This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2021.3130036, IEEE Transactions on Mobile Computing

AirText: One-Handed Text Entry in the Air for COTS Smartwatches

Yi Gao, *Member, IEEE,* Siyu Zeng, Ji Zhao, Wenxin Liu, and Wei Dong, *Member, IEEE*

Abstract—Text entry for smartwatches is a useful service for many applications like sending text messages and replying emails. Traditional touchscreen-based approaches are two-handed text entry methods, that could be cumbersome when the user is performing other tasks with one hand. Therefore, we propose AirText, the first one-handed text entry method which achieves accurate and practical handwriting in the air for commercial smartwatches. By analyzing the inertial readings from the smartwatch worn on the wrist, AirText is able to accurately recognize the in-air handwritten characters. However, the wrist movements, which produce the inertial readings, are harmful to the user to focus on the screen. In order to address this challenge, AirText uses a novel cross-modal supervision design to achieve accurate character recognition from small wrist movements. AirText further includes a novel word recommendation method to speed up the text entry. We implement AirText on five smartwatches and evaluate its performance extensively with eight volunteers and more than 25,000 in-air handwritten characters. Results show that AirText outperforms two baseline methods and achieves comparable text entry speed as two-handed approaches.

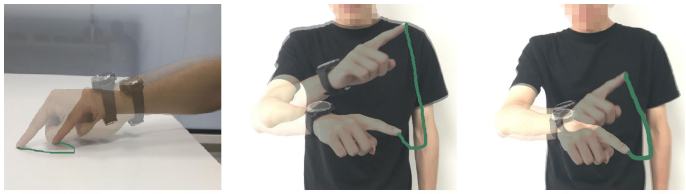
Index Terms—Text Entry, COTS Smartwatches, Cross-modal Supervision

1 Introduction

Smartwatches are evolving from companion gadgets to full-fledged devices with various applications [1], [2], e.g., text messages, emails, fitness tracking, and social networking. To enable more and more complicated applications, effective text entry technique for smartwatches plays an important role.

There are many existing approaches focusing on this smartwatch text entry problem, from both the industry and academia. Touchscreen-based approaches [3], [4], [5], [6], are dominant on both smartphones and smartwatches. However, unlike text entry on smartphones, a user needs to use *both hands* [7] to enter texts on smartwatches, which could be cumbersome when the user is performing other tasks with one hand, e.g., holding an umbrella or holding a child. Speech input is a possible solution to this problem, but it faces practical issues including the following: 1) it may be socially inappropriate in some situations (e.g., at meetings); 2) the environment could be too noisy to support accurate speech input; 3) it could cause privacy exposure.

Therefore, one-handed text entry for smartwatches has attracted much research attention [8], [9], [10], [11], [12] in the past several years. For example, WrisText [9] , FingerT9 [12] and AirDraw [10] are three typical text entry methods for smartwatches. WrisText [9] and FingerT9 [12] implement the keyboards on the smartwathes for the convenience of the users. However, these three approaches require customized hardware or additional Bluetooth devices. Different from these approaches, SHOW [8] is a recent one-handed text entry technique for Commercial Off-The-Shelf (COTS) smartwatches, without requiring hardware modifications. In PhonePointPen [13], a user just needs to hold a smartphone and writes in the air. Like SHOW [8], the trajectory captured by the IMU of the smartphone is the same as the trajectory of the written letter. LightRing [14] designs a ring form factor sensor that consists of a infrared proximity sensor and a 1-axis gyroscope to sense the 2d location of a fingertip on any surface, which has the potential to enable a variety of rich mobile input scenarios. However, it requires the user to perform handwriting on a surface, instead of handwriting in the air, with the forefinger. The biggest difference between handwriting on a surface and handwriting in the air is whether the wrist is allowed to rotate.



(a) on-surface (b) non-rotating-wrist in the air (c) rotating-wrist in the air

Fig. 1. Three one-handed text entry scenarios for smartwatches. From the left to the right, they are (a) handwriting on a surface, (b) handwriting in the air *without* rotating wrist, and (c) handwriting in the air *with* rotating wrist. Since small movements of wrist can help the user focus on the screen, the rotating-wrist scenario is preferred.

Figure 1 shows three different scenarios, i.e., on-surface, non-rotating-wrist in the air, and rotating-wrist in the air. The left one shows handwriting on a surface. In this scenario, the handwriting can be performed by moving the forearm around, and the wrist moves with almost the same trajectory as the fingertip, which is the trajectory of the text. Then the written text can be directly inferred by using machine learning algorithms on the IMU (inertial measure-

- *Y. Gao, S. Zeng, W. Liu and W. Dong are with Zhejiang University and the Alibaba-Zhejiang University Joint Institute of Frontier Technologies, Hangzhou 310027, China. E-mail: gaoyi, zengsy, liuwx, dongw@zju.edu.cn. Wei Dong is the corresponding author.*
- *J. Zhao is with Shanghai Pudong Development Bank Co. ,Ltd. Shanghai Branch, Shanghai, 200000, China. E-mail: zhaoj02@spdb.com.cn.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2021.3130036, IEEE Transactions on Mobile Computing

ment unit, including accelerometer and gyroscope) readings obtained by the smartwatch on the wrist [8], [11]. However, since the movements of the forearm, the smartwatch moves drastically with the handwriting, making the user difficult to focus on the screen. Note that handwriting on a surface can also be done by rotating the wrist, such as the interactions shown in LightRing [14] and SHOW [8]. In LightRing [14] the wrist is allowed to rotate and the smartwatch moves slightly with the handwriting, but a wearable sensor in a ring form factor is required, which can not be found on smartwatches. And in SHOW [8], although the wrist is allowed to rotate since the elbow is used as the support point, the smartwatch still moves drastically with the handwriting, causing the user difficult to constantly monitor the screen content. The middle one shows handwriting in the air, but rotating wrist is not allowed. Since the smartwatch still moves with a similar but smaller trajectory as the fingertip, the written text can also be easily inferred. However, in this scenario, the smartwatch also moves drastically with the handwriting, causing the user difficult to focus on the screen. Since a user needs to constantly monitor the screen content to take necessary actions (e.g., backspacing) during text input, the right case is preferred, i.e., handwriting in the air and the wrist is allowed to rotate and the smartwatch moves slightly with the handwriting (in this case, the wrist is actually acting as an axis of rotation).

In this paper, we propose AirText, the first one-handed text entry method which achieves accurate and practical handwriting in the air for COTS smartwatches. Using only the IMU readings from the smartwatch on the wrist as input, AirText infers the texts written by the fingertip in the air. Since AirText allows the user to rotate her/his wrist when handwriting in the air, the trajectories of the fingertip and the wrist are significantly different. This becomes the major challenge in the design of AirText, i.e., inferring texts written in the air from IMU readings of a smartwatch that does not follow similar trajectories as the fingertip. Figure 2 shows the trajectories of a smartwatch under the three scenarios in Figure 1. We can see that the trajectories of the on-surface case and the non-rotating-wrist scenario are very similar as the characters, while those of the rotating-wrist scenarios are completely different.

In the design of AirText, we propose a deep neural network to infer the trajectories of the fingertip for further analysis. The problem of training such a neural network is that there is no labeled data. It is infeasible to manually label the accurate trajectories of the fingertip. To address this problem, we use Leap Motion [15] to perform cross-modal supervision. Leap Motion is a sensing device which is able to track the skeletons of a hand (including the fingertip) using multiple active infrared sensors. Therefore, using Leap Motion during training, we can extract the trajectories of the fingertip as the supervisory signals for the IMU readings. Then during the test period, AirText only needs the IMU readings as input and does not require the Leap Motion anymore.

In order to achieve accurate and practical text entry on COTS smartwatches, AirText needs to further address the following challenges. First, due to the complex relationships between the movements of the wrist and the fingertip, the inferred fingertip trajectories by cross-modal supervision still include significant noises, causing misclassifications like "D" to "P". Second, smartwatches are resource-constrained devices in terms of computations and memory. It is challenging to design a lightweight yet accurate text entry method for smartwatches, especially with relatively high character misclassification probabilities. Third, different users could write the same character differently, and wear different smartwatches with different IMUs. This poses difficulties for the design of AirText, since it is not feasible to require each user to train a neural network with Leap Motion. Therefore, AirText should be able to achieve high accuracy for new users as well as new smartwatches, without cross-modal supervision using Leap Motion.

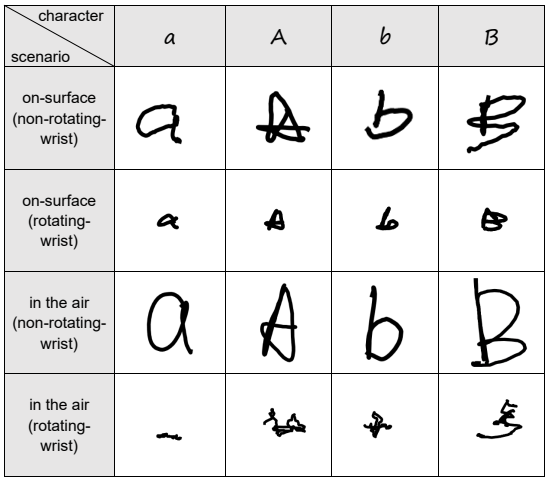


Fig. 2. The trajectories of a smartwatch (recovered by Leap Motion) in the three different scenarios shown in Figure 1, when writing four different characters in the air. We can see that the trajectories in the rotating-wrist scenario are completely different from the written characters. Recovering the fingertip trajectories from the IMU readings produced by these small wrist movements is the major challenge of AirText.

