Learning to Recognize Handwriting Input with Acoustic Features

HUANPU YIN, Beijing University of Posts and Telecommunications
ANFU ZHOU, Beijing University of Posts and Telecommunications
GUANGYUAN SU, Beijing University of Posts and Telecommunications
BO CHEN, Beijing University of Posts and Telecommunications
LIANG LIU, Beijing University of Posts and Telecommunications
HUADONG MA, Beijing University of Posts and Telecommunications

For mobile or wearable devices with a small touchscreen, handwriting input (instead of typing on the touchscreen) is highly desirable for efficient human-computer interaction. Previous passive acoustic-based handwriting solutions mainly focus on print-style capital input, which is inconsistent with people's daily habits and thus causes inconvenience. In this paper, we propose *WritingRecorder*, a novel universal text entry system that enables free-style lowercase handwriting recognition. *WritingRecorder* leverages the built-in microphone of the smartphones to record the handwritten sound, and then designs an adaptive segmentation method to detect letter fragments in real-time from the recorded sound. Then we design a neural network named Inception-LSTM to extract the hidden and unique acoustic pattern associated with the writing trajectory of each letter and thus classify each letter. Moreover, we adopt a word selection method based on language model, so as to recognize legislate words from all possible letter combinations. We implement *WritingRecorder* as an APP on mobile phones and conduct the extensive experimental evaluation. The results demonstrate that *WritingRecorder* works in real-time and can achieve 93.2% accuracy even for new users without collecting and training on their handwriting samples, under a series of practical scenarios.

CCS Concepts: • **Human-centered computing** → **Acoustic-based input**; **Text input**;

Additional Key Words and Phrases: Acoustic sensing, Handwriting recognition

ACM Reference Format:
Huanpu Yin, Anfu Zhou, Guangyuan Su, Bo Chen, Liang Liu, and Huadong Ma. 2020. Learning to Recognize Handwriting Input with Acoustic Features. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4, 2, Article 64 (June 2020), 26 pages. https://doi.org/10.1145/3397334

1 INTRODUCTION

Smartphone, smartwatch and other pocket-sized mobile devices bring convenience to our daily life. Due to the shrinking size of the devices, on-screen keyboard input method cannot be suitable for the small touchscreen due to the fat finger's problem [25]. To solve this problem, intelligent text entry modes such as voice input and handwriting input, have become a hot topic in the mobile human-computer interaction (HCI) area in current

Authors' addresses: Huanpu Yin, yinhuanpu@bupt.edu.cn, Beijing University of Posts and Telecommunications, No.10 Xitucheng Road, Beijing, China, 100876; Anfu Zhou, zhouanfu@bupt.edu.cn, Beijing University of Posts and Telecommunications, Beijing, China; Guangyuan Su, sgy_beijing@bupt.edu.cn, Beijing University of Posts and Telecommunications, Beijing, China; Bo Chen, gujianxiao@bupt.edu.cn, Beijing University of Posts and Telecommunications, Beijing, China; Liang Liu, liangliu@bupt.edu.cn, Beijing University of Posts and Telecommunications, Beijing, China; Huadong Ma, mhd@bupt.edu.cn, Beijing University of Posts and Telecommunications, Beijing, China.

2474-9567/2020/6-ART64 $15.00
https://doi.org/10.1145/3397334

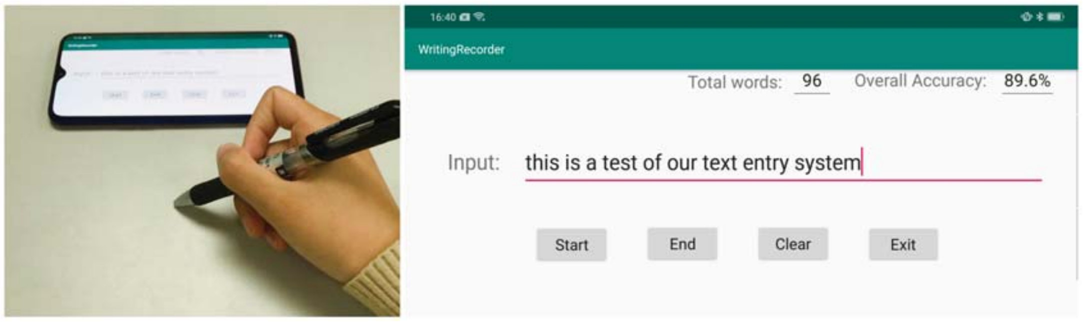


Fig. 1. *WritingRecorder*: a user writes with a pen on the desk. The nearby mobile phone records the handwritten sound through the built-in microphone, and then *WritingRecorder* calculates the written content and displays it to the user.

years. Compared with voice input, handwriting input is more convenient for many quite scenarios, like in offices. Moreover, it is more suitable for drawing and professional input like mathematical expressions. The current handwriting recognition (HWR) systems can be classified as vision-based, sensor-based and acoustic-based. Vision-based solutions [20] often require the image of handwriting, thus rely on lighting conditions and may leak privacy. Sensor-based solutions leverage sensors such as gyroscope [2, 5] to track the movement of a hand. However, they need extra hardware or specific sensors [34]. In contrast, acoustic-based methods have advantages in terms of universality, low cost, and no extra hardware requirement. Therefore, acoustic-based handwriting recognition becomes a promising direction for HWR.

Recently, some works have attempted to recognize the handwriting, according to the subtle sound generated by the pen writing on the table [25]. But these approaches have two shortcomings. *Firstly*, they mainly focus on the print-style capital letters [6, 41] and lack of the ability of free-style lowercase input, which is used more often in our daily life. *Second*, some of these methods build the user-dependent model, *i.e.* they require training a new model for each new user [10, 40]. However, a real-world system should not need to collect training samples from target users, which will degrade the user experience. Therefore, to meet our daily habit, a universal, accurate, free-style lowercase handwriting input system is urgently required.

In this paper, we propose *WritingRecorder*, which targets the universal, free-style lowercase handwriting recognition for mobile devices (see Fig.1). Specifically, *WritingRecorder* leverages a microphone to record the sound generated by pen writing on the solid surfaces and exploits the acoustic pattern hidden in handwritten sound associated with writing trajectory of the letter. We find that this acoustic pattern is robust to the environmental changes, such as the phone type, handwriting location, and far-field noise. By distinguishing the acoustic pattern among different letters, *WritingRecorder* can achieve universal and robust handwriting input on solid surfaces with high-friction in typical office scenarios.

The substantial challenges of *WritingRecorder* lie in two aspects. *(1)* Unlike capital letters with high distinguished writing trajectories, some lowercase letters have similar writing trajectories, *e.g.*, 'h' and 'n', 'a', 'c' and 'o'. The problem lies in how to capture the local time-varying characteristic hidden in acoustic signal for distinguishing trajectories. *(2)* Free-style handwriting makes it difficult to build an accurate universal model for all potential users, because different users usually have different writing habits, *e.g.*, different speeds/strengths, stroke orders and styles.

To tackle these challenges of *WritingRecorder*, we design a neural network named Inception-LSTM, which integrates the deep feature extraction (Inception module) and time-series modeling (Long Short-Term Memory module, LSTM). This network depicts the information of both the time and frequency domains in the real sense.

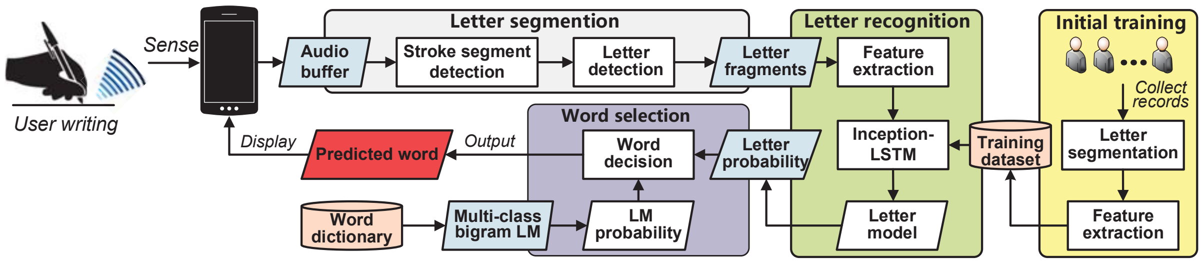


Fig. 2. The system architecture of *WritingRecorder*.

(1) Inception module is used to extract the deep local feature to solve the problem of similar trajectories. The extracted local feature is universal and independent to some specific users. In order to extract the deep local feature for each frame, we feed the Short-time power spectral density (stPSD) feature, which contains both temporal and frequency information, into the Inception module. *(2)* Meanwhile, LSTM module is used for modeling the time-series relations between frames of the acoustic signal, to build a universal model for all users. *(3)* Besides, based on the fact that our goal is to recognize words, we adopt a word selection method based on language model (LM) to further improve recognition accuracy at word-level. It builds a multi-class bi-gram LM via the dictionary to estimate the probability distribution of letters in the word, and then use the LM parameters and the result of letter recognition for word decoding.

Based on the above techniques, we implement *WritingRecorder* as a prototype application on the Android phone. It consists of three major components: letter segmentation, letter recognition, and word selection. To the best of our knowledge, this work is the first universal, acoustic-based, and free-style lowercase input system. Our contributions can be summarized as follows:

- We design a neural network called Inception-LSTM, whose Inception module describes deep local information in the spatial domain and can distinguish lowercase letters with similar trajectories.
- We build the universal model for users with different handwriting habits by adopting LSTM module in the proposed Inception-LSTM network to model time-domain variation of the signal. Besides, we also adopt an LM-based word selection algorithm to further improve the recognition accuracy at word-level.
- We implement a prototype system on the Android phone. The experimental results show that *WritingRecorder* can achieve robust, real-time handwriting recognition on solid surfaces with high friction (with a negligible delay of 89ms), which has 93.2% of word accuracy for users without training.

2 OVERVIEW

Design goal. *WritingRecorder* aims to provide a convenient text entry for pocket-sized mobile devices based on the handwritten sound. Considering its work scenario, the design goals are as follows:

(1) High accuracy. As a text entry system, it first needs to ensure high accuracy. Due to the various writing habits of different users, *WritingRecorder* needs a recognition algorithm that can accurately recognize the free-style lowercase letters with similar writing trajectories.

(2) Timeliness. WritingRecorder should provide real-time processing. When a user finishes writing, the system should give the result immediately, so as to guarantee smooth input. The response time of the system cannot exceed 100ms under normal conditions [14].

(3) Robustness. The system needs to accurately recognize words in different situations, such as the change of writing location, different noise environments.

System architecture. Fig. 2 illustrates the system architecture of *WritingRecorder*, which mainly contains three components: letter segmentation, letter recognition, and word selection.

Letter segmentation (Sec. 3). This component uses the microphone to receive the handwritten sound stream when a user writes on the table. The received sound stream will be stored in the buffer at regular intervals. After that, *WritingRecorder* performs an adaptive-threshold real-time segmentation algorithm on the buffer for stroke segment detection and then combines these segments into the letter fragments. Particularly, when a letter is detected, the letter fragment is sent to *letter recognition* (Sec.4) component immediately, and *word selection* (Sec.5) algorithm is performed after the entire word is detected.

