification.

The remainder of this paper is organized as follows. Section 2 provides a summary of representative works on online handwritten character recognition. Section 3 includes a detailed description of the proposed method. Section 4 presents the designed experiments and numerical results. Finally, Section 5 announces the conclusions derived from this study.

2. Related Work

Motion sensor data, such as accelerometer and gyroscope data, have been used recently for different tasks such as sign language recognition [2], and online handwritten character recognition [12]. To successfully carry out handwriting recognition, it is necessary to synchronize two main steps, the data acquisition, and identification processes. In the former, some works perform data acquisition by using a specially designed device [12–15]. Nevertheless, nowadays, many wearable devices such as smartphones or smartwatches are equipped with accelerometers, gyroscopes, magnetometers, and other wearables such as the shimmer3 IMU or the MYO armband. In the latter, handwriting recognition takes charge of processing collected data from the motion sensors to recognize handwritten characters correctly.

One clue problem found in online handwriting recognition is related to the length of the sequences for characters of the same class, even those produced by the same person. These variations are related to the writing speed of a character, which simultaneously changes the acceleration of the hand movement. On these grounds, to expect sequences with the same length for the same character class is difficult.

However, to eliminate unnecessary data in the sequences for each letter, it is convenient to extract or segment only the part of the signal during which the handwriting process is carried out. A segmentation approach was proposed in [15] and tested in [4,16]. In this process, only the samples belonging to the handwriting should be considered; otherwise, samples not belonging to the written character will be included.

Another approach to only collect the signal during the handwriting process is to consider the information of pen-up and pen-down movements. In some approaches, this task has been addressed using a camera to track hand movements. Afterward, the segmentation is conducted by hand or a specialized writing device (e.g., glove, pen, marker). In this regard, Roy et al. [17] presented a solution to handwriting recognition by developing a user interface to compute numeral recognition in air writing by using a Convolutional Neural Network (CNN). First, they used a fixed-color marker to write in front of a camera,

which was followed by a color-based segmentation to identify and track the trajectory of the pen-marker. Next, the classification was carried out by a trained CNN.

An accelerometer-based digital pen for handwritten digits and gesture trajectory recognition applications was proposed by Wang and Chuang [14]. This pen is based on a triaxial accelerometer to collect the acceleration motion data of the hand; time and frequency domain features are extracted from the obtained acceleration signals. Next, most discriminant features were selected using a hybrid method; kernel-based class separability was applied to select significant features, and Linear Discriminant Analysis (LDA) was used to reduce the dimensionality. Both algorithms were dedicated to training a probabilistic neural network for recognition.

Amma et al. [15] proposed a wearable input system for handwriting recognition using an accelerometer and a gyroscope to capture the handwriting gestures. Later, these data were processed using Support Vector Machines (SVM) to identify which data segments contain handwriting. Subsequently, Hidden Markov Models (HMMs) were used for recognition.

Kim et al. [4] classified 26 lowercase letters of the English alphabet using 3D gyroscope data instead of 3D accelerometer data, and the Dynamic Time-Warping (DTW) algorithm was used for recognition.

Agrawal et al. [18] proposed a PhonePoint Pen system that uses the built-in accelerometers found in mobile phones to recognize handwritten English characters.

Additionally, Li et al. [19] presented a hand gesture recognition based on mobile devices using the accelerometer and gyroscope sensors. Here, the authors applied a filtering process as preprocessing. They also proposed two deep models: a Bidirectional Long-Short Term Memory (BiLSTM) and a Bidirectional Gated Recurrent Unit (BiGRU) using the Fisher criterion, termed F-BiLSTM and F-BiGRU, respectively.

Ardüser et al. [20] transformed the accelerometer and gyroscope signals of a smartwatch into a particular coordinate system on a whiteboard and used the DTW algorithm for recognition.

In the same context, Kwon et al. [21] classified ten hand gestures using a CNN model with six convolutional layers. Lin et al. [22] proposed the system SHOW (Smart Handwriting on Watches), where the users write on horizontal surfaces. Unfortunately, the users need to use the elbow as a support point; due to such inconvenient, this process is not recognized as freestyle handwriting.

Additionally, to show the possibility of motion sensor based eavesdropping on handwriting, Xia et al. [23] introduced a MotionHacker system using a smartwatch application to record the evolution in hand movement. The system requires a preprocessing stage to proceed with the feature extraction, which allows recognizing each letter by training a random forest classifier.

Concerning technologies, some authors have preferred to use the MYO armband [2,7,24]. Meanwhile, others use the shimmer device [25–27]. Our study used an MYO armband to capture the acceleration motion of real dynamic handwriting. Table 1 presents a summary of representative state-of-the-art works related to the proposed framework.

LSTM networks have been applied successfully to sequence classification in other domains. For instance, Ojagh et al. [28] proposed a method for air quality prediction using data from air quality sensors distributed in Calgary, Canada. An edge-based component exploiting both temporal and spatial information was used to clean raw data and fill in missing sensor values. Then, an LSTM network was used for prediction. In another work, Sa-nguannarm et al. [29] proposed a human activity detector based on accelerometer data and an LSTM network. Livieris et al. [30] proposed a CNN-LSTM model for gold price time-series forecasting. Elmaz et al. [31] proposed a method for the prediction of indoor temperature by using a CNN-LSTM architecture.

Table 1. Summary of related approaches.

