UltraWrite: A Lightweight Continuous Gesture Input System with Ultrasonic Signals on COTS Devices

Yongpan Zou, *Member, IEEE*, Weiyu Chen, Yunshu Wang, Canlin Zheng, Wenfeng He, and Kaishun Wu, *Fellow, IEEE*

***Abstract*—Due to the advantages of device ubiquity, natural interaction and privacy preservation, acoustic-based gesture input has received widespread attention. Researchers have proposed various techniques for different applications. However, the existing work has shortcomings of heavy data-collection overhead, non-continuous input, and performance degradation in cross-user scenarios. To overcome these shortcomings, we propose UltraWrite, an acoustic-based gesture input system that only needs extremely low data-collection overhead, supports continuous input, and achieves high cross-user recognition accuracy. The key idea of our solution is to synthesize training data of continuous gestures from isolated ones, build a lightweight continuous gesture recognition model based on connectionist temporal classification (CTC) mechanism, and design a novel decoupled model training strategy to improve its cross-user recognition capability. We have implemented prototype systems on commercial devices and conducted comprehensive experiments to evaluate their performance. The results show that UltraWrite achieves an average top-1 word accuracy of 99.3% and top-1 word error rate of 0.34%. In addition, we have also evaluated UltraWrite's robustness to the sensing distance, angle, background noise, and device. The results reveal that UltraWrite possesses strong robustness to these factors.**

***Index Terms*—Acoustic Sensing; Continuous Gesture Input; Cross-Domain Learning; Connectionist Temporal Classification**

I. Introduction

With the emergence of smart devices, novel human-computer interaction (HCI) techniques have garnered growing attention from academia. Researchers have proposed various HCI approaches for different applications. Among them, human gesture recognition (HGR) is particularly attractive and has been studied extensively which provides a natural, innovative, and modern way of non verbal communication. According to the adopted sensing modality, the existing works can be mainly categorized into vision-based [1, 2], radio frequency-based [3–5], inertial sensor-based [6, 7], and acoustic-based [8–16]. Among different approaches, gesture recognition based on acoustic signals has gained much attention due to the ubiquity of sensors, fine-grained granularity, and high robustness to ambient interference.

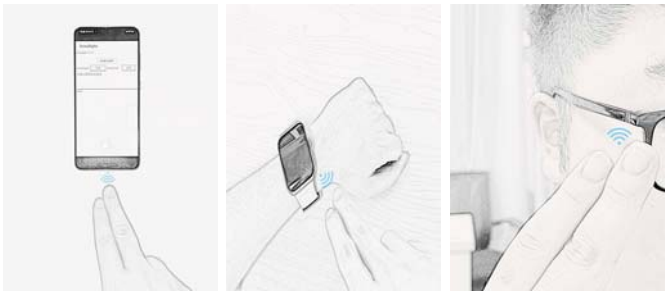


Fig. 1. Possible scenarios where UltraWrite can be applied

However, current works in this field have several key limitations. First, they require collecting a massive amount of training data to train a recognition model, which brings about overwhelming labor overhead. For instance, UltraGesture [13] and Ipanel [17] need to collect 100 and 50 samples for each kind of gesture, respectively. The data-collection overhead is particularly heavy for gesture-based text-input applications since the categories of texts are more diverse. Although model-based approaches such as FingerIO [14], Soundtrak [15], and AMT [16] do not require extensive training, they rely on multiple microphones for localization and tracking. Moreover, due to strict frequency response requirements, these methods are difficult to deploy on mobile devices, lacking broad applicability. Second, the vast majority of existing acoustic HGR systems recognize discrete gestures, meaning there are pauses between gestures. This reduces the efficiency and user experience of human-computer interaction, especially in gesture-based text input applications. Although SonicASL [18] can recognize continuous ASL gestures, they are not able to recognize unseen words as its basic unit of training data is a word. At last, existing works usually encounter severe performance degradation for unseen users due to diverse gesture habits. To address this issue, some works [19, 20] personalize and fine-tune the already trained model, and others [8, 21, 22] make use of domain adaptation techniques for model optimization. However, such methods have two critical

Y. Zou, W. Chen, Y. Wang, C. Zheng, and W. He are with the College of Computer Science and Software Engineering, Shenzhen University, 3688 Nanhai Ave, Shenzhen, Guangdong, China, 518060. E-mail: yongpan@szu.edu.cn; richardcwy426@gmail.com; wangyunshu2018@email.szu.edu.cn; zhengcanlin2020@email.szu.edu.cn.

K. Wu is with the Information Hub, The Hong Kong University of Science and Technology (Guangzhou), 1 Duxue Road, Guangdong, China. E-mail: wuks@ust.hk.

Manuscript received September xx, 2024

deficiencies. On one hand, these models are computationally expensive which are difficult to be supported by mobile devices. On the other hand, these methods require an unseen user to provide samples for retraining the model. For continuous gesture input applications, this becomes unacceptable as the number of feedback samples increases significantly. As a result, a crucial question arises: *Can we design a low-overhead and cross-user-friendly system for continuous gesture text entry?*

In this paper, we utilize active sensing to capture finger-writing activities with high-frequency acoustic signals emitted from a speaker and received by a microphone which are available in commercial devices such as smartphones, tablets, and PCs. To address the aforementioned challenges, we employ several strategies in constructing our system called UltraWrite. First, we propose a training dataset construction approach based on decomposition and reconstruction to minimize system overhead. The collection of continuous gesture data is decomposed into the acquisition of isolated gestures, followed by the synthesis of continuous gestures from these isolated instances. Second, to accomplish continuous gesture input, we formalize it as a natural machine translation (NMT) task and make use of the Connectionist Temporal Classification (CTC) mechanism to tackle the difficulty of annotating continuous gesture data. To reduce the complexity of the model, we apply LeNet and Bi-GRU as feature extractors and translation modules, respectively. Third, to address the cross-user issue without adding usage costs for new users, we devise a decoupled training strategy by integrating domain adaptation during encoding reconstruction and model decoupling. We use encoding reconstruction to train the feature extractor to extract domain-independent features. Based on the above techniques, we design a real-time system that is cost-effective, supports continuous input, and exhibits excellent cross-user performance. Fig. 1 shows the potential application scenarios of UltraWrite.

In summary, our work highlights the following main contributions:

1) We design a novel dataset construction scheme by combining task decomposition and reconstruction with a data augmentation method. This scheme can not only effectively reduce the overhead of collecting training datasets to an extremely low level, but also guarantee a high recognition accuracy.
2) We transform the problem of continuous gesture input into an NMT task and resolve it based on CTC mechanism widely used in the speech recognition domain. In addition, we also propose a cross-user training strategy that combines domain adaptation for encoding reconstruction with model decoupling training.
3) We implement a prototype real-time system on Android platform on commercial mobile devices. The experimental results indicate that our system achieves an accuracy of 99.29% for seen words and 83.04% for unseen words across users, validating the promising practical usability of UltraWrite.

The rest of this paper is organized as follows. Sec. II discusses the related works. Sec. III introduces the details of system design. In Sec. IV and Sec. V, we demonstrate the system implementation, experimental setup, and performance evaluation, respectively. Finally, Sec. VII concludes the paper.

II. Related Work

A. Acoustic-based Gesture Recognition

Gesture recognition based on acoustic signals can be categorized into acoustic feature-based methods [13, 17, 23–32] and model-based methods [14–16]. Zou et al. employed the Doppler effect to create AcouDigit [23], EchoWrite [24], and EchoWrite 2.0 [25], which allow for letter and number input on mobile devices. Similarly, Wang et al. and Liu et al. presented RoboCIR [26] and UltraGesture [13], respectively, to detect changes in channel impulse response (CIR) resulting from gestures, achieving high-precision recognition of 15 and 12 hand gestures. For alphabetic and numeric input, Yang et al. used sound signals to develop WordRecorder [27] and Ipanel [17]. Additionally, Ma et al. sequentially proposed UbiWriter [28] and WritingRecorder [29]. However, feature-based methods typically require the collection of a large amount of training data. Additionally, the aforementioned methods are unable to handle continuous gesture input.

Gollakota et al. proposed FingerIO [14], which implements fine-grained finger tracking for gesture recognition by modeling sound propagation. Similarly, Soundtrak [15] achieves gesture recognition by extracting phase changes from the echo. Zhu et al. proposed the AMT [16], realizing high-precision gesture recognition. Although these approaches eliminate the time-consuming and labor-intensive data collection process, they often require multiple microphones for computation. By leveraging the Doppler effect, we propose a novel cross-domain training strategy and data synthesis approach, enabling universal and lightweight continuous gesture recognition.

B. Continuous Gesture Recognition

Continuous gesture interaction has been shown to enhance the user experience, particularly in scenarios involving text input for human-computer interaction. To this end, Yang et al. utilized motion sensors on smartwatches to develop Signspeaker [33], which enables continuous input of Sign Language gestures. Wang et al. proposed MyoSign [34] and WearSign [35], which employ motion sensors and electromyographic (EMG) signals collected from Myo armband devices [36]. They also developed WriteAS [37] for English input on smartwatches based on IMU signals. Wang et al. developed DeepSLR [38] for hand gesture recognition using two Myo armbands. However, these systems rely on IMU and EMG for gesture input, and DeepSLR [38] and WearSign [35] require the collection of EMG signals to capture subtle muscle movements, limiting their use on non-handheld devices. Furthermore, differences in signal processing schemes between the two media prevent these systems from being interchangeable. Jin et al. proposed SonicASL [18], which implements acoustic-based continuous sign language recognition. The basic recognition unit of SonicASL is a word, which means that the system is not able to recognize words that have not appeared.

In summary, the systems necessitate costly training datasets and do not address cross-user issues.

C. Domain-Adaption Method

Zou et al. proposed JADA [21], and Wang et al. developed Rfree-GR [22], both utilizing an adversarial domain adaptation scheme to design model networks and training strategies to address cross-domain user and cross-environment problems. Yang et al. designed an adversarial domain adaptation deep learning network framework to tackle cross-modal problems in gesture interaction data [39]. Tang et al. proposed WiGr [40], which uses a non-confrontational domain adaptation scheme to measure the similarity of feature spaces between two domains. This approach improves feature extraction for cross-domain invariant features and addresses cross-user and cross-environment challenges. Similarly, Pheng et al. proposed MDO-K [41] to solve cross-modal challenges. Yang et al. proposed AirFi [42], which mitigates the performance degradation in cross-environment testing for WiFi sensing tasks by aligning the feature distributions of the source and target domains to the same prior distribution.

III. System Design

In this section, we elaborate on the design details of our system. As illustrated in Fig. 2, we first characterize gesture movements using the Doppler effect of sound waves. To reduce the cost of constructing the training dataset, we use synthetic instead of real collected data, employing data augmentation methods to expand the dataset. Next, we design a model based on CNN to extract features from the input data. Finally, these features pass through a context encoder to produce output words.

