Air-Writing Recognition using Deep Convolutional and Recurrent Neural Network Architectures

Grigoris Bastas, Kosmas Kritsis and Vassilis Katsouros
Institute for Language and Speech Processing (ILSP)
Athena R.C.
Athens, Greece
{g.bastas, kosmas.kritsis, vsk}@athenarc.gr

***Abstract*—In this paper, we explore deep learning architectures applied to the air-writing recognition problem where a person writes text freely in the three dimensional space. We focus on handwritten digits, namely from 0 to 9, which are structured as multidimensional time-series acquired from a Leap Motion Controller (LMC) sensor. We examine both dynamic and static approaches to model the motion trajectory. We train and compare several state-of-the-art convolutional and recurrent architectures. Specifically, we employed a Long Short-Term Memory (LSTM) network and also its bidirectional counterpart (BLSTM) in order to map the input sequence to a vector of fixed dimensionality, which is subsequently passed to a dense layer for classification among the targeted air-handwritten classes. In the second architecture we adopt 1D Convolutional Neural Networks (CNNs) to encode the input features before feeding them to an LSTM neural network (CNN-LSTM). The third architecture is a Temporal Convolutional Network (TCN) that uses dilated causal convolutions. Finally, a deep CNN architecture for automating the feature learning and classification from raw input data is presented. The performance evaluation has been carried out on a dataset of 10 participants, who wrote each digit at least 10 times, resulting in almost 1200 examples.**

***Index Terms*—air-writing, gesture recognition, deep learning, LSTM, CNN, TCN**

I. Introduction

Automatic gesture recognition has gained the interest of various research projects due to its convenient and natural characteristics in conveying communication information in Human-Computer Interaction (HCI) related applications such as music interaction [1], computer gaming [2], robotics [3], sign-language interpretation [4] and automotive systems [5], amongst others. Different from traditional HCI input means, such as moving the mouse, typing on the keyboard, and touching a screen, modern gesture recognition systems usually receive specific body or hand movements in order to control their output by utilizing different types of sensors, including gyroscopes, optical sensors, electromyography sensors etc.

In this regard, the term "*Air-writing*" refers to the process of writing linguistic characters or words in mid-air or free 3D space by hand or finger gestures. Air-writing differs from conventional handwriting since the latter contains a discrete stroke with pen-up and pen-down motions, while the former lacks such a delimited sequence of writing events. Furthermore, the absence of visual and tactile feedback as well as the high interdependence between the target gestures makes the character recognition task even more challenging.

From the user's point of view, there are three different ways to accomplish the air-writing process, namely the "connected", "isolated" and "overlapped" [6]. Similar to conventional handwriting, with the "connected" method the user may write a sequence of multiple characters in free space, for example from left to right. The second approach allows the user to perform the writing gesture of each character separately, within the boundaries of an imaginary box in the 3D space. Finally, the most complicated and uncommon technique is the "overlapped", since it requires to write adjacent characters that are stacked one over another in the same imaginary box. In general, the size and shape of "connected" characters heavily vary when writing in an unconstrained area, which affects the recognition accuracy. It is difficult even for the same user to repeat the same motion trajectory in the same position. On the contrary, limiting the writing space within an imaginary box facilitates the character segmentation, although this type of writing is not natural [7].

The recent advancements in Motion Capture (MoCap) systems in addition to the computational power of modern computers, motivated the HCI research community to examine the application of air-writing in various interfaces. Some systems employ handheld markers [8] or custom made gloves [9] for capturing the motion trajectory. However, Low-cost depth sensors such as the Microsoft Kinect [10], the Intel RealSense [11] and the Leap Motion Controller (LMC) [12], are commonly employed in air-writing proposals due to the absence of intrusiveness while tracking human motion trajectories in real-time. Kinect-based air-writing interfaces [13] are designed to detect long-range 3D body postures, while the LMC and RealSense sensors provide millimeter-level accuracy and focus on tracking hand and finger motion trajectories [14]–[16]. Even thought such sensors facilitate the motion tracking problem, it is still hard to distinct the actual writing activity from other arbitrary hand movements in the continuous input stream of motion tracking data, considering the absence of a