Algorithm	Methodology	Limitations
Digital Pen [12]	3D accelerometer signals are converted to image which is recognized by a neural network	Ten digits written in a special single stroke font
WIMU-Based Hand Motion Analysis [13]	Movement and attitude features are extracted from motion sensor and magnetometer signals and recognition is completed by DTW	English lowercase letters and digits written in a special single stroke font
Accelerometer-Based Digital Pen [14]	3D accelerometer signals are recognized by a PNN	Ten digits written in a special single stroke font
Air writing [15]	3D accelerometer and gyroscope signals are recognized by HMMs	English uppercase letters and words
Gyroscope-equipped smartphone [4]	3D gyroscope signals are recognized by stepwise lower-bounded dynamic time warping	English lowercase letters written with a smartphone grabbed as a pen
Marker-based Air writing [17]	Handwriting is captured from motion of a marker in a video and recognition is completed by a CNN	Ten digits written in a single stroke
PhonePoint Pen [18]	Basic strokes are detected from 3D accelerometer signals by correlation with templates and handwritten characters are recognized by juxtaposition of basic strokes	English letters and digits written using basic strokes, smartphone grabbed as a pen
Deep Fisher Discriminant Learning [19]	3D accelerometer and gyroscope signals are recognized as hand gestures by an F-BiGRU	Six uppercase English letters and six digits written in a predefined stroke ordering
Motion data from smartwatch [20]	Features are extracted from accelerometer and gyroscope signals and letter recognition is done by DTW	English uppercase letters written on a whiteboard
SHOW [22]	Features are extracted from accelerometer and gyroscope signals and recognition was tested with seven machine learning algorithms	English letters and digits written on a horizontal surface with the elbow as support point
MotionHacker [23]	After preprocessing and segmentation, features are extracted from accelerometer and gyroscope signals and letter recognition is performed by random forest classifier	Demonstration of motion sensors-based eavesdropping on handwriting
AirScript [7]	Recognition is completed by a fusion of a CNN, and two GRU networks using as input an image derived from 2-DifViz features, post-processed 2-DifViz features and standardized raw data, respectively	Ten digits written in the air
Finger Writing with Smartwatch [25]	Energy, posture, motion shape and motion variation features are extracted from accelerometer and gyroscope signals, and three classifiers are tested for recognition (Naive Bayes, linear regression and decision trees)	English lowercase letters written on a surface
Trajectory-Based Air Writing [32]	Trajectory of handwriting with fingertip using a video camera and recognition was completed by a CNN and an LSTM	Ten digits written with a predefined stroke ordering
Air Writing with Interpolation [33]	Motion sensor data are interpolated and then recognized by a 2D-CNN	Uses datasets of others

3. CNN and LSTM for Sequence Recognition

In the following, we elaborate on the justification for using CNN and LSTM networks for sequence recognition. First, one of the advantages of Convolutional Neural Networks (CNN) is that they can perform feature extraction directly from raw data. They perform adaptive feature extraction because they learn to extract the features that are more suitable for the task at hand. Another advantage is that they are able to tolerate a moderate amount of distortion and noise. Another advantage is that they present a good generalization ability because of the use of shared weights, which allows them to incorporate prior knowledge of the problem to be solved [34]. Long Short-Term Memory (LSTM) networks are one of

the best architectures for sequence processing. This is because LSTM networks have the ability to extract long-range dependencies. In fact, these networks were proposed to solve the vanishing gradient problem faced by Recurrent Neural Networks (RNN) [35].

The combination of spatial and temporal feature learning is crucial to reliably performing motion sequence recognition. A hybrid CNN-LSTM network is used in this work to extract and exploit these two types of features. The convolutional part obtains spatial features, while the LSTM modules extract short-term temporal information.

In this section, the CNN and LSTM architectures are presented and described in detail.

3.1. Convolutional Neural Networks

Convolutional Neural Networks (CNN) are a special kind of neural network whose structure was inspired by the biological visual perception system [36]. These neural networks are specialized for feature extraction in 2D systems (e.g., matrix systems); however, their use also has been extended to 1D systems (e.g., sequences). Unlike traditional pattern recognition methods, it is unnecessary to implement or design a feature extractor to gather discriminant information, discarding irrelevant features, and categorize the selected feature vectors into classes for training a supervised classifier. In this scheme, a CNN could be trained with almost raw data; that is, a CNN is a set of layers that transforms the input data into the output class or prediction.

However, the CNN input data in a 2D or 1D system must be nearly normalized in magnitude and centered. CNNs are characterized by the use of several layers, such that a model with depth d can be defined as:

$$C = L_{n_{d-1}}^{n_d} \circ L_{n_{d-2}}^{n_{d-1}} \circ \dots \circ L_{n_1}^{n_2} \circ L_{n_0}^{n_1}, \quad (1)$$

where each L represents a convolutional, a pooling or a fully connected layer. The lower and upper indices represent the input and output size of the layer, respectively. Typically, the output of the convolution operators is passed through an activation function f to form the feature map for the next layer.

3.1.1. Convolutional Layer

Convolutional layers are the core of CNN architectures; the convolution is an operation between two functions, which is mathematically denoted by

$$c(t) = (x * y)(t) = \int_{-\infty}^{\infty} x(\tau)y(t - \tau)d\tau, \quad (2)$$

where $*$ represents the convolution operator. The first argument $x(t)$ is the measured input data and the second $y(t)$ is the kernel or the convolutional filter. The output $c(t)$ is the feature maps. Equation (2) represents the convolution defined in continuous-time terms; however, the data obtained from all sensors must be discretized to be processed on a computer.

In practice, the convolution operator in discrete-time is defined as

$$c(n) = (x * y)(n) = \sum_{k=-\infty}^{\infty} x(k)y(n - k). \quad (3)$$

The inputs are the multidimensional data, and the kernel contains the multidimensional parameters. Such parameters are adapted according to the processed data. In this paper, the data sequences are taken as a bidimensional data array.

3.1.2. Activation Function

A convolutional layer is commonly followed by a nonlinear activation function to increase the capacity of the neural network. Such an activation function helps the neuron to react or not through a nonlinear transformation acting over the input signal. Many

activation functions are defined in neural networks, but the most commonly used are the Gaussian, Sigmoid, Maxout, Hyperbolic Tangent, Leaky ReLU, and ReLU.

The ReLU function $f(x) = \max(0, x)$ is probably the most recurrent in the literature. It is computationally efficient because all neurons are not activated at the same time, converting the negative inputs to zero. Consequently, the neurons are not all activated, allowing the network to converge faster than for other activation functions.

3.1.3. Pooling Layer

This layer can reduce the dimensionality of the input data and usually comes after a convolutional layer. The dimensionality reduction can be of two forms, the average (average pooling) and the maximum within a rectangular neighborhood (Max Pooling). In both cases, the pooling layer helps to have an almost invariant representation of the input data after a small coordinate translation, which is very helpful if it is wanted to know whether some feature is prevailing or not. Pooling is implemented by an algorithm that compresses or generalizes the information, and generally, this layer can reduce the overfitting in the training stage.

3.1.4. Fully Connected Layer (Dense)

After the feature extraction performed by convolutional and pooling layers, it is necessary to recognize such features, which takes place at the last operative block of the CNN architecture using one or more fully connected layers. In 1D systems, the output of the previous layers coming to a fully connected layer is flattened into a vector that will be used to classify the input data into predefined classes. A Soft-Max layer at the top of the network computes the probabilities of each class.

3.2. Long Short-Term Memory Neural Networks

Long Short-Term Memory Neural Networks (LSTM) are a special kind of recurrent neural network (RNN), specialized in processing sequential data. LSTMs have recurrent connections, allowing the learning of long-term dependencies and, consequently, remembering information for long periods of time. The LSTM network core is based on a special unit known as the memory block [35], which is controlled by three structures (or gates): the forget gate f_t , input gate i_t , and output gate O_t .

Figure 1 shows a basic memory block.