A. Doppler Effect

According to the Doppler effect, when a user writes within the range of the speaker's acoustic field, the recorded echo signal by the microphone experiences a frequency shift. The magnitude of this frequency shift is determined by the relative motion velocity between the finger and the device, as described by the equation:

$$\Delta f = f_0 \cdot \left| 1 - \frac{v_s \pm v_f}{v_s \mp v_f} \right| \tag{1}$$

In this equation, the variables f_0, v_s, and v_f represent the frequency of the emitted signal, the speed of signal propagation in the air, and the speed of relative motion between the finger and the device, respectively.

B. Signal Processing

The acoustic signal captured by the microphone is initially converted into a spectrogram using an STFT with a Hamming window. We set the frame length to 8192 and the window step size to 512 based on empirical methods. We use a window size of 100 frames with a 70% overlap for sliding segmentation.

*1) **Denoising**:* For our experiment, the transmission frequency of the microphone is set to 19 KHz, and the velocity of finger movement is measured to be approximately 2.5 m/s. Consequently, the maximum frequency shift is calculated to be 281 Hz according to equation Eq.(1). Thus, the theoretical effective frequency band is $[18722, 19281]$ Hz. To simplify the calculations, we extend the effective frequency range to $[18700, 19300]$ Hz and extract the spectrogram data within this range to eliminate background noise.

The spectrogram reveals that the frequency shift caused by finger movement appears blurred, indicating a low signal-to-noise ratio in the raw signal. This blurring is primarily due to the direct transmission of the acoustic signal from the speaker to the microphone, resulting in a strong frequency component at the center frequency (i.e., 19 KHz), as illustrated in Fig. 3a. Consequently, we use a fourth-order Butterworth band-stop filter to filter out the frequency band near the center frequency.

*2) **Gesture detection**:* Based on the denoised spectrogram, we employ the cumulative sliding window method to intercept the motion gesture signals. Referring to Eq.(1), the spectrogram illustrates an increase in the recorded echo signal's frequency when the finger approaches the device, as depicted in Fig. 3a. Conversely, a decrease in the echo signal's frequency is observed, as shown in Fig. 3b. During periods without gesture actions, all frequency components within the current STFT frame exhibit signal strengths below 0 dB, as demonstrated in Fig. 3c. To implement the sliding window approach, we set the window size (W_{slide}) to 5 frames, with a sliding step size of 2 frames.

Algorithm 1 Gesture Detection Algorithm

Input: Raw Spectrogram P_{Raw}
Output: Gesture Spectrogram P_{Ges}

```
while True do
    W_slide = GetSlideWindow(P_Raw);  {Window Sliding}
    Flag = CheckGestureStart(W_slide); // Returns True if the gesture is the starting point
    StartPoint = W_slide[0];
    while Flag do
        W_slide = GetSlideWindow(P_Raw);
        Flag = CheckGestureEnd(W_slide);
        if Flag = False then
            Flag = CheckNext20StftWindow(P_Raw);
        end if
    end while
    EndPoint = W_slide[0];
    p_Ges = P_Raw[StartPoint : EndPoint];
    PushToBufferArea(P_Ges);
end while
```

The current sliding window is positioned at the beginning of the word gesture when the dominant frequency, which is the frequency component with the highest signal strength, deviates from 19 KHz within all frames encompassed by the sliding window. Consequently, we designate the first window within the sliding window as the starting point of the word gesture. The sliding window progresses in increments of 2 frames. Once the dominant frequency of all frames within the sliding

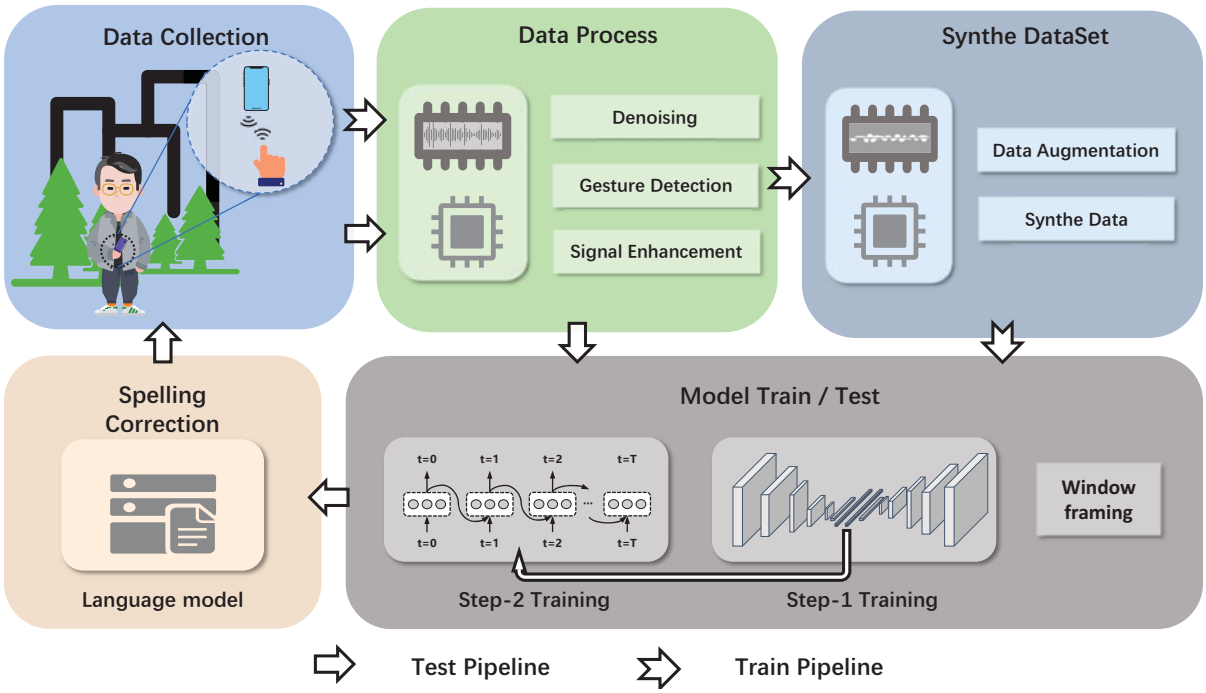
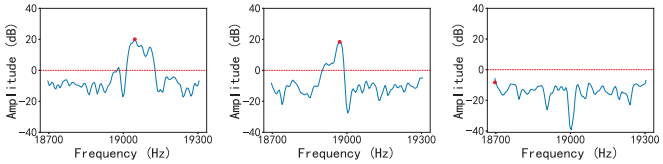


Fig. 2. The overview of UltraWrite's architecture



(a) Colse (b) Remove (c) Norm

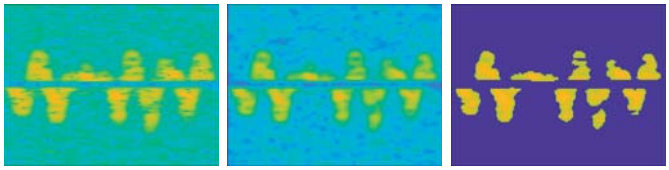
Fig. 3. Relationship between finger relative motion and frequency variation

window equals 19 KHz, it marks the end of the word gesture. In this case, we designate the first STFT window within the sliding window as the tentative endpoint of the gesture action.

Furthermore, users may experience brief pauses between letters while writing. Based on this, after determining the endpoints, the sliding window examines the subsequent 20 frames (approximately 300 ms) to check if they contain gesture instances. The aforementioned gesture detection process is summarized in Algorithm 1.

*3) **Signal enhancement**:* Due to hardware constraints, simultaneous operation of these devices generates some noise. These non-zero components appear as light yellow patches in the spectrogram and are referred to as random noise. As observed in Fig. 3b, the frequency shift induced by gestures remains somewhat masked by this noise. To address this issue, we employ normalization, Gaussian smoothing, and thresholding techniques to enhance the signal of the detected and intercepted gesture spectrogram denoted as P_{Ges}.

Firstly, we utilize zero-one normalization to scale the frequency signal strength of the spectrogram to a value within the range of $[0, 1]$ dB. This normalization helps mitigate the influence of absolute amplitude variations, as demonstrated in Fig. 4a. Next, we employ the Gaussian blur method to smooth the spectrogram. The Gaussian filter, which involves a 2D convolution smoothing operation, effectively blurs the image and removes finer details. This smoothing process is illustrated in Fig. 4b. We set the window size (k_{size}) of Gaussian smoothing to 5, and the standard deviation (σ) is set to 1. After the Gaussian smoothing, we proceed with thresholding. Prior to thresholding, we establish a threshold value (α) to determine which elements of the spectrogram should be set to 0. Fig. 4c illustrates the application of thresholding, where elements exceeding the threshold are preserved. Through experimental observations, we determine that an appropriate value for α that maximizes the quality of the enhanced spectrogram can be set to 6.8 in our system design.



(a) After Normalization (b) After Gaussian smoothing (c) After Thresholding

Fig. 4. Different stages of signal enhancement of extrude the doppler shifts pattern of word gesture "AM"

C. Gesture Design and Analysis

The system aims to achieve word gesture input, but each person has unique handwriting characteristics. Conventionally, building an extensive training dataset requires inviting different volunteers to write word gestures. Moreover, in daily life, there are up to 6200 high-frequency vocabulary words, making

it impractical to include all these words in the training set. We decompose the collection of words into the collection of letters. In this section, we standardize the writing of letters and analyze the factors affecting gesture frequency offset.

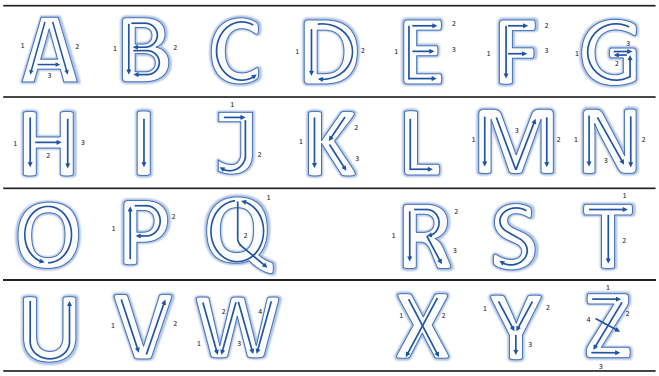
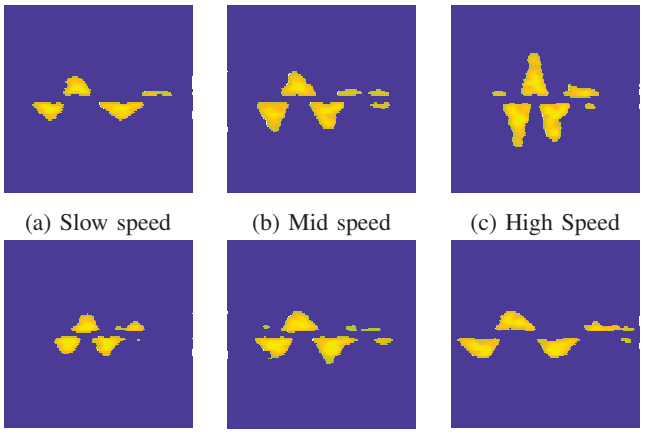


Fig. 5. The basic stokes specification of letters



(a) Slow speed (b) Mid speed (c) High Speed

(d) Small amplitude (e) Mid amplitude (f) Large amplitude

Fig. 6. 'A' 's spectrogram in different speeds and amplitudes

1) ***Gesture design****:* The letter gesture consists of multiple strokes, and the frequency offset formed by writing different strokes varies. Due to variations in writing habits, different users may have different writing sequences for the same letter. Taking the letter 'H' as an example, some users may prefer a vertical-horizontal-vertical writing style, while others may lean towards a vertical-vertical-horizontal style. Similarly, if letter gestures with different writing styles are simultaneously collected, it would increase the construction cost of the training dataset and make recognition more challenging. Therefore, standardizing the writing of letters is necessary. After investigating the writing habits of multiple volunteers and referring to the teaching standards for children's letter writing, we standardize the writing of 26 letters as shown in Fig. 5. We will discuss further in Sec. III-D.

