Swift: Turning Your Hand into a Writing Pad with Unmodified Smartwatches

WENTAO XIE, The Hong Kong University of Science and Technology, Hong Kong SAR
YANKAI ZHAO, Southern University of Science and Technology, China
CHI XU, The Hong Kong University of Science and Technology, Hong Kong SAR
ZHENGYAN LAMBO QIN, The Hong Kong University of Science and Technology, Hong Kong SAR
YIXUAN LIU, The Hong Kong University of Science and Technology, Hong Kong SAR
JIN ZHANG, Southern University of Science and Technology, China
QIAN ZHANG, The Hong Kong University of Science and Technology, Hong Kong SAR

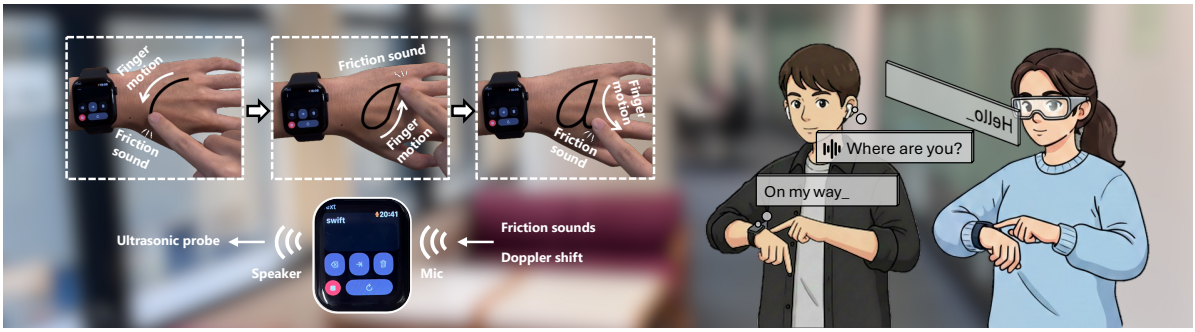


Fig. 1. *Swift* transforms the back of the hand into a handwriting pad to support heads-up text entry, such as in AR/VR/MR.

Smartwatches remain constrained by small touchscreens. To enable expressive text input without extra hardware, we introduce *Swift*, an on-skin writing technique that treats the back of the hand as a passive handwriting pad to capture handwritten English letters and gestures with an unmodified commercial smartwatch. The core design rationale is that *Swift* measures high-frequency finger-on-skin friction acoustics and complements them with active ultrasonic probing to better recognize characters with similar stroke trajectories. A lightweight cross-attention fusion model is designed to reliably integrate the two modalities, while robustness is ensured via various augmentation techniques. Across an offline technical evaluation in lab and everyday environments (N=22), *Swift* achieves a macro F1-score of 0.91 after leave-one-subject-out training and few-shot personalization. A real-time deployment study (N=8) demonstrates phrase composition on the watch with users reporting the system as learnable and engaging. A Wizard-of-Oz comparison study (N=8) further indicates users generally prefer *Swift* over touchscreen handwriting, with preference increasing for more demanding scenarios such as heads-up text entry.

Authors' Contact Information: Wentao Xie, The Hong Kong University of Science and Technology, Hong Kong SAR, wentaox@cse.ust.hk; Yankai Zhao, Southern University of Science and Technology, Shenzhen, China, 12432726@mail.sustech.edu.cn; Chi Xu, The Hong Kong University of Science and Technology, Hong Kong SAR, cxubs@cse.ust.hk; Zhengyan Lambo Qin, The Hong Kong University of Science and Technology, Hong Kong SAR, zlqin@connect.ust.hk; Yixuan Liu, The Hong Kong University of Science and Technology, Hong Kong SAR, yliumb@connect.ust.hk; Jin Zhang, Southern University of Science and Technology, China, zhang.j4@sustech.edu.cn; Qian Zhang (corresponding author), The Hong Kong University of Science and Technology, Hong Kong SAR, qianzh@cse.ust.hk.

ACM 2474-9567/2026/9-ART175
https://doi.org/10.1145/3832006

CCS Concepts: • **Human-centered computing** → **Mobile devices**; **Text input**.

Additional Key Words and Phrases: Smartwatch text entry, on-skin interaction, acoustics, multimodal fusion

ACM Reference Format:
Wentao Xie, Yankai Zhao, Chi Xu, Zhengyan Lambo Qin, Yixuan Liu, Jin Zhang, and Qian Zhang. 2026. *Swift*: Turning Your Hand into a Writing Pad with Unmodified Smartwatches. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 10, 3, Article 175 (September 2026), 31 pages. https://doi.org/10.1145/3832006

1 INTRODUCTION

Smartwatches have moved from simple accessories to mainstream personal computing companions. Their always-on sensing capability, displays, and tight integration with notification ecosystems have made them a primary endpoint for lightweight interactions such as quick messaging, health tracking, and voice assistance. Despite increasingly powerful processors and richer application ecosystems, smartwatch input remains fundamentally constrained. Their small screen form factor and on-the-go usage contexts limit how users can comfortably author text beyond brief canned responses. Consequently, enabling easy-to-use and expressive text input on smartwatches, without encumbering the user with extra hardware, remains an open problem in human–computer interaction.

Current smartwatch text-entry techniques primarily rely on two standard modalities: voice-based dictation and screen-based interaction. Voice dictation [1, 3, 4], including specialized whisper-mode techniques [21, 34], is widely considered a *de facto* input method due to its convenience and low physical demand. However, screen-based alternatives remain indispensable in scenarios where audible or semi-audible vocalizations are unsuitable, such as meetings, classrooms, or shared living spaces. Despite their necessity, current on-screen keyboards and handwriting techniques are constrained by the limited size of the touch screen. This leads to the well-documented 'fat finger' problem [31, 50, 61], characterized by visual occlusion during the input process.

In response to this constraint, prior research has explored extending the input space beyond the watch itself to nearby body or environmental surfaces. The skin on our bodies, in particular, has been designed to serve as a ubiquitous, always-available input canvas with natural tactile feedback [11–13, 15, 32, 39, 43, 58, 73, 74, 77]. These works leverage skin deformation, vibration, or the skin's electrical properties to detect touch gestures performed on the skin. The usability and acceptability of this new skin-as-touchpad metaphor have been confirmed by previous studies [15, 39, 58]. Therefore, if the back of the hand can be conceptualized as a passive **handwriting pad** for smartwatches, supported by Weigel *et al.* [58], this extended interaction space would give the user much greater space to compose text, solving the fat-finger problem. Also, the tactile feedback provided by the skin can further support heads-up interaction in XR scenarios, as previous works show that cutaneous tactile cues and body-based proprioception can support complex, immersive interaction by reducing sensory conflicts [6, 25, 69]. For example, when using AR glasses, the screen may display a notification from a friend. To send a brief reply, the user can simply compose phrases on the back of their hand without needing to look down, thereby maintaining continuous visual engagement with the virtual content. An illustrative use case of this design is shown in Figure 1.

However, although this hand-back input metaphor is intriguing, existing methods either require additional sensing hardware or support only limited on-skin gestures, and none support full-alphabet text entry with unmodified watches, thereby limiting deployability on today's commercial devices. To bridge this gap, we leverage the core insight that finger-on-skin writing produces rich friction-induced acoustic signatures whose patterns vary across letters. These distinct acoustic features can be captured by a smartwatch's built-in microphone to enable handwriting recognition. Crucially, these friction sounds include high-frequency components extending into the near-ultrasonic range, portions less affected by typical ambient human speech and environmental noise, creating an opportunity for more robust sensing. Based on this idea, we introduce *Swift*, a novel on-**s**kin **wr**i**t**ing

system **f**or **t**ext input that recognizes all 26 lowercase English letters plus two slide gestures written directly on the hand back using a commercial smartwatch (Figure 1).

Realizing the above idea raises three key technical challenges. First, many letters have similar stroke trajectories (*e.g.*, c *vs.* o and i *vs.* j), making passive friction sounds alone ambiguous. *Swift* therefore combines passive friction sensing with active ultrasonic probing: the watch speaker emits an inaudible carrier, and finger motion induces Doppler shifts captured by the microphone, which complement the friction cues. Second, reliably integrating these complementary cues is critical. To address this, we design *Swift* with a lightweight, cross-attention-based modality fusion network to achieve a holistic fusion. Third, user-dependent handwriting styles and variable environmental noise introduce substantial intra-class variability and robustness issues, compromising the system's practical use. To address this issue, we utilize extensive data augmentation techniques. We employ the mixup and CutMix sample interpolation strategies [71, 76] to encourage smoother decision boundaries across handwriting style variants, and time–frequency random masking strategies [45, 67] to combat local signal variations. Additionally, we augment the dataset with noise samples from open-source daily environment audio datasets [46, 53] to enhance the model against real-world acoustic interference. Finally, we incorporate a lightweight personalization phase with only a few user-specific exemplars per class to adapt the base model to each individual user, balancing population-level generality with personalized accuracy.

With these components, *Swift* supports on-skin handwriting recognition of 28 characters (26 English letters plus right- and left-slide gestures for space and backspace) on a commercial smartwatch, requiring no hardware modification. We evaluate *Swift* through one technical study and two user studies. First, an offline technical evaluation (N=22) is conducted in a controlled laboratory setting and several everyday environments, where *Swift* achieves an overall F1-score of 0.91, evaluated with leave-one-subject-out (LOSO) training followed by few-shot personalization. Second, a real-time deployment study (N=8) integrates *Swift* into an interactive text-entry application, enabling participants to compose phrases via on-skin writing. Results indicate that the system is learnable and engaging, and qualitative feedback highlights opportunities to improve future on-skin handwriting experiences. Based on these encouraging results, we explore scenarios beyond basic letter entry—such as word-level input and heads-up text entry. To examine user preferences in these higher-demand settings while relaxing current implementation constraints, we conduct a Wizard-of-Oz study (N=8) comparing *Swift* with conventional touchscreen-based handwriting on smartwatches. Participants generally preferred *Swift* for letter-by-letter input, and this preference increased substantially for word-level and heads-up input, suggesting that *Swift*'s larger, continuous on-skin canvas and fluid workflow provide clearer benefits as text complexity and situational demands increase. We summarize the contributions of this work as follows:

- A **new interaction technique** that, to the best of our knowledge, is the first to recognize full-alphabet on-skin handwriting (26 letters + control gestures) using only an unmodified commodity smartwatch.
- A **sensing and modeling pipeline** that reliably fuses passive friction acoustics with active ultrasonic Doppler cues, coupled with robustness techniques (mixup and CutMix, random masking, noise augmentation, and few-shot personalization) to address inter-letter similarity, user variability, and ambient noise.
- A **multi-study evaluation** comprising a technical validation, a real-time deployment study, and a Wizard-of-Oz study, demonstrating (a) high offline recognition performance, (b) practical usability and positive user reception with design insights for future on-skin handwriting workflows, and (c) stronger perceived benefits over touchscreen-based handwriting in more demanding text entry scenarios.

2 RELATED WORK

In this section, we review related literature. First, in the application domain, we review the works that design new interaction methods with smartwatches. Second, in the technical domain, we review the techniques that enable interaction with wearables using acoustics.

2.1 Interacting with Smartwatches

Research on smartwatch interaction has explored extending input beyond the small touchscreen. One direction is to enable gesture recognition on or around the device. To further overcome the screen size limitations, other studies have investigated on-skin gesture recognition.

On/around the watch interaction. There is a line of work that studies optimizing the existing on-screen input methodology using assistive software or hardware. For example, WatchWriter [19] uses advanced decoding algorithms to recognize screen tapping, and WrisText [18], which introduces a novel interaction metaphor by using joystick-like wrist rotations for character selection. Although these methods have proven more effective than traditional screen input, these designs are still constrained by their inherently small form factor. Beyond the conventional touchscreen, research on gesture recognition around the device has evolved. A significant body of work explores acoustic sensing [31, 42, 56, 70, 75, 80], where these systems transform the smartwatch into an active sonar system that transmits inaudible signals and tracks finger movements via returning echoes or probes. Further, WatchHand [30] extends this idea to build a continuous hand pose tracker on smartwatches. Concurrently, a major research thrust focuses on motion-based recognition using built-in inertial sensors. This includes systems for 3D air-writing recognition [5, 68] and finger or hand gesture recognition [59, 65]. In addition, MagSkin [78] designs a soft magnetic skin that can be worn on a user's fingers to support around-the-device interaction using the built-in magnetometer. The above works successfully bypass the screen-size restriction by extending the interaction space into the air. *Swift* is different from these works in that instead of focusing on gestures in the air, we extend the interaction space to the skin.

On-skin interaction. In parallel with the around-the-device interaction research, another line of research targets using the human skin as a ubiquitous, always-available input canvas with natural tactile feedback. This new skin-as-touchpad metaphor has been shown to be highly usable and acceptable in more demanding interaction scenarios, such as eyes-up interaction in VR/AR/MR applications [6, 25, 69]. ViType [13] and Taprint [12] use wearable sensors to turn the hand into a virtual keyboard. ViType employs vibration sensing through piezoelectric sensors to recognize keystrokes on the user's hand surface. Taprint further implements a virtual numeric keypad using knuckles as landmarks and leverages tapping vibrometry for secure biometric authentication. Moving beyond discrete key input, HandPad [39] utilizes capacitive sensing to detect writing gestures directly on the hand, effectively making the skin a touch-sensitive surface. Similarly, SkinTrack [77] enables continuous gesture tracking on the skin by employing high-frequency AC signals for precise motion detection, and Fang *et al.* [15] design a skin-worn flexible sensor to accept touch gestures for eyes-up interaction scenarios. In a different approach, FineType [32] combines IMU and infrared camera data to recognize text entry through finger tapping gestures on any flat surface. SoundScroll [28] demonstrates that wrist-worn in-air and on-skin microphones are capable of recognizing finger sliding gestures to support eyes-free interaction. Although the above works show strong potential in using the skin to accept interaction gestures, they all require additional sensing hardware. Closest to our work, iPand [11] turns the back of the hand into a gesture surface using the unmodified smartwatch microphone. It shows that weak skin-friction acoustics are sufficient for recognizing gestures such as swipes and pinches. Our work builds on this insight but targets the much more complex full-alphabet handwriting, which involves recognizing more delicate and ambiguous finger strokes, as detailed in Section 3.

2.2 Acoustic Techniques in Wearable Interaction

The core technique of this work is to use different acoustic modalities to capture the writing trajectory, as we will discuss in the following sections. Therefore, here we also review the interaction techniques enabled by acoustic-based methods. We group the reviewed works into three categories: passive and active acoustics, and the integration of acoustics with other modalities.

