Hindawi
Human Behavior and Emerging Technologies
Volume 2022, Article ID 6063779, 13 pages
https://doi.org/10.1155/2022/6063779

Research Article

TARNet: An Efficient and Lightweight Trajectory-Based Air-Writing Recognition Model Using a CNN and LSTM Network

Md. Shahinur Alam,[1] **Ki-Chul Kwon**,[2] **Shariar Md Imtiaz**,[2] **Md Biddut Hossain**,[2] **Bong-Gyun Kang**,[2] **and Nam Kim**[2]

[1]*VL2 Center, Gallaudet University, 800 Florida Avenue NE Washington, D.C. 20002, USA*
[2]*Department of Information and Communication Engineering, Chungbuk National University, Cheongju, Chungbuk 28644, Republic of Korea*

Correspondence should be addressed to Nam Kim; namkim@chungbuk.ac.kr

Received 25 March 2022; Revised 11 July 2022; Accepted 22 July 2022; Published 24 September 2022

Academic Editor: Zheng Yan

Air-writing is a growing research topic in the field of gesture-based writing systems. This research proposes a unified, lightweight, and general-purpose deep learning algorithm for a trajectory-based air-writing recognition network (TARNet). We combine a convolutional neural network (CNN) with a long short-term memory (LSTM) network. The architecture and applications of CNN and LSTM networks differ. LSTM is good for time series prediction yet time-consuming; on the other hand, CNN is superior in feature generation but comparatively faster. In this network, the CNN and LSTM serve as a feature generator and a recognizer, optimizing the time and accuracy, respectively. The TARNet utilizes 1-dimensional separable convolution in the first part to obtain local contextual features from low-level data (trajectories). The second part employs the recurrent algorithm to acquire the dependency of high-level output. Four publicly available air-writing digit (RealSense trajectory digit), character (RealSense trajectory character), smart-band, and Abas datasets were employed to verify the accuracy. Both the normalized and nonnormalized conditions were considered. The use of normalized data required longer training times but provided better accuracy. The test time was the same as those for nonnormalized data. The accuracy for RTD, RTC, smart-band, and Abas datasets were 99.63%, 98.74%, 95.62%, and 99.92%, respectively.

1. Introduction

Touchless writing systems are becoming popular with the advent of augmented reality (AR)/virtual reality (VR), especially when conventional writing is not possible or extremely inconvenient. The emerging technique of AR/VR appears to lead the next generation of interaction. However, the input method of these technologies has not yet developed at its mature level. Speech recognition can be the possible solution; nevertheless, it does not fit in a noisy environment or location where privacy is important [1]. Conventional input method varies across techniques such as mouse or keyboard. However, gesture-based writing can be performed well. It allows users to cooperate with an electronic device without physical interaction; this is especially helpful during the COVID-19 pandemic. In terms of gesture numbers, existing gesture-based writing systems are limited [2–7]. Conventionally, a specific combination of a gesture and a posture symbolizes a number or character, but not all characters are often represented. To overcome this issue, we introduce a novel air-writing technique. It can be counted as complex motion gestures. However, it can significantly express arbitrary character/text input for intelligent control systems.

Air-writing employs a finger or a specific device to write a character or word in a virtual environment. Characters or words are written in an imaginary box (a virtual window) by moving the finger or device [8–14] creating a trajectory representing a specific letter or number in the air. In contrast to gesture recognition, air-writing enables a wide range of interaction options; additionally, the user does not require to memorize them. Alternatively, it can work as a natural solution. It can also be used for wearable devices and

egocentric cameras such as Microsoft HoloLens and Google Glass. As writing is based only on the trajectory, it suffers fewer spatial constraints and lacks haptic feedback, providing an unconstrained writing experience. Air-writing is similar to pen-and-paper-based writing; the only difference is the writing medium (air or paper). However, the bounding box varies among users. Moreover, it is not easy to maintain the same virtual window even for the same user. Users may lose orientation in space due to the absence of visual and haptic feedback. These issues can be resolved by applying specific normalization/transformation techniques to the temporal and trajectory data.

Various three-dimensional (3D) devices track trajectories effortlessly, particularly the Microsoft Kinect, Intel RealSense, and Leap Motion. All have an elaborate application programming interfaces (APIs); the user requires only minimal knowledge of tracking algorithms. Recently, some researchers have been focusing on Wi-Fi [15, 16] and radar [17–19] technology. The Kinect and RealSense are mid-to-long-range devices; the Leap Motion is a short-range device but affords millimetre-level accuracy. Thus, the Kinect and RealSense find most applications on desktops; Leap Motion is preferred for AR/VR [20, 21]. All these devices are excellent in their specific purpose and platforms. Here, we focus on PC-based midrange interaction; hence, Intel RealSense camera is utilized.

Recognition is currently the most challenging task in the field. However, deep learning-based recognition systems afford excellent results without requiring feature generation. Such networks include convolutional neural networks (CNNs) and long short-term memory (LSTM). A CNN is optimal for feature generation, whereas an LSTM is a gold standard for sequential data processing. In this research, we propose a versatile deep learning model that fuses both CNN and LSTM features and achieves outstanding results. The model was trained and tested using the publicly available RealSense trajectory digit (RTD) [22], RealSense trajectory character (RTC) [23], smart-band [24], and Abas datasets [25]. The variety of datasets proves that the proposed model is invariant to a different dataset. The benchmark results verify that the model performs better than a standalone CNN or LSTM network.

The rest of the paper is organized as follows. In Section 2, the background and related works are described. The details of the RTD, RTC, smart-band, and Abas datasets are discussed in Section 3. The proposed TARNet is explained in Section 4. The experimental setup and results are discussed in Section 5. Finally, Section 6 contains the conclusion, summary, and future work.

2. Related Work and the Dataset of Air-Writing

Air-writing is a new form of writing; several novel somatosensory sensors and devices have recently emerged. Some are simple and very cheap ($1) [26]; some are expensive [14, 22, 27–29]. There are some special sensors based on accelerometer [30–32], gyroscope [33], wearable gloves [34, 35], and wrist-worn devices [36–39]. Wobbrock et al. developed a $1 device that can easily handle under 100 lines of code [26]. Most of the accelerometer-based research is conducted with cheap accelerometer sensors equipped with wireless communication [30, 31]. However, the easiest way to access such a sensor is via a mobile phone [32]. Accuracy improves when gyroscopic, motion, and acceleration sensors are combined [33]. The support vector machine (SVM) was first introduced to identify the gestures; later, the HMM was applied to recognize sentences. However, the dataset contains only 720 sentences from 9 participants. More sensors increase the accuracy of trajectory information. However, excessive sensors are burdensome for the user to employ in real-life situations. Also, such devices are bulky and difficult to design. Thus, some off-the-shelf ready-to-go devices have been developed. An earlier version of these devices is H.P. slate [40] and Wiimote [14]. These wearable devices were difficult (and sometimes impossible) to handle. The recent Kinect [27, 29, 41], RealSense [22, 23], and Leap Motion [12, 21, 25, 28, 42, 43] devices are vision-based and very user-friendly and do not require additional devices. The RealSense camera is straightforward to ease.