2) ***Gesture analysis****:* According to Formula Eq.(1), the frequency offset caused by gesture motion is determined by the relative velocity between the microphone and the finger. During the writing process, the relative velocity is influenced by both the user's writing speed and amplitude. Furthermore, standardizing writing speed and amplitude is impractical as it directly affects the user's writing experience.

To summarize the impact of these two factors on gesture features, we invite participant (V_1) to write the letter 'A' at different speeds and amplitudes. The experimental results are shown in Fig. 6. It can be observed that the amplitude of the frequency offset increases with a higher writing speed, and the writing time increases with a larger writing amplitude. Additionally, when the writing amplitude is kept constant, an increase in writing speed leads to a shorter writing time. Thus, the frequency offset feature is influenced by both writing speed and amplitude in the time domain, while in the frequency domain, it is only affected by writing speed. We will discuss this in detail in Sec. III-D.

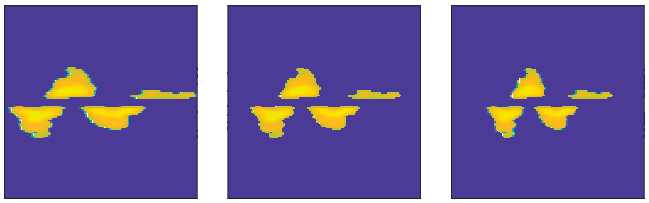
D. Dataset Construction

Based on the analysis in Sec. III-C2, variations in writing speed and amplitude among users introduce significant diversity in the collected data, making data acquisition for gesture recognition systems a challenging and labor-intensive process. However, we observe that for most users, writing speed and amplitude fall within a fixed range. This observation enables us to leverage data augmentation techniques to enhance the original gesture features. Furthermore, we identify that transitions between words can introduce additional Doppler frequency shifts, which we refer to as "reset gestures." Given the complexity of word-level data augmentation and the high cost of data collection, using words as the augmentation unit poses considerable challenges. To address this, we decompose the augmentation unit into individual letters and reset gestures. By applying time-warping techniques, we stretch and compress these gestures in the time domain. Subsequently, the discrete letter and reset gesture segments are synthesized into words following the augmentation strategy, as illustrated in Table I. This letter-level synthesis approach significantly reduces the cost associated with acquiring a comprehensive training dataset while maintaining the integrity of the gesture recognition process.

We analyze the spectrograms of the 676 feasible combinations of 2-gram letter gestures obtained from 26 letter gestures. We categorize the starting and ending points of the gesture into three positions: top, middle, and bottom, considering only the finger's distance in the direction perpendicular to the device. By combining the stroke order of letters, we get six Doppler frequency shift patterns, namely top to top ($t2t$), top to bottom ($t2b$), middle to top ($m2t$), middle to bottom ($m2b$), bottom to top ($b2t$), and bottom to bottom ($b2b$). Furthermore, the $t2t$ and $b2b$ resetting gestures denote relative motions in the direction parallel to the device, without significant frequency shifts. As a result, we disregard these two resetting gestures and ultimately select 26 letters and 4 resetting gestures as our isolated gestures.

The Doppler shift pattern is influenced by the relative motion speed between the finger and the smart device. The relative movement speed is affected by both the writing speed and magnitude. Based on the discussion in Sec. III-C2, the writing speed influences both the time domain and the frequency domain of the Doppler shift pattern, whereas the

writing magnitude solely affects the time domain. Therefore, we focus on constraining the writing magnitude during the collection of letter gestures and resetting gestures. Specifically, the volunteer was instructed to maintain a consistent writing magnitude of 10×16 cm^2. There were no restrictions imposed on the writing speed within a single session, which encompasses the writing of letter gestures from '*A*' to '*Z*'.



(a) $\rho = 0.80$ (b) $\rho = 1.00$ (c) $\rho = 1.20$

Fig. 7. Enhancement results for different values of the factor ρ for letter gesture 'A'

*1) **Data augmentation**:* By implementing the letter and homing gesture collection scheme described above, we enhance the diversity of the gesture data spectrograms in the frequency domain. To further improve the diversity of the spectrograms in the time domain, we employ a data augmentation technique inspired by the time-warping technique used in speech recognition. Specifically, for the spectrograms of the input letter gestures or resetting gestures denoted as $P_{Ges}(t)$, we stretch or compress them in the time domain dimension by factors ρ to generate new data spectrograms, denoted as $P_{Ges}(\rho t)$. Through experimentation, we determine the factors to be $\rho \in [0.80, 0.85, 0.90, 0.95, 1.00, 1.05, 1.10, 1.15, 1.20]$. With this data augmentation technique, each letter gesture or resetting gesture data is expanded into nine new gesture data instances. Fig. 7 displays the spectrograms of the letter gesture '*A*' multiplied by the factors 0.80, 1.00, and 1.20, respectively, illustrating the impact of the time-warping technique.

*2) **Data synthesis**:* In this Section, we identify the training dataset construction scheme for reconstructing word gesture data from letters and resetting gestures. Word gestures are composed of letter gestures, and because there is a writing order for the letter gestures within the word gesture, the letter gestures are sequentially connected in the time domain of the time-frequency map. Based on this observation, we design the specific training dataset construction scheme.

Prior to collecting pseudo-word gesture data, we examine the writing process of real word gestures and analyze the characteristics of the shift pattern generated by these gestures. As mentioned earlier, users may experience brief pauses when writing word gestures. These pauses are reflected in the spectrogram as interruptions in the Doppler frequency shift, particularly in the time domain dimension. Furthermore, in the absence of pauses, when users write these gestures fluently, the Doppler frequency shift patterns formed by two consecutive letter gestures may partially overlap or intersect in the time domain. Moreover, during the word writing process, users maintain a consistent writing speed and magnitude. This implies that the letter and resetting gestures used for synthesizing word gesture data should be written at similar speeds and magnitudes to ensure fidelity to real-word writing. By considering these observations, we can generate pseudo-word gestures that closely resemble the real word gestures.

To ensure that the writing speed and magnitude of the synthesized letters and resetting gestures closely match each other, we select the letters and resetting gestures required for a word from the same session of the data collection, using the same data augmentation factor ρ. During the synthesis process, we need to simulate two scenarios, namely, continuous writing and writing with pauses. Let's consider two spectrograms, P_1 and P_2, both representing letter gestures. First, we generate a random parameter Gap within the range of $[-25, 5]$. If Gap is negative, it simulates the occurrence of a pause during writing. In this case, we create a spectrogram $\tilde{P}$ consisting of $|Gap|$ STFT frames, where each pixel value is set to 0. Next, we concatenate the spectrograms P_1, $\tilde{P}$, and P_2 along the column dimension. In the case where Gap is zero, it signifies continuous writing, and we directly concatenate the spectrograms P_1 and P_2 along the column dimension. However, the processing scheme becomes more intricate when Gap is positive. We superimpose the last Gap STFT frames $P_1^{'}$ of spectrogram P_1 with the initial Gap STFT frames $P_2^{'}$ of P_2. For each pixel point, we select the larger value between the corresponding pixel points of $P_1^{'}$ and $P_2^{'}$, as depicted in Eq.(2).

$$P^{'}(x,y) = \begin{cases} P_1^{'}(x,y) & \text{if } P_1^{'}(x,y) > P_2^{'}(x,y) \\ P_2^{'}(x,y) & \text{otherwise} \end{cases} \tag{2}$$

where $P^{'}$ represents the new spectrogram obtained from the superimposed part, with the number of columns equal to $|Gap|$. The variables x and y denote the coordinates of the pixel points. It is important to note that when P_1 represents a resetting gesture and P_2 represents a letter gesture, the randomly generated parameter Gap can only be positive.

TABLE I: The word template encompasses five parts of speech.

Part of speech	Words
Verb	AM, ARE, IS, CAN, DO, HELLO, MAKE, SAY, GIVE, PUT
Articles	THE
Noun	FAMILY, PERSON, JOB, QUIT, WORLD, ZERO
Pronouns	YOU, THAT
Adverb	HOW, WHAT, WHERE, WHY, BUT, NEXT

When synthesizing word gestures, we need to select a word as a template. To ensure diversity, we have chosen 25 commonly used words from the Corpus of Contemporary American English (COCA) as templates for synthesizing word gestures. These words are selected based on the following criteria: containing all 26 letters of the alphabet, sufficient word length, and their ability to form meaningful sentences. The 25 words are grouped according to their main parts of speech and are summarized in Table I.

Fig. 9 illustrates the synthesis process of "AM", where the letters "A" and "M" undergo a certain degree of stretching and are combined with the corresponding GAP to synthesize the word "AM".