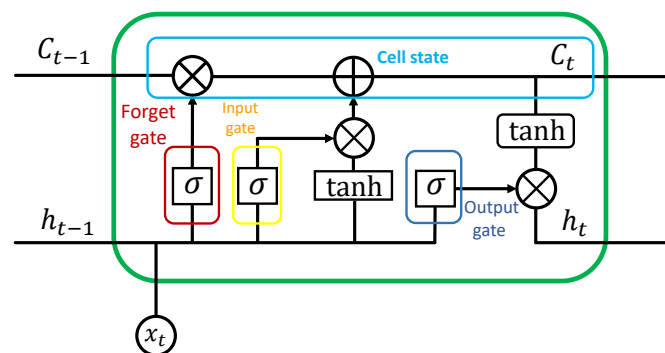


Figure 1. The memory cell and the gates of an LSTM memory block.

Each memory block has two sources of information at each unit time t , the current sample x_t , and the previous memory block state h_{t-1} . This information is processed by the forget gate (a sigmoid layer) to decide what information is ignored from the previous cell state C_{t-1} . The next step is deciding which new information will be stored in the current cell state. This process is performed at the input gate using the previous state and new data.

Subsequently, the information passes through two functions: the sigmoid activation function i_t and the \tanh function \tilde{C}_t . These two partial results are next multiplied. Currently,

the previous state C_{t-1} is updated to a new state or cell state C_t . Finally, the output of the LSTM network h_t is composed of the product of the output gate O_t and the cell state C_t that passes through the \tanh activation function. The overall process in a memory block is summarized as follows:

$$\begin{aligned} i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}C_{t-1} + b_i), \\ f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}C_{t-1} + b_f), \\ C_t &= f_t C_{t-1} + i_t \tilde{C}_t, \\ O_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}C_{t-1} + b_o), \\ h_t &= O_t \tanh(C_t), \end{aligned} \quad (4)$$

where $\tilde{C}_t = \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$, $W_{\{\}}\}$, and $b_{\{\}}\}$ denote the weight matrices and bias terms, respectively.

3.3. Implemented Architecture

In this study, to perform handwriting recognition using accelerometer data, three different architectures were implemented and evaluated: a CNN, an LSTM, and an hybrid CNN-LSTM.

The CNN architecture (Figure 2) basically consists of three convolutional layers followed by *ReLU* activation functions. Indeed, in the first and second layers, 16 convolutional filters are used to extract the same number of features from data, where the size of the filters is 1×3 , and the stride is set to one. After the first two convolutional layers, max-pooling is applied to reduce the dimensionality by half. Additionally, a third convolutional layer was implemented using 32 convolutional filters with the same size and stride as in previous layers, applying the ReLU activation function. A dropout technique with a dropout rate of 0.1 was used in each convolutional layer to prevent overfitting. Finally, a fully connected (Dense) layer is used with a Soft-Max activation function to compute the probability distribution over 36 classes. The proposed LSTM architecture employs 250 memory blocks (Figure 1). This is followed by a fully connected layer with a Soft-Max activation function to compute the probability distribution over all classes. In the hybrid CNN-LSTM model (Figure 3), the output of the last convolutional layer of the CNN is connected to the LSTM for recognition and classification.

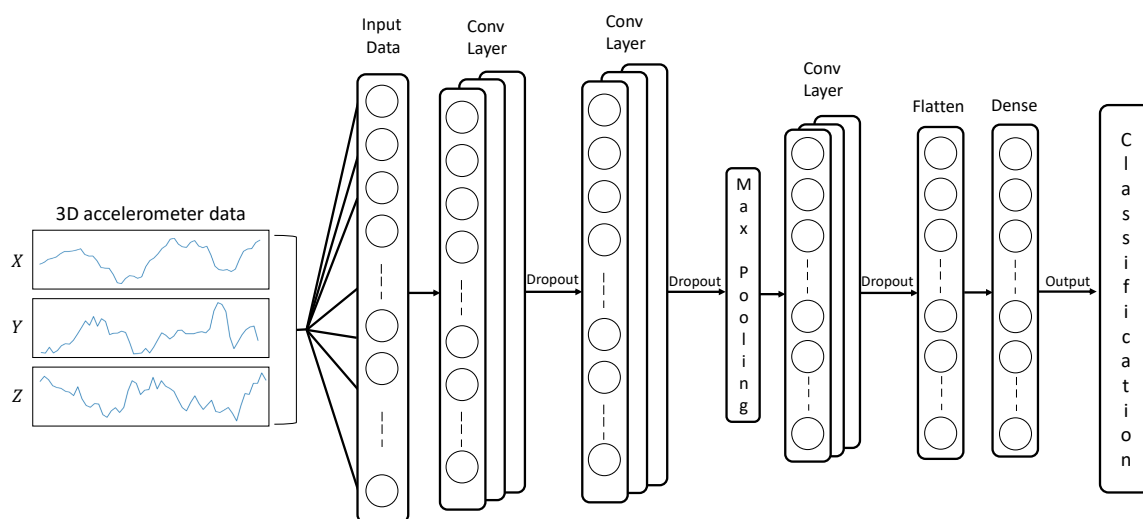


Figure 2. A CNN architecture with three convolutional layers, max pooling, and a fully connected layer. Each convolutional layer and dense layer is followed by an activation function.

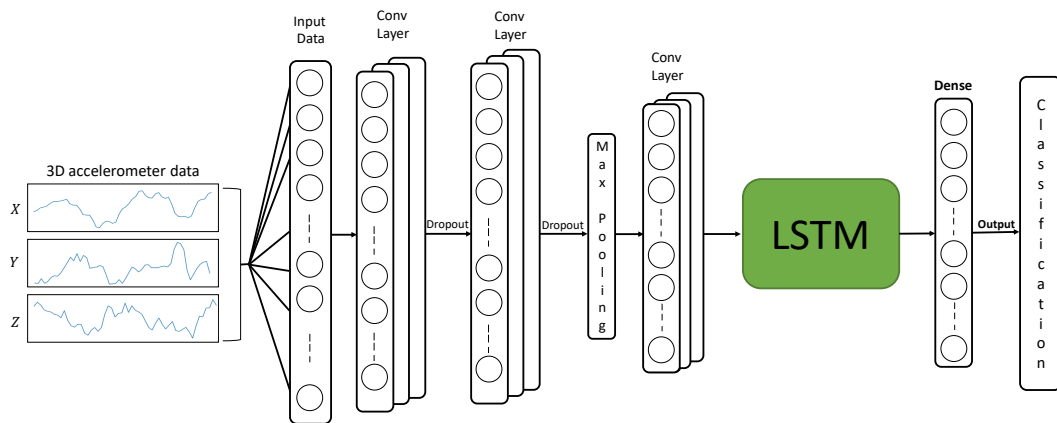


Figure 3. CNN-LSTM architecture.