We acknowledge support of this work by the project "*Computational Sciences and Technologies for Data, Content and Interaction*" (MIS 5002437), sub-project "*Technologies for Content Analysis in Culture*", implemented under the Action "*Reinforcement of the Research and Innovation Infrastructure*", funded by the Operational Programme "*Competitiveness, Entrepreneurship and Innovation*" (NSRF 2014-2020) and co-financed by Greece and EU (ERDF).

978-1-7281-9966-5/20/$31.00 ©2020 IEEE
DOI 10.1109/ICFHR2020.2020.00013

physical delimitation mechanism. This is a substantial problem, different from the character or word recognition task. Furthermore, the majority of the existing air-writing systems focus on letter and digit character recognition due to lack of available word-level training examples [17].

In this sense, our proposal has been designed so as to handle both detection and recognition of air-written characters mixed with other movements without lingering the user experience. Especially, we developed a web interface based on the LMC sensor for collecting 1200 air-written digits (i.e. 0-9). Our method provides visual feedback to the user and uses an efficient fingertip tracking approach for simulating pen-up and pen-down switching. We model the air-writing recognition process as a classification problem. Both, dynamic 3D trajectories (time-series) and their corresponding static 2D stroke projections (images) were used to train and evaluate various deep learning architectures based on convolutional and recurrent operations. The evaluation results present almost perfect recognition rates within only a few milliseconds, indicating that our system can be deployed in a real-time setting.

The rest of the paper is organized as follows. In the next section, we briefly review some relevant prior work in air-writing. Section III describes the overall methodology, including the LMC web interface, the dataset acquisition procedure and the considered deep learning architectures. In Section IV, we specify the experimental setup and present the collected results. Finally, the conclusions and future directions are discussed in Section V.

II. Related Work

During the last decade, various interfaces that employ air-based writing have been proposed. Typically, the main interest is the spatiotemporal trajectory information of the hand motion while writing in the air. Hence, the trajectory information of an air-written character can be segregated into two core dimensional representations. In other words, the air-writing recognition problem can be examined either by focusing on the temporal interdependence between the time-steps of the trajectory sequence, or by considering the spatial projections of the motion trajectory on vertical and perpendicular imaginary 3D planes. We call these two approaches "*dynamic*" and "*static*" respectively.

By following the static modeling approach, the air-writing trajectory can be represented as a 2D image, resembling traditional handwriting; thus allowing to leverage previous techniques that have been applied successfully in the Optical Character Recognition (OCR) problem [18]. On the other hand, the temporal dimension of the motion trajectory is represented by consecutive frames of signal measurements provided by the underlying motion tracking sensor, organized in time-series with a given sampling frequency. Usually, each frame contains features such as the hand's instant velocity and its coordinates relative to the referential point system of the sensor, in addition to other application-defined measurements, for instance the angles between the consecutive points of the trajectory and inertial angular velocity [6], also known as "hand-crafted" features. In this regard, dynamic gesture recognition approaches [1], [19] can be adopted to the air-writing recognition problem [5].

Most prominent methods for analyzing static trajectory images employ Convolutional Neural Networks (CNNs). For instance, in [8] the authors present a marker-based air-writing framework that uses a generic video camera and apply color based segmentation for tracking the writing trajectory. Then a pre-trained CNN model is employed for classifying the static trajectory images. However the illumination fluctuations affect the color-based segmentation and the overall system performance. A marker-less approach was presented in [20], where the 3D fingertip coordinates provided by an LMC sensor were mapped frame by frame to consecutive trajectory images and after a pre-processing phase, the normalized trajectory images were used to train a CNN model, reporting an average recognition rate of 98.8%. However, deep CNN architectures contain lots of redundant filter parameters that lead to huge computational cost, limiting their deployability. In this regard, the authors in [21] propose a unified algorithm to effectively prune the deep CNN models for in-air handwritten Chinese character recognition with only 0.17% accuracy drop, compared to the baseline VGG-like [22] CNN that achieves the state-of-the-art accuracy of 95.33%.