Passive acoustics. TapSense [22] and Lopes *et al.* [36] recognize input objects by segmenting and classifying the sounds generated from the objects. The principle of using surfaces for input is advanced by UbiTap and S-UbiTap [10, 29], which employ sound and the physical phenomenon of dispersion to turn solid surfaces into a touch input space. Similarly, Toffee [62] extends touch interaction beyond a mobile device's screen onto adjacent surfaces like tabletops using acoustic arrival time analysis. Acustico [17] also adopts this form of analysis, incorporating both surface and sound waves to improve performance. Other surface-based sensing includes Braun *et al.* [9], whose system uses an attached acoustic sensor to detect various surface interactions. SAWSense [23] repurposes a voice pick-up unit to capture surface acoustic waves generated by objects for interaction recognition. Ubik [55] localizes keystrokes on a mobile device by analyzing multipath fading with dual microphones. SoundScroll [28] and FingerBar [79] design finger-sliding interfaces by leveraging vibrations from friction. Beyond surfaces, passive acoustic sensing is applied to bio-sensing and bodily gestures. EarSense [47] demonstrates the feasibility of localizing teeth gestures by analyzing their acoustic signatures.

Active acoustics. Active acoustic sensing systems function by emitting a known audio signal and analyzing its reflections to track movement. Works [42, 56, 70, 75, 80] discussed in the previous section all fall into this category. To enhance the generalizability of finger-tracking, MagicInput [44] uses an acoustic-based 1D finger tracking algorithm aided by a novel data augmentation method applied to the MNIST dataset. For robust hand tracking, RobuCIR and UltraGesture [35, 57] introduce a method that uses acoustic reflections for robust hand gesture recognition on mobile devices. Beyond finger and hand tracking, this principle is also applied to facial monitoring. Works [26, 33, 51] emit acoustic signals on an earable device towards the face or outward to track facial expressions or hand-face interaction, while works [52, 64] adopt a similar approach on eyewear.

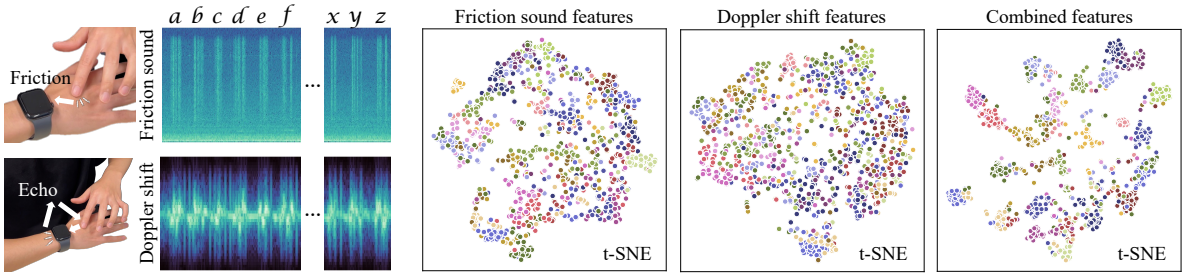
Combining acoustics with other modalities. Research has explored fusing acoustic sensing with other modalities to achieve more robust interaction. SurfaceLink [16] enables gesture-based control for device association and information transfer across a shared surface by a combination of on-device accelerometers, vibration motors, speakers, and microphones for sensing. FingerSound [74] facilitates eyes-free interaction by recognizing unistroke thumb gestures through a thumb-worn ring that integrates a contact microphone with a gyroscope sensor. Further advancing this fusion, AO-Finger [66] proposes a hands-free, fine-grained finger gesture recognition system based on acoustic-opticsensor fusing, implemented via a wristband equipped with a modified stethoscope microphone and two high-speed optical motion sensors.

3 RATIONALE AND PRELIMINARIES

In this section, we introduce the rationale for using the two feature modalities extracted from the smartwatch acoustics of an unmodified smartwatch to recognize on-skin handwriting. We will also present a preliminary study to verify the significance of combining complementary writing friction sounds and hand-induced Doppler shift in this task.

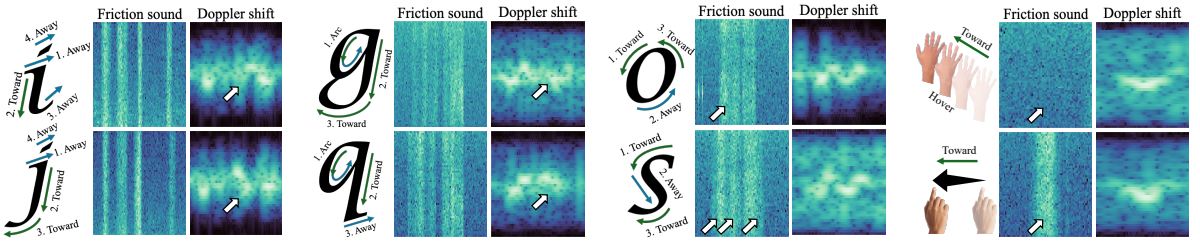
3.1 Finger Friction and Hand Motion in Handwriting

When writing letters on the skin, different stroke sequences will generate distinctive rhythmic sound patterns. For example, writing the letter *c* produces a clear stroke sound, and writing the letter *y* induces a rather complex sound of a series of strokes (first row in Figure 2(a)). Therefore, these sound patterns can be leveraged to infer the handwritten letters. In addition to rhythmic sound patterns, writing letters involves unique hand motions as well. For example, writing the letter *e* involves the hand moving along an arc, while writing the letter *x* involves mostly straight lines. Because modern smartwatches are usually equipped with speaker-microphone pairs, acoustic ranging methods [42, 56, 70] can be leveraged to probe hand motions and, therefore, predict handwriting letters (second row in Figure 2(a)).



(a) Writing-induced friction and motions (b) t-SNE visualization of representations extracted from different features

Fig. 2. Combining friction sounds and Doppler shift for handwriting recognition. (a) Friction sounds and the Doppler shift in handwriting English letters. (b) The t-SNE visualization of features extracted from the friction sound, the Doppler shift, and the combination of these two (letters are color-coded).



(a) Letters *i* and *j* (b) Letters *g* and *q* (c) Letters *o* and *s* (d) A false positive and a slide-left

Fig. 3. Feature ambiguity in recognizing handwriting letters and gestures. (a-b) Some letters share similar stroke rhythms but differ in contour. As a result, these letters will generate very similar friction sounds but different Doppler features. (c-d) Some letters and gestures may involve hand motion contours that resemble but differ in stroke strength and hand hovering patterns. Therefore, these letters and gestures will induce similar Doppler features but different friction patterns.

3.2 Combating Letter Ambiguity with Integrated Features

Although both the friction sounds and hand motions show good potential in supporting handwriting recognition, and previous works have separately demonstrated their feasibility in recognizing hand gestures [28, 42, 55, 70, 74], they fall short in predicting English letters. This is because certain letters share very similar stroke rhythms and contours in writing, which makes them hard to distinguish with a single sensing modality.

Figure 3 presents a case study of such examples. For letter pairs such as *i* vs *j* and *g* vs *q*, they all have similar stroke onset and offset patterns and share articulation cues. Therefore, it is hard to distinguish them solely from their friction sounds. However, for these two pairs, the Doppler shift caused by the stroke contour can well complement the friction feature. For letters *i* and *q*, the tail lift after the main stroke goes rightward, while the tail lift of letters *j* and *g* is leftward. This will result in opposite Doppler frequency shifts in the Doppler domain. Other examples of letter pairs of similar friction rhythm but different Doppler features include *b* vs *h* and *v* vs *r*.

Conversely, letters such as *o* and *s* can trace trajectories that, from the watch's perspective, move toward, away from, and then toward the device in a similar sequence. This produces comparable Doppler signatures despite clear acoustic differences in contact texture and articulation strength. Here, friction sounds help: they capture

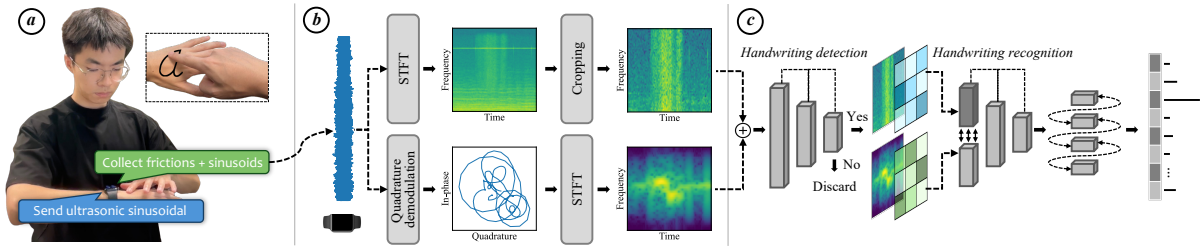


Fig. 4. *Swift* is designed to run on an unmodified smartwatch. (a) It collects the friction sounds and the ultrasonic probes reflected by the moving hands. (b) A signal processing pipeline is applied to extract the friction and Doppler spectrograms. (c) A deep-learning model is applied to predict the handwriting letters and gestures.

high-frequency surface textures, stroke-pressure variations, and onset/offset "timbre," which differ between a smooth loop (*o*) and a segmented, curvature-varying path (*s*). Other examples include *e* vs *t* (if the horizontal stick comes first) and *q* vs *a*. In addition, using the Doppler shift alone is prone to false positives as it characterizes hand motions, regardless of whether the user is writing or not. For example, in Figure 3(d), a hand hovering across the watch from the right will generate almost an identical Doppler shift as the slide-left gesture, but only the slide-left gesture will induce friction sounds where the user actually writes on the skin.

To better understand the complementary information carried by these two feature modalities, we visualize the features extracted from (i) the friction sound, (ii) the Doppler shift, and (iii) their combination with t-distributed Stochastic Neighbor Embedding (t-SNE). The results are shown in Figure 2(b). The models used to extract features are trained in Section 6.4, respectively. This result indicates that t-SNE embeddings from the fused features show tighter, better-separated clusters than either friction-only or Doppler-only embeddings. We will also verify in Section 6.4 that the combined feature modality significantly outperforms the use of either modality alone. With this fusion rationale established, we next describe how Swift acquires, processes, and combines these signals on an unmodified smartwatch for end-to-end handwriting recognition.

4 SYSTEM DESIGN

In this section, we present the system design of *Swift*. As discussed in Section 3, the system passively captures the friction sounds from finger–skin contact and actively detects the Doppler shift from hand motions using ultrasonic reflections to achieve a more robust handwriting representation. Therefore, we first program the smartwatch to continuously emit an 18kHz ultrasonic probe while simultaneously recording both the probe reflections and the friction sounds. Then, the system separates the ultrasonic probe from the friction sounds and derives the Doppler spectrogram from the ultrasonic probe and the friction spectrogram from the remaining signal (Section 4.1). These two spectrograms are subsequently fused and fed to a deep-learning model for predicting the handwritten letter or gesture (Section 4.2). An overview of the design of *Swift* is shown in Figure 4.

4.1 Sound Processing

After receiving the audio recordings, *Swift* first applies a short-time Fourier transform (STFT) to compute the spectrogram of the entire signal piece. At this stage, the derived spectrogram contains the 18kHz ultrasonic probe and large environmental noises as shown in Figure 5. Next, *Swift* extracts the friction sound and processes the ultrasonic probe separately. The parameters of the sound processing algorithms in this section are summarized in Table 1.

Table 1. *Swift* system parameters.

	Friction sound processing					Doppler shift processing				
Sec. 4.1	STFT					CIC filter			STFT	
	Window		*Hopping*		*Cutoff*	*Cutoff*	*Delay*	*Stages*	*Window*	*Hopping*
	20ms		10ms		10-22kHz	<10Hz	2	4	200ms	10ms
Sec. 4.2	Cross-attention fusion					Encoding			Classification	
	Projection		Cross attention		Gating	ResBlock			GRU	MLP
	Friction	*Doppler*	*Head*	*Dim*	*W*	*Layer 1*	*Layer 2*	*Layer 3*		
	<56, 48>	<40, 48>	4	48	<48, 1>	<1, 4>	<4, 16>	<16, 64>	128	<128, 64,>

4.1.1 Passive Friction Sound Processing. Figure 5 shows the spectrogram of a subject writing “hello[space]chi" on the skin. It is easy to observe that the spectral features of a letter form a cluster on the spectrogram, and these features span from the audible band to the near-ultrasound band. Consequently, although the microphone inevitably captures concurrent environmental noise, these noise components and the friction sounds are largely separable in the frequency domain. In *Swift*, we crop the spectrogram and only keep the 10-17.8kHz and the 18.2-22kHz bands to discard the audible noise and to avoid the ultrasonic probe at 18kHz. In this paper, we refer to the cropped spectrogram as the friction spectrogram. The friction spectrogram extracted from the audio in Figure 5 is shown in the first row of Figure 6. The resulting spectrogram has 225 frequency bins. We downsample it to 56 bins with a factor of four to reduce redundancy.

4.1.2 Active Doppler Shift Processing. *Swift* adopts the method discussed in [56] to process the ultrasonic probe. Specifically, we apply a quadrature demodulation with a Cascaded Integrator Comb (CIC) to move the ultrasonic probe to the baseband. Based on the baseband Doppler shift, we again compute the STFT to derive the Doppler spectrogram. The Doppler spectrogram extracted from audio in Figure 5 is shown in the second row of Figure 6. We can observe that, because writing different letters involves different stroke contours, the Doppler spectrogram, which represents hand motions, shows distinctions between different letters. The resulting Doppler spectrogram has 40 frequency bins using the parameters shown in Table 1.

4.2 Handwriting Prediction

After the friction and Doppler spectrograms are derived, *Swift* feeds 1.5s of data of these two streams to deep-learning models for handwriting prediction. Note that this 1.5s detection window is selected because our dataset reveals that most of the handwriting letters are within this duration. We design *Swift* with two models to achieve handwriting recognition - a handwriting detection model and a handwriting recognition model. We first use the detection model to perform a quick binary classification to detect whether the input spectrograms contain handwriting. If so, we forward the spectrograms to the handwriting recognition model to predict the input character. If the detection model does not detect handwriting in the spectrogram, the spectrogram will be discarded. Figure 4 shows the data flow of these two models. When training the models, we employ various data augmentation techniques to enhance the model’s generalizability to users and its robustness against environmental noise.

4.2.1 Model Design. In this section, we first discuss the design of the handwriting recognition model. The architecture of this model is shown in Figure 7, and the model parameters are summarized in the second row of Table 1. The model recognizes on-skin writing by jointly modeling two synchronized signals captured by an

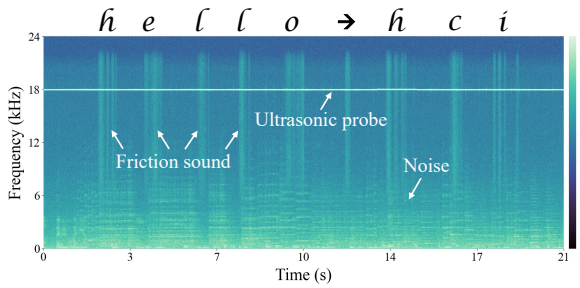


Fig. 5. Raw audio spectrogram.

