

Continuous Motion Numeral Recognition using RNN Architecture in Air-Writing Environment

Adil Rahman¹, Prasun Roy², and Umapada Pal²

¹ Department of Information Technology, Heritage Institute of Technology
adil.rahman.it20@heritageit.edu.in

² Computer Vision and Pattern Recognition Unit, Indian Statistical Institute
prasunroy.pr@gmail.com, umapada@isical.ac.in

Abstract. Air-writing, defined as character tracing in a three dimensional free space through hand gestures, is the way forward for peripheral-independent, virtual interaction with devices. While single unistroke character recognition is fairly simple, continuous writing recognition becomes challenging owing to absence of delimiters between characters. Moreover, stray hand motion while writing adds noise to the input, making accurate recognition difficult. The key to accurate recognition of air-written characters lies in noise elimination and character segmentation from continuous writing. We propose a robust and hardware-independent framework for multi-digit unistroke numeral recognition in air-writing environment. We present a sliding window based method which isolates a small segment of the spatio-temporal input from the air-writing activity for noise removal and digit segmentation. Recurrent Neural Networks (RNN) show great promise in dealing with temporal data and is the basis of our architecture. Recognition of digits which have other digits as their sub-shapes is challenging. Capitalizing on how digits are commonly written, we propose a novel priority scheme to determine digit precedence. We only use sequential coordinates as input, which can be obtained from any generic camera, making our system widely accessible. Our experiments were conducted on English numerals using a combination of MNIST and Pendigits datasets along with our own air-written English numerals dataset (ISI-Air Dataset). Additionally, we have created a noise dataset to classify noise. We observe a drop in accuracy with increase in the number of digits written in a single continuous motion because of noise generated between digit transitions. However, under standard conditions, our system produced an accuracy of 98.45% and 82.89% for single and multiple digit English numerals, respectively.

Keywords: Air Writing · Handwritten Character Recognition · Human-Computer Interaction · Long Short Term Memory · Recurrent Neural Network

1 Introduction

Recent advancements in gesture recognition and motion tracking have opened up new avenues in human-computer interaction. Most conventional devices use a

keyboard, mouse or a touchscreen as their input medium. In some cases, however, these input options may not be available or suitable. Gesture and motion tracking algorithms have introduced simpler and more advanced interaction mechanisms for the user, involving simple patterns in the form of hand gestures that the user memorizes as shortcuts to certain commands. These simple patterns, however, are not sufficient for the detection of complex string input.

“Air-writing”, referred to as the writing of characters using free hand movements in three-dimensional space utilizing six degrees of freedom, is a promising advancement towards equipment-independent human-computer interaction. This bears its own set of challenges for identification of characters, and gets more complex for a string of characters. The initial challenge is to track the motion of an object (finger, pen, etc.) using which the user writes on an imaginary surface. Air-writing lacks haptic feedback. Two dimensional input space has the benefit of concrete anchor points and pen-up/pen-down delimiters to separate consecutive characters. These advantages are lost when the characters are drawn in a three dimensional free space. With the recent advent of advanced motion tracking devices with depth sensors like LEAP Motion [2] and Kinect [1], equivalents of the pen-up and pen-down gestures may be captured, even in three-dimensional space. However, these ancillary sensors are expensive and not available in most of the common devices, inhibiting general accessibility to such systems. Almost all smart devices are equipped with a camera these days. It is thus far more feasible to relay the input directly to a generic device camera for recognition.

The challenge persists for recognition of a string of characters which requires character segmentation. There is an absence of delimiters between characters, since device cameras lack stereoscopic information that is crucial for decent recognition accuracy. Presence of redundant strokes between characters, along with stray marker movements, contributes to the noise present in the air-written input. Moreover, some characters might be sub-shapes of other characters. For instance, 0 is a sub-shape of 9. This makes it hard to determine if a character has been completely constructed and the user is ready to move on to the next character or if the character is still being traced by the user. For words that are constructed from alphabets corresponding to a particular language, recognition accuracy is often boosted by referring a language model [5, 6] that corrects the recognized word to the most likely word. However, this technique is impractical in case of numbers due to the lack of a language model that predicts what the numbers are most likely to be.