Common algorithms for modeling the dynamic aspect of air-writing gestures are Hidden Markov Model (HMM) [6], [9], [15], Dynamic Time Warping (DTW) [13], [14], and support vector machine (SVM) [9]. However these methods work well only with simple gestures that have low inter-class similarities, thus reporting low accuracy in the air-writing recognition task. To overcome this limitation, Recurrent Neural Networks (RNNs) based on bi-directional Long Short-Term Memory units (BLSTM) have presented significant improvement on the given recognition task [17], [23]. Only the recent years a few fusion-based techniques have been proposed, combining CNN with either LSTM [16] or BLSTM classifiers [7], [24]. These networks outperform all of the previous works since they can combine both spatial and temporal feature learning.

III. Methodology

A. Data collection

A simple interactive web interface based on the LMC javascript version 2 API was designed for collecting air-written digits (0-9), requiring minimal instructions. As it is depicted in Fig 1, the LMC is placed on a tabletop orientation in front of a screen that is used to provide optical feedback to the user. Based on a calibration process, an effective air-writing spotting algorithm is applied to identify the start/end points of the dynamic gesture, allowing the user to use the screen as a physical boundary. Furthermore, a blue box on the screen specifies the writing area which can be adjusted by the user without affecting the representation of the captured frame data. For the dataset collection we considered 10 participants (8 males and 2 females) between 23 to 50 years old, without applying any uni-stroke or other writing constrains. The participants wrote each

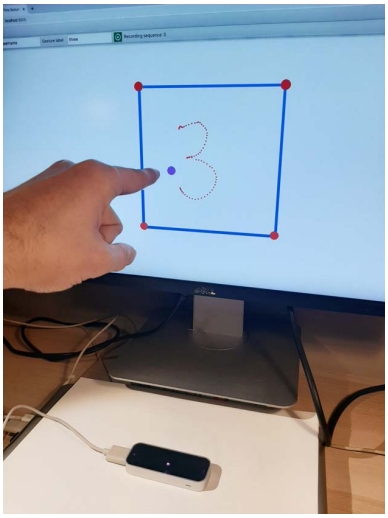


Fig. 1: An example of the LMC orientation and the web interface.

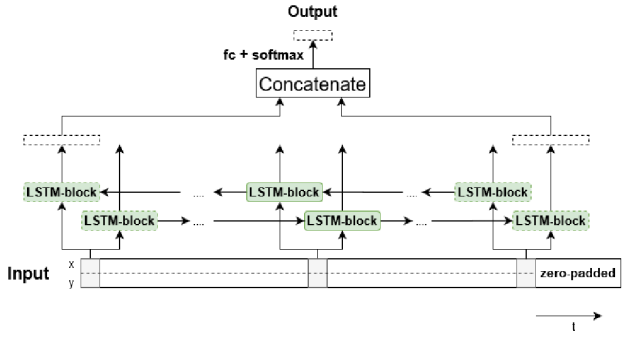


Fig. 2: Bidirectional LSTM (BLSTM) trained on a zero-padded sequence of 2-dimensional digit trail points.

digit at least 10 times resulting in almost 1200 examples. Every recording is a time-series containing raw tracking information provided by the LMC API with a sampling frequency of 60Hz. The web interface and the collected data are hosted online and can be accessed in [25].

B. Architectures

A variety of methods were used in order to tackle the problem of digit recognition. There were two main approaches, the dynamic and the static. Several experiments were ran in order to compare or even combine the two strategies.