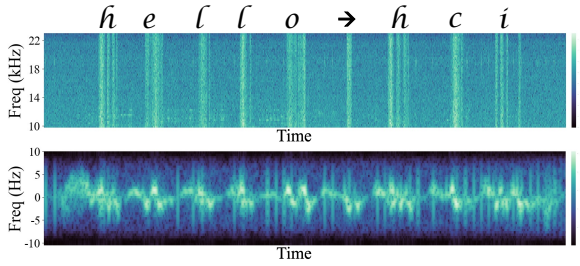


Fig. 6. Friction (top) and Doppler (bottom) spectrograms.

unmodified smartwatch: a friction spectrogram from finger-skin contact and a Doppler spectrogram induced by finger motion. Note that our model performs fusion before encoding. This is to let the encoder learn stroke-relevant patterns from a single, already-integrated representation rather than fusing only at higher layers. The network consists of three stages: (i) input-level cross-modal attention, (ii) a shared residual CNN encoder, and (iii) a temporal classifier with a GRU head.

The input to this model is two spectrograms with different input dimensions. To make the features more consistent, we apply a single-layer Conv1D to project them onto a common dimension. We then apply a minimal bidirectional cross-attention layer to explicitly exchange information across modalities: friction tokens attend to Doppler tokens to incorporate motion cues, while Doppler tokens attend to friction tokens to incorporate contact/friction cues. Rather than introducing a deep transformer stack, we use a single attention module to keep computation lightweight. After the friction and Doppler features are attended to each other, we use a residual gated fusion to combine them:

$$\alpha_F = \sigma(W_F \cdot [O_F;\ O_D]); \quad \alpha_D = \sigma(W_D \cdot [O_F;\ O_D]);$$

$$O_{fuse} = [O_F + \alpha_F \cdot O_F;\ O_D + \alpha_D \cdot O_D],$$

where O_D and O_F are the outputs of the two cross-attention modules, W_D and W_F are learned linear projections, $\sigma(\cdot)$ is the Sigmoid function, which functions as the gating mechanism, and O_{fuse} is the output of the gated fusion. After fusing the two modalities, the fused features are passed through three consecutive ResNet-like convolutional layers for encoding, a GRU layer for processing the temporal features, and an MLP layer for gesture prediction. The output layer of the MLP has 28 units, representing the 26 English letters and the left and right slide gestures.

This model is powerful enough to model the general writing patterns of letters. As we will show in Section 6.2.2, directly applying the trained model to all users yields an F1-score of 0.80 for handwriting prediction. However, different users will have different writing habits. When writing the letter *i*, for instance, some users place the dot first, whereas others add it only after completing the stroke. As a result, we observe significant performance variance across users, with F1-scores ranging from 0.45 to 0.96. To improve *Swift*'s performance for every user, we adopt a personalization phase that adapts the model to a user using a few of their handwriting samples. During adaptation, we only train the GRU and the MLP and freeze other parts. This design provides a fast and efficient way to boost the performance for each individual user.

For the handwriting detection model that functions before the above handwriting recognition model, we adopt a much lighter design due to the simple binary classification task. Specifically, we use the same ResNet encoder architecture discussed above to process the input friction and Doppler spectrogram, and directly append the encoder with a 2-unit linear layer for binary classification.

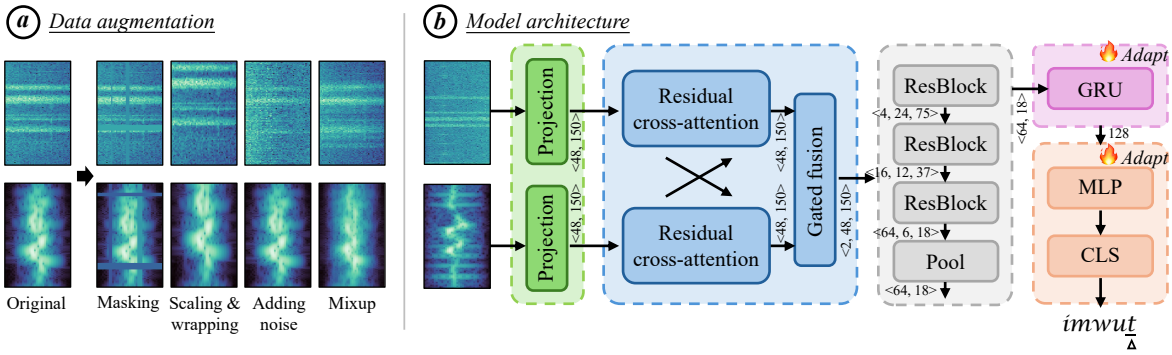


Fig. 7. Handwriting prediction. (a) Data augmentation techniques for training. (b) Prediction model architecture.

4.2.2 Model Training Framework. As discussed in Section 1, *Swift* employs various data augmentation techniques to ensure generalization to new users, as well as the model's robustness to environmental noise. Here, we discuss the methods we use. Illustrative examples of these augmentations are shown on the left-hand side of Figure 7. All the methods are applied on the fly during training.

- **Mixup and CutMix** [71, 76]. We augment the training set by (i) linearly interpolating pairs of samples and their labels (Mixup) and (ii) replacing a cropped region of one sample with a region from another while mixing the labels accordingly (CutMix). These strategies improve generalization and robustness by smoothing the decision boundary and mitigating overconfidence and overfitting. In our design, we apply Mixup or CutMix with a 50% probability and use a mixing coefficient $\alpha = 0.2$.
- **Time-frequency masking** [45]. We randomly mask strips of time and/or frequency features on both spectrograms to force the model to learn generalizable features from the image, rather than focusing on the local characteristics. We use the same augmentation parameters as in [45].
- **Shifting and scaling**. We use these two techniques to simulate cases where the user's handwriting occurs at different timings within the 1.5s detection window and at varying speeds. We randomly shift the sample (-0.5s, 0.5s) along the time and scale the sample with a factor of (0.8, 1.2).

As discussed in Section 4.1, we retain only the near-ultrasound band (10– 22 kHz) of the received audio and discard the remaining frequency components to reduce the impact of environmental noise. Although this filtering suppresses most audible interference (*e.g.*, speech), some noise can extend into the operating band of *Swift* and contaminate the signal, such as engine noise. To improve robustness, we augment the dataset by mixing in common noise sources, encouraging the model to learn handwriting patterns even under noisy conditions.

- **Noise mixing**. We leverage the open-source environmental noise datasets [46, 53] and randomly mix these noise samples with our data samples. The noise sources in these datasets include a wide range of daily sounds, such as conversations, public transportation, household, office, shopping malls, parks, and street sounds. The probability of applying noise mixing is 0.5, and we randomly scale the noise to simulate different noise levels with a scaling factor of $\epsilon \sim \mathcal{N}(1, 0.2^2)$.

We employ the AdamW [38] optimizer to train our model. The loss function is a standard cross-entropy loss with label smoothing regularization ($\alpha = 0.1$) to prevent the model from being overconfident on the training set. Also, gradient clipping (max=1) is used to stabilize training. We schedule the learning rate using cosine annealing [20] and a linear warm-up [37]. The warm-up starts at 1e-3 and gradually increases to 1e-2 within the first 50 epochs (10%), then decays over the next 450 epochs - a configuration similar to that in [67]. The batch size of each epoch is 64 samples.

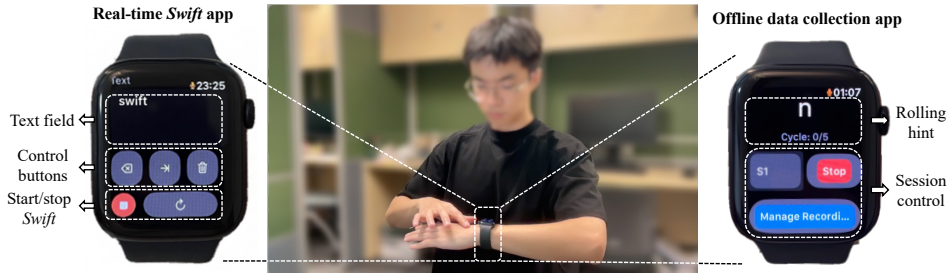


Fig. 8. Data collection and user study setup.

5 IMPLEMENTATION

We implement *Swift* on an Apple Watch Series 7 (2021), with an iPhone 13 (2022) as the transmission relay, and a MacBook Air (2024) as the server running the *Swift* software. The apps on Apple Watch and iPhone are developed in Swift, and the server runs in Python. The Apple Watch transmits an 18kHz sinusoidal signal stored as a WAV file on its hard disk, while recording audio continuously. The recorded audio is sent to the phone in chunks of 0.5 seconds using Apple's WatchConnectivity API. The phone, as a relay, sends the chunk instantly to the server via the WebSocket API through WiFi. After processing the received audio and predicting the handwriting letter or gesture, the server transmits the prediction back to the Apple Watch via the same WebSocket and WatchConnectivity paths. The transmitted audio data is sampled at 48kHz with 16-bit encoding. The chunk size of the real-time data transmission is 1024 bytes. On the server side, the software computes and caches the friction and Doppler spectrograms every 0.2 seconds. After the cache (implemented with a queue) reaches 1.5s in length, the designated prediction window size, the software first checks whether the average spectral power exceeds a threshold (empirically set to 0.2) to preliminarily filter out negative samples. If the power exceeds this threshold, the data will be fed to the deep-learning models for handwriting prediction. After one prediction, the cache will remove the least recent 0.2s data. In other words, the system's prediction step size is 0.2s. Also, after one prediction, the system will set the high-energy part of the friction and Doppler spectrograms into Gaussian noise to prevent its influence on the next character. The deep learning model, implemented in PyTorch, runs on the server's CPU. The real-time app interface is shown on the left-hand side of Figure 8.

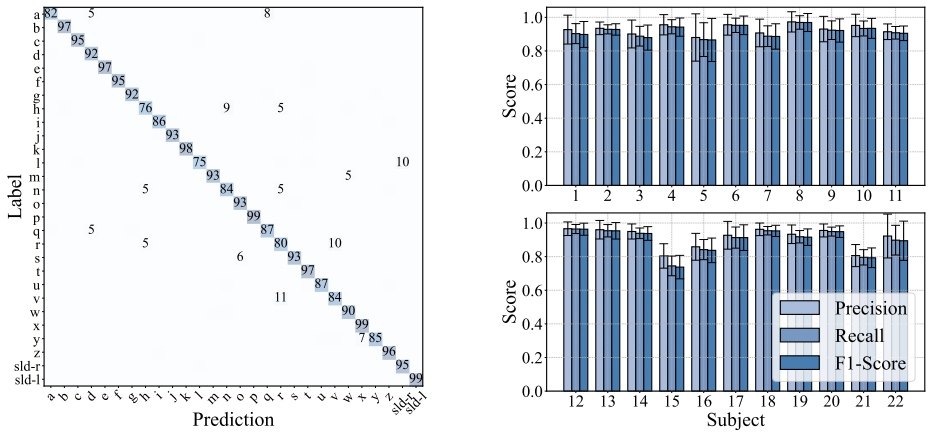
To streamline the data collection process, we also implement an offline version of the software that uses the same audio settings as discussed above. Instead of transmitting the data to the server for handwriting prediction, the data collection app stores the audio locally. Also, the data collection app's UI will display a rolling letter as a hint to the user about the next letter to write, as we discuss in Section 6.1. The data-collection app is shown on the right of Figure 8.

6 TECHNICAL EVALUATION

To study the technical soundness of *Swift* as a new on-skin text entry design, we ask the following research question:

RQ1. How accurately and robustly can Swift recognize handwriting across users and environments, and what are the contributions of its key components?

Here, we evaluate *Swift*'s technical performance in an offline setting, in which the collected data are stored on a hard disk and processed thereafter. We report *Swift*'s handwriting-recognition accuracy, latency, performance in real-world contexts, and the effectiveness of each design component.



(a) Confusion matrix of letters and gestures

(b) *Swift*'s performance on all subjects

Fig. 9. Overall performance. (a) The confusion matrix of handwriting recognition. (b) *Swift*'s performance on each subject.

6.1 Data Collection

This study recruits 22 participants (9 females and 13 males). Most participants are students or staff members from our campus, with an average age of 27.2 years. Among the 22 participants, 6 do not wear a smartwatch daily. Each participant completes 10 data-collection sessions, each lasting approximately 2 minutes. The participant wears the watch according to their own habits, and we do not provide any instructions on how a participant should wear the watch. After the first five sessions, the participant may remove the smartwatch for a few minutes, then rewear it and continue with the remaining sessions. After the experiment, each participant will receive cash compensation. The experiments are conducted in a laboratory with typical activity and background noise, including people walking and talking and air conditioning. The data collection setup is shown in Figure 8.

The data collection experiment begins with an introduction to the study. Subsequently, participants are given time to learn and become familiar with *Swift*. During each data collection session, the participant wears a smartwatch that runs a program that displays visual instructions guiding them to write the target English letters and to perform corresponding hand gestures. The program will repeat for five cycles and then stop. Each letter or gesture will only appear once in one session. This study has been approved by our Institutional Review Board (IRB).

In total, the dataset contains 6,124 samples of handwritten letters and gestures. Note that data samples from some data collection sessions are corrupted due to errors, such as file write failures. These samples are excluded from the dataset. In addition to the letter and gesture samples, there is an additional ≈3,700 seconds of audio used as a blank buffer between consecutive letters/gestures and sessions, which are sampled to generate negative data for training the gesture detection model.

6.2 Handwriting Recognition Performance

Here, we present the result of the above offline study. Note that in this section, to compute *Swift*'s performance on each subject, we first use a leave-one-subject-out (LOSO) validation to train a base model. Then, we use three samples in each class from that subject to personalize the model for them. When evaluating performance, we use the remaining 7 samples per class as the validation set. Notably, all user models are trained using the same set of training parameters as discussed in Section 4.2.2.

Table 2. Latency (W: watch, S: server).

Offline latency						Real-time app latency				
Device	Sound processing		Handwriting recognition		Total	Network (W-S)	Aggregation	Processing + recognition	Network (S-W)	Total
	Friction	*Doppler*	*Detection*	*Recognition*						
Laptop (CPU)	0.95ms	72.5ms	1.97ms	2.06ms	77.5ms	256ms	44.7ms	119ms	271ms	691ms
Server (GPU)	0.98ms	65.3ms	1.44ms	1.22ms	68.9ms					

6.2.1 Overall Performance. We first evaluate the performance of the handwriting detection model - the binary classifier. Note that we do not use personalization for this detection model because the task is straightforward. The precision, recall, and F1 scores of the detection model are 0.97, 0.96, and 0.96, respectively. The performance of the handwriting classification model is shown in Figure 9. Notably, the average precision, recall, and F1-score across all subjects are 0.91, 0.91, and 0.91. While most letters are recognized with fairly good accuracy, we observe that some letters are more likely to be confused with other letters. Here, we list these letters that have the highest false positive rates (FPR): (i) *o* and *s* with an FPR of 6%; (ii) *r*, *v* and *n* can be confused with each other with an FPR up to 11%; (iii) *a*, *d*, *g*, and *q* can be confused with each other with an FPR up to 9%; (iv) *h* can be misclassified into *n* with an FPR of 9%. These confusions arise from inherited similarities in their writing. Even with the complementarity of the two forms of sensing data, these letters remain difficult to distinguish. Future enhancements could adjust the model training parameters and penalize the training loss for these hard classes.