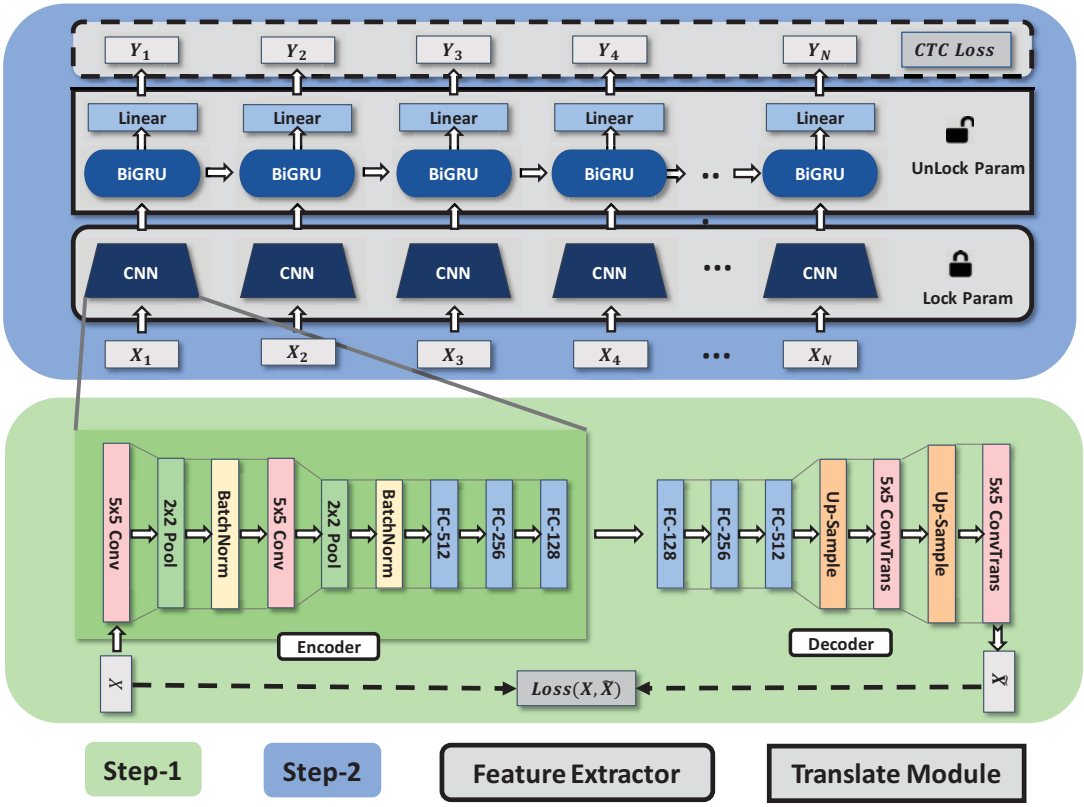


Fig. 8. The model architecture. The feature extractor retains only the encoder from the autoencoder. During the training phase, the parameters of the pre-trained feature extractor are frozen, and only the parameters of the translation module are adjusted.

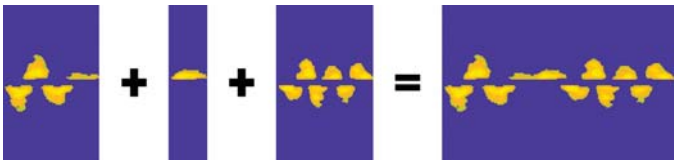


Fig. 9. "AM" is synthesized by combining "A" and "M" with a pause

E. Model Training

In this subsection, we provide a detailed description of the system modeling and the design of the cross-domain training strategy. Fig. 8 illustrates the network framework that we have designed. The framework encompasses our model composition as well as the cross-domain training strategy.

*1) **Model design**:* We apply an analogous transformation to convert the continuous gesture input task into an NMT task [43], aiming to achieve continuous gesture input. The NMT task can be further divided into two subtasks: a) encoding and embedding the source sequence from the original representation space to the feature space, which corresponds to the feature extraction module; b) transcribing the encoded and embedded feature vector sequence with context information, which corresponds to the translation module.

In this system, the source sequence is the spectrogram frame sequence, which can be considered a sequence of images. We employ a CNN to encode and embed the original sequence. We design the feature extractor based on the LeNet model, taking into account factors such as model size, recognition performance, and response time. Although there are other classification models suitable for mobile devices, such as AlexNet, MobileNet [44], and ResNet, they often have specific requirements on the size and complexity of the training dataset. Actually, LeNet proves to be a more suitable choice for our task, as further elaborated in Sec. V-B4.

In our task, the source sequence aligns with the target sequence, thus necessitating precise labeling of the source sequence. To tackle the source sequence labeling difficulty, we employ CTC. Specifically, we utilize the GRU model to construct the Translation Module. Although there are other classification models suitable for mobile devices, we find that Bi-GRU yields the best results after comparing it with Bi-RNN and Bi-LSTM, as detailed in Sec. V-B5. Following the Bi-GRU layer, we include a linear layer that outputs a letter or null label ϵ for each spectrogram frame in the source sequence. Considering the size of the translation module, we opt to use an encoder instead of a decoder during its design.

*2) **Cross-domain training strategy design**:* The data synthesis approach significantly reduces the cost associated with data collection. A crucial question that arises is whether new users can effectively utilize UltraWrite for gesture recognition without providing any samples. In conventional deep learning modeling, it is generally assumed that the training dataset (source domain) and the test dataset (target domain) are independent and identically distributed (i.i.d.). However, this

assumption often fails to hold in real-world application scenarios. In our task, the source domain consists of a synthetically generated dataset, while the target domain comprises real-world data collected from users. This discrepancy introduces a challenge for the model's cross-domain recognition capability, as the distributional shift between the two domains may hinder generalization performance.

We observed that the primary difference between the source and target domains lies in the Doppler frequency bandwidth caused by gestures. This difference can be explained by two factors. First, when different users perform the same gesture, slight variations occur in the movement patterns of the entire arm during the writing process due to uncontrollable factors such as palm size, arm length, and finger thickness. These slight variations in movement are captured through the scattering properties of sound waves, resulting in differences in the width of the frequency bandwidth. Additionally, different users may execute certain strokes differently when performing the same gesture. However, it is important to note that despite these differences, the source and target domains still belong to the same homogeneous data domain, and the Doppler shift similarity of the same word gesture remains high. In summary, although there are some minor differences in the Doppler frequency shift features caused by different users performing the same gesture, the overall trend remains highly similar. Therefore, we need to ensure that the model focuses on the Doppler shift trend in gesture data rather than on minute details.

In conventional encoding recognition domain adaptation training strategies, the feature extraction module, decoder, and classifier are typically jointly trained. Nevertheless, joint training of these sub-modules can introduce coupling relationships that may hinder each sub-module from focusing on its specific task. Based on this consideration, we devise a cross-domain training strategy that involves training the feature extractor and the translation module separately, effectively decoupling the modules. To enable the feature extractor to capture common features, we train it using an autoencoder framework to accomplish source data reconstruction, and the loss function is defined by Eq.(3).

$$L_{rec} = \frac{1}{n}\sum_{i=1}^{n}(X_i - \tilde{X}_i)^2 \tag{3}$$

where n represents the count of spectrogram frames, i denotes the spectrogram frame index, and x refers to the reconstructed spectrogram frame. To prevent bias in the feature extractor during translation module training, we fix its parameters and adjust only those of the translation module, as shown in Fig. 8. The loss calculation during the training of the translation module is obtained by Eq.(4).

$$L_{CTC} = \sum_{(X,Y)\in B} -\log P(Y|X) \tag{4}$$

where X and Y respectively represent the input sequence and the corresponding label of the same batch B.

3) ***Bayesian spelling correction****:* We employ Bayesian inference to correct the model's outputs. Let's assume that a word gesture's spectrogram consists of *T* spectrogram frames. As we know, the model generates a 27-dimensional likelihood vector for each spectrogram frame. This vector includes a blank character ϵ, as required by the CTC encoding rule, and 26 letter characters. We denote the model's output probability matrix as *M*. Based on *M*, we obtain a temporary word $\tilde{W} = \tilde{w}_1, \tilde{w}_2, \cdots, \tilde{w}_T$. To maximize the probability of $\tilde{W}$, the character with the highest probability in the vector is typically selected as the label for the current frame. Its probability can be calculated by Eq.(5).

$$P(\tilde{W}) = \prod_{t=1}^{T} \max_{0\le i\le 26} M_{t,i} \tag{5}$$

$$\tilde{w}_t = l_i = \arg\max_{0\le i\le 26} M_{t,i} \tag{6}$$

where *i* represents the index of the probability vector for each frame, and l_i denotes the character corresponding to index *i* with the highest probability. However, errors in the output of certain frames can lead to misspellings in the final word output W, including letter substitutions, deletions, or insertions. To obtain the correct word, we introduce word frequency into the calculation of the current frame probability.

According to the Markov chain assumption, the output of several previous frames should be taken into account concurrently when determining the label for the current frame. This consideration, which involves only the labels from the previous frame, is referred to as a 2-gram dependency. Let *i* and *j* represent the indices corresponding to the labels of the current and previous frames, respectively. The transition probability $P(j, i)$ can be derived by calculating the corresponding frequencies within a large dataset of words, as follows:

$$P(l_i|l_j) = \frac{C_{w_{t-1},w_t}}{C_{w_{t-1}}} = \frac{C_{l_j,l_i}}{C_{l_j,A\to Z}} \tag{7}$$

where C_{l_j,l_i} denote the frequency of the letter combination (l_j, l_i). $C_{l_j,A\to Z}$ denote the frequency of the letter combination that the first letter $\tilde{w}_{t-1}$ is l_j. When $t-1$ equals 1, meaning that $\tilde{w}_{t-1}$ is the first letter of the temporal word $\tilde{W}$, the corresponding $P(l_i|l_j)$ reduces to $P(l_i)$. There is a special case where the transition probability is set to 1 when one of the characters in the current or previous frame is blank ϵ. Consequently, Eq.(5) and Eq.(6) can be simplified as follows:

$$P(\tilde{W}) = \prod_{t=1}^{T} \max_{0\le i,j\le 26} P(l_i|l_j) \times M(t,i) \tag{8}$$

$$\tilde{w}_t = l_i = \arg\max_{0\le i,j\le 26} P(l_i|l_j) \times M(t,i) \tag{9}$$

Following the aforementioned procedure, probabilities of mapping the input spectrogram frame sequence to different candidate words of the same length T are obtained. Based on these probabilities, we select the top-ranked temporary words $\tilde{W}$ as candidates. Once the temporary word $\tilde{W}$ is obtained, we generate the final output word $W = w_1, w_2, \cdots, w_N$ $(N \le T)$ in accordance with the CTC encoding rule.

IV. Implementation And Experiments

A. Implementation

We implement UltraWrite as an Android application on a Samsung Tab S2 whose speaker emits sinusoidal acoustic wave at 19 KHz and the microphone receives echo signals with a sampling rate of 44.1 KHz. After receiving the echo signals, the system processes signals following the pipeline as shown in Fig. 2. The deep learning model is trained with PyTorch on a a server with an Intel(R) Xeon(R) Platinum 8260 CPU and NVIDIA GeForce RTX 2080Ti GPU. After that, the trained model is packed into a pt file and deployed on the tablet. In this work, we have only tested UltraWrite with a single device owing to the consideration that most commercial smart devices are capable of emitting and receiving 19 KHz acoustic signals, and share similar frequency response. However, speakers on different devices indeed possess some differences in frequency response and can easily have dips or spikes at various output ranges. Such differences may impact the system performance. We leave this evaluation as one of our future works.

B. Experimental Setup

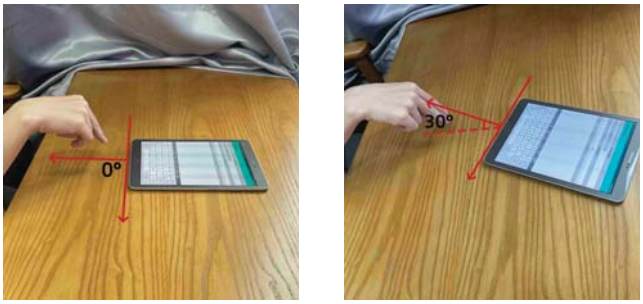
We randomly select 10 volunteers (denoted by V_1 to V_{10}), including one female and nine males, to participate in the data collection process on campus. The ages of the volunteers range from 22 to 25 years old, and all are right-handed. Prior to conducting the specific experiments, we provide the volunteers with gesture writing specifications.