We address the above challenges with various techniques and summarize our contributions as follows.

- We propose AirText, the first one-handed text entry approach which achieves accurate, efficient, and practical handwriting in the air for COTS smartwatches.
- We propose a novel cross-modal supervision method to infer the trajectories of the fingertip from small movements of the wrist. Then the inferred trajectories are classified into different characters in an efficient and accurate manner by the smartwatch alone. We also propose a fast word recommendation algorithm based on a novel dynamic word score calculation method to further improve the word-level input accuracy and speed.
- We implemented AirText and evaluated its performance extensively. Including training and testing, more than 25,000 characters are in-air handwritten by eight different users using five smartwatches. Results show that AirText outperforms two baseline approaches and achieves an in-air handwriting speed of 8.1 words per minute using COTS smartwatches.

The rest of this paper is organized as follows. Section 2 reviews representative one-handed interaction approaches with wearables. Section 3 gives the design considerations

and the overall architecture of AirText. Section 4 and Section 5 describe the two major components of AirText, i.e., the character recognition component and the word recommendation component. Section 6 gives the implementation details of AirText. Section 7 evaluates the performance of AirText extensively, and finally, Section 8 concludes the paper.

2 Related Work

Table 1 summarizes some representative related approaches of AirText, including recent advances of touchscreen-based approaches, on-handed gesture recognition systems, and one-handed text input approaches. In this section, we focus on the one-handed interaction techniques based on wearable devices, which are most closely related to AirText.

Gesture recognition. Using wearable devices, many approaches [16], [17], [18], [19], [20] focus on recognizing gestures of the user. For example, WristFlex [20] is a typical gesture recognition approach using an array of force sensitive resistors worn around the wrist. It is able to distinguish subtle finger pinch gestures, e.g., index pinch and middle pinch. SignSpeaker [16] is a recent real-time sign language recognition system using smartwatches, which is able to recognize more than 100 different sign gestures. ArmTrak [17] exploits the physical constraints of the wrist, the elbow, and the shoulder of a user, and achieves accurate 3D arm tracking and posture recognition with a smartwatch. Float [18] enables touch-free target selection on smartwatches by two gestures, i.e., wrist tilting to *point* and in-air finger tap to *click*.

Different from these gesture recognition approaches, we focus on one-handed text entry for COTS smartwatches in this paper. Generally speaking, text entry and gesture recognition techniques are suitable for different application scenarios. In particular, a major difference of these two kinds of techniques is that the gesture recognition system requires the user and the application to agree on a certain predefined "gesture language" in advance (e.g., push forward to represent single click). Handwriting-based text entry is able to express more complex semantics, which is more suitable for application scenarios like sending text messages.

One-handed text entry. Since handwriting is a natural text entry method, there are already several one-handed handwriting-based text entry approaches proposed in the literature [8], [10], [11]. For example, SHOW [8] is recent text entry approach for COTS smartwatches. A user can perform handwriting *on a surface* using his/her fingertip, then the movements of the smartwatch worn on the wrist can be used for character classification. When the user writes characters on a horizontal surface *without* elbow on the surface, or on a vertical surface, the accuracy drops significantly (54.1% to 71.3%). Similarly, AirDraw [10] also provides on-surface text entry for smartwatches, with the help of a smartphone for calculation. FingerWriting [11] uses a Shimmer [21] IMU worn on the wrist and a smartphone to conduct on-surface handwriting. As mentioned in the introduction section, writing on a surface causes drastic movements of the wrist, which is harmful for the user to focus on the screen. Writing in the air with rotating wrist is more challenging since the trajectories of the fingertip and the wrist could be very different. Further, as writing in the air does not require a nearby surface, it has broader application scenarios.

There are also gesture-based text-entry approaches for smartwatches [9], [22]. WrisText [9] is the state-of-the-art approach within this category. It uses a set of customized hardware to support gesture-based text entry, which includes a Ticwatch 2, 12 infrared proximity sensors, an Arduino DUE, and a laptop for calculation. Based on the customized hardware and a carefully designed keyboard, WrisText achieves accurate and efficient text entry for smartwatches. Different from these approaches, we focus on text entry for COTS smartwatches, without any additional hardware support.

3 Overview

In this section, we will first give several important design considerations of an accurate and practical one-handed text entry method for smartwatches. Then we will describe the overall architecture of AirText, including its main components.

3.1 Design Considerations

Convenient to use. As a text entry method, it is desirable to be convenient to use for novice users. Handwriting using the fingertip is a natural text entry method for a user, almost requiring no extra learning effort.

Cross-device and cross-user. For a practical text entry method, it should not require a user to spend a long time for training his/her own model for each smartwatch. In other words, a trained model should be able to achieve similar performance for a new user and/or a new device as for the user and the device which provide the training set.

Screen stability. During text entry, a user needs to constantly monitor the screen content to take necessary actions (e.g., backspacing, selecting word recommendations). Therefore, it is crucial to keep the screen stable enough for a user to enter text efficiently, especially for smartwatches which suffer from limited screen sizes. As mentioned in the introduction section, this is the major challenge for the design of an in-air handwriting-based text entry method for smartwatches. In AirText, a user can rotate his/her wrist to handwrite a character in the air, without moving the wrist along a similar trajectory as the written character. Due to this screen stability design consideration, gesture-based techniques are not suitable for text entry for smartwatches.

3.2 Overview Of AirText

As shown in Figure 3, the design of AirText consists of four major components.

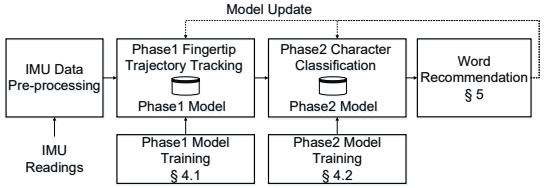


Fig. 3. Overview of AirText.

TABLE 1
Related work of AirText.

Method	COTS Smartwatch	Text Entry	One-handed	In-air Writing	CER	WPM
SHOW [8]	Yes	Yes	Yes	No (on-surface)	0.2%∼45.9%	N/A
SignSpeaker [16]	Yes	No (sign language)	Yes	N/A	N/A	N/A
ArmTrak [17]	Yes	No (arm posture)	Yes	N/A	N/A	N/A
Float [18]	Yes	No (click gesture)	Yes	N/A	N/A	N/A
WristWhirl [19]	No (customized hardware)	No (gesture)	Yes	N/A	N/A	N/A
AirDraw [10]	No (additional smartphone)	Yes	Yes	No (on-surface)	29%	N/A
WrisText [9]	No (customized hardware)	Yes	Yes	Yes	5.92%	9.9
WristFlex [20]	No (customized hardware)	No (pinch gesture)	Yes	N/A	N/A	N/A
FingerWriting [11]	No (additional smartphone)	Yes	Yes	No (on-surface)	5%	N/A
ZoomBoard [6]	Yes	Yes	No	N/A	19.6%	9.8
SplitBoard [3]	Yes	Yes	No	N/A	N/A	9.1
WatchWirter [5]	Yes	Yes	No	N/A	N/A	15.3
SwipeBoard [4]	Yes	Yes	No	N/A	17.5%	24
AirText	Yes	Yes	Yes	Yes	5.8%	8.1

(1) Preprocessing of IMU readings. AirText takes the IMU readings from the smartwatch worn on the wrist of the user as input and preprocesses them for further analysis. The first step is to detect each character, which is accomplished by a threshold-based approach. Since the acceleration variances are significantly different when a user is handwriting a character or not, this simple solution is effective enough for handwriting detection in practice. Then for each detected character, AirText has obtained a number of IMU readings. Due to different sampling frequency of IMUs and different handwriting speeds, the number of IMU readings for each character could be very different. Therefore, AirText re-samples the IMU readings by interpolation and gets a fixed number of IMU readings which are ready for further character recognition. The detailed processing of the IMU data after preprocessing will be described in the next section.

(2) Phase1 of character recognition: fingertip trajectory tracking. Since the trajectory of the fingertip directly reflects the character, AirText includes a deep neural network to infer the fingertip trajectory from the IMU data after preprocessing. As mentioned in the introduction section, the movements of the wrist and the fingertip are significantly different. Nevertheless, AirText manages to use a cross-modal supervision method to train a neural network model, and achieve accurate fingertip trajectory tracking. We refer to this process as *phase1* training and *phase1* character recognition. During the phase1 training, AirText uses a Leap Motion as supervisory signals to train the neural network which is used to infer the fingertip trajectories from the IMU data. AirText exploits the physical constraints of the handwriting to improve the training accuracy. Further, a stroke-cutting algorithm is used in AirText to remove the interference of the extra stroke from the previous character to the current character.