Earlier air-writing recognition methods relied on handcrafted feature matching or classical machine learning-based algorithms. The SVM [21, 29, 44] and hidden Markov model (HMM) [8, 12, 14, 27, 28, 44] were the most widely used machine learning algorithms requiring predefined feature generation techniques. Kumar et al. proposed SVM for sign language recognition and achieved 63.57%; the accuracy was improved then by applying the BLSTM algorithm [21]. They collected 2240 sign language and 28 finger writing sentences from 10 users. Similarly, Qu et al. proposed a digit recognition system using SVM but added dynamic time warping (DTW) for better accuracy [29]. The collected digit vocabulary was 6104 collected by 10 participants. Amma et al. resolved the dataset issue by collecting 60,000 words from speech data [44]. The highest accuracy was 83%. Fu et al. applied HMM algorithm and achieved more than 86% accuracy [8]. A motion gesture concept was introduced for uninterrupted data processing using Leap Motion and HMM algorithm, which achieved more than 98% recognition accuracy [12, 14]. In most cases, the HMM was more accurate than the SVM. However, neither was completely perfect.

The classical machine learning algorithm suffers from recognition accuracy and is solved by recent state-of-the-art deep learning-based algorithms. CNN [15, 16, 24, 42, 43, 45], LSTM [19, 42, 43, 46–49], and BLSTM [19, 21, 28, 50, 51] algorithms have pioneered air-writing and gesture identification. Yanay and Shmueli employed a CNN algorithm and achieved more than 83% accuracy from 21450 sample data [24]. Some researchers focused on technique rather than the data or participants; hence, accuracy suffers in some cases for the CNN algorithm [15, 16]. In contrast to the accuracy, LSTM performs better for the low volume of the dataset [19, 46]. Since the algorithms are designed and developed for specific purposes and reasons, the standalone module cannot get the full essence of the deep learning features. To overcome this situation, we combined the CNN and LSTM algorithms and TARNet, for better feature generation and detection. Recognition accuracy thus improved.

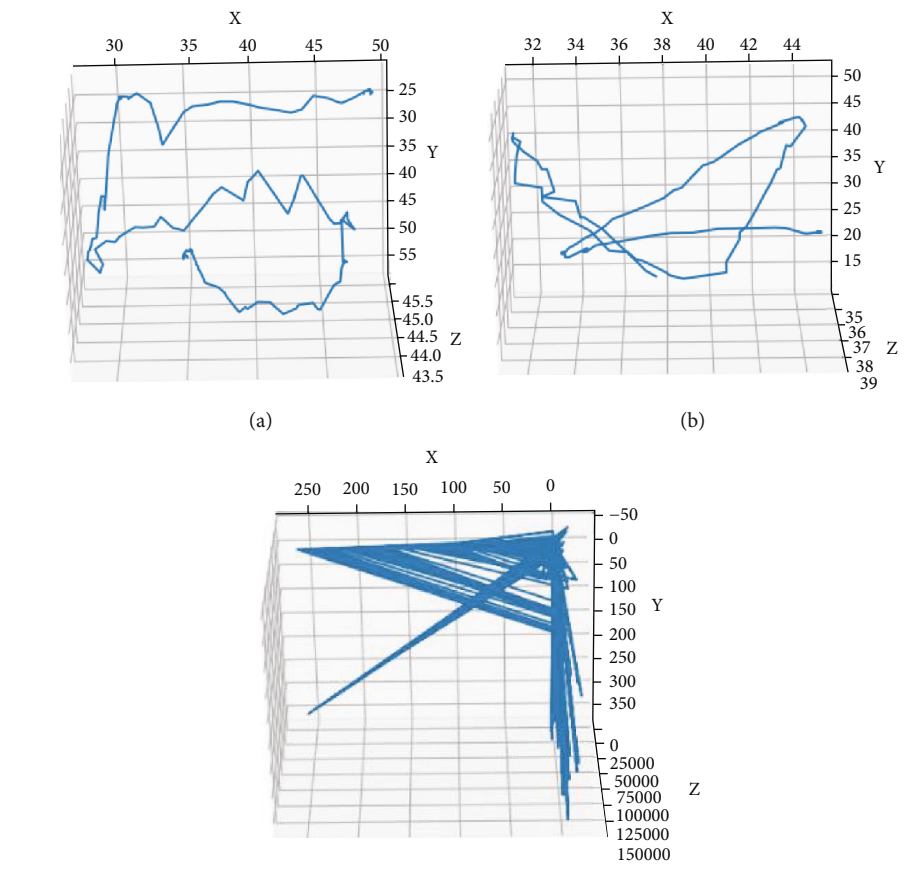

Figure 1: Raw dataset samples. (a) An RTD dataset sample (digit "5"). (b) An RTC dataset sample ("A"). (c) A smart-band dataset sample ("A"). (d) An Abas dataset sample ("A").

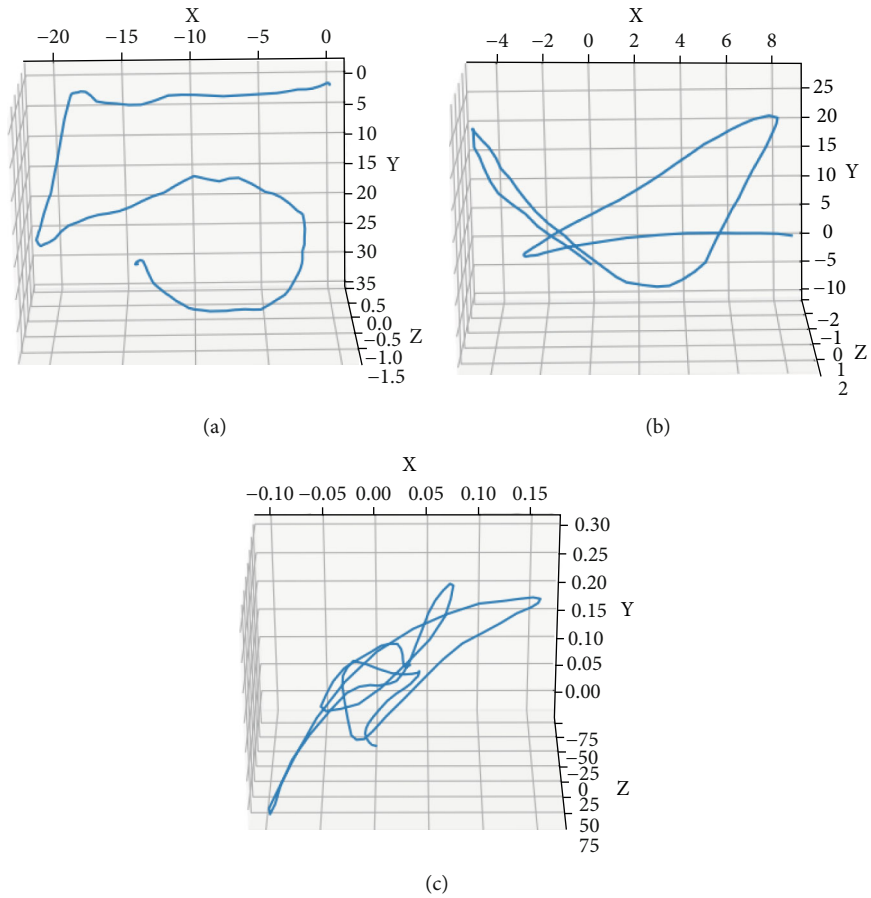

Figure 2: Normalized dataset samples. (a) The RTD dataset for sample digit "5." (b) The RTC dataset for sample character "A." (c) The smart-band dataset for sample character "A." Normalization is not required in case of Abas dataset.