Letter recognition (Sec.4). This component extracts the stPSD feature of the letter fragment and puts the feature into the Inception-LSTM network. The network not only extracts the local depth information of each time frame, but also describes the time-series relations between frames. The output of the network is a 26-dimensional vector that depicts the probabilities for a letter fragment to be recognized as 26 lowercase letters.

Word selection (Sec.5). The input to this component includes two parts: (1) a probability matrix which combined by the probability vectors of the contained letters (the output of Sec.4) after a word is finished; (2) the LM probability parameters of multi-class bi-gram LM built by the dictionary. *WritingRecorder* leverages them to decode the word and finally output the result and display it to the user.

Before running *WritingRecorder*, we need to collect a set of training letters samples for training the Inception-LSTM network. Note that, the network training and language model building are both the off-line process. Therefore, they do not affect the system on-line processing time.

3 LETTER SEGMENTATION

The basic idea of letter segmentation is to detect the acoustic signal of letters by using the general observed phenomenon: the handwritten sound is usually stronger than the ambient noise, due to the close distance between the mobile device and the writing position (see a case in Fig. 3(a)). Therefore, our intuition is that when the energy of signal is higher than a threshold, we regard it is a handwritten sound, otherwise it's noise. However, due to different noise levels in different environments, a fixed threshold is not feasible. Thus, we propose an adaptive threshold real-time segmentation algorithm. The algorithm consists of the following two steps: it first uses the threshold to detect the stroke segments (*stroke segment detection*) contained in the letter and then combines the segments into the letter fragment (*letter detection*).

3.1 Stroke Segment Detection

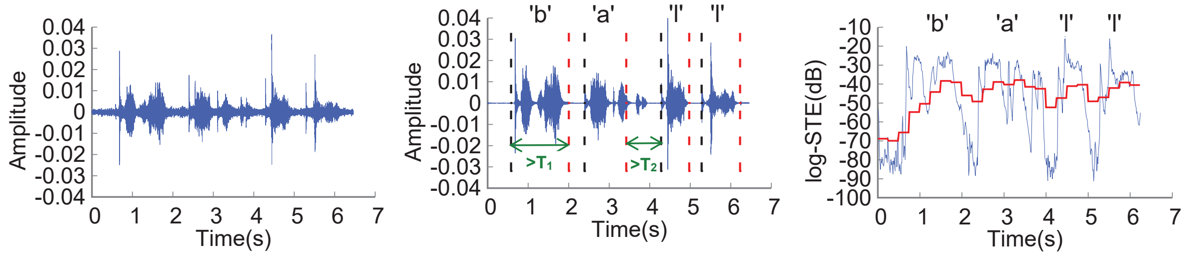
To achieve the goal of real-time detection, we define the processing window as a 0.25s audio buffer, which *WritingRecorder* receives the recorded signal periodically into it[1]. As for each processing window, we determine the adaptive-threshold by calculating the logarithmic short-time energy (log-STE) of the processing window as follows:

At first, we apply Hanning window with the size of S_1 to split the signal $x(t)$ into frames and use the Wiener filter [19] to reduce the effect of noise generated from background environment and devices [2]. The result of denoising as shown in Fig. 3(b). Then we compute the log-STE for ith frame as follows:

$$E(i) = 10\,log \sum_{n=0}^{S_1-1} x_i(n)^2 \,. \tag{1}$$

[1]The main idea of the real-time process is that, when *WritingRecorder* detects a letter fragment in the current processing window, it will recognize the letter and continues to detect the next letter simultaneously.

[2]Noise reduction can provide a more ‘pure’ handwritten sound, in preparation for extracting the better features in the next step of *letter recognition.*



(a) raw handwritten signal (b) letter segmentation after denoising (c) log-STE of signal

Fig. 3. Example for letter segmentation: word ‘ball’

Thus, the average log-STE of jth processing window is calculated as follows:

$$E_P(j) = \begin{cases} (1-\alpha)E_P(j-1) + \frac{\alpha}{P_n}\sum_{i=1}^{P_n} E(i), & j > 1 \\ \frac{1}{P_n}\sum_{i=1}^{P_n} E(i), & j = 1 \end{cases}, \tag{2}$$

where P_n is the number of frames, α is a weight constant. According to the average log-STE of jth processing window, we set the threshold to $\beta E_P(j)$.

Then, we confirm that the stroke segment exists in the jth processing window if the log-STEs of some consecutive frames in this window satisfy the condition of higher than $\beta E_P(j)$, as described in Fig. 3(c).

3.2 Letter Detection

After all the stroke segments in the current processing window are detected, we need to combine them into the letters. Different from previous segmentation methods [6, 41], our method is a real-time processing. The details of the solution are described as follows:

We first define three thresholds: T_1, T_2 and T_3. T_1 is the maximum length of a burst noise segment. T_2 denotes the time interval between the letters, and T_3 denotes the time interval between words. Obviously, T_3 is greater than T_2.

As for the current processing window, starting with the first stroke segment, we first judge that the segment is a real stroke of the letter rather than burst noise (such as the sound of closing the door or knock the table). If the length of the segment is less than T_1, it is judged as burst noise and then removed.

Then, we combine the stroke segments into current letter fragment one by one. The current letter segment is completed until one of the following two conditions is satisfied: (1) The time interval between the current stroke segment and the next one is higher than T_2. (2) The time interval between the last stroke segment and the end of current window is higher than T_2.

At last, we send the detected letter fragment into *letter recognition* component (Sec. 4) and continue to detect next stroke segment and combine it into a new letter fragment. Especially, the word is finished if the time interval between adjacent stroke fragments is greater than T_3. The segmentation result is as shown in Fig. 3(b).

4 LETTER RECOGNITION

In this section, we propose a novel letter recognition method to address the problem of letter trajectories similarity and provide a universal model for different users. The method consists of two parts: feature extraction and Inception-LSTM network.

4.1 Feature Extraction

To capture the unique local time-varying characteristic hidden in acoustic signal for distinguishing different letters, we conduct the experiment to find the suitable feature. We first ask a user to write 10 lowercase letters 'a'-'j' on the table with same writing speed and strength, each letter is written for N times. Then we extract feature and compute the distance for all letters.

The distance $d(l_i, l_j)$ between different letters l_i and l_j is calculated by the following formula:

$$d(l_i, l_j) = \frac{1}{N^2} \sum_{s=1,k=1}^{N} d(l_i(s), l_j(k)), \tag{3}$$

where $l_i(s)$ means the sth sample of letter l_i, $s, k \in [1, N]$, and $d(l_i(s), l_j(k))$ is the Euclidean distance between the features of letter sample $l_i(s)$ and $l_j(k)$. We set $N = 8$. The distance $d(l_i, l_i)$ between the same letters l_i is calculated by the following formula:

$$d(l_i, l_i) = \frac{1}{N^2 - N} \sum_{s=1,k=1}^{N} d(l_i(s), l_i(k)), s \neq k. \tag{4}$$

Furthermore, the difference between same-letter distance and different-letter distance is defined as follows:

$$dif = \frac{1}{n^2 - n} \sum_{i,j=1}^{n} d(l_i, l_j) - \frac{1}{n} \sum_{i=1}^{n} d(l_i, l_i), i \neq j, \tag{5}$$

where n is the number of letters, we set $n = 10$. *The higher dif means that the ability of distinguishing feature is stronger.*

We select the following three common features for comparison:

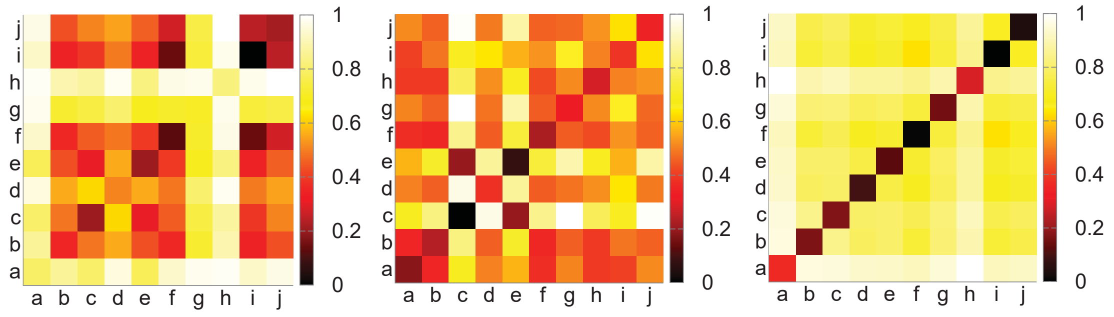
- **Amplitude spectrum density (ASD)** [44] is defined as $FFT(x(t))$, where FFT is fast fourier transformation and $x(t)$ is the amplitude of audio signal at time t.
- **Mel Frequency Cepstral Coefficient (MFCC)[15]** is a typical feature used in voice recognition, which is designed based on human ear auditory characteristics. Its frequency distribution is non-linear, and it is very sensitive to low frequency.
- **Short-time power spectral density(stPSD)** is a time-frequency feature, which divides the signal into frames, then perform FFT transformation and compute PSD for each frame.

Fig. 4 illustrates the distances among 10 letters using ASD, MFCC and stPSD features, respectively. The result shows that the performance of stPSD is better than MFCC, and ASD is the worst. This is because that the handwritten sound is a time-varying and non-stationary signal, ASD is a frequency domain feature which does not consider time-domain variations. Besides, as for both MFCC and ASD with time-varying considerations, power (stPSD) is more capable of capturing time-varying characteristic of signal than cepstral coefficients (MFCC).

Thus, we extract stPSD feature to represent the unique time-varying characteristic of handwritten sound. Specifically, we use the Hanning window for framing in which the length of the window and overlap part is set to 0.02s, 0.01s receptively, and FFT length is equal to the window size. Note that, to avoid the impact of magnitude difference caused by writing with different strengths, we first remove the DC component and apply magnitude normalization before extracting stPSD for letter signal fragment.

Besides, from Fig. 5 we can observe that the useful information of the handwritten sound is mainly concentrated below 10kHZ. Therefore, in order to reduce the calculation cost of the network, we remove the useless high-frequency part (*i.e.* higher than 10kHZ) of stPSD.

Nevertheless, in real-life environments, users have different writing habits, such as stroke order and style, as shown in Fig. 6. Even for the same user, writing speed and writing strength cannot always remain the same. We



(a) ASD, $dif = 0.265$ (b) MFCC, $dif = 0.3394$ (c) stPSD, $dif = 0.6665$

Fig. 4. Comparison of different features: distance matrix of letter 'a'-'j'. The darker the color, the smaller the distance between the two letters.