Data collection takes place in a quiet laboratory environment with noise levels below 50 dB. Specifically, we placed the data collection equipment flat on the table, facing the user's microphone, and then set up a 10×16 cm^2 square centimeter matrix in a vertical direction about 5 centimeters away from the speaker, where the user accomplishes the writing within this range. Fig. 10 illustrates the experimental setup. During the writing process, the user remains quiet, and the equipment remains stationary. Finally, we collect the letter gesture dataset and word gesture dataset in this environment. We invite only one volunteer (V_1) to collect the letter dataset. The volunteer writes from *'A'* to *'Z'* in order, forming a round of letter data. In the same round, the volunteer writes at a similar speed as much as possible between gestures (letter or reset gesture) with an interval of approximately 1 second. In the end, we obtain 15 rounds of a total of 450 gestures, including 390 letter gestures and 60 reset gestures.

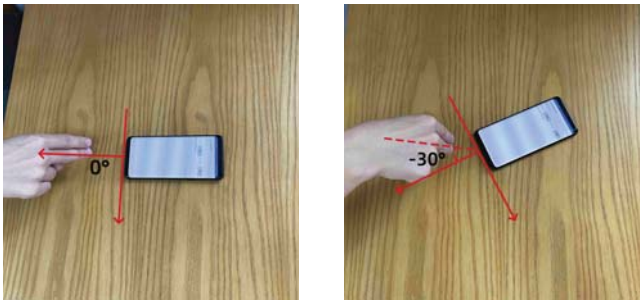
To collect the testing dataset, we invite 10 volunteers (including V_1) to participate in the collection of word gesture data. The data collection task is based on 25 words shown in Table I, with each word written 15 times. After collecting the data, we perform a simple check to remove some samples that are unusable due to device or collection process anomalies. In the end, each volunteer collects an average of 351 word gesture samples.

V. Evaluation

In this section, we evaluate UltraWrite's performance from different aspects. We employ word accuracy (W-Acc) as the main metric which denotes the ratio of the sample count that is correct to the total sample count.



(a) tablet 0° (b) tablet 30°



(c) phone 0° (d) phone −30°

Fig. 10. The experimental setup

$$W - Acc = \frac{Count_{True}}{Count_{All}} \tag{10}$$

We also use char error rate (CER) as the other evaluation metric which is defined by Eq.(11).

$$CER = \frac{N_{sub} + N_{del} + N_{ins}}{N_{GroundTruth}} \tag{11}$$

where N_{sub}, N_{del}, and N_{ins} denotes the number of required substitutions, deletions, and insertions, respectively. The top-k terms are considered when calculating the criterion, as the correction process produces the highest probability candidates.

A. Overall Performance

*1) **User-independent recognition**:* In the user-independent test, we train the model using the synthetic dataset, which is described in Sec. III-D, and evaluate the model on a test dataset of word gestures written by volunteers in real-world scenarios, the output is corrected using Bayesian correction. The output is then corrected using Bayesian correction. Fig. 11 displays the results of the cross-user case. Among the volunteers, we collect the isolated gesture data used to synthesize the word gesture dataset from V_1. Therefore, V_1 is a non-cross-user case with a top-1 CER of 0% after correction by a language model. The remaining volunteers were involved in the cross-user tests. When considering only cross-user cases, the average top-1, top-3, and top-5 W-Acc were 99.29%, 99.67%, and 99.74%, respectively, and the corresponding average CER were 0.38%, 0.24%, and 0.20%, respectively. On average, there were 8 word recognition errors per volunteer. Among them, V_9 had

the worst test results with the top-1, top-3, and top-5 W-Acc of 98.22%, 98.81%, and 99.11%, respectively, and the corresponding CER of 1.14%, 0.89%, and 0.73%, respectively. This is because V_9's fingers may sway while writing, yet the system still maintains a relatively high accuracy.

This experiment demonstrates where the proposed system in this work achieves superior recognition performance in cross-user scenarios. It is observed that the writing trajectories of "I", "J", "L", and "T" in Fig. 5 are similar. However, when our recognition unit is the entire word, this similarity does not impact the recognition accuracy. This is because the system leverages relationships between letters within words for recognition. Additionally, Bayesian modeling is applied to refine the output. Simultaneously, the excellent cross-user performance ensures that new users can have a good experience with UltraWrite without any additional cost.

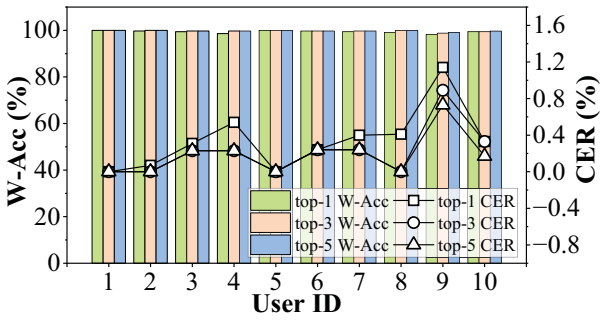


Fig. 11. The recognition performance in cross-user scenarios

*2) **Unseen word recognition**:* When users use the system, there is a possibility that the system has not seen the word gestures during the model training process. To evaluate the system's generalization ability to unseen words, we replace the training dataset while retaining the test dataset. This allows the model to recognize words it has not encountered during training. To this end, we design a data template that is completely different from the words in Table I to reconstruct a synthetic dataset for model training. Specifically, we extract all 2-gram and 3-gram letter combinations of the words in Table I, and modify words of length 2 or 3 by substituting, deleting, or inserting letters. Then, we combine the 2-gram and 3-gram letter combinations to form n-gram ($n \in [2, 9]$) letter combinations, which serve as the template for synthesizing the training dataset, as described in Sec. III-D. During the testing phase, the input word gestures correspond to the words in Table I. This test only changes the training dataset and not the other settings. As shown in Fig. 12, the performance of the system on unseen word gestures is reduced. However, the top-5 W-Acc of all volunteers can still reach over 93%, and the average top-1, top-3, and top-5 W-Acc are 83.04%, 91.65%, and 94.37%, respectively. These results indicate that the system still maintains a reasonable recognition accuracy for unseen word gestures, further affirming the rationality of our system design. Additionally, the experiment validates the applicability of UltraWrite to unseen words, theoretically enabling the recognition of any unseen word.

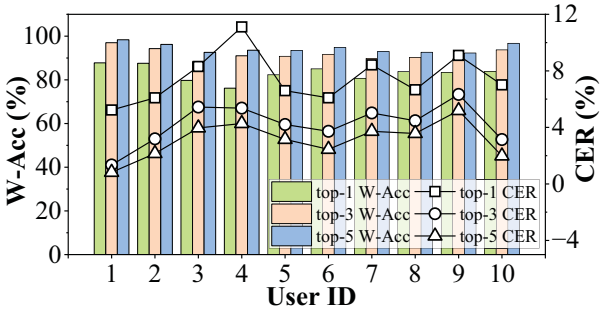


Fig. 12. The recognition performance in unseen word scenarios

B. Ablation Study

In this section, we conduct a series of ablation experiments. The cross-domain training strategy proposed in the Sec. III-E2 improve the model's generalization and the synthesized dataset through the method proposed in Sec. III-D, significantly reducing user costs. Additionally, spelling correction effectively enhances W-Acc. Finally, to further lighten the system while maintaining good performance, we carefully select the feature extractor and the translation module.

*1) **Impact of cross-domain training strategy**:* In this part, we evaluate the effectiveness of the cross-domain training strategy proposed in Sec. III-E2. When testing without the cross-domain training strategy, the training dataset consists of synthetic word gesture dataset (D_9), and only the CTC loss is used as the loss function during the model training.

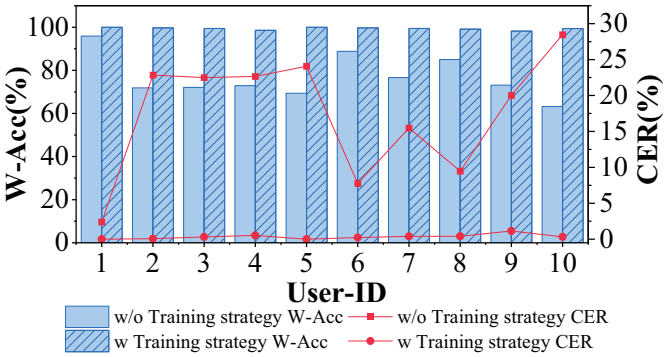


Fig. 13. The recognition performance with or without cross-domain training strategy

Without employing our training strategy, the feature extractor does not utilize the reconstruction encoder and is trained together with the translation module. Compared to the results obtained using the cross-domain training strategy, the system performance without the strategy significantly declines, as shown in Fig. 13. It is noted that V_1 is a non-cross-user test, hence exhibiting minor performance degradation. For the remaining users undergoing inter-user testing, there is a substantial performance decline. The average top-1 W-Acc decreases by 22.45%. Additionally, we calculate the difference in CER for each user between scenarios with and without the cross-domain training strategy. With the cross-domain training strategy, the average top-1, top-3, and top-5 CER of the system

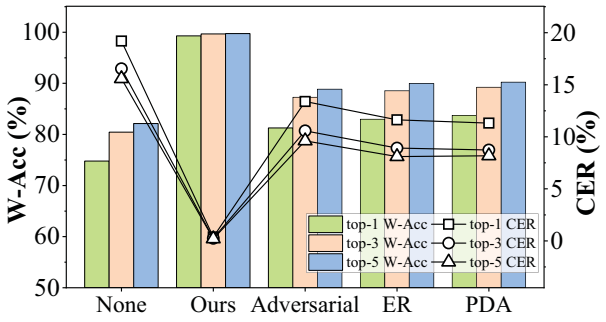


Fig. 14. The recognition performance with different cross-domain training strategies

with cross-domain training strategy decreases by 17.16%, 14.86%, 14.01%, respectively.

In addition, we compare the effects of three data augmentation strategies: Adversarial Domain Adaption [45], Encoding Reconstruction (ER) [46], and Prior Distribution Alignment (PDA) [47], on the cross-user recognition performance of our system. We conduct tests on nine volunteers (V_2 to V_{10}) and the results are presented in Fig. 14. The results indicate that all the three strategies can improve the system's recognition performance. However, the improvement achieved by these strategies is still inferior to that of the cross-domain training strategy proposed in this work. We believe that Adversarial [45] may cause the extracted features to lose original information, which can lead to encoding errors in the translation module. The PDA [47] also faces similar problems, albeit to a lesser degree. On the other hand, the ER [46] is more similar to the strategy proposed in this work. The key difference lies in our proposed cross-domain training strategy fusion's decoupled training approach. These results underscore the necessity and effectiveness of our fusion decoupling training approach. In conclusion, our proposed cross-domain training strategy yields the most significant improvement in the system's recognition performance in cross-user scenarios. Furthermore, this strategy does not require new users to provide personal samples, making it more user-friendly and having greater practical value. These results effectively demonstrate the efficacy of our cross-domain training strategy.