(3) Phase2 of character recognition: character classification. After AirText has inferred the fingertip trajectory, it conducts character classification by a second deep neural network. This *phase2* neural network takes the inferred fingertip trajectory and the preprocessed IMU data as input, then conducts character classification. In order to further improve the classification accuracy, AirText uses the transfer probabilities (i.e., a character is incorrectly recognized as another) of different handwritten characters to fine-tune the classification results. Since the IMU could drift over time [23], and the user may change his/her handwriting habits or smartwatches over time, or a new user comes to use AirText, AirText will updates its phase2 model periodically. This model update is also beneficial for AirText to improve its accuracy of different users as well as different devices. Details are included in Section 4.2.

(4) Word recommendation. In order to improve the word-level text input accuracy and speed up the text input, the word recommendation component of AirText takes the classified characters as input, then recommends several possible words to the user. Considering various misspelling possibilities, i.e., substitution, insertion, deletion, and transposition, AirText is able to successfully recommend the intended word with high probability. Since the search space of possible words could be very large, AirText also includes an efficient and accurate design to select candidate words. Details are described in Section 5.

4 Character Recognition

4.1 Phase1: Fingertip Trajectory Tracking with Physical-constrained Cross-Modal Supervision

In preprocessing, we use a threshold-based approach to split the samples. A sliding window of 33 is acceptable while meeting most people's handwriting speed. After evaluating

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2021.3130036, IEEE Transactions on Mobile Computing

the collected raw sensor data, the timestamp when the standard deviation is larger than 0.2 is determined as the start of a gesture, and less than 0.185 is determined as the end of a gesture. The average length of the IMU sample of a character is 120, which reaches the balance of accuracy and inference time.

After preprocessing, the IMU data is analyzed by the phase1 model to infer the fingertip trajectory. More specifically, each handwritten character includes 130×6 (130 samples, each sample is a row containing a timestamp and three-axes readings measured by the accelerometer and the gyroscope) preprocessed IMU readings. The output of the phase1 model is the positions of three tracking points, i.e., the index fingertip, the second metacarpophalangeal joint, and the wrist shown in Figure 4. In the design of AirText, the output dimension is 120×3×3, which are the 120 consecutive 3D positions (i.e., trajectory) inferred by the phase1 model.

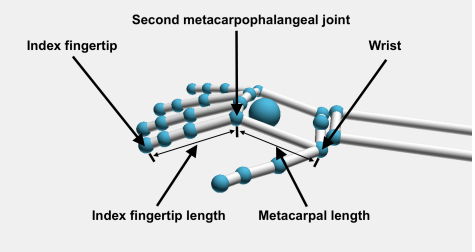


Fig. 4. The output of the phase1 model is the positions of the index fingertip, the second metacarpophalangeal, and the wrist. The two lengths for model calibration are also shown in the figure.

AirText uses a convolutional neural network (CNN) to train the phase1 model. Figure 5 shows the neural network architecture. The dimension of the input layer is 130×12, which is obtained by duplicating the preprocessed IMU data. The main reason of the duplication is to support more layers without padding zeros which will add more noises. Although the duplication does not add more information, it makes the model easier to be trained with good accuracy. We also conducted experiments with different settings: 1) 3-layer network without padding, 2) 5-layer network with padding, 3) 7-layer network without padding. The results show that the current model with duplication achieves the best performance.

Then there are five CNN layers. Each of the first three CNN layer uses 16 3×3 filters and the ReLU activation functions. The fourth CNN layer include 8 3×3 filters and the last CNN layer includes 3 3×2 filters to match the output dimension. After flattening, a full connected layer, and reshaping, the output dimension is 120×3×3, which represents the 120 consecutive 3D positions of the three tracking points, i.e., the index fingertip, the second metacarpophalangeal joint, and the wrist.

Since it is impossible to manually label the positions, AirText uses a Leap Motion sensing device to provide the supervisory signal. By using multiple active infrared sensors, Leap Motion is able to accurately track the skeleton of the hand with an error about 0.7mm on average [24]. We denote the IMU data as **I**, the Leap Motion output as **G**, and the phase1 network as **T**. Then the training objective is to minimize the difference between the inferred positions $\mathbf{T}(\mathbf{I})$ and the ground truth **G** obtained by the Leap Motion:

$$\min_{\mathcal{T}} \sum_{(\mathbf{I},\mathbf{G})} L(\mathbf{T}(\mathbf{I}), \mathbf{G}). \quad (1)$$

The loss function L is almost the summation of squared distances (i.e., $D^2(\cdot)$) of the inferred positions and the ground truth positions for all 120 samples and the three tracking points:

$$L(\mathbf{T}, \mathbf{G}) = c(l^{\mathbf{I}}_{mcp}, l^{\mathbf{G}}_{mcp}) \cdot c(l^{\mathbf{I}}_{if}, l^{\mathbf{G}}_{if}) \cdot \sum_{i=1}^{120} \sum_{j=1}^{3} D^2(\mathbf{T}_{ij}, \mathbf{G}_{ij}), \quad (2)$$

where $\mathbf{T}_{ij}$ and $\mathbf{G}_{ij}$ are the positions of the i-th sample and the j-th tracking point.

Note that there are two additional factors in the loss function, $c(l^{\mathbf{I}}_{mcp}, l^{\mathbf{G}}_{mcp})$ and $c(l^{\mathbf{I}}_{if}, l^{\mathbf{G}}_{if})$, where $l^{\mathbf{I}}_{mcp}$ is the metacarpal length calculated by the inferred positions, $l^{\mathbf{G}}_{mcp}$ is the metacarpal length calculated by the Leap Motion positions, $l^{\mathbf{I}}_{if}$ is the index finger length calculated by the inferred positions, $l^{\mathbf{G}}_{if}$ is the index finger length calculated by the Leap Motion positions.

The intuition of including these two factors is to calibrate the sensing errors by physical constraints. During the handwriting, the length of the index finger and the length of the metacarpal are two constants. However, due to sensing errors, these two lengths calculated by the inferred positions in each sample could be different. In the design of AirText, we use a function $c(\cdot)$ to compensate these sensing errors. Take l_{mcp} as an example, when the difference of $l^{\mathbf{I}}_{mcp}$ and $l^{\mathbf{G}}_{mcp}$ is small, the compensation function will output a small result, which will result in a small loss. This physical-constraint-based loss function calibration can reduce the loss values of samples with high sensing quality, effectively improving the accuracy of the trained model.

Stroke-cutting. When a user is handwriting in the air, the center of each written character keeps almost the same. Therefore, the collected IMU data will include extra strokes from the ending point of the previous character to the starting point of the current character. The IMU data of these extra strokes is harmful for AirText since the strokes of the same character could be very different with different previous characters.

We denote a cut matrix C, in which each element $C_{cur,pre}$ is the percentage of the IMU measurements introduced by the extra stroke given the previous character pre and the current character cur. The goal of the stroke-cutting algorithm is to accurately calculating this matrix C. In the design of AirText, the initial matrix is calculated by analyzing the positions of the starting points and the ending points of English characters from a dataset of handwritten English characters. Since different users may have different handwriting habits in the air, the initial matrix is updated iteratively. More specifically, after AirText collects some samples of each pair of a previous character (e.g., n) and a current character (e.g., p), the stroke-cutting algorithm will update the matrix C. First, the algorithm generates a number of candidate values by increasing and decreasing the current value $C_{\text{“}p\text{”},\text{“}n\text{”}}$. Then the AirText compares the inference accuracy of using each candidate value and update the value to be the one with the best accuracy. By

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2021.3130036, IEEE Transactions on Mobile Computing

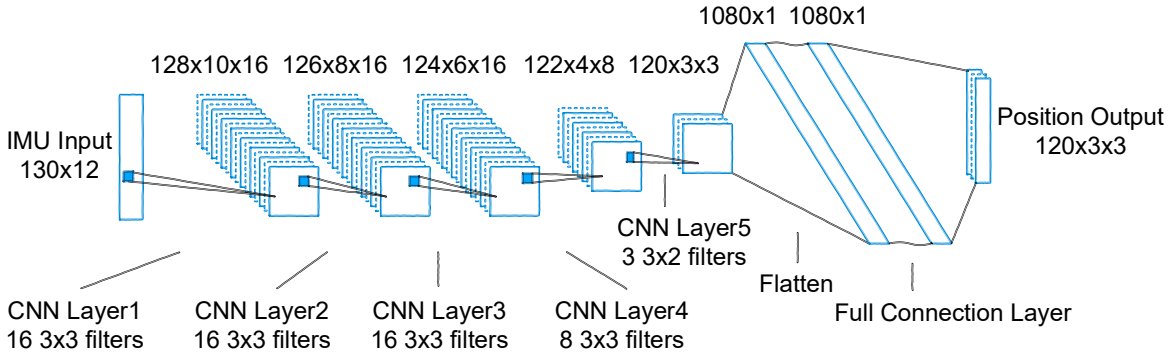


Fig. 5. The phase1 model architecture, including five CNN layers with ReLU activation functions and flattening, a full connection layer, and reshaping. The output is the inferred 120 consecutive 3D positions of the three tracking points.

using this stroke-cutting algorithm, AirText is able to avoid the interference of the extra strokes and achieves better performance.