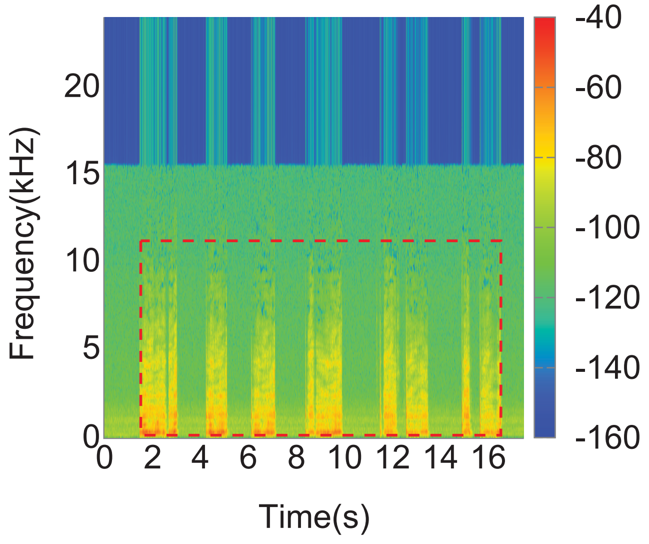


Fig. 5. The stPSD of handwritten sound.

tried to use common classification methods such as k-Nearest Neighbor (KNN), Support Vector Machine (SVM) to classify stPSD features of letters. The details of training and test dataset are described in the part of Dataset in Sec. 6.1. As shown in Tab. 1, the results are barely satisfactory. Thus, stPSD (artificial) feature is not enough, we need to explore deeper, and user-independent features to distinguish different letters.

4.2 Inception-LSTM Network

In this subsection, we design a neural network to extract deeper and user-independent feature. Our initial motivation is to adopt LSTM to solve the user-independent model problem caused by free-style handwriting. LSTM [12] is commonly used to process sequence data and is widely used in speech recognition, stock prediction,

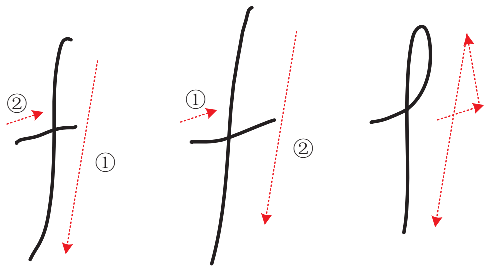


Fig. 6. The letter 'f' with different writing habits. The order of strokes of left and the middle one is different; As for middle and right one, they have two strokes and one stroke, respectively.

Table 1. The performance of letter recogntion for different classification methods.

Algorithm	SVM	KNN	Ours
Accuracy	28.65%	16.54%	73.8%

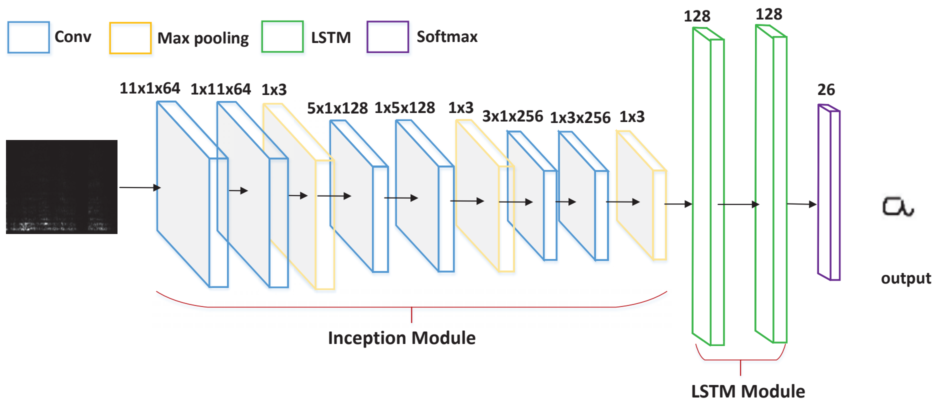


Fig. 7. The Inception-LSTM network structure for letter recognition.

handwriting recognition, *etc*. As mentioned in [24], compared with artificially defined features, feed deeper feature into the LSTM makes it easier to learn the temporal structure within time frames. Therefore, we design a neural network structure named Inception-LSTM as illustrated in Fig. 7. The network contains two modules: Inception module for deeper local feature extraction and LSTM module for learning the time-series relations between frames.

Inception module. Previous studies [3, 6] have shown that convolutional network is robust to writing strength/speed, thus we use the convolution layers of AlexNet [8] as the main structure and decompose the convolution of $n \times n$ structure into a $1 \times n$ convolution and a $n \times 1$ convolution to reduce computation [31]. The convolution kernels n have different sizes: 11, 5 and 3. And the corresponding number of kernels is 64, 128 and 256, respectively. Each convolution layer applies the rectified linear activation (Relu) function.

Different from previous work [6], we treat stPSD as a time-series feature instead of an image to extract depth information from frequency features on each time frame. In particular, Inception module extracts depth information in the frequency domain for each frame of stPSD. For this purpose, we also modify max-pooling layers in which pooling size is 1×3 with the stride of 2. Besides, in order to enhance the generalization of the model, each pooling layer is followed by a local response normalization (LRN) operation.

LSTM module. Due to the different strokes ordering caused by different people's handwriting styles, we need to find out the relationship between strokes in the temporal domain. Thus, after extracting the deep local feature for each frame, we leverage two LSTM layers in which each LSTM layer has 128 hidden cells, for modeling the time-series relations between frames of the handwritten signal. Finally, the output of LSTM module is put into the softmax layer for letter classification. The final result is a 26-dimensional vector, which represents the probabilities that the input letter signal is labeled with 26 lowercase letters.

5 WORD SELECTION

Due to free-style handwriting habit, the problem of inherently-similar writing trajectory (such as letter 'r' and 'v') is not entirely solved by the proposed Inception-LSTM network (Sec. 4). Therefore, considering that our final goal is to recognize words rather than individual letters, we propose a word-selection algorithm which uses language model to further correct recognition result at word-level, to make the result closer to an effective word. It leverages the dictionary database to build a multi-class bi-gram LM, then uses LM parameters to decode the result of letter recognition to get the word. We proceed to the details of two components.

5.1 Building Language Model

Language model is commonly used in machine translation, part of speech tagging, speech recognition [4], which describes the probability distribution of sequences of words in the sentences. Different from the above, the unit of our probability estimate is a letter rather than a word. To meet our requirements, we modify the n-gram algorithm [20] as follows.

As for a word W which contains N letters $\{g_1, g_2, ...g_N\}$, its occurrence probability is expressed as follows:

$$P(W) = \prod_{i=1}^{N} P(g_i \mid g_1, ...g_{i-1}) \approx \prod_{i=1}^{N} P(g_i \mid g_{i-(n-1)}, ...g_{i-1}), \tag{6}$$

where $P(g_i \mid g_{i-(n-1)}, ...g_{i-1})$ means the occurrence probability of the current letter depends on the previous n letters, in this paper, we set $n = 2$, *i.e.* the bi-gram model. Thus, Equ. (6) can be simplified as:

$$P(W) \approx \prod_{i=1}^{N} P(g_i \mid g_{i-1}). \tag{7}$$

Where conditional probability $P(g_i \mid g_{i-1})$ can be calculated with letter frequency:

$$P(g_i \mid g_{i-1}) = \frac{C(g_{i-1}, g_i)}{C(g_{i-1})}, \tag{8}$$

where $C(...)$ is the function which counts the number of occurrences of the letter in the words.

To count the frequency of letters in Equ. (8), we use the top-5000 words which are the commonly used vocabularies of English in the Corpus of Contemporary American English (COCA)[3] as the dictionary. According to the number of letters contained in the word, we divide words in the dictionary into multiple classes W_{set}. For example, we put these words which contain five letters into W_5, and words which contain six letters into W_6. Then, as for each class, we build the model which aims to provide more precise statistical probability distribution.

[3]COCA website: http://www.wordfrequency.info/free.asp

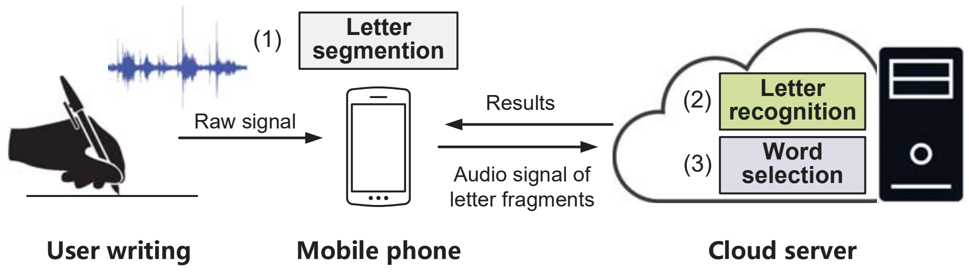


Fig. 8. The implementation of *WritingRecorder* following the app-cloud architecture.

Meanwhile, in order to prepare for the next step of word decoding, we need to calculate the probability $P(g_1)$ which means the probability of g_1 as the first letter in the word. It can be calculated by counting the number of 26 lowercase letters as the first letter in the word.

5.2 Word Decision

In this part, we discuss how to use the above probabilities for word selection in the dictionary. The main idea is as follows. At first, in order to reduce the search scope of the dictionary, we need to determine the candidate word list. The list is determined by searching all words with the same number of letters in the dictionary, according to the number of letters contained in the word got at the end of Sec. 3.

Then, we select the best match word in the candidate word list. For each candidate word W, we define the function $P_W(i)$ as the probability that the first i letters of W match successfully. The function starts at $i = 1$, the initial value is defined as follows:

$$P_W(1) = B(s) * A(1, s), \tag{9}$$

where initial probability $B(s) = P(g_1)$; A is a probability matrix with size of $N * 26$, and $A(1, s)$ denotes the probability of first letter in the test word to be the letter g_1(g_1 is the sth letter in the $L = \{a, b, ...z\}$). We get A at the end of Sec. 4 after a word is finished.

And $P_W(i)$ is based on the following recursive regulation:

$$P_W(i) = P_W(i-1) * A(i, t) * T(s, t), i > 1, \tag{10}$$

where transitions probability T is a probability matrix; $T(s, t)$ denotes the conditional probability $P(g_i \mid g_{i-1})$, where g_{i-1}, g_i is the sth, tth letter in the L.

Finally, we find the word with the highest probability P_W as our final output. Note that if there exist multiple words with the highest probability, we calculate the number of letters N_l that match the result of Sec.4 for each of these words as follows:

$$N_l = \sum_{i=1}^{N} (1 \mid g_i = \underset{l_j}{argmax}(A(i, j))). \tag{11}$$

Then we output the word with largest N_l. Algorithm 1 gives the whole algorithm flow of word selection.