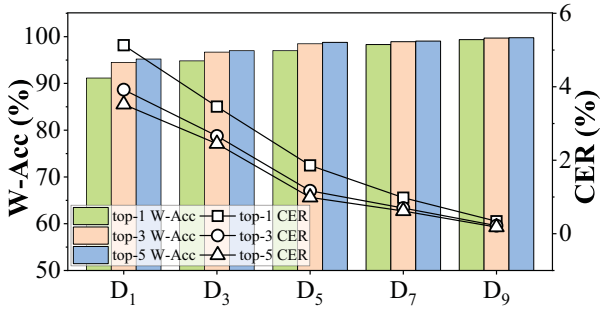


Fig. 15. The recognition performance when training with synthetic datasets of different scales. D_i represents training dataset ($i = 1, 3, 5, 7, 9$), where i indicates that the dataset is increased by i times.

*2) **Impact of training dataset construction**:* In this part, we evaluate the effectiveness of the train dataset construction scheme described in Sec. III-D. We determine the scale of the synthesized dataset by setting the scaling factor ρ. A value of 1.0 for ρ means that data augmentation is not used during dataset synthesis. We expand the value range of ρ by 0.5 increasing two values at a time. The results are shown in Fig. 15. When D_1 is used for training, the top-1, top-3, and top-5 CER are 5.13%, 3.92%, and 3.52%, respectively. The corresponding W-Acc is 91.15%, 94.48%, and 95.22%, respectively. These results indicate that the use of only the synthesized word gesture dataset as the training dataset is feasible. With the addition of data augmentation, the recognition performance of the system improves, especially with the wider value range of factor ρ. Our system achieves the best recognition performance in D_9. The top-1, top-3, and top-5 CER are 0.34%, 0.22%, and 0.18%, respectively, and the corresponding W-Acc is 99.36%, 99.71%, and 99.77%, respectively.

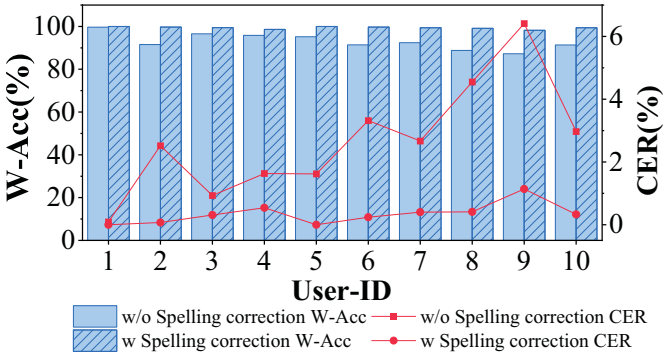


Fig. 16. The recognition performance with or without spelling correction

*3) **Impact of spelling correction**:* In this section, we evaluate the spelling correction proposed in Sec. III-E3. Keeping other settings unchanged, we apply spelling correction and no spelling correction to the words predicted by the network, respectively, as shown in Fig. 16. The average top-1 W-Acc without correction is 92.99%, with a CER of 2.67%; with correction, the average top-1 W-Acc is 99.36%, with a CER of 0.34%. Compared to no spelling correction, the average W-Acc after correction has increased by 6.36%, demonstrating the effectiveness of the spelling correction module.

*4) **Impact of feature extractor**:* In UltraWrite, we design a feature extractor based on LeNet that can effectively extract abstract features from the raw spectrogram frames. We also examine different CNN models including AlexNet [48], MobileNet-v3 [44], and ResNet-18 [49] as the feature extractors on the system's performance. This evaluation only changes the backbone of the feature extractor without any other settings. Fig. 17 presents the evaluation results, which show that the overall recognition performance is best when using LeNet as the feature extractor. The top-1, top-3, and top-5 W-Acc are 99.36%, 99.71%, and 99.77%, respectively. As the complexity of the model increases, the system's recognition performance decreases. As analyzed in Sec. III-E,

the dataset used in this work is relatively simple. When the feature extractor is too complex, the features will be more inclined towards the training dataset, resulting in overfitting. Additionally, LeNet has the smallest model size, at only 0.27 MB. The sizes of AlexNet, MobileNet-v3, and ResNet-18 are 213 times, 16 times, and 42 times larger than LeNet, respectively. Considering the test results and the model size, it is most reasonable to opt for LeNet as the basis for designing the feature extractor.

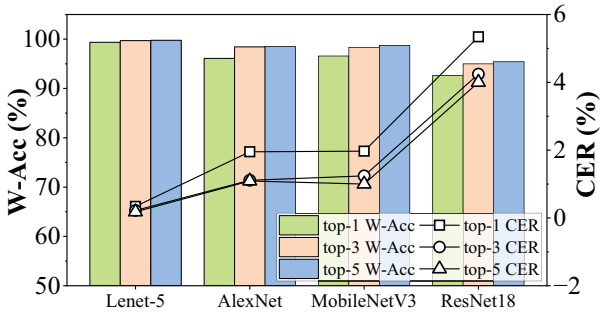


Fig. 17. The recognition performance of different networks acting as feature extractor

*5) **Impact of translation module**:* In UltraWrite, we design a translation module based on Bi-GRU to encode the feature sequence extracted from the feature extractor in context. In this section, we also conduct evaluation of the impact of using different models as the translation module, including RNN, LSTM, GRU, Bi-RNN, Bi-LSTM, and Bi-GRU. For a fair comparison, the layers of these models are all set to be 2, and the bidirectional parameters are set to be true. Eventually, we only change the backbone of the translation module in this test, leaving other settings unchanged. The evaluation result is presented in Fig. 18, which indicates that the bidirectional networks can capture context information better than the corresponding unidirectional networks. After statistical analysis, compared with the corresponding unidirectional networks, the top-1, top-3, and top-5 average W-Acc of the three bidirectional networks increase by 20.91%, 13.40%, and 12.20%, respectively. Among them, Bi-GRU and Bi-LSTM have the best performance, with the top-1 W-Acc of 99.36% and 99.03%, respectively, with a negligible difference between them. However, in terms of model size, Bi-GRU is only 6.70 MB, while Bi-LSTM is 8.93 MB. After combining the test results and the model size, it is the most reasonable choice to design the translation module based on Bi-GRU.

C. System Robustness

To investigate the impact of sensing distance, angle, background noise and device on the results, we invite three volunteers (V_2, V_3, and V_5) to participate in the testing process.

*1) **Impact of distance**:* To evaluate UltraWrite's robustness to different distances, we conduct three sets of contrast experiments, including: 1) *Close*, 5 cm; 2) *Medium*, 15 cm; and 3) *Far*, 25 cm. The experimental settings are illustrated in Fig. 19a. Volunteers are asked to write word gestures at different distances as described above. The evaluation results

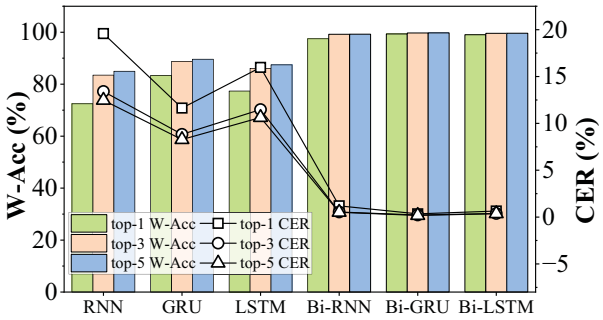
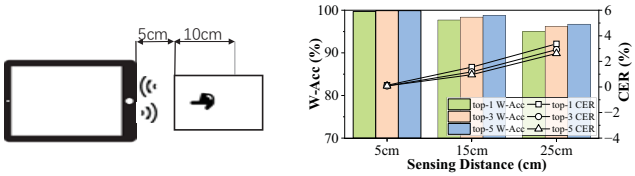


Fig. 18. The recognition performance of different networks acting as translate module

are shown in Fig. 19b. Our findings show that the system performance decreases as the sensing distance increases. When the perceived distance increases to 15 cm, the system performance is slightly degraded, with the top-1, top-3, and top-5 W-Acc decreasing by 2.01%, 1.53%, and 1.12%, respectively. When the perceived distance increases to 25 cm, the top-1, top-3, and top-5 W-Acc decrease by 4.75%, 3.72%, and 3.18%, respectively. Nevertheless, the overall W-Acc remains above 95%, indicating that UltraWrite has great robustness to sensing distance.

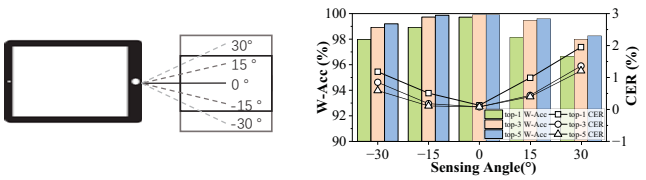


(a) Distance setup (b) Experimental result

Fig. 19. The impact of different distances

*2) **Impact of angle**:* Due to differences in users' writing habits, there may be variations in the writing angle. To evaluate UltraWrite's robustness to sensing angles, we conduct five sets of contrast experiments based on the sensing angles that may be encountered in practical usage. The sensing angles were set to $-30°$, $-15°$, $0°$, $15°$, and $30°$, respectively, as shown in Fig. 20a, where the writing range moves to the left indicating a negative angle. We then instruct volunteers to perform word gestures at different angles.

The result presents in Fig. 20b, demonstrate that the system's recognition performance gradually declines as the angle increases. Notably, the performance degradation is more significant at positive angles compared to negative angles. This phenomenon can be attributed to the placement of the Tab S2 speakers, which are positioned on the positive angle side. When users perform gestures at positive angles, the relative motion between their fingers and the device becomes more pronounced, leading to greater variability in the input signals. The system performs the worst when the sensing angle is $30°$, with the top-1, top-3, and top-5 W-Acc being 96.66%, 97.99%, and 98.26%, respectively. Nevertheless, UltraWrite demonstrates robustness to sensing angles, with an overall W-Acc above 96%.



(a) Angle setup (b) Experimental result

Fig. 20. The impact of different angles

*3) **Impact of noise**:* Due to the ambient noise level in everyday environments being below 70 dB, we conduct two sets of comparison experiments with noise levels of $[50, 60]$ dB and $[60, 70]$ dB, respectively. To control the noise level, we play various audio files of different kinds of noise, such as daily talking, speech, music, rain sounds, etc., at controlled volume levels. Then, we ask volunteers to perform word gestures in different noise. We summarize the results according to the noise level, as shown in Fig. 21. It is apparent that the CER and W-Acc in the lab environment are very close to those in the noisy environment, and the difference is negligible. After analysis, we found that the frequency of environmental noise mainly concentrates below 1 KHz, which can be effectively removed after the signal preprocessing in Sec. III-B.

