2024 IEEE International Conference on Pervasive Computing and Communications (PerCom) | 979-8-3503-2603-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/PERCOM59722.2024.10494485

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

Weiyu Chen†, Canlin Zheng†, Wenfeng He†, Yongpan Zou† and Kaishun Wu‡
† College of Computer Science and Software engineering, Shenzhen University, Shenzhen, China
‡ Information Hub, The Hong Kong University of Science and Technology (Guangzhou), Guangzhou, China
richardcwy426@gmail.com; zhengcanlin2020@email.szu.edu.cn; yongpan@szu.edu.cn; wuks@ust.hk

***Abstract*—Due to the advantages of acoustic sensing such as device ubiquity, hands-free interaction and privacy security, acoustic-based gesture input techniques have gained extensive attention and many excellent works have been proposed. However, these works have the following shortcomings: high cost of system construction, non-continuous input, and degraded performance in cross-user scenarios. To overcome the above shortcomings, we propose UltraWrite, a continuous gesture input system that needs rather low system construction cost, supports continuous input, and achieves high cross-user recognition performance. The key idea of our solution is to synthesize the data of continuous gestures from isolated ones, and build an end-to-end continuous gesture recognition model by introducing the connectionist temporal classification (CTC) mechanism widely used in natural language processing. We have implemented the system on a commercial tablet and conducted comprehensive experiments to evaluate its performance. The results demonstrate that UltraWrite achieves an average word accuracy of 99.3% and word error rate of 0.34% when considering only the first output candidate word. In addition, we have also evaluated the system's robustness to background noise, sensing distance and angle. The results reveal that UltraWrite displays strong robustness to these factors.**

***Index Terms*—Acoustic Sensing; Gesture Input; Cross-Domain Learning**

I. Introduction

With the emergence of new smart devices, the applications of human-computer interaction (HCI) have become increasingly diverse. Among various input schemes, gesture-based methods are particularly attractive due to their non-verbal nature, natural interaction, high acceptance, and high recognition rates. Consequently, gesture-based interaction has been extensively studied by scholars in the field of pervasive computing. These works can be broadly categorized according to the sensing media employed, including vision-based gesture interaction techniques [1], [2], radio-frequency signal-based [3–5], inertial sensor-based [6], [7], and acoustic sensor-based [8–13], among others. Among these approaches, gesture recognition technology based on acoustic perception has been rapidly developed for text input application scenarios, with many works demonstrating superior recognition performance.

However, current methods of hand or finger gesture recognition based on acoustic signals have the following key limitations. First, these methods usually require a large training dataset to achieve superior recognition performance, resulting in high system overhead. For instance, UltraGesture [13] needs collecting 100 times for each gesture, and Ipanel [14] needs to collect 50 times for each gesture. Moreover, most of the existing gesture recognition works face the problem of performance degradation when new users use them. To address this issue, some methods [15], [16] choose to personalize and fine-tune the trained system model for new users, while others [8], [17]. [18] use domain adaptation methods for model optimization. However, these methods require a relatively small sample size from new users and may be computationally expensive. For continuous input gestures with theoretically infinite categories, these methods become impractical since the total number of samples that new users need to provide increases significantly. Therefore, a question naturally comes up: *can we design a low-overhead and cross-user friendly system for continuous gesture text entry?*

Fig. 1. Possible application scenarios of UltraWrite

In this paper, we utilize active sensing to capture finger writing activity with high-frequency acoustic signals emitted from a speaker which is available in commercial devices such as smartphones, tablets, and PC. To address the aforementioned challenges, we employ the following strategies in constructing UltraWrite. We devise a training dataset construction approach based on decomposition and reconstruction to minimize system overhead. The collection of continuous gesture data was decomposed into the acquisition of isolated gestures, followed by the synthesis of continuous gestures from these isolated instances. To achieve continuous gesture input, we formalize it as a natural machine translation (NMT) task. Leveraging the distinctive features of gesture data, we apply LeNet and Bi-GRU as feature extractors and translation

modules, respectively. Tackling the difficulty of annotating continuous gesture data, we utilize Connectionist Temporal Classification (CTC). To address the cross-user issue without adding usage costs for new users, we devise a decoupled training method by integrating domain adaptation during encoding reconstruction and model decoupling. We employ the concept of encoding reconstruction to train the feature extractor for extracting domain-independent features. Ultimately, we design a system based on sound wave perception that is cost-effective, supports continuous input, and exhibits excellent cross-user performance. Fig. 1 shows the possible application scenarios.