6 EVALUATION

We implement a *WritingRecorder* prototype, which is a real-time application that follows the app-cloud architecture. Specifically, the *letter segmentation* component is implemented on the phone, while *letter recognition* and *word selection* are implemented on the cloud server. Fig. 8 illustrates the architecture and workflow of the prototype. The phone receives the handwritten signal in real-time and detects the letter one after one (Sec. 3). When a letter fragment is detected, the phone sends the signal of a letter to the server. The server recognizes the current letter

ALGORITHM 1: Word Selection

```
Input: $L = \{a, b, ...z\}$; $A$; test word: $W_{test}$ contains $N$ letters;
The classified dictionary sets $W_{set} = \{W_1, W_2, ...W_K\}$, where $K$ is the maximum number of letters contained
in a word in the dictionary
Output: The result: $W_{result}$
// Word decision
According to $N$, select the candidate words $W_N$
$T, B = LanguageModel(W_N, N)$
foreach $W \in W_N$ do
    $P_W(1) = B(s) * A(1, s)$,where $g_1$ is the $s$th letter in the $L$.
    for $i = 2 : N$ do
        $P_W(i) = P_W(i-1) * A(i, t) * T(s, t)$, where $g_{i-1}, g_i$ is the $s$th, $t$th letter in the $L$.
    end
end
if multiple words with maximum $P_W(N)$ exist then
    Select the word with largest $N_l$ in the these words as $W_{result}$.
else
    $W_{result}$ is the word with maximum $P_W(N)$
end
// Building language model
Function LanguageModel($W_N, N$)
    Initialize $T$,$B$ to the arrays of all zeros, with size of $(26, 26)$ and 26,respectively
    foreach $W \in W_N$ do
        $B(s) = B(s) + 1$, where $g_1$ is the $s$th letter in the $L$.
        for $i = 2 : N$ do
            $T(s, t) = T(s, t) + 1$, where $g_{i-1}, g_i$ is the $s$th, $t$th letter in the $L$.
        end
    end
    for $i = 1 : 26$ do
        $B(i) = \frac{B(i)}{\sum_{k=1}^{26} B(k)}$
        for $j = 1 : 26$ do
            $T(i, j) = \frac{T(i,j)}{\sum_{k=1}^{26} T(i,k)}$
        end
    end
return $T, B$
```

(Sec. 4) while the user writes the next letter at the same time. When a word is finished, the server performs *word selection* (Sec. 5) and gives the result word back to the phone.

6.1 Experimental Setting

Hardware platform. To evaluate *WritingRecorder*, we implement it as an APP on the *OPPO R17* with Android 8.1 OS. Meanwhile, we use a Dell desktop (Intel Core i7-6700 CPU@3.4GHz and 24GB RAM) as the cloud server. They are connected via the WiFi channel.

Parameter setting. As for the parameters in Sec. 3, we set the Hanning window length $S_1 = 0.02s$ and the overlap length is 0.01s. The parameter $\alpha = 0.3$, $\beta = 0.6$ according to our experiment result. Besides, T_1, T_2 and T_3 are set to 0.08s, 0.35s, 0.75s, respectively, according to related papers [6, 41] and our experimental observation.

As for the neural network, we use the batch size of 128 for 65 epochs. Moreover, we use Adam optimizer with a learning rate of 0.0005 to optimize the network. The loss function is the cross-entropy loss.

Dataset. We have published our dataset on GitHub[4]. Firstly, we recruit 24 volunteers (14 males and 10 females) that use the same gel pen to write the lowercase letters on the wood table. The mobile phone is placed about 15cm away from the writing position. Note that, as for all volunteers, there are no restrictions on writing speed, writing strength and phone types[5], to meet the real usage scenario. The training set consists of 3952 letter samples from 19 volunteers, each of them writes 8 times per letter. To evaluate the performance of *letter recognition*, we ask the remaining 5 volunteers to write the letters 8 times to build the test set.

Meanwhile, as for system-level evaluation, we ask 7 people with training and 5 people without training to write 50 words (including 282 letters) that randomly selected from the dictionary as the test word set. In brief, the test word set consists of 350 words from users with training, and 250 words from users without training.

Baseline Schemes. We compare *WritingRecorder* with following the three state-of-the-art acoustic-based handwriting recognition schemes:

- WordRecorder [6] focuses on capital letter recognition. It extracts normalized spectrogram of handwritten sound and saves it as an image. Then it feeds the image into a convolutional neural network (CNN) for letter classification, and finally adopts a word suggestion to recognize the word. We implement the system according to the description of [6].
- Ipanel [3]: Similar with WordRecoder, it also extracts spectrogram and feeds image into CNN for letter/gesture classification, including 10 digits, 26 lowercase letters and 7 gestures. We implement the network architecture as described in [3], by only changing the output category from 43 to 26, for letter classification. Meanwhile, in order to recognize the word, our word selection algorithm is applied.
- Pentelligence [25] extracts the normalized Hilbert envelope of the signal and uses majority voting networks for digit recognition. To recognize the lowercase letter, the network architecture is implemented similar to the only audio input configuration described in [25], but the output probability vector is 26-dimension instead of 10-dimension. Meanwhile, our word selection algorithm is applied to extend it to recognize the word.

6.2 Micro-benchmark

6.2.1 Letter Segmentation. We first evaluate the performance of letter segmentation. The test object is 600 words written by 12 volunteers as described above. We count the number of words and letters that are correctly divided, the result is shown in Fig. 9. The average segmentation accuracy of letters and words is 99.76%, 99.16%, respectively. The result shows that the performance of letter segmentation is effective.

By analyzing the failure cases, it is mainly caused by two reasons: (1) The length of burst near-field noise is too higher than T_3 (the maximum length of burst noise), it will cause false positive. According to the description in [32], we can leverage the gyroscope to solve it. (2) It consists of unavoidable two outlier situations: the interval of letters is lower than threshold T_2 (the minimum interval of letters), or the interval of burst fragments higher than T_2. The former will cause combine two letters into one segment, while the latter will cause divide a letter into two segments. Nevertheless, the threshold is 'safe' to support our segmentation method because it is based on the statistical results.

[4]We release our dataset at GitHub: https://github.com/xiaopooh/WritingRecorder.

[5]Due to different mobile types, the sampling rates of mobile phones are mostly 44100Hz and 48000Hz. For calculation convenience, we set the default sampling rate to 44100Hz. Handwritten sounds with a sampling rate of 48000 Hz will be re-sampled.

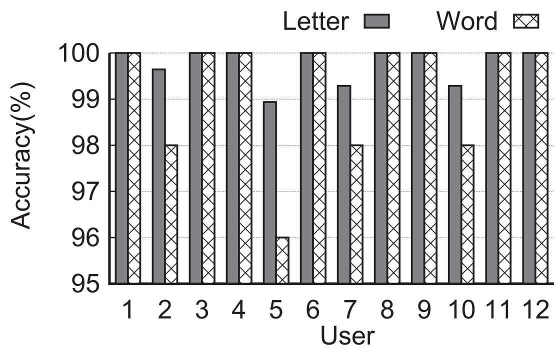


Fig. 9. The result of letter segmentation.

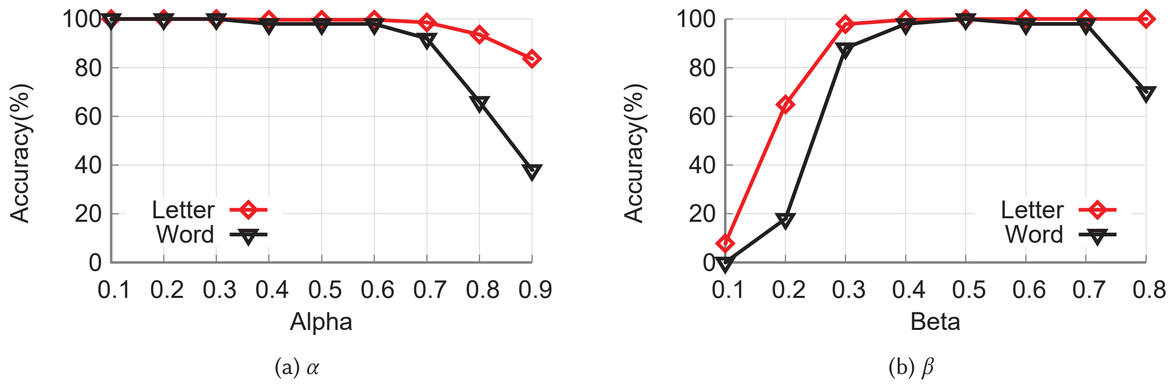


Fig. 10. The effect of parameters α, β on the accuracy of letter segmentation

The parameters of letter segmentation: In this part, we study the sensitivity of letter segmentation performance to the parameters α, β as described in Sec. 3. To test them, we perform letter segmentation on 50 words and counted the correct number of words and letters. The results as shown in Fig. 10.

As for the parameter α, we can see from Fig. 10(a) that the accuracy of words and letters decreases when α is higher than 0.6. According to Equ. (2), we can conclude that the segmentation threshold of the current window mainly depends on the log-STE of the current window, not the previous window. Besides, the accuracy of words drops drastically than letters. This is because that as for a word, it contains multiple letters. Once one letter segmented fails, the whole word is judged as segmented fail. Therefore, the correct rate of the letters is higher than words.

As for the parameter β, we can see from Fig. 10(b) that $\alpha = 0.5$, the performance is the best. When α is set too high (beyond 0.7), the accuracy of words will decrease. Meanwhile, an interesting phenomenon is the letter accuracy still remains stable. By analyzing the phenomenon, we find that a larger α makes a smaller segmentation threshold because the log-STE is a negative value. And a smaller threshold will cause the burst ambient is mistakenly judged as the letter fragment, *i.e.*, false positive. Meanwhile, all letters are correctly segmented.

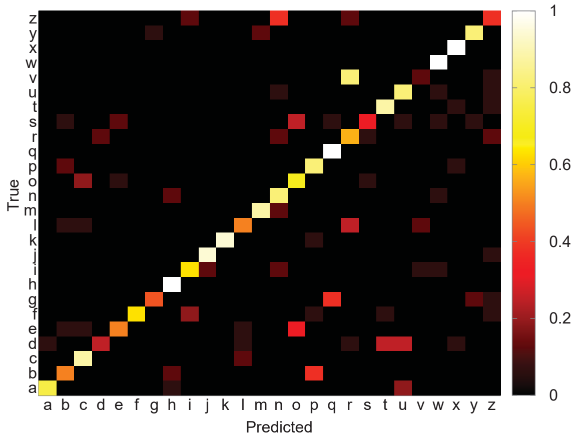


Fig. 11. Confusion matrix of letter recognition.

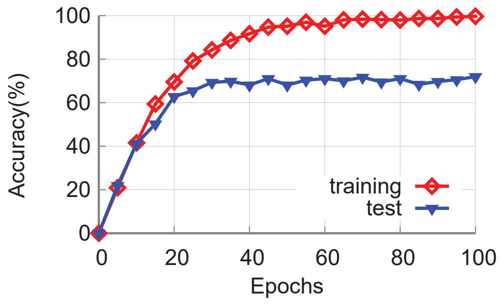


Fig. 12. The effect of training epochs.