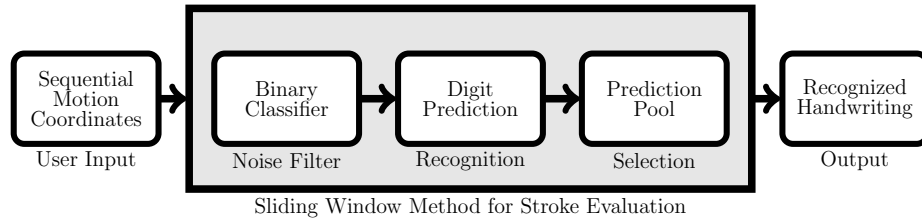


Fig. 1: Air-Writing Numeral Recognition System Overview

Our present work employs marker-based segmentation as introduced by Roy et al. in [13]. Requiring only a simple device camera input, they have made the need for expensive hardware equipment redundant. For the purpose of efficient segmentation, we have incorporated a sliding window approach in our Long-Short Term Memory (LSTM) model. LSTM is a promising neural network model for spatio-temporal data like air-writing. We have incorporated a dual-network architecture for noise elimination and numeral recognition. We have introduced a novel priority scheme to tackle the problem of recognizing digits which involve sub-shapes of other digits. An overview of our system is presented in Figure 1. Our method eliminates the use of language models which are inherently unsuitable for certain forms of input like numbers, names and alpha-numeral entries. We instead present a language-independent model that focuses on character-wise recognition, as opposed to word recognition. We have performed our experiments on English numerals, training our recognition model using Pendigits, MNIST, and ISI-Air, which is our custom English numeral air-writing dataset. For noise elimination, we have created a noise dataset using various possible combinations of digits and stray motions in our air-writing environment.

The rest of this paper is organized as follows. The next section explores the research previously done in this field. Section 3 outlines our proposed methodology. The experimental results have been provided and discussed in Section 4. Section 5 concludes the paper.

2 Related Works

Hardware support has been the crux of most air writing approaches that rely on tracking hand movements and gestures. Amma et al. [3] wirelessly captured hand gestures by attaching motion sensors with accelerometers and gyroscopes to the back of the user’s hand. Chen et al. in [6] came up with a controller-free technique where they used LEAP motion capture device for continuous finger motion tracking. Zhang et al. in [15] and Ye et al. in [14] used Kinect sensors for extracting depth, motion and color information for hand segmentation and finger detection. However, dependency on such equipment and additional hardware requirements prohibits air-writing systems from being commonly accessible. Roy et al. in [13] proposed a technique for hand motion tracking that relied only on the video feed from generic device camera, thus making their air-writing framework accessible for common purposes. The lack of delimiters between characters in air-writing medium poses a challenge for recognition of multiple characters in a continuous flow. To tackle this problem, several authors have defined explicit manual delimiters to distinguish each character from the next. Kim et al. [10] have solved the lack of distinction between pen-ups and pen-downs by using a 3D handwriting ligature model that describes the shapes of pen-up motions. Huang et al. [8] defined explicit hand gestures to signify character segmentation. Roy et al. [13] have used Convolutional Neural Network (CNN) as the character recognition model where they defined a velocity threshold criterion to indicate the end of a character. Oh et al. in [12] used Fisher Discriminant Analysis for character recognition. However, these works remain limited to recognizing single

unistroke characters only and fail in case of multiple characters drawn in a continuous motion. A window-based algorithm has been adopted in previous works [4, 6] but the window properties and operations are based on empirically decided time or velocity constants, both of which are user-dependent factors and thus may not function universally. Amma et al. in [3, 4] used Hidden Markov Model (HMM) for continuous text recognition in air writing, as have Chen et al. in [6] and Lee et al. in [11]. Junker et al. [9] have used a similarity search method for the initial stage in continuous gesture recognition, and have used HMM in the second phase for further classification. These works take advantage of the spatio-temporal property of the input for multiple character recognition. HMM has been a popular choice in continuous writing recognition models, given its effectiveness in working with temporal data and implicit segmentation abilities.

To the best of our knowledge, there hasn't been any previous consideration to a character being a sub-shape of its successor in a string. Moreover, most of these works rely on expensive hardware equipment that are not commonly available. In our work, we use video feed from the generic webcams, with Recurrent Neural Network (RNN) as the recognition model, which is more efficient than HMM for spatio-temporal data. Prediction is done using only sequential coordinate information. Details like velocity and time taken to write haven't been taken into consideration owing to its variability among different users, making it a poor classification feature. An additional priority scheme tackles the issue of recognition of digits which have some other digit as their sub-shape.