4.2 Phase 2: Character Classification with Fine-tuning based on Transfer Probabilities

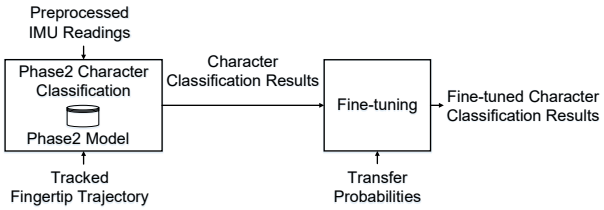


Fig. 6. Phase2 dataflow of character classification, mainly including a CNN-based classification and a fine-tuning module.

Figure 6 shows the dataflow of phase2 character classification of AirText. First, AirText uses a similar CNN with a softmax layer at the last to classify different characters based on the output trajectories of phase1.

The loss function L_2 is the summation of the difference between the Probability Output and labels of each character:

$$L_2 = \sum_{i=1}^{26}(C_i - P_i) \tag{3}$$

where $\mathbf{C}_i$ and $\mathbf{P}_i$ are the probabilities of the i-th letter of the network output and the label.

Since the IMU could drift over time [23], and the user may change his/her handwriting habits or smartwatches over time, or a new user comes to use AirText. To rapidly converge the model, AirText will send the collected IMU samples and corresponding character labels to the cloud to fine-tune the base phase2 model.

Second AirText includes a novel fine-tuning module to further improve the character classification accuracy. The key insight of this fine-tuning module is to exploit the transfer probability matrix. With all the training data, we can obtain a transfer probability matrix which includes all probabilities of incorrectly recognizing a certain character as another one. We denote each entry of this matrix as $P(\alpha|\beta)$, which is the probability of recognizing character β as another character α. Then we view the output softmax score vector of the phase2 CNN as a measurement of the probability of each character, which is denoted as $P(\bar{\alpha})$. For 26 lower case characters, when the phase2 CNN outputs the probability of a character "a" (i.e., $P(\bar{a})$), there are 26 different cases, i.e., it could be correctly recognized by "a" or incorrectly recognized by other 25 characters. Then we have the following system of equations for 26 lower case characters:

$$\begin{cases} P(a)P(a|a) + P(b)P(a|b) + \cdots + P(z)P(a|z) = P(\bar{a}), \\ P(a)P(b|a) + P(b)P(b|b) + \cdots + P(z)P(b|z) = P(\bar{b}), \\ \cdots \\ P(a)P(z|a) + P(b)P(z|b) + \cdots + P(z)P(z|z) = P(\bar{z}). \end{cases} \tag{4}$$

In this system of 26 equations, $P(a), P(b), \ldots, P(z)$ are the 26 unknown probabilities which we want to estimate, $P(a|a), P(a|b), \ldots, P(z|z)$ are 26×26 transfer probabilities which are priori knowledge, and $P(\bar{a}), P(\bar{b}), \ldots, P(\bar{z})$ are the measured probabilities by the phase2 CNN. Therefore, we have 26 equations for 26 unknowns. Due to measurement errors, these equations are usually unsolvable. AirText uses Gauss−Seidel method to calculate an optimal estimation for each unknown probability $P(\alpha)$.

To summarize, this fine-tuning module takes the output of the phase2 CNN (i.e., $P(\bar{\alpha})$) and the priori knowledge of transfer probabilities (i.e., $P(\alpha|\beta)$) as input, then outputs a better estimation of the probability for each character (i.e., $P(\alpha)$). In the evaluation section, we will show the effectiveness of this fine-tuning module.

5 Word Recommendation

After the characters of a word have been recognized, AirText further includes a word recommendation component to increase the word-level accuracy and speed. Due to reasons like incorrect character recognition and word misspelling, the word recommendation component of AirText will recommend several most possible words to the user. In this section, we will describe the details about how AirText finds these most possible words.

We denote a word recognized by the character recognition component as W. Because there is no guarantee that all the characters in W are correctly recognized, W might not be the intended word that the user wants to enter. We use Damerau-Levenshtein distance [25] between two words to describe the dissimilarity between different words.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2021.3130036, IEEE Transactions on Mobile Computing

The Damerau-Levenshtein distance between two words is defined as the minimum number of edit operations (substitutions, insertions, deletions, of a single character, or a transposition of two adjacent characters) required to change one word into another. Further, for candidate words with the same distance from W, AirText will calculate a *word score* $S(W)$ for each candidate word and recommend those with the highest scores.

5.1 Dynamic Word Score Calculation

Word recommendation is not new in the literature. The basic idea of word recommendation is to use a frequency-based statistical model [8], [26], [27], [28], i.e., a word with higher frequency in a specific linguistic database has a higher word score and will get recommended with a higher probability. For example, since the word "top" has a higher frequency than the word "toy" in the linguistic database, it will be recommended with a higher probability when W="tob". We refer to this approach as static-word-frequency-based recommendation, i.e., the word frequency is a static value.

This approach is suitable for keyboard-based text entry methods, but not suitable for AirText. The problem of using this approach in AirText is that it overlooks *how* the user has handwritten the word in the air. Concretely, when a user uses AirText to input a character, the character recognition component will output a fine-tuned probability vector ($P(\alpha)$) of all characters in the alphabet, which in fact carries the information about *how* the user has handwritten the character in the word. In other words, if a user writes the same character twice, the output probabilities vector could be very different and the word score should be calculated dynamically with considering these probabilities.

In the design of AirText, we propose a novel dynamic word score calculation method. Since the most common edit operation is substitution, we use one-character-substitution as an example to describe the dynamic word score calculation process. Given a word $W = C_1C_2 \dots C_N$, where N is the length of the word, we denote $W_{n,\beta}$ be the changed word whose nth character is substituted by another character β. We define the word score $S(W)$ of the word W as follows.

$$S(W) = P(W) \cdot \prod_{i=1}^{N} P_i(C_i), \tag{5}$$

where $P(W)$ is the word frequency in the linguistic database (the static part), and $P_i(C_i)$ is the probability that a character C_i is recognized in the ith position in the word W. The $P_i(C_i)$ is dynamic since it is the output of the character recognition component, which could be different in each time the user handwrites the character. In order to speed up the lookup operations of word frequencies (P(W)), we use a dictionary tree data structure to store the word frequencies.

Then we can calculate the word score for a changed word $W_{n,\beta}$ after substitution as follows.

$$S(W_{n,\beta}) = P(W_{n,\beta}) \cdot \prod_{i=1}^{N} P_i(C_i'),\ C_i' = \begin{cases} \beta, & i = n; \\ C_i, & i \neq n, \end{cases} \tag{6}$$

where C_i' represents a changed ($i = n$) or unchanged ($i \neq n$) character at the ith position of the word W. The word score of each candidate word obtained by other three operations and larger Damerau-Levenshtein distances can be calculated similarly. Given the word score for each candidate word, AirText will recommend the top three words as well as the original recognized word to the user. There are four words are presented on the screen. The left word is the input word of the user, and the other three words are the recommended words given by the word recommendation component. The user interface is shown in Figure 7(a). The user can use a small tilt gesture to select the correct word from the recommended words shown in Figure 7(b). The user can use a "backspace" gesture by shaking the smartwatch up and down once to delete the wrong characters and retype the word if the "wrong" word appears. Since the gesture "shaking" the smartwatch up and down is significantly different from the gestures of hand-writing normal characters in the air, the accuracy of this backspace gesture is almost one hundred percent in the evaluation. But the usage scenarios are in steady situations like sitting still. Under other impacts, like walking, Airtext is not available due to the noises from other irrelevant body movements.

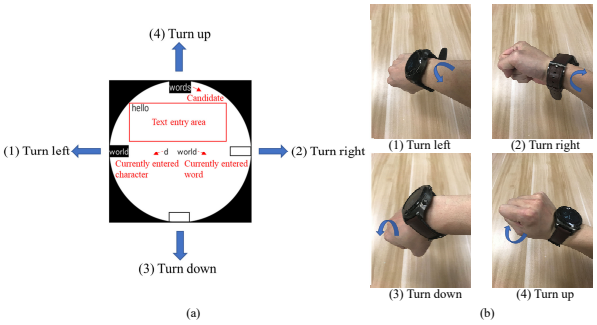


Fig. 7. Four candidate words presented on the top, bottom, left and right of the smartwatch screen, the user can use a small tilt gesture to select the correct word.

TABLE 2
Details of the smartwatches.

Smartwatch	Processors	Battery	RAM	OS	IMU
LG Watch Urbane	1.2 GHz Quad-Core	400 mAh	512 MB	Wear OS 2.20	InvenSense MPU-6515
Huawei Watch 2 Pro	1.1 Ghz Quad-Core	420 mAh	768 MB	Wear OS 2.0	InvenSense ICM-20690
Ticwatch Pro	1.2 GHz Quad-Core	415 mAh	512 MB	Wear OS 2.7	InvenSense MPU-6515
TicWatch 2	1.2 GHz Quad-Core	300 mAh	512 MB	Ticwear OS 4.1	InvenSense MPU-6515

6 Implementation

AirText is implemented on five COTS smartwatches with four different models, including one LG Watch Urbane, two Huawei Watch 2 Pro, one TicWatch Pro, and one TicWatch 2. Table 2 shows the details information about these smartwatches. Note that the IMU InvenSense MPU-6515 is widely used in many other popular smartwatches, e.g., LG G Watch, Moto 360, Samsung Gear 2, and Gear Fit. AirText includes a text entry service running on smartwatches, and a cloud service running on a remote server for model update.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2021.3130036, IEEE Transactions on Mobile Computing

Smartwatch Side. AirText utilizes the built-in accelerometer and gyroscope in smartwatches, and acquires the IMU data through existing AndroidWear APIs. During the training period, we use a Leap Motion to collect the ground truth of the fingertip trajectories, as shown in Figure 8. To avoid the volunteers being influenced by the trajectories displayed on the screen, the monitor is turned aside. All the volunteers handwrote characters in a natural way they like. Using the collected data, the two-phase models are implemented by TensorFlow [29] and trained on an NVIDIA Tesla P100 (16 GB memory, Single-Precision Performance 9.3 teraFLOPS). We use the TensorFlow Lite converter to convert the trained TensorFlow models into an optimized FlatBuffer format, so that they can be used by the TensorFlow Lite interpreter which is supported on smartwatches.