6.2.2 Effectiveness of Personalization. As we discussed in Section 4.2, we apply a personalization technique to customize a model to a new user. Here, we evaluate the effectiveness of this personalization technique. We test the performance using no personalization and personalization with 1-5 samples. To ensure the fairness of the comparison, we use the last five samples from the collected ten samples per subject for evaluation, regardless of how many samples are used for personalization. The result of this evaluation is presented in Figure 10. As the trend suggests, increasing the number of samples generally improves accuracy, ultimately achieving an F1-score of 0.92 with 5 samples. However, performance does not improve significantly beyond three samples. Therefore, we select three samples for personalization to balance the trade-off between performance and user burden.

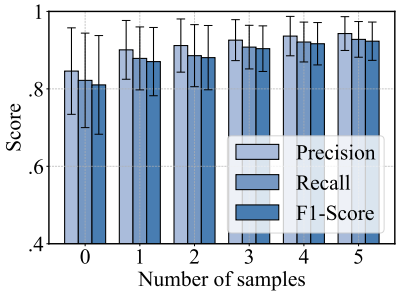
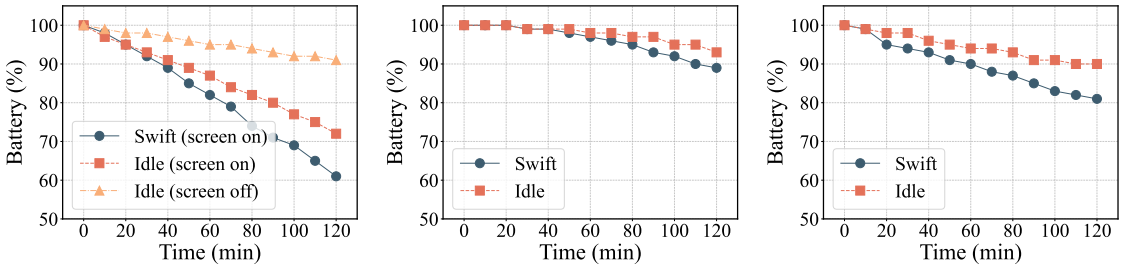


Fig. 10. Effectiveness of personalization.

6.2.3 System Delay. We examine *Swift*'s system latency in this section. We report the average time consumed by the two modules: sound processing and handwriting prediction. We first test the system's modular latency offline. We run *Swift*'s software for five minutes, feeding it randomly generated audio inputs and computing the latency of each component. The average execution time for each component is presented in the left half of Table 2. In general, it takes around 77.43ms for *Swift* to give a gesture prediction. We also conduct the same experiment on a server with higher computational power and GPUs to accelerate model inference, and the results show slightly lower inference times.

The offline tests establish a theoretical lower bound for latency. However, in a real-time deployment, additional engineering factors—most notably network overhead from offloading data to a server via Wi-Fi—increase the overall delay. To evaluate the actual delay perceived by the user in the current implementation, we also measure the end-to-end delay of *Swift*'s real-time application. Specifically, we use a camera to record the user's writing,



(a) Watch (b) Phone (c) Laptop

Fig. 11. Power consumption of *Swift*. (a) On the watch. (b) On the phone. (c) On the laptop.

and label the exact timestamp when the user finishes writing a letter (t_0). Next, we record the timestamp when the prediction window starts to receive a pulse, signaling that the handwriting's audio arrives at the server (t_1). Also, we store the timestamp of a positive threshold check (t_2, discussed in Section 5), implying that the prediction window has aggregated enough data for a prediction. Then, we record the time when a gesture is produced on the server (t_3). Finally, we manually record the timestamp at which the letter appears on the watch from the video (t_4). These five timestamps allow us to decompose the total latency into: user-perceived end-to-end delay ($t_4 - t_0$), network latency ($t_1 - t_0$ and $t_4 - t_3$), data aggregation time ($t_2 - t_1$), and server-side execution time ($t_3 - t_2$). The result is shown on the right of Table 2. Note that the data transmission time between the watch and its paired phone is negligible in our experiments, so we omit this delay in the table. The result indicates an average user-perceived delay of 691ms, which is much larger than the offline study. Notably, network overhead is the dominant factor, accounting for 527ms. In contrast, the combined processing time on the server is only 164ms. These findings suggest that if future iterations are implemented locally on a paired smartphone, rather than relaying data to a server, eliminating network delay could enable sub-200ms character prediction latency. We further discuss the pathway toward achieving commercial-grade responsiveness in Section 9. We also give an analysis of the choice of the 1.5s window size and its impact on latency and accuracy later in Section 6.5.

Even though the current 691ms delay can be perceived to be long for composing long sentences, we believe the system is still usable for short responses, such as *"ok", "brb (be right back)", "on my way", etc.*, especially in the XR scenarios. Also, smartwatch shortcuts, which are originally available only on devices with more sophisticated input modalities, such as keyboards, can be enabled with *Swift*. For example, writing letter *m* opens the map app, letter *t* starts a timer, letter combination *s-o-s* quietly triggers an emergency call. In addition to letters, writing the left/right-slight gesture can be programmed to shortcut functions as well, such as switching to the previous/next music.

6.2.4 Power Consumption. We evaluate *Swift*'s power consumption across the three devices used in our current implementation: a smartwatch (UI and sensing), a smartphone (data relay), and a laptop (server). We run *Swift* continuously for 2 hours and log battery level every 10 minutes on each device, with all devices on battery power and the phone connected to the laptop via WiFi. To measure the worst case, we disable the threshold-based gesture filter (Section 5) so the recognition model runs continuously. It is worth noting that this study presents an extreme use case for the watch, as prolonged typing on it is seldom observed. To decouple *Swift*'s power consumption from other background programs, we run the same test for 2 hours with all three devices in the idle state and record their battery life as the baseline. Because the watch display dominates its energy use and *Swift* keeps it on, we additionally report the watch's idle drain with the screen always on for comparison.

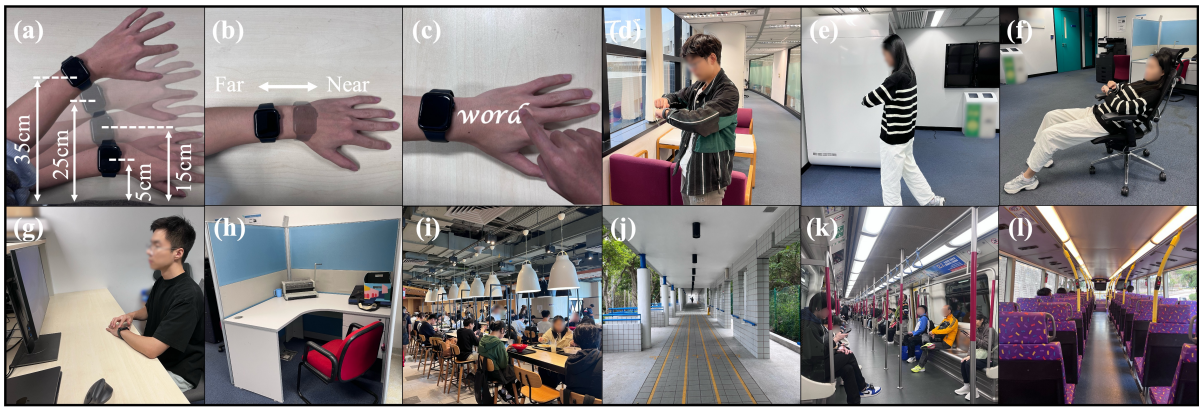


Fig. 12. Scenarios of the robustness study. (a) Different distances between the watch and the chest. (b) Different wearing positions. (c) Writing words. (d) Standing. (e) Walking. (f) Lying down. (g) Heads up. (h) In a lab. (i) In a canteen. (j) On a campus road. (k) Inside a metro train. (l) Inside a bus.

Results in Figure 11 show that, on the watch, 2 hours of idle operation with the screen on consumes about 30% of the battery, while running *Swift* increases this to about 40%, implying an additional ~10% drain attributable to *Swift* under continuous writing. Phone overhead is negligible because it only relays data. On the laptop server, continuous prediction increases battery use by about 10% relative to idle. In practice, server overhead should be lower due to the threshold-based filtering and the use of a lightweight handwriting detection model.

6.3 Robustness Study

In this section, we evaluate *Swift*'s performance in different real-world scenarios. For these experiments, we use the personalized model, trained with three samples per class, to validate performance. Notably, we use the same personalized models as discussed in Section 4 for this evaluation. The only difference is that instead of testing the model's performance on data from a standard lab environment, this evaluation tests across various real-world scenarios.

6.3.1 Variation Across Days. We first test *Swift*'s performance across days. Five participants (two females and three males), who were involved in the initial data collection, are recruited for this experiment approximately two months later. Each participant is asked to contribute three more data collection sessions. We utilize the previously adapted model to evaluate the handwriting recognition performance. The result is shown in the first two columns of Figure 14. The average F1-score drops slightly from 0.93 to 0.90, indicating that the user's writing habits will not change much over a long period and that *Swift*'s performance is stable across days.

6.3.2 User Context. In the standard protocol (Section 6.1), participants write while seated. Here, we evaluate *Swift* under four additional contexts: standing, walking, lying down, and heads-up. For heads-up input, we mirror the watch prompts to a desktop display to simulate mixed-reality use and instruct participants to look at the display instead of the watch. Four participants (two females and two males) are involved in this experiment, with the four scenarios illustrated in Figure 12(d)-(g). The result is shown in columns 10-13 in Figure 14. This result indicates that walking significantly degrades performance (F1=0.53) because body and arm motion induce Doppler shifts that overwhelm the handwriting-induced signal. As illustrated in Figure 13, friction spectrograms remain similar across sitting vs. walking trials, but the walking Doppler spectrogram becomes blurred. To evaluate to what extent these two modalities are affected by such large body motions,

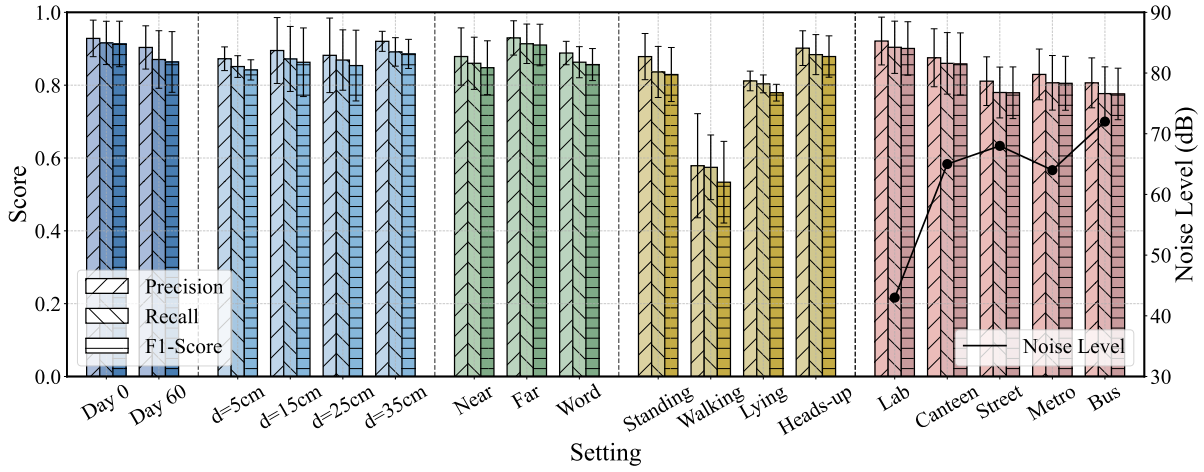


Fig. 14. Robustness study. *Swift*'s performance is compared across days, with different wearing and writing habits, with different user contexts, and in different noisy environments.

we run an additional experiment where we use the single-modality version of *Swift* trained in Section 6.4 to predict handwriting when only the friction or Doppler spectrogram is used. The result shows that during walking, the friction-only model achieves an F1-score of 0.72 (sitting: 0.67), whereas the Doppler-only model achieves only a 0.22 F1-score (sitting: 0.50). This experiment indicates that while body motions have little impact on friction sounds, Doppler shifts are largely compromised, and it is a fundamental limitation that the Doppler shift is sensitive to body motions. Even so, we hypothesize that future work could improve robustness by incorporating large-body motions during data collection or by augmenting training data with motions. To preliminarily validate this, we incorporate the collected data from the walking scenario into our training dataset and retest the results. Note that the only difference between this experiment and the ones from which the results in Figure 14 are produced is that when training the base model, we mix the LOSO training set with other subjects' handwriting data when walking. Our result shows that after mixing the walking data, the F1-score increases immediately from 0.53 to 0.65. Even though this performance increase is limited by the size of the available walking data, it shows good potential for this approach to enhance the robustness of *Swift* in walking scenarios.

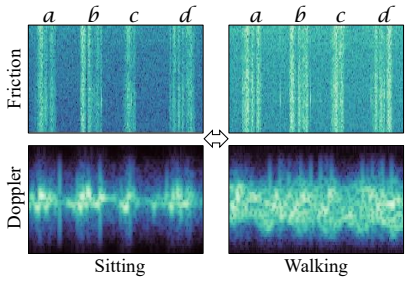


Fig. 13. Handwriting features comparison.

6.3.3 Wearing and Writing Habit. Next, we evaluate *Swift* with different watch-wearing habits. We adjust the distance between the watch and the subject's chest to be 5cm, 15cm, 25cm, and 35cm as described in Figure 12(a). Five participants (two females and three males) are recruited for this experiment, each contributing to three data sessions. The result is shown in the third to sixth columns in Figure 14, where no significant performance degradation is observed. We also evaluate *Swift* under different watch-wearing positions by fixing the watch–chest distance at 15cm and shifting the watch along the wrist. Participants first adjust the band to a comfortable tightness, then move the watch to the left-most and right-most positions to approximate the farthest and nearest distances to the writing area (Figure 12(b)). As shown in the seventh and eighth columns of Figure 14, performance varies negligibly across positions.

Table 3. Ablation study.