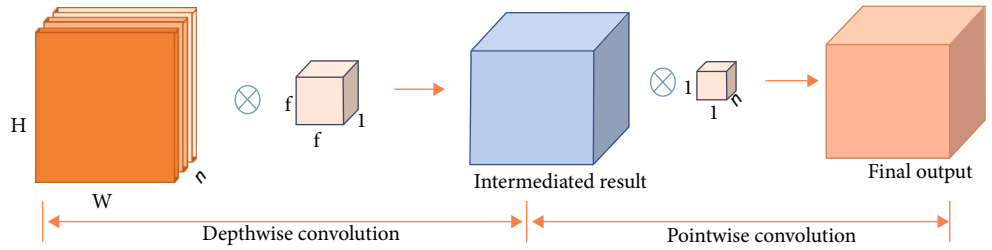

Figure 3: Schematic diagram of the separable convolution layer.

3. The Datasets

We used four air-writing datasets: the RTD [22], RTC [23], smart-band [24], and Abas datasets [25]. The RTD and RTC vision-based datasets were collected using an Intel RealSense sr300 camera. The smart-band is a sensor-based dataset collected using a smartphone and a wrist-worn smart band. On the other hand, the Abas dataset is collected using a Leap Motion device. Although visualization of air-writing characters differs somewhat by how the data are collected, the writing process is the same as the traditional handwriting method.

3.1. The RTD. The RTD dataset is an air-writing digit dataset collected by an Intel RealSense sr300 camera [22]. Graduate students aged between 23 and 30 were employed to collect the dataset. There were 10 participants, and each collected 2100-digit data. They were primarily male (eight). The instructions were similar to those for box-writing [12, 14]. Digits were stored as a sequence of coordinates. The average trajectory length was between 13 and 265. Digits 4 and 1 exhibited the maximum and minimum average lengths, respectively. However, the standard deviation ranged from 15 to 27, indicating that the dataset was consistent. The total RTD dataset size is 21,000, including

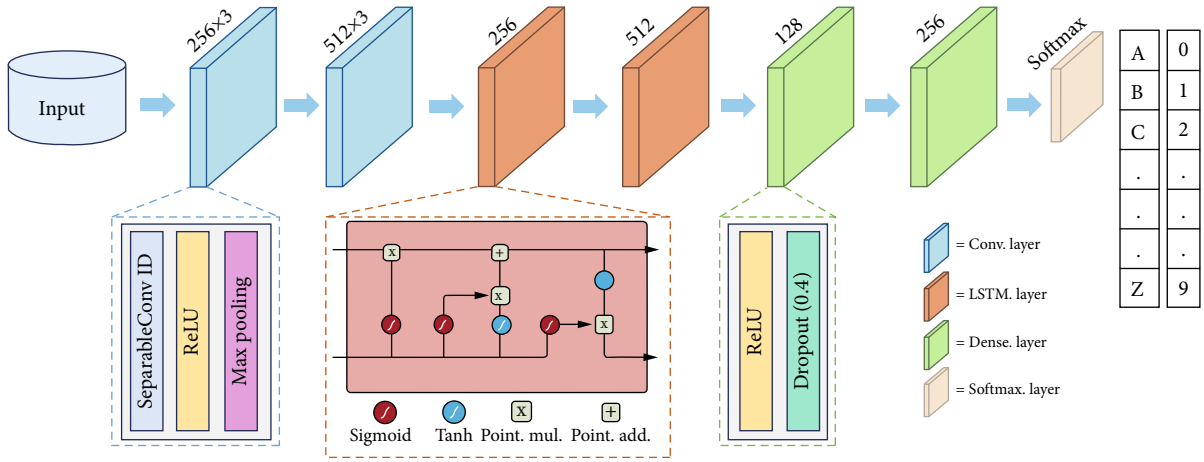

Figure 4: The proposed TARNet architecture.

Table 1: Technical specification of the hardware and the development environment.

Context	Specification
CPU	Intel Core i5-7500 3.40 GHz
Operating system	Windows 10 professional, 64-bit
Memory	16 GB
Environment	Anaconda 64 bit
IDE	Spyder
Programming language	Python
Deep learning library	TensorFlow, Keras

train and test data. RTD raw data sample is shown in Figure 1(a) (sample digit "5").

3.2. RTC. The RTC is an extension of the RTD dataset containing 26 English uppercase letters [23]. Ten graduate students participated; 3,000 trajectory data were collected from each. The sample total is 30,000, including test and training samples. The average trajectory length ranged from 21 to 173. Characters M and C exhibited the maximum and minimum average lengths, respectively. An RTC raw data sample is shown in Figure 1(b) (sample character "A").

3.3. Smart-Band. The smart-band dataset was collected by a "Microsoft band 2" using the Samsung Galaxy S8 smartphone [24]. Accelerometer and gyroscope signals were recorded during air-writing using a dedicated Android app running on a smartphone. Motions were recorded by smart-band sensors. Each sample provided a total of six different dimensions. There were 55 subjects (28 females and 17 males; 46 right-handed) and total 21450 samples. A raw data sample is shown in Figure 1(c) (sample character "A").

3.4. Abas Dataset. Abas dataset contains 22,000 and 8,000 samples for training and testing, respectively [25]. A Leap Motion device was employed to collect the dataset containing a sequence of hand trajectory information. The 3D data are then converted to the tile-based image matrix dimension of 28×28. Hence, each sample data is a 784-size vector data. There were 10 participants aged between 20 and 21 years who participated in this project; each produced around 3,000 handwriting characters throughout a month. The sample data is shown in Figure 1(d). It is already normalized.

4. Proposed Air-Writing Recognition Method

This article proposes a unified general-purpose deep learning algorithm for air-writing recognition. The datasets used are publicly available and were processed to feed the deep learning model. Each dataset was divided into training and test portions. Then, the network was designed using a fusion CNN-LSTM algorithm, trained using the training datasets, and a model generated to test unknown data. Finally, the model was verified by examining the test datasets.

4.1. Dataset Normalization. The raw data collected for air-writing from the sensor or camera is uneven due to its virtual environment (shown in Figure 1); there is no pen-up pen-down system. Hence, normalization is required (except for the Abas dataset). To normalize the dataset, two predominant techniques are used—nearest neighbour and root point normalization. Root point normalization is simple and easy to calculate. Equations (1) to (3) are employed. As there is no virtual window, the trajectories are scattered. Root point normalization transforms all trajectories to the origin:

$$x_i = x_0 - x_i, \tag{1}$$

$$y_i = y_0 - y_i, \tag{2}$$

$$z_i = z_0 - z_i. \tag{3}$$

Here, $[x_0, y_0, z_0]$ and $[x_i, y_i, z_i]$ are the initial and instantaneous points, respectively.