In brief, our work can be summarized by highlighting the following main contributions.

1) We design the dataset construction scheme by combining the task decomposition and reconstruction idea and the data augmentation method. This scheme keeps the system overhead at a very low level.
2) We transform the gesture input as an NMT task and implement continuous gesture input by using acoustic sensing combined with CTC in the speech domain. We propose a training strategy across user scenarios that combines domain adaptation methods for encoding reconstruction with the model decoupling training idea.
3) We implement a prototype system on the Android platform, and deploy it on 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. The results validate 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. VI concludes the paper.

II. Related Work

A. Acoustic-based Gesture Recognition

Zou et al. utilized the Doppler effect to develop AcouDigit [19], EchoWrite [20], and EchoWrite 2.0 [21], which allow for letter and number input on mobile devices. Likewide, RoboCIR [22] and UltraGesture [13] detect changes in channel impulse response (CIR) induced by finger gestures, achieving a high recognition precision of 15 and 12 hand gestures. For alphabetic and numeric input, Yang et al. used sound signals to develop WordRecorder [23] and Ipanel [14]. Additionally, Ma et al. proposed UbiWriter [24] and WritingRecorder [25]. However, our work differs from them in two aspects. For one thing, these works require collecting massive training data to build recognition models which bring about heavy labourious burden. For another thing, they do not deal with the problem of continuous gesture recognition which is rather significant in many HCI applications.

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, Signspeaker [26] utilizes motion sensors on smartwatches to enable continuous input of Sign Language gestures. MyoSign [27] and WearSign [28] employ motion sensors and electromyographic (EMG) signals collected from Myo armband devices [29]. They also developed DeepSLR [30] for hand gesture recognition using two Myo armbands, including simultaneous recognition of both hands, and WriteAS [31] for English input on a smartwatch. However, these systems rely on IMU and EMG for gesture input, and DeepSLR [30] and WearSign [28] require collecting 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. Nonetheless, the systems necessitate costly training datasets and do not address cross-user issues.

III. System Design

In this section, we will elaborate on the design details of our system. As shown 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 data instead of real collected data, and data augmentation methods are employed during this process to expand the dataset. Next, a CNN is designed to extract features from the input data, and finally, these features are passed through a context encoder for encoding to output words.

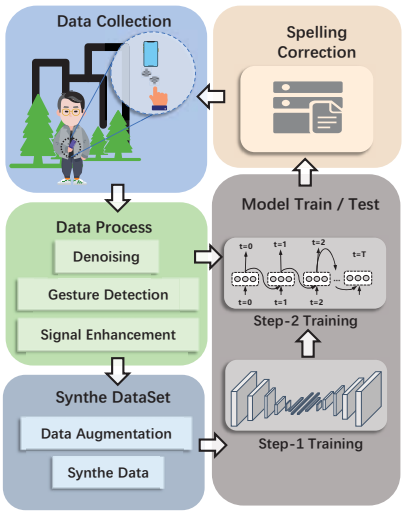


Fig. 2. The overview of UltraWrite's architecture

A. Signal Processing

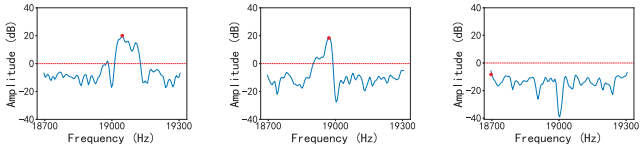
The acoustic signal captured by the microphone is initially converted into a spectrogram using an STFT with a Hamming window. We set the frame length to 8192 and window step size to 512 through empirical means. And we adopt a window size of 100 frames with a 70% overlap for sliding segmentation.

*1) **Denoising**:* The transmit frequency of the microphone is set to 19 KHz. 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 will experience 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 Eq.(1).

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

In the 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. For our experiment, the velocity of finger movement was measured to be approximately 2.5 ms. Consequently, the maximum frequency shift was calculated to be 281 Hz, resulting in a theoretical effective frequency band of $[18722, 19281]$ Hz. To simplify the calculations, we extend the effective frequency range to $[18700, 19300]$ Hz and extract the spectrogram within this band 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 frequency component of significant strength 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.