Variant	Friction	Doppler	Attention fusion	Mixup & CutMix	Masking	Shift & scaling	Noise-mixing	F1 score *Quiet*	F1 score *Noisy*
1	✓	✗	✗	✗	✗	✗	✗	0.667	0.411
2	✗	✓	✗	✗	✗	✗	✗	0.501	0.392
3	✓	✓	✗	✗	✗	✗	✗	0.823	0.515
4	✓	✓	✓	✗	✗	✗	✗	0.842	0.595
5	✓	✓	✓	✗	✗	✗	✓	0.817	0.744
6	✓	✓	✓	✗	✗	✓	✗	0.871	0.601
7	✓	✓	✓	✗	✓	✗	✗	0.870	0.632
8	✓	✓	✓	✓	✗	✗	✗	0.884	0.642
9	✓	✓	✓	✓	✓	✗	✗	0.894	0.635
10	✓	✓	✓	✓	✓	✓	✗	0.908	0.652
11 (*Swift*)	✓	✓	✓	✓	✓	✓	✓	0.894	0.825

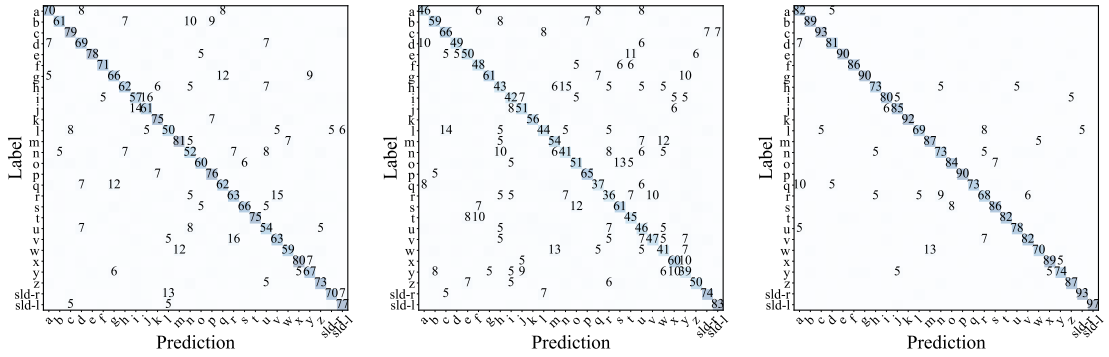
We also envision that *Swift* can support word-level input, where a user can write a word on his/her hand from left to right, resembling the writing experience on paper (Figure 12(c)). We introduce this additional test in which, instead of following the standard data-collection protocol, we ask the user to write a sentence: *"The quick brown fox jumps over a lazy dog."* This sentence includes every letter of the alphabet at least once. When writing this sentence, we ask the participants to write a word from left to right. Note that the current implementation of *Swift* recognizes only a single letter at a time; therefore, participants are asked to write letter by letter, and cursive writing is not allowed. The result is shown in the ninth column of Figure 14. No significant performance variation is observed in the result.

6.3.4 Environmental Noise. Since *Swift* is a sound-based handwriting recognition system, we evaluate its robustness in noisy environments. We recruit three subjects (one female and two males) and ask them to follow the standard data collection protocol (Section 6.1) across five sessions in five different environments: (i) quiet lab, (ii) canteen, (iii) campus road, (iv) metro, and (v) bus. In all sessions, participants remain seated with their hands on their laps. Figure 12(h)-(l) illustrate these environments. We measure ambient noise using a phone app; the corresponding noise levels are 43 dB, 65 dB, 68 dB, 64 dB, and 72 dB. As shown in Figure 14, performance is slightly degraded in noisier settings but remains above a 0.75 F1-score, even on a running bus where noise can exceed 70 dB. Although noise compromises performance to some extent, Section 6.4 shows that, with our augmentation techniques, *Swift* still significantly outperforms the baselines in noisy environments.

6.4 Ablation Study

In this section, we conduct an ablation study to evaluate the effectiveness of the core technical designs in *Swift*. In this study, we use a five-fold cross-validation to train a base model and adapt it to a user with 3-shot adaptation, before evaluating the performance. We first examine the effectiveness of combining the two handwriting-measuring modalities: the friction sound and the Doppler shift. The baseline model begins with a ResNet-based encoder, followed by a GRU and an MLP, as shown in Figure 7. The encoder's input channel is initially set to 1. In the following four design variations, we change the model architectures and do not apply any data augmentation.

- **Variant 1: friction spectrogram only**. We let the model accept friction spectrograms.
- **Variant 2: Doppler spectrogram only**. We let the model accept Doppler spectrograms.
- **Variant 3: naive fusion**. The friction and Doppler spectrograms are concatenated into two tensor channels. The concatenated tensor is then fed to the encoder whose input channel is tuned to 2.



(a) Variant 1: friction only (b) Variant 2: Doppler only (c) Variant 3: naive fusion

Fig. 15. Ablation study: the benefit of combining the friction sounds and the Doppler features.

- **Variant 4: cross-attention fusion**. The complete model architecture. The friction and Doppler spectrograms are fused using the cross-attention module discussed in Section 4.2, before feeding to the encoder blocks.

In the following Variants, we test the effectiveness of the data augmentation techniques while keeping the model architecture—the one that leverages both spectrograms—unchanged.

- **Variants 5-8: using one of the four data augmentations**. In these variants, we only add one data augmentation technique at a time to validate its individual effect.
- **Variants 8-11: gradually adding all data augmentations**. In these variants, we incrementally add mixup/cutmix, masking, shift and scaling, and noise-mixing augmentation techniques to the training pipeline, to evaluate their collective impact.

The results in Table 3 indicate that all designs contribute to the final performance, with the combination of friction sound and the Doppler shift significantly boosting performance, as shown for variants 1-3. Notably, as shown in Figure 15, the naive combination of the two feature modalities resolves much of the ambiguity in recognizing letters, which echoes the fusion design rationale discussed in Section 3. Also, the cross-attention fusion design further enhances performance based on the naive concatenation, comparing variant 4 with 3. We hypothesize that this is because a naive concatenation treats the two modalities equally, even though they carry very different information. As Section 3 shows, for some letters, the friction spectrogram carries more critical information than the Doppler, and sometimes the Doppler carries more weight in some letters. Using an attention-based fusion mechanism with a learned gating function ensures that the weighing is adjustable. In addition to modality fusion, all augmentation techniques have been shown to be effective individually when comparing variants 5-8 with variant 4. Also, their individual enhancements compound when these techniques are stacked together in the training pipeline, as shown by the results for variants 8-11. Notably, the noise-addition technique appears to have a slight negative effect on performance in the quiet condition, as evidenced by comparisons between variants 4 and 5 and between variants 10 and 11. However, this technique significantly boosts the performance under real-world noisy conditions. Note that, in the noisy environment, we evaluate the models' performance on the data collected in Section 6.3, rather than using five-fold cross-validation, as in other tests in this ablation study.

6.5 Impact of Window Size

The current implementation adopts a 1.5s aggregation window for character prediction, a duration that accommodates the majority of handwriting and gesture samples in our dataset. To evaluate the impact of this parameter on *Swift* performance, we additionally test window sizes of 0.5s, 1s, 1.5s, and 2s, measuring character recognition accuracy and end-to-end latency. The resulting accuracies are 0.08, 0.63, 0.91, and 0.91, respectively. Accuracy degrades sharply below the 1s threshold, as the window size is insufficient to capture complete characters. Conversely, increasing the window to 2s yields no further accuracy gains, as the additional 0.5s provides no supplemental information over the 1.5s baseline. The corresponding latencies for these configurations are 669ms, 672ms, 691ms, and 884ms. While reducing the window size marginally lowers latency, network delay remains the dominant factor, as detailed in Section 6.2.3.

7 USER STUDY 1: USABILITY AND USER EXPERIENCE

In addition to the offline performance evaluation, we conduct a user study to evaluate the usability of *Swift*. We implement a real-time version of *Swift* as a WatchOS app through which users can input text with *Swift*. Through this study, we want to answer the following research question:

RQ2. In a real-time deployment, how usable is *Swift* for phrase composition, and what breakdowns or workflow frictions emerge?

7.1 App Design

The app interface on the watch is shown on the left-hand side of Figure 8. It includes a text input field that displays the recognized handwritten characters. If the detected input is a right or left slide, the gesture will be translated into a space or backspace on the text field. The interface also features several buttons that allow users to control the text field.

7.2 Study Procedure

Eight participants (three female; mean age 25) take part in this user study. All have previously participated in the data-collection study (Section 6). The study setup is shown in the middle of Figure 8. After a brief introduction, participants are taught to use the text input app and given time to practice writing with *Swift*. The relaying smartphone and server are hidden from the users. The introduction and practice phase typically lasts about five minutes. When the participant is ready, the researcher verbally prompts them to write all 26 English letters and the two control gestures in sequence, and reminds them to consider whether on-skin writing feels easy and comfortable. Participants then use the app for a few minutes to enter any text they want. Each session typically lasts about 10 minutes. After the session, participants complete the System Usability Scale (SUS) questionnaire [8] and a customized questionnaire on perceived experience and acceptability, adapted from prior work [27, 49, 60, 63].

- 1. Enjoyment. "I feel it is enjoyable to input text with *Swift*."
- 2. Efficiency. "I believe *Swift* provides an efficient text input method."
- 3. Fatigue. "I don't think using *Swift* makes me tired."
- 4. Social concerns. "I don't think using *Swift* in public spaces would raise social concerns."
- 5. Distraction. "I don't think *Swift* is producing any noise."
- 6. Screen blockage. "I don't think using *Swift* will block my vision to see the screen."

Additionally, we ask participants to rate the difficulty and level of comfort associated with writing each letter and gesture when using *Swift*. The specific question we ask is:

- "I believe writing [letter/gesture name] on the back of my hand is easy and comfortable."

All the above questions are answered using a 5-point scale, with the lowest score indicating strong disagreement and the highest score indicating strong agreement. At the end of the study, we conduct an exit interview with each participant to learn about their experience and subjective comments about *Swift*. The experiment takes 20-30 minutes in total.

To quantitatively assess user proficiency with *Swift*, we conduct a transcription test with 7 participants (two females, five males). We randomly select 20 sentences from MacKenzie's sentence set [40] and split them into four blocks. Participants transcribe each block continuously, with 2-minute breaks between blocks. We record block completion time to compute words per minute (WPM). As a baseline, participants complete the same test using Apple Watch Scribble [2] - the watch's built-in handwriting text entry method. As a comparison, we also run the transcript test with other commercial text-entry methods on smart watches - on-screen keyboard and text dictation - both with Apple Watch's default functions. The test lasts 50-60 minutes.

7.3 Study Results

The questionnaire result of the study is presented in Figure 16. Note that in SUS, odd questions are positively worded and even questions are negatively worded.

7.3.1 System Usability. The average SUS item scores appear at the top of Figure 16. Using the method in [8], we compute an overall system usability (SU) score of 87.5/100, indicating high usability of *Swift*. In SUS, Q4 and Q10 reflect learnability, whereas the remaining items assess usability; separating them yields learnability and usability scores of 98.2 and 84.8, respectively, suggesting that *Swift* is easy to learn and use. Participants also praise its intuitiveness: *"I had no problem at all learning to use the system. You just write what you want to say on the skin"* (P2).

Notably, we observe that the eighth question, the one regarding the cumbersome use of the system, receives slightly more negative ratings than the others. We suspect this is because, in some cases, when the system misrecognizes the user's input, they need to write a backspace gesture to delete the incorrect input and then rewrite the letter. According to some participants, this may cause a burden to the user. *"I got frustrated when the system wrongly recognized my writing, especially since I needed to write a backspace gesture again ... I would rather use the button on the screen to do a backspace."* (P3). This can be addressed by integrating function buttons into the handwriting workflow, based on user preference.

7.3.2 Perceived Experience and Acceptability. The average scores of the second questionnaire are shown at the bottom of Figure 16. Overall, all participants agree that using *Swift* for text input is enjoyable, and writing on the skin does not obstruct the user's view. P1 said *"This is a new method for text input, and I felt very excited trying it out...I have never thought about using my skin as a tracking pad."* While most participants believe using *Swift* brings no social concerns and causes little fatigue, some participants are more reserved in Q3 and Q4. Also, a few participants stay neutral in Q2: efficiency. We hypothesize that this is partially due to the 0.7s perceived delay in recognizing characters (Section 7.1), compared with conventional on-screen keyboards or writing input, where feedback is instantaneous. *"The response time seemed to be long if compared with commercial touch screens."* (P7). As we will discuss in Section 9, future work should explore ways to optimize the implementation to reduce response time. We also summarize users' perceptions of writing all the letters and gestures on the hand in Appendix A.

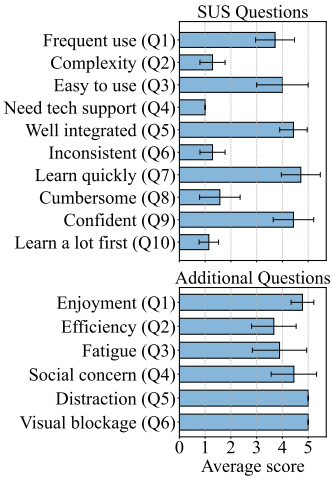


Fig. 16. Questionnaire results.

six daughters and seven sons	*six daaghters and seoen sohs*
house with new electrical panel	*house with hew electrical pahel*
i just cannot figure this out	*i just cahnot figure thrs sut*
they watched the entire movie	*they watfld the ehtire msvic*
the king sends you to the tower	*the kihg sends ysu to the tower*
(a) Sentences to be transcribed	(b) Sentences transcribed by *Swift*

Fig. 17. An example of text transcription. (a) The text. (b) The transcribed text.

7.3.3 Other Comments. Here, we summarize other comments and suggestions from the participants that haven't been discussed above. **Touch typing**. P1 and P6 both appreciate the tactile feedback of on-skin writing, where the user does not need to constantly look at the strokes. *"I think this system should be a very useful touch writing tool in occasions where you can't move away from your eyes, such as in social events."* (P6). **Word-level recognition**. Some users suggest word-level recognition in the future (P2, P6, P7). *"It would be great if I could write a complete word on my skin ... or even cursive writing, which should significantly boost the input efficiency"* (P2). **Control gestures**. P3 argues that some controls could be moved back to the screen: *"...writing on the skin is a good alternative, but I have no problem clicking buttons on the watch screen... [future designs] can integrate the control function as buttons on the screen."*

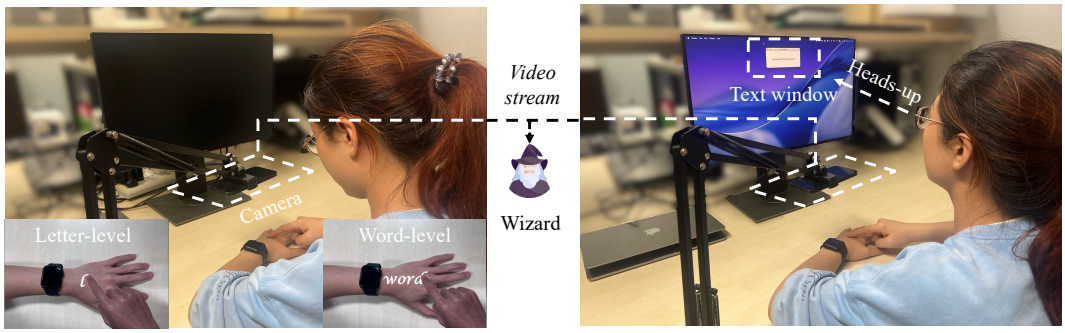
7.3.4 Text Entry Speed and Accuracy. Our transcription test shows that *Swift* achieves 5.7 WPM while Apple Watch Scribble achieves 6.5 WPM. We also compute the average character error rate (CER) to quantify letter-prediction accuracy. We determined the CER by calculating the Levenshtein distance between the predicted and reference texts, then dividing it by the length of the reference string [54]. As a result, *Swift* achieves a CER of 11.6%, which translates into a character-level F1-score of 0.88. As a comparison, Apple Watch Scribble achieves a CER of 6.1%, equivalent to an F1-score of 0.94 in character recognition. An example of the transcribed text are shown in Figure 17. Notably, there is still a gap between *Swift* and commercial-grade accuracy. This can be solved by using more samples for personalization as discussed in Section 6.2.2. Additionally, using a language model as the prediction prior, incorporating top-k prediction, and using auto-correct engines may further enhance performance. We also discuss other pathways to enhance performance in Section 9. Even so, we believe this study demonstrates a strong potential for our proof-of-concept design, which serves as the first step towards commercial-grade accuracy. In the above comparison, *Swift* and Scribble are both handwriting recognition-based text entry methods. We also compare other commercial watch-based text-entry methods, namely, on-screen keyboard input and text dictation. For the on-screen keyboard, it achieves a 15.9 WPM and the CER is 4% (0.97 F1-score). The voice dictation achieves 89.9 WPM and the CER is 1% (0.99 F1-score), where the recruited participants are not native English speakers, so text entry speed among other user groups may differ.