The input data for the networks consist of the 3D accelerometer data represented in (x, y, z) Cartesian coordinates.

The evaluated architectures were trained using the Adam optimization algorithm to find the minimum of the proposed stochastic objective function, and the network parameters were updated using the backpropagation algorithm.

Tables 2–4 show the parameters of the three architectures.

Table 2. Number of parameters of the CNN architecture.

Layer (Type)	Output Shape	Parameters
Input	(None, 116, 3)	0
Conv1D	(None, 114, 16)	160
Dropout	(None, 114, 16)	0
Conv1D	(None, 112, 16)	784
Dropout	(None, 112, 16)	0
MaxPooling	(None, 56, 16)	0
Conv1D	(None, 54, 32)	1568
Dropout	(None, 54, 32)	0
Flatten	(None, 1728)	0
Dense	(None, 36)	62,244
Total		64,756

Table 3. Number of parameters of the LSTM architecture.

Layer (Type)	Output Shape	Parameters
Input	(None, 116, 3)	0
LSTM	(None, 116, 250)	254,000
Flatten	(None, 1728)	0
Dense	(None, 36)	1,044,036
Total		1,298,036

Table 4. Number of parameters of the CNN-LSTM architecture.

Layer (Type)	Output Shape	Parameters
Input	(None, 116, 3)	0
Conv1D	(None, 114, 16)	160
Dropout	(None, 114, 16)	0
Conv1D	(None, 112, 16)	784

Table 4. Cont.

Layer (Type)	Output Shape	Parameters
Dropout	(None, 112, 16)	0
MaxPooling	(None, 56, 16)	0
Conv1D	(None, 54, 32)	1568
Dropout	(None, 54, 32)	0
LSTM	(None, 250)	283,000
Dense	(None, 36)	9036
Total		294,548

4. Experiments and Numerical Evaluation

4.1. Hardware and Data Collection

Handwriting movement was collected using a touchscreen laptop to write the characters and the specialized MYO armband to obtain the dynamic data. The online handwriting database comprises 3D accelerometer data corresponding to twenty-six lowercase (a–z) letters from the English alphabet and ten Arabic numerals (0–9).

Figure 4 shows an example of calligraphic strokes for each character from the database; these points were captured from the touchscreen during a writing task. Each handwritten character class has 399 samples captured at different speeds and sizes; besides, they were collected from independent users. Therefore, the created dataset contains a total number of 14,364 samples.

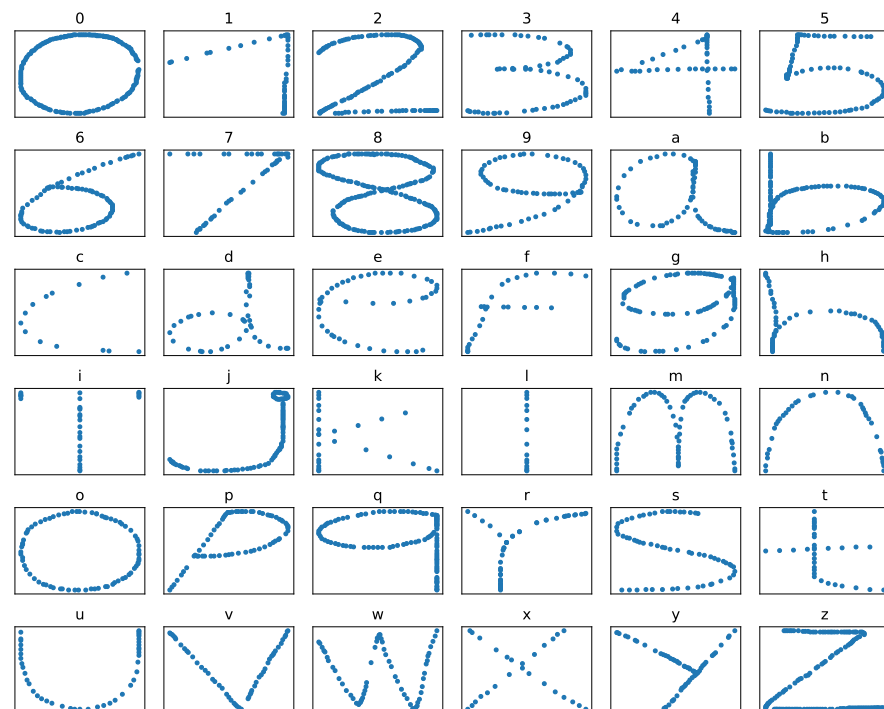


Figure 4. Example of the representative characters written on the touchscreen.

Each handwritten character was written by using a finger or pen on a touchscreen to simulate a proper writing system. The user was free to write using his/her preferred writing style with one or more strokes. The accelerometer sensor in the MYO armband works at a sample rate of 50 Hz. Therefore, the accelerometer data are collected only when the person writes on the touchscreen to segment the accelerometer signals [4,15,16].

4.2. Evaluation and Results

The three proposed neural network architectures were evaluated and extensively compared. For this purpose, the dense layer and the Adam optimization algorithm remained operating without modifications in the stochastic objective functions during the training. The database was divided into ten disjoint subsets using a different number of samples per class. Furthermore, each subset C_i was built with 40, 80, 120, 160, 200, 240, 280, 320, 360, and 399 samples per class. The performance evaluation was conducted using a k-fold cross-validation technique with $k = 10$. Each subset was subdivided to complete the evaluation; 90% of data was used for training, and the remainder, 10%, was used for testing.

Training and test stages were processed on a PC using an Intel® Core™ i7-7700@4.20 GHz, 16 GB RAM, and a GPU NVIDIA GTX 1080 with 8 GB of DDR5 memory, and 2560 CUDA cores.

Figure 5 shows the distribution of classes after applying a dimensionality reduction using the t-Distributed Stochastic Neighbor Embedding (t-SNE) algorithm [37].