3 Proposed Methodology

3.1 Motion Tracking Method

We have adopted this method from the previous work of Roy et al. [13] for detecting and tracking the input coordinates from the video feed. To avoid the complexities of the range of variability of skin tone for finger tracking, a simple marker of a fixed color has been used for tracing characters. The visible portion of the marker can then be segmented from the background using a color segmentation technique. The segmented binary image thus obtained is used for identifying the marker by eliminating the noise. If the color of the marker can be distinctly demarcated from the background, the contour with the largest area in the segmented image is considered to be the marker, with its tip as the top-most point on the contour boundary with the lowest y -coordinate value. The air-writing input coordinates consist of a sequential set of points. The number of frames rendered per second is estimated as $N_{FPS} = \frac{1}{t_{update}}$, where t_{update} is the time taken to process the last video frame. Let the changes in the position of the marker between two consecutive frames in the x and y directions be Δ_x and Δ_y , respectively. The normalized instantaneous velocity of the tip of the marker is computed as:

$$dx = \frac{1}{N_{FPS}} \sum_t^{t+N_{FPS}} \Delta_x \quad \text{and} \quad dy = \frac{1}{N_{FPS}} \sum_t^{t+N_{FPS}} \Delta_y$$

In order to interpret when the user has finished writing a string of characters, a velocity threshold v_t is used. If both dx and dy are less than v_t , it is assumed that the marker is static and the user has finished tracing the string of characters. The trajectory of the marker for the input string of characters is approximated as straight line segments between marker tip positions in each consecutive video frame while the tip is in motion.

3.2 Numeral Recognition

In case of a single digit being traced out in a three-dimensional environment, digit prediction can be done simply by passing the final frame to a recognition model. However, in the case of string of digits written continuously one after the other in a three-dimensional environment, i.e., without the presence of delimiters, it becomes challenging to isolate and recognize the digits from the input. Therefore, for predicting such types of inputs, we need to analyze the entire motion in which the digits have been traced out.

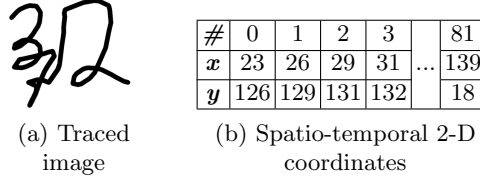


Fig. 2: Air-writing gestures recorded as sequential coordinates

The input to our system is thus in the form of a spatio-temporal series of data as seen in Figure 2. The data is solely in the form of a sequential set of points, as traced by the marker tip in the video frames. Other information like velocity of the marker tip and time taken by the user to draw the stroke is not recorded as input to our system, owing to the variability of writing speeds of different people and the frame lag in different video capturing devices.

Dual Network Configuration: Upon analyzing the input, we observe a lot of noisy signals derived from the presence of stray hand motion and overlapping digits written in a continuous motion. This noise hinders numeral prediction. Thus, to overcome this problem, we propose a dual network configuration. The dual network removes noise and subsequently detects digits. It consists of a binary classifier for noise filtration and another 10-class classifier for digit recognition. Owing to the temporal nature of the input, we have decided to use RNN-LSTM architecture as illustrated in Figure 3 for both the networks, given their suitability for such types of inputs.

The architecture of the employed network (Figure 3) consists of a feature extraction block followed by a classification block. A 28x28 grayscale serves as the input for the feature extraction block which is passed through two consecutive LSTM layers. Both LSTM layers have 128 hidden units and use rectified linear units (ReLU) as their activation function. The classification layer comprises of

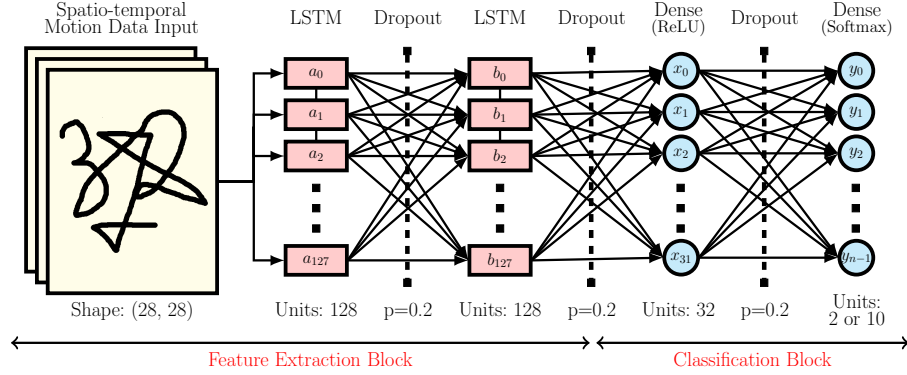


Fig. 3: RNN-LSTM architecture employed in our methodology

two fully connected layers having 32 and n neurons, respectively, where n is 2 in case of binary classifier and 10 in case of 10-class digit classifier. The two fully connected layer use ReLU and softmax as their activation functions, respectively. A dropout layer with a p-value of 0.2 has been employed between every layer in our network to minimize the possibility of overfitting.