Nearest neighbour point normalization averages the adjacent value from the trajectories. The purpose of the nearest neighbour normalization is to transform the trajectories to a human-understandable format. Equations (4) to

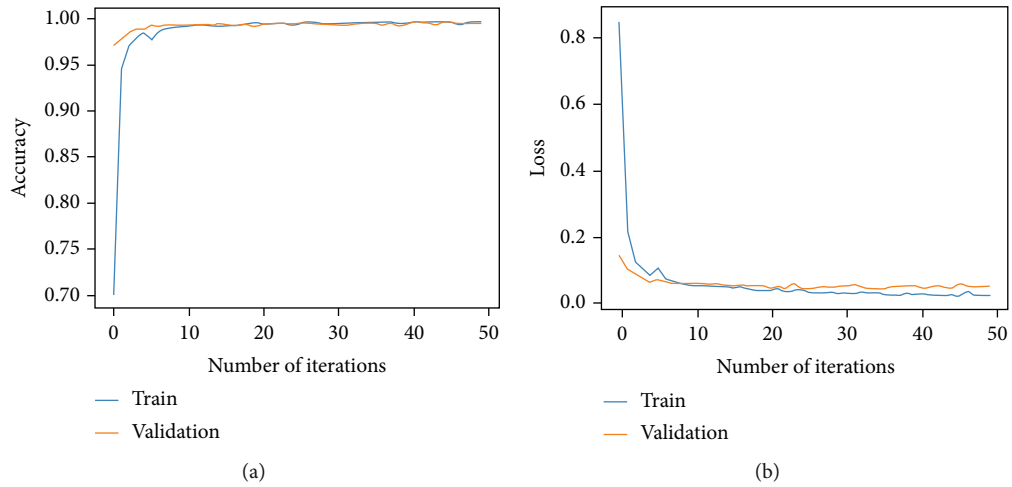

Figure 5: Model accuracy and loss curves for RTD dataset. (a) Accuracy over iterations. (b) Loss over iterations.

(6) are applied for the nearest neighbour normalization.

$$x_i = \frac{1}{n}\sum_{i=0}^{n} x_i, \tag{4}$$

$$y_i = \frac{1}{n}\sum_{i=0}^{n} y_i, \tag{5}$$

$$z_i = \frac{1}{n}\sum_{i=0}^{n} z_i. \tag{6}$$

Here i and n refer to any individual point and the number of points considered over a trajectory, respectively. Consider more points shrink the original shape; hence, it is found that five or six numbers were optimal. In this experiment, six points are considered.

The normalized datasets are shown in Figure 2, which is visually far clear than Figure 1. Especially, dramatic changes were found for the smart-band dataset. In Figure 2, all trajectories are started from the origin that comes from the root point normalization. Besides, the smoothness comes from the nearest neighbour point normalization.

4.2. Separable Convolution Layer. In this work, a fusion of CNN and the LSTM network is utilized. Due to the wide use of CNN, different variants of CNNs are proposed. It is mainly used for image processing and feature detection [52]. A conventional CNN network is very expensive in terms of resources and time, requiring massive calculations. However, the recent depthwise separable convolution processes allow efficient calculation and processing of one-dimensional information and sequential data [53]. Separable convolution features two steps; the first is depthwise convolution and the second 1×1 pointwise convolution to convolve the channels.

Figure 3 shows a schematic diagram of the depthwise separable convolution. A depthwise separable convolution follows the same feature as spatially separable convolutions; splitting the kernels into two smaller ones produces the same result with lesser multiplications. The total multiplications in the depthwise and pointwise convolution are $H \times W \times n \times f \times f$ and $H \times W \times n$, respectively. The principal advantage of pointwise convolution is that the filter is the only multiplicand lacking an extra dimension; hence, the calculation burden falls dramatically.

4.3. The LSTM Network. A recurrent neural network (RNN) is widely used for time series predictions (activities, audio, and text) [53]. An RNN is a generalization of a feedforward neural network that uses memory to store local information. Unlike feedforward networks, RNN uses internal memory (known as the state) to process sequences of inputs. An LSTM is a modified version of RNN that resolves the vanishing gradient problem. It features input, forget, and output gates, of which the most important are the forget and output gates that define the data that should be removed and passed to the output, respectively.

4.4. The TARNet Architecture. The proposed TARNet exploits the advantages of both the CNN and LSTM networks. A CNN and an LSTM are used for feature extraction and sequence recognition, respectively. The complete network diagram is shown in Figure 4. The network features an input, convolution, pooling, an LSTM unit, and dense layers. The inputs are one-dimensional sequential data. The input layer dimensions are set to the maximum lengths of the trajectory sequences, which are 300, 800, 850, and 785 for the RTD, RTC, smart-band, and Abas data, respectively. The first convolution layer features a filter of size 256, followed by a max-pooling layer that downsamples the CNN network. This reduces computational complexity and cost. A filter of size 512 is used for the following convolution layer (which again includes a max-pooling layer). The principal purpose of CNN is feature generation.

The LSTM features two layers, and sizes are 256 and 512 for the first and second layers, respectively. Similar to the previous convolution layers, max-pooling is used in the first LSTM layer. An LSTM layer consisted of sigmoid, tanh, pointwise addition, and multiplications. Flattening is

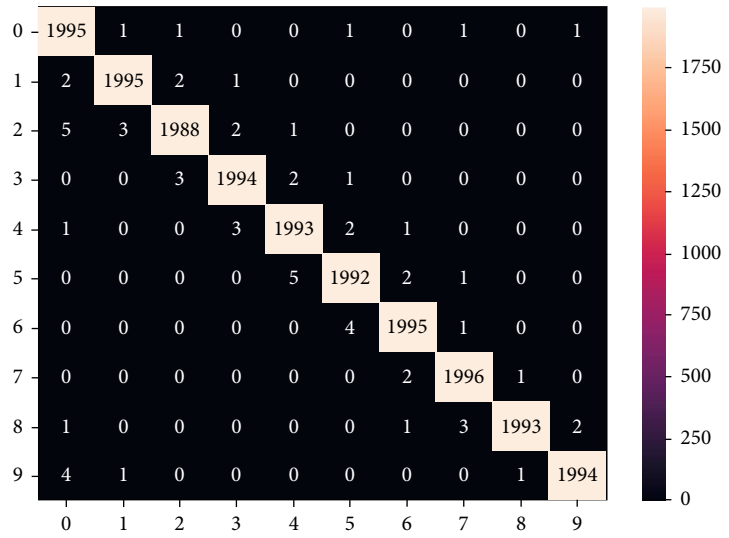

Figure 6: Confusion matrix for the RTD dataset.

Table 2: Character-wise performance comparison between different datasets and algorithms. The accuracies are in %.