8 USER STUDY 2: WIZARD-OF-OZ COMPARISON WITH SCREEN-BASED TEXT ENTRY

The above studies have demonstrated *Swift*'s strong usability as a new text-entry method for smartwatches. In this section, we further examine users' preferences for *Swift* relative to other text input methods on smartwatches. Specifically, the following research question is asked:

RQ3. How does *Swift* compare to touchscreen handwriting in user preference and perceived suitability across text-entry scenarios, and in which scenarios do *Swift*'s benefits become most salient?

To better understand users' perceptions of on-skin text entry without current engineering constraints (e.g., latency and occasional misrecognitions), we run a Wizard-of-Oz study to simulate an ideal *Swift* experience. We



(a) Scenarios 1 and 2: letter and word level text entry

(b) Scenario 3: heads-up text entry

Fig. 18. Wizard-of-Oz study setup.

envision the following three scenarios across different levels of text-entry demands (Figure 18). **(i) Letter-level text entry.** The standard use case where users write in a letter-by-letter manner. **(ii) Word-level text entry.** Users perform word-level text entry, where a word is written from left to right, as they would on paper, with cursive writing allowed. This scenario targets a more advanced, natural, and potentially more efficient workflow than letter-by-letter input. **(iii) Heads-up text entry.** *Swift* can be used for interaction in AR/VR/MR, where the user can keep engaged with a head-mounted display without switching attention and glancing down to write on a screen - heads-up interaction. In this scenario, we simulate it by asking participants to keep their heads up and to look at an eye-level screen that displays their input.

8.1 Study Procedure

The study takes place in a quiet lab. Participants sit at a table with their watch-worn hand resting on the tabletop, while a camera records their handwriting. A researcher is present throughout. For each scenario, the researcher selects sentences from MacKenzie's set [40] for transcription; each scenario is completed twice, once with *Swift* and once with the touchscreen, with method order randomized. After each scenario, participants complete a SUS questionnaire comparing the two methods, and an exit interview follows at the end.

For *Swift* trials, the camera on the table streams the video to another researcher—the wizard—who is in the next room of the experiment venue and, unbeknownst to the participant, watches the feed, infers the intended letters/words, and enters them on the watch (or, in the heads-up scenario, the screen) via a computer program. For touchscreen trials, we ask the participants to use Apple Watch Scribble—the built-in handwriting text-entry method.

We recruit eight participants (3 females and 5 males, aged 21 to 32 years) in this study. All of these participants are students and from our campus. Notably, these participants have not participated in our previous studies, so they have no prior experience with *Swift* to rule out potential biases. The study takes about 30 minutes per participant.

8.2 Study Results

The questionnaire result of this study is presented in Figure 19. Overall, participants prefer *Swift* to the touchscreen, and the preference becomes more prominent when more demanding text-entry tasks are introduced.

In the first scenario, where the participants apply letter-by-letter handwriting, the average SU value for *Swift* is 86.39, and the average SU for the built-in touchscreen method is 82.67. While both scores are high, *Swift* achieves

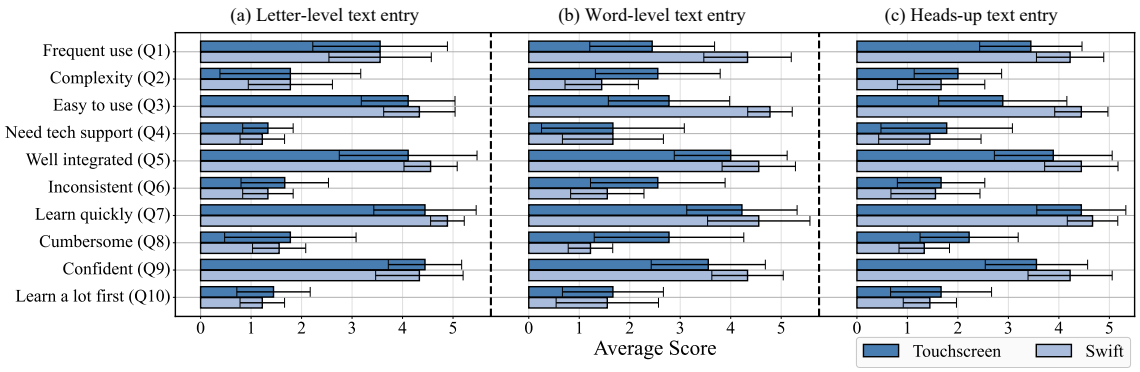


Fig. 19. SUS questionnaire scores of the Wizard-of-Oz study. (a) The letter-level text entry scenario. (b) The word-level text entry scenario. (c) The heads-up text entry scenario.

a much higher usability subscore (84.62 vs 78.67). While agreeing on the larger space provided by *Swift*, both P2 and P7 said they do enjoy the real-time stroke trajectory display on the touchscreen. *"I like the real-time trajectory feedback provided by the touchscreen* (P2)." We believe that future work could consider predicting the real-time stroke trajectory as an auxiliary task, providing natural feedback that resembles pen use.

In the second scenario for word-level input, the SU for *Swift* is 87.78 and it is 64.44 for the touchscreen. In this case, the perceived system usability of *Swift* is significantly higher than that of the touchscreen. According to the interview, all participants commented that the touchscreen is too small to write a complete word, whereas the back of the hand provides much more space for writing. P1 noted: *"The watch's touchscreen is too small for writing a complete word, such as 'electricity', where I need to plan the size of each writing letter and the starting position of writing the first letter. Otherwise, I will run out of space halfway through a word."* P3 and P7 also share a similar concern: *"I suppose the touchscreen can at most host three letters side-by-side"* (P7). This result shows a clear preference for using *Swift* for the more efficient word-level handwriting text input.

In the heads-up interaction scenario, the SU for *Swift* is 86.39 while for the touchscreen, it is 72.22, indicating the users prefer *Swift* strongly over the touchscreen in heads-up scenarios. The major reason for this choice is that writing on the skin provides tactile feedback that confirms each stroke, whereas navigating on the screen is harder without direct visual cues, as P5 noted: *"I may need some time to accurately put my finger on the watch if not looking at it... I think I sometimes accidentally write some strokes outside of the screen."* P6 expressed his strong support for the use of *Swift* in heads-up scenarios: *"The first two scenarios are good alternatives for text input, but not always a must. However, a smartwatch may become an essential accessory for future AR glasses because AR workflows break down when users have to shift their attention. Using the skin keeps text entry readily available without breaking focus."*

Overall, we observe a clear preference for *Swift* over the touchscreen in the WoZ study, particularly for the more demanding heads-up and word-level input tasks. The major reasons are the skin's larger input space and its tactile feedback. We believe the findings from the above two user studies (Sections 7-8) show strong potential for *Swift* as a next-generation text-entry technique that becomes increasingly advantageous as input demands grow and visual attention becomes scarce. We also acknowledge that in this WoZ study, we simulate *Swift* with perfect recognition accuracy and latency, while the actual usability may still depend on these factors, as we will discuss in Section 9. Also, users' preferences of this new on-skin text entry method against non-screen input methods, such as voice/whispered dictation, should be evaluated in the future.

9 DISCUSSION

In this section, we discuss the limitations and the future direction of *Swift*. This discussion includes the potential for commercial-grade performance, the interaction scope of *Swift*, and the practicality and real-world usability.

9.1 Pathways to Commercial-Grade Performance

In Section 6, we evaluate the technical performance of *Swift* as a proof-of-concept system to support handwriting recognition on skin. In this discussion, we analyze the pathways to achieve commercial-grade accuracy and responsiveness.

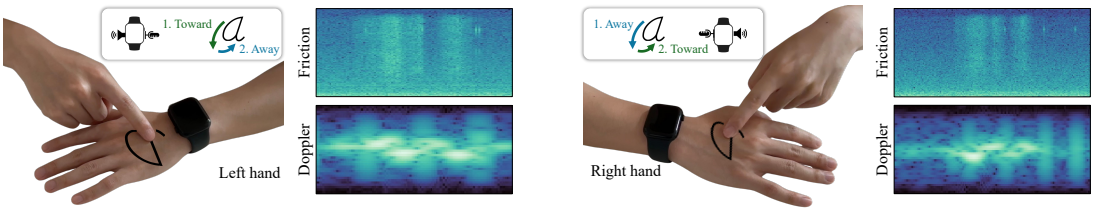
Accuracy. The technical evaluation shows that some letters are more easily misrecognized as other letters. This is partly because of the inherited similarity of the strokes in these letters. To improve performance on these cases, future research may leverage publicly available letter datasets that include a wide variety of letters from a broader population [14, 24, 41] to enhance the model's capabilities. Also, integrating language prior and top-k predictions into the pipeline may further enhance the performance.

Latency. Beyond recognition accuracy, our evaluation reveals an end-to-end latency of approximately 0.7s, primarily stemming from current system overhead. We propose four strategies to mitigate this delay. (i) On-device processing: As discussed in Section 6.2.3, a significant portion of the total latency is attributed to network transmission between the smartwatch and the server. Future implementations should migrate the inference pipeline to the paired smartphone. This is highly feasible given the robust computational power of modern mobile chipsets and would substantially eliminate communication bottlenecks. To briefly demonstrate the feasibility of an on-watch implementation, we run a simulation on the watch. We adopt the same strategy as in the offline latency study in Section 6.2.3, where we implement the system pipeline on the watch and feed the pipeline with random audio data for five minutes. The result shows that the total delay for one gesture prediction is 304 ms on the watch, with signal processing taking 291 ms and model inference around 13 ms. This latency is much smaller than the current watch-laptop implementation. In addition, we also believe that engineering optimization can reduce latency further, as we will discuss next. (ii) Software optimization and streaming inference: Future iterations can improve responsiveness through more efficient buffer coordination and the implementation of streaming recognition. By processing data segments incrementally as the user writes, the system can generate preliminary predictions that are refined upon stroke completion, thereby reducing the user-perceived waiting time. (iii) Word-level and cursive support: Transitioning from discrete letter recognition to word-level modeling would drastically enhance text-entry throughput. Given *Swift*'s high character-level accuracy, integrating language models and probabilistic word-prediction frameworks similar to those in [19, 54] could allow for successful word completion even with partial or uncertain inputs. (iv) Adaptive windowing: While the current 1.5s window accommodates all character lengths, it introduces unnecessary idle time for shorter strokes. Future work could employ an adaptive triggering mechanism using a lightweight classifier to detect stroke termination in real-time. Commencing inference immediately upon detection would optimize the trade-off between recognition accuracy and temporal efficiency.

9.2 Enhancing Practicality and Real-world Usability

Section 7 demonstrates the good usability of the current proof-of-concept implementation of *Swift*, and Section 8 further shows its high potential if *Swift* is used in its ideal capacity. Here, we discuss the future direction to enhance the system's practicality if deployed to real-world users.

Handedness. In this paper, we primarily focus on the design of right-handed use cases, where the user wears the watch on the left hand and writes with the right hand. Intuitively, the same technology can be applied to left-handed users, but with a few necessary modifications. As shown in Figure 20, the hand motions of a left-handed user are roughly symmetric to those of a right-handed user from the perspective of the watch when



(a) Right handed (b) Left handed

Fig. 20. Handedness in *Swift*. (a) The watch is worn on the left hand, and the right hand is used for writing. (b) The watch is worn on the right hand, and the left hand is used for writing.

writing letters. For example, when writing the letter *w*, a right-handed user's writing finger gradually moves away from the watch, whereas a left-handed user's writing finger moves towards the watch. This symmetry will result in opposite Doppler patterns for these two user groups. Therefore, to support both right-hand and left-hand users, future development should also use a left-handed dataset to train a separate model, and the watch application should invoke different models based on the user's handedness configuration.

Handwriting complex characters. In addition to the Latin alphabet, other languages may involve more complex characters for which *Swift* can provide a larger space to write these characters. For example, Chinese, Japanese, and Korean characters involve more strokes and are more structured. *Swift*'s mix of touch sound and motion is naturally good at noticing stroke shape, order, and direction. Therefore, it is feasible to improve *Swift* to support more languages.

Generalization to other watch modes. The current *Swift* implementation is on an Apple Watch Series 7 (2021), while different watch models have different types of speakers and microphones and different layouts. Therefore, future work should examine how the system might generalize well to other devices, such as the effectiveness of personalization on new devices.

9.3 Future Design Considerations

In this work, the primary design space is handwriting characters on the back of the hand. In addition to this, we see a good potential for various interaction methods supported by the underlying techniques of *Swift*.

Flexible interaction design. As suggested by our user study (Section 7.3), future work should explore integrating the traditional virtual keyboard with on-skin handwriting. For example, the controls—space and backspace—can be implemented as virtual buttons, while the letters are entered via handwriting on the skin. Also, in *Swift*, the writing location is on the back of the hand because of its proximity to the watch. According to previous research [58], the forearm is also a well-accepted location for on-skin input, and it is conveniently located near the watch. One potential benefit of using the forearm for handwriting input is that it is smoother and flatter than the back of the hand. Therefore, future research should also examine the feasibility of forearm-based handwriting recognition. Additionally, the palm is well-suited to handwriting and warrants further exploration.

Continuous trackpad. Beyond handwriting and discrete gestures, the underlying sensing modalities of *Swift* have good potential to support continuous 2D tracking, effectively turning the back of the hand into an always-available trackpad. The same fusion of passive friction acoustics and active ultrasonic Doppler that enables *Swift* to recognize character strokes can also be repurposed to track the continuous motion of a finger across the skin, with friction sounds indicating contact and Doppler shifts providing continuous velocity and direction. By training a model to map these acoustic signatures to 2D coordinates, *Swift* could enable a new range of interactions, such as controlling a cursor, navigating menus, or panning and zooming on a map without

touching the watch screen. This would be particularly valuable in heads-up computing scenarios with AR/VR glasses where a continuous, private input method is needed.