Sliding Window: Since the input data is in the form of a continuous motion, we need to isolate parts of the motion to detect the digits. Lack of delimiters between digits necessitates the scanning of motion from the very beginning, until the first digit is identified. Upon identification of the first digit, we record the detected digit and discard all motion data till that point. A fresh scan is started there onward and the process is repeated until the entire motion has been analyzed. We have employed a sliding window-based approach in conjunction with our dual network configuration to facilitate this process. To prevent scanning the entire motion data repeatedly, the sliding window isolates parts of the motion data for analysis, thus improving performance. The sliding window is characterized by four properties - window index, window size, split index and point threshold. Window index (w), initialized as 0, is a variable which determines the starting point of the window relative to the first motion data point. Window size (W) is a constant defining the number of consecutive data points to be analyzed at any particular instance from the entire motion data. A large window size affects performance whereas a small window size affects recognition. Empirically, an optimal value of 40 has been determined. Split index (s) is defined as the index of the point relative to the sliding window where a digit has been detected. The split index is responsible for shifting the window by modifying the value of the window index as $w' = w + s$, where w' is the new window index. In case no digit is detected in the window, the split index defaults to 5 and consequently, the window shifts by 5 data points. Point threshold (θ) is a constant which can be defined as the minimum number of data points required for the sliding window to function. For our experiments, we empirically found 10 to be the optimal point threshold value. This property defines the termination criterion as $n - w \leq \theta$,

where n is the total number of data points. A sliding window example is provided in Figure 4.

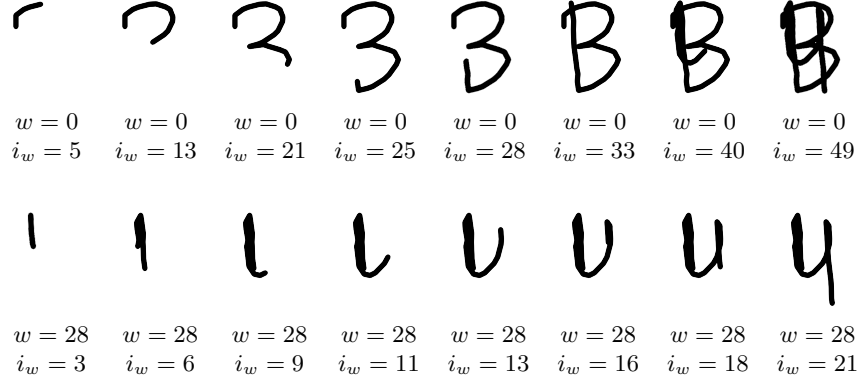


Fig. 4: Example of sliding window with a split at the 28th index, where w denotes the starting index and i_w denotes the index of the frame relative to w

Priority Scheme: In a string of digits, a digit might be a sub-shape of the next digit, like 0 is a sub-shape of 9 and 2 is a sub-shape of 3. Also, since all the digits are traced in a single continuous stroke, the transition between two digits may create an undesirable intermediate digit. For instance, if a '2' is drawn after a '6' in the air-writing environment, the resultant trace may first resemble a '6' and then an '8'. The substring '62' could thus be mispredicted as '8' since it is the final prediction, as demonstrated in Figure 5. Such scenarios may challenge how digit predictions are made in the sliding window methods.

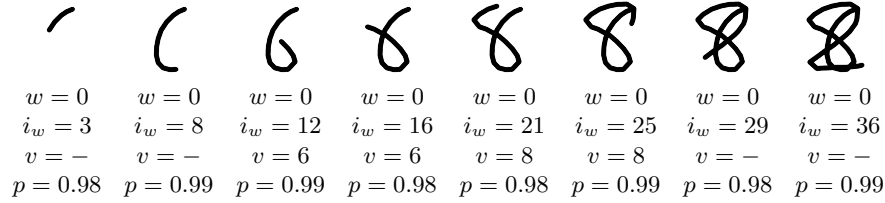


Fig. 5: 2D Trajectory traced out for the string '62' by the user, where w denotes the starting index of the window, i_w denotes the index of the frame relative to w , v denotes the predicted value, and p denotes the prediction confidence