Character	RTC [23]		Smart-band [24]	Abas [25]				
	CNN [22]	LSTM [22]	TARNet	TARNet	MLP-Bp	MLP-Rprop	DBN-Bp	DBN-Rprop	TARNet
A	99.87	100	99.10	97.53	99.35	100	100	100	99.88
B	99.61	99.35	99.87	98.27	9286	99.68	99.03	100	100
C	99.74	99.87	99.87	98.27	94.81	100	99.35	100	99.88
D	100	100	99.86	77.77	92.88	100	98.71	100	99.88
E	99.74	99.74	99.61	98.02	92.51	100	99.35	100	99.76
F	98.71	99.22	97.94	97.53	97.73	99.68	99.03	99.35	99.76
G	99.48	99.74	99.87	97.77	77.67	100	98.38	99.68	99.88
H	99.87	99.87	99.87	97.16	90.29	99.68	99.68	99.68	99.88
I	99.60	99.73	100	98.14	90.58	97.40	95.45	98.38	100
J	100	100	99.91	96.04	96.74	100	100	100	99.76
K	99.87	99.87	99.87	95.67	94.48	99.03	98.70	100	99.64
L	100	100	100	97.40	99.68	100	100	100	99.88
M	99.73	99.47	98.86	98.51	91.53	99.67	99.02	99.67	100
N	99.48	99.48	99.74	98.02	98.38	99.35	99.35	99.35	99.40
O	99.74	99.87	99.87	97.16	95.45	99.03	98.38	99.35	99.52
P	99.35	99.35	99.48	94.07	95.44	99.67	99.67	100	99.64
Q	99.87	99.74	99.87	98.02	91.53	99.67	99.35	99.67	99.88
R	99.48	99.74	99.74	79.90	91.86	99.67	95.44	99.67	100
S	99.74	99.87	99.74	97.16	93.16	99.35	99.35	100	99.76
T	99.61	99.87	100	97.77	93.18	100	99.03	100	99.88
U	99.22	99.87	99.61	98.27	94.16	100	100	100	99.88
V	99.34	99.21	99.73	97.90	99.03	100	100	100	100
W	99.87	98.06	99.87	98.27	95.44	99.35	100	100	99.88
X	99.87	99.87	99.87	98.02	95.11	99.35	100	100	99.76
Y	100	100	100	95.06	98.05	99.35	99.67	99.67	100
Z	100	100	99.92	98.76	77.20	99.35	98.37	98.05	100

performed before feeding the dense layer. The first and second dense layer features 128 and 256 neurons, respectively. The dense layer is fully connected and associated with the ReLU activation function (7). It helps the network by removing the negative part and keeping only the positive value from the neuron. A dropout rate of 0.4 is used in both dense

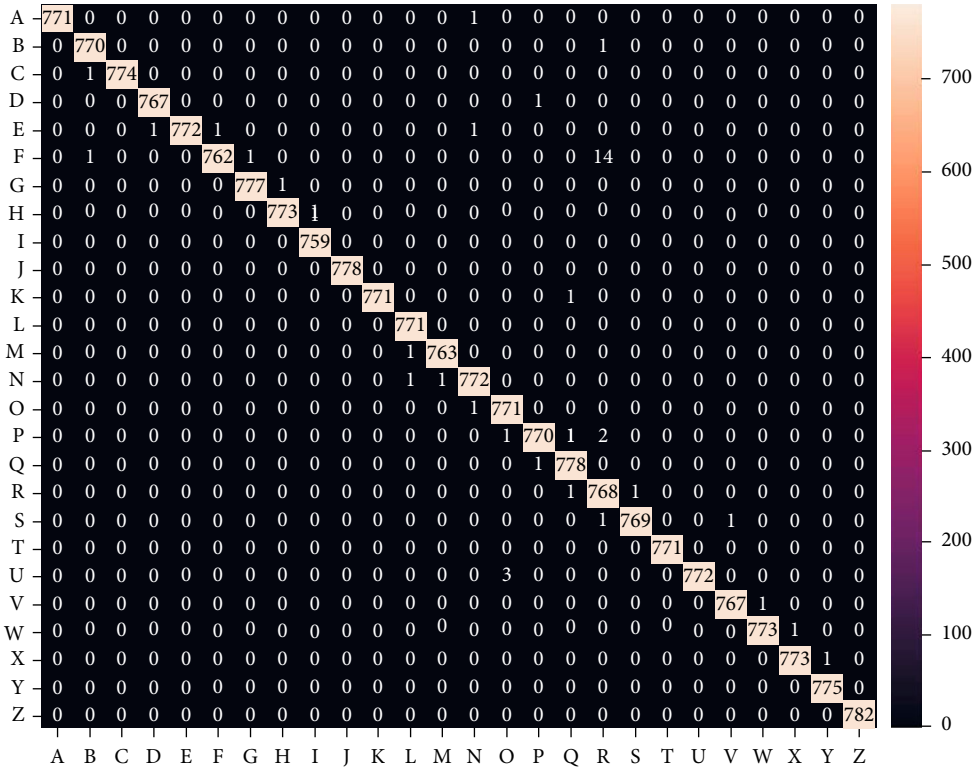

Figure 7: Confusion matrix for the RTC dataset for the proposed CNN-LSTM network.

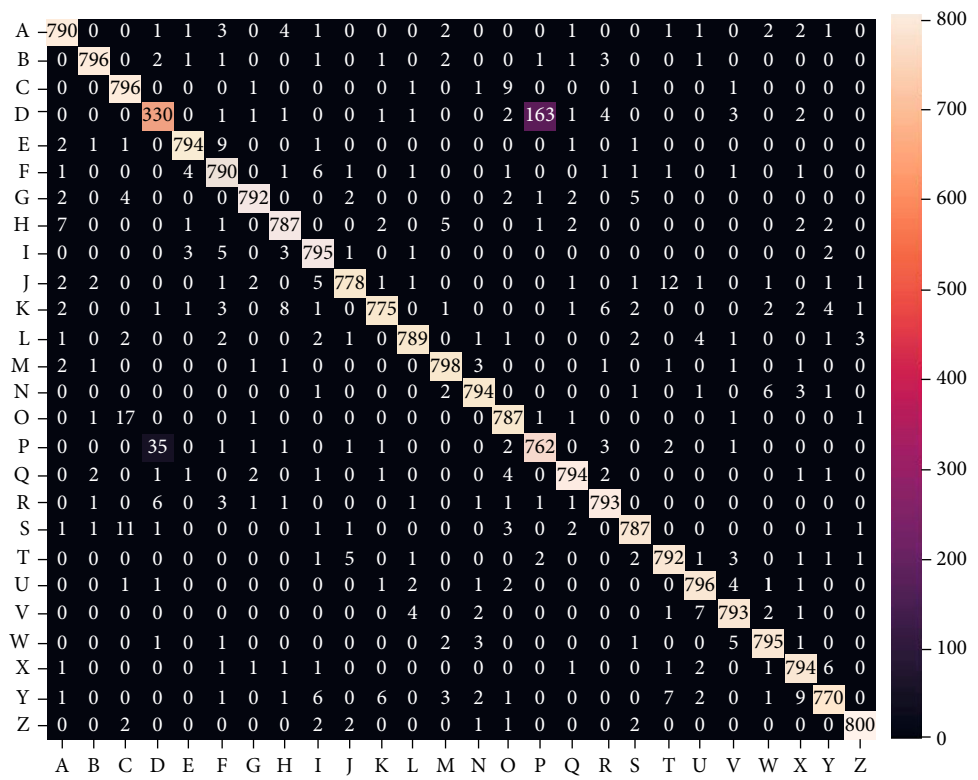

Figure 8: Confusion matrix for the smart-band dataset for the proposed CNN-LSTM network.