1) Dynamic Approach: In this approach, the input is represented as a sequence of points in the 2-dimensional plane. It can be expected that by following the trail of the user's fingertip frame-by-frame, a sequential model can successfully manage the information and make correct predictions. Naturally, since there is more than one way to draw any digit, robustness is an important factor when selecting the models to be tried out. The sequential architectures that were tested are: TCN, LSTM, BLSTM and CNN-LSTM. A "many-to-one"

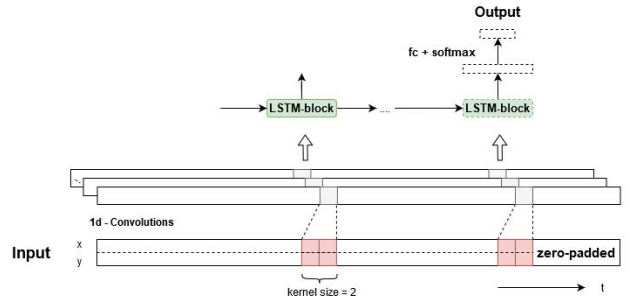


Fig. 3: 1D Convolutional Filtering followed by a LSTM (CNN-LSTM) trained on an oversampled sequence of 2-dimensional digit trail points.

configuration was used in all cases. This means that only the last output was taken under consideration when computing the cross-entropy training loss and, in the case of LSTM's bidirectional configuration, the last outputs of both directions were combined by simply concatenating the two vectors as depicted in Fig. 2.

LSTM: This architecture belongs to the family of RNNs. LSTMs were introduced by Hochreiter & Schmidhuber [26] and comprise a very common architecture for sequential modeling. They are mainly characterized by their capability to grasp long-term dependencies better that other RNNs. A one-layer LSTM and a bidirectional version (BLSTM) of it were used, with 300 hidden units in each block. A linear fully connected layer with log-softmax regression was used to classify each input to one among 10 classes, each for one digit.

CNN-LSTM: In the architecture described right above, an extra layer that performs 1-dimensional convolutions was added right after the input. For the convolutions, 300 filters were used, with kernel size 2 and stride 1, as illustrated in Fig. 3.

TCN-Dynamic: The family of Temporal Convolutional Network (TCN) architectures employs causal convolutions with dilations and handles sequences of data by mapping the extracted features to a sequence of predictions of the same length as the input. In this work, among the various types of TCNs, it is the one proposed in [27] that was selected, having considered its simplicity and the fact that its performance was already tested on a similar task, namely on the Sequential MNIST dataset. In the current configuration, as we move into deeper layers, the dilation factor increases for each layer l by 2^l. We use kernel of size 7, 25 hidden units per layer and a stack of 6 layers, thus ensuring a receptive field of 64 frames in the past.

2) Static Approach: As for the static approach, a representation of the drawn digits as images was required. To achieve this, each point was mapped to a rectangular area so that binary images of size 28x28 could be created by filling in the right pixels. Min-max normalization was applied on each image, in order to ensure small variance in terms of size and positioning of the depicted digits. Two architectures were tried in this