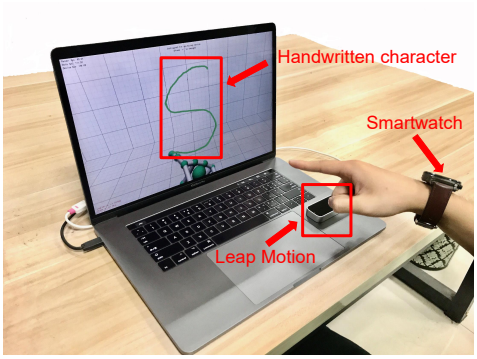


Fig. 8. The setup when using Leap Motion to train the phase1 CNN model in AirText.

Cloud Service for Model Update. The cloud side service of AirText is mainly used for model update. There are two different cases of model update. First, when AirText is used for a new user or a new device, the pre-installed CNN models will be updated frequently to converge to stable performance. Second, since a user may change his/her input habits and the smartwatch IMU may drift over time [23], the models should be updated periodically. In the evaluation section, we will show the results of these two kinds of model update.

Overhead. For the current implementation, the average inference time duration of character classification is 202 ms, including 104 ms for phase1 and 98 ms for phase2, with a standard deviation of 24.5 ms. The average energy consumption of entering each character is about 0.1J. Given the 400 mAh battery capacity and the 3.7 V working voltage, when a user inputs 20 sentences (30 characters per sentence on average) a day using the smartwatch, the total energy consumption is smaller than 2% of the battery capacity.

7 Evaluation

In this section, we evaluate the text input performance of AirText extensively. We will first describe the evaluation methodology, including the dataset, the metrics, and the baselines for comparative study. Then we will give the main evaluation results of AirText as well as two baselines. We will also analyze the performance gain of each key component of AirText in detail. Finally, we will evaluate the robustness of AirText considering different impact factors.

TABLE 3
Distribution of the dataset.

		Training	Testing
Phase1	User8	10,982 characters (With device5)	2,746 characters (With device5)
Phase2	User1~7	7 x 728 characters	7 x 312 characters
	User8	5 x 728 characters (With 5 devices)	5 x 312 characters (With 5 devices)
	User1~7	20 sentences	

7.1 Evaluation Methodology

Dataset [30]. During the training data collection and testing process, we didn't require the volunteers to handwrite characters in a fixed way, that is by rotating their wrists. In contrast, all the volunteers handwrote characters in a natural way they like, AirText allowed volunteers to rotate their fingers relative to their hands. Besides, they were required to sit or stand still to maintain the stability of their bodies. Other impacts like walking will produce too much noises to classify the characters. So the training data contains various IMU data corresponding to different handwriting habits, and volunteers can handwrite naturally while testing. We just recommend the volunteers to handwrite by rotating their wrists, because this makes it easier for the volunteers to focus on the screen, but they don't have to keep handwriting in that way.

Phase1. One of the volunteers (user8) performed in-air handwriting of English words using LG Watch Urbane to provide the dataset for phase1. A total number of 13,728 training and testing samples were collected, and each sample includes the Leap Motion data and the IMU data for one handwritten character.

Phase2. All the eight volunteers performed in-air handwriting to provide the test set which includes a total number of 12,480 (12×1040) test samples. Each sample also has the Leap Motion data, but only for trajectory comparison and extracting the information of the character such as writing speed, size, environment, and time. Note that the test sets provided by the user1 to user7 (7×1040) were collected after the phase2 model was updated by the model update component (evaluated in 7.3.2) for the user1 to user7 respectively, and the phase2 model needn't to be updated for the user8 because he is the original user. User8 also wrote other 5×1040 samples for all five smartwatches mentioned in Section 6, including one LG Watch Urbane, two Huawei Watch 2 Pro, one TicWatch Pro, and one TicWatch 2. We denote the dataset (8×1040) provided from all volunteers with Huawei Watch 2 Pro 1 as the user set $\mathbf{S}_{user}$, the dataset (5×1040) provided from user8 with all five smartwatches as the device set $\mathbf{S}_{device}$. The words handwritten by the volunteers are randomly selected from the Enron mobile email dataset [31] which is widely used for testing mobile text entry methods. In particular, 42 sentences are randomly selected for performance evaluation of AirText.

Usually, the compass is used to calibrate IMU drifts. During data collection, in order to keep the screen stable, the movements of the wrist are small. And short writing time makes the IMU drifts are not significant. Due to the noises of the compass itself, the accuracy is decreased after calibrating.

Metrics. In order to evaluate the text entry performance of AirText, we use the word error rate (*WER*) as the major performance metric. WER is a standard metric for evaluating many text entry systems [32], which considers three types of text entry errors, i.e., deletion error, insertion error, and substitution error. WER is defined as follows:

$$WER = \frac{D + I + S}{D + C + S}, \tag{7}$$

where I is the number of insertion error words, D is the number of deletion error words, S is the number of substitution error words, and C is the number of correctly recognized words.

The character error rate (*CER*) is also an important metric to measure the characters recognition performance, which can be calculated by the ratio of incorrectly recognized characters within all characters. According to WrisText [9] and FingerT9 [12], the CER of below 10% is acceptable and word recommendation will further decrease the WER.

The text entry speed is a metric to measure AirText's efficiency of entering text, which is defined as the number of entered words per minute (WPM):

$$WPM = \frac{N_{total} - N_{wrong}}{T} \tag{8}$$

where N_{total} is the number of the entered words, N_{wrong} it the number of the words which are neither correctly inputted nor recommended, and T is the time spent to enter these words.

After writing a character, there is an average pause time of 0.5s to display the inferred result on the watch screen. It takes about 0.1s to detect the end of writing, and 0.3s for two networks to infer the trajectories and the character. The rest of 0.1s is for preprocessing, word recommendation, and other calculations.

Baselines. Two baselines are used for comparative study, the BLSTM model used in SignSpeaker [16] and the phase2 CNN model in AirText. SignSpeaker is recent state-of-the-art work to use smartwatches to recognize sign gestures. Its core component is a BLSTM model to use IMU data to classify hand gestures. Although the application scenario of SignSpeaker is different from AirText, the BLSTM component in SignSpeaker can be used as a baseline to evaluate the performance of the two-phase CNN model in AirText. In fact, this BLSTM model is the state-of-the-art technique to use IMU data for gesture classification. Therefore, we implemented the BLSTM model and replaced the two-phase CNN model with this BLSTM model to get the baseline approach, which is referred to as BLSTM.

In addition to this BLSTM baseline, we also implemented a variation of AirText as another baseline, which does not include the phase1 model. The phase1 model is to use IMU data to infer the fingertip trajectories based on cross-modal supervision method. Without the phase1 model, the remaining phase2 model is just a CNN model to classify characters from IMU data. Therefore, we refer to this baseline as IMU-CNN in the following.

7.2 Text Entry Accuracy and Speed: A Comparative Study

In this section, we show the evaluation results of the comparative study of AirText and two baselines, in terms of both accuracy and speed.

WER Comparison. We first evaluate the WER performance when the training and testing are conducted by user8 with LG Watch Urbane, i.e., one-user one-device case. The WER results of AirText and two baselines are shown in Figure 9. The average WERs of AirText, IMU-CNN and BLSTM are 3.9%, 30.9% and 57.1%, respectively. The results show that compared with the BLSTM model, directly using CNN model (i.e., IMU-CNN) could achieve a higher accuracy. However, without cross-modal supervision in the phase1 model of AirText, the phase2 CNN model alone cannot achieve sufficiently high accuracy for text input in practice. Note that Sentence index 14 and 37 have nearly zero deviation due to the limitation of the word recommendation. The words that have low frequency in the word dictionary will hardly be ranked in the top.

We further evaluate the performance of AirText under the multi-user ($\mathbf{S}_{user}$) as well as the multi-device ($\mathbf{S}_{device}$).

WER of different users and using different devices. We then show the WER performance of AirText when different users or different devices were involved in the training and testing processes. These experiments are able to show the ability of generalizing the model of AirText to different users as well as different devices.

Figure 10(a) shows the results of one user with five smartwatches. The average WERs of AirText on five smartwatches are 8.3%, 7.5%, 6.5%, 7.7% and 3.9%, respectively. As expected, the accuracy of one-user multi-device case is lower than that in the one-user one-device case. A WER about 7% means there is one incorrectly recognized word in every 15 words on average, which we believe is still acceptable for in-air handwriting.