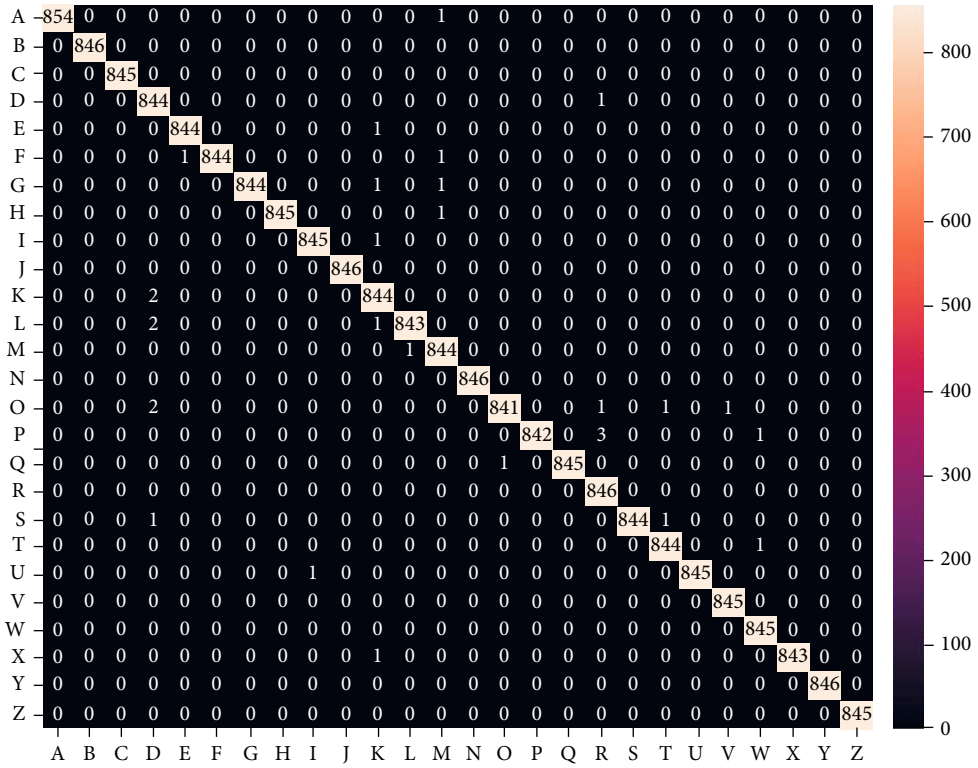

Figure 9: Confusion matrix for the Abas dataset for the proposed CNN-LSTM network.

layers to prevent overfitting. Adam optimizer is used in this network with a learning rate of 0.0005. The output layer is variable here; the output size is 10, 26, 26, and 26 for RTD, RTC, smart-band, and Abas datasets, respectively. To generalize the results, the softmax activation function is employed in the output layer (8). Softmax is a probabilistic method to regularize an output.

$$f(x) = \max(0, x), \tag{7}$$

$$\sigma(Z_i) = \frac{e^{Z_i}}{\sum_{k=1}^{K} e^{Z_k}}. \tag{8}$$

Here, σ represents the softmax function. An input vector Z is assigned to all classes by the multiclass classifier k.

5. Experimental Setup and Performance Analysis

The experimental setup is straightforward and does not require any high-end devices. The specifications are listed in Table 1. An Intel Core i5 processor (clock speed 3.40 GHz), Windows 10, and a 64-bit operating system were employed to train the network. The main memory was 16 GB. The popular deep learning-based library Keras ran on top of TensorFlow. We employed the Spyder integrated development environment (IDE) of the Anaconda system to run the network. All code was written in Python. The code is available on GitHub: https://github.com/shahinur-alam/TARNet

The model was trained using RTD, RTC, smart-band, and Abas datasets containing data on 20,000, 30,000, 21,450, and 30,000 trajectories, respectively. During training, we employed 10-fold cross-validation. Different batch sizes and iterations are performed. However, a batch size of 256 is optimal for the RTD, RTC, smart-band, and Abas datasets. The training accuracies and losses are shown in Figure 5 (number of iterations: 50). However, optimal accuracy was achieved within 15 iterations for all datasets. Figure 5(a) shows the training and validation accuracies by iteration number. The validation accuracy after the first step was almost 0.9. Figure 5(b) shows the losses over the iterations. As for accuracy, the expected outcome was found within 15 iterations. Note that the training and validation accuracies follow the same curve; this indicates network stability, fidelity, and minimal overfitting. Both accuracy and loss verify that the model is a good fit for both RTD, RTC, smart-band, and Abas datasets.

The confusion matrices for the RTD dataset are shown in Figure 6. For the RTD dataset, 0 and 9 were detected with the maximum accuracy and 5 and 2 with the poorest accuracy because they are challenging to write, ambiguous, and similar in shape.

Table 2 and Figure 7 show that the RTC dataset exhibited uniformly good performance. The best results were found for the character I, J, L, T, Y, and Z. All are distinguishable and unique characters. The worst result was that for F. There were 14 false positives for R. This is because of the similar writing patterns. During air-writing, the start and endpoints are the same for F and R. Similarly, there were three false-positive results for O. However, these results are reasonable. All other characters were reliably detected.

Table 3: Performance comparisons.

Dataset	Normalized	Algorithm	Accuracy
RTD [22]	Yes	CNN [22]	99.06
		LSTM [22]	99.17
		CNN [43]	98.5
		TARNet	99.63
	No	CNN [22]	98.26
		LSTM [22]	98.68
		CNN [43]	98.2
		TARNet	99.4
RTC [23]	Yes	CNN [22]	97.73
		LSTM [22]	97.12
		CNN [43]	97.3
		TARNet	98.74
	No	CNN [22]	97.71
		LSTM [22]	96.98
		CNN [43]	97.64
		TARNet	98.4
Smart-band [24]	Yes	CNN [22]	75.97
		LSTM [22]	67.11
		CNN [43]	73.17
		CNN (KNN-DTW)	83.2
		TARNet	95.62
	No	CNN [22]	75.83
		LSTM [22]	67.11
		CNN [43]	75.45
		TARNet	80.72
Abas [25]	Normalization is not necessary	MLP-Bp	93.43
		MLP-Rprop	99.59
		DBN-Bp	99.05
		DBN-Rprop	99.71
		CNN [43]	79.15
		TARNet	99.92

The false-positive rate was higher for the smart-band dataset shown in Table 2 and Figure 8. However, usually, accuracy was satisfactory. The highest false positives were P (163), D (35), and T (12) for D, J, and K, respectively. Except for character D, other results are satisfactory. A more complex algorithm and a better normalization technique might solve the problem.

In the case of the Abas dataset, the best result was found for the character B, M, V, Y, and Z; the details are shown in Table 2 and Figure 9. Some character shows better result for DBN-Rprop than the proposed method. However, the overall result is better than theirs (shown in Table 3).

The character-wise performance of the TARNet is shown in Table 2. We compared the method with previously implemented CNN and LSTM networks [22]. Different algorithms and normalization conditions were considered; the TARNet afforded superior accuracy for all normalized RTD, RTC, smart-band, and Abas data.