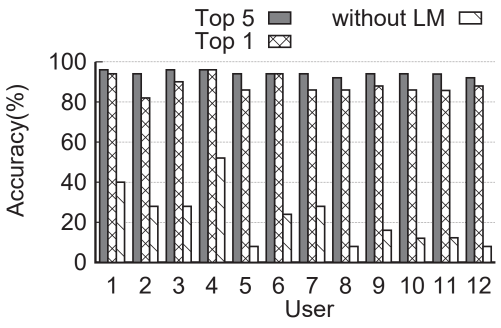


Fig. 13. The result of word recognition.

We also observe that when α is below than 0.4, the segmentation performances have dropped drastically. The main reason is that a letter fragment detected by the higher threshold is much shorter than the actual length. So that the letter fragment will be discarded as burst ambient noise, *i.e.*, false negative.

6.2.2 Letter Recognition. We investigate the letter recognition module in this subsection. In this experiment, we mainly focus on the user without training. We choose the letter with the highest probability as the result of the letter recognition, according to the 26-dimensional probability vector got at the end of Sec. 4.

Fig. 11 illustrates that the average accuracy of 26 lowercase letters is about 73.8%. The accuracy of 5 letters such as ‘f’, ‘x’ reaches up to 100%, while the accuracy of ‘v’ is only 18%. We analyze the result and find that 75% probability of ‘v’ is recognized as ‘r’, far exceeding the correct probability. The main reason is possible that users’ free-style handwriting habits lead to the indistinguishable writing trajectory of the two letters. Therefore, it is necessary to further enhance the recognition performance at word-level via language model.

Determining parameters of Inception-LSTM: The training epochs, learning rates of the Inception-LSTM network will affect the classification performance, in which the training epoch is one of the most important factors. We investigate the impact of it by comparing the recognition accuracy of training and testing under various setting of the epoch number. The result is shown in Fig. 12, which demonstrates that the accuracy of

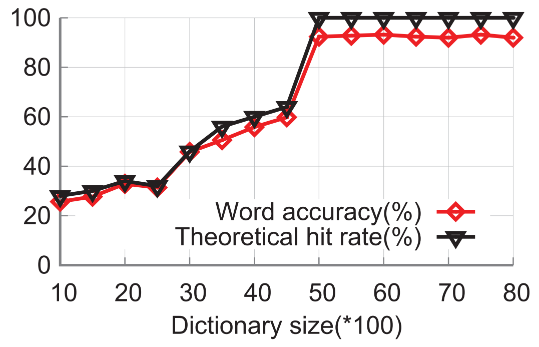


Fig. 14. The effect of dictionary sizes.

training and test increases as the epoch increases. In particular, the training accuracy gradually converges after 65 training epochs, while the test accuracy decreases slightly, since too much training epochs will cause over-fitting. Therefore, we set the default epochs to 65. In addition, we also conduct experiments on various learning rates and optimizer combinations. Experiments show that the learning rate of 0.0005 and Adam optimizer leads to the most accurate recognition.

6.2.3 Word Selection. Finally, we test the performance of this component by comparing the word accuracy with or without our word selection algorithm. As for our method, we give top-1 accuracy and top-5 accuracy. The results of 12 volunteers are shown in Fig. 13.

As for word recognition, we can observe that the average top-1 and top-5 accuracy is 88.5%, 94.17%, respectively. Thus, we decided to display five candidate words for the user to choose, which is also the typical humanization design of modern input methods. Moreover, as for top-5 accuracy, the average accuracy of users with training (user 1 and user 7) is 94.86%, and the average accuracy of users without training (user 8 to user 12) is 93.2%. It demonstrates that *WritingRecorder* is robust to different users.

Besides, the average accuracy of directly letter combination (without our word selection algorithm) is only 22%. It illustrates that the word selection algorithm can improve the recognition performance obviously under the condition of lower raw word accuracy.

6.3 Impact Factors

In this section, we evaluate the impact of different related factors that will influence the system performance, including dictionary size, noise level, phone type and location, writing distance, writing tool and material. We let 5 users without training described in the part of Dataset in the Sec. 6.1 to write using *WritingRecorder*, and calculate the accuracy under each impact factor, respectively.

6.3.1 Dictionary Size. The dictionary size has an impact on the recognition accuracy and hit rate (*i.e.* the probability of test words appearing in the dictionary). Specifically, if the dictionary is too small, it is more likely that the handwritten words do not exist in the dictionary, that is, the hit rate is too low, which directly leads to the recognition failure. Thus, in order to investigate the impact of dictionary size, we use the top-8000 words in the COCA as the dictionary, and conduct experiments with different dictionary size from 1000 to 8000 with step 500, where the dictionary with size 5000 is the baseline. On the one hand, as for each dictionary which the size is less than 5000, we randomly select words of dictionary size from top-5000 dictionary as the dictionary. For example, we randomly select 1000 words from top-5000 in the dictionary as the dictionary with size 1000. On the

Table 2. The effects of noise levels.

Noise	45dB	50dB	55dB	60dB	65dB
Accuracy	92.8%	86%	84%	86%	74.4%

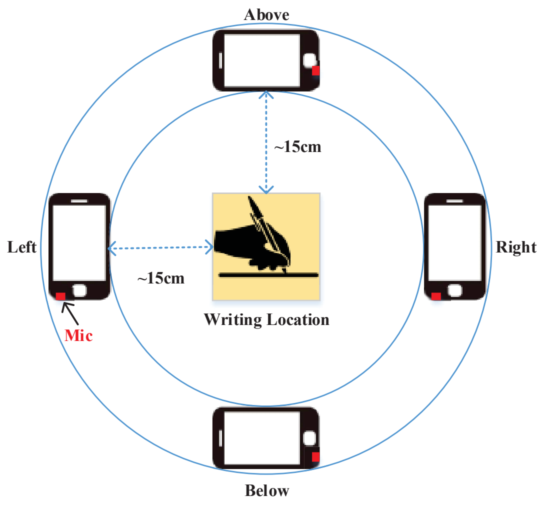


Fig. 15. Different phone locations.

other hand, as for each dictionary which the size is not less than 5000, we select top N (N is the dictionary size) words in the top-8000 dictionary as the dictionary with size N. For example, we select top 6000 words in the top-8000 dictionary as the dictionary with size 6000. Note that, due to these dictionaries contains top 5000 words, thus their hit rate is 100%.

The average word accuracy among different dictionary size is shown in Fig. 14. Besides, we also plot the theoretical hit rate. We can see that the word accuracy increase with the increase of dictionary size when the dictionary size is lower than 5000. This is intuitive: the larger size of the dictionary, the higher the hit rate and the higher the recognition accuracy. This can be verified from the theoretical hit ratio. Moreover, when the dictionary size is greater than 5000, the recognition accuracy remains stable as the dictionary increases. Therefore, we can conclude that when the size of the dictionary is large enough to cover most of the daily words, collecting more words (such as low-frequency used words) does not help with recognition accuracy.

Indeed, the number of similar words increases as the dictionary size grows. But our neural network (Sec. 4) can effectively distinguish different letters in similar words. For example, as for 'some' and 'come', 0% probability of 'c' is recognized as 's' and 0% probability of 's' is recognized as 'c', according to the result of Fig. 11. Thus, we can distinguish these two words accurately. Moreover, our word selection(Sec. 5) can further magnify the differences of similar words with different lengths. For example, 'similar' and 'familiar' are very similar, but they have the different number of letters, which facilitate word differentiation.

6.3.2 Environment Noise. In this part, we evaluate the performance of *WritingRecorder* under the environments with different noise levels. We play the voice babble from the NoiseX-92 library[6] via a laptop in the quiet office.

[6]http://www.speech.cs.cmu.edu/comp.speech/Section1/Data/noisex.html

Table 3. The effect of phone types.

User	1	2	3	4	5
Phone type	HuaWei honor v9	iPhone 6	OnePlus 3	MeiZu pro 6 plus	iPhone 8
Sampling rate(kHz)	48	44.1	48	44.1	48
Accuracy	91.6%	92.8%	92%	93.65%	93.2%

Table 4. The effect of phone locations.

Location	Above	Below	Left	Right
Accuracy	92.8%	94%	92%	94.8%

Table 5. The effect of writing distances.

Distance	20cm	40cm	60cm	80cm	100cm
Accuracy	94.8%	90.8%	86.4%	85.2%	82.8%

The noise level range from 45dB to 65dB in steps of 5dB and we measure it by Sound Meter application. Tab. 2 presents the word accuracy for different noise levels. We can observe that the accuracy is slightly reduced when the noise level is less than 65dB, but the accuracy decreases significantly when the noise level is 65dB. The main reason is that the ambient noise masks most of the handwritten sound when the noise level is 65dB. It will lead to recognition performance is highly degraded. Note that, the noise level of a noisy street scene is usually 65dB. Therefore, we can conclude that *WritingRecorder* is robust in daily office scenes.

6.3.3 Phone Type. The mobile phone has some factors that may affect the performance of *WritingRecorder*, such as the sampling rate, microphone quality, *etc.* To invest the effect of phone type, we ask volunteers to write words using different phones. 50 words are written per user for testing. To avoid the effect of the sampling rate, we use a uniform sampling rate of 44.1 kHz. As shown in Tab. 3, the results show the consistent high accuracy (>90%) of all type of phones. The result demonstrates that *WritingRecorder* can leverage Inception module to extract the in-depth information of written sound, regardless of phone types.

6.3.4 Phone Location. Previous acoustic input-based methods are very sensitive to multi-path effects [32, 44]. Therefore, in this part, we explore the effect of change of writing position on recognition accuracy. In this experiment, we ask volunteers to write the word when the phone is placed on the top, below, left and right of handwriting position, respectively, as shown in Fig. 15. In each position, 50 words are written per user for testing. As depicted in Tab. 4, the result shows that the average accuracy is 93.4%. This is because that *WritingRecorder* extract the feature to exploit acoustic pattern hidden in handwriting trajectory. On the one hand, the handwriting trajectory is independent on the mobile phone's position; On the other hand, the extracted feature is not an amplitude-related feature [41] which is sensitive to multi-path. Therefore, we can conclude that *WritingRecorder* is robust to location variation.

Table 6. The effects of writing conditions.

(a) Writing tools

Pen	Gen pen	Pencil	Ball pen	Hard Paper	Finger
Accuracy	92.8%	90%	92.4%	90%	54%

(b) Surface materials

Material	Wood	Iron	Plastic	Marble	Cardboard	Glass
Accuracy	92.8%	88%	84%	68.8%	66.4%	15.2%

6.3.5 Writing Distance. In this part, we explore the effect of the distance between the handwriting position and the mobile phone. In this experiment, we ask volunteers to write the word when the distance is increased from 20cm to 100cm in steps of 20cm. As for each distance level, 50 words are written per user for testing. The result in Tab. 5 shows that the recognition performance decrease slightly with the increase of distance, but still maintain at least 80% accuracy. This is because the writing sound is transmitted through the surface, which has a lower attenuation rate. Note that, due to the purpose of *WritingRecorder* is text input rather than eavesdropping [41], so the distance between the phone and the handwriting position does not need to be too far. In this paper, we set the distance to about 15cm.