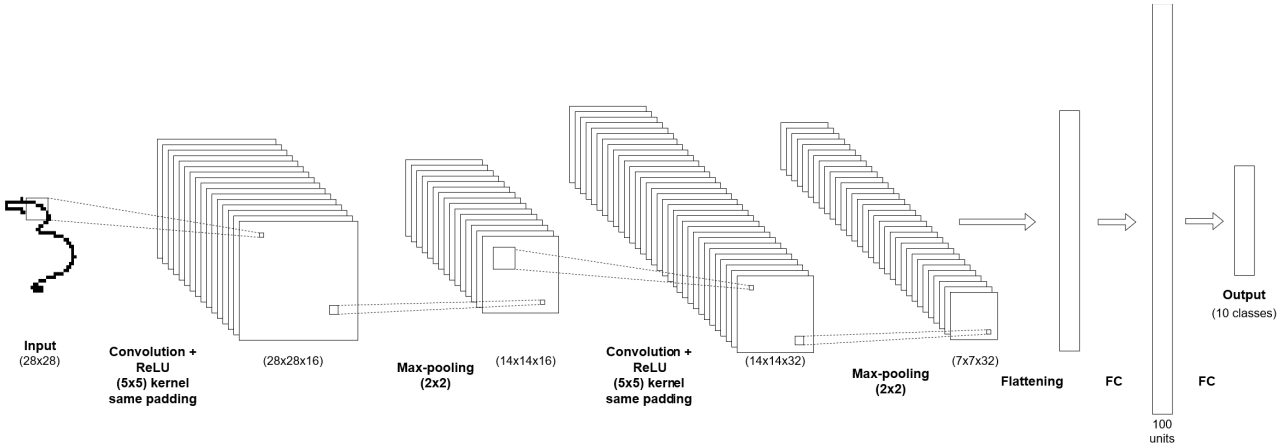


Fig. 4: 2D CNN fed with image representations (28x28) of the drawn digits.

context:

CNN: A model with two 2D convolutional layers, each followed by 2D batch normalization, a ReLU activation function and 2D max pooling. The first layer consisted of 16 filters and the second of 32. The kernel size for each convolutional layer was 5 and the kernel size of the max pooling was 2. The images were given directly as input to this model. The encoded input was then fed to a 2-layer fully-connected network followed by a softmax function.

TCN-Static: A TCN with 8 layers was used following Shaojie Bai et. al [27] in respect to the way they have applied it on the Sequential MNIST dataset. The array representing the images was unfolded in such a manner that each row is appended to the previous one. This sequence of zeroes and ones (i.e. the values of the image pixels) was the actual input to the TCN.

3) Combining the Static and the Dynamic Approach: Employing pairs of static and dynamic architectures was selected as a tactic in order to test weather such a combination could lead to surpass obstacles that each approach might encounter. A Fuzzy model having an LSTM and a CNN as its building blocks, was created. The outputs of each model (i.e. a 10-dimensional vector) were concatenated and fed to a 2-layer linear fully-connected network with a final softmax activation layer.

IV. Evaluation and Results

A. Experimental setup

Among the collected data, there were some faulty recordings (i.e. the right hand of the participant was wrongly mapped as left hand). There were 35 such cases among all the samples. After discarding these samples, the dataset was randomly split to training, validation and test sets. There were 200 samples kept for the test set and 100 for the validation set.

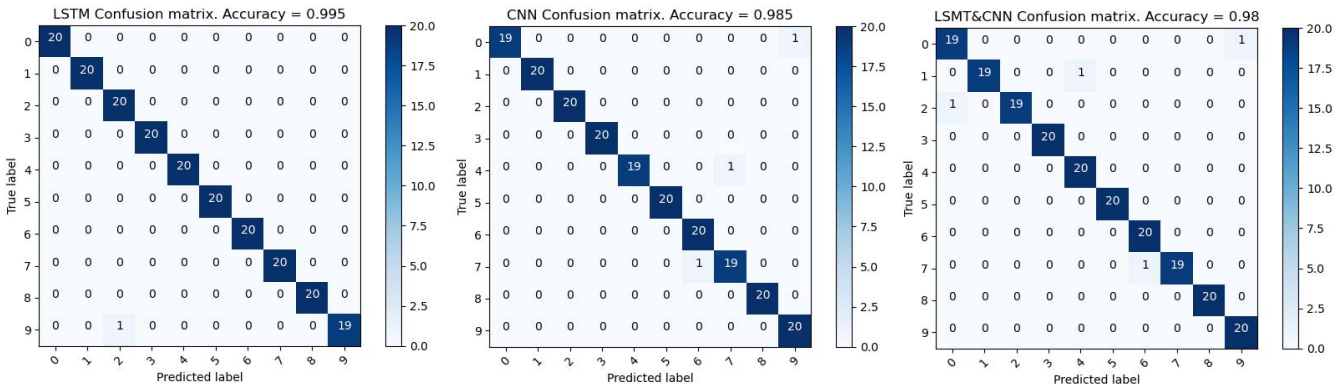
The sequence lengths strongly vary among the recordings. For example the digit "1" usually requires a lot fewer frames for its recording in contrast to the other digits. Zero-padding was employed in order to create batches of the same length (i.e. the maximum sequence length in each batch). This choice implies that, while training, one needs to keep track of the sequence length values in order to take only the last timestep's output into account.