Our approach capitalizes on the way digits are generally written. We use both our custom air-writing dataset and Pendigits dataset which contain the sequential motion coordinate information of each digit. We pass these coordinates as inputs to our sliding-window based classifier, and then obtain the possible classifications that formation of any digit can give (for example: when drawing '3', the sequential classification may first recognize '0', then '2', and then finally '3'). Such results are recorded for each and every digit in our dataset. Using a

statistical model, we extracted the possible digits that may occur before a desired digit is drawn and projected it in a *Priority Matrix (PM)*, as illustrated in Figure 6. Each element in the priority matrix can have 3 values: +1 if digit x is a sub-shape of digit y ($x \neq y$), 0 if $x = y$ and -1 otherwise. If $PM[x, y]$ for two consecutive digits is +1, then the resultant prediction gets updated from x to y . At the end of the window, the values from the classifier and the priority function are passed as the final results.

	0	1	2	3	4	5	6	7	8	9
0	0	-1	+1	+1	-1	-1	+1	-1	+1	+1
1	-1	0	-1	-1	+1	+1	+1	+1	-1	+1
2	-1	-1	0	+1	-1	-1	-1	-1	+1	-1
3	-1	-1	-1	0	-1	-1	-1	-1	-1	-1
4	-1	-1	-1	-1	0	-1	-1	-1	-1	+1
5	-1	-1	-1	-1	-1	0	-1	-1	+1	-1
6	-1	-1	-1	-1	-1	-1	0	-1	-1	-1
7	+1	-1	+1	+1	-1	-1	-1	0	+1	+1
8	-1	-1	-1	-1	-1	-1	-1	-1	0	-1
9	-1	-1	+1	+1	-1	-1	-1	-1	+1	0

Fig. 6: Priority Matrix for English Numerals

Bi-directional Scan: When digits are written in a continuous motion, transitional strokes between digits start adding up as noise. With an increase in the number of digits drawn in a single continuous motion, the noise accumulates, making it increasingly difficult to detect digits as we progress along the air-written string of digits. To improve the accuracy we employ a bi-directional scanning technique where we scan the data points in both forward and reverse directions. The results of the second half of forward scan are then substituted with the reversed results of the first half of the reverse scan. Bi-directional scanning thus employs two starting points for scanning the motion data points, allowing less noise to accumulate and reducing the error rate significantly in the process.

Pipeline: The techniques delineated above are combined in a pipeline to facilitate the numeral recognition module of our system, as illustrated in Figure 7. Spatio-temporal coordinates are taken as inputs and passed into the Frame Generator (FG) along with the default starting index (w) value. The frame generator takes in two inputs: sequential coordinates and starting index value for the sliding window, and returns frames of motion traces that takes place inside that window. The number of frames returned is equal to the size of the sliding window (W). Each generated frame is then passed into a dual-network classifier which predicts the digit that might be present in that frame. All such predictions along with their frame indices are stored in an array which we refer to as

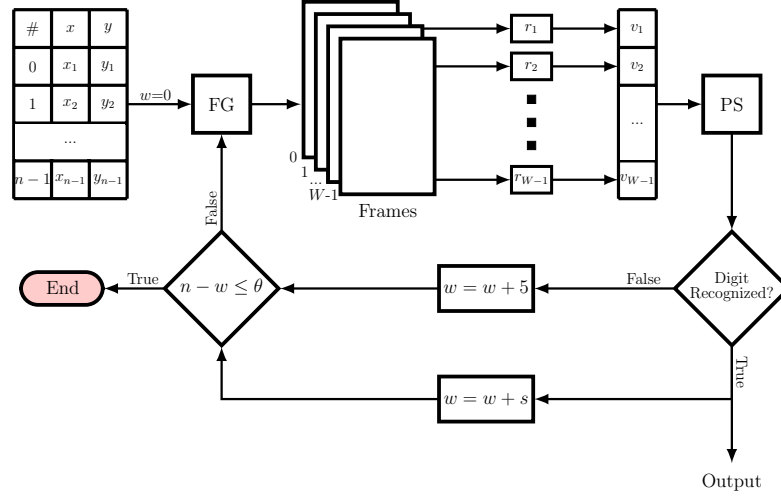


Fig. 7: Pipeline of our system. r_i and v_i denote classifier and predicted value for the i^{th} frame respectively. W denotes the size of the sliding window, w denotes the starting index of the sliding window, s is the split index, θ is the point threshold, and n is the total number of data points. FG and PS stand for Frame Generator and Priority Scheme respectively.

Prediction Pool. The prediction pool is then analyzed using the Priority Scheme (PS). If a digit is recognized, it is considered an output, and the window index is updated using the split index value. If no digit is recognized, then the window index is updated by a default value of 5. Termination criteria is checked, and if not met, the updated window index is passed on to the frame generator and the process is repeated.