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

Fig. 3. The relationship between finger motion and frequency variation

*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, as depicted in Fig. 3a, when the finger approaches the device. 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.

The current sliding window is positioned at the beginning of the word gesture when the dominant frequency, which corresponds to the frequency component with the highest signal strength, differs from 19 KHz within all frames encompassed by the sliding window. Consequently, we designate the first window within the current 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 window equals 19 KHz, it signifies that the current sliding window denotes the end of the word gesture. In this case, we designate the first STFT window within the current 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 ascertain if they contain gesture instances. The aforementioned gesture detection process is summarized in Algorithm 1.

Algorithm 1 Gesture Detection Algorithm

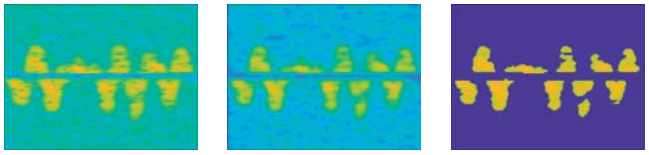
Input: Raw Spectrogram P_{Raw}
Output: Gesture Spectrogram P_{Ges}

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

*3) **Signal enhancement**:* Due to the hardware constraints, simultaneous operation of these devices generates some noise. These non-zero components are visually depicted 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}. Fig. 4a, Fig. 4b, and Fig. 4c illustrate the spectrogram after undergoing normalization, Gaussian smoothing, and thresholding, respectively.

B. Dataset Construction

In order to reduce the cost of data collection for training, we divide the original data acquisition task for training (word) into smaller units (letter). By synthesizing data using these letters, we can minimize the cost of acquiring the training dataset. Besides, we observe that writing between letters may lead to additional Doppler frequency shifts.

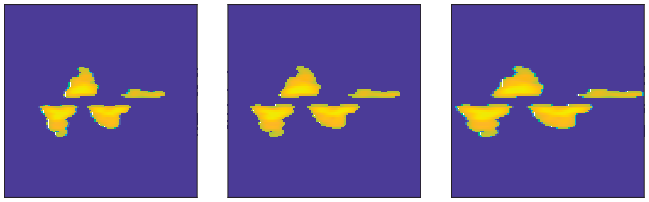


(a) After normalization (b) After Gaussian smoothing (c) After the thresholding

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

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.

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. 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. Based on this observation, 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) $\rho = 1.20$ (b) $\rho = 1.00$ (c) $\rho = 0.80$

Fig. 5. The enhancement results with different values of 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. 5 displays the spectrograms of the letter gesture '*A*' multiplied by the factors 1.20, 1.00, and 0.80, respectively, illustrating the impact of the time-warping technique.

2) ***Data synthesis****:* 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 achieve 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: Words template

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 have been 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 have been grouped according to their main parts of speech and are summarized in Table I.

C. Model Training

In this subsection, we provide a detailed description of the system modeling and the design of the cross-domain training strategy. 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 a NMT task [32], 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 as 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 [33], 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-B1.

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 the 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-B2. 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**:* In the conventional deep learning modeling process, it is typically assumed that the training dataset (source domain) and test dataset (target domain) are independent and identically distributed. However, this assumption often does not hold in practical application scenarios. In our task, the source domain comprises the synthetic dataset, while the target domain consists of the real dataset written by users.

Conventional encoding recognition domain adaptation training strategy, 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. The feature extractor is trained 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. During the training of the translation module, the parameters of the feature extraction module are kept fixed, and only the parameters of the Translation Module are adjusted. 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) **Spelling correction**:* We employ Bayesian inference to correct the model's outputs. Let's assume that a word gesture's spectrogram is divided into 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 commonly 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 be $P(l_i)$. 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 can be 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. But 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 have impact on the system performance. We leave this evaluation as one of our future work.

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 vary from 22 to 25 years old, and all were 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 accomplish the writing within this range. 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 was 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 were unusable due to device or collection process anomalies. In the end, each volunteer has 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.