6.3.6 Writing Tool and Surface. We investigate the word recognition accuracy under different writing conditions: writing tools and surface materials. Writing tools include mechanical pencil, ball pen, gel pen, pen made of hard paper and finger. And surface materials include the wood table, iron pad, plastic pad, marble, glass and cardboard. The basic line is to use the gel pen on a wood table. In each writing condition, 50 words are written per user for testing.

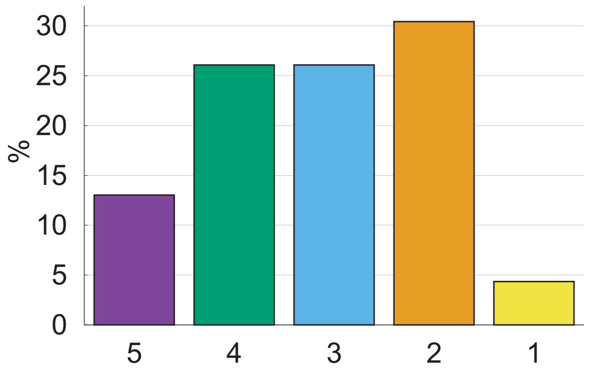
Tab. 6 (a) gives the results of different writing tools, which show that the writing tools have a slight influence on the performance of *WritingRecorder* except for the fingers. This is because compared with other writing tools, the sound produced by finger writing on the table is relatively weak, resulting in low accuracy of word recognition. Therefore, we do not recommend writing with one's finger.

Tab. 6 (b) gives the results of different surface materials, which show that wood has the highest accuracy of 92.8%, followed by iron, plastic, marble and cardboard. The glass surface leads to the worst recognizing of 15.2%. The results show that high-friction solid surfaces (*e.g.* wood, iron, plastic) perform better than smooth solid surfaces (*e.g.* marble and glass) and non-solid surface (*e.g.* cardboard). We note that recognition accuracy relies on the strength of handwritten sounds produced by different materials. Compared with the high-friction solid surface, the smooth solid surface produces less friction and thus generates a weak acoustic signal (*i.e.* marble) or even almost no sound (*i.e.* glass). For non-solid surfaces, the signal strength is even weaker.

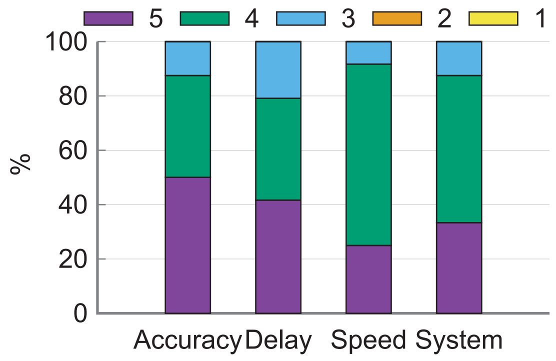
6.4 System Evaluation

6.4.1 Compare with Other Methods. At first, we compare *WritingRecorder* with Ipanel, WordRecorder and Pentelligence at letter-level and word-level recognition. Fig. 17 shows the comparison results. As for users with or without training data, *WritingRecorder* has the highest letter and word accuracy than Ipanel and WordRecorder, Pentelligence is the worst.

Two main reasons are as follows: (1) The input feature. As for Pentelligence, Ipanel and WordRecorder, the envelope extracted by the former is a time-domain feature that only denotes the trend of the signal over time, but

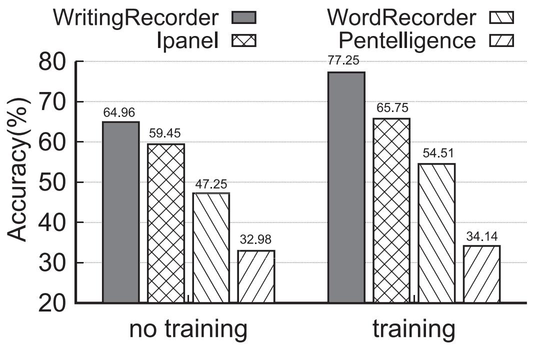


(a) User acceptance of handwriting on the touchscreen

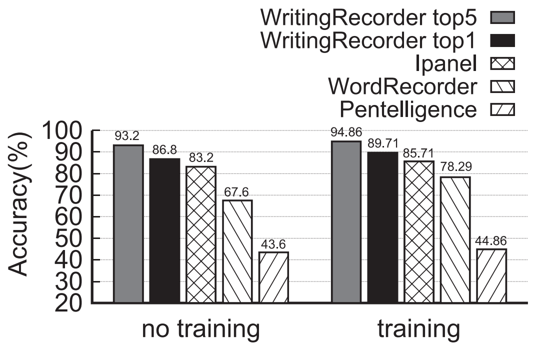


(b) User satisfaction of *WritingRecorder*

Fig. 16. User survey of *WritingRecorder*. The scores range from 1 to 5, where 5 is the highest and 1 is the lowest.



(a) Letter accuracy



(b) Word accuracy

Fig. 17. Recognition accuracy comparison among *WritingRecorder*, WordRecorder and Pentelligence.

STDFT extracted by latter two is a time-frequency feature which describes the information of both the spatial and temporal domains. (2) The network architecture. As for *WritingRecorder*, Ipanel and WordRecorder, the input features are both time-frequency features. Ipanel and WordRecorder treat the feature as a picture and CNN can only extract depth information, and does not consider the time-varying. In contrast, *WritingRecorder* adopts the Inception-LSTM network, in which the former module of it extracts the local depth information of time frame and its latter module models the relationship of time frames. As a result, it achieves higher recognition accuracy. (3) Besides, as for WordRecorder and Ipanel, although the features and network architecture of the two are similar, the accuracy of IPanel is higher because the former extracts gray-scale images while the latter extracts RGB images, which contain more information.

6.4.2 Compare with Handwriting on the Screen. To compare with handwriting directly on the screen of mobile devices, we use the Google Handwriting Input app to write 500 words directly on the screen of a smartphone, which achieves a top-1 accuracy of 94.8%. We found that the accuracy is much better than *WritingRecorder* with top-1 selection (86.8%), but is comparable to *WritingRecorder* with top-5 selection (93.2%).

Besides, according to our evaluation of 20 users (each user writes for 10 minutes while using *WritingRecorder* and mobile's screen, respectively), the average speeds of *WritingRecorder* and on-screen writing are 0.85 letters per second (10.42 WPM or words-per-minute) and 0.95 letters per second (11.12 WPM), respectively. Overall, we can conclude that compared with on-screen handwriting, *WritingRecorder* gains display space without much sacrifice on writing accuracy and speed.

6.4.3 User Survey. We conduct a user survey to investigate the users' satisfaction of the system. The user survey is evaluated in terms of two types: *(i) pre-design survey*: whether users are inconvenient to handwriting on the touchscreen of the phone, and what scenarios would it be inconvenient. *(ii) user study survey* including five aspects: ① recognition accuracy: whether users are satisfied with the recognition accuracy; ② response delay: whether users are satisfied with the system delay; ③ input speed: whether users are satisfied with the writing speed; ④ system acceptability: whether users accept *WritingRecorder* as an input mode of the mobile phone; ⑤ compared with writing on the touchscreen, what scenarios is better suitable for *WritingRecorder*. In particular, we recruit 24 volunteers (including 15 males and 9 females) to use *WritingRecorder* and then score their satisfaction in five levels: very satisfied/convenient (5), satisfied/convenient (4), neutral (3), unsatisfied/inconvenient (2), very unsatisfied/inconvenient (1). We summarize the results of all volunteers in Fig.16.

Fig.16 (a) shows that only 39.13% of users are fully satisfied with writing on the screen of the mobile phone. The inconvenience is mainly due to the following factors: small size of the touchscreen; writing long words; the 'fat' finger; the finger is dirty, *etc.* Fig.16 (b) shows that more than 87% of the volunteers reported that the recognition accuracy, input speed , and system acceptability of *WritingRecorder* met their expectations. 79.16% of the volunteers are satisfied with the current response delay of *WritingRecorder*. Therefore, we plan further to reduce input latency to satisfy the user experience in the next work.

Besides, most users confirmed that rather than writing on the screen, *WritingRecorder* is more suitable for the following scenarios: mobile devices with small screens; when writing long words; scratches on the screen; the whole screen is used for other applications like full-screen video gaming; the aged, *etc.*

6.4.4 System Delay. As mention above, the aim of *WritingRecorder* is real-time text entry. Therefore, we measure the system delay, which is described as the time difference between the user completing the input and *WritingRecorder* displaying the results on the phone. In detail, the system delay mainly consists of three parts: the last letter segmentation and recognition time, and the word recognition time. The main reason is that while the user writes the next letter, the system detects and recognizes the previous letter, and the processing time of letter segmentation and recognition is much less than the letter handwriting time. Thus, when the last letter is written, the previous letters have been detected and recognized. We count the recognition response time of 50 words, and the results show that the average processing delay is 89ms. We can conclude that *WritingRecorder* has no lagging effects for user input [14].

6.4.5 Power Consumption. In this part, we use Android's energy analysis tool to evaluate the power consumption of *WritingRecorder*. In order to comparative analysis, we use a HuaWei P30 Pro phone with Android 10 for running the following three applications, respectively: (1) sound recording with the phone's recorder. (2) browsing the website via WiFi; (3) running *WritingRecorder* and handwriting. Each situation The results show that the average power consumption per minute of these three applications is about 5.16mAh, 7.64mAh, and 6.86mAh. Each case lasts 10 minutes. We can see that the power consumption of the system is quite low, just between the two other applications. The reason lies in that most of the power consumption of the system is consumed in recording, and the rest is spent on transmitting acoustic information to the web server via WiFi.

7 DISCUSSION

Dual-microphone smartphone. Currently, many smartphones have two microphones. As for a typical dual-microphone phone, the front bottom microphone is primary and the back top microphone is secondary. In *WritingRecorder*, we select the first channel (the main microphone) signal of handwritten sound for processing, if running on a dual-microphone smartphone. Using the secondary microphone to suppress environmental noise and enhance recognition ability in a noisy environment is left for future study.

Possible ways to improve accuracy. At present *WritingRecorder* can achieve about 93.18% accuracy for users with or without training, however, it still needs further improve accuracy. To this end, we plan to start from the following aspects.

Firstly, we plan to leverage the connectionist temporal classification (CTC) to recognize word directly, without the step of letter segmentation and letter recognition. In the field of speech recognition, CTC is commonly used to predict an entire time series. It can effectively handle the outlier cases of letter segmentation that the interval of strokes higher than the interval of letters.

Secondly, our goal is to recognize a single word. In the future, we will leverage sequence-to-sequence learning [30] and other natural language processing techniques to translate sentence based on sentence semantics, to further improve accuracy.

Thirdly, in this paper, we randomly select test words with equal probability. But in practical scenarios, most of the words we use are fixed, that is, the occurrence probabilities of words are different. Therefore, we need to consider the frequency of words in the calculation. For example, when the probability of 'but' and 'bot' is approximate, we should choose 'but' according to the word frequency. To verify our hypothesis, we re-compute the accuracy by multiplying each word's occurring frequency from COCA. The result shows the top-1 weighted accuracy is 88%, a little higher than the original accuracy, while the top-5 accuracy of both is nearly the same.