4 Results & Discussion

4.1 Data Collection

To the best of our knowledge, there is no publicly available standard air-writing dataset. For this project, we developed ISI-Air, a custom English numeral air-writing dataset with 8000 samples recorded by 16 persons, each recording 50 samples per digit. The numerals were traced in three dimensional space with a uniformly colored marker and the data was recorded using a generic video camera. The marker was segmented, the marker tip was identified and its trajectory was approximated. The recorded trajectory coordinates were tagged with their numerical labels and saved. Pendigits is a standard online handwritten English numeral dataset, consisting of 5120 samples of online handwritten English numerals. However, its data samples were recorded using a digital pen on a pressure-sensitive tablet which comes with pen-up/pen-down motion. We have tweaked the Pendigits dataset to remove every pen-up and pen-down instance by connecting the coordinates before and after the pen-up and pen-down motion,

respectively, thus emulating the air-writing environment. We have also used the MNIST dataset in our experiments, given its ubiquitous usage and owing to the similarity between handwritten and air-written numerals and for exhaustive benchmarking purposes. It comprises 70000 samples of handwritten English numerals. All the aforementioned datasets are split into training and test sets. The dataset distribution, as used in our experiments, is shown in Table 1.

Table 1: Dataset distribution for English numerals

Index	Type	Source Dataset	#Sample	#Training	#Test
DS-1	Offline, Handwritten	MNIST	70000	60000	10000
DS-2	Online, Handwritten	Pendigits	5120	2560	2560
DS-3	Online, Air-Written	ISI-Air	8000	5000	3000

Noise Dataset: Noise elimination is crucial for obtaining better recognition accuracy in an air-writing environment. While certain stray motions which add to the overall noise in the system are random in nature, a majority of the noise can be attributed to the extra motion in the data while making transitions from one digit to another. To counter this issue, we used the same method as the creation of our air-writing dataset and recorded transitions between every pair of English numerals (00, 01, 02, up to 99). We then extracted the frames of each transition from the point at which the first digit was traced completely and the second digit had started to overlap the first digit, as illustrated in Figure 8. The composition and distribution of our noise dataset is shown in Table 2. Training an LSTM network with this noise dataset yielded a validation accuracy and loss of 99.16% and 0.0245, respectively.



Fig. 8: Some noise samples

Table 2: Composition and distribution of the noise dataset

Index	Type	Source	Class	#Sample	#Training	#Test
-	Noise	Air-Writing	Positive	10000	6000	4000
-	English numerals	DS-1, DS-2, DS-3	Negative	10000	6000	4000
NZ-1	Collective noise dataset	-	-	20000	12000	8000

4.2 Experimental Setup

An LSTM network was trained for noise elimination using the noise dataset (NZ-1). For numeral recognition, an evaluation set (EVAL) was made by combining

the test splits from DS-1, DS-2 and DS-3. A second LSTM network was trained and tested using various combinations of the datasets as follows:

1. Training with DS-1 and testing on EVAL
2. Training with DS-1 and DS-2 combined and testing on EVAL
3. Training with DS-1 and DS-3 combined and testing on EVAL
4. Training with DS-1, DS-2 and DS-3 combined and testing on EVAL

Given the widespread usage and the volume of the MNIST dataset (DS-1), we have maintained it as a baseline and is thus a constant in all training sets.

4.3 Result

We have tabulated the results from our experiments on set EVAL in Table 3, obtained using the aforementioned combinations of datasets for training and validation. We observe maximum validation accuracy and minimum validation loss for training set TS-D.

Table 3: Validation accuracy for various training sets

Index	Training Set	Test Set	Accuracy	Loss
TS-A	DS-1	EVAL	87.74%	0.6817
TS-B	DS-1 + DS-2	EVAL	96.72%	0.1279
TS-C	DS-1 + DS-3	EVAL	96.43%	0.1496
TS-D	DS-1 + DS-2 + DS-3	EVAL	98.07%	0.0682

The final tests on the trained models have been performed on single digits from our ISI-Air dataset and the results have been tabulated in Table 4. Here too, we observe the highest single digit recognition accuracy on training set TS-D which stands at 98.45%. Owing to the difference in evaluation methods and lack of a standardized dataset, the results of our system is not directly comparable with those of previous works. However, under comparable evaluation conditions, the results obtained for the model trained on TS-D when tested with isolated digits from ISI-Air outperforms the recognition accuracy values for isolated English numerals achieved by Roy et al. in [13] by 0.75% and that achieved by Dash et al. [7] by 6.75%.