The overall accuracy of the RTD dataset for normalized and nonnormalized for CNN [22], LSTM [22], CNN [43], and proposed TARNet is 99.06, 99.17, 98.5, and 99.63 and 98.26, 96.68, 98.2, and 99.4, respectively. Similarly, the accuracy for the RTC normalized and nonnormalized is 97.73, 97.12, 97.3, and 98.4 and 97.71, 96.98, 97.64, and 98.74, respectively. The accuracy is almost similar for CNN and TARNet algorithms for RTC normalized data. However, the accuracy significantly varies for nonnormalized data. The CNN [22], LSTM [22], and CNN [43] accuracies for the smart-band dataset were much lower, perhaps because the data were noisy (Figure 1). The actual result is better than CNN [22], LSTM [22], and CNN [43] networks, because the dataset was modelled with the KNN-DTW algorithm [24]. However, the TARNet algorithm still performed well (95.62%). The normalized and nonnormalized data accuracies were 95.62 and 80.72%, respectively. We found that Abas Leap Motion dataset is very organized. Hence, an excellent result is found. The accuracy of DBN-Rprop

was 99.71%; on the other hand, the proposed algorithm achieved 99.92%. Abas dataset is a little different than the trajectory dataset. Abas et al. used a special algorithm during preprocessing to convert 3D trajectory data into a 2D binary image so that it can be compared with the MNIST dataset (Figure 1(d)). That is why normalization is not necessary and has no impact on it.

6. Conclusion

In this work, a combination of CNN and LSTM named TAR-Net is developed where the CNN and the LSTM are simultaneously used for feature generation and recognition, respectively. As a result, the network is time-efficient and exhibits high performance. Accuracy was verified using the publicly available RTD, RTC, smart-band, and Abas datasets. The benchmark result outperforms those of existing standalone CNN and LSTM models. The accuracies were 99.63, 98.74, 95.62%, and 99.92% for the RTD, RTC, smart-band, and Abas datasets. The overall accuracy is remarkable; however, some digits and characters exhibited higher error rates because of differences in writing styles. In the future, we will resolve these issues. Also, more datasets will be collected and tested.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported in part by the National Research Foundation of Korea (NRF) under Grant NRF-2020R1A2C1101258 and 2018R1D1A3B07044041. Special thanks are due to Abas et al. for providing their Leap Motion dataset.

References

[1] Y. Tong, F. Wang, and W. Wang, "Fairies in the box: children's perception and interaction towards voice assistants," *Human Behavior and Emerging Technologies*, vol. 2022, pp. 1–8, 2022.

[2] M. S. Alam, K. C. Kwon, and N. Kim, "Implementation of a character recognition system based on finger-joint tracking using a depth camera," *IEEE Transactions on Human-Machine Systems*, vol. 51, no. 3, pp. 229–241, 2021.

[3] R. Ma, Z. Zhang, and E. Chen, "Human motion gesture recognition based on computer vision," *Complexity*, vol. 2021, 11 pages, 2021.

[4] Z. H. Chen, J. T. Kim, J. Liang, J. Zhang, and Y. B. Yuan, "Real-time hand gesture recognition using finger segmentation," *Scientific World Journal*, vol. 2014, pp. 1–9, 2014.

[5] Y. Luo, G. Cui, and D. Li, "An improved gesture segmentation method for gesture recognition based on CNN and YCbCr," *Journal of Electrical and Computer Engineering*, vol. 2021, 9 pages, 2021.

[6] R. F. Pinto, C. D. B. Borges, A. M. A. Almeida, and I. C. Paula, "Static hand gesture recognition based on convolutional neural networks," *Journal of Electrical and Computer Engineering*, vol. 2019, 12 pages, 2019.

[7] H. Zhang, L. Chen, Y. Zhang et al., "A wearable real-time character recognition system based on edge computing-enabled deep learning for air-writing," *Journal of Sensors*, vol. 2022, 12 pages, 2022.

[8] Z. Fu, J. Xu, Z. Zhu, A. X. Liu, and X. Sun, "Writing in the air with WiFi signals for virtual reality devices," *IEEE Transactions on Mobile Computing*, vol. 18, no. 2, pp. 473–484, 2019.

[9] M. A. Rahim, J. Shin, and M. R. Islam, "Gestural flick input-based non-touch interface for character input," *The Visual Computer*, vol. 36, no. 8, pp. 1559–1572, 2020.

[10] J. Gan, W. Wang, and K. Lu, "In-air handwritten Chinese text recognition with temporal convolutional recurrent network," *Pattern Recognition*, vol. 97, p. 107025, 2020.

[11] D. Moazen, S. A. Sajjadi, and A. Nahapetian, "AirDraw: leveraging smart watch motion sensors for mobile human computer interactions," in *2016 13th IEEE Annual Consumer Communications and Networking Conference*, pp. 442–446, Las Vegas, NV, USA, 2016.

[12] M. Chen, G. AlRegib, and B. H. Juang, "Air-writing recognition - part II: detection and recognition of writing activity in continuous stream of motion data," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 3, pp. 436–444, 2016.

[13] J. Gan and W. Wang, "In-air handwritten English word recognition using attention recurrent translator," *Neural Computing and Applications*, vol. 31, no. 7, pp. 3155–3172, 2017.

[14] M. Chen, G. AlRegib, and B.-H. Juang, "Air-writing recognition—part I: modeling and recognition of characters, words, and connecting motions," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 3, pp. 403–413, 2016.

[15] F. Khan, S. K. Leem, and S. H. Cho, "In-air continuous writing using UWB impulse radar sensors," *IEEE Access*, vol. 8, pp. 99302–99311, 2020.

[16] Y. Fang, Y. Xu, H. Li, X. He, and L. Kang, "Writing in the air: recognize letters using deep learning through WiFi signals," in *Proceedings -2020 6th International Conference on Big Data Computing and Communications*, pp. 8–14, Deqing, China, 2020.

[17] D. Teng, Y. Xiong, L. Liu, and B. Wang, "Multiview three-dimensional display with continuous motion parallax through planar aligned OLED microdisplays," *Optics Express*, vol. 23, no. 5, pp. 6007–6019, 2015.

[18] M. Arsalan, A. Santra, and V. Issakov, "Radar trajectory-based air-writing recognition using temporal convolutional network," in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 1454–1459, Miami, FL, USA, 2020.

[19] M. Arsalan and A. Santra, "Character recognition in air-writing based on network of radars for human-machine interface," *IEEE Sensors Journal*, vol. 19, no. 19, 2019.

[20] S. K. Behera, D. P. Dogra, and P. P. Roy, "Fast recognition and verification of 3D air signatures using convex hulls," *Expert Systems with Applications*, vol. 100, pp. 106–119, 2018.

[21] P. Kumar, R. Saini, S. K. Behera, D. P. Dogra, and P. P. Roy, "Real-time recognition of sign language gestures and air-writing using leap motion," in *2017 Fifteenth IAPR*

International Conference on Machine Vision Applications (MVA), pp. 157–160, Nagoya, Japan, 2017.

[22] M. S. Alam, K.-C. Kwon, M. A. Alam, M. Y. Abbass, S. M. Imtiaz, and N. Kim, "Trajectory-based air-writing recognition using deep neural network and depth sensor," *Sensors*, vol. 20, no. 2, p. 376, 2020.