Each network was developed and trained with PyTorch for 1000 epochs with batch size of 64 samples. At each epoch the training data were shuffled anew. The validation loss and accuracy indicated the final model's parameters to be stored and tested. The experiments were conducted on a computer with a 16 GB RAM, an Intel i7-8750H CPU, and a GeForce GTX 1060 6 GB GPU. The training code can be found in [25].

B. Results

After having stored the best models for each different architecture, they were then all evaluated on the test set. The accuracy for each model is presented on Table I. Both the static and the dynamic approaches lead to remarkable model performance. The LSTM gives the best results among every other architecture. The BLSTM follows with a slightly lower performance. The CNN comes third and outperforms the fuzzy model LSTM&CNN. Then follow the CNN-LSTM and the TCN-Dynamic models that exhibit the same performance. The TCN-Static architecture comes last. Something that should also be noted is the approximately 100% better runtime performance of the two TCN models as compared with the rest. This goes in tandem with their relatively small number of trainable parameters and the use of dilations.

The extracted Confusion Matrices indicate that the sources of error between the two best performing models, the LSTM (Fig. 5a) and the CNN (Fig. 5b), each being representative of the two distinct strategies, were all different. One should notice that the LSTM produced only one mismatched example,where a "9" was classified as "2". This can be explained by the similarity of the trail (when seen sequentially) of this instance of "9" with the trail of many instances of digit "2". As for the CNN, the results present three mismatched samples. In the case of the fuzzy model (LSTM&CNN), a slight decrease



(a) Confusion matrix for LSTM. (b) Confusion Matrix for CNN (c) Confusion Matrix for LSTM&CNN

Fig. 5: Confusion Matrices during testing

TABLE I: Test Accuracy and Runtime

Models	Accuracy	Runtime	Trainable Parameters
CNN	98.5%	0.72 s	94,344
CNN-LSTM	97.5%	0.94 s	726,310
LSTM	99.5%	0.82 s	367,810
BLSTM	99%	0.85 s	367,810
TCN-Dynamic	97.5%	0.07 s	49,410
TCN-Static	97%	0.09 s	66,910
LSTM&CNN	98%	1.21 s	827,374

in accuracy was observed. Interestingly, one can notice in the corresponding Confusion Matrix (see Fig. 5c), that although the mismatch of the LSTM was resolved, all of the CNN's mismatches are present with an additional fourth one.

In the case of CNN, increasing the number of convolutional layers led to inferior performance. This is due to the rather small image representation of the trails (28x28) and the simplicity of the input values (i.e. 0 or 1). Employing more pixels for the image representation was found to be redundant. Similarly, in the case of trail data, increasing the LSTM layers or applying bidirectional configuration only made the training phase slower without improving the overall performance.

In order to examine the dependence of each model on user specific features, we use a participant cross-validation protocol, and more specifically, a setup where one participant is left out for testing while the rest of them are used for training. For each among the 10 participants, we keep 100 samples, 10 from each class. The results presented in Table II reveal a significant decrease in the performance of all models. The best performing ones are those that employ static representations. CNN outperforms all models. The dynamic recognition strategy proves more prone to errors in this setup. The variability of the recognition rates among different folds is also notable, especially in the case of dynamic architectures. This behavior leads us to the conclusion that the number of users and samples in our dataset do not suffice to produce generalized and user independent models.