Table 4: Test accuracy for single digit English numeral recognition

Training Set	TS-A	TS-B	TS-C	TS-D
Accuracy	65.35%	90.09%	98.20%	98.45%

For our experiments on multi-digit recognition, we have considered 100 samples of numeral strings of length 2, 3 and 4 each, and we present the results for both forward and bi-directional scanning of these in Table 5. The cumulative average accuracy achieved for multi-digit English numeral recognition is 82.89%.






Table 5: Test accuracy for multi-digit English numeral recognition

Digit Count	Test Sample	Training Set	Forward Scan Accuracy	Bi-directional Scan Accuracy
2	100	TS-A	51.50%	59.50%
		TS-B	79.00%	88.50%
		TS-C	84.00%	96.50%
		TS-D	84.50%	97.00%
3	100	TS-A	40.33%	48.00%
		TS-B	65.00%	77.33%
		TS-C	72.67%	86.67%
		TS-D	73.33%	87.67%
4	100	TS-A	31.75%	39.25%
		TS-B	55.00%	66.25%
		TS-C	63.75%	71.50%
		TS-D	64.50%	72.25%

4.4 Error Analysis

The confusion matrices derived from our experiments have been shown in Figure 9. The performance of our recognition model improves significantly when the air-writing dataset is incorporated for training. Using a combination of parts of MNIST, Pendigits and our own air-written dataset in EVAL for fine tuning purposes ensures versatility of our system in the kind of data it can handle. As can be observed from Table 4, recognition accuracy is very poor for the model that has only been trained on handwritten digits (TS-A), and improves significantly upon incorporation of Pendigits in TS-B, and further upon using ISI-Air in TS-C and TS-D. In Table 5, we observe a decrease in the recognition accuracy with increase in number of digits in a single stroke. This gradual decrease in accuracy is attributed to noise accumulation. Bi-directional scanning boosts performance by scanning the string in reverse as well to avoid too much noise accumulation by the time the last few digits in the string are reached. The focus of our work is a demonstration of identification of split points in numerals without the use of sensors. As can be observed from the confusion matrices, our framework achieves a fairly robust recognition of air-written numerals in real time. Some noise samples misclassified as digits have been presented in Figure 6.

Table 6: Some misclassified noise samples

Sample					
Actual Class	Noise	Noise	Noise	Noise	Noise
Predicted Class	5	8	7	6	4
Confidence	97.85%	97.94%	98.85%	97.12%	99.35%

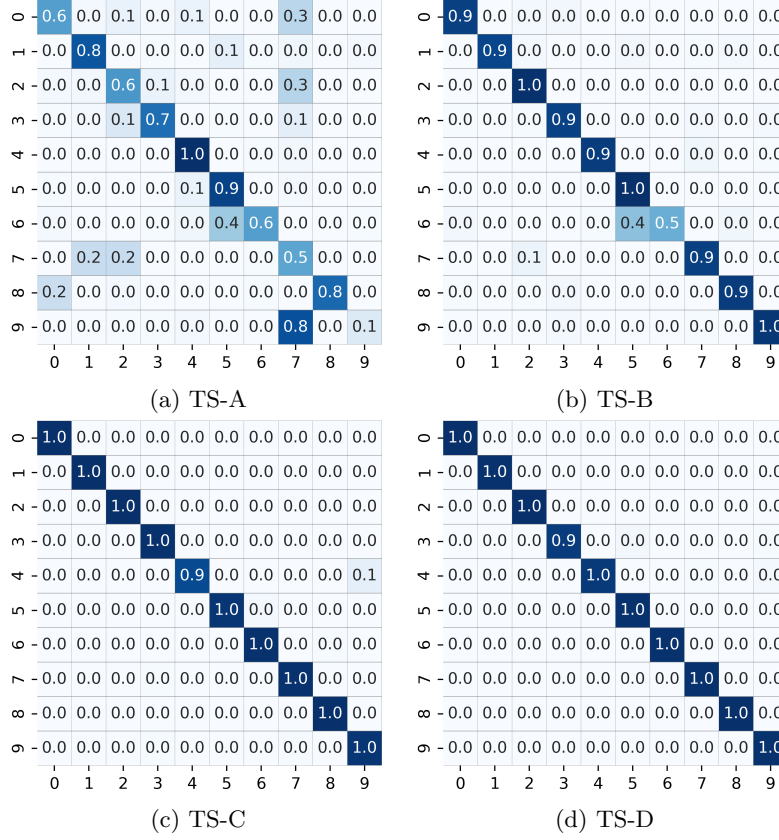


Fig. 9: Confusion matrix for various training sets evaluated on ISI-Air (DS-3) test set