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

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**:* In the user-independent test, we train the model using the synthetic dataset, which is described in Sec. III-B, and evaluate the model on a test dataset of word gestures written by volunteers in real-world scenarios. Fig. 6 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 experiment demonstrates that the proposed system in this work achieves superior 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. This subsection evaluates the system's ability to generalize to unseen word gestures. 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 the words of length 2 or 3 by substituting, deleting, or inserting letters. We then 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-B. 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. 7, 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.

*3) **Response time**:* The response time of UltraWrite mainly consists of the time spent on data processing, word prediction, and spell correction. When tested on a Samsung Tab S2, UltraWrite achieves an average response time of about 3340 ms to recognize a four-ltter English word, which means about 18 words can be inputted per minute. As the response time is closely related to the device's hardware, it should be much less when UltraWrite is implemented on a more newly released product.

B. Ablation Study

*1) **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 [34], MobileNet-v3 [33], and ResNet-18 [35] as the feature extractors on the system's performance. This evaluation only changes the backbone of the feature extractor without any other settings. Fig. 8 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-C, 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. Moreover, LeNet has the smallest model size compared to AlexNet, MobileNet, and ResNet, which are about 213, 16, and 42 times larger, respectively. Considering the test results and the model size, it is most reasonable to design the Feature Extractor based on LeNet.

*2) **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 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. 9, 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.

*3) **Dataset construction**:* In this part, we evaluate the effectiveness of the train dataset construction scheme described in Sec. III-B. 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. D_i represents training dataset ($i = 1, 3, 5, 7, 9$), where i indicates that the dataset is increased by i times. The results are shown in Fig. 10. When D_1 is used for training, the top-1, top-3, and top-5 CER are 5.13%, 3.92%, and 3.52%,

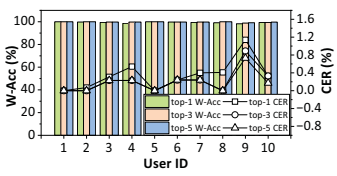


Fig. 6. Results of cross user

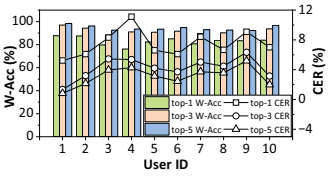


Fig. 7. Results of unseen word

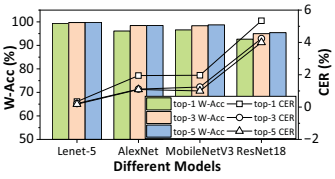


Fig. 8. Different feature extractor

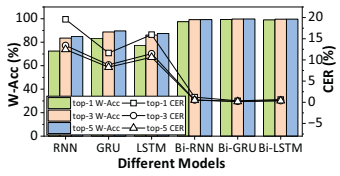


Fig. 9. Different translate module

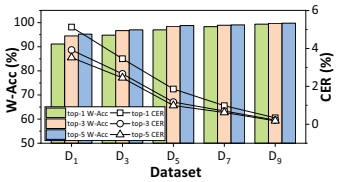


Fig. 10. Dataset construction

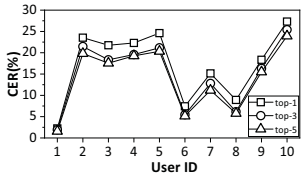


Fig. 11. CER difference

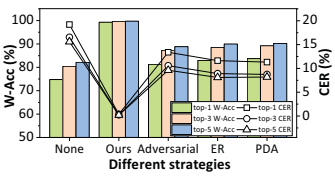


Fig. 12. Different strategies

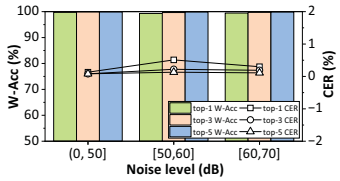


Fig. 13. Impact of noise level

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.

4) ***Cross-domain training strategy****:* In this part, we evaluate the effectiveness of the cross-domain training strategy proposed in Sec. III-C2. When testing without the cross-Domain training strategy, only the CTC loss is used as the loss function during model training. The parameters of the feature extractor and the translation module are adjusted in a joint training mode. The training dataset is the synthesized word gesture dataset (D_9). We calculate the difference in CER between the cases with and without the cross-domain training strategy. The results are shown in Fig. 11. V_1 is a non-cross-user test, so the performance improvement is minimal. Compared to the case without the cross-domain training strategy, the recognition performance of the system is significantly improved. The average top-1, top-3, and top-5 CER of the system with cross-domain training strategy decreases by 17.16%, 14.86%, 14.01%, respectively.