End-to-end prediction pipeline. Following prior research [11, 28, 79] that utilizes non-speech sounds for interaction design, we adopt a traditional time-frequency analysis combined with deep learning to predict handwritten characters. However, given the emergence of end-to-end audio processing architectures [7, 48, 72], future work should explore incorporating such end-to-end designs to further improve predictive performance.

10 CONCLUSION

Swift turns the back of the hand into a practical handwriting pad using an unmodified smartwatch, fusing passive friction acoustics with active ultrasonic Doppler to distinguish similar letters and remain robust in everyday noise. Across three studies, *Swift* achieves strong offline recognition (macro F1=0.91 after few-shot personalization), demonstrates high usability and learnability in a real-time prototype (N=8), and is preferred over touchscreen handwriting in Wizard-of-Oz comparisons (N=8), especially for more demanding scenarios such as word-level and heads-up text entry. These results suggest that *Swift*'s larger, continuous on-skin canvas and fluid workflow deliver clearer benefits as task complexity and situational demands rise, while remaining deployable on commodity hardware without added sensors. Looking forward, moving from isolated letters to word-level and cursive input, extending to non-Latin scripts via stroke- and component-aware decoding, and refining on-device efficiency and rapid personalization can further improve accuracy, speed, and accessibility, positioning *Swift* as a viable path to private, eyes-up text entry on smartwatches.

Acknowledgments

We acknowledge that we have used GPT-5.5 for grammatical checking, and that Figure 1 was generated with Google Nano Banana using real photos. Written consents have been obtained from the volunteers for the use of their photos in illustrative figures in this paper. This research is supported in part by Hong Kong RGC under Contracts CERG 16205824, 16204523, SZSTI25EG20, SRFS2425-6S05, AoE/E-601/22-R, 25260930Z002, and HB016.

References

[1] 2025. Bixby | Apps & Services | Samsung US. https://www.samsung.com/us/apps/bixby/ Accessed: September 2025.
[2] 2025. Enter text on Apple Watch - Apple Support. https://support.apple.com/en-jo/guide/watch/apdaf7837856/watchos Accessed: November 2025.
[3] 2025. Google Assistant, your own Google. https://assistant.google.com/ Accessed: September 2025.
[4] 2025. Siri - Apple. https://www.apple.com/siri/ Accessed: September 2025.
[5] Christoph Amma, Marcus Georgi, and Tanja Schultz. 2012. Airwriting: Hands-Free Mobile Text Input by Spotting and Continuous Recognition of 3d-Space Handwriting with Inertial Sensors. In *2012 16th International Symposium on Wearable Computers*. 52–59. doi:10.1109/ISWC.2012.21 ISSN: 2376-8541.
[6] Oscar Javier Ariza Nunez, André Zenner, Frank Steinicke, Florian Daiber, and Antonio Krüger. 2022. Holitouch: Conveying holistic touch illusions by combining pseudo-haptics with tactile and proprioceptive feedback during virtual interaction with 3duis. *Frontiers in Virtual Reality* 3 (2022), 879845. doi:10.3389/frvir.2022.879845
[7] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems* 33 (2020), 12449–12460.
[8] Aaron Bangor, Philip T. Kortum, and James T. Miller. 2008. An Empirical Evaluation of the System Usability Scale. *International Journal of Human–Computer Interaction* 24, 6 (July 2008), 574–594. doi:10.1080/10447310802205776
[9] Andreas Braun, Stefan Krepp, and Arjan Kuijper. 2015. Acoustic tracking of hand activities on surfaces. In *Proceedings of the 2nd international Workshop on Sensor-based Activity Recognition and Interaction, iWOAR 2015, Rostock, Germany, June 25-26, 2015*, Bodo Urban and Thomas Kirste (Eds.). ACM, 9:1–9:5. doi:10.1145/2790044.2790052
[10] Anish Byanjankar, Yunxin Liu, Yuanchao Shu, Insik Shin, Myeongwon Choi, and Hyosu Kim. 2023. S-UbiTap: Leveraging Acoustic Dispersion for Ubiquitous and Scalable Touch Interface on Solid Surfaces. *IEEE Trans. Mob. Comput.* 22, 11 (2023), 6800–6816. doi:10.1109/TMC.2022.3195226

[11] Shumin Cao, Xin He, Peide Zhu, Mingshi Chen, Xiangyang Li, and Panlong Yang. 2018. iPand: Accurate Gesture Input with Ambient Acoustic Sensing on Hand. In *2018 IEEE 37th International Performance Computing and Communications Conference (IPCCC)*. IEEE, Orlando, FL, USA, 1–8. doi:10.1109/PCCC.2018.8710858

[12] Wenqiang Chen, Lin Chen, Yandao Huang, Xinyu Zhang, Lu Wang, Rukhsana Ruby, and Kaishun Wu. 2019. Taprint: Secure Text Input for Commodity Smart Wristbands. In *The 25th Annual International Conference on Mobile Computing and Networking (MobiCom '19)*. Association for Computing Machinery, New York, NY, USA, 1–16. doi:10.1145/3300061.3300124

[13] Wenqiang Chen, Maoning Guan, Yandao Huang, Lu Wang, Rukhsana Ruby, Wen Hu, and Kaishun Wu. 2018. ViType: A Cost Efficient On-Body Typing System through Vibration. In *2018 15th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. 1–9. doi:10.1109/SAHCN.2018.8397098 ISSN: 2155-5494.

[14] T. E. de Campos, B. R. Babu, and M. Varma. 2009. Character recognition in natural images. In *Proceedings of the International Conference on Computer Vision Theory and Applications, Lisbon, Portugal*.

[15] Fengyi Fang, Hongwei Zhang, Lishuang Zhan, Shihui Guo, Minying Zhang, Juncong Lin, Yipeng Qin, and Hongbo Fu. 2022. Handwriting Velcro: Endowing AR Glasses with Personalized and Posture-adaptive Text Input Using Flexible Touch Sensor. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 4 (Dec. 2022), 1–31. doi:10.1145/3569461

[16] Mayank Goel, Brendan Lee, Md Tanvir Islam Aumi, Shwetak N. Patel, Gaetano Borriello, Stacie Hibino, and Bo Begole. 2014. SurfaceLink: using inertial and acoustic sensing to enable multi-device interaction on a surface. In *CHI Conference on Human Factors in Computing Systems, CHI'14, Toronto, ON, Canada - April 26 - May 01, 2014*, Matt Jones, Philippe A. Palanque, Albrecht Schmidt, and Tovi Grossman (Eds.). ACM, 1387–1396. doi:10.1145/2556288.2557120

[17] Jun Gong, Aakar Gupta, and Hrvoje Benko. 2020. Acustico: Surface Tap Detection and Localization using Wrist-based Acoustic TDOA Sensing. In *UIST '20: The 33rd Annual ACM Symposium on User Interface Software and Technology, Virtual Event, USA, October 20-23, 2020*, Shamsi T. Iqbal, Karon E. MacLean, Fanny Chevalier, and Stefanie Mueller (Eds.). ACM, 406–419. doi:10.1145/3379337.3415901

[18] Jun Gong, Zheer Xu, Qifan Guo, Teddy Seyed, Xiang 'Anthony' Chen, Xiaojun Bi, and Xing-Dong Yang. 2018. WrisText: One-handed Text Entry on Smartwatch using Wrist Gestures. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems, CHI 2018, Montreal, QC, Canada, April 21-26, 2018*, Regan L. Mandryk, Mark Hancock, Mark Perry, and Anna L. Cox (Eds.). ACM, 181. doi:10.1145/3173574.3173755

[19] Mitchell L. Gordon, Tom Ouyang, and Shumin Zhai. 2016. WatchWriter: Tap and Gesture Typing on a Smartwatch Miniature Keyboard with Statistical Decoding. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, San Jose, CA, USA, May 7-12, 2016*, Jofish Kaye, Allison Druin, Cliff Lampe, Dan Morris, and Juan Pablo Hourcade (Eds.). ACM, 3817–3821. doi:10.1145/2858036.2858242

[20] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. 2017. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677* (2017). doi:10.48550/arXiv.1706.02677

[21] Đorđe T Grozdić and Slobodan T Jovičić. 2017. Whispered speech recognition using deep denoising autoencoder and inverse filtering. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 25, 12 (2017), 2313–2322. doi:10.1109/TASLP.2017.2738559

[22] Chris Harrison, Julia Schwarz, and Scott E. Hudson. 2011. TapSense: enhancing finger interaction on touch surfaces. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, Santa Barbara, CA, USA, October 16-19, 2011*, Jeffrey S. Pierce, Maneesh Agrawala, and Scott R. Klemmer (Eds.). ACM, 627–636. doi:10.1145/2047196.2047279

[23] Yasha Iravantchi, Yi Zhao, Kenrick Kin, and Alanson P. Sample. 2023. SAWSense: Using Surface Acoustic Waves for Surface-bound Event Recognition. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, CHI 2023, Hamburg, Germany, April 23-28, 2023*, Albrecht Schmidt, Kaisa Väänänen, Tesh Goyal, Per Ola Kristensson, Anicia Peters, Stefanie Mueller, Julie R. Williamson, and Max L. Wilson (Eds.). ACM, 422:1–422:18. doi:10.1145/3544548.3580991

[24] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2016. Reading Text in the Wild with Convolutional Neural Networks. *International Journal of Computer Vision* 116, 1 (jan 2016), 1–20. doi:10.1007/s11263-015-0823-z

[25] Yei Hwan Jung, Jae-Hwan Kim, and J. Rogers. 2020. Skin-Integrated Vibrohaptic Interfaces for Virtual and Augmented Reality. *Advanced Functional Materials* 31 (2020). doi:10.1002/adfm.202008805

[26] Vimal Kakaraparthi, Qijia Shao, Charles J Carver, Tien Pham, Nam Bui, Phuc Nguyen, Xia Zhou, and Tam Vu. 2021. FaceSense: sensing face touch with an ear-worn system. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 5, 3 (2021), 1–27. doi:10.1145/3478129

[27] Daehwa Kim, Keunwoo Park, and Geehyuk Lee. 2021. AtaTouch: Robust Finger Pinch Detection for a VR Controller Using RF Return Loss. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. ACM, Yokohama Japan, 1–9. doi:10.1145/3411764.3445442

[28] Daehwa Kim, Eric Whitmire, Roger Boldu, Wolf Kienzle, and Hrvoje Benko. 2024. SoundScroll: Robust Finger Slide Detection Using Friction Sound and Wrist-Worn Microphones. In *Proceedings of the 2024 ACM International Symposium on Wearable Computers, ISWC 2024, Melbourne, VIC, Australia, October 5-9, 2024*, Vassilis Kostakos, Judy Kay, and Thuong N. Hoang (Eds.). ACM, 63–70.

[29] Hyosu Kim, Anish Byanjankar, Yunxin Liu, Yuanchao Shu, and Insik Shin. 2018. UbiTap: Leveraging Acoustic Dispersion for Ubiquitous Touch Interface on Solid Surfaces. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems, SenSys 2018, Shenzhen, China, November 4-7, 2018*, Gowri Sankar Ramachandran and Bhaskar Krishnamachari (Eds.). ACM, 211–223. doi:10.1145/3274783.3274848

[30] Jiwan Kim, Chi-Jung Lee, Hohurn Jung, Tianhong Catherine Yu, Ruidong Zhang, Ian Oakley, and Cheng Zhang. 2026. WatchHand: Enabling Continuous Hand Pose Tracking On Off-the-Shelf Smartwatches. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems (CHI '26)*. Association for Computing Machinery, New York, NY, USA, Article 740, 21 pages.

[31] Jiwan Kim, Jiwan Son, and Ian Oakley. 2025. Cross, Dwell, or Pinch: Designing and Evaluating Around-Device Selection Methods for Unmodified Smartwatches. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*. ACM, Yokohama Japan, 1–11. doi:10.1145/3706598.3714308

[32] Chentao Li, Ziheng Xi, Jianjiang Feng, and Jie Zhou. 2025. FineType: Fine-grained Tapping Gesture Recognition for Text Entry. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems, CHI 2025, YokohamaJapan, 26 April 2025- 1 May 2025*, Naomi Yamashita, Vanessa Evers, Koji Yatani, Sharon Xianghua Ding, Bongshin Lee, Marshini Chetty, and Phoebe O. Toups Dugas (Eds.). ACM, 1083:1–1083:20. doi:10.1145/3706598.3714278

[33] Ke Li, Ruidong Zhang, Bo Liang, François Guimbretière, and Cheng Zhang. 2022. Eario: A low-power acoustic sensing earable for continuously tracking detailed facial movements. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 6, 2 (2022), 1–24. doi:10.1145/3534621

[34] Zhaofeng Lin, Tanvina Patel, and Odette Scharenborg. 2023. Improving whispered speech recognition performance using pseudo-whispered based data augmentation. In *2023 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 1–8. doi:10.1109/ASRU57964.2023.10389801

[35] Kang Ling, Haipeng Dai, Yuntang Liu, Alex X Liu, Wei Wang, and Qing Gu. 2020. Ultragesture: Fine-grained gesture sensing and recognition. *IEEE Transactions on Mobile Computing* 21, 7 (2020), 2620–2636. doi:10.1109/SAHCN.2018.8397099

[36] Pedro Lopes, Ricardo Jota, and Joaquim A. Jorge. 2011. Augmenting touch interaction through acoustic sensing. In *ACM International Conference on Interactive Tabletops and Surfaces, ITS 2011, Kobe, Japan, November 13-16, 2011*, Jun Rekimoto, Hideki Koike, Kentaro Fukuchi, Yoshifumi Kitamura, and Daniel Wigdor (Eds.). ACM, 53–56. doi:10.1145/2076354.2076364

[37] Ilya Loshchilov and Frank Hutter. 2016. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983* (2016). doi:10.48550/arXiv.1608.03983

[38] Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017). doi:10.48550/arXiv.1711.05101

[39] Yu Lu, Dian Ding, Hao Pan, Yijie Li, Juntao Zhou, Yongjian Fu, Yongzhao Zhang, Yi-Chao Chen, and Guangtao Xue. 2024. HandPad: Make Your Hand an On-the-go Writing Pad via Human Capacitance. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*. ACM, Pittsburgh PA USA, 1–16. doi:10.1145/3654777.3676328

[40] I Scott MacKenzie and Shawn X Zhang. 1999. The design and evaluation of a high-performance soft keyboard. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. 25–31. doi:10.1145/302979.302983

[41] Anand Mishra, Karteek Alahari, and Cv Jawahar. 2012. Scene Text Recognition using Higher Order Language Priors. In *Proceedings of the British Machine Vision Conference*. BMVA Press, 127.1–127.11. doi:10.5244/C.26.127

[42] Rajalakshmi Nandakumar, Vikram Iyer, Desney Tan, and Shyamnath Gollakota. 2016. FingerIO: Using Active Sonar for Fine-grained Finger Tracking. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*. 1515–1525. doi:10.1145/2858036.2858580