*4) **Impact of devices**:* To account for potential differences in data resulting from speaker and microphone locations, as well as hardware issues, UltraWrite's universality across different smart devices is evaluated through three control experiments involving Samsung Tab S2 (used solely for gesture collection), Samsung Galaxy S9, and Xiaomi MI9. The results, presented in Fig. 22, reveal that Xiaomi MI9 achieves top-1, top-3, and top-5 W-Acc of 96.93%, 98.86%, and 99.32%, respectively, with a slight decrease compared to Samsung Tab S2. Conversely, the performance of Samsung Galaxy S9 shows significant degradation, with top-1, top-3, and top-5 W-Acc of 91.51%, 94.66%, and 95.21%, respectively. These results demonstrate that performance variations may occur due to sensor differences.

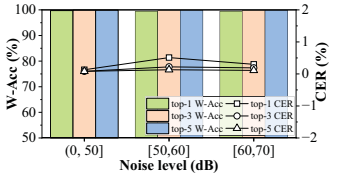


Fig. 21. The impact of different noise levels

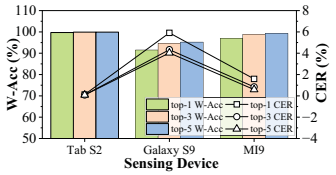


Fig. 22. The impact of different devices

D. System Running Performance

In this section, we conduct a comprehensive performance evaluation of UltraWrite, considering metrics such as response time, CPU and memory utilization, and power consumption. We randomly selected participant V_1 for the word gesture input experiment, where each word gesture is written five times.

*1) **Response time**:* The response time of UltraWrite primarily consists of data processing, word prediction, and spell correction. To eliminate the influence of performance disparities among devices, we conduct tests on Samsung Tab S2, Samsung Galaxy S9, and Xiaomi MI9, as depicted in Fig. 23. The results reveal that the word prediction phase constitutes the largest proportion of the total processing time. This is attributed to our translation module based on GRU, which requires the current time step to calculate the next time step and cannot be parallelized. Additionally, data processing accounts for a significant portion of the response time, as we need to handle empty signal segments. Tab S2 exhibits the longest response time, recognizing only 18 word gestures per minute, while Galaxy S9 and MI9 average approximately 31 and 21 word gestures per minute, respectively. Finally, it is evident that longer input words result in longer response times.

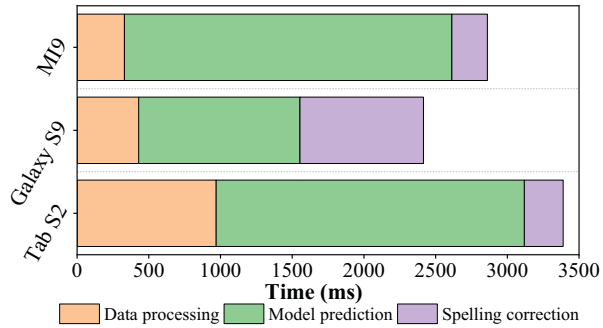


Fig. 23. UltraWrite's response time

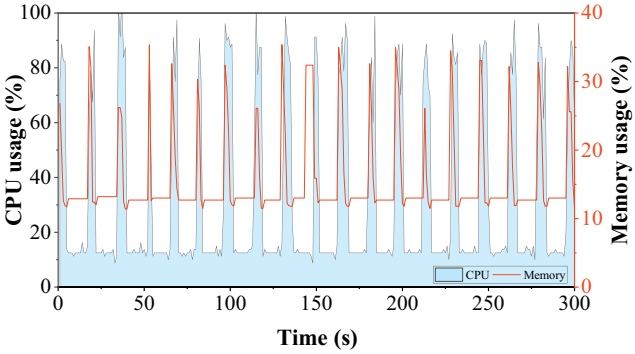


Fig. 24. UltraWrite's CPU and memory occupation

*2) **CPU and memory occupation**:* UltraWrite is deployed on Android, allowing the use of the 'top' command to query resource utilization during system operation. We connect Tab S2 to a PC, enabling the participant to engage in continuous gesture input for five minutes while querying CPU and memory usage every second. The results, as depicted in Fig. 24, indicate that during the gesture input and signal processing stages, CPU and memory usage remain relatively low, approximately 12.5% and 13%, respectively. However, in the word prediction phase, there is a sudden increase in both CPU and memory usage, with the CPU reaching 100%, and memory usage remaining below 36% (memory usage positively correlates with the length of input word gestures). In conclusion, the system consumes substantial device resources during the prediction phase.

TABLE II: Comparison with others continuous gesture input systems

Work	Modality	Training users[1]	Samples[2]	Seen word		Unseen word	
				single-user	cross-user	single-user	cross-user
Signspeaker [33]	IMU	16	11680	98.96%	89.30%	-	-
DeepSLR [38]	IMU+EMG	34	20400	89.20%	-	-	-
MyoSign [34]	IMU+EMG	15	7500	-	93.10%	92.40%	-
WriteAS [37]	IMU	12	22500	97.60%	84.60%	93.40%	83.80%
WearSign [35]	IMU+EMG	15	10375	99.71%	94.50%	91.40%	89.20%
SonicASL [18]	Acoustic	8	9600	90.60%	-	-	-
Ours	**Acoustic**	**1**	**450**	**100%**	**99.74%**	**98.37%**	**93.92%**

Training users[1]: The number of training users.
Samples[2]: The training data samples for constructing a gesture recognition system.

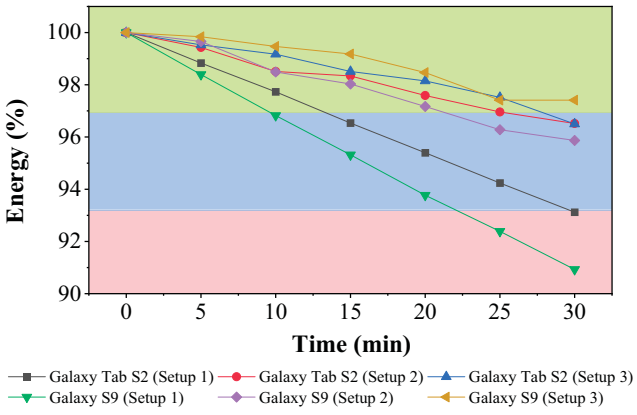


Fig. 25. UltraWrite's battery consumption

*3) **Battery consumption**:* We also test the real-time power consumption of UltraWrite's on the Samsung Tab S2 and Samsung Galaxy S9 (Setup 1). Participants perform continuous gesture input for 30 minutes, with the battery level recorded every 5 minutes. Additionally, we track power consumption for the UltraWrite application being open but with no other actions performed (Setup 2) and for the device screen being simply on (Setup 3). As shown in Fig. 25, after 30 minutes of continuous operation, the Tab S2's battery level drops to 90.93%, with an average decrease of 1.51% every 5 minutes. The Galaxy S9 maintains a battery level above 90%. Moreover, when the device is idle or the app is open but not in use, the battery consumption differs by no more than 1.5%. Compared to Setup 2 and Setup 3, the power consumption during recognition in UltraWrite is acceptable. Under ideal conditions, the proposed system can run continuously for approximately 5.5 hours.

E. Comparison Against Related Approaches

Table II presents a comparison with state-of-the-art related works in terms of recognition accuracy and the dataset cost of training models. As we can see from Table II, only Signspeaker [33], WriteAS [37], and WearSign [35] achieve comparable recognition accuracy with UltraWrite. But they require more than 13 times of training data, while WearSign [37] collects 6,000 word samples and 4,375 sentence samples. Other works have much lower recognition accuracy and a heavier burden of collecting training data. The above results verify that our method shows great superiority in recognition, cost of system construction, and cross-user adaptation capability compared with existing works.

F. User Study

The user experience of continuous gesture recognition with UltraWrite is crucial. In this section, we assess the system's user experience through the implemented system prototype and Standard Usability Scale (SUS) interviews. SUS encompasses evaluations of effectiveness, usability, and satisfaction, representing the main aspects of overall system assessment. Fig. 26 presents the ten statements of SUS, along with histograms and mean scores for each statement. Following SUS guidelines, positive and negative statements are alternately arranged. When users complete the scale, the ten statements are arranged in the order of S1 to S10.

Overall, all volunteers consider UltraWrite to be a viable system. However, one volunteer still holds a negative attitude towards the system, stating, 'I acknowledge the system's recognition accuracy, but the burden of learning the writing standards before using it makes me feel hesitant.' Nevertheless, the score for S4 is 1.8, indicating that the learning cost is not significant, and standardized writing rules are necessary. UltraWrite's SUS usability score is 73, categorizing it as 'GOOD' in the scoring scale, and users are inclined to use the system.

VI. Discussion

A. System Performance

The system has a high response time mainly because the translation module is based on GRU and can only perform serial computation, resulting in slow calculation speed. In the future, we will try to use extended convolutional neural networks instead of GRU as the translation module.

B. Cross-Device Problem

The performance of the system is significantly influenced by different devices, which is caused by the inherent hardware differences that lead to variations in data of the same gestures. Domain adaptation methods may not be effective in addressing this data discrepancy issue, and therefore, few-shot learning techniques could be explored as a potential solution.

1	2	3	4	5	Mean
S1. I think that I would like to use this system frequently.					
		1	7	2	4.1
S3. I think the system is easy to use.					
1		2	6	1	3.6
S5. I found this system made it easier to activate smart actions(events)					
		2	4	4	4.2
S7. I would imagine that most people would learn to use this system very quickly.					
1			4	5	4.2
S9. I felt very confident using the system.					
		3	4	3	4.0
S2. I found the system unnecessarily complex.					
2	2	6			2.4
S4. I think that I would need the support of a technical person to be able to use this system.					
3	6	1			1.8
S6. I thought the system would be obtrusive in my daily life.					
5	3	2			1.7
S8. I found this system very cumbersome to use.					
2	5	2	1		2.2
S10. I needed to learn a lot of things before I could get along with this system.					
1	4	3		2	2.8

Fig. 26. The ten statements of the modified Standard Usability Scale (SUS), as well as the histograms and means of the scores for each statement (S1∼S10). Each statement was scored from 1 (indicating 'strongly disagree') to 5 (indicating 'strongly agree'). For positive statements (S1, S3, S5, S7, S9), higher scores are better. For negative statements (S2, S4, S6, S8, S10), lower scores are better.

C. Gesture Discriminability

In this paper, we only use a pair of audio transceivers for gesture recognition, resulting in low recognition rates for gesture actions with similar writing trajectories, such as 'X' and 'Q'. In the future, it is possible to try using a combination of multiple audio transceivers to increase the dimensionality of gesture information captured by the system, making similar gestures distinguishable.

VII. Conclusion

Motivated by reducing the high cost of constructing an acoustic gesture input system which supports continuous input and good cross-user performance, we perceive word gestures based on the Doppler effect, decompose them into letter gestures for acquisition, and use data augmentation methods to reconstruct word gestures, effectively reducing system construction costs. We propose a network model that can recognize continuous gestures by combining CTC. To address the cross-user problem, we introduce an encoding reconstruction cross-domain method to solve the problem of decreased cross-user recognition performance. In summary, our proposed system is cost-effective and can achieve a high recognition rate for continuous gesture input without providing samples from new users, and can be deployed on devices such as smart glasses, headphones, and watches.