However, fine-tuning our models by injecting samples of the tested user into the training set (and removing them from the test set) is a strategy that can be used to improve performance. We followed this method for the CNN and LSTM models on participants #5 and #8. Results are depicted on Table III. In all cases, we notice a monotonic increase of the recognition rates.

V. Conclusions and Future Work

The success of our relatively small models in predicting digit labels owes a lot to the strategy followed for the data collection. The fact that so few collected data were enough for achieving such a good performance supports this exact point. The use of a vertical plane (e.g. a computer screen), in combination with the distance threshold used to indicate where the drawing is disrupted, should be considered as an important contribution of this work.

Furthermore, the experiments that were conducted, revealed both certain capabilities and limitations of the two different approaches and of the neural architectures that were tested. It is noteworthy that we were able to point out the potentials, not only of the commonly employed static approach for handwriting recognition, but also those of a sequential handling of the problem.

As future work, one idea is to enrich our Leap Motion air-writing dataset with letter characters. We can also engage in a comparison of the performance of the different architectures used herein, with other publicly available datasets such as the 6DMG motion and gesture dataset [28]. It would be interesting to experiment on new ways of combining both the dynamic and the static approaches, seeking how to benefit from the advantages of both. Finally, real-time implementations of the recognition system should be examined, for instance, by employing sequential architectures to predict digit labels in faster time intervals, even before their drawing gets completed.

Acknowledgment

We would like to thank all the colleagues from the *Athena Research Center* for participating in the data collection process.

TABLE II: Test Accuracy for Participant Cross-Validation

Fold No	CNN	LSTM&CNN	TCN-Static	LSTM	CNN-LSTM	TCN-Dynamic
0	93%	86%	84%	92%	89.00%	82%
1	88%	84%	84%	43%	50.00%	46%
2	85%	83%	79%	58%	67.00%	41%
3	98%	97%	89%	77%	80.00%	77%
4	97%	98%	90%	93%	98.00%	96%
5	72%	66%	65%	74%	75.00%	54%
6	90%	94%	74%	91%	74.00%	71%
7	87%	82%	78%	77%	68.00%	77%
8	97%	96%	94%	91%	96.00%	86%
9	95%	90%	83%	88%	90.00%	85%
Average	**90.20%**	**87.60%**	**82.00%**	**78.40%**	**78.70%**	**71.50%**
STD	**7.87%**	**9.71%**	**8.46%**	**16.67%**	**14.97%**	**18.41%**

TABLE III: Recognition rates of subjects with ids #5 and #8 using CNN and LSTM for different number of samples injected into the training set

Models	Number of samples (per class)	Participant #5	Participant #8
CNN	**0**	72.00%	97.00%
	2	80.00%	98.80%
	5	86.00%	100.00%
LSTM	**0**	74.00%	91.00%
	2	87.50%	96.20%
	5	88.00%	98.00%

REFERENCES

[1] K. Kritsis, M. Kaliakatsos-Papakostas, V. Katsouros, and A. Pikrakis, "Deep convolutional and lstm neural network architectures on leap motion hand tracking data sequences," in *2019 27th European Signal Processing Conference (EUSIPCO)*, Sep. 2019, pp. 1–5.

[2] J. Pirker, M. Pojer, A. Holzinger, and C. Gütl, "Gesture-based interactions in video games with the leap motion controller," in *Human-Computer Interaction. User Interface Design, Development and Multimodality*, M. Kurosu, Ed. Springer, 2017, pp. 620–633.

[3] A. Zlatintsi, I. Rodomagoulakis, P. Koutras, A. C. Dometios, V. Pitsikalis, C. S. Tzafestas, and P. Maragos, "Multimodal signal processing and learning aspects of human-robot interaction for an assistive bathing robot," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2018, pp. 3171–3175.