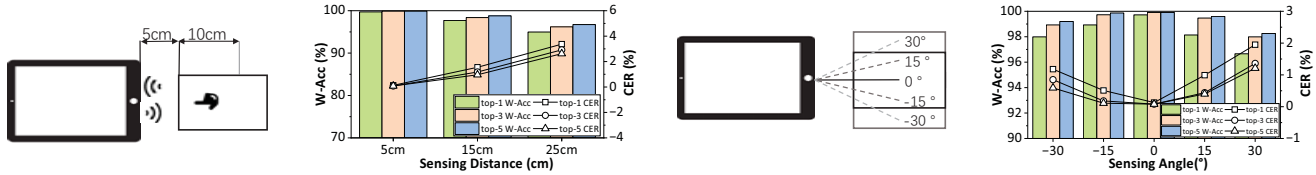
In addition, we compare the effects of three data augmentation strategies: Adversarial [36], Encoding Reconstruction (ER) [37], and Prior Distribution Alignment (PDA) [38], on the cross-user recognition performance of our system. We conduct tests on nine volunteers (V_2 to V_{10}) and the results are presented in Fig. 12. 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 [36] may cause the extracted features to lose original information, which can lead to encoding errors in the translation module. The PDA [38] also faces similar problems, albeit to a lesser degree. On the other hand, the ER [37] 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.

C. System Robustness

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

1) ***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, 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. 13. 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-A.

2) ***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. 14a. Volunteers were then asked to write word gestures in different distance. The evaluation results are shown in



(a) Distance setup (b) Experimental result (c) Angle setup (d) Experimental result

Fig. 14. Impacts of distance and angle.

TABLE II: Comparison with others continuous gesture input systems

Work	Modality	Training users	Sizes	Seen word		Unseen word	
				single-user	cross-user	single-user	cross-user
Signspeaker [26]	IMU	16	11680	98.96%	89.30%	-	-
DeepSLR [30]	IMU+EMG	34	20400	89.20%	-	-	-
MyoSign [27]	IMU+EMG	15	7500	-	93.10%	92.40%	-
WriteAS [31]	IMU	12	22500	97.60%	84.60%	93.40%	83.80%
WearSign [28]	IMU+EMG	15	6000	99.71%	94.50%	91.40%	89.20%
SonicASL [39]	Acoustic	8	9600	90.60%	-	-	-
Ours	**Acoustic**	**1**	**390**	**100%**	**99.74%**	**98.37%**	**93.92%**

Fig. 14b. Our findings show that the system performance decreases as the sensing distance increases. When the perceived distance is increased 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.

*3) **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. 14c, where the writing range moves to the left indicating a negative angle. We then ask volunteers to write word gestures at different angles. The evaluation results are presented in Fig. 14d, which show that the system's performance gradually decreases as the angle shifts. 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 exhibits great robustness to the sensing angle, with an overall W-Acc above 96%.

D. 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 [26], WriteAS [31], and WearSign [28] achieve comparable recognition accuracy with UltraWrite. But they require more than 15 times of training data. Other works have much lower recognition accuracy and a heavier burden of collecting training data. In addition, we can notice that in the cross-user scenario, UltraWrite only requires 390 (26×15) alphabet gestures collected from a single volunteer to achieve an average W-Acc of 99.74%. 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.

VI. 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 in devices such as smart glasses, headphones, and watches.

VII. Acknowledgement

This research was supported in part by China NSFC Grant (62172286, U2001207), Guangdong NSF Grant (2022A1515011509), Guangdong Provincial Key Lab of Integrated Communication, Sensing and Computation for Ubiquitous Internet of Things (2023B1212010007), the Project of DEGP (2023KCXTD042), and Tencent "Rhinoceros Birds"-Scientific Research Foundation for Young Teachers of Shenzhen University. We also highly appreciate the effort of anonymous reviewers and our shepherd Dr. Till Riedel. Yongpan Zou is the corresponding author.

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] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

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

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.

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

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

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

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

[37] 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.

[38] 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.

[39] 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.