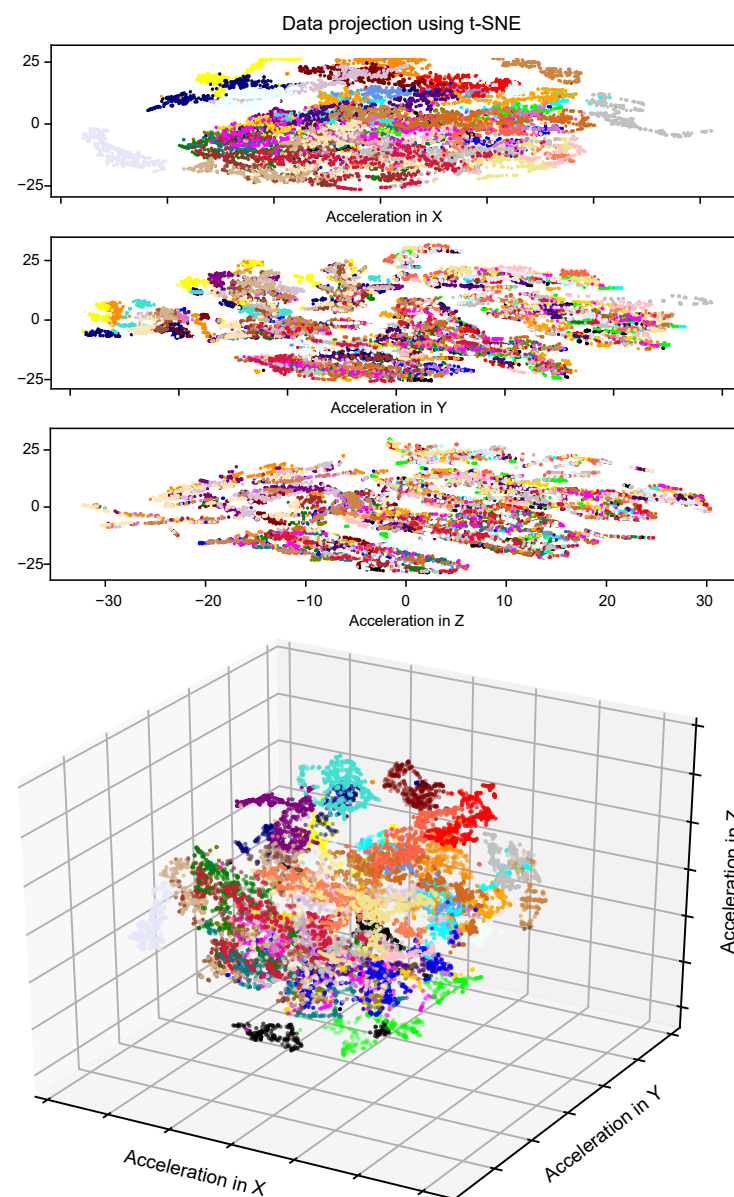


Figure 5. Distribution of the features' space. (Above) For X, Y, and Z axes. (Below) For the three axes.

In order to have a clear idea of how the classes are distributed in the original feature space, the t-SNE method was applied. That method explicitly shows the intrinsic difficulty of classifying each class. The t-SNE method only was used to visualize the data distribution

after reducing the dimension of the original feature space. The feature space dimension was reduced from \mathbb{R}^{116} to \mathbb{R}^2 , and each color represents each of the 36 classes used to classify in Figure 5.

Additionally, the distributions of the feature space obtained from the 3D accelerometer data for all classes in the X, Y, and Z axes are shown in Figure 5. In this paper, the three axes were used in the recognition process. According to Figure 5, clearly separated clusters can be observed for the X axis, whereas for the Y and Z axes, clusters look mostly overlapped. The numerical results are shown in Figure 6. As it can be observed, the three implemented architectures have a comparable performance for more than 120 samples per class.

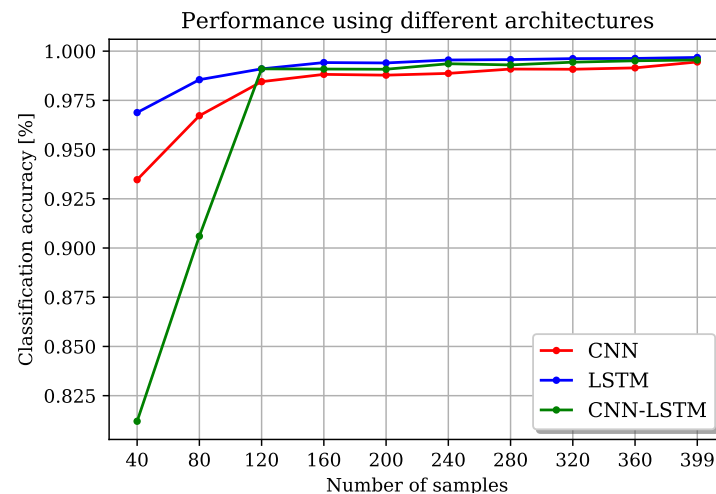


Figure 6. Accuracy of three proposed architectures (CNN, LSTM, and CNN-LSTM) for different numbers of samples per class.

For example, in Figure 6, it was observed that the LSTM architecture presented the best performance using only a few samples. However, when 120 samples are used, the three architectures exhibit almost the same performance. It is important to notice that the three architectures present nearly the same performance for the complete database (i.e., 399 samples per class).

The loss function and the Area Under the Curve (AUC) combined with the Receiver Operating Characteristic (ROC), forming the AUC-ROC parameter, were used to evaluate the performance of the three models in detail. The loss function was used to calculate the model errors during the optimization process, and the ROC curve expresses the ability of a model to classify. The numerical results for the loss function and ROC curve for the proposed CNN, LSTM, and CNN-LSTM architectures are shown in Figure 7, respectively. The numerical results of the AUC-ROC for the three last proposed architectures are 0.99464, 0.99679, and 0.9950.

The loss function used in each proposed architecture was the Cross-Entropy or Log loss. This function measures the performance of a model in classification tasks, where the output is a probability value between 0 and 1. The Cross-Entropy loss function is widely used in Deep Neural Networks [38,39].