[4] M. Simos and N. Nikolaidis, "Greek sign language alphabet recognition using the leap motion device," in *Proceedings of the 9th Hellenic Conference on Artificial Intelligence*. New York, NY, USA: Association for Computing Machinery, 2016.

[5] H. Wu, Y. Wang, J. Liu, J. Qiu, and X. L. Zhang, "User-defined gesture interaction for in-vehicle information systems," *Multimedia Tools and Applications*, vol. 79, no. 1, pp. 263–288, 2020.

[6] M. Chen, G. AlRegib, and B. 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, June 2016.

[7] B. Yana and T. Onoye, "Fusion networks for air-writing recognition," in *MultiMedia Modeling*, K. Schoeffmann, T. H. Chalidabhongse, C. W. Ngo, S. Aramvith, N. E. O'Connor, Y.-S. Ho, M. Gabbouj, and A. Elgammal, Eds. Cham: Springer, 2018, pp. 142–152.

[8] P. Roy, S. Ghosh, and U. Pal, "A cnn based framework for unistroke numeral recognition in air-writing," in *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, Aug 2018, pp. 404–409.

[9] C. Amma, M. Georgi, and T. Schultz, "Airwriting: a wearable handwriting recognition system," *Personal and Ubiquitous Computing*, vol. 18, no. 1, pp. 191–203, feb 2013.

[10] Kinect for windows. Accessed April 2020. [Online]. Available: https://developer.microsoft.com/en-us/windows/kinect/

[11] Intel realsense depth and tracking cameras. Accessed April 2020. [Online]. Available: https://www.intelrealsense.com/

[12] Ultraleap: Digital worlds that feel human. Accessed April 2020. [Online]. Available: https://www.ultraleap.com/

[13] 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, Sep. 2016.

[14] R. Aggarwal, S. Swetha, A. M. Namboodiri, J. Sivaswamy, and C. V. Jawahar, "Online handwriting recognition using depth sensors," in *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, Aug 2015, pp. 1061–1065.

[15] M. Chen, G. AlRegib, and B. 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, June 2016.

[16] 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, Jan 2020.

[17] 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, nov 2017.

[18] Y. LeCun, "Neural networks and gradient-based learning in ocr," in *Neural Networks for Signal Processing VII. Proceedings of the 1997 IEEE Signal Processing Society Workshop*, Sep. 1997, pp. 255–255.

[19] M. Maghoumi and J. J. LaViola, "Deepgru: Deep gesture recognition utility," in *Advances in Visual Computing*, G. Bebis, R. Boyle, B. Parvin, D. Koracin, D. Ushizima, S. Chai, S. Sueda, X. Lin, A. Lu, D. Thalmann, C. Wang, and P. Xu, Eds. Springer International Publishing, 2019, pp. 16–31.

[20] J. Hu, C. Fan, and Y. Ming, "Trajectory image based dynamic gesture recognition with convolutional neural networks," in *2015 15th International Conference on Control, Automation and Systems (ICCAS)*, Oct 2015, pp. 1885–1889.

[21] J. Gan, W. Wang, and K. Lu, "Compressing the cnn architecture for in-air handwritten chinese character recognition," *Pattern Recognition Letters*, vol. 129, pp. 190 – 197, 2020.

[22] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.

[23] V. Frinken and S. Uchida, "Deep blstm neural networks for unconstrained continuous handwritten text recognition," in *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, Aug 2015, pp. 911–915.

[24] 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, pp. 8855–8864, Oct 2019.

[25] Leap motion air-writing web interface, dataset and training code. [Online]. Available: https://github.com/kosmasK/air-writing-recognition

[26] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[27] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[28] M. Chen, G. AlRegib, and B.-H. Juang, "6dmg: A new 6d motion gesture database," in *Proceedings of the 3rd Multimedia Systems Conference*. New York, NY, USA: Association for Computing Machinery, 2012, p. 83–88.