Figure 10(b) shows the results of different users with the same device. The average WERs of AirText of the eight users are 11.2%, 5.9% 4.3%, 4.0%, 3.6% , 5.9%, 4.7% and 3.9%, respectively. We can see that the different users have greater impact on the accuracy compared with different devices. Nevertheless, with the help of the model update component, AirText can still achieve high accuracy for different users. Next, we will show the text entry speed considering the incorrectly recognized words during the test.

Text Entry Speed. The speed of text entry is an overall metric to evaluate the performance of a text entry method. Since the WER of the BLSTM baseline is about 57%, it is impractical to use such a text entry method. Therefore, we compare the speed of AirText and IMU-CNN in terms of words per minute (WPM). Figure 11 shows the results on $\mathbf{S}_{user}$ and $\mathbf{S}_{device}$. We can see that on average, AirText achieves a WPM of 8.1, while the IMU-CNN baseline only has a WPM of 4.6. Note that this input speed is comparable to some two-handed touchscreen-based text entry approaches [4], [6] which achieve 9.8 WPM and 9.1 WPM in practice. These results also show that cross-modal supervision (i.e., the component which the IMU-CNN baseline does not include) significantly improves the performance of AirText.

7.3 Performance Gain Analysis

We then analyze the performance gain of the key components in AirText.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2021.3130036, IEEE Transactions on Mobile Computing

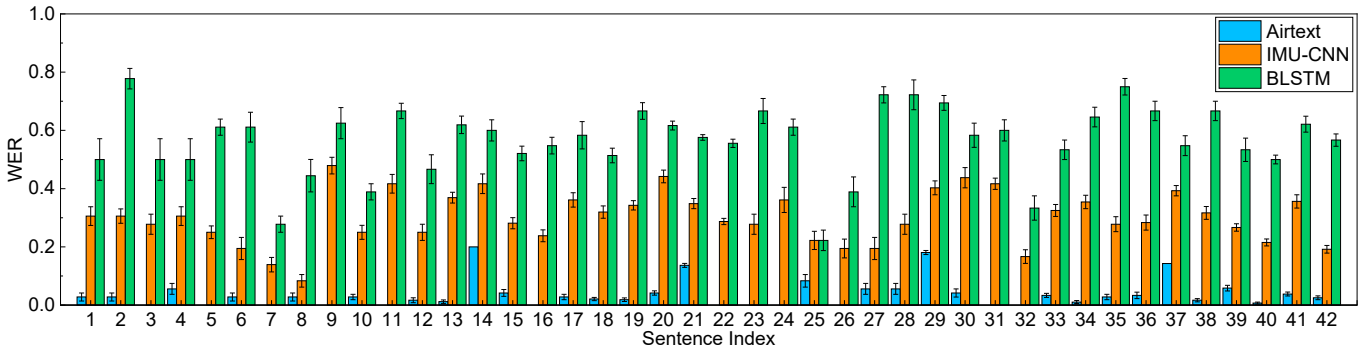
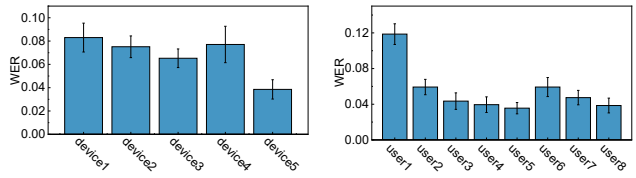
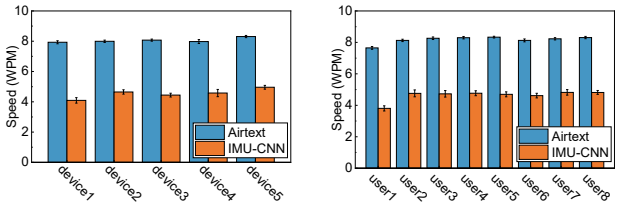


Fig. 9. WER comparison of AirText and two baselines for the 42 sentences. AirText achieves higher accuracy than two baselines for all sentences.



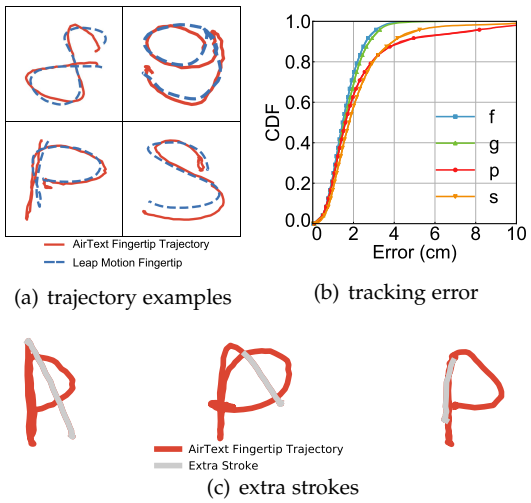
(a) one-user multi-devices (b) one-device multi-users

Fig. 10. WER of a user using different devices and different users using the same device.



(a) one-user multi-devices (b) one-device multi-users

Fig. 11. WPM of AirText and IMU-CNN when a user using different devices and different users using the same device. On overage, AirText achieves 8.1 WPM.



(a) trajectory examples (b) tracking error

(c) extra strokes

Fig. 12. Fingertip trajectories, the corresponding CDF of the tracking error and extra strokes.

7.3.1 Fingertip Trajectory Tracking

The first key component is the fingertip trajectory tracking component based on cross-modal supervision. The accuracy of the fingertip trajectory tracking can be measured as the 3D positioning difference compared with the Leap Motion results. We selected four characters ("f", "g", "p", "s") each 100 times from four users (user2, user4, user6, user8). Figure 12 shows the results, including some example trajectories, the CDF of the positioning errors, and the effect of cutting the extra stroke. The median errors for handwriting the four characters are 1.5cm, 1.5cm, 1.6cm, and 1.8cm, respectively. Since the size of an in-air handwritten character is usually about 12cm×12cm, the fingertip tracking is sufficiently accurate to support accurate character recognition. These results show that the extra strokes can be accurately cut by the stroke-cutting algorithm.

7.3.2 Character Classification

Character classification performance can be measured by the character error rate (CER).

With and without fingertip trajectory tracking. The CER of AirText for all users with and without fingertip trajectory tracking (i.e., the phase1 model) is shown in Figure 13. The average CER of AirText is 5.8%, while the average CER is 26.5% without fingertip trajectory tracking. The result shows that the performance gain of the character classification mainly comes from accurate fingertip trajectory tracking.

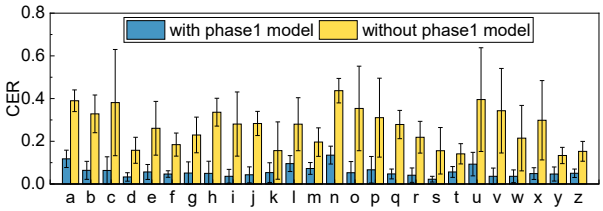


Fig. 13. CER of AirText with and without fingertip trajectory tracking.

Transfer probabilities. Figure 14 shows the transfer probabilities of the character classification of AirText. $\mathbf{S}_{user}$ was used to calculate the transfer probabilities of 26 lower case characters. We can see that all the 26 lower case characters achieve high accuracy. In particular, these transfer probabilities are very different from that of a tradition touch-screen keyboard. For example, character pairs like ("w", "e") are usually easy to incorrectly enter for a touchscreen keyboard. However, the transfer probabilities of these two characters using AirText are almost zero, since their trajectories are significantly different.

Model convergence for new users. When a new user starts to use AirText, the model update component will update the two-phase models to quickly converge to stable

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2021.3130036, IEEE Transactions on Mobile Computing

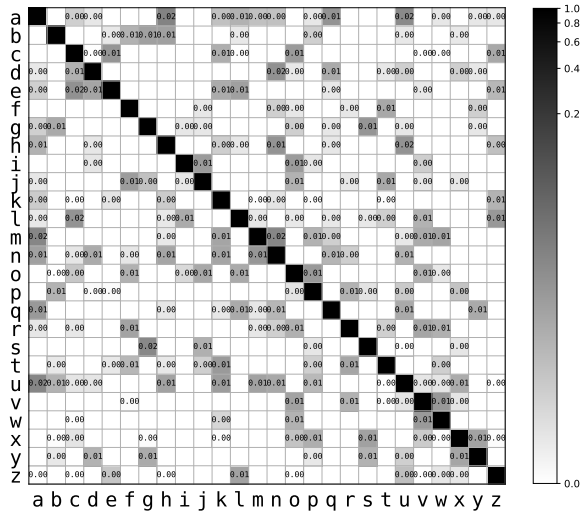


Fig. 14. Transfer probabilities of 26 lower case characters. The value in entry (i, j) represents the probability that a character of the ith row is recognized as a character of the jth column. In order to show the close-to-zero probabilities, the grayscale is non-linear.

performance. We have trained a base model with the training samples provided by user8 in Section 7.2 before. The phase2 models of other users were all updated on the base model of the user 8. Seven users were asked to use AirText to input about 20 sentences a day without Leap Motion. Then AirText updated the model for each user during the test every 8 hours. Figure 15 shows how the CER performance converges for the seven users. After two days (i.e., about 40 sentences), the CERs of six users (user2 to user7) are already below 10%. The CER of the remaining user (user1) also drops below 10% after four days. After seven days, the CERs of seven users (user1 to user7) are 9.4%, 6.1%, 4.8%, 5.0%, 4.6%, 5.7% and 5.2%, respectively. The results show that with the model update module, the performance of AirText for new users is able to converge to an acceptable level quickly.