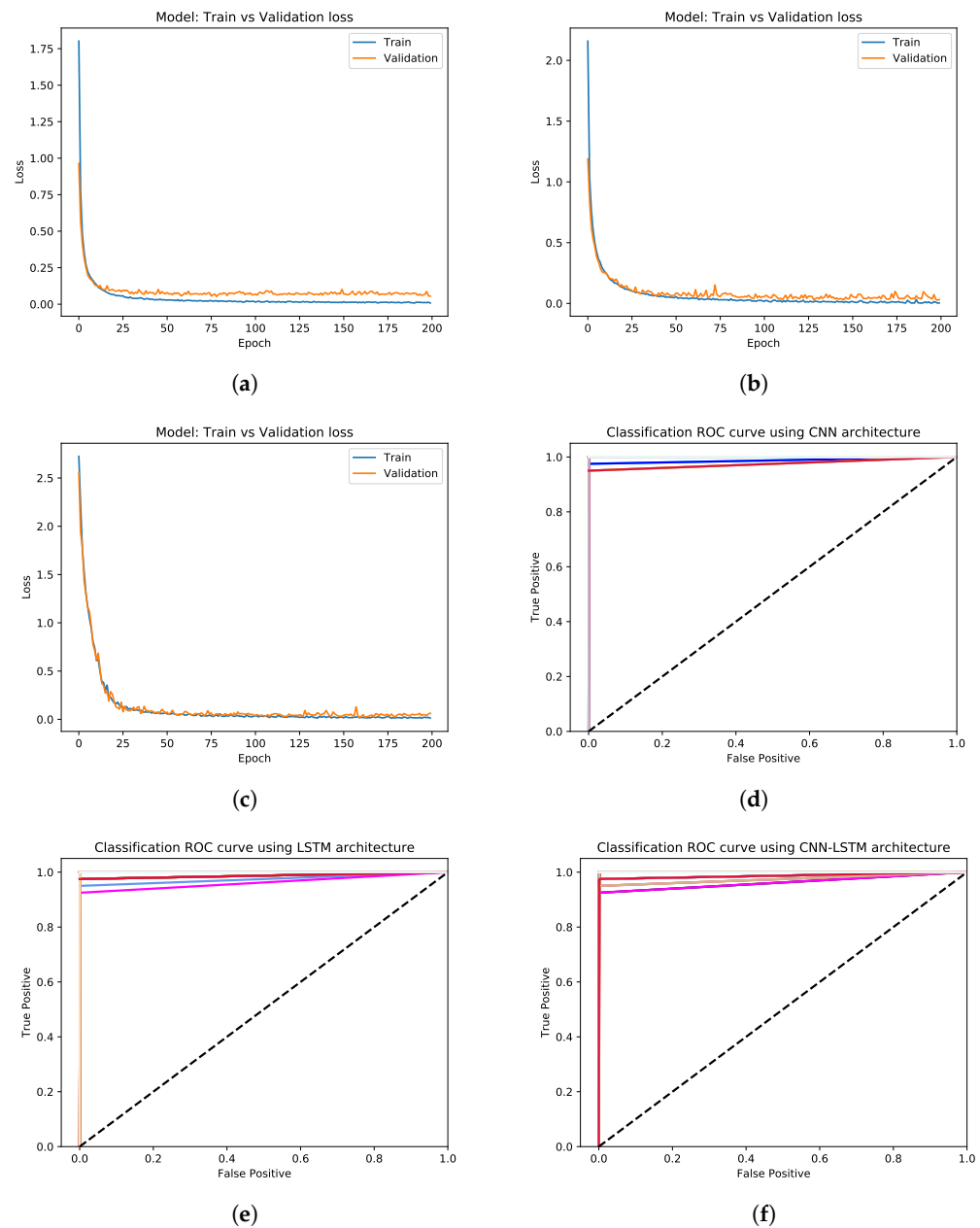


Figure 7. Loss-function and ROC curve for the CNN, LSTM, and CNN-LSTM architectures, the corresponding AUC-ROC metric performances are 0.99464, 0.99679, and 0.99500. (a) Loss function for the CNN architecture. (b) Loss function for the LSTM architecture. (c) Loss function for the CNN-LSTM architecture. (d) ROC curve for the CNN architecture. (e) ROC curve for the LSTM architecture. (f) ROC curve for the CNN-LSTM architecture. In (d–f), the colors indicate different classes and the diagonal dotted line represents the behavior of a random classifier.

4.3. Comparison with the State-of-the-Art

The results of the proposed architectures were compared with the state-of-the-art outcomes. Such comparative results are summarized in Table 5.

Table 5. Comparison with the state-of-the-art related methods.

Method	Accuracy [%]	Precision	F ₁ -Measure
DTW, Kim et al. [4]	95.00	—	—
LR, Xu et al. [25]	94.60	—	—
DTW, Wang and Chuang [14]	98.00	—	—
F-BiLSTM, Li et al. [19]	98.04	—	—
F-BiGRU, Wu et al. [40]	99.15	—	—
FDSVN, Patil et al. [13]	95.21	—	—
CNN *	99.45	0.97	0.97
LSTM *	99.68	0.99	0.99
CNN-LSTM *	99.55	0.98	0.98

(*) Proposed method. (—) Nonavailable.

It is worthwhile to notice that our results were obtained using a proprietary database focused on the handwriting recognition task. Furthermore, the results reported in [4,14,19,25,40] used 26, 26, 10, 12, and 12 different classes, respectively, in contrast to Patil et al. [13] and the results of this study, where 36 different classes were used. In this context, it was noticed that 36 classes were used by Patil et al. [13]. However, they reported only the results for two sets processed separately, obtaining an accuracy of 98.69% and 99.5% for letters (a–z) and digits (0–9), respectively. Table 5 shows the mean of these results as 99.09%.

In this paper, the letters (a–z) and digits (0–9) were processed and classified as a single dataset representing 36 classes. Therefore, the proposed handwriting recognition methodology can recognize a letter or digit from the registered alphanumeric trajectories. The recognition accuracies using CNN, LSTM, and CNN-LSTM architectures were 99.45%, 99.68%, and 99.55%, respectively. The DTW algorithm was also used by Patil et al. [13] and Kim et al. [4], but these last approaches used different methodologies and devices to obtain their corresponding databases. It was observed that the recognition of letters (a–z) in [13] is better than in [4], i.e., 98.69% and 95%, respectively.

4.4. Computational Time Analysis

The total elapsed time during the training and testing was computed for each architecture using a GPU. Table 6 shows a comparison of the time statistics related to training and testing for different state-of-the-art methods and the proposed neural network architecture.

Table 6. Elapsed time expended in training (in min) and testing (in s).

Architecture	Training [min]	Testing [s]
DTW, Kim et al. [4]	—	0.9
LR, Xu et al. [25]	—	—
DTW, Wang and Chuang [14]	16.156	—
F-BiLSTM, Li et al. [19]	—	—
F-BiGRU, Wu et al. [40]	—	—
FDSVN, Patil et al. [13]	—	—
CNN *	5.66	0.0009
LSTM *	155.58	0.0263
CNN-LSTM *	78.48	0.0123

(*) Proposed method. (—) Non-available.

Kim et al. [4] were the only authors to report the result in testing. However, the results are not comparable, because each work used different datasets and computing resources. The CNN architecture was faster during the training stage from the three tested architectures. The CNN needed only 5.68 min to train the architecture using the complete database, while the LSTM required 155.58 min, almost double the time necessary to train

the CNN-LSTM architecture. This discrepancy is because the time needed to recognize a character is very short, requiring a fraction of a second to process the signal representing the character. In this sense, the CNN is the fastest architecture, exhibiting a comparable performance to the proposed LSTM and CNN-LSTM architectures, but the CNN requires more samples to obtain such a performance.