Fourthly, according to the result of Pentelligence [25], motion information together with audio achieves higher recognition performance than audio alone. Therefore, we will combine *WritingRecorder* with the motion information of handwriting based on the acoustic tracking [43].

Finally, we will exploit user feedback for constant improvement automatically. For instance, each time a user replaces an erroneous word with *WritingRecorder*'s prediction, we record the replacement and add it to refresh the training set. In this way, we update the Inception-LSTM network regularly to make it more and more accurate as time goes by.

Extend to other characters and scenarios. We mainly describe the English word-level recognition in this paper. In the future, we can also extend to recognize other languages, such as Chinese. Due to Chinese characters are composed of multiple strokes, thus need to consider integrating more complex language models such as Hidden Markov Model to recognize them.

Also, non-word recognition without semantic information such as password eavesdropping, is an open question worth studying. Moreover, we also use the fact that different users have different handwriting styles for user authentication and identification.

Full implementation on the mobile phone. At present, we implement *WritingRecorder* following an app-cloud architecture. We tried to fully implement *WritingRecorder* on the phone. We used a Tensorflow framework for training the network model on the laptop and transplanted it to the mobile phone. We counted the recognition response time of 50 words, and the results showed that the average processing delay is 685ms, which could not meet the real-time requirements.

In the future, we plan to fully implement *WritingRecorder* on the smartphone without requiring the additional server, to avoid the problem of unstable network connection or even non-network situation. To meet the real-time processing, we plan to adopt Caffe framework to replace Tensorflow, according to the conclusions of [6, 27] that the processing time of Caffe is far below Tensorflow. We will also leverage GPU and NPU (Neural Processing Unit) on the phone to accelerate the calculation.

Besides, in a real scenario, the mobile input method would correct typing errors or characters that are not entered at all [1]. However, *WritingRecorder* requires the entire word must be written (*i.e.*, without using the prediction ability) currently. Thus, we will add the auto-corrector for future mobile deployments.

8 RELATED WORKS

In this section, we discuss the existing handwriting recognition works, which can be roughly grouped into three categories:

8.1 Vision-based Handwriting Methods

Vision-based handwriting methods [18, 20, 35] usually recognize the image of handwriting. These solutions often use the artificially defined feature to learn character (or a whole word) classifiers via the machine learning model, such as support vector machine (SVM) and CNN. Then they leverage LM or dictionary to correct the predicted result, which can make the result closer to the effective natural language. However, these methods require the user's handwriting picture, which is not conducive to real-time and long-term handwriting situations. Meanwhile, they are unsuitable for poor lighting conditions and may violate human privacy. *Compare with these methods, WritingRecorder is a real-time system and avoids the problem of leakage privacy.*

8.2 Sensors-based Handwriting Methods

Sensor-based solutions capture the motion information of handwriting to recognize handwriting [2, 5, 34, 37]. For example, Chen *et al.* propose a hybrid air-writing recognition system based on six-degree-of-freedom hand motion tracking, but they use the extended special hardware [2]. Besides, paper [5, 37] utilize built-in accelerometer and gyroscope sensors to recognize characters, but they require the user to hold a smartphone and smartwatch. Meanwhile, Ubitouch provides an extended virtual touchpad for smartphones, which sense the user's finger movement through proximity and ambient light sensors [34]. However, these specific sensors are not commonly embedded in mobile devices. *Compare with these methods, WritingRecorder is an acoustic-based solution, which has the advantage of universality and without extra hardware.*

8.3 Acoustic-based Handwriting Methods

In recent years, researchers have turned their attention to acoustic sensing because of universality and low cost. The most widely studied methods can be divided into tracking-based and scratch-based.

Tracking-based methods. These methods actively send the acoustic signal and track the motion of the user's hand according to reflection signal, and use MyScript and other extra handwriting recognition tools to recognize characters [45]. For example, CAT [13] integrates distributed Frequency Modulated Continuous Waveform (FMCW) and the Doppler shift for distance estimation at mm-level accuracy, but it requires multiple external speakers. FingerIO [17] and BatTracker [47] track the motion object by calculating the cross-correlation of echo. The former measures the change of two consecutive frames of echo and the latter establishes the echo-object association. LLAP [33] and Strata [43] measure the phase shift of the received signal to estimate the moving distance. *Compare with these methods, WritingRecorder is a passive acoustic-based method, does not need additional devices and is robust to the movement of surrounding objects.*

Scratch-based methods. Similar to *WritingRecorder*, these methods use the microphone to record the scratches generated by handwriting on the table for handwriting recognition.

To recognize capital letters, Li *et al.* [10] apply the template matching which calculates the dynamic time warping (DTW) distance between Mel-frequency cepstral coefficients (MFCC) of handwritten sounds. However, it is a user-dependent model, *i.e.* each user is required to provide the samples before running the recognition algorithm. Recently, WritingHacker [41] collects numerous training samples from different persons and uses letter clustering and dictionary filtering for eavesdropping victim's handwriting information. It only achieves 50%-60% of word accuracy, because MFCC cannot characterize the unique information of the handwritten sound of the same letter samples from different persons. To solve this problem, WordRecorder [6] designs a deep neural network, which extracts the depth information to build a user-independent model. Even though WordRecorder extracts the short-time feature, but it feeds the feature as an image into the neural network. This method does not consider the time-varying of the feature, thus the accuracy of the letters with high discrimination signal such as 'A' and 'M' is not high. The above methods ask for upper-cased input and specify the ordering of strokes. They are unsuitable for most daily situations which require free-style lowercase input.

As for the lowercase letter, Seniuk *et al.* [26] use three template matching schemes to identify the cursive alphabet and limited 26 words for a single user.

Besides, some methods explore the recognition of other characters. For example, Zhang *et al.* [44] extract amplitude spectrum density (ASD) of the received acoustic signal and run the K-Nearest Neighbor (KNN) to recognize strokes. However, this approach is highly dependent on location, *i.e.* once the position of mobile devices or handwriting changes, the system needs to be retrained. In [25], the authors combine writing sound and pen motion information obtained from the embedded-in microphone and the inertial measurement unit (IMU) of the digital pen, feed the data to vote neural networks for digit recognition. Besides, IPanel [3] leverage CNN to recognize 46 different characters, and restricts users to write with a stop between characters, *i.e.* they cannot be joined-up. It does not meet our daily writing habits. *In this paper, we focus on a user-independent model for the free-style handwriting lowercase recognition at word-level.*

8.4 Acoustic Sensing with Mobile Devices

Acoustic sensing combine with mobile devices has enabled varied innovative applications in many areas, such as driving safety [9, 36, 38, 39], health monitoring [7, 16, 21, 22], tracking and localization [23, 28, 29, 42, 46]. For example, Xu *et al.* use the microphone of smartphone to send the acoustic signal, to explore the relationship between different types of driver behavior and Doppler profile of acoustic signals. Acousticcardiogram[21] translates minor chest movement into phase change of FMCW signal, to evaluate the heart rate and extract the heartbeat. These methods require the mobile device to actively emit ultrasonic waves for sensing, but it still be heard by pets and long-term ultrasonic sound will make people feel unwell [3].

Additionally, acoustic sensing could be used for text-entry [32]. It locates keystrokes by classifying the multi-path signatures of recorded keystroke sound. Similarly, some keystroke eavesdrop approaches also passive record the keystroke sound and distinguish keystrokes based time difference of arrival measurements (TDOA) [11, 48]. But these methods are sensitive to the position change of phone.

9 CONCLUSION

In this paper, we propose *WritingRecorder*, a universal text entry system based on acoustic sensing and deep learning. It can recognize the free-style lowercase handwriting word for mobile phones. We propose a network named Inception-LSTM to solve the problem of lowercase letters with similar trajectories and build a universal model for different users. Inception-LSTM not only extracts the deep local feature, but also learns the relationship between time frames. Besides, in order to further improve recognition accuracy, we also design a word selection method based on the multi-class bi-gram LM. The experiment results show that *WritingRecorder* can achieve 93.2% accuracy for users without training under a range of phone locations, users, *etc.* In future work, we plan to further improve recognition accuracy and provide a better user experience.

ACKNOWLEDGMENTS

We sincerely thank reviewers for their insightful comments and suggestions that helped improve the paper. We also thank volunteers in the user study. The work is supported by National Key R&D Program of China (2019YFB2102202), NSFC (61722201, 61720106007, 61832010), the Funds for Creative Research Groups of China (61921003), the 111 Project (B18008), the Fundamental Research Funds for the Central Universities (2019XD-A13).

REFERENCES

[1] Nikola Banovic, Ticha Sethapakdi, Yasasvi Hari, Anind K. Dey, and Jennifer Mankoff. 2019. The Limits of Expert Text Entry Speed on Mobile Keyboards with Autocorrect. In *Proceedings of the 21st International Conference on Human-Computer Interaction with Mobile Devices and Services* (Taipei, Taiwan) *(MobileHCI '19)*. Association for Computing Machinery, New York, NY, USA, Article 15, 12 pages. https://doi.org/10.1145/3338286.3340126

[2] M. Chen, G. AlRegib, and B. Juang. 2016. Air-Writing Recognition–Part I: Modeling and Recognition of Characters, Words, and Connecting Motions. *IEEE Transactions on Human-Machine Systems* 46, 3 (June 2016), 403–413.

[3] Mingshi Chen, Panlong Yang, Jie Xiong, Maotian Zhang, Youngki Lee, Chaocan Xiang, and Chang Tian. 2019. Your Table Can Be an Input Panel: Acoustic-based Device-Free Interaction Recognition. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 3, 1, Article 3 (March 2019), 21 pages. https://doi.org/10.1145/3314390

[4] G. E. Dahl, D. Yu, L. Deng, and A. Acero. 2012. Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition. *IEEE Transactions on Audio, Speech, and Language Processing* 20, 1 (Jan 2012), 30–42.

[5] T. Deselaers, D. Keysers, J. Hosang, and H. A. Rowley. 2015. GyroPen: Gyroscopes for Pen-Input With Mobile Phones. *IEEE Transactions on Human-Machine Systems* 45, 2 (April 2015), 263–271.

[6] H. Du, P. Li, H. Zhou, W. Gong, G. Luo, and P. Yang. 2018. WordRecorder: Accurate Acoustic-based Handwriting Recognition Using Deep Learning. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*. 1448–1456.

[7] Tian Hao, Guoliang Xing, and Gang Zhou. 2013. iSleep: Unobtrusive Sleep Quality Monitoring Using Smartphones. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems* (Roma, Italy) *(SenSys '13)*. ACM, New York, NY, USA, Article 4, 14 pages. https://doi.org/10.1145/2517351.2517359

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 1097–1105.