[23] M. S. Alam, K. C. Kwon, and N. Kim, "Trajectory-based air-writing character recognition using convolutional neural network," in *Proceedings -2019 4th International Conference on Control, Robotics and Cybernetics*, pp. 86–90, Tokyo, Japan, 2019.

[24] T. Yanay and E. Shmueli, "Air-writing recognition using smart-bands," *Pervasive and Mobile Computing*, vol. 66, article 101183, 2020.

[25] A. Setiawan and R. Pulungan, "Deep belief networks for recognizing handwriting captured by leap motion controller," *International Journal of Electrical and Computer Engineering*, vol. 8, no. 6, pp. 4693–4704, 2018.

[26] J. O. Wobbrock, A. D. Wilson, and Y. Li, "Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes," in *Proceedings of the 20th annual ACM symposium on User interface software and technology - UIST '07*, Newport Rhode Island, USA, 2007.

[27] S. Mohammadi and R. Maleki, "Air-writing recognition system for Persian numbers with a novel classifier," *The Visual Computer*, vol. 36, no. 5, pp. 1001–1015, 2020.

[28] P. Kumar, R. Saini, P. P. Roy, and D. P. Dogra, "Study of text segmentation and recognition using leap motion sensor," *IEEE Sensors Journal.*, vol. 17, no. 5, pp. 1293–1301, 2017.

[29] C. Qu, D. Zhang, and J. Tian, "Online kinect handwritten digit recognition based on dynamic time warping and support vector machine," *Journal of Information and Computational Science*, vol. 12, no. 1, pp. 413–422, 2015.

[30] J. S. Wang and F. C. Chuang, "An accelerometer-based digital pen with a trajectory recognition algorithm for handwritten digit and gesture recognition," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 7, pp. 2998–3007, 2012.

[31] S. Kratz and M. Rohs, "Protractor3D," in *Proceedings of the 15th international conference on Intelligent user interfaces - IUI '11*, New York, USA, 2011.

[32] J. Guerra-Casanova, C. S. Avila, G. Bailador, and A. De-Santos-Sierra, "Time series distances measures to analyze in-air signatures to authenticate users on mobile phones," in *2011 Carnahan Conference on Security Technology*, pp. 1–7, Barcelona, Spain, 2011.

[33] Y. Luo, C. C. Tsang, G. Zhang et al., "An attitude compensation technique for a MEMS motion sensor based digital writing instrument," in *Proceedings of 1st IEEE International Conference on Nano Micro Engineered and Molecular Systems*, pp. 909–914, Zhuhai, China, 2006.

[34] H. Kenn, F. van Megen, and R. Sugar, "A glove-based gesture interface for wearable computing applications," in *IFAWC 2007-4th International Forum on Applied Wearable Computing 2007, Proceedings*, pp. 169–178, 2007.

[35] J. Kim, J. He, K. Lyons, and T. Starner, "The gesture watch: a wireless contact-free gesture based wrist interface," in *Proceedings-International Symposium on Wearable Computers, ISWC*, pp. 15–22, 2007.

[36] C. Xu, P. H. Pathak, and P. Mohapatra, "Finger-writing with smartwatch: a case for finger and hand gesture recognition using smartwatch," in *HotMobile 2015-16th International Workshop on Mobile Computing Systems and Applications*, pp. 9–14, Santa Fe New Mexico, USA, 2015.

[37] L. Ardüser, P. Bissig, P. Brandes, and R. Wattenhofer, "Recognizing text using motion data from a smartwatch," in *2016 IEEE International Conference on Pervasive Computing and Communication Workshops, PerCom Workshops 2016*, Sydney, NSW, Australia, 2016.

[38] N. B. LevyAlona, E. Yuval, and S. Erez, "Handwritten signature verification using wrist-worn devices," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 2, no. 3, pp. 1–26, 2018.

[39] H. Wen, J. R. Rojas, and A. K. Dey, "Serendipity: finger gesture recognition using an off-the-shelf smartwatch," in *Conference on Human Factors in Computing Systems-Proceedings*, pp. 3847–3851, San Jose, California, USA, 2016.

[40] H. Nguyen and M. C. Bartha, "Shape writing on tablets: better performance or better experience?," *Proceedings of the Human Factors and Ergonomics Society*, vol. 56, no. 1, pp. 1591–1593, 2012.

[41] S. Poularakis and I. Katsavounidis, "Low-complexity hand gesture recognition system for continuous streams of digits and letters," *IEEE Transactions on Cybernetics*, vol. 46, no. 9, pp. 2094–2108, 2016.

[42] A. Ikram and Y. Liu, "Skeleton based dynamic hand gesture recognition using LSTM and CNN," in *ACM International Conference Proceeding Series*, pp. 63–68, Bangkok Thailand, 2020.

[43] G. Bastas, K. Kritsis, and V. Katsouros, "Air-writing recognition using deep convolutional and recurrent neural network architectures," in *Proceedings of International Conference on Frontiers in Handwriting Recognition, ICFHR*, pp. 7–12, Dortmund, Germany, 2020.

[44] C. Amma, M. Georgi, and T. Schultz, "Airwriting: a wearable handwriting recognition system," *Personal and Ubiquitous Computing.*, vol. 18, no. 1, pp. 191–203, 2014.

[45] P. Roy, S. Ghosh, and U. Pal, "A CNN based framework for unistroke numeral recognition in air-writing," in *Proceedings of International Conference on Frontiers in Handwriting Recognition, ICFHR*, Niagara Falls, NY, USA, 2018.

[46] S. Xu and Y. Xue, "A long term memory recognition framework on multi-complexity motion gestures," in *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pp. 201–205, Kyoto, Japan, 2018.

[47] P. Wang, J. Lin, F. Wang et al., "A gesture air-writing tracking method that uses 24 GHz SIMO radar SoC," *IEEE Access*, vol. 8, 2020.

[48] D. Castells-Rufas, J. Borrego-Carazo, J. Carrabina, J. Naqui, and E. Biempica, "Continuous touch gesture recognition based on RNNs for capacitive proximity sensors," *Personal and Ubiquitous Computing*, pp. 1–18, 2020.

[49] Y. Yu, X. Si, C. Hu, and J. Zhang, "A review of recurrent neural networks: LSTM cells and network architectures," *Neural Computation*, vol. 31, no. 7, pp. 1235–1270, 2019.

[50] B. Yana and T. Onoye, "Air-writing recognition based on fusion network for learning spatial and temporal features," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E101.A, no. 11, pp. 1737–1744, 2018.

[51] R. Saini, P. Kumar, P. P. Roy, and D. P. Dogra, "A novel framework of continuous human-activity recognition using Kinect," *Neurocomputing*, vol. 311, pp. 99–111, 2018.

[52] C. Ito, X. Cao, M. Shuzo, and E. Maeda, "Application of CNN for human activity recognition with FFT spectrogram of acceleration and gyro sensors," in *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*, pp. 1503–1510, Singapore, 2018.

[53] F. Chollet, "Xception: deep learning with depthwise separable convolutions," in *Proceedings -30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, pp. 1800–1807, Honolulu, HI, USA, 2017.