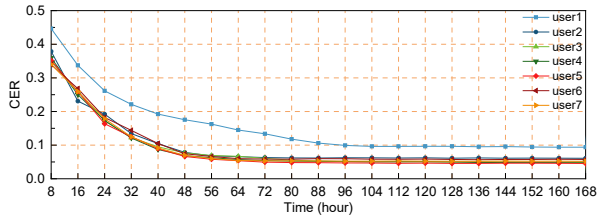


Fig. 15. Model convergence performance. Six new users' CERs decrease below 10% within 48 hours.

Fine-tuning based on transfer probabilities. After the phase2 CNN model, AirText also includes a fine-tuning module to calibrate the probability of each character. This fine-tuning is based on the transfer probabilities of each pair of characters. In fact, Figure 14 includes the transfer probabilities of the 26 lower case characters.

In order to quantify the performance gain of this fine-tuning module, we use the following calculation method. For all the characters which are not correctly recognized by the phase2 CNN model, we compare the probabilities before and after the fine-tuning and report the relative improvement. For example, if a letter "x" is not correctly recognized and its probability is 0.3, the performance gain of the fine-tuning is 10% if the probability becomes 0.33 after fine-tuning. Figure 16 shows the results. On average, the performance gain of the fine-tuning is about 7.4%. This improvement shows that the fine-tuning module based on the transfer probabilities can provide a more accurate estimation of the probability of each handwritten character.

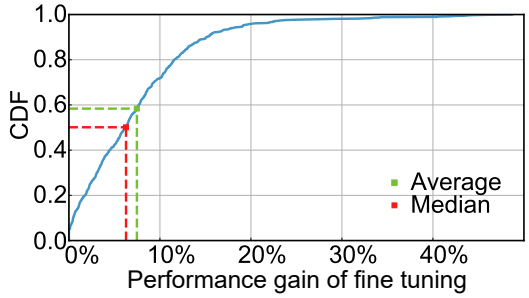


Fig. 16. Performance gain of the fine-tuning method based on the transfer probabilities. On average, the fine-tuning method increases the probabilities of the correct characters by 7.4%.

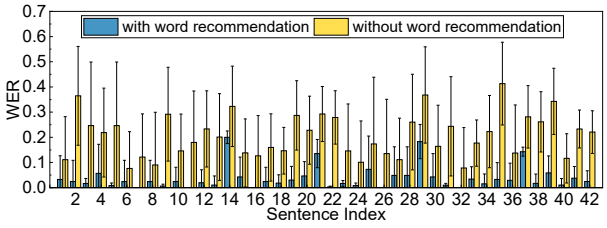


Fig. 17. WER of AirText with and without word recommendation for the 42 sentences. Results show that the word recommendation component of AirText significantly improve the text entry accuracy at the word level, i.e., from a WER of 20.7% to a of WER 3.7%.

7.3.3 *Word Recommendation*

In this experiment, we show the WER of AirText for different users (user1 to user8) when the word recommendation used or not. The results are shown in Figure 17. On average, the WERs with and without the word recommendation component are 3.7% and 20.7%, respectively. We can see that using the word recommendation component can significantly improve the performance of AirText.

7.4 Robustness of AirText

In these experiments, we examine the robustness of AirText in terms of handwriting speed, character size, and environment on $\mathbf{S}_{user}$. As for long-term IMU drift, all users were asked to write extra sentences to test Airtext periodically.

Impact of handwriting speed. Each volunteer handwrite characters with a speed he/she feels comfortable. The handwriting speed is defined as the time cost of handwriting a character. The handwriting time is calculated by the timestamps during data collection. In Figure 18(a) we can see that when the user handwrites a character too fast or too slow, the CER will increase. The reason is that in the training set, the handwriting speed of most of the training samples is about 1.1 second. Therefore, when the writing speed is similar to the training set, the accuracy is maximized.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMC.2021.3130036, IEEE Transactions on Mobile Computing

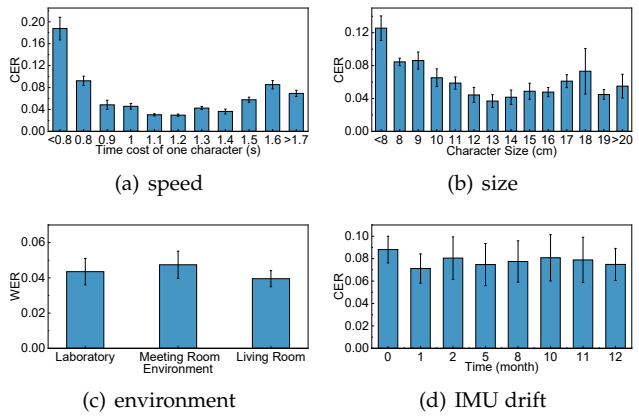


Fig. 18. Robustness of AirText. WER or CER results of AirText with considering different impact factors, i.e., the handwriting speed, the character size, the environment, and the long-term IMU drift.

Impact of character size. Different users handwrite characters in the air with different sizes. The character sizes are extracted from the Leap Motion data. Since the Leap Motion data obtains the positions of the fingertip, the character's height is defined as the highest reading minus the lowest reading, and the same for the width. Figure 18(b) shows the impact of character size. We can see similar results as the impact of handwriting speed, i.e., characters with a size about 13cm can be correctly recognized with the maximal probability. An interesting observation is that for a small number of samples where the character size becomes larger than 19cm, the character recognition accuracy increases. After analyzing the raw data carefully, we find the reason as follows. When a user handwrites a character with a very large size, in addition to the wrist rotation, the wrist will move along the trajectory as the fingertip. These movements provide effective features for character recognition.

Impact of different environments. We also evaluate the impact of different environments on the performance of AirText, including a laboratory, a meeting room, and a living room. Figure 19 shows these three environments, with computers, projectors, and household appliances. Figure 18(c) shows that the AirText achieves high accuracy in all the three environments. In particular, accuracy reaches the most stable in the living room. A possible reason is that compared with the first two environments which are inside a teaching building, the living room inside a residential building introduces weaker magnetic interference.

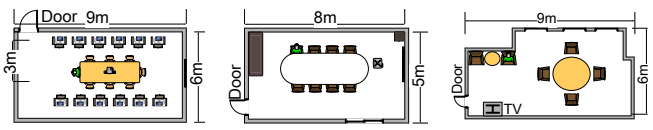


Fig. 19. Three different environments. The first two are inside a teaching building and the last one is inside a residential building.

Impact of long-term IMU drift. IMUs could have long-term drift [23] which is harmful to the text entry performance. In order to evaluate the performance of AirText under this long-term IMU drift, we conducted the following experiments. Given a trained model, we asked the users to test AirText after several months and obtained the results. Figure 18(d) shows the results during the year. Note that these experiments are not done every month. We can see that with the model update module, the CER of AirText keeps stable during the whole year.

8 Conclusion

In this paper, we propose AirText, an accurate, efficient, and practical text entry method for COTS smartwatches. AirText includes a novel cross-modal supervision design to accurately infer the fingertip trajectories from the small movements of the wrist. Then the character recognition component of AirText uses a CNN model to classify characters from their inferred trajectories. Finally, the recognized characters are fed into the word recommendation component, which is able to find the candidate words efficiently and accurately. We implemented AirText on five commercial smartwatches and evaluated its performance extensively. Eight volunteers participated in the evaluation of AirText, and more than 25,000 characters were handwritten in the air. Results show that AirText outperforms two baselines and achieves high accuracy comparable text entry speed as two-handed approaches.

Limitations and future work. The first limitation is that the user needs to pause between the characters. As future work, we will try to train a model to directly infer a word from the fingertip trajectories of the whole word. Since the inference time is shorter than the writing time, another solution is to find a light-weighted cutting method to cut the characters. Therefore, Airtext can infer the previous character while writing the next. Second, currently the smartwatch is put on the dominant hand for a better input experience, because it's uncomfortable to handwrite with the non-dominant hand. Technically, both the left hand and the right hand are supported by AirText. In the future, we will train new models to support both hands. Third, we will further investigate the fine-grained relation between the fingertip movements and the wrist movements. A possible approach is to decompose the handwriting into more basic movements (e.g., left/right circle, slash/backslash, vertical/horizontal line), and train more models to classify characters. The forth limitation of AirText is that if the users use AirText to enter a large amount of text, they may suffer fatigue and the text entry speed is relatively slow. But since AirText is used on smartwatches, users will not use AirText to enter a large amount of text. Instead, they will only use they will only use it for some short message replies (e.g. reply a message "I am meeting" during a meeting). Fifth, although Airtext can adapt to different users and devices easily by fine tuning, considering some state-of-art approaches of domain adaption will improve the performance and robustness of Airtext furthermore. FiDo [33] provides a faster approach to converge to different users and devices. Currently, Airtext is not usable under the impacts of the body movements. EI [34] focuses on eliminating the environmental features and PCIDA [35] focuses on domain adaption on continues domains. In the future, these approaches will be tried to get rid of the impacts of the movements.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation of China (No. 61872437 and No. 61772465), and the Zhejiang Provincial Natural Science Foundation for Distinguished Yound Scholars under No. LR19F020001.