[9] Sugang Li, Xiaoran Fan, Yanyong Zhang, Wade Trappe, Janne Lindqvist, and Richard E. Howard. 2017. Auto++: Detecting Cars Using Embedded Microphones in Real-Time. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 3, Article 70 (Sept. 2017), 20 pages. https://doi.org/10.1145/3130938

[10] Wenzhe Li and Tracy Hammond. 2011. Recognizing Text Through Sound Alone. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence* (San Francisco, California) *(AAAI'11)*. AAAI Press, 1481–1486.

[11] Jian Liu, Yan Wang, Gorkem Kar, Yingying Chen, Jie Yang, and Marco Gruteser. 2015. Snooping Keystrokes with Mm-level Audio Ranging on a Single Phone. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking* (Paris, France) *(MobiCom '15)*. ACM, New York, NY, USA, 142–154. https://doi.org/10.1145/2789168.2790122

[12] A. Manashty, J. Light, and H. Soleimani. 2018. A Concise Temporal Data Representation Model for Prediction in Biomedical Wearable Devices. *IEEE Internet of Things Journal* (2018), 1–1.

[13] Wenguang Mao, Jian He, and Lili Qiu. 2016. CAT: High-precision Acoustic Motion Tracking. In *Proceedings of the 22Nd Annual International Conference on Mobile Computing and Networking* (New York City, New York). ACM, New York, NY, USA, 69–81.

[14] P. Molchanov, X. Yang, S. Gupta, K. Kim, S. Tyree, and J. Kautz. 2016. Online Detection and Classification of Dynamic Hand Gestures with Recurrent 3D Convolutional Neural Networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 4207–4215.

[15] Lindasalwa Muda, Mumtaj Begam, and I. Elamvazuthi. 2010. Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques. *CoRR* abs/1003.4083 (2010). arXiv:1003.4083

[16] Rajalakshmi Nandakumar, Shyamnath Gollakota, and Nathaniel Watson. 2015. Contactless Sleep Apnea Detection on Smartphones. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services* (Florence, Italy) *(MobiSys '15)*. ACM, New York, NY, USA, 45–57. https://doi.org/10.1145/2742647.2742674

[17] Rajalakshmi Nandakumar, Vikram Iyer, Desney Tan, and Shyamnath Gollakota. 2016. FingerIO: Using Active Sonar for Fine-Grained Finger Tracking. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA). ACM, New York, NY, USA, 1515–1525.

[18] V. Pham, T. Bluche, C. Kermorvant, and J. Louradour. 2014. Dropout Improves Recurrent Neural Networks for Handwriting Recognition. In *2014 14th International Conference on Frontiers in Handwriting Recognition*. 285–290.

[19] C. Plapous, C. Marro, and P. Scalart. 2006. Improved Signal-to-Noise Ratio Estimation for Speech Enhancement. *IEEE Transactions on Audio, Speech, and Language Processing* 14, 6 (Nov 2006), 2098–2108. https://doi.org/10.1109/TASL.2006.872621

[20] Arik Poznanski and Lior Wolf. 2016. CNN-N-Gram for Handwriting Word Recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[21] K. Qian, C. Wu, F. Xiao, Y. Zheng, Y. Zhang, Z. Yang, and Y. Liu. 2018. Acousticcardiogram: Monitoring Heartbeats using Acoustic Signals on Smart Devices. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*. 1574–1582. https://doi.org/10.1109/INFOCOM.2018.8485978

[22] Y. Ren, C. Wang, J. Yang, and Y. Chen. 2015. Fine-grained sleep monitoring: Hearing your breathing with smartphones. In *2015 IEEE Conference on Computer Communications (INFOCOM)*. 1194–1202. https://doi.org/10.1109/INFOCOM.2015.7218494

[23] Wenjie Ruan, Quan Z. Sheng, Lei Yang, Tao Gu, Peipei Xu, and Longfei Shangguan. 2016. AudioGest: Enabling Fine-grained Hand Gesture Detection by Decoding Echo Signal. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (Heidelberg, Germany) *(UbiComp '16)*. ACM, New York, NY, USA, 474–485. https://doi.org/10.1145/2971648.2971736

[24] T. N. Sainath, O. Vinyals, A. Senior, and H. Sak. 2015. Convolutional, Long Short-Term Memory, fully connected Deep Neural Networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 4580–4584.

[25] Maximilian Schrapel, Max-Ludwig Stadler, and Michael Rohs. 2018. Pentelligence: Combining Pen Tip Motion and Writing Sounds for Handwritten Digit Recognition. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. ACM, New York, NY, USA, Article 131, 11 pages.

[26] A. Seniuk and D. Blostein. 2009. Pen Acoustic Emissions for Text and Gesture Recognition. In *2009 10th International Conference on Document Analysis and Recognition*. 872–876.

[27] S. Shi, Q. Wang, P. Xu, and X. Chu. 2016. Benchmarking State-of-the-Art Deep Learning Software Tools. In *2016 7th International Conference on Cloud Computing and Big Data (CCBD)*. 99–104.

[28] Ke Sun, Wei Wang, Alex X. Liu, and Haipeng Dai. 2018. Depth Aware Finger Tapping on Virtual Displays. In *Proceedings of the 16th Annual International Conference on Mobile Systems, Applications, and Services* (Munich, Germany) *(MobiSys '18)*. ACM, New York, NY, USA, 283–295. https://doi.org/10.1145/3210240.3210315

[29] Ke Sun, Ting Zhao, Wei Wang, and Lei Xie. 2018. VSkin: Sensing Touch Gestures on Surfaces of Mobile Devices Using Acoustic Signals. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking* (New Delhi, India) *(MobiCom '18)*. ACM, New York, NY, USA, 591–605. https://doi.org/10.1145/3241539.3241568

[30] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 3104–3112. http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf

[31] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. 2016. Rethinking the Inception Architecture for Computer Vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2818–2826.

[32] Junjue Wang, Kaichen Zhao, Xinyu Zhang, and Chunyi Peng. 2014. Ubiquitous Keyboard for Small Mobile Devices: Harnessing Multipath Fading for Fine-grained Keystroke Localization. In *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services* (Bretton Woods, New Hampshire, USA) *(MobiSys '14)*. ACM, New York, NY, USA, 14–27.

[33] Wei Wang, Alex X. Liu, and Ke Sun. 2016. Device-free Gesture Tracking Using Acoustic Signals. In *Proceedings of the 22Nd Annual International Conference on Mobile Computing and Networking* (New York City, New York). ACM, New York, NY, USA, 82–94.

[34] Elliott Wen, Winston Seah, Bryan Ng, Xuefeng Liu, and Jiannong Cao. 2016. UbiTouch: Ubiquitous Smartphone Touchpads Using Built-in Proximity and Ambient Light Sensors. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (Heidelberg, Germany). ACM, New York, NY, USA, 286–297.

[35] YiChao Wu, Fei Yin, and ChengLin Liu. 2017. Improving handwritten Chinese text recognition using neural network language models and convolutional neural network shape models. *Pattern Recognition* 65 (2017), 251 – 264.

[36] Y. Xie, F. Li, Y. Wu, S. Yang, and Y. Wang. 2019. D3-Guard: Acoustic-based Drowsy Driving Detection Using Smartphones. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*. 1225–1233. https://doi.org/10.1109/INFOCOM.2019.8737470

[37] Chao Xu, Parth H. Pathak, and Prasant Mohapatra. 2015. Finger-writing with Smartwatch: A Case for Finger and Hand Gesture Recognition Using Smartwatch. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications (HotMobile '15)*. ACM, New York, NY, USA, 9–14.

[38] X. Xu, J. Yu, Y. Chen, Y. Zhu, and M. Li. 2018. SteerTrack: Acoustic-Based Device-Free Steering Tracking Leveraging Smartphones. In *2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. 1–9. https://doi.org/10.1109/SAHCN.2018.8397115

[39] X. Xu, J. Yu, Y. Chen, Y. Zhu, S. Qian, and M. Li. 2018. Leveraging Audio Signals for Early Recognition of Inattentive Driving with Smartphones. *IEEE Transactions on Mobile Computing* 17, 7 (July 2018), 1553–1567. https://doi.org/10.1109/TMC.2017.2772253

[40] H. Yin, A. Zhou, L. Liu, N. Wang, and H. Ma. 2019. Ubiquitous Writer: Robust Text Input for Small Mobile Devices via Acoustic Sensing. *IEEE Internet of Things Journal* 6, 3 (June 2019), 5285–5296. https://doi.org/10.1109/JIOT.2019.2900355

[41] Tuo Yu, Haiming Jin, and Klara Nahrstedt. 2016. WritingHacker: Audio Based Eavesdropping of Handwriting via Mobile Devices. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (Heidelberg, Germany) *(UbiComp '16)*. ACM, New York, NY, USA, 463–473.

[42] Sangki Yun, Yi-Chao Chen, and Lili Qiu. 2015. Turning a Mobile Device into a Mouse in the Air. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services* (Florence, Italy) *(MobiSys '15)*. ACM, New York, NY, USA, 15–29. https://doi.org/10.1145/2742647.2742662

[43] Sangki Yun, Yi-Chao Chen, Huihuang Zheng, Lili Qiu, and Wenguang Mao. 2017. Strata: Fine-Grained Acoustic-based Device-Free Tracking. In *Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services* (Niagara Falls, New York, USA) *(MobiSys '17)*. ACM, New York, NY, USA, 15–28.

[44] Maotian Zhang, Panlong Yang, Chang Tian, Lei Shi, Shaojie Tang, and Fu Xiao. 2015. SoundWrite: Text Input on Surfaces Through Mobile Acoustic Sensing. In *Proceedings of the 1st International Workshop on Experiences with the Design and Implementation of Smart Objects* (Paris, France) *(SmartObjects '15)*. ACM, New York, NY, USA, 13–17.

[45] Y. Zhang, J. Wang, W. Wang, Z. Wang, and Y. Liu. 2018. Vernier: Accurate and Fast Acoustic Motion Tracking Using Mobile Devices. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*. 1709–1717. https://doi.org/10.1109/INFOCOM.2018.8486365

[46] Zengbin Zhang, David Chu, Xiaomeng Chen, and Thomas Moscibroda. 2012. SwordFight: Enabling a New Class of Phone-to-phone Action Games on Commodity Phones. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services* (Low Wood Bay, Lake District, UK) *(MobiSys '12)*. ACM, New York, NY, USA, 1–14. https://doi.org/10.1145/2307636.2307638

[47] Bing Zhou, Mohammed Elbadry, Ruipeng Gao, and Fan Ye. 2017. BatTracker: High Precision Infrastructure-free Mobile Device Tracking in Indoor Environments. In *Proceedings of the 15th ACM Conference on Embedded Network Sensor Systems* (Delft, Netherlands) *(SenSys '17)*. ACM, New York, NY, USA, Article 13, 14 pages.

[48] Tong Zhu, Qiang Ma, Shanfeng Zhang, and Yunhao Liu. 2014. Context-free Attacks Using Keyboard Acoustic Emanations. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (Scottsdale, Arizona, USA) *(CCS '14)*. ACM, New York, NY, USA, 453–464. https://doi.org/10.1145/2660267.2660296