5. Conclusions and Future Works

In this paper, three neural networks were proposed (CNN, LSTM, and the hybrid CNN-LSTM) to solve the online handwritten character recognition problem efficiently.

Our proposals were evaluated using a proprietary dataset constituted of accelerometer data corresponding to multiple-stroke freestyle handwritten characters: 36 classes (26 lowercase letters from the English alphabet and ten Arabic digits). In addition, the dataset was built using a MYO armband to capture the 3D acceleration data of real-time handwriting. The LSTM architecture achieved the best mean accuracy (99.68%). Although the three proposed methodologies have obtained equivalent results, their processing speed is very dissimilar, being CNN the fastest method.

Our dataset is constituted of 399 samples per class obtained by individual users under uncontrolled conditions only supported by the touchscreen laptop to visualize acquired data. The system was only tested on isolated letters and digits. Thus, it is planned to export it to words or phrases to obtain a fluent handwriting recognition system. One limitation of the proposed system is that it depends on a tablet or touchpad for capturing handwritten characters. Another limitation is that it uses the Myo armband, which is not as common as Android devices. Therefore, future work will address other ways of capturing handwritten strokes. In addition, an extension to allow users to write freely in the air would be desirable.

Author Contributions: Conceptualization, P.L.-R.; data curation, J.L.C.-H. and R.C.; formal analysis, J.L.C.-H. and J.G.A.-C.; funding acquisition, J.G.A.-C.; investigation, P.L.-R., J.R.-P., R.C. and J.G.A.-C.; methodology, J.R.-P. and J.G.A.-C.; software, P.L.-R., J.L.C.-H. and J.G.A.-C.; validation, J.R.-P. and J.G.A.-C.; writing—original draft, P.L.-R.; writing—review and editing, J.R.-P., R.C. and J.G.A.-C. All authors read and agreed to the published version of the manuscript.

Funding: This project was funded by the Mexican National Council of Science and Technology CONACyT, under Grant 495754/703539, and the University of Guanajuato, under Grant 171/2022.

Institutional Review Board Statement: Ethical review and approval are waived for this kind of study.

Informed Consent Statement: No formal written consent was required for this study.

Data Availability Statement: Data available under a formal demand.

Acknowledgments: We gratefully thank the Mexican Council of Science and Technology (CONACyT), and the University of Guanajuato for their support.

Conflicts of Interest: The authors declare that they have 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. Kim, J.; Sin, B.K. Online handwriting recognition. In *Handbook of Document Image Processing and Recognition*; Springer: London, UK, 2014; pp. 887–915. [[CrossRef](#)]
2. Zhang, Q.; Wang, D.; Zhao, R.; Yu, Y. MyoSign. In Proceedings of the 24th International Conference on Intelligent User Interfaces—IUI '19, Marina del Ray, CA, USA, 17–20 March 2019; ACM Press: New York, NY, USA, 2019; pp. 650–660. [[CrossRef](#)]
3. Ignatov, A. Real-time human activity recognition from accelerometer data using Convolutional Neural Networks. *Appl. Soft Comput. J.* **2018**, *62*, 915–922. [[CrossRef](#)]
4. Kim, D.W.; Lee, J.; Lim, H.; Seo, J.; Kang, B.Y. Efficient dynamic time warping for 3D handwriting recognition using gyroscope equipped smartphones. *Expert Syst. Appl.* **2014**, *41*, 5180–5189. [[CrossRef](#)]
5. Mannini, A.; Intille, S. Classifier Personalization for Activity Recognition using Wrist Accelerometers. *IEEE J. Biomed. Health Inf.* **2018**, *23*, 1585–1594. [[CrossRef](#)] [[PubMed](#)]