REFERENCES

[1] Z. Wang, T. Zhao, J. Ma, H. Chen, K. Liu, H. Shao, Q. Wang, and J. Ren, "Hear sign language: A real-time end-to-end sign language recognition system," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2020.

[2] Y. Jiang, Z. Li, and J. Wang, "Ptrack: Enhancing the applicability of pedestrian tracking with wearables," *IEEE Transactions on Mobile Computing*, vol. 18, no. 2, pp. 431–443, 2019.

[3] J. Hong, S. Heo, P. Isokoski, and G. Lee, "Splitboard: A simple split soft keyboard for wristwatch-sized touch screens," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI)*, 2015, pp. 1233–1236.

[4] X. A. Chen, T. Grossman, and G. Fitzmaurice, "Swipeboard: A text entry technique for ultra-small devices that supports novice to expert transitions," in *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST)*, 2014.

[5] M. Gordon, T. Ouyang, and S. Zhai, "Watchwriter: Tap and gesture typing on a smartwatch miniature keyboard with statistical decoding," in *Proceedings of the 34th AnnualACM Conference on Human Factors in Computing Systems (CHI)*, 2016, pp. 3817–3821.

[6] S. Oney, C. Harrison, A. Ogan, and J. Wiese, "Zoomboard: A diminutive qwerty soft keyboard using iterative zooming for ultra-small devices," in *Proceedings of the 31th Annual ACM Conference on Human Factors in Computing Systems (CHI)*, 2013, pp. 2799–2802.

[7] R. Jang, C. Jung, D. Mohaisen, K. Lee, and D. Nyang, "A one-page text entry method optimized for rectangle smartwatches," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2021.

[8] X. Lin, Y. Chen, X.-W. Chang, X. Liu, and X. Wang, "Show: Smart handwriting on watches," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies (UbiComp)*, vol. 1, no. 4, pp. 151:1–151:23, 2018.

[9] J. Gong, Z. Xu, Q. Guo, T. Seyed, X. A. Chen, X. Bi, and X.-D. Yang, "Wristext: One-handed text entry on smartwatch using wrist gestures," in *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*, 2018, pp. 181:1–181:14.

[10] D. Moazen, S. A. Sajjadi, and A. Nahapetian, "Airdraw: Leveraging smart watch motion sensors for mobile human computer interactions," in *Proceedings of the 13th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 2016, pp. 442–446.

[11] C. Xu, P. H. Pathak, and P. Mohapatra, "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)*, 2015, pp. 9–14.

[12] P. C. Wong, K. Zhu, and H. Fu, "Fingert9: Leveraging thumb-to-finger interaction for same-side-hand text entry on smartwatches," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 2018, pp. 1–10.

[13] S. Agrawal, I. Constandache, S. Gaonkar, R. Roy Choudhury, K. Caves, and F. DeRuyter, "Using mobile phones to write in air," in *Proceedings of the 9th international conference on Mobile systems, applications, and services*, 2011, pp. 15–28.

[14] W. Kienzle and K. Hinckley, "Lightring: Always-available 2d input on any surface," in *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, 2014, p. 157–160.

[15] "Leap motion," 2019. [Online]. Available: https://www.leapmotion.com/

[16] J. Hou, X.-Y. Li, P. Zhu, Z. Wang, Y. Wang, J. Qian, and P. Yang, "Signspeaker: A real-time, high-precision smartwatch-based sign language translator," in *The 25th Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2019.

[17] S. Shen, H. Wang, and R. Roy Choudhury, "I am a smartwatch and i can track my user's arm," in *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2016, pp. 85–96.

[18] K. Sun, Y. Wang, C. Yu, Y. Yan, H. Wen, and Y. Shi, "Float: One-handed and touch-free target selection on smartwatches," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI)*, 2017, pp. 692–704.

[19] J. Gong, X.-D. Yang, and P. Irani, "Wristwhirl: One-handed continuous smartwatch input using wrist gestures," in *Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST)*, 2016, pp. 861–872.

[20] A. Dementyev and J. A. Paradiso, "Wristflex: Low-power gesture input with wrist-worn pressure sensors," in *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST)*, 2014, pp. 161–166.

[21] "Shimmer," 2019. [Online]. Available: http://www.shimmersensing.com/

[22] K. Katsuragawa, J. R. Wallace, and E. Lank, "Gestural text input using a smartwatch," in *Proceedings of the International Working Conference on Advanced Visual Interfaces (AVI)*, 2016, pp. 220–223.

[23] Y. Yuksel, N. El-Sheimy, and A. Noureldin, "Error modeling and characterization of environmental effects for low cost inertial mems units," in *IEEE/ION Position, Location and Navigation Symposium (PLANS)*, 2010, pp. 598–612.

[24] F. Weichert, D. Bachmann, B. Rudak, and D. Fisseler, "Analysis of the accuracy and robustness of the leap motion controller," *Sensors*, vol. 13, no. 5, pp. 6380–6393, 2013.

[25] G. V. Bard, "Spelling-error tolerant, order-independent pass-phrases via the damerau-levenshtein string-edit distance metric," in *Proceedings of the fifth Australasian symposium on ACSW frontiers (ACSW)*, 2007, pp. 117–124.

[26] A. Fazly and G. Hirst, "Testing the efficacy of part-of-speech information in word completion," in *Proceedings of the 2003 EACL Workshop on Language Modeling for Text Entry Methods (TextEntry)*, 2003.

[27] J. Goodman, G. Venolia, K. Steury, and C. Parker, "Language modeling for soft keyboards," in *Proceedings of the 7th international conference on Intelligent user interfaces (IUI)*, 2002, pp. 194–195.

[28] C. P. Willmore, N. K. Jong *et al.*, "Text prediction using combined word n-gram and unigram language models," 2017, patent NO 9,785,630, Filed May 5th., 2015, Issued Oct. 10th., 2017.

[29] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[30] "Airtext dataset," 2020. [Online]. Available: https://www.dropbox.com/sh/d0793k9lyg4bpmz/AADLmuF7Tg33S2sD90DojZRBa?dl=0

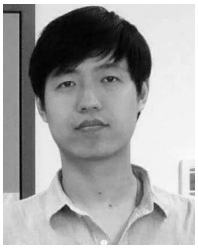
[31] K. Vertanen and P. O. Kristensson, "A versatile dataset for text entry evaluations based on genuine mobile emails," in *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI)*, 2011, pp. 295–298.

[32] A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks," in *Proceedings of the 31st International conference on machine learning (ICML)*, 2014, pp. 1764–1772.

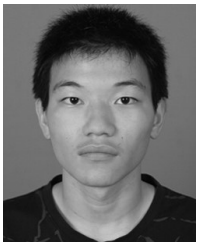
[33] X. Chen, H. Li, C. Zhou, X. Liu, D. Wu, and G. Dudek, "Fido: Ubiquitous fine-grained wifi-based localization for unlabelled users via domain adaptation," in *Proceedings of The Web Conference 2020*, 2020, pp. 23–33.

[34] W. Jiang, C. Miao, F. Ma, S. Yao, Y. Wang, Y. Yuan, H. Xue, C. Song, X. Ma, D. Koutsonikolas *et al.*, "Towards environment independent device free human activity recognition," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, 2018, pp. 289–304.

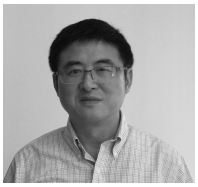
[35] H. Wang, H. He, and D. Katabi, "Continuously indexed domain adaptation," *arXiv preprint arXiv:2007.01807*, 2020.



Yi Gao (M'15) received the B.S. and Ph.D. degrees from Zhejiang University, China, in 2009 and 2014, respectively. From 2015 to 2016, he visited McGill University as a Visiting Scholar. He is currently an Associate Professor with Zhejiang University. His research interests include Internet of Things, network measurement, mobile and edge computing. He is a member of the IEEE and the ACM.



Siyu Zeng received the M.S. degree in computer technology from Zhejiang University, China, in 2021. He is currently a software development engineer. His research interests include sensor networks and wireless sensing.



Ji Zhao received the Ph.D. degree in computer science from Zhejiang University, China. His research interests include data mining, financial informatization and fintech. He is a review expert of Shanghai government procurement, the head of information Technology risk Team of Shanghai Banking Network Industry Association.

Wenxin Liu received the M.S. degree in computer science from Zhejiang University in 2020. He is currently a Software Development Engineer. His research interests include sensor networks and wireless sensing.

Wei Dong (S'08–M'12) received the B.S. and Ph.D. degrees from the College of Computer Science at Zhejiang University in 2005 and 2011, respectively. He is currently a full professor in the College of Computer Science at Zhejiang University. He leads the Embedded and Networked Systems (EmNets) lab in Zhejiang University. He has published over 100 papers in prestigious conferences and journals including MobiCom, INFOCOM, ICNP, ToN, TMC, etc. His research interests include Internet of Things and sensor networks, wireless and mobile computing, and network measurement. He is a member of the IEEE and the ACM.