[43] Masa Ogata, Yuta Sugiura, Yasutoshi Makino, Masahiko Inami, and Michita Imai. 2013. SenSkin: adapting skin as a soft interface. In *Proceedings of the 26th annual ACM symposium on User interface software and technology*. ACM, St. Andrews Scotland, United Kingdom, 539–544. doi:10.1145/2501988.2502039

[44] Hao Pan, Yi-Chao Chen, Qi Ye, and Guangtao Xue. 2021. MagicInput: Training-free Multi-lingual Finger Input System using Data Augmentation based on MNISTs. In *Proceedings of the 20th International Conference on Information Processing in Sensor Networks (Co-Located with CPS-IoT Week 2021)* (Nashville, TN, USA) *(IPSN '21)*. Association for Computing Machinery, New York, NY, USA, 119–131. doi:10.1145/3412382.3458261

[45] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. 2019. SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition. (2019), 2613–2617. doi:10.21437/INTERSPEECH.2019-2680

[46] Karol J. Piczak. 2015. ESC: Dataset for Environmental Sound Classification. In *Proceedings of the 23rd Annual ACM Conference on Multimedia* (Brisbane, Australia, 2015-10-13). ACM Press, 1015–1018. doi:10.1145/2733373.2806390

[47] Jay Prakash, Zhijian Yang, Yu-Lin Wei, Haitham Hassanieh, and Romit Roy Choudhury. 2020. EarSense: earphones as a teeth activity sensor. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking*. 1–13. doi:10.1145/3372224.3419197

[48] Mirco Ravanelli and Yoshua Bengio. 2018. Speaker recognition from raw waveform with sincnet. In *2018 IEEE spoken language technology workshop (SLT)*. IEEE, 1021–1028. doi:10.1109/SLT.2018.8639585

[49] Jaemin Shin, Seungjoo Lee, Taesik Gong, Hyungjun Yoon, Hyunchul Roh, Andrea Bianchi, and Sung-Ju Lee. 2022. MyDJ: Sensing Food Intakes with an Attachable on Your Eyeglass Frame. In *CHI Conference on Human Factors in Computing Systems*. ACM, New Orleans LA USA, 1–17. doi:10.1145/3491102.3502041

[50] Katie A. Siek, Yvonne Rogers, and Kay H. Connelly. 2005. Fat Finger Worries: How Older and Younger Users Physically Interact with PDAs. In *Human-Computer Interaction - INTERACT 2005, IFIP TC13 International Conference, Rome, Italy, September 12-16, 2005, Proceedings (Lecture Notes in Computer Science, Vol. 3585)*, Maria Francesca Costabile and Fabio Paternò (Eds.). Springer, 267–280.

[51] Xingzhe Song, Kai Huang, and Wei Gao. 2022. Facelistener: Recognizing human facial expressions via acoustic sensing on commodity headphones. In *2022 21st ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. IEEE, 145–157. doi:10.1109/IPSN54338.2022.00019

[52] Tao Sun, Yankai Zhao, Wentao Xie, Jiao Li, Yongyu Ma, and Jin Zhang. 2024. EyeGesener: Eye gesture listener for smart glasses interaction using acoustic sensing. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 8, 3 (2024), 1–28. doi:10.1145/3678541

[53] Joachim Thiemann, Nobutaka Ito, and Emmanuel Vincent. 2013. The diverse environments multi-channel acoustic noise database (demand): A database of multichannel environmental noise recordings. In *Proceedings of Meetings on Acoustics*, Vol. 19. Acoustical Society of America, 035081. doi:10.1121/1.4799597

[54] Keith Vertanen, Haythem Memmi, Justin Emge, Shyam Reyal, and Per Ola Kristensson. 2015. VelociTap: Investigating Fast Mobile Text Entry using Sentence-Based Decoding of Touchscreen Keyboard Input. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, CHI 2015, Seoul, Republic of Korea, April 18-23, 2015*. ACM, 659–668. doi:10.1145/2702123.2702135

[55] Junjue Wang, Kaichen Zhao, Xinyu Zhang, and Chunyi Peng. 2014. Ubiquitous keyboard for small mobile devices: harnessing multipath fading for fine-grained keystroke localization. In *The 12th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys'14, Bretton Woods, NH, USA, June 16-19, 2014*, Andrew T. Campbell, David Kotz, Landon P. Cox, and Zhuoqing Morley Mao (Eds.). ACM, 14–27. doi:10.1145/2594368.2594384

[56] Wei Wang, Alex X. Liu, and Ke Sun. 2016. Device-free gesture tracking using acoustic signals. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking, MobiCom 2016, New York City, NY, USA, October 3-7, 2016*. ACM, 82–94. doi:10.1145/2973750.2973764

[57] Yanwen Wang, Jiaxing Shen, and Yuanqing Zheng. 2020. Push the Limit of Acoustic Gesture Recognition. In *39th IEEE Conference on Computer Communications, INFOCOM 2020, Toronto, ON, Canada, July 6-9, 2020*. IEEE, 566–575. doi:10.1109/INFOCOM41043.2020.9155402

[58] Martin Weigel, Vikram Mehta, and Jürgen Steimle. 2014. More than touch: understanding how people use skin as an input surface for mobile computing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Toronto, Ontario, Canada) *(CHI '14)*. Association for Computing Machinery, New York, NY, USA, 179–188. doi:10.1145/2556288.2557239

[59] Hongyi Wen, Julian Ramos Rojas, and Anind K. Dey. 2016. Serendipity: Finger Gesture Recognition using an Off-the-Shelf Smartwatch. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, San Jose, CA, USA, May 7-12, 2016*, Jofish Kaye, Allison Druin, Cliff Lampe, Dan Morris, and Juan Pablo Hourcade (Eds.). ACM, 3847–3851. doi:10.1145/2858036.2858466

[60] Yueting Weng, Chun Yu, Yingtian Shi, Yuhang Zhao, Yukang Yan, and Yuanchun Shi. 2021. FaceSight: Enabling Hand-to-Face Gesture Interaction on AR Glasses with a Downward-Facing Camera Vision. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems (CHI '21)*. Association for Computing Machinery, New York, NY, USA, 1–14. doi:10.1145/3411764.3445484

[61] Haijun Xia, Tovi Grossman, and George Fitzmaurice. 2015. NanoStylus: Enhancing input on ultra-small displays with a finger-mounted stylus. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology (UIST '15)*. 447–456. doi:10.1145/2807442.2807500

[62] Robert Xiao, Greg Lew, James Marsanico, Divya Hariharan, Scott E. Hudson, and Chris Harrison. 2014. Toffee: enabling ad hoc, around-device interaction with acoustic time-of-arrival correlation. In *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services, MobileHCI 2014, Toronto, ON, Canada, September 23-26, 2014*, Aaron J. Quigley, Sara Diamond, Pourang Irani, and Sriram Subramanian (Eds.). ACM, 67–76. doi:10.1145/2628363.2628383

[63] Wentao Xie, Huangxun Chen, Jing Wei, Jin Zhang, and Qian Zhang. 2024. RimSense: Enabling Touch-based Interaction on Eyeglass Rim Using Piezoelectric Sensors. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 7, 4, Article 191 (Jan. 2024), 24 pages. doi:10.1145/3631456

[64] Wentao Xie, Qian Zhang, and Jin Zhang. 2021. Acoustic-based Upper Facial Action Recognition for Smart Eyewear. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 5, 2 (2021), 41:1–41:28. doi:10.1145/3448105

[65] Chao Xu, Parth H. Pathak, and Prasant Mohapatra. 2015. Finger-writing with Smartwatch: A Case for Finger and Hand Gesture Recognition using Smartwatch. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications, HotMobile 2015, Santa Fe, NM, USA, February 12-13, 2015*, Justin Manweiler and Romit Roy Choudhury (Eds.). ACM, 9–14. doi:10.1145/2699343.2699350

[66] Chenhan Xu, Bing Zhou, Gurunandan Krishnan, and Shree K. Nayar. 2023. AO-Finger: Hands-free Fine-grained Finger Gesture Recognition via Acoustic-Optic Sensor Fusing. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, CHI 2023, Hamburg, Germany, April 23-28, 2023*, Albrecht Schmidt, Kaisa Väänänen, Tesh Goyal, Per Ola Kristensson, Anicia Peters, Stefanie Mueller, Julie R. Williamson, and Max L. Wilson (Eds.). ACM, 306:1–306:14. doi:10.1145/3544548.3581264

[67] Xuhai Xu, Haitian Shi, Xin Yi, WenJia Liu, Yukang Yan, Yuanchun Shi, Alex Mariakakis, Jennifer Mankoff, and Anind K. Dey. 2020. EarBuddy: Enabling On-Face Interaction via Wireless Earbuds. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20)*. Association for Computing Machinery, New York, NY, USA, 1–14. doi:10.1145/3313831.3376836

[68] Tomer Yanay and Erez Shmueli. 2020. Air-writing recognition using smart-bands. *Pervasive Mob. Comput.* 66 (2020), 101183. doi:10.1016/J.PMCJ.2020.101183

[69] Xingen Yu, Zhaoqian Xie, Yang Yu, Jungyup Lee, A. Vázquez-Guardado, H. Luan, Jasper Ruban, Xin Ning, Aadeel Akhtar, Dengfeng Li, Bowen Ji, Yiming Liu, Rujie Sun, Jingyue Cao, Q. Huo, Yishan Zhong, C. Lee, SeungYeop Kim, P. Gutruf, Changxing Zhang, Yeguang Xue, Qinglei Guo, Aditya Chempakasseril, Peilin Tian, W. Lu, J. Jeong, Yongjoon Yu, Jesse Cornman, CheeSim Tan, B. Kim, Kunhyuk Lee, Xueting Feng, Yonggang Huang, and J. Rogers. 2019. Skin-integrated wireless haptic interfaces for virtual and augmented reality. *Nature* 575 (2019), 473 – 479. doi:10.1038/s41586-019-1687-0

[70] Sangki Yun, Yi-Chao Chen, Huihuang Zheng, Lili Qiu, and Wenguang Mao. 2017. Strata: Fine-grained acoustic-based device-free tracking. In *Proceedings of the 15th annual international conference on mobile systems, applications, and services (MobiSys '17)*. 15–28. doi:10.1145/3081333.3081356

[71] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. 2019. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*. 6023–6032. doi:10.1109/ICCV.2019.00612

[72] Neil Zeghidour, Olivier Teboul, Félix De Chaumont Quitry, and Marco Tagliasacchi. 2021. LEAF: A learnable frontend for audio classification. *arXiv preprint arXiv:2101.08596* (2021). doi:10.48550/arXiv.2101.08596

[73] Cheng Zhang, AbdelKareem Bedri, Gabriel Reyes, Bailey Bercik, Omer T. Inan, Thad E. Starner, and Gregory D. Abowd. 2016. TapSkin: Recognizing On-Skin Input for Smartwatches. In *Proceedings of the 2016 ACM International Conference on Interactive Surfaces and Spaces*. ACM, Niagara Falls Ontario Canada, 13–22. doi:10.1145/2992154.2992187

[74] Cheng Zhang, Anandghan Waghmare, Pranav Kundra, Yiming Pu, Scott M. Gilliland, Thomas Ploetz, Thad E. Starner, Omer T. Inan, and Gregory D. Abowd. 2017. FingerSound: Recognizing unistroke thumb gestures using a ring. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 3 (2017), 120:1–120:19. doi:10.1145/3130985

[75] Cheng Zhang, Qiuyue Xue, Anandghan Waghmare, Sumeet Jain, Yiming Pu, Sinan Hersek, Kent Lyons, Kenneth A. Cunefare, Omer T. Inan, and Gregory D. Abowd. 2017. SoundTrak: Continuous 3D Tracking of a Finger Using Active Acoustics. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 2 (2017), 30:1–30:25. doi:10.1145/3090095

[76] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. 2018. mixup: Beyond Empirical Risk Minimization. (2018). https://openreview.net/forum?id=r1Ddp1-Rb

[77] Yang Zhang, Junhan Zhou, Gierad Laput, and Chris Harrison. 2016. SkinTrack: Using the Body as an Electrical Waveguide for Continuous Finger Tracking on the Skin. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, San Jose, CA, USA, May 7-12, 2016*, Jofish Kaye, Allison Druin, Cliff Lampe, Dan Morris, and Juan Pablo Hourcade (Eds.). ACM, 1491–1503. doi:10.1145/2858036.2858082

[78] Zeyu Zhang, Hua Huang, Penghao Dong, Shanshan Yao, and Shan Lin. 2025. MagSkin: Enabling Touchless Human-Computer Interaction with Soft Epidermal Magnetic Material. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 9, 4 (Dec. 2025), 240:1–240:30. doi:10.1145/3770675

[79] Yankai Zhao, Wentao Xie, Haorui Li, Jiao Li, Tao Sun, Qian Zhang, and Jin Zhang. 2026. FingerBar: A Mid-Air Touch Bar Interface for Earphones Using Finger-Generated Acoustics. In *Proceedings of the 2026 CHI Conference on Human Factors in Computing Systems (CHI '26)*. Association for Computing Machinery, New York, NY, USA, Article 736, 22 pages. doi:10.1145/3772318.3790462

[80] Yongpan Zou, Qiang Yang, Rukhsana Ruby, Yetong Han, Sicheng Wu, Mo Li, and Kaishun Wu. 2019. EchoWrite: An Acoustic-based Finger Input System Without Training. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. 778–787. doi:10.1109/ICDCS.2019.00082 ISSN: 2575-8411.

Appendix

A USER FEEDBACK ON EACH LETTER AND GESTURE

We summarize participants' ratings of comfort and difficulty for each letter and gesture in Figure 21, where higher scores indicate more positive perceptions. All letters score above the neutral midpoint (3). However, letters with tails (e.g., *j*, *q*, *g*) and long vertical strokes (e.g., *h*, *d*, *k*) are rated less favorably. Participants attribute this to fingernail contact and uneven skin on the back of the hand: *"Writing letters with long tails can be hard for users with longer nails..."* (P4) and *"...the skin on the back of my hand is...uneven. So I prefer writing smaller letters"* (P5). We expect these issues to be mitigated with clearer guidelines. For example, we can encourage the user to position their finger more parallel to the skin to reduce nail contact. Additionally, we may instruct users to clench their palms into fists as much as possible, rather than spreading them, to reduce the unevenness on the palmar aspect of the hand caused by the bones.

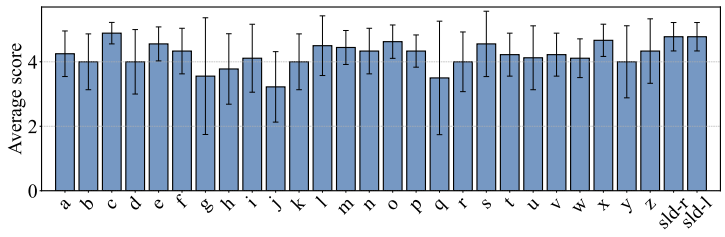


Fig. 21. Users' evaluation of letters.