6. Garcia-Ceja, E.; Uddin, M.Z.; Torresen, J. Classification of Recurrence Plots' Distance Matrices with a Convolutional Neural Network for Activity Recognition. *Procedia Comput. Sci.* **2018**, *130*, 157–163. [\[CrossRef\]](#)
7. Dash, A.; Sahu, A.; Shringi, R.; Gamboa, J.; Afzal, M.Z.; Malik, M.I.; Dengel, A.; Ahmed, S. AirScript—Creating Documents in Air. In Proceedings of the 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto, Japan, 9–15 November 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 908–913. [\[CrossRef\]](#)
8. Saha, S.; Saha, N. A Lightning fast approach to classify Bangla Handwritten Characters and Numerals using newly structured Deep Neural Network. *Procedia Comput. Sci.* **2018**, *132*, 1760–1770. [\[CrossRef\]](#)
9. Abdulhussain, S.H.; Mahmmoud, B.M.; Naser, M.A.; Alsabah, M.Q.; Ali, R.; Al-Haddad, S.A.R. A Robust Handwritten Numeral Recognition Using Hybrid Orthogonal Polynomials and Moments. *Sensors* **2021**, *21*, 1999. [\[CrossRef\]](#) [\[PubMed\]](#)
10. Rani, L.; Sahoo, A.K.; Sarangi, P.K.; Yadav, C.S.; Rath, B.P. Feature Extraction and Dimensionality Reduction Models for Printed Numerals Recognition. In Proceedings of the 2022 9th International Conference on Computing for Sustainable Global Development (INDIACom), New Delhi, India, 23–25 March 2022; pp. 798–801. [\[CrossRef\]](#)
11. Lawhern, V.J.; Solon, A.J.; Waytowich, N.R.; Gordon, S.M.; Hung, C.P.; Lance, B.J. EEGNet: A compact convolutional neural network for EEG-based brain–computer interfaces. *J. Neural Eng.* **2018**, *15*, 056013. [\[CrossRef\]](#)
12. Ghosh, D.; Goyal, S.; Kumar, R. Digital pen to convert handwritten trajectory to image for digit recognition. In *Advances in Communication, Devices and Networking*; Bera, R., Sarkar, S.K., Chakraborty, S., Eds.; Springer: Singapore, 2018; pp. 923–932. [\[CrossRef\]](#)
13. Patil, S.; Kim, D.; Park, S.; Chai, Y. Handwriting Recognition in Free Space Using WIMU-Based Hand Motion Analysis. *J. Sens.* **2016**, *2016*, 3692876. [\[CrossRef\]](#)
14. Wang, J.S.; Chuang, F.C. An Accelerometer-Based Digital Pen With a Trajectory Recognition Algorithm for Handwritten Digit and Gesture Recognition. *IEEE Trans. Ind. Electron.* **2012**, *59*, 2998–3007. [\[CrossRef\]](#)
15. Amma, C.; Georgi, M.; Schultz, T. Airwriting: A wearable handwriting recognition system. *Pers. Ubiquitous Comput.* **2014**, *18*, 191–203. [\[CrossRef\]](#)
16. Wijewickrama, R.; Maiti, A.; Jadliwala, M. deWristified. In Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks—WiSec '19, Miami, FL, USA, 15–17 May 2019; ACM Press: New York, NY, USA, 2019; pp. 49–59. [\[CrossRef\]](#)
17. Roy, P.; Ghosh, S.; Pal, U. A CNN Based Framework for Unistroke Numeral Recognition in Air-Writing. In Proceedings of the 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR), Niagara Falls, NY, USA, 5–8 August 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 404–409. [\[CrossRef\]](#)
18. Agrawal, S.; Constandache, I.; Gaonkar, S.; Roy Choudhury, R.; Caves, K.; DeRuyter, F. Using mobile phones to write in air. In Proceedings of the MobiSys '11, the 9th International Conference on Mobile Systems, Applications, and Services, Washington, DC, USA, 28 June–1 July 2011; pp. 15–28. [\[CrossRef\]](#)
19. Li, C.; Xie, C.; Zhang, B.; Chen, C.; Han, J. Deep Fisher discriminant learning for mobile hand gesture recognition. *Pattern Recognit.* **2018**, *77*, 276–288. [\[CrossRef\]](#)
20. Ardüser, L.; Bissig, P.; Brandes, P.; Wattenhofer, R. Recognizing text using motion data from a smartwatch. In Proceedings of the 2016 IEEE International Conference on Pervasive Computing and Communication Workshops, PerCom Workshops, Sydney, Australia, 14–18 March 2016. [\[CrossRef\]](#)
21. Kwon, M.C.; Park, G.; Choi, S. Smartwatch user interface implementation using CNN-based gesture pattern recognition. *Sensors* **2018**, *18*, 2997. [\[CrossRef\]](#) [\[PubMed\]](#)
22. Lin, X.; Chen, Y.; Chang, X.W.; Liu, X.; Wang, X. SHOW: Smart Handwriting on Watches. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* **2017**, *151*, 23. [\[CrossRef\]](#)
23. Xia, Q.; Hong, F.; Feng, Y.; Guo, Z. MotionHacker: Motion sensor based eavesdropping on handwriting via smartwatch. In Proceedings of the INFOCOM 2018—IEEE Conference on Computer Communications Workshops, Honolulu, HI, USA, 15–19 April 2018; pp. 468–473. [\[CrossRef\]](#)
24. Rahagiyanto, A.; Basuki, A.; Sigit, R.; Anwar, A.; Zikky, M. Hand Gesture Classification for Sign Language Using Artificial Neural Network. In Proceedings of the 2017 21st International Computer Science and Engineering Conference (ICSEC), Bangkok, Thailand, 15–18 November 2017; Volume 6, pp. 205–209. [\[CrossRef\]](#)
25. Xu, C.; Pathak, P.H.; Mohapatra, P. Finger-writing with Smartwatch: A Case for Finger and Hand. In Proceedings of the International Workshop on Mobile Computing Systems and Applications, Santa Fe, NM, USA, 12–13 February 2015; pp. 9–14. [\[CrossRef\]](#)
26. Varkey, J.P.; Pompili, D.; Walls, T.A. Erratum to: Human motion recognition using a wireless Sensor-Based wearable system. *Pers. Ubiquitous Comput.* **2012**, *16*, 897–910. [\[CrossRef\]](#)
27. Jalloul, N.; Poree, F.; Viardot, G.; Hostis, P.L.; Carrault, G. Activity Recognition Using Complex Network Analysis. *IEEE J. Biomed. Health Inf.* **2018**, *22*, 989–1000. [\[CrossRef\]](#)
28. Ojagh, S.; Cauteruccio, F.; Terracina, G.; Liang, S.H. Enhanced air quality prediction by edge-based spatiotemporal data preprocessing. *Comput. Electr. Eng.* **2021**, *96*, 107572. [\[CrossRef\]](#)
29. Sa-nguannarm, P.; Elbasani, E.; Kim, B.; Kim, E.H.; Kim, J.D. Experimentation of human activity recognition by using accelerometer data based on LSTM. In *Advanced Multimedia and Ubiquitous Engineering*; Park, J.J., Loia, V., Pan, Y., Sung, Y., Eds.; Springer: Singapore, 2021; pp. 83–89.

30. Livieris, I.E.; Pintelas, E.; Pintelas, P. A CNN–LSTM model for gold price time-series forecasting. *Neural Comput. Appl.* **2020**, *32*, 17351–17360. [[CrossRef](#)]
31. Elmaz, F.; Eyckerman, R.; Casteels, W.; Latré, S.; Hellinckx, P. CNN-LSTM architecture for predictive indoor temperature modeling. *Build. Environ.* **2021**, *206*, 108327. [[CrossRef](#)]
32. Alam, M.S.; Kwon, K.C.; Alam, M.A.; Abbass, M.Y.; Imtiaz, S.M.; Kim, N. Trajectory-Based Air-Writing Recognition Using Deep Neural Network and Depth Sensor. *Sensors* **2020**, *20*, 376. [[CrossRef](#)]
33. Abir, F.A.; Siam, M.A.; Sayeed, A.; Hasan, M.A.M.; Shin, J. Deep Learning Based Air-Writing Recognition with the Choice of Proper Interpolation Technique. *Sensors* **2021**, *21*, 8407. [[CrossRef](#)]
34. LeCun, Y.; Bengio, Y. Convolutional networks for images, speech, and time series. *Handb. Brain Theory Neural Netw.* **1995**, *3361*, 1995.
35. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [[CrossRef](#)]
36. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-Based Learning Applied to Document Recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [[CrossRef](#)]
37. Van der Maaten, L.; Hinton, G. Visualizing data using t-SNE. *J. Mach. Learn. Res.* **2008**, *9*, 2579–2605.
38. Sukhbaatar, S.; Bruna, J.; Paluri, M.; Bourdev, L.; Fergus, R. Training Convolutional Networks with Noisy Labels. In Proceedings of the ICLR 2015, San Diego, CA, USA, 7–9 May 2015; pp. 1–11.
39. Zhang, Z.; Sabuncu, M.R. Generalized cross entropy loss for training deep neural networks with noisy labels. *Adv. Neural Inf. Process. Syst.* **2018**, *31*, 8778–8788.
40. Wu, J.; Pan, G.; Zhang, D.; Qi, G.; Li, S. Gesture recognition with a 3-D accelerometer. In *Lecture Notes in Computer Science; (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 25–38.