References

[1] J. Pirker, M. Pojer, A. Holzinger, and C. Gütl, "Gesture-based interactions in video games with the leap motion controller," in *ACM CHI*. Springer, 2017, pp. 620–633.

[2] A. D'Eusanio, S. Pini, G. Borghi, A. Simoni, and R. Vezzani, "Unsupervised detection of dynamic hand gestures from leap motion data," in *CVPL ICIAP*. Springer, 2022, pp. 414–424.

[3] R. Gao, W. Li, Y. Xie, E. Yi, L. Wang, D. Wu, and D. Zhang, "Towards robust gesture recognition by characterizing the sensing quality of wifi signals," *ACM IMWUT*, vol. 6, no. 1, pp. 1–26, 2022.

[4] R. Gao, M. Zhang, J. Zhang, Y. Li, E. Yi, D. Wu, L. Wang, and D. Zhang, "Towards position-independent sensing for gesture recognition with wi-fi," *ACM IMWUT*, vol. 5, no. 2, pp. 1–28, 2021.

[5] C. Wang, J. Liu, Y. Chen, H. Liu, L. Xie, W. Wang, B. He, and S. Lu, "Multi-touch in the air: Device-free finger tracking and gesture recognition via cots rfid," in *IEEE INFOCOM*. IEEE, 2018, pp. 1691–1699.

[6] C. Liang, C. Hsia, C. Yu, Y. Yan, Y. Wang, and Y. Shi, "Drg-keyboard: Enabling subtle gesture typing on the fingertip with dual imu rings," *ACM IMWUT*, vol. 6, no. 4, pp. 1–30, 2023.

[7] K. Guo, H. Zhou, Y. Tian, W. Zhou, Y. Ji, and X.-Y. Li, "Mudra: A multi-modal smartwatch interactive system with hand gesture recognition and user identification," in *IEEE INFOCOM*. IEEE, 2022, pp. 100–109.

[8] L. Wang, X. Zhang, Y. Jiang, Y. Zhang, C. Xu, R. Gao, and D. Zhang, "Watching your phone's back: Gesture recognition by sensing acoustical structure-borne propagation," *ACM IMWUT*, vol. 5, no. 2, pp. 1–26, 2021.

[9] P. Wang, R. Jiang, and C. Liu, "Amaging: Acoustic hand imaging for self-adaptive gesture recognition," in *IEEE INFOCOM*. IEEE, 2022, pp. 80–89.

[10] Y. Wang, J. Shen, and Y. Zheng, "Push the limit of acoustic gesture recognition," *IEEE TMC*, vol. 21, no. 5, pp. 1798–1811, 2020.

[11] H. Chen, F. Li, and Y. Wang, "Echotrack: Acoustic device-free hand tracking on smart phones," in *IEEE INFOCOM*. IEEE, 2017, pp. 1–9.

[12] D. Li, J. Liu, S. I. Lee, and J. Xiong, "Fm-track: pushing the limits of contactless multi-target tracking using acoustic signals," in *ACM SenSys*, 2020, pp. 150–163.

[13] K. Ling, H. Dai, Y. Liu, A. X. Liu, W. Wang, and Q. Gu, "Ultragesture: Fine-grained gesture sensing and recognition," *IEEE TMC*, vol. 21, no. 7, pp. 2620–2636, 2020.

[14] R. Nandakumar, V. Iyer, D. Tan, and S. Gollakota, "Fingerio: Using active sonar for fine-grained finger tracking," in *ACM CHI*, 2016, pp. 1515–1525.

[15] C. Zhang, Q. Xue, A. Waghmare, S. Jain, Y. Pu, S. Hersek, K. Lyons, K. A. Cunefare, O. T. Inan, and G. D. Abowd, "Soundtrak: Continuous 3d tracking of a finger using active acoustics," *ACM IMWUT*, vol. 1, no. 2, pp. 1–25, 2017.

[16] P. Wang, R. Jiang, J. Hu, Y. Zhu, H. Jiang, M. Li, and C. Liu, "Amt^+: Acoustic multi-target tracking with smartphone mimo system," *IEEE TMC*, 2024.

[17] M. Chen, P. Yang, J. Xiong, M. Zhang, Y. Lee, C. Xiang, and C. Tian, "Your table can be an input panel: Acoustic-based device-free interaction recognition," *ACM IMWUT*, vol. 3, no. 1, pp. 1–21, 2019.

[18] Y. Jin, Y. Gao, Y. Zhu, W. Wang, J. Li, S. Choi, Z. Li, J. Chauhan, A. K. Dey, and Z. Jin, "Sonicasl: An acoustic-based sign language gesture recognizer using earphones," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 5, no. 2, pp. 1–30, 2021.

[19] J. P. Sahoo, A. J. Prakash, P. Pławiak, and S. Samantray, "Real-time hand gesture recognition using fine-tuned convolutional neural network," *Sensors*, vol. 22, no. 3, p. 706, 2022.

[20] P. Wang, W. Li, S. Liu, Y. Zhang, Z. Gao, and P. Ogunbona, "Large-scale continuous gesture recognition using convolutional neural networks," in *IEEE ICPR*. IEEE, 2016, pp. 13–18.

[21] H. Zou, J. Yang, Y. Zhou, and C. J. Spanos, "Joint adversarial domain adaptation for resilient wifi-enabled device-free gesture recognition," in *IEEE ICMLA*. IEEE, 2018, pp. 202–207.

[22] C. Dian, D. Wang, Q. Zhang, R. Zhao, and Y. Yu, "Towards domain-independent complex and fine-grained gesture recognition with rfid," *ACM CHI*, vol. 4, no. ISS, pp. 1–22, 2020.

[23] Y. Zou, Q. Yang, Y. Han, D. Wang, J. Cao, and K. Wu, "Acoudigits: Enabling users to input digits in the air," in *IEEE PerCom*. IEEE, 2019, pp. 1–9.

[24] K. Wu, Q. Yang, B. Yuan, Y. Zou, R. Ruby, and M. Li, "Echowrite: An acoustic-based finger input system without training," *IEEE TMC*, vol. 20, no. 5, pp. 1789–1803, 2020.

[25] Y. Zou, Z. Xiao, S. Hong, Z. Guo, and K. Wu, "Echowrite 2.0: A lightweight zero-shot text-entry system based on acoustics," *IEEE T HUM-MACH SYST*, vol. 52, no. 6, pp. 1313–1326, 2022.

[26] Y. Wang, J. Shen, and Y. Zheng, "Push the limit of acoustic gesture recognition," *IEEE TMC*, vol. 21, no. 5, pp. 1798–1811, 2020.

[27] H. Du, P. Li, H. Zhou, W. Gong, G. Luo, and P. Yang, "Wordrecorder: Accurate acoustic-based handwriting recognition using deep learning," in *IEEE Infocom*. IEEE, 2018.

[28] H. Yin, A. Zhou, L. Liu, N. Wang, and H. Ma, "Ubiquitous writer: Robust text input for small mobile devices via acoustic sensing," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5285–5296, 2019.

[29] H. Yin, A. Zhou, G. Su, B. Chen, L. Liu, and H. Ma, "Learning to recognize handwriting input with acoustic features," *ACM IMWUT*, vol. 4, no. 2, pp. 1–26, 2020.

[30] Y. Wang, J. Shen, and Y. Zheng, "Push the limit of acoustic gesture recognition," *IEEE TMC*, vol. 21, no. 5, pp. 1798–1811, 2020.

[31] D. Li, J. Liu, S. I. Lee, and J. Xiong, "Room-scale hand gesture recognition using smart speakers," in *ACM Sensys*, 2022, pp. 462–475.

[32] Y. Zou, Y. Wang, H. Dong, Y. Wang, Y. He, and K. Wu, "Pregesnet: Few-shot acoustic gesture recognition based on task-adaptive pretrained networks," *IEEE TMC*, 2024.

[33] 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 *ACM Mobicom*, 2019, pp. 1–15.

[34] Q. Zhang, D. Wang, R. Zhao, and Y. Yu, "Myosign: enabling end-to-end sign language recognition with wearables," in *ACM IUI*, 2019, pp. 650–660.

[35] Q. Zhang, J. Jing, D. Wang, and R. Zhao, "Wearsign: Pushing the limit of sign language translation using inertial and emg wearables," *ACM IMWUT*, vol. 6, no. 1, pp. 1–27, 2022.

[36] T. Labs. MYO : Gesture control armband by Thalmic Labs. 2013. [Online]. Available: https://developerblog.myo.com/

[37] Q. Zhang, D. Wang, R. Zhao, Y. Yu, and J. Jing, "Write, attend and spell: Streaming end-to-end free-style handwriting recognition using smartwatches," *ACM IMWUT*, vol. 5, no. 3, pp. 1–25, 2021.

[38] 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 TMC*, vol. 21, no. 7, pp. 2398–2410, 2020.

[39] S. Xu, Y. Xue, X. Zhang, and L. Jin, "A novel unsupervised domain adaptation method for inertia-trajectory translation of in-air handwriting," *Pattern Recognition*, vol. 116, p. 107939, 2021.

[40] X. Zhang, C. Tang, K. Yin, and Q. Ni, "Wifi-based cross-domain gesture recognition via modified prototypical networks," *IEEE Internet of Things Journal*, vol. 9, no. 11, pp. 8584–8596, 2021.

[41] X. Shi, Y. Jin, Q. Dou, J. Qin, and P.-A. Heng, "Domain adaptive robotic gesture recognition with unsupervised kinematic-visual data alignment," in *IEEE/RSJ IROS*. IEEE, 2021, pp. 9453–9460.

[42] D. Wang, J. Yang, W. Cui, L. Xie, and S. Sun, "Airfi: Empowering wifi-based passive human gesture recognition to unseen environment via domain generalization," *IEEE Transactions on Mobile Computing*, vol. 23, no. 2, pp. 1156–1168, 2024.

[43] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," in *IEEE ICML*. PMLR, 2017, pp. 1243–1252.

[44] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for mobilenetv3," in *IEEE/CVF ICCV*, 2019, pp. 1314–1324.

[45] Y. Ganin and V. Lempitsky, "Unsupervised domain adaptation by backpropagation," in *IEEE ICML*. PMLR, 2015, pp. 1180–1189.

[46] M. Ghifary, W. B. Kleijn, M. Zhang, D. Balduzzi, and W. Li, "Deep reconstruction-classification networks for unsupervised domain adaptation," in *Springer ECCV*. Springer, 2016, pp. 597–613.

[47] J. Wang, J. Chen, J. Lin, L. Sigal, and C. W. de Silva, "Discriminative feature alignment: Improving transferability of unsupervised domain adaptation by gaussian-guided latent alignment," *Pattern Recognition*, vol. 116, p. 107943, 2021.

[48] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[49] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE CVPR*, 2016, pp. 770–778.

Jane Doe Biography text here without a photo.