5 Conclusion

In this study, a robust and hardware-independent framework is proposed for multi-digit unistroke air-written numeral recognition. While the tests are performed on English numerals, the proposed methodology is language-independent. To make the framework universally applicable, usage of language models have been avoided since they are redundant for some input forms. The priority scheme proposed in our methodology avoids recognition of any undesirable digit in a unistroke motion and resolves the issue of recognition of digits which are sub-shapes of their succeeding digits. In our experiments, the proposed methodology achieved an accuracy of 98.45% and 82.89% for the recognition of single and multi-digit numerals, respectively. The work is independent of any external motion tracking or depth sensing hardwares like Kinect and LEAP motion, making it highly accessible. Our custom online air-written English numeral dataset ISI-Air is freely available for research purposes at <https://github.com/adildsw/ISI-Air>. We are presently working on improving the accuracy for lengthier

strings and eliminating the need of dedicated noise datasets for individual languages by analyzing the sub-strokes involved in tracing numerals.

References

1. Leap Motion Inc. LEAP Motion. <https://www.leapmotion.com/> (2010)
2. Microsoft Corporation. Kinect. <https://developer.microsoft.com/en-us/windows/kinect/> (2010)
3. Amma, C., Gehrig, D., Schultz, T.: Airwriting recognition using wearable motion sensors. In: Proceedings of the 1st Augmented Human international Conference. p. 10. ACM (2010)
4. Amma, C., Georgi, M., Schultz, T.: Airwriting: Hands-free mobile text input by spotting and continuous recognition of 3d-space handwriting with inertial sensors. In: 2012 16th International Symposium on Wearable Computers. pp. 52–59. IEEE (2012)
5. Chen, M., AlRegib, G., Juang, B.H.: Air-writing recognition—part i: Modeling and recognition of characters, words, and connecting motions. *IEEE Transactions on Human-Machine Systems* **46**(3), 403–413 (2015)
6. Chen, M., AlRegib, G., Juang, B.H.: Air-writing recognition—part ii: Detection and recognition of writing activity in continuous stream of motion data. *IEEE Transactions on Human-Machine Systems* **46**(3), 436–444 (2015)
7. Dash, A., Sahu, A., Shringi, R., Gamboa, J., Afzal, M.Z., Malik, M.I., Dengel, A., Ahmed, S.: Airstcript-creating documents in air. In: 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR). vol. 1, pp. 908–913. IEEE (2017)
8. Huang, Y., Liu, X., Jin, L., Zhang, X.: Deepfinger: A cascade convolutional neuron network approach to finger key point detection in egocentric vision with mobile camera. In: 2015 IEEE International Conference on Systems, Man, and Cybernetics. pp. 2944–2949. IEEE (2015)
9. Junker, H., Amft, O., Lukowicz, P., Tröster, G.: Gesture spotting with body-worn inertial sensors to detect user activities. *Pattern Recognition* **41**(6), 2010–2024 (2008)
10. Kim, D.H., Choi, H.I., Kim, J.H.: 3d space handwriting recognition with ligature model. In: International Symposium on Ubiquitous Computing Systems. pp. 41–56. Springer (2006)
11. Lee, H.K., Kim, J.H.: An hmm-based threshold model approach for gesture recognition. *IEEE Transactions on Pattern Analysis & Machine Intelligence* (10), 961–973 (1999)
12. Oh, J.K., Cho, S.J., Bang, W.C., Chang, W., Choi, E., Yang, J., Cho, J., Kim, D.Y.: Inertial sensor based recognition of 3-d character gestures with an ensemble classifiers. In: Ninth International Workshop on Frontiers in Handwriting Recognition. pp. 112–117. IEEE (2004)
13. Roy, P., Ghosh, S., Pal, U.: A cnn based framework for unistroke numeral recognition in air-writing. In: 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR). pp. 404–409. IEEE (2018)
14. Ye, Z., Zhang, X., Jin, L., Feng, Z., Xu, S.: Finger-writing-in-the-air system using kinect sensor. In: 2013 IEEE International Conference on Multimedia and Expo Workshops (ICMEW). pp. 1–4. IEEE (2013)
15. Zhang, X., Ye, Z., Jin, L., Feng, Z., Xu, S.: A new writing experience: Finger writing in the air using a kinect sensor. *IEEE MultiMedia* **20**(4), 85–93 (2013)